Project: PySpark on Kubernetes: Word Count + PageRank

1. Create a GKE Cluster

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

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ gcloud container clusters create spark --num-nodes=1 --machine-type=e2-highmem-2 --region=us-west1
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, pleas
e pass the `--no-enable-ip-alias` flag
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See https://cloud.google.com/kuberne
tes-engine/docs/how-to/disable-kubelet-readonly-port for ways to check usage and for migration instructions.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster spark in us-west1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/cs570-big-data-424809/zones/us-west1/clusters/spark].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-west1/spark?project=cs570-big-data-424809
kubeconfig entry generated for spark.
NAME: spark
LOCATION: us-west1
MASTER_VERSION: 1.29.4-gke.1043002
MASTER_IP: 34.145.60.220
MACHINE_TYPE: e2-highmem-2
NODE_VERSION: 1.29.4-gke.1043002
NUM_NODES: 3
STATUS: RUNNING
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

2. Install the NFS Server Provisioner

```
helm repo add stable https://charts.helm.sh/stable
helm install nfs stable/nfs-server-provisioner --set persistence.enabled=true,persistence.size=5Gi
```

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ helm install nfs stable/nfs-server-provisioner --set persistence.enabled=true,persistence.size=5Gi
WARNING: This chart is deprecated
NAME: nfs
LAST DEPLOYED: Sat Jun 29 13:23:06 2024
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
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

3. Create Persistent Volume and Pod

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
```

Apply the YAML descriptor:

```
kubectl apply -f spark-pvc.yaml
```

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ kubectl apply -f spark-pvc.yaml
persistentvolumeclaim/spark-data-pvc created
pod/spark-data-pod configured
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

4. 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 \;
```

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples*-exec
cp {} /tmp/my.jar \;
Unable to find image 'bitnami/spark:latest' locally
latest: Pulling from bitnami/spark
2031e0569596: Pull complete
Digest: sha256:5011c72e0f6e09d899715d431b9d8c457a8c456bc197eb5aad53d20ff0dff785
Status: Downloaded newer image for bitnami/spark:latest
spark 01:03:29.61 INFO  ==>
spark 01:03:29.61 INFO  ==> Welcome to the Bitnami spark container
spark 01:03:29.62 INFO  ==> Subscribe to project updates by watching https://github.com/bitnami/containers
spark 01:03:29.62 INFO  ==> Submit issues and feature requests at https://github.com/bitnami/containers/issues
spark 01:03:29.62 INFO  ==> Upgrade to Tanzu Application Catalog for production environments to access custom-configured and pre-packaged software components. Ga
in enhanced features, including Software Bill of Materials (SBOM), CVE scan result reports, and VEX documents. To learn more, visit https://bitnami.com/enterpris
e
spark 01:03:29.62 INFO  ==>

find: paths must precede expression: `cp'
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

5. Add a Test File

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

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ echo "how much wood could a woodpecker chuck if a woodpecker could chuck wood" > /tmp/test.txt
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

6. Copy Files to PVC

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

7. Verify Files Inside the Persistent Volume

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

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ kubectl exec -it spark-data-pod -- ls -al /data
total 1540
drwxrwsrwx 2 root root    4096 Jun 30 01:12 .
drwxr-xr-x 1 root root    4096 Jun 29 22:51 ..
-rw-r--r-- 1 1001 root 1564260 Jun 30 01:11 my.jar
-rw-rw-r-- 1 1000 1000      72 Jun 30 01:12 test.txt
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

8. Deploy Spark on Kubernetes using the shared volume

spark-chart.yaml:

```
service:
  type: LoadBalancer
worker:
  replicaCount: 3
  extraVolumes:
    - name: spark-data
      persistentVolumeClaim:
        claimName: spark-data-pvc
  extraVolumeMounts:
    - name: spark-data
      mountPath: /data
```

