B0M33BDT – How to get/put data to Hadoop, NiFi, Kafka, Architecture patterns

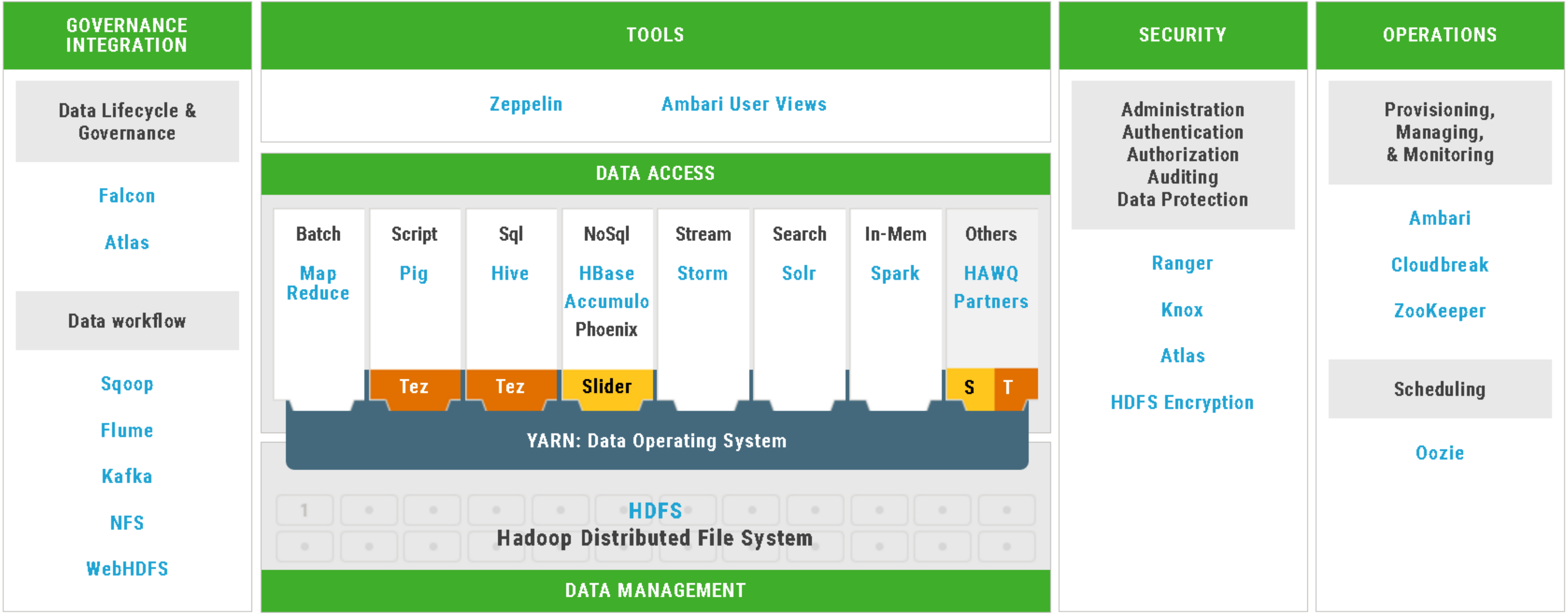
Josef Vonášek

1. Listopadu 2023

Getting Data

Hadoop ZOO

PROFINIT



How to put data to hadoop

PROFINIT

- Webservice – webHDFS
- SFTP & hdfs command
- NFS gateway
- Sqoop – for databases – jdbc
  - Full, increcemtal load
  - configuration via metadata
- Flume (messages) – deprecated
- Flink - streaming

Commercial solutions

- Commercial solutions
- Informatica, Talend, Oracle, …
- Typically as a bundle to something else

- Spark and JDBC?
  - Yes, it works as well
  - But
    - Performance issues
    - Error handling

Apache Nifi

NIFI (Niagara Files, Hortonworks Data Flow (HDF))

PROFINIT

- NiFi is a processing engine designed to manage a continuous flow of information as a series of events in an ecosystem
- Visual creation and management of directed graphs of processors
- Highly concurrent model without a developer having to worry about the typical complexities of concurrency
- Natural error handling
- Cohesive and loosely coupled components which can then be reused

