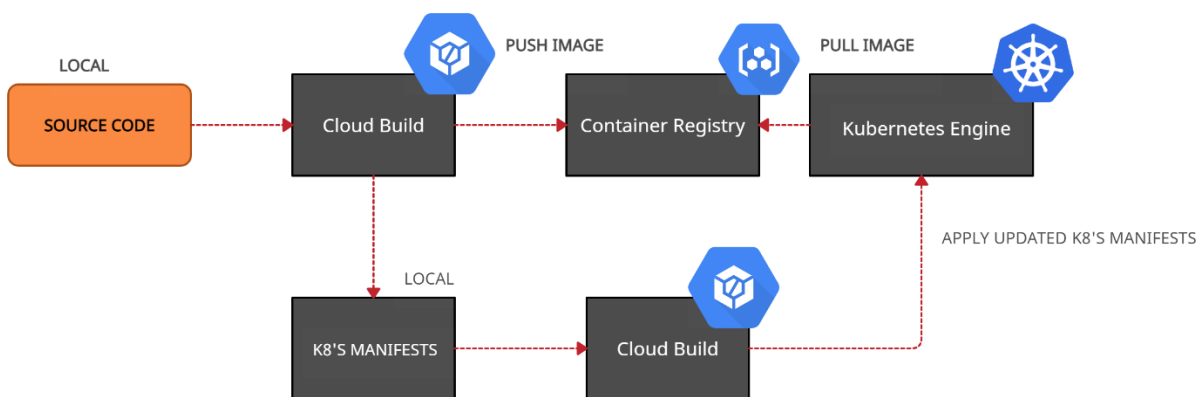


# GCP DEVOPS PROJECT

**Problem Statement:** Create a sample application using any technology along with microservices to demonstrate implementation of the DevOps principle using GCP services. Establish a pipeline for continuous integration, continuous testing, and continuous deployment.



**Motivation:** As with growth of Machine Learning, most of the organization are shifting their whole ML workloads to cloud platform so that they can achieve High Availability and utilize high performance, As there are a lot of cloud platforms available to deploy our whole ML models. Some are AWS, GCP, Azure. For this Project I have selected GCP as one of the cloud platforms.

**Application:** A simple web app with minimal framework using PyTorch and Streamlit to showcase an image classification model, where user will input any kind of image and just wait for the results which can be use as fun quizzes .

5 predictions should be outputted from highest probability to lowest probability.

We can input any image from our local system and it will give us the 5 closest answers to the image ranking from most matching probability to least probability.

```

1  """Create an Image Classification Web App using PyTorch and Streamlit."""
2  # import Libraries
3  from PIL import Image
4  from torchvision import models, transforms
5  import torch
6  import streamlit as st
7
8  # set title of app
9  st.title("Simple Image Classification Application")
10 st.write("")
11
12 # enable users to upload images for the model to make predictions
13 file_up = st.file_uploader("Upload an image", type = "jpg")
14
15
16 def predict(image):
17     """Return top 5 predictions ranked by highest probability.
18
19     Parameters
20     -----
21     :param image: uploaded image
22     :type image: jpg
23     :rtype: list
24     :return: top 5 predictions ranked by highest probability
25     """
26     # create a ResNet model
27     resnet = models.resnet101(pretrained = True)
28
29     # transform the input image through resizing, normalization
30     transform = transforms.Compose([
31         transforms.Resize(256),
32         transforms.CenterCrop(224),
33         transforms.ToTensor(),
34         transforms.Normalize(
35             mean = [0.485, 0.456, 0.406],
36             std = [0.229, 0.224, 0.225]
37         )])
38
39     # Load the image, pre-process it, and make predictions
40     img = Image.open(image)
41     batch_t = torch.unsqueeze(transform(img), 0)
42     resnet.eval()
43     out = resnet(batch_t)
44
45     with open('imagenet_classes.txt') as f:
46         classes = [line.strip() for line in f.readlines()]
47
48     # return the top 5 predictions ranked by highest probabilities
49     prob = torch.nn.functional.softmax(out, dim = 1)[0] * 100
50     _, indices = torch.sort(out, descending = True)
51     return [(classes[idx], prob[idx].item()) for idx in indices[0][:5]]
52
53
54 if file_up is not None:
55     # display image that user uploaded
56     image = Image.open(file_up)
57     st.image(image, caption = 'Uploaded Image.', use_column_width = True)
58     st.write("")
59     st.write("Just a second ...")
60     labels = predict(file_up)
61
62     # print out the top 5 prediction labels with scores
63     for i in labels:
64         st.write("Prediction (index, name)", i[0], ",   Score: ", i[1])
65

```

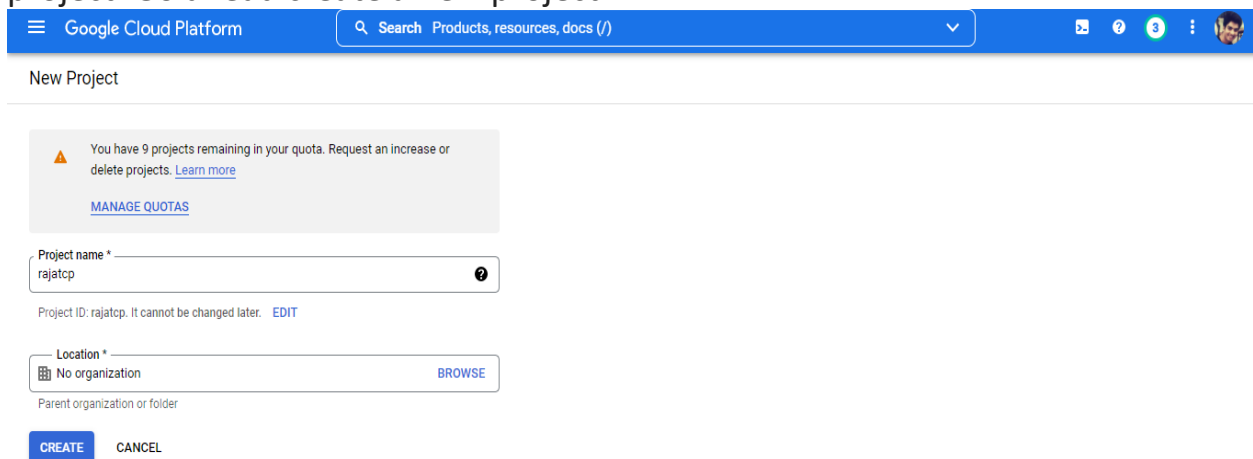
## Introduction

Once we have developed and deployed the application/product, It needs to be continuously updated based on user feedback or the addition of new features. This process should be automated, as without automation we have to run the same development and deployment steps/commands again and again for every change to the application.

