

# Flight Delay Prediction App

## Load Testing

---

### Objective:

The goal is to stress test the service. We will create a model and api service for the model to send request and get the prediction based on data.

In the following document you will find various methods of deploying the application. At first we will deploy this application on local system to test the functionality and will do stress test.

With time, we can improve the infrastructure of the application (for eg. Running in kubernetes cluster or cloudrun service from google.)

---

### Bookmarks :

1. Expose the serialized model via REST API.
  - You can use the already serialized model (pickle\_model.pkl) or rebuild it.
2. Automate the process of building and deploying the API, using one or more cloud services.
3. Stress test the API with the exposed model with at least 50,000 requests for 45 seconds.  
For this, you must use wrk tool and present the metrics obtained.
  - How could you improve the performance of the previous tests?
4. The infrastructure creation process must be done with Terraform.
5. What would be the ideal mechanisms so that only authorized systems can access this API?
  - Does this mechanism add any level of latency to the consumer? Because?
6. What would be the SLIs and SLOs that you would define and why?
7. Monitoring.

## App running on local system-

---

### Performance:

Here we can assume that application will get affected if the system goes down and to scale up the infrastructure, we need manual intervention.

### Repository:

Git

### Steps Overview:

- :Pikling file is created and provided in drive.
- :Create api and web service that will run on local system directly.
- :Api request will be done using browser.

-Let's create a model.

### Pre-requisite-

Create python virtual environment and install dependencies.

`python3 -m venv sre-challenge`

`pip install flask`

`pip install flask-restful`

`pip install bumpy`

`pip install pandas`

`pip install scikit-learn`

Will download a new dataset which consist flight related details.

Download flight data from <https://raw.githubusercontent.com/plotly/datasets/master/flightdata.csv>

### Goal:

In a model.py file, we will develop and train our model.

Load the dataset.

Clean the dataset

Create model.pkl file based on the dataset.

### Clean and prepare data:

- Figure out “feature ” Columns that are relevant to the output we are predicting.
- Eliminate Missing values either by deleting rows or columns or adding meaningful values

NOTE: Check for null value in the data set by creating new notebook-  
(cleanandpreparedataset.ipynb notebook for test and validate results, added to repo)

-Added file model.py.

-Model.pkl file generated and stored in root directory

The screenshot shows a Jupyter Notebook environment with the following details:

- EXPLORER:** Shows the project structure under "LATAM-SRE" and "SRE-challenge". Files listed include: \_\_pycache\_\_, .ipynb\_checkpoints, bin, datasets (flightdata.csv), etc, include, lib, share, .gitignore, api.py, cleanandpreparedataset.ipynb, model.ipynb, model.pkl, model.py, pickle\_model.pkl, pyvenv.cfg, readPkl.py, requirements.txt, webapp.py, README.md.
- CELL OUTPUT:** Displays the output of a cell that fills a DataFrame with zeros. The code was: 

```
df = df.fillna({"dep_delay": 1})
```

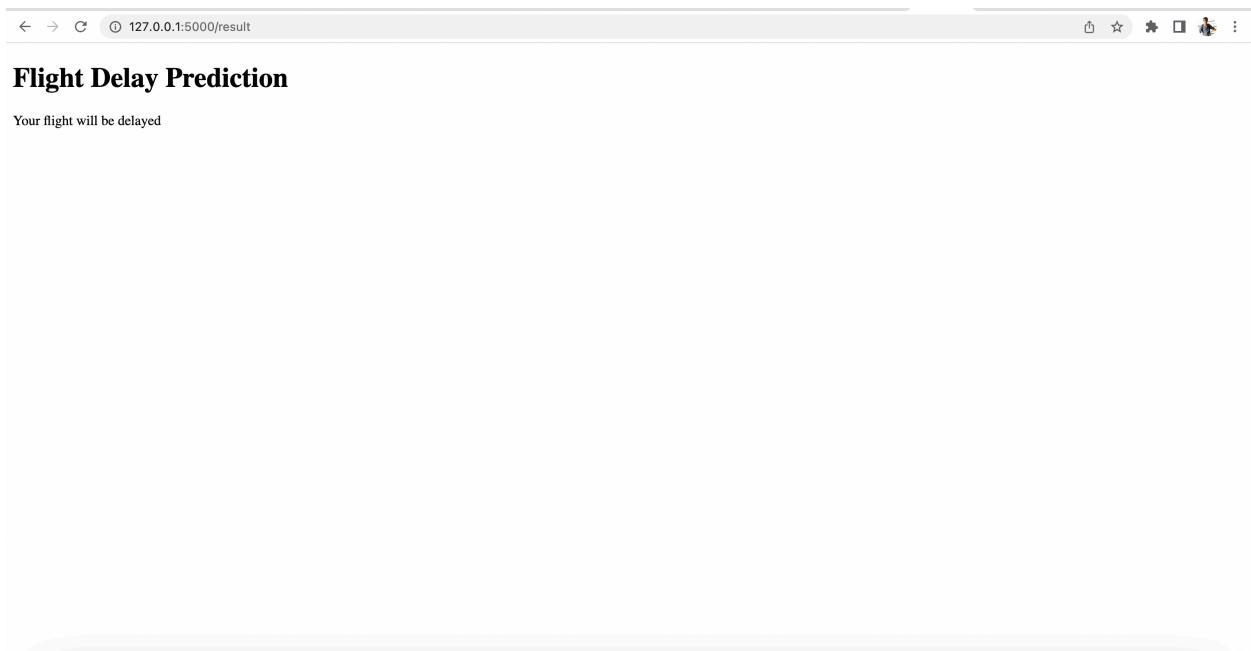
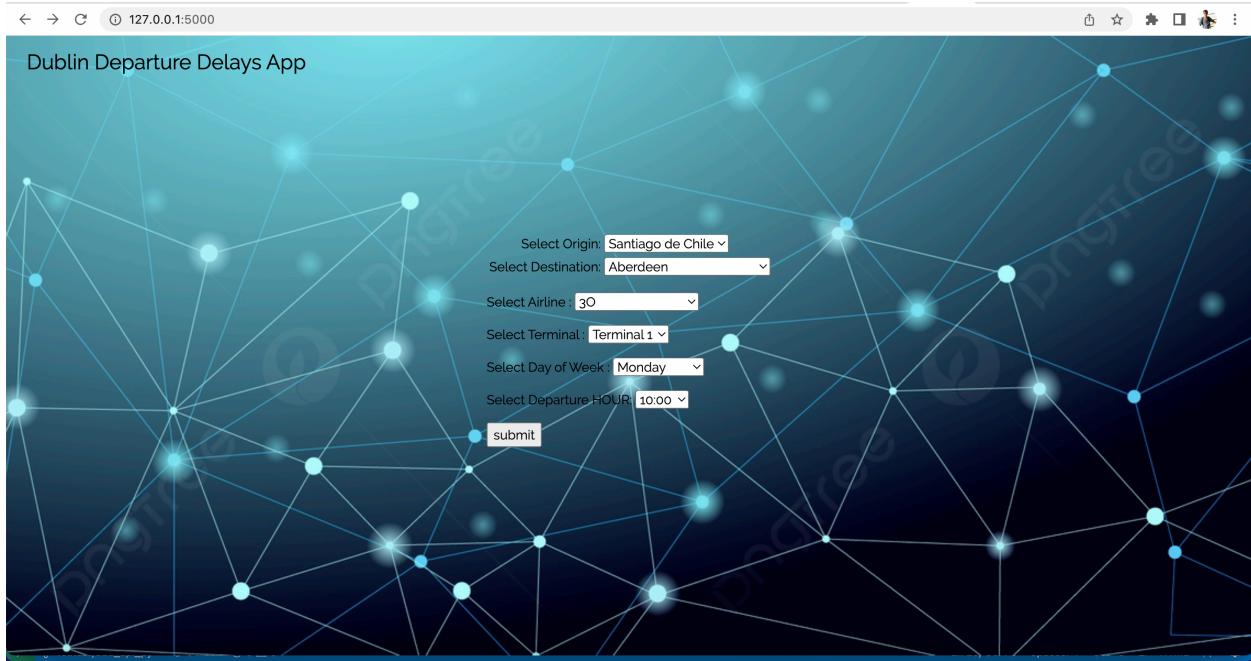
 and the output is:

year	month	day	sched_dep_time	sched_arr_time	carrier	origin	dest	dep_delay	arr_delay	dtype
0	0	0	0	0	0	0	0	0	0	int64
- PROBLEMS:** Shows a warning: "STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT." with a link to scikit-learn documentation.
- TERMINAL:** Shows the command run: 

```
(sre-challenge) tanmay@tanmays-MacBook-Pro SRE-challenge % python readPkl.py
```
- CODE EDITOR:** Displays the Python code for reading the dataset and printing its summary statistics.
- OUTPUT:** Shows the full output of the code, which includes the entire dataset summary (2196 rows, 19 columns) and a note about output size limit.

-Uploaded changes to git [https://github.com/tjtanmay/latam-sre/tree/feature/add\\_api\\_py](https://github.com/tjtanmay/latam-sre/tree/feature/add_api_py)

-Execute python api.py to launch.



---

## Load Testing

-Install wrk benchmark tool

```
[tanmay@Tanmays-MacBook-Pro ~ % sudo port install wrk
--> Computing dependencies for wrk
The following dependencies will be installed:
luajit
openssl13
zlib
Continue? [Y/n]: Y
--> Fetching archive for luajit
--> Attempting to fetch luajit-2.1.0-beta3_6.darwin_21.arm64.tbz2 from https://packages.macports.org/luajit
--> Attempting to fetch luajit-2.1.0-beta3_6.darwin_21.arm64.tbz2.rmd160 from https://packages.macports.org/luajit
--> Installing luajit @2.1.0-beta3_6
--> Activating luajit @2.1.0-beta3_6
--> Cleaning luajit
--> Fetching archive for zlib
--> Attempting to fetch zlib-1.2.13_0.darwin_21.arm64.tbz2 from https://packages.macports.org/zlib
--> Attempting to fetch zlib-1.2.13_0.darwin_21.arm64.tbz2.rmd160 from https://packages.macports.org/zlib
--> Installing zlib @1.2.13_0
--> Activating zlib @1.2.13_0
--> Cleaning zlib
--> Fetching archive for openssl13
--> Attempting to fetch openssl13-3.0.8_1+legacy.darwin_21.arm64.tbz2 from https://packages.macports.org/openssl13
--> Attempting to fetch openssl13-3.0.8_1+legacy.darwin_21.arm64.tbz2 from http://mirror.fcix.net/macports/packages/openssl13
--> Attempting to fetch openssl13-3.0.8_1+legacy.darwin_21.arm64.tbz2 from https://ywg.ca.packages.macports.org/mirror/macports/packages/openssl13
--> Fetching distfiles for openssl13
--> Attempting to fetch openssl13-3.0.8.tar.gz from https://www.openssl.org/source
--> Verifying checksums for openssl13
--> Extracting openssl13
--> Configuring openssl13
--> Building openssl13
--> Staging openssl13 into destroot
--> Installing openssl13 @3.0.8_1+legacy
--> Activating openssl13 @3.0.8_1+legacy
--> Cleaning openssl13
--> Fetching archive for wrk
--> Attempting to fetch wrk-4.2.0_0.darwin_21.arm64.tbz2 from https://packages.macports.org/wrk
--> Attempting to fetch wrk-4.2.0_0.darwin_21.arm64.tbz2.rmd160 from https://packages.macports.org/wrk
--> Installing wrk @4.2.0_0
--> Activating wrk @4.2.0_0
--> Cleaning wrk
--> Updating database of binaries
--> Scanning binaries for linking errors
--> No broken files found.
--> No broken ports found.
tanmay@Tanmays-MacBook-Pro ~ % ]
```

Result:

After running load test for 30 seconds, using 12 threads, and keeping 400 HTTP connections open

```
tanmay@Tanmays-MacBook-Pro ~ % wrk -t12 -c400 -d30s http://
127.0.0.1:5000/
Running 30s test @ http://127.0.0.1:5000/
 12 threads and 400 connections
 Thread Stats      Avg      Stdev     Max    +/- Stdev
  Latency       46.12ms   36.43ms  280.88ms   82.78%
  Req/Sec      158.69     100.23   595.00    66.61%
 10673 requests in 30.03s, 219.88MB read
  Socket errors: connect 0, read 10978, write 369, timeout 0
 Requests/sec:      355.43
 Transfer/sec:      7.32MB
```

```
tanmay@Tanmays-MacBook-Pro ~ % wrk -t12 -c50000 -d45s http://
127.0.0.1:5000/
Running 45s test @ http://127.0.0.1:5000/
 12 threads and 50000 connections
 Thread Stats      Avg      Stdev     Max    +/- Stdev
```

```

Latency      104.15ms   162.43ms    1.61s      98.06%
Req/Sec       25.00      46.66     424.00      92.23%
4117 requests in 45.10s, 84.82MB read
Socket errors: connect 47447, read 652563, write 3326, timeout 0
Requests/sec:      91.29
Transfer/sec:      1.88MB

```

- Added post.lua script with body to request from API and triggered with script.

```

[tanmay@Tanmays-MacBook-Pro SRE-challenge % wrk -t12 -c50000 -d45s -s ./scripts/post.lua http://127.0.0.1:5000/
./scripts/post.lua: ./scripts/post.lua:2: unexpected symbol near '10'
Running 45s test @ http://127.0.0.1:5000/
 12 threads and 50000 connections
 Thread Stats      Avg      Stdev      Max      +/- Stdev
  Latency      41.88ms   35.65ms  306.64ms    74.34%
  Req/Sec       115.76    112.21   820.00      77.90%
 19001 requests in 45.10s, 391.63MB read
  Socket errors: connect 47447, read 18422, write 450, timeout 0
Requests/sec:      421.30
Transfer/sec:      8.68MB
tanmay@Tanmays-MacBook-Pro SRE-challenge %

```

## Conclusion:

Here we can see the performance of the service depends upon availability of resource of the local system. We have to manually bring up the service every time if service goes down. This kind of infrastructure is not good for production use rather, developer can follow the above steps to test their application locally quickly without depending upon infra.

# App running on Docker-

## Performance:

As the application will run on Docker, maintaining container will be easy. We can build the application image which makes the application portable and we don't need to spend time on configuring infrastructure for the application.

## Repository:

Git

## Steps Overview:

- :Create dockerfile with steps required to build the application
- :Validate functionality of the application and push the image to docker hub.
- :Stress test the application.
- :Automate the build process.

## Pre-requisite:

- docker installed on local machine.
- Docker hub account.

-Added docker file and build image for the application.

