BUILDING A LARGE SCALE INTRUSION DETECTION SYSTEM USING BIG DATA TECHNOLOGIES

Liviu Vâlsan, Pablo Panero, Vincent Brillault, Cristian Schuszter

International Symposium on Grids & Clouds 2018, 22nd of March 2018

WHAT IS A SECURITY OPERATIONS CENTER?

- **Centralised** system for the detection, containment and remediation of IT threats.

- Ensures that security incidents are properly:
  - Identified
  - Analysed (real time and historical data)
  - Reported
  - Acted upon

SYSTEM DESIGN

- Unified platform for:
  - Data ingress
  - Storage
  - Analytics
- Multiple data access / view patterns:
  - Web based dynamic dashboards for querying and reporting
  - Command line interface that can be easily scripted
- Extensible, pluggable, modular architecture
- Unified data access control policies

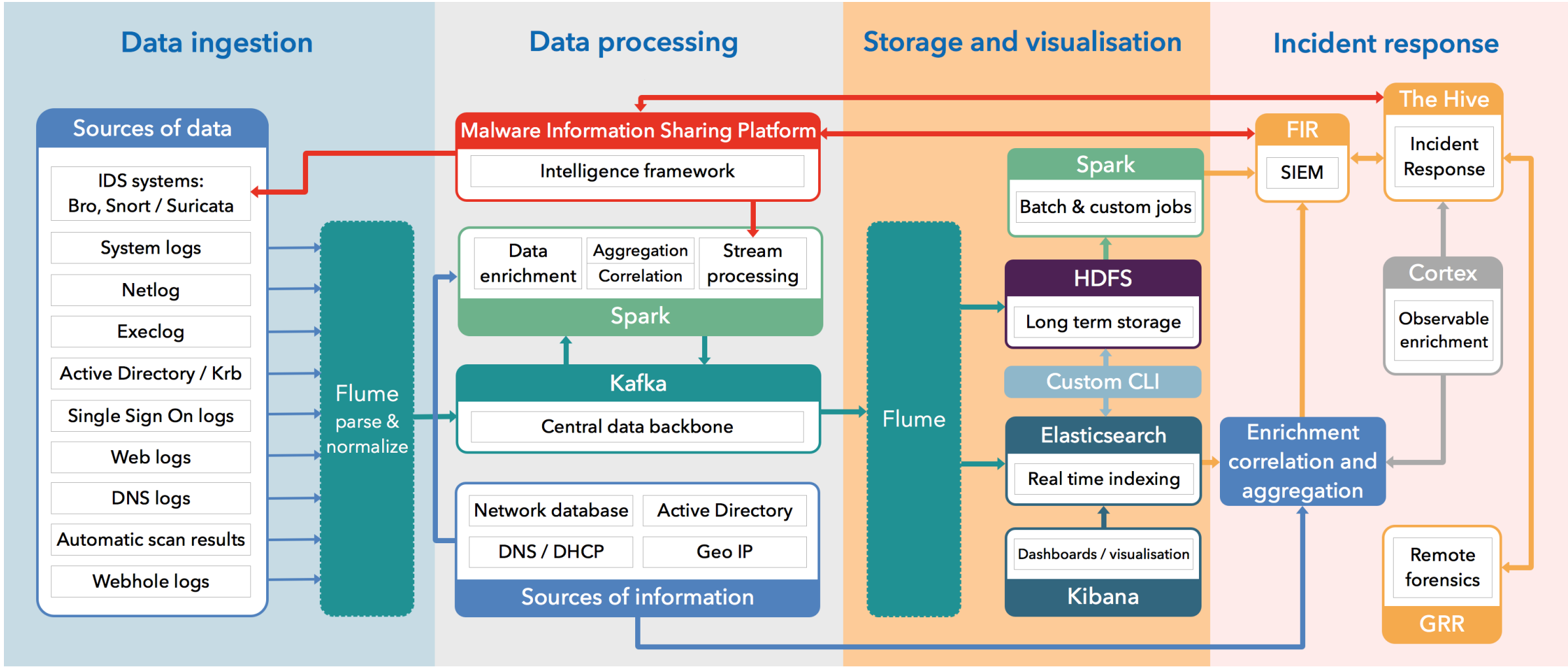
TECHNOLOGY GOALS

- Scale out, not scale up
- Integrated with the rest of the CERN IT ecosystem
- Use of commodity hardware (as much as possible)
- Use of cheap, massively-scalable storage (standard disk arrays)
- Deployment inside OpenStack (whenever possible)
- Configuration management done via Puppet

PRIVACY/SECURITY CONCERNS

- Every component follows strong security requirements:
  - Data transfers encrypted
    - Using TLS
  - Authentication used for all data accesses
    - Mostly Kerberos, password for Elasticsearch
  - Authorization & ACLs
    - Data only accessible to the Computer Security Team & Service Managers
  - Spark master-executors communications are protected

