Week 10 Homework 1: Machine Learning on Kubernetes

Machine Learning on Kubernetes
Set up a functional Kubernetes cluster

1. Start minikube in GCP

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ minikube start
* minikube v1.34.0 on Ubuntu 24.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Updating the running docker "minikube" container ...
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
* Verifying Kubernetes components...
  - Using image registry.k8s.io/ingress-nginx/controller:v1.11.2
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.3
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.3
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Verifying ingress addon...
* Enabled addons: storage-provisioner, ingress, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
rpuranda464@cloudshell:~ (cs571-project-1-438018)$
```

2. Create a cluster with three nodes and check if the nodes are running.

- gcloud container clusters create kubia --num-nodes=1 --machine --region=us-west1
- gcloud container clusters list
- kubectl get nodes

```
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.30.5-gke.1443001
MASTER_IP: 34.168.58.89
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.30.5-gke.1443001
NUM_NODES: 3
STATUS: RUNNING
```

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ kubectl get nodes
NAME                                   STATUS   ROLES    AGE    VERSION
gke-kubia-default-pool-bf6cd5bd-c1sn   Ready    <none>   8m8s   v1.30.5-gke.1443001
gke-kubia-default-pool-d63119e9-bmr7   Ready    <none>   8m7s   v1.30.5-gke.1443001
gke-kubia-default-pool-def4c471-9nhm   Ready    <none>   8m8s   v1.30.5-gke.1443001
```

Week 10 Homework 1: Machine Learning on Kubernetes

Creating and uploading necessary files in GCP- Cloud Shell Terminal

3. Create requirements.txt file using the following command
   - nano requirements.txt

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ nano requirements.txt
```

Then enter the following contents in it :-

```
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy==1.19.5 # Adjusted to a version before np.float deprecation
scipy>=0.15.1
scikit-learn==0.24.2 # Ensure compatibility with numpy version
matplotlib>=1.4.3
pandas>=0.19
flasgger==0.9.4
```

4. Upload logreg.pkl file by clicking the three dots in the top-right part of the Cloud Shell Terminal and then choose upload.
   Then upload the logreg.pkl file which you can get it from the github link provided at the bottom of this document.

5. Create flask_api.py file using the below command and enter the following contents in it.
   - nano flask_api.py

```
from flask import Flask, request
import numpy as np
import pickle
import pandas as pd
from flasgger import Swagger

app = Flask(__name__)
Swagger(app)

# Load the model
pickle_in = open("logreg.pkl", "rb")
model = pickle.load(pickle_in)

@app.route('/')
def home():
    return "Welcome to the Flask API!"
```

```
@app.route('/predict', methods=["GET"])
def predict_class():
    """Predict if Customer would buy the product or not.
    ---
    parameters:
      - name: age
        in: query
        type: number
        required: true
      - name: new_user
        in: query
        type: number
        required: true
      - name: total_pages_visited
        in: query
        type: number
        required: true
    responses:
      200:
        description: Prediction
    """
    age = int(request.args.get("age"))
    new_user = int(request.args.get("new_user"))
    total_pages_visited = int(request.args.get("total_pages_visited"))
    prediction = model.predict([[age, new_user, total_pages_visited]])
    return "Model prediction is " + str(prediction)

@app.route('/predict_file', methods=["POST"])
def prediction_test_file():
    """Prediction on multiple input test file.
    ---
    parameters:
      - name: file
        in: formData
        type: file
        required: true
    responses:
      200:
        description: Test file Prediction
    """
    df_test = pd.read_csv(request.files.get("file"))
    prediction = model.predict(df_test)
    return str(list(prediction))
```

```
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

6. Create a Dockerfile by using the below command and then enter the following content in it.
   - nano dockerfile

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ nano requirements.txt
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ nano flask_api.py
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ nano Dockerfile
```

