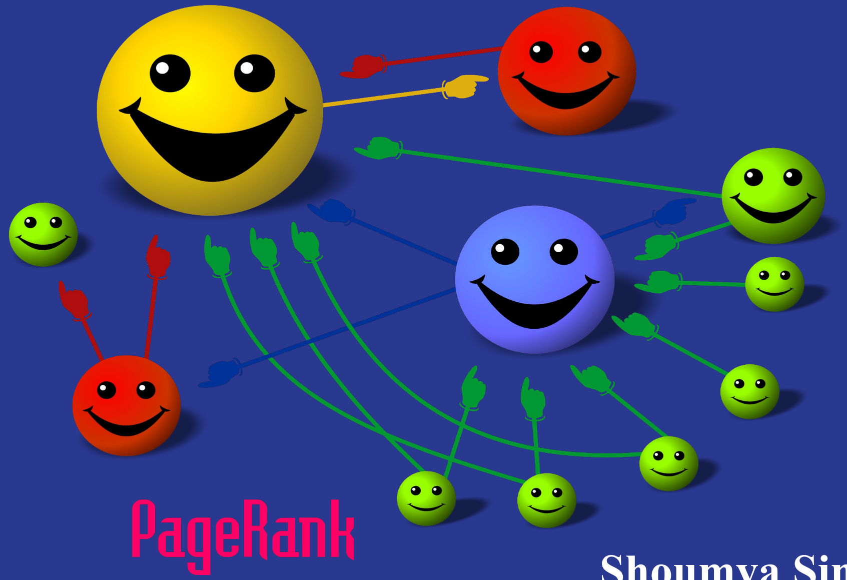
Wordcount + PageRank + Apache Spark + GKE

Kubernetes Project



Shoumya Singh

Table of Content

Introduction

- ❑ Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
- ❑ Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more.
- ❑ Apache Spark with native Kubernetes support combines the best of the two prominent open source projects — Apache Spark, a framework for large-scale data processing; and Kubernetes.
- ❑ PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing your data in a distributed environment.

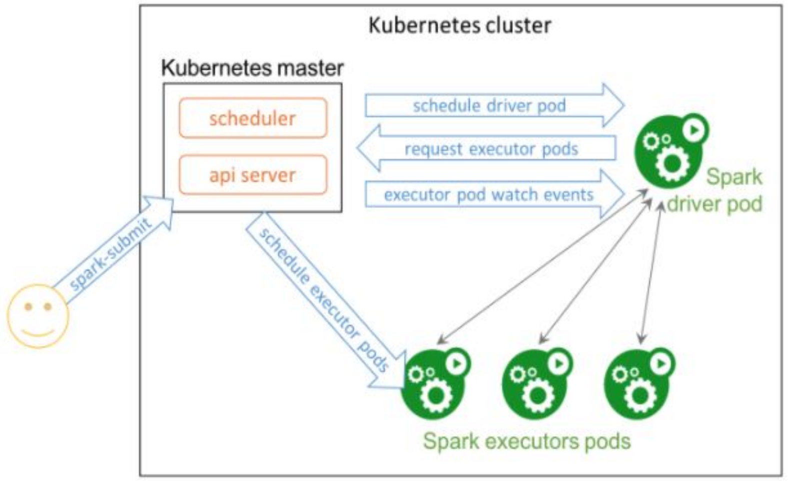
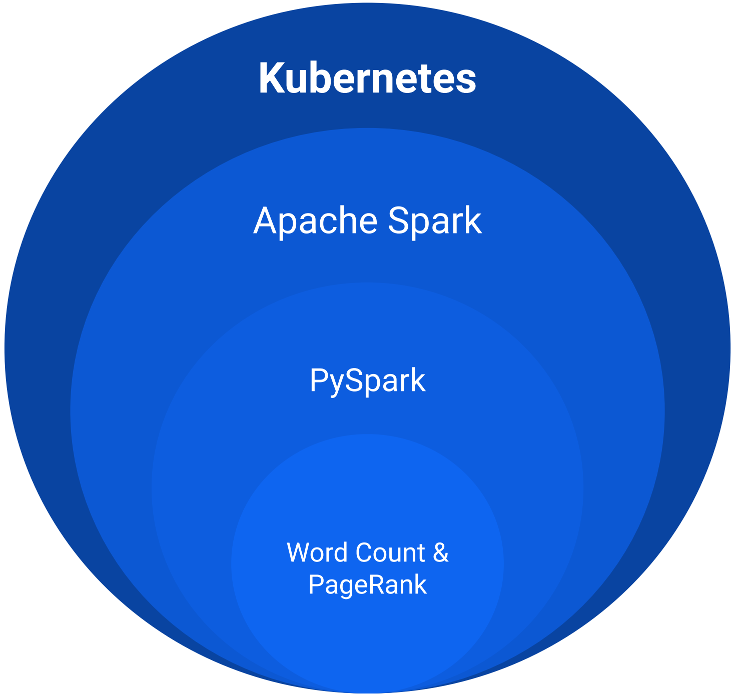


Table of Content

Project Description

➔ **Using PySpark to implement Word Count and PageRank on Apache Spark running on Kubernetes.**



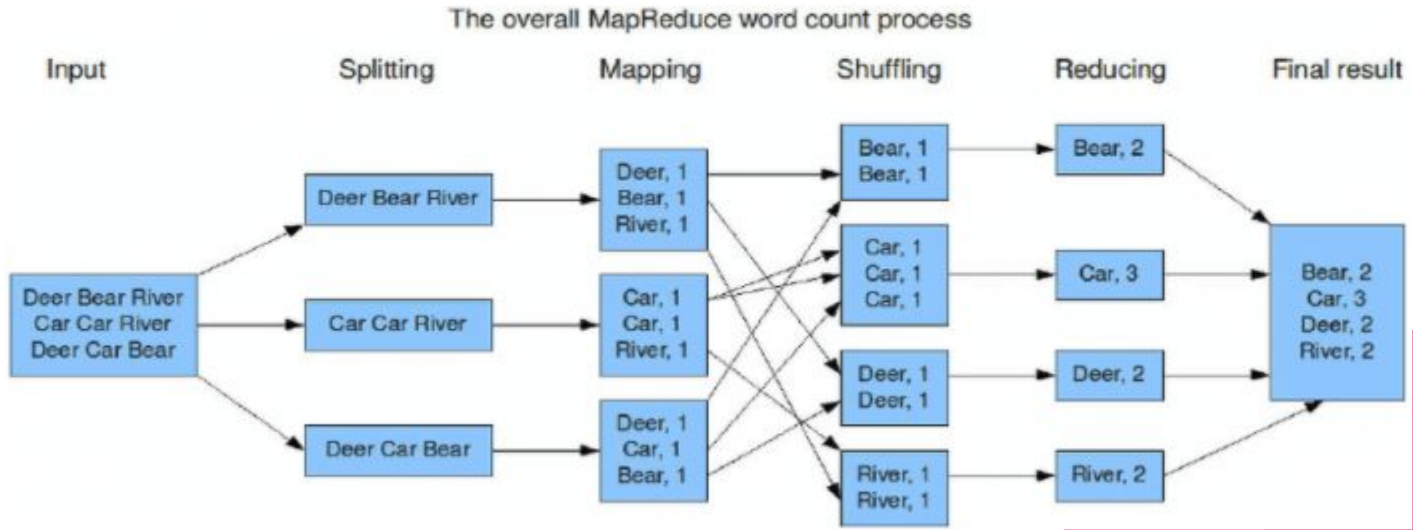
➔ **Technologies Used**
- Google Kubernetes Engine
- Apache Spark - PySpark

