Running Apache Samza on Kubernetes



Weiqing Yang Jian He

11/19/2019

Agenda

Introduction	About Apache Samza
Deep Dive	Executing Samza Jobs on Kubernetes
Demo	Demo
Deployment	Standalone, Yarn, Kubernetes
Comparison	Kubernetes for Other Big Data Processing Engines
	Q & A

About Apache Samza

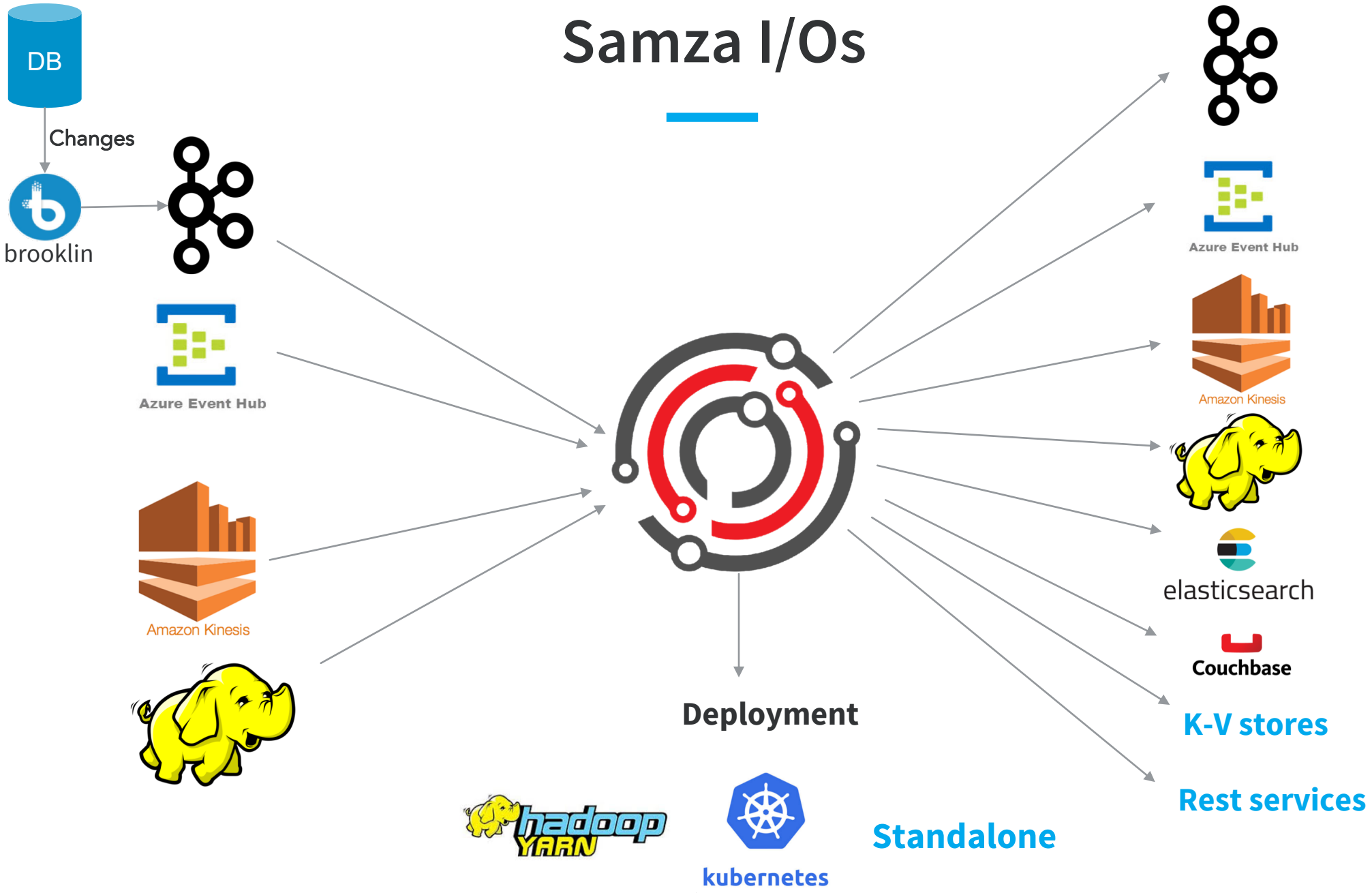
Samza is a distributed stream processing framework that allows you to build stateful applications that process data at scale in real-time.

Users: LinkedIn, Intuit, Slack, TripAdvisor, Optimizely, Redfin, VMWare ...

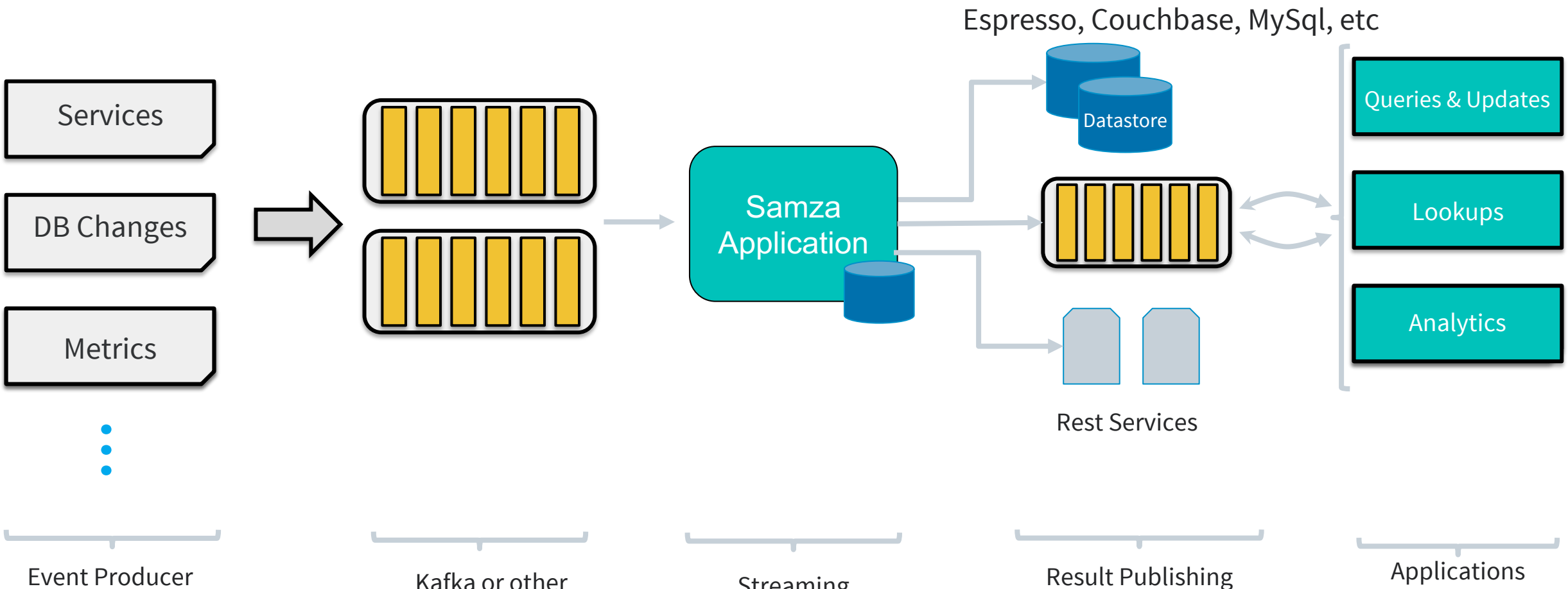
A distributed stream processing framework developed at LinkedIn in 2013

10 major releases and 26 committers since Dec. 2014

Samza I/Os


DB
Changes
brooklin
Azure Event Hub
Amazon Kinesis
Azure Event Hub
Amazon Kinesis
elasticsearch
Couchbase
K-V stores
Rest services
Deployment
hadoop YARN
kubernetes
Standalone

Typical Use Cases of Samza



Samza Features

Flexible API

- API to write stream processing jobs in SQL, Java, Python.

