

CRIBL **AMAZING**

**DATA
STORIES**



**YOUR GUIDE
TO THE
DATA ENGINE OF TOMORROW**



Copyright © 2024 Cribl, Inc. | cribl.io

All rights reserved. No portion of this book may be reproduced in any form without written permission from Cribl, Inc., except in the case of copying for personal, non-commercial use or as otherwise permitted by U.S. copyright law. For permission requests, please email legal@cribl.io.

This book is designed to provide accurate and authoritative information for the use of Cribl, Inc.'s products. While best efforts have been used in preparing this book, Cribl, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaims any liability in connection with the use or result from the use of the information in this book. The information contained in this book might not be suitable for your situation. The actual use of Cribl, Inc.'s products must be in accordance with the applicable software license agreements and relevant documentation. Please visit docs.cribl.io for official documentation of Cribl, Inc.'s features and functionality.

The Cribl trademarks and brand names displayed in this book, including, without limitation, Cribl®, Cribl.Cloud®, Cribl Stream™, Cribl Edge™, Cribl Search™, Cribl Lake™, CriblCon™, Data Engine for IT and Security™, AppScope®, and the Cribl logo displayed above, are the property of Cribl, Inc. You may not use or display any trademarks or service marks owned by Cribl, Inc. without our prior written consent. All other product names, logos, brands, trademarks, registered trademarks, and other marks listed in this document are the property of their respective owners. All such marks are provided for identification and informational purposes only. The use of these marks does not indicate affiliation, endorsement, or ownership of the marks or their respective owners.

Cribl Amazing Data Stories

Your Guide to the Data Engine
of Tomorrow

Contents

Foreword	01
Preface	05
Acknowledgments/Editor's Note	07
Operating Manual for Spaceship Cribl	09
Optimize Data	15
1. Optimize Data Using Routing.	17
2. Optimize Data Using Transformation	31
3. Mask and Redact Sensitive Data	53
4. Replay Data from Object Storage with Cribl Stream	65
Simplify Tooling	81
5. Tool Comparison and Migration at Blazing Speed	83
6. Kubernetes Monitoring Simplified with Cribl Edge	95
Go Deeper with Cribl Search, Lake, and AI	105
7. Faster, Better, Cheaper: Logs to Metrics with Cribl Search, Stream, and Lake	107
8. Summarize Events over Time with Cribl Search	117
9. Refine Retrieved Data with Cribl Search	125
10. Cribl, AI, and You	137

Foreword

by Clint Sharp, Dritan Bitincka, and Ledion Bitincka, Cribl

At Cribl, we stand for choice. Choice for what to do with your data and what tools best fit your needs, today and in the future. After all, your data is your data. That's why the Cribl product suite is vendor-agnostic by design and purpose-built for you, the IT and Security professionals who keep the business running.

Your needs are unique and ever-changing. And what got you through the past 10 years won't get you through the next 10 years. Cribl is here to help you transform your data strategy and, ultimately, we want data to work for you, freeing up your time to focus on the many mission-critical tasks at hand.

From the outset, we've taken a first principles approach to innovating for you. We seek to understand your biggest problems, then we build the right solutions and continuously iterate to ensure our products work for you. As the founders, we have firsthand experience with many of the challenges facing IT and Security professionals today, but we don't pretend to know everything. We're insatiably curious and partner closely with you, our customers and users, as we continue to build our products.

Before we dive in further, we want to give you a brief overview of our product suite. We started back in 2018 with Cribl Stream, the industry's leading observability pipeline that can collect data from almost any source, process billions of events per second, and transform and route data to optimize your analytics, storage, and retention.

Since then, we've broadened the scope of the company to meet your needs across data's entire lifecycle. We launched Cribl Edge to move intelligence closer to the data source and simplify managing fleets of agents, enabling you to deal with thousands more agents than you ever dreamed possible.

Next came Cribl Search, the industry's first search-in-place solution. Search gives you the insights you need, while minimizing the cost and complexity of ingesting and shipping data before analysis.

Most recently we launched Cribl Lake, a turnkey data lake solution that you can set up and configure in a few clicks, offering automated optimization of storage and open formats for easy access and retrieval without being a data expert.

That's where we are today on this journey, together with all of you, as we empower you to transform your data strategy. Now, you might be asking yourself: So what's this book all about? This is our guide for you, full of tips and techniques to get the most value out of your data, based on insights from the people who build and support our products and solutions for customers. The stories in this book illustrate a variety of approaches to getting your data to the right places, in the most optimized formats, and in the most efficient ways.

Whether you skim the chapters to see what's possible, or directly apply these techniques using the resources we provide, we hope this book will help you discover new ways to unlock the full value of IT and Security data.

We also hope you have fun reading it. We know this is serious work, but we can still have fun doing it.

So we invite you to dive right in and join us on this ongoing journey. It's not science fiction. It's not a mystery. It's Cribl, the Data Engine for IT and Security.





Preface

by Michael Katz, Cribl

In 2018, a virtual rocket launch vaulted humanity into a new orbit and a new phase of existence: The Criblarity. As soon as Cribl released its first product, LogStream 1.0, observability and security engineers gained a new level of choice and control over all their tooling. They also found new ways to optimize routing and analysis to keep up with massive data growth – enabling them to scale their resources, and to retain and analyze more of what matters.

Throughout our years of listening carefully to users, and building new technology to meet their needs, Cribl has continued to innovate. We now offer an expanded, integrated product portfolio. We’ve achieved remarkable acceptance among our home planet’s largest enterprises, those that process and analyze the highest volumes of data.

Just as importantly, customers really like our software and our distinctly human support, as well as what Cribl can do for them. That’s what they tell us all the time.

Boldly going, Cribl now offers you a warp-drive Data Engine for IT and Security. Or if you prefer, a formidable Dyson Sphere to superpower your data management to a whole new level.

In these pages, we share techniques for optimizing your data throughput, storage, analysis, compliance, security posture, and incident investigation. These are the voyages of Cribl’s galactic explorers – elite practitioners who’ve joined our federation, in many cases after starting out as

enthusiastic users of our products. Here, they present the starlogs of some of the solutions they've built for Cribl users. Read on and leap into a future so technologically advanced, your colleagues might find it indistinguishable from magic.

Cribl's continuing mission: to explore strange, new datatypes; to seek out new efficiencies and new capabilities; to help you go where no data practitioner has gone before!

There are two sides to every amazing story, and you'll notice that there are two sides to this book. This side starts with a brief overview of Cribl's products and resources, and then presents techniques relevant to both IT and Security practitioners. The flip side focuses on techniques and stories of particular (but not exclusive) interest to Security sleuths. We hope you'll read both!

Welcome aboard. The countdown has already finished, and your spaceflight is departing now.

—

(For the most up-to-date version of this book's contents, see cribl.io/amazing.)



Acknowledgments/ Editor's Note

by Michael Katz, Cribl

Grigori Melnik had the vision for this book and set it in motion. Abby Strong, Joel Vincent, Erin Sweeney, and Meredith Torres kept it moving. Christopher Gales shaped it from an amorphous space blob into something flightworthy, adding a delicate balance of creativity and realism. Eric Wolfe and Mara Sahleanu provided graphics and illustration magic, developing some imagery in collaboration with ideogram.ai and Canva.

Mara and Alyssa Houk at Cribl, and Tamara Kreiss and Lesley Harrison at Bay Area Graphics, provided the overall design, layout, and production talent that gave this bookness. Douglas Howatt deftly copy-edited the whole manuscript and kept it real. Maliha Balala, Emil Mikhailov, Ed Bailey, Jackie McGuire, Igor Gifrin, and Josh Biggley provided crucial connections, content, encouragement, and more illustrations. Jennifer Marandola graciously allowed me a break from developing ongoing Cribl technical guidance to assemble this, working from the blueprint of a von Neumann probe. Marissa Dahlson, Diane Chiu, and Matt Lanza helped arrange my space boarding pass.

The greatest thanks go to all the authors for their ingenuity in writing these recipes, and for their patience in reworking text and graphics with me. And to Clint Sharp, Dritan Bitincka, and Ledion Bitincka for launching the generational starship that is Cribl. Any errors, misstatements, or anachronisms are my bad.



Operating Manual for Spaceship Cribl

Access to Tools

by Michael Katz, Cribl

This book is designed to be read in different ways. If you're just evaluating Cribl for the first time, you might approach it as a book of armchair adventures. If you're an experienced user of Cribl products, you can use (and apply) it as a book of practical recipes for solving real-world data-management problems.

But perhaps you're somewhere in the middle. Are you a new Cribl user who's eager to learn more and do more with our offerings? In that case, here's your exclusive boarding opportunity to blast off and ride our spaceship to the stars.

THE CRIBL SPACE FLEET

At this writing, Cribl offers four distinct products that you can readily combine, as needed, to facilitate your workflow. They'll take your data wherever it needs to go across the universe. They can even place it in cryogenic suspension to survive the rigors of interstellar travel or long-term retention for compliance.

Cribl Stream, our flagship product, offers the view from the starship's bridge. It's your perch to observe, transform, optimize, and route data in motion.

Cribl Edge, a complementary product that shares much of Stream's user interface, invites you to teleport to the Engine Room. With Edge, you can collect data at its source, and you can configure logical Fleets to centrally manage large numbers of similar agents that keep your data flowing.

And Edge can forward data to Stream for more-intensive processing, with no additional ingest charges. (We know you'll want to save some credits to spend when you drop in at the nearest spaceport cantina.)

Cribl Search is your von Neumann space probe to examine distant data at rest, before deciding whether to bring it onboard your ship. Search is the perfect analytical counterpart to data that you store in commodity object storage like Cribl Lake. Search enables you to query and analyze that data while minimizing the ingest or storage charges you'd incur on downstream analytics platforms.

Cribl Lake is your expandable wormhole in the cosmos, where you can store masses of data in hidden dimensions and can curve spacetime. Move your data analysis forward at warp speed. And replay full-fidelity artifacts from the distant past, if you need to analyze their technosignatures.

(Some descriptions in this section are fanciful and not literally scientific, but you get the idea.)

LAUNCHING YOUR CRIBL STARSHIP

If you're encountering Cribl for the first time, we've got all the resources you'll need to grab the controls and steer our products through the recipes and examples presented throughout the book.

FLIGHT SIMULATOR

Cribl Sandboxes (sandbox.cribl.io) are free online tutorials that guide you through using fully functional deployments of Cribl software to process real data. Each Sandbox is a dedicated cloud environment, with upstream and/or downstream services already running for you, and sample data ready to work with. Spend zero time setting up infrastructure, and spend as long as you like working through each of our hosted scenarios.

YOUR TICKET TO THE STARS

When you're ready to take the next step, kick off the training wheels and start processing your own data by signing up for a free Cribl.Cloud account at cribl.io/blastoff. You'll get a free, functioning version of each of our apps (with some capacity restrictions). Here again, spend zero time setting up infrastructure – we'll have everything up and running for you. Newer Sandboxes will automatically create a persistent, free Cribl.Cloud account for you.

When you need more capacity, you can readily upgrade to a paid account, managing flexible credits that you can freely use across all our products. With a paid account, your Cribl.Cloud account can also manage Cribl Stream Groups that you choose to run on your own infrastructure. For more about our Cribl.Cloud deployment model, see cribl.io/cribl-cloud and docs.cribl.io/stream/deploy-cloud.

THE ACTUAL MANUAL

As you plot your course through the infinite dataspace, visit our online technical guidance at docs.cribl.io. You'll find comprehensive, free documentation on all of our products. Each product's Introduction section will get you right into the pilot's seat to fly that product. In particular, the Cribl Stream Introduction section (docs.cribl.io/stream) offers simple to

progressively more advanced tutorials on deploying and using Cribl Stream on infrastructure that you choose.

STARSHIP ACADEMY

Cribl University offers certification courses, and concise non-certification courses, on how to architect and solve real-world problems using Cribl's product suite. Create your free account at cribl.io/university, and you can get started on learning immediately. Cribl training is always free.

GALACTIC ALLIANCE

Once you're in flight, guidance is available from Cribl's in-app Copilot AI chatbot, and from the resources described above. When you need the human touch, we have free peer-to-peer resources to help you. Sign up for Cribl Community Slack at cribl-community.slack.com, and/or sign up for our Cribl Curious threaded forums at community.cribl.io.

In both venues, you'll get help from fellow users, and also from Cribl's stellar Support and Solutions Engineering crewmembers. We also offer monthly (virtual) user group meetings, biweekly office hours, and webinars that provide live demonstrations. With paid plans, you can reserve dedicated, direct support from Cribl staff.

FREE FUEL

Cribl Packs are bundles of preconfigured processing infrastructure and sample data that you can snap into Cribl Stream or Cribl Edge just like a Pipeline. Packs help you quickly accelerate your Cribl Data Engine to warp speed, by freeing you from the need to configure Pipelines or Routes from scratch.

Like other Cribl resources, Packs are completely free. You can directly install them from within Cribl Stream or Cribl Edge. Or you can browse and download Packs of interest from the Cribl Packs Dispensary (packs.cribl.io) and then install them from files.

Several of this book's chapters are written around Cribl-authored Packs designed for specific applications. Cribl will likely correct and enhance these Packs between revisions of this book, to offer even more capabilities.

INTERSTELLAR TRAVEL IS A GROUP EFFORT

If you need help anywhere along your Cribl journey to the stars, please contact us at one of the resources listed above under "Galactic Alliance," or at cribl.io. We're here to help you get the most out of Cribl products, as quickly as possible.





SECTION 01

Optimize Data

01 | Optimize Data Using Routing

Split and Optimize Data Streams for
Different Teams and Destinations

by Joseph Eustaquio, Cribl

PROBLEM

Different teams in your organization might need to access the same incoming data, but to access and handle it in different forms. For example, IT Ops might want to analyze aggregated data. Or SecOps might want to analyze sampled individual events, and share only redacted data with their IT counterparts.

From there, each team might want to format the data for different downstream tools. And both teams might care about preserving a full-fidelity copy of everything. Satisfying all these diverse needs is where data proliferation and tool sprawl happens.

SOLUTION

Cribl Stream is built to function as a universal connector in your data management ecosystem. It efficiently routes data from almost any source to multiple destinations and formats. UI-based configuration makes it easy to optimize the data format, and to apply any required transformations, for each receiver and end use.

WHY CONSIDER THIS APPROACH?

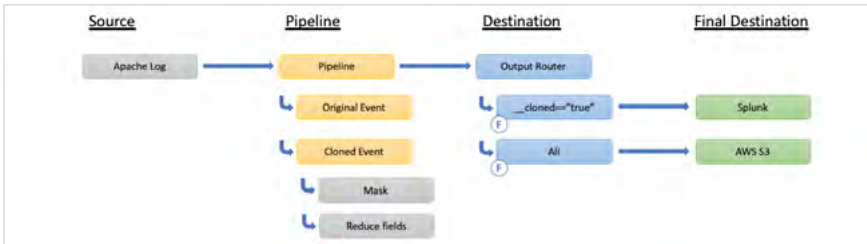
In this example, we'll look at splitting data from Apache logs to a Splunk destination (with some redaction) and to an Amazon S3 bucket (in full fidelity, for retention and compliance). But this is a simplified illustration of much wider possibilities.

Cribl apps can ingest data from other traditional log formats (like syslog or Windows Event Logs), from metrics protocols (like SNMP or Graphite), and from APIs exposed by custom data sources. Output options include sending:

- Security-related data to your SIEM(s).
- Operational data to analytics platforms (such as Elasticsearch or Grafana).
- Historical data to object storage (Cribl Lake, S3-compatible stores, Azure Blob Storage, or Google Cloud Storage).
- Specific data formats or protocols to legacy systems that require them.

SCENARIO

Assume that our middleware team wants us to ingest their Apache logs, and send them out to Splunk. But en route, they want to optimize events and to mask some sensitive data. Also, in parallel, the team wants us to archive the original logs to an S3 bucket for long-term archive retention.



Schematic view of data routing to build in Cribl Stream

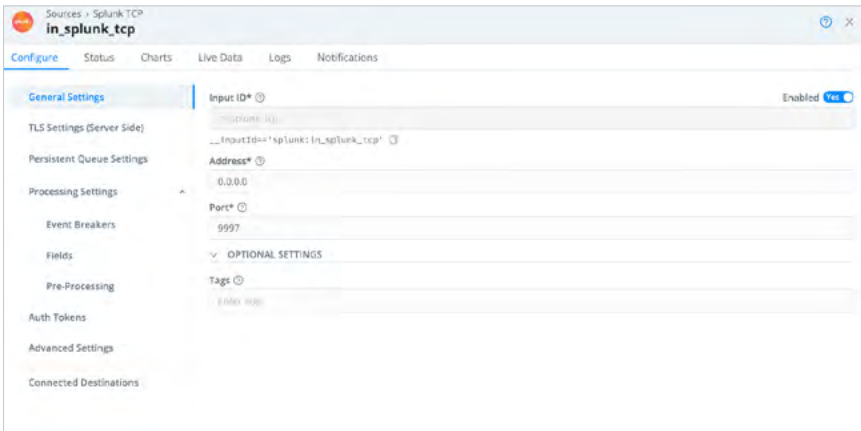
WHAT WE'LL DO

To bifurcate our data to meet these requirements, we'll draw on the following options in Cribl Stream:

- A TCP-based Source to ingest our Apache logs.
- A Clone Function to duplicate the log data streams.
- A Mask Function to redact a sensitive field's values.
- An Eval Function to tune the outbound data for our downstream analytics tool.
- Destination configurations to send the modified data to analytics, and send the original full-fidelity data to object storage.

SET UP THE SOURCE

To import the Apache log data, we'll rely on a Cribl Stream Splunk TCP Source that's already set up for our multiple Splunk Universal Forwarders (UFs). Apache logs come in from the UFs running on middleware hosts. All we need to know from this Source is the corresponding sourcetype to use as a filter for the routing that we'll set up in the upcoming steps.



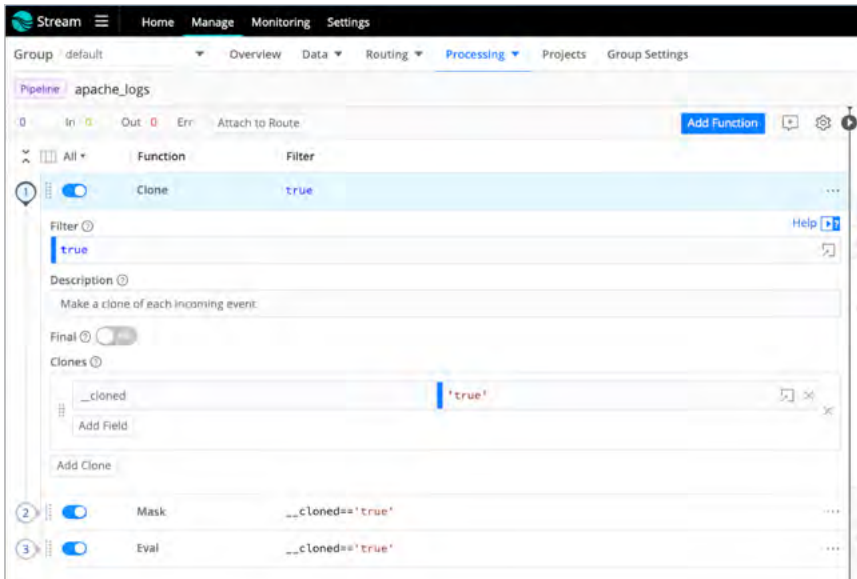
Source configuration for Apache logs

SET UP THE PIPELINE

Next, we'll set up a Cribl Stream Pipeline and add three of Stream's built-in Functions to process our data.

CLONE FUNCTION

The Clone Function, with its default broad Filter expression (`true`), duplicates all the incoming events. We're adding a new internal field named `__cloned` to the resulting cloned records, setting its value also to `true`. This additional field will enable us to later distinguish the cloned events from their originals, and to route them differently.

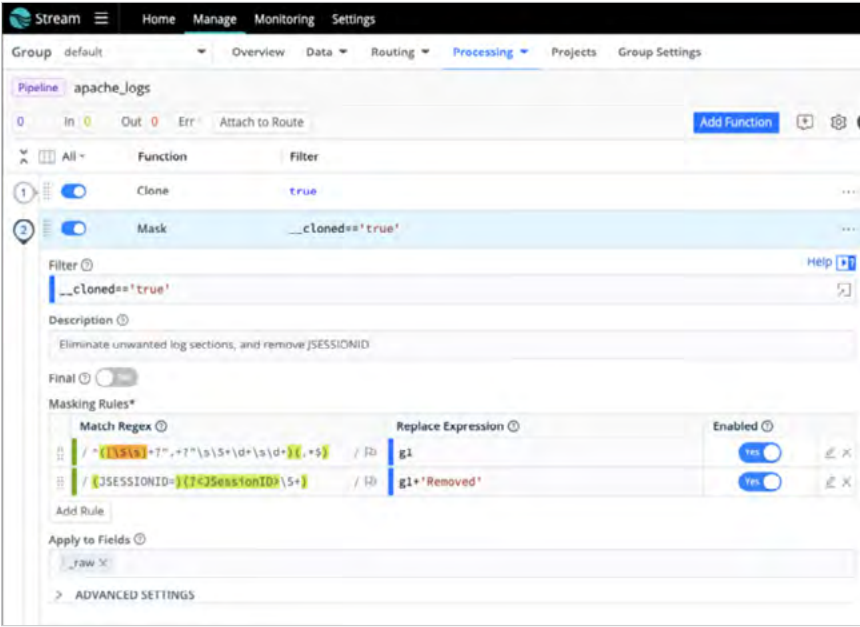


Clone Function to duplicate events

MASK FUNCTION

From the events that go into downstream analytics, our Security team has asked to eliminate the `JSESSIONID` values, because we consider them to be sensitive data. We can also remove trailing browser information and other metadata that IT Ops analysts won't need.

A Mask Function takes care of these redactions. Here, we've set a narrow Filter expression to apply these transformations only to the cloned events.

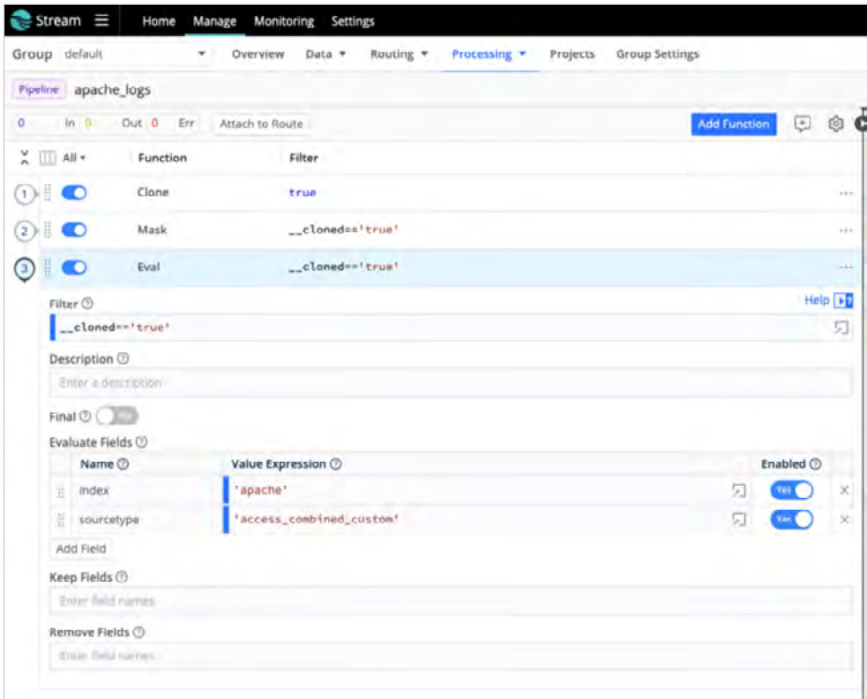


Mask Function to anonymize events

The benefit here isn't just in sanitizing the data – we're also optimizing it. Removing the extraneous fields saved an impressive 44% of licensing costs on the downstream service. It also reduces the storage required to write all these events to disk on that platform. These savings enable us to stretch our budget to analyze more of the data that really matters.

