Arkime Stream Processing with Kafka

Why use Kafka and how.

Owen McGill

Arkimeet 2023 – May 23rd

Outline

- Context
- Solution
- Use cases
- Processors

About Me

- **Dev/Ops Engineer at Open Systems AG**
- **Using Arkime since 2021**
- **Love Go (the programming language)**
- **First time talking at a conference**

@mcgillowen

Context

Why?

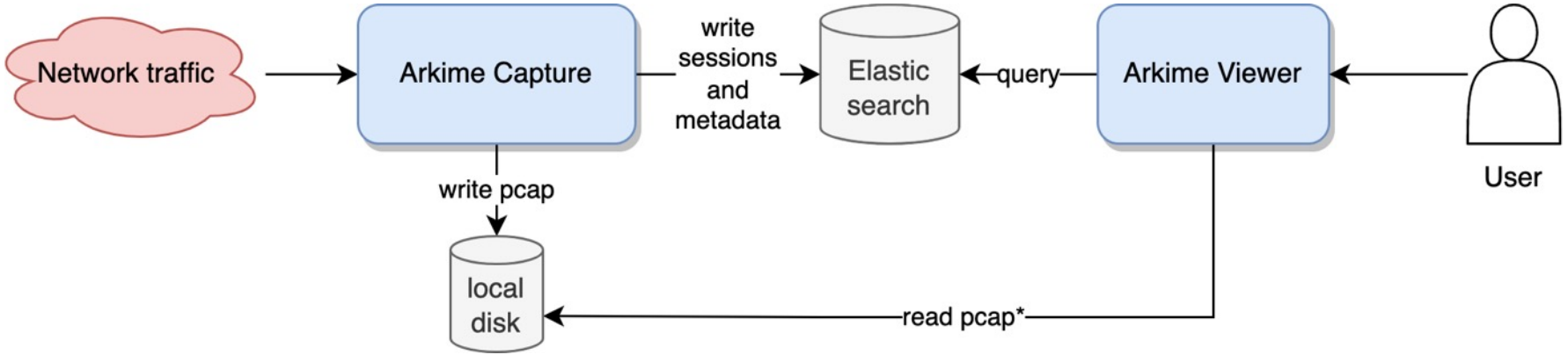
“Vanilla” Arkime

Arkime Capture

- Listen to a network interface
- Write pcaps to (local) disk
- Write metadata and SPI (sessions) to ES

Arkime Viewer

- Query ES and displays sessions
- Read pcaps if/when colocated or relay query to appropriate Viewer



Limitations of “vanilla” Arkime

What is missing or lacking

Enrichment

- **External IOCs**
- **Extra metadata (eg. GeoIP2 Enterprise)**

Archival

- **SPI data in distributed storage (eg. HDFS)**
- **PCAP files from subset of Sessions**