With Continuous Integration and Continuous Delivery Pipeline, we can automate the complete workflow from building, testing, packaging, and deploying, which will be triggered when there are any changes to an existing application or we can say if there is any new commit to an existing code repository.

## Project Setup

Once the account is created, on the home page there is an option to create a project. Go ahead create a new project.



The screenshot shows the 'New Project' page in the Google Cloud Platform console. At the top is a blue header with the Google Cloud Platform logo, a search bar, and navigation links. Below the header, the page title 'New Project' is displayed. A warning message states: 'You have 9 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)'. Below this is a link to 'MANAGE QUOTAS'. The 'Project name' field is set to 'rajatcp' with a help icon. Below it, a note says 'Project ID: rajatcp. It cannot be changed later. [EDIT](#)'. The 'Location' field is set to 'No organization' with a 'BROWSE' link. At the bottom are 'CREATE' and 'CANCEL' buttons.

Google Cloud Platform

Search Products, resources, docs (/)

New Project

You have 9 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name \*  
rajatcp

Project ID: rajatcp. It cannot be changed later. [EDIT](#)

Location \*  
No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

I have created a project, **rajatcp** so I will use that for setting up a complete pipeline.

## Container Registry

It is used to store Docker Container images, Similar to DockerHub, AWS ECS, and other private cloud container registries.

### Benefits of Container Registry

- Secure, Private Docker Registry
- Build and deploy automatically
- In-depth vulnerability scanning
- Lockdown risky images
- Native Docker support
- Fast, high-availability access

To use the service, we need to activate the API, search for **Container registry** in the search bar, and on the next page click on **Enable Container Registry API** button.

## Google Kubernetes Engine (GKE)

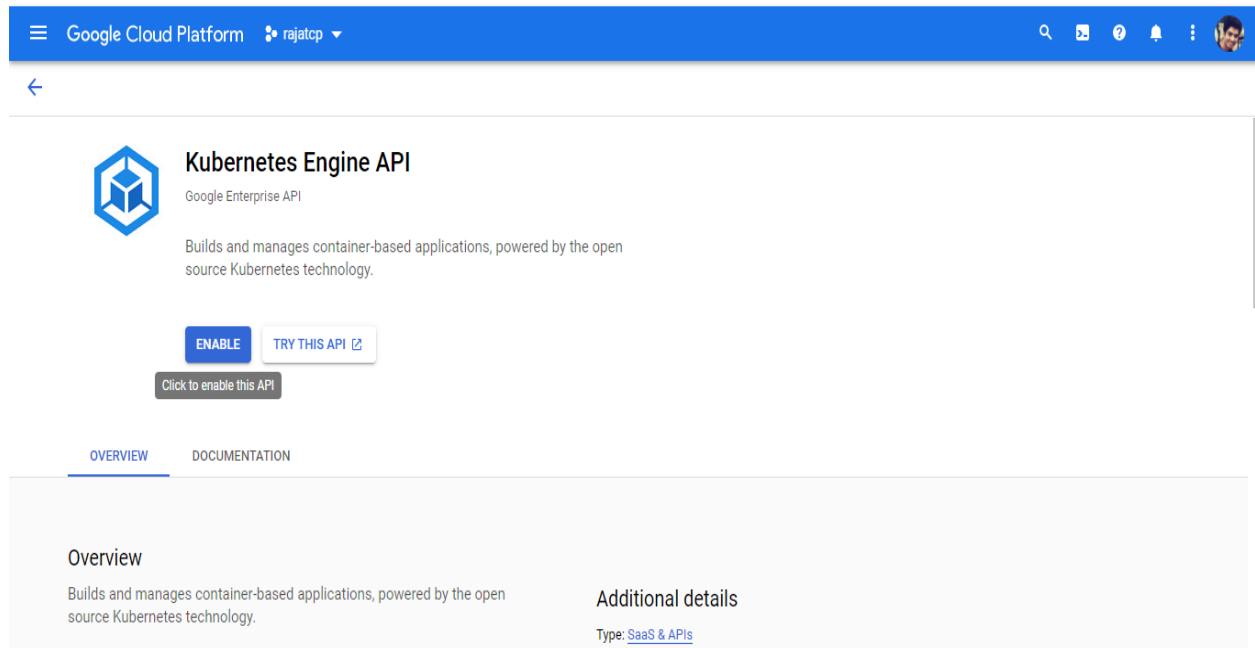
[Kubernetes](#), also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

GKE is Kubernetes managed by Google infrastructure. Some of the benefits are:

- Auto Scale
- Auto Upgrade
- Auto Repair
- Logging and Monitoring
- Load Balancing

Now to use the service, we need to enable the API. Search for Kubernetes Engine API in the search bar and enable the API.

Once the API is enabled, we need to create a cluster. For that open cloud shell, you will find an icon on the top right side for Activate Cloud Shell.



Google Cloud Platform rajatcp

Kubernetes Engine API  
Google Enterprise API

Builds and manages container-based applications, powered by the open source Kubernetes technology.

ENABLE TRY THIS API

Click to enable this API

OVERVIEW DOCUMENTATION

Overview

Builds and manages container-based applications, powered by the open source Kubernetes technology.

Additional details

Type: [SaaS & APIs](#)

Now to create a cluster, enter the following command in the cloud shell :

```
gcloud container clusters create project-kube --zone "us-west1-b" --  
machine-type "n1-standard-1" --num-nodes "1"
```

```
Google Cloud Platform rajatcp Search Products, resources, docs (/)
CLOUD SHELL Terminal (rajatcp) x +
Welcome to Cloud Shell! Type "help" to get started.
To set your Cloud Platform project in this session use "gcloud config set project [PROJECT_ID]"
rajatpaliwal03@cloudshell:~$ ^C
rajatpaliwal03@cloudshell:~$ gcloud config set project rajatcp
Updated property [core/project].
rajatpaliwal03@cloudshell:~$ (rajatcp)$ git clone https://github.com/rajatpaliwal03/GCP-devops-project.git
Cloning into 'GCP-devops-project'...
Username for 'https://github.com': rajatpaliwal03
Password for 'https://rajatpaliwal03@github.com':
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: Authentication failed for 'https://github.com/rajatpaliwal03/GCP-devops-project.git/'
rajatpaliwal03@cloudshell:~$ (rajatcp)$ git clone https://github.com/rajatpaliwal03/GCP-devops-project
Cloning into 'GCP-devops-project'...
Username for 'https://github.com': rajatpaliwal03
Password for 'https://rajatpaliwal03@github.com':
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: Authentication failed for 'https://github.com/rajatpaliwal03/GCP-devops-project/'
rajatpaliwal03@cloudshell:~$ (rajatcp)$ git clone https://github.com/rajatpaliwal03/GCP-devops-project.git
Cloning into 'GCP-devops-project'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 15 (delta 1), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (15/15), 146.66 KiB | 658.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
rajatpaliwal03@cloudshell:~$ (rajatcp)$ gcloud container clusters create project-kube --zone "us-west1-b" --machine-type "n1-standard-1" --num-nodes "1"
```

