Scalable and Cost-effective Log Analytics Solution at Fourkites

June, 2024

FOURKITES

Intro

Arpit Garg

SENIOR PRINCIPAL ARCHITECT

- Have experience in building distributed platforms.
- Have worked on large sized DWH on oracle (@amazon) &
  Hadoop + Hive + Presto (@Uber).
- Seen the DB/data world since - waiting for a slot/queue from DBA's/Operations Team was something normal.
- Enjoy working with SQL.
- Based in India
- arpit.garg@fourkites.com
- Linkedin - https://www.linkedin.com/in/arpit-garg-38039020/

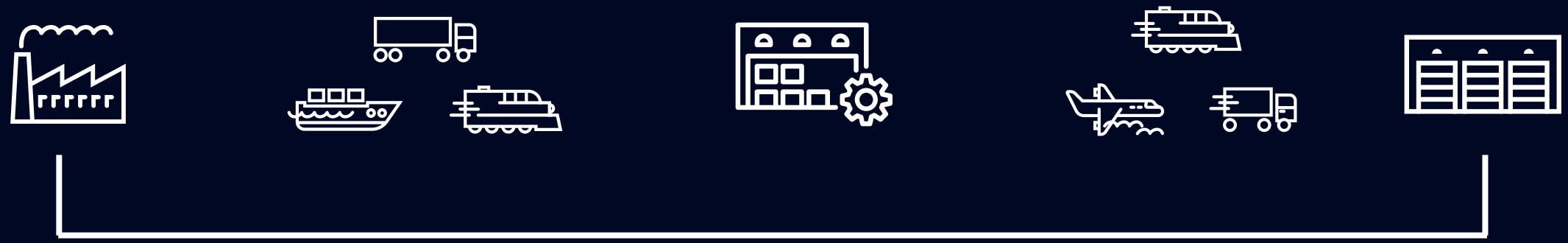
FOURKITES

FourKites Helps Your Supply Chain Run Better with Visibility Everywhere

Inbound
Visibility from Suppliers

Facilities
And Warehouse Visibility

Transportation
Visibility to B2B/Stores/B2C



End-to-end
Order and Inventory Visibility

FOURKITES

FourKites - The world's largest network of real-time supply chain data

45,000
TL Carriers

90%
of all global ocean volume

100%
of terminals in NA

17,000
airports and 87 Air Centers

96%
of LTL in US

3,500
Telematics Integrations

100%
rail coverage in NA, Europe and ANZ

16 Million
Locations (geofenced destinations)

3 Million
Shipments Tracking Daily 18M with FedEx

950,000
Mobile App Downloads

FOURKITES

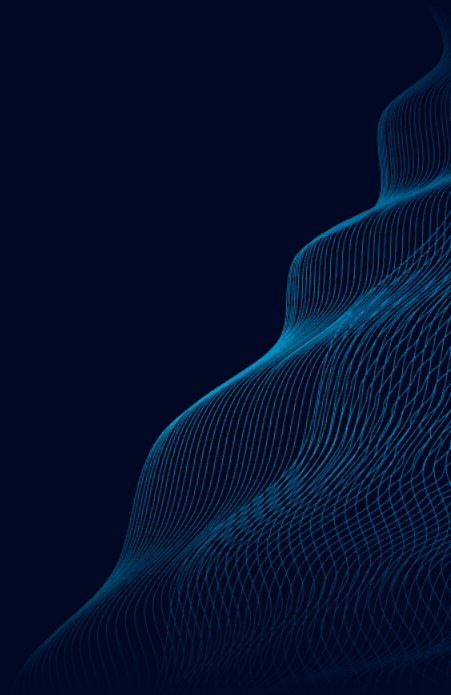
Motivation

- We love  producing logs (unfortunate but true).
- 120+ microservices/workers produce **7-8TB of logs daily.**
  - **~1 PB in 150 days**
- Operations & Support team needs logs with 120+ days retention for debugging and troubleshooting tickets.
- Using 3rd party log solutions like Logentries, NewRelic is expensive ~ $500K with limited retention of logs.
  - Difficult to join logs in these systems with our data stores.

FOURKITES

Approach

- Open source solution backed by lower cost storage like object store + allow querying of data by other engines.
- Ability to provide 120+ days data retention
- Ability to query logs and join it with data in other internal systems to make troubleshooting faster
- Reduce dependency on 3rd party log solutions



