Google

Running Apache Spark on Kubernetes

A Technical Overview

Data Riders Meetup, February 24, 2018

Agenda

- What is Kubernetes?
- Motivation
- Brief History of the Project
- The Kubernetes Scheduler Backend
- Under the Hood
- Demo
- Future Work

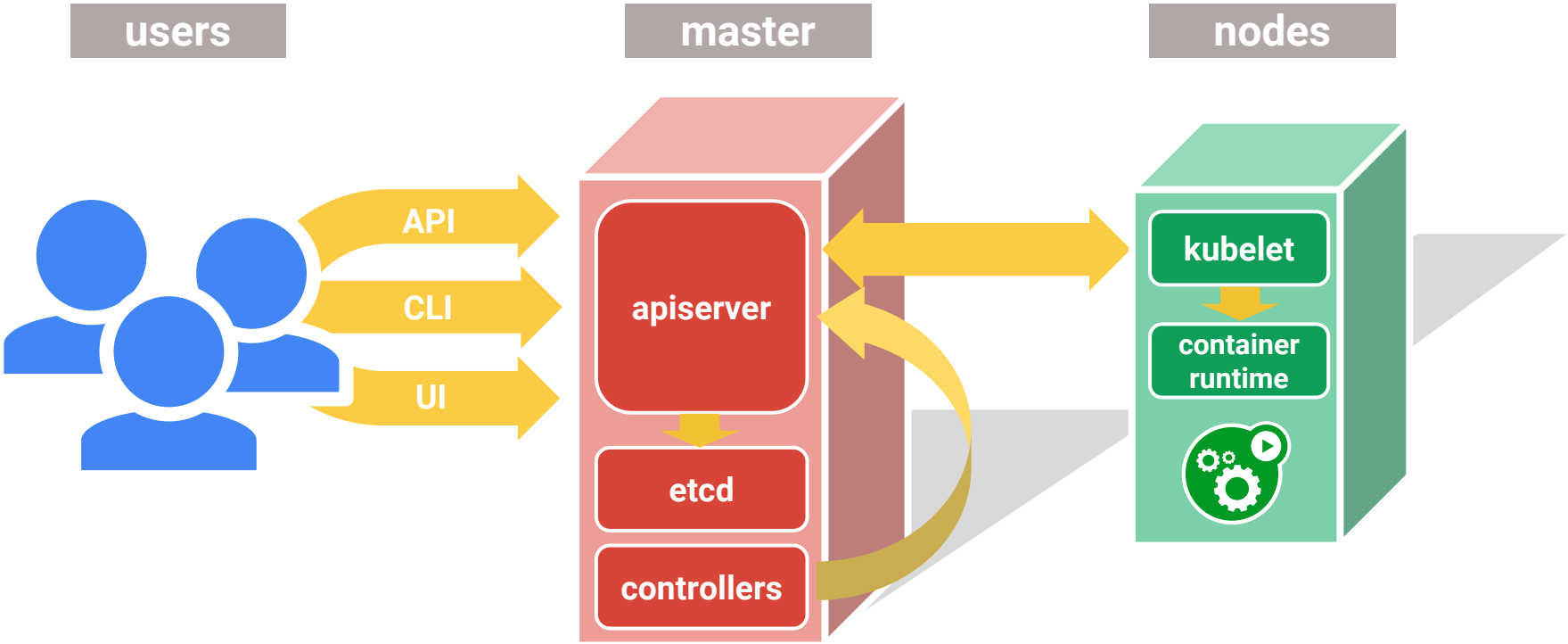
What is Kubernetes?

An open-source container orchestration system for automating deployment, scaling, and management of containerized applications.