It will create a Kubernetes cluster, 'project-kube' with 1 compute node of n1-standard-1 machine type.

```
rajatpaliwal03@cloudshell:~$ (rajatcp)$ gcloud container clusters create project-kube --zone "us-west1-b" --machine-type "n1-standard-1" --num-nodes "1"
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, please pass the '--no-enable-ip-alias' flag
Note: Your Pod address range ('--cluster-ip-v4-cidr') can accommodate at most 1008 node(s).
Creating cluster project-kube in us-west1-b... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/rajatcp/zones/us-west1-b/clusters/project-kube].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/_gcloud/us-west1-b/project-kube?project=rajatcp
kubeconfig entry generated for project-kube.
NAME: project-kube
LOCATION: us-west1-b
MASTER VERSION: 1.21.6-gke.1500
MASTER IP: 35.227.158.194
MACHINE TYPE: n1-standard-1
NODE VERSION: 1.21.6-gke.1500
NUM NODES: 1
STATUS: RUNNING
rajatpaliwal03@cloudshell:~$ (rajatcp)$
```


Now we have to create the following configuration files for our pipeline:

**Dockerfile:** A simple file that consists of instructions to build a Docker Image. Each instruction in a docker file is a command/operation, for example, what operating system to use, what dependencies to install or how to compile the code, and many such instructions which act as a layer.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal displays a Dockerfile script with 18 lines of code. The code includes comments and instructions for building a Docker image based on Python 3.8.5, installing dependencies, and running a Streamlit application.

```
1  # python
2  FROM python:3.8.5
3
4  RUN apt-get update
5
6  # Copy local code to the container image.
7  ENV APP_HOME /app
8  WORKDIR $APP_HOME
9  COPY . ./
10
11 RUN ls -la $APP_HOME/
12
13 # Install dependencies
14 RUN pip install -r requirements.txt
15
16 # Run the streamlit on container startup
17 CMD [ "streamlit", "run", "--server.enableCORS", "false", "streamlit_ui.py" ]
18
```

**Deployment YAML:** To run an application we need to create a Deployment object and we can do that using a YAML




```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: imgclass
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: imageclassifier
10   template:
11     metadata:
12       labels:
13         app: imageclassifier
14     spec:
15       containers:
16       - name: cv-app
17         image: gcr.io/rajatcp/streamlit-ui:v1
18         ports:
19         - containerPort: 8501
20
```

- apiVersion: Version of Kubernetes API
- kind: Kind of object to create, here its Deployment
- metadata: Data about objects to identify them
- spec: Specifications of the object, it includes replicas(no. of pods), labels, container image



**Service YAML:** To expose an application running on a set of Pods as a network service we need a Service YAML file.



```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: imageclassifier
5  spec:
6    type: LoadBalancer
7    selector:
8      app: imageclassifier
9    ports:
10     - port: 80
11       targetPort: 8501
```

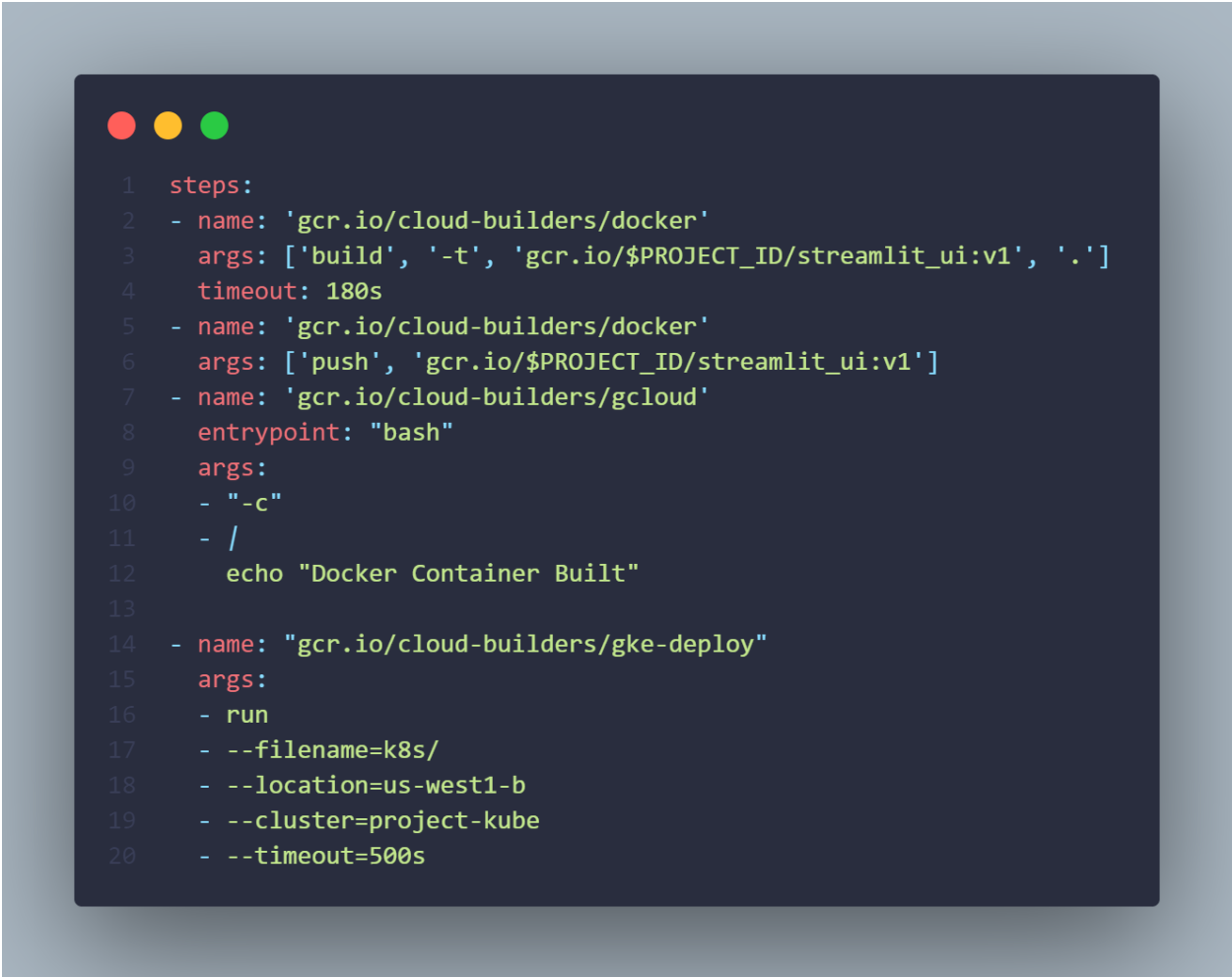
In this file, we have specified kind: Service, in spec type: LoadBalancer to automatically distribute the load, and app name is same as in Deployment YAML file. It has port mapping which targets container port 8501.

