OPEN
AI + DATA
FORUM

THE LINUX FOUNDATION

OPEN
SOURCE
SUMMIT
EUROPE

Automating Cloud-native Spark Jobs with Argo Workflows

Caelan Urquhart & Darko Janjić, Pipekit

#OSSUMMIT

@pipekitio

pipekit

AGENDA

- Intro & goals for the talk
- Background: Apache Spark, Kubernetes, and Argo Workflows
- Common problems with Spark
- Solutions with Kubernetes and Argo
- Demo & How-to
- Next steps
- Q&A

Intro & goals for the talk

OPEN
AI + DATA
FORUM

#OSSUMMIT

GOALS FOR THE TALK

- Learn about the scaling and stability advantages of running Spark jobs on Kubernetes
- Learn how to use Argo Workflows to deploy and automate Spark jobs on Kubernetes
- Learn what else is possible once you're running Spark on Kubernetes

ABOUT US

pipekit

Caelan Urquhart

Co-founder, CEO @ Pipekit

Contributer to the Argo Workflows project

Focus: Data engineering architecture, product development

Darko Janjić

Senior Software Engineer @ Pipekit

Focus: Distributed systems, virtualization, and cloud infrastructure engineering

Background: Apache Spark, Kubernetes, and Argo Workflows

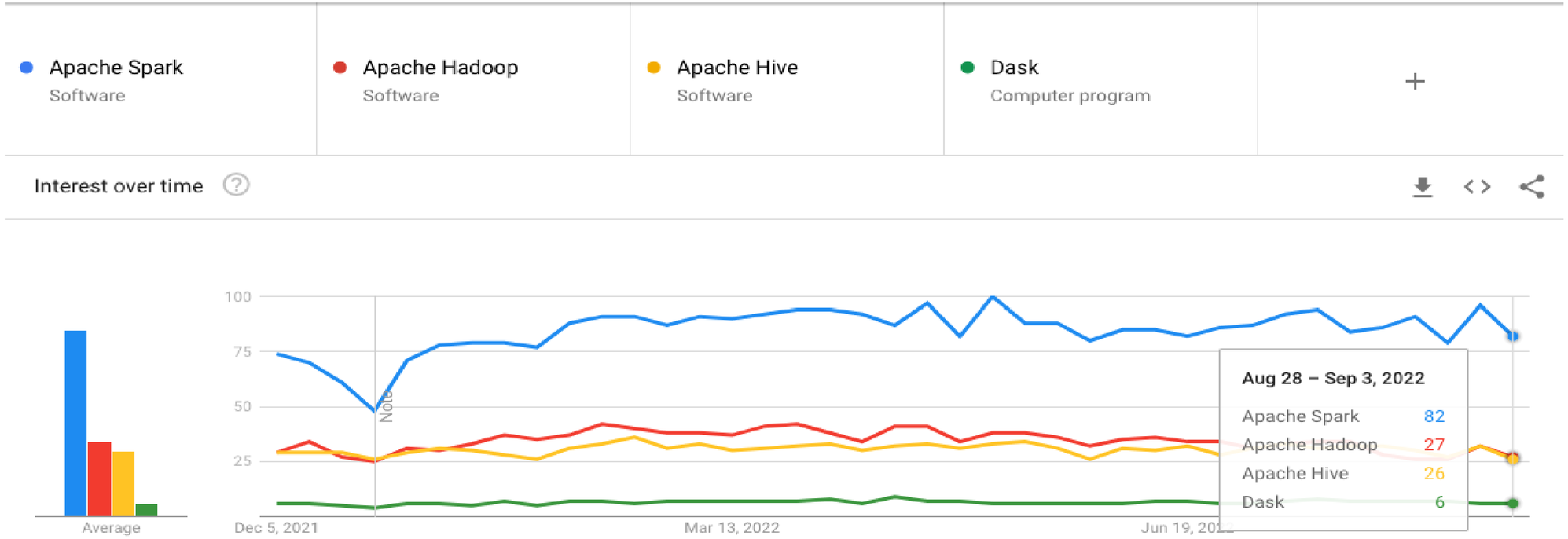
OPEN AI + DATA FORUM

#OSSUMMIT

BACKGROUND: APACHE SPARK

The most popular big data compute framework

- Q: Why? A: Versatility
  - Structured Streaming, MLlib
- Use Python, SQL, Java, Scala or R



PROBLEMS

The typical Spark deployment using YARN …

- Requires global installs (cluster for Spark 3.0 vs. Spark 3.3)
- Doesn't support Docker natively
- Consumes more resources (JVM on each node)
- No autoscaling

… makes solving these problems frustrating

- How do I manage dependencies between different libraries or frameworks?
- How can I run experiments across local and prod environments?
- How can I use different GPUs on one cluster?
- How do I provision resources for "spikey" data jobs?