The screenshot shows a VS Code interface with the following details:

- EXPLORER:** Shows files in the project: datasets, etc, include, lib, scripts, share, static, templates, .dockerignore, .gitignore, api.py, cleanandpreparedataset.ipynb, dockerfile, model\_v1.pkl, model.ipynb, model.pdf, model.py, pyenv.cfg, readPkl.py, requirements.txt, webapp.py, x\_data, README.md.
- Dockerfile — latam-sre:** The code editor displays the Dockerfile content:

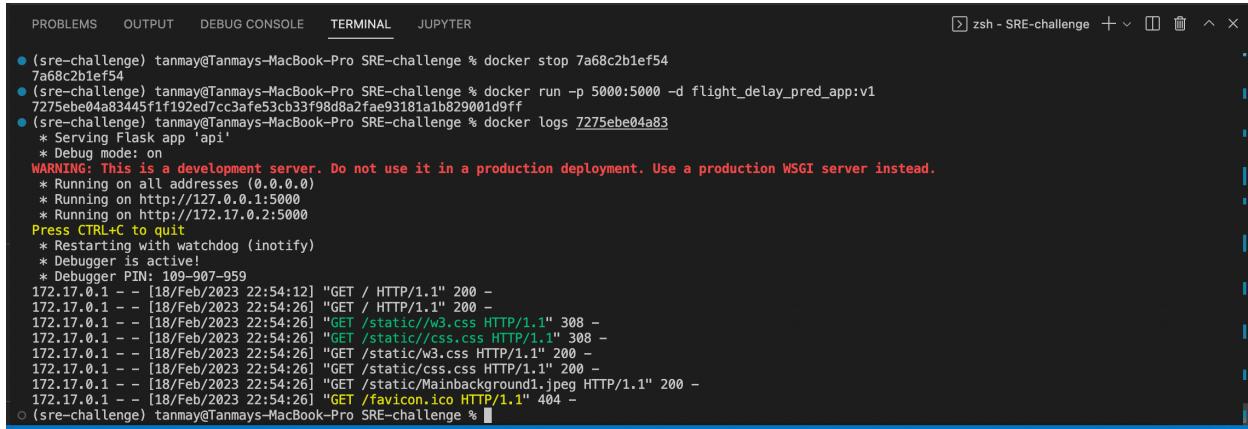
```
FROM python
COPY ./requirements.txt /app/requirements.txt
WORKDIR /app
# install the dependencies and packages in the requirements file
RUN pip install -r requirements.txt
# copy every content from the local file to the image
COPY . /app
# configure the container to run in an executed manner
ENTRYPOINT ["python"]
CMD ["api.py"]
```

- PROBLEMS:** Shows a single error: "dockerfile: model.py share".
- OUTPUT:** Shows the build logs for "flight\_delay\_pred\_app:v1":

```
[4] Building 335.3s [18/10] FINISHED
--> [internal] load build definition from Dockerfile
--> [internal] load .dockerignore
--> [internal] load context: 35B
--> [internal] load metadata for docker.io/library/python:latest
--> CACHED [4] 1.1MB from docker.io/library/python@sha256:a5b723f5b78cb73300b647ff49fba4ceea22c5ec68c547fa3291978e6b692259
--> [internal] load build context
--> [internal] transfer context 3.84KB
--> [2/5] COPY ./requirements.txt /app/requirements.txt
--> [3/5] WORKDIR /app
--> [4/5] RUN pip install -r requirements.txt
--> [5/5] COPY . /app
--> exporting to image
--> exporting layers
--> writing image sha256:acdc93905c3599d619e139b1bc12ea3f6e78f284a8d57f65351a73e038e485c2
--> naming to docker.io/library/flight_delay_pred_app:v1
(sre-challenge) tammy@Tammys-MacBook-Pro SRE-Challenge %
```

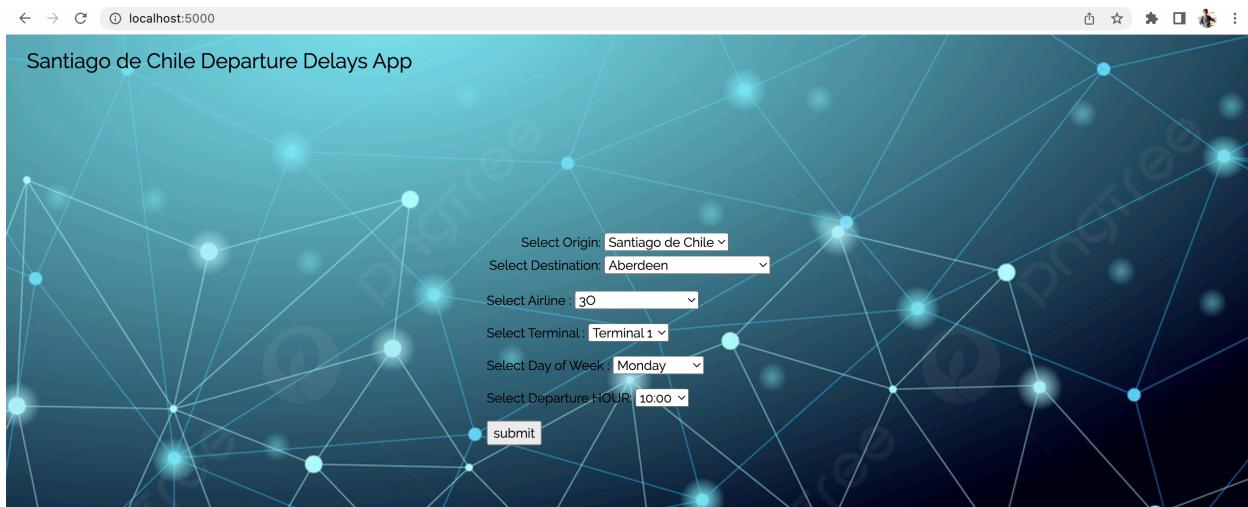
- TERMINAL:** Shows a terminal window with the command "Dockerfile".
- STATUS:** Shows file statistics: Lines: 18, Columns: 16, Spaces: 4, Encoding: UTF-8, LF, Dockerfile.

-Run the image and access it from localhost-



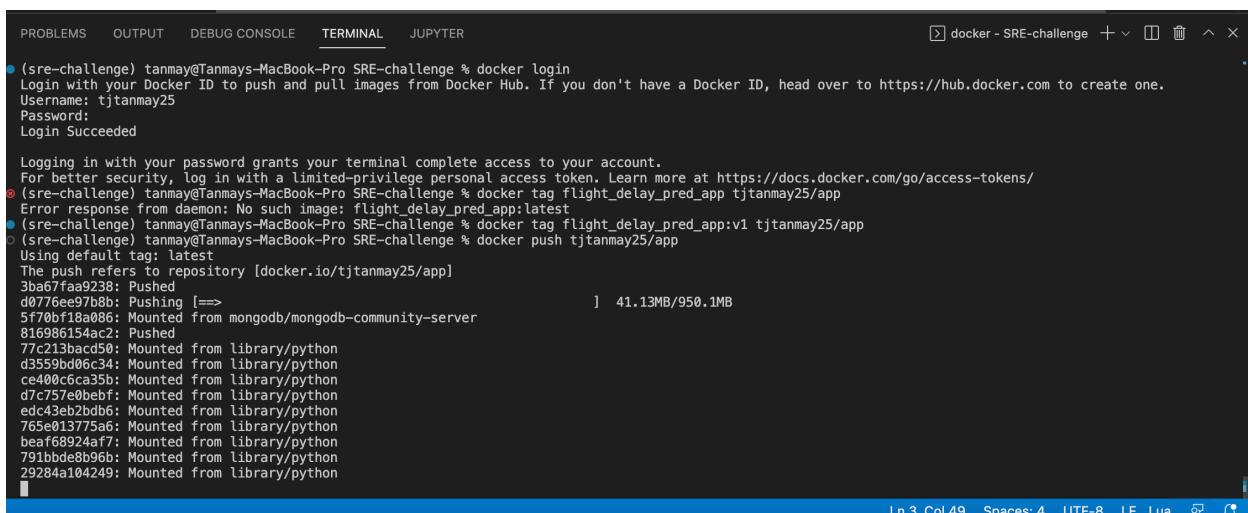
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER zsh - SRE-challenge + × □ □ ^ ×

