

CS501- Week 10 Homework 1: Machine Learning on Kubernetes

Name: Sirak Hadgu

ID: 20191

https://hc.labnet.sfbu.edu/~henry/sfbu/course/cloud_computing/genai/slide/exercise_kubernetes.html

Q2 ==> Machine Learning on Kubernetes Machine Learning on Kubernetes

Creating and uploading necessary files in GCP- Cloud Shell Terminal

```
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$ kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
gke-kubia-default-pool-021f4515-cpk8  Ready    <none>    2m14s  v1.30.5-gke.1443001
gke-kubia-default-pool-3fc93e1d-3j53  Ready    <none>    2m14s  v1.30.5-gke.1443001
gke-kubia-default-pool-adb8682a-8jdc  Ready    <none>    2m15s  v1.30.5-gke.1443001
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$
```

1. Start minikube in Google Cloud Platform

```
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$ minikube start
* minikube v1.34.0 on Ubuntu 24.04 (amd64)
- MINIKUBE_FORCE_SYSTEMD=true
- MINIKUBE_HOME=/google/minikube
- MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Downloading Kubernetes v1.31.0 preload ...
  > preloaded-images-k8s-v18-v1...: 326.69 MiB / 326.69 MiB 100.00% 203.59
  > gcr.io/k8s-minikube/kicbase...: 487.90 MiB / 487.90 MiB 100.00% 80.11 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
- kubelet.cgroups-per-qos=false
- kubelet.enforce-node-allocatable=""
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
- Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$
```

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

```
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$ nano requirements.txt
```

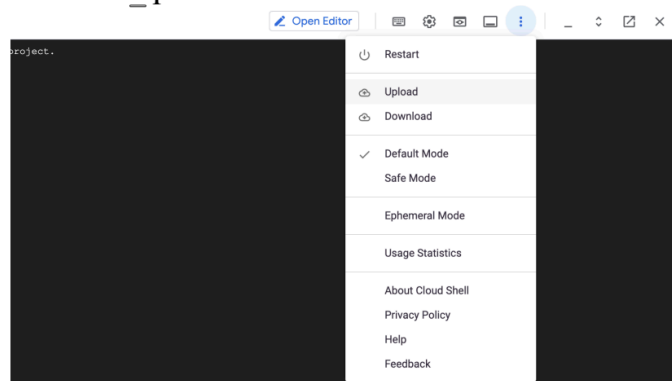
Then enter the following contents

```
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
```

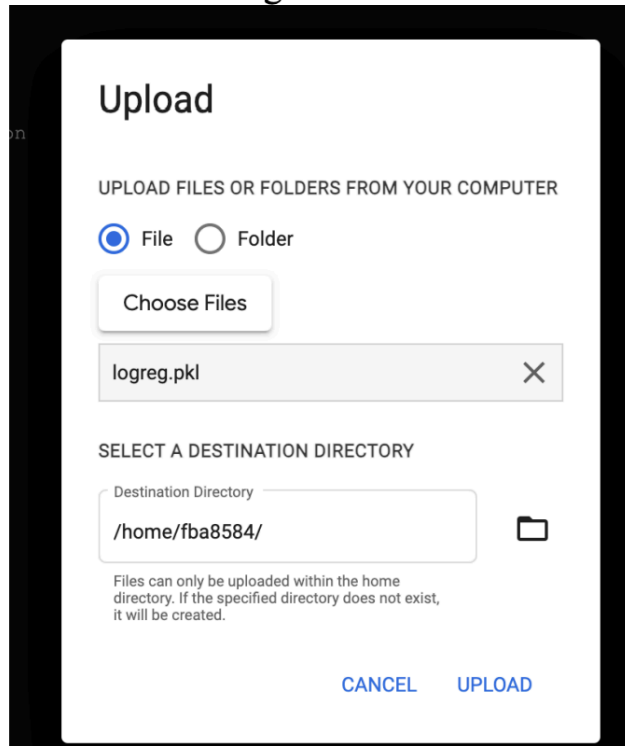


The screenshot shows a Cloud Shell terminal window with the title "CLOUD SHELL" and "Terminal". The terminal content is the same list of package versions as shown in the previous block, entered into a GNU nano 5.4 editor. The packages listed are Flask, gunicorn, itsdangerous, Jinja2, MarkupSafe, Werkzeug, numpy, scipy, scikit-learn, matplotlib, pandas, and flasgger, each followed by its version number and some explanatory comments for numpy, scipy, and scikit-learn.

