DATA+AI SUMMIT 2022

An Adaptive File Connector for Spark to Hunt for Cyber Attacks

Wojciech Indyk

Big Data Engineer, Hunters

Ada Sharoni

Senior Software Engineer and Team Leader, Hunters

ORGANIZED BY databricks

Ada Sharoni

Software Engineer Team Lead @ Hunters.ai

1. ML & Big Data
2. 7 Years, different security solutions:
   a. Enterprise Network Security
   b. WAF (Web Application Firewall)
   c. Fraud Detection
3. Fun Fact: I started out as a Hardware Engineer

https://twitter.com/AdaSharon

https://www.linkedin.com/in/ada-sharoni-47ba26b8/

Wojciech Indyk

Big Data Engineer @ Hunters.ai

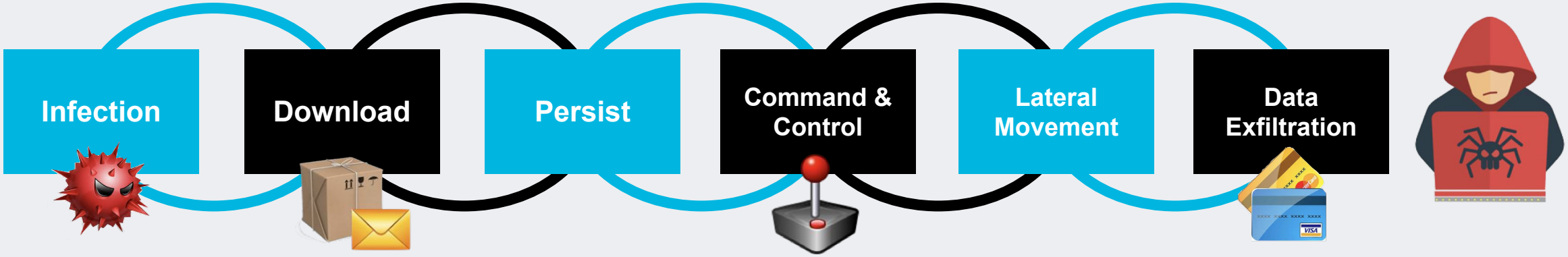
- Big Data and Data Science for over 10 years
- 11 scientific papers regarding distributed machine learning systems
- I love open-source! ❤️