● (sre-challenge) tanmay@Tannays-MacBook-Pro SRE-challenge % docker stop 7a68c2b1ef54
7a68c2b1ef54
● (sre-challenge) tanmay@Tannays-MacBook-Pro SRE-challenge % docker run -p 5000:5000 -d flight_delay_pred_app:v1
7275ebe04a834451f1f192ed7cc3afe53cb33f98d8a2fae93181a1b829001d9ff
● (sre-challenge) tanmay@Tannays-MacBook-Pro SRE-challenge % docker logs 7275ebe04a83
* Serving Flask app "api"
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with watchdog (inotify)
* Debugger is active!
* Debugger PIN: 109-907-959
172.17.0.1 -- [18/Feb/2023 22:54:12] "GET / HTTP/1.1" 200 -
172.17.0.1 -- [18/Feb/2023 22:54:26] "GET / HTTP/1.1" 200 -
172.17.0.1 -- [18/Feb/2023 22:54:26] "GET /static//w3.css HTTP/1.1" 308 -
172.17.0.1 -- [18/Feb/2023 22:54:26] "GET /static//css.css HTTP/1.1" 308 -
172.17.0.1 -- [18/Feb/2023 22:54:26] "GET /static/w3.css HTTP/1.1" 200 -
172.17.0.1 -- [18/Feb/2023 22:54:26] "GET /static/css.css HTTP/1.1" 200 -
172.17.0.1 -- [18/Feb/2023 22:54:26] "GET /static/Mainbackground1.jpeg HTTP/1.1" 200 -
172.17.0.1 -- [18/Feb/2023 22:54:26] "GET /favicon.ico HTTP/1.1" 404 -
○ (sre-challenge) tanmay@Tannays-MacBook-Pro SRE-challenge %
```



So our application which is running in Docker is accessible.

-Let's push the image in dockerhub repository.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER docker - SRE-challenge + × □ □ ^ ×

● (sre-challenge) tanmay@Tannays-MacBook-Pro SRE-challenge % docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: tjtanmay25
Password:
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
● (sre-challenge) tanmay@Tannays-MacBook-Pro SRE-challenge % docker tag flight_delay_pred_app tjtanmay25/app
Error response from daemon: No such image: flight_delay_pred_app:latest
● (sre-challenge) tanmay@Tannays-MacBook-Pro SRE-challenge % docker tag flight_delay_pred_app:v1 tjtanmay25/app
● (sre-challenge) tanmay@Tannays-MacBook-Pro SRE-challenge % docker push tjtanmay25/app
Using default tag: latest
The push refers to repository [docker.io/tjtanmay25/app]
3ba67faa9238: Pushed
d0776ea97b8b: Pushing [==>] 41.13MB/950.1MB
5f70bf18a080: Mounted from mongodb/mongodb-community-server
816986154ac2: Pushed
77c213bacd50: Mounted from library/python
d3559bd06c34: Mounted from library/python
ce400c6ca35b: Mounted from library/python
d7c757e0bebfb: Mounted from library/python
edc43eb2bdbb: Mounted from library/python
765e013775ab: Mounted from library/python
beaf68924af7: Mounted from library/python
791bbde8b96b: Mounted from library/python
29284a104249: Mounted from library/python
```

---

## Load Testing

```
[tanmay@Tanmays-MacBook-Pro SRE-challenge % docker logs flight_delay_pred_app
Error: No such container: flight_delay_pred_app
[tanmay@Tanmays-MacBook-Pro SRE-challenge % docker logs flight_delay_pred_app:v1
Error: No such container: flight_delay_pred_app:v1
[tanmay@Tanmays-MacBook-Pro SRE-challenge % wrk -t12 -c50000 -d45s -s ./scripts/post.lua http://127.0.0.1:5000/
./scripts/post.lua: ./scripts/post.lua:2: unexpected symbol near '10'
Running 45s test @ http://127.0.0.1:5000/
 12 threads and 50000 connections
 Thread Stats      Avg      Stdev     Max   +/- Stdev
  Latency    197.99ms  364.31ms   1.99s   90.22%
  Req/Sec    104.48     80.42   660.00   68.77%
 26989 requests in 45.09s, 556.27MB read
  Socket errors: connect 47447, read 4850, write 383, timeout 1281
Requests/sec:    598.62
Transfer/sec:    12.34MB
tanmay@Tanmays-MacBook-Pro SRE-challenge % ]
```

---

## Automate build process

We will use Git action to automate the build process and will create CI pipeline, which will build and push the image into docker hub.

- Created new branch [https://github.com/tjtanmay/latam-sre/tree/feature/add\\_git\\_workflow](https://github.com/tjtanmay/latam-sre/tree/feature/add_git_workflow)
- Added the workflow file in master and merged latest changes in master branch.
- After successfully executing GitHub action pipeline, image will be pushed to Docker hub.

## Github Action pipeline run- [Link](#)

The screenshot shows a GitHub Actions pipeline run for the repository `tjtanmay/latam-sre`. The pipeline has completed successfully. The main summary table lists the following steps:

Step	Description	Duration
> Set up job		1s
> Run actions/checkout@v1		10s
> Login to Docker Hub		0s
> Build Docker image		10m 44s
> Tag Docker image		0s
> Publish Docker image		59s
> Complete job		0s

## Image pushed to docker hub- [Link](#)

The screenshot shows the Docker Hub interface for the image `tjtanmay25/app:latest`. The image was pushed a minute ago by the user `tjtanmay25`. The image details are as follows:

- DIGEST: `sha256:49b187ae330f33bc577be99a94a4a6945d551db4f63efbbb1d729d2fa6f0267f`
- OS/ARCH: `linux/amd64`
- COMPRESSED SIZE: `1002.38 MB`
- LAST PUSHED: `a minute ago`
- TYPE: `Image`

Below the image details, there are tabs for `Image Layers` and `Vulnerabilities`. The `Image Layers` tab shows two layers:

Layer	Command	Size
1	ADD file ... in /	52.5 MB
2	CMD ["bash"]	0 B

The `Command` field in the second layer shows the command `ADD file:b03d13d345c29f69557f410c8504e748226756d1f48e5abdb63cd40179b2640c in /`.