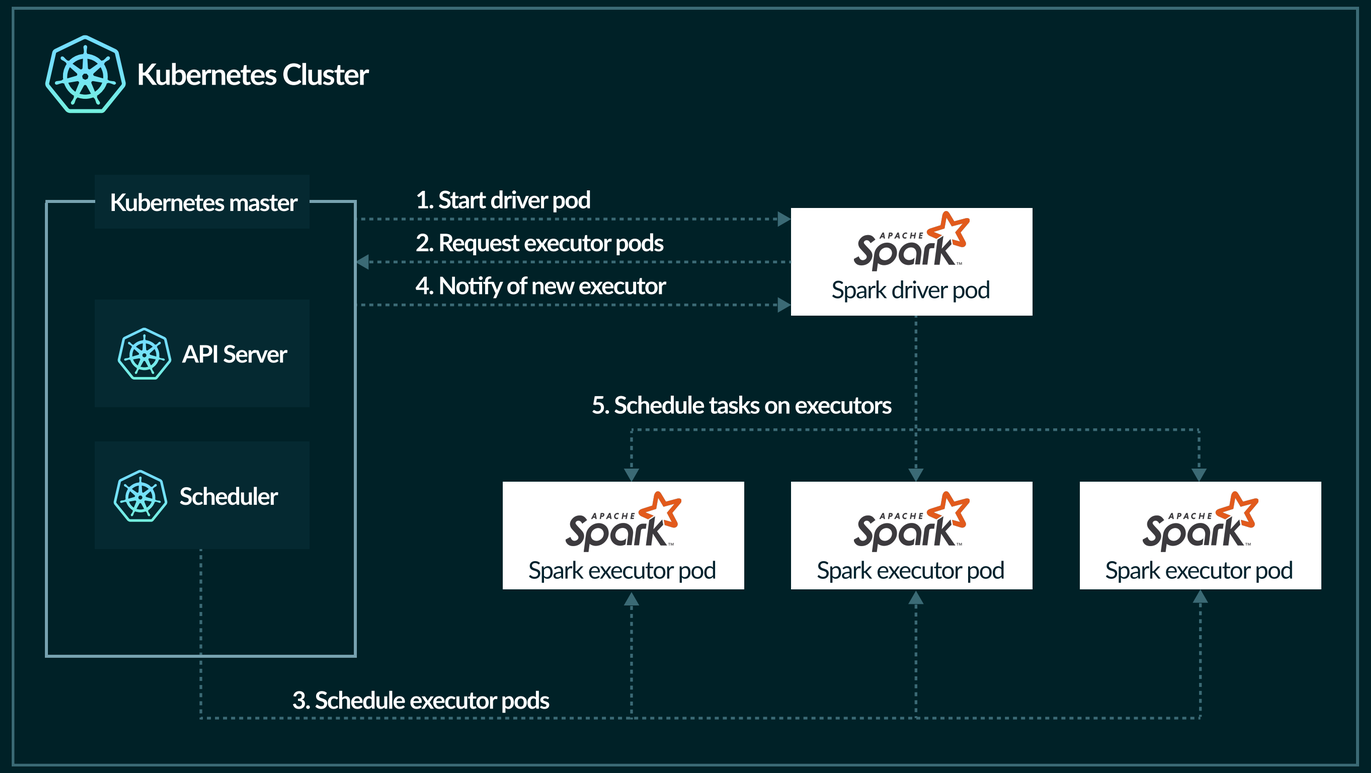
BACKGROUND: KUBERNETES

Kubernetes is a framework for managing and automating containerized applications.

Why use Kubernetes for data science?

- When **containers are your building blocks** you get…
  - Reproducibility
  - Reliability
- **Declaritive** approach to data pipelines
- **Vertical autoscaling**: handle “spikey” data jobs
- Tap into the **cloud-native ecosystem** for additonal tooling: CI/CD, observaibility, etc.

