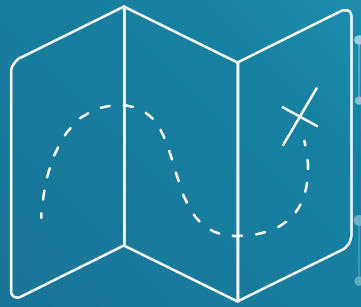
PySpark on Kubernetes: Word Count + PageRank

Belsabel Woldemichael

Contents



Introduction

Objective: Implement two fundamental big data algorithms, Word Count and PageRank, using PySpark on Apache Spark deployed on Kubernetes. This project aims to demonstrate proficiency in distributed computing and analytics using scalable technologies.

Technologies Used:

- Apache Spark: Distributed computing framework for processing large-scale data sets.
- PySpark: Python API for Apache Spark, enabling data manipulation and algorithm implementation.
- Kubernetes: Container orchestration platform for deploying and managing Spark clusters in a scalable and resilient manner.

Design

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			

Page Rank

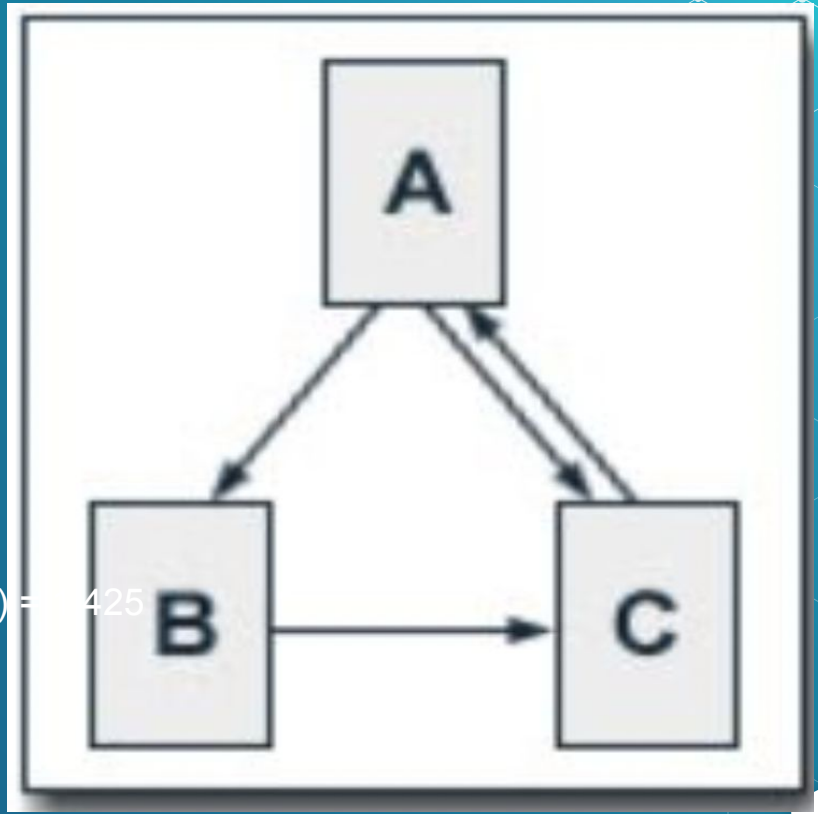
The following is the manual calculation of the diagram below.

Webpage A links to B and C.
Webpage B links to C.
Webpage C links back to A.

Initial Setup:
Each webpage starts with a PageRank value of 1.
Damping factor (d) = 0.85.

First Iteration:
PR(A) = 1−d+d×(PR(C)/1) = 1−0.85+0.85×1 = 1
PR(B) = 1−d+d×(PR(A)/2) = 1−0.85+0.85×1/2 = 0.575
PR(C) = 1−d+d×((PR(A)/2)+PR(B)/1) = 1−0.85+0.85×(0.5+ 1) = 1.425

Second Iteration:
PageRank (A) = 1 – 0.85 + 0.85 * 1.425 = 1.36125
PageRank (B) = 1 – 0.85 + 0.85 * 0.5 = 0.575
PageRank (C) = 1 – 0.85 + 0.85 * 1.075 = 1.0637



