

## Deploying on Kubernetes-

### Jupyter Notebook

1. Create a Kubernetes cluster on GCP - gcloud container clusters create --machine-type n1-standard-2 --num-nodes 2 --zone us-central1-a --cluster-version latest adityadwubernetescluster
2. docker pull adityadw/jupyter\_notebook
3. docker tag adityadw/jupyter\_notebook:latest gcr.io/genuine-grid-327615/adityadw/jupyter\_notebook:1
4. docker push gcr.io/genuine-grid-327615/adityadw/jupyter\_notebook:1

```
adiwivedi96@cloudshell:~ (genuine-grid-327615)$ docker pull adityadw/jupyter_notebook
Using default tag: latest
latest: Pulling from adityadw/jupyter_notebook
284055322776: Pull complete
5031b1992b2a: Pull complete
b6329deb9320: Pull complete
56769b3e91cb: Pull complete
c3b5faa0dcbf: Pull complete
Digest: sha256:758ab9230acb3707902c3bc1c2604b711dd570c24eee6925dcc85a4028582b81
Status: Downloaded newer image for adityadw/jupyter_notebook:latest
docker.io/adityadw/jupyter_notebook:latest
adiwivedi96@cloudshell:~ (genuine-grid-327615)$ docker tag adityadw/jupyter_notebook:latest gcr.io/genuine-grid-327615/adityadw/jupyter_notebook:1
The push refers to repository [gcr.io/genuine-grid-327615/adityadw/jupyter_notebook]
38301074c5ed: Pushed
aa81112cal6f: Pushed
2889f2c4cf15: Pushed
2d652f97bd28: Pushed
824bf060fd3d: Layer already exists
1: digest: sha256:758ab9230acb3707902c3bc1c2604b711dd570c24eee6925dcc85a4028582b81 size: 1371
adiwivedi96@cloudshell:~ (genuine-grid-327615)$
```

5. Create deployment file for jupyter notebook as follows, here the internal port on which service is running is 8888 -

```
jupyter_notebook_deployment.yaml > ...
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: jupyter
5    labels:
6      app: jupyter
7  spec:
8    selector:
9      matchLabels:
10     app: jupyter
11   replicas: 2
12   minReadySeconds: 15
13   strategy:
14     type: RollingUpdate
15     rollingUpdate:
16       maxUnavailable: 1
17       maxSurge: 1
18   template:
19     metadata:
20       labels:
21         app: jupyter
22     spec:
23       containers:
24         - image: adityadw/jupyter_notebook
25           imagePullPolicy: Always
26           name: jupyter
27           ports:
28             - containerPort: 8888
29
```

6. kubectl apply -f jupyter\_notebook\_deployment.yaml
7. Create service file in order to deploy jupyter notebook as follows. Presently it has been set as loadbalancing service so that the testing can be done via browser -

```

jupyter_service.yaml  iupyter_notebook_deployment.yaml
jupyter_service.yaml > ...
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: jupyter
5  spec:
6    type: LoadBalancer
7    ports:
8      - port: 8888
9        protocol: TCP
10       targetPort: 8888
11    selector:
12      app: jupyter

```

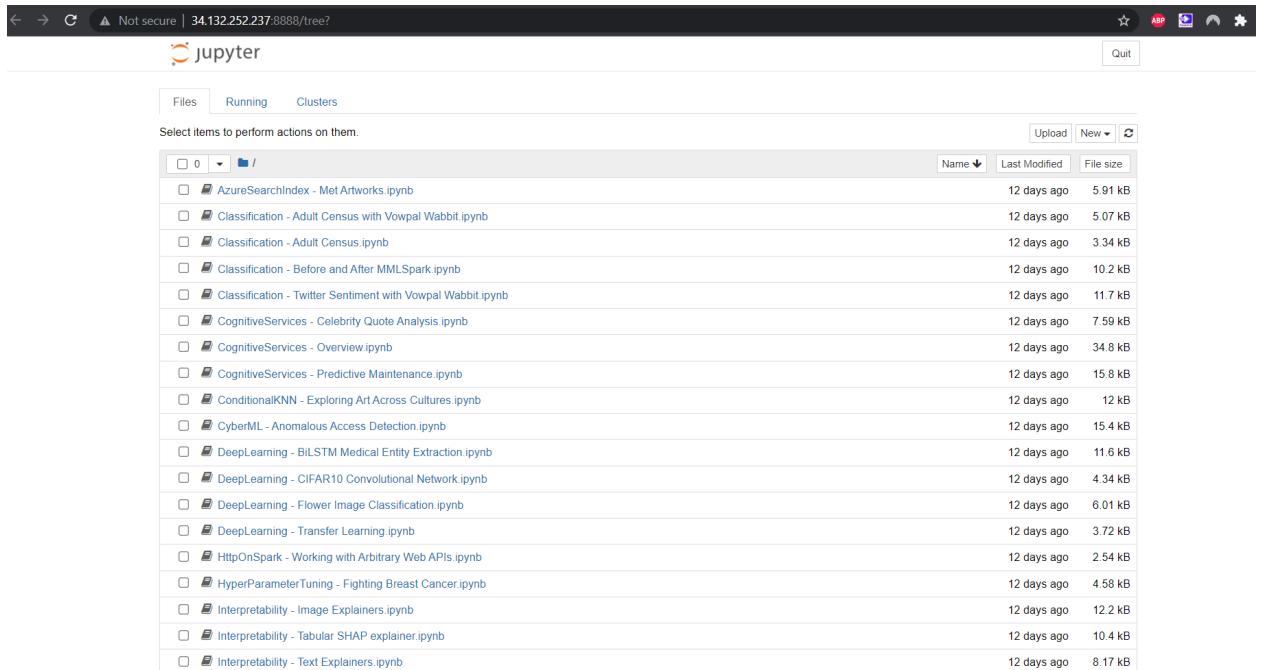
## 8. kubectl apply -f jupyter\_service.yaml

```

adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f jupyter_notebook_deployment.yaml
deployment.apps/jupyter created
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f jupyter_service.yaml
service/jupyter created
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f jupyter_service.yaml
service/jupyter configured
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ 

