



Apache Kafka + Apache Mesos

Highly Scalable Streaming Microservices with Kafka Streams

Kai Waehner

Technology Evangelist
kontakt@kai-waehner.de
LinkedIn
@KaiWaehner
www.kai-waehner.de



Abstract

Microservices establish many benefits like agile, flexible development and deployment of business logic. However, a Microservice architecture also creates many new challenges like increased communication between distributed instances, the need for orchestration, new fail-over requirements, and resiliency design patterns.

This session discusses how to build a highly scalable, performant, mission-critical microservice infrastructure with Apache Kafka and Apache Mesos. Apache Kafka brokers are used as powerful, scalable, distributed message backbone. Kafka's Streams API allows to embed stream processing directly into any external microservice or business application; without the need for a dedicated streaming cluster. Apache Mesos can be used as scalable infrastructure for both, the Apache Kafka brokers and external applications using the Kafka Streams API, to leverage the benefits of a cloud native platforms like service discovery, health checks, or fail-over management.

A live demo shows how to develop real time applications for your core business with Kafka messaging brokers and Kafka Streams API and how to deploy / manage / scale them on a Mesos cluster using different deployment options.

Key takeaways for the audience

- Successful Microservice architectures require a highly scalable messaging infrastructure combined with a cloud-native platform which manages distributed microservices
- Apache Kafka offers a highly scalable, mission critical infrastructure for distributed messaging and integration
- Kafka's Streams API allows to embed stream processing into any external application or microservice
- Mesos allows management of both, Kafka brokers and external applications using Kafka Streams API, to leverage many built-in benefits like health checks, service discovery or fail-over control of microservices
- See a live demo which combines the Apache Kafka streaming platform and Apache Mesos

- 1) Scalable Microservices
- 2) Apache Kafka and Confluent Platform
- 3) Kafka Streams
- 4) Scalable Microservices with Kafka and DC/OS
- 5) Live Demo

1) Scalable Microservices

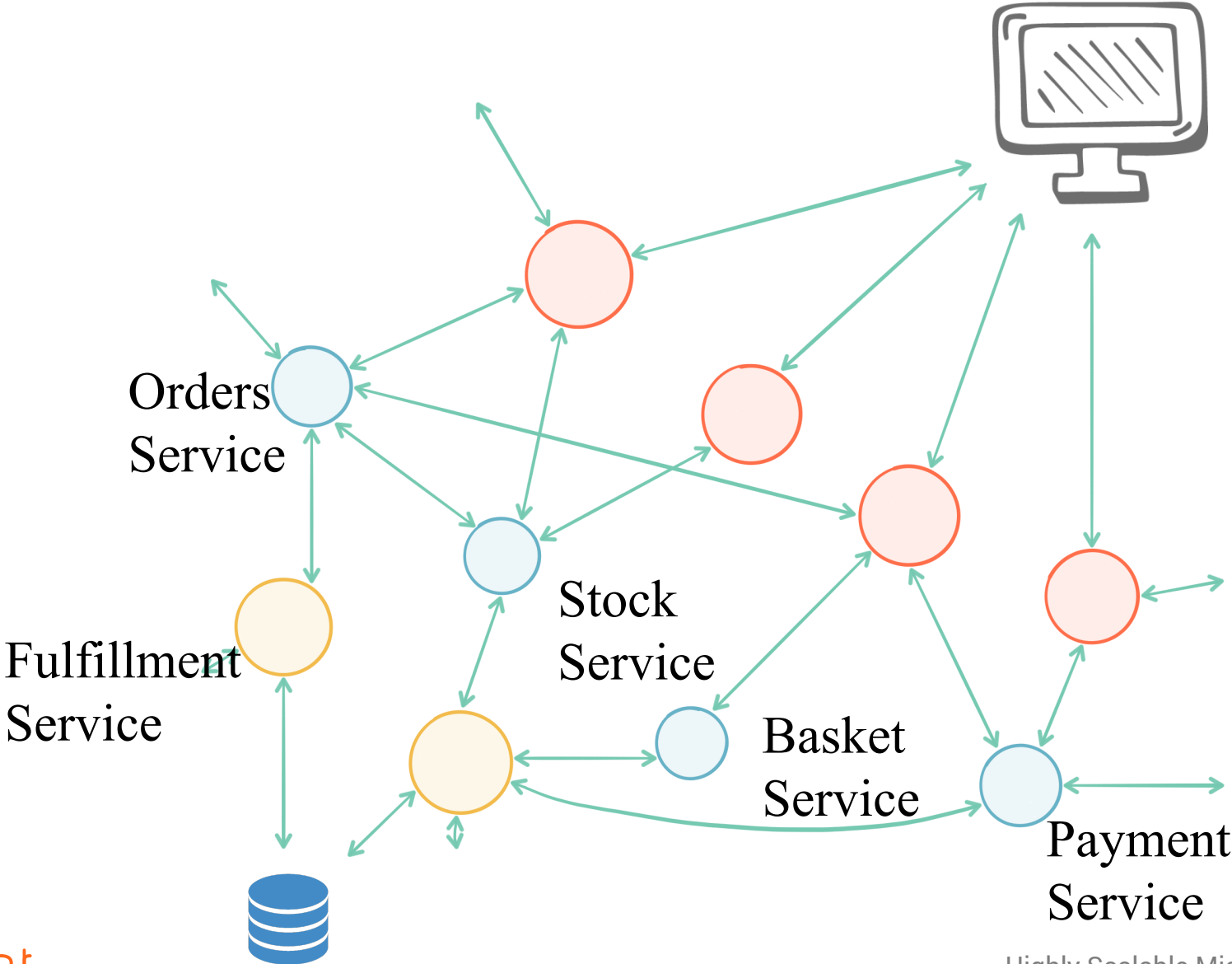
2) Apache Kafka and Confluent Platform

3) Kafka Streams

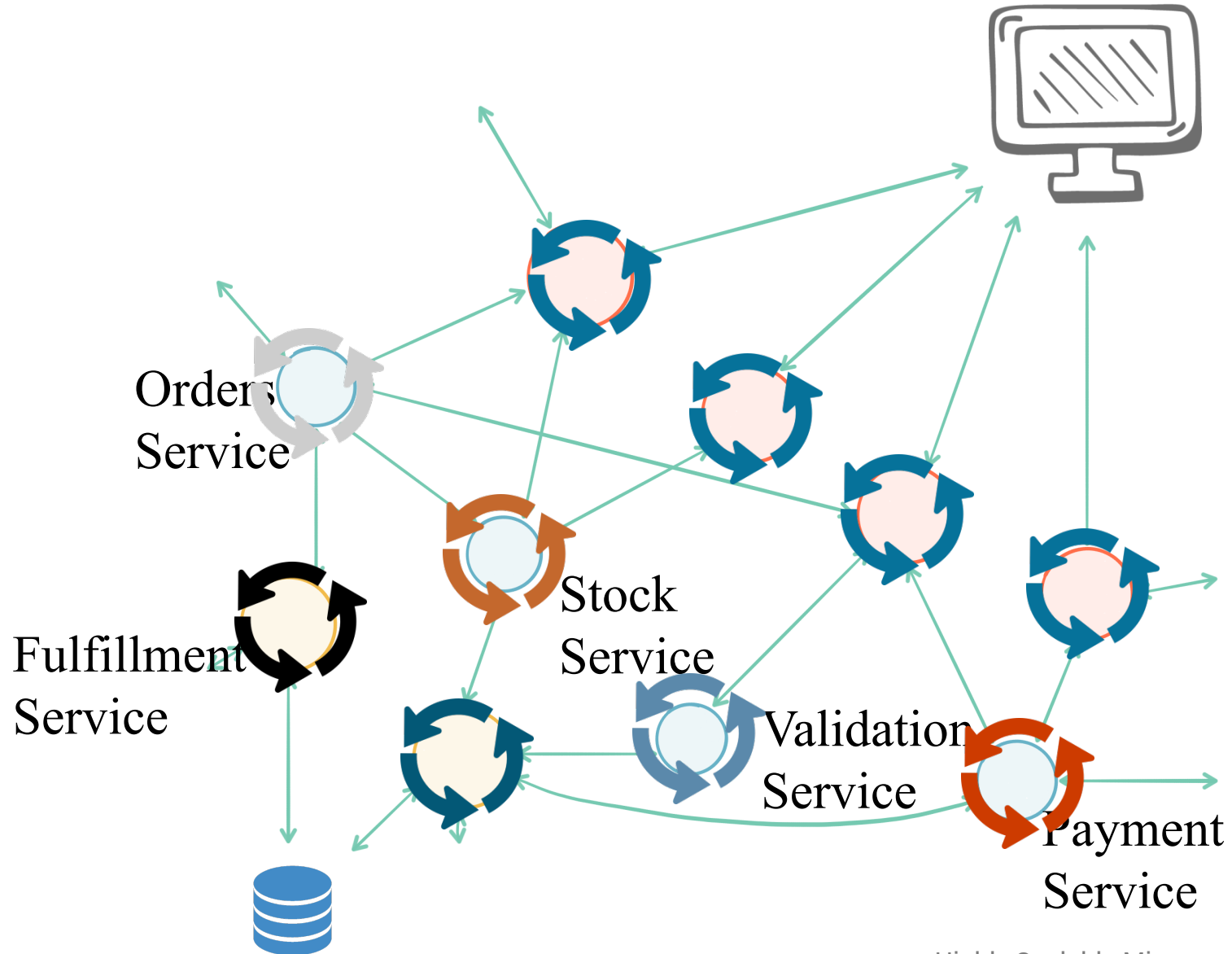
4) Scalable Microservices with Kafka and DC/OS