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 as following



4. Create flask_api.py file using the command
- *nano flask_api.py*

```
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$ nano flask_api.py
```

GNU nano 5.4

```
# -*- coding: utf-8 -*-
"""
Created on Mon May 25 12:50:04 2020

@author: pramod.singh
"""

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

app = Flask(__name__)
Swagger(app)

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
    """
```

```
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
```

```

@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))

@app.route('/apidocs')
def api_docs():
    return "API Documentation goes here."

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

```

Step 4: Dockerfile 1. Create Dockerfile using command - nano Dockerfile

```
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$ docker build -t ml_app_docker
```

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

```

GNU nano 5.4
FROM python:3.8-slim
WORKDIR /app
COPY . /app
EXPOSE 5000
RUN pip install -r requirements.txt
CMD ["python", "flask_api.py"]

```

1. 'FROM python:3.8-slim'

- This line sets the base image for the Docker image you are creating. It tells Docker to start with the 'python:3.8-slim' image, which is an official Python image with Python 3.8 installed on it. The 'slim' version is a smaller version of the image that has fewer packages pre-installed, making the image size smaller.

2. 'WORKDIR /app' • This instruction sets the working directory within the Docker container to */app*. All subsequent commands will be executed in this directory within the container.

3. 'COPY . /app' • This line copies everything from the current directory (on the host machine where you're running the Docker build command, indicated by the first**) into the */app directory inside the Docker image (the second ' /app*').
4. 'EXPOSE 5000' • The 'EXPOSE' instruction informs Docker that the container listens on the specified network port at runtime. In this case, it tells Docker that the container will listen on port 5000. It's worth noting that this does not actually publish the port—it serves as documentation and is used by the 'docker run -p' command to map the container port to a port on the Docker host.
5. 'RUN pip install -r requirements.txt' • This command tells Docker to run pip install' inside the container, which will install the Python dependencies listed in the requirements.txt file. These dependencies are necessary for the Flask application to run correctly.
6. CMD ["python", "flask_api.py"] • This is the command that will be executed by default when the Docker container starts. In this case, it's telling Docker to run 'flask_api.py using Python. This is the Flask application you want to run inside the container.

