



B0M33BDT – Apache Kafka, Architecture patterns

Josef Vonášek

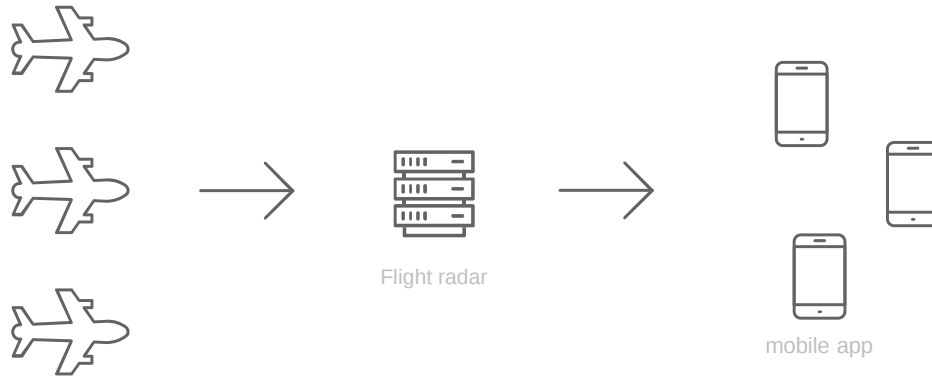
5. Listopadu 2025



Apache Kafka

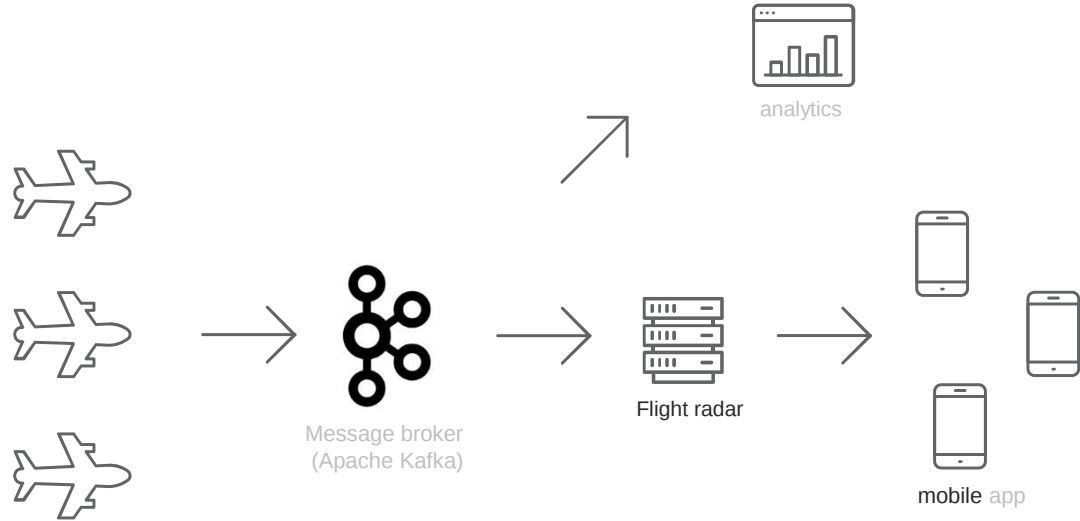
Motivation for message broker

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



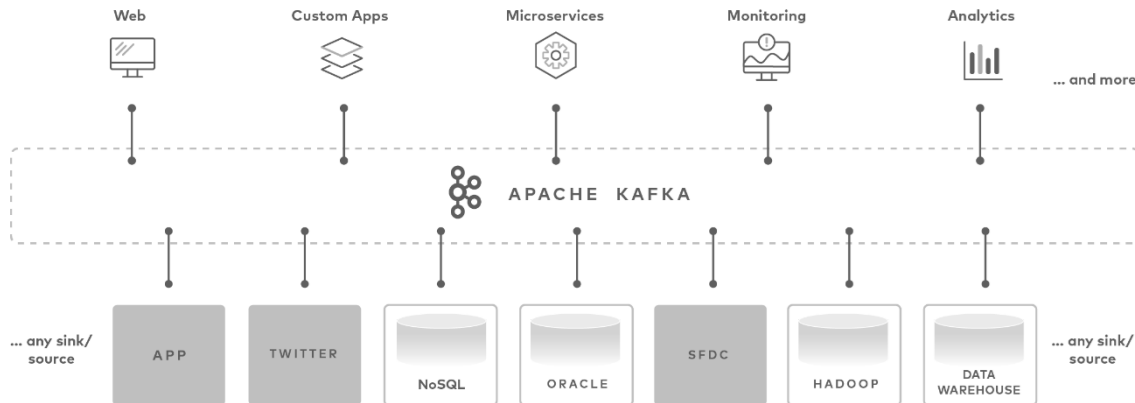
Getting rid of peer-to-peer data transfer

