ENRICHMENT: BETTER DATA IN → BETTER RESPONSE TIMES OUT

Shawn Cannon
Threat Management Consultant/
Splunk Engineer, Aflac

Carley Rosato
Staff Solutions Engineer,
Cribl

SHAWN CANNON

Threat Management Consultant/ Splunk Engineer, Aflac

Shawn has more than 26 years of IT experience working in systems administration, client hardware implementations, managed security services and big data.

His current focus is managing the SIEM and AWS environment for the SIEM, working to bring in new data as needed and improving on the existing data ingestion process.

Outside of work, Shawn enjoys bowling, watching movies, and attending sporting events (Atlanta Braves, Alabama Crimson Tide).

Fun fact: He loves wearing crazy, unique socks! He has over 50 different pairs of socks which consist of movies, games, sports teams, and pop culture. If you see him the rest of the week, you will see some of these socks!

CRYPTOCON! 2023

Aflac

CARLEY ROSATO

Staff Solutions Engineer,
Cribl

Carley has spent the entirety of her career in Technology, focusing her efforts on optimizing security and operational data. She draws on her past experience with complex logging platforms to support organizations in reaching their business goals.

As a Solutions Engineer at Cribl, **Carley works alongside customers to design, build, and deploy** modernized data solutions.

When “out of office,” you will most likely find her hiking or biking.

CRIBLCON! 2023

Aflac

WHY CRIBL STREAM?

Cribl Stream replaced antiquated syslog solution—**huge win!**

Over 2 years, we...

Replaced:

- HTTP Event Collector for AWS sources
- Script collectors (moved from Splunk HFs)
- REST collectors

Onboarded & optimized:

- Crowdstrike FDR
- Multiple S3 sources (Cloudtrail, Cloudfront, Incapsula/Imperva)

Fixed mixed data:

- Applied pipelines and functions to fix mixed data that had JSON and non-JSON
- Joined JSON key/value pairs into searchable fields before sending to Splunk

And many more!

CHALLENGE

- **34 million row Lookup file** to enrich data in Splunk
- **Cribl + MongoDB + 3rd-party solution** becoming too costly
- Needed new, affordable solution **fast**!

ENRICHING DATA WITH CRIBL

Lookup Options in Cribl Stream		
	CSV	**Redis**
Deployment Options	On leader and pushed to workers	Standalone, Clustered, Sentinel
Number of Rows	<1 to 5 million	>5 million
Cribl Resource Requirements	More Memory required	Less Memory
Network Requirements	Not required as it is deployed on Worker Infrastructure	Ideally, same Subnet as Workers to minimize latency
Frequency of Data Changes	Rarely needs updating	Regular Updates and Changes
Updates to the Lookup Data	Manual of via Script	Apply with Cribl Built-in Functions
Ideal Use Cases	GeoIP, smaller environment asset tagging, HTTP method description, network protocol (DNS vs. http)	Threat Intel [enriching from multiple feeds], large environment asset tagging

SOLUTION

Cribl Stream + Redis Function FTW!

1. Created Redis Elasticache in AWS
2. Populate lookup file
3. Add Redis function in Cribl to add needed fields to the data

REDIS CONFIGURATION

To create the Redis cache in AWS, determine the needed instance size. Redis docs say:

- 1 million small keys → String Value pairs use **~85MB of memory**.
- So, 34 million small keys → String pairs would take approximately **2.8GB of memory**.
- I was able to use the smallest Redis Elasticache instance size of cache.r6g.large, which is **2 CPUs** and **13.07GB of memory**.
- As I have the CSV file as a backup, I was not concerned with setting up a Redis cluster, so I opted to not use clustering. I went with 1 shard and 3 nodes (1 node being primary and the other 2 as replicas).