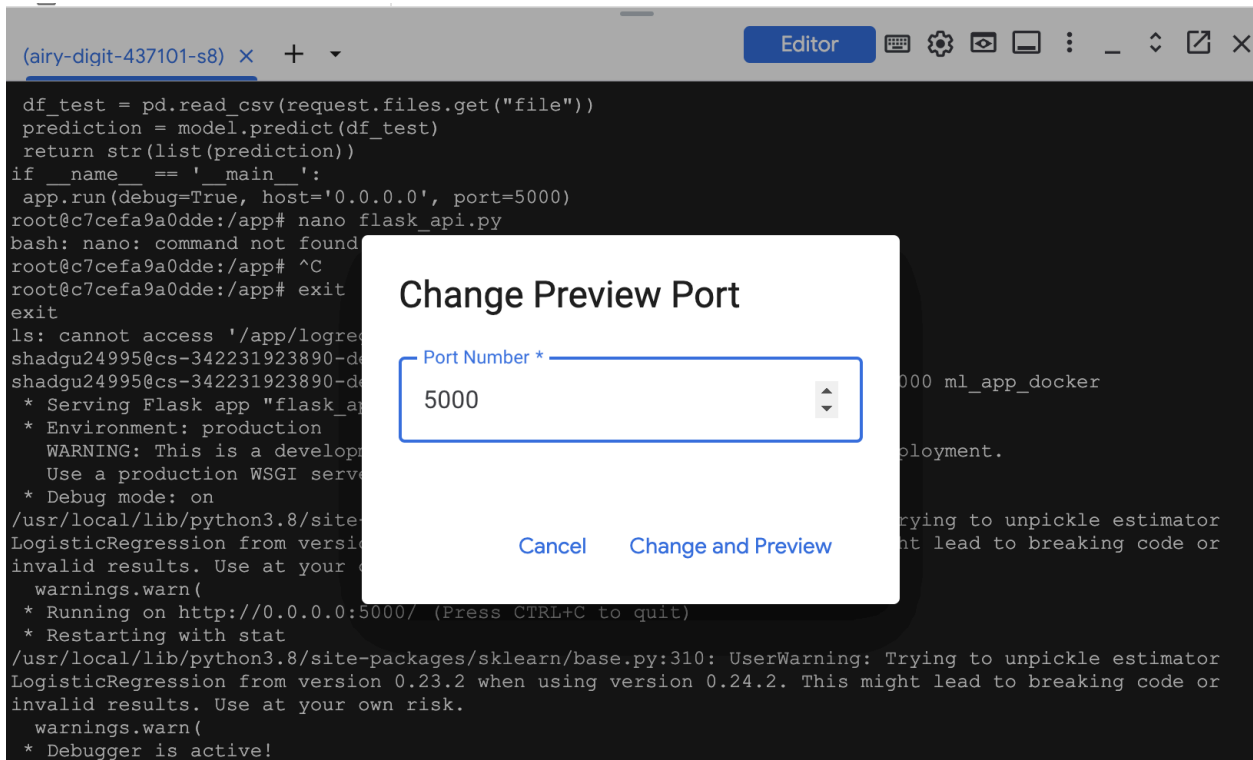
Step 5: Running the Docker Container 1. To build the docker image use the command - sudo docker build -t ml_app_docker .

```
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$ docker build -t ml_app_docker .
[+] Building 15.3s (9/9) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 169B                               0.0s
=> [internal] load metadata for docker.io/library/python:3.8-slim 1.0s
=> [auth] library/python:pull token for registry-1.docker.io      0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [1/4] FROM docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d 5.9s
=> => resolve docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d 0.0s
=> => sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382 10.41kB / 10.41kB 0.0s
=> => sha256:314bc2fb0714b7807bf5699c98f0c73817e579799f2d91567ab7e951 1.75kB / 1.75kB 0.0s
=> => sha256:b5f62925bd0f63f48cc8acd5e87d0c3a07e2f229cd2fb0a9586e68ed 5.25kB / 5.25kB 0.0s
=> => sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce 29.13MB / 29.13MB 0.7s
=> => sha256:030d7bdc20a63e3d22192b292d006a69fa3333949f536d62865d1bd0 3.51MB / 3.51MB 0.2s
=> => sha256:a3f1dfe736c5f959143f23d75ab522a60be2da902efac236f4fb2a 14.53MB / 14.53MB 0.4s
=> => sha256:3971691a363796c39467aae4cdce6ef773273fe6bfc67154d01e1b589bef 248B / 248B 0.3s
=> => extracting sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047f 3.2s
=> => extracting sha256:030d7bdc20a63e3d22192b292d006a69fa3333949f536d62865d1bd050668 0.3s
=> => extracting sha256:a3f1dfe736c5f959143f23d75ab522a60be2da902efac236f4fb2a153cc14 1.2s
=> => extracting sha256:3971691a363796c39467aae4cdce6ef773273fe6bfc67154d01e1b589befb 0.0s
=> [internal] load build context                                4.7s
=> => transferring context: 360.39MB                             4.7s
```