5) Live Demo

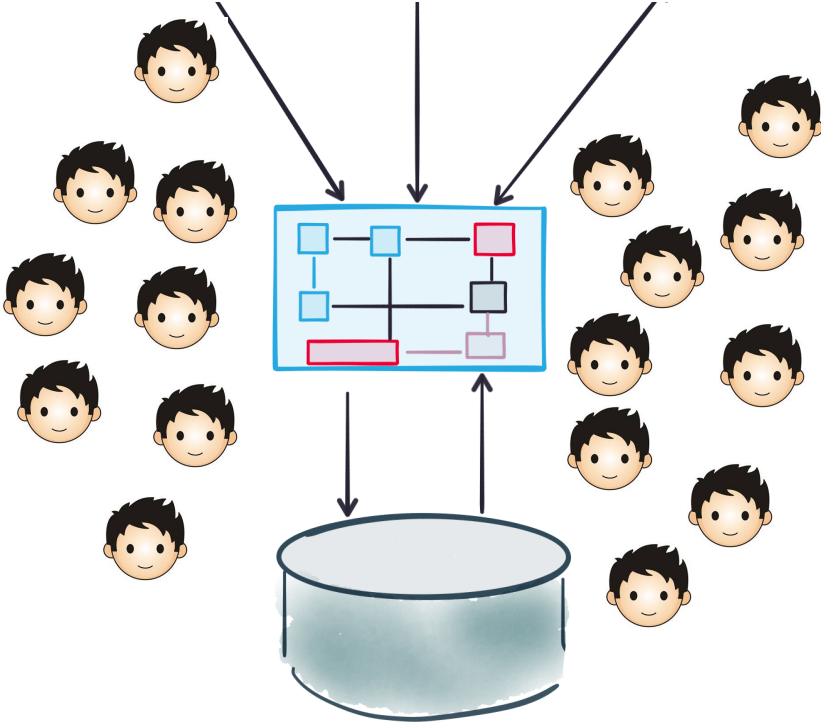
Microservices Architecture



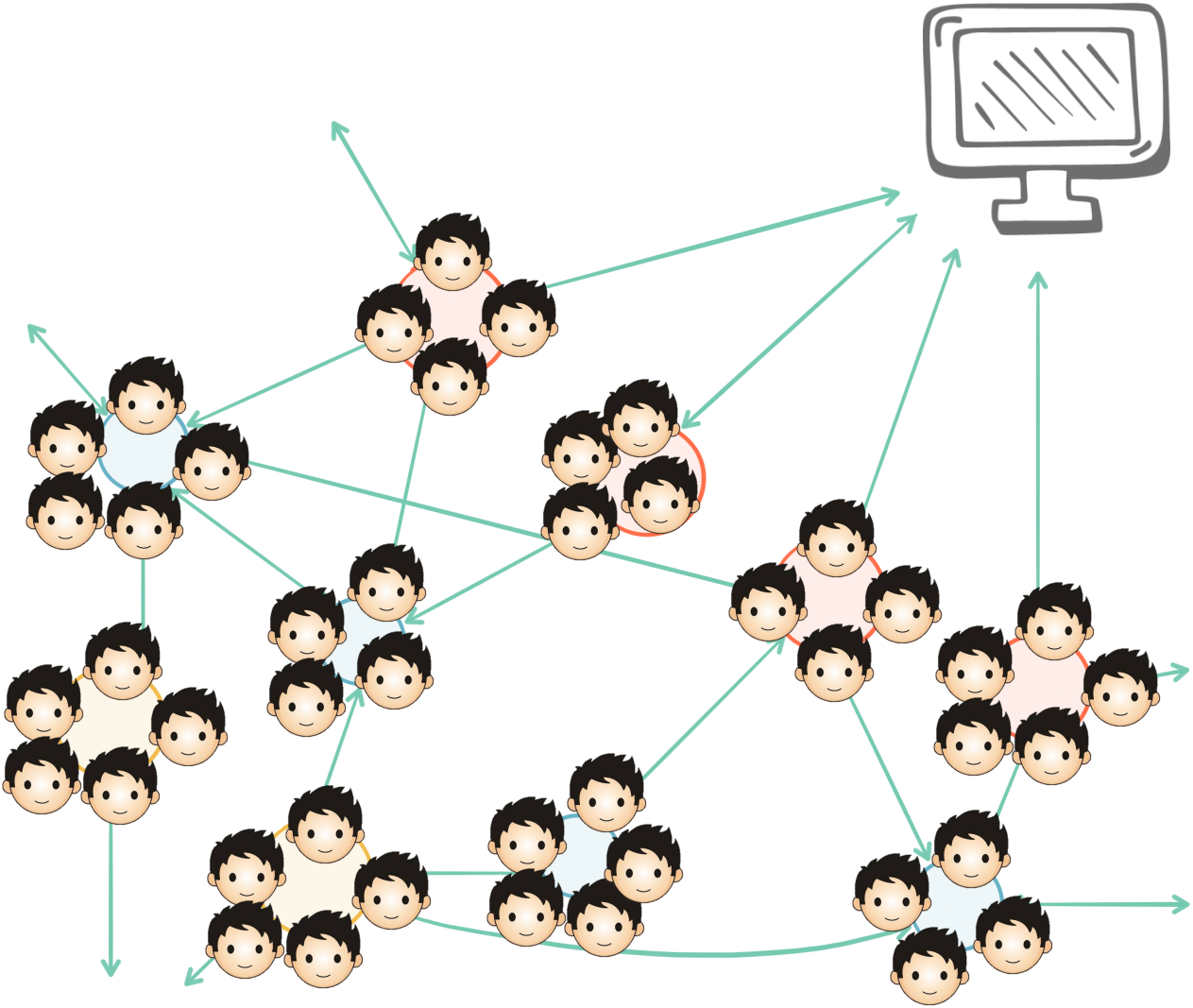
Independently Deployable



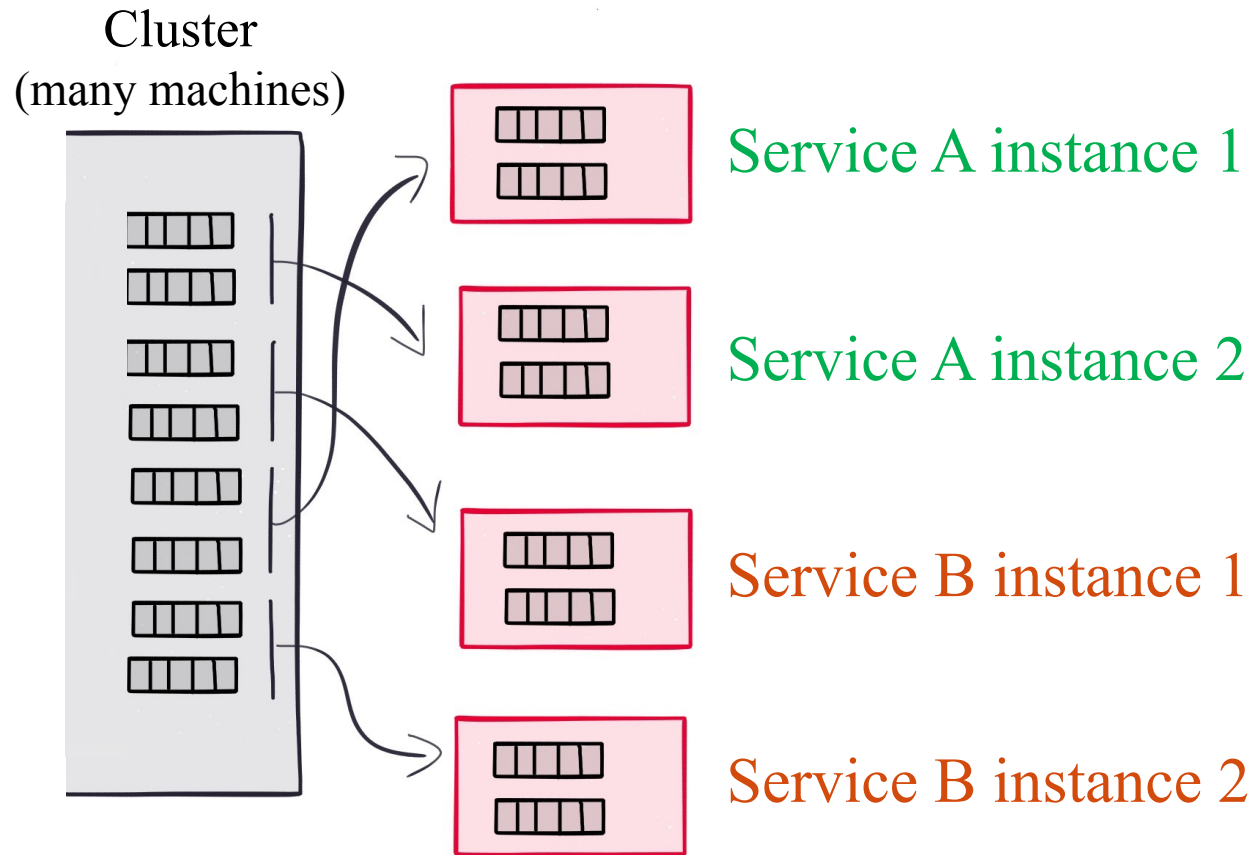
Allows us to scale



Scale in people terms



Scale in infrastructure terms



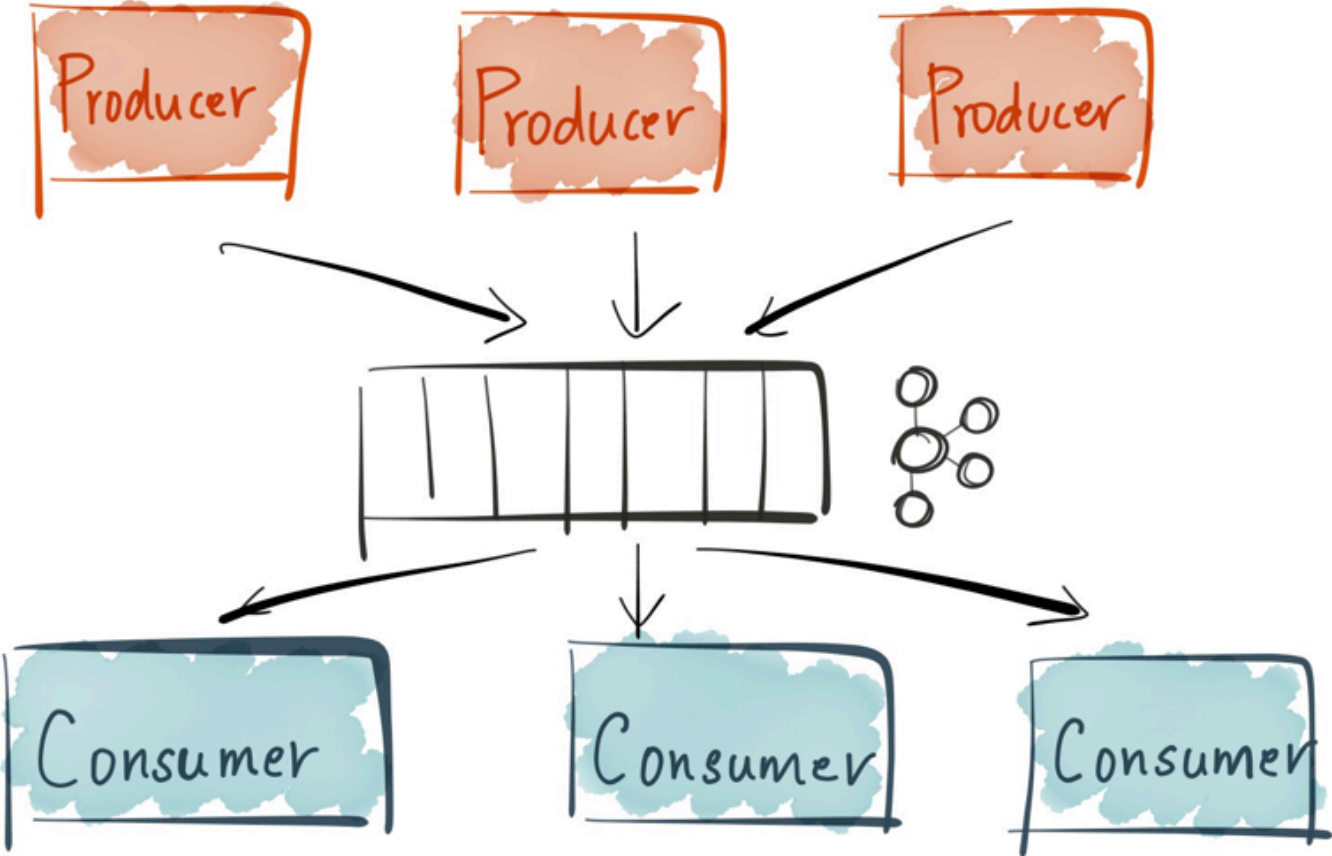


How do we get there?

