
ONEWAY
SOLUTION

One Way Solution

Best Practices & Guidelines

Data Engineering – [Day 5]

LUAN MORENO

CEO & Data Architect

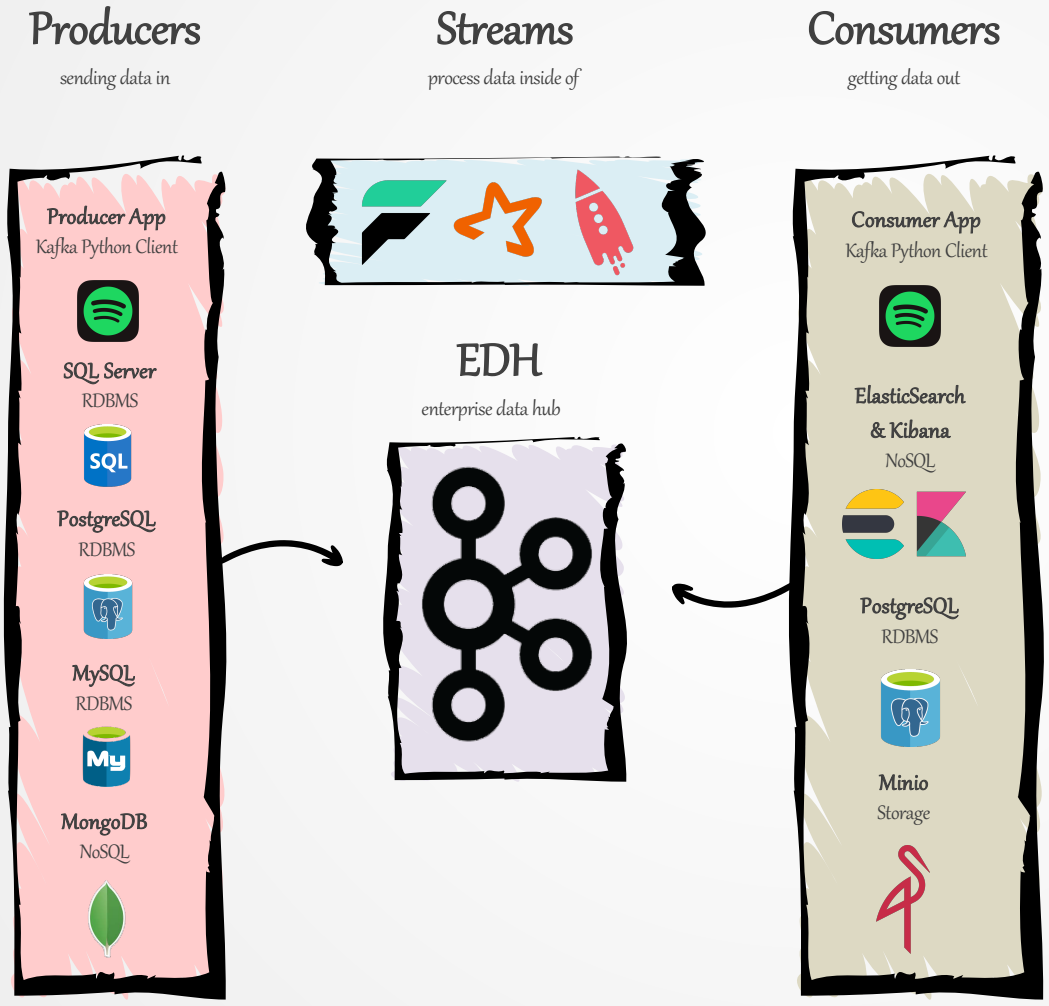
Data Engineer & MVP

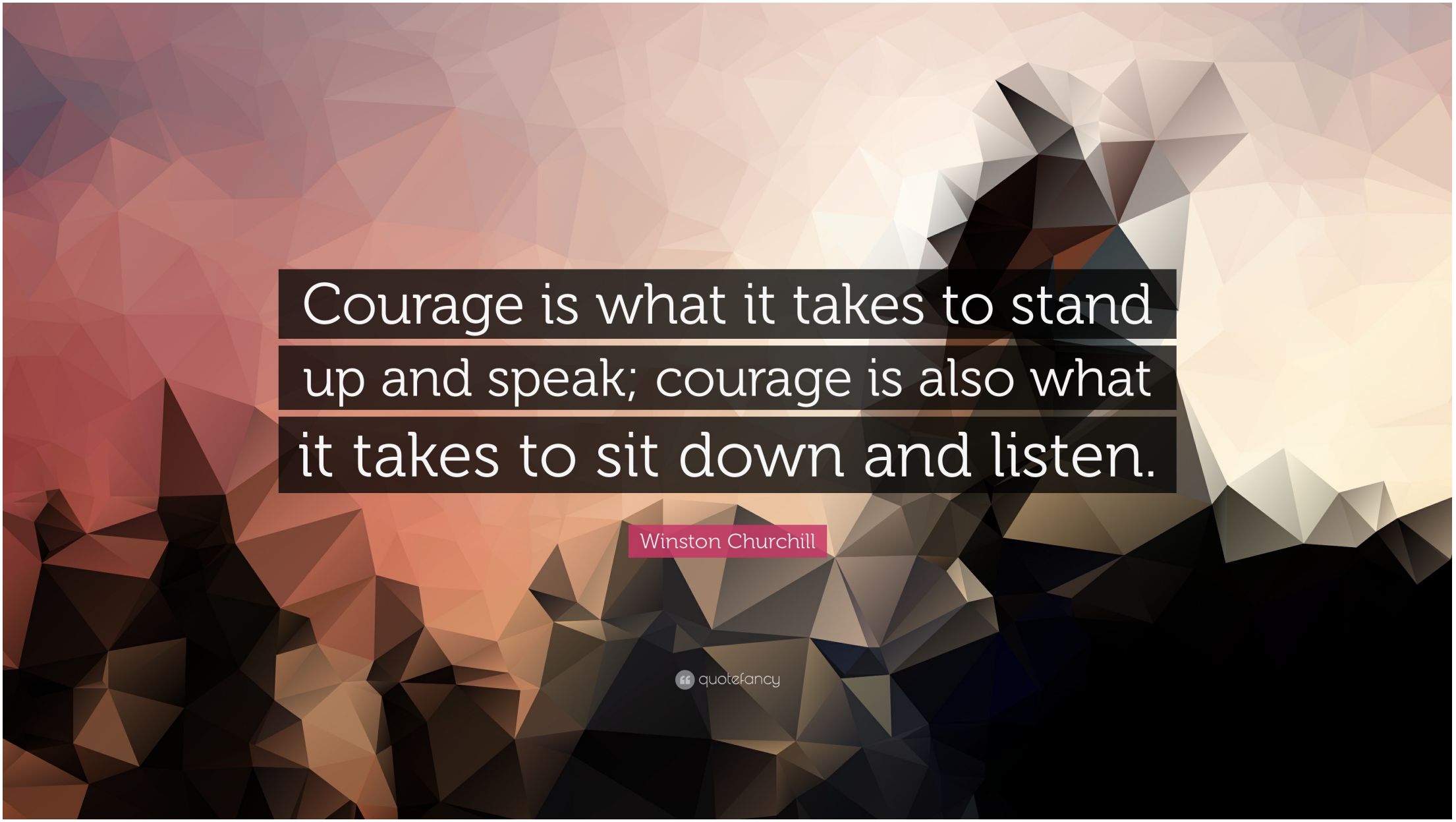
MATEUS OLIVEIRA

Big Data Architect

Data In-Motion Specialist

Use Case ~ Music Streaming using [Apache Kafka]




Courage is what it takes to stand up and speak; courage is also what it takes to sit down and listen.
Winston Churchill
quotefancy

Being deeply loved by someone gives you strength, while loving someone deeply gives you courage.

Lao Tzu

Advanced Strategies for Apache Kafka Development

Message Ordering in Apache Kafka

Tip: if ordering is necessary and we can work in producer configs then we must use only 1 partition

Apache Kafka: guarantee message ordering in two levels, the producer level with eos configuration or in partition level, for example if we have only one partition, we always will receive the events in the right order

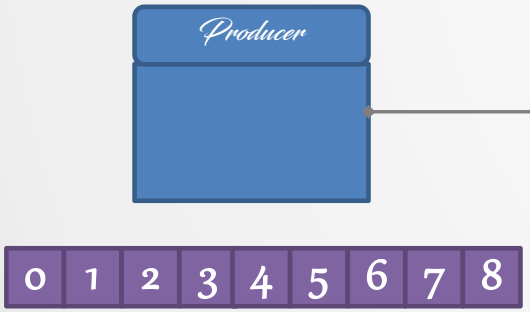
Downside: with only 1 partition its not possible to scale write and reads applications because is based on partition strategy

Apache Broker

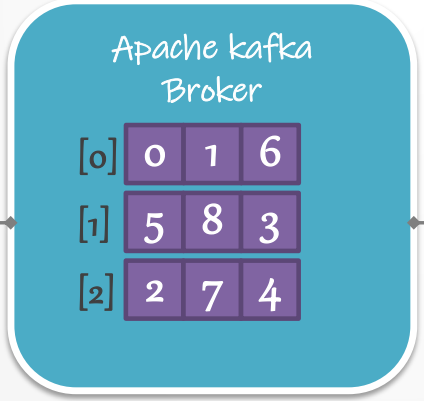
if the producer doesn't have eos enabled with the producer configurations, then kafka can't guarantee ordering with more than one partition

Producer Configs

- Exactly-Once Enabled
- max.inflight.connectors <= 5
- acks=all

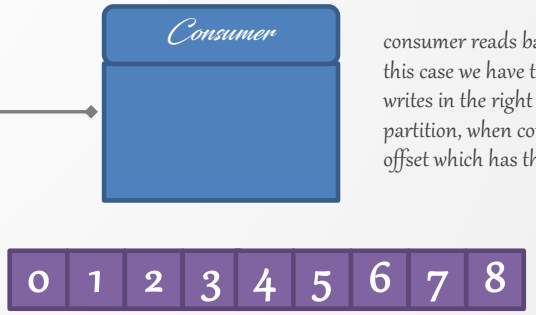


events that is going to be written into kafka sequence of number 0 - 8



Consumer Configs

order is placed on the producer side, without the need to specify on the consumption



consumer reads based on the offset and partition in this case we have the ordering because the producer writes in the right order into the topic, segregate by partition, when consume kicks in it reads the last offset which has the last number of the sequence

events will be read in sequence

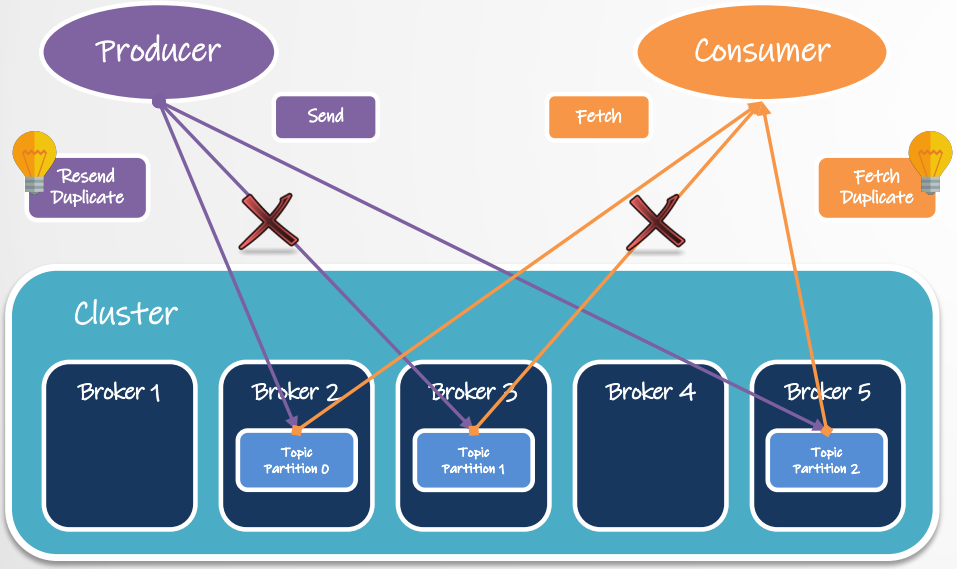
Performing Transactions with Producers & Consumers

