PySpark on Kubernetes: Word Count + PageRank

CS570 Big Data Processing Project
By Feven Araya
Instructor: Dr. Chang, Henry

Table of contents

1. Introduction
2. Design
3. Implementation
4. Testing
5. Enhancement
6. Conclusion
7. References

01

Introduction

- This presentation will explore the implementation of a PySpark project on Kubernetes, specifically focusing on the Word Count and PageRank algorithms.
- It will detail the process from setup to execution and testing within a Kubernetes environment, leveraging GCP (Google Cloud Platform) and GKE (Google Kubernetes Engine).

Purpose: The aim is to demonstrate the power and scalability of PySpark in processing large datasets efficiently, as well as to highlight the practical application of Kubernetes in managing containerized applications.

02
Design

Architecture Overview:

- Brief on the Kubernetes architecture used, including nodes, pods, and persistent volumes.
- Explanation of the PySpark environment setup, including Spark configuration and dependencies.

Component Design:

- Describe the setup of persistent volumes using NFS (Network File System) for data persistence across the cluster.
- Outline the configuration of the Spark cluster within Kubernetes, focusing on resource allocation and management.

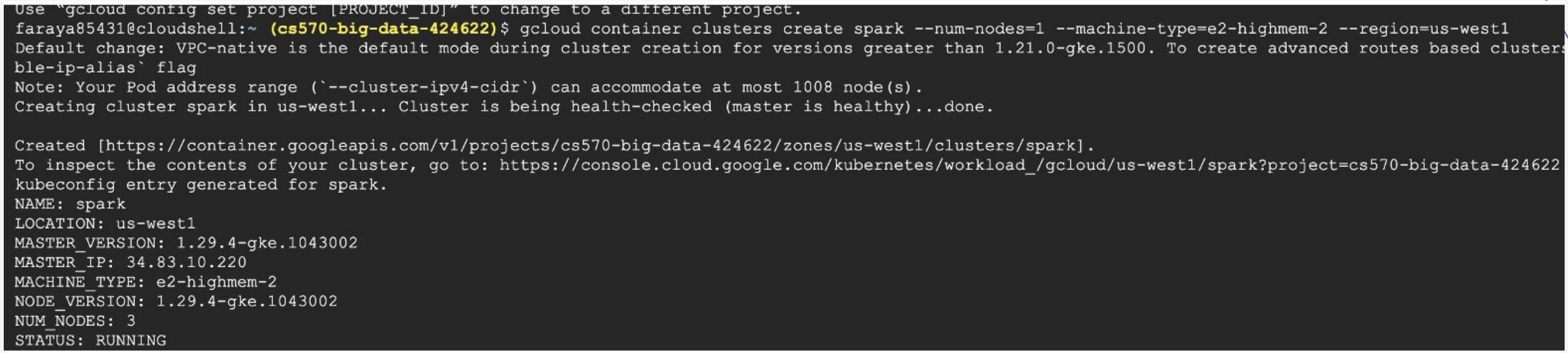
Data Flow:

- Detailed walkthrough of the data flow for the Word Count and PageRank algorithms from input to output.
- Illustration of how data is distributed and processed across the Spark cluster.

03

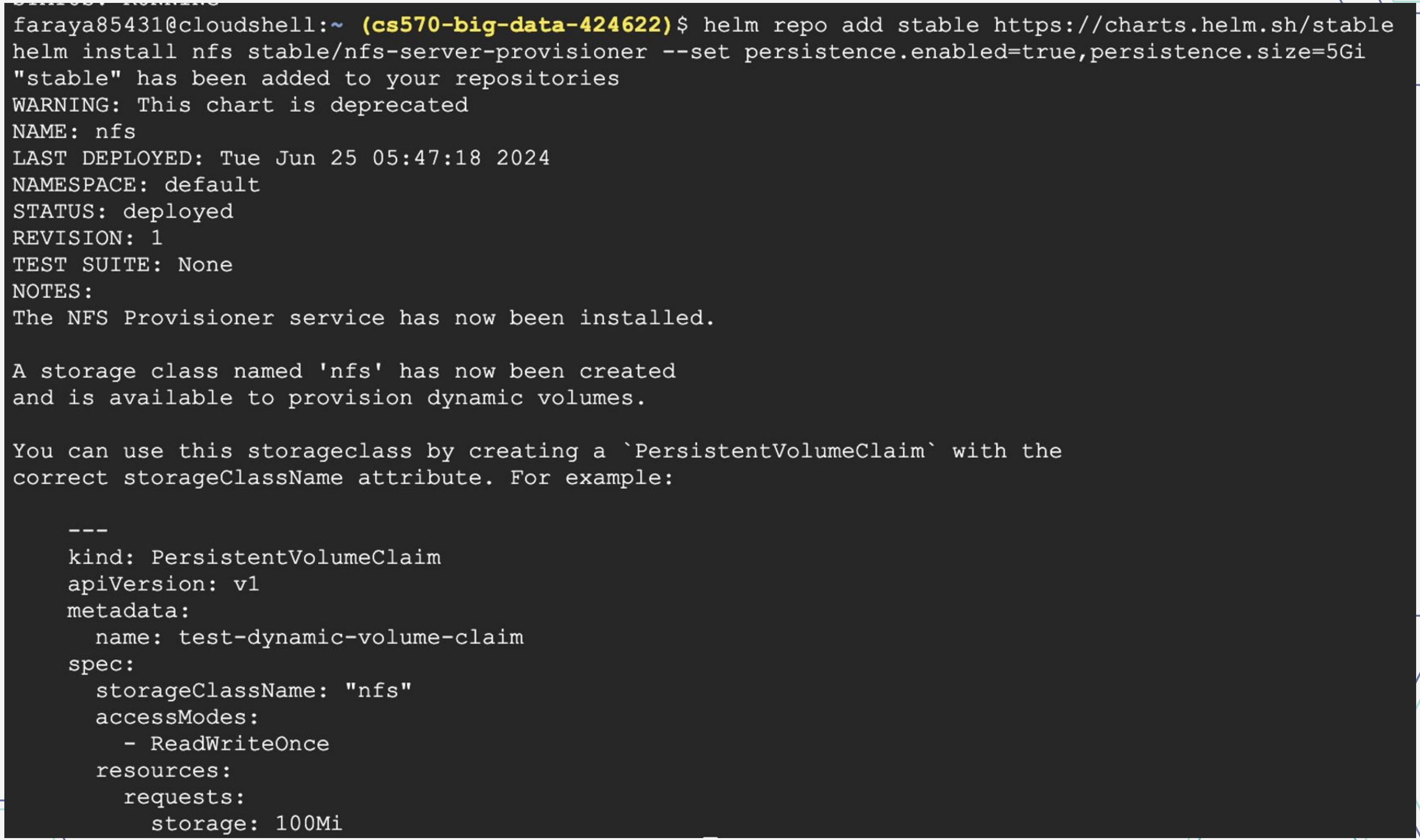
Implementation

Create a cluster on GKE



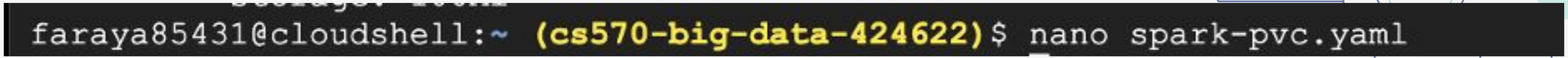
```
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
faraya85431@cloudshell:~ (cs570-big-data-424622)$ gcloud container clusters create spark --num-nodes=1 --machine-type=e2-highmem-2 --region=us-west1
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters
ble-ip-alias` flag
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-424622/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-424622
kubeconfig entry generated for spark.
NAME: spark
LOCATION: us-west1
MASTER_VERSION: 1.29.4-gke.1043002
MASTER_IP: 34.83.10.220
MACHINE_TYPE: e2-highmem-2
NODE_VERSION: 1.29.4-gke.1043002
NUM_NODES: 3
STATUS: RUNNING
```

