Instituto de Física de Cantabria

CSIC Consejo Superior de Investigaciones Científicas

UC Universidad de Cantabria

INFN CNAF

Seminar: Streaming data processing with Kafka in Kubernetes

Aida Palacio Hoz (aidaph@ifca.unican.es)
Host Alessandro Costantini

July, 1st 2024
INFN - CNAF

Index

- Contextualizing stream data processing
- Introduce use case I: Batch and stream processing with network traffic
- Stream processing ingestion System: Apache Kafka
  - Benefits, components, architecture and security
- Deploying Kafka in kubernetes
  - Testbed in K8s (WIP)
- Use case II:Accounting records (cASO) processing with Kafka
  - cASO & Kafka integration: Producer and Consumer configuration
  - Monitoring accounting records with Grafana (WIP)

Stream data processing

Stream data processing

- Many different sources generate massive or not massive data: structured, semi-structured, no-structured that needs a refactoring to provide deep insights for the user
- Stream processing bridges the gap between online data and fruitful information

Stream processing is a continuous method of ingesting, processing and analyzing data as it is generated

Main Benefits

- **React to anomaly events in real-time**
- **Adjust operations and resources as the actions occur**

Example in stream processing: Monitoring network traffic streams

Monitoring network traffic

- An Anomaly Instrusion Detection System (**AIDS**) can detect zero-day attacks
  Issue I: can also result in a high false positive and false negative rate

  1 Collect and store the traffic for the application of ML techniques

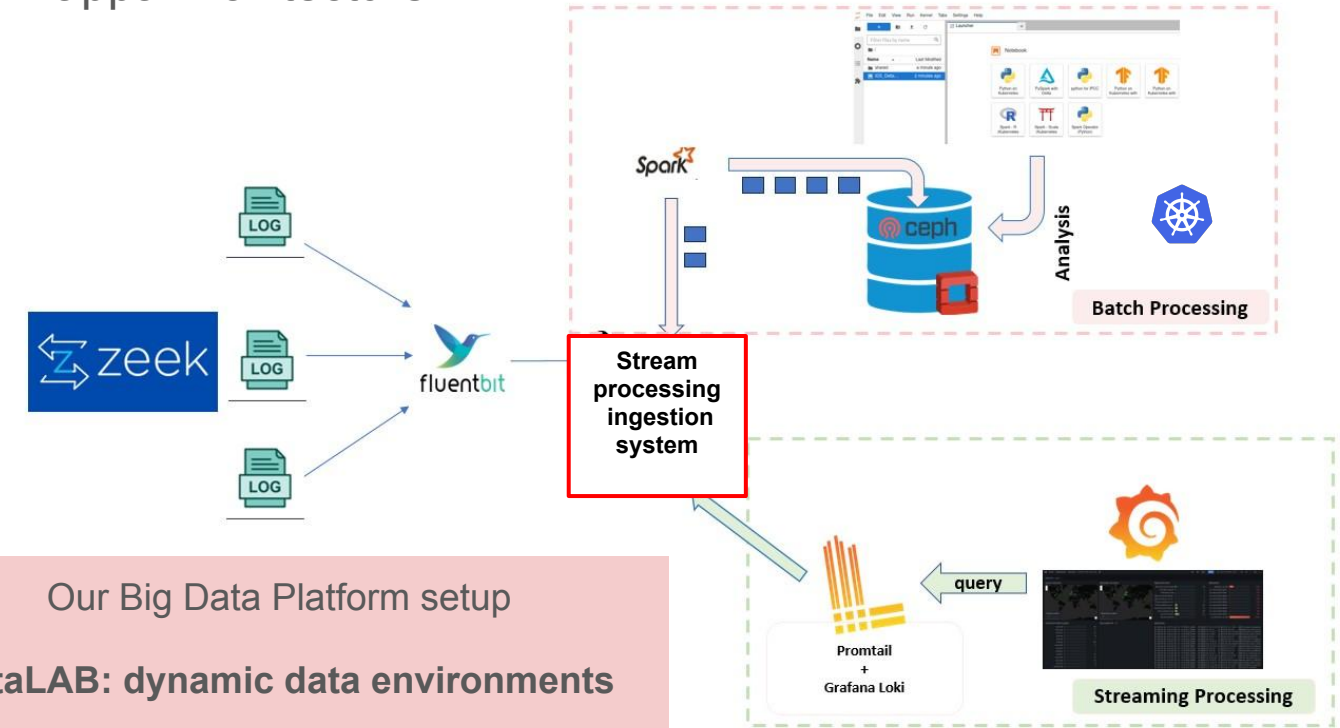
- Issue II: An **AIDS** like zeek generates ~ 100.000 events per day in a “small” infrastructure