Write Once Run Anywhere

- Flexible deployment options to run anywhere - from public clouds to containerized environments to bare-metal hardware

Massive Scale

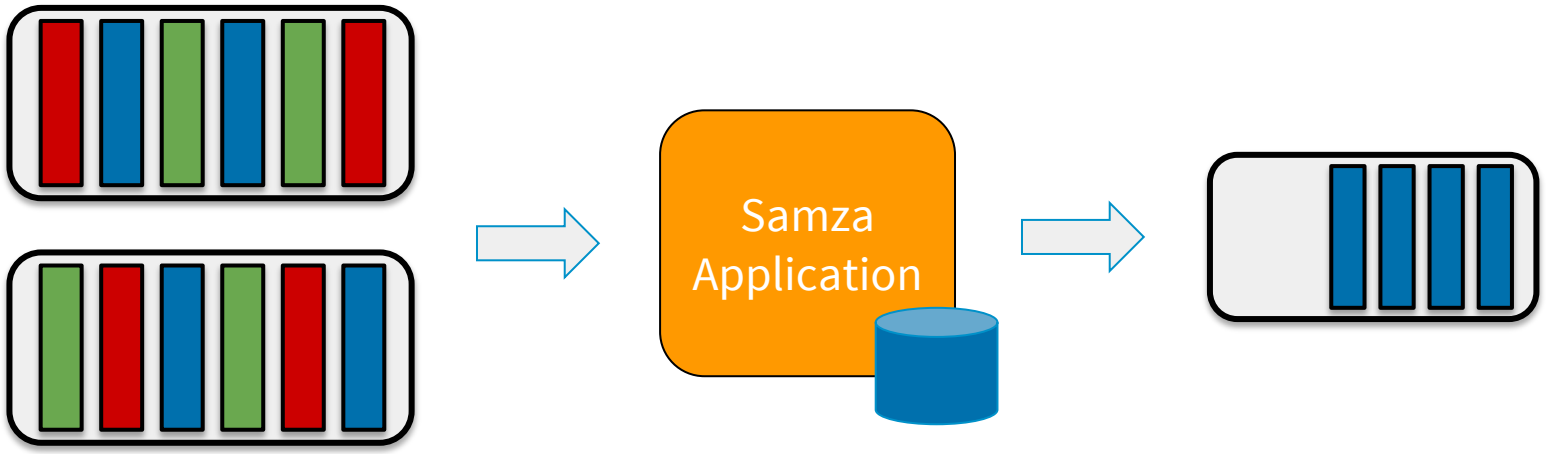
- Battle-tested on applications that use several terabytes of state and run on thousands of cores.

Fault Tolerance

- Transparently migrate associated state in the event of failures.
- At-Least-Once processing semantics

Samza Concept Overview

Samza processes streams. A stream is composed of immutable messages of a similar type or category. In Kafka a stream is a topic.