< PROFINIT >



Apache Kafka

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



Kafka 101 – naming convention

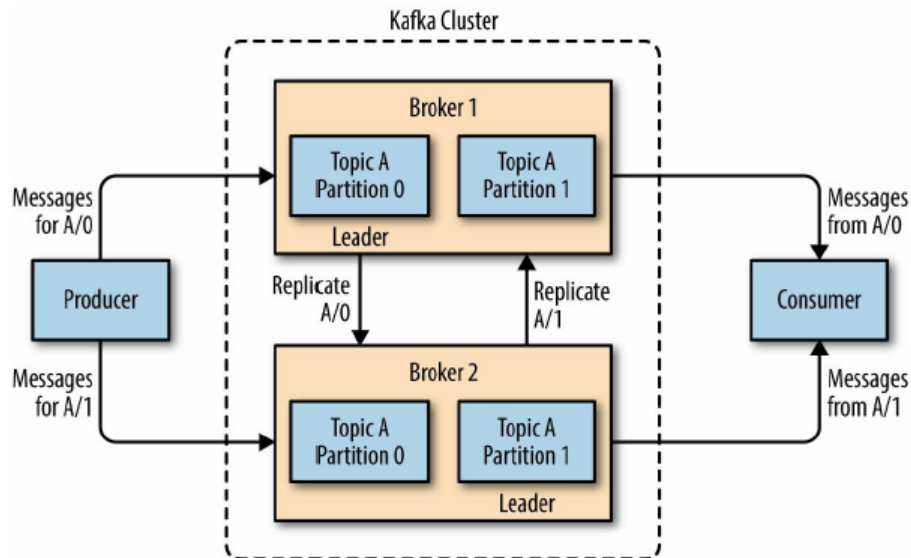
- Consumer
 - An application that is reading data from Kafka
- Consumer Group
 - A group of an application processes that read data from Kafka
- Producer
 - An application that is writing data to Kafka
- Broker
 - Kafka process (single server) that is receiving data from producers, storing data on disk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

Kafka 101 – naming convention

- 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

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



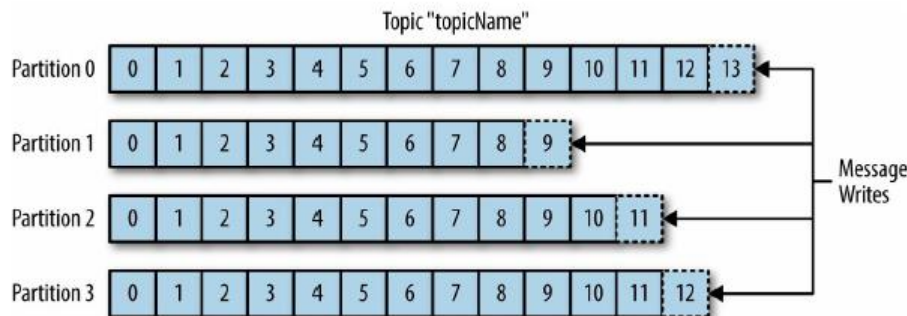
> Topic

- 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 seeking 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

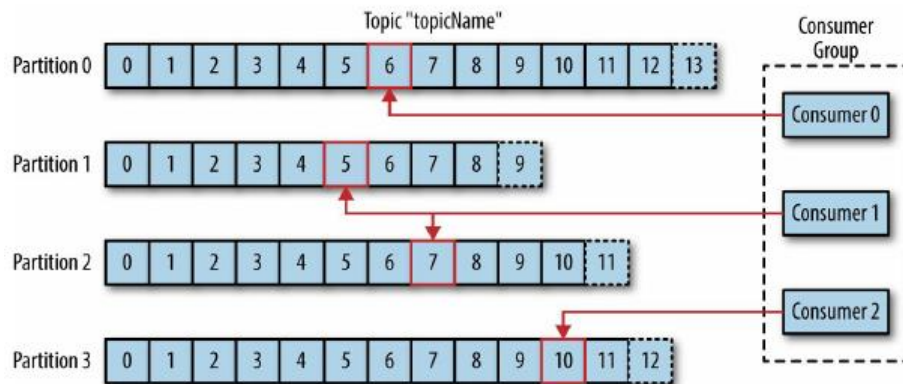
> Partitioning

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



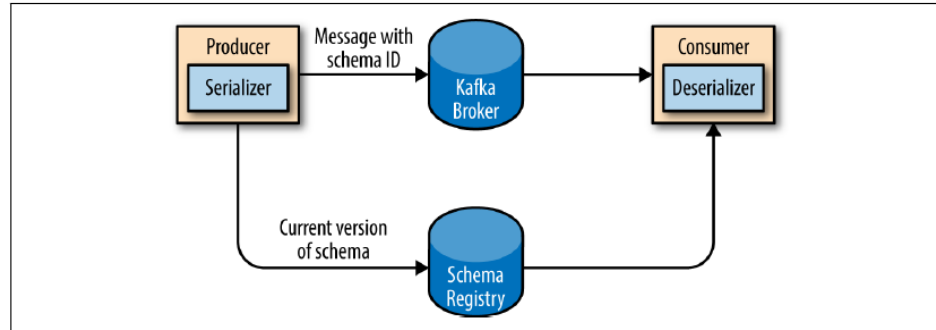
Kafka basics - Offset

- > 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

- Key
- Value
- Header
- 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



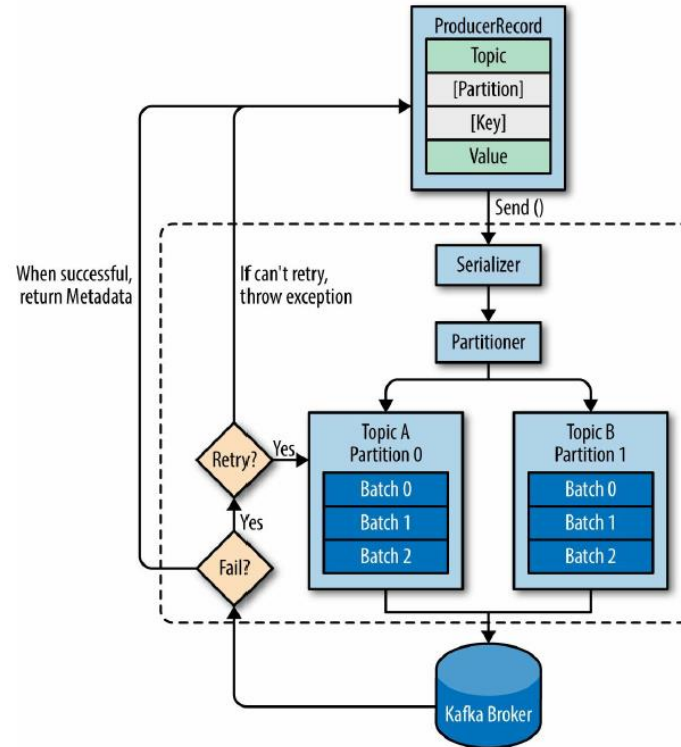
- Message order
 - Kafka guarantees to maintain a message order **per partition**
 - That doesn't guarantee an order of messages **per topic**
- Messages with the same key are guaranteed to be send to the same partition
 - By default a hash partitioned is used
 - Messages without a key will be uniformly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

Kafka basic - Producer

- 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 ISR is active.