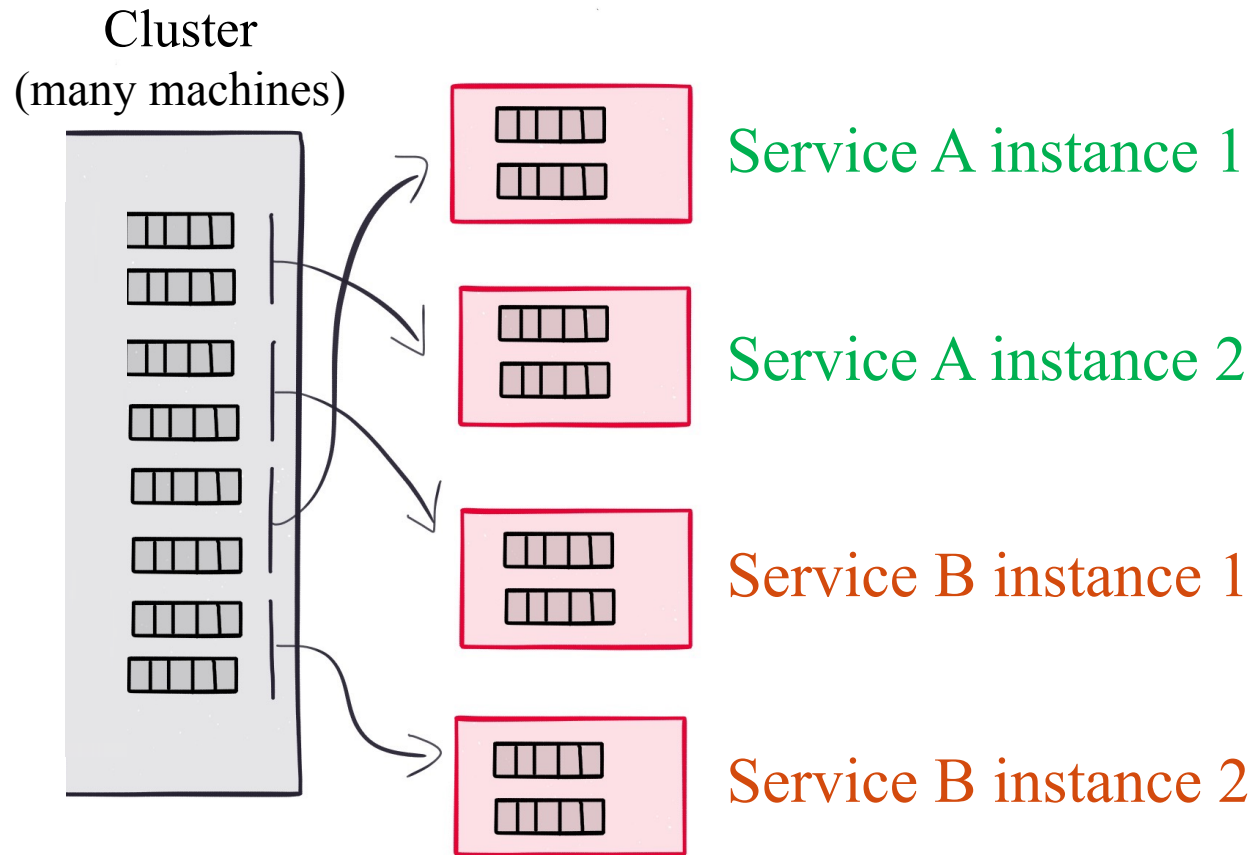
- Loose Coupling
- Data Enabled
- Event Driven
- Operational Transparency

- 1) Scalable Microservices
- 2) Apache Kafka and Confluent Platform**
- 3) Kafka Streams
- 4) Scalable Microservices with Kafka and DC/OS
- 5) Live Demo

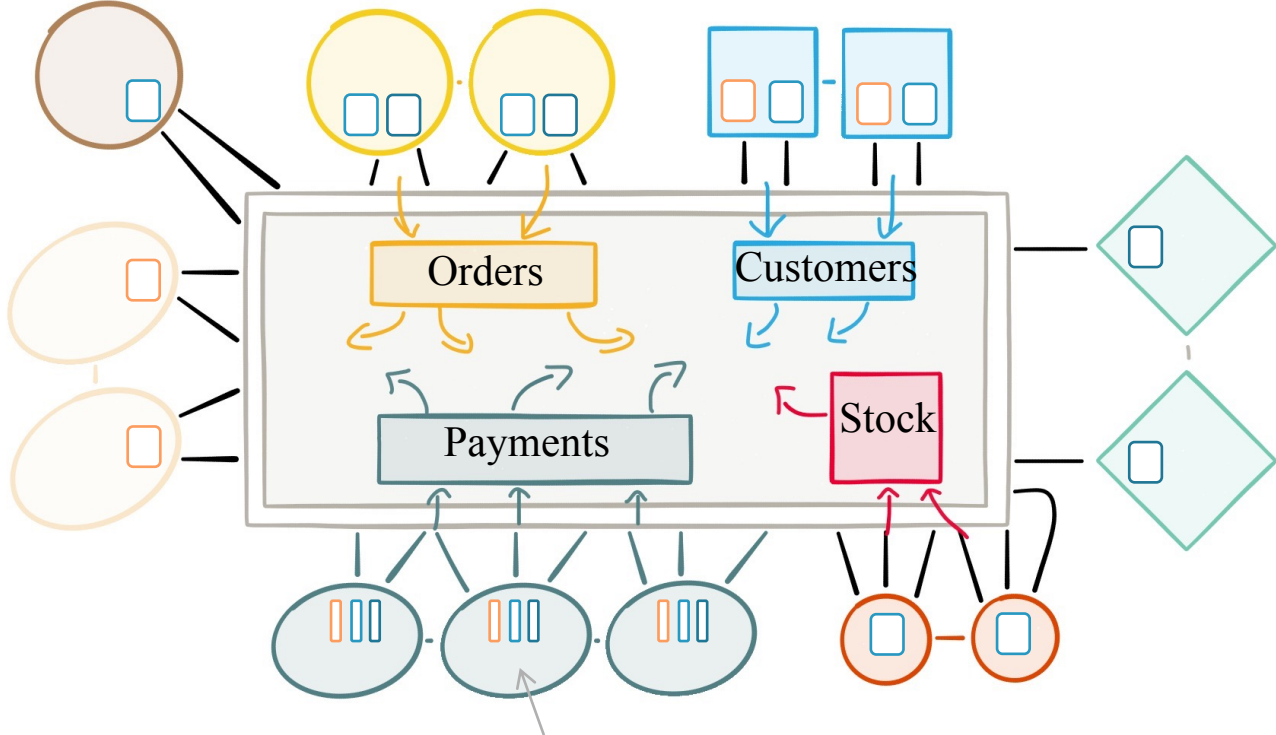
Apache Kafka – A Distributed, Fault-Tolerant, Scalable Commit Log



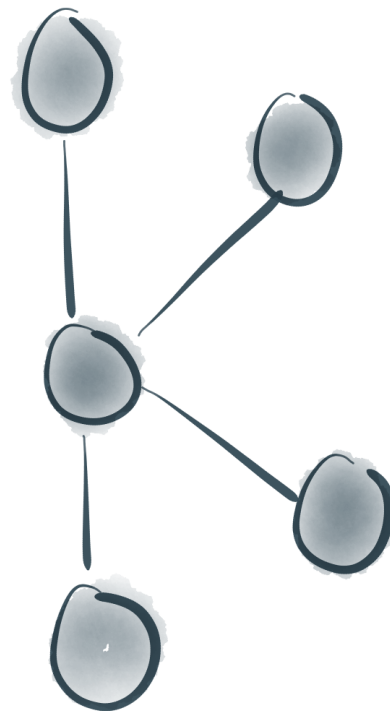
Scale in infrastructure terms



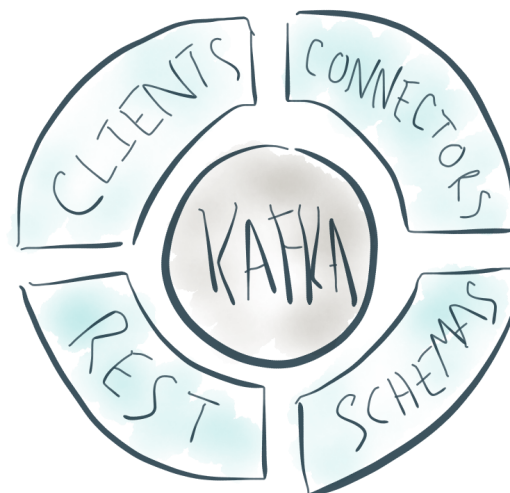
Single, Shared Source of Truth



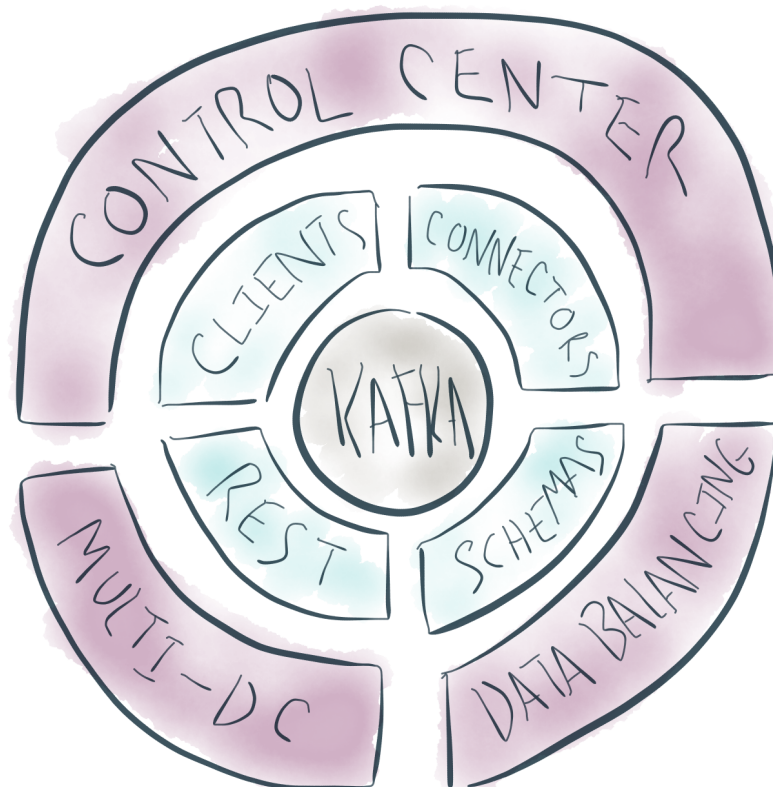
A MAJOR NEW ECOSYSTEM



CONFLUENT OPEN SOURCE



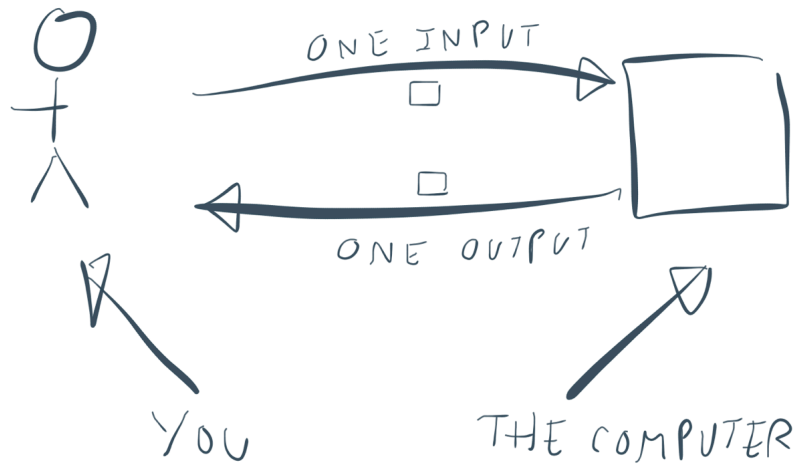
CONFLUENT ENTERPRISE



- 1) Scalable Microservices
- 2) Apache Kafka and Confluent Platform
- 3) Kafka Streams**
- 4) Scalable Microservices with Kafka and DC/OS
- 5) Live Demo

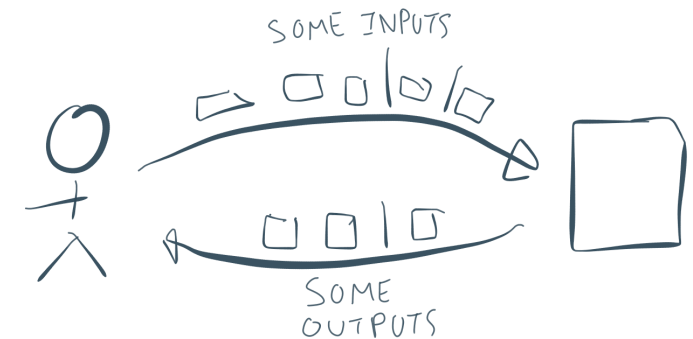
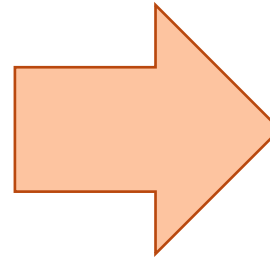
Definition of Stream Processing