  2 Automatize the ingestion and storing tasks with high-performance tehnologies



Platform Architecture for monitoring Network traffic

Based on Kappa Architecture

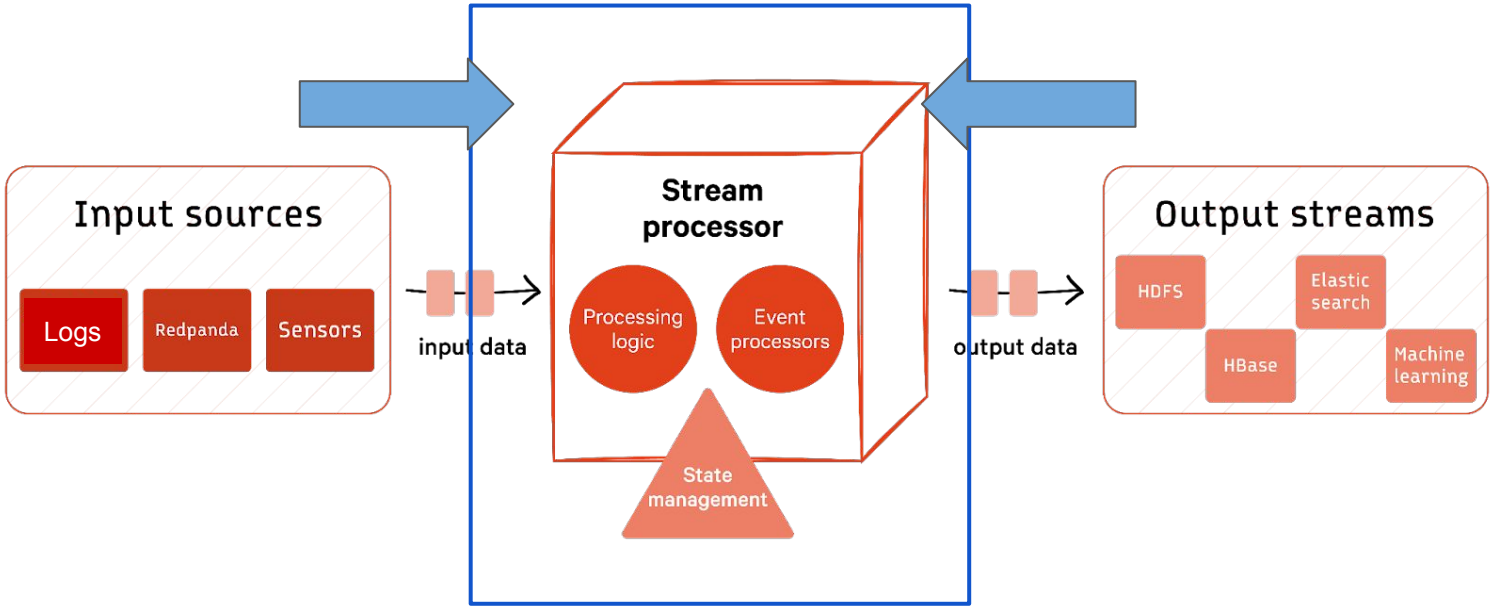


Our Big Data Platform setup

DataLAB: dynamic data environments

Key in Stream processing: **Event-driven architecture**

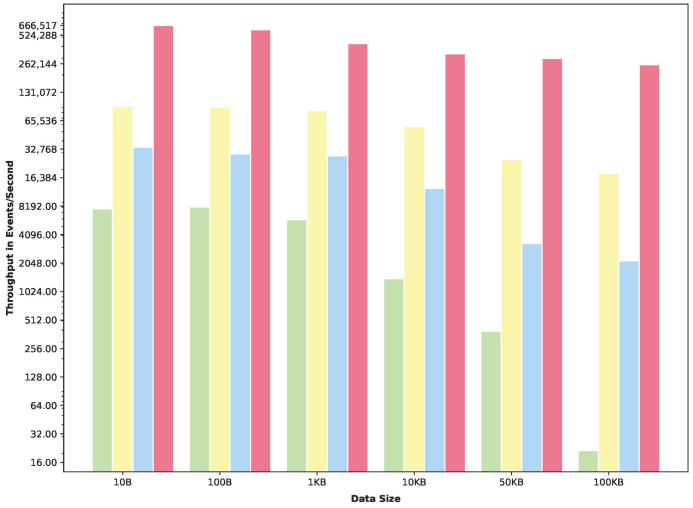
Event-driven architecture is the design of a platform using technologies for handling event data, enabling real-time processing and response to events as they occur



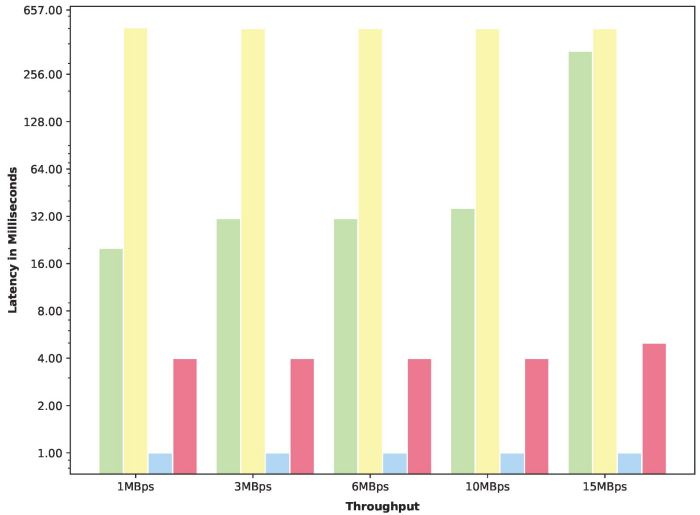
Stream processing ingestion system: Apache Kafka

Why using kafka for stream processing?

[Benchmarking Message Queues] in a single laptop with 16GB RAM and 512 GB SSD



	10B	100B	1KB	10KB	50KB	100KB
Artemis	7652.99	8006.72	5872.98	1400.05	389.26	21.34
RabbitMQ	92,658	91,326	83,400	56,692	25,342	18,221
Redis	34,347	29,111	27,734	12,624	3298.67	2157.69
Kafka	666,517	599,794	427,361	333,145	298,590	255,285



	1MBps	3MBps	6MBps	10MBps	15MBps
Artemis	20.0	31.0	31.0	36.0	360.0
RabbitMQ	507.0	501.0	500.0	501.0	500.0
Redis	1.0	1.0	1.0	1.0	1.0
Kafka	4.0	4.0	4.0	4.0	5.0

Streaming Ingestion System: Apache Kafka

Apache Kafka is a **publish-subscribe message processing system of stream events** where one event can be a type of action, an incident, a change in a system, etc.