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
shagos90499@cloudshell:~ (cs570-big-data-424809)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Sun Jun 30 01:24:40 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: spark
CHART VERSION: 9.2.4
APP VERSION: 3.5.1
```

Deploy Apache Spark:

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

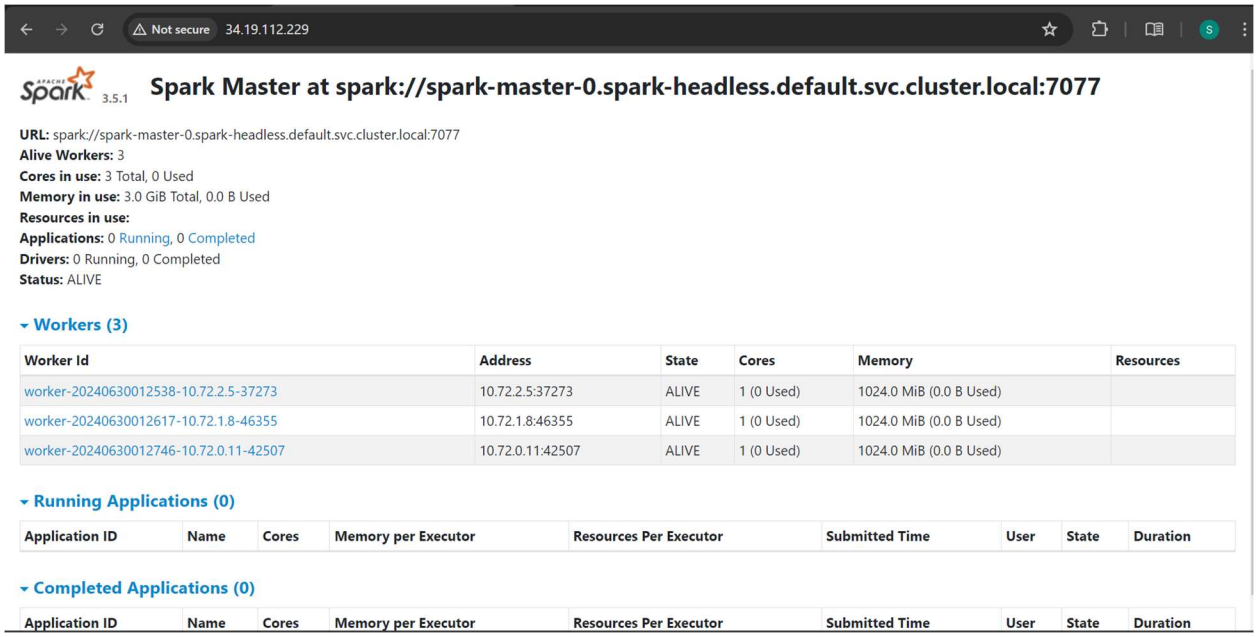
```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
shagos90499@cloudshell:~ (cs570-big-data-424809)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Sun Jun 30 01:24:40 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: spark
CHART VERSION: 9.2.4
APP VERSION: 3.5.1

** Please be patient while the chart is being deployed **

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'] }")
```

9. Get External IP

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

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ 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>                        4m13s
spark-master-svc   LoadBalancer   34.118.235.164   34.19.112.229   7077:32067/TCP,80:30490/TCP   4m13s
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

10. Open the external ip on your browser,



11. Submit 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://34.19.112.229:7077 --deploy-mode cluster --class
org.apache.spark.examples.JavaWordCount /data/my.jar /data/test.txt
```