EVAL FUNCTION

Next, we add an Eval Function to customize fields on the cloned events. (Notice the same restrictive Filter expression.) On those events only, we're setting the `index` value, and adjusting the `sourcetype` value to land in the right place in the downstream analytics service.

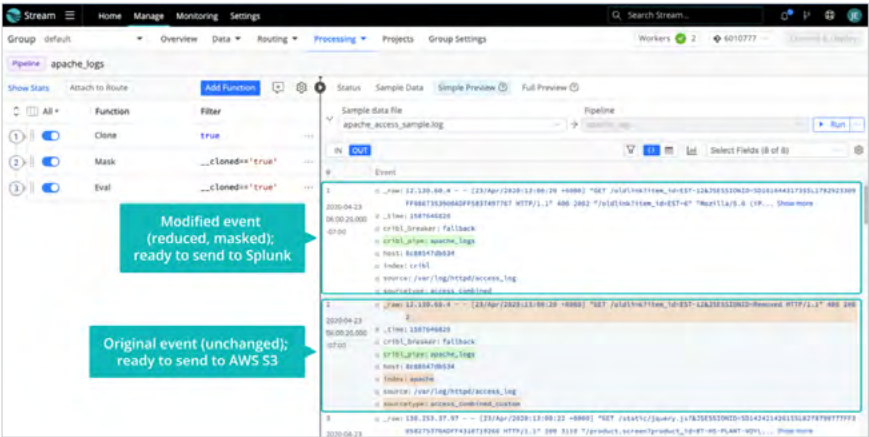


Eval Function to set index and sourcetype fields

PIPELINE OUTPUT

In Stream's left pane, we can see our assembled Pipeline of three Functions stacked together. We've named this Pipeline literally by its purpose – namely, processing `apache_logs`.

In the right Preview pane, the **OUT** tab shows us a sample of both kinds of events. In the cloned events bound for analytics, the color-coding shows us the redacted `_raw` field and the altered `index` and `sourcetype` values. In the original events bound for archival storage, Stream has simply added a `cribl_pipe` field to identify the Pipeline that processed them.



Both streams of events exit the same Pipeline – distinguished by `index` and `sourcetype`

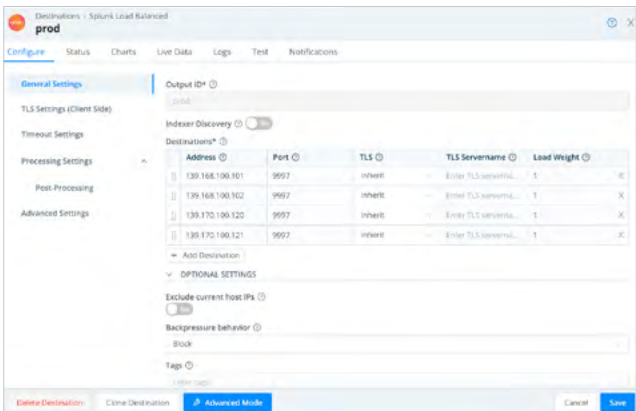
SET UP THE DESTINATIONS

Now that we have the Source configured to ingest the data, and Pipeline Functions configured to transform it, we'll next configure Stream Destinations to export the data to where we want it to go.

We want three Destinations: one for Splunk analytics, a second for Amazon S3 object storage, and an Output Router that will allow us to filter events between those two.

SPLUNK LOAD BALANCED DESTINATION

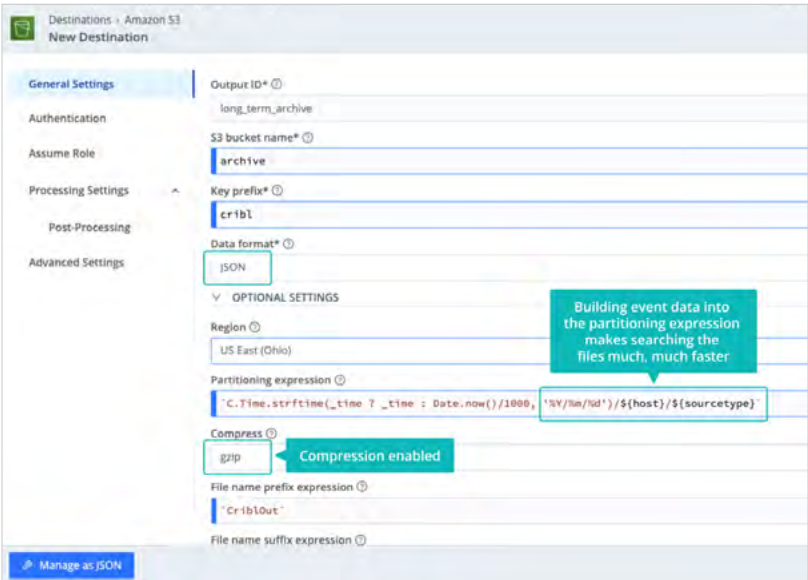
For Splunk analytics, we choose a Splunk Load Balanced Destination, configured to balance among four IP addresses.



Splunk Load Balanced Destination config

AMAZON S3 DESTINATION

To write to object storage, Stream’s Amazon S3 Destination enables us to select JSON as the output format, and to enable compression (because full-fidelity doesn’t need to mean full-size).



Amazon S3 Destination config

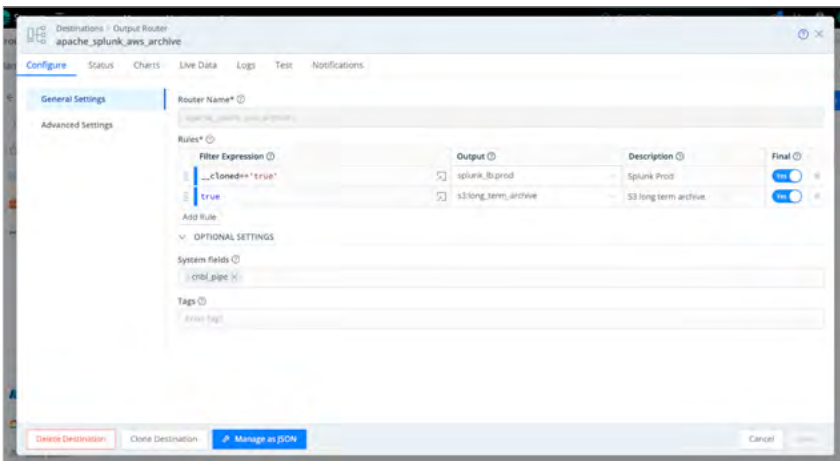
This S3 Destination also defines a partitioning expression that will make the S3 files easier to search in Cribl Search or in replay through Cribl Stream.

OUTPUT ROUTER DESTINATION

Output Router is a hybrid Destination that enables us to branch data to two or more final Destinations. We'll send *both streams* of data from our Pipeline to this custom Destination, where we'll use familiar-looking Filter expressions to handle the branching.

The order of these expressions matters, because Stream evaluates them from the top down. So we'll start with the most-specific filtering. Here, we divert the transformed events (marked with our `__cloned` internal field), and send these *only* to our Splunk Load Balanced (Splunk Prod) Destination. We've accepted the default **Final** toggle here, to ensure that matching events aren't evaluated against the next Filter expression.

Second, we take all remaining events (with the broad `true` value that we've seen before), and we output those full-fidelity events to our configured Amazon S3:long_term_archive Destination.



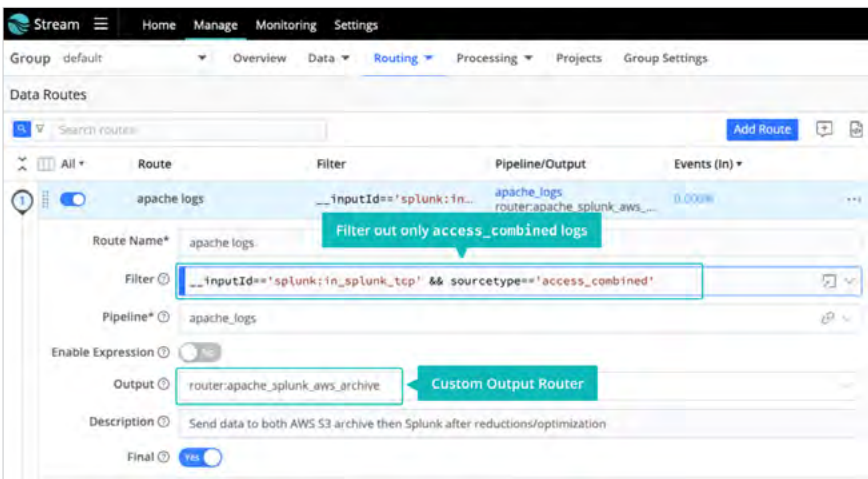
Output Router config, with Filter expressions matching each final output

SET UP THE ROUTE

Now that we've configured input, processing, and filtered outputs, we need to configure a Stream Route to tie everything together. We've given this Route the same name we earlier gave its corresponding Pipeline, `apache_logs`.

In the Route config below, we ensure that the Route will send only the incoming Apache log events – that is, events that match our Splunk TCP Source's Input ID – through this Route's Pipeline.

As the Route's **Output** (Destination), we specify that the events will get bifurcated by our configured Output Router.



Route config to tie everything together

This Route has yet another **Final** toggle enabled, to ensure that matching events will go only through this Route's Pipeline and Destination. They won't be evaluated by any other rules further down the Routing table.

WHAT WE'VE ACCOMPLISHED

We've demonstrated how data engineering teams can use Cribl Stream to take a single source of any data and split it up across multiple final destinations. Stream can give different outbound streams different formats and content to serve each team's needs.

Built-in redaction options keep sensitive data secure, while also optimizing license usage on downstream services. As a result, you can analyze more of the data that matters, while cost-effectively storing a full-fidelity copy for future investigation using tools such as Cribl Search.

VARIATIONS

You can readily extend the techniques shown here – such as field transformations and Output Router branching – to support additional data sources, destinations, teams, and other options. Some scenarios to consider include:

- Send the same data sources to multiple places, in the same format.
NOC (IT Ops) and SOC (SecOps) teams might need the same data in the same format, but in different tools.
- Send full or redacted data to multiple places, in the same format, but including only selective events.
NOC and SOC teams might need the data in the same format, but neither team needs all the events for their analysis. You can use Cribl Stream's built-in Functions to aggregate log events into metrics for the NOC team, while dropping or sampling uninteresting events for the SOC team.

- Send different versions of data to multiple places (different contents, formats, and so on).

The Stream tools and techniques demonstrated in our simple example readily extend beyond bifurcation, to support branching the data in many different ways.

WHAT'S NEXT?

For details on the options we discussed here, see Cribl's online documentation:

- Stream Source configurations (docs.cribl.io/stream/sources).
- Output Router (docs.cribl.io/stream/destinations-output-router).
- Clone Function (docs.cribl.io/stream/clone-function).
- Mask Function (docs.cribl.io/stream/mask-function).
- Eval Function (docs.cribl.io/stream/eval-function).
- Aggregations Function (docs.cribl.io/stream/aggregations-function).
- Drop Function (docs.cribl.io/stream/drop-function).
- Sampling Function (docs.cribl.io/stream/sampling-function).
- Cribl Lake (docs.cribl.io/lake).



02 | Optimize Data Using Transformation

Aggregating and Sampling Log Events

by Dritan Bitincka and George Merhej, Cribl

PROBLEM

Data engineers are grappling with constantly increasing data volumes, in both the IT and security domains. Data grows faster than budgets – hindering analytics performance, obscuring critical insights, and sometimes overwhelming pricey storage.

Traditional approaches often involve sending raw data directly to analytics tools. This sets up an unwelcome choice: Either use up valuable license quota for data that might or might not be relevant now (but might be “someday”); or analyze too little of the data, decreasing the accuracy and quality of your analysis.

SOLUTION

Cribl products work together to optimize data – and for data in motion, Cribl Stream offers flexible, easily configured options to transform and reduce data before feeding it to your chosen analytics platforms.

These options include:

- Aggregating events.
- Sampling events.
- Formatting and cleaning whitespaces.
- Removing nonessential fields.
- Filtering data out, based on granular criteria.

Optimizing your data in any of these ways focuses your analytics budget on what matters. This chapter focuses on Stream's aggregation and sampling options. Because Cribl Stream can multicast data to multiple destinations, you can retain the full-fidelity, unprocessed data in much cheaper object storage, to support later iterative investigations.

WHY CONSIDER THIS APPROACH?

Aggregation enables you to group and summarize data based on specific fields. This significantly reduces data volume, by sending a single record representing multiple original events. For example, if you're analyzing network traffic by source IP address and destination port, Cribl Stream can aggregate the total bytes transferred for each unique combination of IP and port.

Sampling enables you to forward only a defined fraction of similar events for analysis. This is ideal for high-volume, low-value data streams, where it's not critical to examine every event. For example, firewall logs contain

lots of successful connections (status code 200), which aren't essential for ongoing monitoring. You can sample only a fraction of these events, while forwarding all potential errors (non-200 status codes).

HANDS-ON RESOURCES

Cribl Packs offer prebuilt processing infrastructure. They're free resources that you can directly import into Cribl Stream (and Cribl Edge) from the Cribl Packs Dispensary at packs.cribl.io.

Several Packs include Cribl Pipelines tailored to specific data sources and destinations. The Redis Knowledge Pack provides already-configured stateful aggregation options.

SCENARIOS

For the aggregations examples below, we've used Palo Alto Networks Traffic data. However, many other great candidates for aggregations include Palo Alto Networks Firewalls, Web application firewalls, Web proxies, IIS data, and VPC Flow Logs. A sample of the PAN Traffic data, in its CSV format, looks like this:

```

_raw: Feb  4 16:00:01 1,2014/02/04 16:00:01,0009C101998,TRAFFIC,start,1,2014/02/04 16:00:01,10.160.44.107,10.160.20.2,0.0.0.0,0.0.0.0,Systex to DN
5,,,dns,sys1,Systex_DNZ,Trust,acl.613,acl.902,LoggingToPanorama,2014/02/04 16:00:01,1520856,1,42349,53,0,0,0x4000,udp,allow,93,93,0,1,2014/0
2/04 16:00:02,0,any,0,4974797769,0x0,10.0.0.0-10.255.255.255,10.0.0.0-10.255.255.255,0,1,0 Show less
time: 1675512001

```

PAN Traffic Sample

To demonstrate event sampling, we'll look at a combination of HTTP access_combined and VPC Flow Logs.

WHAT WE'LL DO

We'll demonstrate:

- Aggregating log events: First aggregate by fields, then rebuild and rewrite `_raw`.
- Sampling events: First sample by source type, then filter on multiple extracted fields.
- Dynamic sampling for even greater efficiency.

AGGREGATING LOG EVENTS

Let's look at three aggregation scenarios, from simple to complex, based on the same sample data. Each scenario is designed to output a different format. All three reformat the data using Cribl Stream's built-in Aggregations Function.

These examples are designed to impose minimal impacts on your downstream analytics system's existing rules and searches, while optimizing data composition. In these examples, we're sending data to a SIEM (security information and event management) system, but the same principles apply to aggregating events for IT Ops analytics.

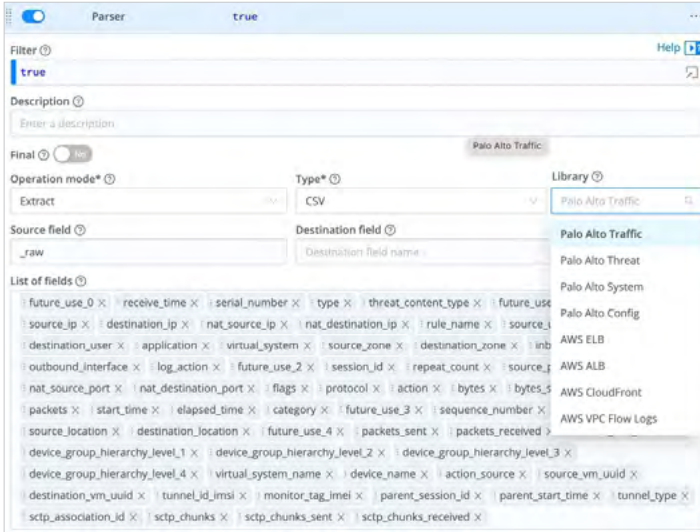
SCENARIO 1: AGGREGATE BY INGEST-TIME FIELDS

We want to group aggregates by the following five fields found in our PAN source data. (Depending on your own data, you might need to add more fields to group accurately:)

- `source_ip`
- `destination_ip`
- `destination_port`
- `action`
- `protocol`

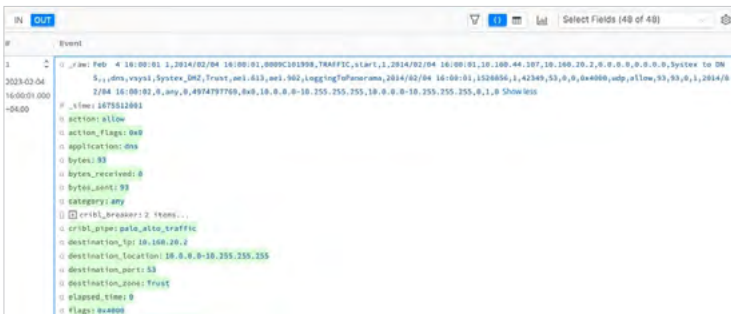
PARSE THE FIELDS OF INTEREST

We first need to parse the data to extract these fields. As shown below, Cribl Stream's Parser Function (docs.cribl.io/stream/parser-function), combined with Stream's embedded Parser Library for Palo Alto Traffic events, predefines the PAN fields to extract.



Parser Function

Previewing the Parser Function's output, we can see the fields broken out.



Parsed Fields

SPECIFYING HOW TO AGGREGATE

Next, use Stream's Aggregations (docs.cribl.io/stream/aggregations-function) Function to aggregate events by the fields we've parsed. Although we're combining events, we'll still pass the number of original log events to our destination. This is very powerful in SIEM use cases that rely on the number of events, such as brute-force attacks.

Below, in the **Aggregates** list we've defined, you can see the three byte-count fields we're summing up, and the count of all original events. You can also see the five **Group by** fields that we chose upfront. In both places where you see `true`, this is Stream's most general filter expression, meaning "all events passing through here."

The screenshot shows the configuration for the 'Aggregations' function in Stream. At the top, there's a toggle for 'Aggregations' which is turned on, and a 'true' value next to it. Below this is a 'Filter' section with a dropdown set to 'true' and a 'Help' link. The 'Description' section has a text input field with the placeholder 'Enter a description'. The 'Final' section has a toggle set to 'No'. The 'Time window*' section has a dropdown set to '10s'. The 'Aggregates*' section contains a list of four aggregation functions: 'sum(bytes).as(totalbytes)', 'sum(bytes_received).as(totalbytes_received)', 'sum(bytes_sent).as(totalbytes_sent)', and 'count(true).as(numberofaggregatedevents)'. Each function has a small 'x' icon to its right. Below the list is an 'Add Aggregate' button. The 'Group by fields' section shows a list of five fields: 'source_ip', 'destination_ip', 'destination_port', 'action', and 'protocol', each with a small 'x' icon to its right.

Aggregations Function setup

When we preview the Aggregations Function's output, it will look something like this:

IN OUT	
#	Event
579	# _time: 1675512000
2023-02-04	▫ action: allow
16:00:00.000	▫ cribl_host: ip-172-31-8-110.us-west-2.compute.internal
+04:00	▫ cribl_pipe: palo_alto_traffic
	▫ destination_ip: 199.204.218.7
	▫ destination_port: 8000
	# endtime: 1675512010
	# numberofaggregatedevents: 8
	▫ protocol: tcp
	▫ source_ip: 10.14.1.98
	# starttime: 1675512000
	# totalbytes: 27350
	# totalbytes_received: 17324
	# totalbytes_sent: 10026

Aggregations Output

SCENARIO 2: RESTORE `_raw` TO INGEST-TIME FIELDS

You might have noticed what's missing after we ran those aggregations – we've lost the original `_raw` field. Depending on your use case, you might need to maintain the `_raw` field and pass it to your destination. We can do this by adding two details to our Aggregations Function:

- Add the `_raw` field of the first aggregated event into a temporary message field. We're naming this field `__temporary_message`, leading with two underscores to mark it as a Cribl internal field.

- Below that, we use the Aggregates Function's **Evaluate Fields** option to add this buffer field's contents into the *aggregated* output's `_raw` field.

Aggregates* ?

sum(bytes).as(totalbytes)

sum(bytes_received).as(totalbytes_received)

sum(bytes_sent).as(totalbytes_sent)

count(true).as(numberofaggregatedevents)

first(_raw).as(__temporary_message)

Add Aggregate

Group by fields ?

source_ip × destination_ip × destination_port × action × protocol ×

Evaluate Fields ?

Name ?	Value expression ?
_raw	`\$__temporary_message`

Maintain_raw

Our aggregated output format now looks almost like one of the original log events:

```

179      @_raw Feb 4 16:00:01,2014/02/04 16:00:01,0000C101908,TRAFFIC,end,1,2014/02/04 16:00:01,10.14.1.98,199.204.218.7,204.107.14
2023-02-04      1.240,199.204.218.7,RFC1918 to Internet,,web-browsing,vsys,Trust,Untrust,acl.902,acl.1000,LoggingToPanorama,2014/02/0
16:00:00.000      4 16:00:01,1562407,1,59576,0000,20230,0000,0x000000,tcp,allow,4616,1380,3236,13,2014/02/04 15:59:29,30,private-ip-adre
+04:00      sses,0,4974797733,0x0,10.0.0.0-10.255.255.255,United States,0,7,6 Show less

      @_time: 1675512000
      @ action: allow
      @ cribl_host: ip=172-31-8-110-us-west-2.compute.internal
      @ cribl_pipe: palo_alto_traffic
      @ destination_ip: 199.204.218.7
      @ destination_port: 8000
      @ endtime: 1675512010
      @ numberofaggregatedevents: 8
      @ protocol: tcp
      @ source_ip: 10.14.1.98
      @ starttime: 1675512000
      @ totalbytes: 27350
      @ totalbytes_sent: 17324
      @ totalbytes_received: 18026

```

Output with `_raw` field restored

If desired, you could then clean up fields you don't need by extending your Pipeline with Stream's built-in Eval Function. We'll see an Eval example in our final aggregation scenario, just below.

SCENARIO 3: UPDATE THE `_raw` FIELD

Let's look at some final optimization of our aggregated output. The `_raw` field is now restored, but it contains the values of the first event added to the aggregations. So the values of `bytes`, `bytes_received`, and `bytes_sent` are the original values of the first aggregated event, which are logically not equal to the `totalbytes`, `totalbytes_received`, and `totalbytes_sent`.

You could just update your downstream service's rules, reports, and dashboards to use these new fields. But more likely, you'd want to limit the changes needed downstream. Luckily, you can instead use Stream Functions to update the `_raw` field with aggregated numbers, requiring no downstream changes.