Kubernetes

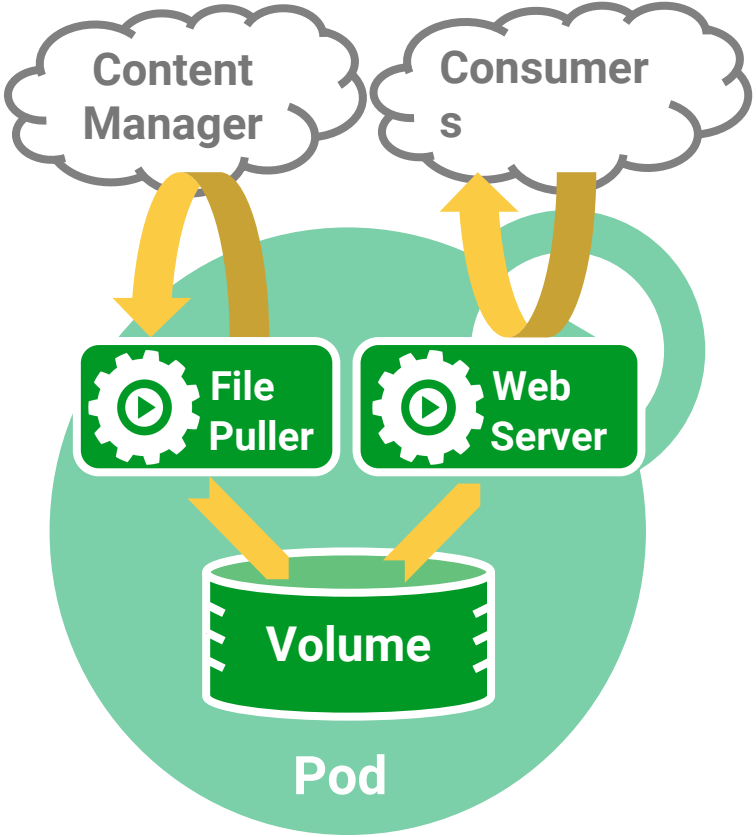
- Large OSS community: 1500+ contributors and 60k+ commits
- Ecosystem and partnership: 100+ organizations involved
- One of the top projects overall on Github: 30k+ stars and 11k+ forks
- Large production deployments both on-prem and on all major cloud providers
- Built with multi-tenant and multi-cloud in mind

API-centric Control Plane



Pods

- **Small group** of containers & volumes
- **Tightly** coupled
- The **atom** of scheduling & placement
- Each pod has its own IP address
- Shared namespace
  - share IP address & localhost
  - share IPC, etc.
- Supports many types of storage volumes



Motivation

- Docker and the container ecosystem
- Kubernetes – Lots of addon services and tooling support
  - Third-party logging, monitoring, and security tools
- Multi-cloud and hybrid environments
- Resource sharing between batch, serving, and stateful workloads
  - Streamlined developer experiences
  - Reduced operational costs
  - Improved infrastructure utilization

