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 adityadwkubernetescluster
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

```
adidwivedi96@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
adidwivedi96@cloudshell:~ (genuine-grid-327615)$ docker tag adityadw/jupyter_notebook:latest gcr.io/genuine-grid-327615/adityadw/jupyter_notebook:1
adidwivedi96@cloudshell:~ (genuine-grid-327615)$ docker push gcr.io/genuine-grid-327615/adityadw/jupyter_notebook:1
The push refers to repository [gcr.io/genuine-grid-327615/adityadw/jupyter_notebook]
38301074c5ed: Pushed
aa81112ca16f: Pushed
2889f2c4cf15: Pushed
2d652f97bd28: Pushed
824bf068fd3d: Layer already exists
1: digest: sha256:758ab9230acb3707902c3bc1c2604b711dd570c24eee6925dcc85a4028582b81 size: 1371
```

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

jupyter_notebook_deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jupyter
  labels:
    app: jupyter
spec:
  selector:
    matchLabels:
      app: jupyter
  replicas: 2
  minReadySeconds: 15
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: jupyter
    spec:
      containers:
        - image: adityadw/jupyter_notebook
          imagePullPolicy: Always
          name: jupyter
          ports:
            - containerPort: 8888

```

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 × jupyter_notebook_deployment.yaml

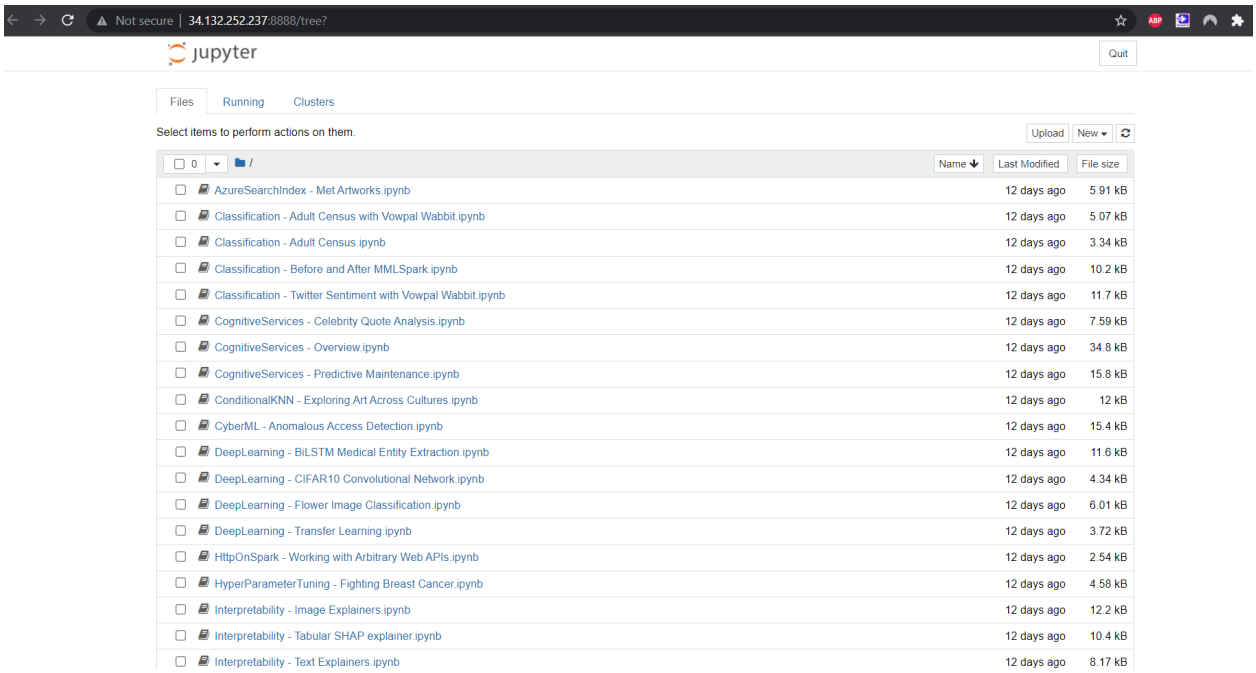
jupyter_service.yaml > ...