Implementation and Test

1. Create a cluster on GKE with
gcloud container clusters create spark --num-nodes=1 --machinetype=e2-highmem-2 --region=us-west1

```
Created [https://container.googleapis.com/v1/projects/myfinalproject-426304/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=myfinalproject-426304
kubeconfig entry generated for spark.
NAME: spark
LOCATION: us-west1
MASTER_VERSION: 1.29.4-gke.1043002
MASTER_IP: 104.198.13.218
MACHINE_TYPE: e2-highmem-2
NODE_VERSION: 1.29.4-gke.1043002
NUM_NODES: 3
STATUS: RUNNING
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$
```

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

```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ helm repo add stable https://charts.helm.sh/stable
"stable" already exists with the same configuration, skipping
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ 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!⎈
```

Create image and deploy spark to kubernetes

3. Install the NFS Server Provisioner

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

```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ helm install nfs-provisioner stable/nfs-server-provisioner \
  --set persistence.enabled=true,persistence.size=5Gi
WARNING: This chart is deprecated
NAME: nfs-provisioner
LAST DEPLOYED: Thu Jun 27 18:44:43 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
```

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

```
belsabelteklemariam@cloudshell:~ (cs570-427815)$ cat spark-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
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

belsabelteklemariam@cloudshell:~ (cs570-427815)$
```

4. Apply the above yaml descriptor

- kubectl apply -f spark-pvc.yaml

```
belsabelteklemariam@cloudshell:~ (cs570-427815)$ 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 \;
```

```
-bash: /opt/bitnami/spark/examples/jars/: No such file or directory
belsabelteklemariam@cloudshell:~ (cs570-427815)$ docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples* -exec cp {} /tmp/my.jar \;

spark 15:36:34.46 INFO  ==>
spark 15:36:34.47 INFO  ==> Welcome to the Bitnami spark container
spark 15:36:34.47 INFO  ==> Subscribe to project updates by watching https://github.com/bitnami/containers
spark 15:36:34.47 INFO  ==> Submit issues and feature requests at https://github.com/bitnami/containers/issues
spark 15:36:34.47 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 15:36:34.48 INFO  ==>

belsabelteklemariam@cloudshell:~ (cs570-427815)$
```

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

```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ echo "how much wood could a woodpecker chuck if a woodpecker could  chuck wood" > /tmp/test.txt
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$
```

```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ cat /tmp/test.txt
how much wood could a woodpecker chuck if a woodpecker could  chuck wood
```

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

```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$  kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ 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

```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ kubectl exec -it spark-data-pod -- ls -al /data
total 1540
drwxrwsrwx 2 root root    4096 Jun 27 20:01 .
drwxr-xr-x 1 root root    4096 Jun 27 19:35 ..
-rw-r--r-- 1 1001 root 1564260 Jun 27 20:01 my.jar
-rw-rw-r-- 1 1000 1000      74 Jun 27 20:01 test.txt
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$
```

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

```
  GNU nano 6.2                          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

```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ kubectl get pods
NAME                                            READY   STATUS    RESTARTS   AGE
nfs-provisioner-nfs-server-provisioner-0        1/1     Running   0          87m
spark-data-pod                                  1/1     Running   0          36m
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$
```

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

```
spark-data-pod                                  1/1     Running helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositoriesect-426304)$ helm repo add bitnami https://charts.bitnami.com/bitnami
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$
```