FOURKITES

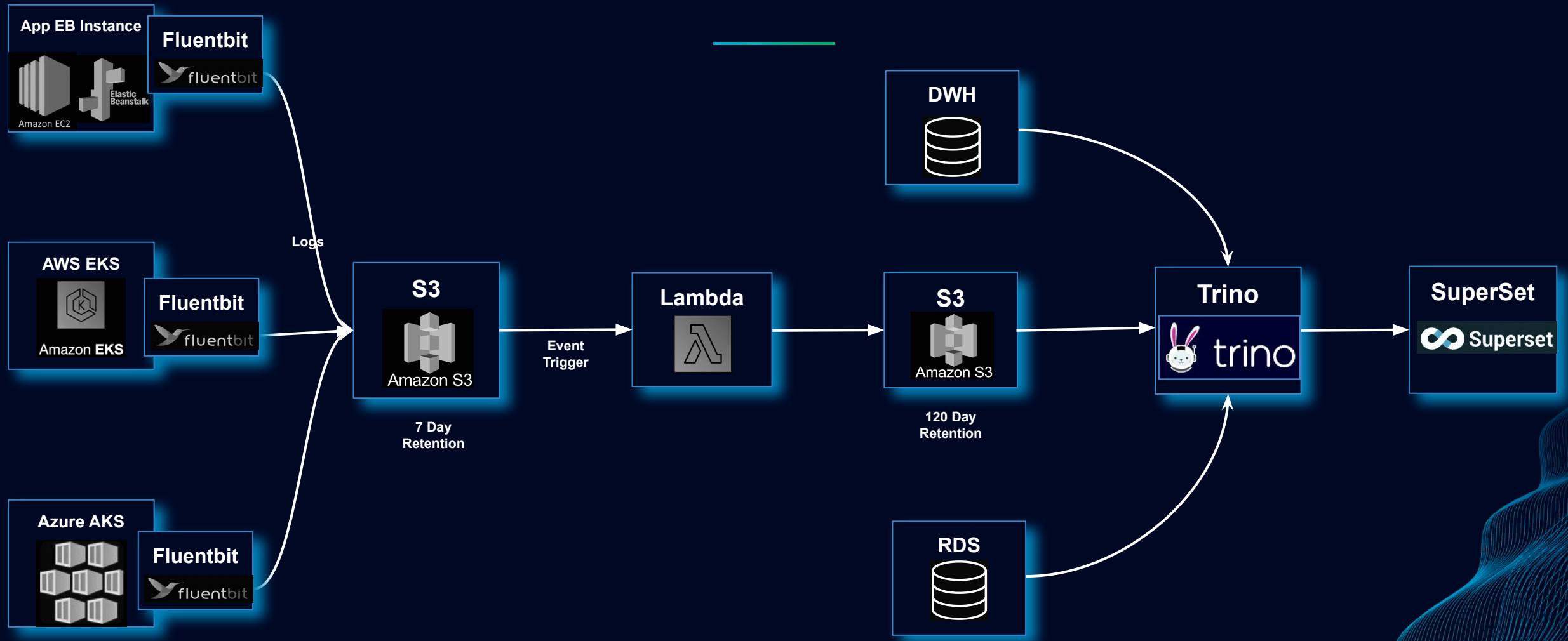
Solution - SPOG

- **S**ingle **P**ane **O**f **G**lass
- It is not only the log-analytics tool
- It is a super tool which can allow you to access any data in fourkites if it is in S3 or via direct connection to different databases
  - Currently live for logs, DWH (Redshift + Snowflake) & also metadata db of Trino+superset
  - Supports dashboard creation - some dashboards are Jira bug dashboard, spog usage metrics and aws+azure cost data
  - Alerts creation based on patterns in logs, cost alerts etc.

SPOG Architecture

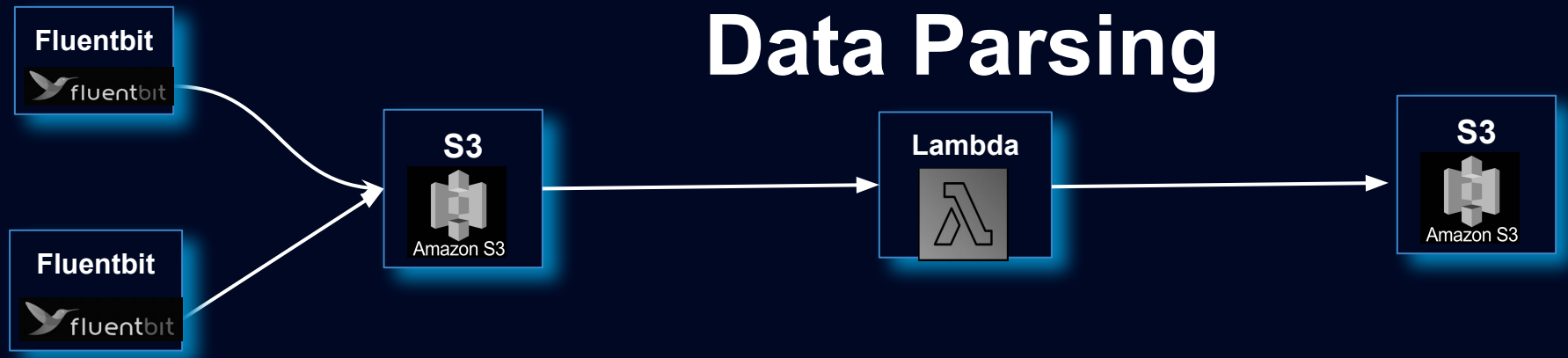
- Key components
  - Fluent-bit to ingest data
    - We use aws EB (Elastic beanstalk)
    - Aws EKS (Elastic Kubernetes Service)
    - Azure AKS(Azure kubernetes service)
  - Lambda - Transformer - for data transformation & field extraction
  - Objectstore - S3 as storage layer
    - parquet format
  - Trino as distributed query engine
  - Superset as BI/Dashboard/SQL-query tool
  - Jenkins for deployment of all components

Architecture



FOURKITES

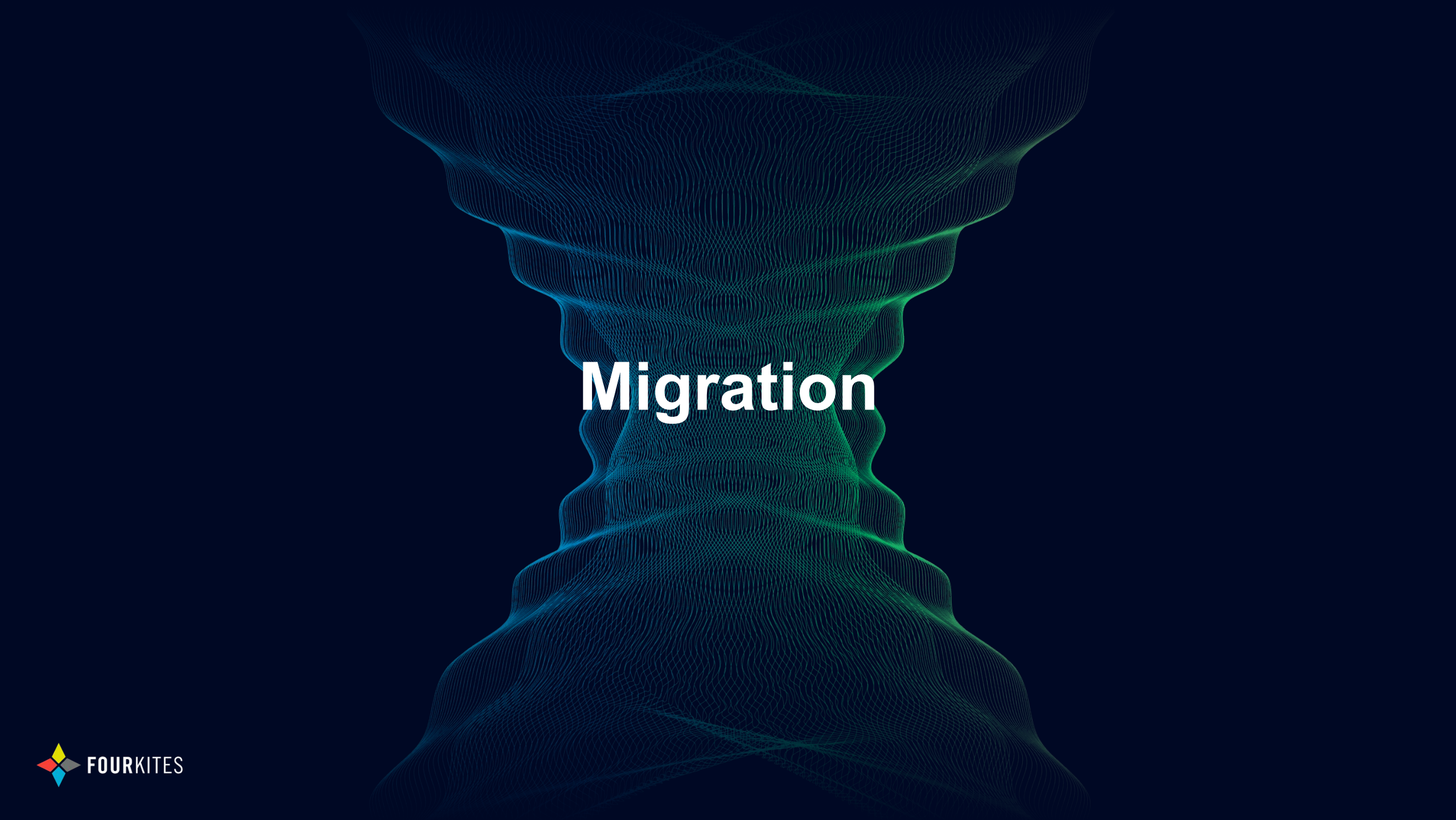
Data Parsing




```
2024-04-20 02:00:33.760, INFO - [f3456bf64836] PRMS: {"MessageType"=>"FETCH_ELD_LOCATION",
"EldParams"=>{"load_reference_key"=>"565656565", "shipper_id"=>"acme-inc",
"carrier_id"=>"carrier-for-acme-inc", "tracking_id"=>4909090909090,
"is_tracked_by_carrier"=>true, "is_tracked_by_shipper"=>false}, "MessageSource"=>"AWS",
"sqs_publisher"=>"global-worker-tracking", "sender"=>"global-worker-tracking",
"timestamp"=>1713578428981, "sqs_message_id"=>"xxxxxxxxx-03f9-4015-8043-yyyyyyyyy",
"sent_timestamp"=>1713578433730}
```

