

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

Maryam Taherzadeh ID:19529

## 1. Create a cluster on GKE with

- **gcloud container clusters create spark --num-nodes=1 --machinetype=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  35.233.228.208  e2-highmem-2  1.18.16-gke.502  3          RUNNING
taherzadeh19529@cloudshell:~ (phonic-axle-307118)$ █
```

## 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**

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
```

```
taherzadeh19529@cloudshell:~ (phonic-axle-307118)$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "bitnami" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. *Happy Helming!*
taherzadeh19529@cloudshell:~ (phonic-axle-307118)$ █
```

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ helm install nfs stable/nfs-server-provisioner \
> --set persistence.enabled=true,persistence.size=5Gi
WARNING: This chart is deprecated
NAME: nfs
LAST DEPLOYED: Fri Apr 23 18:58:34 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
```

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ cat spark-pv.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
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ █
```

#### 4. Apply the above yaml descriptor

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl apply -f spark-pvc.yaml
persistentvolumeclaim/spark-data-pvc created
pod/spark-data-pod created
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

#### 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**

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples* -exec cp {} /tmp/my.jar \;
19:31:33.71 Welcome to the Bitnami spark container
19:31:33.71 Subscribe to project updates by watching https://github.com/bitnami/bitnami-docker-spark
19:31:33.72 Submit issues and feature requests at https://github.com/bitnami/bitnami-docker-spark/issues
19:31:33.72
```

#### 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**

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ echo "how much wood could a woodpecker chuck if a woodpecker could chuck wood" > /tmp/test.txt
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl cp /tmp/test.txt spark-data-pod:/data/test.txt
```

## 8. Make sure the files are inside the persistent volume

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl exec -it spark-data-pod -- ls -al /data
total 1504
drwxrwsrwx 2 root root    4096 Apr 23 19:34 .
drwxr-xr-x 1 root root    4096 Apr 23 19:27 ..
-rw-r--r-- 1 1001 root 1527168 Apr 23 19:34 my.jar
-rw-r--r-- 1 1000 1001     72 Apr 23 19:34 test.txt
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

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

```
bash: cd: spark-chart.yaml: Not a directory
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

## 10. Check the pods are running:

- `kubectl get pods`

```
spark master svc  LoadBalancer   10.0.2.22.70  55.127.101.0  79.77.30.69,101.88.50.13,102.21.2.11
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
nfs-nfs-server-provisioner-0  1/1     Running   0          81m
spark-data-pod  1/1     Running   0          52m
```

## 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`

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories

"bitnami" has been added to your repositories
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Fri Apr 23 19:50:46 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'] ")

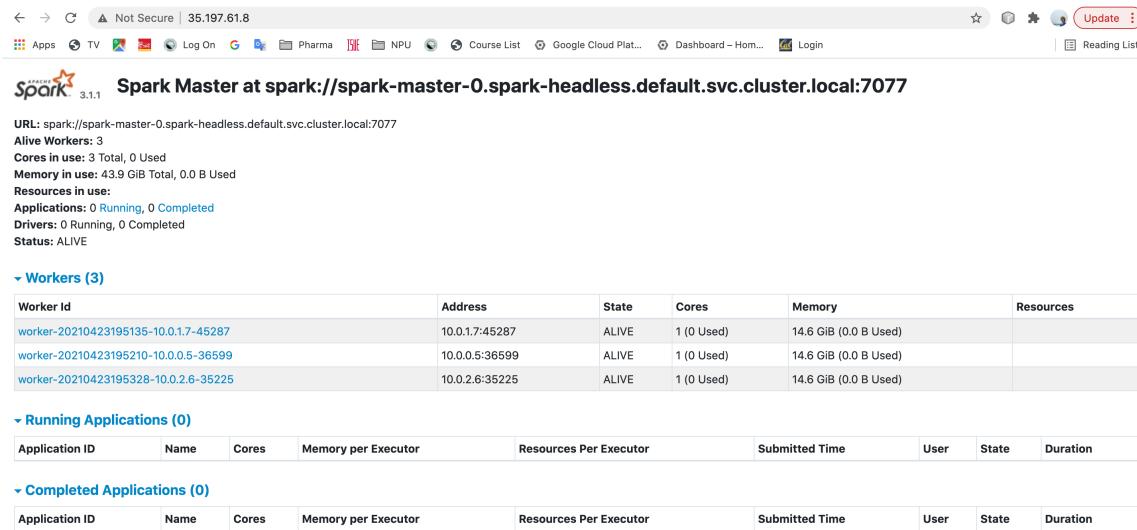
kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
--image docker.io/bitnami/spark:3.1.1-debian-10-r42 \
--spark-submit --master spark://$SUBMIT_IP:7077 \
--deploy-mode cluster
```

