Apache Spark on Kubernetes

WordCount+PageRank+GKE

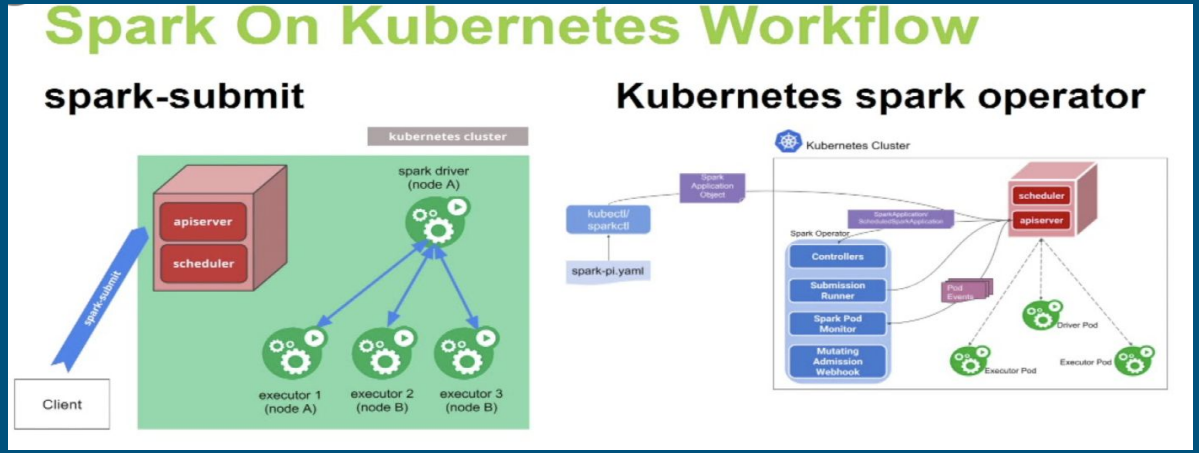
By
Maryam Taherzadeh
Prepared Under the direction of Professor Henry Chang

School of Engineering
Northwestern Polytechnic University
117 Fourier Avenue, Fremont, CA 94539

April 2021

Table of Content:

Introduction



Spark on Kubernetes

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

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

Design

Spark Concept

Kubernetes :

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

Design
Spark Concept

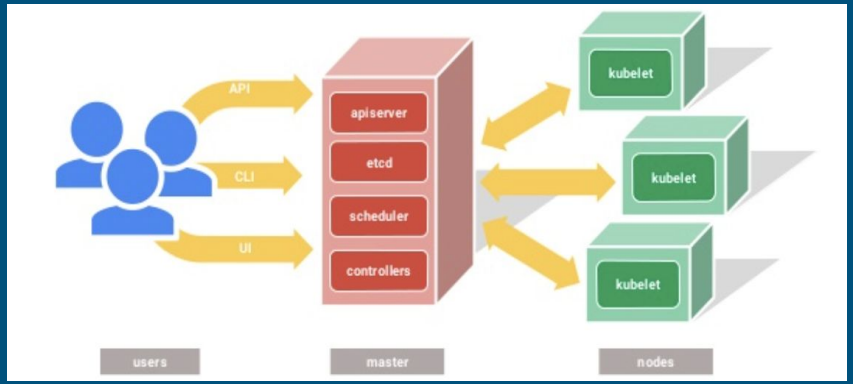
Containers:

- Repeatable Builds and Workflows
- Application Portability
- High degree of Control over software
- Faster Development Cycle
- Reduced devops load
- Improved Infrastructure Utilization

Design
Spark Concept

Node and Pods:

- A pod is a set of co-located containers
- Created by a declarative specification supplied to the master
- Each pod has its own IP address
- Volumes can be local or network -attached



Design
Spark Concept

Spark

- Apache Spark is a general-purpose cluster in-memory computing system
- Provides high level APIs in Java, Scala and Python, and an optimized engine that supports general execution graphs.
- Provides various high level tool like Spark SQL for structured data processing, Mlib for Machine Learning and more