## CloudBuild

Cloud Build is a service that executes your builds on Google Cloud Platform's infrastructure.

Cloud Build can import source code from a variety of repositories or cloud storage spaces, execute a build to your specifications, and produce artifacts such as Docker containers or Java archives.

It executes the commands in steps and is similar to executing commands in a script.



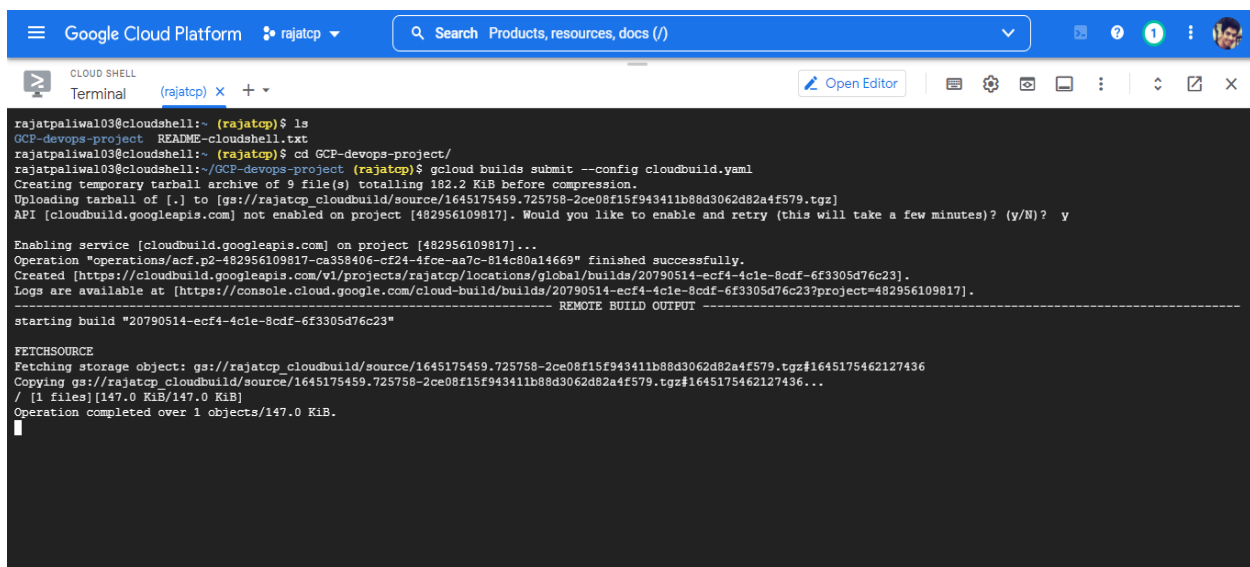
```
1  steps:
2  - name: 'gcr.io/cloud-builders/docker'
3    args: ['build', '-t', 'gcr.io/$PROJECT_ID/streamlit_ui:v1', '.']
4    timeout: 180s
5  - name: 'gcr.io/cloud-builders/docker'
6    args: ['push', 'gcr.io/$PROJECT_ID/streamlit_ui:v1']
7  - name: 'gcr.io/cloud-builders/gcloud'
8    entrypoint: "bash"
9    args:
10   - "-c"
11   - /
12     echo "Docker Container Built"
13
14  - name: "gcr.io/cloud-builders/gke-deploy"
15    args:
16   - run
17   - --filename=k8s/
18   - --location=us-west1-b
19   - --cluster=project-kube
20   - --timeout=500s
```

In the first step, we will build the Docker image and in the next step, we will push the image to Google Container Registry. The final step is to deploy the application on the Kubernetes cluster, filename is the folder directory that will have

Deployment and Service YAML files, specify the image and cluster name that we have created earlier.

## KickStart Pipeline:

```
gcloud builds submit -config cloudbuild.yaml
```

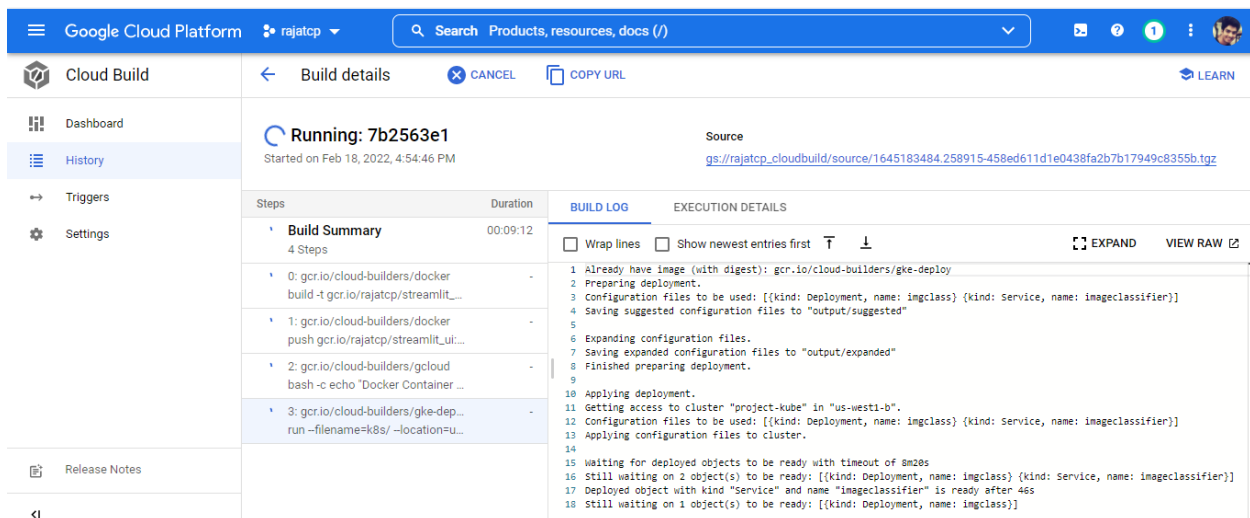


```
rajatpaliwal03@cloudshell:~ (rajatcp) $ ls
GCP-devops-project README-cloudshell.txt
rajatpaliwal03@cloudshell:~ (rajatcp) $ cd GCP-devops-project/
rajatpaliwal03@cloudshell:~/GCP-devops-project (rajatcp) $ gcloud builds submit --config cloudbuild.yaml
Creating temporary tarball archive of 9 file(s) totalling 182.2 KiB before compression.
Uploading tarball of [.] to [gs://rajatcp-cloudbuild/source/1645175459.725758-2ce08f15f943411b88d3062d82a4f579.tgz]
API [cloudbuild.googleapis.com] not enabled on project [482956109817]. Would you like to enable and retry (this will take a few minutes)? (y/N)? y

Enabling service [cloudbuild.googleapis.com] on project [482956109817]...
Operation "operations/acf.p2-482956109817-ca358406-cf24-4fce-aa7c-814c80a14669" finished successfully.
Created [https://cloudbuild.googleapis.com/v1/projects/rajatcp/locations/global/builds/20790514-ecf4-4c1e-8cdf-6f3305d76c23].
Logs are available at [https://console.cloud.google.com/cloud-build/builds/20790514-ecf4-4c1e-8cdf-6f3305d76c23?project=482956109817].
----- REMOTE BUILD OUTPUT -----
starting build "20790514-ecf4-4c1e-8cdf-6f3305d76c23"

FETCHSOURCE
Fetching storage object: gs://rajatcp-cloudbuild/source/1645175459.725758-2ce08f15f943411b88d3062d82a4f579.tgz#1645175462127436
Copying gs://rajatcp-cloudbuild/source/1645175459.725758-2ce08f15f943411b88d3062d82a4f579.tgz#1645175462127436...
/ [1 files][147.0 KiB/147.0 KiB]
Operation completed over 1 objects/147.0 KiB.
```