## Validate image:

-Let's test the image and run-



```
[tanmay@Tanmays-MacBook-Pro SRE-challenge % docker pull tjtanmay25/app:latest
latest: Pulling from tjtanmay25/app
1e4aec178e08: Pull complete
6c1024729fee: Pull complete
c3aa11fbcb85a: Pull complete
aa54add66b3a: Pull complete
9e3a60c2bce7: Pull complete
3b2123ce9d0d: Pull complete
05df7720fcb8: Pull complete
972ab8743e38: Pull complete
ae9f20f2cd37: Pull complete
66657012bff4: Pull complete
c94b04b84a68: Pull complete
039406b7ecf8: Pull complete
Digest: sha256:49b187ae330f33bc577be99a94a4a6945d551db4f63efbbb1d729d2fa6f0267f
Status: Downloaded newer image for tjtanmay25/app:latest
docker.io/tjtanmay25/app:latest
```

-Running locally



```
[tanmay@Tanmays-MacBook-Pro SRE-challenge % docker run -p 5001:5000 tjtanmay25/app
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
 * Serving Flask app 'api'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.3:5000
Press CTRL+C to quit
 * Restarting with watchdog (inotify)
 * Debugger is active on 0.0.0.0:5001
 * Documentation: http://127.0.0.1:5001/docs/
172.17.0.1 - - [19/Feb/2023 00:49:42] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2023 00:49:42] "GET /static/css.css HTTP/1.1" 308 -
172.17.0.1 - - [19/Feb/2023 00:49:42] "GET /static/w3.css HTTP/1.1" 308 -
172.17.0.1 - - [19/Feb/2023 00:49:43] "GET /static/Mainbackground1.jpeg HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2023 00:49:43] "GET /static/w3.css HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2023 00:49:43] "GET /static/css.css HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2023 00:49:43] "GET /favicon.ico HTTP/1.1" 404 -
/usr/local/lib/python3.11/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but AdaBoostClassifier was fitted with feature names
warnings.warn
172.17.0.1 - - [19/Feb/2023 00:50:21] "POST /result HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2023 00:50:21] "GET /static/css.css HTTP/1.1" 304 -
```

Application is working fine.

## Conclusion:

We deployed the application in docker and pushed the image into docker hub repository.

We can see number of request handled by the application increased significantly. Also we automated the process of building docker image using GitHub actions.

## App running on Kubernetes cluster-

### Performance:

Now we will deploy the application on kubernetes cluster. This will make scalability easier and as per the load usage, we can easily scale up the infrastructure i.e. cluster.

### Repository:

[Git](#)

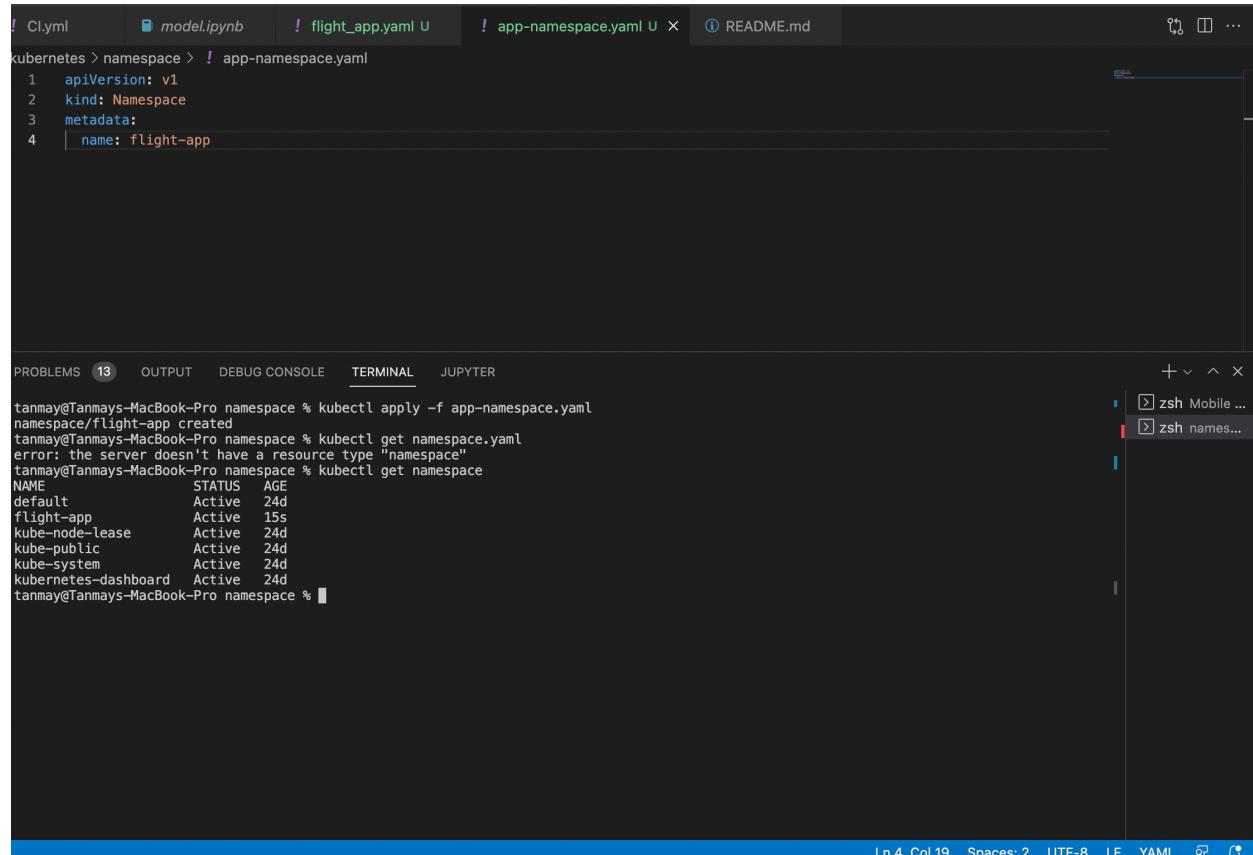
### Steps Overview:

- :Write deployment and service file.
- :Expose the service.
- :Stress test the application.

### Pre-requisite:

- minikube installed on local system.

### -Create namespace-



The screenshot shows a dark-themed code editor interface with a terminal window open at the bottom. The terminal window displays the following command and its output:

```
tanmay@Tammays-MacBook-Pro namespace % kubectl apply -f app-namespace.yaml
namespace/flight-app created
tanmay@Tammays-MacBook-Pro namespace % kubectl get namespace.yaml
error: the server doesn't have a resource type "namespace"
tanmay@Tammays-MacBook-Pro namespace % kubectl get namespace
NAME      STATUS   AGE
default   Active   24d
flight-app   Active   15s
kube-node-lease   Active   24d
kube-public   Active   24d
kube-system   Active   24d
kubernetes-dashboard   Active   24d
tanmay@Tammays-MacBook-Pro namespace %
```

The terminal window is part of a larger interface with tabs for Cl.yaml, model.ipynb, flight\_app.yaml, app-namespace.yaml, and README.md. The bottom status bar indicates the terminal has 13 problems.