```

## 9. We have the service running on public IP as follows -



## Apache Spark

1. Follow the steps as above to load the docker image from the third party to gcr - docker pull bitnami/spark
2. docker tag bitnami/spark:latest gcr.io/genuine-grid-327615/bitnami/spark:1
3. docker push gcr.io/genuine-grid-327615/bitnami/spark:1

```

didwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker pull bitnami/spark
Using default tag: latest
latest: Pulling from bitnami/spark
3b664af592e: Pull complete
da19ddc65be: Pull complete
d21c74b4228: Pull complete
0c515956a4b: Pull complete
b20ff6fabed: Pull complete
51825bf66a9: Pull complete
7cd8e55f83fb: Pull complete
7800b61fa67: Pull complete
89ae2b97f94: Pull complete
78a7ee58f2a: Pull complete
Digest: sha256:373048f025775d4f8b0931fa704e87f8603469e5bfef5068c18855c00affca1
Status: Downloaded newer image for bitnami/spark:latest
docker.io/bitnami/spark:latest
didwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker tag bitnami/spark:latest gcr.io/genuine-grid-327615/bitnami/spark:1
didwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker push gcr.io/genuine-grid-327615/bitnami/spark:1
The push refers to repository [gcr.io/genuine-grid-327615/bitnami/spark]
572bf3e81bc: Pushed
6ef99e19e35: Pushed
51feae1f1d69: Pushed
2cc2963a3da: Pushed
a31da56e69e: Pushed
d9bb1c6effa: Pushed
f5d1f7c2c53a: Pushed
353863c967e: Pushed
e5996b4e039: Pushed
75a52abf3b1: Layer already exists

```

#### 4. Create deployment file as follows -

```

$ spark_deployment.yaml > ...
1   apiVersion: apps/v1
2   kind: Deployment
3   metadata:
4     name: spark
5     labels:
6       app: spark
7   spec:
8     selector:
9       matchLabels:
10      app: spark
11    replicas: 2
12    minReadySeconds: 15
13    strategy:
14      type: RollingUpdate
15      rollingUpdate:
16        maxUnavailable: 1
17        maxSurge: 1
18    template:
19      metadata:
20        labels:
21          app: spark
22    spec:
23      containers:
24        - image: bitnami/spark
25          imagePullPolicy: Always
26          name: spark
27          ports:
28            - containerPort: 8080
29

```

#### 5. kubectl apply -f spark\_deployment.yaml

6. Create service file as follows, again we are using loadbalancer service so that we get a public IP which can be tested in the browser

```

spark_service.yaml > ...
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: spark
5  spec:
6    type: LoadBalancer
7    ports:
8      - port: 8080
9        protocol: TCP
10       targetPort: 8080
11    selector:
12      app: spark

```

7. `kubectl apply -f spark_service.yaml`

```

1: digest: sha256:373048f025775d4f8b093f1fa704e87f8603469e5bfef5068c18855c00affcal size: 2425
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f spark_deployment.yaml
deployment.apps/spark configured
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f spark_service.yaml
service/spark unchanged

```

8. We see the following output on the public IP

The screenshot shows the Spark UI interface. At the top, it says "Spark 3.2.0" and the URL "Spark Master at spark://spark-f9bd9c64f-8x7lj:7077". Below this, there's a summary section with metrics:

- URL: spark://spark-f9bd9c64f-8x7lj:7077
- Alive Workers: 0
- Cores in use: 0 Total, 0 Used
- Memory in use: 0.0 B Total, 0.0 B Used
- Resources in use:
- Applications: 0 Running, 0 Completed
- Drivers: 0 Running, 0 Completed
- Status: ALIVE

Below the summary are three expandable sections:

- Workers (0)**: A table with columns: Worker Id, Address, State, Cores, Memory, Resources.
- Running Applications (0)**: A table with columns: Application ID, Name, Cores, Memory per Executor, Resources Per Executor, Submitted Time, User, State, Duration.
- Completed Applications (0)**: A table with columns: Application ID, Name, Cores, Memory per Executor, Resources Per Executor, Submitted Time, User, State, Duration.