Install the NFS Server Provisioner using Helm

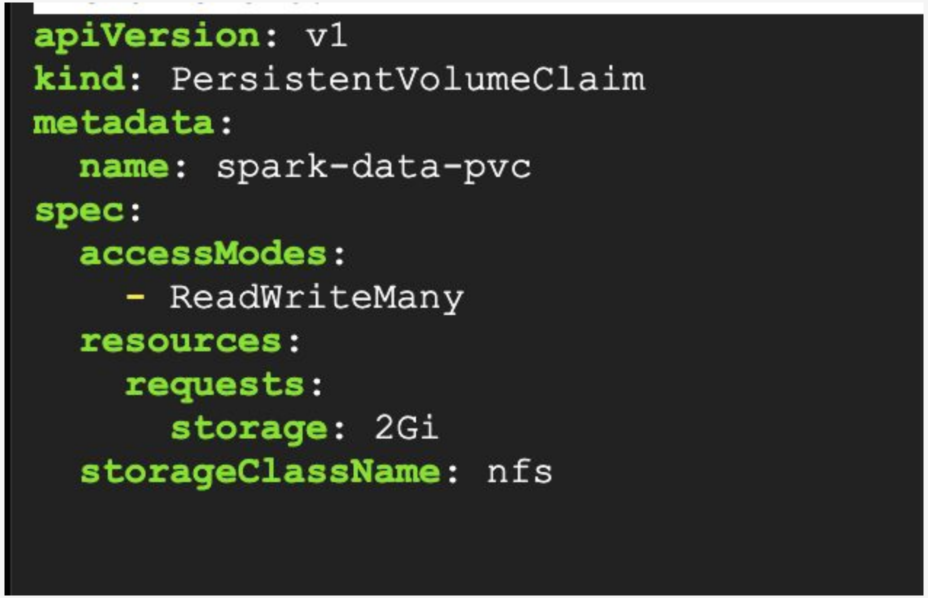


```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ helm repo add stable https://charts.helm.sh/stable
helm install nfs stable/nfs-server-provisioner --set persistence.enabled=true,persistence.size=5Gi
"stable" has been added to your repositories
WARNING: This chart is deprecated
NAME: nfs
LAST DEPLOYED: Tue Jun 25 05:47:18 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
```

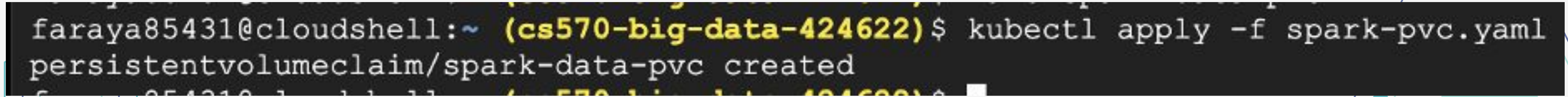
YAML for Persistent Volume Claim



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ nano spark-pvc.yaml
```



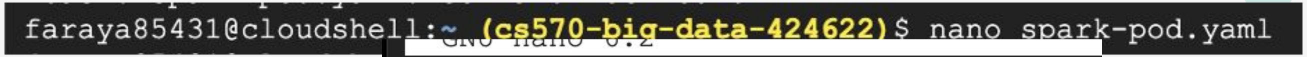
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: spark-data-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 2Gi
  storageClassName: nfs
```



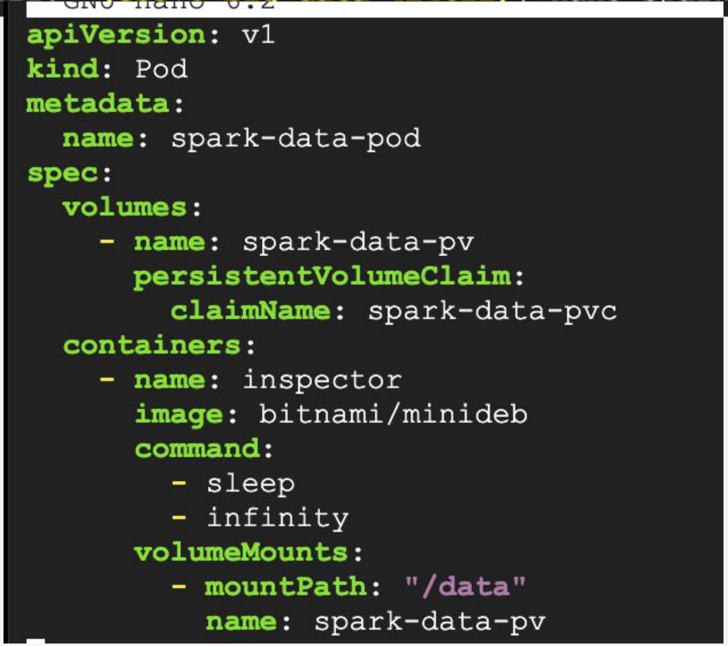
```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl apply -f spark-pvc.yaml
persistentvolumeclaim/spark-data-pvc created
```

This YAML configuration creates a Persistent Volume Claim named spark-data-pvc that will request 2 GiB of storage with ReadWriteMany access mode, using the NFS storage class provided by the NFS Server Provisioner you set up earlier.