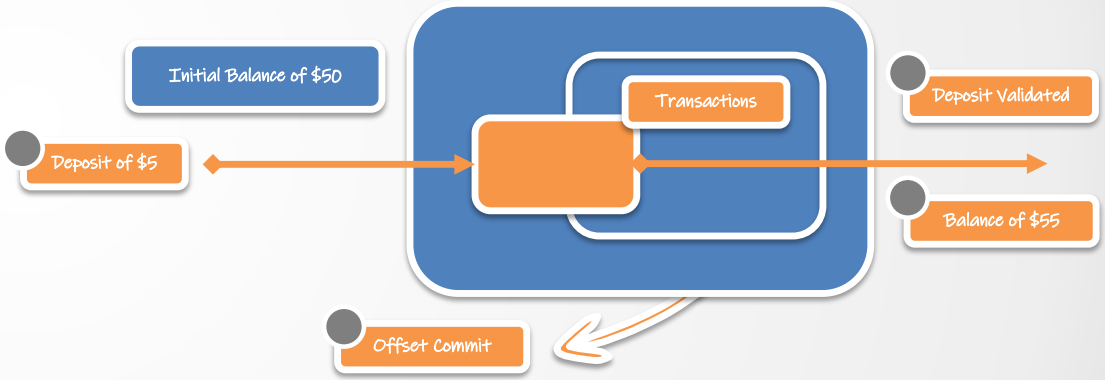
Two Opportunities for Duplicates

- **Producer** ~ Duplication Upon Retrying Write Operation
- **Consumer** ~ Failure During Increment Offset Write

Transactions ~ Multi-Operation-Commits

- **State Store** ~ Disk-Residence Hash Table Backed By Apache Kafka
- Input and Output Topics in an Atomic Way
- Stream Processors ~ Kafka Streams and KSQLDB





Transactions ~ In a Nutshell

- Stream Applications with **Read-Process-Write** Pattern
- Exactly-Once-Semantics with Atomicity & Facilitating Zombie Fencing
- Transaction Coordinator (Kafka Broker) & Log (Kafka Topic)
- 1KB Records, 100 MS ~ 3% of Degradation in Throughput

Stream-Table Duality

Stream

event stream records the history of what has happened in the world as a sequence of events

1) Immutable Data
2) Only Inserting (Appending) New Events
3) Persistent, Durable & Fault Tolerant
4) Keyed Stream (Key)

Stream

Immutable with Historic Information

Table

represents the state of the world (snapshot)

1) Mutable Data
2) Insert, Update, Delete
3) Keyed Event Identifies Row
4) Persistent, Durable & Fault Tolerant

Table

Mutable without Historic Information

Stream-Table Duality

Stream-Table

- Stream into a Table by Aggregating Events using COUNT() or SUM()
- Streams = Apache Kafka Topics with Enforced Schema

Table-Stream

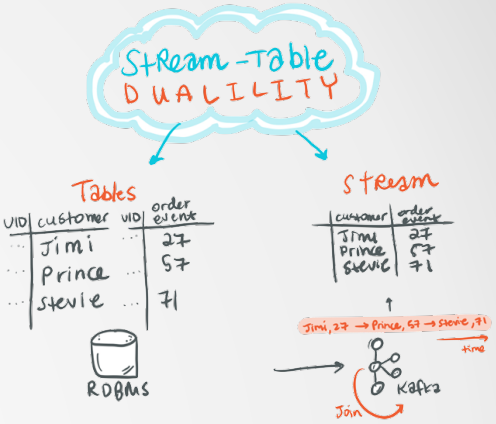
- Capture Changes (Insert, Update & Delete) into a Change Stream, (CDC)
- Tables = Topics using Log Compression (Key)

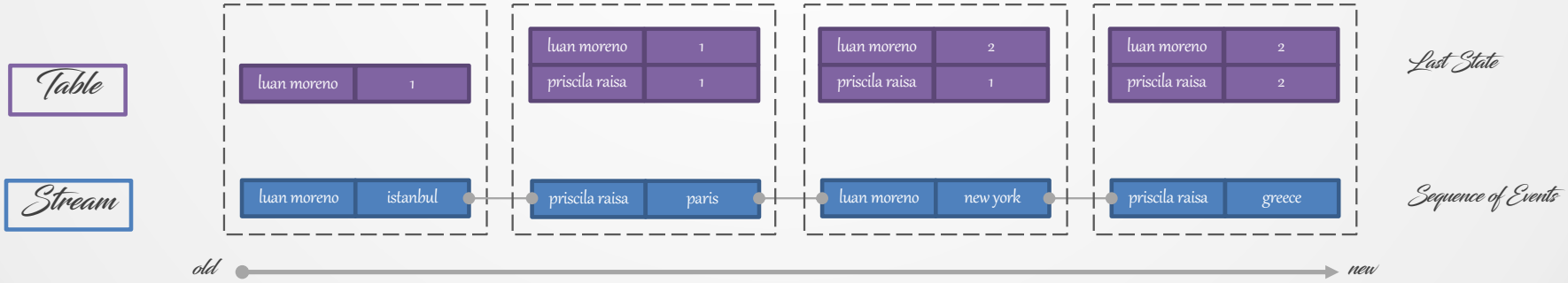
Streams = Record History

Table = Represent State

Agg = SUM | COUNT

Table Changes





Advanced Windowing [Stream-Stream Join]

Window Type	Mode	Description
Session	Session-Based	Dynamically-Sized, Non-Overlapping, Data-Driven Windows
Hopping	Time-Based	Fixed-Duration, Overlapping Windows
Tumbling	Time-Based	Fixed-Duration, Non-Overlapping, Gap-Less Windows

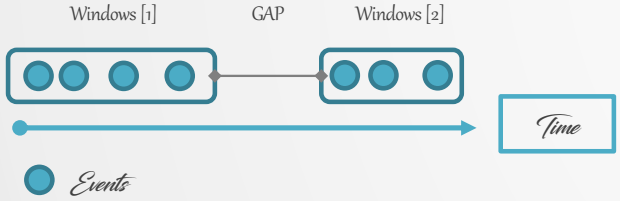
Window Types

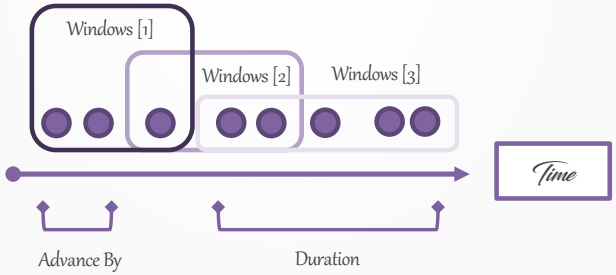
most operations on streams are windowed operations

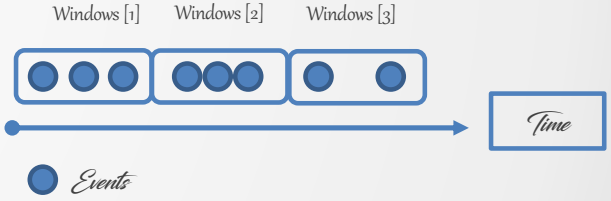
Session = aggregates records into a session, which represents a period of activity separated by a specific gap of inactivity (idleness). If a record's timestamp occurs outside of the session gap, new session is created

Hopping = based on time intervals. it is defined by two properties, duration and it's advance [HOP] interval, this HOP specifies how far a window moves forward in time relative to the previous window

Tumbling = special case of hopping, it is based in time interval too. it is defined by a single property: windows duration. this type of window never overlap a record belong once and only one window








```
--session
SELECT regionid, COUNT(*)
FROM pageviews
WINDOW SESSION (60 SECONDS)
GROUP BY regionid
EMIT CHANGES;
```

```
--hopping
SELECT regionid, COUNT(*)
FROM pageviews
WINDOW HOPPING (SIZE 30 SECONDS, ADVANCE BY 10 SECONDS)
GROUP BY regionid
EMIT CHANGES;
```

```
--tumbling
SELECT orderzip_code, TOPK(order_total, 5)
FROM orders
WINDOW TUMBLING (SIZE 1 HOUR)
GROUP BY order_zipcode
EMIT CHANGES;
```

Advanced Strategies for Apache Kafka Administration

Log Skimming on Apache Kafka using ElasticSearch

Apache Kafka

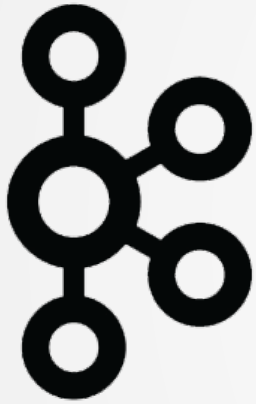
produce logs of all his activities

Filebeat

access log folders and copy the logs info inside elasticsearch

Elasticsearch

database where logs are stored, fully integrated with filebeat and kibana



Kibana

visualize and explore logs of data captured by elasticsearch

Info

information about activities, some troublesome behaviors

Warn

information about out of normal behavior

Error

information about critical error, most import ones

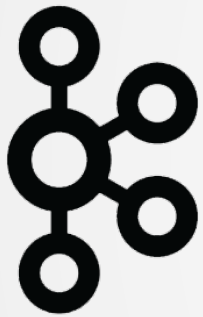
Storage Resizing for Apache Kafka

JBOD=Just a Bunch of Disks

in KIP-112 we have jbod handle fix, and we recommend the use in kafka deployments, specially if we are talking about kubernetes deployments, we don't need to resize only add more disk

Broker

apache kafka core, stores the events written by producers



before rebalance

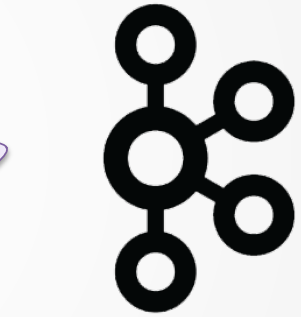
Partition Reassign

after we add new disks, we must run reassign partitions command

after rebalance

Broker

apache kafka core, stores the events written by producers



resizing the disk to increase the volume capacity

adding more disks

after rebalance the data is distributed between all disks

Apache Kafka Partition Reassign - Rebalance

Partition Reassign

Apache Kafka is a storage in real-time for streaming events which depend on disks to store segments, and we must be careful to not reach 100% disk occupation, but if we need to add more disks, we need to execute a rebalance equals to partition reassign command line after add the new disks

Warnings

- only execute out of business hours, the topics becomes offline during the rebalance, nobody writes or reads
- simple math 10 topics with 8 partitions each and data from 1mi to 1.000 can take something close to 30 seconds to rebalance, but this depends a lot of how unbalanced are things and if we are using SSD

How partition reassign works

List Topics & Create JSON File

create a json file base in the topics in this pattern:
{ "topics": [{ "topic": "_consumer_offsets" }, { "topic": "_schemas" }