## Building start:



Steps		Duration	BUILD LOG	EXECUTION DETAILS
<b>Build Summary</b> 4 Steps		00:09:12		
0: gcr.io/cloud-builders/docker build -t gcr.io/rajatcp/streamlit...	-		1 Already have image (with digest): gcr.io/cloud-builders/gke-deploy 2 Preparing deployment. 3 Configuration files to be used: [{"kind: Deployment, name: imgclass} {"kind: Service, name: imageclassifier}] 4 Saving suggested configuration files to "output/suggested" 5 6 Expanding configuration files. 7 Saving expanded configuration files to "output/expanded" 8 Finished preparing deployment. 9	
1: gcr.io/cloud-builders/docker push gcr.io/rajatcp/streamlit...	-		10 Applying deployment. 11 Getting access to cluster "project-kube" in "us-west1-b". 12 Configuration files to be used: [{"kind: Deployment, name: imgclass} {"kind: Service, name: imageclassifier}] 13 Applying configuration files to cluster. 14	
2: gcr.io/cloud-builders/gcloud bash -c echo "Docker Container ..."	-		15 Waiting for deployed objects to be ready with timeout of 8m20s 16 Still waiting on 2 object(s) to be ready: [{"kind: Deployment, name: imgclass} {"kind: Service, name: imageclassifier}] 17 Deployed object with kind "Service" and name "imageclassifier" is ready after 46s 18 Still waiting on 1 object(s) to be ready: [{"kind: Deployment, name: imgclass}]	
3: gcr.io/cloud-builders/gke-dep... run --filename=k8s/ --location=u...	-			

## Building successful:

Google Cloud Platform

rajatcp

Search Products, resources, docs (/)

Cloud Build

Build details

REBUILD

COPY URL

LEARN

Dashboard

History

Triggers

Settings

Successful: 7b2563e1

Started on Feb 18, 2022, 4:54:46 PM

Source

[gs://rajatcp\\_cloudbuild/source/1645183484.258915-458ed611d1e0438fa2b7b17949c8355b.tgz](gs://rajatcp_cloudbuild/source/1645183484.258915-458ed611d1e0438fa2b7b17949c8355b.tgz)

Steps	Duration	BUILD LOG	EXECUTION DETAILS
<div>Build Summary</div> <div>4 Steps</div>	00:09:41	<div><input type="checkbox"/> Wrap lines <input type="checkbox"/> Show newest entries first</div> <div><div>EXPAND</div><div>VIEW RAW</div></div>	
<div>0: gcr.io/cloud-builders/docker build -t gcr.io/rajatcp/streamlit...</div>	00:02:28		
<div>1: gcr.io/cloud-builders/docker push gcr.io/rajatcp/streamlit-ui...</div>	00:04:58		
<div>2: gcr.io/cloud-builders/gcloud bash -c echo "Docker Container ..."</div>	00:00:00		
<div>3: gcr.io/cloud-builders/gke-dep... run --filename=k8s/ --location=u...</div>	00:02:06		

1 Already have image (with digest): gcr.io/cloud-builders/gke-deploy

2 Preparing deployment.

3 Configuration files to be used: [{"kind: Deployment, name: imgclass"} {"kind: Service, name: imageclassifier"}]

4 Saving suggested configuration files to "output/suggested"

5

6 Expanding configuration files.

7 Saving expanded configuration files to "output/expanded"

8 Finished preparing deployment.

9

10 Applying deployment.

11 Getting access to cluster "project-kube" in "us-west1-b".

12 Configuration files to be used: [{"kind: Deployment, name: imgclass"} {"kind: Service, name: imageclassifier"}]

13 Applying configuration files to cluster.

14

15 Waiting for deployed objects to be ready with timeout of 8m20s

16 Still waiting on 2 object(s) to be ready: [{"kind: Deployment, name: imgclass"} {"kind: Service, name: imageclassifier"}]

17 Deployed object with kind "Service" and name "imageclassifier" is ready after 46s

18 Still waiting on 1 object(s) to be ready: [{"kind: Deployment, name: imgclass"}]

19 Still waiting on 1 object(s) to be ready: [{"kind: Deployment, name: imgclass"}]

20

21

22

23

24

25

26

27

28

29

30

31

32

33 Workloads:

34 Services & Ingress:

35 Applications:

36 Configuration:

37 Storage:

Google Cloud Platform

rajatcp

Search Products, resources, docs (/)

Cloud Build

Build details

REBUILD

COPY URL

LEARN

Dashboard

History

Triggers

Settings

Successful: 7b2563e1

Started on Feb 18, 2022, 4:54:46 PM

Source

[gs://rajatcp\\_cloudbuild/source/1645183484.258915-458ed611d1e0438fa2b7b17949c8355b.tgz](gs://rajatcp_cloudbuild/source/1645183484.258915-458ed611d1e0438fa2b7b17949c8355b.tgz)