## **12. Get the external IP of the running pod**

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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>        5m54s
spark-master-svc LoadBalancer  10.3.252.70  35.197.61.8  7077:30693/TCP,80:30513/TCP  5m54s
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

### **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-ipADDRESS: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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axie-307118)$ 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://35.197.61.8:7077 \
> --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
21/04/23 20:27:01 INFO SecurityManager: Changing view acls to: spark
21/04/23 20:27:01 INFO SecurityManager: Changing modify acls to: spark
21/04/23 20:27:01 INFO SecurityManager: Changing view acls groups to:
21/04/23 20:27:01 INFO SecurityManager: Changing modify acls groups to:
21/04/23 20:27:01 INFO SecurityManager: Changing 'authz' provider to: ui
21/04/23 20:27:01 INFO SecurityManager: ui acls disabled; ui acls disabled; users with view permissions: Set(spark); groups with view permissions: Set(); users with modify permissions: Set(spark); groups with modify permissions: Set()
21/04/23 20:27:02 INFO Utils: Successfully started service 'driverClient' on port 45657.
21/04/23 20:27:02 INFO TransportClientFactory: Successfully created connection to /35.197.61.8:7077 after 76 ms (0 ms spent in bootstraps)
21/04/23 20:27:02 INFO ClientEndpoint: Driver successfully submitted as driver-20210423202702-0000
21/04/23 20:27:02 INFO ClientEndpoint: ... waiting before polling master for driver state
21/04/23 20:27:07 INFO ClientEndpoint: ... polling master for driver state
21/04/23 20:27:07 INFO ClientEndpoint: State of driver-20210423202702-0000 is RUNNING
21/04/23 20:27:07 INFO ClientEndpoint: Driver running on 10.0.1.7:45287 (worker-20210423195135-10.0.1.7-45287)
21/04/23 20:27:07 INFO ShutdownHookManager: Shutdown hook called
21/04/23 20:27:07 INFO ShutdownHookManager: Deleting directory /tmp/spark-5c98909d-8319-42c9-bc39-dc5c9cb98f51
pod "spark-client" deleted
taherzadeh19529@cloudshell:~/wordcount (phonic-axie-307118)$
```

And on your browser, you should see this task finished

URL: spark://spark-master-0.spark-headless.default.svc.cluster.local:7077  
 Alive Workers: 3  
 Cores in use: 3 Total, 0 Used  
 Memory in use: 43.9 GiB Total, 0.0 B Used  
 Resources in use:  
 Applications: 0 Running, 1 Completed  
 Drivers: 0 Running, 1 Completed  
 Status: ALIVE

**Workers (3)**

Worker Id	Address	State	Cores	Memory	Resources
worker-20210423195135-10.0.1.7-45287	10.0.1.7:45287	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	
worker-20210423195210-10.0.0.5-36599	10.0.0.5:36599	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	
worker-20210423195328-10.0.2.6-35225	10.0.2.6:35225	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	

**Running Applications (0)**

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

**Running Drivers (0)**

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class	Duration
---------------	----------------	--------	-------	-------	--------	-----------	------------	----------

**Completed Applications (1)**

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210423202707-0000	JavaWordCount	2	1024.0 MiB		2021/04/23 20:27:07	spark	FINISHED	12 s

**Completed Drivers (1)**

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20210423202702-0000	2021/04/23 20:27:02	worker-20210423195135-10.0.1.7-45287	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordCount

## View the output of the completed jobs

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

Completed Drivers (1)							
Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20210423202702-0000	2021/04/23 20:27:02	worker-20210423195135-10.0.1.7-45287	FINISHED	1	1024.0 MB		org.apache.spark.examples.JavaWordCount

For example, my worker node ip address is 10.0.1.7

### 2. Get the name of the worker node

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl get pods -o wide | grep 10.0.1.7
spark-worker-0          1/1     Running   0        40m    10.0.1.7    gke-spark-default-pool-bcde3d62-wpb  <none>           <none>
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

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

- **kubectl exec -it spark-worker-0 -- bash**

- **cd /opt/bitnami/spark/work**

- **cat <taskname>/stdout**

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl exec -it spark-worker-0 -- bash
I have no name!@spark-worker-0:/opt/bitnami/spark$ cd /opt/bitnami/spark/work
I have no name!@spark-worker-0:/opt/bitnami/spark/work$ cat driver-20210423202702-0000/stdout
if:
at 0
how: 1
could: 2
wood: 2
woodpecker: 2
much: 1
chuck: 2
I have no name!@spark-worker-0:/opt/bitnami/spark/work$
```