➔ **Applications**
- Wordcount
- PageRank

Table of Content

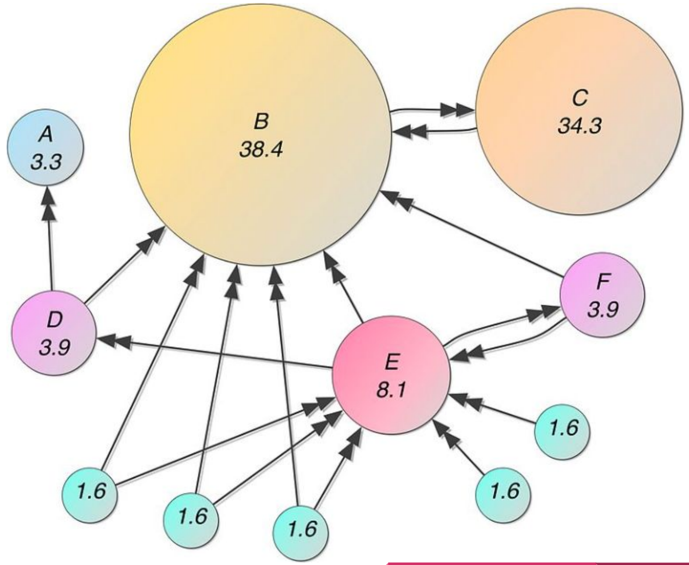
Wordcount

- ❑ **Wordcount** is a simple spark application where it counts how often each word appears in a collection of text documents.
- ❑ A distributed computing framework that can run WordCount efficiently in parallel at scale can likely handle much larger and more interesting compute problems.
- ❑ **MapReduce** is a processing technique and a program model for distributed computing based on java. The **MapReduce** algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

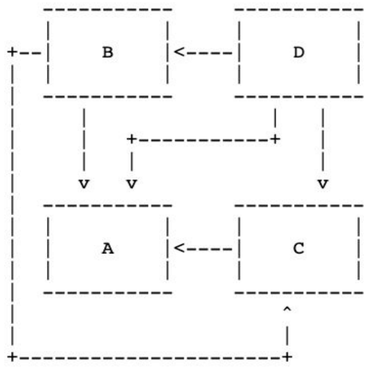


PageRank

- PageRank is one of many algorithms Google uses to work out which order to display search results.
- The co-founders of Google, Sergey Brin and Larry Page developed the PageRank algorithm in 1996 at Stanford University.
- Increasing the PageRank score of a web page will mean that page is displayed higher than other pages in a search engine listing, which means more visitors and therefore potentially more customers or money generated from a web page.
- The formula used to calculate PageRank is:
- $PR(A) = (1 - d) + d\,(PR(t1) / C(t1) + \ldots + PR(tn)/C(tn))$
- PageRank is an iterative algorithm which means you repeat the calculation for each page multiple times until the values eventually settle on the final PageRank scores for each page.



PageRank Calculation Example



- ➔ The initial PageRank value for each webpage is 1.
  - ■ PR(A) = 1
  - ■ PR(B) = 1
  - ■ PR(C) = 1
- ➔ Page B has a link to pages C and A
- ➔ Page C has a link to page A
- ➔ Page D has links to all three pages

1. A's PageRank is
   PR(A) = (1-d) + d * (PR(B) / 2 + PR(C) / 1 + PR(D) / 3)
2. B's PageRank is
   PR(B) = (1-d) + d * (PR(D) / 3)
3. C's PageRank is
   PR(C) = (1-d) + d * (PR(B) / 2 + PR(D) / 3)
4. C's PageRank is
   PR(D) = 1-d
5. Damping factor is 0.85 (default value)

PageRank Calculation Example

Then after 1st iteration

A. Output

a. Page B would transfer half of its existing value, or 0.5, to page A and the other half, or 0.5, to page C.

b. Page C would transfer all of its existing value, 1, to the only page it links to, A.

c. Since D had three outbound links, it would transfer one third of its existing value, or approximately 0.33, to A.

B. Input

a. PR(A)
= (1-d) + d * (PR(B) / 2 + PR(C) / 1 + PR(D) / 3)
= (1-0.85) + 0.85 * (0.5 + 1 + 0.33)
= 1.71

b. PR(B)
= (1-d) + d * (PR(D) / 3)
= (1-0.85) + 0.85 * 0.33
= 0.43

c. PR(C)
= (1-d) + d * (PR(B) / 2 + PR(D) / 3)
= (1-0.85) + 0.85 * (0.5 + 0.33)
= 0.86

d. PR(D)
= 1-d
= 0.15

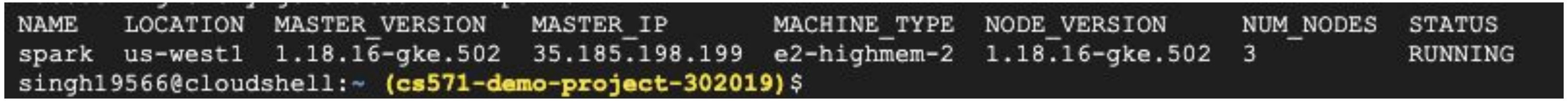
C. Observation of PageRank

- **The more inputs a node has the more its PageRank will increase in the long run.**
- **A node does not have input will have**
  - **constant PageRank: 1-d**
  - **the smallest PageRank**

Table of Content

Implementation

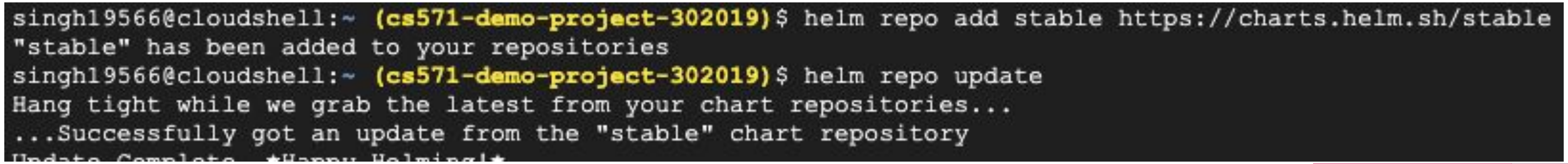
1. **Create a cluster on Google Kubernetes Engine with**
   - gcloud container clusters create spark --num-nodes=1 --machine- type=e2-highmem-2 --region=us-west1



```
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)$
```