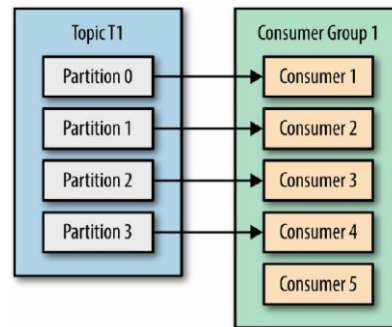
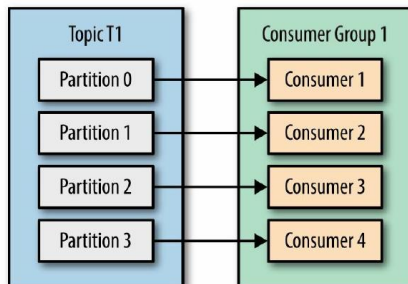
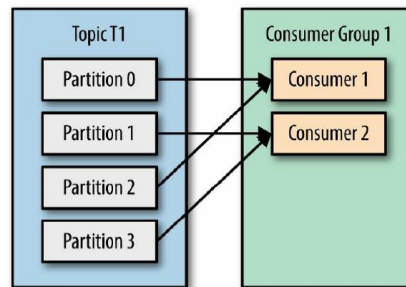
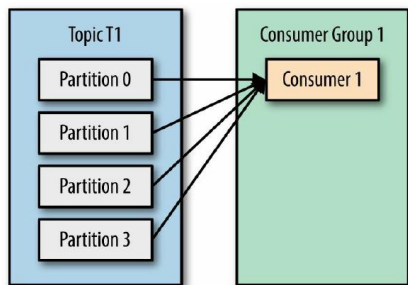
Kafka basic - Producing data



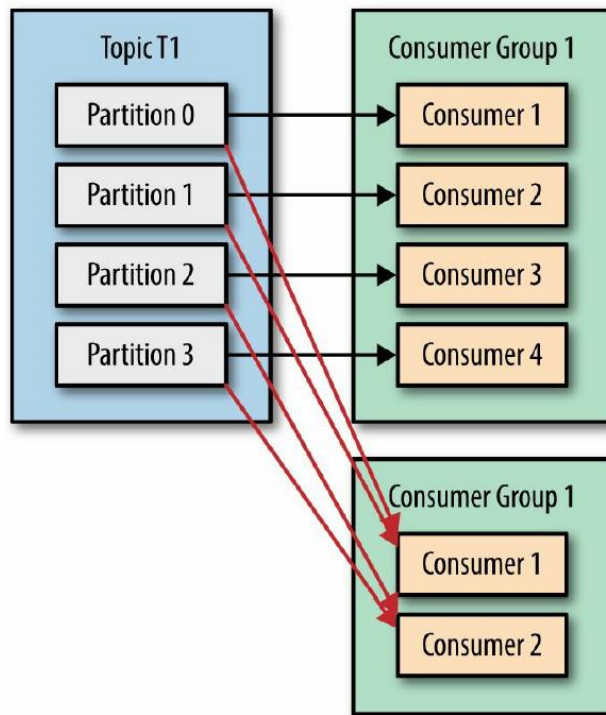
Kafka basic - Consumer

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

Kafka basics – Consumer Group

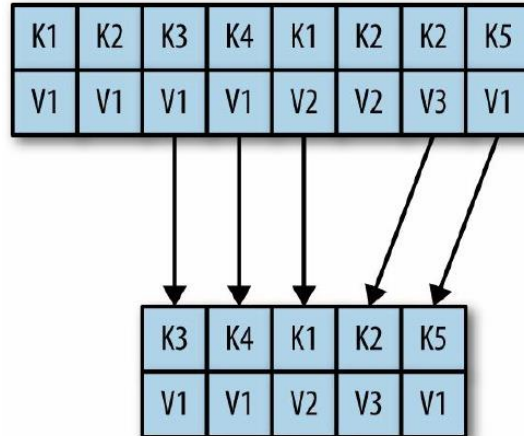


Kafka basics – Consumer Groups



Log compaction

- 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



- Centralized repository for managing schemas of kafka messages
- Key component for data governance
 - Schema management including validation
 - Compatibility check
 - Versioning
- Supported schema types
 - AVRO
 - JSON
 - Protobuff



Message delivery

- 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

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



Recap

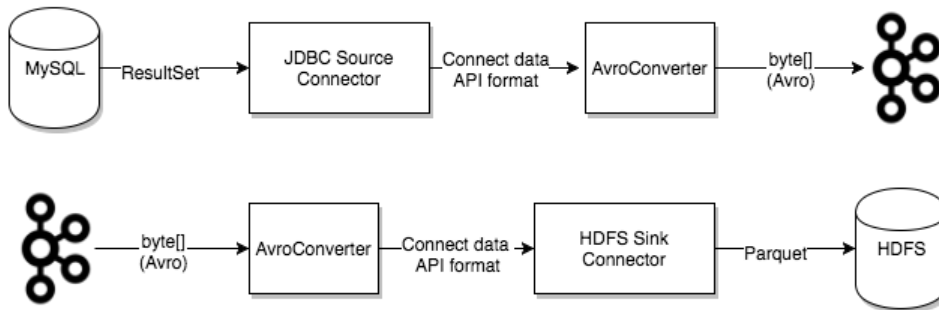
Latest releases

- **KRaft** : Zookeeper
- **Tiered Storage**: „hot“ a „cold“ storage
- **Cloud-native Kafka**: Improved elasticity - autoscaling
- **Queues support**: More consumers for one partition
- **New Rebalancing**: Reatime rebalancing

Kafka ecosystem

- 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

Kafka Connect- Example



```
{
  "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"
  }
}
```

- 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
- Stateful support (RockDB)

Kafka Streams naming convention

> 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.

