

## Table of Contents

7.1a: Terraform AWS Guestbook.....	3
7.1.1 Terraform .....	3
7.1.2 Setup .....	3
7.1.3 Initial configuration.....	3
7.1.4 Launching configuration .....	3
7.1.5 Adding network access .....	4
7.1.6 Adding ssh access.....	4
7.1.7 Adding the Guestbook application .....	4
7.1.8 View the Guestbook .....	5
7.1.9 Clean Up .....	5
7.1g: Terraform GCP Guestbook .....	5
7.1.1 Terraform .....	6
7.1.2 Setup .....	6
7.1.3 Initial configuration.....	6
7.1.4 Launching configuration .....	6
7.1.5 Adding an external IP address.....	6
7.1.6 Adding ssh access.....	7
7.1.7 Adding the Guestbook application .....	8
7.1.8 View the Guestbook .....	9
7.1.9 Clean Up .....	10
7.2: Kubernetes Guestbook .....	10
7.2.1 Kubernetes .....	10
7.2.2 Setup .....	10
7.2.3 Assigning privileges .....	10
7.2.4 Create Kubernetes cluster .....	10
7.2.5 Prepare a container image .....	12
7.2.6 Kubernetes.yaml .....	12
7.2.7 Deploy the configuration .....	12

7.2.8 View the Guestbook .....	12
7.2.9 Delete workload and service .....	14
7.2.10 CI/CD build automation .....	14
7.2.11 Configure build automation.....	14
7.2.12 Deploy and view application.....	14
7.2.13 Clean up .....	15
7.3: APIs (Slack, Knowledge Graph) .....	15
7.3.1 Slack and Knowledge Graph integration .....	15
7.3.2 Code .....	15
7.3.3 Code .....	16
7.3.4 Knowledge Graph setup .....	17
7.3.5 Create a Slack workspace .....	17
7.3.6 Configure and Deploy .....	17
7.3.7 Create Slack command .....	17
7.3.8 Test the command.....	17
7.4: ML APIs.....	17
7.4.1 APIs #1 (Vision, Speech, Translate, Natural Language APIs) .....	17
7.4.2 IAM service account setup .....	17
7.4.3 Vision .....	17
7.4.4 Speech .....	20
7.4.5 Translate.....	21
7.4.6 Natural Language .....	22
7.4.7 Integration .....	23
7.4.8 Code .....	23
7.4.9 Test integration.....	25
7.4.10 APIs #2 (Video Intelligence API) .....	27
7.4.11 Video setup.....	27
7.4.12 Video Intelligence labeling script .....	27
7.4.13. Video Intelligence .....	27

7.4.14 APIs #3 (Web site integration) .....	28
7.4.15 IAM service account setup .....	28
7.4.16 Application.....	28
7.4.17 Code .....	28
7.4.18 Clean up .....	30

