The Goal of this project is to demonstrate how to migrate a simple spark application from running on YARN to running on the Ezmeral Container Platform. The application simply ingests CSV into Parquet in overwrite mode, given the original CSV file and the target Parquet file.

To run the application

1. Clone the spark application
   1. git clone <https://github.com/mpojeda84/demo-spark-csv-parquet>

This application simply takes a CSV with headers and saves it as a single Parquet file

It receives two parameters:

1. Source CSV file in the HDFS filesystem
2. Target Parquet file in the HDFS filesystem
3. Compile the spark application
   1. Move into the directory: cd demo-spark-csv-parquet
   2. Package the application: mvn clean package
4. Make sure you have a valid ticket as the mapr user
   1. Create ticket: maprlogin password -user mapr
   2. Enter password
5. Launch the application on YARN
   1. Execute scripts test-cluster.sh

Before executing change the parameters to two valid paths

This will execute the application with YARN as master in the default queue

Install ECP: <https://ps-gitlab.s3-us-west-1.amazonaws.com/hpe-cp-rhel-release-5.2-3020.bin>

Configure K8 Hosts

Create K8 compute cluster

Create K8 datafabric cluster

Test Datafabric cluster

Test Datafabric is accessible from the k8 cluster

Prepare the Spark Environment (<https://docs.containerplatform.hpe.com/52/reference/kubernetes-applications/spark/Preparing_the_Spark_Environment.html>)

1. Create a Tenant in the Kubernetes cluster

Graphical user interface, text, application, email

Description automatically generated

1. Create a User
   1. Credentials: **maikel / mapr**

Graphical user interface, application

Description automatically generated

1. From the Tenant Edit View, assign the user created to the Tenant as an Admin

Graphical user interface, application

Description automatically generated

1. From the User Edit view, you should see the assigned Tenant

Graphical user interface, application

Description automatically generated

1. Configure kubectl and kubectl-hpecp plugin locally (see Ivan’s doc)
2. When prompted, log in as the recently created user (maikel) from the terminal using kubectl
3. Finish this part

Configure Clusters to communicate

1. Copy gen-external-secrets.sh from <https://github.com/mapr/private-kubernetes/tree/5.2-GA/tools> into a datafabric node
2. Bash into the admin-cli pod and container **kubectl exec -it admincli-0 -n mapr-datafabric-cluster --container admincli -- /bin/bash**
3. Sudo su
4. Maprlogin password -user mapr
5. touch /opt/mapr/conf/mapr-pass
6. Write the mapr password in the file and save
7. Copy from /opt/bluedata/common-install/scripts/mapr/gen-external-secrets.sh on any datafabric cluster node
8. Run ./gen-external-secrets.sh
9. Copy the file out of the container into the Kubernetes compute master: **scp hpe-external-secrets.yaml root@10.163.168.79:/root**
10. Make changes to adapt it:
    1. Change services for IPs directly
11. Copy file to kubernetes compute cluster and apply

kubectl apply -f hpe-external-secrets.yaml

secret/mapr-user-secrets created

secret/mapr-server-secrets created

secret/mapr-client-secrets created

configmap/mapr-external-cm created

1. Create Tenant CR, examples in <https://github.com/mapr/private-kubernetes/tree/5.2-GA/examples/ecp52/tenant>
2. Create the ticket manually spark-user-secret
3. Create hdfs path for spark logs
4. Pi is roughly 3.141675708378542

Copy jar into maprfs

Create SparkApplication yaml file

Point yaml to jar

Point yaml to main class name