Design
Spark Concept

PySpark

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

Design

Spark Concept

Fundamental of PySpark

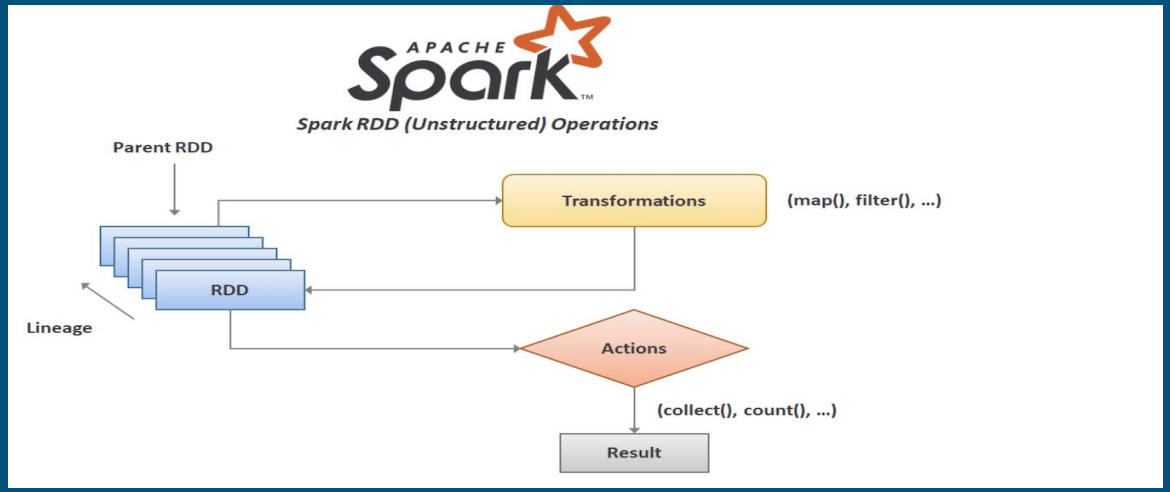
RDDs:

- **RDD is a distributed collections of objects created from**
  - Python
  - Java
  - Scala
  - User-defined classes created
    - by loading an external dataset, or
    - by distributing a collection of objects in their driver program.

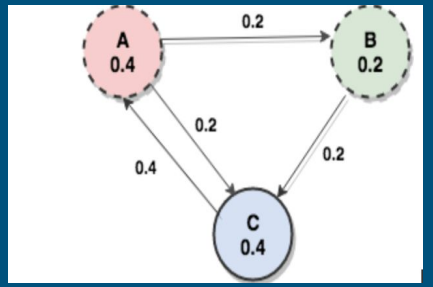
Design
Spark concept
Fundamental of PySpark

Two types of operation supported by Spark RDDs:

Transformations, which create a new RDD by transforming from an existing RDD, and ***actions*** which compute and write a value to the driver program.



Design
Spark concept



PageRank:
PageRank is one of many algorithms Google applies to work out which order to display search results.

The algorithm is describe as follows:

1. Initialize the vertices with a starting PageRank of 1/N, where N is the number of vertices in the graph.
2. Loop:
   - For each vertex, transmit a PageRank of **1/M** along each outbound edge, where **M** is the outdegree of the vertex.
   - At each vertex receiving incoming PageRanks from adjacent vertices, sum these up and make that the new PageRank for the vertex.
   - If PageRanks haven't significantly changed across the graph since the previous iteration, then exit.

The formula for page rank :

$$PR(A) = (1 - d) + d\,(PR(t_1) / C(t_1) + \ldots + PR(t_n)/C(t_n))$$

Design
Spark concept

WordCount:

How to count the occurrence words in a text line.