-Added deployment file for pod creation and executed-

The screenshot shows a code editor interface with several tabs at the top: Cl.yaml, model.ipynb, flight\_app.yaml (active), app-namespace.yaml, and README.md. The flight\_app.yaml tab contains the following deployment configuration:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flight-app
  namespace: flight-app
  labels:
    app: flight-app
spec:
  selector:
    matchLabels:
      app: flight-app
  template:
    metadata:
      labels:
```

Below the editor, a terminal window displays the execution of the deployment command:

- tanmay@Tanmays-MacBook-Pro deployment % kubectl apply -f flight\_app.yaml
- deployment.apps/flight-app created
- tanmay@Tanmays-MacBook-Pro deployment % kubectl get pods -n flight-app

NAME	READY	STATUS	RESTARTS	AGE
flight-app-6b46bb87f8-8svcq	0/1	ContainerCreating	0	36s

- tanmay@Tanmays-MacBook-Pro deployment %

*Note: for minikube, you have to download image manually into minikube context using below command-*

minikube ssh docker pull tjtanmay25/app:latest

The terminal window shows the process of pulling the Docker image "tjtanmay25/app:latest" into the minikube context:

- Normal Pulling 35s (x2 over 2m46s) kubelet Pulling image "tjtanmay25/app:latest"
- tanmay@Tanmays-MacBook-Pro deployment % minikube ssh docker pull tjtanmay25/app:latest
- latest: Pulling from tjtanmay25/app
- 1e4aec178e08: Already exists
- 6c1024729fee: Already exists
- c3aa1fbc85a: Already exists
- aa54add66b3a: Already exists
- 9e3a60c2bce7: Already exists
- 3b2123ce9d0d: Already exists
- 05df7720fc88: Already exists
- 972ab8743e38: Already exists
- ae9f20f2cd37: Already exists
- 919f778810df: Pull complete
- 1b689b3ea0c5: Pull complete
- e2f05f17fa30: Pull complete
- Digest: sha256:6b1f9a89c635fb2d225b09787ef87d189172932095ebbf4ed839eaf0d1a88658
- Status: Downloaded newer image for tjtanmay25/app:latest
- docker.io/tjtanmay25/app:latest
- tanmay@Tanmays-MacBook-Pro deployment % kubectl apply -f flight\_app.yaml
- deployment.apps/flight-app unchanged
- tanmay@Tanmays-MacBook-Pro deployment % kubectl get pods -n flight-app

NAME	READY	STATUS	RESTARTS	AGE
flight-app-5d858c4d87-dwx8j	1/1	Running	0	6m51s

- tanmay@Tanmays-MacBook-Pro deployment %

-We have to expose the application, for this added service.yaml

```
● tanmay@Tanmays-MacBook-Pro kubernetes % cd service
● tanmay@Tanmays-MacBook-Pro service % kubectl apply -f service.yaml
service/flight-app created
✖ tanmay@Tanmays-MacBook-Pro service % kubectl describe service flight-app
Error from server (NotFound): services "flight-app" not found
● tanmay@Tanmays-MacBook-Pro service % kubectl describe service flight-app -n flight-app
Name:                  flight-app
Namespace:             flight-app
Labels:                <none>
Annotations:           <none>
Selector:              app=flight-app
Type:                  LoadBalancer
IP Family Policy:     SingleStack
IP Families:          IPv4
IP:                   10.99.73.134
IPs:                  10.99.73.134
Port:                 <unset>  5000/TCP
TargetPort:            5000/TCP
NodePort:              <unset>  31453/TCP
Endpoints:             172.17.0.2:5000
Session Affinity:      None
External Traffic Policy: Cluster
Events:                <none>
○ tanmay@Tanmays-MacBook-Pro service %
```

service.yaml — latam-sre

! Cl.yml ! flight\_app.yaml U < home.html ! service.yaml U X ⚡ .gitignore U ! app-namespace.yaml U ⓘ README.md ⌂ ⌂ ⌂

kubernetes > service > ! service.yaml

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: flight-app
5   namespace: flight-app
6 spec:
7   selector:
8     app: flight-app
9   ports:
10    - port: 5000
11      targetPort: 5000
12   type: LoadBalancer
13   externalIPs:
14     - 192.168.49.2
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

- tanmay@Tammays-MacBook-Pro latam-sre % kubectl get service -n flight-app

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
flight-app	NodePort	10.99.73.134	<none>	5000:31453/TCP	35m
- tanmay@Tammays-MacBook-Pro latam-sre % minikube ip  
192.168.49.2
- tanmay@Tammays-MacBook-Pro latam-sre % ls

README.md	dockerfile	model.py	share
__pycache__	etc	model_v1.pkl	static
api.py	kubernetes	pyenv.cfg	templates
bin	lib	readPkl.py	webapp.py
cleanandpreparedataset.ipynb	model.ipynb	requirements.txt	x_data
datasets	model.pdf	scripts	
- tanmay@Tammays-MacBook-Pro latam-sre % cd kubernetes/service
- tanmay@Tammays-MacBook-Pro service % ls
- tanmay@Tammays-MacBook-Pro service % kubectl apply -f service.yaml
- tanmay@Tammays-MacBook-Pro service % kubectl apply -f service.yaml
- tanmay@Tammays-MacBook-Pro service % kubectl apply -f service.yaml
- tanmay@Tammays-MacBook-Pro service % kubectl get service -n flight-app

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
flight-app	LoadBalancer	10.99.73.134	127.0.0.1,192.168.49.2	5000:31453/TCP	65m
- tanmay@Tammays-MacBook-Pro service %

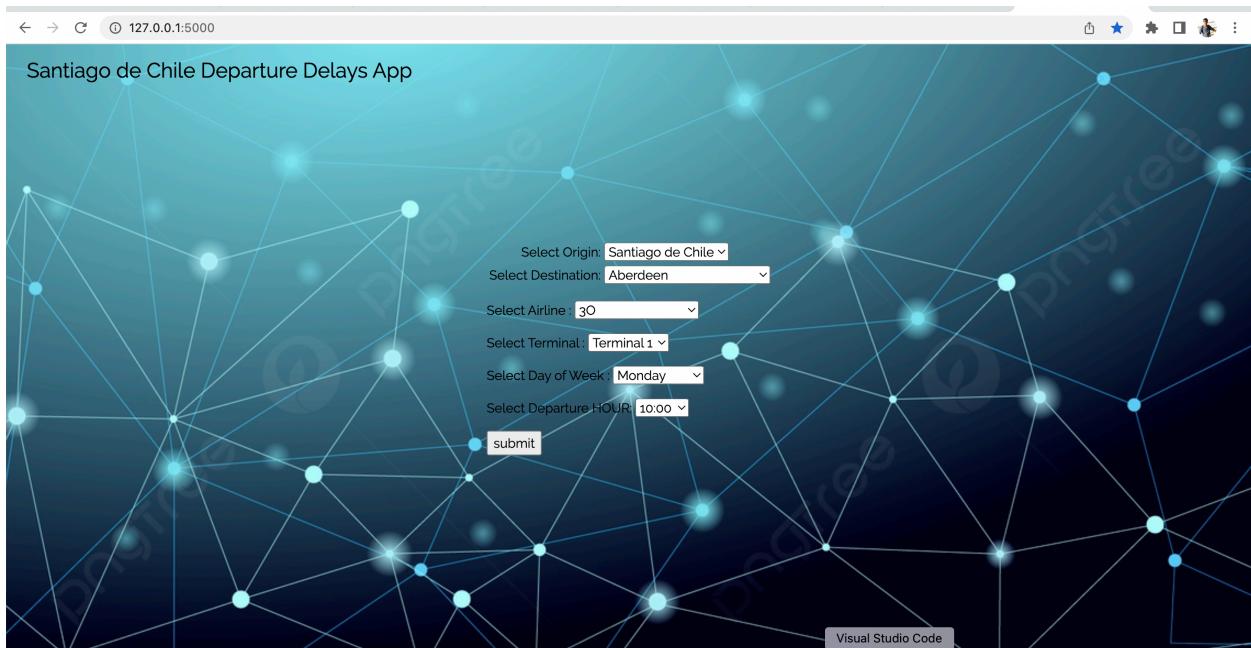
Note: For minikube, need to open tunnel in order to access the service

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
tanmay@Tanmays-MacBook-Pro latam-sre % minikube tunnel
  ✓ Tunnel successfully started

  ⚠ NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...
  ⚡ Starting tunnel for service flight-app.
```

Result:

Application is accessible



### Autoscale:

```
kubectl autoscale deployment flight-app --cpu-percent=80 --min=1 --max=4
```

This will increase pods to a maximum of four replicas (--max=4) when the PHP web application deployment experiences more than 80% CPU use (--cpu-percent=80) over a sustained period.