## Apache Hadoop

1. `docker pull bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-jar8`
2. `docker tag bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-jar8 gcr.io/genuine-grid-327615/bde2020/hadoop-namenode:1`

3. docker push gcr.io/genuine-grid-327615/bde2020/hadoop-namenode:1
4. docker pull bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
5. docker tag bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8  
gcr.io/genuine-grid-327615/bde2020/hadoop-datanode:1
6. docker push gcr.io/genuine-grid-327615/bde2020/hadoop-datanode:1

```
7948c3e5790c: Layer already exists
1: digest: sha256:51x4d9293ec52083c503ef0aab00c3dd7d6335ddf495cc1257f97a272cab4c0 size: 3033
adividivedi96@cloudshell1:~/clou-infra/Project (genuine-grid-327615)$ docker pull bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
2.0.0-hadoop3.2.1-java8: Pulling from bde2020/hadoop-datanode
3192219afdf04: Already exists
7127a1d8cccd: Already exists
883a89599000: Already exists
77920a3e824f: Already exists
92329e01aee4: Already exists
f373219fec59: Already exists
aa53513fe997: Already exists
8b1800105b98: Already exists
c3a84a3e49c8: Already exists
a65640a64a76: Already exists
3ca2ec07978c: Pull complete
26c2dd45430e: Pull complete
13c9c87a46c: Pull complete
Digest: sha256:ddff6e9ad5af4f73d2ccb6d31d8e3331ff894d5f046126db4f40aa348d484bf
Status: Downloaded newer image for bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
docker.io/bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
adividivedi96@cloudshell1:~/clou-infra/Project (genuine-grid-327615)$ docker tag bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8 gcr.io/genuine-grid-327615/bde2020/hadoop-datanode:1
The push refers to repository [gcr.io/genuine-grid-327615/bde2020/hadoop-datanode]
6a6a167fd39: Pushed
70da79e16472: Pushed
```

7. Create deployment yaml for master apache hadoop node as follows. Here we have env section which sets our hadoop cluster's name, hdfs location and other environment variables picked up from docker-compose's env file [here](#). The internal port to be exposed is 9870 and 9000. Replicas are also set to 1 for 1 master node -

```
hadoop_master.yaml X  hadoop_worker.yaml  hadoop_worker_service.yaml
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: master-hadoop
5   labels:
6     app: master-hadoop
7 spec:
8   selector:
9     matchLabels:
10    app: master-hadoop
11   replicas: 1
12   minReadySeconds: 15
13   strategy:
14     type: RollingUpdate
15     rollingUpdate:
16       maxUnavailable: 1
17       maxSurge: 1
18   template:
19     metadata:
20       labels:
21         app: master-hadoop
22     spec:
23       containers:
24         - image: bde2020/hadoop-namenode
25           imagePullPolicy: Always
26           name: master-hadoop
27           env:
28             - name: CLUSTER_NAME
29               value: "test"
30             - name: CORE_CONF_fs_defaultFS
```

```

27   env:
28     - name: CLUSTER_NAME
29       value: "test"
30     - name: CORE_CONF_fs_defaultFS
31       value: "hdfs://master-hadoop:9000"
32     - name: CORE_CONF_hadoop_http_staticuser_use
33       value: "root"
34     - name: CORE_CONF_hadoop_proxyuser_hue_hosts
35       value: "*"
36     - name: CORE_CONF_hadoop_proxyuser_hue_groups
37       value: "*"
38     - name: CORE_CONF_io_compression_codecs
39       value: "org.apache.hadoop.io.compress.SnappyCodec"
40     - name: HDFS_CONF_dfs_webhdfs_enabled
41       value: "true"
42     - name: HDFS_CONF_dfs_permissions_enabled
43       value: "false"
44     - name: HDFS_CONF_dfs_namenode_datanode_registration_ip__hostname__check
45       value: "false"
46   ports:
47     - containerPort: 9870
48     - containerPort: 9000
49

```

8. kubectl apply -f hadoop\_master.yaml
9. Create the service file for hadoop master as follows. Here again we have a loadbalancer service which maps internal port 9870 to external 9870 and similarly for port 9000 -

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: master-hadoop
5  spec:
6    type: LoadBalancer
7    ports:
8      - port: 9870
9        name: port1
10       protocol: TCP
11       targetPort: 9870
12      - port: 9000
13        name: port2
14        protocol: TCP
15        targetPort: 9000
16    selector:
17      app: master-hadoop

```

10. kubectl apply -f hadoop\_master\_service.yaml