Word Count

MapReduce

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

```
import sys

from pyspark import SparkContext, SparkConf

if __name__ == "__main__":

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

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

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

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

Design
Spark concept

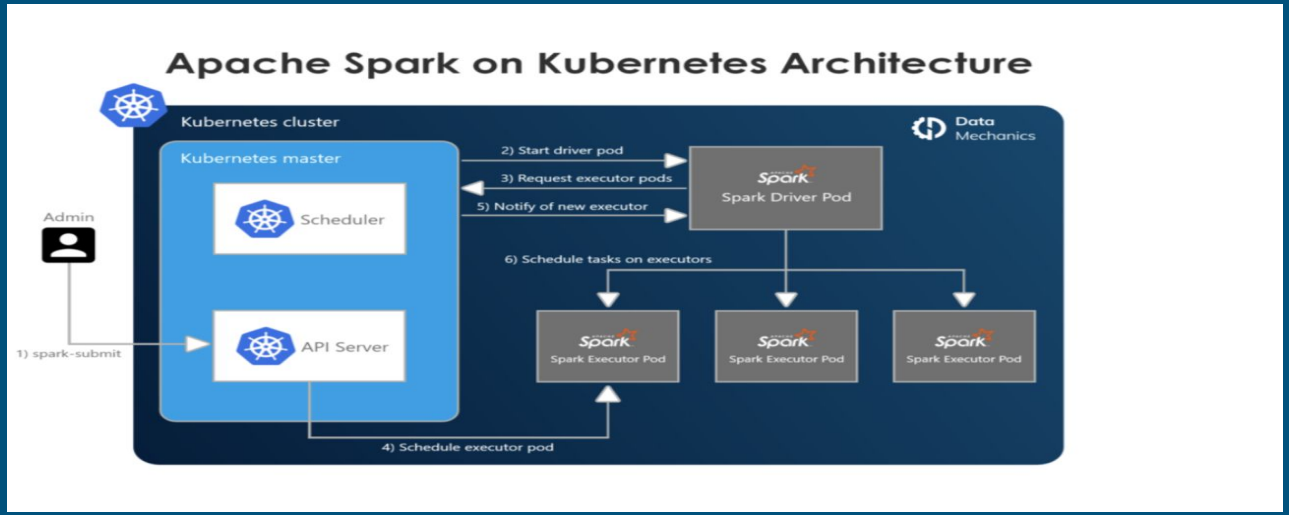
Why spark on Kubernetes:

- Docker and container Ecosystem
- Kubernetes
  - - Lots of addon services: third party logging, monitoring, and security tools
- Resources sharing between batch, serving and stateful workloads
  - - Streamlined developer experience
  - - Reduced operational cost
  - - Improved infrastructure utilization

Design
Spark concept
Apache Spark on Kubernetes Architecture

Spark-Submit can be directly used to submit a Spark application to a Kubernetes cluster. The submission mechanism works as follows:

Spark creates a Spark driver running within a Kubernetes pod.The driver creates executors which are also running within Kubernetes pods and connects to them, and executes application code.When the application completes, the executor pods terminate and are cleaned up, but the driver pod persists logs and remains in “completed” state in the Kubernetes API until it’s eventually garbage collected or manually cleaned up.The driver and executor pod scheduling is handled by Kubernetes. Communication to the Kubernetes API is done via fabric8. It is possible to schedule the driver and executor pods on a subset of available nodes through a node selector using the configuration property for it



Implementation:

Wordcount running on spark, delpoying to kubernetes on GKE

1. Create a cluster on GKE with

gcloud container clusters create spark --num-nodes=1 --machinetype=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   35.233.228.208  e2-highmem-2  1.18.16-gke.502  3          RUNNING
taherzadeh19529@cloudshell:~ (phonic-axle-307118)$
```

Implementation:

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
```

```
taherzadeh19529@cloudshell:~ (phonic-axle-307118)$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "bitnami" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈Happy Helming!⎈
taherzadeh19529@cloudshell:~ (phonic-axle-307118)$
```

Implementation:

Create image and deploy spark to kubernetes

2.Install the NFS Server Provisioner

helm install nfs stable/nfs-server-provisioner \

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ helm install nfs stable/nfs-server-provisioner \
> --set persistence.enabled=true,persistence.size=5Gi
WARNING: This chart is deprecated
NAME: nfs
LAST DEPLOYED: Fri Apr 23 18:58:34 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
```