Brief History of the Project

Nov 2016
Design

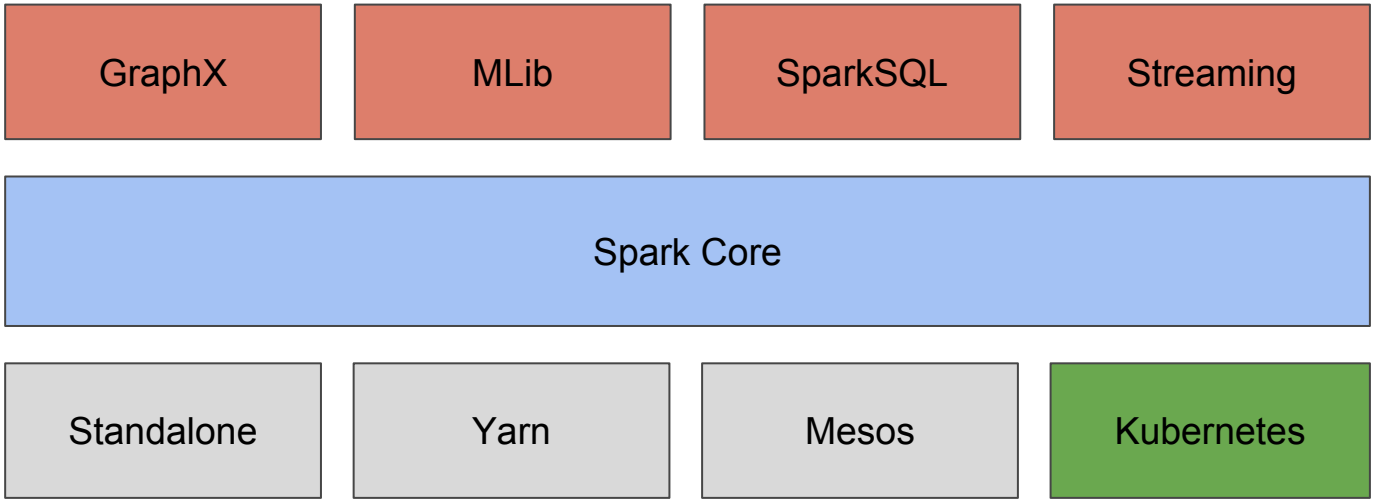
Dec 2016
Development began in a fork

Mid 2017
Port to Spark 2.2
And beta release

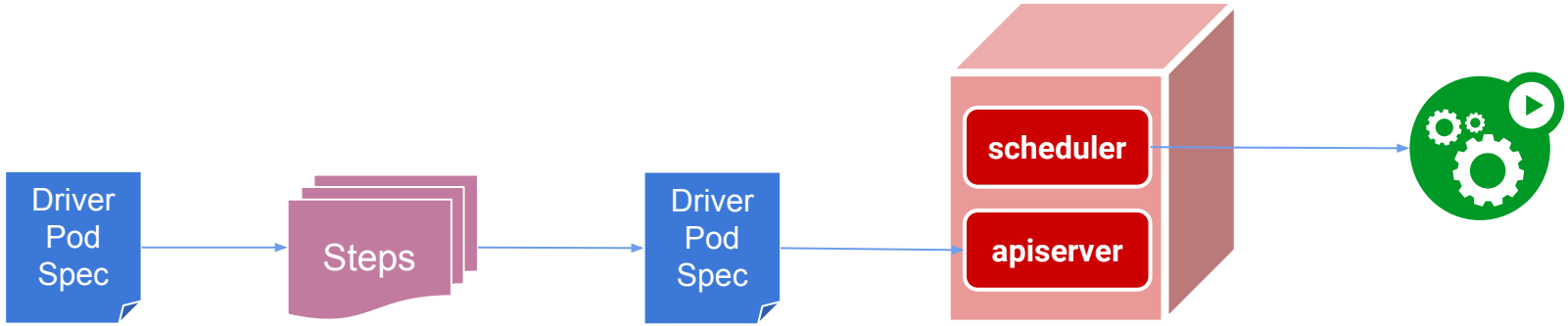
Sep 2017
SPIP passed

Dec 2017 and Jan 2018