```

adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f hadoop_master.yaml
deployment.apps/master-hadoop created
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f hadoop_master_service.yaml
service/hadoop-master created

```

11. Create worker hadoop deployment as follows. Here there is an environment variable called Service\_Precondition which points the hadoop worker to master using kubedns and its value is set to master-hadoop service that we deployed in the previous step. We copy the other remaining environment variables similar to the previous step from docker compose's env file. Here we have set the replicas as 2 for two data-nodes.

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: worker-hadoop
5  labels:
6    app: worker-hadoop
7  spec:
8    selector:
9      matchLabels:
10     app: worker-hadoop
11    replicas: 2
12    minReadySeconds: 15
13    strategy:
14      type: RollingUpdate
15      rollingUpdate:
16        maxUnavailable: 1
17        maxSurge: 1
18    template:
19      metadata:
20        labels:
21          app: worker-hadoop
22      spec:
23        containers:
24          - image: bde2020/hadoop-datanode
25            imagePullPolicy: Always
26            name: worker-hadoop
27            env:
28              - name: SERVICE_PRECONDITION
29                value: master-hadoop:9000

```

			- name: CORE_CONF_fs_defaultFS   value: "hdfs://master-hadoop:9000" - name: CORE_CONF_hadoop_http_staticuser_use   value: "root" - name: CORE_CONF_hadoop_proxyuser_hue_hosts   value: "*" - name: CORE_CONF_hadoop_proxyuser_hue_groups   value: "*" - name: CORE_CONF_io_compression_codecs   value: "org.apache.hadoop.io.compress.SnappyCodec" - name: HDFS_CONF_dfs_webhdfs_enabled   value: "true" - name: HDFS_CONF_dfs_permissions_enabled   value: "false" - name: HDFS_CONF_dfs_namenode_datanode_registration_ip_hostname_check   value: "false"
--	--	--	--

## 12. kubectl apply -f hadoop\_worker.yaml

```

deployment.apps/worker-hadoop configured
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f hadoop_worker.yaml
deployment.apps/worker-hadoop configured

```

## 13. We get the following hadoop master view on its public IP

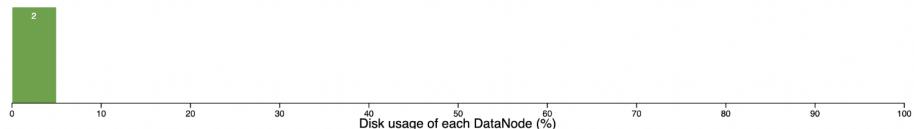
Not Secure | 34.136.215.85:9870/dfshealth.html#tab-datanode

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

## Datanode Information

✓ In service   
 ● Down   
 ○ Decommissioning   
 ○ Decommissioned   
 ○ Decommissioned & dead  
↗ Entering Maintenance   
 ⚡ In Maintenance   
 ⚡ In Maintenance & dead

Datanode usage histogram



### In operation

Show 25 entries											
		Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version		

Not Secure | 34.136.215.85:9870/dfshealth.html#tab-overview

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

## Overview 'master-hadoop:9000' (active)

Started:	Tue Nov 23 17:00:28 -0600 2021
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 10:56:00 -0500 2019 by rohitsharmaks from branch-3.2.1
Cluster ID:	CID-186cc3e1-bf43-4f68-a79c-d56984e4c605
Block Pool ID:	BP-705668945-10.8.1.13-1637708424685

## Summary

Security is off.  
Safemode is off.  
1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).  
Heap Memory used 60.44 MB of 114.19 MB Heap Memory. Max Heap Memory is 1.76 GB.  
Non Heap Memory used 48.41 MB of 49.63 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	188.58 GB
Configured Remote Capacity:	0 B

## Sonarqube and Sonar-Scanner

1. docker pull adityadw/sonarqube\_sonarscanner
2. docker tag adityadw/sonarqube\_sonarscanner:latest  
gcr.io/genuine-grid-327615/adityadw/sonarqube\_sonarscanner:1
3. docker push gcr.io/genuine-grid-327615/adityadw/sonarqube\_sonarscanner:1

```
adidivedi96@cloudshell:~/clou-infra/Project_(genuine-grid-327615)$ docker tag adityadw/sonarqube_sonarscanner:latest gcr.io/genuine-grid-327615/adityadw/sonarqube_sonarscanner:1
adidivedi96@cloudshell:~/clou-infra/Project_(genuine-grid-327615)$ docker push gcr.io/genuine-grid-327615/adityadw/sonarqube_sonarscanner:1
The push refers to repository [gcr.io/genuine-grid-327615/adityadw/sonarqube_sonarscanner]
6ccb66713d8e: Layer already exists
e30e2f421d4ed: Layer already exists
93d8874a9cc2: Layer already exists
e2eb06d8af82: Layer already exists
1: digest: sha256:76354bd0d878124ed3a9160874a078cfdc2020144dea399c32a40d7a2d0acb3e5 size: 1159
adidivedi96@cloudshell:~/clou-infra/Project_(genuine-grid-327615)$
```