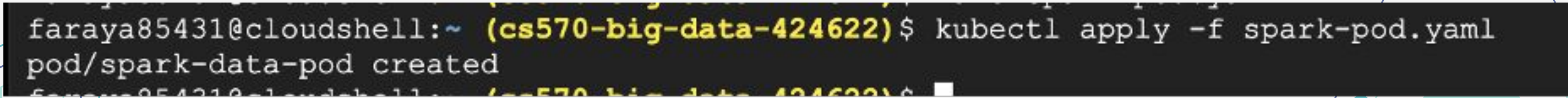
YAML for Pod



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ nano spark-pod.yaml
```

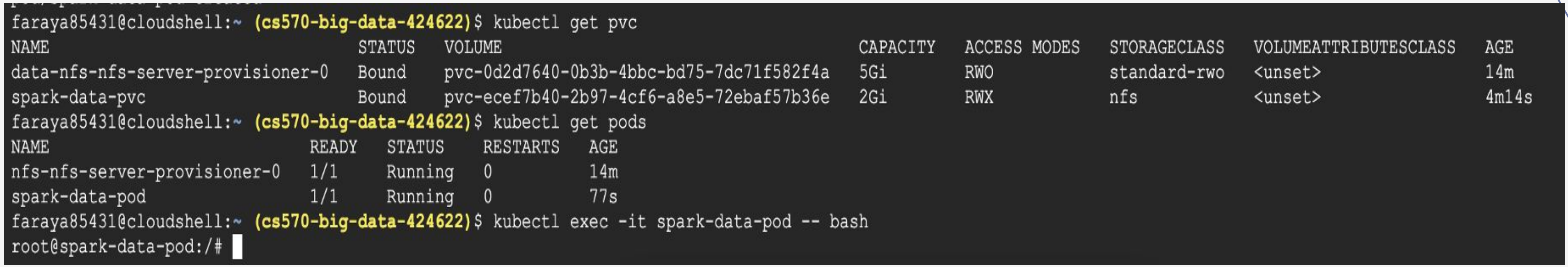


```
apiVersion: v1
kind: Pod
metadata:
  name: spark-data-pod
spec:
  volumes:
    - name: spark-data-pv
      persistentVolumeClaim:
        claimName: spark-data-pvc
  containers:
    - name: inspector
      image: bitnami/minideb
      command:
        - sleep
        - infinity
      volumeMounts:
        - mountPath: "/data"
          name: spark-data-pv
```



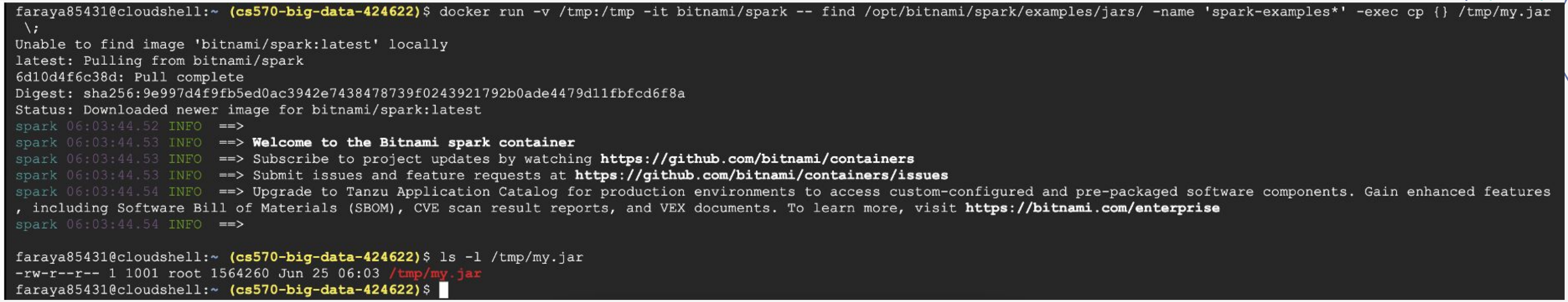
```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl apply -f spark-pod.yaml
pod/spark-data-pod created
```

This YAML defines a Pod named `spark-data-pod` that mounts the PVC `spark-data-pvc` at `/data`. The Pod runs a basic container from the `bitnami/minideb` image, which just sleeps indefinitely to keep the Pod running for inspection or debugging purposes.



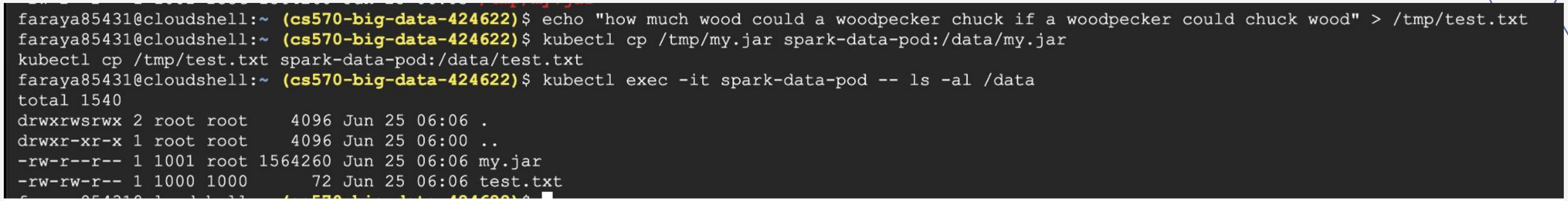
```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl get pvc
NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
data-nfs-nfs-server-provisioner-0   Bound    pvc-0d2d7640-0b3b-4bbc-bd75-7dc71f582f4a   5Gi        RWO            standard-rwo   <unset>                 14m
spark-data-pvc                      Bound    pvc-ecef7b40-2b97-4cf6-a8e5-72ebaf57b36e   2Gi        RWX            nfs            <unset>                 4m14s
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
nfs-nfs-server-provisioner-0   1/1     Running   0          14m
spark-data-pod                 1/1     Running   0          77s
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl exec -it spark-data-pod -- bash
root@spark-data-pod:/#
```

- **Check the PVC**: Ensure it's correctly bound to a Persistent Volume by running:
- **Check the Pod**: Verify that the Pod is running and not in an error state:
- **Inspect the Pod's operation**: If you need to check the mounting and the behavior of the Pod, you can enter the container:



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ 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
6d10d4f6c38d: Pull complete
Digest: sha256:9e997d4f9fb5ed0ac3942e7438478739f0243921792b0ade4479d11fbfcd6f8a
Status: Downloaded newer image for bitnami/spark:latest
spark 06:03:44.52 INFO  ==>
spark 06:03:44.53 INFO  ==> Welcome to the Bitnami spark container
spark 06:03:44.53 INFO  ==> Subscribe to project updates by watching https://github.com/bitnami/containers
spark 06:03:44.53 INFO  ==> Submit issues and feature requests at https://github.com/bitnami/containers/issues
spark 06:03:44.54 INFO  ==> Upgrade to Tanzu Application Catalog for production environments to access custom-configured and pre-packaged software components. Gain enhanced features
, including Software Bill of Materials (SBOM), CVE scan result reports, and VEX documents. To learn more, visit https://bitnami.com/enterprise
spark 06:03:44.54 INFO  ==>

faraya85431@cloudshell:~ (cs570-big-data-424622)$ ls -l /tmp/my.jar
-rw-r--r-- 1 1001 root 1564260 Jun 25 06:03 /tmp/my.jar
faraya85431@cloudshell:~ (cs570-big-data-424622)$
```

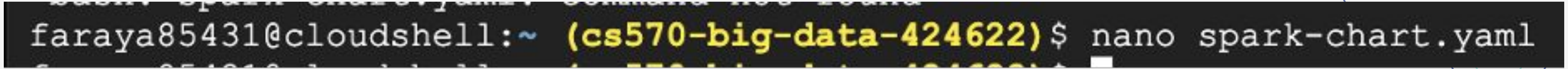
Use a Docker container to locate and copy a JAR file from the Spark examples directory to a designated location on your host machine