https://pl.linkedin.com/in/wojciechindyk

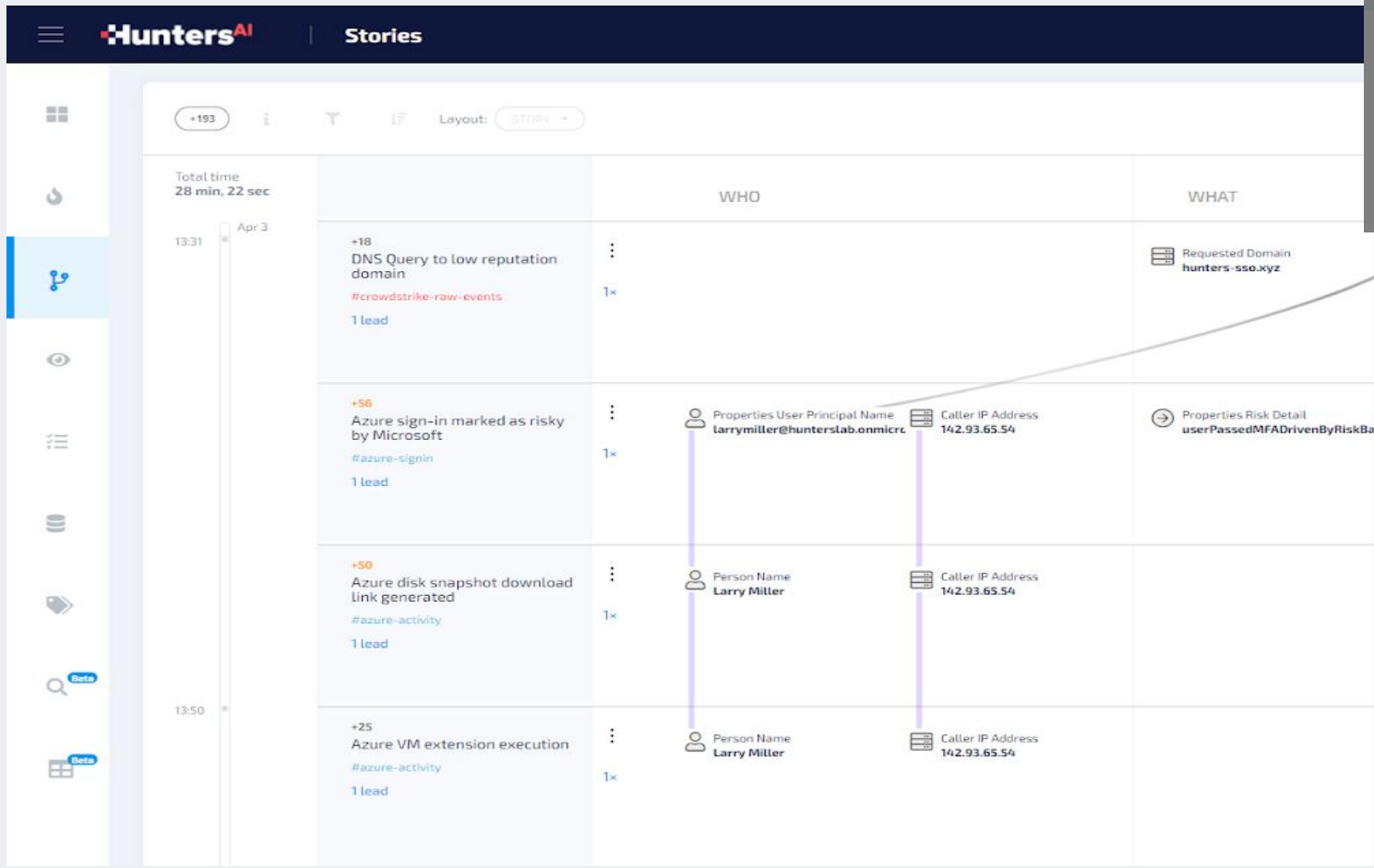
Hunters

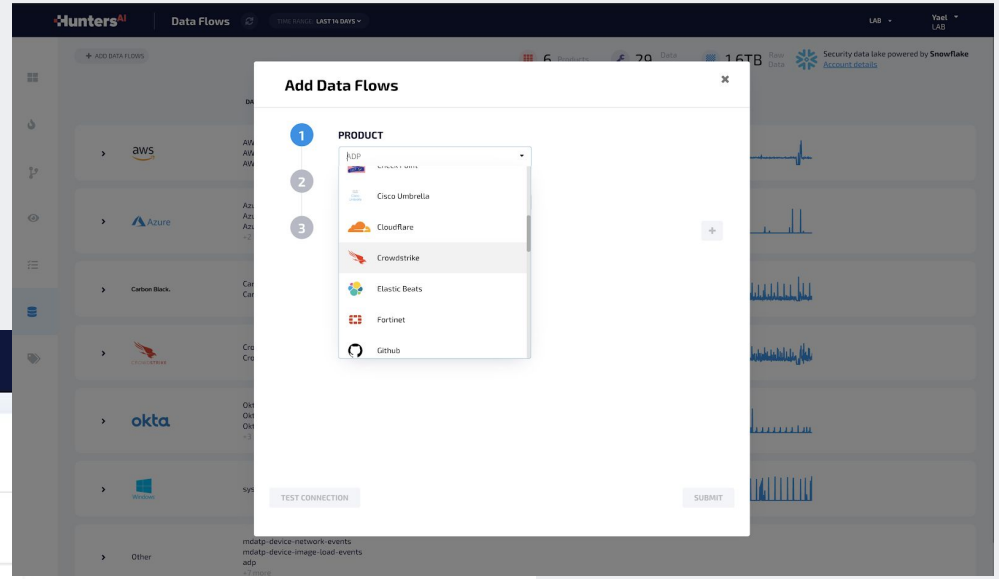
Security Operations Platform

- Help security teams understand the full attack story
- Correlate existing telemetry and sources across surfaces
  - Network
  - Endpoint
  - Cloud
  - SasS
  - Email
  - etc