APACHE nifi

Naming Convention

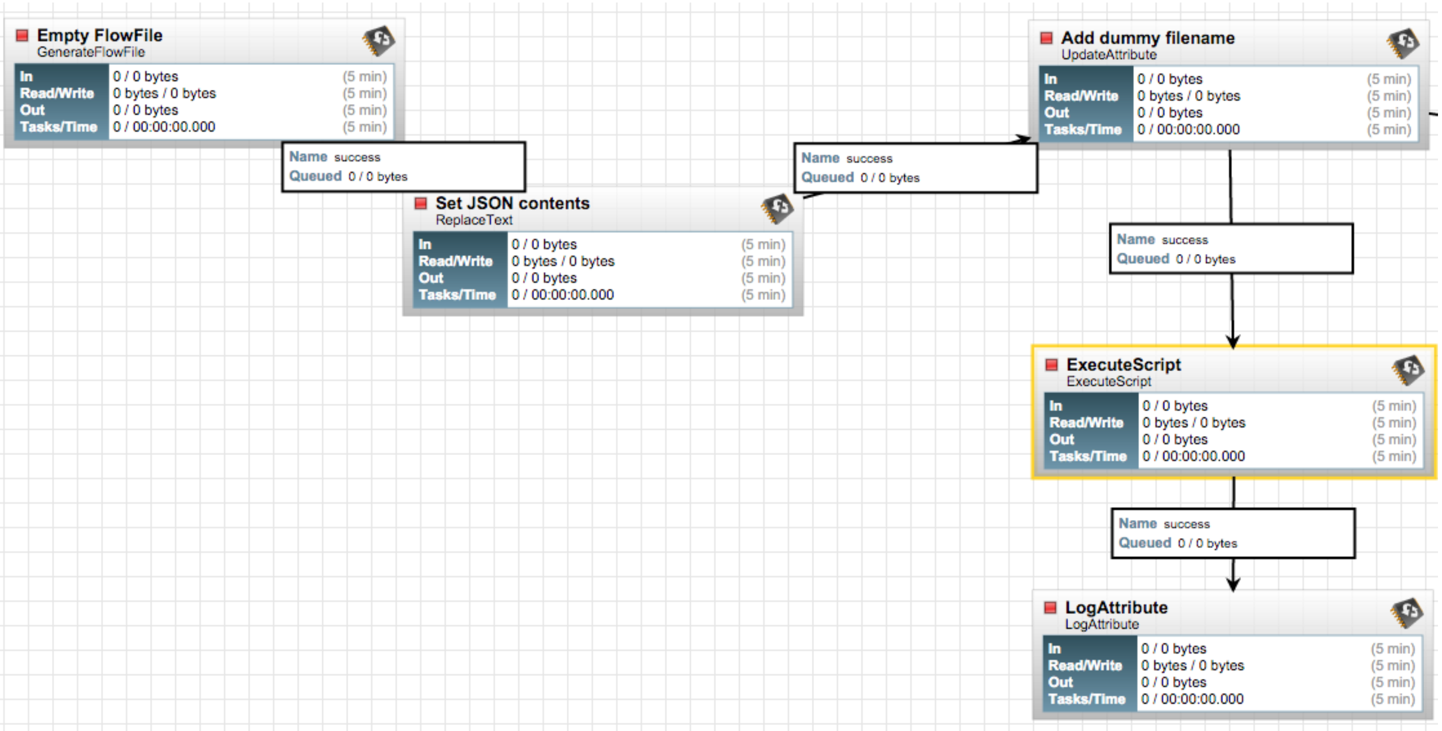
PROFINIT

- **Flowfiles** - Information Packet. Represents each object moving through the system and for each one. Map object (key, value pair)
- **Processors** - Perform the work - data routing, transformation, or mediation between systems.
- **Connections** - Connections provide the actual linkage between processors. These act as queues and allow various processes to interact at differing rates.
- **Flow Controller** – Scheduler. The Flow Controller maintains the knowledge of how processes connect and manages the threads and allocations thereof which all processes use

APACHE nifi

Hello World - Example

PROFINIT

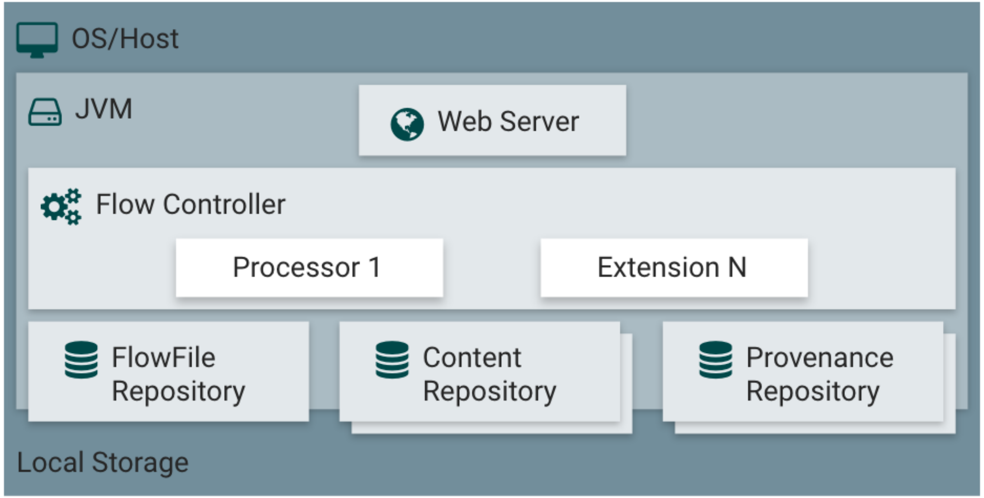


https://nifi.apache.org/docs.html

Architecture

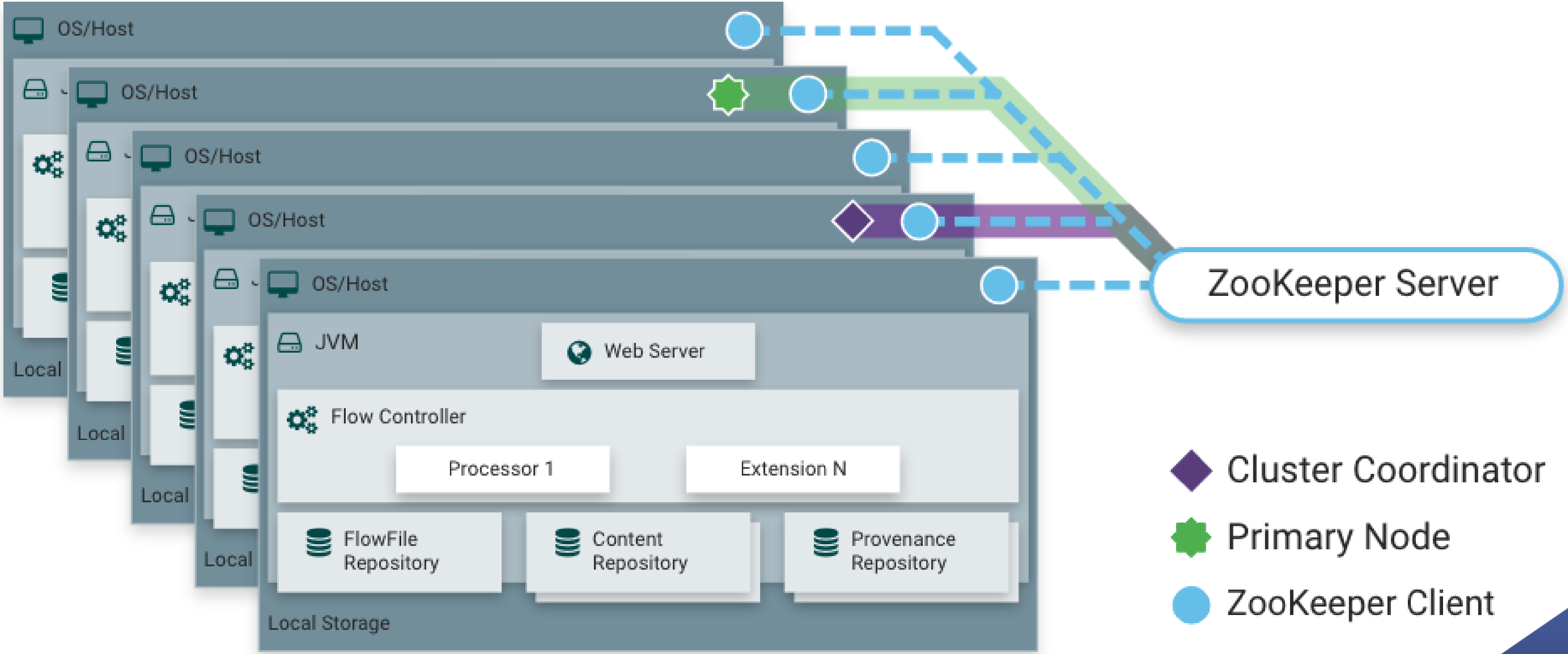
PROFINIT

- **FlowFile Repository** - where NiFi keeps track of the state of what it knows about a given FlowFile that is presently active in the flow.
- **Content Repository** - where the actual content bytes of a given FlowFile live
- **Provenance Repository** - where all provenance event data is stored.