correlation_id	message_type	shipper_id	carrier_id	load_number	tracking_id	sqs_publisher
f3456bf64836	FETCH_ELD_LOCATION	acme-inc	carrier-for-acme-inc	565656565	4909090909090	global-worker-tracking

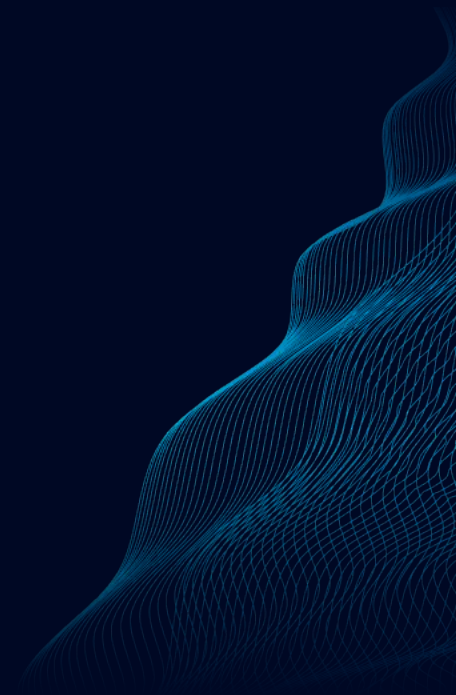
FOURKITES



FOURKITES

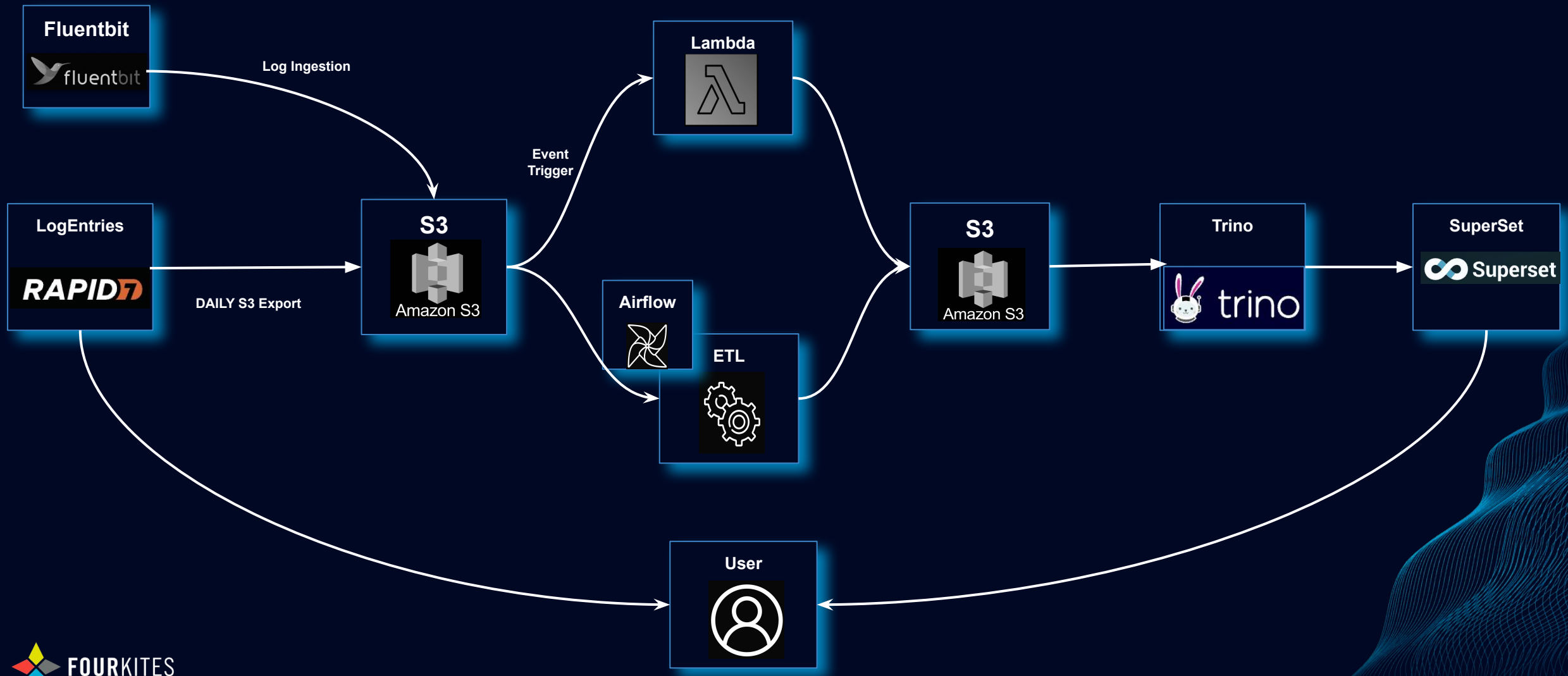
Migration to SPOG

- Logentries - logs
  - Before spog logentries (Rapid7) was our centralized logging solution
  - Once spog was launched we started pushing logs in parallel to both systems
  - Historical data was moved from logentries daily S3 export to spog via airflow pipelines