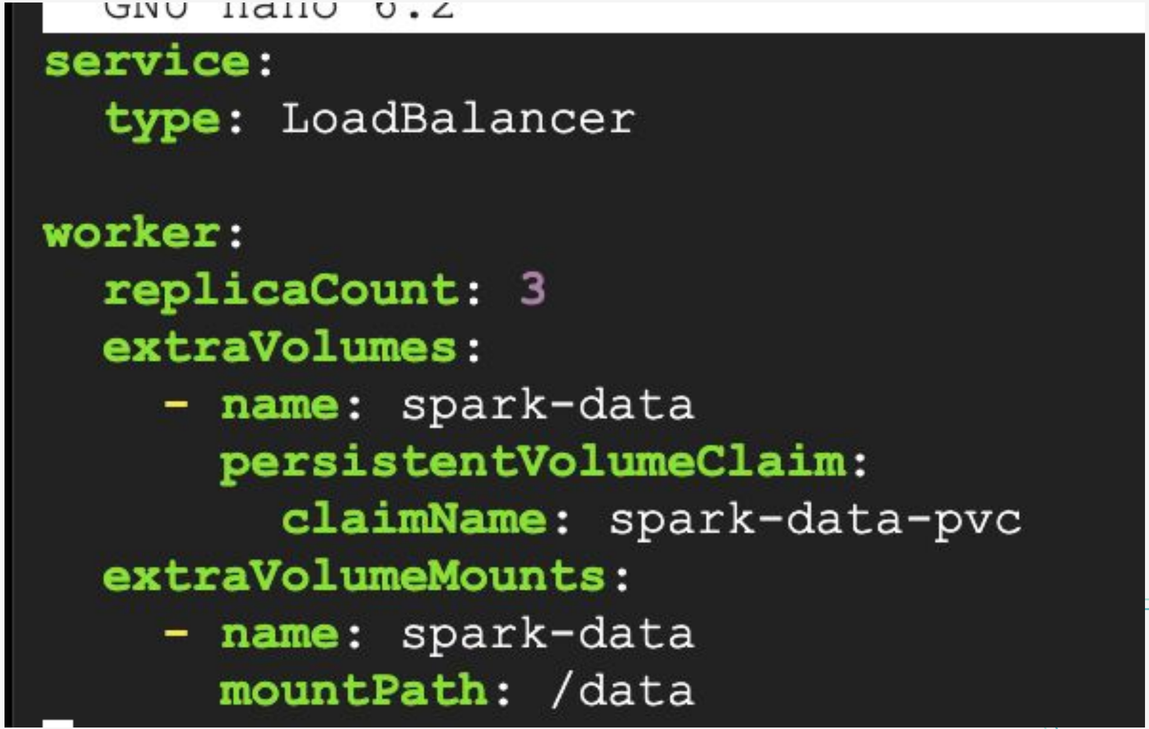
```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ echo "how much wood could a woodpecker chuck if a woodpecker could chuck wood" > /tmp/test.txt
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
kubectl cp /tmp/test.txt spark-data-pod:/data/test.txt
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl exec -it spark-data-pod -- ls -al /data
total 1540
drwxrwsrwx 2 root root    4096 Jun 25 06:06 .
drwxr-xr-x 1 root root    4096 Jun 25 06:00 ..
-rw-r--r-- 1 1001 root 1564260 Jun 25 06:06 my.jar
-rw-rw-r-- 1 1000 1000      72 Jun 25 06:06 test.txt
```

The commands executed include creating a text file with the content ”how much wood could a woodpecker chuck if a woodpecker could chuck wood” in /tmp/test.txt, copying a JAR file from /tmp/my.jar to the /data directory in the spark-data-pod pod, and verifying the files in the /data directory of the spark-data-pod pod, which includes my.jar and test.txt.

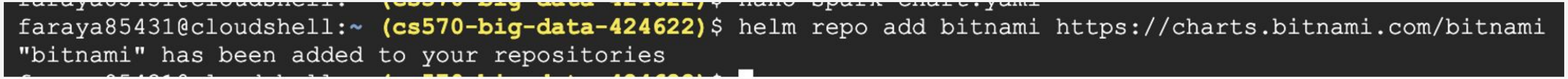
Create spark-chart.yaml file



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ nano spark-chart.yaml
```

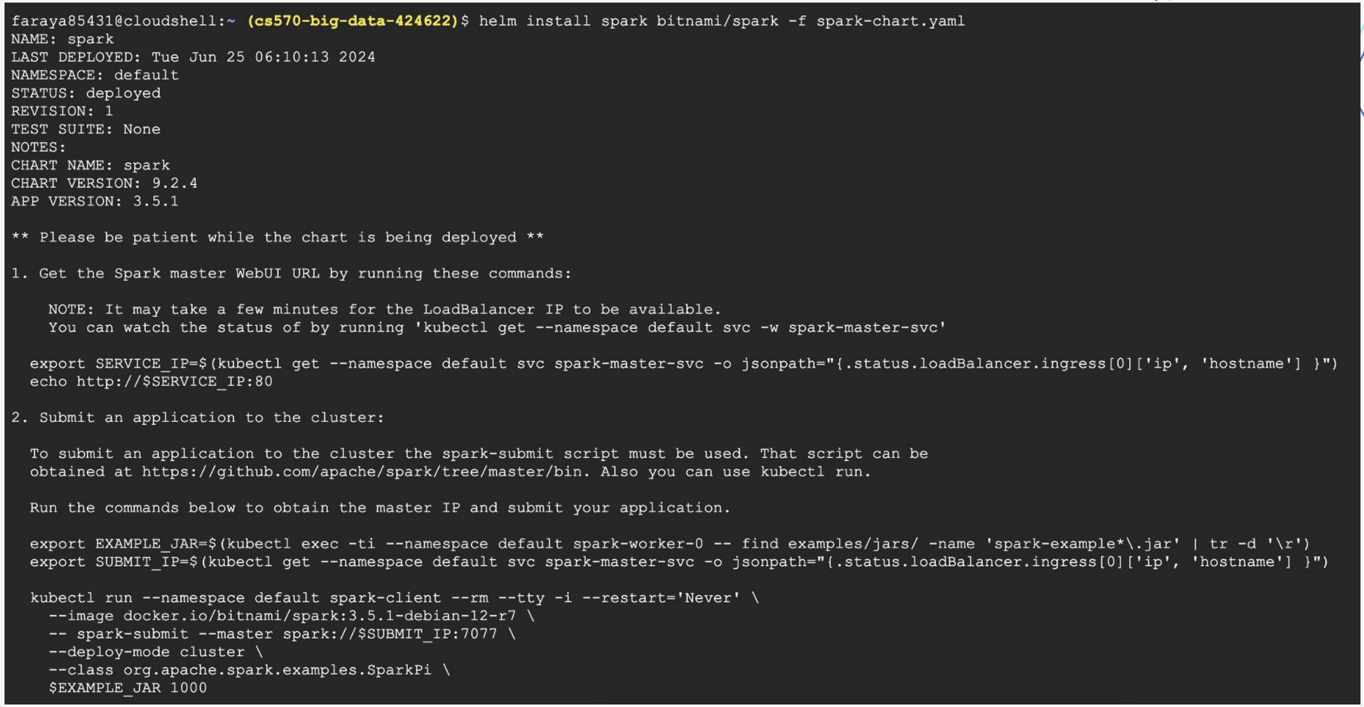