|Cluster

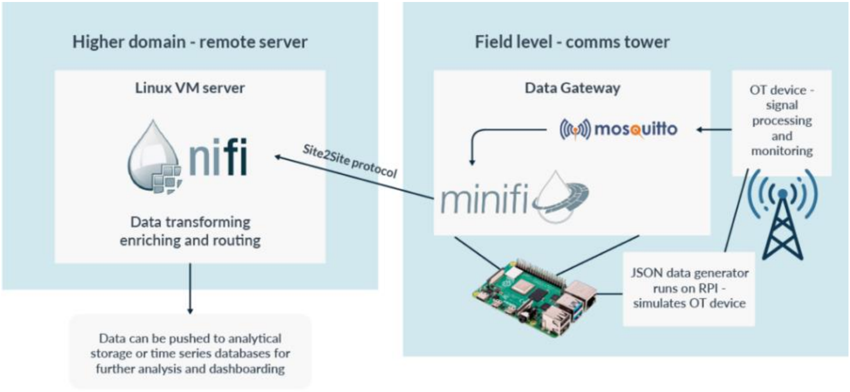
PROFINIT



Minifi

PROFINIT

- A complementary data collection approach that supplements the core tenets of NiFi
- Small size and low resource consumption
  - binary (3.2MB)
  - Original Java agent (50MB)
- Integration with NiFi for follow-on dataflow management

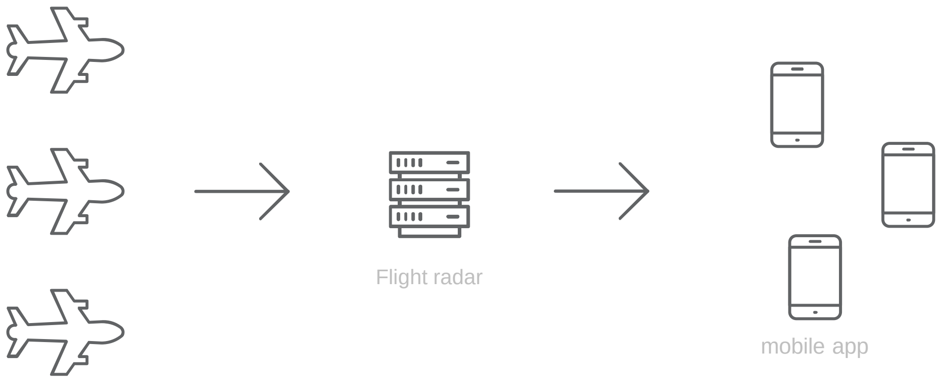


Apache Kafka

Motivation for message broker

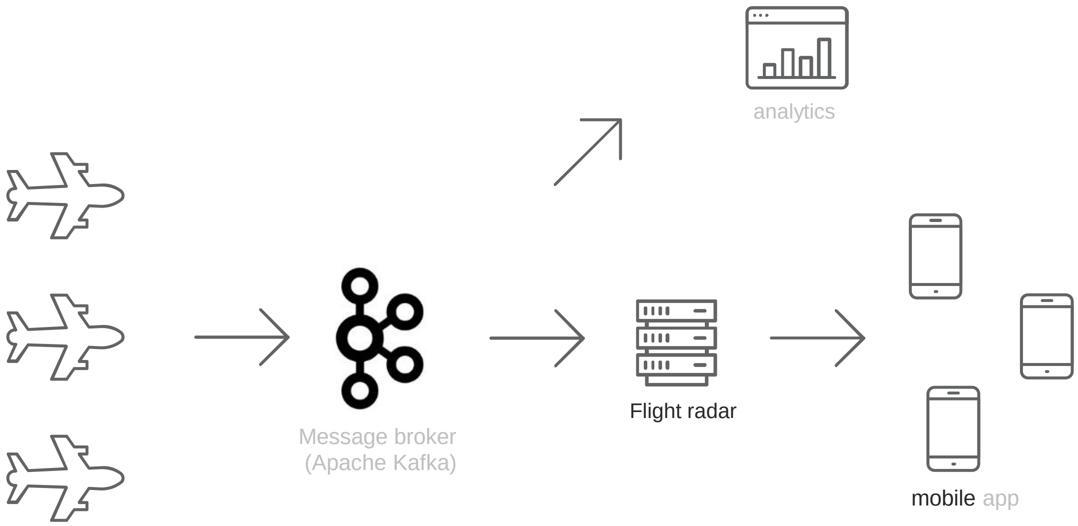
PROFINIT

- It is a vacation day, your flight is scheduled for 14:00
- Is my flight on time?



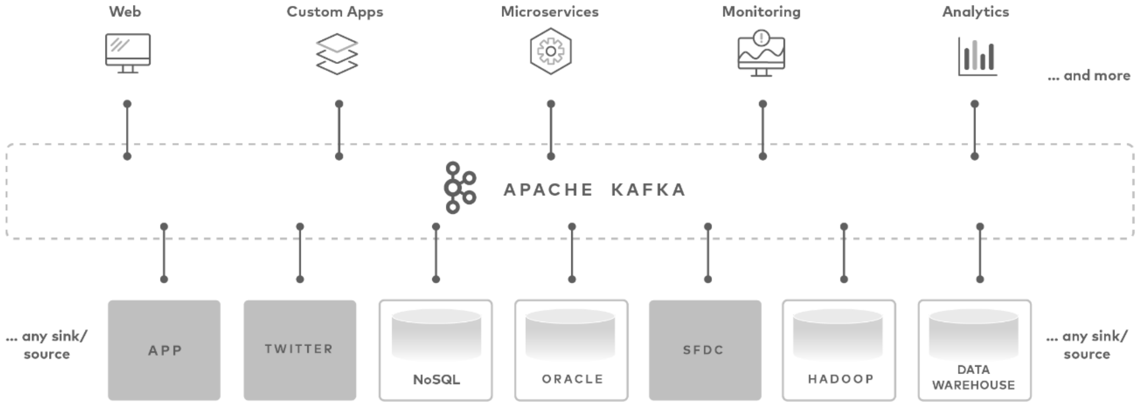
Getting rid of peer-to-peer data transfer

PROFINIT



Apache Kafka

- Started as message broker at Linkedin, now it’s a data processing ecosystem
- Key characteristics
  - High-throughput
  - Distributed
  - Scalable



https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines

Kafka 101 – naming convention

PROFINIT

- Consumer
  - An application that is reading data from Kafka
- Consumer Group
  - A group of an application processes that read data from Kafka
- Producer
  - An application that is writing data to Kafka
- Broker
  - Kafka process (single server) that is receiving data from producers, storing data on disk and provide them to consumers