- helm install spark bitnami/spark -f spark-chart.yaml

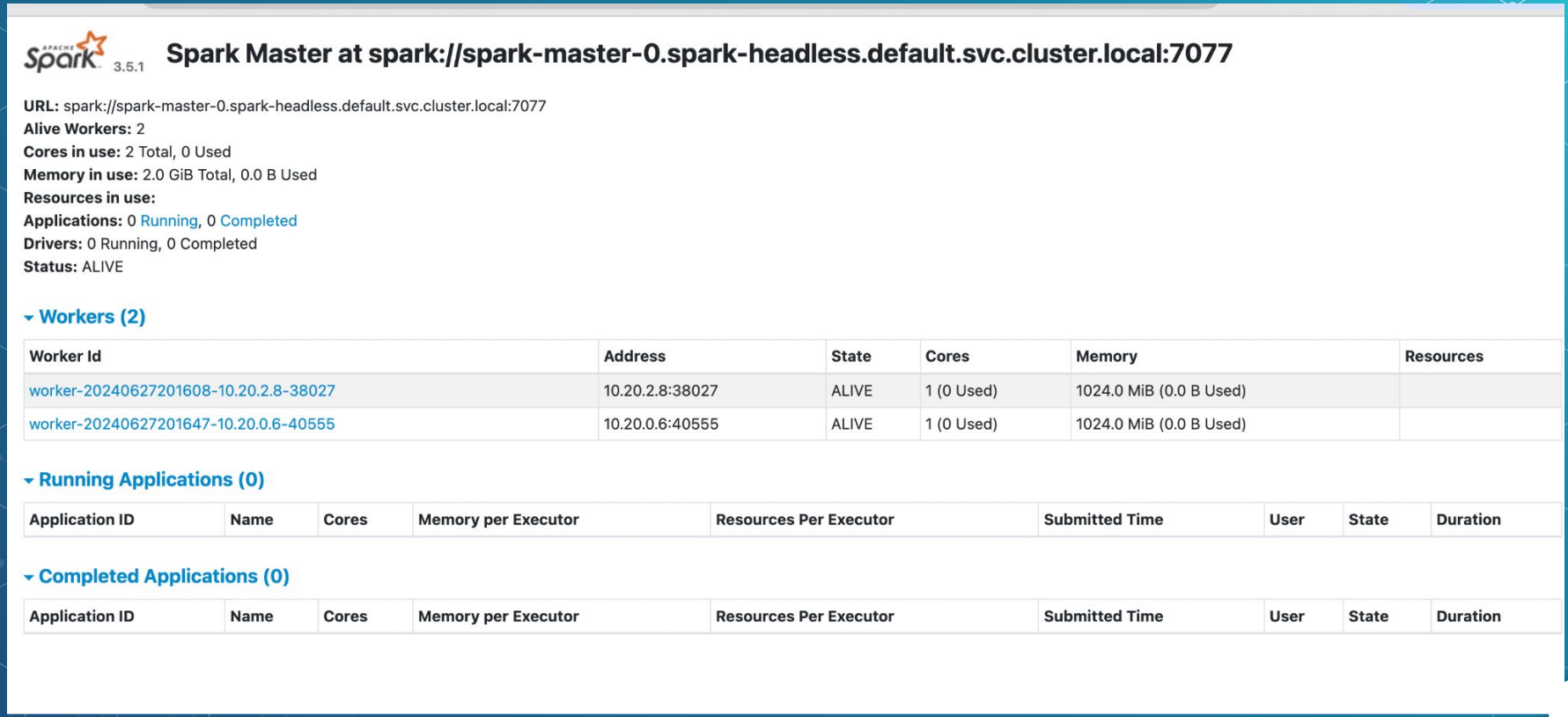
```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Thu Jun 27 20:15:12 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: spark
CHART VERSION: 9.2.4
APP VERSION: 3.5.1

** 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.
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'
```

12. Get the external IP of the running pod

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

```
belsabelteklemariam@cloudshell:~ (myfinalproject-426304)$ 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>                       71s
spark-master-svc   LoadBalancer   34.118.227.173   34.83.199.201   7077:30799/TCP,80:32493/TCP   71s
```