GETTING THE CSV FILE INTO REDIS

1. CSV file is created every morning.
2. Scheduled Cron job runs to ingest the CSV into Redis.

Here is an example of the command using my test system:

```
awk -v rediskey=demo -v uidcolumn=1 -f /tmp/csv2redis.awk /tmp/lookup.csv |
redis-cli localhost:6379 –pipe
```

```
"ACCT_NUMBER","CUSTOMER_ID"
X123456,111111111
Y111111,222222222
3456789,333333333
9876543,444444444
B135790,555555555
B111111,666666666
C111111,777777777
D111111,888888888
E111111,999999999
F111111,101010101
```

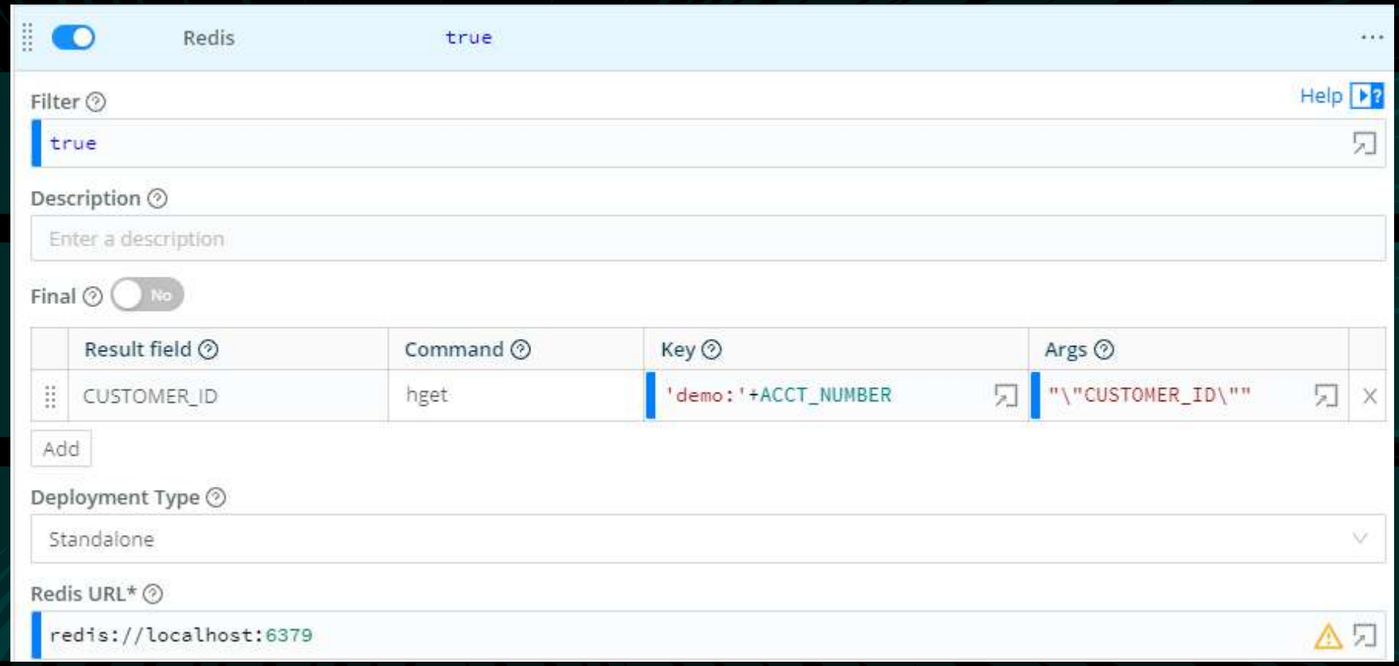
The process:

- awk command reads the csv file line by line using this script.
- Use demo as the Redis key and Column 1 in the CSV as the uid
- As each line is read, it is sent to Redis by piping to redis-cli.
- Each record gets stored in Redis in the format “demo:ACCT_NUMBER”

Can share the csv2redis.awk file I found online.

USING REDIS FUNCTION IN CRIBL STREAM

- Redis instance populated.
- Data needing enrichment (CUSTOMER_ID field) passes thru pipeline with Redis function.
- Matches ACCT_NUMBER field to the key in Redis and returns the CUSTOMER_ID field.