Variety of Data Sources





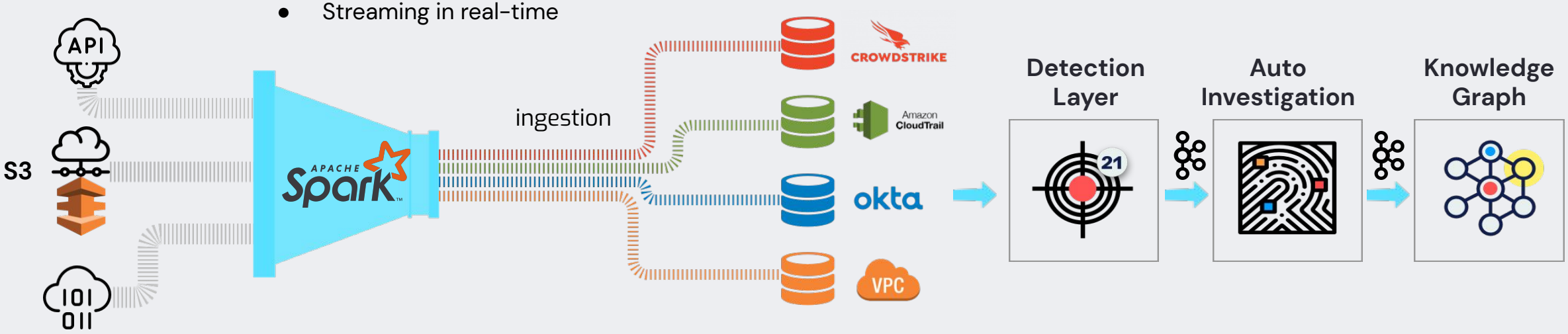
Streaming Security Data in Real-Time

Data Sources

Flexible Ingestion

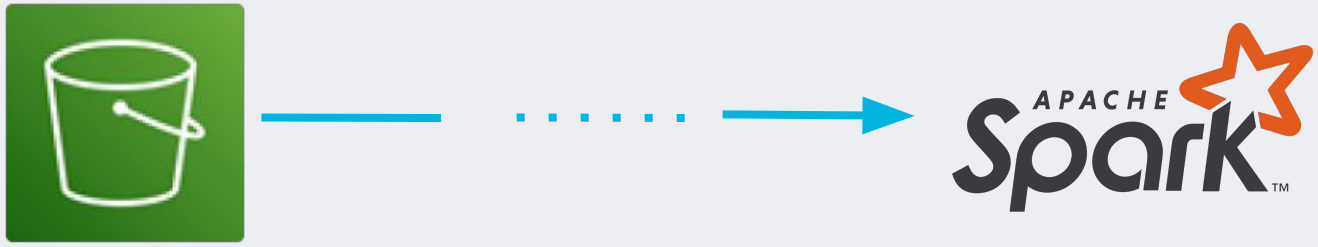
- multiple formats
- multiple sources
- Streaming in real-time

Data Lake



Problem definition

1. 50% Of security data logs get shipped to object store directly (e.g. S3)
2. S3 Bucket belong to customers -> no retention, huge bucket size
3. We need to detect live cyber attacks in real-time → streaming



huge & growing !

Unfortunately …

S3 is not HDFS !

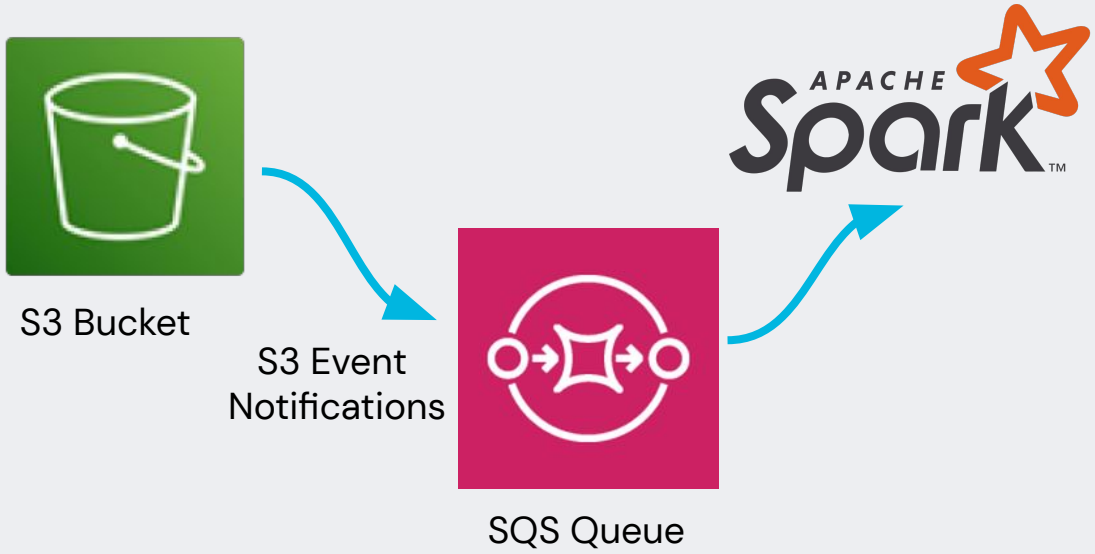
1. S3 is an object store
2. Listing huge buckets is problematic:
   a. Takes ~24h
   b. Operation costs money

There Are Existing Solutions

S3 Streaming via SQS

Open source: [qubole/s3-sqs-connector]

Managed: [Databricks's auto-loader]



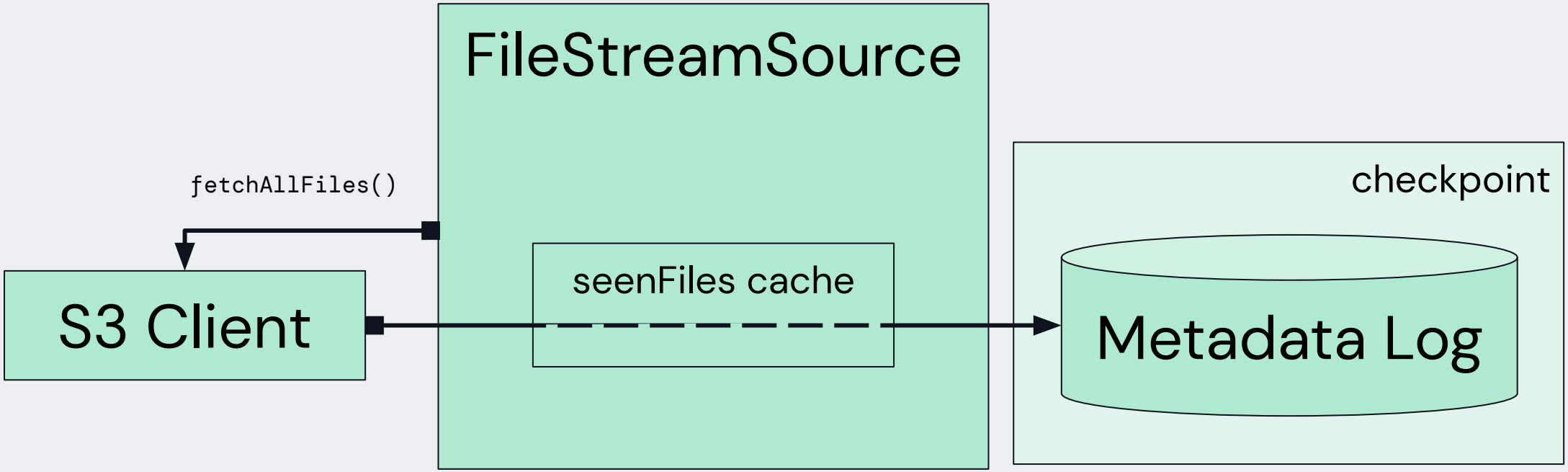
Pain Points:

- Increases complexity of the system
- Customers don't want to grant sensitive permissions

So how can I stream data from s3 buckets ?

Spark-core current state

FileStreamSource - how does it work?

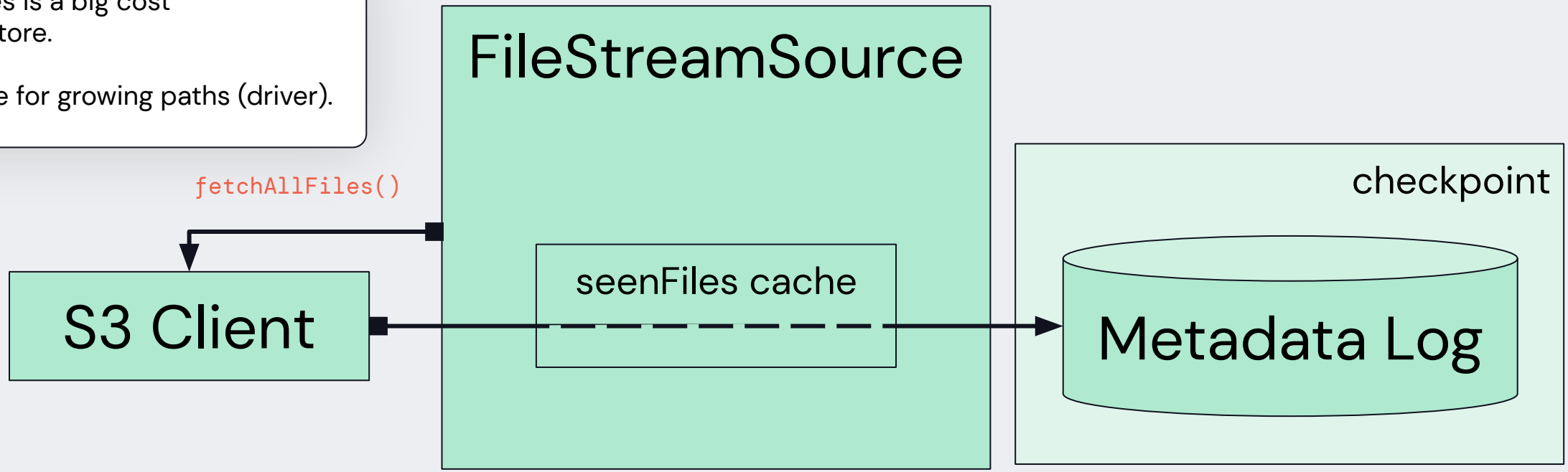


Spark-core current state

FileStreamSource - how does it work?

Fetch all files is a big cost
for object store.

Not scalable for growing paths (driver).



Solution design

Path structure

Directory structure for time-series data usually looks like:

s3:// <bucket_name> / <account>/ <service_name>/ <YYYY> / <MM> / <DD> /

Solution design

Time-series characteristic

Directory structure for time-series data usually looks like:

s3:// <bucket_name> / <account>/ <service_name>/ <YYYY> / <MM> / <DD> /

this part is
dynamic in time !

Solution design

Dynamic Path Generator

s3:// <bucket_name> / <account>/ <service_name>/ <YYYY> / <MM> / <DD> /

Static Part

(known upfront)