2. **Create image and deploy spark to Kubernetes**

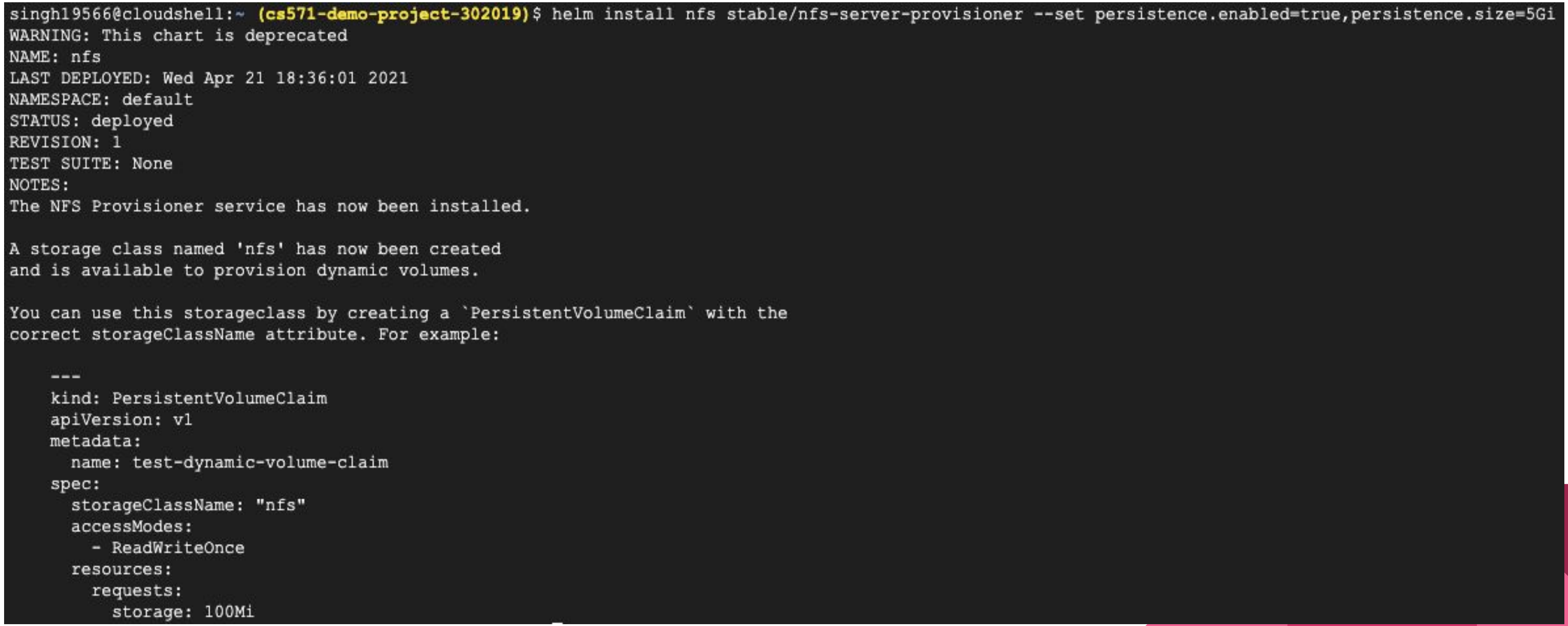
- Install the NFS Server Provisioner
  - helm repo add stable https://charts.helm.sh/stable
  - helm repo update



```
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
```

Implementation

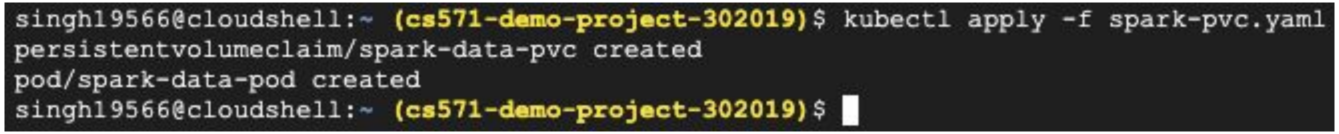
- helm install nfs stable/nfs-server-provisioner\
- set persistence.enabled=true,persistence.size=5Gi



```
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
```

Implementation

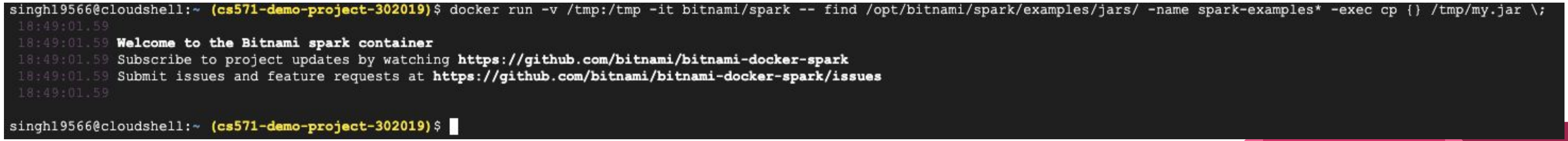
3. Create a persistent disk volume and a pod to use NFS spark-pvc.yaml:
   - kubectl apply -f spark-pvc.yaml



```
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)$
```

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 \;