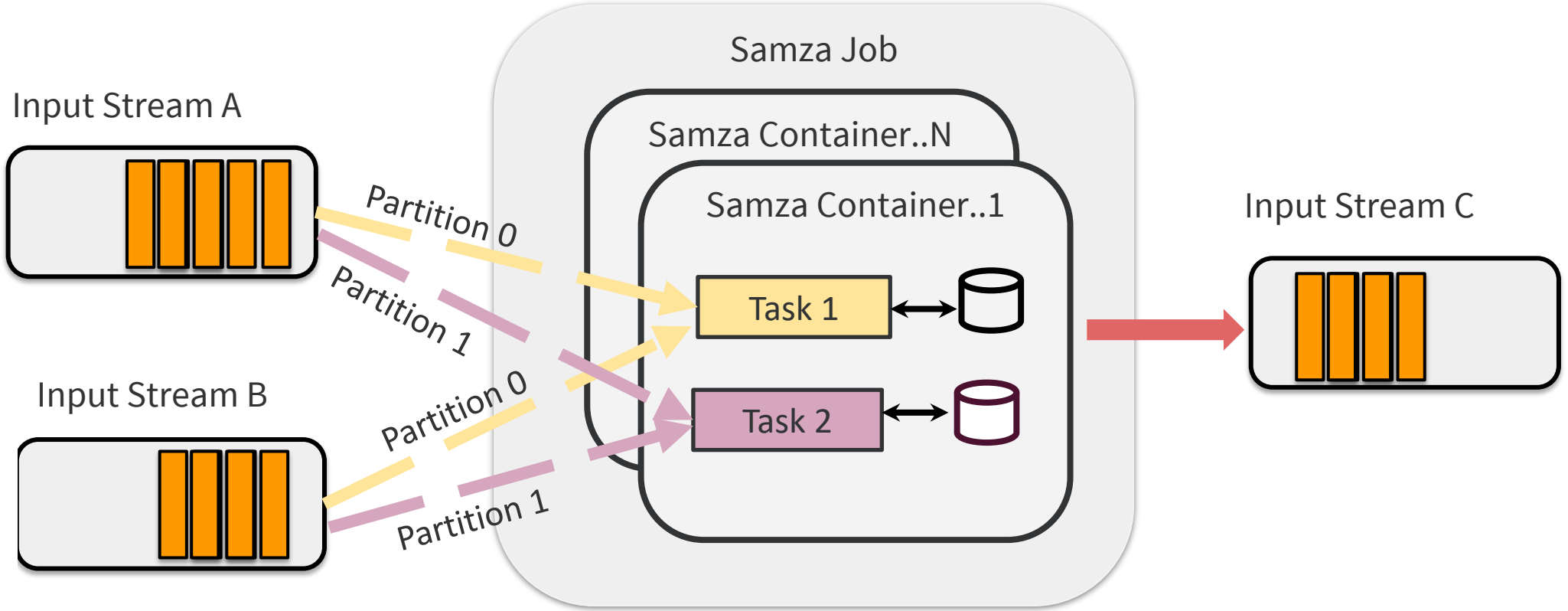
Kafka streams with messages

Samza application that only filters certain kind of messages here blue

Kafka streams with filtered messages

Advanced Concept Overview

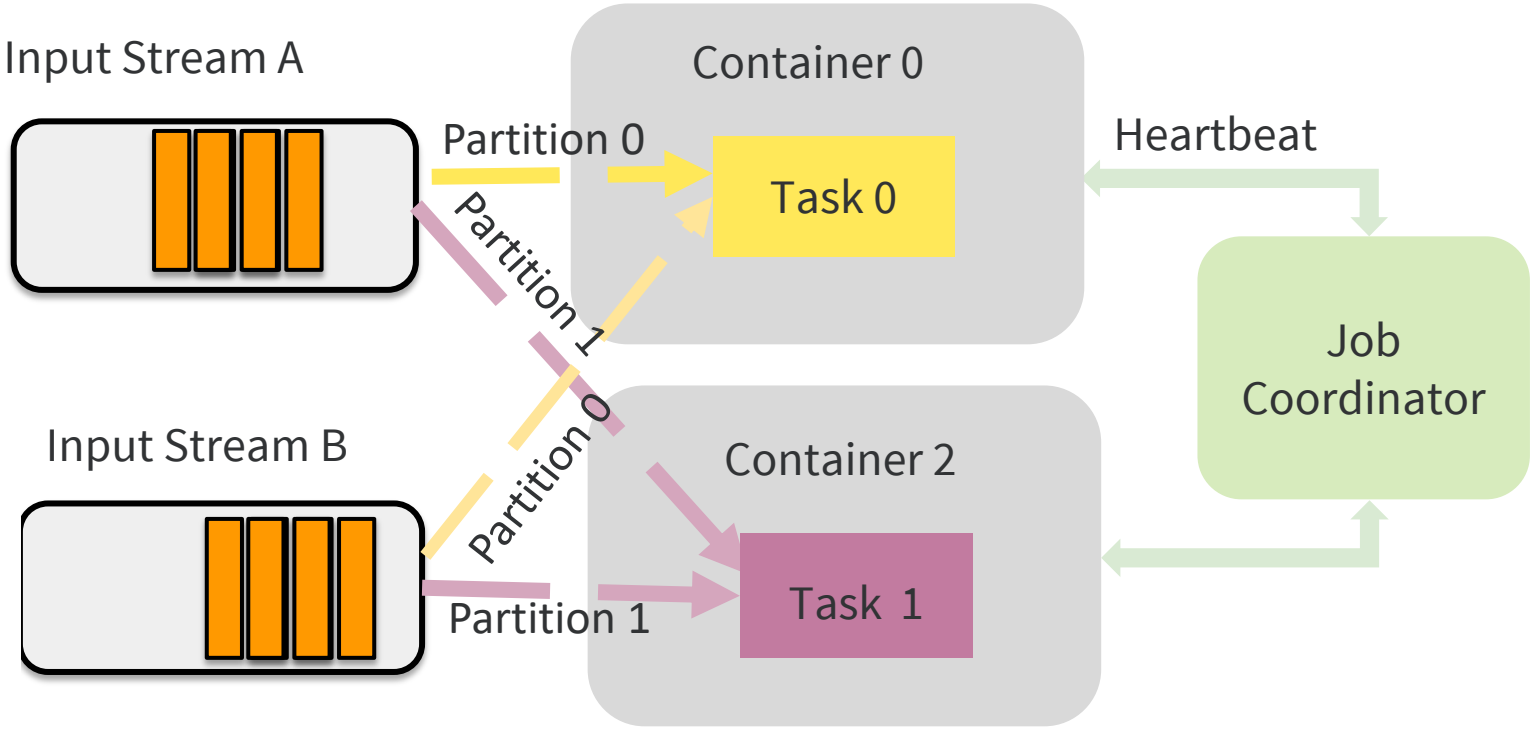
Partition: each stream is broken into one or more partitions, which is an ordered, replayable sequence of records. **Task**: the unit of parallelism of the job, just as the partition is to the stream.



Advanced Concept Overview

Job Coordinator:

- manage the assignment of tasks across the individual containers
- monitor the liveness of individual containers
- redistribute the tasks among the remaining ones during a failure