FOURKITES

Migration + Parallel Run



FOURKITES

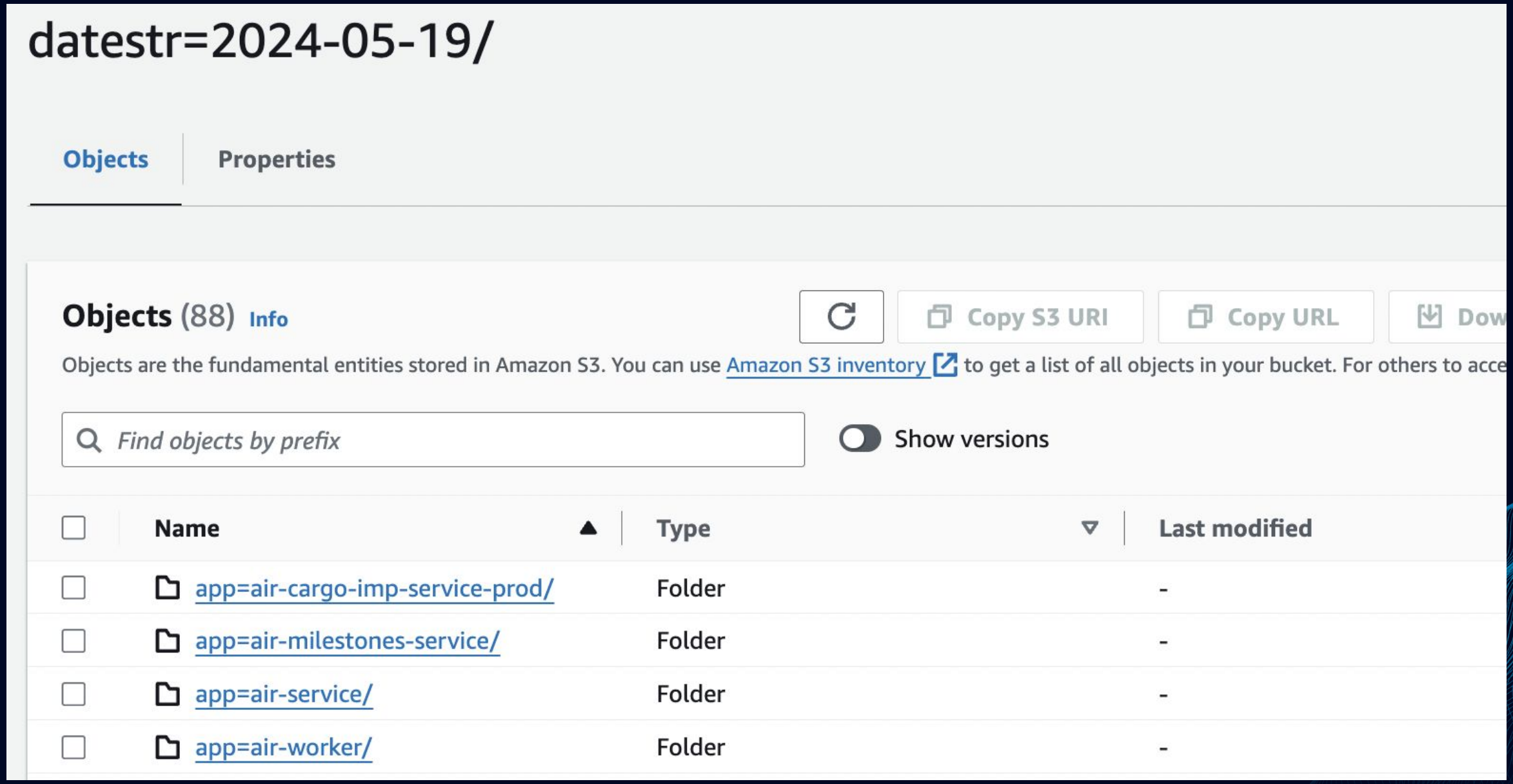
Trino Configuration

FOURKITES

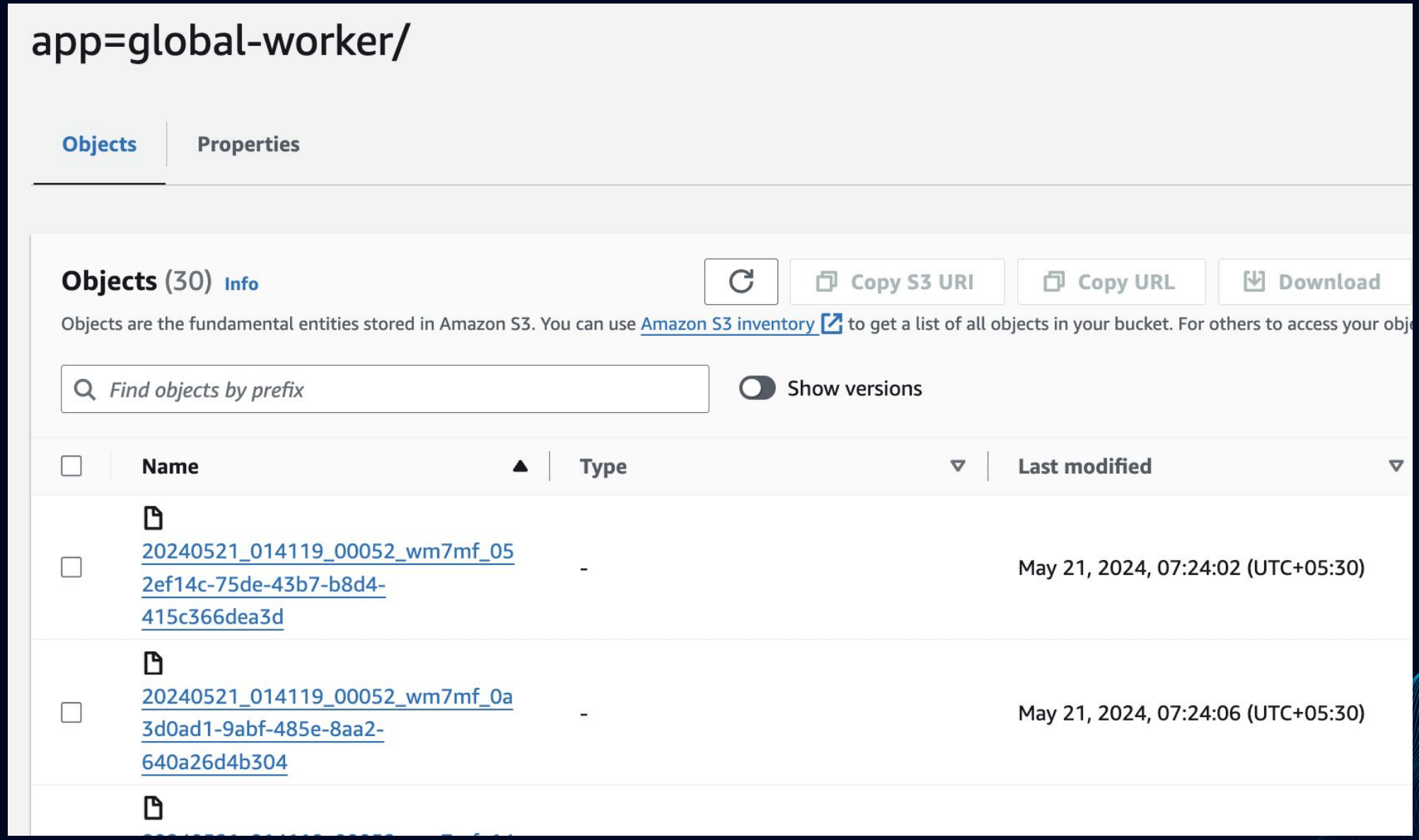
Trino table partition

- One main table - all_logs
- partition strategy source/datestr/app
  - s3://bucket/log-management/source=aws/datestr=2024-06-13/app=cfw/
  - s3://bucket/log-management/source=ebs/datestr=2024-06-13/app=iw/
- Partition refresh
  - Call system.sync_partition_metadata() & system.register_partition() for partition refresh.
  - Use airflow pipeline to refresh partitions by polling s3 bucket for new folders and calling register_partition()

partition - s3 path



s3 path


app=global-worker/
Objects
Properties
Objects (30) Info
Copy S3 URI
Copy URL
Download
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your obj
Find objects by prefix
Show versions
Name
Type
Last modified
20240521_014119_00052_wm7mf_052ef14c-75de-43b7-b8d4-415c366dea3d
-
May 21, 2024, 07:24:02 (UTC+05:30)
20240521_014119_00052_wm7mf_0a3d0ad1-9abf-485e-8aa2-640a26d4b304
-
May 21, 2024, 07:24:06 (UTC+05:30)

Trino - RLS

- Serving logs to our partners (limited access to data)
  - We have our partners who also want to access logs
  - They are supposed to have access to logs for customers they are managing.
  - We introduced a flag in each log row to check for the customer identifier and flag that row as visible/invisible
  - Created views on top of base table

Trino Auth

- configFile: "rules.json" option for roles/users
- Catalog + Schema + table level privileges

```
{
    "user": "trino_read_write",
    "schema": "log",
    "owner": true
  },
  {
    "user": "vcat_user",
    "schema": "vcat",
    "owner": true
  },
  {
    "user": "core_user",
    "schema": "core",