REQUEST/RESPONSE



Data at Rest

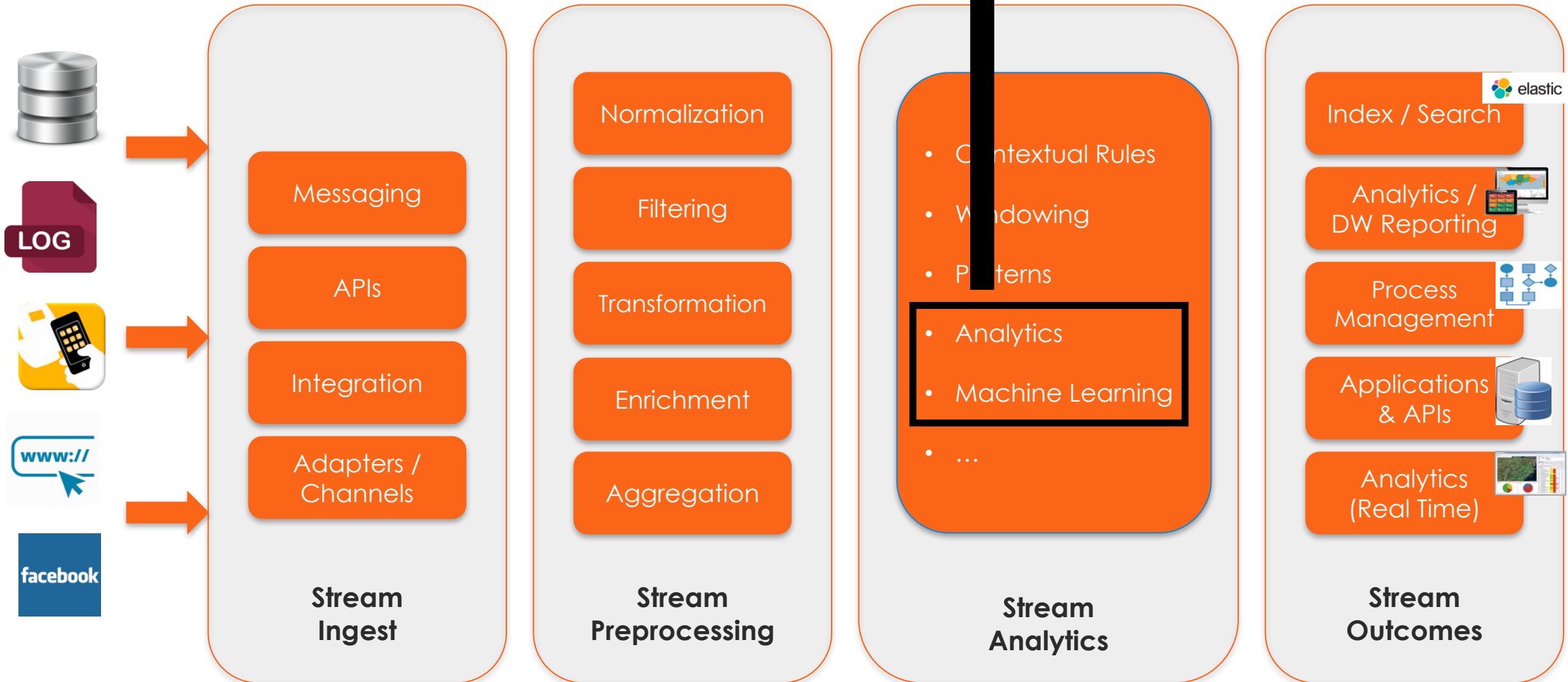
STREAM PROCESSING



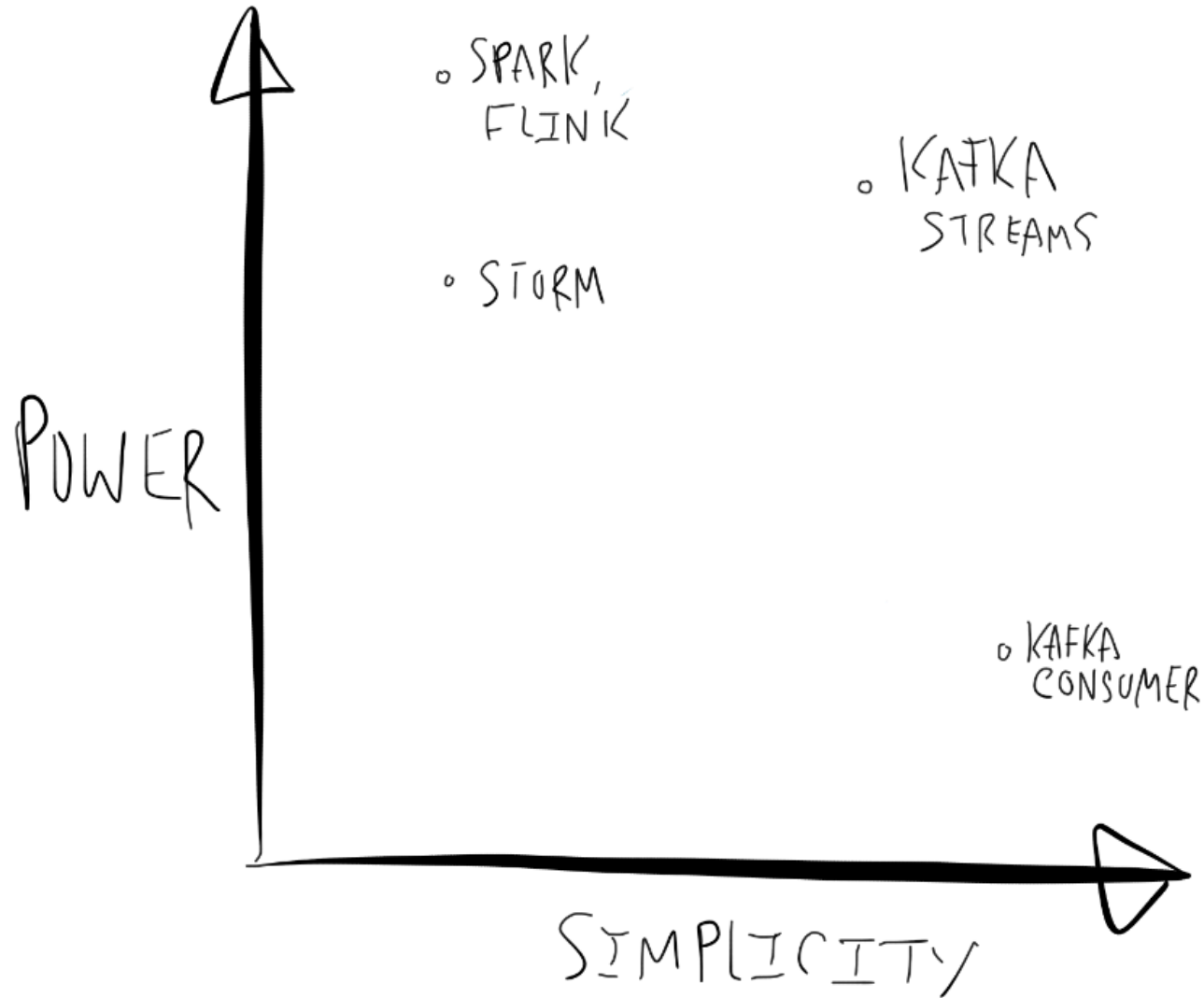
Data in Motion

Stream Processing Pipeline

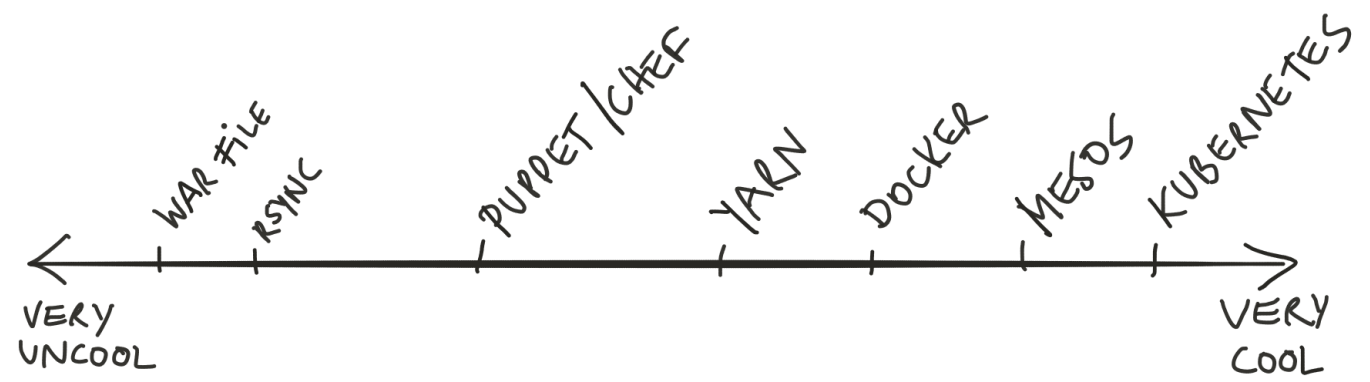
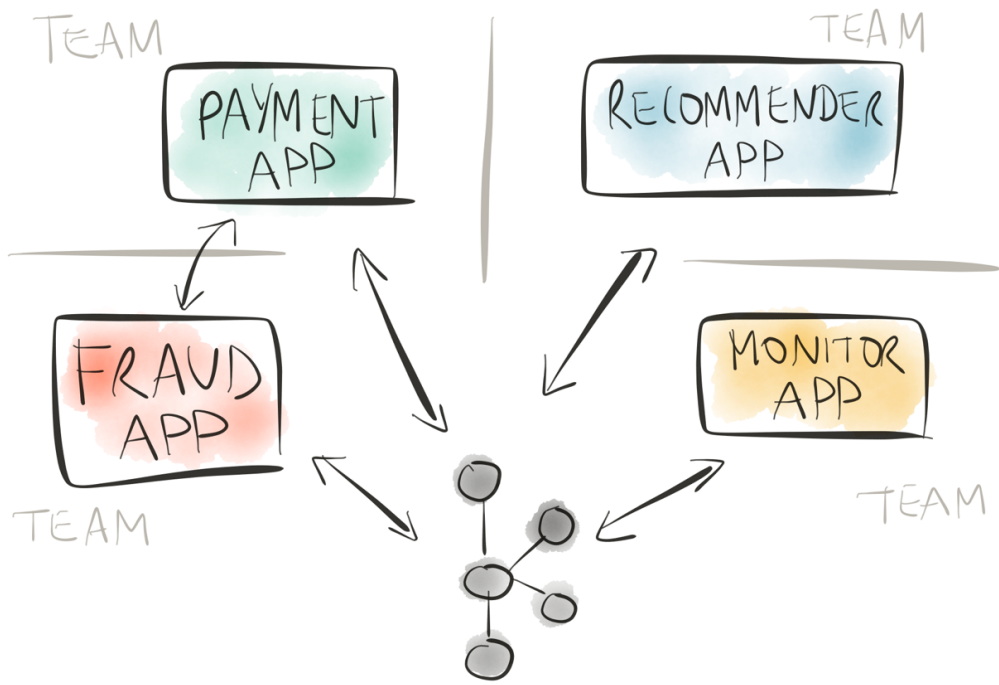
Applying an Analytic Model is just a piece of the puzzle!



When to use Kafka Streams for Stream Processing?



When to use Kafka Streams for Stream Processing?