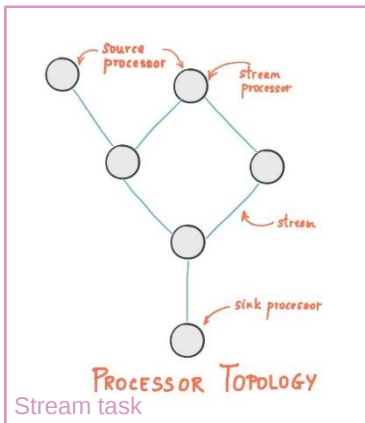
> Stream processor

- represents a processing step in a topology - it is used to transform data. Standard operations are map or filter, joins, and aggregations.

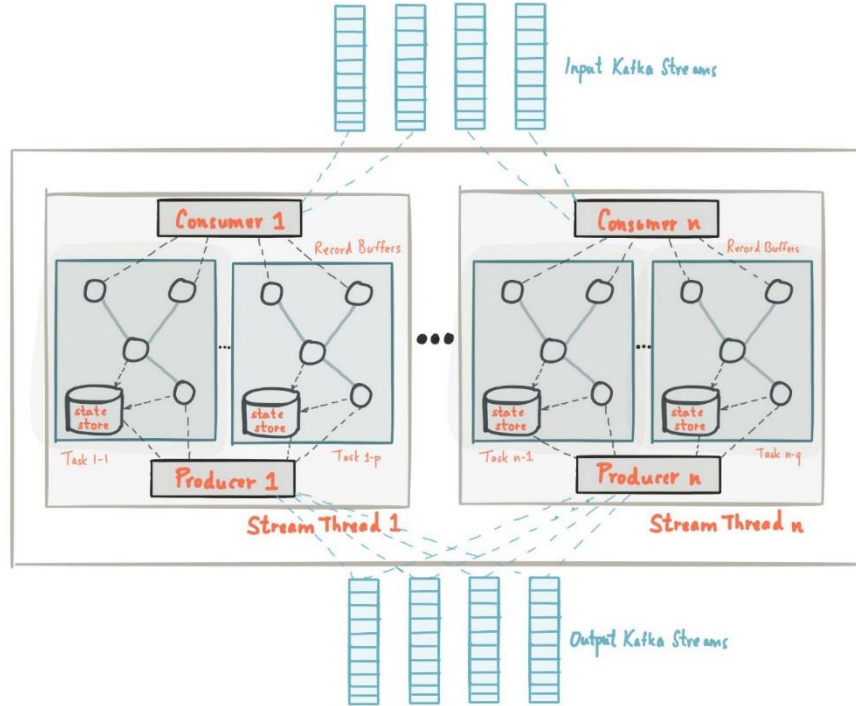
> Stream task

- smallest unit of work within a Kafka Streams

application instance. The number of tasks is determined by an application's source topic with the highest number of partitions.



Kafka Streams Architecture

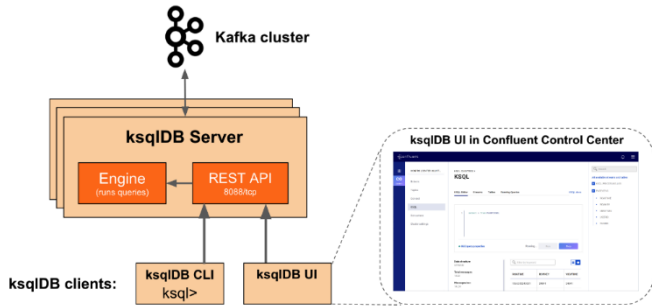


ksqlDB



- 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 interact with Server

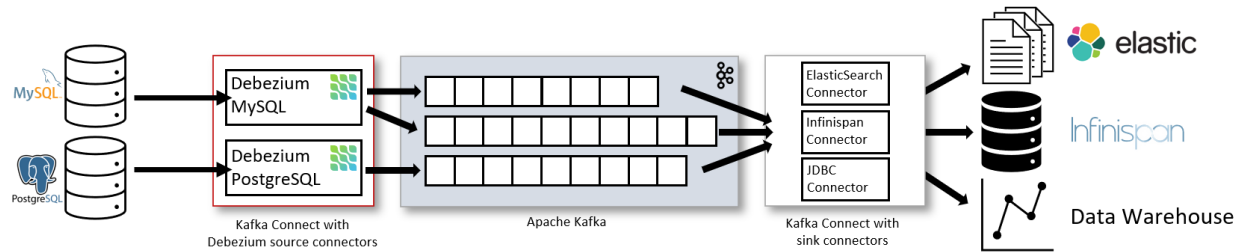
ksqlDB architecture and components



Kafka Connect - Debezium

Debezium

- Debezium is a set of distributed services that capture row-level changes in your databases (CDC – change data capture)



Debezium - sources

- > MySQL
- > MariaDB
- > MongoDB
- > Oracle
- > MSSQL server
- > Cassandra
- > ...