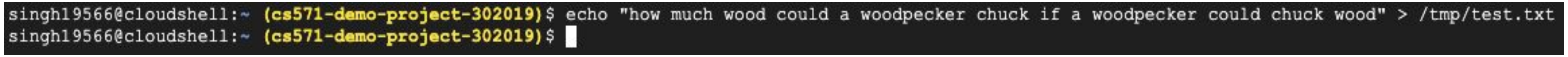
   After running the above command, you should see this



```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples* -exec cp {} /tmp/my.jar \;
 18:49:01.59
 18:49:01.59 Welcome to the Bitnami spark container
 18:49:01.59 Subscribe to project updates by watching https://github.com/bitnami/bitnami-docker-spark
 18:49:01.59 Submit issues and feature requests at https://github.com/bitnami/bitnami-docker-spark/issues
 18:49:01.59

singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

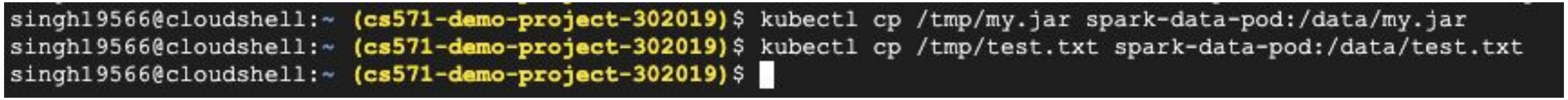
Implementation

5. 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



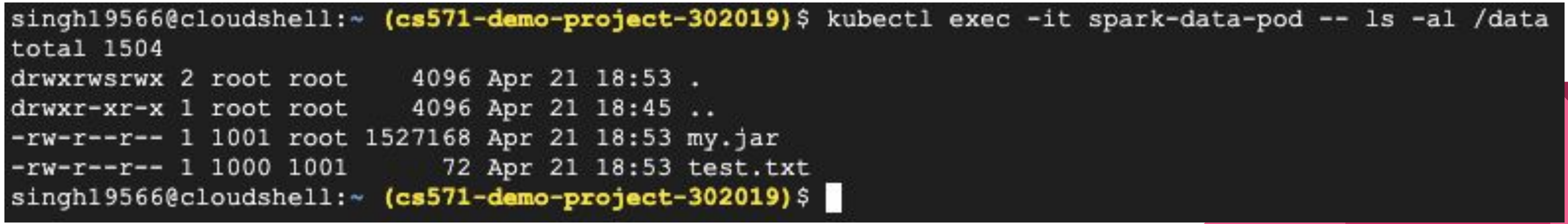
```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ echo "how much wood could a woodpecker chuck if a woodpecker could chuck wood" > /tmp/test.txt
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

6. 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



```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl cp /tmp/test.txt spark-data-pod:/data/test.txt
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

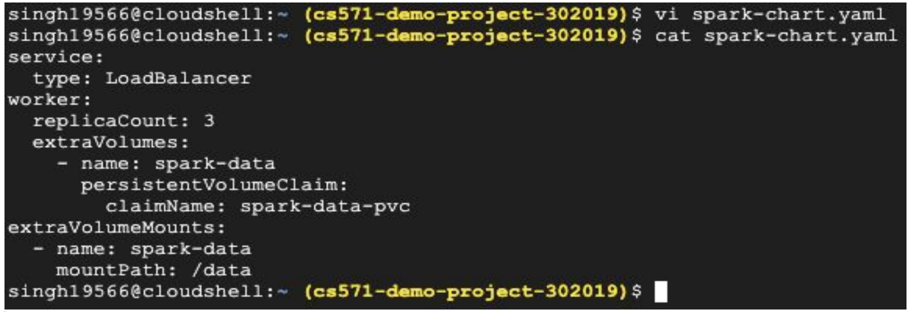
7. Make sure the files a inside the persistent volume
   - kubectl exec -it spark-data-pod -- ls -al /data



```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl exec -it spark-data-pod -- ls -al /data
total 1504
drwxrwsrwx 2 root root    4096 Apr 21 18:53 .
drwxr-xr-x 1 root root    4096 Apr 21 18:45 ..
-rw-r--r-- 1 1001 root 1527168 Apr 21 18:53 my.jar
-rw-r--r-- 1 1000 1001      72 Apr 21 18:53 test.txt
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

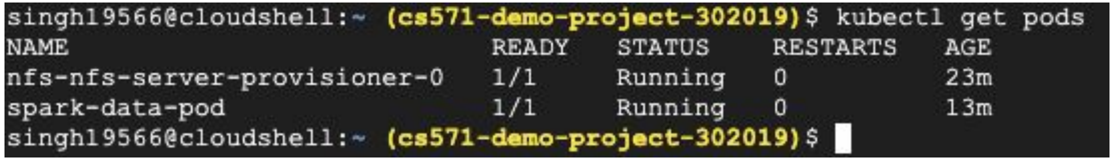
Implementation

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



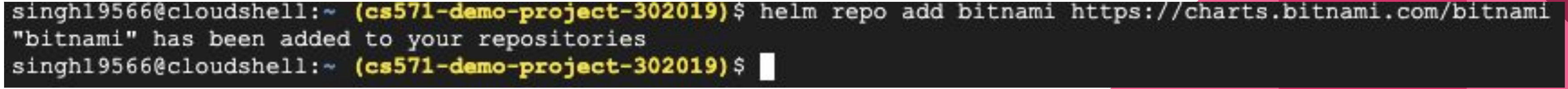
```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ vi spark-chart.yaml
singh19566@cloudshell:~ (cs571-demo-project-302019)$ cat spark-chart.yaml
service:
  type: LoadBalancer
worker:
  replicaCount: 3
  extraVolumes:
    - name: spark-data
      persistentVolumeClaim:
        claimName: spark-data-pvc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

9. 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)$
```

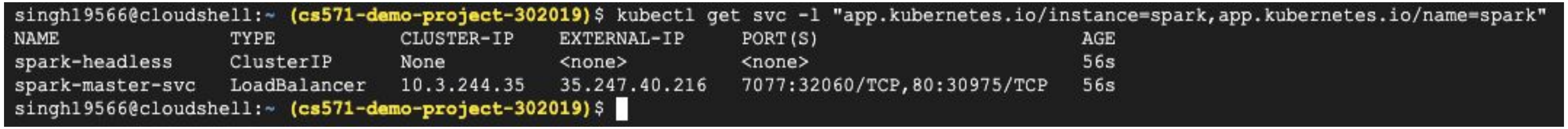
10. Deploy Apache Spark on the Kubernetes cluster using the Bitnami Apache Spark Helm chart and supply it with the configuration file above
    - helm repo add bitnami https://charts.bitnami.com/bitnami
    - helm install spark bitnami/spark -f spark-chart.yaml



```
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)$
```