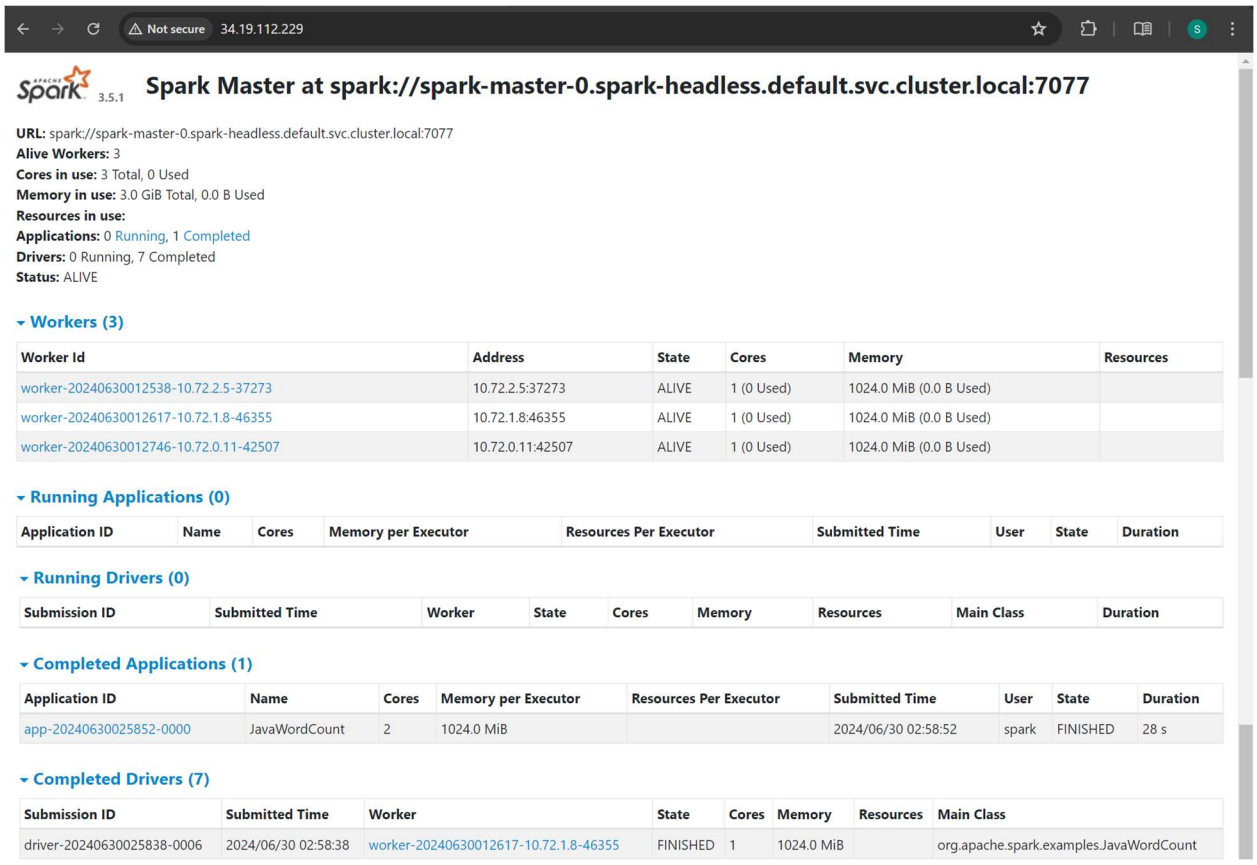
```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ kubectl run --namespace default spark-client --rm --tty -i --restart='Never' --image docker.io/bitnami/spark:3.
0.1-debian-10-r115 -- spark-submit --master spark://LOAD-BALANCER-External-ip-ADDRESS:7077 --deploy-mode cluster --class org.apache.spark.examples.JavaWordCount
/data/my.jar /data/test.txt

If you don't see a command prompt, try pressing enter.

log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.NativeCodeLoader).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
24/06/30 01:53:13 INFO SecurityManager: Changing view acls to: spark
24/06/30 01:53:13 INFO SecurityManager: Changing modify acls to: spark
24/06/30 01:53:13 INFO SecurityManager: Changing view acls groups to:
24/06/30 01:53:13 INFO SecurityManager: Changing modify acls groups to:
24/06/30 01:53:13 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(spark); groups with view pe
rmissions: Set(); users  with modify permissions: Set(spark); groups with modify permissions: Set()
24/06/30 01:53:13 INFO Utils: Successfully started service 'driverClient' on port 34209.
Exception in thread "main" org.apache.spark.SparkException: Exception thrown in awaitResult:
        at org.apache.spark.util.ThreadUtils$.awaitResult(ThreadUtils.scala:302)
        at org.apache.spark.rpc.RpcTimeout.awaitResult(RpcTimeout.scala:75)
        at org.apache.spark.rpc.RpcEnv.setupEndpointRefByURI(RpcEnv.scala:101)
        at org.apache.spark.rpc.RpcEnv.setupEndpointRef(RpcEnv.scala:109)
        at org.apache.spark.deploy.ClientApp.$anonfun$start$2(Client.scala:247)
```

Since this step failed to submit the job, one method to resolve this issue is to submit the job from the master node itself. With the following command:

```
kubectl exec -it spark-master-0 -- spark-submit --master
spark://34.19.112.229:7077 --deploy-mode cluster --class
org.apache.spark.examples.JavaWordCount /data/my.jar /data/test.txt
```