```yaml
apiVersion: v1
kind: Service
metadata:
  name: jupyter
spec:
  type: LoadBalancer
  ports:
    - port: 8888
      protocol: TCP
      targetPort: 8888
  selector:
    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

```
ddidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker pull bitnami/spark
Using default tag: latest
latest: Pulling from bitnami/spark
23b664af592e: Pull complete
9da89ddc65be: Pull complete
9d21c74b4228: Pull complete
c0c515956a4b: Pull complete
eb28f6fabedd: Pull complete
251825bf66a9: Pull complete
fcd8e55f83fb: Pull complete
17800b61fa67: Pull complete
189ae2b97f94: Pull complete
178a7ee58f2a: Pull complete
Digest: sha256:373048f025775d4f8b093f1fa704e87f8603469e5bfef5068c18855c00affca1
Status: Downloaded newer image for bitnami/spark:latest
docker.io/bitnami/spark:latest
ddidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker tag bitnami/spark:latest gcr.io/genuine-grid-327615/bitnami/spark:1
ddidwivedi96@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
16ef99e19e35: Pushed
251feaef1d69: Pushed
12cc2963a3da: Pushed
1a31da56e68e: Pushed
1d9bb1c6effa: Pushed
3f5d1f7c253a: Pushed
9353863c9e7e: Pushed
1e5896b4e039: Pushed
f75a52abf3b1: Layer already exists
```

4. Create deployment file as follows -

spark_deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spark
  labels:
    app: spark
spec:
  selector:
    matchLabels:
      app: spark
  replicas: 2
  minReadySeconds: 15
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: spark
    spec:
      containers:
        - image: bitnami/spark
          imagePullPolicy: Always
          name: spark
          ports:
            - containerPort: 8080
```

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

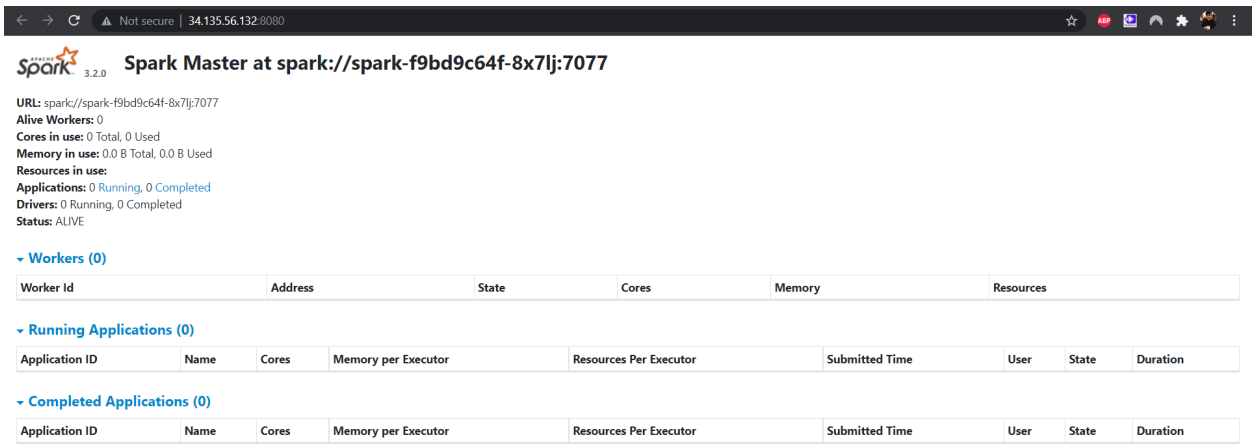
spark_service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: spark
spec:
  type: LoadBalancer
  ports:
    - port: 8080
      protocol: TCP
      targetPort: 8080
  selector:
    app: spark
```

