Apache spark on Kubernetes

Ashritha Lopelli - 19764

Introduction

Spark on Kubernetes

When you submit a **Spark** application, you talk directly to **Kubernetes**, the API server, which will schedule the driver pod, so the **Spark** driver container and then the **Spark** driver and the **Kubernetes** Cluster will talk to each other to request and launch **Spark** executors, which will also be scheduled on pods (one pod per executor). If dynamic allocation is enabled the number of Spark executors dynamically evolves based on load, otherwise it’s a static number.

In this project, with the help of PySpark (which is an open-source cluster-computing framework) we want to implement Word Count on Apache Spark running on Kubernetes and Using PySpark to implement PageRank on Apache Spark running on Kubernetes.

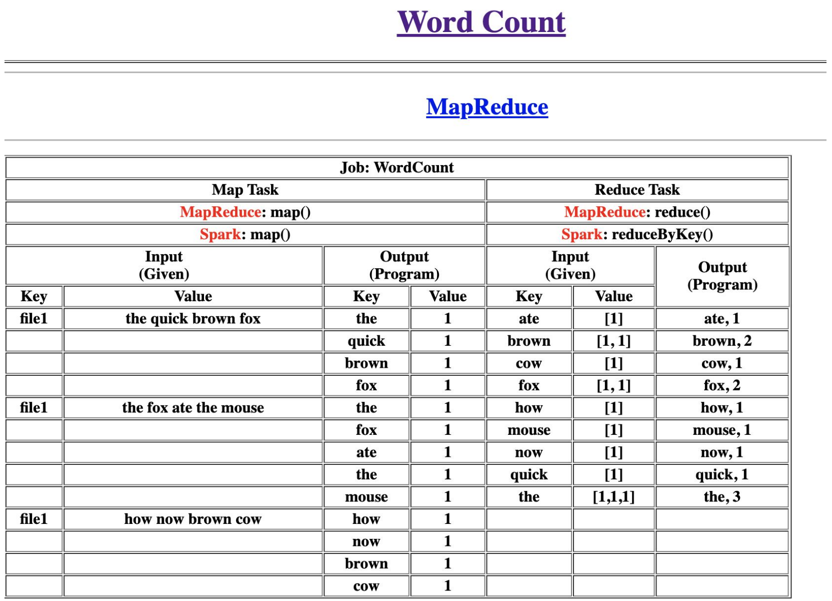
Pyspark:

PySpark is the collaboration of Apache Spark and Python.
Apache Spark is an open-source cluster-computing framework, built around speed, ease of use, and streaming analytics whereas **Python** is a general-purpose, high-level programming language. It provides a wide range of libraries and is majorly used for Machine Learning and Real-Time Streaming Analytics.Python comes with a wide range of libraries like numpy, pandas, scikit-learn, seaborn, matplotlib etc.

Kubernetes :

It is a fast growing open-source system for automating deployment, scaling, and management

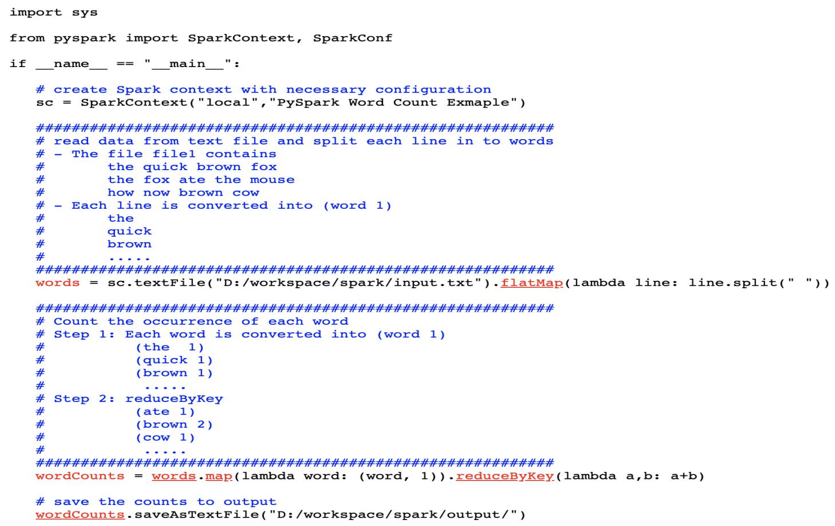
Word Count:



Word Count

MapReduce

Job: WordCount						
Map Task				Reduce Task		
MapReduce: map()				MapReduce: reduce()		
Spark: map()				Spark: reduceByKey()		
Input (Given)		Output (Program)		Input (Given)		Output (Program)
Key	Value	Key	Value	Key	Value	
file1	the quick brown fox	the	1	ate	[1]	ate, 1
		quick	1	brown	[1, 1]	brown, 2
		brown	1	cow	[1]	cow, 1
		fox	1	fox	[1, 1]	fox, 2
file1	the fox ate the mouse	the	1	how	[1]	how, 1
		fox	1	mouse	[1]	mouse, 1
		ate	1	now	[1]	now, 1
		the	1	quick	[1]	quick, 1
		mouse	1	the	[1,1,1]	the, 3
file1	how now brown cow	how	1			
		now	1			
		brown	1			
		cow	1			