4. Create the sonarqube deployment as follows, exposing internal port 9000

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: sonar
5    labels:
6      app: sonar
7  spec:
8    selector:
9      matchLabels:
10     | app: sonar
11   replicas: 2
12   minReadySeconds: 15
13   strategy:
14     type: RollingUpdate
15     rollingUpdate:
16       maxUnavailable: 1
17       maxSurge: 1
18   template:
19     metadata:
20       labels:
21         app: sonar
22     spec:
23       containers:
24         - image: adityadw/sonarqube_sonarscanner
25           imagePullPolicy: Always
26           name: sonar
27           ports:
28             - containerPort: 9000
```

5. kubectl apply -f sonarqube\_deployment.yaml

6. Create sonarqube service as follows, mapping internal port 9000 to external port 9000. It's of type LoadBalancer so that it can be accessed via public IP on browser for testing

```
sonarqube_service.yaml > ...
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: sonar
5  spec:
6    type: LoadBalancer
7    ports:
8      - port: 9000
9        protocol: TCP
10       targetPort: 9000
11    selector:
12      app: sonar
```

7. kubectl apply -f sonarqube\_service.yaml

```
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f sonarqube_deployment.yaml
deployment.apps/sonar created
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f sonarqube_service.yaml
service/sonar created
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ █
```

8. We get the following output when we visit the sonarqube

## 9. We can run sonar-scanner commands within the pod and get results as seen -

```

bash-5.1# sonar-scanner -D sonar.password='helloworld'
INFO: Scanner configuration file: /opt/sonarqube/sonar-scanner-4.6.2.2472/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.6.2.2472
INFO: Java 11.0.11 Alpine (64-bit)
INFO: Linux 5.4.144+ amd64
INFO: User cache: /root/.sonar/cache
INFO: Scanner configuration file: /opt/sonarqube/sonar-scanner-4.6.2.2472/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: Analyzing on SonarQube server 9.1.0
INFO: Default locale: "en_US", source code encoding: "UTF-8"
INFO: Load global settings
INFO: Load global settings (done) | time=338ms
INFO: Server id: BF41A1F2-AXzF2NSm20tCr7wvP_on
INFO: User cache: /root/.sonar/cache
INFO: Load/download plugins
INFO: Load plugins index
INFO: Load plugins index (done) | time=255ms
INFO: Load/download plugins (done) | time=6196ms
INFO: Process project properties
INFO: Process project properties (done) | time=2ms
INFO: Execute project builders
INFO: Execute project builders (done) | time=4ms
INFO: Project key: myProject
INFO: Base dir: /opt/sonarqube
INFO: Working dir: /opt/sonarqube/.scannerwork
INFO: Load project settings for component key: 'myProject'
INFO: Load quality profiles
INFO: Load quality profiles (done) | time=677ms
INFO: Load active rules
INFO: Load active rules (done) | time=8233ms
WARN: SCM provider autodetection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable the SCM Sensor in the project settings.
INFO: Indexing files...
INFO: Project configuration:
INFO: Load project repositories

```

## 10. To do so the commands used are -

- a. kubectl exec --stdin --tty sonar-56b9b86486-t82gv -- /bin/bash
- b. sonar-scanner -D sonar.password='password'

## GUI

1. docker pull adityadw/gui
2. docker tag adityadw/gui:latest gcr.io/genuine-grid-327615/adityadw/gui:1
3. docker push gcr.io/genuine-grid-327615/adityadw/gui:1

```

△ Problems  ✘ genuine-grid-327615 ×
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker pull adityadw/gui
Using default tag: latest
latest: Pulling from adityadw/gui
647acf3d48c2: Pull complete
b02967ef0034: Pull complete
e1ad2231829e: Pull complete
5576ce26bf1d: Pull complete
a66b7f31b095: Pull complete
05189b5b2762: Pull complete
af08e8fd006: Pull complete
287d56f7527b: Pull complete
dc0580965fb6: Pull complete
f7974ddcbcfa8a: Pull complete
5762174dac3a: Pull complete
b875051cce3f: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:13f3ba3455ced4d5368f2c48a472984e627d03e6e48b23df3366e1df0f199631
Status: Downloaded newer image for adityadw/gui:latest
docker.io/adityadw/gui:latest
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker tag adityadw/gui:latest gcr.io/genuine-grid-327615/adityadw/gui:1
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker push gcr.io/genuine-grid-327615/adityadw/gui:1
The push refers to repository [gcr.io/genuine-grid-327615/adityadw/gui]
5f70bf18a086: Layer already exists
a32d630a773f: Pushed
074a518b6d67: Pushed
883d8ad12854: Pushed
42c6e1e841fa: Layer already exists
a9b125f164c3: Layer already exists
e24045f8c247: Layer already exists
b7b662b31e70: Layer already exists
6f5234c0aacd: Layer already exists
8a5844586fdb: Layer already exists
a4ab4e659b40: Layer already exists
549f2905579: Layer already exists
a36ba9e322f7: Layer already exists
1: digest: sha256:13f3ba3455ced4d5368f2c48a472984e627d03e6e48b23df3366e1df0f199631 size: 3054
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$

```

4. Create deployment yaml file as follows with image referring to above imported image and environment variables which set the public IPs for hadoop, spark, jupyter and sonar respectively declared in the dockerfile and container port as 8080

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: gui
5  labels:
6    app: gui
7  spec:
8    selector:
9      matchLabels:
0        app: gui
1        replicas: 2
2        minReadySeconds: 15
3        strategy:
4          type: RollingUpdate
5          rollingUpdate:
6            maxUnavailable: 1
7            maxSurge: 1
8        template:
9          metadata:
0            labels:
1              app: gui
2            spec:
3              containers:
4                - image: adityadw/gui
5                  imagePullPolicy: Always
6                  name: gui
7                  env:
8                    - name: HADOOP_URL
9                      value: http://34.136.215.85:9870/
0                    - name: SPARK_URL