7. kubectl apply -f spark_service.yaml

```
1: digest: sha256:373048f025775d4f8b093f1fa704e87f8603469e5bfef5068c18855c00affca1 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



Apache Hadoop

1. docker pull bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
2. docker tag bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8 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: sha256:51ad9293ec52083c5003ef0aaab00c3dd7d6335ddf495cc1257f97a272cab4c0 size: 3033
adidwivedi96@cloudshell:~/clud-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
3192219afd04: Already exists
7127a1d8ccad: Already exists
883a89599900: Already exists
77920a3e82af: Already exists
92329e81aec4: Already exists
f373218fec59: Already exists
aa53513fe997: Already exists
8b1800105b98: Already exists
c3a84a3e49c8: Already exists
a65640a64a76: Already exists
3ca2ec07878c: Pull complete
26c2dd45430e: Pull complete
13c9c87a46cb: Pull complete
Digest: sha256:ddf6e9ad55af4f73d2ccb6da31d9e3331ffb94d5f046126db4f40aa348d484bf
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
adidwivedi96@cloudshell:~/clud-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
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker push gcr.io/genuine-grid-327615/bde2020/hadoop-datanode:1
The push refers to repository [gcr.io/genuine-grid-327615/bde2020/hadoop-datanode]
6a6a1867fd39: 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 | hadoop_worker.yaml | hadoop_worker_service.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: master-hadoop
  labels:
    app: master-hadoop
spec:
  selector:
    matchLabels:
      app: master-hadoop
  replicas: 1
  minReadySeconds: 15
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: master-hadoop
    spec:
      containers:
        - image: bde2020/hadoop-namenode
          imagePullPolicy: Always
          name: master-hadoop
          env:
            - name: CLUSTER_NAME
              value: "test"
            - name: CORE_CONF_fs_defaultFS
```

```
      env:
        - name: CLUSTER_NAME
          value: "test"
        - 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"
      ports:
        - containerPort: 9870
        - containerPort: 9000
```

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 -

```yaml
apiVersion: v1
kind: Service
metadata:
  name: master-hadoop
spec:
  type: LoadBalancer
  ports:
    - port: 9870
      name: port1
      protocol: TCP
      targetPort: 9870
    - port: 9000
      name: port2
      protocol: TCP
      targetPort: 9000
  selector:
    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
```

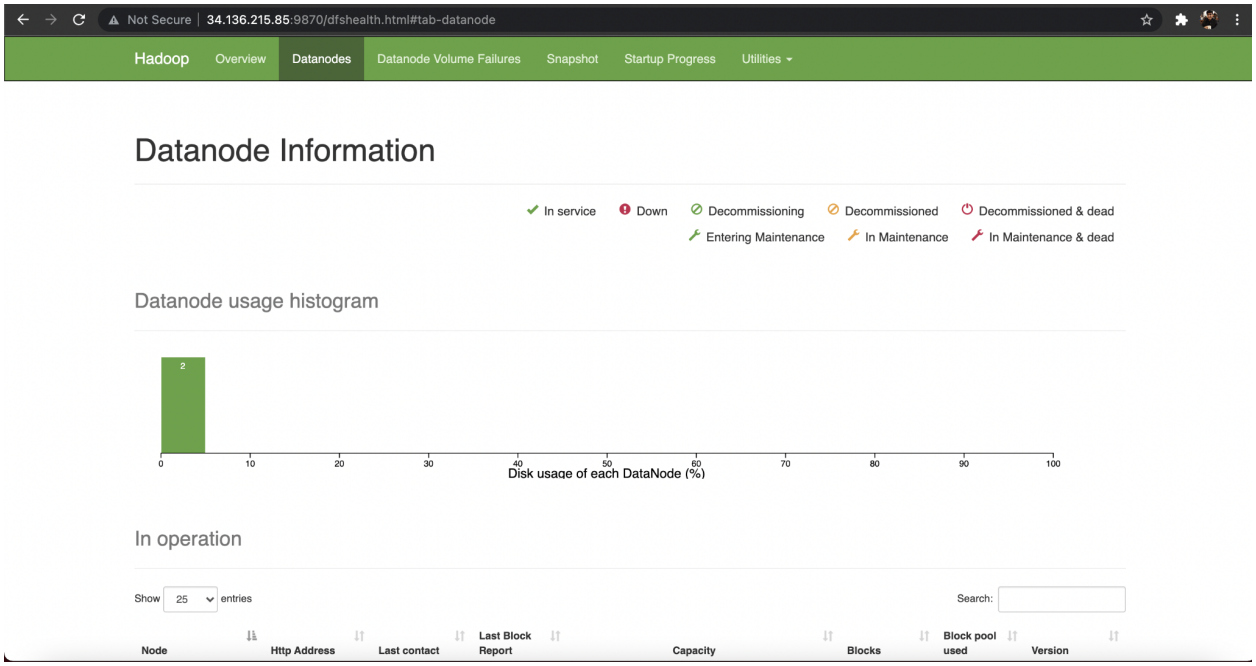
11. Create worker hadoop deployment as follows. Here there is an environment variable called Service_Precondition which points the hadoop worker to master using kubedns and its value is set to master-hadoop service that we deployed in the previous step. We copy the other remaining environment variables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

