

Cribl Search Documentation Manual

Version: v4.15

Generated: 2025-12-03 09:53:25

Copyright (c) 2017- 2025 Cribl, Inc. All rights reserved.

1. About Cribl Search 15
1.1. Basic Concepts 18
1.2. Quick Setup 20
1.3. Cribl Search Tour 26
2. Connect Cribl Search to Your Data 41
2.1. Connect Cribl Search to Cribl Lake 43
2.2. Connect Cribl Search to Amazon S3 47
2.3. Connect Cribl Search to Cribl Edge 57
2.4. Connect Cribl Search to AWS 62
2.4.1. Grant Access to AWS 63
2.4.2. Connect Cribl Search to Amazon Security Lake 71
2.4.3. Connect Cribl Search to AWS API 78
2.4.4. Connect Cribl Search to Data Lake Amazon S3 82
2.5. Connect Cribl Search to Azure 90
2.5.1. Connect Cribl Search to Azure Blob Storage 91
2.5.2. Connect Cribl Search to Azure Data Explorer 100
2.5.3. Connect Cribl Search to Azure API 102
2.6. Connect Cribl Search to Google 105
2.6.1. Connect Cribl Search to Google Cloud Storage 106
2.6.2. Connect Cribl Search to Google Cloud Platform API 113
2.6.3. Connect Cribl Search to Google Workspace API 115
2.7. Connect Cribl Search to ClickHouse 118
2.8. Connect Cribl Search to Elasticsearch 122
2.9. Connect Cribl Search to a Generic HTTP API 124
2.10. Connect Cribl Search to Microsoft Graph API 129
2.11. Connect Cribl Search to Okta API 134
2.12. Connect Cribl Search to OpenSearch 136
2.13. Connect Cribl Search to Prometheus 138
2.14. Connect Cribl Search to Snowflake 145
2.15. Connect Cribl Search to Tailscale API 148
2.16. Connect Cribl Search to Zoom API 150
2.17. Configure Data Connection Options in Cribl Search 152
2.17.1. Datatypes 153
2.17.2. Storage Classes 187
2.17.3. Supported Data Formats 190
3. Search Your Data 195

3.1. Build a Search 197
3.2. Write Your First Query 210
3.3. Write Queries Using Cribl Copilot 223
3.4. Common Query Examples 227
3.5. Use Search Results 237
3.6. Cribl Search Notebooks 241
3.7. Example Searches by Provider 250
3.7.1. Searching Cribl Lake 251
3.7.2. Searching Cribl Edge 254
3.7.3. Built-In Cribl Edge Datasets 258
3.7.4. Searching Amazon S3 264
3.7.5. Searching AWS API 268
3.7.6. Searching Okta API 271
3.7.7. Searching Zoom API 273
3.7.8. Searching Multiple Providers 275
4. Visualize Search Results 280
4.1. Charts 282
4.1.1. Types of Charts in Cribl Search 285
4.1.1.1. Area Chart 286
4.1.1.2. Column Chart 288
4.1.1.3. Donut Chart 291
4.1.1.4. Funnel Chart 293
4.1.1.5. Gauge Chart 295
4.1.1.6. Horizontal Bar Chart 296
4.1.1.7. Line Chart 299
4.1.1.8. Map Chart 301
4.1.1.9. Scatter Chart 305
4.1.1.10. Single Value Chart 307
4.1.2. Results Tables 309
4.2. Dashboards 314
4.2.1. Create a Dashboard 316
4.2.2. Edit a Dashboard 319
4.2.3. Add Interactions to Your Cribl Search Dashboard 327
4.2.4. Add Inputs to Your Cribl Search Dashboard 339
4.2.5. Add Visualizations Using Cribl Copilot 346
4.2.6. Manage and Share Dashboards 352
4.3. Third-Party Visualization Tools 353
4.3.1. Use Cribl Search with Grafana 353

5. Send Alerts With Cribl Search 358
5.1. Scheduled Searches 359
5.2. Notifications 362
5.2.1. Email Notifications 367
5.2.2. AWS SNS Notifications 373
5.2.3. PagerDuty Notifications 378
5.2.4. Slack Notifications 381
5.2.5. Webhook Notifications 384
5.2.6. Notifications via Cribl Stream 392
6. Enrich and Automate 397
6.1. Lookups 398
6.2. Parsers 402
6.3. Regexes 405
6.4. Grok Patterns 407
6.5. Macros 408
6.6. Cribl Search Packs 414
7. Manage Cribl Search and Cribl.Cloud 430
7.1. Cribl.Cloud Access and Usage 432
7.1.1. Manage Cribl.Cloud Organizations 433
7.1.2. Manage Billing in FinOps Center 435
7.2. Workspaces 441
7.2.1. Configure Workspaces 444
7.2.2. Configure Workspace Teams 447
7.3. Usage Settings 449
7.3.1. Usage Groups 451
7.3.2. Limits 454
7.4. Sharing Search Resources 456
8. Optimize Cribl Search 460
8.1. Optimize Searches 461
8.2. Optimize Paths 467
8.3. Investigative Searching 473
8.4. Productivity Tips 477
8.5. Useful Queries (Analytic Cookbook) 480
9. Troubleshooting Cribl Search 490
9.1. Known Issues 491
9.2. KQL Extensions 492
9.3. Lakehouse Search Differences 493
9.4. Search Details 495

9.5. Share Diagnostics 500
9.6. Common Issues and Resolutions 501
10. Language Reference 506
10.1. Language Reference Index 509
10.2. Operators in Cribl Search 523
10.2.1. Aggregation Operators 530
10.2.1.1. count 531
10.2.1.2. eventstats 533
10.2.1.3. summarize 536
10.2.1.4. timestats 540
10.2.2. Data Operators 543
10.2.2.1. centralize 544
10.2.2.2. export 546
10.2.2.3. extend 551
10.2.2.4. externaldata 553
10.2.2.5. extract 556
10.2.2.6. foldkeys 561
10.2.2.7. ip-lookup 567
10.2.2.8. join 570
10.2.2.9. lookup 575
10.2.2.10. mv-expand 579
10.2.2.11. mv-pull 582
10.2.2.12. pivot 585
10.2.2.13. send 587
10.2.2.14. top-hitters 591
10.2.2.15. union 593
10.2.3. Display Operators 595
10.2.3.1. limit 596
10.2.3.2. order 598
10.2.3.3. print 600
10.2.3.4. project 602
10.2.3.5. project-away 605
10.2.3.6. project-rename 607
10.2.3.7. range 609
10.2.3.8. render 611
10.2.3.9. sort 613
10.2.3.10. take 615
10.2.3.11. top 616

10.2.4. Filter Operators 618
10.2.4.1. dedup 619
10.2.4.2. distinct 621
10.2.4.3. search 623
10.2.4.4. where 627
10.2.5. Logical Operators 634
10.2.6. Numerical Operators 635
10.2.7. Search Operators 637
10.2.7.1. cribl 638
10.2.7.2. find 646
10.2.8. String Operators 651
10.2.8.1. == 654
10.2.8.2. != 656
10.2.8.3. =~ 658
10.2.8.4. !~ 660
10.2.8.5. contains 662
10.2.8.6. !contains 664
10.2.8.7. contains_cs 666
10.2.8.8. !contains_cs 668
10.2.8.9. endswith 670
10.2.8.10. !endswith 672
10.2.8.11. endswith_cs 674
10.2.8.12. !endswith_cs 676
10.2.8.13. has 678
10.2.8.14. !has 680
10.2.8.15. has_cs 682
10.2.8.16. !has_cs 684
10.2.8.17. has_all 686
10.2.8.18. !has_all 687
10.2.8.19. has_any 688
10.2.8.20. !has_any 689
10.2.8.21. hasprefix 690
10.2.8.22. !hasprefix 692
10.2.8.23. hasprefix_cs 694
10.2.8.24. !hasprefix_cs 696
10.2.8.25. hassuffix 698
10.2.8.26. !hassuffix 700
10.2.8.27. hassuffix_cs 702

10.2.8.28. !hassuffix_cs 704
10.2.8.29. in 706
10.2.8.30. !in 708
10.2.8.31. in~ 710
10.2.8.32. !in~ 712
10.2.8.33. matches regex 714
10.2.8.34. startswith 716
10.2.8.35. !startswith 718
10.2.8.36. startswith_cs 720
10.2.8.37. !startswith_cs 722
10.3. Commands 724
10.3.1. .cancel 728
10.3.2. .show 731
10.3.3. .clear 732
10.3.4. .generate 733
10.3.5. objects 734
10.3.6. options 737
10.3.7. stats 741
10.3.8. queries 743
10.4. Statements 745
10.4.1. let 746
10.4.2. set 749
10.5. Functions 761
10.5.1. Context Functions 774
10.5.2. Cribl Functions 776
10.5.2.1. findearliest 778
10.5.2.2. findearliestif 779
10.5.2.3. findfirst 780
10.5.2.4. findfirstif 782
10.5.2.5. findlast 783
10.5.2.6. findlastif 785
10.5.2.7. findlatest 786
10.5.2.8. findlatestif 787
10.5.2.9. list 788
10.5.2.10. median 789
10.5.2.11. medianif 790
10.5.2.12. persecond 791
10.5.2.13. persecondif 792

10.5.2.14. rate 793
10.5.2.15. rateif 794
10.5.2.16. sumsq 795
10.5.2.17. sumsqif 796
10.5.2.18. values 797
10.5.3. Scalar Functions 798
10.5.3.1. Binary Functions 808
10.5.3.1.1. binary_and 809
10.5.3.1.2. binary_not 810
10.5.3.1.3. binary_or 811
10.5.3.1.4. binary_shift_left 812
10.5.3.1.5. binary_shift_right 813
10.5.3.1.6. binary_xor 814
10.5.3.1.7. from_binary_string 815
10.5.3.1.8. to_binary_string 816
10.5.3.2. Conditional Functions 817
10.5.3.2.1. case 818
10.5.3.2.2. coalesce 820
10.5.3.2.3. iff 821
10.5.3.2.4. iif 822
10.5.3.2.5. max_of 823
10.5.3.2.6. min_of 824
10.5.3.3. Conversion Functions 825
10.5.3.3.1. bin 826
10.5.3.3.2. bin_auto 827
10.5.3.3.3. floor 829
10.5.3.3.4. gettype 831
10.5.3.3.5. tobool 833
10.5.3.3.6. todecimal 835
10.5.3.3.7. todouble 836
10.5.3.3.8. toint 837
10.5.3.3.9. tolong 838
10.5.3.3.10. toreal 839
10.5.3.3.11. tostring 840
10.5.3.4. DateTime Functions 841
10.5.3.4.1. ago 844
10.5.3.4.2. datetime_add 846
10.5.3.4.3. datetime_diff 848

10.5.3.4.4. datetime_part 850
10.5.3.4.5. dayofmonth 853
10.5.3.4.6. dayofweek 854
10.5.3.4.7. dayofyear 855
10.5.3.4.8. endofday 856
10.5.3.4.9. endofmonth 857
10.5.3.4.10. endofweek 858
10.5.3.4.11. endofyear 859
10.5.3.4.12. format_datetime 860
10.5.3.4.13. format_timespan 864
10.5.3.4.14. getmonth 868
10.5.3.4.15. getyear 869
10.5.3.4.16. hourofday 870
10.5.3.4.17. make_datetime 871
10.5.3.4.18. make_timespan 872
10.5.3.4.19. monthofyear 874
10.5.3.4.20. now 875
10.5.3.4.21. startofday 877
10.5.3.4.22. startofmonth 878
10.5.3.4.23. startofweek 879
10.5.3.4.24. startofyear 880
10.5.3.4.25. strftime 881
10.5.3.4.26. strptime 883
10.5.3.4.27. todatetime 885
10.5.3.4.28. totimespan 886
10.5.3.4.29. unixtime_microseconds_todatetime 887
10.5.3.4.30. unixtime_milliseconds_todatetime 888
10.5.3.4.31. unixtime_nanoseconds_todatetime 889
10.5.3.4.32. unixtime_seconds_todatetime 890
10.5.3.4.33. week_of_year 891
10.5.3.5. Dynamic Functions 892
10.5.3.5.1. bag_has_key 894
10.5.3.5.2. bag_keys 895
10.5.3.5.3. bag_merge 896
10.5.3.5.4. bag_pack 898
10.5.3.5.5. bag_pack_columns 900
10.5.3.5.6. bag_remove_keys 902
10.5.3.5.7. bag_set_key 904

10.5.3.5.8. bag_zip 907
10.5.3.5.9. make_bag 909
10.5.3.5.10. make_bag_if 912
10.5.3.5.11. zip 915
10.5.3.6. Cryptographic Functions 917
10.5.3.6.1. encrypt 918
10.5.3.6.2. decrypt 920
10.5.3.7. Hash Functions 922
10.5.3.7.1. hash 923
10.5.3.7.2. hash_combine 925
10.5.3.7.3. hash_many 926
10.5.3.7.4. hash_md5 927
10.5.3.7.5. hash_sha1 929
10.5.3.7.6. hash_sha256 931
10.5.3.7.7. hash_xxhash64 933
10.5.3.8. INET Functions 934
10.5.3.8.1. ipv4_compare 935
10.5.3.8.2. ipv4_is_in_range 937
10.5.3.8.3. ipv4_is_in_any_range 939
10.5.3.8.4. ipv4_is_match 941
10.5.3.8.5. ipv4_is_private 943
10.5.3.8.6. ipv4_netmask_suffix 945
10.5.3.8.7. ipv6_compare 947
10.5.3.8.8. ipv6_is_match 949
10.5.3.8.9. format_bytes 951
10.5.3.8.10. format_ipv4 953
10.5.3.8.11. format_ipv4_mask 954
10.5.3.9. Mathematical Functions 955
10.5.3.9.1. abs 957
10.5.3.9.2. acos 958
10.5.3.9.3. asin 959
10.5.3.9.4. atan 960
10.5.3.9.5. atan2 961
10.5.3.9.6. beta_cdf 962
10.5.3.9.7. beta_inv 964
10.5.3.9.8. beta_pdf 966
10.5.3.9.9. ceil 968
10.5.3.9.10. ceiling 969

10.5.3.9.11. cos 970
10.5.3.9.12. cot 971
10.5.3.9.13. degrees 972
10.5.3.9.14. exp 973
10.5.3.9.15. exp2 974
10.5.3.9.16. exp10 975
10.5.3.9.17. gamma 976
10.5.3.9.18. isfinite 977
10.5.3.9.19. isinf 978
10.5.3.9.20. isnan 980
10.5.3.9.21. log 981
10.5.3.9.22. log2 982
10.5.3.9.23. log10 983
10.5.3.9.24. loggamma 984
10.5.3.9.25. not 985
10.5.3.9.26. pi 987
10.5.3.9.27. pow 988
10.5.3.9.28. radians 989
10.5.3.9.29. rand 990
10.5.3.9.30. range 991
10.5.3.9.31. round 993
10.5.3.9.32. sign 995
10.5.3.9.33. sin 996
10.5.3.9.34. sqrt 997
10.5.3.9.35. tan 998
10.5.3.10. String Functions 999
10.5.3.10.1. base64_decode_toarray 1002
10.5.3.10.2. base64_decode_tostring 1003
10.5.3.10.3. base64_encode_fromarray 1004
10.5.3.10.4. base64_encode_tostring 1005
10.5.3.10.5. countof 1006
10.5.3.10.6. extract 1009
10.5.3.10.7. extract_all 1011
10.5.3.10.8. extract_json 1014
10.5.3.10.9. has_any_index 1016
10.5.3.10.10. indexof 1018
10.5.3.10.11. isempty 1021
10.5.3.10.12. isnotempty 1023

10.5.3.10.13. isnotnull 1025
10.5.3.10.14. isnull 1026
10.5.3.10.15. match_regex 1027
10.5.3.10.16. parse_csv 1029
10.5.3.10.17. parse_ipv4 1031
10.5.3.10.18. parse_ipv4_mask 1033
10.5.3.10.19. parse_ipv6 1035
10.5.3.10.20. parse_ipv6_mask 1037
10.5.3.10.21. parse_json 1039
10.5.3.10.22. parse_url 1041
10.5.3.10.23. parse_urlquery 1043
10.5.3.10.24. parse_version 1045
10.5.3.10.25. replace_regex 1046
10.5.3.10.26. reverse 1048
10.5.3.10.27. split 1049
10.5.3.10.28. strcat 1051
10.5.3.10.29. strcat_delim 1053
10.5.3.10.30. strcmp 1055
10.5.3.10.31. strlen 1057
10.5.3.10.32. strrep 1058
10.5.3.10.33. substring 1060
10.5.3.10.34. tolower 1062
10.5.3.10.35. toupper 1063
10.5.3.10.36. translate 1064
10.5.3.10.37. trim 1066
10.5.3.10.38. trim_end 1068
10.5.3.10.39. trim_start 1069
10.5.3.10.40. url_decode 1070
10.5.3.10.41. url_encode 1071
10.5.4. Statistical Functions 1072
10.5.4.1. avg 1074
10.5.4.2. avgif 1076
10.5.4.3. count 1078
10.5.4.4. countif 1079
10.5.4.5. dcount 1080
10.5.4.6. dcountif 1083
10.5.4.7. max 1086
10.5.4.8. maxif 1087

10.5.4.9. min 1088
10.5.4.10. minif 1089
10.5.4.11. percentile 1091
10.5.4.12. stdev 1093
10.5.4.13. stdevif 1095
10.5.4.14. stdevp 1096
10.5.4.15. sum 1098
10.5.4.16. sumif 1099
10.5.4.17. variance 1100
10.5.4.18. varianceif 1101
10.5.4.19. variancep 1102
10.5.5. Window Functions 1103
10.5.5.1. next 1104
10.5.5.2. prev 1106
10.5.5.3. row_cumsum 1108
10.5.5.4. row_number 1110
10.5.5.5. row_rank_dense 1111
10.5.5.6. row_rank_min 1113
10.5.5.7. row_window_session 1115
10.6. Virtual Tables 1117
10.6.1. $vt_datasets 1118
10.6.2. $vt_dataset_providers 1120
10.6.3. $vt_dummy 1122
10.6.4. $vt_jobs 1125
10.6.5. $vt_list 1127
10.6.6. $vt_lookups 1129
10.6.7. $vt_results 1131
10.7. Types 1135
10.7.1. bool 1137
10.7.2. datetime 1141
10.7.3. decimal 1142
10.7.4. double 1143
10.7.5. dynamic 1144
10.7.6. int 1148
10.7.7. long 1149
10.7.8. null 1150
10.7.9. real 1154
10.7.10. string 1155

10.7.11. timespan 1157
10.8. Regex Matching 1159

1. About Cribl Search

Get a a quick walk-through of Cribl Search capabilities, compatibility, and usage.

Cribl Search helps you explore data in place, right where it lives. You can query logs, metrics, or any sort of machine data immediately, without first moving it to specialized storage. You can automate your searches, visualize and share the results, and get alerts about changing trends. Cribl Search offers native support for a wide array of data sources:

- Data lakes – including Cribl Lake, Amazon Security Lake, and Amazon S3 and S3-compatible lakes.
- Object stores – including Amazon S3, Azure Blob Storage, Google Cloud Storage, and more.
- Cloud data warehouses – including Snowflake and Clickhouse.
- Analytics services and platforms – including Azure Data Explorer, Elasticsearch, OpenSearch, Prometheus, and more.
- API endpoints – including Azure, AWS, Google Workspace, Okta, Zoom, and even a Generic HTTP API option that enables you to query any HTTP API.

Beyond Cribl Lake, Cribl Search seamlessly integrates with Cribl Stream and Cribl Edge.

Depending on the data provider you're searching, Cribl Search either handles all the data processing itself, or takes advantage of the data provider's native compute resources to offload and optimize processing.

For the full list of natively supported data sources, see: List of Supported Data Sources.

What Can You Do with Cribl Search?

Cribl Search enables you to search, transform, and analyze your data, regardless of its original source or format.

Cribl Search is offered as a service via Cribl.Cloud. Your data can reside anywhere – in the public or private cloud, on-prem, etc. Cribl Search processes log events in the location where they're stored, requiring no indexes.

Who Is Cribl Search For?

Cribl Search is built for administrators, managers, and users of operational/DevOps and security intelligence products and services. This includes system administrators, data scientists, incident responders, and threat hunters.

How Does Cribl Search Work?

Cribl Search leverages the Kusto Query Language (KQL). Even if you are unfamiliar with Kusto, Cribl Search offers typeahead help in building your queries – and even an AI assistant that can translate your native-language queries to KQL syntax.

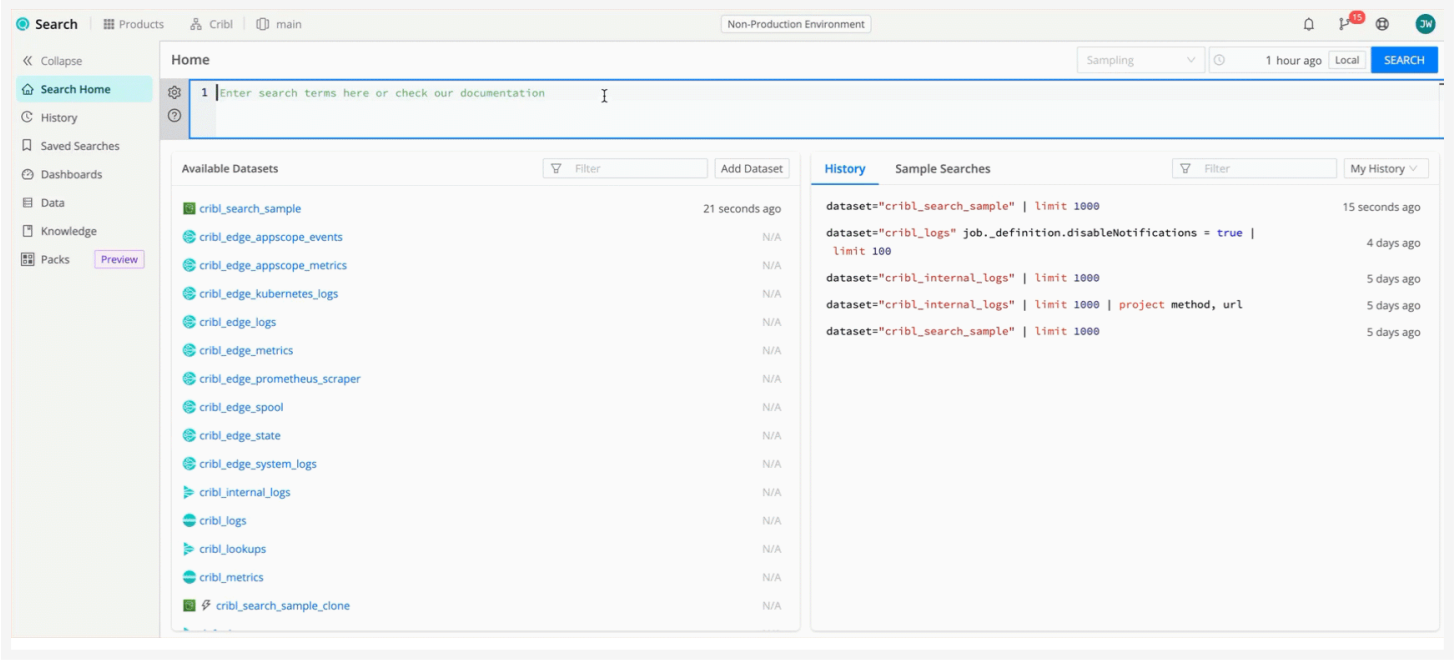
Cribl Search then queries data in place, using the specific language or technology associated with the data's storage location. By automatically converting your KQL query into the required target query language, Cribl Search spares users from needing to learn the query syntax of each underlying system.

Cribl Search dynamically spins up many parallel search executors to ensure that your search runs in the fastest wall-clock time possible. Cribl Search coordinates and federates the results streaming in from multiple systems, interprets the results, and presents a single, unified view of your result set. Cribl Search can extract structured field names and values even from unstructured source data.

Cribl Search in Action

Let's see a basic search. Try this yourself in Cribl Search, or just watch the animation:

1. Log in to Cribl Search as an Admin or Editor.
2. From the sidebar, select **Search Home**.
3. From the list of **Available Datasets**, select `cribl_search_sample` to place a basic query in the query box.
4. Press Enter/Return, or select the **Search** button.
5. Results for this search will appear on the **Events** tab, with a histogram.



Basic search

What’s Next?

Ready to put Cribl Search to work? See Basic Concepts.

1.1. Basic Concepts

Learn fundamental Cribl Search concepts, terminology, and usage relationships, with links to further information.

Datasets

Datasets are a way to organize and reference a set of data. Think of a Dataset as a definition that describes **what** we need to query, or as a "container" of some data.

A Dataset might contain information about the path(s) that need to be queried for specific data at a specific source, such as a filesystem, an S3 bucket, or an Edge Worker. For example, a Dataset named `myVPCFlowlogs` contains Amazon VPC Flow logs, and is referenced as `dataset=myVPCFlowlogs` in a query.

Data is broken into discrete events and searchable fields by assigning Datatypes to a Dataset.

Dataset Providers

Dataset Providers enable Cribl Search to categorize and connect Datasets using specific interfaces to external systems. Think of a Dataset Provider as a definition that describes **where** we need to send the query. A Dataset Provider can contain connection information required to obtain the data, such as API keys or secrets.

Searches

A search is a query request that Cribl Search uses to process data and return results. The request is a plain-text string composed of multiple operators separated by pipes: `|`. For example, the search `dataset=myVPCFlowlogs | limit 1000` will retrieve data from the Dataset `myVPCFlowlogs` and will return up to 1,000 results. For further details, see Cribl Search Tour.

Operators and Functions

As data passes through a search, it is handled by operators that process the data based on functions. For example, in this search `dataset=myVPCFlowlogs | summarize count() by srcport` the `summarize` operator will aggregate all events in the Dataset `myVPCFlowlogs`. The `count()` function will count the number of events, and the `by` clause will group them by `srcport` values.

Commands and Statements

Commands are simple instructions that you can type right in the query box. They're designed primarily for Admins and Editors, who use them to manage search jobs or learn more about the Organization's Datasets.

Commands always start with a period (`.`) as you can see in this example: `.show running queries`.

Statements help you build more readable and efficient searches. Use `let` statements to give names to expressions or entire queries, and then reference those names in the context of the same search, which in turn allows you to `join` Datasets. The `set` statement, on the other hand, lets you configure advanced search options.

Search Types

The search type is based on the type of results returned, either raw or aggregate. If the query runs the `summarize`, `eventstats`, or `timestats` operators along with an aggregation function, it returns tabular results that are chartable; otherwise, it returns raw results in a table.

Cribl and statistical are aggregation functions that are used in conjunction with the `summarize`, `eventstats`, and `timestats` operators to aggregate your data.

You can recognize aggregation functions and operators by this icon next to their titles:  .

Cribl Lake and Lakehouses

Cribl Lake is Cribl's storage solution for your data, requiring little to no configuration on your part. Lake Datasets work as Cribl Search Datasets out-of-the-box, so you can instantly start searching them. You can also assign Lake Datasets to Lakehouses, for dramatically faster search responses and more-predictable billing. For more information, see the Cribl Lake docs.

1.2. Quick Setup

Create a Cribl.Cloud Organization and run your first searches.

Sign Up for Cribl.Cloud

Your first step is to create your Cribl.Cloud Organization.

1. Start at: https://cribl.cloud/signup/.
2. Select the **New User? Free signup** option, and register with your work email address.
3. Use the verification code from Cribl's email to confirm your registration.
4. On the **Create Organization** page, optionally enter an **Organization Name** (a friendly alias for the randomly generated ID that Cribl will assign to your Organization).
5. Select an AWS Region to host your Cribl.Cloud Leader. Cribl currently supports the following Regions:
    - `US West (Oregon)`
    - `US East (Virginia)`
    - `US East (Ohio)`
    - `Europe (Frankfurt)`
    - `Europe (London)`
    - `Europe (Zurich)`
    - `Asia Pacific (Sydney)`
    - `Asia Pacific (Singapore)`
    - `Canada (Central)`

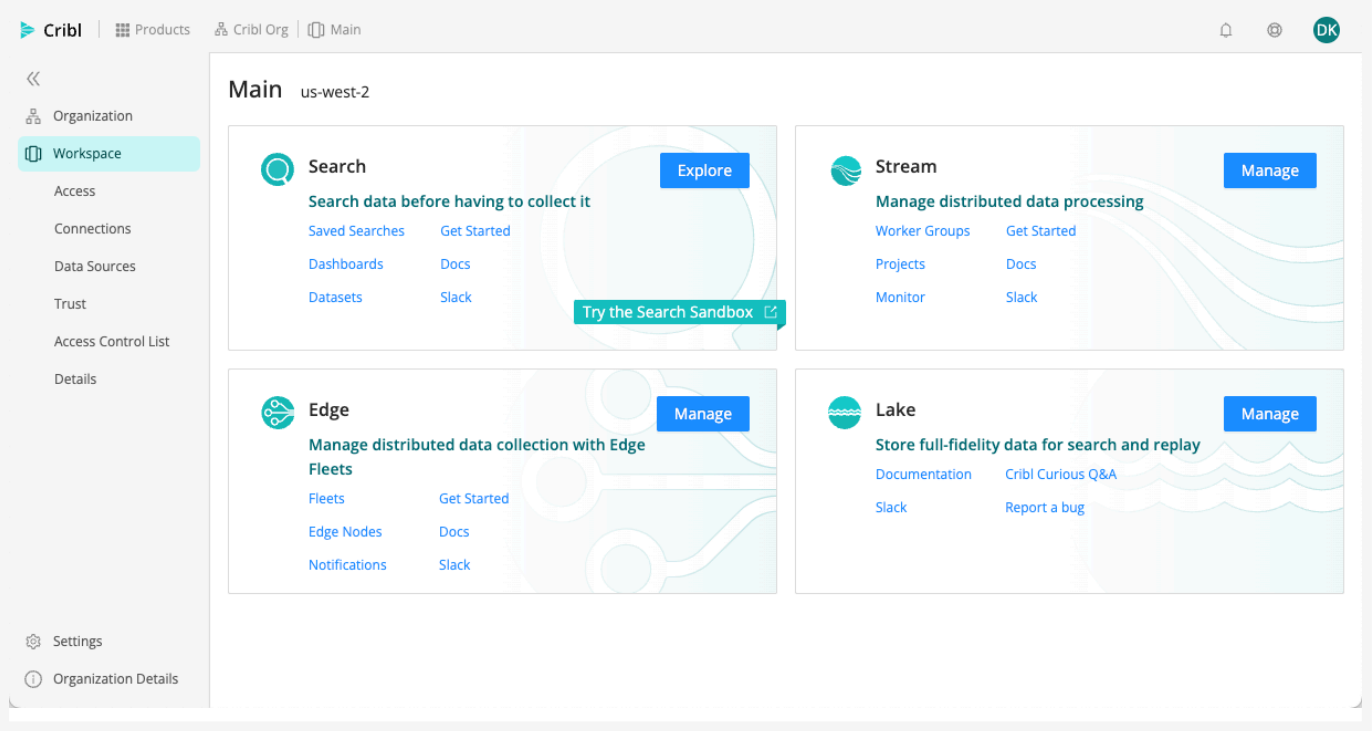
    The Organization region defines where the Leader Node resides.

    Once you select a region when registering your Organization, you cannot change it. You can, however, create Cribl-managed Stream Worker Groups in different AWS and Azure regions.

    In Cribl Search, the Organization region is the physical location where searches are executed. In Cribl Lake, the region defines the location where data is stored.
6. Select **Create Organization**.
7. On the resulting page labeled **We're spinning up your Organization...**, complete the quick survey to let us know how you plan to use Cribl products.

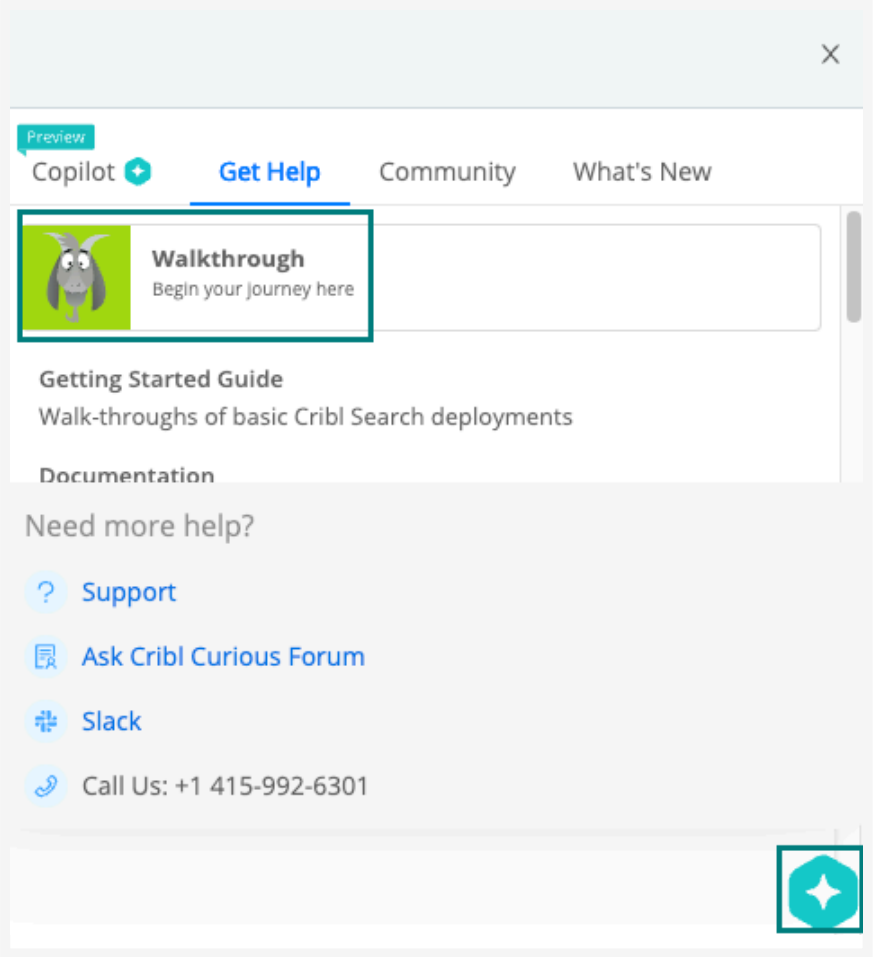
The goals that you select here, and the Sources and Destinations that you choose on the next page, will display personalized guidance as soon as you open your Organization.

8. Select **Complete** to access your new Organization.
9. Select the **Search** tile's **Explore** button to open Cribl Search.



Cribl.Cloud landing page

10. The first time you sign in, you'll see a tile with a goat who can walk you through setting up Cribl Search. If you skip the walkthrough now, you can start it at any later time by opening the Support/Copilot drawer's **Get Help** tab, where you'll find the goat.

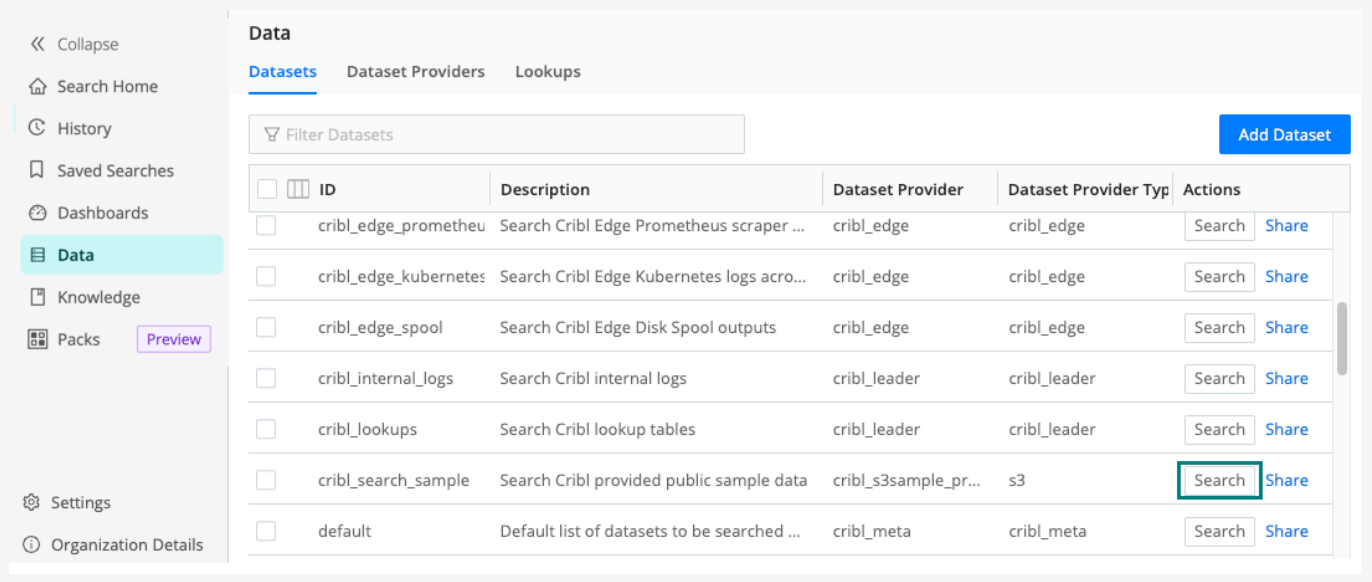


Walkthrough

Searching

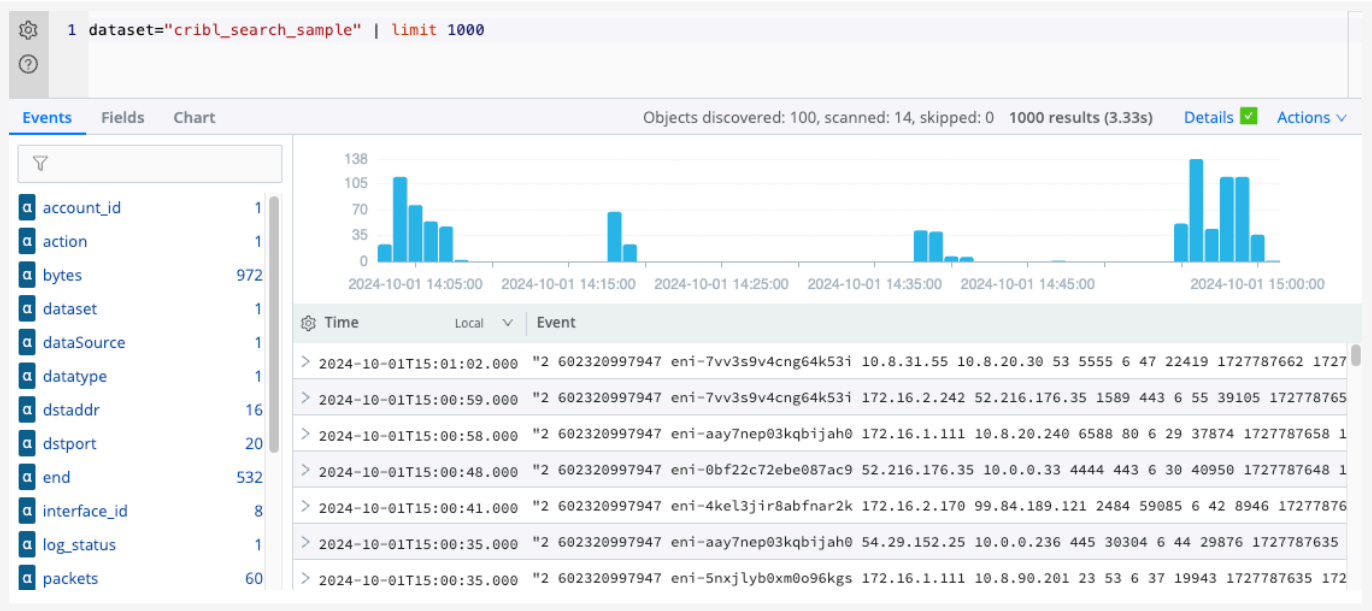
In this guide, you'll use a built-in sample Dataset named `cribl_search_sample`.

1. From the Cribl Search sidebar, select **Data** to open the default **Datasets** tab. Find the `cribl_search_sample` Dataset within the **Datasets** table.
2. Select the **Search** button on the right side of the `cribl_search_sample` row.



Datasets

3. **Search Home** opens and runs a basic search on the Dataset.



Limit Dataset results

We've referenced the Dataset with the field `dataset` and its `ID`, in this case, `cribl_search_sample`. The pipe | character indicates a new process for the query to run. Here, it's added a `limit` operator so the search only returns up to 1,000 events from the Dataset.

4. Next, let's run an aggregating search – counting the number of events per `source` (a built-in field that contains the name of the data object that included the event).

```
dataset="cribl_search_sample"
| limit 1000
| summarize count() by source
```

Copy Run in Cribl Search



Count by source

Aggregate results are displayed on the **Chart** tab. Cribl Search comes with rich charting options out-of-the-box, allowing you to adjust charts as needed.

Next Steps

Now that you have a Cribl.Cloud Organization and have seen a couple of basic searches, see the following topics:

- Cribl.Cloud for details on management.
- Cribl Search Tour for an overview of the Cribl Search UI.
- How to build a search for details on query writing.
- Common search examples with common use case examples.

These next topics walk you through searching your own data:

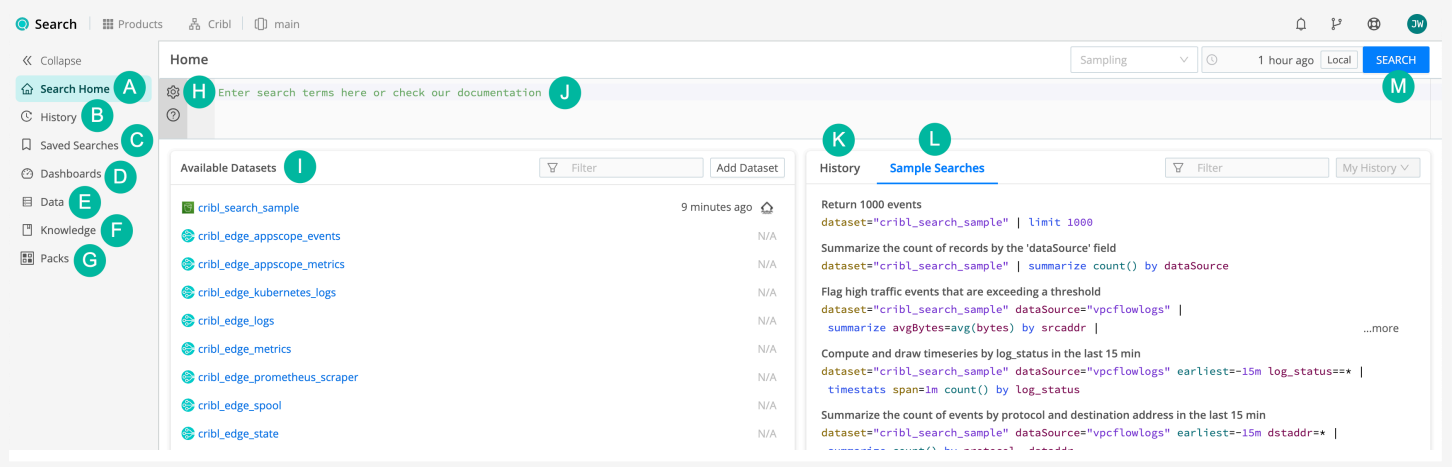
- Connect to Cribl Edge and Searching Cribl Edge – assumes that you have Cribl Edge Nodes running.
- Connect to Amazon S3 and Searching Amazon S3 – configuring and using an Amazon S3 Dataset Provider and Dataset. (This is a specific example of configuring Cribl Search to query other object stores.)

1.3. Cribl Search Tour

Find your way around the Cribl Search UI.

Search Home

Search Home is where you run your current search.

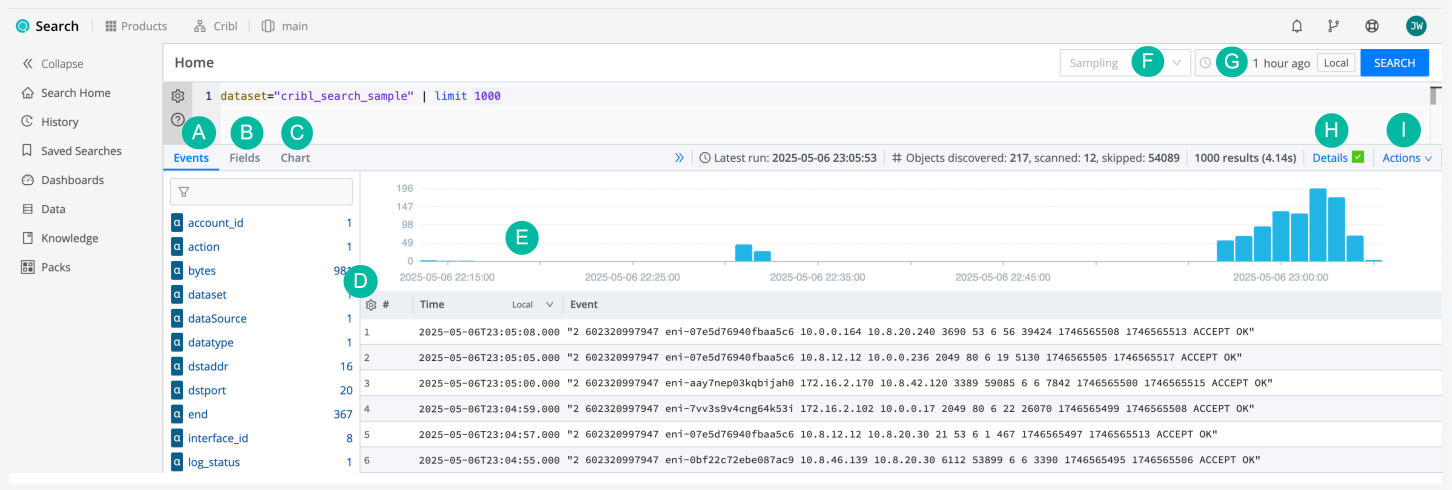


Search Home, before running a search

Letter	Description
A	**Search Home** is where you run the current search.
B	**History** keeps a detailed account of previous searches.
C	**Saved Searches** contain any searches you save.
D	**Dashboards** enable you to visualize your search results in a variety of ways.
E	**Data** is where you can manage your Datasets, Dataset Providers, and Datatypes.
F	**Knowledge** contains your lookups, Parsers, regexes, Grok patterns, and Macros
G	**Packs** is where you manage your Cribl Search Packs.
H	This gear button lets you change the UI options of the query box.
I	**Available Datasets**. The icon indicates Cribl Lake Datasets attached to a Lakehouse.
J	The query box, where you write your query.

Letter	Description
K	This **History** displays the most recent searches.
L	**Sample Searches** are predefined searches that enable you to explore Cribl Search.
M	The **SEARCH** button runs the query.

After running a search, your **Search Home** page looks like this:



Search Home, after running a search

Letter	Description
A	The **Events** tab shows non-aggregate search results as events.
B	The **Fields** tab displays search results as fields.
C	The **Chart** tab visualizes aggregate search results.
D	This gear button lets you change the results view options.
E	The Histogram lets you view the results for the time you select.
F	The **Sampling** drop-down controls the sampling ratio of the current search.
G	The **Time range** field sets the date and time range of the current search.
H	The **Details** modal shows the search details, such as logs and metrics.
I	The **Actions** drop-down lets you save the current search, export the results, and more.

If you enable Cribl Copilot, your **Search Home** might also show query suggestions, and other Copilot

features.

History

History is where you can view, rerun, save, and delete previously run searches, filter and examine their results, and add them to Dashboards.

Search Admins see the history of all searches run within their Organization. Search Users and Editors see only their own searches.

At the upper left of **History**, use the **Filter history** field to define general filters for the searches. At the upper right, filter the searches by selecting **All**, **Scheduled**, or **Not Scheduled**.

To refresh the searches, select the refresh button ↻ next to the **Last refreshed** timestamp.

History includes the following columns:

Column	Description
Search String	The search string used in the search. For scheduled searches, this column also displays the scheduled search **name** (hover over to copy) and the number of executions (select to view the execution history).
Search ID	Unique ID for the search. Select a search's ID to launch the **Details** modal for the search.
Event Count	Number of events that the search returned.
Type	The search's job type: `standard`, `scheduled`, `command`, or `dashboard`.
Status	The search's status: `Running`, `Completed`, or `Canceled`. If a `Completed` search includes items, the status is represented by the number of items and a green icon.
Dataset	The name of the Dataset that is defined for the search.
User	The name of the Member who ran the search.
Duration (s)	Duration of the search's run, in seconds.
Latest Run	Date and time when the search was run.
Actions	Menu of available actions for the search. Select the Actions button ⋯ to access these options.

Most of the columns in **History** support sorting and filtering:

- To sort, hover over a column heading to display the sorting arrows ⇕ at the right side of the column label. Select the up or down arrow to change the column's sort order.
- To filter, hover over a column heading, select the funnel button and define the filter you want to apply.

History also provides options for opening and deleting searches:

- To open a search in **Search Home**, select its row. Mind that this doesn't rerun the search, but only loads the cached results.
- To delete one or more searches, select the check boxes on the left side of their rows and select **Delete Selected Jobs** at the bottom of the page.

Actions Options

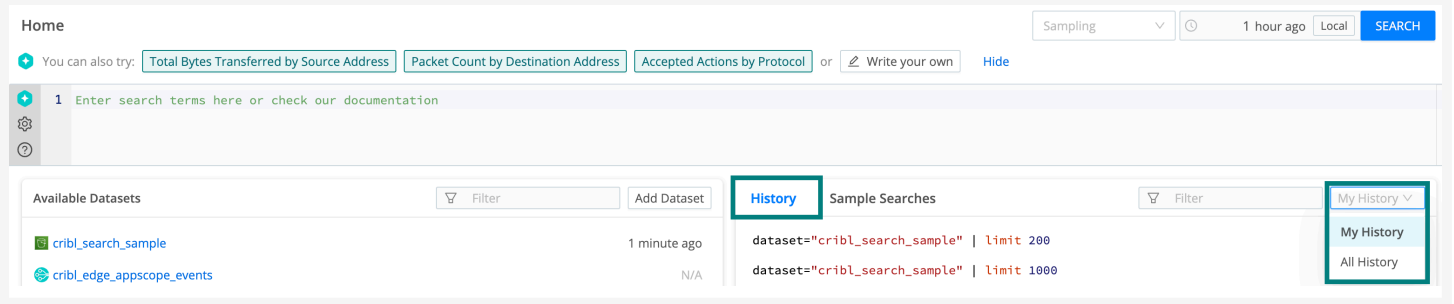
The **Actions** menu in **History** includes the following options:

- **Details**: Launches the **Details** modal.
- **Rerun**: Opens **Search Home** and reruns the search.
- **Save**: Launches the **Save Search** modal, in which you can name, describe, and adjust the search and save it for future access **Saved Searches**.
- **Search the Results**: Opens a new browser tab, on which you can query the result set of the search.
- **Add to Dashboard**: Turn the search into a visualization panel and add it to a new or existing Dashboard.
- **Cancel**: Cancel the search. Included in the **Actions** menu only for searches that are still running.
- **Delete**: Delete the search.

Toggle Between Personal and Organization History

As an Admin Search Member, you can quickly switch between viewing the history of your own searches and the history of all searches run in your Organization.

1. From the Cribl Search sidebar, select **Search Home**.
2. Select **History** next to **Sample Searches** (note that it's separate from the main **History**).
3. From the drop-down to the right, select either **My History** (default) or **All History**.



Saved Searches

The **Saved Searches** left tab is where you can add, run, clone, and delete saved searches, and add them to Dashboards. You can also export, share, and import saved searches in Packs.

Search Admins see all searches saved within their Organization. (Admins can also schedule searches saved by active users.) Search Users and Editors see only their own saved searches.

At the upper left of **Saved Searches**, use the **Filter saved searches** field to define general filters for the searches. To create a new saved search, select **Add Search** at the upper right. Filter the saved searches by selecting **All**, **Scheduled**, or **Not Scheduled**.

After you save a search, you can schedule it to run at a specified time interval.

Saved Searches include the following columns:

Column	Description
Name	The user-provided name and description of the search.
Search String	The search string used in the search. Select `...more` to launch the **Save Search** modal, in which you can adjust the search and save it as a new search.
Earliest	The earliest time within the search's time range. If the value is `0`, the search includes all relevant information without restrictions on how far back in time it looks.
Latest	The latest time within the search's time range. If the value is `now`, the time range includes the most recent information available at the time the search is executed.
Schedule	For scheduled searches, the schedule on which the search runs. For unscheduled searches, `Off`.
Next Run	For scheduled searches, the date and time at which the search will run next. For unscheduled searches, `N/A`.
Created By	The name of the user who saved the search.
Notifications	Indicates whether at least one Notification is enabled on a saved search. Select the indicator to open the search's configuration modal, where you can access the alert's details.
Actions	Menu of available actions for the search. Select **Run** to run the saved search. Select the Actions button ⋯ to access these options.

Most of the columns in **Saved Searches** support sorting and filtering:

- To sort, hover over a column heading to display the sorting arrows ⇕ at the right side of the column label. Select the up or down arrow to change the column's sort order.
- To filter, hover over a column heading, select the funnel button ▽ and define the filter you want to apply.

Select any row in the table to launch the **Save Search** modal, in which you can adjust the saved search and save it as a new search for future access from **Saved Searches**.

To delete one or more searches, select the check boxes on the left side of their rows and select **Delete Selected searches** at the bottom of the page. If the search you want to delete belongs to a Pack, see Delete a Pack Resource.

Actions Options

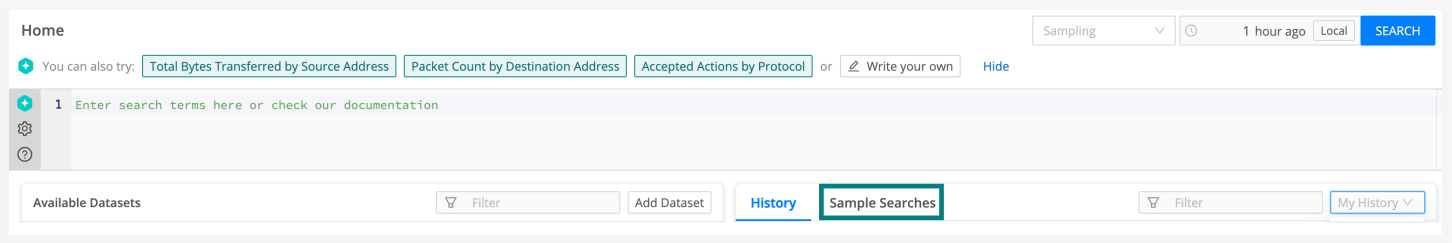
The **Actions** menu in **Saved Searches** includes the following options:

- **Clone**: Launches the **Save Search** modal, in which you can adjust the search and save it as a new search.
- **Add to Dashboard**: Turn the search into a visualization panel and add it to a new or existing Dashboard.

> **Changing Saved Search Ownership**
>
> If the creator of a saved search is no longer an active Cribl Search Member, Search Admins with access to the search can continue to run it ad hoc. (They will now appear as the user requesting the search.) However, they will be unable to schedule the search. The Admin will need to copy the search's details, and to then re-create, save, and schedule the search under their ownership.

Sample Searches

Cribl Search provides several searches against its own internal logs. You can find them in **Search Home** > **Sample Searches**.



Sample Searches

Search Results

A search returns results on three tabs:

- **Events** – displays non-aggregated search results. See Events Tab for details.
- **Fields** – displays all of the returned fields on a table. See Fields Tab for details.
- **Chart** – displays aggregated search results on a Chart and table. See Chart Tab for details.

Based on the type of the results, Cribl Search automatically selects the appropriate tab. However, you can override this by using the `render` operator.

Track the Progress of Your Search

Once a search starts running, you can see its progress next to the **Details** indicator.



Search progress

Select « to expand the toolbar. Depending on the type and state of your search, you'll see the following details:

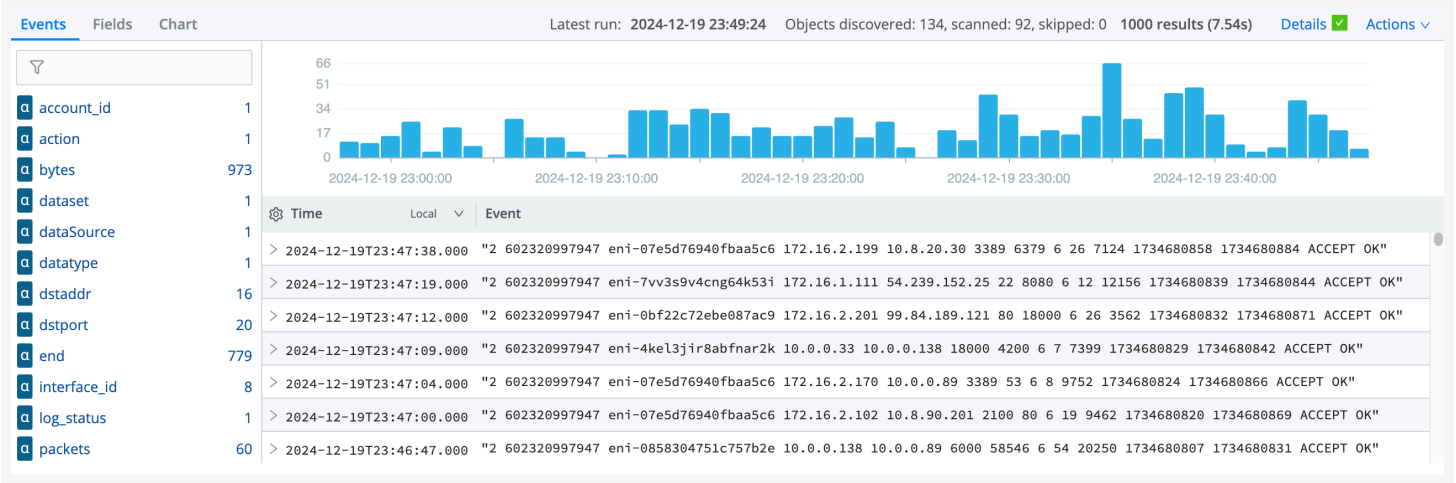
- Name of the search (if it's a saved search).
- **Latest run**: The timestamp of the most recent initiation of that search. If you opened the search from History, this tells you how old the displayed results are.
- Total number of objects that Cribl Search has **Discovered**, successfully **Scanned**, or **Skipped** (for example, because they were in an unsupported Amazon S3 storage class).
- Number of **results** returned.
- Duration of the search.

Events Tab

The **Events** tab displays a table with the raw log events returned from the search, along with a histogram, and the field browser.

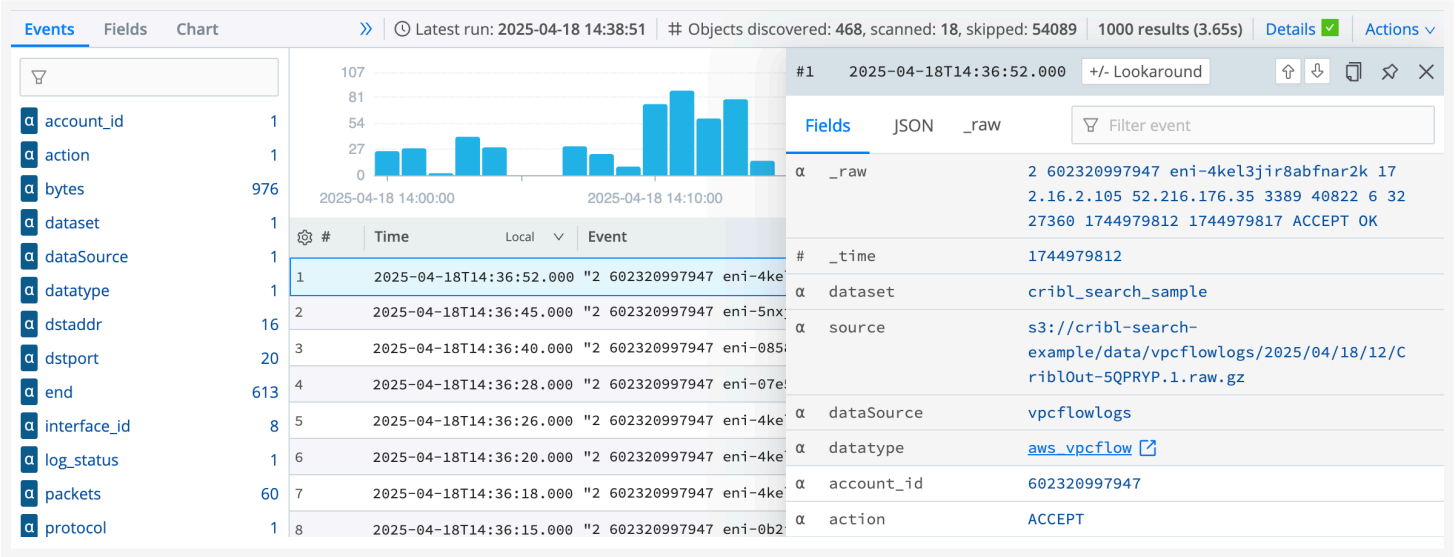


To display results in the **Events** tab regardless of the search's type, append your query with `| render event`.



Events tab

Select any of the rows to open the event details drawer, which displays the fields of the selected event.



Event details drawer

Now, you can navigate between events using your keyboard's **Up** and **Down** arrow keys.

In the drawer, you can:

- Filter the fields displayed. Type into the search bar at the top right.
- View the fields in JSON format. Select **JSON** at the top.
- View the `_raw` field, which contains the original event data. Select **_raw** at the top.
- Look up events outside of the time frame of the current search. Use the **Lookaround** at the top.
- Copy the event's JSON to the system clipboard. Select at the top.
- Pin the drawer open. Select at the top right.

You can also quickly incorporate any of the fields into your next search. Select a field to see these options:

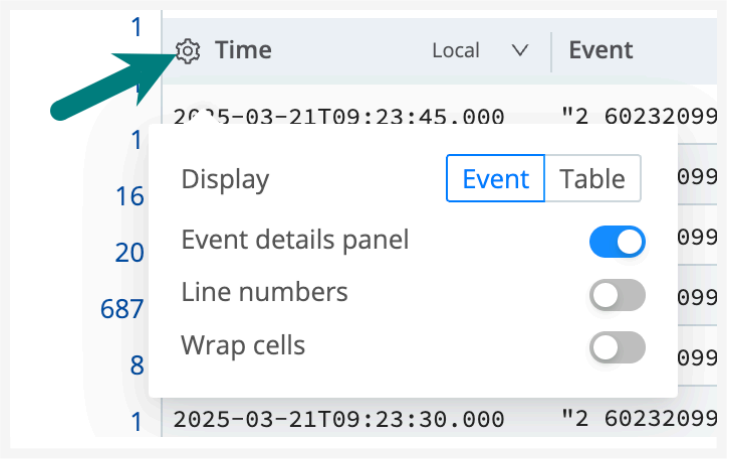
- **Add field in search**: Include this field name and value in the current query. If your query already includes this field, you'll see the **Remove field from search** option.
- **Exclude field from search**: Exclude this field value in the query (for example, `action!="ACCEPT"`).
- **New search with field**: Open another Cribl Search window, and include this field name and value in the new query.
- **Copy value to clipboard**: Copy the value of this field to the system clipboard.

To close the drawer, click anywhere outside of it, or select **X** at the top right.

Change the Event View Options

When viewing the **Events** tab, you can change how the results are displayed.

Select the gear button in the heading row (to the left of **Time**).



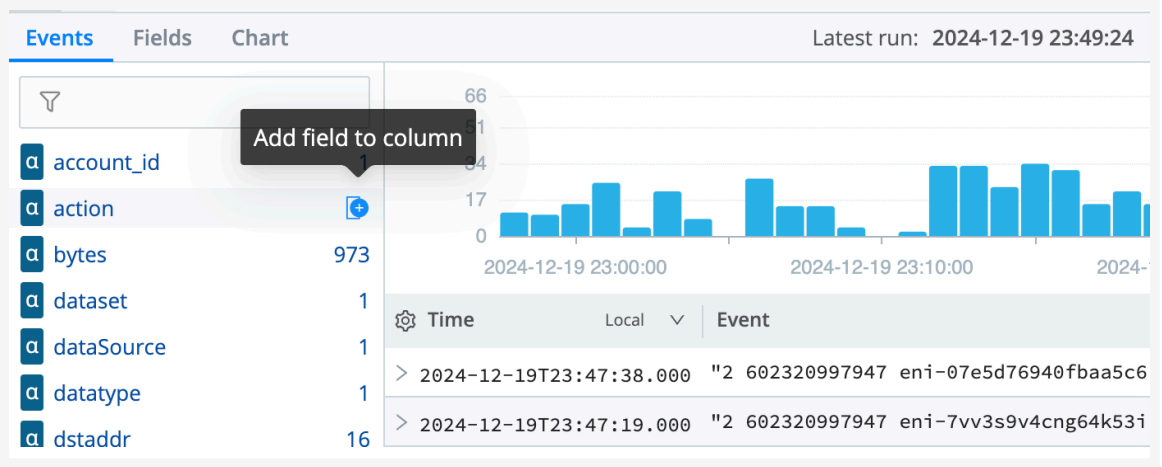
Event view options

Here, you can:

- Toggle **Event details panel** off to disable the event details drawer. You'll now be able to view fields directly in the table, by selecting **>** at the left of an event row.
- Toggle **Display** to view the original **Event**s, or a **Table** with only the returned fields.
  (The **Table** option allows you to adjust the order of columns displayed. Select and drag a column to the desired location.)
- Toggle **Line numbers** on or off to display or hide line numbers.
- Toggle **Wrap cells** on to prevent values from overflowing.

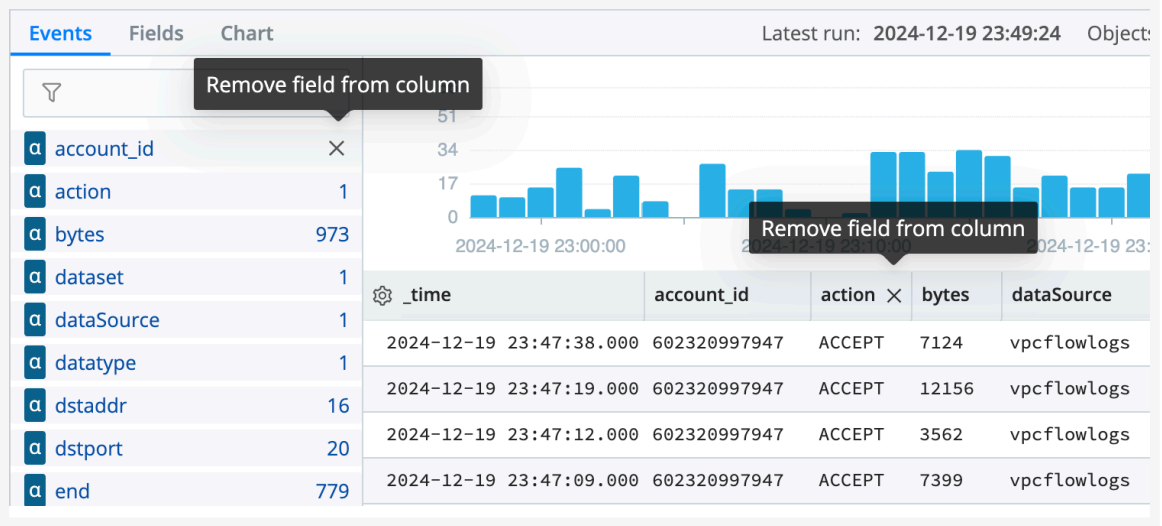
Events Shortcuts

In Events view, you can hover over field names in the sidebar to display an **Add field to column** button. Select these buttons to quickly build a custom Table view, showing only the fields that interest you.



Add columns

In Table view, you can hover over field names in the sidebar or heading row to display a close box. Select the **x** to remove their columns from the table.

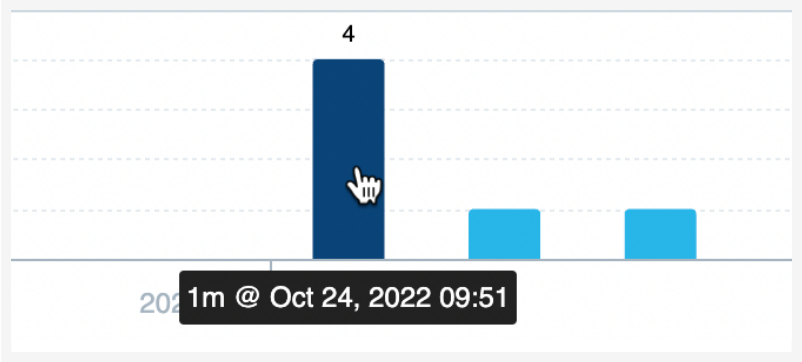


Remove columns

Columns that you hide using these shortcuts are not permanently hidden. They will reappear when you refresh the page or reload this search.

Histogram

You can select bars in the histogram to view results for only the selected times. Use shift+select to select multiple bars.



Histogram bar selection

Approximate Versus Precise Events Counts

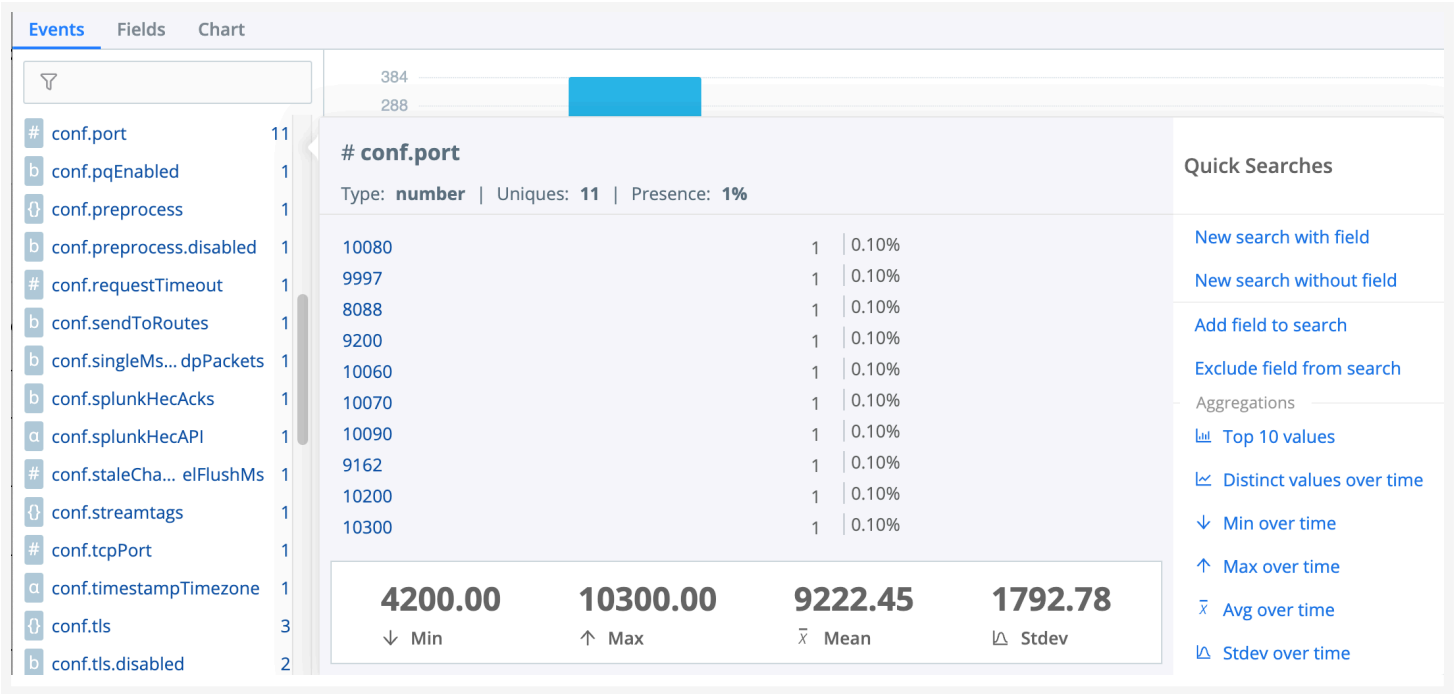
In the **Events** field browser's left column, and on the Fields tab, both covered below: The displayed event counts are approximations, and are not expected to be exact. Top-*N* lists are generated using probabilistic analysis, and are similarly not exact. However, you can compute precise results for a given field using a query of this form:

```
dataset=<DatasetName> | summarize count() by <fieldName>
```

Copy

Field Browser

The field browser on the **Events** tab allows you to easily identify information about the fields returned from your search. Use the search bar to filter the returned fields. The browser gives you the data type, the unique count of values the field has, and the percentage of log events returned with the field.



Event field browser

The options in the **Quick Searches** panel automatically generate and run a new search:

Quick Search	Query Example
New search with field	`response_time="*"`
New search without field	`response_time!="*"`
Add field to search	`dataset="cribl_search_sample" response_time="*" | limit 1000`
Exclude field from search	`dataset="cribl_search_sample" response_time!="*" | limit 1000`
Top 10 values	`dataset="cribl_search_sample" | limit 1000 | top-hitters 10 of response_time`
Distinct values over time	`dataset="cribl_search_sample" | limit 1000 | timestats dcount(response_time)`
Min over time	`dataset="cribl_search_sample" | limit 1000 | timestats min(response_time)`
Max over time	`dataset="cribl_search_sample" | limit 1000 | timestats max(response_time)`
Avg over time	`dataset="cribl_search_sample" | limit 1000 | timestats avg(response_time)`

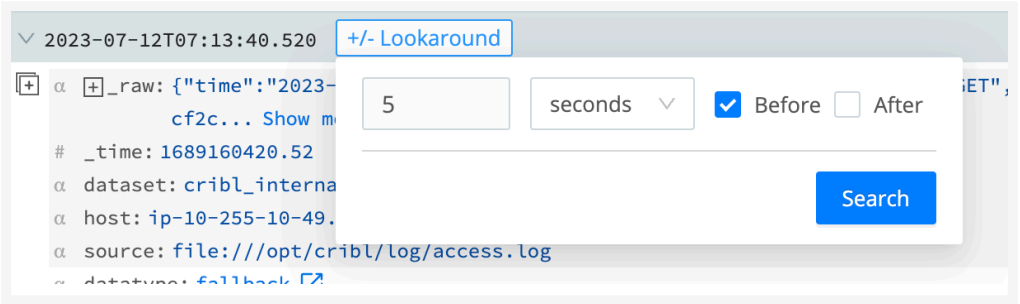
Quick Search	Query Example
Stdev over time	`dataset="cribl_search_sample" | limit 1000 | timestats stdev(response_time)`

See the Fields tab for a table of all of the returned fields.

Lookaround

Lookaround allows you to filter search results by adding or subtracting seconds, minutes, hours, or days, enabling quick exploration of surrounding events.

Select an event's `Time` field or expand a row and select **+/- Lookaround** to view the modal.



Lookaround UI

Export Search Results as NDJSON

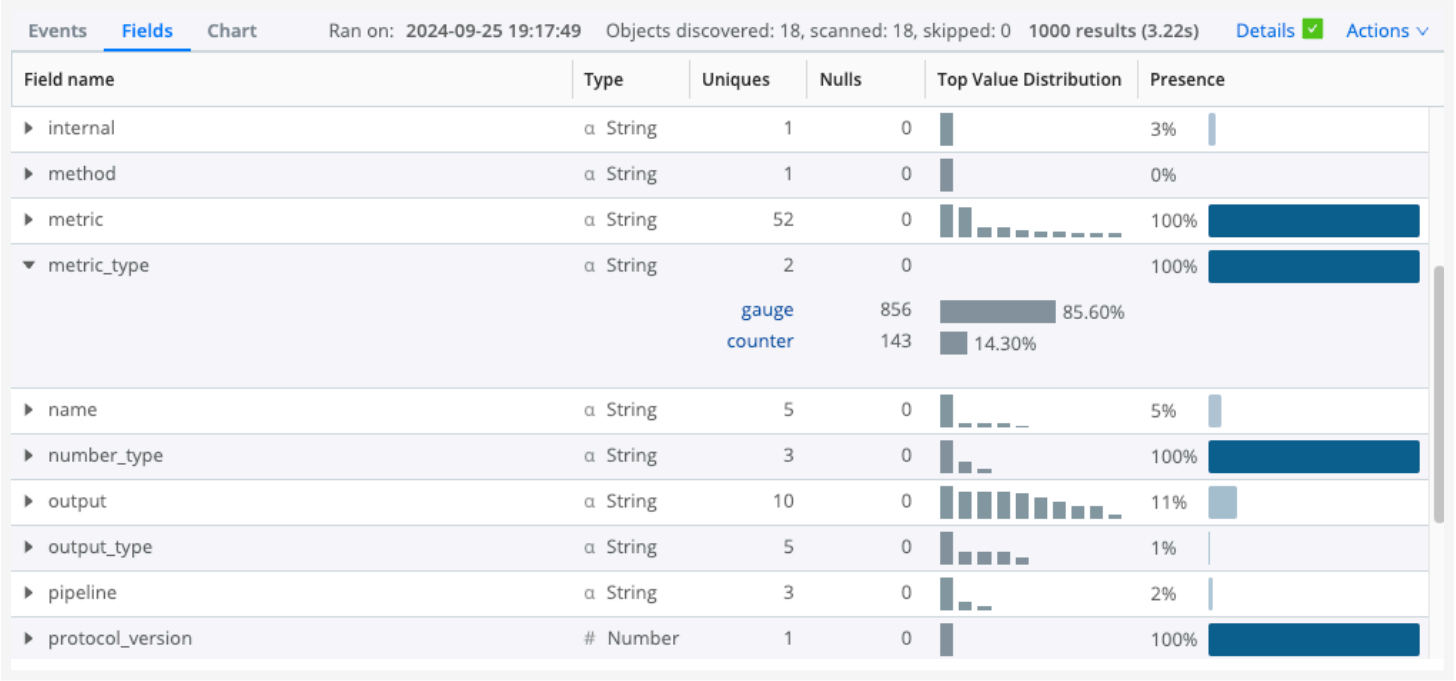
You can export the raw results of any search in the Newline Delimited JSON (NDJSON) format.

1. Run a search.
2. At the bottom right of the query box, select **Actions**.
3. Select **Export results as NDJSON**. The results are downloaded to your default download location.

Fields Tab

The **Fields** tab displays all of the returned fields on a table by the following dimensions:

- **Type**: Data type.
- **Uniques**: Number of unique values.
- **Nulls**: Number of `null` values.
- **Top Value Distribution**: How often values occur using the standard cumulative beta distribution function.
- **Presence**: Percentage of results that contain the field.



Fields tab

The table supports sorting and filtering, and allows you to adjust the order of columns.

- Select a column heading to change its sorting order.
- Hover over a column heading and select the funnel icon to define a filter.
- Select the triangle to the left of a **Field name** to expand and collapse the list of returned values.



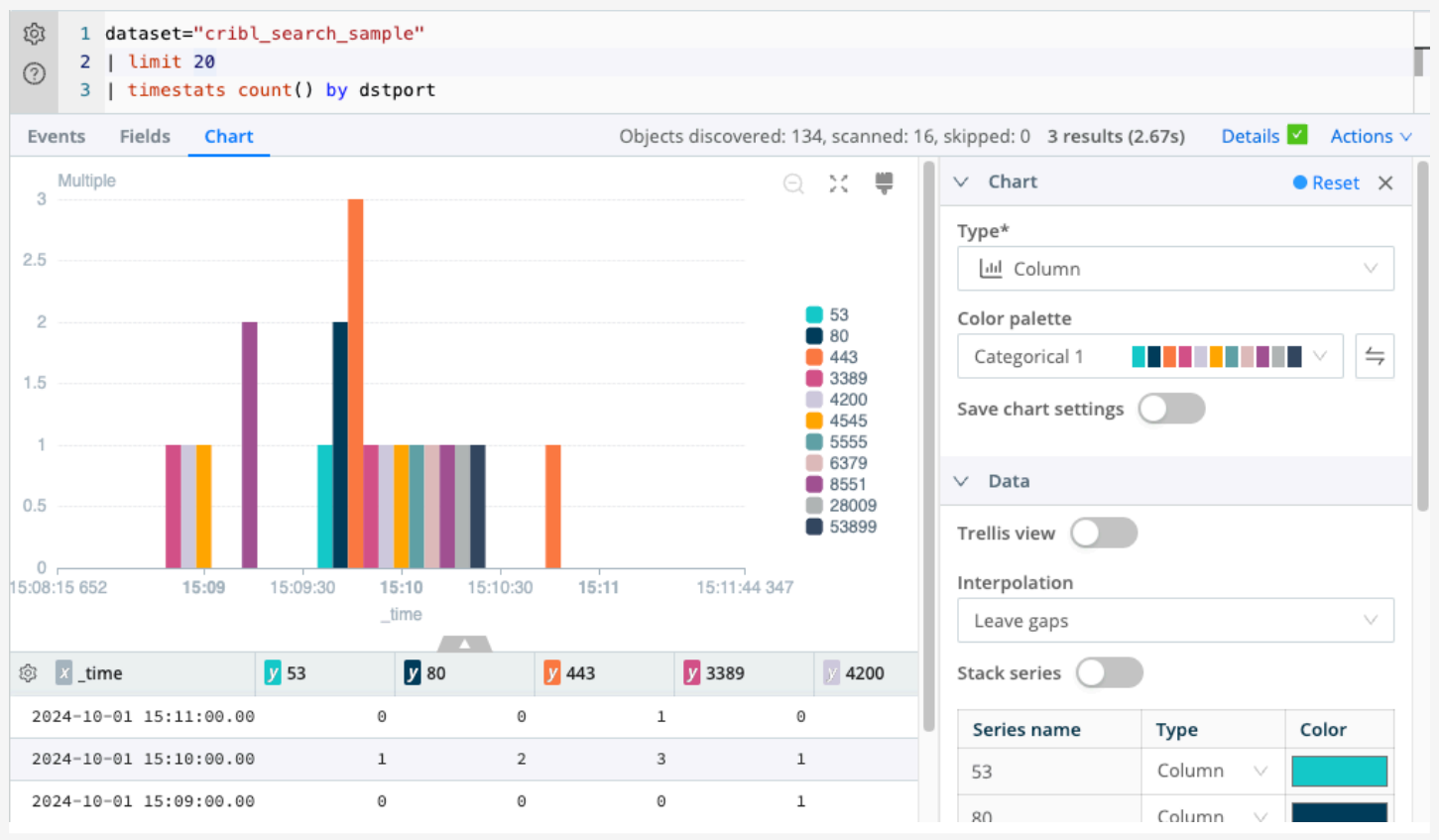
See the note above about Approximate Versus Precise Events Counts.

Chart Tab

When you run the `summarize`, `eventstats`, or `timestats` operator along with an aggregation function, the **Chart** tab automatically displays your results in a Chart, along with a corresponding results table. You can select from various Chart types and color palettes, manipulate how your results are plotted, and customize the results table display.



To display results in the **Chart** tab regardless of the search's type, append your query with `| render table`.



Aggregate results area on a Bar Chart

For detailed information about manipulating data and visualizations in aggregate search results, see Charting.

2. Connect Cribl Search To Your Data

Link Cribl Search to your data by setting up Dataset Providers and Datasets.

Cribl Search offers native support for a wide array of data sources. To start searching any of them, set up a Dataset Provider and a Dataset. For that, you need to be a Cribl Search **Editor** or **Admin**.

Add a Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. To start adding a new one, go to **Data** > **Dataset Providers** > **Add Provider**.

For detailed instructions on how to set up each Dataset Provider type, follow the relevant link from the list of supported data sources.

Add a Dataset

A Dataset tells Cribl Search what data to search from the Dataset Provider. To start adding a new one, go to **Data** > **Datasets** > **Add Dataset**.

For detailed instructions on how to set up each Dataset type, follow the relevant link from the list of supported data sources.

List of Supported Data Sources

Cribl Search offers native support for the data sources listed below. Follow the links to learn how to set up each one:

- Amazon S3
- Amazon Security Lake
- AWS API
- Azure API
- Azure Blob Storage
- Azure Data Explorer
- ClickHouse
- Cribl Edge
- Cribl Lake / Lakehouses
- Cribl Stream's "Data Lake Amazon S3" Destination

- Elasticsearch
- Google Cloud Platform API
- Google Cloud Storage
- Google Workspace API
- Generic HTTP API
- Microsoft Graph API
- Okta API
- OpenSearch
- Prometheus
- Snowflake
- Tailscale API
- Zoom API

If there's no native support for your data source type, there still might be ways to connect to your data. You can set up a Generic HTTP API Dataset, use the `externaldata` operator, or reach out to Cribl Support for finding a solution.

Connect Cribl Search to Other Cribl Products

Cribl Search seamlessly integrates with Cribl Stream, Cribl Edge, and Cribl Lake (with or without Lakehouses), as they're all part of your Cribl.Cloud Organization. In most cases, you can start searching your data right away. For more information, see the pages listed below.

Cribl Edge:

- Connect to Cribl Edge
- Built-In Cribl Edge Datasets
- Searching Cribl Edge Data

Cribl Lake:

- Connect to Cribl Lake
- Searching Cribl Lake
- Search a Lakehouse

2.1. Connect Cribl Search To Cribl Lake

Configure Cribl Search to query your Cribl Lake or Lakehouse data.

Cribl Lake Datasets work as Cribl Search Datasets out-of-the-box, so you can instantly start searching them.

You can assign Lake Datasets to Lakehouses, for dramatically faster search responses and more-predictable billing.

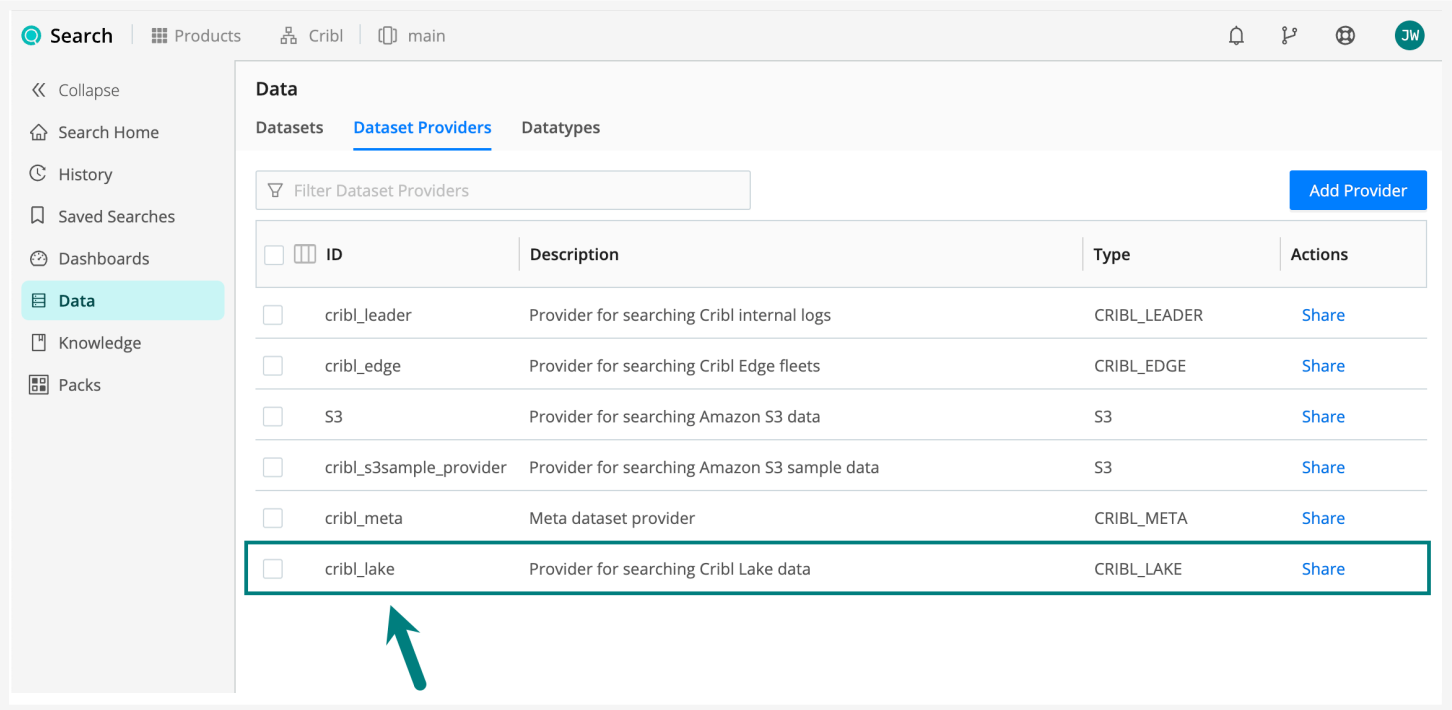
Where you need direct ownership of your data, for compliance or other purposes, you can use Storage Locations to create Lake Datasets directly on Amazon S3. These Datasets use the Amazon S3 Standard storage class.

For more information on Lake and Lakehouses, see the Cribl Lake docs.

Add a Cribl Lake Dataset Provider

Cribl Search comes with a preconfigured Dataset Provider for Cribl Lake (called `cribl_lake`), so you don't need to add it explicitly.

You can see it by going to **Products** (on the top bar) > **Search** > **Data** > **Dataset Providers**.



See? It's already there!

Add a Cribl Lake Dataset

You add Lake Datasets in the Cribl Lake UI. To find out how, see the Cribl Lake docs: Create a New Dataset.

Send Cribl Search Results to Cribl Lake

After you create a Dataset in Cribl lake, you can send Cribl Search results to it, using the `export` operator. Here is a simple example:

```
// export Cribl Search results to an existing Lake Dataset
dataset="cribl_search_sample"
| export to myLakeDataset
```

Copy

(You can find more `export` examples here.)

Typically, you're exporting data in which Cribl Search has already parsed field names and values. This makes the data compatible with Lakehouses. However, if you happen to send **unparsed** data, you won't be able to search it at Lakehouse speed.

Search a Cribl Lake Dataset

You can query your Lake Datasets just like any other Search Datasets. See Searching Cribl Lake for some query examples.

Search a Lakehouse

Searching a Lake Dataset that's assigned to a Lakehouse is significantly faster.

To create a new Lakehouse, see the Cribl Lake docs: Add a New Lakehouse.

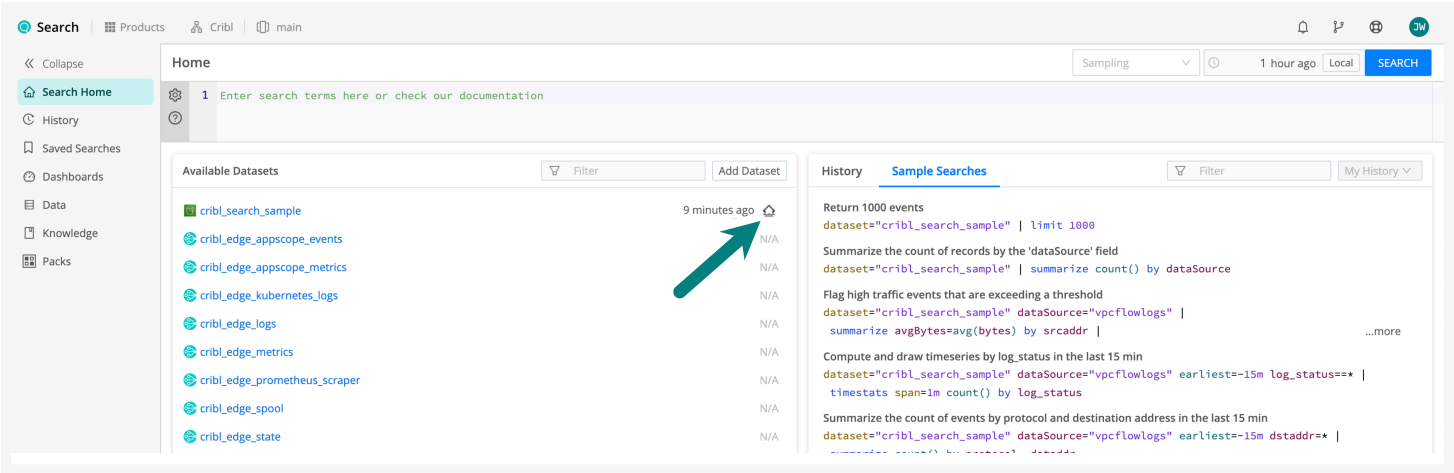
See how to:

- Find Out if a Dataset is Assigned to a Lakehouse
- Verify Lakehouse Use
- Search Multiple Lakehouse Datasets
- Disable Lakehouse Use

See also: Cribl Search Differences with Lakehouse.

Find Out if a Dataset is Assigned to a Lakehouse

To see if a Dataset you're about to search is assigned to a Lakehouse or not, look for the icon. For example, look at the list of available datasets in your **Search Home**:



A Dataset assigned to a Lakehouse

You can also go to **Products** (on the top bar) > **Search** > **Data** > **Datasets** and look at the **Lakehouse** column.

Verify Lakehouse Use

To find out whether a search successfully used a Lakehouse, take a look at the tracking bar.

If the search failed to use a Lakehouse, the bar presents information about potential reasons.

Search Multiple Lakehouse Datasets

You can run a single query against multiple Lakehouse-assigned Datasets. For the query to execute at Lakehouse speed, all Datasets in the query must be Lakehouse-assigned, and your query must also meet one of these conditions:

- Include only operators from the following group: `cribl`, `centralize`, `extend`, `extract`, `foldkeys`, `limit`, `mv-expand`, `pivot`, `project`, `project-away`, `project-rename`, `search`, and `where`.
- Or, if you use any other operators, insert the `centralize` or `limit` operator before them. (Result counts will be further constrained by usage group limits.)

If neither of the above conditions is met, or if your query includes non-Lakehouse Datasets, the query will run at standard speed.

Disable Lakehouse Use

You can use a `set` statement with the `lakehouse` option to control whether a query can use a Lakehouse or not.

For example, to test how a Dataset assigned to a Lakehouse performs without it:

```
set lakehouse="off" dataset="lakehouse_dataset"
```

Copy

Cribl Search Differences with Lakehouse

Executing Cribl Search queries against a Lakehouse-assigned Dataset changes some behavior and results, compared to executing the same queries without Lakehouse caching. For details, see Lakehouse Search Differences.

2.2. Connect Cribl Search To Amazon S3

Configure Cribl Search to query your Amazon S3 data.

Amazon Simple Storage Service (Amazon S3) is an object storage service that allows you to store and retrieve data from anywhere.

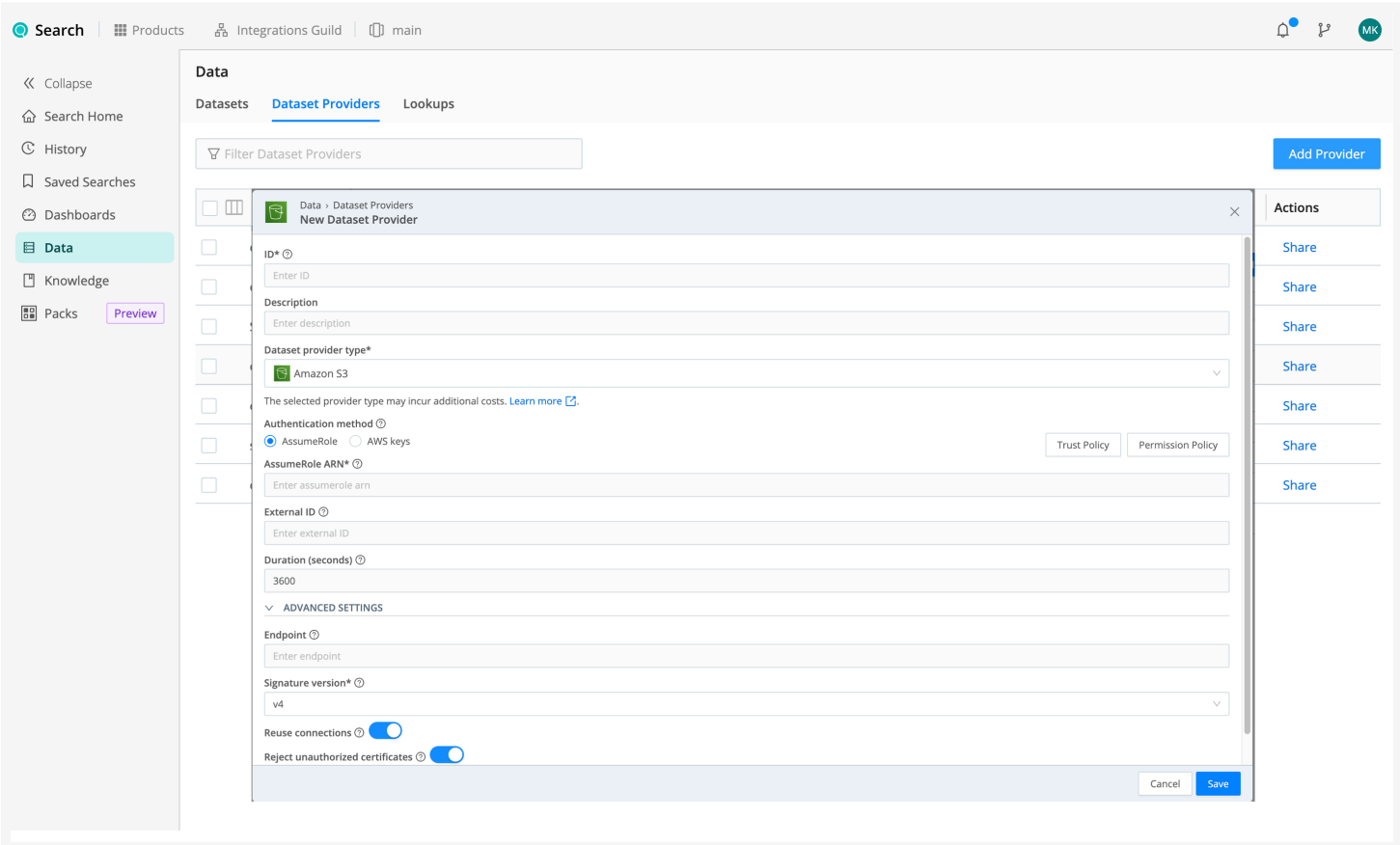
In this guide, you'll set up a Dataset Provider and a Dataset to search objects in your Amazon S3 bucket.

Data transfer costs and other charges may apply. For cost and compatibility details, see the Data Transfer Charges and Storage Class Compatibility sections below.

Add an Amazon S3 Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an Amazon S3 Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.



S3 Dataset Provider configuration

Basic Configuration

Configure the **New Dataset Provider** modal as follows:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, (for example, `my_dataset_provider_1`)
2. **Description** is optional. Here, you can enter a summary that will clarify this Dataset Provider's purpose to other users.
3. Set the **Dataset Provider type** to `Amazon S3`.
4. **Authentication method** provides two options, **Assume Role** and **AWS keys**. For details on configuring each option, see Authentication Method below.

Advanced Settings

This section provides the following optional configurations:

- **Endpoint**: S3 service or compatible endpoint. If empty, defaults to AWS Region-specific endpoint.
- **Signature version**: Signature version to use for signing S3 requests. Defaults to `v4`.
- **Reuse connections**: Whether to reuse connections between requests. Toggling on (default) can improve performance.
- **Reject unauthorized certificates**: Whether to reject certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.
- **Apply ABAC source-ip tagging**: Displayed only if you've set the **Authentication method** to **AssumeRole**. For details, see ABAC Tagging below.

Wrapping Up

Select **Save** when your configuration is complete.

Authentication Method

Your choices here are **AssumeRole** or **AWS keys**. For rich details on both options, beyond the configuration basics in this section, see Grant Access to AWS.

AssumeRole Authentication

Selecting **AssumeRole** requires the IAM role's ARN (**AssumeRole ARN**), and also exposes the following options:

- The **External ID** on this Dataset Provider must match the external ID defined in the IAM Role Trust Policy.
- The **Duration (seconds)** defines how long the AssumeRole's session lasts. Minimum is `900` (15 minutes), default is `3600` (1 hour), and maximum is `43200` (12 hours).

- The ABAC tagging controls described in the next section.

To the right, select the **Trust Policy** and **Permission Policy** buttons to see the default policies that you can copy and expand, so you don't have to write them from scratch. For more information, see Example Trust Policy and Expand the Permission Policy.

AWS Keys Authentication

Selecting **AWS keys** requires the IAM user's **Access key** and **Secret key**.

To the right, select the **Permission Policy** button to see the default policy that you can copy and expand, so you don't have to write it from scratch. For more information, see Expand the Permission Policy.

ABAC Tagging

Select **Apply ABAC source-ip tagging** if you want to make your AssumeRole sessions more secure by enabling attribute-based access controls (ABAC). By enabling this option, you can hard-code Cribl's Virtual Private Cloud Endpoint (VPCE) into your IAM trust policy.

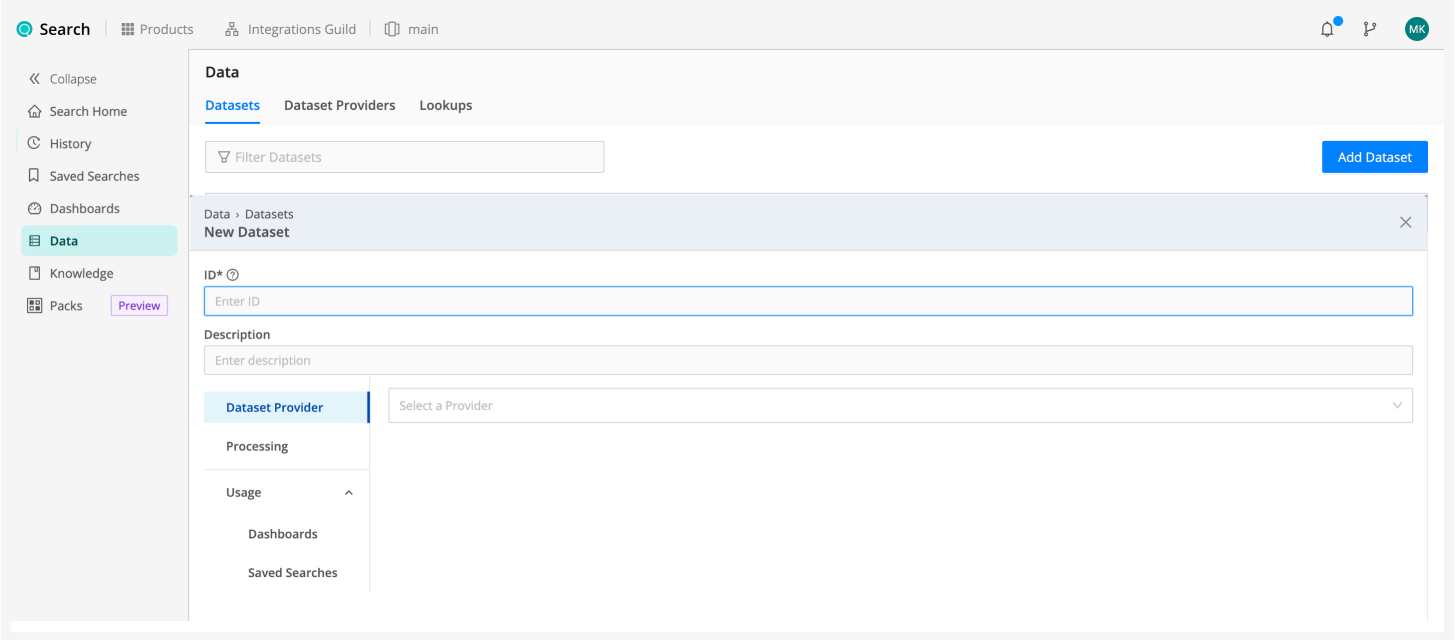
Selecting this option exposes **VPC Endpoint IDs**. Use the adjacent Copy button to transfer the VPCE ID to your IAM trust policy, as shown in Trust Policy with ABAC Tagging.

Completing both parts of this configuration will make your credentials harder to impersonate. Specifically, your AssumeRole sessions will include a `source-ip` tag, enabling you to enforce a condition that `aws:RequestTag/source-ip` equals `${aws:SourceIp}`.

Add an Amazon S3 Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.



S3 Dataset configuration

Configure the **New Dataset** modal as follows.

Identify the Dataset

Use the first few fields to uniquely identify this Dataset and specify its type:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`). You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset (for example, `dataset=my_dataset_1`).
2. **Description** is optional. Here, you can enter a summary that will clarify this Dataset's purpose to other users.
3. Set **Dataset Provider** to the **ID** of an Amazon S3 Dataset Provider.

Set Up Paths, Regions, Partitioning

Define at least one path, Region, and partitioning configuration:

1. **Skip event time filter** is optional. Toggle this on only if you're sure you want to set the time range of your searches by partition time-boundaries rather than events' timestamps
2. **Bucket path** is the path to the S3 objects you'd like to search. Start the path with the bucket name, and add partitioning details. Tokens and key-value pairs are supported. For example: `<my-bucket>/<path>/${_time:%Y}/${_time:%m}/${_time:%d}/${_time:%H}/`.
   For details, see Amazon S3 Bucket Paths.
3. Toggle off **Auto-detect region** if you want to use the adjacent **Region** drop-down to specify where the S3 bucket is located. When toggled on, Cribl automatically detects the Region and locks the drop-down.

4. **Path filter** is a JavaScript filter expression that is evaluated against the S3 bucket path. Defaults to `true`, which matches all data, but you can customize this value.
5. The **Partitioning scheme** defaults to `Defined in Path`. For Splunk-specific alternatives (and corresponding options), see Amazon S3 Partitioning Scheme.

Multiple Path Configs

Select **Add bucket/path** to define as many other path configurations as you want for this Dataset. Each path can have its own filter, Region, and partitioning scheme.

As with parallel Datasets, each partitioning scheme applies per path. When searching a Dataset, Cribl Search will search all its configured paths. However: **a failed search on any of a Dataset's paths will fail the whole search.**

Set Up Datatypes and Storage Classes

On the **Processing** left tab, you specify **Datatypes** to break data down into discrete events and define fields so they're ready to search.

Use the drop-down to set the first rule to `AWS Datatypes`. Optionally, click **Add Datatype Ruleset** to select more rulesets – each of which contains rules applied to the data searched in your Dataset. For details, see Datatypes.

On the **Storage classes** left tab, select the storage classes that you want this Dataset to search against. You can minimize retrieval costs by selecting only warmer classes. You must select at least one class. For details, see Storage Classes.

Save the Configuration

Bypass the **Usage** left tabs for now, and select **Save** when your configuration is complete.

Path/Partitioning Considerations

This section offers details and guidance on configuring search paths, filters, and partitioning schemes. For best practices covering path design across all object stores, see Optimize Paths.

Amazon S3 Bucket Paths

Each **Bucket path** specifies a separate bucket of data within the Dataset. It defines the scope of this data, to narrow down what data is in the Dataset. This field supports tokens and key-value pairs. For example:

- `my-bucket/${data}/` – where `data` becomes a field for all events of that Dataset.
- `my-bucket/${data}/${*}` – equivalent, the wildcarded path is skipped.

- `my-bucket/<key=value>/<someVarOfInterest>` – see Hive-Style Paths just below.

Hive-Style Paths

To search paths of the form `my-bucket/<key=value>/<someVarOfInterest>`, your expression should place the wildcard `{*}` in the ordinal position, to ignore the hive (K-V) segment:

```
my-bucket/${*}/${someVarOfInterest}
```

This will allow Cribl Search to find the wildcarded subdirectory automatically.

Basic Tokens

Basic tokens' syntax follows that of JS template literals: `${token_name}` – where `token_name` is the field (name) of interest.

For example, if the path was set to `/var/log/${hostname}/${dataSource}/`, you could use a filter such as `hostname=='myHost' && dataSource=='mydataSource'` to specify data only from the `/var/log/myHost/mydataSource/` subdirectory.

Time-Based Tokens

Paths with time notation can be referenced with tokens, having a direct effect on the earliest and latest boundaries. The supported time fields are:

- `_time` is the raw event's timestamp.
- `__earliest` is the search start time.
- `__latest` is the search end time.

Time-based tokens are processed as follows:

- For each path, times must be notated in descending order. So Year/Month/Day order is supported, but Day/Month/Year is not.
- Paths may contain more than one time component. For example, `/my/path/2020-04/20/`.
- In a given path, each time component can be used only once.
  So `/my/path/${_time:%Y}/${_time:%m}/${_time:%d}/...` is a valid expression format, but `/my/path/${_time:%Y}/${_time:%m}/${host}/${_time:%Y}/...` (with a repeated `Y`) is not supported.
- For each path, all extracted dates/times are considered in UTC.



Cribl recommends that your path always include and tokenize the largest available time fields, proceeding down to the smallest desired time fields. Otherwise, your searches might yield unexpected

results, because Cribl Search will default the omitted fields to their earliest allowed value.

The following `strptime` format components are allowed:

- `Y`, `y` for years
- `m`, `B`, `b`, `e` for months
- `d`, `j` for days
- `H`, `I` for hours
- `M` for minutes
- `S` for seconds
- `s` for Unix-style Epoch times (seconds since 1/1/1970)

Time-based token syntax follows that of a slightly modified JS template literal: `${_time: some_strptime_format_component}`. Examples:

Path	Matches
`/path/${_time:%Y}/${_time:%m}/${_time:%d}/...`	`/path/2020/04/20/...`
`/path/${_time:year=%Y}/${_time:month=%m}/${_time:day=%d}/...`	`/path/year=2020/month=`
`/path/${_time:%Y-%m-%d}/...`	`/path/2020-05-20/...`

Set Query Time Range by Partition

If your data is partitioned by time (so your paths probably use time-based tokens), you can tell Cribl Search to ignore events' timestamps, and instead set the time range using the partition time-boundaries.

For example, an event with the timestamp of `2025-07-01T23:59:00.000` might be stored in the July 2 partition. Queries for July 1 or July 2 might not find it: the first due to wrong partition, the second due to wrong `_time`. Using partition time for the query range avoids this issue.

To search by partition time range, toggle on **Skip event time filter** when setting up your paths. Then, any time range you set (in the Cribl Search UI or with `earliest/latest` in the query text) will use partition time instead of event time.



If you toggle on the **Skip event time filter** option, Cribl Search will ignore the `_time` field in your events. Do **not** enable this option if your paths don't use time-based tokens.

Amazon S3 Path Filter

Each **Path filter** is a JavaScript expression that Cribl Search evaluates against the corresponding **Bucket path**. The **Path filter** field's value defaults to `true`, which matches all data. However, you can customize this value almost arbitrarily.

For example, with this **Path filter**:
`source.endsWith('.log') || source.endsWith('.txt')`
...this configuration will search only files/objects with `.log` or `.txt` extensions.

At each **Path filter** field's right edge are a Copy button and an Expand button that opens a validation modal.

With DDSS or SmartStore partitioning, you can provide the index name as the **Path filter**, for example:
`index=="index_y"`

Amazon S3 Partitioning Schemes

Each **Partitioning scheme** drop-down offers alternative options for partitioning inbound data from Splunk. You can choose among `Defined in Path`, `Splunk DDSS`, and `Splunk SmartStore`.

With any option, in the corresponding **Bucket path** field, provide the path where indexes are stored (that is, your index's parent folder).

For Splunk SmartStore, the full path takes the form:
`<parent_folder>/<indexName>/db/<2-letter-hash>/<2-letter-hash>/<bucket_id_number-origin_guid>/<"guidSplunk"-uploader_guid>/`

For Splunk DDSS, the full path takes the form:
`<parent_folder>/<indexName>`

With either of these options, if you organize your bucket using this default file path, Cribl Search will automatically discover its content.

Splunk SmartStore Time Range on S3

When you select the `Splunk SmartStore` partitioning scheme for all buckets and paths within an Amazon S3 Dataset, the **Dataset Provider** tab adds an **Advanced** section.

Use this section's **Time range** controls to define a rolling, retrospective time window over which Cribl Search will scan your Dataset to optimize search performance against it. The time window is evaluated against event timestamps. Its value defaults to `30 Days` back, and you can set it to anywhere between `1 Day(s)` and `All Time`.

Search Amazon S3

Now that you have a Dataset Provider and Dataset, you're ready to start searching Amazon S3.

Data Transfer Charges

Data transfer to Cribl Search is free if your S3 bucket is in the same AWS Region as your Cribl.Cloud Workspace.

Data transfer to Cribl Search is subject to AWS data transfer costs if your S3 bucket is not in the Regions supported by Cribl.Cloud. AWS also charges API costs in some circumstances. For details, see the Amazon S3 Pricing topic's **Data transfer** section. Cribl Search users are responsible for paying Amazon data transfer and API costs; these are not covered by Cribl.Cloud credits.

Cribl Search will try to minimize the amount of data transferred, to the extend possible, by using path filters, pruning, and other methods.

Other Charges

Other charges may apply. For example, **LIST** and **GET** requests to your S3 buckets will be charged per AWS S3. See the **Requests & data retrievals** section in Amazon S3 Pricing.

Storage Class Compatibility

Cribl Search supports searching Amazon S3 storage classes as listed below.

S3 Storage Class	Cribl Search Compatibility
S3 Standard	Will attempt to read
S3 Intelligent-Tiering	Will attempt to read objects that have **not** aged into one of the unsupported classes below
S3 Express One Zone	Will attempt to read
S3 Standard-IA (infrequent access)	Will attempt to read
S3 One Zone-IA (infrequent access)	Will attempt to read
S3 Glacier Instant Retrieval	Will attempt to read
S3 Glacier Flexible Retrieval	**Unsupported** - will skip
S3 Glacier Deep Archive	**Unsupported** - will skip

When encountering objects from an unsupported storage class, Cribl Search skips them. You'll see the number of such objects next to the total count of search results, under the **skipped** metric:



Search progress

To find out which objects were ignored, go to **Details** > **Logs**, and find the logs containing the word `skipped`.

Search Archived Amazon S3 Objects

To read an object from an unsupported storage class (such as S3 Glacier Deep Archive), first temporarily restore the object in Amazon S3, and then make sure your Cribl Search Dataset is configured to include restored objects.

1. In Amazon S3, restore the archived object. The time this takes depends on the object's size, storage class, and other factors.
2. Once the object is successfully restored, open Cribl Search, and go to **Data** > **Datasets**.
3. Open the Dataset that contains the object, and select the **Storage classes** left tab.
4. Make sure the **Restored from Archive** check box is selected.
5. Select **Save**.

Now, your next search on this Dataset will attempt to read the restored object.

2.3. Connect Cribl Search To Cribl Edge

Configure Cribl Search to query your Cribl Edge Nodes.

With Cribl Edge, you can collect logs, metrics, application data, etc., and send them to Cribl Stream or any other supported destination in real time.

If you already have Edge Nodes running you don't need to set up anything and can start searching. For example searches, see Searching Cribl Edge. To search data not specified in a built-in Dataset, you can add a custom Dataset pointing to anywhere in the filesystem that Edge can read.

In this guide, you'll add a Cribl Edge Node on an Amazon EC2 instance to search its internal logs and metrics. For an in-depth installation guide of Cribl Edge see the Cribl Edge Deployment Guide.

Cribl Edge Access from Cribl Search

Cribl Search users can search the Edge Nodes to which they have access via Cribl Edge Permissions or Roles. Cribl Search Admins and Editors can create Datasets pointing to any Node (for example, using wildcards in the path). However, at search execution time, Cribl Search queries – and returns results from – only the Nodes to which the querying user has access.

For details on how to manage access to Edge Nodes, see the Cribl Edge docs:

- Access Management
- Product-Level Permissions (Stream and Edge)
- Group- and Fleet-Level Permissions

Configure Cribl Edge

To add a new Edge Node, follow the instructions in Add and Update Edge Nodes to add or bootstrap a new Node.

Configure EC2 Instance

We'll launch an Ubuntu instance using the AWS Management Console. For an in-depth tutorial, see how to get started with Amazon EC2 Linux instances.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the EC2 console dashboard, in the **Launch instance** box, select **Launch instance**. From the options that appear, select **Launch instance** again.

3. Under **Name and tags**, for **Name**, enter a descriptive name for your instance.
4. Under **Application and OS Images (Amazon Machine Image)**, select **Quick Start**, and then select **Ubuntu**. This is the operating system (OS) for your instance.
5. Under **Key pair (login)**, for **Key pair name**, select or create a key pair.
6. Keep the default selections for the other configuration settings for your instance.
7. Select **Launch instance**.
8. A confirmation page lets you know that your instance is launching. Select **View all instances** to close the confirmation page and return to the console.
9. On the **Instances** screen, you can view the status of the launch. It takes a short time for an instance to launch. When you launch an instance, its initial state is pending. After the instance starts, its state changes to running and it receives a public DNS name.
10. It can take a few minutes for the instance to be ready for you to connect to it. Check that your instance has passed its status checks; you can view this information in the **Status check** column.
11. Now you'll connect to the instance as the root user using the Amazon EC2 console. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. See how to connect using EC2 Instance Connect for more details.
12. In the navigation pane, select **Instances**, then the instance, then **Connect**.
13. Select the **EC2 Instance Connect** tab and verify the user name, then select **Connect** to open a terminal window.
14. In the terminal, to gain root access, run the command `sudo su`.
15. In the terminal as root, paste the bootstrap script you copied from Cribl Edge in the above section.
16. This will install the Edge Node in your EC2 instance and you can navigate back to Cribl Edge's **Edge Nodes** tab to view the new Node.

Built-In Cribl Edge Datasets

Cribl Search comes with several Cribl Edge Datasets, detailed in Built-In Cribl Edge Datasets. Each Dataset tells Cribl Search what data to search.

You do not need to make any changes to these Datasets, and you can start searching them immediately. However, you can edit these built-in Datasets, or add new ones to specify other logs anywhere in the filesystem that Edge can read.

To access Cribl Edge Datasets, select **Data** from the sidebar, then look for them on the default **Datasets** tab. The following sections explain the configurations available on a Dataset.

Path

The **Path** specifies the data that the Dataset consists of. It defines the scope of data, to narrow down what data is in the Dataset. This is a JavaScript expression that supports environment variables, wildcards, and tokens. For example,

- `$CRIBL_STATE_DIR/system_metrics`
- `/foo/B*/*/baz`
- `/foo/${country}/${state}/${city}/` – where `country`, `state`, and `city` now become fields for all events of that Dataset.

Basic Tokens

Basic tokens' syntax follows that of JS template literals: `${token_name}` – where `token_name` is the field (name) of interest.

For example, if the path was set to `/var/log/${hostname}/${dataSource}/`, you could use a filter such as `hostname=='myHost' && dataSource=='mydataSource'` to specify data only from the `/var/log/myHost/mydataSource/` subdirectory.

Time-Based Tokens

Paths with time notation can be referenced with tokens, having a direct effect on the earliest and latest boundaries. The supported time fields are:

- `_time` is the raw event's timestamp.
- `__earliest` is the search start time.
- `__latest` is the search end time.

Time-based tokens are processed as follows:

- For each path, times must be notated in descending order. So Year/Month/Day order is supported, but Day/Month/Year is not.
- Paths may contain more than one time component. For example, `/my/path/2020-04/20/`.
- In a given path, each time component can be used only once.
  So `/my/path/${_time:%Y}/${_time:%m}/${_time:%d}/...` is a valid expression format, but `/my/path/${_time:%Y}/${_time:%m}/${host}/${_time:%Y}/...` (with a repeated `Y`) is not supported.
- For each path, all extracted dates/times are considered in UTC.



Cribl recommends that your path always include and tokenize the largest available time fields, proceeding down to the smallest desired time fields. Otherwise, your searches might yield unexpected results, because Cribl Search will default the omitted fields to their earliest allowed value.

The following `strptime` format components are allowed:

- `Y`, `y` for years
- `m`, `B`, `b`, `e` for months
- `d`, `j` for days
- `H`, `I` for hours
- `M` for minutes
- `S` for seconds
- `s` for Unix-style Epoch times (seconds since 1/1/1970)

Time-based token syntax follows that of a slightly modified JS template literal: `${_time: some_strptime_format_component}`. Examples:

Path	Matches
`/path/${_time:%Y}/${_time:%m}/${_time:%d}/...`	`/path/2020/04/20/...`
`/path/${_time:year=%Y}/${_time:month=%m}/${_time:day=%d}/...`	`/path/year=2020/month=`
`/path/${_time:%Y-%m-%d}/...`	`/path/2020-05-20/...`

Path Filter

This is a JavaScript filter expression that is evaluated against the provided Dataset path. The **Path Filter** value defaults to `true`, which matches all data, but this value can be customized almost arbitrarily.

For example, if a Dataset has this **Path Filter**:

```
source.endsWith('.log') || source.endsWith('.txt')
```

...then only files/objects with `.log` or `.txt` extensions will be searched.

At the **Path Filter** field's right edge are a Copy button and an Expand button that opens a validation modal.

With DDSS or SmartStore partitioning, you can provide the index name as **Path Filter**, for example: `index=="index_y"`.

Partitioning Scheme

Advanced Settings > **Partitioning scheme** lets you select the scheme for partitioning data incoming from Splunk.

You can choose between `Splunk DDSS` and `Splunk SmartStore`.

In both cases, provide the path where indices are stored (parent folder of your index) as **Bucket path**.

For Splunk DDSS, the full path takes the form:

```
<parent_folder>/<indexName>/db/db_<latestTime>_<earliestTime>_<bucketId>/rawdata/jou
```

or

```
<parent_folder>/<indexName>/db_<latestTime>_<earliestTime>_<bucketId>/rawdata/journa
```

For SmartStore, it is:

```
<parent_folder>/<indexName>/db/<2-letter-hash>/<2-letter-
hash>/<bucket_id_number-origin_guid>/<"guidSplunk"-uploader_guid>/
```

In both cases, if your bucket is organized using this default file path, Cribl Search will automatically discover its content.

Processing

Processing is done with **Datatypes**, to break data into discrete events and define fields so they're ready to search. These consist of rules that are applied to the data searched in your Dataset. See Datatypes for details.

Search Cribl Edge

Now that you have an Edge Node on an Amazon EC2 instance, you're ready to start searching Cribl Edge.

2.4. Connect Cribl Search To AWS

Link Cribl Search to your AWS data.

Grant Access to AWS

Allow Cribl Search to access to your AWS data, by configuring an IAM role or user. See various trust policy examples. See: Grant Access to AWS.

Amazon Security Lake

Configure Cribl Search to query your Amazon Security Lake data. See: Amazon Security Lake.

AWS API

Configure Cribl Search to query an AWS API endpoint. See: AWS API.

Data Lake Amazon S3

Configure Cribl Search to query Cribl Stream's Data Lake Amazon S3 Destination. See: Data Lake Amazon S3.

2.4.1. Grant Access To AWS

Allow Cribl Search to access to your AWS data, by configuring an IAM role or user. See various trust policy examples.

Cribl Search uses Amazon Web Services (AWS) IAM to access and search your data in AWS. When creating an Amazon Dataset Provider in Cribl Search, you'll need to provide it with credentials.

Access to AWS services is managed through AWS Identity and Access Management (IAM). You can grant access either at the IAM role level, or at the IAM user level using an access and secret key pair. Cribl recommends role-based access, as described below.

IAM User

You can use an IAM user's access keys (access key IDs and secret access keys) to grant access to your data in AWS. If creating a new IAM user, give it **Programmatic access** so AWS assigns the user access keys.

IAM Role

Cribl recommends using IAM role–based access for increased security. An IAM role is similar to a user – it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of passing IAM user credentials, here a user assumes a role and triggers AWS to dynamically generate temporary security credentials. With these temporary credentials, the user then accesses an AWS resource by its Amazon Resource Name (ARN).

> For details about configuring the IAM role in the AWS console, see Create a role using custom trust policies. For details about using permissions policies to restrict an IAM role's capabilities on AWS, see Permissions boundaries for IAM entities.

The process for granting access is:

1. Set up a new AWS IAM role in your account.

   You need to either build the IAM role and policy using your own infrastructure-management solution (such as Terraform or CloudFormation), or else build it via the AWS console.

   You can create a new role with Cribl's trust policy instead of attaching it – see the next step.

2. Attach the trust policy, provided by Cribl, to the role in your account.

   Modify your IAM role to include Cribl Search's trust policy. You can do this with the AWS console, CLI, or API.

3. Add the appropriate permissions to your IAM role using an identity-based policy. For example, to allow access to an S3 bucket called `cribl-search-data`, you'd use the following sample **permission policy**:

```
{
"Version": "2012-10-17",
"Statement": [{
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::cribl-search-data/*"
   },
   {
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::cribl-search-data"
   }
]
}
```

Copy

4. Specify the IAM role's ARN on an AWS Dataset Provider.

Trust Policy

Cribl Search builds a trust policy that creates a trust relationship between Cribl Search and a role in your AWS account. This policy controls and restricts access to only those resources to which you grant access. This trust relationship can be revoked at any time, without having to also revoke credentials (and thus, revoke access to the data).

The provided trust policy contains the AWS Role ARN and external ID of Cribl Search's IAM role that you'll need to trust. The policy allows Cribl Search to perform the `sts:AssumeRole` operation toward your external account.

The external ID is a custom string that you create to enhance authentication and authorization when accessing the S3 resource. The crucial requirement is to ensure that the strings match on both the Cribl Dataset Provider the and AWS trust policy.



The external ID must meet the length and character requirements that AWS specifies in IAM name requirements.

Example Trust Policy

Here is a basic trust policy – an anonymized version of the policy you can copy from your Dataset Provider modal's **Trust Policy** button (when setting up the Authentication Method). See also the variations below the example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<TENANT_ID>:role/search-exec-main"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<EXTERNAL_ID>"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<TENANT_ID>:role/search-exec-main"
      },
      "Action": "sts:TagSession"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<TENANT_ID>:role/search-exec-main"
      },
      "Action": "sts:SetSourceIdentity"
    }
  ]
}
```

Copy

Trust Policy with Multiple Workspaces

If you have configured multiple workspaces, replace `<TENANT_ID>:role/search-exec-main` above with `<TENANT_ID>:role/search-exec-<WORKSPACE-NAME>`, as needed.

Alternatively, you can use this wildcard syntax to enable searching all workspaces. But use wildcards with caution, to avoid creating an overly permissive policy:

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "*"
  },
```

Copy

Trust Policy with ABAC Tagging

You can make your AssumeRole sessions more secure by adding Cribl's VPCE (Virtual Private Cloud Endpoint) ID to your trust policy. This enables attribute-based access control (ABAC).

For an overview of how ABAC works on AWS, with links to further resources, see Define permissions based on attributes with ABAC authorization.

To set up ABAC on a supported Dataset Provider: After you enable the **Apply ABAC source-ip tagging** option, copy the adjacent **VPCE ID** to use later in this section. For convenience, we'll start with the default trust policy shown above:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<TENANT_ID>:role/search-exec-main"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<EXTERNAL_ID>"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<TENANT_ID>:role/search-exec-main"
      },
      "Action": "sts:TagSession"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<TENANT_ID>:role/search-exec-main"
      },
      "Action": "sts:SetSourceIdentity"
    }
  ]
}
```

Copy

Expand the Trust Policy

First, expand the `"Action": "sts:TagSession"` element by adding the sub-element shown below. Paste in the `"${aws:SourceIp}"` variable as-is. This enables passing Cribl Search's `source-ip` value, making your sessions more secure by enforcing a condition that the `source-ip` tag must match the IP address requesting to assume the role:

```
    "Action": "sts:TagSession",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/source-ip": "${aws:SourceIp}"
        }
    }
```

Copy

Expand the Permission Policy

For the remaining steps, we'll start with this anonymized default policy that you can copy from your Dataset Provider modal's **Permission Policy** button (when setting up the Authentication Method).

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": ["arn:aws:s3:::<MY_S3_BUCKET>/*"]
      },
      {
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": ["arn:aws:s3:::<MY_S3_BUCKET>"]
      }
    ]
  }
```

Copy

Expand the `s3:ListBucket` element by adding the sub-element shown below. Replace `vpce-<someVpceId>` with the VPCE ID that you copied from your Dataset Provider's config modal:

```
{
 "Effect": "Allow",
 "Action": "s3:ListBucket",
 "Resource": ["arn:aws:s3:::<MY_S3_BUCKET>"],
 "Condition": {
     "StringEquals": {
         "aws:SourceVpce": "vpce-<someVcpeId>"
     }
 }
}
```

Copy

Finally, copy the whole expanded `s3:ListBucket` element that you just modified. In the duplicated element shown below, paste in the `"${aws:SourceIp}"` variable as-is.

```
{
 "Effect": "Allow",
 "Action": "s3:ListBucket",
 "Resource": ["arn:aws:s3:::<MY_S3_BUCKET>"],
 "Condition": {
     "StringEquals": {
         "aws:SourceVpce": "vpce-<someVcpeId>"
     }
 }
},
{
 "Effect": "Allow",
 "Action": "s3:ListBucket",
"Resource": ["arn:aws:s3:::<MY_S3_BUCKET>"],
 "Condition": {
     "StringEquals": {
         "aws:PrincipalTag/source-ip": "${aws:SourceIp}"
     }
 }
}
```

Copy

Complete Permission Policy

To wrap up, you'll end up with this expanded permission policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": ["arn:aws:s3:::<MY_S3_BUCKET>/*"]
      },
      {
       "Effect": "Allow",
       "Action": "s3:ListBucket",
       "Resource": ["arn:aws:s3:::<MY_S3_BUCKET>"],
       "Condition": {
           "StringEquals": {
               "aws:SourceVpce": "vpce-<someVcpeId>"
           }
       }
      },
      {
       "Effect": "Allow",
       "Action": "s3:ListBucket",
    "Resource": ["arn:aws:s3:::<MY_S3_BUCKET>"],
       "Condition": {
           "StringEquals": {
               "aws:PrincipalTag/source-ip": "${aws:SourceIp}"
           }
       }
      }
  ]
}
```

Copy

2.4.2. Connect Cribl Search To Amazon Security Lake

Configure Cribl Search to query your Amazon Security Lake data.

Using Amazon Security Lake, you can automatically consolidate security data from across AWS environments, SaaS providers, on-prem systems, and cloud sources into a purpose-built data lake.

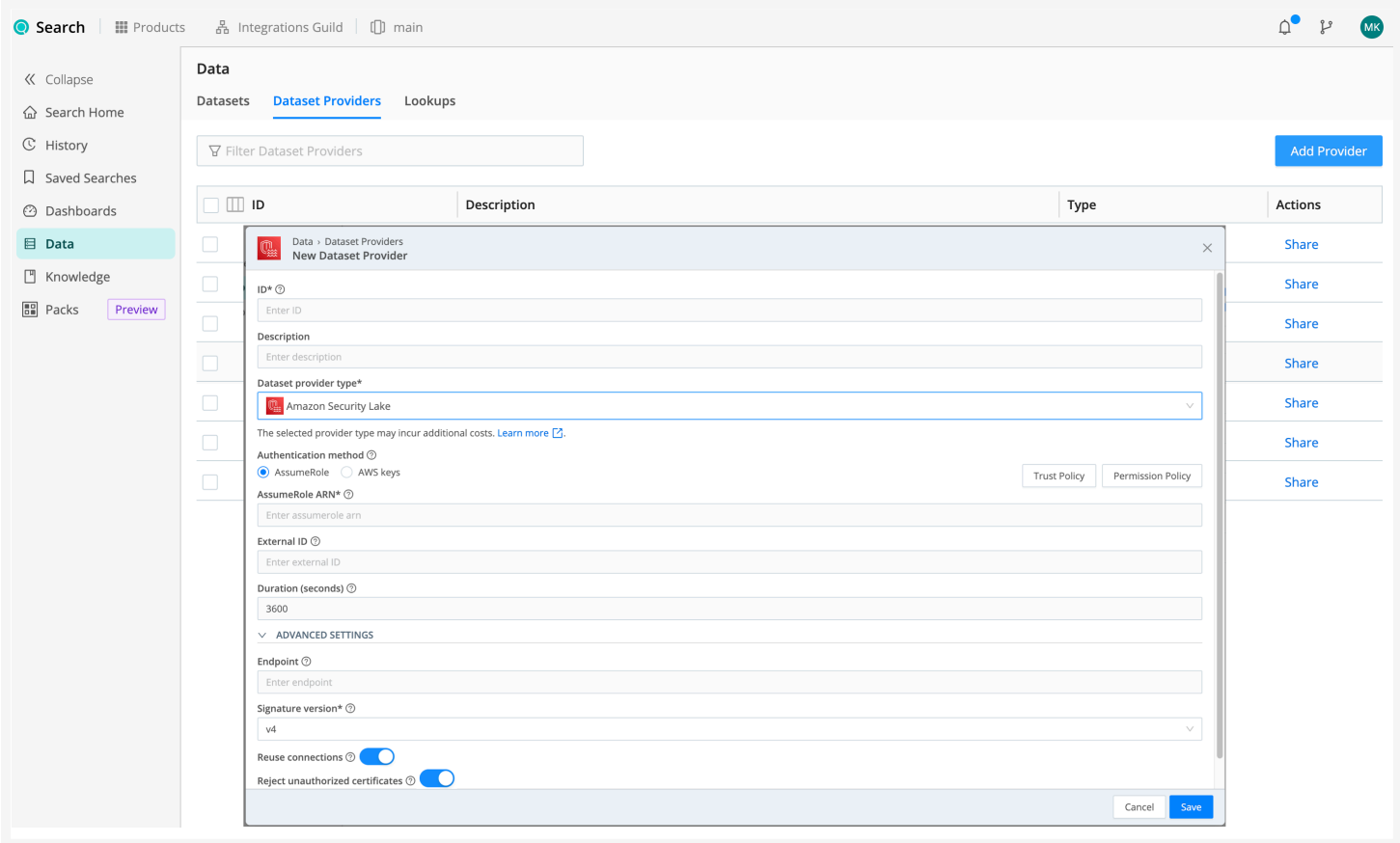
In this guide, you'll set up a Dataset Provider and a Dataset to search objects in your Amazon Security Lake.

Data transfer costs and other charges may apply. For details, see the Data Transfer Charges section below.

Add an Amazon Security Lake Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an Amazon Security Lake Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.



Amazon Security Lake Dataset Provider selection

Basic Configuration

Configure the **New Dataset Provider** modal as follows:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional. Here, you can enter a summary that will clarify this Dataset Provider's purpose to other users.
3. Set the **Dataset Provider type** to `Amazon S3`.
4. **Authentication method** provides two options, **Assume Role** and **AWS keys**. For details on configuring each option, see Authentication Method below.

Advanced Settings

This section provides the following optional configurations:

- **Endpoint**: S3 service or compatible endpoint. If empty, defaults to AWS Region-specific endpoint.
- **Signature version**: Signature version to use for signing S3 requests. Defaults to `v4`.
- **Reuse connections**: Whether to reuse connections between requests. Toggling on (default) can improve performance.
- **Reject unauthorized certificates**: Whether to reject certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.
- **Apply ABAC source-ip tagging**: Displayed only if you've set the **Authentication method** to **AssumeRole**. For details, see ABAC Tagging below.

Wrapping Up

Select **Save** when your configuration is complete.

Authentication Method

Your choices here are **AssumeRole** or **AWS keys**. For rich details on both options, beyond the configuration basics in this section, see Grant Access to AWS.

AssumeRole Authentication

Selecting **AssumeRole** requires the IAM role's ARN (**AssumeRole ARN**), and also exposes the following options:

- The **External ID** on this Dataset Provider must match the external ID defined in the IAM Role Trust Policy.
- The **Duration (seconds)** defines how long the AssumeRole's session lasts. Minimum is `900` (15 minutes), default is `3600` (1 hour), and maximum is `43200` (12 hours).
- The ABAC tagging controls described in the next section.

To the right, select the **Trust Policy** and **Permission Policy** buttons to see the default policies that you can copy and expand, so you don't have to write them from scratch. For more information, see Example Trust Policy and Expand the Permission Policy.

AWS Keys Authentication

Selecting **AWS keys** requires the IAM user's **Access key** and **Secret key**.

To the right, select the **Permission Policy** button to see the default policy that you can copy and expand, so you don't have to write it from scratch. For more information, see Expand the Permission Policy.

ABAC Tagging

Select **Apply ABAC source-ip tagging** if you want to make your AssumeRole sessions more secure by enabling attribute-based access controls (ABAC). By enabling this option, you can hard-code Cribl's Virtual Private Cloud Endpoint (VPCE) into your IAM trust policy.

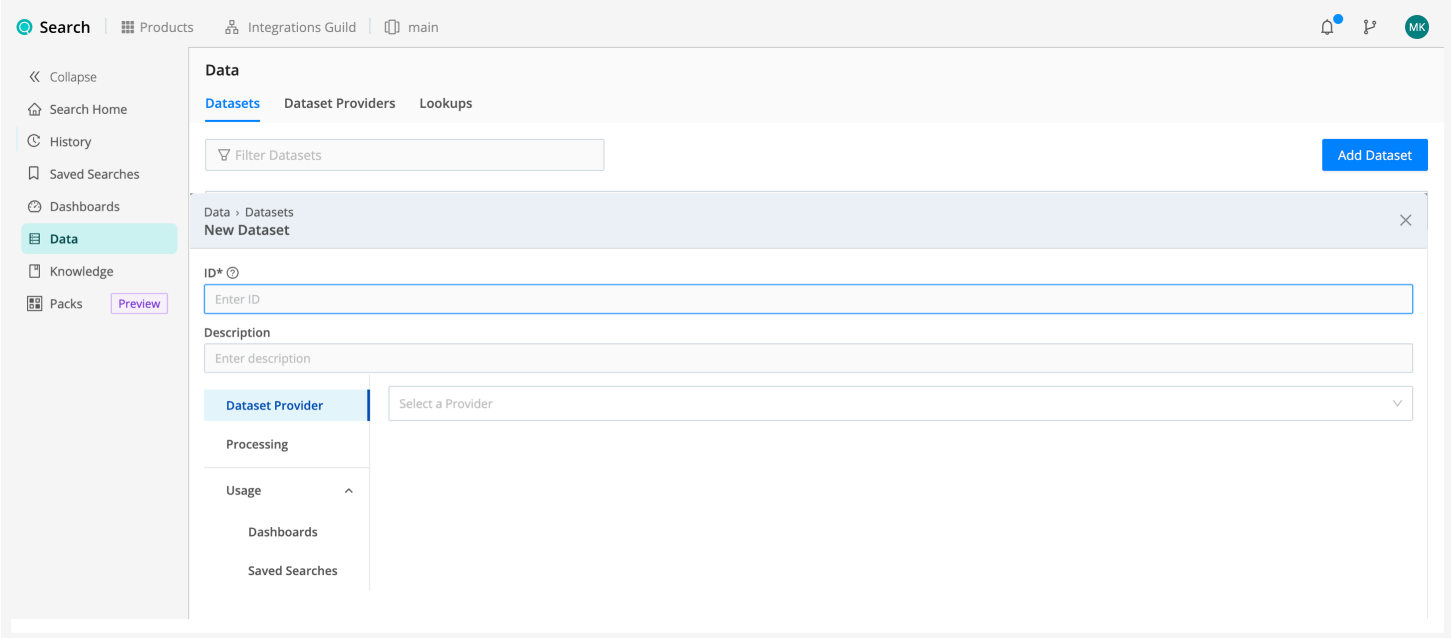
Selecting this option exposes **VPC Endpoint IDs**. Use the adjacent Copy button to transfer the VPCE ID to your IAM trust policy, as shown in Trust Policy with ABAC Tagging.

Completing both parts of this configuration will make your credentials harder to impersonate. Specifically, your AssumeRole sessions will include a `source-ip` tag, enabling you to enforce a condition that `aws:RequestTag/source-ip` equals `${aws:SourceIp}`.

Add an Amazon Security Lake Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.



Datasets selection

Configure the **New Dataset** modal as follows.

Identify the Dataset

Use the first few fields to uniquely identify this Dataset and specify its type:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of an Amazon Security Lake Dataset Provider.

Set the Path and Filter

Define the Dataset's bucket path and filtering:

1. **Path** is the path within the S3 buckets you'd like to search. Tokens and key-value pairs are supported. The path is prefilled with a recommended path string, that you can use for guidance. For details, see Set Path for Amazon Security Lake.
2. **Path Filter** is a JavaScript filter expression that is evaluated against the S3 bucket path. It defaults to `true`, which matches all data, but you can customize this value.

Set Up Buckets and Regions

Under **Selected buckets**, define at least one S3 bucket/Region pair for this Dataset to search:

1. Select **Add bucket** to add the first pair. In the resulting table row:
2. **Bucket name** is the name of the S3 bucket.
3. **Region** is the AWS Region where the bucket is located.

Select **Add bucket** again for each additional bucket/Region pair you want to append to the table.

Set Up Datatypes

1. On the **Processing** left tab, you specify **Datatypes** to break data down into discrete events and define fields so they're ready to search. Use the drop-down to set the first rule to `OCSF Datatypes`. For details, see Datatypes.
2. Optionally, click **Add Datatype Ruleset** to define more rulesets – each of which contains rules applied to the data searched in your Dataset.

3. Create a rule on a Datatype that converts the time field from milliseconds to seconds (For example, `_time = time/1000`). For details, see the Assign `time` Field section below.

Save the Configuration

Bypass the **Usage** left tabs for now, and select **Save** when your configuration is complete.

Path/Filter Considerations

This section offers details and guidance on configuring the search path, filter, and time extraction.

Set Path for Amazon Security Lake

The **Path** specifies the data that the Dataset consists of. It defines the scope of data, to narrow down what data is in the Dataset. This is a JavaScript expression that supports tokens and key-value pairs. For example,

- `path/to/folder/${data}/` – where `data` becomes a field for all events of that Dataset.
- `path/to/folder/${data}/${*}` – where `data` and the wildcarded path becomes a field for all events of that Dataset.

Basic Tokens

Basic tokens' syntax follows that of JS template literals: `${token_name}` – where `token_name` is the field (name) of interest.

For example, if the path was set to `/var/log/${hostname}/${dataSource}/`, you could use a filter such as `hostname=='myHost' && dataSource=='mydataSource'` to specify data only from the `/var/log/myHost/mydataSource/` subdirectory.

Time-Based Tokens

Paths with time notation can be referenced with tokens, having a direct effect on the earliest and latest boundaries. The supported time fields are:

- `_time` is the raw event's timestamp.
- `__earliest` is the search start time.
- `__latest` is the search end time.

Time-based tokens are processed as follows:

- For each path, times must be notated in descending order. So Year/Month/Day order is supported, but Day/Month/Year is not.
- Paths may contain more than one time component. For example, `/my/path/2020-04/20/`.

- In a given path, each time component can be used only once.
  So `/my/path/${_time:%Y}/${_time:%m}/${_time:%d}/...` is a valid expression format, but `/my/path/${_time:%Y}/${_time:%m}/${host}/${_time:%Y}/...` (with a repeated `Y`) is not supported.
- For each path, all extracted dates/times are considered in UTC.



Cribl recommends that your path always include and tokenize the largest available time fields, proceeding down to the smallest desired time fields. Otherwise, your searches might yield unexpected results, because Cribl Search will default the omitted fields to their earliest allowed value.

The following `strptime` format components are allowed:

- `Y`, `y` for years
- `m`, `B`, `b`, `e` for months
- `d`, `j` for days
- `H`, `I` for hours
- `M` for minutes
- `S` for seconds
- `s` for Unix-style Epoch times (seconds since 1/1/1970)

Time-based token syntax follows that of a slightly modified JS template literal: `${_time: some_strptime_format_component}`. Examples:

Path	Matches
`/path/${_time:%Y}/${_time:%m}/${_time:%d}/...`	`/path/2020/04/20/...`
`/path/${_time:year=%Y}/${_time:month=%m}/${_time:day=%d}/...`	`/path/year=2020/month=`
`/path/${_time:%Y-%m-%d}/...`	`/path/2020-05-20/...`

Set Path Filter for Amazon Security Lake

The **Path filter** is a JavaScript filter expression that Cribl Search evaluates against the corresponding **Container path**. The **Path filter** field's value defaults to `true`, which matches all data. However, you can customize this value almost arbitrarily.

For example, if a Dataset has this **Filter**:
`source.endsWith('.log') || source.endsWith('.txt')`
...then only files/objects with `.log` or `.txt` extensions will be searched.

At the **Path filter** field's right edge are a Copy button and an Expand button that opens a validation modal.

Assign `time` Field for Amazon Security Lake

Amazon Security Lake automatically partitions incoming data from natively supported AWS services into storage-efficient and query-efficient Parquet files. Additionally, it transforms data from native AWS services into the open-source OCSF format.

The time is assigned to the `time` field, which is an epoch time expressed in milliseconds.

To use this field, you need to set its value to the Cribl Search `_time` field. Because Cribl Search uses seconds instead of milliseconds, you must also convert it by dividing by 1,000. For this to work, add this field to a Datatype:
`_time = time/1000.`

Search Amazon Security Lake

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

Data Transfer Charges

Data transfer to Cribl Search is free if your S3 bucket is in the same Region as your Cribl.Cloud Workspace.

Data transfer to Cribl Search is subject to AWS data transfer costs if your S3 bucket is not in the Regions supported by Cribl.Cloud. See the **Data transfer** section in Amazon S3 Pricing.

Cribl Search will try to minimize the amount of data transferred, to the extend possible, by using path filters, pruning, and other methods.

Other Charges

Other charges may apply. For example, **LIST** and **GET** requests to your S3 buckets will be charged per AWS S3. See the **Requests & data retrievals** section in Amazon S3 Pricing.

2.4.3. Connect Cribl Search To AWS API

Configure Cribl Search to query an AWS API endpoint.

Amazon Web Services (AWS) offers scalable and cost-effective cloud computing solutions.

In this guide, you'll set up a Dataset Provider and a Dataset to search the AWS API supporting the following endpoints:

Product	Endpoints
EC2	EC2 Instances, EC2 Volumes, EC2 Security Groups
Lambda	Lambda Functions
IAM	Users, Roles, Groups, Policies, MFA Devices
CloudFormation	StackSets, Stacks, Exports
DynamoDB	Backups
RDS	Clusters, Cluster Endpoints, Instances, Security Groups, Certificates
CloudTrail	Events
VPC	VPCs, Subnets, Network Interfaces
EFS	File Systems

Add an AWS API Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an AWS API Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **AWS API**.

4. **Authentication method** provides two options, **Assume Role** and **AWS keys**. See how to grant access to AWS for details on each option.
5. Select **Add Configuration** to specify your AWS account. The configuration depends on the **Authentication method** selected and you can use only one method for all configurations. In the **Account Configurations** table:
    - **Assume Role** requires the IAM role's ARN (**AssumeRole ARN**) and has options to define an **External ID** and **Duration**.
        - The **External ID** on the Dataset Provider must match the external ID defined in the IAM Role Trust Policy.
        - **Duration** defines the Assumed Role's session length of time, in seconds. Minimum is `900` (15 minutes), default is `3600` (1 hour), and maximum is `43200` (12 hours).
    - **AWS keys** requires the IAM user's account **Name**, **Access key**, and **Secret key**.
6. Select **Save** when finished.

For details on obtaining your AWS credentials, see Grant Access to AWS.

Permission Requirements for AWS API

Accessing specific AWS endpoints requires the following permissions:

Endpoint	Permission

`ec2_instances`
`ec2_volumes`
`ec2_security_groups`
	`ec2:DescribeInstances`
	`ec2:DescribeVolumes`
	`ec2:DescribeSecurityGroups`
	
	or
	
	`ec2:Describe*`

`lambda_functions`	`lambda:ListFunctions`

`iam_users`
`iam_groups`
`iam_policies`
`iam_roles`
`iam_mfa_devices`
	`iam:ListPolicies`
	`iam:ListRoles`
	`iam:ListUsers`
	`iam:ListGroups`
	`iam:ListMFADevices`

`cloudformation_stacks`
`cloudformation_stacksets`
`cloudformation_exports`
	`cloudformation:ListExports`
	`cloudformation:ListStacks`
	`cloudformation:ListStackSets`

`dynamodb_backups`	`dynamodb:ListBackups`

Endpoint	Permission

`rds_clusters`
`rds_cluster_endpoints`
`rds_instances`
`rds_security_groups`
`rds_certificates`
	`rds:DescribeDBInstances`
	`rds:DescribeDBClusterEndpoints`
	`rds:DescribeDBSecurityGroups`
	`rds:DescribeCertificates`
	`rds:DescribeDBClusters`
	
	or
	
	`rds:Describe*`

`cloudtrail_events`	`cloudtrail:LookupEvents`

`vpc_vpcs`
`vpc_subnets`
`vpc_network_interfaces`
	`ec2:DescribeNetworkInterfaces`
	`ec2:DescribeVpcs`
	`ec2:DescribeSubnets`

`efs_file_systems`	`elasticfilesystem:DescribeFileSystems`

Add an AWS API Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of an AWS Dataset Provider.
4. Under **Enabled endpoint**, select **Add Endpoints** to select the endpoints for your Dataset. Select an endpoint from the drop-down menu. Your options are:
    - `ec2_instances`
    - `ec2_volumes`
    - `ec2_security_groups`
    - `lambda_functions`
    - `iam_users`
    - `iam_roles`
    - `iam_groups`

- iam_policies
- iam_mfa_devices
- cloudformation_stacks
- cloudformation_stacksets
- cloudformation_exports
- dynamodb_backups
- rds_exports
- rds_backups
- rds_clusters
- rds_cluster_endpoints
- rds_instances
- rds_security_groups
- rds_certificates
- cloudtrail_events
- vpc_subnets
- vpc_network_interfaces
- efs_file_systems

5. Under **AWS Regions**, select **Add Regions** to specify the AWS Regions to query for the endpoint(s).
6. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
7. In **Snapshots**, you can set up API Snapshots.
8. Select **Save** when finished.

Search AWS API

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

2.4.4. Connect Cribl Search To Data Lake Amazon S3

Configure Cribl Search to query Cribl Stream's Data Lake Amazon S3 Destination.

Cribl Stream's Data Lake Amazon S3 Destination sends data to Amazon S3 with a partitioning scheme tailored for Cribl Search. This scheme optimizes how objects are searched relative to the search's time range.

In this guide, you'll set up a Dataset Provider and a Dataset to search objects that Cribl Stream has exported to your Amazon S3 bucket via a Data Lake Amazon S3 Destination.

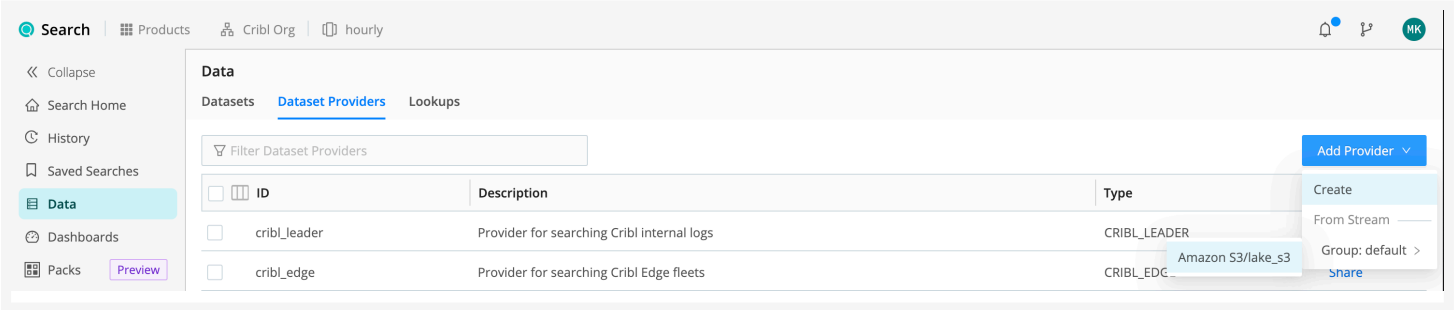
Data transfer costs and other charges might apply. For details, see the Data Transfer Charges section below.

Add a Data Lake Amazon S3 Dataset Provider

A Dataset Provider tells Cribl Search where to query, and contains access credentials. Here, you will add an Amazon S3 Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

From the resulting drop-down, select the Cribl Stream Worker Group and the Data Lake Amazon S3 Destination you want to search.



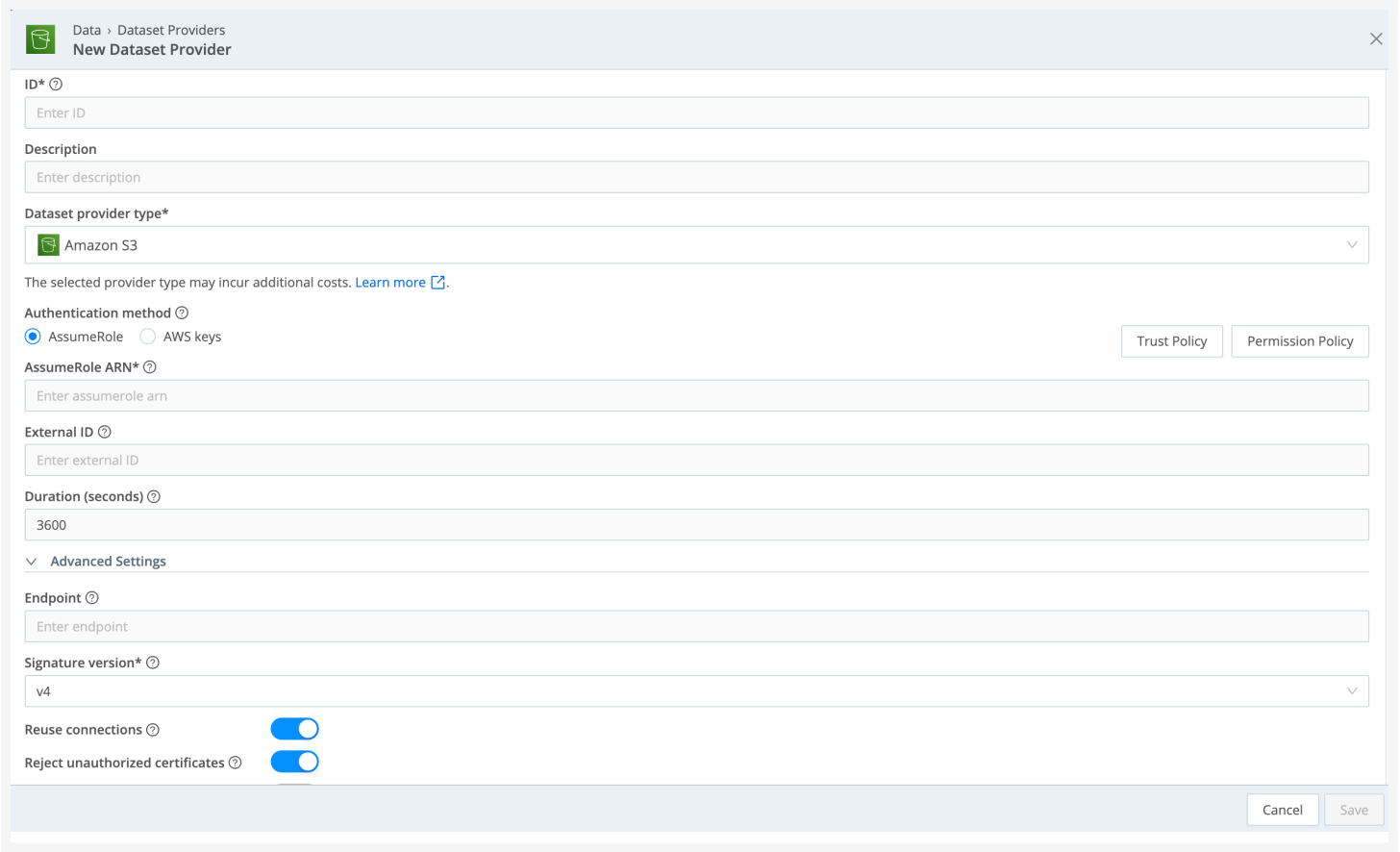
Creating a Data Lake Amazon S3 Dataset Provider

Basic Configuration

Configure the resulting **New Dataset Provider** modal as follows:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional. Here, you can enter a summary that will clarify this Dataset Provider's purpose to other users.
3. Set the **Dataset Provider type** to `Amazon S3`.

4. **Authentication method** is automatically populated with the credentials used by the Data Lake Amazon S3 Destination. You can use different credentials if desired – you have two options, **Assume Role** and **AWS keys**. For details on configuring each option, see Authentication Method below.



Configuring a Data Lake Amazon S3 Dataset Provider

Advanced Settings

This section provides the following optional configurations:

- **Endpoint**: S3 service or compatible endpoint. If empty, defaults to AWS Region-specific endpoint.
- **Signature version**: Signature version to use for signing S3 requests. Defaults to `v4`.
- **Reuse connections**: Whether to reuse connections between requests. Toggling on (default) can improve performance.
- **Reject unauthorized certificates**: Whether to reject certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Wrapping Up

Select **Save** when your configuration is complete.

Authentication Method

Your choices here are **AssumeRole** or **AWS keys**. For rich details on both options, beyond the configuration basics in this section, see Grant Access to AWS.

AssumeRole Authentication

Selecting **AssumeRole** requires the IAM role's ARN (**AssumeRole ARN**), and also exposes the following options:

- The **External ID** on this Dataset Provider must match the external ID defined in the IAM Role Trust Policy.
- The **Duration (seconds)** defines how long the AssumeRole's session lasts. Minimum is `900` (15 minutes), default is `3600` (1 hour), and maximum is `43200` (12 hours).
- The ABAC tagging controls described in the next section.

To the right, select the **Trust Policy** and **Permission Policy** buttons to see the default policies that you can copy and expand, so you don't have to write them from scratch. For more information, see Example Trust Policy and Expand the Permission Policy.

AWS Keys Authentication

Selecting **AWS keys** requires the IAM user's **Access key** and **Secret key**.

To the right, select the **Permission Policy** button to see the default policy that you can copy and expand, so you don't have to write it from scratch. For more information, see Expand the Permission Policy.

Permissions for Data Lake Amazon S3

The following permissions are needed to read from an Amazon S3 bucket:

- `s3:GetObject`
- `s3:ListBucket`

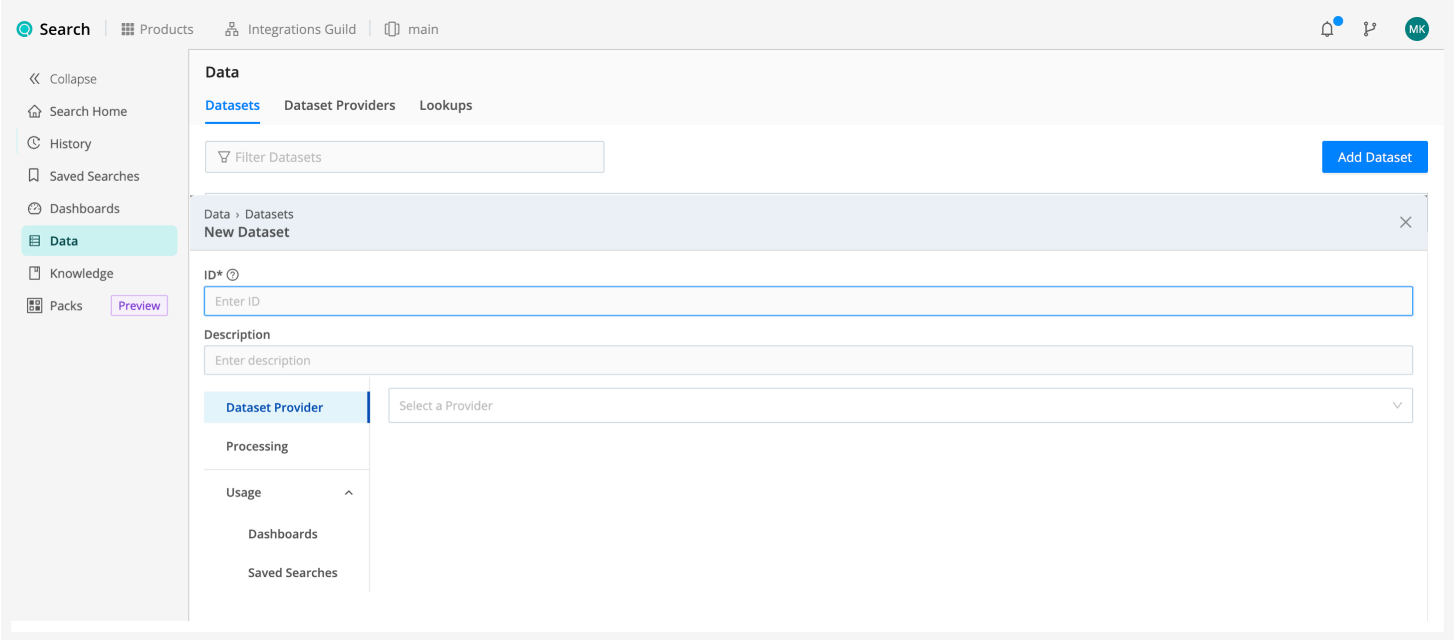
If you're using the same IAM policy for this Dataset as the Data Lake Amazon S3 Destination, you need to provide permissions to read and write:

- `s3:GetObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

Add a Data Lake Amazon S3 Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.



Dataset configuration

Configure the **New Dataset** modal as follows.

Identify the Dataset

1. **ID** is an identifier unique both in Cribl Search and in your Amazon S3 data lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional. Here, you can enter a summary that will clarify this Dataset's purpose to other users.
3. Set **Dataset Provider** to the Amazon S3 Dataset Provider (its **ID**) that you created in the above Add a Data Lake Amazon S3 Dataset Provider section.

Set Up Path and Region

Configure the path and Region as follows:

1. **Bucket path** is the path to the S3 objects you'd like to search. Through Cribl Stream's Data Lake Amazon S3 Destination, data is partitioned specifically for Cribl Search and sent to Amazon S3. A time-based search scheme optimizes how objects are searched.

   The **Bucket path** is automatically populated to use this scheme, starting with the bucket name followed by time tokens: `<my-bucket-name>/${*}-${_time:%Y}/${*}-${_time:%m}/${*}-${_time:%d}/${*}-${_time:%H}`.

   Tokens and key-value pairs are supported, see bucket path for details.

2. **Path filter** is a JavaScript filter expression that is evaluated against the S3 bucket path. Defaults to `true`, which matches all data, but you can customize this value.
3. Toggle off **Auto-detect region** if you want to use the adjacent **Region** drop-down to specify the AWS Region where the S3 bucket is located. When toggled on, Cribl automatically detects the Region and locks the drop-down.

   The Amazon S3 Dataset type supports multiple path/filter/region configurations per bucket. For details about this option, see Multiple Path Configs.

Set Up Datatypes and Storage Classes

On the **Processing** left tab, you specify **Datatypes** to break data down into discrete events and define fields so they're ready to search.

Use the drop-down to set the first rule to `AWS Datatypes`. Optionally, click **Add Datatype Ruleset** to select more rulesets – each of which contains rules applied to the data searched in your Dataset. For details, see Datatypes.

On the **Storage classes** left tab, select the storage classes that you want this Dataset to search against. You can minimize retrieval costs by selecting only warmer classes. You must select at least one class. For details, see Storage Classes.

Save the Configuration

Bypass the **Usage** left tabs for now, and select **Save** when your configuration is complete.

Path/Filter Considerations

This section offers details and guidance on configuring search paths and filters.

Bucket Path

The **Bucket path** specifies the data that the Dataset consists of. It defines the scope of data, to narrow down what data is in the Dataset. This is a JavaScript expression that supports tokens and key-value pairs. For example,

- `my-bucket/${data}/` – where `data` becomes a field for all events of that Dataset.
- `my-bucket/${data}/${*}` – where `data` and the wildcarded path becomes a field for all events of that Dataset.

To search partitioned data in S3, the bucket path includes the partition keys and their corresponding values using URL encoding, where `%3D` replaces `=`. For example, the bucket path for data in `env` and `service` partitions is `my-bucket/env%3D${env}/service%3D${service}/`.

Basic Tokens

Basic tokens' syntax follows that of JS template literals: `${token_name}` – where `token_name` is the field (name) of interest.

For example, if the path was set to `/var/log/${hostname}/${dataSource}/`, you could use a filter such as `hostname=='myHost' && dataSource=='mydataSource'` to specify data only from the `/var/log/myHost/mydataSource/` subdirectory.

Time-Based Tokens

Paths with time notation can be referenced with tokens, having a direct effect on the earliest and latest boundaries. The supported time fields are:

- `_time` is the raw event's timestamp.
- `__earliest` is the search start time.
- `__latest` is the search end time.

Time-based tokens are processed as follows:

- For each path, times must be notated in descending order. So Year/Month/Day order is supported, but Day/Month/Year is not.
- Paths may contain more than one time component. For example, `/my/path/2020-04/20/`.
- In a given path, each time component can be used only once.
  So `/my/path/${_time:%Y}/${_time:%m}/${_time:%d}/...` is a valid expression format, but `/my/path/${_time:%Y}/${_time:%m}/${host}/${_time:%Y}/...` (with a repeated `Y`) is not supported.
- For each path, all extracted dates/times are considered in UTC.



Cribl recommends that your path always include and tokenize the largest available time fields, proceeding down to the smallest desired time fields. Otherwise, your searches might yield unexpected results, because Cribl Search will default the omitted fields to their earliest allowed value.

The following `strptime` format components are allowed:

- `Y`, `y` for years
- `m`, `B`, `b`, `e` for months
- `d`, `j` for days
- `H`, `I` for hours
- `M` for minutes

- `S` for seconds
- `s` for Unix-style Epoch times (seconds since 1/1/1970)

Time-based token syntax follows that of a slightly modified JS template literal: `${_time: some_strptime_format_component}`. Examples:

Path	Matches
`/path/${_time:%Y}/${_time:%m}/${_time:%d}/...`	`/path/2020/04/20/...`
`/path/${_time:year=%Y}/${_time:month=%m}/${_time:day=%d}/...`	`/path/year=2020/month=`
`/path/${_time:%Y-%m-%d}/...`	`/path/2020-05-20/...`

Path Filter

This is a JavaScript filter expression that Cribl Search evaluates against the corresponding **Bucket path**. The **Path filter** field's value defaults to `true`, which matches all data. However, you can customize this value almost arbitrarily.

For example, with this **Path filter**:
`source.endsWith('.log') || source.endsWith('.txt')`
...this configuration will search only files/objects with `.log` or `.txt` extensions.

At the **Path filter** ffield's right edge are a Copy button and an Expand button that opens a validation modal.

Search Data Lake Amazon S3

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Data Transfer Charges

Data transfer to Cribl Search is free if your S3 bucket is in the same Region as your Cribl.Cloud Workspace.

Data transfer to Cribl Search is subject to AWS data transfer costs if your S3 bucket is not in the Regions supported by Cribl.Cloud. See the **Data transfer** section in Amazon S3 Pricing.

Cribl Search will try to minimize the amount of data transferred, to the extend possible, by using path filters, pruning, and other methods.

Other Charges

Other charges may apply. For example, **LIST** and **GET** requests to your S3 buckets will be charged per AWS S3. See the **Requests & data retrievals** section in Amazon S3 Pricing.

2.5. Connect Cribl Search To Azure

Link Cribl Search to your Microsoft Azure data.

Azure Blob Storage

Configure Cribl Search to query your Azure Blob Storage data. See: Azure Blob Storage.

Azure Data Explorer

Configure Cribl Search to query Azure Data Explorer. See: Azure Data Explorer.

Azure API

Configure Cribl Search to query an Azure API endpoint. See: Azure API.

2.5.1. Connect Cribl Search To Azure Blob Storage

Configure Cribl Search to query your Azure Blob Storage data.

Azure Blob Storage enables users to store large amounts of unstructured data on Microsoft's data storage platform. Blobs are binary large objects, including images and multimedia.

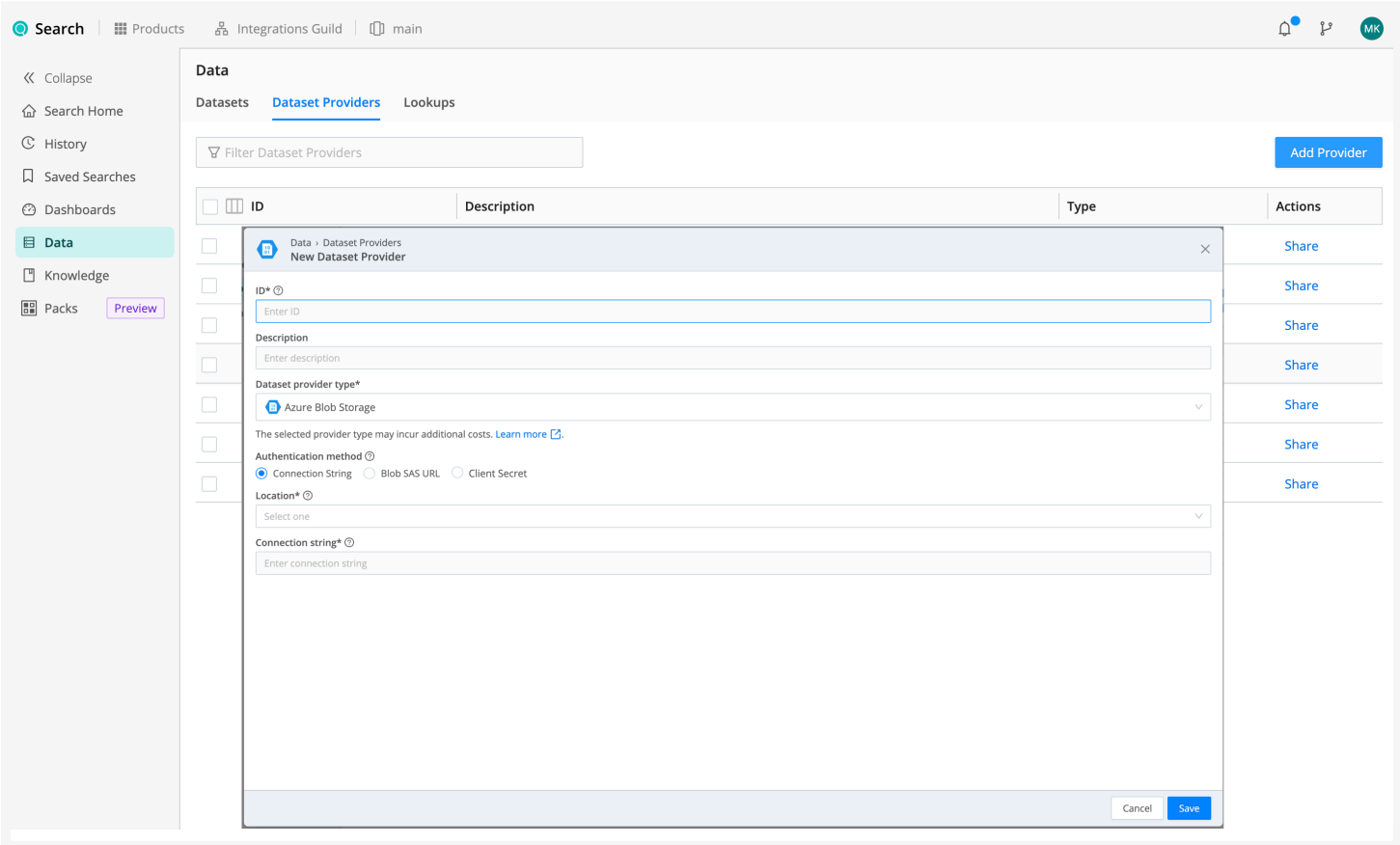
In this guide, you'll set up a Dataset Provider and a Dataset to search objects in Azure Blob Storage.

Data transfer costs and other charges might apply. For details, see Data Transfer Charges below.

Add an Azure Blob Storage Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an Azure Blob Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.



Azure Blob Storage Dataset Provider configuration

Configure the **New Dataset Provider** modal as follows:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional. Here, you can enter a summary that will clarify this Dataset Provider's purpose to other users.
3. Set the **Dataset Provider Type** to `Azure Blob`.
4. **Authentication method** provides three options, **Connection String**, **Blob SAS URL**, and **Client Secret**. For details on each option, see the next section.
5. Select **Save** when finished.

Azure Blob Storage Authentication

Your Azure Blob Storage access policy should include the **Read** and **List** permissions.

In the Cribl Search **New Dataset Provider** modal, use the **Authentication method** buttons to select one of these options:

- Connection String
- Blob SAS URL
- Client Secret

Connection String

The **Connection String** gives admin-level access to Search. This includes the authorization information required to access data in your Azure Storage account at runtime using Shared Key authorization. See the Azure Blob Storage documentation for more details. To view and copy your storage account connection string from the Azure portal:

1. In the Azure portal, go to your storage account.
2. Under **Security + networking**, select **Access keys**. Your account access keys appear, as well as the complete connection string for each key.
3. Select **Show keys** to show your access keys and connection strings and to enable buttons to copy the values.
4. Copy the entire **Connection string** of the key you want to use.

In the Cribl Search **New Dataset Provider** modal, enter the following information under **Authentication method**.

1. **Location** is your Azure account region, for example, `East US`. This can be found on the Azure portal **Overview** page. See the Azure Blob Storage documentation for more details.
2. **Connection String** is where you paste the **Connection string** of the key you want to use.

3. Select **Save** when finished.

Blob SAS URL

Blob SAS URL provides a more-restrictive access token called a "Shared Access Signature", which allows the user to specify which resources to grant access to. For details see, Create SAS tokens for your storage containers. To create SAS tokens for your storage containers:

1. In the Azure portal, navigate to your container.
2. Right-click the container or file and select **Generate SAS** from the drop-down menu.
3. Specify the permission levels, allowed IP addresses, and allowed protocols.
4. Review and then select **Generate SAS token and URL**.
5. The **Blob SAS token** query string and **Blob SAS URL** are displayed in the lower area of the window.
6. Copy and paste the **Blob SAS token** and **URL** values in a secure location. They'll only be displayed once and cannot be retrieved once the window is closed.

In the Cribl Search **New Dataset Provider** modal, enter the following information under **Authentication method**.

1. **Location** is your Azure account region, for example, `East US`. This can be found on the Azure portal **Overview** page. See the Azure Blob Storage documentation for more details.
2. Select **Add configurations** to specify the container(s).
    - **Container Name** is the name of the Azure Blob Storage container.
    - **Blob SAS URL** is where you paste the container-specific Blob SAS URL.
3. Select **Save** when finished.

Client Secret

The **Client Secret** is a type of authentication available for a service principal of your Microsoft Entra (formerly Azure Active Directory) application. To use this authentication method, first do the following in your Azure account:

1. Create and register a service principal for your Microsoft Entra application. To find out how, see Register a Microsoft Entra app and create a service principal from Microsoft.
2. Grant the service principal access to your data. To find out how, see Authorize access to blobs using Microsoft Entra ID.

In the Cribl Search **New Dataset Provider** modal, enter the following information under **Authentication method**.

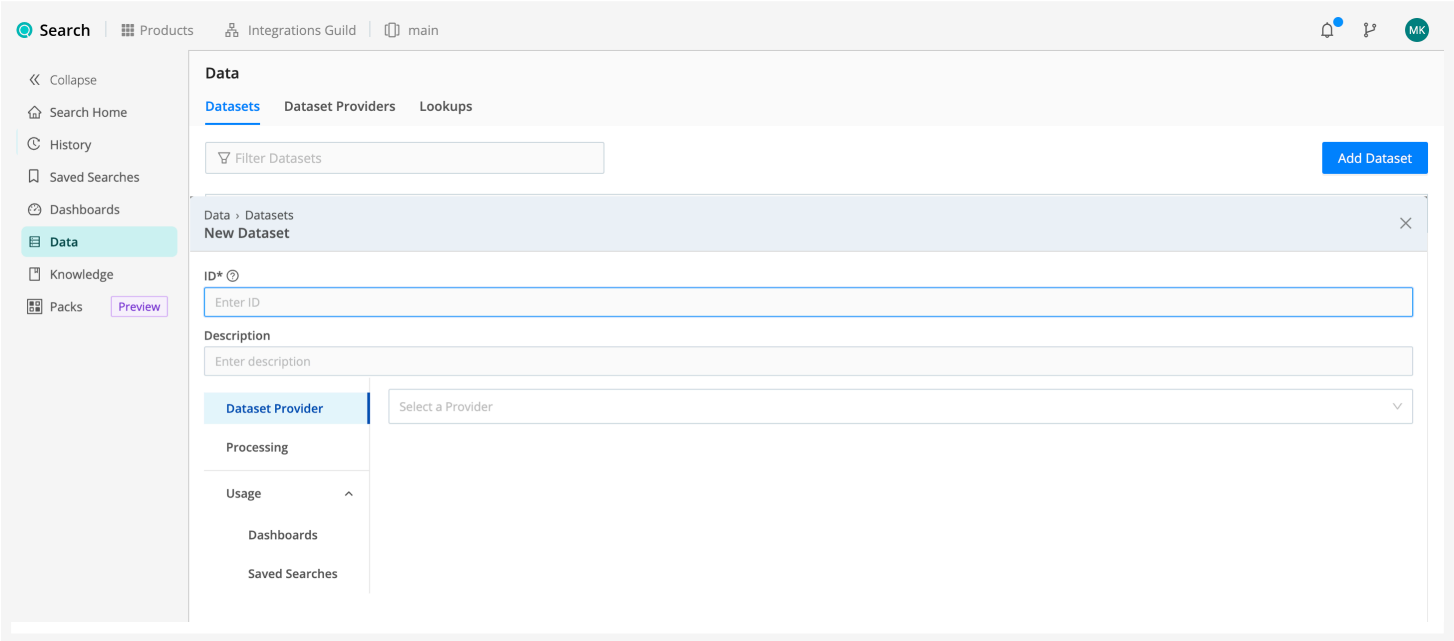
1. **Location** is your Azure account region, for example, `East US`.

2. **Storage account name** is the name of your Azure storage account.
3. **Tenant ID** is your Microsoft Entra tenant ID. To find it, see How to find your Microsoft Entra tenant ID.
4. **Client ID** is the client ID of the Azure service principal. To find it, see Sign in to the application.
5. **Client secret** is the secret to use when connecting to your Microsoft Entra application. To get it, see Create a new client secret.

Add an Azure Blob Storage Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.



Datasets selection

Configure the **New Dataset** modal as follows.

Identify the Dataset

Use the first few fields to uniquely identify this Dataset and specify its type:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`). You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset (for example, `dataset=my_dataset_1`).
2. **Description** is optional. Here, you can enter a summary that will clarify this Dataset's purpose to other users.
3. Set **Dataset Provider** to the **ID** of an Azure Blob Dataset Provider.

Set Up Paths, Regions, Partitioning

Define at least one path, region, and partitioning configuration:

1. **Skip event time filter** is optional. Toggle this on only if you're sure you want to set the time range of your searches by partition time-boundaries rather than events' timestamps
2. **Container path** is the Azure Blob Storage container with your data. See the Azure Blob Storage documentation for details.
3. **Path filter** is a JavaScript filter expression that is evaluated against the container. Defaults to `true`, which matches all data, but you can customize this value.
4. The **Partitioning scheme** defaults to `Defined in Path`. For Splunk-specific alternatives, see Azure Blob Storage Partitioning Scheme.

Multiple Path Configs

Select **Add container/path** to define as many other path configurations as you want for this Dataset. Each path can have its own filter and partitioning scheme.

As with parallel Datasets, each partitioning scheme applies per path. When searching a Dataset, Cribl Search will search all its configured paths. However: **a failed search on any of a Dataset's paths will fail the whole search.**

Set Up Datatypes and Storage Classes

On the **Processing** left tab, you specify **Datatypes** to break data down into discrete events and define fields so they're ready to search.

Use the drop-down to set the first rule to `AWS Datatypes`. Optionally, click **Add Datatype Ruleset** to select more rulesets – each of which contains rules applied to the data searched in your Dataset. For details, see Datatypes.

On the **Storage classes** left tab, select the access tiers that you want this Dataset to search against. You can minimize retrieval costs by selecting only warmer tiers. You must select at least one tier. For details, see Storage Classes.

Save the Configuration

Bypass the **Usage** left tabs for now, and select **Save** when your configuration is complete.

Path/Partitioning Considerations

This section offers details and guidance on configuring search paths, filters, and partitioning schemes. For best practices covering path design across all object stores, see Optimize Paths.

Container Paths

Each **Container path** specifies a separate blob of data within the Dataset. It defines the scope of this data, to narrow down what data is in the Dataset. This field supports tokens and key-value pairs. For example:

- `mycontainer/data` – where `data` becomes a field for all events of that Dataset.
- `mycontainer/${data}/${*}` – equivalent, the wildcarded path is skipped.
- `mycontainer/<key=value>/<someVarOfInterest>` – see Hive-Style Paths just below.

Hive-Style Paths

To search paths of the form `my-bucket/<key=value>/<someVarOfInterest>`, your expression should place the wildcard `{*}` in the ordinal position, to ignore the hive (K-V) segment:

```
my-bucket/${*}/${someVarOfInterest}
```

This will allow Cribl Search to find the wildcarded subdirectory automatically.

Basic Tokens

Basic tokens' syntax follows that of JS template literals: `${token_name}` – where `token_name` is the field (name) of interest.

For example, if the path was set to `/var/log/${hostname}/${dataSource}/`, you could use a filter such as `hostname=='myHost' && dataSource=='mydataSource'` to specify data only from the `/var/log/myHost/mydataSource/` subdirectory.

Time-Based Tokens

Paths with time notation can be referenced with tokens, having a direct effect on the earliest and latest boundaries. The supported time fields are:

- `_time` is the raw event's timestamp.
- `__earliest` is the search start time.
- `__latest` is the search end time.

Time-based tokens are processed as follows:

- For each path, times must be notated in descending order. So Year/Month/Day order is supported, but Day/Month/Year is not.
- Paths may contain more than one time component. For example, `/my/path/2020-04/20/`.
- In a given path, each time component can be used only once.
  So `/my/path/${_time:%Y}/${_time:%m}/${_time:%d}/...` is a valid expression format, but

`/my/path/${_time:%Y}/${_time:%m}/${host}/${_time:%Y}/...` (with a repeated `Y`) is not supported.

- For each path, all extracted dates/times are considered in UTC.



Cribl recommends that your path always include and tokenize the largest available time fields, proceeding down to the smallest desired time fields. Otherwise, your searches might yield unexpected results, because Cribl Search will default the omitted fields to their earliest allowed value.

The following `strptime` format components are allowed:

- `Y`, `y` for years
- `m`, `B`, `b`, `e` for months
- `d`, `j` for days
- `H`, `I` for hours
- `M` for minutes
- `S` for seconds
- `s` for Unix-style Epoch times (seconds since 1/1/1970)

Time-based token syntax follows that of a slightly modified JS template literal: `${_time: some_strptime_format_component}`. Examples:

Path	Matches
`/path/${_time:%Y}/${_time:%m}/${_time:%d}/...`	`/path/2020/04/20/...`
`/path/${_time:year=%Y}/${_time:month=%m}/${_time:day=%d}/...`	`/path/year=2020/month=`
`/path/${_time:%Y-%m-%d}/...`	`/path/2020-05-20/...`

Set Query Time Range by Partition

If your data is partitioned by time (so your paths probably use time-based tokens), you can tell Cribl Search to ignore events' timestamps, and instead set the time range using the partition time-boundaries.

For example, an event with the timestamp of `2025-07-01T23:59:00.000` might be stored in the July 2 partition. Queries for July 1 or July 2 might not find it: the first due to wrong partition, the second due to wrong `_time`. Using partition time for the query range avoids this issue.

To search by partition time range, toggle on **Skip event time filter** when setting up your paths. Then, any time range you set (in the Cribl Search UI or with `earliest/latest` in the query text) will use partition time instead of event time.

If you toggle on the **Skip event time filter** option, Cribl Search will ignore the `_time` field in your events. Do **not** enable this option if your paths don't use time-based tokens.

Azure Blob Storage Path Filter

Each **Path filter** is a JavaScript expression that Cribl Search evaluates against the corresponding **Container path**. The **Path filter** field's value defaults to `true`, which matches all data. However, you can customize this value almost arbitrarily.

For example, if a Dataset has this **Path filter**:
`source.endsWith('.log') || source.endsWith('.txt')`
...this configuration will search only files/objects with `.log` or `.txt` extensions.

At the **Path filter** field's right edge are a Copy button and an Expand button that opens a validation modal.

With DDSS or SmartStore partitioning, you can provide the index name as **Path filter**, for example: `index=="index_y"`.

Azure Blob Storage Partitioning Scheme

Each **Partitioning scheme** drop-down offers alternative options for partitioning inbound data from Splunk. You can choose between `Splunk DDSS` and `Splunk SmartStore`.

With either option, in the corresponding **Container path** field, provide the path where indexes are stored (that is, your index's parent folder).

For Splunk DDSS, the full path takes the form:
`<parent_folder>/<indexName>/db/db_<latestTime>_<earliestTime>_<bucketId>/rawdata/journal`
...or:
`<parent_folder>/<indexName>/db_<latestTime>_<earliestTime>_<bucketId>/rawdata/journal.gz`

For SmartStore, the full path takes the form:
`<parent_folder>/<indexName>/db/<2-letter-hash>/<2-letter-hash>/<bucket_id_number-origin_guid>/<"guidSplunk"-uploader_guid>/`

With either option, if you organize your container using this default file path, Cribl Search will automatically discover its content.

Search Azure Blob Storage

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

It can take a few minutes for a search to start returning results.

Data Transfer Charges

Cribl Search uses both Read and Iterative Read Operations in Azure Blob Storage. To understand the fees associated with these operations, refer to Azure Blob Storage's Operation and data transfer.

Additionally, retrieval fees may apply depending on the storage tier used. For details, see Microsoft's Retrieval Fees topic.

Using Cribl Search to read data from the Azure data centers might also result in Data transfer charges for egress. For detailed information, refer to Microsoft's Data Transfer pricing.

2.5.2. Connect Cribl Search To Azure Data Explorer

Configure Cribl Search to query Azure Data Explorer.

Cribl Search allows you to search database tables from Azure Data Explorer (ADX). ADX stores log data in databases, which in turn contain one or more of the tables defined in the Azure namespace.



Azure Data Explorer has a 500,000 result limit per search.

In this guide, you'll set up a Dataset Provider and a Dataset to search database tables in Azure Data Explorer.

Add an Azure Data Explorer Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an Azure Data Explorer Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Azure Data Explorer**.
4. The following inputs are retrieved from Azure Data Explorer after creating and registering a Microsoft Entra service principal and then authorizing it to access an Azure Data Explorer database. See Microsoft's guide - Create a Microsoft Entra application registration in Azure Data Explorer for details.
    - **Tenant ID** is the ID of the Directory (tenant) in Azure Data Explorer to retrieve information from.
    - **Client ID** is the ID of the Application (client) in Azure Data Explorer.
    - **Client Secret** is the client secret that will be used as the secret in the connection to the application.
5. Select **Save** when finished.

Add an Azure Data Explorer Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of an Azure Data Explorer provider.
    - Enter the **Cluster** name and select its region from the **Location** drop-down.
    - **Database** is the database name.
    - **Table or Query** supports the Kusto Query Language. You can specify the name of a table to query, like `logs`, or enter a Kusto query, such as `StormEvents | summarize PropertyDamage = sum(DamageProperty) by State | join kind=innerunique PopulationData on State`.
    - **Timestamp field** is the event field with a timestamp. You need to specify a timestamp field to query by date and time in Cribl Search.
    - **Timestamp field contents** is the data type of the timestamp field. Defaults to Kusto's datetime data type.
4. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
5. Select **Save** when finished.

Search Azure Data Explorer

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

It can take a few minutes for a search to start returning results. Here's an example of an initial exploratory search:

```
dataset="adx" | limit 1000
```

Copy

2.5.3. Connect Cribl Search To Azure API

Configure Cribl Search to query an Azure API endpoint.

Microsoft Azure is a public cloud computing platform that offers a range of services that include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

In this guide, you'll set up a Dataset Provider and a Dataset to search the disks, networkSecurityGroups, virtualMachines, and webapps endpoints in the Azure API.

Azure API Authorization

Set up an Azure service account with a client secret credential for Search. For details, see Create a Microsoft Entra application and service principal that can access resources. You will need the account credentials to Create a Dataset Provider.

Also, you can assign the built-in role of Reader to the application so it has read access to all endpoints. To limit access to the current Cribl Search endpoints (listed below), create a custom role:

- Microsoft.Compute/disks
- Microsoft.Network/networkSecurityGroups
- Microsoft.Compute/virtualMachines
- Microsoft.Web/sites

You can modify permissions as the application adds more endpoints. For details, see Create an Azure custom role.

Add an Azure API Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an Azure API Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Azure API**.

4. Select **Add Configuration** to specify your Azure account(s).
    - **Account Name** is the Azure account name.
    - **Tenant ID** is the ID of the Microsoft Entra ID to retrieve information from.
    - **Client ID** is the ID of the application that will connect to Microsoft Entra ID Active Directory. For details, see Register an application.
    - **Client Secret** is the key that will be used as the secret in the connection to Microsoft Entra ID. For details, see Add a client secret.
5. Select **Save** when finished.

Add an Azure API Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of an Azure API Dataset Provider.
4. Select **Add endpoint** to select the endpoints for your Dataset.
5. **Enabled endpoints**: Select an endpoint from the drop-down menu. Your options are:
    - virtualMachines
    - disks
    - networkSecurityGroups
    - webapps
6. **Subscription IDs** is a list of the Subscription IDs within the tenant to query with this Dataset.
7. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
8. In **Snapshots**, you can set up API Snapshots.
9. Select **Save** when finished.

Search Azure API

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

2.6. Connect Cribl Search To Google

Link Cribl Search to your Google services.

Google Cloud Storage

Configure Cribl Search to query your Google Cloud Storage data. See: Google Cloud Storage.

Google Cloud Platform API

Configure Cribl Search to query a Google Cloud Platform API endpoint. See: Google Cloud Platform API.

Google Workspace API

Configure Cribl Search to query a Google Workspace API endpoint. See: Google Workspace API.

2.6.1. Connect Cribl Search To Google Cloud Storage

Configure Cribl Search to query your Google Cloud Storage data.

Google Cloud Storage is a managed service for storing unstructured data.

In this guide, you'll set up a Dataset Provider and a Dataset to search your Google Cloud Storage account.

Data transfer costs and other charges may apply. For details, see the Data Transfer Charges section below.

Add a Google Cloud Storage Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add a Google Cloud Storage Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Configure the **New Dataset Provider** modal as follows:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Google Cloud Storage**.
4. The **Service Account Credentials** are the contents of your Google Cloud service account (JSON keys). You can also drag and drop a file into the field, or upload a file by clicking the upload button.
   For more information on the permissions needed, see the Google Cloud Permissions section.
   For more information on obtaining your Google Cloud Storage credentials, see Google's Authenticate to Cloud Storage.
5. Specify an **Endpoint**. If empty, the endpoint will default to `https://storage.googleapis.com`. For a list of the endpoints, see the Google Cloud Storage API Reference doc.
6. Select **Save** when finished.

Google Cloud Permissions

To let Cribl Search access your Google Cloud data, your Google Cloud service account needs at the minimum all of the following roles:

- `Storage Legacy Bucket Reader`
- `Storage Legacy Object Reader`

Alternatively, you can define a custom role that has all of the following permissions:

- `storage.buckets.get`
- `storage.objects.list`

For more information, see Google's IAM roles for Cloud Storage.

Add a Google Cloud Storage Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Configure the **New Dataset** modal as follows.

Identify the Dataset

Use the first few fields to uniquely identify this Dataset and specify its type:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`). You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset (for example, `dataset=my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of a Google Cloud Storage Dataset Provider.

Set Up Paths and Partitioning

Define at least one path, filter, and partitioning configuration:

1. **Skip event time filter** is optional. Toggle this on only if you're sure you want to set the time range of your searches by partition time-boundaries rather than events' timestamps
2. **Bucket Path** is the path to the Google Cloud Storage objects you'd like to search. Start the path with the bucket name, for example: `my-bucket/data/*`. Tokens and key-value pairs are supported. For details, see bucket path.
3. **Path Filter** is a JavaScript filter expression that is evaluated against the S3 bucket path. Defaults to `true`, which matches all data, but you can customize this value.
4. The **Partitioning scheme** defaults to `Defined in Path`. For Splunk-specific alternatives, see Partitioning Scheme.

Multiple Path Configs

Select **Add bucket/path** to define as many other path configurations as you want for this Dataset. Each path can have its own filter and partitioning scheme.

As with parallel Datasets, each partitioning scheme applies per path. When searching a Dataset, Cribl Search will search all its configured paths. However: **a failed search on any of a Dataset's paths will fail the whole search.**

Set Up Datatypes and Storage Classes

On the **Processing** left tab, you specify **Datatypes** to break data down into discrete events and define fields so they're ready to search.

Use the drop-down to set the first rule to `AWS Datatypes`. Optionally, click **Add Datatype Ruleset** to select more rulesets – each of which contains rules applied to the data searched in your Dataset. For details, see Datatypes.

On the **Storage classes** left tab, select the storage classes that you want this Dataset to search against. You can minimize retrieval costs by selecting only warmer classes. You must select at least one class. For details, see Storage Classes.

Save the Configuration

Bypass the **Usage** left tabs for now, and select **Save** when your configuration is complete.

Path/Partitioning Considerations

This section offers details and guidance on configuring search paths, filters, and partitioning schemes. For best practices covering path design across all object stores, see Optimize Paths.

Bucket Paths

The **Bucket Path** specifies the data that the Dataset consists of. It defines the scope of data, to narrow down what data is in the Dataset. This is a JavaScript expression that supports tokens and key-value pairs. For example,

- `my-bucket/${data}/` – where `data` becomes a field for all events of that Dataset.
- `my-bucket/${data}/${*}` – equivalent, the wildcarded path is skipped.
- `my-bucket/<key=value>/<someVarOfInterest>` – see Hive-Style Paths just below.

Hive-Style Paths

To search paths of the form `my-bucket/<key=value>/<someVarOfInterest>`, your expression should place the wildcard `{*}` in the ordinal position, to ignore the hive (K-V) segment:

```
my-bucket/${*}/${someVarOfInterest}
```

This will allow Cribl Search to find the wildcarded subdirectory automatically.

Basic Tokens

Basic tokens' syntax follows that of JS template literals: `${token_name}` – where `token_name` is the field (name) of interest.

For example, if the path was set to `/var/log/${hostname}/${dataSource}/`, you could use a filter such as `hostname=='myHost' && dataSource=='mydataSource'` to specify data only from the `/var/log/myHost/mydataSource/` subdirectory.

Time-Based Tokens

Paths with time notation can be referenced with tokens, having a direct effect on the earliest and latest boundaries. The supported time fields are:

- `_time` is the raw event's timestamp.
- `__earliest` is the search start time.
- `__latest` is the search end time.

Time-based tokens are processed as follows:

- For each path, times must be notated in descending order. So Year/Month/Day order is supported, but Day/Month/Year is not.
- Paths may contain more than one time component. For example, `/my/path/2020-04/20/`.
- In a given path, each time component can be used only once.
  So `/my/path/${_time:%Y}/${_time:%m}/${_time:%d}/...` is a valid expression format, but `/my/path/${_time:%Y}/${_time:%m}/${host}/${_time:%Y}/...` (with a repeated `Y`) is not supported.
- For each path, all extracted dates/times are considered in UTC.



Cribl recommends that your path always include and tokenize the largest available time fields, proceeding down to the smallest desired time fields. Otherwise, your searches might yield unexpected results, because Cribl Search will default the omitted fields to their earliest allowed value.

The following `strptime` format components are allowed:

- `Y`, `y` for years
- `m`, `B`, `b`, `e` for months

- `d`, `j` for days
- `H`, `I` for hours
- `M` for minutes
- `S` for seconds
- `s` for Unix-style Epoch times (seconds since 1/1/1970)

Time-based token syntax follows that of a slightly modified JS template literal: `${_time: some_strptime_format_component}`. Examples:

Path	Matches
`/path/${_time:%Y}/${_time:%m}/${_time:%d}/...`	`/path/2020/04/20/...`
`/path/${_time:year=%Y}/${_time:month=%m}/${_time:day=%d}/...`	`/path/year=2020/month=`
`/path/${_time:%Y-%m-%d}/...`	`/path/2020-05-20/...`

Set Query Time Range by Partition

If your data is partitioned by time (so your paths probably use time-based tokens), you can tell Cribl Search to ignore events' timestamps, and instead set the time range using the partition time-boundaries.

For example, an event with the timestamp of `2025-07-01T23:59:00.000` might be stored in the July 2 partition. Queries for July 1 or July 2 might not find it: the first due to wrong partition, the second due to wrong `_time`. Using partition time for the query range avoids this issue.

To search by partition time range, toggle on **Skip event time filter** when setting up your paths. Then, any time range you set (in the Cribl Search UI or with `earliest/latest` in the query text) will use partition time instead of event time.



If you toggle on the **Skip event time filter** option, Cribl Search will ignore the `_time` field in your events. Do **not** enable this option if your paths don't use time-based tokens.

Path Filter

Each **Path filter** is a JavaScript expression that Cribl Search evaluates against the corresponding **Bucket path**. The **Path filter** field's value defaults to `true`, which matches all data. However, you can customize this value almost arbitrarily.

For example, with this **Path filter**:
`source.endsWith('.log') || source.endsWith('.txt')`
...this configuration will search only files/objects with `.log` or `.txt` extensions.

At each **Path filter** field's right edge are a Copy button and an Expand button that opens a validation modal.

With DDSS or SmartStore partitioning, you can provide the index name as the **Path filter**, for example: `index=="index_y"`

Partitioning Scheme

Each **Partitioning scheme** drop-down offers alternative options for partitioning inbound data from Splunk. You can choose between `Splunk DDSS` and `Splunk SmartStore`.

With either option, in the corresponding **Bucket path** field, provide the path where indexes are stored (that is, your index's parent folder).

For Splunk DDSS, the full path takes the form:
`<parent_folder>/<indexName>/db/db_<latestTime>_<earliestTime>_<bucketId>/rawdata/journal`
...or:
`<parent_folder>/<indexName>/db_<latestTime>_<earliestTime>_<bucketId>/rawdata/journal.gz`

For SmartStore, the full path takes the form:
`<parent_folder>/<indexName>/db/<2-letter-hash>/<2-letter-hash>/<bucket_id_number-origin_guid>/<"guidSplunk"-uploader_guid>/`

With either option, if you organize your bucket using this default file path, Cribl Search will automatically discover its content.

Search Google Cloud Storage

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

Data Transfer Charges

Cribl Search uses both Class A and Class B operations in Google Cloud Storage. To understand the fees associated with these operations, refer to the Google Cloud Storage's Operation charges.

Additionally, retrieval fees may apply depending on the storage class of the bucket. See the Retrieval Fees section for details.

Using Cribl Search to read data from US regions might also result in General network usage egress charges. These charges fall under the "Egress Worldwide Destinations (excluding Asia & Australia)" category. For detailed information, refer to the General network usage.

2.6.2. Connect Cribl Search To Google Cloud Platform API

Configure Cribl Search to query a Google Cloud Platform API endpoint.

Google Cloud Platform provides a suite of computing services including data management, web and video delivery, AI and machine learning tools.

In this guide, you'll set up a Dataset Provider and a Dataset to search the instances, disks, security policies, and locations endpoints in your Google Cloud Platform account.

Add a Google Cloud Platform API Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add a Google Cloud Platform API Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Google Cloud Platform API**.
4. Select **Add Configuration** to specify your Google Cloud Platform account. In the **Account Configurations** table, enter:
    - **Name**: Google Cloud Platform account name.
    - **Service Account Credentials**: The contents of your Google Cloud service account (JSON keys) file. You can also drag and drop a file into the field, or upload a file by clicking the upload button.
5. Select **Save** when finished.

Add a Google Cloud Platform API Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of a Google Cloud Platform Dataset Provider.
4. Select **Add endpoint** to select the endpoints for your Dataset.
5. **Enabled endpoints**: Select an endpoint from the drop-down menu. For details on the endpoints, see the Compute Engine API reference docs. Your options are:
    - `v1.instances`
    - `v1.disks`
    - `v1.securityPolicies`
    - `v1.projects.locations.functions`
    - `v2.projects.locations.functions`

    1. **Region**: The region of your Google Cloud Platform account, provided for certain endpoints, indicates its location.
6. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
7. In **Snapshots**, you can set up API Snapshots.
8. Select **Save** when finished.

Search Google Cloud Platform

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

2.6.3. Connect Cribl Search To Google Workspace API

Configure Cribl Search to query a Google Workspace API endpoint.

A Google Workspace plan offers business-specific email as well as collaboration tools such as Gmail, Calendar, Meet, Chat, Drive, Docs, Sheets, Slides, Forms, and Sites.

In this guide, you'll set up a Dataset Provider and a Dataset to search the activities of the admin, devices, drives, logins, and tokens endpoints on your Google Workspace account.

Google Workspace API Authorization

To allow Cribl Search to read your data, do the following on the Google Workspace side:

1. Create or use an existing Google Workspace service account.
2. Configure the service account with domain-wide delegation for the following scope:

   ```
   https://www.googleapis.com/auth/admin.reports.audit.readonly
   ```

   Copy

   Read more on domain-wide delegation.
3. Obtain the service account credentials. You'll need the API keys in JSON format when creating the Google Workspace API Dataset Provider.
4. Create or use an existing Google Workspace user account. This can be a real user or just a service placeholder. You'll need the user account's email address when creating the Google Workspace API Dataset Provider.
5. Authorize the user account with the Super Admin role. Due to how the Google Workspace APIs work, the user account must have the full Super Admin permissions for the resources that you want to search.

Add a Google Workspace API Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add a Google Workspace API Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Google Workspace API**.
4. Select **Add Configuration** to specify your Google Workspace account(s).
    - **Account Name** is the name of your Google Workspace account.
    - **Impersonated Account's Email Address** is the email address of the **user account** that's used for searching the APIs. For details, see Authorization.
    - **Service Account Credentials** is the JSON keys of your Google Workspace **service account**. For details, see Authorization.
5. Select **Save** when finished.

Add a Google Workspace API Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of a Google Workspace Dataset Provider.
4. Select **Add endpoint** to select the endpoints for your Dataset.
5. **Enabled endpoints**: Select an endpoint from the drop-down menu. For details on the endpoints, see the Reports API Overview reference docs. Your options are:
    - `admin`
    - `mobile`
    - `devices`
    - `drive`
    - `login`
    - `token`
6. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.

7. In **Snapshots**, you can set up API Snapshots.
8. Select **Save** when finished.

Search Google Workspace API

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

2.7. Connect Cribl Search To ClickHouse

Configure Cribl Search to query ClickHouse Cloud or a self-managed ClickHouse server.

ClickHouse is a column-oriented SQL database management system for online analytical processing, available both as open-source software and a cloud offering.

To start searching your ClickHouse databases with Cribl Search, you need:

- A ClickHouse Dataset Provider, which contains access credentials, and specifies what ClickHouse server or Cloud service to query.
- A ClickHouse Dataset, which points Cribl Search at a specific table, view, or query in a ClickHouse database.

Each Dataset Provider can have multiple Datasets assigned to it.

Add a ClickHouse Dataset Provider

Before adding a ClickHouse Dataset Provider, do the following:

1. Make sure your ClickHouse server or Cloud service has the HTTP interface enabled. Ideally, use HTTPS over the default port 8443.
2. Check your ClickHouse URL, username, and password.

To add a ClickHouse Dataset Provider in Cribl Search:

1. Log in to Cribl Search as an Admin or Editor Search Member.
2. Select **Data**, then **Dataset Providers**, then **Add Provider**.
3. In **ID**, enter a unique identifier. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_clickhouse_database_1`).
   This is how you or other Search Members reference your Dataset Provider when adding Datasets to it.
4. In **Description**, you can briefly describe your Dataset Provider, so that other Search Members know its purpose.
5. Set **Dataset Provider Type** to **ClickHouse**.
6. Enter the following:
   - **Username**: Your ClickHouse username, for authentication.
   - **Password**: Your ClickHouse password, for authentication.
   - **Endpoint**: Your ClickHouse server or Cloud service URL.

7. Select **Save**.

Now, you're ready to add a ClickHouse Dataset.

Add a ClickHouse Dataset

With a ClickHouse Dataset Provider ready, you can start adding ClickHouse Datasets:

1. Log in to Cribl Search as an Admin or Editor Search Member.
2. Select **Data**, then **Datasets**, then **Add Dataset**.
3. In **ID**, enter a unique identifier. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `clickhouse_dataset_1`).
   You and other Search Members will reference this ID when searching your Dataset (`dataset="clickhouse_dataset_1"`).
4. In **Description**, you can briefly describe your Dataset, so that other Search Members know its purpose.
5. In **Select a Provider**, pick the ID of a ClickHouse Dataset Provider.
6. Enter the following:
   - **Database name**: Case-sensitive name of the ClickHouse database. If not set, defaults to the ClickHouse `default_database` setting.
   - **Table name, view, or query**: The name of the table or view, or a query. May be a simple name (`logs`) or an SQL query (`select * from logs`). Mind that for queries, and for tables without sorting keys, results will be limited to 100,000.
   - **Timestamp field**: Optional, case-sensitive name of the column that holds the timestamp of the event to query.
7. Select **Save**.

List the Contents of a ClickHouse Dataset

You can quickly list all tables and views accessible through a ClickHouse Dataset, using the `.show objects` command. For example:

```
.show objects(clickhouse_dataset_ID)
```

Copy

Search a ClickHouse Dataset

After adding a ClickHouse Dataset Provider and Dataset, you're ready to start searching.

Cribl Search submits the query to ClickHouse. Results will start showing up after the ClickHouse SQL query completes. How long it takes depends on how much data there is in your ClickHouse database. Searching columns that contain large strings may result in a slower search.

Supported ClickHouse Data Types

Cribl Search supports the following ClickHouse data types:

- Strings: `String`, `FixedString`
- Integer types: `UInt8`, `UInt16`, `UInt32`, `UInt64`, `UInt128`, `UInt256`, `Int8`, `Int16`, `Int32`, `Int64`, `Int128`, `Int256`
- Floating-point numbers: `Float32`, `Float64`, `Decimal`
- Boolean: `Bool`
- Dates: `Date`, `Date32`, `DateTime`, `DateTime64`
- `UUID`
- Low cardinality types: `Enum`, `LowCardinality`
- `SimpleAggregateFunction`
- `Nullable`
- IP addresses: `IPv4`, `IPv6`

The following ClickHouse data types are **not supported**:

- `JSON`
- `Array`
- `Map`
- `AggregateFunction`
- `Tuple`
- Nested data structures (`Nested`)
- Geo types: `Point`, `Ring`, `Polygon`, `MultiPolygon`
- Special data types (including `Expression`, `Set`, `Nothing` and `Interval`)

Known ClickHouse Limitations

Searching ClickHouse comes with the following limitations:

- **Limited unsorted data:** When your ClickHouse Dataset targets a query, or a table without sorting keys, the maximum number of results is 100,000.

- **ClickHouse Cloud startup lag:** If you're using ClickHouse Cloud, and the service is not awake, the first search might be slow as ClickHouse wakes the service. This can take 20–30 seconds or more.
- **Structured data type parsing:** ClickHouse will error out if a parameter's value for structured data types like Bool, IPv4, IPv6, or UUID cannot be parsed. For example, using a value like `1.0.0` for an IPv4 column will fail with an error of the form: `Code: 675. DB::Exception: Cannot parse IPv4: value 1.0.0 cannot be parsed as IPv4 for query parameter.`
- **Wildcard/regex matching on non-string columns:** The `startswith` operator, and others that use `ilike` in ClickHouse SQL, will fail with an error for non-string columns such as `UUID` or `Int32`. For example, attempting to match `*2345*` on a numeric column will trigger an error of the form: `DB::Exception: Illegal type Int32 of argument of function ilike`. As a workaround, you can force a cast to `toString()` for regex and wildcard matches on non-string columns, to ensure that they work correctly.
- **Operators with incorrect formats:** Operators like `!=`, `==`, `<`, `>`, `<=`, `>=` will fail if the data format is incorrect. For example, comparing `1.2.3` with an IPv4 column will trigger an error if `1.2.3` is not a valid IP. Although casting to `toString()` is a potential workaround, it's better to validate the format first, to benefit from ClickHouse's type advantages.

2.8. Connect Cribl Search To Elasticsearch

Configure Cribl Search to query an Elasticsearch index.

Elasticsearch is a proprietary search engine for data-intensive applications.

In this guide, you'll set up a Dataset Provider and a Dataset to search an Elasticsearch index.

Add an Elasticsearch Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an Elasticsearch Dataset Provider.

To add a new Dataset Provider:

1. From the sidebar, select **Data**, then **Dataset Providers**.
2. Select the **Add Provider** button on the right.

Next, set the following configurations in the **New Dataset Provider** modal:

1. In **ID**, enter a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. Optionally, add **Description** to quickly describe the Dataset Provider.
3. Set **Dataset Provider Type** to **Elasticsearch**.
4. Enter the following:
    - **Username**: Your Elasticsearch account name.
    - **Password**: Your password for Elasticsearch.
    - **Endpoint**: URL for the Elasticsearch endpoint, for example: `https://es.mycompany.com`.
5. Select **Save** when finished.

Add an Elasticsearch Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Next, set the following configuration in the resulting **New Dataset** modal:

1. In **ID**, enter a unique identifier for the Dataset. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. Optionally, add **Description** to quickly describe the Dataset.
3. Set **Dataset Provider** to the **ID** of an Elasticsearch Dataset Provider.
4. In **Index**, point to the identifier of the Elasticsearch index you want to search. If you have rolling indices (indices with dynamic naming), use the name of the index and the wildcard character `*`. For example, if your indices are named `my-index-00001`, `my-index-00002`, and so on, use `my-index*` in this field.
5. Make sure **Timestamp field** contains the name of the field that contains event timestamps. In most cases, this will be `@timestamp` for time series data in Elasticsearch.
6. Select **Save** when finished.

Search Elasticsearch

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

Elasticsearch Results Limit

The number of Elasticsearch results is capped by the `max_result_window` setting of your Elasticsearch index. By default, it's 10,000.

If you need to raise this limit, make sure you research and test for potential side effects.

2.9. Connect Cribl Search To A Generic HTTP API

Configure Cribl Search to query any HTTP API.

This guide shows you how to set up a Dataset Provider and Datasets to query static or dynamic HTTP API endpoints.



You can also quickly fetch data from an HTTP(S) URL by using the `externaldata` operator.

Add a Generic HTTP API Dataset Provider

A Dataset Provider tells Cribl Search where to query, and contains access credentials.

To add a new HTTP API Dataset Provider:

1. In Cribl Search, select **Data**, then **Dataset Providers**, then **Add Provider**.
2. In **ID**, enter the ID for the new Dataset Provider. This is how you'll reference the provider when assigning your HTTP API Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
3. In **Description**, you can add a few words to make your new Dataset Provider easy to recognize.
4. Set **Dataset Provider Type** to **Generic HTTP API**. You'll see two new sections: **Endpoints** and **Authentication**.
5. In **Endpoints**, specify the endpoints that your Datasets will be able to reference. For each endpoint, set the following:
    - **Name**: This is how you'll recognize the endpoint when setting up your HTTP API Datasets.
    - **Data field**: Enter the name of the field in the response JSON to pull data from. Leave blank if the result is not JSON or is an array of values. You can define parameters, in the form of `${parameterName}`.
    - **URL**: The URL for the endpoint. This is where Cribl Search will query for data. You can define parameters, in the form of `${parameterName}`.
    - **Method**: The HTTP method to use when making API requests.
    - **Headers**: Any HTTP headers needed to authenticate or authorize requests.
      For example, you might need to add an `Authorization` header with a bearer token. You can define parameters, in the form of `${parameterName}`.
6. In **Authentication**, you can select and configure an optional authentication method to use when making API requests. The default option is **None**.

7. When done, select **Save**.

Now, you can add a Dataset that tells Cribl Search what data to search from your new Dataset Provider.

Add a Generic HTTP API Dataset

A Dataset tells Cribl Search what data to search from the Dataset Provider.

To add a new HTTP API Dataset:

1. In Cribl Search, select **Data**, then **Datasets**, then **Add Dataset**.
2. In **ID**, enter the ID for the new Dataset. This is how you'll reference the Dataset to search it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
3. In **Description**, you can add a few words to make your new Dataset easy to recognize.
4. Set **Dataset Provider** to the **ID** of an HTTP API Dataset Provider.
5. Select the endpoints that you want to search with this Dataset.
   To configure more endpoints, go to the Dataset Provider.
6. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
7. In **Snapshots**, you can set up API Snapshots.
8. When done, select **Save**.

Authentication Methods for Generic HTTP API

Cribl Search supports the following authentication methods for Generic HTTP API Dataset Providers:

- **None**: No authentication. This is the default option.
- Basic
- OAuth
- Login

Basic Authentication

To configure HTTP Basic authentication for a Generic HTTP API Dataset Provider in Cribl Search:

1. Start adding a Generic HTTP API Dataset Provider, or editing an existing one.
2. In the **Authentication** section, set **Authentication method** to **Basic**.
3. Enter the HTTP Basic **Username** and **Password** to use when making API requests.
4. Select **Save**.

Login Authentication

Login is the best authentication method to use for all services that return a JSON web token (JWT) or other authentication token whose content type is `application/json` or `content/plain`.

To configure Login authentication for a Generic HTTP API Dataset Provider in Cribl Search:

1. Start adding a Generic HTTP API Dataset Provider, or editing an existing one.
2. In the **Authentication** section, set **Authentication method** to **Login**.
3. Configure the following:
    - **Login URL**: URL for the Login API call, which is normally a POST call.
    - **Login username**: Login username.
    - **Login password**: Login password.
    - **POST body**: Template for POST body to send with the login request. The `${username}` and `${password}` variables specify the corresponding credentials' locations in the message.
    - **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes. If the response from the auth request provides the JWT as `plain/text`, leave this field blank.
    - **Authorization header**: Authorization header key to pass to each endpoint's API request. Defaults to the literal name `Authorization`.
    - **Authorize expression**: JavaScript expression used to compute an Authorization header to pass to each endpoint's API request. Uses `${token}` to reference the token obtained from a Login or OAuth authorization request.
    - **Authentication headers**: Optionally, select **Add Header** for each custom auth header you want to define.

      Enter each header's **Name** and **Value**. In **Value**, you can define parameters, in the form of `${parameterName}`.

OAuth Authentication

To configure OAuth 2.0 authentication for a Generic HTTP API Dataset Provider in Cribl Search:

1. Start adding a Generic HTTP API Dataset Provider.
2. When you get to the **Authentication** section, set **Authentication method** to **OAuth**.
3. Configure the following:
    - **Login URL**: Endpoint for the OAuth API call, which is normally a POST call.
    - **Client secret parameter**: The name of the parameter to send with the **Client secret value**.
    - **Client secret value**: The OAuth client secret to authorize for an access token.

- **Extra authentication parameters**: Optionally, select **Add parameter** for each additional OAuth request parameter you want to send in the body of POST requests. Automatically sets the `Content-Type` header to `application/x-www-form-urlencoded`.

  Enter each parameter's **Name** and **Value**. In **Value**, you can define parameters, in the form of `${parameterName}`.

- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes.
- **Authorization header**: Authorization header key to pass to each endpoint's API request. Defaults to the literal name `Authorization`.
- **Authorize expression**: JavaScript expression used to compute an Authorization header to pass to each endpoint's API request. Uses `${token}` to reference the token obtained from a Login or OAuth authorization request.
- **Authentication headers**: Optionally, select **Add Header** for each custom auth header you want to define.

  Enter each header's **Name** and **Value**. In **Value**, you can define parameters, in the form of `${parameterName}`.

Search a Generic HTTP API Dataset

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

Dynamically Query HTTP Endpoints

You can dynamically query different HTTP API endpoints, without having to change the Dataset Provider configuration for each query.

To be able to do that, configure your Generic HTTPI API Dataset Provider with parameters in the form of `${parameterName}`. Then you can provide values for those parameters as key-value pairs (`parameterName=parameterValue`) in your query.

For example:

1. Start creating a Generic HTTP API Dataset Provider.
2. In the **Available endpoints** section, let's define two sets of parameters, as shown below:
   - In **URL**, enter: `https://${domain}/${path}`.
   - In **Headers**, set **Value** to: `${headerVal1} ${headerVal2}`.

(Note that you can use parameters to define the **Data field**, too.)



Generic HTTP API: Configuring dynamic query parameters

3. Use the new Dataset Provider to add a Generic HTTP API Dataset. Let's call the Dataset `my_http_api`.
4. Now, when searching the `my_http_api` Dataset, you'll be able to write queries like this one:

```
dataset="my_http_api" domain='cat-fact.herokuapp.com' path='facts' headerVal1='f
```

Copy

2.10. Connect Cribl Search To Microsoft Graph API

Configure Cribl Search to query a Microsoft Graph API endpoint.

With Microsoft Graph, you can access data across all Microsoft 365 services.

In this guide, you'll set up a Dataset Provider and a Dataset to search your Microsoft Entra ID or Microsoft 365 account(s) supporting the following endpoints:

- agreementAcceptances
- agreements
- applicationTemplates
- applications
- authenticationMethodConfigurations
- certificateBasedAuthConfiguration
- chats
- connections
- contacts
- contracts
- dataPolicyOperations
- devices
- directoryObjects
- directoryRoleTemplates
- directoryRoles
- domainDnsRecords
- domains
- drives
- filterOperators
- functions
- groupLifecyclePolicies
- groupSettingTemplates
- groupSettings
- groups

- identityProviders
- invitations
- localizations
- oauth2PermissionGrants
- organization
- permissionGrants
- places
- schemaExtensions
- scopedRoleMemberships
- servicePrincipals
- shares
- sites
- subscribedSkus
- subscriptions
- teamsTemplates
- teams
- users

Add a Microsoft Graph API Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add a Microsoft Graph API Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Microsoft Graph API**.
4. Select **Add Configuration** to specify your Microsoft Graph account(s).
    - **Account Name** is the Microsoft Graph account name.
    - **Tenant ID** is the ID of the Microsoft Entra ID or Microsoft 365 to retrieve information from.

- **Client ID** is the ID of the application that will connect to Microsoft Entra ID or Microsoft 365 account.
- **Client Secret** is the key that will be used as the secret in the connection to Microsoft Entra ID or Microsoft 365 account.

5. Select **Save** when finished.

Add a Microsoft Graph API Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of a Microsoft Graph API Dataset Provider.
4. Select **Add endpoint** to select the endpoints for your Dataset.
5. **Enabled endpoints**: Select an endpoint from the drop-down menu. For details on the endpoints, see the Microsoft Graph Rest API reference docs. Your options are:
   - `invitations`
   - `users`
   - `applicationTemplates`
   - `authenticationMethodConfigurations`
   - `identityProviders`
   - `applications`
   - `certificateBasedAuthConfiguration`
   - `contacts`
   - `contracts`
   - `devices`
   - `directoryObjects`
   - `directoryRoles`
   - `directoryRoleTemplates`
   - `domainDnsRecords`

- `domains`
- `groups`
- `groupSettings`
- `groupSettingTemplates`
- `localizations`
- `oauth2PermissionGrants`
- `organization`
- `permissionGrants`
- `scopedRoleMemberships`
- `servicePrincipals`
- `subscribedSkus`
- `places`
- `drives`
- `shares`
- `sites`
- `schemaExtensions`
- `groupLifecyclePolicies`
- `filterOperators`
- `functions`
- `agreementAcceptances`
- `agreements`
- `dataPolicyOperations`
- `subscriptions`
- `connections`
- `chats`
- `teams`
- `teamsTemplates`

6. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
7. In **Snapshots**, you can set up API Snapshots.
8. Select **Save** when finished.

Search Microsoft Graph API

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

2.11. Connect Cribl Search To Okta API

Configure Cribl Search to query an Okta API endpoint.

Okta is a cloud-based identity solution that links your apps, logins, and devices.

In this guide, you'll set up a Dataset Provider and a Dataset to search the users, apps, devices, groups, and logs endpoints in your Okta account.

Add an Okta API Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an Okta API Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Okta API**.
4. Select **Add Configuration** to specify your Okta account. In the **Account Configuration** table, enter:
    - **Account Name**: Okta account name.
    - **Domain Endpoint**: URL for the subdomain of your organization. For example, `subdomain.okta.com`. For details, see Find your Okta domain.
    - **API Token**: API token for authorizing requests. For details, see Create an API token.
5. Select **Save** when finished.

Add an Okta API Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can

use letters, numbers, and underscores (for example, `my_dataset_1`).

2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of an Okta Dataset Provider.
4. Select **Add endpoint** to select the endpoints for your Dataset.
5. **Enabled endpoints**: Select an endpoint from the drop-down menu. Your options are:
    - users
    - apps
    - devices
    - groups
    - logs
6. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
7. In **Snapshots**, you can set up API Snapshots.
8. Select **Save** when finished.

Search Okta API

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

2.12. Connect Cribl Search To OpenSearch

Configure Cribl Search to query an OpenSearch index.

OpenSearch is an open-source search engine for data-intensive applications.

In this guide, you'll set up a Dataset Provider and a Dataset to search an OpenSearch index.

Add an OpenSearch Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add an OpenSearch Dataset Provider.

To add a new Dataset Provider:

1. From the sidebar, select **Data**, then **Dataset Providers**.
2. Select the **Add Provider** button on the right.

Next, set the following configurations in the **New Dataset Provider** modal:

1. In **ID**, enter a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. Optionally, add **Description** to quickly describe the Dataset Provider.
3. Set **Dataset Provider Type** to **OpenSearch**.
4. Enter the following:
    - **Username**: Your OpenSearch account name.
    - **Password**: Your password for OpenSearch.
    - **Endpoint**: URL for the OpenSearch endpoint, for example: `https://opensearch.thecriblgoat.farm`.
5. Select **Save** when finished.

Add an OpenSearch Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Next, set the following configuration in the resulting **New Dataset** modal:

1. In **ID**, enter a unique identifier for the Dataset. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. Optionally, add **Description** to quickly describe the Dataset.
3. Set **Dataset Provider** to the **ID** of an OpenSearch Dataset Provider.
4. In **Index**, point to the identifier of the OpenSearch index you want to search. If you have rolling indices (indices with dynamic naming), use the name of the index and the wildcard character `*`. For example, if your indices are named `my-index-00001`, `my-index-00002`, and so on, use `my-index*` in this field.
5. Make sure **Timestamp field** contains the name of the field that contains event timestamps. In most cases, this will be `@timestamp` for time series data in OpenSearch.
6. Select **Save** when finished.

Search OpenSearch

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

OpenSearch Results Limit

The number of OpenSearch results is capped by the `max_result_window` setting of your OpenSearch index. By default, it's 10,000.

If you need to raise this limit, make sure you research and test for potential side effects.

2.13. Connect Cribl Search To Prometheus

Configure Cribl Search to query your Prometheus instance.

Prometheus is a an open-source toolkit that collects and stores metrics as time-series data.

In this guide, you'll set up a Dataset Provider and a Dataset to search data at rest in your Prometheus instance.

Add a Prometheus Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add a Prometheus Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Configure the resulting **New Dataset Provider** modal as follows:

1. In **ID**, enter a unique identifier for your new Dataset Provider. You'll reference this ID when adding Datasets. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. In **Description**, you can add a few words to make your new Dataset Provider easy to recognize.
3. Set **Dataset Provider Type** to **Prometheus**.
4. In **Endpoint**, enter the full URL of your Prometheus API endpoint, for example: `https://prometheus.goats.biz`. If you're using Grafana, the URL may look like this: `https://prometheus-foo-42-prod-us-central-0.grafana.net/api/prom`.
5. In **Concurrency limit**, set the maximum number of concurrent API requests. The default is `3`. Higher values may improve performance but put more load on the target service (with the risk of being rate limited).
6. In **Authentication type**, select how Cribl Search should authenticate to your Prometheus instance.
    - **Basic**: Select this if your Prometheus instance uses basic authentication. You'll need to enter a username and password.
    - **Token**: Select this if you're using token-based authentication. You'll need to enter a bearer token that Cribl Search will include with the authorization header.
7. Select **Save** when finished.

Add a Prometheus Dataset

After creating a Prometheus Dataset Provider, you can add a Prometheus Dataset. Datasets tell Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Configure the resulting **New Dataset** modal as follows:

Basic Configuration

1. In **ID**, enter an identifier unique for both Cribl Search and Cribl Lake. You'll reference this ID from the query box, when searching your new Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. In **Description**, you can add a few words to make your new Dataset easy to recognize.
3. Set **Dataset Provider** to the ID of a Prometheus Dataset Provider.

Prometheus Options

This exposes the following Prometheus-specific options:

1. **Number of data points**: How many data points you want to get each time you search this Dataset. Set this to a value between `1` and `11000` (a limit imposed by Prometheus). The default is `250`.
2. **Step mode**: Defines how to use the set **Number of data points** when calculating the interval between data points. Select one of the following:
    - **Exact number of data points**: Adjust the interval between data points to match the exact **Number of data points** set above.
    - **Snap to nearest unit**: Target the **Number of data points** set above, but snap the calculated interval up or down to the nearest Prometheus time unit. For example, `1043ms` will snap to `1s`, and `43 days` will snap to `6w`.
3. **Metric name pattern**: Optionally, enter a regular expression to filter metric names. Cribl Search will include only those metrics whose names match this pattern. This filtering lets you control access to your metrics, as described below.
4. Select **Save** when finished.

Control Access to Your Prometheus Metrics

If you want a group of users to have access only to a subset of your Prometheus metrics, you can add separate Prometheus Datasets with different **Metric name pattern** configurations.

For example, you can add a single Dataset Provider, but two separate Datasets:

1. Add a Prometheus Dataset Provider. This means both of your Prometheus Datasets can use the same authentication settings.
2. Add a Prometheus Dataset (say, `prometheus_dataset_1`). In the **Metric name pattern** field, add a regular expression that filters the names of your metrics, for example: `^abc`. When searching this

Dataset, Cribl Search will include only those metrics whose names start with `abc`.

3. Add another Prometheus Dataset (say, `prometheus_dataset_2`), with a different **Metric name pattern**, for example: `^xyz`.
4. Share `prometheus_dataset_1` with one group of users (or a Team), and `prometheus_dataset_2` with another. Use the **Read Only** permission.
5. Now, one group of users can search only the `abc` metrics, and another only the `xyz` metrics.

List the Contents of a Prometheus Dataset

You can quickly list all metrics accessible through a Prometheus Dataset, by using the `.show objects` command. For example:

```
.show objects(prometheus_dataset_ID)
```

Copy

This returns a list of metrics that your Prometheus Dataset consists of. Each metric comes with the following fields:

Field	Description
`_time`	The `.show objects` command's execution time. It's the same for every returned metric.
`success`	Shows whether the command was successful.
`dataset`	The ID of the examined Dataset.
`datasetType`	The type of the examined Dataset.
`name`	The name of the metric.

Additionally, if your Prometheus instance supports the `/api/v1/targets/metadata` API, you'll also get the following metadata fields:

Field	Description
`type`	The type of the metric. For example, `counter`.
`unit`	What units the metric value conveys (usually empty).
`help`	A description of the metric.

Now, you can use the fields above to filter for specific metrics. For example, to list only gauge metrics:

```
.show objects(prometheus_dataset_ID)
 | where type == "gauge"
```

Copy

Search a Prometheus Dataset

After creating a Prometheus Dataset Provider and Dataset, you're ready to start searching. See the examples just below.

Search for any metrics whose names, labels, or label values start with `node_`:

```
dataset="prometheus_dataset_ID" node_*
```

Copy

Search the metric named `node_cpu_seconds_total` for values higher than `42`:

```
dataset="prometheus_dataset_ID" node_cpu_seconds_total > 42
```

Copy

Filter by multiple fields, and project only the `_time` and `cpu` fields:

```
dataset="prometheus_dataset_ID" node_cpu_seconds_total > 42 mode=idle cpu=7
 | project _time, cpu
```

Copy

Alternatively, you can search by using the Prometheus `__name__` field:

```
dataset="prometheus_dataset_ID" __name__=node_*
```

Copy

You can also search by Prometheus labels alone. This allows you to search any metrics that have the given label values, for example:

```
dataset="prometheus_dataset_ID" mode=idle cpu=7
```

Copy

Prometheus Results

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in your Prometheus instance.

Each data point returned may include the following fields:

Field	Description
Built-in fields	Cribl Search's meta fields, such as `_raw` or `_time`. They're not part of the metrics, but can be referenced in queries.
Metric + value	The name of the queried metric, along with its value.
`__name__`	Prometheus's field that contains the name of the queried metric.
`step`	The `step` value used for the Prometheus query request, which translates to the interval between data points returned.
Others	Any other labels that apply to this data point, which show up as event fields.

Additionally, if your Prometheus instance supports the `/api/v1/targets/metadata` API, you'll also get a `metadata` field, which contains the following:

Field	Description
`type`	The type of the metric. For example, `counter`.
`unit`	What units the metric value conveys (usually empty).
`help`	A description of the metric.

Prometheus: Number of Data Points Returned

When searching a Prometheus Dataset, you can can decide **how many data points** you want to spread across the time range of your search.

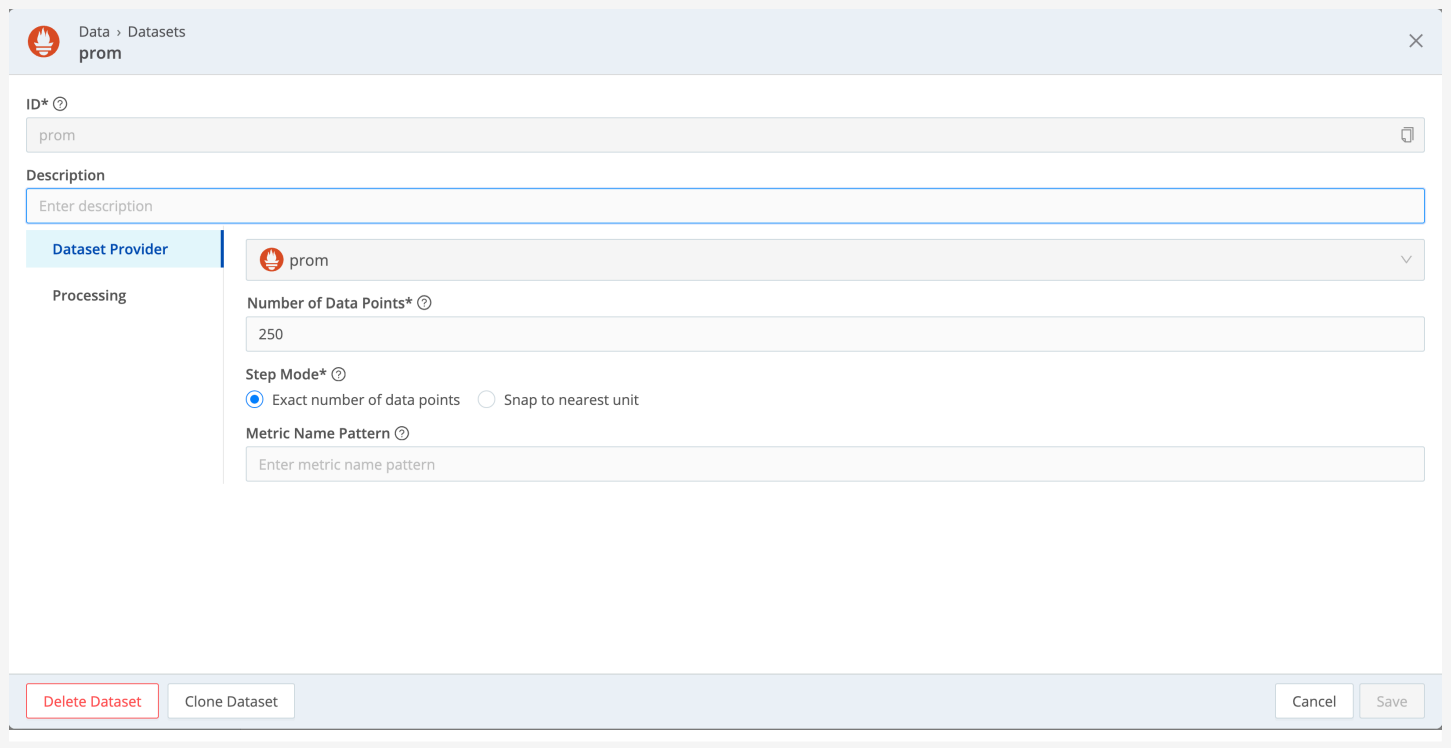
For example, a search can produce 500 data points for a 1-hour time range, giving you 1 data point every 7.2 seconds.

To adjust this to your needs, you can:

- Set the default number of data points returned. Cribl Search will adjust the interval between data points to match this number.
- Optionally, set a specific interval between data points for a given search. Cribl Search will then adjust the number of data points returned to match the specified interval.

Prometheus: Set the Default Number of Data Points Returned

You can set the default number of data points when editing or creating your Prometheus Dataset. Enter the desired value into the **Number of data points** field.



Configuring the number of data points returned

With **Step mode**, you can tell Cribl Search how to use the set **Number of data points** when calculating the interval between data points.

To always get the same, exact number of data points, set **Step mode** to **Exact number of data points**. Then, if you set the **Number of data points** to `250`, a search will always give you 250 data points, spread across the time range of your search. For example, searching a 60-day time range will give you exactly 250 data points, with the interval of 5.76 hours between them.

To round the interval to the nearest time unit, set **Step mode** to **Snap to nearest unit**. Then, if you set the **Number of data points** to `250`, and search a 60-day time range, Cribl Search will snap the 5.76-hour interval to 6 hours, and adjust the number of data points accordingly.

Prometheus: Set the Interval Between Data Points Returned (`step`)

Every Prometheus search can override the calculated interval between data points, by using an explicit `step` value. For example:

```
dataset="prometheus_dataset_ID" step="30s"
```

Copy

This will adjust the number of data points returned to match the specified interval.

The value of `step` must match the following pattern:

```
^\\d+(ms|s|m|h|d|w|y)$

// for example:
// 10s
// 5m
```

Copy

2.14. Connect Cribl Search To Snowflake

Configure Cribl Search to query your Snowflake warehouse.

Snowflake is a cloud platform that offers data warehousing as a service.

In this guide, you'll set up a Dataset Provider and a Dataset to search a Snowflake warehouse.

Add a Snowflake Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add a Snowflake Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Next, configure the resulting **New Dataset Provider** modal as follows:

1. In **ID**, enter a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. Optionally, add **Description** to quickly describe the Dataset Provider.
3. Set **Dataset Provider Type** to **Snowflake**.
4. Enter the following:
    - **Account identifier**: Case-sensitive Snowflake account identifier, in the format `<org_name>-<account_name>`. For example: `myorg-account123`.
    - **Username**: Case-sensitive Snowflake username.
    - **Private key**: The contents of your Snowflake private key (PEM) file. To upload a file, select the upload button at this field's upper right.
    - **Private key passphrase**: An optional passphrase used to protect your Snowflake private key.
    - **Executors limit**: Maximum number of concurrent Cribl Search executors dispatched for processing Snowflake data partitions.
    - **Endpoint**: Optional URL to Snowflake REST API. If empty, defaults to `https://<org_name>-<account_name>.snowflakecomputing.com`.
5. Select **Save** when finished.

Add a Snowflake Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Next, configure the resulting **New Dataset** modal as follows:

1. In **ID**, enter a unique identifier for the Dataset. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. Optionally, add **Description** to quickly describe the Dataset.
3. Set **Dataset Provider** to the ID of a Snowflake Dataset Provider.
4. Enter the following:
    - **Warehouse**: Case-sensitive name of the Snowflake warehouse. If not set, defaults to the `DEFAULT_WAREHOUSE` property for the Snowflake user.
    - **Database name**: Case-sensitive name of the Snowflake database. If not set, defaults to the database value in the `DEFAULT_NAMESPACE` for the Snowflake user.
    - **Schema name**: Case-sensitive name of the Snowflake schema that contains the tables you plan to query. If not set, defaults to the schema value in the `DEFAULT_NAMESPACE` for the Snowflake user.
    - **Table or query**: The name of the table or view, or a query. May be a simple name (`logs`) or an SQL query (`select * from logs`).
    - **Timestamp field**: Optional, case-sensitive name of the column that holds the timestamp of the event to query.
    - **Role**: Optional Snowflake role to use when running the SQL query. We recommend using a read-only role.
5. Select **Save** when finished.

List the Contents of a Snowflake Dataset

You can quickly list all tables and views accessible through a Snowflake Dataset, by using the `.show objects` command. For example:

```
.show objects(snowflake_dataset_ID)
```

Copy

Search a Snowflake Dataset

Now that you have a Snowflake Dataset Provider and Dataset, you're ready to start searching.

Cribl Search submits the query to Snowflake. Results will start showing up after the Snowflake SQL query completes. How long it takes depends on how much data there is in your Snowflake warehouse. Searching

columns that contain large strings may result in a slower search.

Snowflake: Supported SQL Data Types

When querying Snowflake, Cribl Search supports the following SQL data types:

- `NUMBER`, `DECIMAL`, `DEC`, `NUMERIC`, `INT`, `INTEGER`, `BIGINT`, `SMALLINT`, `TINYINT`, `BYTEINT`, `FLOAT`, `FLOAT4`, `FLOAT8`, `DOUBLE`, `DOUBLE PRECISION`, `REAL`
- `VARCHAR`, `CHAR`, `CHARACTER`, `STRING`, `TEXT`
- `BOOLEAN`
- `DATE`, `DATETIME`, `TIME`, `TIMESTAMP`, `TIMESTAMP_LTZ`, `TIMESTAMP_NTZ`, `TIMESTAMP_TZ`
- `VARIANT`, `OBJECT`

2.15. Connect Cribl Search To Tailscale API

Configure Cribl Search to query a Tailscale API endpoint.

Tailscale VPN makes your devices and applications accessible from anywhere in the world, securely and easily.

In this guide, you'll set up a Dataset Provider and a Dataset to search the acls and devices endpoints in your Tailscale account.

Add a Tailscale API Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add a Tailscale API Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Tailscale API**.
4. Select **Add Configuration** to specify your Tailscale account. In the **Account Configuration** table, enter:
    - **Account Name**: Tailscale account name.
    - **Client ID**: Tailscale Client ID.
    - **Client Secret**: Tailscale Client Secret.
5. Select **Save** when finished.

For details on obtaining your Tailscale credentials, see Authentication.

Add a Tailscale API Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of a Tailscale Dataset Provider.
4. Select **Add endpoint** to select the endpoints for your Dataset.
5. **Enabled endpoints**: Select an endpoint from the drop-down menu. Your options are:
    - acls
    - devices
6. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
7. In **Snapshots**, you can set up API Snapshots.
8. Select **Save** when finished.

Search Tailscale API

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

2.16. Connect Cribl Search To Zoom API

Configure Cribl Search to query a Zoom API endpoint.

Zoom enables users to communicate via video, audio, phone, and chat.

In this guide, you'll set up a Dataset Provider and a Dataset to search the user, group, and meeting endpoints in your Zoom account.

Add a Zoom API Dataset Provider

A Dataset Provider tells Cribl Search where to query and contains access credentials. Here, you will add a Zoom API Dataset Provider.

To add a new Dataset Provider, select **Data**, then **Dataset Providers**, then **Add Provider**.

Set the following configurations in the **New Dataset Provider** modal:

1. **ID** is a unique identifier for the Dataset Provider. This is how you'll reference it when assigning Datasets to it. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_provider_1`).
2. **Description** is optional.
3. Set **Dataset Provider Type** to **Zoom API**.
4. Select **Add Configuration** to specify your Zoom account. In the **Account Configuration** table, enter:
    - **Account Name**: Zoom account name.
    - **Account ID**: Unique ID associated with your Zoom account.
    - **Client ID**: Zoom Client ID.
    - **Client Secret**: Zoom Client Secret.
5. Select **Save** when finished.

For details on obtaining your Zoom credentials, see Server-to-Server OAUTH.

Add a Zoom API Dataset

Now you'll add a Dataset that tells Cribl Search what data to search from the Dataset Provider.

To add a new Dataset, select **Data**, then **Datasets**, then **Add Dataset**.

Set the following configurations in the **New Dataset** modal:

1. **ID** is an identifier unique for both Cribl Search and Cribl Lake. You'll use this to specify the Dataset in a query's scope, telling Cribl Search to search the Dataset. Start the ID with a letter; the rest of the ID can use letters, numbers, and underscores (for example, `my_dataset_1`).
2. **Description** is optional.
3. Set **Dataset Provider** to the **ID** of a Zoom Dataset Provider.
4. Select **Add endpoint** to select the endpoints for your Dataset.
5. **Enabled endpoints**: Select an endpoint from the drop-down menu. Your options are:
    - users
    - groups
    - meetings
    - pastMeetings
6. In **Processing**, you can apply rules for breaking data into discrete events. For more information, see Datatypes.
7. In **Snapshots**, you can set up API Snapshots.
8. Select **Save** when finished.

Search Zoom API

Now that you have a Dataset Provider and Dataset, you're ready to start searching.

Search results can start showing up within a second or two, but when the search completes depends on how much data there is in the account.

2.17. Configure Data Connection Options In Cribl Search

Process data into discrete events, control your object storage costs, and learn about how Cribl Search processes the most common data formats.

Datatypes

Make searching easier by processing data into discrete events and defining fields. See: Datatypes.

Storage Classes

Control your costs by selectively enabling which object storage classes (access tiers) you want each Dataset to search. See: Storage Classes.

Supported Data Formats

Learn about how Cribl Search processes the most common data formats. See: Supported Data Formats.

2.17.1. Datatypes

Make searching easier by processing data into discrete events and defining fields.

Datatypes are sets of rules that Cribl Search follows to break data (bytes) from Datasets into discrete events, timestamp them, and parse them as needed. This helps with categorizing events and makes searching easier via addressing through a `datatype` field.

Cribl Search ships with a number of built-in Datatypes, and you can author, test, and validate custom Datatype rules using the Cribl Search UI.

With appropriate Permissions, you can also export, share, and import Datatypes in Packs.

Using Datatypes

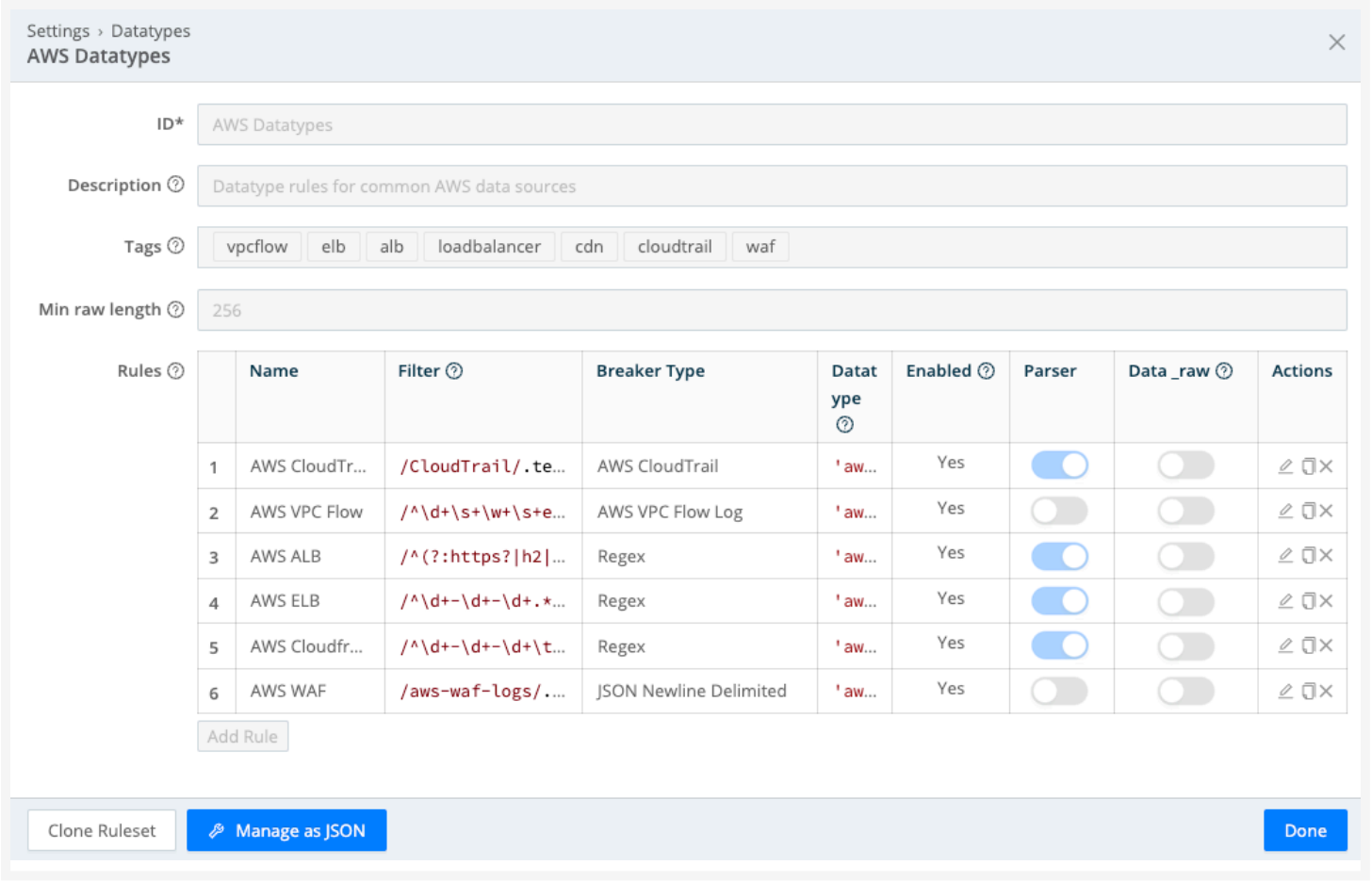
To search a Dataset with a specific rule's results, use the `Datatype` field. For example, `dataset=myDataset Datatype=aws_vpcflow`. Note that the value is not the `ID` of a Datatype – rather, it is the `Datatype` of the rule.

A Dataset can be associated with one or more Datatype rulesets. rulesets are evaluated top-down, and they consist of an ordered list of rules that are also evaluated top-down.

Cribl Search will attempt to match the data against all the ordered rules, and it will process the search results using the first rule that matches.

When you write data to an object store for later analysis by Cribl Search, ensure that events are homogeneously formatted within each file. This consistent structure is necessary in order for Cribl Search to accurately apply its Event Breakers and Parsers. Cribl Search determines a file's Datatype by evaluating the file's initial data against Datatype filter conditions, and then applies the first-matched Datatype to the remainder of the file.

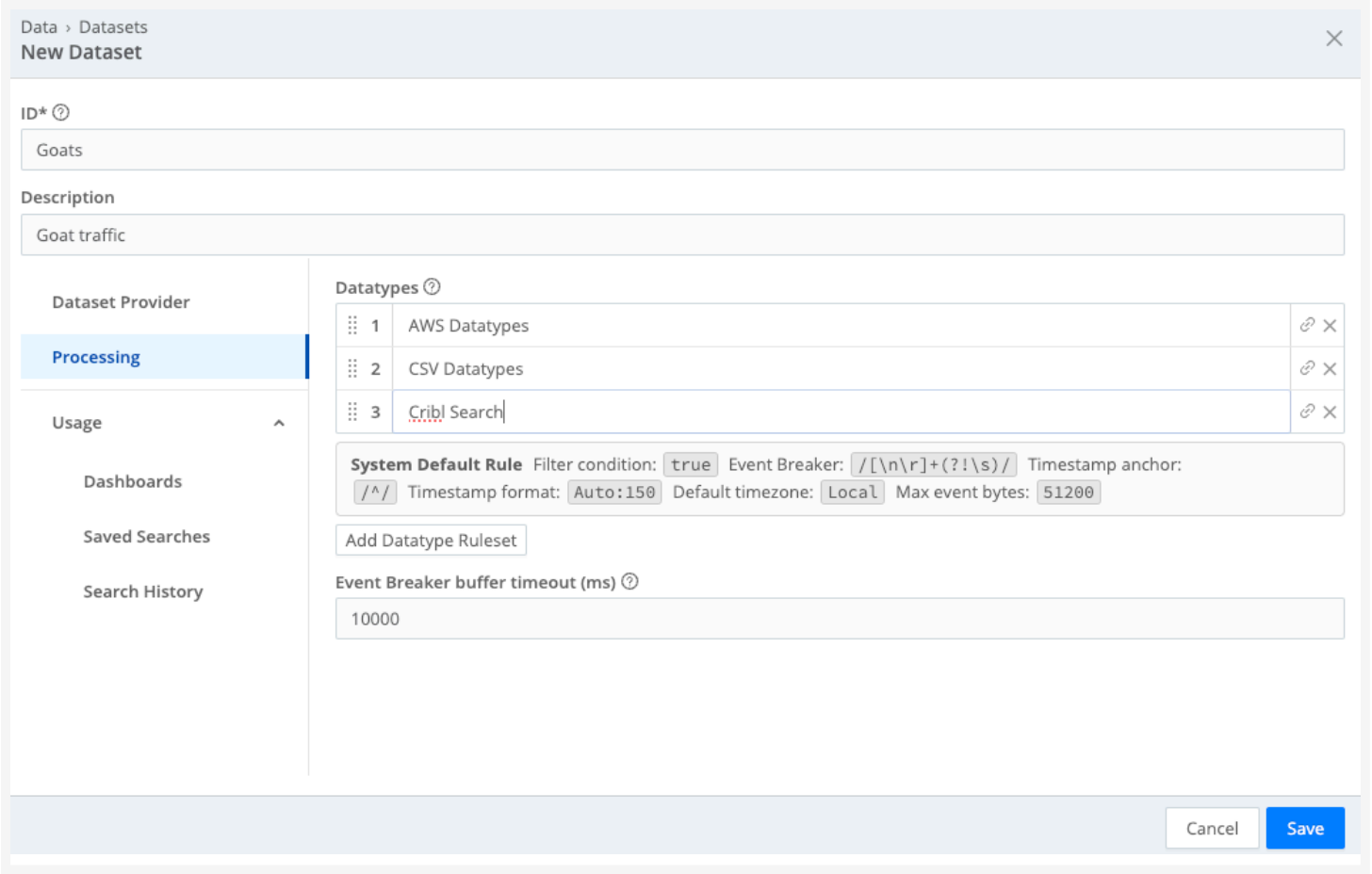
Here's an example of a Datatype ruleset called "AWS Datatypes":



AWS Datatypes

Datatyping is applied before search terms - meaning that if a field is added, for example through parsing, you can reference it in the query, such as in filter or projected terms.

Here's an example of a Dataset configured with multiple Datatype rulesets:



Dataset with wildcarded Datatype rulesets

Use wildcards * to specify multiple Datatype rulesets with similar names. Matching is case sensitive. For example,

- A* matches Datatype IDs starting with the letter A.
- *A matches Datatype IDs ending with the letter A.
- *A* matches Datatype IDs with the letter A.

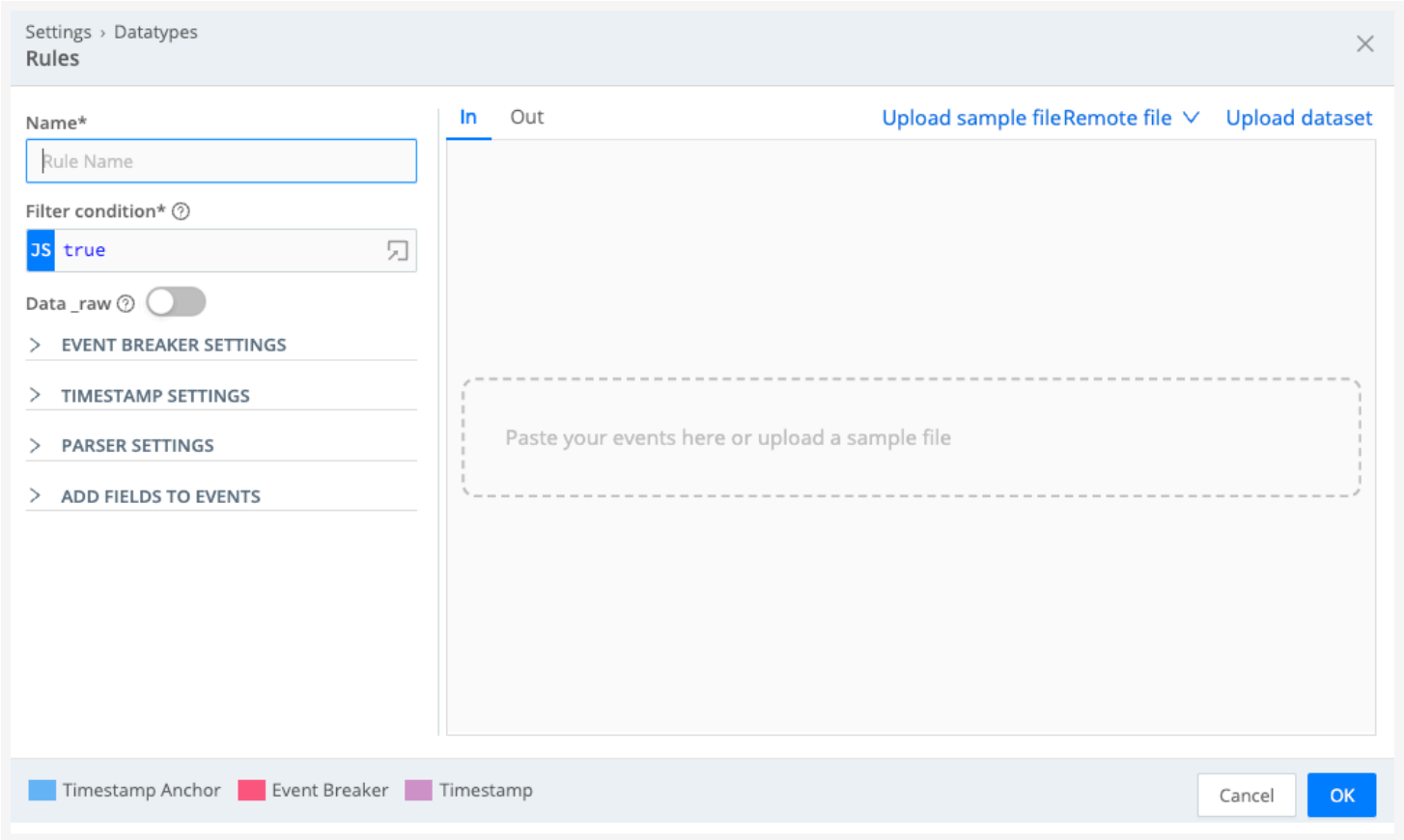
Order of Operations

The process of datatyping is as follows:

- Event Breaking – Breaks raw bytes into discrete events.
- Timestamping – Assigns timestamps to events.
- Parsing – Parses fields from events.
- Add Fields – Adds additional fields.

Create a rule

On the **Datatypes** page, select an existing Datatype ruleset, or select **Add Datatype** to create a new ruleset. Within a Datatype configuration modal, select **Add Rule** to open the **Rules** modal shown below:



Adding a new rule

Each rule includes the following components.

Filter Condition

As data is processed, a rule's filter expression (JavaScript) is applied. If the expression evaluates to `true`, the rule configurations are engaged on the Dataset. Else, the next rule down the line is evaluated.

Event Breaker Settings

Event Breakers configure how Cribl Search converts data into discrete events. Cribl Search provides several different formats for Event Breakers. See below for specific information on different Event Breaker Types.

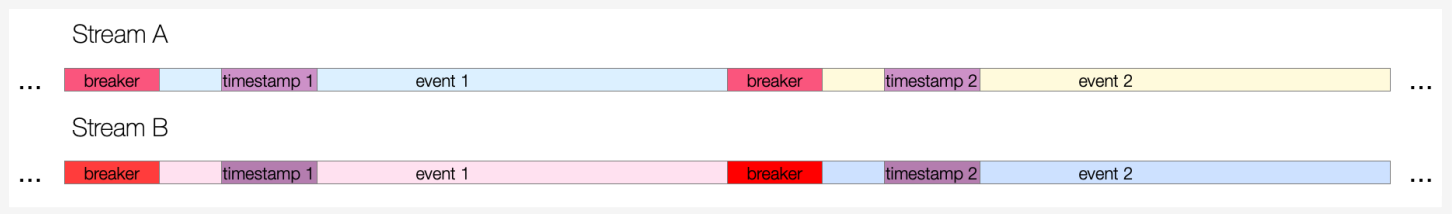
Timestamp Settings

After data is processed into events, Cribl Search will attempt timestamping. First, a timestamp anchor will be located inside the event. Next, starting there, the engine will try to do one of the following:

- Scan up to a configurable depth into the event and autotimestamp, **or**
- Timestamp using a manually supplied `strptime` format, **or**

- Timestamp the event with the current time.

The closer an anchor is to the timestamp pattern, the better the performance and accuracy – especially if multiple timestamps exist within an event. For the manually supplied option, the anchor must lead the engine **right before** the timestamp pattern begins.



Anchors preceding timestamps

Parser Settings

The Parser associated with a rule can be used to extract fields out of events that you can reference in searches. Each field in the parsed event contains a key-value pair, where the field name is the key.

Parsers are very similar to the Parser function that can be configured in Cribl Stream pipelines.

You can configure a new Parser or select one from the library to configure. Based on the **Type** selected, you'll have the following configurations:

- **Type**: Select the format of your data, CSV, `Extended Log File Format`, `Common Log Format`, `Key=Value Pairs`, `JSON Object`, `Delimited values`, `Regular Expression`, or `Grok`.
- **Library**: Select a Parser defined in the library. The library is located at **Knowledge** > **Parsers**.
- **Source field**: Define the field the Parser will use to extract from. Defaults to `_raw`.
- **Destination field**: Provide a name for the field to which the extracted data will be assigned.
- **List of fields**: The fields to be extracted in the desired order. If you don't specify any fields, the Parser will generate them automatically.
- **Fields to keep**: The fields to keep. Supports wildcards `*`. Takes precedence over **Fields to remove**.
- **Fields to remove**: The fields to remove. Supports wildcards `*`. You cannot remove fields that match **Fields to keep**.
- **Fields filter expression**: Expression evaluated against `{index, name, value}` context. Return `truthy` to keep a field, or `falsy` to remove it.

The following configuration lists are categorized by **Type**:

Key=Value Pairs

- **Clean fields**: Whether to clean field names by replacing non-alphanumeric characters `[a-zA-Z0-9]` with an underscore `_`.

Advanced Settings

- **Allowed key characters**: A list of characters that can appear in a key name, even though they're normally separator or control characters.
- **Allowed value characters**: A list of characters that can appear in a value name, even though they're normally separator or control characters.

Delimited Values

- **Delimiter**: Delimiter character to use to split values.
- **Quote character**: Character used to quote values. Required if values contain the delimiter character.
- **Escape character**: Character used to escape characters within a value.
- **Null value**: String value that should be treated as null or undefined.

Regular Expression

- **Regex**: Regex literal with named capturing groups, for example, `(?<foo>bar)`. Or with `_NAME_` and `_VALUE_` capturing groups, for example, `(?<_NAME_0>[^ =]+)=(?<_VALUE_0>[^,]+)`.
- **Additional regex**: Add another regular expression to match against other data.
- **Max exec**: The maximum number of times to apply the regex to the source field when the global flag is set, or when using named capturing groups.
- **Field name format expression**: JavaScript expression to format field names when `_NAME_n` and `_VALUE_n` capturing groups are used. The original field name is in the global variable `name`. For example, to append `XX` to all field names: `${name}_XX` (backticks are literal). If empty, names will be sanitized using this regex: `/^[_0-9]+|[^a-zA-Z0-9_]+/g`. You can access other field values via `__e.<fieldName>`.
- **Overwrite existing fields**: Overwrite existing event fields with extracted values. If set to **No**, existing fields will be converted to an array.

Grok

- **Pattern**: Grok pattern to extract fields. Syntax supported: `%{PATTERN_NAME:FIELD_NAME}`.
- **Additional Grok patterns**: Add another pattern to match other data.

Add Fields to Events

After events have been timestamped, one or more fields can be added here as key-value pairs. In each field's **Value Expression**, you can fully evaluate the field value using JavaScript expressions. Each rule should be configured to add one field named `datatype` and its corresponding value. This way, it will be possible to determine which Datatype each event belongs to by looking at that field.

Event Breaker Types

Several types of Event Breakers can be applied to data:

- Regex
- JSON New Line Delimited
- JSON Array
- File Header
- Timestamp
- CSV
- AWS CloudTrail
- AWS VPC Flow Log



For details about who can access and modify these resources, see Search Member Permissions.

Regex Event Breaker

The **Regex** breaker uses regular expressions to find breaking points in data.

Breaking will occur at the beginning of the match, and the matched content will be consumed/thrown away. If necessary, you can use a positive lookahead regex to keep the content – for example: `(?=pattern)`

Capturing groups are **not allowed** to be used anywhere in the Event Breaker pattern, as they will further break the data – which is often undesirable. Breaking will also occur if **Max Event Bytes** has been reached.



The highest **Max Event Bytes** value that you can set is about 128 MB (`134217728` bytes). Events exceeding the maximum will be split into separate events, but left unbroken. Cribl Search will set these events' `__isBroken` internal field to `false`.

Example

With the following configurations, let's see how this sample data is datatyped:

- **Event Breaker:** – `[\n\r]+`

- **Parser** – `AWS VPC Flow Logs` from the Library.
- **Add Field** – `"datatype": "aws_vpcflow"`

```
--input--
2 123456789012 eni-0b2fc5457066bc156 10.0.0.164 54.239.152.25 41050 443 6 8 1297 1679
2 123456789013 eni-0b2fc5457066bc157 10.0.0.164 54.239.152.25 41050 443 6 8 1297 1679

--- output event 1 ---
{
 "_raw": "2 123456789012 eni-0b2fc5457066bc156 10.0.0.164 54.239.152.25 41050 443 6 8
 "_time": 1679339697,
 "datatype": "aws_vpcflow",
 "version": "2",
 "account_id": "123456789012",
 "interface_id": "eni-0b2fc5457066bc156",
 "srcaddr": "10.0.0.164",
 "dstaddr": "54.239.152.25",
 "srcport": "41050",
 "dstport": "443",
 "protocol": "6",
 "packets": "8",
 "bytes": "1297",
 "start": "1679339697",
 "end": "1679340000",
 "action": "ACCEPT",
 "log_status": "OK",
 "dataset": "s3_vpcflowlogs",
}

--- output event 2 ---
{
 "_raw": "2 123456789013 eni-0b2fc5457066bc157 10.0.0.164 54.239.152.25 41050 443 6 8
 "_time": 1679339698,
 "datatype": "aws_vpcflow",
 "version": "2",
 "account_id": "123456789013",
 "interface_id": "eni-0b2fc5457066bc156",
 "srcaddr": "10.0.0.164",
 "dstaddr": "54.239.152.25",
 "srcport": "41050",
 "dstport": "443",
 "protocol": "6",
 "packets": "8",
 "bytes": "1297",
 "start": "1679339698",
```

```
 "end": "1679340000",
 "action": "REJECT",
 "log_status": "OK",
 "dataset": "s3_vpcflowlogs",
}
```

Copy

JSON New Line Delimited Event Breaker

You can use the **JSON New Line Delimited** breaker to break and extract fields in newline-delimited JSON data.

Example

Using default values, let's see how this sample data breaks up:

Sample Event - Newline Delimited JSON Breaker

```
--- input ---
{"time":"2020-05-25T18:00:54.201Z","cid":"w1","channel":"clustercomm","level":"info",
{"time":"2020-05-25T18:00:54.246Z","cid":"w0","channel":"clustercomm","level":"info",


--- output event 1 ---
{
  "_raw": "{\"time\":\"2020-05-25T18:00:54.201Z\",\"cid\":\"w1\",\"channel\":\"cluste
  "time": "2020-05-25T18:00:54.201Z",
  "cid": "w1",
  "channel": "clustercomm",
  "level": "info",
  "message": "metric sender",
  "total": 720,
  "dropped": 0,
  "_time": 1590429654.201,
}

--- output event 2 ---
{
  "_raw": "{\"time\":\"2020-05-25T18:00:54.246Z\",\"cid\":\"w0\",\"channel\":\"cluste
  "time": "2020-05-25T18:00:54.246Z",
  "cid": "w0",
  "channel": "clustercomm",
  "level": "info",
  "message": "metric sender",
  "total": 720,
  "dropped": 0,
  "_time": 1590429654.246,
}
```

Copy

JSON Array Event Breaker

You can use the **JSON Array** breaker to extract events from an array in a JSON document (for example, an Amazon CloudTrail file).

- **Array Field**: Optional path to array in a JSON event with records to extract. For example, `Records`.
- **Timestamp Field**: Optional path to timestamp field in extracted events. For example, `eventTime` or `level1.level2.eventTime`.

- **JSON Extract Fields**: Enable this slider to auto-extract fields from JSON events. If disabled, only `_raw` and `time` will be defined on extracted events.
- **Timestamp Format**: If **JSON Extract Fields** is set to **No**, you **must** set this to **Autotimestamp** or **Current Time**. If **JSON Extract Fields** is set to **Yes**, you can select any option here.

Example

Using the values above, let's see how this sample data breaks up:

Sample Event - JSON Document (Array)

```
--- input ---
{"Records":[{"eventVersion":"1.05","eventTime":"2020-04-08T01:35:55Z","eventSource":'
{"eventVersion":"1.05","eventTime":"2020-04-08T01:35:56Z","eventSource":"ec2.amazonav

--- output event 1 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-08T01:35:55Z\",\"eventS
  "_time": 1586309755,
  "cribl_breaker": "j-array"
}

--- output event 2 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-08T01:35:56Z\",\"eventS
  "_time": 1586309756,
  "cribl_breaker": "j-array"
}
```

Copy

File Header Event Breaker

You can use the **File Header** breaker to break files with headers, such as IIS or Bro logs. This type of breaker relies on a header section that lists field names. The header section is typically present at the top of the file, and can be single-line or greater.

After the file has been broken into events, fields will also be extracted, as follows:

- **Header Line**: Regex matching a file header line. For example, `^#`.
- **Field Delimiter**: Field delimiter regex. For example, `\s+`.

- **Field Regex**: Regex with one capturing group, capturing all the fields to be broken by field delimiter. For example, `^ields[:]?\s+(.*)`
- **Null Values**: Representation of a null value. Null fields are not added to events.
- **Clean Fields**: Whether to clean up field names by replacing non-alphanumeric characters `[a-zA-Z0-9]` with an underscore `_`.

Example

Using the values above, let's see how this sample file breaks up:

Sample Event - File Header

```
--- input ---
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
#types  time    string  addr    port    addr    port    enum
1331904608.080000       -     192.168.204.59  137     192.168.204.255 137     udp
1331904609.190000       -     192.168.202.83  48516   192.168.207.4   53      udp


--- output event 1 ---
{
  "_raw": "1331904608.080000       -     192.168.204.59  137     192.168.204.255 137
  "ts": "1331904608.080000",
  "id_orig_h": "192.168.204.59",
  "id_orig_p": "137",
  "id_resp_h": "192.168.204.255",
  "id_resp_p": "137",
  "proto": "udp",
  "_time": 1331904608.08
}

--- output event 2 ---
{
  "_raw": "1331904609.190000       -     192.168.202.83  48516   192.168.207.4   53
  "ts": "1331904609.190000",
  "id_orig_h": "192.168.202.83",
  "id_orig_p": "48516",
  "id_resp_h": "192.168.207.4",
  "id_resp_p": "53",
  "proto": "udp",
  "_time": 1331904609.19
}
```

Copy

Timestamp Event Breaker

You can use the **Timestamp** breaker to break events at the beginning of any line in which Cribl Search finds a timestamp. This type enables breaking on lines whose timestamp pattern is not known ahead of time.

Example

Using default values, let's see how this sample data breaks up:

Sample Event - Timestamp Based Breaker

```
--- input ---
{"level":"debug","ts":"2021-02-02T10:38:46.365Z","caller":"sdk/sync.go:42","msg":"Har
{"level":"debug","ts":"2021-02-02T10:38:56.365Z","caller":"sdk/sync.go:42","msg":"Har


--- output event 1 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-02T10:38:46.365Z\",\"caller\":\"sdk/
  "_time": 1612262326.365
}

--- output event 2 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-02T10:38:56.365Z\",\"caller\":\"sdk/
  "_time": 1612262336.365
}
```

Copy

CSV Event Breaker

The **CSV** breaker extracts fields in CSV data that include a header line. Selecting this type exposes these extra fields:

- **Delimiter**: Delimiter character to use to split values. Defaults to: `,`.
- **Quote Char**: Character used to quote literal values. Defaults to: `"`.
- **Escape Char**: Character used to escape the quote character in field values. Defaults to: `"`.

Example

Using default values, let's see how this sample data breaks up:

Sample Event - CSV Breaker

```
--- input ---
time,host,source,model,serial,bytes_in,bytes_out,cpu
1611768713,"myHost1","anet","cisco","ASN4204269",11430,43322,0.78
1611768714,"myHost2","anet","cisco","ASN420423",345062,143433,0.28


--- output event 1 ---
{
  "_raw": "\"1611768713\",\"myHost1\",\"anet\",\"cisco\",\"ASN4204269\",\"11430\",\"4
  "time": "1611768713",
  "host": "myHost1",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN4204269",
  "bytes_in": "11430",
  "bytes_out": "43322",
  "cpu": "0.78",
  "_time": 1611768713
}

--- output event 2 ---
{
  "_raw": "\"1611768714\",\"myHost2\",\"anet\",\"cisco\",\"ASN420423\",\"345062\",\"1
  "time": "1611768714",
  "host": "myHost2",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN420423",
  "bytes_in": "345062",
  "bytes_out": "143433",
  "cpu": "0.28",
  "_time": 1611768714
}
```

Copy

With **Type: CSV** selected, an Event Breaker will properly add quotes around all values, regardless of their

initial state.

AWS CloudTrail Event Breaker

Use the **AWS CloudTrail** breaker to break and parse AWS CloudTrail events much faster.

See the example of input and output events below.

Sample Event - AWS CloudTrail Breaker

```
--- input ---
{"Records":[{"eventVersion":"1.0","userIdentity":{"type":"IAMUser","principalId":"EX_

--- output event 1 ---
{
  "_raw": "{\"eventVersion\":\"1.0\",\"userIdentity\":{\"type\":\"IAMUser\",\"princip
  "_time": 1711112256.322
}

--- output event 2 ---
{
  "_raw": "{\"eventVersion\":\"1.0\",\"userIdentity\":{\"type\":\"IAMUser\",\"princip
  "_time": 1711112355.478
}

--- output event 3 ---
{
  "_raw": "{\"eventVersion\":\"1.0\",\"userIdentity\":{\"type\":\"IAMUser\",\"princip
  "_time": 1711112355.478
}
```

Copy

AWS VPC Flow Log Event Breaker

Use the **AWS VPC Flow Log** breaker to break and parse AWS VPC Flow Log events as fast as possible.

This breaker works properly with files formatted as follows:

- Each file starts with a header. That header is in the same format as other lines but describes the column names.
- Every line has the same number of columns.

- Each column is separated by the space ( ) character. There is no space before the first or after the last column.
- Each line ends with the newline character (`\n`).
- A single dash (`-`) indicates a null value for the column where it appears.

See the example of input and output events below.

Sample Event - AWS VPC Flow Log Breaker

```
--- input ---
version account_id interface_id srcaddr dstaddr srcport dstport protocol packets byte
2 123456789014 eni-07e5d76940fbaa5c6 231.234.233.57 83.70.34.149 19856 15726 17 3048
2 123456789010 eni-0858304751c757b2e 31.200.171.164 22.109.125.129 58551 456 17 10824
2 123456789013 eni-07e5d76940fbaa5c6 78.128.152.196 67.42.235.136 25164 29139 17 5380

--- output event 1 ---
{
  "_raw": "2 123456789014 eni-07e5d76940fbaa5c6 231.234.233.57 83.70.34.149 19856 157
  "version": "2",
  "account_id": "123456789014",
  "interface_id": "eni-07e5d76940fbaa5c6",
  "srcaddr": "231.234.233.57",
  "dstaddr": "83.70.34.149",
  "srcport": "19856",
  "dstport": "15726",
  "protocol": "17",
  "packets": "3048",
  "bytes": "380",
  "start": "1262332800",
  "end": "1262332900",
  "action": "ACCEPT",
  "log_status": "NODATA",
  "_time": 1611768713
}

--- output event 2 ---
{
  "_raw": "2 123456789010 eni-0858304751c757b2e 31.200.171.164 22.109.125.129 58551 4
  "version": "2",
  "account_id": "123456789010",
  "interface_id": "eni-0858304751c757b2e",
  "srcaddr": "31.200.171.164",
  "dstaddr": "22.109.125.129",
  "srcport": "58551",
  "dstport": "456",
  "protocol": "17",
  "packets": "10824",
  "bytes": "352",
  "start": "1262332801",
  "end": "1262333301",
  "action": "ACCEPT",
```

```
  "log_status": "OK",
  "_time": 1611768714
}

--- output event 3 ---
{
  "_raw": "2 123456789013 eni-07e5d76940fbaa5c6 78.128.152.196 67.42.235.136 25164 29
  "version": "2",
  "account_id": "123456789013",
  "interface_id": "eni-07e5d76940fbaa5c6",
  "srcaddr": "78.128.152.196",
  "dstaddr": "67.42.235.136",
  "srcport": "25164",
  "dstport": "29139",
  "protocol": "17",
  "packets": "5380",
  "bytes": "160",
  "start": "1262332802",
  "end": "1262332902",
  "action": "REJECT",
  "log_status": "OK",
  "_time": 1611768715
}
```

Copy

With **Type: AWS VPC Flow Log** selected, an Event Breaker will properly add quotes around all values, regardless of their initial state.

Cribl rulesets versus Custom rulesets

Datatype rulesets shipped by Cribl will be listed under the **Cribl** tag, while the user-built ones will be listed under **Custom**. Over time, Cribl will ship more patterns, so this distinction allows for both sets to grow independently. In the case of an ID/Name conflict, the Custom pattern takes priority in listings and search.

Exporting and Importing Datatype rulesets

You can export and import Custom (or Cribl) rulesets as JSON files. This facilitates sharing across Cribl.Cloud organizations.

To export a Datatype ruleset:

1. Select to open an existing ruleset, or create a new ruleset.

2. In the resulting modal, select **Manage as JSON** to open the JSON editor.
3. You can now modify the ruleset directly in JSON, if you choose.
4. Select **Export**, select a destination path, and name the file.

To import any Dataset ruleset that has been exported as a valid JSON file:

1. Create a new ruleset.
2. In the resulting modal, select **Manage as JSON** to open the JSON editor.
3. Select **Import**, and choose the file you want.
4. Select **OK** to get back to the **New ruleset** modal.
5. Select **Save**.



Every Dataset ruleset must have a unique value in its top `id` key. Importing a JSON file with a duplicate `id` value will fail at the final **Save** step, with a message that the ruleset already exists. You can remedy this by giving the file a unique `id` value.

Datatype Rulesets Shipped with Cribl Search

You can use these Datatype rulesets out of the box:

- `Apache Datatypes`
- `AWS Datatypes`
- `Azure Datatypes`
- `Cisco Datatypes`
- `Cribl Search _raw Data`
- `Cribl Search`
- `CSV Datatypes`
- `Microsoft Graph API Datatypes`
- `Microsoft O365 Datatypes`
- `Microsoft Windows Datatypes`
- `OCSF Datatypes`
- `Palo Alto Datatypes`
- `Syslog Datatypes`
- `Zeek Datatypes`

Apache Datatypes

Parses Apache web server logs.

This Datatype ruleset consists of the following rules, evaluated top-down:

	Rule name	Value of the `datatype` field added to the event
1	Apache Common	apache_access
2	Apache Combined	apache_access_combined

See some examples:

Common Combined

```
--- input ---
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2

--- output ---
{
  "_raw": "127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] \"GET /apache_pb.gif HTT
  "_time": 971211336,
  "datatype": "apache_access",
  "clientip": "127.0.0.1",
  "ident": "-",
  "user": "frank",
  "timestamp": "10/Oct/2000:13:55:36 -0700",
  "request": "GET /apache_pb.gif HTTP/1.0",
  "status": "200",
  "bytes": "2326"
}
```

Copy

AWS Datatypes

Parses common AWS log formats, such as VPC Flow, CloudFront, WAF.

This Datatype ruleset consists of the following rules, evaluated top-down:

	Rule name	Value of the `datatype` field added to the event
1	AWS CloudTrail	aws_cloudtrail
2	AWS VPC Flow	aws_vpcflow
3	AWS ALB	aws_alb_accesslogs
4	AWS ELB	aws_elb_accesslogs
5	AWS Cloudfront Web	aws_cloudfront_accesslogs
6	AWS WAF	aws_waf

See some examples:

CloudTrail | VPC Flow | ALB | ELB | CloudFront | WAF

```
---input---
{
  "eventTime": "2014-03-06T21:22:54Z",
  "eventSource": "ec2.amazonaws.com",
  "eventName": "StartInstances",
  "userIdentity": {
    "accountId": "123456789012",
    "userName": "Hatter"
  }
}

---output---
{
  "_raw": "{\"eventTime\":\"2014-03-06T21:22:54Z\",\"eventSource\":\"ec2.amazonaws
  "_time": 1394140974,
  "datatype": "aws_cloudtrail",
  "eventTime": "2014-03-06T21:22:54Z",
  "eventSource": "ec2.amazonaws.com",
  "eventName": "StartInstances",
  "userIdentity": {
    "accountId": "123456789012",
    "userName": "Hatter"
  }
}
```

Copy

Azure Datatypes

Parses Azure-specific logs, such as Network Security Group flow logs.

This Datatype ruleset consists of a single rule:

Rule name	Value of the `datatype` field added to the event
`Network Security Group`	`azure_networksecuritygroup`

Here's an example of how this Datatype processes an NSG flow log event:

Input Output


```
{
  "records": [
    {
      "time": "2023-05-15T12:00:00.0000000Z",
      "systemId": "00000000-0000-0000-0000-000000000000",
      "category": "NetworkSecurityGroupFlowEvent",
      "resourceId": "/SUBSCRIPTIONS/12345678-1234-1234-1234-123456789012/RESOURCEG
      "operationName": "NetworkSecurityGroupFlowEvents",
      "properties": {
        "Version": 2,
        "flows": [
          {
            "rule": "UserRule_Allow-HTTPS-Inbound",
            "flows": [
              {
                "mac": "000D3A146A48",
                "flowTuples": ["10.0.0.4,10.0.0.5,4096,443,T,O,A,1623840360,162384
              }
            ]
          }
        ]
      }
    }
  ]
}
```

Copy

Cisco Datatypes

Parses Cisco device logs.

This Datatype ruleset consists of the following rules, evaluated top-down:

	Rule name	Value of the `datatype` field added to the event
1	Cisco ASA	cisco_asa
2	Cisco FWSM	cisco_fwsm
3	Cisco Estreamer	cisco_estreamer_data

See some examples:

Cisco ASA | Cisco FWSM | Cisco Estreamer

```
--- input ---
%ASA-1-104001: Switching to ACTIVE - Primary interface is up

--- output ---
{
  "_raw": "%ASA-1-104001: Switching to ACTIVE - Primary interface is up",
  "_time": 1685613600,
  "datatype": "cisco_asa",
  "device": "ASA-1",
  "event": "104001",
  "status": "Switching to ACTIVE",
  "interface": "Primary"
}
```

Copy

Cribl Search

Parses newline-delimited JSON (NDJSON) data.

This Datatype ruleset consists of a single rule:

Rule name	Value of the `datatype` field added to the event
`ndjson`	`cribl_json`

Here's an example:

Input Output

```
{"timestamp":"2024-06-11T10:23:45.432Z","level":"warn","message":"Failed login att
```

Copy

`Cribl Search \_raw Data`

Parses NDJSON-formatted logs where the actual event payload is in the `_raw` field.

This Datatype ruleset consists of a single rule:

Rule name	Value of the `datatype` field added to the event
`ndjson`	`cribl_json_raw`

Here's an example:

Input Output

```
{"_raw":"{\"timestamp\":\"2024-06-11T10:24:13.123Z\",\"level\":\"info\",\"action\"
```

Copy

`CSV Datatypes`

Parses CSV-formatted logs.

This Datatype ruleset consists of a single rule:

Rule name	Value of the `datatype` field added to the event
csv	csv

Input Output

```
"id","timestamp","level","message"
1,"2023-08-01T12:00:00Z","INFO","Service started"
2,"2023-08-01T12:05:00Z","WARN","Connection slow"
3,"2023-08-01T12:10:00Z","ERROR","Service failed"
```

Copy

Microsoft Graph API Datatypes

Parses Microsoft Graph API logs.

This Datatype ruleset consists of the following rules, evaluated top-down:

	Rule name	Value of the `datatype` field added to the event
1	Security Alerts	microsoft_graph_securityalert
2	Security Alerts - Legacy	microsoft_graph_securityalert_legacy
3	Service Health	microsoft_graph_servicehealth
4	Service Issues	microsoft_graph_serviceissues
5	Messages	microsoft_graph_messages
6	Users	microsoft_graph_users
7	Devices	microsoft_graph_devices
8	Groups	microsoft_graph_groups
9	Applications	microsoft_graph_applications

See some examples:

Security Alerts Users Applications

```
--- input ---
[
  {
    "id": "12345-67890",
    "title": "Suspicious Activity",
    "severity": "High",
    "status": "NewAlert",
    "lastModifiedDateTime": "2023-05-20T15:30:00Z",
    "description": "User reported suspicious login."
  }
]

--- output ---
{
  "_raw": "{\"id\":\"12345-67890\",\"title\":\"Suspicious Activity\",\"severity\":
  "_time": 1684596600,
  "datatype": "microsoft_graph_securityalert",
  "id": "12345-67890",
  "title": "Suspicious Activity",
  "severity": "High",
  "status": "NewAlert",
  "lastModifiedDateTime": "2023-05-20T15:30:00Z",
  "description": "User reported suspicious login."
}
```

Copy

Microsoft O365 Datatypes

Parses Microsoft 365 (formerly Office 365) logs.

This Datatype ruleset consists of the following rules, evaluated top-down:

	Rule name	Value of the `datatype` field added to the event
1	`status`	`microsoft_office365_status`
2	`messages`	`microsoft_office365_messages`
3	`mgmt-activity`	`microsoft_office365_mgmt-activity`

	Rule name	Value of the `datatype` field added to the event
4	msg-trace	microsoft_office365_msg-trace

See some examples:

Status | Messages | Management Activity | Message Trace

```
---input---
{
  "value": [
    {
      "Id": "ServiceStatus",
      "Workload": "Exchange",
      "StatusDisplayName": "Service degradation",
      "Status": "ServiceDegradation",
      "StatusTime": "2023-05-15T12:00:00Z",
      "WorkloadDisplayName": "Exchange Online",
      "IncidentIds": ["EX123456"]
    }
  ]
}


---output---
{
  "_raw": "{\"Id\":\"ServiceStatus\",\"Workload\":\"Exchange\",\"StatusDisplayName
  "_time": 1684152000,
  "datatype": "microsoft_office365_status",
  "Id": "ServiceStatus",
  "Workload": "Exchange",
  "StatusDisplayName": "Service degradation",
  "Status": "ServiceDegradation",
  "StatusTime": "2023-05-15T12:00:00Z",
  "WorkloadDisplayName": "Exchange Online",
  "IncidentIds": [
    "EX123456"
  ]
}
```

Copy

Microsoft Windows Datatypes

Parses Windows Event Log formats (classic and XML).

This Datatype ruleset consists of the following rules, evaluated top-down:

	Rule name	Value of the `datatype` field added to the event
1	`Windows Event Log Classic`	`windows_event_classic`
2	`Windows XML Events`	`windows_event_xml`

See some examples:

Windows Event Log Classic | Windows XML Event

```
---input---
06/15/2023 07:20:51 PM
LogName: Security
Source: Microsoft-Windows-Security-Auditing
EventID: 4624
Level: Information
User: N/A
Computer: WIN-7PC
Description:
An account was successfully logged on.

---output---
{
  "_raw": "06/15/2023 07:20:51 PM LogName: Security Source: Microsoft-Windows-Secu
  "_time": 1718511651,
  "datatype": "windows_event_classic",
  "log_name": "Security",
  "source": "Microsoft-Windows-Security-Auditing",
  "event_id": "4624",
  "level": "Information",
  "user": "N/A",
  "computer": "WIN-7PC",
  "description": "An account was successfully logged on."
}
```

Copy

OCSF Datatypes

Parses logs in the Open Cybersecurity Schema Framework (OCSF) format.

This Datatype ruleset consists of a single rule:

Rule name	Value of the `datatype` field added to the event
`ndjson`	Value of the source event `category_name` field. For example, for `"category_name": "Network Activity"`, Cribl Search sets the `datatype` field to `"Network Activity"`

Here's an example:

Input Output


```
{
  "category_uid": 1,
  "category_name": "Network Activity",
  "class_uid": 100,
  "class_name": "Network Flow",
  "activity_id": 1001,
  "activity_name": "Connection Allowed",
  "metadata": {
    "version": "1.1.0",
    "product": "Example Product",
    "vendor_name": "Example Vendor",
    "profile": "network_activity"
  },
  "time": 1718000000000,
  "src_endpoint": {
    "ip": "192.168.0.10",
    "port": 44321
  },
  "dst_endpoint": {
    "ip": "93.184.216.34",
    "port": 443
  },
  "protocol": "TCP",
  "duration": 5042,
  "bytes_in": 23456,
  "bytes_out": 11432,
  "outcome": "SUCCESS"
}
```


Copy

Palo Alto Datatypes

Parses Palo Alto firewall logs.

This Datatype ruleset consists of the following rules, evaluated top-down:

	Rule name	Value of the `datatype` field added to the event
1	Palo Alto Traffic	pan_traffic
2	Palo Alto Threat	pan_threat
3	Palo Alto System	pan_system
4	Palo Alto Config	pan_config

See some examples:

Traffic Threat System Config

```
--- input ---
{
  "receive_time": "2023-05-15T12:00:00.0000000Z",
  "serial_number": "1234567890",
  "type": "TRAFFIC",
  "threat_content_type": "None",
  "generated_time": "2023-05-15T12:00:00.0000000Z",
  "source_ip": "192.168.0.10",
  "destination_ip": "93.184.216.34"
}

---output ---
{
  "_raw": "{\"receive_time\":\"2023-05-15T12:00:00.0000000Z\",\"serial_number\":\"
  "_time": 1718000000,
  "datatype": "pan_traffic",
  "receive_time": "2023-05-15T12:00:00.0000000Z",
  "serial_number": "1234567890",
  "type": "TRAFFIC",
  "threat_content_type": "None",
  "generated_time": "2023-05-15T12:00:00.0000000Z",
  "source_ip": "192.168.0.10",
  "destination_ip": "93.184.216.34"
}
```

Copy

Syslog Datatypes

Parses syslog messages.

This Datatype ruleset consists of the following rules, evaluated top-down:

	Rule name	Value of the `datatype` field added to the event
1	RFC3164	syslog
2	RFC5424	syslog

See some examples:

RFC3164 RFC5424

```
---input---
<34>Oct 11 22:14:15 mymachine su: 'su root' failed for user on /dev/pts/3

---output---
{
  "_raw": "<34>Oct 11 22:14:15 mymachine su: 'su root' failed for user on /dev/pts
  "_time": 436917255,
  "datatype": "syslog",
  "priority": 34,
  "timestamp": "Oct 11 22:14:15",
  "host": "mymachine",
  "app_name": "su",
  "message": "'su root' failed for user on /dev/pts/3"
}
```

Copy

Zeek Datatypes

Parses Zeek logs.

This Datatype ruleset consists of a single rule:

Rule name	Value of the `datatype` field added to the event
Zeek	`zeek`

Here's an example:

Input Output

```
192.168.1.100   443     192.168.1.50    53212   tcp     ssl     1532476123.119112
```

Copy

2.17.2. Storage Classes

Control your costs by selectively enabling which object storage classes (access tiers) you want each Dataset to search.

For each Dataset that targets Amazon S3, Azure Blob Storage, or Google Cloud Storage, you can specify which storage classes the Dataset should search. (Azure refers to storage classes as access tiers.) This configuration option offers two advantages:

- You can control costs by skipping over "colder" classes that are expensive to read.
- You can avoid errors, because Cribl Search automatically skips over the classes that it cannot read on demand.

How Storage Classes (Access Tiers) Work

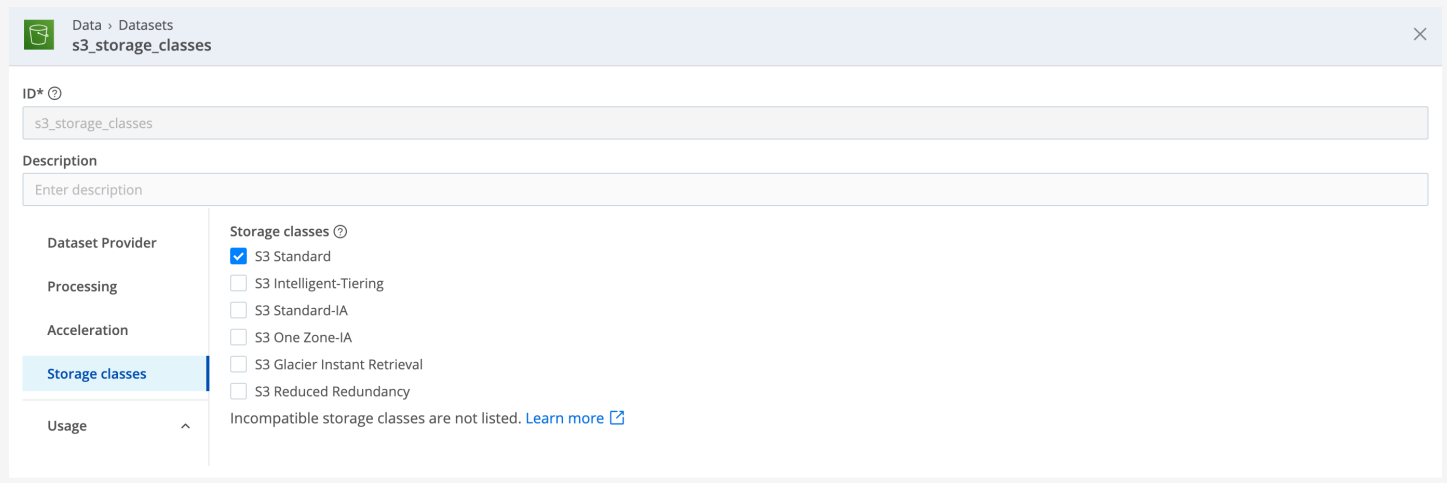
When you create a new Dataset on an object store Dataset Provider, the configuration modal's **Storage classes** tab will default to a single class that offers rapid retrieval at low cost. As shown below, you can enable other storage classes that you want the Dataset to search.

By design, this tab omits classes that Cribl Search cannot search on the corresponding Dataset Provider. Objects in these classes can't be read on demand, because they must first be rehydrated to a warmer class. (See also Handling Automatically Aged Objects below.)

At least one storage class must be selected, before you can save or resave the Dataset's configuration.

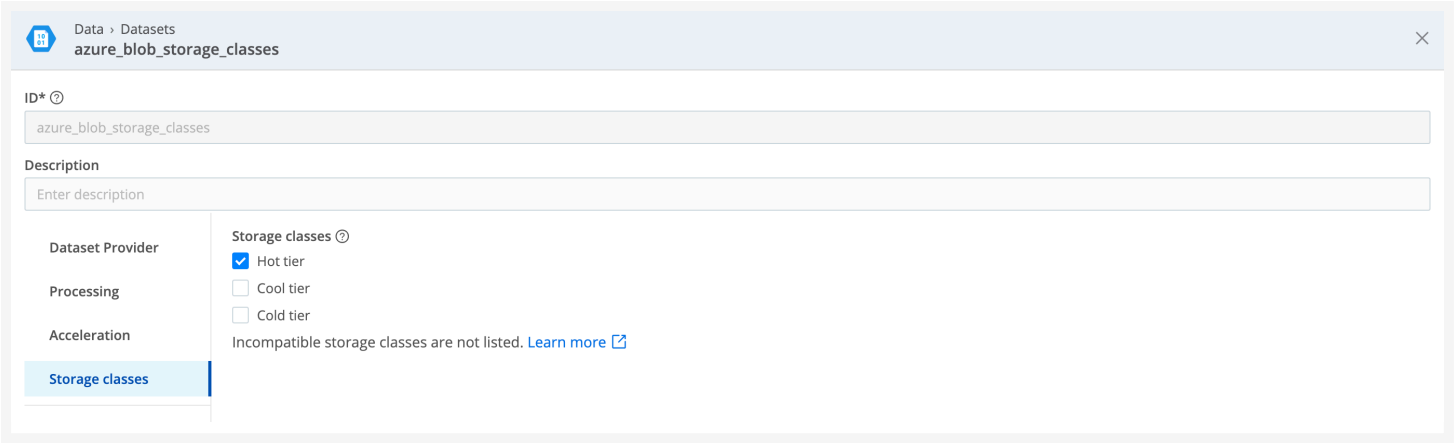
Default Storage Classes

On Amazon S3 Datasets, the default storage class is **S3 Standard**. The tab omits unsupported tiers.



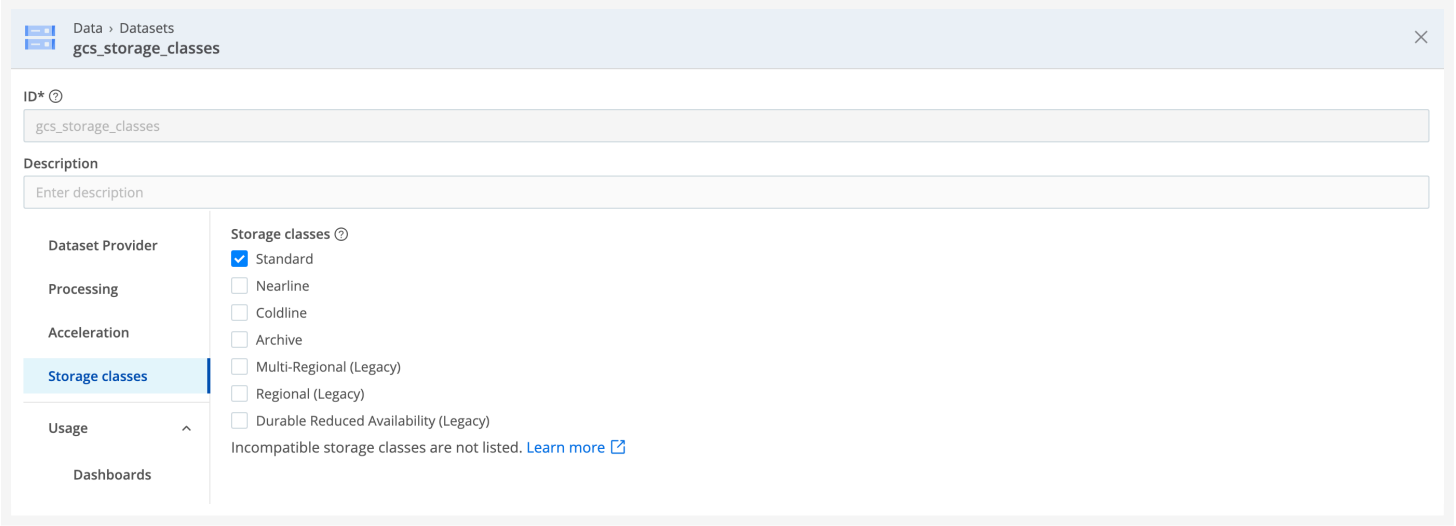
S3 Dataset: default storage classes

On Azure Blob Storage Datasets, the default storage class is **Hot tier**. The tab omits the unsupported archive tier.



Azure Blob Storage Dataset: default storage classes

On Google Cloud Storage Datasets, the default storage class is **Standard**.



Google Cloud Storage Dataset: default storage classes

Storage Classes on Existing Datasets

On Datasets that were configured before Cribl Search added storage class selection, **all** supported storage classes are initially enabled. This is to avoid disruption to searches that you've scheduled, or that you explicitly rerun.

However, you can optimize your future costs by using the above options to deselect classes that have higher retrieval costs. As we'll see next, these classes typically store older data that you'll consider less relevant to your current searches.

Handling Automatically Aged Objects

On Amazon S3 Datasets, S3 Intelligent-Tiering is a storage **class** that you can enable, but objects will gradually age into archive **tiers** that effectively behave like Cribl Search's unsupported Glacier tiers. On Datasets where you've enabled **S3 Intelligent-Tiering**, Cribl Search will try to read these objects, will gracefully skip them when reading fails, and will log a warning.

On Azure Blob Storage, enabling automatic lifecycle management, with a permissive policy, can similarly cause your blobs to age onto the archive tier. Blobs on this tier cannot be retrieved unless you rehydrate them into a warmer tier. Here, Cribl Search will similarly skip reading these blobs, and will log warnings. You can avoid the issue (and retain the option to make your data searchable) by configuring a lifecycle management policy that excludes the archive tier.

On Google Cloud Storage, enabling Autoclass behavior can similarly cause your blobs to age onto cooler tiers that you haven't enabled your Dataset to search. Google Cloud Storage throws no warnings or errors, and will retrieve these objects at fast STANDARD speed (but with added retrieval costs). Therefore, be aware that enabling Autoclass can increase your costs.

2.17.3. Supported Data Formats

Learn about how Cribl Search processes the most common data formats.

List of Supported Data Formats

You can search data in most of the commonly used formats, including text files, binary files, archives, compressed files, and more.

Data Format Example	File Extension Examples
Gzip	`.gz .gzip .tgz .tar.gz application/x-gzip`
Journal	`.journal .journal~`
LZ4	`.lz4`
Snappy	`.snappy`
Parquet	`.parquet .pqt .parq`
Splunk rawdata	
Tar	`.tar .tgz .tar.gz`
Text	`.log .csv .json .ndjson .txt`
ZIP	`.zip`
Zstd	`.zst`

Text Data Formats

Text files contain human-readable data. They're often used to store logs, but can also contain structured data, in formats such as JSON or CSV.

You can search text files in most of the commonly used formats, including `.log`, `.json`, `.ndjson`, `csv`, and more.

Binary Data Formats

Binary files are non-human-readable files that use formats optimized for storage and processing efficiency. They're often used to store structured data, such as Parquet files, or system and application events, such as journals.

When you search files in formats that Cribl Search can't recognize, they're processed as binary files.

Journals

Journals are binary files used for logging and storing system and application events.

You can search both `.journal` and temporary `.journal~` files, which are helpful when investigating unexpected process failures.

Parquet

Parquet binary files are columnar storage format files that support nested data structures. This makes them an ideal choice for big data analytics.

You can search Parquet files with the `.parquet`, `.pqt`, and `.parq` extensions.

Using Parquet Time Fields

When processing Parquet files, by default, the Cribl Search `_time` field is assigned the value of now. You can instead use a time field of the event, by assigning its value to `_time`:

1. Create or open a Datatype by navigating to **Data** > **Datatypes**.
2. Select **Add rule**.
3. At the bottom, in the **Add fields to events** section, select **Add field**.
4. Assign the **Name** as `_time` and its **Value expression** to the time field in the event. For example, `_time = time`.
5. Save the Datatype changes.
6. Navigate to the Dataset that specifies your Parquet data and open the **Processing** section.
7. Assign the Datatype to the Dataset and save your changes.

Now, when searching your Parquet Dataset, the time from the event will be used as the reference for the search's time range.



Example: Amazon Security Lake

Amazon Security Lake logs data into Parquet format following the schema defined by OCSF. The time is assigned to the field `time` and is an epoch time expressed in milliseconds.

> To use this field, you need to set its value to the Cribl Search field `_time`. Since Cribl Search uses seconds instead of milliseconds, you also need to convert it by dividing by 1,000. In this case, the field you'd add to a Datatype would be: `_time = time/1000`.

Compression Data Formats

Compression formats are used to reduce the size of data to optimize storage space and speed up data transmission.

You can search files compressed with a wide range of algorithms, including LZ4, zstd, gzip, Snappy, and more.

LZ4

LZ4 is a popular compression format that offers multiple compression levels and a good balance between compression ratio and speed.

Zstd

Zstandard, also known as zstd, is an open-source compression format developed at Facebook. It's often used in scenarios where high compression speeds are critical.

Splunk Rawdata

Splunk's compressed rawdata files store event data ingested by indexers. They also contain related journal information.

Gzip

A popular compression format, gzip is often used to compress log files.

In Cribl Search, you can search both `.gz` and `.gzip` files.

Snappy

Snappy, previously known as Zippy, is an open-source compression format developed at Google, popular for its high compression speeds.

In Cribl Search, you can search `.snappy` files.

Archive Data Formats

Archive formats are used to package multiple files together into a single file.

You can search archives in most of the commonly used formats, including tar, ZIP, and more.

Tar

The tar (tape archive) file format is used by the UNIX-based utility tar to package files together into sets known as “tarballs”.

ZIP

ZIP is a popular archive format that supports lossless data compression. `.zip` files can contain compressed or non-compressed data, and can use different compression levels.

Searching Tarballs and ZIP Archives

When searching tarballs or ZIP archives, you can specify which of the archive’s files you want to search.

When processing files with `.tar`, `.tgz`, `.tar.gz`, or `.zip` extensions, Cribl Search processes all files inside the archive and adds a field named `cribl_archive_source` to each file’s events. The value of `cribl_archive_source` is the name of the file inside the archive.

For example, let’s say an archive named `archive.tgz` contains files `dir1/file1.log` and `dir2/file2.log`. An event from `dir1/file1.log` returned from a search would look like this:

```
{
...
source: archive.tgz
cribl_archive_source: dir1/file1.log
...
}
```

Copy

Now, to search a specific file inside the `archive.tgz` archive, you can refer to the `cribl_archive_source` field in your query. For example: `cribl_archive_source=dir1/file1.log`.

Internal Fields

Cribl Search maintains an internal `__raw` field that contains each original event’s state after Event Breakers are applied, but before other Datatype rules. (Where an event contains no `_raw` field, Cribl Search constructs one by serializing all the event’s top-level and nested fields.)

You can access this field to determine the size of the original event, including its metadata.

3. Search Your Data

Learn how to build Cribl Search queries, with examples.

Build a Search

Learn about query syntax, operators, time ranges, sampling, and more. See: Build a Search.

Write Your First Query

Write simple queries in Kusto Query Language, using operators such as `limit`, `project`, `where`, `count`, `order`, and others. See: Write Your First Query.

Write Queries Using Copilot

Use the Cribl Copilot KQL assistant to generate Kusto Query Language expressions from natural language. See: Write Queries Using Copilot.

Common Query Examples

Examine, run, and adapt some typical search queries. See: Common Query Examples.

Use Search Results

Export, reuse, and manage the results from search queries you've already run. See: Use Search Results.

Notebooks

Run an entire investigation in one tab, and share your work with others. See: Notebooks.

Example Searches by Provider

Learn how to search data coming from different Dataset Providers. See: Example Searches by Provider.

Search and Retrieve Results with the Cribl API

Use the Cribl API to execute searches and retrieve Cribl Search results. See: Search and Retrieve Results.

3.1. Build A Search

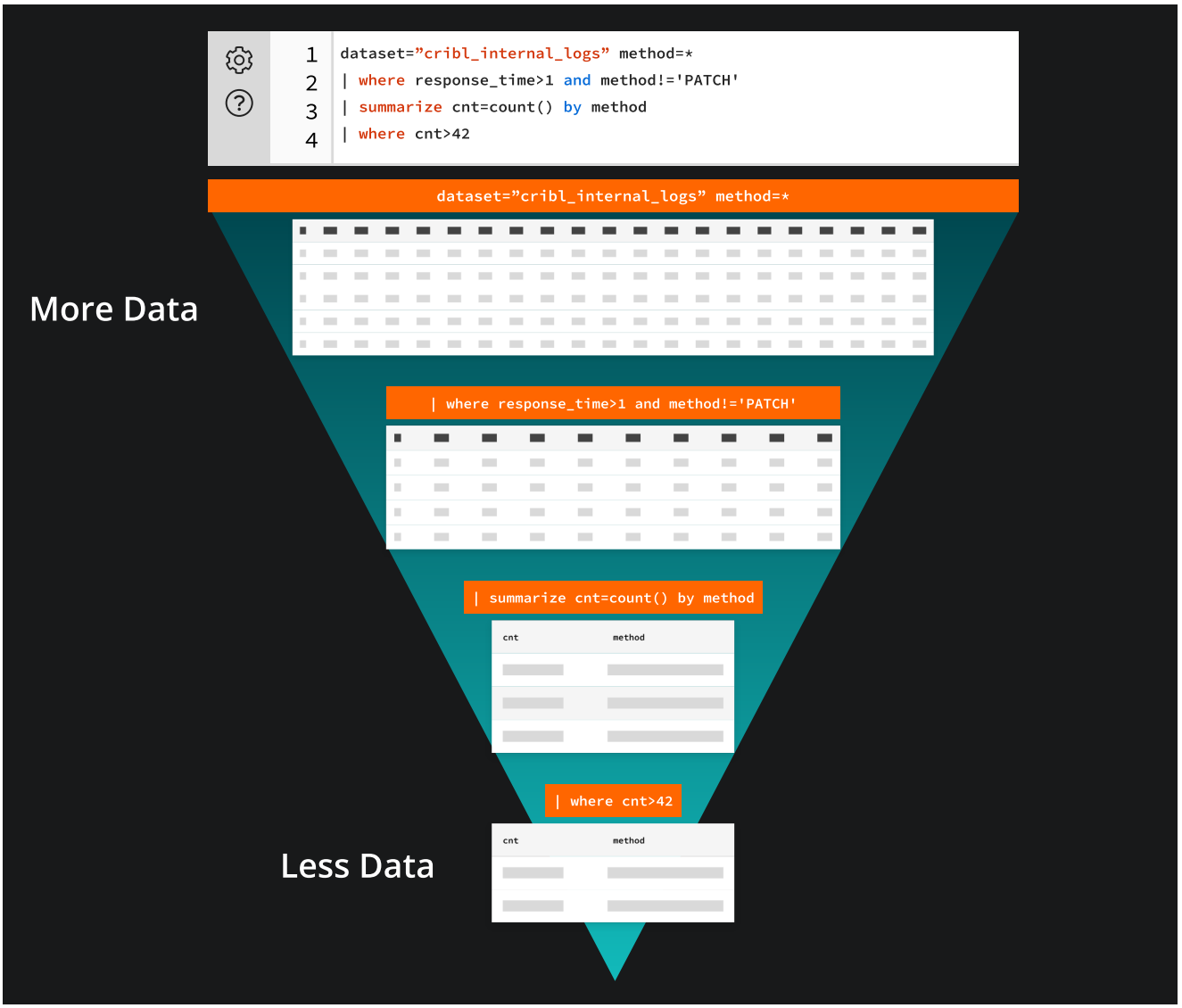
Learn about query syntax, operators, time ranges, sampling, and more.

A search needs a query and a time range. Queries specify the data to search and run functions and operators from our ergonomic and powerful search language.

Queries contain zero or more operators separated by a pipe `|`. Data flows, or is piped, from one operator to the next. Events are filtered or manipulated at each operation and then fed into the following operation.

You can think of the data flow as a funnel, where you start out with selected events. Each time the data passes through another operator, it is filtered, rearranged, or summarized. Because the piping of information from one operator to another is sequential, the query operator order is important and can affect both results and performance. At the end of the funnel, you're left with a refined output.

The following diagram shows how each operation in a query changes search results.



Data flow diagram

Query Syntax

`[Statements;] Scope [| Processing [| ... ]]`

- **Statements** - Optional statements that can set query options and assign names to expressions (which allows you to use `join` or `union` later in the query). Each statement needs to end with a semicolon (`;`).
- **Scope** - Define Datasets and use the `cribl` operator to specify the events to search. Wildcards `*` represent one or more characters. For example, `source=/*/foo.log "error"` returns lines from `/*/foo.log` containing the term `error`.
  - Dataset syntax: `dataset=DatasetID`. E.g, `dataset=myCDNLogs`.
    - Data from different Datasets can be easily combined in a search. For example, `dataset=myVPCFlowlogs or dataset=myCDNLogs`.
  - The `default` Dataset is searched when no Dataset is defined.
  - The `cribl` operator is implicit when an operator isn't specified. You can stop this behavior by enabling **Strict Kusto Mode** in the UI Options.
  - The `cribl`, `externaldata`, and `find` operators are used in the scope.
  - The `search` and `timestats` operators may be used in the scope.
- **Processing** - Optional operators that can be used together with context, cribl, statistical, and scalar functions. Use a pipe `|` between operators.

Every stage of a query can also include Macros (for example, `${MacroID}`).

Examples

- Find events with `error` in the `cribl_internal_logs` Dataset:

```
dataset="cribl_internal_logs" "error"
```

Copy | Run in Cribl Search

- Find and return up to 1,000 logs:

```
dataset="cribl_search_sample"
| limit 1000
```

Copy | Run in Cribl Search

- Aggregate events by the `method` field:

```
dataset="cribl_internal_logs"
| limit 1000
| summarize count() by method
```

Copy | Run in Cribl Search

- Count the events in each minute, grouping by the `method` field, from the last 15 minutes:

```
dataset="cribl_internal_logs" earliest=-15m method=*
| limit 1000
| timestats span=1m count() by method
```

Copy | Run in Cribl Search

- Merge the events of two different Datasets:

```
let query1 = dataset="dataset1" | where x < 10;

dataset="dataset2"
| join query1 on commonField
```

Copy

Syntax Highlighting

Within the query box, the following color coding distinguishes components of your query:

Component	Light Mode	Dark Mode
Literal	Magenta	Magenta
Field name	Pink	Pink
Search term	Orange	Orange
Operator	Blue	Blue
Function	Teal	Teal
Connecting keyword	Amber	Amber

Component	Light Mode	Dark Mode
Error	Red	Red
Comment	Green	Green

(Dark mode uses lighter shades to reduce contrast.)

Field Names

Each event field has a unique name that is used as its reference in a query. Each field contains a key-value pair, where the field name is the key. Field names follow these rules:

- They're case-sensitive.
- Support letters, digits, underscores `_`, spaces, dots `.`, and dashes `-`.
- Fields containing a special character (spaces, dots, or dashes) require quoting (see below).

Quoting Keywords

Field names that are identical to some query language keywords, or have special characters, require quoting when they're referenced directly by a query:

Query text	Description
`field`	Field names that don't include special characters or map to some language keyword don't require quoting.
`['field-name']`	Field names that include special characters must be quoted using `['` and `']` or using `["` and `"]`.
`["where"]`	Field names that are language keywords must be quoted using `['` and `']` or using `["` and `"]`.



You can normalize field names that contain separators, using the `foldkeys` operator.

For example, events from the `cribl_internal_logs` Dataset have a `time` field which is a reserved keyword requiring quoting. To reference the field in a query you'd specify it as:

```
['time']="2023-03-31T12:24:10.770Z"
```

Copy

Similarly, to exclude `null` fields from the search results, you must bracket and quote the `null` keyword. To see how this works, let's start with the following query. Here, the `method` field has a value of null. We'll want to remove that field:

```
// Count of method over time with a null field value.
dataset="cribl_internal_logs" earliest=-15m
 | limit 1000
 | timestats count() by method
```

Copy | Run in Cribl Search

This first attempt will **not** work, because `null` is unquoted:

```
// This won't work because null is a keyword and must be quoted within brackets.
dataset="cribl_internal_logs" earliest=-15m
 | limit 1000
 | timestats count() by method
 | project-away null
```

Copy | Run in Cribl Search

Here is the same query, with `null` properly bracketed and quoted. If you run it, watch the `null` field from the original query go away:

```
// This will work because null is properly quoted.
dataset="cribl_internal_logs" earliest=-15m
 | limit 1000
 | timestats count() by method
 | project-away ["null"]
```

Copy | Run in Cribl Search

Generated Fields

Cribl Search generates fields from expressions and returns them in your search results. The expression is evaluated and all non-word characters are replaced with underscores `_`. For example, `count(myField)` returns as `count_myField`.

If there's only one field, a single underscore is appended to it. For example, `count` returns as `count_`.

Maximum Field Length

Generated fields can be a maximum of 20 characters long. Fields are truncated from the left of the expression after the function name to keep within the 20-character limit. For example, `countif(channel, elapsedMS > 0)` returns as `countif_elapsedms_0`.

If the left field does not start with an alphabetic character, the farthest right field is truncated.

Built-in Fields

Cribl Search uses several fields to assist in the handling of data. These "meta" fields are not part of an event, but they are accessible, and can be referenced in queries.

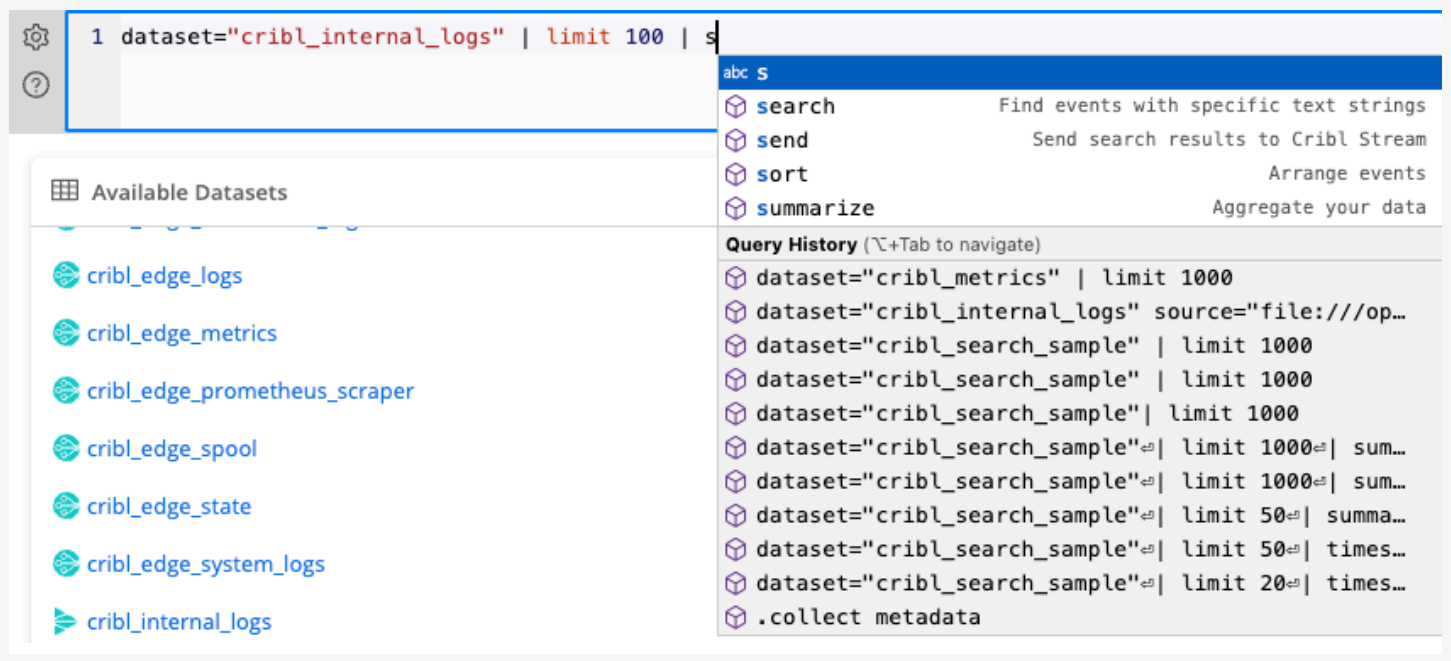
Field	Description
`_raw`	The raw event. Where an event contains no `_raw` field, Cribl Search constructs one by serializing all the event's top-level and nested fields.
`_time`	The raw event's timestamp. If Cribl Search detects no timestamp, it uses the last modified timestamp.
`cribl_fleet`	The name of the Edge Fleet that contained the event.
`dataset`	The Dataset that was queried to retrieve the event.
`datatype`	The Datatype processed against the event.
`host`	The hostname of the device that contained the event.
`source`	The name of the data object that contained the event.

If an event has a field with the same name as a Cribl Search built-in field (except `_time`), the event's field is prepended with `data_`. For example, `data_source`.

Autocomplete

As you type in the query box, a drop-down modal with suggestions is displayed to make query writing easier. Use the down and up keyboard keys to highlight a suggestion. Press Enter/Return or Tab to add the highlighted suggestion to your query, or select a suggestion.

Autocomplete adapts its suggestions as you type, referencing built-in fields, Dataset IDs, and the query language.



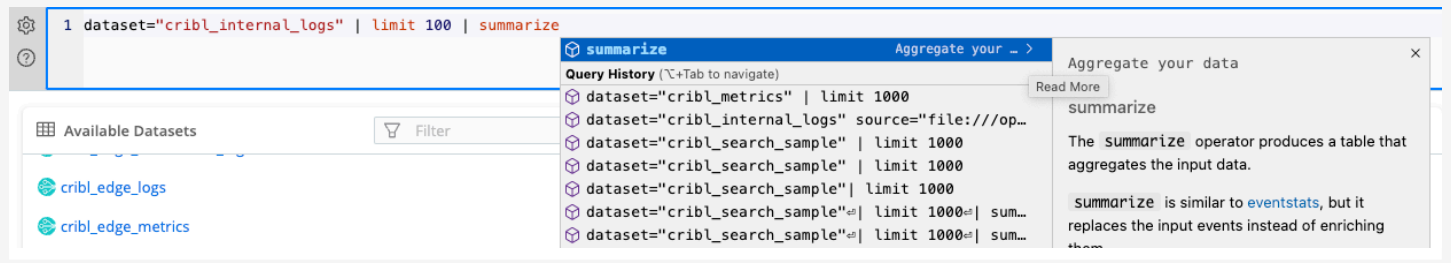
Autocomplete

The bottom section of the autocomplete modal provides a list of recent queries. When clicked, the query is added to the query box.

You can reuse query text across different searches, by using Macros.

Look Up the Query Language Docs

Operators and functions have a greater-than character > displayed on the right of the autocomplete suggestion when highlighted. When clicked, autocomplete expands to display its documentation for reference.



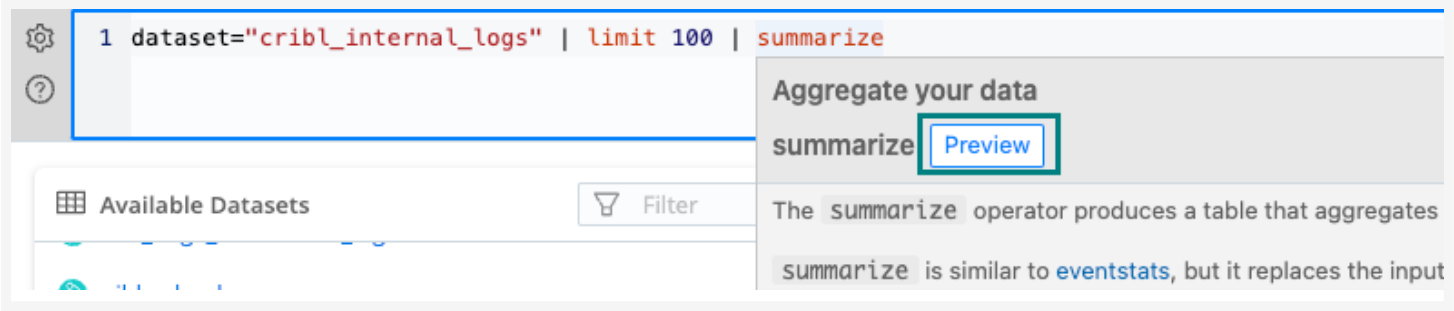
Viewing the docs inside Cribl Search

Preview

Preview provides a testing mode where operators can be applied to a previously run query, allowing you to quickly preview the changes without executing the search on the actual data source. You can preview operators as many times as you need, without incurring the time or cost of executing them on actual data providers until you're ready.

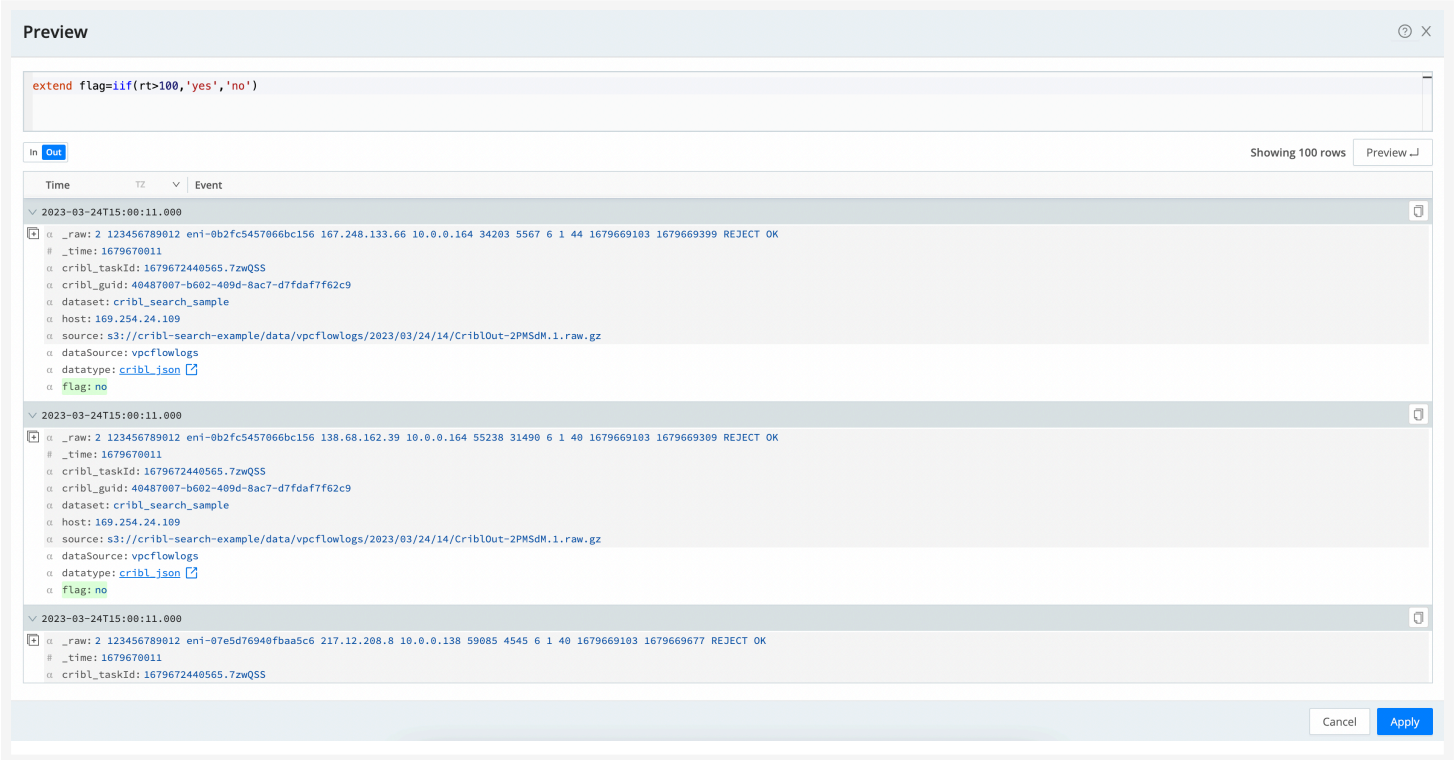
- Up to 100 results from the initial search are used in Preview.
- The `centralize` and `send` operators are not supported.

To open Preview, hover over an operator and select **Preview**. For non-aggregate queries, you can first select bars in the histogram to view results for only the selected times. Use shift+select to select multiple bars.



Preview button

Type the operators you'd like to test against the initial search results and press Enter/Return or select **Preview**.



Preview example

Notice the results table switches to **Out** instead of **In**, indicating the displayed results are what the operator would return.

- New data is highlighted in green.
- Modified data is highlighted in orange.
- Deleted data is highlighted in red.

Select **Apply** to have your test query added to your original query.

Date and Time Range

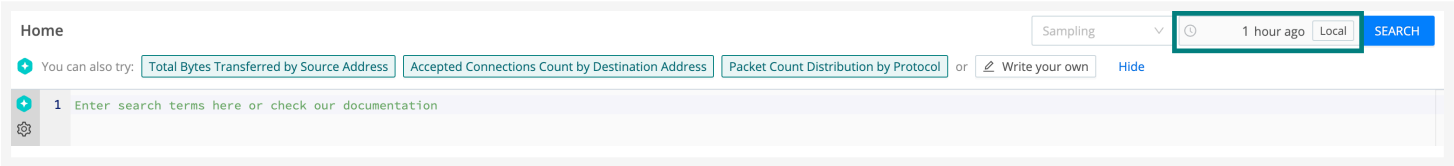
You can use the **Date Range** modal to find events within a specified date or time range. This narrows down event results, adding another layer of granularity to your search.



Some operators, like `cribl` and `timestats`, support their own time arguments that can override the time range set in the UI.

Subqueries can also use their own time arguments, but the main query time range still applies to the final results.

To begin, select the time field at the top right of **Search Home**.



The date/time field

Copying Date Ranges

To copy and paste predefined date ranges into other Search browser windows, hover over the time field and select the **Copy** button. Navigate to another browser containing Search, hover over the time field, and select the **Paste** button.

Good deal! You just saved yourself a few minutes of keyboard fatigue.

Time Range Options

Once you click into the time field, the date and time modal will appear. Here, you can select or enter a relative time, enter a specific date/time range, or define a duration of time around a specific date.

Relative Time

To find events that occurred between the selected time and now, use Relative time.

Select one of the following options:

- **seconds ago**: Number of seconds in the past.
- **minutes ago**: Number of minutes in the past.
- **hours ago**: Number of hours in the past.

- **days ago**: Number of days in the past.
- **All time**: All events.

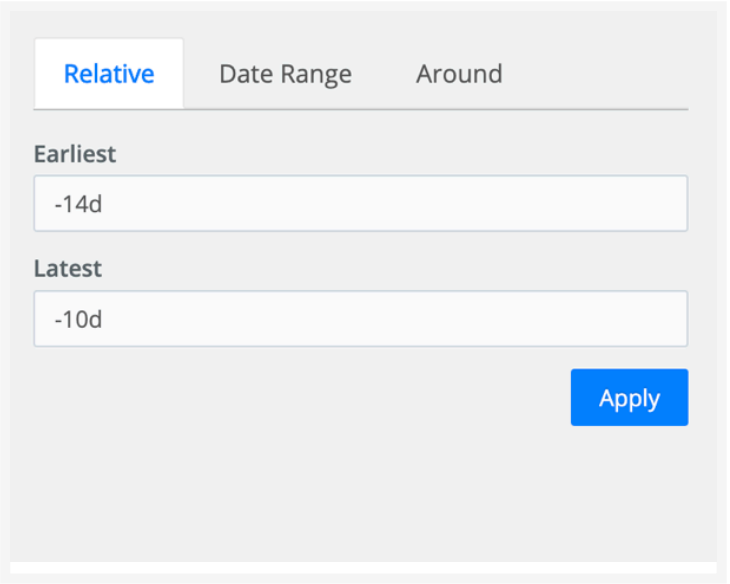
For scheduled searches, `now` is the **scheduled** time, not the actual execution time.

For example, if you schedule a search for 1:00, but an unplanned outage causes the search to run at 1:05, the value of `now` will be 1:00, not 1:05.

Relative Tab

The **Relative** tab is another way to find events within a specified time range. To find events that occurred between two times, select the **Relative** tab and enter an **Earliest** (-90 days maximum) and a **Latest** time.

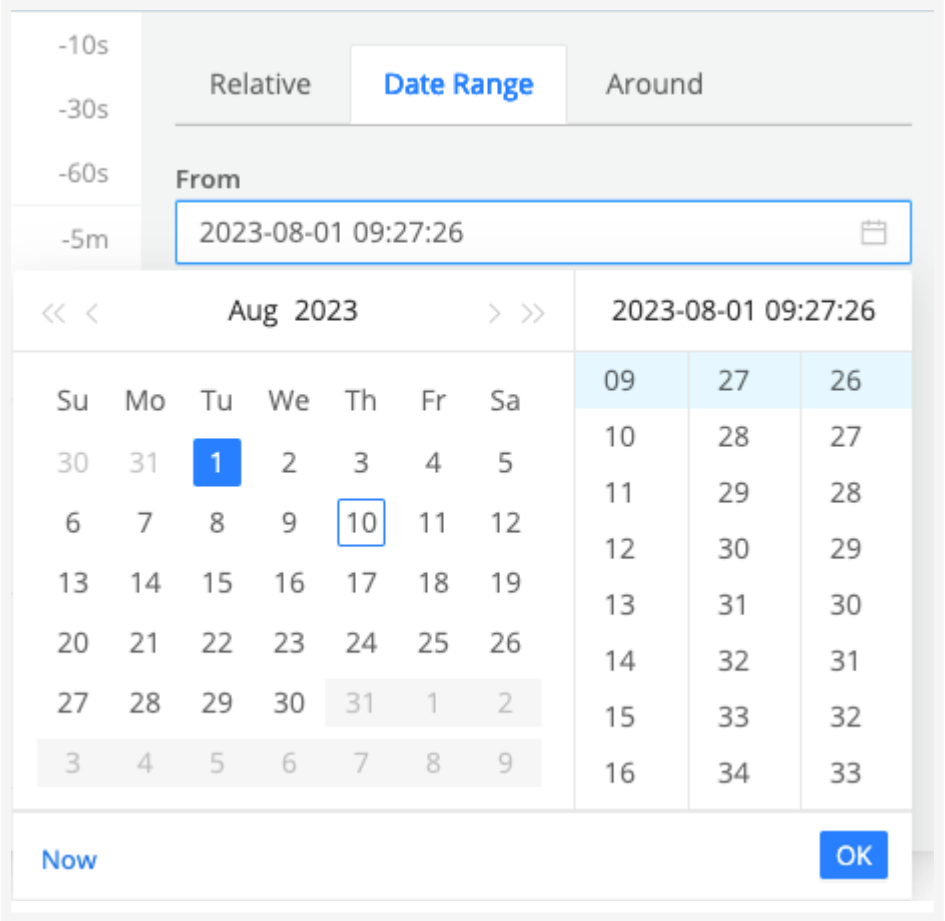
In this example, the defined Relative time range will return events that occurred between 14 days ago and 10 days ago.



Earliest and Latest relative time range

Date Range

To define a custom date and time range to search within, use the **Date Range** tab. Select the desired start and end dates and times from the calendar. Select **Apply** to show search results for the date range.



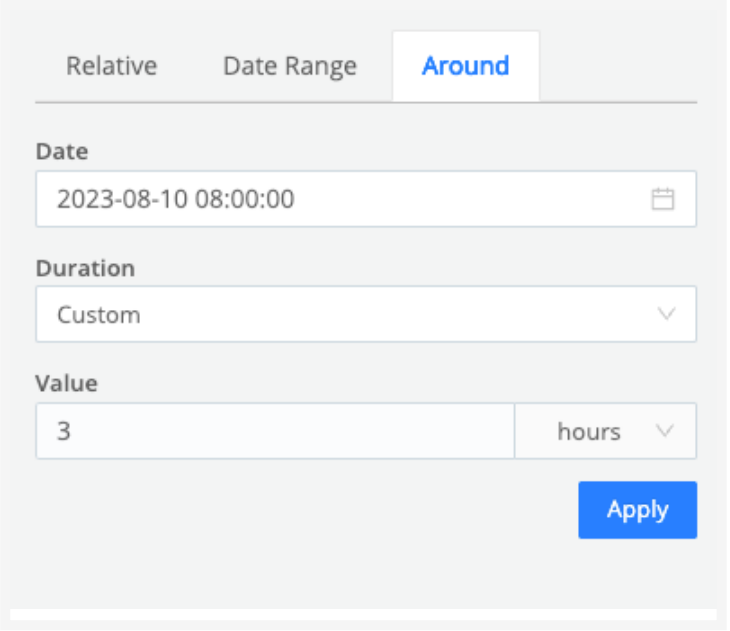
Custom date and time range options in the Date Range tab

Around

To select a custom date and specify a duration of time “around” that date, use the **Around** tab.

For example, you can select 07-03-2023 at 10:00 +/- 2 hours, the results of which will show you all events that happened from 8:00 am to 12:00 pm.

This is useful for finding important events that you know happened around a particular date, but you don’t know exactly when.



The Around tab

Time Zone

To apply a specific time zone to your search, select the time zone under **Current Timezone**. You can select a time zone, or search by time zone, country, city, or offset (+8 for example).

Notes

- Any selection made in the time zone selector is automatically applied to the **Time** column on the **Events** tab.
- The **Local** time zone is based on the time zone set of the computer that is running Search.

Sampling

Sampling uses a ratio to reduce the number of results returned from a search. For example, if a search returns 1,000 events, a 1:10 sampling ratio returns about 100 events.

Select **Sampling** to the left of the time range box to set a sampling ratio.

Comments

You can add comments to your query with double forward slashes `//` prepending the comment. Any characters on the same line after `//` are part of a comment. For example:

```
goats | summarize count() // count the goats
```

Copy

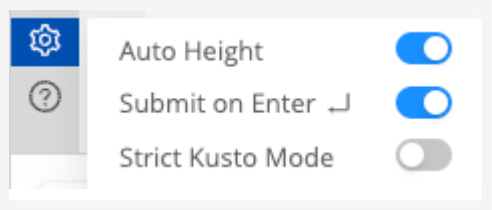
Autoformat Query Text

To make a complex query more organized and easier to read, you can autoformat its text.

When in the query box, press the `Ctrl + |` (Linux, Windows) or `Command + |` (MacOS) key combination.

UI Options

Select the gear button to the left of the query box to open the options menu.



Search options

- **Auto Height**: Enabled by default. Automatically adjusts the height of the query box based on the size of the query. When disabled, hover over the bottom line of the query box, then select and drag the line down or up to adjust its size.
- **Submit on ENTER**: Enabled by default. Runs the search when you press the Enter/Return key, so you don't have to select **Search**.
- **Strict Kusto Mode**: Disabled by default. Stops the `cribl` operator from being implicit in the **scope**. When enabled, a term is interpreted as a Dataset ID.

3.2. Write Your First Query

Write simple queries in Kusto Query Language, using operators such as `limit`, `project`, `where`, `count`, `order`, and others.

Cribl Search is based on Kusto Query Language (KQL), which lets you delve into your data to discover patterns, identify anomalies and outliers, and create statistical models.



To generate queries from natural language, try the Cribl Copilot KQL assistant.

Example Scenario

Imagine you're a security operations analyst working for a company that relies heavily on cloud infrastructure. You want to analyze a Dataset containing virtual private cloud (VPC) flow logs for insight into network traffic patterns, potential security threats, and resource utilization in different regions.

What You'll Learn

You'll write simple queries in KQL to analyze a sample Dataset of VPC flow logs. You'll learn how to:

- Understand the structure of a Cribl Search query.
- Return a specific number of rows by using the `limit` operator.
- Extract information from the `_raw` field using the `extract` operator and Parser.
- Filter data by using the `where` operator.
- Calculate available events by using the `count` operator.

Your Learning Goals

By the end of this tutorial, you'll be able to write a query with the most commonly used operators in KQL to analyze VPC flow logs, understand network traffic patterns, and identify potential security threats within your cloud infrastructure.

The Structure of a Cribl Search Query

Organizations across various sectors process vast amounts of data and aim to extract valuable insight from it. In the VPC flow logs example, you have a Dataset with network traffic data from your organization's cloud infrastructure. This tutorial will guide you through the fundamental structure of the Kusto Query Language to analyze and interpret the Dataset.

What's a Cribl Search Query?

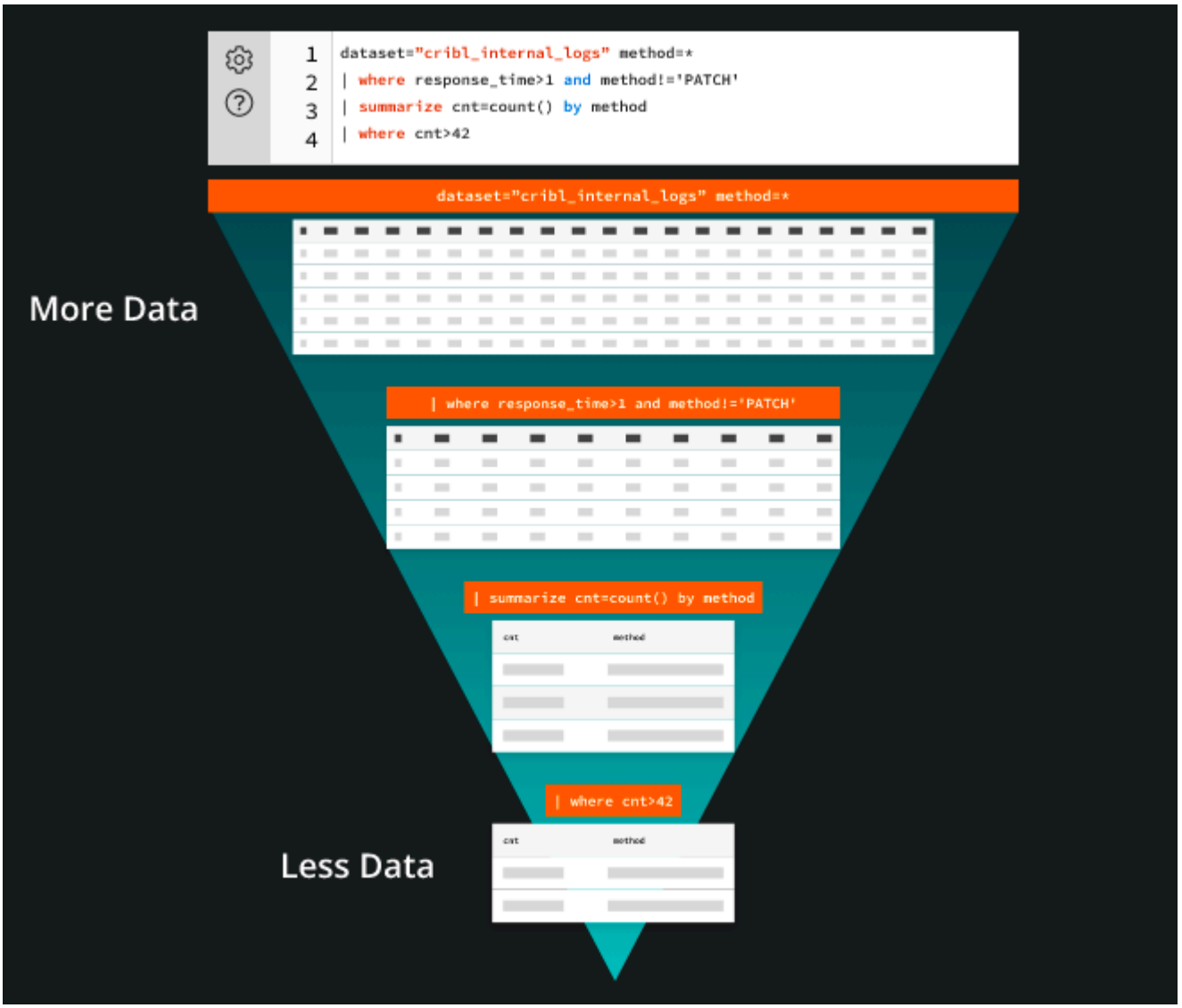
A Cribl Search query is a read-only request expressed in plain text that processes data and returns results in a tabular or graphical format. It consists of one or more query statements. The predominant type of query statement is a tabular expression statement.

What's a Tabular Statement?

In a tabular expression statement, both input and output consist of tables or tabular Datasets.

Tabular statements hold zero or more operators, each beginning with tabular input and yielding tabular output. Operators are connected by a pipe (|), allowing data to flow from one operator to the next for filtering or manipulation. This sequential piping of information makes the query's operator order crucial.

Think of it like a funnel, where you start with a full data table. The data is filtered, rearranged, or summarized as it passes through each operator. At the end of the funnel, you obtain a refined output.



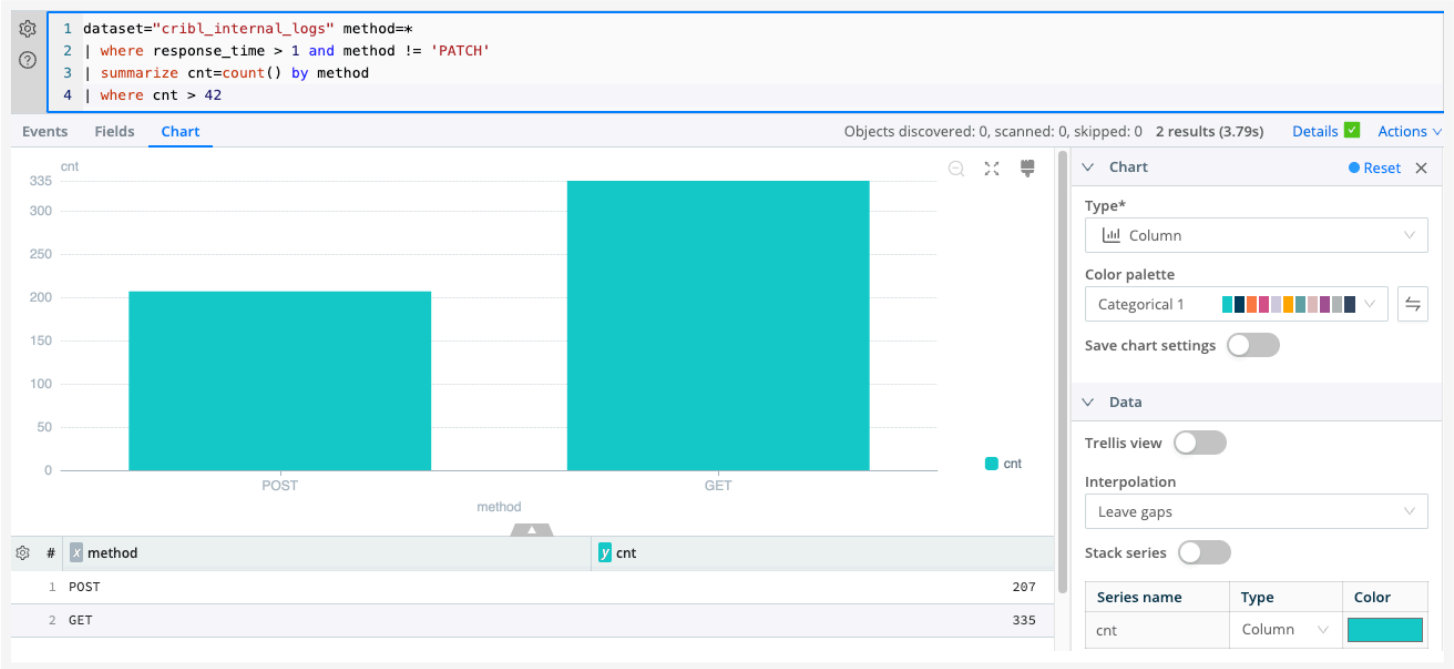
Data filtered through a funnel

Here's an example query:

```
dataset="cribl_internal_logs" method=*
| where response_time > 1 and method != 'PATCH'
| summarize cnt=count() by method
| where cnt > 42
```

Copy | Run in Cribl Search

This query contains a single tabular expression statement that references a Dataset called "cribl_internal_logs" and filters the data based on the method column. The data rows are filtered by the response_time column's value and exclude rows with the method "PATCH". The query then summarizes the count of rows for each method and filters the results to only display methods with a count greater than 42.



Results of a query with a tabular expression statement

Return a Specific Number of Rows by Using the `limit` Operator

You can use a Cribl Search query to explore Datasets and gain insights. In this scenario, you have an unfamiliar Dataset of VPC flow logs called "cribl_search_sample" that ships out of the box with Cribl Search. You want to explore what you can learn from this data.

In this task, you'll examine the structure of the data to understand the kinds of questions you can ask about these network flows.

Write Your First Query

This query returns a small subset of the data to help you familiarize yourself with the columns and types of data in the table.

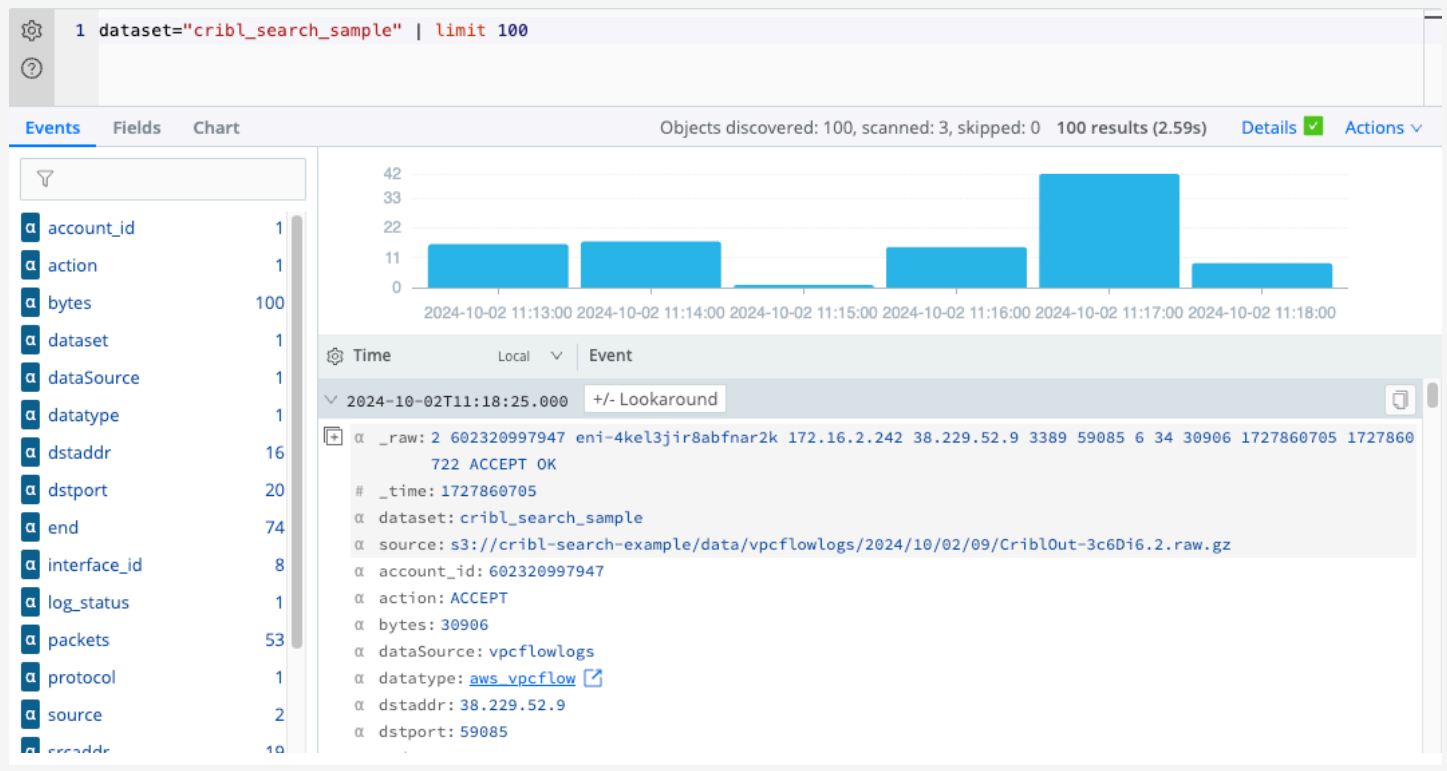
The `limit` operator is perfect for this task, as it returns a specific number of arbitrary rows.

1. Copy this query to your clipboard:

   ```
   dataset="cribl_search_sample" | limit 100
   ```

   Copy | Run in Cribl Search

2. Paste the query into the search box.
   Tip: You can autoformat the query text.
3. Notice that the query begins with a reference to the data table, `cribl_search_sample`. This data is piped into the first and only operator, which then selects 100 arbitrary rows.
4. Run the query by selecting **Search** or pressing **Enter**.
5. Review the results. The sample Dataset includes various data columns such as `_raw`, `source`, `host`, `dataSource`, and `datatype`.



Results of a query using the `limit` operator

Extract Information from the `_raw` Field by Using the `extract` Operator and Parser

Notice in the previous example that the useful information in the `_raw` field is unparsed. To analyze the data effectively, you need to extract and parse the information into separate fields.

Write a Query to Parse the `_raw` Field

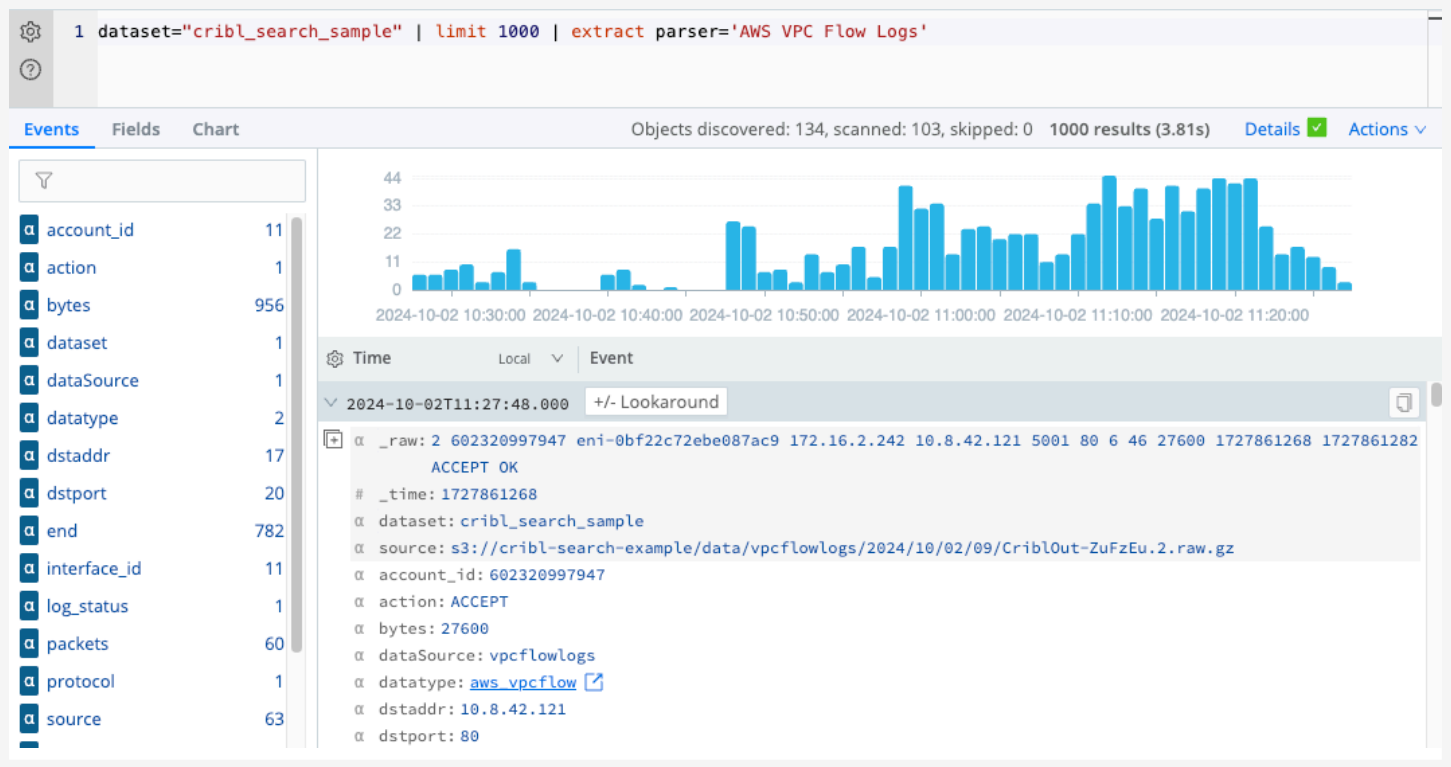
To parse the data, you can use the `extract` operator along with the appropriate Parser (in this case, **AWS VPC Flow Logs**).

1. Copy this query to your clipboard:

```
dataset="cribl_search_sample" | limit 1000 | extract parser='AWS VPC Flow Logs'
```

Copy | Run in Cribl Search

2. Paste the query into the search box and select **Search** or press **Enter**.
3. Check that your parsed results are similar to the following example. The actual data in the rows might differ because the rows are selected arbitrarily. The parsed results should now contain separate fields for each data point in the `_raw` field.



Results of a query using the extract operator

Now that the data is parsed and organized into separate fields, you can perform further analysis and create more advanced queries to gain insight into the VPC flow logs.

Filter Data by Using the `where` Operator

In the previous sections, you learned how to extract and parse information from the VPC flow logs Dataset. Now let's explore how to filter the data using the `where` operator to answer specific questions about the VPC flow logs.

Use the `where` Operator

All the operators you've used so far have returned selected columns. Now let's take a look at specific rows of the data.

The `where` operator filters results that satisfy a certain condition. In this first example, you'll compare an integer column to a minimum value by using the numerical operator greater-than (`>`). Specifically, you want to see only flows that have a certain number of packets. So you'll look at rows of data where the number of packets is greater than a given threshold.

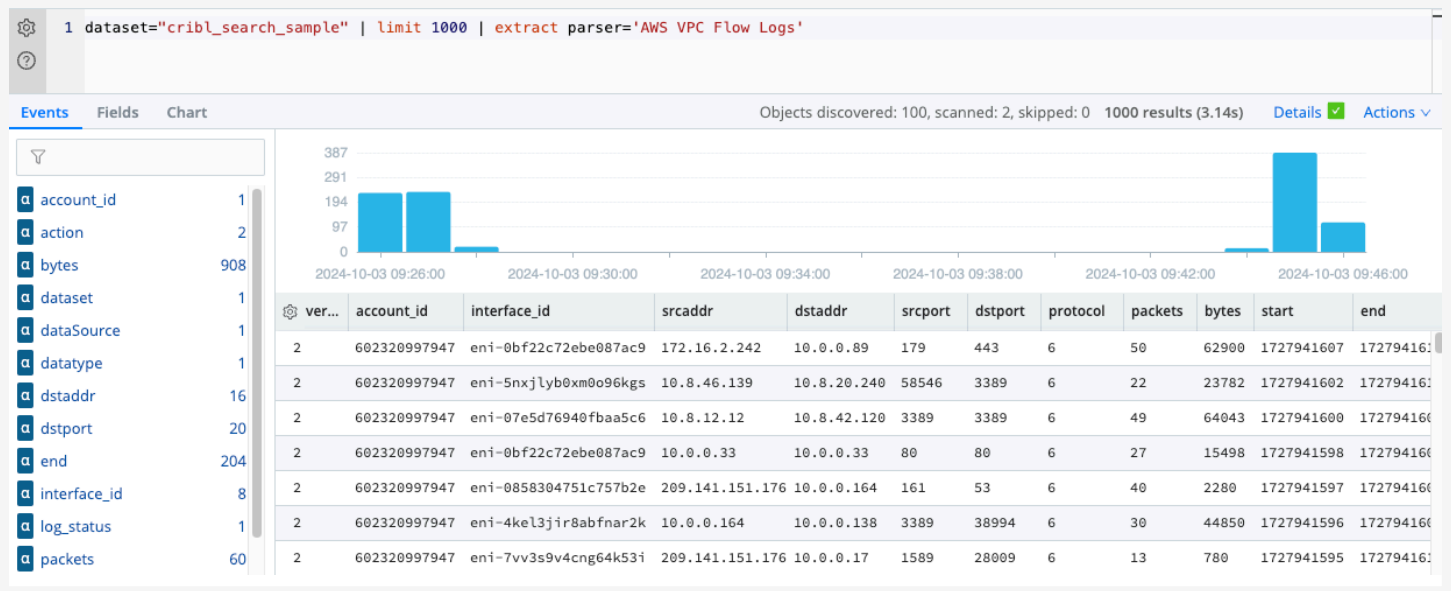
Run the following query:

```
dataset="cribl_search_sample"
| extract parser='AWS VPC Flow Logs'
| where packets > 5
| project _time, srcaddr, dstaddr, packets
| take 10
```

Copy Run in Cribl Search

The `extract parser` line is superfluous, because Cribl Search applies the parser automatically, but it's included for illustration.

Notice that all rows returned have packet values greater than 5.

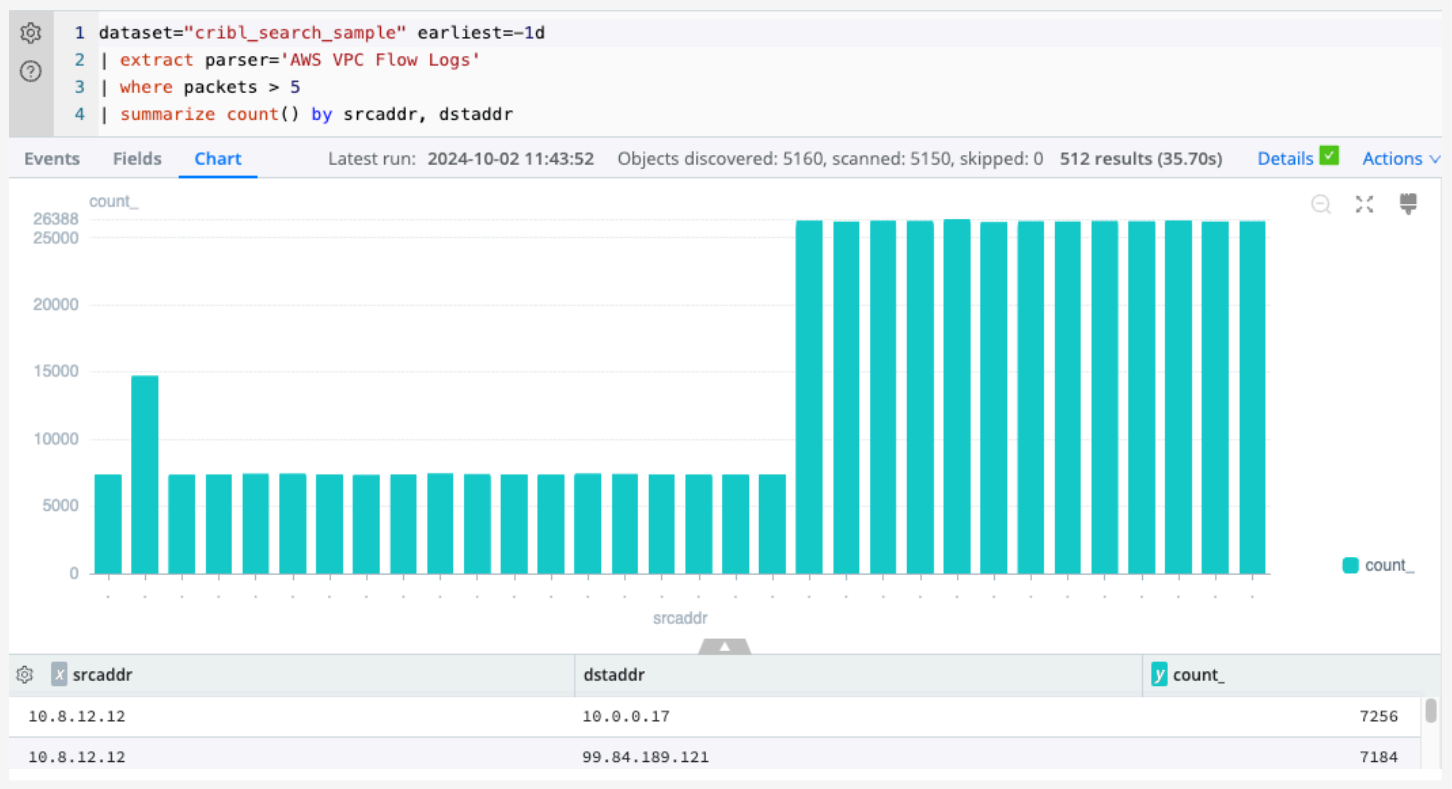


Results of a query using the where operator to return values

Similarly, you can filter for flow events that have a start time that's more recent than a certain time. For example, run the following query, where `earliest=-30m` means last 30 minutes:

```
dataset="cribl_search_sample" earliest=-30m
| extract parser='AWS VPC Flow Logs'
| where packets > 5
| summarize count() by srcaddr, dstaddr
```

Copy Run in Cribl Search



Results of a query using the where operator to filter for start times

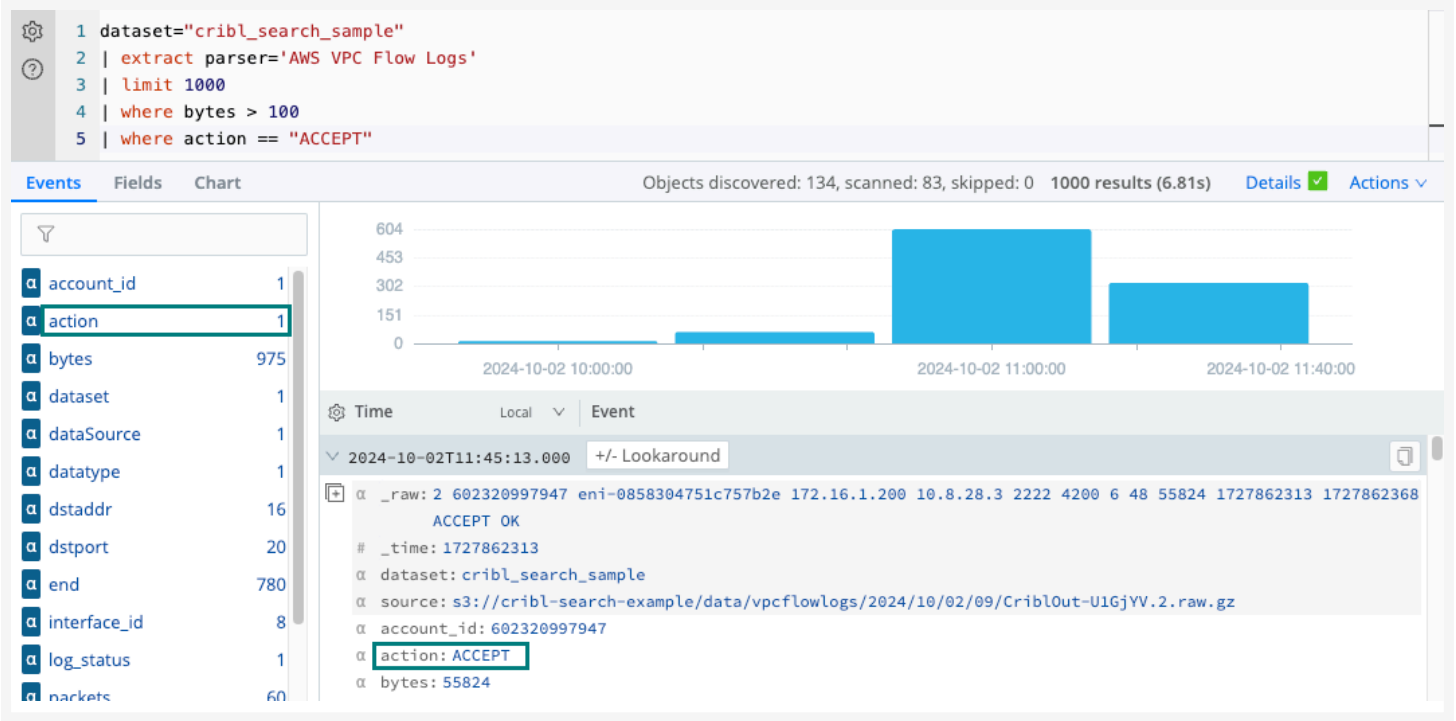
Filter by Using a String Value

Let's narrow down the flows to those that were accepted. Run the following query, which uses a second `where` operator with the string value "ACCEPT":

```
dataset="cribl_search_sample" bytes > 100 and action = "ACCEPT"
| limit 1000
```

Copy Run in Cribl Search

Notice that all records returned from this query have the action "ACCEPT" and bytes greater than 100.



Results of a query using the where operator to filter for a string value

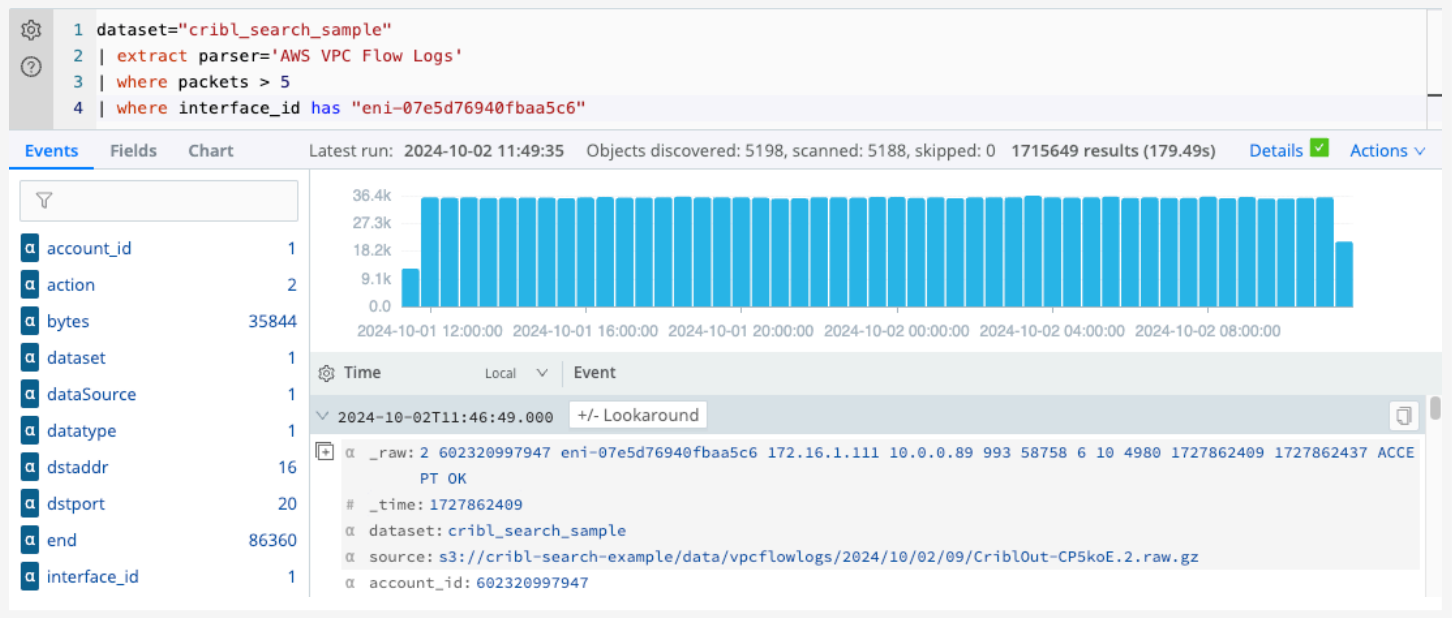
Filter by Using the `has` Operator

You can search for specific strings in the flow log data by using the `has` operator. The `has` operator is a case-insensitive search that matches a full term.

For example, let's filter the VPC flow logs by a specific interface ID:

```
dataset="cribl_search_sample"
| extract parser='AWS VPC Flow Logs'
| where packets > 5
| where interface_id has "eni-07e5d76940fbaa5c6"
```

Copy | Run in Cribl Search



Results of a query using the has operator

Filter on Time Range Values

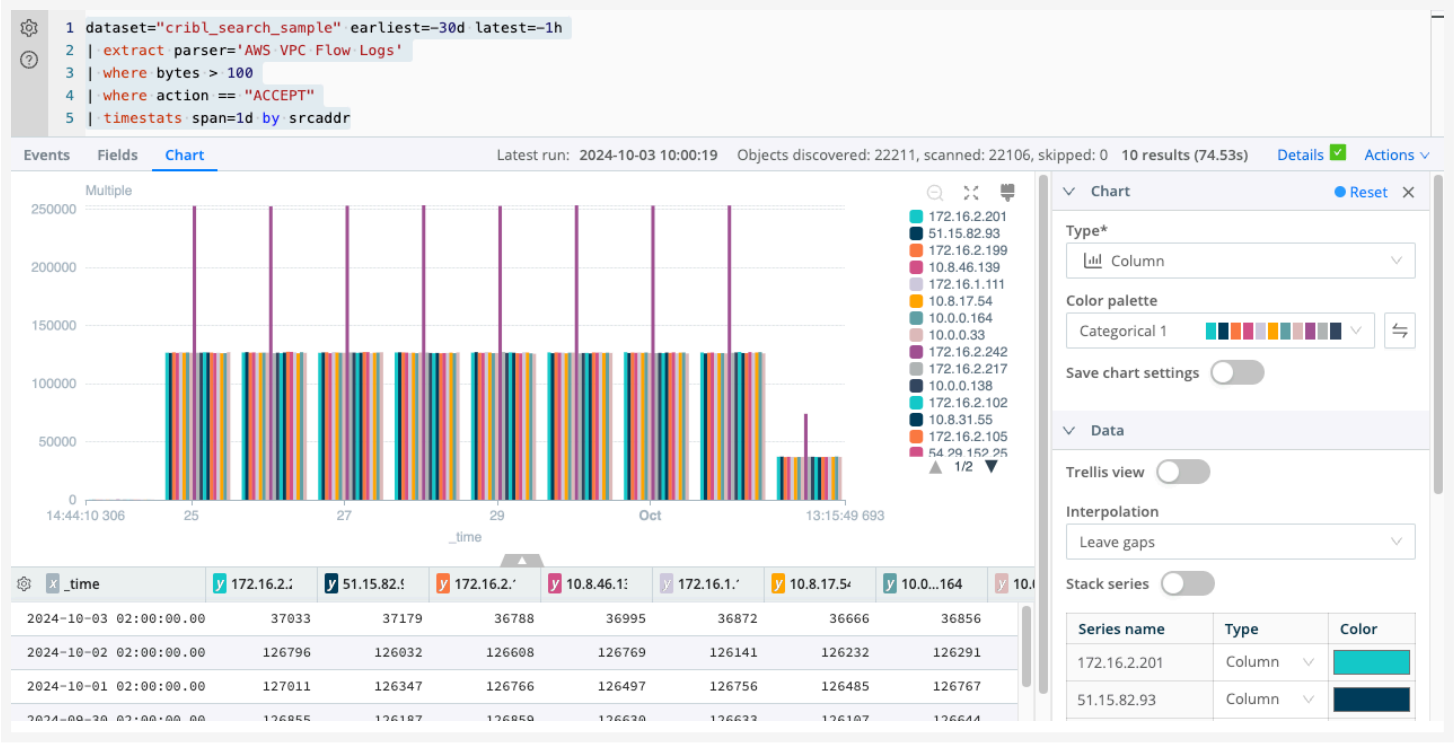
Let's look more closely at the flows within a specific time range. Because time ranges are bounded by two extremes, it's most efficient to limit your query to events within these two times.

To construct a date range, use `earliest` and `latest` at the top level, with the following syntax: `[+|-]#TimeUnit[@SnapToTimeUnit]`

Let's incorporate the time range into a kind of query you've already seen. Run the following query:

```
dataset="cribl_search_sample" earliest=-35m latest=-5m
| limit 1000
| extract parser='AWS VPC Flow Logs'
| where bytes > 100
```

Copy Run in Cribl Search



Results of a query filtering on a time range

In this query, you filter the events in the Dataset by a specific time range, from 35 minutes ago to 5 minutes ago, using the `earliest` and `latest` operators. You then extract and parse the data using the `AWS VPC Flow Logs` Parser. To focus on flows with more than 100 bytes and action "ACCEPT", use the `where` operator with the respective conditions.

Finally, you use the `timestats` operator to aggregate the data using a daily time span (`span=1d`) and group the results by `srcaddr`.

The query will return results that show the aggregated statistics of VPC flows with action "ACCEPT" and bytes greater than 100 for each `srcaddr`, separated by day, during the specified time period.

Calculate Available Events by Using the `count` Operator

Now that you've refined your query using helpful operators, such as the `limit` operator that returns a specified number of events, suppose you want to determine the total number of events available within a specific Dataset.

In this case, you can use the `count` operator. For example, say you want to count VPC flow events where bytes are greater than 100 after extracting the logs using the AWS VPC Flow Logs parser.

To do this, run this query:

```
dataset="cribl_search_sample"
| limit 1000
| where bytes > 100
```

Copy | Run in Cribl Search



Results of a query using the count operator

In this example, you first use the `extract` operator to parse the data in the `cribl_search_sample` Dataset using the `AWS VPC Flow Logs` Parser. Next, filter the extracted VPC flow logs for flows with bytes greater than 100. Finally, use the `count` operator to calculate the total number of events that meet this criterion.

The query will return a single record containing the count of VPC flow events with bytes greater than 100.

Summary

Throughout this tutorial, you've explored various aspects of Cribl Search queries and their application to the analysis of VPC flow logs. Here's a recap of what you've learned:

1. **Return a specific number of rows by using the `limit` operator**
   You learned how to get a sample of rows from the Dataset using the `limit` operator, which provided quick insight into the structure of the data.
2. **Extract information from the `_raw` field by using the extract operator and Parser**
   You extracted and parsed information from the `_raw` field by applying the `extract` operator along with the `AWS VPC Flow Logs` Parser.

3. **Filter data by using the `where` operator**
   You explored how to filter the data using the `where` operator to answer specific questions about the VPC flow logs, such as filtering based on certain conditions, string values, or time ranges.
4. **Calculate the number of events by using the `count` operator**
   You learned how to use the `count` operator to find the total number of input events in a Dataset that satisfy specific conditions, allowing you to get a quick overview of the data that met your filtering criteria.

By mastering these operators and techniques, you can effectively analyze and gain insights from the VPC flow logs Dataset. Armed with this knowledge, you can perform more advanced data analysis, create more complex queries to answer specific questions, and better understand the behavior and trends within your network flows.

3.3. Write Queries Using Cribl Copilot

Use the Cribl Copilot KQL assistant to generate Kusto Query Language expressions from natural language.

The Cribl Copilot KQL assistant enables you to define search queries in natural language. The assistant translates these into Kusto Query Language (KQL). You can use all basic KQL features – including grouping, aggregating, filtering, and sorting – without having to write anything in KQL. You'll usually get results in around 5 to 30 seconds.

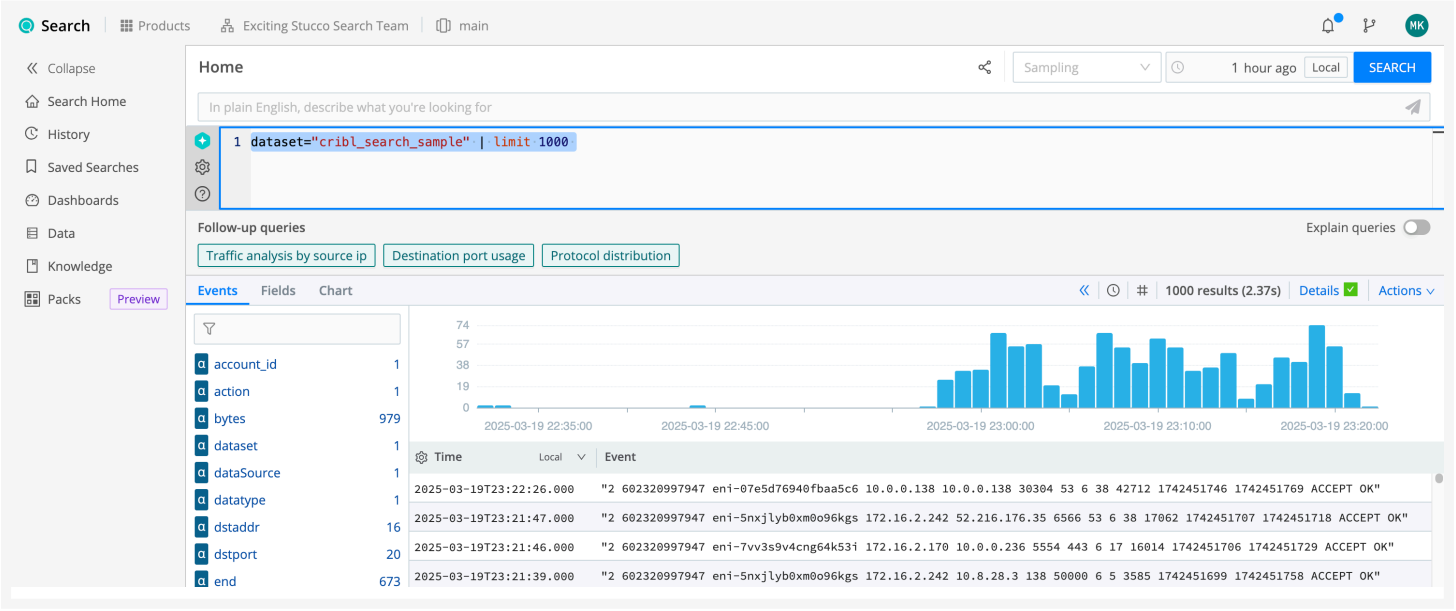
Additional options here are:

- Selecting from automatically suggested follow-up queries.
- Pasting KQL queries into the assistant to generate natural-language summaries of their logic.
- Pasting problematic KQL queries into the assistant, to debug syntax that's triggered errors.

To globally enable or disable Cribl Copilot for all Cribl products in your Organization, select **Settings** > **Global** > **AI Settings**.

Select from Cribl Copilot Suggestions

On the **Search Home** page, below the query box, Cribl Copilot automatically suggests follow-up queries that you might find useful, based on your most recently used query and Dataset, and on other recent search history.

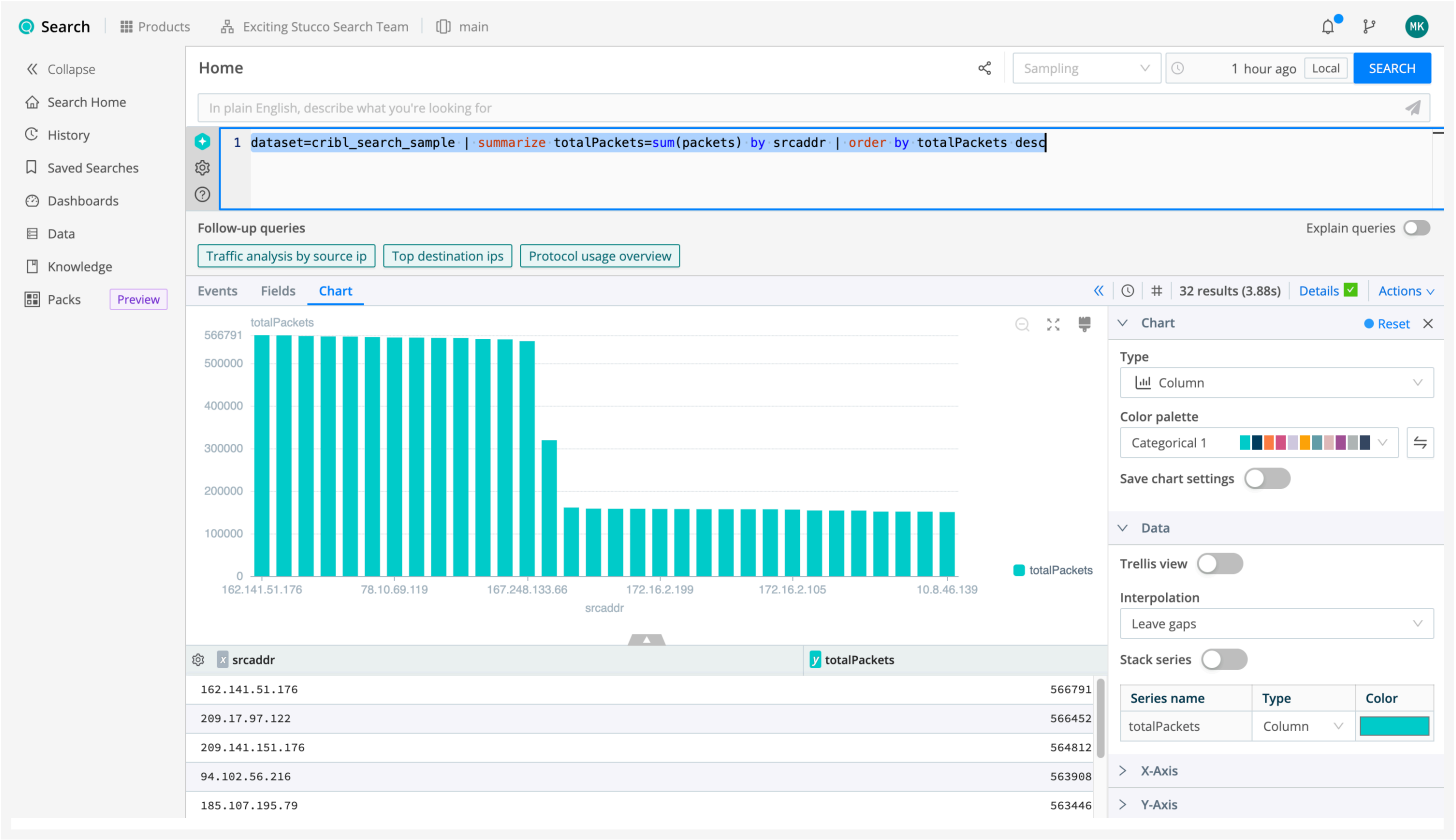


Cribl Copilot autocomplete options

You have multiple options to respond to these suggestions, covered in the following subsections.

Accept a Suggestion

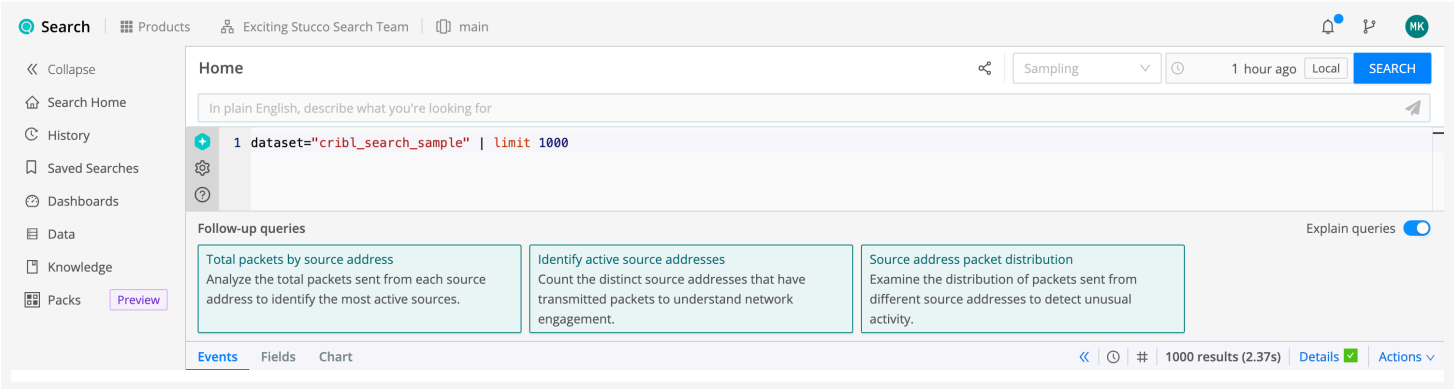
You can select one of the three follow-up query tiles. As shown below, this will immediately execute the query, while placing it in the query box for you to refine, as desired.



Suggested query after selection

Request Explanations

You can toggle on **Explain queries** to expand all the suggested-query tiles. This will reveal details about each suggestion.



Explain queries selected

Roll Your Own

You can simply ignore the suggestions, and build your new follow-up queries in the query box. Or, use the upper query box to describe your desired search in plain (natural) language. As described in the next section, Cribl Copilot will generate a corresponding query in KQL.

Generate a Query Using Cribl Copilot

To generate a KQL query from natural language:

1. On the **Search Home** page, beside the search query box, select the Cribl Copilot button .
2. In the upper query box, describe what you want to find, using ordinary language.
3. Select Send to generate a KQL version of your query.
4. Edit the resulting query if you want to make it more precise, and then select **Search**.

> You can also use the Cribl Copilot chatbot to generate queries. Select the Cribl Copilot button at the bottom right of any page, and then from the left drop-down select **Generate KQL**.
>
> Unlike the KQL assistant, the chatbot has no access to Cribl Search, your query history, or your Datasets. So KQL generation here is limited. However, this might be a good option for organizations with strict privacy policies.

How to Use Prompts Successfully

To ensure the best results from your natural language query:

- **Be specific.** If you know the names of specific Datasets or fields, or have other constraints, include these in the prompt. Example: `Sort results by column <my_timestamp>`
- **Refine your queries.** If the generated query is not exactly what you were looking for, refine your request to improve the result.
- **Provide feedback.** Select the thumbs-up or thumbs-down button to share feedback on responses. This helps improve the accuracy and relevance of Cribl Copilot.

How the KQL Assistant Works

When you enter a natural language query using the Cribl Copilot KQL assistant, a large language model (LLM) uses information about KQL, your 10 last-used Datasets, and the query you've entered, and combines this information to generate a valid KQL query.

3.4. Common Query Examples

Learn common ways of searching data.

Show n Rows

Let's see some data. What's in a random sample of five rows?

```
dataset="cribl_search_sample"
 | take 5
```

Copy Run in Cribl Search

Select a Subset of Fields

Use `project` to pick out only the fields you want.

```
dataset="cribl_search_sample"
 | project method, source, status, url
```

Copy Run in Cribl Search

Returns:

method	source	status	url
GET	file://opt/cribl/log/access.log	200	/api/v1/health
…	…	…	…

Filter by Expression

We'll use an expression to find specific field and value matches. Let's see only `GET` events in the `access.log`:

```
dataset="cribl_internal_logs" source="*/access.log" method="GET"
 | project method, source, status, url
```

Copy Run in Cribl Search

Returns:

method	source	status	url
GET	file://opt/cribl/log/access.log	200	/api/v1/health
GET	file://opt/cribl/log/access.log	...	...

Compute Derived Fields

Create a new field by computing a value in every row:

```
dataset="cribl_search_sample"
 | limit 5
 | extend query=query()
```

Copy Run in Cribl Search

This query uses the context function `query()`.

It's possible to reuse a column name and assign a calculation result to the same column.

Example:

```
 | extend x = x + 1, y = x
 | extend x = x + 1
```

Copy

Aggregate Groups of Rows

To count the number of events by `src` we'll use the count function.

```
dataset="cribl_search_sample"
 | limit 1000
 | summarize cnt=count() by srcaddr
```

Copy Run in Cribl Search

Returns:

src	cnt
10.255.0.248	2978
...	...

The `summarize` operator groups together rows that have the same values in the `by` clause, and then uses an aggregation function (for example, `count`) to combine each group in a single row. In this case, there's a row for each `src` and a field for the count of rows in that `src`.

You can use several aggregation functions in one `summarize` operator to produce several computed columns. For example, we could get the count of events per `src`, and the sum of unique types of events per `src`.

```
dataset="cribl_search_sample" dataSource="access*"
 | limit 1000
 | summarize eventCount=count(), useragentCount=dcount(useragent) by clientip
```

Copy Run in Cribl Search

Returns:

src	eventCount	message
10.255.0.248	5610	2
...	...	...

In the results of a `summarize` operator:

- Each field is named in the `by` clause.
- Each computed expression has a field.
- Each combination of `by` values has a row.

Display Multiple Series

Use multiple values in a `summarize by` clause to create a separate row for each combination of values:

```
dataset="cribl_search_sample" dataSource="access_combined"
 | limit 1000
 | summarize count() by host, clientip
```

Copy Run in Cribl Search

Display a Time Series

The `timestats` operator aggregates events by time periods or bins, which is excellent for displaying a time series. Here we slice the results into one-minute sections, or bins:

```
dataset="cribl_search_sample"
 | limit 1000
 | timestats span=1m
```

Copy Run in Cribl Search

Display Results as Events or Tables

Use the `render` operator to display results as a list of events or a table.

To display results as a list of events under the **Events** tab:

```
dataset=$vt_dummy event<100
 | extend parity=iif(event%2==0, 'even', 'odd')
 | project event, parity
 | render table
```

Copy Run in Cribl Search

To display results as a table under the **Chart** tab:

```
dataset=$vt_dummy event<100
 | extend parity=iif(event%2==0, 'even', 'odd')
 | project event, parity
 | render event
```

Copy Run in Cribl Search

Search Cribl Lake

You can search Cribl Lake Datasets the same way you use regular Cribl Search Datasets:

```
dataset="my_lake_dataset"
```

Copy

See Searching Cribl Lake for more examples.

Search Arrays

In this example, `arr1` is an array of strings, and `arr2` is an array of numbers:

```
dataset="ClickhouseData"
arr1[0]='s1'
arr2[0]=1
| limit 1000
```

Copy

As shown here, you can reference specific elements in an array – using zero-based index notation – to retrieve only the data you specify. Your query must specify the field name or data point at each array index. (Cribl Search does not support querying on the index to discover array members.)

Where a Dataset Provider returns array columns as strings, Cribl Search parses these columns back to arrays.

Search Nested JSON Objects

In this example, `bar` is a field nested within three levels of JSON objects:

```
dataset="my_warehouse" json1.obj2.nestedObj2.nestedObj3='bar'
 | limit 1000
```

Copy

Subqueries

Subqueries allow you to generate different sets of results in a single search. This lets you perform dynamic lookups with no need to create static, stored lookup tables.

You can use subqueries to append one set of results to another, through the `union` operator. You can also use them to merge data coming from different Datasets and Dataset Providers, with the help of the `join` operator.

There are two types of subqueries:

- Inline subqueries
- `let` subqueries

Inline Subqueries

An inline subquery is nested inside another query, using round brackets `()`.

You may find inline subqueries easier to write than `let` subqueries, but:

- You can reference their results only as vectors, using `union` or `join`.
- If you opt to use the `cribl` operator, you must explicitly include it in the inline subquery.

```
// main query
dataset=$vt_dummy event<10
 | extend foo=42
 | union (
    // inline subquery
    cribl dataset=$vt_dummy event<10
     | extend bar=24
 )
```

Copy Run in Cribl Search

You could achieve the same result as above by using a `let` subquery:

```
let subquery = dataset=$vt_dummy event<10
 | extend bar=24;

dataset=$vt_dummy event<10
 | extend foo=42
 | union subquery
```

Copy Run in Cribl Search

The following example uses an inline subquery to join two different scopes of data:

```
dataset=$vt_dummy event<10
 | extend foo=42
 | join (
    // inline subquery
    cribl dataset=$vt_dummy event<10
     | extend bar=24
 ) on event;
```

Copy Run in Cribl Search

`let` Subqueries

You define `let` subqueries at the beginning of a query, using `let` statements. You can then reference them by name, like this:

```
// each let statement ends with a semicolon
let x = 1000;

dataset="cribl_search_sample"
 | limit x
```

Copy Run in Cribl Search

Depending on what a `let` subquery evaluates to, you can reference it as a vector or as a scalar.

Similarly to inline subqueries, you can use `let` subqueries with the `join` and `union` operators:

```
let subquery = dataset="$vt_dummy" event < 10
 | extend bar=24;

dataset="$vt_dummy" event < 10
 | extend foo=42
 | join subquery on event
```

Copy Run in Cribl Search

`let` subqueries can also reference one another, for example:

```
let subquery = dataset="$vt_dummy" event < 1
 | extend bar=24;

dataset="$vt_dummy" event < 10
 | extend foo=42, subqueryBar=subquery.bar
```

Copy | Run in Cribl Search

Reference a `let` Subquery as a Vector

When a `let` subquery evaluates to a vector, you can use its results to filter the results of your main query, with help of the `in`/`!in`/`in~`/`!in~` operators.

Mind that this functionality is not supported by the implicit `cribl` operator, so you need to use `search`, `find`, or `where`. For example:

```
// dummy list of blocklisted IPs
let pretendBlocklistedIps = dataset="cribl_search_sample" dataSource="vpcflowlogs"
 | limit 10
 | distinct srcaddr
 | project blocklistedIp=srcaddr;

dataset="cribl_search_sample" dataSource="vpcflowlogs"
 | where srcaddr in pretendBlocklistedIps.blocklistedIp;
```

Copy | Run in Cribl Search

Reference a `let` Subquery as a Scalar

To reference a `let` subquery as a scalar, mind the following:

- The subquery needs to produce exactly one event.
- You must specify a field, using dot notation (`subquery1.text`). You can also reference a nested field (`subquery1.headers.test`).
- The field you specify must be a scalar value, not an object.
- You can't specify fields that don't exist.

See the following example, which calculates the 95th percentile of requests in the Cribl internal logs.

```
let num_percentile = 95;

let response_time_percentile = dataset="cribl_internal_logs" method status response_t
 | summarize total=count(), response_time=percentile(response_time, num_percentile);

let long_responses = dataset="cribl_internal_logs" method status response_time
 | where response_time > response_time_percentile.response_time
 | summarize count=count(), maxResponseTime=max(response_time);

print strcat('There are ', long_responses.count, ' requests in the ', num_percentile,
```

Copy | Run in Cribl Search

Subqueries and Time Ranges

A subquery can have its own time range. However, the time range of the main query (either set in the query itself, or configured through the UI) still applies to the final results.

For example, the following search won't get you any `earlysample` results, because they're outside of the time range of the main query:

```
// subquery time range between 50 and 45 minutes ago
let earlysample = dataset="cribl_search_sample" earliest=-50m@m latest=-45m@m
 | extend stage="one"
 | limit 100;

// main query time range between 10 and 5 minutes ago
dataset="cribl_search_sample" earliest=-10m@m latest=-5m@m
 | extend stage="two"
 | limit 100
 | union earlysample
```

Copy | Run in Cribl Search

To go around this, you can apply that second time range to another subquery, and then let the main query contain both time ranges:

```
let earlysample = dataset="cribl_search_sample" earliest=-50m@m latest=-45m@m
 | extend stage="one"
 | limit 100;

let latesample = dataset="cribl_search_sample" earliest=-10m@m latest=-5m@m
 | extend stage="two"
 | limit 100;

dataset="cribl_search_sample" earliest=-50m@m latest=-5m@m
 | limit 100
 | union earlysample, latesample
```

Copy Run in Cribl Search

3.5. Use Search Results

See how to export, reuse, and manage search results.

When you obtain each search's results, you have several options to export the results from Cribl Search, and to reuse them within the product. You can also control how long Cribl Search retains your results.

Export Search Results

Use the `export` operator to export search results to either a lookup table or to a Cribl Lake Dataset.

```
dataset="cribl_search_sample"
 | export to lake my_lake_dataset
```

Copy

Reuse Search Results

You can reuse the results of previous searches. This might let you avoid running the same search multiple times, saving you time and resources. Your options include:

- View the Cached Results of a Previous Search
- Search the Result Set of a Previous Search
- Use the Results of a Previous Search in a New Query
- Rerun a Search
- Automatically Reuse Search Results

You can access previous search results for as long as they're kept in the system. For retention details, see:

- Manage Search Results

View the Cached Results of a Previous Search

You can quickly display the cached results of any search kept in **History**. This doesn't consume any credits.

1. In Cribl Search, select **History**.
2. Select the row of the search you're interested in.

   This displays the cached results of the search.

You can also select **View Results** in the **Details** of each search.

Search the Result Set of a Previous Search

You can quickly query the results of a previous search. This doesn't rerun the original query itself, but runs an actual search on the result set, so can lead to minor credit consumption.

1. In Cribl Search, select **History**.
2. In the row of the search you're interested in, select the Actions ⋯ button at the far right.
3. Select **Search the Results**.

   In a new window, Cribl Search runs a search on the result set, using the `$vt_results` virtual table. You can now modify the query for further analysis.

You can also select **Search the Results** in the **Details** of each search. Read more about virtual tables here.

Reuse the Results of a Previous Search in a New Query

When writing a query, you can reference the result set of a previous search, using the `$vt_results` virtual table. This lets you treat the results as a regular Dataset.

While doing this doesn't rerun the original query itself, it runs an actual search on the result set, so can lead to minor credit consumption.

Get the ID of the search you're interested in (you'll find it in the **Details** tab), and use the `jobId` predicate:

```
dataset="$vt_results" jobId="1704236905683.wgocax"
```

Copy

You can also load the results of multiple searches at once, for example:

```
dataset="$vt_results"
 | where jobId > "1704236905600" and jobId < "1704236906000"
```

Copy

To reuse the results of a saved or scheduled search, add the search name by using the `jobName` predicate. This will load the results of the latest execution of the saved search. For example:

```
dataset="$vt_results" jobName="mySavedSearch"

// or, for example
dataset="$vt_results"
 | where jobName startswith "my"
```

Copy

To load a specific execution of a saved search, use the `execution` parameter, for example:

```
// load the last run of mySavedSearch (default)
dataset="$vt_results" jobName="mySavedSearch" execution = 0

// load the run before last
dataset="$vt_results" jobName="mySavedSearch" execution = -1
```

Copy

For more information and examples, see the `$vt_results` page.

Rerun a Search

You can easily run a new search that uses exactly the same query text and settings as a previous search.

1. In Cribl Search, select **History**.
2. In the row of the search you're interested in, select the Actions ⋯ button at the far right.
3. Select **Rerun**.

You can also select **Rerun** in the **Details** of each search.

In certain cases, rerunning the search may not be necessary, and you'll find it faster and cheaper to query the result set or simply view the cached results.

Automatically Reuse Search Results

When writing a query, you can allow Cribl Search to automatically reuse the results of analogous searches that were run recently in your organization. This is especially useful for configuring Dashboards, or when you run the same query multiple times, potentially producing the same results.

Cribl Search treats two searches as analogous when they share the same:

- Search plan: both searches can be broken down into the same set of pipelines. For example, `sort by x desc | limit 10` has the same search plan as `top 10 by x`.
- Datasets: both searches touch (and have access to) the same Datasets.
- `set` statements that can affect results.
- Relative time range.
- Sample rate.

When a search reuses previous results, the **Details** tab describes such a search as an **alias job**, and provides the ID of the job that produced the original results. To hide potentially sensitive information, you can't access the logs, metrics, and diagnostics of the alias job or the original job.

To allow for automatic reuse, at the beginning of your query add a `set` statement with the `allow_previous_results` option set to a time interval.

For example, to allow Cribl Search to reuse results from the last 10 minutes, add the following statement:

```
set allow_previous_results="10min";

// here, add your query
```

Copy

Manage Search Results

Search results are stored temporarily on your Organization's Leader. We use an industry-standard AES-256 algorithm to encrypt your data at rest.

You can control how long Cribl retains your search results at **Settings** > **Search** > **Limits** > **Search history TTL**. For details, see Limits.

3.6. Cribl Search Notebooks

Run an entire investigation in one tab, and share your work with others.

Why Use Notebooks

A Notebook is a document-like workspace in Cribl Search where you and other data analysts can combine search queries, data visualizations, and Markdown notes into persistent, shareable investigations. Here's what you can do with Notebooks:

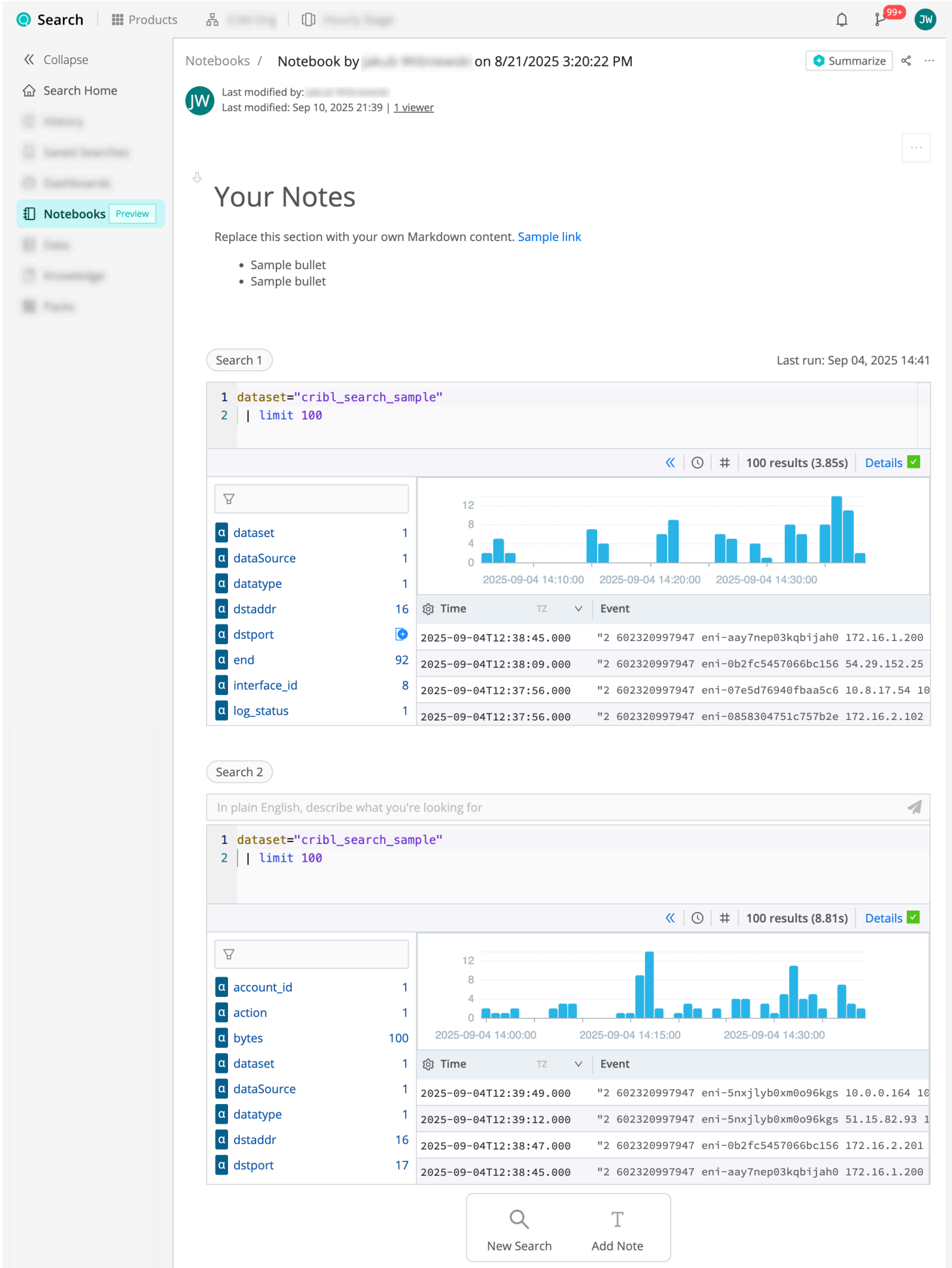
- **Iterate**: Run and maintain multiple searches next to one another, for faster, more-efficient investigations.
- **Annotate**: Add context and clarity with markdown notes for rich storytelling.
- **Collaborate**: Share your work through fine-grained edit or read-only access.
- **Control**: See who last edited your Notebook, and when.
- **Summarize**: Use Cribl Copilot to generate summaries of your findings, and to run queries from natural-language prompts.

Start working with Notebooks from anywhere in Cribl.Cloud: On the top bar, select **Products** > **Search** > **Notebooks**.

What's in a Notebook

A Notebook consists of **search cells** and **note cells**.

When a Notebook is shared, **Maintainers** can add and edit cells, drag cells up and down, and clone the Notebook. **Read Only** users can view the cells and their results. Read more about Notebooks access.


Search
Products
99+
JW
Collapse
Search Home
Notebooks
Preview
Notebooks / Notebook by on 8/21/2025 3:20:22 PM
Summarize
Last modified by:
Last modified: Sep 10, 2025 21:39 | 1 viewer
Your Notes
Replace this section with your own Markdown content. Sample link
Sample bullet
Sample bullet
Search 1
Last run: Sep 04, 2025 14:41
1 dataset="cribl_search_sample"
2 | limit 100
100 results (3.85s)
Details
dataset 1
dataSource 1
datatype 1
dstaddr 16
dstport
end 92
interface_id 8
log_status 1
12
8
4
0
2025-09-04 14:10:00
2025-09-04 14:20:00
2025-09-04 14:30:00
Time
TZ
Event
2025-09-04T12:38:45.000 "2 602320997947 eni-aay7nep03kqbijah0 172.16.1.200
2025-09-04T12:38:09.000 "2 602320997947 eni-0b2fc5457066bc156 54.29.152.25
2025-09-04T12:37:56.000 "2 602320997947 eni-07e5d76940fbaa5c6 10.8.17.54 10
2025-09-04T12:37:56.000 "2 602320997947 eni-0858304751c757b2e 172.16.2.102
Search 2
In plain English, describe what you're looking for
1 dataset="cribl_search_sample"
2 | limit 100
100 results (8.81s)
Details
account_id 1
action 1
bytes 100
dataset 1
dataSource 1
datatype 1
dstaddr 16
dstport 17
12
8
4
0
2025-09-04 14:00:00
2025-09-04 14:15:00
2025-09-04 14:30:00
Time
TZ
Event
2025-09-04T12:39:49.000 "2 602320997947 eni-5nxjlyb0xm0o96kgs 10.0.0.164 10
2025-09-04T12:39:12.000 "2 602320997947 eni-5nxjlyb0xm0o96kgs 51.15.82.93 1
2025-09-04T12:38:47.000 "2 602320997947 eni-0b2fc5457066bc156 172.16.2.201
2025-09-04T12:38:45.000 "2 602320997947 eni-aay7nep03kqbijah0 172.16.1.200
New Search
Add Note

A Notebook with two sample searches

Start a New Notebook

From Scratch | From Current Search | From Saved Search | From History

You can create an empty Notebook, and then add searches and notes as needed.

1. Go to the **Notebooks** page in Cribl Search: On the top bar, select **Products** > **Search** > **Notebooks**.
2. Select **Add Notebook**. Your new Notebook opens and gets autosaved.
3. Start running searches and adding notes.

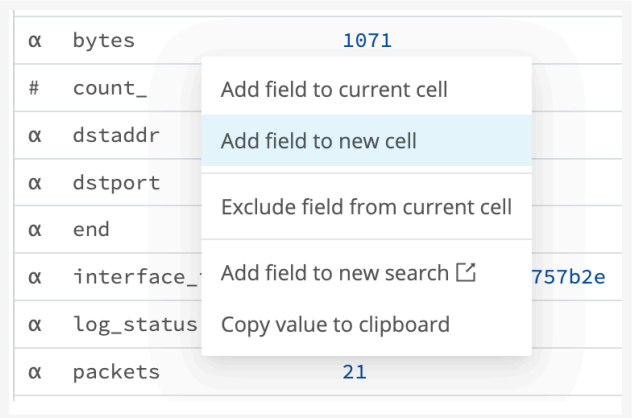
Add Searches to Your Notebook

Inside the Notebook | From Current Search | From Saved Search | From History

You can run multiple new searches directly from your Notebook. You need to be the Notebook Maintainer for this.

1. Open an existing Notebook or create a new one.
2. At the bottom of the Notebook, select **New Search**. A cell with a query box opens.
3. Run your query, as you would anywhere else in Cribl Search. To learn how, see some quick examples, or this tutorial: Write Your First Query. If your Organization has Cribl Copilot enabled, you can also use your search cells to run natural-language queries.
4. To add another query, select **New Search** again. You can start another search while the first one is still running.

Alternatively, you can create a new search cell by selecting a field within a field summary or event details drawer, and then selecting **Add field to new cell** from the resulting context menu.



Add Notes to Your Notebook

You can add markdown-formatted notes to your Notebook using headings, lists, links, and more. See Markdown Guide for basic syntax.

You need to be the Notebook Maintainer for this.

1. Open an existing Notebook or create a new one.
2. At the bottom of the Notebook, select **Add Note**. A new note cell opens.
3. Write your notes using markdown. The Notebook gets autosaved.

Summarize Your Notebook With Cribl Copilot

If your Organization has Cribl Copilot enabled, you can generate an AI summary of your Notebook findings.

1. Open an existing Notebook or create a new one.
2. In the top-right corner, select **Summarize**.

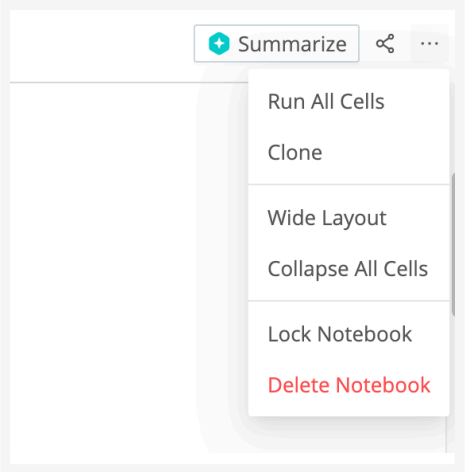
A summary of your Notebook appears in a new note cell at the top of the Notebook. You can edit the summary as needed.

Customize Notebook Display

To make your Notebook easier to skim, you can manage how much detail the Notebook displays. In the top-left corner of each cell, a Collapse/Expand toggle enables you to reduce the cell's vertical depth to a summary view.

In the top-right corner of the Notebook itself, the Actions ⋯ drop-down provides two toggles to manage the appearance of the whole Notebook:

- Select **Collapse All Cells** or **Expand All Cells** to control the vertical spread of all cells at once.
- Select **Wide Layout** or **Default Layout** to control the horizontal width available for Notebook contents.



Summarize and Actions controls

Export Notebook Search Results

You can export the results of a Notebook search as a CSV or NDJSON file.

1. Open an existing Notebook or create a new one.
2. In a search cell, select the Actions ··· button.
3. From the drop-down, select **Export as**, and then either **Export Results as CSV** or **Export Results as NDJSON**.

Export a Notebook Chart

You can export a Chart contained in a Notebook search cell, as a JPG or PNG file.

1. Open an existing Notebook or create a new one.
2. In a search cell, select the Actions ··· button.
3. From the drop-down, select **Export as**, and then either **Export Chart as JPG** or **Export Results as PNG**.

Share Your Notebook

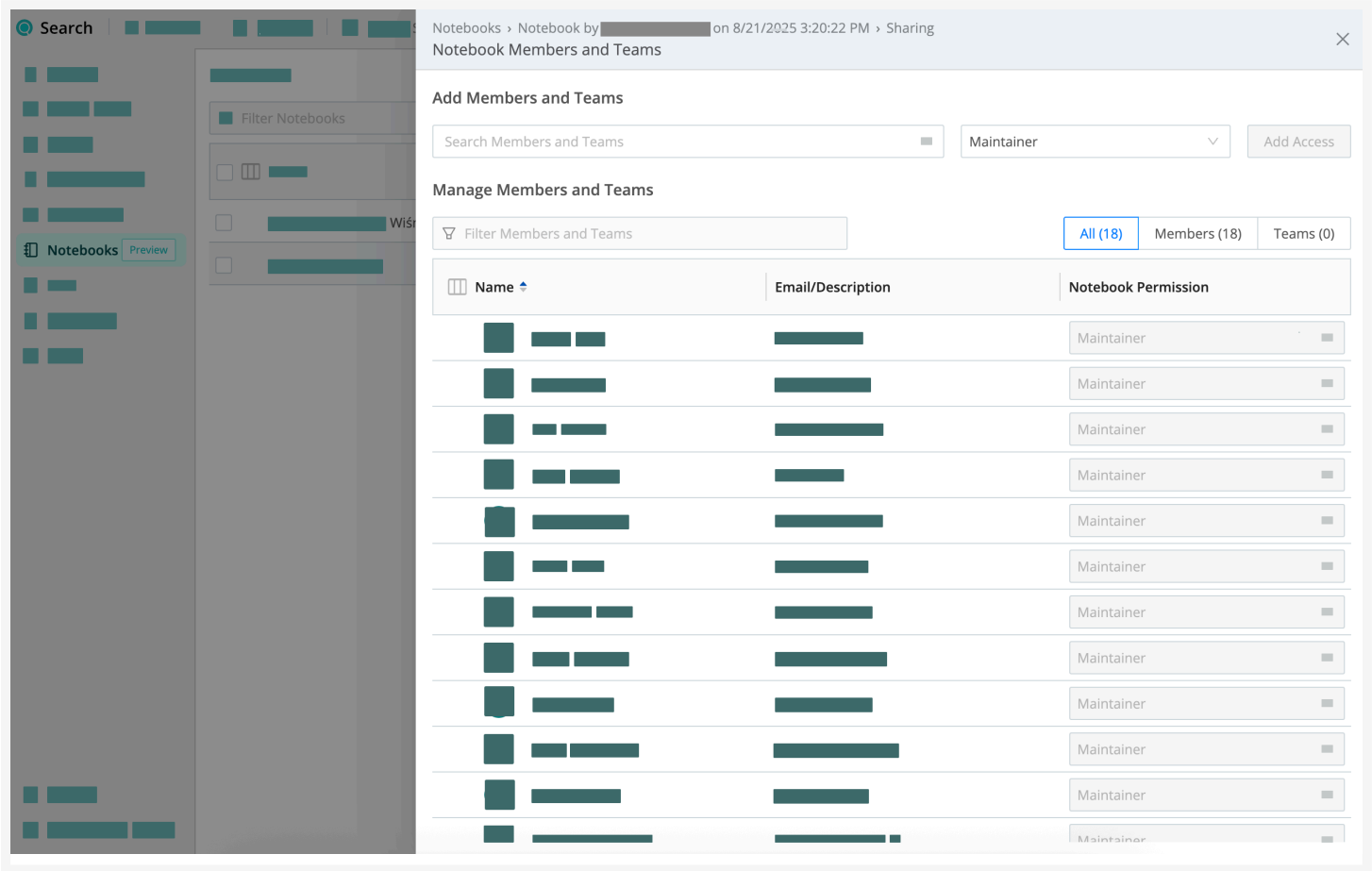
You can allow others to view or edit your Notebook at any time. For example, you might want to:

- Invite colleagues to join the investigation and contribute their expertise.
- Let others retrace your steps and pick up where you left off.
- Tell the full story behind your analysis, so stakeholders can understand how you reached your conclusions and review any assumptions you made.

You can share a Notebook with other Members or Teams. You need to be the Notebook Maintainer for this.

1. Open an existing Notebook or create a new one.

2. In the top-right corner, select the Share button. Now, you can see who has access to the Notebook and at what level.
3. Search for the Members or Teams you want.
4. Select the access level (**Read Only** or **Maintainer**). For details, see Notebooks Access.
5. Confirm with **Add Access**.



Sharing a Notebook

Notebooks Access

By default, your Notebooks are available to the Notebook creator, Search Admins, Organization Admins, and Organization Owners.

Search Member Type	Create Notebooks	List Notebooks	Delete Notebooks
Search Editor		Only own or shared	Only own or shared
Search User		Only own or shared	Only own or shared
Search Admin			
Organization Admin			

Search Member Type	Create Notebooks	List Notebooks	Delete Notebooks
Organization Owner			

When you share a Notebook, you can assign one of two access levels: **Read Only** or **Maintainer**. Here's what each level allows:

Action	Read Only	Maintainer
View the Notebook		
View Results		
Export Results		
Open in Cribl Search		
Rerun cells		
Add cells		
Edit cells		
Delete cells		
Share the Notebook		
Lock or unlock the Notebook		
Delete the Notebook		

Synchronize or Lock Edits

As a Notebook creator, or with a Maintainer or higher Permission: When you save changes to a Notebook, a pop-up will alert you to any changes that other collaborators have made since your last save. In this read-only state, you will be required to reload the Notebook before saving.

You also have the option to lock a shared Notebook into a read-only state for other collaborators (as well as yourself). This is useful if you need to freeze a completed investigation's results, to preserve their integrity against further changes.

In the Actions ⋯ drop-down at the top-right corner of the Notebook, select the **Lock Notebook** toggle to preserve the current Notebook state. Select **Unlock Notebook** to make the Notebook editable again.

Notebooks Retention

Notebooks have a hard-coded 30-day retention period to facilitate extended investigations. Exceeding the **Search history job limit** will cause other jobs to be removed before Notebook jobs, to respect this extension.

Goat-to Resources

Eating our own goat food. How Cribl support engineers use Notebooks to troubleshoot complex issues

Blog

3.7. Example Searches By Provider

See specific examples of searching data from different Dataset Providers.

Examples by Provider

For examples of searching data from a specific Dataset Provider, see:

- Searching Cribl Edge
- Searching Cribl Lake
- Searching Amazon S3
- Searching AWS API
- Searching Okta API
- Searching Zoom API

For more Dataset Providers, see the List of Supported Data Sources.

Search Multiple Providers

You can query data from multiple Dataset Providers at once. For an example, see: Searching Multiple Providers.

3.7.1. Searching Cribl Lake

Learn how to search your Cribl Lake data.

This page provides example searches that you can run with Cribl Lake and Lakehouses.

Setting Up

To begin integrating Cribl Search with these data lakes, first see Connect to Cribl Lake.

Certain operators, functions, and data types behave differently when you search Cribl Lake Datasets with (versus without) Lakehouse caching. To avoid unexpected search results, see Lakehouse Search Differences.

Examples

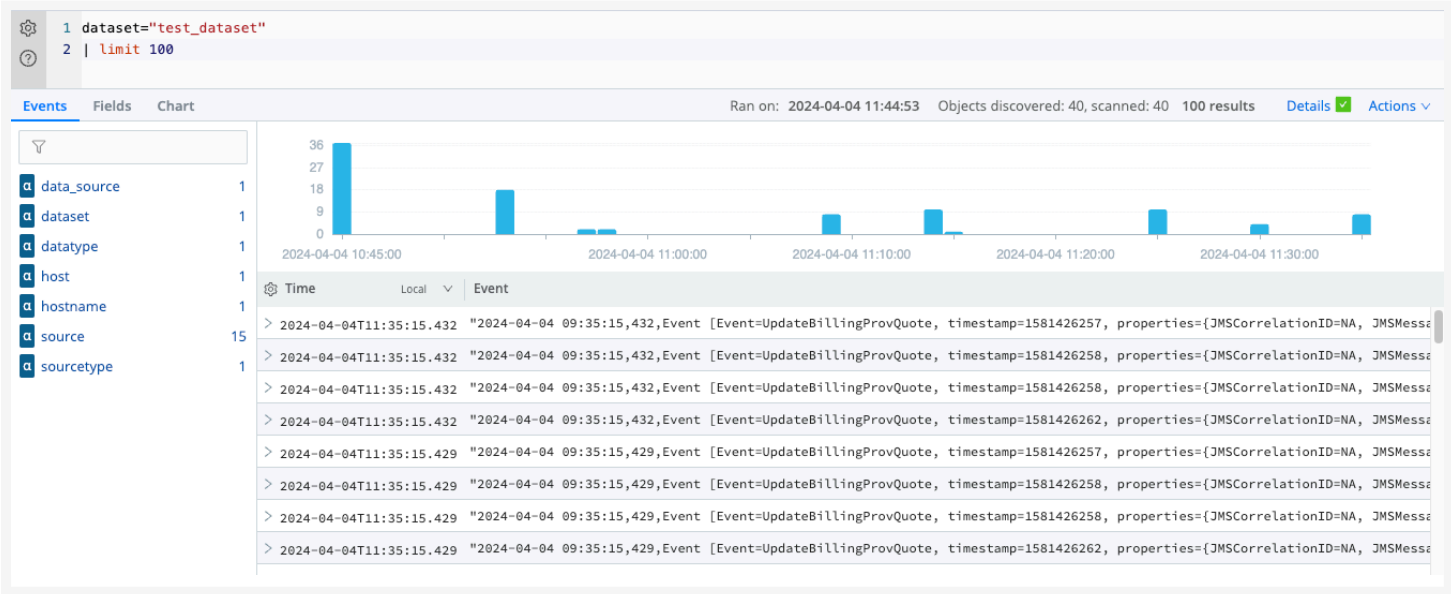
Use these examples as starting points for your own searches.

Basic Search

This search specifies a Dataset (`test_dataset`) and limits the number of results.

```
dataset="test_dataset"
| limit 100
```

Copy



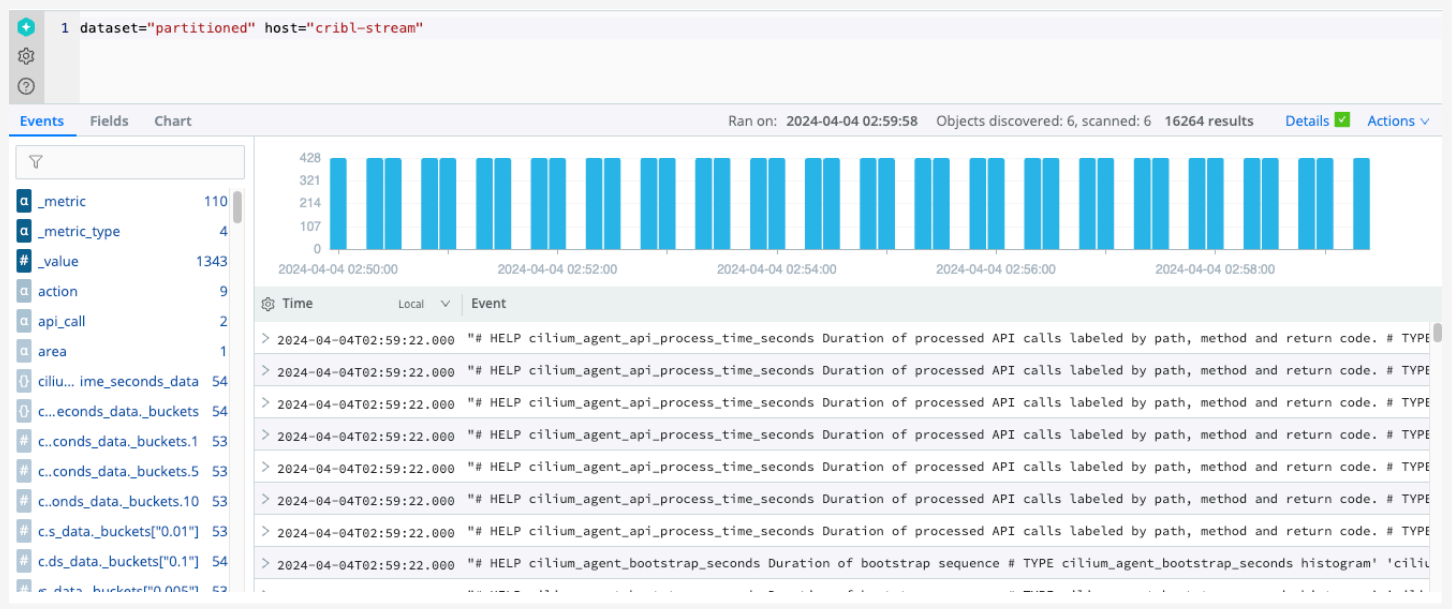
Sample Cribl Lake Search

Lake Partition

This search uses a Lake partition named `sourcetype` that is configured for the `partitioned` Dataset to speed up retrieval:

```
dataset="partitioned" host="cribl-stream"
```

Copy



Sample Cribl Lake Search: Using partitions

Exported Data

The `export` operator lets you export search results to a Cribl Lake Dataset. You can later search this Dataset to extract relevant data from it.

An efficient way to search exported data is to provide the search job ID to the `where` operator:

```
dataset="exported_data"
| where source contains "1713177481843.9AOqxI"
```

Copy

You can find the search job ID in search details after running it, or in **History**, in the **Search ID** column.

You can also label exported events using the `extend` operator and then include the added fields in your search. For example, during export you can include the user that performed the search:

```
dataset="cribl_search_sample"
| extend user = user()
| export to lake exported_data
```

Copy | Run in Cribl Search

You can then search for data by this user:

```
dataset="exported_data"
| where user == "John Doe"
```

Copy

3.7.2. Searching Cribl Edge

Learn how to search your Cribl Edge data.

Cribl Search comes with several Cribl Edge Datasets allowing you to quickly search Cribl Edge's internal logs and metrics that provide comprehensive information about an instance's status/health, inputs, outputs, Pipelines, Routes, Functions, and traffic. You can edit these built-in Datasets or create new ones to specify other logs anywhere in the filesystem that Edge can read.

Don't have a Cribl Edge Node? See the setup guide for Cribl Edge.

This document provides examples of searching Edge logs and metrics. To start, navigate to **Search Home**, where searches are run.



Searches over Cribl Edge execute in real time, at rest, directly on Edge Nodes. Broad or expensive queries can impact CPU and I/O on your nodes, especially over large time ranges. To minimize impact, see Best Practices for Searching Cribl Edge.

Best Practices for Searching Cribl Edge

- **Scope by Fleet**: Use the built-in `cribl_fleet` field to target a specific Fleet (for example, `cribl_fleet=MyFleet`). If you omit this filter, your query will run across all Edge Nodes in the system, which includes all configured Fleets.
- **Keep timeframes tight**: Start with the narrowest time range that can answer your question, then expand only as needed.
- **Avoid excessive wildcards**: Prefer specific field filters and paths over broad wildcard matches, especially in early query stages.
- **Validate your scope first**: Use **Explore** > **Files** in Fleet to confirm you are targeting the right locations before running large searches. For details, see Explore Cribl Edge.

Examples

Load Average

Imagine you are conducting an investigation and you need to determine which Edge Nodes have a higher-than-usual load average.

Let's go to the Dataset called `cribl_edge_metrics` and tell the search to return all the events that have the `node_load` term in them. And then let's ask it to build a time series of the load average of one minute (the default span) and split it by host:

```
dataset="cribl_edge_metrics" node_load*
| limit 1000
| timestats load1=max(node_load1) by host
```

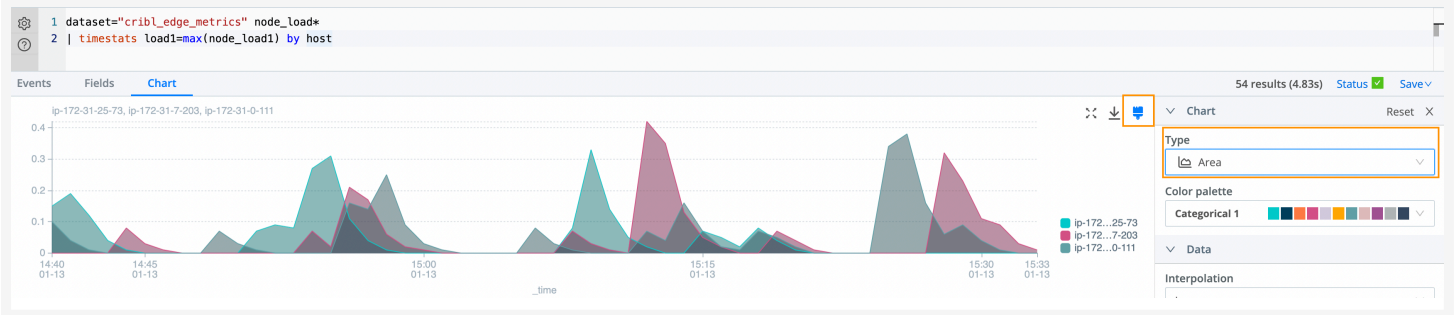
Copy | Run in Cribl Search



Load average by host

We haven't collected this data in advance, we haven't moved it or pre-processed or ETL it in any way. The search is executed in real-time, at rest, directly on the edge.

Let's change the charting option to **Area** to improve visibility:



Area Chart of the load average by host

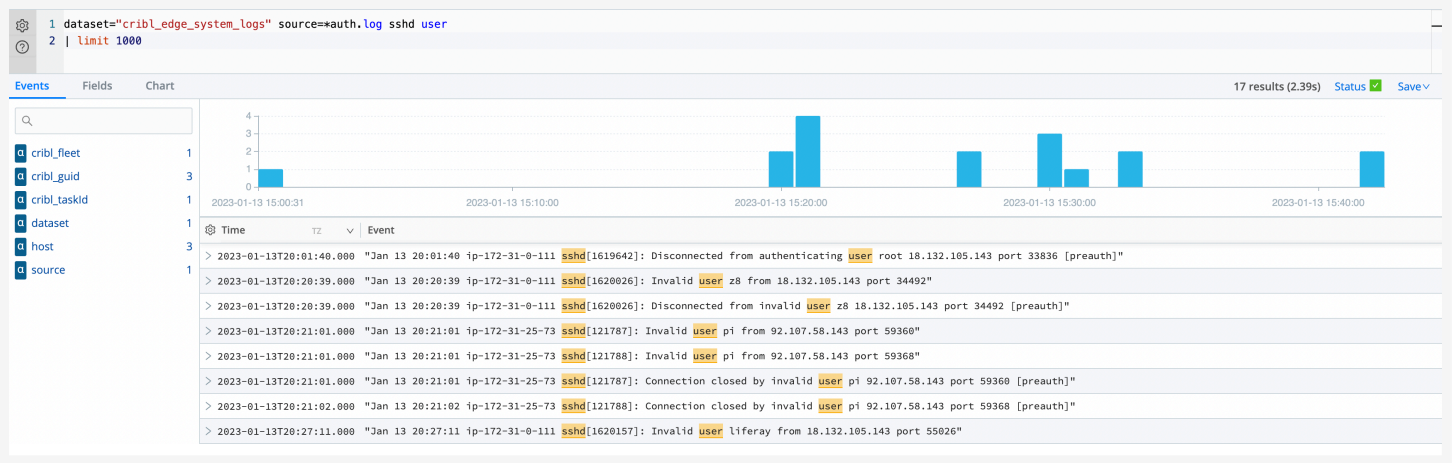
Cribl Search comes with rich charting options out of the box, allowing you to adjust Charts as needed.

System Logs

Now, let's check the `/var/log` directories containing the system logs of all Edge Nodes in the Fleet. We will query the built-in `cribl_edge_system_logs` Dataset and tell the search to get all the files that match `*auth.log` and have the terms `sshd` and `users`. We also set a limit of 1,000 results to return:

```
dataset="cribl_edge_system_logs" source=*auth.log sshd user
| limit 1000
```

Copy | Run in Cribl Search



System logs search

Cribl Search understands that these events are coming from Cribl Edge, and we can select the link and teleport directly to that Edge Node and see that file.



Open in Cribl Edge

Search Just a Single Fleet

You can easily filter Cribl Edge search results to a specific Fleet, using the built-in field `cribl_fleet`.

For example, the following query searches the `cribl_edge_system_logs` Dataset for events that:

- Match `*auth.log`.

- Contain the terms `sshd` and `users`.
- Come from a Fleet called `VermontGoatFarm`.

```
dataset="cribl_edge_system_logs" source=*auth.log sshd user cribl_fleet=VermontGoatFa
| limit 1000
```

Copy | Run in Cribl Search

3.7.3. Built-In Cribl Edge Datasets

Learn about built-in Cribl Edge Datasets available in Cribl Search.

Cribl Search ships with built-in Cribl Edge Datasets so you can quickly search internal logs and metrics for your Edge Nodes. Using these Datasets, you can find comprehensive information about an instance's status/health, inputs, outputs, Pipelines, Routes, Functions, and traffic. You can edit these built-in Datasets, or create new ones to specify other logs anywhere in the filesystem that Edge can read.

Cribl Edge comes out-of-the-box with these Datasets:

- `cribl_edge_logs`
- `cribl_edge_state`
- `cribl_edge_metrics`
- `cribl_edge_system_logs`
- `cribl_edge_spool`
- `cribl_edge_appscope_events`
- `cribl_edge_appscope metrics`

Details of Built-In Cribl Edge Datasets

Below are the type of data each Dataset returns, along with some event samples:

Dataset Name	Type of Data	Event Sample

cribl_edge_logs	Cribl Edge provides API server logs, Edge Node process(es), and Fleet logs. For details, see Types of Logs and Internal Logs.	{
		"dataset": "cribl_edge_logs",
		"source": "file:///opt/cribl/log/cribl.log
		"host": "1e7a817de3be",
		"_raw": "{\"time\":\"2023-04-19T10:19:15.5
		"time": "2023-04-19T10:19:15.559Z",
		"cid": "api",
		"channel": "MetricsStore",
		"level": "info",
		"message": "active metrics",
		"numMetrics": 37,
		"_time": 1681899555.559,
		"datatype": "cribl_json",
		"cribl_taskId": "1681903132338.sOY7mn",
		"cribl_guid": "a315e568-cecb-45de-a53e-207
		"cribl_fleet": "default_fleet"
		}

cribl_edge_state	System state provides a snapshot of the host system's current state, including details on host info, groups, and users. For details, see System State.	{
		"dataset": "cribl_edge_state",
		"source": "file:///opt/cribl/state/system_
		"host": "1e7a817de3be",
		"state_source": "userGroup",
		"_raw": "{\"_time\":1681902601.362,\"host\
		"_time": 1681902601.362,
		"name": "root",
		"groupId": 0,
		"users": ["root"],
		"datatype": "cribl_json",
		"cribl_taskId": "1681902737896.8EOlpP",
		"cribl_guid": "a315e568-cecb-45de-a53e-207
		"cribl_fleet": "default_fleet"
		}

Dataset Name	Type of Data	Event Sample

`cribl_edge_metrics`	Cribl Edge collects metrics from the host on which it is running, and can populate some standard metrics Dashboards right out of the box. For details, see Linux System Metrics Details.	{
		"dataset": "cribl_edge_metrics",
		"source": "file:///opt/cribl/state/system_
		"host": "1e7a817de3be",
		"metrics_source": "system",
		"_raw": "{\"_time\":1681902001.56,\"host\"
		"_time": 1681902001,
		"nodename": "1e7a817de3be",
		"release": "5.10.104-linuxkit",
		"sysname": "Linux",
		"version": "#1 SMP PREEMPT Thu Mar 17 17:0
		"node_uname_info": 1,
		"node_cpu_count": 5,
		"datatype": "cribl_json",
		"cribl_taskId": "1681902797516.BskPaQ",
		"cribl_guid": "a315e568-cecb-45de-a53e-207
		"cribl_fleet": "default_fleet"
		}

`cribl_edge_system_logs`	Search through Linux host's OS logs in `/var/log`. For details, see Internal Logs.	{
		"dataset": "cribl_edge_system_logs",
		"source": "file:///var/log/bootstrap.log",
		"host": "1e7a817de3be",
		"_raw": "gpgv: Signature made Thu Apr 23 1
		"_time": 1587663257,
		"datatype": "cribl_json",
		"cribl_taskId": "1681903034051.Q4gkk4",
		"cribl_guid": "a315e568-cecb-45de-a53e-207
		"cribl_fleet": "default_fleet"
		}

Dataset Name	Type of Data	Event Sample

`cribl_edge_spool`	Search through events stored in Cribl Edge's data spool. For details, see Disk Spool Destination).	{
		"_raw": "{\"_raw\":\"2024-01-31 11:40:12
		"_time": 1706730012.611,
		"cribl_breaker": "fallback",
		"cribl_fleet": "default_fleet",
		"dataset": "cribl_edge_spool",
		"host": "C4GQHK-zpF3",
		"source": "file:///cribl/dist/ributed/ma
		"data__raw": "2024-01-31 11:40:12.61168
		"data_source": "/private/var/log/com.app
		"datatype": "cribl_json",
		"output_id": "disk_spool"
		}

Dataset Name	Type of Data	Event Sample

`cribl_edge_appscope_events`	AppScope emits events for the applications being scoped. For details, see What Events Do.	{
		"_raw": "{\"type\":\"evt\",\"id\":\"ip-10-
		"source": "file:///home/ubuntu/.scope/hist
		"host": "ip-10-8-102-86",
		"type": "evt",
		"id": "ip-10-8-102-86-curl-curl wttr.in/Sa
		"_channel": "119018778058933",
		"body": {
		"sourcetype": "http",
		"_time": 1681767085.239414,
		"source": "http.req",
		"host": "ip-10-8-102-86",
		"proc": "curl",
		"cmd": "curl wttr.in/SanFrancisco",
		"pid": 653261,
		"data": {
		"http_method": "GET",
		"http_target": "/SanFrancisco",
		"http_flavor": "1.1",
		"http_scheme": "http",
		"http_host": "wttr.in",
		"http_user_agent": "curl/7.68.0",
		"net_transport": "IP.TCP",
		"net_peer_ip": "5.9.243.187",
		"net_peer_port": 80,
		"net_host_ip": "10.8.102.86",
		"net_host_port": 56622
		}
		},
		"_time": 1681767085.5878348,
		"datatype": "cribl_json",
		"cribl_fleet": "default_fleet"
		}

Dataset Name	Type of Data	Event Sample

`cribl_edge_appscope_metrics`	AppScope emits metrics for the applications being scoped. For details, see What Metrics Do.	{
		"_raw": "{\"type\":\"metric\",\"body\":{\"
		"source": "file:///home/ubuntu/.scope/hist
		"host": "ip-10-8-102-86",
		"type": "metric",
		"body": {
		"_metric": "http.req",
		"_metric_type": "counter",
		"_value": 1,
		"http_target": "/SanFrancisco",
		"http_status_code": 200,
		"proc": "curl",
		"pid": 653261,
		"host": "ip-10-8-102-86",
		"unit": "request",
		"summary": "true",
		"_time": 1681767085.589768
		},
		"_time": 1681767085.5878348,
		"datatype": "cribl_json",
		"cribl_fleet": "default_fleet"
		}

3.7.4. Searching Amazon S3

Learn how to search your Amazon S3 data.

Cribl Search supports searching Amazon S3 buckets. You need a Dataset Provider and a Dataset, see the setup guide for Amazon S3 and Data Lake Amazon S3 for details.

This document provides examples of searching objects from Amazon S3. To start, navigate to **Search Home**, where searches are run.

Examples

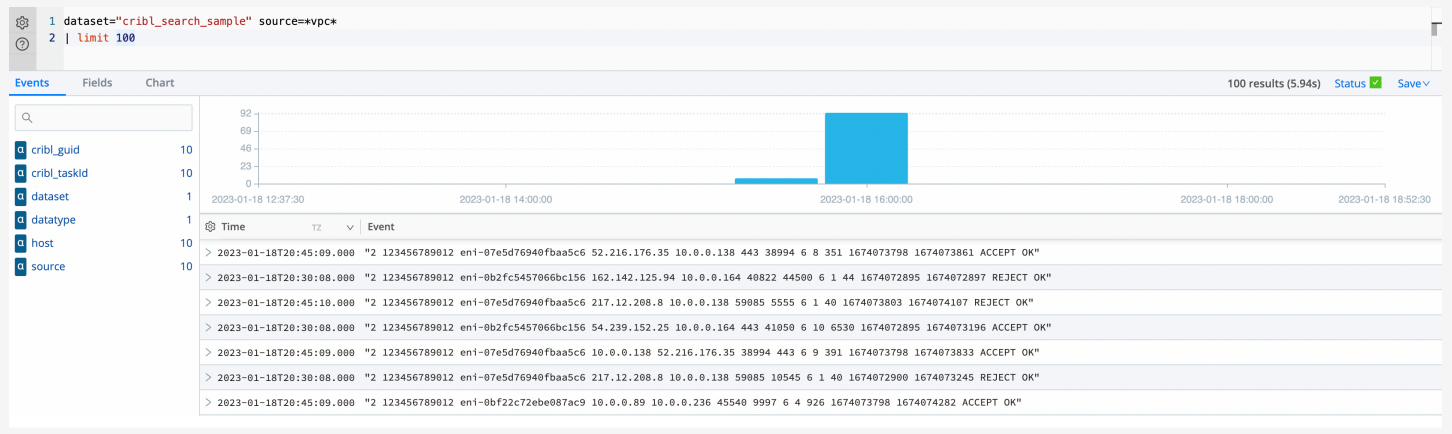
The following examples reference a built-in Dataset with the **ID** `cribl_search_sample`.

Basic Search

This search specifies the Dataset and limits the number of results.

```
dataset="cribl_search_sample" source=*vpc*
| limit 100
```

Copy | Run in Cribl Search



Basic search

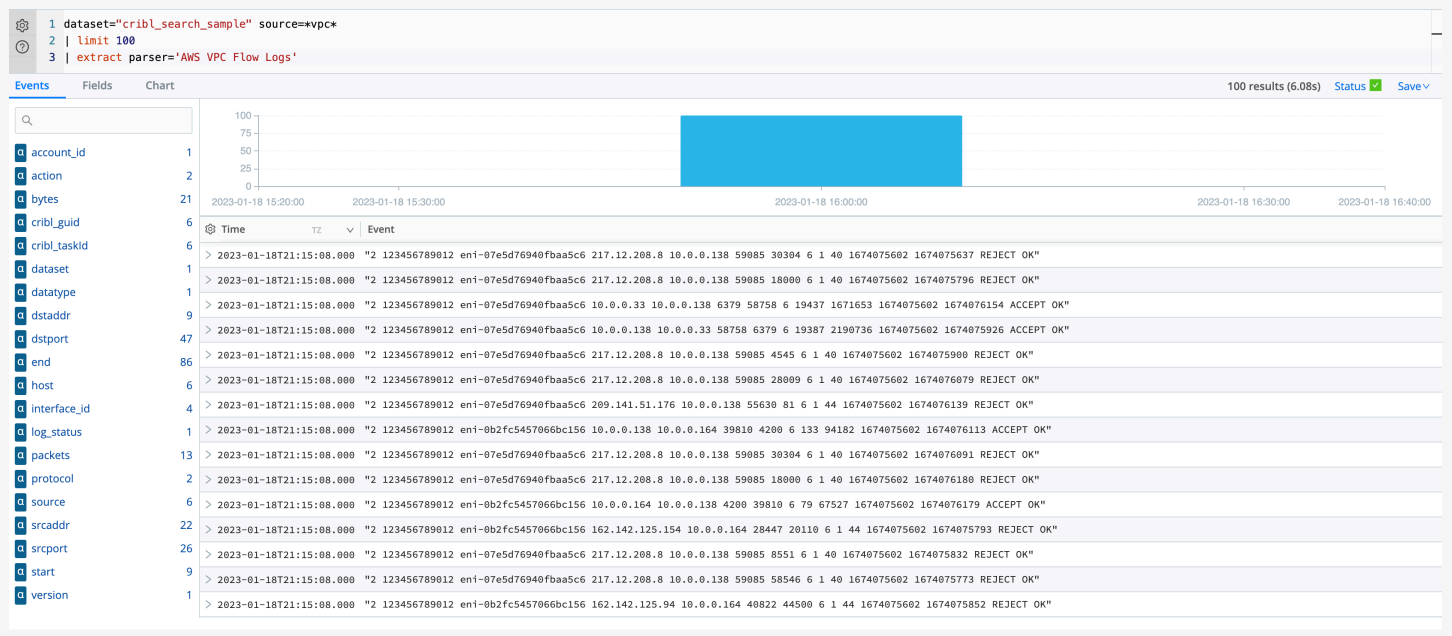
Built-in Parser

Cribl Search comes with parsing libraries so you don't have to write them manually. Use the `extract` operator to reference the built-in Parser for VPC Flow Logs.

```
dataset="cribl_search_sample" source=*vpc*
| limit 100
| extract parser='AWS VPC Flow Logs'
```

Copy | Run in Cribl Search

Note the additional fields shown in the field browser.



Parser search

If you want to see the built-in parser's effects, first run the `extract` command below. Then add the `extract` command:

```
dataset="$vt_dummy"
 | extend _raw = "2 602320997947 eni-5nxjlyb0xm0o96kgs 52.216.176.35 10.0.0.138 1311

// Run the above portion first to see the logs unparsed. Add the extract operator bel
 | extract parser='AWS VPC Flow Logs'
```

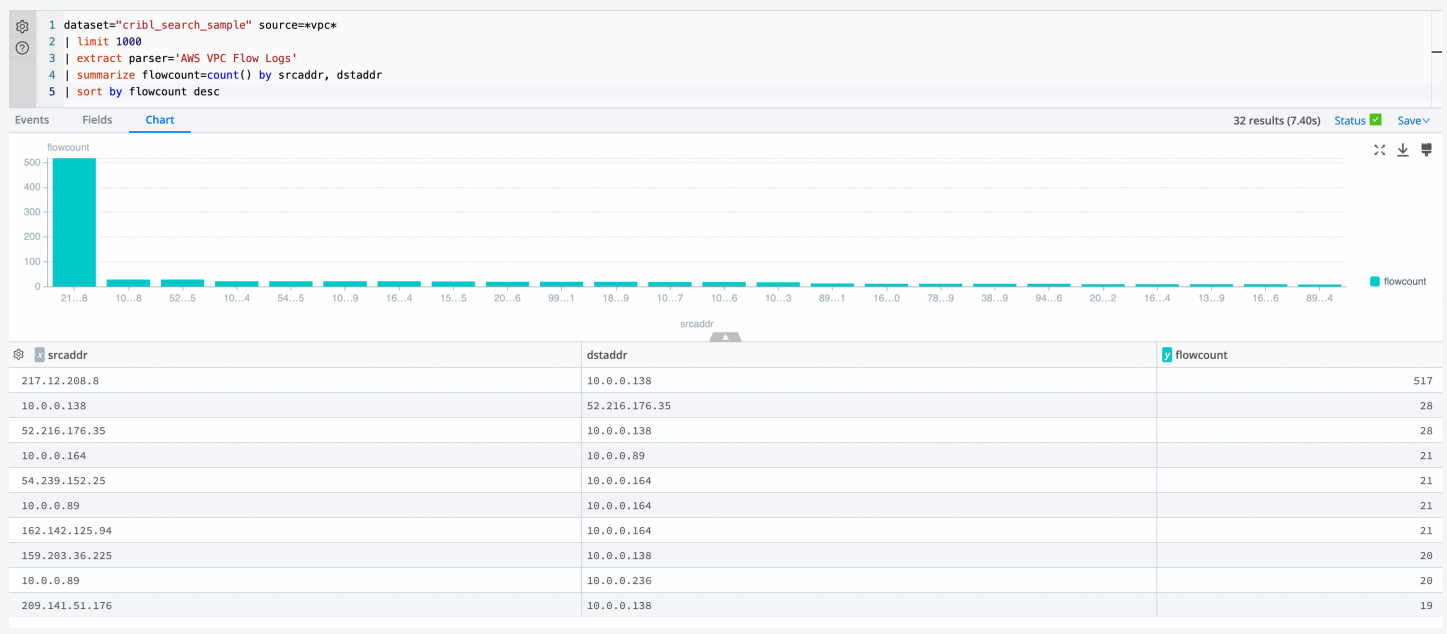
Copy | Run in Cribl Search

Aggregate

To aggregate, we'll use the `summarize` operator with the count function grouping by IP address pairs.

```
dataset="cribl_search_sample" source=*vpc*
| limit 1000
| extract parser='AWS VPC Flow Logs'
| summarize flowcount=count() by srcaddr, dstaddr
| sort by flowcount desc
```

Copy | Run in Cribl Search



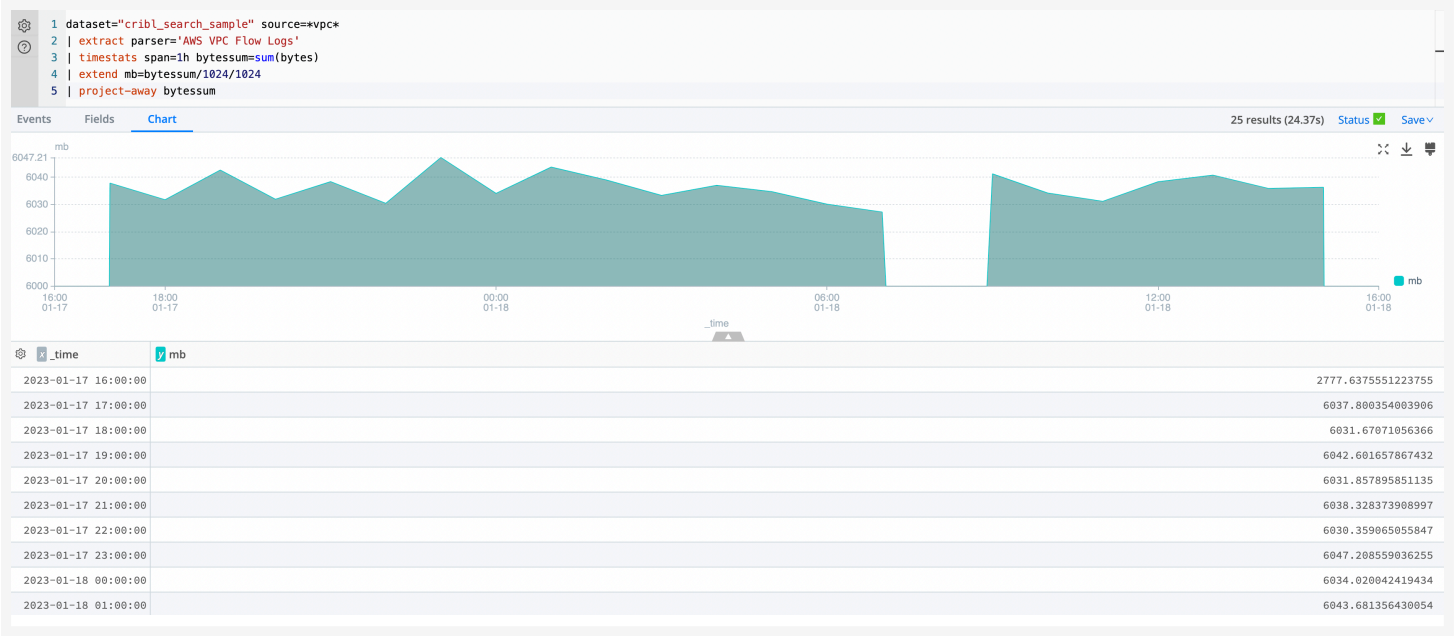
Aggregating search

Aggregate over time

To aggregate over time, we'll use the `timestats` operator.

```
dataset="cribl_search_sample" source=*vpc*
| limit 1000
| extract parser='AWS VPC Flow Logs'
| timestats span=1h bytessum=sum(bytes)
| extend mb=bytessum/1024/1024
| project-away bytessum
```

Copy | Run in Cribl Search



Aggregating over time search

Cribl Search comes with rich charting options out of the box, allowing you to adjust Charts as needed. The above example is an Area Chart with a Y-axis minimum of 6,000.

3.7.5. Searching AWS API

Learn how to search your AWS data.

Cribl Search supports searching AWS API endpoints. You need a Dataset Provider and a Dataset, see the setup guide for AWS API for details.

This document provides examples of searching endpoints from the AWS API. To start, navigate to **Search Home**, where searches are run.

Account and Endpoint Identifiers

When you search the AWS API, events in the results will contain `accountName` and `endpointName` fields, which reflect the account and endpoint configured on the Dataset.

If your Dataset has multiple accounts and/or endpoints enabled, these fields distinguish each event's source. They also serve as potential filters that you can express in predicates, as you'll see in examples below.

Examples

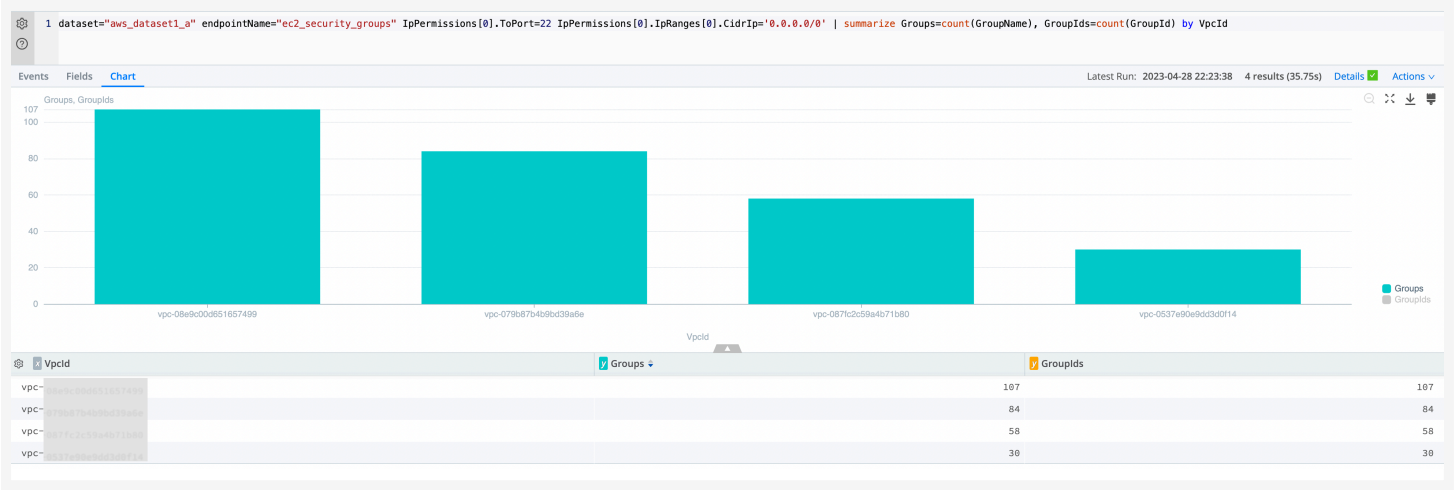
The following examples reference a Dataset with the **ID** `aws_dataset1_a`.

EC2 Security Groups API Search

This search interrogates the `ec1_security_groups` endpoint for number of Security Groups that allow inbound to port `22` from `0.0.0.0/0`, grouped by `VPCID`?

```
dataset="aws_dataset1_a" endpointName="ec2_security_groups" IpPermissions[0].ToPort=2
| summarize Groups= count(GroupName), GroupsIds=count(GroupId) by VpcId
```

Copy



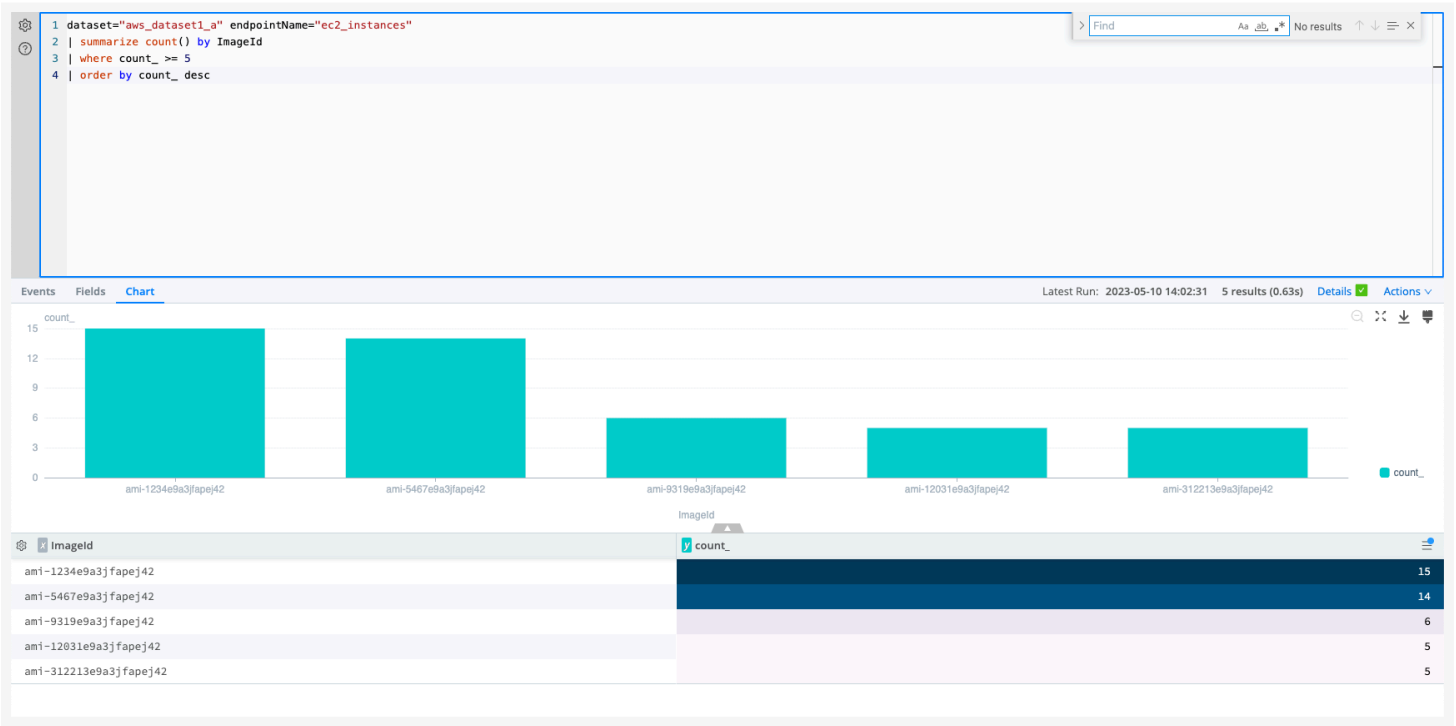
Sample AWS Search: EC2 Security Groups

EC2 Instances API Search

This search hits the `ec2_instances` API endpoints to find out top 5 most commonly used Amazon Machine Images (AMIs) and summarizes the results by `ImageId`.

```
dataset="aws_dataset1_a" endpointName="ec2_instances"
| summarize count () by ImageId
| where count >= 5
| order by count_desc
```

Copy



Cribl Search comes with rich charting options out of the box, allowing you to adjust Charts as needed.

3.7.6. Searching Okta API

Learn how to search Okta API endpoints.

Cribl Search supports searching Okta API endpoints. You need a Dataset Provider and a Dataset, see the setup guide for Okta API for details.

This document provides examples of searching endpoints from the Okta API. To start, navigate to **Search Home**, where searches are run.

Examples

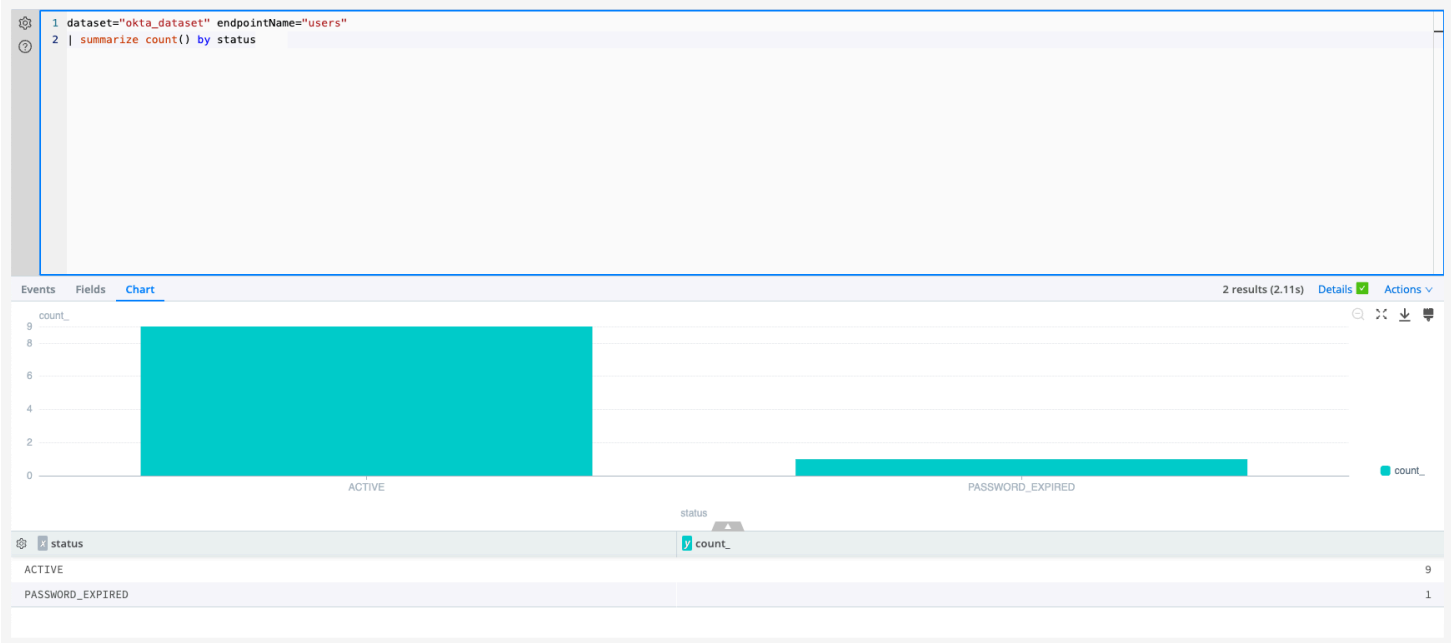
The following examples reference a Dataset with the **ID** `okta_dataset`.

Users API Search

This search interrogates the `users` endpoint for active and expired passwords.

```
dataset="okta_dataset" endpointName="users"
| summarize count () by status
```

Copy



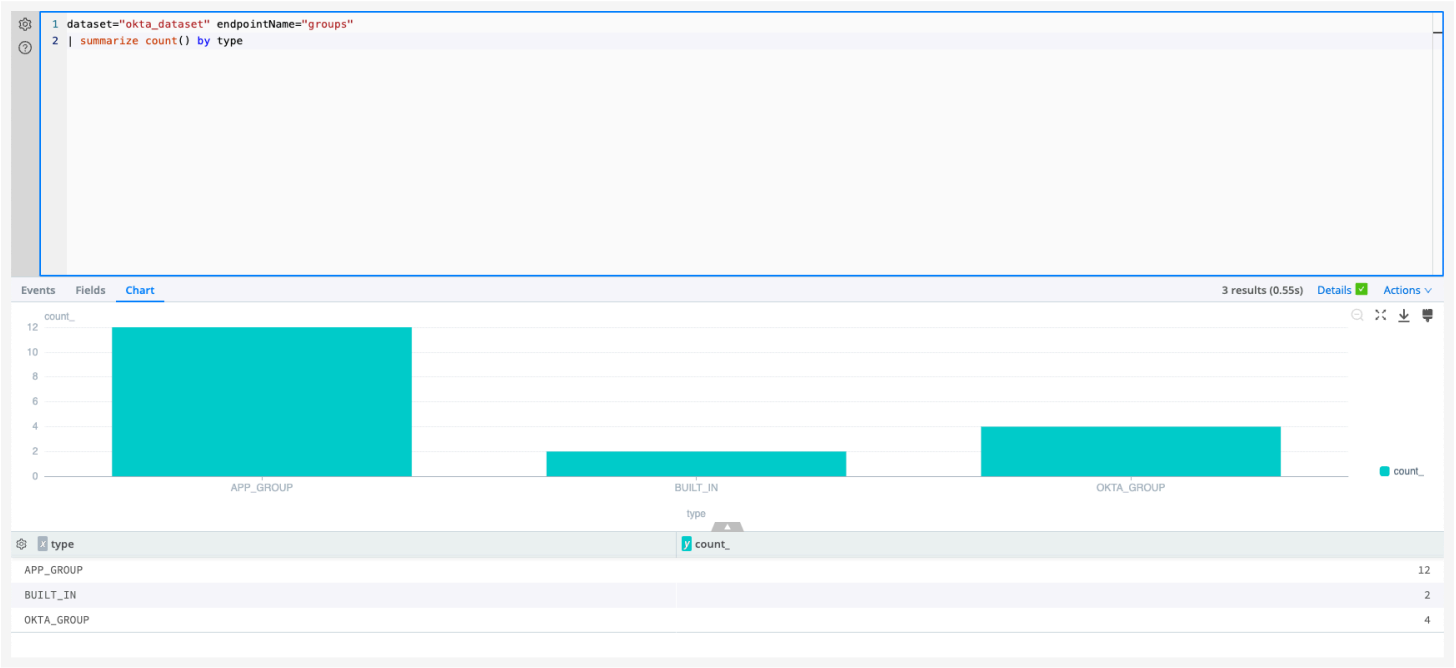
Sample Okta Search: User status

Groups API Search

This search hits the `groups` API endpoints and summarizes the results by type.

```
dataset="okta_dataset" endpointName="groups"
| summarize count () by type
```

Copy



Sample Okta Search: Groups by Type

Cribl Search comes with rich charting options out of the box, allowing you to adjust Charts as needed.

3.7.7. Searching Zoom API

Learn how to search Zoom API endpoints.

Cribl Search supports searching Zoom API endpoints. You need a Dataset Provider and a Dataset, see the setup guide for Zoom API for details.

This document provides examples of searching endpoints from the Zoom API. To start, navigate to **Search Home**, where searches are run.

Examples

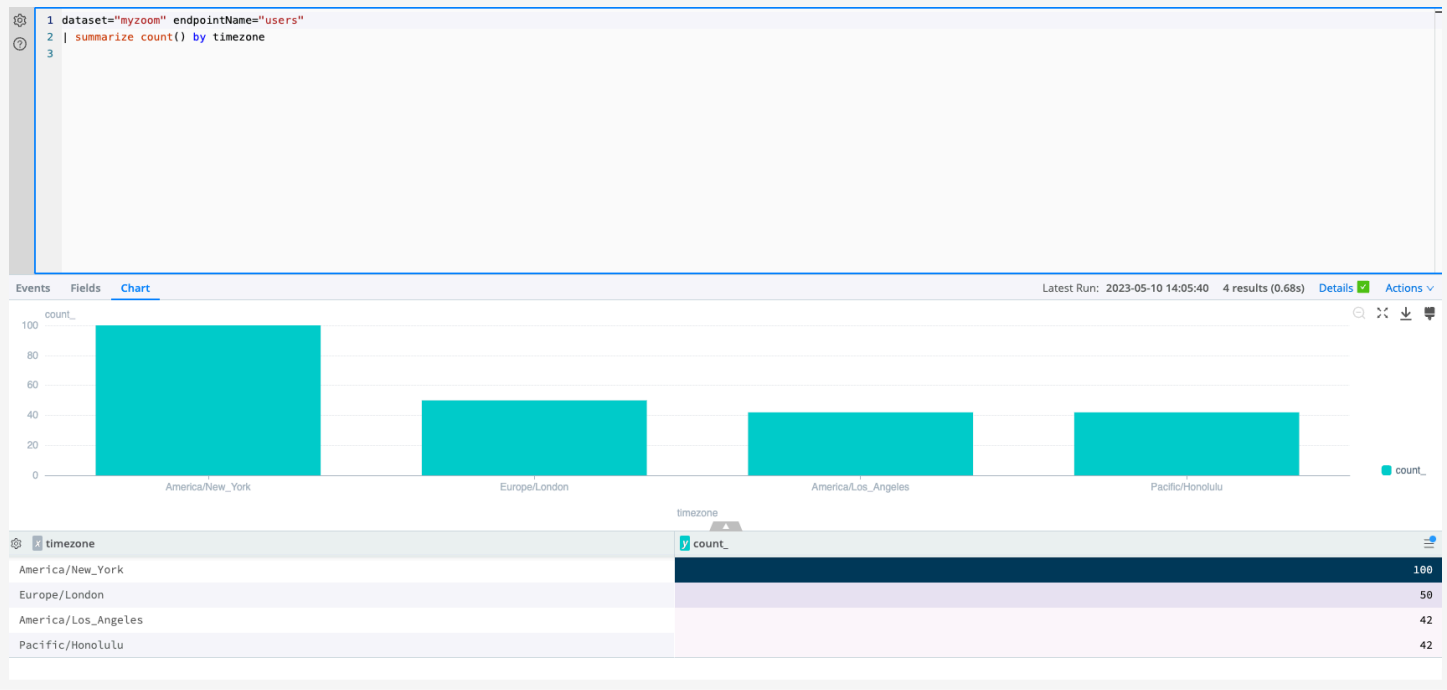
The following examples reference a Dataset with the **ID** `myzoom`.

User API Search

This search specifies the Dataset and aggregates the results by timezone.

```
dataset="myzoom" endpointName="users"
| summarize count () by timezone
```

Copy



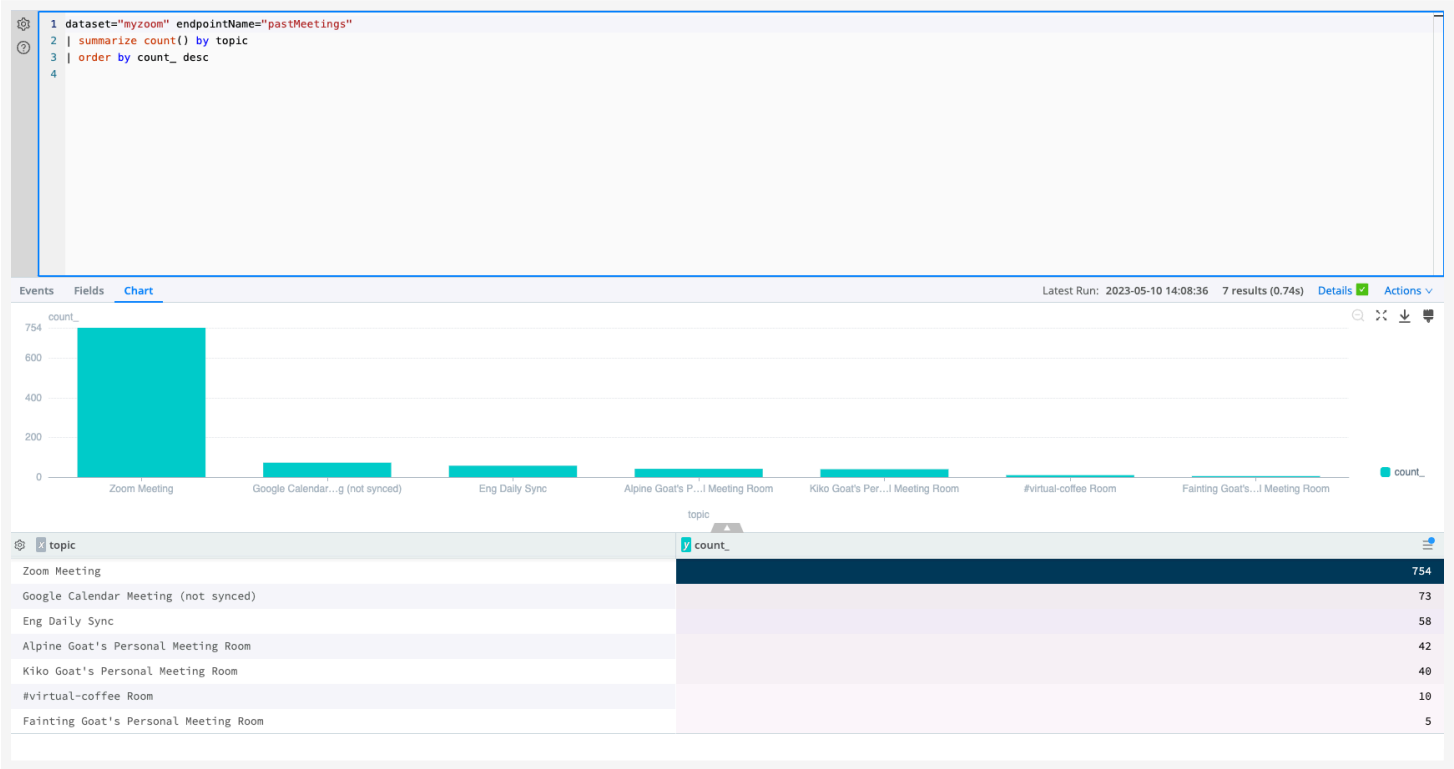
Sample Zoom Search: Aggregate by Timezone

Meeting API Search

This search hits the `pastMeetings` API endpoints and summarizes the results by topic.

```
dataset="zoom_cribl1" endpointName="pastMeetings"
| summarize count () by topic
| order by count_desc
```

Copy



Sample Zoom Search: Past Meetings by Topic

Cribl Search comes with rich charting options out of the box, allowing you to adjust Charts as needed.

3.7.8. Searching Multiple Providers

Set up and search multiple Dataset Providers.

Cribl Search supports searching across multiple providers. In this example, we will illustrate how to configure and search logs across all three major object stores (Amazon S3, Azure Blob, and Google Cloud Storage).

Why would you want to search multiple providers, you might ask? Here are a few reasons:

- Investigate the same data across multiple providers more quickly (for example, by flowing logs across all three main providers).
- Ensure conformance across all object stores - Eventually you're going to have to search data across multiple object stores; to make sure they all follow the same standard, use Cribl Search to run conformance-based searches.
- Give users access to the data - Analytics systems can be expensive. What if those infrequent or ad hoc users could access your data through Cribl Search without incurring additional downstream licensing or cloud egress costs?
- Analyze historical data before replaying to your analysis system - We often need to review historical data for investigations, capacity exercises, or other reasons. Instead of replaying or rehydrating all the data for a given time, Cribl Search allows you to find the relevant data PRIOR to rehydrating.
- And more!

So now that you know searching across multiple providers can be useful, let's connect your object stores and start searching.

Connecting the Providers

This example searches Virtual Private Cloud (VPC) flow logs stored in Amazon S3, Azure Blob, and Google Cloud Storage.

Configuring and Routing VPC Flow Logs

VPC flow logs enable you to capture information about the IP traffic going to and from network interfaces in your VPC. For details on how you can configure the flow logs to be routed to each provider's storage object, see:

- Amazon S3: Publish flow logs to Amazon S3
- Azure Blob: Create a flow log and traffic analytics workspace
- Google Cloud Storage: Route logs with the Log Router

To process your VPC flow logs faster, use the **AWS VPC Flow Log** event breaker.

Configuring Cribl Search Data Providers

Once you have flow logs available in the respective object stores, you can set up data providers in Cribl Search to access the logs.

To give all three Datasets a uniform naming convention, append each Dataset **ID** with `_flowlogs`. This option allows us to use a wildcard in the query. For details, see the section below on Running a Search Across Multiple Dataset Providers.

Additionally, you can combine data from different Datasets in a single query by specifying each Dataset's ID. For example: `dataset=myAWSFlowlogs or dataset=myGCSFlowlogs or dataset=myAzureFlowlogs`. For details, see Syntax.

Amazon S3

Let's get started by setting up your Amazon S3 Dataset Provider and Dataset in Cribl Search.

Google Cloud Storage (GCS)

In Cribl Search, create a Google Cloud Storage Dataset Provider and Dataset.

Azure Blob

In Cribl Search, create an Azure Blob Dataset Provider and Dataset.

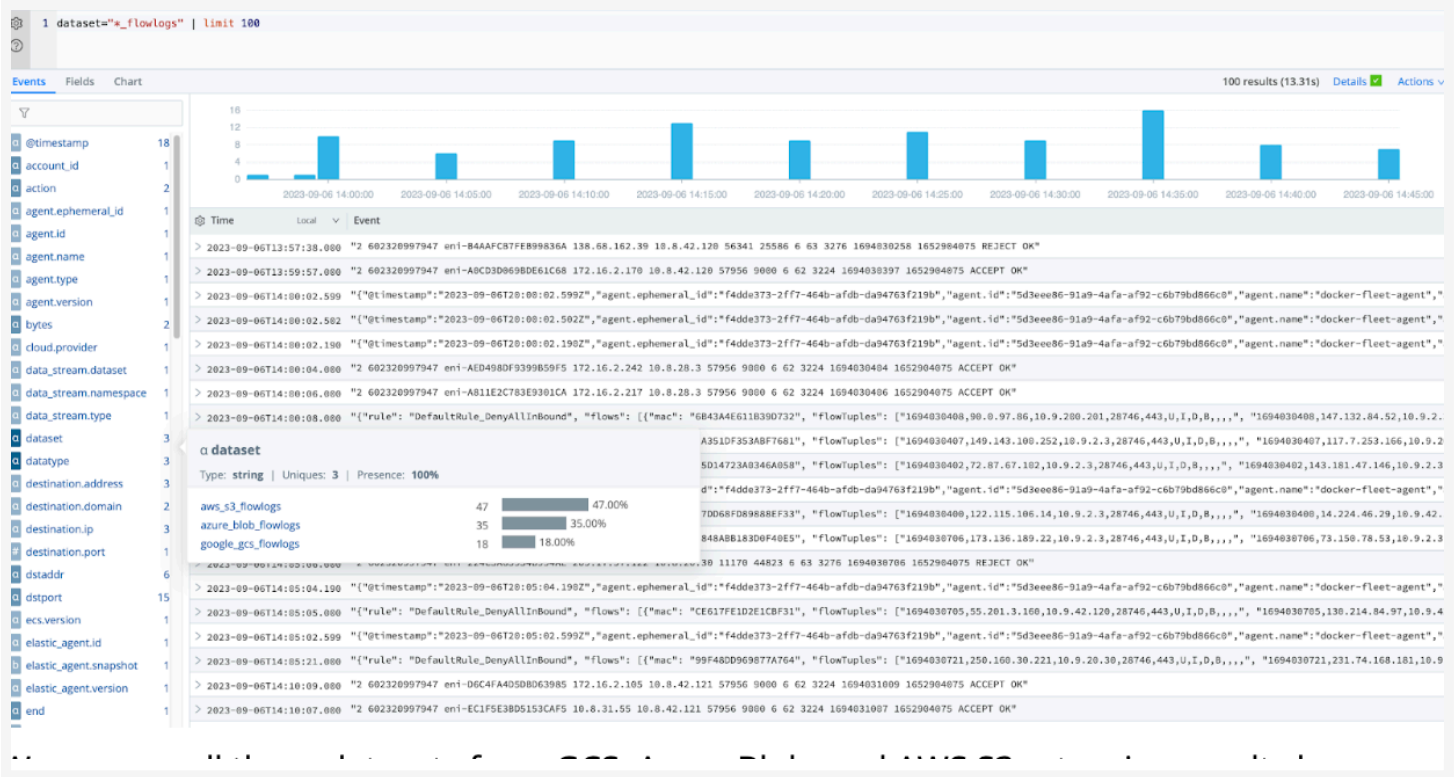
Running a Search Across Multiple Dataset Providers

We now have all three providers configured within Cribl Search. It's time to run some searches.

Since we appended the Dataset IDs with `_flowlogs`, this allows us to run the first search and combine data from all three Datasets:

```
dataset="*_flowlogs"
| limit 1000
```

Copy



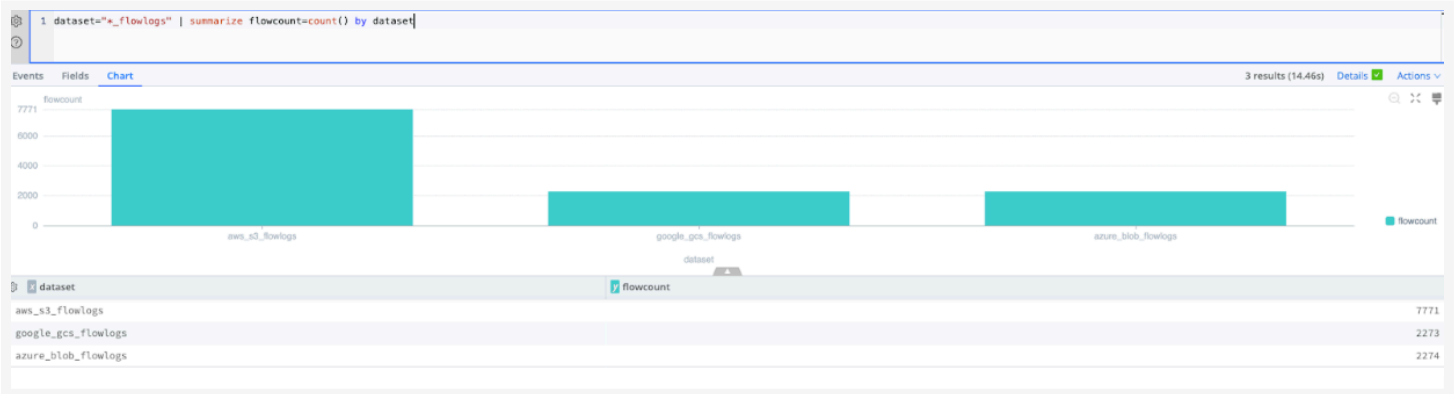
Combined data across Datasets

We can see all three Datasets from GCS, Azure Blob, and Amazon S3 returning results here. With Cribl Search, you can easily gather all your flow logs from your multi-cloud environments for analysis.

Let's take that search and turn it into a Bar Chart with counts of each Dataset's flow log. Flow log volumes can be visualized for each provider to identify anomalies.

```
dataset="*_flowlogs"
| summarize flowcount=count() by Dataset
```

Copy



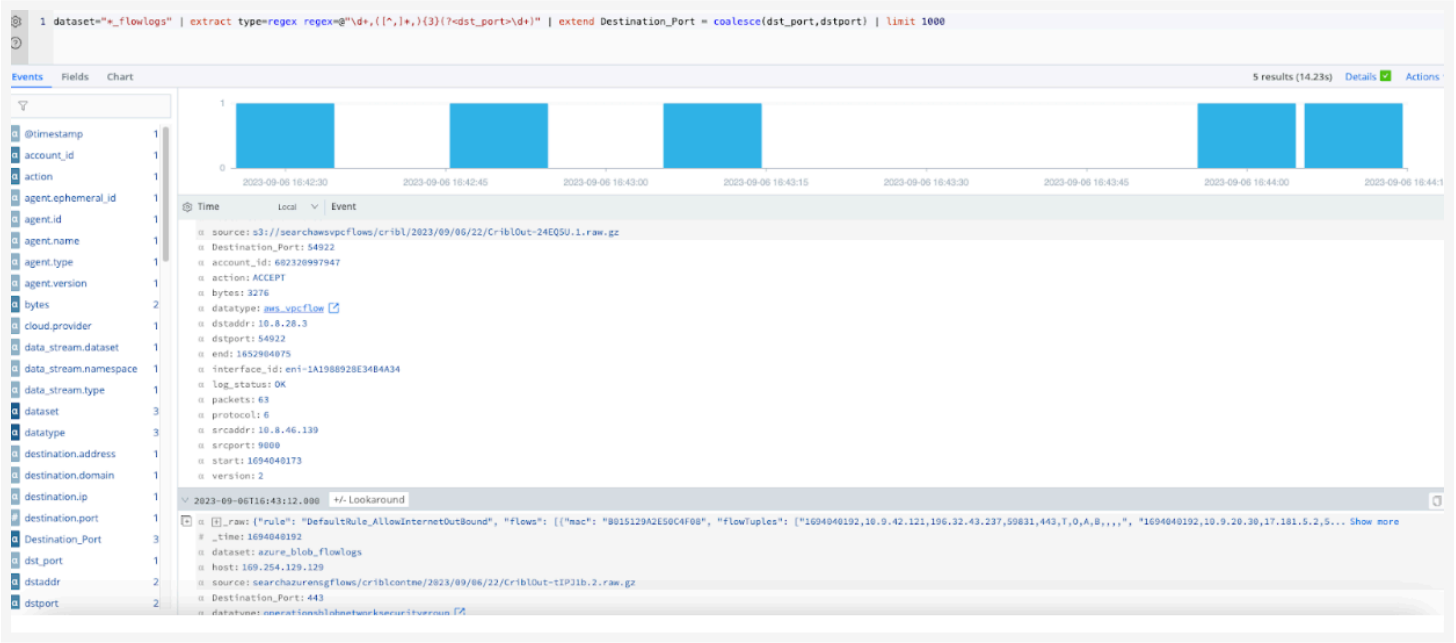
Flow logs count across Datasets

It's likely that the three providers will also share fields. They are, however, not normalized. For our Amazon S3 flow logs, we already have a `dstport`; however, Azure organizes flow logs into `tuples` from which we must extract the destination port.

As part of the next search, we try to accomplish just that, and then go one step further by using the coalesce function to create a normalized field, `Destination_Port`, across our Azure and Amazon S3 flows. It's also possible to create Datatypes with normalized fields for each Dataset. For details, see Datatypes. Now that we've normalized fields, we can analyze data across all three providers agnostically.