```
FROM python:3.8-slim
WORKDIR /app
COPY . /app
EXPOSE 5000
RUN pip install -r requirements.txt
CMD ["python", "flask_api.py"]
```

Running the Docker Container

7. To build the docker image use the below command
   - docker build -t ml_app_docker .

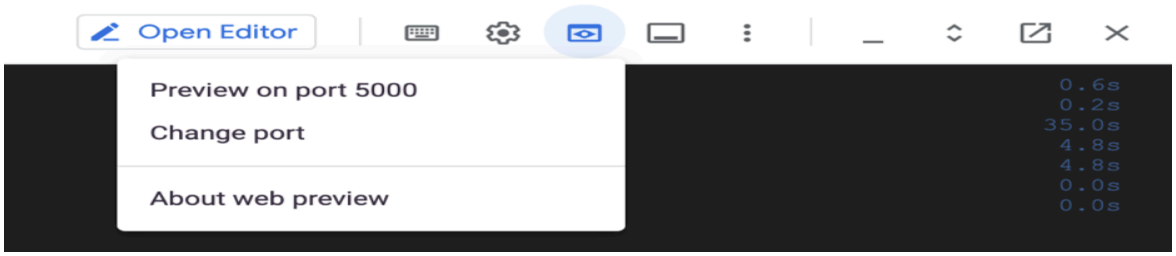
```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ sudo docker build -t ml_app_docker .
[+] Building 54.7s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 315B
 => [internal] load metadata for docker.io/library/python:3.8-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f
 => => resolve docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f
 => => sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f 10.41kB / 10.41kB
 => => sha256:314bc2fb0714b7807bf5699c98f0c73817e579799f2d91567ab7e9510f5601a5 1.75kB / 1.75kB
 => => sha256:b5f62925bd0f63f48cc8acd5e87d0c3a07e2f229cd2fb0a9586e68ed17f45ee3 5.25kB / 5.25kB
 => => sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab 29.13MB / 29.13MB
 => => sha256:030d7bdc20a63e3d22192b292d006a69fa3333949f536d62865d1bd0506685cc 3.51MB / 3.51MB
 => => sha256:a3f1dfe736c5f959143f23d75ab522a60be2da902efac236f4fb2a153cc14a5d 14.53MB / 14.53MB
 => => sha256:3971691a363796c39467aae4cdce6ef773273fe6bfc67154d01e1b589befb912 248B / 248B
 => => extracting sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab
 => => extracting sha256:030d7bdc20a63e3d22192b292d006a69fa3333949f536d62865d1bd0506685cc
 => => extracting sha256:a3f1dfe736c5f959143f23d75ab522a60be2da902efac236f4fb2a153cc14a5d
 => => extracting sha256:3971691a363796c39467aae4cdce6ef773273fe6bfc67154d01e1b589befb912
 => [internal] load build context
 => => transferring context: 199.28MB
 => [2/4] WORKDIR /app
 => [3/4] COPY . /app
 => [4/4] RUN pip install -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:55c40c019e8366549812e2344eed7cf3639cb21721c30a2d73d0375cc7a1dc6e
 => => naming to docker.io/library/ml_app_docker
```

8. Below command runs a Docker container from the ml_app_docker image:
   - docker container run -p 5000:5000 ml_app_docker

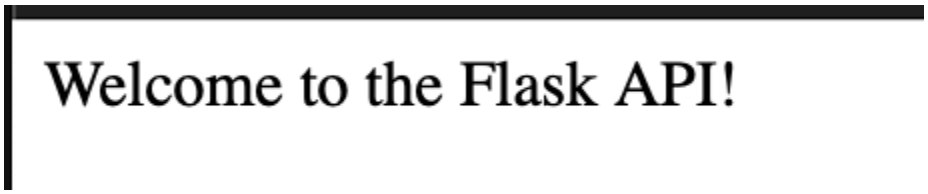
Week 10 Homework 1: Machine Learning on Kubernetes

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ docker container run -p 5000:5000 ml_app_docker
 * Serving Flask app "flask_api" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
/usr/local/lib/python3.8/site-packages/sklearn/base.py:310: UserWarning: Trying to unpickle estimator Lo
ing code or invalid results. Use at your own risk.
  warnings.warn(
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
/usr/local/lib/python3.8/site-packages/sklearn/base.py:310: UserWarning: Trying to unpickle estimator Lo
ing code or invalid results. Use at your own risk.
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 339-369-829
```