Part of the work upstreamed into Spark 2.3

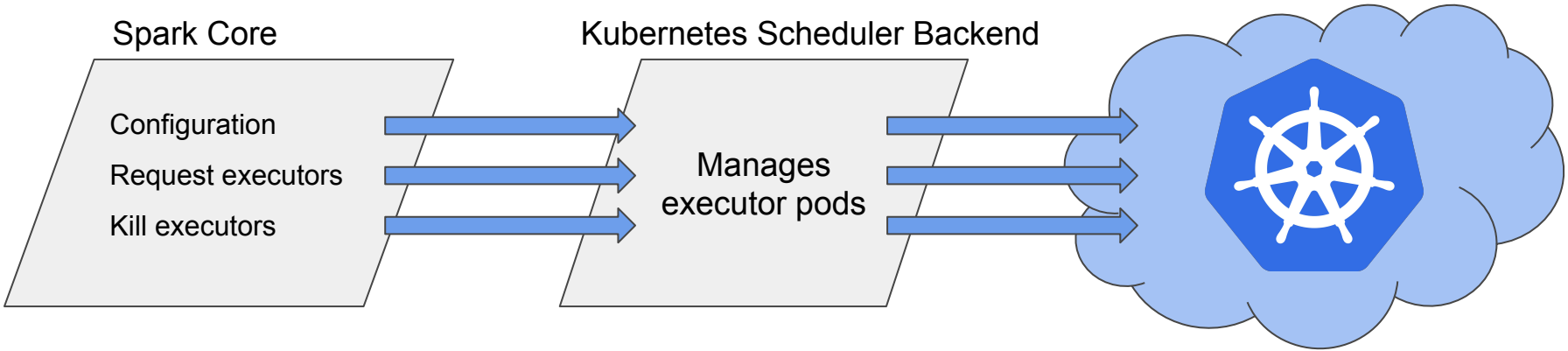
Spark meets Kubernetes



Kubernetes Specific Submission Client



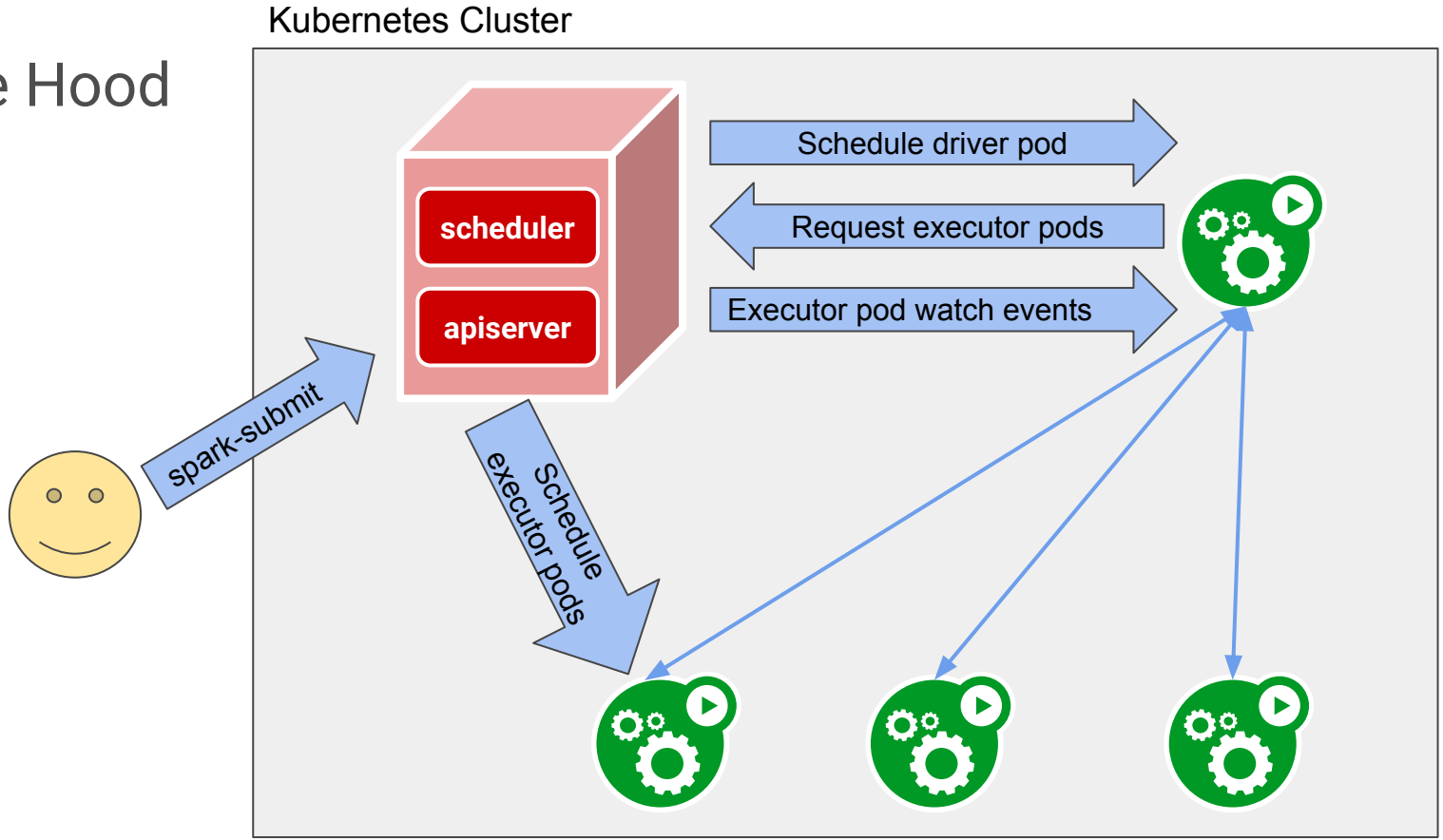
The Kubernetes Scheduler Backend



- Handles task scheduling