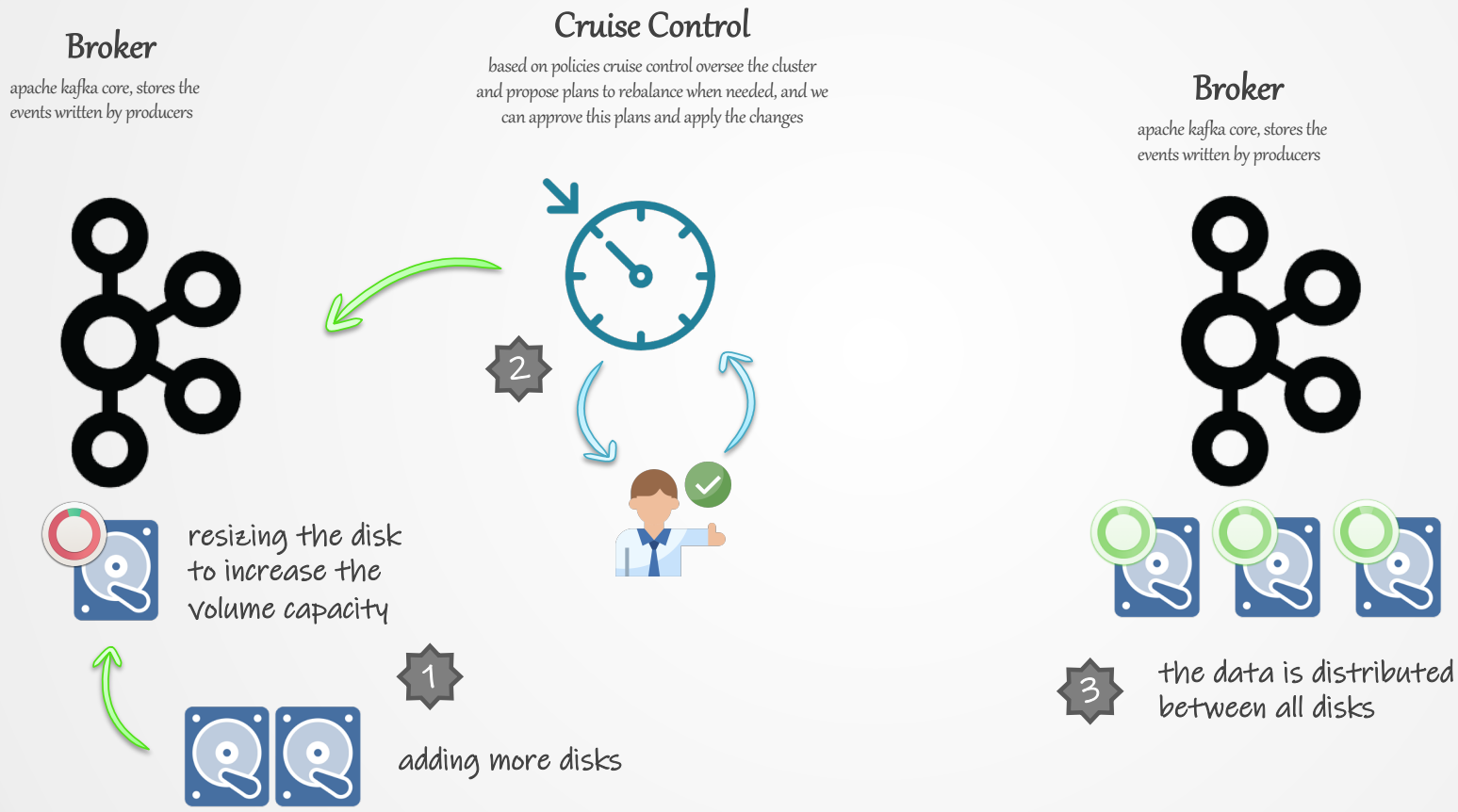
Create a Reassign JSON File with Output from Command Line

"kubectl exec edh-kafka-0 -c kafka -i -t -- bin/kafka-reassign-partitions.sh --bootstrap-server localhost:9092 --broker-list "0,1,2" --topics-to-move-json-file /tmp/topics-to-move.json –generate"

Execute Command with JSON Reassign File

"kubectl exec edh-kafka-0 -c kafka -i -t -- bin/kafka-reassign-partitions.sh --bootstrap-server localhost:9092 --reassignment-json-file /tmp/cluster-reassign.json –execute"

Automating Administrative Tasks using Cruise Control



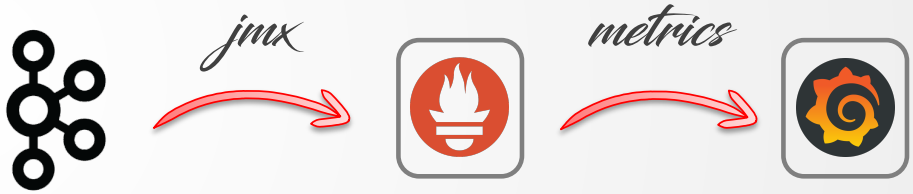
Monitoring [Apache Kafka] with Prometheus & Grafana

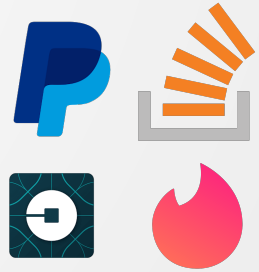
Prometheus

- Open-Source System for Monitoring & Alerting ~ 2012
- Time-Series DB ~ Streams of Timestamped Values
- Built-In ~ Scraping, Storing, Querying & Graphing System
- PromQL ~ Flexible Query Language
- Pull over HTTP
- Written in Go

Grafana

- Open-Source Visualization & Analytics Software
- Query, Visualize, Alert & Metrics
- Time Series Database ~ Graphs & Visualizations
- Data Sources ~ Prometheus, ElasticSearch, MySQL, PostgreSQL





Monitoring Apache Kafka with ElasticSearch and Prometheus

DataOps with Lenses

Deploy Everywhere

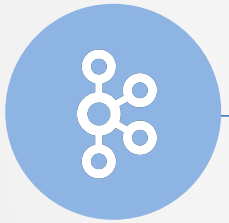
- Virtual Machines (VMs)
- Docker & Kubernetes
- Public Cloud

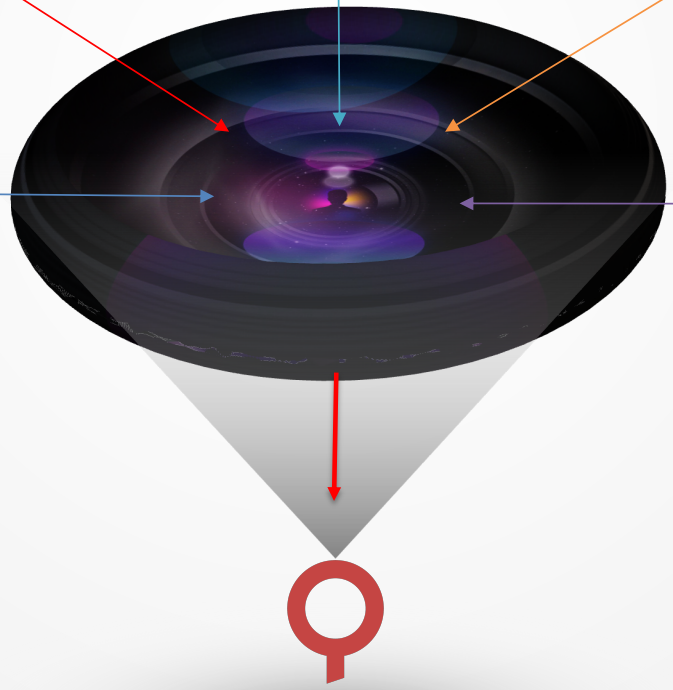
Features

- App Catalog
- Centralized Schema Management
- Consumer Lag Monitoring
- Custom Alert Routing
- Data Catalog
- Data Discovery
- Data Injection
- Data Masking Rule

Fancy UI

Explore Topics, Events, Schemas, Partitions, Health of Connectors, Apache Brokers & Apache Zookeepers





Observability Apache Kafka

Lenses Gives a 360° View of Apache Kafka Components:

- Apache Broker
- Apache Zookeeper
- Kafka Connect
- Schema Registry

Explore Events using SQL

Query Data using SQL with Abstraction of Streams and Tables with Similar KSQLDB Experience

DataOps using Lenses

Advanced Deployment Techniques for Apache Kafka

Apache Kafka & Confluent [Services]

Apache Kafka [Vanilla]

Kafka Broker
Kafka Connect
Schema Registry (Open-Source)
KSQLDB (Open-Source)