Existing enrichment methods

- **WISE**
- **Lua**
- **Tagger**

Existing archival methods

- **Cron jobs (only for SPI)**

Constraints

What limitations do we have

Capture node resources

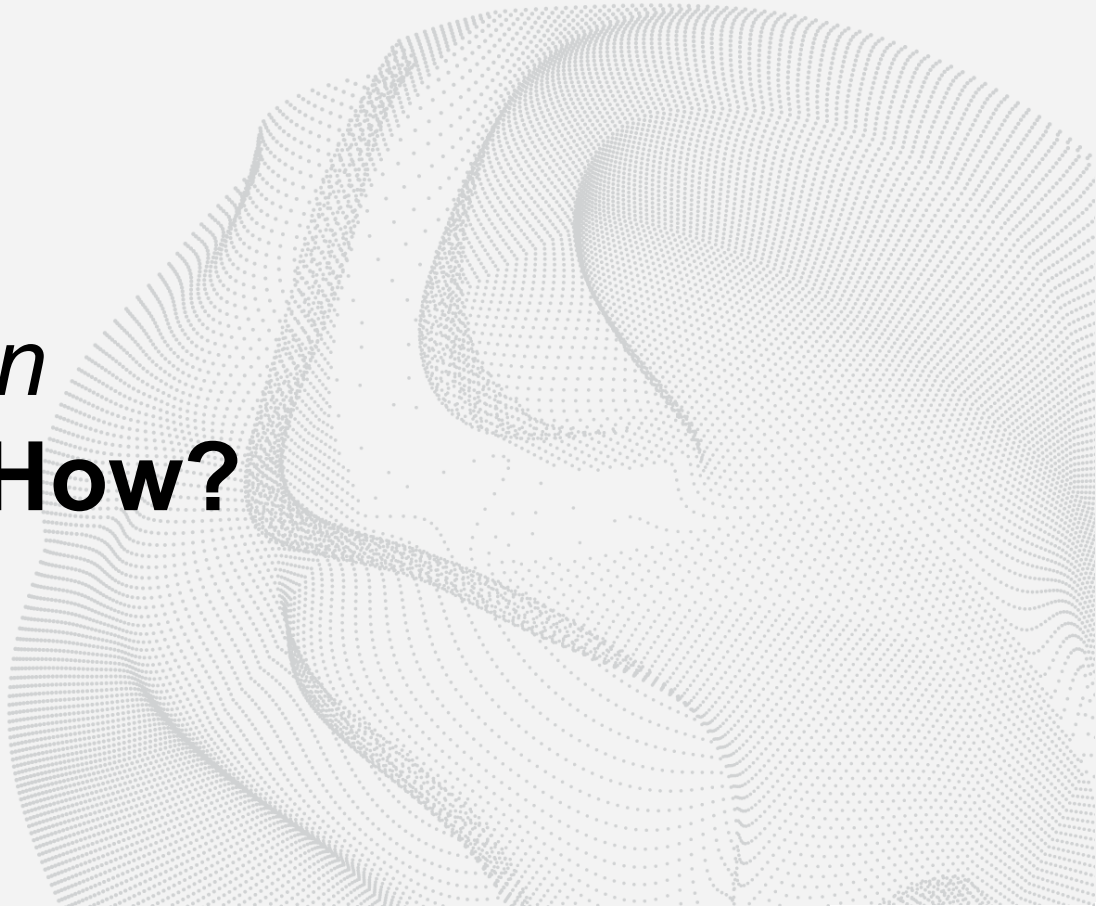
- Capture nodes are finite and static
  - Physical constraints, eg. space, heat, NPB ports
- Large pool of resources in our Kubernetes cluster
- Very high network rate per node

Lack of C developers

- Most are Scala developers
- Some Go developers
- Very few with a good understanding of C

Solution

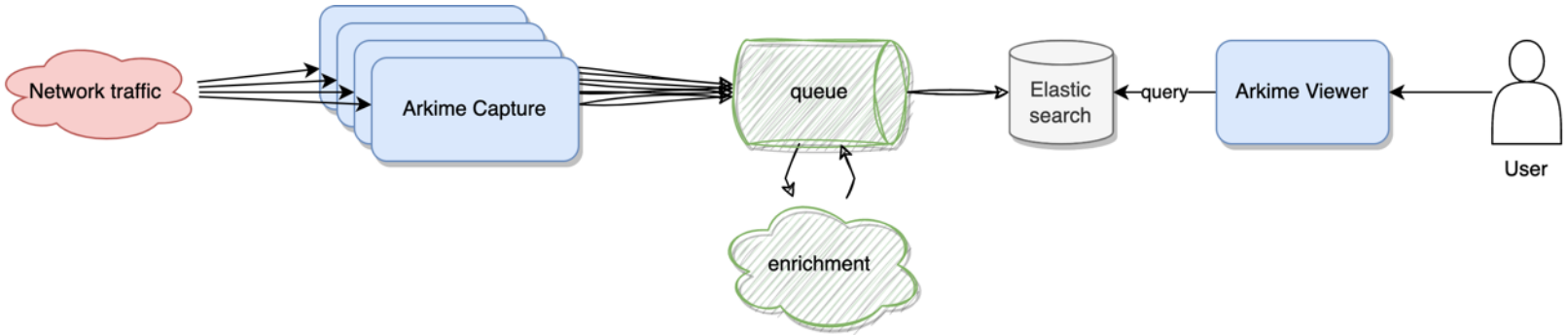
How?



Solution => Decoupling

Decouple Arkime Sessions Bulk Indexing

- Arkime writes to a queue/buffer
- Queue/buffer gets indexed in ES
- Enrichment done on data in queue/buffer and then added back to queue/buffer



What is Kafka?