```
dataset="*_flowlogs"
| limit 1000
| extract type=regex regex=@"\d+,([^,]*,){3}(?<dst_port>\d+)"
| extend Destination_Port= coalesce(dst_port, dstport)
```

Copy



Flow logs across all providers

Conclusion

Multi-cloud environments and object stores continue to grow in popularity, and Cribl Search allows you to bring all of those Datasets together, normalize them, and generate insights across all platforms at once rather than keeping them separated.

Last updated by: Dritan Bitincka

4. Visualize Search Results

Visualize your Cribl Search results with Charts, Tables, Dashboards, and other visualization tools.

Charts

When you run an aggregate search, your results display on a Chart. Learn how to use Charts in the following sections:

- Charts: Learn the basics, export a Chart as an image, save and reset Chart settings.
- Chart Types: See all available types of Charts, and learn how to customize each of them.
- Tables: Learn how to format the results table that displays below every Chart.

Dashboards

Cribl Search offers customizable Dashboards with a variety of visualization panels that you can tailor to your needs. Learn how to use Dashboards in the following sections:

- Dashboards: Get an overview and learn about built-in sample Dashboards.
- Create a Dashboard: Learn different ways of creating a Dashboard, and how to fill it with visualizations.
- Edit a Dashboard: Add, edit, hide, and arrange the visualization panels of your Dashboard.
- Apply a Parent Search: Aggregate results from multiple searches into a single visualization.
- Add Inputs: Link visualizations to specific data sources, using inputs.
- Add Interactions: Enable Dashboard viewers to drill down into specific values in a visualization panel.
- Add Markdown: Enrich your Dashboards with custom Markdown panels.
- Refresh Your Dashboards Automatically: Run a Dashboard on schedule, or configure its caching.
- Add Visualizations Using Cribl Copilot: Use the Cribl Copilot AI assistant to generate Dashboard visualizations.
- Manage and Share Dashboards: Collect, export, and share your Dashboards.

Third-Party Visualization Tools

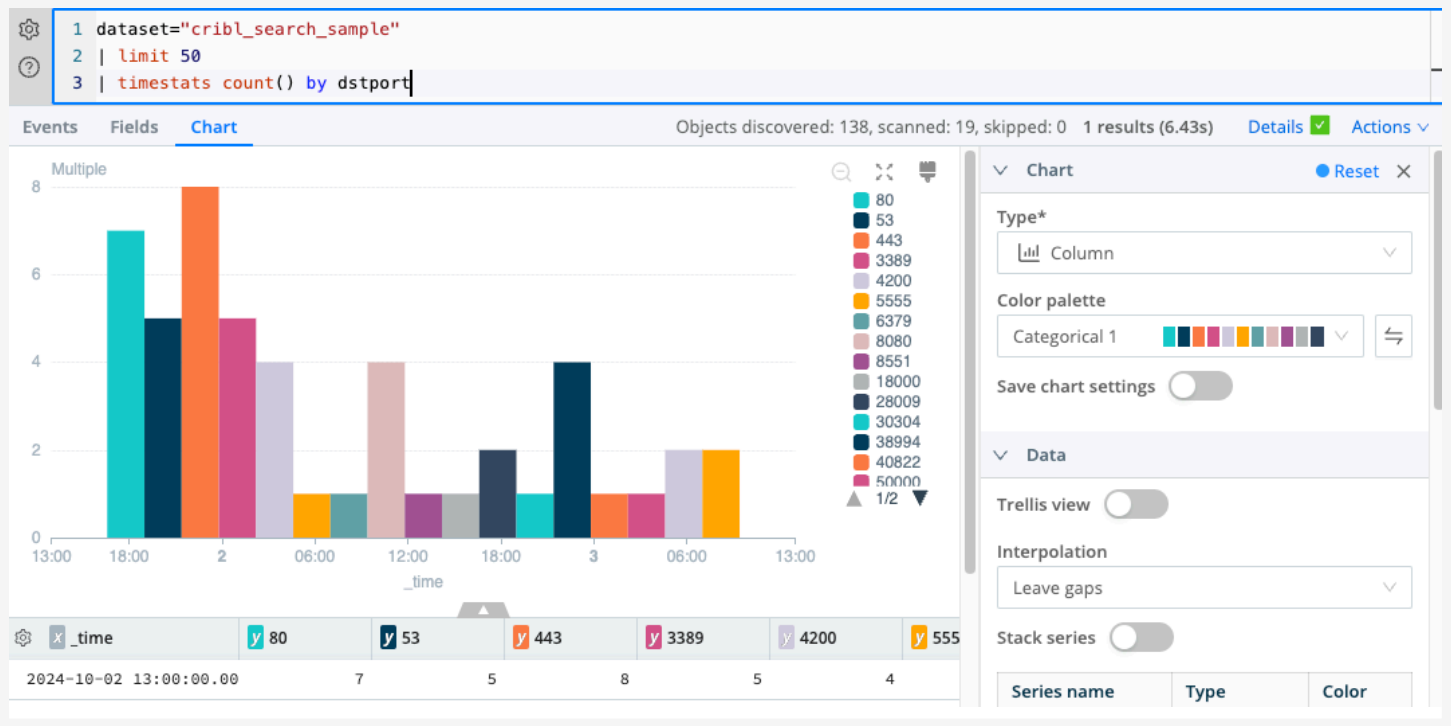
You can easily visualize your Cribl Search results in Grafana, using the official Cribl Search plugin. See: Use Cribl Search with Grafana.

4.1. Charts

Visualize your search results, using Charts.

Cribl Search provides Charts to display aggregate search results. Cribl and statistical functions are used together with the `summarize`, `eventstats`, and `timestats` operators to aggregate your data.

When you run an aggregate search, your results display on a Chart. You can select from various Chart types and color palettes, and you can manipulate how your results are plotted.



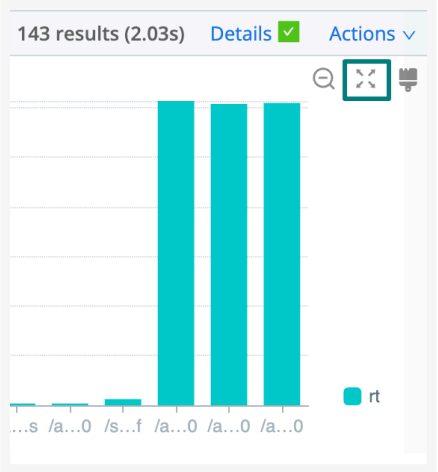
Aggregate results on a Column Chart

See the following topics:

- Types of Charts in Cribl Search – Each page explains an available Chart type, the types of data it's best used for, and the settings available.
- Results Tables – covers the available options for the results table associated with a Chart, including sorting, filtering, and formatting tabular results.

See a Chart in Full Screen

To zoom your Chart up to a larger view, select the expand (Full screen) button at the Chart's top right.

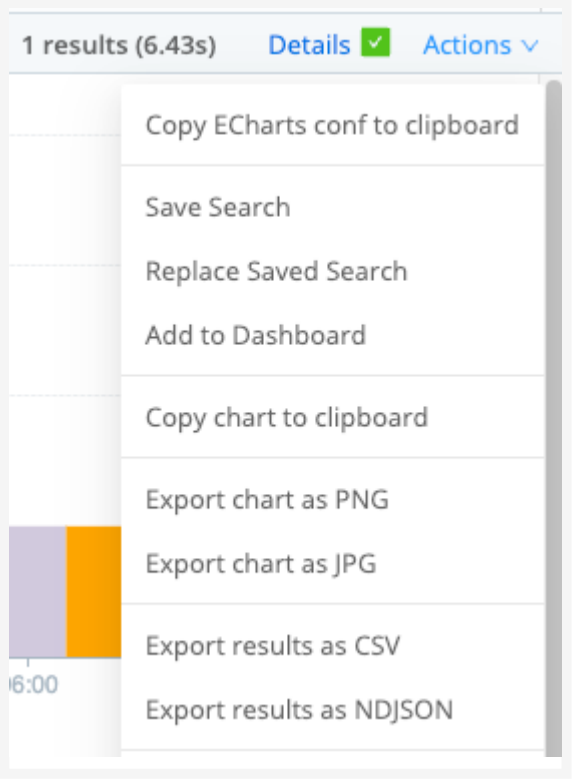


Full screen

Export a Chart as an Image

To export your search results or Chart, open the **Actions** drop-down and select one of the following:

- **Export chart as PNG**.
- **Export chart as JPG**.
- **Export results as CSV**.
- **Export results as JSON**.



Export aggregate results

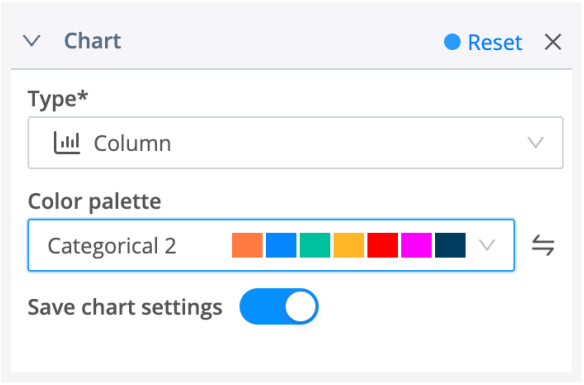
Save or Reset Chart Settings

The **Chart** tab provides a **Save chart settings** option (off by default). Enable this if you want to persist your Chart customizations when running later queries on the same page. If the Chart re-renders in unexpected ways (because some custiomizations conflict with new query terms), try turning this back off.

Selecting the **Reset** link will restore Chart defaults. This includes restoring the default Chart type (Column) and turning off the **Save chart settings** setting itself.



On Dashboards, visualizations do not provide this **Save** option. This is to maintain Dashboards' dynamic nature.



Save and Reset controls

4.1.1. Types Of Charts In Cribl Search

Learn what types of Charts Cribl Search offers.

List of Available Chart Types

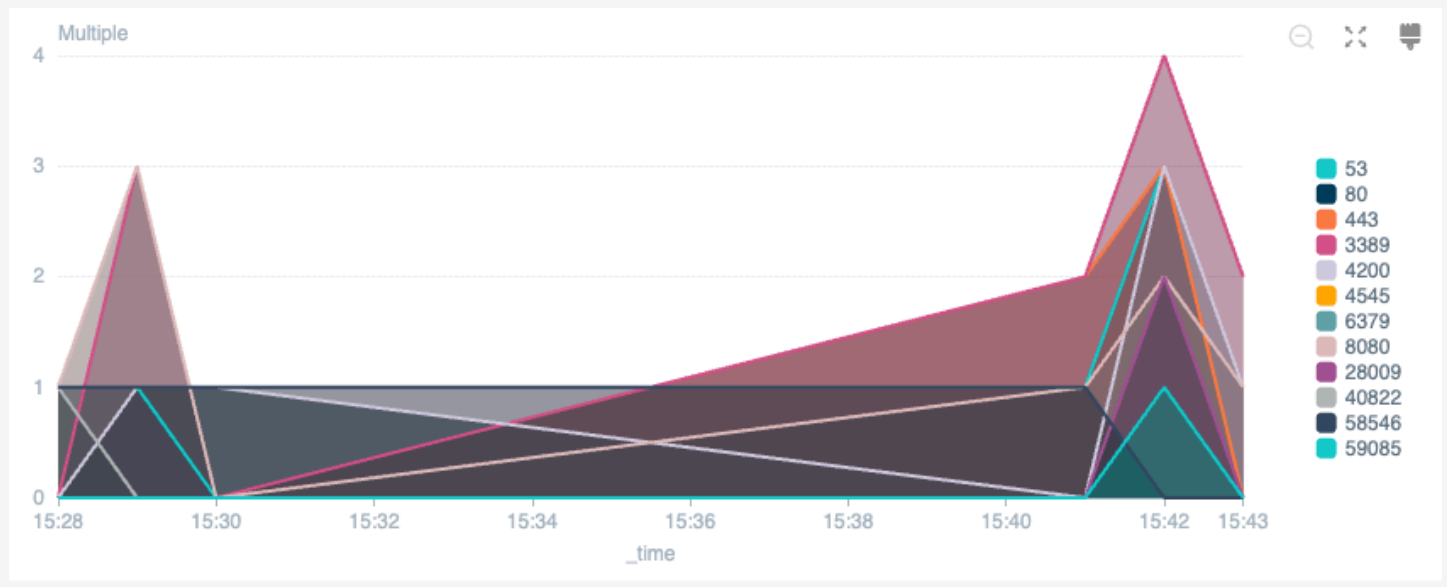
Cribl Search provides the following types of Charts. To learn more about each of them, follow the relevant link.

- Area Chart
- Column Chart
- Donut Chart
- Funnel Chart
- Gauge Chart
- Horizontal Bar Chart
- Line Chart
- Map Chart
- Scatter Chart
- Single Value Chart

4.1.1.1. Area Chart

Visualize your search results, using an Area Chart.

An **Area** Chart is similar to a Line Chart, except the area representing values is filled in with the selected color. These are suited for data that accumulates, not so much for highly fluctuating data.



Area Chart

Chart Settings

Color palette: Select the color scheme you want to use in your Chart.

Data Settings

Trellis view: Enable to divide the Chart into subsets of data based on fields in your search. Select a field from the **Split view by** list (available only when your search query includes two or more fields.)

Interpolation: Select **Connect null values** to display null values. The default **Leave gaps** option hides null values.

Stack series: When enabled, data in the same category are stacked together. (This option is only configurable for Bar and Area Charts.)

Series table: You can edit the name of a series in the **Series name** column. The **Type** column allows you to set a different Chart type for each series. Use the **Color** column to set the desired color for the series.

X-Axis Settings

X-axis field: The data field to use for the X axis of the Chart. Defaults to a field chosen automatically based on the selected search.

Title: The label shown on the x-axis. Defaults to the field name that the axis references. Enter a custom name if needed.

Label orientation: The angle of rotation to display additional labels. Defaults to **0°**.

Position: Where to display the axis labels on the Chart.

Y-Axis Settings

Y-axis fields: The data fields to use for the Y axis of the chart. Defaults to fields chosen automatically based on the selected search.

Title: Defaults to the field name the axis references. Enter a custom name if needed.

Position: Where to display the axis labels on the chart. Defaults to **Left**.

Scale: Sets the distance between results. The default **Linear** scale uses an equal rate of change. A **Logarithmic** scale uses a percentage change.

Interval: The size of the axis measurement unit. For example, each interval on the y-axis can represent 2 or even 10 units, while each interval on the x-axis represents 1 unit.

Min: The minimum limit of units to display.

Max: The maximum limit of units to display. Defaults to `dataMax` where the maximum unit of data is used.

Show gridlines: Disable to hide the horizontal lines on the chart.

Legend Settings

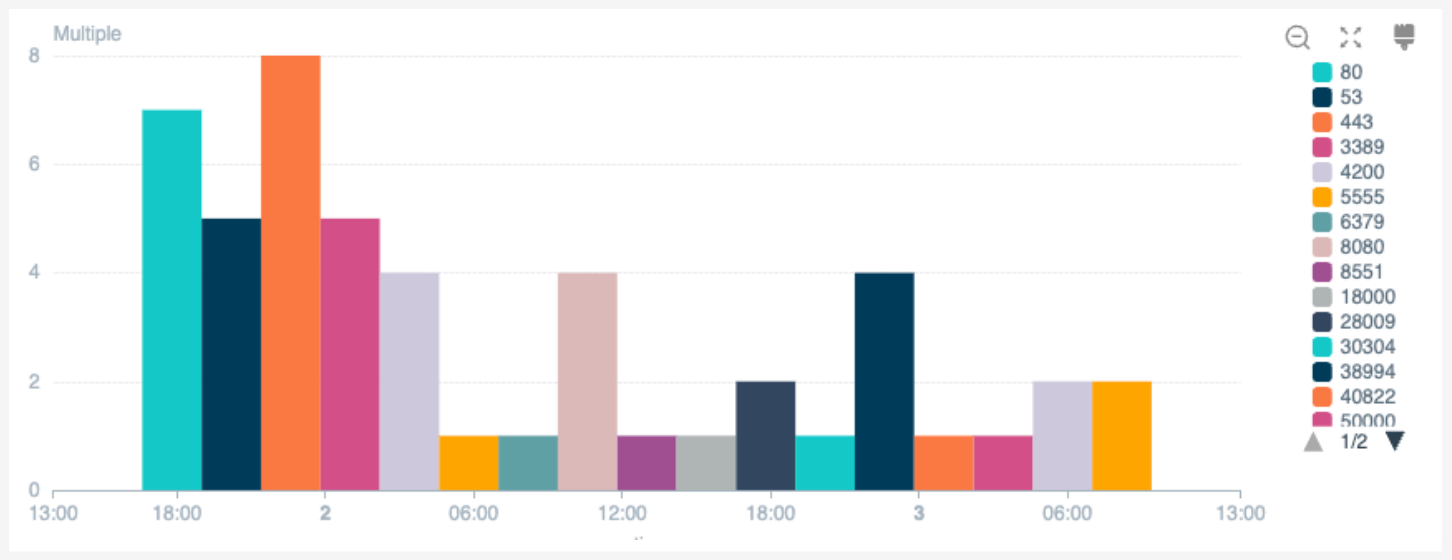
Position: Where to display the legend relative to the chart.

Truncate: By default, Cribl Search truncates labels to fit your browser viewport, substituting a `...` for omitted characters. Toggle this off if you prefer to not alter labels.

4.1.1.2. Column Chart

Visualize your search results, using a Column Chart.

A **Column** Chart is best used to visualize categories of data. You could use a **Column** Chart to compare a distribution of values over time, to monitor the number of CPUs used by machine type, or to visualze the number of requests handled by a Pod.



Column Chart

Column is Cribl Search's default Chart type. (This type was labeled **Bar** in earlier Cribl Search versions.) To switch the vertical orientation, select the Horizontal Bar Chart type.

Chart Settings

Color palette: Select the color scheme you want to use in your Chart.

Data Settings

Trellis view: Enable to divide the Chart into subsets of data based on fields in your search. Select a field from the **Split view by** list (available only when your search query includes two or more fields.)

Interpolation: Select **Connect null values** to display null values. The default **Leave gaps** option hides null values.

Stack series: When enabled, data in the same category are stacked together. (This option is only configurable for Bar and Area Charts.)

Series table: You can edit the name of a series in the **Series name** column. The **Type** column allows you to set a different Chart type for each series. Use the **Color** column to set the desired color for the series.

X-Axis Settings

X-axis field: The data field to use for the X axis of the Chart. Defaults to a field chosen automatically based on the selected search.

Title: The label shown on the x-axis. Defaults to the field name that the axis references. Enter a custom name if needed.

Label orientation: The angle of rotation to display additional labels. Defaults to **0°**.

Position: Where to display the axis labels on the Chart.

Y-Axis Settings

Y-axis fields: The data fields to use for the Y axis of the chart. Defaults to fields chosen automatically based on the selected search.

Title: Defaults to the field name the axis references. Enter a custom name if needed.

Position: Where to display the axis labels on the chart. Defaults to **Left**.

Scale: Sets the distance between results. The default **Linear** scale uses an equal rate of change. A **Logarithmic** scale uses a percentage change.

Interval: The size of the axis measurement unit. For example, each interval on the y-axis can represent 2 or even 10 units, while each interval on the x-axis represents 1 unit.

Min: The minimum limit of units to display.

Max: The maximum limit of units to display. Defaults to `dataMax` where the maximum unit of data is used.

Show gridlines: Disable to hide the horizontal lines on the chart.

Legend Settings

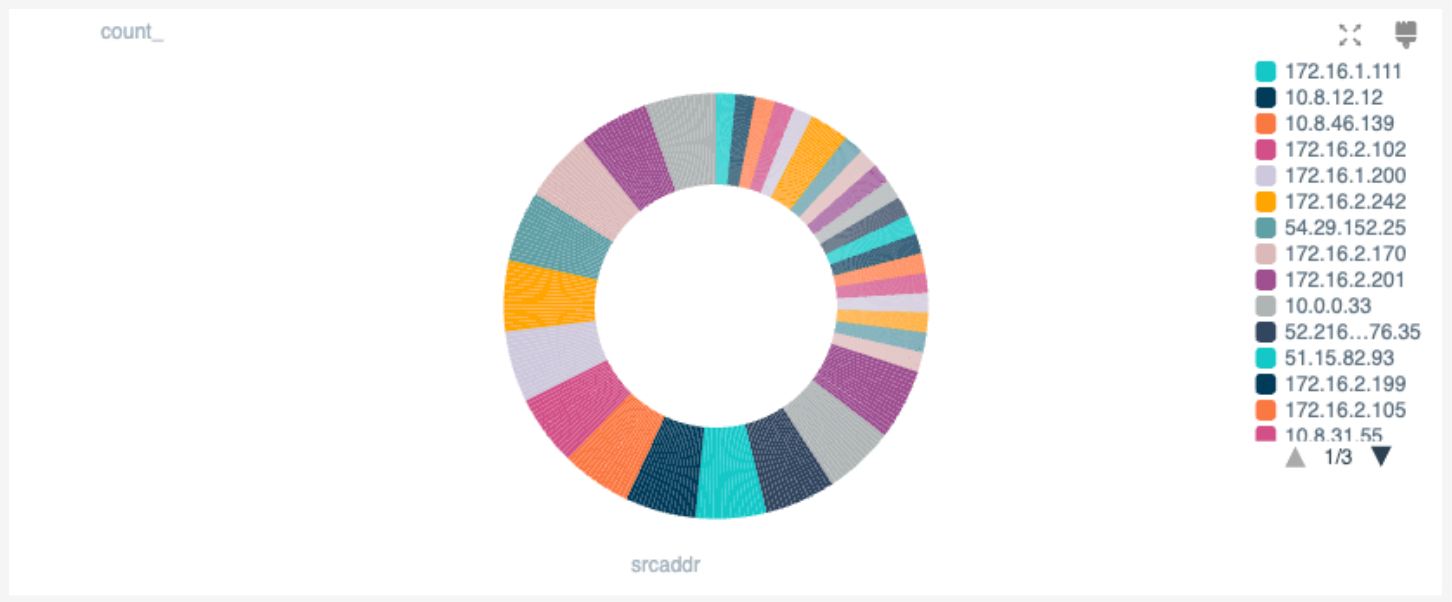
Position: Where to display the legend relative to the chart.

Truncate: By default, Cribl Search truncates labels to fit your browser viewport, substituting a `...` for omitted characters. Toggle this off if you prefer to not alter labels.

4.1.1.3. Donut Chart

Visualize your search results, using a Donut Chart.

Donut Charts summarize how a total amount is divided into numerical proportions. Use a **Donut** Chart when you want to compare parts of a whole against the whole and its other parts, such as the distribution of operating system usage.



Donut Chart

Chart Settings

Color palette: Select the color scheme you want to use in your Chart.

Data Settings

Show only top N values: An optional numerical value that sets the Chart to display only the larger values of a Dataset, with smaller values filtered out. To display the smaller values grouped into a single category, enable **Summarize other values**.

Legend Settings

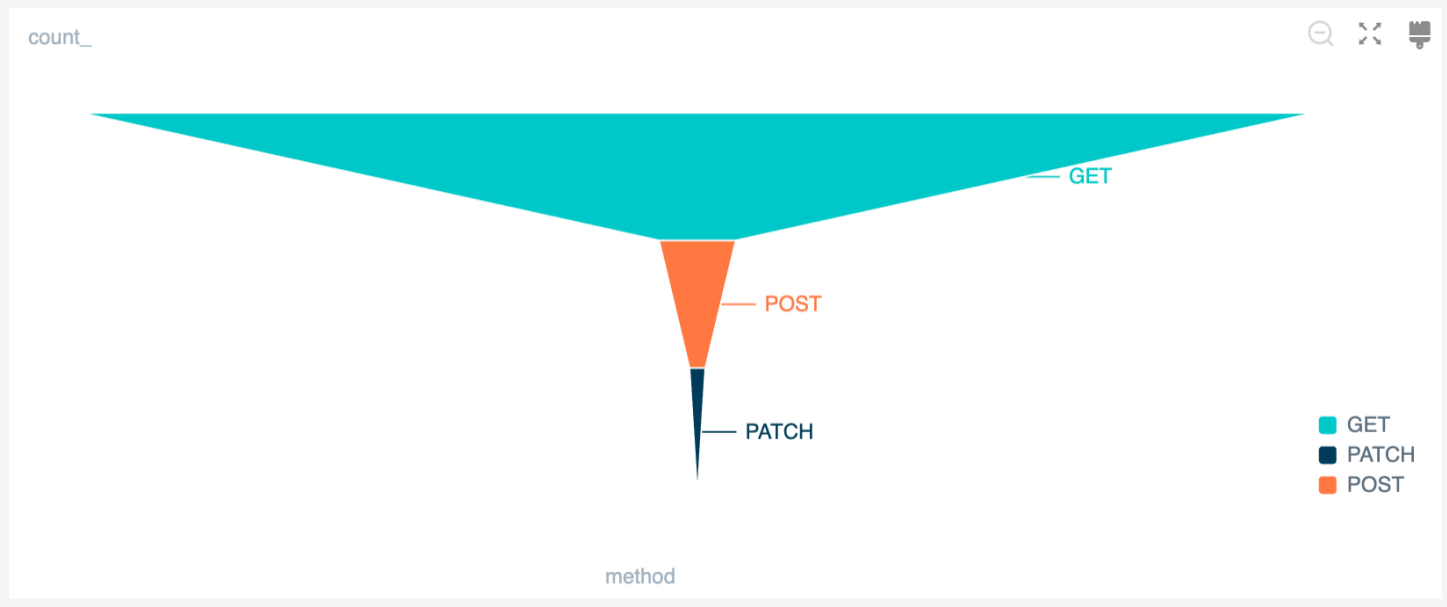
Position: Where to display the legend relative to the chart.

Truncate: By default, Cribl Search truncates labels to fit your browser viewport, substituting a `...` for omitted characters. Toggle this off if you prefer to not alter labels.

4.1.1.4. Funnel Chart

Visualize your search results, using a Funnel Chart.

A **Funnel** Chart helps visualize how data values move through a process in stages. **Funnel** Charts are useful for monitoring data that moves through a linear process in sequential, connected stages.



Funnel Chart

Chart Settings

Color palette: Select the color scheme you want to use in your Chart.

Data Settings

Show only top N values: An optional numerical value that sets the Chart to display only the larger values of a Dataset, with smaller values filtered out. To display the smaller values grouped into a single category, enable **Summarize other values**.

Legend Settings

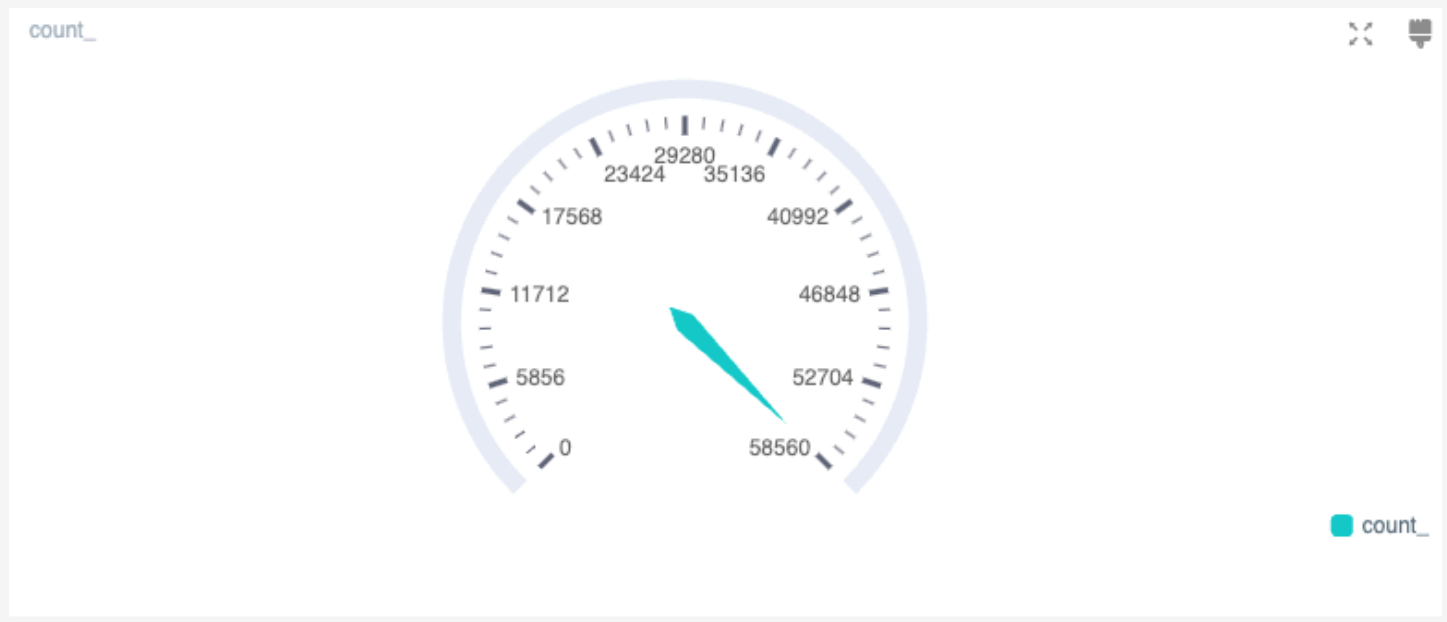
Position: Where to display the legend relative to the chart.

Truncate: By default, Cribl Search truncates labels to fit your browser viewport, substituting a `...` for omitted characters. Toggle this off if you prefer to not alter labels.

4.1.1.5. Gauge Chart

Visualize your search results, using a Gauge Chart.

Gauge Charts (also known as dial or speedometer Charts) show progress toward a specific target or metric, displaying a specific value as a measure of progress toward that target.



Gauge Chart

Chart Settings

Apply threshold: Apply custom color threshold settings.

Enable **Apply threshold**, and then change or add custom colors and threshold values to change the display of the Chart.

Legend Settings

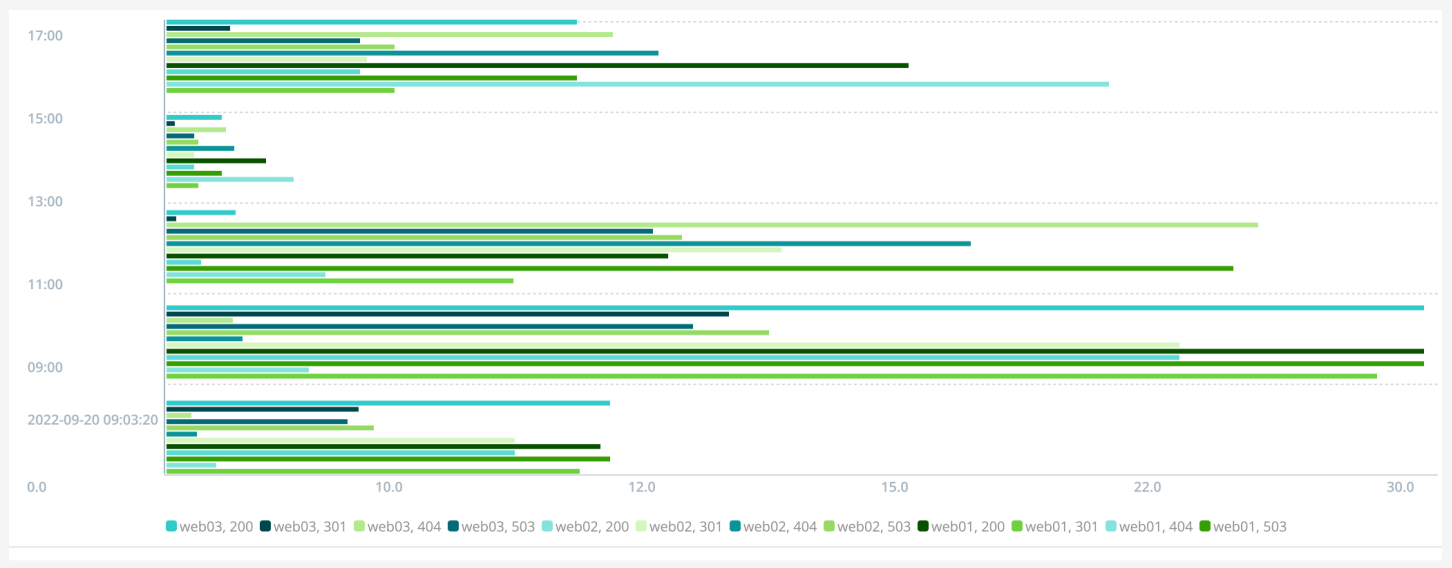
Position: Where to display the legend relative to the chart.

Truncate: By default, Cribl Search truncates labels to fit your browser viewport, substituting a `...` for omitted characters. Toggle this off if you prefer to not alter labels.

4.1.1.6. Horizontal Bar Chart

Visualize your search results, using a Horizontal Bar Chart.

A **Horizontal Bar** Chart is best used to render categorical results (such as "Top X Values") or to render a large numbers of results.



Horizontal Bar Chart

If you prefer a bar Chart with vertical orientation, select the Column Chart type.

Chart Settings

Color palette: Select the color scheme you want to use in your Chart.

Save chart settings: Whether or not to persist Chart customizations when running new queries. For details, see Saving or Resetting Customizations.

Data Settings

Interpolation: Select **Connect null values** to display null values. The default **Leave gaps** option hides null values.

Stack series: When enabled, data in the same category are stacked together. (This option is only configurable for Bar and Area Charts.)

Series table: You can edit the name of a series in the **Series name** column. The **Type** column allows you to set a different Chart type for each series. Use the **Color** column to set the desired color for the series.

The **Trellis view** option is not displayed, and the **Stack series** option is not selectable, on this Chart type.

X-Axis Settings

X-axis field: The data field to use for the X axis of the Chart. Defaults to a field chosen automatically based on the selected search.

Title: The label shown on the x-axis. Defaults to the field name that the axis references. Enter a custom name if needed.

Label orientation: The angle of rotation to display additional labels. Defaults to **0°**.

Position: Where to display the axis labels on the Chart.

Y-Axis Settings

Y-axis fields: The data fields to use for the Y axis of the chart. Defaults to fields chosen automatically based on the selected search.

Title: Defaults to the field name the axis references. Enter a custom name if needed.

Position: Where to display the axis labels on the chart. Defaults to **Left**.

Scale: Sets the distance between results. The default **Linear** scale uses an equal rate of change. A **Logarithmic** scale uses a percentage change.

Interval: The size of the axis measurement unit. For example, each interval on the y-axis can represent 2 or even 10 units, while each interval on the x-axis represents 1 unit.

Min: The minimum limit of units to display.

Max: The maximum limit of units to display. Defaults to `dataMax` where the maximum unit of data is used.

Show gridlines: Disable to hide the horizontal lines on the chart.

Legend Settings

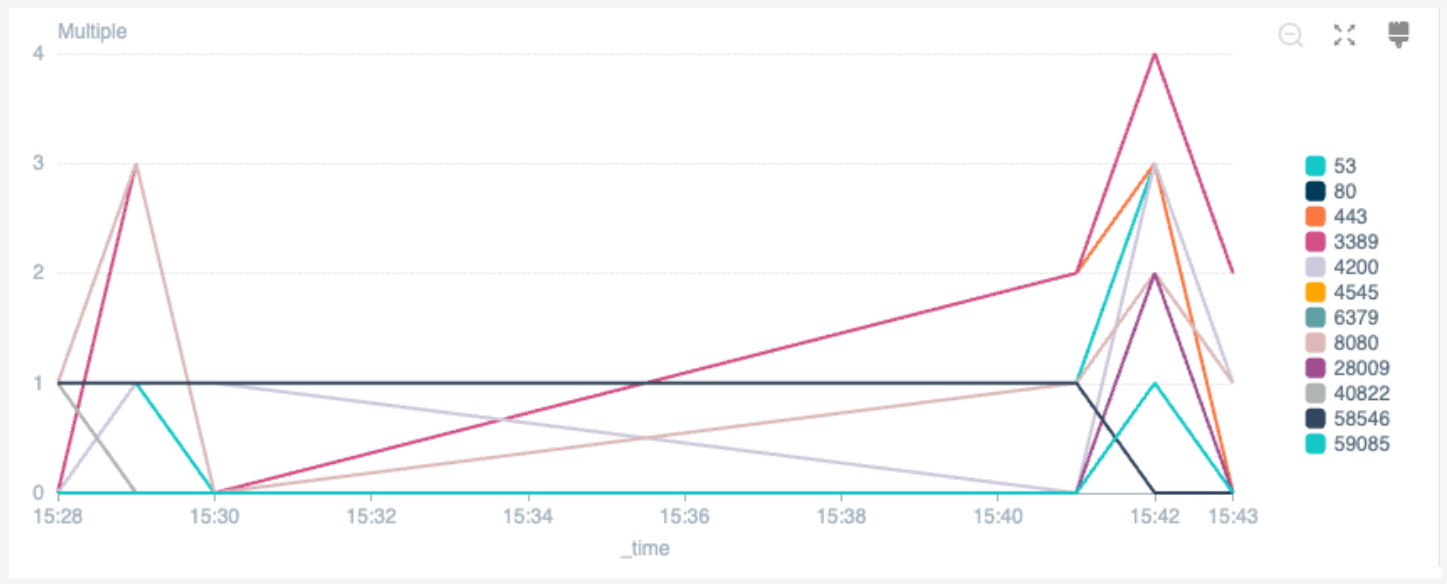
Position: Where to display the legend relative to the chart.

Truncate: By default, Cribl Search truncates labels to fit your browser viewport, substituting a `...` for omitted characters. Toggle this off if you prefer to not alter labels.

4.1.1.7. Line Chart

Visualize your search results, using a Line Chart.

A **Line** Chart is ideal for visualizing data over time, such as hardware usage and error events. These Charts work well with large and continuous sets of data.



Line Chart

Chart Settings

Color palette: Select the color scheme you want to use in your Chart.

Data Settings

Trellis view: Enable to divide the Chart into subsets of data based on fields in your search. Select a field from the **Split view by** list (available only when your search query includes two or more fields.)

Interpolation: Select **Connect null values** to display null values. The default **Leave gaps** option hides null values.

Stack series: When enabled, data in the same category are stacked together. (This option is only configurable for Bar and Area Charts.)

Series table: You can edit the name of a series in the **Series name** column. The **Type** column allows you to set a different Chart type for each series. Use the **Color** column to set the desired color for the series.

X-Axis Settings

X-axis field: The data field to use for the X axis of the Chart. Defaults to a field chosen automatically based on the selected search.

Title: The label shown on the x-axis. Defaults to the field name that the axis references. Enter a custom name if needed.

Label orientation: The angle of rotation to display additional labels. Defaults to **0°**.

Position: Where to display the axis labels on the Chart.

Y-Axis Settings

Y-axis fields: The data fields to use for the Y axis of the chart. Defaults to fields chosen automatically based on the selected search.

Title: Defaults to the field name the axis references. Enter a custom name if needed.

Position: Where to display the axis labels on the chart. Defaults to **Left**.

Scale: Sets the distance between results. The default **Linear** scale uses an equal rate of change. A **Logarithmic** scale uses a percentage change.

Interval: The size of the axis measurement unit. For example, each interval on the y-axis can represent 2 or even 10 units, while each interval on the x-axis represents 1 unit.

Min: The minimum limit of units to display.

Max: The maximum limit of units to display. Defaults to `dataMax` where the maximum unit of data is used.

Show gridlines: Disable to hide the horizontal lines on the chart.

Legend Settings

Position: Where to display the legend relative to the chart.

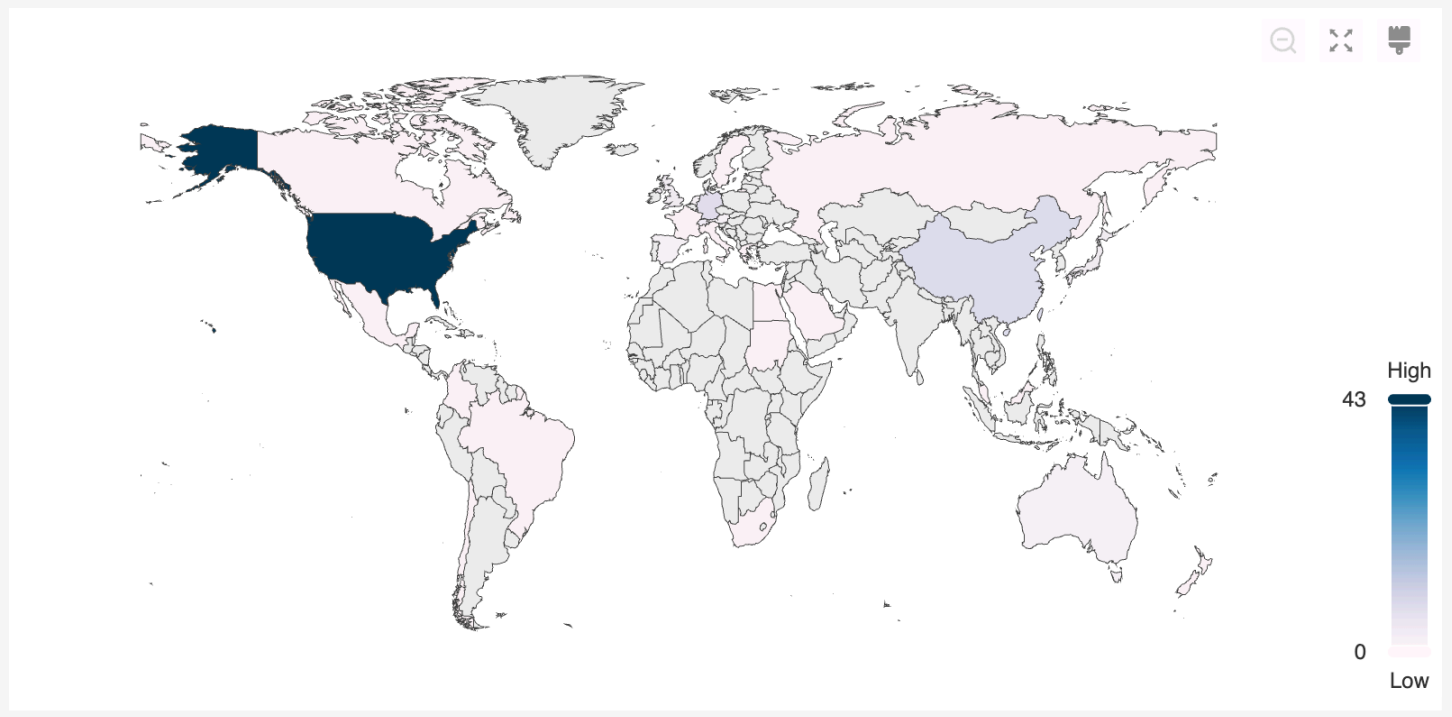
Truncate: By default, Cribl Search truncates labels to fit your browser viewport, substituting a `...` for omitted characters. Toggle this off if you prefer to not alter labels.

4.1.1.8. Map Chart

Visualize your search results, using a Map Chart.

Map Chart use geographical maps to display data points or categories associated with specific geographic locations.

To create Map Charts, your search needs to return supported geolocation data. You can use the `ip-lookup` operator, which references geolocation data from MaxMind's databases, or extract this information from your data. The Map Chart uses the first column (X-axis) to match the polygon name, and the second column (Y-axis) as the data.



Map Chart

Chart Settings

Color palette: Select the color scheme you want to use in your Chart.

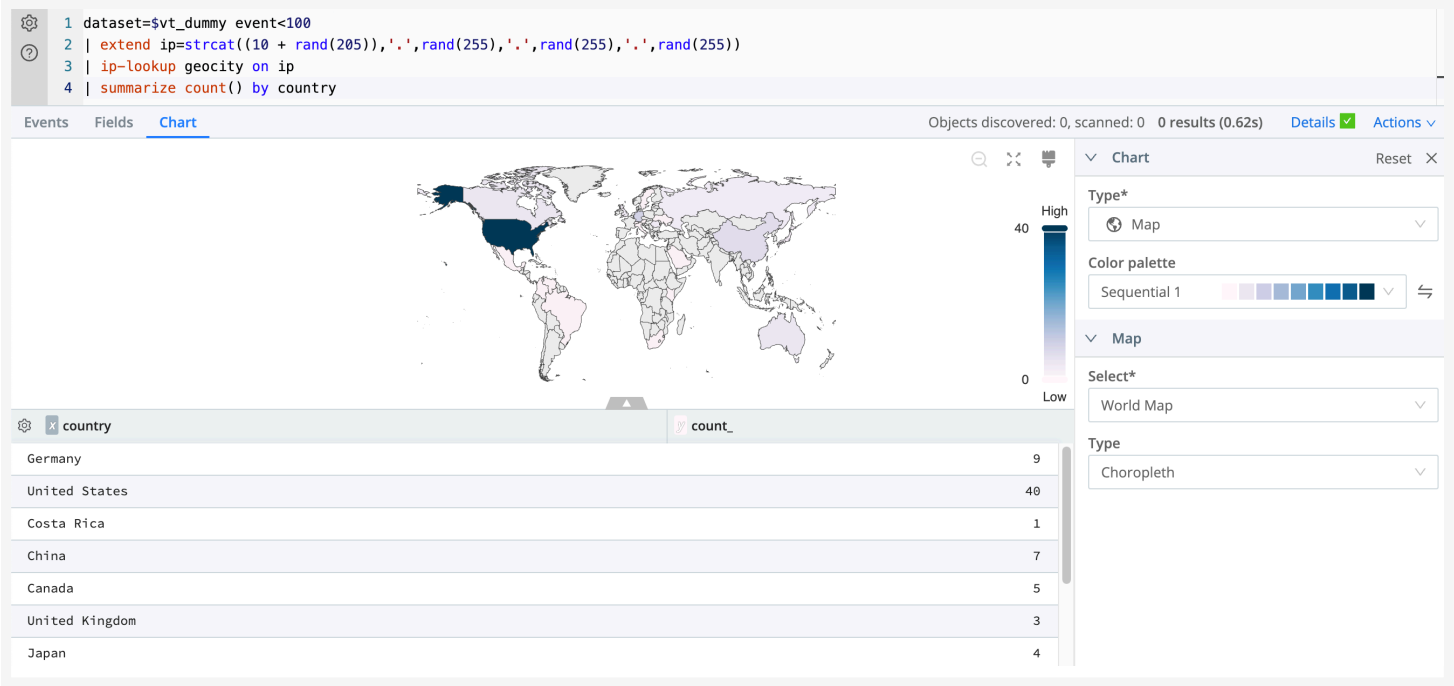
Map Settings

First select either **World Map** or **USA Map**. Then choose whether you want to display the map as **Choropleth** or **Geo Map**.

A **choropleth** map displays divided geographical regions colored in relation to a data variable. This Chart uses the first column (X-axis) to match the polygon name, and the second column (Y-axis) as the data. A choropleth

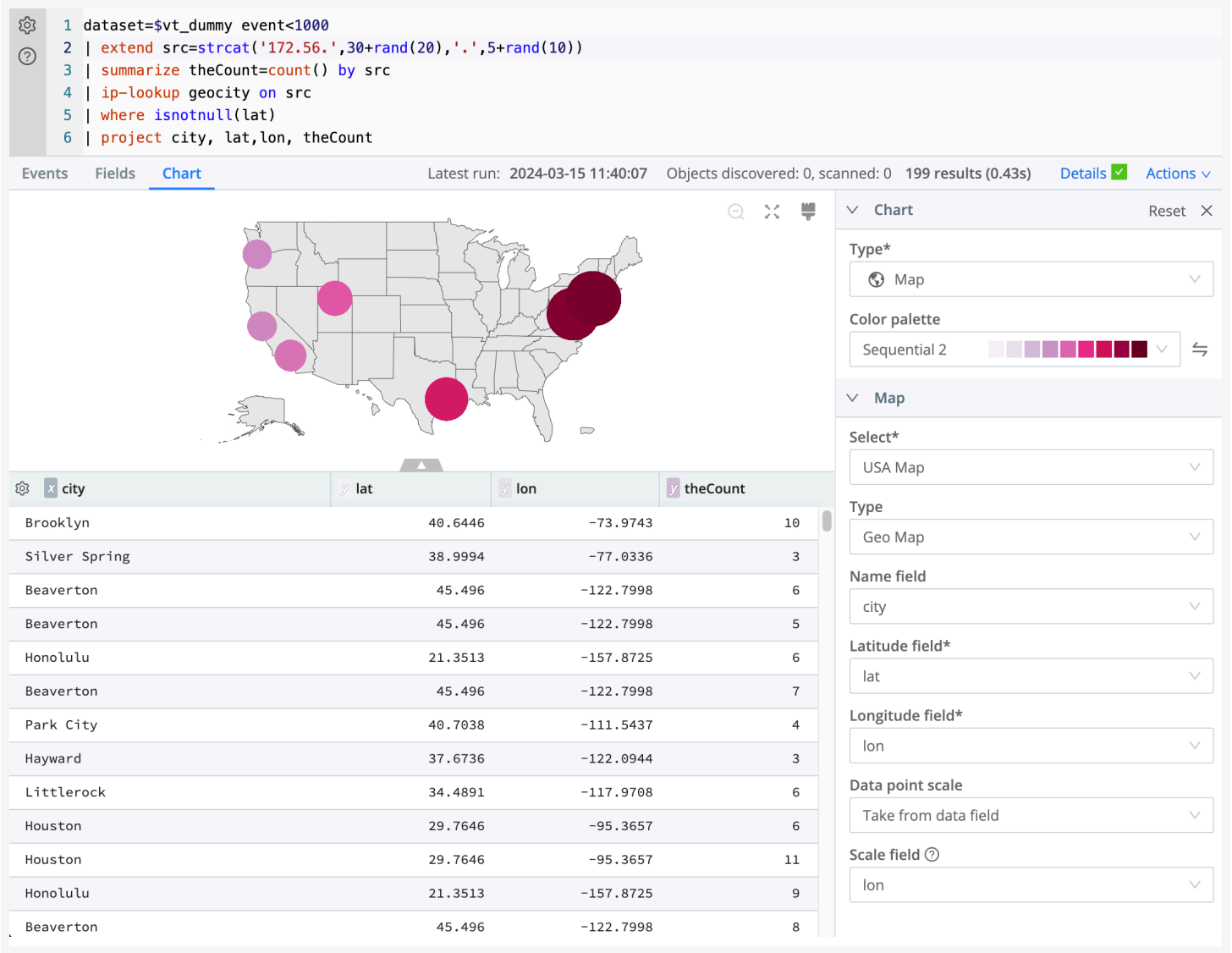
map is good for comparing high-level data at a glance.

To narrow the Chart's display of data in a choropleth map to show only a certain range of values, drag the **High/Low slider** handles from the top or bottom. To see individual data points highlighted on the map, hover your pointer over different parts of the **High/Low** slider.



Choropleth map example

A **geo** (geographic) map displays scaled data points based on a numeric value. You can customize the visualization by specifying the **Name**, **Latitude**, and **Longitude** fields, and you can select a specific scale or apply scale based on a data set in your query.



Geo map example

Sample Searches

The following example searches use the GeoLite2-City database in a lookup table called `geocity`. Note that to set them up, you'll need to first download a MaxMind `.mmdb` file and install it as a lookup table (follow the instructions in ip-lookup):

World Map

```
dataset=$vt_dummy event<100
| extend ip=strcat((10 + rand(205)),'.',rand(255),'.',rand(255),'.',rand(255))
| ip-lookup geocity on ip
| summarize count() by country
```

Copy Run in Cribl Search

USA Map

```
dataset=$vt_dummy event<1000
| extend src=strcat('172.56.',30+rand(20),'.',5+rand(10))
| summarize theCount=count() by src
| ip-lookup geocity on src
| where isnotnull(lat)
| project city, lat,lon, theCount
```

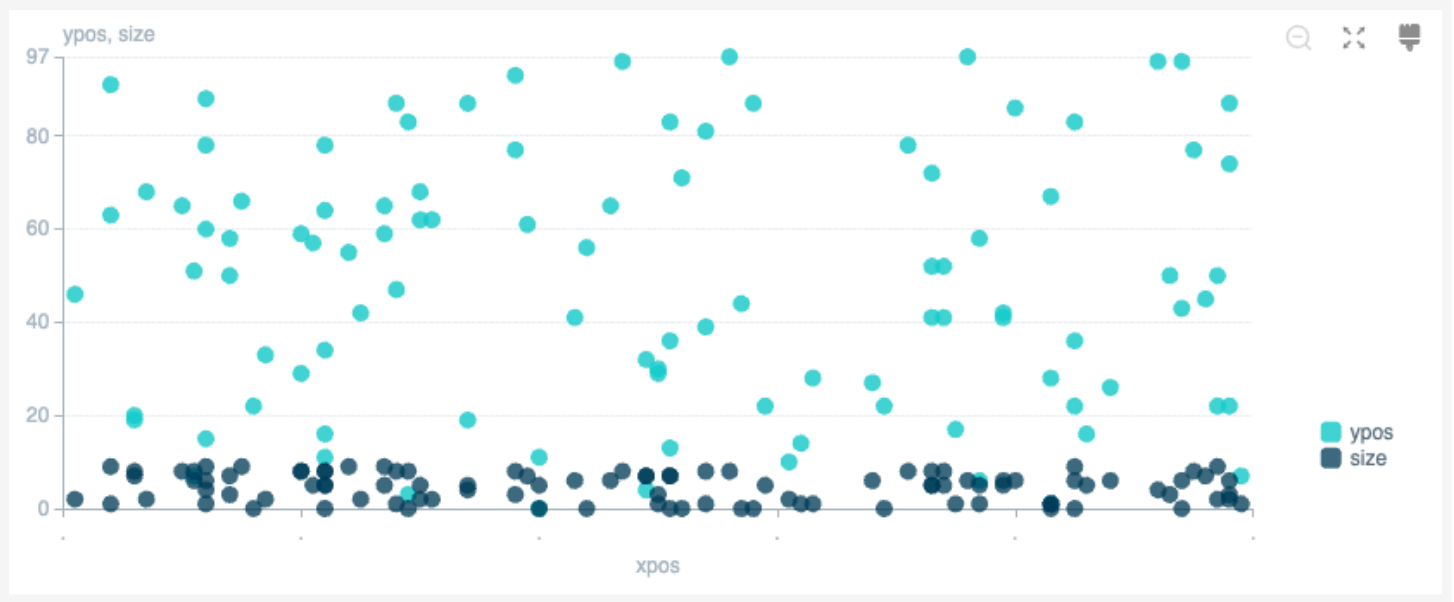
Copy Run in Cribl Search

After running these searches, change the chart `type` to `map`, to see a map of the results.

4.1.1.9. Scatter Chart

Visualize your search results, using a Scatter Chart.

Use a **Scatter** (or scatter plot) Chart to determine whether two numeric variables in a set of individual data points have a relationship or correlation. Dots or circles represent these data points.



Scatter Chart

Chart Settings

Color palette: Select the color scheme you want to use in your Chart.

Data Settings

Lines: Enable to connect all the data points with a line, similar to a **Line** Chart.

Name field: Adds labels to data points.

Data point scale: Sets the display size for data points. Select **Take from data field** to scale the data point size based on the selected field.

Scale field: The field to use for the data point scale, if **Take from data field** is selected.

X-Axis Settings

Title: The label shown on the x-axis. Defaults to the field name that the axis references. Enter a custom name if needed.

Label orientation: The angle of rotation to display additional labels. Defaults to **0°**.

Position: Where to display the axis labels on the Chart.

X-axis field: The field that will be used for the X-axis.

Y-Axis Settings

Title: Defaults to the field name the axis references. Enter a custom name if needed.

Position: Where to display the axis labels on the Chart. Defaults to **Left**.

Scale: Sets the distance between results. The default **Linear** scale uses an equal rate of change. A **Logarithmic** scale uses a percentage change.

Y-axis field: The field that will be used for the Y-axis.

Interval: The size of the axis measurement unit. For example, each interval on the y-axis can represent 2 or even 10 units, while each interval on the x-axis represents 1 unit.

Min: The minimum limit of units to display.

Max: The maximum limit of units to display. Defaults to `dataMax` where the maximum unit of data is used.

Show gridlines: Disable to hide the horizontal lines on the Chart.

Legend Settings

Position: Where to display the legend relative to the chart.

Truncate: By default, Cribl Search truncates labels to fit your browser viewport, substituting a `...` for omitted characters. Toggle this off if you prefer to not alter labels.

4.1.1.10. Single Value Chart

Visualize your search results, using a Single Value Chart.

A **Single Value** Chart allows you to display one search result. If there is more than one result, the first result is charted. These are helpful when you have an important value to track.



Single Value Chart

Chart Settings

Label: Defaults to the field name referenced on the Chart. Select the color block to the right of the **Label** field to choose a display color for the label and value.

Style: Select whether the color selected in the **Label** field appears as the background or foreground of the Chart.

Prefix: Add text before the single value.

Suffix: Append text after the single value.

Decimal places: Select the number of digits to the right of a decimal point.

Use 1000 separator (,): Display the single value with the , thousands separator. Enabled by default.

Apply threshold: Apply custom color threshold settings. Note that applying thresholds overwrites any color you may have selected in the **Label** field.

Enable **Apply threshold**, and then change or add custom colors and threshold values to change the display of the Chart.

4.1.2. Results Tables

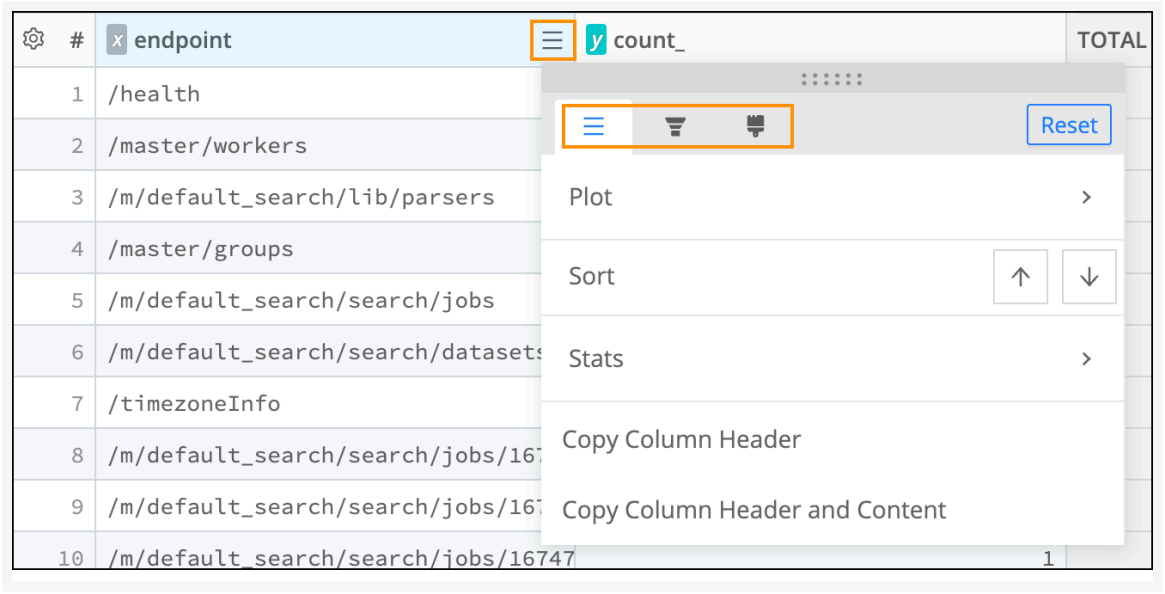
Learn how to format tables that display search results.

Below every Chart is a results table that displays the Chart's data in numbers and/or text.

You can modify the table display, as well as some Chart visualization settings, in several ways. You'll find most of the available settings in each column's Settings drop-down:

1. On the **Chart** tab, scroll down to the table below the Chart.
2. Hover over the column header that contains the data you want to work with.
3. Select the Settings button ☰ to open the drop-down.
4. Select one of the three tabs on the drop-down:

- **General** ☰ – for plot settings, column sorting, stats info, and copying
- **Filter** – for filtering results based on search terms
- **Format** – for applying prefixes, suffixes, decimal places, and column colors



Settings drop-down/modal

You can use the grab handle at the top of the Settings drop-down to tear it off as a modal, and move it anywhere on the page. This way, it won't block the column you're modifying, and you can see your changes as you make them.

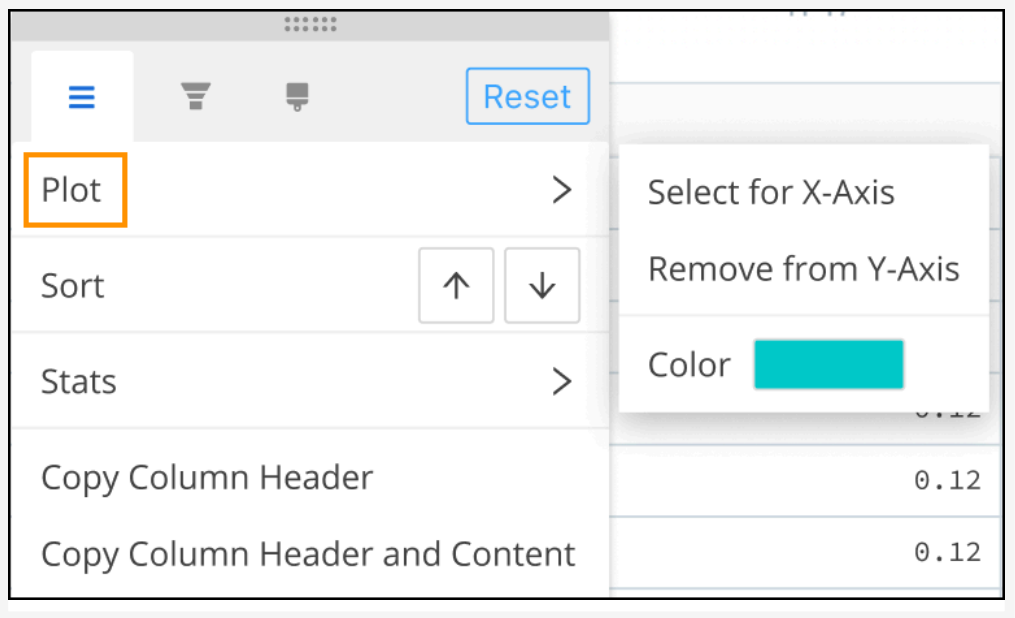
Changes made in the Settings drop-down apply only to the column selected when you opened the drop-down.

Adjust Plot Settings and Axis Colors

Adjust plot settings to change the display of column results in the associated Chart. You can choose which columns to display on the x-axis and y-axis, and you can change the color of column data in the Chart display.

You can open the plot menu in two ways:

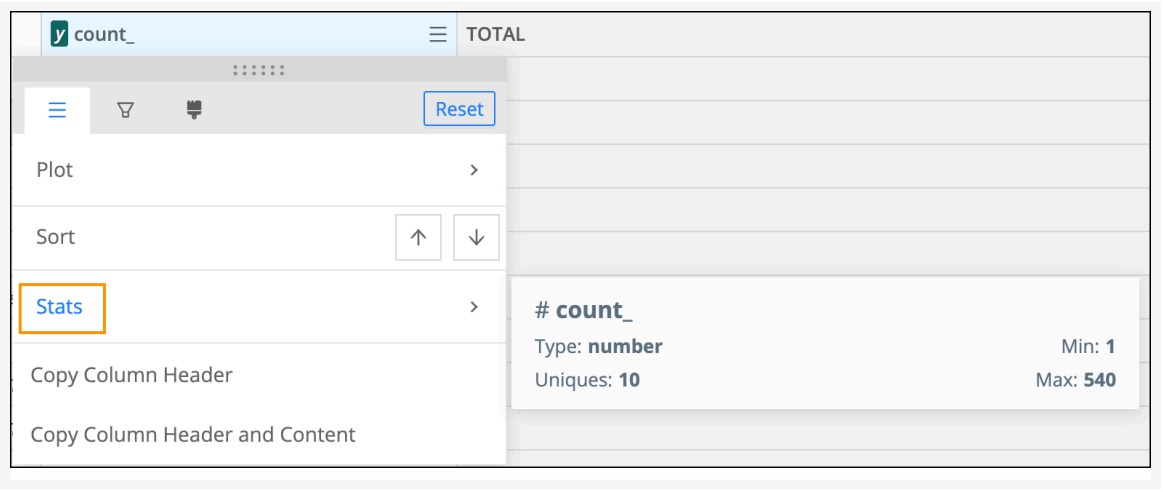
- Select the plot icon in a column header.
- On the **General** tab , select **Plot**.



Plot options on the General tab

View Stats

On the **General** tab , select **Stats** to see information about the fields returned from your search, including the data type and the count of unique field values.

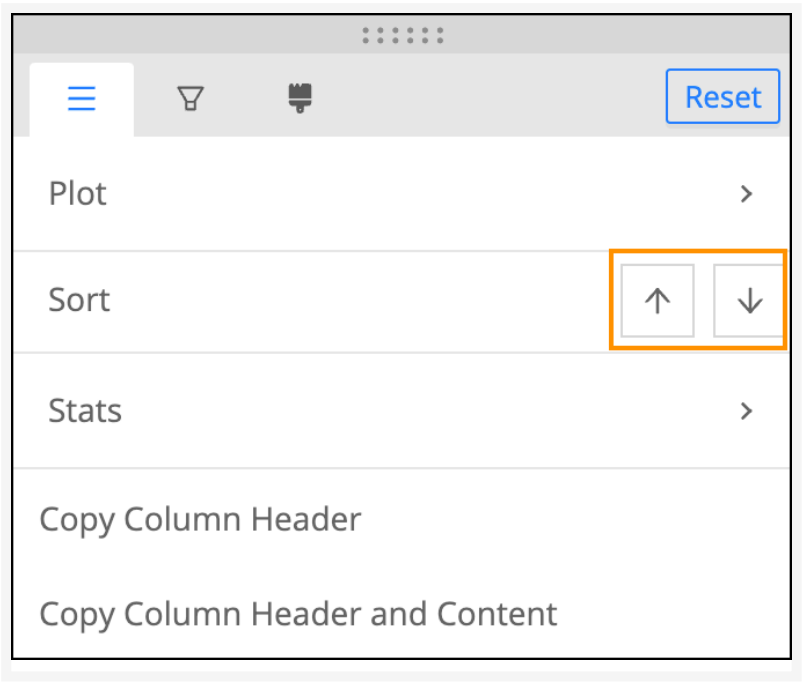


Stats on the General tab

Sort Data

You can sort column results in ascending or descending order in two ways:

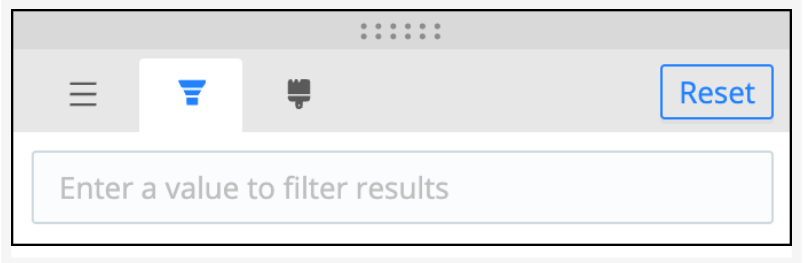
- Select the arrow buttons ⇕ that appear when you hover over a column header.
- On the **General** tab ☰, select the arrow buttons next to **Sort**.



Sort arrows on the General tab

Filter Data

On the **Filter** tab ▼, you can filter the results in a column by searching for a particular value.



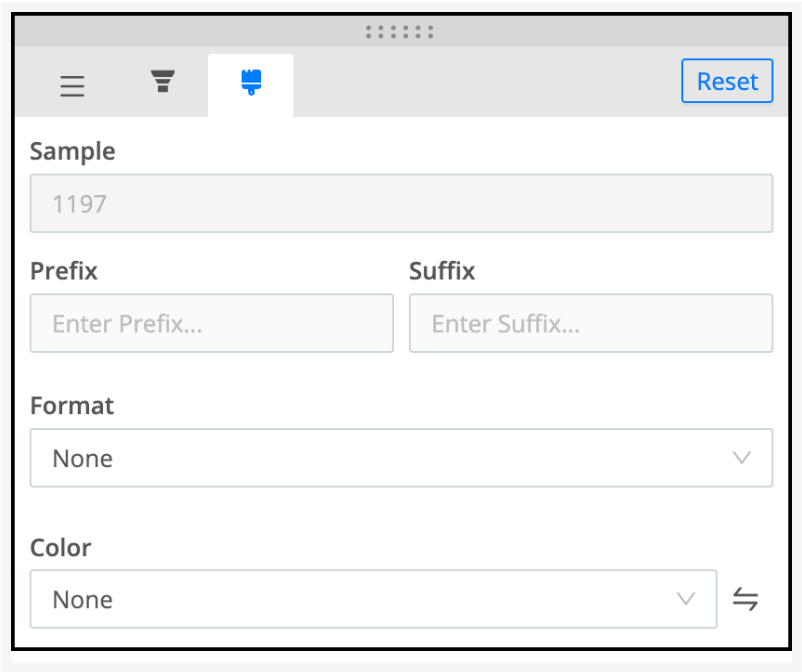
Filter tab

Apply Column Formatting

The **Format** tab 🖌 provides the following options:

- Enter a **Prefix** or **Suffix** to be applied to every result in a column.

- **Format** can be a **Number**, **Currency**, or **Percentage**. You can specify the number of decimal places to show in a column. To display or hide the thousands separator, select or clear the **Use 1000 Separator (,)** check box.
    - **Currency** provides a drop-down with supported currencies.
- **Color** allows you to apply a value-based color palette to a column.

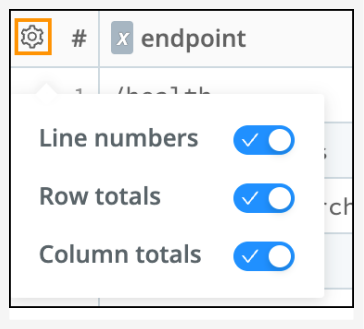


Format tab

When any field on this tab is modified, the Settings icon in the column header displays a blue dot to indicate that formatting is applied.

Show Line Numbers, Column Totals, and Row Totals

Select the gear button to the left of the results table to show or hide the display of line numbers, row totals, or column totals.



Toggles for line numbers, row totals, and column totals

By default, row and column totals are pinned in place, for visibility. Select the pin button 📌 to unpin them.

4.2. Dashboards

Visualize your search results, using Dashboards.

Cribl Search offers customizable Dashboards with a variety of visualizations, in the form of panels, that you can tailor to fit your needs. Visualization panels are discrete units of displayed data that you can organize, interact with, and modify.

You might add visualizations to a Dashboard to monitor server performance by time series and event timeline. Or, you might add visualizations to display customer behavior with donut charts and top-hitters lists.



Sample Dashboard

To start working with Dashboards, select **Dashboards** from the Cribl Search sidebar.

For details about who can access and modify this resource, see Search Member Permissions. With appropriate Permissions, you can export, share, and import Dashboards in Packs.

To zoom any Dashboard visualization panel up to a larger view, select the Actions ⋯ button at the panel's top right, then select **Full Screen**.

Cribl Search includes two Dashboards that you can use right away:

- **Search Monitoring** displays search status and CPU seconds over time, as well as details about completed searches and searches by user.
- **Sample Dashboard** provides examples of the types of data displays you can create with visualization panels.

These two Dashboards are standard for all users and cannot be edited. However, you can clone the Sample Dashboard and revise it for your own needs.



To quickly access your most recent Dashboards from any Cribl Suite product, select **Products** in the top bar, then hover over **Dashboards.**

4.2.1. Create A Dashboard

Learn how to create a Dashboard in Cribl Search.

You can create new Dashboards or add to existing Dashboards in several ways.

Create Dashboard From the Dashboards Tab

On the Dashboards page, you can create a new Dashboard and add visualization panels.

1. From the Cribl Search sidebar, select **Dashboards**.
2. Select **Add Dashboard**.
3. Give your new Dashboard a name and (optionally) a description.
4. Optionally, select or create a **Collection** to contain this Dashboard.
5. Optionally, configure a **Schedule** to automatically refresh your Dashboard at a set interval. For more details on Dashboard scheduling, see Refresh Your Dashboards Automatically.
6. When your basic configuration is complete, select **Save**.

Add Visualizations to a Dashboard

In the empty Dashboard that appears next, you'll see the following options:

- **Add Visualization Using Copilot**: This prompts you to specify a Dataset, and then prompts the Cribl Copilot AI to suggest visualizations for that Dataset. For details, see Add a Visualization when Creating a Dashboard.
- **Add Visualization**: This prompts you to specify a new or saved search, and then define a corresponding chart type. For details, see New Search and Existing Search.
- **Add Input**: This prompts you to define a visualization panel constrainted by a new or saved search, a range of values or numbers, a time range, or freeform text. For details, see Add Inputs to Your Cribl Search Dashboard.
- **Add Markdown**: This prompts you to define a visualization panel that offers users explanations or instructions on using your Dashboard. You can use markdown formatting to add arbitrary text, tables, and/or images. For details, see Add Markdown.

The three last options above are also available on the **Add** drop-down at the upper right. To get suggestions from Cribl Copilot after you've configured a Dashboard, see Add a Visualization when Editing a Dashboard.

Create Dashboard From the Products Sidebar

You can create a new Dashboard from the **Products** sidebar, too:

1. Select **Products** on the top bar.
2. In the **Products** sidebar, under **Search**, hover over **Dashboards** and select the **+** button.
3. In the **New Dashboard** modal, give your new Dashboard a name and (optionally) a description.
4. Optionally, select or create a **Collection** to contain this Dashboard.
5. Optionally, configure a **Schedule** to automatically refresh your Dashboard at a set interval.
   For more details on Dashboard scheduling, see Refresh Your Dashboards Automatically.
6. When your basic configuration is complete, select **Save**.

Create Dashboard From Search Results

After performing a search, you can add it as a visualization panel to a new or existing Dashboard.

1. Select the **Actions** drop-down, and select **Add to Dashboard**.
2. Select whether you want to add the visualization to a new or existing Dashboard.
3. Select or name the Dashboard.
4. Enter the **Panel title**. This will be the name of the visualization panel in your Dashboard.
5. Select **Add & Go to Dashboard**.

Create Dashboard From the History or Saved Tab

Both in **History** and **Saved Searches**, you can turn any search in the list into a visualization panel, and add it to a new or existing Dashboard.

1. Select the Actions ⋯ button to the right of a search and select **Add to Dashboard**.
2. Select whether you want to add the visualization to a new or existing Dashboard.
3. Select or name the Dashboard.
4. Enter the **Panel title**. This will be the name of the visualization panel in your Dashboard.
5. Select **Add & Go to Dashboard**.



Wondering about the difference between saving a search and adding a search to a Dashboard? A saved search is a query that can be rerun to return new or updated data. You can add saved searches to Dashboards in the form of visualization panels. All saved searches are available to add to Dashboards, but adding a search to a Dashboard does not add it to the **Saved Searches** list.

Create Dashboard From a Pack

You can import pre-built Dashboards from Packs, and you can also add Dashboards within a Pack.

4.2.2. Edit A Dashboard

Learn how to edit a Dashboard in Cribl Search.

Within a Dashboard, you can add and edit visualization panels. You can also add Dashboard-level Inputs that apply to all visualizations on the Dashboard.

To edit the built-in Sample Dashboard, you need to clone it first. Go to **Dashboards** > **Sample Dashboard** > ⋯ > **Clone**.

Add a Visualization

1. In an open Dashboard, select **Edit** at the top right, or press **e** on your keyboard.
2. From the **Add** drop-down, select **Visualization**.

Next, choose how to define your visualization: by reviewing Cribl Copilot AI suggestions; or by configuring it yourself, based on a new or existing search.

To have Cribl Copilot suggest visualizations that display your data effectively:

1. At the top of the page, select the Cribl Copilot button .
2. The right drawer, now labeled **Copilot**, will prompt you to select or change the base Dataset for this visualization.
3. From the tiles that Cribl Copilot suggests, select **Add** to add the visualization you choose.

For details, see Add a Visualization when Editing a Dashboard.

New Search

If you want to base your visualization on a **New Search** (the default setting in the right drawer):

1. Either enter your query directly, or select a **Parent search ID** to pre-populate your query from a parent search (see Apply a Parent Search below).
2. Optionally, if you've started with a parent search, you can append terms to the query before proceeding.
3. Select **Search**.

To start over, select **Reset**. This clears the query and all chart settings.

Existing Search

If you prefer to base your visualization on an existing saved search (which can be a scheduled search):

1. In the right drawer, select **Saved**.
2. Select a **Saved search ID** to populate your query.
3. Optionally, append terms to the query before proceeding.
4. Set the **Run mode**:
    - **Run as new search** – Always rerun the query, even if the Dashboard has data caching enabled.
    - **Fetch results from last run** – Get results from the most recent run. (If the Dashboard has data caching enabled, Cribl Search will ignore this setting.) Cribl recommends selecting this option if you're basing this visualization on a scheduled search.
5. Select **Search**.

To start over, select **Reset**. This clears the query and all chart settings.

Edit a Visualization

To edit an existing visualization:

1. Open an existing Dashboard.
2. At the top right of the visualization panel you want to modify, select **Edit** . Depending on the layout, you might need to select the Actions ⋯ button first.
3. Edit the visualization just like you edit a Chart display. The options available depend on the Chart type used.

If you add a visualization by creating a new search, you'll see additional options to configure the type, format, and data display of the visualization.

If you create a visualization from a saved search, you can select **Edit** at the visualization panel's top right to revise the visualization.

Hide a Visualization

You can hide any visualization panel by editing it and selecting the **Hide this panel** check box.

- In Dashboard edit mode, hidden panels are collapsed into a **Hidden Panels** section.
- To unhide a panel, edit the panel and clear the **Hide this visualization** check box.

Interact with Visualizations in a Dashboard

Within any visualization panel, you can hover over graphics or expand list events to see more details. Also, the Actions ⋯ buttons on the Dashboard, and on individual panels, offer further options.

Rearrange Visualization Panels

To arrange the visualizations in a Dashboard, select **Edit** at the top right, or press **e** on your keyboard. Use the drag handle at the top of any panel to drag it to a new position, and use the resize handle at the bottom right or left to resize it.

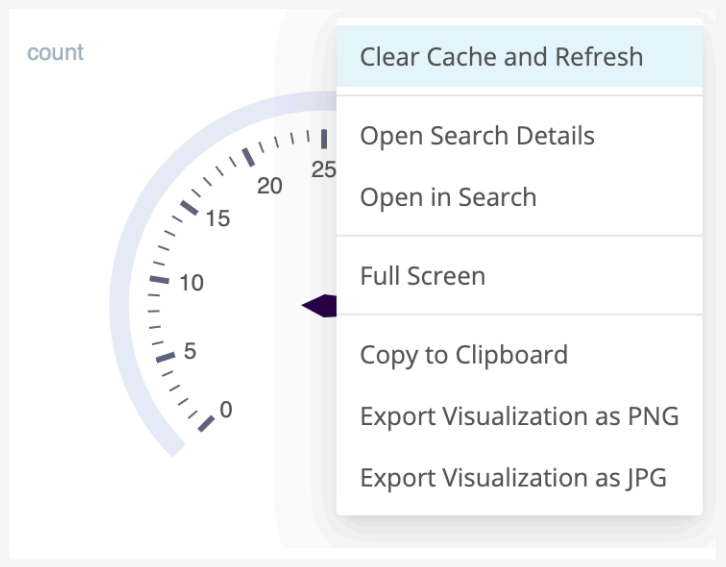
Refresh Visualizations

Refresh the data in a visualization by clicking the Refresh button at the top right of the panel.

To refresh all of the Dashboard's visualizations automatically, see Refresh Your Dashboards Automatically.

Manage Visualizations

Select a visualization panel's own Actions button to expose the menu options shown below.



Visualization actions

The options here include:

- **Edit**: Opens the visualization panel for editing.
- **Clear Cache and Refresh**: A hard refresh of the visualization, to match current underlying data. To refresh all of the Dashboard's visualizations automatically, see Refresh Your Dashboards Automatically.
- **Open Search Details**: Opens a read-only version of the Details modal, listing the search's query expression, execution times, and duration.
- **Open in Search**: Opens an editable version of the query expression.
- **Full Screen**: Zooms up an expanded display of the visualization, in a modal within your browser window.
- **Copy to Clipboard**: Copies the current visualization, as a static image, to the system clipboard.

- **Export Visualization as PNG/JPG**: Saves the current visualization as a static image in `.png` or `.jpg` format.

Apply a Parent Search

To reduce effort and significantly improve Dashboard loading time, you can reuse search results by saving them as **parent searches**. You can instantly add an existing search query to a visualization panel by designating another visualization in the same Dashboard as the parent search. This enables you to aggregate results from both searches into the same visualization.

1. Edit an existing Dashboard, and then select **Edit** at the top right of the visualization panel you want to modify.
2. Under **Parent search**, select the name of the visualization you want to use as the parent.
3. The parent search query becomes the new first line of your query in the **Search** box. Select **Search** to test your new aggregate query, and then select **Save** to save your Dashboard.

Hide a Parent Search

You can hide any visualization by editing it and selecting the **Hide this panel** check box. This can be especially useful for visualizations you create for the sole purpose of acting as parent searches.

Parent Search Limitations

You can apply only one parent search per visualization panel.

When you apply a parent search, child searches wait until the parent search has completed. So you will not see "preview" results.

When you've applied a parent search to a visualization, and you open that visualization (using **Actions** > **Open in Search**), you won't be able to edit the child search's time parameters. This is because the child search is using the parent search's time span.

Add Inputs

To enable viewers to filter and control visualizations in your Dashboard, you can add Dashboard-level Inputs. For more information, see Add Inputs to Your Cribl Search Dashboard.

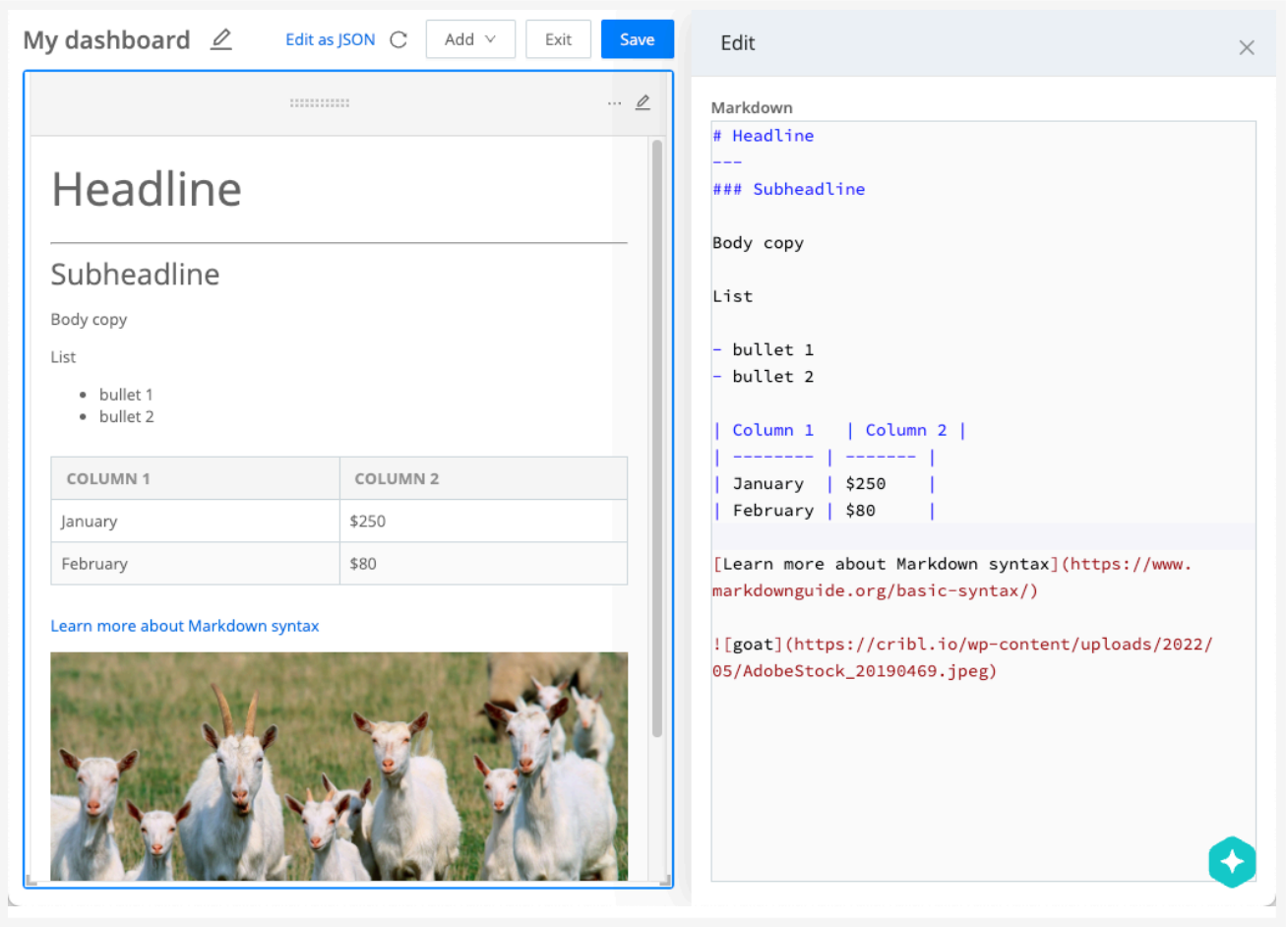
Add Interactions

To enable viewers to drill down into details when they select Dashboard data points, you can add Interactions to visualization panels. For more information, see Add Interactions to Your Cribl Search Dashboard.

Add Markdown

You can add a Markdown panel to your Dashboard to enrich it with any custom text. Use markdown to add explanations, captions, and additional information to your Dashboard. To add a Markdown panel:

1. In an open Dashboard, select **Edit** at the top right, or press **e** on your keyboard.
2. From the **Add** drop-down menu, select **Markdown**.
3. In the **Edit** pane, replace the sample text with any markdown content you want.
   The panel lets you use the features of Github-flavored markdown, including links, tables, codeblocks, and images (linking to external sources).
   You can also use tokens, defined in Inputs, surrounded by `$`. For example, if the Input ID is `firstname`, reference `$firstname$` in the markdown.
   As you edit, you will see a live preview of the rendered text.
4. Select **Save**.



Markdown panel with the pre-built markdown sample

The Markdown panel doesn't support raw HTML.

Rename a Dashboard

To change the name of an existing Dashboard:

1. From the Cribl Search sidebar, select **Dashboards**.
2. Select the row of the Dashboard that you want to rename.
3. At the top, next to the current Dashboard name, select Settings, and then select the **Edit** button.
4. Enter the new Dashboard name, and select **Save**.

Edit Dashboards as JSON

You can edit your Dashboards in a JSON editor built into Cribl Search, or import and export JSON files with complete configurations. This can significantly speed up setting up more complex Dashboards.

To edit an existing Dashboard as JSON:

1. From the Cribl Search sidebar, select **Dashboards**, and open a Dashboard.
2. At the top right, select the Actions ⋯ button.
3. Select one of the following:
    - **Edit as JSON** to open the Dashboard configuration in the Cribl Search JSON editor.
    - **Copy Dashboard JSON** to copy the Dashboard JSON to clipboard.
    - **Export Dashboard JSON** to download a JSON file with the Dashboard configuration.
    - **Import Dashboard JSON** to upload a JSON file with a new configuration.

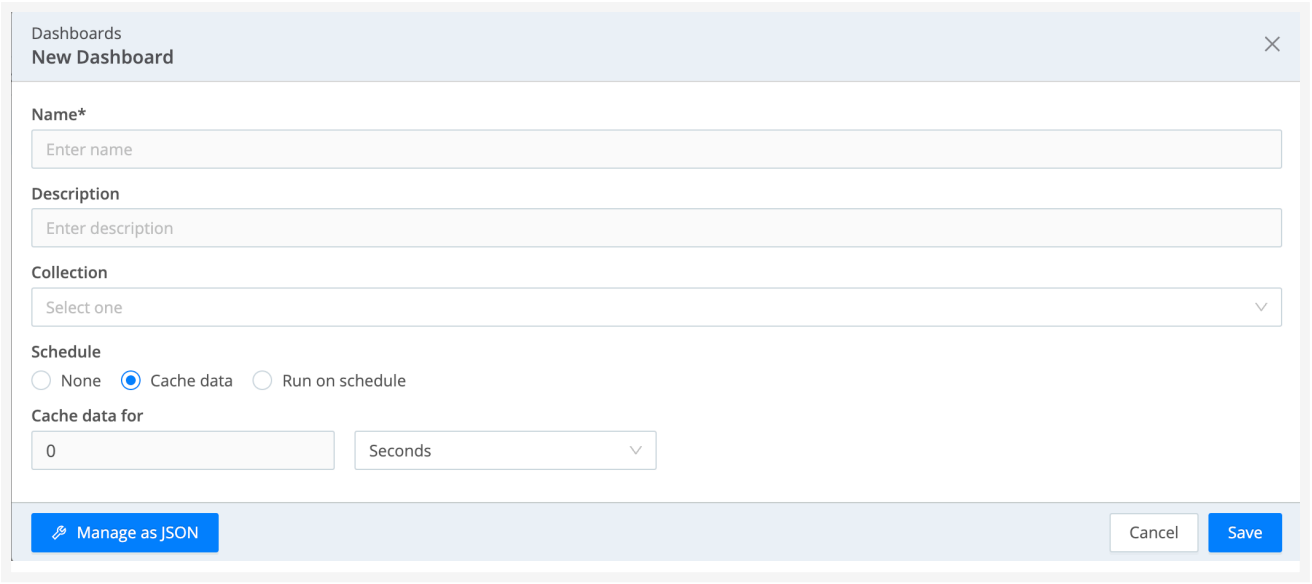
Refresh Your Dashboards Automatically

You can refresh a Dashboard automatically, either by running it on schedule, or by configuring its caching.

Configure a Dashboard's Caching

You can modify the time for which a Dashboard is cached (in other words, how often it is refreshed).

1. Start creating or editing a Dashboard.
2. Under **Schedule**, select **Cache data**.
3. In **Cache data for**, set the time-to-live (TTL) for the Dashboard cache. This setting defines how often data in the Dashboard is refreshed.



Configuring caching for a new Dashboard

Run a Dashboard on Schedule

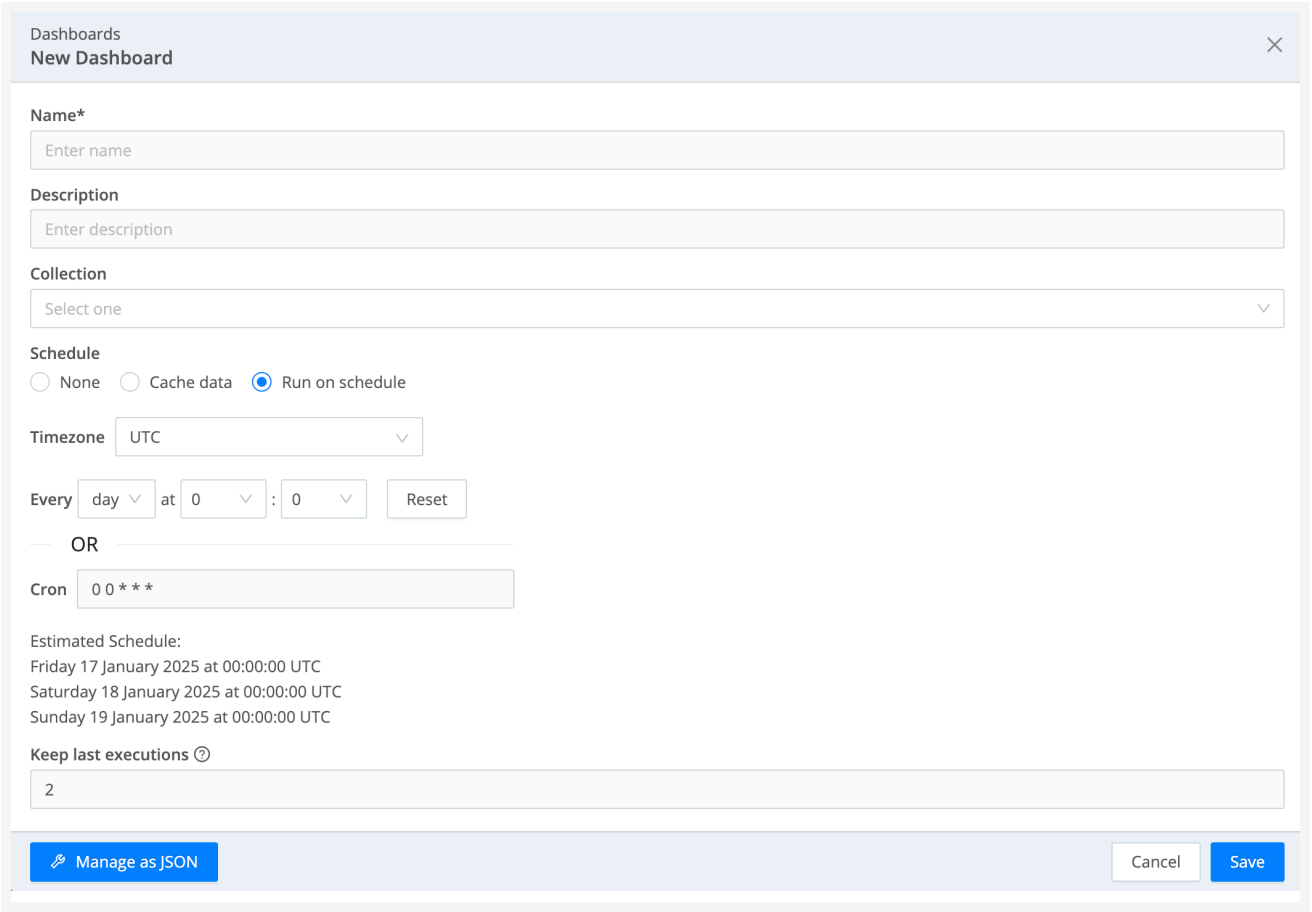
You can schedule the searches on **all** of a Dashboard's visualizations to refresh on the default interval you define.

This can be advantageous on a Dashboard where you don't need to display up-to-the-minute data, but you prefer not to wait for the display to refresh. Schedule the underlying searches to refresh daily, hourly, or every few minutes. When the visualizations render, they'll fetch the results from the last scheduled run.

After scheduling the Dashboard as a whole, you retain the option to customize the schedules of the searches that underlie individual visualization panels and Inputs.

To run a Dashboard on a schedule:

1. Start creating or editing a Dashboard.
2. Under **Schedule**, select **Run on schedule**.
3. Configure one of the two:
    - Define the timezone, the base interval (anything from `year` down to `minute`), and refresh time.
    - Or, define a **Cron** schedule string.
4. In **Keep last executions**, define how many past versions of the Dashboard data to keep.
   This is a rolling buffer – on each refresh of the Dashboard, Cribl Search will purge the oldest version beyond this count. Defaults to `2`.



Setting a schedule for a new Dashboard

Changing Scheduled Dashboard Ownership

If the creator of a scheduled Dashboard is no longer an active Cribl Search Member, active Members with access to the Dashboard will be unable to re-run the Dashboard. They will need to export the Dashboard's configuration, then re-create the Dashboard by importing the config, and finally schedule the new Dashboard to run under their ownership. {.box .warning}

4.2.3. Add Interactions To Your Cribl Search Dashboard

Enable Dashboard viewers to drill down into specific values in a visualization panel.

Why Use Dashboard Interactions

Interactions enable Dashboard viewers to take action directly from visualizations. By clicking on data points, users can trigger context-driven searches, follow external URLs, or open other Dashboards filtered by the value clicked.

This helps you design a seamless workflow that connects your Dashboard to other Cribl or third-party tools and resources, resulting in a smooth data experience for people consuming your Dashboards.

Dashboard Interaction Types

An Interaction controls what happens when you click on a specific data point on your Dashboard. Each Interaction can trigger one of the following actions:

- Run a new search, filling the query with the value clicked and other contextual information.
- Open an external link, enriching the URL with the value clicked or with metadata about the user's Cribl.Cloud Workspace.
- Open another Dashboard, populating one of its Inputs with the value clicked or other context.

Add a Dashboard Interaction

To an Existing Visualization | To a New Visualization | To a New Dashboard

1. Go to the **Dashboards** page in Cribl Search: On the top bar, select **Products** > **Search** > **Dashboards**.
2. Open a Dashboard.
3. Select **Edit**
   at the top right, or press **E** on your keyboard.
4. At the top right of the visualization panel you want to modify, select the **Edit** button. The **Edit Visualization** drawer opens.
5. At the bottom of the drawer, select **Interactions**.
6. Configure the Interaction you want. For more information, see Interaction Types.
7. At the top of the Dashboard, select **Save**.

Run a New Search From a Visualization

You can set up a visualization so that clicking on it runs a new search.

For this, add a **Run a New Search** Interaction:

1. Start adding or editing a Dashboard visualization panel.
2. At the bottom of the **Edit Visualization** drawer, select **Interactions**.
3. From the **Action** drop-down, select **Run a new search**.
4. Enter your search query. (See the examples below.)
5. At the top of the Dashboard, select **Save**.

Your search query can reference the following tokens:

- `$earliest$`
- `$environment.baseUrl$`
- `$environment.type$`
- `$environment.workspace$`
- `$field.name$`
- `$field.value$`
- `$latest$`
- `$rowData.<fieldname>$`
- `$value$`

Example 1 Example 2

If you configure a **Run a new search** Interaction with this query using the $value$ token...

```
dataset="users" email=$value$ | project id, email, last_login
```

Copy

...then clicking on `sam@example.com` triggers this query...

```
dataset="users" email="sam@example.com" | project id, email, last_login
```

Copy

...encoded in this URL:

```
https://<yourWorkspace>-<yourCloudInstance>.cribl.cloud/search/new?q=dataset%3D%22
```

Copy

For an Interaction to pass time parameters, it must either link to a **Time Range** Input directly, or have no time picker at all. If an Interaction links to a parent search that has a time picker, it will fail.

Open a URL From a Visualization

You can set up a visualization so that clicking on it opens an external URL.

For this, add an **Open External Link** Interaction:

1. Start adding or editing a visualization on your Dashboard.
2. At the bottom of the **Edit Visualization** drawer, select **Interactions**.
3. From the **Action** drop-down, select **Open external link**.
4. Enter a URL. (See the example below.)
5. At the top of the Dashboard, select **Save**.

The URL can reference the following tokens:

- `$earliest$`
- `$environment.baseUrl$`
- `$environment.type$`
- `$environment.workspace$`
- `$field.name$`
- `$field.value$`
- `$latest$`
- `$rowData.<fieldname>$`
- `$value$`

Example

If you configure an **Open external link** Interaction with this URL...

```
https://www.virustotal.com/gui/domain/$value$
```

Copy

...then clicking on a data point with the value of `server1` opens this URL:

```
https://www.virustotal.com/gui/domain/server1
```

Copy

Include Your Organization and Workspace in a URL

To construct a URL that includes your Cribl.Cloud Organization and Workspace, use the `$environment.baseUrl$` token.

For example, if you configure an **Open external link** Interaction with this URL...

```
$environment.baseUrl$
```

Copy

...then clicking on the visualization in a Workspace named `main`, in an Organization called `amazing-heady-brcc3nr`, opens this URL:

```
https://main-amazing-heady-brcc3nr.cribl.cloud
```

Copy

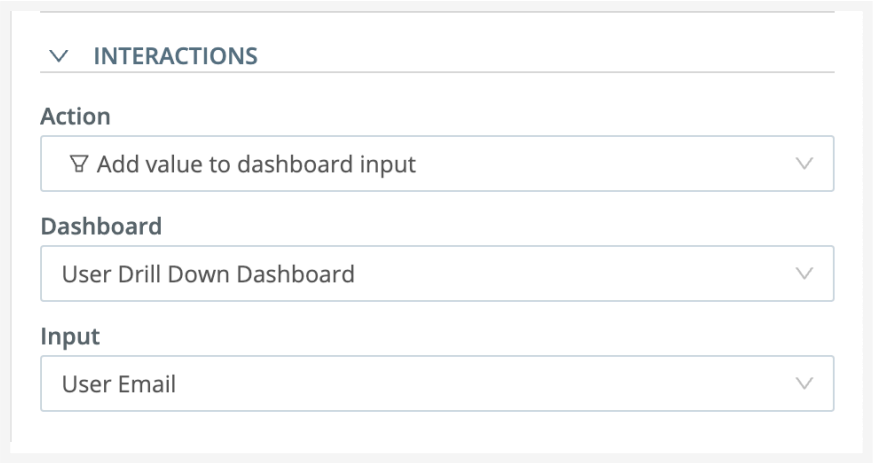
Open Another Dashboard From a Visualization

You can set up a visualization so that clicking on it opens another Dashboard.

You can also pass the value clicked to one of the target Dashboard's Inputs, effectively filtering the new Dashboard by that value.

1. Start adding or editing a visualization on your Dashboard.
2. At the bottom of the **Edit Visualization** drawer, select **Interactions**.
3. From the **Action** drop-down, select **Add value to dashboard input**.
4. From the **Dashboard** drop-down, select the target Dashboard.

5. From the **Input** drop-down, select one of the target Dashboard's Inputs.
6. At the top of the Dashboard, select **Save**.



Interaction with an Input

Dashboard Interaction Tokens Reference

Tokens are placeholders that get replaced with actual values when an Interaction is triggered.

A token can be a **static** value, such as a specific string, or **dynamic**, such as the value of a field that was clicked.

You can use the following tokens in your Interactions:

Token	Type	Description
`$earliest$`	Static	The start timestamp of a time range to use.
`$environment.baseUrl$`	Static	The base URL of the user's Cribl.Cloud Organization.
`$environment.type$`	Static	The type of the user's Cribl.Cloud Workspace (`development`, `staging`, or `production`).
`$environment.workspace$`	Static	The name of the user's Cribl.Cloud Workspace.
`$field.name$`	Dynamic	The name of the field clicked.
`$field.value$`	Dynamic	The value of the field clicked.
`$latest$`	Static	The end timestamp of a time range to use.
`$rowData.<fieldname>$`	Dynamic	The value of the specified field in the clicked row.
`$value$`	Dynamic	The value of the data point clicked.

$earliest$

The start timestamp of a time range to use with the `timestats` operator.

Token Type	Interactions Types Supported
Static	Open an external link Run a new search

For example, if you configure a **Run a new search** Interaction with this query...

```
dataset="cribl_search_sample" | timestats count by host | where _time >= $earliest$
```

Copy

...and the current time range is `earliest=1729290112 latest=1729376512`, clicking the visualization triggers this query...

```
dataset="cribl_search_sample" | timestats count by host | where _time >= 1729290112
```

Copy

...encoded in this URL:

```
https://<yourWorkspace>-<yourCloudInstance>.cribl.cloud/search/new?q=dataset%3D%22cri
```

Copy

$environment.baseUrl$

The fully qualified base URL of the Cribl.Cloud Organization in which the user triggered the Interaction.

Token Type	Interactions Types Supported
Static	Open an external link Run a new search

The base URL includes:

- `https://` prefix
- Workspace name (`$environment.workspace$`)

- Organization ID (e.g., `amazing-heady-brcc3nr`)
- Environment type (`$environment.type$`)
- `cribl.cloud` suffix

For example, if you configure an **Open external link** Interaction with this URL...

```
$environment.baseUrl$/status
```

Copy

...for an Organization with the base URL `https://main-amazing-heady-brcc3nr.cribl.cloud`, the Interaction opens this URL:

```
https://main-amazing-heady-brcc3nr.cribl.cloud/status
```

Copy

`$environment.type$`

The type of the Cribl.Cloud Workspace in which the user triggered the Interaction. Possible values are: `development`, `staging`, or `production`.

Token Type	Interactions Types Supported
Static	Open an external link Run a new search

For example, if you configure an **Open external link** Interaction with this URL...

```
status.$environment.type$.example.com
```

Copy

...for a Workspace of type `production`, the Interaction opens this URL:

```
status.production.example.com
```

Copy

$environment.workspace$

The name of the Cribl.Cloud Workspace in which the user triggered the Interaction.

Token Type	Interactions Types Supported
Static	Open an external link Run a new search

For example, if you configure a **Open external link** Interaction with this URL...

```
dashboards.example.com/$environment.workspace$
```

Copy

...for a Workspace named `marketing`, the Interaction opens this URL:

```
dashboards.example.com/marketing
```

Copy

$field.name$

The name of the field clicked.

Token Type	Interactions Types Supported
Dynamic	Open an external link Run a new search

For example, if you configure a **Run a new search** Interaction with this query...

```
dataset="cribl_search_sample" | where $field.name$="foo"
```

Copy

...then clicking on a field named `host` triggers this query...

```
dataset="cribl_search_sample" | where host="foo"
```

Copy

...encoded in this URL:

```
https://<yourWorkspace>-<yourCloudInstance>.cribl.cloud/search/new?q=dataset%3D%22cri
```

Copy

$field.value$

The value of the field clicked.

Token Type	Interactions Types Supported
Dynamic	Open an external link Run a new search

For example, if you configure a **Run a new search** Interaction with this query...

```
dataset="cribl_search_sample" | where status=$field.value$
```

Copy

...then clicking on a field with the value `200` triggers this query...

```
dataset="cribl_search_sample" | where status="200"
```

Copy

...encoded in this URL:

```
https://<yourWorkspace>-<yourCloudInstance>.cribl.cloud/search/new?q=dataset%3D%22cri
```

Copy

$latest$

The end timestamp of a time range to use with the `timestats` operator.

Token Type	Interactions Types Supported
Static	Open an external link Run a new search

For example, if you configure a **Run a new search** Interaction with this query...

```
dataset="cribl_search_sample" | timestats count by host | where _time <= $latest$
```

Copy

...and the current time range is `earliest=1729290112 latest=1729376512`, clicking the visualization triggers this query...

```
dataset="cribl_search_sample" | timestats count by host | where _time <= 1729376512
```

Copy

...encoded in this URL:

```
https://<yourWorkspace>-<yourCloudInstance>.cribl.cloud/search/new?q=dataset%3D%22cri
```

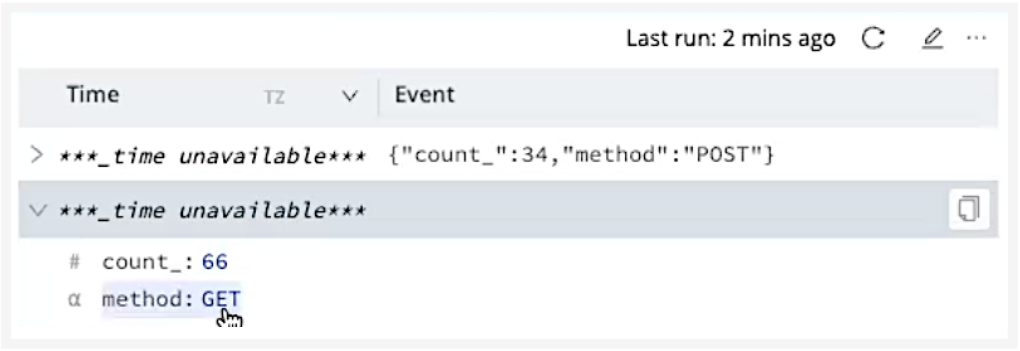
Copy

`$rowData.<fieldname>$`

The value of a specific field in the clicked row (regardless of which row was actually clicked). Replace `<fieldname>` with a real field name.

Token Type	Interactions Types Supported
Dynamic	Open an external link Run a new search

Example



A visualization panel of the **Events** type

If you configure the visualization above with a **Run a new search** Interaction like this...

```
dataset="cribl_search_sample" | where method=$rowData.method$ & count=$rowData.count_
```

Copy

...then no matter whether the user selects `count\_: 66` or `method: GET`, the Interaction runs a query with both...

```
dataset="cribl_search_sample" | where method=GET & count=66
```

Copy

...encoded in this URL:

```
https://<yourWorkspace>-<yourCloudInstance>.cribl.cloud/search/new?q=dataset%3D%22cri
```

Copy

$value$

The value of the data point clicked.

Token Type	Interactions Types Supported
Dynamic	Open an external link Run a new search

For example, if you configure a **Run a new search** Interaction with this query...

```
dataset="cribl_search_sample" | where host=$value$
```

Copy

...then clicking on a data point with the value of `server1` triggers this query...

```
dataset="cribl_search_sample" | where host="server1"
```

Copy

...encoded in this URL:

```
https://<yourWorkspace>-<yourCloudInstance>.cribl.cloud/search/new?q=dataset%3D%22cri
```

Copy

4.2.4. Add Inputs To Your Cribl Search Dashboard

Enable Dashboard viewers to control visualizations through interactive widgets.

Why Use Dashboard Inputs

An Input is a small interactive area at the top of your Dashboard. When a user selects or enters values, the linked visualizations update to reflect the new criteria. This allows Dashboard viewers to quickly filter data, without needing to directly edit the underlying search queries.

Dashboard Input Types

Using different types of Inputs, you can enable Dashboard viewers to:

- Change the time range of visualizations, with a **Time Range** Input.
- Filter visualizations by freeform text, with a **Text** Input.
- Filter visualizations by numbers, with a **Number** Input.
- Filter visualizations by a preconfigured set of dynamic values, with a **Dropdown** Input.



Different types of Dashboard Inputs

Add a Dashboard Input

To add an Input to a new or existing Dashboard:

1. Go to the **Dashboards** page in Cribl Search: On the top bar, select **Products** > **Search** > **Dashboards**.
2. In an open Dashboard, select **Edit** at the top right, or press **E** on your keyboard.
3. At the top right, select **Add** > **Input**. Give the Input a title.
4. Set the Input ID. You'll reference this ID in the searches that you want the Input to control. Use alphanumeric characters, underscores, and hyphens only.
5. In the **Type** drop-down, select the Input type you want. See the details about each type:
    - Time Range
    - Text
    - Number
    - Dropdown
6. When done, select **Save** at the top of the Dashboard. (To skip this step, see Auto-Apply Input Changes.)

Now, link the Input to one or more visualizations in the Dashboard.

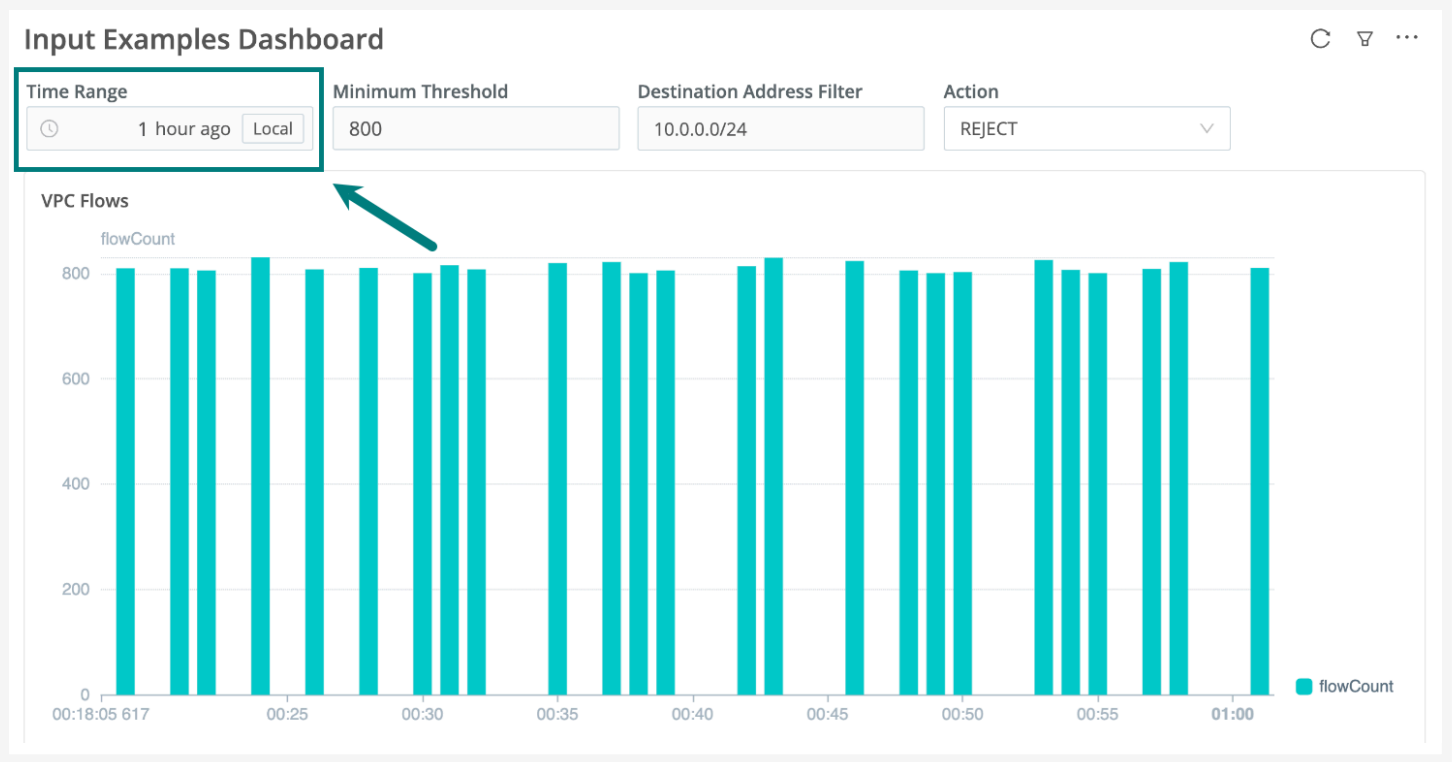
Link a Dashboard Input To a Visualization

After adding an Input to a Dashboard, you need to link it to one or more visualization panels. This way, when a user interacts with the Input, the linked visualizations update accordingly.

1. Add an Input to your Dashboard.
2. Add or edit a visualization panel that you want to control.
3. Depending on the Input type, follow the specific linking instructions:
    - Time Range
    - Text
    - Number
    - Dropdown

Enable Viewers to Change a Visualization's Time Range

Using a **Time Range Input**, you can enable viewers to control the time range of the visualization panels with a time picker at the top of your Dashboard.



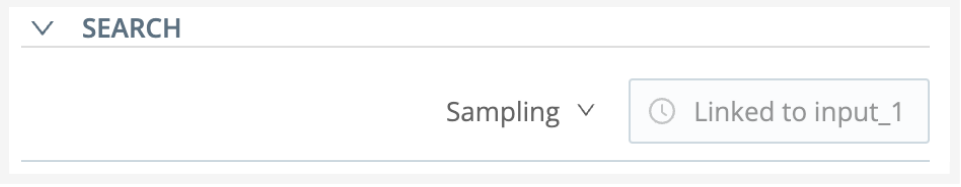
A Time Range Input

First, add a **Time Range Input**:

1. In an open Dashboard, select **Edit** at the top right, or press **E** on your keyboard.
2. At the top right, select **Add** > **Input**. Give the Input a title.
3. Set the Input ID. Use alphanumeric characters, underscores, and hyphens only.
4. In the **Type** drop-down, select **Time Range**.
5. Set the time picker's **Default value**. This is the time range applied when the Dashboard first loads.
6. When done, select **Save** at the top of the Dashboard. (To skip this step, see Auto-Apply Input Changes.) The new Input appears at the top of your Dashboard, where you can edit, clone, or delete it.

Now, link your Time Range Input to one or more visualizations:

1. Add or edit a visualization panel that you want to control.
2. Select the clock button in the date and time selector.
3. Select the **Link** tab.
4. Select the Time Range Input you just created. The Input ID appears in the visualization's date and time field.

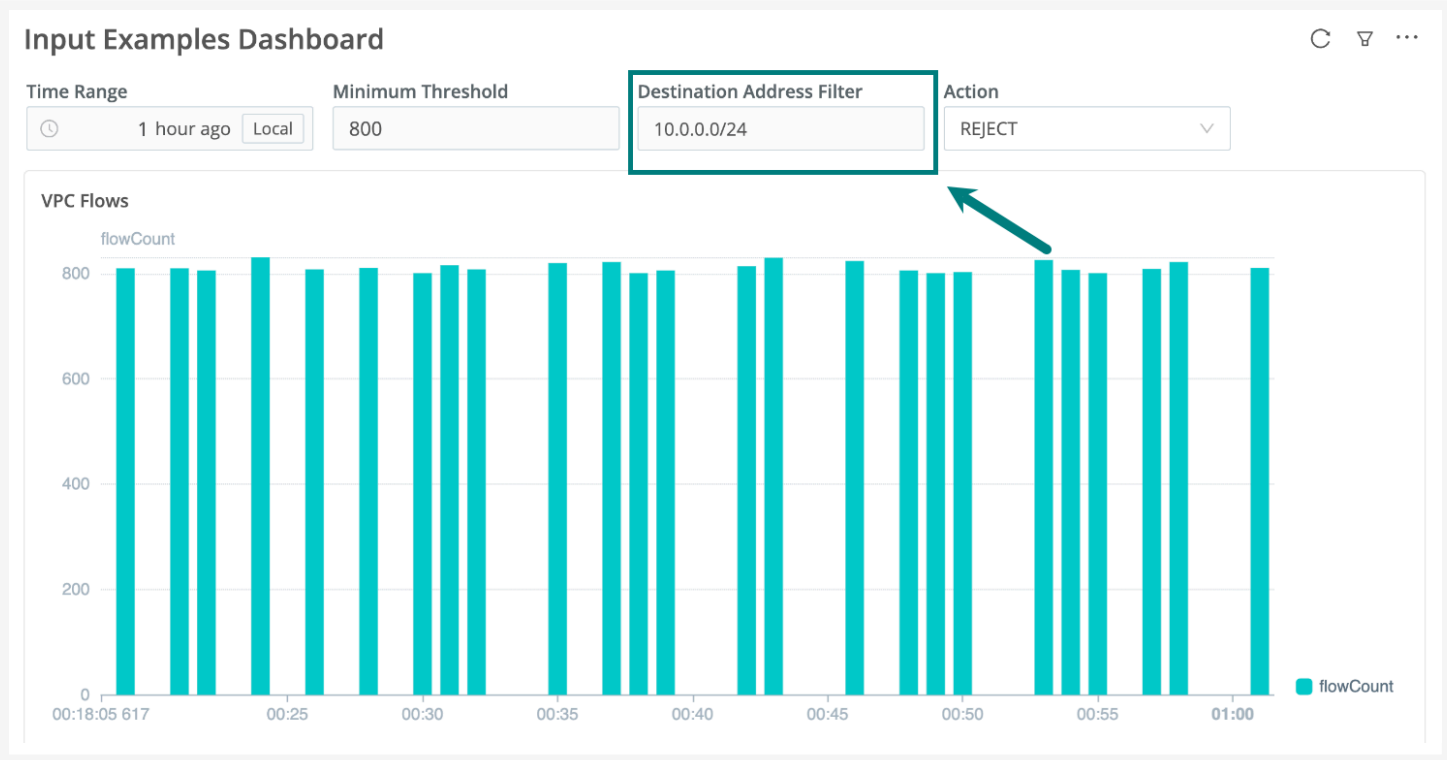


Time range with linked Input

Now, when viewers select a time range in the Input, the linked visualizations update.

Enable Viewers to Filter a Visualization by Text

Using a **Text Input**, you can enable Dashboard viewers to enter freeform text so that linked visualization panels display only data with that text value. Think of this as a free-text search box.



A Text Input

First, add a **Text Input**:

1. In an open Dashboard, select **Edit** at the top right, or press **E** on your keyboard.
2. At the top right, select **Add** > **Input**. Give the Input a title that gives the Dashboard viewer an idea of what they can type.
3. Set the Input ID. Use alphanumeric characters, underscores, and hyphens only.
4. In the **Type** drop-down, select **Text**.
5. Set the text field's **Default value**. This is the filter applied when the Dashboard first loads.
6. When done, select **Save** at the top of the Dashboard. (To skip this step, see Auto-Apply Input Changes.) The new Input appears at the top of your Dashboard, where you can edit, clone, or delete it.

Then, link your Time Range Input to one or more visualizations:

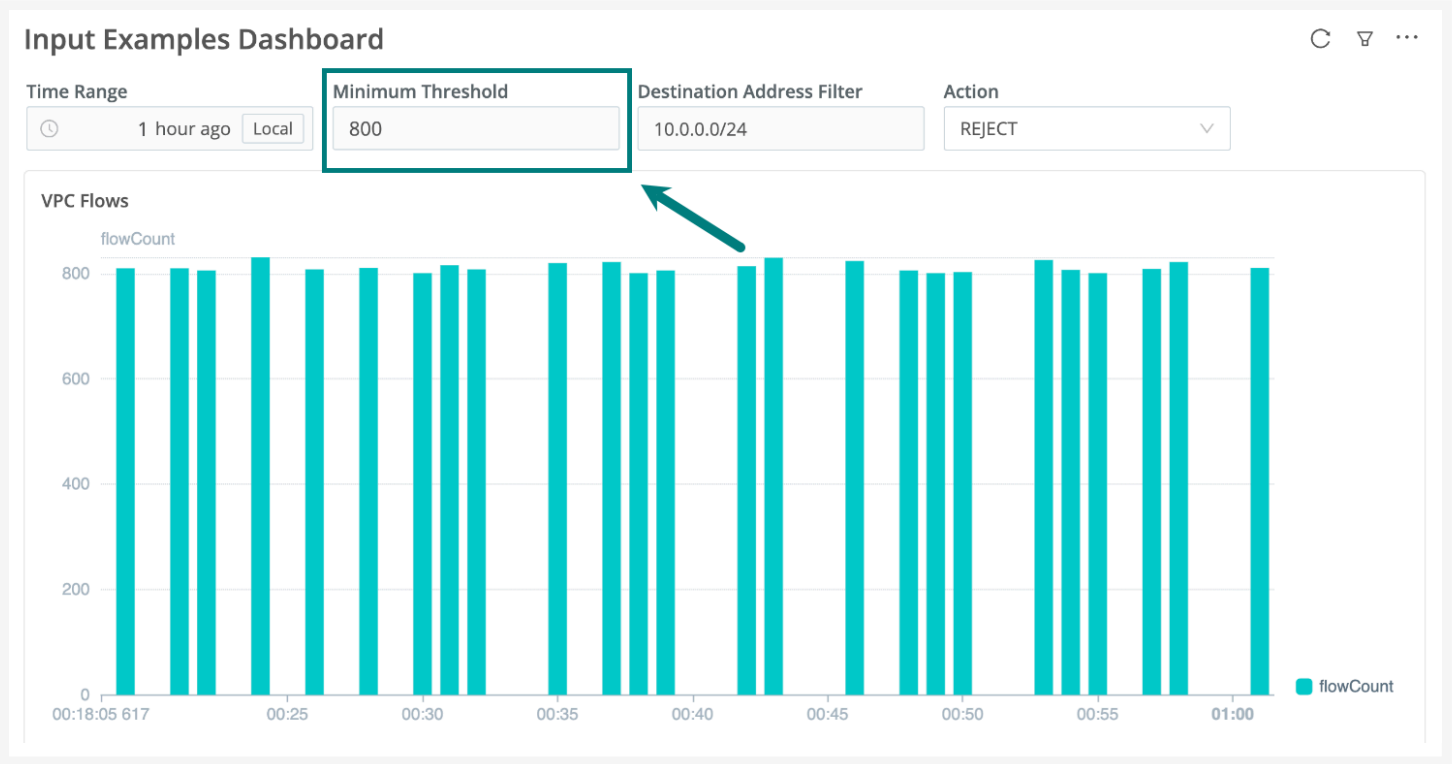
1. Add or edit a visualization panel that you want to filter.

2. In the visualization's search query, reference the **Input ID** of the Text Input you just created. Surround the ID by `$`. For example, if the Input ID is `firstname`, use `$firstname$`.

Now, when viewers type into the Input's text field, the linked visualizations update.

Enable Viewers to Filter a Visualization by Numbers

Using a **Number Input**, you can enable Dashboard viewers to filter visualizations by a number they enter into an simple text area at the top of your Dashboard.



A Number Input

First, add a **Number Input**:

1. In an open Dashboard, select **Edit** at the top right, or press **E** on your keyboard.
2. At the top right, select **Add** > **Input**. Give the Input a title that gives the Dashboard viewer an idea of what they can type.
3. Set the Input ID. Use alphanumeric characters, underscores, and hyphens only.
4. In the **Type** drop-down, select **Number**.
5. Set the number field's **Default value**. This is the filter applied when the Dashboard first loads.
6. You can set the **Minimum value** and **Maximum value**, to restrict what viewers can enter.
7. When done, select **Save** at the top of the Dashboard. (To skip this step, see Auto-Apply Input Changes.) The new Input appears at the top of your Dashboard, where you can edit, clone, or delete it.

Then, link your Number Input to one or more visualizations:

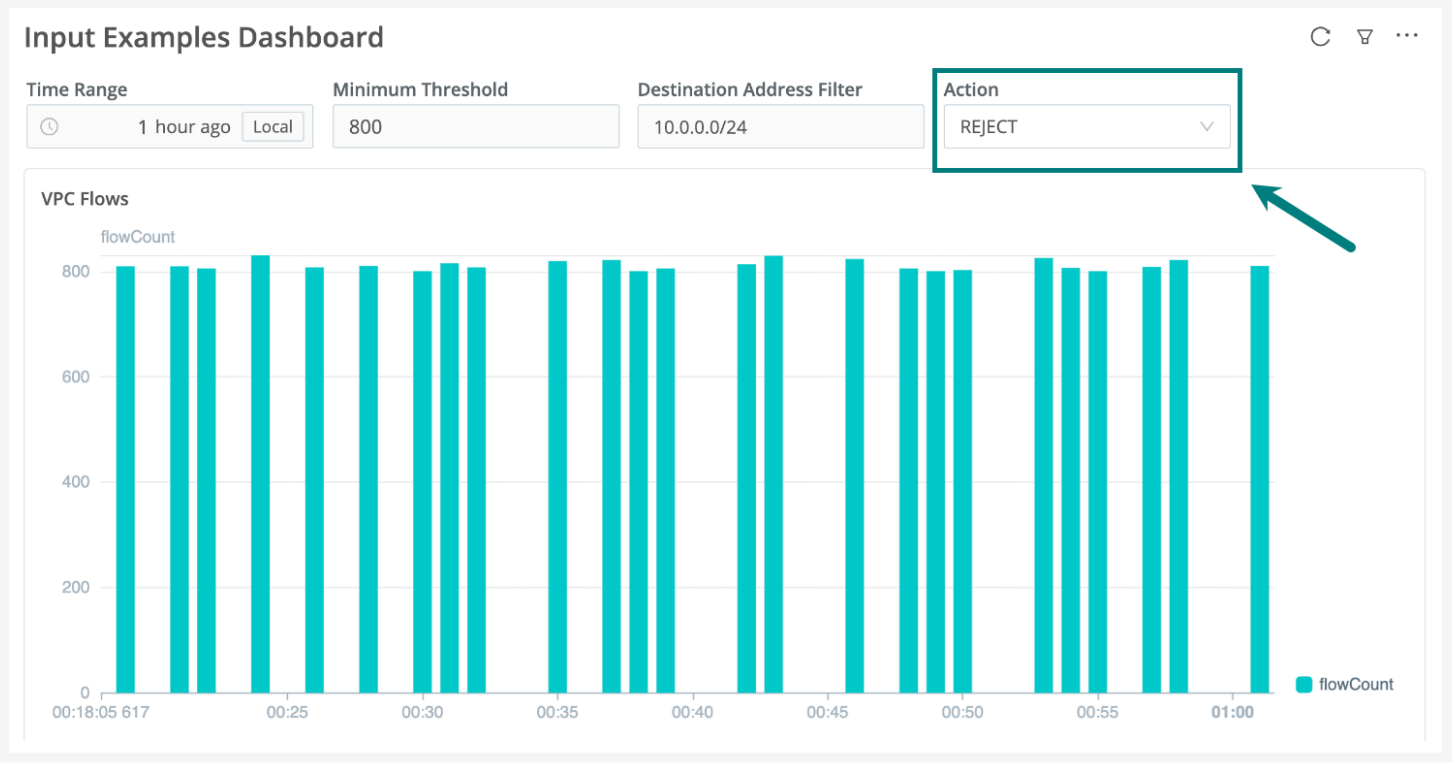
1. Add or edit a visualization panel that you want to filter.
2. In the visualization's search query, reference the **Input ID** of the Number Input you just created. Surround the ID by `$`. For example, if the Input ID is `value`, use `$value$`.

Now, when viewers type into the Input's number field, the linked visualizations update.

Enable Viewers to Filter a Visualization by Drop-Down Options

Using a **Dropdown Input**, you can enable Dashboard viewers to filter visualizations by selecting one or more values from a drop-down list at the top of your Dashboard.

The list of values can be static (where you configure all options available to the user), or dynamic (where the results of a search provide the values for the drop-down).



A Dropdown Input

First, add a **Dropdown Input**:

1. In an open Dashboard, select **Edit** at the top right, or press **E** on your keyboard.
2. At the top right, select **Add** > **Input**.
3. Set the Input ID. Use alphanumeric characters, underscores, and hyphens only.
4. In the **Type** drop-down, select **Dropdown**.

5. Select how to populate the drop-down list: **New search**, **Saved search**, or **Manually add dropdown items**.
6. In the **Field name** (with the search options) or in the **Values** (with the manual option), enter a caption to help the viewer understand the item's purpose.
7. When done, select **Save** at the top of the Dashboard. (To skip this step, see Auto-Apply Input Changes.) The new Input appears at the top of your Dashboard, where you can edit, clone, or delete it.

Then, link your Dropdown Input to one or more visualizations:

1. Add or edit a visualization panel that you want to control.
2. In the visualization's search query, reference the **Input ID** of the Number Input you just created. Surround the ID by `$`. For example, if the Input ID is `firstname`, use `$firstname$`.

Auto-Apply Input Changes

You can auto-apply your Dashboard Inputs changes, without having to confirm them. To set this up for a Dashboard:

1. Go to the **Dashboards** page in Cribl Search: On the top bar, select **Products** > **Search** > **Dashboards**.
2. Open a Dashboard.
3. At the top right of the Dashboard, select the Actions ⋯ button.
4. Select **Apply Input Changes Automatically**.

Goat-to Resources

Building Dashboard and Dashboard Inputs in Cribl Search — Blog

4.2.5. Add Visualizations Using Cribl Copilot

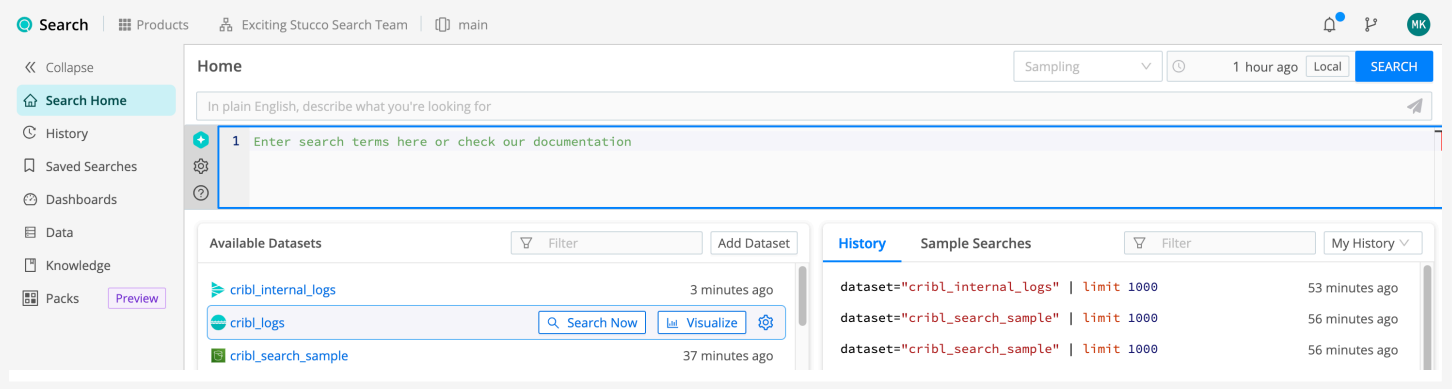
Use the Cribl Copilot AI assistant to generate Dashboard visualizations.

As an alternative to designing Dashboard visualizations from scratch, you can use the Cribl Copilot visualization assistant to analyze your Dataset and suggest effective display options. You can then adapt one of the initial suggestions, request different suggestions, or reject all suggestions and configure your own visualization.

To globally enable or disable Cribl Copilot for all Cribl products in your Organization, select **Settings** > **Global** > **AI Settings**.

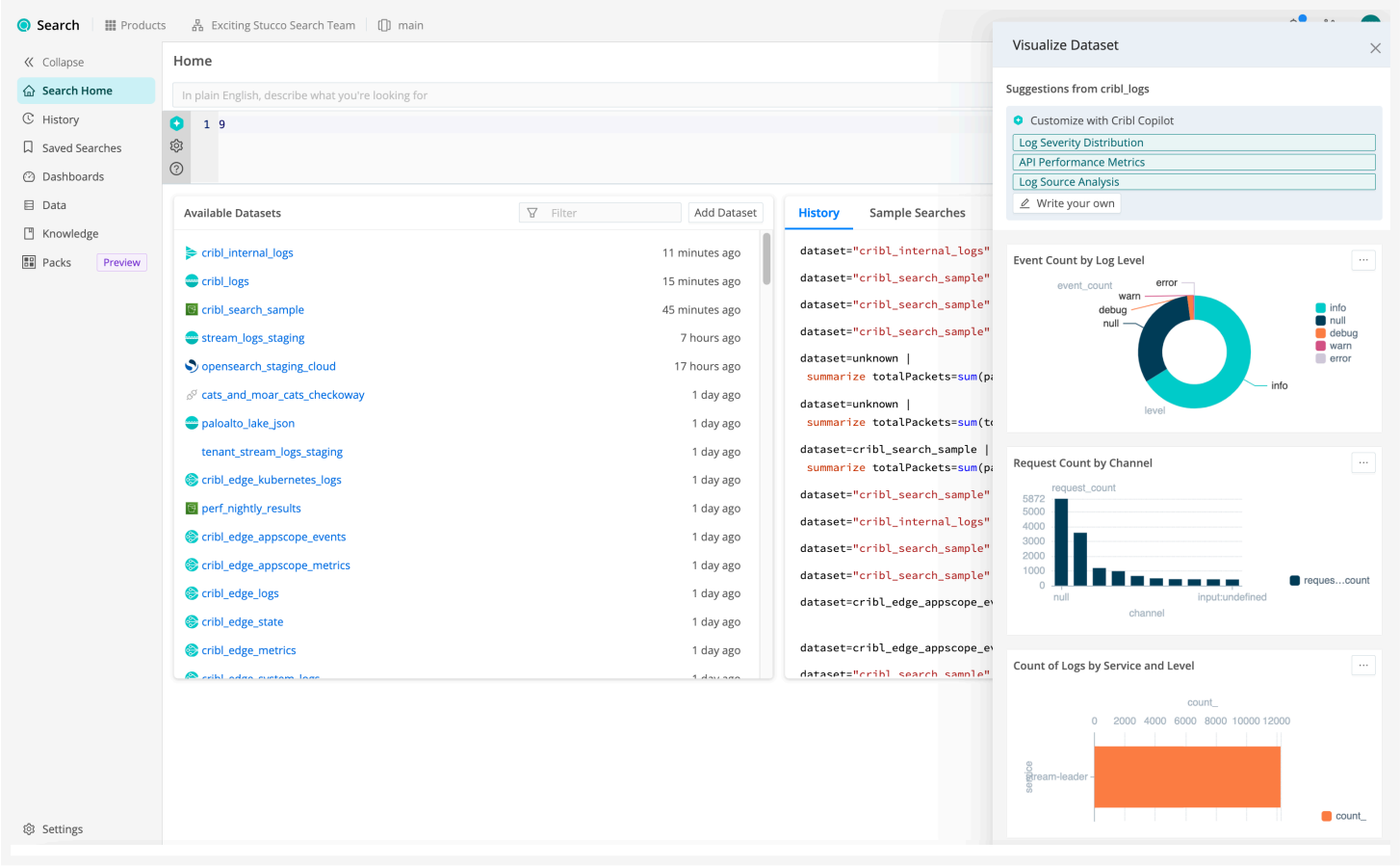
Add a Visualization from a Dataset

Cribl Copilot can suggest visualizations directly from the **Search Home** page. Hover your pointer over any Dataset to expose a **Visualize** button, as shown below.



Cribl Copilot Visualize button

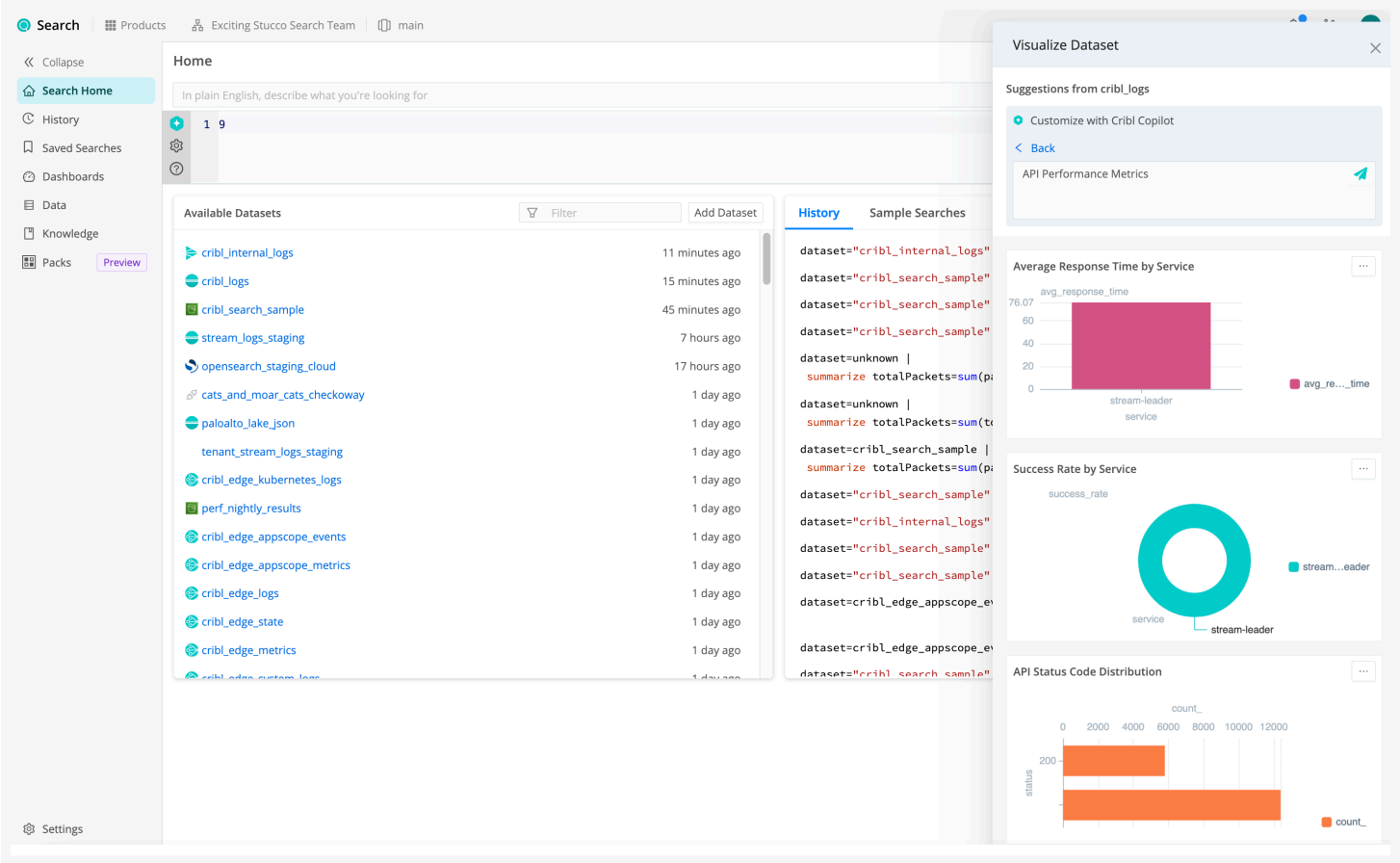
Select the button, and a **Visualize Dataset** drawer will open on the right. Here, Cribl Copilot will analyze the selected Dataset. After a few seconds, if the analysis is successful, the drawer will suggest three visualizations that might be useful in displaying the discovered data. Above the visualization panels, you will also see several buttons to generate alternative queries.



Suggested visualizations for your Dataset

You have the following options here:

- To accept one of the suggested visualizations, select its Actions ⋯ button (upper right), then select **Add to Dashboard**.
- To substitute a new query and generate a different set of visualizations, select one of the three buttons at the top right. As shown below, this will open a corresponding natural language prompt in a new query box. Refine the query as desired, then select the Send button – or press `Enter/return` – to see your new visualization options.
- Select **Write your own**. This will open a similar query box, where you can describe your own natural language query from scratch. As with the predefined query buttons, this will generate new suggested visualizations.

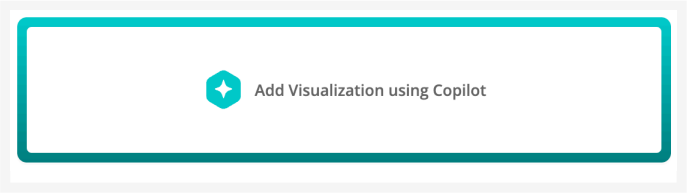


Adapt a new query to suggest new visualizations

Add a Visualization when Creating a Dashboard

When you add a new Dashboard, you can use Cribl Copilot to generate suggested visualizations.

1. From the Cribl Search sidebar, select **Dashboards**.
2. Select **Add Dashboard**.
3. Give your Dashboard a name, configure any additional settings you want, and then select **Save**.
4. Select **Add Visualization Using Copilot**.



Add Visualization Using Copilot button

5. Select the Dataset you want to use as your visualization's base.

   Soon, Cribl Copilot will display three recommended visualizations. (This could take up to several minutes with large Datasets.)

6. Select **Add** to add any suggested visualization to your new Dashboard, or select **Refresh** ↻ to generate three new visualizations.
7. You're now working in edit mode (described in the following section).

   To add more visualizations, select the Cribl Copilot button near the upper right.

Add a Visualization when Editing a Dashboard

When you're editing an existing Dashboard, you can use Cribl Copilot to generate suggested visualizations.

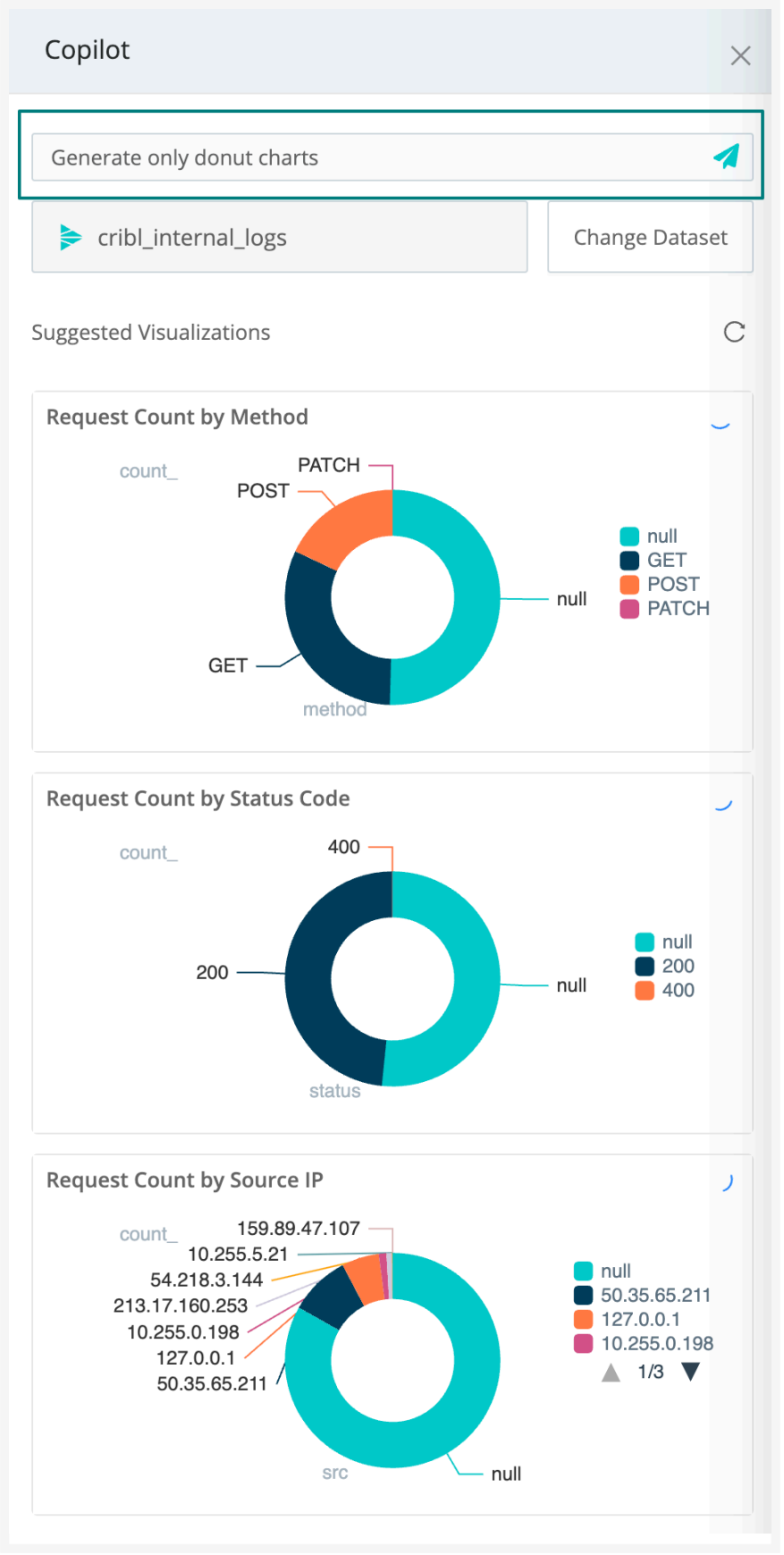
1. From the Cribl Search sidebar, select **Dashboards**.
2. Select the Dashboard you want to edit.
3. Select the Actions ⋯ button at the upper right, and then select **Edit**.
4. Near the upper right, select the Cribl Copilot button .

   If the Dashboard is empty, you'll instead see the **Add Visualization Using Copilot** option, described in the preceding section.
5. From the list on the right, select the Dataset you want to use as your visualization's base.

   Soon, Cribl Copilot will display three recommended visualizations. (This could take up to several minutes with large Datasets.)
6. Select **Add** to add any visualization to your new Dashboard, or select **Refresh** ↻ to generate three new suggested visualizations.

You have two additional options when editing an existing Dashboard:

- To replace the Dataset used for your visualization, select **Change Dataset**.
- To refine your results, describe your preferences in everyday language in the text box at the top of the Copilot drawer.



Sample prompt to suggest only donut charts

For more information about working with Dashboards in Cribl Search, see Create a Dashboard and Edit a Dashboard.

4.2.6. Manage And Share Dashboards

Collect, export, and share your Cribl Search Dashboards.

You can create custom Collections to group Dashboards together. You can export Dashboard visualization panels as image files. And you can share any Dashboard that you own.

Group Dashboards into Collections

You can create custom Collections in which to organize groups of Dashboards.

On the Dashboards page, select **Add Collection**, give your Collection a unique name, and then select **Add**. New Collections appear in the list on the left, in alphabetical order.

You can also create a Collection when creating a new Dashboard.

Add Dashboards to Collections by selecting a Collection name from the **Collection** drop-down in either the **Edit Dashboard** modal (when editing an existing Dashboard) or the **New Dashboard** modal (when adding a new Dashboard).

Move a Dashboard that you own by clicking the Dashboard's Actions ⋯ button and selecting **Move**.

Rename or delete a Collection by clicking the button next to the Collection name.



Moving a Dashboard generally discards any sharing Permissions that were set on the original Dashboard. (However, moving a Dashboard **between** Collections does not change its access Permissions.) Share the Dashboard from its new location to add new Permissions.

Export a Visualization

To export a visualization panel as a `.jpg` or `.png` file, select the Actions ⋯ button at the top right, and then choose the file type you want to export.

Share a Dashboard

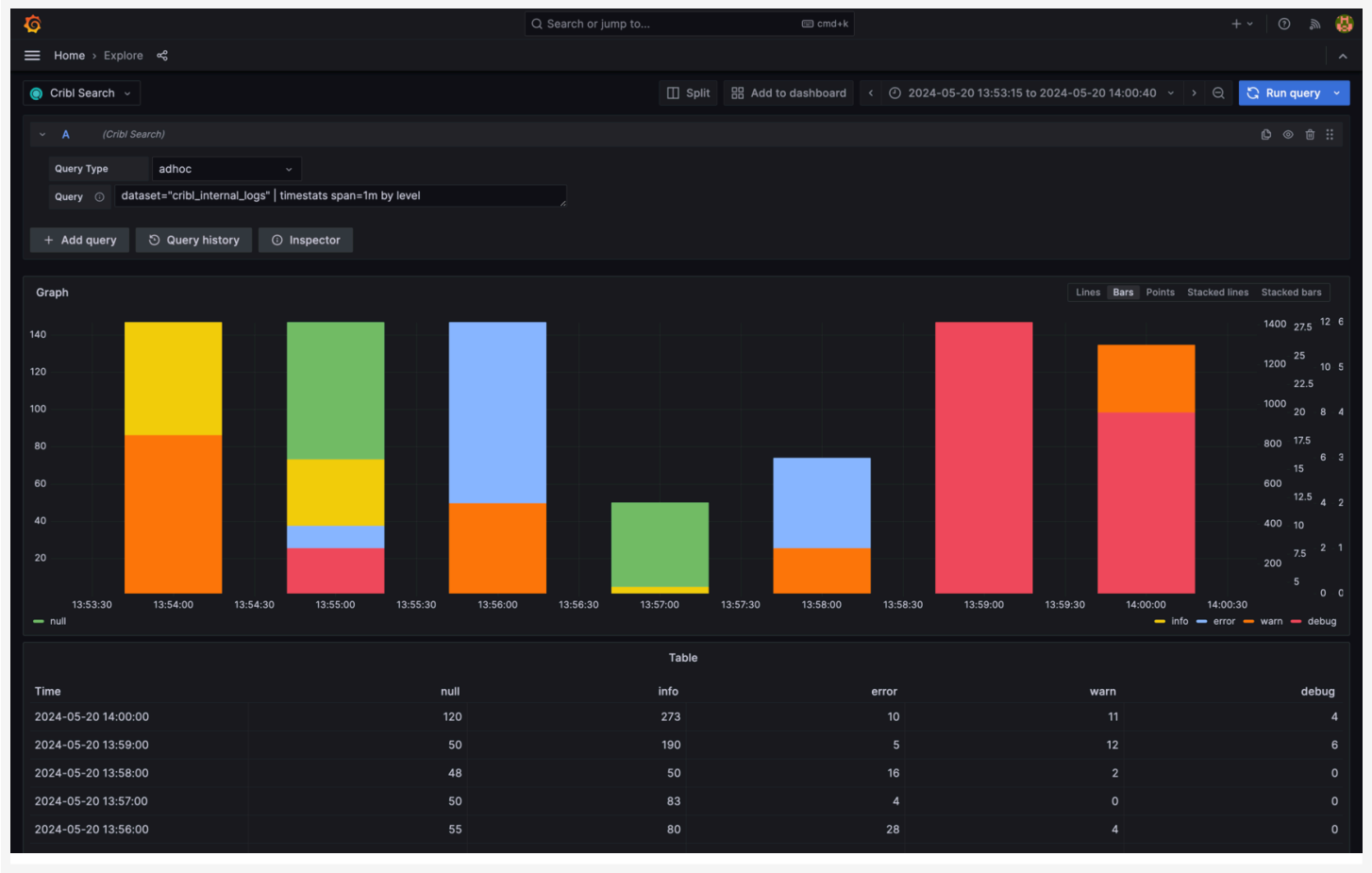
You can share any Dashboard that you own by clicking the Dashboard's Actions ⋯ button, and selecting **Share**.

4.3. Third-Party Visualization Tools

4.3.1. Use Cribl Search With Grafana

Visualize your search results in Grafana, using the official Cribl Search plugin.

In addition to the Charts and Dashboards included in Cribl Search, you might want to present your search results using Grafana's visualization tools. For that, use the official **Cribl Search plugin for Grafana**.



Cribl Search plugin for Grafana

Once the plugin is configured, you can:

- Run any Cribl Search query directly in Grafana.
- Feed your Grafana dashboards with up-to-date data, using scheduled searches.

Connect Grafana to Cribl Search

To start using Cribl Search with Grafana, you need Organization administrator permissions in Grafana, and Owner or Admin permissions in Cribl.Cloud.

1. In Grafana, install the Cribl Search plugin.
2. In Cribl.Cloud, get a Cribl.Cloud credential.
3. In Grafana, add a Cribl Search data source.

Install the Cribl Search Plugin in Grafana

To install the Cribl Search plugin in Grafana:

1. Log in to Grafana as an Organization administrator.
2. Go to the Cribl Search plugin page, and select **Install**.

Create a Cribl.Cloud Credential for Grafana

To let Grafana pull data from Cribl Search, create an API Credential in Cribl.Cloud:

1. Log in to Cribl.Cloud, and select an Organization for which you're an Owner or Admin.
2. Go to **Account** > **Organization** > **API Management**.
3. Select **Add Credential**, and follow the steps to create an API Credential.
4. Once you created the credential, keep the following at hand:
    - Client ID.
    - Client Secret.
    - The URL of your Cribl.Cloud Organization (for example, `https://<workspaceName>-<organizationId>.cribl.cloud`).

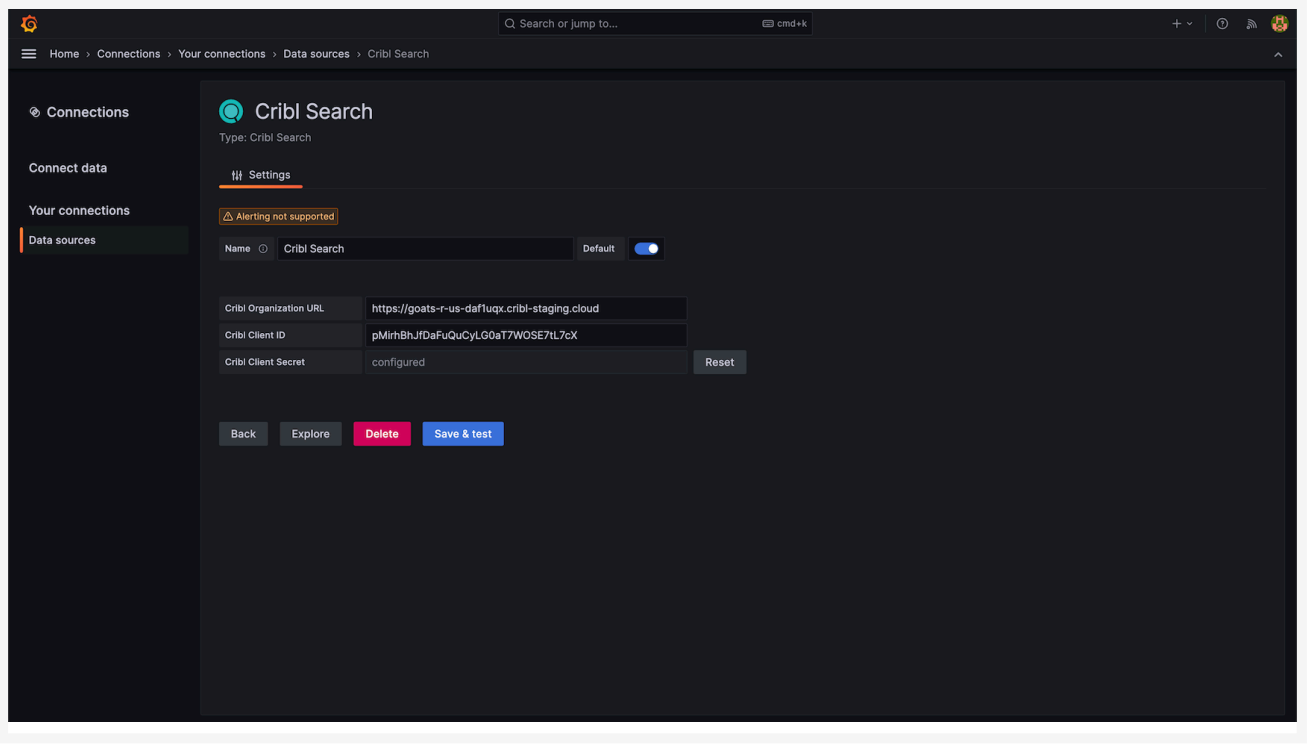
Add a Cribl Search Data Source in Grafana

With the Cribl Search plugin installed, and your Cribl.Cloud credential ready, add a Cribl Search data source in Grafana:

1. Log in to Grafana as an Organization administrator.
2. In Grafana, follow Grafana's instructions for adding a new data source. Select the Cribl Search plugin as the data source type.
3. On the Cribl Search plugin page:
    - In **Cribl Organization URL**, enter the URL of your Cribl.Cloud Organization (for example, `https://<workspaceName>-<organizationId>.cribl.cloud`).
    - In **Cribl Client ID**, enter the Client ID of the Cribl.Cloud credential.

- In **Cribl Client Secret**, enter the Client Secret of the Cribl.Cloud credential.

4. Select **Save & test**.



Cribl Search plugin configuration

Run a Cribl Search Query in Grafana

Once you set up the Cribl Search plugin, you can run any Cribl Search query right there in Grafana:

1. In Grafana, go to your Cribl Search data source.
2. In **Query Type**, select **adhoc**.
3. In **Query**, enter your Cribl Search query.

> Your query can include Grafana variables.

4. At the top right, adjust the time range.
5. Select **Run query**.

Schedule Searches to Load Grafana Dashboards Faster

To render your Grafana dashboards faster, keep them up to date by automatically pulling the latest results of scheduled searches from Cribl Search.

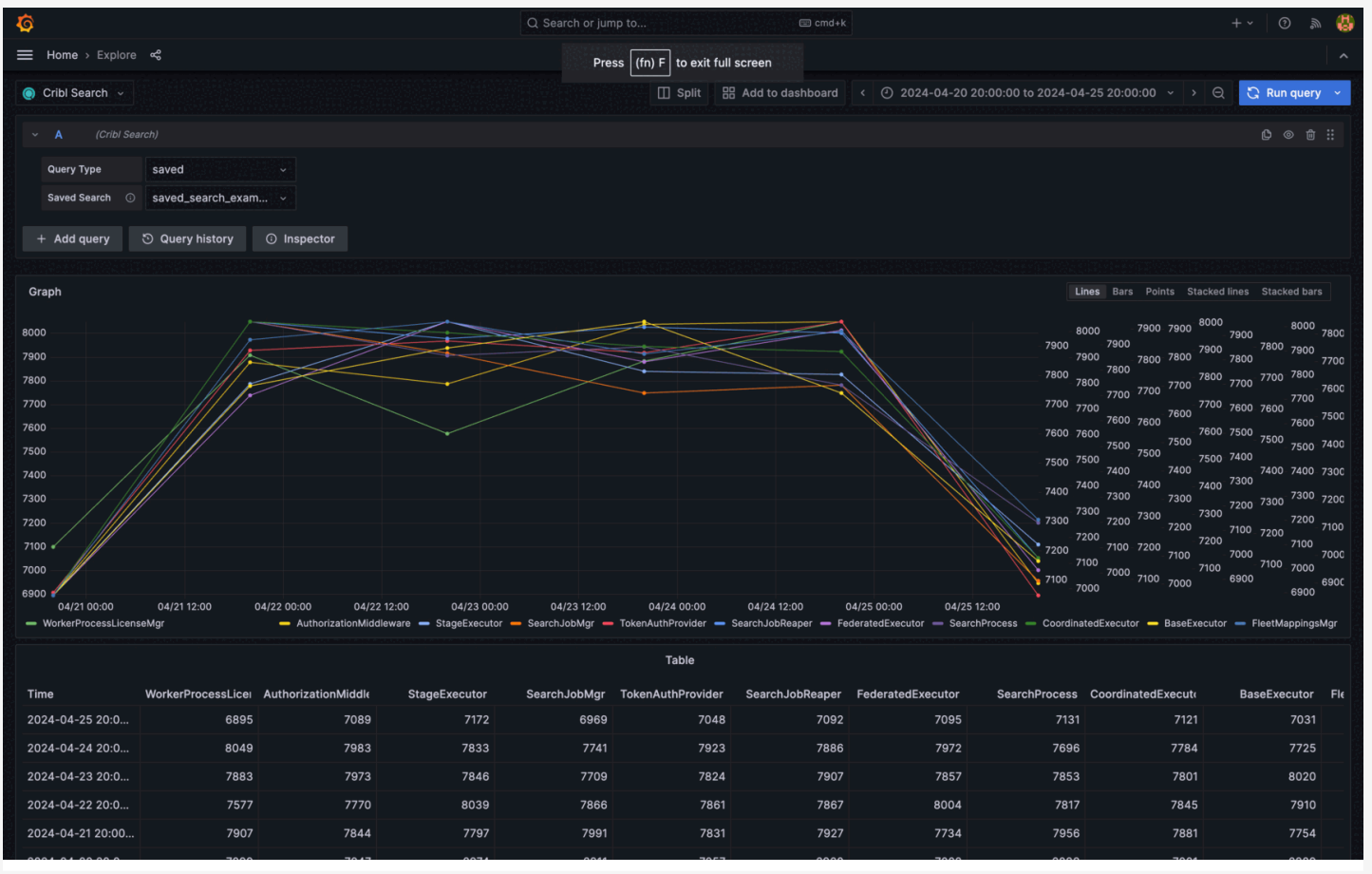
First, get the ID of the scheduled search that you want to visualize in Grafana:

1. From the Cribl Search sidebar, select **Saved Searches**.
2. Open an existing saved search, or select **Add Search** to save a new one.
3. Set up your scheduled search, by following this section: Schedule a Search.
4. At the top, copy the scheduled search **ID** (for example, `cribl_search_finished_1h`).

With the scheduled search ID ready, go to Grafana:

1. In Grafana, go to your Cribl Search data source.
2. In **Query Type**, select **saved**.
3. In **Saved Search**, enter the ID of your Cribl Search scheduled search.
4. Select **Run query**.

Now, Grafana will automatically fetch the latest results of your scheduled search. When you open Grafana, the dashboard dependent on your search will load immediately, since your Cribl Search results are already there.



Pulling scheduled search results in Grafana

Get Grafana Alerts Based on Cribl Search Results

You can create Grafana alert rules based on results produced by Cribl Search. For example, you can get Grafana alerts when the number of your Cribl Search results crosses a certain threshold, or when a result set

contains a specific value.

1. Make sure your Grafana instance is connected to Cribl Search.
2. When configuring an alert rule in Grafana, select a Cribl Search data source, and add your ad-hoc or scheduled Cribl Search query.

For more information on setting up alerts, see Grafana's documentation.

5. Send Alerts With Cribl Search

Receive conditional Notifications about your results, using scheduled searches and Notification targets.

Scheduled Searches

Schedule a search to run at a specific frequency.

Notifications

Set up a Notification target to send alerts about your scheduled searches to:

- System Messages: Banner messages in the Cribl Search UI (the default target).
- Email messages.
- Amazon SNS topics.
- PagerDuty accounts.
- Slack channels.
- Webhook connections.

5.1. Scheduled Searches

Run a search at a scheduled frequency, and set up conditional Notifications.

Scheduled searches allow you to automate data analysis and save valuable time. By scheduling saved searches you can effectively monitor systems, optimize workflows, and send Notifications based on the evaluation of search results against a specific condition. Use scheduled searches to aggregate data, compare results, identify anomalies, and analyze long-term trends.

For example, instead of manually running a search to gather information about login types and failures from the previous month, you can schedule a search to automatically aggregate this data at midnight on the first day of each month.

Scheduled searches can also send Notifications, based on specific conditions. This helps you stay informed about important events with no manual effort. You can send Notifications to different targets, including:

- System Messages: Banner messages in the Cribl Search UI (the default target).
- Email messages.
- Amazon SNS topics.
- PagerDuty accounts.
- Slack channels.
- Webhook connections.

Schedule a Search

To schedule a search, save or edit an existing saved search, and then toggle **Schedule** on. The option to save a search is available in two places:

- At the bottom right of the query box, under the **Actions** menu.
- In **Saved Searches**, through the **Add Search** button.

Either way, the resulting modal provides these configuration options:

Name: Give the search a meaningful name that will help you identify it easily.

Description: A description is optional.

Results Access: Defaults to **Private** where only you can view the result set. Select **All Members** to share the results. When a result set is created its access level is defined. If you edit this setting, only subsequent results will have the new setting.

You can also use the Cribl API to save and schedule searches. Read Save Searches and Get Alerts for more information.

Query

Search string: This is the query; edit it if needed. For details, see Build a Search.

- **Sampling**: Sampling uses a ratio to reduce the number of results returned from a search. For example, if a search returns 1,000 events, a 1:10 sampling ratio returns 100 events.
- **Time range**: Set the query time range, relative to the time the search is scheduled to run. See time range for details.

Schedule

Schedule: Enable **Run on schedule** to configure the run frequency. You can then set the frequency either by using the drop-downs, or by entering a custom cron expression. The cron expression defaults to `0 0 * * *`, running every day at 12:00 AM.

- **Timezone**: Select the TZ for the scheduled search to use. Defaults to UTC.

Notifications

For details about setting options on the **Notifications** tab, see Configure Notifications.

Advanced

Advanced provides a **Keep last executions** control. This determines the minimum number of result sets stored. Every hour, Cribl Search rotates result sets, preserving at least this number of past results. The minimum allowed is `1`, the default is `2`, and the maximum is `1000`. This setting overrides the **Search history maximum jobs** and **Search history TTL** limits.

Rules

- You can edit the schedule or query of a scheduled search and it will be used on subsequent runs.
- Daylight savings time is automatically incorporated based on the selected time zone.

Results

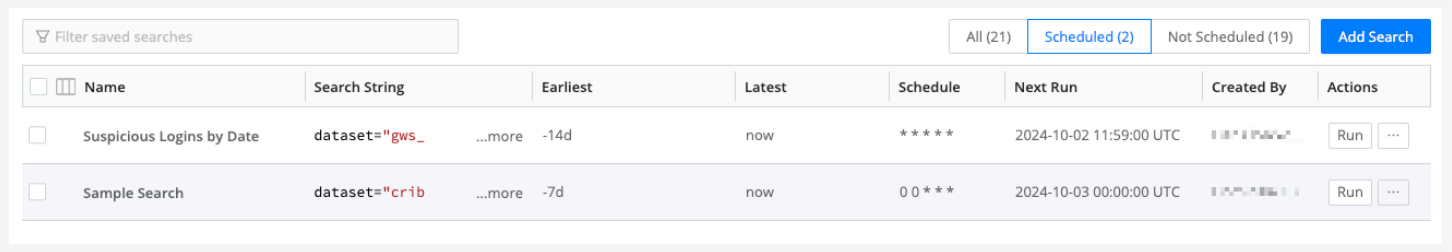
Result sets from scheduled searches can be utilized by:

- using the send operator to automatically send the results to Cribl Stream, for further routing and filtering.

- opening the search from the History tab for review.

History provides a filter on the top right of the table for **Scheduled** searches. The table provides details of each scheduled search, including its query, status, user, last run duration, and latest run time.

- Select a row to view the scheduled search's first result set.
- Select **N Items** to view all of the result sets. From the shown result sets:
  - Select a row to view the results.
  - Select the **Search ID** to view the search's details, helpful for troubleshooting.
- The **Actions** column has options to **Rerun** and **Save** the search.
- Delete one or many result sets by selecting check boxes on the left of a column and then clicking **Delete Selected Jobs** at the bottom left of the table.



Result Sets

Manage Scheduled Searches

You can view and manage your scheduled searches from **Saved Searches**. The saved searches table has a **Scheduled** filter and columns for **Schedule** and **Next Run**.

- Select a row to view, edit, or delete a saved search.
- The **Actions** column has options to **Rerun** and **Clone** the search.
- To stop a scheduled search, toggle its schedule off when editing it.

If you want to view the results of a previously run scheduled search, go to **History**. See the above Results section for details.

5.2. Notifications

Send Notifications about your scheduled searches.

Cribl Search can alert you when your scheduled searches generate results that match triggering conditions that you specify. You set up alerts in these basic stages:

1. Optionally, configure any custom Notification targets to which you want to send alerts. (Cribl Search comes preconfigured with at least one default `System Messages`. target, Some plans add a preconfigured `system_email`.)
2. Save and schedule each search from which you want to trigger alerts.
3. Configure one or more Notifications on each scheduled search.



You can't configure Notifications on a saved or scheduled search that resides within a Pack.

Notification Targets

A Notification target specifies the delivery method for a Notification. Every Notification must have at least one target. Available target types are:

- System Messages: Banner messages in the Cribl Search UI (the default target).
- Email messages.
- Amazon SNS topics.
- PagerDuty accounts.
- Slack channels.
- Webhook connections.

By default, any Notification that you configure will have a `System Messages` target. Also, your Notification will include this target regardless of any other target that you add. When a Notification condition is triggered, Cribl Search will add an indicator on the top nav's **Messages** button. Here, select this button to view details in the **Messages** drawer.

Add a Notification Target

To add a new Notification target:

1. In Cribl Search, go to **Settings** > **Search** > **Notification Targets**.
2. Select **Add Target** to open the **New Target** modal.

3. Give this target a unique **Target ID**.
4. Select the desired **Target type**.
5. Configure the target as outlined on that target's appropriate page (see Notification Targets).

You can also use the Cribl API to create a Notification target.

When you create a Notification target in Cribl Search, Stream, or Edge, it will also be available to you in the other products. For example, if you create a Notification target in Cribl Search, you can also access that target in Cribl Stream and Cribl Edge. (Where you would, of course, configure very different Notification conditions.)

Manage Notification Targets

You manage existing targets on the same **Notification Targets** page where you create targets.

Select any existing Notification target to open a modal where you can manage the target, using buttons at the bottom of the configuration modal.

Select **Clone Target** to copy an existing Notification target. This refreshes the modal, enabling you to customize the new target.

Select **Delete Target** to delete an existing target.

To edit any target's definition in a JSON text editor, select **Manage as JSON** at the bottom of the **Notifications Targets** modal. You can directly edit multiple values, and you can use the **Import** and **Export** buttons to share existing target configurations as `.json` files.

Configure Notifications

To configure how, when, and where to send Notifications:

1. Schedule a search, or open an existing scheduled search.
2. In the search's configuration modal, open the **Notifications** left tab.
3. Toggle **Send notifications** on, and configure the options listed below.



The **Notifications** left tab unlocks only after you enable the **Schedule** tab's **Run on schedule** option.

To configure further Notifications on the same scheduled search, select **Add Notification**.

You can also use the Cribl API to create a Notification.

Notification Options

As you configure each Notification, the **Notifcations** tab offers the following options.

- **When**: Select the type of trigger condition, then define the corresponding trigger. (See Notification Triggers.)
- **Send Notification to**: Select an existing Notification target to deliver the Notification, or select **Create** to create a new target. (For target options, see Notification Targets.)



All Notifications, regardless of their configured target, will also generate a message in the Cribl Search UI.

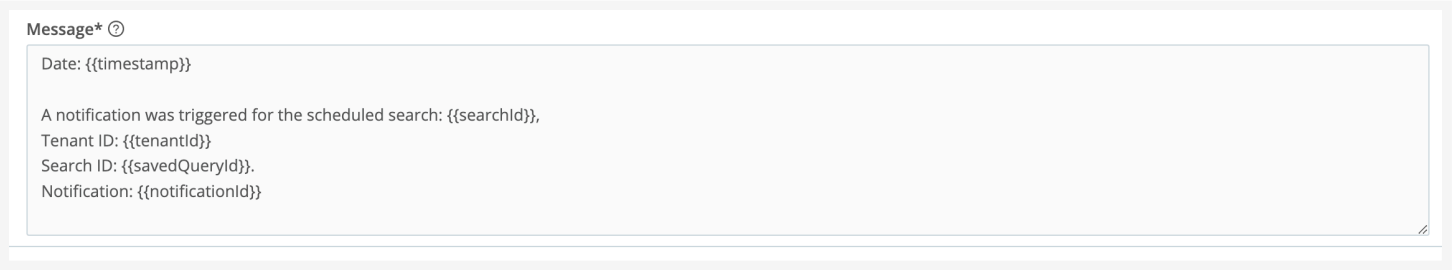
- **Include search results**: You must toggle this on if you want to insert the `{{resultSet}}` template variable into the **Message** field, in order to include or attach complete search results with Notifications. (See Message Options below for details and payload limits.) However, this is not required in order to include the contents of specified events and fields, in the format: `{{resultSet[0].<field-name>}}`. To see how each alert type implements this option, follow the links from Notification Targets.
- **Message**: Message payload template. For details, see Message Options below.

Message Options

The message sent with each Notification can contain a maximum of 1,000 characters, and a maximum of 100 events x 20 fields.

In the **Message** pane, you can customize each Notification's contents by inserting the template variables listed below. These are placeholders to populate your message with fields generated by the search. Insert a variable between pairs of double braces, in this format: `{{savedQueryId}}`.

Here's an example that inserts some literal text followed by a corresponding variable: `Date: {{timestamp}}`. This is the first string in the default template, shown below.



Message pane's Default template

The following template variables are available.

Variable	Description
`resultSet`	Array containing complete results of the search, up to the maximum message size.

Variable	Description
`{{resultSet[<event-number>].<field-name>}}`	Contents of a specified event and field from the search results.
`savedQueryId`	ID of the saved search that triggered the Notification.
`searchId`	ID of the search job.
`searchResultsUrl`	URL corresponding to the search job results.
`notificationId`	ID of this Notification (autogenerated).
`timestamp`	Date when this Notification was triggered.
`tenantId`	ID of the Cribl Organization.

If you choose to explicitly add the `{{resultSet}}` variable to the message body, this will have different effects with different targets:

- With a PagerDuty or Slack Notification, you can include `{{resultSet}}` to control where inline search results will be displayed within alerts.
- With an Email Notification, including `{{resultSet}}` will insert placeholder text indicating that the results will appear either "in the table below" (if you've selected **Send as**: **Inline table**) or "attached" (if you've selected **Send as**: **CSV** or **JSON**).
- With an Amazon SNS or Webhook Notification, including `{{resultSet}}` will send duplicate result sets – one each in the message and on the payload.



In order to insert the complete `{{resultSet}}` template variable, you must toggle on **Include search results**. This is not required in order to insert individual events/fields.

Notification Triggers

On the **Notifications** left tab, use the **When...** controls to define the trigger condition that will send the Notification. The drop-down here provides two options.

- **Where**: Select this to define a custom condition. See Notification Trigger Examples below.
- **Count of Results**: Exposes a drop-down where you can select a comparison operator, and a field where you can enter the corresponding threshold number.

How Notification Triggers Work

Cribl Search will send a Notification when at least one event satisfies the trigger condition. This can be useful if you want to define a search query that returns multiple results, but want alerts sent only when a specific condition among these results is met.

Notification Trigger Examples

For a query like this:

```
dataset="cribl_internal_logs" error
| summarize errorCount=count() by message
| top 10 by errorCount
```

Copy | Run in Cribl Search

You could enter this **Where** condition to alert when `Request failed` is in the top 10 errors:

```
message == "Request failed"
```

Copy

For a query like this:

```
dataset="cribl_search_sample" log_status="OK"
| summarize count() by action
```

Copy | Run in Cribl Search

You could enter this **Where** condition to alert upon connection errors:

```
action == "REJECT"
```

Copy

5.2.1. Email Notifications

Send email Notifications to alert on your scheduled searches.

If you're a Cribl Search Admin, email Notifications make it easy for you (or designated other people) to receive alerts about operational issues that merit your attention, such as a scheduled search's execution details.

An email Notification requires two things – a configured Notification and an email Notification target. Organizations on an Enterprise plan also have access to a preconfigured email Notification target.

Email Notifications do not provide an unsubscribe option. Recipients who do not want to receive particular email Notifications should contact their Cribl Search Admins.

Email Notifications are sent on a no-reply basis.

Email Notification Targets

You can send Notifications by email, using an SMTP server of your choice. Once you have configured the email Notification target, you can specify the recipients, and customize the subject line and message content. For details, see Configure Email Notifications.

To add an email Notification target in Cribl Search, do one of the following:

- Open a saved or scheduled search, then select **Notifications**. Toggle **Send notifications** on, if necessary, then configure the target.
- From the Cribl Search sidebar, select **Settings**, then **Search**, then **Notification Targets**, then **Add Target**.

If you have an Enterprise plan, you can use the preconfigured default Email Notification target.

General Settings

Target ID: Enter a unique name to identify this email Notification target.

Configuration

Address: Identify the SMTP server by its hostname or IP address.

Port: Set the SMTP port. Use port `587` for SMTP Secure (SMTPS). You can also use port `25`. Use `465` when SSL/TLS is enabled. You can also use port `2525` if your email service provider supports this port as a backup when other ports are blocked by a network provider or a firewall.

From: Identify the email address of the sender.

Encryption type: Specify the encryption type used to secure SMTP communication. Options include:

- **STARTTLS**: Select this option to start the connection as plaintext, then upgrade it to a TLS-encrypted one if the server supports it. If the server doesn't support it, the connection remains plaintext.

  STARTTLS upgrades the connection to be encrypted but does not authenticate the server. Take additional steps to prevent man-in-the-middle attacks.

- **Require STARTTLS**: Select this option to require TLS. If the server doesn't support a secure connection, the connection will be dropped.
- **TLS (SMTPS)**: Select this option to use an encrypted connection from the start without requiring a subsequent connection upgrade.
- **None**: Select this option to use a plaintext connection.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path** or by another trusted CA (such as the system's CA).

Authentication

Username: The authentication principal (if required).

Password: The authentication credential (if required).

Test Email Notification Targets

For arrival tests and log indicators, see Test the Email Notification Target.

Default Email Notification Target for Cribl.Cloud Enterprise Orgs

For Cribl.Cloud Enterprise orgs, a default email Notification target is available for Cribl Stream, Edge, and Search. You can use this target to route any email Notification to any valid email address.

This target appears in the **Notifications** > **Target** modal as `system_email`. You cannot modify or remove this target.

The `system_email` Notification target is managed by Cribl. It will be disabled in the unlikely event of abuse. If this target is disabled for a Workspace, a log entry will appear in the Notifications Service logs. Select **Monitoring** > **Logs** > **Notifications Service** to view this log.

The `system_email` email Notification target is available only for Organizations on an Enterprise plan. For details, see Pricing.

Send Domain for Cribl.Cloud Email Notifications

Every Cribl.Cloud Organization and Workspace has a unique address for its email Notification target. Messages sent using this target will have a sender's address in the form `do-not-reply@<workspaceName>-<organizationId>.criblcloud.email`. To ensure delivery, recipients should add the `criblcloud.email` domain to their email allowlists.

Configure Email Notifications

To create an email Notification:

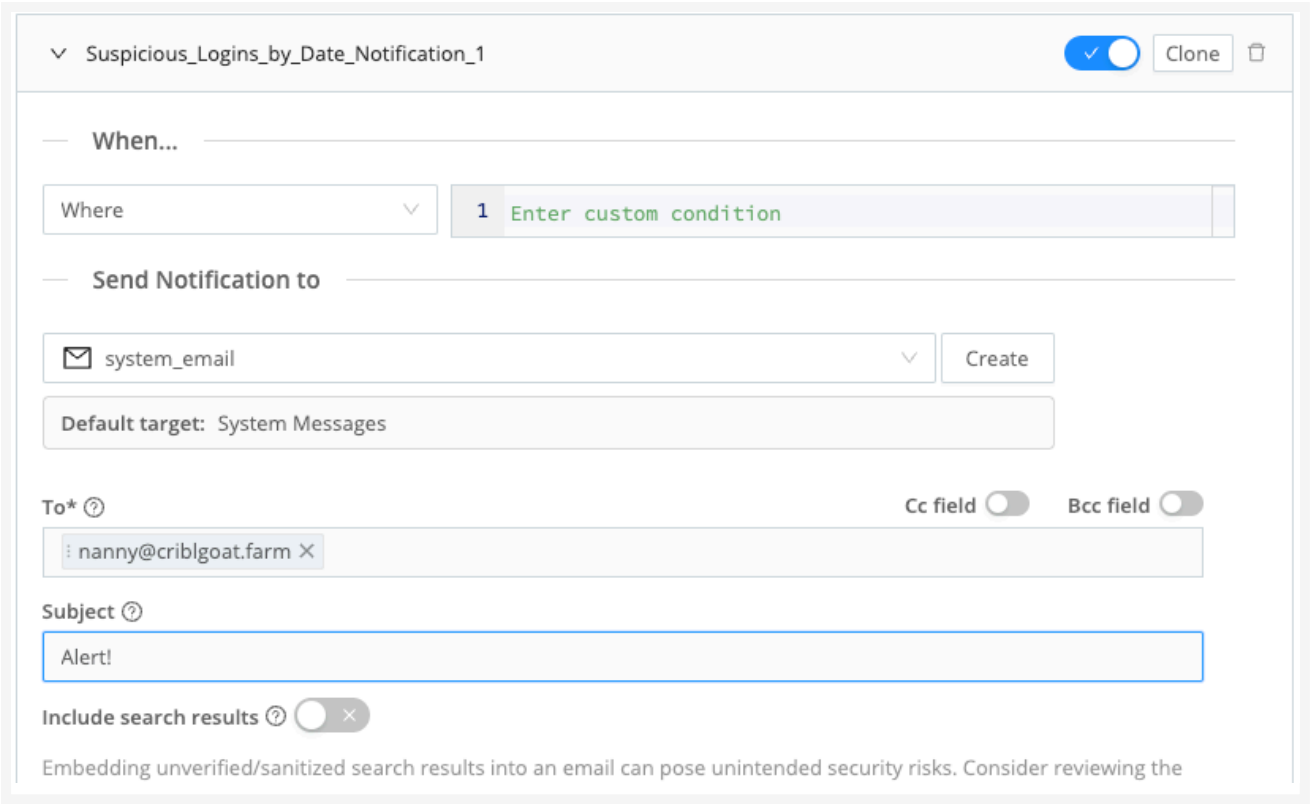
1. Create a scheduled search.

   To schedule an existing search: Save it, then open its configuration modal from the **Saved Searches** tab.

   On that modal's **Schedule** left tab, enable **Run on schedule**.

2. On the **Notifications** left tab, enable **Send notifications** to unlock the options listed below.

Field	Description
When…	Use these controls to define the condition that will send the Notification (see Notification Triggers).
Send Notification to	Select an existing email Notification target from the drop-down. Or select **Create** to define a new target (see Email Notification Targets).



Email Notification with target selected

Once you've selected an email target, the following fields are available.

Field	Description
To	Email address(es) of one or more primary recipients.
Cc field	Toggle this on to expose a field where you can enter additional recipients' addresses.
Bcc field	Toggle this on to expose a field where you can enter blind-copy recipients' addresses.
Include search results	For details about this option, see Include Search Results With Email Notifications.
Subject	The subject line of the email Notification. You can use variables here.
Message	The body of the email Notification. You can use variables here.

Cribl Search does not limit the total number of recipients of an email Notification, but your email service might set a limit.

Email Notification Variables

Email variables are placeholders in the email template, which Cribl Search replaces with actual values when each email is sent. You can use these variables in the Notification's **Subject** or **Message** (body) field. Insert a variable between pairs of double braces, in this format: `{{timestamp}}`.

Variable	Description
`resultSet`	Array containing results of the search.
`savedQueryId`	ID of the saved search that triggered the Notification.
`searchId`	ID of the search job.
`searchResultsUrl`	URL corresponding to the search job results.
`notificationId`	ID of this Notification (autogenerated).
`timestamp`	Date when this Notification was triggered.
`tenantId`	ID of the Cribl Organization.

Include Search Results With Email Notifications

Your email Notifications can include a subset of the search results that you're alerting on. You can choose to include the results as a table within each message body, or as an attached CSV or JSON file.

To configure this:

1. Set up an email Notification target.
2. Open a scheduled search, and select **Notifications**.
3. Configure the Notifications, using the email Notification target you've created.
4. As you configure, toggle on the **Include search results** slider.
5. Select one of the following:
    - **Inline table**, to include the results within the email body. If you choose this option, see the considerations in the next section.
    - **CSV**, to include the results as a CSV attachment.
    - **JSON**, to include the results as a JSON attachment.

    If you select either attachment option, each attachment will automatically be assigned a unique file name, concatenating the following elements: the search ID, an epoch timestamp (representing when the search job was created), a random six-character string, and the appropriate file extension.

Inline Table Notifications

If you select the **Inline table** option, keep these details in mind:

The results table can accommodate up to 100 rows and 20 columns. If a table exceeds this size, Cribl Search will truncate it.

In the table, text will wrap. To make the results more legible, limit the number of fields sent (use the `project` operator) and the length of each field (use the `trim` function).

Security Risk

Embedding unverified or unsanitized search results into an email can pose unintended security risks. Cribl recommends reviewing or sanitizing the results prior to including them.

Troubleshoot Email Notifications

For troubleshooting, see Common Issues and Resolutions.

5.2.2. AWS SNS Notifications

Send AWS SNS Notifications about your scheduled searches.

You can send Notifications to an Amazon Simple Notification Service (SNS) topic. This gives you access to a broad array of alert destinations, such as various AWS services, mobile push notifications, or text messages.

To add an Amazon SNS Notification target in Cribl Search, select **Settings** > **Search** > **Notification Targets** > **Add Target**.

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. It can't be changed later, so make sure you like it.

Configuration

Destination type: Defaults to **Topic ARN**. For the **Phone number** option, see Text Message (SMS) Notifications.

Default Topic ARN: The Amazon Resource Name (ARN) of the Amazon SNS topic to which you want to send Notifications. Enter the ARN in the following format:

```
arn:aws:sns:region:account-id:MyTopic

// for example:
// arn:aws:sns:us-west-2:000000000000:my-topic
```

Copy

If you use a non-AWS URL, the format must be:

```
{url}/myQueueName

// for example:
// https://host:port/myQueueName
```

Copy

The default topic ARN must be a JavaScript expression (which can evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only at initialization time. For example, if you're referencing a global

variable: `` `https://host:port/myQueue-${C.vars.myVar}` ``. This value can be overridden by the Notification event `__topicArn` field.

Topic type: The type of the topic selected in AWS SNS. Can be **Standard** or **FIFO**.

Message group ID: The tag that specifies that a message belongs to a specific message group. Messages that belong to the same message group are processed in a first in, first out (FIFO) manner. Must be a JavaScript expression (which can evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only at initialization time. For example, referencing a global variable: `` `https://host:port/myQueue-${C.vars.myVar}` ``.

Region: The Region associated with the Amazon S3 bucket.

Phone number allowlist: A wildcard list of phone numbers that are allowed to receive SMS notifications. This is used when **Destination type** is set to **Phone number**.

Authentication

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance, local credentials, sidecar, or other source. The attached IAM role grants Cribl access to authorized AWS resources. Can also use the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. Works only when running on AWS.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference.

The **Manual** option exposes these additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

The values for **Access key** and **Secret key** can be a constant, or, a JavaScript expression enclosed in quotes or backticks. For environment variables, use a JavaScript expression, for example: `${C.env.MY_VAR}`.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. The Secret option exposes this additional field:

- **Secret key pair**: Use the drop-down to select an API key/secret key pair that you've configured in Cribl Search's secrets manager. To store a new, reusable secret, select **Create**.

Assume Role

Enable for SNS: Toggle on to define an IAM Role to use, instead of automatically detecting one locally.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID to delegate third-party access. For details, see AWS documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Post-Processing

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Search Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Search Node that processed the event.
- `cribl_input` – Search Source that processed the event.
- `cribl_output` – Search Destination that processed the event.
- `cribl_route` – Search Route (or QuickConnect) that processed the event.
- `cribl_wp` – Search Worker Process that processed the event.

Advanced Settings

Maximum number of retries: The maximum number of retries before the output returns an error. The retries use an exponential backoff policy.

Endpoint: The SNS service endpoint. If empty, defaults to AWS Region-specific endpoint. Otherwise, it must point to an SNS-compatible endpoint.

Signature version: Signature version to use for signing SNS requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Toggle off to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled off.

Send Text Message (SMS) Notifications

You can use an Amazon SNS Notification target to send text messages (SMS) to a list of phone numbers. To do this, you'll need to set up a allowlist of phone numbers that are permitted to receive Notifications.

1. Select **Settings** > **Search** > **Notification Targets** > **Add Target**.
2. Enter a unique **Target ID**.

3. Set the **Target type** to **AWS SNS**.
4. Set **Destination type** to **Phone number**.
5. Set **Region** to the Region of the Amazon S3 bucket.
6. In **Default Phone number**, enter a comma-separated list of phone numbers that are allowed to receive Notifications. This value can be overridden by the Notification event `__phoneNumber` field. You can use `*` as the wildcard character.
   For example: `+15555550123, +15555551***`.
7. If desired, use **Phone number allowlist** to specify a wildcard list of allowed phone numbers.
8. Configure the remaining sections of the target, as described in Post-Processing and Advanced Settings.
9. Select **Save**.

Now, when you set up Notifications, you can select the new Amazon SNS target and specify any phone number that matches the configured allowlist.

Include Search Results with AWS SNS Notifications

You can send AWS SNS Notifications that include a subset of search results in the Notification payload. Once you've configured your AWS SNS target, you specify this option separately on each Notification that uses the target.

To set this up:

1. Set up an AWS SNS Notification target, as outlined on this page.
2. Open a scheduled search, and select **Notifications**.
3. Configure the Notifications, using the AWS SNS Notification target you've created.
4. Within the configuration, enable **Include search results**.

Take the following into account:

- The results table can accommodate up to 100 rows and 20 columns. If it's larger, Cribl Search will truncate it.
- To make the results more legible, you can limit the number of fields sent (use the `project` operator) and the length of each field (use the `trim` function).



Security Risk

Embedding unverified or unsanitized search results into an AWS SNS Notification payload can pose unintended security risks. Cribl recommends reviewing result sets before including results with Notifications.

5.2.3. PagerDuty Notifications

Send PagerDuty Notifications about your scheduled searches.

You can send Notifications to PagerDuty, a real-time incident response platform, using Search's native integration with the PagerDuty API. Select Select **PagerDuty** in the **New Target** modal to expose the following additional options on the modal's (single) **General Settings** left tab:

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. It can't be changed later, so make sure you like it.

Configuration

Routing key: Enter your 32-character Integration key on a PagerDuty service or global ruleset.

Group: Optionally, specify a PagerDuty default group to assign to Cribl Search Notifications.

Class: Optionally, specify a PagerDuty default class to assign to Cribl Search Notifications.

Component: Optionally, a PagerDuty default component value to assign to Cribl Search Notification. (This field is prefilled with `logstream`.)

Severity: Set the default message severity for events sent to PagerDuty. Defaults to `info`; you can instead select `error`, `warning`, or `critical`. (Will be overridden by the `__severity` value, if set.)

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Search limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Search applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Search first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Search continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Search will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Search should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Search will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Search should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

After you've set and saved all of the above target configuration, you can set up Notifications that use the new PagerDuty target as their destination.

Include Search Results with PagerDuty Notifications

You can send PagerDuty Notifications that include a subset of search results in the Notification payload. Once you've configured your PagerDuty target, you specify this option separately on each Notification that uses the target.

To set this up:

1. Set up a PagerDuty Notification target, as outlined above.
2. Open a scheduled search, and select **Notifications**.
3. Configure the Notifications, using the PagerDuty Notification target you've created.
4. Within the configuration, enable **Include search results**.

Take the following into account:

- The results table can accommodate up to 100 rows and 20 columns. If it's larger, Cribl Search will truncate it.
- To make the results more legible, you can limit the number of fields sent (use the `project` operator) and the length of each field (use the `trim` function).

Security Risk

Embedding unverified or unsanitized search results into an AWS SNS Notification payload can pose unintended security risks. Cribl recommends reviewing result sets before including results with Notifications.

5.2.4. Slack Notifications

Send Slack Notifications about your scheduled searches.

You can send Notifications to a Slack channel, using Slack Incoming Webhooks.



You can also use a Webhook Destination to send specific event data to Slack for a team's attention. For more information, see our Slack/Webhook integration topic.

First, in your Slack workspace, use a Slack app to enable incoming webhooks.

Then, in the **New Target** modal, select **Slack** to expose the following additional options on the modal's (single) **General Settings** left tab:

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. You can't change it later, so make sure you like it.

Configuration

Webhook URL: Add the full URL of your Slack Incoming Webhook. For example: `https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX`.

Retries

Honor Retry-After header: When toggled on and the `retry-after` header is present, Search honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Search always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Search will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled off (default), Search ignores all `retry-after` headers.

Settings for failed HTTP requests: Automatically retries after unsuccessful response status codes, such as `429` (Too Many Requests) or `503` (Service Unavailable). Selecting **Add Setting** reveals a table where you can add retry parameters for individual failed HTTP requests. These include:

- **HTTP status code**: The individual status code for which the retry parameters apply.

- **Pre-backoff interval (ms)**: How long, in milliseconds, Search should wait before initiating backoff. The maximum interval is 600,000 ms (10 minutes).
- **Backoff multiplier**: Base for exponential backoff. A value of `2` (default) means Search will retry after 2 seconds, then 4 seconds, then 8 seconds, and so forth.
- **Backoff limit (ms)**: The maximum backoff interval, in milliseconds, Search should apply. Default (and minimum) is 10,000 ms (10 seconds); maximum is 180,000 ms (180 seconds).

Retry timed-out HTTP requests: Toggle on if you want Search to automatically retries HTTP requests that have timed out. Defaults to off. Enabling this option exposes the following settings:

- **Pre-backoff interval (ms)**: How long, in milliseconds, Search should wait before initiating backoff. Maximum interval is `600,000 ms` (10 minutes).
- **Backoff multiplier**: Base for exponential backoff. A value of `2` (default) means Search will retry after 2 seconds, then 4 seconds, then 8 seconds, and so forth.
- **Backoff limit (ms)**: The maximum backoff interval, in milliseconds, Search should apply. Default (and minimum) is 10,000 ms (10 seconds); maximum is 180,000 ms (180 seconds).

After you've set and saved all of the above target configuration, you can set up Notifications that use the new Slack target as their destination.

Include Search Results with Slack Notifications

You can send Slack Notifications that include a subset of search results in the Notification payload. Once you've configured your Slack target, you specify this option separately on each Notification that uses the target.

To set this up:

1. Set up a Slack Notification target, as outlined above.
2. Open a scheduled search, and select **Notifications**.
3. Configure the Notifications, using the Slack Notification target you've created.
4. Within the configuration, enable **Include search results**.

Take the following into account:

- The results table can accommodate up to 100 rows and 20 columns. If it's larger, Cribl Search will truncate it.
- To make the results more legible, you can limit the number of fields sent (use the `project` operator) and the length of each field (use the `trim` function).



Security Risk

Embedding unverified or unsanitized search results into an AWS SNS Notification payload can pose unintended security risks. Cribl recommends reviewing result sets before including results with Notifications.

5.2.5. Webhook Notifications

Send webhook Notifications about your scheduled searches.

You can send Notifications to an arbitrary webhook. This topic outlines how to configure a Webhook target, and then how to send individual alerts to the corresponding endpoint

Target Configuration

In the **New Target** modal, select the **Webhook** radio button to expose multiple left tabs that provide the configuration options outlined below.

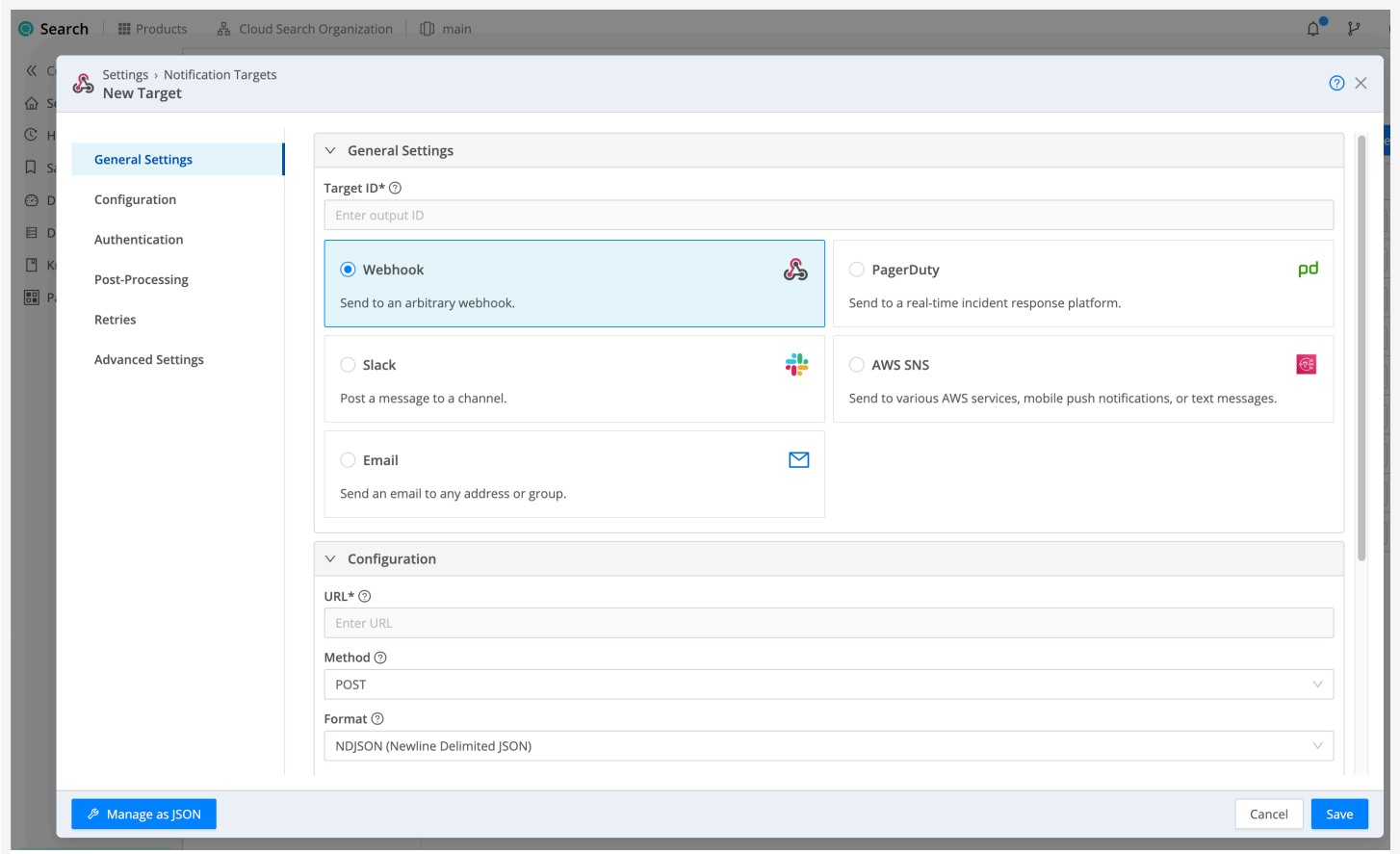


For a streamlined, illustrated example of configuring a Webhook target, see Notifications via Cribl Stream.

General Settings

On this first left tab, fill in the following field.

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. You can't change it later, so choose an ID that will make this target's purpose clear to you and other users .



Webhook Notification target – General Settings and Configuration tabs

Configuration

This second left tab exposes the added options listed in this section.

URL: The endpoint that should receive Cribl Search Notification events.

To proxy outbound HTTP/S requests, see System Proxy Configuration.

Method: Select the appropriate HTTP verb for requests: `POST` (the default), `PUT`, or `PATCH`.

Format: Specifies how to format Notification events before sending them to the endpoint. Select one of the following:

- `NDJSON` (newline-delimited JSON, the default).
- `JSON Array`.
- `Custom`, which exposes these additional fields:
  - **Source expression**: JavaScript expression whose evaluation shapes the event that Search sends to the endpoint – for example: `notification=${_raw}`. For other fields you can use, see Expression Fields. If empty, Search will send the full Notification event as stringified JSON.

- **Drop when null**: Toggle on if you want to drop events where the above **Source expression** evaluates to `null`.
- **Event delimiter**: Delimiter string to insert between individual events. Defaults to newline character (`\n`).
- **Content type**: Defaults to `application/x-ndjson`. You can substitute a different content type for requests sent to the endpoint. This entry will be overridden by any content types set in this modal's **Advanced Settings** tab > **Extra HTTP Headers** section.
- **Batch expression**: Expression specifying how to format the payload for each batch. Defines a wrapper object in which to include the formatted events, such as a JSON document. This enables requests to APIs that require such objects. To reference the events to send, use the `${events}` variable. An example expression to send the batch inside a JSON object would be: `{"items" : [${events}] }`.

Expression Fields

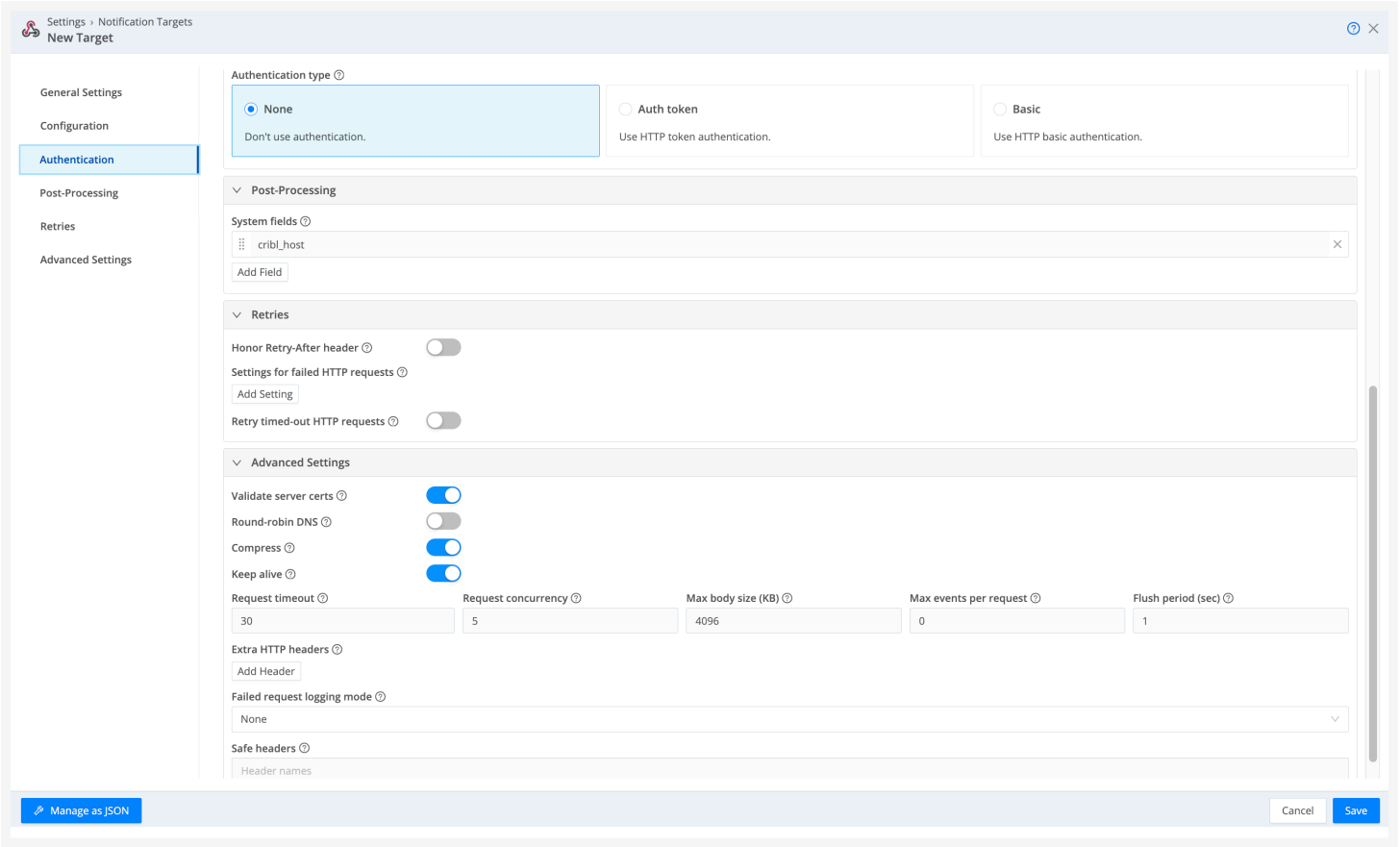
When building the Source expression used with the `Custom` **Format**, you can use the following fields:

- `starttime`: Beginning of the time bucket where this condition was reported. All Notifications have this field.
- `endtime`: End of the time bucket where this condition was reported. All Notifications have this field.
- `_time`: Timestamp when this Notification was created. All Notifications have this.
- `cribl_host`: Hostname of the (physical or virtual) machine on which this Notification was created. All Destination and Source Notifications have this.
- `cribl_notification`: Configured name/ID of this Notification. All Destination and Source Notifications have this.
- `origin_metadata`: Object containing metadata about the Notification origin, with the following fields for all Destination Notifications:
  - `type`: "output".
  - `id`: ID of the affected Destination.
  - `subType`: Destination's type (where applicable).
- `origin_metadata`: Object containing metadata about the Notification origin, with the following fields for all Source Notifications:
  - `type`: "input".
  - `id`: ID of the affected Source.
  - `subType`: Source's type (where applicable).

Authentication

On this third left tab, select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.
- **Basic**: In the resulting **Username** and **Password** fields, enter HTTP Basic authentication credentials.



Webhook Notification target – Authentication and lower tabs

Post-Processing Settings

System fields: In this section, you can specify a list of fields to automatically add to events relayed through this target. By default, includes `cribl_pipe` (identifying the Search Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Search Node that processed the event.
- `cribl_input` – Search Source that processed the event.
- `cribl_output` – Search Destination that processed the event.
- `cribl_route` – Search Route (or QuickConnect) that processed the event.
- `cribl_wp` – Search Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Search limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Search applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Search first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Search continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Search will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Search should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and configure the following settings for retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Search will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Search should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates not authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Search to cycle through them in the order returned.

Compress: Toggle on (default) to compress the payload body before sending.



If you accept this default, make sure that the endpoint receiving these Notifications supports compression.

Keep alive: By default, Search sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle **Keep alive** off if you want Search to close the connection immediately after sending a request.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Search hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB, but you can set this limit to as high as 500 MB (512000 KB).

High values can cause high memory usage per Edge Node, especially if a dynamically constructed URL causes this target to send events to more than one URL. The actual request body size might exceed the specified value because the target adds bytes when it writes to the downstream receiver. Therefore, we recommend that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field; headers added by this method take precedence over headers defined in the **Extra HTTP headers** table.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Search will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Used to declare headers that are safe to log as plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Alerts Using the Target

After you've set and saved all of the above target configuration, you can set up Notifications that use the new Webhook target as their destination.

Include Search Results with Webhook Notifications

You can send Webhook Notifications that include a subset of search results in the Notification payload. Once you've configured your Webhook target, you specify this option separately on each Notification that uses the target.

To set this up:

1. Set up a Webhook Notification target, as outlined above.
2. Open a scheduled search, and select **Notifications**.
3. Configure the Notifications, using the Webhook Notification target you've created.
4. Within the configuration, enable **Include search results**.

Take the following into account:

- The results table can accommodate up to 100 rows and 20 columns. If it's larger, Cribl Search will truncate it.
- To make the results more legible, you can limit the number of fields sent (use the `project` operator) and the length of each field (use the `trim` function).



Security Risk

Embedding unverified or unsanitized search results into an AWS SNS Notification payload can pose unintended security risks. Cribl recommends reviewing result sets before including results with Notifications.

5.2.6. Notifications Via Cribl Stream

Send Notifications about your scheduled searches through Cribl Stream to multiple downstream services.

This topic shows how to send Notifications to Cribl Stream Worker Groups, which can forward them to multiple downstream services, using workflows you've configured to also handle Notifications from Cribl Stream itself (and other Cribl products).

This technique is a convenient alternative to configuring multiple Webhook Notification targets in Cribl Search. Here, you configure a single Webhook target for the Worker Group, and let Cribl Stream multiplex the outbound Notifications from there.

We outline this technique in the following general steps:

- Identify Ingest Address for your Worker Group
- Add Webhook Notification Target in Cribl Search
- Schedule and Save Search and Notification
- Verify Inflow to Cribl Stream
- Configure Outflow from Cribl Stream

Identify Ingest Address

To find the ingest address of a Cribl-managed Worker Group, start in the Cribl.Cloud Organization that hosts your Cribl Search instance:

1. From the top bar, select **Products** > **Cribl**.
2. From the resulting sidebar, select **Data Sources**.
3. Use the **Group** drop-down to select the Cribl-managed Worker Group where you want to send the alerts.
4. Under **Sources Enabled by Default**, find the `http` entry.
5. Here, copy the **Ingest Address** (URL) of your Worker Group. This will point to port `10080`. The general format is:
   `https://<groupName>.<workspaceName>.<organizationId>.cribl.cloud:10080`

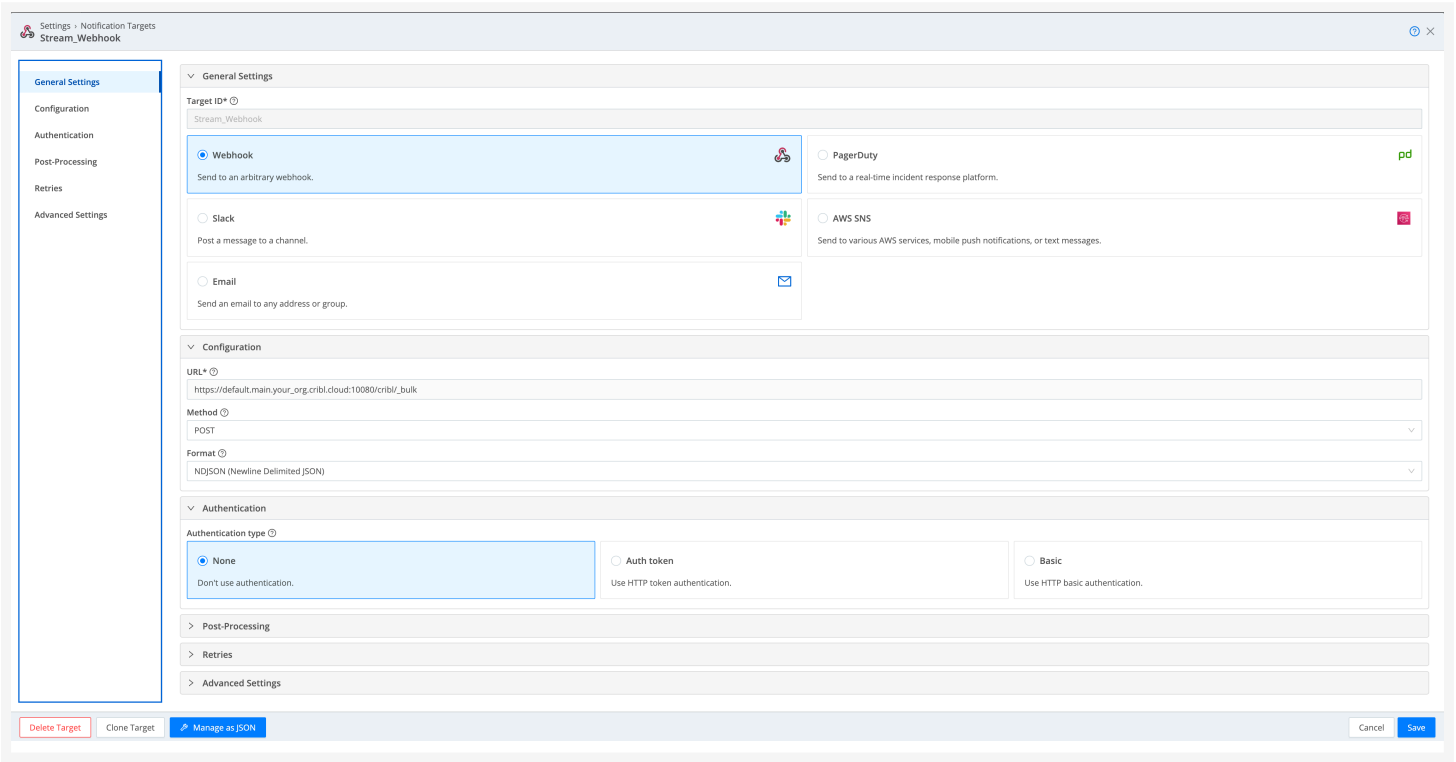
   An example with the most typical Group and Workspace names, and with a fictitious Organization name: `https://default.main.goat-farm.cribl.cloud:10080`



For a hybrid Group running on a host that you manage, the ingest address will typically be configured through your load balancer.

Add Webhook Notification Target

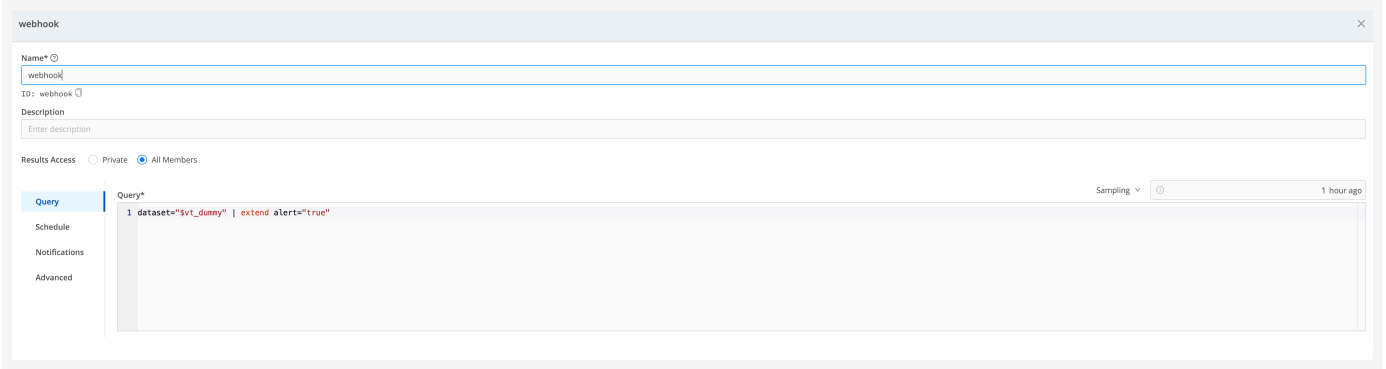
1. Navigate back to Cribl Search: Select **Products** > **Search**.
2. Select **Settings** > **Search** > **Notification Targets**.
3. Select the **Webhook** type.
4. Select **Add Target**.
5. Paste the **Ingest Address** URL that you copied from your Cribl.Cloud Organization, appending `/cribl/_bulk` as in this example:
   `https://default.main.<organizationId>.cribl.cloud:10080/cribl/_bulk`
6. Save the new target as `stream_webhook` or your desired name.



Configuring the Webhook Notification target

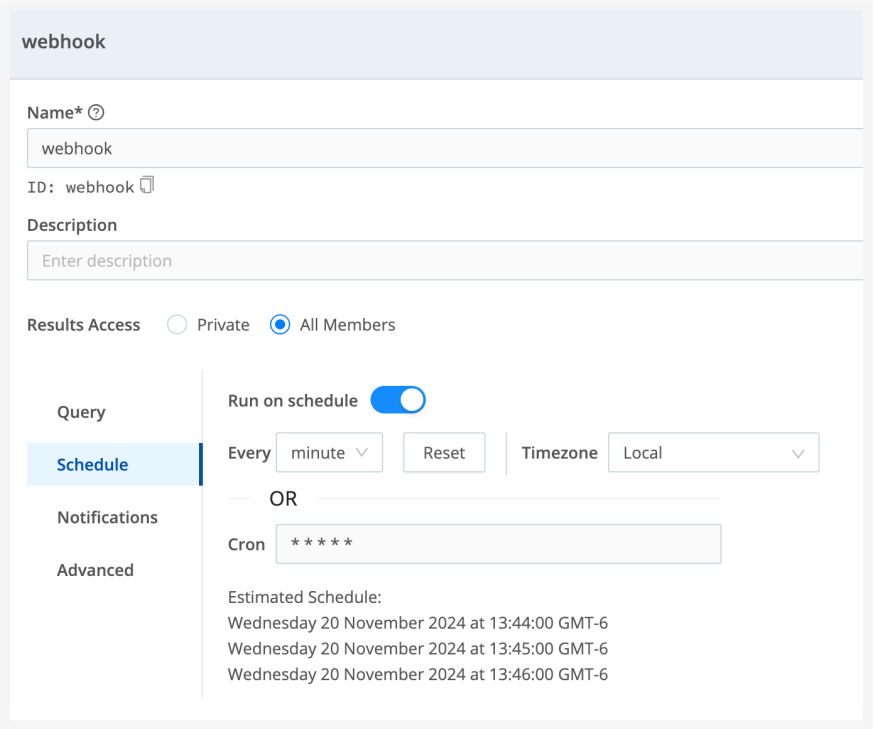
Schedule and Save Search and Notification

1. Create a search, customized according to your needs. The example shown here uses this query:
   `dataset="$vt_dummy" | extend alert="true"`
2. Select **Actions** > **Save** to open the configuration modal shown below.



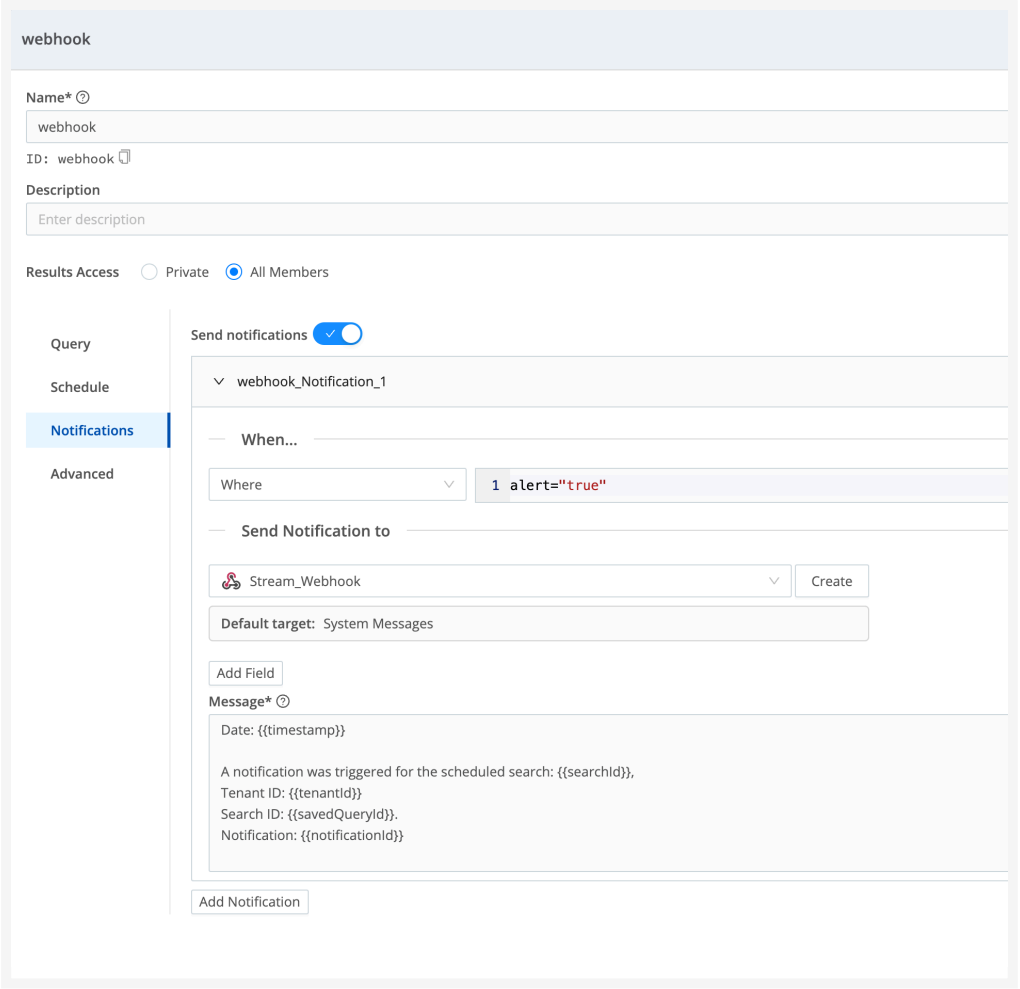
Saving the search configuration

3. Schedule the search, according to your needs. This example sets a `* * * * *` cron schedule, to repeat the search every minute.



Scheduling the search

4. Add the Notification, pointing to the Webhook target you configured above.



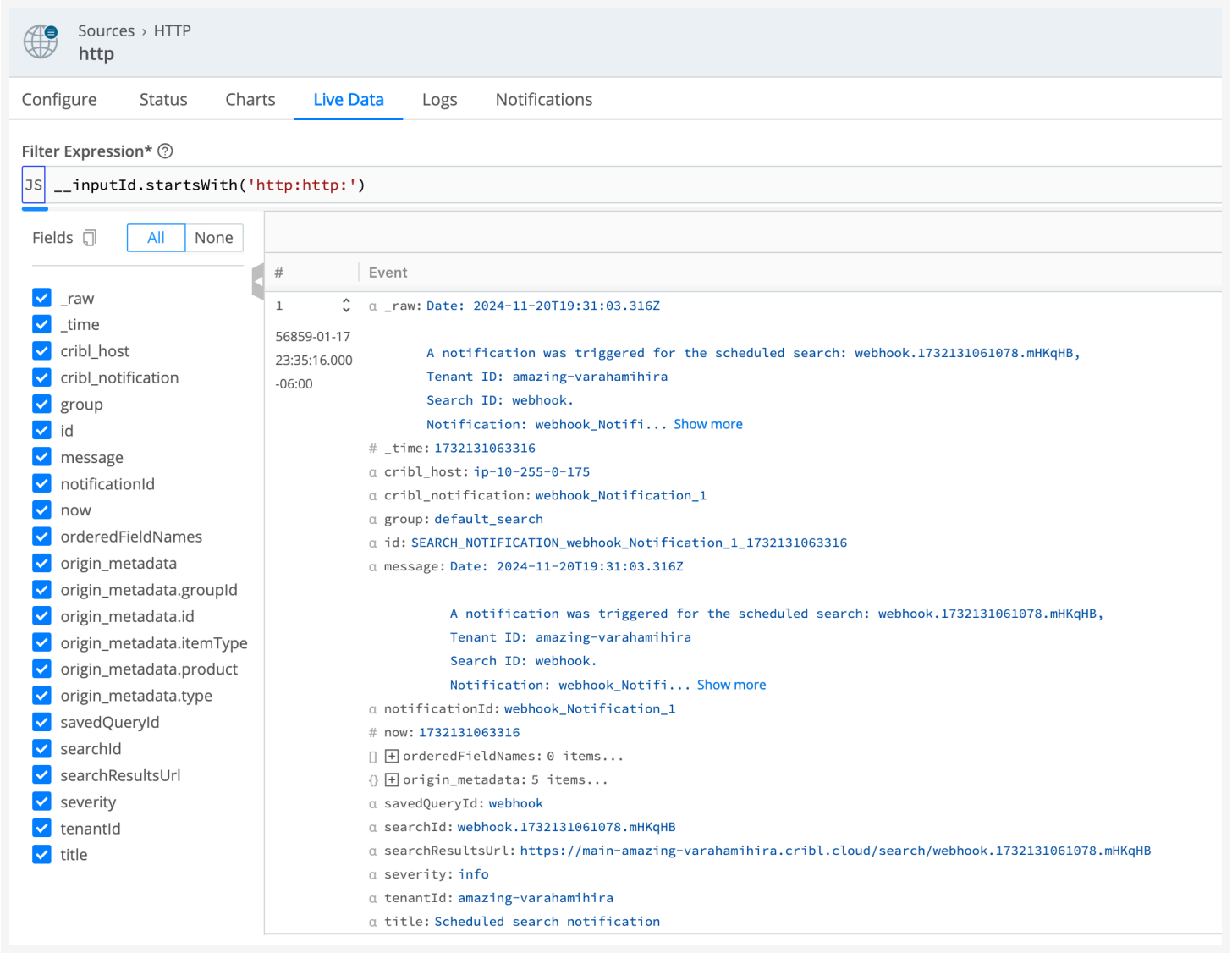
Specifying the Notification

5. **Save** the search, with its configured schedule and Notification.

Verify Inflow to Cribl Stream

Next, make sure that Cribl Search is forwarding your Notifications to Cribl Stream:

1. In Cribl Stream, navigate to your targeted Worker Group.
2. Open the HTTP Source config modal: Select **Sources** > **HTTP**.
3. Select the **Live Data** tab, then its **Capture** button.
4. Set a long time window (several minutes), then select **Start**.
5. Wait. In about a minute, a properly configured alert will appear.



Previewing the Notification's arrival in Cribl Stream

Configure Outflow from Cribl Stream

1. In Cribl Stream, configure or reconfigure a Destination, Pipeline, and Route to relay the alerts to your chosen downstream service(s).
2. Verify connectivity all the way through to these receivers.
3. That's it!

6. Enrich And Automate

Use and customize Knowledge libraries to enrich your data, Macros to automate your searches, and Packs to share pre-built Cribl Search configurations.

You can access and manage Cribl Search Knowledge libraries and Macros by selecting **Knowledge** from the sidebar. You'll now see upper tabs where you can configure the enrichment and automation resources listed below.

- Lookups are data tables that the `lookup` and `ip-lookup` operators use to enrich events with relevant geographic, identity, threat, or other details.
- Parsers in Cribl Search are definitions for Datatypes and for the `extract` operator.
- Regexes are definitions for Parsers.
- Grok patterns are definitions for Parsers.
- Macros are saved query snippets that you can reuse across different searches, providing consistency and a single point of maintenance.

Select **Packs** from the sidebar to import, modify, create, and export pre-built Cribl Search resources. Packs can include Dashboards, saved searches, Datatypes, Macros, and lookups – in any combination required to support a given data source or workflow.

6.1. Lookups

Enrich events with lookup tables.

Lookups are tables of data referenced by the `lookup` and `ip-lookup` operators to enrich events. For example, you can use lookups to map user IDs to additional attributes like names or addresses, and return this enriched data in the search results.

You can access the **Lookups** page, which provides a management interface for all lookups, by selecting **Knowledge** > **Lookups**.

This lookups table is searchable, and you can add Tags to each lookup as necessary. There's full support for `.csv` files. Compressed files are supported, but must be in gzip format (`.gz` extension).



Lookups page

For details about who can access and modify this resource, see Search Member Permissions. With appropriate Permissions, you can export, share, and import lookups in Packs.

Add a Lookup

As an alternative to using the **Lookups** page to add a lookup, you can use the `export` operator in a query to create or update a lookup. The `export` operator can write a maximum of 10,000 rows to the table.

1. On the top right, select **Add Lookup File**.
2. Select **Upload a New File** or **Create with Text Editor**.
3. Give your lookup a unique **Filename**. You'll use this to identify the lookup table in a lookup operation.
4. Optionally, enter a **Description**.
5. Optionally, enter any desired **Tags**.
6. If you selected **Create with Text Editor**, enter your lookup table in the text box. Use CSV (comma-separated values) format with the header values on the first row as the field names, and the field values in the subsequent rows.

Lookup field names must use letters, numbers, and underscores only.

7. Select **Save**.

You can't use the `export` operator to create Pack lookups.

Example Lookup Table

```
service_name,port_number,transport
tcpmux,1,tcp
tcpmux,1,udp
compressnet,2,tcp
compressnet,2,udp
compressnet,3,tcp
compressnet,3,udp
rje,5,tcp
rje,5,udp
echo,7,tcp
```

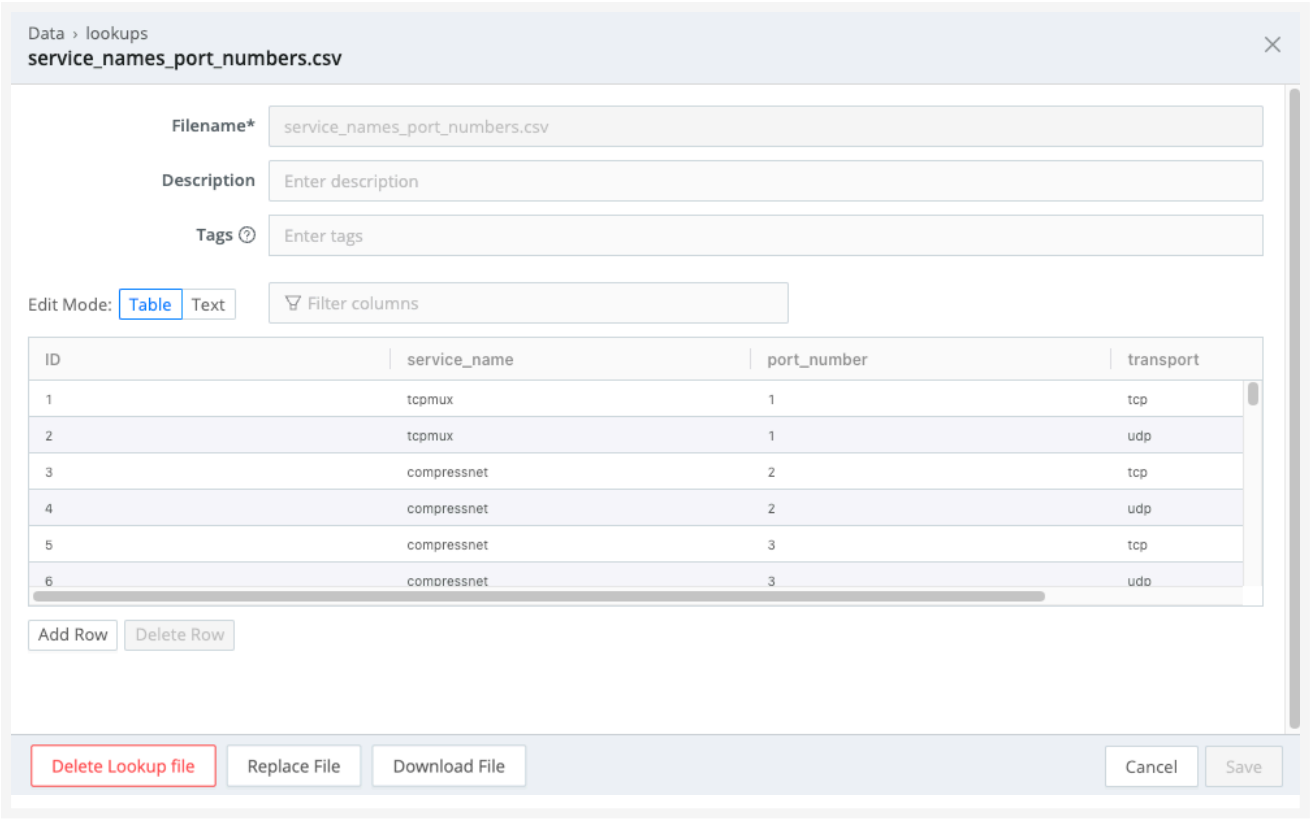
Copy

Modify a Lookup

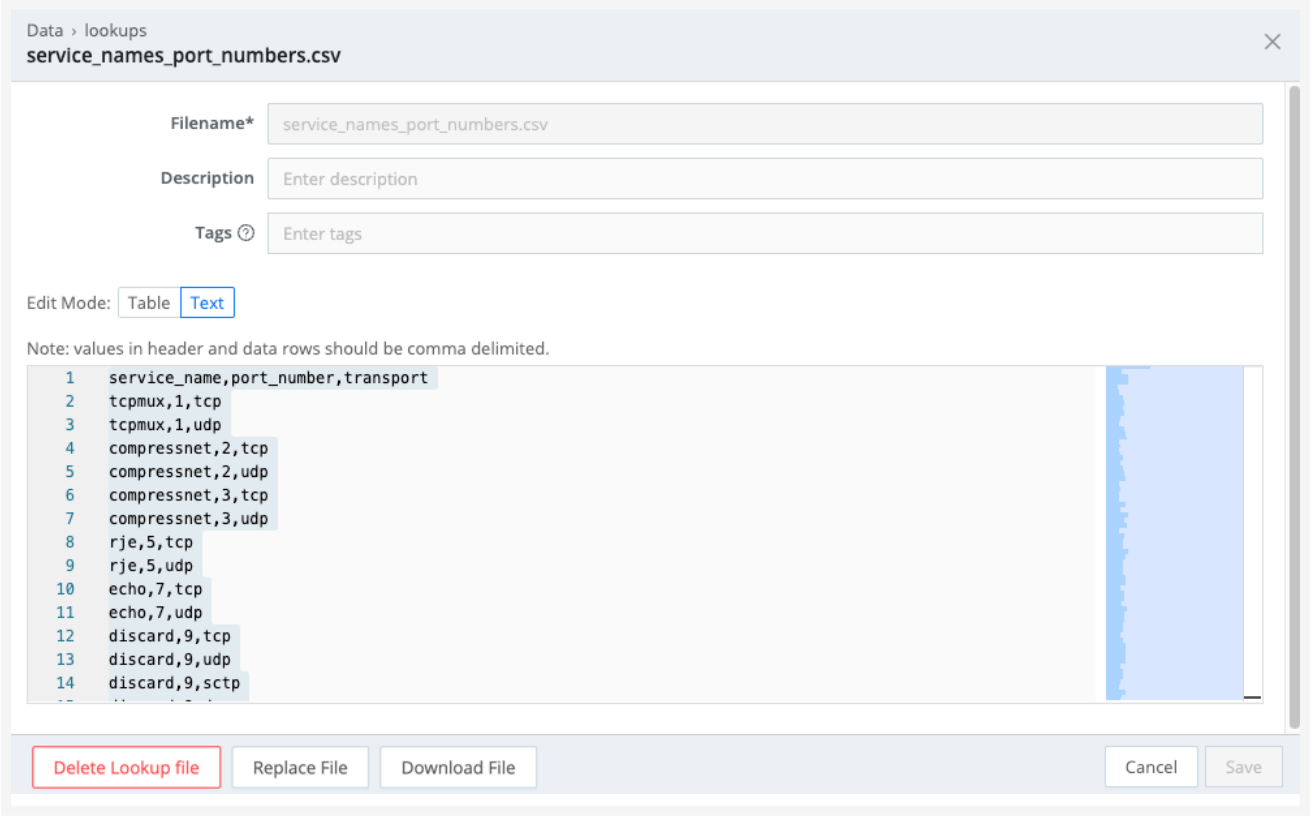
To re-upload, expand, edit, or delete an existing lookup, select its row on the **Lookups** page.

If the lookup you want to delete belongs to a Pack, see Delete a Pack Resource.

In the resulting modal, you can edit files in **Table** or **Text** mode. However, **Text** mode is disabled for files larger than 1 MB. The maximum lookup size is 10,000 rows.



Editing in Table mode



Editing in Text mode

Editing Regex with Quotation Marks

> When a regex contains quotes, edit it in **Text** mode. Do not edit it in **Table** mode, which can cause the regex to behave in unexpected ways.

You can't use the `export` operator to modify Pack lookups.

Export a Lookup

To export a lookup file, including any modifications that you've made, you have two options:

- From a lookup's configuration modal, select the **Download File** button.
- From the **Lookups** page, Open the Actions ⋯ menu on the lookup file's table row, then select **Download**.

List Available Lookups

You can list all your lookup files straight from the query box, using the `$vt_lookups` virtual table:

```
// lists all lookups available to the current user
dataset="$vt_lookups"
```

Copy | Run in Cribl Search

Search the Contents of Your Lookups

You can use the `$vt_lookups` virtual table to search the contents of your lookups.

For example, to get the contents of a lookup named `service_names_port_numbers.csv`:

```
dataset="$vt_lookups" lookupFile="service_names_port_numbers"
```

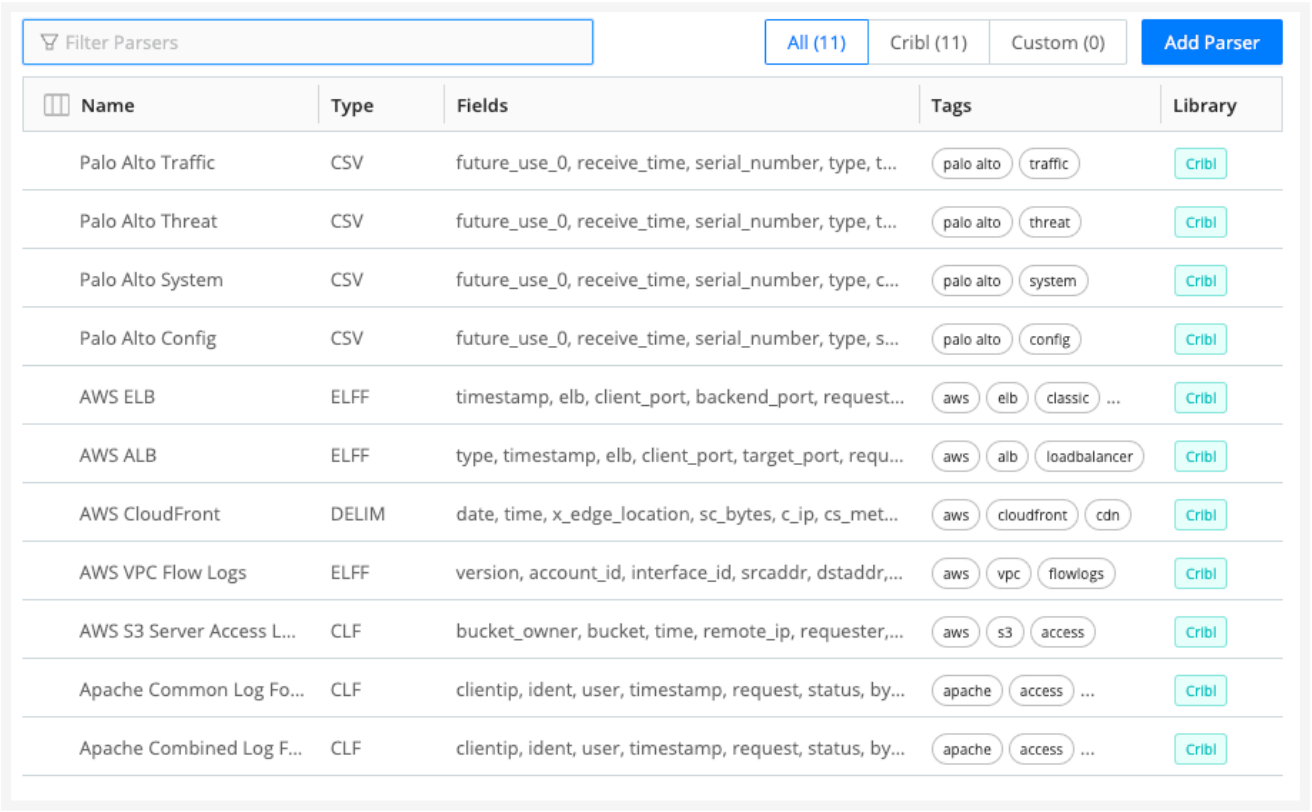
Copy | Run in Cribl Search

6.2. Parsers

Define data to extract, using Parsers.

Parsers in Cribl Search are definitions for Datatypes and the `extract` operator.

To open the **Parsers** page, select **Knowledge** > **Parsers**. The Parsers page provides an interface for creating and editing Parsers. The table is searchable, and you can add Tags to each Parser as necessary.



Parsers page

For details about who can access and modify this resource, see Search Member Permissions.

Supported Parser Types

- **CSV**: Comma-separated values.
- **Extended Log File Format**: Extended Log File Format.
- **Common Log Format**: Common Log Format.
- **Key=Value Pairs**: A set of data that represents two associated groups through a key and a value.

- **JSON Object**: JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax.
- **Delimited values**: A character identifies the beginning or the end of a character string.
- **Regular Expression**: A sequence of characters that specifies a match pattern in text.
- **Grok**: A string of special characters and regular expressions (pattern) that match data.

Creating a Parser

To create a Parser, follow these steps:

1. Go to **Knowledge** > **Parsers** and select **Add Parser**.
2. Enter a unique **ID**.
3. Optionally, enter a **Description**.
4. Optionally, enter any desired **Tags**.
5. Select a Parser **Type** (see the supported types above).
6. Enter the **List of fields** expected to be extracted, in order. Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.
7. Based on the **Type** selected, you'll also have additional configurations. See the below sections for details, either Key=Value Pairs, Delimited Values, Regular Expressions, or Grok.
8. Select **Save** when you're finished.

Key=Value Pairs

- **Clean fields**: Whether to clean field names by replacing non-alphanumeric characters `[a-zA-Z0-9]` with an underscore `_`.
- **Allowed key characters**: A list of characters that can appear in a key name, even though they're normally separator or control characters.
- **Allowed value characters**: A list of characters that can appear in a value name, even though they're normally separator or control characters.

Delimited Values

- **Delimiter**: Delimiter character to use to split values.
- **Quote character**: Character used to quote values. Required if values contain the delimiter character.
- **Escape character**: Character used to escape characters within a value.
- **Null value**: String value that should be treated as null or undefined.

Regular Expression

- **Regex**: Regex literal with named capturing groups, for example, `(?<foo>bar)`. Or with `_NAME_` and `_VALUE_` capturing groups, for example, `(?<_NAME_0>[^ =]+)=(?<_VALUE_0>[^,]+)`.
- **Additional regex**: Add another regular expression to match against other data.
- **Max exec**: The maximum number of times to apply the regex to the source field when the global flag is set, or when using named capturing groups.
- **Field name format expression**: JavaScript expression to format field names when `_NAME_n` and `_VALUE_n` capturing groups are used. The original field name is in the global variable `name`. For example, to append `XX` to all field names: `${name}_XX` (backticks are literal). If empty, names will be sanitized using this regex: `/^[_0-9]+|[^a-zA-Z0-9_]+/g`. You can access other field values via `__e.<fieldName>`.
- **Overwrite existing fields**: Overwrite existing event fields with extracted values. If set to **No**, existing fields will be converted to an array.

Grok

- **Pattern**: Grok pattern to extract fields. Syntax supported: `%{PATTERN_NAME:FIELD_NAME}`.
- **Additional Grok patterns**: Add another pattern to match other data.

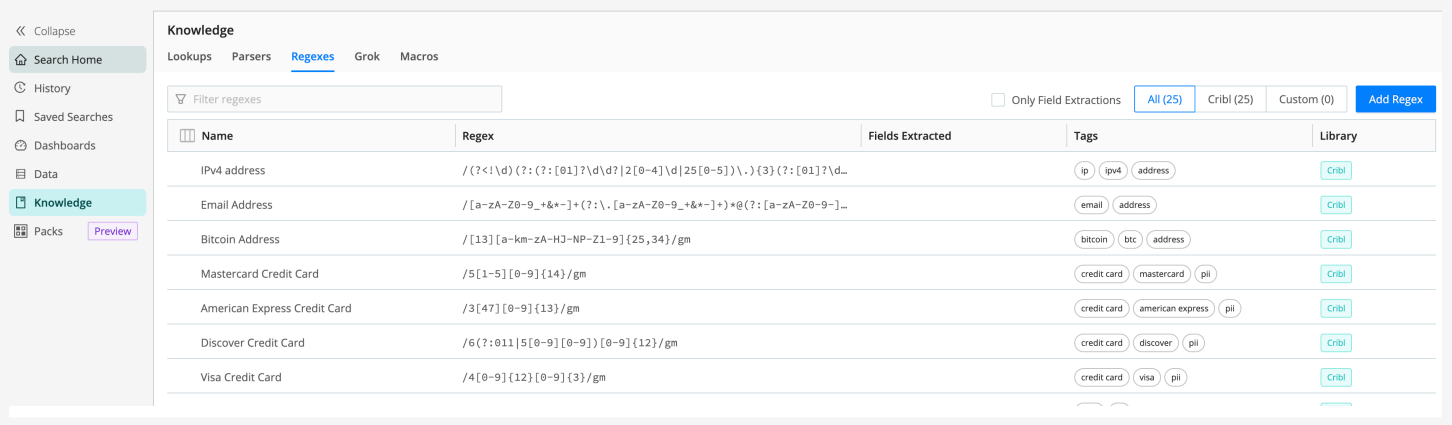
6.3. Regexes

Use the built-in library of regular expressions, or define your own.

A regular expression (regex) is a sequence of characters that specifies a match pattern in text. Cribl Search uses regexes in definitions for Parsers.

Cribl Search ships with many commonly used regular expressions. The **Regexes** page provides an interface for creating and editing regular expressions. The table is searchable, and you can add Tags to each regex as necessary.

To open the **Regexes** page, select **Knowledge** > **Regexes**.



Regexes page

For details about who can access and modify this resource, see Search Member Permissions.

Add Custom Regex

1. On the top right, select **Add Regex**.
2. In the resulting modal, give your custom regex a unique **ID** and optionally, provide a **Description**.
3. Enter your **Regex pattern** and any **Sample data** the regex matches against.
4. Optionally, you can assign **Tags** for further organization or categorization in Cribl.
5. Select **Save**.

Exporting and Importing Regexes

You can export and import Custom (or Cribl) Regexes as JSON files. This facilitates sharing Regexes among Worker Groups or Cribl Stream and Cribl Edge deployments.

To export a Regex:

1. Select to open an existing Regex, or create a new Regex.
2. In the resulting modal, select **Manage as JSON** to open the JSON editor.
3. You can now modify the Regex directly in JSON, if you choose.
4. Select **Export**.

To import any Regex that has been exported as a valid JSON file:

1. Create a new Regex.
2. In the resulting modal, select **Manage as JSON** to open the JSON editor.
3. Select **Import**, and choose the file you want.
4. Select **OK** to get back to the **New regex** modal.
5. Select **Save**.



Every Regex must have a unique value in its top `id` key. Importing a JSON file with a duplicate `id` value will fail at the final **Save** step, with a message that the Regex already exists. You can remedy this by giving the file a unique `id` value.

Cribl Regexes Versus Custom Regexes

Regexes shipped by Cribl are listed under the **Cribl** tag, while user-built regexes are listed under **Custom**. Over time, Cribl will ship more patterns, so this distinction allows for both sets to grow independently. In the case of an ID/Name conflict, the Custom pattern takes priority.

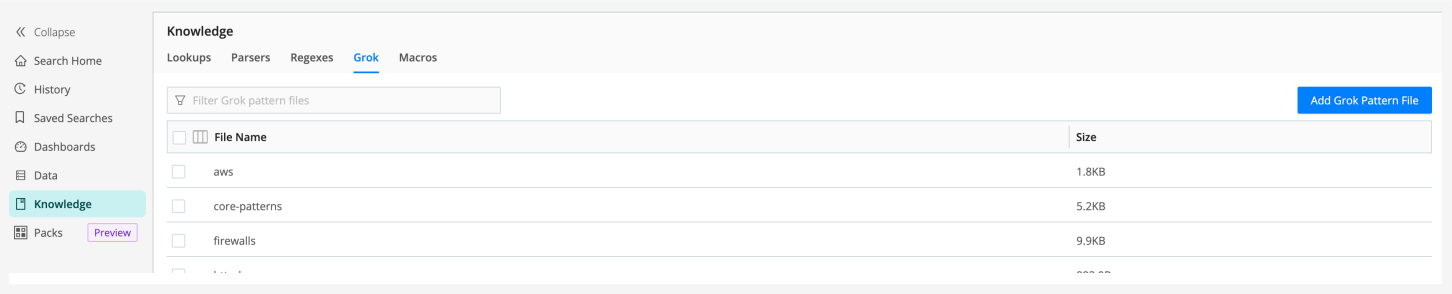
6.4. Grok Patterns

Use the built-in library of Grok patterns, or define your own.

A Grok pattern is a string of special characters and regular expressions that match data. Cribl Search uses Grok patterns in definitions for Parsers.

Cribl Search ships with many commonly used Grok patterns. The **Grok Patterns** page provides an interface for creating and editing them.

To open the **Grok Patterns** page, select **Knowledge** > **Grok Patterns**.



Grok Patterns page

For details about who can access and modify this resource, see Search Member Permissions.

Add a Grok Pattern

1. On the top right, select **Add Grok Pattern File**.
2. Give your custom Grok pattern a unique **File name**.
3. Enter your Grok pattern in the **Content** pane.
4. Select **Save**.

6.5. Macros

Reuse query text across different searches, by using Macros.

Macros are reusable query snippets that you can reference by name when writing your query, using the `${...}` syntax. They can contain any valid Cribl Search query text, including functions, operators, or literals.

For example, you can create a Macro called `externalVpcReject` that consists of the following text:

```
(action=reject srcip!=10.* srcip!=172.16.*)
```

Copy

...and then use it anywhere in a query, like this:

```
${externalVpcReject}
```

Copy

...which is equivalent to writing:

```
(action=reject srcip!=10.* srcip!=172.16.*)
```

Copy

With Macros, you can construct queries faster and keep them consistent, especially if you frequently rerun the same complex query in various configurations.

You can share Macros with other Members of your Organization. To share Macros more broadly, you can export and import them in Packs.

Macro Parameters

Macro parameters are numbered placeholders (`$1`, `$2`, `$3`, ...) that you fill in with different values each time you inject the Macro into a query. They work similarly to Bash positional parameters.

For example, you can create a Macro called `filterByActionAndAnswer` that consists of the following text:

```
// define three parameters
action=$1 answer=$2 $3
```

Copy

...and then use it anywhere in a query, like this:

```
// pass the same number of arguments, in the same order
${filterByActionAndAnswer, 'reject', 42, ips="172.16.*"}
```

Copy

...which is equivalent to writing:

```
action='reject' answer=42 ips="172.16.*"
```

Copy

You can pass any type of value as an argument (including other Macros with parameters). If an expression contains commas (`,`), wrap it in quotes, for example: `${myMacro, "one, two, three", 42}`.

View Macros Available to You

Certain Macros may already be available to you, depending on your organization's configuration:

1. In Cribl Search, go to **Knowledge** > **Macros**.
2. Here, you can search and filter the available Macros:
    - To see only Macros built into Cribl Search by default, select the **Cribl** filter at the top right.
    - To see only Macros that you created or that were shared with you, select the **Custom** filter.
    - To search for a specific Macro, type its ID in the **Filter Macros** box.
3. To see a Macro's details, select its row.

Use a Macro in a Query

You can insert a Macro anywhere in any query. Type `$` followed by the Macro's ID in curly braces `{}`:

```
${macroId}
```

Copy

If the Macro is part of a Pack, you'll need to include the Pack's ID as well. See: Use a Pack Macro.

For example, to insert Macros named `myMacro1` and `myMacro2`:

```
dataset=myDataset ${myMacro1}
   | limit 1000 ${myMacro2}
```

Copy

When a Macro's definition includes a number of parameters, you need to pass the same number of arguments, in the same order. Pass your arguments after the Macro ID, separated by commas. For example:

```
dataset=myDataset ${myMacro1, argument1, argument2, argument3}
```

Copy

If you reference a Macro inside a string or another literal (like regex), it will behave as plain text. For example:

```
// here, "bar" won't be expanded
| where foo="${bar}"
```

Copy

Cribl Search's autocomplete will suggest available Macros as you type.

Expand Macros Used in a Query

After you run a query that contains Macros, you can expand the Macros to show the text that each Macro resolved to. This way, you can see the query exactly as it was executed, which is helpful for debugging your searches.

1. Run a search.
2. After the search starts running, select **Details** under the query box.
3. In the **Key** column, **query** row, select **Expand Macros**. The query expands to show the text that each Macro resolved to during the search execution.

Create a Macro

To create a new Macro:

1. In Cribl Search, select **Knowledge**, then **Macros**.
2. At the top right, select **Add Macro**. The **New Macro** modal appears.
    - In **ID**, enter your Macro's ID. This is how you and others will reference the Macro in your queries. The ID must be unique among all Macros within your organization (even those unavailable to you). For details, see Naming Convention for Macros.
    - In **Description**, enter a brief description of the Macro's purpose.
    - In **Tags**, enter any tags that can help you and others find your Macro.
    - In **Definition**, enter the query text that the Macro will expand to when referenced in a query.
3. In the bottom right, select **Save**.



You can also clone an existing Macro by selecting its row, and then **Clone Macro**.

Naming Convention for Macros

As Macros can be shared with multiple Members of your organization, and they all share a single namespace, every Macro ID must be unique across the entire Cribl.Cloud Search tenant.

To manage your namespace in a way that makes the most sense for your organization, we suggest following a consistent naming convention.

For example, you can use the following convention: `<teamName>_<DatasetName>_<functionName>`. This might create IDs like:

- `ops_flowlogs_uniqueAddresses`
- `IRT_emailAnomalies_lookupName`

Edit a Macro

To edit a Macro:

1. Make sure you have the **Maintainer** Permission for the Macro.
2. In Cribl Search, select **Knowledge**, then **Macros**.
3. Select the row of the Macro that you want to edit. The Macro's details appear.
4. Edit the Macro's **Description**, **Tags**, and **Definition**. For details, see the Create a Macro section.
5. In the bottom right, select **Save**.

Delete a Macro

To delete a Macro:

1. Make sure you have the **Maintainer** Permission for the Macro.
2. In Cribl Search, select **Knowledge**, then **Macros**.
3. Select the row of the Macro that you want to delete. The Macro's details appear.
4. In the bottom left, select **Delete Macro**.
5. Type **DELETE**, and then select **Delete Selected Macro**.



If the Macro you want to delete belongs to a Pack, see Delete a Pack Resource.

Sharing Macros with Other Members

You can share your Macros with other Members of your organization, to build a library of useful functions and transformations.

Macro Permissions

You can share Macros with individual users or entire Teams, granting one of the two access levels, **Read Only** or **Maintainer**:

Permission	Read Only	Maintainer
View the Macro		
Use the Macro in a query		
Edit the Macro		
Delete the Macro		

When you create a Macro, you automatically become its **Maintainer**.

See Who Can Access Your Macros

By default, Macros that you create are available only to you and the Search Admin.

To see who exactly has access to a Macro:

1. In Cribl Search, select **Knowledge**, then **Macros**.

2. In the row of the Macro that you want to share, select **Share**. The **Macro Members and Teams** modal appears.
3. Here, you can see who has Permission for the Macro.

Share a Macro

To share a Macro with another Member of your organization:

1. In Cribl Search, select **Knowledge**, then **Macros**.
2. In the row of the Macro that you want to share, select **Share**. The **Macro Members and Teams** modal appears.
3. Search for the Member or Team that you want to share the Macro with.
4. Select the Permission level that you want to grant to the Member or Team.
5. Select **Add Access**.

6.6. Cribl Search Packs

Import, export, and share pre-built Cribl Search resources.

Why Use Packs

A Pack is a collection of preconfigured Cribl Search assets, such as ready-to-use Dashboards, saved searches, or lookup files. Packs can help you and your team get started faster, quickly see what Cribl Search is capable of, and easily move complex configurations across different environments.

What's in a Pack

A Pack can contain any of the following resources:

- Dashboards
- Saved searches
- Datatypes
- Macros
- Lookups

A Pack can't reference other Packs, but it can reference other resources within the same Pack. It can also reference global, non-Pack resources. For details, see Pack Dependencies and Packs Limitations.

Who Can Use Packs

Packs are available to Cribl Search **Admins** and **Editors**. For reference, see Search Member Permissions.

Admins and Editors can:

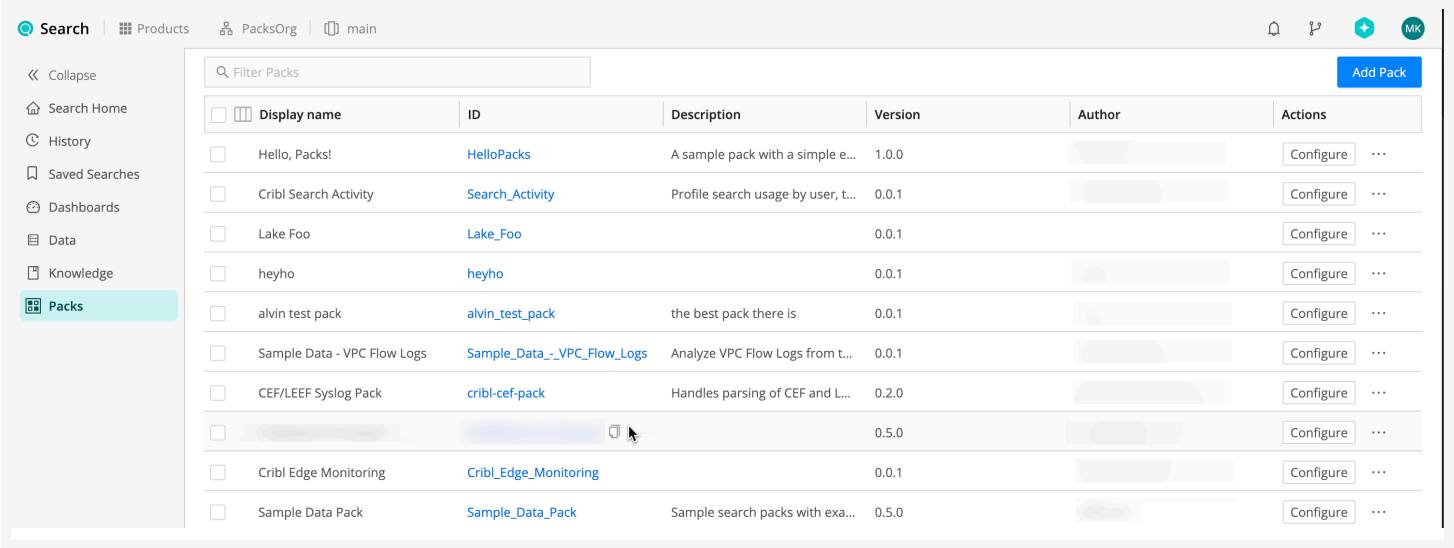
- View Packs installed in the Organization.
- Use a Pack's resources, if the Pack is set to allow global access.
- Get new Packs from the Cribl Packs Dispensary, or import them from a file, URL, or Git repo.
- Modify existing Packs, including adding more resources, or modifying metadata.
- Create new Packs from scratch, using the Cribl Search UI.
- Share Packs, allowing access to Pack resources, or exporting a Pack to a file.

View Packs

To see Packs installed in your Organization, go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.

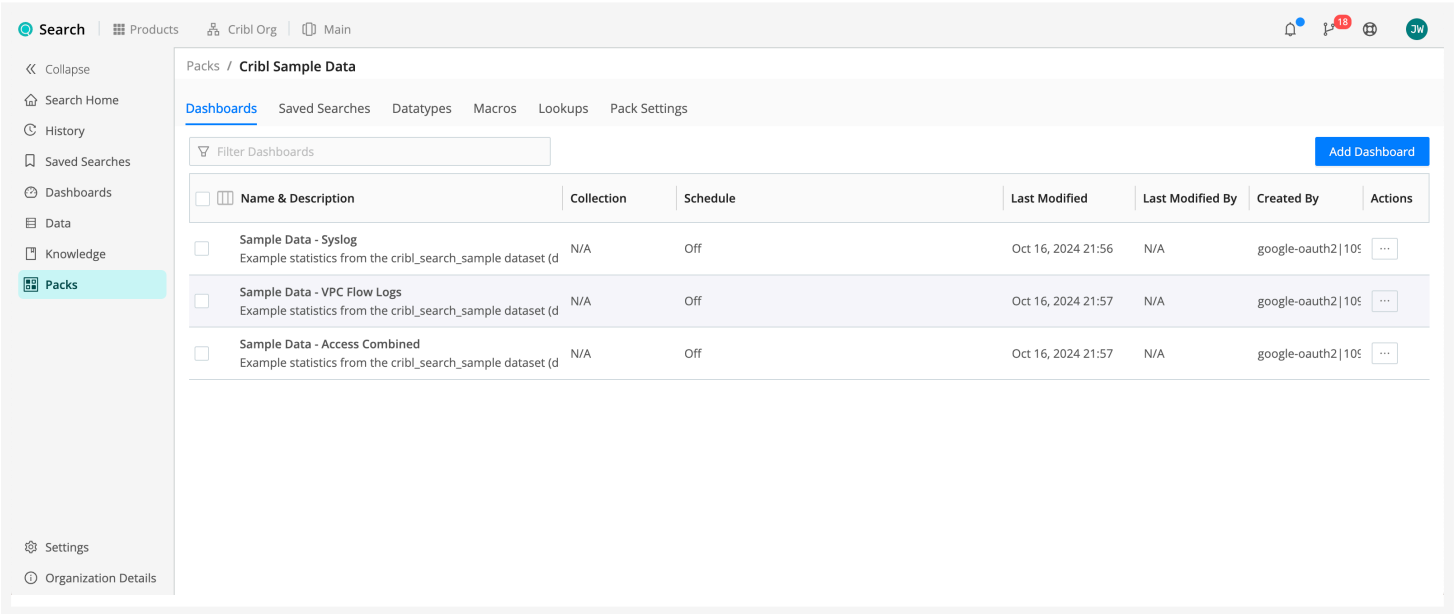
The **ID** column shows each Pack's ID, by which you can reference the Pack's resources.

You can hover over an ID and select to copy it to the system clipboard.



Packs page

Select a Pack to view the resources it contains. You can then switch among the upper tabs to view the Pack's **Dashboards**, **Saved Searches**, **Datatypes**, **Macros**, and **Lookups**.



Dashboards within a Pack

Use Packs

If a Pack is set to allow global access, you can use its resources similarly to any other resources in Cribl Search. For example, you can run saved searches included in the Pack, reference a Pack Macro in a query, or use a Pack lookup.

Use a Pack Macro

To use a Pack Macro from outside its Pack, make sure the Pack is set to allow global access. Then, reference the Pack's ID (`packId`) and the Macro's ID (`macroId`), in the following format:

```
${pack(packId).macroId}
```

Copy

For example, to use `myMacro` residing in a Pack called `cribl-sample-data`:

```
${pack(cribl-sample-data).myMacro}
```

Copy

Use a Pack Lookup

To use a Pack lookup from outside its Pack, make sure the Pack is set to allow global access. Then, reference the Pack's ID (`packId`) and the lookup's ID (`lookupId`), in the following format:

```
lookup pack(packId).lookupId
```

Copy

For example, to look for a common field in the event and in a `lookupTable` residing in `myPack`:

```
dataset=myDataset
 | lookup pack(myPack).lookupTable on commonField
```

Copy

You can't create or update Pack lookups with the `export` operator. For example, `| export to lookup pack(packId).lookupId` will **not** work.

Use a Pack Saved Search

To use a Pack saved search from outside its Pack, make sure the Pack is set to allow global access. Then, reference the Pack's ID (`packId`) and the saved search's ID (`savedSearchId`), in the following format:

```
packId.savedSearchId
```

Copy

Use a Pack Datatype

To use a Pack Datatype from outside its Pack, make sure the Pack is set to allow global access. Then, reference the Pack's ID (`packId`) and the Datatype's ID (`datatypeId`), in the following format:

```
packId.datatypeId
```

Copy

Get Packs

As a Cribl Search Admin or Editor, you can install Packs in your Organization by importing them from the following sources:

- Packs Dispensary
- File
- URL
- Git repo

You can also create a new Pack from scratch, or modify an existing Pack.

Add a Pack from Dispensary

The Cribl Packs Dispensary is a Cribl-hosted resource for you to find and share Packs. Cribl, our partners, and community users develop Packs and submit them to the Dispensary for easy sharing. Cribl tests submissions before publication, to ensure each Pack's quality, security guardrails, and stability.

You can install Dispensary Packs as follows:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select **Add Pack** > **Add from Dispensary**. The Packs Dispensary will open in a drawer.
3. Using the drawer controls, browse or search for the Pack(s) you want. (You can filter for only Packs **Built by Cribl**.)

4. Select any Pack tile to display its details page with its README. This will typically outline the Pack's purpose, compatibility, requirements, and installation.
5. To proceed, select **Add Pack** on this page.
6. That's it! You'll see a banner confirming that the Pack is now installed.

Import a Pack from File

To import a new Pack, or an updated version of an existing Pack, from your local filesystem:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select **Add Pack** > **Import from File**.
3. Select the file to import.
4. Optionally, give the Pack an explicit, unique **New Pack ID**. (For details about this option, see Upgrade a Pack.)
5. Select **Import** to confirm the import.



When you import Packs that were exported in `merge` mode, make sure to re-enter any required secrets to restore full functionality. This is necessary because Cribl Search deletes all encrypted fields during the export process, to ensure security and prevent the accidental sharing of sensitive information.

Import a Pack from URL

To import a Pack from a known, public or internal, URL:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select **Add Pack** > **Import from URL**.
3. Enter a valid URL for the Pack's source. (This field's input is validated for URL format, but not for accuracy.)
4. Optionally, give the Pack an explicit, unique **New Pack ID**. (See Upgrade a Pack.)
5. If the Pack matches an installed Pack's **Pack ID**, confirm that you want to **Overwrite** the existing Pack.
6. Select **Import** to confirm the import.

Import a Pack from Git

You can import a Pack from a known public or private Git repo.

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select **Add Pack** > **Import from Git**.

3. Enter the source repo's valid URL.

   This field's input is validated for URL format, but not for completeness or accuracy. When targeting a private repo, use the format: `https://<username>:<token/password>@<repo-address>`. Public repos need only `https://<repo-address>`.

4. Optionally, give the Pack an explicit, unique **New Pack ID**. (See Upgrade a Pack.)
5. If the Pack matches an installed Pack's **Pack ID**, confirm that you want to **Overwrite** the existing Pack.
6. Optionally, enter a **Branch or tag** to filter the import source using the repo's metadata. You can specify a branch (such as `main`) or a tag (such as a release number like `0.5.1`).
7. Select **Import** to confirm the import.

Pack Dispensary GitHub Repo

A particularly useful public repo is the Cribl Pack Dispensary on GitHub. This repo was established prior to the Cribl-hosted Cribl Packs Dispensary site, and it is a place to collaborate on developing Packs prior to submitting them to the newer site.

You can install Packs directly from this repo using the Import from Git option. However, if you prefer, you can click through to any Dispensary repo's release page, download the corresponding `.crbl` file, and then upload the file into Cribl Search.

Modify Packs

As a Cribl Search Admin or Editor, you can modify existing Packs:

- Add a resource to a Pack
- Modify a Pack's metadata
- Set a Pack's logo
- Upgrade a Pack
- Overwrite a Pack

Add a Resource to a Pack

You can fill a Pack with Dashboards, saved searches, Datatypes, Macros, and lookups.

For example, to add a Dashboard to a Pack:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select the Pack you're interested in. Use the search box to filter the list.
3. From the **Dashboards** tab, select **Add Dashboard** to access the two options listed below:

- **Add Existing**: Opens an **Add Existing Dashboards** modal, where you can select one or more Dashboards that are available to you, and add them to this Pack. (The list displayed in this modal will, by design, omit Dashboards already included in the Pack.)
- **Add New**: Opens a **New Dashboard** modal, where you can configure a new Dashboard using the controls covered in Create a Dashboard.

For details on how to reference different types of Pack resources, see Pack Dependencies.

Delete a Pack Resource

To remove a Pack resource (Dashboard, saved search, Datatype, Macro, or lookup), you need to delete it from within its Pack.

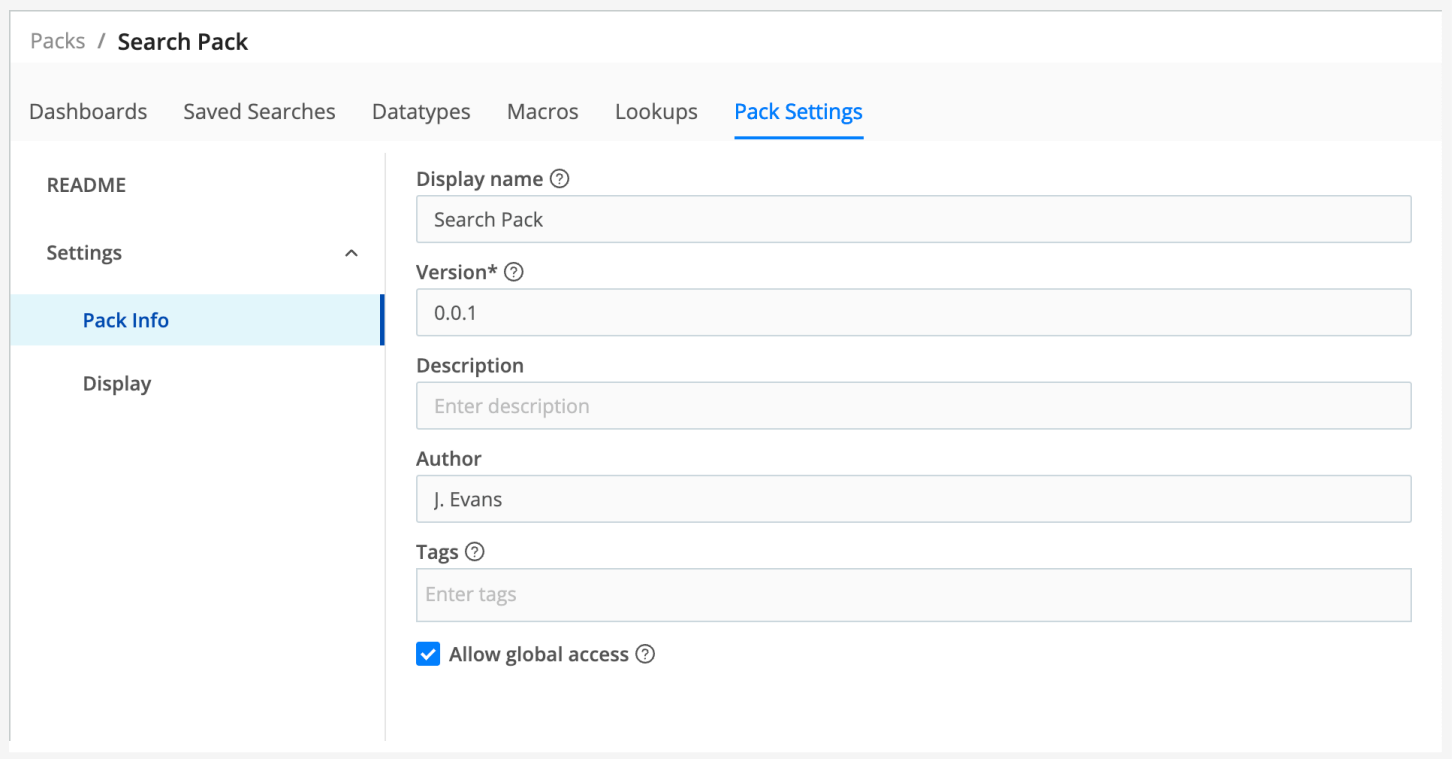
For example, to delete a Pack Dashboard:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select the Pack you're interested in. Use the search box to filter the list.
3. From the **Dashboards** tab, select the checkbox of the Dashboard to delete.
4. Select **Delete selected Dashboards**.

Modify a Pack's Metadata

You can modify a Pack's metadata, such as display name, version, description, and others.

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select the Pack you're interested in. Use the search box to filter the list.
3. Select the **Pack Settings** upper tab to expose two options on the left:
   - **README**: Here, you can edit extended information about the Pack, including its purpose, requirements (prerequisites), and usage instructions.
   - **Settings**: Here, you can update the Pack's **Display name**, **Version** number, short **Description**, **Author**, and arbitrary **Tags**. You can also control access to the Pack.
4. Save your changes.



Pack Settings

Set a Pack's Logo

We recommend adding a logo to each custom Pack, to visually distinguish the Pack UI from the surrounding Cribl Search UI (as well as from other Packs).

You can upload a `.png` or `.jpg`/`.jpeg` file, up to a maximum size of 2 MB and 350x350 px. Cribl recommends a transparent image, sized approximately 280x50 px.

To set a logo for an existing Pack:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select the Pack you're interested in. Use the search box to filter the list.
3. Go to **Pack Settings** > **Display**.
4. Select **Choose File** to upload a new logo, and confirm.

Upgrade a Pack

To upgrade an existing Pack to a newer version:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Open the Actions ⋯ menu on a Pack's row, and then select **Upgrade**.

When upgrading a Pack, we recommend that you:

- Initially, import the updated Pack with a new ID, and with a new **Display name** that includes the version number. (For example, `Cribl Search Activity 0.5`.) This enables you to review and adjust new behavior against currently–deployed configurations.
- Do a side–by–side comparison of the previous and new versions of the Pack. Remember to review the new Pack's README.
- If the Pack includes any user–modified versions of default Cribl Search Knowledge resources (for example, lookups): Be sure to copy the modified files locally for safekeeping, before upgrading the Pack. After you install the upgrade, copy those files back to the upgraded Pack, overwriting the default versions in the Pack.
- Test, test, and test!
- If you're confident in the new version, you have the option to now overwrite the existing Pack, using the same ID.
- Commit and Deploy.

Overwrite a Pack

Each Pack that you install within a given Organization must have a unique ID. The ID is based on the internal configuration of the Pack – not on its Display name, nor on its parent file name. You cannot share an ID between two (or more) installed Packs.

If you import a Pack whose internal ID matches an installed Pack – whether an update, or just a duplicate – you'll be prompted to assign a unique **New Pack ID** to import it as a separate Pack.

Alternatively, you have the option to **Overwrite** the installed Pack, reusing the same ID.



With the **Overwrite** option, the imported Pack completely replaces your existing Pack. We recommend creating a local backup copy of the existing Pack first.

If you've modified an installed Pack, Cribl Search will block the overwrite of the Pack, to prevent deletion of your locally created resources.

Create Packs

As a Cribl Search Admin or Editor, you can create a new Pack from scratch, using the Cribl Search UI.

You can also create new Packs by importing, modifying, upgrading, or overwriting existing Packs.

When adding resources to a Pack, learn about how Pack dependencies work.

Create a Pack from UI

You can create a new Pack from scratch, using the Cribl Search UI.

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select **Add Pack** > **Create Pack**.
3. Enter the following details:
    - **Display name** is an arbitrary, human-readable name to identify this Pack.
    - **Pack ID** is required, and must be unique within your Cribl Search Organization.
    - **Version** is required, and identifies this Pack's own versioning.
    - **Description** and **Author** are optional identifiers.
    - **Tags** are optional, arbitrary labels that you can use to filter/search and organize Packs.
    - If you want to allow global access to this Pack, check **Allow global access**. This will make all of the Pack's resources visible and usable from anywhere in the current Workspace.
4. Select **Save**.
5. On the **Packs** page, select the new Pack's row to open the Pack.
6. Configure the resources you want to pack up, using the the standard Cribl Search controls for Dashboards, saved searches, Datatypes, Macros, or lookups. As you save changes in the UI, they're saved to the Pack.

   If you want to reference resources that reside outside the Pack, learn about how Pack dependencies work.
7. Set the pack's logo, to visually distinguish the Pack UI from the surrounding Cribl Search UI (as well as from other Packs).

Share Packs

As a Cribl Search Admin or Editor, you can share Packs in the following ways:

- Allow access to all of a Pack's resources, making them usable from anywhere in the current Workspace.
- Move a Pack resource out of its Pack, to make the resource available globally.
- Move a resource into a Pack, to make the resource available only within that Pack.
- Move a Pack resource to another Pack.
- Clone a Pack resource, to play around with it without affecting the original.
- Export an entire Pack to a file, so that you can share it with other Organizations.



To move a sample resource shipped by Cribl, you need to clone it first.

Allow Global Access to a Pack

To make all of a Pack's resources visible and usable from anywhere in the current Workspace:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select the Pack you're interested in. Use the search box to filter the list.
3. Go to **Pack Settings** > **Pack Info**.
4. Check **Allow global access**, and confirm with **Save**.

Move a Resource Out of a Pack

To make a resource (Dashboard, saved search, Datatype, Macro, or lookup) available outside its original Pack, you can move that resource to your Organization's global level.

For example, to move a Dashboard out of its Pack:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select the Pack whose resource you want to move.
3. On the **Dashboards** tab, open a Dashboard's Actions ⋯ menu.
4. Under **Move**, select **All Dashboards**.

Move a Resource Into a Pack

You can move a global resource (Dashboard, saved search, Datatype, Macro, or lookup) into a Pack. This makes the resource available only within that Pack, and removes it from the global context.

For example, to move a Dashboard into a Pack:

1. Go to the **Dasbboards** page in Cribl Search: On the top bar, select **Products** > **Search** > **Dashboards**.
2. Open the Actions ⋯ menu of the custom Dashboard you want to move.
3. Under **Move**, hover over **Packs**, and select a Pack.



Moving a Dashboard into a Pack changes the Dashboard's **Created by** and **Last Modified by** indicators to match your identity. The move also discards any sharing Permissions that were set on the original Dashboard. Share the Dashboard from its new location to add new Permissions.

Move a Resource to Another Pack

You can move a resource (Dashboard, saved search, Datatype, Macro, or lookup) from one Pack to another.

For example, to move a Dashboard to another Pack:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select the Pack whose resource you want to move.

3. On the **Dashboards** tab, open a Dashboard's Actions ⋯ menu.
4. Under **Move**, hover over **Packs**, and select the Pack into which you want to move the resource.

Clone a Pack Resource

To play around with a Pack resource (Dashboard, saved search, Datatype, Macro, or lookup) without affecting the original, you can create a clone.

Cloned resources are saved to your current UI location. For example, if you clone a Dashboard while viewing its Pack, the clone is saved as part of the Pack. If you clone the same Dashboard while viewing the global **Dashboards** page, the clone is saved as a global resource.

For example, to clone a Dashboard into its Pack:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. Select the Pack whose resource you want to clone.
3. On the **Dashboards** tab, open a Dashboard's Actions ⋯ menu.
4. Select **Clone**. The copy is saved to the Pack, and its visibility outside the Pack depends on the global access setting.

To clone a Pack Dashboard into the global **Dashboards** page:

1. Go to the **Dasbboards** page in Cribl Search: On the top bar, select **Products** > **Search** > **Dashboards**.
2. Find the Pack Dashboard you want to clone. Look for the Pack's name in the **Pack** column.
3. Open the Dashboard's Actions ⋯ menu.
4. Select **Clone**. The copy is saved to the global **Dashboards** page.

Export a Pack

To download an existing Pack as a single file, for sharing with other Organizations:

1. Go to the **Packs** page in Cribl Search: On the top bar, select **Products** > **Search** > **Packs**.
2. In the row of the Pack you're interested in, select the Actions ⋯ menu, then **Export**.
3. In the resulting **Export Pack** modal, the filename defaults to the Pack's current ID, with the version number appended. You can modify this, as desired, before confirming the export.

If the Pack references any resources residing outside the Pack (for example, a global lookup file), you'll see a warning, saying `This Pack has external dependencies.`



When you export Packs in `merge` mode, make sure to re-enter any required secrets after importing the Pack, to restore full functionality. This is necessary because Cribl Search deletes all encrypted fields

during the export process, to ensure security and prevent the accidental sharing of sensitive information.

Pack Dependencies

When you're creating a new Pack, or adding resources to an existing Pack, you can reference resources residing outside that Pack ("global" resources). Or, the other way around: you might want to reference a Pack resource from outside the Pack. Keep in mind the following guidelines about such dependencies.

Pack resources:

- Can reference other resources within the same Pack.
- Can reference global, non-Pack resources.
- Can't reference other Packs.

Global resources, in turn, can reference a Pack's resources only when the Pack is set to allow global access.

Reference a Macro

To reference a **global** Macro from the **global** context (for example, simply using a Macro in a query), use the standard Macro syntax.

```
${macroId}
```

Copy

To reference a **Pack** Macro from the **global** context, make sure the Pack is set to allow global access. Then, specify the Pack's ID with `pack(packId)`, like this:

```
${pack(packId).macroId}
```

Copy

To reference a **global** Macro **from within a Pack**, specify the global Pack called `cribl`.

```
${pack(cribl).macroId}
```

Copy

To reference a **Pack** Macro **from within the same Pack**, you can use the standard Macro syntax:

```
${macroId}
```

Copy

A Pack can't reference a Macro residing in another Pack.

Reference a Lookup

To reference a **global** lookup from the **global** context (for example, simply using the `lookup` operator in a query), use the standard lookup syntax. For example:

```
dataset=myDataset
| lookup lookupId on commonField
```

Copy

To reference a **Pack** lookup from the **global** context, make sure the Pack is set to allow global access. Then, specify the Pack's ID with `pack(packId)`, like this:

```
dataset=myDataset
| lookup pack(packId).lookupId on commonField
```

Copy

To reference a **global** lookup **from within a Pack**, specify the global Pack called `cribl`.

```
dataset=myDataset
| lookup pack(cribl).lookupId on commonField
```

Copy

To reference a **Pack** lookup **from within the same Pack**, you can use the standard lookup syntax:

```
dataset=myDataset
| lookup lookupId on commonField
```

Copy

The following limitations apply:

- A Pack can't reference a lookup residing in another Pack.
- You can't create or update Pack lookups with the `export` operator.

Reference a Saved Search

To reference a **global** saved search from the **global** context (for example, when basing a Dashboard visualization on an existing saved search), use the saved search's ID:

```
savedSearchId
```

Copy

To reference a **Pack** saved search from the **global** context, make sure the Pack is set to allow global access. Then, specify the Pack's ID (`packId`) like this:

```
packId.savedSearchId
```

Copy

To reference a **global** saved search **from within a Pack**, specify the global Pack called `cribl` in the following format.

```
cribl.savedSearchId
```

Copy

To reference a **Pack** saved search **from within the same Pack**, you can use the standard syntax:

```
savedSearchId
```

Copy



A Pack can't reference a saved search residing in another Pack.

Reference a Datatype

To reference a **global** Datatype from the **global** context (for example, when adding a Dataset), you normally use the Datatype's ID:

```
datatypeId
```

Copy

To reference a **Pack** Datatype from the **global** context, make sure the Pack is set to allow global access. Then, specify the Pack's ID (`packId`) like this:

```
packId.datatypeId
```

Copy



You can't reference a Datatype **from within** a Pack.

Packs Limitations

Here's what you can't do with Packs:

- Packs can't reference other Packs.
- You can't configure Notifications on a saved search that resides within a Pack.
- You can't create or update Pack lookups with the `export` operator. For example, `| export to lookup pack(packId).lookupId` will **not** work.

7. Manage Cribl Search And Cribl.Cloud

Manage Cribl Search, your Cribl.Cloud Organization, and its Members, Permissions, usage, and billing.

Your Cribl Search instance is a part of a Cribl.Cloud Organization. This section covers how to manage the Organization itself, your Plan (credits and billing), and several aspects of access management and resource sharing.

Management Options by Permission

If you're an Organization Owner or Organization Admin, you can manage various aspects of your Cribl.Cloud Organization. You also automatically inherit all the Permissions of a Search Admin.

If you're a Search Admin, you can control most aspects of your Cribl Search environment.

For an overview of who can do what, see:

- Organization Member Permissions
- Search Member Permissions

Cribl.Cloud Organizations and Members

Learn how to manage your Cribl.Cloud Organization in the following sections:

- Manage Cribl.Cloud Organizations: Manage your Organization, configure network settings, and control user access.
- Manage Cribl.Cloud Members: Invite Cribl.Cloud Members, configuring their Permissions at the Organization or Cribl Search level.
- Cribl.Cloud Enterprise Plan: Explore the benefits of the Cribl.Cloud Enterprise Plan.
- FinOps Center: Check your plan and credits, track data usage, and download invoices.

Workspaces

Workspaces are isolated environments within your Cribl.Cloud Organization, each with its own Leader and access control. To make the most of them, see the following sections:

- Configure Workspaces: Add, edit, or delete your Cribl.Cloud Workspaces.
- Members and Teams: Fine-tune access control within your Workspaces, by configuring Members and Teams.

Usage Settings

Manage Notification targets, Usage Groups, and capacity limits, in the Cribl Search Settings.

Sharing

Share your Datasets, Dataset Providers, search results, and Dashboards with other Search Members.

7.1. Cribl.Cloud Access And Usage

Get an overview of the Cribl.Cloud environment that hosts Cribl Search and complementary Cribl products.

Cribl.Cloud provides a dedicated environment that hosts Cribl Search, Cribl Stream, Cribl Edge, and Cribl Lake, enabling you to start using Cribl products immediately. Cribl manages and scales infrastructure on your behalf.

Cribl.Cloud Resources

See these topics that cover creating and administering your Cribl.Cloud account, Organization, and authorization:

- Quick Setup: Register a Cribl.Cloud Organization to get started with Cribl Search.
- Manage Cribl.Cloud: Learn to switch between Cribl.Cloud Organizations and configure your Organization details.
- Manage Cribl.Cloud Members: Manage Cribl.Cloud access control – Members, invites, and Permissions.
- FinOps Center: Track your credit usage and billing across all Cribl products.

Cribl.Cloud Enterprise Plan

With a Cribl.Cloud Enterprise plan, you gain additional flexibility and options. These options include:

- Full role-based access control over Member Permissions and resources.
- Integrating external SSO authentication.
- Configuring and managing multiple Stream Worker Groups and Edge Fleets.
- Hybrid Stream Worker Groups.



For other Enterprise features, see Pricing.

7.1.1. Manage Cribl.Cloud Organizations

Manage your Cribl.Cloud Organization, configure network settings, and control user access.

Switch Organizations

When you own or are a Member of multiple Cribl.Cloud Organizations, you can switch between the Organizations:

1. In the top bar, hover over the name of your Organization.
2. In the drop-down, select the Organization you want to switch to.

Alternatively, in the drop-down you can select **Manage Organizations**. You will move to an Organization landing page, displayed to an Organization's Owners and Admins, where you can view details of all your Organizations, switch to any of them, add a new one or delete one, if you have proper permissions.

Configure Organization Details

To customize your Organization details, in the sidebar, select **Organization**.

You can modify the following information:

Field name	Description
Organization Name	A "friendly" name for your Organization. Upon signing in, Members will see this alias above the Organization ID on the Select Organization page.
Description	Add further details about your Organization.
Opt in to beta features	If displayed, this check box enables access to new options that Cribl has not yet made generally available. As with all beta features, expect some instability in exchange for advancing to the cutting edge of your Cribl.Cloud.

Select **Save** to immediately apply your changes.

You can check additional information about your Organization, including the AWS Cloud region in which it is located, the software version, and license, by selecting **Organization Details** at the bottom of the sidebar.

Get Support

In the top bar, select the ⛑ **Support** icon to access a set of links with additional resources about Cribl Search, including:

- Sandboxes (free, interactive tutorials on fully hosted integrations)
- Cribl University
- Concept/demo videos

Refer to Cribl Status for information about Cribl.Cloud availability, including current and historical uptime data and incidents, split by component and AWS Region.

You can also subscribe there to incident notifications via email, text, Slack, or RSS.

Manage Personal Settings

Use the **Account** menu at the top right of the Cribl.Cloud portal to configure your personal settings.

Select **My profile** to update your personal information. Here, you can also reset the configured biometric authentication. Do this, if you want to re-configure the authentication, or if you don't want to be prompted to log in through fingerprint or face recognition.

7.1.2. Manage Billing In FinOps Center

Track your credit usage and billing across all Cribl products.

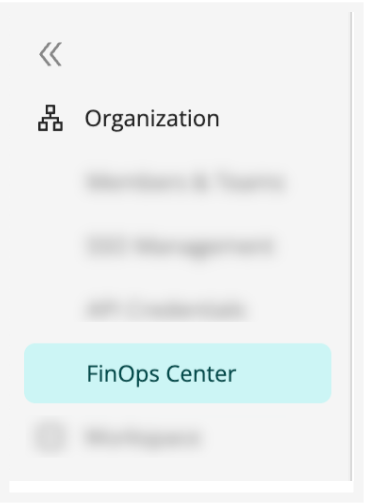
The Cribl FinOps Center is designed to simplify the process of tracking your product usage and credit consumption. You can also download finalized invoices and export them to external spend analysis products.

For more information about Cribl.Cloud billing and pricing, see our Cribl Pricing page.

Access Your Billing Information

The **FinOps Center** page is a single source of truth for your Cribl.Cloud Organization billing and usage data. To access your credits usage and invoice:

- From the Cribl home page, select **Organization** from the sidebar, then select **FinOps Center**.
- From within a product, select the top bar's **Products** button, then select **Cribl** > **Organization**. From the resulting sidebar, select **FinOps Center**.



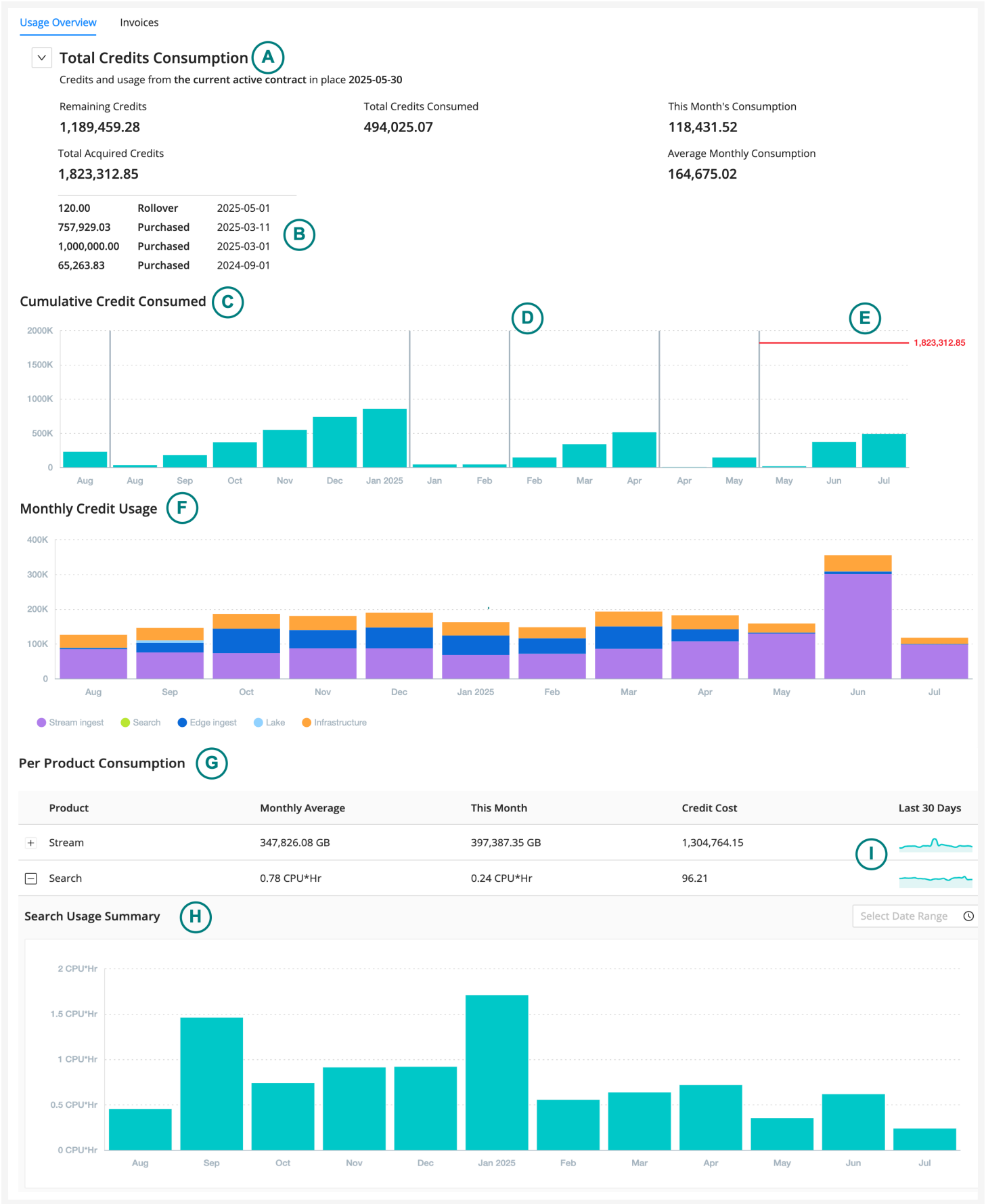
Organization sidebar menu

On the **FinOps Center** page, you can:

- Access, download, and export your invoices in CSV and JSON format.
- Understand your billing data at a glance.

Explore the FinOps Center

Select the **Usage Overview** tab for a holistic view.

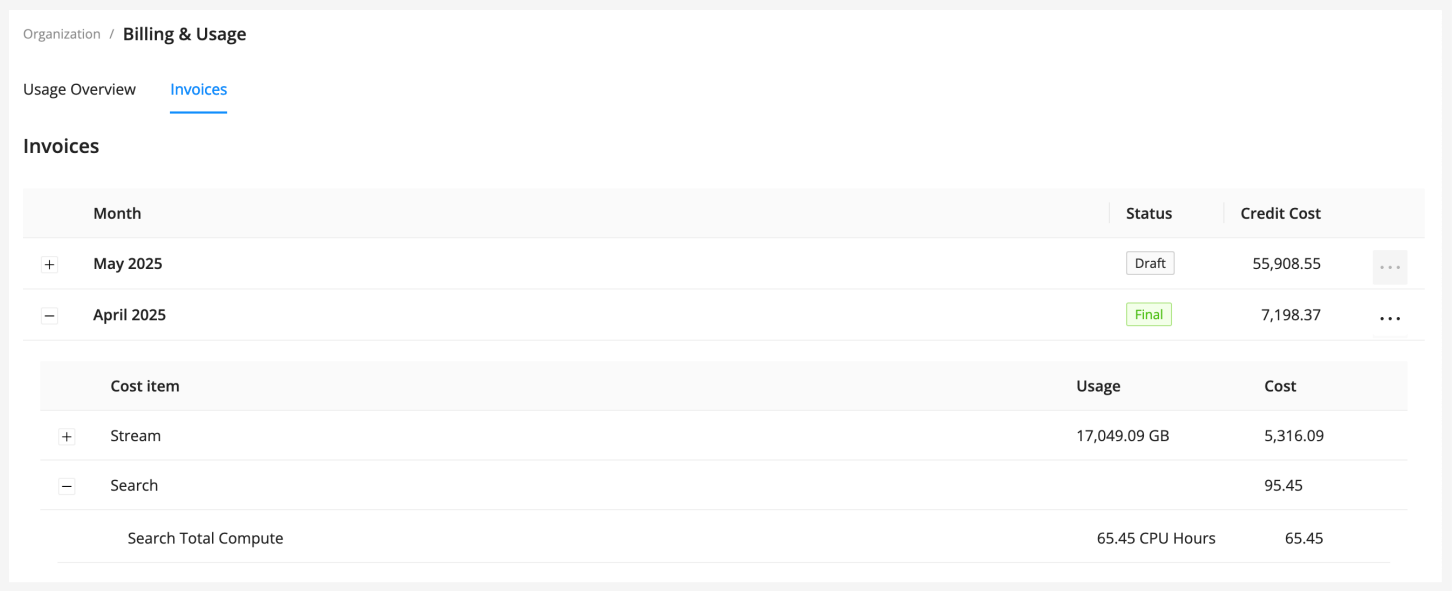


Explore the Cribl FinOps Center

Callout	Area	Description
A	**Total credits consumption**	View an overview of your credits for the current active contract. Twirl open the arrow to reveal a credit ledger, including any credits that rolled over from your last contract period.
B	**Cumulative credit consumption**	View a column chart containing each month's credit consumption across all products.
C	**Contract renewal indicator**	The vertical grey line indicates the point at which your Cribl contract was renewed. Because credits are consumed on a per-contract basis, you might see a drop in consumed credits directly after the renewal indicator, even if it is mid-month.
D	**Date range**	Use the date range selector to view usage and credits consumed in a specific date and time period.
E	**Total acquired credits**	The horizontal red line indicates total acquired credits in relation to consumed credits, including refunds and rollovers.
F	**Monthly usage**	View a column chart containing monthly usage by product. Hover over each month to view details.
G	**Per product consumption**	Find detailed, per-product credit consumption information in this section. Expand a product name for the single product usage view.
H	**Product usage summary**	View monthly usage for the expanded product.
I	**Sparklines**	View usage changes at-a-glance with the per-product sparkline chart.

Understand Your Monthly Invoice

The monthly invoice offers details about the subscription quota and total compute costs you have incurred in the current and previous months, along with a cost item breakdown for more granularity.



Monthly invoice information

To view your monthly Cribl invoice, make sure you're logged in to your Cribl account and you've selected your Organization.

1. From the sidebar, select **FinOps Center**.
2. The **Invoices** tab is where you'll find your monthly invoices.
    - Each invoice is ordered by month, most recent to oldest.
    - The invoice status will be **Draft** for the current month until your billing period ends, as defined in your contract.
    - You can download **Final** invoices as JSON or CSV using the ⋯ menu.
3. To view the details of each monthly invoice, select the plus [+] button.

Here, you'll see the breakdown of credits consumed by product and infrastructure:

Product	Cost item
Cribl Search	• Search Subscription (flat-rate pricing) • Search Total Compute (CPU x Hours)
Cribl Lake	Cribl Lake Total Storage (dataset size volume per GB on an assumed 30 day data retention schedule)

Product	Cost item
Cribl Stream	• Cloud Worker Group Net Data Ingest • Hybrid Workers GBs Received • Connected Stream GB In (on-prem Leaders that are passing usage data to Cribl.Cloud)
Cribl Edge	• Edge Node GBs Ingress • Connected Edge GBs In

Download Invoices

You can download your monthly Cribl.Cloud invoices as CSV or JSON files. Then, use them in external tools to help with budget planning and analytics or store them for historical purposes.



You can only download invoices with a **Final** status. **Draft** invoices can't be downloaded.

To download an invoice:

1. In the sidebar, select **Organization**, then **FinOps Center**.
2. On the **FinOps Center** submenu, select **Invoices**.
3. Hover over Actions ⋯ for the invoice you want to download.
4. Select **Download CSV** or **Download JSON**.

Frequently Asked Questions

Here are answers to some common questions about usage and billing:

How Is Usage Calculated?

For Cribl Search, your total spend equates to the amount of compute resources used to deliver search results. We charge in compute hours, but most searches take less than a second.

For Cribl Lake, usage equates to the amount of compressed data stored.

For Cribl Stream and Cribl Edge, your total spend equates to data ingest plus the infrastructure resources needed for processing your data. We charge on ingress, but you never pay for egress, so you can send your processed data to as many receivers as you'd like.

How Is Cost Calculated?

In Cribl.Cloud, one credit equals one US dollar. Each of our four products (Cribl Stream, Cribl Edge, Cribl Lake, Cribl Search) consumes credits at a different rate. For a detailed explanation, see our Cribl Pricing Guide or contact Cribl Sales.

Here's a quick summary of Cribl.Cloud credit consumption:

Product	General Calculation
Cribl Stream	Ingest pricing (infrastructure pricing based on region): • 0.32 Credits/GB per Cloud Worker • 0.26 Credits/GB per Hybrid Worker
Cribl Edge	Ingest pricing: • 0.21 Credits/GB per Edge Node
Cribl Search	Compute pricing: • 1 Credit per CPU hour If a Search is entirely satisfied by one or more Lakehouses, you are not billed Search CPU hours.
Cribl Lake	Storage pricing based on 30-day retention: • .05 Credits/GB if Cloud Worker is managed by Cribl • .02 Credits/BG if self-managed
Lakehouse	Lakehouse billing begins as soon as Cribl provisions the Lakehouse, shortly after you select **Save**. You're billed based on how long the Lakehouse has run. Price per month depends on the Lakehouse capacity you purchased. For more details about Lakehouse provisioning, see our Lakehouse docs.

How Often Is Credit Data Refreshed in Cribl.Cloud?

Cribl.Cloud refreshes credit data every five minutes to provide you with the most accurate information about your usage and cost.

7.2. Workspaces

Use Workspaces to create isolated environments within your Cribl.Cloud Organization.

Workspaces offer a multi-tenancy capability, enabling you to create multiple isolated instances in your Cribl.Cloud Organization. This lets you strengthen your security, and fulfill compliance and isolation requirements.

Each Workspace offers a dedicated virtual private cloud (VPC) that acts as a separate environment within your Organization. Workspaces do not share product configurations, resources, or data flows, and each has a separate Leader Node. They do share a layer of access control (Organization-level Permissions and SSO configuration), as well as API credentials, licenses, and billing.

A centralized interface allows you to manage all your Workspaces and control access for Members and Teams to individual Workspaces.

An example use case scenario for creating multiple Workspaces is setting up separate environments for different business units in an enterprise. Data management, security, development, and any other units can have their own federated Workspaces, with completely separate Members and Permissions lists. This way, they can work in isolation without risk of interference and with increased focus.

Limitations

You can create up to five Workspaces per Organization.

If you need more Workspaces in your Organization (up to 50), contact Cribl Support to discuss extension options.



Workspaces require certain a Cribl.Cloud plan. For details, see Pricing.

View Workspace Details

Workspaces provide a set of information that you can use when configuring data flow through their Edge Nodes.

View Access Details

To view detailed access information of your Workspace, in the sidebar, select **Workspace** and then **Access**.

This page presents a summary of information about your Cribl.Cloud Organization, as well as the **Cribl.Cloud URL** for the current Workspace.

Cribl.Cloud URL

Cribl.Cloud URL is a static address associated with the load balancer that is in front of the Leader. Hybrid Stream Workers will connect to this address on port 4200, while the Leader UI is served from this address on port 443.

You can use this URL for certain API calls and certain Collection operations coordinated by the Leader.

Static External IPs for the Leader

The **Leader NLB IPs** field in **Workspace** > **Access** lists the IPs for the Leader Network Load Balancers associated with the Workspace.

Typically, only 2 of the 3 addresses will be active (returned by DNS) at any time, while the inactive address is swapped in during infrastructure maintenance events.

You can add all those IPs to your firewall allowlist to ensure that the load balancers are accessible across your hybrid deployment.

Get ARN

To get the ARN (Amazon Resource Name) for your Workspace, in the sidebar, select **Workspace** and then **Trust**.

You can copy and paste the **Worker ARN**s listed in the **Trust** page to attach a Trust Relationship to an AWS account's IAM role. Use the **Group** drop-down to display the ARN for any Group of Cribl-managed Stream Worker Groups in Cribl.Cloud.

Attaching a Trust Relationship enables the `AssumeRole` action, providing cross-account access. For usage details, see the AWS Cross-Account Data Collection topic's **Account B Configuration** section.



This option applies only to your Cribl-managed Stream Workers in Cribl.Cloud. You cannot use this technique to enable access to customer-managed hybrid Workers.

Set Up ACL

To set up Access Control List (ACL) Rules, in the sidebar, select **Workspace** and then **Access Control List**.

ACL Rules (IPv4 CIDR ranges) let you restrict data sent to your data sources. The Rules you define here are global to all Cribl-managed Worker Groups in Cribl.Cloud in the current Workspace. You can set up a maximum of 6 ACL Rules.

The default `0.0.0.0/0` rule (modifiable) imposes no limits. End a rule with `/32` to specify a single IP address, or with `/24` to enable a whole CIDR block from `x.x.x.0` to `x.x.x.255`.

Select **Save** after adding, modifying, or removing rules. Each change takes up to 5 minutes to propagate. Cribl.Cloud will display a banner, notifying you that rules edits are temporarily disabled to prevent conflicts. A successful update proceeds silently – you will not see a confirmation message.



The **ACL** options apply only to Cribl-managed Stream Workers in Cribl.Cloud. You cannot use them to set access rules on customer-managed hybrid Workers.

7.2.1. Configure Workspaces

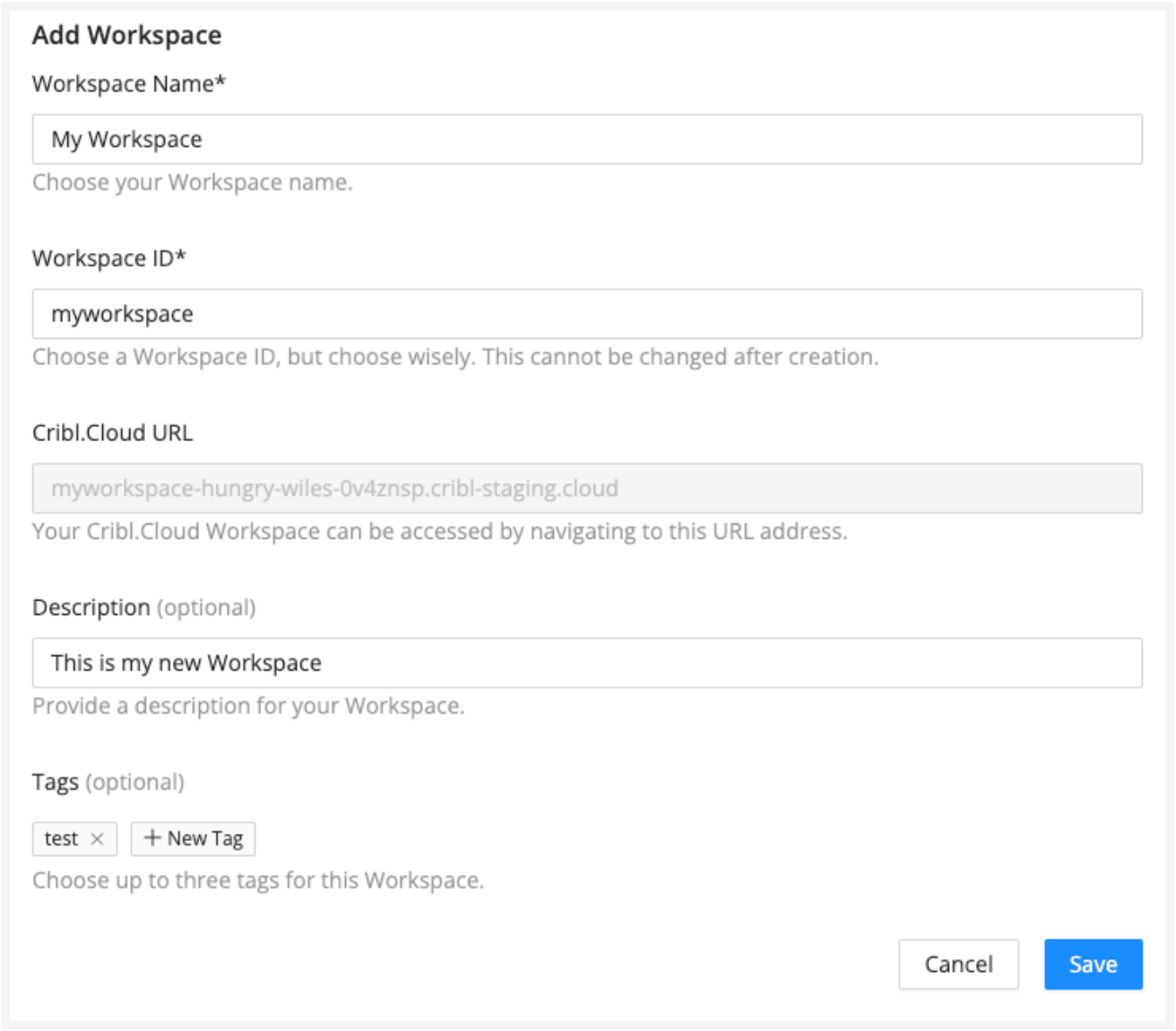
Add, edit, or delete your Cribl.Cloud Workspaces.

Add New Workspace

To create a new Workspace:

1. On the top bar, select your current Workspace name. This opens a modal with a list of all Workspaces to which you have access.
2. Select **Add Workspace**.
3. Fill the form with the following information:

Field	Description	Notes
Workspace Name	Human-readable name for the Workspace.	
Workspace ID	Internal ID for the Workspace. This ID will be part of the Workspace's Cribl.Cloud URL.	**Cannot be changed** once the Workspace is created.
Description	Additional Workspace description.	Optional.
Tags	Up to 3 tags you can use to filter Workspaces.	Optional.



Creating a new Workspace

4. Confirm by selecting **Save**.

The Workspace can take up to several minutes to spin up. While the process continues, you can navigate away and keep working with your Cribl.Cloud instance as usual.

Edit or Delete a Workspace

To edit a Workspace:

1. On the top bar, select your current Workspace name. This opens a modal with a list of all Workspaces you have access to.
2. Next to the Workspace you want to edit, select **Settings** in the **Actions** column to open the **Details** page for the Workspace.
3. On the **Details** page, you can change the name of the Workspace, description, and tags. However, you can't change the Workspace ID after it has been created.

4. Select **Save**.

For all Workspaces except `main`, you can select **Delete Workspace** on the **Details** page to delete it. You will need to manually type the name of the Workspace as a security measure.



It is not possible to delete the `main` Workspace.

Only Members with the Organization Owner Permission can delete Workspaces.

7.2.2. Configure Workspace Teams

Fine-tune access control within your Workspaces, by configuring Teams and their Members.

One of the main benefits of Workspaces is to efficiently provide granular control over which users have access to which parts of your Cribl Organization. You manage access by assigning Permissions to individual Members and Teams within a Workspace.



If a Member has overlapping assigned Permissions, then their highest Permission is active.

Members

When inviting a new user to your Organization, configure Workspace-level Permissions under **Workspace Access**.

- Select a Member Permission in the **Permission** drop-down next to the Workspace.
  When you choose the **Member** Permission, you will next configure access for each Cribl product separately:
- Alternatively, select the check box next to one or more Workspaces and choose a Permission under **Set bulk permission**. This Permission will then be granted to the user for all selected Workspaces.

Teams

Teams are groups of Members who share the same Workspace and product-level Permissions. Teams allow you to efficiently control access, by assigning Permissions to the Team rather than by configuring each Member individually.

You can share Datasets with Teams and individual Members.

To configure a Team:

1. On the sidebar, expand **Organization** and then select **Member & Teams**.
2. Give focus to the **Team** upper tab.
3. Select **Create Team**. This displays a configuration modal.
4. Provide a meaningful **Name**, and optionally, a **Description**.
5. Select the desired Workspace, and the desired access level, to assign to the Team. (If you select the **Members** level, you can then assign granular product-level Permissions.)
6. In **Team Members**, select the Members you want to add, then **Add** them. (To appear in this drop-down, a Member must already be created with your Organization.) The table will update to show each Member

you've added.

7. When finished, **Save** your new Team.

The main **Teams** tab displays the Teams configured on your Workspace. You can **Edit** or delete these, as needed, from the **Actions** column.

7.3. Usage Settings

Manage Notification targets, Usage Groups, and capacity limits.

You can manage Cribl Search settings (as well as Cribl global settings) by selecting **Settings** from the sidebar. When you do so from within Cribl Search, the **Settings** > **Search** upper tab automatically gains focus.

You'll now see left tabs where you can configure the Cribl Search–specific settings listed below. (Adjacent tabs provide access to **Global** settings that govern your whole Organization, and to settings specific to other Cribl products.)

- Notification targets receive notifications from scheduled searches.
- Usage Groups define groups for controlling limits.
- Limits control searches and the UI.



Certain Cribl Search settings can be overridden by `set` statements in your queries.

Set the Limits of Search Results

By default, Cribl Search returns up to **50,000** results per search.

You can change this limit at three levels:

- Organization level (between 1 and 10 million).
- Usage Group level (between 1 and the Organization limit).
- Per-search level (between 1 and the Usage Group limit).

Certain Dataset Provider types also have their own results limits:

- Elasticsearch Results Limit
- OpenSearch Results Limit

Set the Results Limit for the Organization

A Search Admin can set the system-level limit that applies to all searches in the Organization, by changing the **Results limit** parameter of the **system** Usage Group:

1. Go to the **Usage Groups** page in Cribl Search: On the top bar, select **Products** > **Search** > **Settings** > **Usage Groups**.
2. Select the **system** Usage Group.

3. Change the **Results limit** to any value between `1` and `10000000`.
4. Save your changes.

Now, all searches in your Organization will be limited to the number of results you specified.

Set the Results Limit for a Usage Group

A Search Admin can set the results limit for a specific Usage Group. This limit will apply to all searches run by the group's members, and will be capped by the Organization limit.

1. Go to the **Usage Groups** page in Cribl Search: On the top bar, select **Products** > **Search** > **Settings** > **Usage Groups**.
2. Select the Usage Group you want to change.
3. Change the **Results limit** to any value between `1` and `10000000`.
4. Save your changes.

Set the Results Limit for a Search

Apart from using the `limit` operator, any user can set the results limit for a specific search by adding the `max_results_per_search` option in a `set` statement at the beginning of the query.

```
set max_results_per_search=1000;

dataset="cribl_search_sample"
```

Copy | Run in Cribl Search

For a regular user, this limit will be capped by their Usage Group limit, which in turn is capped by the Organization limit. For a Search Admin, the `max_results_per_search` value can override the Organization limit up to the maximum of 10 million.

7.3.1. Usage Groups

Control access to resources for specific Search Members, by creating Usage Groups.

Usage Groups enable you to assign limits to searches by specific users. By setting limits, you can prevent inefficient queries either by disallowing them, or by terminating their execution if they exceed a limit.



To set up general limits that apply to all searches in your Organization, see the limits topic.

Cribl Search provides two Usage Groups by default:

- **system** – Provides system-level limits that apply to all searches.
- **default** – Covers all ad hoc searches, unless covered by a different group.

Creating Usage Groups

To create a Usage Group, select **Settings** > **Search** > **Usage Groups**, then select **Add Usage Group**. This opens a **New Usage Group** modal.

On the modal's **Configuration** tab, give the group a name, and set up desired limits that members of this group will be bound by. Activate the group by making sure the **Enabled** toggle is selected before you save the configuration.

To assign users to a group, on the **Members** tab, select **Add Members** . All users who have the **User** or **Editor** Search Member Permissions are eligible to be group Members.

Usage Limits

You can configure the following usage limits for each group:

- **Earliest relative time range** – How far back in time the search can go. For example: `60m`, `30d`, `6mon`, `1y`. To specify time in seconds, enter a number with no time unit. This limit can be overridden by `max_earliest_relative`.
- **Time range limit** – Maximum time range for one search query, for example 3 days (`3d`). This limit can be overridden by `max_timerange_width`.



When managing each of the following three **concurrent** limits: Indicate a percentage (of CPU capacity on the Leader) by appending `%`. Reduce the default `100%` if you notice an impact on UI responsiveness, or on workloads like Stream Collector jobs. A Leader with only 1 CPU will treat all

> values as `100%`. You also have the option to allocate an absolute number of CPUs on the Leader, by omitting the `%`.

- **Concurrent ad hoc search limit** – Maximum volume of ad hoc searches that all Members of this Group can run simultaneously. Queries exceeding this limit will get queued. This limit can be overridden by `max_searches_per_user`, and by the **Overall concurrent search limit** value in this modal.
- **Concurrent scheduled search limit** – Maximum volume of scheduled searches that all Members of this Group can run simultaneously. Queries exceeding this limit will get queued. This limit can be overridden by `max_scheduled_searches_per_user`, and by the **Overall concurrent search limit** value in this modal.
- **Overall concurrent search limit** – Maximum number of simultaneous searches allowed across your Organization.
    - Percentage values for all **concurrrent** limts are calculated as: `Math.min(1, numCPU x maxConcurrentSearches/100)`.
    - Cribl Search will be entirely disabled if this number is set to `0`.
- **Results limit** – Maximum number of events that can be returned per search.
    - Default is `50000`.
    - Minimum of `1`.
    - Maximum of `10000000`.
    - This limit can be overridden by `max_results_per_search`.
- **Executors limit** – Maximum number of executors dispatched per search.
    - Default is `50`.
    - Minimum of `1`.
    - This limit can be overridden by `max_executors`.
- **Byte limit** – Maximum number of bytes that can be read per search. When the limit is exceeded, the search gets canceled. This limit can be overridden by `max_bytes_read_per_search`.
- **Running time limit** – Maximum time, in seconds, that a single search is allowed to run.
    - Default is `28800` seconds (8 hours).
    - This limit can be overridden by `max_running_time_per_search`.
- **Coordinator heap memory limit** – Memory allocation (in MB) for the search coordinator process.
    - Default is literally labeled `default`, which delegates to the Node.js default.
    - Set a slightly higher value if you experience out-of-memory errors due to high Dataset cardinality or other conditions.
    - Be aware that very high settings might compromise other applications.

Overriding Limits

Certain usage limits can be overridden by `set`-statement options, for example: `set max_searches_per_user=100000`.

Admins can also run a query as a specific Usage Group with: `set usage_group="secops"`.

The following `set` options correspond to the limits configured in the UI described above:

Limit	Setting
Earliest relative time range	`max_earliest_relative`
Time range limit	`max_timerange_width`
User concurrent ad hoc search limit	`max_searches_per_user`
User concurrent scheduled search limit	`max_scheduled_searches_per_user`
Results limit	`max_results_per_search`
Executors limit	`max_executors`
Byte limit	`max_bytes_read_per_search`
Running time limit	`max_running_time_per_search`

7.3.2. Limits

Configure limits that apply to all searches performed in your Cribl Search instance.

You can configure limits that control search performance, search history, and the granularity of search summaries displayed in the UI.

Limits listed in this section apply to all searches performed in your Organization. (You can define specific limits per user by configuring Usage Groups.)

Certain limits can be overridden by `set` statements in your queries.

List of Configurable System Limits

The following limits are available at **Settings** > **Search** > **Limits**:

- **Length of the search job queue**: Sets the number of search queries that can be queued. If the maximum number of queries has already been reached, new queries will be added to a queue and executed later. The limit can be expressed either as a percentage of the maximum number of concurrent queries or as an absolute number, allowing you to control the length of the queue.
    - Default is `500%`.
    - Minimum of `0`.
- **Warm pool size**: Maximum number of warm coordinator search processes to run on the Leader or Worker Node. `auto` sets 1 process on the Leader and 0 on Worker Nodes.
    - Default is `auto`.
    - Minimum of `0`.
- **Search history TTL**: Amount of time to keep search artifacts around before removing them.
    - Default is `7d` (7 days).
    - Minimum of `0`, which instructs Cribl Search to not retain search artifacts at all.
    - You can override this limit for a specific scheduled search, using the **Keep last executions** setting.
    - Notebooks have a hard-coded 30-day retention period to facilitate extended investigations. Exceeding the **Search history job limit** will cause other jobs to be removed before Notebook jobs, to respect this extension.
- **Search history job limit**: Maximum number of search jobs to retain before removing the oldest ones.
    - Default is `1000`.
    - Minimum of `0`.

    - You can override this limit for a specific scheduled search, using the **Keep last executions** setting.
- **Field summary nested depth limit**: Maximum depth supported when building field summaries on nested object values. For example, `x.y.z` corresponds to the default depth of `3`. Increase this default if you need to go deeper. For complex datasets with deeply nested structures, this limit prevents excessive resource consumption during field discovery.
    - Default is `3`.
    - Minimum of `1`.
- **Field summary field limit**: Maximum number of distinct fields to include (before truncation) when generating field summaries on objects. Reduce this limit to improve performance with complex datasets.
    - Default is `200`.
    - Minimum of `0`.
- **Compress cache**: When set to **Yes**, object cache artifacts get compressed.
    - Default is **No**.
- **Prevent revealing Dataset Provider secrets** (Admin only): When enabled, credentials used for configuring Dataset Providers (such as secret keys) are obfuscated.
    - Default is **Yes**.

List of Fixed System Limits

The following limits are not configurable:

- **Field size limit**: Maximum per-field payload size enforced on fields (like `_raw`). When results are rendered, Cribl Search will truncate or omit oversized payloads to maintain responsiveness. This limit applies to all searches.
    - Set to `2` MB per field, and not configurable.

7.4. Sharing Search Resources

Share your Datasets, Dataset Providers, search results, and Dashboards, with other Search Members.

Cribl Search enables Admins to share Datasets (including Cribl Lake Datasets), Dataset Providers, Dashboards, and search results with other Members and Teams of your Organization. There are two layers of Permissions for Members and Teams, and these determine what resources are available to them.

At the product level, a Search Member or Team can have the **User**, **Editor**, or **Admin** Permission. For details, see Search Member Permissions.

At the resource level, Members and Teams can be assigned the **Maintainer**, **Read Only**, or **No Access** Permission on Datasets and Dataset Providers.

Share Datasets and Dataset Providers

Members and Teams with the Search **User** Permission do not, by default, have access to view, edit, or search Datasets and Dataset Providers. (**Editor** and **Admin** Members/Teams do have such access by default – they automatically inherit the **Maintainer** Permissions on resources at this level.)

As an Admin, here is how to share a Dataset or Dataset Provider with a Member or Team whose Search Permission is **User**:

1. Select **Data** from Cribl Search's sidebar.
2. Select the **Datasets** or **Dataset Providers** upper tab.
3. In the **Actions** column of the resource you want to share, select **Share**.

In the resulting **Sharing** drawer, for Members or Teams whose Search Member Permission is **User**, the **Dataset Permission** drop-down enables you to select one of these resource-level Permissions:

- **Maintainer**: Full access to manage and search.
- **Read Only**: Can view and search configurations, but not modify them.
- **No access**: Cannot view, edit, or search.

Revoke Dataset Access

If an Admin un-shares a Dataset from a User or Team, or removes a User from an authorized Team, the User's past interaction with the Dataset determines their remaining access.

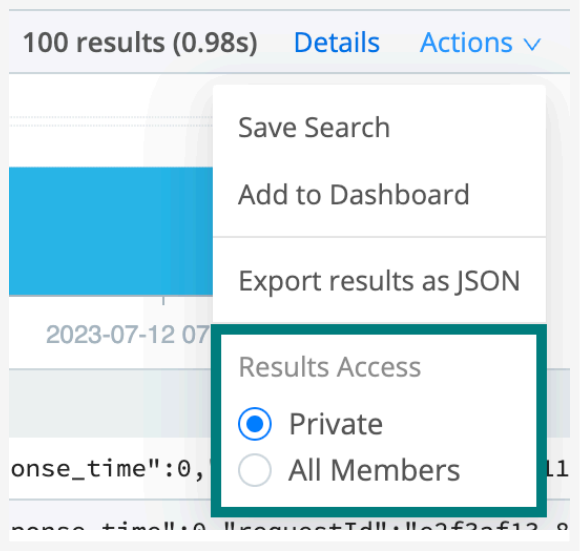
If a User has performed a search against the Dataset, they retain access to its data and search results in the search history.

However, the Dataset itself is no longer visible to the User, and the User cannot re-run searches against that Dataset.

Share Search Results

You can set search results as either **Private** (the default) or **All Members**. **Private** results are visible only to you, and are hidden from other Members and Teams.

After running a search and viewing its results, you can open the **Actions** drop-down to change the search's **Results Access**.



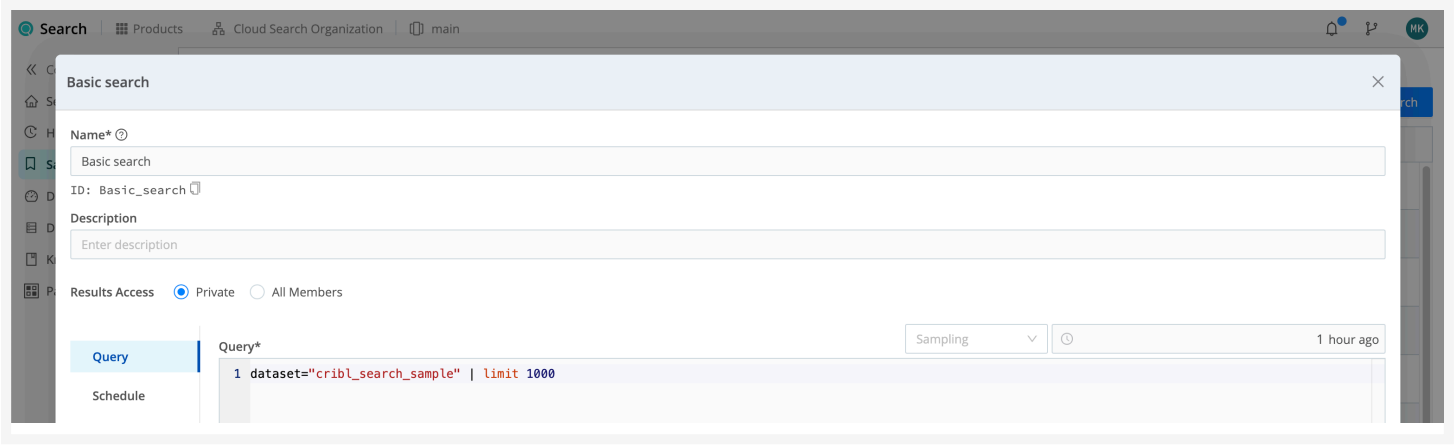
Share Search Results

You can also change the visibility of your searches with a `set` statement, using the `job_visibility` option.

The **Actions** menu provides more options on the **Fields** and **Charts** tab than are displayed in the above screenshot, which is from the **Events** tab.

Share Scheduled Search Results

Open a scheduled search's configuration modal to adjust its **Results Access** setting. Your saved setting applies to any subsequently created result sets from this search.



Managing access to a scheduled search

Share Search Results via URL

You can share displayed search results by copying the URL from your browser's address bar.

This URL will encapsulate the search's details and state – the query string, time-range selection, sample-ratio selection, and any chart and table settings applied in aggregated results.

Another Search Member, with appropriate Permissions, can paste this URL into their browser's address bar to reload the search. This is equivalent to loading a cached search from History, so it consumes no additional credits.

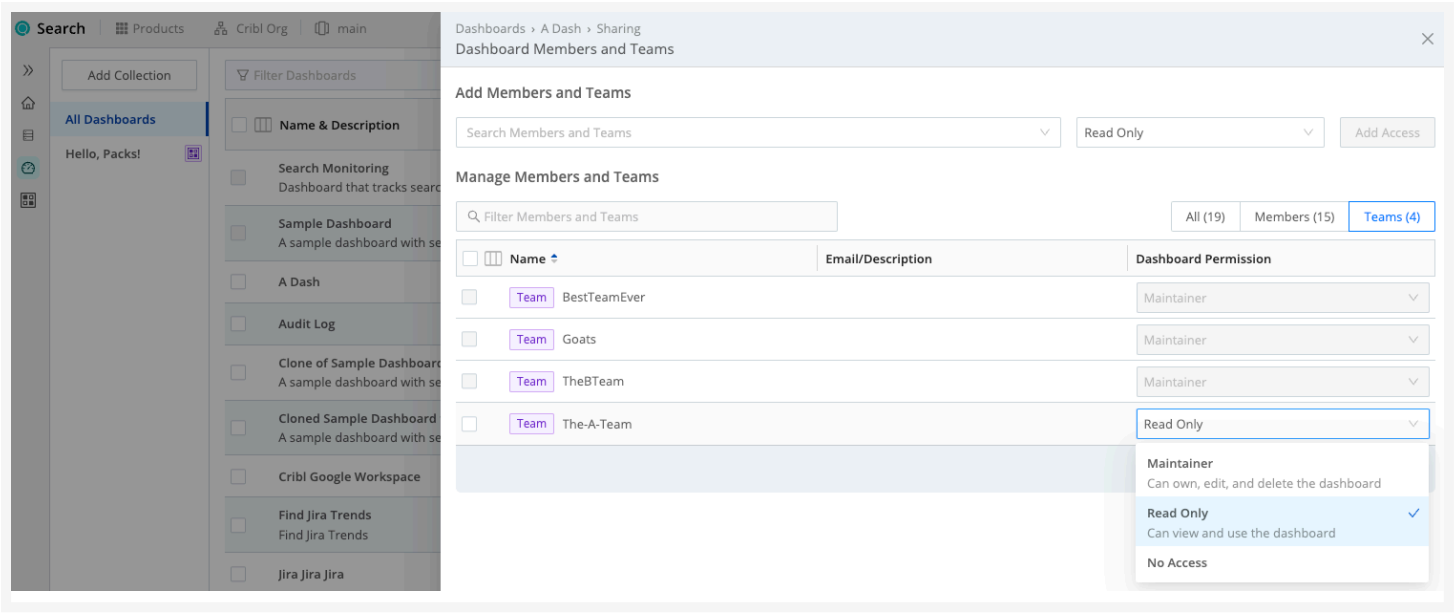
Share Dashboards

You can share any Dashboard on which you have the **Owner** or **Maintainer** Permission with other users.

To do this, select **Dashboards** from Cribl Search's sidebar, and locate your Dashboard. Select this Dashboard's Actions ⋯ button, and select **Share**.

In the resulting drawer, for Members or Teams whose Search Member Permission is **User**, the **Dataset Permission** drop-down enables you to select one of these Permissions on the Dashboard:

- **Maintainer**: Can edit, share, and delete the Dashboard.
- **Read Only**: Can view the Dashboard.
- **No Access**: Cannot view or edit the Dashboard.



Sharing a Dashboard

8. Optimize Cribl Search

Query, partitioning, and workflow strategies to maximize Cribl Search efficiency.

The following topics offer strategies and tips for optimizing your searches. They provide guidelines for designing efficient queries, partitioning paths effectively, and other practices to get the most out of Cribl Search.

- Optimize Searches
- Optimize Paths
- Investigative Searching
- Productivity Tips
- Useful Queries (Analytic Cookbook)

For a quick-reference summary of Cribl Search compatibility, sample operators and expressions for common queries, and optimization suggestions, download our Cribl Search Tip Sheet.

8.1. Optimize Searches

Get strategies and tips for optimizing your searches.

The practices on this page help you make searches more efficient and faster.

Constrain Initial Query with Limit, Count, or Time Window

Until you're certain how large your Dataset is, you should add a `limit`, `count()`, or time limit to your initial query.

Instead of this:

```
dataset="cribl_search_sample"
```

Copy

Do this:

```
dataset="cribl_search_sample" | limit 1000
```

Copy

Or this:

```
dataset="cribl_search_sample" dataSource="access_combined"
 | limit 1000
 | summarize count() by host, clientip
```

Copy

Or this, to constrain results to a 2-month window:

```
dataset="cribl_search_sample" earliest=-2mon
```

Copy

Unstructured Searches Are Fastest

Unstructured term searches will return faster than structured searches. Consider using a broad search (with a limit) in initial investigations.

Instead of this:

```
dataset="my_dataset" email="jim@company.com" | limit 1000
```

Copy

Do this:

```
dataset="my_dataset" "jim@company.com" | limit 1000
```

Copy

Place Filters Early in Queries

Try to push filtering expressions to the left-most side of your query wherever possible.

Instead of this:

```
dataset="<my-datasource>"
| where tenantId == "foo-bar-12345"
| where proc == "bash"
| where data_source == "stdout"
```

Copy

Do this:

```
dataset="<my-datasource>" tenantId="foo-bar-12345" proc="bash" data_source="stdout"
```

Copy

Place Intensive Operations Last

To minimize costs, try to move functions like `lookup` and `sort` to the right-most part of a query. First, filter or aggregate your Dataset to reduce the data volume sent to these functions.

Instead of this:

```
dataset="<my-datasource>" dataSource="VPC Flow Logs" | lookup service_names on dst_pc
```

Copy

Do this:

```
dataset="<my-datasource>" dataSource="VPC Flow Logs" | summarize count() by dst_port
```

Copy

Summarize to Search Faster

Any search that returns a large number of raw events back to the UI will be slower than a summary result set.

Instead of this:

```
dataset="my_data"
```

Copy

Do this:

```
dataset="my_data" | summarize count() by cid
```

Copy

Summarize Before Join

When using joins with aggregation operators, it's more efficient to summarize-then-join rather than join-then-summarize.

Instead of this:

```
let URLMethods=dataset="access_common_data";

dataset="my-datasource" | join URLMethods on URL | summarize count() by method
```

Copy

Do this:

```
let URLMethods=dataset="access_common_data" | summarize count() by URL, method;

dataset="my-datasource" | summarize count() by URL, method | join URLMethods on URL
```

Copy

Search Faster with Comma Syntax

Many operators can perform multiple functions simultaneously if you link your query together using commas instead of pipes.

Instead of this:

```
... | extend field1="foo" | extend field2="bar" | extend field3="pike"
```

Copy

Do this:

```
... | extend field1="foo", field2="bar", field3="pike"
```

Copy

Move Partition Tokens to Queries

Token-based partitions in your Dataset can drastically increase your search time if the directory paths are very broad. If you instead specify the tokens as part of your search, this will reduce the search span to only those sub-trees. Ideally, keep time to the left-most portion of your path, and keep tokens to the right wherever possible, as shown here.

If this is your Dataset definition:

```
data/${dataSource}/${_time:%Y}/${_time:%m}/${_time:%d}/${_time:%H}
```

Copy

Then instead of this:

```
dataset="myDataset" | summarize count() by destination | where dataSource="cisco"
```

Copy

Do this:

```
dataset="myDataset" dataSource="cisco" | summarize count() by destination
```

Copy

Send Exclusively to Cribl Stream to Speed Up Large Result Sets

When search results expand beyond several thousand events, sending the results to Cribl Stream via the `send` operator is faster than returning events to the Cribl Search UI.

Instead of this:

```
dataset="my_data" | send tee=true
```

Copy

Do this:

```
dataset="my_data" | send
```

Copy

Lakehouse: Use `where` for Case-Sensitive Searches

When performing a case-sensitive search against a Lakehouse, using the case-sensitive `where ==` syntax can provide faster performance than using the implicit `cribl` operator.

Instead of this (implicitly invoking `cribl`, which is case-insensitive):

```
dataset=my_dataset field1=foo field2=bar
```

Copy

Do this:

```
dataset=my_dataset | where field1=="foo" and field2=="bar"
```

Copy

With the `where` operator, you must place quotes around string literals that you use in comparisons, as shown just above. With the `cribl` operator, the quotes are optional.

Optimize Parquet with `project`

When searching Parquet files, use `project` in the second query clause, to narrow the subsequent expressions to only your fields of interest.

Instead of this:

```
dataset="a-parquet-datasource" | summarize sum(bytes) by customer, account
```

Copy

Do this:

```
dataset="a-parquet-datasource" | project bytes, customer, account | summarize sum(byt
```

Copy

8.2. Optimize Paths

Partition paths for efficient searchability.

Best Practices for Optimizing Paths in Object Stores

Use these guidelines when you define bucket paths and partitioning for Cribl Search Datasets on Amazon S3, Azure Blob Storage, and Google Cloud Storage to ensure efficient search performance and cost management.

Prioritize Time

Make time the left-most partition. Keep time tokens at the start of the bucket path. For example:

```
my-bucket/${_time:%Y}/${_time:%m}/${_time:%d}/${_time:%H}/....
```

Copy

By putting time first, you ensure the static search prefix is maximized, which lets Cribl Search quickly skip irrelevant data, resulting in faster search performance.

Set Time Granularity

Aim for a balance where each time partition directory contains a manageable number of objects (typically in the hundreds or low thousands). For high-volume Datasets, partition down to `hourly` or `per-minute` level to allow Cribl Search to skip entire folders for irrelevant hours.

Place Non-Time Attributes After Time

If you need to partition by attributes like `region`, `environment`, `account`, or `sourcetype`, add those segments under the time hierarchy. For example:

```
my-bucket/${_time:%Y}/${_time:%m}/${_time:%d}/${region}/${sourcetype}/....
```

Copy

Use search filters (such as, `where region == "us-east-1"`) instead of placing these attributes before time. This structure also simplifies applying different S3 retention policies for various data types.

Limit Wildcards

Use wildcards only when necessary (for example, to skip `key=value` segments). Place them only on the right side of the path and make sure they are applied to segments that do not impact search filtering. For example:

```
my-bucket/${_time:%Y}/${_time:%m}/${*}/${someVarOfInterest}
```

Copy

Static Values: Use Separate Datasets

If a path segment has a small, known list of values (for example, limited `dataImportance` or `environment` values), define multiple datasets or multiple bucket paths with static values rather than using a wildcard across that segment.

Use Sourcetype/Datatype as a Key Partition

When you control ingestion (for example, using Cribl Stream/Edge), consistently encode `sourcetype` or similar `datatype` signals in the path so related events land together and parsing is predictable.

Prefer Cribl Lake

Land data in Cribl Lake and search Lake Datasets instead of manually managing complex partitioning schemes in raw object stores. Cribl Lake automatically applies an optimized Dataset partitioning scheme that aids in `newest-first` discovery while maintaining intuitive time organization.

Common Pitfalls When Setting Up Paths

Avoid these common path design pitfalls when configuring your object store datasets.

Avoid Early Wildcards

Any wildcard near the left of the path forces Cribl Search to list and examine every key under that prefix before it can apply time filtering, dramatically increasing search time and cost.

Avoid patterns like:

```
my-bucket/${*}/${_time:%Y}/${_time:%m}/
```

or

```
my-bucket/data/${category:-*}/${_time:%Y}/.....
```

Avoid Large Directories

Avoid single directories containing tens or hundreds of thousands of objects. For high-volume data, partition down to at least the hourly level.

Avoid Over-Partitioning

Per-second or overly fine per-minute directories can cause excessive list operations and metadata overhead. If the majority of per-minute directories are empty or contain very few objects, coarsen your time partitioning (such as, from per-minute to hourly).

Avoid Placing Routing Fields Before Time

Placing routing fields (like `region` or `account`) before time prevents Search from leveraging time as the longest static prefix. This forces the object store to scan multiple, separate prefixes (one for every `region/account` queried) instead of using a single, efficient time-based lookup.

Avoid patterns like:

```
my-bucket/${region}/${account}/${_time:%Y}/${_time:%m}/...
```

Don’t Hide Key Attributes

If segments such as `dataImportance`, `environment`, or `sourcetype` are key to your search criteria, do not place them in segments you intend to wildcard. Placing them behind a wildcard (such as, `my-bucket/${*}/${_time:%Y}/...`) negates their value for path pruning and increases search time and cost.

Newest-First Ordering

The order in which Cribl Search traverses objects in public storage depends directly on the alphanumeric sorting (lexicographic order) of the object keys returned by the object store provider.

Default Behavior (Oldest-First)

- When you use a standard time hierarchy in your path (such as, `YYYY/MM/DD/...`), the object store API lists keys in ascending, oldest-first order.
- Although Cribl Search reads events within each file in reverse-chronological order, the overall traversal of objects is constrained by this oldest-first listing order.
- While this approach is efficient for smaller directory listings (hundreds of objects), for larger listings (thousands or more), Cribl Search must still respect this order to maintain predictable and efficient behavior.

Ordering in Cribl Lake

- Cribl Lake Datasets automatically apply an optimized “magic prefix” to the bucket path.
- This prefix encodes time in a way that forces the lexicographic listing order to correspond to reverse-chronological (newest-first) order.
- This optimization allows Cribl Search to discover and scan the newest objects first, significantly improving the speed of typical time-range queries.

Approximating Newest-First in Raw Object Stores

While Cribl Lake automatically handles optimal time partitioning for “newest-first” retrieval, you can achieve similar results when writing data directly to S3, Azure, or GCS using Cribl Stream or Edge.

Use `C.Time.s3TimePartition()`

Use the `C.Time.s3TimePartition()` function in Cribl Stream to generate your time prefix. This function encodes time in a special format that causes the resulting object keys to be ordered reverse-chronologically.

Incorporate this function into the bucket path expression within your Cribl Stream or Edge Destination:

```
my-bucket/${C.Time.s3TimePartition(_time, 'h')}/${sourcetype}/${host}/
```

Copy

The function takes a time field (`_time`) and a granularity specifier:

- `C.Time.s3TimePartition(_time, 'h')` yields an hourly prefix.
- `C.Time.s3TimePartition(_time, 'd')` yields a daily prefix.



This technique applies only to data written through Cribl Stream or Edge. Native producer formats (like CloudTrail or Splunk SmartStore) retain their own partitioning schemes. You must still configure Search Datasets with paths that align with your chosen prefix strategy.

Example: Manage Longest Static Prefix in S3

The Amazon S3 API uses your bucket path’s longest static prefix for fetching data. So when using two partitions, they must be supplied in every search in order for your time fields to be considered in the prefilter fetch stage.

For example: If you specify `field1=foo` and `field2=bar`, at noon on November 1, 2024, the longest prefix for fetching data would be:

```
<mybucket>/inputID/foo/bar/2024/11/01/12/00/
```

Copy

This is exactly what you want, and very efficient for the files only in that S3 partition. But more efficiently, you'd want:

```
<mybucket>/2024/11/01/12/00/inputID/foo/bar
<mybucket>/2025/11/01/12/00/inputID/foo/bar
```

Copy

And ideally, you'd want them reverse-chronological, and lexicographically associated:

```
<mybucket>/baca-2025/ac-11/aa-01/ad-12/00/inputID/foo/bar
<mybucket>/bada-2024/ac-11/aa-01/ad-12/00/inputID/foo/bar
```

Copy

On the other hand, if you didn't specify `field2`, but used the same criteria, the longest prefix would then be:

```
<mybucket>/inputID/foo/
```

Copy

And here, Cribl Search would have to pull every file within that partition boundary across the S3 API.

Another approach is to reverse the order of your criteria, like this:

```
<mybucket>/inputId/year/month/day/hour/minute/FIELD_1/FIELD_2/
```

Copy

This eliminates the churn caused by omitting either of those criteria.

Note that you'd still get the correct data in your result set if you didn't specify `field2`, because Cribl Search will still filter each event. However, the pool of objects searched could potentially be much larger. This would increase your Search wall-clock time and your proportional retrieval cost.



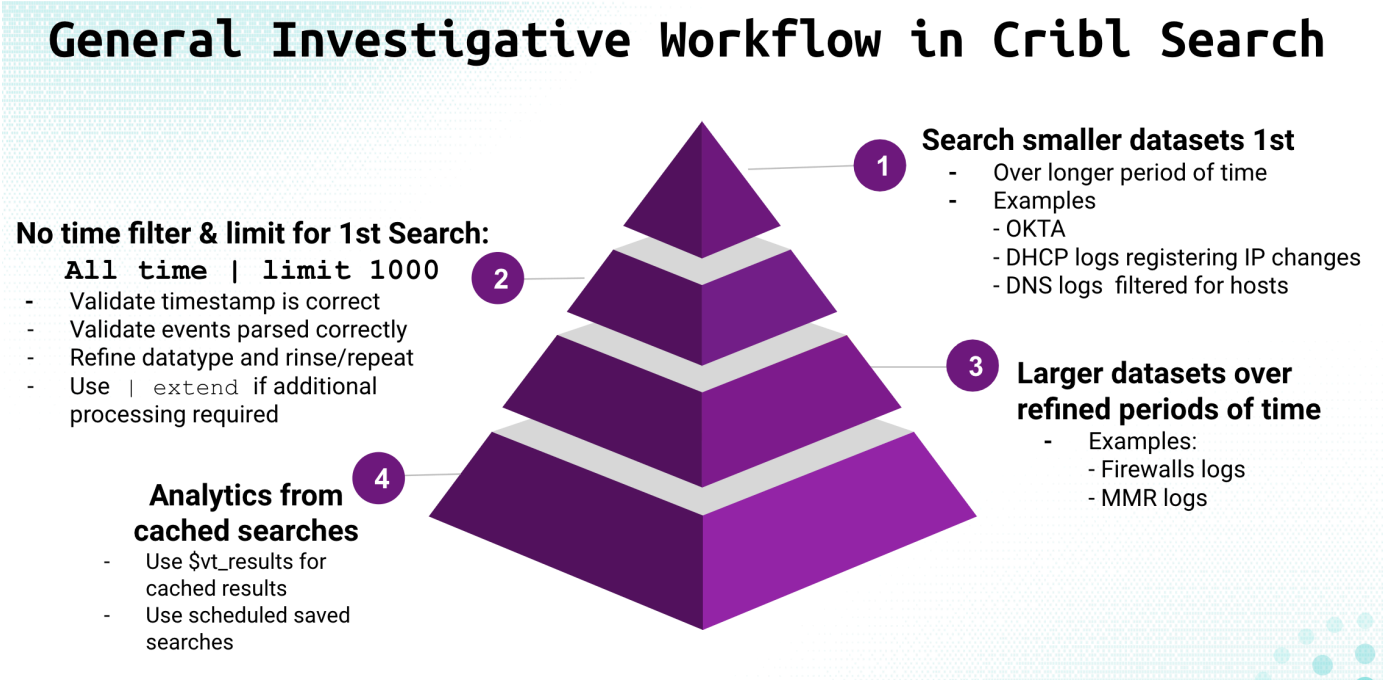
In Cribl Stream, the Amazon Security Lake Destination automatically partitions buckets this way. It creates time partitions on top, and you can explicitly add further partitioning fields, which this Destination adds below the time boundaries.

8.3. Investigative Searching

Optimize investigations with Cribl Search.

When performing investigations across multiple Datasets, an ideal workflow is:

- Search the smallest Dataset first, across all time, with a limit of 1,000 events.
- Alternatively, if your smallest Dataset has been searched before, search that Dataset first, for your time window of interest.



Ideal investigative workflow

S3: Discover Unknown Bucket Directory Structure

You can investigate this in the AWS Management Console. Or, in Cribl Search:

1. Create a Dataset whose path is just the S3 bucket name.
2. Run this query:

```
.show objects ("<dataset_name>") | where name contains "<xxx>"
```

Copy

Specify Time Formats in Paths

If directories are partitioned as year, month, day, hour, second, use:

```
${_time:%<parameter>}
```

Copy

For example:

```
. ${_time:%Y-%m-%d}
```

Copy

Or:

```
<mybucket>/${_time:%Y}/${_time:%m}/${_time:%d}/...
```

Copy

Or, if directories have `earliest` and `latest` in seconds, use those tokens:

```
<mybucket>/${__earliest/${__latest}/...
```

Copy

If the path has multiple time ranges:

- Use the most accurate field(s).
- Name the other field conspicuously (examples. `time_from_dir` or `other_time`).

Time Format Examples

If time in the directory format is:

```
<mybucket>/2024-128-19/xxx.gz
```

Copy

The path expression can be:

```
<mybucket>/prefix/${_time:%Y-%m-%d}
```

Copy

If time in directory format is:

```
<mybucket>t/2024/08/19/
```

Copy

The path expression can be:

```
<mybucket>/prefix/${_time:%Y}/${_time:%m}/${_time:%d}
```

Copy

For the following example:

```
aws/12-17-2024/db_0507829470_1165276110_8475_4288F9E6-9A3A-43B4-CCE9-4247DB82FCF9.csv
```

Copy

The path expression can be:

```
aws/${other_time}/db_${latest}_${earliest}_$(bucketid}_${suffix}.${extension}
```

Copy

Specific Versus Parameterized Dataset Paths

With specific file paths, you avoid `repository=*` problems:

```
<bucket>/sophos/${_time:%m}-${_time:%d}-${_time:%Y}/
<bucket>/okta/${_time:%m}-${_time:%d}-${_time:%Y}/
```

Copy

But to maximize search performance, parameterize your path structure, while specifying as many of the available parameterized fields as possible. Consider this structure:

```
<bucket>/${repository}/${_time:%m}-${_time:%d}-${_time:%Y}/
```

Copy

A simple query term like this would retrieve files from all repositories:

```
dataset=cribl_lake_logs
```

Copy

Whereas a specification like this – over the same timeframe – would filter for a much smaller subset of files. This would produce a faster and more efficient search:

```
dataset=cribl_lake_logs repository=okta
```

Copy

8.4. Productivity Tips

Efficiently view your data and format search queries.

These tips help you make the most of Cribl Search.

Preview Search Modifications

Use the Operator Preview feature to test adding new operators to a previously run query, Preview works with a subset of your retrieved data, so you can quickly model the changes without incurring the time or cost of rerunning a search until you're ready.

1. Add an operator to your search (for example, `| extend` or `| extract`).
2. Hover over the operator keyword and select the **Preview** button.
3. Update your search in the resulting **Preview** modal, then select its own **Preview** button to preview the results on up to 100 rows of of your data.
4. When you have validated your new query, select **Apply**. This sends the modified search back to your query box.

Quickly Format Queries

To make your search expression more readable in the query box, press the `Ctrl + |` (Linux, Windows) or `Command + |` (MacOS) key combination. This wraps the query at each pipe symbol.

Instead of this:

```
dataset="<my-datasource>" dataSource="VPC Flow Logs" | summarize count() by dst_port
```

Copy

See this:

```
dataset="<my-datasource>" dataSource="VPC Flow Logs"
| summarize count() by dst_port
| lookup service_names on dst_port
```

Copy

Simplify Queries with Reusable Macros

Cribl Search Macros enable you to store snippets of strings – and optionally, variables – that you can reuse in multiple queries.

Instead of this:

```
dataset="aws_ftc_audit_accesslogs"
| extend _raw=replace_regex(_raw, @'""', @'"')
| extract type=json
```

Copy

Insert this:

```
${ftc_audit_accesslogs}
```

Copy

Instead of this:

```
dataset="aws_ftc_audit_accesslogs" _raw!="\"_time\",source,host,sourcetype,\"_raw\",\
| extend _raw=replace_regex(_raw, @'""', @'"')
| extract type=json
```

Copy

Insert this:

```
${ftc_audit_accesslogs} ${exclude_header}
```

Copy

Manage Lookups

To list lookup files' names and contents, use this simple query:

```
dataset="cribl_lookups"
```

Copy

Filter Data with `.show`

To see file names in a Dataset:

```
.show objects("cribl_search_sample")
```

Copy | Run in Cribl Search

To see file names that do not end with `.gz`

```
.show objects("cribl_search_sample")
| where name !endswith('.gz')
```

Copy | Run in Cribl Search

8.5. Useful Queries (Analytic Cookbook)

Kusto Query Language (KQL) recipes to cook up productive searches.

This page presents KQL analytic search queries that are useful for various specific purposes. Entries describe how the queries work, and show sample results.

Search DDSS Bucket Time Ranges

You can use this query to return DDSS buckets' time ranges, formatted and sorted:

```
.show objects(<bucket name>)
| extend  path3=split(name,"/",2) | extend dbpath=split(path3,"_") | extend starttime
| extend starttime=format_datetime(starttime_epoch, 'yy-MM-dd hh:mm:ss tt'), endtime=
| project name, size, starttime, endtime
| sort by starttime asc
```

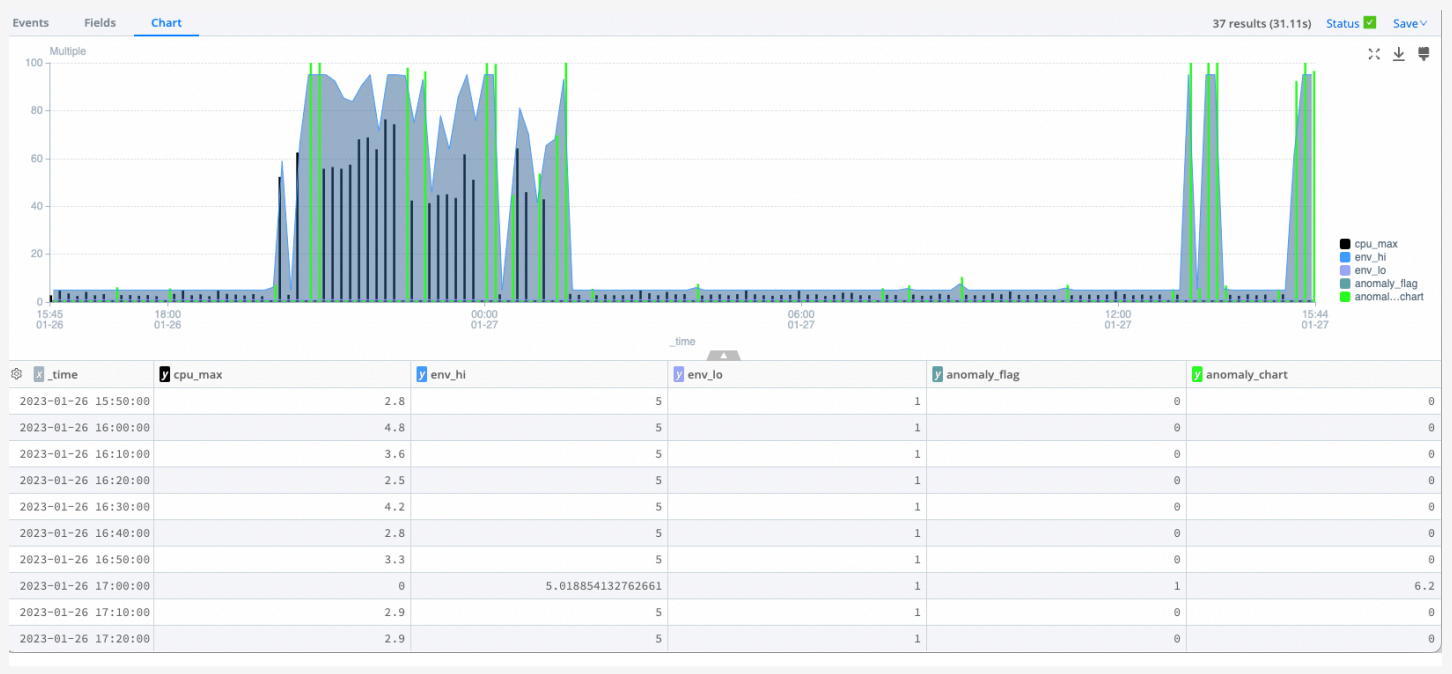
Copy

Z-Score–Based Anomaly Detection

This analytic takes a time series Dataset, applies a mean and standard deviation to it, then computes a high-and-low envelope based on `mean +/ n*stdev`, where N is called the "Z-score." This is what is commonly referred to as "N standard deviations above/below the average."