Steps	Duration	BUILD LOG	EXECUTION DETAILS
<div>Build Summary</div> <div>4 Steps</div>	00:09:41	<div><input type="checkbox"/> Wrap lines <input type="checkbox"/> Show newest entries first</div> <div><div>EXPAND</div><div>VIEW RAW</div></div>	
<div>0: gcr.io/cloud-builders/docker build -t gcr.io/rajatcp/streamlit...</div>	00:02:28		
<div>1: gcr.io/cloud-builders/docker push gcr.io/rajatcp/streamlit-ui...</div>	00:04:58		
<div>2: gcr.io/cloud-builders/gcloud bash -c echo "Docker Container ..."</div>	00:00:00		
<div>3: gcr.io/cloud-builders/gke-dep... run --filename=k8s/ --location=u...</div>	00:02:06		

20 Deployed object with kind "Deployment" and name "imgclass" is ready after 1m55.5s

21 Finished applying deployment.

22

23

24 > Deployed Objects

25

26

27

28

29

30

31

32

33 Workloads:

34 Services & Ingress:

35 Applications:

36 Configuration:

37 Storage:

```
Google Cloud Platform rajatcp Search cloud

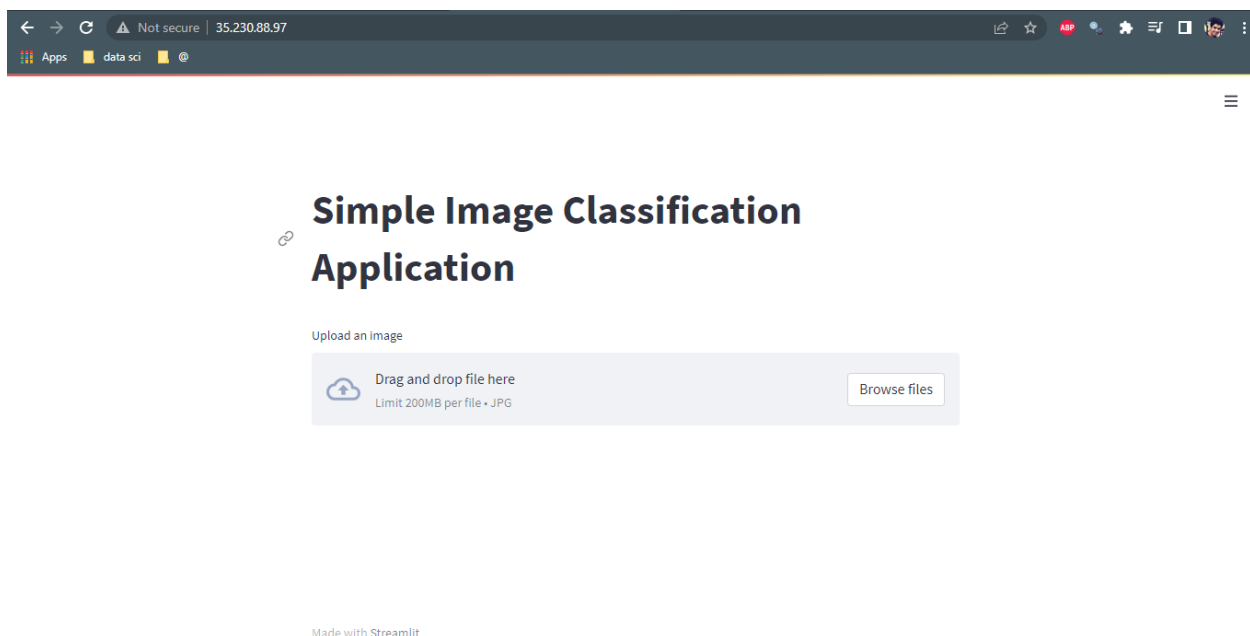
CLOUD SHELL Terminal (rajatcp) + - Open Editor

Step #3: Deployed object with kind "Deployment" and name "imgclass" is ready after 1m55.5s
Step #3: Finished applying deployment.
Step #3: #####
Step #3: > Deployed Objects
Step #3:
Step #3: NAMESPACE   KIND       NAME           READY
Step #3: default      Deployment  imgclass       Yes
Step #3: default      Service    imageclassifier Yes    http://35.230.88.97
Step #3: #####
Step #3: > GKE
Step #3:
Step #3: Workloads:      https://console.cloud.google.com/kubernetes/workload?project=rajatcp
Step #3: Services & Ingress: https://console.cloud.google.com/kubernetes/discovery?project=rajatcp
Step #3: Applications:   https://console.cloud.google.com/kubernetes/application?project=rajatcp
Step #3: Configurations: https://console.cloud.google.com/kubernetes/config?project=rajatcp
Step #3: Storage:       https://console.cloud.google.com/kubernetes/storage?project=rajatcp
Step #3:
Finished Step #3
PUSH
DONE

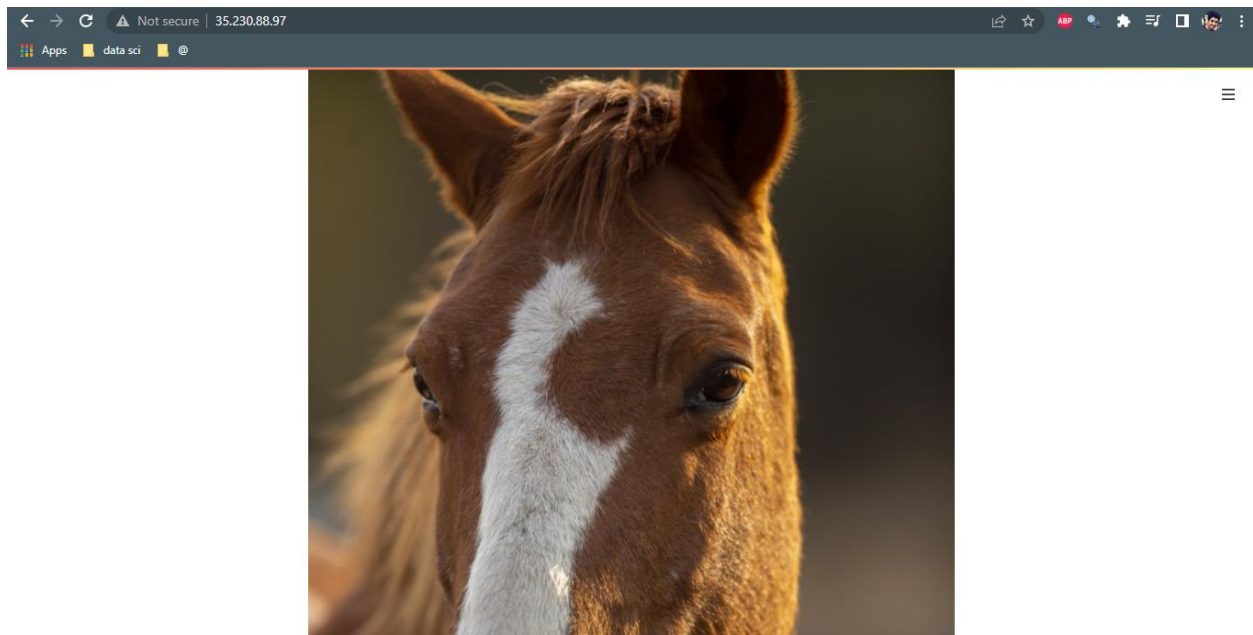
ID: 7b2563e1-0b6f-4775-a596-38c07cf34f9d
CREATE TIME: 2022-02-18T11:24:46+00:00
DURATION: 9M41S
SOURCE: gs://rajatcp_cloudbuild/source/1645183484.258915-458ed611d1e0438fa2b7b17949c8355b.tgz
IMAGES: -
STATUS: SUCCESS
rajatpaliwal03@cloudshell:~/GCP-devops-project (rajatcp) $
```

```
ID: 7b2563e1-0b6f-4775-a596-38c07cf34f9d
CREATE TIME: 2022-02-18T11:24:46+00:00
DURATION: 9M41S
SOURCE: gs://rajatcp_cloudbuild/source/1645183484.258915-458ed611d1e0438fa2b7b17949c8355b.tgz
IMAGES: -
STATUS: SUCCESS
rajatpaliwal03@cloudshell:~/GCP-devops-project (rajatcp) $ kubectl get svc
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
imageclassifier      LoadBalancer 10.20.1.208    35.230.88.97   80:31786/TCP     5m54s
kubernetes           ClusterIP    10.20.0.1     <none>         443/TCP          159m
rajatpaliwal03@cloudshell:~/GCP-devops-project (rajatcp) $
```