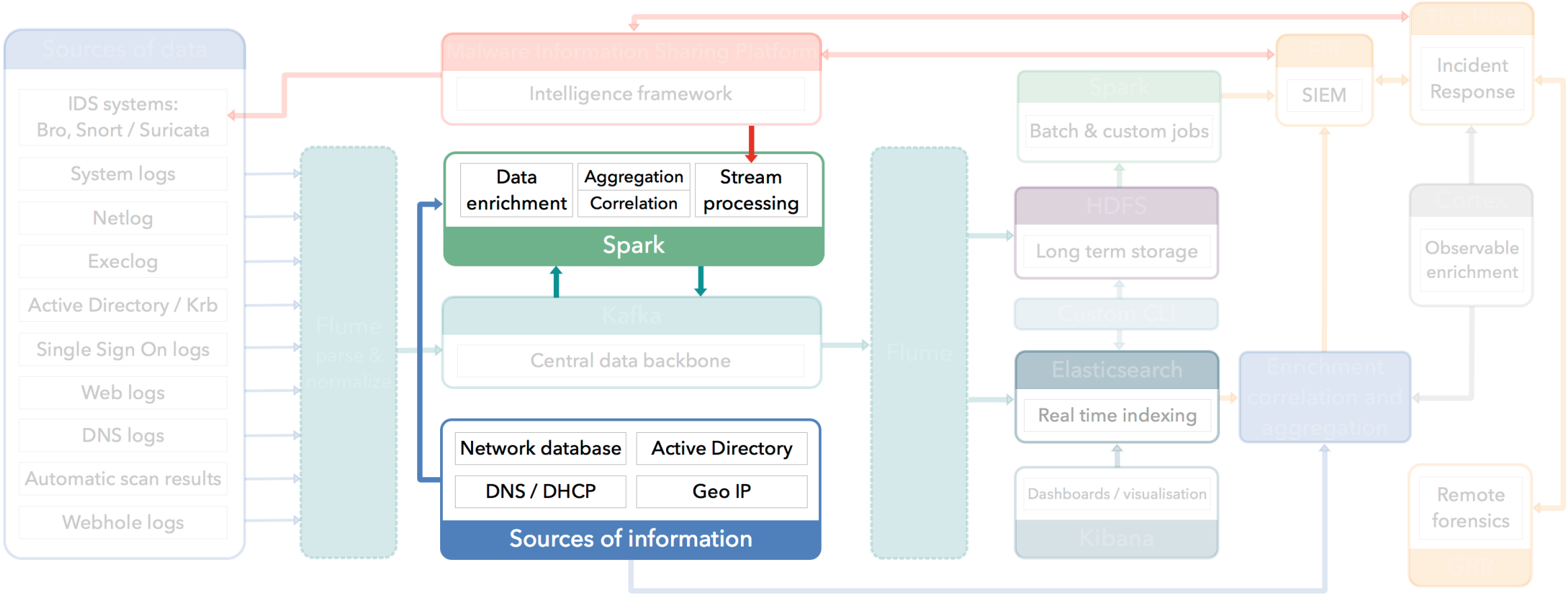
SYSTEM ARCHITECTURE



TECHNOLOGY STACK USED

- **Telemetry Capture Layer:** **Apache Flume**
- **Data Bus (Transport):** **Apache Kafka**
- **Analytics:** **Apache Spark**
- **Long-Term Data Store:** **Hadoop HDFS**
- **Real-Time Index & Search:** **Elasticsearch**
- **Visualisation:** **Kibana & custom CLI**
- **Intrusion Detection:** **Bro & Snort**
- **Web frontends:** **OpenShift**

SPARK STRUCTURED STREAMING



SPARK STRUCTURED STREAMING

- Using Spark 2.3.0 structured streaming
  - Jobs launched and monitored using Nomad
  - Running on one of the central CERN IT Hadoop clusters (YARN)
- Data ingested from Kafka
- Different types of jobs:
  - Data enrichment:
    - DNS (forward and reverse DNS resolutions)
    - GeoIP
  - Intrusion detection:
    - Based on IoCs from MISP
    - Custom, advanced rules
  - Monitoring
  - More to come

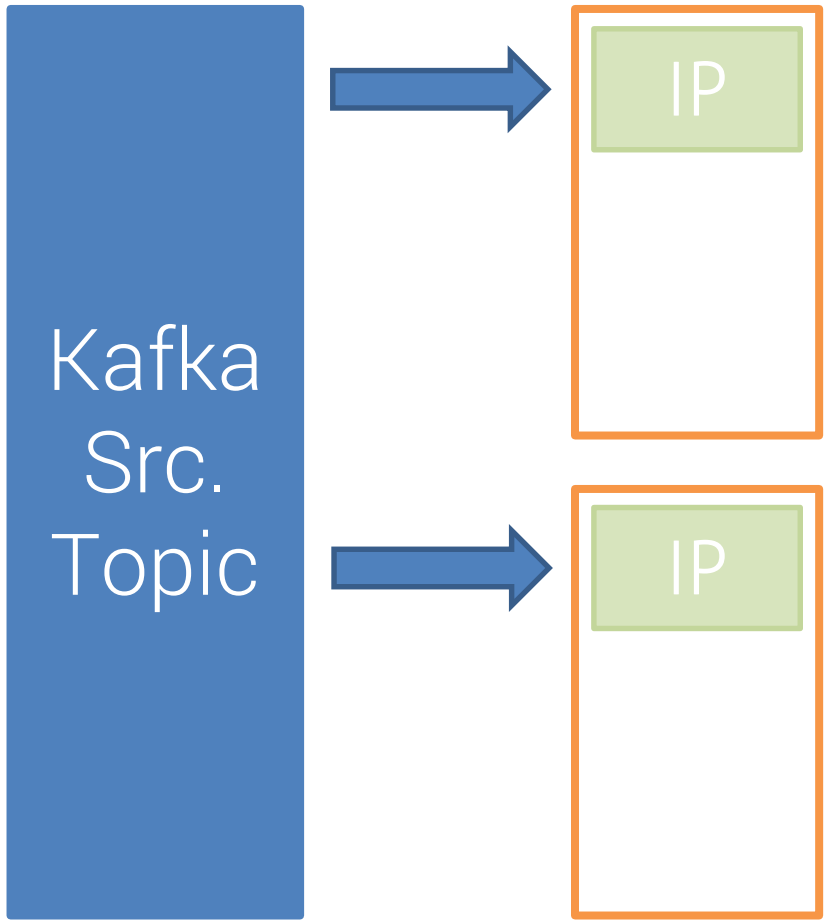
DATA ENRICHMENT

- Very fast, not guaranteed to be 100% accurate
- Two levels of caching
  - 1st level - per executor cache (Storehaus)
  - 2nd level - central Redis database shared between executors
- Currently investigating Apache HBase and Alluxio
  - Would allow moving to one level of caching

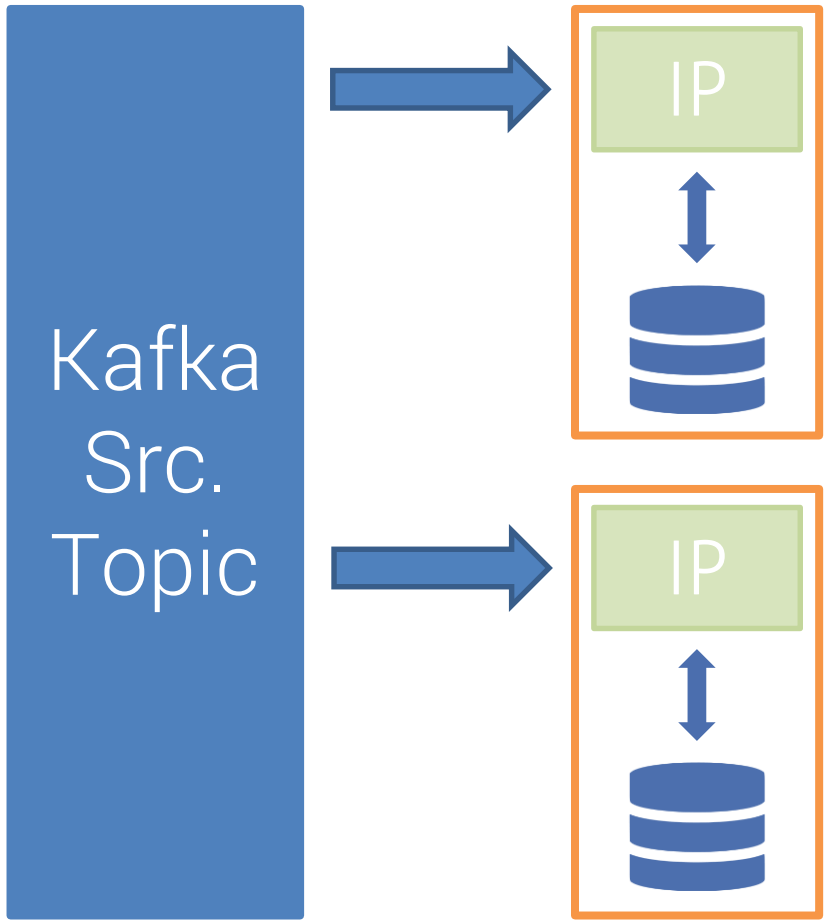
- DNS resolution
  - Average 0,04s / resolution
  - ~1-3s delay for entries that can not be resolved
  - Filtering what messages to enrich