    "owner": true
  },
```

FOURKITES

Trino Auth

```
"tables": [
  {
    "user": "trino_admin",
    "catalog": ".*",
    "schema": ".*",
    "table": ".*",
    "privileges": ["SELECT",
"INSERT", "DELETE",
"GRANT_SELECT", "OWNERSHIP"]
  }, {
    "user": "trino_read_write",
    "catalog": "hive",
    "schema": ".*",
    "table": ".*",
    "privileges": ["SELECT",
"INSERT", "DELETE",
"GRANT_SELECT", "OWNERSHIP"]
  },
```

```
"tables": [
  {
    "user": "trino_read_only",
    "catalog": ".*",
    "schema": ".*",
    "table": ".*",
    "privileges": ["SELECT"]
  },
  {
    "user": "vcat_user",
    "catalog": "hive",
    "schema": "vcat",
    "table": ".*",
    "privileges": ["SELECT",
"INSERT", "DELETE",
"GRANT_SELECT", "OWNERSHIP"]
  },
```

FOURKITES

Trino User Authentication

- We use http-server.authentication.type=PASSWORD

```
auth:
  passwordAuth: |-
    trino_read_write:xxxxxx
    trino_read_only:xxxzzzzz
```

Reference - https://trino.io/docs/current/security/password-file.html#security-password-file--page-root

FOURKITES

FOURKITES

Trino Optimization

- Performance test for trino for concurrency and large queries
  - Adopted memory intensive r5.2xLarge machines
- Trino do not like large number of small files & fluent-bit pushes us multiple small files depending on frequency of file cutover.
  - Compaction to rescue
  - Daily compaction jobs to merge files - this is critical for us and we make sure we leave data consistent
- Timeout for long running queries
  - Running risk of someone running bad SQL
- Auto Scaling for trino pods
  - Cost saving
  - possibility of reducing number of trino pods on weekends.
- Aggregate/subset tables for app logs which are large in size
  - Reduced query time and improved response time