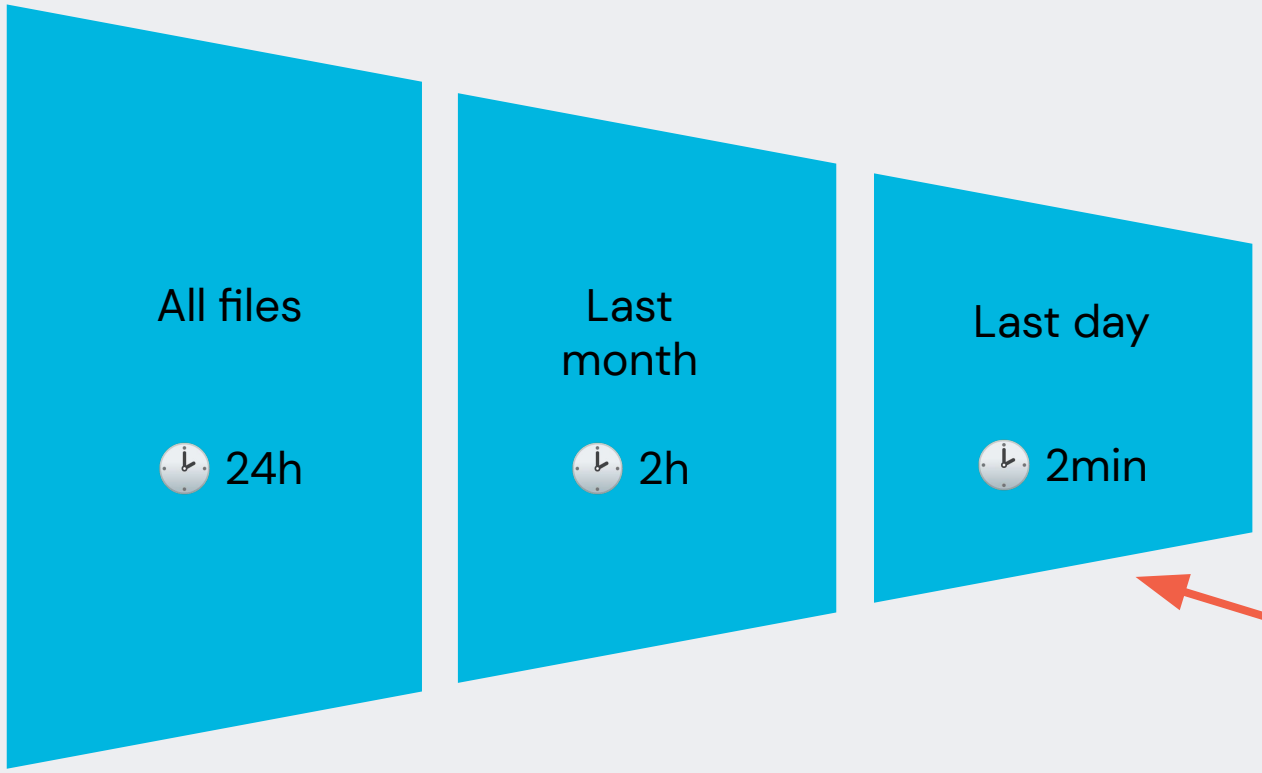
Dynamic Part

(includes time dimension)

Solution design

In streaming we only need to fetch the latest data

/ <YYYY> / <MM> / <DD> /



For most streaming cases we need only 1 day

Recap

Recap

Solution design

If you:

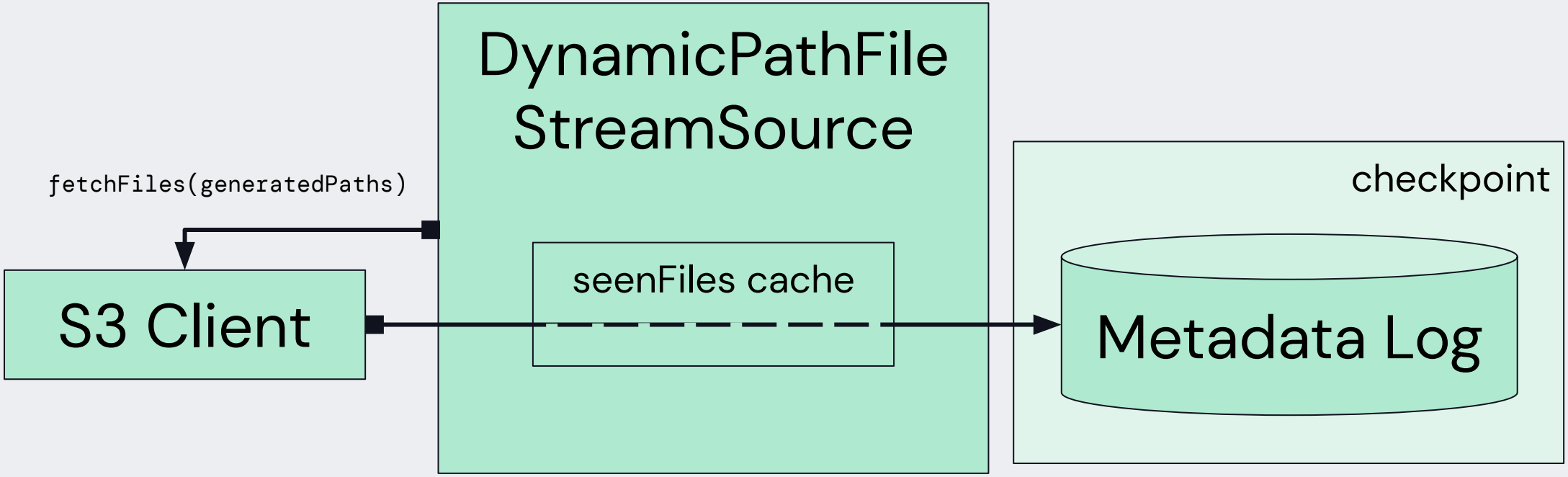
1. Are using Spark-Streaming
2. Know the structure of your S3 paths
3. Have dynamically changing part of your path
   a. (And you know how it's changing, e.g. data partitioned by day)

Then you should pay attention to our adaptive file connector for spark !