Implementation:

Create image and deploy spark to kubernetes

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ cat spark-pvc.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: spark-data-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 2Gi
  storageClassName: nfs
---
apiVersion: v1
kind: Pod
metadata:
  name: spark-data-pod
spec:
  volumes:
    - name: spark-data-pv
      persistentVolumeClaim:
        claimName: spark-data-pvc
  containers:
    - name: inspector
      image: bitnami/minideb
      command:
        - sleep
        - infinity
      volumeMounts:
        - mountPath: "/data"
          name: spark-data-pv
```

Implementation:

Create image and deploy spark to kubernetes

4. Apply the above yaml descriptor

- kubectl apply -f spark-pvc.yaml

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl apply -f spark-pvc.yaml
persistentvolumeclaim/spark-data-pvc created
pod/spark-data-pod created
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

Implementation:

Create image and deploy spark to kubernetes

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 \;
```

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ docker run -v /tmp:/tmp -it bitnami/spark -- find/optaherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ docker
run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples* -exec cp {} /tmp/my.jar \;
 19:31:33.71
 19:31:33.71 Welcome to the Bitnami spark container
 19:31:33.71 Subscribe to project updates by watching https://github.com/bitnami/bitnami-docker-spark
 19:31:33.72 Submit issues and feature requests at https://github.com/bitnami/bitnami-docker-spark/issues
 19:31:33.72
```

Implementation:

Create image and deploy spark to kubernetes

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ echo "how much wood could a woodpecker chuck if a woodpecker could chuck wood" > /tmp/test.txt
```

Implementation:

Create image and deploy spark to kubernetes

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl cp /tmp/test.txt spark-data-pod:/data/test.txt
```

Implementation:

Create image and deploy spark to kubernetes

8. Make sure the files a inside the persistent volume

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl exec -it spark-data-pod -- ls -al /data
total 1504
drwxrwsrwx 2 root root    4096 Apr 23 19:34 .
drwxr-xr-x 1 root root    4096 Apr 23 19:27 ..
-rw-r--r-- 1 1001 root 1527168 Apr 23 19:34 my.jar
-rw-r--r-- 1 1000 1001      72 Apr 23 19:34 test.txt
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

Implementation:

Create image and deploy spark to kubernetes

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

```
bash: cd: spark-chart.yaml: Not a directory
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

Implementation:

Create image and deploy spark to kubernetes

10. Check the pods is running:

- kubectl get pods

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
nfs-nfs-server-provisioner-0      1/1     Running   0          81m
spark-data-pod                    1/1     Running   0          52m
```

Implementation:

Create image and deploy spark to kubernetes

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
```

Implementation:

Create image and deploy spark to kubernetes

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

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

```
"bitnami" has been added to your repositories
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Fri Apr 23 19:50:46 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
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:3.1.1-debian-10-r42 \
    -- spark-submit --master spark://$SUBMIT_IP:7077 \
    --deploy-mode cluster \
```

Implementation:

Create image and deploy spark to kubernetes

12. Get the external IP of the running pod

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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>                        5m54s
spark-master-svc   LoadBalancer   10.3.252.70   35.197.61.8   7077:30693/TCP,80:30513/TCP   5m54s
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

Implementation:

Create image and deploy spark to kubernetes

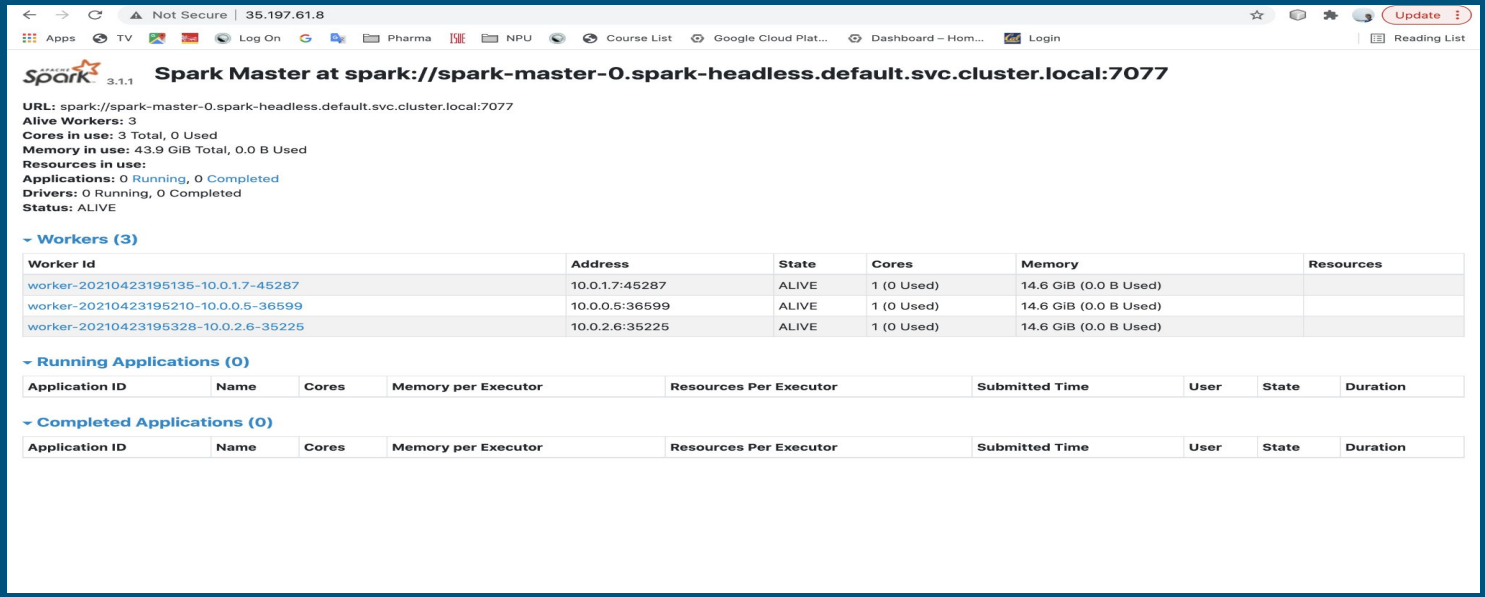
12. Get the external IP of the running pod

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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>                        5m54s
spark-master-svc   LoadBalancer   10.3.252.70   35.197.61.8   7077:30693/TCP,80:30513/TCP   5m54s
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

Test Result:

1. Open the external ip on your browser



Test Result:

Word Count on Spark
Submit a word count task :