Optimizing Lambda - Transformation Layer

FOURKITES

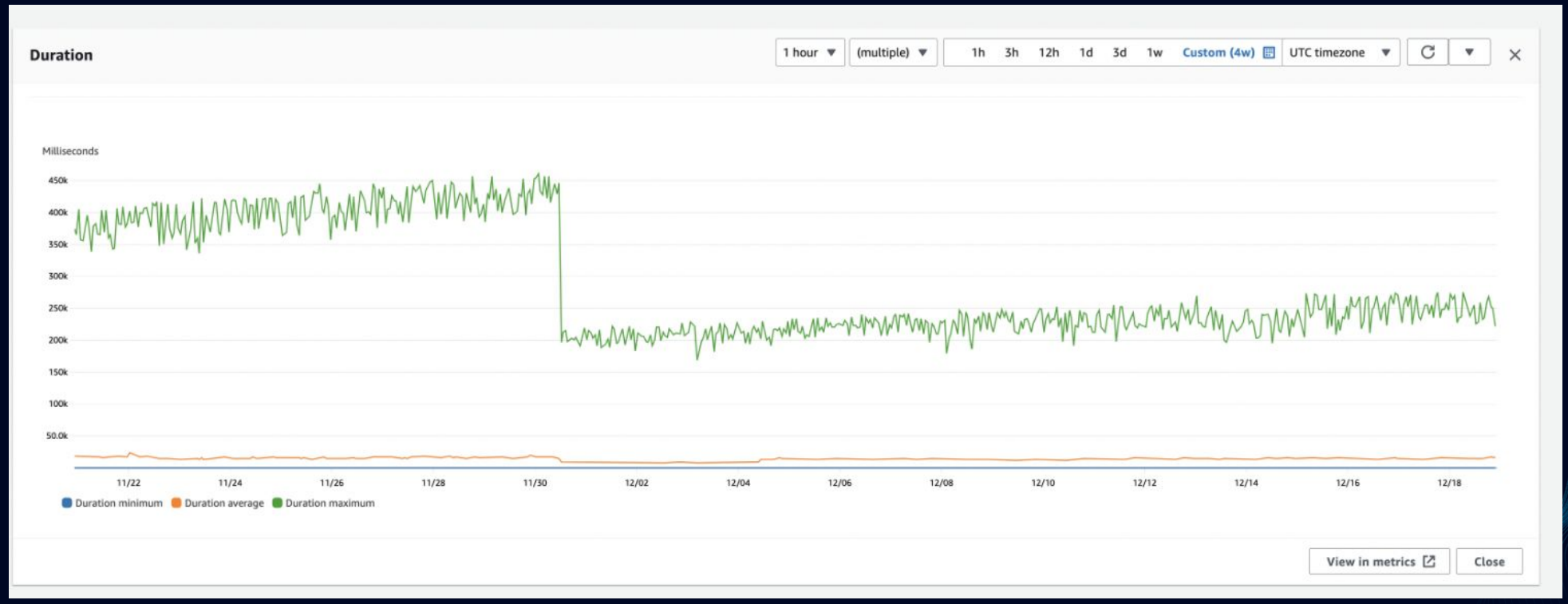
SPOG Lambda

- Extract key fields from raw json logs and convert to parquet
- Metrics - Daily Invocation 0.8 million (Prod), peak of 250 concurrent invocation/min
  - Lambda billing is related to memory allocated and duration of execution
  - Monitor resource consuming invocations and tune them
    - High billed invocation query
      - filter @type = "REPORT"
      - | fields @requestId, @billedDuration
      - | sort by @billedDuration desc
      - | limit 100
    - Max memory usage query
      - filter @type = "REPORT" and @maxMemoryUsed=@memorySize
      - | stats
      - count_distinct(@requestId)
      - by bin(30m)
    - Find lambda invocations which timed out
      - These are the ones which gets billed for entire duration and do not return results
      - These are great cost saving candidates

FOURKITES

SPOG Lambda

- Captured metrics/stats around spog queries and usage of columns
  - removed parsing of not needed columns
- Started with 3-4GB of memory allocation and reduced it to 900MB (prod)
- Reduce cost by a margin of 50% by reducing memory allocated and optimizing processing



FOURKITES

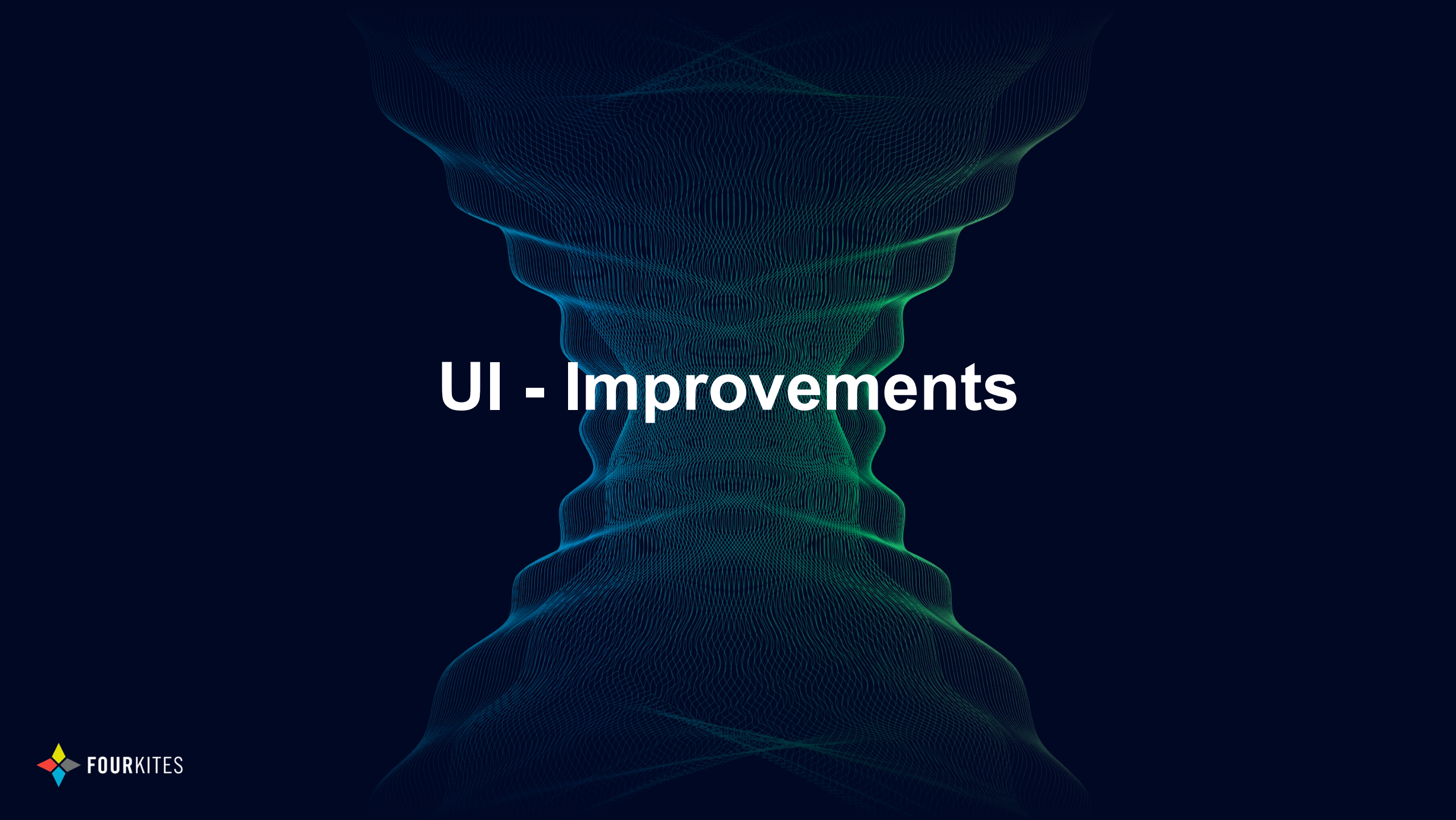
SPOG Storage (Compression + Intelligence)

Daily Metrics - ingestion of 7-8TB. 1 million+ files in s3

Retention of 120 days.

- S3 is cheap (but.. - unless used carefully and smartly).
  - Not cheap when data volumes are 7-8TB/daily ingestion
  - Standard tier is not cheap at large volumes.
  - We had raw layer and transformed layer.
    - Initially we persisted data in both raw layer and transformed layer for backup purpose
    - but as we matured in our logic/alerts/system we reduced backup duration from 90 days to 7 days only.
    - logs retention is for 120 days which is compressed logs in parquet.
    - S3 archival/delete policies