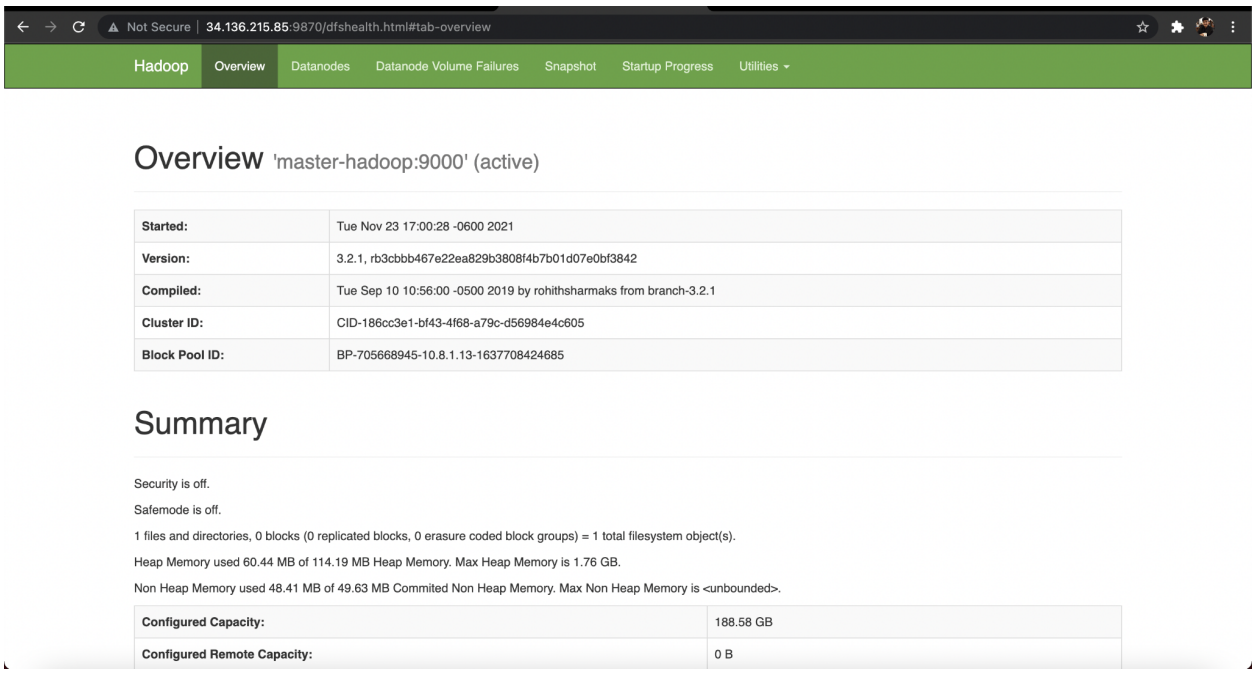
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: worker-hadoop
  labels:
    app: worker-hadoop
spec:
  selector:
    matchLabels:
      app: worker-hadoop
  replicas: 2
  minReadySeconds: 15
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: worker-hadoop
    spec:
      containers:
        - image: bde2020/hadoop-datanode
          imagePullPolicy: Always
          name: worker-hadoop
          env:
            - name: SERVICE_PRECONDITION
              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





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

```
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ docker tag adityadw/sonarqube_sonarscanner:latest gcr.io/genuine-grid-327615/adityadw/sonarqube_sonarscanner:1
adidwivedi96@cloudshell:~/clud-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]
6cbbe6713d8e: Layer already exists
e3e2f421d4ed: Layer already exists
93d8874a9cc2: Layer already exists
e2eb06d8af82: Layer already exists
1: digest: sha256:76354b0d878124ed3a9160874a078cfdc2020144dea399c32a40d7a2d0acb3e5 size: 1159
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$
```

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

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sonar
  labels:
    app: sonar
spec:
  selector:
    matchLabels:
      app: sonar
  replicas: 2
  minReadySeconds: 15
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: sonar
    spec:
      containers:
        - image: adityadw/sonarqube_sonarscanner
          imagePullPolicy: Always
          name: sonar
          ports:
            - 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