SPARK ON K8S ARCHITECTURE

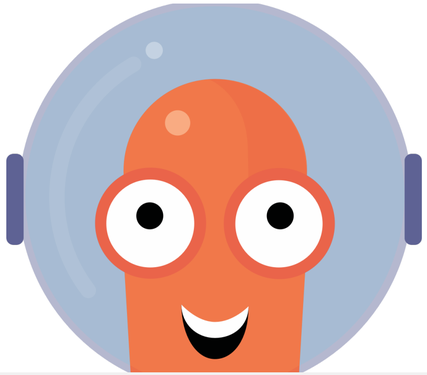


BACKGROUND: ARGO WORKFLOWS

Argo Workflows is the best way to run workloads on K8s

It's **generalizable** - use it to automate anything:
- ETL, ELT, batch data processing
- ML training, serving
- CI/CD workloads

It's a workflow engine (aka DAG orchestrator) that's **lightweight to deploy**



12K+ Github Stars

5K+ Slack Users

Use **YAML or Python** to define workflows

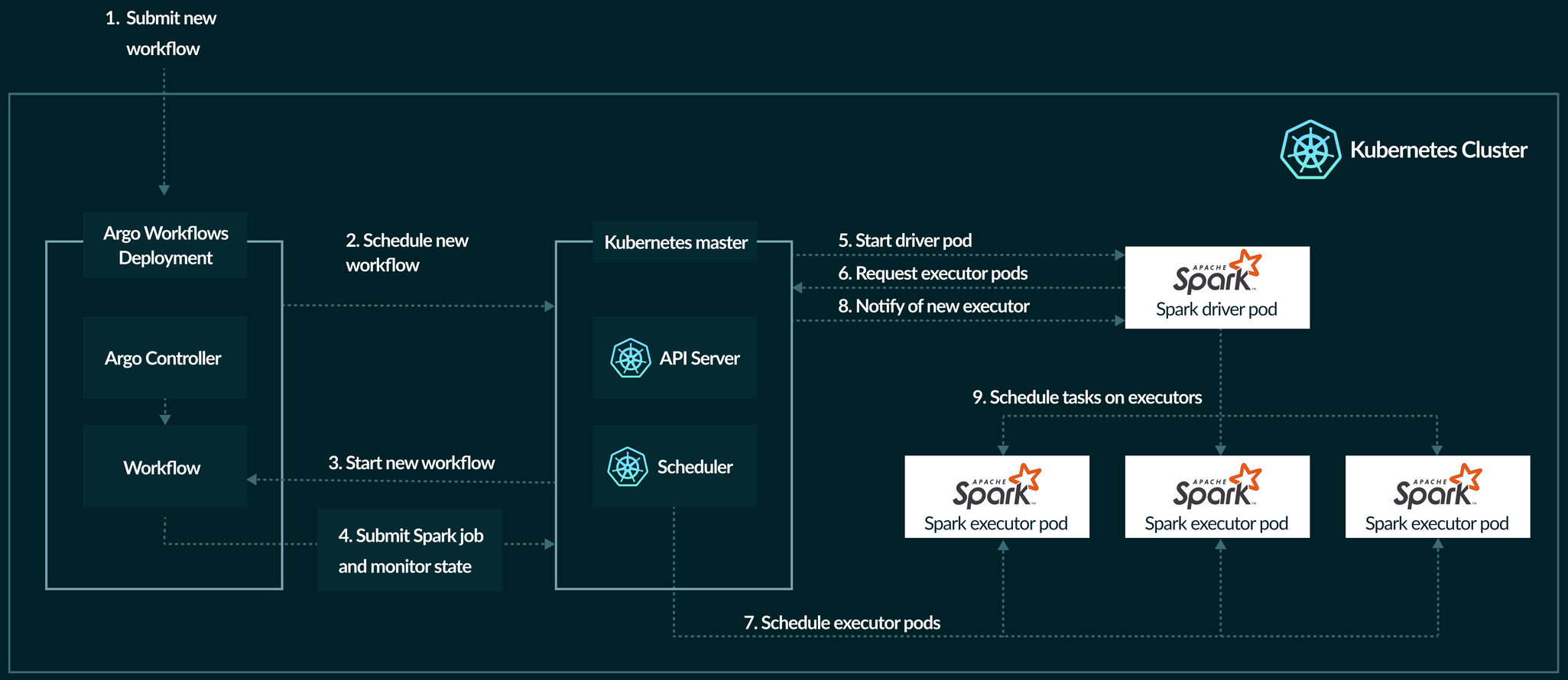
CNCF open source project

Solution

OPEN
AI + DATA
FORUM

#OSSUMMIT

SPARK ON K8S ARCHITECTURE WITH ARGO



SOLUTION

Deploying Spark with K8s and Argo Workflows solves many of the issues with a traditional Spark deployment.