- Kafka is based on the concept of **commit logs**, splitting the data into partitions for scaling-out systems. The events are modelled as key/value pairs: internally, they are a sequence of bytes, but externally are usually JSON, JSON schema or Avro.
- The translation between language and bytes is called **serialization** and **deserialization**.
- **Log Aggregation**: It can serve as a centralized log aggregation system for applications and microservices.

Benefits of Apache Kafka

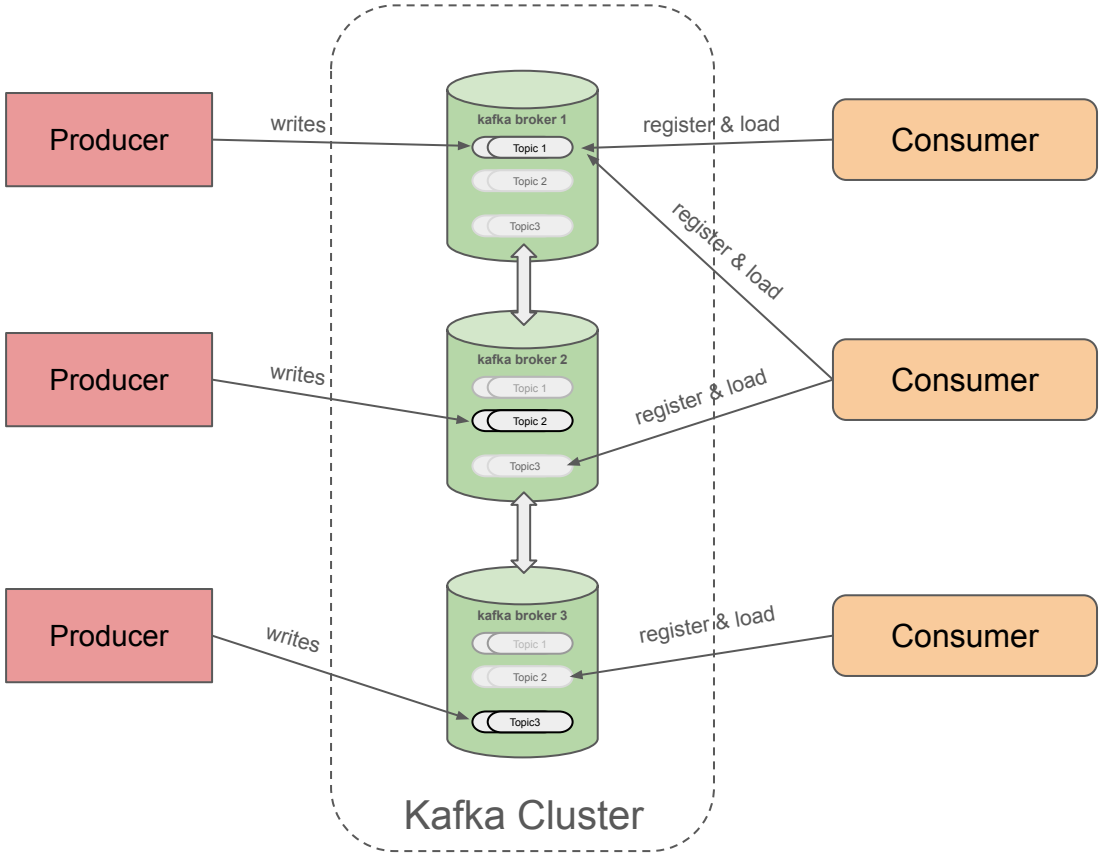
- **Log aggregation**: integrate multiple data source in the same centralized system: Collect banch of **different type of data**
- **Stream processing**: ingest, store and process streams as the data is generated, at any scale
- **Distributed system: fault tolerance, resilience and scalability**
- **Data persistence**: store data at disk until it is served by the subscribers
- Consume “kafka” data from different applications

Examples: Monitoring operational data such as logs, anomaly detection, IoT, energy systems, hospitals history data, etc.

Apache Kafka: Components

- **Broker:** the main servers responsible for storing data and managing the requests.
  - Multiple brokers compose a cluster where each broker can host one or more partitions depending on the leadership policy
  - Communicate with each other for data synchronization and leader election
- **Topic:** the object where data is stored, like a “queue”.
  - Producer publish data to topics and consumers load from them
  - Represents the factor of replication
  - Distributes often data among multiple partitions for scalability and redundancy
- **Partition:** The unit of parallelism in Kafka
  - Ordered and immutable sequence of records
  - Allows consumers to process data concurrently
- **Producer:** responsible of writing data to topics
  - Can also specify keys to control how data is distributed among partitions
- **Consumer:** “application” which load data from topics
  - Can specify offsets to track their progress reading messages
- **Data Storage:** “local” broker space where the partitions are stored

Architecture Kafka architecture



Apache Kafka: Security

Not recommended: only for testbeds: PLAINTEXT

The communication between **1. brokers** or **2. broker - client** might be:

- Authenticated: with SSL or SASL
  - SASL mechanisms:
    - SASL/GSSAPI (Kerberos)
    - SASL/PLAIN: username - password
    - SASL/SCRAM-SHA-256 o SASL/SCRAM-SHA-512: username encrypted
    - SASL/OAUTHBEARER: OAuth2 but needs additional implementation
- Encrypted: Only SSL