Samza & Kubernetes: Working Together

Streaming Engine

01 Large-scale distributed stream processing

02 Scalable and durable local state

03 Fault-tolerance and fast recovery

Container-Orchestration System

kubernetes

01 Container orchestration

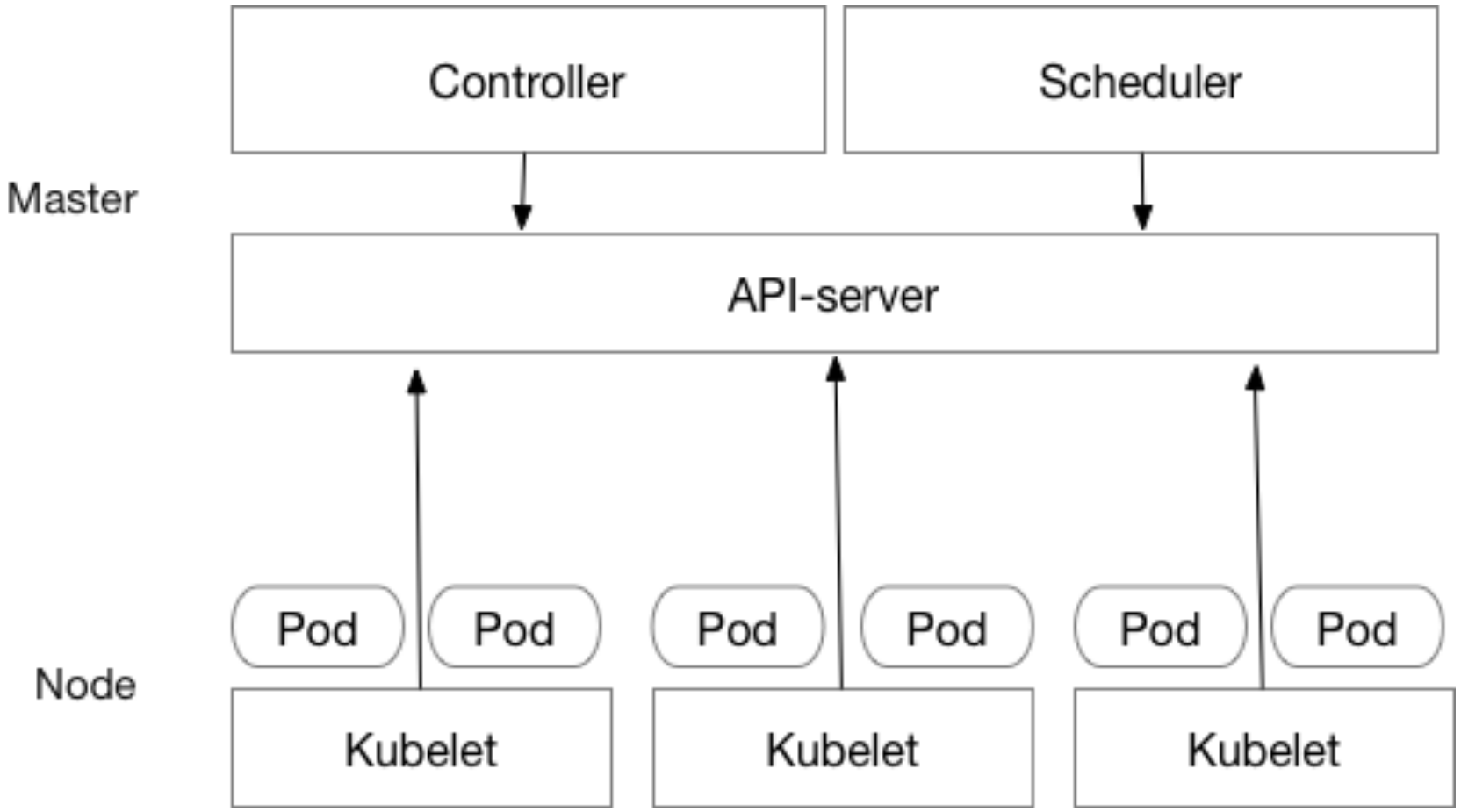
02 Remote or local persistent volume

03 Health checks & operators

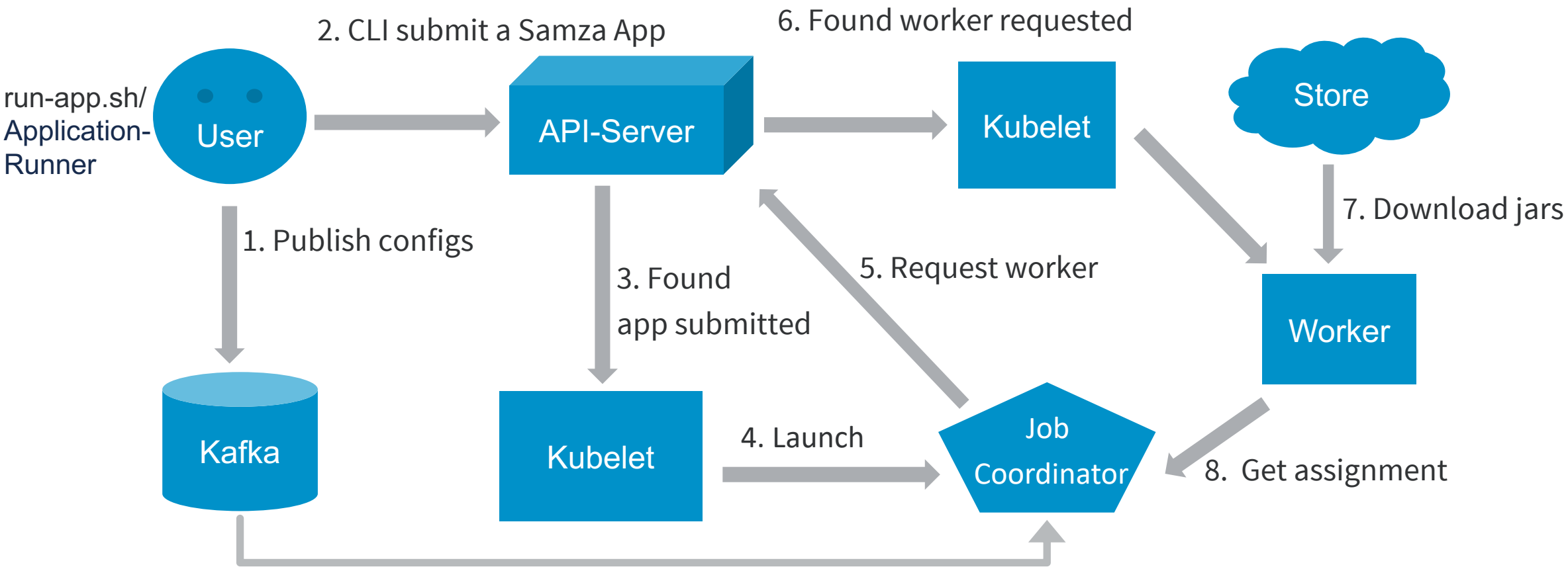
Agenda

Introduction	About Apache Samza
Deep Dive	Executing Samza Jobs on Kubernetes
Demo	Demo
Deployment	Standalone, Yarn, Kubernetes
Comparison	Kubernetes for Other Big Data Processing Engines
	Q & A