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

```
shadgu24995@cs-342231923890-default:~/Python-3.10.12$ docker 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 LogisticRegression from version 0.23.2 when using version 0.24.2. This might lead to breaking 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 LogisticRegression from version 0.23.2 when using version 0.24.2. This might lead to breaking code or invalid results. Use at your own risk.
warnings.warn(
* Debugger is active!
* Debugger PIN: 593-848-457
```

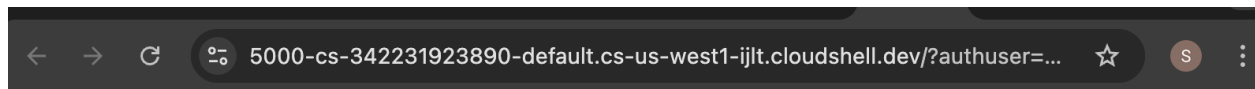
3. In the right-upper side of the terminal click the eye shaped button and then click Preview on port 5000. Change port if it is not 5000 by default.



The screenshot shows a terminal window with a dark background. The terminal text includes Python code for a Flask API, followed by shell commands like `nano flask_api.py` and `app.run(debug=True, host='0.0.0.0', port=5000)`. A modal dialog box titled "Change Preview Port" is centered over the terminal. It has a text input field labeled "Port Number *" containing the value "5000". At the bottom of the dialog are two buttons: "Cancel" and "Change and Preview". The terminal text in the background shows the application starting on port 5000 and includes a warning about unpickling an estimator.

```
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)
root@c7cefa9a0dde:/app# nano flask_api.py
bash: nano: command not found
root@c7cefa9a0dde:/app# ^C
root@c7cefa9a0dde:/app# exit
exit
ls: cannot access '/app/logre': No such file or directory
shadgu24995@cs-342231923890-d...
shadgu24995@cs-342231923890-d...
* Serving Flask app "flask_api"
* Environment: production
WARNING: This is a development server.
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
LogisticRegression from version 0.23.2 when using version 0.24.2. This might lead to breaking 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
LogisticRegression from version 0.23.2 when using version 0.24.2. This might lead to breaking code or
invalid results. Use at your own risk.
warnings.warn(
* Debugger is active!
```

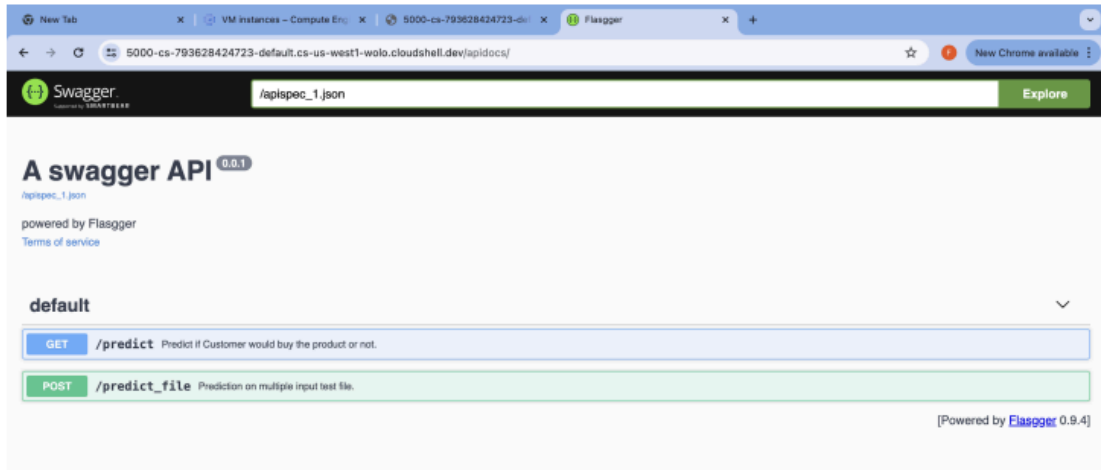
4. You will see this using the web preview.