Adaptive S3 connector for Spark

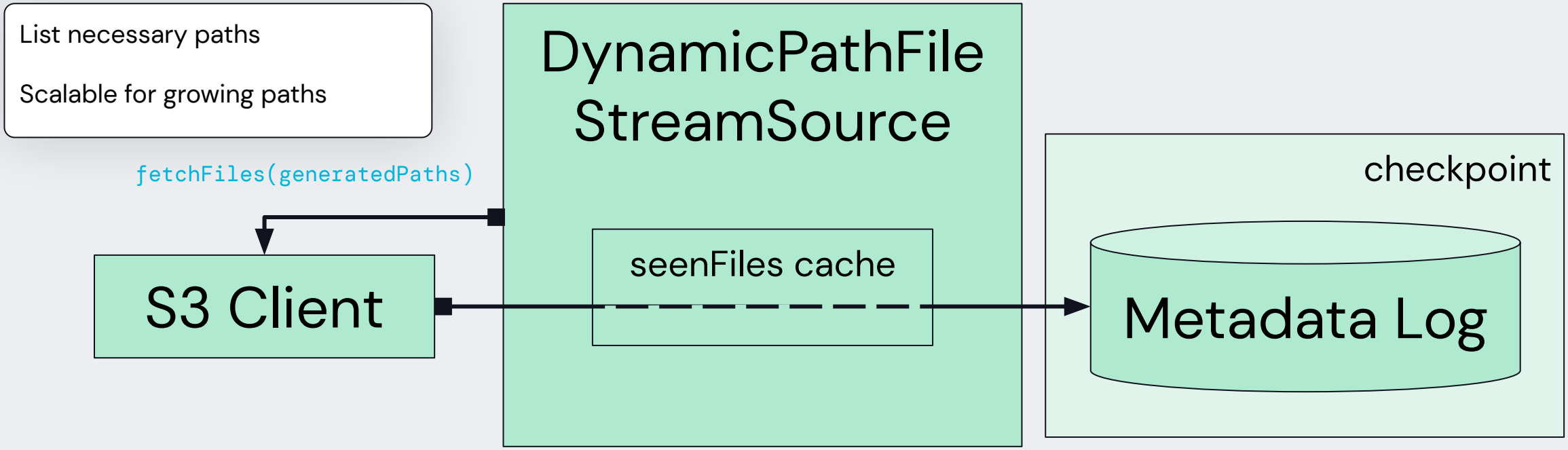
Adaptive S3 connector for Spark

Architecture



Adaptive S3 connector for Spark

Architecture



Adaptive S3 connector for Spark

Easy Usage

```
val readFileJsonStream = spark.readStream
      .format("dynamic-paths-file")
      .option("fileFormat", "json")
      .schema(yourSchema)
      .load("s3://my-bucket/prefix/{YYYY}/{MM}/{DD}")
```

Adaptive Source Connector

```
val readFileJsonStream = spark.readStream
      .format("json")
      .schema(yourSchema)
      .load("s3://my-bucket/prefix/*/*/*")
```

Native Spark File Source Connector

Adaptive S3 connector for Spark

Interface (Extending is also easy)

