Task 3: Apache Spark & Kubernetes

In this second part, we cover some basics of how to deploy Apache Spark on top of state-of-the-art cloud orchestration Kubernetes (k8s).

Part 2: Kubernetes

In this second part, we cover some basic concepts of k8s and, as an example, we are going to deploy a small Apache Spark cluster on top of k8s. For the sake of simplicity, the filesystem part will not be addressed and it is left as an open topic for students to contribute through final projects.

Exercise 1: our own k8s cluster

To do this part 2, it is required that students are able to access a k8s cluster. For this, the recommended choice is minikube. Other alternatives include Docker Desktop (which has experimental support for k8s), Rancher Desktop, and kind (k8s in Docker).

Students should check the https://minikube.sigs.k8s.io/docs/start/ to find the most appropriate solution for their setups. For the laboratories, a rootless docker engine will be used (more details in moodle).

Once minikube is installed, we have to check that k8s is properly working. For that, we advise to follow the tutorial at https://kubernetes.io/docs/tutorials/stateless-application/guestbook/.

Exercise 2: defining master and worker images

The first step in our task is defining what our Pods should run, this is, build images for the roles of our Spark cluster. For that purpose, three Dockerfiles are recommended:

1. Base dockerfile: it should download and install Apache Spark and Hadoop in /opt. They can be http://archive.apache.org/dist/spark/spark-${SPARK_VERSION}/spark-${SPARK_VERSION}-bin-hadoop3.tgz and http://archive.apache.org/dist/hadoop/common/hadoop-${HADOOP_VERSION}/hadoop-${HADOOP_VERSION}.tar.gz respectively, where SPARK_VERSION and HADOOP_VERSION should be defined with some reasonable values.
2. Master dockerfile: it should launch the master process.
3. Worker dockerfile: it should launch the worker process connecting to the masters.
4. (Optional) Driver dockerfile: it should launch a PySpark example application. See next section in case you need it. It is helpful to run a test application in case Spark is not able to find the driver host due to NAT or forwarding issues in minikube (or similar environments).

Proposed scripts for both dockerfiles can be found in moodle. Bear in mind that paths or hostnames should be adjusted by the students.

Exercise 3: make everything work

Once the images are built, we are going to use them in two k8s Deployments, one for the master and another one for the workers. Bear in mind that master should be able to talk to any worker using port 7077. Also, in order to use local images, you may need to specify an image pull policy in the container spec section (i.e. imagePullPolicy: IfNotPresent). As first templates, use the files of exercise 1 and build incrementally the desired Deployments.

To expose the cluster to the host, a Service should also be deployed. Create a Service of the proper type (NodePort, LoadBalancer, …) to expose the cluster to port 30077 of the host.

Bear in mind that when using some toy k8s environments such as minikube, even after exposing services with the appropriate Service, they might need an extra step to do some kind of port forwarding.

In this case, a Python script that performs a basic test on Apache Spark. It is important to pay attention to the creation of the SparkSession setting correctly the master and driver.

```
spark = SparkSession.builder.appName(APP_NAME).master(SPARK_URL) \
.config("spark.driver.host", "<The IP of the Host to the exterior>") \
.getOrCreate()
```

If you have problems, you can use another k8s pod to run a test app. In such case, the spark URL should reflect <name of service>.<namespace>.svc.cluster.local and the local port, so that the pod finds the Spark master node. Also, the driver host IP address might not be necessary, since its automatic value should be the internal IP address of the pod.

Exercise 4: why this is useful

Provide instructions for the following:

1. Scale up and down the number of workers. Are the changes automatically detected by the Spark cluster? To check it, you may need to expose port 8080 of the master.
2. Delete the Apache Spark (without deleting minikube).
3. Deploy two separate Spark clusters on the same k8s infrastructure. They must be totally independent. What changes should be done to the YAML files?