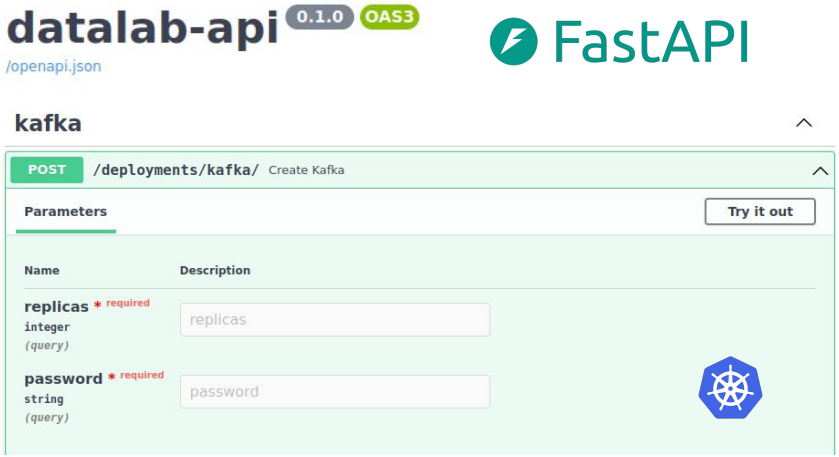
BEST Solution: SASL (Auth) + SSL (Encrypt) = SASL_SSL protocol

Deploying Kafka in Kubernetes

Kafka in Kubernetes Testbed (WIP)

Motivation

- Integrate easily with our Big Data Platform (called DataLAB) based on Kubernetes
- Interesting to automatize the deployment of the Kafka cluster: define number of replicas **on-demand** deployment for specific use cases



Work in progress:

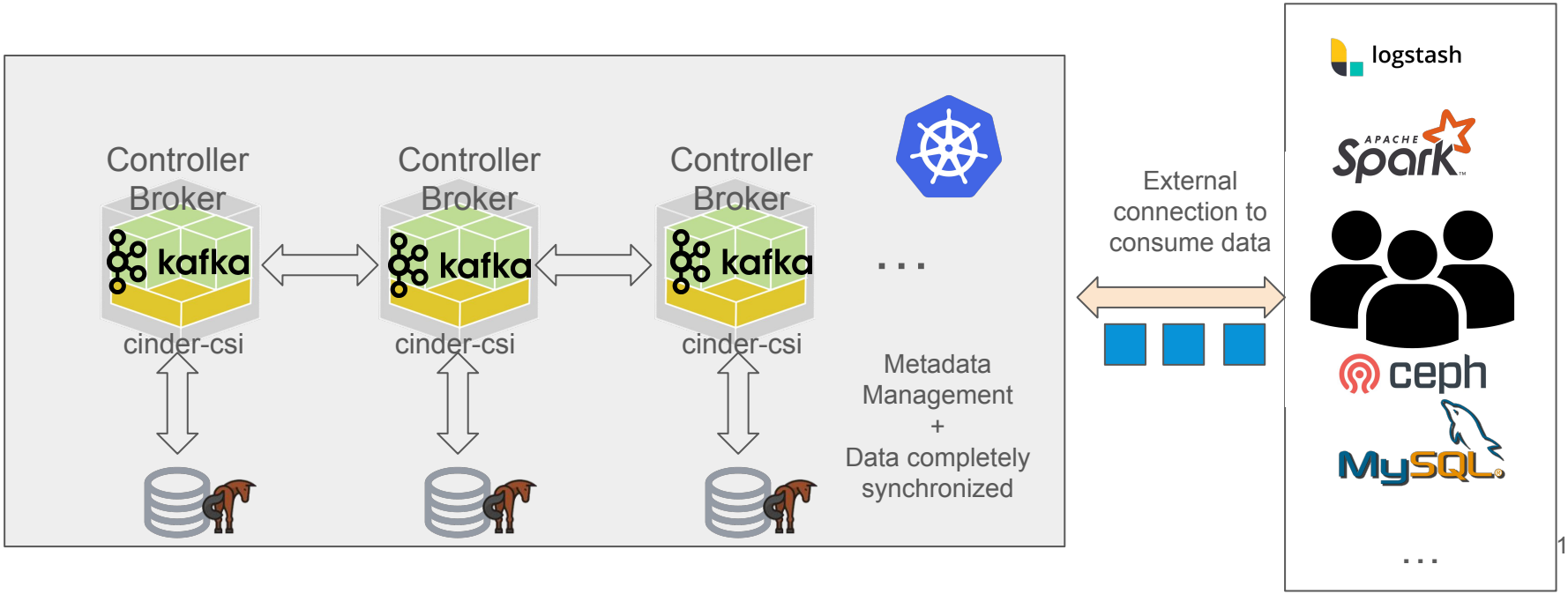
- Kafka Cluster testbed deployed as statefulset object
  - Works if **manually predefine the amount of replicas**
  - Many issues to scale up the cluster turning up "**dynamic**" quorum voters

Deploying Apache Kafka: key points

- **Replicated mode** with data sync among the multiple brokers
- **Each broker is exposed publicly** to be reachable from external applications
- Scalable kafka can mean a cluster composed by few or lot of instances
  - Issue I: **one public-ip per server**
- Related to Issue I: Running behind a proxy
  - Issue II: **Kafka traffic mode tcp** instead of the common http in the proxy “needs tricky config”
- Security: Kafka native with **SASL_SSL protocol** instead of in the proxy
- **Persistence space** via cinder volumes

Our setup under K8s: traditional approach

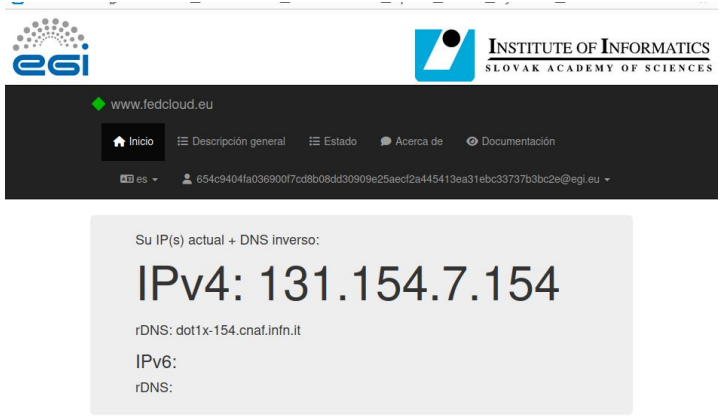
In **replicated** mode: each broker is a container, a virtual machine or a physical machine susceptible to failure (**replica**)