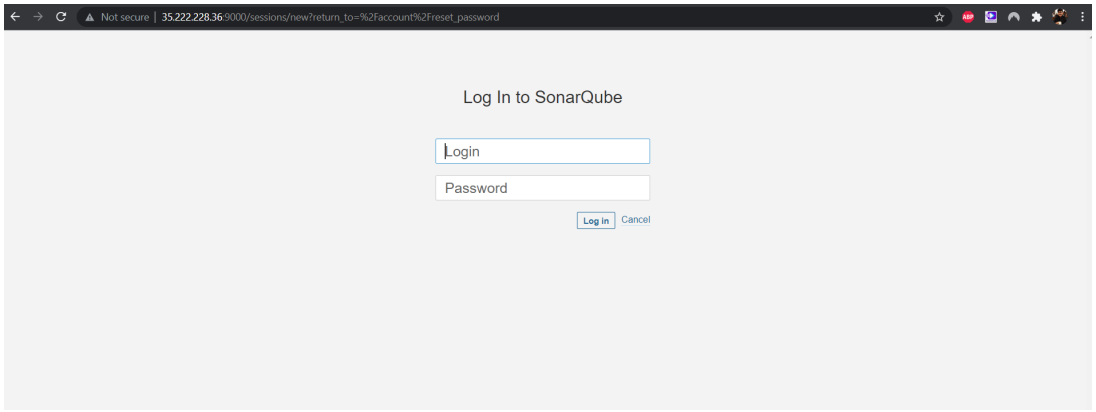
sonarqube_service.yaml > ...

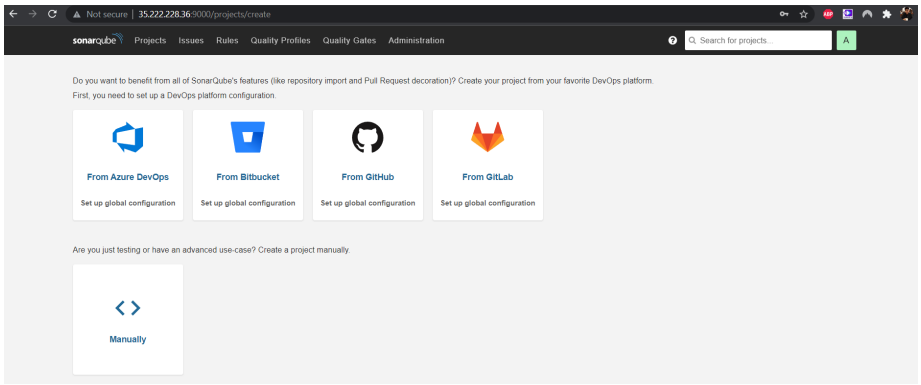
```yaml
apiVersion: v1
kind: Service
metadata:
  name: sonar
spec:
  type: LoadBalancer
  ports:
    - port: 9000
      protocol: TCP
      targetPort: 9000
  selector:
    app: sonar
```

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
af08e8fda0d6: Pull complete
287d56f7527b: Pull complete
dc0580965fb6: Pull complete
f7974dcbcf8a: 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
42ce61e841fa: Layer already exists
a9b125f164c3: Layer already exists
e24045f8c247: Layer already exists
b7b662b31e70: Layer already exists
6f5234c0aacd: Layer already exists
8a5844586fdb: Layer already exists
a4aba4e59b40: Layer already exists
5499f2905579: 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

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gui
  labels:
    app: gui
spec:
  selector:
    matchLabels:
      app: gui
  replicas: 2
  minReadySeconds: 15
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: gui
    spec:
      containers:
        - image: adityadw/gui
          imagePullPolicy: Always
          name: gui
          env:
          - name: HADOOP_URL
            value: http://34.136.215.85:9870/
          - 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
```

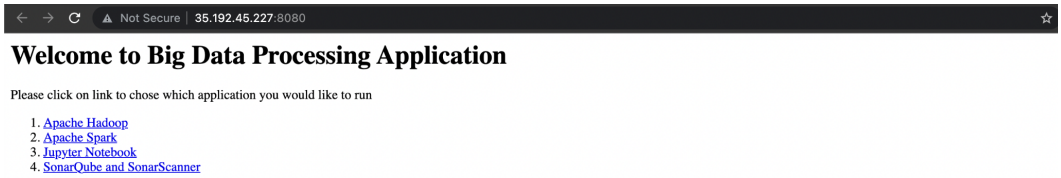
5. Create a load-balancer service to expose using public IP and map external port 8080 to target port 8080

