PySpark on Kubernetes: Word Count and PageRank

Shruti Kavishwar

San Francisco Bay University

Guided By: Dr. Henry Chang

Agenda

- Introduction
- Creating a GKE Cluster
- Deploying Spark on Kubernetes
- Word count Project
- PageRank Project
- Conclusion

Introduction

- **Objectives**
  - Create and manage a GKE cluster.
  - Deploy Apache Spark on Kubernetes.
  - Execute Word Count and PageRank tasks.
- **Technologies Used**
  - Apache Spark
  - Kubernetes Google Kubernetes Engine (GKE)
- **Importance**
  - Integrates big data processing with container orchestration.
  - Utilizes cloud infrastructure for large-scale data tasks.
  - Provides practical experience with modern technologies.
- **Expected Outcomes**
  - Manage a GKE cluster.
  - Deploy and configure Spark on Kubernetes.
  - Execute and validate Word Count and PageRank.

Creating a GKE Cluster

```
$ gcloud clusters create spark –num-nodes=1 –machine-type=e2-highmem-2 –region=us-east1
```

This command is useful for setting up a small, memory-intensive Kubernetes cluster for development or testing purposes. The cluster could be used for running applications that require significant memory resources, such as big data processing or machine learning workloads. By specifying a single node and a high-memory machine type, you ensure that the cluster can handle memory-intensive tasks without incurring the cost of additional nodes.

```
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$ gcloud container clusters create spark --num-nodes=1 --machine-type=e2-highmem-2 --region=us-east1
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, plea
se pass the `--no-enable-ip-alias` flag
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster spark in us-east1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/mapreduce-week2-hw1-cs570/zones/us-east1/clusters/spark].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-east1/spark?project=mapreduce-week2-hw1-cs570
kubeconfig entry generated for spark.
NAME: spark
LOCATION: us-east1
MASTER_VERSION: 1.29.4-gke.1043002
MASTER_IP: 34.74.12.71
MACHINE_TYPE: e2-highmem-2
NODE_VERSION: 1.29.4-gke.1043002
NUM_NODES: 3
STATUS: RUNNING
```

Creating a GKE Cluster

Install NFS server provisioner

```
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$ helm repo add stable ht
tps://charts.helm.sh/stable
"stable" has been added to your repositories
```

```
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$ helm install nfs stable
/nfs-server-provisioner \
--set persistence.enabled=true,persistence.size=5Gi
WARNING: This chart is deprecated
NAME: nfs
LAST DEPLOYED: Wed Jun 26 20:41:58 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
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$
```

$ helm repo add stable

https://charts.helm.sh/stable

$ helm install nfs stable /nfs-server-provisioner \

–set persistence.enabled=true,persistence.size=5Gi

Deploying Spark On Kubernetes

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

```
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$ vi spark-pvc.yaml
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$ 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
```

Deploying Spark On Kubernetes

- Apply the YAML file to create the PVC
  - $ kubectl apply -f spark-pvc.yaml
- Check if the persistent volume is created
  - $ kubectl get pvc

```
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$ kubectl apply -f spark-pvc
yaml
persistentvolumeclaim/spark-data-pvc created
pod/spark-data-pod created
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$ kubectl get pvc
NAME                                    STATUS   VOLUME
       CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
data-nfs-nfs-server-provisioner-0   Bound    pvc-8f1fc44a-b6d0-4fa5-afaf-36e197
143d   5Gi        RWO            standard-rwo   <unset>                 20m
spark-data-pvc                      Bound    pvc-ed57117e-4780-42ea-9f68-4f93cc
0fcb   2Gi        RWX            nfs            <unset>                 14s
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$
```

Deploying Spark On Kubernetes

- Create and Prepare your application JAR files
  - $ docker run -v /tmp:/tmp -it bitnami/spark – find /opt/bitnami/spark/examples/jars/ -name spark-examples* cp {} /tmp/my.jar \;

```
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ docker run -v /t
p:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-e
amples* 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 21:07:56.03 INFO  ==>
spark 21:07:56.03 INFO  ==> Welcome to the Bitnami spark container
spark 21:07:56.03 INFO  ==> Subscribe to project updates by watching https://git
ub.com/bitnami/containers
spark 21:07:56.04 INFO  ==> Submit issues and feature requests at https://github
com/bitnami/containers/issues
spark 21:07:56.04 INFO  ==> Upgrade to Tanzu Application Catalog for production
nvironments to access custom-configured and pre-packaged software components. Ga
n enhanced features, including Software Bill of Materials (SBOM), CVE scan resul
 reports, and VEX documents. To learn more, visit https://bitnami.com/enterprise
spark 21:07:56.04 INFO  ==>

find: paths must precede expression: `cp'
```