```

```

    spec:
      containers:
        - image: adityadw/gui
          imagePullPolicy: Always
          name: gui
          env:
            - name: HADOOP_URL
              value: http://34.136.215.85:9870/
            - name: SPARK_URL
              value: http://34.135.56.132:8080/
            - name: JUPYTER_URL
              value: http://34.133.108.43:8888/
            - name: SONAR_URL
              value: http://35.222.228.36:9000/
          ports:
            - containerPort: 8080

```

5. Create a load-balancer service to expose using public IP and map external port 8080 to target port 8080

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: gui
5  spec:
6    type: LoadBalancer
7    ports:
8      - port: 8080
9        name: port
10       protocol: TCP
11       targetPort: 8080
12    selector:
13      app: gui

```

6. `kubectl apply -f gui_deployment.yaml`

7. `kubectl apply -f gui_service.yaml`

```

adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f gui_deployment.yaml
deployment.apps/gui configured
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f gui_service.yaml
service/gui unchanged

```

The GUI loads as following webpage -

## Welcome to Big Data Processing Application

Please click on link to chose which application you would like to run

1. [Apache Hadoop](#)
2. [Apache Spark](#)
3. [Jupyter Notebook](#)
4. [SonarQube and SonarScanner](#)

## Deployment output

<input type="checkbox"/>	Name ↑	Status	Type	Endpoints	Pods	Namespace	Clusters	
<input type="checkbox"/>	gui	✓ OK	External load balancer	35.192.45.227:8080	2/2	default	adityadwkubernetescluster	
<input type="checkbox"/>	jupyter	✓ OK	External load balancer	34.133.108.43:8888	1/1	default	adityadwkubernetescluster	
<input type="checkbox"/>	master-hadoop	✓ OK	External load balancer	34.136.215.85:9870	1/1	default	adityadwkubernetescluster	▼
<input type="checkbox"/>	sonar	✓ OK	External load balancer	35.222.228.36:9000	1/1	default	adityadwkubernetescluster	
<input type="checkbox"/>	spark	✓ OK	External load balancer	34.135.56.132:8080	2/2	default	adityadwkubernetescluster	

Start your free trial with \$300 in credit. Don't worry – you won't be charged if you run out of credit. [Learn more](#)

[DISMISS](#) [ACTIVATE](#)

Google Cloud Platform My First Project Search products and resources

Kubernetes Engine Workloads [REFRESH](#) [DEPLOY](#) [DELETE](#) [OPERATIONS](#)

Clusters Workloads Services & Ingress Applications Configuration Storage Object Browser Migrate to containers Config Management Marketplace Release notes

Workloads are deployable units of computing that can be created and managed in a cluster.

OVERVIEW COST OPTIMISATION PREVIEW

Filter Is system object : False Filter workloads

<input type="checkbox"/>	Name ↑	Status	Type	Pods	Namespace	Cluster
<input type="checkbox"/>	gui	✓ OK	Deployment	2/2	default	adityadwkubernetescluster
<input type="checkbox"/>	jupyter	✓ OK	Deployment	1/1	default	adityadwkubernetescluster
<input type="checkbox"/>	master-hadoop	✓ OK	Deployment	1/1	default	adityadwkubernetescluster
<input type="checkbox"/>	sonar	✓ OK	Deployment	1/1	default	adityadwkubernetescluster
<input type="checkbox"/>	spark	✓ OK	Deployment	2/2	default	adityadwkubernetescluster
<input type="checkbox"/>	worker-hadoop	✓ OK	Deployment	2/2	default	adityadwkubernetescluster

```
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl get svc
NAME          TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)           AGE
gui           LoadBalancer  10.12.3.188  35.192.45.227  8080:30374/TCP  37m
jupyter       LoadBalancer  10.12.6.178  34.133.108.43  8888:32581/TCP  95m
kubernetes    ClusterIP    10.12.0.1    <none>        443/TCP          27d
master-hadoop LoadBalancer  10.12.10.131  34.136.215.85  9870:31891/TCP,9000:30961/TCP  27d
sonar         LoadBalancer  10.12.15.141  35.222.228.36  9000:32661/TCP  27d
spark          LoadBalancer  10.12.6.231   34.135.56.132  8080:30921/TCP  27d
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$
```



## Welcome to Big Data Processing Application

Please click on link to chose which application you would like to run

1. Apache Hadoop
2. Apache Spark
3. Jupyter Notebook
4. SonarQube and SonarScanner

## Building docker images

### Jupyter Notebook

1. docker pull mcr.microsoft.com/mmlspark/release
2. Create the following docker file to automate running of jupyter notebook