And why we love it

- Distributed real-time streaming platform
- Handles billions of messages per second
- Allows multiple writers and readers per data stream
- We can quickly change the ordering of the processing
- We easily handle 10Gb/s in and out on our Kafka nodes

Kafka plugin

- In use since 2020 internally
- Developped by Benoit Perroud, with some additions from Andy Wick
- 3 message formats:
  - bulk: Identical data as ES bulk call per Kafka message
  - bulk1: Single session with ES bulk action line per Kafka message
  - doc: Single session without ES bulk action line per message, index in session JSON
- Uses librdkafka library which is very fast and efficient

Bulk

```
{"index": {"_index": …, "_id": …}}
{"firstPacket": …,"lastPacket": …,"length": …, …}
{"index": {"_index": …, "_id": …}}
{"firstPacket": …,"lastPacket": …,"length": …, …}
{"index": {"_index": …, "_id": …}}
{"firstPacket": …,"lastPacket": …,"length": …, …}
…
```

Bulk1

```
{"index": {"_index": …, "_id": …}}
{"firstPacket": …,"lastPacket": …,"length": …, …}
```

Doc

```
{"index": …,"firstPacket": …,"lastPacket": …,"length": …, …}
```

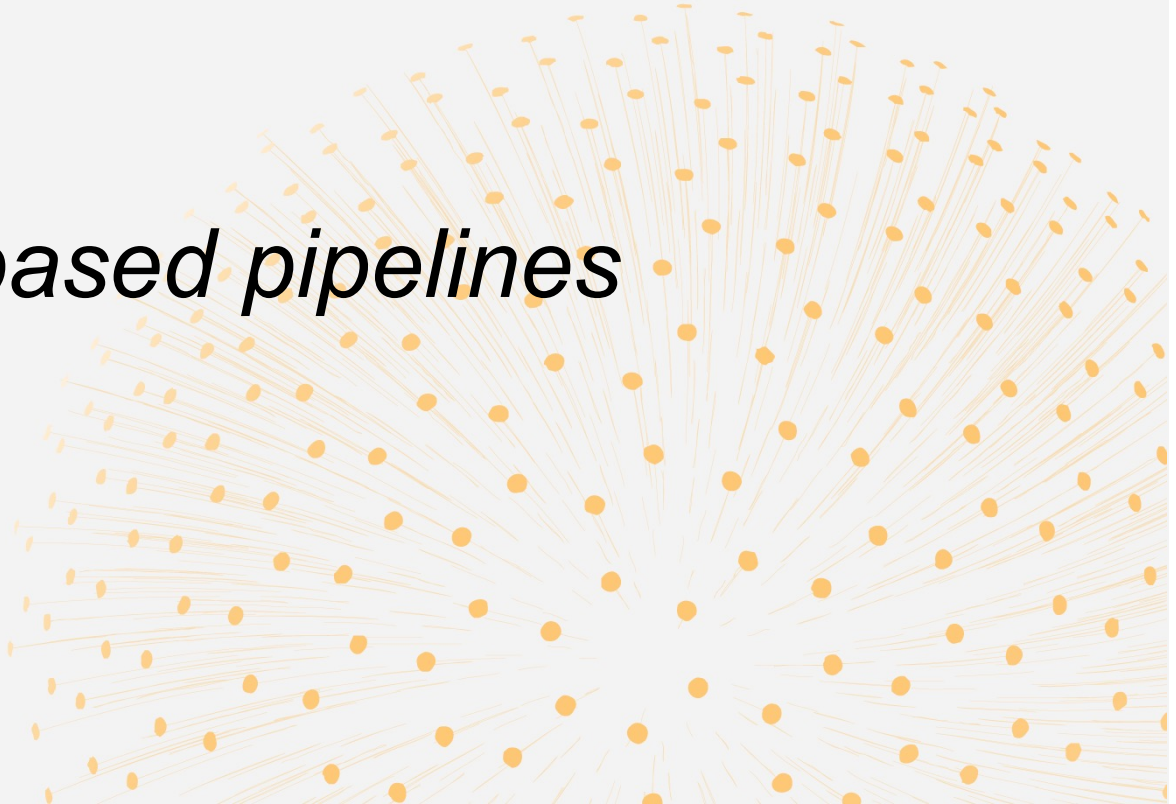
Advantages

Why use Kafka?