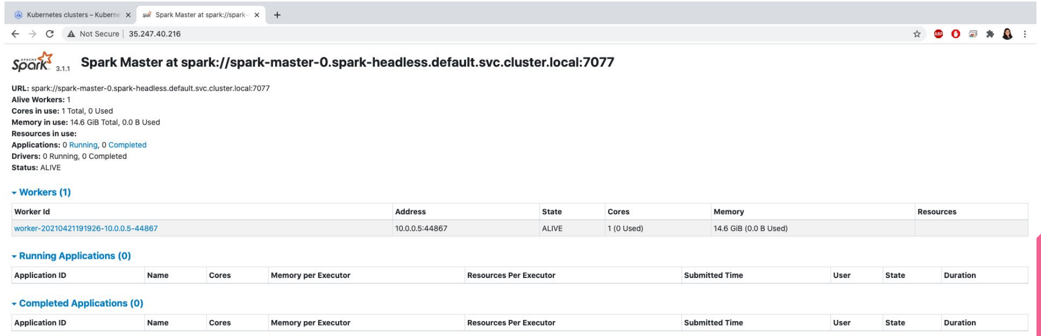
Implementation

11. 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)$
```

12. Open the external ip on your 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: 1
Cores in use: 1 Total, 0 Used
Memory in use: 14.6 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210421191926-10.0.0.5-44867	10.0.0.5:44867	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

Completed Applications (0)

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

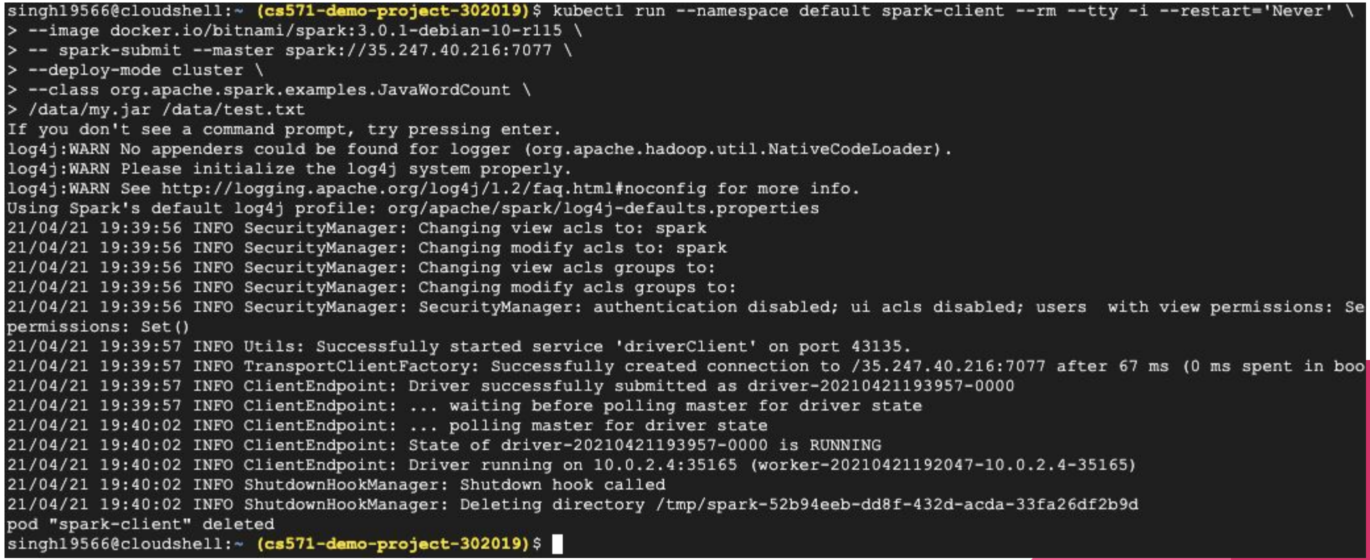
Table of Content

Test Result - Wordcount 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

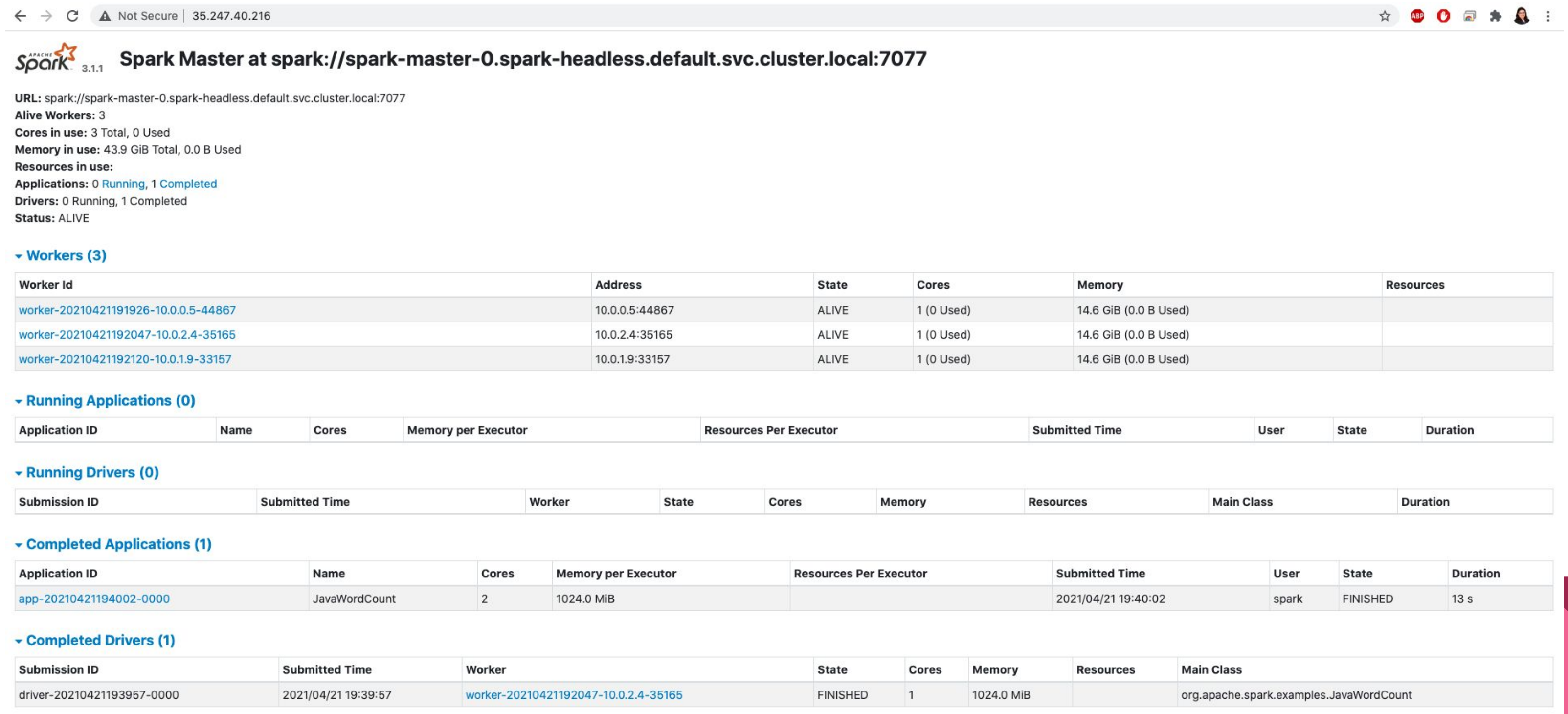
You should see something like this after the above command



