Spark and Zeppelin on Kubernetes

Toader Sebastian / Magyari Sándor

BANZAICLOUD

Spark and Zeppelin on Kubernetes

Agenda

- Apache Spark
- Spark on Kubernetes
- Zeppelin
- Zeppelin on Kubernetes
- Demo
- QA

Spark

- Apache Spark is a fast and general-purpose cluster computing system
- Spark is becoming the enterprise standard for data processing and streaming
- It can run in standalone mode
- It can run using several external cluster manager
  - Mesos
  - Yarn
  - Kubernetes

Spark on Kubernetes

- Mesos/YARN
  - Were not architected for cloud environment
  - Workload islands
- Containerisation enables improved scalability and reduced costs
- Kubernetes evolved fast and become the enterprise standard orchestration framework designed primarily for the cloud.
  - Container runtime orchestration
  - Abstracts away the infrastructure containers run
  - Service discovery and exposure
  - Scale and load balance containers
  - Encourages microservice oriented approach
- Cloud providers support running Kubernetes (AKS, GKE, EKS)

BANZAICLOUD

Spark on Kubernetes

- Spark Standalone on Kubernetes:
  - https://github.com/kubernetes/examples/tree/master/staging/spark
  - Requires ZK for Master Node HA
  - Supports simple FIFO scheduler across applications
  - Spark unaware of the size of the Kubernetes cluster it runs on
    - Dynamic allocation limited to the size of the Spark cluster
  - Resizing system resources allocated for the cluster requires redeployment of Spark cluster

Spark on Kubernetes

- Spark 2.2 on Kubernetes fork:
  - https://github.com/apache-spark-on-k8s/spark
  - Based on Spark 2.2
  - Uses Kubernetes as cluster manager to request resources