SPARK DNS ENRICHMENT

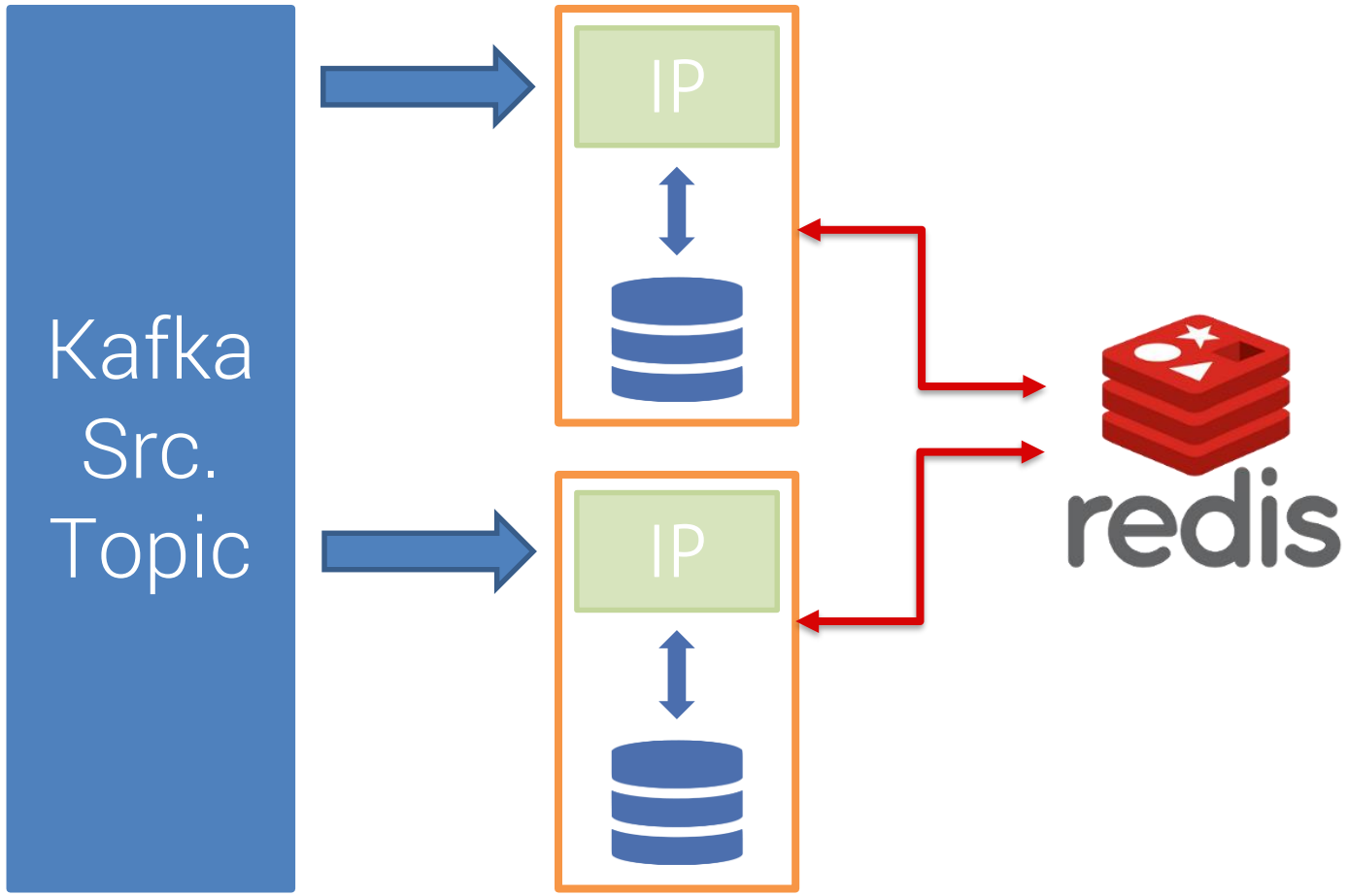
SPARK DNS ENRICHMENT



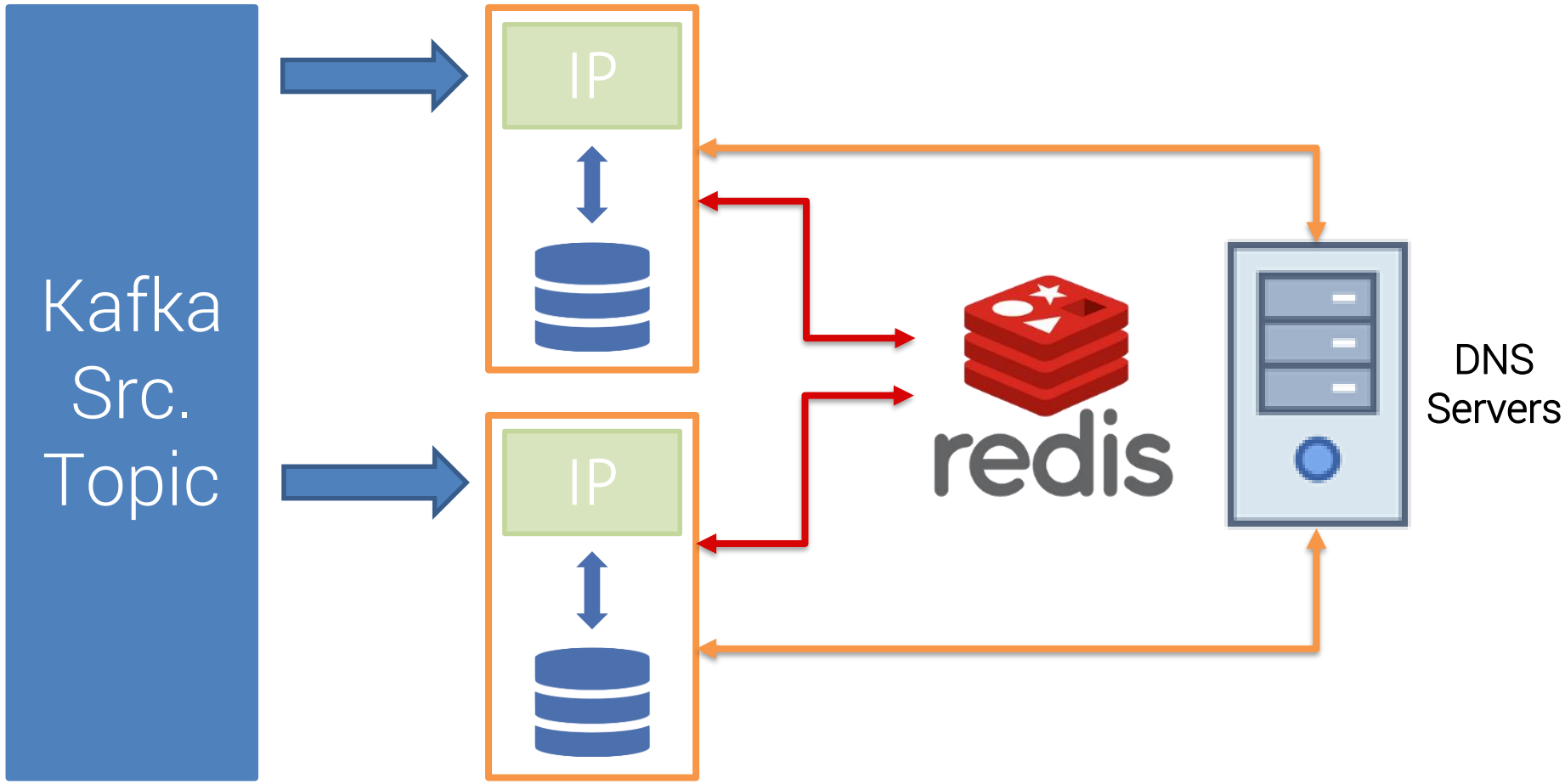
SPARK DNS ENRICHMENT



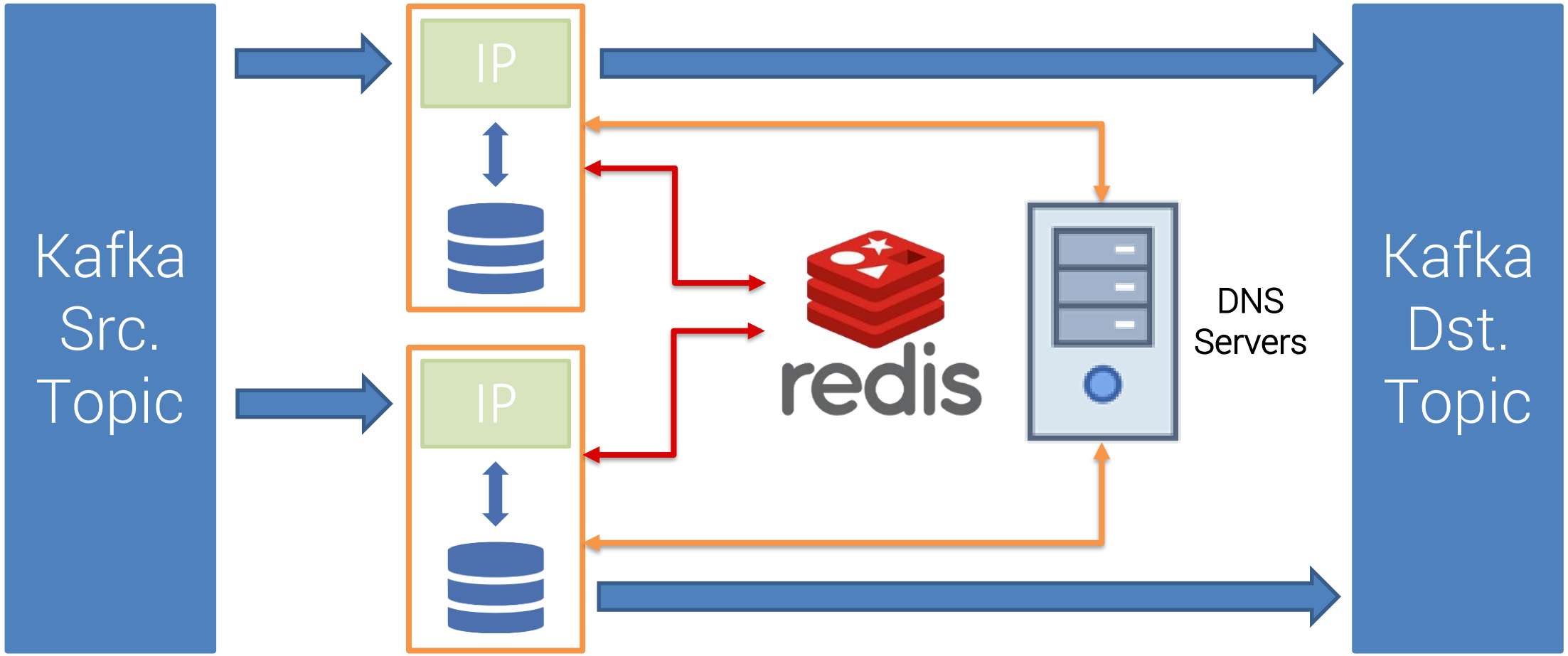
SPARK DNS ENRICHMENT



SPARK DNS ENRICHMENT



SPARK DNS ENRICHMENT



DATA ENRICHMENT - MONITORING

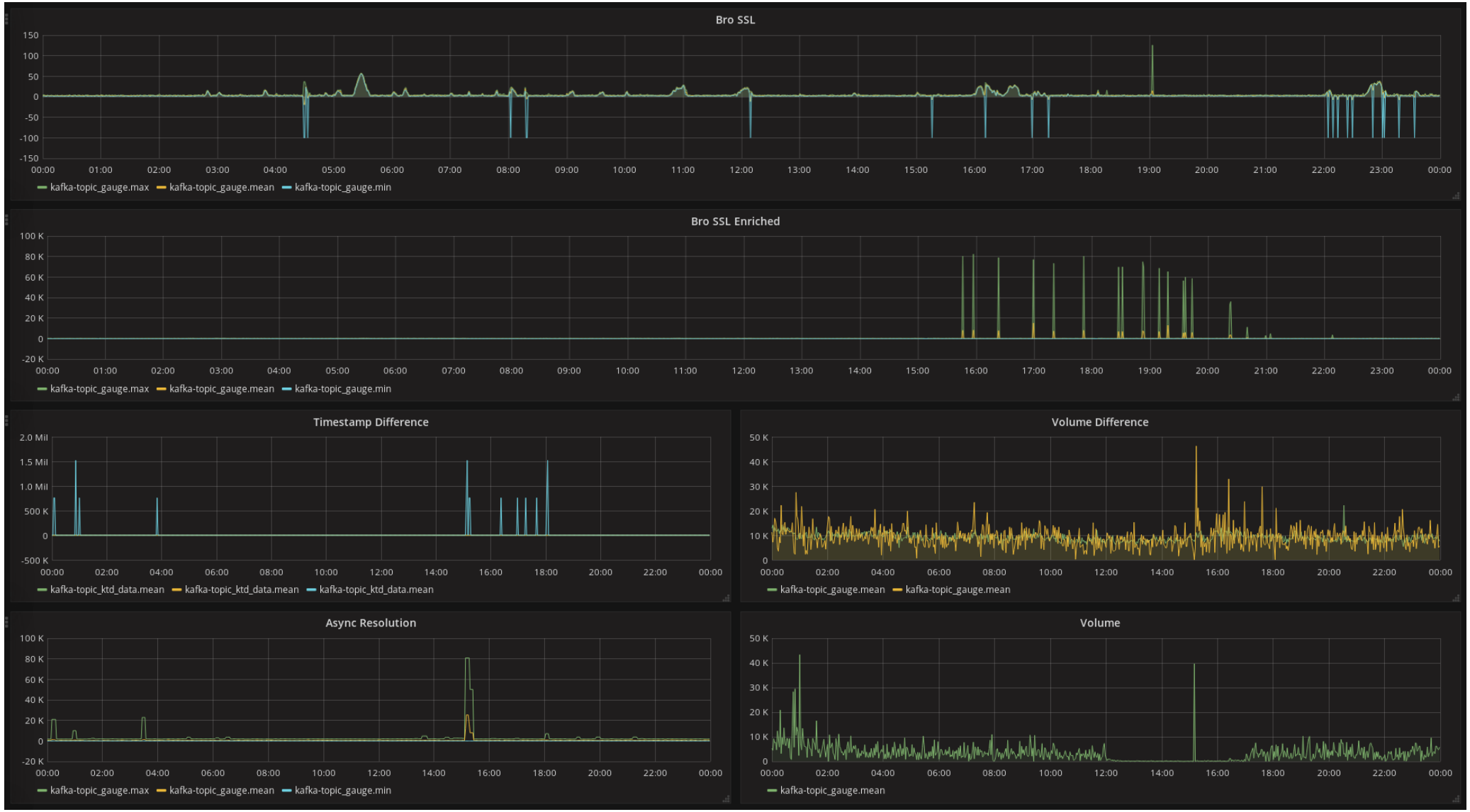
- Problems:
  - Spark does not commit offsets to Kafka (only to HDFS)
  - Spark does not set a consumer group ID
- Solution
  - Nomad job
  - Check most recent timestamp and volume of data in source and destination Kafka topics
  - Future work:
    - Read checkpoints from HDFS
    - Write to __offset topic in Kafka?

MONITORING

- Collectd plugins:
  - Redis upstream
  - Custom developed plugins