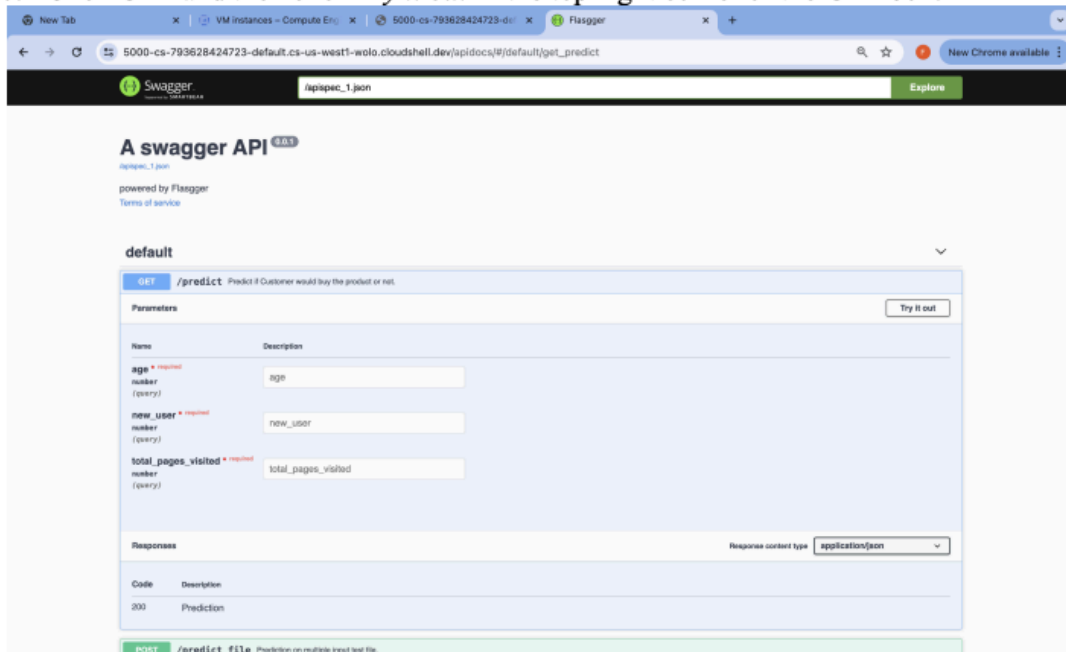
4. You will see this using the web preview.



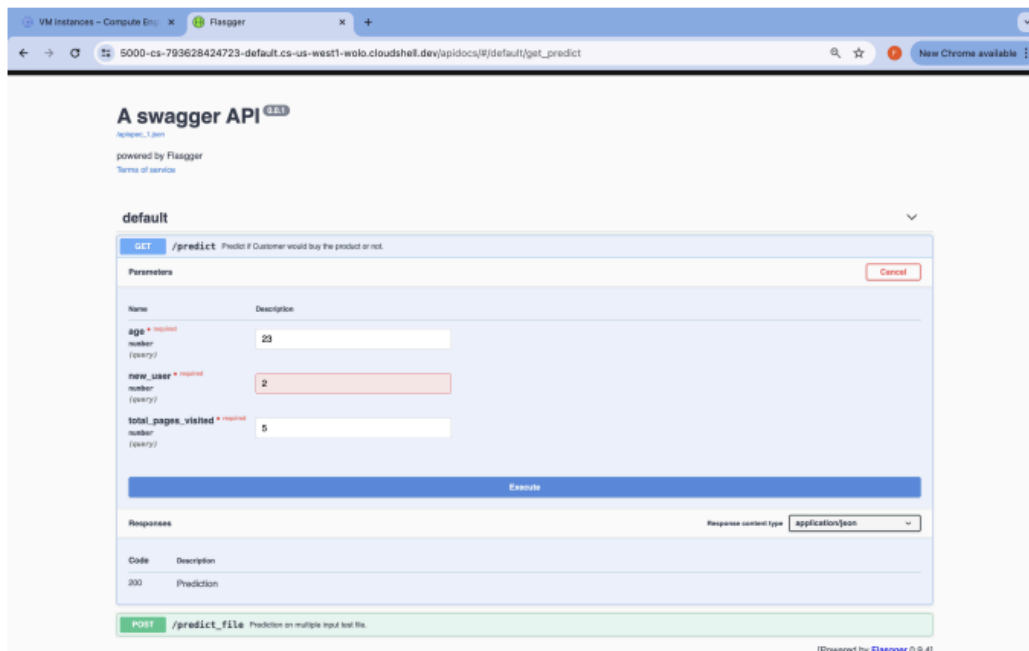
5. Add `/apidocs/` at the end of the link to access the running ml- app as following
- There are two tabs GET and POST.