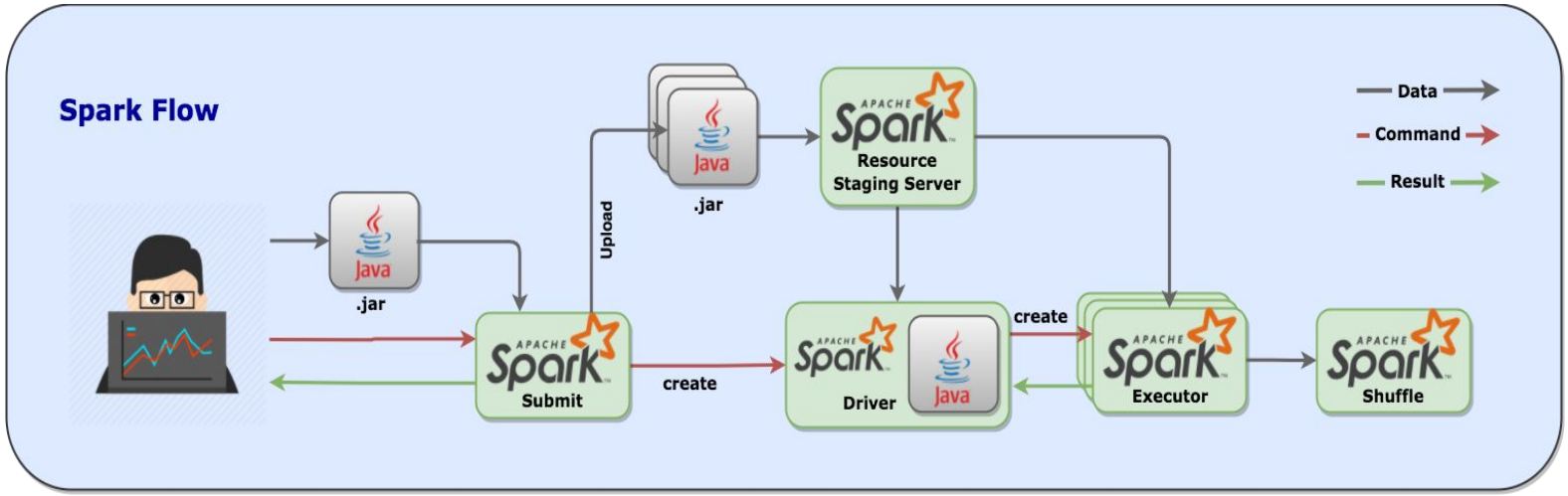
BANZAICLOUD

Spark on Kubernetes: Deep dive

- spark-submit
  - ‘client‘ mode not supported

```
/bin/spark-submit \
  --class <main-class> \
  --master k8s://https://<k8s-apiserver-host>:<k8s-apiserver-port> \
  --deploy-mode cluster \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

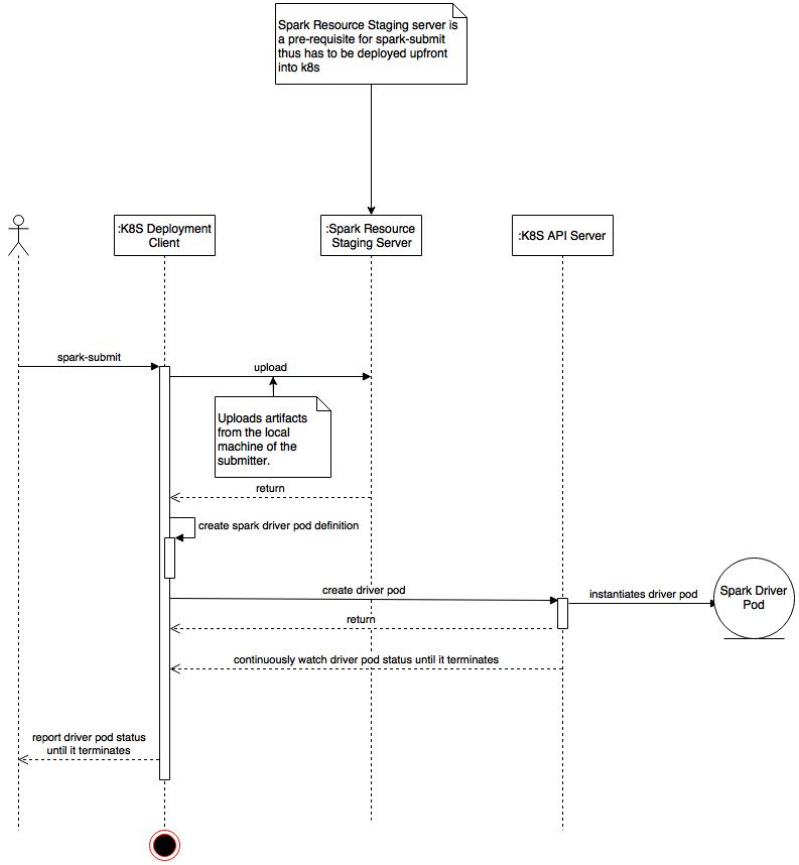
Spark on Kubernetes: Deep dive



Spark on Kubernetes: Deep dive

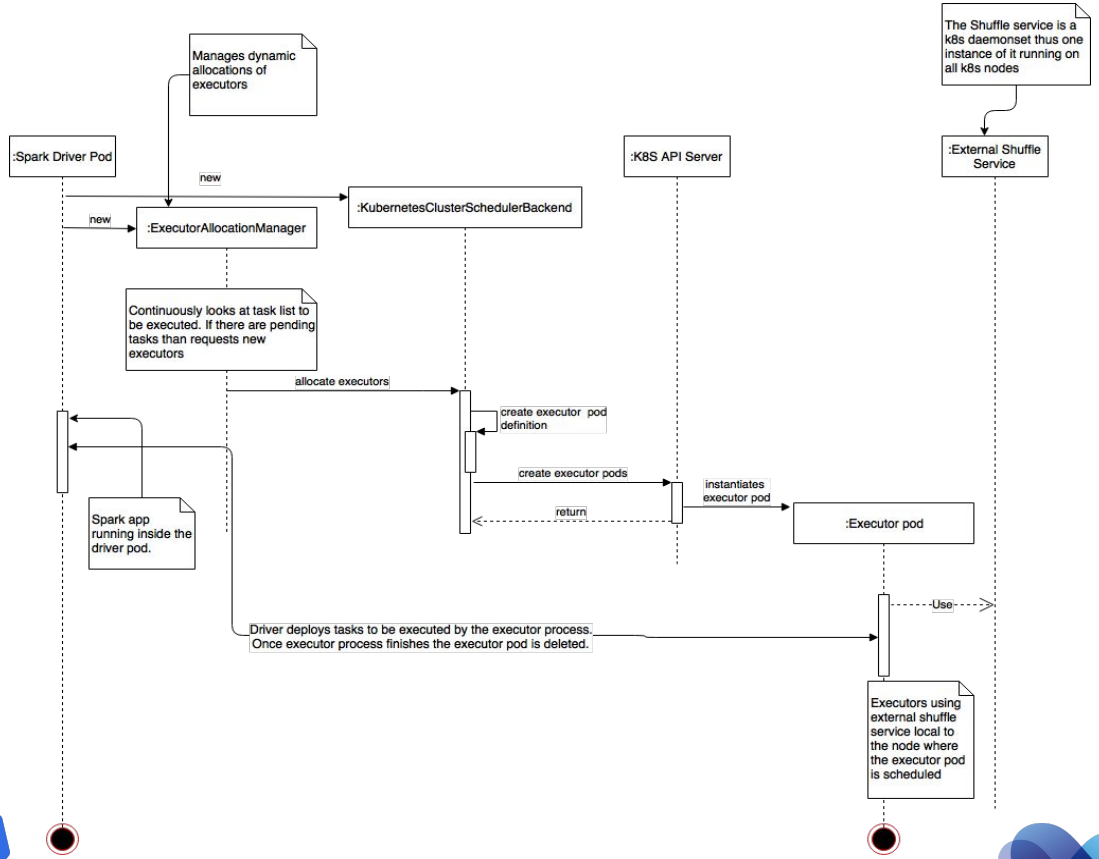
- **Spark Driver** - this is the component where the execution of Spark application starts. It's responsible for creating actionable tasks from the Spark application it executes; manages and coordinates the executors.
- **Executor** - the component responsible for executing tasks
- **External Shuffle Service** - used only when dynamic executor scaling is enabled. The external shuffle service is responsible for persisting shuffle files beyond the lifetime of the executors, allowing the number of executors to scale up and down without losing computation.
- **Resource Staging Server(RSS)** - only used when the compiled code of the Spark application is hosted locally on the machine where the 'spark-submit' is issued. Also used when dependencies of the Spark application are located on the local machine. These locally hosted files are made available to Spark Driver and Executor inside K8S via this component.

Spark on Kubernetes: Deep dive



BANZAICLOUD

Spark on Kubernetes: Deep dive



BANZAICLOUD

Spark on Kubernetes: Deep dive

- The scheduler keeps track of the **expected to total number of executors.**
- In the same time the scheduler keeps track the **number of executor pods** that it **requested** already from Kubernetes.
- Tracks number of executors that actually **registered** with the driver.
- The scheduler runs with a periodicity of spark.kubernetes.allocation.batch.delay (if not specified defaults to 1 second).

Spark on Kubernetes: Deep dive