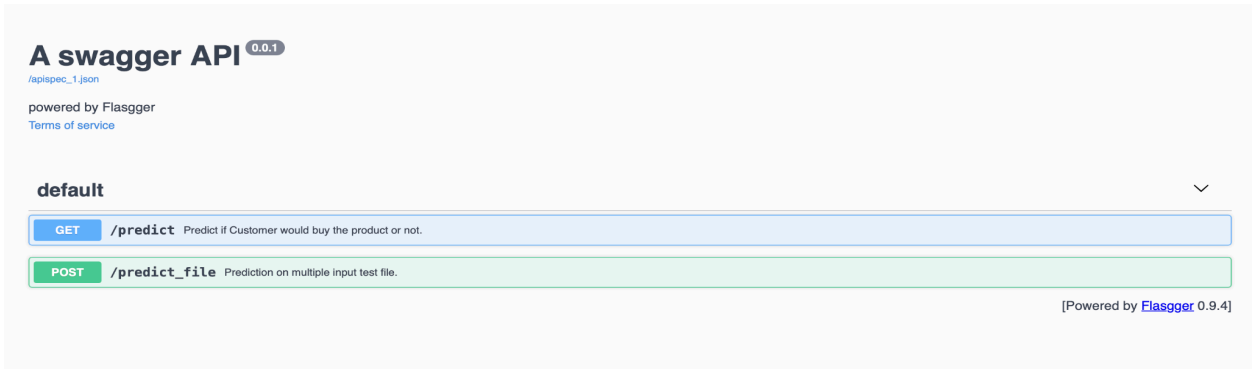
9. In the right-upper side of the terminal click the eye shaped button and then click Preview on port 5000. Change port if it is not 5000 by default.



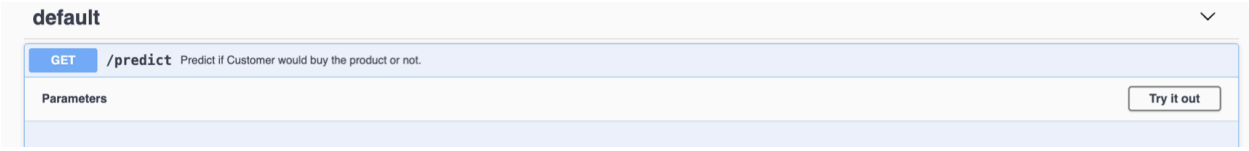
10. You will see this using the web preview.



11. Add /apidocs/ at the end of the link to access the running ml- app as following
    - There are two tabs GET and POST.