Above helped us reduce our S3 cost by 75% - as raw logs were json not compressed



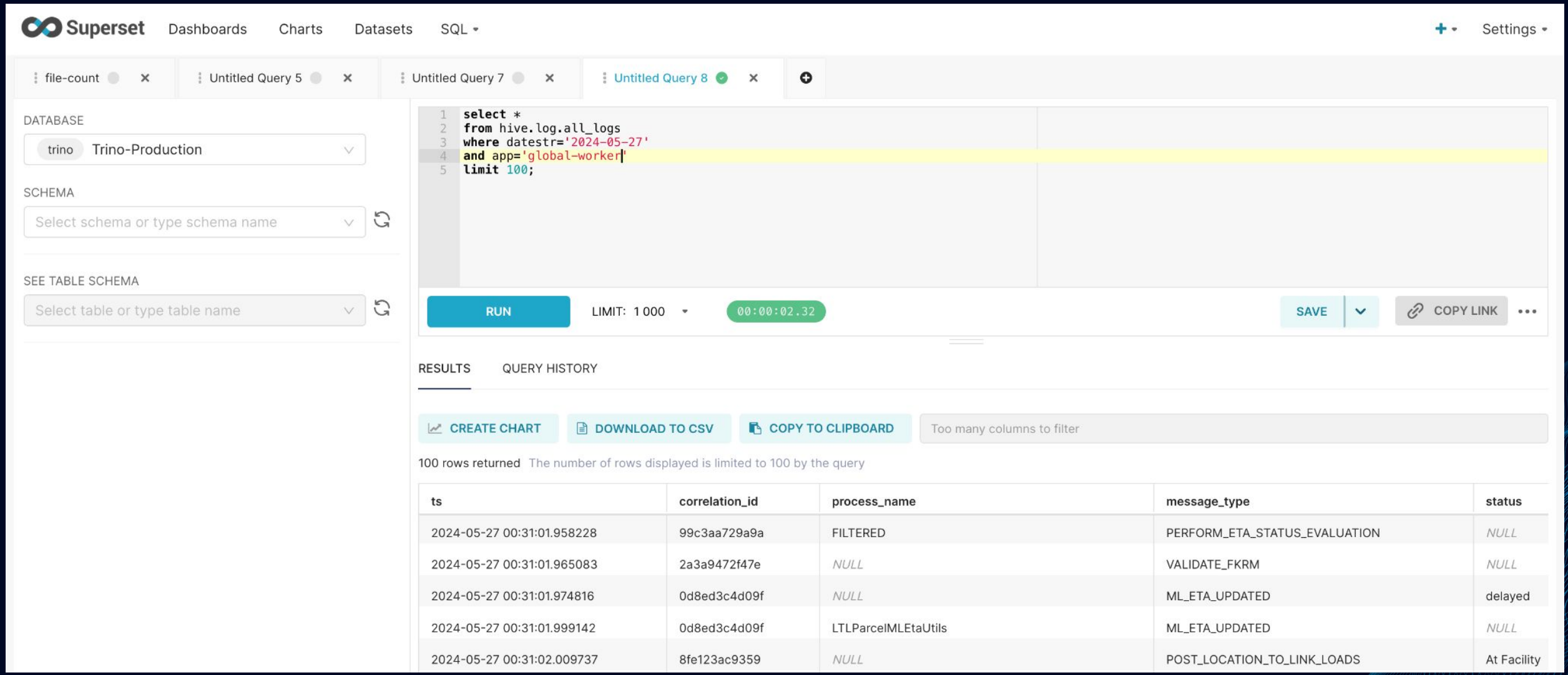
UI - Improvements

FOURKITES

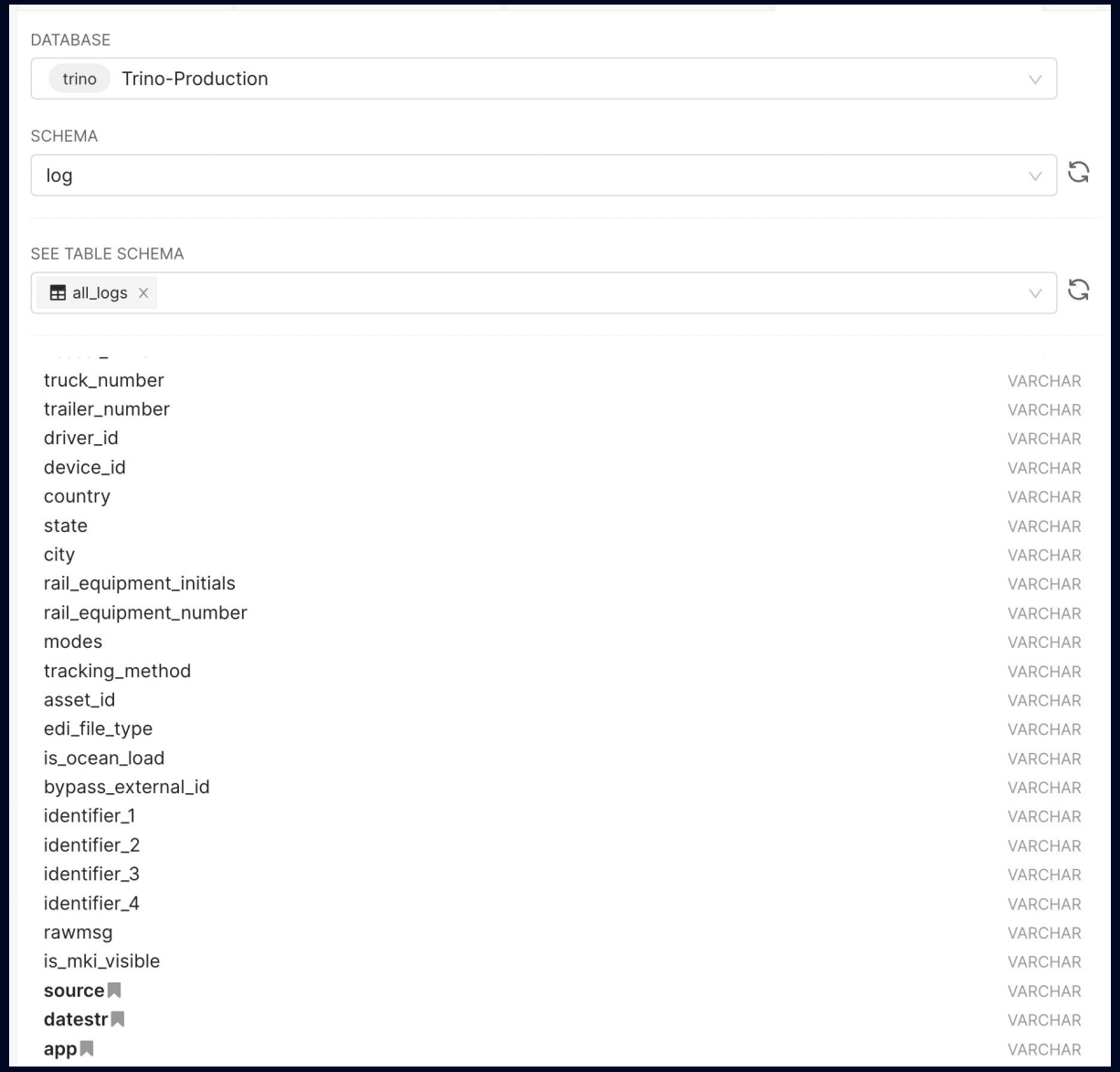
SPOG UI Scaling & Improvements

- On launch day SPOG UI crashed multiple times - as we anticipated less internal traffic, but actual traffic turned out more.
  - gevent:
    - Superset uses gunicorn which by default used ‘gthread’ worker model.
    - Each thread will handle 1 request at a time and scalability is achieved by increasing workers (equal to available processors) and threads per worker.
    - This does not fit well since the workload is primarily IO bound and waits for database to execute queries and scale only to a certain limit.
    - We moved ‘gevent’ model - which helped in scaling concurrent request handling. Page rendering time improved by 60% to 75% with 100 concurrent requests.
  - Async query execution:
    - Superset by default used sync query mode and this also limited the concurrent query handling and also timeouts were observed.
    - We adopted ‘async’ mode for query execution. Superset app will receive query, and forward the task celery worker. It will execute the query and store results in redis cache.
    - UI will only do a status check for query results availability. This helped in improving performance and allowing superset app server to handle more requests.

Demo


Superset
Dashboards
Charts
Datasets
SQL
Settings
file-count
Untitled Query 5
Untitled Query 7
Untitled Query 8
DATABASE
trino
Trino-Production
SCHEMA
Select schema or type schema name
SEE TABLE SCHEMA
Select table or type table name
select *
from hive.log.all_logs
where datestr='2024-05-27'
and app='global-worker'
limit 100;
RUN
LIMIT: 1 000
00:00:02.32
SAVE
COPY LINK
RESULTS
QUERY HISTORY
CREATE CHART
DOWNLOAD TO CSV
COPY TO CLIPBOARD
Too many columns to filter