To do this, we'll parse the aggregated events' `_raw` field, then run an Eval Function to replace the `bytes` with the `totalbytes`, and then finally serialize the resulting data. The Eval Function (docs.cribl.io/stream/eval-function) is configured like this:

3 ☒ Parser true

4 ☒ Eval true

Filter ☒ true

Description

Final ☒ No

Evaluate Fields

Name	Value Expression	Enabled
bytes	totalbytes	<input checked="" type="checkbox"/> Yes
bytes_received	totalbytes_received	<input checked="" type="checkbox"/> Yes
bytes_sent	totalbytes_sent	<input checked="" type="checkbox"/> Yes

Add Field

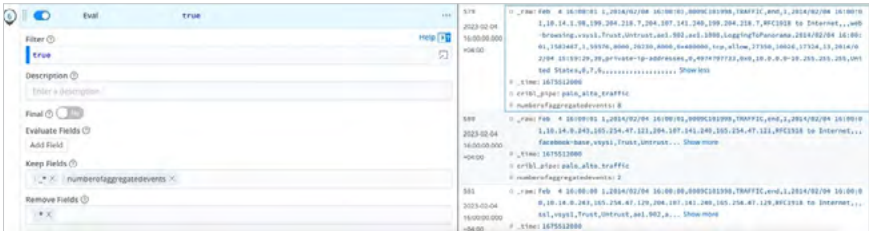
Keep Fields

Remove Fields

5 ☒ Serialize true

Aggregations back to `_raw`

With those replacements, we can use a second Eval Function to clean up the output, retaining only `_raw`, `_time`, and the number of aggregated events.



Final Eval

SUPERCHARGING AGGREGATIONS OUTPUT

Cribl Stream is built on a shared-nothing architecture, meaning that each of its Worker Processes operates separately. For the steepest aggregations, you want a way to aggregate events distributed to separate Worker Processes.

You can share state across Processes using a caching tier, such as Redis. Stream has a built-in Redis Function for this purpose, and you can import Cribl's Redis Knowledge Pack to snap in prebuilt configurations around this Function.

Where an in-memory database is not an option, but where the architecture allows this, you can dedicate a separate Worker Group with a single Worker to aggregate data from a chatty source (such as a firewall in a DMZ), and get good aggregation results.

SAMPLING EVENTS

Let's turn to cases where you don't want to summarize events, because you need to send individual events to your analytics system, keeping those events intact. But you don't need to analyze *all* events. Some types of events are way less valuable than others – yet they consume storage and compute equally.

Cribl's Sampling Functions make it easy to filter these events. Their flexible configuration options make them ideal for high-volume, low-value data scenarios. And Stream's flexible routing makes it easy to isolate and sample these defined events, while letting parallel Pipelines ingest all events from high-value data streams in full fidelity.

SAMPLING FUNCTION CONFIGURATION

In Cribl Stream's out-of-the-box Sampling Function (docs.cribl.io/docs/sampling-function), you can add rules to apply sampling to the entire data stream, or to a subset that you define by field values, other simple filters, or arbitrary expressions. Multiple filters allow for smart and highly selective sampling. Some examples of rules, filtering, and individual sample rates:

- Sample all `sourcetype=access_combined` events at a 3:1 rate (pass through only 1/3 of matching events).
- Sample all `sourcetype=access_combined` events with `status=200` at 5:1.
- Sample all `sourcetype=access_combined` events with `status=200` at 5:1, but only if they're from a host that starts with `web*`.
- Sample all `sourcetype=aws:cloudwatchlogs:vpcflow` events with `action=ACCEPT`, at 8:1.

Each event emitted by this Function gets an added index-time field named `samplerate:<rate>`, which you can use in statistical functions as necessary. For example, if you've applied a 5:1 sampling rate to `status=200` events that have `sourcetype=access_combined`, then in order to estimate the original number of 200's, you can multiply the current number of these `samplerate:5` events by 5.

For all the examples below, let's assume that you have Stream running, with a Pipeline named `main` that handles our traffic of interest.

SAMPLING SCENARIO 1:

SAMPLE ALL ACCESS_COMBINED EVENTS AT 5:1

Below, we've configured a Sampling Function with an inbound `sourcetype==access_combined` filter. In the **Sampling Rules** section, we have a `true` filter that says "sample all the events that have made it this far." We've entered a **Sampling Rate** of 5, to pass through only 1/5 of these events.

The screenshot shows the configuration for a Sampling Function. It includes a Filter section with the expression `sourcetype==access_combined`, a Description field with the text "Sample ALL access_combined at 5:1", and a Final field set to "No". Below these is a Sampling Rules table with one rule where the Filter is `true` and the Sampling Rate is 5.

Filter	Sampling Rate
<code>true</code>	5

Sample all matching events at 5:1.

SCENARIO 2:

SAMPLE DIFFERENT STATUSES AT DIFFERENT RATES

Let's look at how to handle a more complex use case. Assume that among our `access_combined` events, we want to sample status `3xx` redirection events at 2:1, but status `200` success events at 5:1.

And we want to sample status `200` from local hosts at an even thinner 8:1 rate. Here, we'll mix in Stream's Regex Extract Function (docs.cribl.io/stream/regex-extract-function) to detect those statuses.

Below is the Regex Extract config, with the same inbound filter we've used before. In the Regex section, we've entered `/"\s{?<__status>[1-5][0-9]{2}}/` to extract status values from `_raw` to a Cribl internal field named `__status`. This extracted field will be available to downstream Functions.

The screenshot shows the 'Regex Extract' configuration interface. The 'Filter' field contains 'sourcetype='access_combined''. The 'Description' field contains 'Extract an internal "status" field'. The 'Final' section has a 'No' radio button selected. The 'Regex' field contains the pattern `/"\s{?<__status>[1-5][0-9]{2}}/`. The 'Source Field' is set to `_raw`.

Extract event status to an internal field

Next we have a Sampling Function, using the same inbound filter. But here, in the **Sampling Rules** section, we've entered our three different sampling rates, one per status code:

Filter: `__status == 200 && host.endsWith('.local')` Rate: 8

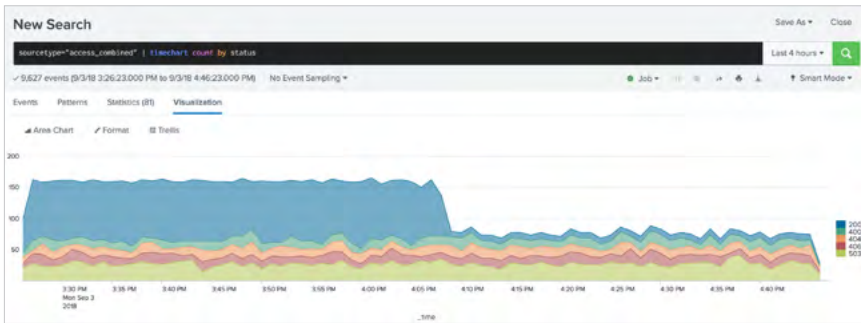
Filter: `__status == 200` Rate: 5

Filter: `__status.startsWith('3')` Rate: 2



Three rules specify three different sample rates

Downstream, we can see a substantial overall reduction of event flow, starting when our sampling was applied.



Extract event status to an internal field

SCENARIO 3: FILTER BY ACTION

To wrap up these examples, let's assume that we want to sample only events whose origin is `aws:cloudwatchlogs:vpcflow`, with action `ACCEPT`, at 6:1.

Similar to Scenario 2, we'd first use a Regex Extract Function to extract a Cribl internal field called `__action`. Then we'd add a Sampling Function configured as you see below.



Extract event status to an internal field

FINDING CANDIDATES FOR SAMPLING

To determine the best candidates for sampling, one approach is to first identify a source/sourcetype of interest, and then optionally `count by` another field. Some examples, in the same vein we've examined:

```
sourcetype=access_combined | stats count by status
sourcetype=aws:cloudwatchlogs:vpcflow | stats count by action
```

Ideal sources are those where `count by` results span at least one order of magnitude. Here are some potential candidates for sampling:

WEB ACCESS TYPE LOGS

- Apache Access
- AWS ELB/ALB
- Amazon Cloudfront
- Amazon S3 Server Access

FIREWALL TYPE LOGS

- Cisco ASA
- Palo Alto Firewall
- Amazon VPC Flow Logs
- Juniper Firewalls

SIMPLE SAMPLING WRAP-UP

When you're analyzing operational trends, having an appropriately sampled subset of data is definitely better than having no data at all. Or, as Winston Churchill didn't exactly say: "Sample data beats no data 5:1."

VARIATION: DYNAMIC SAMPLING

If you want even greater control over data volume, Cribl Stream provides a Dynamic Sampling Function. Compared to simple sampling – where you select a static sampling rate – dynamic sampling allows for adjusting sampling rates based on incoming data volume per sample group. You just set the aggressiveness/coarseness of this adjustment, and the Function does the rest automatically.

ADVANTAGES OF DYNAMIC SAMPLING

Keyed Sample Groups: You can define sample groups by a key expression, and give each its own sampling rate. Here's a sample key: ``${domain}:${httpCode}`` – meaning that each combination of `domain` and `httpCode` will get its own sample rate, without knowing their values a priori. This is pretty powerful on its own!

Sample Rate based on volume: This Function will dynamically calculate the sample rate for each key group based on its volume in the previous sampling period. You set the aggressiveness of sampling through a **Sample Mode** setting, with two options:

- **Logarithmic (natural) mode.** Default, less aggressive than Square Root.
Example: `currentRate = Math.ceil(Math.log(lastPeriodVolume))`
- **Square Root mode.**
Example: `currentRate = Math.ceil(Math.sqrt(lastPeriodVolume))`

Lower chance of starvation: Because simple sampling applies a static sample rate to all groups, lower-volume groups might be starved beyond a meaningful number. With dynamic sampling, you set a number of **Minimum Events** that must be received (in the previous sample period) in order for the sampling mode to be applied. If the Dynamic Sampling Function receives fewer events than this threshold value, it substitutes a uniform 1:1 sample rate.

DYNAMIC SAMPLING EXAMPLE

Let's see how this works. As each event passes through, the Function evaluates it against the **Sample Group Key** expression to determine which sample group to associate it with.

When a sample group is new, it will initially have a sample rate of 1:1 for the first **Sample Period (sec)** interval. (You can define the interval, with a

default of 30 seconds.) Once this period has elapsed, the Function will derive a sample rate, using the sample group’s event volume during the preceding sample period.

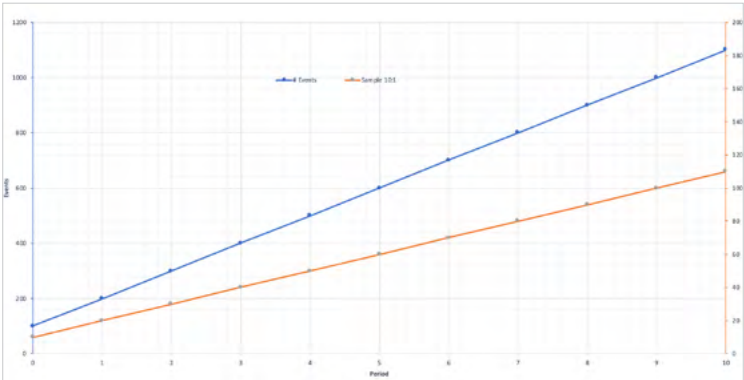
This sample rate will differ depending on whether you select Logarithmic or Square Root mode. To see how this works, assume the following data stream, with a continually increasing number of events:

Period	# Events	Sample 10:1	Dynamic: LOG	Dynamic: SQRT
0	100	10	100	100
1	200	20	40	20
2	300	30	50	20
3	400	40	67	23
4	500	50	84	25
5	600	60	86	27
6	700	70	100	28
7	800	80	115	30
8	900	90	129	32
9	1000	100	143	34
10	1100	110	158	35

Three rules specify three different sample rates

SIMPLE SAMPLING: 10:1

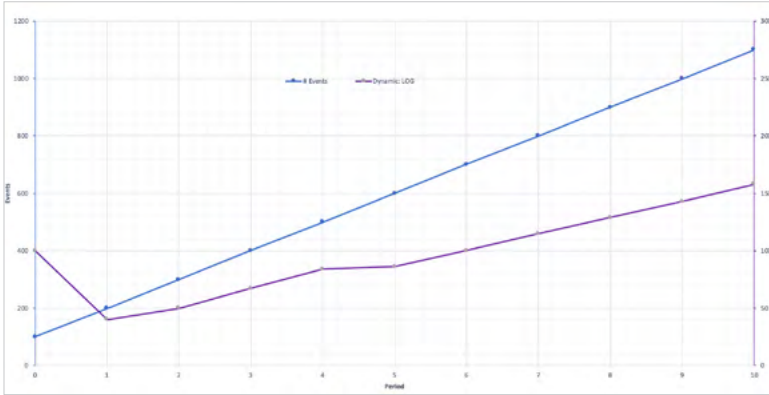
Notice the linear relationship between the two lines. The lower line indicates sampled events. (Its axis is on the right, and at a different scale.)



Simple sampling at a 10:1 rate

DYNAMIC SAMPLING IN LOGARITHMIC (NATURAL) MODE

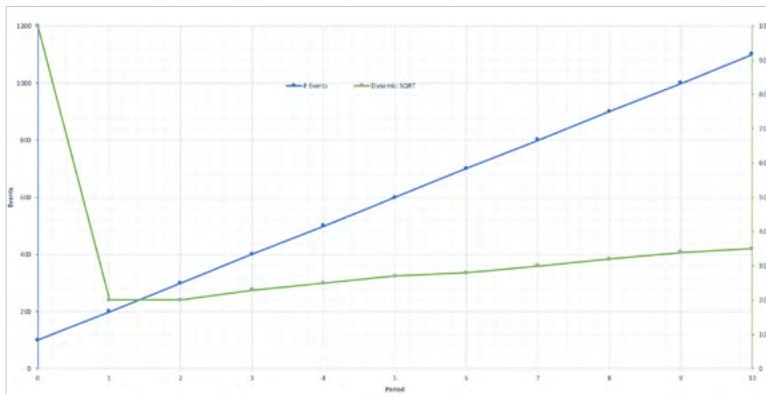
Notice the relationship between the two curves. The lower curve indicates dynamically sampled events. (Again, its axis is on the right, and at a different scale.)



Dynamic sampling, Logarithmic (natural) mode

DYNAMIC SAMPLING IN SQUARE ROOT MODE

Notice how aggressive Square Root mode is. The L-shaped curve's axis is on the right, and at a different scale.



Dynamic sampling, Square Root mode

CONFIGURING DYNAMIC SAMPLING

Assume we want to dynamically sample based on the unique values of an `ip:port:action` tuple (similar to the examples above). We could configure Stream's Dynamic Sampling Function as shown below. Other than the **Sample Group Key** expression, most other settings here are simply the Function's defaults. Now sit back and enjoy automatic, volume-adaptive dynamic sampling in action.

6Dynamic Samplingtrue

Filtertrue

DescriptionDynamic sample based on <ip:port:action> tuple

Final

Sample Mode*Logarithmic

Sample Group Key*`'${ip}:${port}:${action}'`

ADVANCED SETTINGS

Sample Period (sec)30

Minimum Events30

Max Sampling Rate100

Dynamic Sampling configuration: key expression with defaults

WHAT'S NEXT?

You can explore these techniques for yourself in our free Sandbox titled **Getting to Know Cribl Stream** (sandbox.cribl.io/course/deepdive-stream). Here, you'll find a live Stream instance and sample data. The Control section covers Stream's Aggregations, Sampling, and related Suppress Functions.

For further details on these and other Cribl Functions that help you reshape and reduce streaming data, check out our online documentation:

- Aggregations Function (docs.cribl.io/stream/aggregations-function).
- Sampling Function (docs.cribl.io/docs/sampling-function).
- Dynamic Sampling Function (docs.cribl.io/stream/dynamic-sampling-function).
- Suppress Function (docs.cribl.io/stream/suppress-function).
- Parser Function (docs.cribl.io/stream/parser-function).
- Regex Extract Function (docs.cribl.io/stream/regex-extract-function).
- Redis Function (docs.cribl.io/stream/redis-function).
- Eval Function (docs.cribl.io/stream/eval-function).



03 | Mask and Redact Sensitive Data

Confidentiality with Convenience in Cribl Stream

by Joseph Eustaquio, Cribl

PROBLEM

Removing sensitive information from log events, metrics, and traces is important but time-consuming work. System administrators can find themselves burdened with developing, tuning, and testing complex regex statements to catch every possible pattern in multiple formats and changing datasets.

SOLUTION

We'll look at using Cribl Stream's graphical UI and built-in Functions to both hash and redact sensitive data from incoming log events. We'll consider two scenarios: addressing known patterns with regex using Stream's Mask Function; and extracting and hashing more-elusive patterns, using Stream's Parser and Eval Functions.

WHY CONSIDER THIS APPROACH?

Cribl Stream facilitates sanitizing data in motion before it reaches your downstream systems of analysis. You can operate at the field level, and can match patterns using plain English. If you choose to use regex, you can validate your statements as you build them. Either way, you can test your logic by visually previewing its effects on samples of your data.

KNOWN PATTERNS: MASK FUNCTION WITH REGEX

In this first scenario, we know the patterns of the sensitive data in our logs, and we can easily find these patterns in one or more strings. We'll solve the problem like this:

- Identify the patterns of sensitive data.
- Add a Stream Mask Function to match and anonymize those patterns, using regex.
- Validate the regex, without leaving Stream.
- Preview and test the Function's effects on a sample, before applying it to live data.

Assume that while onboarding a new dataset, our business team identified three fields that need to be addressed. If they are present, we want to either do an md5 hash of the original values, or replace them, as shown below.

Sensitive Field Content	Field Label	Action if Found
Social Security Number	social	Hash
Electronic Serial Number	esn	Redact with a string of 12 "X" characters
Card Number	cardNumber	Mark as "Removed"

Sure enough, when previewing samples of our data in Stream, we see these sensitive fields – and their values – in cleartext.

[illegible]

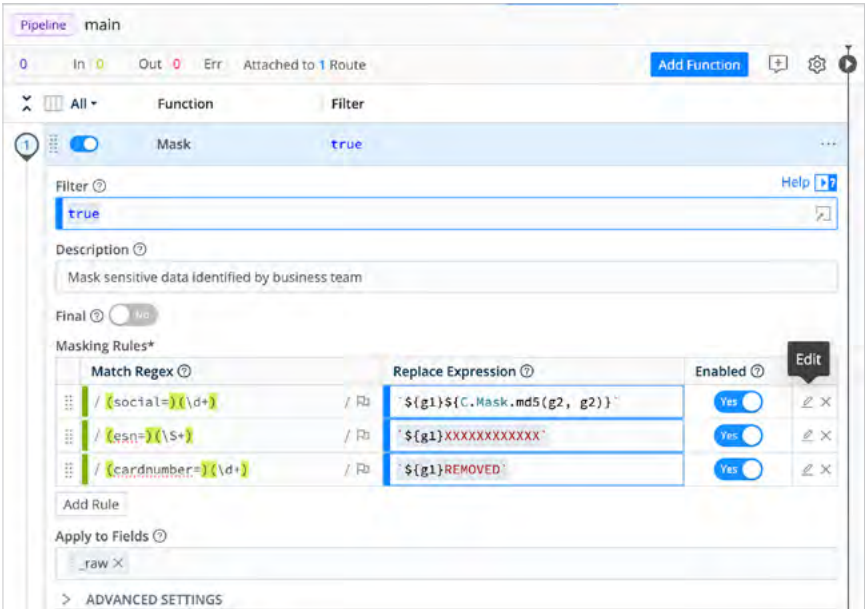
Previewing log events shows sensitive data

BUILD THE REGEX

Because we know the patterns (field names) to look for, we can build and validate some relatively simple regex statements to modify the sensitive data. We'll do this by adding a Mask Function (docs.cribl.io/stream/mask-function) to a Cribl Stream Pipeline.

This is a handy Function for replacing values with simple regex matches. A key benefit here is that you can add multiple, simple regex patterns without relying on one complex, large regex pattern that might need more maintenance over time. Find the three patterns you are looking for, and replace them with the values that the business team requested.

Our Mask Function looks like this:



Mask Function to hash or redact three patterns

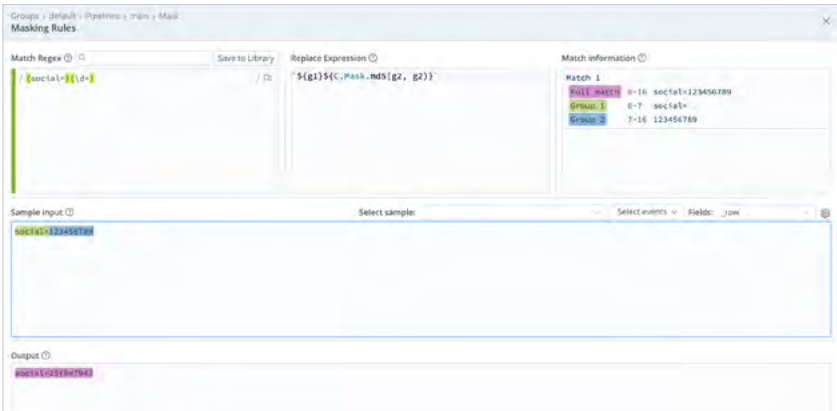
REVIEW THE LOGIC

Let's break down what we configured above:

- `(social=)(\d+)` matches the string `social=` followed by some digits. The **Replace Expression** ``${g1}${C.Mask.md5(g2, g2)}`` uses Stream's built-in `C.Mask` expression to create an MD5 hash of each of these Social Security numbers.
- `(esn=)(\S+)` matches the string `esn=` followed by non-whitespace characters. We use ``${g1}XXXXXXXXXXXX`` to redact this with a string of X's.
- `(cardnumber=)(\d+)` matches credit-card numbers. We use ``${g1}REMOVED`` to replace these with a legible word.

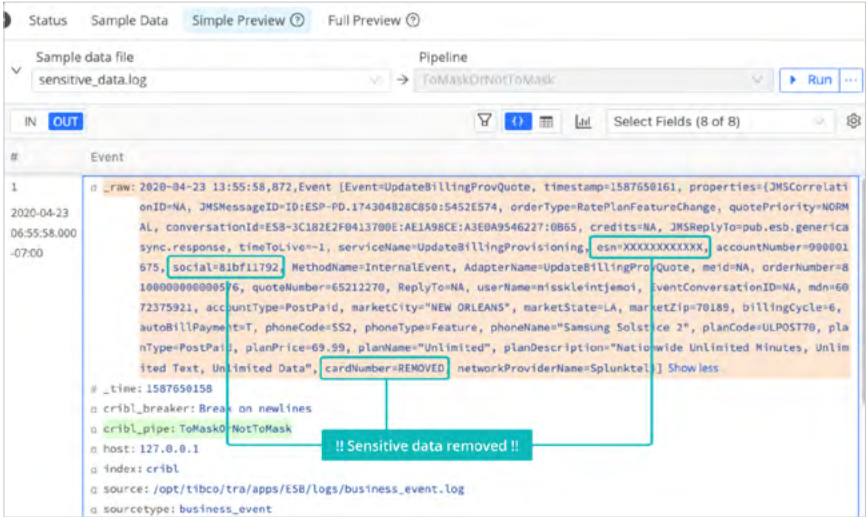
VALIDATE AND PREVIEW RESULTS

We can validate the regex as we build it up. Click the Edit button (with a pencil symbol) to the right of each Masking Rule, to test its effect on sample strings.



Validating each Masking Rule's regex

Before we deploy this configuration to affect the live data stream, we can test the proposed changes in Stream’s right Preview pane. Running the same sample data through our Pipeline, we can confirm that the three patterns are being anonymized as we intended.



Previewing Mask Function's output: Data is anonymized

EXTRACTING PATTERNS WITH PARSER AND FRIENDS

This second scenario presents a more intricate challenge, but with a more transparent (regex-free) solution. This time, let's assume that we know the patterns of the sensitive data, but we need to look for the field names in events. Here, we want to avoid regex – we'll reference the patterns in simple English, legible to a wide range of collaborators. If the sensitive info is present, we'll MD5-hash all three categories.