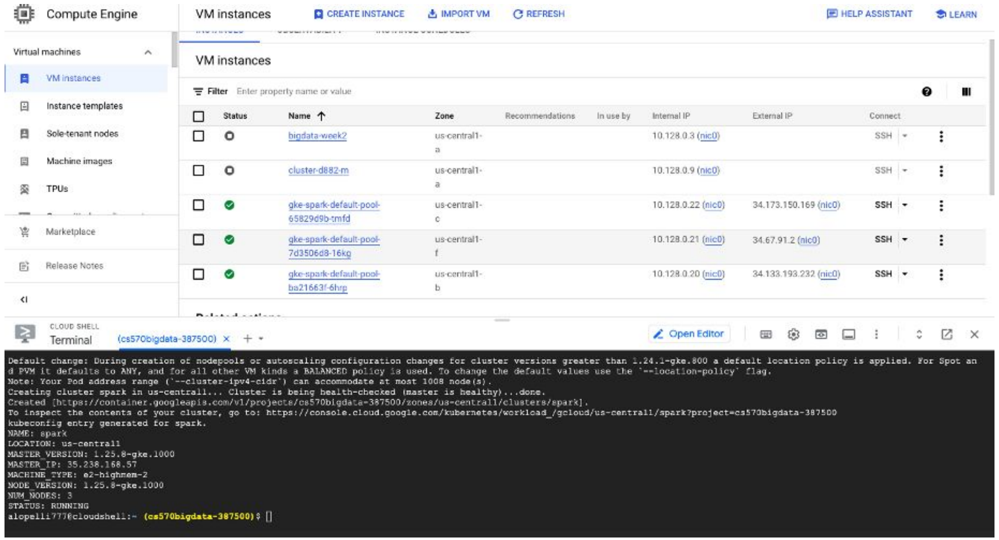


```
import sys

from pyspark import SparkContext, SparkConf

if __name__ == "__main__":

   # create Spark context with necessary configuration
   sc = SparkContext("local","PySpark Word Count Exmaple")

   ##########################################################
   # read data from text file and split each line in to words
   # - The file file1 contains
   #        the quick brown fox
   #        the fox ate the mouse
   #        how now brown cow
   # - Each line is converted into (word 1)
   #        the
   #        quick
   #        brown
   #        .....
   ##########################################################
   words = sc.textFile("D:/workspace/spark/input.txt").flatMap(lambda line: line.split(" "))

   ##########################################################
   # Count the occurrence of each word
   # Step 1: Each word is converted into (word 1)
   #           (the  1)
   #           (quick 1)
   #           (brown 1)
   #            .....
   # Step 2: reduceByKey
   #           (ate 1)
   #           (brown 2)
   #           (cow 1)
   #            .....
   ##########################################################
   wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda a,b: a+b)

   # save the counts to output
   wordCounts.saveAsTextFile("D:/workspace/spark/output/")
```

Implementation

1. Create a cluster on Google Kubernetes Engine(GKE) by running the below command on the cloud shell on GCP

```
gcloud container clusters create spark --num-nodes=1 --machine-type=e2-highmem-2 --region=us-central1
```



VM instances

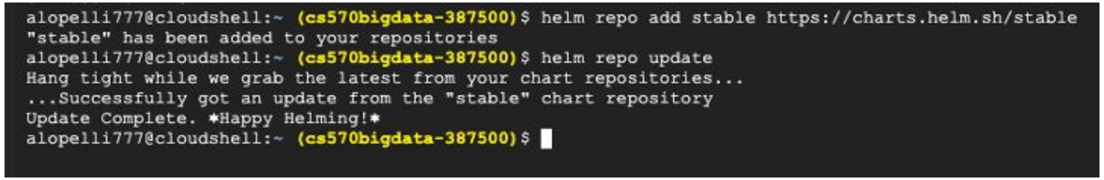
Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
○	bigdata-week2	us-central1-a			10.128.0.3 (nic0)		SSH
○	cluster-d882-m	us-central1-a			10.128.0.9 (nic0)		SSH
✓	gke-spark-default-pool-65829d9b-tmfd	us-central1-c			10.128.0.22 (nic0)	34.173.150.169 (nic0)	SSH
✓	gke-spark-default-pool-7d3506d8-16kg	us-central1-f			10.128.0.21 (nic0)	34.67.91.2 (nic0)	SSH
✓	gke-spark-default-pool-ba21663f-6hrp	us-central1-b			10.128.0.20 (nic0)	34.133.193.232 (nic0)	SSH

```
Default change: During creation of nodepools or autoscaling configuration changes for cluster versions greater than 1.24.1-gke.800 a default location policy is applied. For Spot an
d PVM it defaults to ANY, and for all other VM kinds a BALANCED policy is used. To change the default values use the `--location-policy` flag.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster spark in us-central1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/cs570bigdata-387500/zones/us-central1/clusters/spark].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-central1/spark?project=cs570bigdata-387500
kubeconfig entry generated for spark.
NAME: spark
LOCATION: us-central1
MASTER_VERSION: 1.25.8-gke.1000
MASTER_IP: 35.238.168.57
MACHINE_TYPE: e2-highmem-2
NODE_VERSION: 1.25.8-gke.1000
NUM_NODES: 3
STATUS: RUNNING
alopelli777@cloudshell:~ (cs570bigdata-387500)$ []
```