```
dataset="cribl_edge_metrics" cribl_fleet="default_fleet" host="ip-172-31-91-42.ec2.ir
| timestats span=10m cpu_max=max(node_cpu_percent_active_all), cpu_avg=avg(node_cpu_p
| extend z_score=4, delta=cpu_stdev*z_score // create a variable z-score, and a delta
| extend env_hi=cpu_avg+delta, env_lo=cpu_avg-delta // create an envelope above and b
| extend env_hi=min_of(env_hi,95), env_hi=max_of(env_hi,5) // fudge factor for the hi
| extend env_lo=max_of(env_lo,1)  // fudge factor for the low limit to be at least 1%
| extend anomaly_flag=iif(cpu_max>env_hi , 1,0), anomaly_chart = cpu_max * anomaly_fl
| project-away z_score, delta, cpu_stdev, cpu_avg // remove the intermediate computat
```

Copy



_time	cpu_max	env_hi	env_lo	anomaly_flag	anomaly_chart
2023-01-26 15:50:00	2.8	5	1	0	0
2023-01-26 16:00:00	4.8	5	1	0	0
2023-01-26 16:10:00	3.6	5	1	0	0
2023-01-26 16:20:00	2.5	5	1	0	0
2023-01-26 16:30:00	4.2	5	1	0	0
2023-01-26 16:40:00	2.8	5	1	0	0
2023-01-26 16:50:00	3.3	5	1	0	0
2023-01-26 17:00:00	0	5.018854132762661	1	1	6.2
2023-01-26 17:10:00	2.9	5	1	0	0
2023-01-26 17:20:00	2.9	5	1	0	0

Z-score results

Notes on Z-Score Detection

The various extend operators compute the z-score envelope (`env_hi` and `env_lo`) and then compute an `anomaly_flag` if the actual Dataset (here, `cpu_max`) is greater than the high envelope.

We use `anomaly_chart` to visually display the anomalies (points higher than `env_hi`) in a separate color, highlighting them. (This is a bit of a kludge, since we can't currently repaint individual data points in a series).

AWS Inventory by Datatype, Objects, Events, TotalBytes

This analytic looks in your S3-based Dataset and counts the number of objects (files, usually), the number of events in those files, and the number of bytes in those events. You can run this over a large sample of data – say, several days – to get a feel for how much data is in your S3 bucket.

The Unsampled Version



This version is expensive, in terms of time and CPU*seconds. Compare it to the sampled version below.

```
dataset="cribl_search_sample"
| extend eventBytes=strlen(_raw)
| summarize NumEvents=count(), NumObjects=dcount(source), totalBytes=sum(eventBytes)
```

Copy

#	datatype	NumEvents	NumObjects	totalBytes
1	vpcflowlogs	1724032	587	191470980
2	syslog	1723998	582	205655801
3	access_common	1724024	582	162145184
	0	5172054	1751	559271965

233.40s	11557.688	539.54MB	5175676	50
Time elapsed	CPU*s	Scanned	Events Returned	Executors

Unsampled results

Note: This search took almost 4 minutes of wall clock time and over 11,000 CPU*Seconds to run, equivalent to more than **three** Cribl credits.

The Sampled Version

The sampled version, run at 1:1000 sampling. (Use this one!) Byte and Event counts are corrected for the sampling error.

```
dataset="cribl_search_sample"
| extend eventBytes=strlen(_raw)
| summarize NumEvents=count(), NumObjects=dcount(source), totalBytes=sum(eventBytes)
| extend NumEvents=NumEvents*1000, totalBytes=totalBytes*1000
```

Copy

9.60s	221.507	556.42MB	8928	50
Time elapsed	CPU*s	Scanned	Events Returned	Executors

#	datatype	NumEvents	NumObjects	totalBytes
1	vpcflowlogs	1721000	547	191191000
2	syslog	1756000	544	209625000
3	access_common	1704000	541	159600000
	0	5181000	1632	560416000

Sampled results

Notes on AWS Optimization

This search took less than 10 seconds of wall clock time to run, and about 220 CPU*seconds, equivalent to only 0.06 Cribl credits.

When run over a large time range (say, 1 day), the sampled version is almost as accurate as the unsampled version (within two or three significant figures). It runs about 25x faster, and costs about 50x less in terms of CPU*seconds.

AWS Cost and Usage Report (CUR) – Parquet Analysis

This analytic is tricky only because the CUR is in Parquet format, and doesn't have `_time` in each event. Instead, there's a field called `identity_time_interval`, which inexplicably has **two** timestamps, with a slash separating them. By using a regex to extract them, we can then use one (here, the `intervalstarttime`) to cast back into `_time`, and then events will behave as desired.

```
dataset="billing"
| extract source=identity_time_interval type=regex regex=@"(?<intervalstarttime>.+)\/
| extend intervalstarttime=todatetime(intervalstarttime), intervalendtime=todatetime(
| extend _time=intervalstarttime
| timestats span=1h totalusage=sum(line_item_usage_amount) by product_group
```

Copy

This particular analytic simply summarizes `line_item_usage_amount` by `product_group`, but once the data has been properly formatted, any further analytics are fairly straightforward.

Internal Search Utilization

You can use this search to obtain a more granular view of your own Cribl Search usage.

```
dataset="cribl_internal_logs" message="search finished"
| summarize  sum(stats.cpuMetrics.totalCPUSeconds) by stats.user
```

Copy

Note that this example is split by `user`, but you can change it to `split by` anything, or eliminate the `split by` entirely.

Correct for dollars:

```
dataset="cribl_internal_logs" message="search finished"
| extend totalCost=stats.cpuMetrics.totalCPUSeconds/3600
| timestats span=1d  sum(totalCost) by stats.userDisplayName
```

Copy

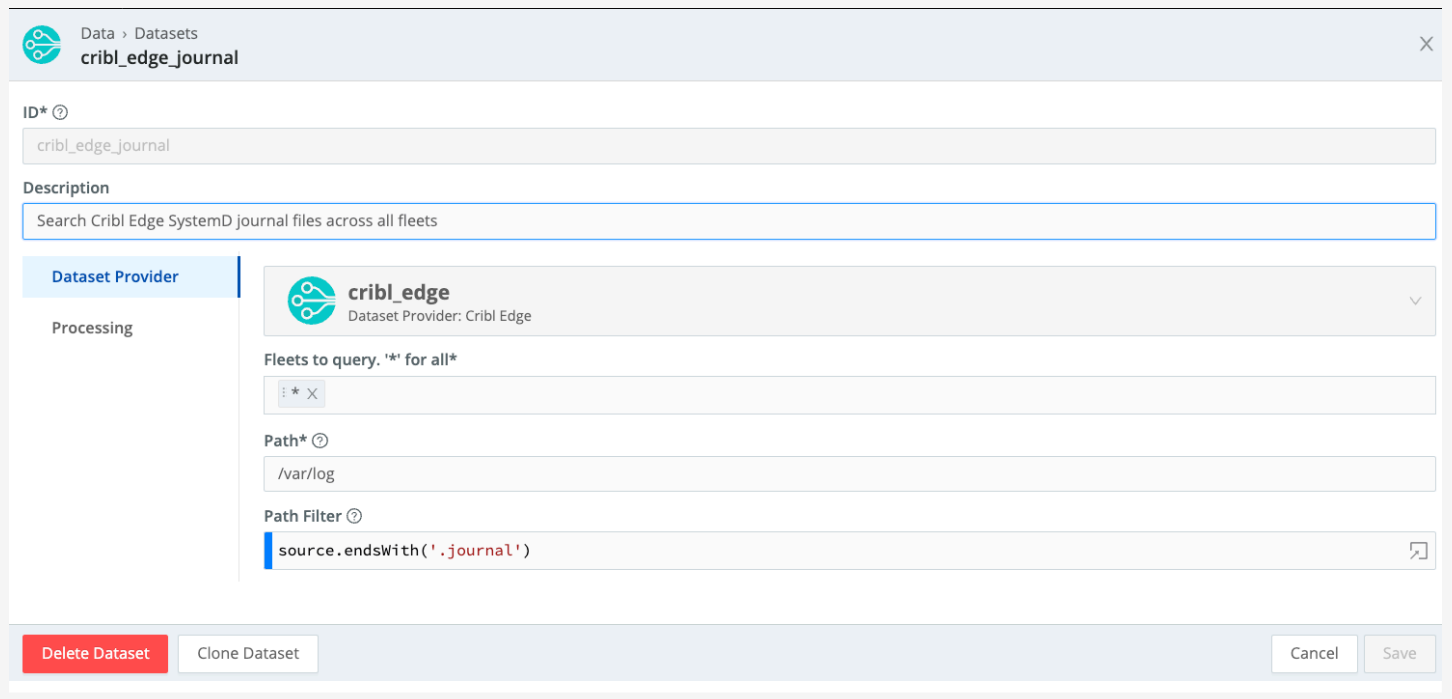
JournalD Search via Edge

This search will connect to your Edge fleet and search JournalD files on all hosts in the fleet for Command Line arguments.

```
dataset="cribl_edge_journal" journald._CMDLINE=*
| limit 100
| project _time,host,journald._CMDLINE,journald._PID,journald._UID,journald._GIDf
```

Copy

This requires that you've created a Dataset called `cribl_edge_journal`, which is of type **Edge**, and filtered to look at journal files in `/var/log`. Here's an example of how to set up the Dataset:



Edge Dataset configuration

You can query specific Fleets by tailoring the **Fleets to query** field.

Splunk VPC Flow Logs Search via S3

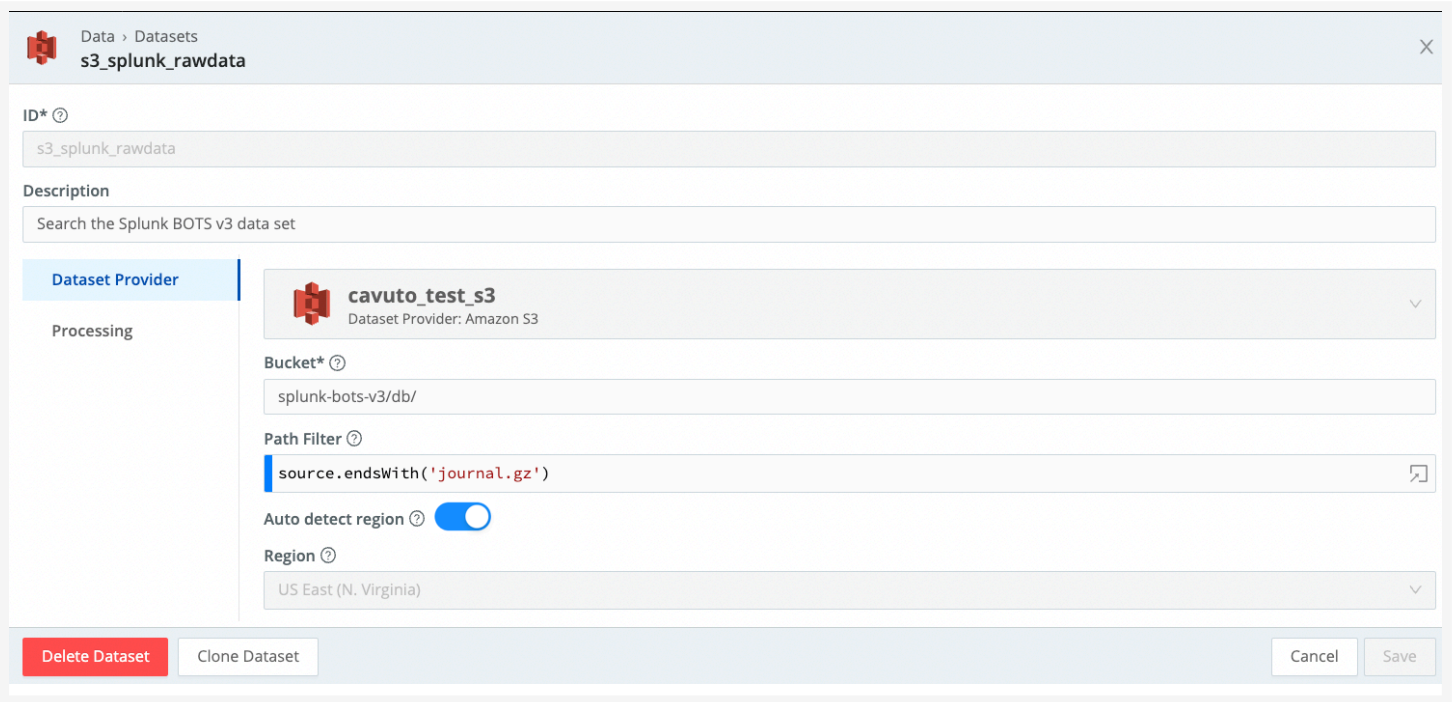
This earches a set of Splunk rawdata (`journal.gz`) files in a bucket in S3, then extracts the results using an existing parser.

```
dataset="s3_splunk_rawdata" sourcetype=*vpc*
| limit 1000
| extract parser='AWS VPC Flow Logs'
```

Copy

This requires a Dataset pointing to a set of Splunk files. It will descend into subfolders – need not be flat.

All Splunk compression formats are supported, including `.zip`, `.gz`, and `.lzw`.



Splunk Dataset configuration

Search a Data Lake Populated from Cribl Stream

Any specific search will work on this Dataset.

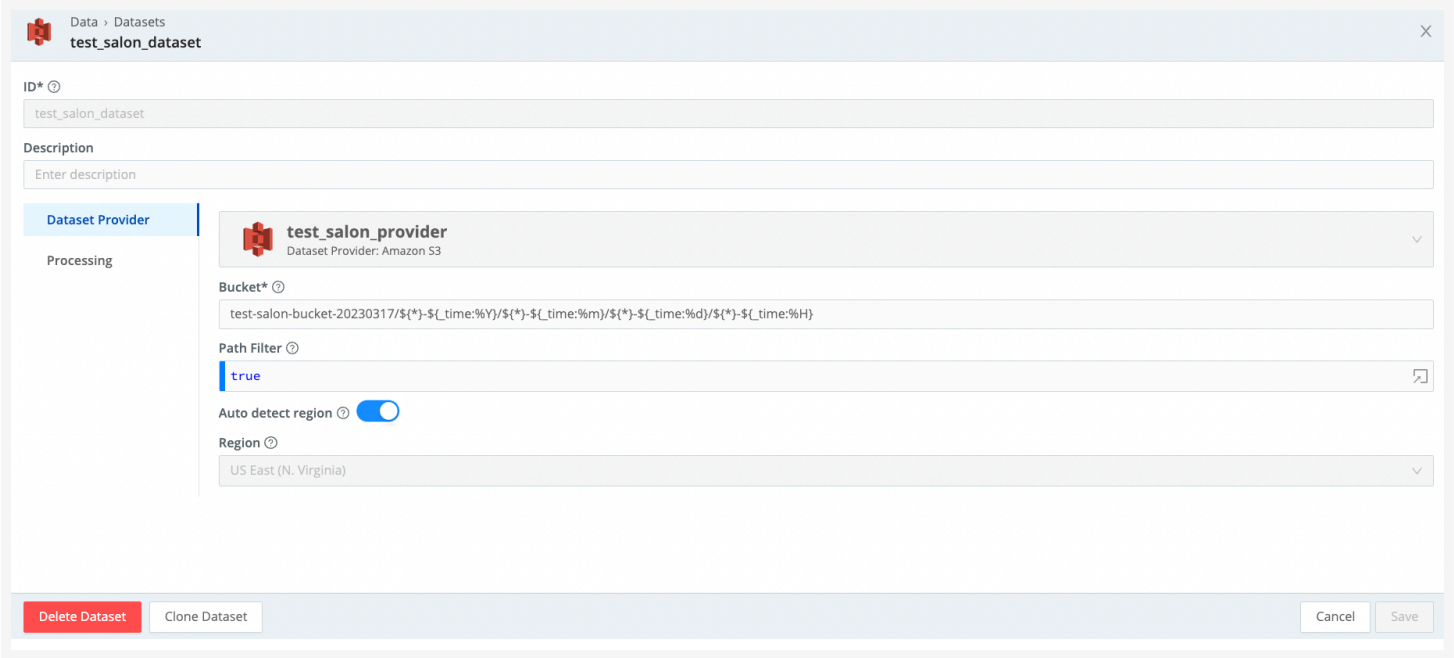
```
dataset="data_lake" | limit 1000
```

Copy

Partitioning scheme needs to set to:

```
${}-${_time:%Y}/${}-${_time:%m}/${}-${_time:%d}/${}-${_time:%H}
```

Copy



Lake Dataset configuration

Send Cribl Search Results to Cribl Stream

Enables you to take arbitrary Cribl Search results and send them to Cribl Stream with the `send` operator.

```
dataset="s3_splunk_rawdata" sourcetype=*vpc*
| extract parser='AWS VPC Flow Logs'
| where srcaddr in~ ("34.215.24.225", "172.16.0.178") and dstaddr in~ ("172.16.0.178"
| send tee=false group=default
```

Copy

This is just an example of how to drill down into a Dataset (here, using the Splunk BOTS Dataset), and then send the results to Stream.

Internal-to-External Network Conversations

This analytic works with BOTS v3 Data:

```
dataset="bots_v3" sourcetype in ("stream:tcp", "stream:udp")
| limit 1000
| extract type=json
| where isnotnull(src_ip) and isnotnull(dest_ip)
| lookup matchMode='cidr' cidr_1918_blocks on src_ip=cidr_block
| extend src_network_type=network_type
| lookup matchMode='cidr' cidr_1918_blocks on dest_ip=cidr_block
| extend dest_network_type=network_type
| where src_network_type=="private" and dest_network_type=="public"
| summarize values(dest_port), session_count=count() by src_ip, dest_ip
```

Copy

This analytic works with VPC Flow Logs:

```
dataset="cribl_search_sample" dataSource=*vpc* | limit 1000
| where isnotnull(srcaddr) and isnotnull(dstaddr)
| lookup matchMode='cidr' cidr_1918_blocks on srcaddr=cidr_block
| extend src_network_type=network_type
| lookup matchMode='cidr' cidr_1918_blocks on dstaddr=cidr_block
| extend dest_network_type=network_type
| where src_network_type=="private" and dest_network_type=="public"
| summarize values(dstport), session_count=count() by srcaddr, dstaddr
```

Copy

Requires creating this lookup table:

```
cidr_block,network_type
10.0.0.0/8,private
127.0.0.0/8,loopback
172.16.0.0/12,private
192.168.0.0/16,private
0.0.0.0/0,public
```

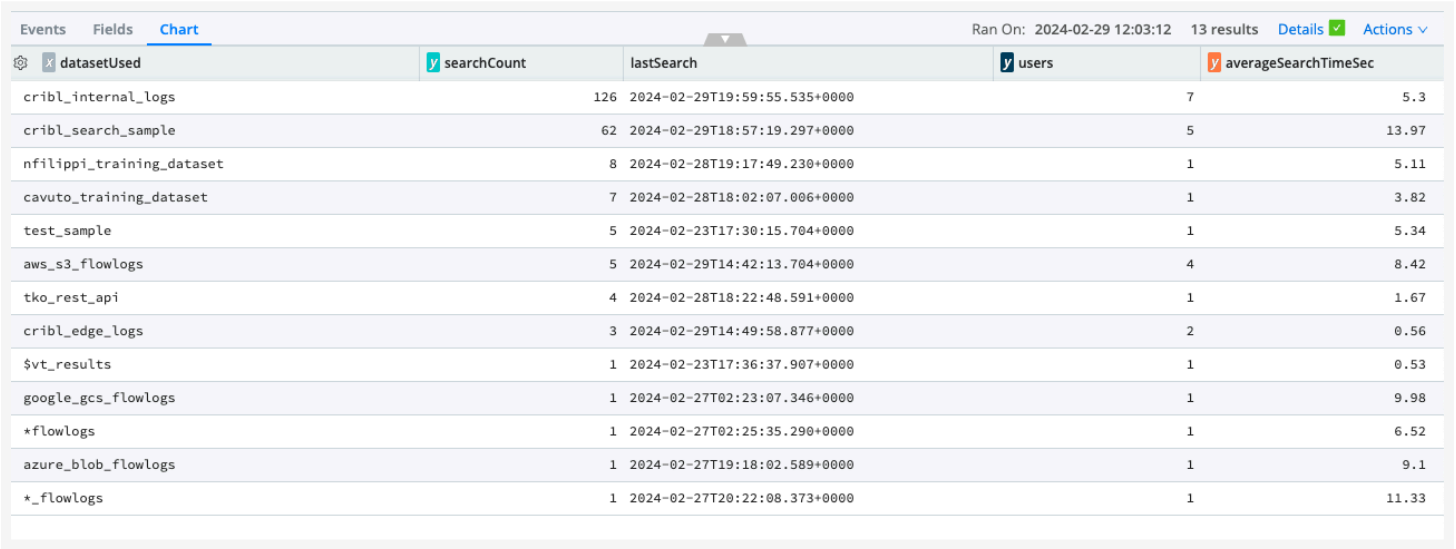
Copy

Dataset Usage Stats

Analyze Cribl Search logs to understand the Datasets with the highest number of searches, number of users who have searched, when the last search was, and what the average search time is. Admins can use this analytic to help discover which Datasets would offer the highest return if optimized.

```
dataset="cribl_internal_logs" source=="file:///opt/cribl/log/searches.log" stats.stat
 | limit 1000
 | extract source=stats.query @'dataset=\"(?<datasetUsed>\S+)\"'
 | where datasetUsed
 | summarize searchCount = dcount(stats.id), lastSearch = max(_time), users = dcount(
 | extend lastSearch = strftime(lastSearch, '%Y-%m-%dT%H:%M:%S.%L%Z')
 | sort by searchCount desc
```

Copy



Events Fields Chart — Ran On: 2024-02-29 12:03:12 — 13 results — Details — Actions

datasetUsed	searchCount	lastSearch	users	averageSearchTimeSec
cribl_internal_logs	126	2024-02-29T19:59:55.535+0000	7	5.3
cribl_search_sample	62	2024-02-29T18:57:19.297+0000	5	13.97
nfilippi_training_dataset	8	2024-02-28T19:17:49.230+0000	1	5.11
cavuto_training_dataset	7	2024-02-28T18:02:07.006+0000	1	3.82
test_sample	5	2024-02-23T17:30:15.704+0000	1	5.34
aws_s3_flowlogs	5	2024-02-29T14:42:13.704+0000	4	8.42
tko_rest_api	4	2024-02-28T18:22:48.591+0000	1	1.67
cribl_edge_logs	3	2024-02-29T14:49:58.877+0000	2	0.56
$vt_results	1	2024-02-23T17:36:37.907+0000	1	0.53
google_gcs_flowlogs	1	2024-02-27T02:23:07.346+0000	1	9.98
*flowlogs	1	2024-02-27T02:25:35.290+0000	1	6.52
azure_blob_flowlogs	1	2024-02-27T19:18:02.589+0000	1	9.1
*_flowlogs	1	2024-02-27T20:22:08.373+0000	1	11.33

Dataset usage analysis

Timerange Distribution

What timerange do users most often use when running ad hoc (standard) searches?

```
dataset="searches" stats.status in ("completed","failed","canceled") stats.type="star
 | extract source=stats.searchTimeRange type=regex @'(?<timeRangeValue>\d*)'
 | extract source=stats.searchTimeRange type=regex @'\d*(?<timeRangeModifier>\D*)'
 | extend timeRangeMultiplier = case (timeRangeModifier == "ms", 0.000000277777778,
                                      timeRangeModifier == "s", 0.000277777778,
                                      timeRangeModifier == "m", 0.01666666666,
                                      timeRangeModifier == "h", 1,
                                      timeRangeModifier == "d", 24,
                                      timeRangeModifier == "M", 730,
                                      timeRangeModifier == "y", 8760,
                                      "none")
 | where timeRangeMultiplier != "none"
 | extend timeRangeHours = round(timeRangeValue * timeRangeMultiplier,2)
 | summarize within_1h=sum(iff(timeRangeHours <= 1,1,0)),
             within_1d=sum(iff(timeRangeHours <= 24,1,0)),
             within_7d=sum(iff(timeRangeHours <= 168,1,0)),
             within_14d=sum(iff(timeRangeHours <= 336,1,0)),
             within_30d=sum(iff(timeRangeHours <= 730,1,0)),
             within_1y=sum(iff(timeRangeHours <= 8760,1,0)),
             total=count()
| project within_1h = strcat(within_1h, ' (', round(within_1h/total*100,2), '%)'),
          within_1d = strcat(within_1d, ' (', round(within_1d/total*100,2), '%)'),
          within_7d = strcat(within_7d, ' (', round(within_7d/total*100,2), '%)'),
          within_14d = strcat(within_14d, ' (', round(within_14d/total*100,2), '%)'),
          within_30d = strcat(within_30d, ' (', round(within_30d/total*100,2), '%)'),
          within_1y = strcat(within_1y, ' (', round(within_1y/total*100,2), '%)'),
          all = strcat(total, ' (', round(within_total/total*100,2), '%)')
```

Copy

9. Troubleshooting Cribl Search

Diagnose and solve Cribl Search issues.

Known Issues

Find out if a bug is already fixed, or discover temporary workarounds. See: Known Issues.

KQL Extensions

Identify where Cribl KQL operators, functions, or types differ from their Kusto counterparts. See: KQL Extensions.

Lakehouse Search Differences

Identify where operators, functions, or types differ when searching a Lakehouse versus other Cribl Lake Datasets. See: Lakehouse Search Differences.

Search Details

Identify problems with your query by examining the search's execution logic and logs. See: Search Details.

Share Diagnostics

Share a diagnostic bundle with Cribl Support, to help us solve your issue. See: Share Diagnostics.

Common Issues and Resolutions

Diagnose and resolve common errors in working with Cribl Search. See: Common Issues and Resolutions.

9.1. Known Issues

View filterable Known Issues for Cribl Search. Check if issues are resolved or find temporary workarounds.

9.2. KQL Extensions

Understand Cribl KQL extensions and variations from the Kusto language.

Cribl Search KQL is built on top of the Microsoft Kusto Query Language, with additional extensions and operators.

This page lists known areas where Cribl KQL operators, functions, and types differ from their similarly named Kusto counterparts.

These differences might require changes when you programmatically manage Cribl Search using API requests (or other automation) that was originally written around those Kusto counterparts.

- The `dayofweek` function returns an integer between `0` and `6`, representing the day of the week, beginning on Sunday. This differs from the standard Kusto `dayofweek` function, which returns a timespan.
- The `make_timespan` function converts the specified time period into a number of seconds. This output format differs from the standard Kusto `make_timespan` function, which returns a timespan.
- The `timespan` data type represents a time interval, in seconds. This representation differs from the standard Kusto `timespan` type, which represents a literal timespan.
- The `totimespan` function converts the input expression into a time interval, in seconds. This output format differs from the standard Kusto `totimespan` function, which returns a timespan.
- The `tostring` function, when applied to a `null` input value, returns `null`. This output format differs from the standard Kusto `tostring` function, which returns an empty string for `null` values.

9.3. Lakehouse Search Differences

Identify Cribl Search behavior that differs when searching Lakehouses versus standard Cribl Lake Datasets.

When you run Cribl Search queries against a Lakehouse-assigned Dataset, some behavior and results differ from a corresponding search executed against a Cribl Lake Dataset without Lakehouse caching. This page identifies operators, functions, data types, and other details to keep in mind when searching Lakehouses.

You can disable using Lakehouse for a specific query by using the `set` statement with the `lakehouse` option set to `off`.

Unlimited Events with `sort`

The `sort` operator's `MaxNoOfOutputEvents` parameter defaults to unlimited with Lakehouse, rather than to `10000` events as in a regular search. However, you can set this limit parameter as desired.

Strict Boolean Comparisons

Boolean comparisons in Lakehouse searches are strongly typed. Assume a query condition like `field1==field2`, in which one field is a boolean, while the other is a boolean-compatible value (such as `"yes"`, `"y"`, `"t"`, `"true"`, or `"1"` for `true`; or `"no"`, `"f"`, `"false"`, or `"0"` for `false`).

In a mixed comparison – for example, where `field1`'s value is `true` and `field2`'s value is `"yes"` – note that `field1==field2` will not match. (When searching the same data without Lakehouse caching, looser typing allows these values to match.)

Statistical Aggregations with Booleans

A statistical aggregation that aggregates mathematically on a boolean value will return a `null` in Lakehouse searches. (Running the same aggregation without Lakehouse caching will return a numeric value – either `1` for `true` or `0` for `false`.)

This `null` return value applies to the following functions when their `Expression` argument takes a boolean value: `avg`, `percentile`, `stdev`, `stdevp`, `sum`, `variance`, and `variancep`. (A boolean value in these functions' `Predicate` argument does not cause a `null` return value.)

The `null` return value also occurs with a boolean value in the following functions' sole `Expression` argument: `avgif`, `stdevif`, `sumif`, and `varianceif`.

`dcount` Excludes `null` Values

When applying the `dcount` aggregation function, Lakehouse searches do not count `null` values toward the total, whereas searches without Lakehouse caching do.

This means that any `null` values in your data will cause the `dcount` value to be off by 1, compared to running the same query against the same data without Lakehouse caching (or in a different data store).

Extra `null` Column with `timestats`

Applying the `timestats` aggregation function to a search against a Lakehouse-assigned Dataset might return a `null` column when only empty values are found. Searches without Lakehouse caching do not return this column.

Higher `dedup` and `eventstats` Result Counts

Applying the `dedup` or `eventstats` operator to a search against a Lakehouse-assigned Dataset can return a higher result count, compared to searching the same data without Lakehouse caching. This is because `dedup` and `eventstats` in Lakehouse mode can process unlimited events, compared to the 50,000-event limit applied to `dedup` and `eventstats` in searches without caching.

Different `extract type=regex overwrite` Behavior

When using the `extract` operator with `type=regex` and the `overwrite` option for existing event fields, the behavior is different with and without Lakehouse caching. Without Lakehouse, if `overwrite` is set to `false`, the extraction will convert any existing field values to an array. However, Lakehouse searches will omit any existing field values, regardless of whether `overwrite` is set to `true` or `false`.

For example, if you extract `key src_ip` with value `10.2.3.3`, and the field already exists with value `10.1.2.2`, a non-Lakehouse search with `ow=false` will yield a field value of `["10.1.2.2", "10.2.3.3"]`. But a Lakehouse search will yield a field value of `10.2.3.3`.

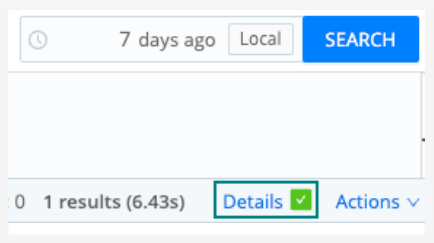
To consistently preserve the field's original value, you can assign that value to a different variable earlier in the query.

9.4. Search Details

Identify problems with your query by looking up the details of the search.

Details Modal

To troubleshoot issues, you can view the search's details, pipeline plan, logs, metrics, and more. At the bottom right of the query box, select **Details**.



Search details

The resulting **Details** modal (shown below) displays the following:

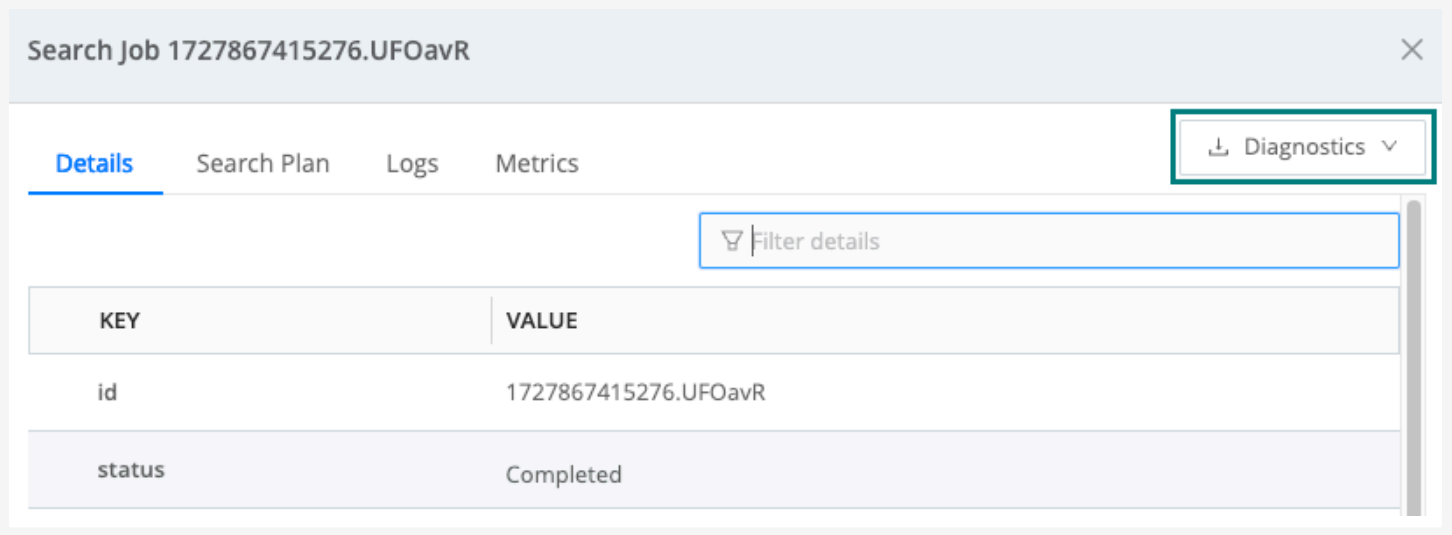
- **id**: Unique identifier of the search. You can use it to, for example, reuse search results. To quickly copy the ID, hover over the **Search Job** panel at the top, and select the **Copy to clipboard** icon.
- **status**: Search status - `new`, `running`, `completed`, `failed`, or `canceled`.
- **user**: User that ran the search.
- **query**: Search query string. If the query contains Macros, you can see what they resolved to, by selecting Expand Macros.
- **earliest**: Beginning of query time range, in a relative time format or milliseconds.
- **latest**: End of query time range, in a relative time format or milliseconds.
- **timeCreated**: Time when the search was created.
- **timeStarted**: Time when the search started running.
- **timeCompleted**: Time when the search was completed.
- **timeElapsed**: Total time the search ran.
- **timeInQueue**: How much time the search spent waiting in the queue.
- **sampleRate**: Ratio to reduce results, see sampling.
- **set options**: Any `set`-statement options affecting the search, like `maxResultsPerSearch` or `allow_previous_results`.

You can also select the following:

- **Search the Results**, to quickly query the result set of the search.
- **Rerun**, to run a new search that uses exactly the same query text and settings.
- **View Results**, to display the cached results of the search.

Diagnostics

The **Diagnostics** drop-down provides options to download a compressed folder with many backend logs, configuration files, and optionally, your search results.



Diagnostics drop-down

Select **Exclude Results** to remove the search results from the downloaded diagnostics folder.

If you're sending the diagnostics folder to someone, like Cribl Support, we recommend selecting this option to remove the results from the folder. This will keep the file size manageable.

For more about sending diagnostics to Cribl Support, see: Share Diagnostics.

Search Plan

The **Search plan** tab shows the backend processes your search ran. Your query was converted to a set of pipelines that work on the data. Pipelines are broken into the following categories:

- **Federated**: Pipelines executed by the remote end.
- **Coordinated**: Pipelines executed by the coordinator process.
- **Combined**: Combined **Federated** and **Coordinated** view.

Typically, the first function you'll see in the **Federated** pipeline is **Drop**. This is a filtering function, which drops data that does **not** match its **Filter** expression.

If your search consists of multiple stages, you can view the pipelines for each stage separately. To see a specific stage, select its ID on the left (for example, **root**).

Export a Search Plan

You can export the plan of a search as a JSON file:

1. Run a search.
2. Once the search starts running, select **Details**, then **Search Plan**.
3. If your search consists of multiple stages, select the ID of the stage whose plan you want to export (for example, **root**).
4. At the bottom left, select **View as JSON** to see the search plan in JSON format.
5. At the top right, select **Export**. The search plan is downloaded as a JSON file.

Logs

Cribl Search creates log events of your search. You'll see informational and debugging level entries with details on every process run for your search.

Logs are separated into either **Coordinated** or **Executors** types:

- **Coordinated**: Organize query execution and do post-processing, for example, merging, sorting, aggregation, and persisting the search results.
- **Executors**: Scan data, for example, reading from S3, decompressing, filtering, and projection.

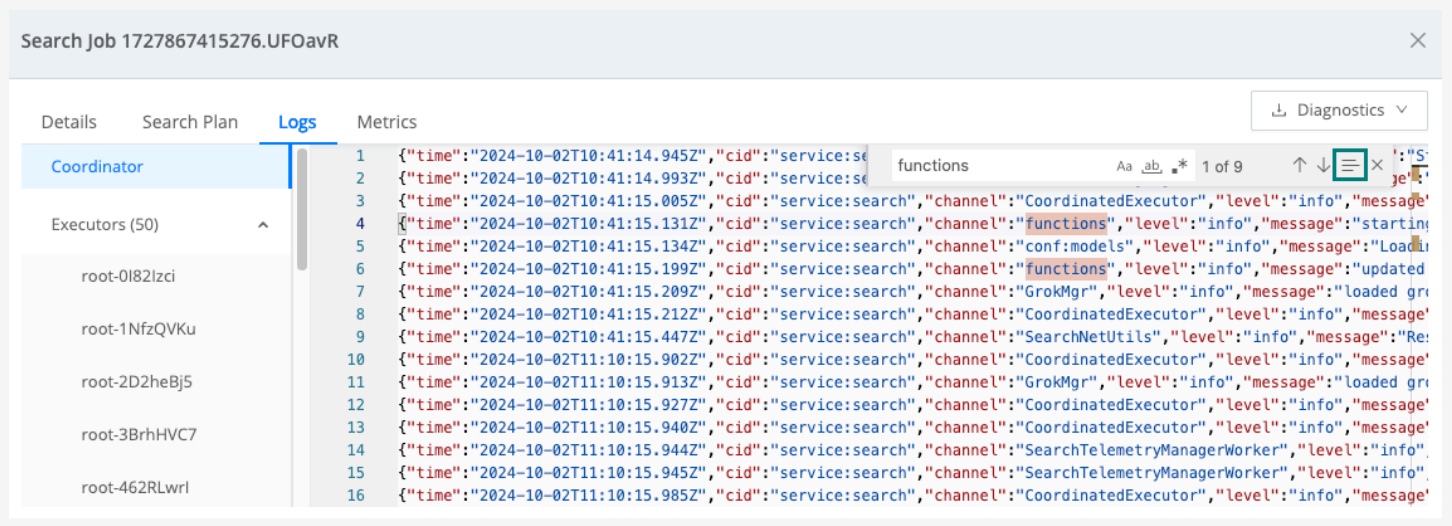
Find Logs

Select inside of the pane with the logs and then press Control+F (Windows) or Command+F (Mac) to open the **Find** bar.

You can search by plain text and have three advanced search options:

- Match Case
- Match Whole Word
- Regular Expression

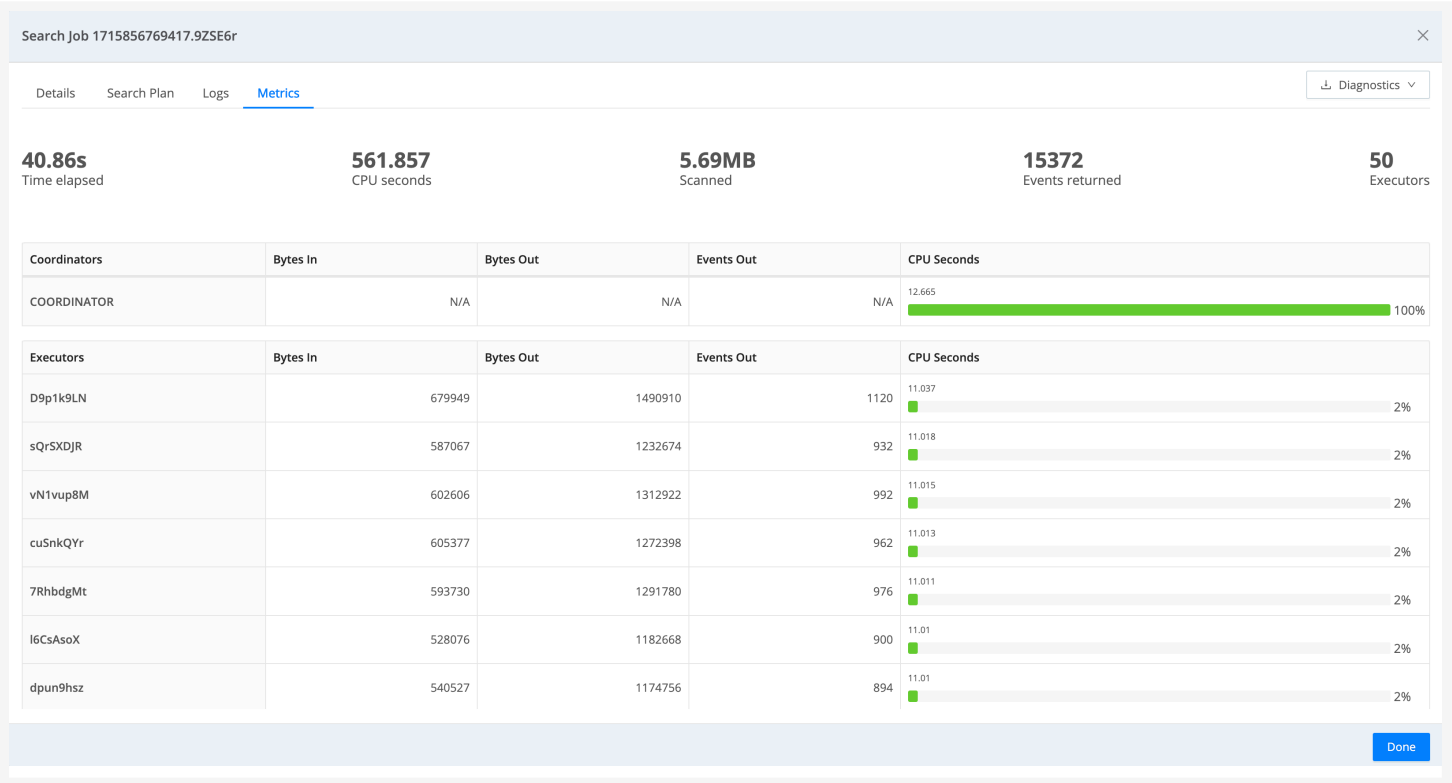
By default, searches run against all logs. To search against only specific text, highlight the desired text and click the three horizontal lines icon.



Find in certain logs

Metrics

Metrics from your search are provided to give you insight into the search's performance and the amount of data it touched.



Search metrics

The top area provides a high-level summary of the stats for the search:

- **Time elapsed**: How much time the search took.
- **CPU*s**: Sum total of CPU seconds spent on the search.

- **Scanned**: Total volume of data ingested by executors and coordinators, calculated after decompression.
- **Events returned**: Number of events the search returned.
- **Executors**: Number of workers the search was split across.

Coordinators

The **Coordinators** table displays a summary of the work done by the coordinator locally. If no Datasets were processed by the coordinator, **Bytes In**, **Bytes Out**, and **Events Out** stats will be **N/A**.

Executors

The **Executors** table displays metrics of the work done by the federated executors, reported on a per-executor basis.

9.5. Share Diagnostics

Share a diagnostic bundle with Cribl Support, to help us solve your issue.

A diagnostic bundle contains a snapshot of configuration files and logs at the time the bundle was created, and gives troubleshooters insights into how Cribl Search was configured and operating at that time.

Generate a Diag File

1. From **Search Home**, load the problematic search into the query box.
2. Select **Details** to open this search in a **Details** modal.
3. From the **Diagnostics** drop-down, select either **Include Results** or **Exclude Results**. This will download the corresponding search details and logs as a .tgz file whose name is a generously long hash.



For details about whether to include or exclude results, see Diagnostics.

Submit Diags

You can submit diagnostics directly via the Support Portal after logging in. If access to this portal conflicts with your organization's policies, please work directly with your Support engineer to find a solution.

9.6. Common Issues And Resolutions

Diagnose and resolve common errors in working with Cribl Search.

This page lists errors and issues that you might commonly encounter when working with Cribl Search, and provides recommendations for diagnosing and resolving the problems.

General Troubleshooting Tree

If you are getting unexpected results from a search against an object store or data lake, start by following this general troubleshooting path:

1. Are you retrieving objects? (Check this using the `.show objects` command.)
2. If you've retrieved no objects, check bucket path(s), Search Permissions to access the Dataset and Dataset Provider, and access permissions on the object store.
3. If you are retrieving objects, search for raw events with no time filtering.
4. If you find no events, check the data format and permissions.
5. If you find events, check your original time filtering and time parsing – is time recognized accurately? Also check Datatypes that contain Event Breakers.

Amazon Connectivity Issues

This section covers issues specific to Amazon Datasets.

S3 Returns No Elements

You might find no elements returned in response to your query.

Possible Cause

Check Search Permissionss to access the Dataset and Dataset Provider. Also check Datatypes and Event Breakers.

Recommendations

Check contents (ListBuckets in S3) to see, and adjust, the Cribl user's or role's permissions on the bucket.

KMS Authorization Fails on S3

If you are using a custom KMS (key management system), you might find that Cribl Search cannot authorize on S3.

Possible Cause

Cribl Search cannot access your keys.

Recommendations

Include `AssumeRole` permissions in the KMS console and in the trust policy, to allow Cribl Search to access the keys.

KMS Decrypt Errors on S3

Retrieving results from objects in an S3 bucket fail, and internal logs show errors of the form: `User is not authorized to perform kms:Decrypt on resource.`

Possible Cause

Your IAM role is missing permissions required to access S3 objects that are encrypted with a KMS key.

Recommendations

Add this sub-element to the applicable IAM role:

```
"Action": [
    "kms:Decrypt",
    "kms:Describe*",
    "kms:Get*",
    "kms:List*",
    "kms:GenerateDataKey"
],
```

Copy

Existing AssumeRole Fails to Authorize on S3

You might find that your existing `AssumeRole` from Cribl Stream/Edge fails to authorize Cribl Search on S3.

Possible Cause

Cribl Search requires a different `Principal` element.

Recommendations

Add the required `Principal` to your trust policy. (Follow the link to see how to copy this `Principal` out of your Dataset Provider modal.)

Can’t Add Buckets to Amazon Security Lake Dataset

You might find that you cannot add new buckets to your Dataset.

Possible Cause

Insufficient permissions on Amazon Security Lake.

Recommendations

Check whether you have added the `securitylake:ListDataLakes` permission to your `AssumeRole`.

Performance Fixes

This section covers workarounds for slow, inefficient, or failing searches.

Scheduled Search Jobs Canceled Due to Queue Length Limit

You might find that some of your scheduled search jobs were canceled. because of the default queue length limit.

Possible Cause

At **Settings** > **Search**, the limit on **Length of the search job queue** defaults to `500%`. This value represents 5x the maximum number of concurrent queries, which you can think of as 5x CPU capacity.

Recommendations

To check the state of the queue, enter `.show queue stats` in the query box.

Try increasing the **Length of the search job queue** to `1500%` (15x maximum concurrent queries), or to a high absolute value like `1000`.

Feature Workarounds

This section covers substitutes for features that Cribl Search doesn’t provide out-of-the-box.

Get the Length of an Array Field

Cribl Search currently has no native function to return the length of an array’s field, without stringifying the value.

Recommendations

1. Strip everything from the array except for commas; apply the `strlen()` function; and then add 1 to the result.
2. Try a query of the form: `print <arrayName> = dynamic([1,2,3]) | extend len = <arrayName>.length`

Email Notifications Issues

The section details the troubleshooting steps you can take if an email Notification fails to reach its intended recipient.

Test the Email Notification Target

To prevent email Notifications from failing in production, you can check the target's configuration as follows:

1. Open the target (in **Settings** > **Search** > **Notification Targets**).
2. Select **Test Target**.
3. Add one or more email addresses to the **Test Target** modal. (Include at least one address where you can check incoming messages.)
4. Select **Send Test Email**.
5. If the test succeeds, check the specified inbox(es) to verify that the test message actually arrived. If not, re-examine the target's configuration.



The preconfigured `system_email` target does not enable displaying this **Test Target** option or its parent modal.

Check Notification Logs

A failed email Notification leaves a log entry. You can search Cribl internal logs for SMTP errors using a query of this form:

```
dataset="cribl_internal_logs" level=error channel="output:*" ioName=smtp
```

Copy | Run in Cribl Search

Within any returned log events, unwrap the `reason` field to find details about the root cause of the outbound email failure.

10. Language Reference

A comprehensive reference for the Cribl Search implementation of KQL.

Cribl Search is based on Kusto Query Language (KQL), which lets you delve into your data to discover patterns, identify anomalies and outliers, and create statistical models.

While the Cribl implementation of KQL mostly follows the original, there are certain KQL differences (for example, we provide the implicit `cribl` operator). To get the details right, follow the language usage guidelines presented in these sections:

- Operators
- Functions
- Statements
- Commands
- Virtual Tables



You can also browse and link to all KQL operators and functions in our Index of Operators and Functions.

Operators

An operator in Cribl Search is a query component that processes data, performing actions such as filtering, counting, or transforming events. Operators can use functions, and are delimited by the pipe character `|`.

For example, the `limit` operator sets the maximum number of events returned from a search:

```
dataset="cribl_search_sample"
 | limit 100
```

Copy | Run in Cribl Search

To learn more, see: Operators.

Functions

A function in Cribl Search is a unit of logic that processes data based on arguments passed to it.

For example, the `max` function returns the maximum value of a field:

```
dataset="cribl_search_sample"
 | summarize LatestEvent=max(start)
```

Copy | Run in Cribl Search

To learn more, see: Functions.

Statements

A statement in Cribl Search is a special keyword that sets options (`set`) or assigns names to expressions (`let`).

For example, this statement sets the maximum number of events returned by the current search:

```
set max_results_per_search=1000;
```

Copy | Run in Cribl Search

To learn more, see: Statements.

Commands

A command in Cribl Search is an instruction used to manage searches directly from the query box. Unlike functions and operators, commands start with a period.

For example, the `.cancel` command can stop searches:

```
.cancel running queries
```

Copy | Run in Cribl Search

To learn more, see: Commands.

Virtual Tables

A virtual table in Cribl Search is a dynamically generated dataset that provides system information useful for troubleshooting, performance analysis, and testing.

For example, the `$vt_dummy` virtual table generates sample data:

```
dataset="$vt_dummy" event<10
```

Copy

To learn more, see: Virtual Tables.

10.1. Language Reference Index

An alphabetical index of all Cribl Search KQL elements.

This page provides a global, alphabetical index of all elements of the Cribl Search KQL language implementation: operators, functions, commands, statements, virtual tables, and supported types.

Mathematical symbols for equality and inequality operators precede letters. Each building block is linked to its reference page and category.

Operator, Function, Element	Category
== (equals, case-sensitive)	Operator
=~ (equals, case-insensitive)	Operator
!= (not equals, case-sensitive)	Operator
!~ (not equals, case-insensitive)	Operator
abs	Mathematical function
acos	Mathematical function
ago	DateTime function
asin	Mathematical function
atan	Mathematical function
atan2	Mathematical function
avg	Statistical function
avgif	Statistical function
bag_has_key	Dynamic function
bag_keys	Dynamic function
bag_merge	Dynamic function
bag_pack	Dynamic function
bag_pack_columns	Dynamic function

Operator, Function, Element	Category
bag_remove_keys	Dynamic function
bag_set_key	Dynamic function
bag_zip	Dynamic function
base64_decode_toarray	String function
base64_decode_tostring	String function
base64_encode_fromarray	String function
base64_encode_tostring	String function
beta_cdf	Mathematical function
beta_inv	Mathematical function
beta_pdf	Mathematical function
bin	Conversion function
bin_auto	Conversion function
binary_and	Binary function
binary_not	Binary function
binary_or	Binary function
binary_shift_left	Binary function
binary_shift_right	Binary function
binary_xor	Binary function
bool	Type
.cancel	Command
case	Conditional function
ceil	Mathematical function
ceiling	Mathematical function
.clear	Command

Operator, Function, Element	Category
centralize	Operator
coalesce	Conditional function
contains	Operator
contains_cs	Operator
!contains	Operator
!contains_cs	Operator
cos	Mathematical function
cot	Mathematical function
count	Operator
count	Statistical function
countif	Statistical function
countof	String function
cribl	Operator
datetime	Type
datetime_add	DateTime function
datetime_diff	DateTime function
datetime_part	DateTime function
dayofmonth	DateTime function
dayofweek	DateTime function
dayofyear	DateTime function
dcount	Statistical function
dcountif	Statistical function
decimal	Type
decrypt	Scalar function

Operator, Function, Element	Category
dedup	Operator
degrees	Mathematical function
distinct	Operator
double	Type
dynamic	Type
encrypt	Scalar function
endofday	DateTime function
endofmonth	DateTime function
endofweek	DateTime function
endofyear	DateTime function
endswith	Operator
!endswith	Operator
endswith_cs	Operator
!endswith_cs	Operator
eventstats	Operator
exp	Mathematical function
exp10	Mathematical function
exp2	Mathematical function
export	Operator
extend	Operator
externaldata	Operator
extract	String function
extract	Operator
extract_all	String function

Operator, Function, Element	Category
extract_json	String function
find	Operator
findearliest	Cribl function
findearliestif	Cribl function
findfirst	Cribl function
findfirstif	Cribl function
findlast	Cribl function
findlastif	Cribl function
findlatest	Cribl function
findlatestif	Cribl function
floor	Conversion function
foldkeys	Operator
format_bytes	INET function
format_datetime	DateTime function
format_ipv4	INET function
format_ipv4_mask	INET function
format_timespan	DateTime function
from_binary_string	Binary function
gamma	Mathematical function
.generate	Command
getmonth	DateTime function
gettype	Conversion function
getyear	DateTime function
has	Operator

Operator, Function, Element	Category
!has	Operator
has_all	Operator
!has_all	Operator
has_any	Operator
!has_any	Operator
has_any_index	String function
has_cs	Operator
!has_cs	Operator
!hasprefix	Operator
!hasprefix_cs	Operator
!hassuffix	Operator
!hassuffix_cs	Operator
hash	Hash function
hash_combine	Hash function
hash_many	Hash function
hash_md5	Hash function
hash_sha1	Hash function
hash_sha256	Hash function
hash_xxhash64	Hash function
hasprefix	Operator
hasprefix_cs	Operator
hassuffix	Operator
hassuffix_cs	Operator
hourofday	DateTime function

Operator, Function, Element	Category
iff	Conditional function
iif	Conditional function
in	Operator
indexof	String function
int	Type
in~	Operator
!in	Operator
!in~	Operator
ip-lookup	Operator
ipv4_compare	INET function
ipv4_is_in_any_range	INET function
ipv4_is_in_range	INET function
ipv4_is_match	INET function
ipv4_is_private	INET function
ipv4_netmask_suffix	INET function
ipv6_compare	INET function
ipv6_is_match	INET function
isempty	String function
isfinite	Mathematical function
isinf	Mathematical function
isnan	Mathematical function
isnotempty	String function
isnotnull	String function
isnull	String function

Operator, Function, Element	Category
join	Operator
let	Statement
limit	Operator
list	Cribl function
log	Mathematical function
log10	Mathematical function
log2	Mathematical function
loggamma	Mathematical function
long	Type
lookup	Operator
make_bag	Dynamic function
make_bag_if	Dynamic function
make_datetime	DateTime function
make_timespan	DateTime function
match_regex	String function
matches regex	Operator
max	Statistical function
max_of	Conditional function
maxif	Statistical function
median	Cribl function
medianif	Cribl function
min	Statistical function
min_of	Conditional function
minif	Statistical function

Operator, Function, Element	Category
monthofyear	DateTime function
mv-expand	Operator
mv-pull	Operator
next	Window function
not	Mathematical function
now	DateTime function
null	Type
objects	Command literal
options	Command literal
order	Operator
parse_csv	String function
parse_ipv4	String function
parse_ipv4_mask	String function
parse_ipv6	String function
parse_ipv6_mask	String function
parse_json	String function
parse_url	String function
parse_urlquery	String function
parse_version	String function
percentile	Statistical function
persecond	Cribl function
persecondif	Cribl function
pi	Mathematical function
pivot	Operator

Operator, Function, Element	Category
pow	Mathematical function
prev	Window function
print	Operator
project	Operator
project-away	Operator
project-rename	Operator
queries	Command literal
radians	Mathematical function
rand	Mathematical function
range	Operator
range	Mathematical function
rate	Cribl function
rateif	Cribl function
real	Type
render	Operator
replace_regex	String function
reverse	String function
round	Mathematical function
row_cumsum	Window function
row_number	Window function
row_rank_dense	Window function
row_rank_min	Window function
row_window_session	Window function
search	Operator

Operator, Function, Element	Category
send	Operator
set	Statement
.show	Command
sign	Mathematical function
sin	Mathematical function
sort	Operator
split	String function
sqrt	Mathematical function
startofday	DateTime function
startofmonth	DateTime function
startofweek	DateTime function
startofyear	DateTime function
startswith	Operator
!startswith	Operator
startswith_cs	Operator
!startswith_cs	Operator
stats	Command literal
stdev	Statistical function
stdevif	Statistical function
stdevp	Statistical function
strcat	String function
strcat_delim	String function
strcmp	String function
strftime	DateTime function

Operator, Function, Element	Category
string	Type
strlen	String function
strptime	DateTime function
strrep	String function
substring	String function
sum	Statistical function
sumif	Statistical function
summarize	Operator
sumsq	Cribl function
sumsqif	Cribl function
take	Operator
tan	Mathematical function
timespan	Type
timestats	Operator
to_binary_string	Binary function
tobool	Conversion function
todatetime	DateTime function
todecimal	Conversion function
todouble	Conversion function
toint	Conversion function
tolong	Conversion function
tolower	String function
top	Operator
top-hitters	Operator

Operator, Function, Element	Category
toreal	Conversion function
tostring	Conversion function
totimespan	DateTime function
toupper	String function
translate	String function
trim	String function
trim_end	String function
trim_start	String function
union	Operator
unixtime_microseconds_todatetime	DateTime function
unixtime_milliseconds_todatetime	DateTime function
unixtime_nanoseconds_todatetime	DateTime function
unixtime_seconds_todatetime	DateTime function
url_decode	String function
url_encode	String function
values	Cribl function
variance	Statistical function
varianceif	Statistical function
variancep	Statistical function
$vt_dataset_providers	Virtual table
$vt_datasets	Virtual table
$vt_dummy	Virtual table
$vt_jobs	Virtual table
$vt_list	Virtual table

Operator, Function, Element	Category
$vt_lookups	Virtual table
$vt_results	Virtual table
week_of_year	DateTime function
where	Operator
zip	Dynamic function

10.2. Operators In Cribl Search

A comprehensive list of all operators supported in Cribl Search, grouped by category.

An operator in Cribl Search is a query component that processes data, performing actions such as filtering, counting, or transforming events. Operators can use functions, and are delimited by the pipe character `|`.

For example, the `limit` operator sets the maximum number of events returned from a search:

```
dataset="cribl_search_sample"
 | limit 100
```

Copy | Run in Cribl Search

Aggregation Operators

Aggregation operators summarize data by grouping it based on specified fields and applying aggregation functions like `sum`, `avg`, or `max` to produce meaningful insights.

Name	Description
`count`	Returns the number of all input events.
`eventstats`	Aggregates events and adds the results as new fields to the source events.
`summarize`	Produces a table that aggregates the input data.
`timestats`	Aggregates events by time periods or bins..

Data Operators

Data operators transform, enrich, or manipulate events, enabling actions like renaming, joining, or exporting.

Name	Description
`centralize`	Forces subsequent operators to the coordinator.
`export`	Exports search results to a lookup or a Cribl Lake Dataset.
`extend`	Appends fields created by expressions.

Name	Description
`extract`	Extracts data from a field.
`foldkeys`	Folds hierarchical field names into a nested structure.
`ip-lookup`	Enriches events with IP address data.
`join`	Merges events from two different data scopes.
`lookup`	Enriches events with lookup files.
`mv-expand`	Expands an object into multiple events.
`mv-pull`	Pulls key-value pairs from array objects into events or objects/bags.
`project-rename`	Renames fields.
`union`	Appends one set of results to another.

Display Operators

Display operators control how data is presented in the output, allowing formatting, sorting, or visualizing results for better readability.

Name	Description
`limit` (`take`)	Limits the number of events.
`order` (`sort`)	Arranges events into order by one or more fields.
`print`	Outputs expression results.
`project`	Keeps only the fields specified, and can also rename fields and insert new computed fields.
`project-away`	Excludes specific fields from the results..
`range`	Generates a series of events.
`render`	Enforces a specific visualization of the search results.
`top`	Returns the first N events sorted by the specified fields.

Filter Operators

Filter operators exclude or include events based on specified conditions, allowing you to narrow down the Dataset to relevant records.

Name	Description
`dedup`	Filters out duplicate events.
`distinct`	Finds unique field values.
`search`	Finds events that contain the specified text.
`where`	Filters events based on the specified predicate.

Logical Operators

Logical operators perform comparisons and evaluations.

Operator	Description
`AND`	Returns `true` if both operands are `true`, otherwise returns `false`.
`NOT` or `!`	Returns `true` if the operand is `false`, otherwise returns `false`.
`OR`	Returns `true` if at least one operand is `true`, otherwise returns `false`.

Numerical Operators

Numerical operators perform arithmetic operations on numerical values, enabling calculations like addition or subtraction.

Operator	Description
`==`	Equal
`!=`	Not Equal
`>`	Greater Than
`>=`	Greater Than or Equal To
`<`	Less Than
`<=`	Less Than or Equal To

Operator	Description
`+`	Add
`-`	Subtract
`*`	Multiply
`/`	Divide
`%`	Modulo

Search Operators

Search operators retrieve events based on defined criteria, enabling efficient data discovery.

Name	Description
`cribl`	Finds specific events. The fundamental Cribl Search operator, implicit in queries that do not specify an operator.
`find`	Finds specific events.

String Operators

String operators manipulate and transform text, enabling actions like concatenation, trimming, replacement, or extraction.

Operator	Description	Case-Sensitive	Example (yields `true`)
`==`	Equals	Yes	`"aBc" == "aBc"`
`!=`	Not equals	Yes	`"abc" != "ABC"`
`=~`	Equals	No	`"abc" =~ "ABC"`
`!~`	Not equals	No	`"aBc" !~ "xyz"`
`contains`	RHS occurs as a subsequence of LHS	No	`"FabriKam" contains "BRik"`
`!contains`	RHS doesn't occur in LHS	No	`"Fabrikam" !contains "xyz"`

Operator	Description	Case-Sensitive	Example (yields `true`)
`contains_cs`	RHS occurs as a subsequence of LHS	Yes	`"FabriKam" contains_cs "Kam"`
`!contains_cs`	RHS doesn't occur in LHS	Yes	`"Fabrikam" !contains_cs "Kam"`
`endswith`	RHS is a closing subsequence of LHS	No	`"Fabrikam" endswith "Kam"`
`!endswith`	RHS isn't a closing subsequence of LHS	No	`"Fabrikam" !endswith "brik"`
`endswith_cs`	RHS is a closing subsequence of LHS	Yes	`"Fabrikam" endswith_cs "kam"`
`!endswith_cs`	RHS isn't a closing subsequence of LHS	Yes	`"Fabrikam" !endswith_cs "brik"`
`has`	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
`!has`	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
`has_all`	Same as `has` but works on all of the events	No	`"North and South America" has_all("south", "north")`
`!has_all`	Not all of the RHS terms are present in LHS	No	`"North and South America" !has_all("south", "east")`
`has_any`	Same as `has` but works on any of the events	No	`"North America" has_any("south", "north")`
`!has_any`	None of the RHS terms are present in LHS	No	`"North and South America" !has_any("east", "west")`
`has_cs`	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
`!has_cs`	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`
`hasprefix`	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
`!hasprefix`	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`

Operator	Description	Case-Sensitive	Example (yields `true`)
`hasprefix_cs`	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
`!hasprefix_cs`	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`
`hassuffix`	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
`!hassuffix`	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
`hassuffix_cs`	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
`!hassuffix_cs`	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`
`in`	Equal to any of the events	Yes	`"abc" in ("123", "345", "abc")`
`!in`	Not equal to any of the events	Yes	`"bca" !in ("123", "345", "abc")`
`in~`	Equal to any of the events	No	`"Abc" in~ ("123", "345", "abc")`
`!in~`	Not equal to any of the events	No	`"bCa" !in~ ("123", "345", "ABC")`
`matches regex`	LHS contains a match for RHS	Yes	`"Fabrikam" matches regex "b.*k"`
`startswith`	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
`!startswith`	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
`startswith_cs`	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
`!startswith_cs`	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

10.2.1. Aggregation Operators

A list of aggregation operators supported by Cribl Search.

Aggregation operators summarize data by grouping it based on specified fields and applying aggregation functions like `sum`, `avg`, or `max` to produce meaningful insights.

Name	Description
`count`	Returns the number of all input events.
`eventstats`	Aggregates events and adds the results as new fields to the source events.
`summarize`	Produces a table that aggregates the input data.
`timestats`	Aggregates events by time periods or bins..

10.2.1.1. Count

The `count` operator returns the number of all input events.



`count` is a short form of `summarize count=count()`.

Syntax

```
Scope | count
```

Arguments

- **Scope**: The events to search.

Results

Returns a single record and column of type long. The value is the number of records in the scope.

Example

- Count events where `status=200` and `response_time>2`.

```
dataset=myDataset status=200 response_time>2
| count
```

Copy

- Count the number of events.

```
dataset=$vt_dummy event<10
| count
```

Copy Run in Cribl Search

- Show the number of search jobs by their status and the user that created it. `$vt_jobs` requires **Admin** Permissions.

```
dataset=$vt_jobs
| summarize jobs=count() by status, user
```

Copy Run in Cribl Search

10.2.1.2. EVENTSTATS

The `eventstats` operator aggregates events and adds the results as new fields to the source events.

`eventstats` is similar to `summarize`, but it enriches the input events instead of replacing them.

By default, `eventstats` can aggregate up to 50,000 events at a time. You can change this limit with the **MaxNoOfAggregatedEvents** parameter.

Syntax

```
Scope | eventstats [max_events=MaxNoOfAggregatedEvents] [[AggregatedField =]
AggregationFunction [, ...]] [by [GroupField =] GroupingExpression [, ...]]
```

Arguments

- **Scope**: The events to aggregate and enrich.
- **MaxNoOfAggregatedEvents**: The maximum number of events to aggregate. After reaching this limit, aggregation stops, and all of the input events are enriched with the same, most recent aggregation results. Default: `50000`.
- **AggregatedField**: Optional name for a field that contains an aggregation result. Defaults to a name derived from the corresponding **AggregationFunction**.
- **GroupField**: Optional name for a group field. Defaults to a name derived from the corresponding **GroupingExpression**.
- **AggregationFunction**: A Cribl or statistical function, with field names as arguments. You can add multiple functions, separated by a comma. Wildcards are not supported for field names in aggregation functions.
- **GroupingExpression**: The expression by which `eventstats` groups the input events before aggregating them. You can add multiple expressions, separated by a comma.

Results

First, the input events are arranged into groups where the corresponding **GroupingExpression**s evaluate to the same values.

Then, the specified **AggregationFunction**s process each group. The results are added to the input events as new fields.

Examples

Calculate the average response time for all events, and add a new field that contains the result.

```
dataset="cribl_internal_logs"
| limit 100
| eventstats avgResponseTime = avg(response_time)
```

Copy | Run in Cribl Search

Calculate the average response time separately for each distinct value of the `src` field. Name the result field `srcAvgResponseTime`.

```
dataset="cribl_internal_logs"
| limit 100
| eventstats srcAvgResponseTime = avg(response_time) by src
```

Copy | Run in Cribl Search

Show only those events that have a response time greater than the average.

```
dataset="cribl_internal_logs"
| limit 100
| eventstats avgResponseTime = avg(response_time)
| where response_time > avgResponseTime
```

Copy | Run in Cribl Search

Calculate the ratio of events for each HTTP method.

```
dataset="cribl_internal_logs" method
| limit 1000
| summarize cnt=count() by method
| eventstats total=sum(cnt)
| project method, ratio = (cnt * 100 / total)
```

Copy | Run in Cribl Search

```
dataset=$vt_dummy event<1000
| extend randomNumber=rand(10)
| eventstats avg(randomNumber)
```

Copy | Run in Cribl Search

10.2.1.3. SUMMARIZE

The `summarize` operator produces a table that aggregates the input data.

`summarize` is similar to eventstats, but it replaces the input events instead of enriching them.

Syntax

```
Scope | summarize [[AggregatedField =] AggregationFunction [, ...]] [by [GroupField =] GroupingExpression [, ...]]
```

Arguments

- **Scope**: The events to search.
- **AggregatedField**: Optional name for a field that contains an aggregation result. Defaults to a name derived from the corresponding **AggregationFunction**.
- **GroupField**: Optional name for a group field. Defaults to a name derived from the **GroupingExpression**.
- **AggregationFunction**: Cribl and statistical functions, with field names as arguments. Wildcards are not supported for field names in aggregation functions.
- **GroupingExpression**: An expression that can reference the input data. The output will have as many events as there are distinct values of all the grouping expressions.

When the input table is empty, the output depends on whether **GroupingExpression** is used:

- If **GroupingExpression** is not provided, the output will be a single (empty) event.
- If **GroupingExpression** is provided, the output will have no events.

Results

The input events are arranged into groups having the same values of the `by` expressions. Then the specified aggregation functions are computed over each group, producing an event for each group. The result contains the `by` fields and also at least one field for each computed aggregate. (Some aggregation functions return multiple fields.)

The result has as many events as there are distinct combinations of `by` values (which may be zero). If there are no group keys provided, the result has a single record.

Examples

Minimum and maximum timestamp

Finds the minimum and maximum timestamp of all events in the Dataset. There is no `by` clause, so there is just one event in the output:

```
dataset="cribl_search_sample"
| summarize Min = min(_time), Max = max(_time)
```

Copy | Run in Cribl Search

Minimum and maximum across two fields

Finds the minimum `start` time and maximum `end` times across events in the Dataset. With no `by` clause, there is just one event in the output:

```
dataset="cribl_search_sample"
| summarize earliest = min(start), latest = max(end)
```

Copy | Run in Cribl Search

Distinct count

Create an event for each source address, showing a count of distinct ports. Because there are few values for `srcaddr`, no grouping function is needed in the `by` clause:

```
dataset="cribl_search_sample"
| summarize sources=dcount(srcport) by srcaddr
```

Copy | Run in Cribl Search

Create an event for each continent, showing a count of the cities in which activities occur. Because there are few values for `continent`, no grouping function is needed in the `by` clause:

```
dataset=myDataset
| summarize cities=dcount(city) by continent
```

Copy

Summarize a count of distinct destination ports, by both destination address and day start.

```
dataset="cribl_search_sample"
| summarize destinations=dcount(dstport) by dstaddr, startofday(start)
```

Copy | Run in Cribl Search

Count and total by category and month

Create a table with sell transactions and the total amount per fruit and sell month. The output fields show the count of transactions, transaction worth, fruit, and the `datetime` of the beginning of the month in which the transaction was recorded.

```
dataset=myDataset
| summarize NumTransactions=count(), Total=sum(UnitPrice * NumUnits) by Fruit, StartO
```

Copy

Count items in intervals

- Create a table that shows how many items have prices in each interval: [0,10.0], [10.0,20.0], and so on. This example has a field for the count and a field for the price range.

  ```
  dataset=myDataset
  | summarize count() by price_range=bin(price, 10.0)
  ```

  Copy

- Count events by 60-second intervals. formatting the results as human-readable times.

  ```
  dataset="cribl_search_sample"
  | summarize count() by minute=bin(start, 60), strftime(minute,"%X")
  ```

  Copy | Run in Cribl Search

- Count by parity.

  ```
  dataset=$vt_dummy event<10
  | extend answer=42, parity=iif(event%2==0, 'even', 'odd')
  | summarize count() by parity
  ```

Copy Run in Cribl Search

- Generate a table of users and the credits they have consumed. `$vt_jobs` requires **Admin** Permissions.

```
dataset=$vt_jobs
| summarize totalSeconds=sum(cpuMetrics.totalExecCPUSeconds) by user
| extend CreditsUsed=totalSeconds/3600
| project CreditsUsed, user
| render table
```

Copy Run in Cribl Search

10.2.1.4. TIMESTATS

The `timestats` operator aggregates events by time periods or bins.

Syntax

```
[... |] timestats [ span=Time[ SnapToTime ] | numBins=Bins ] [ timeSource=TimeSource ] [ timeLabel=TimeLabel ] [[ FieldName= ] Aggregation... ] [ by GroupExpression [, GroupExpression ]... ]
```

You can define either `span` or `numBins`, not both. If you define neither, this operator will automatically compute a suitable time span based on the search's time range.

Arguments

- **Time**: Time period. Supports these relative times – `s[econds]`, `m[inutes]`, `h[ours]`, `d[ays]`, `w[eeks]`, `mon[ths]`, and `y[ears]`. Values without units get interpreted as seconds. For example, `1 = 1s`.
- **SnapToTime**: Round down (backward) to the nearest instance of **Time**. Append the `@` modifier, followed by the same or more granular `Time`. For example,
  - `1d@d` snaps back to the beginning of today, 12:00 AM (midnight) UTC.
  - `1d@h` snaps to 4:00 PM the next day if it's currently 4:00 PM.
- **Bins**: Numeric literal, desired number of bins. Bins are split as close as possible based on the search's time range.
- **TimeSource**: String. Name of the time field to use for aggregation. Defaults to `_time`. The field needs to have time in seconds.
- **TimeLabel**: String. Name of the output time field. Defaults to the same field as **TimeSource**.
- **FieldName**: String. Name to give to the field with the results created by **Aggregation**.
- **Aggregation**: Cribl and statistical functions. Wildcards are not supported for field names in aggregation functions.
- **GroupExpression**: A scalar expression that can reference the input data. The output will have as many events as there are distinct values of all the group expressions.

Examples

- Create two fields per host in the format `Host:totalWaitTime` and `Host:avgResponseTime`, showing the aggregates for the time-bins (rows) of one minute:

```
timestats span=1m totalWaitTime=sum(responseTime), avgResponseTime=avg(responseT
```

Copy

- Create roughly 42 time bins, labeled as `when`, and create output fields in the form `srcaddr:totalRequest` and `dstaddr:totalRequest`:

```
dataset="cribl_search_sample"
| timestats timeLabel="when" numBins=42 totalRequest=count() by srcaddr, dstaddr
```

Copy Run in Cribl Search

- Aggregate time durations by interface, bin every minute:

```
dataset="cribl_search_sample"
| timestats span=1m totalTime=sum(end-start) by interface
```

Copy Run in Cribl Search

- Count events over time, bin every minute:

```
dataset=$vt_dummy event<600
| extend _time=_time-rand(600)
| timestats span=1m count()
```

Copy Run in Cribl Search

- Count events over time, using the `timeSource` option:

```
dataset=$vt_dummy event<600
| extend rando=_time-rand(600)
| timestats span=1m timeSource=rando count()
```

Copy Run in Cribl Search

- Count events over time, split by the field `method`, bin every minute:

```
dataset=$vt_dummy event<600
| extend _time=_time-rand(600), method=iif(event%2>0, "GET", "POST")
| timestats span=1m count() by method
```

10.2.2. Data Operators

A list of data operators supported by Cribl Search.

Data operators transform, enrich, or manipulate events, enabling actions like renaming, joining, or exporting.

Name	Description
`centralize`	Forces subsequent operators to the coordinator.
`export`	Exports search results to a lookup or a Cribl Lake Dataset.
`extend`	Appends fields created by expressions.
`extract`	Extracts data from a field.
`foldkeys`	Folds hierarchical field names into a nested structure.
`ip-lookup`	Enriches events with IP address data.
`join`	Merges events from two different data scopes.
`lookup`	Enriches events with lookup files.
`mv-expand`	Expands an object into multiple events.
`mv-pull`	Pulls key-value pairs from array objects into events or objects/bags.
`project-rename`	Renames fields.
`union`	Appends one set of results to another.

10.2.2.1. Centralize

The `centralize` operator enforces the execution of operators on the coordinator rather than on federated executors. This is useful where executors lack network visibility to the destination. You can centralize the results on a coordinator before sending them. This operator overrides the normal division of labor, in which:

- Executors process as close to your data as possible, on the remote end. They scan data – for example, reading from S3, decompressing, filtering, and projecting.
- Coordinators receive the partial results from the executors and perform post-aggregation and merging of the results.

Syntax

```
... | centralize
```

Examples

Centralize your data on a coordinator before sending it to a Cribl Stream Destination.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| centralize
| send
```

Copy Run in Cribl Search

```
dataset="cribl_search_sample"
| where dstport <= 1024 and action == REJECT
| centralize
| send
```

Copy Run in Cribl Search

```
dataset=myDataset method or status
| where status > 299
| centralize
| send
```

Copy

10.2.2.2. Export

The `export` operator does one of the following:

- Sends results generated by a search to an existing Cribl Lake Dataset. (You must create the Dataset first.)
- Creates or updates a lookup from search results.



Only Admin and Editor Search Members can run the `export` operator. See also specific limitations on exporting to Datasets and to lookups.

Syntax

Export to Cribl Lake:

```
Scope | export [ suppressPreviews=Previews ] to [ lake ] LakeDatasetName [ tee=Tee ]
```

Copy



The `lake` keyword is optional and has no effect.

Export to a lookup:

```
Scope | export [ mode=Mode ]
              [ description=Description ]
              [ suppressPreviews=Previews ]
              [ fieldMapping=PairsToMap ]
              [ compress=Compress ]
              to lookup LookupName
              [ tee=Tee ]
              [ maxEvents=MaxEvents ]
```

Copy

Arguments



Arguments are case-insensitive.

`lake`: Export to a Cribl Lake Dataset

- **Scope**: The events to search.
- **Previews**: Boolean. If `false` (default), Cribl Search displays intermediate statistics (previews) during the export operation. If `true` (so, `suppressPreviews=true`), you'll see only the final statistics, once the export operation completes.
- **LakeDatasetName**: The Cribl Lake Dataset to export results to. You must choose an already existing Dataset.
- **Tee**: Boolean. If set to `true`, the search results are shown as usual. If `false` (which is the default), instead of the results, Cribl Search displays statistics related to the export.

Dataset Limitations

Typically, you're exporting data in which Cribl Search has already parsed field names and values. This makes the data compatible with Lakehouses. However, if you happen to send **unparsed** data, you won't be able to search it at Lakehouse speed.

The `export` operator can't send data to a Cribl Lake Dataset that is populated via Splunk Cloud Direct Access.

`lookup`: Export to a Lookup

- **Scope**: The events to search.
- **Mode**: Supports `create`, `overwrite`, and `append`. Defaults to `create`. If the lookup doesn't exist, it defaults to `create`, simplifying scheduled searches. When using `append` mode, you can include `description`, `fieldMapping`, and `compress` parameters, and if the lookup exists, they are ignored. This streamlines the process of appending data to a lookup as part of scheduled searches, as the table is generated during the first scheduled run and continues to append new content in subsequent runs.
- **Description**: String. The description to give the lookup.
- **Previews**: Boolean. If `false` (default), Cribl Search displays intermediate statistics (previews) during the export operation. If `true` (so, `suppressPreviews=true`), you'll see only the final statistics, once the export operation completes.
- **PairsToMap**: String. Comma-separated list of `sourceName:targetName` pairs to map event property names to CSV output names. Defaults to all of the last known list of projected fields.
- **Compress**: String. Defaults to `auto` where files larger than 16 KB are compressed. Set as `true` to compress no matter the size and `false` to not compress the file.
- **LookupName**: The lookup file name, without the file extension. For example: a file name of `users.csv` is entered as `users`.

- **Tee**: Boolean. Defaults to `false` where search results are not shown in Cribl Search. Instead, you get an event with the URL and the number of bytes and events sent or dropped. When `true` the search results are displayed and no stats are provided. For example: `tee=true`.
- **MaxEvents**: The maximum number of events that `export` will write to the table. Cribl Search will drop any events beyond this limit. Defaults to `10000` (10,000 rows). This is also the maximum number of events that this operator's `lookup` mode can handle, so entering a higher **MaxEvents** still effectively sets a limit of `10000`.

Lookup Limitations

You can't use the `create` mode on an existing lookup. This will trigger an error.

You can't use the `export` operator to create or update Pack lookups. For example, `| export to lookup pack(packId).lookupId` will **not** work.

`_raw` in Exports to Cribl Lake

Exports to Lake Datasets include the `_raw` field. If you want to remove this field, use the `project-away` operator.

`_time` in Exports to Cribl Lake

Cribl Lake handles the `_time` field automatically, so Datatype timestamp settings in Cribl Search don't apply.

If the source event has no `_time` field, Cribl Lake adds this field, setting its value to `now()` at write time.

Examples

Export search results to the `myDataset` Lake Dataset:

```
dataset="cribl_search_sample"
| export to lake myDataset

// you can also drop the `lake` keyword
dataset="cribl_search_sample"
| export to myDataset
```

Copy

Export search results to a Lake Dataset, and remove the `_raw` field:

```
dataset="cribl_search_sample"
| export to lake myDataset
| project-away _raw
```

Copy

Create a lookup named `mylookup` with the `time` and `status` fields.

```
dataset="cribl_search_sample" status="*"
| limit 5
| export description="My lookup" fieldMapping="_time:time,status:status" to lookup my
```

Copy Run in Cribl Search

You can verify the contents of the lookup with this command – replacing `mylookup` with the name of your own lookup file:

```
dataset="$vt_lookups" lookupFile="mylookup"
```

Copy Run in Cribl Search

Export a lookup with two columns, `method` and `cnt`.

```
dataset=$vt_dummy event<600
| extend _time=_time-rand(600), method=iif(event%2>0, "GET", "POST")
| summarize cnt=count() by method
| export mode=append description="Table with http methods count" to lookup mymethods
```

Copy Run in Cribl Search

To verify that table was exported.

```
dataset="cribl_lookups" lookup_table=mymethods
| project method, cnt
| render table
```

Copy Run in Cribl Search

10.2.2.3. Extend

The `extend` operator calculates one or more expressions and assigns the results to fields.

Syntax

```
Scope | extend [FieldName | (FieldName[, ...]) =] Expression [, ...]
```

Arguments

- **Scope**: The events to search.
- **FieldName**: Optional. The name of the field to add or update. If omitted, the name will be generated.
    - If **Expression** returns more than one field, a list of field names can be specified in parentheses. In this case, **Expression**'s output fields will be given the specified names, dropping the rest of the output fields, if there are any.
    - If a list of the field names is not specified, all **Expression**'s output fields with generated names will be added to the output.
- **Expression**: A calculation over the fields of the input.

Results

Resulting fields are returned in one of the following ways:

- Field names noted by `extend` that already exist in the input are removed and appended as their new calculated values.
- Field names noted by `extend` that do not exist in the input are appended as their new calculated values.

Examples

To calculate and add fields called `Duration` and `IsSevere`:

```
dataset=myDataset
| extend Duration = CreatedOn - CompletedOn
, IsSevere = Level == "Critical" or Level == "Error"
```

Copy

Monitoring Disk Operations by Ratio and Status

If you're using Cribl Edge, you can use `extend` to calculate the ratio of read and write operations on each Edge Node. Based on the ratio, you can use the iif function to give each Node a status, indicating any imbalances or I/O issues:

```
dataset="cribl_edge_metrics"
| extend Disk_Operation_Ratio = node_disk_reads_completed_all_total / node_disk_write
| extend operation_status = iif(Disk_Operation_Ratio > 2, 'High Read', iif(Disk_Opera
| summarize count() by operation_status
```

Copy | Run in Cribl Search

Monitoring CPU Usage by Level

You can use `extend` to monitor CPU usage on each Edge Node. The following query breaks CPU usage into two categories: `High` for CPU usage greater than 70%, and `Low` for anything below that threshold. Then, the `timestats` operator allows you to observe how these categories change over time.

```
dataset=cribl_edge_metrics
| extend CPU_Usage_Per_Host = iif(node_cpu_percent_active_all > 70, 'High', 'Low'), H
| timestats by CPU_Usage_Per_Host
```

Copy | Run in Cribl Search

```
dataset=$vt_dummy event<10
| extend answer=42, parity=iif(event%2==0, 'even', 'odd')
```

Copy | Run in Cribl Search

10.2.2.4. EXTERNALDATA

The `externaldata` operator fetches external data from HTTP(S) URLs, including public APIs.

If you frequently use `externaldata` to search the same API endpoints, consider setting up a Generic HTTP API Dataset. See also the Usage Note below.

Syntax

```
externaldata [ ConnectionURL [, ...] ] [ with ( PropertyName = PropertyValue [, ...]) ]
```

The outer brackets surrounding **ConnectionURL** are required and literal. They do not indicate an optional argument (as the inner brackets do).

Arguments

Name	Type	Required	Description
ConnectionURL	string	Yes	The HTTP(S) URL from which to fetch data.
PropertyName, **PropertyValue**	string	No	A list of optional properties that determine how to interpret the retrieved data.

Properties

Property	Type	Description
dataField	string	The name of the field (in the response JSON) to pull data from. Leave blank if the result is an array.
datatype	string	The data type to use to parse the data. If none is specified, tries to parse the data as a JSON array.
headers	string	A JSON object containing the headers to send with the request.
method	`GET` or `POST`	The HTTP method used when making the API request.

Usage Note

Unlike most other operators, `externaldata` includes – in the query – all that is needed to establish connectivity to remote data. This might include sensitive key information in cleartext.

The query is logged (like other queries), and the log events might include this sensitive information. The Dataset where it's logged is readable only by users with elevated privileges. However, to avoid any exposure of credentials in cleartext, consider configuring a generic HTTP API Dataset as an alternative to this operator.

Examples

Simple

```
externaldata
[
  "https://cat-fact.herokuapp.com/facts"
]
```

Copy

With a dataField

```
externaldata
[
  "https://vpic.nhtsa.dot.gov/api/vehicles/getallmanufacturers?format=json"
]
with(
  dataField="Results"
)
```

Copy

With a Datatype (ndjson file)

```
externaldata
[
  "https://gist.githubusercontent.com/rfmcnally/0a5a16e09374da7dd478ffbe6ba52503/raw/
]
with(
  datatype="Cribl Search"
)
```

Copy

Specifying HTTP headers

```
externaldata
[
  "https://vpic.nhtsa.dot.gov/api/vehicles/getallmanufacturers?format=json"
]
with(
  dataField="Results",
  headers='{"user-agent": "Foo", "otherHeader": "Bar"}'
)
```

Copy

Get weather data for New York City.

```
externaldata ["https://wttr.in/NewYork?format=j1"]
| mv-expand weather.0.hourly
```

Copy Run in Cribl Search

10.2.2.5. Extract

The `extract` operator extracts information from a field (which defaults to `_raw`), using either a predefined Parser or various `Type` options (such as `delimited`, `regex`, `kvp`, `json`, `csv`, `elff`, or `clf`).



To extract a substring from a source string, using an RE2 regular expression, see the `extract` scalar string function.

Syntax

```
... | extract [ SourceOption ] [ type=Type | parser=Parser ]
```

- `[ type=delimited [ DelimOption ] FieldList ]`
- `[ [ type=regex ] [ RegexOption ] RegexLiteral [RegexLiteral[...]] ]`
- `[ type=keyvalue [ FieldList ]]`
- `[ type=json [ FieldList ]]`
- `[ type=csv FieldList ]`
- `[ type=elff FieldList ]`
- `[ type=clf [ FieldList ]]`

Rules

This operator automatically detects the `type` when you define **RegexLiteral** or **DelimOption**. For example, if you provide **RegexLiteral**, it's assigned as `type=regex`. Or, if you specify **DelimOption**, it's assigned as `type=delimited`.

You can define either **DelimOption** or **RegexOption**, not both.

You can have multiple **FieldList**s. These will be merged into one larger field list.

A **FieldList** is optional for the key value, JSON, and CLF types. If no **FieldList** is specified:

- Key value and JSON types extract all fields.
- CLF type extracts `clientip`, `ident`, `user`, `timestamp`, `request`, `status`, and `bytes`.

Arguments

Here are syntax details and options for each of this operator's arguments.

SourceOption

Syntax – `[ SourceIdent=FieldName ]`.

Defaults to `source=_raw`.

- **SourceIdent**: Define as `source`, `sourceField`, or `src`.
- **FieldName**: Name of the field to extract from. Case sensitive.

Type

Syntax – `[ Type=TypeName ]`. Example: `type=csv`

- **Type**: Define as either `type` or `format`.
- **TypeName**: See the following table.

TypeName	Define as
Delimiter	`delimited`, `del`, or `delim`
Regex	`regex`, `regexp`, or `re`
Key Value	`kvp`, `keyvalue`, or `kv`
JSON	`json` or `js`
CSV	`csv`
ELF	`elff` or `extended`
CLF	`clf` or `common` (case insensitive)

FieldList

Syntax – `"FieldName" [, "FieldNameList" ]`.

Case-sensitive. Example: `"field1,field2"`

Alternative syntax – `FieldIdent="FieldNameList"`.

Example: `fields=field1,field2`

- **FieldIdent**: Define as either `field` or `fields`.
- **FieldNameList**: Syntax – `FieldName [, FieldNameList ]`. Case sensitive.

Parser

Syntax – `parser="KnownParserName"`.

Use a predefined Parser. Case-sensitive.

RegexLiteral

Syntax – `@'NamedRegularExpression'`.

At least one named capture group is required. Example: `@'metric1=(?<metric1>\d+)'`

To extract key-value pairs and assign them to fields, use the capture group names `_NAME_0` and `_VALUE_0`. Example: `@'\"(?<_NAME_0>\w+)\s\/(?<_VALUE_0>.*)\"'`

Alternative syntax – `RegexIdent=@'NamedRegularExpression'`.

Example: `regex=@'metric1=(?<metric1>\d+)'`

- **RegexIdent**: Define as either `regex` or `regexp`.

DelimOption

Syntax – `[[ DelimCharOpt ] [ EscapeCharOpt ] [ QuoteCharOpt] [ NullValOpt ]]`

Argument	Define as	Syntax	Default Value	Example
DelimCharOpt	`delimchar`, `delimiter`, or `delim`	`DelimCharOpt=char`	`;`	`delimiter=:`
EscapeCharOpt	`escapechar`, `escape`, or `esc`	`EscapeCharOpt=char`	`\`	`esc=\`
QuoteCharOpt	`quotechar` or `quote`	`QuoteCharOpt=char`	`"`	`quotechar='`
NullValOpt	`nullvalue` or `undefined`			

RegexOption

Syntax – `[[ IterationOpt ] [ OverwriteOpt ]]`

Argument	Define as	Syntax	Default Value	Example
IterationOpt	`numiterations`, `iterations`, or `iter`	`IterationOpt=integer`	`100`	`iterations=11`
OverwriteOpt	`overwrite` or `ow`	`OverwriteOpt=boolean`	`false`	`ow=true`

Examples

This example extracts the domain name from an email address:

```
dataset=$vt_dummy event<10
| extend email="test@example.com"
| extract source=email @'@(?<domainName>.*$)'
```

Copy | Run in Cribl Search

This example extracts delimited field values:

```
dataset=$vt_dummy event<10
| extend myField="value1|value2|value3"
| extract source=myField type=delim delimChar="|" "field1,field2,field3"
```

Copy | Run in Cribl Search

This example extracts and names four fields from the CSV-formatted source field `foobar`:

```
dataset=$vt_dummy events < 10
| extend foobar="foo,bar,baz,quux"
| extract source=foobar type=csv "field1,field2,field3,field4"
```

Copy | Run in Cribl Search

This example extracts all key-value pairs from the default source field `_raw`:

```
dataset=myDataset
| extract type=keyvalue
```

Copy

This example extracts all key-value pairs from the named source field `payload`:

```
dataset=$vt_dummy events<10
| extend payload="time=now key=value message=\"many words here\""
| extract source=payload type=keyvalue
```

Copy | Run in Cribl Search

This examplee uses a Parser to extract all fields from `mySource`:

```
dataset=myDataset
| extract source="mySource" parser="Apache Common Log Format"
```

Copy

This example extracts delimited values (from `_raw`) using the `|` as the delimiter. The `type` specification could be omitted in this case, because the `type` is implicitly given by specifying the `delimChar` option:

```
dataset=myDataset
| extract type=delim delimChar="|" "field1,field2"
```

Copy

This example extracts the fields `metric1` and `foo` from `_raw` (default source) via named regular expressions. The `type` specification could be omitted in this case, because the `type` is implicitly given by specifying regular expressions.

```
dataset=myDataset
| extract type=regex @'metric1=(?<foo>\d+)'
```

Copy

10.2.2.6. FOLDKEYS

The `foldkeys` operator takes field names that contain a separator character, and folds them into a nested structure. This operator supports regex literals.

```
// fold all field names by `/`, and delete the original entries
Scope
 | foldkeys separator="/"
```

Copy

Purpose

Use `foldkeys` to normalize hierarchical field names. If you have data where a hierarchy of fields is encoded in the name (for example, `email.address` and `email.user`), `foldkeys` can transform the keys (not the values) into a nested structure.

Syntax

```
Scope | foldkeys [delete_original=true|false] [separator="<Separator>"] ["<SelectionRegex>"]
```

Parameters

Parameter	Required	Description
`Scope`		Events to search.
`delete_original`		`true` (default): output only the folded field names. `false`: output both the folded and original field names.
`separator`		Set a `Separator` character to use as the path separator, for example: `separator="/"`. The default is `.`.
`SelectionRegex`		Optional regular expression. If defined, only field names that match this expression are folded (along with the `SeparatorCharacter`).

To pass regex literals as **SelectRegex** parameters, see syntax details at Regex Examples, Regex Flags, and Disambiguate Regex Characters.

Returns

All the input events, but with hierarchical field names transformed into nested structures.

Examples

Fold All Field Names

Fold all field names, using the default `.` separator. Don't keep the original field names in the output.

```
dataset="$vt_dummy" event < 1
 | extend ["email.address"]="foo", ["email.user"]="bar"
 | foldkeys
```

Copy | Run in Cribl Search

Input (before `foldkeys`):

```
{
  "_time": 1723036428.379,
  "event": 0,
  "dataset": "$vt_dummy",
  "email.address": "foo",
  "email.user": "bar"
}
```

Copy

Output (after `foldkeys`):

```
{
  "_time": 1723036570.006,
  "event": 0,
  "dataset": "$vt_dummy",
  "email": {
    "address": "foo",
    "user": "bar"
  }
}
```

Copy

Fold Selected Field Names and Keep the Originals

Fold all field names that start with `eMail` (case-insensitive), using `/` as a separator. Keep the original field names in the output.

```
dataset="$vt_dummy" event < 3
 | extend ["email/name"]=strcat("foo", event +5), ["email/number"]=event * 3, ["rolf/
 | foldkeys delete_original=false separator="/" '/^eMail/i'
```

Copy Run in Cribl Search

Input (before `foldkeys`):

```
// event 0
{
  "_time": 1723036048.423,
  "event": 0,
  "dataset": "$vt_dummy",
  "email/name": "foo5",
  "email/number": 0,
  "rolf/sub/field": 10
}

// event 1
{
  "_time": 1723036048.423,
  "event": 1,
  "dataset": "$vt_dummy",
  "email/name": "foo6",
  "email/number": 3,
  "rolf/sub/field": 11
}

// event 2
{
  "_time": 1723036048.423,
  "event": 2,
  "dataset": "$vt_dummy",
  "email/name": "foo7",
  "email/number": 6,
  "rolf/sub/field": 12
}
```

Copy

Output (after `foldkeys`):

```
// event 0
{
  "_time": 1723036188.255,
  "event": 0,
  "dataset": "$vt_dummy",
  "email/name": "foo5",
  "email/number": 0,
  "rolf/sub/field": 10,
  "email": {
    "name": "foo5",
    "number": 0
  }
}

// event 1
{
  "_time": 1723036188.255,
  "event": 1,
  "dataset": "$vt_dummy",
  "email/name": "foo6",
  "email/number": 3,
  "rolf/sub/field": 11,
  "email": {
    "name": "foo6",
    "number": 3
  }
}

// event 2
{
  "_time": 1723036188.255,
  "event": 2,
  "dataset": "$vt_dummy",
  "email/name": "foo7",
  "email/number": 6,
  "rolf/sub/field": 12,
  "email": {
    "name": "foo7",
    "number": 6
  }
}
```

Copy

10.2.2.7. IP-LOOKUP

The `ip-lookup` operator retrieves geolocation data of IP addresses using MaxMind's GeoIP2 and GeoLite2 or IPinfo's IP address databases to enrich events with details such as city, continent, country, latitude, longitude, postal code, region, and time zone based on the provided IP address field. You can customize which fields to return, add prefixes, and their language.



You can display `ip-lookup` results in a Map Chart.

Prerequisites

The `ip-lookup` operator needs a `.mmdb` database file. You need to download the file from MaxMind or IPinfo and then create a lookup table with it. This file contains the geolocation data needed for IP address enrichment.

- MaxMind's GeoIP2 City, GeoIP2 Country, and GeoLite2 are supported. The GeoIP2 databases are a more accurate version of the free GeoLite2 City database.
- To download the free GeoLite2 City database you need a free GeoLite2 account. For details, see MaxMind's Accessing GeoLite2 Free Geolocation Data documentation.
- IPinfo's IP Geolocation, IP to privacy detection, ASN, IP to Company, Abuse contact, and WHOIS databases are supported.
- IPinfo also provides the following free IP databases: IP to Country, IP to ASN, and IP to Country ASN.

Syntax

```
Scope | ip-lookup [ output=OutputField[, ...] ] [ prefix=Prefix ] [ lang=Lang ]
LookupTable [ on IPField ]
```

Arguments

- **Scope**: The events to search.
- **OutputField**: Field(s) to return from the lookup. By default, all available fields are returned.
  - `city` – for example, `Georgetown`
  - `continent` – for example, `North America`
  - `continent_code` – for example, `NA`

    - `country` – for example, `United States`
    - `country_code` – for example, `US`
    - `lat` – numeric latitude coordinate
    - `lon` – numeric longitude coordinate
    - `postal` – for example, `40324`
    - `region` – for example, `Kentucky`
    - `region_code` – for example, `KY`
    - `time_zone` – for example, `America/Los_Angeles`
- **Prefix**: A prefix to add to the output fields. For example, if the prefix is set to `ip_`, output fields will be named `ip_city ip_country`, etc.
- **Lang**: Specifies the language for data retrieval. Defaults to `en` for English. Supports Brazilian Portuguese (pt-BR), English (en), French (fr), German (de), Japanese (ja), Russian (ru), Simplified Chinese (zh-CN), and Spanish (es).
- **LookupTable**: The lookup filename that contains the database. Do not add the `.mmdb` file extension. For example, a filename of `GeoLite2-city.mmdb` is entered as `GeoLite2-city`.
- **IPField**: The field name with an IP address. Defaults to `ip`.

Examples

The runnable examples below require you to first acquire and upload a compatible `.mmdb` file, as outlined above in Prerequisites.

- Lookup geolocation data on the `dstaddr` field:

```
dataset="cribl_search_sample" dataSource=*vpc* | limit 1000  | ip-lookup geocity
```

Copy | Run in Cribl Search

- Lookup latitude and longitude data on the `dstaddr` field:

```
dataset="cribl_search_sample" dataSource=*vpc* | limit 1000  | ip-lookup output=
```

Copy | Run in Cribl Search

- Lookup geolocation data on the `ip` field:

```
dataset=myDataset
| ip-lookup 'GeoLite2-City'
```

Copy

- Lookup geolocation data on the `my_ip` field:

```
dataset=myDataset
| ip-lookup 'GeoLite2-City' on my_ip
```

Copy

- Lookup geolocation data on the `ip_address` field and return only the prefixed `ip_lat` and `ip_lon` fields:

```
dataset=myDataset
| ip-lookup output=lat,lon prefix=ip_ 'GeoLite2-City' on ip_address
```

Copy

- Lookup geolocation data on the `ip_address` field and return only the `city` and `region` fields in Spanish:

```
dataset=myDataset
| ip-lookup output=city,region lang=es 'GeoIP2-City' on ip_address
```

Copy

10.2.2.8. Join

The `join` operator merges events from two different data scopes.

You can merge data coming from different Datasets and Dataset Providers, and then process the output for further analysis.

Syntax

Using a `let` statement:

```
let RightScopeName = RightScope;

LeftScope
  | join [ JoinOptions ] RightScopeName on JoinConditions
```

Copy

Using an inline subquery:

```
LeftScope
  | join [ JoinOptions ] (
    RightScope
    )
    on JoinConditions
```

Copy

Arguments

- **RightScopeName**: The name for the **RightScope**. Spaces ( ) are not allowed.
- **RightScope**: The data on the right-hand side of the join operation. Needs to be assigned a name, using the let statement. Can be a single value, an expression, or an entire query, for example: `dataset="cribl_internal_logs" | limit 1000`.
- **LeftScope**: The data on the left-hand side of the join operation. Can be a single value, an expression, or an entire query.
- **JoinOptions**: The join operation's optional settings. Syntax: `option1=value1[, option2=value2, ...]`. The following options are available:

- `kind=String`: The type of join to perform. Can be `inner`, `innerunique`, `leftanti`, or `leftouter`. Default: `inner`.
- `overwrite=Bool`: Applies to fields with the same name on both **LeftScope** and **RightScope**. If set to `true`, this operator uses the values of the **RightScope** fields. If `false`, it uses the values of the **LeftScope** fields. Default: `false`.
- `caseInsensitive=Bool`: If `true`, the join operation is case-insensitive. Default: `false`.

- **JoinConditions**: The conditions on which the events are merged. Separate multiple conditions with a comma (`,`). Use the following syntax:
  - `FieldName`, if the fields you want to match have the same name in both Datasets.
  - `$left.LeftFieldName == $right.RightFieldName`, if you want to match fields that have different names.

Rules

The **RightScope** must:

- Either be assigned a name, using a let statement.
- Or be an inline subquery.

The join operation includes the first 50,000 events of the **RightScope**. The remaining events are ignored.

Types of Joins

By default, Cribl Search performs an **inner** join, which returns the following:

Scope	What's included in the output
LeftScope	Events that match the join conditions, including the matching keys
RightScope	Events that match the join conditions, including the matching keys

To perform an **innerunique** join, use the `kind=innerunique` setting, which returns the following:

Scope	What's included in the output
LeftScope	Unique events that match the join conditions, including the matching keys
RightScope	Unique events that match the join conditions, including the matching keys

To perform a **leftanti** join, use the `kind=leftanti` setting, which returns the following:

Scope	What's included in the output
LeftScope	Events that don't match the join conditions
RightScope	No events

To perform a **leftouter** join, use the `kind=leftouter` setting, which returns the following:

Scope	What's included in the output
LeftScope	All events
RightScope	Events that match the join conditions, including the matching keys

Examples

Using an inline subquery, join internal Cribl logs related to HTTP requests with a Dataset containing HTTP status codes.

```
// LeftScope
dataset="cribl_internal_logs" method status
 | where isnotnull(status)
 | join kind=leftouter (
    // RightScope (inline subquery)
    cribl dataset=$vt_dummy event < 4
     | extend x =
            case(event == 0, dynamic({"status": 200, "text": 'OK'}),
                 event == 1, dynamic({"status": 308, "text": 'Redirect'}),
                 event == 2, dynamic({"status": 404, "text": 'Not found'}),
                             dynamic({"status": 500, "text": 'Internal Error'}))
     | project statusNum=x.status, statusText=x.text
 ) on $left.status == $right.statusNum
```

Copy | Run in Cribl Search

Join two scopes of data, based on a single namesake field (`status`). Returns only those `LeftScope` events that have `status`. Those events are also enriched with the `status` fields from the `RightScope`.

```
// RightScope
let codeMap = dataset=$vt_dummy event < 4
 | extend x =
   case(event == 0, dynamic({"status": 200, "text": 'OK'}),
        event == 1, dynamic({"status": 308, "text": 'Redirect'}),
        event == 2, dynamic({"status": 404, "text": 'Not found'}),
                    dynamic({"status": 500, "text": 'Internal Error'}))
 | project status=x.status, statusText=x.text;

dataset="cribl_internal_logs" method status
 | where isnotnull(status) // LeftScope
 | join kind=leftouter codeMap on status
```

Copy Run in Cribl Search

Show blocklisted IPs in VPC flow logs and sensitive ports their network requests are accessing.

```
// first RightScope (dummy list of blocklisted IPs)
let pretendBlocklistedIps = dataset="cribl_search_sample" dataSource="vpcflowlogs"
 | limit 10
 | distinct srcaddr
 | project blocklistedIp=srcaddr;

// second RightScope
let servicePortsToWatch = dataset="cribl_lookups" lookup_table="service_names_port_nu
 | project port_number, service_name;

// LeftScope
dataset="cribl_search_sample" dataSource="vpcflowlogs"
 | limit 1000
 | join pretendBlocklistedIps on $left.srcaddr == $right.blocklistedIp
 | join servicePortsToWatch on $left.dstport == $right.port_number;
```

Copy Run in Cribl Search

Perform a join with the `overwrite` option set to `true`. Returns only those `LeftScope` events that have the `name` field, but the `RightScope` `name` values overwrite the `LeftScope` `name` values.

```
let RightScope = dataset=$vt_dummy event < 10
 | extend name=strcat('NameFromRightScope', event);

dataset=$vt_dummy event < 10
 | extend name=strcat('NameFromLeftScope', event) // LeftScope
 | join overwrite=true RightScope on event;
```

Copy Run in Cribl Search

Create a code map for different event statuses and then join it with another Dataset to produce a distinct list of methods along with their corresponding status codes and descriptive texts.

```
let codeMap = dataset=$vt_dummy event < 4
 | extend x =
   case(event == 0, dynamic({"status": 200, "text": 'OK'}),
        event == 1, dynamic({"status": 308, "text": 'Redirect'}),
        event == 2, dynamic({"status": 404, "text": 'Not found'}),
                    dynamic({"status": 500, "text": 'Internal Error'}))
 | project status=x.status, statusText=x.text;

dataset="cribl_int*" method status
 | where isnotnull(status)
 | join codeMap on status
 | distinct method, status, statusText;
```

Copy Run in Cribl Search

10.2.2.9. Lookup

The `lookup` operator retrieves fields from a lookup through a first-match lookup process. It returns all rows of the table by default, incorporating any additional fields from the lookups wherever there is a match.



To enrich events with IP address data, use the `ip-lookup` operator.

Syntax

```
Scope | lookup [ output=OutputFields[, ...] [ defaultVal=Default ]]
               [ ignoreCase=Case ]
               [ matchMode=Mode [ matchType=Type ]]
               LookupTable on Conditions[, ...]
```

Copy

Arguments

The arguments are case-insensitive.

- **Scope**: The events to search.
- **OutputFields**: Field(s) to add from the lookup after matching. Defaults to **all** if not specified.
- **Default**: Assigns a default value to **OutputFields** when no value is present. Ignored if `output` is not defined.
- **Case**: Matching ignores case by default; set as `true`. Accepts boolean values – `yes`, `true`, `t`, `1` or `no`, `false`, `f`, `0`.
- **Mode**: Set to the matching logic of the lookup field(s). Supports `exact`, `cidr`, and `regex`. Defaults to `exact`. For example, if the lookup field contains a regular expression you'd use `regex`.
- **Type**: How to resolve multiple matches for `cidr` and `regex` modes. Defaults to `first`.
  - `first` returns the first matching entry.
  - `specific` scans all entries for the most specific match.
  - `all` returns all matches in the output, as arrays.
- **LookupTable**: The lookup filename, with or without the extension (you can enter `foo.csv` as `foo`). If the filename contains spaces, surround it with quotes (enter `foo bar.csv` as `"foo bar"`, `'foo bar'`, or `['foo bar']`).

- **Conditions**: Field(s) to look for in the lookup. There are three supported syntax variations:
  - Looking in the lookup for the **exact** field name as it appears in events. Syntax: `CommonFieldName`.
  - Single mapping between the event field and the lookup field name. Syntax: `EventFieldName = lookupFieldName`.
  - A join condition that specifies how to combine the data from the two tables. Syntax: `$left.eventFieldName == $right.lookupFieldName` where the event is referenced as `$left` and the lookup as `$right`. You can have `$left` and `$right` on either side of the expression.

Examples

Look for a common field in the events and the `service_names_port_numbers.csv` lookup file.

```
dataset="cribl_search_sample"
 | limit 100
 | lookup service_names_port_numbers on commonField
```

Copy Run in Cribl Search

Map an event field (`dstport`) to a lookup field (`port_number`).

```
dataset="cribl_search_sample"
 | limit 100
 | lookup service_names_port_numbers on dstport=port_number
```

Copy Run in Cribl Search

Look for a common field, a mapped field, and use a join condition.

```
dataset="cribl_search_sample"
 | limit 100
 | lookup service_names_port_numbers on commonField, eventField1=lookupField1, $left.
```

Copy Run in Cribl Search

Look up an IP address range, using CIDR notation.

```
dataset="cribl_search_sample"
 | limit 100
 | lookup matchMode='cidr' matchType='all' service_names_port_numbers on $left.hostip
```

Copy Run in Cribl Search

Match using a regular expression.

```
dataset="cribl_search_sample"
 | limit 100
 | lookup matchMode='regex' matchType='all' service_names_port_numbers on eventfield=
```

Copy Run in Cribl Search

Create a lookup to use in the next example. This search won't display any results, but it creates a lookup file named `mymethods.csv`.

```
dataset=$vt_dummy event<600
 | extend _time=_time-rand(600), method=iif(event%2>0, "GET", "POST")
 | summarize cnt=count() by method
 | export description="Table with http methods count" to lookup mymethods
```

Copy Run in Cribl Search

Enrich events with a field `cnt` that comes from the `mymethods.csv` lookup file created in the example above.

```
dataset=$vt_dummy event<100
 | extend method=split("GET,POST", ",")
 | mv-expand method
 | lookup mymethods on method
```

Copy Run in Cribl Search

Use a lookup that's part of a Pack:

```
dataset=myDataset
 | lookup pack(myPack).LookupTable on commonField
```

Copy

10.2.2.10. Mv-Expand

Expand an array or object within an event into multiple events, where each of the generated events contains a single value of the original array or object. If you're curious, mv stands for multiple value.

Syntax

- `Scope | mv-expand [ bagexpansion=ExpansionMode ] [ with_itemindex=IndexFieldName ] FieldName [, FieldName... ] [ limit Rowlimit ]`
- `Scope | mv-expand [ bagexpansion=ExpansionMode ] [ Name=NewName ] ArrayExpression [, [ Name=NewName ] ArrayExpression... ] [ limit Rowlimit ]`

Arguments

Name	Type	Required	Description
ExpansionMode	string	No	`bag` (default) expands into single-entry property bags `array` expands into two-element `[key,value]` array structures, allowing uniform access to keys and values.
FieldName, **ArrayExpression**	string	Yes	A field reference, or a scalar expression with a value of type `dynamic` that holds an array or a property bag. The individual top-level elements of the array or property bag get expanded into multiple events. When **ArrayExpression** is used and **NewName** doesn't equal any input field name, the expanded value is extended into a new field in the output. Otherwise, the existing **FieldName** is replaced. Wildcards are not supported.
IndexFieldName	string	No	If `with_itemindex` is specified, the output includes another field named **IndexFieldName** that contains the index starting at 0 of the item in the original expanded collection.
NewName	string	No	A name for the new field.
RowLimit	int	No	The maximum number of rows generated from each original row. The default is `9007199254740991`.

Returns

For each event in the input, the operator returns zero, one, or many events in the output, as determined in the following way:

1. Input fields that aren't expanded appear in the output with their original value. If a single input event is expanded into multiple output events, the value is duplicated to all events.
2. For each **FieldName** or **ArrayExpression** that is expanded, the number of output events is determined for each value based on the **ExpansionMode**. For each input event, the maximum number of output events is calculated. All arrays or property bags are expanded in parallel. so that missing values (if any) are replaced by null values. Elements are expanded into rows in the order that they appear in the original array/bag.
3. If the dynamic value is null, then a single event is produced for that value (null). If the dynamic value is an empty array or property bag, no event is produced for that value. Otherwise, as many events are produced as there are elements in the dynamic value.



The expanded fields are of type `dynamic`.

Examples

Single field - expansion

```
dataset=myDataset
| mv-expand b
```

Copy

Single field - bag expansion to key-value pairs

A simple bag expansion to key-value pairs:

```
dataset=myDataset
| mv-expand bagexpansion=array b
| extend key = b[0], val=b[1]
```

Copy

Cartesian product of two fields

If you want to get a Cartesian product of expanding two fields, expand one after the other:

```
dataset=myDataset
| mv-expand b
| mv-expand c
```

Copy

Using with_itemindex

Expansion of an array with `with_itemindex`:

```
| extend x=dynamic([17, 3]), y=dynamic({'foo': 'bar', 'cribl': 'search'})
| mv-expand bagexpansion=array with_itemindex=i x, z=y limit 3
```

Copy

Filter a nested array

Expand a nested array and filter by its values:

```
dataset="myDataset"
| mv-expand events
| mv-expand events.parameters
| where events.parameters.name = 'suspicious' and events.parameters.boolValue == true
```

Copy

Create an event for each of the values of field `method`. You should see 200 events total.

```
dataset=$vt_dummy event<100
| extend method=split("GET,POST", ",")
| mv-expand method
```

Copy Run in Cribl Search

10.2.2.11. Mv-Pull

Pull key-value pairs from array objects into a top-level event, or into a dedicated object/bag. If you're curious, mv stands for multiple value.

Syntax

```
  <scope> | mv-pull [key=<nameOfKeyFieldInData>] [value=<nameOfValueFieldInData>]
[delete_original=<boolean>] <nameOrPathOfFieldToArray> [as <targetFieldName>]
```

Arguments

Name	Type	Required	Description
delete_original	boolean	No	If set to `true`, deletes the source array in the event after pulling up all key-value pairs. With the `false` default, the source array is unchanged.
nameOfKeyFieldInData	string	No	Name (or column path) of the key in each array element. Defaults to literal `Name`.
nameOfValueFieldInData	string	No	Name (or column path) of the value in each array element. Defaults to literal `Value`.
nameOrPathOfFieldToArray	string	Yes	Name (or column path) of the field that contains the array of objects from which to pull key-value pairs.
targetFieldName	string	No	Name or column path of an object/bag to contain the pulled fields. Specify this to group all pulled fields into one object. If omitted, the output will be separate fields at the event's root level.

Returns

If **targetFieldName** is specified, returns a corresponding object/bag containing all the extracted key-value pairs as fields. Otherwise, returns the extracted key-value pairs as separate top-level fields.

Examples

This example pulls a simple array's elements into a single event. Uncomment the indicated line to display the elements within that event:

```
print eventData=dynamic( { "eventArray": [ { "Name": "TotalBytes", "Value": 536 }, {
| mv-pull eventData.eventArray
//| project-away eventData // uncomment this line to see the results by themselves
| extend  _time=now()
| render event
```

Copy | Run in Cribl Search

A query with this argument would pull key-value pairs from an array named `eventData` into separate top-level fields in the target event:

```
... | mv-pull eventData | ...
```

Copy

The output might take the following form (with fictitious data):

```
{
  _time: 1724241414.121,
  dataset: "metrics-store",
  eventData: [
    { Name: "numBytesRead", Value: 1234 },
    { Name: "numBytesWrite", Value: 200 },
    { Name: "numRequests", Value: 42 },
  ],
  numBytesRead: 1234,
  numBytesWrite: 200,
  numRequests: 42,
}
```

Copy

Here is the same query with additional arguments, targeting an array whose key-value pairs have unusual naming:

```
... | mv-pull key="n" value="v" delete_original=true otherEventData as itsMyBag ...
```

Copy

The output would instead have this format:

```
{
  _time: 1724241414.121,
  dataset: "metrics-store",
  itsMyBag: { numBytesRead: 1234, numBytesWrite: 200, numRequests: 42 }
}
```

Copy

10.2.2.12. Pivot

The `pivot` operator turns each unique field **value** into a field **name** (rendered as a column header).

Syntax

```
Scope | pivot DataFields over ColumnFields [by LabelField]
```

Arguments

- **Scope**: The events to search.
- **DataFields**: One or more fields to populate the cells of the output table. Wildcards are not supported for field names.
- **ColumnFields**: One or more fields whose unique **values** become column headers (field **names**).
- **LabelField**: An additional field added to the left-most column of the output table, used to label the rows (events). If not specified, defaults to `_time`.



You can specify nested fields by using dot notation (`foo.bar`).

Results

For each unique **value(s)** combination of the **ColumnFields**, `pivot` returns a column whose:

- Column header (field name) is taken from that **ColumnFields** value(s).
- Column cells (field values) are populated by the **DataFields**, calculated separately for each returned event.

Examples

Turn each unique value of the `colLabel` field (`Red`, `Blue`, and `Green`) into a column, and populate the columns' cells with random 1-100 numbers:

```
dataset=$vt_dummy event < 10
| extend dataField = floor(rand() * 100),
         colLabel = case(event % 3 == 0, 'Red', event % 3 == 1, 'Blue', 'Green')
| pivot dataField over colLabel by event
```

Copy Run in Cribl Search

Perform a pivot on two fields, `colLabel1` and `colLabel2`:

```
dataset=$vt_dummy event < 10
| extend dataField1 = floor(rand() * 100),
         dataField2 = 200 - dataField1,
         colLabel1 = case(event % 3 == 0, 'Red', event % 3 == 1, 'Blue', 'Green'),
         colLabel2 = iif(event % 2 == 0, 'yes', 'no')
| pivot dataField1, dataField2 over colLabel1, colLabel2 by event
```

Copy Run in Cribl Search

10.2.2.13. SEND

The `send` operator forwards search results to Cribl Stream, or to compatible endpoints.

Usage

By default, `send` forwards search results to a Cribl HTTP Source in a distributed Cribl Stream deployment. There, you can transform, route, and relay the results in multiple ways.

When you target a Cribl-managed Stream Worker Group within the same Workspace over Cribl HTTP, you incur no additional ingest charges. You consume credits only for the CPU time that `send` uses in Cribl Search.



The `send` operator can't target Cribl-managed Worker Groups in a Workspace that has enabled an Access Control List (ACL).

Permissions

Search Editors and Admins can send events to any Cribl-managed Stream Worker Group in the same Organization.

Search Admins can also send events to any endpoint that accepts a POST request and NDJSON content. (Ingest charges might apply outside the Cribl HTTP scenario outlined above.)

Here's a full breakdown of where different Search Member Types can send events:

Search Member Type	Default Stream Worker Group	Named Stream Worker Group	Custom URL
Search User			
Search Editor			
Search Admin			
Organization Admin			
Organization Owner			

Rules

- Aggregate results are sent once the search completes, so you won't see results until then.

- By default, events are sent to the Cribl HTTP Source in the same Cribl.Cloud Organization as your Cribl Search instance.
- Also use a Cribl HTTP Source to receive data on hybrid Worker Groups.
- Use the `centralize` operator when sending results from Edge Worker Nodes that lack access to the internet or to your Cribl.Cloud Organization.
- The `group` parameter is only for Cribl-managed Worker Groups in Cribl.Cloud, and cannot be used together with the `URL` argument.
- If any events are dropped, the search logs display an appropriate error message.
- The `send` operator sets the `Content-Type: application/x-ndjson` header on POST requests.

Syntax

```
... | send [ tee=TeeBoolean ] [ group=WorkerGroup | "URL" ]
```

Arguments

- **TeeBoolean**: Defaults to `false`, where search results are not shown in Cribl Search. Instead, you get an event with the URL, the status, and the number of bytes and events sent or dropped. When set to `true`, the search results are displayed, and no stats are provided. For example, `tee=true`.
- **WorkerGroup**: The Cribl Stream Worker Group to send data to. The default value is literally `default`. For example, `group=default`.
- **URL**: Your domain for inbound data, with the appropriate port. Defaults to your Cribl.Cloud ingest address and port `10200`, the default port for the Cribl HTTP Source. The general format is: `https://<groupName>.<workspaceName>.<organizationId>.cribl.cloud:10200`

  An example with the most typical Worker Group and Workspace names, and with a fictitious Organization name: `https://default.main.goat-farm.cribl.cloud:10200`



To check the ingest address for Cribl-managed Groups: From your Organization's top bar, select **Products** > **Cribl**. From the resulting sidebar, select **Data Sources**, then select the target **Group**. Under **Sources Enabled by Default**, find the `in_cribl_http` entry.

For a hybrid Group running on a host that you manage, the ingest address will typically be configured through your load balancer.

Examples

- Send events to the `default` Worker Group:

```
dataset=myDataset
| send
```

Copy

- Send up to 100 events and display the results:

```
dataset=myDataset
| limit 100
| send tee=true
```

Copy

- Send events to a Worker Group named Goat:

```
dataset=myDataset
| send group=goat
```

Copy

- Send events to a hybrid Worker Group:

```
dataset=myDataset
| send "https://in.your-tenant.com:10200"
```

Copy

- Send aggregate results by specifying the URL to your Cribl Stream internal Cribl HTTP Source:

```
dataset=myDataset
| summarize count() by action
| send "https://in.main-default-domain.cribl.cloud:10200"
```

Copy

- Send results to the Cribl.Cloud Stream default Worker Group:

```
dataset=$vt_dummy event<10
| extend _raw=iif(event%2>0, "This is a test event", "This is another event")
| send
```

Copy Run in Cribl Search

10.2.2.14. Top-Hitters

The `top-hitters` operator counts distinct value combinations and returns the most frequent combinations in descending order. It provides valuable insights into the most popular and impactful data combinations within your input Dataset.

This operator is conceptually equivalent to using the `summarize` operator with `count` and `top`:

```
Scope | summarize C=count() by ValueExpression | top NumberOfValues by C desc
```

Syntax

```
Scope | top-hitters NumberOfValues of ValueExpression [ by SummingExpression ]
```

Arguments

- **Scope**: The events to search.
- **NumberOfValues**: The number of distinct values of **ValueExpression**.
- **ValueExpression**: An expression or list of fields that operate over the **Scope** whose distinct values are returned. Supports single expressions like `x / 42` or `abs(status)` as well as more complex expressions such as `top-hitters 10 of method, status [by SummingExpression]`.
- **SummingExpression**: If specified, a numeric expression over the **Scope** whose sum per distinct value of **ValueExpression** establishes which values to emit. If not specified, the count of each distinct value of **ValueExpression** is used instead.

Examples

- Return the top 3 distinct `status` values.

```
dataset="cribl_search_sample" status="*"
| limit 1000
| top-hitters 3 of status
```

Copy | Run in Cribl Search

- Sum `bytes` and returns the top 2 `host` values.

```
dataset="cribl_search_sample" status="*" host="*"
| limit 1000
| top-hitters 3 of host by bytes
```

Copy | Run in Cribl Search

- Sum `bytes`, and return the top 10 distinct combinations of `clientip` and `useragent` values based on the `bytes` field.

```
dataset="cribl_search_sample" clientip="*" useragent="*"
| limit 1000
| top-hitters 5 of clientip, useragent by bytes
```

Copy | Run in Cribl Search

10.2.2.15. Union

The `union` operator appends one set of results to another.

Results are automatically sorted by time in ascending order if a time field is present. Otherwise, the order will be random.

Syntax

Using a `let` statement:

```
let SubqueryName = Subquery;

MainQuery | union SubqueryName
```

Copy

Using an inline subquery:

```
MainQuery | union (
  Subquery
  )
```

Copy

Arguments

- **SubqueryName**: The name for the **Subquery** expression. Spaces ( ) are not allowed.
- **Subquery**: The data to append to the **MainQuery**.
- **MainQuery**: The data to which the results of **Subquery** are appended.

Rules

The union operation includes the first 50,000 events of **SubqueryName**. The remaining events are ignored.

Examples

Combine two sets of data.

```
let dataset1 = range x from 1 to 5 step 1 | extend dataField = x * 2;
range y from 6 to 10 step 1 | extend dataField = y * 3 | union dataset1;
```

Copy | Run in Cribl Search

Extend the results of three searches on the `$vt_dummy` Dataset, each filtered with `event < 10`, and combine them using the `union` operator.

```
let stage1 = search in($vt_dummy) event<10 | extend foo=42;
let stage2 = search in($vt_dummy) event<10 | extend bar=24;
search in($vt_dummy) event<10 | extend baz=84 | union stage1, stage2;
```

Copy | Run in Cribl Search

Append data, using an inline subquery.

```
// main query
print x = 1 | union (
  // inline subquery
  print y = 1 | extend y = 2
  )
```

Copy

10.2.3. Display Operators

A list of display operators supported by Cribl Search.

Display operators control how data is presented in the output, allowing formatting, sorting, or visualizing results for better readability.

Name	Description
`limit` (`take`)	Limits the number of events.
`order` (`sort`)	Arranges events into order by one or more fields.
`print`	Outputs expression results.
`project`	Keeps only the fields specified, and can also rename fields and insert new computed fields.
`project-away`	Excludes specific fields from the results..
`range`	Generates a series of events.
`render`	Enforces a specific visualization of the search results.
`top`	Returns the first N events sorted by the specified fields.

10.2.3.1. Limit

The `limit` operator returns up to the specified number of events (or fewer, if fewer events match the query). You can use this operator to size your result sets, and to control resource usage.

Alias: `take` (`limit` and `take` are synonyms).

Syntax

```
Scope | limit Integer
```

Arguments

- **Scope**: The events to search.
- **Integer**: A positive integer.

Usage

Bear in mind the following execution details:

- `limit` doesn't guarantee any consistency in its results when executing multiple times, even if the Dataset hasn't changed.
- In interactive searches (from the query box), query execution with `limit` displays previews of the results. These previews reflect only a sample of events, up to the `limit` threshold.
- Scheduled and programmatic searches omit these previews.
- In Lakehouse searches, `limit` constrains the number of output rows, but the query continues iterating over the whole population of events. Here, Cribl Search will true up aggregations as the query completes.
- Outside of Lakehouse searches, the `limit` operator's argument sets a hard threshold. Query execution will end when Cribl Search finds a matching number of events, or can be canceled during a preview. So aggregations will be accurate for the `limit` sample, but not necessarily for the overall population of events.

Example

```
dataset=myDataset
| limit 1000
```

Copy

Return only 42 events.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| limit 42
```

Copy Run in Cribl Search

10.2.3.2. Order

The `order` operator arranges events into order by one or more fields.

Alias: `sort` (`sort` and `order` are synonyms.)

Syntax

```
Scope | order [ topN=MaxNoOfOutputEvents ] [ maxEvents=MaxNoOfInputEvents ] by Field [ asc | desc ] [ nulls first | nulls last ] [, ...]
```

Arguments

Name	Type	Required	Description
Scope	String	Yes	The events to search.
MaxNoOfOutputEvents	Int	No	Maximum number of events to produce. The output is limited to `10000` events, so entering a higher **MaxNoOfOutputEvents** value effectively sets a limit of `10000`.
MaxNoOfInputEvents	Int	No	Maximum number of events to handle and arrange. Usually, this value is already determined by the `limit` operator used earlier in the query, but you can also set it explicitly here.
Field	String	Yes	Field to sort by. The type of the field values must be numeric, date, time, or string.
`asc` or `desc`	String	No	`asc` sorts into ascending order, low to high. Default is `desc`, high to low. For more details, see Sorting Rules.
`nulls first` or `nulls last`	String	No	`nulls first` will place the null values at the beginning and `nulls last` will place the null values at the end. Default for `asc` is `nulls first`. Default for `desc` is `nulls last`.

Sorting Rules

- Numeric values appear before other data types. An exception to that may be `null`, whose behavior depends on the `nulls first/nulls last` setting above.

- Numeric strings are converted to numbers when sorted. For example, “`100`” and “`5`” are compared as `100` and `5`.
- By default: for ascending order, `null`s appear first, and for descending order, `null`s appear last. You can change this with the `nulls first/nulls last` setting above.

Example

All events with a specific `ClientRequestId`, ordered by their `Timestamp`.

```
dataset=myDataset
| where ClientRequestId == "5a848f70-9996-eb17-15ed-21b8eb94bf0e"
| order by Timestamp asc
```

Copy

Order results by the field `event` in descending order.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| order by event desc
```

Copy Run in Cribl Search

10.2.3.3. Print

The `print` operator evaluates one or more scalar expressions, and returns the results as a single row.

Use `print` to verify an expression before you include it in a larger query.



To produce sample events, use the `$vt_dummy` virtual table.

Syntax

```
print [FieldName =] Expression [, ...]
```

Arguments

- **FieldName**: The name to assign to the output field. If omitted, the name is generated from the corresponding **Expression**.
- **Expression**: The scalar expression to evaluate. A scalar expression is any combination of values and operators that evaluates to a single value. For example, `1 + 2` is a scalar expression that evaluates to `3`.

Results

Returns a single row with as many columns as there are **Expression** arguments.

Examples

- Check a calculation result.

```
// returns 6
print 1 + 2 + 3
```

Copy

- Check a string concatenation.

```
// returns "Hello, World!"
print strcat("Hello", ", ", "World!")
```

Copy

- Check multiple expressions, displaying the results as a single event.

```
// returns 200, "56", and 1692736230.271
print ["Check result"] = iif(false, 100 , 200), Check = substring("456", 2), ago
```

Copy

- Check an expression, displaying the results as multiple events.

```
// returns two events with "Word" and "Index" fields:
// Word: "foo", Index: 0
// Word: "bar", Index: 1
print Word=split("foo bar", " ")
| mv-expand with_itemindex=Index Word
```

Copy

- Print a single event with field `foo` set to 42.

```
print foo=42
```

Copy Run in Cribl Search

10.2.3.4. Project

The `project` operator keeps only the fields specified, and can also rename fields and insert new computed fields.

The order of the arguments determines the order of the fields in the result.

See also `extend` and `print`.



Unlike `project-rename`, `project` drops all fields that were not specified.

Syntax

- `Scope | project FieldName [= Expression] [, ...]`
- `Scope | project [FieldName | (FieldName[,]) =] Expression [, ...]`

Arguments

- **Scope**: The events to search.
- **FieldName**: Optional if there's an **Expression**. The name of the field to appear in the output. If omitted, the name will be generated.
    - If there is no **Expression**, then **FieldName** is mandatory and a field of that name must appear in the input.
    - If **Expression** returns more than one field, a list of field names can be specified in parentheses. In this case **Expression**'s output fields will be given the specified names, dropping all the rest of the output fields, if there are any. If a list of the field names is not specified, all **Expression**'s output fields with generated names will be added to the output.
- **Expression**: Optional if there's a **FieldName**. A scalar expression referencing the input fields.



You can return a new calculated field with the same name as an existing field in the input.

Results

Returns a table that has the fields named as arguments, and as many rows as the input table.

Examples

Show only two fields, `cost` and `price`.

```
dataset=myDataset
| project cost=price*quantity, price
```

Copy

The following example shows several kinds of manipulations that can be done using the `project` operator. The Dataset has three fields of type `int`: `A`, `B`, and `C`.

```
dataset=myDataset
| project
    X=C,                        // Rename field C to X
    A=2*B,                      // Calculate a new field A from the old B
    C=strcat("-",tostring(C)), // Calculate a new field C from the old C
    B=2*B                       // Calculate a new field B from the old B
```

Copy

You can also use `project` with nested data, using dot notation. In the following example, two new fields (`user` and `query`) are created based on an existing nested field (`stats`).

```
dataset="cribl_internal_logs" message="search finished"
| project _time, user=stats.userDisplayName, query=stats.query // show only the `_tim
```

Copy Run in Cribl Search

Show only `_time`, `event`, and `parity` fields.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| project _time, event, parity
```

Copy Run in Cribl Search

Show `_time`, `event`, and `parity` fields and evaluate two new fields, `foo` and `bar`.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| project _time, event, parity, foo=42, bar=foo*2
```

Copy Run in Cribl Search

10.2.3.5. Project-Away

The `project-away` operator excludes specific fields from the results.

Syntax

```
Scope | project-away ColumnNameOrPattern [, ...]
```

Arguments

- **Scope**: The events to search.
- **ColumnNameOrPattern**: The name of the field or field wildcard-pattern to be removed from the results.
  - Wildcards `*` represent zero or more characters in a field name. For example, `Sta*`.
  - Wildcards are not supported in the middle of a field name and consecutive wildcards are not allowed. For example, `St*us` and `S**s` are invalid.

Results

A table with fields that were not named as arguments. Contains the same number of rows from the data piped to the operation.

Example

```
dataset=myDataset
| project-away price, quantity, zz*
```

Copy

Exclude all fields that start with the letter "p".

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| project-away p*
```

Copy Run in Cribl Search

10.2.3.6. Project-Rename

The `project-rename` operator renames fields.



Unlike `project`, `project-rename` keeps all fields that were not specified.

Syntax

```
Scope | project-rename NewFieldName = ExistingFieldName [, ...]
```

Arguments

- **Scope**: The events to search.
- **NewFieldName**: The new field name.
- **ExistingFieldName**: The name of the existing field to rename.

Results

A table that has the fields in the same order as in an existing table, with its fields renamed.

Example

```
dataset=myDataset
| project-rename new_b=b, new_a=a
```

Copy

Rename the field `parity` to `oddEven`.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| project-rename oddEven=parity
```

Copy Run in Cribl Search

10.2.3.7. RANGE

The `range` operator generates a series of events, each containing a generated numeric field and a `_time` field. You can use this operator as syntactic sugar on the `$vt_dummy` virtual table, to generate a defined range of events with equally spaced values.



This is a data-generating operator, not a data-consuming operator.

Syntax

```
range <FieldName> from <Start> to <Stop> [step <Step>]
```

Arguments

Name	Type	Required	Description
FieldName	String	Yes	The name of the single column in the output table.
Start	Int, Long, Real, Datetime, or Timespan	Yes	The smallest value to generate.
Stop	Int, Long, Real, Datetime, or Timespan	Yes	The highest value to generate; or, if **Step** would generate a value exceeding this parameter, a bound on the highest value.
Step	Int, Long, Real, Datetime, or Timespan	No	The difference between two consecutive values in the output.

Results

A series of events, each containing a field called `<FieldName>`, whose values are: `<Start>`, `<Start>` + `<Step>`, ... up to and including `<Stop>`.

Examples

Generate 10 equally spaced events, numbered `1` to `10`:

```
range theField from 1 to 10
```

Copy Run in Cribl Search

Generate five equally spaced events, numbered 1 to 9:

```
range theField from 1 to 10 step 2
```

Copy Run in Cribl Search

Generate a series of hourly events, starting three days back:

```
range theField from ago(3d) to ago(2d) step 1h | project strftime(theField,'%Y-%m-%dT
```

Copy Run in Cribl Search

Generate a week's worth of daily events:

```
range _time from ago(7d) to now() step 1d
```

Copy

Generate 10 event values from 10 to 100, incremented by 10:

```
range myField from 10 to 100 step 10
```

Copy

10.2.3.8. Render

The `render` operator enforces a specific visualization of the search results. It's useful when you want to override the default rendering of a query.

The `render` operator supports the following modes:

- `event` – renders the results as a list of events, under the **Events** tab.
- `table` – renders the results as a table, under the **Chart** tab.

Syntax

```
Scope | render RenderMode
```

Arguments

- **Scope**: The events to search.
- **RenderMode**: The rendering mode to use. Can be `event` or `table`.

Examples

```
dataset=myDataset
| render table
```

Copy

Render a table of enumerated fields.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| project event, parity
| render table
```

Copy Run in Cribl Search

10.2.3.9. Sort

The `sort` operator arranges events into order by one or more fields.

Alias: `order` (`order` and `sort` are synonyms.)

Syntax

```
Scope | sort [ topN=MaxNoOfOutputEvents ] [ maxEvents=MaxNoOfInputEvents ] by Field [ asc | desc ] [ nulls first | nulls last ] [, ...]
```

Arguments

Name	Type	Required	Description
Scope	String	Yes	The events to search.
MaxNoOfOutputEvents	Int	No	Maximum number of events to produce. The output is limited to `10000` events, so entering a higher **MaxNoOfOutputEvents** value effectively sets a limit of `10000`.
MaxNoOfInputEvents	Int	No	Maximum number of events to handle and arrange. Usually, this value is already determined by the `limit` operator used earlier in the query, but you can also set it explicitly here.
Field	String	Yes	Field to sort by. The type of the field values must be numeric, date, time, or string.
`asc` or `desc`	String	No	`asc` sorts into ascending order, low to high. Default is `desc`, high to low. For more details, see Sorting Rules.
`nulls first` or `nulls last`	String	No	`nulls first` will place the null values at the beginning and `nulls last` will place the null values at the end. Default for `asc` is `nulls first`. Default for `desc` is `nulls last`.

Sorting Rules

- Numeric values appear before other data types. An exception to that may be `null`, whose behavior depends on the `nulls first/nulls last` setting above.

- Numeric strings are converted to numbers when sorted. For example, “`100`” and “`5`” are compared as `100` and `5`.
- By default: for ascending order, `null`s appear first, and for descending order, `null`s appear last. You can change this with the `nulls first/nulls last` setting above.

Example

All events with a specific `ClientRequestId`, sorted by their `Timestamp`.

```
dataset=myDataset
| where ClientRequestId == "5a848f70-9996-eb17-15ed-21b8eb94bf0e"
| sort by Timestamp asc
```

Copy

Sort results by the field `event` in descending order.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| order by event desc
```

Copy Run in Cribl Search

10.2.3.10. Take

The `take` operator returns up to the specified number of events.

Alias: `limit` (`limit` and `take` are synonyms.)



`take` doesn't guarantee any consistency in its results when executing multiple times, even if the data set hasn't changed.

Syntax

```
Scope | take Integer
```

Arguments

- **Scope**: The events to search.
- **Integer**: A positive integer.

Example

```
dataset=myDataset
| take 1000
```

Copy

Return only 42 events.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| take 42
```

Copy Run in Cribl Search

10.2.3.11. Top

The `top` operator returns the first N events sorted by the specified fields.



`top 5 by name` is equivalent to the expression `sort by name | take 5` both from semantic and performance perspectives.

When sorting different data types, numeric values always appear first.

Syntax

```
Scope | top NumberOfRows by Expression [ asc | desc ] [ nulls first | nulls last ]
```

Arguments

Name	Type	Required	Description
Scope	Tabular	Yes	The events to search.
NumberOfRows	Int	Yes	The number of rows to return.
Expression	String	Yes	Scalar expression by which to sort. The type of the values must be numeric, date, time or string.
`asc` or `desc`	String	No	Controls whether the selection is from the “bottom” or “top” of the range. Default desc. Numeric values always appear before other data types.
`nulls first` or `nulls last`	String	No	Controls whether null values appear at the “bottom” or “top” of the range. Default for `asc` is `nulls first`. Default for `desc` is `nulls last`.

Examples

Show the top 3 events with the most errors.

```
dataset=myDataset
| top 3 by errorCount
```

Copy

Return the top 42 results sorted by the `event` field in descending order.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| top 42 by event desc
```

Copy | Run in Cribl Search

10.2.4. Filter Operators

A list of filter operators supported by Cribl Search.

Filter operators exclude or include events based on specified conditions, allowing you to narrow down the dataset to relevant records.

Name	Description
`dedup`	Filters out duplicate events.
`distinct`	Finds unique field values.
`search`	Finds events that contain the specified text.
`where`	Filters events based on the specified predicate.

10.2.4.1. Dedup

The `dedup` operator filters out duplicate events.

You need to specify the field that you want to inspect for duplicates (**FieldName**). When that field has the same value in multiple events, those events are considered duplicates.

You can also specify multiple fields. In that case, events are considered duplicates only if **all** of the specified fields are duplicated. For example, if you specify fields named `Phone` and `Email`, events are considered duplicates only if they have an identical `Phone` value **and** an identical `Email` value.

By default, the `dedup` operator outputs only the first duplicate it finds, and drops the rest. You can change this behavior by specifying the number of duplicates to keep (**NumberOfDuplicatesToKeep**).

When looking for duplicates, the `dedup` operator compares events timed within 30 seconds of each other. You can also specify a different time window (**TimeWindow**).

Syntax

```
Scope | dedup [time_window=TimeWindow] [num_duplicates=NumberOfDuplicatesToKeep] by FieldName [, ...]
```

Arguments

- **TimeWindow** (int): Sets a time interval, across event timestamps, over which this operator will apply the **NumberOfDuplicatesToKeep** limit on the specified **FieldName**(s). For event pairs outside this time interval, `dedup` will release the limit, allowing the next duplicate event to filter through to results. Default: `30` seconds.
- **NumberOfDuplicatesToKeep** (int or expression): The number of duplicate events to keep. Default: `1`.
- **FieldName**: The name of the field that you want to inspect for duplicates. Allowed formats: `fieldName` or `["field name"]`. Separate multiple fields with a comma: `fieldName1, ["field name 2"], ...`.

Results

Filters out events identified as duplicates.

Examples

Filter out events that contain duplicate instances of a random number:

```
dataset=$vt_dummy event<1000
| extend randomNumber=rand(10)
| dedup by randomNumber
```

Copy | Run in Cribl Search

Filter out events that have the same value in the `Name` field.

```
dataset=myDataset
| dedup by Name
```

Copy

Filter out events that have the same values in the corresponding `Name`, `Home address`, and `Work address` fields.

```
dataset=myDataset
| dedup by Name, ['Home address'], ["Work address"]
```

Copy

Filter out events that contain `Name` duplicates, and that were found within a minute of each other. If there are more than 5 such events, keep only the first 5.

```
dataset=myDataset
| dedup time_window=60 num_duplicates=5 by Name
```

Copy

10.2.4.2. Distinct

The `distinct` operator identifies unique values within each of the specified fields. It's used to retrieve distinct values for each field independently. `distinct` finds unique values within each provided field separately, helping you to analyze and explore data uniqueness within individual fields.

Syntax

```
Scope | distinct [ MaxCombinations ] [ MaxDepth ] FieldName [, ...]
```

Arguments

- **Scope**: The events to search.
- **MaxCombinations**: The maximum number of distinct combinations. Defaults to `10000`.
- **MaxDepth**: The maximum depth for searching unique combinations of field values. This controls how deep the search goes when identifying distinct combinations within nested or hierarchical data structures. Defaults to `15`.
- **FieldName**: One or more fields you want to find distinct values for. This operator allows you to identify unique values within the specified field(s) from your Dataset. A wildcard `*` indicates all available fields.

Examples

Find the distinct values for all fields in your Dataset.

```
dataset=myDataset
| distinct *
```

Copy

Find unique values within the `host` and `port` fields.

```
dataset=myDataset
| distinct host, port
```

Copy

```
dataset=$vt_dummy event<1000
| extend randomNumber=rand(10)
| distinct randomNumber
```

Copy Run in Cribl Search

10.2.4.3. Search

The `search` operator finds events with specific text strings.



We recommend using the `cribl` operator, which has easier syntax and more capabilities.

Syntax

```
[Scope |] search [in (Dataset, ...)] [kind=CaseSensitivity] Predicate
```

Arguments

Scope: An optional expression that acts as a data source to be searched over. Cannot appear together with the optional phrase that includes **FieldName**.

CaseSensitivity: An optional flag that controls the behavior of all string scalar operators with respect to case sensitivity. Valid values are the two synonyms `default` and `case_insensitive` (which is the default for operators such as `has`, namely being case-insensitive) and `case_sensitive` (which forces all such operators into case-sensitive matching mode).

FieldName: An optional comma-separated list of "wildcarded" field names to take part in the search.

Predicate: An expression that defines what to search for and is evaluated for every record in the input. The expression must return a `bool` value. If the expression returns `true`, the record is outputted. The syntax for **Predicate** extends and modifies the normal syntax for expressions:

- **String matching extensions**: String literals that appear as terms in the **Predicate** indicate a term match between all fields and the literal using `has`, `hasprefix`, `hassuffix`, and the inverted `!` or case-sensitive `cs` versions of these operators. Asterisks are not supported within the literal, nor as wildcard characters.
- **Field restriction**: By default, string matching extensions attempt to match against all fields of the Dataset. It is possible to restrict this matching to a particular field by using the following syntax: `FieldName:StringLiteral`.
- **String equality**: Exact matches of a field against a string value (instead of a term-match) can be done using the syntax `FieldName==StringLiteral`.
- **Other expressions**: The syntax supports all expressions that return a `bool` value. For example, `"error" and x==123` means: search for records that have the term `error` in any of their fields, and have the value `123` in the `x` field."

- **Regex match**: Indicate regular expression matching using the syntax `FieldName matches regex RegularExpression`. For syntax details, see Regex Examples, Regex Flags, and Disambiguate Regex Characters.

String Matching

#	Syntax	Meaning (equivalent `where`)
1	`search "err"`	`where * has "err"`
2	`search in (T1,T2) "err"`	`union T1,T2 | where * has “err”`
3	`search col:"err"`	`where col has "err"`
4	`search col=="err"`	`where col=="err"`
10	`search col matches regex "..."`	`where col matches regex "..."`
11	search kind=case_sensitive	All string comparisons are case-sensitive
12	`search "abc" and ("def" or "hij")`	`where * has "abc" and (* has "def" or * has hij")`
13	`search "err" or (A>a and A<b)`	`where * has "err" or (A>a and A<b)`

Examples

Simple term search over all Datasets and fields:

```
search "goats"
```

Copy

Looking only for records that match both terms:

```
search "goats" and ("Billy" or "Nanny")
```

Copy

Performing a case-sensitive match of all terms:

```
search kind=case_sensitive "BillB" and ("SteveB" or "SatyaN")
```

Copy

Restricting the match to certain fields:

```
search CEO:"goats" or CSA:"goats"
```

Copy

Specific time limit:

```
search "goats" and Timestamp >= datetime(1981-01-01)
```

Copy

Return results that match “test event”:

```
dataset=$vt_dummy event<10
| extend _raw=iif(event%2>0, "This is a test event", "This is another event")
| search _raw has "test event"
```

Copy Run in Cribl Search

In all the above examples, `search` behaves as a filtering operator. You can use `search` as a data-**producing** operator in simple expressions of the form `search in (<Dataset>)`:

```
search in (cribl_search_sample) "mozilla"
```

Copy Run in Cribl Search

However, in the next example, `search` again behaves as a filtering operator. It follows this behavior whenever a `search` expression is placed after a data-producing operator (such an explicit or implict `cribl`, or even another `search` operator in the pattern shown in the preceding example). This returns results where strings in the `useragent` field match “Mozilla”:

```
datatype="apache_access_combined"
| limit 10
| search useragent in ("Mozilla")
```

Copy Run in Cribl Search

10.2.4.4. Where

The `where` operator filters events based on the specified predicate.

Syntax

```
Scope | where Predicate
```

Arguments

- **Scope**: The events to search.
- **Predicate**: An expression that returns a `bool` value for each event. This can match on strings, numeric values, or regular expressions.