HOW IT WORKS

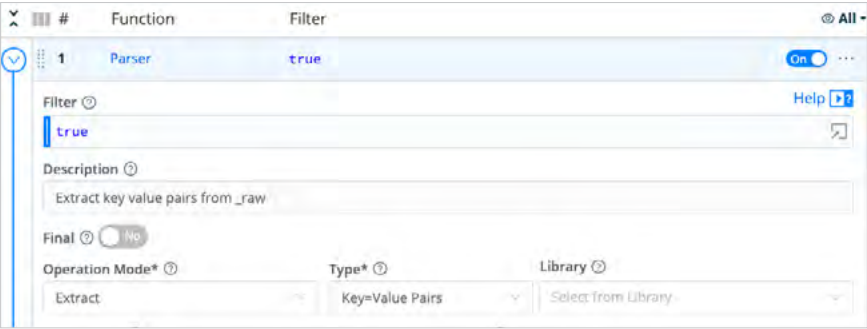
Starting with the same list of sensitive-data patterns, we'll solve the problem like this:

- Break events into fields, using a simple Stream Parser Function.
- Hash all three fields' values, using an Eval Function (no regex here).
- Reassemble the anonymized fields into one `_raw` event, using a Serialize Function.
- Optionally, strip out the events' nonessential fields, using a second Eval Function.
- Preview and verify the results on a sample, before applying it to live data.

Sensitive Field Content	Field Label	Action if Found
Social Security Number	<code>social</code>	Hash
Electronic Serial Number	<code>esn</code>	Hash
Card Number	<code>cardNumber</code>	Hash

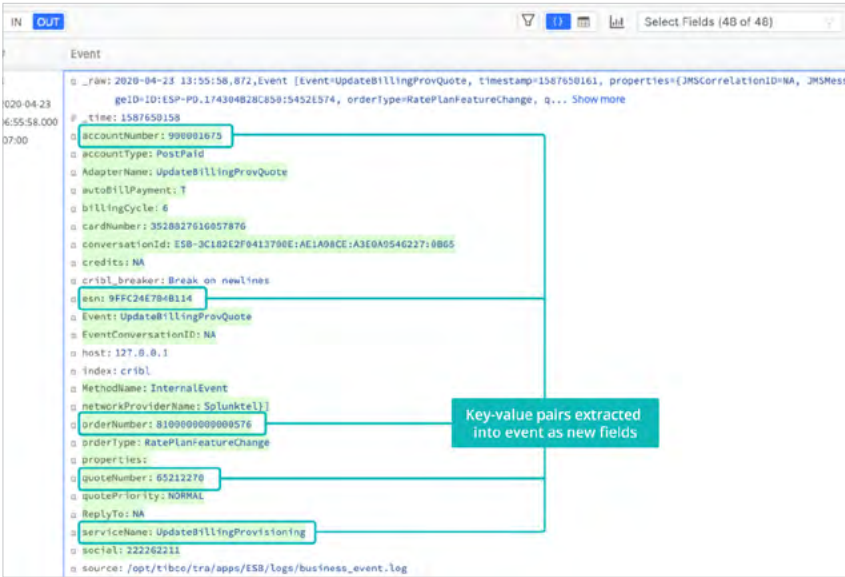
FIND FIELDS WITH PARSER

To discover and hash the secret stuff, we'll add three simple built-in Functions to a Stream Pipeline. First, we'll add a Parser Function (docs.cribl.io/stream/parser-function) to automatically identify key-value pairs nested in `_raw` events, and to extract the resulting fields with their values. So, in the simple Parser config below, we select the matching **Operation Mode** and **Type**.



Parser configured to extract key-value pairs

When we save the Pipeline and preview its output, we clearly see the key-value pairs broken out of `_raw`. (The highlighting indicates fields that the Pipeline added.) Now we have fields and values to work with.

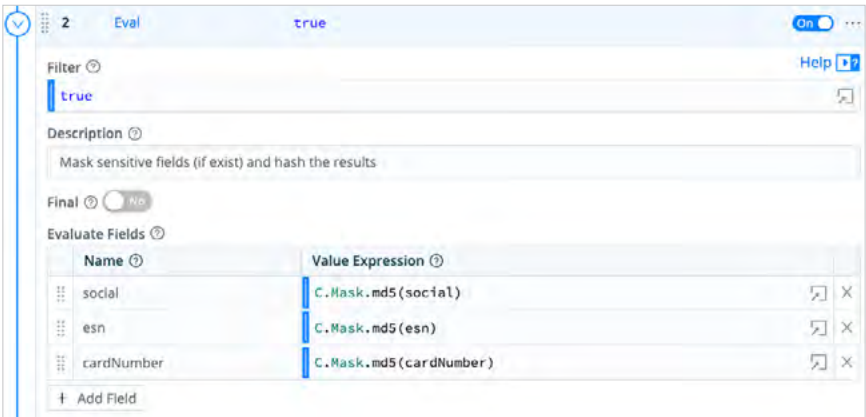


Parser configured to extract key-value pairs

HASH VALUES WITH EVAL

With the fields broken out, we'll next add an Eval Function (docs.cribl.io/stream/eval-function) to our Pipeline. Eval works on fields, offering options to remove, keep, or (in this case) transform them.

Here, we're specifying the same three sensitive data fields by **Name**. If they're present in events, our **Value Expression** column calls the same `C.Mask.md5()` expressions to hash them. We configure all this in cleartext – no regexes allowed in the clubhouse!



Eval configured to hash the bad stuff

Save the Pipeline, preview its output, and we can confirm that all three fields' values are getting hashed as we want.

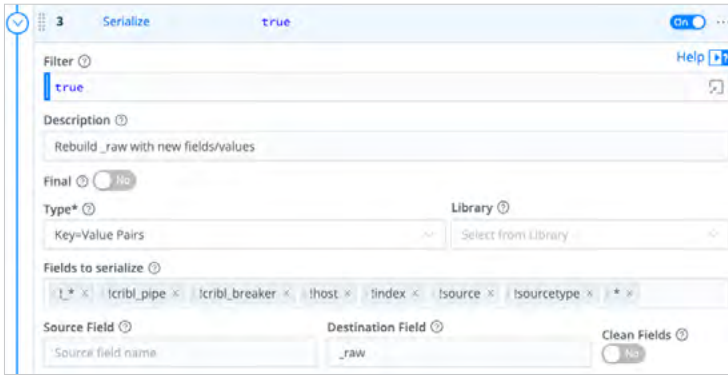
```
IN OUT Select Fields (48 of 48)
Event
020-04-23 13:55:58,872,Event [Event=UpdateBillingProvQuote, timestamp=1587659161, properties={JMSCorrelationID=NA, JMSMess
6:55:58.000 geID=ID:ESP-PD.174304828C850:5452E574, orderType=RatePlanFeatureChange, q... Show more
37:00
  _time: 1587659161
  accountNumber: 990991675
  accountType: PostPaid
  AdapterName: UpdateBillingProvQuote
  autoBillPayment: T
  billingCycle: 6
  cardNumber: 5f64941b1626420de8a4ee17bf91179
  conversationId: ES9-3C182E2F0413700E:AELA98CE:A3E6A9546227:0865
  credits: NA
  cribl_breaker: Break on newlines
  cribl_pipe: ToMaskOrNotToMask
  esn: 130487ec6257ed3a12c385e4b6b1f6af
  Event: UpdateBillingProvQuote
  EventConversationID: NA
  host: 127.0.0.1
  index: cribl
  JMSCorrelationID: NA
  JMSMessageID: ID:ESP-PD.174304828C850:5452E574
  JMSReplyTo: pub.esb.genericasync.response
  planName: Unlimited
  planPrice: 69.99
  planType: PostPaid
  properties:
  quoteNumber: 65212278
  quotePriority: NORMAL
  ReplyTo: NA
  serviceName: UpdateBillingProvisioning
  social: 81bf11792d5b55c86d0415e73b1d493a
  source: /opt/tibco/tra/apps/ESB/logs/business_event.log
  sourcetype: business_event
```

Targeted fields' values anonymized

REASSEMBLE EVENTS WITH SERIALIZE

Parser and Eval make a perfect pair, but they prefer a third wheel. To reassemble our parsed key-value pairs back into one `_raw` event, our Pipeline needs a Serialize Function (docs.cribl.io/stream/serialize-function).

In the Serialize config below, we get key-value pairs as the default **Type**, and `_raw` as the default **Destination Field**. The **Fields to serialize** are also defaults. After a bunch of `!` exclusions, the final `*` wildcard simply says to roll all remaining fields per event back into `_raw`.



Serialize: Make me one with everything

WANT MORE FOR LESS?

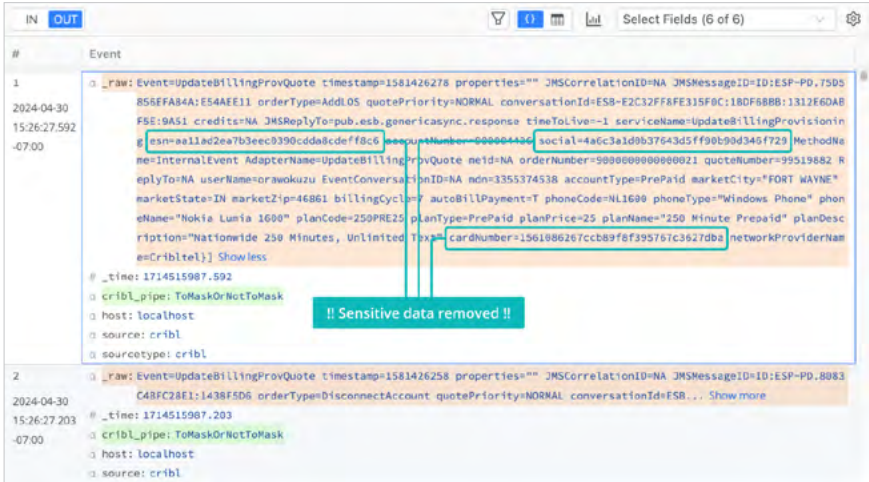
As a final step, Eval might want back into the picture. Optionally, we could append a second Eval Function to the Pipeline to slim down outbound events to only necessary fields. Here, we're relying on Eval's **Keep** and **Remove** options.

In the config below, we've specified several event fields to keep before our reassembled `_raw`, and specified one field (`_time`) to place after it. The `*` wildcard in **Remove Fields** tells Stream to remove everything else.



Eval configured to retain only essential fields

With or without that extra step, the final result is similar to what we achieved with regex in the Mask Function: In outbound events' `_raw` field, sensitive data has been anonymized.



The screenshot shows the Cribl Stream interface with two event logs. The first event (ID 1) is a JSON object containing sensitive information like phone numbers and account details. The second event (ID 2) is a similar JSON object. In both events, the `_raw` field is highlighted, and a green box with the text "!! Sensitive data removed !!" points to the redacted data in the `properties` field.

```

1  # Event
   2024-04-30 15:26:27.592 -07:00
   {
     "_raw": "Event=updateBillingProvQuote timestamp=1581426278 properties={} JMSCorrelationID=NA JMSMessageID=ID:ESP-PD.75D5-856FFAB4A:E54AEE11 orderType=Addt.05 quotePriority=NORMAL conversationId=ESB-E2C32FF8FE315F0C:180F688B:1312F6DAB-FBE:9A51 credits=NA JMSReplyTo=pub.esb.genericasync.response timeToLive=-1 serviceName=updateBillingProvisioning g esn=a11a2ea7b3ec8390cdda8cdeff8c6 accountNumber=90000000000021 social=4a6c3a1d9b37643d5ff0b99d345f729 MethodName=InternalEvent AdapterName=updateBillingProvQuote neId=NA orderNumber=90000000000021 quoteNumber=99519882 ReplyTo=NA userName=orawokuzu EventConversationID=NA ndn=1355374538 accountType=PrePaid marketCity=FORT WAYNE marketState=IN marketZip=46861 billingCycle= autoBillPayment=T phoneCode=NL1699 phoneType=Windows Phone phoneName=Nokia Lumia 1600 planCode=250PRE25 planType=PrePaid planPrice=25 planName=250 Minute Prepaid planDescription=Nationwide 250 Minutes, Unlimited Text cardNumber=1561086267ccb89f8f395767c3627db3 networkProviderName=CriblTel[] Showless",
     "_time": 1714515987.592,
     "cribl_pipe": "ToMaskOrNotToMask",
     "host": "localhost",
     "source": "cribl",
     "sourcetype": "cribl"
   }

2  # Event
   2024-04-30 15:26:27.203 -07:00
   {
     "_raw": "Event=updateBillingProvQuote timestamp=1581426258 properties={} JMSCorrelationID=NA JMSMessageID=ID:ESP-PD.8083-C48FC28E1:1438F5D6 orderType=DisconnectAccount quotePriority=NORMAL conversationId=ESB... Show more",
     "_time": 1714515987.203,
     "cribl_pipe": "ToMaskOrNotToMask",
     "host": "localhost",
     "source": "cribl"
   }

```

Outbound events hashed

WHAT'S NEXT?

You can explore these techniques for yourself in our free Sandbox titled **Getting to Know Cribl Stream** (sandbox.cribl.io/course/deepdive-stream). Here, you'll find a live Stream instance and sample data. The **Find & Replace** section covers Stream's Mask Function, and the **Parse** section covers the Parser and Eval Functions.

If you're already using Cribl Stream, the Cribl Knowledge Pack provides model Pipelines demonstrating all the above Functions, ready to modify and run. It's a free resource on the Cribl Packs Dispensary at packs.cribl.io.

For details on using all Cribl Stream Functions, see docs.cribl.io/stream.



4 | Replay Data from Object Storage with Cribl Stream

Data So Nice, You Might Ingest It Twice

by Cam Borgal, Cribl

PROBLEM

Data retention directly in analytics systems is expensive. Data retrieval from legacy formats can mismatch current analytics' requirements. What's missing is a way to replay data on demand, from a flexible choice of cost-effective storage.

SOLUTION

Cribl Stream's Collectors, and related replay features, make data-on-demand a reality. You can store full-fidelity log and metrics data in your choice of cost-effective object storage, and then later ingest only the events you need to analyze. You can use the same techniques to warm up cold data for analysis in current tools.

WHY CONSIDER THIS APPROACH?

Commodity object storage, like Cribl Lake or the Azure alternative we'll explore here, offers deep cost savings compared to storing data in systems of analysis. You can retain a full-fidelity copy of your data for longer periods, at lower cost. Combining commodity storage with Cribl Stream's replay options helps you retain more data for compliance requirements – and to also analyze more data on downstream analytics tools when you need more advanced investigations.

Replay serves both IT and Security practitioners. It facilitates re-examining historic data for incident response, threat-hunting, building new detections, and other purposes.

For an alternative approach that relies on Cribl Search to condense log events into metrics before analysis, see Chapter 7, “Faster, Better, Cheaper: Logs to Metrics with Cribl Apps.”

SCENARIO

In this example, we'll use Cribl Stream to route syslog data to Azure Blob Storage, and to then replay selected events and forward them to CrowdStrike Falcon LogScale for analysis.

WHAT WE'LL DO

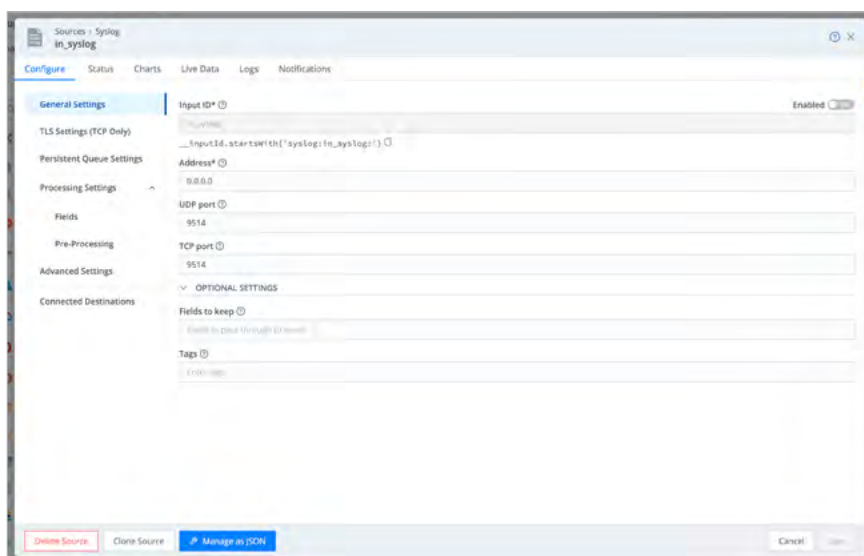
We'll walk through a full round-trip of some representative data:

- Ingest syslog data into Cribl Stream.
- Route it to your chosen object storage (Azure Blob Storage in this example).
- Tune the partitioning expression to facilitate later replay.

- Retrieve a particular subset of stored events to analyze, using a dedicated Stream Collector.
- Send only these interesting events to your system of analysis (CrowdStrike Falcon LogScale in this example).

CONFIGURE SYSLOG INPUT TO STREAM

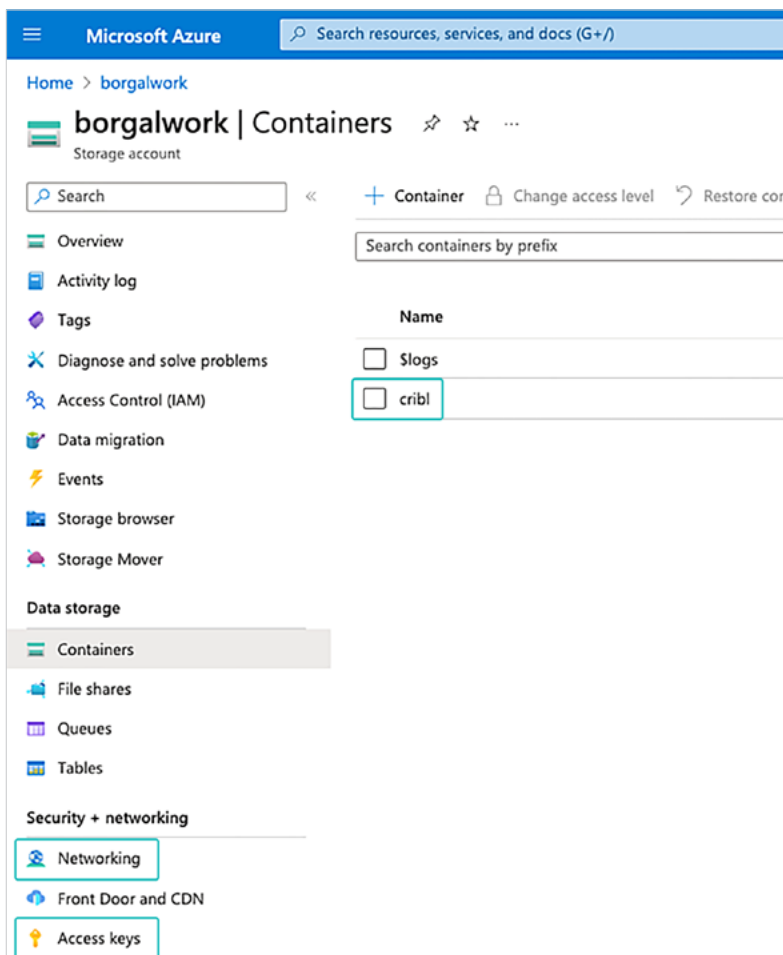
In this example, let's assume that we're monitoring syslog data. We can ingest the data by enabling Cribl Stream's preconfigured Syslog Source, using its defaults.



Stream Syslog Source, out-of-the-box configuration

CONFIGURE AZURE ACCOUNT

Here's an example configuration for the Azure Blob Storage account that we'll write to. We're using a storage account named `borgalwork`. Here, we've created a container named `cribl`, and enabled access for the appropriate IPs. Under **Access keys**, we grabbed the connection string (not shown here):



Azure Blob Storage example account setup

CONFIGURE STREAM'S AZURE BLOB STORAGE OUTPUT

To connect Cribl Stream's output to our container, we configure a matching Azure Blob Storage Destination. Here, we're specifying the `cribl` container that we created in Azure, and we're specifying `demo` as the root directory. We're accepting the default JSON output format, and under **Authentication**, we've created a stored secret to pass our Azure account's connection string.

The screenshot shows the 'Destinations > Azure Blob Storage Blob-Archive' configuration page. The 'Configure' tab is selected. On the left, a sidebar lists 'General Settings' (selected), 'Processing Settings', 'Post-Processing', and 'Advanced Settings'. The main area contains the following fields:

- Output ID***: A dropdown menu.
- Container name***: A text input field containing the value `'cribl'`.
- Blob prefix***: A text input field containing the value `''`.
- Data format***: A dropdown menu set to `JSON`.
- Authentication**: A section with a dropdown for **Authentication method** set to `Secret`.
- Connection string (text secret)***: A dropdown menu set to `Blob-Archive`.

A 'Create' button is located at the bottom right of the configuration area.

Stream Syslog Source, out-of-the-box configuration

CONFIGURE PARTITIONING EXPRESSION

In the same Destination, we configure a **Partitioning expression** to tell Cribl Stream how to form the directory structure when writing data to Azure Blob Storage. This is a key to configuring archive storage for later replay, because Stream can use the directory structure to filter and selectively replay just the data we need.

Stream's out-of-the-box Destination gives us almost everything we want here. (And the two lower expressions highlighted in the following screen

capture are just straight defaults.) In the **Partitioning expression**, we've just appended the last two terms below, to partition by host, as explained below:

```
C.Time.strftime(_time ? _time : Date.now()/1000, '%Y/%m/%d') + '/' + `${host}`
```

OPTIONAL SETTINGS

Create container ☐

Partitioning expression

Compress

File name prefix expression

File name suffix expression

Backpressure behavior

Tags

Partitioning expression

OPTIMAL PARTITIONING

When Stream filters the Azure directory structure for replay, it can use events' metadata as definitions. The full partitioning expression above references the event's timestamp, followed by the event's `host` value.

It's ideal to tune your partition expression to keep the cardinality under 2,000 on each time bucket. An excessively high cardinality can overwhelm Stream Workers with too many open files and directories. A very low cardinality will write larger files to Azure Blob Storage, which means less granularity (more data downloaded) during replay.

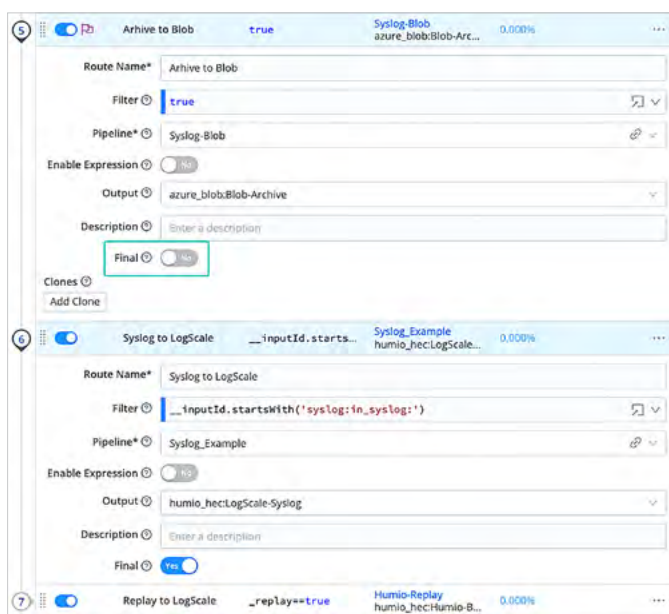
Beyond the timestamp/host combination shown here, think of common search terms that you might use on your own datasets. For web server logs, you might partition by status code and host. For firewall logs, you might partition by zone and timestamp.

ARCHIVE DATA

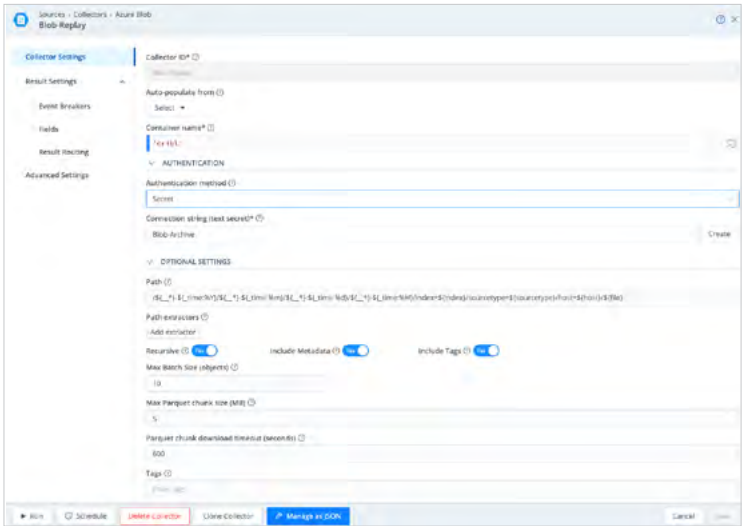
Now that we've configured Stream's syslog input and its Azure Blob Storage output, we just need to connect them. Below, we've set up a pair of parallel Stream Routes.