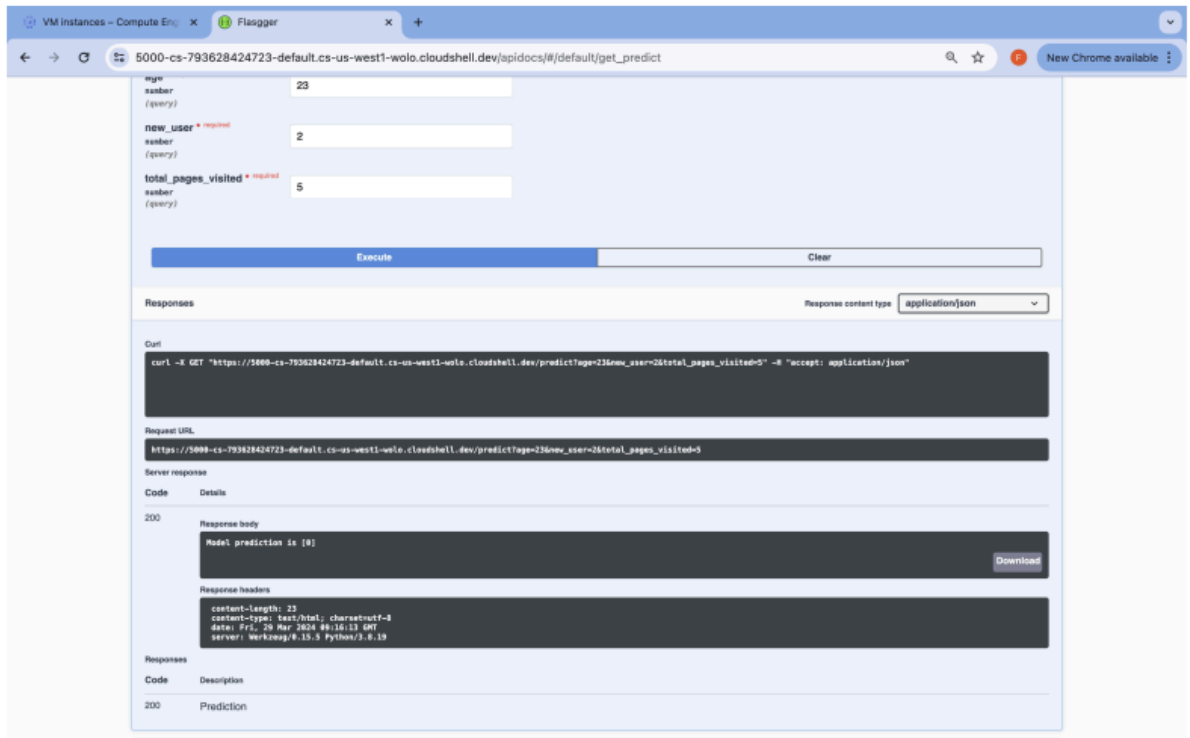
6. Click **GET** and then click *Try it out* in the top-right corner of the GET box.