Deploying Spark On Kubernetes

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

```
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ echo "how much w
od could a woodpecker chuck if a woodpecker could chuck wood" > /tmp/test.txt
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ cat /tmp/test.tx

how much wood could a woodpecker chuck if a woodpecker could chuck wood
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$
```

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

```
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ kubectl cp /tmp/
y.jar spark-data-pod:/data/my.jar
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ kubectl cp /tmp/
est.txt spark-data-pod:/data/test.txt
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$
```

Deploying Spark On Kubernetes

- Verify the files are inside the pvc
  - $ kubectl exec -it spark-data-pod – ls -al /data

```
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ kubectl exec -it
spark-data-pod -- ls -al /data
total 1540
drwxrwsrwx 2 root root    4096 Jun 26 21:20 .
drwxr-xr-x 1 root root    4096 Jun 26 21:02 ..
-rw-r--r-- 1 1001 root 1564260 Jun 26 21:19 my.jar
-rw-rw-r-- 1 1000 1000      72 Jun 26 21:20 test.txt
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$
```

Deploying Spark On Kubernetes

- Deploy Apache on Kubernetes using shared volume. Create a **spark-chart.yaml**

```
skavishw276@cloudshell:~ (mapreduce-week2-hw1-cs570)$ 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
```

Deploying Spark On Kubernetes

- Deploy Apache Spark on Kubernetes cluster using bitnami Apache Spark Helm chart and supply ot with the configuration file.
  - $ helm install spark bitnami/spark -f spark-chart.yaml

```
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ helm install spa
k bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Wed Jun 26 22:04:37 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 **
```

Deploying Spark On Kubernetes

```
** 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

** IMPORTANT: When submit an application the --master parameter should be set to the service IP, if not, the application will not resolve the master. **
```