Kafka 101 – naming convention

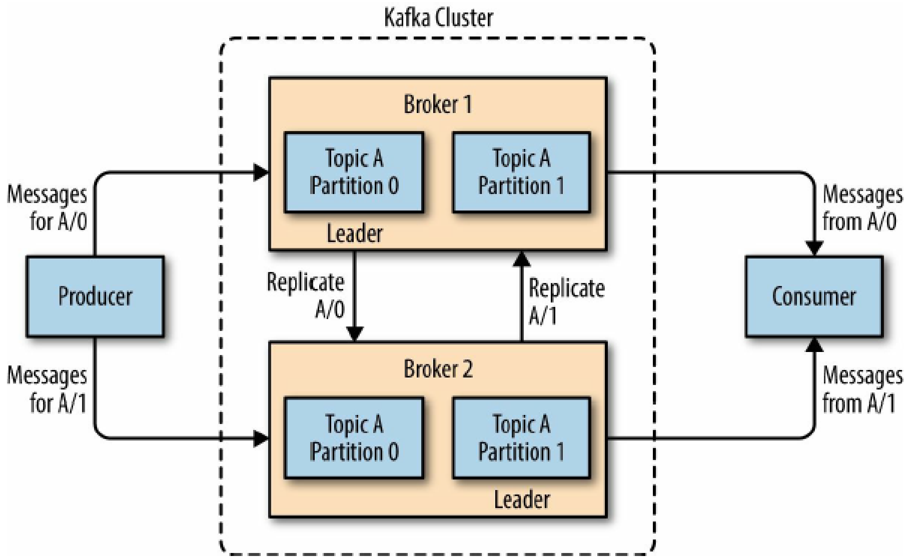
PROFINIT

- Topic
  - Named „message queue“
- Partition
  - Topics are broken down into partitions
- Offset
  - The position of a last committed message of a consumer in a topic/partition

Kafka cluster

PROFINIT

- Broker – a single Kafka server
- Kafka cluster – collection of brokers that work together



Kafka basics – Topic and partitioning

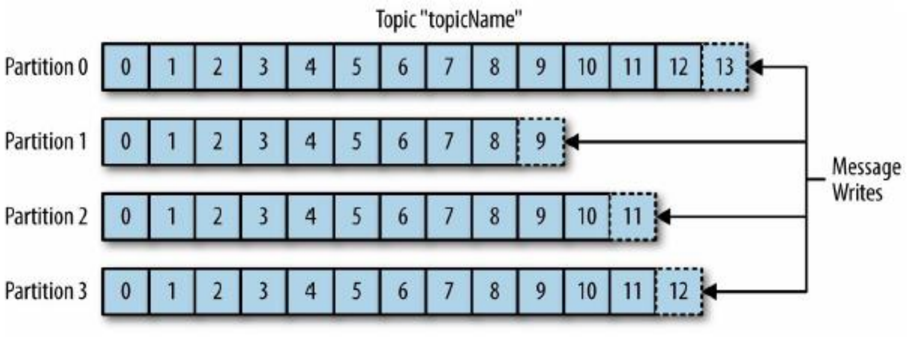
PROFINIT

- Topic
  - Named container for similar events
    - Usually there are many topics in a system
    - Data in one topic can be duplicated with data in another topic
  - Durable logs of events
    - New message always on the end
    - Can be read by seaking arbitrary offset
    - Are immutable - once something has happened, it is exceedingly difficult to make it un-happen
    - Are durable – stored on filesystem

Kafka basics – Topic and partitioning

PROFINIT

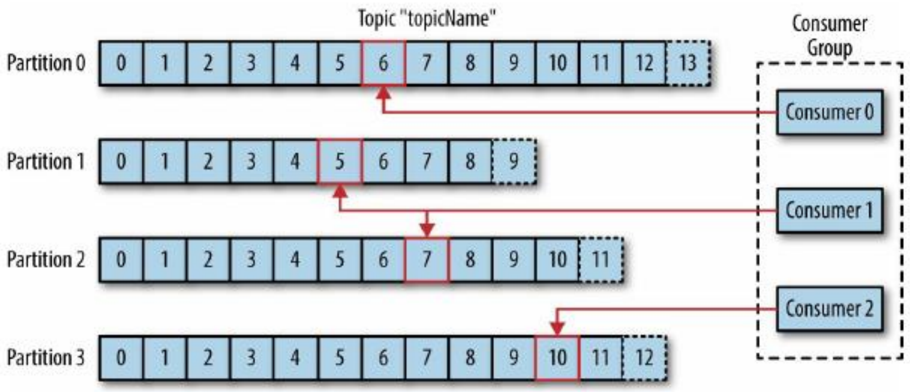
- Partitioning
  - Breaked single topic log to multiple logs that can live independently
  - Are spread out across a cluster



Kafka basics - Offset

PROFINIT

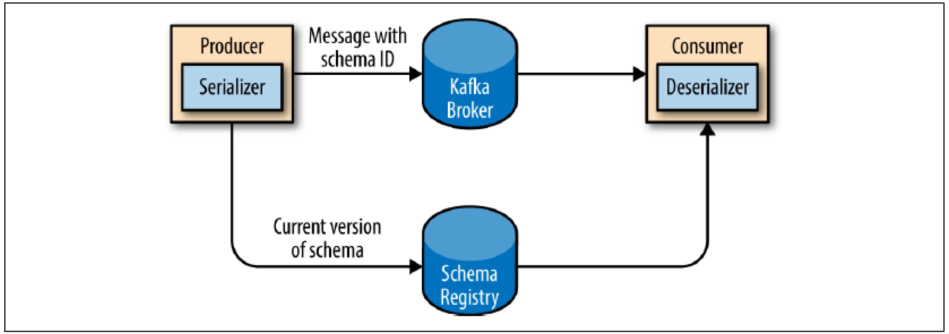
- The consumer offset is a way of tracking the sequential order in which messages are received by Kafka topics.
- Consumer offsets are persisted



Message structure

PROFINIT

- Key
- Value
- TopicName
- Key and Value can be any type (as long as we can serialize/deserialize them to bytes)
  - We can easily send json records as Values
  - Another very popular format to serialize data is Avro, but it requires Schema Registry



Message ordering in Kafka

PROFINIT

- Message order
  - Kafka guarantees to maintain a message order **per partition**
  - That doesn’t guarantee an order of messages **per topic**
- Messages with the same key are guaranteed to be send to the same partition
  - By default a hash partitioned is used
  - Messages without a key will be uniformly distributed between partitions