```
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

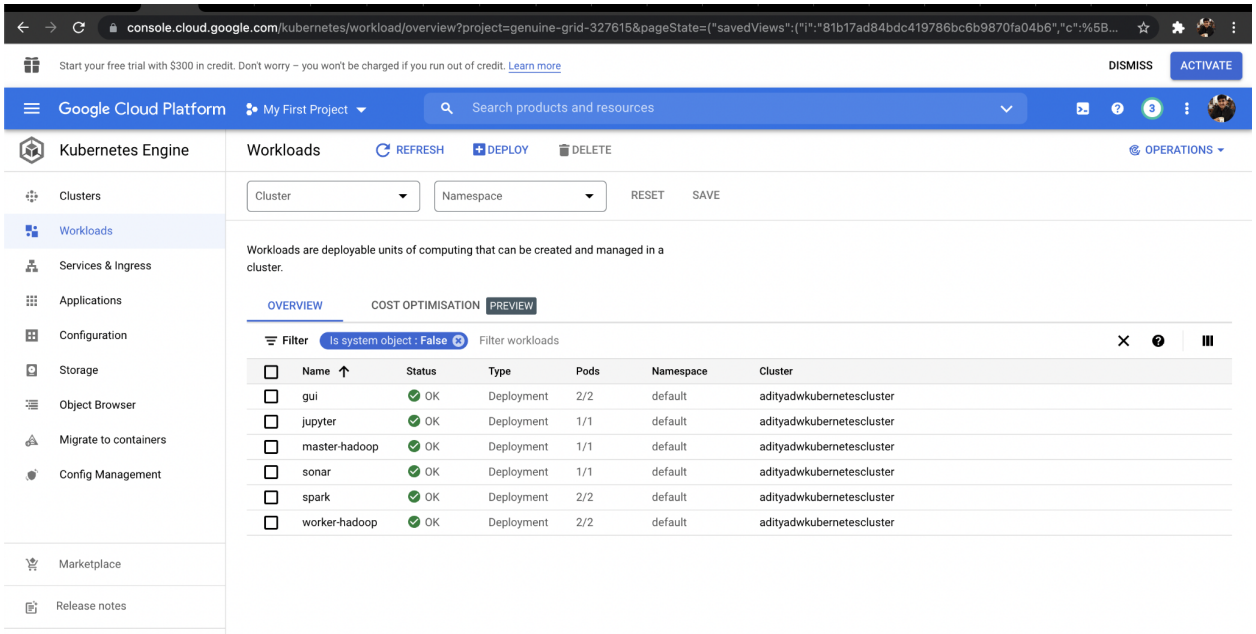
```
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f gui_deployment.yaml
deployment.apps/gui configured
adidwivedi96@cloudshell:~/clud-infra/Project (genuine-grid-327615)$ kubectl apply -f gui_service.yaml
service/gui unchanged
```

The GUI loads as following webpage -

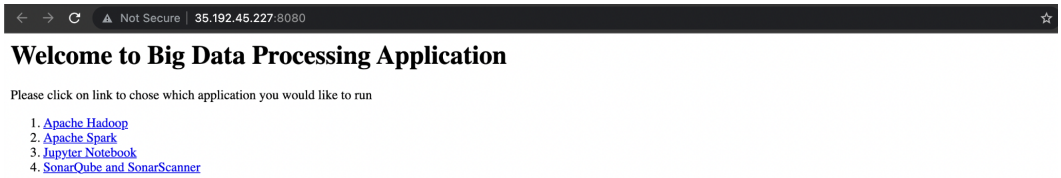


Deployment output

Name	Status	Type	Endpoints	Pods	Namespace	Clusters
gui	OK	External load balancer	35.192.45.227:8080	2/2	default	adityadwkubernetescluster
jupyter	OK	External load balancer	34.133.108.43:8888	1/1	default	adityadwkubernetescluster
master-hadoop	OK	External load balancer	34.136.215.85:9870	1/1	default	adityadwkubernetescluster
sonar	OK	External load balancer	35.222.228.36:9000	1/1	default	adityadwkubernetescluster
spark	OK	External load balancer	34.135.56.132:8080	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)$
```



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