```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ 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://35.247.40.216:7077 \
> --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
21/04/21 19:39:56 INFO SecurityManager: Changing view acls to: spark
21/04/21 19:39:56 INFO SecurityManager: Changing modify acls to: spark
21/04/21 19:39:56 INFO SecurityManager: Changing view acls groups to:
21/04/21 19:39:56 INFO SecurityManager: Changing modify acls groups to:
21/04/21 19:39:56 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Se
permissions: Set()
21/04/21 19:39:57 INFO Utils: Successfully started service 'driverClient' on port 43135.
21/04/21 19:39:57 INFO TransportClientFactory: Successfully created connection to /35.247.40.216:7077 after 67 ms (0 ms spent in boo
21/04/21 19:39:57 INFO ClientEndpoint: Driver successfully submitted as driver-20210421193957-0000
21/04/21 19:39:57 INFO ClientEndpoint: ... waiting before polling master for driver state
21/04/21 19:40:02 INFO ClientEndpoint: ... polling master for driver state
21/04/21 19:40:02 INFO ClientEndpoint: State of driver-20210421193957-0000 is RUNNING
21/04/21 19:40:02 INFO ClientEndpoint: Driver running on 10.0.2.4:35165 (worker-20210421192047-10.0.2.4-35165)
21/04/21 19:40:02 INFO ShutdownHookManager: Shutdown hook called
21/04/21 19:40:02 INFO ShutdownHookManager: Deleting directory /tmp/spark-52b94eeb-dd8f-432d-acda-33fa26df2b9d
pod "spark-client" deleted
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

Test Result - Wordcount on Spark

➔ **And on your browser, you should see this task finished**



Not Secure | 35.247.40.216

Spark 3.1.1 **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: 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-20210421191926-10.0.0.5-44867	10.0.0.5:44867	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	
worker-20210421192047-10.0.2.4-35165	10.0.2.4:35165	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	
worker-20210421192120-10.0.1.9-33157	10.0.1.9:33157	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)

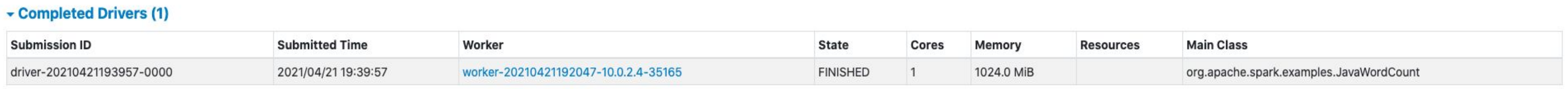
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210421194002-0000	JavaWordCount	2	1024.0 MiB		2021/04/21 19:40:02	spark	FINISHED	13 s

Completed Drivers (1)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20210421193957-0000	2021/04/21 19:39:57	worker-20210421192047-10.0.2.4-35165	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordCount

Test Result - Wordcount on Spark

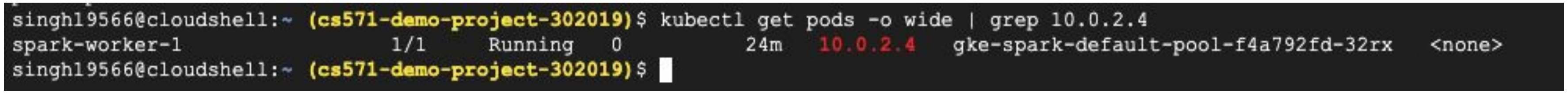
- **View the output of the completed jobs**
  - On the browser, you should see the worker node ip address of the finished task



▾ Completed Drivers (1)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20210421193957-0000	2021/04/21 19:39:57	worker-20210421192047-10.0.2.4-35165	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordCount

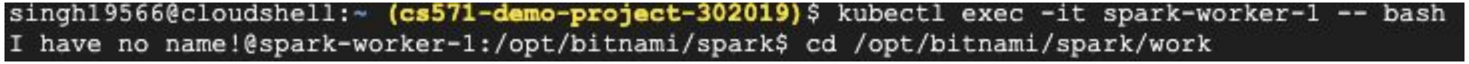
- For example, my worker node ip address is 10.0.2.4
  - Get the name of the worker node
    - kubectl get pods -o wide | grep WORKER-NODE-ADDRESS
    - kubectl get pods -o wide | grep 10.0.2.4



```
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)$ 
```