Kafka basic - Producer

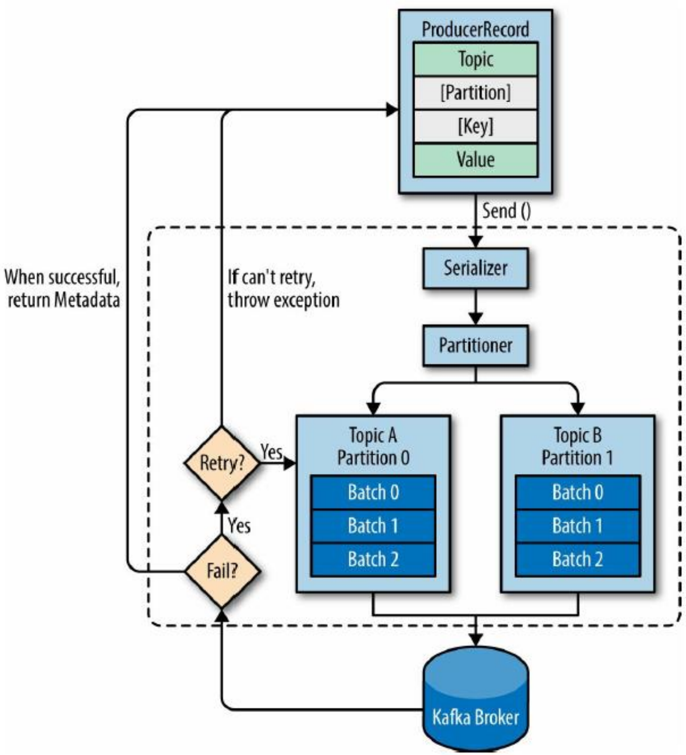
PROFINIT

- Client application
- Write data to appropriate kafka topic and broker
  - Serialize data
  - Define partition
  - Compress data
  - Handle errors

Acks	Throughput	Latency	Durability
0	High	Low	No Guarantee. The producer does not wait for acknowledgment from the server.
1	Medium	Medium	Leader writes the record to its local log, and responds without awaiting full acknowledgment from all followers.
-1	Low	High	Leader waits for the full set of in-sync replicas (ISRs) to acknowledge the record. This guarantees that the record is not lost as long as at least one IRS is active.

Kafka basic - Producing data

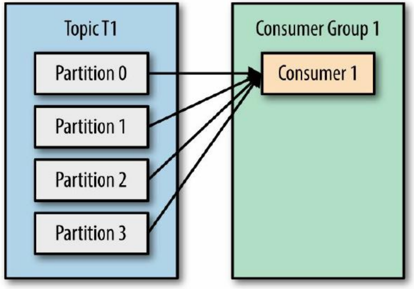
PROFINIT

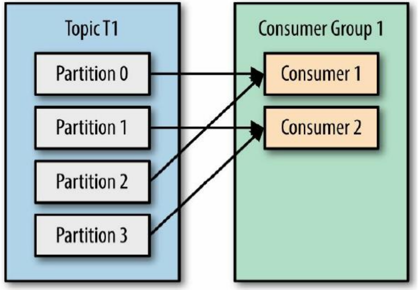


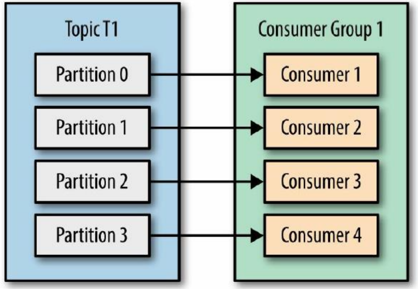
Kafka basic - Consumer

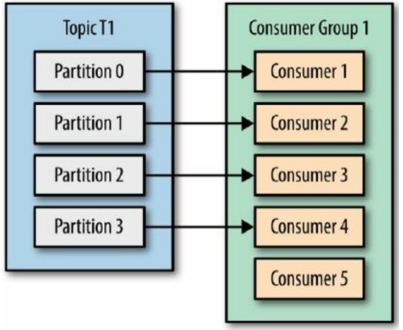
- Client application
- Read data from Kafka topic
- Scallable – organized to consumer groups
- Keep up to date metadata (offset)

Kafka basics – Consumer Group

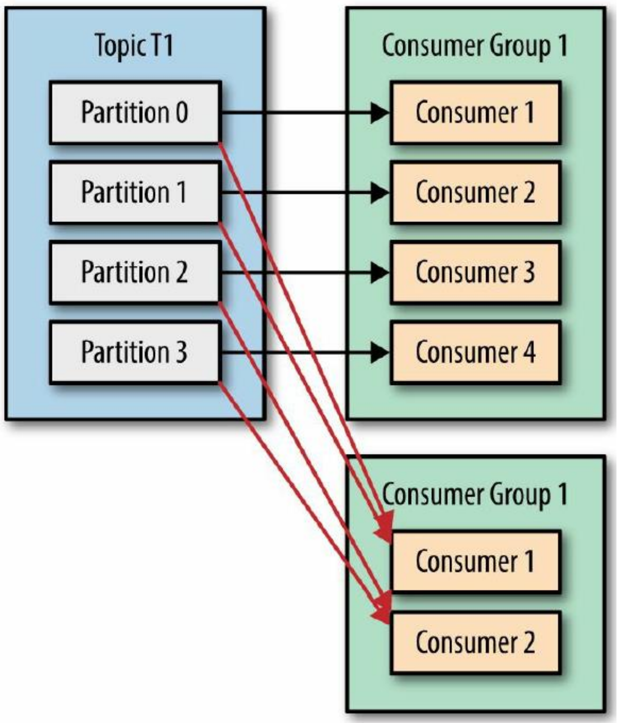








Kafka basics – Consumer Groups



Message delivery semantics

PROFINIT

- Exactly once
  - Every message is delivered only once
  - We need to keep track which messages were delivered and processed
- At least once
  - A message might be delivered more than once
  - Might be OK given pragmatic consideration
- At most once
  - The system will never try to deliver a message again once it was sent
  - Good option for non-critical data, that quickly become irrelevant

At least once example

```
def consume_loop(consumer, topics):
    try:
        consumer.subscribe(topics)
        msg_count = 0
        while running:
            msg = consumer.poll(timeout=1.0)
            msg_process(msg)
            msg_count += 1
            consumer.commit(async=False)
    finally:
        # Close down consumer to commit final offsets.
        consumer.close()
```

At most once example

PROFINIT

```
def consume_loop(consumer, topics):
    try:
        consumer.subscribe(topics)
        while running:
            msg = consumer.poll(timeout=1.0)
            consumer.commit(async=False)
            msg_process(msg)
    finally:
        # Close down consumer to commit final offsets.
        consumer.close()
```