100 rows returned
The number of rows displayed is limited to 100 by the query
ts
correlation_id
process_name
message_type
status
2024-05-27 00:31:01.958228
99c3aa729a9a
FILTERED
PERFORM_ETA_STATUS_EVALUATION
NULL
2024-05-27 00:31:01.965083
2a3a9472f47e
NULL
VALIDATE_FKRM
NULL
2024-05-27 00:31:01.974816
0d8ed3c4d09f
NULL
ML_ETA_UPDATED
delayed
2024-05-27 00:31:01.999142
0d8ed3c4d09f
LTLParcelMLEtaUtils
ML_ETA_UPDATED
NULL
2024-05-27 00:31:02.009737
8fe123ac9359
NULL
POST_LOCATION_TO_LINK_LOADS
At Facility

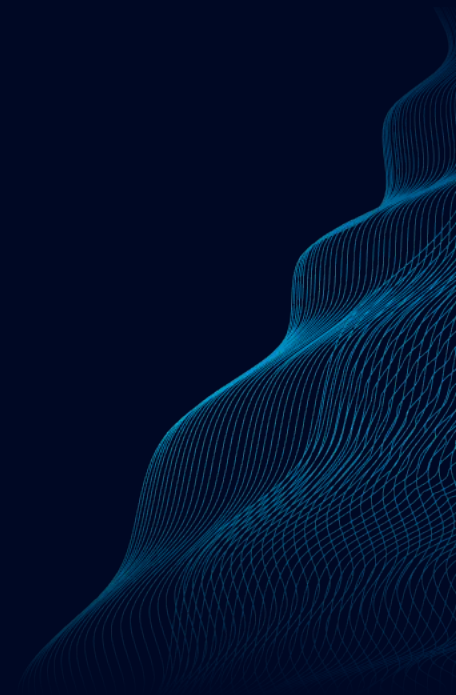
Demo



FOURKITES

Metrics

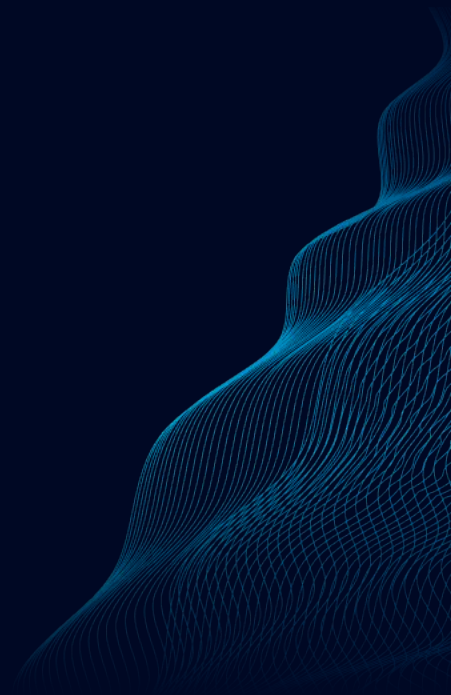
- Supporting around 350 application logs (across aws/azure cloud)
- Total users - 300
- Total files ingested ~1 million+ (Daily)
- Data Scanned is in TB’s



FOURKITES

Success

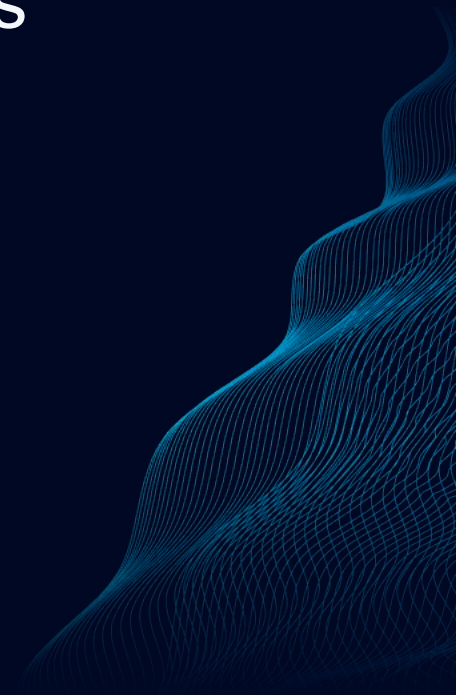
- Solution to query and analyze logs with larger retention of data
- Costing around $120k for supporting approx 300 (internal + external users (engg + operations + support))
- 120 days of data retention
- Quick and easy onboarding for any application
- Capability to join/lookup data within fourkites
- Unlimited opportunities to build reports/query data etc.



FOURKITES

Lesson Learned

- S3 is not cheap if data volume is large.
- Standardisation of logging format across apps has its benefits
- Reduce log generation at app level itself Vs ignoring the emitted logs by drop filters/similar
- Lambda cost will increase drastically if there are timeouts happening consistently.



FOURKITES

Next Steps

- S3 intelligent tiering
- Explore - s3 Glacier storage class adoption for increased retention at lower cost. (need to evaluate cost benefits)
- Audit the data coming into system
- Improve monitoring (especially around fluent-bit + ingestion)
- Cost optimization opportunities
- Learnings from Trino-fest 2024

FOURKITES