Our setup under K8s: Solving Issue I

Each broker reachable from outside

Dynamic DNS in the EGI Federated Cloud provides a unified, federation-wide dynamic support for VMs in EGI infrastructure.



www.fedcloud.eu — the Dynamic DNS service for EGI Federated cloud

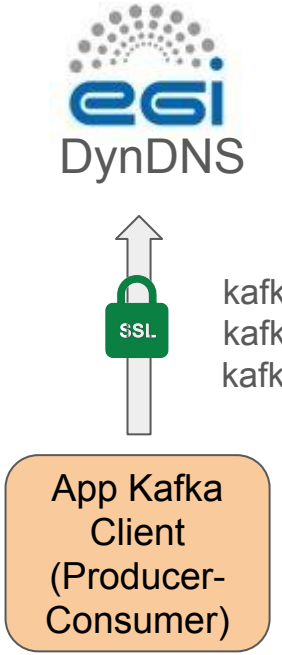
EGI users can register DNS hostnames behind a given domain name and assign them to public IPs of their servers.

Update DNS entries with:

curl
https://**kafka.**datalab.ifca.es:7LSgckXsac@nsupdate.fedcloud.eu/nic/update
?myip=193.146.75.243 ← Proxy IP

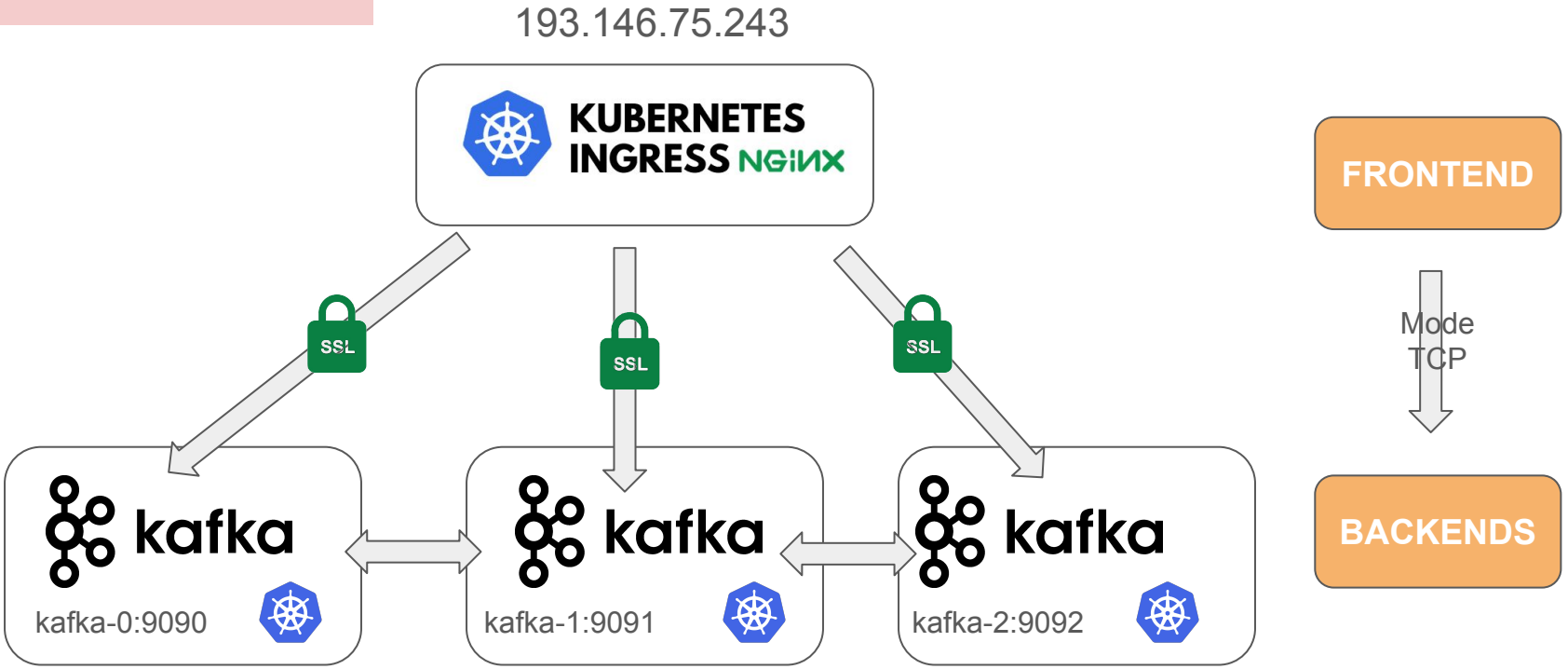
EGI DynDNS for Kafka cluster

Host Comentario	Disponible	Fallos C / S	Direcciones IPv4 (última actualización)
api.datalab.ifca.es	sí	1 / 0	193.146.75.243 (hace 2 meses, 2 semanas, TLS)
ids.datalab.ifca.es	sí	2 / 0	193.146.75.243 (hace 1 año, TLS)
ipcc.datalab.ifca.es	sí	0 / 0	193.146.75.243 (hace 1 año, TLS)
kafka.datalab.ifca.es	sí	1 / 0	193.146.75.243 (hace 2 meses, 2 semanas, TLS)
longhorn.datalab.ifca.es	sí	0 / 0	193.146.75.243 (hace 2 meses, 2 semanas, TLS)