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ kubectl exec -it spark-master-0 -- spark-submit --master spark://34.19.112.229:7077 --deploy-mode cluster --cla
ss org.apache.spark.examples.JavaWordCount /data/my.jar /data/test.txt
24/06/30 02:58:36 INFO SecurityManager: Changing view acls to: spark
24/06/30 02:58:36 INFO SecurityManager: Changing modify acls to: spark
24/06/30 02:58:36 INFO SecurityManager: Changing view acls groups to:
24/06/30 02:58:36 INFO SecurityManager: Changing modify acls groups to:
24/06/30 02:58:36 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: spark; groups with view permissi
ons: EMPTY; users with modify permissions: spark; groups with modify permissions: EMPTY
24/06/30 02:58:36 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/06/30 02:58:38 INFO Utils: Successfully started service 'driverClient' on port 46121.
24/06/30 02:58:38 INFO TransportClientFactory: Successfully created connection to /34.19.112.229:7077 after 183 ms (0 ms spent in bootstraps)
24/06/30 02:58:38 INFO ClientEndpoint: ... waiting before polling master for driver state
24/06/30 02:58:38 INFO ClientEndpoint: Driver successfully submitted as driver-20240630025838-0006
24/06/30 02:58:43 INFO ClientEndpoint: State of driver-20240630025838-0006 is RUNNING
24/06/30 02:58:43 INFO ClientEndpoint: Driver running on 10.72.1.8:46355 (worker-20240630012617-10.72.1.8-46355)
24/06/30 02:58:43 INFO ClientEndpoint: spark-submit not configured to wait for completion, exiting spark-submit JVM.
24/06/30 02:58:43 INFO ShutdownHookManager: Shutdown hook called
24/06/30 02:58:43 INFO ShutdownHookManager: Deleting directory /tmp/spark-63ddb163-7923-4e29-9621-0ceaf8bdcbc1
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

The output on the web should display the application running first and then completed after some time.



11. View Output of Completed Jobs

- Get the worker node IP address from the browser.

Completed Drivers (7)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20240630025838-0006	2024/06/30 02:58:38	worker-20240630012617-10.72.1.8-46355	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordCount

- Find the name of the worker node:

```
kubectl get pods -o wide | grep <WORKER-NODE-ADDRESS>
```

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ kubectl get pods -o wide | grep 10.72.1.8
spark-worker-1                1/1     Running   0          100m   10.72.1.8    gke-spark-default-pool-2be7e14d
```

- Execute the pod and see the result:

```
kubectl exec -it <worker-node-name> -- bash
cd /opt/bitnami/spark/work
cat <taskname>/stdout
```

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ 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-20240630025838-0006/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$
```

12. Running PageRank on PySpark

- Execute the Spark master pod:

```
kubectl exec -it spark-master-0 – bash
```

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ kubectl exec -it spark-master-0 -- bash
I have no name!@spark-master-0:/opt/bitnami/spark$ pyspark
```

- Start PySpark:

```
Pyspark
```

Error:

```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ kubectl exec -it spark-master-0 -- bash
I have no name!@spark-master-0:/opt/bitnami/spark$ pyspark
Error: pyspark does not support any application options.

Usage: ./bin/pyspark [options]

Options:
  --master MASTER_URL         spark://host:port, mesos://host:port, yarn,
                              k8s://https://host:port, or local (Default: local[*]).
  --deploy-mode DEPLOY_MODE   Whether to launch the driver program locally ("client") or
                              on one of the worker machines inside the cluster ("cluster")
                              (Default: client).
```

- Solution: Run the following script from a github repo steps to resolve the issue, where in the pyspark script the name argument passed was the issue.

  Link to repo: https://github.com/bitnami/containers/issues/38139

```
export PYTHONPATH=/opt/bitnami/spark/python/lib/py4j-0.10.9.7-
src.zip:/opt/bitnami/spark/python/:/opt/bitnami/spark/python/:
export PYTHONSTARTUP=/opt/bitnami/spark/python/pyspark/shell.py
exec "${SPARK_HOME}"/bin/spark-submit pyspark-shell-main
exit()
```

- Run PageRank:
  - Go to directory where pagerank.py script is located

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

  - Run command: Notice, /opt is an example directory, you can enter any directory you like, and 2 is the number of iterations you want the pagerank to run, you can also change to any numbers you like

    ```
    spark-submit pagerank.py /opt 2
    ```

    Here is the output with two iterations.