KAFKA STREAMS

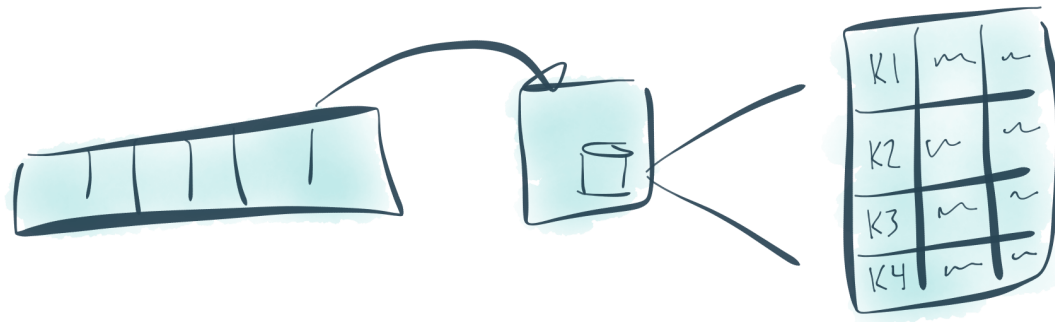
- SIMPLE LIBRARY
- CONVENIENT DSL
- DATAFLOW STYLE WINDOWING
- REPROCESSING
- NO MICROBATCH
- LOCAL STATE

KEY OPERATIONS

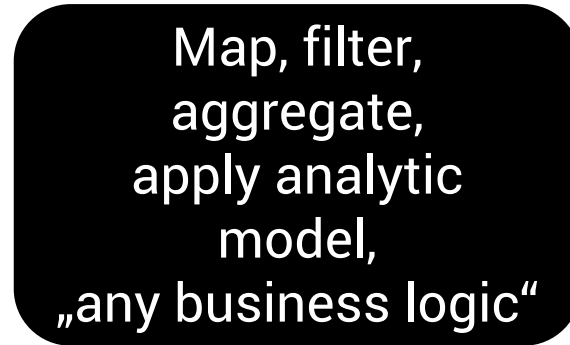
- MAP
- FILTER
- AGGREGATE (COUNT, SUM, ETC)
- JOIN

Like Streams library or scala collections or reactive thingies BUT stateful, fault-tolerant, distributed

UNIFY TABLES & STREAMS



Kafka Streams (shipped with Apache Kafka)



Input Stream
(Kafka Topic)

**Stream Processing
Microservice**
(Kafka Streams)

Output Stream
(Kafka Topic)

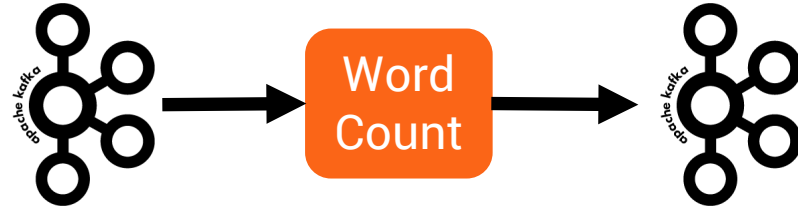
Kafka Cluster

Deployed Anywhere

Java App, Docker,
Kubernetes, Mesos,
“you-name-it”

Kafka Cluster

A complete streaming microservice, ready for production at large-scale



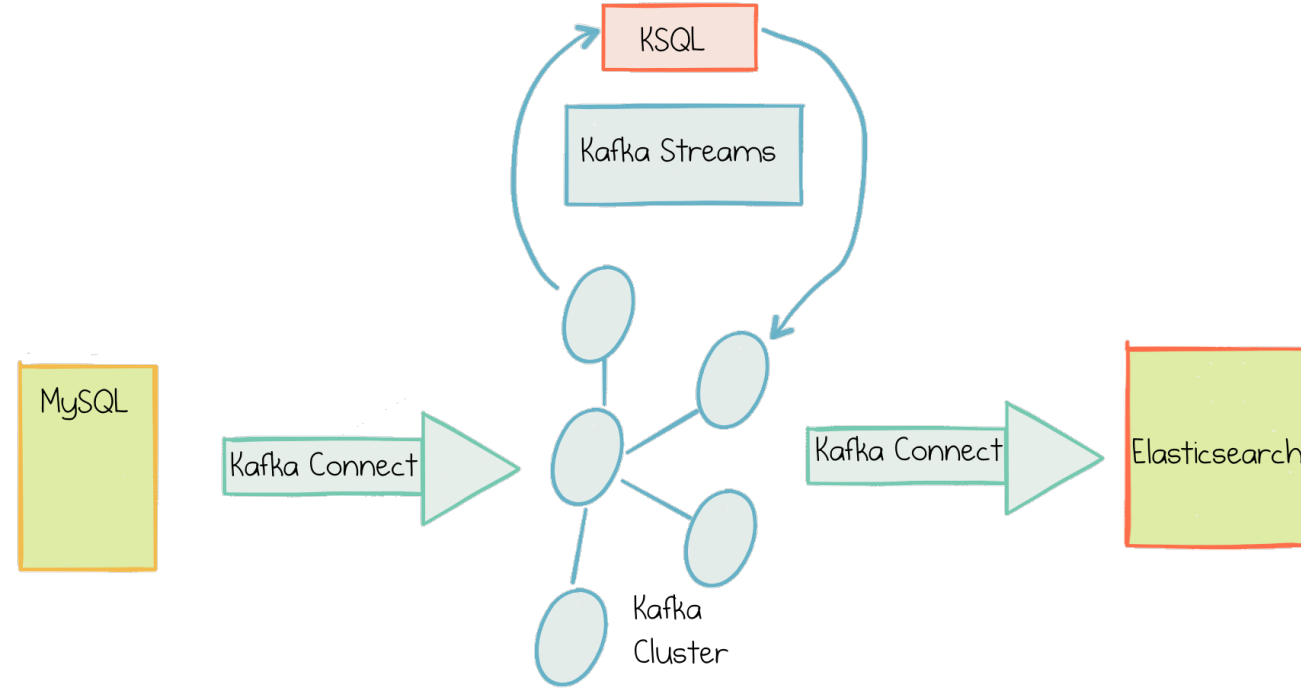
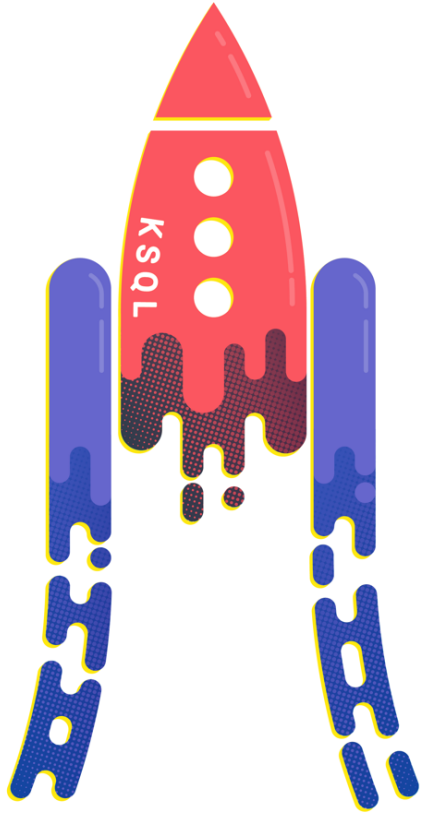
```
1 public static void main(final String[] args) throws Exception {
2     Properties config = new Properties();
3     config.put(StreamsConfig.APPLICATION_ID_CONFIG, "wordcount-example");
4     config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker1:9092");
5     config.put(StreamsConfig.KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName());
6     config.put(StreamsConfig.VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName());
7
8     KStreamBuilder builder = new KStreamBuilder();
9     KStream<String, String> textLines = builder.stream("TextLinesTopic");
10    KStream<String, Long> wordCounts = textLines
11        .flatMapValues(value -> Arrays.asList(value.toLowerCase().split("\\W+")))
12        .groupBy((key, word) -> word)
13        .count("Counts")
14        .toStream();
15    wordCounts.to(Serdes.String(), Serdes.Long(), "WordsWithCountsTopic");
16
17    KafkaStreams streams = new KafkaStreams(builder, config);
18    streams.start();
19 }
```

App configuration

Define processing (here: WordCount)

Start processing

KSQL – A Streaming SQL Engine for Apache Kafka

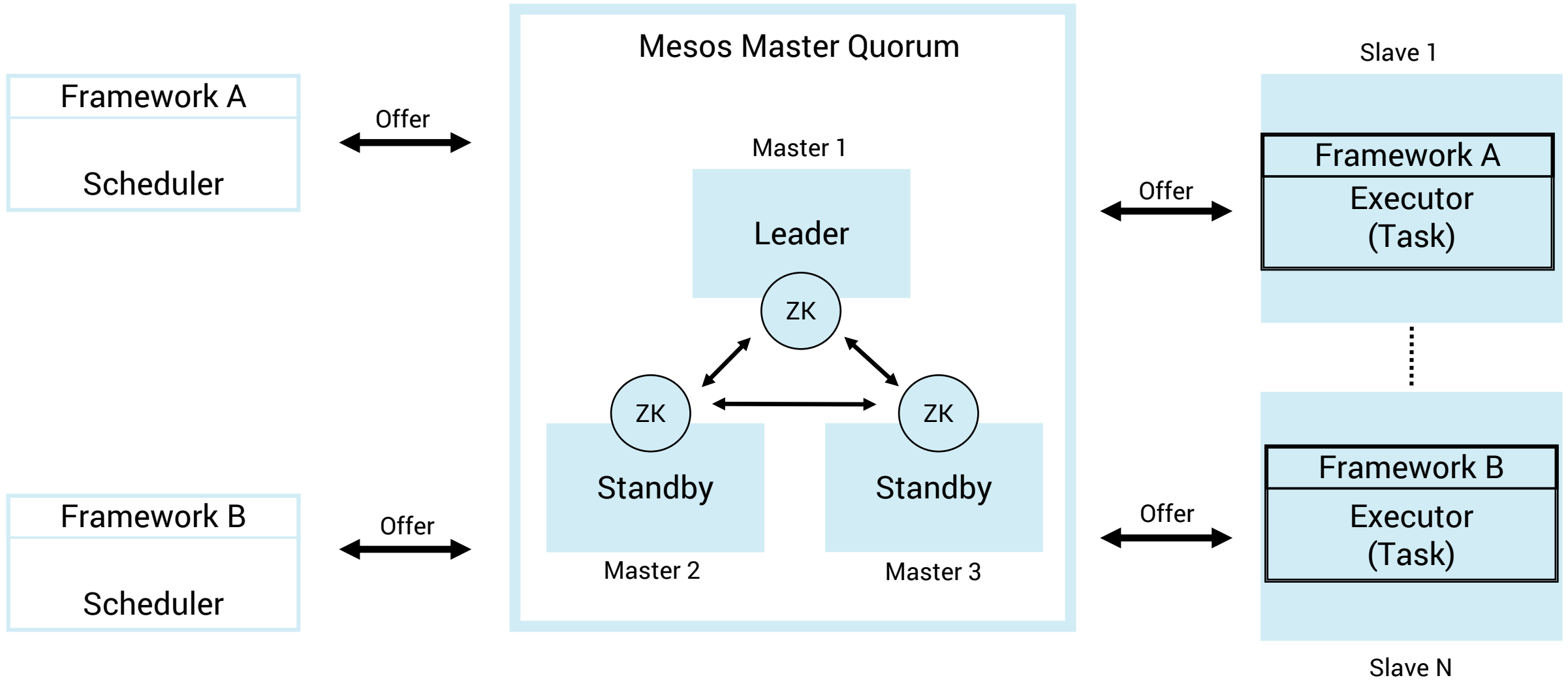


```
SELECT STREAM
  CEIL(timestamp TO HOUR) AS timeWindow, productId, COUNT(*) AS hourlyOrders, SUM(units) AS units
FROM Orders GROUP BY CEIL(timestamp TO HOUR), productId;
```