```
FROM mcr.microsoft.com/mmlspark/release

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

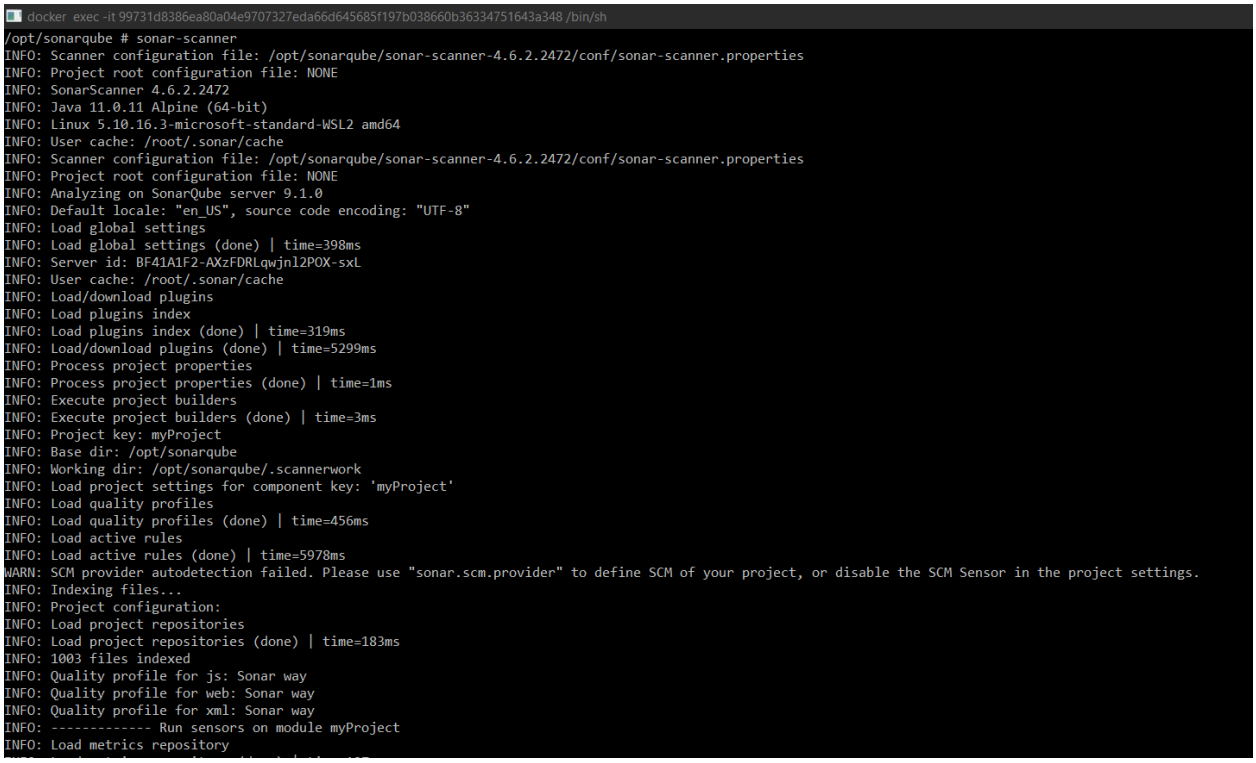
1. docker pull sonarqube
2. Download sonar-scanner for any platform which requires pre-installed JVM from here https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.6.2.2472.zip
3. Edit sonar-scanner configuration file located under config folder as follows -

```
#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 -



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

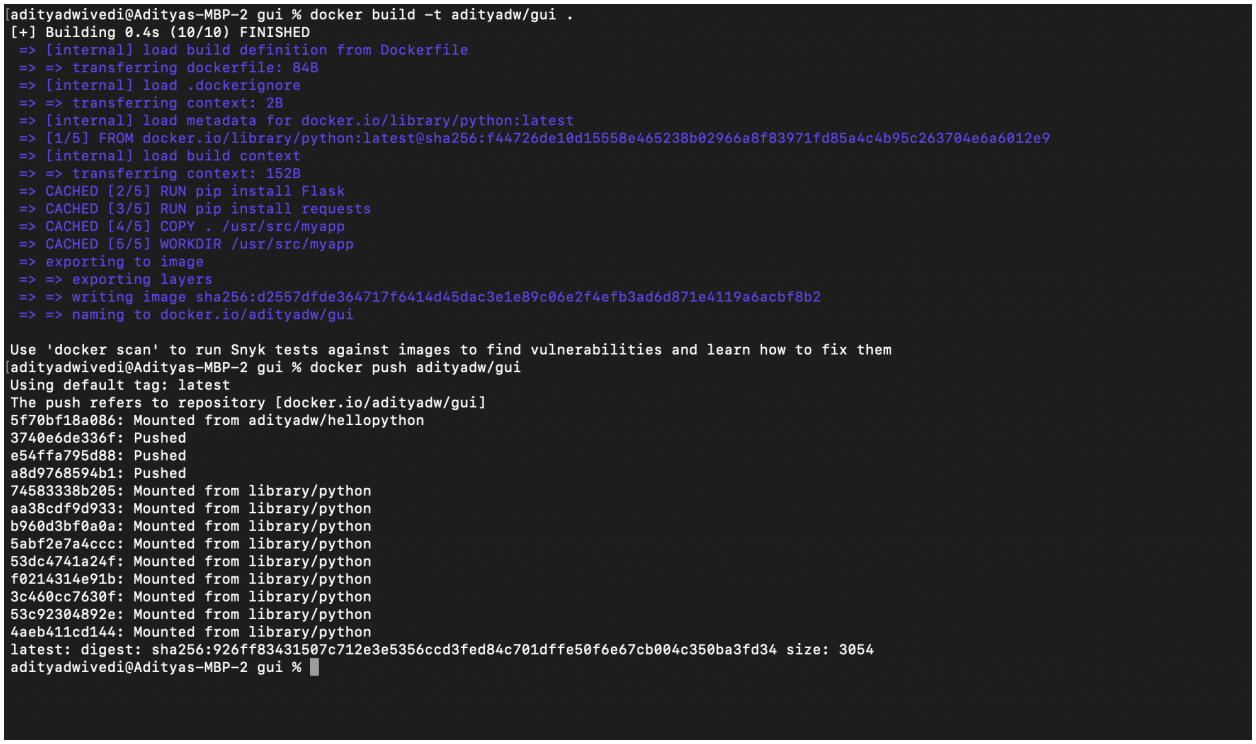
1. The following code sets up a flask app which returns HTML for the GUI, the URLs are passed as environment variables mentioned in the dockerfile and kubernetes deployment file. The requests library is used to check if the services are up and running