On top is our archival Route, which connects up the configurations we've just seen. It filters for our syslog input. Next, it sends matching data through a Stream Pipeline that we've configured to keep events' `_raw` field, but to strip out other fields. Then, the **Output** UI field's value sends the data to our configured Azure Blob Storage Destination.

The lower Route filters for the same incoming syslog data. But it sends it through a different Pipeline that we've set up to JSON-ify the syslog message without stripping fields. Then it sends this data out to CrowdStrike Falcon LogScale.



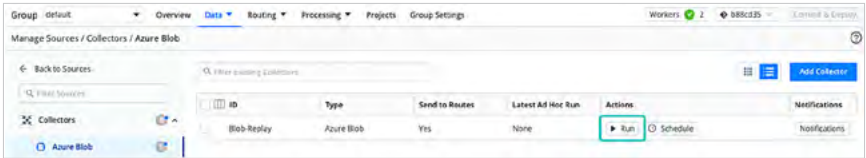
Routes to Azure Blob Storage and CrowdStrike



Adding an indicator field for replayed data

REPLAY FROM STORAGE

With the Collector configured, the real fun begins. We can now run our first replay job on this Azure data. The first step is to click **Run** beside the Collector we've configured.



Running a Collector

Next, we get several options about how to run the collection job, over what time range. In this example, we've selected a **Full Run** (collect and process any found events), and we've set a very prescriptive time range. In the **Filter**, we're looking specifically for logs from host `fw.borgal.net`. (And since we used `host` in our partitioning expression, we can use it here as a filter.)

Time range ⓘ

Earliest ⓘ

Latest ⓘ

Range timezone ⓘ

Absolute

Relative

2024-03-25 01:03:01

2024-03-25 05:07:01

UTC

Filter ⓘ

host=='fw.borgal.net'

> ADVANCED SETTINGS

Specifying a narrow Collector run

After we kick off the Collector with this configuration, we can use a link under **Latest Ad Hoc Run** to see the status of the Replay job.

Group: default

Overview

Data

Routing

Processing

Projects

Group Settings

Workers: 2

883c3d5

Logout & Logout

Manage Sources / Collectors / Azure Blob

Back to Sources

Filter Sources

Collectors

Azure Blob

Filter existing Collectors

id

Type

Send to Routes

Latest Ad Hoc Run

Actions

Notifications

Blob-Reply

Azure Blob

Yes

1711396209.2569.adhoc

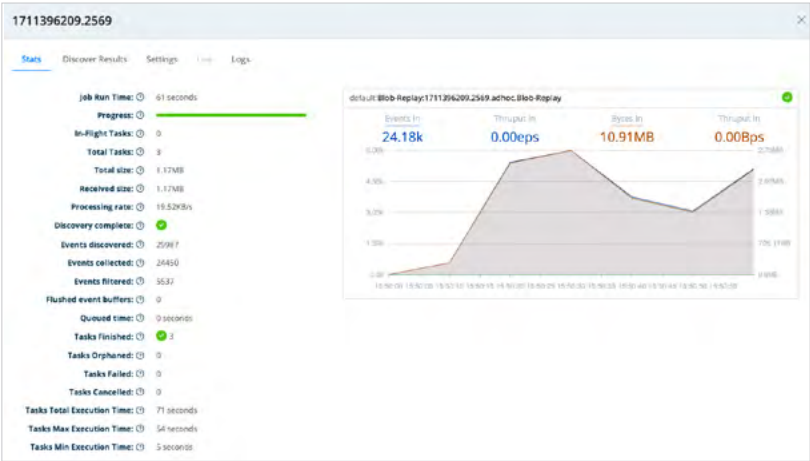
Run

Schedule

Notifications

Accessing run status

We'll also find details like the collection job's run time, events discovered and collected, etc.



Collector run details

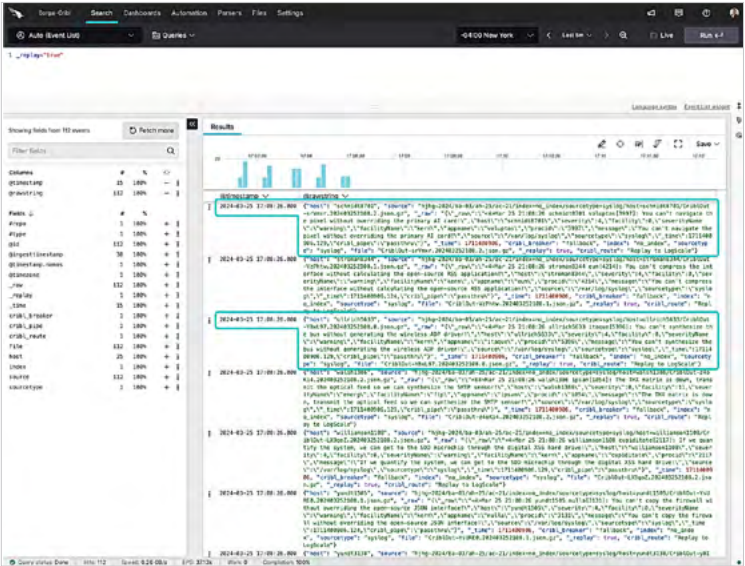
SCALING COLLECTION JOBS

Larger collection jobs can take longer, so the above window might take a while to populate. In production, it's a good practice to dedicate a Stream Worker Group to replay, in order to avoid disrupting real-time streaming data. This can be a temporary instance that's spun up only when replay is running ad hoc (as we saw here) or scheduled (which we'll get to below).

VERIFYING THE REPLAY LOOP

Finally, let's look at how these replayed events land back in CrowdStrike Falcon LogScale. (When looped back through Stream, they filter through the CrowdStrike Route, just like any other incoming syslog events.)

The highlighted events are the ones we've replayed back into CrowdStrike. They contain the original raw message in their `_raw` field, along with the `_replay` field we added to mark these events for later searching.



Collector run details

You can see how, in the parallel archival Route, we could filter for the absence of this `_replay` field to avoid storing duplicate copies of replayed events back in Azure Blob Storage.

VARIATIONS

Cribl Stream also supports replay from Cribl Lake, Amazon S3, Google Cloud Storage, the local filesystem, and other options. Stream includes other dedicated Collectors that work like the Azure Blob Storage Collector we saw in this example. This flexibility allows you to choose the option best suited to your organization's storage conventions, and to optimize for your use cases.

In addition to ad hoc collection, as we saw here, Stream makes it straightforward to schedule repeated collection runs on intervals of your choice.

To collect data from API endpoints, rather than from object storage buckets, Stream also provides a REST Collector with very flexible configuration options.

And if none of these options meets your needs, Stream also provides the option to build your own custom Collector.

WHAT'S NEXT?

You can explore Cribl Stream archiving and replay for yourself in Cribl's free Sandboxes. Here, you'll find a live Stream instance, live upstream/downstream integrations, and sample data:

- Data Collection & Replay (sandbox.cribl.io/course/data-collection).
- Full Fidelity Replay (sandbox.cribl.io/course/replay-raw).
- Archive to S3 (sandbox.cribl.io/course/s3archive).
- REST API Collector (sandbox.cribl.io/course/rest).

For further details on the techniques discussed above, see Cribl's online documentation:

- Cribl Stream Collector Sources (docs.cribl.io/stream/collectors).
- Scheduling and Running (docs.cribl.io/stream/collectors-schedule-run).
- Creating a Custom Collector (docs.cribl.io/stream/usecase-rest-create-collector).
- Cribl Lake (docs.cribl.io/lake).

If you choose to create a custom Collector, free model Collectors are available on Cribl's public Collector Templates repo at:

- github.com/cribl-io/collector-templates/tree/main/collectors/rest





SECTION 02

Simplify Tooling

5 | Tool Comparison and Migration at Blazing Speed

Manage Renewals, Evaluations,
or Mergers with No Data Loss

by Jacob Gorney and Chris Breshears, Cribl

PROBLEM

Comparing logs, metrics, or analytics systems becomes necessary in several scenarios: You might want to evaluate a replacement for a service you don't want to renew, or can't afford to renew. Or your organization might have merged with an organization that uses very different services. What's essential is to compare the alternatives against the same data, while keeping the data flowing undisturbed to current analytics and/or storage destinations until you decide where to consolidate.

SOLUTION

Cribl Stream, Cribl Edge, and Cribl Search offer you full control over your data, helping you quickly and safely make vendor-agnostic choices among

best-of-breed tools. Here, we'll show you how Cribl used our own apps to evaluate and migrate the telemetry tooling and vendors we use, working against a tight two-week deadline.

WHY CONSIDER THIS APPROACH?

In this real-world example, we'll mention several services that we considered, and the choices that we settled on. But we're not endorsing or discouraging any of these well-established vendors. Our decision process is just representative of how flexibly you can use Cribl apps to evaluate and migrate your own IT and Security infrastructure.

Also, the migration we describe here was the first step in an iterative migration toward running on Cribl's own services. Because we're migrating services that the whole company relies on – not just Cribl.Cloud monitoring – we're doing this in stages.

HANDS-ON RESOURCES

Cribl Stream and Cribl Edge can directly import Cribl Packs: prebuilt processing resources tailored to several common telemetry and security integrations. These are free resources on the Cribl Packs Dispensary at packs.cribl.io.

SCENARIO

The Cribl.Cloud team had two weeks to completely migrate away from our original telemetry tool to use something different, while minimizing the impact of the transition.

WHAT WE'LL DO

We'll look at:

- The migration challenge we were given, in detail.
- The technical requirements that framed our evaluation.
- The alternative tools we considered.
- Rapid prototyping in Cribl Stream's visual QuickConnect UI.
- Our complete bake-off, routing to four different destinations.
- The tools we landed on.
- Did we reach the finish line in time? (Cliffhanger – so please read on!)

OUR MIGRATION CHALLENGE

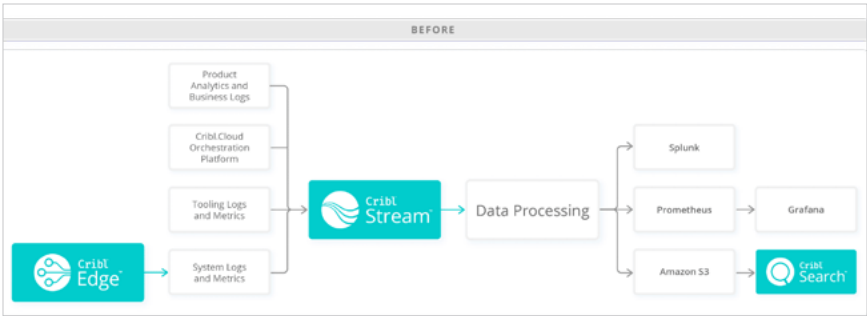
Cribl.Cloud has grown substantially since its launch, and our IT- and Security-related telemetry practice has developed in parallel. Gone are the early days of manageable logs and metrics. As we continue to grow, that problem will become even more challenging.

THE STARTING POINT: BEFORE

We used Splunk internally as our primary event management system. With Cribl Edge Nodes deployed across our entire Cribl.Cloud Fleet, we collected logs and metrics, and sent them from Edge to Cribl Stream for processing and routing.

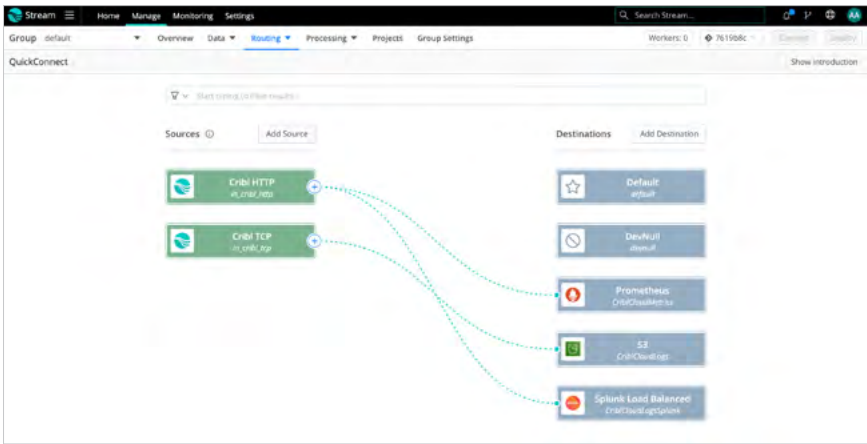
From there, we shipped data to three destinations: Splunk as our front end for events, Prometheus/Grafana for metrics, and Amazon S3 for long-term retention. Given S3's cost-effectiveness, we've used Cribl Search with our S3 store to look for information we normally wouldn't send to Splunk or Grafana for ingestion and indexing.

Schematically, our original monitoring infrastructure looked like this:



Our original data management system

Within Cribl Stream, the connections could be represented like this (in Stream's QuickConnect UI). We use the Cribl HTTP and Cribl TCP Sources to ingest data from Edge (not shown) into Stream.



Original routing within Cribl Stream

RIPPING OUT THE FOUNDATIONS

This method scaled well, getting us through our first few hundred Cribl.Cloud Organizations to thousands today. But we needed to plan for the next phase – and we needed a stay-or-go decision before our next license renewal. Perhaps you've been in a similar situation?

What happens when a critical component needs to be removed from something that has scaled well? In our case, we needed to perform a complete migration away from Splunk to another solution (or multiple solutions) within a few short weeks.

We also needed to minimize impacts across the company. Splunk had deep roots for us. Beyond our engineering teams, we'd used it across the business. It had been more than just a component of Cribl.Cloud, but also a part of Cribl's broader operations.

SO...NOW WE DO WHAT?

Our specific challenge was that we had only two weeks to complete this migration. We were up to this challenge, and we knew we had the right tools to plan and execute the transition. Namely, Cribl tools.

We use them all the time to fork data across multiple different downstream services. Cribl Stream makes it very easy to just add new Destination configurations to connect with – and evaluate – new downstream event-management solutions.

OUR REQUIREMENTS

Immediately, we broke the task into different workstreams, each representing one of the high-impact needs that our new solution(s) would need to address. These needs were:

- An easy-to-use event management platform, suitable for all Cribl employees.
- Scalable and resilient – it will need to grow with us.
- Robust query language.
- Advanced dashboarding capabilities.
- Security and user management capabilities (role-based access controls, permission policies, etc.).

INTEGRATIONS WE CONSIDERED

We knew quickly that we wanted to use Cribl Search as much as possible. Most of our data from Cribl Edge ends up in Amazon S3 for long-term retention, so Search is a perfect fit.

But we also knew that Cribl Search was a new product for us, and might not give us all the out-of-the-box functionality that we needed, at least until we built that capability into the product. So beyond increasing our internal adoption of Cribl Search, we considered the following two platforms to fill the gaps.

ELASTIC / OPENSEARCH

Initially, we perceived advantages and at least one disadvantage here:

- Document-driven, prefers denormalization of data.
- Kibana-based user interface with rich dashboarding.
- Many different query languages – whose differences we sought to understand.
- Scalable to meet our needs as we grow.

GRAFANA LOKI

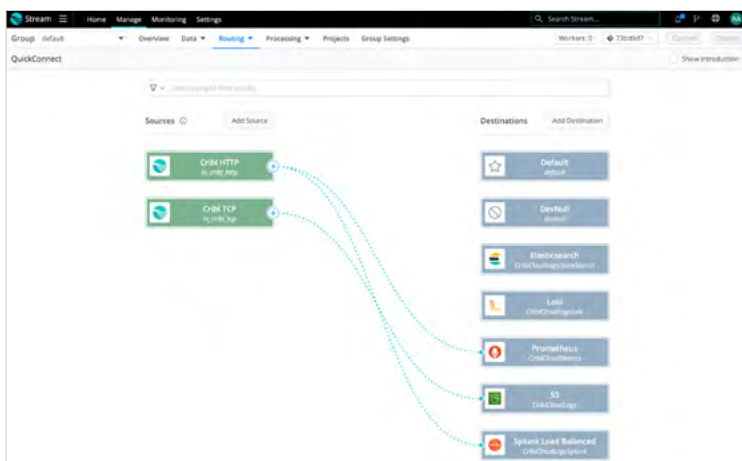
Here, too, we perceived both advantages and disadvantages:

- Included in our Grafana suite of tools. and has native integration points with Grafana for visualization.
- LogQL is similar to PromQL but can get complicated when writing more complex queries.
- Provides a similar UI experience to Grafana, but not as elegant.
- Metadata indexed as labels.

OUR EVALUATION: RAPID PROTOTYPING

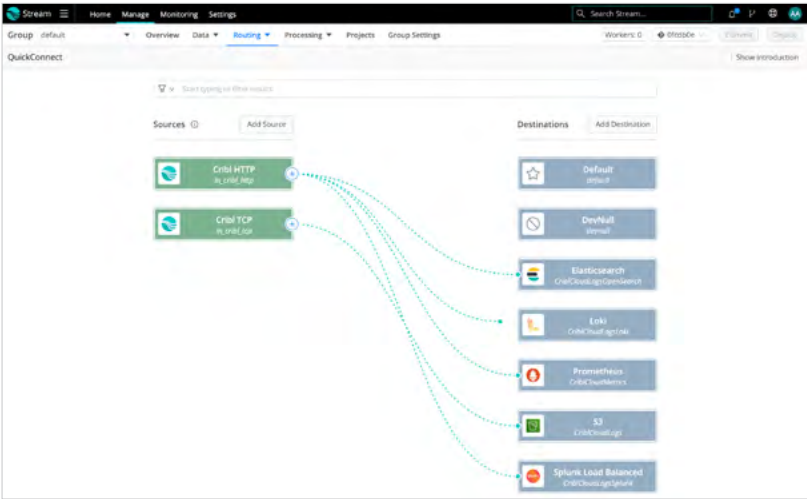
We were able to evaluate each of these tools within a day! We did this by adding two new Destinations in Cribl Stream: one for an Amazon OpenSearch Service cluster that we stood up, and another for our Grafana Loki endpoint.

Within Stream, we now had the resources shown here. OpenSearch output is handled by the compatible Elasticsearch Destination:



OpenSearch via Elasticsearch and Loki Destinations, configured

Our next step was to run duplicate Pipelines from our Cribl HTTP Source (which ingests the data from Edge) to the new Elasticsearch Destination, and then to the Loki Destination, as shown here:

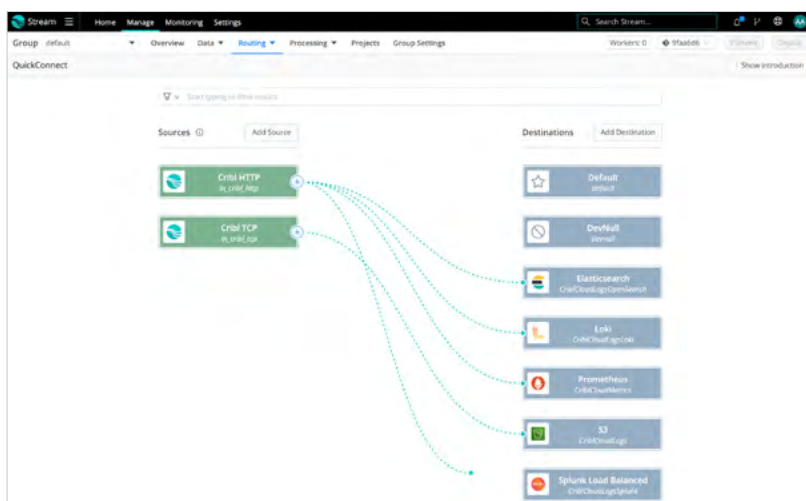


Duplicating connections to Elasticsearch and Loki Destinations

In this way, we could maintain parallel data flow to both our original and our new destinations, with zero interruption. This enabled us to carefully verify that the combination of OpenSearch and Loki (with a Grafana sidecar) met our requirements.

To preview where our evaluation ended up: These combined services provided analytics and visualizations comparable to our original downstream service. So we were confident in migrating.

At this point, we could have disconnected the parallel original Pipeline to Splunk, as previewed below. But we weren't quite done yet!

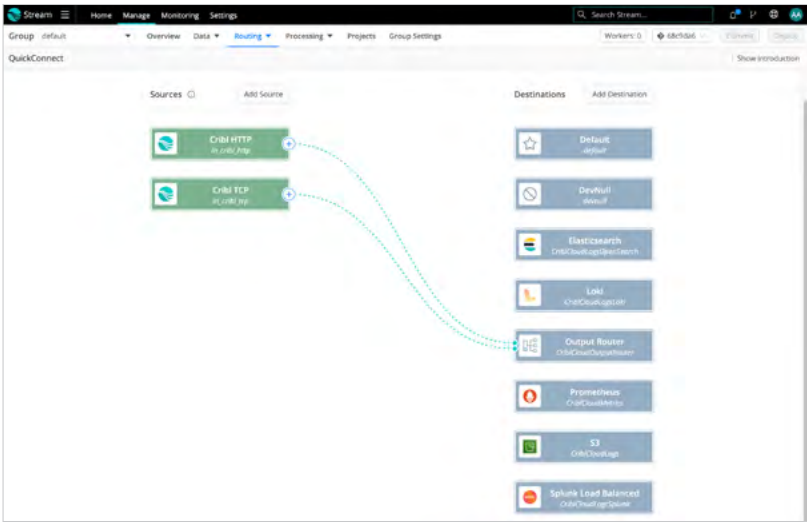


A preview of the ultimate connections

TAKING OUR EVALUATION FURTHER

To extend and ensure a thorough comparison, we did a further iteration: We added a new Output Router Destination that cloned our data across all four destinations (Splunk, S3, OpenSearch/Elasticsearch, and Loki). And we added post-processing Pipelines for OpenSearch and Loki, so that we could tweak data along the way.

Viewed in QuickConnect, as shown below, our infrastructure would now look deceptively simple. This is because the Output Router Destination's internal filtering rules now handled the routing from both the Cribl HTTP and Cribl TCP Sources to the four outputs.



All routing now managed by Output Router Destination

After this further analysis, we determined that we were ready to cut over to our new downstream services.

WHAT WE DISCOVERED

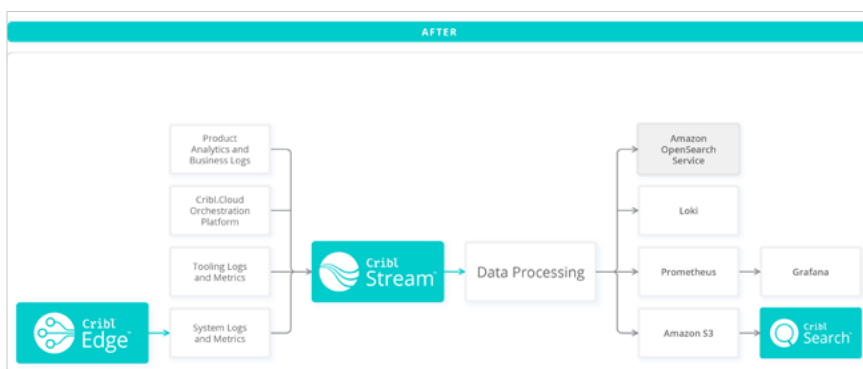
By realistically evaluating prospective new tools against our live data, we quickly learned that instead of simply selecting one tool to replace our event management system, we needed to use many. Here were our selections, by our needs:

- We'd heavily used Splunk to derive metrics from events. We identified those use cases, migrated those metrics to Prometheus, and added Grafana for visualization.
- We'd also used Splunk to parse diagnostic data on Cribl products. We easily solved those use cases using Amazon S3 storage and Cribl Search.