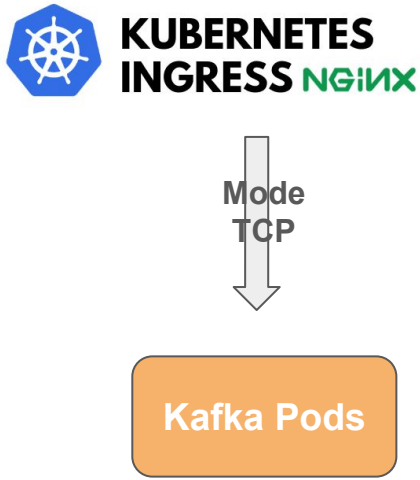
Our setup: Solving Issue II - Ingress-nginx controller

HA + tcp mode support



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ingress-nginx-tcp
  namespace: ingress-nginx
data:
  "9090": kafka/kafka-headless:9090
  "9091": kafka/kafka-headless:9091
  "9092": kafka/kafka-headless:9092
```

ConfigMap in the ingress-nginx controller to redirect external TCP traffic to internal broker ports



KAFKA_LISTENERS=SASL_SSL://0.0.0.0:909$KAFKA_NODE_ID

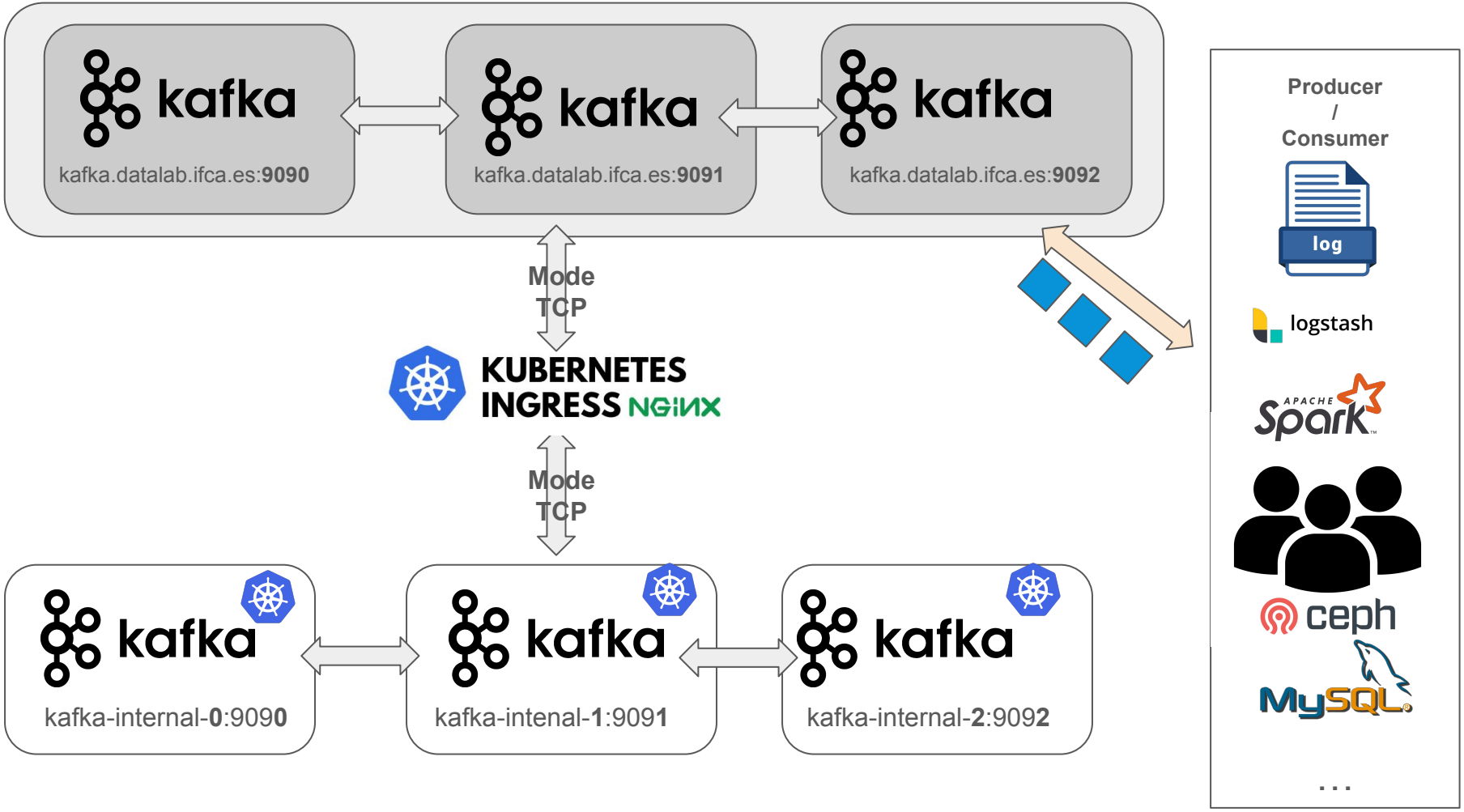
Manage Security at Kafka: SASL + SSL

1. **SSL: multidomain CA cert and Certs + Keys for each kafka instance *.datalab.ifca.es**

2. **SASL: Current PLAINTEXT -> Testing OauthBearer (Merge with the Datalab AAI)**