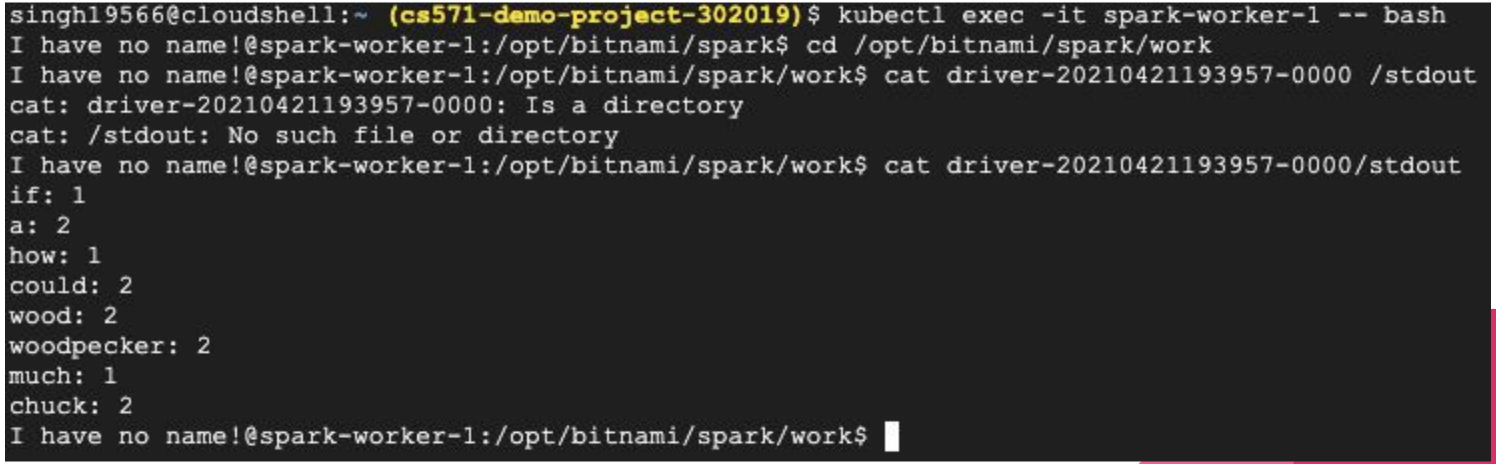
Test Result - Wordcount on Spark

➔ **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
```

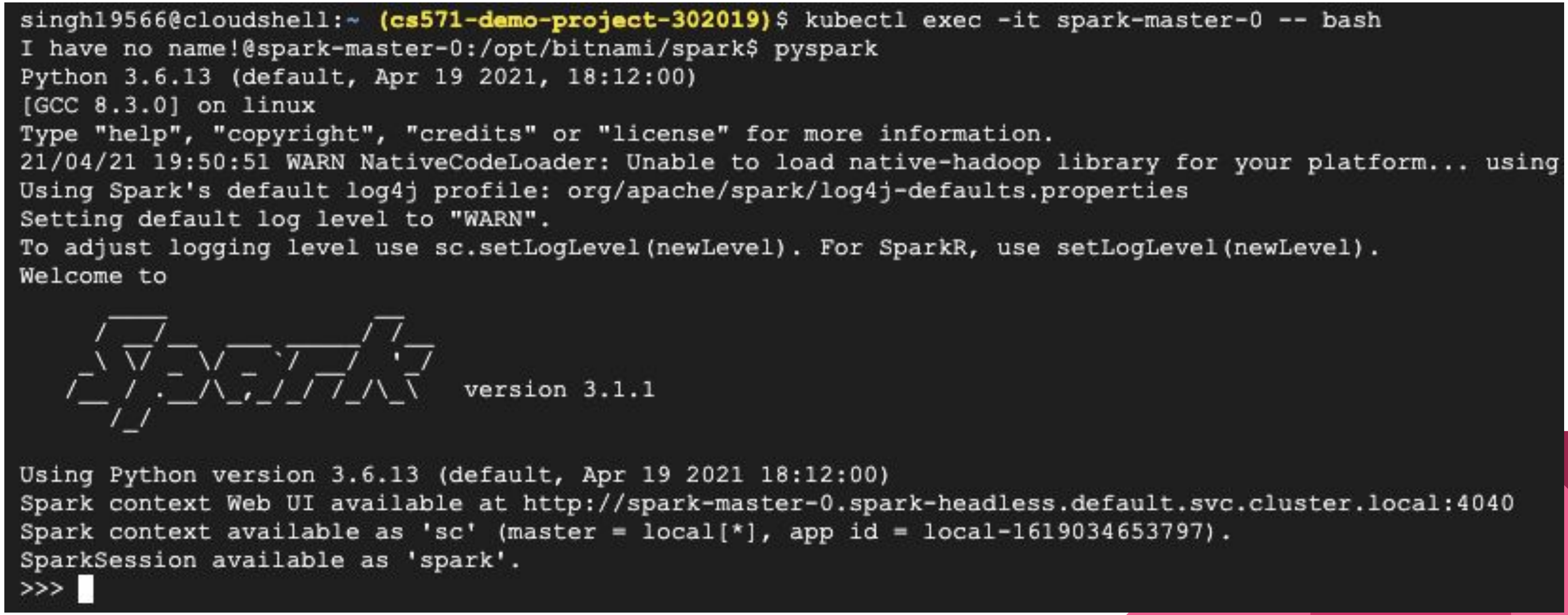
- cd /opt/bitnami/spark/work
- cat <taskname>/stdout



```
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$
```

Test Result - PageRank on PySpark on the pods

- ➔ **Execute the spark master pods**
  - kubectl exec -it spark-master-0 -- bash
- ➔ **Stark pyspark**
  - pyspark

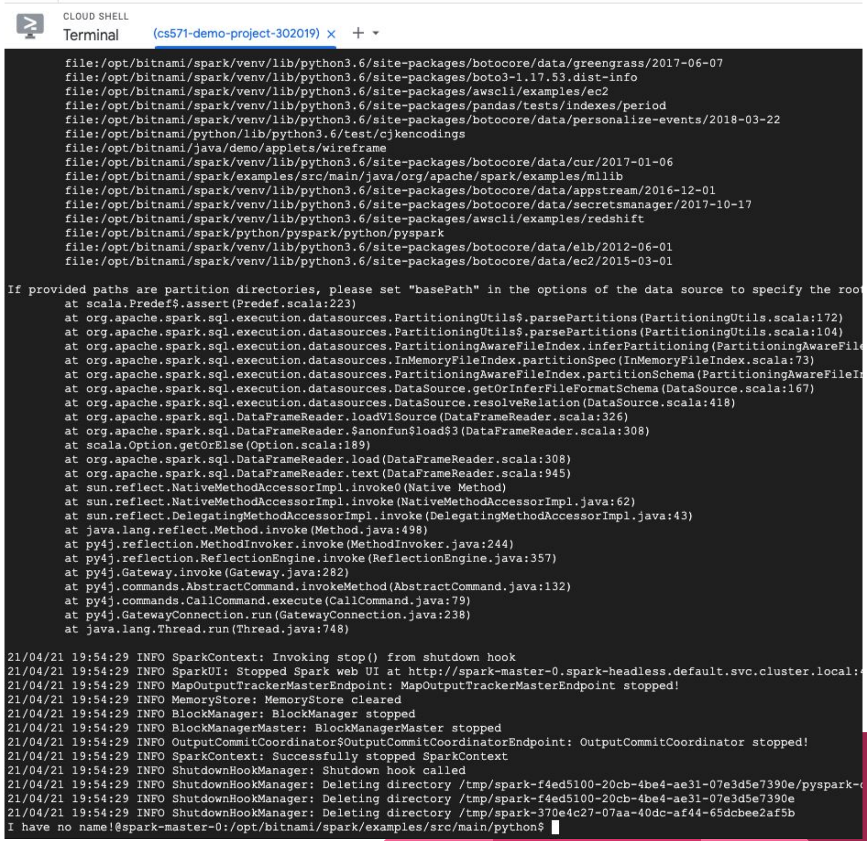