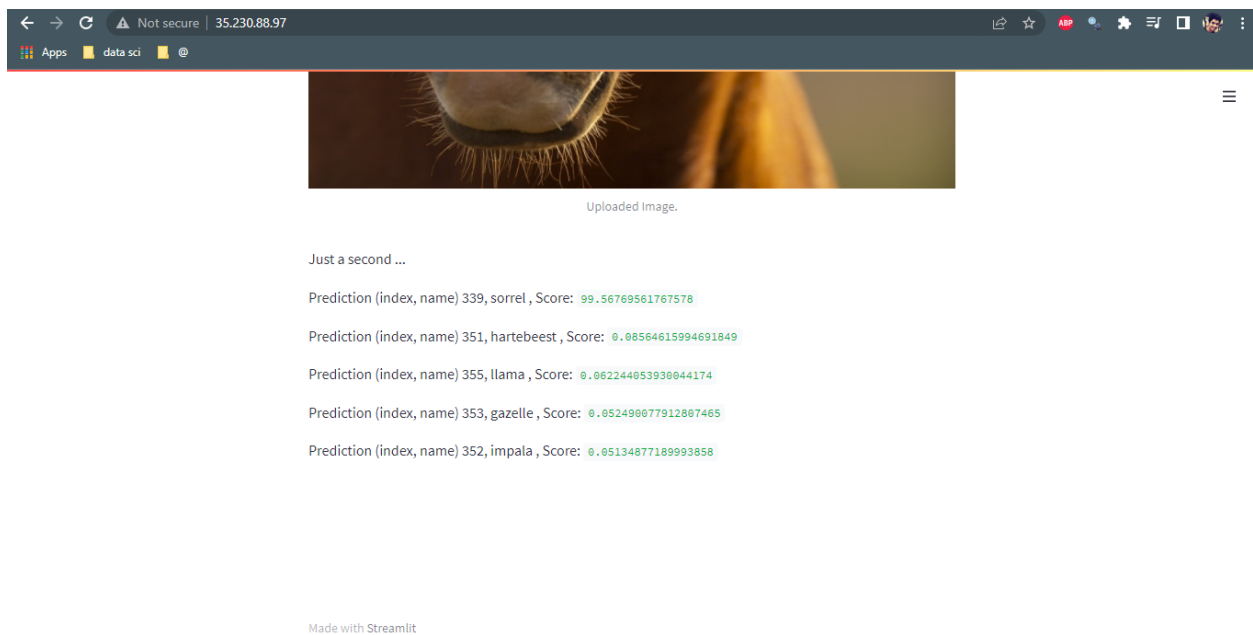
Now we can go to the endpoint and check our application.



Let's try to input any image in that:

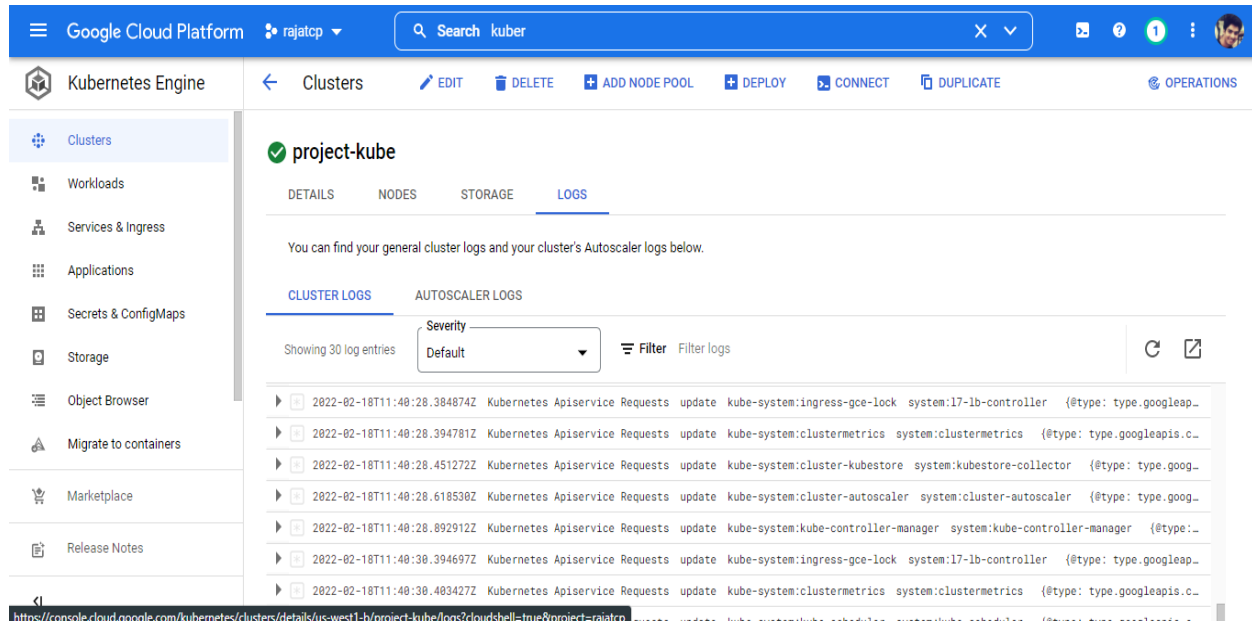


## Results:



Top result give us the horse type as “sorrel” with matching probability of 99.56%.