2. Create image and deploy spark to Kubernetes

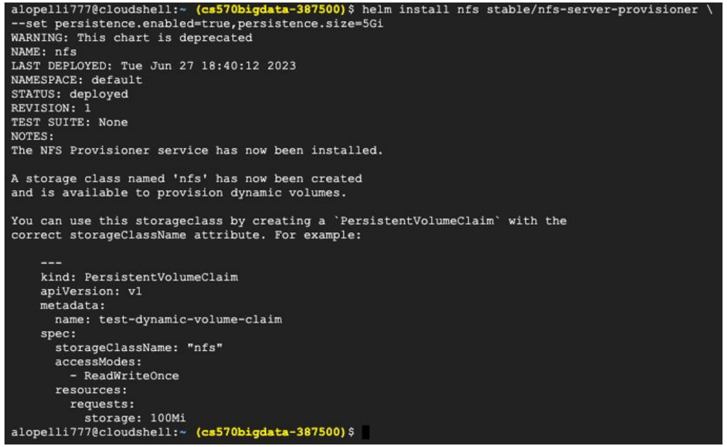
· Install the NFS Server Provisioner

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



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
alopelli777@cloudshell:~ (cs570bigdata-387500)$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "stable" chart repository
Update Complete. *Happy Helming!*
alopelli777@cloudshell:~ (cs570bigdata-387500)$
```

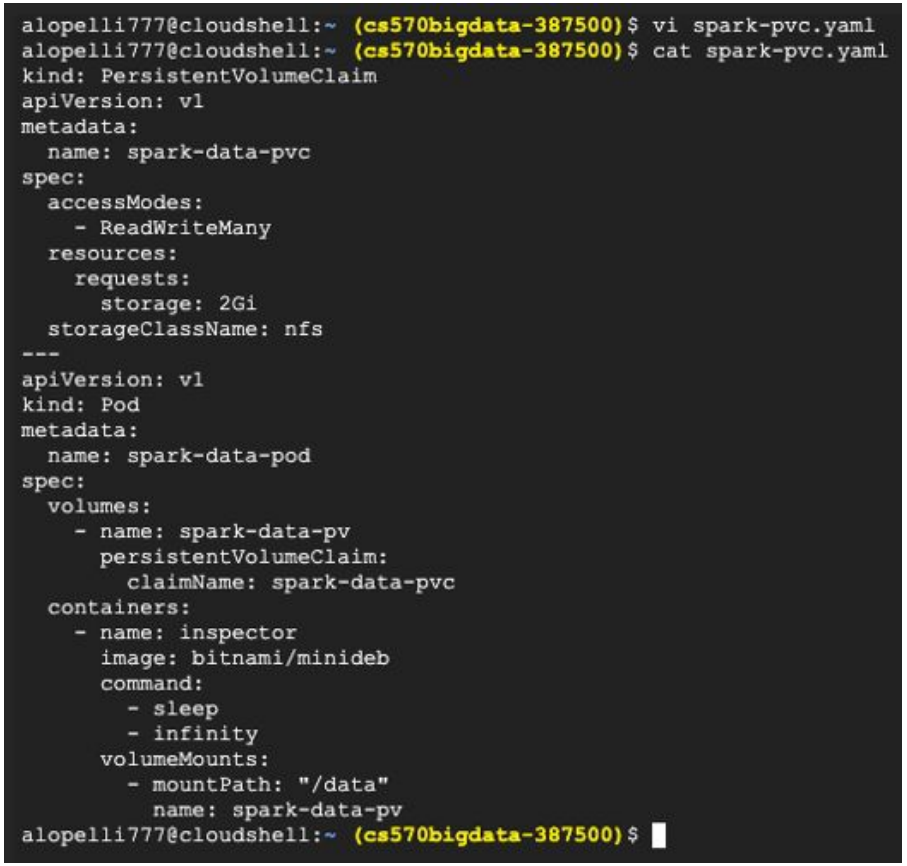
```
helm install nfs stable/nfs-server-provisioner \
--set persistence.enabled=true,persistence.size=5Gi
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ helm install nfs stable/nfs-server-provisioner \
--set persistence.enabled=true,persistence.size=5Gi
WARNING: This chart is deprecated
NAME: nfs
LAST DEPLOYED: Tue Jun 27 18:40:12 2023
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
alopelli777@cloudshell:~ (cs570bigdata-387500)$
```

3. To create a persistent disk volume and a pod to use NFS - create a yaml file with name spar-pvc.yaml and insert the code

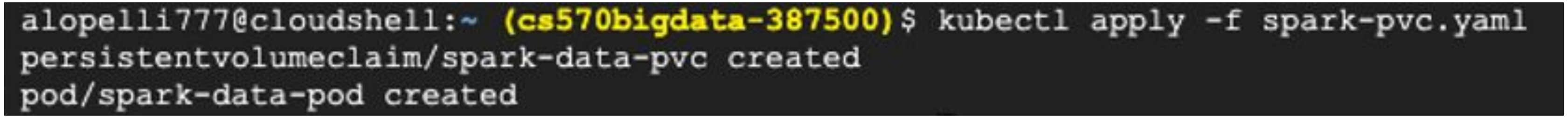
```
vi spark-pvc.yaml