```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ kubectl exec -it spark-master-0 -- bash
I have no name!@spark-master-0:/opt/bitnami/spark$ pyspark
Python 3.6.13 (default, Apr 19 2021, 18:12:00)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
21/04/21 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'.
>>> 
```

Test Result - PageRank on PySpark on the pods

- **Exit pyspark with**
  - exit()
- **Go to the directory where pagerank.py located**
  - cd/opt/bitnami/spark/examples/src/main/python
- **Run the page rank using pyspark**
  - spark-submit pagerank.py /opt 2

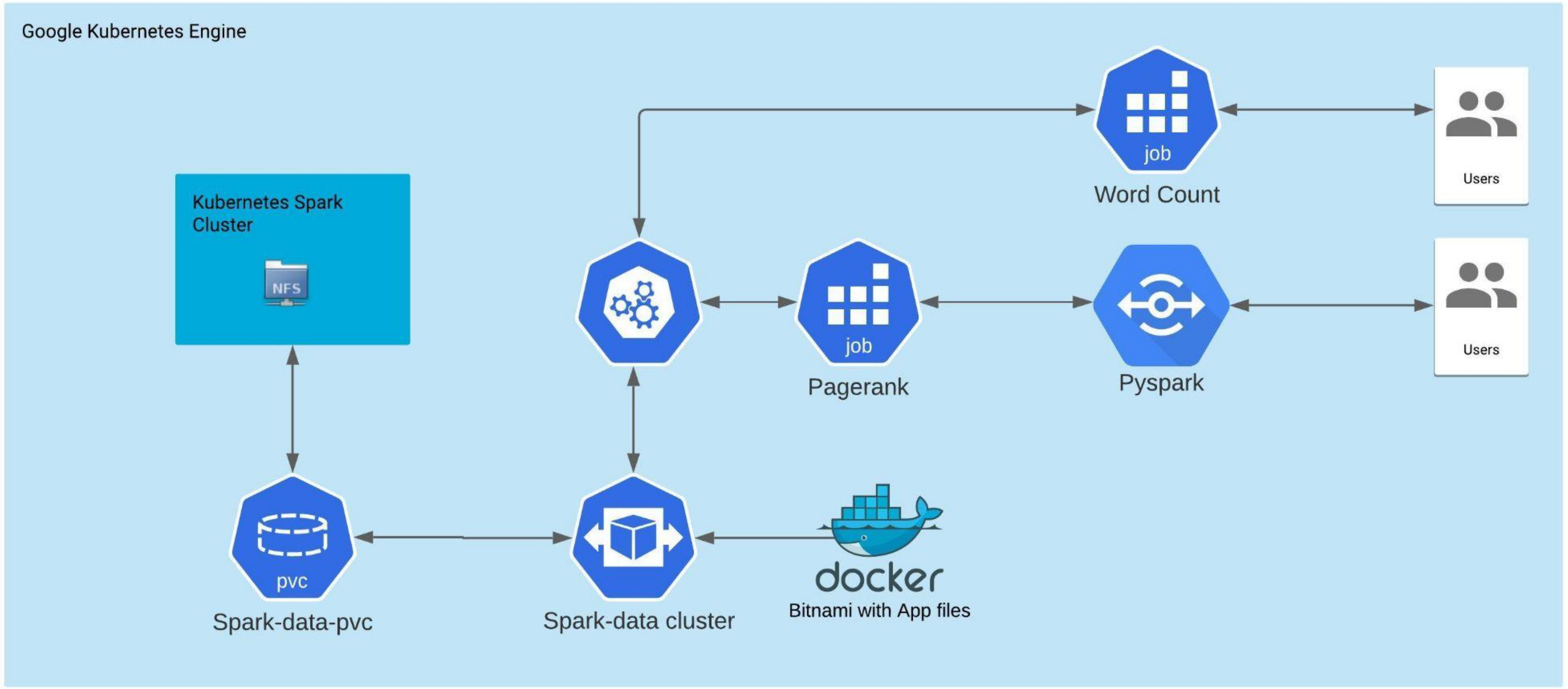
Note, /opt is an example directory and 2 is the number of iterations you want the page rank to run, we can also change to any numbers, here is my output of running the page rank for directory /opt with 2 iterations


CLOUD SHELL Terminal (cs571-demo-project-302019)

```
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/greengrass/2017-06-07
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/boto3-1.17.53.dist-info
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/awscli/examples/ec2
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/pandas/tests/indexes/period
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/personalize-events/2018-03-22
        file:/opt/bitnami/python/lib/python3.6/test/cjkencodings
        file:/opt/bitnami/java/demo/applets/wireframe
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/cur/2017-01-06
        file:/opt/bitnami/spark/examples/src/main/java/org/apache/spark/examples/mllib
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/appstream/2016-12-01
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/secretsmanager/2017-10-17
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/awscli/examples/redshift
        file:/opt/bitnami/spark/python/pyspark/python/pyspark
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/elb/2012-06-01
        file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/ec2/2015-03-01

If provided paths are partition directories, please set "basePath" in the options of the data source to specify the root
        at scala.Predef$.assert(Predef.scala:223)
        at org.apache.spark.sql.execution.datasources.PartitioningUtils$.parsePartitions(PartitioningUtils.scala:172)
        at org.apache.spark.sql.execution.datasources.PartitioningUtils$.parsePartitions(PartitioningUtils.scala:104)
        at org.apache.spark.sql.execution.datasources.PartitioningAwareFileIndex.inferPartitioning(PartitioningAwareFile
        at org.apache.spark.sql.execution.datasources.InMemoryFileIndex.partitionSpec(InMemoryFileIndex.scala:73)
        at org.apache.spark.sql.execution.datasources.PartitioningAwareFileIndex.partitionSchema(PartitioningAwareFileI
        at org.apache.spark.sql.execution.datasources.DataSource.getOrInferFileFormatSchema(DataSource.scala:167)
        at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:418)
        at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:326)
        at org.apache.spark.sql.DataFrameReader.$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)

21/04/21 19:54:29 INFO SparkContext: Invoking stop() from shutdown hook
21/04/21 19:54:29 INFO SparkUI: Stopped Spark web UI at http://spark-master-0.spark-headless.default.svc.cluster.local:
21/04/21 19:54:29 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/04/21 19:54:29 INFO MemoryStore: MemoryStore cleared
21/04/21 19:54:29 INFO BlockManager: BlockManager stopped
21/04/21 19:54:29 INFO BlockManagerMaster: BlockManagerMaster stopped
21/04/21 19:54:29 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/04/21 19:54:29 INFO SparkContext: Successfully stopped SparkContext
21/04/21 19:54:29 INFO ShutdownHookManager: Shutdown hook called
21/04/21 19:54:29 INFO ShutdownHookManager: Deleting directory /tmp/spark-f4ed5100-20cb-4be4-ae31-07e3d5e7390e/pyspark-
21/04/21 19:54:29 INFO ShutdownHookManager: Deleting directory /tmp/spark-f4ed5100-20cb-4be4-ae31-07e3d5e7390e
21/04/21 19:54:29 INFO ShutdownHookManager: Deleting directory /tmp/spark-370e4c27-07aa-40dc-af44-65dcbee2af5b
I have no name!@spark-master-0:/opt/bitnami/spark/examples/src/main/python$
```

Table of Content

Schematic of Project



Conclusion

- ★ Apache Spark is a cluster computing platform designed to be fast, speed side and extends the popular MapReduce model to efficiently supports more type of computations, including interactive queries and stream processing.
- ★ Since Spark integrates closely with other big data tool, hence this tight integration is the ability to build an application that seamlessly combines different computation model.
- ★ Spark is also highly fault-tolerant; if one node fails, the failed tasks are distributed across the other nodes.
- ★ **The entire project is developed in Google Cloud Platform which provides different tools from which we are using Google Kubernetes Engine(GKE).**
- ★ **Then using PySpark API we implemented Word Count and PageRank applications on Apache Spark running on Google Kubernetes Engine.**

Bibliography/References

- https://npu85.npu.edu/~henry/npu/classes/mapreduce/word_count/slide/wordcount_explain.html
- https://kubernetes.io/blog/2018/03/apache-spark-23-with-native-kubernetes/
- https://databricks.com/blog/2018/03/06/apache-spark-2-3-with-native-kubernetes-support.html
- https://spark.apache.org/docs/latest/api/python/index.html
- https://npu85.npu.edu/~henry/npu/classes/learning_spark/key_value_pair/slide/Example_PageRank.html
- https://www.sciencedirect.com/topics/computer-science/apache-spark
- https://stanford.edu/~rezab/sparkclass/slides/itas_workshop.pdf