```
    if total_registered_executors < total_requested_executors {
        // there are still pending executor pods to be created by Kubernetes
        // , don't request new executor pods
    } else if total_expected_executors <= total_registered_executors {
        // maximum allowed executor limit reached. Not scaling up further.
    } else {
        int to_be_requested_num = min(total_expected_executors - total_registered_executors, allocation_batch_size)

        for (i = 0; i < to_be_requested_num; i++) {
            request_new_executor_from_kubernetes()
            total_requested_executors++
        }
    }
```

BANZAICLOUD

Spark on Kubernetes: Monitoring

- Prometheus is an open source monitoring and alerting system
- Part of the official **Cloud Native Computing Foundation** project so every Kubernetes related component is using/will use Prometheus for monitoring and alerting
- Spark has no out of the box support for Prometheus
- We added support for publishing metrics to Prometheus
  - PR: https://github.com/apache/spark/pull/19775
- Externalized Prometheus sink: https://github.com/banzaicloud/spark-metrics
  - Future work: expose it as Spark package

Spark on Kubernetes: Resiliency

- Current Spark driver implementation is not resilient to node failures
- We added resiliency by using Kubernetes Jobs
  - We'll submit a PR soon
- Future work
  - Make resource staging server resiliency to node failure

BANZAICLOUD

Spark 2.3 on Kubernetes

- Officially running Spark workloads natively on Kubernetes was introduced in Apache Spark 2.3
  - Only a subset of the features from the spark-on-k8s fork were migrated into Apache Spark 2.3
  - There is no resource staging server
  - No support for dynamic allocation

BANZAICLOUD

Zeppelin

- Apache Zeppelin is a web-based notebook that enables data-driven interactive data analytics, supporting many different languages and backends, a convenient way to play with your Spark code, visualise your data