Open the external ip on your browser



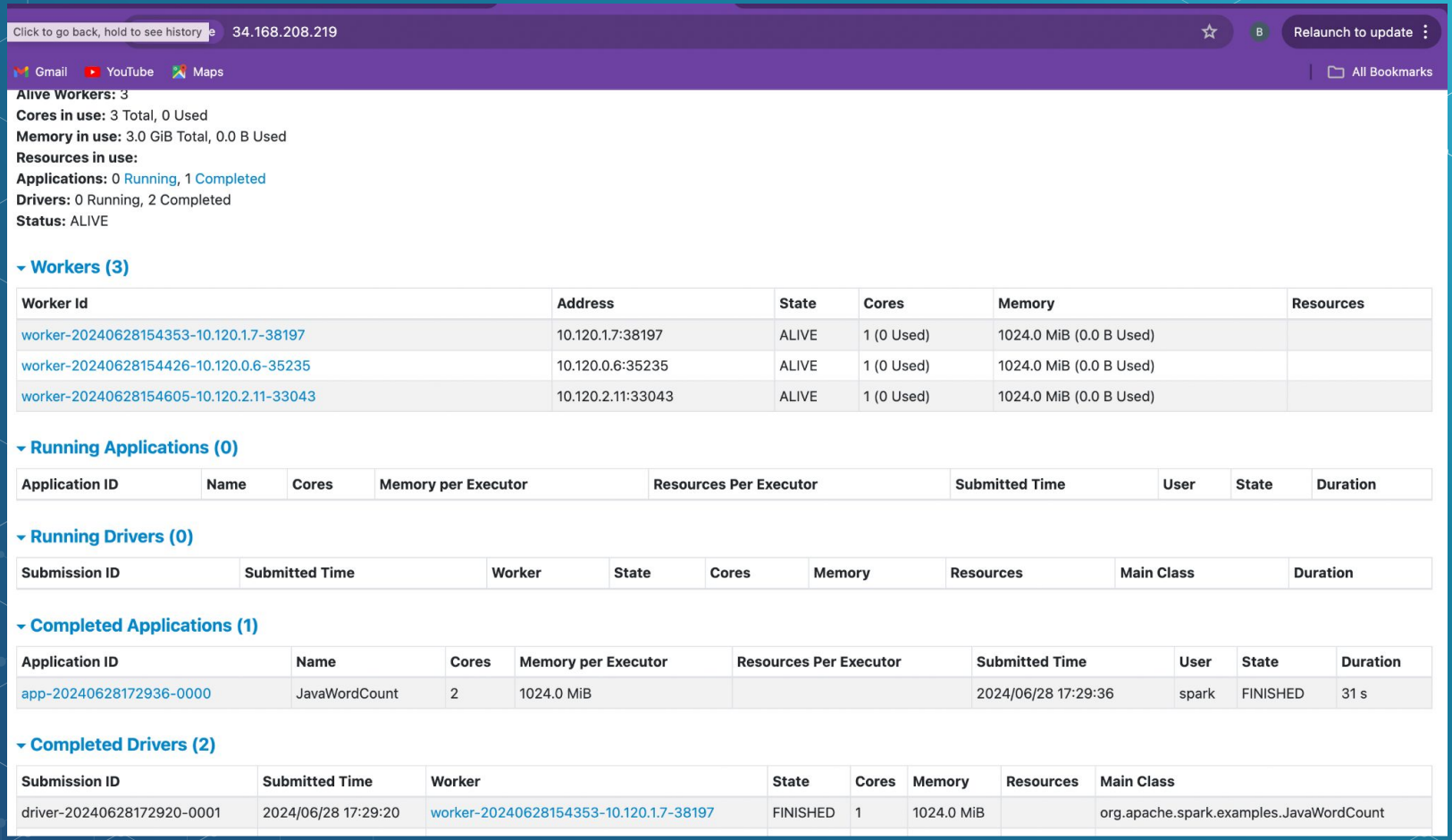
Word Count on Spark

Submit a word count task :

kubectl exec -it spark-master-0 -- spark-submit --master spark://34.168.208.219:7077 --deploy-mode cluster --class org.apache.spark.
examples.JavaWordCount /data/my.jar /data/test.txt

```
belsabelteklemariam@cloudshell:~ (cs570-427815)$ 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://34.168.208.219:7077 \
    --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
24/06/28 15:47:36 INFO SecurityManager: Changing view acls to: spark
24/06/28 15:47:36 INFO SecurityManager: Changing modify acls to: spark
24/06/28 15:47:36 INFO SecurityManager: Changing view acls groups to:
24/06/28 15:47:36 INFO SecurityManager: Changing modify acls groups to:
24/06/28 15:47:36 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(spark); groups with view permissions: Set()
ers  with modify permissions: Set(spark); groups with modify permissions: Set()
24/06/28 15:47:37 INFO Utils: Successfully started service 'driverClient' on port 32821.
24/06/28 15:47:37 INFO TransportClientFactory: Successfully created connection to /34.168.208.219:7077 after 55 ms (0 ms spent in bootstraps)
```

And on your browser, you should see this task finished



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, 1 Completed
Drivers: 0 Running, 2 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20240628154353-10.120.1.7-38197	10.120.1.7:38197	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20240628154426-10.120.0.6-35235	10.120.0.6:35235	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20240628154605-10.120.2.11-33043	10.120.2.11:33043	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

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-20240628172936-0000	JavaWordCount	2	1024.0 MiB		2024/06/28 17:29:36	spark	FINISHED	31 s

