Cloudera Streaming Analytics - Kubernetes Operator 1.2.0

Getting Started with Flink

Date published: 2024-06-15
Date modified: 2024-02-28

https://docs.cloudera.com/

Legal Notice

Contents

Deploying a Flink application

This Getting Started guide walks you through how to deploy a Flink application on a Kubernetes cluster using the Cloudera Streaming Analytics - Kubernetes Operator.

What is Apache Flink?

Flink is a distributed processing engine and a scalable data analytics framework. You can use Flink to process data streams at a large scale and to deliver real-time analytical insights about your processed data with your streaming application.

Flink is designed to run in all common cluster environments, perform computations at in-memory speed and at any scale. Furthermore, Flink provides communication, fault tolerance, and data distribution for distributed computations over data streams. A large variety of enterprises choose Flink as a stream processing platform due to its ability to handle scale, stateful stream processing, and event time.

About this task

This Getting Started guide walks you through how to deploy a Flink application on a Kubernetes cluster using the Cloudera Streaming Analytics - Kubernetes Operator. The Getting Started guide uses the Flink Kubernetes Tutorial from the *Flink tutorials public repository*.

Note: The following steps require internet connectivity. In an air-gapped environment additional steps are required to build the repository. For more information, see the *Installing Cloudera Streaming Analytics - Kubernetes Operator in an air-gapped environment*.

Procedure

1. Clone the Flink Tutorials repository, and build the tutorial Docker image that contains the built Flink JAR file.

```
git clone https://github.com/cloudera/flink-tutorials.git -b CSA-OPERATO
R-1.2.0
cd flink-tutorials/flink-kubernetes-tutorial
mvn clean package
docker build -t flink-kubernetes-tutorial .
```

2. Tag the tutorial image to attach the required information about the Docker registry that you will use in the next step.

```
docker image tag flink-kubernetes-tutorial [***REGISTRY
 HOST***]:[***PORT***]/[***PROJECT***]/flink-kubernetes-tutorial:latest
```

3. Push the newly tagged tutorial image to your Docker registry. This way the Kubernetes nodes can download the image from the (local or public) Docker registry.

```
docker push [***REGISTRY HOST***]:[***PORT***]/[***PROJECT***]/flink-kuber
netes-tutorial:latest
```

   You will see the Image successfully pushed message when the image is correctly pushed.

4. Create the `FlinkDeployment` configuration file by saving the following example as flink-deployment.yaml. Make sure to set spec.image to the image that you have pushed in the previous step.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: flink-kubernetes-tutorial
spec:
```

```
  image:  [***REGISTRY HOST***]:[***PORT***]/[***PROJECT***]/flink-kuber
netes-tutorial:latest
  flinkVersion: v1_19
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "4"
  serviceAccount: flink
  mode: native
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    args: ["--rowsPerSec", "10"]
    jarURI: local:///opt/flink/usrlib/flink-kubernetes-tutorial.jar
    parallelism: 4
    state: running
    upgradeMode: stateless
```

5. Run the Flink job by applying the `FlinkDeployment` configuration file to the cluster using the following command:

   ```
   kubectl -n flink apply -f flink-deployment.yaml
   ```

   The Flink Operator automatically recognizes the new resource and starts the execution of the Flink job, which is reflected by JOB STATUS switching to RUNNING:

   ```
   kubectl -n flink get FlinkDeployment
   NAME                        JOB STATUS   LIFECYCLE STATE
   flink-kubernetes-tutorial   RUNNING      STABLE
   ```

6. Configure access to the Flink Dashboard at http://localhost:8081 (or IP address/domain name of your machine, if it differs from localhost) using port-forwarding with the following command:

   ```
   kubectl -n flink port-forward service/flink-kubernetes-tutorial-rest 808
   1:8081
   ```

   All requests to the http://localhost:8081 URL will be forwarded to the Flink service while the kubectl command is running. To stop the port-forwarding, exit the kubectl command by pressing CTRL+C.

Related Information

Flink tutorials public repository

Installing CSA Operator in an air-gapped environment