```
service:
  type: LoadBalancer

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

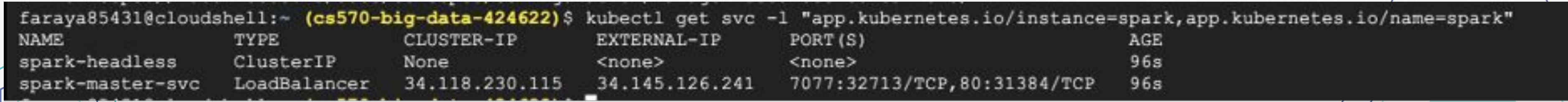


```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
```

You need to add the Bitnami repository to your Helm setup to access their charts:



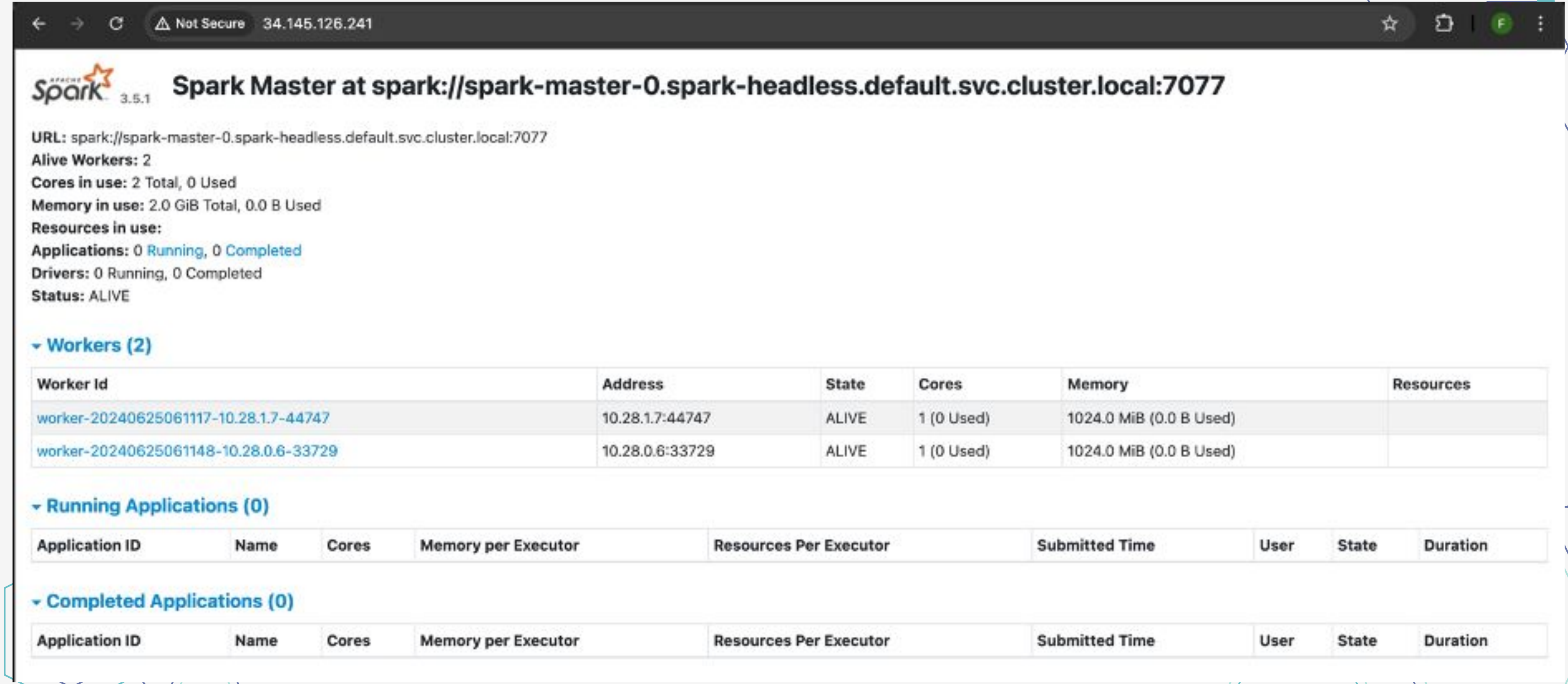
```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Tue Jun 25 06:10:13 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: spark
CHART VERSION: 9.2.4
APP VERSION: 3.5.1

** 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'] }")

  kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
    --image docker.io/bitnami/spark:3.5.1-debian-12-r7 \
    -- spark-submit --master spark://$SUBMIT_IP:7077 \
    --deploy-mode cluster \
    --class org.apache.spark.examples.SparkPi \
    $EXAMPLE_JAR 1000
```