- Manages executor pods
- Watches events on executor pods
- Communicates with K8S API server

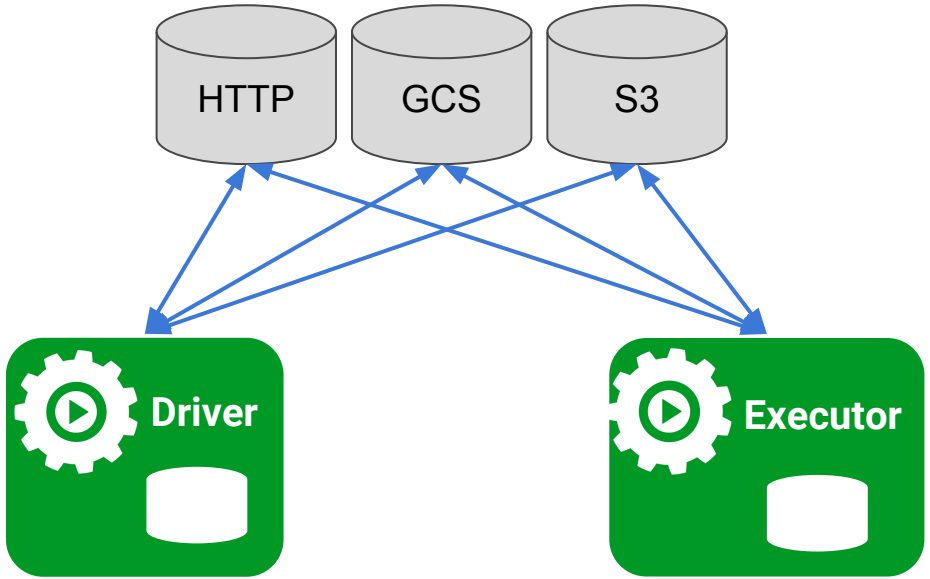
- Schedule and runs driver and executor pods
- Provides networking support
- Provides logging support