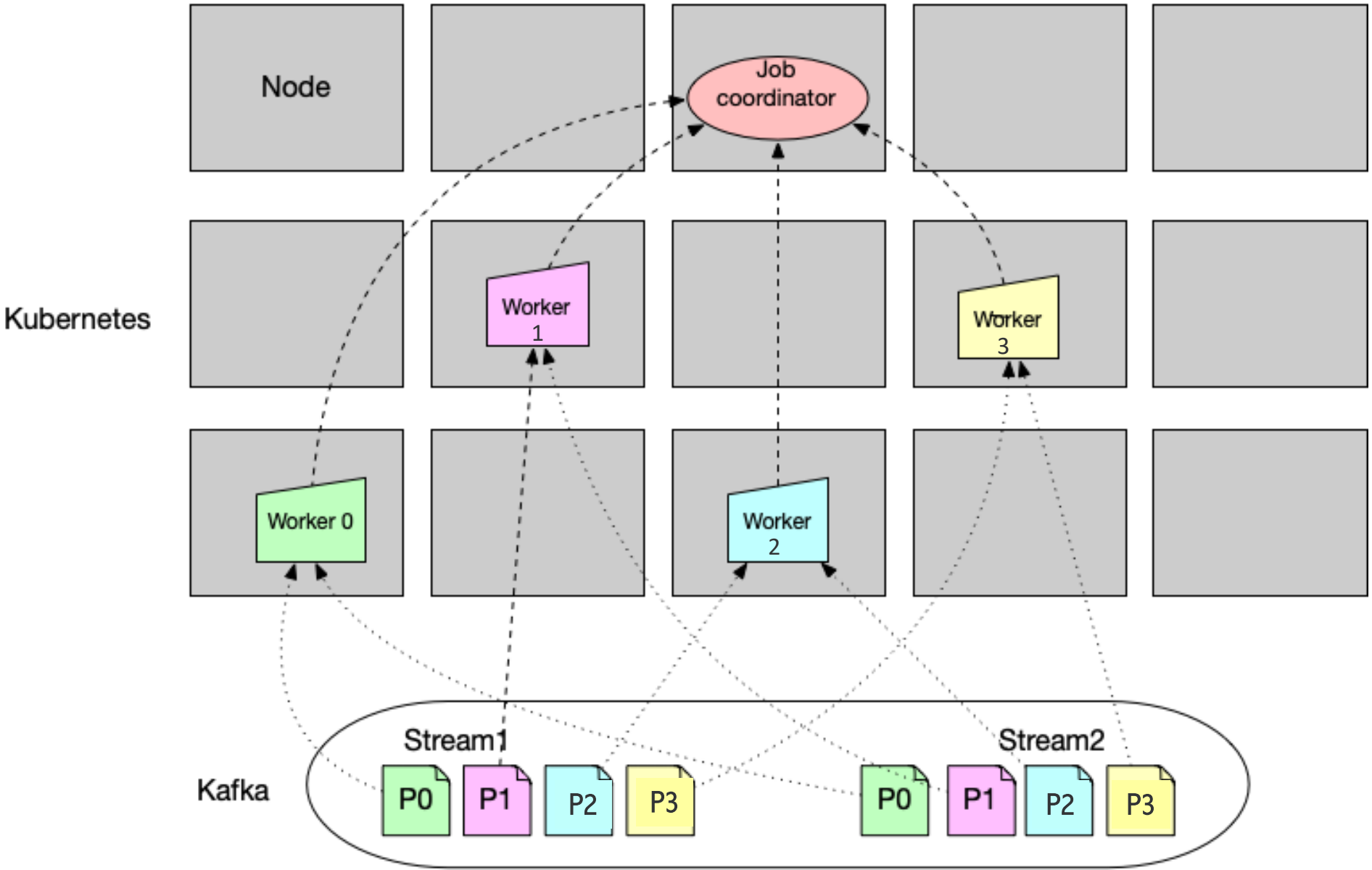
Kubernetes Recap



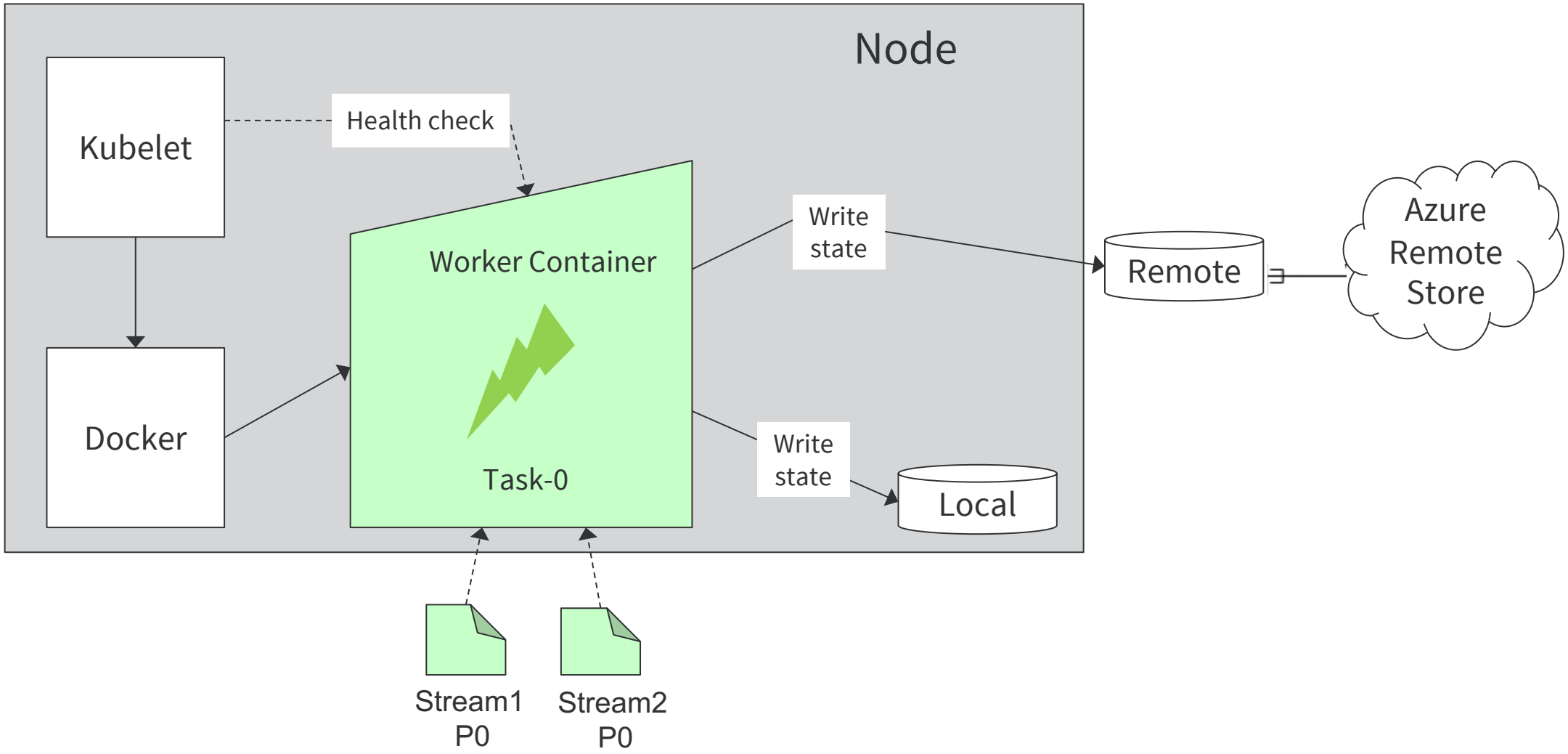
Workflow


run-app.sh/
Application-
Runner
User
2. CLI submit a Samza App
API-Server
6. Found worker requested
Kubelet
Store
7. Download jars
1. Publish configs
3. Found
app submitted
5. Request worker
Worker
Kafka
Kubelet
4. Launch
Job
Coordinator
8. Get assignment

Overview - Samza on Kubernetes



Node – Zoom In


Node
Kubelet
Health check
Worker Container
Task-0
Docker
Write state
Write state
Remote
Azure Remote Store
Local
Stream1 P0
Stream2 P0

Agenda

Introduction	About Apache Samza
Deep Dive	Executing Samza Jobs on Kubernetes
Demo	Demo
Deployment	Standalone, Yarn, Kubernetes
Comparison	Kubernetes for Other Big Data Processing Engines
	Q & A

Demo

Run “wikipedia-application” job on Azure Kubernetes cluster (AKS):

- Samza version: 1.3 (latest master-branch code)
- Docker image: weiqingyang/hello-samza-new:v0
- Task: consumes the real-time feeds from Wikipedia, extracts the metadata of the events, and calculates statistics of all edits in a 10-second window. The application code can be found here: https://tinyurl.com/t8gy87h
  - Merge wikipedia, wiktionary, and wikinews events into one stream
  - Parse each event to a more structured format
  - Aggregate some stats over a 10s window
  - Format each window output for public consumption
  - Send the window output to the **wikipedia-stats** Kafka topic. The messages in the stats topic look like this:

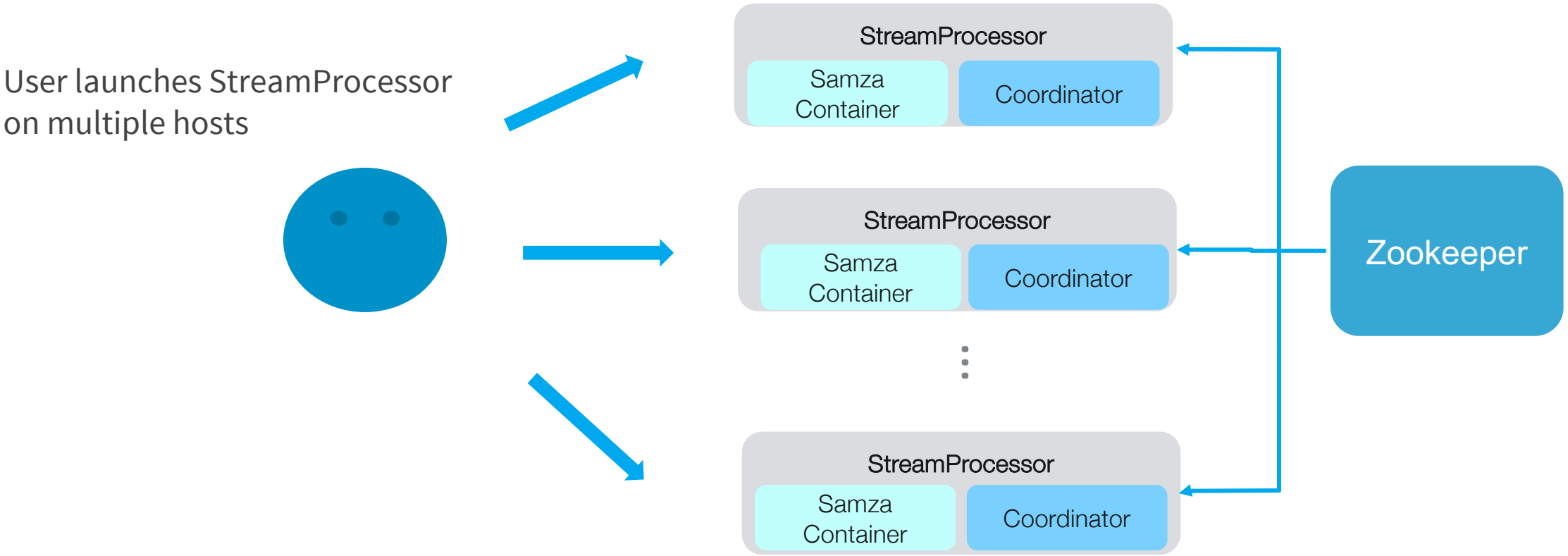
```
{"edits":1,"editsAllTime":0,"bytesAdded":445,"uniqueTitles":1,"counts":{}}
{"edits":2,"editsAllTime":0,"bytesAdded":1,"uniqueTitles":2,"counts":{"is-minor":1}}
{"edits":2,"editsAllTime":0,"bytesAdded":-301,"uniqueTitles":2,"counts":{}}
…
```

Agenda

Introduction	About Apache Samza
Deep Dive	Executing Samza Jobs on Kubernetes
Demo	Demo
Deployment	Standalone, Yarn, Kubernetes
Comparison	Kubernetes for Other Data Processing Engines
	Q & A

Samza Standalone

- Samza StreamProcessor is statically placed on hosts
- ZooKeeper for membership management and task coordination