Rest Proxy

Client Access with
Rest API Requests

Confluent Schema Registry

Schema Enforcement
Schema Evolution

Control Center

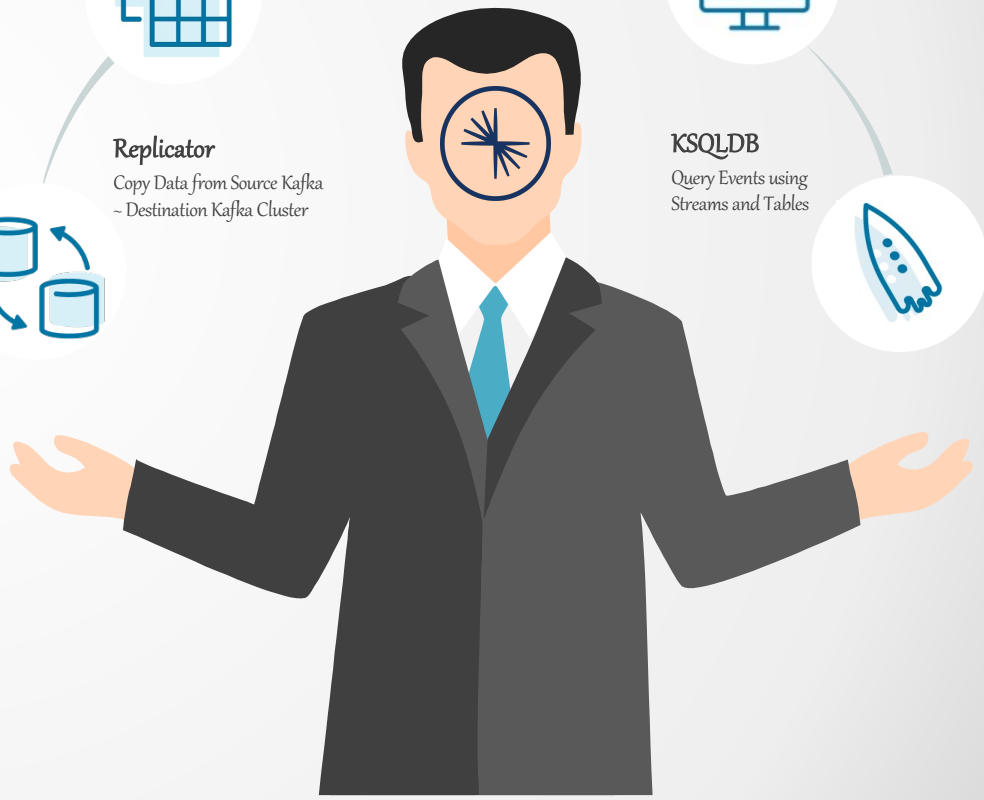
UI for Kafka Management

Replicator

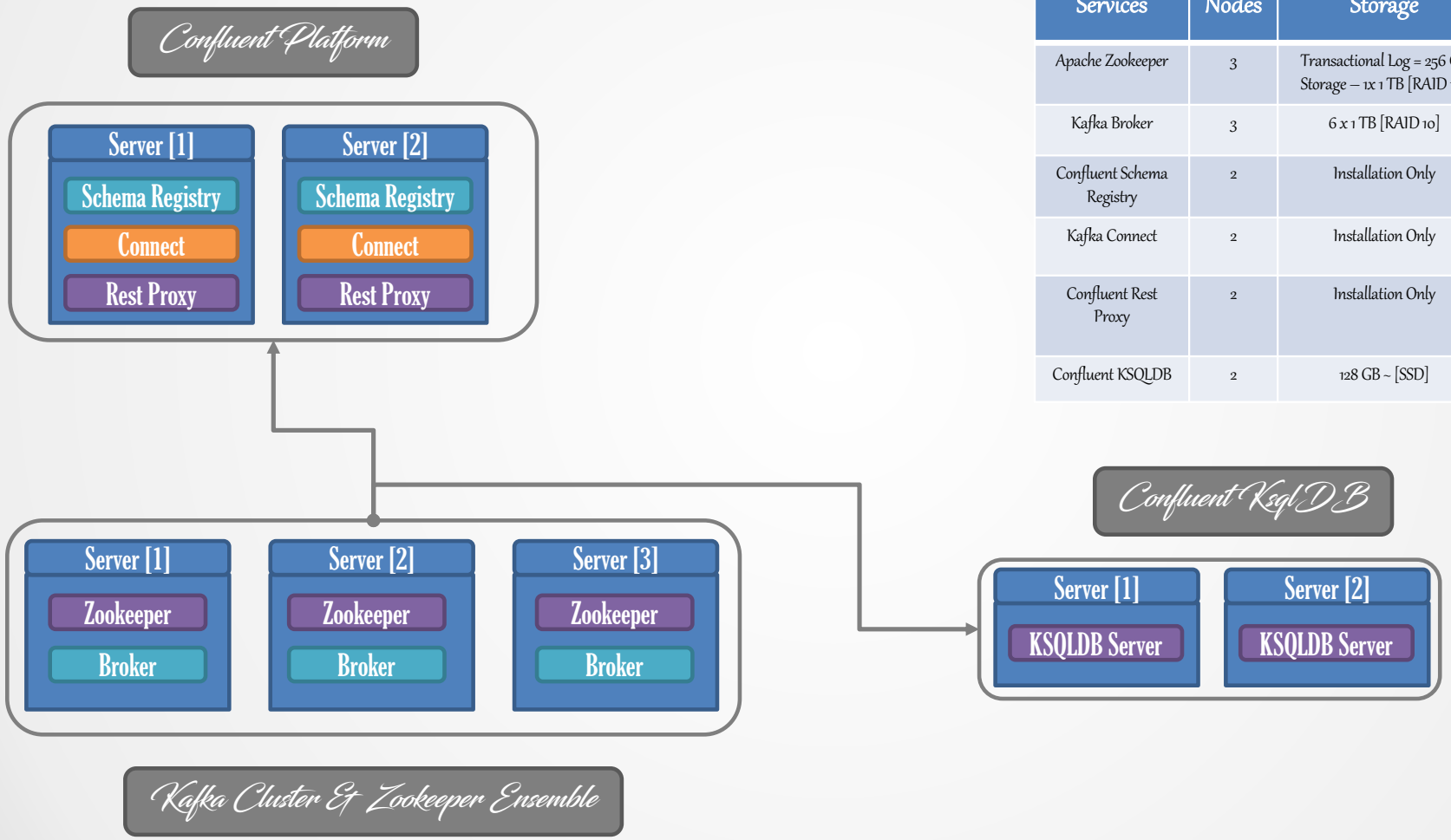
Copy Data from Source Kafka
~ Destination Kafka Cluster

KSQLDB

Query Events using
Streams and Tables



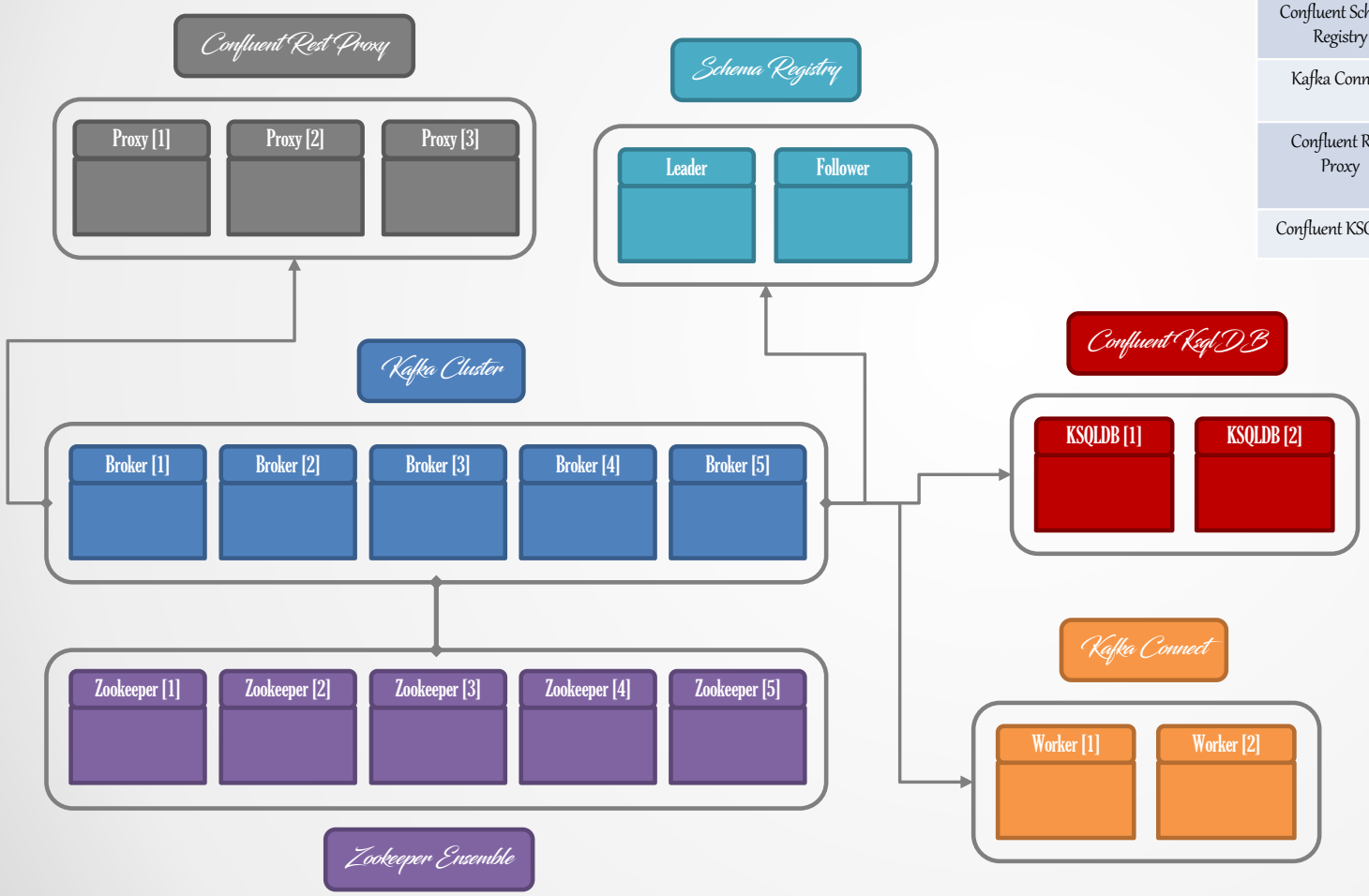
Apache Kafka & Confluent [Architecture] ~ Small

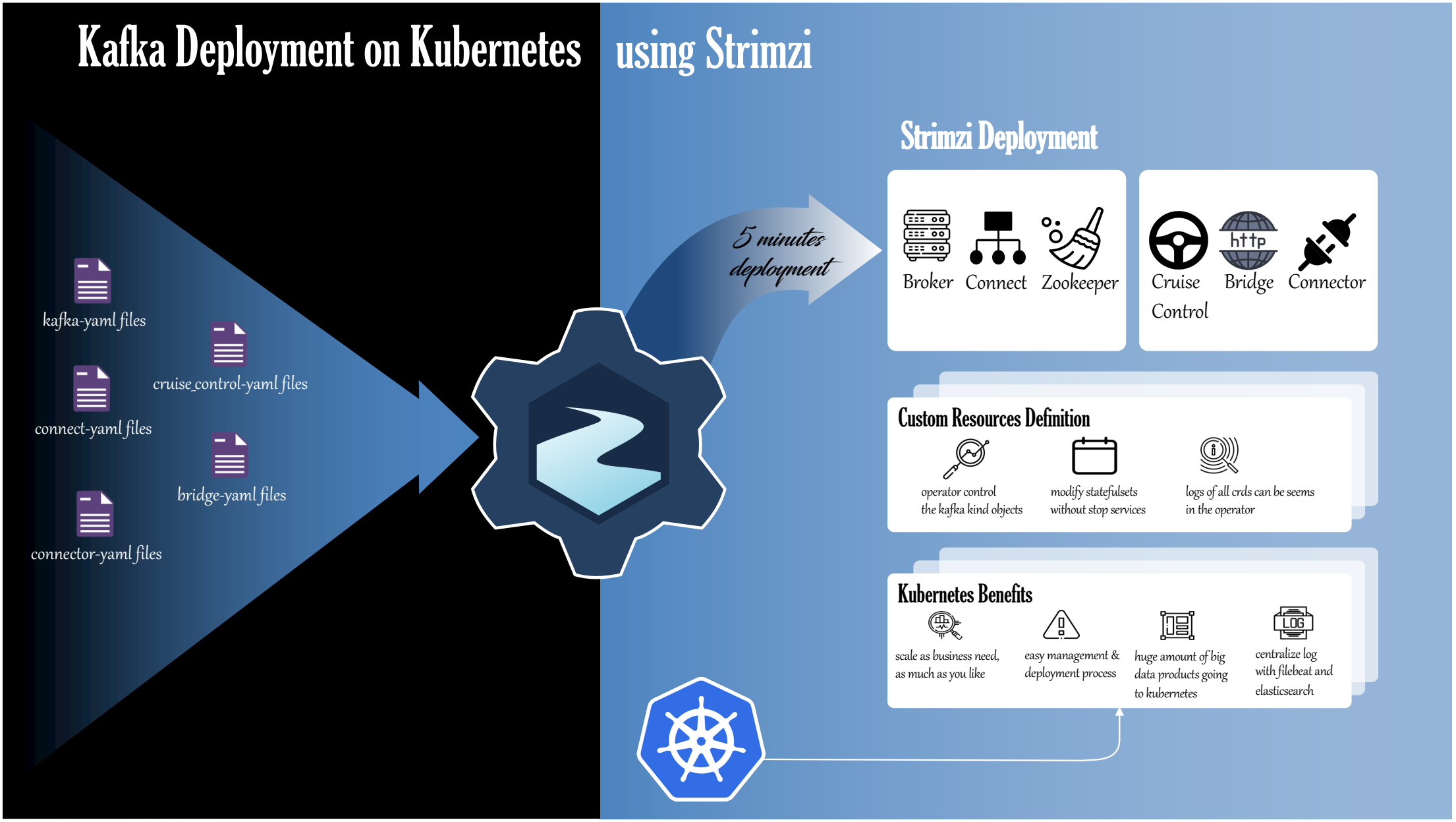


Services	Nodes	Storage	Memory	CPU
Apache Zookeeper	3	Transactional Log = 256 GB Storage – 1x 1 TB [RAID 10]	16 GB	2~4
Kafka Broker	3	6 x 1 TB [RAID 10]	32 GB +	6
Confluent Schema Registry	2	Installation Only	1 GB Heap Size	2~4
Kafka Connect	2	Installation Only	2 GB Heap Size	2~4
Confluent Rest Proxy	2	Installation Only	Producer = 512 MB Consumer = 16 MB	16
Confluent KSQLDB	2	128 GB ~ [SSD]	10 GB	2