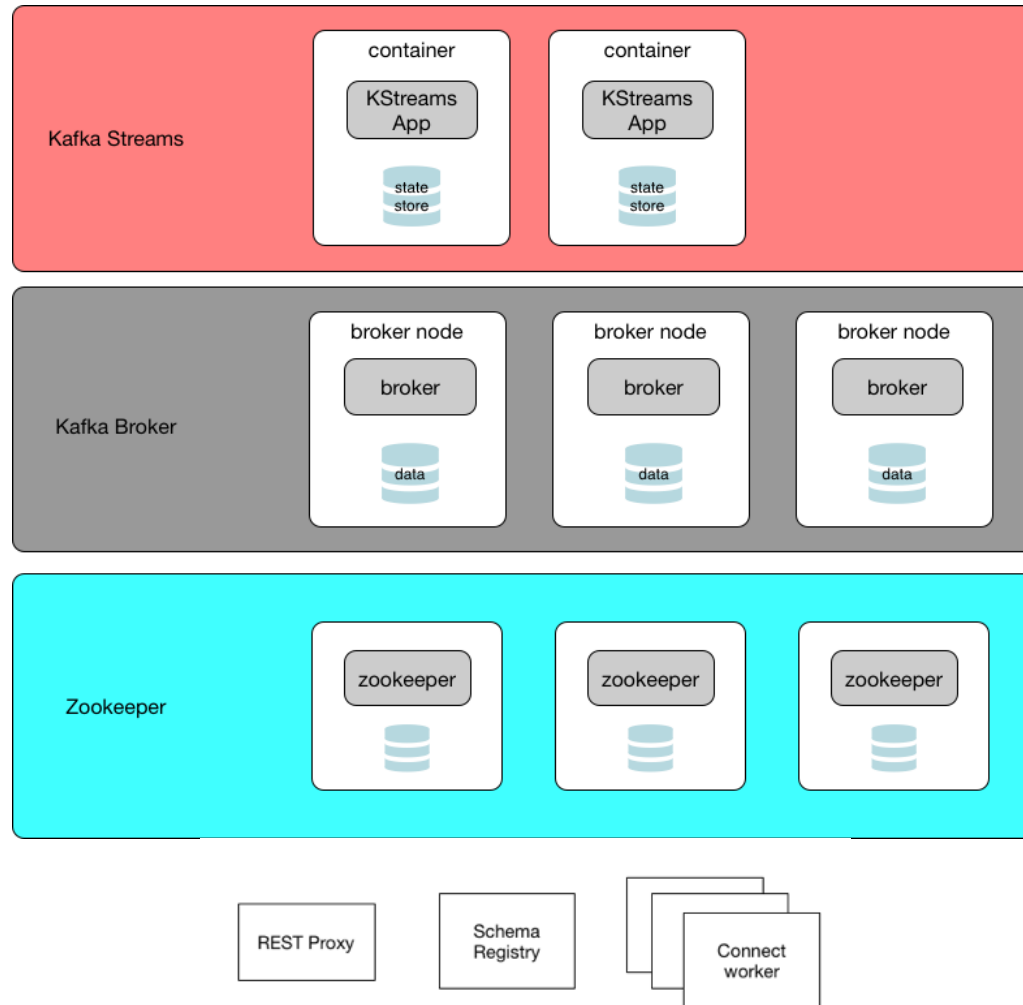
timeWindow	productId	hourlyOrders	units
08:00:00	10	2	5
08:00:00	20	1	8
09:00:00	10	4	22
09:00:00	40	1	45
...

- 1) Scalable Microservices
- 2) Apache Kafka and Confluent Platform
- 3) Kafka Streams
- 4) Scalable Microservices with Kafka and DC/OS**
- 5) Live Demo

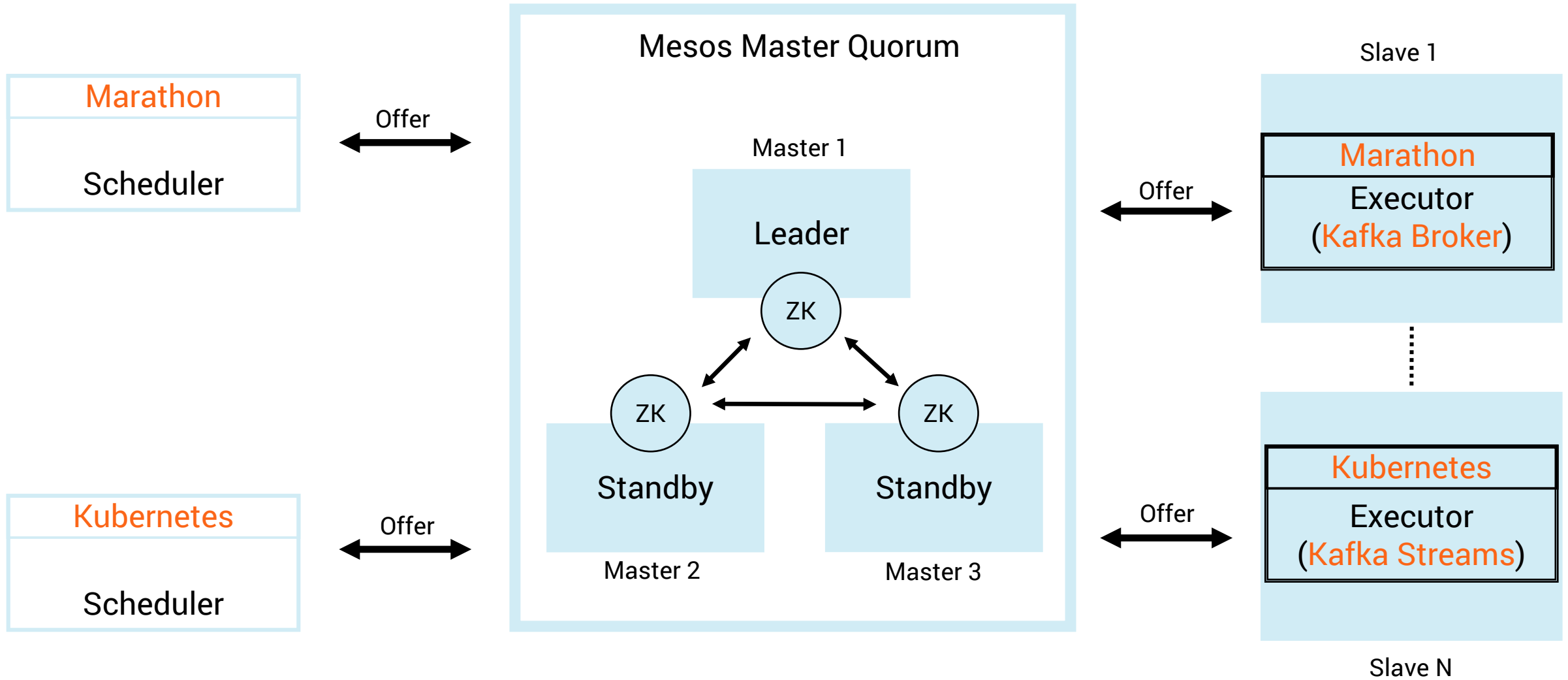
Mesos Architecture



Components of a Kafka Cluster



Mesos Architecture



Why DC/OS for the Kafka Ecosystem?



- **Automated provisioning and upgrading of Kafka components**
 - Broker, REST Proxy, Schema Registry, Connect ...
 - Kafka applications (Java / Go / .NET / Python Clients, Kafka Streams, KSQL)
 - Monitoring (Confluent Control Center, etc.)
- **Unified management and monitoring**
 - Easy interactive installation
 - Multiple Kafka Cluster on one infrastructure + multi-tenancy
 - Combination with other Big Data components (Spark, Cassandra, etc.) on one infrastructure
 - Integration with syslog-compatible logging services for diagnostics and troubleshooting
- **Elastic scaling, fault tolerance and self-healing**
 - Stateful and stateless services
 - Service discovery and routing (using the corresponding Mesos framework, i.e. Marathon or Kubernetes)
 - Kafka VIP Connection (one “static” bootstrap server url)
 - Storage volumes for enhanced data durability, known as Mesos Dynamic Reservations and Persistent Volumes

- 1) Scalable Microservices
- 2) Apache Kafka and Confluent Platform
- 3) Kafka Streams
- 4) Scalable Microservices with Kafka and DC/OS
- 5) Live Demo**

Use Case:

Airline Flight Delay Prediction

Machine Learning Algorithm:

Gradient Boosting (GBM)
using Decision Trees

Technologies:

DC/OS

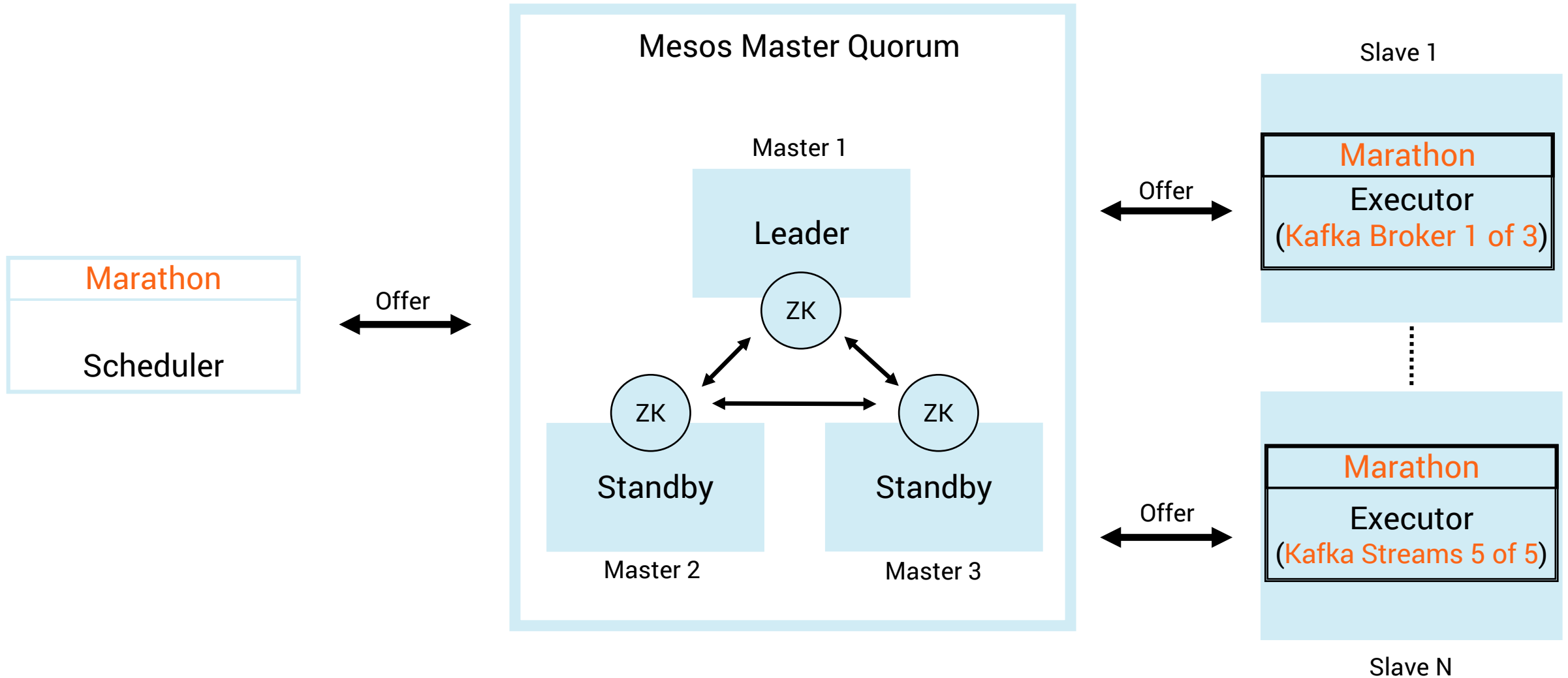
Kafka Broker

Kafka Streams

H2O.ai



Architecture – Live Demo



DC/OS on AWS



Secure | <https://downloads.dcos.io/dcos/stable/aws.html>

DC/OS stable AWS CloudFormation

- For more information on creating a DC/OS cluster, see the [DC/OS AWS documentation](#).