- OpenSearch is good at exploring and filtering event data, but it is not as powerful as other tools for creating dashboards. Also, it doesn't provide an easy way to pull in metric data from Prometheus. So instead of figuring out how to write complex queries in OpenSearch – some of which would not work – we integrated our OpenSearch endpoint as a data source in Grafana, using Grafana's OpenSearch plugin. Now, we can use Grafana as our dashboarding tool of choice, and show views with filtered event data and metrics.
- We saw an opportunity to use Loki to store AWS CloudTrail event data that we instrument for Cribl.Cloud – keeping it in S3, and using Cribl Search to query it.

THE FINISH LINE: AFTER

We launched our new strategy ahead of schedule. Our support, analytics, business, and engineering teams now use OpenSearch, Cribl Search, and Grafana without significant interruption. People are happy. Our data flow now looks like this:



Our data management system, after migration

We've taken this opportunity to level up our entire company's telemetry tools. From metrics stored in Prometheus and visualized using Grafana,

to event data in OpenSearch, to diagnostic data in Cribl Search, we have one major takeaway: It's normally hard to make split-second data decisions at scale, but Cribl's own tools facilitated this nicely. We were gratified that Cribl Stream enabled us to evaluate new solutions quickly, and to change our strategy at a moment's notice.

VARIATIONS

With Cribl Lake now available, and Cribl Search continually adding new capabilities, we're now evaluating migrating S3 and OpenSearch to Cribl's own object storage and search solutions, respectively. We're already using Cribl Search ([cribl.io/search](https://docs.cribl.io/search)) as the front end for several of our OpenSearch datasets.

WHAT'S NEXT?

For details about the techniques reviewed here, see these online resources:

- OpenSearch as a Cribl Search dataset provider (docs.cribl.io/search/set-up-opensearch).
- Cribl Lake (docs.cribl.io/lake).
- Splunk HEC Source (docs.cribl.io/stream/sources-splunk-hec).
- Splunk HEC Destination (docs.cribl.io/stream/destinations-splunk-hec).
- Elasticsearch Destination (docs.cribl.io/stream/destinations-elastic).
- Amazon S3 Destination (docs.cribl.io/stream/destinations-s3).
- Output Router Destination (docs.cribl.io/stream/destinations-output-router).
- QuickConnect (docs.cribl.io/stream/quickconnect).



6 | Kubernetes Monitoring Simplified with Cribl Edge

by Raanan Dagan, Cribl

PROBLEM

Monitoring Kubernetes logs, metrics, and traces is essential to ensuring that your containerized applications stay healthy. But monitoring dynamic Kubernetes environments – relying on multiple agents, across multiple containers – is cumbersome. It can feel like trying to navigate a busy city at night with only a single, flickering street lamp. Furthermore, sending all K8s data into analytics systems is costly.

SOLUTION

Cribl Edge can act as a central hub and overhead camera for all your Kubernetes data. It provides global monitoring, administration, and GUI-based ease of use that are lacking in other agents. Edge can help you access huge amounts of endpoint data, improving incident analysis by vastly increasing the scope of investigation.

Edge integrates smoothly with other Cribl applications, opening up Cribl Stream's power to transform and route your data, and Cribl Search's options for efficiently searching static data at its origin.

WHY CONSIDER THIS APPROACH?

Cribl Edge's centralized management and common UI can simplify your monitoring of multiple K8s clusters. As a comprehensive solution for collecting, processing, and forwarding data, Edge enables you to:

- Centralize data collection, configuration, and upgrades across multiple containers – eliminating the need to juggle multiple agents and tools on multiple containers.
- Simplify deployment into your Kubernetes clusters, using prebuilt Helm charts. You automatically get an Edge instance on each node.
- Efficiently parse, filter, and enrich K8s logs.
- Identify root causes of metrics like resource utilization spikes.
- Correlate application log events to metrics on Pods, nodes, and clusters.
- *Teleport* into individual clusters (that is, directly manage them) when you need to.
- Get real-time insights, by forwarding Edge data to your chosen analytics and visualization tools.

However, Cribl Edge might not be optimal for you if:

- You already have a well-established monitoring solution that meets your specific needs, and that integrates seamlessly with your existing infrastructure.
- You have a small, resource-constrained Kubernetes environment where a simpler, lightweight monitoring solution might suffice.

WHAT WE'LL EXPLORE

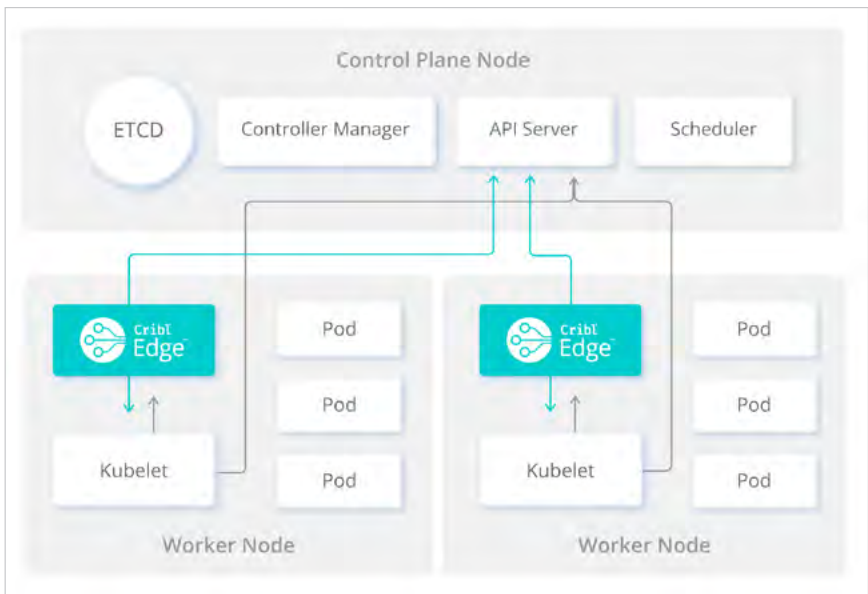
We'll first look at how Cribl Edge can collect and streamline K8s logs. Next, we'll look at how Edge can collect K8s metrics, and can spool them to disk for efficient querying and analysis in Cribl Search.

COLLECTING KUBERNETES LOGS

The three most common approaches to collecting logs from Kubernetes are:

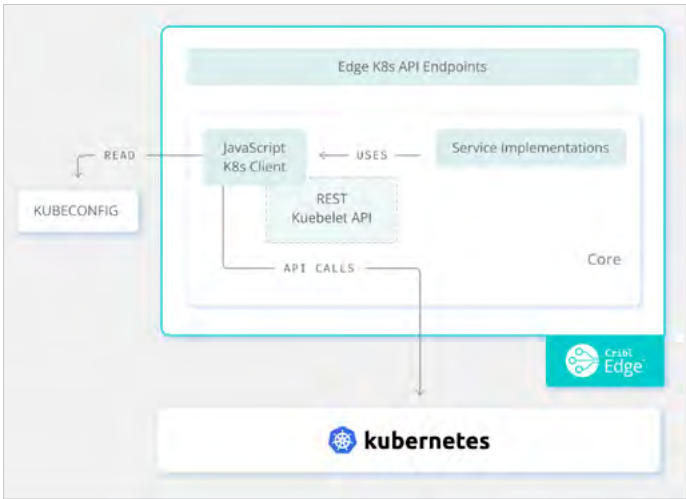
1. Use a *node-level logging agent*, running on every node.
2. Include a dedicated *sidecar container* for logging per application Pod.
3. Push logs directly to a backend from within the monitored application.

Cribl Edge follows a different approach. Cribl recommends that you deploy Edge as a Kubernetes DaemonSet. This enables an Edge logging agent to run on every node in a K8s cluster, at the node level, while optimizing logs collection.



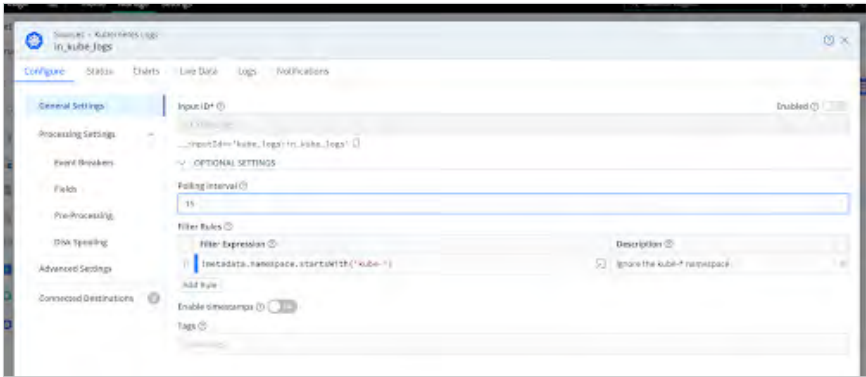
Edge agent per node

Cribl Edge ingests logs from the agents via its Kubernetes Logs Source (docs.cribl.io/edge/sources-kubernetes-logs). This integration connects to the Client JavaScript API (github.com/kubernetes-client/javascript).



Edge logs ingestion from K8s

The Kubernetes Logs Source loads the lists of Pods on each node, doing so on a configurable polling interval. In this Source's config modal below, you can see the interval's default: once every 15 seconds.



Kubernetes Logs Source configuration

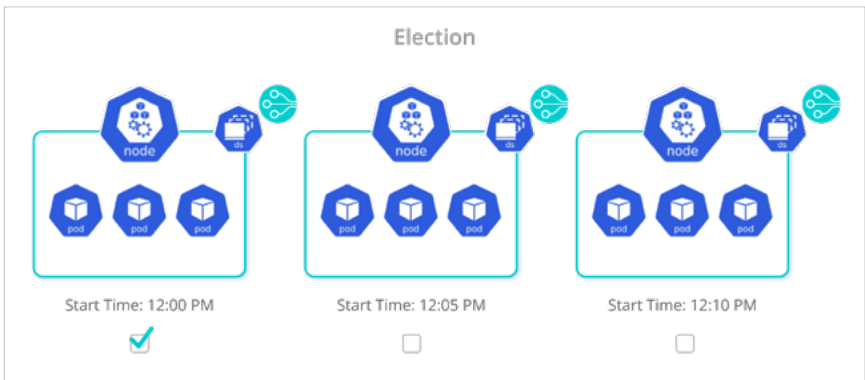
FILTERING PODS

With great data centralization comes great data volume! Luckily, you can define multiple Filter Rules to select which Pods to report on. Above, you

COLLECTING KUBERNETES METRICS

Cribl Edge provides a dedicated Kubernetes Metrics Source (docs.cribl.io/edge/sources-kubernetes-metrics), with configuration options similar to those we saw for K8s logs. You get the same options to specify metrics collection, via Filter Rules. But the most important guardrail against overconsumption happens behind the scenes.

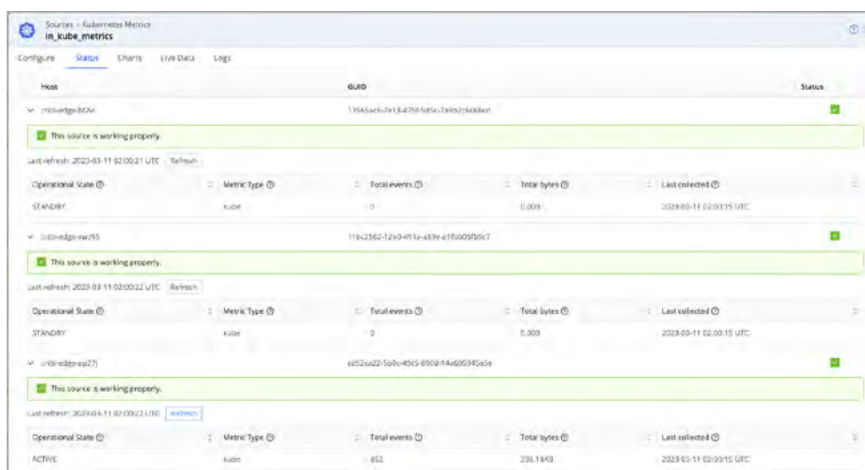
To avoid collecting the same metrics from multiple K8 nodes, Edge uses an election algorithm to sample your cluster's overall health. This election happens every 5 minutes, across all of the cluster's nodes. The oldest node in the cluster wins.



Election algorithm for sampling nodes across a cluster

METRICS FROM WHICH NODE?

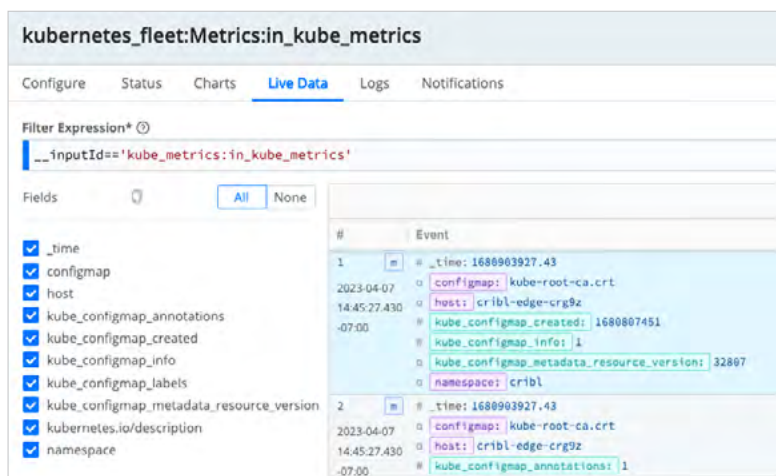
You can easily identify which node you're monitoring during each election. From the Kubernetes Metrics Source's config modal, select the **Status** tab. Then expand each host's details, and check its **Operational State** column. Here, an **Active** state indicates that a node won the election. A **Standby** state means it's waiting to be re-elected.



Who won the election? The Status tab tells you

RICH METRICS

As with Kubernetes Logs, we can examine and more narrowly filter the metrics that Cribl Edge is collecting. To do this, we select the Kubernetes Metrics Source's **Live Data** tab.



Sample K8s metrics input

HOW EDGE COLLECTS METRICS

To collect Kubernetes metrics, Cribl Edge uses the `/metrics` and `/metrics/cadvisor` endpoints on the kubelet. This information gives us a really good idea of what's going on in each node and Pod.

MAKING LOGS AND METRICS SEARCHABLE

Cribl Edge's Kubernetes Logs and Metrics integrations each provide a Disk Spooling option, with configurable time span and disk footprint. Disk spooling enables Cribl Search (cribl.io/search) to query the logs and metrics that Edge collects.

By combining Edge's flexible data collection with Cribl Search, you stoke a powerful and comprehensive data engine. Data engineers can quickly access and analyze logs and metrics of interest from K8s clusters, without needing to first ingest large datasets into other security or analytics tools. Examine, decide what you need, and forward only what's relevant.

VARIATIONS

Cribl Edge also provides a predefined Kubernetes Events Source (docs.cribl.io/edge/sources-kubernetes-events) that you can use to collect cluster-level events, relying on the Kubernetes API's watch capabilities.

See our blog post on *Managing Kubernetes Events with Cribl Edge* (cribl.io/blog/managing-kubernetes-events-with-cribl-edge) for a detailed walk-through on collecting, routing, and filtering these events, and on aggregating events to metrics and metrics dashboards.

WHAT'S NEXT?

You can explore these techniques for yourself in our free **Cribl Edge Sandbox** (sandbox.cribl.io/course/edge). Here, you'll find a live Edge instance and sample data. The **Log File Source** section covers logs ingestion, using a Source very similar to the K8s-specific Kubernetes Logs Source we highlighted above.

For details on the Cribl resources we've covered here, see our online documentation:

- Kubernetes Logs Source (docs.cribl.io/edge/sources-kubernetes-logs).
- Kubernetes Metrics Source (docs.cribl.io/edge/sources-kubernetes-metrics).
- Kubernetes Events Source (docs.cribl.io/edge/sources-kubernetes-events).
- Cribl Search (cribl.io/search).





SECTION 03

Go Deeper with Cribl Search, Lake, and AI

7 | Faster, Better, Cheaper: Logs to Metrics with Cribl Search, Stream and Lake

Achieving 99.94% Volume Reduction
and Accelerated Performance

by David Maislin, Cribl

PROBLEM

IT and Security practitioners are often forced to retain and analyze far less log data than would be ideal, because infrastructure and license costs collide with constrained budgets. As data volumes increase much faster than budgets, the problem worsens.

SOLUTION

Here, we'll demonstrate how to optimize log data by sending full-fidelity logs to cost-effective storage, converting a copy into efficient metrics, and forwarding only those essential metrics for analysis. This approach can reduce ingest costs and increase efficient use of long-term storage by an order of magnitude. And it's easy to achieve by combining Cribl Search, Cribl Stream, and Cribl Lake (or your preferred object storage).

WHY CONSIDER THIS APPROACH?

Cribl Search and Cribl Stream can skillfully navigate object storage, transform logs into actionable metrics, and seamlessly channel your really vital data to your systems of analysis. Cribl Lake offers easily configured, cost-effective object storage. In this case study, you'll see a 99.94% reduction in volume, enhanced efficiency, and substantial cost savings.

For an alternative approach that relies on Cribl Stream Collectors to ingest full log events with arbitrary filtering, this book's Chapter 4, "Replay Data from Object Storage with Cribl Stream."

SCENARIO

In the example, we're sending 132.54 MB per hour of logs to storage in Cribl Lake or an Amazon S3 bucket. Using Cribl Search, we run a scheduled search every hour to generate 70.96 KB in metrics from those stored logs. (Note the change in units below.)

We relay these metrics to a Prometheus endpoint, in this case, Grafana. Substitute your own desired metrics, data lake or object storage, and analytics, and you can expect to achieve similar results.



Big logs converted to small metrics – note the byte counts at right

This is a jaw-dropping reduction of nearly 99.94%, dramatically easing storage requirements. This kind of transformation is the power of converting logs into efficient metrics, making data management not just efficient but downright magical.

HOW IT WORKS

To unpack this, we'll look at:

- Routing full-fidelity, compressed events to cost-effective storage.
- Converting logs to metrics.
- Optionally, storing the resulting metrics in a data lake for further analysis.

ORIGINAL DATA AND STORAGE COSTS

Sending approximately 132.54 megabytes of Apache web logs per hour to an Amazon S3 bucket, consistently over a month, equates to roughly 94.29 GB of uncompressed data. As a first mitigation, we can shrink this to a lean 11.79 GB using gzip, Cribl Stream's default compression method.

EXAMPLE EVENTS: APACHE LOGS

For comparison with your own logs' verbosity, our original events look like this:

```
192.168.0.1 - - [07/JUL/2023:12:00:01 +0000] "GET /SEARCH?QUERY=LAPTOP&SESSION_ID=7F1C5D8B-2A32-4B85-9DF7-094A8E1EF19C&USER_ID=JSMITH@GMAIL.COM HTTP/1.1" 200 2048
"HTTPS://TECHBOSS.COM/" "MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64) APPLEWEBKIT/537.36 (KHTML, LIKE GECKO) CHROME/91.0.4472.124 SAFARI/537.36" 156

192.168.0.1 - - [07/JUL/2023:12:01:02 +0000] "GET /PRODUCT?PRODUCT_ID=XYZ789&PRODUCT_NAME=LAPTOP&SESSION_ID=7F1C5D8B-2A32-4B85-9DF7-094A8E1EF19C&USER_ID=JSMITH@GMAIL.COM HTTP/1.1" 200 4096
"HTTPS://TECHBOSS.COM/SEARCH?QUERY=LAPTOP" "MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64) APPLEWEBKIT/537.36 (KHTML, LIKE GECKO) CHROME/91.0.4472.124 SAFARI/537.36" 241

192.168.0.1 - - [07/JUL/2023:12:02:03 +0000] "GET /REVIEWS?PRODUCT_ID=XYZ789&PRODUCT_NAME=LAPTOP&SESSION_ID=7F1C5D8B-2A32-4B85-9DF7-094A8E1EF19C&USER_ID=JSMITH@GMAIL.COM HTTP/1.1" 200 3072
"HTTPS://TECHBOSS.COM/PRODUCT?PRODUCT_ID=XYZ789" "MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64) APPLEWEBKIT/537.36 (KHTML, LIKE GECKO) CHROME/91.0.4472.124 SAFARI/537.36" 189

192.168.0.1 - - [07/JUL/2023:12:03:04 +0000] "POST /CART/ADD?PRODUCT_ID=XYZ789&PRODUCT_NAME=LAPTOP&PRICE=1979.97&QUANTITY=1&SESSION_ID=7F1C5D8B-2A32-4B85-9DF7-094A8E1EF19C&USER_ID=JSMITH@GMAIL.COM HTTP/1.1" 302 512
"HTTPS://TECHBOSS.COM/PRODUCT?PRODUCT_ID=XYZ789" "MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64) APPLEWEBKIT/537.36 (KHTML, LIKE GECKO) CHROME/91.0.4472.124 SAFARI/537.36" 178
```

COST-EFFECTIVE VERSUS PREMIUM DATA LAKES

When storing this compressed data in an Amazon S3 bucket for just a month, the cost as of mid-2024 is a mere \$0.23 – an impressively frugal choice. Over the course of a year, this translates to an annual expenditure of approximately \$2.76, highlighting the cost-effectiveness of commodity storage. In addition to delivering these substantial savings, Amazon S3 (like Cribl Lake) provides top-notch features such as high availability and data durability. Cribl Stream replay and Cribl Search natively support retrieval from S3 or Cribl Lake.

In contrast, many premium search vendors demand prices well above \$600 per year for similar data volumes – making them at least 185 times more costly. Now, picture the potential financial impact if we were dealing with terabytes or even petabytes of data instead of megabytes.

By using commodity object storage, you not only save on storage costs, but also gain the flexibility to schedule Cribl searches, and to generate a wide array of metrics across any desired timeframe.

CRIBL SEARCH: LOGS TO METRICS

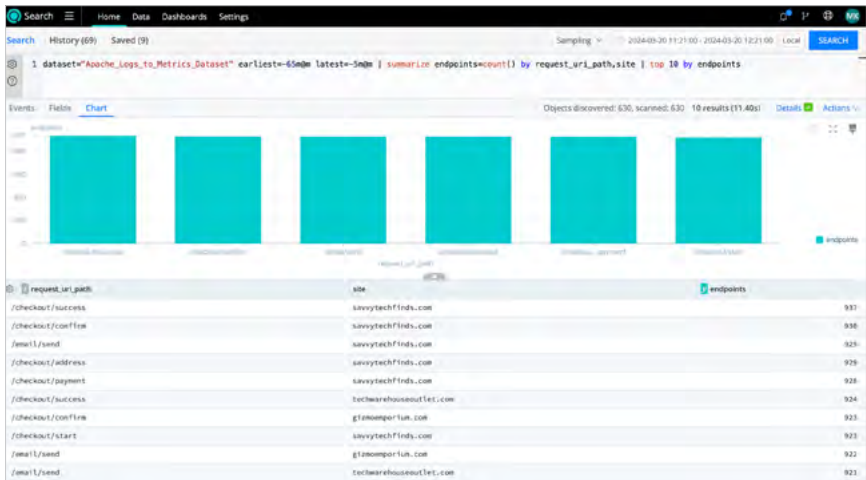
To further optimize our data, let's look at using Cribl Search to convert logs to metrics.

TOP 10 URLS BY SITE

In Cribl Search, we've created a dataset corresponding to our lake of Apache server logs. In the Search query below, we analyze them within a one-hour time window. (This query is designed to be scheduled for hourly runs.) The analysis summarizes the most requested endpoints by site, and then presents the top 10 endpoints based on how frequently they're accessed:

```
dataset="Apache_Logs_to_Metrics_Dataset" earliest=-65m@m latest=-5m@m |
summarize endpoints=count() by request_uri_path,site | top 10 by endpoints
```

Search's **Charts** tab renders results like these. Lots of `/checkout/` activity - our site is moving some product!