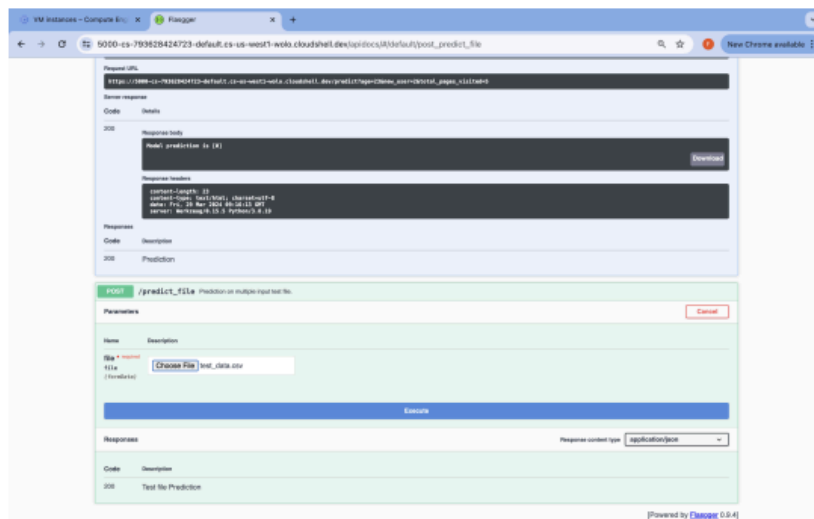
7. Fill values for the input parameters and then click **Execute**.



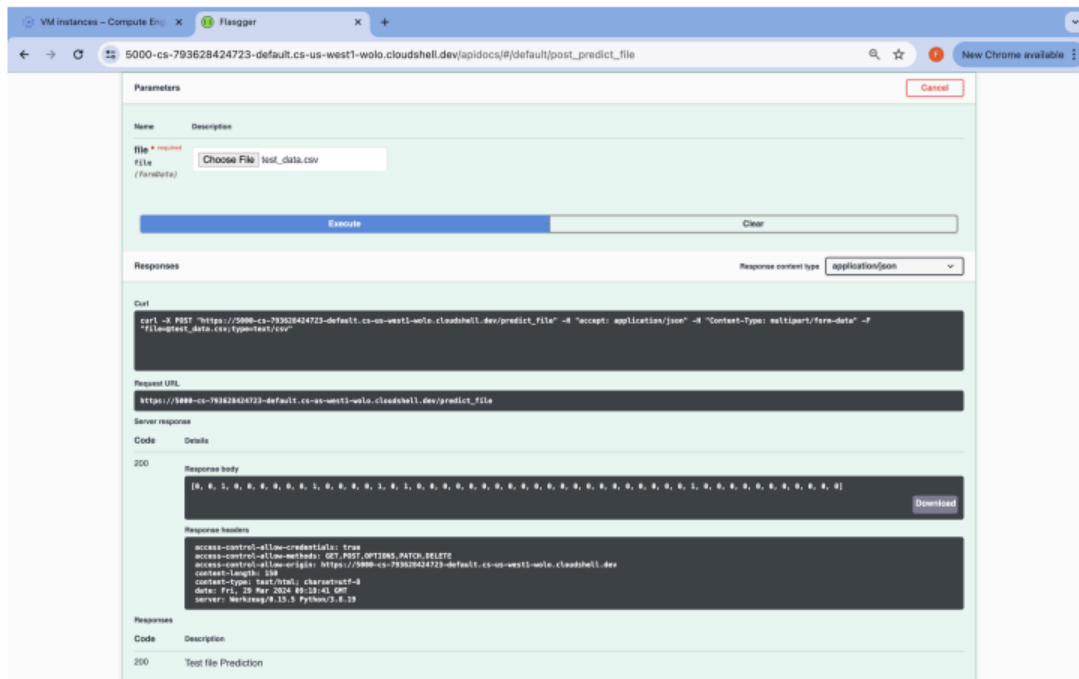
8. Upon the execution call, the request goes to the app, and predictions are made by the model.
 - The result of the model prediction is displayed in the Prediction section of the page as following



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



10. 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 execute as following.



Step 6: Stopping/killing the running container 1. Use `docker ps` to list running Docker containers

```
shadgu24995@cloudshell:~ (airy-digit-437101-s8)$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------