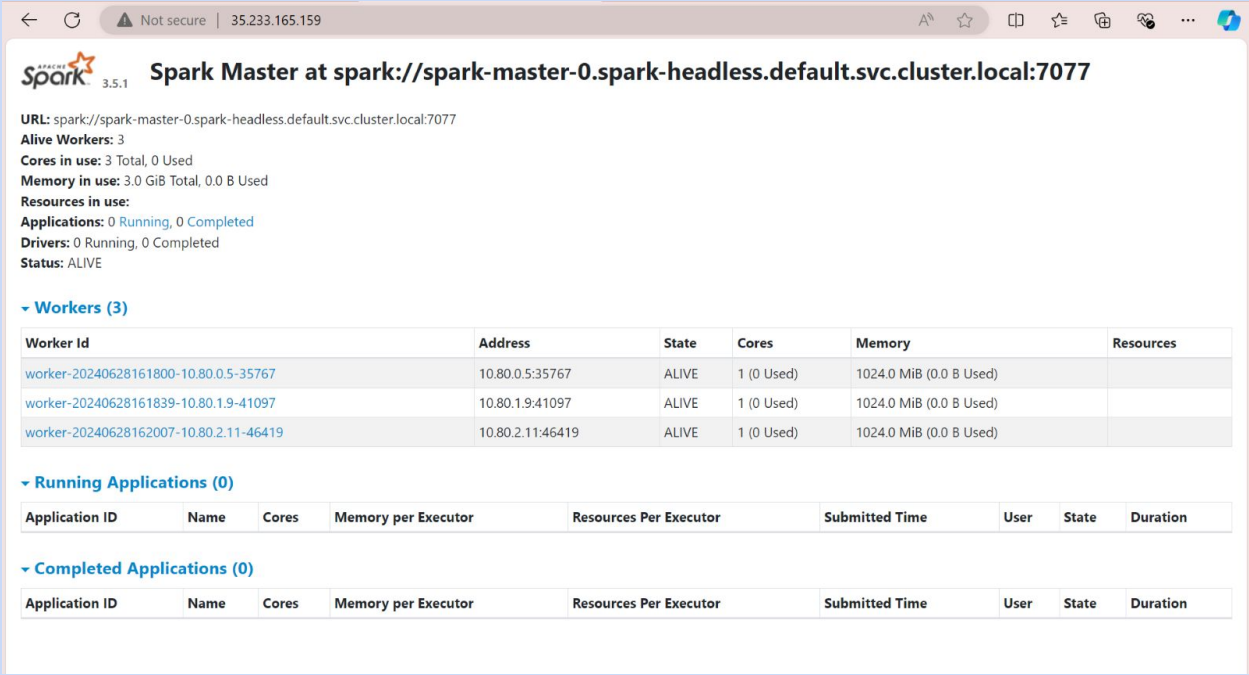
Deploying Spark On Kubernetes

- Get the external IP of the spark master svc pod
  - $ kubectl get svc -l “app.kubernetes.io/instances=spark,app.kubernetes.io/name=spark*

```
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ kubectl get s
vc -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>
                     25m
spark-master-svc    LoadBalancer   34.118.234.95   35.233.165.159   7077:32629
/TCP,80:30548/TCP    25m
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$
```

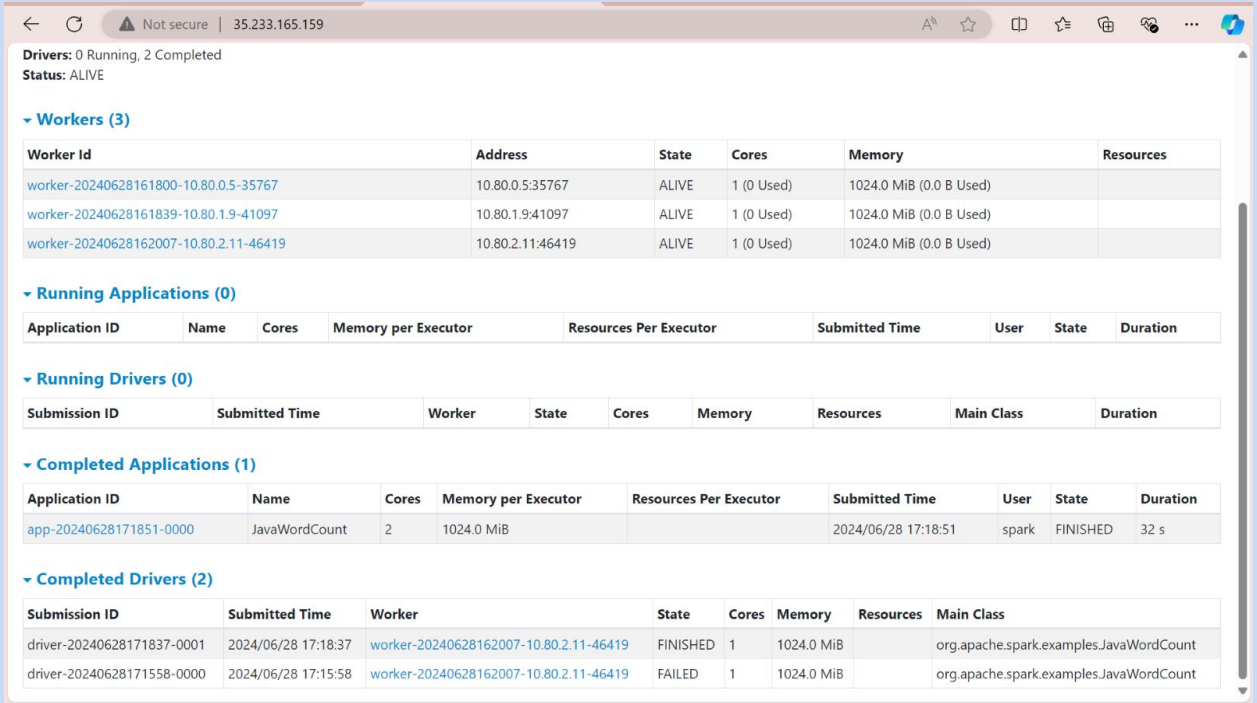
Deploying Spark On Kubernetes

- In your browser use the external IP to access the Spark Master UI.