Exactly once – Kafka support

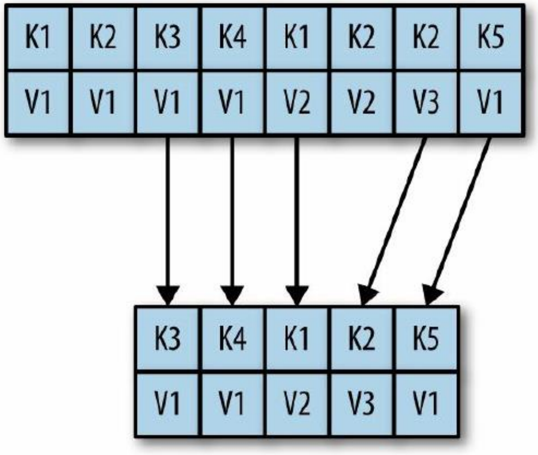
PROFINIT

- Idempotence: Exactly-once in order semantics per partition
  - Safeguard against duplicates in retry logic, that might be caused by broker or producer failure (hash
  - The message will be written to the Kafka topic once
  - Enable enable.idempotence=true in producer configuration
- Transactions: Atomic writes across multiple partitions
  - New transation API - atomic writes across multiple partitions
  - Consumer side - configuration
    - isolation.level (read_committed, read_uncommitted)
- Kafka streams
    - processing.guarantee=exactly_once

Log compaction

PROFINIT

- When the data is no longer needed (after retention period) the default action is to delete the message
- Kafka has special retention policy called “compaction” in case we want to store most recent message for each key



Kafka ecosystem

Kafka Connect

PROFINIT

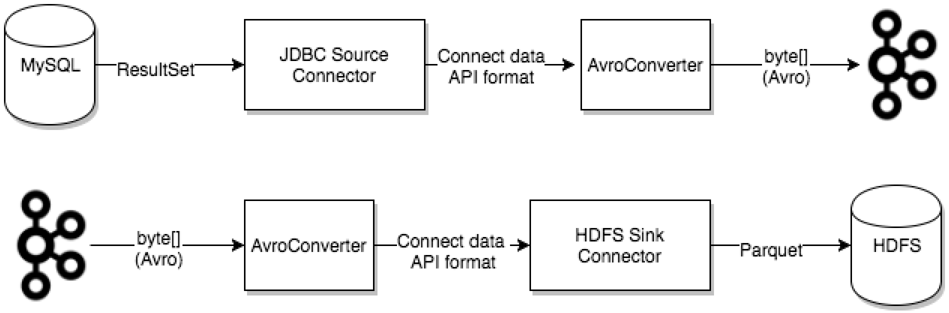
- Common framework for building connectors to integrate various data stores with Kafka
- Allows both getting data into and from Kafka
- If you find yourself trying to get data into or from Kafka, there is probably a connector for that
- The ingest process can be speeded up by running multiple connector instances in distributed fashion
- Aside from plain piping the data from system a to Kafka, also supports simple transformations of data records in transition

https://docs.confluent.io/cloud/current/connectors/index.html

https://docs.confluent.io/kafka-connectors/self-managed/supported.html#supported-self-managed-connectors

Kafka Connect- Example

PROFINIT

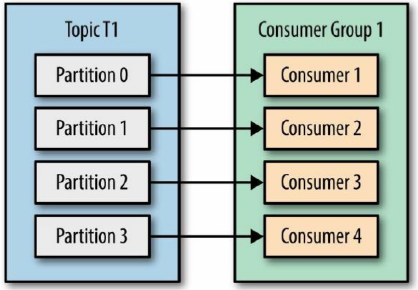



```
{
 "name": "gcs-sink-01",
 "config": {
   "connector.class": "io.confluent.connect.gcs.GcsSinkConnector",
   "tasks.max": "1",
   "topics": "gcs_topic",
   "gcs.bucket.name": "<my-gcs-bucket>",
   "storage.class": "io.confluent.connect.gcs.storage.GcsStorage",
   "format.class": "io.confluent.connect.gcs.format.avro.AvroFormat",
   "partitioner.class": "io.confluent.connect.storage.partitioner.DefaultPartitioner",
   "value.converter": "io.confluent.connect.avro.AvroConverter",
   "value.converter.schema.registry.url": "http://localhost:8081",
   "schema.compatibility": "NONE",
   "confluent.topic.bootstrap.servers": "localhost:9092",
    "errors.tolerance": "all",
   "errors.deadletterqueue.topic.name": "dlq-gcs-sink-01"
 }
}
```

Kafka Streams

PROFINIT

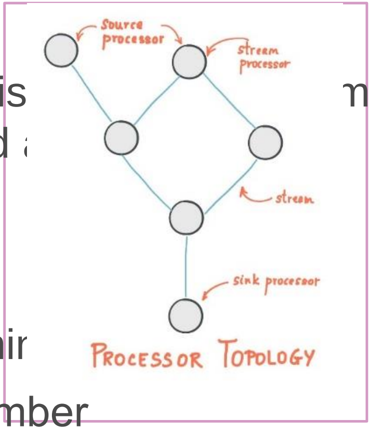
- Library build on-top of Kafka Producer/Consumer API for real-time stream processing
- It's best suited for reading data from Kafka topic, doing some work and then writing data to another Kafka topic
- Application instance is a JVM process
- The parallelism of a Kafka Streams application is primarily determined by how many partitions the input topics have



Kafka Streams naming convention

PROFINIT

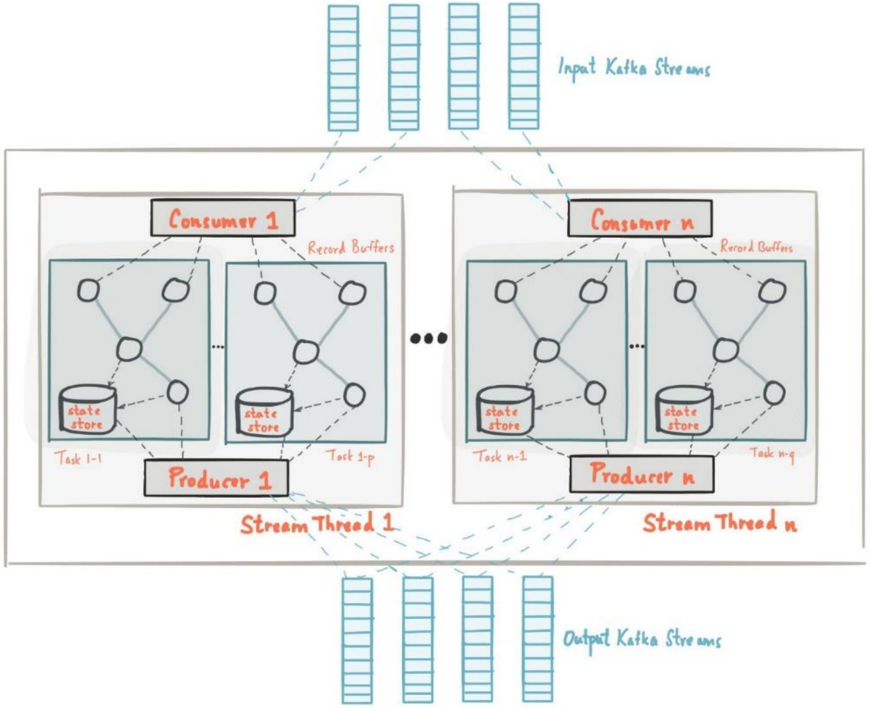
- Source processor
  - produces an input stream to its topology from one or multiple Kafka topics by consuming records from these topics.
- Sink processor
  - sends any received records from its up-stream processors to a specified Kafka topic.
- Stream processor
  - represents a processing step in a topology - it is … n data. Standard operations are map or filter, joins, and a…
- Stream task
  - smallest unit of work within a Kafka Streams application instance. The number of tasks is determir… by an application's source topic with the highest number of partitions