```
controller.quorum.voters=0@kafka-0.kafka-headless.kafka.svc.cluster.local:29093,1@kafka-1.kafka-headless.kafka.s
vc.cluster.local:29093,2@kafka-2.kafka-headless.kafka.svc.cluster.local:29093
listeners=SASL_SSL://0.0.0.0:9090,CONTROLLER://172.16.44.141:9094
advertised.listeners=SASL_SSL://kafka.datalab.ifca.es:909$KAFKA_NODE_ID
inter.broker.listener.name=SASL_SSL
```

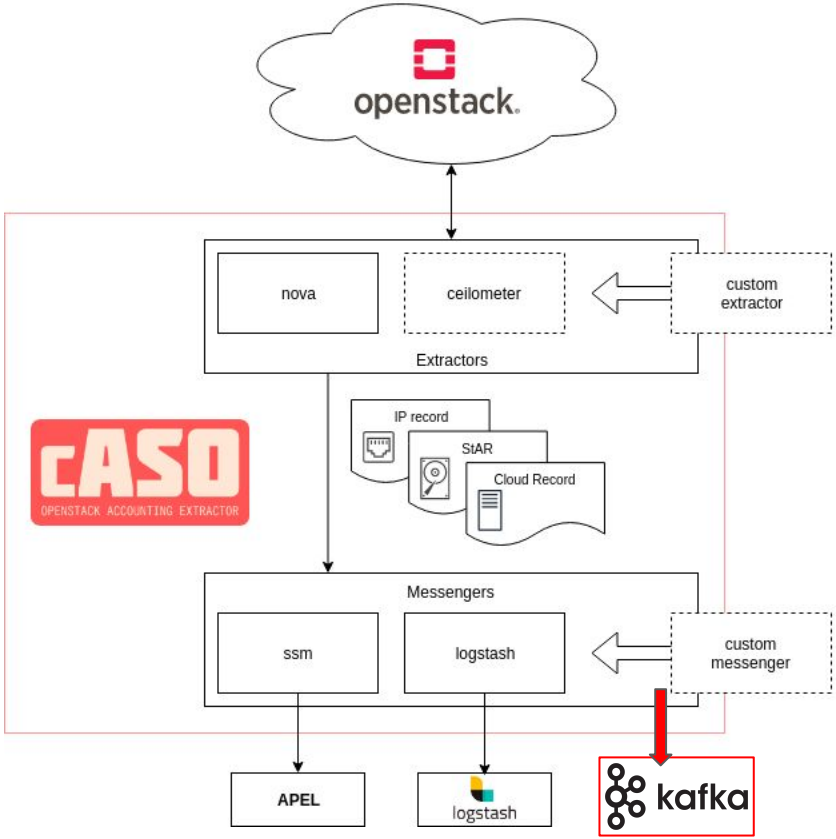
kafka-broker.properties


kafka
kafka.datalab.ifca.es:9090
kafka
kafka.datalab.ifca.es:9091
kafka
kafka.datalab.ifca.es:9092
Mode
TCP
KUBERNETES
INGRESS NGINX
Mode
TCP
kafka
kafka-internal-0:9090
kafka
kafka-intenal-1:9091
kafka
kafka-internal-2:9092
Producer
/
Consumer
log
logstash
APACHE
Spark
ceph
MySQL
...

Accounting records (cASO) processing with Kafka

w/@Alessandro Costantini

Accounting records (cASO) processing with Kafka



Issues with current messengers:

- Lack of information in the APEL accounting: GPU and Storage accounting
- Real-time streaming data

New custom messenger Kafka for sending accounting records and persisting data

cASO & Kafka Integration: Kafka configuration

[...]

listener.name.**sasl_ssl**.**plain**.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \

  username="admin" \

  password="4dmin-s3cr3t" \

  user_admin="4dmin-s3cr3t" \

  user_testuser="t3st123456789" \

  user_casouser="H2T}f]H12345";

[...]

kafka-broker.properties

cASO & Kafka Integration: cASO configuration (**Producer**)

New Messenger in Kafka language means a new producer that collects the records after the caso extraction and “be stored” until a third-party application, such as Logstash or Grafana Loki consumes those records

/etc/caso/**caso**.conf

```
messengers = kafka

[kafka]

brokers =
kafka.datalab.ifca.es:9090,kafka.datalab.ifca.es:9091,kafka.datalab.ifca.es:9092

topic = test-caso

username = casouser

password = H2T}f]H12345
```

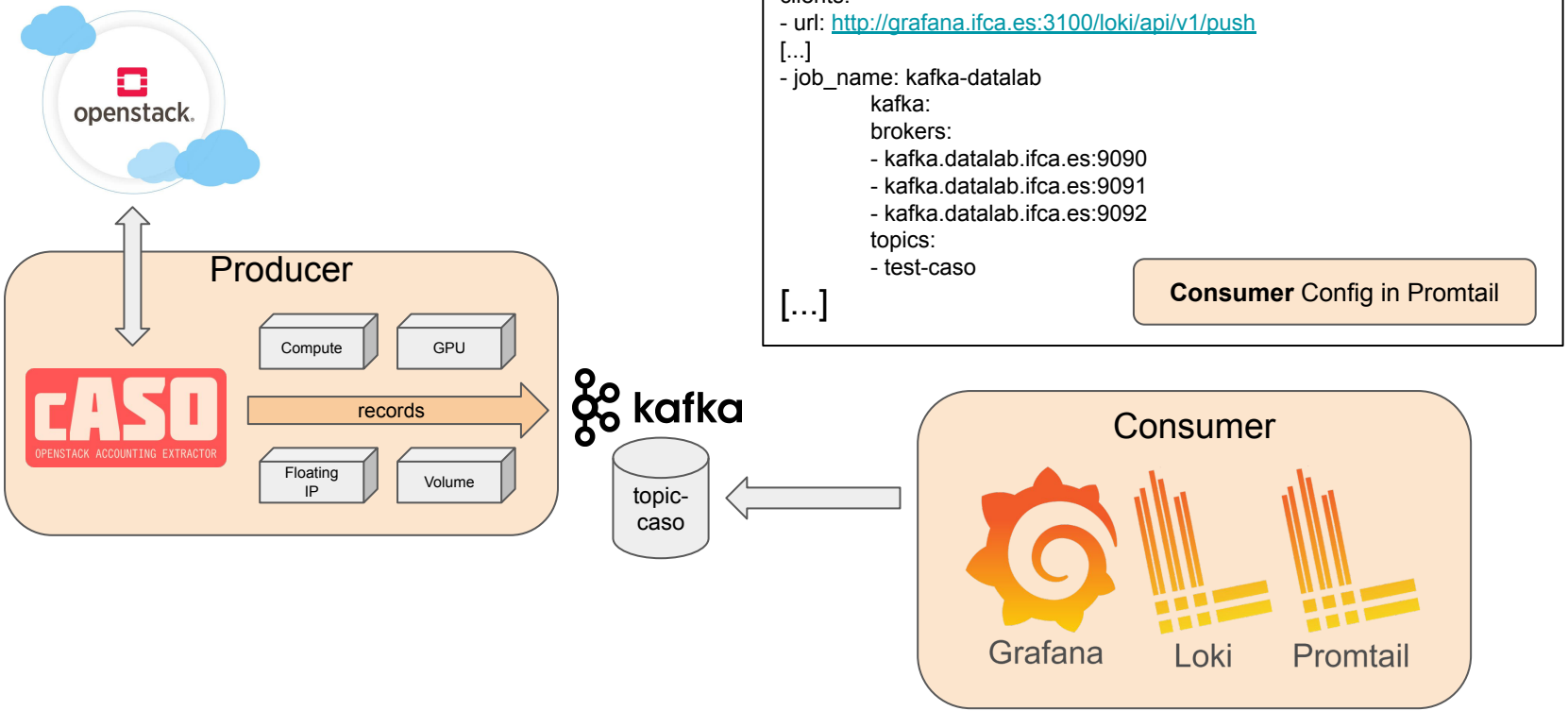
cASO & Kafka Integration: cASO configuration (Producer)

Kafka Messenger