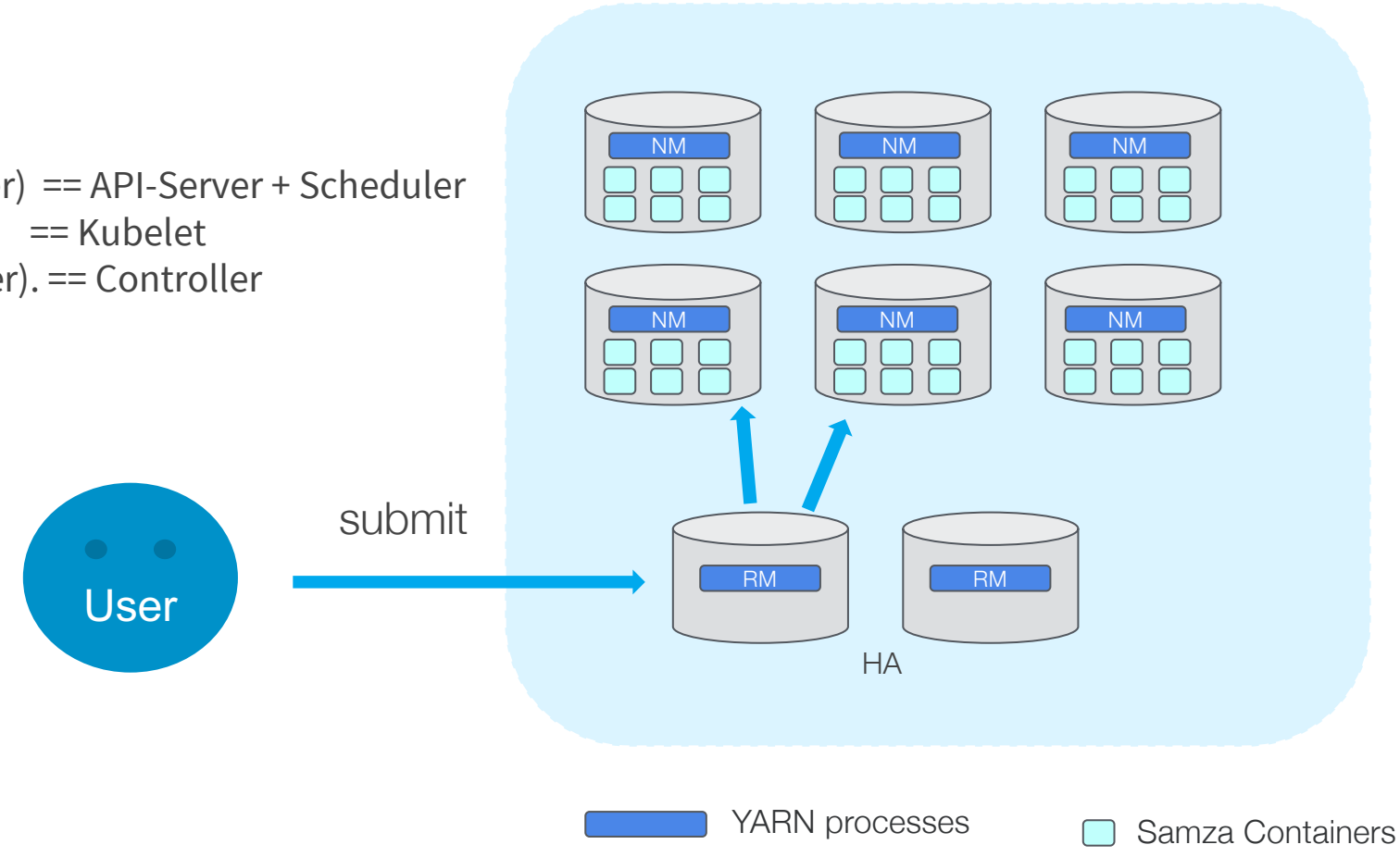


Samza on YARN

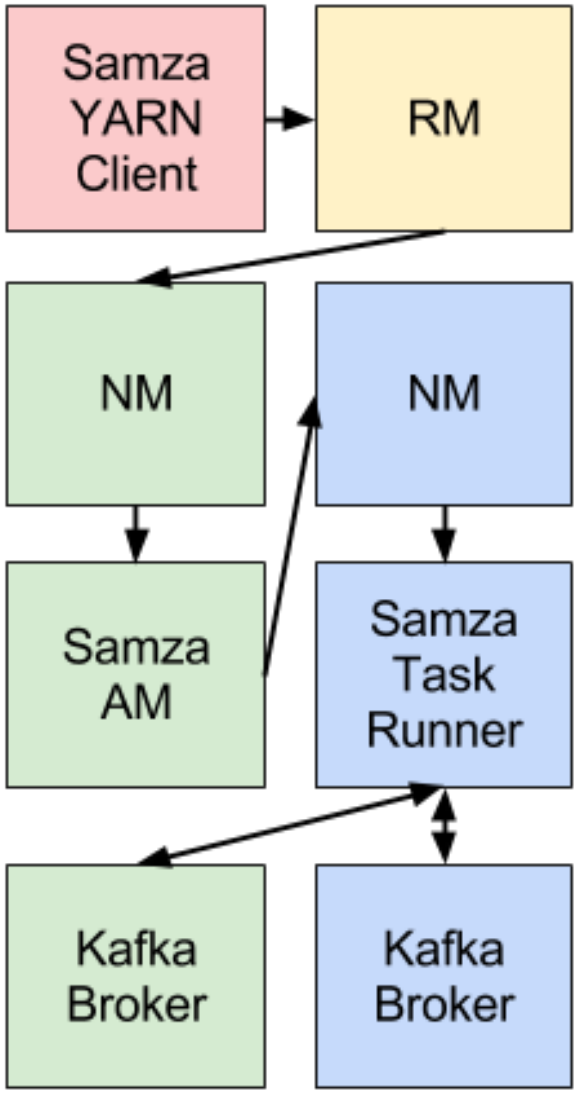
Samza leverages YARN for scheduling, resource-management, and deployment.

Yarn Cluster

RM(Resource Manager) == API-Server + Scheduler
NM(Node Managers) == Kubelet
AM(Application Master). == Controller



Samza on YARN



1. Client Submits to ResourceManager
2. ResourceManager talks to NodeManager to launch Samza ApplicationMaster
3. Samza ApplicationMaster asks N container and launch on Samza Task on NodeManagers
4. SamzaTasks then reads partition streams from Broker

* Picture from Apache Samza document

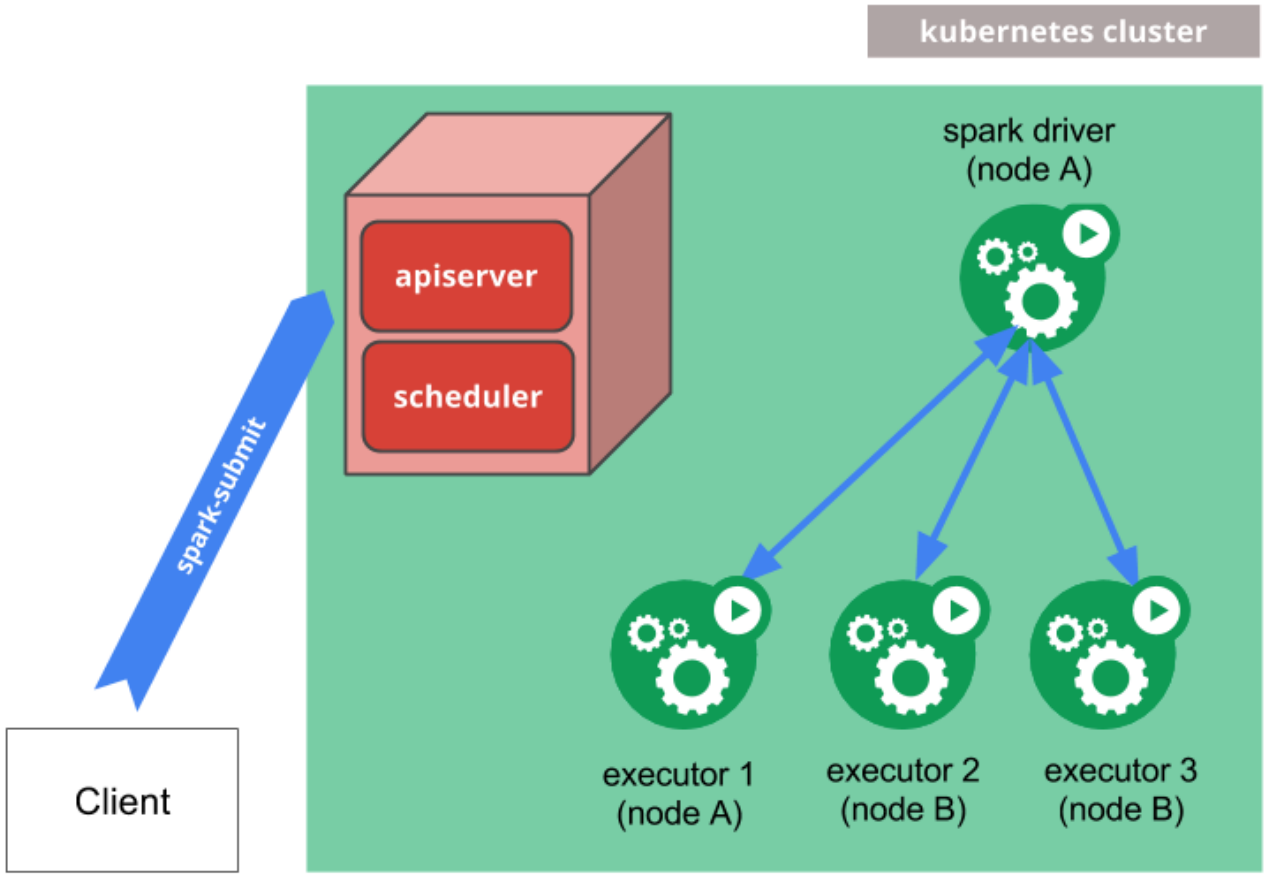
Kubernetes vs Apache Yarn

Kubernetes	- Good for long running service - Level triggered design principal: Driving current state towards desired state - Self - healing: Ideal for automated daily operations
YARN	- Big data ecosystems: Spark, Flink, Samza, Mapreduce - Good for batch Jobs - First-class ‘Job’ concept, e.g. Job priority. Job scheduling. Compared with pod in Kubernetes

Agenda

Introduction	About Apache Samza
Deep Dive	Executing Samza Jobs on Kubernetes
Demo	Demo
Deployment	Standalone, Yarn, Kubernetes
Comparison	Kubernetes for Other Data Processing Engines
	Q & A