    - Consumer groups and Kafka topics

- Ad hoc scripts to produce monitoring of the monitoring data (i.e. inject dummy data)

MONITORING


Bro SSL
150
100
50
0
-50
-100
-150
00:00 01:00 02:00 03:00 04:00 05:00 06:00 07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00 22:00 23:00 00:00
kafka-topic_gauge.max
kafka-topic_gauge.mean
kafka-topic_gauge.min
Bro SSL Enriched
100 K
80 K
60 K
40 K
20 K
0
-20 K
00:00 01:00 02:00 03:00 04:00 05:00 06:00 07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00 22:00 23:00 00:00
kafka-topic_gauge.max
kafka-topic_gauge.mean
kafka-topic_gauge.min
Timestamp Difference
2.0 Mil
1.5 Mil
1.0 Mil
500 K
0
-500 K
00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00 18:00 20:00 22:00 00:00
kafka-topic_ktd_data.mean
kafka-topic_ktd_data.mean
kafka-topic_ktd_data.mean
Volume Difference
50 K
40 K
30 K
20 K
10 K
0
00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00 18:00 20:00 22:00 00:00
kafka-topic_gauge.mean
kafka-topic_gauge.mean
Async Resolution
100 K
80 K
60 K
40 K
20 K
0
-20 K
00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00 18:00 20:00 22:00 00:00
kafka-topic_gauge.max
kafka-topic_gauge.mean
kafka-topic_gauge.min
Volume
50 K
40 K
30 K
20 K
10 K
0
00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00 18:00 20:00 22:00 00:00
kafka-topic_gauge.mean

SPARK BASED INTRUSION DETECTION

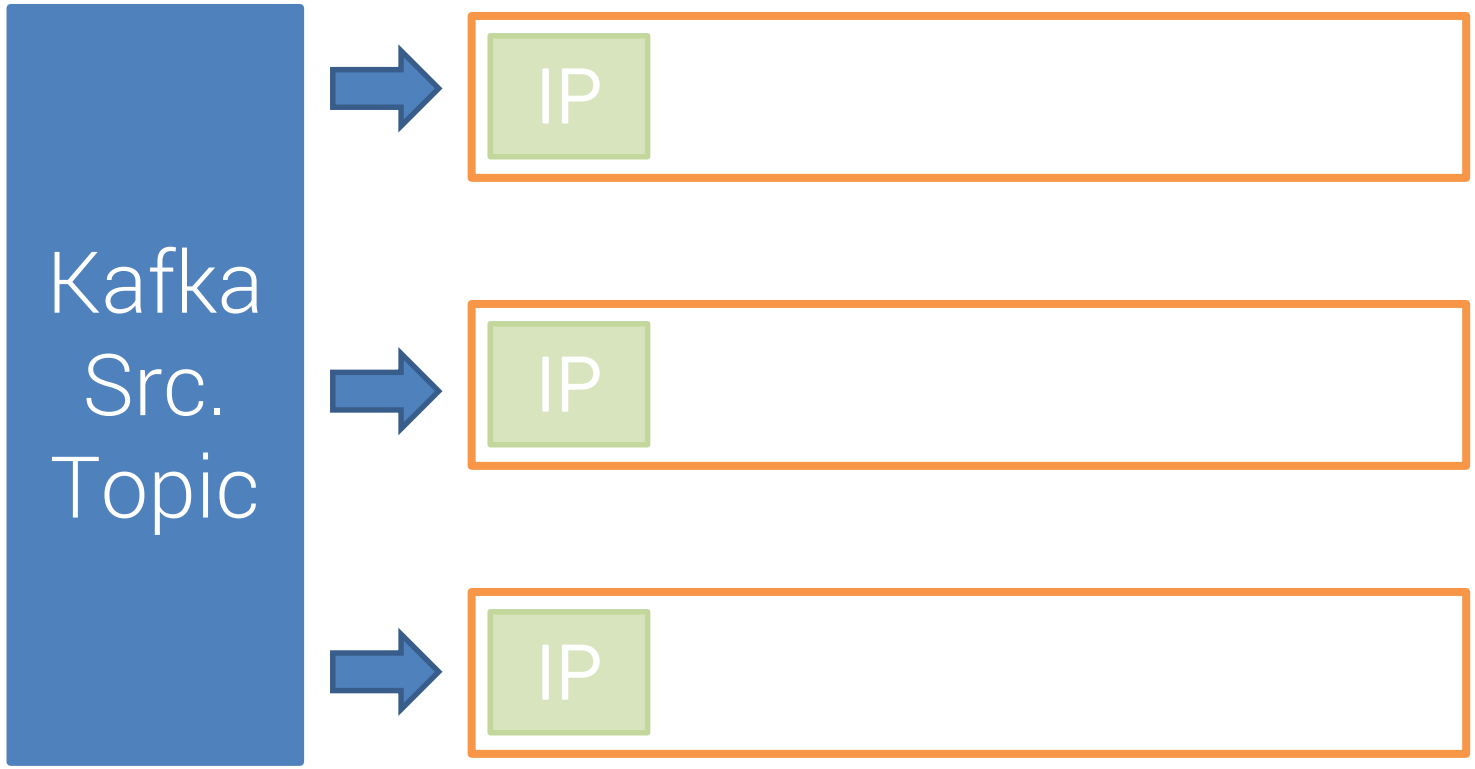
- Large number of IoCs→Dynamic Bloom filter
  - One Bloom filter per type of IoCs