```
self.brokers = CONF.kafka.brokers
self.topic = CONF.kafka.topic
self.username = CONF.kafka.username
self.password = CONF.kafka.password
[...]
producer = Producer(**conf)
"""Push records to kafka"""
for record in all_caso_records:
    #serialization of record
    rec=record.serialization_message()
    try:
        producer.poll(0)
        producer.produce(self.topic, value=json.dumps(rec).encode('utf-8'),
                         callback=delivery_report)
```

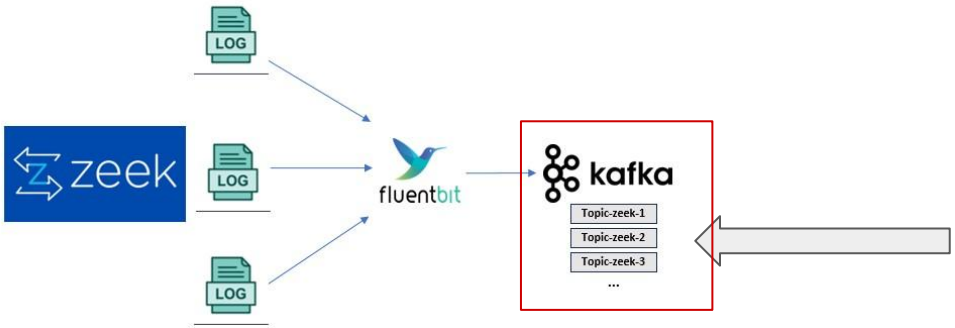
Producer Config in cASO

Consume cASO records from Kafka

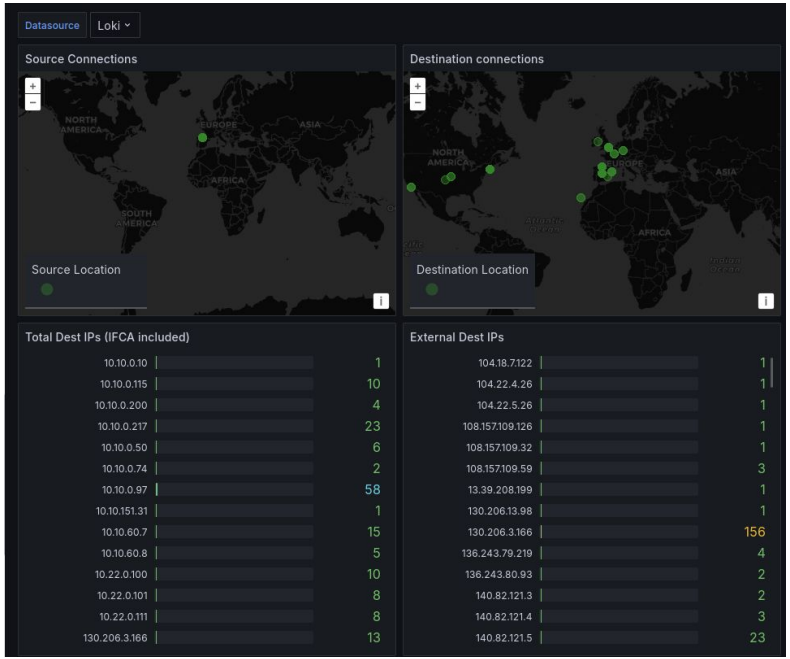


Monitoring Accounting data with Grafana (WIP)

Example Use case I: Monitoring Network traffic events from an AIDS



Grafana Loki Promtail



Instituto de Física de Cantabria

CSIC Consejo Superior de Investigaciones Científicas

UC Universidad de Cantabria

INFN CNAF

Seminar: Streaming data processing with Kafka in Kubernetes

QUESTIONS?

Aida Palacio Hoz (aidaph@ifca.unican.es)
Host Alessandro Costantini

Thanks to:
- SDDS department
- BDP working group
- Luca Dell'Agnello for hosting me :)

July, 1st 2024

INFN - CNAF