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ 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>                         96s
spark-master-svc   LoadBalancer   34.118.230.115   34.145.126.241   7077:32713/TCP,80:31384/TCP    96s
```

Now you'll deploy Apache Spark using the Helm chart with your configuration file:

Run in 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: 2
Cores in use: 2 Total, 0 Used
Memory in use: 2.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20240625061117-10.28.1.7-44747	10.28.1.7:44747	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20240625061148-10.28.0.6-33729	10.28.0.6:33729	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

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

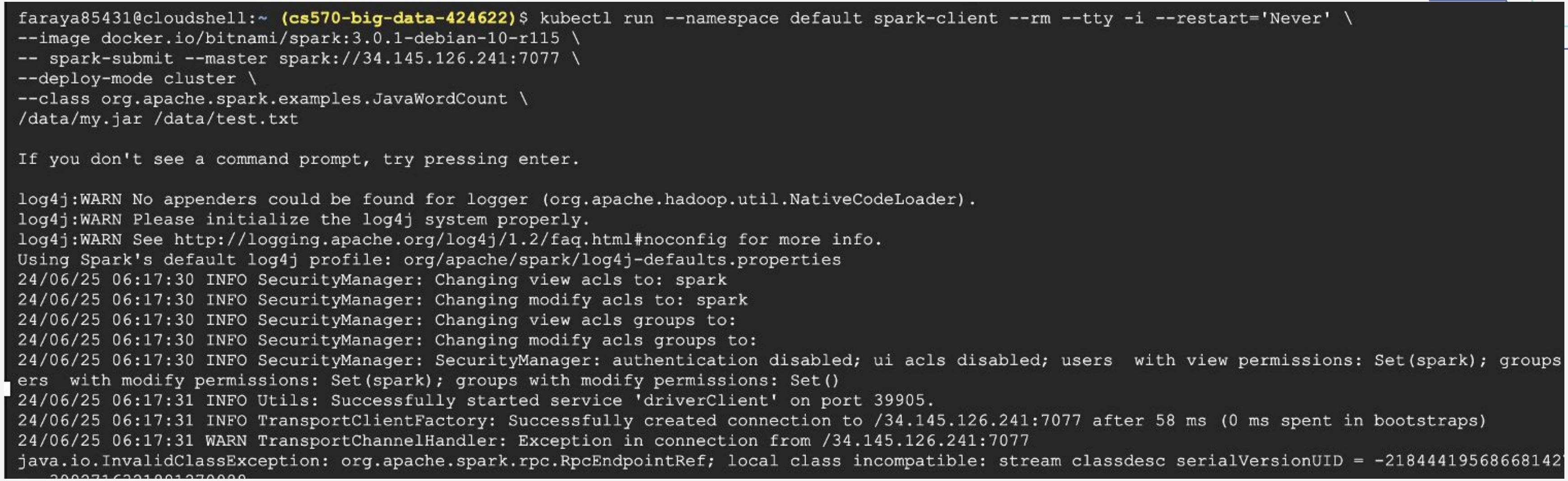
Completed Applications (0)

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

04

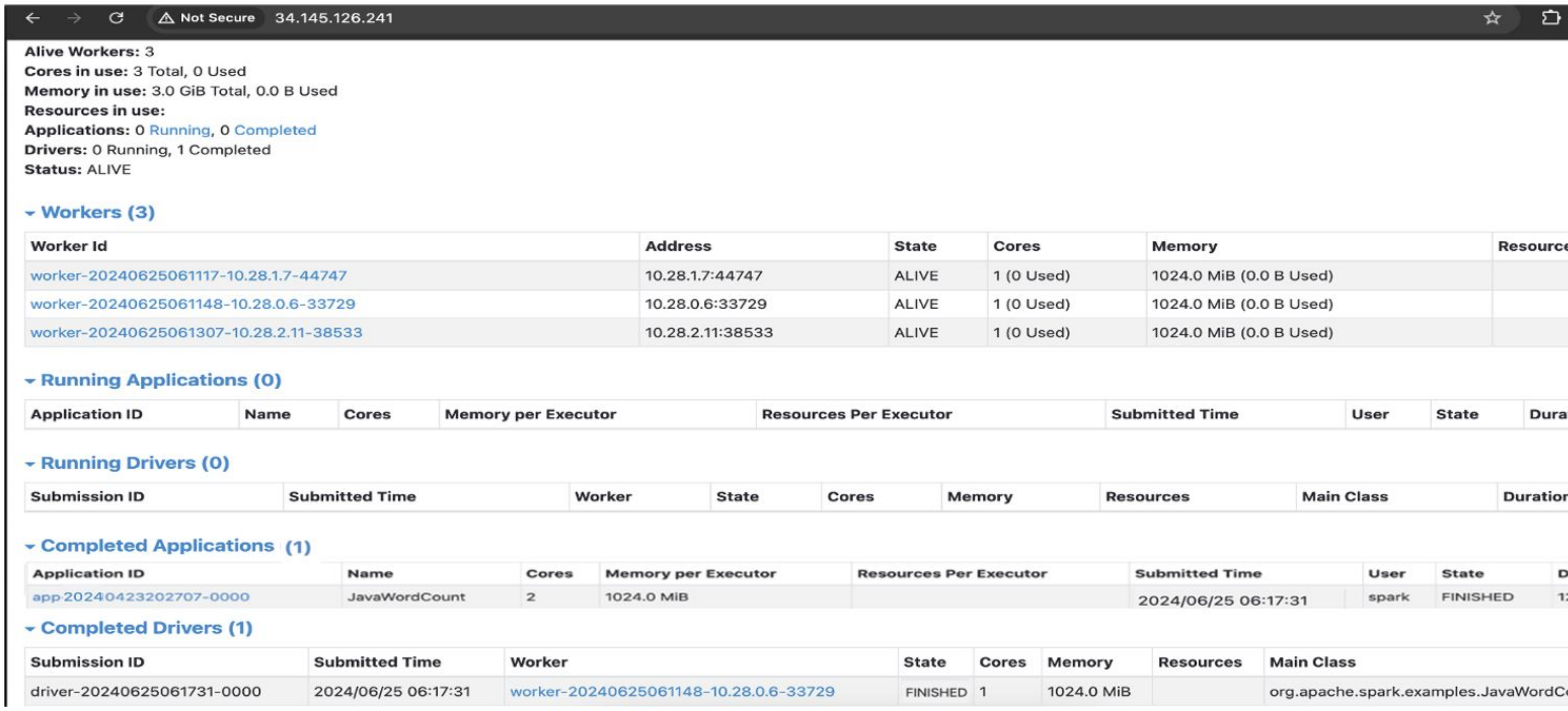
Test

Word Count on Spark



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ 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://34.145.126.241:7077 \
--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
24/06/25 06:17:30 INFO SecurityManager: Changing view acls to: spark
24/06/25 06:17:30 INFO SecurityManager: Changing modify acls to: spark
24/06/25 06:17:30 INFO SecurityManager: Changing view acls groups to:
24/06/25 06:17:30 INFO SecurityManager: Changing modify acls groups to:
24/06/25 06:17:30 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(spark); groups
ers  with modify permissions: Set(spark); groups with modify permissions: Set()
24/06/25 06:17:31 INFO Utils: Successfully started service 'driverClient' on port 39905.
24/06/25 06:17:31 INFO TransportClientFactory: Successfully created connection to /34.145.126.241:7077 after 58 ms (0 ms spent in bootstraps)
24/06/25 06:17:31 WARN TransportChannelHandler: Exception in connection from /34.145.126.241:7077
java.io.InvalidClassException: org.apache.spark.rpc.RpcEndpointRef; local class incompatible: stream classdesc serialVersionUID = -2184419568668142
```

Word Count on Spark



Not Secure 34.145.126.241

Alive Workers: 3
Cores in use: 3 Total, 0 Used
Memory in use: 3.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 1 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resource
worker-20240625061117-10.28.1.7-44747	10.28.1.7:44747	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20240625061148-10.28.0.6-33729	10.28.0.6:33729	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20240625061307-10.28.2.11-38533	10.28.2.11:38533	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

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

Running Drivers (0)

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

Completed Applications (1)

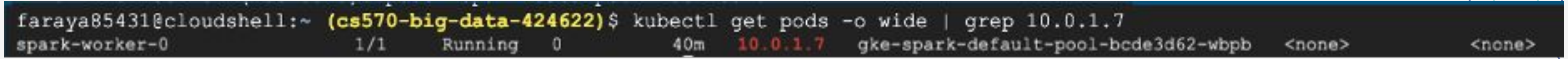
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	D
app-20240423202707-0000	JavaWordCount	2	1024.0 MiB		2024/06/25 06:17:31	spark	FINISHED	1

Completed Drivers (1)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20240625061731-0000	2024/06/25 06:17:31	worker-20240625061148-10.28.0.6-33729	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordC

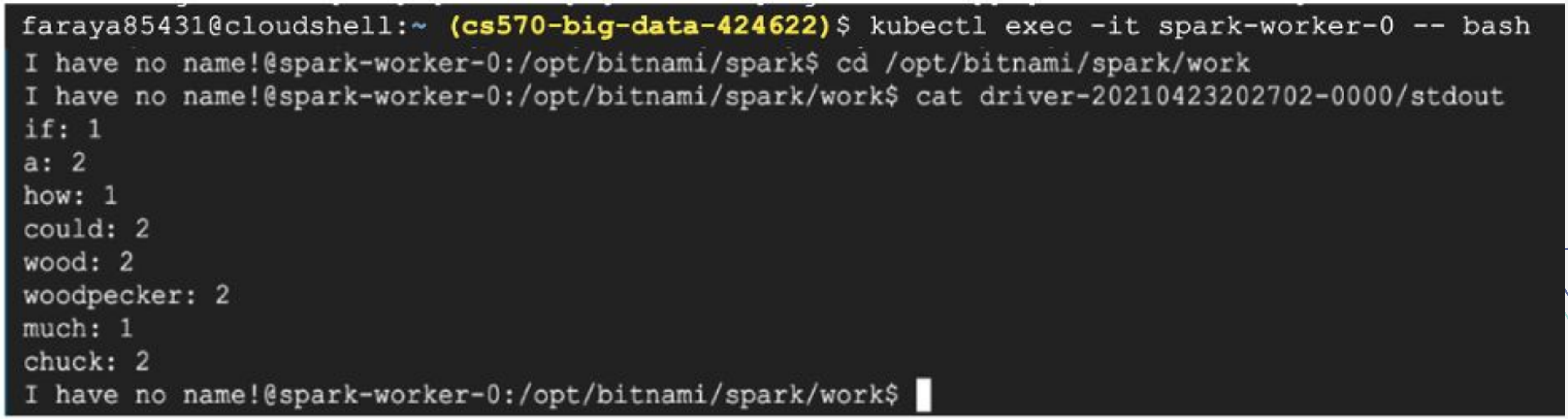
Get the name of the worker node

- kubectl get pods -o wide l grep WORKER-NODE-ADDRESS



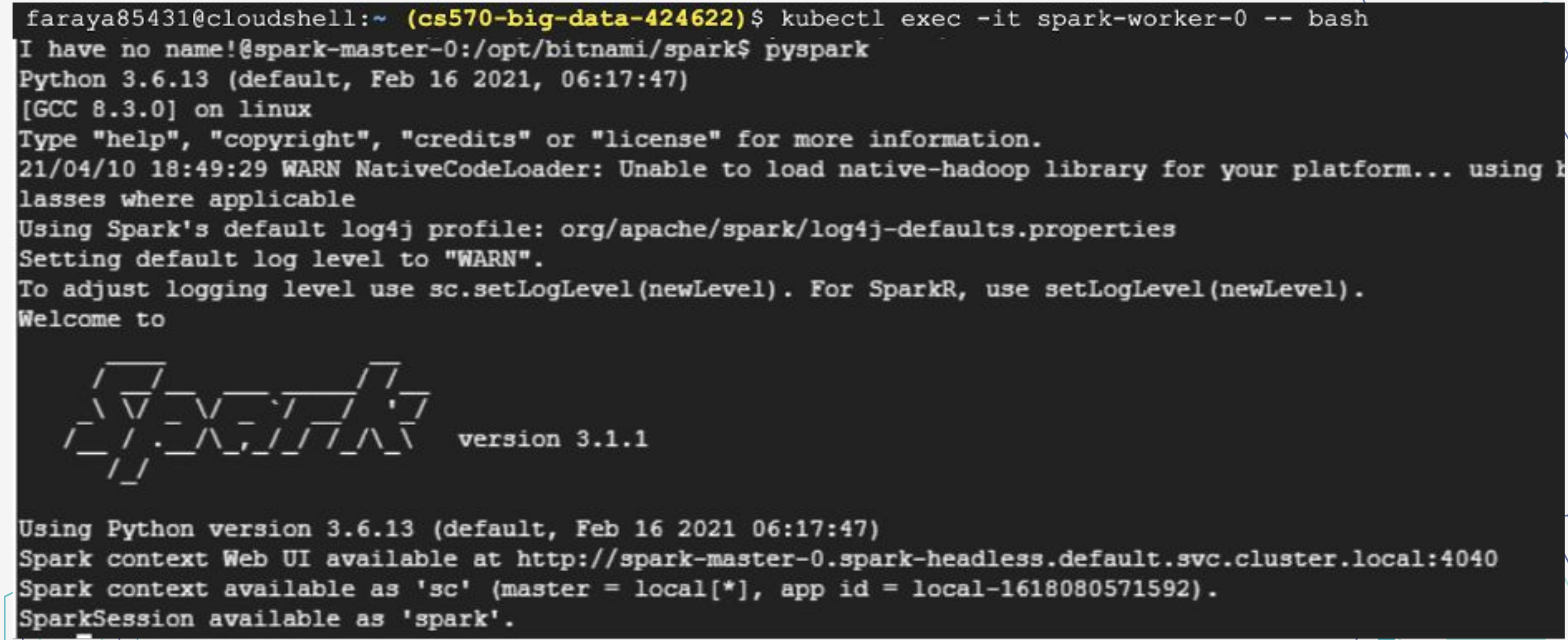
```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl get pods -o wide | grep 10.0.1.7
spark-worker-0                 1/1     Running    0          40m   10.0.1.7    gke-spark-default-pool-bcde3d62-wbpb   <none>           <none>
```