Debezium – connector operations

- CDC – debezium send changes
- Full snapshot – debezium send full mirror of table
- Incremental snapshot - debezium send records based on provided filter (request)

Use Cases

- Messaging - replacement of messages brokers (RabbitMQ, Active MQ)
- Log Aggregation – stream of log events with applied business logic
- Web Tracking - user activity tracking pipeline with publish subscribe model
- Stream Processing – RSS Feed
- Event sourcing – Application style (current (at every a point of time) state can be reconstruct based on past events)
- CommitLog – support for distributed system
- Reporting – operational monitoring

Cloudification

Confluent cloud

- Cloud streaming platform build on Apache Kafka
 - Available on AWS, Azure, Google Cloud
- Streaming data accross different cloud providers
- Integration with Apache Flink (streaming platform)
- New Kora Engine (better performance – latency)
- Pricing
 - Basic – Limited storage
 - Standard – HA 99,99
 - Enterprise - Private networking

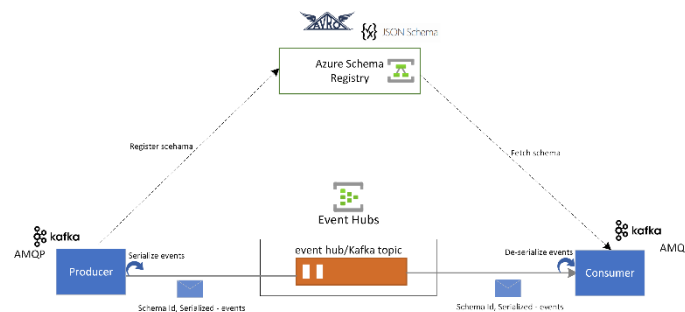
Azure Event Hubs

> Streaming platform with native Apache Kafka support

- Advanced Message Queuing Protocol (AMQP)
- Apache Kafka
- HTTPS protocols

> Pricing

- Basic – without Kafka
- Standard – Kafka support, pay per event
- Premium – 90 days retention
- Dedicated – 10 TB retention



Amazon Managed Streaming for Apache Kafka (MSK)

◀ PROFINIT ▶

> Fully managed Kafka service

- MSK Cluster
- MSK Connect
- MSK Replicator

> MSK serverless

- Serverless cluster

> Pricing

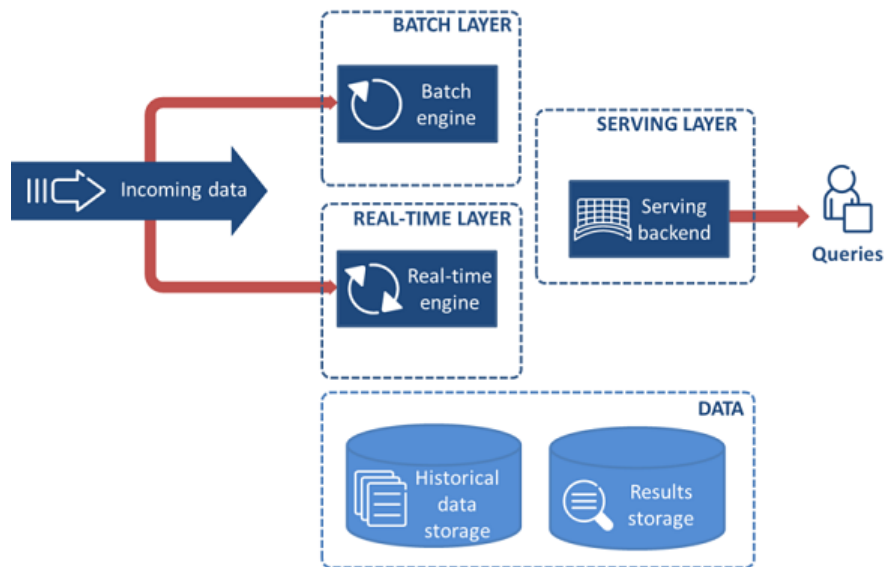
- Cluster instances (cluster size per hour)
- Serverless (price per hour - cluster, partition, storage, data in, data out)
- Connect (MCU per hour)