## Running python PageRank on PySpark on the pods

## 1. Execute the spark master pods

- **kubectl exec -it spark-master-0 -- bash**

## 2. Stark pyspark

## - pyspark

```
taherzadeh@19529@cloudshell:~/wordcount [phonic-axe-307118]$ 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.
21/04/23 20:51:05 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
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

$$\sqrt{v} \cdot \sum_{i=1}^n \frac{1}{\sqrt{v_i}} \cdot \frac{1}{\sqrt{v_i}}$$

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-1619211067920).
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

```
at org.apache.spark.sql.execution.datasources.DataSource.getOrInferFileFormatSchema(DataSource.scala:167)
at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:418)
at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:326)
at org.apache.spark.sql.DataFrameReader.loadV1Source$anonfun$load$3(DataFrameReader.scala:308)
at org.apache.spark.util.Option$.get(DataFrameReader.scala:111)
at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:308)
at org.apache.spark.sql.DataFrameReader.text(DataFrameReader.scala:945)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
at py4j.Gateway.invoke(Gateway.java:282)
at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
at py4j.commands.CallCommand.execute(CallCommand.java:79)
at py4j.GatewayConnection.run(GatewayConnection.java:238)
at java.lang.Thread.run(Thread.java:748)

21/04/23 20:52:44 INFO SparkContext: Invoking stop() from shutdown hook
21/04/23 20:52:44 INFO SparkUI: Stopped Spark web UI at http://spark-master-0.spark-headless.default.svc.cluster.local:4040
21/04/23 20:52:44 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/04/23 20:52:44 INFO MemoryStore: MemoryStore cleared
21/04/23 20:52:44 INFO BlockManager: BlockManager stopped
21/04/23 20:52:44 INFO BlockManagerMaster: BlockManagerMaster stopped
21/04/23 20:52:44 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/04/23 20:52:44 INFO SparkContext: Successfully stopped SparkContext
21/04/23 20:52:44 INFO ShutdownHookManager: Shutdown hook called
21/04/23 20:52:44 INFO ShutdownHookManager: Deleting directory /tmp/spark-65e26f5e-99fe-40ab-alie-2f753a90b940
21/04/23 20:52:44 INFO ShutdownHookManager: Deleting directory /tmp/spark-65e26f5e-99fe-40ab-alie-2f753a90b940/pyspark-16725e8a-5068-4bd2-af9d-016c1d462f89
21/04/23 20:52:44 INFO ShutdownHookManager: Deleting directory /tmp/spark-df67652d-97af-46a9-99df-30c667da65e3
I have no name!@spark-master-0:/opt/bitnami/spark/examples/src/main/python\$
```