Comparison and String Operators in `where`

You can construct a **Predicate** using the comparison and string operators shown below. Note the abbreviations used in these tables:

- Operators with a `_cs` suffix are case-sensitive.
- RHS = right-hand side of the predicate.
- LHS = left-hand side of the predicate.

Comparison Operators in `where`

To compare on wildcards (`*`), use the comparison operators of the `cribl` operator. The `where` operator supports wildcards only inside regex literals (`/^abc.*xyz$/`).

When comparing values of different types, Cribl Search performs automatic type conversion wherever possible, giving priority to number comparisons.

Some comparison examples:

- `"1.0" == 1` is `true`
- `"1.0" == "1"` is `true`
- `5 < 100` is `true`
- `"5" < "100"` is `true`
- `"abc" < "100"` is `false`
- `"abc" < 100` is `null`

Operator	Description	Example (Returns `true`)	Case-Sensitive	Accepts `null`	Supports Regex	Regex Example
`==`	Equals	`"aBc" == "aBc"`	Yes	Yes	Only in regex literals	`foo == /^abc.*xyz$/`
`!=`	Not equals	`"abc" != "ABC"`	Yes	Yes	Only in regex literals	`foo != /^abc.*xyz$/`
`=~`	Equals	`"abc" =~ "ABC"`	No	Yes	Only in regex literals	`foo =~ /^abc.*xyz$/`
`!~`	Not equals	`"aBc" !~ "xyz"`	No	Yes	Only in regex literals	`foo !~ /^abc.*xyz$/`
>	Greater than		–	No	–	–
<	Less than		–	No	–	–
>=	Greater than or equal to		–	No	–	–
<=	Less than or equal to		–	No	–	–
!	Unary inversion (NOT) Logical values only; string values are unsupported		–	No	–	–

String Operators in `where`

To search on wildcards (`*`), use the string operators of the `cribl` operator. The `where` operator supports wildcards only inside regex literals (`/^abc.*xyz$/`).

Operator	Description	Example (Returns `true`)	Case-Sensitive	Regex	Regex Example
`contains`	RHS occurs as a	`"FabriKam" contains "BRik"`	No	No	–

Operator	Description	Example (Returns `true`)	Case-Sensitive	Regex	Regex Example
	subsequence of LHS				
`!contains`	RHS doesn't occur in LHS	`"Fabrikam" !contains "xyz"`	No	No	–
`contains_cs`	RHS occurs as a subsequence of LHS	`"FabriKam" contains_cs "Kam"`	Yes	No	–
`!contains_cs`	RHS doesn't occur in LHS	`"Fabrikam" !contains_cs "Kam"`	Yes	No	–
`endswith`	RHS is a closing subsequence of LHS	`"Fabrikam" endswith "Kam"`	No	No	–
`!endswith`	RHS isn't a closing subsequence of LHS	`"Fabrikam" !endswith "brik"`	No	No	–
`endswith_cs`	RHS is a closing subsequence of LHS	`"Fabrikam" endswith_cs "kam"`	Yes	No	–
`!endswith_cs`	RHS isn't a closing subsequence of LHS	`"Fabrikam" !endswith_cs "brik"`	Yes	No	–
`has`	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	`"North America" has "america"`	No	No	–
`:`	Alias for `has`	`"North America" : "america"`	No	No	–
`!has`	RHS isn't a full term in LHS	`"North America" !has "amer"`	No	No	–

Operator	Description	Example (Returns `true`)	Case-Sensitive	Regex	Regex Example
has_all	Same as has but works on all of the events	"North and South America" has_all("south", "north")	No	No	–
has_any	Same as has but works on any of the events	"North America" has_any("south", "north")	No	No	–
!has_all	Same as has but works on all of the events	"North and South America" has_all("south", "north")	No	No	–
!has_any	Same as has but works on any of the events	"North America" has_any("south", "north")	No	No	–
has_cs	RHS is a whole term in LHS	"North America" has_cs "America"	Yes	No	–
!has_cs	RHS isn't a full term in LHS	"North America" !has_cs "amer"	Yes	No	–
hasprefix	RHS is a term prefix in LHS	"North America" hasprefix "ame"	No	No	–
!hasprefix	RHS isn't a term prefix in LHS	"North America" !hasprefix "mer"	No	No	–
hasprefix_cs	RHS is a term prefix in LHS	"North America" hasprefix_cs "Ame"	Yes	No	–
!hasprefix_cs	RHS isn't a term prefix in LHS	"North America" !hasprefix_cs "CA"	Yes	No	–
hassuffix	RHS is a term suffix in LHS	"North America" hassuffix "ica"	No	No	–
!hassuffix	RHS isn't a term suffix in	"North America" !hassuffix	No	No	–

Operator	Description	Example (Returns `true`)	Case-Sensitive	Regex	Regex Example
	LHS	`"americ"`			
`hassuffix_cs`	RHS is a term suffix in LHS	`"North America" hassuffix_cs "ica"`	Yes	No	–
`!hassuffix_cs`	RHS isn't a term suffix in LHS	`"North America" !hassuffix_cs "icA"`	Yes	No	–
`in`	Equal to any of the events	`"abc" in ("123", "345", "abc")`	Yes	Only in regex literals	`foo in ("ye*ah", /^no.+way/i, 'whee')`
`!in`	Not equal to any of the events	`"bca" !in ("123", "345", "abc")`	Yes	Only in regex literals	`foo !in ("ye*ah", /^no.+way/i, 'whee')`
`in~`	Equal to any of the events	`"Abc" in~ ("123", "345", "abc")`	No	Only in regex literals	`foo in~ ("ye*ah", /^no.+way/i, 'whee')`
`!in~`	Not equal to any of the events	`"bCa" !in~ ("123", "345", "ABC")`	No	Only in regex literals	`foo !in~ ("ye*ah", /^no.+way/i, 'whee')`
`matches regex`	LHS contains a match for RHS	`"Fabrikam" matches regex "b.*k"`	Yes	Only in regex literals	`foo matches regex /^some.+(body|thing)$/i`
`startswith`	RHS is an initial subsequence of LHS	`"Fabrikam" startswith "fab"`	No	No	–
`!startswith`	RHS isn't an initial subsequence of LHS	`"Fabrikam" !startswith "kam"`	No	No	–

Operator	Description	Example (Returns `true`)	Case-Sensitive	Regex	Regex Example
`startswith_cs`	RHS is an initial subsequence of LHS	`"Fabrikam" startswith_cs "Fab"`	Yes	No	–
`!startswith_cs`	RHS isn't an initial subsequence of LHS	`"Fabrikam" !startswith_cs "fab"`	Yes	No	–

Results

Returns events for which the **Predicate** evaluates to `true`.

Examples

Here are a few examples of how to use the `where` operator in queries.

Simple Comparisons First

This example retrieves events that are no older than 1 hour, come from a source called `MyCluster`, and have two fields containing the same value:

```
dataset=myDataset
| where Timestamp > ago(1h)
    and Source == "MyCluster"
    and ActivityId == SubActivityId
```

Copy

Columns Contain String

Retrieve all the events in which the word "Cribl" appears on word boundaries in `field`:

```
dataset=myDataset
| where field has "Cribl"
```

Copy

In `where` string comparisons, note that `:` is a substitute for the `has` operator. So in the above example, `where field:"Cribl"` is equivalent to `where field has "Cribl"`.

Match on AND Condition

Return results where the field `event` is greater than 20 and `parity` is odd.

```
dataset=$vt_dummy event<100
| extend parity=iif(event%2==0, 'even', 'odd')
| where event>20 and parity=='odd'
```

Copy | Run in Cribl Search

Match on OR Condition

Return results that match either of two conditions:

```
dataset="cribl_internal_logs" | limit 50 | where method == "GET" or method == "POST"
```

Copy | Run in Cribl Search

Match with Regex

The `where` operator supports regular expressions to match against strings or regex literals. You can see this in action in our Regex Examples.

Search on Non-String Values

When using `where` to search on non-`string` values (such as `int` numbers), omit the quotes (`""`, `''`). For example, instead of `dataset=$vt_dummy event<"10"`, do this:

```
dataset=$vt_dummy event<10
```

Copy | Run in Cribl Search

An exception to this is searching for high-precision decimal numbers.

10.2.5. Logical Operators

A list of logical operators supported by Cribl Search.

Operator	Description
`AND`	Returns `true` if both operands are `true`, otherwise returns `false`.
`NOT` or `!`	Returns `true` if the operand is `false`, otherwise returns `false`.
`OR`	Returns `true` if at least one operand is `true`, otherwise returns `false`.

See also:

- `null` and Binary Operators
- `bool` and the Logical NOT Operator
- `null` and the Logical NOT Operator

10.2.6. Numerical Operators

A list of numerical operators supported by Cribl Search.

Numerical operators perform arithmetic operations on values of the following types:

- `double` (aliases: `real`, `decimal`)
- `int` (alias: `long`)

You can use the following numerical operators between pairs of the above value types:

Operator	Description
==	Equal
!=	Not Equal
>	Greater Than
>=	Greater Than or Equal To
<	Less Than
<=	Less Than or Equal To
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo



Don't use the `+` operator to concatenate strings (with the `"string1" + "string2"` notation). Use the `strcat-delim` or `strcat` functions instead.

See also: `null` and Binary Operators.

10.2.7. Search Operators

A list of search operators supported by Cribl Search.

Search operators retrieve events based on defined criteria, enabling efficient data discovery.

Name	Description
`cribl`	Finds specific events. The fundamental Cribl Search operator, implicit in queries that do not specify an operator.
`find`	Finds specific events.

10.2.7.1. CRIBL

The `cribl` operator finds specific events. It is the fundamental Cribl Search operator, and is implicit in queries that do not specify an operator.

Syntax

```
[cribl] StringExpression ComparisonExpression [ BooleanOperator ] [ StringExpression | ComparisonExpression ]
```

Either a **StringExpression** or a **ComparisonExpression** is required.

Rules

- `cribl` is implicit. You don't need to specify it in a query.
- But to use `cribl` in a subquery, you must specify it.
- `cribl`, like `find`, can be used to initiate a query expression.
- `cribl` doesn't support functions (or any kind of expressions). For this, you need to use the `where` operator.
- Wildcards `*` represent zero or more characters.
- An escaped `\*` represents a literal asterisk.
- `cribl` doesn't support including both a wildcard `*` and a literal `\*` in the same query. To perform this type of search, you need to use the `where` operator.
- `cribl` interprets both the `==` and `=` tokens as comparison (not assignment) operators.
- `cribl` implicitly matches against the `_raw` field if you specify no field; otherwise, it matches against an extracted field.

Arguments

The `cribl` operator accepts a flexible range of arguments.

StringExpression

Case-insensitive string of characters. For example, `"error"` or `error`. To see if you can omit the double quotes `""`, see Quotes in `cribl` Operator. Either a **StringExpression** or a **ComparisonExpression** is required.

BooleanOperator

`not`, `or`, `and`, in that order of precedence. This order differs from other operators. Use parentheses `()` to override this precedence.

Here, whitespace is equivalent to `and`. For example, `foo and bar` is the same as `foo bar`.

ComparisonExpression

Compare numbers, strings, and/or regular expressions; perform evaluations that evaluate to a boolean. Either a **ComparisonExpression** or a **StringExpression** is required.

Syntax: `Key ComparisonOperator Value` or `Key in (ListOfValues)` or `TimeExpression`

We break down these parameters below.

Key

This is a field name.

Comparison Operators in `cribl`

Formal comparison operators are `=`, `==`, `=~`, `!=`, `!==`, `>`, `>=`, `<`, `<=`, `=~`, and `!~`. (For comparisons against regex literals, we support only the `==`, `=~`, `!=`, and `!~` operators.) See also String Operators below.

Operator	Description	Case-Sensitive	Wildcards	Regex	Examples	Wildcard Example	Regex Example

Equal `=` or `==`	Returns `true` if the operands are equal.	No	Yes	Yes	`3 == var1`
					`"3" == var1`
					`3 == '3'`
						`foo == "abc*"`	`foo == /^abc.*xyz`

Not equal `!=`	Returns `true` if the operands are not equal.	No	Yes	Yes	`var1 != 4` `var2 != "3"`	`foo != "abc*"`	`foo != /^abc.*xyz`
Greater than `>`	Returns `true` if the left operand is greater than the right operand.	No	No	No	`var2 > var1` `"12" > 2`		
Less than `<`	Returns `true` if the left operand is less than	No	No	No	`var1 < var2` `"2" < 12`		

Operator	Description	Case-Sensitive	Wildcards	Regex	Examples	Wildcard Example	Regex Example
	the right operand.						
Greater than or equal to `>=`	Returns `true` if the left operand is greater than or equal to the right operand.	No	No	No	`var2 >= var1` `var1 >= 3`		
Less than or equal to `<=`	Returns `true` if the left operand is less than or equal to the right operand.	No	No	No	`var1 <= var2` `var2 <= 5`		
!	Unary inversion (NOT) – logical values only; string values are unsupported	No	Yes	Yes			

String Operators in `cribl`

The **ComparisonOperator** parameter also supports the following string operators.

Operator	Description	Case-Sensitive	Wildcards	Regex	Example (yields `true`)	Regex Example
`in`	Equal to any of the events	Yes	No	Yes	`"abc" in ("123", "345", "abc")`	`foo in ("ye*ah", /^no.+way/i, 'whee')`
`!in`	Not equal to any of the events	Yes	No	Yes	`"bca" !in ("123", "345", "abc")`	`foo !in ("ye*ah", /^no.+way/i, 'whee')`

Operator	Description	Case-Sensitive	Wildcards	Regex	Example (yields `true`)	Regex Example
`in~`	Equal to any of the events	No	No	Yes	`"Abc" in~ ("123", "345", "abc")`	`foo in~ ("ye*ah", /^no.+way/i, 'whee')`
`!in~`	Not equal to any of the events	No	No	Yes	`"bCa" !in~ ("123", "345", "ABC")`	`foo !in~ ("ye*ah", /^no.+way/i, 'whee')`

Value

This is a **StringExpression**.

- **in**: `in` (or `!in`, or `in~`, or `!in~`) filters events with **ListOfValues**.

ListOfValues

This is a comma-separated list of one or more **StringExpression**s. For example: `(StringExpression, StringExpression, ...)`

TimeExpression

Relative or absolute time range. Syntax: `earliest=TimeString` and/or `latest=TimeString`. `earliest` is inclusive `>=`, while `latest` is exclusive `<`. The limit of `earliest` is 90 days (7776000 seconds) ago. See time syntax for details.

RegularExpression

Regex literal. Syntax: `/<regular expression>/<flags>`. For syntax details, see Regex Examples, Regex Flags, and Disambiguating Regex Characters.

ComparisonExpression Examples

- `foo=bar`, `foo23.4`, `bar42`, `foo!=bar*`
- `level in (INFO, DEBUG, ERROR)`
- `method in ("GET", "POST")`
- `earliest=-1h`, `latest=-42m`, `earliest=-42m@h latest=now`, `earliest=-2h@m latest=+2h@m`, `earliest=1700511360.123 latest=1700511420.123`
- `method == /ge*/i`
- `method =~ /ge*|p.*t/i`

- `method in ("GET", /^po.*/i)`

(In the final example above, note that your **ComparisonExpression** can mix string and regex target values.)

Quotes in `cribl` Operator

You **don't** need to enclose your StringExpression in double quotes `""`, as long as it's made of the following characters only:

- `a-z`
- `A-Z`
- `0-9`
- `$ _ * . -`

If your StringExpression contains any other character, you **do** need to enclose it in double quotes `""`, for example:

Characters	Need quotes `""`	Examples
`@ + - \ / ( ) [ ]`		`"john.doe@whatever.com"` `"/api/events"`
Space ( )		`"test event"`
Keywords `dataset`, `earliest`, and `latest`		`"earliest=testEvent"`
`a-z A-Z 0-9`		`testEvent0` or `"testEvent0"`
`$ _ * . -`		`us-*` or `"us-*"`

Cribl Search keywords, such as operator or function names, don't need double quotes. The only the exceptions are `dataset`, `earliest`, and `latest`.

To escape double quotes, use the backslash `\`, for example: `"style=\"goatee\""`

Time Syntax

The `TimeString` used for `earliest` and `latest` can be one of the following:

- A relative timestamp, expressed in the human-readable format described below. For example, `earliest=-3h`.
- An absolute timestamp, expressed in Unix time. For example, `earliest=1729290112`.

For relative timestamps, use the following syntax:

`[+|-]TimeNumberTimeUnit[@SnapToTimeUnit]`

Argument	Values Supported
`+` or `-`	Optional, defaults to `now`. Use `-` for times in the past, or `+` for times in the future.
`TimeNumber`	A positive integer.
`TimeUnit`	Unit of time. Supports `s[econds]`, `m[inutes]`, `h[ours]`, `d[ays]`, `w[eeks]`, `mon[ths]`, `q[uarters]`, `y[ears]`.
`@SnapToTimeUnit`	Optional. Append the `@` modifier, followed by any of the above `TimeUnits`, to round down to the nearest instance of that unit. (See the next section for details.)

For absolute timestamps, use the following syntax:

`TimeNumber`

Argument	Values Supported
`TimeNumber`	A Unix time value (number of seconds since 1 Jan 1970). Can be either whole seconds (for example, `1700511360`) or fractional to express milliseconds (for example, `1700511420.123`).

SnapToTimeUnit Syntax

The `@` snap modifier always rounds **down** (backwards) from any specified time. This is true even in relative time expressions with `+` (future) offsets. For example:

- `@d` snaps back to the beginning of today, 12:00 AM (midnight) UTC.
- `+128m@h` looks forward 128 minutes, then snaps back to the nearest round hour. (If you specified this in the `Latest` field, and ran the Collector at 4:20 PM, collection would end at 6:00 PM. The expression would look forward to 6:28 PM, but snap back to 6:00 PM.)

Other options:

- `@w` or `@w7` to snap back to the beginning of the week – defined here as the preceding Sunday.
- To snap back to other days of a week, use `w1` (Monday) through `w6` (Saturday).
- `@mon` to snap back to the 1st of a month.
- `@q` to snap back to the beginning of the most recent quarter – Jan. 1, Apr. 1, Jul. 1, or Oct. 1.
- `@y` to snap back to Jan. 1.

Examples

Simple term search over all events in the `default` Dataset:

```
"goats"
```

Copy

Looking only for records that match both terms:

```
"goats" and ("climb" or "rock climb")
```

Copy

Looking in the `network` field for "sector7":

```
network in ("sector7")
```

Copy

Looking in the `region` field for values that start with `us-`:

```
region in (us-*)
```

Copy

Restricting the search to certain fields:

```
disk="f9" or process="index"
```

Copy

Specific time limit:

```
earliest=-2h@h latest=-1h@min
```

Copy

Searching Datasets that start with `foo` for specific terms and field values within a specified time range:

```
dataset=foo* (blocked or "access denied") method in (POST, GET) az !in (us-east*, us-
```

Copy

Return results that match “test event”.

```
dataset=$vt_dummy event<10
| extend _raw=iif(event%2>0, "This is a test event", "This is another event")
| cribl "test event"
```

Copy Run in Cribl Search

10.2.7.2. FIND

The `find` operator finds specific events.



We recommend using the `cribl` operator, which has easier syntax and more capabilities.

Either `cribl` or `find` can be used to initiate a query expression.

Syntax

- `find [withsource=FieldName] [in (Dataset [, Dataset, ...])] where Predicate [project-smart | project FieldName [:FieldType] [, FieldName[:FieldType], ...]`
- `find Predicate [project-smart | project FieldName[:FieldType] [, FieldName[:FieldType], ...]`

Arguments

- `withsource=`**FieldName**: Optional. By default, the output will include a field called `dataset` whose values indicate which Dataset has contributed to each result. If specified, **FieldName** will be used instead of `dataset`.
- **Predicate**: An expression over the fields of the Datasets that returns a `bool` value. For a summary of some filtering functions, see the `where` operator.
- **Dataset**: Optional. By default, find will look in all the Datasets for:
  - The name of a Dataset, such as `Events`.
  - A query expression, such as `(Events | where id==42)`.
- `project-smart | project`: If not specified, `project-smart` will be used by default. For more information, see output-schema below.

Output Schema

`dataset` field

The `find` operator output will always include a `dataset` field with the Dataset name. The field can be renamed using the `withsource` parameter.

Results fields

Datasets that don't contain any fields used by the predicate evaluation will be filtered out.

When using `project-smart`, the fields that will appear in the output will be:

- Fields that appear explicitly in the predicate.
- Fields that are common to all the filtered Datasets.

The rest of the fields will be packed into a property bag and will appear in an additional `pack_` field. A field that is referenced explicitly by the predicate and has multiple types will have a different field in the result schema for each such type. Each of the field names will be constructed from the original field name and type, separated by an underscore.

When using `project FieldName[:FieldType] [, FieldName[:FieldType], ...]:

The results will include the fields specified in the list. If a Dataset doesn't contain a certain field, the values in the corresponding rows will be null. When specifying a **FieldType** with a **FieldName**, this field will have the given type, and the values will be cast to that type if needed. The casting won't have an effect on the field type when evaluating the predicate.

`pack_` field

This field will contain a property bag with the data from all the fields that don't appear in the output schema. The Dataset name will serve as the property name and the field value will serve as the property value.

Examples

Term lookup across the `default` Dataset

The query finds all events from the `default` Dataset where any field includes the word `Goat`.

```
find "Goat"
```

Copy

Term lookup across the `default` Dataset matching a name pattern

The query finds all events from the `default` Dataset whose name starts with `G`, and in which any field includes the word `Cribl`.

```
find in ("cribl_search_sample") where _raw has "GET"
| limit 100
```

Copy

Examples of find output results

The following examples show how find can be used over two Datasets: EventsTable1 and EventsTable2. Assume we have the next content of these two Datasets:

EventsTable1

Session_Id	Level	EventText	Version
acbd207d-51aa-4df7-bfa7-be70eb68f04e	Information	Some Text1	v1.0.0
acbd207d-51aa-4df7-bfa7-be70eb68f04e	Error	Some Text2	v1.0.0
28b8e46e-3c31-43cf-83cb-48921c3986fc	Error	Some Text3	v1.0.1
8f057b11-3281-45c3-a856-05ebb18a3c59	Information	Some Text4	v1.1.0

EventsTable2

Session_Id	Level	EventText	EventName
f7d5f95f-f580-4ea6-830b-5776c8d64fdd	Information	Some Other Text1	Event1
acbd207d-51aa-4df7-bfa7-be70eb68f04e	Information	Some Other Text2	Event2
acbd207d-51aa-4df7-bfa7-be70eb68f04e	Error	Some Other Text3	Event3
15eaeab5-8576-4b58-8fc6-478f75d8fee4	Error	Some Other Text4	Event4

Search in common fields, project common and uncommon fields, and pack the rest

```
find in (EventsTable1, EventsTable2)
     where Session_Id == 'acbd207d-51aa-4df7-bfa7-be70eb68f04e' and Level == 'Error'
     project EventText, Version, EventName
```

Copy

dataset	EventText	Version	EventName	pack_
EventsTable1	Some Text2	v1.0.0		{“Session_Id”:“acbd207d-51aa-4df7-bfa7-be70eb68f04e”, “Level”:“Error”}
EventsTable2	Some Other Text3		Event3	{“Session_Id”:“acbd207d-51aa-4df7-bfa7-be70eb68f04e”, “Level”:“Error”}

Search in common and uncommon fields

```
find Version == 'v1.0.0' or EventName == 'Event1' project Session_Id, EventText, Vers
```

Copy

dataset	Session_Id	EventText	Version	EventName
EventsTable1	acbd207d-51aa-4df7-bfa7-be70eb68f04e	Some Text1	v1.0.0	
EventsTable1	acbd207d-51aa-4df7-bfa7-be70eb68f04e	Some Text2	v1.0.0	
EventsTable2	f7d5f95f-f580-4ea6-830b-5776c8d64fdd	Some Other Text1		Event1

In practice, EventsTable1 rows will be filtered with Version == ‘v1.0.0’ predicate and EventsTable2 rows will be filtered with EventName == ‘Event1’ predicate.

Use abbreviated notation to search across the `default` Dataset

```
find Session_Id == 'acbd207d-51aa-4df7-bfa7-be70eb68f04e'
```

Copy

dataset	Session_Id	Level	EventText	pack_
EventsTable1	acbd207d-51aa-4df7-bfa7-be70eb68f04e	Information	Some Text1	{“Version”:“v1.0.0”}
EventsTable1	acbd207d-51aa-4df7-bfa7-be70eb68f04e	Error	Some Text2	{“Version”:“v1.0.0”}
EventsTable2	acbd207d-51aa-4df7-bfa7-be70eb68f04e	Information	Some Other Text2	{“EventName”:“Event2”}

dataset	Session_Id	Level	EventText	pack_
EventsTable2	acbd207d-51aa-4df7-bfa7-be70eb68f04e	Error	Some Other Text3	{“EventName”:“Event3”}

Return the results from each row as a property bag

```
find Session_Id == 'acbd207d-51aa-4df7-bfa7-be70eb68f04e'
```

Copy

dataset	pack_
EventsTable1	{“Session_Id”:“acbd207d-51aa-4df7-bfa7-be70eb68f04e”, “Level”:“Information”, “EventText”:“Some Text1”, “Version”:“v1.0.0”}
EventsTable1	{“Session_Id”:“acbd207d-51aa-4df7-bfa7-be70eb68f04e”, “Level”:“Error”, “EventText”:“Some Text2”, “Version”:“v1.0.0”}
EventsTable2	{“Session_Id”:“acbd207d-51aa-4df7-bfa7-be70eb68f04e”, “Level”:“Information”, “EventText”:“Some Other Text2”, “EventName”:“Event2”}
EventsTable2	{“Session_Id”:“acbd207d-51aa-4df7-bfa7-be70eb68f04e”, “Level”:“Error”, “EventText”:“Some Other Text3”, “EventName”:“Event3”}

Return results that match “test event”.

```
dataset=$vt_dummy event<10
| extend _raw=iif(event%2>0, "This is a test event", "This is another event")
| find where * has "test event"
```

Copy Run in Cribl Search

10.2.8. String Operators

A list of string operators supported by Cribl Search.

String operators manipulate and transform text, enabling actions like concatenation, trimming, replacement, or extraction.

Note that these operators' support, and their case-sensitivity, vary depending on their parent search operators and filter operators. For exact details, see these pages:

- `cribl` operator.
- `where` operator.

We use the following abbreviations in the table below:

- Operators with a `_cs` suffix are case-sensitive.
- RHS = right-hand side of the expression.
- LHS = left-hand side of the expression.

Operator	Description	Case-Sensitive	Example (yields `true`)
`==`	Equals	Yes	`"aBc" == "aBc"`
`!=`	Not equals	Yes	`"abc" != "ABC"`
`=~`	Equals	No	`"abc" =~ "ABC"`
`!~`	Not equals	No	`"aBc" !~ "xyz"`
`contains`	RHS occurs as a subsequence of LHS	No	`"FabriKam" contains "BRik"`
`!contains`	RHS doesn't occur in LHS	No	`"Fabrikam" !contains "xyz"`
`contains_cs`	RHS occurs as a subsequence of LHS	Yes	`"FabriKam" contains_cs "Kam"`
`!contains_cs`	RHS doesn't occur in LHS	Yes	`"Fabrikam" !contains_cs "Kam"`
`endswith`	RHS is a closing subsequence of LHS	No	`"Fabrikam" endswith "Kam"`

Operator	Description	Case-Sensitive	Example (yields `true`)
`!endswith`	RHS isn't a closing subsequence of LHS	No	`"Fabrikam" !endswith "brik"`
`endswith_cs`	RHS is a closing subsequence of LHS	Yes	`"Fabrikam" endswith_cs "kam"`
`!endswith_cs`	RHS isn't a closing subsequence of LHS	Yes	`"Fabrikam" !endswith_cs "brik"`
`has`	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
`!has`	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
`has_all`	Same as `has` but works on all of the events	No	`"North and South America" has_all("south", "north")`
`!has_all`	Not all of the RHS terms are present in LHS	No	`"North and South America" !has_all("south", "east")`
`has_any`	Same as `has` but works on any of the events	No	`"North America" has_any("south", "north")`
`!has_any`	None of the RHS terms are present in LHS	No	`"North and South America" !has_any("east", "west")`
`has_cs`	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
`!has_cs`	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`
`hasprefix`	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
`!hasprefix`	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
`hasprefix_cs`	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
`!hasprefix_cs`	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

Operator	Description	Case-Sensitive	Example (yields `true`)
`hassuffix`	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
`!hassuffix`	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
`hassuffix_cs`	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
`!hassuffix_cs`	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`
`in`	Equal to any of the events	Yes	`"abc" in ("123", "345", "abc")`
`!in`	Not equal to any of the events	Yes	`"bca" !in ("123", "345", "abc")`
`in~`	Equal to any of the events	No	`"Abc" in~ ("123", "345", "abc")`
`!in~`	Not equal to any of the events	No	`"bCa" !in~ ("123", "345", "ABC")`
`matches regex`	LHS contains a match for RHS	Yes	`"Fabrikam" matches regex "b.*k"`
`startswith`	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
`!startswith`	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
`startswith_cs`	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
`!startswith_cs`	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

10.2.8.1. ==

The `==` (equals) operator applies a case-sensitive string, or regex literal, filter and returns events that match.

The following table provides a comparison of the `equal` operators:

Operator	Description	Case-Sensitive	Example (yields `true`)
`==`	Equal	Yes	`"aBc" == "aBc"`
`!=`	Not equal	Yes	`"abc" != "ABC"`
`=~`	Equal	No	`"abc" =~ "ABC"`
`!~`	Not equal	No	`"aBc" !~ "xyz"`

When comparing values of different types, Cribl Search performs automatic type conversion wherever possible, giving priority to number comparisons.

For more details on comparison rules, see:

- Comparison operators of the `cribl` operator
- Comparison operators of the `where` operator

Syntax

```
Scope | where Field == (Expression, ... )
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression used to filter.



To pass regex literals as **Expression** arguments, see syntax details at Regex Examples, Regex Flags, and Disambiguate Regex Characters.

Example

```
dataset=myDataset
| where state == "Kansas"
| count
```

Copy

10.2.8.2. !=

The `!=` (not equals) operator applies a case-sensitive string, or regex literal, filter, and excludes events that match.

The following table provides a comparison of the `equal` operators:

Operator	Description	Case-Sensitive	Example (yields `true`)
`==`	Equal	Yes	`"aBc" == "aBc"`
`!=`	Not equal	Yes	`"abc" != "ABC"`
`=~`	Equal	No	`"abc" =~ "ABC"`
`!~`	Not equal	No	`"aBc" !~ "xyz"`

When comparing values of different types, Cribl Search performs automatic type conversion wherever possible, giving priority to number comparisons.

For more details on comparison rules, see:

- Comparison operators of the `cribl` operator
- Comparison operators of the `where` operator

Syntax

```
Scope | where Field != (Expression, ... )
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression used to filter.

> To pass regex literals as **Expression** arguments, see syntax details at Regex Examples, Regex Flags, and Disambiguate Regex Characters.

Example

```
dataset=myDataset
| where state != "Kansas"
| project EventId, State
```

Copy

10.2.8.3. =~

The `=~` (equals) operator applies a case-insensitive string filter and returns events that match.

The following table provides a comparison of the `equal` operators:

Operator	Description	Case-Sensitive	Example (yields `true`)
`==`	Equal	Yes	`"aBc" == "aBc"`
`!=`	Not equal	Yes	`"abc" != "ABC"`
`=~`	Equal	No	`"abc" =~ "ABC"`
`!~`	Not equal	No	`"aBc" !~ "xyz"`

When comparing values of different types, Cribl Search performs automatic type conversion wherever possible, giving priority to number comparisons.

For more details on comparison rules, see:

- Comparison operators of the `cribl` operator
- Comparison operators of the `where` operator

Syntax

```
Scope | where Field =~ (Expression, ... )
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression used to filter.

Example

```
dataset=myDataset
| where state =~ "Kansas"
| project EventId, State
```

Copy

10.2.8.4. !~

The `!~` (not equals) operator applies a case-insensitive string filter and excludes events that match.

The following table provides a comparison of the `equal` operators:

Operator	Description	Case-Sensitive	Example (yields `true`)
`==`	Equal	Yes	`"aBc" == "aBc"`
`!=`	Not equal	Yes	`"abc" != "ABC"`
`=~`	Equal	No	`"abc" =~ "ABC"`
`!~`	Not equal	No	`"aBc" !~ "xyz"`

When comparing values of different types, Cribl Search performs automatic type conversion wherever possible, giving priority to number comparisons.

For more details on comparison rules, see:

- Comparison operators of the `cribl` operator
- Comparison operators of the `where` operator

Syntax

```
Scope | where Field !~ (Expression, ... )
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where (State !~ "texas") and (event_count > 3000)
| project State, event_count
```

Copy

10.2.8.5. Contains

The `contains` operator applies a case-insensitive string filter and returns events that match.

The following table compares the `contains` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`contains`	RHS occurs as a subsequence of LHS	No	`"FabriKam" contains "BRik"`
`!contains`	RHS doesn't occur in LHS	No	`"Fabrikam" !contains "xyz"`
`contains_cs`	RHS occurs as a subsequence of LHS	Yes	`"FabriKam" contains_cs "Kam"`
`!contains_cs`	RHS doesn't occur in LHS	Yes	`"Fabrikam" !contains_cs "Kam"`

Syntax

```
Scope | where Field contains String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State contains "enn"
```

Copy

10.2.8.6. !Contains

The `!contains` operator applies a case-insensitive string filter and excludes events that match.

The following table compares the `contains` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`contains`	RHS occurs as a subsequence of LHS	No	`"FabriKam" contains "BRik"`
`!contains`	RHS doesn't occur in LHS	No	`"Fabrikam" !contains "xyz"`
`contains_cs`	RHS occurs as a subsequence of LHS	Yes	`"FabriKam" contains_cs "Kam"`
`!contains_cs`	RHS doesn't occur in LHS	Yes	`"Fabrikam" !contains_cs "Kam"`

Syntax

```
Scope | where Field !contains String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State !contains "enn"
```

Copy

10.2.8.7. CONTAINS_CS

The `contains_cs` operator applies a case-sensitive string filter and returns events that match.

The following table compares the `contains` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`contains`	RHS occurs as a subsequence of LHS	No	`"FabriKam" contains "BRik"`
`!contains`	RHS doesn't occur in LHS	No	`"Fabrikam" !contains "xyz"`
`contains_cs`	RHS occurs as a subsequence of LHS	Yes	`"FabriKam" contains_cs "Kam"`
`!contains_cs`	RHS doesn't occur in LHS	Yes	`"Fabrikam" !contains_cs "Kam"`

Syntax

```
Scope | where Field contains_cs String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State contains_cs "AS"
```

Copy

10.2.8.8. !CONTAINS_CS

The `!contains_cs` operator applies a case-sensitive string filter and excludes events that match.

The following table compares the `contains` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`contains`	RHS occurs as a subsequence of LHS	No	`"FabriKam" contains "BRik"`
`!contains`	RHS doesn't occur in LHS	No	`"Fabrikam" !contains "xyz"`
`contains_cs`	RHS occurs as a subsequence of LHS	Yes	`"FabriKam" contains_cs "Kam"`
`!contains_cs`	RHS doesn't occur in LHS	Yes	`"Fabrikam" !contains_cs "Kam"`

Syntax

```
Scope | where Field !contains_cs String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State !contains_cs "AS"
```

Copy

10.2.8.9. ENDSWITH

The `endswith` operator applies a case-insensitive ending string filter and returns events that match.

The following table compares the `endswith` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`endswith`	RHS is a closing subsequence of LHS	No	`"Fabrikam" endswith "Kam"`
`!endswith`	RHS isn't a closing subsequence of LHS	No	`"Fabrikam" !endswith "brik"`
`endswith_cs`	RHS is a closing subsequence of LHS	Yes	`"Fabrikam" endswith_cs "kam"`
`!endswith_cs`	RHS isn't a closing subsequence of LHS	Yes	`"Fabrikam" !endswith_cs "brik"`

Syntax

```
Scope | where Field endswith String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize Events=count() by State
| where State endswith "sas"
```

Copy

10.2.8.10. !ENDSWITH

The `!endswith` operator applies a case-sensitive ending string filter and excludes events that match.

The following table compares the `endswith` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`endswith`	RHS is a closing subsequence of LHS	No	`"Fabrikam" endswith "Kam"`
`!endswith`	RHS isn't a closing subsequence of LHS	No	`"Fabrikam" !endswith "brik"`
`endswith_cs`	RHS is a closing subsequence of LHS	Yes	`"Fabrikam" endswith_cs "kam"`
`!endswith_cs`	RHS isn't a closing subsequence of LHS	Yes	`"Fabrikam" !endswith_cs "brik"`

Syntax

```
Scope | where Field !endswith String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize Events=count() by State
| where State !endswith "is"
```

Copy

10.2.8.11. ENDSWITH_CS

The `endswith_cs` operator applies a case-sensitive ending string filter and returns events that match.

The following table compares the `endswith` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`endswith`	RHS is a closing subsequence of LHS	No	`"Fabrikam" endswith "Kam"`
`!endswith`	RHS isn't a closing subsequence of LHS	No	`"Fabrikam" !endswith "brik"`
`endswith_cs`	RHS is a closing subsequence of LHS	Yes	`"Fabrikam" endswith_cs "kam"`
`!endswith_cs`	RHS isn't a closing subsequence of LHS	Yes	`"Fabrikam" !endswith_cs "brik"`

Syntax

```
Scope | where Field endswith_cs String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize Events=count() by State
| where State endswith_cs "NA"
```

Copy

10.2.8.12. !ENDSWITH_CS

The `!endswith_cs` operator applies a case-sensitive ending string filter and excludes events that match.

The following table compares the `endswith` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`endswith`	RHS is a closing subsequence of LHS	No	`"Fabrikam" endswith "Kam"`
`!endswith`	RHS isn't a closing subsequence of LHS	No	`"Fabrikam" !endswith "brik"`
`endswith_cs`	RHS is a closing subsequence of LHS	Yes	`"Fabrikam" endswith_cs "kam"`
`!endswith_cs`	RHS isn't a closing subsequence of LHS	Yes	`"Fabrikam" !endswith_cs "brik"`

Syntax

```
Scope | where Field !endswith_cs String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize Events=count() by State
| where State !endswith_cs "is"
```

Copy

10.2.8.13. Has

The `has` operator applies a case-insensitive string filter and returns events that match.

The following table compares the `has` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`has`	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
`!has`	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
`has_cs`	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
`!has_cs`	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`

Syntax

```
Scope | where Field has String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State has "New"
```

Copy

10.2.8.14. !Has

The `!has` operator applies a case-insensitive string filter and excludes events that match.

The following table compares the `has` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
has	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
!has	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
has_cs	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
!has_cs	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`

Syntax

```
Scope | where Field !has String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State !has "New"
```

Copy

10.2.8.15. HAS_CS

The `has_cs` operator applies a case-sensitive string filter and returns events that match.

The following table compares the `has` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`has`	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
`!has`	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
`has_cs`	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
`!has_cs`	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`

Syntax

```
Scope | where Field has_cs String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State has_cs "FLORIDA"
```

Copy

10.2.8.16. !Has_cs

The `!has_cs` operator applies a case-sensitive string filter and excludes events that match.

The following table compares the `has` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
has	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
!has	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
has_cs	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
!has_cs	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`

Syntax

```
Scope | where Field !has_cs String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State !has_cs "New"
```

Copy

10.2.8.17. HAS_ALL

The `has_all` operator applies one or more case-insensitive string filters and returns events that match.

Syntax

```
Scope | where Field has_all (Expression, ... )
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression that specifies the values to search. An expression can be a type value or expression that produces a set of values.

Example

```
dataset=myDataset
| where EpisodeNarrative has_all ("cold", "strong", "afternoon", "hail")
| summarize Count=count() by EventType
```

Copy

10.2.8.18. !HAS_ALL

The `!has_all` operator applies one or more case-insensitive string filters and excludes events that do not match.

Syntax

```
Scope | where Field !has_all (Expression, ... )
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression that specifies the values to search. An expression can be a type value or expression that produces a set of values.

Example

```
dataset=myDataset
| where EpisodeNarrative !has_all ("cold", "strong", "afternoon", "hail")
| summarize Count=count() by EventType
```

Copy

10.2.8.19. HAS_ANY

The `has_any` operator applies any case-insensitive string filters and returns events that match.

Syntax

```
Scope | where Field has_any (Expression, ... )
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression that specifies the values to search. An expression can be a type value or expression that produces a set of values.

Examples

```
dataset=cribl_search_sample
| where action has_any ("ACCEPT", "REJECT")
| summarize count() by action
```

Copy | Run in Cribl Search

```
dataset=myDataset
| where State has_any ("CAROLINA", "DAKOTA", "NEW")
| summarize count() by State
```

Copy

10.2.8.20. !HAS_ANY

The `!has_any` operator applies any case-insensitive string filters and excludes events that do not match.

Syntax

```
Scope | where Field !has_any (Expression, ... )
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression that specifies the values to search. An expression can be a type value or expression that produces a set of values.

Example

```
dataset=myDataset
| where State !has_any ("CAROLINA", "DAKOTA", "NEW")
| summarize count() by State
```

Copy

10.2.8.21. HASPREFIX

The `hasprefix` operator applies a case-insensitive starting string filter and returns events that match. Any of the words in the string can have the prefix.

The following table compares the `hasprefix` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`hasprefix`	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
`!hasprefix`	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
`hasprefix_cs`	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
`!hasprefix_cs`	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

Syntax

```
Scope | where Field hasprefix Expression
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The expression for which to search.

Example

```
dataset=myDataset
| summarize Events=count() by State
| where State hasprefix "la"
```

Copy

10.2.8.22. !HASPREFIX

The `!hasprefix` operator applies a case-insensitive starting string filter and excludes events that match. Any of the words in the string can have the prefix.

The following table compares the `hasprefix` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`hasprefix`	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
`!hasprefix`	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
`hasprefix_cs`	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
`!hasprefix_cs`	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

Syntax

```
Scope | where Field !hasprefix Expression
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The expression for which to search.

Example

```
dataset=myDataset
| summarize Events=count() by State
| where State !hasprefix "N"
```

Copy

10.2.8.23. HASPREFIX_CS

The `hasprefix_cs` operator applies a case-sensitive starting string filter and returns events that match. Any of the words in the string can have the prefix.

The following table compares the `hasprefix` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`hasprefix`	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
`!hasprefix`	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
`hasprefix_cs`	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
`!hasprefix_cs`	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

Syntax

```
Scope | where Field hasprefix_cs Expression
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The expression for which to search.

Example

```
dataset=myDataset
| summarize Events=count() by State
| where State hasprefix_cs "P"
```

Copy

10.2.8.24. !HASPREFIX_CS

The `!hasprefix_cs` operator applies a case-sensitive starting string filter and excludes events that match. Any of the words in the string can have the prefix.

The following table compares the `hasprefix` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`hasprefix`	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
`!hasprefix`	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
`hasprefix_cs`	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
`!hasprefix_cs`	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

Syntax

```
Scope | where Field !hasprefix_cs Expression
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The expression for which to search.

Example

```
dataset=myDataset
| summarize Events=count() by State
| where State !hasprefix_cs "P"
```

Copy

10.2.8.25. HASSUFFIX

The `hassuffix` operator applies a case-insensitive ending string filter and returns events that match. Any of the words in the string can have the suffix.

The following table compares the `hassuffix` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`hassuffix`	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
`!hassuffix`	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
`hassuffix_cs`	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
`!hassuffix_cs`	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`

Syntax

```
Scope | where Field hassuffix Expression
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The scalar or literal expression to search.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State hassuffix "o"
```

Copy

10.2.8.26. !HASSUFFIX

The `!hassuffix` operator applies a case-insensitive ending string filter and excludes events that match. Any of the words in the string can have the suffix.

The following table compares the `hassuffix` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`hassuffix`	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
`!hassuffix`	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
`hassuffix_cs`	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
`!hassuffix_cs`	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`

Syntax

```
Scope | where Field !hassuffix Expression
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The scalar or literal expression to search.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State !hassuffix "A"
```

Copy

10.2.8.27. HASSUFFIX_CS

The `hassuffix_cs` operator applies a case-sensitive ending string filter and returns events that match. Any of the words in the string can have the suffix.

The following table compares the `hassuffix` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`hassuffix`	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
`!hassuffix`	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
`hassuffix_cs`	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
`!hassuffix_cs`	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`

Syntax

```
Scope | where Field hassuffix_cs Expression
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The scalar or literal expression to search.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State hassuffix_cs "AS"
```

Copy

10.2.8.28. !HASSUFFIX_CS

The `!hassuffix_cs` operator applies a case-sensitive ending string filter and excludes events that match. Any of the words in the string can have the suffix.

The following table compares the `hassuffix` operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`hassuffix`	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
`!hassuffix`	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
`hassuffix_cs`	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
`!hassuffix_cs`	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`

Syntax

```
Scope | where Field !hassuffix_cs Expression
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The scalar or literal expression to search.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State !hassuffix_cs "AS"
```

Copy

10.2.8.29. In

The `in` operator applies a case-sensitive string or regex literal filter.

The following table provides a comparison of the `in` operators:

Operator	Description	Case-Sensitive	Example (yields true)
`in`	Equal to one of the events	Yes	`"abc" in ("123", "345", "abc")`
`!in`	Not equal to any of the events	Yes	`"bca" !in ("123", "345", "abc")`
`in~`	Equal to any of the events	No	`"Abc" in~ ("123", "345", "abc")`
`!in~`	Not equal to any of the events	No	`"bCa" !in~ ("123", "345", "ABC")`

Syntax

```
Scope | where Field in (Expression, ... )
```

Or:

```
Scope in ListOfValues
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression that specifies the values to search. An expression can be a type value, an expression that produces a set of values, or a regex literal expression that returns a `bool` value.

  To pass regex literals as **Expression** arguments, see syntax details at Regex Examples, Regex Flags, and Disambiguate Regex Characters.

- **ListOfValues** Comma-separated list of one or more `StringExpressions`. For example, `(StringExpression, StringExpression, ...)`.

Example

The following query shows how to use `in` with a list of values.

```
dataset=myDataset
 | where State in ("FLORIDA", "GEORGIA", "NEW YORK")
 | count
```

Copy

Filter the `logs` Dataset for `host` fields that have a case-sensitive match in the `foo` query.

```
let foo =dataset="$vt_dummy" event < 3
 | extend x = case(
    event == 0, dynamic({"method": "GET"}),
    event == 1, dynamic({"method": "POST"}),
    dynamic({"method": "PATCH"})
    );

dataset="cribl_internal_logs"
| where  method in foo.x.method
```

Copy

10.2.8.30. !In

The `!in` operator applies a case-sensitive string or regex literal filter, and excludes events that match the specified criteria.

The following table provides a comparison of the `in` operators:

Operator	Description	Case-Sensitive	Example (yields true)
`in`	Equal to one of the events	Yes	`"abc" in ("123", "345", "abc")`
`!in`	Not equal to any of the events	Yes	`"bca" !in ("123", "345", "abc")`
`in~`	Equal to any of the events	No	`"Abc" in~ ("123", "345", "abc")`
`!in~`	Not equal to any of the events	No	`"bCa" !in~ ("123", "345", "ABC")`

Syntax

```
Scope | where Field !in (Expression, ... )
```

Or:

```
Scope !in ListOfValues
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression that specifies the values to search. An expression can be a type value, an expression that produces a set of values, or a regex literal expression that returns a `bool` value.

  To pass regex literals as **Expression** arguments, see syntax details at Regex Examples, Regex Flags, and Disambiguate Regex Characters.

- **ListOfValues** Comma-separated list of one or more `StringExpressions`. For example, `(StringExpression, StringExpression, ...)`.

Example

The following query shows how to use `!in` with a list of values.

```
dataset=myDataset
  | where State !in ("FLORIDA", "GEORGIA", "NEW YORK")
  | count
```

Copy

Filter the `logs` Dataset for `host` fields that don't have a case-sensitive match in the `foo` query.

```
let foo = dataset=api
  | summarize count=count() by hostname

dataset=logs
  | where host !in foo.hostname
```

Copy

10.2.8.31. In~

The `in~` operator applies a case-insensitive string or regex literal filter.

The following table provides a comparison of the `in` operators:

Operator	Description	Case-Sensitive	Example (yields true)
`in`	Equal to one of the events	Yes	`"abc" in ("123", "345", "abc")`
`!in`	Not equal to any of the events	Yes	`"bca" !in ("123", "345", "abc")`
`in~`	Equal to any of the events	No	`"Abc" in~ ("123", "345", "abc")`
`!in~`	Not equal to any of the events	No	`"bCa" !in~ ("123", "345", "ABC")`

Syntax

```
Scope | where Field in~ (Expression, ... )
```

Or:

```
Scope in~ ListOfValues
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression that specifies the values to search. An expression can be a type value, an expression that produces a set of values, or a regex literal expression that returns a `bool` value.

  To pass regex literals as **Expression** arguments, see syntax details at Regex Examples, Regex Flags, and Disambiguate Regex Characters.

- **ListOfValues** Comma-separated list of one or more `StringExpressions`. For example, `(StringExpression, StringExpression, ...)`.

Example

The following query shows how to use `in~` with a list of values.

```
dataset=myDataset
  | where State in~ ("FLORIDA", "GEORGIA", "NEW YORK")
  | count
```

Copy

Filter the `logs` Dataset for `host` fields that have a case-insensitive match in the `foo` query.

```
let foo = dataset=api
  | summarize count=count() by hostname

dataset=logs
  | where host in~ foo.hostname
```

Copy

10.2.8.32. !IN~

The `!in~` operator applies a case-insensitive string or regex literal filter, and excludes events that match the specified criteria.

The following table provides a comparison of the `in` operators:

Operator	Description	Case-Sensitive	Example (yields true)
`in`	Equal to one of the events	Yes	`"abc" in ("123", "345", "abc")`
`!in`	Not equal to any of the events	Yes	`"bca" !in ("123", "345", "abc")`
`in~`	Equal to any of the events	No	`"Abc" in~ ("123", "345", "abc")`
`!in~`	Not equal to any of the events	No	`"bCa" !in~ ("123", "345", "ABC")`

Syntax

```
Scope | where Field !in~ (Expression, ... )
```

Or:

```
Scope !in~ ListOfValues
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: An expression that specifies the values to search. An expression can be a type value, an expression that produces a set of values, or a regex literal expression that returns a `bool` value.

  To pass regex literals as **Expression** arguments, see syntax details at Regex Examples, Regex Flags, and Disambiguate Regex Characters.

- **ListOfValues** Comma-separated list of one or more `StringExpressions`. For example, `(StringExpression, StringExpression, ...)`.

Example

The following query shows how to use `!in~` with a list of values.

```
dataset=myDataset
  | where State !in~ ("FLORIDA", "GEORGIA", "NEW YORK")
  | count
```

Copy

Filter the `logs` Dataset for `host` fields that don't have a case-insensitive match in the `foo` query.

```
let foo = dataset=api
  | summarize count=count() by hostname

dataset=logs
  | where host !in~ foo.hostname
```

Copy

10.2.8.33. Matches Regex

The `matches regex` operator applies a case-sensitive regex, or a regex literal (with specified case-sensitivity), and returns events that match.

Syntax

```
Scope | where Field matches regex Expression
```

For detailed syntax, see Regex Flags and Disambiguate Regex Characters.

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **Expression**: The string or regex literal to search.

Examples

Match a string, with wildcard:

```
dataset=myDataset
| summarize event_count=count() by State
| where State matches regex "K.*S"
```

Copy

Match a regex literal (all `POST` requests, case-insensitive):

```
dataset="cribl_internal_logs" method=*
| limit 1000
| where method matches regex /^po.*/i
```

Copy Run in Cribl Search

For further examples using the `where` operator, see Regex Examples.

10.2.8.34. STARTSWITH

The `startswith` operator applies a case-insensitive string starting sequence filter and returns events that match.

The following table compares the startswith operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`startswith`	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
`!startswith`	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
`startswith_cs`	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
`!startswith_cs`	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

Syntax

```
Scope | where Field startswith String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State startswith "Lo"
```

Copy

10.2.8.35. !STARTSWITH

The `!startswith` operator applies a case-insensitive string starting sequence filter and excludes events that match.

The following table compares the startswith operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`startswith`	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
`!startswith`	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
`startswith_cs`	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
`!startswith_cs`	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

Syntax

```
Scope | where Field !startswith String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State !startswith "Lo"
```

Copy

10.2.8.36. STARTSWITH_CS

The `startswith_cs` operator applies a case-sensitive string starting sequence filter and returns events that match.

The following table compares the startswith operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`startswith`	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
`!startswith`	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
`startswith_cs`	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
`!startswith_cs`	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

Syntax

```
Scope | where Field startswith_cs String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State startswith_cs "Lo"
```

Copy

10.2.8.37. !STARTSWITH_CS

The `!startswith_cs` operator applies a case-sensitive string starting sequence filter and excludes events that match.

The following table compares the startswith operators using these abbreviations:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields true)
`startswith`	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
`!startswith`	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
`startswith_cs`	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
`!startswith_cs`	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

Syntax

```
Scope | where Field !startswith_cs String
```

Arguments

- **Scope**: The input tabular result set to filter.
- **Field**: The field to filter.
- **String**: The string used to filter.

Example

```
dataset=myDataset
| summarize event_count=count() by State
| where State !startswith_cs "Lo"
```

Copy

10.3. Commands

Run commands, and `set`-statement options, from the query box to manage searches.

Cribl Search supports the following commands:

- `.cancel` – Stop queued, running, or all queued and running searches.
- `.clear` – Disable `set`-statement options.
- `.generate` – Produce statistics about the results of a search.
- `.show` - List objects in a Dataset, set-statement options or searches.

Commands can be combined with the following literals:

- `objects` – List the contents of a Dataset before you search it.
- `options` – View `set`-statement options configured for your account.
- `stats`
- `queries` – List queued, running, or all queued and running searches run by all users or specific users.

Basic Command Usage

Unlike functions and operators, commands start with a period. For example:

```
.show objects
```

Copy

How commands work depends on your Search Member Permissions. For example, **User** Search Members can manage only their own searches, but **Admin** Search Members can manage the searches of all users in the organization.

You can combine commands with operators. For example, to show queued queries that were created more than 10 minutes ago:

```
.show queued queries
 | where timeCreated < ago(10min)
```

Copy Run in Cribl Search

However, you can't use commands in subqueries. For example, this won't work:

```
// invalid example
let stage1 = .show objects(cribl_search_sample);
```

Copy

For more transparency, you can provide a reason why you're using a particular command. The reason will be added to the Cribl Search audit log. For example:

```
.cancel running queries with(reason = "Time is up.")
```

Copy Run in Cribl Search

Manage Searches with Commands

You can manage your or your users' searches straight from the query box, using the following commands:

- `.show queries` – View searches based on their IDs, status, or the users running them.
- `.cancel` – Cancel queued or running searches.

View Searches

To display searches based on their IDs, status, or the users running them, use the `.show queries` command.

User and **Editor** Search Members can view only their own searches. **Admin** Search Members can view the searches of all users in the organization.

To view all queued or running searches:

```
.show queries
```

Copy Run in Cribl Search

To view all queued searches:

```
.show queued queries
```

Copy Run in Cribl Search

To view searches that are being run by specific users:

```
.show running queries by user "Jane*"
```

Copy | Run in Cribl Search

The results are not affected by the set time range.

Cancel Searches

To stop queued or running searches, use the `.cancel` command.

User and **Editor** Search Members can cancel only their own searches. **Admin** Search Members can cancel the searches of all users in the organization.

To cancel a specific search:

```
.cancel query "1693827597495.ji5y5g"
```

Copy

To cancel searches that are queued or being run by specific users:

```
.cancel queries by user("John Doe", "Jane*")
```

Copy | Run in Cribl Search

To cancel all currently running searches:

```
.cancel running queries
```

Copy | Run in Cribl Search

The results are not affected by the set time range.

Manage Set-Statement Options with Commands

You can manage your or your users' `set`-statement options straight from the query box, using the following commands:

- `.clear options` – Disable `set`-statement options.
- `.show options` – View `set`-statement options configured for your account.

View Set-Statement Options

To see `set`-statement options configured for your account, use the `.show options` command.

To view all options configured for you:

```
.show options
```

Copy | Run in Cribl Search

To view only those options that are **not** overridden by other settings:

```
.show active options
```

Copy | Run in Cribl Search

Disable Set-Statement Options

To disable `set`-statement options, use the `.clear options` command.

To disable all options configured for your own account:

```
.clear options
```

Copy

As an Admin Search Member, to disable all options for all users in the usage group:

```
.clear global options
```

Copy

10.3.1. .Cancel

The `.cancel` command stops queued, running, or all queued and running searches.

`.cancel` ignores the set time range.

```
// cancel all running searches
.cancel running queries with(reason = "Time is up.")
```

Copy | Run in Cribl Search

Purpose

Use `.cancel` to manage system resources and optimize performance by terminating queries that are no longer needed, taking too long, consuming too many resources, or made in error.

Permissions

Search Member Type	Permissions
Admin	Can cancel all searches in their Organization.
Editor or User	Can cancel only their own searches.

Syntax

```
.cancel query SearchId [with(reason = CancellationReason)]
```

Or:

```
.cancel Status queries
              [by user (User[, ...])]
              [with(reason = CancellationReason)]
```

Copy

You can also combine this command with operators.

Parameters

Parameter	Type	Description
`SearchId`	`string`	The ID of the search to cancel.
`CancellationReason`	`string`	The reason for running the `.cancel` command, which is added to the Cribl Search audit log.
`Status`	`string`	The status of the queries to cancel: • `queued`: Cancel only queued searches. • `running`: Cancel only running searches. • `all` or `*`: Cancel both queued and running searches.
`User`	`string`	The name of the user whose searches you want to cancel.

Returns

Returns information about the canceled searches.

Examples

Cancel a specific search.

```
.cancel query "1693827597495.ji5y5g"
```

Copy

Cancel searches queued or being run by specific users.

```
.cancel queries by user("John Doe", "Jane*") with(reason = "I don't need these searcl
```

Copy

Cancel all searches started by users whose names start with `John` or `Jane`.

```
.cancel queries by user("John*", "Jane*") with(reason = "Cleaning up.")
```

Copy Run in Cribl Search

Cancel queued searches that were created more than 10 minutes ago.

```
.cancel queued queries
 | where timeCreated < ago(10min)
```

Copy Run in Cribl Search

10.3.2. .Show

The `.show` command can be used with:

- `objects` literal to list objects included in a Dataset, including files, tables, and endpoints, along with information about each object. The results are filtered by the set time range.
- `options` literal to list `set`-statement options configured for your account.
- `queries` literal to list queued, running, or all pending (queued and running) searches. You can limit the list to only searches run by a specific user.

```
// list all objects in the cribl_search_sample Dataset
.show objects(cribl_search_sample)
```

Copy Run in Cribl Search

10.3.3. .Clear

The `.clear` command is used with the `options` literal to disable `set`-statement options.

```
// disable all set-statement options configured for your account
.clear options
```

Copy Run in Cribl Search

10.3.4. .Generate

The `.generate` command is used with the `stats` literal to produce statistics about the results of a search.

```
// produce statistics for the search 1749594773597.yzoXZy, if available to the currer
.generate stats job("1749594773597.yzoXZy")
```

Copy

10.3.5. Objects

The `objects` literal is used with the `.show` command to list objects included in a Dataset, including files, tables, and endpoints, along with information about each object. The results are filtered by the set time range.



Cribl Edge Datasets are not supported.

Purpose

Use `.show objects` to get detailed information about the contents of a Dataset before you search it.

Permissions

Search Member Type	Permissions
Admin	Can see the objects of all Datasets in their organization.
Editor	Can see the objects of only those Datasets that they created themselves or that were shared with them.
User	Can see the objects of only those Datasets that were shared with them.

Syntax

```
.show objects(DatasetID[, ...]) [with(reason = "InvestigationReason")]
```

You can also combine this command with operators.

Parameters

Parameter	Type	Description
`DatasetID`	`string`	The ID of the Dataset whose objects you want to list. Separate multiple Dataset IDs with a comma `,`. Wildcards `*` are supported at the end or in the middle of an ID, as in `cribl_*_logs*`.
`InvestigationReason`	`string`	The reason for running the `.show objects` command, which is added to the Cribl Search audit log.

Returns

The `.show objects` command returns a list of files, tables, endpoints, metrics, and any other units that the specified Dataset contains. Each object in the Dataset is represented by one row in the results table.

The details about each object vary according to which objects the Dataset contains. Depending on the Dataset, the results may include the following details:

Field	Description
`_time`	The `.show objects` command's execution time. It's the same for every returned object.
`success`	Shows whether the command was successful.
`dataset`	The ID of the examined Dataset.
`datasetType`	The type of the examined Dataset.
`name`	The name of the object.
`size`	The size of the object in bytes.
`lastModified`	The object's last modification time.

Examples

List all objects in the `cribl_internal_logs` and `cribl_search_sample` Datasets.

```
.show objects(cribl_internal_logs, cribl_search_sample)
```

Copy Run in Cribl Search

List the five largest objects in the `cribl_internal_logs` Dataset.

```
.show objects(cribl_internal_logs) | top 5 by size
```

Copy Run in Cribl Search

See how many objects larger than 50,000 bytes exist in all Datasets whose IDs start with `cribl_int`.

```
.show objects(cribl_int*)
    with(reason = "Investigating number of large objects")
    | where size > 50000 | count
```

Copy Run in Cribl Search

10.3.6. Options

The `options` literal can be used with:

- `.clear` command to disable `set`-statement options.
- `.show` command to list `set`-statement options configured for your account.

Show Options

Use `.show options` to get a list of `set`-statement options that you, or an Admin Search Member, have configured for your account. This command lets you:

- See the values of the options.
- Find out which options are active and which are overridden by other settings.
- Identify which options were set by you and which were set the Search Admin.

Permissions

All Search Members can see all `set`-statement options that apply to their own account.

Syntax

```
.show [Scope] options [with(reason = "InvestigationReason")]
```

Parameters

Parameter	Type	Description
`Scope`	`string`	Scope for listing the options: • `all` or `*`: List all options configured for your account. • `active`: List only those options that may affect your searches at the moment (that is, options that are not being overridden by other settings).
`InvestigationReason`	`string`	The reason for running the `.show options` command, which is added to the Cribl Search audit log.

Returns

Returns a list of `set`-statement options configured for your account. The results include the following details:

Field	Description
`_time`	Execution time of the `.show options` command.
`success`	`true` if the `.show options` command was successful.
`option`	The name of the `set`-statement option.
`value`	The value of the option.
`source`	The scope in which the option was set.
`active`	`true` if the option affects your searches. `false` if the option doesn't affect you at the moment because it's being overridden by another setting.

Examples

See all `set`-statement options that affect you at the moment.

```
.show active options
            with(reason = "Just checking options active for my account.")
```

Copy Run in Cribl Search

Clear Options

Use `.clear options` to disable options that were configured by `set` statements.

To see which `set`-statement options affect your account at the moment before running `.clear options`, use the `.show options` command.

Permissions

Search Member Type	Permissions
Admin	Can disable any options configured for any scope.
Editor or User	Can disable only those options that they configured themselves.

Syntax

```
.clear [Scope] options [with(reason="DeactivationReason")]
```

Parameters

Parameter	Type	Description
`Scope`	`string`	Scope of deactivation: • `user` (default): Deactivate all options that the current user set for their own account. • `global` (Admin only): Deactivate all options set for all users in the matching Usage Group. • `all` (Admin only): Deactivate according to both the `user` and `global` scopes.
`DeactivationReason`	`string`	A `string` that provides a reason for disabling the options, to add to the Cribl Search audit log.

Returns

Returns a single event that confirms the options are disabled.

Examples

Disable all `set`-statement options configured for your own account.

```
.clear user options
            with(reason = "Clearing all my options.")
```

Copy

Disable all `set`-statement options of all users in the Usage Group.

```
// Admin only
.clear global options
              with(reason = "Clearing all options for my users.")
```

Copy

Disable all `set`-statement options, both for your own account and for all users in the Usage Group.

```
// Admin only
.clear all options
           with(reason = "Clearing all options for me and my users.")
```

Copy

10.3.7. Stats

The `stats` literal is used with the `.generate` command to produce statistics about the results of a search.

Purpose

A standard Cribl Search query generates two types of statistics about its result set:

- **Timeline stats**: Distribution of data over time, such as the number of events per time bucket, or the earliest and latest timestamps in the result set. You can see these stats in the histogram of the search results.
- **Field stats**: Distribution of data across fields, such as the number of unique values in a field, or the minimum and maximum values of a field. You can see these stats in the field browser of the search results.

Certain types of searches (for example, Dashboard or scheduled searches) don't generate these statistics by default, to optimize performance.

The `.generate stats` command allows you to generate these statistics for any search, without having to rerun the search itself.

Permissions

Search Member Type	Permissions
Admin	Can generate stats for all searches in the Organization.
Editor or User	Can generate stats for public searches, or searches that they ran themselves.

Syntax

```
.generate [timeline|field|all] stats("SearchId") [force] [with(reason = "Investigatic
```

Copy

Parameters

Parameter	Required	Description
`timeline` or `field` or `all`	No	Type of statistics to generate. Use `timeline` for timeline stats, `field` for field stats, or `all` for both. If omitted, defaults to `all`.
`SearchId`	Yes	The ID of the search for which to generate statistics. Doesn't support wildcards.
`force`	No	Force the (re)generation of stats even if they were already generated.
`Investigation Reason`	No	The reason for running the command, added to the Cribl Search audit log.

Returns

Returns a message indicating whether the statistics were successfully generated or not.

If the search ID is invalid or the user doesn't have permission to access the search, returns an error message.

Examples

Generate timeline statistics for a search, overwriting any existing stats.

```
.generate timeline stats("1749594773597.yzoXZy") force with (reason="I have my reasor
```

Copy

10.3.8. Queries

The `queries` literal is used with the `.show` command to list queued, running, or all pending (queued and running) searches. You can limit the list to only searches run by a specific user.

Purpose

Use `.show queries` to monitor search status, track individual and overall search performance, and identify searches that are taking longer than expected. This command is especially useful in multi-user environments.

Permissions

Search Member Type	Permissions
Admin	Can see all searches in their Organization.
Editor or User	Can see only those searches that they ran themselves or that were shared with them.

Syntax

```
.show [Status] queries [by user User] [with(reason = "InvestigationReason")]
```

You can also combine this command with operators.

Parameters

Parameter	Type	Description
`Status`	`string`	The status of the queries to list: • `queued`: Show only queued searches. • `running`: Show only running searches. • `all` or `*` (default): Show both queued and running searches.
`User`	`string`	The names of the user whose queries you want to list.
`InvestigationReason`	`string`	The reason for running the `.show queries` command, which is added to the Cribl Search audit log.

Returns

Returns the details of the specified searches.

Examples

Each of these queries demonstrates runnable syntax, but this command will return results only if you have pending queries in a state matching the `Status` parameter.

Show all running searches:

```
.show running queries with(reason = "Checking what's taking so long.")
```

Copy | Run in Cribl Search

Show searches queued by specific users:

```
.show queued queries by user "John*" with(reason = "Just checking.")
```

Copy

Show searches being run by specific users:

```
.show running queries by user "John*" with(reason = "Thinking of resources.")
```

Copy

10.4. Statements

A list of special statements supported by Cribl Search.

A statement in Cribl Search is a special keyword that sets options (`set`) or assigns names to expressions (`let`).

Name	Description
`let`	Assigns a referenceable name to a value, expression, or an entire subquery.
`set`	Configures options that affect one or more searches.

10.4.1. Let

The `let` statement assigns a name to a value, expression, or an entire query.

You can then reference that name in the context of the same search. This allows you to write more complex queries that consist of several stages and can reuse their results.

`let` statements are often used together with the `join` and `union` operators.



You can also perform unions and joins by using inline subqueries.

Syntax

```
let Name = Expression;
```

Each `let` statement needs to end with a semicolon (`;`).

Arguments

- **Name**: String. The name to assign to the **Expression**. Spaces ( ), and the keyword `root` are not allowed.
- **Expression**: The value to assign to the **Name**. Can be a single value, an expression, or an entire query.

Examples

Return no more than 1000 events.

```
let x = 1000;

dataset="cribl_search_sample"
 | limit x
```

Copy | Run in Cribl Search

Find all VPC flow logs in the `cribl_search_sample` Dataset that are DNS-related requests.

```
// returns 1 record (a scalar)
let dns = dataset="cribl_lookups" lookup_table="service_names_port_numbers" service_r

dataset="cribl_search_sample" dataSource='vpcflowlogs'
 | where dstport == dns.port_number
```

Copy Run in Cribl Search

Join internal Cribl logs related to HTTP requests with a Dataset containing HTTP status codes.

```
let codeMap = dataset=$vt_dummy event < 4
 | extend x =
   case(event == 0, dynamic({"status": 200, "text": 'OK'}),
        event == 1, dynamic({"status": 308, "text": 'Redirect'}),
        event == 2, dynamic({"status": 404, "text": 'Not found'}),
                    dynamic({"status": 500, "text": 'Internal Error'}))
 | project status=x.status, statusText=x.text;

dataset="cribl_internal_logs" method status
 | where isnotnull(status)
 | join kind=leftouter codeMap on status
```

Copy Run in Cribl Search

Calculate the 95th percentile of requests in the Cribl internal logs.

```
let num_percentile = 95;

let response_time_percentile = dataset="cribl_internal_logs" method status response_t
 | summarize total=count(), response_time=percentile(response_time, num_percentile);

let long_responses = dataset="cribl_internal_logs" method status response_time
 | where response_time > response_time_percentile.response_time
 | summarize count=count(), maxResponseTime=max(response_time);

print strcat('There are ', long_responses.count, ' requests in the ', num_percentile,
```

Copy Run in Cribl Search

10.4.2. Set

The `set` statement configures options that affect one or more searches.

```
// return no more than 1000 results in this search
set max_results_per_search=1000;
```

Copy | Run in Cribl Search

Purpose

Use `set` statements to control search behavior. For example, you can configure how long the current search can run before getting canceled, or set the default maximum time range for each of your searches.



Use `set` statements carefully and always make sure you understand the option that you want to change. If in doubt, read the docs, or contact the Cribl Support team.

To see which `set`-statement options affect your account at the moment, use the `.show options` command.

To disable `set`-statement options, use the `.clear options` command.

Syntax

```
set [Scope:]OptionName=OptionValue [, ...];
```

Each `set` statement needs to end with a semicolon `;`.

Parameters

Parameter	Description
`Scope`	The scope to which the option will apply. Can be either `user:` or `global:` (only when set by Admin Search Members). When omitted, affects only the current search.
`OptionName`	The name of the option to set.
`OptionValue`	The value of the option to set.

Scope of Set Statements

You can apply `set` statements in the following scopes:

Scope	Example	Impact	Can be set by
Single search	`set max_results_per_search=1000;`	Applies only to the current search. Depending on the permissions, can override user settings and Usage Group settings.	All Search Members
`user:`	`set user:max_results_per_search=1000;`	Applies to every search by the current user. Depending on the permissions, can override Usage Group settings.	All Search Members
`global:`	`set global:max_results_per_search=1000;`	Applies to every search by every member of the Usage Group.	Admin Search Members

Depending on your permissions and the options specified, `set` statements can also override certain Cribl Search settings, limits, and Usage Group configurations.

A single `set` statement can apply multiple options to multiple scopes at once. For example:

```
set global:max_executors=10, user:max_bytes_read_per_search=536870912, max_resu
```

Copy

Returns

Search results are returned as usual, but with the specified settings applied.

The settings that were applied to a search are always logged in that search's details, under **set options**.

Examples

Set the maximum number of results to 100,000.

```
set max_results_per_search=100000;

dataset="cribl_internal_logs"
```

Copy | Run in Cribl Search

If an analogous search ran within the last 10 minutes, automatically reuse its results instead of running the same search again.

```
set allow_previous_results="10min";

dataset="cribl_internal_logs" method status
| summarize count() by method, status
```

Copy | Run in Cribl Search

Available Set-Statement Options

The `set` statement accepts the following options.

allow_incomplete_results

Enable querying uncompleted search jobs.

When you set this property to `true`, the `$vt_results` virtual table will enable querying the results of jobs that ended in `failed` or `canceled` status. (By default, `$vt_results` allows reading only successfully completed jobs.)

When you set this property to `true`, the output will contain a `jobStatus` property, in addition to the original data fields. Usage example:

```
set allow_incomplete_results=true;
dataset="$vt_results" jobId=<some-ID>
```

Copy

You can set this option to one of the following:

Setting	Behavior
`false` or `"no"` or `0` (default)	Disable querying `failed` or `canceled` search results.

Setting	Behavior
`true` or `"yes"` or `1`	Enable querying `failed` or `canceled` search results.

allow_previous_results

Automatically reuse the results of analogous searches that were run in the specified time range.

You can set this option to a time string such as `1h` (1 hour), `10min` (10 minutes), or `10` (10 seconds).

```
// allow this search to reuse analogous search results from the last 10 minutes
set allow_previous_results="10min";

// allow my every search to reuse analogous search results from the last 10 minutes
set user:allow_previous_results="10min";

// (admin only) allow every user to reuse analogous search results from the last 10 m
set global:allow_previous_results="10min";
```

Copy

disable_previews

Disable generating preview of search results.

Certain Cribl Search functions generate preliminary results that get replaced as the search scans more data. This way, you quickly can see initial search results. The `disable_previews` option suppresses generating those previews, which, in certain scenarios, may speed the overall search up and get you the final results faster.

You can set this option to one of the following:

Setting	Behavior
`false` or `"no"` or `0` (default)	If possible, generate the preview of the search results.
`true` or `"yes"` or `1`	Don't generate the preview of the search results.

For example:

```
// in this search, don't generate the preview of the results
set disable_previews=true;

// in all your searches, don't generate previews
set user:disable_previews=1;

// (admin only) disable previews for all searches run by any user
set global:disable_previews="yes";
```

Copy

job_visibility

Set the default visibility of the search and its results.

You can also change the visibility of a search in **Actions** > **Results Access**.

You can set this option to one of the following:

Setting	Who can see the search and its results
`private` (default)	Only you and the Search Admin. Same as **Private**.
`public`	Everyone in your organization. Same as **All Members**.

For example:

```
// make this search visible only to you and the Search Admin
set job_visibility="private";

// make all your searches visible to everyone
set user:job_visibility="public";

// (admin only) make all users' searches visible to everyone
set global:job_visibility="public";
```

Copy

lakehouse

When querying a Cribl Lake Dataset linked to a Lakehouse, define whether the search should use the Lakehouse or not.

Setting	Behavior
on (default)	Use only Lakehouse-cached data.
off	Don't use Lakehouse data. Run the search at regular speed.

For example:

```
// query the Lake Dataset using Lakehouse caching
set lakehouse="on";

// query the Dataset without Lakehouse caching
set lakehouse="off";
```

Copy

max_bytes_read_per_search

Maximum number of bytes that can be read per search. When the limit is exceeded, the search gets canceled.

You can set this option to an int (as bytes), for example:

```
// allow this search to read up to 512MB of data
set max_bytes_read_per_search=536870912;

// allow my every search to read up to 512MB of data
set user:max_bytes_read_per_search=536870912;

// (admin only) allow every user to read up to 512MB of data per search
set global:max_bytes_read_per_search=536870912;
```

Copy

Depending on your permissions, this option can override the **Byte limit** of the matching Usage Group:

Search Member Type	Permissions
Admin	Can increase or decrease the **Byte limit** value.
Editor or User	Can only decrease the **Byte limit** value.

max_earliest_relative

How far back in time the search can go.

You can set this option to a negative time string, for example: `-1h` (1 hour), `-10min` (10 minutes), or `-10` (10 seconds).

```
// allow this search to scan data as old as 90 days
set max_earliest_relative=-90d;

// allow my every search to scan data as old as 90 days
set user:max_earliest_relative=-90d;

// (admin only) allow all users to scan data as old as 90 days
set global:max_earliest_relative=-90d;
```

Copy

Depending on your permissions, this option can override the **Earliest relative time range** of the matching Usage Group.

Search Member Type	Permissions
Admin	Can increase or decrease the value of the **Earliest relative time range**.
Editor or User	Can only decrease the value of the **Earliest relative time range**.

max_executors

Maximum number of executors that the search can use.



More executors don't necessarily cause better performance. Sometimes, more executors can lead to an overload of the coordinator, slowing search execution down.

You can set this option to an int or to the string `"auto"`.

With `max_executors="auto"`, the search starts with a few executors and then dynamically scales these up or down based on collected throughput metrics. In the **Details** of the search, you'll see an additional metric showing the **average** number of executors.

```
// allow this search to invoke up to 5 executors
set max_executors=5;
// automatically adjust the number of executors for this search
set max_executors="auto";

// allow my every search to invoke up to 5 executors
set user:max_executors=5;
// automatically adjust the number of executors for my every search
set user:max_executors="auto";

// (admin only) allow every user to invoke up to 5 executors per search
set global:max_executors=5;
// (admin only) automatically adjust the number of executors for all users
set global:max_executors="auto";
```

Copy

Depending on your permissions, this option can override the **Executors limit** of the matching Usage Group.

Search Member Type	Permissions
Admin	Can increase or decrease the value of the **Executors limit**.
Editor or User	Can only decrease the value of the **Executors limit**.

max_results_per_search

Maximum number of events returned by the search.

You can set this option to an int between `1` and `10000000`, for example:

```
// allow this search to store up to 100,000 result events
set max_results_per_search=100000;

// allow my every search to store up to 100,000 result events
set user:max_results_per_search=100000;

// (admin only) allow every user to store up to 100,000 result events per search
set global:max_results_per_search=100000;
```

Copy

Depending on your permissions, this option can override the **Results limit** of the matching Usage Group. For details, see: Set the Limits of Search Results.

Search Member Type	Permissions
Admin	Can increase or decrease the value of the **Results limit**.
Editor or User	Can only decrease the value of the **Results limit**.

max_running_time_per_search

How long the search can run. When the limit is exceeded, the search gets canceled.

You can set this option to an int (as seconds), for example:

```
// allow this search to run for up to 1 hour
set max_running_time_per_search=3600;

// allow my every search to run for up to 1 hour
set user:max_running_time_per_search=3600;

// (admin only) allow every search by every user to run for up to 1 hour
set global:max_running_time_per_search=3600;
```

Copy

Depending on your permissions, this option can override the **Running time limit** of the matching Usage Group.

Search Member Type	Permissions
Admin	Can increase or decrease the value of the **Running time limit**.
Editor or User	Can only decrease the value of the **Running time limit**.

When a search does exceed the `max_running_time_per_search`, it can take up to an additional 10 seconds for the job to fully terminate. This ensures proper collection of internal logs and metrics during a graceful shutdown.

max_scheduled_searches_per_user

Maximum number of concurrent scheduled searches per user. If the user tries to execute more scheduled searches than specified by this limit, the searches get queued.

You can set this option to an int, for example:

```
// allow yourself to run up to 2 scheduled searches at once
set max_scheduled_searches_per_user=2;
// or:
set user:max_scheduled_searches_per_user=2;

// (admin only) allow every user to run up to 2 scheduled searches at once
set global:max_scheduled_searches_per_user=2;
```

Copy

Depending on your permissions, this option can override the **User concurrent scheduled search limit** of the matching Usage Group:

Search Member Type	Permissions
Admin	Can increase or decrease the value of the **User concurrent scheduled search limit**.
Editor or User	Can only decrease the value of the **User concurrent scheduled search limit**.

max_searches_per_user

Maximum number of concurrent ad-hoc searches per user. If the user tries to execute more queries than specified by this limit, the searches get queued.

You can set this option to an int, for example:

```
// allow yourself to run up to 3 searches at the same time
set max_searches_per_user=3;
// or:
set user:max_searches_per_user=3;

// (admin only) allow every user to run up to 3 searches at the same time
set global:max_searches_per_user=3;
```

Copy

Depending on your permissions, this option can override the **User concurrent ad hoc search limit** of the matching Usage Group:

Search Member Type	Permissions
Admin	Can increase or decrease the value of the **User concurrent ad hoc search limit**.
Editor or User	Can only decrease the value of the **User concurrent ad hoc search limit**.

max_timerange_width

Maximum time range (the value between `earliest` and `latest`) that a single search can cover.

You can set this option to a time string such as `1h` (1 hour), `10min` (10 minutes), or `10` (10 seconds).

```
// allow this search to scan up to 2 hours of data
set max_timerange_width=7200;

// allow my every search to scan up to 2 hours of data
set user:max_timerange_width=7200;

// (admin only) allow every user to scan up to 2 hours of data per search
set global:max_timerange_width=7200;
```

Copy

Depending on your permissions, this option can override the **Time range limit** of the matching Usage Group.

Search Member Type	Permissions
Admin	Can increase or decrease the value of the **Time range limit**.
Editor or User	Can only decrease the value of the **Time range limit**.

usage_group

Override the current Usage Group limits with the limits of another Usage Group.

Only Admin Search Members can use this option.

You can set this option to a string (the name of the Usage Group to use), for example:

```
// run this search as a Member of `someOtherGroup`
set usage_group="someOtherGroup";
```

Copy

10.5. Functions

A comprehensive list of all functions supported in Cribl Search, grouped by category.

Context Functions

Context functions return contextual information about your search.

Name	Description
`createdTime()`	Time when the search was created, in seconds.
`earliestTime()`	Beginning of the search's time range, in seconds.
`jobID()`	Unique search identifier.
`latestTime()`	End of the search's time range, in seconds.
`query()`	Query string.
`user()`	Username of the user who created the search.
`displayUsername()`	Friendly display name (typically first + last name) of the user who created the search.

Cribl Functions

Cribl functions can be used together with the `summarize`, `eventstats`, and `timestats` operators to aggregate your data. We refer to those additional functions as **Cribl** functions, since they're specific to Cribl Search.

Name	Description
`findearliest`	Returns the earliest value of an expression across the group.
`findearliestif`	Returns the earliest value of an expression across the group for which a predicate evalutes to `true`.
`findfirst`	Returns the first observed value of an expression across the group.
`findfirstif`	Returns the first observed value of an expression across the group for which a predicate evalutes to `true`.

Name	Description
findlast	Returns the last observed value of an expression across the group.
findlastif	Returns the last observed value of an expression across the group for which a predicate evalutes to true.
findlatest	Returns the latest value of an expression across the group.
findlatestif	Returns the latest value of an expression across the group for which a predicate evalutes to true.
list	Returns the list of values of an expression across the group
median	Returns the middle value of an expression across the group.
medianif	Returns the middle value of an expression across the group for which a predicate evalutes to true.
persecond	Returns the per-second rate of an expression across the group
persecondif	Returns the per-second rate of an expression across the group for which a predicate evalutes to true.
rate	Returns the rate observed value of an expression across the group.
rateif	Returns the rate observed value of an expression across the group for which a predicate evalutes to true.
sumsq	Returns the sum of squares of an expression across the group.
sumsqif	Returns the sum of squares of an expression across the group for which a predicate evalutes to true.
values	Returns all of the distinct values of an expression across the group.

Scalar Functions

Scalar functions perform calculations, transformations, or conversions.

Binary Functions

Name	Description
binary_and	Returns a result of the bitwise and operation between two values.

Name	Description
`binary_not`	Returns a bitwise negation of the input value.
`binary_or`	Returns a result of the bitwise or operation of the two values.
`binary_shift_left`	Returns binary shift left operation on a pair of numbers.
`binary_shift_right`	Returns binary shift right operation on a pair of numbers.
`binary_xor`	Returns the bitwise `xor` operation of the two values.
`from_binary_string`	Takes a binary string and returns a number.
`to_binary_string`	Takes a number and returns a binary string.

Conditional Functions

Name	Description
`case`	Evaluates a list of predicates and returns the first result expression whose predicate is satisfied.
`coalesce`	Evaluates a list of expressions and returns the first non-null (or non-empty for string) expression.
`iif` (`iff`)	Evaluates the first argument (the predicate), and returns the value of either the second or third arguments, depending on whether the predicate evaluated to `true` (second) or `false` (third).
`max_of`	Returns the maximum value of several evaluated numeric expressions.
`min_of`	Returns the minimum value of several evaluated numeric expressions.

Conversion Functions

Name	Description
`bin`	Rounds values down to an integer multiple of a given bin size.
`bin_auto`	Rounds values down to a fixed-size bin.
`floor`	Rounds values down to an integer multiple of a given floor size.
`gettype`	Returns the type of the input value.

Name	Description
tobool	Converts the input to a bool value.
todouble (toreal, todecimal)	Converts the input to a double (real, decimal) value.
toint (tolong)	Converts the input to an int (long) value.
tostring	Converts the input to a string value.

DateTime Functions

Name	Description
ago	Subtracts the given timespan from the current UTC clock time.
datetime_add	Calculates a new datetime from a specified datepart multiplied by a specified amount, added to a specified datetime.
datetime_diff	Calculates the number of the specified periods between two datetime values.
datetime_part	Extracts the requested date part as an integer value.
dayofmonth	Returns the integer number representing the day number of the given month.
dayofweek	Returns an integer between 0 and 6 representing the day of the week, beginning on Sunday.
dayofyear	Returns the integer number representing the day number of the given year.
endofday	Returns the end of the day containing the date, shifted by an offset, if provided.
endofmonth	Returns the end of the month containing the date, shifted by an offset, if provided.
endofweek	Returns the end of the week containing the date, shifted by an offset, if provided.
endofyear	Returns the end of the year containing the date, shifted by an offset, if provided.
format_datetime	Formats a datetime according to the provided format.
format_timespan	Formats a timespan according to the provided format.

Name	Description
getmonth (monthofyear)	Gets the month number (1–12) from a datetime.
getyear (yearofyear)	Returns the year part of a datetime.
hourofday	Returns the integer number representing the hour number of the given date.
make_datetime	Converts the specified date and time into a datetime value (Unix time in seconds).
make_timespan	Converts the specified time period into a datetime value (Unix time in seconds).
now	Returns the current UTC clock time as a datetime value (Unix time in seconds).
startofday	Returns the start of the day containing the date, shifted by an offset, if provided.
startofmonth	Returns the start of the month containing the date, shifted by an offset, if provided.
startofweek	Returns the start of the week containing the date, shifted by an offset, if provided.
startofyear	Returns the start of the year containing the date, shifted by an offset, if provided.
strftime (format_time)	Converts a datetime (date) object to a human-readable string.
strptime (parse_time)	Converts a string to a datetime.
todatetime	Converts the input into a datetime value (Unix time in seconds).
totimespan	Converts the input into a number of seconds.
unixtime_microseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime_milliseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime-nanoseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime_seconds_todatetime	Converts the input into a datetime value (Unix time in seconds).

Name	Description
week_of_year	Returns an integer representing the week number.

Dynamic Functions

Dynamic scalar functions allow you to manipulate objects by operating on dynamic values, including dynamic arrays and property bags.

Name	Description
bag_has_key	Checks whether a property bag contains a given key.
bag_keys	Lists all root keys of a property bag.
bag_merge	Merges multiple property bags, discarding duplicate keys.
bag_pack	Creates a property bag from an alternating list of keys and values.
bag_pack_columns	Creates a property bag from a list of columns.
bag_remove_keys	Removes key-value pairs from a property bag.
bag_set_key	Adds or overwrites a key-value pair in a property bag.
bag_zip	Creates a property bag from two dynamic arrays.
make_bag	Creates a property bag from multiple input bags.
make_bag_if	Creates a property bag from those input bags that meet the specified condition.
zip	Merges dynamic arrays, grouping elements by index.

The following string functions support `dynamic` types as well:

Function	Usage with dynamic data types
extractjson(`path,object`)	Uses path to navigate into object.
parse_json(`source`)	Turns a JSON string into a dynamic object.
range(`from,to,step`)	Generates an array of values.

Cryptographic Functions

Name	Description
`encrypt`	Encrypts data with a key managed by a Cribl Stream Worker Group.
`decrypt`	Decrypts data with a key managed by a Cribl Stream Worker Group.

Hash Functions

Name	Description
`hash`	Returns a hash value for the input value.
`hash_combine`	Combines hash values of two or more hashes.
`hash_many`	Returns a combined hash value of multiple values.
`hash_md5`	Returns an MD5 hash value for the input value.
`hash_sha1`	Returns a SHA1 hash value for the input value.
`hash_sha256`	Returns a SHA-256 hash value for the input value.
`hash_xxhash64`	Returns a 64-bit hash value for the input value.

INET Functions

Name	Description
`ipv4_compare`	Compares two IPv4 strings.
`ipv4_is_in_range`	Checks if IPv4 string address is in IPv4-prefix notation range.
`ipv4_is_in_any_range`	Checks whether IPv4 string address is in any of the specified IPv4 address ranges.
`ipv4_is_match`	Matches two IPv4 strings.
`ipv4_is_private`	Checks if IPv4 string address belongs to a set of private network IPs.
`ipv4_netmask_suffix`	Returns the value of the IPv4 netmask suffix from IPv4 string address.
`ipv6_compare`	Compares two IPv6 or IPv4 network address strings.
`ipv6_is_match`	Matches two IPv6 or IPv4 network address strings.

Name	Description
format_bytes	Converts the input into a string that represents data size.
format_ipv4	Parses input with a netmask and returns string representing IPv4 address.
format_ipv4_mask	Parses input with a netmask and returns string representing IPv4 address as CIDR notation.

Mathematical Functions

Name	Description
abs	Calculates the absolute value of the input.
acos	Calculates the angle whose cosine is the specified number.
asin	Calculates the angle whose sine is the specified number.
atan	Returns the angle whose tangent is the specified number.
atan2	Calculates the angle, in radians, between the positive x-axis and the ray from the origin to the point (y, x).
beta_cdf	Returns the standard cumulative beta distribution function.
beta_inv	Returns the inverse of the beta cumulative probability beta density function.
beta_pdf	Returns the probability density beta function.
ceil	Rounds up a numeric expression's value to the nearest integer.
ceiling	Rounds up a numeric expression's value to the nearest integer.
cos	Returns the cosine function.
cot	Calculates the trigonometric cotangent of the specified angle, in radians.
degrees	Converts angle value in radians into value in degrees.
exp	Calculates the base-e exponential function of x.
exp2	Calculates the base-2 exponential function of x.
exp10	Calculates the base-10 exponential function of x.
gamma	Computes gamma function.

Name	Description
isfinite	Returns whether input is a finite value.
isinf	Returns whether input is an infinite value.
isnan	Returns whether input is Not-a-Number (NaN) value.
log	Returns the natural logarithm function.
log2	Returns the (base-2) logarithm function.
log10	Returns the common (base-10) logarithm function.
loggamma	Computes log of absolute value of the loggamma function.
not	Reverses the value of its boolean argument.
pi	Returns the constant value of Pi.
pow	Returns a result of raising to power.
radians	Converts angle value in degrees into value in radians.
rand	Returns a random number.
range	Generates a dynamic array, holding a series of equally spaced values.
round	Returns the rounded source to the specified precision.
sign	Returns the sign of a numeric expression.
sin	Returns the sine of a numeric expression.
sqrt	Returns the square root function.
tan	Returns the tangent function.

String Functions

Name	Description
base64_decode_toarray	Decodes a base64 string to an array of long values.
base64_decode_tostring	Decodes a base64 string to a UTF-8 string.

Name	Description
base64_encode_fromarray	Encodes a base64 string from a bytes array.
base64_encode_tostring	Encodes a string as base64 string.
countof	Counts occurrences of a substring in a string.
extract	Gets a match for an RE2 regular expression from a source string.
extract_all	Gets all matches for an RE2 regular expression from a source string.
extract_json	Gets a specified element out of a JSON text using a path expression.
has_any_index	Gets a match for an RE2 regular expression from a source string.
indexof	Reports the zero-based index of the first occurrence of a specified string within the input string.
isempty	Returns `true` if the argument is an empty string or is null.
isnotempty (notempty)	Returns `true` if the argument isn't an empty string, and it isn't null.
isnotnull (notnull)	Returns `true` if the argument is not null.
isnull	Indicates whether the argument evaluates to a null value.
match_regex	Searches a text string for a specific pattern defined by a regular expression.
parse_csv	Splits a given string representing a single record of comma-separated values.
parse_ipv4	Converts IPv4 string to long (signed 64-bit) number representation in big-endian order.
parse_ipv4_mask	Converts the input string of IPv4 and netmask to a signed, 64-bit wide, long number representation in big-endian order.
parse_ipv6	Converts IPv6 or IPv4 string to a canonical IPv6 string representation.
parse_ipv6_mask	Converts IPv6/IPv4 string and netmask to a canonical IPv6 string representation.
parse_json (todynamic)	Interprets a string as a JSON value and returns the value as dynamic.
parse_url	Parses an absolute URL string and returns a dynamic object that contains URL parts.

Name	Description
parse_urlquery	Returns a dynamic object that contains the Query parameters.
parse_version	Converts the input string representation of version to a comparable decimal number.
replace_regex	Replaces all RE2 regular expression matches with another string.
reverse	Reverses the order of the input string.
split	Splits a given string according to a given delimiter.
strcat	Concatenates between 1 and 64 arguments.
strcat_delim	Concatenates between 2 and 64 arguments, with a delimiter.
strcmp	Compares two strings.
strlen	Returns the length, in characters, of the input string.
strrep	Repeats given string specified number of times.
substring	Extracts a substring from a source string starting from some index to the end of the string.
tolower	Converts a string to lower case.
toupper	Converts a string to upper case.
translate	Replaces a set of characters with another set of characters in a given string.
trim	Removes all leading and trailing matches of the specified string or regular expression.
trim_end	Removes trailing match of the specified regular expression.
trim_start	Removes leading match of the specified regular expression.
url_decode	Converts encoded URL into a to regular URL representation.
url_encode	Converts characters of the input URL into a format that can be transmitted over the Internet.

Statistical Functions

Name	Description
avg	Calculates the average across the group.
avgif	Calculates the average across the group where a predicate evaluates to `true`.
count	Counts events per summarization group.
countif	Counts events based on a predicate.
dcount	Calculates an estimate of the number of distinct values.
dcountif	Calculates an estimate of the number of distinct values for those rows where a predicate evaluates to `true`.
max	Finds the maximum value across the group.
maxif	Finds the maximum value for which a predicate evaluates to `true`.
min	Finds the minimum value across the group.
minif	Finds the minimum value which a predicate evaluates to `true`.
percentile	Returns an estimate for the specified nearest-rank percentile of the population defined.
stdev	Calculates the standard deviation of an expression across the group.
stdevif	Calculates the standard deviation of an expression which a predicate evaluates to `true`.
stdevp	Calculates the standard deviation of an expression across the group, considering the group as a population.
sum	Calculates the sum of an expression across the group.
sumif	Calculates the sum of an expression for which a predicate evaluates to `true`.
variance	Calculates the variance of an expression.
varianceif	Calculates the variance of an expression for which a predicate evaluates to `true`.
variancep	Calculates the variance of an expression across the group, considering the group as a population.

Window Functions

Name	Description
`next`	Returns the value of a specific field in a subsequent row.
`prev`	Returns the value of a specific field in a previous row.
`row_cumsum`	Calculates the cumulative sum for a specified field across all previous rows.
`row_number`	Assigns a unique row number to each row within the results.
`row_rank_dense`	Assigns a unique numerical position (rank) to each row within the results
`row_rank_min`	Assigns a minimal numerical position (rank) to each row within the results
`row_window_session`	Identifies the value at the beginning of each session for a specified field within the results.

10.5.1. Context Functions

Context functions return contextual information about your search.

Name	Description
`createdTime()`	Time when the search was created, in seconds.
`earliestTime()`	Beginning of the search's time range, in seconds.
`jobID()`	Unique search identifier.
`latestTime()`	End of the search's time range, in seconds.
`query()`	Query string.
`user()`	Username of the user who created the search.
`displayUsername()`	Friendly display name (typically first + last name) of the user who created the search.

Examples

Return multiple details about a search:

```
dataset="cribl_search_sample" action=="*"
| summarize eventsPerSecond = (count() / (latestTime() - earliestTime())) by action
| project action, eventsPerSecond, reportTime=createdTime(), reportId=jobID(), report
```

Copy | Run in Cribl Search

Reference the search's time range:

```
dataset=myDataset
| summarize eventsPerSecond = (count() / (latestTime() - earliestTime())) by host
```

Copy

Filter events by the current user:

```
dataset=myDataset
| where user == user()
```

Copy

Enrich results with search information:

```
dataset=myDataset
| summarize NumTransactions=count(), Total=sum(UnitPrice * NumUnits) by Fruit, StartO
| extend query=query(), searchid=jobID()
```

Copy

10.5.2. Cribl Functions

Cribl Search supports additional functions that you can use together with the `summarize`, `eventstats`, and `timestats` operators to aggregate your data. We refer to those additional functions as **Cribl** functions, since they're specific to Cribl Search.

Cribl Search supports the following Cribl functions:

Name	Description
`findearliest`	Returns the earliest value of an expression across the group.
`findearliestif`	Returns the earliest value of an expression across the group for which a predicate evalutes to `true`.
`findfirst`	Returns the first observed value of an expression across the group.
`findfirstif`	Returns the first observed value of an expression across the group for which a predicate evalutes to `true`.
`findlast`	Returns the last observed value of an expression across the group.
`findlastif`	Returns the last observed value of an expression across the group for which a predicate evalutes to `true`.
`findlatest`	Returns the latest value of an expression across the group.
`findlatestif`	Returns the latest value of an expression across the group for which a predicate evalutes to `true`.
`list`	Returns the list of values of an expression across the group
`median`	Returns the middle value of an expression across the group.
`medianif`	Returns the middle value of an expression across the group for which a predicate evalutes to `true`.
`persecond`	Returns the per-second rate of an expression across the group
`persecondif`	Returns the per-second rate of an expression across the group for which a predicate evalutes to `true`.
`rate`	Returns the rate observed value of an expression across the group.
`rateif`	Returns the rate observed value of an expression across the group for which a predicate evalutes to `true`.

Name	Description
`sumsq`	Returns the sum of squares of an expression across the group.
`sumsqif`	Returns the sum of squares of an expression across the group for which a predicate evalutes to `true`.
`values`	Returns all of the distinct values of an expression across the group.

10.5.2.1. FINDEARLIEST

The `findearliest` aggregation function returns the earliest value (based on `_time`) of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
findearliest( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Examples

Return the earliest event by `_time`:

```
dataset="cribl_search_sample"
| summarize findearliest(_time)
```

Copy | Run in Cribl Search

Return the earliest event by a custom field's value:

```
dataset=myDataset
| summarize findearliest(receiptTime)
```

Copy

10.5.2.2. FINDEARLIESTIF

The `findearliestif` aggregation function returns the earliest value (based on `_time`) of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
findearliestif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Examples

This example returns the 10 earliest events with an `ACCEPT` action:

```
dataset="cribl_search_sample" | limit 10
| eventstats startTime=findearliestif(_time, action=="ACCEPT")
| extend timeDelta=_time-startTime
```

Copy | Run in Cribl Search

This example returns the earliest date for all products that have more than 4 letters.

```
dataset=myDataset
| summarize findearliestif(date, strlen(product) > 4)
```

Copy

10.5.2.3. FINDFIRST

The `findfirst` aggregation function returns the first observed non-`null` value of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
findfirst( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Usage

To find an earliest event with respect to the `_time` field, instead use `findearliest`.

You can use `findfirst` after a `sort` or `order` operator when sorting by a non-time field. Once events are sorted, this function acts much like `min`.

Examples

Here is a basic example:

```
dataset=myDataset
| summarize findfirst(channel)
```

Copy

This example effectively finds `min(status)` and `max(status)`:

```
dataset="cribl_internal_logs" status=/[0-9]+/
| order by status asc
| summarize x = findfirst(status), y = findlast(status)
```

Copy Run in Cribl Search

10.5.2.4. FINDFIRSTIF

The `findfirstif` aggregation function returns the first observed non-`null` value of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
findfirstif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Usage

To find an earliest event with respect to the `_time` field, instead use `findearliestif`.

You can use `findfirstif` after a `sort` or `order` operator when sorting by a non-time field. Once events are sorted, this function acts much like `minif`.

Example

This example returns the birthday for all names that have more than 4 letters.

```
dataset=myDataset
| summarize findfirstif(day_of_birth, strlen(name) > 4)
```

Copy

10.5.2.5. FINDLAST

The `findlast` aggregation function returns the last observed non-`null` value of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
findlast( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Usage

To find a latest event with respect to the `_time` field, instead use `findlatest`.

You can use `findlast` after a `sort` or `order` operator when sorting by a non-time field. Once events are sorted, this function acts much like `max`.

Examples

Here is a basic example:

```
dataset=myDataset
| summarize findlast(channel)
```

Copy

This example effectively finds `min(status)` and `max(status)`:

```
dataset="cribl_internal_logs" status=/[0-9]+/
| order by status asc
| summarize x = findfirst(status), y = findlast(status)
```

Copy Run in Cribl Search

10.5.2.6. FINDLASTIF

The `findlastif` aggregation function returns the last observed non-`null` value of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
findlastif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Usage

To find a latest event with respect to the `_time` field, instead use `findlatestif`.

You can use `findlastif` after a `sort` or `order` operator when sorting by a non-time field. Once events are sorted, this function acts much like `maxif`.

Example

This example returns the birthday for all names that have more than 4 letters.

```
dataset=myDataset
| summarize findlastif(day_of_birth, strlen(name) > 4)
```

Copy

10.5.2.7. FINDLATEST

The `findlatest` aggregation function returns the latest value (based on `_time`) of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
findlatest( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Example

```
dataset=myDataset
| summarize findlatest(receiptTime)
```

Copy

10.5.2.8. FINDLATESTIF

The `findlatestif` aggregation function returns the latest value (based on `_time`) of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
findlatestif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Example

This example returns the latest date for all products that have more than 4 letters.

```
dataset=myDataset
| summarize findlatestif(date, strlen(product) > 4)
```

Copy

10.5.2.9. List

The `list` aggregation function returns the list of values of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
list( Expression [, Max ] )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Max**: An integer that limits the number of values returned. The default is `100`. If set to `0`, all values are returned.

Examples

This example lists methods (HTTP verbs) on API requests, up to a limit:

```
dataset="cribl_search_sample" dataSource=access*
| limit 1000
| summarize list(request_method)
```

Copy | Run in Cribl Search

10.5.2.10. Median

The `median` aggregation function returns the middle value of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
median( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Example

```
dataset=myDataset
| summarize median(age)
```

Copy

10.5.2.11. MEDIANIF

The `medianif` aggregation function returns the middle value of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
medianif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Example

```
dataset=myDataset
| summarize medianif(data, host=="Cribl.local")
```

Copy

10.5.2.12. PERSECOND

The `persecond` aggregation function returns the per second rate (based on `_time`) of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
persecond( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Example

```
dataset=myDataset
| summarize persecond(transactions)
```

Copy

10.5.2.13. PERSECONDIF

The `persecondif` aggregation function returns the per second rate (based on `_time`) of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
persecondif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Example

```
dataset=myDataset
| summarize persecondif(transactions, location=="town")
```

Copy

10.5.2.14. Rate

The `rate` aggregation function returns the rate (based on `_time`) observed value of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
rate( Expression, Time )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Time**: Unit of time, specified with a positive integer and time period. Supports weeks (`w`), days (`d`), hours (`h`), minutes (`m`), and seconds (`s`). For example, one day would be `1d`.

Example

```
dataset=myDataset
| summarize rate(goats, "5m")
```

Copy

10.5.2.15. RATEIF

The `rateif` aggregation function returns the rate (based on `_time`) observed value of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
rateif( Expression, Time, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Time**: Unit of time, specified with a positive integer and time period. Supports weeks (`w`), days (`d`), hours (`h`), minutes (`m`), and seconds (`s`). For example, one day would be `1d`.
- **Predicate**: Expression that will be used to filter rows.

Example

```
dataset=myDataset
| summarize rateif(goats, "5s", host=="Cribl.local")
```

Copy

10.5.2.16. SUMSQ

The `sumsq` aggregation function returns the sum of squares of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
sumsq( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Example

```
dataset=myDataset
| summarize sumsq(goats)
```

Copy

10.5.2.17. SUMSQIF

The `sumsqif` aggregation function returns the sum of squares of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
sumsqif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Example

```
dataset=myDataset
| summarize sumsqif(goats, host=="Cribl.local")
```

Copy

10.5.2.18. Values

The `values` aggregation function returns all of the distinct values of **Expression** across the group. This allows you to quickly identify and understand all the values a field has in your data.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
values( Expression [, Max [, ErrorRate] ] )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Max**: An integer that limits the number of values returned. The default is `0` where all distinct values are returned.
- **ErrorRate**: Controls how accurately the function counts distinct values. Range is `0–1`. The default value is `0.01`. Higher values allow higher error rates (fewer unique values recognized), with the offsetting benefit of less memory usage.

Results

The response field is an array of unique values (a "set"), up to either a specified limit or the default limit of `100`. There is no guarantee that this array's elements will be returned in any particular order.

Examples

This example lists unique methods (HTTP verbs) on API requests, up to the supplied limit:

```
dataset="cribl_search_sample" dataSource=access*
| limit 1000
| summarize values(request_method)
```

Copy | Run in Cribl Search

10.5.3. Scalar Functions

A list of all scalar functions supported by Cribl Search, grouped by category.

Binary Functions

Name	Description
`binary_and`	Returns a result of the bitwise `and` operation between two values.
`binary_not`	Returns a bitwise negation of the input value.
`binary_or`	Returns a result of the bitwise or operation of the two values.
`binary_shift_left`	Returns binary shift left operation on a pair of numbers.
`binary_shift_right`	Returns binary shift right operation on a pair of numbers.
`binary_xor`	Returns the bitwise `xor` operation of the two values.
`from_binary_string`	Takes a binary string and returns a number.
`to_binary_string`	Takes a number and returns a binary string.

Conditional Functions

Name	Description
`case`	Evaluates a list of predicates and returns the first result expression whose predicate is satisfied.
`coalesce`	Evaluates a list of expressions and returns the first non-null (or non-empty for string) expression.
`iif` (`iff`)	Evaluates the first argument (the predicate), and returns the value of either the second or third arguments, depending on whether the predicate evaluated to `true` (second) or `false` (third).
`max_of`	Returns the maximum value of several evaluated numeric expressions.
`min_of`	Returns the minimum value of several evaluated numeric expressions.

Conversion Functions

Name	Description
bin	Rounds values down to an integer multiple of a given bin size.
bin_auto	Rounds values down to a fixed-size bin.
floor	Rounds values down to an integer multiple of a given floor size.
gettype	Returns the type of the input value.
tobool	Converts the input to a bool value.
todouble (toreal, todecimal)	Converts the input to a double (real, decimal) value.
toint (tolong)	Converts the input to an int (long) value.
tostring	Converts the input to a string value.

DateTime Functions

Name	Description
ago	Subtracts the given timespan from the current UTC clock time.
datetime_add	Calculates a new datetime from a specified datepart multiplied by a specified amount, added to a specified datetime.
datetime_diff	Calculates the number of the specified periods between two datetime values.
datetime_part	Extracts the requested date part as an integer value.
dayofmonth	Returns the integer number representing the day number of the given month.
dayofweek	Returns an integer between 0 and 6 representing the day of the week, beginning on Sunday.
dayofyear	Returns the integer number representing the day number of the given year.
endofday	Returns the end of the day containing the date, shifted by an offset, if provided.

Name	Description
endofmonth	Returns the end of the month containing the date, shifted by an offset, if provided.
endofweek	Returns the end of the week containing the date, shifted by an offset, if provided.
endofyear	Returns the end of the year containing the date, shifted by an offset, if provided.
format_datetime	Formats a datetime according to the provided format.
format_timespan	Formats a timespan according to the provided format.
getmonth (monthofyear)	Gets the month number (1–12) from a datetime.
getyear (yearofyear)	Returns the year part of a datetime.
hourofday	Returns the integer number representing the hour number of the given date.
make_datetime	Converts the specified date and time into a datetime value (Unix time in seconds).
make_timespan	Converts the specified time period into a datetime value (Unix time in seconds).
now	Returns the current UTC clock time as a datetime value (Unix time in seconds).
startofday	Returns the start of the day containing the date, shifted by an offset, if provided.
startofmonth	Returns the start of the month containing the date, shifted by an offset, if provided.
startofweek	Returns the start of the week containing the date, shifted by an offset, if provided.
startofyear	Returns the start of the year containing the date, shifted by an offset, if provided.
strftime (format_time)	Converts a datetime (date) object to a human-readable string.
strptime (parse_time)	Converts a string to a datetime.
todatetime	Converts the input into a datetime value (Unix time in seconds).

Name	Description
totimespan	Converts the input into a number of seconds.
unixtime_microseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime_milliseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime-nanoseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime_seconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
week_of_year	Returns an integer representing the week number.

Dynamic Functions

Dynamic scalar functions allow you to manipulate objects by operating on dynamic values, including dynamic arrays and property bags.

Name	Description
bag_has_key	Checks whether a property bag contains a given key.
bag_keys	Lists all root keys of a property bag.
bag_merge	Merges multiple property bags, discarding duplicate keys.
bag_pack	Creates a property bag from an alternating list of keys and values.
bag_pack_columns	Creates a property bag from a list of columns.
bag_remove_keys	Removes key-value pairs from a property bag.
bag_set_key	Adds or overwrites a key-value pair in a property bag.
bag_zip	Creates a property bag from two dynamic arrays.
make_bag	Creates a property bag from multiple input bags.
make_bag_if	Creates a property bag from those input bags that meet the specified condition.
zip	Merges dynamic arrays, grouping elements by index.

The following string functions support `dynamic` types as well:

Function	Usage with dynamic data types
`extractjson(`path,object`)`	Uses path to navigate into object.
`parse_json(`source`)`	Turns a JSON string into a dynamic object.
`range(`from,to,step`)`	Generates an array of values.

Cryptographic Functions

Name	Description
`encrypt`	Encrypts data with a key managed by a Cribl Stream Worker Group.
`decrypt`	Decrypts data with a key managed by a Cribl Stream Worker Group.

Hash Functions

Name	Description
`hash`	Returns a hash value for the input value.
`hash_combine`	Combines hash values of two or more hashes.
`hash_many`	Returns a combined hash value of multiple values.
`hash_md5`	Returns an MD5 hash value for the input value.
`hash_sha1`	Returns a SHA1 hash value for the input value.
`hash_sha256`	Returns a SHA-256 hash value for the input value.
`hash_xxhash64`	Returns a 64-bit hash value for the input value.

INET Functions

Name	Description
`ipv4_compare`	Compares two IPv4 strings.

Name	Description
ipv4_is_in_range	Checks if IPv4 string address is in IPv4-prefix notation range.
ipv4_is_in_any_range	Checks whether IPv4 string address is in any of the specified IPv4 address ranges.
ipv4_is_match	Matches two IPv4 strings.
ipv4_is_private	Checks if IPv4 string address belongs to a set of private network IPs.
ipv4_netmask_suffix	Returns the value of the IPv4 netmask suffix from IPv4 string address.
ipv6_compare	Compares two IPv6 or IPv4 network address strings.
ipv6_is_match	Matches two IPv6 or IPv4 network address strings.
format_bytes	Converts the input into a string that represents data size.
format_ipv4	Parses input with a netmask and returns string representing IPv4 address.
format_ipv4_mask	Parses input with a netmask and returns string representing IPv4 address as CIDR notation.

Mathematical Functions

Name	Description
abs	Calculates the absolute value of the input.
acos	Calculates the angle whose cosine is the specified number.
asin	Calculates the angle whose sine is the specified number.
atan	Returns the angle whose tangent is the specified number.
atan2	Calculates the angle, in radians, between the positive x-axis and the ray from the origin to the point (y, x).
beta_cdf	Returns the standard cumulative beta distribution function.
beta_inv	Returns the inverse of the beta cumulative probability beta density function.
beta_pdf	Returns the probability density beta function.
ceil	Rounds up a numeric expression's value to the nearest integer.

Name	Description
ceiling	Rounds up a numeric expression's value to the nearest integer.
cos	Returns the cosine function.
cot	Calculates the trigonometric cotangent of the specified angle, in radians.
degrees	Converts angle value in radians into value in degrees.
exp	Calculates the base-e exponential function of x.
exp2	Calculates the base-2 exponential function of x.
exp10	Calculates the base-10 exponential function of x.
gamma	Computes gamma function.
isfinite	Returns whether input is a finite value.
isinf	Returns whether input is an infinite value.
isnan	Returns whether input is Not-a-Number (NaN) value.
log	Returns the natural logarithm function.
log2	Returns the (base-2) logarithm function.
log10	Returns the common (base-10) logarithm function.
loggamma	Computes log of absolute value of the loggamma function.
not	Reverses the value of its boolean argument.
pi	Returns the constant value of Pi.
pow	Returns a result of raising to power.
radians	Converts angle value in degrees into value in radians.
rand	Returns a random number.
range	Generates a dynamic array, holding a series of equally spaced values.
round	Returns the rounded source to the specified precision.
sign	Returns the sign of a numeric expression.

Name	Description
sin	Returns the sine of a numeric expression.
sqrt	Returns the square root function.
tan	Returns the tangent function.

String Functions

Name	Description
base64_decode_toarray	Decodes a base64 string to an array of long values.
base64_decode_tostring	Decodes a base64 string to a UTF-8 string.
base64_encode_fromarray	Encodes a base64 string from a bytes array.
base64_encode_tostring	Encodes a string as base64 string.
countof	Counts occurrences of a substring in a string.
extract	Gets a match for an RE2 regular expression from a source string.
extract_all	Gets all matches for an RE2 regular expression from a source string.
extract_json	Gets a specified element out of a JSON text using a path expression.
has_any_index	Gets a match for an RE2 regular expression from a source string.
indexof	Reports the zero-based index of the first occurrence of a specified string within the input string.
isempty	Returns `true` if the argument is an empty string or is null.
isnotempty (notempty)	Returns `true` if the argument isn't an empty string, and it isn't null.
isnotnull (notnull)	Returns `true` if the argument is not null.
isnull	Indicates whether the argument evaluates to a null value.
match_regex	Searches a text string for a specific pattern defined by a regular expression.
parse_csv	Splits a given string representing a single record of comma-separated values.

Name	Description
parse_ipv4	Converts IPv4 string to long (signed 64-bit) number representation in big-endian order.
parse_ipv4_mask	Converts the input string of IPv4 and netmask to a signed, 64-bit wide, long number representation in big-endian order.
parse_ipv6	Converts IPv6 or IPv4 string to a canonical IPv6 string representation.
parse_ipv6_mask	Converts IPv6/IPv4 string and netmask to a canonical IPv6 string representation.
parse_json (todynamic)	Interprets a string as a JSON value and returns the value as dynamic.
parse_url	Parses an absolute URL string and returns a dynamic object that contains URL parts.
parse_urlquery	Returns a dynamic object that contains the Query parameters.
parse_version	Converts the input string representation of version to a comparable decimal number.
replace_regex	Replaces all RE2 regular expression matches with another string.
reverse	Reverses the order of the input string.
split	Splits a given string according to a given delimiter.
strcat	Concatenates between 1 and 64 arguments.
strcat_delim	Concatenates between 2 and 64 arguments, with a delimiter.
strcmp	Compares two strings.
strlen	Returns the length, in characters, of the input string.
strrep	Repeats given string specified number of times.
substring	Extracts a substring from a source string starting from some index to the end of the string.
tolower	Converts a string to lower case.
toupper	Converts a string to upper case.
translate	Replaces a set of characters with another set of characters in a given string.

Name	Description
`trim`	Removes all leading and trailing matches of the specified string or regular expression.
`trim_end`	Removes trailing match of the specified regular expression.
`trim_start`	Removes leading match of the specified regular expression.
`url_decode`	Converts encoded URL into a to regular URL representation.
`url_encode`	Converts characters of the input URL into a format that can be transmitted over the Internet.

10.5.3.1. Binary Functions

A list of all binary functions supported by Cribl Search.

Name	Description
`binary_and`	Returns a result of the bitwise `and` operation between two values.
`binary_not`	Returns a bitwise negation of the input value.
`binary_or`	Returns a result of the bitwise or operation of the two values.
`binary_shift_left`	Returns binary shift left operation on a pair of numbers.
`binary_shift_right`	Returns binary shift right operation on a pair of numbers.
`binary_xor`	Returns the bitwise `xor` operation of the two values.
`from_binary_string`	Takes a binary string and returns a number.
`to_binary_string`	Takes a number and returns a binary string.

10.5.3.1.1. BINARY_AND

The `binary_and` function returns a result of the bitwise `and` operation between two values.

Syntax

```
binary_and( Number1, Number2 )
```

Arguments

- **NumberX**: A Long number. The Long data type represents a signed, 64-bit wide integer.

Example

```
dataset=myDataset
| where bit != binary_and(101,101)
```

Copy

10.5.3.1.2. BINARY_NOT

The `binary_not` function returns a bitwise negation of the input value.

Syntax

```
binary_not( Number )
```

Arguments

- **Number**: A number.

Example

```
dataset=myDataset
| where bit != binary_not(101)
```

Copy

10.5.3.1.3. BINARY_OR

The `binary_or` function returns a result of the bitwise `or` operation of the two values.

Syntax

```
binary_or( Number1, Number2 )
```

Arguments

- **NumberX**: A Long number. The Long data type represents a signed, 64-bit wide integer.

Example

```
dataset=myDataset
| where bit != binary_or(101,101)
```

Copy

10.5.3.1.4. BINARY_SHIFT_LEFT

The `binary_shift_left` function returns binary shift left operation on a pair of numbers.

Syntax

```
binary_shift_left( Number1, Number2 )
```

Arguments

- **NumberX**: An integer.

Example

```
dataset=myDataset
| where bit != binary_shift_left(101,101)
```

Copy

10.5.3.1.5. BINARY_SHIFT_RIGHT

The `binary_shift_right` function returns binary shift right operation on a pair of numbers.

Syntax

```
binary_shift_right( Number1, Number2 )
```

Arguments

- **NumberX**: A Long number. The Long data type represents a signed, 64-bit wide integer.

Example

```
dataset=myDataset
| where bit != binary_shift_right(101,101)
```

Copy

10.5.3.1.6. BINARY_XOR

The `binary_xor` function returns the bitwise `xor` operation of the two values.

Syntax

```
binary_xor( Number1, Number2 )
```

Arguments

- **NumberX**: A Long number. The Long data type represents a signed, 64-bit wide integer.

Example

```
dataset=myDataset
| where bit != binary_xor(101,101)
```

Copy

10.5.3.1.7. FROM_BINARY_STRING

The `from_binary_string` function takes a binary string and returns a number.

Syntax

```
from_binary_string( String )
```

Arguments

- **String**: A binary value.

Example

```
// returns 42
from_binary_string('101010')
```

Copy

10.5.3.1.8. TO_BINARY_STRING

The `to_binary_string` function takes a number and returns a binary string.

Syntax

```
to_binary_string( Number )
```

Arguments

- **Number**: An integer.

Example

```
// returns '101010'
to_binary_string(42)
```

Copy

10.5.3.2. Conditional Functions

A list of all conditional functions supported by Cribl Search.

Name	Description
`case`	Evaluates a list of predicates and returns the first result expression whose predicate is satisfied.
`coalesce`	Evaluates a list of expressions and returns the first non-null (or non-empty for string) expression.
`iif` (`iff`)	Evaluates the first argument (the predicate), and returns the value of either the second or third arguments, depending on whether the predicate evaluated to `true` (second) or `false` (third).
`max_of`	Returns the maximum value of several evaluated numeric expressions.
`min_of`	Returns the minimum value of several evaluated numeric expressions.

10.5.3.2.1. Case

The `case` function evaluates a list of predicates and returns the first result expression whose predicate is satisfied.

If neither of the predicates return `true`, the result of the last expression (the `else`) is returned. All odd arguments (count starts at 1) must be expressions that evaluate to a `boolean` value. All even arguments (the `thens`) and the last argument (the `else`) must be of the same type.

Syntax

```
case( Predicate_1, Then_1, Predicate_2, Then_2, Predicate_3, Then_3, Else )
```

Arguments

- **Predicate_i**: An expression that evaluates to a `boolean` value.
- **Then_i**: An expression that gets evaluated and its value is returned from the function if **Predicate_i** is the first predicate that evaluates to `true`.
- **Else**: An expression that gets evaluated and its value is returned from the function if neither of the **Predicate_i** evaluate to `true`.

Results

The value of the first **Then_i** whose **Predicate_i** evaluates to `true`, or the value of **Else** if neither of the predicates is satisfied.

Example

```
dataset=myDataset
| extend bucket = case(Size <= 3, "Small",
                       Size <= 10, "Medium",
                       "Large")
```

Copy

10.5.3.2.2. Coalesce

The `coalesce` function evaluates a list of expressions and returns the first non-null expression.

Syntax

```
coalesce( Expression_1, Expression_2, ... )
```

Arguments

Expression_i: A scalar expression, to be evaluated.

- All arguments must be of the same type.
- Maximum of 64 arguments is supported.

Results

The value of the first **Expression_1** whose value isn't null (or not-empty for string expressions).

Example

- `coalesce(tolong("not a number"), tolong("42"), 33)` returns as `42`

10.5.3.2.3. Iff

The `iff` function evaluates the first argument (the predicate), and returns the value of either the second or third arguments, depending on whether the predicate evaluated to `true` (second) or `false` (third).

The second and third arguments must be of the same type.

Alias: iif (`iff` and `iif` are synonyms.)

Syntax

```
iff( Predicate, IfTrue, IfFalse )
```

Arguments

- **Predicate**: An expression that evaluates to a `boolean` value.
- **IfTrue**: An expression that gets evaluated and its value returned from the function if **Predicate** evaluates to `true`.
- **IfFalse**: An expression that gets evaluated and its value returned from the function if **Predicate** evaluates to `false`.

Results

This function returns the value of **IfTrue** if predicate evaluates to `true`, or the value of **IfFalse** otherwise.

Example

- `| extend day = iff(floor(Timestamp, 1d)==floor(now(), 1d), "today", "anotherday")`

10.5.3.2.4. IIF

The `iif` function evaluates the first argument (the predicate), and returns the value of either the second or third arguments, depending on whether the predicate evaluated to `true` (second) or `false` (third).

The second and third arguments must be of the same type.

Alias: iff (`iif` and `iff` are synonyms.)

Syntax

```
iif( Predicate, IfTrue, IfFalse )
```

Arguments

- **Predicate**: An expression that evaluates to a `boolean` value.
- **IfTrue**: An expression that gets evaluated and its value returned from the function if **Predicate** evaluates to `true`.
- **IfFalse**: An expression that gets evaluated and its value returned from the function if **Predicate** evaluates to `false`.

Results

This function returns the value of **IfTrue** if predicate evaluates to `true`, or the value of **IfFalse** otherwise.

Example

- `| extend day = iif(floor(Timestamp, 1d)==floor(now(), 1d), "today", "anotherday")`

10.5.3.2.5. MAX_OF

The `max_of` function returns the maximum value of several evaluated numeric expressions.

Syntax

```
max_of( Expression_1, Expression_2, ... )
```

Arguments

Expression_i: A scalar expression, to be evaluated.

- All arguments must be of the same type.
- Maximum of 64 arguments is supported.
- Non-null values take precedence to null values.

Results

The maximum value of all argument expressions.

Example

- `max_of(10, 1, -3, 17)` returns as `17`

10.5.3.2.6. MIN_OF

The `min_of` function returns the minimum value of several evaluated numeric expressions.

Syntax

```
min_of( Expression_1, Expression_2, ... )
```

Arguments

Expression_i: A scalar expression, to be evaluated.

- All arguments must be of the same type.
- Maximum of 64 arguments is supported.
- Non-null values take precedence to null values.

Results

The minimum value of all argument expressions.

Example

- `min_of(10, 1, -3, 17)` returns as `-3`

10.5.3.3. Conversion Functions

A list of all conversion functions supported by Cribl Search.

Name	Description
`bin`	Rounds values down to an integer multiple of a given bin size.
`bin_auto`	Rounds values down to a fixed-size bin.
`floor`	Rounds values down to an integer multiple of a given floor size.
`gettype`	Returns the type of the input value.
`tobool`	Converts the input to a `bool` value.
`todouble` (`toreal`, `todecimal`)	Converts the input to a `double` (`real`, `decimal`) value.
`toint` (`tolong`)	Converts the input to an `int` (`long`) value.
`tostring`	Converts the input to a `string` value.

10.5.3.3.1. Bin

The `bin` function rounds values down to an integer multiple of a given bin size. If you have a scattered set of values, they'll be grouped into a smaller set of specific values.

Null values, a null bin size, or a negative bin size results in `null`.

You can use this function with the `summarize` and `eventstats` operators.

Syntax

```
bin( Value, RoundTo )
```

Arguments

- **Value**: A number, date, or timespan.
- **RoundTo**: The "bin size". A number or timespan that divides value.

Returns

The nearest multiple of **RoundTo** below value.

Examples

- `bin(4.5, 1)` results in `4.0`
- `bin(time(16d), 7d)` results in `14d`
- `bin(datetime(1970-05-11 13:45:07), 1d)` results in `datetime(1970-05-11)`

The following expression calculates a histogram of durations, with a bucket size of 1 second:

```
dataset=myDataset
| summarize Hits=count() by bin(Duration, 1s)
```

Copy

10.5.3.3.2. BIN_AUTO

The `bin_auto` function rounds values down to a fixed-size "bin", with control over the bin size and starting point provided by an option property.

You can use this function with the `summarize` and `eventstats` operators.

Syntax

```
bin_auto( Expression )
```

with option:

```
set OptionName [ = OptionValue ] ; ... | ... bin_auto( Expression )
```

Arguments

- **Expression**: A scalar expression of a numeric type indicating the value to round.
- **OptionName**:
  - `query_bin_auto_size`: Time period. Supports these relative times – `s[econds]`, `m[inutes]`, `h[ours]`, and `d[ays]`.
  - `query_bin_auto_buckets`: Numeric literal, desired number of buckets to be created by `bin_auto`. Buckets are split as close as possible based on the search's time range.

Rules

- The total number of buckets will vary based on the time range.
- Values without units get interpreted as seconds. (For example, `-1 = -1s`.)

Example

```
set query_bin_auto_size=1h;
dataset=myDataset
| summarize count() by bin_auto(timestamp)
```

Copy

10.5.3.3.3. Floor

The `floor` function rounds values down to an integer multiple of a given floor size. If you have a scattered set of values, they'll be grouped into a smaller set of specific values.

Null values, a null floor size, or a negative floor size results in `null`.

You can use this function with the `summarize` and `eventstats` operators.

Syntax

```
floor( Value [, RoundTo ] )
```

Arguments

- **Value**: A number, date, or timespan.
- **RoundTo**: The "floor size". A number or timespan that divides value. Defaults to `1`.

Returns

The nearest multiple of **RoundTo** below value.

Examples

- `floor(4.5, 1)` results in `4`
- `floor(3.14)` results in `3`.
- `floor(time(16d), 7d)` results in `14d`
- `floor(datetime(1970-05-11 13:45:07), 1d)` results in `datetime(1970-05-11)`

The following expression calculates a histogram of durations, with a bucket size of 1 second:

```
dataset=myDataset
| summarize Hits=count() by floor(Duration, 1s)
```

Copy

10.5.3.3.4. GETTYPE

The `gettype` function returns the type of the input value.

Syntax

```
gettype( Expression )
```

Arguments

- **Expression**: The value whose type you want to check.

Returns

Depending on the input, returns one of the following strings:

Input	Returned string	Input examples
A boolean	`bool`	`true`, `false`, `3 < 5`
A number	`int` for values in range `-2^31…2^31-1`.	`3`, `1000`
	`long` for values in range `-2^63…2^63-1` but greater than `int`.	`2 ** 40`
	`decimal` for values greater than `long`.	`2 ** 65`
	`real`	`3.15`, `3 / 5`
	`NaN`	`NaN`
A `bigint`	Same as a number: `int`, `long`, or `decimal`, based on the value range.	`BigInt(1234567890123456789)`
A string	`string`	`'Aloa'`
`[]`	`array`	`[17, 3]`
`{}`	`dictionary`	`{ foo: 'bar' }`
`null`	`null`	`null`
Undefined	`null`	Undefined field reference

Input	Returned string	Input examples
Anything else, like functions	unknown	`coalesce(33)`

Examples

Input	Output
`true`	`bool`
`false`	`bool`
`0`	`int`
`42`	`int`
`2 ** 40`	`long`
`BigInt(2 ** 60)`	`long`
`3.1415`	`real`
`2 ** 65`	`decimal`
`BigInt(2 ** 65)`	`decimal`
`[17, 3]`	`array`
`{ foo: "bar" }`	`dictionary`
`Number.NaN`	`NaN`
`null`	`null`
Undefined	`null`
`() => 42`	`unknown`

10.5.3.3.5. TOBOOL

The `tobool` function converts the input to a value of type `bool`.

Aliases: `bool`.

Syntax

```
tobool( Expression )
```

Arguments

- **Expression**: Expression to convert to a `bool` value.

Returns

If the type conversion is successful, returns a `bool` value.

If the type conversion fails, returns `null`.

Examples

Query	Result
`tobool("true")`	`true`
`tobool("TRUE")`	`true`
`tobool("yes")`	`true`
`tobool("false")`	`false`
`tobool("f")`	`false`
`tobool(1)`	`true`
`tobool("1")`	`true`
`tobool(123)`	`true`
`tobool(0)`	`false`

Query	Result
`tobool(0.0)`	`false`
`tobool(“abc”)`	`null`
`tobool(null)`	`null`

10.5.3.3.6. TODECIMAL

The `todecimal` function converts the input to a value of type `decimal` (`double`, `real`).

Aliases: `decimal`, `double`, `todouble`, `real`, `toreal`.

Syntax

```
todecimal( Expression )
```

Arguments

- **Expression**: Expression whose value will be converted to a value of type `decimal`.

Returns

If the type conversion is successful, the result is a value of type `decimal` (`double`, `real`).

If the type conversion fails, returns `null`.

Examples

- `todecimal("123.4")` returns `123.4`
- `todecimal("1e3")` returns `1000`
- `todecimal("hello")` returns `null`

10.5.3.3.7. TODOUBLE

The `todouble` function converts the input to a value of type `double` (`real`, `decimal`).

Aliases: `double`, `decimal`, `todecimal`, `real`, `toreal`.

Syntax

```
todouble( Expression )
```

Arguments

- **Expression**: Expression whose value will be converted to a value of type `double`.

Returns

If the type conversion is successful, the result is a value of type `double` (`real`, `decimal`).

If the type conversion fails, returns `null`.

Example

- `todouble("123.4")` returns `123.4`
- `todouble("1e3")` returns `1000`
- `todouble("hello")` returns `null`

10.5.3.3.8. TOINT

The `toint` function converts the input to a value of type `int` (`long`).

Aliases: `int`, `long`, `tolong`.

Syntax

```
toint( Expression )
```

Arguments

- **Expression**: Expression to convert to a value of type `int`.

Returns

If the type conversion is successful, returns an `int` value. Decimal places are truncated.

If the type conversion fails, returns `null`.

Examples

- `toint(123)` returns `123`
- `toint("123.456")` returns `123`
- `toint("abc")` returns `null`

10.5.3.3.9. Tolong

The `tolong` function converts the input to a value of type `long` (`int`).

Aliases: `long`, `int`, `toint`.

Syntax

```
tolong( Expression )
```

Arguments

- **Expression**: Expression to convert to a value of type `long`.

Returns

If the type conversion is successful, returns a `long` number. Decimal places are truncated.

If the type conversion fails, returns `null`.

Example

- `tolong(123)` returns `123`
- `tolong("123.456")` returns `123`
- `tolong("abc")` returns `null`

10.5.3.3.10. TOREAL

The `toreal` function converts the input to a value of type `real` (`double`, `decimal`).

Aliases: `real`, `decimal`, `todecimal`, `double`, `todouble`.

Syntax

```
toreal( Expression )
```

Arguments

- **Expression**: Expression whose value will be converted to a value of type `real`.

Returns

If the type conversion is successful, returns a value of type `real` (`double`, `decimal`).

If the type conversion fails, returns `null`.

Example

- `toreal("123.4")` returns `123.4`
- `toreal("1e3")` returns `1000`
- `toreal("hello")` returns `null`

10.5.3.3.11. TOSTRING

The `tostring` function converts the input to a value of type `string`.

Syntax

```
tostring( Expression )
```

Arguments

- **Expression**: Expression to convert to a string.

Returns

If the **Expression** value is non-null, returns a string representation of **Expression**.

If the **Expression** value is `null`, returns `null` as well.



This is different from standard Kusto Query Language (KQL), in which `tostring` returns an empty string for `null` values.

Example

- `tostring(123)` returns `"123"`
- `tostring("hello")` returns `"hello"`
- `tostring(null)` returns `null`

10.5.3.4. DateTime Functions

A list of all DateTime functions supported by Cribl Search.

Name	Description
`ago`	Subtracts the given timespan from the current UTC clock time.
`datetime_add`	Calculates a new datetime from a specified datepart multiplied by a specified amount, added to a specified datetime.
`datetime_diff`	Calculates the number of the specified periods between two datetime values.
`datetime_part`	Extracts the requested date part as an integer value.
`dayofmonth`	Returns the integer number representing the day number of the given month.
`dayofweek`	Returns an integer between `0` and `6` representing the day of the week, beginning on Sunday.
`dayofyear`	Returns the integer number representing the day number of the given year.
`endofday`	Returns the end of the day containing the date, shifted by an offset, if provided.
`endofmonth`	Returns the end of the month containing the date, shifted by an offset, if provided.
`endofweek`	Returns the end of the week containing the date, shifted by an offset, if provided.
`endofyear`	Returns the end of the year containing the date, shifted by an offset, if provided.
`format_datetime`	Formats a datetime according to the provided format.
`format_timespan`	Formats a timespan according to the provided format.
`getmonth` (`monthofyear`)	Gets the month number (1–12) from a datetime.
`getyear` (`yearofyear`)	Returns the year part of a datetime.

Name	Description
hourofday	Returns the integer number representing the hour number of the given date.
make_datetime	Converts the specified date and time into a datetime value (Unix time in seconds).
make_timespan	Converts the specified time period into a datetime value (Unix time in seconds).
now	Returns the current UTC clock time as a datetime value (Unix time in seconds).
startofday	Returns the start of the day containing the date, shifted by an offset, if provided.
startofmonth	Returns the start of the month containing the date, shifted by an offset, if provided.
startofweek	Returns the start of the week containing the date, shifted by an offset, if provided.
startofyear	Returns the start of the year containing the date, shifted by an offset, if provided.
strftime (format_time)	Converts a datetime (date) object to a human-readable string.
strptime (parse_time)	Converts a string to a datetime.
todatetime	Converts the input into a datetime value (Unix time in seconds).
totimespan	Converts the input into a number of seconds.
unixtime_microseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime_milliseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime-nanoseconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
unixtime_seconds_todatetime	Converts the input into a datetime value (Unix time in seconds).
week_of_year	Returns an integer representing the week number.

10.5.3.4.1. Ago

The `ago` function subtracts the given timespan from the current UTC clock time.

Like now, this function can be used multiple times in a statement and the UTC clock time being referenced will be the same for all instantiations.

Syntax

```
ago( Timespan )
```

Arguments

- **Timespan**: Interval to subtract from the current UTC clock time.

Returns

```
now() - Timespan
```

Examples

This example subtracts two days from the current time, using, strftime to format the result as a human-readable date:

```
print newDate = ago(2d)
| extend prettyDate = strftime( newDate, '%x' )
```

Copy Run in Cribl Search

This example returns all rows with a timestamp in the past hour:

```
dataset=myDataset
| where Timestamp > ago(1h)
```

Copy

10.5.3.4.2. DATETIME_ADD

The `datetime_add` function calculates a new datetime from a specified datepart multiplied by a specified amount, added to a specified datetime.

Syntax

```
datetime_add( Period, Amount, Datetime )
```

Arguments

- **Period**: Possible values are: `Year`, `Quarter`, `Month`, `Week`, `Day`, `Hour`, `Minute`, `Second`, `Millisecond`, `Microsecond`, `Nanosecond`.
- **Amount**: An integer.
- **Datetime**: A datetime.

Returns

A date after a certain time/date interval has been added, represented as `datetime` (Unix time).

Examples

This example adds two days to the specified date:

```
print newDate=datetime_add( 'day', 2, make_datetime( 2025,12,31 ) )
| extend prettyDate = strftime( newDate, '%x'  )
```

Copy Run in Cribl Search

The following examples illustrate calling this function with different values for the `Period` argument:

```
print year = datetime_add('year',1,make_datetime(2025,1,1))
```

Copy Run in Cribl Search

```
print month = datetime_add('month',1,make_datetime(2025,1,1))
```

Copy Run in Cribl Search

```
print quarter = datetime_add('quarter',1,make_datetime(2025,1,1))
```

Copy Run in Cribl Search

```
print week = datetime_add('week',1,make_datetime(2025,1,1))
```

Copy Run in Cribl Search

```
print day = datetime_add('day',1,make_datetime(2025,1,1))
```

Copy Run in Cribl Search

```
print hour = datetime_add('hour',1,make_datetime(2025,1,1))
```

Copy Run in Cribl Search

```
print minute = datetime_add('minute',1,make_datetime(2025,1,1))
```

Copy Run in Cribl Search

```
print second = datetime_add('second',1,make_datetime(2025,1,1))
```

Copy Run in Cribl Search

10.5.3.4.3. DATETIME_DIFF

The `datetime_diff` function calculates the number of the specified periods between two datetime values.

Syntax

```
datetime_diff( Period, Datetime1, Datetime2 )
```

Arguments

- **Period**: Possible values are: `Year`, `Quarter`, `Month`, `Week`, `Day`, `Hour`, `Minute`, `Second`, `Millisecond`, `Microsecond`, `Nanosecond`.
- **DatetimeX**: A datetime.

Returns

An integer, which represents amount of **Period**s in the result of subtraction `(Datetime1 - Datetime2)`.

Examples

The following examples illustrate calling this function with different values for the `Period` argument:

```
printyear = datetime_diff('year',datetime(2017-01-01),datetime(2000-12-31))
```

Copy Run in Cribl Search

```
printquarter = datetime_diff('quarter',datetime(2017-07-01),datetime(2017-03-30))
```

Copy Run in Cribl Search

```
printmonth = datetime_diff('month',datetime(2017-01-01),datetime(2015-12-30))
```

Copy Run in Cribl Search

```
printweek = datetime_diff('week',datetime(2017-10-29 00:00),datetime(2017-09-30 23:59
```

Copy Run in Cribl Search

```
printday = datetime_diff('day',datetime(2017-10-29 00:00),datetime(2017-09-30 23:59)
```

Copy Run in Cribl Search

```
printhour = datetime_diff('hour',datetime(2017-10-31 01:00),datetime(2017-10-30 23:59
```

Copy Run in Cribl Search

```
printminute = datetime_diff('minute',datetime(2017-10-30 23:05:01),datetime(2017-10-3
```

Copy Run in Cribl Search

```
printsecond = datetime_diff('second',datetime(2017-10-30 23:00:10.100),datetime(2017-
```

Copy Run in Cribl Search

```
printmillisecond = datetime_diff('millisecond',datetime(2017-10-30 23:00:00.200100),d
```

Copy Run in Cribl Search

```
printmicrosecond = datetime_diff('microsecond',datetime(2017-10-30 23:00:00.1009001),
```

Copy Run in Cribl Search

```
printnanosecond = datetime_diff('nanosecond',datetime(2017-10-30 23:00:00.0000000),da
```

Copy Run in Cribl Search

10.5.3.4.4. DATETIME_PART

The `datetime_part` function extracts the requested date part as an integer value.

Syntax

```
datetime_part( Part, Datetime )
```

Arguments

- **Part**: Possible values are: `Year`, `Quarter`, `Month`, `week_of_year`, `Day`, `DayOfYear`, `Hour`, `Minute`, `Second`, `Millisecond`, `Microsecond`, `Nanosecond`.
- **Datetime**: A datetime.

Returns

An integer representing the extracted part.



`week_of_year` returns an integer that represents the week number. The week number is calculated from the first week of a year, which is the one that includes the first Thursday.

Examples

The following examples illustrate calling this function with different values for the `Part` argument:

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print year = datetime_part("year", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print quarter = datetime_part("quarter", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print month = datetime_part("month", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print weekOfYear = datetime_part("week_of_year", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print day = datetime_part("day", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print dayOfYear = datetime_part("dayOfYear", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print hour = datetime_part("hour", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print minute = datetime_part("minute", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print second = datetime_part("second", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print millisecond = datetime_part("millisecond", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print microsecond = datetime_part("microsecond", dt)
```

Copy Run in Cribl Search

```
let dt = datetime(2017-10-30 01:02:03.7654321);
print nanosecond = datetime_part("nanosecond", dt)
```

Copy Run in Cribl Search

10.5.3.4.5. DAYOFMONTH

The `dayofmonth` function returns the integer number representing the day number of the given month.

Syntax

```
dayofmonth( Date )
```

Arguments

- **Date**: A `datetime`.

Returns

The day number of the given month.

Example

This example returns `14`:

```
print dayofmonth(datetime(2015-12-14))
```

Copy | Run in Cribl Search

10.5.3.4.6. DAYOFWEEK

The `dayofweek` function returns an integer between `0` and `6` representing the day of the week, beginning on Sunday.



This differs from the standard Kusto `dayofweek` function, which returns a timespan.

Syntax

```
dayofweek( Date )
```

Arguments

- **Date**: A `datetime`.

Returns

The `timespan` since midnight at the beginning of the preceding Sunday, rounded down to an integer number of days.

Example

This example returns `2 for Tuesday:

```
print dayofweek( datetime(2025-12-16) )
```

Copy Run in Cribl Search

10.5.3.4.7. DAYOFYEAR

The `dayofyear` function returns the integer number representing the day number of the given year.

Syntax

```
dayofyear( Date )
```

Arguments

- **Date**: A `datetime`.

Returns

The day number of the given year.

Example

This example returns `348`:

```
print dayofyear( datetime(2025-12-14) )
```

Copy | Run in Cribl Search

10.5.3.4.8. ENDOFDAY

The `endofday` function returns the end of the day containing the date, shifted by an offset, if provided.

Syntax

```
endofday( Date [, Offset] )
```

Arguments

- **Date**: The input date.
- **Offset**: An optional number of offset days from the input date (integer, default - 0).

Returns

A `datetime` representing the end of the day for the given **Date** value, with the offset, if specified.

Example

This example returns `1,766,707,199.999`, representing `Thu Dec 25 2025 23:59:59 GMT+0000`:

```
project dayEnd = endofday( datetime(2025-12-24), 1 )
```

Copy | Run in Cribl Search

10.5.3.4.9. ENDOFMONTH

The `endofmonth` function returns the end of the month containing the date, shifted by an offset, if provided.

Syntax

```
endofmonth( Date [, Offset] )
```

Arguments

- **Date**: The input date.
- **Offset**: An optional number of offset months from the input date (integer, default - 0).

Returns

A `datetime` representing the end of the month for the given **Date** value, with the offset, if specified.

Example

This example returns `1,769,903,999.999`, representing `Sat Jan 31 2026 23:59:59 GMT+0000`:

```
print monthEnd = endofmonth( datetime(2025-12-21 10:10:17), 1 )
```

Copy | Run in Cribl Search

10.5.3.4.10. ENDOFWEEK

The `endofweek` function returns the end of the week containing the date, shifted by an offset, if provided.

Last day of the week is considered to be a Saturday.

Syntax

```
endofweek( Date [, Offset] )
```

Arguments

- **Date**: The input date.
- **Offset**: An optional number of offset weeks from the input date (integer, default - 0).

Returns

A `datetime` representing the end of the week for the given **Date** value, with the offset, if specified.

Example

This example returns `1,767,484,799.999`, representing `Sat Jan 31 2026 23:59:59 GMT+0000`:

```
print weekEnd = endofweek( datetime(2025-12-21 10:10:17), 1)
```

Copy | Run in Cribl Search

10.5.3.4.11. ENDOFYEAR

The `endofyear` function returns the end of the year containing the date, shifted by an offset, if provided.

Syntax

```
endofyear( Date [, Offset] )
```

Arguments

- **Date**: The input date.
- **Offset**: An optional number of offset years from the input date (integer, default - 0).

Returns

A `datetime` representing the end of the year for the given **Date** value, with the offset, if specified.

Example

This example returns `1,798,761,599.999`, representing `Wed Dec 31 2026 23:59:59 GMT+0000`:

```
print yearEnd = endofyear( datetime(2025-10-31 10:10:17), 1)
```

Copy | Run in Cribl Search

10.5.3.4.12. FORMAT_DATETIME

The `format_datetime` function formats a `datetime` according to the provided format.

Syntax

```
format_datetime( Datetime, Format)
```

Arguments

- **Datetime**: A `datetime`.
- **Format**: Format specifier string, consisting of one or more format elements.

Returns

The string with the format result.

Supported Formats

Format specifier	Description	Examples
`d`	The day of the month, from 1 through 31.	2009-06-01T13:45:30 -> 1, 2009-06-15T13:45:30 -> 15
`dd`	The day of the month, from 01 through 31.	2009-06-01T13:45:30 -> 01, 2009-06-15T13:45:30 -> 15
`f`	The tenths of a second in a date and time value.	2009-06-15T13:45:30.6170000 -> 6, 2009-06-15T13:45:30.05 -> 0
`ff`	The hundredths of a second in a date and time value.	2009-06-15T13:45:30.6170000 -> 61, 2009-06-15T13:45:30.0050000 -> 00
`fff`	The milliseconds in a date and time value.	6/15/2009 13:45:30.617 -> 617, 6/15/2009 13:45:30.0005 -> 000
`ffff`	The ten thousandths of a second in a date and time value.	2009-06-15T13:45:30.6175000 -> 6175, 2009-06-15T13:45:30.0000500 -> 0000
`fffff`	The hundred thousandths of a second in a date and time value.	2009-06-15T13:45:30.6175400 -> 61754, 2009-06-15T13:45:30.000005 -> 00000

Format specifier	Description	Examples
`ffffff`	The millionths of a second in a date and time value.	2009-06-15T13:45:30.6175420 -> 617542, 2009-06-15T13:45:30.0000005 -> 000000
`fffffff`	The ten millionths of a second in a date and time value.	2009-06-15T13:45:30.6175425 -> 6175425, 2009-06-15T13:45:30.0001150 -> 0001150
`F`	If non-zero, the tenths of a second in a date and time value.	2009-06-15T13:45:30.6170000 -> 6, 2009-06-15T13:45:30.0500000 -> (no output)
`FF`	If non-zero, the hundredths of a second in a date and time value.	2009-06-15T13:45:30.6170000 -> 61, 2009-06-15T13:45:30.0050000 -> (no output)
`FFF`	If non-zero, the milliseconds in a date and time value.	2009-06-15T13:45:30.6170000 -> 617, 2009-06-15T13:45:30.0005000 -> (no output)
`FFFF`	If non-zero, the ten thousandths of a second in a date and time value.	2009-06-15T13:45:30.5275000 -> 5275, 2009-06-15T13:45:30.0000500 -> (no output)
`FFFFF`	If non-zero, the hundred thousandths of a second in a date and time value.	2009-06-15T13:45:30.6175400 -> 61754, 2009-06-15T13:45:30.0000050 -> (no output)
`FFFFFF`	If non-zero, the millionths of a second in a date and time value.	2009-06-15T13:45:30.6175420 -> 617542, 2009-06-15T13:45:30.0000005 -> (no output)
`FFFFFFF`	If non-zero, the ten millionths of a second in a date and time value.	2009-06-15T13:45:30.6175425 -> 6175425, 2009-06-15T13:45:30.0001150 -> 0001150
`h`	The hour, using a 12-hour clock from 1 to 12.	2009-06-15T01:45:30 -> 1, 2009-06-15T13:45:30 -> 1
`hh`	The hour, using a 12-hour clock from 01 to 12.	2009-06-15T01:45:30 -> 01, 2009-06-15T13:45:30 -> 01
`H`	The hour, using a 24-hour clock from 0 to 23.	2009-06-15T01:45:30 -> 1, 2009-06-15T13:45:30 -> 13
`HH`	The hour, using a 24-hour clock from 00 to 23.	2009-06-15T01:45:30 -> 01, 2009-06-15T13:45:30 -> 13
`m`	The minute, from 0 through 59.	2009-06-15T01:09:30 -> 9, 2009-06-15T13:29:30 -> 29
`mm`	The minute, from 00 through 59.	2009-06-15T01:09:30 -> 09, 2009-06-15T01:45:30 -> 45
`M`	The month, from 1 through 12.	2009-06-15T13:45:30 -> 6

Format specifier	Description	Examples
MM	The month, from 01 through 12.	2009-06-15T13:45:30 -> 06
s	The second, from 0 through 59.	2009-06-15T13:45:09 -> 9
ss	The second, from 00 through 59.	2009-06-15T13:45:09 -> 09
y	The year, from 0 to 99.	0001-01-01T00:00:00 -> 1, 0900-01-01T00:00:00 -> 0, 1900-01-01T00:00:00 -> 0, 2009-06-15T13:45:30 -> 9, 2019-06-15T13:45:30 -> 19
yy	The year, from 00 to 99.	0001-01-01T00:00:00 -> 01, 0900-01-01T00:00:00 -> 00, 1900-01-01T00:00:00 -> 00, 2019-06-15T13:45:30 -> 19
yyyy	The year as a four-digit number.	0001-01-01T00:00:00 -> 0001, 0900-01-01T00:00:00 -> 0900, 1900-01-01T00:00:00 -> 1900, 2009-06-15T13:45:30 -> 2009
tt	AM / PM hours	2009-06-15T13:45:09 -> PM

Supported Delimiters

Format specifier can include following delimiter characters:

Delimiter	Comment
	Space
/	
-	Dash
:	
,	
.	
_	
[	
]	

Examples

This example returns `17-01-29 [09:00:05]`:

```
let dt = datetime(2017-01-29 09:00:05);
print v1=format_datetime(dt,'yy-MM-dd [HH:mm:ss]')
```

Copy Run in Cribl Search

This example returns `2017-1-29 [9:00:05]`:

```
let dt = datetime(2017-01-29 09:00:05);
print v2=format_datetime(dt, 'yyyy-M-dd [H:mm:ss]')
```

Copy Run in Cribl Search

This example returns `17-01-29 [09:00:05 AM]`:

```
let dt = datetime(2017-01-29 09:00:05);
print v3=format_datetime(dt, 'yy-MM-dd [hh:mm:ss tt]')
```

Copy Run in Cribl Search

10.5.3.4.13. FORMAT_TIMESPAN

The `format_timespan` function formats a timespan according to the provided format.

Syntax

```
format_timespan( Timespan, Format)
```

Arguments

- **Timespan**: A timespan.
- **Format**: Format specifier string, consisting of one or more format elements.

Returns

The string with the format result.

Supported Formats

Format specifier	Description	Examples
`d-ddddddd`	The number of whole days in the time interval. Padded with zeros if needed.	15.13:45:30: d -> 15, dd -> 15, ddd -> 015
`f`	The tenths of a second in the time interval.	15.13:45:30.6170000 -> 6, 15.13:45:30.05 -> 0
`ff`	The hundredths of a second in the time interval.	15.13:45:30.6170000 -> 61, 15.13:45:30.0050000 -> 00
`fff`	The milliseconds in the time interval.	6/15/2009 13:45:30.617 -> 617, 6/15/2009 13:45:30.0005 -> 000
`ffff`	The ten thousandths of a second in the time interval.	15.13:45:30.6175000 -> 6175, 15.13:45:30.0000500 -> 0000
`fffff`	The hundred thousandths of a second in the time interval.	15.13:45:30.6175400 -> 61754, 15.13:45:30.000005 -> 00000
`ffffff`	The millionths of a second in the time interval.	15.13:45:30.6175420 -> 617542, 15.13:45:30.0000005 -> 000000

Format specifier	Description	Examples
`fffffff`	The ten millionths of a second in the time interval.	15.13:45:30.6175425 -> 6175425, 15.13:45:30.0001150 -> 0001150
`F`	If non-zero, the tenths of a second in the time interval.	15.13:45:30.6170000 -> 6, 15.13:45:30.0500000 -> (no output)
`FF`	If non-zero, the hundredths of a second in the time interval.	15.13:45:30.6170000 -> 61, 15.13:45:30.0050000 -> (no output)
`FFF`	If non-zero, the milliseconds in the time interval.	15.13:45:30.6170000 -> 617, 15.13:45:30.0005000 -> (no output)
`FFFF`	If non-zero, the ten thousandths of a second in the time interval.	15.13:45:30.5275000 -> 5275, 15.13:45:30.0000500 -> (no output)
`FFFFF`	If non-zero, the hundred thousandths of a second in the time interval.	15.13:45:30.6175400 -> 61754, 15.13:45:30.0000050 -> (no output)
`FFFFFF`	If non-zero, the millionths of a second in the time interval.	15.13:45:30.6175420 -> 617542, 15.13:45:30.0000005 -> (no output)
`FFFFFFF`	If non-zero, the ten millionths of a second in the time interval.	15.13:45:30.6175425 -> 6175425, 15.13:45:30.0001150 -> 000115
`H`	The hour, using a 24-hour clock from 0 to 23.	15.01:45:30 -> 1, 15.13:45:30 -> 13
`HH`	The hour, using a 24-hour clock from 00 to 23.	15.01:45:30 -> 01, 15.13:45:30 -> 13
`m`	The number of whole minutes in the time interval that are not included as part of hours or days. Single-digit minutes do not have a leading zero.	15.01:09:30 -> 9, 15.13:29:30 -> 29
`mm`	The number of whole minutes in the time interval that are not included as part of hours or days. Single-digit minutes have a leading zero.	15.01:09:30 -> 09, 15.01:45:30 -> 45
`s`	The number of whole seconds in the time interval that are not included as part of hours, days, or minutes. Single-digit seconds do not have a leading zero.	15.13:45:09 -> 9

Format specifier	Description	Examples
ss	The number of whole seconds in the time interval that are not included as part of hours, days, or minutes. Single-digit seconds have a leading zero.	15.13:45:09 -> 09

Supported Delimiters

Format specifier can include following delimiter characters:

Delimiter	Comment
	Space
/	
-	Dash
:	
,	
.	
_	
[	
]	

Examples

This example returns `29.09:00:05:12`:

```
let t = make_timespan(1,12,30,55.123);
print v1=format_timespan(t, 'dd.hh:mm:ss:FF')
```

Copy Run in Cribl Search

This example returns `001.12:30:55 [1230000]`:

```
let t = make_timespan(1,12,30,55.123);
print v2=format_timespan(t, 'ddd.h:mm:ss [fffffff]')
```

Copy Run in Cribl Search

10.5.3.4.14. GETMONTH

The `getmonth` function gets the month number (1–12) from a `datetime`.

Alias: `monthofyear`

Syntax

```
getmonth( Datetime )
```

Arguments

- **Datetime**: A `datetime`.

Example

This example returns `10`:

```
print month = getmonth( datetime(2025-10-14) )
```

Copy | Run in Cribl Search

10.5.3.4.15. GETYEAR

The `getyear` function returns the year part of a `datetime`.

Alias: `yearofyear`

Syntax

```
getyear( Datetime )
```

Arguments

- **Datetime**: A `datetime`.

Example

This example returns `2025`:

```
print year = getyear( datetime(2025-07-14) )
```

Copy | Run in Cribl Search

10.5.3.4.16. HOUROFDAY

The `hourofday` function returns the integer representing the hour number of the given date.

Syntax

```
hourofday( Datetime )
```

Arguments

- **Datetime**: A `datetime`.

Returns

The hour number of the day (0–23).

Example

This example returns `18`:

```
print hourofday( datetime(2025-11-30 18:54) )
```

Copy | Run in Cribl Search

10.5.3.4.17. MAKE_DATETIME

The `make_datetime` function converts the specified date and time into a `datetime` value (Unix time in seconds).



To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
make_datetime( Year, Month, Day [, Hour] [, Minute] [, Second])
```

Arguments

- **Year**: year (an integer value, from 0 to 9999)
- **Month**: month (an integer value, from 1 to 12)
- **Day**: day (an integer value, from 1 to 28–31)
- **Hour**: hour (an integer value, from 0 to 23)
- **Minute**: minute (an integer value, from 0 to 59)
- **Second**: second (a real value, from 0 to 59.9999999)

Returns

If creation is successful, result will be a `datetime` value; otherwise, the result will be `null`.

Example

This example returns `1506729600`:

```
print year_month_day = make_datetime(2017,10,0)
```

Copy Run in Cribl Search

10.5.3.4.18. MAKE_TIMESPAN

The `make_timespan` function converts the specified time period into a number of seconds.



This output format differs from the standard Kusto `make_timespan` function, which returns a timespan.

To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
make_timespan( [ Day ,] Hour, Minute [, Second])
```

Arguments

- **Day**: day (an integer value, from 1 to 28–31)
- **Hour**: hour (an integer value, from 0 to 23)
- **Minute**: minute (an integer value, from 0 to 59)
- **Second**: second (a real value, from 0 to 59.9999999)

Returns

If creation is successful, the result will be the number of seconds in the interval. Otherwise, the result will be `null`.

Examples

This example returns `131455.123`:

```
print make_timespan(1,12,30,55.123)
```

Copy Run in Cribl Search

This example returns `3600`:

```
print make_timespan(1, 0)
```

Copy Run in Cribl Search

10.5.3.4.19. MONTHOFYEAR

The `monthofyear` function returns the integer number representing the month number of the given year.

Syntax

```
monthofyear( Datetime )
```

Arguments

- **Datetime**: A `datetime`.

Returns

The month number of the given year.

Example

This example returns `11`:

```
print monthofyear( datetime("2025-11-30") )
```

Copy Run in Cribl Search

10.5.3.4.20. Now

The `now` function returns the current UTC clock time as a datetime value (Unix time in seconds), optionally offset by a given timespan.

You can use `now` multiple times in a statement and the clock time being referenced will be the same for all instances.



To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
now( [ Offset ] )
```

Arguments

- **Offset**: A timespan, added to the current UTC clock time. Default: 0.

Returns

The current UTC clock time as a `datetime`.

Examples

This returns the current time, in Unix epoch time:

```
print now()
```

Copy Run in Cribl Search

This returns the current time two days earlier, in Unix epoch time:

```
print now(-2d)
```

Copy Run in Cribl Search

10.5.3.4.21. STARTOFDAY

The `startofday` function returns the start of the day containing the date, shifted by an offset, if provided.

Syntax

```
startofday( Date, Offset )
```

Arguments

- **Date**: The input date.
- **Offset**: An optional number of offset days from the input date (integer, default - 0).

Returns

A `datetime` representing the start of the day for the given **Date** value, with the offset, if specified.

Example

This example returns `1641081600`:

```
print dayStart = startofday( datetime(2022-01-01 10:10:17), 1 )
```

Copy | Run in Cribl Search

10.5.3.4.22. STARTOFMONTH

The `startofmonth` function returns the start of the month containing the date, shifted by an offset, if provided.

Syntax

```
startofmonth( Date, Offset )
```

Arguments

- **Date**: The input date.
- **Offset**: An optional number of offset months from the input date (integer, default - 0).

Returns

A `datetime` representing the start of the month for the given **Date** value, with the offset, if specified.

Example

This example returns `1664582400`:

```
print monthStart = startofmonth( datetime(2022-09-01 10:10:17), 1 )
```

Copy | Run in Cribl Search

10.5.3.4.23. STARTOFWEEK

The `startofweek` function returns the start of the week containing the date, shifted by an offset, if provided.

Start of the week is considered to be a Sunday.

Syntax

```
startofweek( Date, Offset )
```

Arguments

- **Date**: The input date.
- **Offset**: An optional number of offset weeks from the input date (integer, default - 0).

Returns

A `datetime` representing the start of the week for the given **Date** value, with the offset, if specified.

Example

This example returns `1,662,249,600`:

```
print weekStart = startofweek( datetime(2022-09-01 10:10:17), 1 )
```

Copy | Run in Cribl Search

10.5.3.4.24. STARTOFYEAR

The `startofyear` function returns the start of the year containing the date, shifted by an offset, if provided.

Syntax

```
startofyear( Date, Offset )
```

Arguments

- **Date**: The input date.
- **Offset**: An optional number of offset years from the input date (integer, default - 0).

Returns

A `datetime` representing the start of the year for the given **Date** value, with the offset, if specified.

Example

This example returns `1,767,225,600`:

```
print yearStart = startofyear( datetime(2025-12-01 10:10:17), 1 )
```

Copy | Run in Cribl Search

10.5.3.4.25. STRFTIME

The `strftime` function converts a `datetime` (date) object to a human-readable string.

Alias: `format_time`

Syntax

```
strftime( TimeSeconds, FormatString )
```

Arguments

- **TimeSeconds**: The time value in seconds to convert to a string.
- **FormatString**: The time pattern to convert the date into.

Time Pattern

The time pattern string supports the following directives:

- `%a` - abbreviated weekday name.
- `%A` - full weekday name.
- `%b` - abbreviated month name.
- `%B` - full month name.
- `%c` - the locale's date and time, such as `%x, %X`.
- `%d` - zero-padded day of the month as a decimal number [01,31].
- `%e` - space-padded day of the month as a decimal number [1,31]; equivalent to `%_d`.
- `%f` - microseconds as a decimal number [000000, 999999].
- `%g` - ISO 8601 week-based year without century as a decimal number [00,99].
- `%G` - ISO 8601 week-based year with century as a decimal number.
- `%H` - hour (24-hour clock) as a decimal number [00,23].
- `%I` - hour (12-hour clock) as a decimal number [01,12].
- `%j` - day of the year as a decimal number [001,366].
- `%m` - month as a decimal number [01,12].
- `%M` - minute as a decimal number [00,59].
- `%L` - milliseconds as a decimal number [000, 999].

- `%p` - either AM or PM.
- `%q` - quarter of the year as a decimal number [1,4].
- `%Q` - milliseconds since UNIX epoch.
- `%s` - seconds since UNIX epoch.
- `%S` - second as a decimal number [00,61].
- `%u` - Monday-based (ISO 8601) weekday as a decimal number [1,7].
- `%U` - Sunday-based week of the year as a decimal number [00,53].
- `%V` - ISO 8601 week of the year as a decimal number [01, 53].
- `%w` - Sunday-based weekday as a decimal number [0,6].
- `%W` - Monday-based week of the year as a decimal number [00,53].
- `%x` - the locale's date, such as `%-m/%-d/%Y`.
- `%X` - the locale's time, such as `%-I:%M:%S %p`.
- `%y` - year without century as a decimal number [00,99].
- `%Y` - year with century as a decimal number, such as `1999`.
- `%Z` - time zone offset, such as `-0700`, `-07:00`, `-07`, or `Z`.
- `%%` - a literal percent sign (`%`).

For more details, see format specifiers.

Example

This example returns `"2022-06-27T17:00:26.499+0000"`:

```
print strftime( 1656349226.499, '%Y-%m-%dT%H:%M:%S.%L%Z' )
```

Copy | Run in Cribl Search

10.5.3.4.26. STRPTIME

The `strptime` function converts a string to a `datetime`.

Alias: `parse_time`

Syntax

```
strptime( TimeString, FormatString )
```

Arguments

- **TimeString**: The string to parse to a `datetime`.
- **FormatString**: The time pattern that matches the string to parse.

Time Pattern

The time pattern string supports the following directives:

- `%a` - abbreviated weekday name.
- `%A` - full weekday name.
- `%b` - abbreviated month name.
- `%B` - full month name.
- `%c` - the locale's date and time, such as `%x, %X`.
- `%d` - zero-padded day of the month as a decimal number [01,31].
- `%e` - space-padded day of the month as a decimal number [1,31]; equivalent to `%_d`.
- `%f` - microseconds as a decimal number [000000, 999999].
- `%g` - ISO 8601 week-based year without century as a decimal number [00,99].
- `%G` - ISO 8601 week-based year with century as a decimal number.
- `%H` - hour (24-hour clock) as a decimal number [00,23].
- `%I` - hour (12-hour clock) as a decimal number [01,12].
- `%j` - day of the year as a decimal number [001,366].
- `%m` - month as a decimal number [01,12].
- `%M` - minute as a decimal number [00,59].
- `%L` - milliseconds as a decimal number [000, 999].

- `%p` - either AM or PM.
- `%q` - quarter of the year as a decimal number [1,4].
- `%Q` - milliseconds since UNIX epoch.
- `%s` - seconds since UNIX epoch.
- `%S` - second as a decimal number [00,61].
- `%u` - Monday-based (ISO 8601) weekday as a decimal number [1,7].
- `%U` - Sunday-based week of the year as a decimal number [00,53].
- `%V` - ISO 8601 week of the year as a decimal number [01, 53].
- `%w` - Sunday-based weekday as a decimal number [0,6].
- `%W` - Monday-based week of the year as a decimal number [00,53].
- `%x` - the locale's date, such as `%-m/%-d/%Y`.
- `%X` - the locale's time, such as `%-I:%M:%S %p`.
- `%y` - year without century as a decimal number [00,99].
- `%Y` - year with century as a decimal number, such as `1999`.
- `%Z` - time zone offset, such as `-0700`, `-07:00`, `-07`, or `Z`.
- `%%` - a literal percent sign (`%`).

For more details, see format specifiers.

Example

This example returns `1,647,518,400`:

```
print strptime( '2022-03-17T12:00:00.000Z', '%Y-%m-%dT%H:%M:%S.%L%Z' )
```

Copy | Run in Cribl Search

10.5.3.4.27. TODATETIME

The `todatetime` function converts the input into a `datetime` value (Unix time in seconds).



To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
todatetime( Expression )
```

Arguments

- **Expression**: Expression that will be converted into `datetime`.