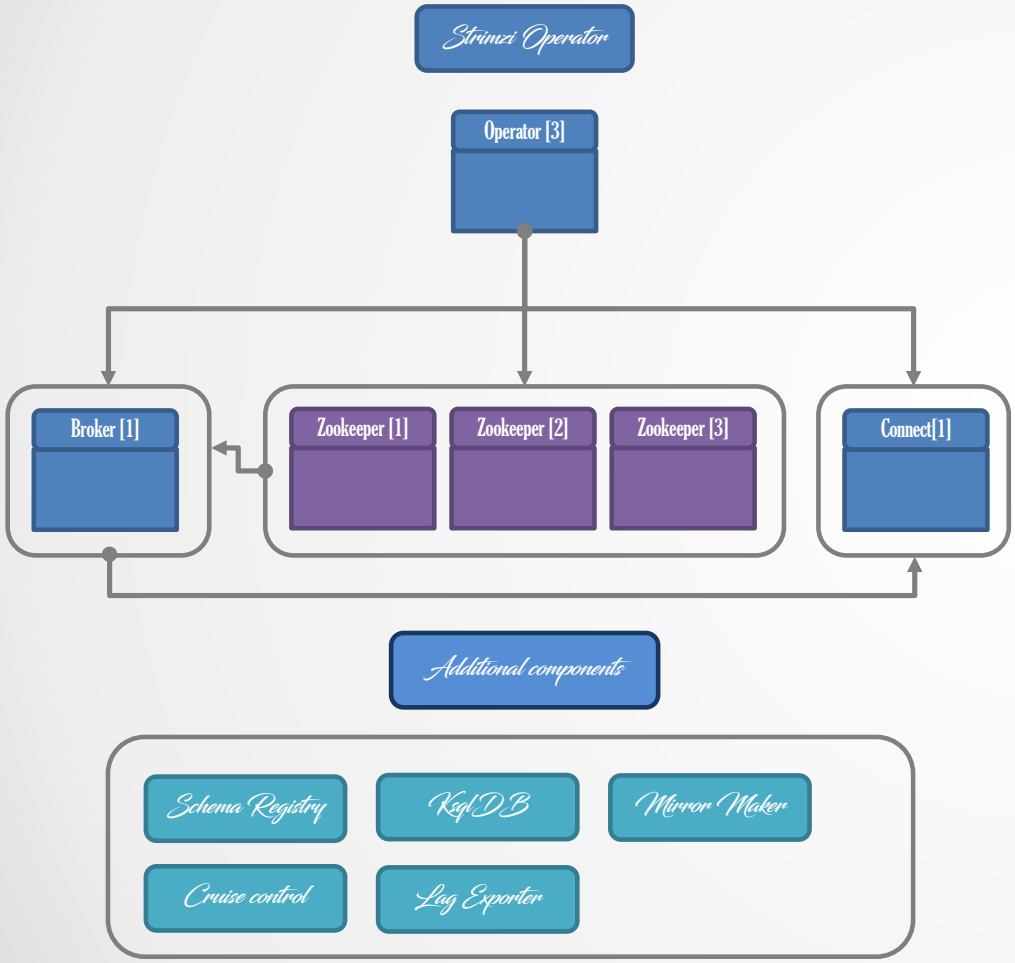
Apache Kafka & Confluent [Architecture] ~ Large

Services	Nodes	Storage	Memory	CPU
Apache Zookeeper	5	Transactional Log = 512 GB Storage – 2x 1 TB [RAID 10]	32 GB	2~4
Kafka Broker	5	12 x 1 TB [RAID 10]	64 GB +	12
Confluent Schema Registry	2	Installation Only	2 GB Heap Size	2~4
Kafka Connect	2	Installation Only	4 GB Heap Size	2~4
Confluent Rest Proxy	3	Installation Only	Producer = 1 GB Consumer = 32 MB	16
Confluent KSQLDB	2	256 GB ~ [SSD]	20 GB	4

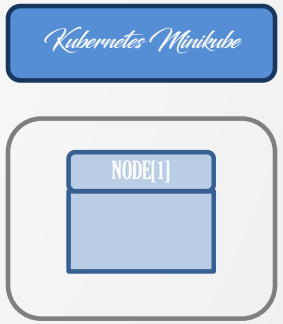



Kafka Deployment on Kubernetes
using Strimzi
kafka-yaml files
cruise_control-yaml files
connect-yaml files
bridge-yaml files
connector-yaml files
5 minutes deployment
Strimzi Deployment
Broker
Connect
Zookeeper
Cruise Control
http
Bridge
Connector
Custom Resources Definition
operator control the kafka kind objects
modify statefulsets without stop services
logs of all crds can be seems in the operator
Kubernetes Benefits
scale as business need, as much as you like
easy management & deployment process
huge amount of big data products going to kubernetes
LOG
centralize log with filebeat and elasticsearch

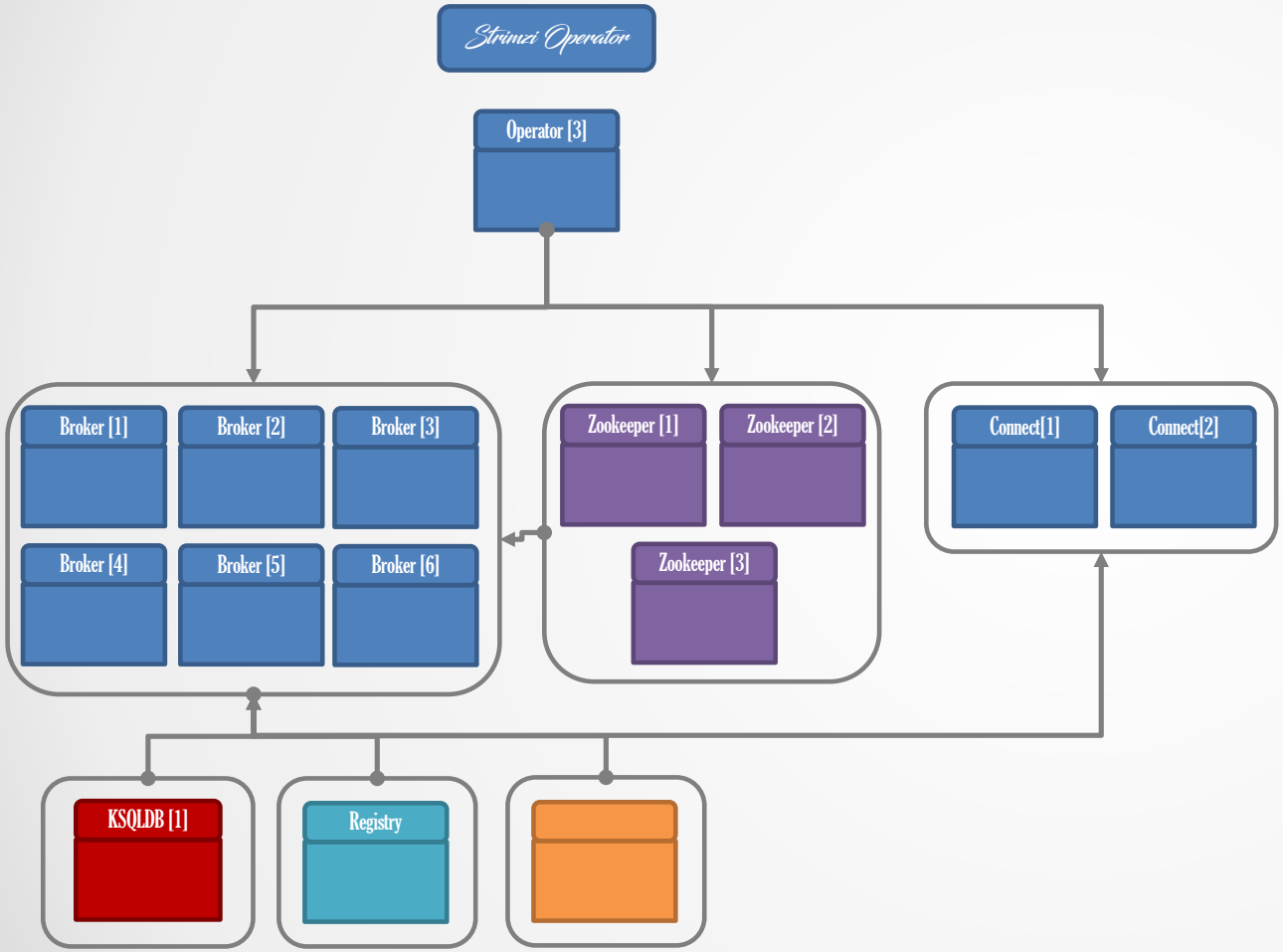
Kafka Deployment on Kubernetes [K*S] ~ Small



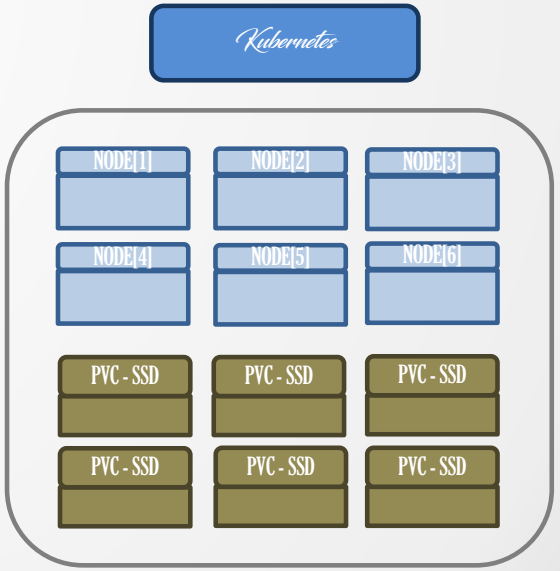
Services	Pods	Storage	Memory	CPU
Strimzi Operator	1	Ephemeral	Requests = 348Mi Limits = 348Mi	Requests = 200m Limits = 1000m
Apache Zookeeper	3	5Gi	Requests = 256Mi Limits = 512Mi	Requests = 250m Limits = 500m
Kafka Broker	1	40Gi or Ephemeral	Requests = 500Mi Limits = 1Gi	Requests = 250m Limits = 500m
Kafka Connect	1	Ephemeral	Requests = 500Mi Limits = 1Gi	Requests = 250m Limits = 500m
Strimzi bridge	1	Ephemeral	Requests = 100Mi Limits = 200Mi	Requests = 100m Limits = 200m
Confluent Schema Registry	1	Ephemeral	Requests = 100Mi Limits = 100Mi	Requests = 100m Limits = 200m
Confluent KSQL	1	Ephemeral	Requests = 512Mi Limits = 2Gi	Requests = 250m Limits = 500m