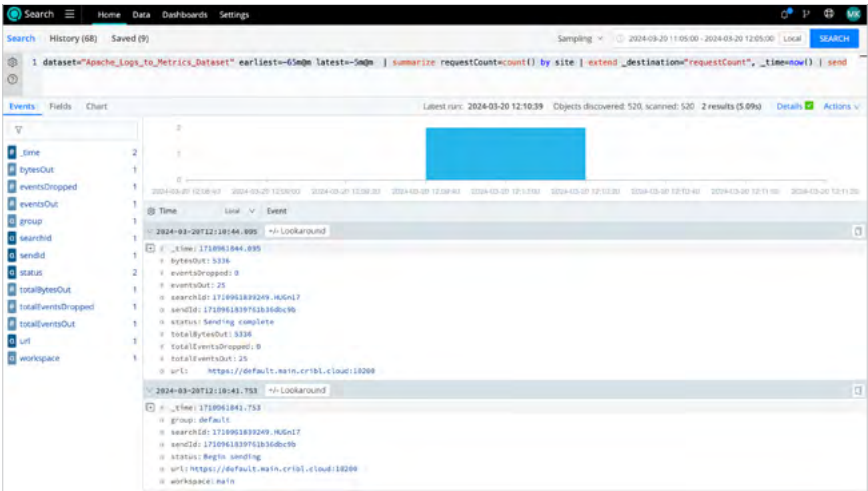
Cribl Search rendering of hourly endpoints analysis

SENDING METRICS AS EASY AS PIPE

If we want to send these metrics (as well as the raw logs) to our preferred data lake, we can just append a `| send` to the query above. This will loop them back through Cribl Stream to the lake:

```
dataset="Apache_Logs_to_Metrics_Dataset" earliest=-65m@h latest=-5m@m
| summarize endpoints=count() by request_uri_path, site | extend _
destination="endpoints", _time=now() | send
```

Notice, in the upper event below, the summarized report of how many metric events were sent back through Cribl Stream and on to our chosen object storage destination.



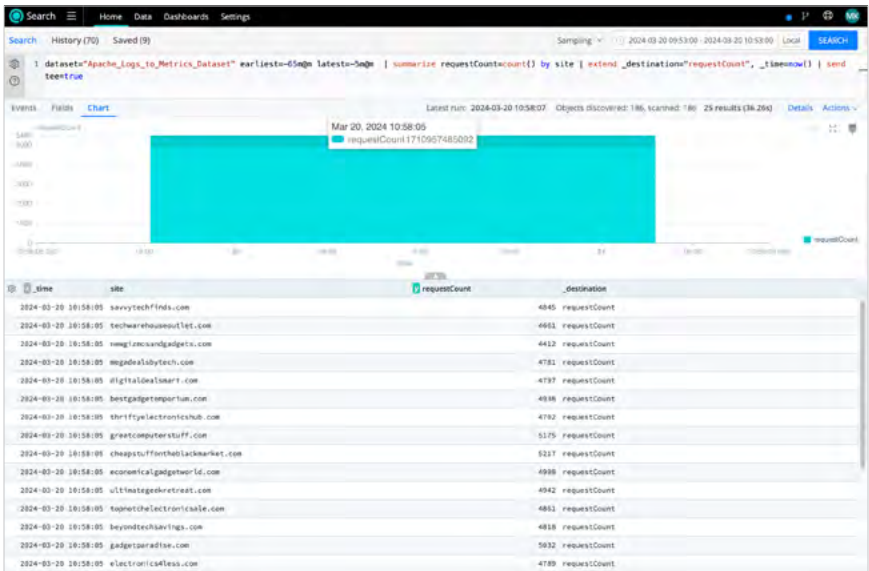
Hourly summary query and report

TEE FOR TRUE

If we want more detail than the summary provides, we can append `tee=true` after the `send` command, like this:

```
dataset="Apache_Logs_to_Metrics_Dataset" earliest=-65m latest=-5m@m
| summarize endpoints=count() by request_uri_path, site | extend _
destination="endpoints", _time=now() | send tee=true
```

This will show us the exact metrics that will be sent, as in the example below:

*Detailed view of exported metrics*

LOGS IN, VALUE OUT

So far, we've taken the bloated logs in; used Cribl Stream to compress the logs (with full fidelity), and to route them to our data lake; and then used Cribl Search to extract only our needed metrics, and to send them back through Cribl Stream.

Let's look at how Stream routes the metrics to our preferred destination. In this example, we're sending to a Prometheus endpoint in Grafana Cloud.

CRIBL STREAM ROUTES

We're using two parallel Routes in Stream. The top one routes the raw events to our data lake. The lower one routes the metrics to our Grafana analytics. Their configuration looks like this:

The screenshot shows the Cribl Stream web interface. At the top, there's a navigation bar with 'Stream' logo and tabs for 'Home', 'Manage', 'Monitoring', and 'Settings'. Below this is a breadcrumb trail: 'Group MAISLIN' > 'Overview' > 'Data' > 'Routing'. The main section is titled 'Data Routes'. It contains a table with two routes. The first route, 'Logs_2_AWS_Datalake', has a filter '.__inputId=='datagen...' and is connected to a pipeline 'PACK Apache-Logs-To-Metrics (Apache-Logs-To-Metrics)' with an output 'dl_s3:Apache_Logs_to_M...'. The second route, 'Cribl_Search_2_Grafana_Prometheus', has a filter '.__inputId=='cribl_http:in_cribl_http'' and is connected to a pipeline 'Grafana_Prometheus prometheus:Prometheus' with an output 'prometheus:Prometheus.'. Both routes have a 'Final' status set to 'Yes'.

Route	Filter	Pipeline/Output	Events (In)
Logs_2_AWS_Datalake	.__inputId=='datagen:WebFunnel'	PACK Apache-Logs-To-Metrics (Apache-Logs-To-Metrics) dl_s3:Apache_Logs_to_M...	49,335/s
Cribl_Search_2_Grafana_Prometheus	.__inputId=='cribl_http:in_cribl_http'	Grafana_Prometheus prometheus:Prometheus	1.317/s

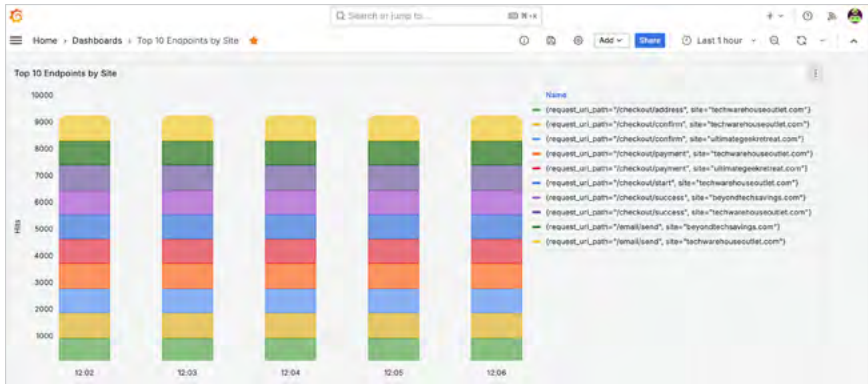
Cribl Stream dual Routes

CHARTING GRAFANA PROMETHEUS METRICS

A quick search in Grafana to display the top 10 endpoints – by site – would look similar to this query:

```
topk(10, sum(endpoints) by (request_uri_path, site))
```


The resulting dashboard would look something like this:



Metrics of interest, as displayed in Grafana

WHAT WE'VE ACCOMPLISHED

Converting logs to metrics is a game-changer in data management. This transformation can reduce data volume by nearly 99.94%, yielding substantial cost savings and operational efficiency.

Routing your full-fidelity logs to affordable, commodity object storage provides the added benefits of high availability and data durability. You'll also get easy access, thanks to features like Cribl Stream's replay via Collectors, and Cribl Search's scheduled query pipelines (illustrated above).

Using the combination of Cribl Search and Stream, running tailored searches and generating metrics is a breeze, making data optimization more accessible than ever before. Embrace this transformative shift from logs to metrics, and unlock new possibilities in data management and cost-effectiveness.

WHAT'S NEXT?

To get started with Cribl Search, sign up for a free Cribl.Cloud account at cribl.io/blastoff. Search will be ready to use immediately – start with its sample datasets or import your own working data.

For a guided tutorial on Search basics, take our free **Cribl Search** Sandbox for a spin at sandbox.cribl.io/course/overview-search. For in-depth Cribl Search documentation, see docs.cribl.io/search.

For an overview of Cribl Stream to route data to object storage and replay via Collectors (an alternative to scheduled Search queries), see Using S3 Storage and Replay (docs.cribl.io/stream/usecase-replay-s3).

Learn more about Cribl Lake at docs.cribl.io/lake.



8 | Summarize Events over Time with Cribl Search

Generating and Comparing Statistics Using `eventstats`

by David Cavuto, Cribl

PROBLEM

When analyzing a large dataset, you might need to compare individual data points against overall statistics. When does a performance metric rise above its historical average? When does that metric's variance increase past a certain threshold? What's the distribution of IP addresses connecting to your public web portal?

Uncovering such trends in your data can help you proactively address potential issues, and can provide broader context for informed decision-making.

SOLUTION

Cribl Search enables you to search data in place, and provides a dedicated `eventstats` operator to facilitate these comparisons. You use `eventstats`

to generate a group aggregate and then enrich the underlying events for comparison.

You can think of this as if Search goes through your dataset twice: First it generates the group aggregate; then it enriches each event with a field containing the group aggregate – which encompasses later events.

WHY CONSIDER THIS APPROACH?

Where you don't need to compare individual events against overall aggregates, other approaches – including Cribl Search's `summarize` and `timestats` operators – can generate both time-series and overall aggregates.

Where `eventstats` shines is in comparing individual events against aggregates, enabling you to answer more nuanced questions about specific values against a baseline number.

WHAT WE'LL DO

We'll unpack three use cases:

- Comparing individual events against an average, and then computing standard deviations.
- Calculating a particular field's values as percentages of the whole, by adding a `summarize` operator.
- Disaggregating aggregates of individual field values, by adding a `group-by` clause.

COMPARE EVENTS AGAINST AVERAGE AND STANDARD DEVIATIONS

To use Cribl Search to compare events against their aggregate, you create a query pipeline that includes the `eventstats` operator, one or more aggregation operators (with optional renaming), and zero to multiple fields to summarize.

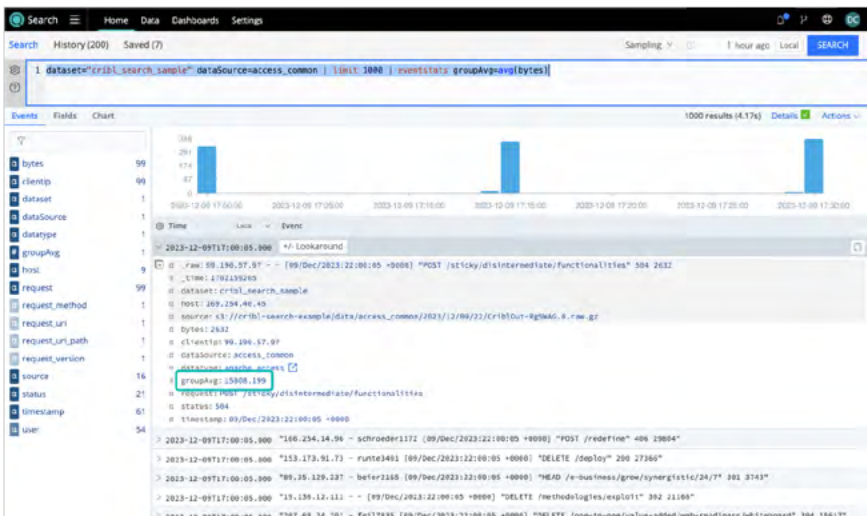
Let's look at an example. This uses Search's out-of-the-box dataset named `cribl_search_sample`, whose events simulate typical Web server and network logs.

COMPUTE AGGREGATE WITH EVENTSTATS

Assume that we're interested in events' `bytes` value. In Search's native KQL (Kusto Query Language), we can first generate a group average for this value, using `eventstats` like this:

```
dataset="cribl_search_sample" dataSource=access_common | limit 1000
| eventstats groupAvg=avg(bytes)
```

This operator computes a summary bytes statistic for the whole dataset, placing it in a field called `groupAvg`, whose value is identical in all events.



The groupAvg field added to each event

COMPARE WITH WHERE

Next, we can compare each event's value against this summary field, by appending a `where` operator:

```
dataset="cribl_search_sample" dataSource=access_common | limit 1000  
| eventstats groupAvg=avg(bytes) | where bytes > groupAvg
```

This new query returns only events whose bytes exceed the average. Running it against a limit of `1000` events, we get about half the number of results (around 500). This is unsurprising: assuming a field with uniform distribution, we would expect about half the events to be larger than the average (and half to be smaller).

FIND OUTLIERS WITH STDDEV

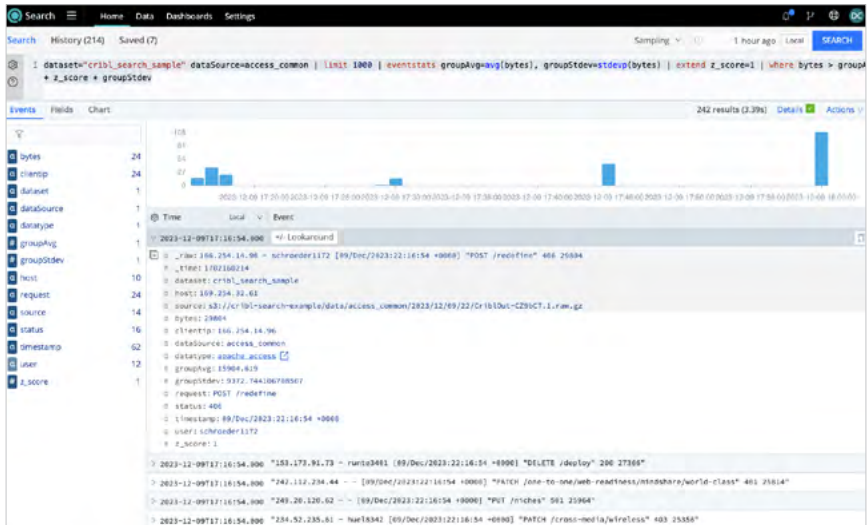
If we're curious about those values' distribution, Cribl Search enables us to easily take a closer look at the distribution's tails. To find outliers, we can create an additional metric using Search's aggregate standard-deviation function, `stddev`.

We're normally interested in 2 or 3 standard deviations away from the mean, or even 4 – well out on our assumed bell curve. Search enables us to specify the number of standard deviations as a Z-Score, and to tune the `z_score` value to set a threshold of interest for each dataset.

But when examining `bytes` across our `cribl_search_sample` dataset (which contains pseudo-randomly generated server and network logs), a Z-Score of 2 or higher won't find any anomalies. Here's how we can specify a `z_score` of `1`, to find values at least 1 standard deviation above the average:

```
dataset="cribl_search_sample" dataSource=access_common | limit 1000
| eventstats groupAvg=avg(bytes), groupStdev=stdevp(bytes) | extend z_
score=1 | where bytes > groupAvg + z_score * groupStdev
```

This analytic finds about a quarter of the limited events (242 out of 1,000).



About 25% of events are 1 to 2 standard deviations above the group average

CALCULATE A VALUE'S PERCENTAGE OF THE WHOLE

We often want to see a given subset's percentage of its parent dataset. Cribl Search (like other tools) can readily display this share visually – select Search's **Chart** tab for pie/donut charts or other graph formats.

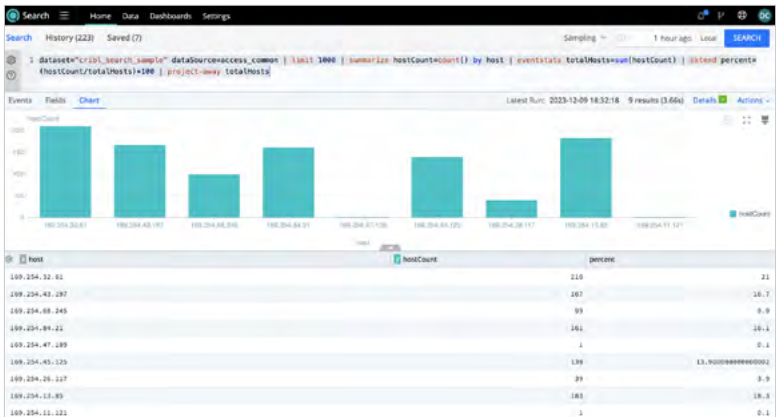
But we can calculate precise percentages by using both the `eventstats` and `summarize` operators together. Let's assume that we want to know individual IP addresses' percentages of the hosts found across all events.

We can use the `summarize` operator to aggregate counts across the `host` field to a `hostCount` field. Then we use `eventstats` to compute the total

number of hosts in the set, storing that to a `totalHosts` field. Dividing the two, and multiplying by 100, gives us each host's percentage of our total 1,000 events:

```
dataset="cribl_search_sample" dataSource=access_common | limit 1000
| summarize hostCount=count() by host | eventstats
totalHosts=sum(hostCount) | extend percent=(hostCount/totalHosts)*100
| project-away totalHosts
```

The **Chart** tab provides raw subtotals, percentages, and a bar graph:



Individual IP addresses' percentages of total hosts

From here, you could click the Format button (paintbrush icon) if you prefer to change the chart type to a donut graph or other visualization.

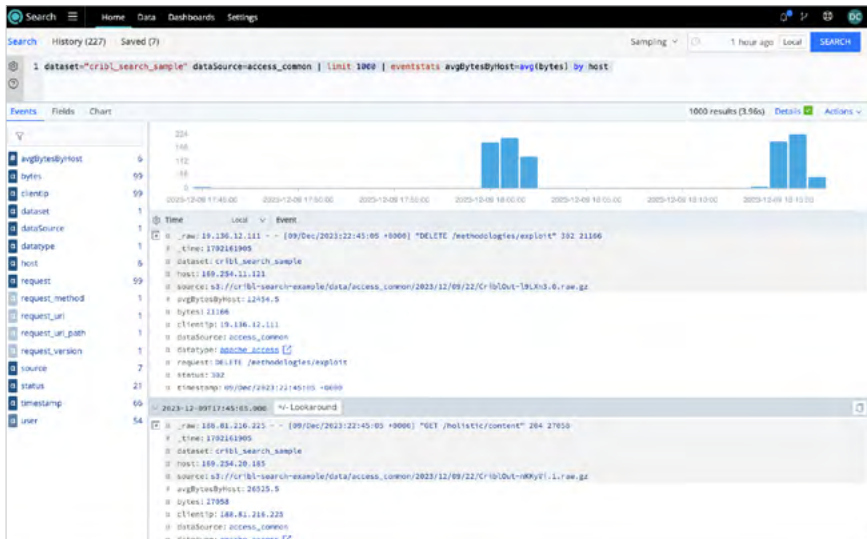
DISAGGREGATE BY VALUES, USING A GROUP-BY CLAUSE

So far, we've created fields that contain constant values across all events. But we can also use the `eventstats` operator's `group-by` clause to create different sets of statistics based on the field(s) in the group. These statistics might be different from event to event.

Below, we start with our original query for average bytes, but we append `by host` to calculate separate averages by each server host. This might be a more helpful metric for finding anomalies, given how different servers will likely serve pages with different statistical characteristics:

```
dataset="cribl_search_sample" dataSource=access_common | limit 1000
| eventstats avgBytesByHost=avg(bytes) by host
```

We've stored the result to a new field called `avgBytesByHost`, and each event is now enriched with this field. Its value represents the group average of bytes *across each event's respective host*:



Average bytes by host – this average varies by host IP

You can similarly apply a group-by clause to break down aggregate statistics based on other fields of interest in your dataset.

WHAT'S NEXT?

To more closely explore your datasets, Cribl Search's `eventstats` and `summarize` operators can each make use of a rich set of statistical functions, documented at docs.cribl.io/search/statistical-functions.

To get started with Cribl Search, sign up for a free Cribl.Cloud account at cribl.io/blastoff. Search will be ready to use immediately – start with its sample datasets or import your own working data.

For a guided tutorial on Search basics, take our free Cribl Search Sandbox for a spin at sandbox.cribl.io/course/overview-search.



9 | Refine Retrieved Data with Cribl Search

Minimize Cost and Latency
with Virtual Tables

by David Cavuto, Cribl

PROBLEM

Often, an IT or Security analyst must search through a given set of data many times to refine their question and analytics. Repeating the same search can incur slow response times and excess transport and infrastructure costs.

SOLUTION

Cribl Search supports searching into cached previous results – accelerating analysis for iterative search development.

WHAT WE'LL DO

We'll look at using Cribl Search's virtual tables to reload cached data from:

- Ad hoc searches.
- Saved or Scheduled searches.

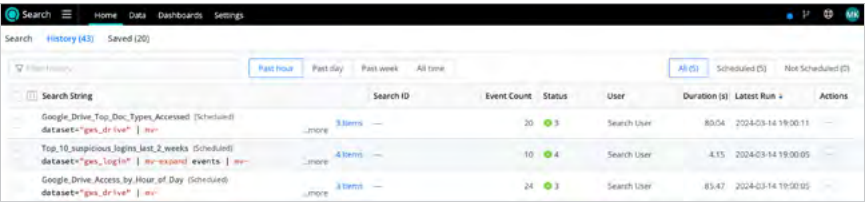
RECALL, REFINE, REPEAT

When you're iterating on a search, it's often useful to start with a simple dataset, and then add incremental Search operators to perform transformations, summarizations, arithmetical and statistical functions, and data enrichment. This avoids re-executing the full search, which would add latency as well as cost.

And Cribl Search makes this easy.

SEARCH REMEMBERS

All searches in Cribl Search generate result sets, which Search caches. You can click Search's **History** tab (docs.cribl.io/search/search-overview) to display previously executed Search jobs, by user.



The screenshot shows the 'History' tab in the Cribl Search interface. It displays a table of search jobs with columns: Search String, Search ID, Event Count, Status, User, Duration (s), Latest Run, and Actions. Three search jobs are listed, each with a 'more' link and a 'Items' count.

Search String	Search ID	Event Count	Status	User	Duration (s)	Latest Run	Actions
Google Drive Top Doc Types Accessed (Scheduled) dataset="gcs_drive" mv--	...	3 items	20	Search User	80.54	2024-03-14 19:00:11	...
Top 10 suspicious logins last 2 weeks (Scheduled) dataset="gcs_logins" mv--expand events mv--	...	4 items	10	Search User	4.15	2024-03-14 19:00:05	...
Google Drive Access by Hour of Day (Scheduled) dataset="gcs_drive" mv--	...	8 items	24	Search User	83.47	2024-03-14 19:00:02	...

Search History

From the History, you can access each ad hoc search's JobId, as we'll see below.

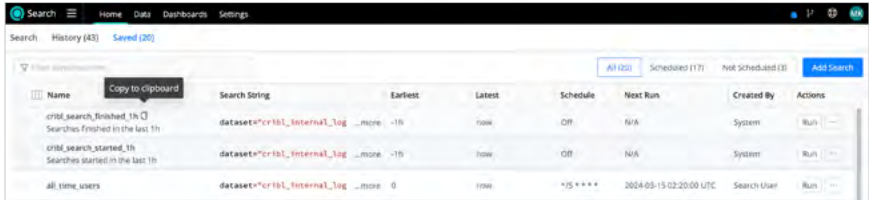


The screenshot shows the 'Details' page for a specific search job. The search job ID is 1710386026359.cbokty. The page has tabs for Details, Search Plan, Logs, and Metrics. A table at the bottom shows the key-value pair for the job ID.