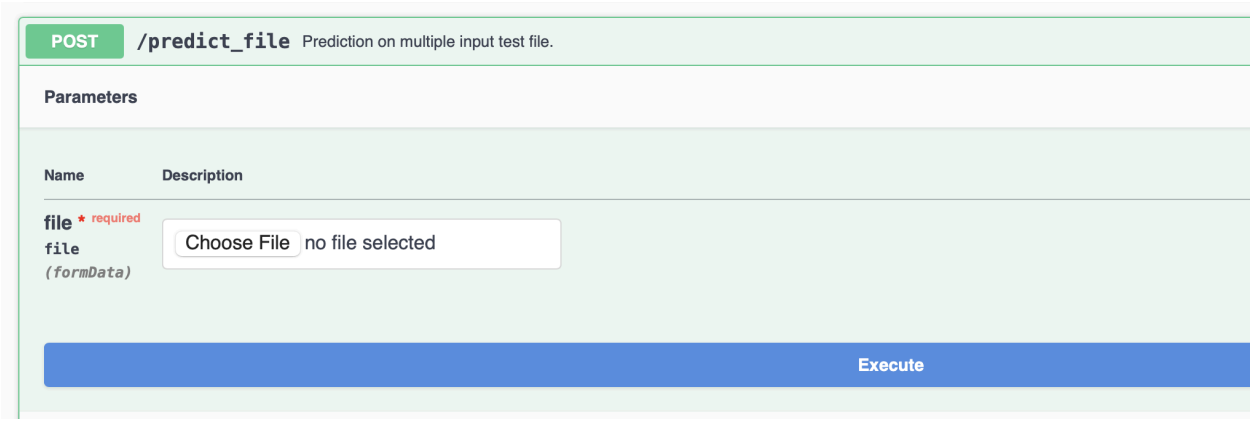


12. Click GET and then click Try it out in the top-right corner of the GET box.

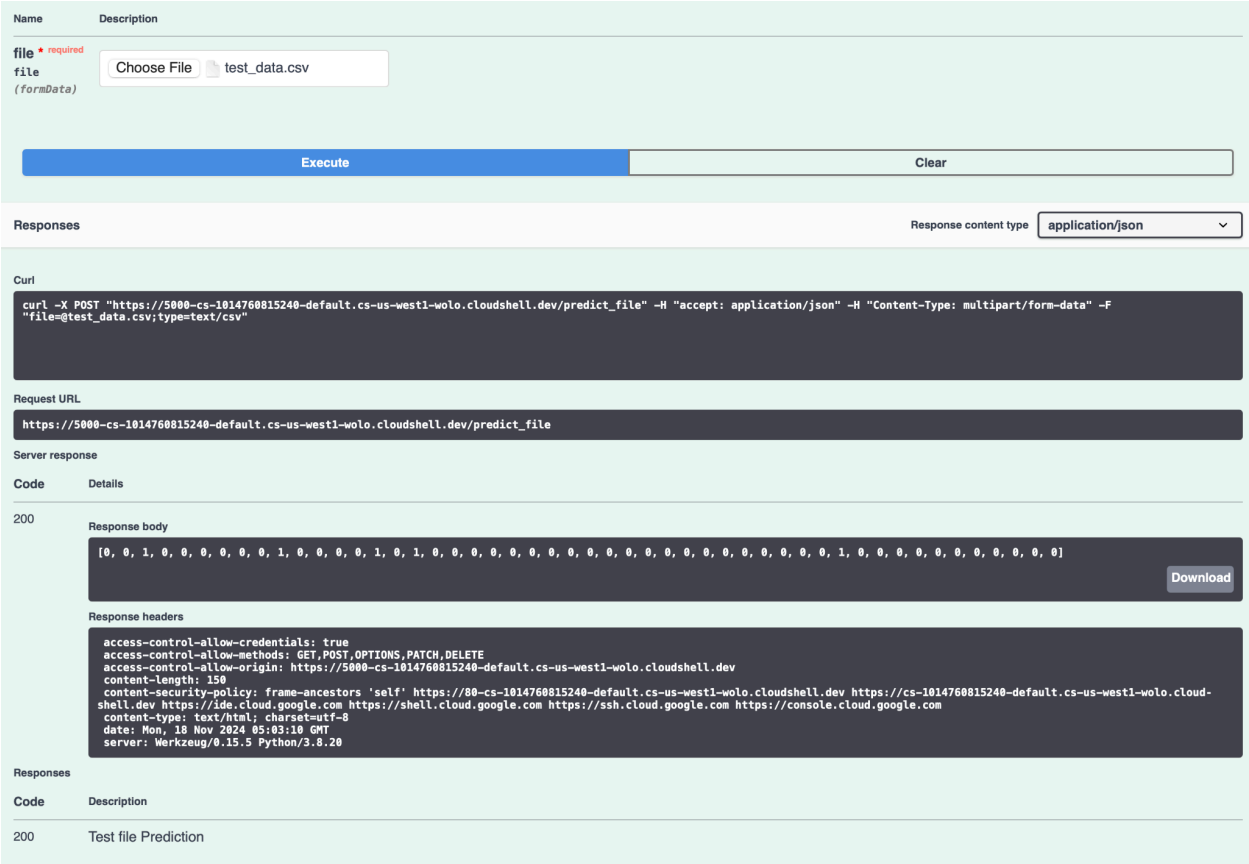


13. Fill values for the input parameters and then click Execute , you will get below response. Upon the execution call, the request goes to the app, and predictions are made by the model. The result of the model prediction is displayed in the Prediction section of the page.

14. The next prediction that can be done is for a group of customers (test data) via a post request.



15. Upload the test data file containing the same parameters in a similar order. The model would make the prediction, and the results would be displayed upon execution as follows. You can download the test_data.csv from the github link at the end of the document and then upload.



Push the image to your docker hub

16. Login your docker hub by using below command and then Tag the image and then push the image to docker hub
    - docker login
    - docker tag ml_app_docker yourdockerhubusername/yourrepositoryname
    - docker push yourdockerhubusername/yourrepositoryname

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ docker push rash0101/ml_app_docker
Using default tag: latest
The push refers to repository [docker.io/rash0101/ml_app_docker]
558d290d94f4: Pushed
7918e383fec4: Pushed
e81fabe4d4f3: Pushed
d2a2207b52a4: Mounted from library/python
5d2d143f3d7f: Mounted from library/python
c3772b569c3a: Mounted from library/python
8d853c8add5d: Mounted from library/python
latest: digest: sha256:5a2c416ba70c99f6db4c222c1def317ed616723813ead41a97292ea33af3e44d size: 1791
```

Week 10 Homework 1: Machine Learning on Kubernetes

Deploy your ML app to GKE

17. Use the GKE we have created in Step 1.
    - Create a deployment.yaml with the following contents.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ml-app-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ml-app
  template:
    metadata:
      labels:
        app: ml-app
    spec:
      containers:
      - name: ml-app-container
        image: rash0101/w10hw1
        ports:
        - containerPort: 5000
```