Region Name	Region Code	Single Master	HA: Three Master
US West (N. California)	us-west-1	Launch Stack	Launch Stack
US West (Oregon)	us-west-2	Launch Stack	Launch Stack
US East (N. Virginia)	us-east-1	Launch Stack	Launch Stack
South America (Sao Paulo)	sa-east-1	Launch Stack	Launch Stack
EU (Ireland)	eu-west-1	Launch Stack	Launch Stack
EU (Frankfurt)	eu-central-1	Launch Stack	Launch Stack
Asia Pacific (Tokyo)	ap-northeast-1	Launch Stack	Launch Stack
Asia Pacific (Singapore)	ap-southeast-1	Launch Stack	Launch Stack
Asia Pacific (Sydney)	ap-southeast-2	Launch Stack	Launch Stack

CloudFormation - Stacks - Create Stack

Create stack

Review

Template

Template URL: <https://s3-us-west-2.amazonaws.com/downloads.dcos.io/dcos/stable/commits/38ab2aa28077c8ab75f1d3c8ff7b0d06d1a461cloudformation/single-master-cloudformation.json>

Description: DC/OS AWS CloudFormation Template

Estimate cost: Cost

Details

Stack name: kai-kafka-mesos

AdminLocation: 0.0.0.0/0

KeyName: UserKeyKafkaTutorial

QueueEnabled: true

PublicSlaveInstanceCount: 1

SlaveInstanceCount: 5

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

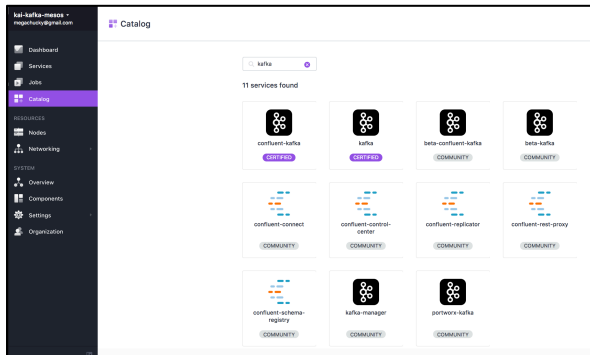
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
	i-013bdb38400aecb9e	m3.xlarge	eu-central-1a	running	2/2 checks ...
	i-048abc70961b7d4b6	m3.xlarge	eu-central-1a	running	2/2 checks ...
	i-09d6186b86154dcd1	m3.xlarge	eu-central-1a	running	2/2 checks ...
	i-0a9a98db0e89b01...	m3.xlarge	eu-central-1a	running	2/2 checks ...
	i-0bda7b21bf5830778	m3.xlarge	eu-central-1a	running	2/2 checks ...
	i-0be452e5dd96007...	m3.xlarge	eu-central-1a	running	2/2 checks ...
	i-0bf7d13a12f872c6	m3.medium	eu-central-1a	running	2/2 checks ...
	i-0f867ba1505ef307f	m3.xlarge	eu-central-1a	running	2/2 checks ...


Logical ID	Physical ID	Type	Status
AdminSecurityGroup	sg-4ddf0c27	AWS::EC2::SecurityGroup	CREATE_COMPLETE
DHCPOptions	dopt-3d785355	AWS::EC2::DHCPOptions	CREATE_COMPLETE
ElasticLoadBalancer	kai-kafka-ElasticL-1CC7BQF1NQU83	AWS::ElasticLoadBalancing::...	CREATE_COMPLETE
ExhibitorS3Bucket	kai-kafka-mesos-exhibitors3bucket-fv16h731h5i8	AWS::S3::Bucket	CREATE_COMPLETE
GatewayToInternet	kai-k-Gatew-N67EBATROY0G	AWS::EC2::VPCGatewayAtta...	CREATE_COMPLETE
InboundNetworkAclEntry	kai-k-Inbou-15NK4D5HF8YIL	AWS::EC2::NetworkAclEntry	CREATE_COMPLETE
InternalMasterLoadBal...	kai-kafka-Internal-1MH6OA6BQ9G5D	AWS::ElasticLoadBalancing::...	CREATE_COMPLETE
InternetGateway	igw-e2f4288a	AWS::EC2::InternetGateway	CREATE_COMPLETE
LbSecurityGroup	sg-4cdf0c26	AWS::EC2::SecurityGroup	CREATE_COMPLETE
MasterInstanceProfile	kai-kafka-mesos-MasterInstanceProfile-JOUX3T5WN R0K	AWS::IAM::InstanceProfile	CREATE_COMPLETE
MasterLaunchConfig	kai-kafka-mesos-MasterLaunchConfig-1MB0MEPZ4 DD70	AWS::AutoScaling::LaunchCo...	CREATE_COMPLETE
MasterRole	kai-kafka-mesos-MasterRole-1UCTGMIDFEWK	AWS::IAM::Role	CREATE_COMPLETE
MasterSecurityGroup	sg-4ede0d24	AWS::EC2::SecurityGroup	CREATE_COMPLETE
MasterServerGroup	kai-kafka-mesos-MasterServerGroup-4O40KL2JNEP L	AWS::AutoScaling::AutoScali...	CREATE_COMPLETE
MasterToMasterIngress	MasterToMasterIngress	AWS::EC2::SecurityGroupIngr...	CREATE_COMPLETE
MasterToPublicSlaveIn...	MasterToPublicSlaveIngress	AWS::EC2::SecurityGroupIngr...	CREATE_COMPLETE

Filter: Active By Stack Name Showing 1 stack

Stack Name	Created Time	Status	Description
kai-kafka-mesos	2017-10-16 08:07:49 UTC+0200	CREATE_COMPLETE	DC/OS AWS CloudFormation Template

Kafka Brokers on DC/OS





confluent-kafka

CERTIFIED 2.0.1-3.3.0e

By deploying you agree to the [terms and conditions](#)

[CONFIGURE](#) [DEPLOY](#)

Description
Apache Kafka by Confluent

Documentation: <https://www.confluent.io/whitepaper/deploying-confluent-platform-with-mesosphere>

Pre-Install Notes
This DC/OS Service is currently a beta candidate undergoing testing as part of a formal beta test program. There may be bugs, incomplete features, incorrect documentation, or other discrepancies. Contact Mesosphere and Confluent before deploying this beta candidate service. Product support is available to approved participants in the beta test program. Mutual Mesosphere and Confluent customers are eligible to participate in the beta program. Contact your rep at either company or partner-support@confluent.io.

Information
Maintainer: partner-support@confluent.io

Licenses
Apache License v2: <https://raw.githubusercontent.com/confluentinc/kafka/trunk/LICENSE>

Services > confluent-kafka Running (1)

Instances Configuration Debug

Showing 4 of 4 tasks [Clear](#)

Filter ALL 4 ACTIVE 4 COMPLETED 0

ID	NAME	HOST	STATUS	HEALTH	CPU	MEM
kafka-2-broker_6788e2bc-afc2-4e52-95d5-eca5d6d9b138	kafka-2-bro...	10.0.2.211	Running	●	1	4 GiB
kafka-1-broker_76a6c666-4e40-4798-87c4-993ace89f204	kafka-1-brok...	10.0.2.118	Running	●	1	4 GiB
kafka-0-broker_d52aac1a-e380-4cae-a481-9d036a002707	kafka-0-bro...	10.0.2.155	Running	●	1	4 GiB
confluent-kafka.612c0982-b23b-11e7-b6be-0280a10bfd8	confluent-ka...	10.0.2.155	Running	●	1	1 GiB

Exhibitor for ZooKeeper v1.8.4

Control Panel Explorer Config Log

Path: /dcos-service-confluent-kafka/brokers/topics/AirlineInputTopic

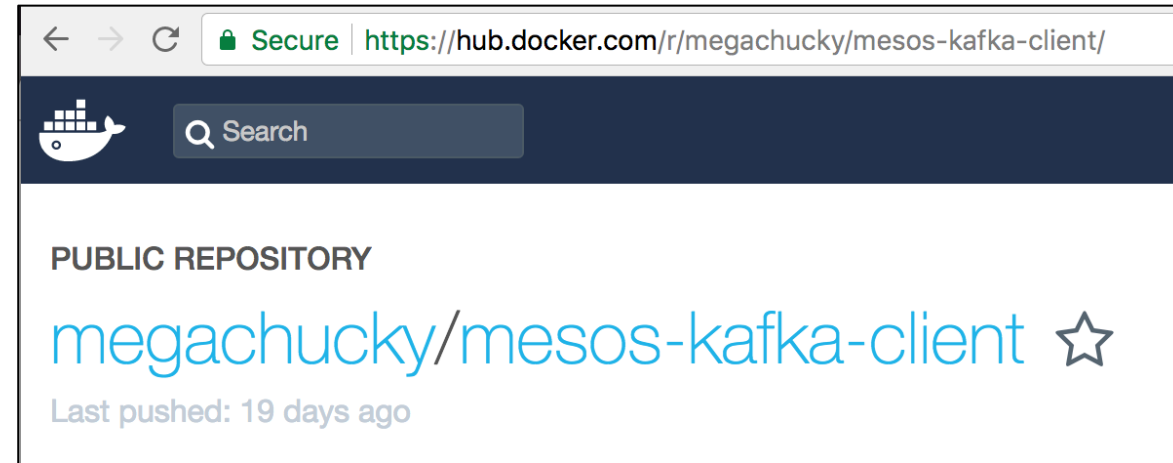
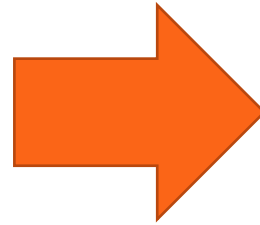
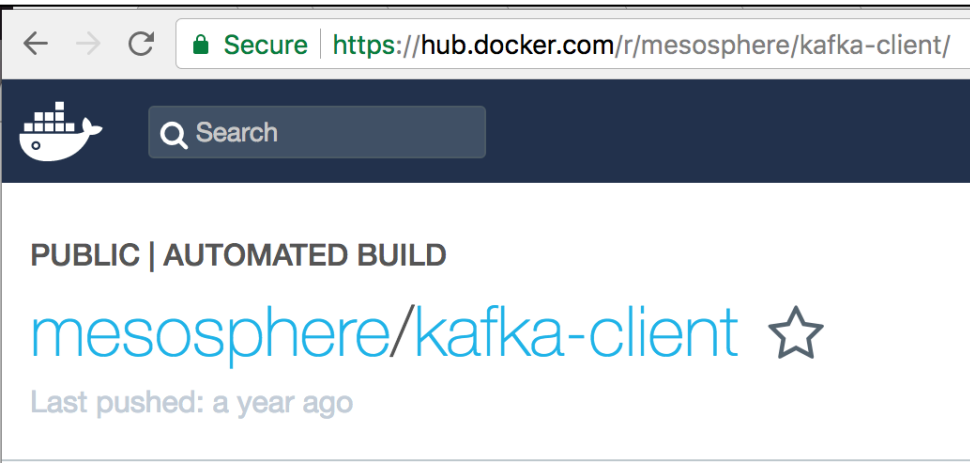
Stat: czxid: 441, mxzid: 441, ctime: 1508136708140, mtime: 1508136708140, version: 0, cversion: 1, aversion: 0, ephemeralOwner: 0, dataLength: 148, numChildren: 1, pxxid: 445

Data Bytes: 7b 22 76 65 72 73 69 6f 6e 22 3a 31 2c 22 70 61 72 74 69 69 6f 73 22 3a 7b 22 38 22 3a 5b 31 2c 32 2c 30 5d 2c 22 34 22 3a 5b 30 2c 32 2c 31 5d 2c 22 39 22 3a 5b 32 2c 31 2c 30 5d 2c 22 35 22 3a 5b 31 2c 30 2c 32 5d 2c 22 36 22 3a 5b 32 2c 30 2c 31 5d 2c 22 31 22 3a 5b 30 2c 31 2c 32 5d 2c 22 30 22 3a 5b 32 2c 30 2c 31 5d 2c 22 32 22 3a 5b 31 2c 32 2c 30 5d 2c 22 37 22 3a 5b 30 2c 31 2c 32 5d 2c 22 33 22 3a 5b 32 2c 31 2c 30 5d 7d 7d

Data as String ("version":1,"partitions":{"0":{"1,2,0},"4":{"0,2,1},"9":{"2,1,0},"5":{"1,0,2},"6":{"2,0,1},"1":{"0,1,2},"0":{"2,0,1},"2":{"1,2,0},"7":{"0,1,2},"3":{"2,1,0}})

```
~: dcos kafka --name confluent-kafka endpoints broker
{
  "address": [
    "10.0.2.155:1025",
    "10.0.2.118:1025",
    "10.0.2.211:1025"
  ],
  "dns": [
    "kafka-0-broker.confluent-kafka.autoip.dcos.thisdcos.directory:1025",
    "kafka-1-broker.confluent-kafka.autoip.dcos.thisdcos.directory:1025",
    "kafka-2-broker.confluent-kafka.autoip.dcos.thisdcos.directory:1025"
  ],
  "vip": "broker.confluent-kafka.l4lb.thisdcos.directory:9092"
}
```

Kafka Client (compatible with Kafka Streams) on DC/OS



From Apache Kafka 0.9 to 0.11 (Kafka Streams messages require timestamps)

```
dcos node ssh --master-proxy --leader  
docker run -it megachucky/mesos-kafka-client
```

Dockerfile:

```
FROM java:openjdk-8-jreMAINTAINER Kai Waehner  
curl http://apache.mirrors.spacedump.net/kafka/0.11.0.1/kafka_2.11-0.11.0.1.tgz | tar xvz --strip-components=1  
WORKDIR /bin
```

Kafka Streams Microservice



Public repository page for `megachucky/kafka-streams-machine-learning-docker-microservice`. The page shows a short description: "A Kafka Streams Microservice (using Machine Learning) deployed as a Docker Container". The Docker Pull Command is `docker pull megachucky/kafka-stream`. The owner is listed as megachucky.