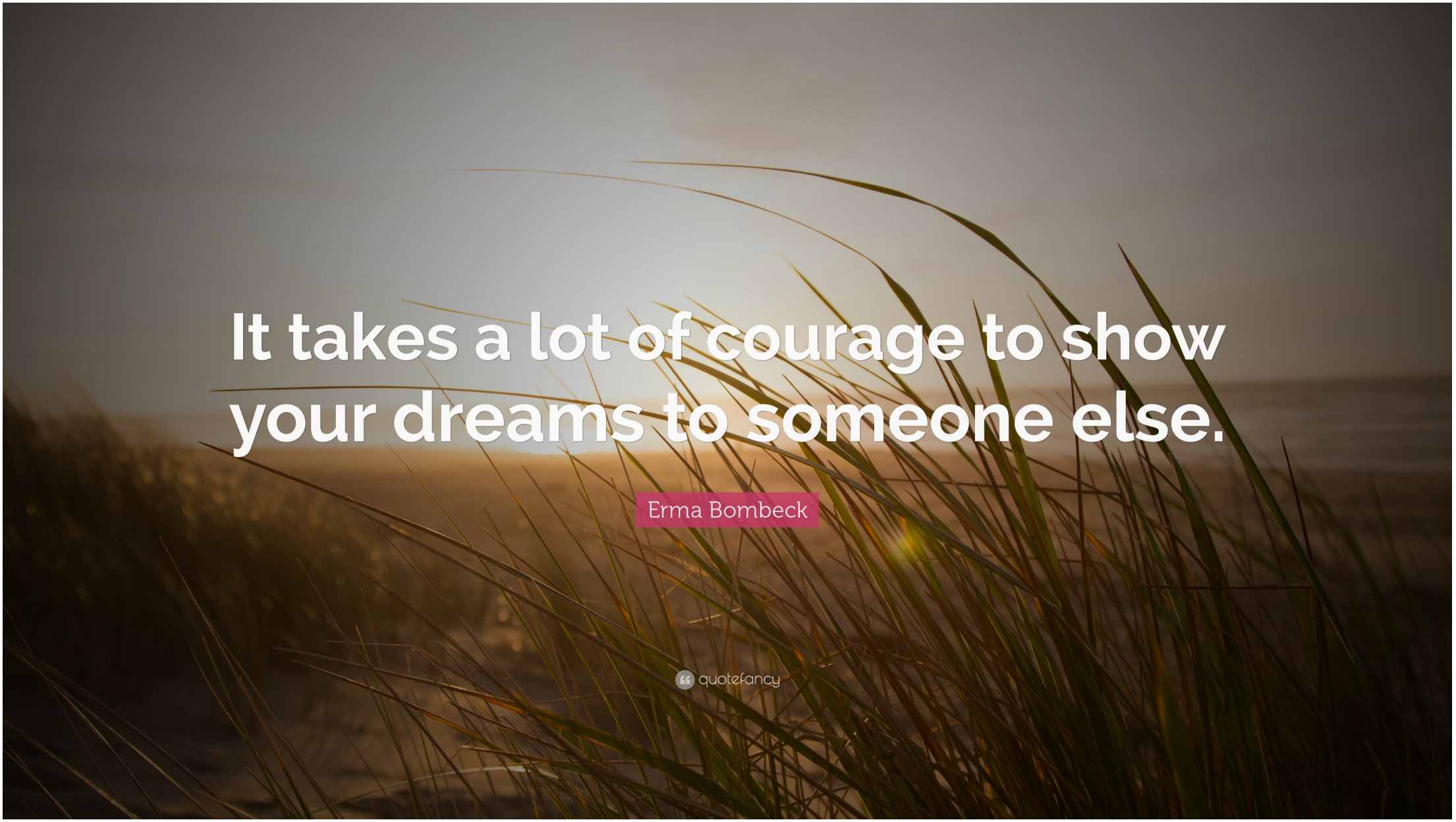


Kafka Deployment on Kubernetes [K*S] ~ Large



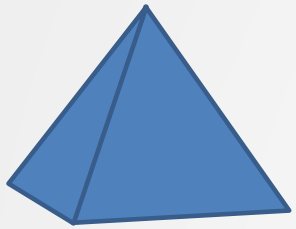
Services	Pods	Storage	Memory	CPU
Strimzi Operator	1	Ephemeral	Requests = 348Mi Limits = 348Mi	Requests = 200m Limits = 1000m
Apache Zookeeper	3	5Gi	Requests = 256Mi Limits = 512Mi	Requests = 250m Limits = 500m
Kafka Broker	6	100Gi JBOD	Requests = 1Gi Limits = 2Gi	Requests = 250m Limits = 500m
Kafka Connect	2	Ephemeral	Requests = 1Gi Limits = 2Gi	Requests = 250m Limits = 500m
Strimzi bridge	1	Ephemeral	Requests = 100Mi Limits = 200Mi	Requests = 100m Limits = 200m
Confluent Schema Registry	1	Ephemeral	Requests = 100Mi Limits = 100Mi	Requests = 100m Limits = 200m
Confluent KSQL	1	Ephemeral	Requests = 1Gi Limits = 3Gi	Requests = 500m Limits = 1000m




It takes a lot of courage to show your dreams to someone else.
Erma Bombeck
quotefancy

Best Practices for Apache Kafka

Optimizing Kafka for [Throughput]

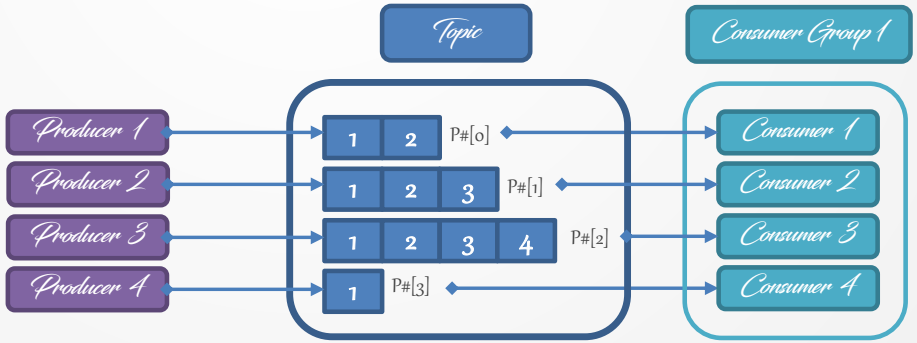


Optimize for Throughput

to optimize for throughput, the producers, brokers, and consumers need to move as much data as they can within a given amount of time. for high throughput, you are trying to maximize the rate at which this data moves. this data rate should be as fast as possible. a topic partition is the unit of parallelism in kafka. messages to different partitions can be sent in parallel by producers, written in parallel by different brokers, and read in parallel by different consumers. in general, a higher number of topic partitions results in higher throughput, and to maximize throughput, you want enough partitions to utilize all brokers in the cluster.

Producer	Value	Default
Batch.Size	100.000 ~ 200.000	16.384
Linger.Ms	10 ~ 100	0
Compression.Type	LZ4	Without Compression
ACKS	1	1
Buffer Memory	Increase [Partitions]	33.554.432

Consumer	Value	Default
Fetch.Min.Bytes	100.000	0

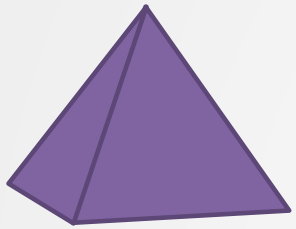


Optimizing Kafka for [Latency]

Broker	Value	Default
Num.Replicas.Fetchers	Increase [Followers]	1

Producer	Value	Default
Linger.Ms	0	0
Compression.Type	NONE	Without Compression
ACKS	1	1

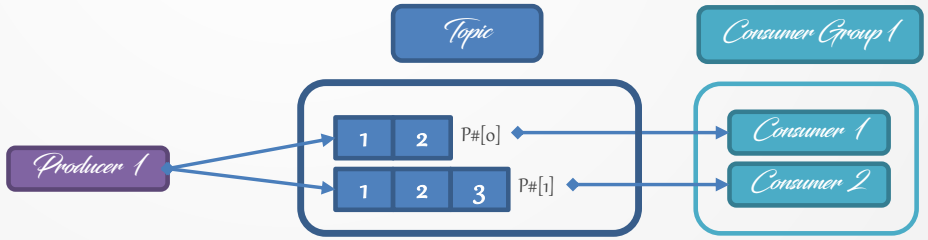
Consumer	Value	Default
Fetch.Min.Bytes	1	1



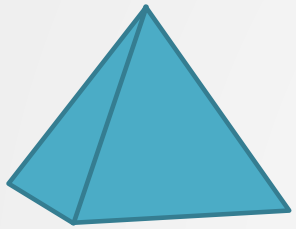
Optimize for Latency

because a partition is a unit of parallelism in kafka, an increased number of partitions may increase throughput. however, there is a tradeoff in that an increased number of partitions may also increase latency. a broker by default uses a single thread to replicate data from another broker, so it may take longer to replicate a lot of partitions shared between each pair of brokers and consequently take longer for messages to be considered committed. no message can be consumed until it is committed, so this can ultimately increase end-to-end latency.

Latency

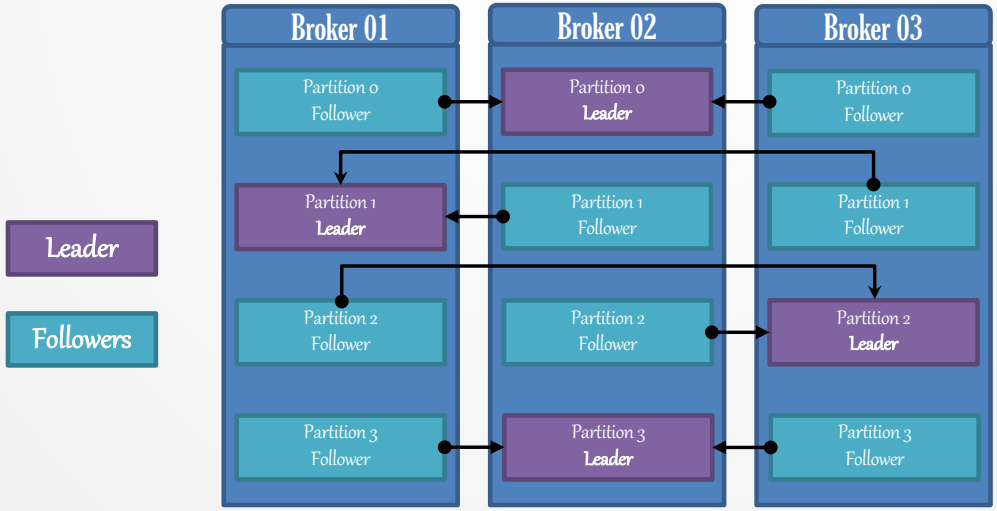


Optimizing Kafka for [Durability]



Optimize for Durability

durability is all about reducing the chance for a message to get lost. the most important feature that enables durability is replication, which ensures that messages are copied to multiple brokers. if a broker has a failure, the data is available from at least one other broker. topics with high durability requirements should have the configuration parameter which will ensure that the cluster can handle a loss of two brokers without losing the data.



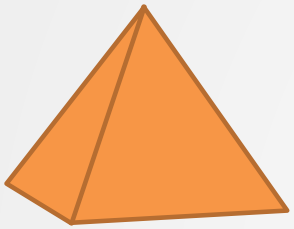
Producer	Value	Default
ACKS	ALL	1
Enable.Idempotence	True	False
Max.In.Flight.Requests .per.Connection	1	5

Streams	Value	Default
Replication Factor Configuration	3	1
Exactly Once Semantics	True	At Least Once

Consumer	Value	Default
Enable.Auto.Commit	False	True
Isolation.Level	Read Committed [EOS]	NONE

Broker	Value	Default
Default.Replication.Factor	3	1
Auto.Create.Topics.Enable	False	True
Min.InSync.Replicas	2	1
Unclean.Leader.Election.Enable	False	False
Log.Flush.Interval Message & MS	Low	OS

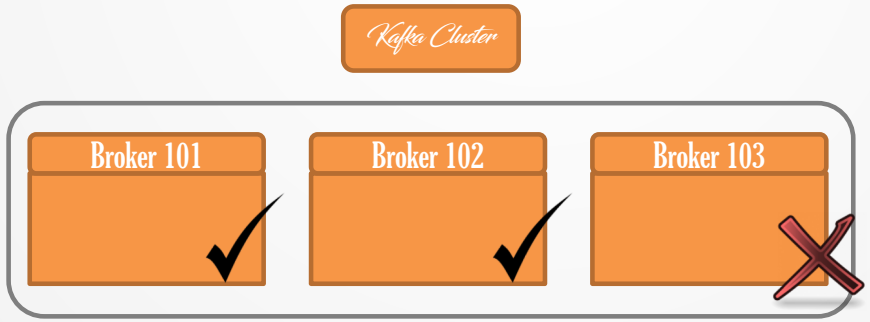
Optimizing Kafka for [Availability]



Optimize for Availability

optimize for high availability, you should tune kafka to recover as quickly as possible from failure scenarios. higher partition counts may increase parallelism but having more partitions can also increase recovery time in the event of a broker failure. all produce and consume requests will pause until the leader election completes, and these leader elections happen per partition. so, take this recovery time into consideration when choosing partition counts.