Architectures

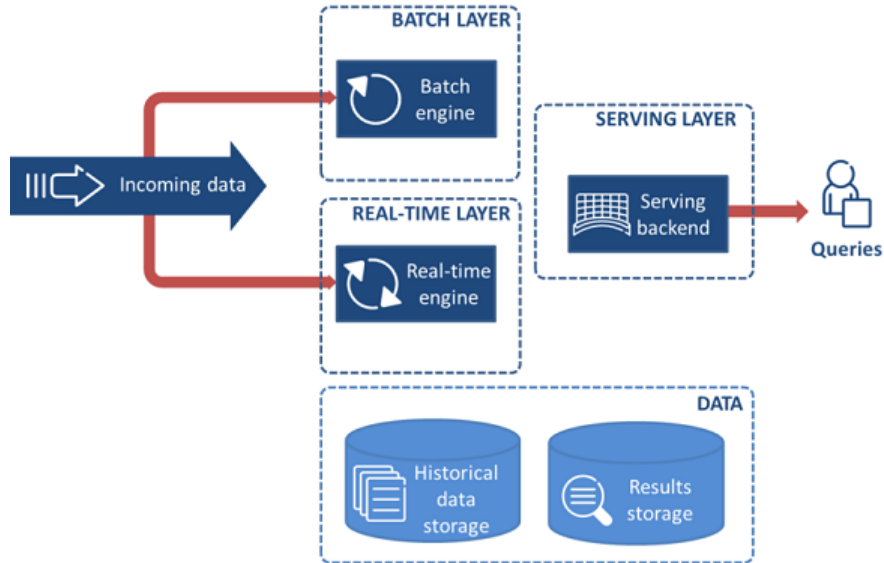
Lambda

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

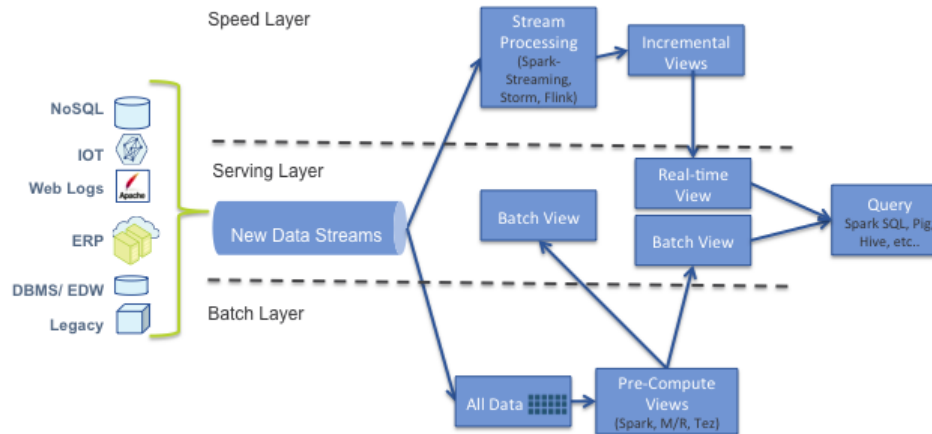
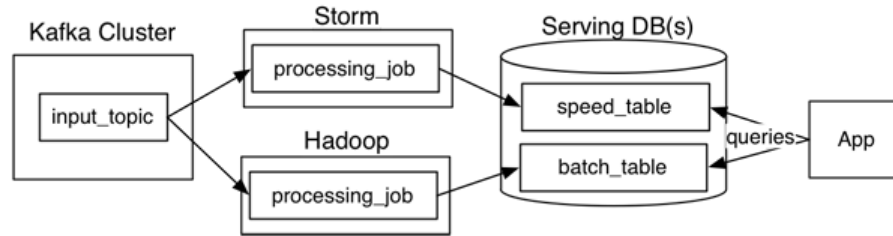