```
1 FROM mcr.microsoft.com/mmlspark/release
2
3 CMD ["jupyter", "notebook"]
```

3. docker build -t adityadw/jupyter\_notebook .
4. docker run -it -p 8888:8888 adityadw/jupyter\_notebook
5. docker push adityadw/jupyter\_notebook

### Apache Spark

1. docker pull bitnami/spark
- URL - <https://hub.docker.com/r/bitnami/spark/>

### Apache Hadoop

1. docker pull bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
2. docker pull bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

URLs - <https://hub.docker.com/r/bde2020/hadoop-namenode>,  
<https://hub.docker.com/r/bde2020/hadoop-datanode>

## SonarQube and Sonarscanner

1. docker pull sonarqube
2. Download sonar-scanner for any platform which requires pre-installed JVM from here  
<https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.6.2.2472.zip>
3. Edit sonar-scanner configuration file located under config folder as follows -

```
#Configure here general information about the environment, such as SonarQube server connection details for example
#No information about specific project should appear here

[----- Default SonarQube server
sonar.host.url=http://localhost:9000

[----- Default source code encoding
sonar.sourceEncoding=UTF-8

[Default sonarqube connection and project config
sonar.login=admin
sonar.password=admin
sonar.projectKey=myProject
```

4. Copy the sonar-scanner directory to container and set the PATH variable to bin directory of sonar-scanner as follows

```
1 FROM sonarqube
2 COPY . /opt/sonarqube
3 ENV PATH=$PATH:/opt/sonarqube/sonar-scanner-4.6.2.2472/bin/|
```

5. Sonar-scanner command works in container as follows -

```
[root@adityadw ~]# docker exec -it 99731d8386ea80a04e9707327eda66d645685f197b038660b36334751643a348 /bin/sh
/opt/sonarqube # sonar-scanner
INFO: Scanner configuration file: /opt/sonarqube/sonar-scanner-4.6.2.2472/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.6.2.2472
INFO: Java 11.0.11 Alpine (64-bit)
INFO: Linux 5.10.16.3-microsoft-standard-WSL2 amd64
INFO: User cache: /root/.sonar/cache
INFO: Scanner configuration file: /opt/sonarqube/sonar-scanner-4.6.2.2472/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: Analyzing on SonarQube server 9.1.0
INFO: Default locale: "en_US", source code encoding: "UTF-8"
INFO: Load global settings
INFO: Load global settings (done) | time=398ms
INFO: Server id: BF41A1P2-AZzFDRLqwjnl2POX-sxl
INFO: User cache: /root/.sonar/cache
INFO: Load/download plugins
INFO: Load plugins index
INFO: Load plugins index (done) | time=319ms
INFO: Load/download plugins (done) | time=5299ms
INFO: Process project properties
INFO: Process project properties (done) | time=1ms
INFO: Execute project builders
INFO: Execute project builders (done) | time=3ms
INFO: Project key: myProject
INFO: Base dir: /opt/sonarqube
INFO: Working dir: /opt/sonarqube/.scannerwork
INFO: Load project settings for component key: 'myProject'
INFO: Load quality profiles
INFO: Load quality profiles (done) | time=456ms
INFO: Load active rules
INFO: Load active rules (done) | time=5978ms
WARN: SCM provider autodetection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable the SCM Sensor in the project settings.
INFO: Indexing files...
INFO: Project configuration:
INFO: Load project repositories
INFO: Load project repositories (done) | time=183ms
INFO: 1003 files indexed
INFO: Quality profile for js: Sonar way
INFO: Quality profile for web: Sonar way
INFO: Quality profile for xml: Sonar way
INFO: ----- Run sensors on module myProject
INFO: Load metrics repository
```

6. docker build -t adityadw/sonarqube\_sonarscanner .

7. docker push adityadw/sonarqube\_sonarscanner

## Terminal

Created python program to simulate the terminal

Created a dockerfile from python base image to run the script

docker build -t adityadw/terminal .

docker push adityadw/terminal

## GUI

1. The following code sets up a flask app which returns HTML for the GUI, the URLs are passed as environment variables mentioned in the dockerfile and kubernetes deployment file. The requests library is used to check if the services are up and running

```

Users > adityadwivedi > Desktop > Aditya > Cloud-Infra > gui > gui.py
 1  #reference - https://flask.palletsprojects.com/en/2.0.x/quickstart/
 2  from flask import Flask
 3  import sys,requests,time,os
 4
 5  app = Flask(__name__)
 6  hadoop_url=''
 7  spark_url=''
 8  jupyter_url=''
 9  sonarqube_url=''
10
11 #function to check if the micro-services are up and running by making the get request to their addresses
12 def checkServices():
13     hadoop=requests.get(hadoop_url).status_code
14     spark=requests.get(spark_url).status_code
15     jupyter=requests.get(jupyter_url).status_code
16     sonarqube=requests.get(sonarqube_url).status_code
17
18     if hadoop==200 and spark==200 and jupyter==200 and sonarqube==200:
19         return True
20     else:
21         return False
22
23
24 @app.route("/")
25 def terminal():
26     #returning the webpage with hyperlinks
27     return """

