CS571 Extra Project - Wordcount, PageRank, running on spark, deploying to Kubernetes on GKE

Shoumya Singh
ID- 19566

1. Create a cluster on GKE with

   - gcloud container clusters create spark --num-nodes=1 --machine-type=e2-highmem-2 --region=us-west1

```
NAME   LOCATION  MASTER_VERSION   MASTER_IP       MACHINE_TYPE  NODE_VERSION     NUM_NODES  STATUS
spark  us-west1  1.18.16-gke.502  35.185.198.199  e2-highmem-2  1.18.16-gke.502  3          RUNNING
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

Create image and deploy spark to Kubernetes

2. Install the NFS Server Provisioner

   - helm repo add stable https://charts.helm.sh/stable
   - helm repo update

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
singh19566@cloudshell:~ (cs571-demo-project-302019)$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "stable" chart repository
```

   - helm install nfs stable/nfs-server-provisioner \
   - --set persistence.enabled=true,persistence.size=5Gi

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ helm install nfs stable/nfs-server-provisioner --set persistence.enabled=true,persistence.size=5Gi
WARNING: This chart is deprecated
NAME: nfs
LAST DEPLOYED: Wed Apr 21 18:36:01 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The NFS Provisioner service has now been installed.

A storage class named 'nfs' has now been created
and is available to provision dynamic volumes.

You can use this storageclass by creating a `PersistentVolumeClaim` with the
correct storageClassName attribute. For example:

    ---
    kind: PersistentVolumeClaim
    apiVersion: v1
    metadata:
      name: test-dynamic-volume-claim
    spec:
      storageClassName: "nfs"
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 100Mi
```

3. Create a persistent disk volume and a pod to use NFS spark-pvc.yaml:

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ vi spark-pvc.yaml
singh19566@cloudshell:~ (cs571-demo-project-302019)$ cat spark-pvc.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: spark-data-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 2Gi
  storageClassName: nfs
---
apiVersion: v1
kind: Pod
metadata:
  name: spark-data-pod
spec:
  volumes:
    - name: spark-data-pv
      persistentVolumeClaim:
        claimName: spark-data-pvc
  containers:
    - name: inspector
      image: bitnami/minideb
      command:
        - sleep
        - infinity
      volumeMounts:
        - mountPath: "/data"
          name: spark-data-pv
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

4. Apply the above yaml descriptor

   - kubectl apply -f spark-pvc.yaml

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl apply -f spark-pvc.yaml
persistentvolumeclaim/spark-data-pvc created
pod/spark-data-pod created
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

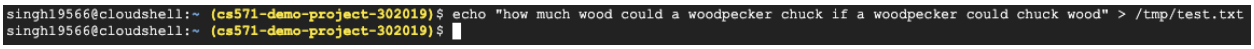
5. Create and prepare your application JAR file

   - docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples* -exec cp {} /tmp/my.jar \;

   After running the above command, you should see this

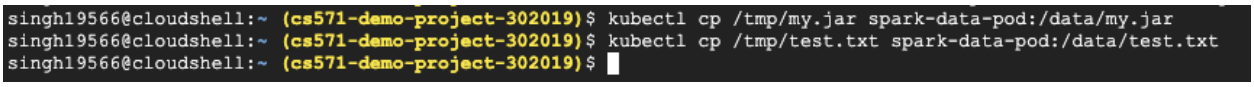
6. Add a test file with a line of words that we will be using later for the word count test

   - echo "how much wood could a woodpecker chuck if a woodpecker could chuck wood" > /tmp/test.txt



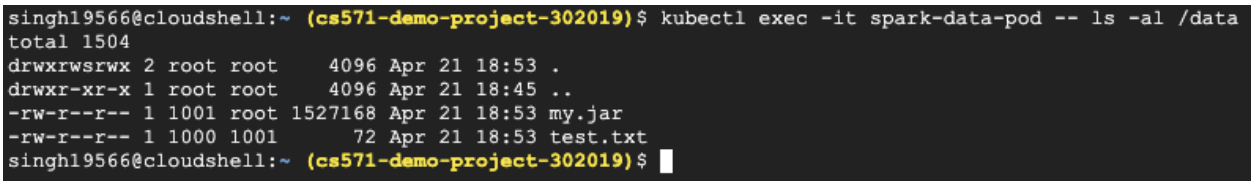
7. Copy the JAR file containing the application, and any other required files, to the PVC using the mount point

   - kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
   - kubectl cp /tmp/test.txt spark-data-pod:/data/test.txt



8. Make sure the files a inside the persistent volume

   - kubectl exec -it spark-data-pod -- ls -al /data



9. Deploy Apache Spark on Kubernetes using the shared volume spark-chart. yaml:

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ vi spark-chart.yaml
singh19566@cloudshell:~ (cs571-demo-project-302019)$ cat spark-chart.yaml
service:
  type: LoadBalancer
worker:
  replicaCount: 3
  extraVolumes:
    - name: spark-data
      persistentVolumeClaim:
        claimName: spark-data-pvc
extraVolumeMounts:
  - name: spark-data
    mountPath: /data
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

10. Check the pods is running:
    - kubectl get pods

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
nfs-nfs-server-provisioner-0      1/1     Running   0          23m
spark-data-pod                    1/1     Running   0          13m
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

11. Deploy Apache Spark on the Kubernetes cluster using the Bitnami Apache Spark Helm chart and supply it with the configuration file above

    - helm repo add bitnami https://charts.bitnami.com/bitnami
    - helm install spark bitnami/spark -f spark-chart.yaml

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

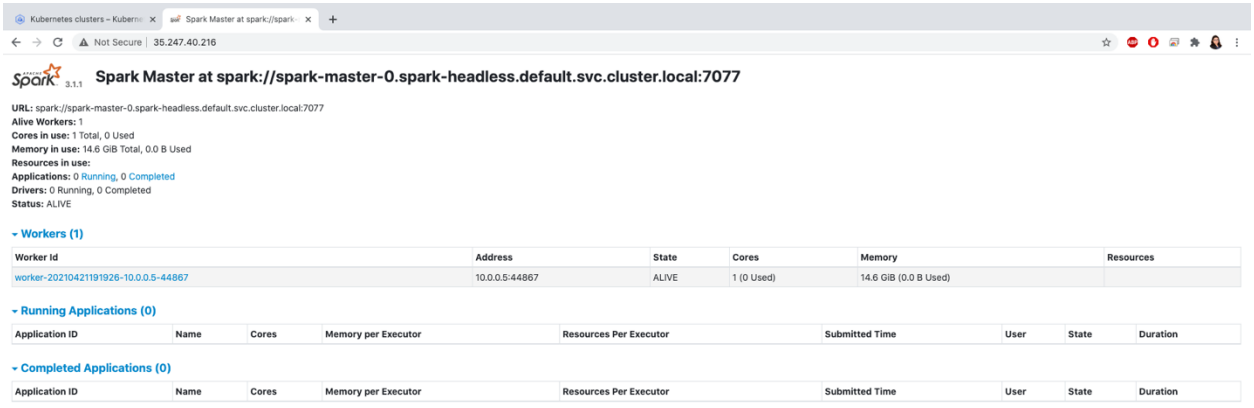
```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Wed Apr 21 19:18:39 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
1. Get the Spark master WebUI URL by running these commands:

    NOTE: It may take a few minutes for the LoadBalancer IP to be available.
    You can watch the status of by running 'kubectl get --namespace default svc -w spark-master-svc'

  export SERVICE_IP=$(kubectl get --namespace default svc spark-master-svc -o jsonpath="{.status.loadBalancer.ingress[0]['ip', 'hostname'] }")
  echo http://$SERVICE_IP:80

2. Submit an application to the cluster:

  To submit an application to the cluster the spark-submit script must be used. That script can be
  obtained at https://github.com/apache/spark/tree/master/bin. Also you can use kubectl run.

  Run the commands below to obtain the master IP and submit your application.

  export EXAMPLE_JAR=$(kubectl exec -ti --namespace default spark-worker-0 -- find examples/jars/ -name 'spark-example*\.jar' | tr -d '\r')
  export SUBMIT_IP=$(kubectl get --namespace default svc spark-master-svc -o jsonpath="{.status.loadBalancer.ingress[0]['ip', 'hostname'] }")

  kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
    --image docker.io/bitnami/spark:3.1.1-debian-10-r42 \
    -- spark-submit --master spark://$SUBMIT_IP:7077 \
    --deploy-mode cluster \
    --class org.apache.spark.examples.SparkPi \
    $EXAMPLE_JAR 1000