Completed Drivers (2)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20240628172920-0001	2024/06/28 17:29:20	worker-20240628154353-10.120.1.7-38197	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordCount

2. Get the name of the worker node

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

```
belsabelteklemariam@cloudshell:~ (cs570-427815)$  kubectl get pods -o wide | grep 10.120.1.7
spark-worker-0                1/1     Running   0          130m   10.120.1.7    gke-spark-default-pool-aa5ac230-xm49   <none>           <none>
belsabelteklemariam@cloudshell:~ (cs570-427815)$
```

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

```
belsabelteklemariam@cloudshell:~ (cs570-427815)$ 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-20240628172920-0001/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

1. **Execute the spark master pods**

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

2. **Stark pyspark**

```
I have no name!@c501b6cd3cf5:/opt/bitnami/spark$ spark-submit --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18, OpenJDK 64-Bit Server VM, 17.0.11
Branch HEAD
Compiled by user heartsavior on 2024-02-15T11:24:58Z
Revision fd86f85e181fc2dc0f50a096855acf83a6cc5d9c
Url https://github.com/apache/spark
Type --help for more information.
I have no name!@c501b6cd3cf5:/opt/bitnami/spark$
```

Transferred 1 item

Running python PageRank on PySpark on the pods

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

```
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/oam/2022-06-10
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/topics
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/examples/s3api/wait
file:/opt/bitnami/python/lib/python3.11/test/test_multiprocessing_fork
file:/opt/bitnami/java/legal/jdk.jshell
file:/opt/bitnami/python/lib/python3.11/test/leakers
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/numpy/f2py/tests/src/f2cmap
file:/opt/bitnami/spark/examples/src/main/scala/org/apache/spark/examples/extensions
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/timestream-write/2018-11-01
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/examples/iot-data
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/io/xml
file:/opt/bitnami/java/legal/jdk.charsets
file:/opt/bitnami/python/lib/python3.11/test/test_tomllib/data/valid/array
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/s3/2006-03-01
file:/opt/bitnami/python/lib/python3.11/test/configdata
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/examples/deploy/wait
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/apigateway/2015-07-09
file:/opt/bitnami/java/legal/jdk.security.jgss
file:/opt/bitnami/spark/python/pyspark/pandas
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/appsync/2017-07-25
file:/opt/bitnami/spark/data/streaming
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/copy_view/index
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/arrays/floating
file:/opt/bitnami/java/conf/security/policy/limited
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/arrays/datetimes
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pyasn1/type
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/appmesh/2019-01-25
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/examples/connect
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pyasn1-0.6.0.dist-info
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/network-firewall/2020-11-12
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/comprehendmedical/2018-10-30
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/numpy/core/include/numpy/random
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/examples/mediastore
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/indexes/multi
```

Enhancement Ideas

Streaming Data Processing:

- Integrate Apache Kafka or Google Cloud Pub/Sub for real-time data ingestion and processing, extending Word Count and PageRank algorithms to handle streaming data.

Advanced Analytics and Machine Learning Integration:

- Utilize Spark's machine learning libraries (MLlib) to perform sentiment analysis or classification on text data processed by Word Count.
- Explore graph-based machine learning algorithms in Spark to enhance PageRank insights or perform community detection.

Graph Visualization:

- Integrate with graph visualization tools (e.g., D3.js, Gephi) to visualize PageRank results and graph structures, providing intuitive insights into network relationships.

Conclusion

Achievements:

- Successfully implemented Word Count and PageRank algorithms using PySpark on Apache Spark deployed on Kubernetes.
- Demonstrated proficiency in distributed computing, data processing, and algorithm implementation in a cloud-native environment.

Learnings:

- Acquired hands-on experience in configuring and managing Apache Spark clusters on Kubernetes, leveraging Kubernetes' scalability and resource management capabilities.
- Expanded knowledge of PySpark programming for data manipulation, iterative algorithms (PageRank), and real-time data processing challenges.

Challenges Overcome:

- Overcame challenges related to cluster setup, resource allocation, and optimization of Spark jobs for efficient data processing at scale.
- Managed complexities of distributed computing, fault tolerance, and data pipeline orchestration in a Kubernetes environment.

References

Word Count

Page Rank

Word_count_Example

Apache Spark Word Count Program | Using PySpark