- Bloom filter:
  - Space-efficient data structure to test if an element is member of a set
  - Might return false positives ⟶ Possibly in set
  - Can never return false negatives ⟶ Definitely not in set

SPARK BASED INTRUSION DETECTION

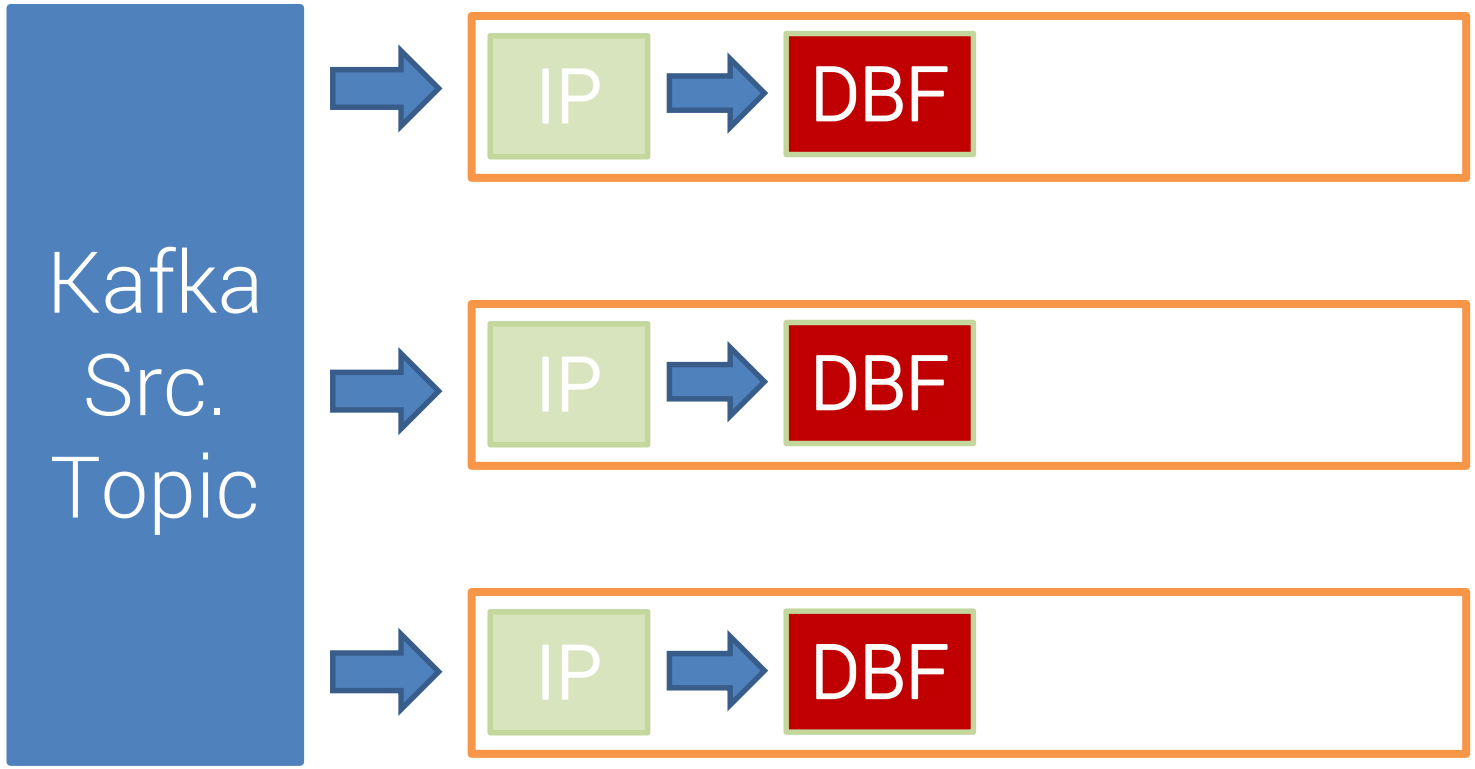
SPARK BASED INTRUSION DETECTION

Parse JSON



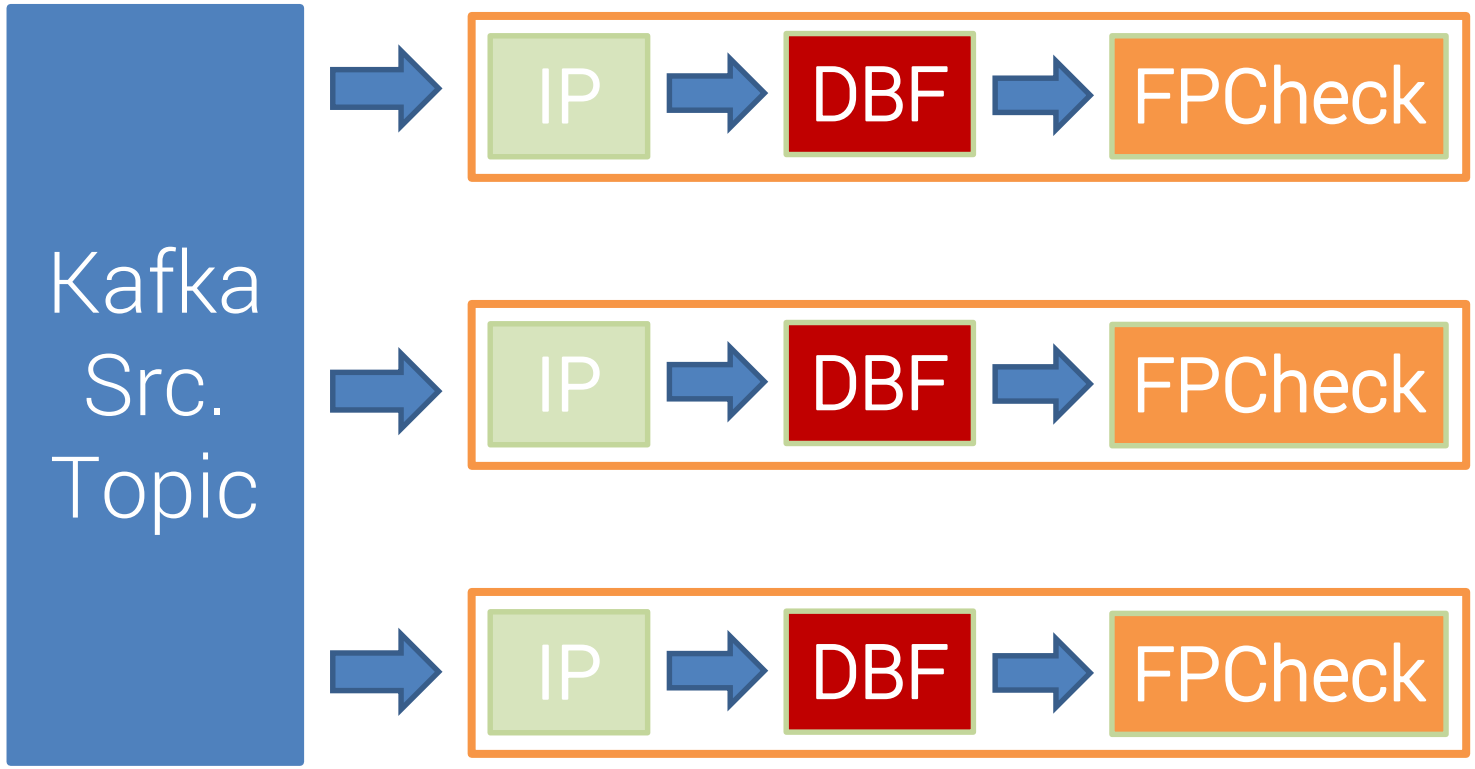