Under the Hood



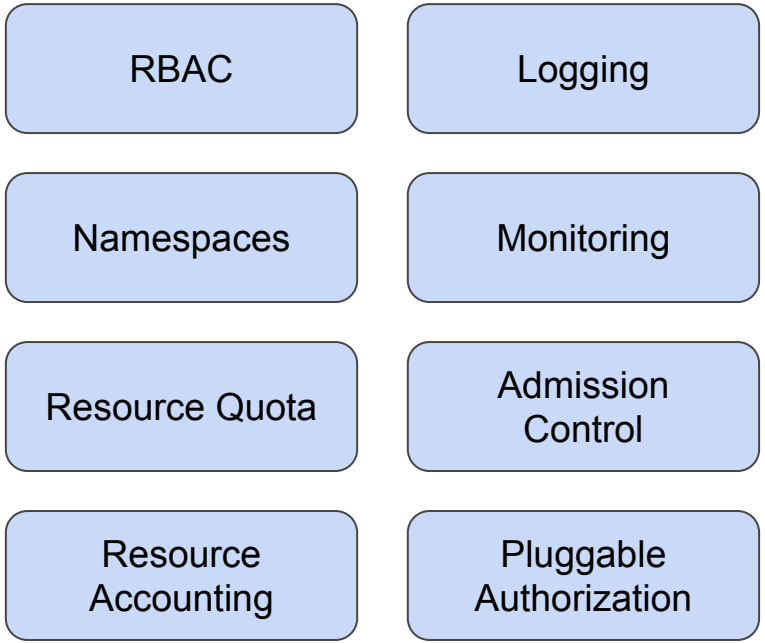
Dependency Management

- Uses container local dependencies through **custom** container images
- Uploads dependencies to remote storage, e.g., HTTP servers, Cloud Storage, S3, etc.



Administration

- Launch Spark jobs as a particular user into a specific namespace
- RBAC and namespace-level isolation and resource quotas
- Audit logging for clusters
- Several monitoring solutions to see node, cluster and pod-level statistics

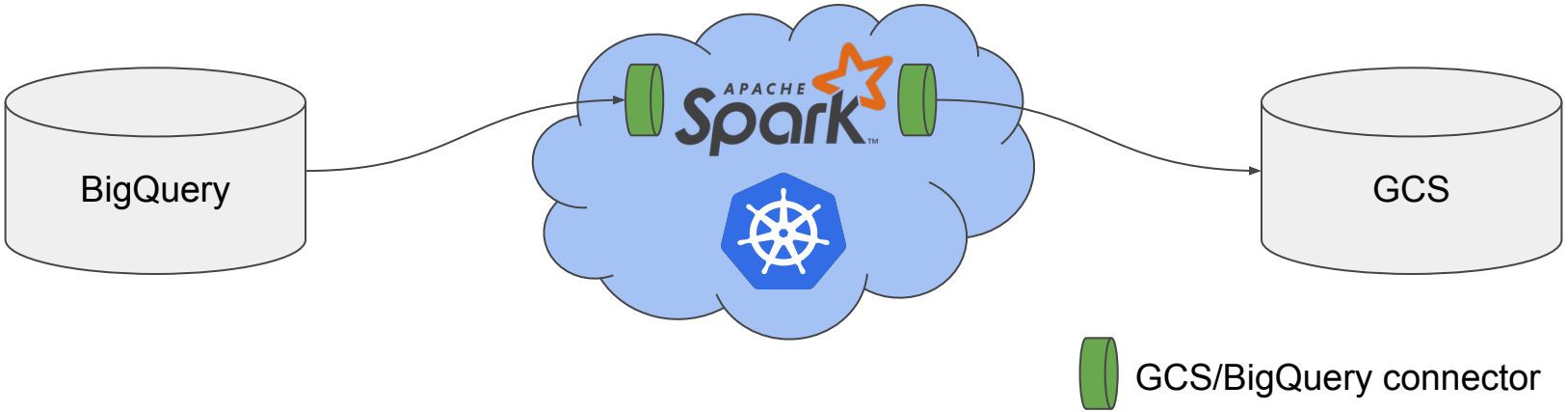


Features Upstreamed in Spark 2.3

- Basic Kubernetes submission client + scheduler backend
- Java/Scala support
- Static resource allocation
- Support for container-local and remote dependencies
- Support for driver and executor pod customization
  - Specifies hard CPU and memory limits
  - Sets environment variables
  - Sets Kubernetes labels and annotations
  - Mounts Kubernetes secrets
  - Specifies node selectors
- Support for RBAC and service account

Demo

- BigQuery word count (github)
  - Reads input data from a BigQuery table and writes output to a Cloud Storage bucket
  - Uses a custom container image with the GCS/BigQuery connector installed and configured