Lambda

> 4 layers

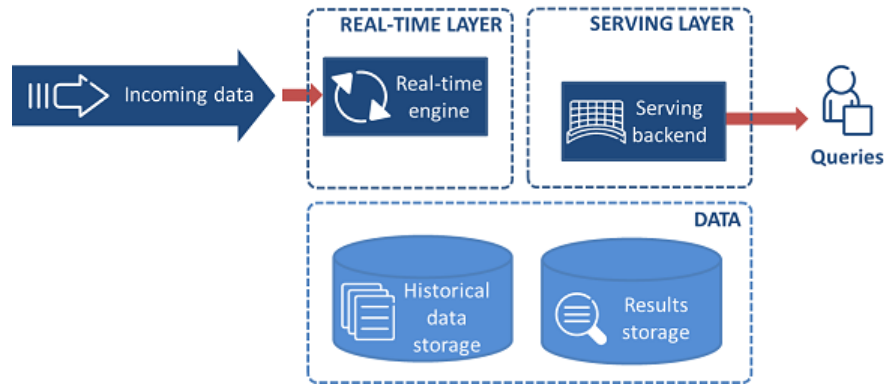


> Mapping to the technologies



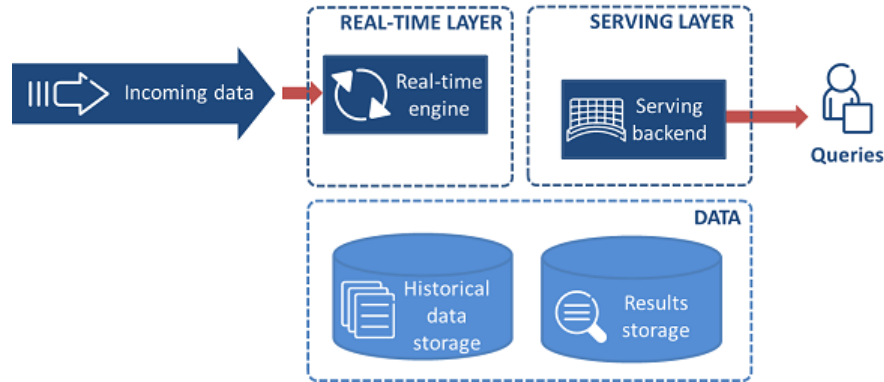
Kappa

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

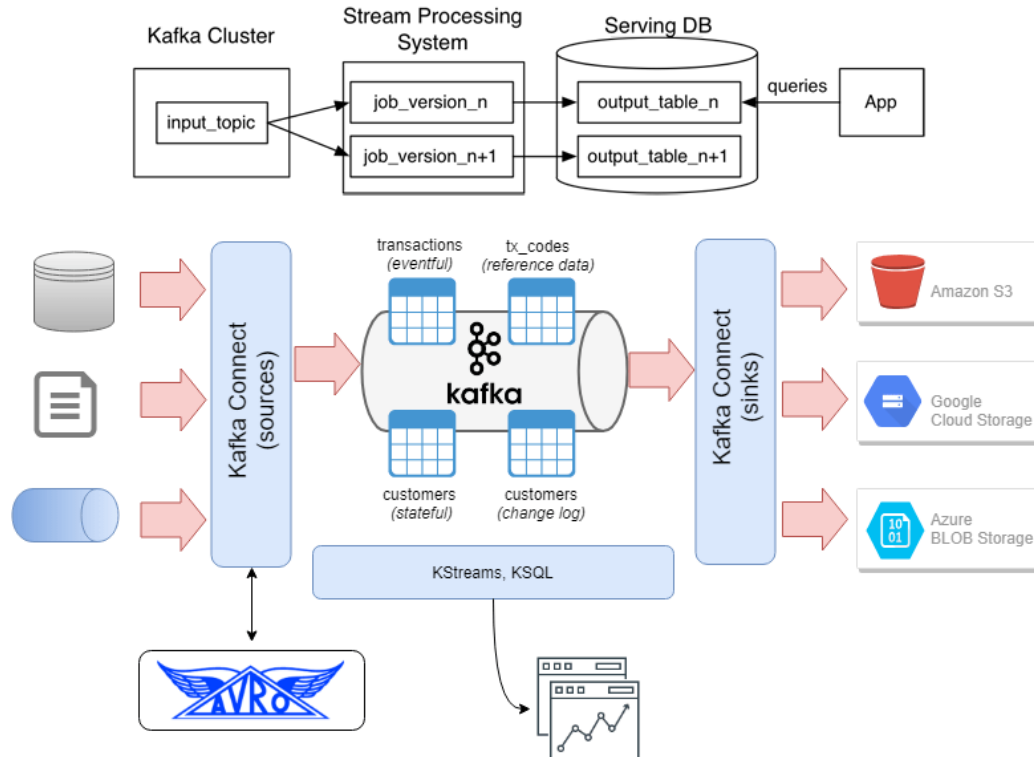


Kappa

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



> Mapping to technologies





Apache Nifi

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



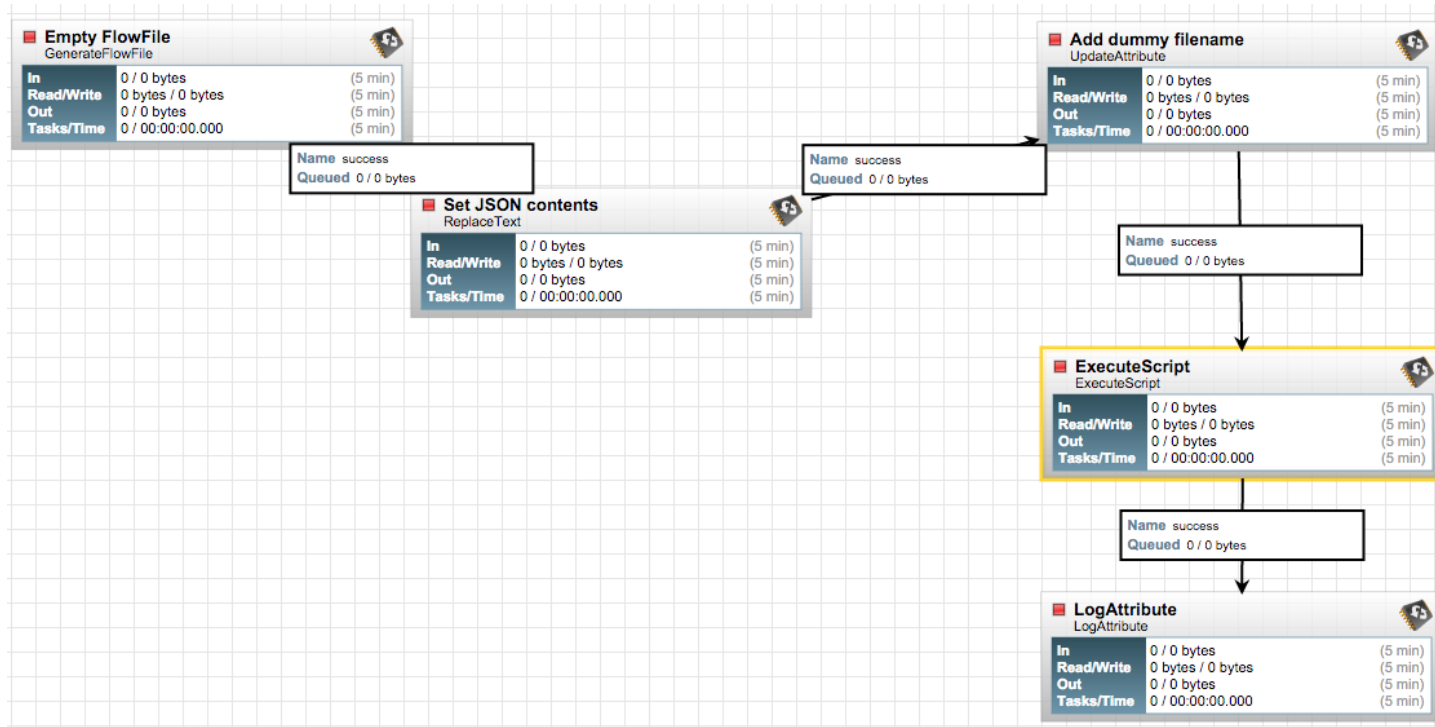
- NiFi is a processing engine designed to manage a continuous flow of information as a series of events in an ecosystem
- Visual creation and management of directed graphs of processors
- Highly concurrent model without a developer having to worry about the typical complexities of concurrency
- Natural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