- Handles spikes in traffic
- Allows processing SPIs multiple times
- Offload enriching of SPIs from capture nodes to compute nodes
- Allows using any programming language, in our case Go

Use cases

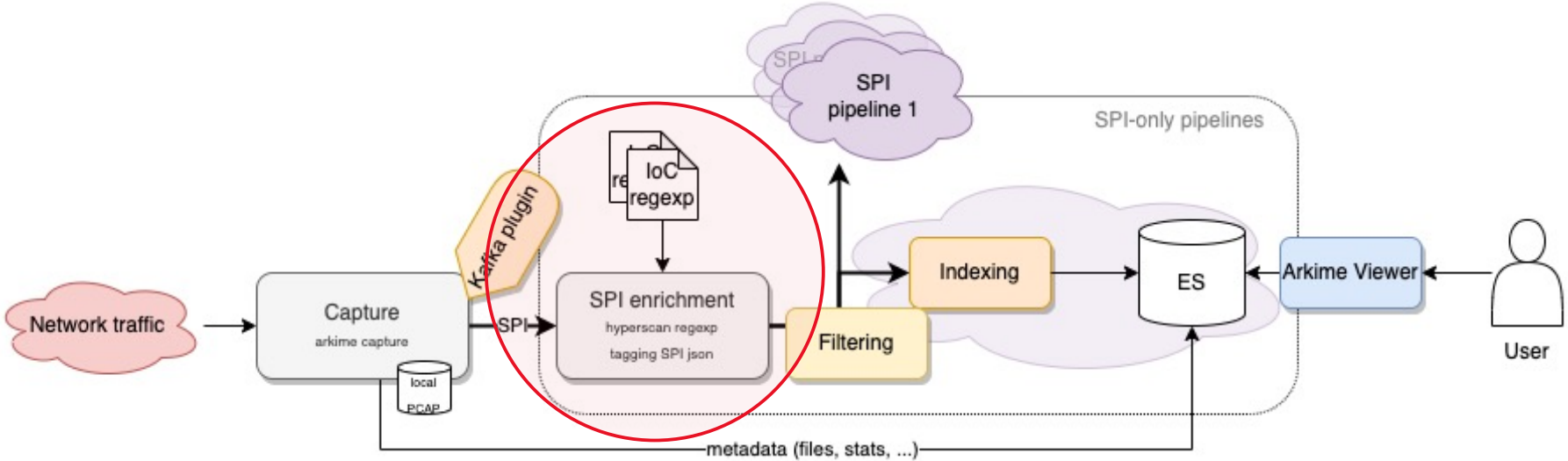
SPI based pipelines



Generic SPI enrichments

Enrichments pre indexing in ES

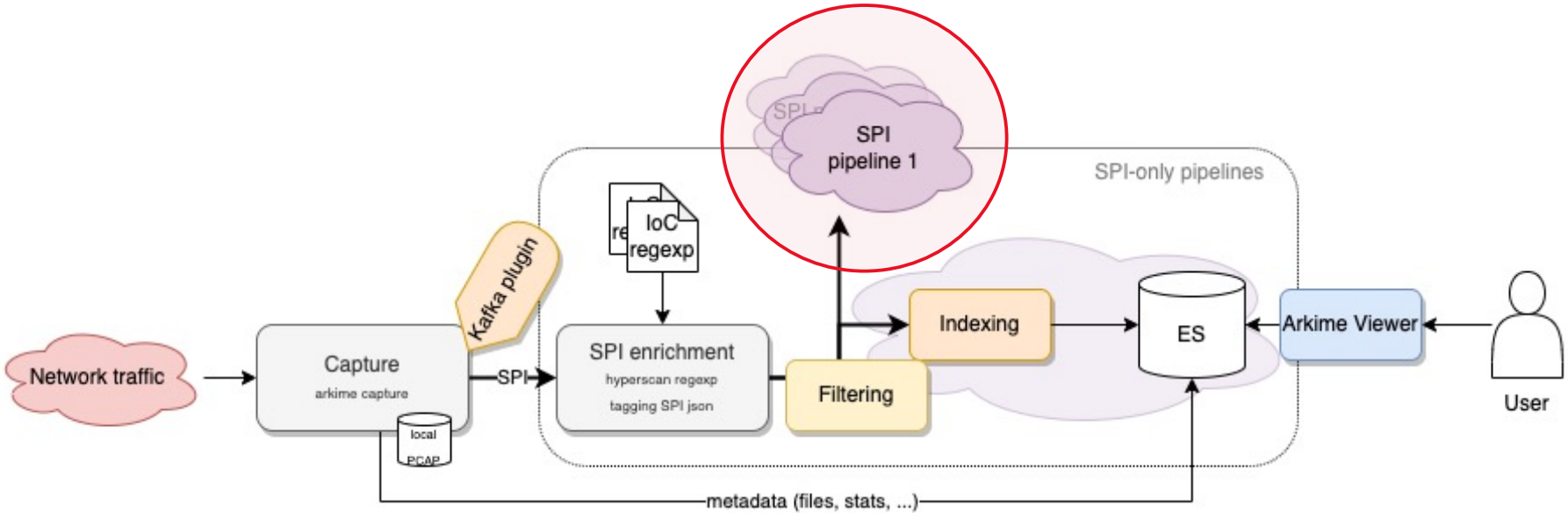
- Hyperscan regexps for tagging SPIs, like the Tagger plugin
- Filtering of SPIs based on content, like Arkime/bpf Rules
- Adding information from external sources, like WISE
- Archiving SPIs for longterm storage