Returns

If the conversion is successful, the result will be a `datetime` value, otherwise, the result will be `null`.

Example

This example returns `1,485,216,000`:

```
print todatetime( "2015-25-24" )
```

Copy | Run in Cribl Search

10.5.3.4.28. TOTIMESPAN

The `totimespan` function converts the input expression into a time interval, in seconds.



This output format differs from the standard Kusto `totimespan` function, which returns a timespan.

To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
totimespan( Expression )
```

Arguments

- **Expression**: Expression that will be converted to timespan.

Returns

If the conversion is successful, the result will be the number of seconds in the interval. Otherwise, the result will be `null`.

Example

This example returns `3660`:

```
print totimespan( "0.01:01:00" )
```

Copy Run in Cribl Search

10.5.3.4.29. UNIXTIME_MICROSECONDS_TODATETIME

The `unixtime_microseconds_todatetime` function converts the input into a `datetime` value (Unix time in seconds).



To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
unixtime_microseconds_todatetime( Microseconds )
```

Arguments

- **Microseconds**: An epoch timestamp in microseconds. Time that occurs before the Unix epoch (1970-01-01 00:00:00) has a negative timestamp value.

Returns

If the conversion is successful, the result will be a `datetime` value (Unix time in seconds). Otherwise, the result will be `null`.

Example

This example returns `1,546,300,800`:

```
print unixtime_microseconds_todatetime( 1546300800000000 )
```

Copy | Run in Cribl Search

10.5.3.4.30. UNIXTIME_MILLISECONDS_TODATETIME

The `unixtime_milliseconds_todatetime` function converts the input into a `datetime` value (Unix time in seconds).

To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
unixtime_milliseconds_todatetime( Milliseconds )
```

Arguments

- **Milliseconds**: An epoch timestamp in milliseconds. Time that occurs before the Unix epoch (1970-01-01 00:00:00) has a negative timestamp value.

Returns

If the conversion is successful, the result will be a `datetime` value (Unix time in seconds). Otherwise, the result will be `null`.

Example

This example returns `1,546,300,800`:

```
print unixtime_milliseconds_todatetime( 1546300800000 )
```

Copy | Run in Cribl Search

10.5.3.4.31. UNIXTIME_NANOSECONDS_TODATETIME

The `unixtime_nanoseconds_todatetime` function converts the input into a `datetime` value (Unix time in seconds).



To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
unixtime_nanoseconds_todatetime( Nanoseconds )
```

Arguments

- **Nanoseconds**: An epoch timestamp in nanoseconds. Time that occurs before the Unix epoch (1970-01-01 00:00:00) has a negative timestamp value.

Returns

If the conversion is successful, the result will be a `datetime` value (Unix time in seconds). Otherwise, the result will be `null`.

Example

This example returns `1,546,300,800`:

```
print unixtime_nanoseconds_todatetime( 1546300800000000000 )
```

Copy | Run in Cribl Search

10.5.3.4.32. UNIXTIME_SECONDS_TODATETIME

The `unixtime_seconds_todatetime` function converts the input into a `datetime` value (Unix time in seconds).



To convert `datetime` (Unix time) into a `string`, or vice versa, use the `strftime` and `strptime` functions.

Syntax

```
unixtime_seconds_todatetime( Seconds )
```

Arguments

- **Seconds**: An epoch timestamp in seconds. Time that occurs before the Unix epoch (1970-01-01 00:00:00) has a negative timestamp value.

Returns

If the conversion is successful, the result will be a `datetime` value (Unix time in seconds). Otherwise, the result will be `null`.

Example

This example returns `1,546,300,800`:

```
print date_time = unixtime_seconds_todatetime( 1546300800 )
```

Copy | Run in Cribl Search

10.5.3.4.33. WEEK_OF_YEAR

The `week_of_year` function returns an integer representing the week number. The week number is calculated from the first week of a year, which is the one that includes the first Thursday, according to ISO 8601.

Syntax

```
week_of_year( Datetime )
```

Arguments

- **Datetime**: A `datetime`.

Returns

The week number of the given year.

Examples

This example returns `53`:

```
print week_of_year( datetime(2020-12-31) )
```

Copy Run in Cribl Search

This example returns `1`:

```
print week_of_year( datetime(1970-01-01) )
```

Copy Run in Cribl Search

10.5.3.5. Dynamic Functions

A list of all dynamic functions supported by Cribl Search.

Dynamic scalar functions allow you to manipulate objects by operating on `dynamic` values, including dynamic arrays and property bags.

Name	Description
`bag_has_key`	Checks whether a property bag contains a given key.
`bag_keys`	Lists all root keys of a property bag.
`bag_merge`	Merges multiple property bags, discarding duplicate keys.
`bag_pack`	Creates a property bag from an alternating list of keys and values.
`bag_pack_columns`	Creates a property bag from a list of columns.
`bag_remove_keys`	Removes key-value pairs from a property bag.
`bag_set_key`	Adds or overwrites a key-value pair in a property bag.
`bag_zip`	Creates a property bag from two dynamic arrays.
`make_bag`	Creates a property bag from multiple input bags.
`make_bag_if`	Creates a property bag from those input bags that meet the specified condition.
`zip`	Merges dynamic arrays, grouping elements by index.

The following string functions support `dynamic` types as well:

Function	Usage with dynamic data types
`extractjson(\`path,object\`)`	Uses path to navigate into object.
`parse_json(\`source\`)`	Turns a JSON string into a dynamic object.
`range(\`from,to,step\`)`	Generates an array of values.

10.5.3.5.1. BAG_HAS_KEY

The `bag_has_key` function checks whether a property bag contains a given key.

Syntax

```
bag_has_key( Bag, Key )
```

Parameters

Name	Type	Required	Description
Bag	dynamic	Yes	The property bag to check.
Key	string	Yes	The key for which to search.

Returns

Returns `true` or `false`, depending on whether the key exists in the bag.

Example

```
print bag = dynamic({ "foo": 42, "bar": true })
 | extend fooExists = bag_has_key(bag, "foo"),
          blubbExists = bag_has_key(bag, "blubb")

// fooExists: `true`
// blubbExists: `false`
```

Copy Run in Cribl Search

10.5.3.5.2. BAG_KEYS

The `bag_keys` function lists all root keys of a property bag.

Syntax

```
bag_keys( Bag )
```

Parameters

Name	Type	Required	Description
Bag	dynamic	Yes	The property bag whose root keys you want to list.

Returns

Returns an array of strings. The order of the keys is undetermined.

Example

```
print bag = dynamic({ "a": "b", "c": { "d": 123 }})
 | extend listOfKeys = bag_keys(bag)

// returns: ['a', 'c']
```

Copy Run in Cribl Search

10.5.3.5.3. BAG_MERGE

The `bag_merge` function merges two or more property bags, discarding duplicate keys.

Syntax

```
bag_merge( Bag1, Bag2[, ...] )
```

Parameters

Name	Type	Required	Description
Bag	dynamic	Yes	The property bags to merge.

Returns

Returns a single property bag containing the merged results of all input bags.

If the same key is present in multiple input bags, only the value associated with the key from the leftmost argument is kept.

The order of the key-value pairs returned is undetermined.

Example

```
print foo = dynamic({'A1':12, 'B1':2, 'C1':3}), bar = dynamic({'A2':81, 'B2':82, 'A1'
 | project result = bag_merge(foo, bar)
```

Copy Run in Cribl Search

Input (before `bag_merge`):

```
{
  "foo": {
    "A1": 12,
    "B1": 2,
    "C1": 3
  },
  "bar": {
    "A2": 81,
    "B2": 82,
    "A1": 1
  }
}
```

Copy

Output (after `bag_merge`):

```
{
  "result": {
    "A2": 81,
    "B2": 82,
    "A1": 12,
    "B1": 2,
    "C1": 3
  }
}
```

Copy

10.5.3.5.4. BAG_PACK

The `bag_pack` function creates a property bag from an alternating list of keys and values.

Syntax

```
bag_pack( Key1, Value1, Key2, Value2, ... )
```

Parameters

Name	Type	Required	Description
`Key`	`string`	Yes	The key.
`Value`	`dynamic`	Yes	The key's value.

You need to provide an even number of arguments (each `Key` needs a `Value`).

Returns

Returns a single property bag made from the input keys and values.

The order of the key-value pairs returned is undetermined.

Example

```
print bag = bag_pack("Level", "Information", "ProcessID", 1234, "Data", bag_pack("url
```

Copy Run in Cribl Search

Output:

```
{
  "bag": {
    "Level": "Information",
    "ProcessID": 1234,
    "Data": {
      "url": "www.cribl.io"
    }
  }
}
```

Copy

10.5.3.5.5. BAG_PACK_COLUMNS

The `bag_pack_columns` function creates a property bag from a list of fields (columns).

Syntax

```
bag_pack( FieldName[, ...] )
```

Parameters

Name	Type	Required	Description
`FieldName`	Any scalar data type	Yes	The name of a field to add to the output property bag.

Returns

Returns a property bag made from the specified fields. The source field names serve as property names (keys), and the field values as property values.

Example

```
dataset="$vt_dummy"
 | extend foo=1
 | extend bar=2
 | project Packed = bag_pack_columns(foo, bar)
```

Copy Run in Cribl Search

Output:

```
{
  "Packed": {
    "foo": 1,
    "bar": 2
  }
}
```

Copy

10.5.3.5.6. BAG_REMOVE_KEYS

The `bag_remove_keys` function removes key-value pairs from a property bag.

Syntax

```
bag_remove_keys( Bag, Keys )
```

Parameters

Name	Type	Required	Description
`Bag`	`dynamic`	Yes	The property bag from which to remove keys.
`Keys`	`dynamic`	Yes	A key, or a list of keys, to remove.

Returns

Returns the input property bag stripped of the specified keys and their values.

The order of the key-value pairs returned is undetermined.

Example

```
print bag = dynamic({ "foo": 42, "bar": true })
 | project result = bag_remove_keys(bag, "foo")
```

Copy Run in Cribl Search

Input (before `bag_remove_keys`):

```
{
  "bag": {
    "foo": 42,
    "bar": true
  }
}
```

Copy

Output (after `bag_remove_keys`):

```
{
  "result": {
    "bar": true
  }
}
```

Copy

10.5.3.5.7. BAG_SET_KEY

The `bag_set_key` function adds a key-value pair to a property bag, or, if the specified key already exists, overwrites the key's value.

Syntax

```
bag_set_key( Bag, Key, Value )
```

Parameters

Name	Type	Required	Description
`Bag`	`dynamic`	Yes	The property bag to modify.
`Key`	`string`	Yes	The key to set.
`Value`	`dynamic`	Yes	The value to set.

Returns

Returns the input property bag, adding the specified key-value pair if the `Key` doesn't exist, or overwriting the value if the `Key` is already there.

The order of the key-value pairs returned is undetermined.

If the `Bag` specified is not a valid property bag, `bag_set_key` returns `null`.

Examples

Add a Key-Value Pair to a Property Bag

```
print bag = dynamic({ "foo": 42, "bar": true })
 | project result = bag_set_key(bag, "new", 1)
```

Copy Run in Cribl Search

Input (before `bag_set_key`):

```
{
  "bag": {
    "foo": 42,
    "bar": true
  }
}
```

Copy

Output (after `bag_set_key`):

```
{
  "result": {
    "foo": 42,
    "bar": true,
    "new": 1
  }
}
```

Copy

Overwrite a Key-Value Pair in a Property Bag

```
print bag = dynamic({ "foo": 42, "bar": true })
 | project result = bag_set_key(bag, "foo", 100)
```

Copy Run in Cribl Search

Input (before `bag_set_key`):

```
{
  "bag": {
    "foo": 42,
    "bar": true
  }
}
```

Copy

Output (after `bag_set_key`):

```
{
  "result": {
    "foo": 100,
    "bar": true
  }
}
```

Copy

10.5.3.5.8. BAG_ZIP

The `bag_zip` function creates a property bag from two input dynamic arrays. The first array provides keys, the second array provides values.

Syntax

```
bag_zip( KeysArray, ValuesArray )
```

Parameters

Name	Type	Required	Description
`KeysArray`	Array of strings	Yes	Keys to set in the output property bag.
`ValuesArray`	`dynamic`	Yes	Values in the output property bag.

Returns

Returns a property bag where keys come from the first array, and values from the second array. The keys and values are matched in the order in which they are listed in the input (`key1: value1, key2: value2`).

- If there are more keys than values, missing values are filled with `null`.
- If there are more values than keys, values with no matching keys are ignored.
- Keys that aren't strings are ignored.

The order of the key-value pairs returned is undetermined.

Example

```
print foo = dynamic(['key1', 'key2']), bar = dynamic(['value1', 'value2'])
 | project bag_zip(foo, bar)
```

Copy | Run in Cribl Search

Input (before `bag_zip`):

```
{
  "foo": [
    "key1",
    "key2"
  ],
  "bar": [
    "value1",
    "value2"
  ]
}
```

Copy

Output (after `bag_zip`):

```
{
  "bag_zip_foo_bar": {
    "key1": "value1",
    "key2": "value2"
  }
}
```

Copy

10.5.3.5.9. MAKE_BAG

The `make_bag` aggregation function creates a property bag from multiple input bags.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
make_bag( Expression, [ MaxOutputSize ] )
```

Parameters

Name	Type	Required	Description
`Expression`	`dynamic`	Yes	Expression used for the aggregation calculation.
`MaxOutputSize`	`int`	No	Maximum number of elements returned. Default: `1048576`. Max: `2^53`.

Returns

Returns a single property bag, made of those `Expression` values that are property bags (other values are ignored).

If a key appears in the input more than once, only its **first value** is kept.

The order of the key-value pairs returned is undetermined.

Example

```
range v from 1 to 3
 | extend foo = case(v == 1, dynamic({"a": 1, "foo": 42}), v == 2, dynamic({"b": 2, '
 | summarize mb=make_bag(foo)
```

Copy Run in Cribl Search

Input (before `make_bag`):

```
// event 1
{
  "_time": 1735830806.91,
  "v": 1,
  "foo": {
    "a": 1,
    "foo": 42
  }
}

// event 2
{
  "_time": 1735830806.91,
  "v": 2,
  "foo": {
    "b": 2,
    "foo": 17
  }
}

// event 3
{
  "_time": 1735830806.91,
  "v": 3,
  "foo": {
    "c": 3
  }
}
```

Copy

Output (after `make_bag`):

```
{
  "mb": {
    "a": 1,
    "foo": 42,
    "b": 2,
    "c": 3
  }
}
```

Copy

10.5.3.5.10. MAKE_BAG_IF

The `make_bag_if` aggregation function creates a property bag from those input bags that meet the specified condition.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
make_bag_if( Expression, Predicate, [ MaxOutputSize ] )
```

Parameters

Name	Type	Required	Description
`Expression`	`dynamic`	Yes	Expression used for the aggregation calculation.
`Predicate`	`bool`	Yes	Predicate that must evalute to `true` for an input bag to be included in the output.
`MaxOutputSize`	`int`	No	Maximum number of elements returned. Default: `1048576`. Max: `2^53`.

Returns

Returns a single property bag, made of those `Expression` values that are property bags (other values are ignored) and for whose the `Predicate` evaluates to `true`.

If a key appears in the input more than once, only its **first value** is kept.

The order of the key-value pairs returned is undetermined.

Example

```
range v from 1 to 3
 | extend foo = case(v == 1, dynamic({"a": 1, "foo": 42}), v == 2, dynamic({"b": 2, '
 | summarize mb=make_bag_if(foo,v > 1)
```

Copy Run in Cribl Search

Input (before `make_bag_if`):

```
// event 1
{
  "_time": 1735831049.413,
  "v": 1,
  "foo": {
    "a": 1,
    "foo": 42
  }
}

// event 2
{
  "_time": 1735831049.413,
  "v": 2,
  "foo": {
    "b": 2,
    "foo": 17
  }
}

// event 3
{
  "_time": 1735831049.413,
  "v": 3,
  "foo": {
    "c": 3
  }
}
```

Copy

Output (after `make_bag_if`):

```
{
  "mb": {
    "b": 2,
    "foo": 17,
    "c": 3
  }
}
```

Copy

10.5.3.5.11. Zip

The `zip` function merges two or more dynamic arrays, grouping elements of the same index.

Syntax

```
zip( Arrays )
```

Parameters

Name	Type	Required	Description
`Arrays`	`dynamic`	Yes	The dynamic array values to zip.

Returns

Returns an array whose elements are each an array holding the elements of the input arrays of the same index.

Example

```
print foo = zip(dynamic([1,3,5]), dynamic([2,4,6]))
```

Copy Run in Cribl Search

Output:

```
{
  "foo": [
    [
      1,
      2
    ],
    [
      3,
      4
    ],
    [
      5,
      6
    ]
  ]
}
```

Copy

10.5.3.6. Cryptographic Functions

A list of all cryptographic functions supported by Cribl Search.

Name	Description
`encrypt`	Encrypts data with a key managed by a Cribl Stream Worker Group.
`decrypt`	Decrypts data with a key managed by a Cribl Stream Worker Group.

10.5.3.6.1. Encrypt

The `encrypt` function encrypts data with a key managed by a Cribl Stream Worker Group.

For more information on how to set up encryption keys, see the Cribl Stream docs:

- Encryption of Data in Motion
- Create and Manage Encryption Keys
- C.Crypto – Data Encryption and Decryption

Syntax

```
encrypt(value, workerGroup, keyId)
```

Parameters

Name	Type	Required	Description
`value`	`string`	Yes	A valid KQL expression. This is the data to encrypt.
`workerGroup`	`string`	Yes	The name of the Stream Worker Group that has the encryption key.
`keyId`	`string`	Yes	The ID of the encryption key.

Returns

Returns the input data, encrypted using the specified key from the specified Stream Worker Group, encoded as a Base64 string.

Permissions

You need access to the Stream Worker Group that contains the encryption key.

Example

Encrypt a specific field (`dstport`), and export it to a Cribl Lake Dataset.

```
dataset="cribl_search_sample"
 | limit 1000
 | extend dstport = encrypt(dstport, <workerGroup>, <keyId>)
 | export to lake myLakeDataset
```

Copy

10.5.3.6.2. Decrypt

The `decrypt` function decrypts data with a key managed by a Cribl Stream Worker Group.

For more information on how to set up encryption keys, see the Cribl Stream docs:

- Encryption of Data in Motion
- Create and Manage Encryption Keys
- C.Crypto – Data Encryption and Decryption

Syntax

```
decrypt(value, workerGroup)
```

Parameters

Name	Type	Required	Description
`value`	`string`	Yes	A valid KQL expression, containing data encrypted with a key from the specified `workerGroup`.
`workerGroup`	`string`	Yes	The name of the Stream Worker Group that has the encryption key.

Returns

Returns the input data, decrypted using the key from the specified Stream Worker Group.

Permissions

You need access to the Stream Worker Group that contains the encryption key.

Examples

Get the results of a past search (with ID `1704236905683.wgocax`), and decrypt a specific field (`dstport`).

```
dataset="$vt_results" jobId='1704236905683.wgocax'
 | extend dstport = decrypt(dstport, <workerGroup>)
```

Copy

10.5.3.7. Hash Functions

A list of all hash functions supported by Cribl Search.

Name	Description
`hash`	Returns a hash value for the input value.
`hash_combine`	Combines hash values of two or more hashes.
`hash_many`	Returns a combined hash value of multiple values.
`hash_md5`	Returns an MD5 hash value for the input value.
`hash_sha1`	Returns a SHA1 hash value for the input value.
`hash_sha256`	Returns a SHA-256 hash value for the input value.
`hash_xxhash64`	Returns a 64-bit hash value for the input value.

10.5.3.7.1. Hash

The `hash` function returns a hash value for the input value.

Syntax

```
hash( Source, Mod )
```

Arguments

- **Source**: The value to be hashed.
- **Mod**: An optional modulo value to be applied to the hash result, so that the output value is between `0` and `Mod - 1`.

Returns

The hash value of **Source**. If **Mod** is specified, the function returns the hash value modulo the value of **Mod**.

Examples

- `hash("World")` returns as `1846988464401551951`
- `hash("World", 100)` returns as `51 (1846988464401551951 % 100)`
- `hash(datetime("2015-01-01"))` returns as `1380966698541616202`

Sampling

You can use the `hash` function for sampling data if the values in one of its columns are uniformly distributed. In the following example, StartTime values are uniformly distributed and the function is used to run a query on 10% of the data.

```
dataset=Dataset
| where hash(StartTime, 10) == 0
| summarize StormCount = count(), TypeOfStorms = dcount(EventType) by State
| top 5 by StormCount desc
```

Copy

10.5.3.7.2. HASH_COMBINE

The `hash_combine` function combines hash values of two or more hashes.

Syntax

```
hash_combine( Hash1 , Hash2 [, Hash3 ...] )
```

Arguments

- **Hash1**: Long value representing the first hash value.
- **Hash2**: Long value representing the second hash value.
- **Hash3**: Long value representing Nth hash value.

Returns

The combined hash value of the given scalars.

Example

```
dataset=myDataset
| extend h1 = hash(value1), h2=hash(value2)
| extend combined = hash_combine(h1, h2)
```

Copy

10.5.3.7.3. HASH_MANY

The `hash_many` function returns a combined hash value of multiple values.

Syntax

```
hash_many( Value1 , Value2 [, ValueN ...] )
```

Arguments

- **Value1**, **Value2**, **ValueN**: Input values that will be hashed together.

Returns

The `hash` function is applied to each of the specified scalars. The resulting hashes are combined into a single hash and returned.

Example

```
dataset=myDataset
| extend combined = hash_many(value1, value2)
```

Copy

10.5.3.7.4. HASH_MD5

The `hash_md5` function returns an MD5 hash value for the input value.

Syntax

```
hash_md5( Source )
```

Arguments

- **Source**: The value to be hashed.

Returns

The MD5 hash value of the given scalar, encoded as a hex string (a string of characters, each two of which represent a single Hex number between 0 and 255).

Examples

- `h1=hash_md5("World")`
- `h2=hash_md5(datetime(2020-01-01))`

Aggregate by hash

This example uses the `hash_md5` function to aggregate StormEvents based on the State's MD5 hash value.

```
dataset=StormEvents
| summarize StormCount = count() by State, StateHash=hash_md5(State)
| top 5 by StormCount
```

Copy

Returns the following:

State	StateHash	StormCount
TEXAS	3b00dbe6e07e7485a1c12d36c8e9910a	4701
KANSAS	e1338d0ac8be43846cf9ae967bd02e7f	3166

State	StateHash	StormCount
IOWA	6d4a7c02942f093576149db764d4e2d2	2337
ILLINOIS	8c00d9e0b3fcd55aed5657e42cc40cf1	2022
MISSOURI	2d82f0c963c0763012b2539d469e5008	2016

10.5.3.7.5. HASH_SHA1

The `hash_sha1` function returns a SHA1 hash value for the input value.

Syntax

```
hash_sha1( Source )
```

Arguments

- **Source**: The value to be hashed.

Returns

The SHA1 hash value of the given scalar, encoded as a hex string (a string of characters, each two of which represent a single Hex number between 0 and 255).

Examples

- `h1=hash_sha1("World")`
- `h2=hash_sha1(datetime(2020-01-01))`

Aggregate by hash

This example uses the `hash_sha1` function to aggregate StormEvents based on the State's SHA1 hash value.

```
dataset=StormEvents
| summarize StormCount = count() by State, StateHash=hash_sha1(State)
| top 5 by StormCount desc
```

Copy

Returns the following:

State	StateHash	StormCount
TEXAS	3128d805194d4e6141766cc846778eeacb12e3ea	4701
KANSAS	ea926e17098148921e472b1a760cd5a8117e84d6	3166

State	StateHash	StormCount
IOWA	cacf86ec119cfd5b574bde5b59604774de3273db	2337
ILLINOIS	03740763b16dae9d799097f51623fe635d8c4852	2022
MISSOURI	26d938907240121b54d9e039473dacc96e712f61	2016

10.5.3.7.6. HASH_SHA256

The `hash_sha256` function returns a SHA-256 hash value for the input value.

Syntax

```
hash_sha256( Source )
```

Arguments

- **Source**: The value to be hashed.

Returns

The SHA-256 hash value of the given scalar, encoded as a hex string (a string of characters, each two of which represent a single Hex number between 0 and 255).

Examples

- `h1=hash_sha256("World")`
- `h2=hash_sha256(datetime(2020-01-01))`

Aggregate by hash

This example uses the `hash_sha256` function to aggregate StormEvents based on the State's SHA-256 hash value.

```
dataset=StormEvents
| summarize StormCount = count() by State, StateHash=hash_sha256(State)
| top 5 by StormCount desc
```

Copy

Returns the following:

State	StateHash	StormCount
TEXAS	9087f20f23f91b5a77e8406846117049029e6798ebbd0d38aea68da73a00ca37	4701

State	StateHash	StormCount
KANSAS	c80e328393541a3181b258cdb4da4d00587c5045e8cf3bb6c8fdb7016b69cc2e	3166
IOWA	f85893dca466f779410f65cd904fdc4622de49e119ad4e7c7e4a291ceed1820b	2337
ILLINOIS	ae3eeabfd7eba3d9a4ccbfed6a9b8cff269dc43255906476282e0184cf81b7fd	2022
MISSOURI	d15dfc28abc3ee73b7d1f664a35980167ca96f6f90e034db2a6525c0b8ba61b1	2016

10.5.3.7.7. HASH_XXHASH64

The `hash_xxhash64` function returns a 64-bit hash value for the input value.

Syntax

```
hash_xxhash64( Source, Mod )
```

Arguments

- **Source**: The value to be hashed.
- **Mod**: An optional modulo value to be applied to the hash result, so that the output value is between `0` and `Mod - 1`.

Returns

The hash value of **Source**. If **Mod** is specified, the function returns the hash value modulo the value of **Mod**.

Examples

- `xxhash64("World")` returns as `1846988464401551951`
- `xxhash64("World", 100)` returns as `51 (1846988464401551951 % 100)`
- `xxhash64(datetime("2015-01-01"))` returns as `1380966698541616202`

10.5.3.8. INET Functions

A list of all INET functions supported by Cribl Search.

Name	Description
`ipv4_compare`	Compares two IPv4 strings.
`ipv4_is_in_range`	Checks if IPv4 string address is in IPv4-prefix notation range.
`ipv4_is_in_any_range`	Checks whether IPv4 string address is in any of the specified IPv4 address ranges.
`ipv4_is_match`	Matches two IPv4 strings.
`ipv4_is_private`	Checks if IPv4 string address belongs to a set of private network IPs.
`ipv4_netmask_suffix`	Returns the value of the IPv4 netmask suffix from IPv4 string address.
`ipv6_compare`	Compares two IPv6 or IPv4 network address strings.
`ipv6_is_match`	Matches two IPv6 or IPv4 network address strings.
`format_bytes`	Converts the input into a string that represents data size.
`format_ipv4`	Parses input with a netmask and returns string representing IPv4 address.
`format_ipv4_mask`	Parses input with a netmask and returns string representing IPv4 address as CIDR notation.

10.5.3.8.1. IPV4_COMPARE

The `ipv4_compare` function compares two IPv4 strings. The two IPv4 strings are parsed and compared while accounting for the combined IP-prefix mask calculated from argument prefixes, and the optional **PrefixMask** argument.

Syntax

```
ipv4_compare( Expr1, Expr2 [, PrefixMask ] )
```

Arguments

- **Expr1, Expr2**: A string expression representing an IPv4 address. Denote a subnet mask with a forward slash `/`.
- **PrefixMask**: An integer from 0 to 32 representing the number of most-significant bits that are taken into account.

Results

- `0`: If the long representation of the first IPv4 string argument is equal to the second IPv4 string argument
- `1`: If the long representation of the first IPv4 string argument is greater than the second IPv4 string argument
- `-1`: If the long representation of the first IPv4 string argument is less than the second IPv4 string argument
- `null`: If conversion for one of the two IPv4 strings wasn't successful.

Examples

This example returns `0`:

```
print ipv4_compare("127.0.0.1", "127.0.0.1")
```

Copy Run in Cribl Search

This example returns `-1`:

```
print ipv4_compare('192.168.1.1', '192.168.1.255')
```

Copy Run in Cribl Search

This example returns `0`:

```
print ipv4_compare('192.168.1.1/24', '192.168.1.255/24')
```

Copy Run in Cribl Search

This example returns `0`:

```
print ipv4_compare('192.168.1.1', '192.168.1.255', 24)`
```

Copy Run in Cribl Search

10.5.3.8.2. IPV4_IS_IN_RANGE

The `ipv4_is_in_range` function checks if IPv4 string address is in IPv4-prefix notation range.

Syntax

```
ipv4_is_in_range( Ipv4Address, Ipv4Range )
```

Arguments

- **Ipv4Address**: A string expression representing an IPv4 address.
- **Ipv4Range**: A string expression representing an IPv4 address with a subnet mask denoted with a forward slash `/`.

Results

- `true`: If the long representation of the first IPv4 string argument is in range of the second IPv4 string argument.
- `false`: Otherwise.
- `null`: If conversion for one of the two IPv4 strings wasn't successful.

Examples

This example returns `true`:

```
print ipv4_is_in_range("127.0.0.1", "127.0.0.1")
```

Copy Run in Cribl Search

This example returns `true`:

```
print ipv4_is_in_range('192.168.1.6', '192.168.1.1/24')
```

Copy Run in Cribl Search

This example returns `false`:

```
print ipv4_is_in_range('192.168.1.1', '192.168.2.1/24')
```

Copy Run in Cribl Search

10.5.3.8.3. IPV4_IS_IN_ANY_RANGE

The `ipv4_is_in_any_range` function checks whether IPv4 string address is in any of the specified IPv4 address ranges.

Syntax

```
ipv4_is_in_any_range( Ipv4Address , Ipv4Range [, Ipv4Range ...] )
```

Arguments

- **Ipv4Address**: A string expression representing an IPv4 address.
- **Ipv4Range**: A string expression representing an IPv4 address with a subnet mask denoted with a forward slash `/`.

Results

- `true`: If the IPv4 address is in the range of any of the specified IPv4 networks.
- `false`: Otherwise.
- `null`: If conversion for one of the two IPv4 strings wasn't successful.

Examples

This example returns `true`:

```
print ipv4_is_in_any_range('192.168.1.6', '192.168.1.1/24', '10.0.0.1/8', '127.1.0.1/
```

Copy Run in Cribl Search

This example returns `false`:

```
print ipv4_is_in_any_range('192.168.1.1', '192.168.2.1/24', '10.0.0.1/8', '127.1.0.1/
```

Copy Run in Cribl Search

10.5.3.8.4. IPV4_IS_MATCH

The `ipv4_is_match` function matches two IPv4 strings. The two IPv4 strings are parsed and compared while accounting for the combined IP-prefix mask calculated from argument prefixes, and the optional **PrefixMask** argument.

Syntax

```
ipv4_is_match( Expr1, Expr2 [, PrefixMask ] )
```

Arguments

- **Expr1**, **Expr2**: A string expression representing an IPv4 address. Denote a subnet mask with a forward slash `/`.
- **PrefixMask**: An integer from 0 to 32 representing the number of most-significant bits that are taken into account.

Results

- `true`: If the long representation of the first IPv4 string argument is equal to the second IPv4 string argument.
- `false`: Otherwise.
- `null`: If conversion for one of the two IPv4 strings wasn't successful.

Examples

This example returns `true`:

```
print ipv4_is_match("127.0.0.1", "127.0.0.1")
```

Copy Run in Cribl Search

This example returns `false`:

```
print ipv4_is_match('192.168.1.1', '192.168.1.255')
```

Copy Run in Cribl Search

This example returns `true`:

```
print ipv4_is_match('192.168.1.1/24', '192.168.1.255/24')
```

Copy Run in Cribl Search

This example returns `true`:

```
print ipv4_is_match('192.168.1.1', '192.168.1.255', 24)
```

Copy Run in Cribl Search

10.5.3.8.5. IPV4_IS_PRIVATE

The `ipv4_is_private` function checks if IPv4 string address belongs to a set of private network IPs.

Private network addresses were originally defined to assist in delaying IPv4 address exhaustion. IP packets originating from or addressed to a private IP address cannot be routed through the public internet.

Syntax

```
ipv4_is_private( Expression )
```

Arguments

- **Expression**: A string expression representing an IPv4 address. Denote a subnet mask with a forward slash `/`.

Results

- `true`: If the IPv4 address belongs to any of the private network ranges.
- `false`: Otherwise.
- `null`: If parsing of the input as IPv4 address string wasn't successful.

Examples

This example returns `true`:

```
print ipv4_is_private('192.168.1.1/24')
```

Copy Run in Cribl Search

This example returns `true`:

```
print ipv4_is_private('10.1.2.3/24')
```

Copy Run in Cribl Search

This example returns `false`:

```
print ipv4_is_private('202.1.2.3')
```

Copy Run in Cribl Search

This example returns `false`:

```
print ipv4_is_private("127.0.0.1")
```

Copy Run in Cribl Search

10.5.3.8.6. IPV4_NETMASK_SUFFIX

The `ipv4_netmask_suffix` function returns the value of the IPv4 netmask suffix from IPv4 string address.

Private network addresses were originally defined to assist in delaying IPv4 address exhaustion. IP packets originating from or addressed to a private IP address cannot be routed through the public internet.

Syntax

```
ipv4_netmask_suffix( Expression )
```

Arguments

- **Expression**: A string expression representing an IPv4 address. Denote a subnet mask with a forward slash `/`.

Results

- The value of the netmask suffix of the IPv4 address. If suffix is not present in the input, a value of `32` (full netmask suffix) is returned.
- `null`: If parsing of the input as IPv4 address string wasn't successful.

Examples

This example returns `24`:

```
print ipv4_netmask_suffix('192.168.1.1/24')
```

Copy Run in Cribl Search

This example returns `32`:

```
print ipv4_netmask_suffix('192.168.1.1')
```

Copy Run in Cribl Search

10.5.3.8.7. IPV6_COMPARE

The `ipv6_compare` function compares two IPv6 or IPv4 network address strings. The two IPv6 strings are parsed and compared while accounting for the combined IP-prefix mask calculated from argument prefixes, and the optional **PrefixMask** argument.

Syntax

```
ipv6_compare( Expr1, Expr2 [, PrefixMask ] )
```

Arguments

- **Expr1, Expr2**: A string expression representing an IPv4 or IPv6 address. Denote a subnet mask with a forward slash `/`.
- **PrefixMask**: An integer from 0 to 128 representing the number of most-significant bits that are taken into account.

Results

- `0`: If the long representation of the first IPv6 string argument is equal to the second IPv6 string argument.
- `1`: If the long representation of the first IPv6 string argument is greater than the second IPv6 string argument.
- `-1`: If the long representation of the first IPv6 string argument is less than the second IPv6 string argument.
- `null`: If conversion for one of the two IPv6 strings wasn't successful.

Examples

This example returns `0`:

```
print ipv6_compare('::ffff:7f00:1', '127.0.0.1')
```

Copy Run in Cribl Search

This example returns `-1`:

```
print ipv6_compare('fe80::85d:e82c:9446:7994', 'fe80::85d:e82c:9446:7995')
```

Copy Run in Cribl Search

This example returns `0`:

```
print ipv6_compare('192.168.1.1/24', '192.168.1.255/24')
```

Copy Run in Cribl Search

This example returns `0`:

```
print ipv6_compare('fe80::85d:e82c:9446:7994/127', 'fe80::85d:e82c:9446:7995/127')
```

Copy Run in Cribl Search

This example returns `0`:

```
print ipv6_compare('fe80::85d:e82c:9446:7994', 'fe80::85d:e82c:9446:7995', 127)
```

Copy Run in Cribl Search

10.5.3.8.8. IPV6_IS_MATCH

The `ipv6_is_match` function matches two IPv6 or IPv4 network address strings. The two IPv6/IPv4 strings are parsed and compared while accounting for the combined IP-prefix mask calculated from argument prefixes, and the optional **PrefixMask** argument.

Syntax

```
ipv6_is_match( Expr1, Expr2 [, PrefixMask ] )
```

Arguments

- **Expr1**, **Expr2**: A string expression representing an IPv6 or IPv4 address. Denote a subnet mask with a forward slash `/`.
- **PrefixMask**: An integer from 0 to 128 representing the number of most-significant bits that are taken into account.

Results

- `true`: If the long representation of the first IPv6/IPv4 string argument is equal to the second IPv6/IPv4 string argument.
- `false`: Otherwise.
- `null`: If conversion for one of the two IP strings wasn't successful.

Examples

This example returns `true`:

```
print ipv6_is_match('::ffff:7f00:1', '127.0.0.1')
```

Copy Run in Cribl Search

This example returns `false`:

```
print ipv6_is_match('fe80::85d:e82c:9446:7994', 'fe80::85d:e82c:9446:7995')
```

Copy Run in Cribl Search

This example returns `true`:

```
print ipv6_is_match('192.168.1.1/24', '192.168.1.255/24')
```

Copy Run in Cribl Search

This example returns `true`:

```
print ipv6_is_match('fe80::85d:e82c:9446:7994/127', 'fe80::85d:e82c:9446:7995/127')
```

Copy Run in Cribl Search

This example returns `true`:

```
print ipv6_is_match('fe80::85d:e82c:9446:7994', 'fe80::85d:e82c:9446:7995', 127)
```

Copy Run in Cribl Search

10.5.3.8.9. FORMAT_BYTES

The `format_bytes` function converts the input into a string that represents data size.

```
// returns a string: "512 MB"
format_bytes(536870912)
```

Copy

Purpose

Use `format_bytes` to display large, raw byte values in a user-friendly way.

Syntax

```
format_bytes( ByteCount [, Precision [, TargetUnit ] ] )
```

Parameters

Name	Type	Required	Description
`ByteCount`	`int`	Yes	The value to convert into a data-size string.
`Precision`	`int`	No	The number of digits that the input value is rounded to after the decimal point. Default: `0`.
`TargetUnit`	`string`	No	The unit of the output data size: `Bytes`, `KB`, `MB`, `GB`, `TB`, `PB`, or `EB`. If empty, Cribl Search selects the unit, based on input value.

Returns

Returns a `string` that represents data size, using the specified (or automatically selected) unit.

If the conversion is unsuccessful, the function returns an empty string.

Examples

```
// returns "564 Bytes"
print format_bytes(564)
```

Copy Run in Cribl Search

```
// returns "10.1 KB"
print format_bytes(10332, 1)
```

Copy Run in Cribl Search

```
// returns "19 MB"
print format_bytes(20010332)
```

Copy Run in Cribl Search

```
// returns "19.08 MB"
print format_bytes(20010332, 2)
```

Copy Run in Cribl Search

```
// returns "19541 KB"
print format_bytes(20010332, 0, "KB")
```

Copy Run in Cribl Search

10.5.3.8.10. FORMAT_IPV4

The `format_ipv4` function parses input with a netmask and returns string representing IPv4 address.

Syntax

```
format_ipv4( Expression [, PrefixMask ] )
```

Arguments

- **Expression**: A string or number representation (in big-endian order) of the IPv4 address.
- **PrefixMask**: An integer from 0 to 32 representing the number of most-significant bits that are taken into account. If argument isn't specified, all bit-masks are used (32).

Results

If the conversion is successful, returns a `string` representing IPv4 address.

If the conversion fails, returns an empty `string`.

Examples

This example returns `"192.168.1.0"`:

```
print format_ipv4('192.168.1.255', 24)
```

Copy Run in Cribl Search

This example returns `"192.168.1.0"`:

```
print format_ipv4(3232236031, 24)
```

Copy Run in Cribl Search

10.5.3.8.11. FORMAT_IPV4_MASK

The `format_ipv4_mask` function parses input with a netmask and returns string representing IPv4 address as CIDR notation.

Syntax

```
format_ipv4_mask( Expression [, PrefixMask ] )
```

Arguments

- **Expression**: A string or number representation (in big-endian order) of the IPv4 address as CIDR notation.
- **PrefixMask**: An integer from 0 to 32 representing the number of most-significant bits that are taken into account. If argument isn't specified, all bit-masks are used (32).

Results

If the conversion is successful, returns a `string` representing the IPv4 address in CIDR notation.

If the conversion fails, returns an empty `string`.

Examples

This example returns `"192.168.1.0/24"`:

```
print format_ipv4_mask('192.168.1.255', 24)
```

Copy Run in Cribl Search

This example returns `"192.168.1.0/24"`:

```
print format_ipv4_mask(3232236031, 24)
```

Copy Run in Cribl Search

10.5.3.9. Mathematical Functions

A list of all mathematical functions supported by Cribl Search.

Name	Description
`abs`	Calculates the absolute value of the input.
`acos`	Calculates the angle whose cosine is the specified number.
`asin`	Calculates the angle whose sine is the specified number.
`atan`	Returns the angle whose tangent is the specified number.
`atan2`	Calculates the angle, in radians, between the positive x-axis and the ray from the origin to the point (y, x).
`beta_cdf`	Returns the standard cumulative beta distribution function.
`beta_inv`	Returns the inverse of the beta cumulative probability beta density function.
`beta_pdf`	Returns the probability density beta function.
`ceil`	Rounds up a numeric expression's value to the nearest integer.
`ceiling`	Rounds up a numeric expression's value to the nearest integer.
`cos`	Returns the cosine function.
`cot`	Calculates the trigonometric cotangent of the specified angle, in radians.
`degrees`	Converts angle value in radians into value in degrees.
`exp`	Calculates the base-e exponential function of x.
`exp2`	Calculates the base-2 exponential function of x.
`exp10`	Calculates the base-10 exponential function of x.
`gamma`	Computes gamma function.
`isfinite`	Returns whether input is a finite value.
`isinf`	Returns whether input is an infinite value.

Name	Description
isnan	Returns whether input is Not-a-Number (NaN) value.
log	Returns the natural logarithm function.
log2	Returns the (base-2) logarithm function.
log10	Returns the common (base-10) logarithm function.
loggamma	Computes log of absolute value of the loggamma function.
not	Reverses the value of its boolean argument.
pi	Returns the constant value of Pi.
pow	Returns a result of raising to power.
radians	Converts angle value in degrees into value in radians.
rand	Returns a random number.
range	Generates a dynamic array, holding a series of equally spaced values.
round	Returns the rounded source to the specified precision.
sign	Returns the sign of a numeric expression.
sin	Returns the sine of a numeric expression.
sqrt	Returns the square root function.
tan	Returns the tangent function.

10.5.3.9.1. Abs

The `abs` function calculates the absolute value of the input.

Syntax

```
abs( X )
```

Arguments

- X: An integer or real number, or a timespan value.

Returns

Absolute value of x.

Examples

```
print num = -9999999
| extend newField = abs( num )
```

Copy | Run in Cribl Search

10.5.3.9.2. Acos

The `acos` function calculates the angle whose cosine is the specified number (the inverse operation of `cos`).

Syntax

```
acos( X )
```

Arguments

- **X**: A real number in range [-1, 1].

Results

The value of the arc cosine of **X**. Returns `null` if **X** < -1 or **X** > 1

Examples

This example returns `3.141592653589793`:

```
print num = -1
| extend newField = acos( num )
```

Copy | Run in Cribl Search

10.5.3.9.3. Asin

The `asin` function calculates the angle whose sine is the specified number (the inverse operation of `sin`).

Syntax

```
asin( X )
```

Arguments

- **X**: A real number in range [-1, 1].

Returns

Returns the value of the arc sine of **X**. Returns `null` if **X** < -1 or **X** > 1.

Examples

This example returns `1.5707963267948966`;

```
print num = 1
| extend newField = asin( num )
```

Copy Run in Cribl Search

10.5.3.9.4. Atan

The `atan` function returns the angle whose tangent is the specified number (the inverse operation of `tan`).

Syntax

```
atan( X )
```

Arguments

- **X**: A real number.

Returns

The value of the arc tangent of **X**.

Examples

This example returns `1.5485777614681775`:

```
print num = 45
| extend newField = atan( num )
```

Copy Run in Cribl Search

10.5.3.9.5. ATAN2

The `atan2` function calculates the angle, in radians, between the positive x-axis and the ray from the origin to the point (y, x).

Syntax

```
atan2( Y, X )
```

Arguments

- **X**: X coordinate (a real number).
- **Y**: Y coordinate (a real number).

Returns

Returns the angle in radians between the positive x-axis and the ray from the origin to the point (y, x).

Examples

This example returns `1.571`:

```
print atan2( 1,0 )
```

Copy | Run in Cribl Search

10.5.3.9.6. BETA_CDF

The `beta_cdf` function returns the standard cumulative beta distribution function.

The beta distribution is commonly used to study variation in the percentage of something across samples, such as the fraction of the day people spend watching television.

Syntax

```
beta_cdf( X, Alpha, Beta )
```

Arguments

- **X**: A value at which to evaluate the function.
- **Alpha**: A parameter of the distribution.
- **Beta**: A parameter of the distribution.

Returns

The cumulative beta distribution function.

Note:

- If any argument is nonnumeric, beta_cdf() returns null value.
- If x < 0 or x > 1, beta_cdf() returns NaN value.
- If alpha ≤ 0 or alpha > 10000, beta_cdf() returns the NaN value.
- If beta ≤ 0 or beta > 10000, beta_cdf() returns the NaN value.

Examples

```
dataset="cribl_search_sample" dataSource="access_combined"
| limit 100
| summarize combinations = count() by host, clientip
| extend betaDistrib = beta_cdf( combinations, 10, 20 )
```

Copy Run in Cribl Search

10.5.3.9.7. BETA_INV

The `beta_inv` function returns the inverse of the beta cumulative probability beta density function.

The beta distribution can be used in project planning to model probable completion times given an expected completion time and variability.

Syntax

```
beta_inv( Probability, Alpha, Beta )
```

Arguments

- **Probability**: A probability associated with the beta distribution.
- **Alpha**: A parameter of the distribution.
- **Beta**: A parameter of the distribution.

Returns

The inverse of the beta cumulative probability density function beta_cdf.

Note:

- If any argument is nonnumeric, **beta_inv** returns null value.
- If alpha ≤ 0 or beta ≤ 0, **beta_inv** returns the null value.
- If probability ≤ 0 or probability > 1, **beta_inv** returns the NaN value.
- Given a value for probability, **beta_inv** seeks that value x such that beta_cdf(x, alpha, beta) = probability.

Examples

```
dataset="cribl_search_sample" dataSource="access_combined"
| limit 100
| summarize combinations = count() by host, clientip
| extend invBetaDensity = beta_inv( combinations, 10, 20 )
```

Copy Run in Cribl Search

10.5.3.9.8. BETA_PDF

The `beta_pdf` function returns the probability density beta function.

The beta distribution is commonly used to study variation in the percentage of something across samples, such as the fraction of the day people spend watching television.

Syntax

```
beta_pdf( X, Alpha, Beta )
```

Arguments

- **X**: A value at which to evaluate the function.
- **Alpha**: A parameter of the distribution.
- **Beta**: A parameter of the distribution.

Returns

The probability beta density function.

Note:

- If any argument is nonnumeric, beta_pdf() returns null value.
- If x ≤ 0 or 1 ≤ x, beta_pdf() returns NaN value.
- If alpha ≤ 0 or beta ≤ 0, beta_pdf() returns the NaN value.

Examples

```
dataset="cribl_search_sample" dataSource="access_combined"
| limit 100
| summarize combinations = count() by host, clientip
| extend probBetaDensity = beta_pdf( combinations, 10, 20 )
```

Copy Run in Cribl Search

10.5.3.9.9. Ceil

The `ceil` function rounds up an input numeric expression's value to the nearest matching integer.

Alias: `ceiling` (`ceiling` and `ceil` are synonyms).

Syntax

```
ceil(number)
```

Arguments

Name	Type	Required	Description
number	int, long, or real	Yes	The value to round up.

Results

The smallest integer equal to, or greater than, the value of the specified numeric expression.

Examples

This example returns `100`:

```
print num = 99.01
| extend newField = ceil( num )
```

Copy Run in Cribl Search

10.5.3.9.10. Ceiling

The `ceiling` function rounds up an input numeric expression's value to the nearest matching integer.

Alias: `ceil` (`ceil` and `ceiling` are synonyms).

Syntax

```
ceiling(number)
```

Arguments

Name	Type	Required	Description
number	int, long, or real	Yes	The value to round up.

Results

The smallest integer equal to, or greater than, the value of the specified numeric expression.

Examples

This example returns `100`:

```
print num = 99.01
| extend newField = ceiling( num )
```

Copy Run in Cribl Search

10.5.3.9.11. Cos

The `cos` function returns the cosine of the specified numeric expression.

Syntax

```
cos( X )
```

Arguments

- X: A real number.

Examples

This example returns `0.5403023058681398`:

```
print num = 1
| extend newField = cos( num )
```

Copy | Run in Cribl Search

10.5.3.9.12. Cot

The `cot` function calculates the trigonometric cotangent of the specified angle, in radians.

Syntax

```
cot( X )
```

Arguments

- **X**: A real number.

Results

The cotangent function value for **X**.

Examples

This example returns `0.6173696237835551`:

```
print num = 45
| extend newField = cot( num )
```

Copy | Run in Cribl Search

10.5.3.9.13. Degrees

The `degrees` function converts angle value in radians into value in degrees, using formula `degrees = (180 / PI ) * angle_in_radians`.

Syntax

```
degrees( X )
```

Arguments

- X: Angle in radians (a real number).

Results

The corresponding angle in degrees for an angle specified in radians.

Examples

This example returns approximately `360` degrees:

```
print num = 6.283
| extend newField = degrees( num )
```

Copy | Run in Cribl Search

10.5.3.9.14. Exp

The `exp` function calculates the base-e exponential function of x, which is e raised to the power x: `e^x`.

Syntax

```
exp( X )
```

Arguments

- **X**: A real number, value of the exponent.

Results

Exponential value of **X**.

Examples

This example returns `54.598`:

```
print exp( 4 )
```

Copy | Run in Cribl Search

10.5.3.9.15. Exp2

The `exp2` function calculates the base-2 exponential function of x, which is 2 raised to the power x: `2^x`.

Syntax

```
exp2( X )
```

Arguments

- **X**: A real number, value of the exponent.

Results

Exponential value of **X**.

Examples

This example returns `16`:

```
print exp2( 4 )
```

Copy | Run in Cribl Search

10.5.3.9.16. Exp10

The `exp10` function calculates the base-10 exponential function of x, which is 10 raised to the power x: `10^x`.

Syntax

```
exp10( X )
```

Arguments

- **X**: A real number, value of the exponent.

Results

Exponential value of **X**.

Examples

This example returns `10,000`:

```
print exp10( 4 )
```

Copy | Run in Cribl Search

10.5.3.9.17. GAMMA

The `gamma` function computes gamma function.

Syntax

```
gamma( X )
```

Arguments

- **X**: Parameter for the gamma function.

Results

Gamma function of **X**.

Examples

This example returns `6`:

```
print gamma( 4 )
```

Copy Run in Cribl Search

10.5.3.9.18. ISFINITE

The `isfinite` function returns whether input is a finite value (is neither infinite nor NaN).

Syntax

```
isfinite( X )
```

Arguments

- **X**: A real number.

Results

A non-zero value (true) if **X** is finite, and zero (false) otherwise.

Examples

This returns `true`:

```
print isfinite( 420 )
```

Copy Run in Cribl Search

This returns `false`:

```
print isfinite( hello )
```

Copy Run in Cribl Search

10.5.3.9.19. ISINF

The `isinf` function returns whether input is an infinite (positive or negative) value.

Syntax

```
isinf( X )
```

Arguments

- **X**: A real number.

Results

A non-zero value (true) if **X** is a positive or negative infinite, and zero (false) otherwise.

Examples

This returns `false`:

```
print isinf( 420 )
```

Copy Run in Cribl Search

This returns `true`:

```
print x = 42/0
| extend y = isinf( x )
```

Copy Run in Cribl Search

This returns `true`:

```
print isinf( world )
```

Copy Run in Cribl Search

10.5.3.9.20. ISNAN

The `isnan` function returns whether input is Not-a-Number (NaN) value.

Syntax

```
isnan( X )
```

Arguments

- **X**: A real number.

Results

A non-zero value (true) if **X** is NaN, and zero (false) otherwise.

Examples

This returns `false`:

```
print isnan( 42 )
```

Copy Run in Cribl Search

This returns `true`:

```
print x = toint( 'pizza' )
| extend y = isnan( x )
```

Copy Run in Cribl Search

10.5.3.9.21. Log

The `log` function returns the natural logarithm function.

Syntax

```
log( X )
```

Arguments

- X: A real number greater than 0.

Results

- The natural logarithm is the base-e logarithm: the inverse of the natural exponential function (exp).
- `null` if the argument is negative or null or can't be converted to a `real` value.

Examples

This returns `4`:

```
print log( 54.598 )
```

Copy | Run in Cribl Search

10.5.3.9.22. Log2

The `log2` function returns the (base-2) logarithm function.

Syntax

```
log2( X )
```

Arguments

- X: A real number greater than 0.

Results

- The logarithm is the base-2 logarithm: the inverse of the exponential function (exp) with base 2.
- `null` if the argument is negative or null or can't be converted to a `real` value.

Examples

This returns `4`:

```
print log2( 16 )
```

Copy | Run in Cribl Search

10.5.3.9.23. Log10

The `log10` function returns the common (base-10) logarithm function.

Syntax

```
log10( X )
```

Arguments

- X: A real number greater than 0.

Results

- The common logarithm is the base-10 logarithm: the inverse of the exponential function (exp) with base 10.
- `null` if the argument is negative or null or can't be converted to a `real` value.

Examples

This returns `4`:

```
print log10( 10000 )
```

Copy | Run in Cribl Search

10.5.3.9.24. LOGGAMMA

The `loggamma` function computes log of absolute value of the loggamma function.

Syntax

```
loggamma( X )
```

Arguments

- X: Parameter for the gamma function.

Results

Returns the natural logarithm of the absolute value of the gamma function of X.

Examples

This returns `4.787`:

```
print loggamma( 6 )
```

Copy Run in Cribl Search

10.5.3.9.25. Not

The `not` function reverses the value of its boolean argument.

Syntax

```
not( Expression )
```

Arguments

- **Expression**: A boolean expression to be reversed.

Results

Returns the reversed logical value of its boolean argument.

If the argument is `null`, or cannot be converted to `bool` (using the `tobool` function), returns `null`.

Examples

These examples return `false`:

```
print not( true )
```

Copy Run in Cribl Search

```
print not( "true" )
```

Copy Run in Cribl Search

```
print not( 1 )
```

Copy Run in Cribl Search

```
print not( 100 )
```

Copy Run in Cribl Search

These examples return `true`:

```
print not( false )
```

Copy Run in Cribl Search

```
print not( "false" )
```

Copy Run in Cribl Search

```
print not( 0 )
```

Copy Run in Cribl Search

These examples return `null`:

```
print not( "abc" )
```

Copy Run in Cribl Search

```
print not( null )
```

Copy Run in Cribl Search

10.5.3.9.26. Pi

The `pi` function returns the constant value of Pi.

Syntax

```
pi()
```

Results

The double value of Pi (3.1415926...).

Examples

This returns a rounded `3.142`:

```
print pi()
```

Copy Run in Cribl Search

10.5.3.9.27. Pow

The `pow` function returns a result of raising to power.

Syntax

```
pow( Base, Exponent )
```

Arguments

- **Base**: Base value.
- **Exponent**: Exponent value.

Results

Returns base raised to the power exponent: **Base** ^ **Exponent**.

Examples

This returns `387,420,489`:

```
print pow( 9,9 )
```

Copy Run in Cribl Search

10.5.3.9.28. RADIANS

The `radians` function converts angle value in degrees into value in radians, using formula `radians = (PI / 180 ) * angle_in_degrees`.

Syntax

```
radians( X )
```

Arguments

- X: Angle in degrees (a real number).

Results

The corresponding angle in radians for an angle specified in degrees.

Examples

This example returns an approximation of π:

```
print num = 180
| extend newField = radians( num )
```

Copy | Run in Cribl Search

10.5.3.9.29. RAND

The `rand` function returns a random number.

Syntax

- `rand()` - returns a value of type `real` with a uniform distribution in the range [0.0, 1.0).
- `rand( N )` - returns a value of type `real` chosen with a uniform distribution from the set {0.0, 1.0, ..., N - 1}.

Examples

```
print rand()
```

Copy Run in Cribl Search

```
print rand( 1000 )
```

Copy Run in Cribl Search

10.5.3.9.30. RANGE

The `range` function generates a dynamic array, holding a series of equally spaced values.



To generate a series of events, use the `range` operator.

Syntax

```
range( Start, Stop[, Step] )
```

Arguments

- **Start**: The value of the first element in the resulting array.
- **Stop**: The value of the last element in the resulting array, or the least value that is greater than the last element in the resulting array and within an integer multiple of step from start.
- **Step**: The difference between two consecutive elements of the array. The default value for step is `1` for numeric and `1h` for `timespan` or `datetime`.

Results

Dynamic array whose values are: `Start`, `Start + Step`, ... up to and including `Stop`. The array will be truncated if the maximum number of values is reached.

Examples

This example returns `[1, 4, 7]`

```
print range( 1, 8, 3 )
```

Copy | Run in Cribl Search

This example returns an array holding all days in the year 2015, as epoch times:

```
print range( datetime(2015-01-01), datetime(2015-12-31), 1d )
```

Copy | Run in Cribl Search

This example returns `[1,2,3]`:

```
print range( 1, 3 )
```

Copy Run in Cribl Search

This example returns `1`:

```
print range( 1,1000000000 ) | mv-expand r | count
```

Copy Run in Cribl Search

10.5.3.9.31. Round

The `round` function returns the rounded source to the specified precision.

Syntax

```
round( Source [, Precision] )
```

Arguments

- **Source**: The source scalar the round is calculated on.
- **Precision**: Number of digits the source will be rounded to. (default value is 0)

Results

The rounded source to the specified precision.

Round is different than `bin` and `floor` in that the first rounds a number to a specific number of digits while the last rounds value to an integer multiple of a given bin size (round(2.15, 1) returns 2.2 while bin(2.15, 1) returns 2).

Examples

This example returns `2.2`:

```
print round( 2.15, 1 )
```

Copy Run in Cribl Search

This example returns `2`. It's equivalent to `print round( 2.15, 0 )`:

```
print round( 2.15 )
```

Copy Run in Cribl Search

This example returns `-100`:

```
print round( -50.55, -2 )
```

Copy Run in Cribl Search

This example returns `20`:

```
print round( 21.5, -1 )
```

Copy Run in Cribl Search

10.5.3.9.32. Sign

The `sign` function returns the sign of a numeric expression.

Syntax

```
sign( X )
```

Arguments

- X: A real number.

Results

The positive (+1) zero (0) or negative (-1) sign of the specified expression.

Examples

This returns `1` for positive:

```
print sign ( 10000 )
```

Copy Run in Cribl Search

This returns `-1` for negative:

```
print sign ( -9999 )
```

Copy Run in Cribl Search

10.5.3.9.33. Sin

The `sin` function returns the sine of a numeric expression.

Syntax

```
sin( X )
```

Arguments

- X: A real number.

Examples

This example returns `0.9092974268256817`:

```
print num = 2
| extend newField = sin( num )
```

Copy Run in Cribl Search

10.5.3.9.34. Sqrt

The `sqrt` function returns the square root function.

Syntax

```
sqrt( X )
```

Arguments

- X: A real number >= 0.

Returns

- A positive number such that `sqrt(x) * sqrt(x) == x`.
- `null` if the argument is negative or cannot be converted to a `real` value.

Examples

This returns `4`:

```
print sqrt ( 16 )
```

Copy Run in Cribl Search

This returns `null`:

```
print sqrt ( -64 )
```

Copy Run in Cribl Search

10.5.3.9.35. Tan

The `tan` function returns the tangent of a numeric expression.

Syntax

```
tan( X )
```

Arguments

- X: A real number.

Examples

This example returns `-1.995200412208242`:

```
print num = 90
| extend newField = tan( num )
```

Copy | Run in Cribl Search

10.5.3.10. String Functions

A list of all string functions supported by Cribl Search.

Name	Description
`base64_decode_toarray`	Decodes a base64 string to an array of long values.
`base64_decode_tostring`	Decodes a base64 string to a UTF-8 string.
`base64_encode_fromarray`	Encodes a base64 string from a bytes array.
`base64_encode_tostring`	Encodes a string as base64 string.
`countof`	Counts occurrences of a substring in a string.
`extract`	Gets a match for an RE2 regular expression from a source string.
`extract_all`	Gets all matches for an RE2 regular expression from a source string.
`extract_json`	Gets a specified element out of a JSON text using a path expression.
`has_any_index`	Gets a match for an RE2 regular expression from a source string.
`indexof`	Reports the zero-based index of the first occurrence of a specified string within the input string.
`isempty`	Returns `true` if the argument is an empty string or is null.
`isnotempty` (`notempty`)	Returns `true` if the argument isn't an empty string, and it isn't null.
`isnotnull` (`notnull`)	Returns `true` if the argument is not null.
`isnull`	Indicates whether the argument evaluates to a null value.
`match_regex`	Searches a text string for a specific pattern defined by a regular expression.
`parse_csv`	Splits a given string representing a single record of comma-separated values.
`parse_ipv4`	Converts IPv4 string to long (signed 64-bit) number representation in big-endian order.

Name	Description
parse_ipv4_mask	Converts the input string of IPv4 and netmask to a signed, 64-bit wide, long number representation in big-endian order.
parse_ipv6	Converts IPv6 or IPv4 string to a canonical IPv6 string representation.
parse_ipv6_mask	Converts IPv6/IPv4 string and netmask to a canonical IPv6 string representation.
parse_json (todynamic)	Interprets a string as a JSON value and returns the value as dynamic.
parse_url	Parses an absolute URL string and returns a dynamic object that contains URL parts.
parse_urlquery	Returns a dynamic object that contains the Query parameters.
parse_version	Converts the input string representation of version to a comparable decimal number.
replace_regex	Replaces all RE2 regular expression matches with another string.
reverse	Reverses the order of the input string.
split	Splits a given string according to a given delimiter.
strcat	Concatenates between 1 and 64 arguments.
strcat_delim	Concatenates between 2 and 64 arguments, with a delimiter.
strcmp	Compares two strings.
strlen	Returns the length, in characters, of the input string.
strrep	Repeats given string specified number of times.
substring	Extracts a substring from a source string starting from some index to the end of the string.
tolower	Converts a string to lower case.
toupper	Converts a string to upper case.
translate	Replaces a set of characters with another set of characters in a given string.
trim	Removes all leading and trailing matches of the specified string or regular expression.

Name	Description
`trim_end`	Removes trailing match of the specified regular expression.
`trim_start`	Removes leading match of the specified regular expression.
`url_decode`	Converts encoded URL into a to regular URL representation.
`url_encode`	Converts characters of the input URL into a format that can be transmitted over the Internet.

10.5.3.10.1. BASE64_DECODE_TOARRAY

The `base64_decode_toarray` function decodes a base64 string to an array of single-character strings.

Syntax

```
base64_decode_toarray( String )
```

Arguments

- **String**: Input base64 string to be decoded from base64 to an array of single-character strings.

Results

Returns an array of single-character strings decoded from a base64 string.

Examples

This example returns an array of letters that spell out “Cribl”:

```
print dec=base64_decode_toarray("Q3JpYmw=")
```

Copy Run in Cribl Search

This example decodes a base64 encoding of an array that began as `['C',"420.5",'b','l']`:

```
print dec=base64_decode_toarray("QzQyMC41Ymw=")
```

Copy Run in Cribl Search

10.5.3.10.2. BASE64_DECODE_TOSTRING

The `base64_decode_tostring` function decodes a base64 string to a UTF-8 string.

Syntax

```
base64_decode_tostring( String )
```

Arguments

- **String**: Input string to be decoded from base64 to UTF-8 string.

Results

Returns UTF-8 **String** decoded from base64 string.

Trying to decode a base64 string that was generated from invalid UTF-8 encoding will return null.

Example

```
print
encoded_string=base64_encode_tostring("Cribl!$#@^%#. foo bar baz "),
decoded_string=base64_decode_tostring(encoded_string)
```

Copy | Run in Cribl Search

10.5.3.10.3. BASE64_ENCODE_FROMARRAY

The `base64_encode_fromarray` function encodes a base64 string from an array of characters or strings.

Syntax

```
base64_encode_fromarray( BytesArray )
```

Arguments

- **BytesArray**: Input array of characters or strings to be concatenated and encoded into base64. (Numbers within the array will be stringified as individual numeric characters.)

Results

Returns the base64 string encoded from the array of characters.

Trying to encode a base64 string from an invalid character array, which was generated from invalid UTF-8–encoded string, will return `null`.

Examples

This example encodes an array of familiar characters into its base64 string equivalent:

```
print enc=base64_encode_fromarray(dynamic(['C',"ri",'b','l']))
```

Copy Run in Cribl Search

This example round-trips the input array to yield `["C","4","2","0",".","5","b","l"]`:

```
print enc=base64_encode_fromarray(dynamic(['C',"420.5",'b','l'])),
dec=base64_decode_toarray(enc)
```

Copy Run in Cribl Search

10.5.3.10.4. BASE64_ENCODE_TOSTRING

The `base64_encode_tostring` function encodes a string as base64 string.

Syntax

```
base64_encode_tostring( String )
```

Arguments

- **String**: Input string to be encoded as base64 string.

Results

Returns the **String** encoded as base64 string.

Example

```
print
encoded_string=base64_encode_tostring("Cribl!$#@^%#. foo bar baz "),
decoded_string=base64_decode_tostring(encoded_string)
```

Copy Run in Cribl Search

10.5.3.10.5. COUNTOF

The `countof` function counts occurrences of a substring in a string. Plain string matches may overlap; regex matches don't.

Syntax

```
countof( Source, Search [, Kind] )
```

Arguments

- **Source**: A string.
- **Search**: The plain string or regular expression to match inside source.
- **Kind**: `normal` or `regex`, defaults to `normal`.

Results

The number of times that the search string can be matched in the container. Plain string matches may overlap; regex matches don't.

Examples

This example returns `3`:

```
print countof("Hello, world!", "l")
```

Copy Run in Cribl Search

This example returns `1`:

```
print countof("Hello, world!", "ll")
```

Copy Run in Cribl Search

The next few examples show the different treatment of overlapping substrings in `normal` versus `regex` searches.

This example returns `3`:

```
print countof("aaa", "a")
```

Copy Run in Cribl Search

This example returns `3` (not `2`!) because it matches all overlapping occurrences:

```
print countof("aaaa", "aa")
```

Copy Run in Cribl Search

This example returns `2`:

```
print countof("ababa", "ab", "normal")
```

Copy Run in Cribl Search

This example returns `2`:

```
print countof("ababa", "aba")
```

Copy Run in Cribl Search

This example returns `1`, because the `regex` argument prevents matches on overlapping occurrences:

```
print countof("ababa", "aba", "regex")
```

Copy Run in Cribl Search

This example returns `2`:

```
print countof("abcabc", "a.c", "regex")
```

Copy Run in Cribl Search

This example returns `6`:

```
print countof("abcdefedcba", "[abc]", "regex")
```

Copy Run in Cribl Search

This example returns 2:

```
print countof("abc1234abcd1234", "[0-9]+", "regex")
```

Copy Run in Cribl Search

10.5.3.10.6. Extract

The `extract` function gets a match for an RE2 regular expression from a source string. Optionally, convert the extracted substring to the indicated type.



To extract fields, field values, or key-value pairs from events, using a wide range of flexible options, see the `extract` operator.

Syntax

```
extract( Regex, CaptureGroup, Source [, TypeLiteral] )
```

Arguments

- **Regex**: An RE2 regular expression.
- **CaptureGroup**: A positive `int` constant indicating the capture group to extract. 0 stands for the entire match, 1 for the value matched by the first '('parenthesis')' in the regular expression, 2 or more for subsequent parentheses.
- **Source**: A string to search.
- **TypeLiteral**: An optional type literal (for example, `typeof(long)`). If provided, the extracted substring is converted to this type.

Results

If regex finds a match in **Source**: the substring matched against the indicated capture group **CaptureGroup**, optionally converted to **TypeLiteral**.

If there's no match: a blank string.

If the type conversion fails: `null`.

Examples

Extract a Single Field Value

This example works with Apache access log events of this form:

```
128.241.220.82 - - [07/Mar/2025:18:30:13 +0000] "GET /product.screen?product_id=Goats
```

Copy

You could extract the `JSESSIONID` value into a field called `sessionId` using this expression:

```
dataset="cribl_search_sample" dataSource="access_combined"
| limit 10000
| extend sessionId = extract(@'JSESSIONID=(.*?)\s', 1, _raw)
```

Copy Run in Cribl Search

Extract and Convert a Capture Group

This example works with VPC Flow Logs events of this form:

```
2 602320997947 eni-7vv3s9v4cng64k53i 10.0.0.138 10.0.0.138 22 3389 6 60 32460 1741637
```

Copy

By default, a Cribl Search Datatype rule parses a `start` field that identifies the session start time – in seconds, in Unix epoch time – represented as a `string`. But this format makes it hard to identify when a session started. Using `extract`, we parse the session start time from the raw log, and also convert the result to a `datetime` object. This creates a new field called `sessionStart`, which represents the start time in human-readable form.

```
 dataset="cribl_search_sample" dataSource="vpcflowlogs"
| limit 1000
// Extract the flow 'start' time and convert it from a string to a datetime object.
| extend sessionStart = extract(@'^(?:[^\s]+\s){10}([^\s]+)', 1, _raw, typeof(datetim
```

Copy Run in Cribl Search

Given the example event shown above, this expression would parse the start value of `1741638523` from the raw log, and show it as `2025-03-10T20:28:43.000Z`.

10.5.3.10.7. EXTRACT_ALL

The `extract_all` function gets all matches for an RE2 regular expression from a source string. Optionally, retrieve a subset of matching groups.

Syntax

```
extract_all(Regex, [CaptureGroups,] Source)
```

Arguments

- **Regex**: A regular expression containing between 1 and 16 capture groups.
    - Example of a valid regex: `@"(\d+)"`
    - Example of an invalid regex: `@"\d+"`
- **CaptureGroups**: A dynamic array constant that indicates the capture group to extract. Valid values are from 1 to the number of capturing groups in the regular expression. Named capture groups are allowed.
    - Named capture group syntax: `?<groupName>`
- **Source**: A string to search.

Results

- If **Regex** finds a match in **Source**: Returns dynamic array including all matches against the indicated capture groups **CaptureGroups**, or all of capturing groups in the **Regex**.
- If number of **CaptureGroups** is 1: The returned array has a single dimension of matched values.
- If number of **CaptureGroups** is more than 1: The returned array is a two-dimensional collection of multi-value matches per **CaptureGroups** selection, or all capture groups present in the regex if **CaptureGroups** is omitted.
- If there's no match: `null`.

Examples

Extract two capture groups

Breaks `interface_id` into two parts, `Family` and `Name`:

```
dataset="cribl_search_sample" | extend interfaceParts=extract_all(@"(?<Family>\w{3})\
```

Copy | Run in Cribl Search

Extract a single capture group

Returns hex-byte representation (two hex-digits) of the GUID. Let Id="82b8be2d-dfa7-4bd1-8f63-24ad26d31449":

```
| extend guid_bytes = extract_all(@"([\da-f]{2})", Id)
```

Copy

ID	guid_bytes
82b8be2d-dfa7-4bd1-8f63-24ad26d31449	["82","b8","be","2d","df","a7","4b","d1","8f","63","24","ad","26","d3",

Extract several capture groups

Uses a regular expression with three capturing groups to split each GUID part into first letter, last letter, and whatever is in the middle. Let Id="82b8be2d-dfa7-4bd1-8f63-24ad26d31449":

```
| extend guid_bytes = extract_all(@"(\w)(\w+)(\w)", Id)
```

Copy

ID	guid_bytes
82b8be2d-dfa7-4bd1-8f63-24ad26d31449	[["8","2b8be2","d"],["d","fa","7"],["4","bd","1"], ["8","f6","3"],["2","4ad26d3144","9"]]

Extract a subset of capture groups

Shows how to select a subset of capturing groups. The regular expression matches the first letter, last letter, and all the rest. The captureGroups parameter is used to select only the first and the last parts. Let Id="82b8be2d-dfa7-4bd1-8f63-24ad26d31449":

```
| extend guid_bytes = extract_all(@"(\w)(\w+)(\w)", dynamic([1,3]), Id)
```

Copy

ID	guid_bytes
82b8be2d-dfa7-4bd1-8f63-24ad26d31449	[["8","d"],["d","7"],["4","1"],["8","3"],["2","9"]]

Using named capture groups

You can use named capture groups of RE2 in `extract_all`. The **CaptureGroups** uses both capture group indexes and named capture group reference to fetch matching values. Let Id="82b8be2d-dfa7-4bd1-8f63-24ad26d31449":

```
| extend guid_bytes = extract_all(@"(?&lt;first>\w)(?&lt;middle>\w+)(?&lt;last>\w)",
```

Copy

ID	guid_bytes
82b8be2d-dfa7-4bd1-8f63-24ad26d31449	[["8","2b8be2","d"],["d","fa","7"],["4","bd","1"],["8","f6","3"],["2","4ad26d3144","9"]]

10.5.3.10.8. EXTRACT_JSON

The `extract_json` function gets a specified element out of a JSON text using a path expression. Optionally convert the extracted string to a specific type.

Syntax

```
extract_json( JsonPath, DataSource, Type )
```

Arguments

- **JsonPath**: A JSONPath string that defines an accessor into the JSON document. The `[ ]` notation and dot `.` notation are equivalent; see the examples below.
- **DataSource**: A JSON document.
- **Type**: An optional type literal (for example, `typeof(long)`). If provided, the extracted value is converted to this type.

Results

This function performs a JSONPath query into dataSource, which contains a valid JSON string, optionally converting that value to another type depending on the third argument.

Examples

Extract response times to integers:

```
dataset="cribl_internal_logs" response_time
| limit 10
| extend foo=extract_json("$.response_time", _raw, typeof(int))
```

Copy | Run in Cribl Search

Extract hosts' available MB to integers:

```
extract_json("$.hosts[1].AvailableMB", EventText, typeof(int))
```

Copy

Same as the previous query, but with equivalent brackets notation:

```
extract_json("$['hosts'][1]['AvailableMB']", EventText, typeof(int))
```

Copy

10.5.3.10.9. HAS_ANY_INDEX

The `has_any_index` function searches an input string for items specified in an input array, and returns the array position (zero-based) of the first item found in the string.

Syntax

```
has_any_index( String, LookupArray )
```

Arguments

- **String**: A string.
- **LookupArray**: Array of scalar or literal expressions to look up. The values should be of type long, integer, double, decimal, string, or guid.

Results

Zero-based index position of the first item in **LookupArray** that is found in **String**.

Examples

This example matches the first lookup item:

```
print has_any_index("this is an example", dynamic(['this', 'example']))
```

Copy Run in Cribl Search

This example matches the last lookup item:

```
print has_any_index("this is an example", dynamic(['not', 'example']))
```

Copy Run in Cribl Search

This example matches none of the lookup items:

```
print has_any_index("this is an example", dynamic(['not', 'found']))
```

Copy Run in Cribl Search

This example matches within the range of integers:

```
print has_any_index("Example number 2", range(1, 3, 1))
```

Copy Run in Cribl Search

This example doesn't match against the empty lookup array:

```
print has_any_index("this is an example", dynamic([]))
```

Copy Run in Cribl Search

10.5.3.10.10. INDEXOF

The `indexof` function reports the zero-based index of the first occurrence of a specified string within the input string.

If the lookup or input string isn't of `string` type, the function forcibly casts the value to `string`.

Syntax

```
indexof( Source, Lookup [, StartIndex [, Length [, Occurrence ]]] )
```

Arguments

- **Source**: Input string.
- **Lookup**: String to look up.
- **StartIndex**: Search start position. A negative value will offset the starting search position from the end of the **Source** by this many steps: `abs(StartIndex)`.
- **Length**: Number of character positions to examine. A value of -1 means unlimited length.
- **Occurrence**: The number of the occurrence. Default is `1`.

Results

Zero-based index position of lookup.

- Returns -1 if the string isn't found in the input.
- For irrelevant inputs (occurrence < 0 or length < -1) - returns `null`.

Examples

This example returns `3`:

```
print indexof("Hello, world!", "lo")
```

Copy Run in Cribl Search

This example returns `17`, because the `StartIndex` value of `5` skips over the first match:

```
print indexof("Hello, world, hello!", "lo", 5)
```

Copy Run in Cribl Search

This example returns `2`:

```
print indexof("abcdefg", "cde")
```

Copy Run in Cribl Search

This example returns `2`:

```
print indexof("abcdefg", "cde", 1, 4)
```

Copy Run in Cribl Search

This example returns `-1` for not found. The search starts from index `1`, but stops after 2 characters, so the full lookup string can't be found:

```
print indexof("abcdefg", "cde", 1, 2)
```

Copy Run in Cribl Search

This example returns `-1` for not found. The `3` value supplied for `StartIndex` is beyond where the lookup string begins:

```
print indexof("abcdefg", "cde", 3, 4)
```

Copy Run in Cribl Search

This example returns `5`. The negative `StartIndex` begins the search at the `b` character, allowing the lookup string to match:

```
print indexof("abcdefg", "cde" ,-5)
```

Copy Run in Cribl Search

This example returns `4`. The first two arguments are forcibly cast to strings `12345` and `5`:

```
print indexof(1234567, 5, 1, 4)
```

Copy Run in Cribl Search

This example returns `2`:

```
print indexof("abcdefg", "cde", 2, -1)
```

Copy Run in Cribl Search

This example returns `9`:

```
print indexof("abcdefgabcdefg", "cde", 1, 11, 2)
```

Copy Run in Cribl Search

This example returns `-1` for not found. The third occurrence of the lookup string is outside the specified range:

```
print indexof("abcdefgabcdefg", "cde", 1, -1, 3)
```

Copy Run in Cribl Search

10.5.3.10.11. ISEMPTY

The `isempty` function returns `true` if the argument is an empty string, array, or object, or is null.

Syntax

```
isempty( [ Value ] )
```

Results

A returned boolean value of `true` indicates that the argument is empty or null.

x	isempty(x)
`""`	true
`"x"`	false
`parse_json("")`	true
`parse_json("[]")`	true
`parse_json("{}")`	true

Examples

Report on missing events:

```
dataset="cribl_internal_logs"
| where isempty(stats.eventsFound)
```

Copy | Run in Cribl Search

Report on empty values in an arbitrary field:

```
dataset=myDataset
| where isempty(fieldName)
```

Copy

10.5.3.10.12. ISNOTEMPTY

The `isnotempty` function returns `true` if the argument isn't an empty string, array, or object, and isn't null.

Alias: `notempty`

Syntax

```
isnotempty( [ Value ] )
```

Results

A returned boolean value of `true` indicates that the argument is not empty or null.

x	isnotempty(x)
`""`	false
`"x"`	true
`parse_json("")`	false
`parse_json("[]")`	false
`parse_json("{}")`	false

Examples

This example reports on discovered events:

```
dataset="cribl_internal_logs"
| where isnotempty(stats.eventsFound)
```

Copy | Run in Cribl Search

This example returns `false`, because the argument is an empty string:

```
isnotempty("")
```

Copy

10.5.3.10.13. ISNOTNULL

The `isnotnull` function returns `true` if the argument is not null.

Alias: `notnull`

Syntax

```
isnotnull( [ Value ] )
```

Example

This example returns `true`:

```
print isnotnull('Hello World!')
```

Copy | Run in Cribl Search

This example would return all events in which a field named `PossiblyNull` is not null:

```
dataset=myDataset
| where isnotnull(PossiblyNull)
```

Copy

10.5.3.10.14. ISNULL

The `isnull` function evaluates its sole argument and returns a `bool` value indicating whether the argument evaluates to a null value.

Syntax

```
isnull( [ Expression ] )
```

Results

True or false, depending on whether the value is or is not null.

`string` values cannot be null. Use isempty to determine whether a value of type `string` is empty.

x	isnull(x)
""	false
“x”	false
parse_json("")	true
parse_json("[]")	false
parse_json("{}")	false

Example

This example returns `true`:

```
print isnull( parse_json("No JSON here") )
```

Copy Run in Cribl Search

10.5.3.10.15. MATCH_REGEX

The `match_regex` function searches a text string for a specific pattern defined by a regular expression. It returns a `bool` value indicating whether the pattern was found in the text. This function is commonly used for pattern matching and validation tasks in text processing and data extraction.

Syntax

```
match_regex(Field, Regex)
```

Arguments

- **Field**: The field or String to match against.
- **Regex**: An RE2 regular expression as a String wrapped in forward slashes, `"/regex/"`.

Results

Returns a `bool` value, where matches are `true` and non-matches are `false`.

Examples

This example checks for source addresses that include the substring `42`:

```
dataset="cribl_search_sample"
| where match_regex(srcaddr, "/42/")
```

Copy | Run in Cribl Search

This example finds the Goats on the farm:

```
match_regex("Goats on a farm.", "/Goats/")
```

Copy

This example looks for the substring `is 2`:

```
match_regex("Number is 2.000000", @'/is (\d+)/')
```

Copy

10.5.3.10.16. PARSE_CSV

The `parse_csv` function splits a given string representing a single record of comma-separated values and returns a string array with these values.

Syntax

```
parse_csv( Source )
```

Arguments

- **Source**: The source string representing a single record of comma-separated values.

Results

A string array that contains the split values.

Embedded line feeds, commas, and quotes may be escaped using the double quotation mark `""`. This function doesn't support multiple records per row (only the first record is taken).

Examples

This example splits a simple comma-separated phrase to a simple array:

```
print theData="Hello, world" | extend parsed_data=parse_csv(theData) | render event
```

Copy | Run in Cribl Search

This example:

```
parse_csv('aa,"b,b,b",cc,"Escaping quotes: ""Title""","line1\nline2"')
```

Copy

...returns:

```
[
  "aa",
  "b,b,b",
  "cc",
  "Escaping quotes: "Title"",
  "line1\nline2"
]
```

Copy

This example:

```
parse_csv('record1,a,b,c\nrecord2,x,y,z')
```

Copy

...returns:

```
[
  "record1",
  "a",
  "b",
  "c"
]
```

Copy

10.5.3.10.17. PARSE_IPV4

The `parse_ipv4` function converts IPv4 string to long (signed 64-bit) number representation in big-endian order.

Syntax

```
parse_ipv4( Expression )
```

Arguments

- **Expression**: String expression representing IPv4 that will be converted to long. Denote a subnet mask with a forward slash `/`.

Results

If the conversion is successful, returns a `long` number.

If the conversion fails, returns `null`.

Examples

This example returns `3232235778`:

```
print theData='192.168.1.2' | extend parsed_data=parse_ipv4(theData) | render event
```

Copy | Run in Cribl Search

This example also returns `3232235776`:

```
print theData='192.168.1.2/24' | extend parsed_data=parse_ipv4(theData) | render ever
```

Copy | Run in Cribl Search

This example returns `2130706433`:

```
parse_ipv4("127.0.0.1")`
```

Copy

This example returns `2130706432`:

```
parse_ipv4("127.0.0.1/24")
```

Copy

This example returns `true`:

```
parse_ipv4('192.1.168.1') < parse_ipv4('192.1.168.2')
```

Copy

10.5.3.10.18. PARSE_IPV4_MASK

The `parse_ipv4_mask` function converts the input string of IPv4 and netmask to a signed, 64-bit wide, long number representation in big-endian order.

Syntax

```
parse_ipv4_mask( Expression, PrefixMask )
```

Arguments

- **Expression**: String expression representing IPv4 that will be converted to long.
- **PrefixMask**: An integer from 0 to 32 representing the number of most-significant bits that are taken into account.

Results

If the type conversion is successful, returns a `long` number.

If the conversion fails, returns `null`.

Examples

This example returns `3232235776`:

```
print theData='192.168.1.2'  | extend theMask=24, parsed_data=parse_ipv4_mask(theDat
```

Copy Run in Cribl Search

This example returns `2130706432`:

```
parse_ipv4_mask("127.0.0.1", 24)
```

Copy

This example returns `parse_ipv4_mask('192.1.168.3', 31)`:

```
parse_ipv4_mask('192.1.168.2', 31)` returns as `parse_ipv4_mask('192.1.168.3', 31)
```

Copy

10.5.3.10.19. PARSE_IPV6

The `parse_ipv6` function converts IPv6 or IPv4 string to a canonical IPv6 string representation.

Syntax

```
parse_ipv6( Expression )
```

Arguments

- **Expression**: String expression representing IPv6/IPv4 network address that will be converted to canonical IPv6 representation. Denote a subnet mask with a forward slash `/`.

Results

If the conversion is successful, returns a `string` representing a canonical IPv6 network address.

If the conversion fails, returns `null`.

Examples

```
print theData='192.168.1.2'  | extend parsed_data=parse_ipv6(theData) | render event
```

Copy | Run in Cribl Search

```
print theData='abcd::ffff:c0a8:0102'  | extend parsed_data=parse_ipv6(theData) | renc
```

Copy | Run in Cribl Search

This example returns `'0000:0000:0000:0000:0000:ffff:7f00:0001'`:

```
parse_ipv6("127.0.0.1")
```

Copy

This example returns `'fe80:0000:0000:0000:085d:e82c:9446:7994'`:

```
parse_ipv6(":fe80::85d:e82c:9446:7994")
```

Copy

10.5.3.10.20. PARSE_IPV6_MASK

The `parse_ipv6_mask` function converts an IPv6/IPv4 string and netmask to a canonical IPv6 string representation.

Syntax

```
parse_ipv6_mask( Expression, PrefixMask )
```

Arguments

- **Expression**: String expression representing IPv6 that will be converted to long.
- **PrefixMask**: An integer from 0 to 128 representing the number of most-significant bits that are taken into account.

Results

If the conversion is successful, returns a `string` representing a canonical IPv4 or IPv6 network address.

If the conversion fails, returns `null`.

Examples

```
print theData='abcd::ffff:c0a8:0102'  | extend theMask=24, parsed_data=parse_ipv6_mas
```

Copy | Run in Cribl Search

This example returns `'0000:0000:0000:0000:0000:ffff:7f00:0000'`:

```
parse_ipv6_mask("127.0.0.1", 24)
```

Copy

This example returns `'fe80:0000:0000:0000:085d:e82c:9446:7900'`:

```
parse_ipv6_mask(":fe80::85d:e82c:9446:7994", 120)
```

Copy

10.5.3.10.21. PARSE_JSON

The `parse_json` function interprets a `string` as a JSON value and returns the value as `dynamic`. Where possible, it converts the value into relevant data types.



For strict parsing with no data type conversion, use the extract or extract_json function instead.

Alias: `todynamic`

This function is better than `extract_json` when you need to extract more than one element of a JSON compound object.

Syntax

```
parse_json( JSON )
```

Arguments

- **JSON**: An expression of type `string`. It represents a JSON-formatted value, or an expression of type dynamic, representing the actual `dynamic` value.

Results

An object of type `dynamic` that is determined by the value of **JSON**:

- If **JSON** is of type `dynamic`, its value is used as-is.
- If **JSON** is of type `string`, and is a properly formatted JSON string, then the string is parsed, and the value produced is returned.
- If **JSON** is of type `string`, but it isn't a properly formatted JSON string, then the returned value is an object of type `dynamic` that holds the original `string` value.

Example

```
print theData='{"theValue":123,"theString":"Hello!"}' | extend parsed_data=parse_jsor
```

Copy Run in Cribl Search

In the following example, when `context_custom_metrics` is a `string` that looks like this:

```
{"duration":{"value":118.0,"count":5.0,"min":100.0,"max":150.0,"stdDev":0.0,"sampled\
```

Copy

...then the following KQL fragment retrieves the value of the `duration` slot in the object, and from that it retrieves two slots, `duration.value` and `duration.min` (`118.0` and `110.0`, respectively):

```
dataset=myDataset
| extend d=parse_json(context_custom_metrics)
| extend duration_value=d.duration.value, duration_min=d["duration"]["min"]
```

Copy

10.5.3.10.22. PARSE_URL

The `parse_url` function parses an absolute URL `string` and returns a `dynamic` object that contains URL parts.

Syntax

```
parse_url( URL )
```

Arguments

- **URL**: A string representing a URL or the query part of the URL.

Results

An object of type `dynamic` that included the URL components: Scheme, Host, Port, Path, Username, Password, Query Parameters, Fragment.

Example

```
print theData="scheme://username:password@the.host.com:1234/this/is/a/path?k1=v1&k2=v
```

Copy Run in Cribl Search

```
print theData="scheme://username:password@the.host.com:1234/this/is/a/path?k1=v1&k2=v
```

Copy Run in Cribl Search

This example:

```
parse_url("scheme://username:password@host:1234/this/is/a/path?k1=v1&k2=v2#fragment")
```

Copy

...returns:

```
{
        "Scheme":"scheme",
        "Host":"host",
        "Port":"1234",
        "Path":"this/is/a/path",
        "Username":"username",
        "Password":"password",
        "Query Parameters":"{"k1":"v1", "k2":"v2"}",
        "Fragment":"fragment"
 }
```


Copy

10.5.3.10.23. PARSE_URLQUERY

The `parse_urlquery` function returns a `dynamic` object that contains URL query parameters from an input string.

Syntax

```
parse_urlquery( Query )
```

Arguments

- **Query**: A string represents a URL query. Format should follow URL query standards `key=value& ....`

Results

An object of type `dynamic` that includes URL query parameters from the input string.

Example

```
print theData="k1=v1&k2=v2&k3=v3" | extend parsed_data=parse_urlquery(theData), theQu
```

Copy | Run in Cribl Search

This example:

```
parse_urlquery("k1=v1&k2=v2&k3=v3")
```

Copy

...returns:

```
{
  "Query Parameters":"{"k1":"v1", "k2":"v2", "k3":"v3"}",
}
```

Copy

10.5.3.10.24. PARSE_VERSION

The `parse_version` function converts the input string representation of a version number to a comparable decimal number.

Syntax

```
parse_version( Expression )
```

Arguments

- **Expression**: A scalar expression of type `string` that specifies the version to be parsed.
    - Input string must contain from one to four version parts, represented as numbers and separated with dots `.`.
    - Each part of version may contain up to eight digits, with the max value at 99999999.
    - If the number of parts is less than four, all the missing parts are considered as trailing (`1.0` == `1.0.0.0`).

Results

If the conversion is successful, returns a decimal.

If the conversion fails, returns `null`.

Example

This example converts the two input strings to decimal values, and then compares the values:

```
print theData="9.2", theData2="9.11.3" | extend v1=parse_version(theData), v2=parse_v
```

Copy Run in Cribl Search

10.5.3.10.25. REPLACE_REGEX

The `replace_regex` function replaces all RE2 regular expression matches with another string.

Syntax

```
replace_regex( Text, Regex, Rewrite )
```

Arguments

- **Text**: A string.
- **Regex**: The RE2 regular expression to search text. The expression can contain capture groups in parentheses.
- **Rewrite**: The replacement regex for any match made by **Regex**. Use `\0` to refer to the whole match, `\1` for the first capture group, `\2` and so on for subsequent capture groups.

Results

Source after replacing all matches of **Regex** with evaluations of **Rewrite**. Matches do not overlap.

Examples

This example replaces the input string's "is" with "was":

```
print theText="Number is 2.000000"| extend theNewText=replace_regex(theText, @'is (\d
```

Copy | Run in Cribl Search

This example returns `Number was: 2.000000`:

```
replace_regex("Number is 2.000000", @'is (\d+)', @'was: \1')
```

Copy

This example returns `Number was: 5.000000`:

```
replace_regex("Number is 5.000000", @'is (\d+)', @'was: \1')
```

Copy

10.5.3.10.26. Reverse

The `reverse` function reverses the order of the input `string`. If the input value isn't of type string, then the function forcibly casts the value to type `string`.

Syntax

```
reverse( Source )
```

Arguments

- **Source**: Input value.

Results

The reverse order of a string value.

Examples

This example returns `olleH`:

```
print str1="Hello" | extend newstr=reverse(str1)
```

Copy | Run in Cribl Search

This example returns `ZYXWVUTSRQPONMLKJIHGFEDCBA`:

```
print reverse("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
```

Copy | Run in Cribl Search

10.5.3.10.27. Split

The `split` function splits a given string according to a given delimiter and returns a string array with the contained substrings. Optionally, a specific substring can be returned if it exists.

Syntax

```
split( Source, Delimiter [, RequestedIndex] )
```

Arguments

- **Source**: The source string that will be split according to the given delimiter.
- **Delimiter**: The delimiter that will be used to split the source string.
- **RequestedIndex**: An optional zero-based index `int`. If provided, the returned string array will contain the requested substring if it exists.

Results

A string array that contains the substrings of the given source string that are delimited by the given delimiter.

Examples

- `split("aa_bb", "_")` returns as `["aa","bb"]`
- `split("aaa_bbb_ccc", "_", 1)` returns as `["bbb"]`
- `split("", "_")` returns as `[""]`
- `split("a__b", "_")` returns as `["a","","b"]`
- `split("aabbcc", "bb")` returns as `["aa","cc"]`

Example Query: Split URL Path Segments

```
dataset=$vt_dummy
| extend url = '/api/v1/items/42'
| extend segments = split(url, '/'),
         resource = split(url, '/', 3),   // ["items"]
         id = split(url, '/', 4)          // ["42"]
```

Copy Run in Cribl Search

Example Query: Split Comma-Separated Values

```
dataset=$vt_dummy
| extend line = 'alpha,beta,gamma'
| extend parts = split(line, ','),
         second = split(line, ',', 1)    // ["beta"]
```

Copy Run in Cribl Search

Example Query: Isolate Filename Extension

```
dataset=$vt_dummy
| extend filename = 'awesome.log'
| extend basename = split(filename, '.', 0) // ["awesome"]
```

Copy Run in Cribl Search

10.5.3.10.28. Strcat

The `strcat` function concatenates between 1 and 64 arguments to a single string.

Syntax

```
strcat( Argument1, Argument2 [, ArgumentN ] )
```

Arguments

- **Argument1 ... ArgumentN**: Expressions to be concatenated.

If an argument is not of the `string` type, it's forcibly converted to `string`.

If an argument is `null`, it's converted to a blank string.

Results

Arguments concatenated to a single string.

Usage Note

Concatenate strings using the `strcat` or `strcat-delim` function. Do not not rely on the `+` operator (with `"string1" + "string2"` notation).

Examples

The first two examples return `hello world`:

```
print str1="hello", str2="world" | extend newstr=strcat(str1," ", str2)
```

Copy Run in Cribl Search

```
print strcat("hello", " ", "world")
```

Copy Run in Cribl Search

This example returns `"The value is: "`:

```
print strcat('The value is: ', null)
```

Copy | Run in Cribl Search

10.5.3.10.29. STRCAT_DELIM

The `strcat_delim` function concatenates between 2 and 64 arguments, with the specified delimiter.

Syntax

```
strcat_delim( Delimiter, Argument1, Argument2 [, ArgumentN ] )
```

Arguments

- **Delimiter**: String expression, which will be used as separator.
- **Argument1 ... ArgumentN**: Expressions to be concatenated.

If an argument is not of the `string` type, it's forcibly converted to `string`.

If an argument is `null`, it's converted to a blank string.

Results

Arguments, concatenated to a single string with **Delimiter**.

Usage Note

Concatenate strings using the `strcat-delim` or `strcat` function. Do not not rely on the `+` operator (with `"string1" + "string2"` notation).

Examples

```
print str1="hello", str2="world" | extend newstr=strcat_delim("|",str1,str2)
```

Copy Run in Cribl Search

```
print strcat_delim('-', 1, '2', 'A', 1s)
```

Copy Run in Cribl Search

10.5.3.10.30. STRCMP

The `strcmp` function compares two strings.

The function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until the end of the shorter string is reached.

Syntax

```
strcmp( String1, String2 )
```

Arguments

- **String1**: First input string for comparison.
- **String2**: Second input string for comparison.

Results

Returns an integral value indicating the relationship between the strings:

- `<0` : the first character that does not match has a lower value in **String1** than in **String2**
- `0` : the contents of both strings are equal
- `>0` : the first character that does not match has a greater value in **String1** than in **String2**

Examples

This example returns `-1` for lower value:

```
print str1="ABC", str2="ABCd" | extend match=strcmp(str1,str2)
```

Copy Run in Cribl Search

This example returns `0` for equality:

```
print strcmp(ABC,ABC)
```

Copy Run in Cribl Search

This example returns `1` for greater value:

```
print strcmp(abc,ABC)
```

Copy Run in Cribl Search

This example returns `1` for greater value:

```
print strcmp(abcde,abc)
```

Copy Run in Cribl Search

10.5.3.10.31. Strlen

The `strlen` function returns the length, in characters, of the input string.

Syntax

```
strlen( Source )
```

Arguments

- **Source**:The source string that will be measured for string length.

Results

Returns the length, in characters, of the input string.

This function counts Unicode code points.

Examples

This example returns `5`:

```
print str1="hello" | extend fieldlen=strlen(str1)
```

Copy Run in Cribl Search

10.5.3.10.32. STRREP

The `strrep` function repeats given string specified number of times. If first or third argument is not of a string type, it will be forcibly converted to string.

Syntax

```
strrep( Value, Multiplier, [ Delimiter ] )
```

Arguments

- **Value**: Input expression.
- **Multiplier**: Positive integer value (from 1 to 1024).
- **Delimiter**: An optional string expression (default: empty string).

Results

Value repeated for a specified number of times, concatenated with **Delimiter**.

If **Multiplier** is more than maximum allowed value (1024), input string will be repeated 1024 times.

Examples

This example returns `HelloHelloHello`:

```
print str1="Hello" | extend newstr=strrep(str1,3)
```

Copy Run in Cribl Search

This example returns `Hello-Hello-Hello`:

```
print str1="Hello" | extend newstr=strrep(str1,3,"-")
```

Copy Run in Cribl Search

This example returns `ABCABC`:

```
print strrep('ABC', 2)
```

Copy Run in Cribl Search

This example returns `123.123.123`:

```
print strrep(123,3,'.')
```

Copy Run in Cribl Search

This example returns `3 3`:

```
print strrep(3s,2,' ')
```

Copy Run in Cribl Search

10.5.3.10.33. Substring

The `substring` function extracts a substring from a source string starting from some index to the end of the string.

Optionally, the length of the requested substring can be specified.

Syntax

```
substring( Source, StartingIndex [, Length] )
```

Arguments

- **Source**: The source string that the substring will be taken from.
- **StartingIndex**: The zero-based starting character position of the requested substring. Can be a negative number, in which case the substring will be retrieved from the end of the source string.
- **Length**: An optional parameter that can be used to specify the requested number of characters in the substring.

Results

A substring from the given string. The substring starts at the **StartingIndex** (zero-based) character position and continues to the end of the string (or to **Length** characters, if specified).

Examples

This example returns `23456`:

```
print substring("123456", 1)
```

Copy Run in Cribl Search

This example returns `34`:

```
print substring("123456", 2, 2)
```

Copy Run in Cribl Search

This example returns `AB`:

```
print substring("ABCD", 0, 2)
```

Copy Run in Cribl Search

This example returns `56`:

```
print substring("123456", -2, 2)
```

Copy Run in Cribl Search

10.5.3.10.34. Tolower

The `tolower` function converts a string to lower case.

If the input string is `null`, the function returns a blank string.

Syntax

```
tolower( String )
```

Arguments

- **String**: A string.

Example

This example returns lowercase `hello, world`:

```
print str1="Hello, World" | extend newstr=tolower(str1)
```

Copy | Run in Cribl Search

10.5.3.10.35. Toupper

The `toupper` function converts a string to upper case.

If the input string is `null`, the function returns a blank string.

Syntax

```
toupper( String )
```

Arguments

- **String**: A string.

Example

This example returns uppercase `HELLO, WORLD`:

```
print str1="Hello, World" | extend newstr=toupper(str1)
```

Copy | Run in Cribl Search

10.5.3.10.36. Translate

The `translate` function replaces a set of characters with another set of characters in a given string. The function searches for characters in the `SearchList` and replaces them with the corresponding characters in `ReplacementList`.

Syntax

```
translate( SearchList, ReplacementList, Source )
```

Arguments

- **SearchList**: The list of characters that should be replaced.
- **ReplacementList**: The list of characters that should replace the characters in `SearchList`.
- **Source**: A string to search.

Results

Source after replacing all occurrences of characters in `ReplacementList` with the corresponding characters in `SearchList`.

Examples

This example returns `xxx`:

```
print translate("abc", "x", "abc")
```

Copy Run in Cribl Search

This example returns `kusto`:

```
print translate("krasp", "otsku", "spark")
```

Copy Run in Cribl Search

This example returns `hello world`:

```
print translate("abcdefghijklmopqrstuvwxyz", "zyxwvutsrqpomlkjihgfedcba", "svool dlio
```

Copy | Run in Cribl Search

This example also returns `hello world`, by calling `reverse`:

```
print translate("abcdefghijklmopqrstuvwxyz", reverse("abcdefghijklmopqrstuvwxyz"), "s
```

Copy | Run in Cribl Search

10.5.3.10.37. Trim

The `trim` function removes all leading and trailing matches of the specified string or regular expression.

Syntax

```
trim( {Regex | String}, Source )
```

Arguments

- **Regex**: Regular expression or string to trim from the beginning and/or the end of **Source**. Pass a regular expression in the format `@'<regex>'`; pass a string the format `"<string>"`. If you omit this argument, the function applies regex to match all whitespace.
- **Source**: A string.

Results

Source, after trimming matches of **Regex** or **String** found in the beginning and/or the end of **Source**.

Examples

Add some leading characters with `strcat`, then remove them with `trim`:

```
print str1="ABC"
| extend str2=strcat("  >>",str1),
    thestr=trim("  >>",str2)
```

Copy | Run in Cribl Search

Remove leading and trailing `-` characters, **Regex** first argument:

```
print trim(@"\-+", "---Hello, world!---")
```

Copy | Run in Cribl Search

Automatically remove leading and trailing whitespace, using implicit regex:

```
dataset="$vt_dummy"
| extend string_to_trim = "    Hello, world!    "
| project
    string_to_trim,
    trimmed_string = trim(string_to_trim)
```

Copy Run in Cribl Search

10.5.3.10.38. TRIM_END

The `trim_end` function removes a trailing match of the specified regular expression.

Syntax

```
trim_end( Regex, Source )
```

Arguments

- **Regex**: String or regular expression to be trimmed from the end of **Source**.
- **Source**: A string.

Results

Source after trimming matches of **Regex** found at the end of **Source**.

Example

Remove trailing - characters:

```
print trim_end(@"\-+", "---Hello, world!---")
```

Copy | Run in Cribl Search

10.5.3.10.39. TRIM_START

The `trim_start` function removes a leading match of the specified regular expression.

Syntax

```
trim_start( Regex, Source )
```

Arguments

- **Regex**: String or regular expression to be trimmed from the beginning of **Source**.
- **Source**: A string.

Results

Source after trimming matches of **Regex** found in the beginning of **Source**.

Example

Remove leading `-` characters:

```
print trim_start(@"\-+", "---Hello, world!---")
```

Copy | Run in Cribl Search

10.5.3.10.40. URL_DECODE

The `url_decode` function converts encoded URL into a regular URL representation.

Syntax

```
url_decode( EncodedURL )
```

Arguments

- **EncodedURL**: Encoded URL (string).

Results

URL (string) in a regular representation.

Examples

This example decodes endpoints from API calls:

```
dataset="cribl_internal_logs" source="file:///opt/cribl/log/access.log"
| where match_regex(url, "/filterExp/")
| extend parsedUrl=url_decode(url)
```

Copy Run in Cribl Search

10.5.3.10.41. URL_ENCODE

The `url_encode` function converts characters of the input URL into a format that can be transmitted over the internet.

Syntax

```
url_encode( URL )
```

Arguments

- **URL**: Input URL (string).

Results

URL (string) converted into a format that can be transmitted over the internet.

Examples

This example encodes user names – for example, replacing spaces with their `%20` hexadecimal equivalents:

```
dataset="cribl_internal_logs" source="file:///opt/cribl/log/access.log" user="*"
| extend encodedName=url_encode(user)
```

Copy | Run in Cribl Search

10.5.4. Statistical Functions

Statistical functions are used together with the `summarize`, `eventstats`, and `timestats` operators to aggregate your data. Cribl Search supports the following statistical functions:

Name	Description
`avg`	Calculates the average across the group.
`avgif`	Calculates the average across the group where a predicate evaluates to `true`.
`count`	Counts events per summarization group.
`countif`	Counts events based on a predicate.
`dcount`	Calculates an estimate of the number of distinct values.
`dcountif`	Calculates an estimate of the number of distinct values for those rows where a predicate evaluates to `true`.
`max`	Finds the maximum value across the group.
`maxif`	Finds the maximum value for which a predicate evaluates to `true`.
`min`	Finds the minimum value across the group.
`minif`	Finds the minimum value which a predicate evaluates to `true`.
`percentile`	Returns an estimate for the specified nearest-rank percentile of the population defined.
`stdev`	Calculates the standard deviation of an expression across the group.
`stdevif`	Calculates the standard deviation of an expression which a predicate evaluates to `true`.
`stdevp`	Calculates the standard deviation of an expression across the group, considering the group as a population.
`sum`	Calculates the sum of an expression across the group.
`sumif`	Calculates the sum of an expression for which a predicate evaluates to `true`.
`variance`	Calculates the variance of an expression.
`varianceif`	Calculates the variance of an expression for which a predicate evaluates to `true`.
`variancep`	Calculates the variance of an expression across the group, considering the group as a population.

10.5.4.1. Avg

The `avg` aggregation function calculates the average (arithmetic mean) across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
avg( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Records with `null` values are ignored and not included in the calculation. Wildcards are not supported for field names.

Results

Returns the average value of **Expression** across the group.

Examples

This example summarizes average byte count across the specified number of events:

```
dataset="cribl_search_sample" | limit 1000 | summarize AverageSessionBytes = avg(byte
```

Copy Run in Cribl Search

This example summarizes average byte count, and corresponding (small-sample) standard deviation, by source address:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs" | summarize avg(bytes), stdev(
```

Copy Run in Cribl Search

This example summarizes average byte count, and two measures of variance (sample- and population-based), by source address:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize avg(bytes), variance(bytes), variancep(bytes) by srcaddr
```

10.5.4.2. AVGIF

The `avgif` aggregation function calculates the average of **Expression** across the group where **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
avgif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Records with `null` values are ignored and not included in the calculation. Wildcards are not supported for field names.
- **Predicate**: Predicate that if `true`, the **Expression** calculated value will be added to the average.

Results

Returns the average value of **Expression** across the group where Predicate evaluates to `true`.

Examples

This example returns average byte count from source address `10.0.0.164`:

```
dataset="cribl_search_sample"
| limit 1000
| summarize AverageSessionBytesForSrcAddr=avgif(bytes, srcaddr=="10.0.0.164")
```

Copy Run in Cribl Search

This example returns average byte count by private source address:

```
dataset="cribl_search_sample"
| limit 1000
| summarize AverageSessionBytesForPrivateAddr=avgif(bytes, ipv4_is_private(srcaddr))
```

Copy Run in Cribl Search

10.5.4.3. Count

The `count` aggregation function counts the number of non-null events per summarization group, or the total number of non-null events if summarization is done without grouping.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.



If you need to count boolean `true` values, use the `countif` aggregation function.

Syntax

```
count( [Expression] )
```

Arguments

- **Expression**: Optional. Expression used for aggregation calculation. Wildcards are not supported for field names.

Results

Returns a count of the events per summarization group (or in total, if summarization is done without grouping).

Example

This example summarizes the number of non-null events by source address:

```
dataset="cribl_search_sample"
| limit 1000
| summarize NumberOfEvents=count() by srcaddr
```

Copy | Run in Cribl Search

10.5.4.4. COUNTIF

The `countif` aggregation function counts events based on a predicate.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.



If you need to count non-null values regardless of truthiness, use the `count` aggregation function.

Syntax

```
countif( Predicate )
```

Arguments

- **Predicate**: An expression used for aggregation calculation. Use any scalar expression that returns a `bool` value. Wildcards are not supported for field names.

Results

Returns a count of rows for which **Predicate** evaluates to `true`.

Examples

This example summarizes byte counts (with a minimum value of 11), by source address:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize gtthan10Count=countif(bytes > 10) by srcaddr
```

Copy | Run in Cribl Search

10.5.4.5. DCOUNT

The `dcount` aggregation function calculates an estimate of the number of distinct values that are taken by an expression in the summary group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.



The `dcount()` aggregation function is primarily useful for estimating the cardinality of huge sets. It trades performance for accuracy, and might return a result that varies between executions. The order of inputs might have an effect on its output.

Syntax

```
dcount( Expression[, Accuracy] )
```

Arguments

- **Expression**: A scalar expression whose distinct values are to be counted. Wildcards are not supported for field names.
- **Accuracy**: Optional. An integer that defines the requested estimation accuracy. If unspecified, the default value is `1`.

Results

Returns an estimate of the number of distinct values of **Expression** in the group.

Example

This example summarizes the estimated cardinality of destination ports by destination address:

```
dataset="cribl_search_sample"
| summarize distinctCountPorts=dcount(dstport) by dstaddr
```

Copy | Run in Cribl Search

Estimation Accuracy

This function uses a variant of the HyperLogLog (HLL) algorithm, which does a stochastic estimation of set cardinality. The algorithm provides a knob that can be used to balance accuracy and execution time per

memory size:

Accuracy	Error (%)	Entry count
0	1.6	212
1	0.8	214
2	0.4	216
3	0.28	217
4	0.2	218



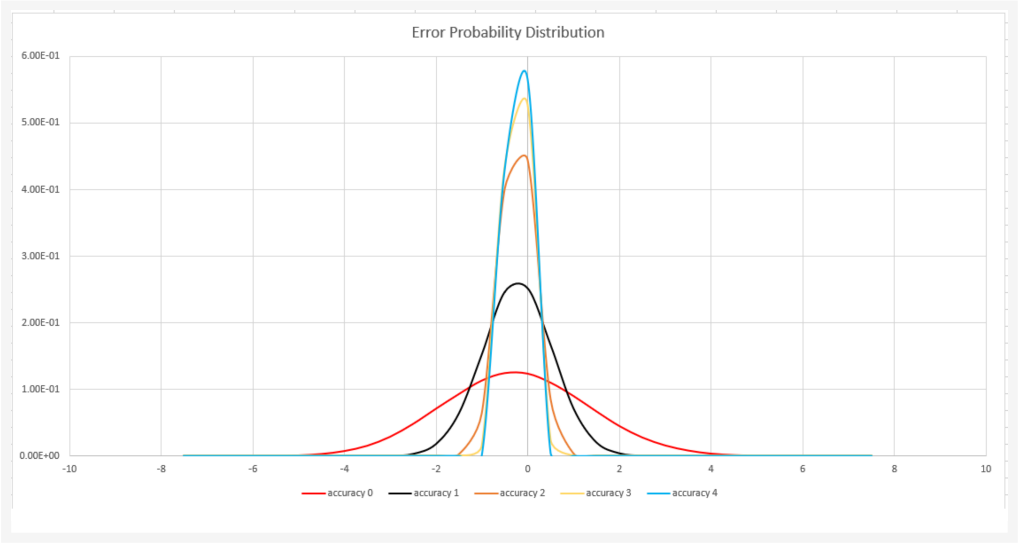
The **Entry count** column is the number of 1-byte counters in the HLL implementation.

The algorithm includes some provisions for doing a perfect count (zero error), if the set cardinality is small enough:

- When the accuracy level is `1` – 1,000 values are returned
- When the accuracy level is `2` – 8,000 values are returned

The error bound is probabilistic, not a theoretical bound. The value is the standard deviation of error distribution (the sigma), and 99.7% of the estimations will have a relative error of under 3 x sigma.

The following image shows the probability distribution function of the relative estimation error, in percentages, for all supported accuracy settings:



Search page

10.5.4.6. DCOUNTIF

The `dcountif` aggregation function calculates an estimate of the number of distinct values of **Expression** of rows for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
dcountif( Expression, Predicate, [, Accuracy] )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Use any scalar expression that returns a `bool` value. Wildcards are not supported for field names.
- **Predicate**: Expression used to filter rows.
- **Accuracy**: Optional. An integer that defines the requested estimation accuracy. If unspecified, the default value is `1`.

Results

Returns an estimate of the number of distinct values of **Expression** of rows for which **Predicate** evaluates to `true` in the group.

`dcountif` could return an error in cases where all or none of the rows pass the **Predicate** expression.

Example

This example summarizes the estimated cardinality of destination ports 80 and 443, by destination address:

```
dataset="cribl_search_sample"
| summarize distinctCountWebPorts=dcountif(dstport, dstport in (80,443)) by dstaddr
```

Copy | Run in Cribl Search

Estimation Accuracy

This function uses a variant of the HyperLogLog (HLL) algorithm, which does a stochastic estimation of set cardinality. The algorithm provides a knob that can be used to balance accuracy and execution time per memory size:

Accuracy	Error (%)	Entry count
0	1.6	212
1	0.8	214
2	0.4	216
3	0.28	217
4	0.2	218

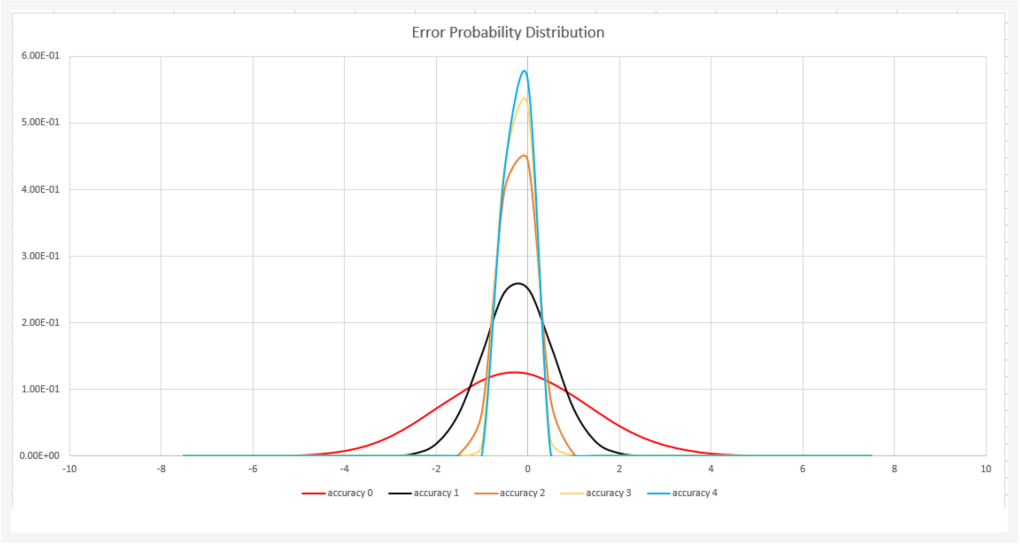
The **Entry count** column is the number of 1-byte counters in the HLL implementation.

The algorithm includes some provisions for doing a perfect count (zero error), if the set cardinality is small enough:

- When the accuracy level is `1` – 1,000 values are returned
- When the accuracy level is `2` – 8,000 values are returned

The error bound is probabilistic, not a theoretical bound. The value is the standard deviation of error distribution (the sigma), and 99.7% of the estimations will have a relative error of under 3 x sigma.

The following image shows the probability distribution function of the relative estimation error, in percentages, for all supported accuracy settings:



Search page

10.5.4.7. Max

The `max` aggregation function finds the maximum value across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
max( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Results

Returns the maximum value of **Expression** across the group.

Numeric values are considered before alpha or alphanumeric values.

Example

This example summarizes the maximum traffic, in bytes, for each source address:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize maxBytes=max(bytes) by srcaddr
```

Copy | Run in Cribl Search

10.5.4.8. MAXIF

The `maxif` aggregation function calculates the maximum value across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
maxif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Results

Returns the maximum value of **Expression** across the group for which **Predicate** evaluates to `true`.

Numeric values are considered before alpha or alphanumeric values.

Example

This example returns the maximum byte count for source address `10.0.0.33`:

```
dataset="cribl_search_sample"
| limit 1000
| summarize maxif(bytes,srcaddr=="10.0.0.33")
```

Copy | Run in Cribl Search

10.5.4.9. Min

The `min` aggregation function finds the minimum value across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
min( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Results

Returns the minimum value of **Expression** across the group.

Numeric values are considered before alpha or alphanumeric values.

Example

This example summarizes the minimum traffic, in bytes, for each source port:

```
dataset="cribl_search_sample"
| limit 1000
| summarize min(bytes) by srcport
```

Copy | Run in Cribl Search

10.5.4.10. MINIF

The `minif` aggregation function calculates the minimum value across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
minif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Results

Returns the minimum value of **Expression** across the group for which **Predicate** evaluates to `true`.

Numeric values are considered before alpha or alphanumeric values.

Examples

This example returns the minimum byte count for source address `10.0.0.33`:

```
dataset="cribl_search_sample" | limit 1000 | summarize minif(bytes,srcaddr=="10.0.0.3
```

Copy Run in Cribl Search

This example summarizes minimum byte counts by private source address:

```
dataset="cribl_search_sample"
| limit 1000
| summarize minif(bytes, ipv4_is_private(srcaddr)) by srcaddr
```

Copy Run in Cribl Search

10.5.4.11. Percentile

The `percentile` aggregation function returns an estimate for the specified nearest-rank percentile of the population defined by **Expression**. The accuracy depends on the density of population in the region of the percentile. The percentiles aggregate provides an approximate value using T-Digest.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
percentile( Expression, Percentile )
```

Arguments

- **Expression**: Expression that will be used for aggregation calculation. Does not support wildcards for field names.
- **Percentile**: A double constant that specifies the percentile.

Results

Returns an estimate for **Expression** of the specified percentiles in the group.

Examples

This example summarizes 95th-percentile traffic, in bytes, by source address, excluding low values:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize perc95=percentile(bytes, 95) by srcaddr
| where perc95 > 100
```

Copy Run in Cribl Search

This example summarizes 95th-percentile traffic, in bytes, by destination address, excluding `null` values:

```
dataset="cribl_search_sample" datatype="aws_vpcflow"
| where isnotnull(bytes) and isnotnull(dstaddr)
| summarize p95=percentile(bytes, 95) by dstaddr
```

Copy Run in Cribl Search

10.5.4.12. Stdev

The `stdev` aggregation function calculates the standard deviation of **Expression** across the group, using Bessel's correction for a small data set that is considered a sample.

For a large data set that is representative of the population, use the stdevp aggregation function.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Used formula:

$$\sqrt{\frac{1}{n-1} \times \sum (X - \frac{\sum (X)}{n})^2}$$

stdev formula

Syntax

```
stdev( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Results

The standard deviation value of **Expression** across the group.

Example

This example summarizes average traffic in bytes, and two measures of standard deviation (sample- and population-based), by source address:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize avg(bytes), stdev(bytes), stdevp(bytes) by srcaddr
```

Copy | Run in Cribl Search

10.5.4.13. STDEVIF

The `stdevif` aggregation function calculates the stdev of **Expression** across the group for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
stdevif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: When `true`, the **Expression** calculated value will be added to the standard deviation.

Results

Returns the standard deviation of **Expression** across the group for which **Predicate** evaluates to `true`.

Example

This example summarizes standard deviations in byte count, by source address, for events whose destination port is higher than `1024`:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize stdevif(bytes, dstport>1024) by srcaddr
```

Copy | Run in Cribl Search

10.5.4.14. STDEVP

The `stdevp` aggregation function calculates the standard deviation of **Expression** across the group, considering the group as a population for a large data set that is representative of the population.

For a small data set that is a sample, use the stdev aggregation function.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Used formula:

$$\sqrt{\frac{1}{N} \times \sum (X - \frac{\sum (X)}{N})^2}$$

stdevp formula

Syntax

```
stdevp( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Results

The standard deviation value of **Expression** across the group.

Example

This example summarizes average traffic in bytes, and two measures of standard deviation (sample- and population-based), by source address:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize avg(bytes), stdev(bytes), stdevp(bytes) by srcaddr
```

Copy Run in Cribl Search

10.5.4.15. Sum

The `sum` aggregation function calculates the sum of **Expression** across the group.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
sum( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Results

Returns the sum value of **Expression** across the group.

Example

This example sums up the total number of packets sent to each combination of destination address and destination port:

```
dataset="cribl_search_sample"
| summarize totalPackets=sum(packets) by dstaddr,dstport
```

Copy | Run in Cribl Search

10.5.4.16. SUMIF

The `sumif` aggregation function calculates the sum of **Expression** for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
sumif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Expression that will be used to filter rows.

Results

Returns the sum of **Expression** across the group for which **Predicate** evaluates to `true`.

Examples

This example sums up the number of packets sent to destination address `10.8.20.30` through each destination port:

```
dataset="cribl_search_sample"
| summarize packetsToBob=sumif(packets, dstaddr=="10.8.20.30") by dstport
```

Copy | Run in Cribl Search

10.5.4.17. Variance

The `variance` aggregation function calculates the variance of **Expression** across the group, considering the group as a sample.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Used formula:

$$\frac{1}{n-1} \times \sum (X - \frac{\sum (X)}{n})^2$$

variance formula

Syntax

```
variance( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Results

The variance value of **Expression** across the group.

Example

This example summarizes the average byte count, and corresponding variance, by source address:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize avg(bytes), variance(bytes) by srcaddr
```

Copy Run in Cribl Search

10.5.4.18. VARIANCEIF

The `varianceif` aggregation function calculates the variance of **Expression** for which **Predicate** evaluates to `true`.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Syntax

```
varianceif( Expression, Predicate )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.
- **Predicate**: Predicate that if `true`, the **Expression** calculated value will be added to the variance.

Results

Returns the variance value of **Expression** across the group for which **Predicate** evaluates to `true`.

Example

This example summarizes the variance in byte count, by source address, for events whose destinatino port is higher than `1024`:

```
dataset="cribl_search_sample" dataSource="vpcflowlogs"
| summarize varianceif(bytes, dstport>1024) by srcaddr
```

Copy Run in Cribl Search

10.5.4.19. VARIANCEP

The `variancep` aggregation function calculates the variance of **Expression** across the group, considering the group as a population.

Use this function with the `summarize`, `eventstats`, and `timestats` operators.

Used formula:

$$\frac{1}{N} \times \sum (X - \frac{\sum (X)}{N})^2$$

variancep formula

Syntax

```
variancep( Expression )
```

Arguments

- **Expression**: Expression used for aggregation calculation. Wildcards are not supported for field names.

Results

The variance value of **Expression** across the group.

Example

```
dataset=myDataset
| summarize variancep(x)
```

Copy

10.5.5. Window Functions

Window functions operate on multiple rows (events) in a row set at a time. Unlike aggregation functions, window functions require that the rows in the row set be ordered or sorted.

Cribl Search supports window functions on queries that use the `extend` operator. With this combination, Cribl Search implicitly applies the `centralize` operator. Cribl Search does not support window functions on queries with the `project` or `where` operator.

The following window functions are available:

Name	Description
`next`	Returns the value of a specific field in a subsequent row.
`prev`	Returns the value of a specific field in a previous row.
`row_cumsum`	Calculates the cumulative sum for a specified field across all previous rows.
`row_number`	Assigns a unique row number to each row within the results.
`row_rank_dense`	Assigns a unique numerical position (rank) to each row within the results
`row_rank_min`	Assigns a minimal numerical position (rank) to each row within the results
`row_window_session`	Identifies the value at the beginning of each session for a specified field within the results.

10.5.5.1. Next

The `next` function returns the value of a specific field in a subsequent row. The subsequent row is located at a specified offset relative to the current row within the results.

Syntax

```
next(Field [, Offset ] [, DefaultValue ] )
```

Arguments

- **Field**: The field from which to get the values.
- **Offset**: The number of rows to move from the current row. Default is `1`.
- **DefaultValue**: The default value when there's no value in the later row. The default is `null`.

Scope

Cribl Search supports this function in the `extend` operator, but not in the `project` or `where` operator.

Examples

This example returns the time difference between adjacent events.

```
dataset="cribl_search_sample" dataSource="access*" host="web01.cribl.io"
 | limit 100
 | sort by _time asc
 | extend time_next_delta = next(_time) - (_time)
```

Copy Run in Cribl Search

This example filters for events with a time difference greater than 250 ms:

```
dataset=myDataset
| where SensorName == 'sensor-9'
| sort by Timestamp asc
| extend timeDiffInMilliseconds = datetime_diff('millisecond', next(Timestamp, 1), Ti
| where timeDiffInMilliseconds > 250
```

Copy

10.5.5.2. Prev

The `prev` function returns the value of a specific field in a previous row. The previous row is located at a specified offset relative to the current row within the results.

Syntax

```
prev(Field [, Offset ] [, DefaultValue ] )
```

Arguments

- **Field**: The field from which to get the values.
- **Offset**: The offset to go back in rows. The default is `1`.
- **DefaultValue**: The default value to be used when there are no previous rows from which to take the value. The default is `null`.

Scope

Cribl Search supports this function in the `extend` operator, but not in the `project` or `where` operator.

Examples

This example returns the time difference between adjacent events.

```
dataset="cribl_search_sample" dataSource="access*" host="web01.cribl.io"
 | limit 100
 | sort by _time asc
 | extend time_prev_delta = (_time) - prev(_time)
```

Copy Run in Cribl Search

This example filters for events with a time difference greater than 250 ms:

```
dataset=myDataset
| where SensorName == 'sensor-9'
| sort by Timestamp asc
| extend timeDiffInMilliseconds = datetime_diff('millisecond', Timestamp, prev(Timest
| where timeDiffInMilliseconds > 250
```

Copy

10.5.5.3. Row_cumsum

The `row_cumsum` function calculates the cumulative sum for a specified field across all previous rows.

Syntax

```
row_cumsum(Term [, Restart ] )
```

Arguments

- **Term**: An expression that indicates the value to be summed. Supports `int`, `long`, or `real`.
- **Restart**: An expression that returns a `bool` value to indicate when the accumulation operation should restart or be set back to 0. It can be used to indicate partitions in the data.

Scope

Cribl Search supports this function in the `extend` operator, but not in the `project` or `where` operator.

Examples

This first example returns total bytes per a specified host.

```
ddataset="cribl_search_sample" dataSource="access*" host="web01.cribl.io"
 | limit 100
 | sort by _time asc
 | extend total_bytes=row_cumsum(bytes)
 | project _time, bytes, total_bytes
```

Copy Run in Cribl Search

This second example returns total bytes per a specified host, restarting the count when the `clientip` is identical in adjacent rows.

```
dataset="cribl_search_sample" dataSource="access*" host="web01.cribl.io"
 | limit 100
 | sort by _time asc
 | extend total_bytes_from_client=row_cumsum(bytes, clientip!=prev(clientip))
 | project _time, clientip, bytes, total_bytes_from_client
```

Copy Run in Cribl Search

10.5.5.4. ROW_NUMBER

The `row_number` function assigns a unique row number to each row within the results. The row index starts by default at 1 for the first row, and is incremented by 1 for each additional row. Optionally, the row index can start at a different value than 1.

Syntax

```
row_number(StartingIndex [, Restart ] )
```

Arguments

- **StartingIndex**: The value of the row index to start at or restart to. The default value is `1`. Supports `long`.
- **Restart**: An expression that returns a `bool` value to indicate when the numbering operation should restart to the **StartingIndex** value. The default is `false`.

Scope

Cribl Search supports this function in the `extend` operator, but not in the `project` or `where` operator.

Example

This example restarts the cumulative counter each time it encounters identical `clientip` values on adjacent rows.

```
dataset="cribl_search_sample" dataSource="access*" host="web01.cribl.io"
 | limit 100
 | sort by _time asc
 | extend rownum=row_number(0, clientip!=prev(clientip))
 | project _time, clientip, bytes, rownum
```

Copy Run in Cribl Search

10.5.5.5. ROW_RANK_DENSE

The `row_rank_dense` function assigns a unique numerical position (rank) to each row within the results, preserving the order even for rows with equal values. Unlike regular ranking, there are no gaps in the assigned ranks.

The row rank starts by default at 1 for the first row, and is incremented by 1 whenever the provided **Term** is different than the previous row's **Term**.

Syntax

```
row_rank_dense( Term [, Restart ] )
```

Arguments

- **Term**: An expression that indicates the value to consider for the rank. The rank is increased whenever the **Term** changes.
- **Restart**: An expression that returns a `bool` value to indicate when the ranking operation should restart to the **Term** value. The default is `false`.

Scope

Cribl Search supports this function in the `extend` operator, but not in the `project` or `where` operator.

Examples

This example uses `row_rank_dense` to return a count of distinct hosts, and also uses `row_rank_min` to rank the hosts.

```
dataset="cribl_search_sample" dataSource="access*"
 | limit 100
 | sort by host asc
 | extend row_rank_dense=row_rank_dense(host) // incremented from 1
 | extend row_rank_min=row_rank_min(host) // actual row number (starting from row 1)
 | project _time, host, row_rank_dense, row_rank_min
```

Copy | Run in Cribl Search

This snippet partitions data by `appname`:

```
| extend Rank=row_rank_dense(appname, prev(appname) != appname)
```

Copy

10.5.5.6. ROW_RANK_MIN

The `row_rank_min` function assigns a minimal numerical position (rank) to each row within the results, considering rows with equal values to have the same rank.

The rank is the minimal row number that the current row's **Term** appears in.

Syntax

```
row_rank_min( Term [, Restart ] )
```

Arguments

- **Term**: An expression that indicates the value to consider for the rank. The rank is the minimal row number for **Term**.
- **Restart**: An expression that returns a `bool` value to indicate when the ranking operation should restart to the **Term** value. The default is `false`.

Scope

Cribl Search supports this function in the `extend` operator, but not in the `project` or `where` operator.

Examples

This example uses `[row_rank_min]` to return a ranking of hosts, and also uses `row_rank_dense` to count distinct hosts.

```
dataset="cribl_search_sample" dataSource="access*"
 | limit 100
 | sort by host asc
 | extend row_rank_dense=row_rank_dense(host) // incremented from 1
 | extend row_rank_min=row_rank_min(host) // actual row number (starting from row 1)
 | project _time, host, row_rank_dense, row_rank_min
```

Copy | Run in Cribl Search

This snippet partitions data by `appname`:

```
| extend Rank=row_rank_min(appname, prev(appname) != appname)
```

Copy

10.5.5.7. ROW_WINDOW_SESSION

The `row_window_session` function identifies the value at the beginning of each session for a specified field within the results.

The function has the following conceptual calculation model:

1. Goes over the input sequence of **Expr** values in order.
2. For every value, determines if it establishes a new session.
3. If it establishes a new session, it emits the value of **Expr**. Otherwise, emits the previous value of **Expr**.

Syntax

```
  row_window_session( Expr , MaxDistanceFromFirst , MaxDistanceBetweenNeighbors [, Restart] )
```

Arguments

- **Expr**: An expression whose values are grouped together in sessions. Null values produce null values, and the next value starts a new session. **Expr** must be a scalar expression of type `datetime`.
- **MaxDistanceFromFirst**: Establishes one criterion for starting a new session: The maximum distance between the current value of **Expr** and the value of **Expr** at the beginning of the session. It's a scalar constant of type `timespan`.
- **MaxDistanceBetweenNeighbors**: Establishes a second criterion for starting a new session: The maximum distance from one value of **Expr** to the next. It's a scalar constant of type `timespan`.
- **Restart**: An expression that returns a `bool` value to indicate when the operation should restart. If specified, every value that evaluates to `true` will immediately restart the session.

Scope

Cribl Search supports this function in the `extend` operator, but not in the `project` or `where` operator.

Example

```
| sort by ID asc, Timestamp asc
| extend SessionStarted = row_window_session(Timestamp, 1h, 5m, ID != prev(ID))
```

Copy

10.6. Virtual Tables

Use virtual tables to dynamically generate system information useful for troubleshooting, performance analysis, and testing.

Available Virtual Tables

Name	Description
`$vt_datasets`	Lists available Datasets. Useful for finding Datasets of a specific type or with specific name patterns.
`$vt_dataset_providers`	Lists available Dataset Providers. Useful for a quick overview of your Dataset Providers.
`$vt_dummy`	Generates dummy data. Useful for testing your queries.
`$vt_jobs`	Lists previously run searches. Useful for troubleshooting or performance analysis.
`$vt_list`	Lists available virtual tables. Useful for looking up the names and descriptions of other virtual tables.
`$vt_lookups`	Lists available lookups, or gets the contents of a specific lookup. Useful for searching through your lookup files.
`$vt_results`	Loads the results of one or more previous searches, so you can reuse them.

10.6.1. $VT_DATASETS

The `$vt_datasets` virtual table lists available Datasets.

```
// returns a list of all Datasets available to the current user
dataset="$vt_datasets"
```

Copy | Run in Cribl Search

Purpose

Use `$vt_datasets` to quickly find Datasets of a specific type or with specific name patterns.

Permissions

Search Member Type	Permissions
Admin	Can see all Datasets in their organization.
Editor	Can see only those Datasets that they created themselves or that were shared with them.
User	Can see only those Datasets that were shared with them.

Syntax

```
dataset="$vt_datasets"
```

Copy | Run in Cribl Search

Returns

Returns one event for each available Dataset. Each event contains the following fields:

Field	Description
`id`	The Dataset ID.

Field	Description
`description`	The Dataset description.
`provider`	The `id`, `description`, and `type` of the Dataset Provider.

Examples

List all your S3 Datasets.

```
dataset="$vt_datasets"
  | where provider.type == "s3"
```

Copy Run in Cribl Search

Count the number of Datasets of each type.

```
dataset="$vt_datasets"
  | summarize count() by provider.type
```

Copy Run in Cribl Search

10.6.2. $VT_DATASET_PROVIDERS

The `$vt_dataset_providers` virtual table lists available Dataset Providers.

```
// returns a list of all Dataset Providers available to the current user
dataset="$vt_dataset_providers"
```

Copy | Run in Cribl Search

Purpose

Use `$vt_dataset_providers` for a quick overview of your Dataset Providers.

Permissions

Search Member Type	Permissions
Admin	Can see all Dataset Providers in their organization.
Editor	Can see only those Dataset Providers that they created themselves or that were shared with them.
User	Can see only those Dataset Providers that were shared with them.

Syntax

```
dataset="$vt_dataset_providers"
```

Copy | Run in Cribl Search

Returns

Returns one event for each available Dataset Provider. Each event contains the following fields:

Field	Description
`id`	The ID of the Dataset Provider.

Field	Description
`description`	The description of the Dataset Provider.
`type`	The type of the Dataset Provider. For example, `cribl_edge`.

Examples

List all your S3 Dataset Providers.

```
dataset="$vt_dataset_providers"
  | where type == "s3"
```

Copy Run in Cribl Search

List all built-in Dataset Providers.

```
dataset="$vt_dataset_providers"
  | where type startswith "cribl"
```

Copy Run in Cribl Search

10.6.3. $VT_DUMMY

The `$vt_dummy` virtual table generates dummy events.

```
// returns 10 dummy events
dataset="$vt_dummy" event<10
```

Copy | Run in Cribl Search

Purpose

Use `$vt_dummy` to test your queries with dummy data.

The `event` predicate lets you specify the number of events, and the `second` predicate lets you simulate a long-running query. Using both lets you simulate a burst of events over a specific time range.

Permissions

All types of Search Members can use the `$vt_dummy` virtual table.

Syntax

```
dataset="$vt_dummy" event<5 second<3
```

Copy

Parameters

Name	Type	Required	Description
`NumberOfEvents`	`int`	No	The number of events to generate. If used together with `SearchRuntime`, generates the `NumberOfEvents` per second, over the course of the `SearchRuntime`. If not specified, generates one event.
`SearchRuntime`	`int`	No	The desired runtime of the simulated search, in seconds.

Returns

Returns dummy events according to the specified criteria. Each event contains the following fields:

Field	Description
`_time`	The event's timestamp.
`event`	The event's ordinal number.

Examples

Generate one dummy event, with an additional field `foo` set to `42`.

```
dataset="$vt_dummy"
  | extend foo=42
```

Copy Run in Cribl Search

Generate five dummy events.

```
dataset="$vt_dummy" event<5
```

Copy Run in Cribl Search

Generate one event per second, over the course of 10 seconds.

```
dataset="$vt_dummy" second<10
```

Copy Run in Cribl Search

Generate three events per second, over the course of 10 seconds.

```
dataset="$vt_dummy" event<3 and second<10
```

Copy Run in Cribl Search

10.6.4. $Vt_jobs

The `$vt_jobs` virtual table lists previously executed searches.

```
// returns a list of all searches that were run within the past hour and are availabl
dataset="$vt_jobs" earliest=-1h
```

Copy | Run in Cribl Search

Purpose

Use `$vt_jobs` for troubleshooting or performance analysis, or to monitor search activity in your organization.

Permissions

Search Member Type	Permissions
Admin	Can see all searches in their organization.
Editor or User	Can see only those searches that they ran themselves or that were shared with them.

Syntax

```
dataset="$vt_jobs"
```

Copy

Returns

Returns one event for every search run within the specified time range, filtered through any additional criteria you specify.

Each event contains fields with the search's details.

Examples

Aggregate searches from the past seven days by their status.

```
dataset="$vt_jobs" earliest=-7d
  | summarize count() by status
```

Copy Run in Cribl Search

10.6.5. $VT_LIST

The `$vt_list` virtual table lists available virtual tables.

```
// returns a list of all virtual tables available to the current user
dataset="$vt_list"
```

Copy | Run in Cribl Search

Purpose

Use `$vt_list` to quickly look up the names and descriptions of virtual tables available in Cribl Search.

Permissions

All types of Search Members can see the list of all virtual tables.

Syntax

```
dataset="$vt_list"
```

Copy

Returns

Returns one event for each of Cribl Search's virtual tables, containing the table's name and description.

Examples

List virtual tables whose names start with `$vt_object`.

```
dataset="$vt_list"
  | where name startswith "$vt_object"
```

Copy | Run in Cribl Search

10.6.6. $VT_LOOKUPS

The `$vt_lookups` virtual table lists available lookups, or gets the contents of a specific lookup.

```
// returns the contents of the service_names_port_numbers.csv lookup file
dataset="$vt_lookups" lookupFile="service_names_port_numbers"
```

Copy | Run in Cribl Search

Purpose

Use `$vt_lookups` to:

- Get a quick overview of all lookup files available to you.
- Search the contents of a specific lookup.

Permissions

All types of Search Members can see the list of all lookup files in the organization. For details about who can export to `$vt_lookups`, see Search Member Permissions.

Syntax

```
dataset="$vt_lookups"
  // optionally, get the contents of a specific lookup file:
  lookupFile="LookupName"
```

Copy

Parameters

Name	Type	Description
`LookupName`	`string`	The name of the lookup file whose content you want to access. Don't include the file extension.

Returns

Returns one event for each available lookup. Each event contains the following fields:

Field	Description
`name`	The name of the lookup.
`description`	The description of the lookup.
`rowCount`	The number of rows in the lookup.
`size`	The size of the lookup file, in bytes.
`compressed`	`true`: the lookup file is compressed. `false`: the lookup file is not compressed.

Examples

List all lookups available to the current user.

```
dataset="$vt_lookups"
```

Copy Run in Cribl Search

List available lookups that have more than 10,000 rows.

```
dataset="$vt_lookups"
  | where rowCount>10000
```

Copy Run in Cribl Search

Get only specific entries of the `service_names_port_numbers.csv` lookup file.

```
dataset="$vt_lookups" lookupFile="service_names_port_numbers"
 | where transport == "udp"
```

Copy Run in Cribl Search

10.6.7. $VT_RESULTS

The `$vt_results` virtual table retrieves the result sets of searches completed (or attempted) in the past.

```
// get the results of the search 1704236905683.wgocax, if available to the current us
dataset="$vt_results" jobId="1704236905683.wgocax"
```

Copy | Run in Cribl Search

Purpose

Use `$vt_results` to access the results of previously executed searches without having to rerun them. This can be particularly useful for iterative analysis or when you want to reference the results of multiple searches in a single query.

Mind that although `$vt_results` doesn't rerun the actual query, it **searches** the results of the specified searches, which can lead to minor credit consumption. This is different than opening a search from **History**, which only loads the cached results.

Permissions and Limits

Search Member Type	Permissions
Admin	Can access the results of all searches in the organization.
Editor or User	Can access the results of only those searches that they ran themselves or that were shared with them.

You can access previous search results for as long as they're kept in the system. By default it's 7 days, but you can change this in Settings > **Search** > **Limits > **Search history TTL**.

`$vt_results` cannot access the results of searches that are still running.

Variations

By default, `$vt_results` can access only the results of successfully completed searches. However, you precede this Dataset with the `set` statement's `allow_incomplete_results` option to also access the results of failed or canceled searches. This option is `false` by default. Set it to `true` using syntax like this:

```
set allow_incomplete_results=true;
dataset="$vt_results" jobId=<some-ID>
```

Copy

Syntax

```
dataset="$vt_results"
  // either:
  jobId="SearchId"
  // or:
  jobName="SavedSearchName" [execution > -NumberOfPreviousRuns]
```

Copy

Parameters

Name	Type	Description
`SearchId`	`string`	The ID of the search whose results you want to access. You can find the ID of every search in its **Details**, or in **History**. For multiple IDs, use square brackets and commas: `jobId=["SearchId1", "SearchId2"]`.
`SavedSearchName`	`string`	The **name** (not ID) of the saved search whose results you want to access. By default, you'll get the results of the latest execution that saved search, unless you use the the `execution` parameter. For multiple names, use square brackets and commas, for example: `jobName=["SavedSearchName1", "SavedSearchName2"]`.

`NumberOfPreviousRuns`	`int`
	
	or
	
	`*`
		Which of the past executions of the saved search to load. If not specified, only the latest run is loaded.
		
		`execution = 0` (default) loads the last run.
		`execution = -1` loads the run before last.
		`execution > -1` loads the last two runs together.
		`execution > -2` loads the last three runs together.
		`execution = *` loads all runs kept in history.
		
		Works only with the `SavedSearchName` specified.

Returns

Returns the results of the specified searches, in their original format.

Examples

Get the Results of Multiple Previous Searches

Get the results of all completed searches whose IDs are between `1704236905600` and `1704236906000`.

```
dataset="$vt_results"
  | where jobId>"1704236905600" and jobId<"1704236906000"
```

Copy | Run in Cribl Search

Get the Results of a Previous Run of a Saved Search

Get the results of the last run of a saved search named `mySavedSearch`.

```
dataset="$vt_results" jobName="mySavedSearch" execution = 0

// or simply:

dataset="$vt_results" jobName="mySavedSearch"
```

Copy | Run in Cribl Search

Get the run before last:

```
dataset="$vt_results" jobName="mySavedSearch" execution = -1
```

Copy | Run in Cribl Search

Get the Results of Multiple Previous Runs of a Saved Search

Get the results of the last two executions of a saved search named `mySavedSearch`.

```
dataset="$vt_results" jobName="mySavedSearch" execution > -1
```

Copy Run in Cribl Search

Get the last three runs:

```
dataset="$vt_results" jobName="mySavedSearch" execution > -2
```

Copy Run in Cribl Search

10.7. Types

Supported types of data in Cribl Search.

Every data value (such as the value of an expression, or the parameter to a function) has a **type**.

The following table lists the types supported by Cribl Search, alongside additional aliases you can use to refer to them.

Type	Aliases
`bool`	`boolean`
`datetime`	`date`
`double`	`real`, `decimal`
`dynamic`	
`int`	`long`
`string`	
`timespan`	`time`

All types include the special `null` value, which represents the lack of data or a mismatch of data. For example, attempting to ingest the string `"abc"` into an `int` column results in `null`.

Unsupported types

The following table shows the types unsupported by Cribl Search:

Type	Alias
`float`	
`int16`	
`uint16`	
`uint32`	`uint`
`uint64`	`ulong`

Type	Alias
uint8	byte

10.7.1. Bool

The `bool` data type can have one of two states: `true` or `false`, as well as the `null` value.

Alias: `boolean`.

`bool` Definition

A value is considered `true` in any of the following cases:

- If it's the actual `bool` value `true`.
- If it's one of the following case-insensitive strings: `'true'`, `'t'`, `'yes'`, `'y'`.
- If it's a numeric value equal to `1` (including `1.0`).

A value is considered `false` in any of the following cases:

- If it's the actual `bool` value `false`.
- If it's one of the following case-insensitive strings: `'false'`, `'f'`, `'no'`, `'n'`.
- If it's a numeric value equal to `0` (including `0.0`).

Any values that couldn't be converted to `bool` are considered `null`.

Comparing `bool` to Numbers

When comparing `bool` values to numbers, only `1` is considered `true`. For example:

Expression	Result
`true == 1`	`true`
`true == 100`	`false`

However, when converting a value to a `bool`, all non-zero numbers are converted to `true`:

Expression	Result
`tobool(100)`	`true`
`bool(-1)`	`true`
`iff(100, 1, 0)`	`1`

Expression	Result
`not(100)`	`false`

`bool` as a Filter

To search for any values that are considered `true`, filter on the literal `true` value:

- `dataset="foo" something=true`
- `dataset="foo" | where something==true`

These will match on:

- Booleans with value `true`.
- The following case-insensitive strings: `'true'`, `'t'`, `'yes'`, `'y'`.
- Numeric values equal to `1`.

To search for any values that are considered `false`, filter on the literal `false` value:

- `dataset="foo" something=false`
- `dataset="foo" | where something==false`

These will match on:

- Booleans with value `false`.
- The following case-insensitive strings: `'false'`, `'f'`, `'no'`, `'n'`.
- Numeric values equal to `0`.

Note that searching on `"true"` or `"false"` literals with quotes does a string match instead. This means that:

```
dataset="foo" | where something=="true"
```

Will match on `"true"` and `true`, but not on `1`, `"yes"` or any other `bool` values.

`bool` as Function Input

For any functions that take `bool` arguments, the values will be automatically converted to `true`, `false` or `null`, according to our `bool` definition. Mind that **all non-zero numbers are converted to `true`**.

For conditional functions, `null` is treated as `false`.

Expression	Result
`iif(true, 'true', 'false')`	`'true'`
`iif('yes', 'true', 'false')`	`'true'`
`iif(1, 'true', 'false')`	`'true'`
`iif(123, 'true', 'false')`	`'true'`
`iif('false', 'true', 'false')`	`'false'`
`iif(0, 'true', 'false')`	`'false'`
`iif('abc', 'true', 'false')`	`'false'`

`bool` and the Logical NOT Operator

Both the logical NOT `!` operator and the `not` function:

- Return `false` for all values considered `true` by the `bool` definition. Mind that **all non-zero numbers are converted to `true`**.
- Return `true` for all values considered `false` by the `bool` definition.
- Return `null` if passed `null`, or for any values that can't be converted to `true` or `false`.

Expression	Result
`not(true)`	`false`
`not('true')`	`false`
`not('yes')`	`false`
`not(1)`	`false`
`not(100)`	`false`
`not(false)`	`true`
`not('false')`	`true`
`not('no')`	`true`
`not(0)`	`true`

Expression	Result
`not(null)`	`null`
`not("hello")`	`null`

10.7.2. Datetime

The `datetime (date)` data type represents an instant in time. A `datetime` value can be any of the following:

1. A complete timestamp, expressed in Unix epoch time. Examples:
    - `1700511360` (a whole number, representing seconds)
    - `1700511420.123` (a fractional/decimal number, representing milliseconds)
2. Time only, assuming "today" UTC as the date. Examples:
    - `12:34`
    - `12:34:56`
    - `12:34:56.789`
3. Date only, assuming midnight UTC as the date line. Example:
    - `2025-01-23`
4. A combination of date and time. Examples:
    - `2025-01-23 12:34:56.789` (supports nonstandard format with space)
    - `2025-01-23T12:34:56.789Z` (supports ISO "Z" for time zone)
    - `2025-01-23T12:34:56.789+00:00` (supports ISO variation with offset)

10.7.3. Decimal

The `decimal` data type represents a 64-bit wide, double-precision, floating-point number.

Aliases: `double`, `real`.

- `decimal("123.4")` returns `123.4`
- `decimal("1e3")` returns `1000`
- `decimal("hello")` returns `null`

10.7.4. Double

The `double` data type represents a 64-bit wide, double-precision, floating-point number.

Aliases: `decimal`, `real`.

- `double("123.4")` returns `123.4`
- `double("1e3")` returns `1000`
- `double("hello")` returns `null`

10.7.5. Dynamic

The `dynamic` scalar data type is special in that it can take on any value of other scalar data types from the list below, as well as arrays and property bags.

Specifically, a `dynamic` value can be any of the following:

- A value of any of the primitive scalar data types: `bool`, `datetime`, `int`, `long`, `real`, `string`, and `timespan`.
- A **dynamic array** (an array of `dynamic` values), holding zero or more values with zero-based indexing.
- A **property bag** (or **dictionary**) that maps keys (unique `string` values) to `dynamic` values. A property bag has zero or more such mappings (called **slots**), indexed by the keys. Slots are unordered.
  Cribl Search doesn't attempt to preserve the order of key-to-value mappings in a property bag, and so you can't assume the order to be preserved. It's entirely possible for two property bags with the same set of mappings to yield different results when they are represented as `string` values.
- `null`.

`dynamic` Literals

A literal of type `dynamic` looks like this:

`dynamic(` *Value* `)`

Value can be:

- `null`, in which case the literal represents the null dynamic value: `dynamic(null)`.
- Another scalar data type literal, in which case the literal represents the `dynamic` literal of the "inner" type. For example, `dynamic(4)` is a dynamic value holding the value 4 of the long scalar data type.
- An array of dynamic or other literals: `[` *ListOfValues* `]`. For example, `dynamic([1, 2, "hello"])` is a dynamic array of three elements, two `long` values and one `string` value.
- A property bag: `{` *Key* `=` *Value* `...` `}`. For example, `dynamic({"a":1, "b":{"a":2}})` is a property bag with two slots, `a`, and `b`, with the second slot being another property bag.

`dynamic` Object Accessors

To subscript a dictionary (in other words, reference a key in a property bag), use either dot notation (`dict.key`) or bracket notation (`dict["key"]`). When the subscript is a string constant, both options are equivalent.

To use an expression as the subscript, use bracket notation. When using arithmetic expressions, the expression inside the brackets must be wrapped in parentheses.

In the examples below `dict` and `arr` are columns of dynamic type:

Expression	Accessor expression type	Meaning	Comments
`dict[col]`	Entity name (column)	Subscripts a dictionary using the values of the column `col` as the key	Column must be of type string
`arr[index]`	Entity index (column)	Subscripts an array using the values of the column `index` as the index	Column must be of type integer or boolean
`arr[-index]`	Entity index (column)	Retrieves the 'index'-th value from the end of the array	Column must be of type integer or boolean
`arr[(-1)]`	Entity index	Retrieves the last value in the array	
`arr[toint(indexAsString)]`	Function call	Casts the values of column `indexAsString` to int and use them to subscript an array	
`dict[['where']]`	Keyword used as entity name (column)	Subscripts a dictionary using the values of column `where` as the key	Entity names that are identical to some query language keywords must be quoted
`dict.['where']` or `dict['where']`	Constant	Subscripts a dictionary using `where` string as the key	

Casting `dynamic` Objects

After subscripting a dynamic object, you must cast the value to a simple type.

Expression	Value	Type
X	`parse_json('[100,101,102]')`	array
`X[0]`	`parse_json('100')`	dynamic

Expression	Value	Type
`toint(X[1])`	`101`	int
`Y`	`parse_json('{"a1":100, "a b c":"2015-01-01"}')`	dictionary
`Y.a1`	`parse_json('100')`	dynamic
`Y["a b c"]`	`parse_json("2015-01-01")`	dynamic
`todate(Y["a b c"])`	`datetime(2015-01-01)`	datetime

Cast functions are:

- tolong
- todouble
- todatetime
- totimespan
- tostring
- parse_json

Functions Over `dynamic` Types

Dynamic scalar functions allow you to manipulate objects by operating on `dynamic` values, including dynamic arrays and property bags.

Name	Description
`bag_has_key`	Checks whether a property bag contains a given key.
`bag_keys`	Lists all root keys of a property bag.
`bag_merge`	Merges multiple property bags, discarding duplicate keys.
`bag_pack`	Creates a property bag from an alternating list of keys and values.
`bag_pack_columns`	Creates a property bag from a list of columns.
`bag_remove_keys`	Removes key-value pairs from a property bag.
`bag_set_key`	Adds or overwrites a key-value pair in a property bag.
`bag_zip`	Creates a property bag from two dynamic arrays.

Name	Description
make_bag	Creates a property bag from multiple input bags.
make_bag_if	Creates a property bag from those input bags that meet the specified condition.
zip	Merges dynamic arrays, grouping elements by index.

The following string functions support `dynamic` types as well:

Function	Usage with dynamic data types
extractjson(`path,object`)	Uses path to navigate into object.
parse_json(`source`)	Turns a JSON string into a dynamic object.
range(`from,to,step`)	Generates an array of values.

10.7.6. Int

The `int` data type represents a signed, 64-bit wide, integer.

Aliases: `long`.

`int` Examples

- `int("123.456")` returns `123`
- `int("abc")` returns `null`

10.7.7. Long

The `long` data type represents a signed, 64-bit wide, integer.

Aliases: `int`.

`long` Examples

- `long("123.456")` returns `123`
- `long("abc")` returns `null`

10.7.8. Null

The `null` value in Cribl Search can indicate:

- The absence of a field or value.
- A type mismatch in a conversion, operation, or function. For example, `toint('abc')` and `123 / 'hello'` both evaluate to `null` because strings can't be converted to numbers.

`null` Literals

A `null` literal is represented in Cribl Search as `null`. For example:

- `where field==null`
- `extend result=iff(value > 100, value, null)`

`null` Equality and Inequality

Cribl Search supports the `null` value in equality/inequality comparisons. These comparisons are equivalent to using the `isnull` and `isnotnull` functions.



This is different from standard Kusto Query Language (KQL), which returns `null` if both sides of the comparison are `null`.

See the following examples:

Expression	Result
`5 == null`	`false`
`5 != null`	`true`
`null == 5`	`false`
`'abc' == null`	`false`
`'null' == null`	`false`
`0 == null`	`false`
`int('abc') == null`	`true`

Expression	Result
`int('abc') == int('def')`	`true`

`null` and Strings

Cribl Search supports the `null` value in `string` comparisons.



This is different from standard Kusto Query Langauge (KQL), which does not allow the `null` value for strings.

See the following examples:

Expression	Result
`tostring(null)`	`null`
`string(null)`	`null`
`isnull('abc')`	`false`
`'abc' == null`	`false`

`null` and Binary Operators

Binary operators are those that perform an operation on two values. These operators include `and`, `or`, `==`, `!=`, `>=`, `<=`, numeric operators (`+`, `-`, `*`), and others.

If one or both of the values passed to the binary operator are `null`, then the output of the binary operator is also `null`. For example, `null + 10` returns `null`.

Exceptions to the rule above are:

- For the equality `==` and inequality `!=` operators, either or both sides of the operation can be `null`, and will return `true` or `false` rather than `null` (as described in `null` Equality and Inequality).
- For the logical `AND` operator, if one of the values is `false`, the result is also `false`.
- For the logical `OR` operator, if one of the values is `true`, the result is also `true`.

See the following examples:

Expression	Result
`5 > 3`	`true`
`null > 3`	`null`
`null > int('abc')`	`null`
`null == int('abc')`	`true`
`null >= int('abc')`	`true`
`null >= 3`	`null`
`null > int('abc')`	`null`
`10 + null`	`null`
`10 + 'abc'`	`null`
`10 - ''`	`null`
`10 - null`	`null`
`10 + unknown_field`	`null`
`10 - toint('abc')`	`null`
`null * 10`	`null`
`null > 3 and true`	`null`
`null > 3 and false`	`false`
`null > 3 or true`	`true`
`null > 3 or false`	`null`

`null` and the `where` Operator

The `where` operator treats `null` as `false`. This means that events with conditions evaluating to `null` won't appear in the search results.

For example, `dataset="foo" | where tobool("abc")` returns no results because `tobool("abc")` evaluates to `null`.

`null` and the Logical `NOT` Operator

If the value passed to the logical `NOT` (`!`) operator is `null`, then the output is also `null`.

Expression	Result
`not(true)`	`false`
`not(false)`	`true`
`not(null)`	`null`
`not(int('abc'))`	`null`
`!int('abc')`	`null`

10.7.9. Real

The `real` data type represents a 64-bit wide, double-precision, floating-point number.

Aliases: `decimal`, `double`.

- `real("123.4")` returns `123.4`
- `real("1e3")` returns `1000`
- `real("hello")` returns `null`

10.7.10. String

The `string` data type represents a sequence of zero or more Unicode characters.

`string` Literals

There are several ways to encode literals of the `string` data type in a query text:

- Enclose the string in double-quotes `"`:
  `"This is a string literal. Single quote characters (') don't require escaping. Double quote characters (\") are escaped by a backslash (\\)."`
- Enclose the string in single-quotes `'`:
  `'Another string literal. Single quote characters (\') require escaping by a backslash (\\). Double quote characters (") do not require escaping.'`

In the two representations above, the backslash `\` character indicates escaping. The backslash is used to escape the enclosing quote characters, tab characters `\t`, newline characters `\n`, and itself `\\`.

The newline character `\n` and the return character `\r` can't be included as part of the string literal without being quoted.

Verbatim `string` Literals

Verbatim string literals are also supported. You specify them by prepending an asperand `@` character. In this form, the backslash character `\` stands for itself, and not as an escape character.

- Enclose in double-quotes `"`:
  `@"This is a verbatim string literal that ends with a backslash\. Double quote characters ("") are escaped by a double quote ("")."`
- Enclose in single-quotes `'`:
  `@'This is a verbatim string literal that ends with a backslash\. Single quote characters ('') are escaped by a single quote ('')."`

Concatenation

Joining two or more string literals is supported. For example:

```
dataset="$vt_dummy"
 | extend x = strcat('Hello', ', ', "world!")
```

Copy Run in Cribl Search

Returns `x` as `Hello, world!`.

Obfuscation

Obfuscating strings is not supported, however, the query will not complain if attempted.

`string` Examples

- `string(123)` returns `"123"`
- `string("hello")` returns `"hello"`
- `string(null)` returns `null`

10.7.11. TIMESPAN

The `timespan(time)` data type represents a time interval, in seconds.



This representation differs from the standard Kusto `timespan` type, which represents a literal timespan.

`timespan` Literals

Literals of type `timespan` have the syntax `timespan(value)`, where a number of formats are supported for `value`, as indicated by the following table:

Value	Length of time
`2d`	2 days
`1.5h`	1.5 hour
`30m`	30 minutes
`10s`	10 seconds
`0.1s`	0.1 second
`100ms`	100 millisecond
`10microsecond`	10 microseconds
`1tick`	100ns
`time(15 seconds)`	15 seconds
`time(2)`	2 days
`time(0.12:34:56.7)`	`0d+12h+34m+56.7s`

The special form `time(null)` is the null value.

`timespan` Operators

Two values of type `timespan` may be added, subtracted, and divided. The last operation returns a value of type `real` representing the fractional number of times one value can fit the other.

Examples

The following examples calculate, in several ways, how many seconds are in a day:

- `1d / 1s`
- `time(1d) / time(1s)`
- `24 * 60 * time(00:01:00) / time(1s)`

10.8. Regex Matching

Regular expressions usage, with examples, supported flags, and syntax details.

The `matches regex`, `search`, `where`, `in`, `!in`, `in~`, and `!in~` operators can use regex syntax to match against strings and regex literals.

To match against regex literals, the supported comparison operators are `==`, `=~`, `!=`, and `!~`.

Regex Examples

The examples below focus on using regular expressions with the `where` operator.

Match on String Values

Match a corresponding regex literal, with wildcard. This is case-sensitive by default:

```
dataset=myDataset
| summarize event_count=count() by State
| where State == /K.*S/
```

Copy

Case-insensitive version of the above query, using the `/i` flag:

```
dataset=myDataset
| summarize event_count=count() by State
| where State == /k.*s/i
```

Copy

Case-insensitive, using the `=~` operator instead of the `/i` flag:

```
dataset=myDataset
| summarize event_count=count() by State
| where State =~ /k.*s/
```

Copy

Match on `in` Lists or Combined Targets

Mix and match string targets with regex literal targets:

```
dataset="cribl_internal_logs" method=*
| limit 1000
| where method in ("GET", /^po.*/i)
```

Copy Run in Cribl Search

Same query, using the `in~` operator for implicit `/i`:

```
dataset="cribl_internal_logs" method=*
| limit 1000
| where method in~ ("GET", /^po.*/)
```

Copy Run in Cribl Search

An "or" query with | notation:

```
dataset="cribl_internal_logs" method=*
| limit 1000
| where method == /post|get/i
```

Copy Run in Cribl Search

A negation query:

```
dataset="cribl_internal_logs" method=*
| limit 1000
| where method != /post|get/i
```

Copy Run in Cribl Search

Find the string value `bar` in the field `foo`:

```
cribl foo = /bar/
```

Copy

Regex Flags

Evaluation is case-sensitive by default. You can ignore case by using the `/i` flag or `=~` operator, as shown in the examples below.

This operator supports any combination of the `/i`, `/m`, `/s`, and `/u` flags:

Flag	Description
`i`	Case-insensitive match.
`m`	Multi-line mode: Allows `^` and `$` to match next to newline characters.
`s`	Single-line ("dotail") mode: Allows `.` to match newline characters.
`u`	Unicode (full) mode; Allows matching against 4-byte Unicode characters.

Disambiguate Regex Characters

In some expressions, you might need to distinguish regex literal delimiters from division symbols. Consider this expression:

```
dataset=myDataset
| <query>
| where value == 10/20/5
```

Copy

The `/20/` portion **could** be a regular expression. But given the numeric characters surrounding it, Cribl Search interprets as division. However, a similar expression with non-numeric characters is ambiguous:

```
dataset=myDataset
| <query>
| where value == x/y/z
```

Copy

Here, Cribl Search would try to interpret `/y/` as a regular expression. You can disambiguate the expression as a series of divisions by inserting blank spaces, conceptually like this:

```
where value == x / y / z
```

Copy

But to insert the blank spaces within regex syntax, you'd use the `\s` character qualifier:

```
dataset=myDataset
| <query>
| where value == /x\s\/\sy\s\/\sz/
```

Copy



In the above style of regular expression, `/` must be escaped as `\/`.