BEFORE REDIS

Data coming into Cribl Stream **before** the Redis function is applied.

```
a _raw: 2023-05-24 10:39:12.949, ACCT_NUMBER="X123456", NAME="JOHN DOE"
# _time: 1684939152.949
a host: host
a index: temp
a source: DATABASE
a sourcetype: TEST
```

```
a _raw: 2023-05-24 10:39:12.949, ACCT_NUMBER="Y111111", NAME="JANE DOE"
# _time: 1684939152.949
a host: host
a index: temp
a source: DATABASE
a sourcetype: TEST
```

AFTER REDIS

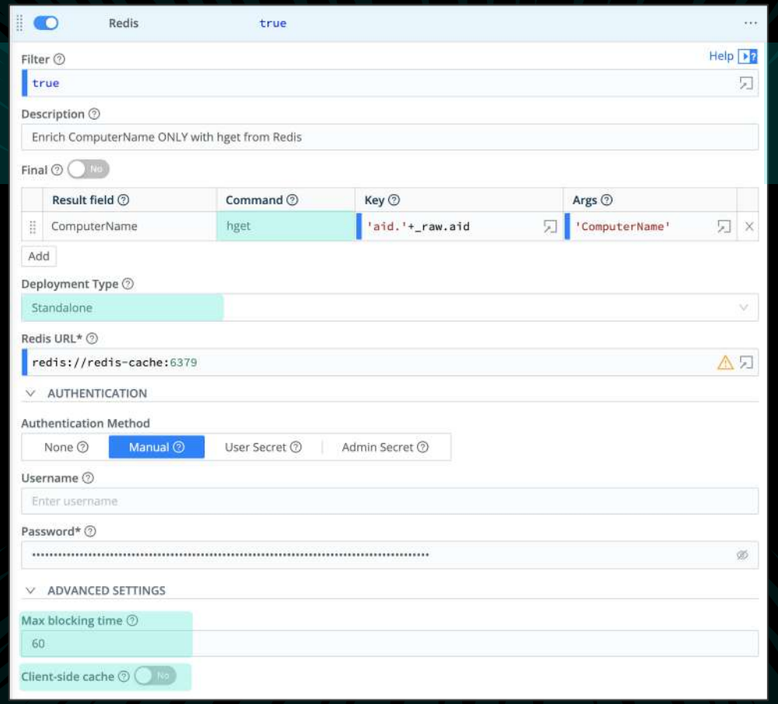
Data coming into Cribl Stream **after** the Redis function is applied.

```
a _raw: 2023-05-24 10:40:12.346, ACCT_NUMBER="C111111", NAME="STEVEN STRANGE"
# _time: 1684939212.346
a ACCT_NUMBER: C111111
a cribl_pipe: demo-redis
a CUSTOMER_ID: 777777777
a host: host
a index: temp
a NAME: STEVEN STRANGE
a source: DATABASE
a sourcetype: TEST
a _raw: 2023-05-24 10:40:12.346, ACCT_NUMBER="D111111", NAME="LUKE SKYWALKER"
# _time: 1684939212.346
a ACCT_NUMBER: D111111
a cribl_pipe: demo-redis
a CUSTOMER_ID: 888888888
a host: host
a index: temp
a NAME: LUKE SKYWALKER
a source: DATABASE
a sourcetype: TEST
```

ADVANCED DEPLOYMENT TOPICS

Deployment Type

- Redis can be **deployed** in Standalone, Clustered or Sentinel mode. All three are supported in Stream.
- Consider latency in retrieving the data depending on the **location** where is it deployed.



Advanced Settings

- **Basic Commands** can be called from the dropdown, but additional commands can be manually entered.
- **Caching** helps to minimize the number of stale keys and amount of time they are in your cache.
- **Timeouts** are used to determine if Redis is unavailable and events should continue to be passed downstream.

REDIS RESOURCES

Redis Pack

In this pack, you'll find **template policies** to incorporate within your existing pipelines. The scenarios covered include aggregation, sampling, correlation, enrichment and quota enforcement.

Visit us at the **CooLab** to preview solutions for popular use cases.

- Redis How-To Video.
- Managing Large Lookup Tables with Redis.
- Enrichment at Scale Blog.
- Large Lookups with Redis Blog Part II.
- Redis Function Configurations.


CRIBLCON!
2023