Availability

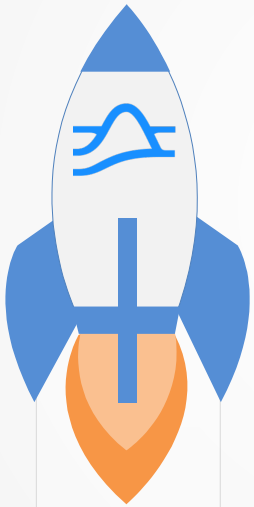


Streams	Value	Default
Stand By Replicas Configuration	1	0

Consumer	Value	Default
Session.Timeout.Ms	3.000	10.000

Broker	Value	Default
Unclean.Leader.Election.Enable	True	False
Min.InSync.Replicas	1	1
Num.Recovery.Threads.per.Data.Dir	5	1

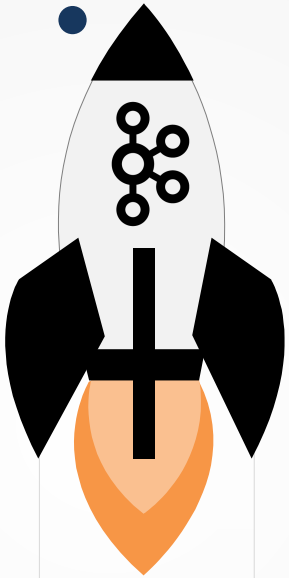
Benchmarking [Apache Kafka]



Apache Pulsar

Throughput (MB/s)
305 MB/s

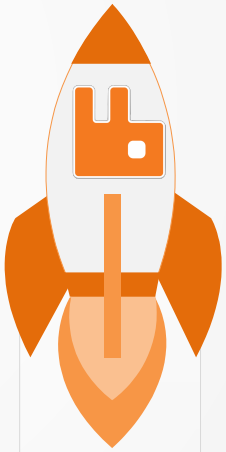
p99 latency (ms)
25 ms
(200 MB/s Load)



Apache Kafka

Throughput (MB/s)
605 MB/s

p99 latency (ms)
5 ms
(200 MB/s Load)



RabbitMQ

Throughput (MB/s)
38 MB/s

p99 latency (ms)
1 ms
(Reduced 30 MB/s Load)

Apache Kafka Security [Features]

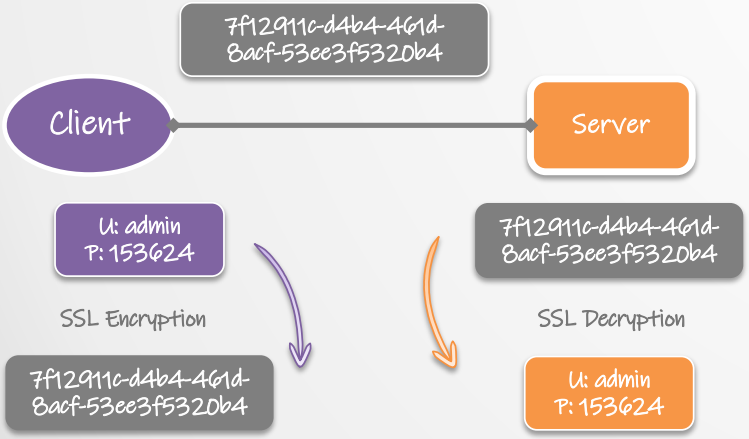
Apache Kafka is the Wild-West without Security

Problems Security Solves

- **Encryption of Data In-Flight using SSL & TLS** ~ Encrypted Between Producers, Apache Kafka & Consumers
- **Authentication using SSL or SASL** ~ Producers & Consumers Authenticate on Apache Kafka Cluster. Identify Verification
- **Authorization using ACLs** ~ Access Control List for Authorized ~ Read & Write Topics

Encryption SSL

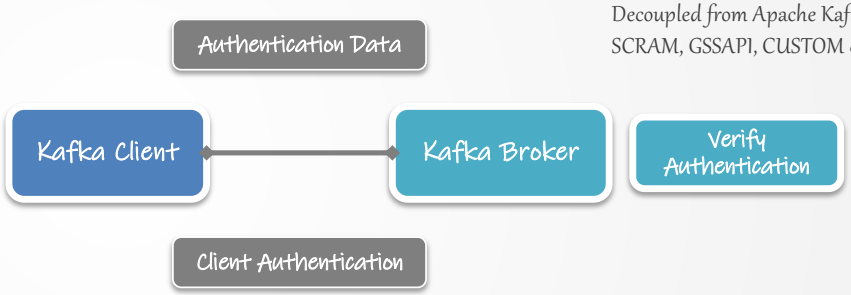
for in-flight encryption only, disk is unencrypted



Authentication (SSL & SASL)

two ways to authenticate the kafka clients

- **SSL Authentication** ~ 2-Ways Authentication By Issuing Certificates for Clients Signed By Authority (Most-Common)
- **SASL Authentication** ~ Simple Authorization Service Layer, Decoupled from Apache Kafka Protocol, Types: PLAINTEXT, SCRAM, GSSAPI, CUSTOM & OAUTHBEARER



Authorization (ACL)

permissions to access resources

user luan.moreno can view topic transactions

user mateus.oliveira cannot view topic users


Freedom lies in being bold.
Robert Frost
quotefancy

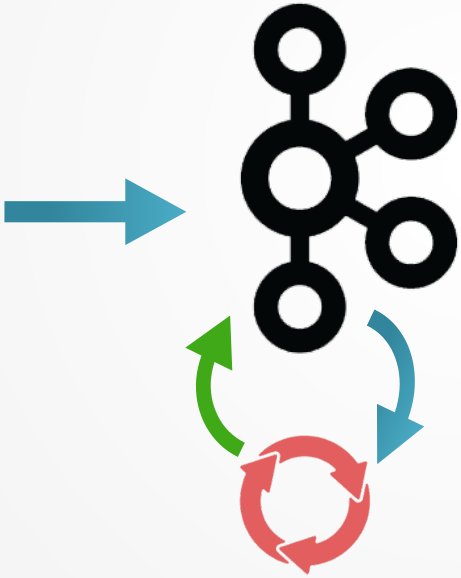
Business Use-Cases - Architecture Scenarios for Apache Kafka

Data Comes from Everywhere

data comes from databases, applications, need to work with financial data, geo-special data, commerce data, multiple formats and different schemas, sometimes even without schema, and we need in real-time but demand for batching

Central Nervous System

with data comes from all places we need a platform that can ingest from many formats and be access from multiple consumers, all data in organization flows through this platform call apache kafka



Streaming Processing

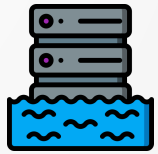
some process to transform the data into meaningful data for the organization

APIs

applications consuming from kafka to access the output events, microservices era

Data Warehouses

output data into modern data warehouses to be consume by visualization apps



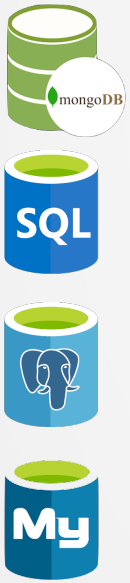
Data Lake

output data or raw data sink into data lake to be consume by data scientists and other data users.

Apache Kafka as Integration Hub using KafkaConnect

Data Sources

databases such mongodb, sql server, postgres and mysql

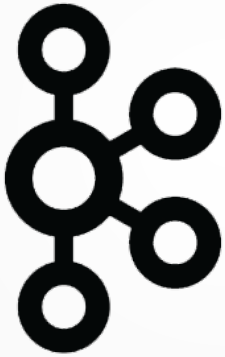


Kafka Connect Source

connectors build using kafka connect api to connect into databases and extract data

Apache Kafka

apache kafka receive database data using kafka connect

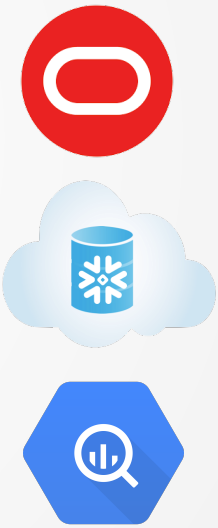


Kafka Connect Sink

connectors build using kafka connect api to connect into databases and load data

Data Stores

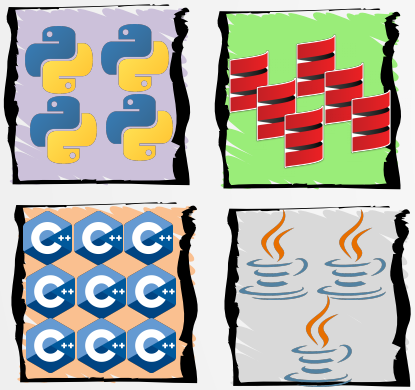
databases such oracle, snowflake, bigquery



Apache Kafka as a Shock Absorber System

APIs - Microservices

apis from different languages and services send data to data layer, somethings need to scale such as black Friday scenarios



Apache Kafka

apache kafka works as shock absorber receiving the data from apis and scale if is needed

Brokers Scale-Out

Kafka Connect Sink

connectors build using kafka connect api to connect into databases and load data, inserting data in the pace that database can handle, and the data still store in kafka if need reprocessing or inserting for more time

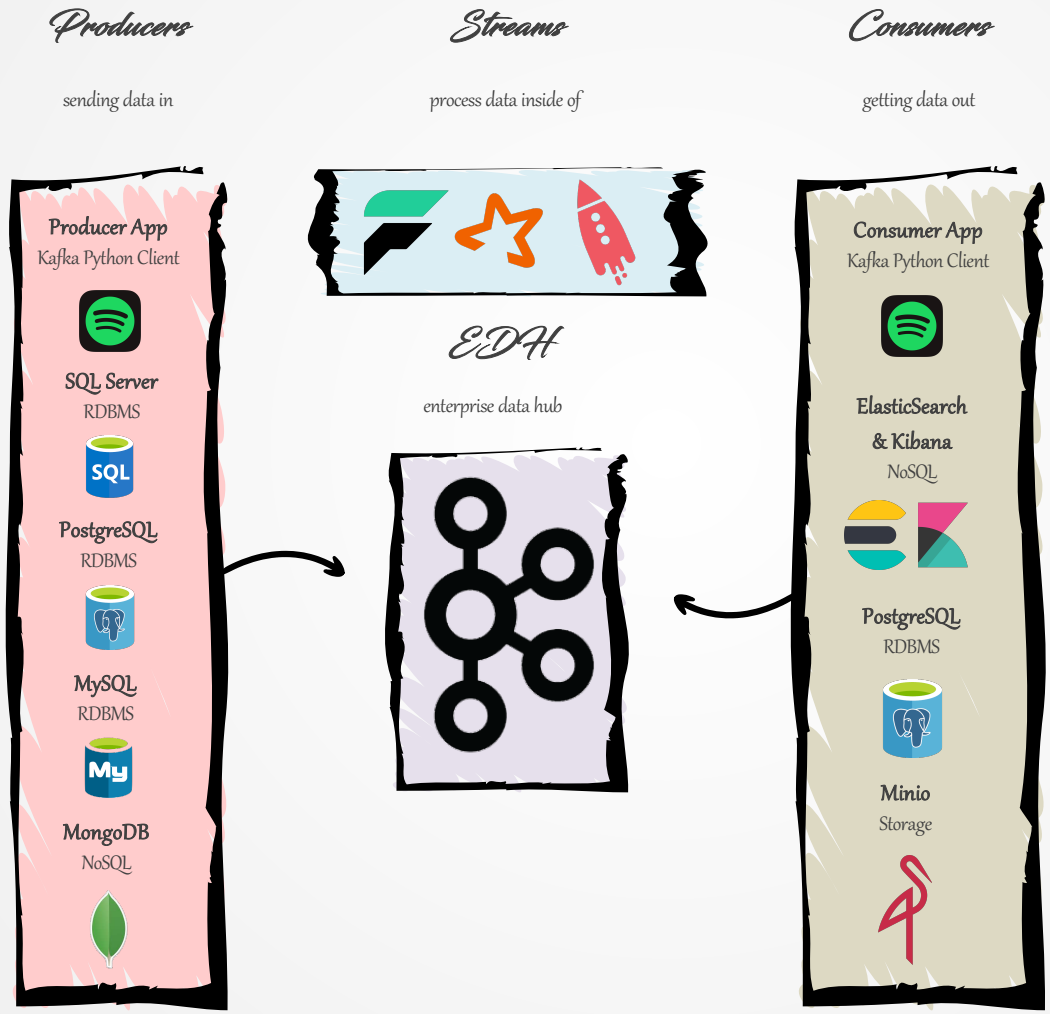
Database

relational databases, not distributed systems


Without fear there cannot be courage.
Christopher Paolini
quotefancy

Use-Case – Sneak Peak

Kappa Architecture



[Orion] - Big Data as a Service

data-driven architecture for microservices & data pipelines. this process allows tighten and seamless integration between applications and data stores for analytics at scale.