cat spark-pvc.yaml
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ vi spark-pvc.yaml
alopelli777@cloudshell:~ (cs570bigdata-387500)$ 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
alopelli777@cloudshell:~ (cs570bigdata-387500)$
```

4. Apply the above yaml descriptor

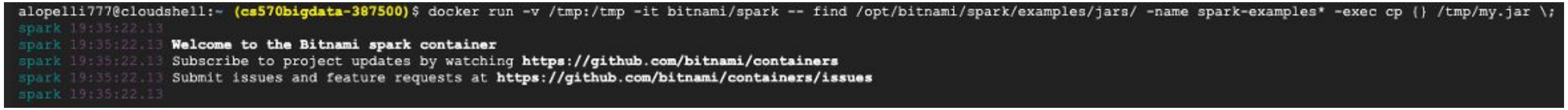
```
kubectl apply -f spark-pvc.yaml
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl apply -f spark-pvc.yaml
persistentvolumeclaim/spark-data-pvc created
pod/spark-data-pod created
```

5.Create and prepare your application JAR file

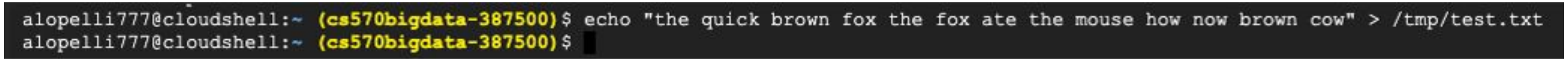
```
docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/
-name spark-examples* -exec cp {} /tmp/my.jar \;
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples* -exec cp {} /tmp/my.jar \;
spark 19:35:22.13
spark 19:35:22.13 Welcome to the Bitnami spark container
spark 19:35:22.13 Subscribe to project updates by watching https://github.com/bitnami/containers
spark 19:35:22.13 Submit issues and feature requests at https://github.com/bitnami/containers/issues
spark 19:35:22.13
```

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

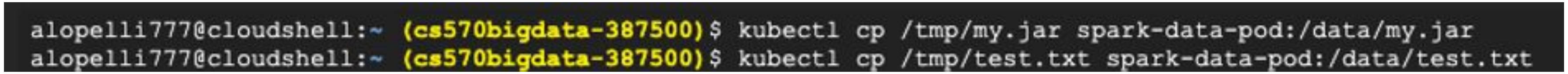
```
echo "the quick brown fox the fox ate the mouse how now brown cow" > /tmp/test.txt
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ echo "the quick brown fox the fox ate the mouse how now brown cow" > /tmp/test.txt
alopelli777@cloudshell:~ (cs570bigdata-387500)$ 
```

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

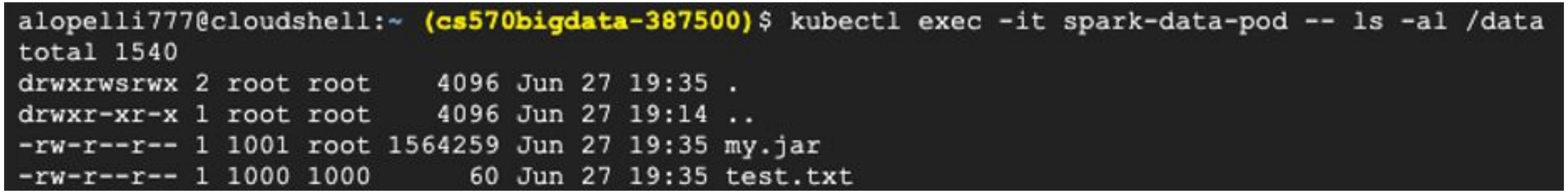
```
kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
kubectl cp /tmp/test.txt spark-data-pod:/data/test.txt