---

## Load Testing

```
tanmay@Tanmays-MacBook-Pro latam-sre % wrk -t12 -c50000 -d45s -s ./scripts/post.lua http://127.0.0.1:5000/
./scripts/post.lua: ./scripts/post.lua:2: unexpected symbol near '10'
[./scripts/post.lua: ./scripts/post.lua:2: unexpected symbol near '10'
./scripts/post.lua: ./scripts/post.lua:2: unexpected symbol near '10'
Running 45s test @ http://127.0.0.1:5000/
  12 threads and 50000 connections
 Thread Stats      Avg      Stdev     Max   +/- Stdev
  Latency    845.40ms  481.26ms  2.00s   60.71%
  Req/Sec     27.22     40.99   370.00   90.37%
  1031 requests in 45.09s, 21.26MB read
  Socket errors: connect 47447, read 650130, write 250, timeout 634
Requests/sec:    22.87
Transfer/sec:   482.74KB
tanmay@Tanmays-MacBook-Pro latam-sre % wrk -t12 -c50000 -d45s -s ./scripts/post.lua http://127.0.0.1:5000/
```

## Conclusion:

We have successfully deployed the application in kubernetes infrastructure. As we can see, load test results not showing improvement. This is because, kubernets is running locally and application is running on top of that. So performance depends upon the local system limit. In this case, kubernetes is using significant amount of resources from local system.

---

## Managing infrastructure using Terraform

As of now, we have automated build process for the application using Github action which is pushing build artifacts to docker hub repository. Then we created Kubernetes cluster manually and deployed the application. Let's introduce terraform to manage the Kubernetes infrastructure. We will write configuration for Kubernetes infrastructure in .tf files.

### Git

- Add provider.tf file to manage the Kubernetes API.
- Add k8.tf defining the resource.
  
- Run terraform init to download modules and registry.
- Run terraform validation to check the configuration.
- Run terraform plan for dry run.



```
provider.tf — latam-sre
EXPLORER
LATAM-SRE
> etc
< kubernetes
deployment
! flight_app.yaml
> namespace
> service
> lib
> scripts
> share
> static
templates
home.html
results.html
> terraform
> .terraform
.terraform.lock.hcl
k8.tf
provider.tf
.gitignore
api.py
cleanandpreparedataset.ipynb
dockerfile
model_v1.pkl
model.joblib
OUTLINE
> terraform
> required_providers
> kubernetes
source
version
> provider "kubernetes"
TIMELINE
provider.tf — latam-sre
README.md .gitignore M ! flight_app.yaml provider.tf x k8.tf M
terraform > provider.tf > terraform
1  terraform {
2    required_providers {
3      kubernetes = [
4        source = "hashicorp/kubernetes"
5        version = "2.11.0"
6      ]
7    }
8  }
9
10 provider "kubernetes" {
11   config_path    = "~/.kube/config"
12   config_context = "minikube"
13 }

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
(sre-challenge) tanmay@Tanmays-MacBook-Pro terraform % terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/kubernetes versions matching "2.11.0"...
- Installing hashicorp/kubernetes v2.11.0...
- Installed hashicorp/kubernetes v2.11.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.
(sre-challenge) tanmay@Tanmays-MacBook-Pro terraform % terraform plan
Ln 1, Col 1 Spaces: 4 UTF-8 LF Terraform
```

Terraform plan gives output-

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
+ vsphere_volume {
+   fs_type      = (known after apply)
+   volume_path = (known after apply)
}
}
}
}
}

# kubernetes_namespace.flight-app will be created
+ resource "kubernetes_namespace" "flight-app" {
+   id = (known after apply)

+   metadata {
+     generation      = (known after apply)
+     name           = "flight-app"
+     resource_version = (known after apply)
+     uid            = (known after apply)
}
}

Plan: 2 to add, 0 to change, 0 to destroy.
```

-Now apply using terraform apply

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following Terraform code and its execution results:

```
+ vsphere_volume {
+   fs_type    = (known after apply)
+   volume_path = (known after apply)
}
}
}
}

# kubernetes_namespace.flight-app will be created
+ resource "kubernetes_namespace" "flight-app" {
+   id = (known after apply)

+   metadata {
+     generation      = (known after apply)
+     name           = "flight-app"
+     resource_version = (known after apply)
+     uid            = (known after apply)
}
}

Plan: 2 to add, 0 to change, 0 to destroy.
```

**Result:**

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following Terraform code and its execution results:

```
40   selector = local.flighthash_labels
41   port {
42     port      = 5000
43     target_port = 5000
44     node_port = 32001
45   }
46   type = "LoadBalancer"
47 }
48 }
```

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.

Enter a value: yes

```
kubernetes_service.flight-app-service: Creating...
kubernetes_deployment.flight-app: Creating...
kubernetes_service.flight-app-service: Creation complete after 0s [id=default/flight-app-service]
kubernetes_deployment.flight-app: Creation complete after 4s [id=default/flight-app]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

(sre-challenge) tamay@Tanmays-MacBook-Pro:~/Desktop/latam-sre\$ minikube service flight-app-service --url  
http://127.0.0.1:60873  
Because you are using a Docker driver on darwin, the terminal needs to be open to run it.

## Conclusion:

Now we can manage our infrastructure using terraform easily. This doesn't affect the performance of the application as terraform manages underlying infrastructure which is Kubernetes in this case.

## Implementing Monitoring using Prometheus and Grafana

We can leverage prometheus for monitoring our infra and application .

- Install Prometheus using helm:

```
$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts  
$ helm repo update
```

```
$ helm install prometheus prometheus-community/prometheus
```

- Install Grafana using helm:

```
$ helm repo add grafana https://grafana.github.io/helm-charts  
$ helm install grafana stable/grafana
```

- Expose the services for prometheus and grafana to access the dashboard.
- Follow instruction for the admin credential for grafana dashboard.
- Add prometheus datasource in grafana.
- Access Prometheus and grafana dashboards.
- Fetch service url for both the dashboards.

The screenshot shows a terminal window with several commands run on a minikube cluster. The commands include:

- `kubectl get pods`: Shows pods for Prometheus components (kube-state-metrics, node-exporter, alertmanager, pushgateway, server) and Grafana.
- `kubectl get operator`: Returns an error message about missing 'operator' resource type.
- `kubectl get crd`: Returns 'No resources found'.
- `kubectl get svc`: Shows services for various components like kubernetes, nginx-service, and Prometheus components.
- `minikube service prometheus-server-np --url`: Prints the URL for the Prometheus service.
- `http://127.0.0.1:16124`: Prints the URL for the Prometheus service.