Kappa Architecture

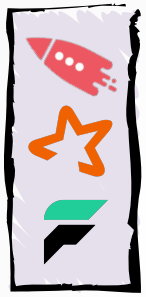
Application Ingestion Layer
Kafka Producer API

Data Store Ingestion Layer
Kafka Connect Source API

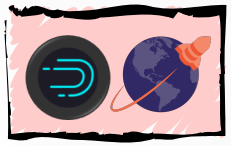
Enterprise Data Hub [EDH]
Apache Kafka [Strimzi]

Deep Storage
MinIO

Processing Layer
KSQLDB, Apache Spark & Faust



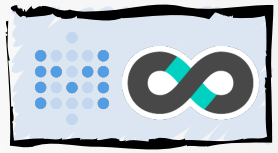
OSS Data Serving
Apache Druid & YugaByteDB



Cloud Data Serving
Redshift, BigQuery, Synapse Analytics

Integration & Exploration Layer
Trino & Zeppelin

Data Visualization
Metabase & Superset



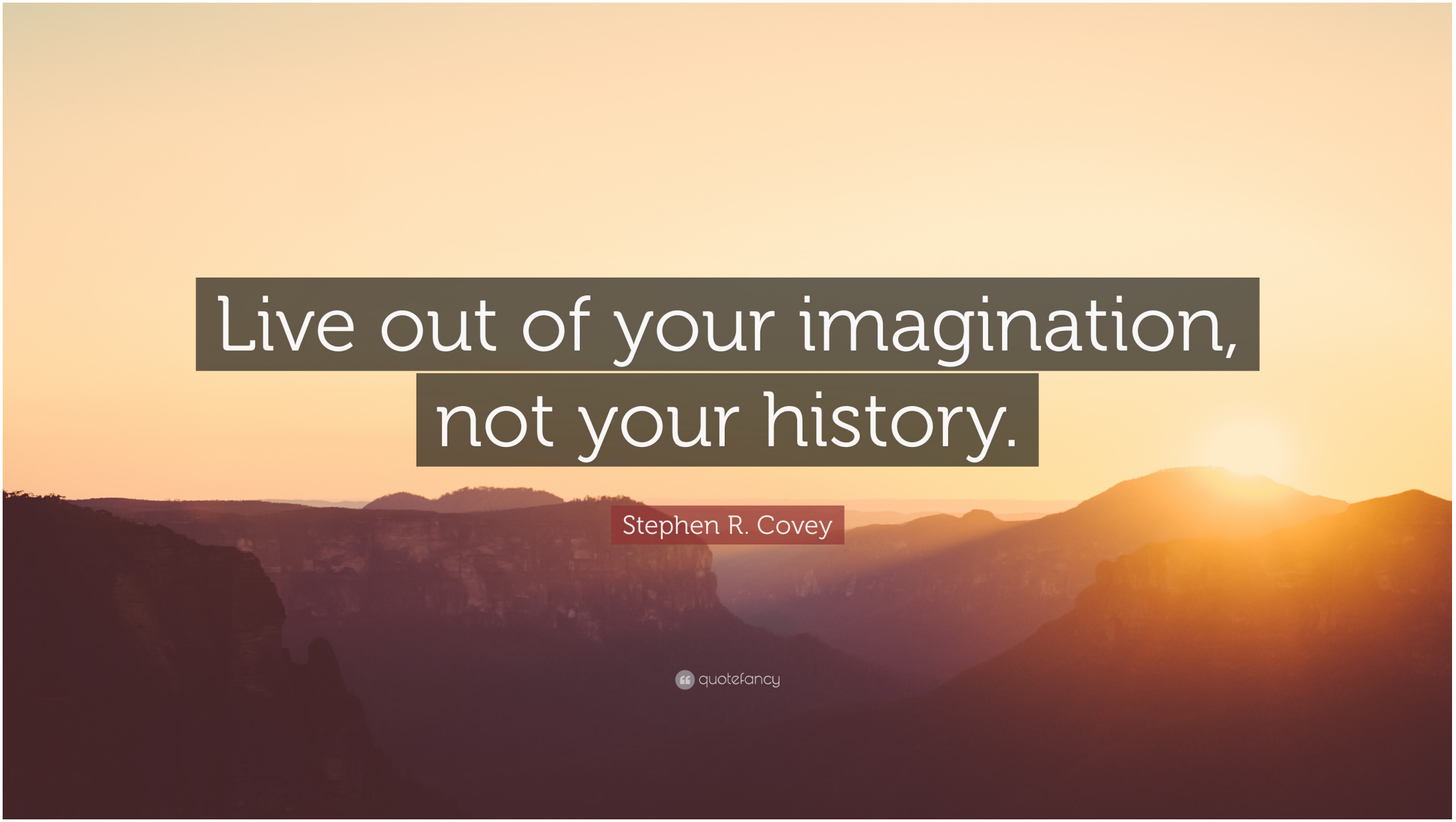
Continuous Delivery
ArgoCD - GitOps

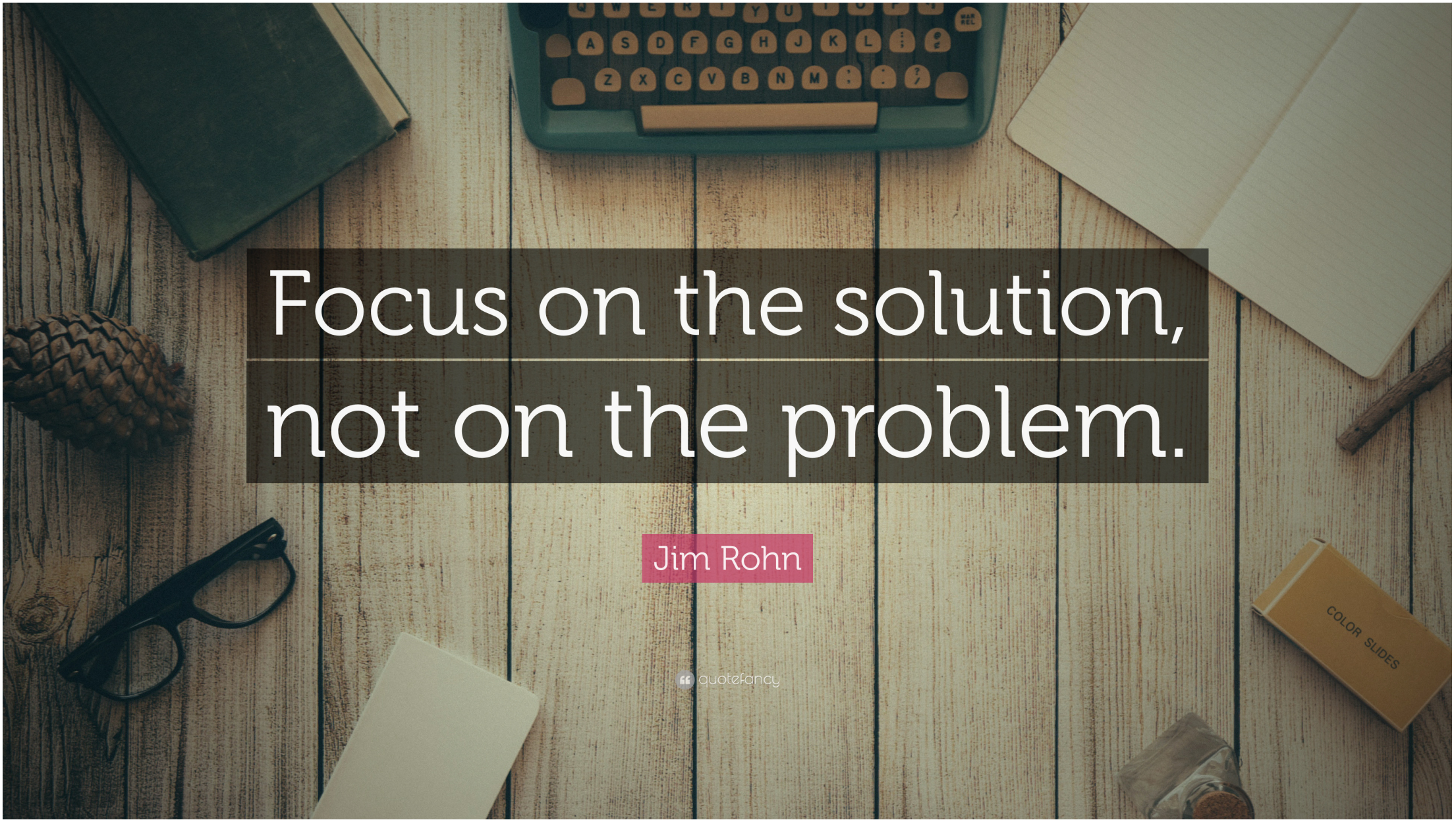
Orchestration Layer
Airflow

Logging
FileBeat, ElasticSearch & Kibana

Monitoring
Prometheus, Alert Manager & Grafana

Kubernetes
Open-Source Container-Orchestration System


Live out of your imagination,
not your history.
Stephen R. Covey
quotefancy


Focus on the solution,
not on the problem.
Jim Rohn
quotefancy
COLOR SLIDES

Data Engineer [Certifications]

Data Engineer Career ~ Part 1

Microsoft Certified: Azure Data Engineer Associate

design and implement the management, monitoring, security, and privacy of data using the full stack of azure data services to satisfy business needs.

Google Professional Data Engineer

enables data-driven decision making by collecting, transforming, and publishing data.

AWS Certified Big Data - Specialty

certification is intended for individuals who perform complex big data analyses with at least two years of experience using aws technology.

Confluent Certified Developer for Apache Kafka (CCDAK)

- Application Design ~ 40%
- Development ~ 30%
- Deployment, Testing & Monitoring ~ 30%

Confluent Certified Operator for Apache Kafka (CCOAK)

- Kafka Fundamentals ~ 15%
- Managing, Configuring & Optimizing ~ 30%
- Kafka Security ~ 15%
- Designing, Troubleshooting & Integrating Systems ~ 40%

Data Engineer [Study]

Data Engineer Career ~ Part 2

Apache Kafka Series ~ Apache Kafka for Beginners

- Introduction
- Theory
- Java Programming 101
- Producer & Consumer

Apache Kafka Series ~ Kafka Connect Hands-On

- Connect Concepts
- Connect Cluster
- Troubleshooting
- Source & Sink

Apache Kafka Series ~ Kafka Streams for Data Processing

- Kafka Streams App
- KStreams & KTables
- Exactly Once Semantics
- Joins using KStream & GlobalKTable
- Testing Kafka Streams App

Apache Kafka Series ~ KSQL for Stream Processing

- Setup
- Basics
- Intermediate
- Extensions
- Production

Luan Moreno M. Maciel

YouTube
luanmorenommaciel

LinkedIN
Luan Moreno Medeiros Maciel

Facebook
Luan Moreno Medeiros Maciel

Instagram
engenhariadedados

Meetup
BSB-AI-Big-Data-Analytics

Thank You

One Way Solution


ONEWAY
SOLUTION