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl cp /tmp/test.txt spark-data-pod:/data/test.txt
```

8.Make sure the files a inside the persistent volume

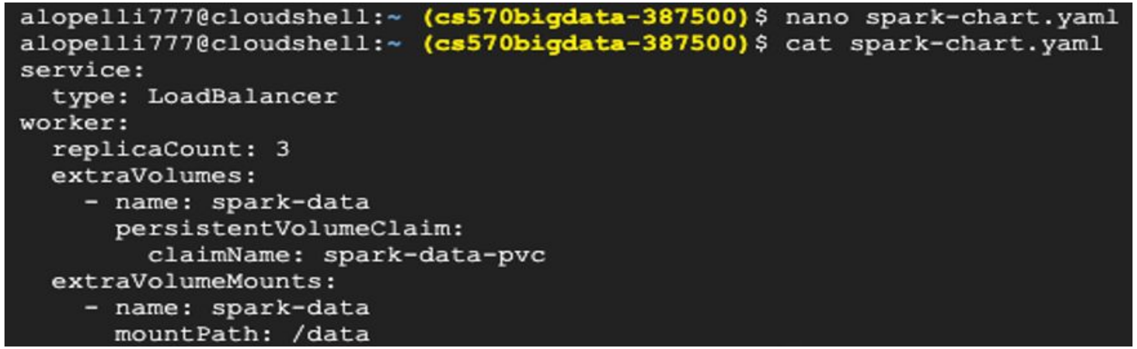
```
kubectl exec -it spark-data-pod -- ls -al /data
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl exec -it spark-data-pod -- ls -al /data
total 1540
drwxrwsrwx 2 root root    4096 Jun 27 19:35 .
drwxr-xr-x 1 root root    4096 Jun 27 19:14 ..
-rw-r--r-- 1 1001 root 1564259 Jun 27 19:35 my.jar
-rw-r--r-- 1 1000 1000      60 Jun 27 19:35 test.txt
```

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

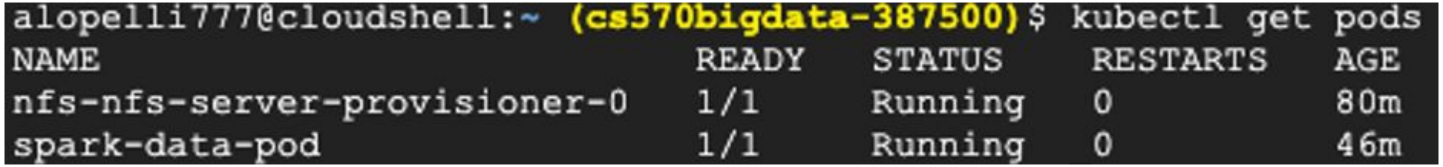
```
nano spark-chart.yaml
cat spark-chart.yaml
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ nano spark-chart.yaml
alopelli777@cloudshell:~ (cs570bigdata-387500)$ 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
```

10. Check the pods is running:

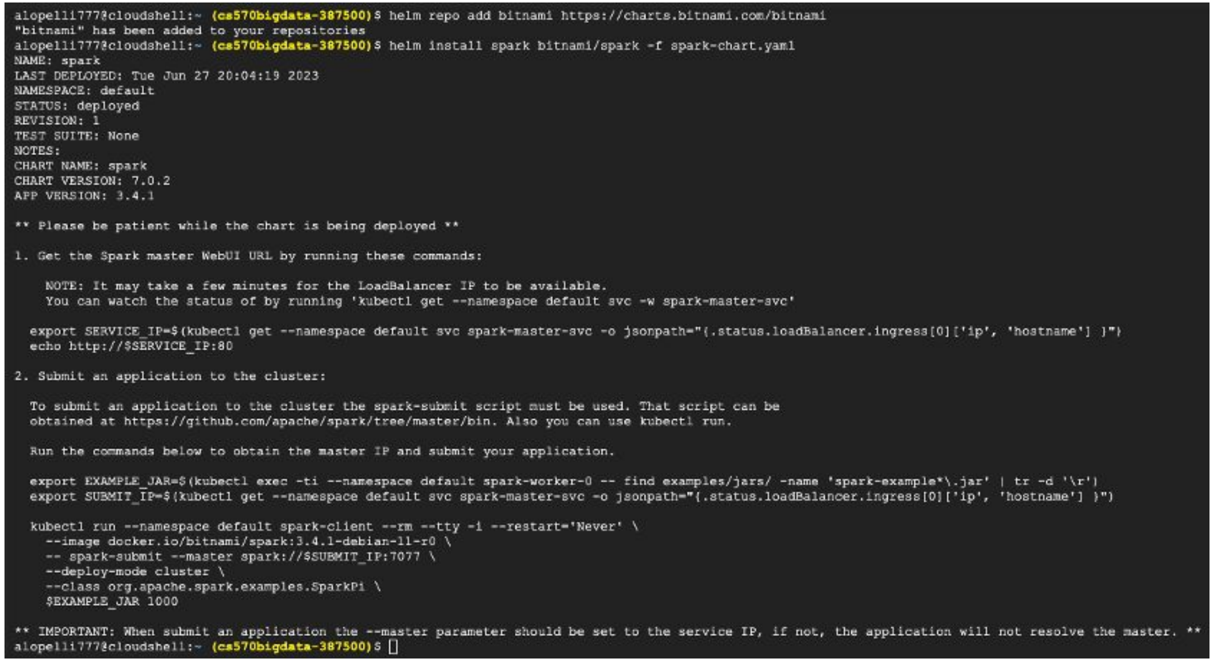
```
kubectl get pods
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
nfs-nfs-server-provisioner-0   1/1     Running   0          80m
spark-data-pod                 1/1     Running   0          46m
```

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

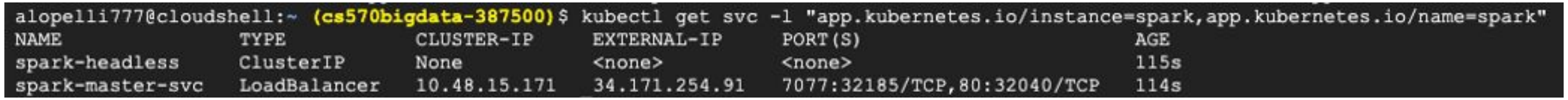
```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm install spark bitnami/spark -f spark-chart.yaml
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
alopelli777@cloudshell:~ (cs570bigdata-387500)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Tue Jun 27 20:04:19 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: spark
CHART VERSION: 7.0.2
APP VERSION: 3.4.1