Submitting the Word Count Project

- *This command runs a Spark job on a Kubernetes cluster, using spark-submit to execute a Java class (JavaWordCount) from the JAR file (/data/my.jar) on a Spark master node (spark://35.233.165.159:7077), processing the input file (/data/test.txt).*
  - $ kubectl exec -it spark-master-0 – spark-submit –master spark://35.233.165.159:7077 –deploy-mode cluster –class ord.apache.spark.examples.JavaWordCount /data/my.jar /data/test.txt



Output for the Word Count Project

- View output on the worker node
  - $ kubectl exec -it spark-worker-2 – bash

```
skavishw276@cloudshell:~/wordcount (mapreduce-week2-hw1-cs570)$ kubectl exec
-it spark-worker-2 -- bash
I have no name!@spark-worker-2:/opt/bitnami/spark$ cd /opt/bitnami/spark/work
I have no name!@spark-worker-2:/opt/bitnami/spark/work$ ls
driver-20240628171558-0000  driver-20240628171837-0001
I have no name!@spark-worker-2:/opt/bitnami/spark/work$ cat driver-2024062817
1558-0000/
cat: driver-20240628171558-0000/: Is a directory
I have no name!@spark-worker-2:/opt/bitnami/spark/work$ cat driver-2024062817
1558-0000/stdout
I have no name!@spark-worker-2:/opt/bitnami/spark/work$ cat driver-2024062817
1837-0001/stdout
if: 1
a: 2
how: 1
could: 2
wood: 2
woodpecker: 2
much: 1
chuck: 2
I have no name!@spark-worker-2:/opt/bitnami/spark/work$
```

Submitting PageRank Project

- Spark-submit –version

```
I have no name!@spark-master-0:/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!@spark-master-0:/opt/bitnami/spark$
```

- Go to the directory where the pagerank.py is located
  - $ cd /opt/bitnami/spark/examples/src/main/python

Output PageRank Project

- Run pagerank.py using pyspark (/opt is the directory, 2 is the number of iterations to run pagerank
  - $ spark-submit pagerank.py /opt 2

```
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/examples/organizations
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/route53-recovery-cluster/2019-12-02
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/marketplace-deployment/2023-01-25
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/elasticache/2015-02-02
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/io/pytables
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/docutils/parsers/rst/include
file:/opt/bitnami/spark/data/mllib/ridge-data
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/scalar/timestamp/methods
file:/opt/bitnami/python/lib/python3.11/xmlrpc
file:/opt/bitnami/spark/r/lib
file:/opt/bitnami/python/lib/python3.11/test/test_import/data/package2
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/tzdata/zoneinfo/europe
file:/opt/bitnami/python/lib/python3.11/test/test_importlib/import_
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/boto3/data/ec2/2016-09-15
file:/opt/bitnami/spark/python/pyspark/streaming
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/examples/waf-regional
file:/opt/bitnami/java/legal/jdk.crypto.ec
file:/opt/bitnami/java/legal/jdk.management.jfr
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/honeycode/2020-03-01
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/cloudfront/2016-08-01
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore-1.34.103.dist-info
file:/opt/bitnami/spark/python/pyspark/errors
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/numpy/typing/tests
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/amplifybackend/2020-08-11
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/awscli/examples/mediapackage
file:/opt/bitnami/java/legal/jdk.xml.dom
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/reshape/concat
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/pandas/tests/strings
file:/opt/bitnami/python/lib/python3.11/test/test_importlib/builtin
file:/opt/bitnami/python/lib/python3.11/test/test_importlib
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/s3transfer
file:/opt/bitnami/spark/python/pyspark/errors/exceptions
file:/opt/bitnami/spark/python/pyspark/pandas/plot
file:/opt/bitnami/python/lib/python3.11/distutils/tests
file:/opt/bitnami/python/lib/python3.11/test/test_doctest
file:/opt/bitnami/spark/venv/lib/python3.11/site-packages/botocore/data/ssm-contacts/2021-05-03
```

Conclusion

- **Project Summary**
  - Successfully implemented Word Count and PageRank using PySpark on Kubernetes.
  - Achieved seamless integration of Apache Spark with Kubernetes on Google Kubernetes Engine (GKE).
- **Key Achievements**
  - Created and managed a GKE cluster.
  - Deployed Apache Spark on Kubernetes using persistent volumes.
  - Executed Word Count and PageRank tasks with PySpark.
- **Challenges and Solutions**
  - Encountered deployment and execution errors.
  - Resolved issues with correct configurations and commands.
- **Learnings and Insights**
  - Gained hands-on experience with container orchestration and big data processing.
  - Understood the importance of cloud infrastructure in handling large-scale data tasks.
- **Future Work**
  - Explore additional big data applications on Kubernetes.
  - Optimize performance and scalability of current implementations.
  - Implement monitoring and logging for better management and troubleshooting.

Github Link: ShrutiK02 (github.com)