** IMPORTANT: When submit an application the --master parameter should be set to the service IP, if not, the application will not resolve the master. **

** Please be patient while the chart is being deployed **
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

12. Get the external IP of the running pod

    - kubectl get svc -l "app.kubernetes.io/instance=spark,app.kubernetes.io/name=spark"

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl get svc -l "app.kubernetes.io/instance=spark,app.kubernetes.io/name=spark"
NAME               TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)                       AGE
spark-headless     ClusterIP      None          <none>          <none>                        56s
spark-master-svc   LoadBalancer   10.3.244.35   35.247.40.216   7077:32060/TCP,80:30975/TCP   56s
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

13. Open the external ip on your browser,

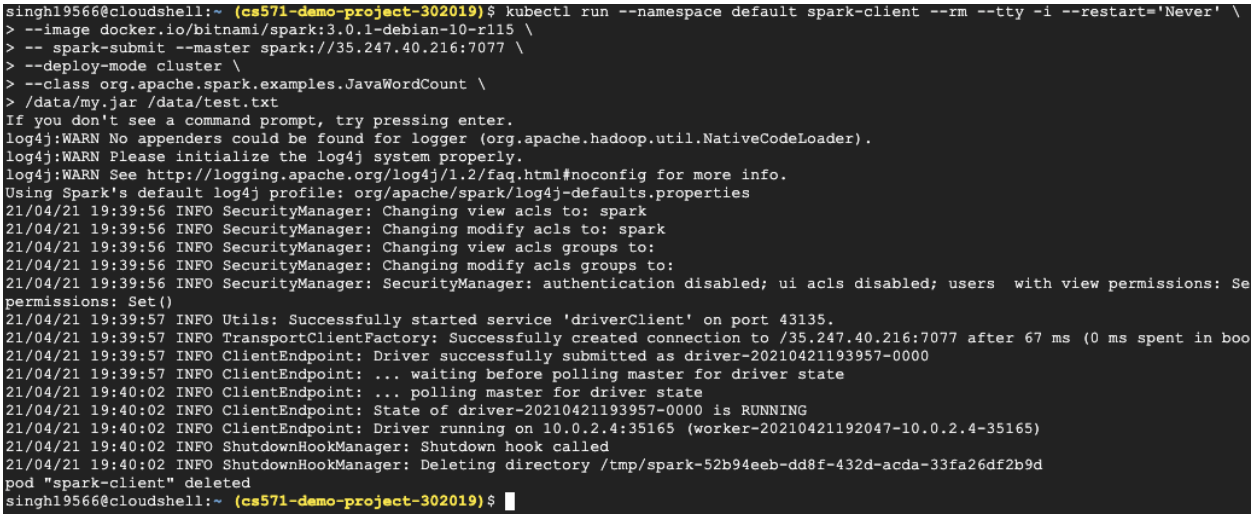


Word Count on Spark

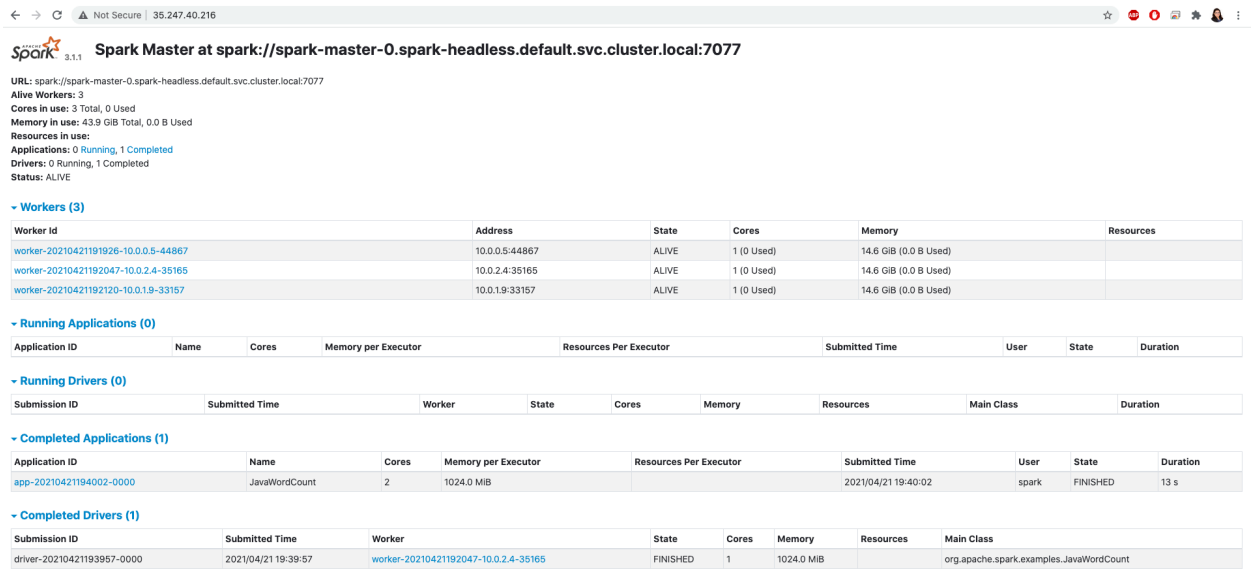
Submit a word count task :

- kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
  --image docker.io/bitnami/spark:3.0.1-debian-10-r115 \
  -- spark-submit --master spark://LOAD-BALANCER-External-ip-ADDRESS:7077 \ --deploy-mode cluster \
  --class org.apache.spark.examples.JavaWordCount \
  /data/my.jar /data/test.txt

You should see something like this after the above command

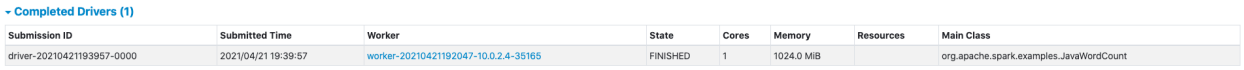


And on your browser, you should see this task finished



View the output of the completed jobs

1. On the browser, you should see the worker node ip address of the finished task



For example, my worker node ip address is 10.0.2.4

2. Get the name of the worker node

   - kubectl get pods -o wide | grep WORKER-NODE-ADDRESS
   - kubectl get pods -o wide | grep 10.0.2.4

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl get pods -o wide | grep 10.0.2.4
spark-worker-1                  1/1     Running   0          24m   10.0.2.4   gke-spark-default-pool-f4a792fd-32rx   <none>
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

3. Execute this pod and see the result of the finished tasks

   - kubectl exec -it <worker node name> -- bash
   - kubectl exec -it spark-worker-1 -- bash

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl exec -it spark-worker-1 -- bash
I have no name!@spark-worker-1:/opt/bitnami/spark$ cd /opt/bitnami/spark/work
```

   - cd /opt/bitnami/spark/work
   - cat <taskname>/stdout

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl exec -it spark-worker-1 -- bash
I have no name!@spark-worker-1:/opt/bitnami/spark$ cd /opt/bitnami/spark/work
I have no name!@spark-worker-1:/opt/bitnami/spark/work$ cat driver-20210421193957-0000 /stdout
cat: driver-20210421193957-0000: Is a directory
cat: /stdout: No such file or directory
I have no name!@spark-worker-1:/opt/bitnami/spark/work$ cat driver-20210421193957-0000/stdout
if: 1
a: 2
how: 1
could: 2
wood: 2
woodpecker: 2
much: 1
chuck: 2
I have no name!@spark-worker-1:/opt/bitnami/spark/work$ exit
exit
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

Running python PageRank onPySpark on the pods

1. Execute the spark master pods
   - kubectl exec -it spark-master-0 -- bash

2. Stark pyspark
   - pyspark

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl exec -it spark-master-0 -- bash
I have no name!@spark-master-0:/opt/bitnami/spark$ pyspark
Python 3.6.13 (default, Apr 19 2021, 18:12:00)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
21/04/21 19:50:51 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.1
      /_/

Using Python version 3.6.13 (default, Apr 19 2021 18:12:00)
Spark context Web UI available at http://spark-master-0.spark-headless.default.svc.cluster.local:4040
Spark context available as 'sc' (master = local[*], app id = local-1619034653797).
SparkSession available as 'spark'.
>>>
```

3. Exit pyspark with
   - exit()

4. Go to the directory where pagerank.py located

   - cd /opt/bitnami/spark/examples/src/main/python

5. Run the page rank using pyspark

   - spark-submit pagerank.py /opt 2

   Note, /opt is an example directory and 2 is the number of iterations you want the page rank to run, we can also change to any numbers, here is my output of running the page rank for directory /opt with 2 iterations