** 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'] }")

  kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
    --image docker.io/bitnami/spark:3.4.1-debian-11-r0 \
    -- 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. **
alopelli777@cloudshell:~ (cs570bigdata-387500)$ 
```

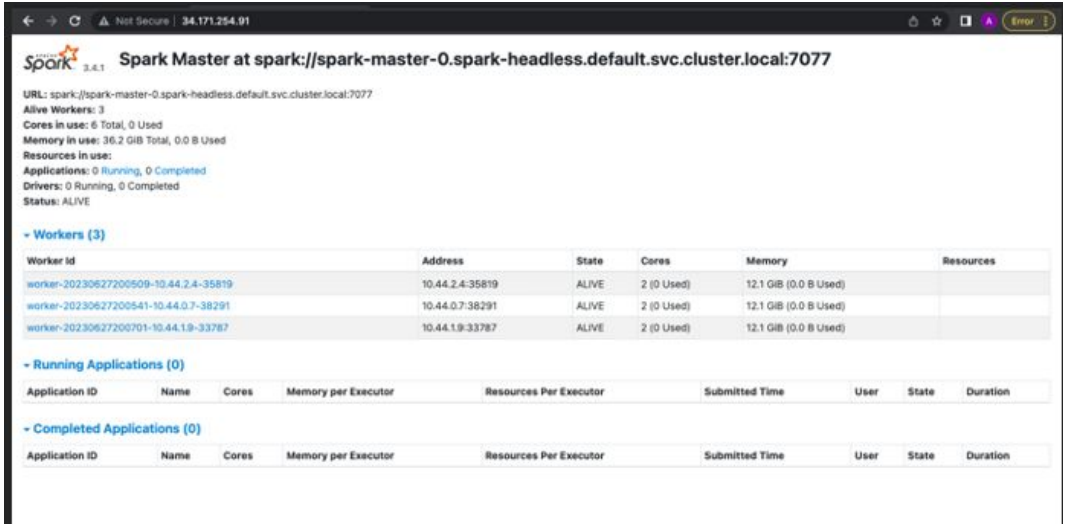
12. Get the external IP of the running pod

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



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ 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>                         115s
spark-master-svc   LoadBalancer   10.48.15.171   34.171.254.91   7077:32185/TCP,80:32040/TCP    114s
```

13. Open the external ip on your browser(I did by pasting the 34.171.254.91 in a separate browser)



Spark Master at spark://spark-master-0.spark-headless.default.svc.cluster.local:7077

URL: spark://spark-master-0.spark-headless.default.svc.cluster.local:7077
Alive Workers: 3
Cores in use: 6 Total, 0 Used
Memory in use: 36.2 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20230627200509-10.44.2.4-35819	10.44.2.4:35819	ALIVE	2 (0 Used)	12.1 GiB (0.0 B Used)	
worker-20230627200541-10.44.0.7-38291	10.44.0.7:38291	ALIVE	2 (0 Used)	12.1 GiB (0.0 B Used)	
worker-20230627200701-10.44.1.9-33787	10.44.1.9:33787	ALIVE	2 (0 Used)	12.1 GiB (0.0 B Used)	

Running Applications (0)

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

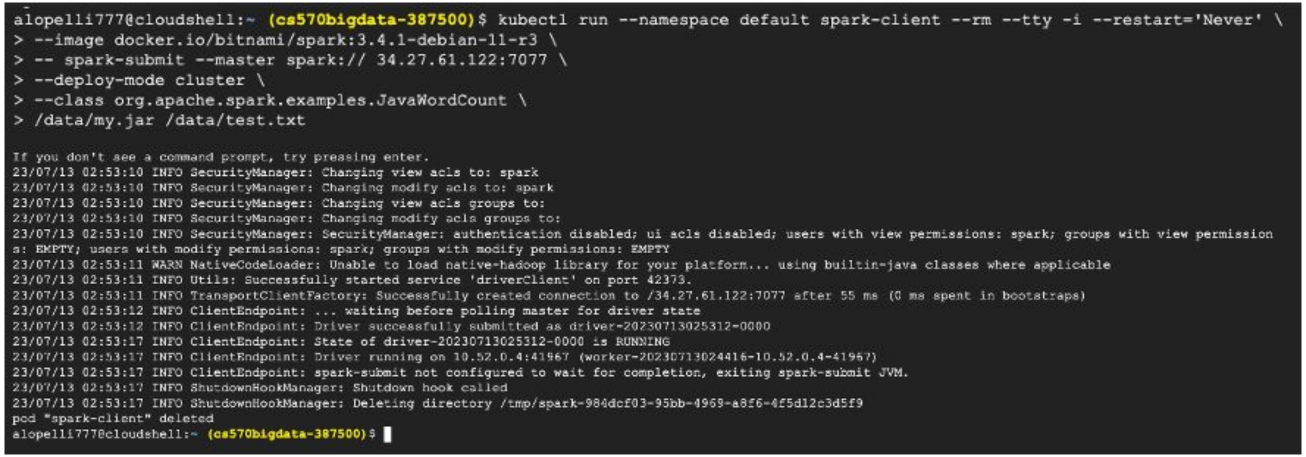
Completed Applications (0)