A status bar at the bottom indicates: "Because you are using a Docker driver on darwin, the terminal needs to be open to run it."

helm.tf — latam-sre

① README.md    helm.tf    provider.tf

terraform > helm.tf > locals > flightapp\_labels > Tier

```

1
2
3 locals {
4   flightapp_labels = {
5     App = "flight-app"
6     Tier = "frontend"
7   }
8 }
9
10 resource "helm_release" "flightapp" {
11   name  = "flight-app"
12   chart = "${abspath(path.root)}/charts/flightapp-chart"
13 }
14

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

grafana.default.svc.cluster.local

Get the Grafana URL to visit by running these commands in the same shell:

```

export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=grafana,app.kubernetes.io/instance=grafana" -o jsonpath='{.items[0].metadata.name}')
kubectl --namespace default port-forward $POD_NAME 3000

```

3. Login with the password from step 1 and the username: admin

```

#####
##### WARNING: Persistence is disabled!!! You will lose your data when #####
#####           the Grafana pod is terminated. #####
#####
● tanmay@tanmays-MacBook-Pro latam-sre % kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-np

```

service/grafana-np exposed

● tanmay@tanmays-MacBook-Pro latam-sre % kubectl get secret --namespace default grafana -o jsonpath=".data.admin-password" | base64 --decode ; echo

```

ZIRISvksXqUJbeXqYxUwZ8wsJKpuJB0LcNcvwrEf
○ tanmay@tanmays-MacBook-Pro latam-sre % minikube service grafana-np --url

```

<http://127.0.0.1:61675>

! Because you are using a Docker driver on darwin, the terminal needs to be open to run it.

- Access the url's

Prometheus Time Series Collector    Prometheus - Data sources -

127.0.0.1:61624/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show\_exemplars=0&g0.range\_input=1h

Prometheus    Alerts    Graph    Status    Help

Use local time     Enable query history     Enable autocomplete     Enable highlighting     Enable linter

Expression (press Shift+Enter for newlines)    Execute

Table    **Graph**

Evaluation time

No data queried yet

Add Panel    Remove Panel

The screenshot shows the 'Data Sources / Prometheus' configuration page in the Prometheus UI. The URL is 127.0.0.1:61675/datasources/edit/rmZDGXJ4k. The 'Settings' tab is selected. The 'Name' field is set to 'Prometheus'. Under 'HTTP', the 'URL' is 'http://prometheus-server:80'. The 'Auth' section includes options for 'Basic auth', 'TLS Client Auth', 'Skip TLS Verify', and 'Forward OAuth Identity', all of which are disabled.

---

## SLO's

- **SLOs: Service Level Objective**
  - What you have internally set as a target, driving your measuring threshold (for example, on dashboards and alerting). In general, it should be stricter than your SLA.
  - Example: "99.9%" availability (the so called "three 9s").
  - Keyword: *thresholds*
- **SLIs: Service Level Indicators**
  - What you actually measure, to ascertain whether your SLOs are on/off-target.
  - Example: error ratios, latency
  - Keyword: *metrics*

-Will use iter8 tool to find the SLO-

```
iter8 k launch -c load-test-http --noDownload \
--set url=http://127.0.0.1:60093 \
--set SL0s.http/error-rate=0 \
--set SL0s.http/latency-mean=50 \
--set SL0s.http/latency-p90=100 \
--set SL0s.http/latency-p95=200
```

Using this configuration, validates that the service satisfies the following SLOs.

- Error rate is 0
- Mean latency is under 50 msec
- 90 percentile latency is under 100 msec
- 95 percentile latency is under 200 msec

```
[tanmay@Tanmays-MacBook-Pro charts % iter8 k assert -c completed -c nofailure -c slos --timeout 120s
INFO[2023-02-21 00:26:52] initied Helm config
INFO[2023-02-21 00:26:52] experiment completed
INFO[2023-02-21 00:26:52] experiment has no failure
INFO[2023-02-21 00:26:52] experiment does not involve any SLOs
INFO[2023-02-21 00:26:52] SLOs are satisfied
INFO[2023-02-21 00:26:52] all conditions were satisfied
[tanmay@Tanmays-MacBook-Pro charts % ls
charts           iter8lib          load-test-grpc        load-test-http        slo-validation-istio
tanmay@Tanmays-MacBook-Pro charts % iter8 k report
INFO[2023-02-21 00:27:08] initied Helm config

Experiment summary:
*****
Experiment completed: true
No task failures: true
Total number of tasks: 1
Number of completed tasks: 1

Latest observed values for metrics:
*****
Metric          |value
----|----
http/error-count |0.00
http/error-rate  |0.00
http/latency-max (msec) |4.33
http/latency-mean (msec) |0.98
http/latency-min (msec) |0.25
http/latency-p50 (msec) |0.82
http/latency-p75 (msec) |1.36
http/latency-p90 (msec) |1.96
http/latency-p95 (msec) |3.00
http/latency-p99 (msec) |4.00
http/latency-p99.9 (msec) |4.30
http/latency-stdev (msec) |0.83
http/request-count |100.00

tanmay@Tanmays-MacBook-Pro charts % iter8 k log
INFO[2023-02-21 00:27:32] initied Helm config
INFO[2023-02-21 00:27:32] experiment logs from Kubernetes cluster      indented-trace=below ...
time=2023-02-21 06:26:05 level=info msg=task 1: gen-load-and-collect-metrics-http : started
time=2023-02-21 06:26:17 level=info msg=task 1: gen-load-and-collect-metrics-http : completed
```

The screenshot shows a web browser window titled "Experiment Report" with the URL "file:///Users/tanmay/Desktop/projects/repo/latam-sre/scripts/report1.html". The main content is titled "Iter8 Experiment Report". A modal dialog box is open, titled "Experiment Status" with a green thumbs-up icon. It displays the message: "Experiment completed. Experiment has no failures. 2 out of 2 tasks are complete." Below this, there is a section titled "Service level objectives (SLOs)" with the subtitle "Whether or not SLOs are satisfied". A table lists four SLO conditions, each with a green checkmark in the "Satisfied" column:

SLO Conditions	Satisfied
<code>http/error-rate ≤ 0</code>	✓
<code>http/latency-mean (msec) ≤ 50</code>	✓
<code>http/latency-p90 (msec) ≤ 100</code>	✓
<code>http/latency-p95 (msec) ≤ 200</code>	✓

## Application Security

Endpoints must be authenticated before they are allowed to make requests in an application. Authentication means that the endpoint has an existing session and is unique to a specific user.

There are several ways to create an authentication layer in web applications one of them is Token Based Authentication. However, it is worth noting that token-based authentication scales better than that of a session because tokens are stored on the client-side while the session makes use of the server memory so it might become an issue when there is a large number of users using the system at once.

- Easy to implement.
- Provides a good level of security (by industry standards).
- Fast and performant.

## What is a Security Token?

In a broad way a token is a "number that proves something", for example: When you finish making a bank transfer, the bank sends a confirmation "token" serves as proof to validate that the transaction exists and it's valid. That confirmation number could be also called a confirmation token.

Tokens used for authentication need to be more than normal just numbers, they need to be almost impossible to fake, predict or break.

- Non-consecutive, that will make them very predictable, hackers will guess the next one.
- Infinite (almost)
- Non-reusable: There are cases of re-usable tokens, but in general once a token is generated no one else should every use it but you.
- Validatable: The token must follow some hidden pattern (encryption) that allows validating the token without compromising the owner or author.

The most simple way to implement authentication in your database and API:

1. Create a `User` table/model that represents every user inside your application.
2. That User table must contain email and password for every user.
3. Create one api endpoint called `POST /token` that generates a token only if it receives an email and password that matches in the database.
4. The `POST /token` endpoint will return the token to the front-end if everything is ok.
5. Then, on every other endpoint in your database you will have to validate if the token exists in the request header and if it does you will have to validate it.

## Every token is a session

The moment you generate the token you can decide if you want it to expire, same way web sessions expire when you log in into your online bank account.

When a client successfully authenticates it will receive that unique token and it will be able to attach to the request headers of every request it makes from that moment on, that token will be the "User session".

It is recommended to save that token in the cookies or localStorage of your front-end application.