- The concept of interpreter allows any language/data-processing-backend to be plugged into Zeppelin, supporting 20+ different interpreters: Spark, BigQuery, Cassandra, HBase ...
- Flexible: you can switch between languages within a single notebook, passing variable values between the languages
- Built-in integration for Apache Spark, providing following Spark interpreters:
  - Scala
  - Python
  - R
  - SQL

BANZAICLOUD

Zeppelin running Spark on Kubernetes

- Zeppelin seems to be an easy candidate for integration with K8S, as it already has the capability to execute interpreters remotely, running outside of Zeppelin VM
- In case of Spark, Zeppelin can:
  - connect to an already running interpreter process
  - launch a **RemoteInterpreterServer** running inside Spark Driver using **spark-submit**

BANZAICLOUD

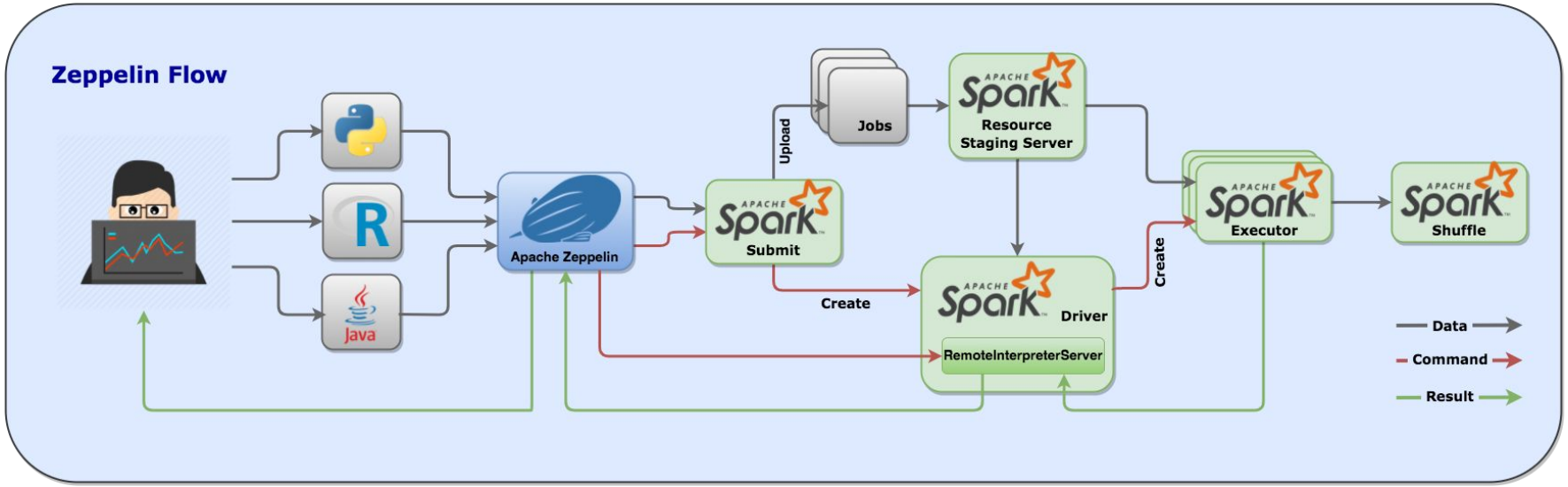
Zeppelin running Spark on Kubernetes

Couple of changes required:

- Direct communication with K8S API, allowing service creation, watching for driver pod events
- Connecting to **RemoteInterpreterServer** through a K8S service, instead of directly connecting using a callback mechanism, for better resiliency
- Uploading **RemoteInterpreterServer** and its dependencies to RSS

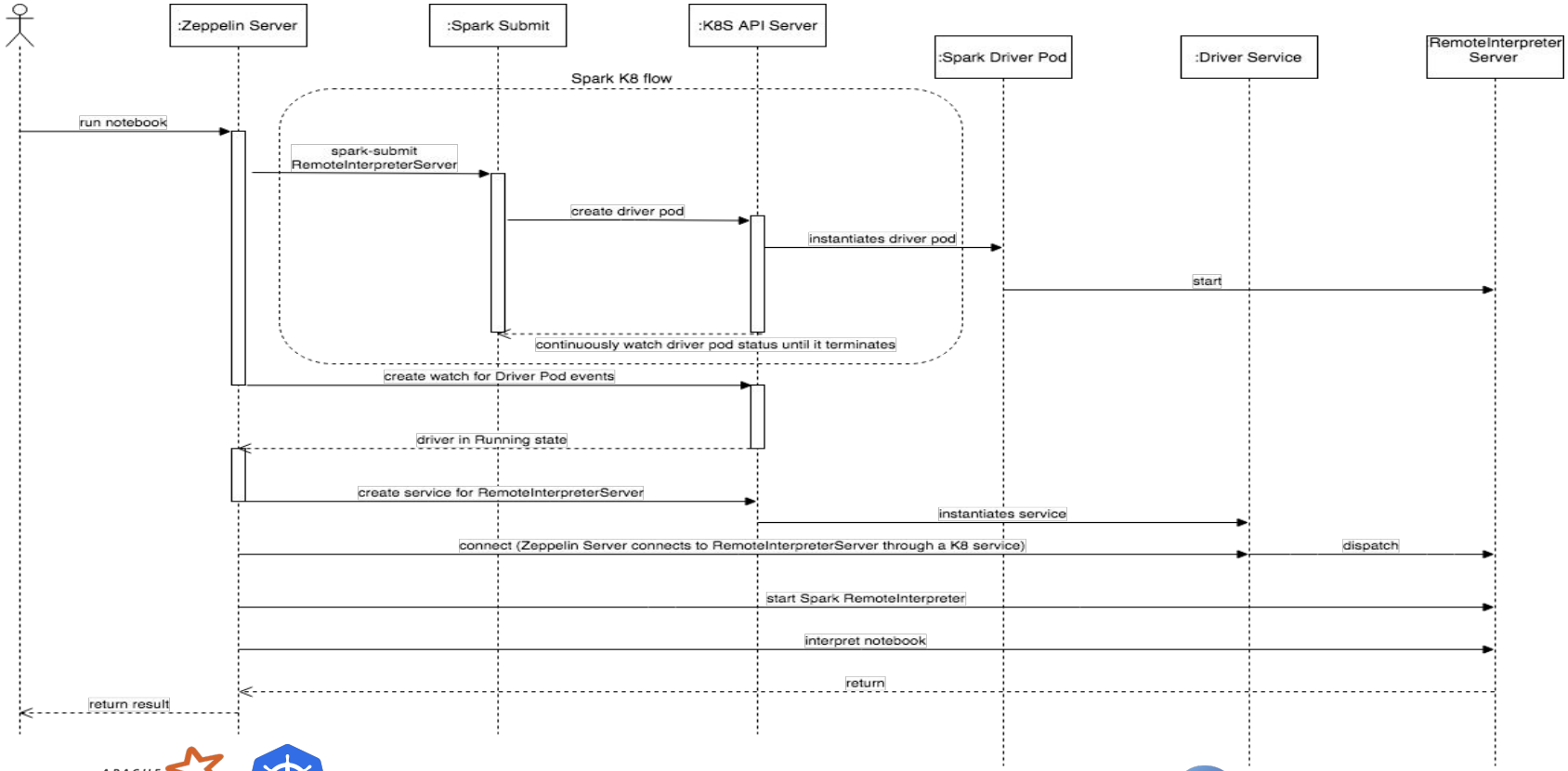
Related PR https://github.com/apache/zeppelin/pull/2637

Zeppelin on Kubernetes



BANZAICLOUD

Zeppelin on Kubernetes



BANZAICLOUD

Zeppelin running Spark on Kubernetes

- The flow is nearly identical with standard Spark on K8S flow, just that Zeppelin launches **RemoteInterpreterServer** with spark-submit
- Watches for Driver pod to be created
- Creates a K8S service
- Connects to **RemoteInterpreterServer**
- Starts all **Spark** interpreters

BANZAICLOUD

Pipeline

A Next Generation **Heroku** / **Cloud Foundry** like PaaS on steroids.

A full managed runtime for enterprise software.

Pipeline

https://github.com/banzaicloud/pipeline


Do you have any
questions?