Future Work

- Client mode support
- Dynamic resource allocation + external shuffle service
- Resource staging server (RSS) for local application resources
- Python and R support
- (Kerberized) HDFS support
- Kubernetes CustomResourceDefinition (CRD) and Spark Operator
- Priorities and preemption for pods
- Batch scheduling

Community

- Organizations involved alphabetically
  - Bloomberg
  - Google
  - Haiwen
  - Hyperpilot
  - Intel
  - Palantir
  - Pepperdata
  - Redhat

Documentation:
github.com/apache/spark/blob/master/docs/running-on-kubernetes.md

Slack:
https://kubernetes.slack.com/messages/sig-big-data

Fork (branch-2.2-kubernetes):
https://github.com/apache-spark-on-k8s/spark

Spark Operator
github.com/GoogleCloudPlatform/spark-on-k8s-operator

Q & A

Github: https://github.com/liyinan926

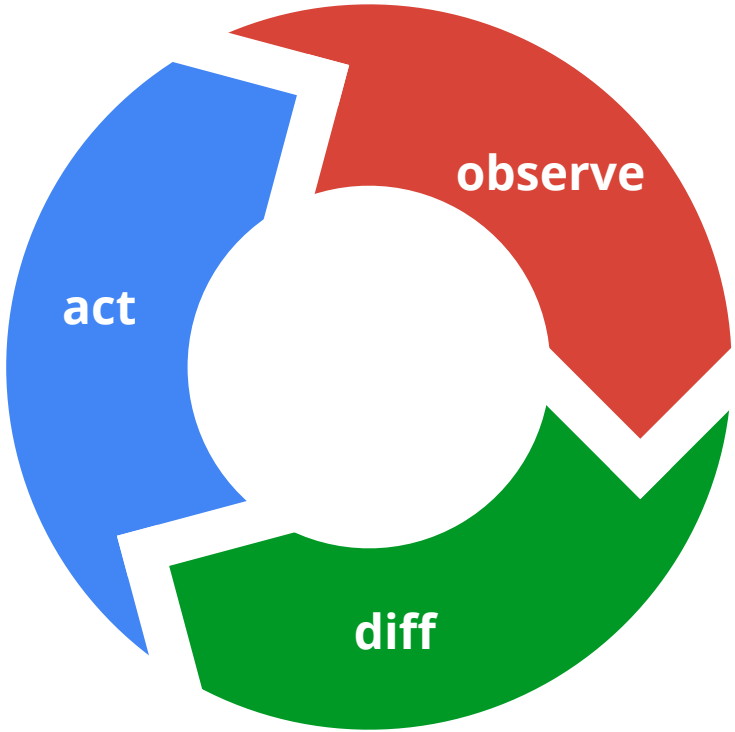
LinkedIn: https://www.linkedin.com/in/yinan-li-91a3b214

Twitter: @liyinan926

Backup Slides - Declarative desired state model

- Controllers continuously drive

  observed state → desired state
- Open World -- anything can happen
- Status is reported asynchronously
- All state is visible via the API
- `Watch` API to distribute state changes