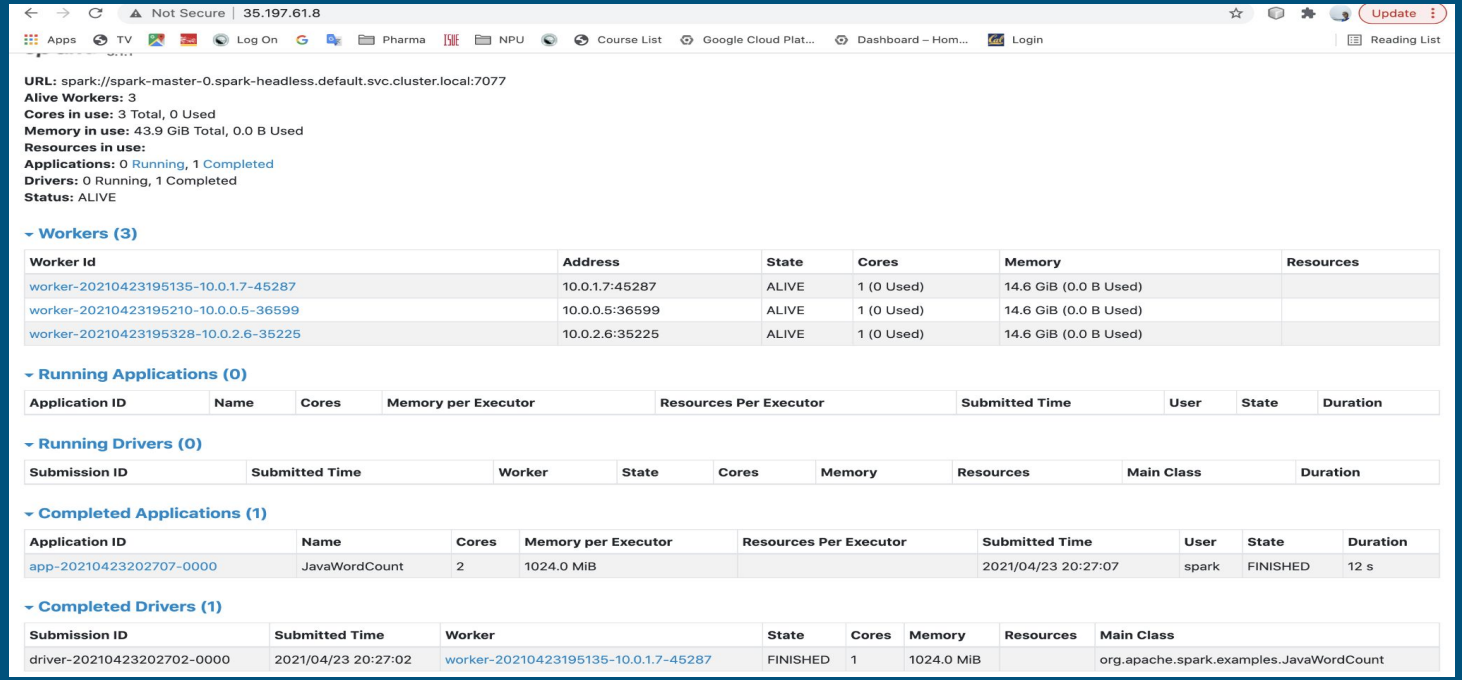
```
kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \

--image docker.io/bitnami/spark:3.0.1-debian-10-r115 \

-- spark-submit --master spark://LOAD-BALANCER-External-ipADDRESS:7077 \ --deploy-mode cluster \

--class org.apache.spark.examples.JavaWordCount \
/data/my.jar /data/test.txt
```

Test Result:

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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://35.197.61.8:7077 \
> --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
21/04/23 20:27:01 INFO SecurityManager: Changing view acls to: spark
21/04/23 20:27:01 INFO SecurityManager: Changing modify acls to: spark
21/04/23 20:27:01 INFO SecurityManager: Changing view acls groups to:
21/04/23 20:27:01 INFO SecurityManager: Changing modify acls groups to:
21/04/23 20:27:01 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(spark); groups with view permissions: Set(); us
ers  with modify permissions: Set(spark); groups with modify permissions: Set()
21/04/23 20:27:02 INFO Utils: Successfully started service 'driverClient' on port 45657.
21/04/23 20:27:02 INFO TransportClientFactory: Successfully created connection to /35.197.61.8:7077 after 76 ms (0 ms spent in bootstraps)
21/04/23 20:27:02 INFO ClientEndpoint: Driver successfully submitted as driver-20210423202702-0000
21/04/23 20:27:02 INFO ClientEndpoint: ... waiting before polling master for driver state
21/04/23 20:27:07 INFO ClientEndpoint: ... polling master for driver state
21/04/23 20:27:07 INFO ClientEndpoint: State of driver-20210423202702-0000 is RUNNING
21/04/23 20:27:07 INFO ClientEndpoint: Driver running on 10.0.1.7:45287 (worker-20210423195135-10.0.1.7-45287)
21/04/23 20:27:07 INFO ShutdownHookManager: Shutdown hook called
21/04/23 20:27:07 INFO ShutdownHookManager: Deleting directory /tmp/spark-5c98909d-8319-42c9-bc39-dc5c9cb98f51
pod "spark-client" deleted
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

TestResult

And on your browser, you should see this task finished



Not Secure | 35.197.61.8

Update

Apps TV Log On Pharma NPU Course List Google Cloud Plat... Dashboard – Hom... Login Reading List