```
Users > adityadwivedi > Desktop > Aditya > Cloud-Infra > gui > gui.py
#reference - https://flask.palletsprojects.com/en/2.0.x/quickstart/
from flask import Flask
import sys,requests,time,os

app = Flask(__name__)
hadoop_url=''
spark_url=''
jupyter_url=''
sonarqube_url=''

#function to check if the micro-services are up and running by making the get request to their addresses
def checkServices():
    hadoop=requests.get(hadoop_url).status_code
    spark=requests.get(spark_url).status_code
    jupyter=requests.get(jupyter_url).status_code
    sonarqube=requests.get(sonarqube_url).status_code

    if hadoop==200 and spark==200 and jupyter==200 and sonarqube==200:
        return True
    else:
        return False


@app.route("/")
def terminal():
    #returning the webpage with hyperlinks
    return "<H1> Welcome to Big Data Processing Application </H1> <p> Please click on link to chose which application you would like to run </p>"\
        +"<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

#setting the urls using env variables
hadoop_url=os.environ['HADOOP_URL']
spark_url=os.environ['SPARK_URL']
jupyter_url=os.environ['JUPYTER_URL']
sonarqube_url=os.environ['SONAR_URL']

#check every 3 seconds if micro-services are up or not
while(checkServices()==False):
    print('Services not yet up')
    time.sleep(3)

app.run(host='0.0.0.0',port=8080)
```


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

```
FROM python:latest
RUN pip install Flask
RUN pip install requests
COPY . /usr/src/myapp
WORKDIR /usr/src/myapp
ENV HADOOP_URL http://34.136.215.85:9870/
ENV SPARK_URL http://34.135.56.132:8080/
ENV JUPYTER_URL http://34.133.108.43:8888/
ENV SONAR_URL http://35.222.228.36:9000/
EXPOSE 8080
CMD ["python3","gui.py"]
```

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



References

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://piazza.com/class/ksnwtglzkui2vr?cid=109