SPI specific pipelines

Analysis external of Arkime

- C2 identification
- Any other analysis that can be done only on the data available in the SPIs



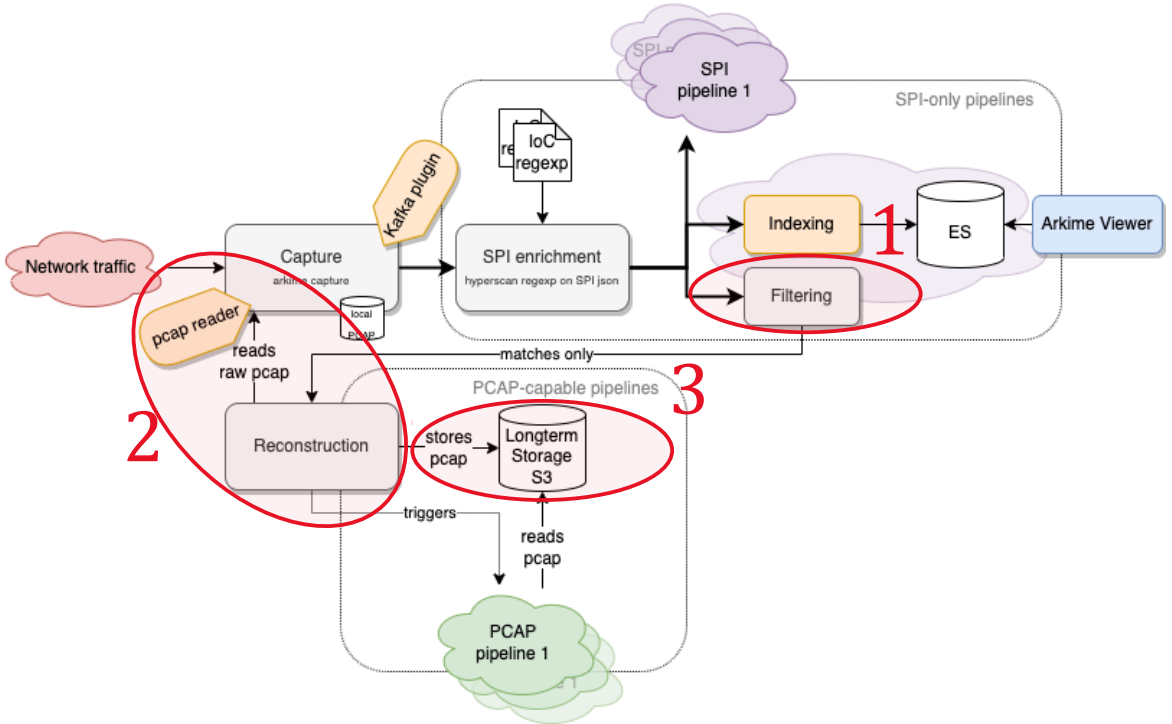
PCAP based pipelines

PCAP reconstruction pipeline

Longterm storage of PCAPs and reuse

1. Filter SPIs
2. Reconstruct, read PCAPs from capture node
3. Store in S3

Enables longterm storage decoupled from Arkime

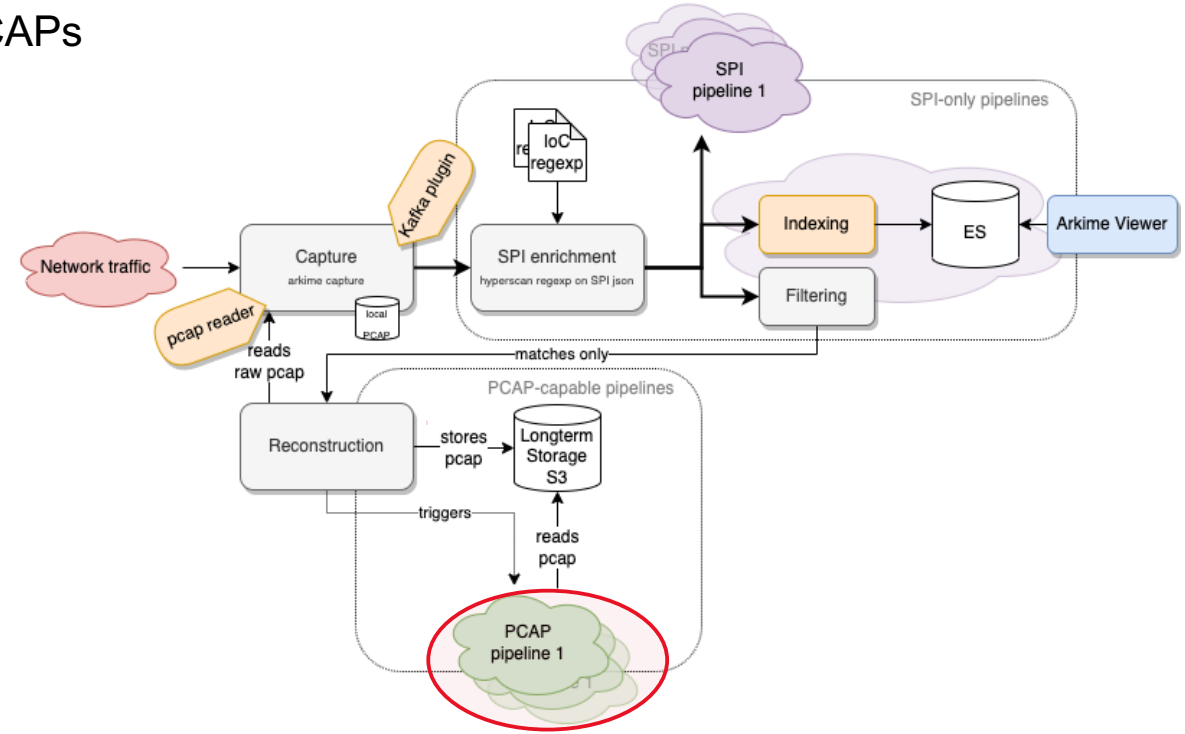


Generic PCAP pipelines

Reuse PCAPs

Various pipelines that require PCAPs

- e.g. Zeek



Processors

Processors we use

Processors we have developped:

- Arkime Kafka Indexer
- Arkime Kafka Filterer
- Arkime Kafka WISE
- Arkime Kafka Matcher

Arkime Kafka Indexer

Required component with the Kafka plugin

- Reads Kafka messages and sends them to ES, using Bulk API
- Can buffer contents of Kafka messages for improved ES performance
- Can preserve documents that failed to be indexed
- Load balances across ES data nodes if running in K8s
- Performance is only limited by the ES cluster, as far as we've noticed

Arkime Kafka Filterer

Optional component

- Can filter on any key in the SPI, with a matching value or any value
- SPI documents can be either kept or dropped when matched
- Can handle approximately 125’000 documents per second

Arkime Kafka WISE

Optional component

- Functions similarly to the WISE capture plugin
- Does not communicate with the WISE service
- Currently only supports enriching using data from MaxMind databases
- Provides the PUG templates for the viewer and creates the documents in the fields index

Arkime Kafka Matcher

Optional component

- Functions similarly to the Tagger plugin, but without field or type limitations
- Uses Hyperscan matching library
- Can handle approx 2 Million different regular expressions applied to each session

Questions?

Thank you