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-20210423195135-10.0.1.7-45287	10.0.1.7:45287	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	
worker-20210423195210-10.0.0.5-36599	10.0.0.5:36599	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	
worker-20210423195328-10.0.2.6-35225	10.0.2.6:35225	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-20210423202707-0000	JavaWordCount	2	1024.0 MiB		2021/04/23 20:27:07	spark	FINISHED	12 s

Completed Drivers (1)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20210423202702-0000	2021/04/23 20:27:02	worker-20210423195135-10.0.1.7-45287	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordCount

TestResult

2. Get the name of the worker node

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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>
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$
```

Test Result

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

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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$
```

Test Result

Running python PageRank onPySpark on the pods

1. Execute the spark master pods

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

2. Stark pyspark

- pyspark

```
taherzadeh19529@cloudshell:~/wordcount (phonic-axle-307118)$ 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.
21/04/23 20:51:05 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes 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, 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-1619211067920).
SparkSession available as 'spark'.
```

Test Result

Running python PageRank on PySpark on the pods

3. Exit pyspark with

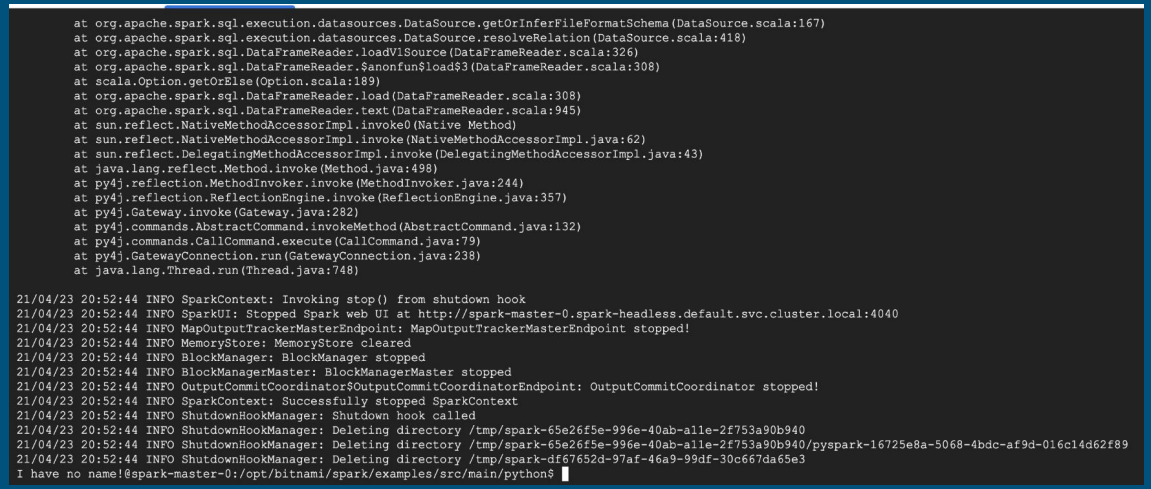
exit()

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



Conclusion

This project applied PySpark to implement Word Count and Page Rank on Apache Spark running on Kubernetes. Using Spark on Kubernetes has many benefits. First one is Containerization which means Containers make your applications more portable, they simplify the packaging of dependencies, they enable repeatable and reliable build workflows. The second one is Efficient resource sharing leading to big cost savings. The third one is Integration in a rich ecosystem - cloud-agnostic & with less vendor lock-in

References

- https://www.datamechanics.co/blog-post/pros-and-cons-of-running-apache-spark-on-kubernetes
- https://towardsdatascience.com/how-to-guide-set-up-manage-monitor-spark-on-kubernetes-with-code-examples-c5364ad3aba2
- https://www.datamechanics.co/apache-spark-on-kubernetes
- https://spark.apache.org/docs/latest/running-on-kubernetes.html
- https://npu85.npu.edu/~henry/npu/classes/master_apache_spark/kubernetes/slide/exercise_kubernetes.html
- https://npu85.npu.edu/~henry/npu/classes/master_apache_spark/kubernetes/slide/index_slide.html