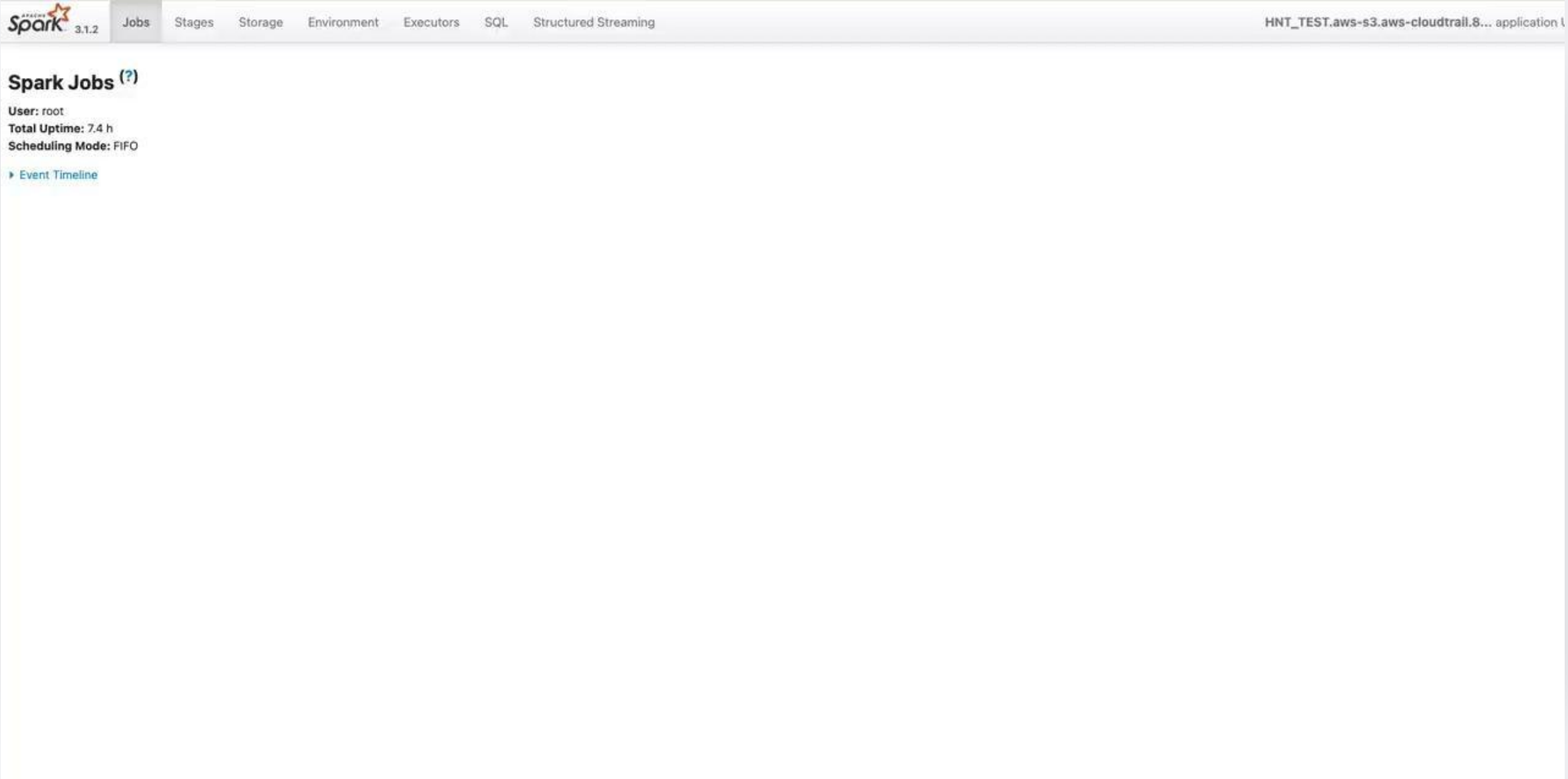
```
trait PathGenerator (
    def getPaths: Set[String]
)

new DynamicPathGenerator extends PathGenerator (
    wildcardedGlob: String,    // e.g. s3://my_bucket/data/{YYYY}/{MM}/{DD}/
    maxNumberOfDaysToRead: Int,
    timeZone: ZoneId       // timezone of data might be different from the server
)

```

Demo Time!

Demo – files to be processed

```
~/repos/hunters                                                                  ✔ 12:50:39
aws --profile airflow-lab-to-hnt-test s3 --human-readable ls hunters-lab-audit/AWSLogs/591644025889/CloudTrail/us-west-1/2022/06/22/
```

Demo – Native Spark File Source



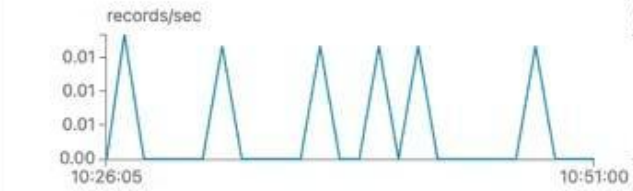
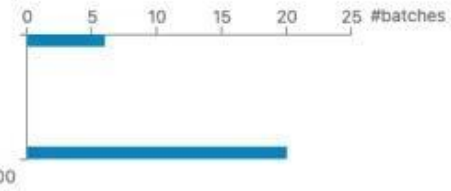
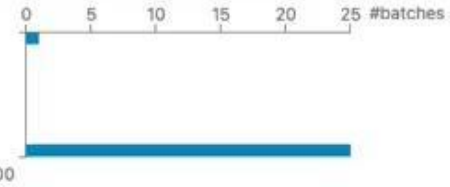
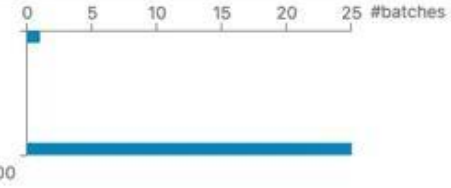
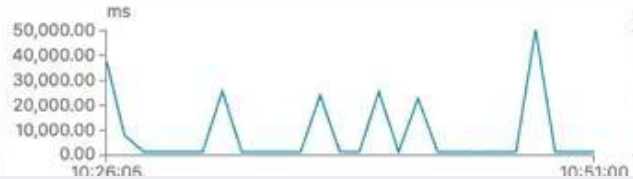
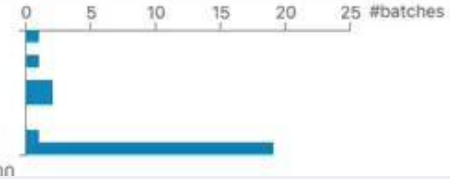
Demo- Adaptive File Source

Spark 3.1.2 | Jobs | Stages | Storage | Environment | Executors | SQL | Structured Streaming | HNT_TEST.aws-s3.aws-cloudtrail.8... application UI