Then apply the created deployment file.
- kubectl apply -f deployment.yaml

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ kubectl apply -f deployment.yaml
deployment.apps/ml-app-deployment created
rpuranda464@cloudshell:~ (cs571-project-1-438018)$
```

18. Wait for couple minutes and list all the pods created
    - kubectl get pods

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
ml-app-deployment-5c8474bc65-xs9fc   1/1     Running   0          50s
rpuranda464@cloudshell:~ (cs571-project-1-438018)$
```

19. Create a service.yaml

```
apiVersion: v1
```

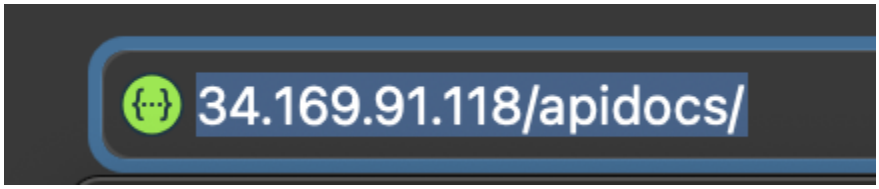
```
kind: Service
metadata:
  name: ml-app-service
spec:
  selector:
    app: ml-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
  type: LoadBalancer
```

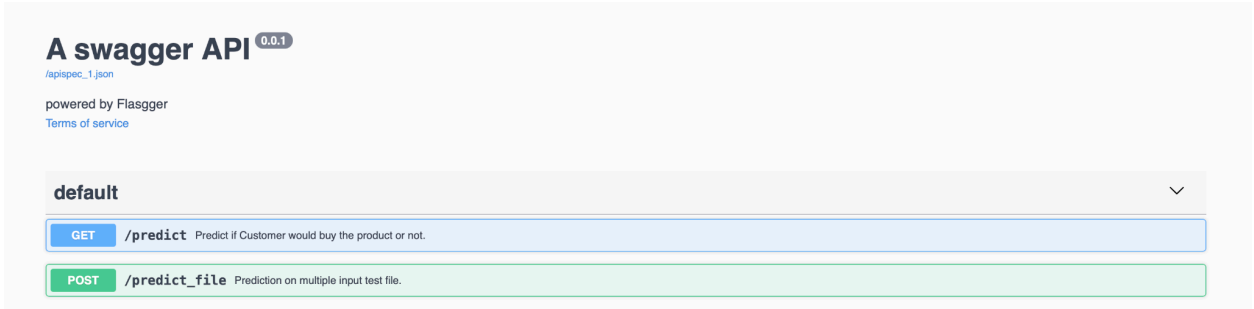
```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ nano service.yaml
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ kubectl apply -f service.yaml
service/ml-app-service created
rpuranda464@cloudshell:~ (cs571-project-1-438018)$
```

20. Get service external ip
    - kubectl get svc
    -

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ kubectl get svc
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
kubernetes       ClusterIP      34.118.224.1    <none>          443/TCP        5h58m
ml-app-service   LoadBalancer   34.118.234.49   34.169.91.118   80:32190/TCP   50s
```

21. Access using browser:
    - external-ip/apidocs





Week 10 Homework 1: Machine Learning on Kubernetes

Stopping/Killing the running container

22. Use docker ps to list running Docker containers

```
^Crpuranda464@cloudshell:~ (cs571-project-1-438018)$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS
                     NAMES
acf836fdbab4   gcr.io/k8s-minikube/kicbase:v0.0.45   "/usr/local/bin/entr…"   29 minutes ago   Up 29 minutes   127.0.0.1:32768->22/tcp,
```

- The CONTAINER_ID is given as acf836fdbab4

23. Use the command
    - docker kill <CONTAINER ID> to kill the running container as follows.

```
rpuranda464@cloudshell:~ (cs571-project-1-438018)$ docker kill acf836fdbab4
acf836fdbab4
rpuranda464@cloudshell:~ (cs571-project-1-438018)$
```