SPARK BASED INTRUSION DETECTION

Query Dynamic Bloom Filter



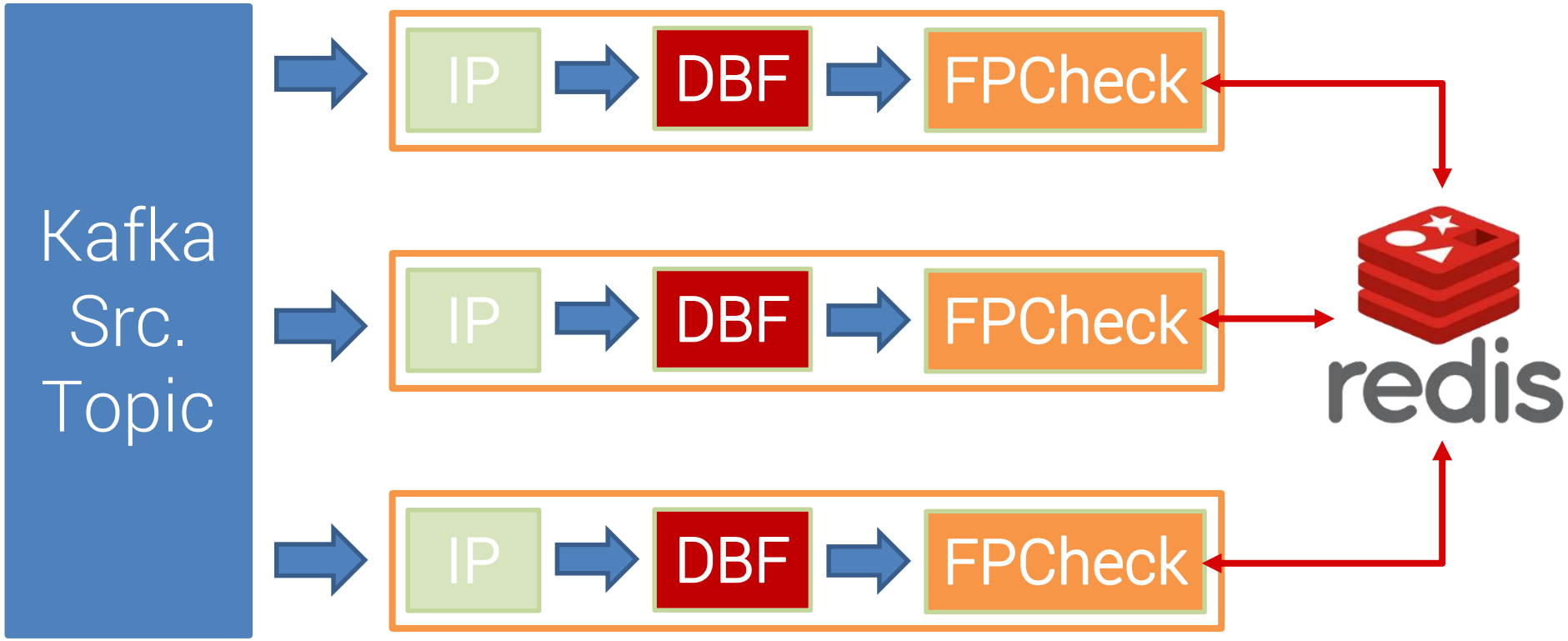
SPARK BASED INTRUSION DETECTION

0 False Negatives



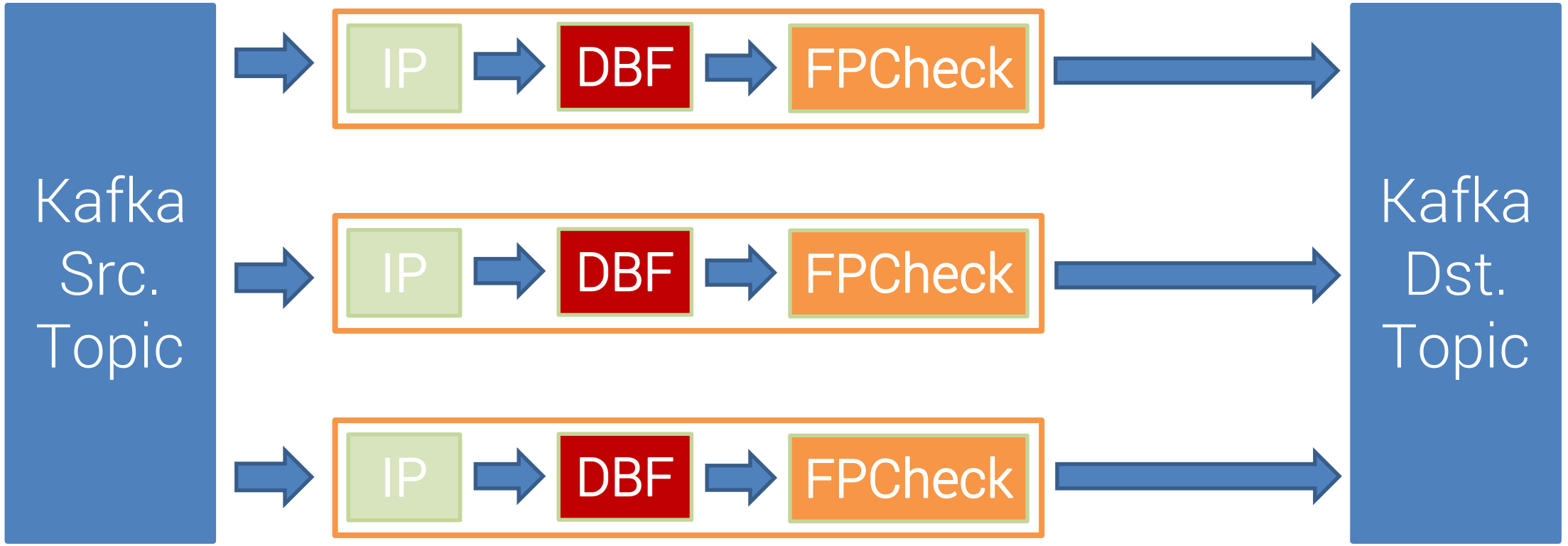
SPARK BASED INTRUSION DETECTION

Query Redis to check for False Positives

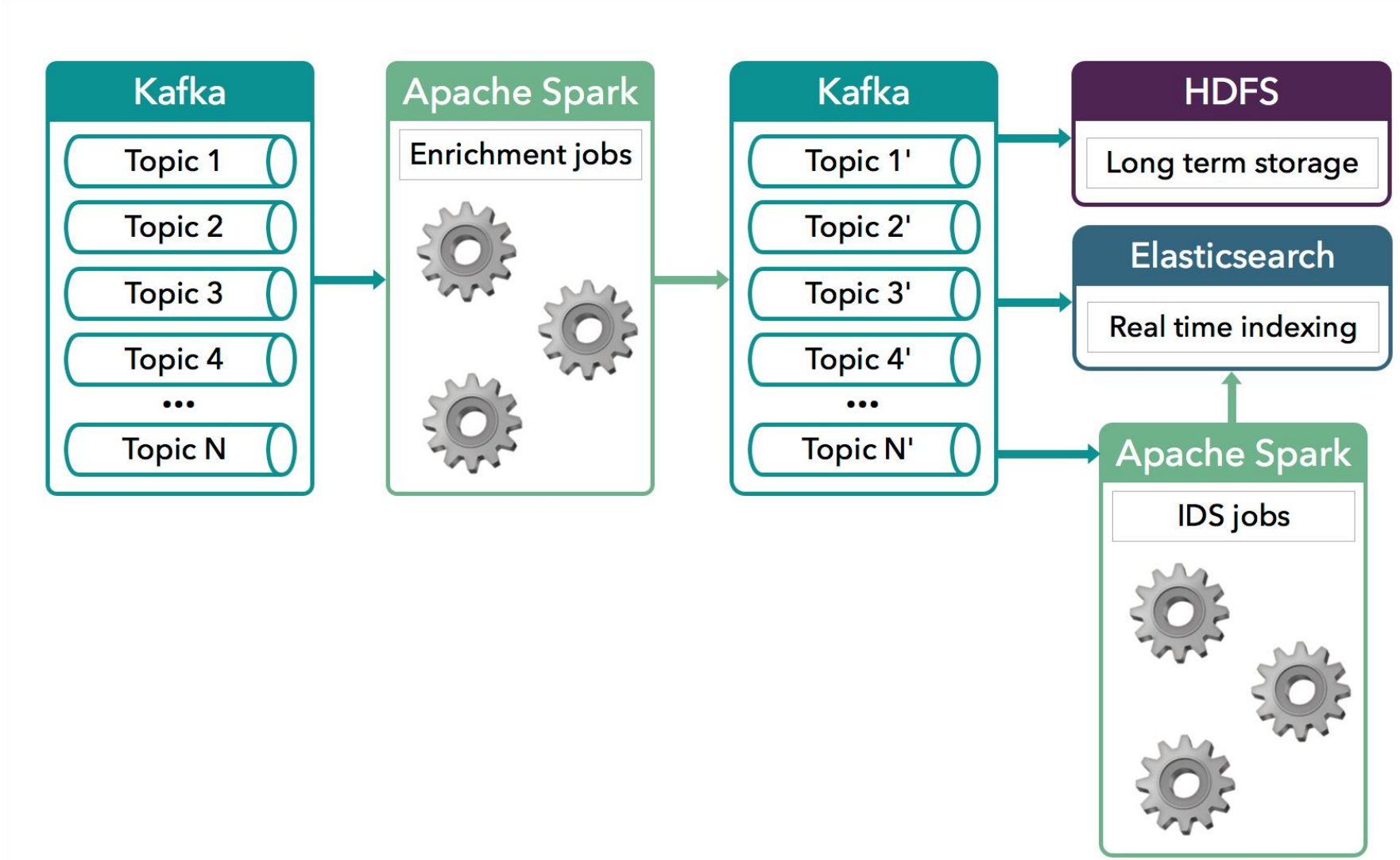


SPARK BASED INTRUSION DETECTION

Push alert to Kafka

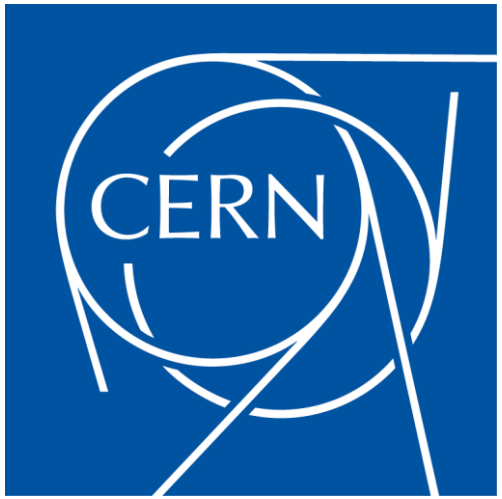


ARCHITECTURE OF SPARK JOBS



CONCLUSION

- Large scale Security Operations Centre
  - Scale out, modular, architecture
  - Different sources of data already, more to be added
- Intensive use of Spark structured streaming
  - Data enrichment
  - Intrusion detection
- Machine learning & anomaly detection capabilities currently being developed


CERN