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

Word count on Spark

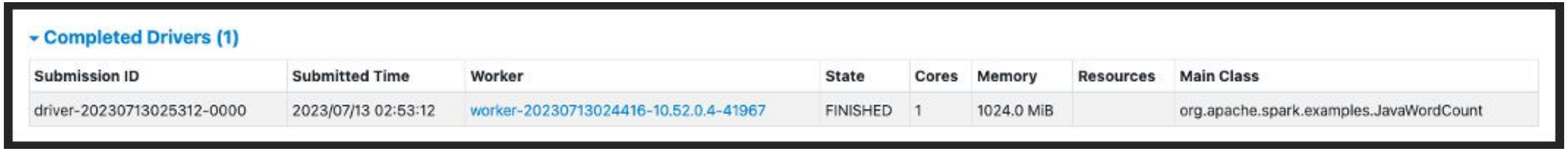
1. Submit a word count task and you see the below content after running the command

```
kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
    --image docker.io/bitnami/spark:3.4.1-debian-11-r3 \
    -- spark-submit --master spark://34.27.61.122:7077 \
    --deploy-mode cluster \
    --class org.apache.spark.examples.JavaWordCount \
    /data/my.jar /data/test.txt
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
> --image docker.io/bitnami/spark:3.4.1-debian-11-r3 \
> -- spark-submit --master spark:// 34.27.61.122:7077 \
> --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.
23/07/13 02:53:10 INFO SecurityManager: Changing view acls to: spark
23/07/13 02:53:10 INFO SecurityManager: Changing modify acls to: spark
23/07/13 02:53:10 INFO SecurityManager: Changing view acls groups to:
23/07/13 02:53:10 INFO SecurityManager: Changing modify acls groups to:
23/07/13 02:53:10 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: spark; groups with view permission
s: EMPTY; users with modify permissions: spark; groups with modify permissions: EMPTY
23/07/13 02:53:11 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
23/07/13 02:53:11 INFO Utils: Successfully started service 'driverClient' on port 42373.
23/07/13 02:53:11 INFO TransportClientFactory: Successfully created connection to /34.27.61.122:7077 after 55 ms (0 ms spent in bootstraps)
23/07/13 02:53:12 INFO ClientEndpoint: ... waiting before polling master for driver state
23/07/13 02:53:12 INFO ClientEndpoint: Driver successfully submitted as driver-20230713025312-0000
23/07/13 02:53:17 INFO ClientEndpoint: State of driver-20230713025312-0000 is RUNNING
23/07/13 02:53:17 INFO ClientEndpoint: Driver running on 10.52.0.4:41967 (worker-20230713024416-10.52.0.4-41967)
23/07/13 02:53:17 INFO ClientEndpoint: spark-submit not configured to wait for completion, exiting spark-submit JVM.
23/07/13 02:53:17 INFO ShutdownHookManager: Shutdown hook called
23/07/13 02:53:17 INFO ShutdownHookManager: Deleting directory /tmp/spark-984dcf03-95bb-4969-a8f6-4f5d12c3d5f9
pod "spark-client" deleted
alopelli777@cloudshell:~ (cs570bigdata-387500)$
```

2.And on your browser, you should see this task finished.

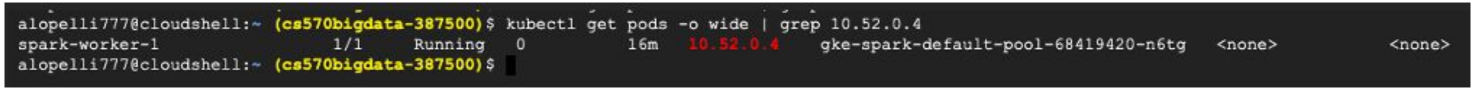


▾ Completed Drivers (1)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20230713025312-0000	2023/07/13 02:53:12	worker-20230713024416-10.52.0.4-41967	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordCount

3. Get the name of the worker node(my worker node address is 10.52.0.4)

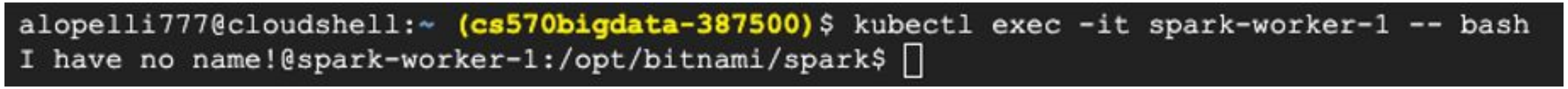
```
kubectl get pods -o wide | grep WORKER-NODE-ADDRESS
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl get pods -o wide | grep 10.52.0.4
spark-worker-1              1/1     Running   0          16m   10.52.0.4    gke-spark-default-pool-68419420-n6tg   <none>           <none>
alopelli777@cloudshell:~ (cs570bigdata-387500)$
```