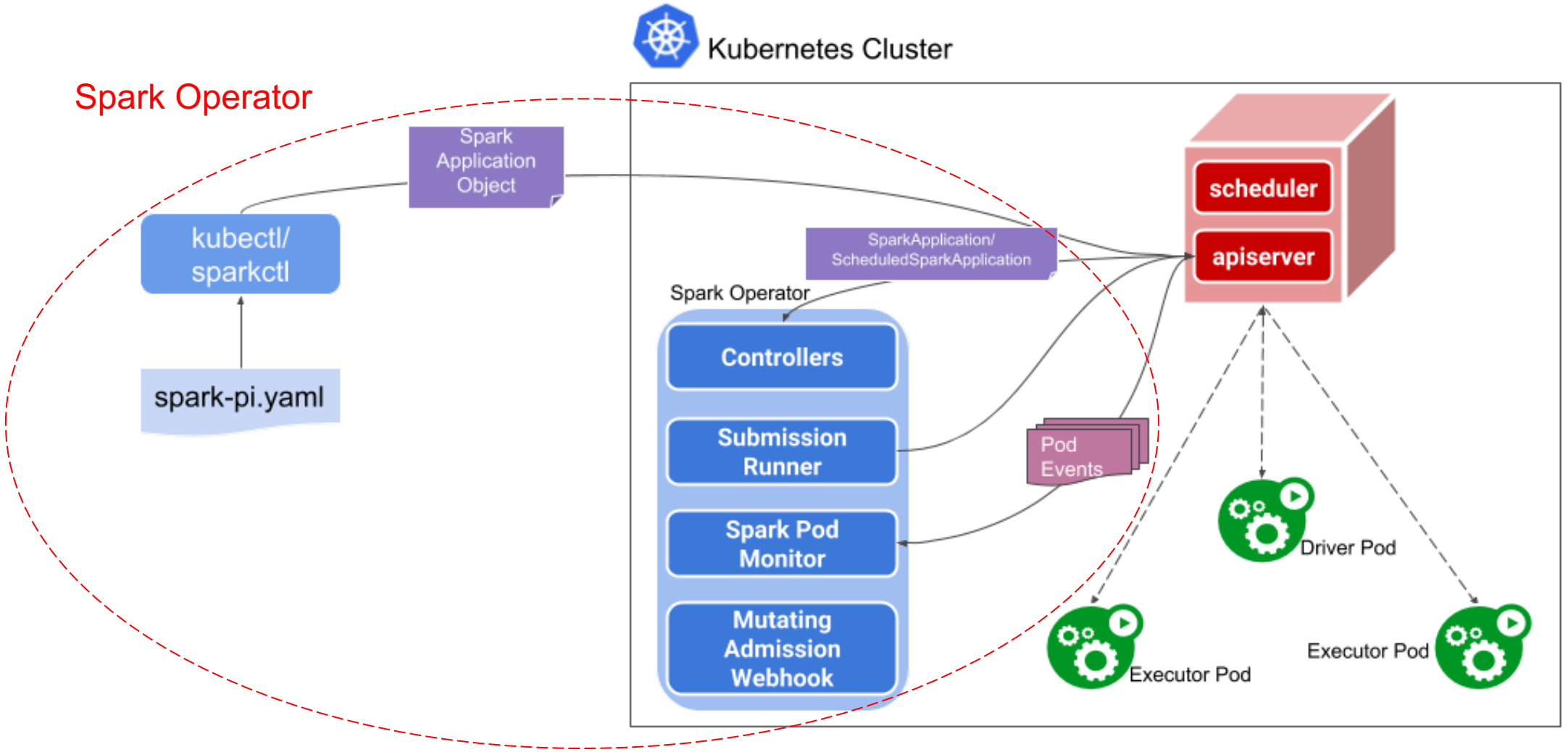
Kubernetes for Spark



- Similar to Samza on Kubernetes
- Spark driver == Samza job coordinator
- Spark executor == Samza worker
- Client submits **a Spark Driver** pod
- Scheduler watches the pod and assigns a node
- The node launches the driver pod.
- The driver pod creates N pods to run **executors**

* Picture from Apache Spark website

Spark Operator on Kubernetes



* Picture from Spark operator website

Spark Operator - Spark Application Definition

Spec

```yaml
apiVersion: sparkoperator.k8s.io/v1beta2
kind: SparkApplication
metadata:
  ...
spec:
  deps: {}
  driver:
    coreLimit: 1200m
    cores: 1
    labels:
      version: 2.3.0
    memory: 512m
    serviceAccount: spark
  executor:
    cores: 1
    instances: 1
    labels:
      version: 2.3.0
    memory: 512m
  image: gcr.io/ynli-k8s/spark:v2.4.4
  mainApplicationFile: local:///opt/spark/examples/jars/spark-examples_2.11-2.3.0.j
  mainClass: org.apache.spark.examples.SparkPi
  mode: cluster
  restartPolicy:
      type: OnFailure
      onFailureRetries: 3
      onFailureRetryInterval: 10
      onSubmissionFailureRetries: 5
      onSubmissionFailureRetryInterval: 20
  type: Scala
```

Status

```yaml
status:
  sparkApplicationId: spark-5f4ba921c85ff3f1cb04bef324f9154c9
  applicationState:
    state: COMPLETED
  completionTime: 2018-02-20T23:33:55Z
  driverInfo:
    podName: spark-pi-83ba921c85ff3f1cb04bef324f9154c9-driver
    webUIAddress: 35.192.234.248:31064
    webUIPort: 31064
    webUIServiceName: spark-pi-2402118027-ui-svc
    webUIIngressName: spark-pi-ui-ingress
    webUIIngressAddress: spark-pi.ingress.cluster.com
  executorState:
    spark-pi-83ba921c85ff3f1cb04bef324f9154c9-exec-1: COMPLETED
  LastSubmissionAttemptTime: 2018-02-20T23:32:27Z
```

* Pictures from Spark operator website

Kubernetes for Flink

- Flink - another popular streaming processing engine
- Flink is composed of **JobManager** (Samza Job Coordinator) and **TaskManager** (Samza Worker)
- Use K8s **Deployment** primitive to launch JobManager and **N** replicated TaskManagers
- Pro: leverage existing robust K8s workload primitive, minimal code changes
- Cons: not as flexible as Samza or Spark approach, e.g. Run a pod on a specific node

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: flink-jobmanager
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: flink
        component: jobmanager
    spec:
      containers:
      - name: jobmanager
        image: flink:latest
        workingDir: /opt/flink
        command: ["/bin/bash", "-c", "$FLINK_HOME/bin/jobmanager.sh start;\
```

JobManager Deployment for 1 replica

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: flink-taskmanager
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: flink
        component: taskmanager
    spec:
      containers:
      - name: taskmanager
        image: flink:latest
        workingDir: /opt/flink
        command: ["/bin/bash", "-c", "$FLINK_HOME/bin/taskmanager.sh start; \
```

TaskManager Deployment for 2 replicas

* Pictures from Apache Flink website

Thank you !