Naming Convention

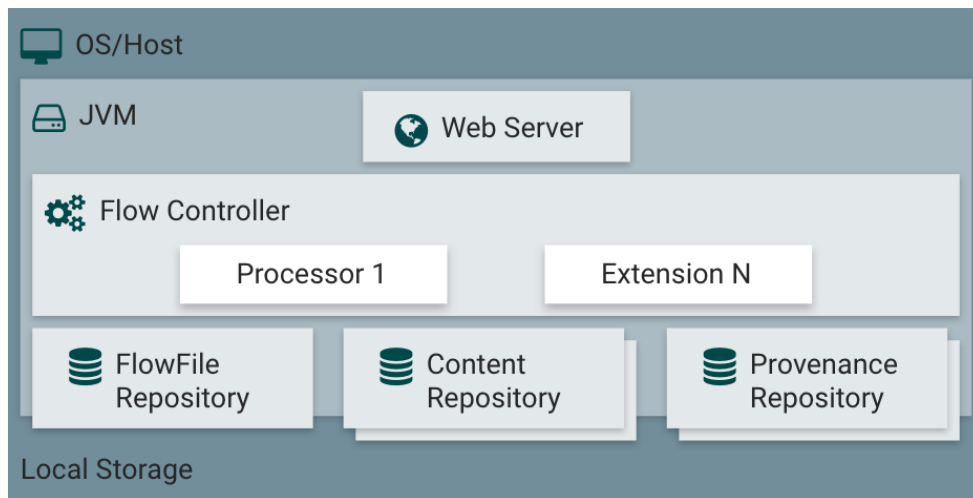
- **Flowfiles** - Information Packet. Represents each object moving through the system and for each one. Map object (key, value pair)
- **Processors** - Perform the work - data routing, transformation, or mediation between systems.
- **Connections** - Connections provide the actual linkage between processors. 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

Hello World - Example



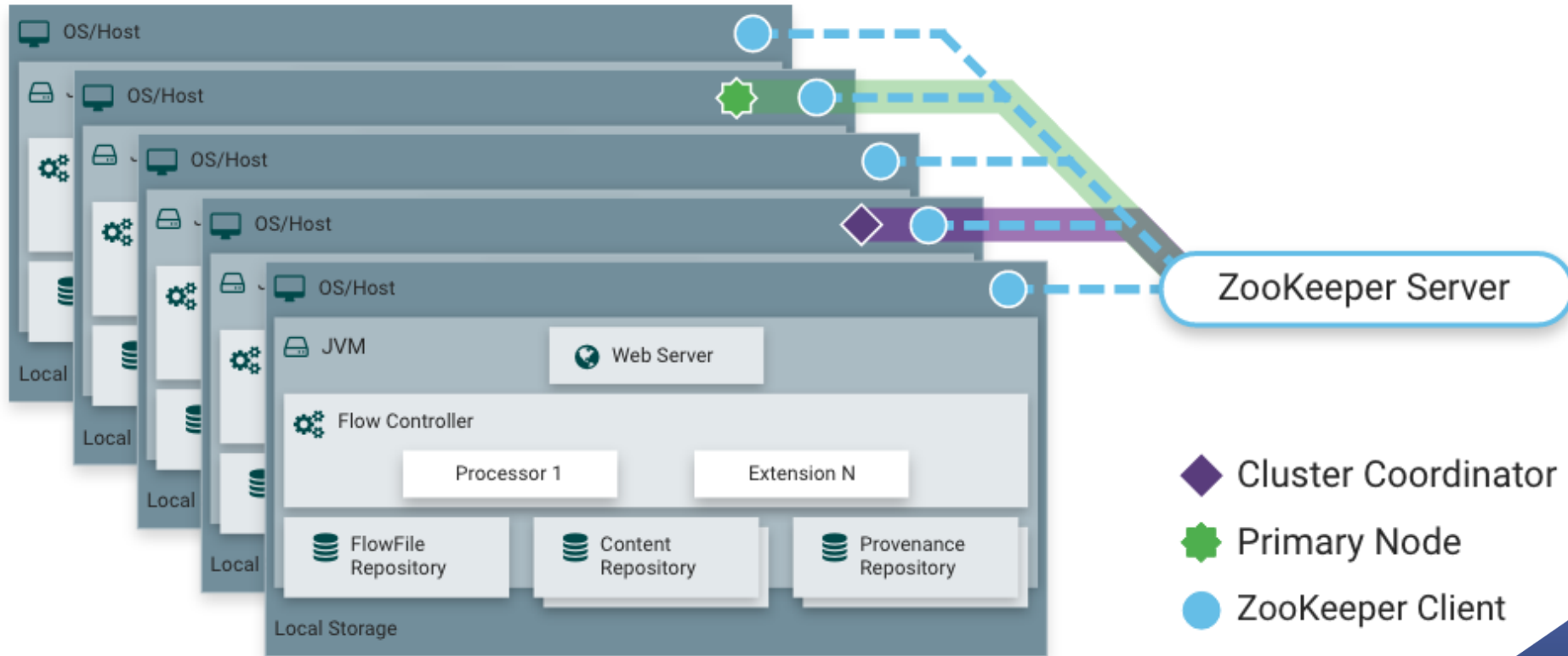
Architecture

- **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.

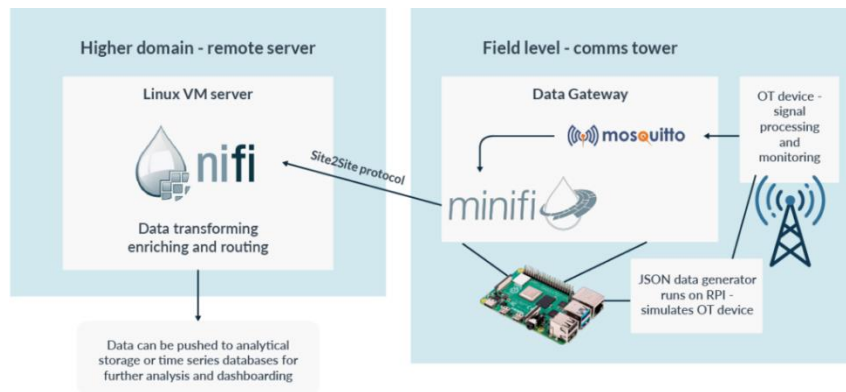


|Cluster

< PROFINIT >



- 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



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](https://twitter.com/profinit_EU)

 FACEBOOK
facebook.com/Profinit.EU

 YOUTUBE
Profinit EU, s.r.o.