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

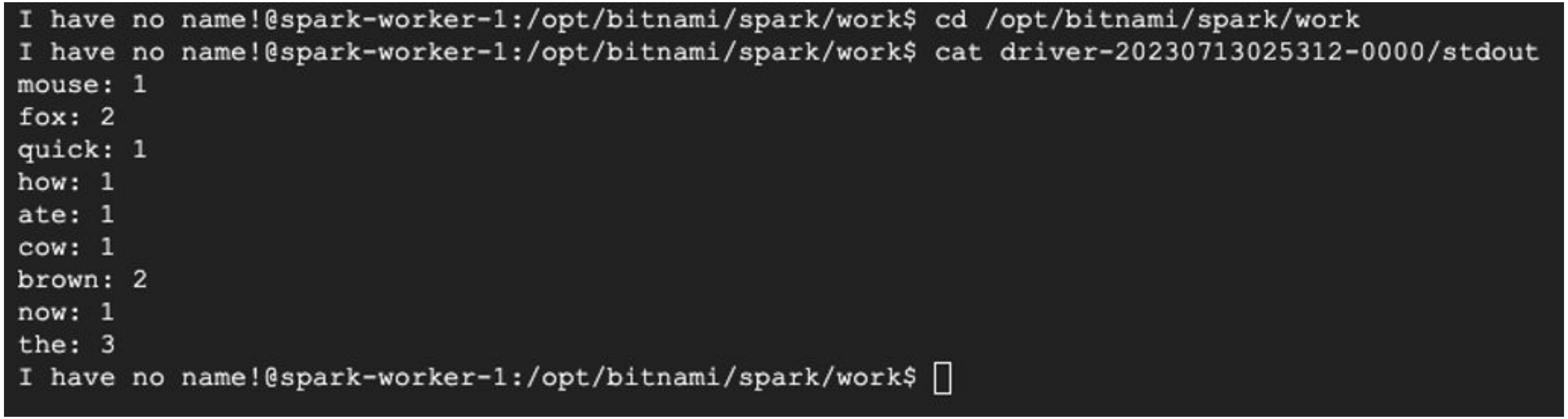
```
kubectl exec -it <Worker node name> -- bash
```



```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl exec -it spark-worker-1 -- bash
I have no name!@spark-worker-1:/opt/bitnami/spark$
```

```
cd /opt/bitnami/spark/work
cat <task-name>/stdout
```

The task name here is the Submission ID in the completed Drivers section of the URL

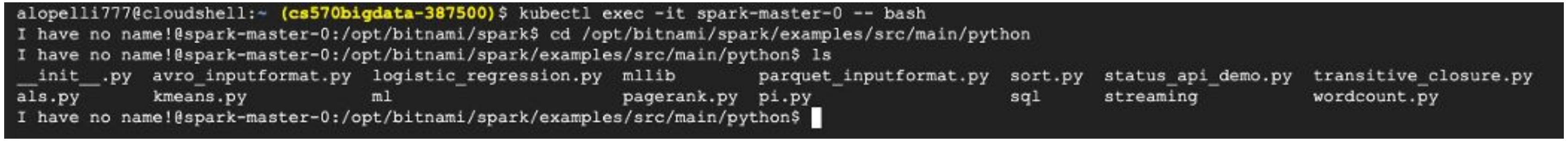


```
I have no name!@spark-worker-1:/opt/bitnami/spark/work$ cd /opt/bitnami/spark/work
I have no name!@spark-worker-1:/opt/bitnami/spark/work$ cat driver-20230713025312-0000/stdout
mouse: 1
fox: 2
quick: 1
how: 1
ate: 1
cow: 1
brown: 2
now: 1
the: 3
I have no name!@spark-worker-1:/opt/bitnami/spark/work$ 
```

Running python PageRank on PySpark on the pods

1. Execute the spark master pods and Go to the directory where pagerank.py located

```
kubectl exec -it spark-master-0 – bash
cd /opt/bitnami/spark/examples/src/main/python
```

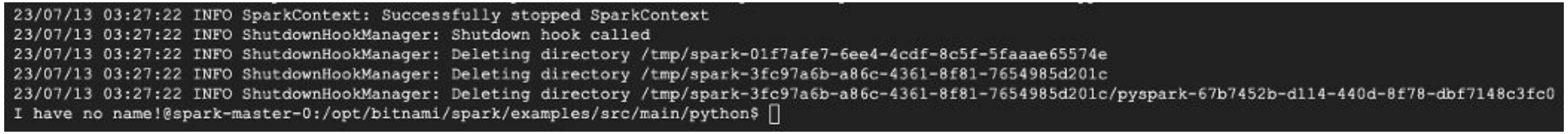


```
alopelli777@cloudshell:~ (cs570bigdata-387500)$ kubectl exec -it spark-master-0 -- bash
I have no name!@spark-master-0:/opt/bitnami/spark$ cd /opt/bitnami/spark/examples/src/main/python
I have no name!@spark-master-0:/opt/bitnami/spark/examples/src/main/python$ ls
__init__.py  avro_inputformat.py  logistic_regression.py  mllib        parquet_inputformat.py  sort.py  status_api_demo.py  transitive_closure.py
als.py       kmeans.py            ml                      pagerank.py  pi.py                   sql      streaming           wordcount.py
I have no name!@spark-master-0:/opt/bitnami/spark/examples/src/main/python$
```

2.Run the pagerank using pyspark

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

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



```
23/07/13 03:27:22 INFO SparkContext: Successfully stopped SparkContext
23/07/13 03:27:22 INFO ShutdownHookManager: Shutdown hook called
23/07/13 03:27:22 INFO ShutdownHookManager: Deleting directory /tmp/spark-01f7afe7-6ee4-4cdf-8c5f-5faaae65574e
23/07/13 03:27:22 INFO ShutdownHookManager: Deleting directory /tmp/spark-3fc97a6b-a86c-4361-8f81-7654985d201c
23/07/13 03:27:22 INFO ShutdownHookManager: Deleting directory /tmp/spark-3fc97a6b-a86c-4361-8f81-7654985d201c/pyspark-67b7452b-d114-440d-8f78-dbf7148c3fc0
I have no name!@spark-master-0:/opt/bitnami/spark/examples/src/main/python$ 
```

Thank You!!!