GitHub repository page for `kaiwaeher/kafka-streams-machine-learning-examples`. The page shows a description: "This project contains examples which demonstrate how to deploy analytic models to mission-critical, scalable production environments leveraging Apache Kafka and its Streams API. Models are built with Python, H2O, TensorFlow, DeepLearning4 and other technologies." The repository has 45 commits, 1 branch, 0 releases, and 1 contributor. The latest commit is by Kai Waehner, adding documentation for DL4J example.

```
Dockerfile:
FROM java:8
ADD /opt/kafka-streams-h2o-docker-microservice-1.0-SNAPSHOT-jar-with-dependencies.jar /opt/
ENTRYPOINT ["java", "-jar", "/opt/kafka-streams-h2o-docker-microservice-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

```
dcos kafka --name confluent-kafka topic create AirlineInputTopic --partitions 10 --replication 3
dcos kafka --name confluent-kafka topic create AirlineOutputTopic --partitions 10 --replication 3
```

<https://hub.docker.com/r/megachucky/kafka-streams-machine-learning-docker-microservice/>
<https://github.com/kaiwaeher/kafka-streams-machine-learning-docker-microservice>

Kafka Streams Microservice on DC/OS



Run a Service
JSON EDITOR
REVIEW & RUN

Service

Networking

Volumes

Health Checks

Environment

Service

Configure your service below. Start by giving your service an ID.

SERVICE ID ⓘ

Give your service a unique name within the cluster, e.g. my-service.

INSTANCES

CONTAINER IMAGE ⓘ

Enter a Docker image you want to run, e.g. nginx.

CPUs *

Memory (MiB) *

COMMAND ⓘ

A shell command for your container to execute.

[MORE SETTINGS](#)

Container Runtime ⓘ

The container runtime is responsible for running your service. We support the Mesos and Docker containerizers.

DOCKER ENGINE

Docker's container runtime. No support for multiple containers (Pods) or GPU resources.

MESOS RUNTIME

Universal Container Runtime (UCR) using native Mesos engine. Supports Docker file format, multiple containers (Pods) and GPU resources.

Services >
kafka-streams Running (1)
> kafka-streams

Details Files Logs

ERROR (STDERR)

OUTPUT (STDOUT)

↓

```

INFO task [0_8] Initializing state stores (org.apache.kafka.streams.processor.internals.StreamTask:140)
INFO task [0_8] Initializing processor nodes of the topology (org.apache.kafka.streams.processor.internals.StreamTask:333)
INFO stream-thread [StreamThread-1] Creating active task 0_9 with assigned partitions [AirlineInputTopic-9] (org.apache.kafka.streams.processor.internals.StreamThread:858)
INFO task [0_9] Initializing state stores (org.apache.kafka.streams.processor.internals.StreamTask:140)
INFO task [0_9] Initializing processor nodes of the topology (org.apache.kafka.streams.processor.internals.StreamTask:333)
INFO stream-thread [StreamThread-1] State transition from ASSIGNING_PARTITIONS to RUNNING. (org.apache.kafka.streams.processor.internals.StreamThread:163)
INFO stream-client [kafka-streams-h2o-gbm-example-20b7fce8-274d-4bee-9a5f-4138f994f012] State transition from REBALANCING to RUNNING. (org.apache.kafka.streams.KafkaStreams:224)
INFO stream-thread [StreamThread-1] Committing all tasks because the commit interval 30000ms has elapsed (org.apache.kafka.streams.processor.internals.StreamThread:767)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_0 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_1 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_2 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_3 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_4 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_5 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_6 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_7 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_8 (org.apache.kafka.streams.processor.internals.StreamThread:805)
INFO stream-thread [StreamThread-1] Committing task StreamTask 0_9 (org.apache.kafka.streams.processor.internals.StreamThread:805)
#####
Flight Input:1987,10,14,3,741,730,912,849,PS,1451,NA,91,79,NA,23,11,SAN,SFO,447,NA,NA,0,NA,0,NA,NA,NA,NA,YES,YES
Label (aka prediction) is flight departure delayed: YES
Class probabilities: 0.4319916897116479,0.5680083102883521
#####
                    
```

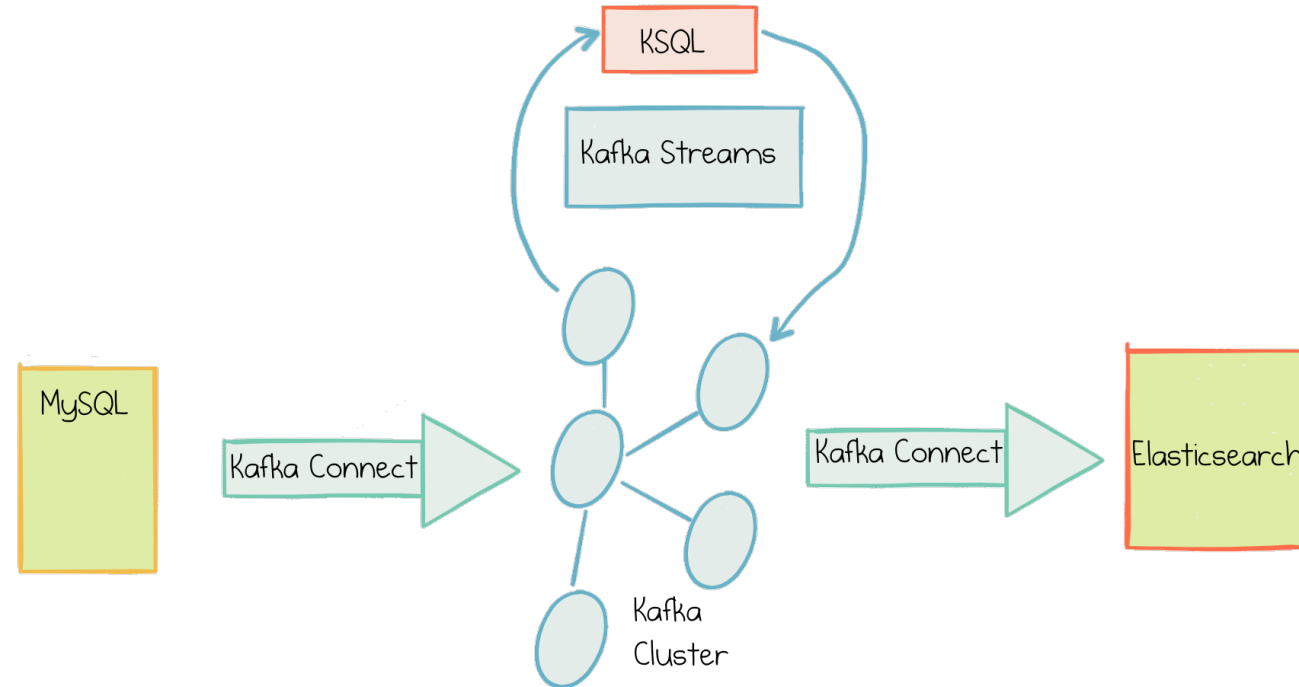
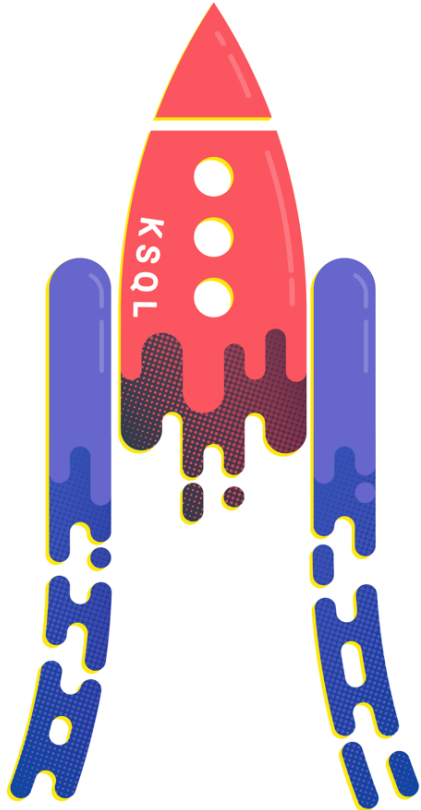
Services >
kafka-streams Running (5)

Instances Configuration Debug

Showing 5 of 38 tasks [Clear](#)

ID	NAME	HOST	STATUS	HEALTH	CPU	MEM	UPDATED	VERSION
<input type="checkbox"/>	kafka-streams.470c70c7-b243-11e7-b6be...	kafka-streams 10.0.2.140	Running	●	2	1 GiB	16 seconds ago	10/16/2017, 9:26:08 AM
<input type="checkbox"/>	kafka-streams.470d3419-b243-11e7-b6be...	kafka-streams 10.0.2.140	Running	●	2	1 GiB	16 seconds ago	10/16/2017, 9:26:08 AM
<input type="checkbox"/>	kafka-streams.470c49b6-b243-11e7-b6be...	kafka-streams 10.0.1.40	Running	●	2	1 GiB	16 seconds ago	10/16/2017, 9:26:08 AM
<input type="checkbox"/>	kafka-streams.470cbee8-b243-11e7-b6be...	kafka-streams 10.0.2.118	Running	●	2	1 GiB	16 seconds ago	10/16/2017, 9:26:08 AM
<input type="checkbox"/>	kafka-streams.f313b4b5-b242-11e7-b6be...	kafka-streams 10.0.2.211	Running	●	2	1 GiB	3 minutes ago	10/16/2017, 9:23:47 AM

KSQL on DC/OS – Why not?



```
SELECT STREAM
  CEIL(timestamp TO HOUR) AS timeWindow, productId, COUNT(*) AS hourlyOrders, SUM(units) AS units
FROM Orders GROUP BY CEIL(timestamp TO HOUR), productId;
```

timeWindow	productId	hourlyOrders	units
08:00:00	10	2	5
08:00:00	20	1	8
09:00:00	10	4	22
09:00:00	40	1	45
...

Key Take-Aways



- Apache Kafka Ecosystem on DC/OS for Highly Scalable, Fault-Tolerant Microservices
- DC/OS offers many Kafka Features out-of-the-box (one-click-provisioning, VIP connection, ...)
- Kafka Streams Microservices run and scale on DC/OS via Marathon or Kubernetes



Questions? Feedback?
Please contact me!



Kai Waehner

Technology Evangelist

kontakt@kai-waehner.de

[@KaiWaehner](#)

www.kai-waehner.de

[LinkedIn](#)