KEY	VALUE
id	1710386026359.cbokty

Ad hoc search's JobId

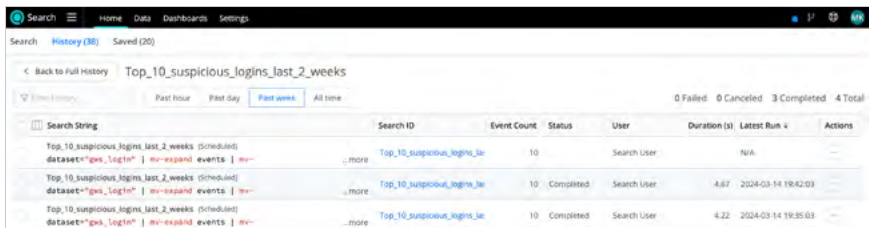
It's even easier to identify Saved and Scheduled Searches, because a **Search Name** is displayed as part of their configuration.



Name	Search String	Earliest	Latest	Schedule	Next Run	Created By	Actions
cribl_search_finished_1h Searches finished in the last 1h	dataset="cribl_internal_tag" ...more	-1h	now	Off	N/A	System	Run
cribl_search_started_1h Searches started in the last 1h	dataset="cribl_internal_tag" ...more	-1h	now	Off	N/A	System	Run
all_time_users	dataset="cribl_internal_tag" ...more	0	now	* * * * *	2024-05-15 02:20:00 UTC	Search User	Run

Saved Search > Search Name

When you click on any row of an executed search, you'll see the search's previous results. But Cribl Search does not re-execute these searches; instead, it displays the cached results of previous executions. These results render very quickly, because they've already been fetched from the original source, and any transformations are already applied.



Search String	Search ID	Event Count	Status	User	Duration (s)	Latest Run	Actions
Top_10_suspicious_logins_last_2_weeks (scheduled) dataset="ps_logins" mv-expand events mv- ...more	Top_10_suspicious_logins_la	10		Search User	N/A		
Top_10_suspicious_logins_last_2_weeks (scheduled) dataset="ps_logins" mv-expand events mv- ...more	Top_10_suspicious_logins_la	10	Completed	Search User	4.67	2024-03-14 19:42:03	
Top_10_suspicious_logins_last_2_weeks (scheduled) dataset="ps_logins" mv-expand events mv- ...more	Top_10_suspicious_logins_la	10	Completed	Search User	4.22	2024-03-14 19:35:03	

An ad hoc search's past results

RESTORING PAST SEARCHES: VIRTUAL TABLES

To restore previous Search results as a basis for new searches, you can specify a virtual table. Virtual tables access Cribl Search's internal metadata. Each virtual table begins with the `$vt_` convention, and each displays different types of metadata.

In Search queries, you can use virtual table names in place of dataset names. Use either of these query strings to see all the virtual tables accessible by your Cribl Search account:

- `$vt_list`
- `dataset=$vt_list`

THE \$VT_RESULTS VIRTUAL TABLE

A very useful virtual table that you might see after executing the above search is `$vt_results`. This virtual table requires a parameter of either `jobId` or `jobName`. As we mentioned above, you can use `jobId` to specify any ad hoc search, and can use `jobName` with Saved or Scheduled Searches. Running `$vt_results` without either of those parameters will display a reminder to add one.

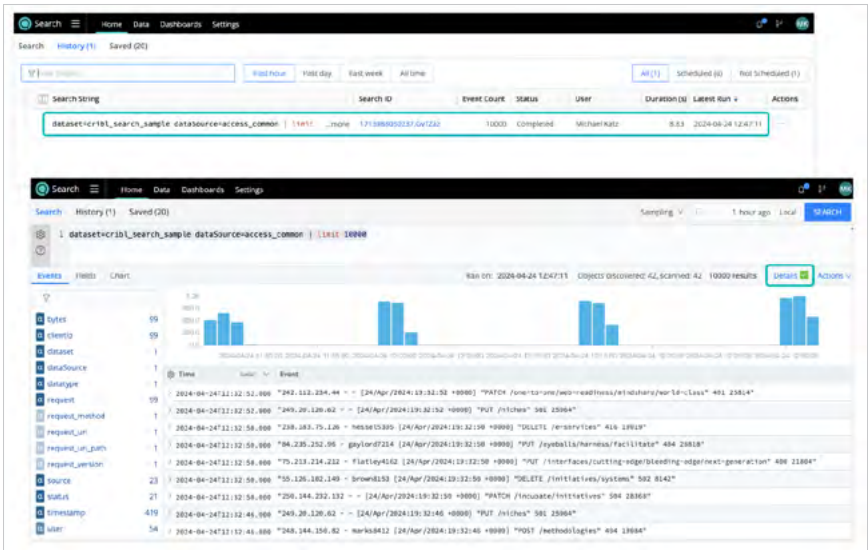
EXECUTING AND REPEATING A SIMPLE SEARCH

To experiment with `$vt_results`, we can run a simple search against Cribl Search's out-of-the-box `cribl_search_sample` dataset. Three types of `dataSource` are currently prepopulated in this dataset: `syslog`, `vpcflowlogs`, and web server `access_common` logs. For this example, we can choose any of those data types. So we could use the following query to get a quick sample of `access_common` log data:

```
dataset=cribl_search_sample dataSource=access_common | limit 10000
```

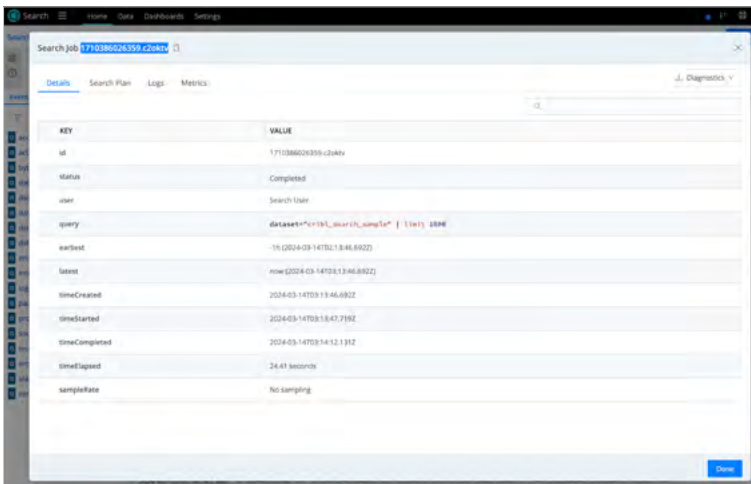
UNPACKING A SAMPLE

The above query should give us a large sample of those logs (10,000 events), which should take 10–20 seconds to capture and download. After the search completes, we can click Search's **History** tab, then click the search's top row to display the full result set, then click **Details** at the upper right.



Initial search's result set

From this resulting modal's upper left, beside **Search job**, we can copy the Job ID.

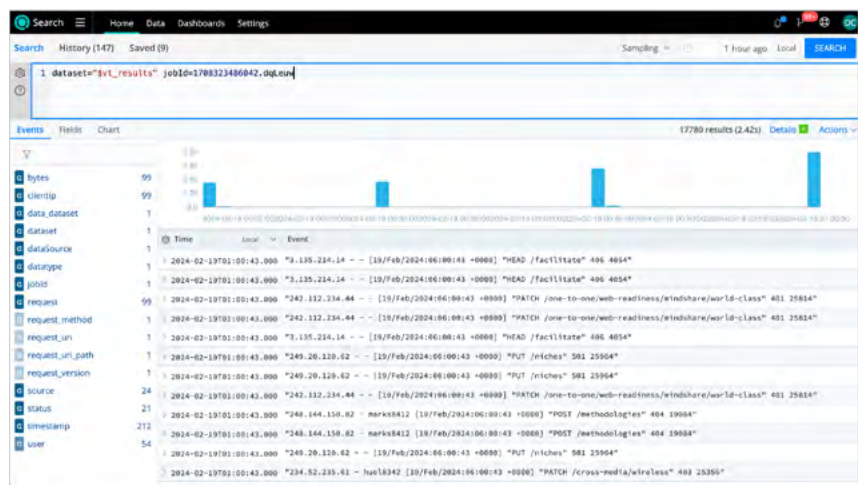


Getting the Job ID

RESTORING A SEARCH

Next, we can create a new search, specifying the `$vt_results` virtual table as the dataset, along with the Search Job ID we copied from the **Details** modal:

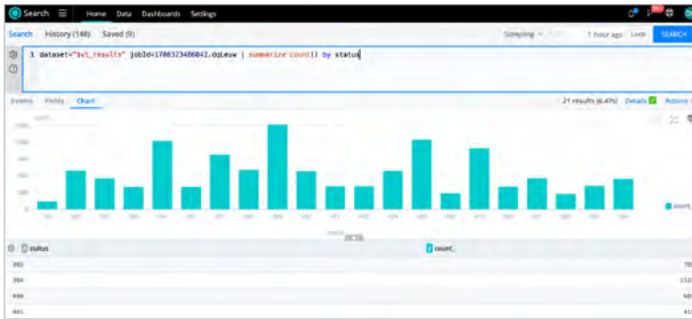
```
dataset=$vt_results jobId=<copied_job_id>
```



Restoring a sample with a Job ID

This cached search will immediately return the identical results from the prior search. But now we can iterate on the same result set by appending more operators to the query. For example, we might want to summarize by a certain field:

```
dataset=$vt_results jobId=<copied_job_id> | summarize count() by status
```

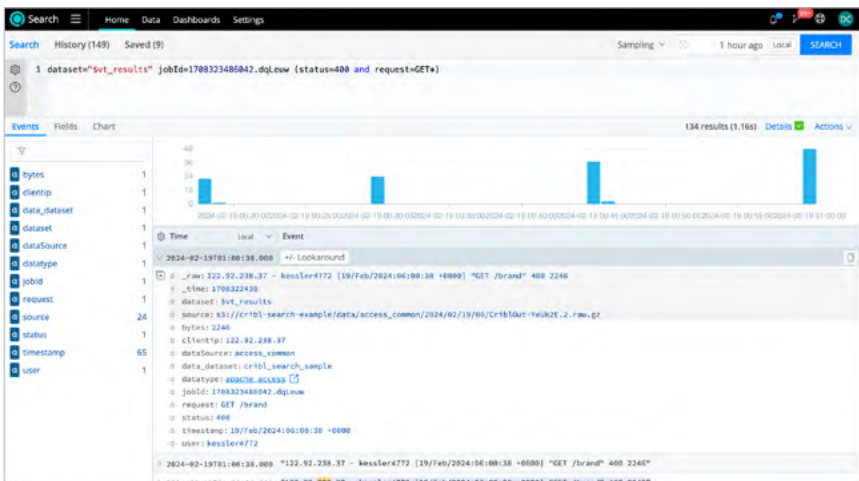
Iterating on a prior search

This search should also run almost instantaneously. (If you're used to Cribl Search's normal behavior, be aware that when using `$vt_results`, Search ignores the Time Picker at the upper right.)

FILTERING THE RAW EVENTS

We can examine some raw events by adding a filter that uses two of the field values from the previous searches:

`dataset=$vt_results jobId=1708317608676.YpHC8L (status=400 and request=GET*`



Virtual table's raw events

Again, you should see fast access to results. This allows you to experiment with your search, and to quickly iterate to the query expression that gives you your desired results. (Remember, however, that these results are all being computed over the single, cached result set, so they might or might not be representative of current-time data.)

REMOVING THE LIMIT

Once you've established that the virtual table search has retrieved the search and values you want, you can go back searching your original dataset. Substitute your desired parameters, remove any result limits, and – now that the Time Picker is again available – use a time range as your primary filter of these results.

Initially, it's always a good practice to try piping an unknown result set to `| count` to ensure that you won't return more results to your browser than you intend:

```
dataset=cribl_search_sample dataSource=access_common (status=400 and request=GET*) |count
```

After this returns how many events there will be, you can remove the `| count` operator:

```
dataset=cribl_search_sample dataSource=access_common (status=400 and request=GET*)
```

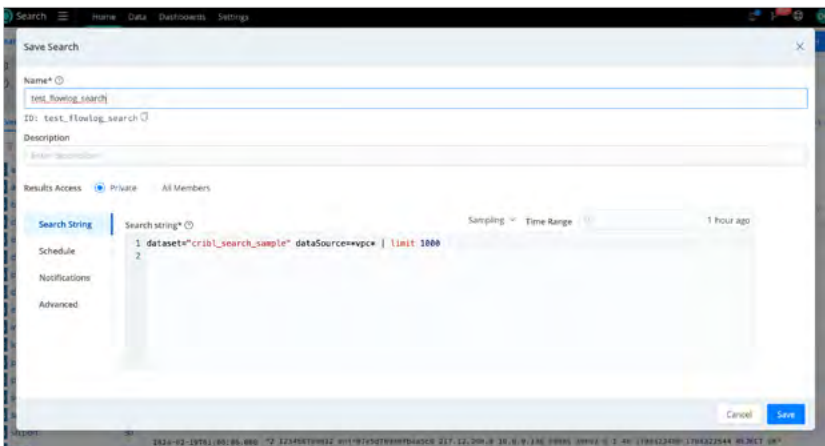
REPEATING SAVED OR SCHEDULED SEARCHES BY NAME

So far, we've seen how to reload previous ad hoc search results by their `$jobId` parameter. But for any Search job that you've explicitly saved, you can refer to it by its saved name.

As an example, we could create a new search, specifying the same sample dataset but a different `dataSource`, and save it after it executes:

```
dataset="cribl_search_sample" dataSource=*vpc* | limit 1000
```

After this executes, we'd open the **Actions** menu at Cribl Search's upper right, and select **Save Search**. Next, in the modal shown below, we'd enter a name (like `test_flowlog_sample`) and click **Save**.



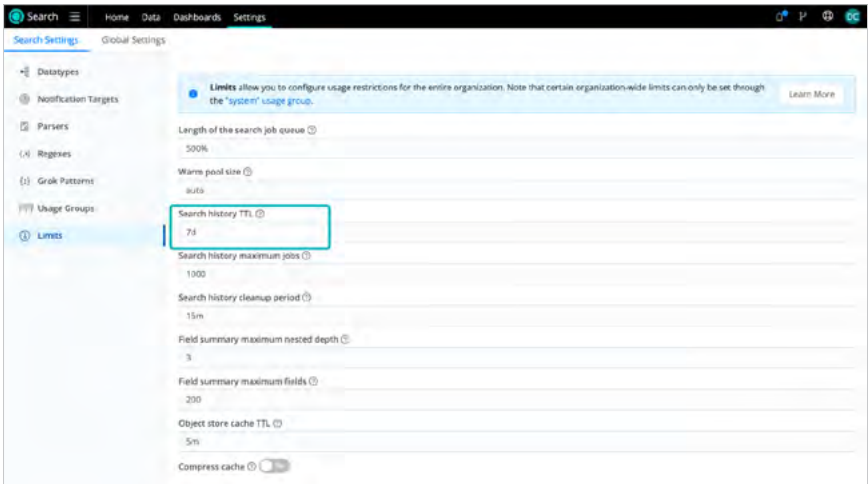
Saving a Search

Having executed this search, we can reload it by specifying `$vt_results`, this time with a `jobName` parameter, plus the name we just assigned. Note the quotes:

```
dataset=$vt_results jobName="test_flowlog_sample"
```

TROUBLESHOOTING NAMED SEARCHES

To reload a saved search by name, make sure that the search has been executed at least once, and that it hasn't been deleted by the automatic cleanup mechanism specified in **Settings > Limits > Search History TTL**.



Check this setting if you can't reload a previous search

Let's look at how to avoid worrying about that "executed at least once" prerequisite.

SCHEDULING TO ENSURE SEARCH HISTORY

If you toggle on your search's **Schedule** option, you can guarantee that it will have executed at least once. After scheduling, when you search for a scheduled search by name, you'll get the results from the most recent execution.

ACCELERATING DASHBOARDS

You can use named `$vt_results` virtual tables to run new ad hoc searches. You can also use virtual tables in Cribl Search dashboards, to make those dashboards execute faster. Dashboards' **Parent Search** option (docs.cribl.io/search/editing-dashboards) enables you to transparently use this technique within a Search panel, without having to explicitly specify `$vt_results` in your search.

JOINING VIRTUAL TABLES

One final note: You can combine the results of multiple previous searches, by specifying multiple `jobId` or `jobName` parameters around a boolean `or` term. For example, you could specify:

```
dataset=$vt_results (jobId=<job_id_1> or jobId=<job_id_2>)
```

This syntax will essentially create the union of both result sets, enabling you to search into the combined results.

WHAT'S NEXT?

To get started with Cribl Search, sign up for a free Cribl.Cloud account at cribl.io/blastoff. Search will be ready to use immediately – start with its sample datasets or import your own working data.

For a guided tutorial on Search basics, take our free **Cribl Search** Sandbox for a spin at sandbox.cribl.io/course/overview-search.

Find the following details in Cribl Search's online documentation:

- History tab and Saved Searches (docs.cribl.io/search/search-overview).
- Scheduled Searches (docs.cribl.io/search/scheduled-searches).
- Reusing search results (docs.cribl.io/search/common-examples/#reuse).
- Virtual Tables (docs.cribl.io/search/virtual-tables).
- Dashboards with Parent Search (docs.cribl.io/search/editing-dashboards).
- Operators (docs.cribl.io/search/operators).



10 | Cribl, AI, and You

by Nikhil Mungel and Anna Casey, Cribl

PROBLEM

As the volume and complexity of IT and security data continue their rapid ascent, the practitioners who need to make sense of all this data face constraints on budgets, resources, staffing, and knowledge. Cloud and hybrid infrastructure is distributed, diverse, difficult to secure, and often so complex that it's difficult for human beings to design and monitor.

Specialized skills are required to architect this infrastructure. To write robust regex expressions, JavaScript expressions, and search queries. To design efficient, complex processing pipelines and routing. To troubleshoot code and commit histories. And even to make the most effective use of data-visualization options and results. Organizations can't find enough of these highly skilled people, so the practitioners on the job are stretched thin.

SOLUTION

Cribl is adding powerful Artificial Intelligence capabilities throughout our product suite. These tools diversify who can use Cribl apps, easing skills bottlenecks. They help all practitioners design infrastructure – and gain accurate, actionable insights into their IT and/or Security data – at higher velocity.

We outline individual initiatives in the sections that follow. You will see these and other AI features progressively emerge across Cribl's product suite. We'll start with a day-in-the-life illustration.

SCENARIO

Cribl's approach is to make AI work for you, not the other way around. We focus on bringing in the technology to assist you at critical moments in your workflow, with Cribl's AI, known as Cribl Copilot, shouldering some of the load to help you work faster and more effectively.

Imagine that you're just starting out with Cribl Stream. You need to onboard a new data Source and send it to your SIEM (security information and event management) system, while also sending a full-fidelity copy to Cribl Lake or to other cost-effective, long-term storage.

You ask Cribl Copilot, for step-by-step instructions for how to set up your new Source and how to route data to more than one Destination. Cribl Copilot synthesizes the relevant information from a variety of documentation topics and other resources, and presents you with a clear procedure.

Your configured Source initially looks fine, but you start to see a recurring error message from Stream. You ask Cribl Copilot about the error, and the chatbot helps you identify a problem with your auth token on the Source side. With this guidance, you can resolve the problem without sifting through error logs or asking someone for help.

With your data now flowing, you want to transform and reduce the data sent to your SIEM, knowing that the complete original data will be preserved in cold storage. You know what you want your logs to look like after they're processed by the processing Pipeline, but you aren't sure which Stream Functions will get you there, and regex isn't one of your strengths.

Using Copilot's natural language interface, you describe your data's ideal end state, and it then assembles the Pipeline Functions for you and shows you the results.

Now that you've routed everything – sending reduced logs to your SIEM, and full-fidelity logs to cold storage – you need to show that you can still access those full-fidelity logs for regulatory compliance.

You open up Cribl Search and see that in addition to its native KQL (Kusto Query Language), Copilot offers you a natural language interface to build queries. You get results quickly and learn more about KQL along the way.

Finally, to help you better visualize your data, Cribl Search's AI-assisted dashboard capabilities evaluate that data and recommend visualization types. Search also offers insights and analytics on the displayed data, to help you understand patterns and potential outliers, refine your insights, and plan your next steps.

In this chapter's remaining sections, we outline some of Cribl Copilot's specific AI initiatives by category and use case.

IN-PRODUCT CHATBOT POWERED BY COPILOT

These in-app chat features are accelerators for users of all Cribl products.

Answer Product Questions

Provides answers about how to use and administer Cribl products. Offers inline help on how to complete common tasks, configure advanced settings, and troubleshoot problems, all without leaving the product.

Regex and JS Expression Generation

Automatically generates regular expressions and JavaScript expressions from user-provided descriptions. This streamlines the process of writing complex code snippets – ensuring accuracy and saving time in developing search patterns and scripts.

Error and Git Commit Summarization

AI-driven summarization techniques distill error logs, and git commit histories, into succinct, informative overviews. This facilitates faster diagnosis of issues and efficient tracking of code changes.

NL ↔ KQL INTERFACE

These natural-language and code-based features are Cribl Search additions to accelerate query building.

Natural Language to KQL

Converts users' natural-language input to Kusto Query Language, by leveraging LLMs (large language models). This enables users to query data without needing extensive knowledge of KQL syntax. So data querying and analysis become more accessible. Expect to see support extend to other input types, like SQL.

KQL Optimization

Enhances the performance and accuracy of KQL queries, using advanced optimization algorithms. This feature is vital for handling large datasets and complex query scenarios, ensuring fast response times and high-quality data retrieval.

Prompt-Based KQL Generation

Uses AI to interpret user prompts, control-plane data, and event data to automatically generate relevant KQL queries. This facilitates rapid query formulation – especially helpful to users who might not be proficient in KQL, but who are familiar with the data schema or domain context.

KQL Typeahead Autocomplete

Incorporates predictive text input behavior, akin to other copilot-like coding assistants, for crafting efficient KQL queries. This aids power users in query writing by suggesting relevant keywords and structures, enhancing speed and accuracy of query building.

NL-DRIVEN PIPELINE MANAGEMENT

These natural-language interfaces help users build processing Pipelines faster and more accurately.

Pipeline Explanation and Setup

Translates natural language instructions into specific Pipeline configurations. This includes setting up data Sources, defining processing steps, and configuring Destinations – in all, making Pipeline creation and modification more accessible. Users can hand-edit events to indicate how they should be transformed, and use this feature to automatically generate the Pipeline.

Pipeline Optimization

Applies internal knowledge base to optimize data flow through Pipelines, improving throughput and efficiency. This feature is essential for managing large-scale data processing – reducing bottlenecks and backpressure, and ensuring optimal resource utilization.

NL INTERFACES FOR DASHBOARDS

These natural-language interfaces are Cribl Search additions to accelerate and improve users' dashboard construction and analysis.

Insight Generation from Visualizations

Analyzes and interprets dashboard data to extract actionable insights, using advanced data analysis algorithms. This feature allows users to quickly understand complex data visualizations and derive meaningful conclusions, supporting informed decision-making. It informs you when insights about your data change.

Visualization Recommendation Engine

Leverages an AI model to suggest dashboard panels and layouts based on the underlying data analysis. This tool is instrumental in customizing dashboards to highlight key metrics and trends, tailored to a user's specific data context.

Dashboard Construction via NL

Enables users to construct and customize dashboards using natural language. This simplifies dashboard creation, allowing users to specify data sources, visualization types, and layout preferences in a no-code manner.

TROUBLESHOOTING

Just don't prompt this (or any) AI to optimize for the production of paperclips, and everything should be fine.

WHAT'S NEXT?

For applications of these features, please see our online documentation:

- Cribl Copilot (docs.cribl.io/suite/cribl-copilot).
- Cribl Search (docs.cribl.io/search).
- Cribl Stream (docs.cribl.io/stream).
- Cribl Edge (docs.cribl.io/edge).
- Cribl Lake (docs.cribl.io/lake).