- **Container-native**:
  - run different Spark versions for different jobs on the same cluster
  - manage dependencies easily
  - stop downloading dependencies at runtime = faster devex
  - standardize environments to enable reproducibility

- **Lower resource requirements**

- **Autoscale your compute vertically**

SOLUTION

But wait, there's more! Adopting K8s and Argo Workflows opens up bonus benefits:

- **Duplicable**: use a similar architecture for other frameworks: Dask, Ray, etc.
- **Extendable**: bring other data processing and ML tooling into the same cluster: Apache Kafka, Beam, etc.
- **Cloud-native ecosystem**: add logging, metrics, and more
- **Scalable**: enable CI/CD, GitOps for your data platform
- **Agnostic**: avoid cloud vendor lock-in

Demo & How-to

OPEN
AI + DATA
FORUM

#OSSUMMIT

DEMO & HOW-TO

In this demo, we will...

- Create a Spark job that analyses a Kaggle dataset
- Create and run an Argo Workflow that executes the Spark job on K8s
- Simplify the Argo Workflow manifest by using a WorkflowTemplate

Prerequisites:

- K8s cluster runing locally (K3d, Docker Desktop, Minikube, etc.)
- Spark operator installed
- Argo Workflows v3.3 installed
- Assumes you already have a Docker image for your Spark job

View repo on GitHub: https://github.com/pipekit/talk-demos/tree/main/oss-2022

DEMO & HOW-TO

Spark job (Scala):

```scala
import org.apache.spark.sql.SparkSession

import java.util.concurrent.TimeUnit

object BikeRideLengthJob {

  val fileLocation = "/app/df_1_year.csv"


  def main(args: Array[String]): Unit = {
    Logger.getLogger("org").setLevel(Level.ERROR)

    val spark = SparkSession
      .builder
      .appName("BikeRideLengthJob")
      .getOrCreate()

    val rdd = spark.sparkContext.textFile(fileLocation).map(Bike.toBikeRide)

    val bikeTypeStats = rdd
      .map(x => (x.bikeType, x.rideLength))
      .reduceByKey((x, y) => x + y)
      .sortBy(x => x._2, ascending = false).collect()

    for (elem <- bikeTypeStats) {
      println(s"BikeType ${elem._1}, rideLength in hours ${TimeUnit.MILLISECONDS.toHours(elem._2)}")
    }

    spark.stop()
  }


}
```

DEMO & HOW-TO

Argo Workflows manifest for our Spark jobs as a DAG:

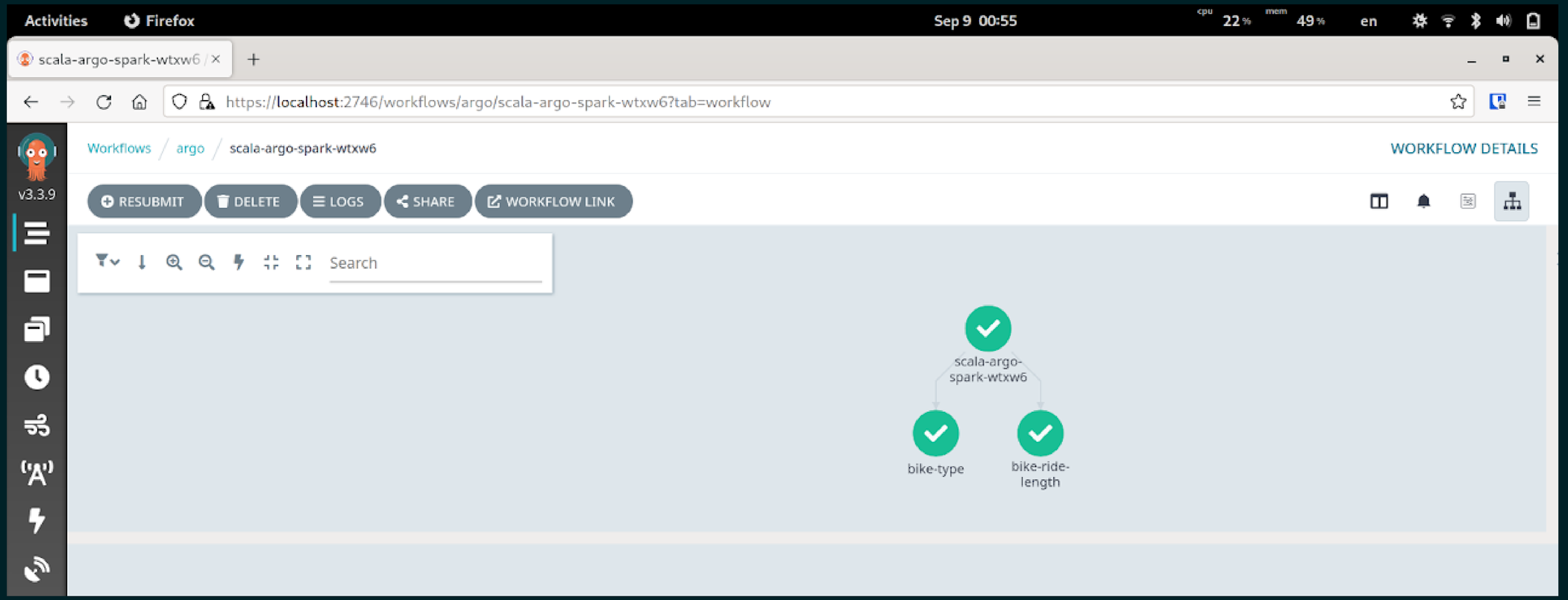
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: scala-argo-spark-
  namespace: argo
spec:
  entrypoint: dag
  templates:
    - name: dag
      dag:
        tasks:
          - name: bike-type
            template: bike-type
          - name: bike-ride-length
            template: bike-ride-length
```

Defining the resources for our Spark job:

```yaml
    resource:
      action: create
      successCondition: status.applicationState.state == COMPLETED
      failureCondition: status.applicationState.state == FAILED
      manifest: |
        apiVersion: "sparkoperator.k8s.io/v1beta2"
        kind: SparkApplication
        metadata:
          generateName: spark-bike-type-
          namespace: default
        spec:
          type: Scala
          mode: cluster
          image: "argo-spark-integration-scala:latest"
          imagePullPolicy: Never
          mainClass: io.pipekit.oss.spark.argo.BikeTypeJob
          mainApplicationFile: "local:///app/bike.jar"
          sparkVersion: "3.1.1"
          restartPolicy:
            type: Never
          driver:
            coreRequest: "200m"
            coreLimit: "99999"
            memory: "1024m"
            serviceAccount: spark-driver
            labels:
              version: 3.1.1
          executor:
            coreRequest: "200m"
            coreLimit: "99999"
            instances: 2
            memory: "1024m"
            labels:
```

DEMO & HOW-TO

A successfully ran Spark job in the Argo UI:



Next steps

OPEN
AI + DATA
FORUM

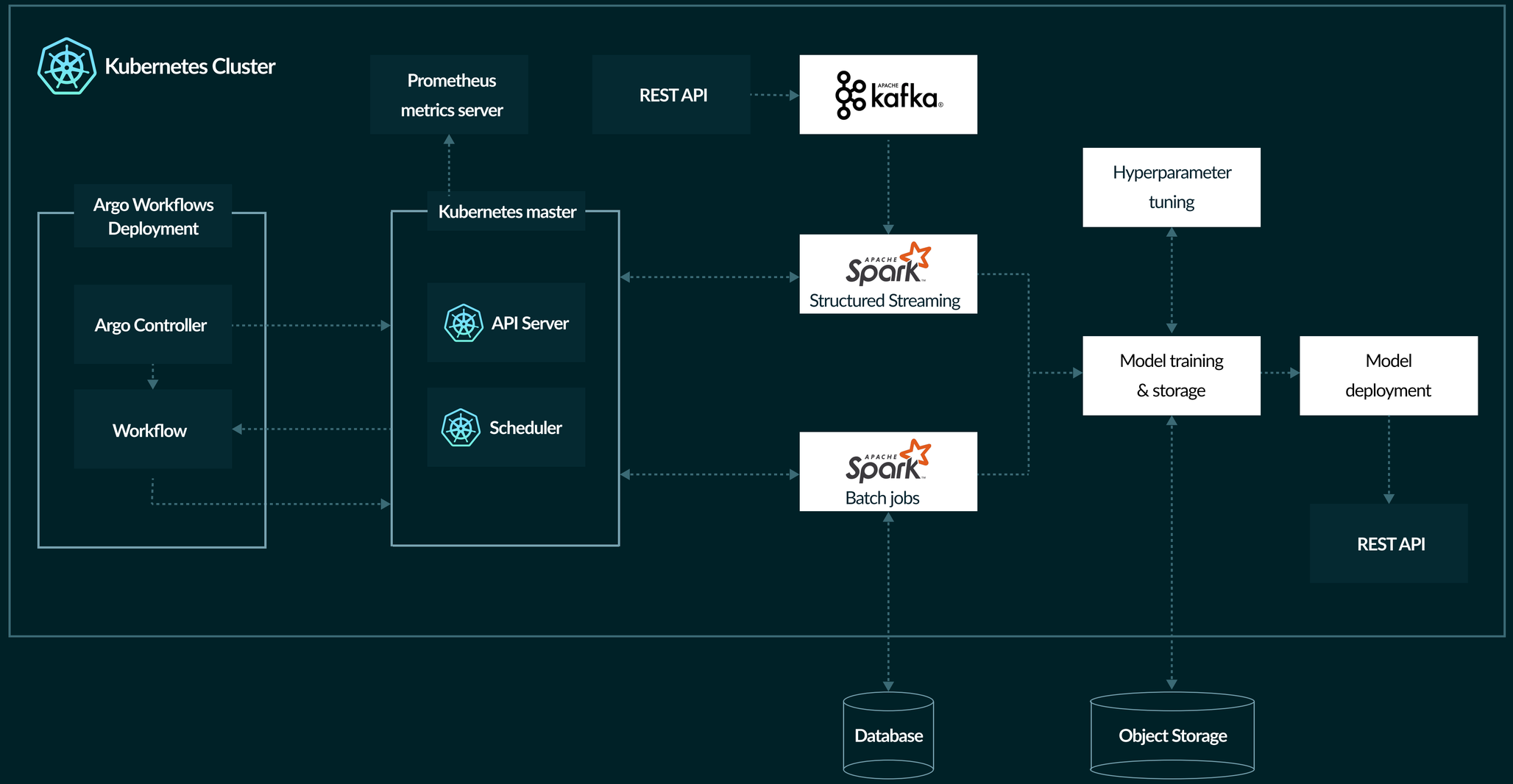
#OSSUMMIT

NEXT STEPS

Now that we are running Spark on K8s, what can we do next?

- Enable CRON workflows
- Define different node pools
- Cluster autoscaling - dynamically scale your Spark cluster on K8s
- Extend your data or ML pipeline with more capabilities

NEXT STEPS


Kubernetes Cluster
Prometheus metrics server
REST API
kafka
Hyperparameter tuning
Argo Workflows Deployment
Kubernetes master
Spark Structured Streaming
Argo Controller
API Server
Model training & storage
Model deployment
Workflow
Scheduler
Spark Batch jobs
REST API
Database
Object Storage

Resources & References

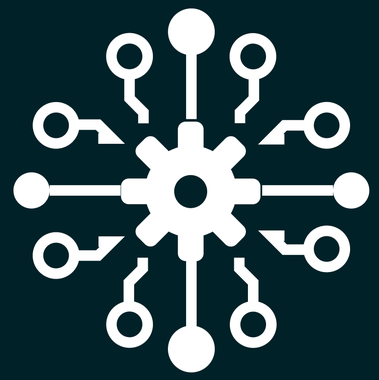
RESOURCES & REFERENCES

- Visit our repo: https://github.com/pipekit/talk-demos/tree/main/oss-2022
- Visit our blog: https://pipekit.io/blog/
- Follow @pipekitio on Twitter
- Spark on K8s Operator GitHub repo
- Kubernetes resource management with Argo Workflows docs

APPENDIX: ARGO WORKFLOWS VS. APACHE AIRFLOW

Argo workflows compared to Apache Airflow

- Lighter weight
- Container-native
- Dynamic, programmatic data pipelines out-of-the-box
- What about Python? Argo has a Python SDK, called Hera



OPEN
AI + DATA
FORUM

THE LINUX FOUNDATION

OPEN
SOURCE
SUMMIT
EUROPE