Stream task

Kafka Streams Architecture

PROFINIT

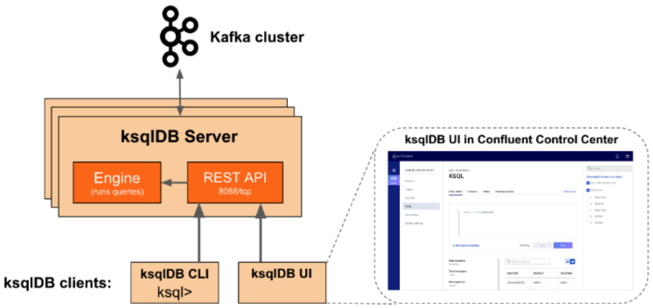


ksqlDB

PROFINIT

- DB-like abstraction on top of Kafka Streams
- Provides table-like interface over Kafka topic (using extended SQL syntax)
- Main components
  - KSQL Server – processes SQL statements and queries
  - KSQL CLI – CLI program to i

ksqlDB architecture and components



Kafka use-cases

- IoT and any message-oriented application
  - Sensor data
  - Financial transactions
  - Stock market
  - Logs

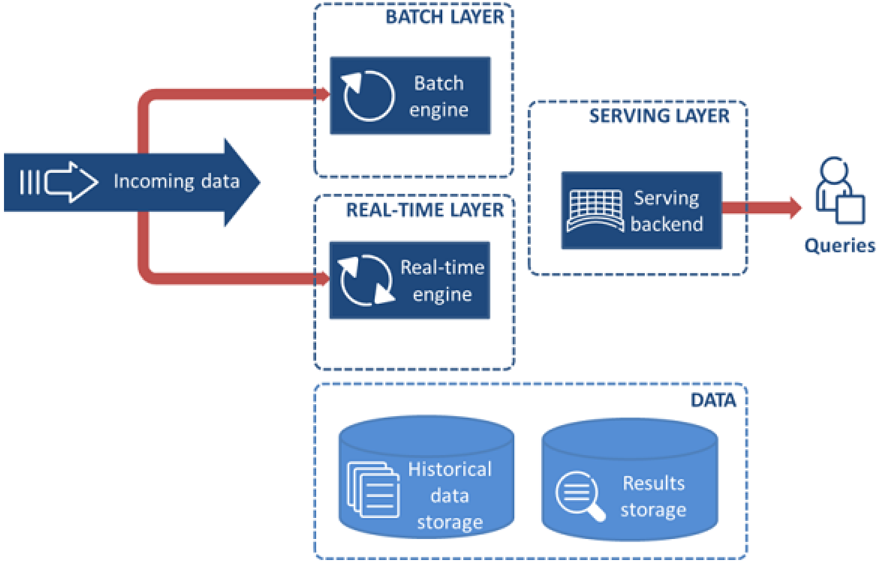
- Asynchronous application communication
- Publish subscribe messaging
- Data integration (Kappa & Lambda architecture)

Architectures

Lambda

PROFINIT

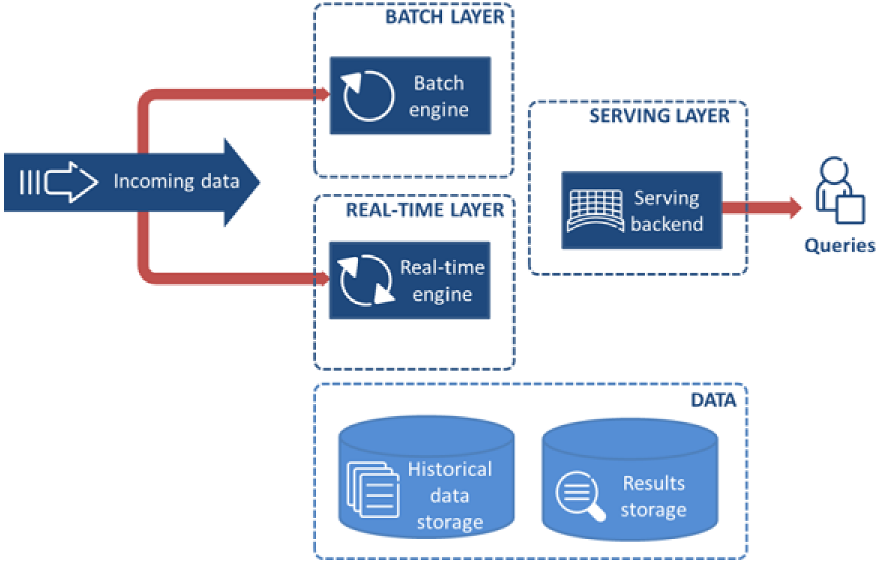
- From Apache Storm
- Nathan Marz, 2011
- http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html
- Yahoo, Netflix