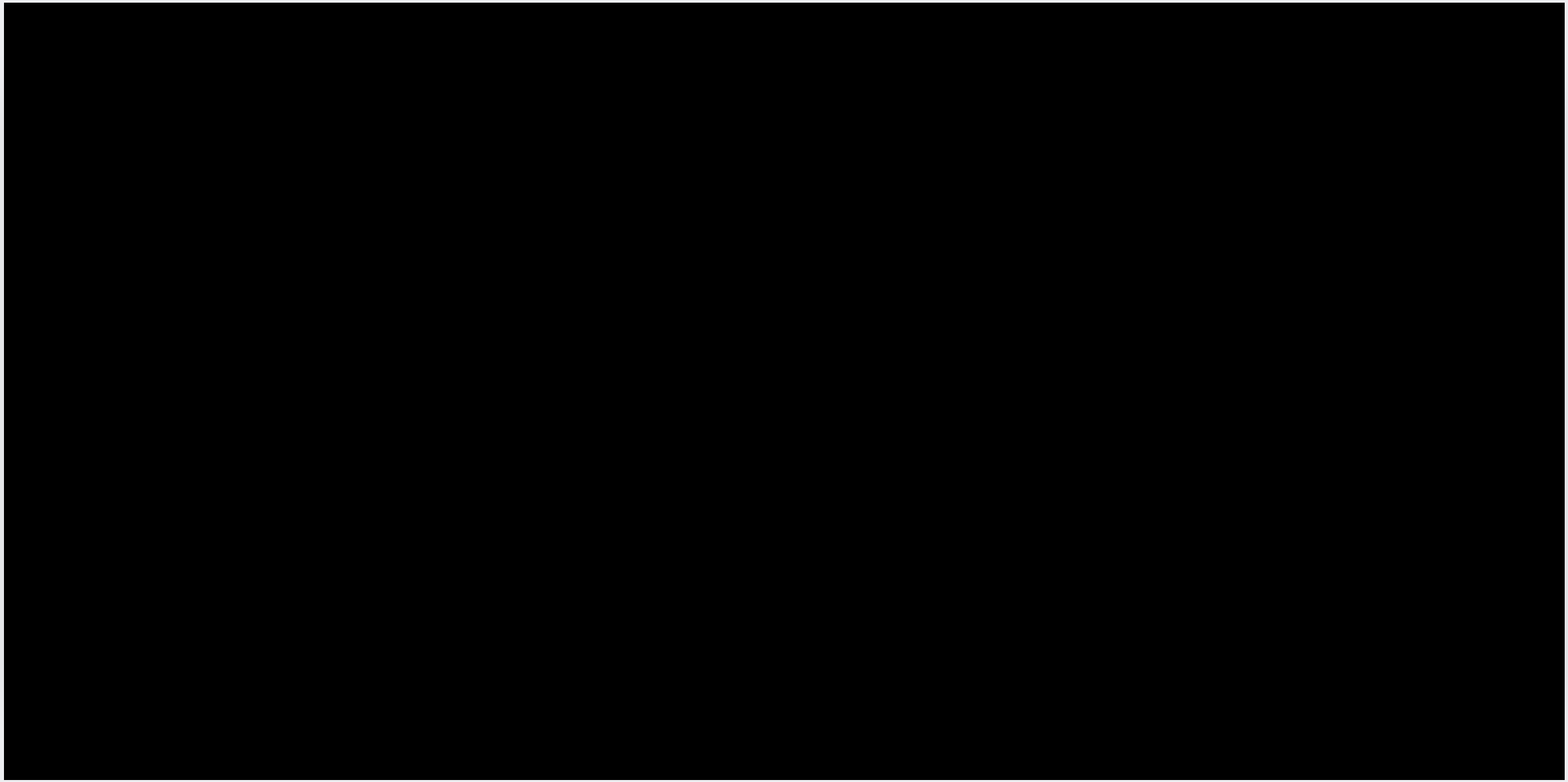
Streaming Query Statistics

Running batches for **25 minutes 52 seconds** since **2022/06/22 10:26:04** (**8** completed batches)

Name: spark
Id: f5b962e8-ef0b-46b7-9ba9-94297b821a42
RunId: 1251a54c-807b-4eeb-bc73-34dc2f465ba8

	Timelines	Histograms
Input Rate (?)		
Process Rate (?)		
Input Rows (?)		
Batch Duration (?)		

Simulated Real Time Attack



The solution is now Open-Sourced!

https://github.com/hunters-ai/spark-adaptive-file-connector

Sponsored by

HUNTERS

Roadmap

Advanced S3 connector for Spark

1. Flexible specification of the time dimension- any spark-supported time formatting
2. Blacklist/whitelist pattern parameter
3. Extract more info from the bucket partitioning as extras columns - e.g. account, region, etc.
4. Try against Azure Blob Storage & Google Cloud Storage
5. Migrate Source from v1 to v2 for support of Continuous Processing mode

DATA+AI
SUMMIT 2022

Thank you

Wojciech Indyk

Big Data Engineer, Hunters

wojciech.indyk@hunters.ai

Ada Sharoni

Senior Software Engineer
and Team Leader, Hunters

ada@hunters.ai