Execute this pod and see the result of the finished tasks

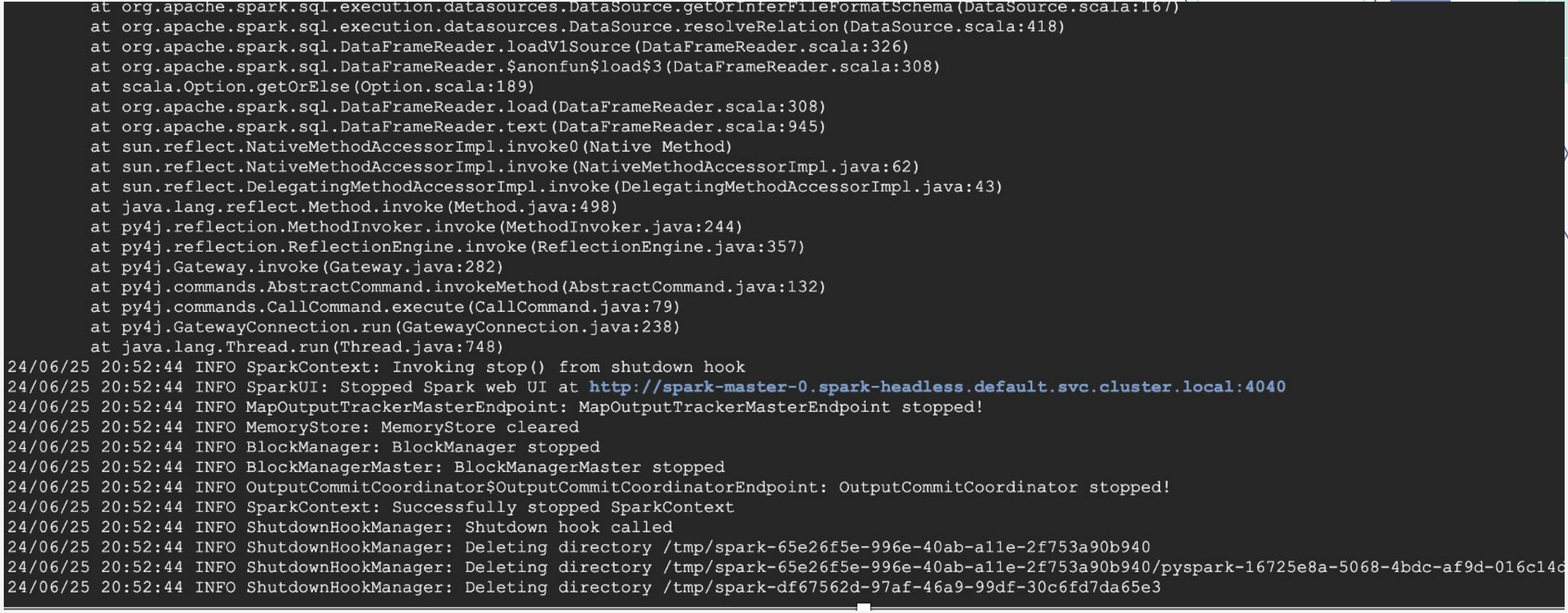


```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl exec -it spark-worker-0 -- bash
I have no name!@spark-worker-0:/opt/bitnami/spark$ cd /opt/bitnami/spark/work
I have no name!@spark-worker-0:/opt/bitnami/spark/work$ cat driver-20210423202702-0000/stdout
if: 1
a: 2
how: 1
could: 2
wood: 2
woodpecker: 2
much: 1
chuck: 2
I have no name!@spark-worker-0:/opt/bitnami/spark/work$
```

Running python PageRank onPySpark on the pods



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ kubectl exec -it spark-worker-0 -- bash
I have no name!@spark-master-0:/opt/bitnami/spark$ pyspark
Python 3.6.13 (default, Feb 16 2021, 06:17:47)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
21/04/10 18:49:29 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
lasses 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
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.1
      /_/

Using Python version 3.6.13 (default, Feb 16 2021 06:17:47)
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-1618080571592).
SparkSession available as 'spark'.
```

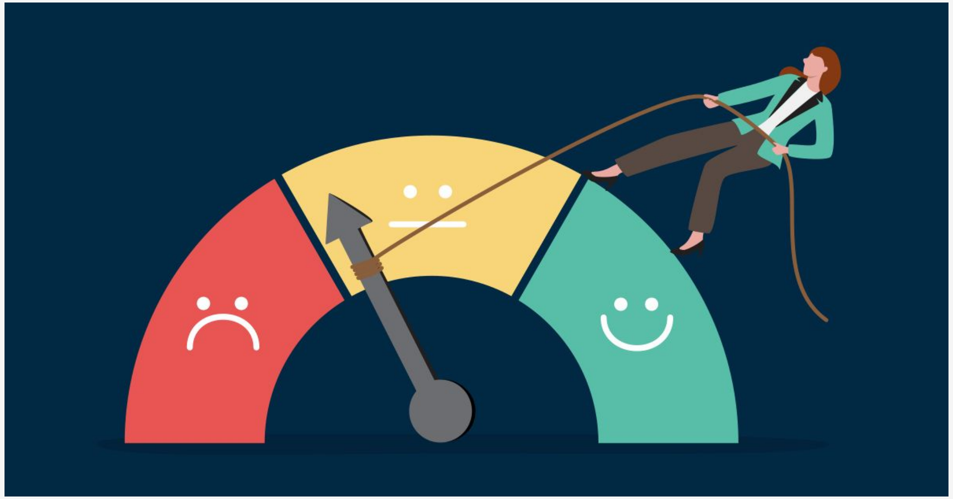


```
        at org.apache.spark.sql.execution.datasources.DataSource.getOrInferFileFormatSchema(DataSource.scala:167)
        at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:418)
        at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:326)
        at org.apache.spark.sql.DataFrameReader.$anonfun$load$3(DataFrameReader.scala:308)
        at scala.Option.getOrElse(Option.scala:189)
        at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:308)
        at org.apache.spark.sql.DataFrameReader.text(DataFrameReader.scala:945)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
        at py4j.Gateway.invoke(Gateway.java:282)
        at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
        at py4j.commands.CallCommand.execute(CallCommand.java:79)
        at py4j.GatewayConnection.run(GatewayConnection.java:238)
        at java.lang.Thread.run(Thread.java:748)
24/06/25 20:52:44 INFO SparkContext: Invoking stop() from shutdown hook
24/06/25 20:52:44 INFO SparkUI: Stopped Spark web UI at http://spark-master-0.spark-headless.default.svc.cluster.local:4040
24/06/25 20:52:44 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
24/06/25 20:52:44 INFO MemoryStore: MemoryStore cleared
24/06/25 20:52:44 INFO BlockManager: BlockManager stopped
24/06/25 20:52:44 INFO BlockManagerMaster: BlockManagerMaster stopped
24/06/25 20:52:44 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
24/06/25 20:52:44 INFO SparkContext: Successfully stopped SparkContext
24/06/25 20:52:44 INFO ShutdownHookManager: Shutdown hook called
24/06/25 20:52:44 INFO ShutdownHookManager: Deleting directory /tmp/spark-65e26f5e-996e-40ab-a11e-2f753a90b940
24/06/25 20:52:44 INFO ShutdownHookManager: Deleting directory /tmp/spark-65e26f5e-996e-40ab-a11e-2f753a90b940/pyspark-16725e8a-5068-4bdc-af9d-016c14d
24/06/25 20:52:44 INFO ShutdownHookManager: Deleting directory /tmp/spark-df67562d-97af-46a9-99df-30c6fd7da65e3
```

Go to the directory where pagerank.py located cd /opt/bitnami/spark/examples/src/main/python
Then Run the pagerank using pyspark spark-submit pagerank.py /opt 2
Here is my output of running the pagerank for directory /opt with 2 iterations

05

Enhancements



Implement dynamic scaling and resource tuning.

Performance Optimization: Dynamic scaling ensures that your application has the resources it needs when the workload increases, thereby preventing performance bottlenecks. During periods of low activity, scaling down reduces resource waste. Fine-tuning the resource requests and limits for Kubernetes pods allows the cluster to manage its resources more efficiently, which can drastically reduce job execution times and increase throughput.

06

Conclusion

Key Outcomes:

- Summary of the results obtained from the Word Count and PageRank computations.
- Discussion on the performance metrics, noting the efficiency and scalability achieved through Kubernetes.

Lessons Learned:

- Insights into the challenges faced during the implementation and how they were overcome.
- Best practices derived from deploying PySpark applications on Kubernetes.

07

References

Apache Spark with Python - Big Data withPySpark and Spark

Exciting Spark Project Ideas & Topics For Beginners

Thanks!