Apache Spark on Kubernetes: A Technical Deep Dive

Yinan Li @ Google
Github: @liyinan926
Agenda

• Why Native Integration with Kubernetes?
• Brief Project History
• Technical Deep Dive
• Existing Features
• Roadmap
• Q & A
A Bit About Me

• Software Engineer on the GKE Workloads team at Google
• Co-chair of Kubernetes SIG Big Data
• Apache Spark Committer
  • One of the main contributors to Spark on Kubernetes
Why Spark on Kubernetes?

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

- Currently the most popular framework for large-scale data processing.
- Widely used by some of the world’s largest companies.
- Supports various language bindings: Scala/Java, Python, R, SQL.
- Supports batch, streaming, graph processing, ML, and SQL workloads - the so-called unified analytics engine.
- 19K+ stars and 17K+ forks on Github
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
Oct 2018: more features were added into Spark 2.4
Kubernetes Scheduler Backend

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

Spark Core

- Configuration
- Request executors
- Kill executors

Kubernetes Scheduler Backend

- Manages executor pods
- Schedule and runs driver and executor pods
- Provides networking support
- Provides logging support
Under the Hood

Kubernetes Cluster

- **scheduler**
- **apiserver**

- Schedule driver pod
- Request executor pods
- Executor pod watch events

spark-submit

Schedule executor pods

executor pods
Under the Hood

Driver

Executor

Execut or Pod Spec

Steps

Execut or Pod Spec

scheduler

apiserver

Local dir (emptyDir)
Executors Connecting to Driver

- Executors connect to the driver using its FQDN
- Submission client creates a headless service for the driver
Dependency Management

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

- In Spark 2.3.x
  - An **init-container** is used for downloading remote dependencies
  - Files get downloaded to a user-specified location.
- In Spark 2.4.x
  - Remote dependencies are downloaded using a **Spark-native** way.
    - Driver container runs spark-submit in client mode.
  - Files get downloaded to the local dir and are resolved using SparkFiles.get().
Existing Features

- Kubernetes submission client + scheduler backend
- Languages supported: Java, Scala, Python, R
- Static resource allocation
- Support for container-local and remote dependencies
- Support for driver and executor pod customization
  - Specifies hard CPU and memory limits
  - Sets labels, annotations, and environment variables
  - Mounts Kubernetes secrets
  - Volumes: hostPath, emptyDir, PVCs
  - Image pull secrets
- Cluster mode + limited client mode support
Client Mode (2.4)

- Useful for interactive apps, e.g., notebooks and spark-shell.
- Supports drivers running both inside and outside the cluster.
- **Garbage collection** of executor pods supported for in-cluster.
- Users are responsible for setting up **network connectivity** from executors to the driver.
  - E.g., a headless service for in-cluster.
Kerberos Support (3.0)

• Necessary for secure HDFS access.
• Needs both a Delegation Token (DT) and Hadoop configuration
• Does not yet support delegation token renewal.
Kubernetes Operator for Spark

- Kubernetes CRD + custom controller
- Supports extensive pod customization through a mutating admission webhook
- Native Cron support for running scheduled applications
- Automatic application restart with a configurable restart policy
- Supports exporting application-level metrics and driver/executor metrics to Prometheus
- Supports installation with Helm
- Comes with a command-line tool `sparkctl`

---

```
apiVersion: "sparkoperator.k8s.io/v1alpha1"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: default
spec:
  type: Scala
  mode: cluster
  image: "gcr.io/spark-operator/spark:v2.4.0"
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "..."
  driver:
    memory: "512m"
    serviceAccount: spark
  executor:
    instances: 1
    memory: "512m"
  monitoring:
    exposeDriverMetrics: true
    exposeExecutorMetrics: true
    prometheus:
      port: 8090
  restartPolicy: Never
```
Roadmap (3.0 and Beyond)

- Support for using a **pod template** to customize the driver and executor pods.
  - No more new configuration properties
- Dynamic resource allocation and external shuffle service.
  - New shuffle service work in progress
- Better support for local application dependencies on client machines.
- Driver resilience for Spark Streaming applications.
- Better scheduling support.
Getting Involved

- [github.com/apache/spark](https://github.com/apache/spark): code under resource-managers/kubernetes
- Documentation: [http://spark.apache.org/docs/latest/running-on-kubernetes.html](http://spark.apache.org/docs/latest/running-on-kubernetes.html)
- Spark user & dev mailing lists
- Jira (use Kubernetes for Component)
- Slack sig-big-data: [https://kubernetes.slack.com/messages/sig-big-data](https://kubernetes.slack.com/messages/sig-big-data)
Questions?