Lambda

PROFINIT

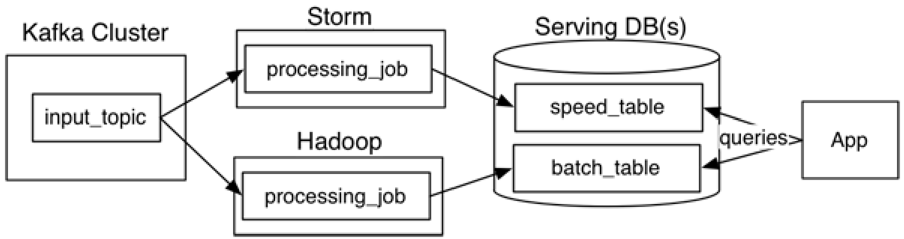
- 4 layers

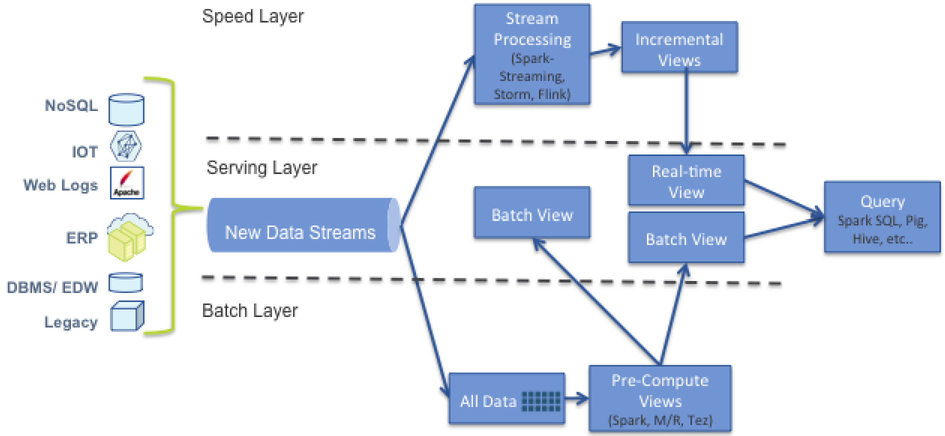


Lambda

PROFINIT

- Mapping to the technologies

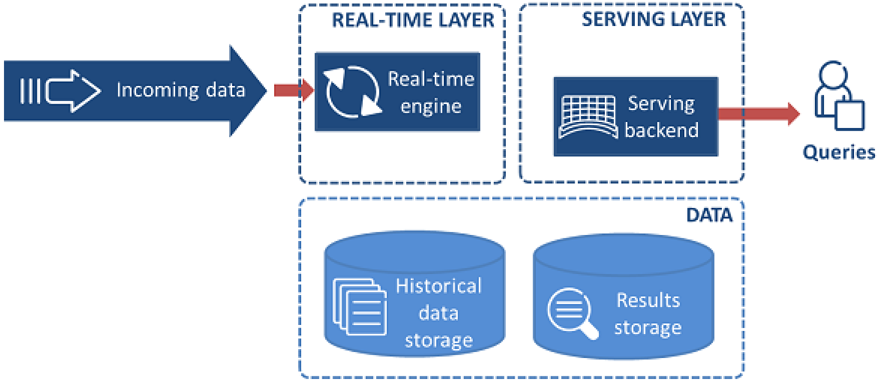




Kappa

PROFINIT

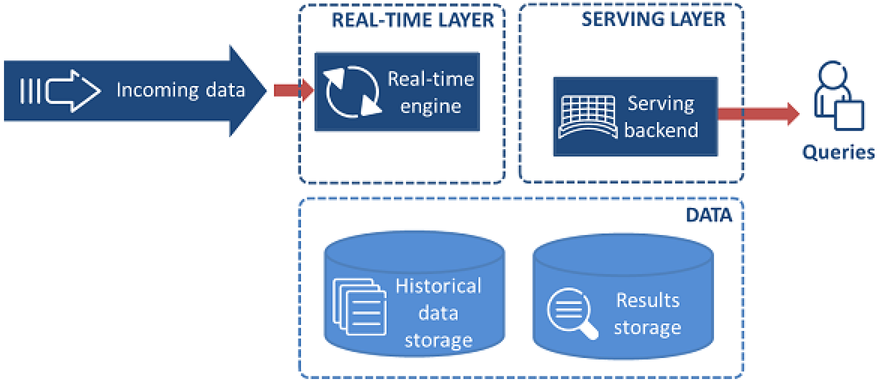
- 2014 Jay Kreps – Linkedin
- https://www.oreilly.com/ideas/questioning-the-lambda-architecture



Kappa

PROFINIT

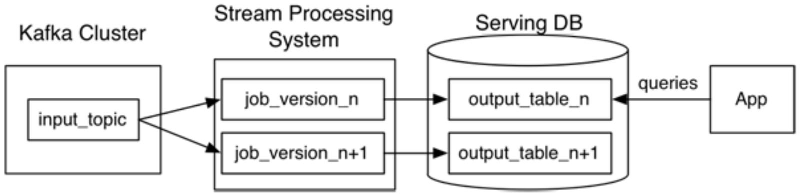
- 3 layers – batch layer removed
- Long retention can be used
- How to work with state – microbatches?

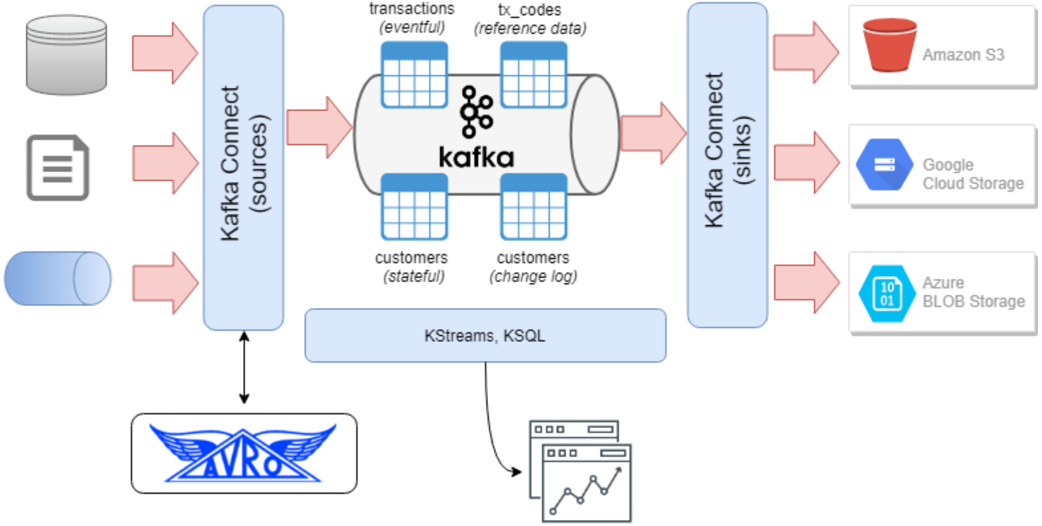


Kappa

PROFINIT

- Mapping to technologies





Díky za pozornost

Profinit EU, s.r.o.
Tychonova 2, 160 00 Praha 6

Tel.: + 420 224 316 016, web: www.profinit.eu

LINKEDIN
linkedin.com/company/profinit

TWITTER
@profinit_EU

FACEBOOK
facebook.com/Profinit.EU

YOUTUBE
Profinit EU, s.r.o.