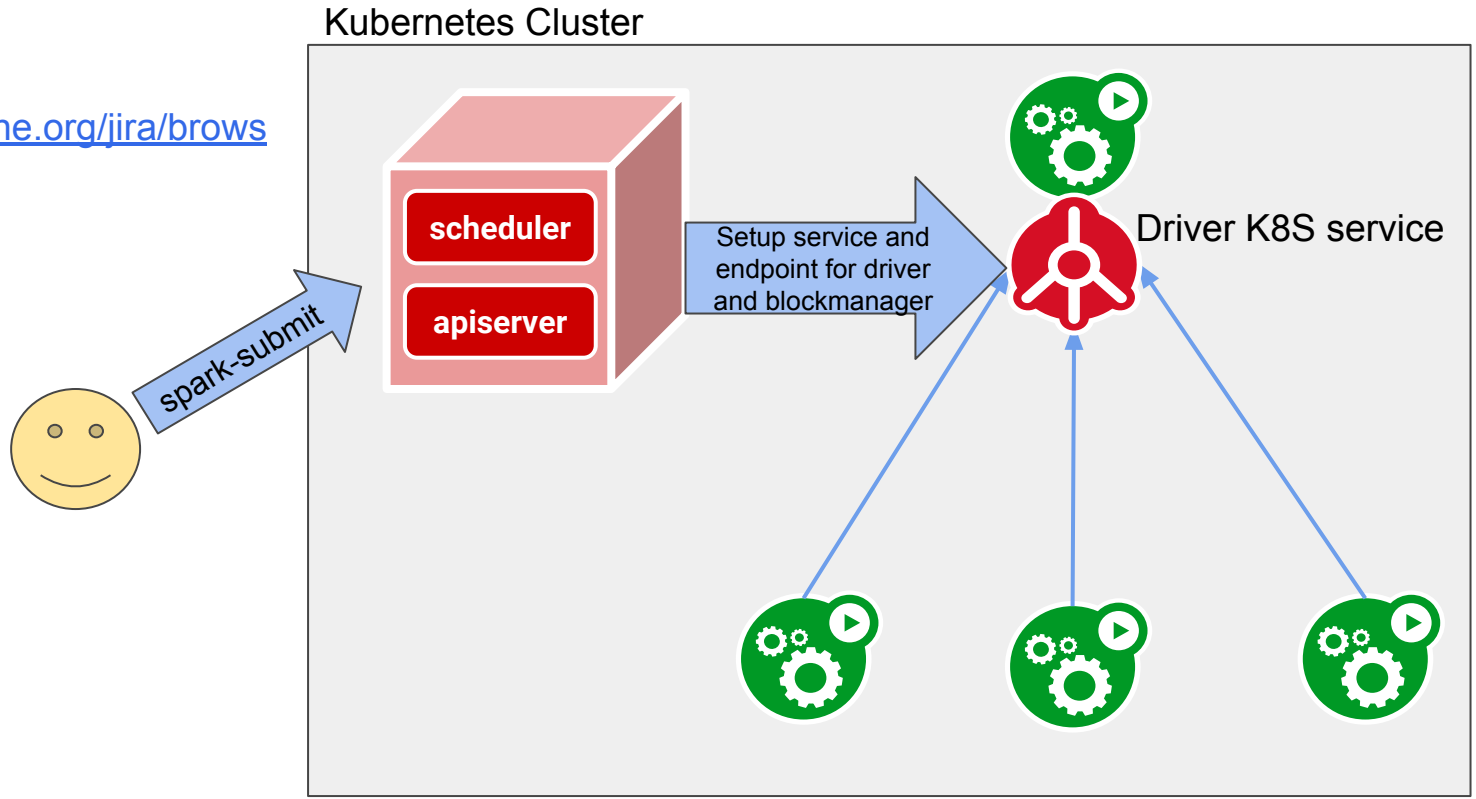
Backup Slides - Handling Spark Configuration

- Spark configuration properties are passed to driver through environment variables in the driver container
  - E.g., `SPARK_JAVA_OPTS_1=”-Dkey=value”`
- When the driver is started, the properties (in the form above) are collected from the environment variables and added onto the driver classpath

```
env | grep SPARK_JAVA_OPT_ | sed 's/[^=]*=\(.*\)/\1/g' > /tmp/java_opts.txt
readarray -t SPARK_JAVA_OPTS < /tmp/java_opts.txt
${JAVA_HOME}/bin/java "${SPARK_JAVA_OPTS[@]}"
```

Backup Slides - Executors Connecting to Driver

Due to https://issues.apache.org/jira/browse/SPARK-21642



Backup Slides - Handling Remote Dependencies