# Welcome to Big Data Processing Application </h1> <p> Please click on link to chose which application you would like to run </p>""" 28 +"<ol> <li> <a href='"+hadoop_url+"> Apache Hadoop</a></li> <li> <a href='"+spark_url+"> Apache Spark </a></li> <li> <a href='"+jupyter_url+"> Jupyter Notebook</li>"" 29 30 #setting the urls using env variables 31 hadoop_url=os.environ['HADOOP_URL'] 32 spark_url=os.environ['SPARK_URL'] 33 jupyter_url=os.environ['JUPYTER_URL'] 34 sonarqube_url=os.environ['SONAR_URL'] 35 36 #check every 3 seconds if micro-services are up or not 37 while(checkServices()==False): 38 print('Services not yet up') 39 time.sleep(3) 40 41 app.run(host='0.0.0.0',port=8080)


```

Restart Visual Studio Code to apply the latest update

2. Create dockerfile as follows based on python image which sets up flask and requests requirement and declare the environment variables as follows and expose port 8080

```

1  FROM python:latest
2  RUN pip install Flask
3  RUN pip install requests
4  COPY . /usr/src/myapp
5  WORKDIR /usr/src/myapp
6  ENV HADOOP_URL http://34.136.215.85:9870/
7  ENV SPARK_URL http://34.135.56.132:8080/
8  ENV JUPYTER_URL http://34.133.108.43:8888/
9  ENV SONAR_URL http://35.222.228.36:9000/
10 EXPOSE 8080
11 CMD ["python3","gui.py"]

```

3. docker build -t adityadw/gui .
4. docker push adityadw/gui

```
[adityadwivedi@Adityas-MBP-2 gui % docker build -t adityadw/gui .
[+] Building 0.4s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 84B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:latest
=> [1/6] FROM docker.io/library/python:latest@sha256:f44726de10d15558e465238b02966a8f83971fd85a4c4b95c263704e6a6012e9
=> [internal] load build context
=> => transferring context: 152B
=> CACHED [2/5] RUN pip install Flask
=> CACHED [3/5] RUN pip install requests
=> CACHED [4/5] COPY . /usr/src/myapp
=> CACHED [5/5] WORKDIR /usr/src/myapp
=> exporting to image
=> => exporting layers
=> => writing image sha256:d2557dfde364717f6414d45dac3e1e89c06e2f4efb3ad6d871e4119a6acbf8b2
=> => naming to docker.io/adityadw/gui

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[adityadwivedi@Adityas-MBP-2 gui % docker push adityadw/gui
Using default tag: latest
The push refers to repository [docker.io/adityadw/gui]
5f70bf18a086: Mounted from adityadw/hellopython
3740e6ede33cf: Pushed
e54ffa795d88: Pushed
a8d9768594b1: Pushed
7458338b205: Mounted from library/python
aa38cdf9d933: Mounted from library/python
b900d3bf0aaa: Mounted from library/python
5abf2e7a4ccc: Mounted from library/python
53dc4741a24f: Mounted from library/python
f0214314e91b: Mounted from library/python
3c460cc7630f: Mounted from library/python
53c92304892e: Mounted from library/python
4aeb411cd144: Mounted from library/python
latest: digest: sha256:926ff83431507c712e3e5356cc3fed84c701dff50f6e67cb004c350ba3fd34 size: 3054
adityadwivedi@Adityas-MBP-2 gui % ]
```

## References

- [https://hub.docker.com/\\_/microsoft-mmlspark-release](https://hub.docker.com/_/microsoft-mmlspark-release),
- <https://github.com/big-data-europe/docker-hadoop/blob/master/docker-compose.yml>,
- <https://www.datamechanics.co/blog-post/optimized-spark-docker-images-now-available>,
- <https://computingforgeeks.com/how-to-install-apache-spark-on-ubuntu-debian/>,
- <https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/>
- <https://www.geeksforgeeks.org/python-3-input-function/>
- <https://stackoverflow.com/questions/27093612/in-a-dockerfile-how-to-update-path-environment-variable>
- <https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/>
- <https://www.edureka.co/community/7415/copy-is-not-working-in-docker>
- <https://github.com/mohamedfarag/14-848-extra-credit-project>
- <https://github.com/rinormaloku/k8s-mastery/tree/master/resource-manifests>
- <https://kubernetes.io/docs/tasks/debug-application-cluster/get-shell-running-container/>
- <https://github.com/big-data-europe/docker-hadoop>
- <https://flask.palletsprojects.com/en/2.0.x/quickstart/>
- <https://github.com/rinormaloku/k8s-mastery/blob/master/sa-webapp/Dockerfile>
- [https://www.w3schools.com/html/html\\_lists.asp](https://www.w3schools.com/html/html_lists.asp)
- <https://piazza.com/class/ksnwtglzkui2vr?cid=109>