## For checking cluster logs :



## Monitoring:

To add an email notification channel, do the following:

- 1) In the Cloud Console, select **Monitoring**
- 2) Click **Alerting** and then click **Edit notification channels**.
- 3) In the **Email** section, click **Add new**.
- 4) Complete the dialog and click **Save**.

## Step 1:

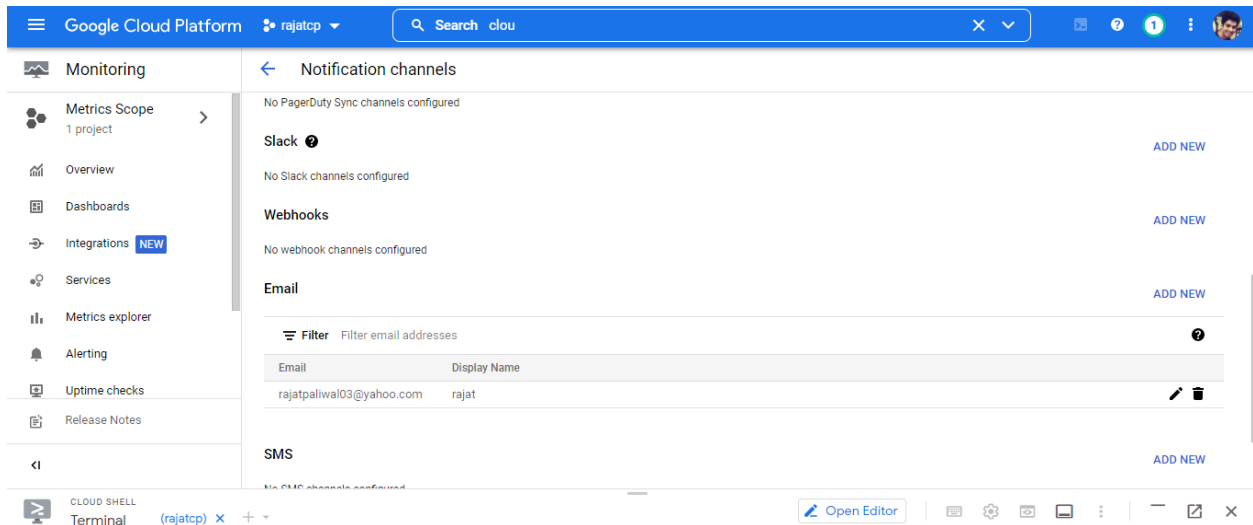
The screenshot shows the Google Cloud Platform Monitoring Overview page. The left sidebar contains navigation links: Overview, Dashboards, Integrations, Services, Metrics explorer, Alerting (selected), Uptime checks, Groups, and Release Notes. The main content area is titled 'Overview' and features a progress bar for 'GET STARTED WITH MONITORING' at 25% complete. Below the progress bar, there are four steps: 'Integrate with Google Cloud services' (selected), 'Install an agent' (checked), 'Create a dashboard', and 'Create an alert'. The 'Google Cloud integration' section provides information about discovering and monitoring Google Cloud resources. The right sidebar contains 'Recommended for you' links: 'Introduction to Cloud Monitoring', 'Metrics, time series, and resources', and 'Google Cloud operations suite agents'. At the bottom, there is a 'CLOUD SHELL' terminal window and an 'Open Editor' button.

## Step 2:

The screenshot shows the Google Cloud Platform Monitoring Notification channels page. The left sidebar contains navigation links: Metrics Scope (1 project), Overview, Dashboards, Integrations (NEW), Services, Metrics explorer, Alerting, Uptime checks, and Release Notes. The main content area is titled 'Notification channels' and lists various notification methods: Slack, Webhooks, Email, SMS, and Cloud Pub/Sub. Each method has a status indicator (e.g., 'No PagerDuty Sync channels configured') and an 'ADD NEW' button. The right sidebar contains 'You might also like' links: 'Concepts' and 'Tutorials'. At the bottom, there is a 'CLOUD SHELL' terminal window and an 'Open Editor' button.

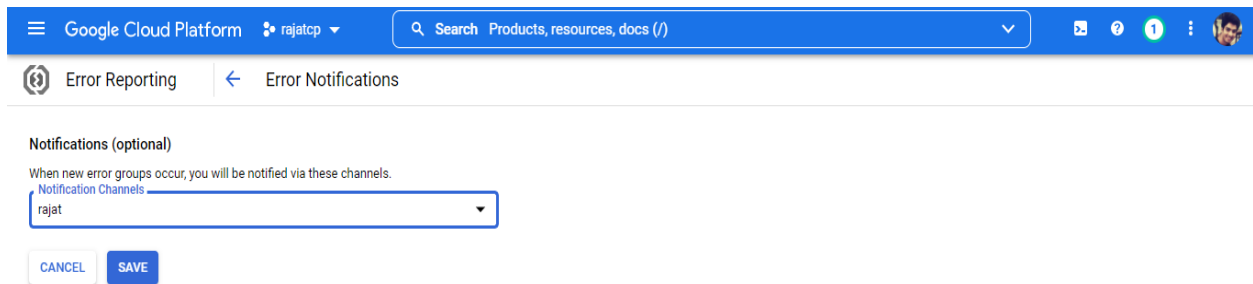


### Step 3:



The screenshot shows the Google Cloud Platform interface for the 'rajatcp' project. The left sidebar contains the 'Monitoring' menu with options like Metrics Scope, Overview, Dashboards, Integrations (marked as NEW), Services, Metrics explorer, Alerting, Uptime checks, and Release Notes. The main content area is titled 'Notification channels' and lists four categories: PagerDuty Sync, Slack, Webhooks, and Email. Each category has an 'ADD NEW' link. The 'Email' section is expanded, showing a table with one entry: 'rajatpaliwal03@yahoo.com' with the display name 'rajat'. A filter bar is visible above the table. At the bottom, there is a 'CLOUD SHELL' terminal window and an 'Open Editor' button.

### Step 4:



The screenshot shows the 'Error Reporting' section of the Google Cloud Platform interface. The 'Error Notifications' tab is selected. Under the heading 'Notifications (optional)', there is a message: 'When new error groups occur, you will be notified via these channels.' Below this is a dropdown menu labeled 'Notification Channels' with 'rajat' selected. At the bottom of the dialog are two buttons: 'CANCEL' and 'SAVE'.

Complete the dialog and click **Save**.