## 7.1a: Terraform AWS Guestbook

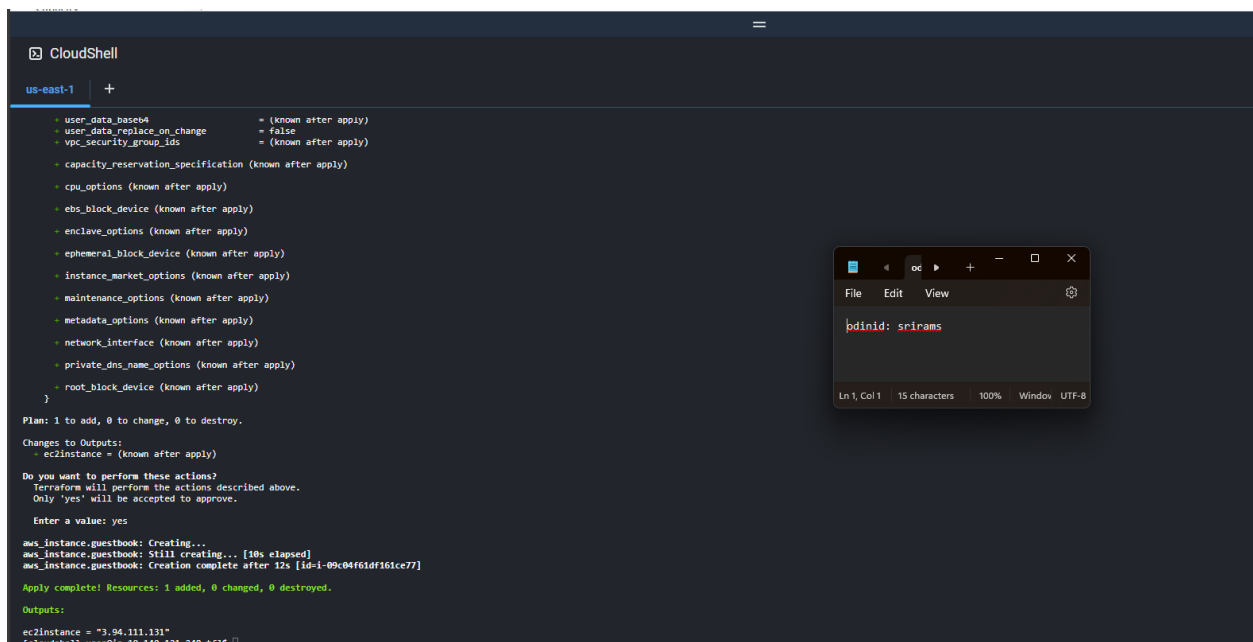
### 7.1.1 Terraform

### 7.1.2 Setup

### 7.1.3 Initial configuration

### 7.1.4 Launching configuration

- Take a screenshot showing the completion of the command including its output



```

CloudShell
us-east-1 +

user_data_base64      = (known after apply)
user_data_replace_on_change = false
vpc_security_group_ids = (known after apply)

capacity_reservation_specification (known after apply)

cpu_options (known after apply)

ebs_block_device (known after apply)

enclave_options (known after apply)

ephemeral_block_device (known after apply)

instance_market_options (known after apply)

maintenance_options (known after apply)

metadata_options (known after apply)

network_interface (known after apply)

private_dns_name_options (known after apply)

root_block_device (known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  ec2instance = (known after apply)

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

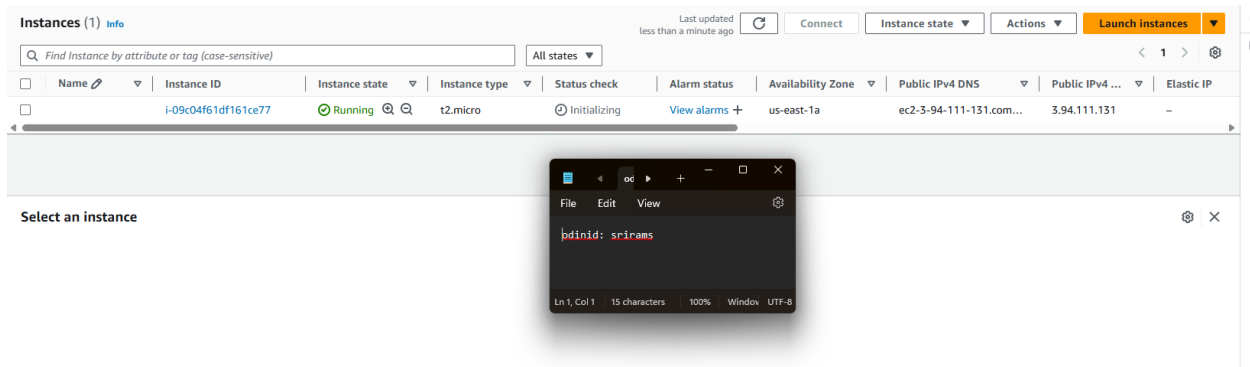
aws_instance.guestbook: Creating...
aws_instance.guestbook: Still creating... [10s elapsed]
aws_instance.guestbook: Creation complete after 12s [id=i-09c04f61df161ce77]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:
ec2instance = "3.94.111.131"
[cloudshell-user@ip-10-140-121-248 tf]$

```

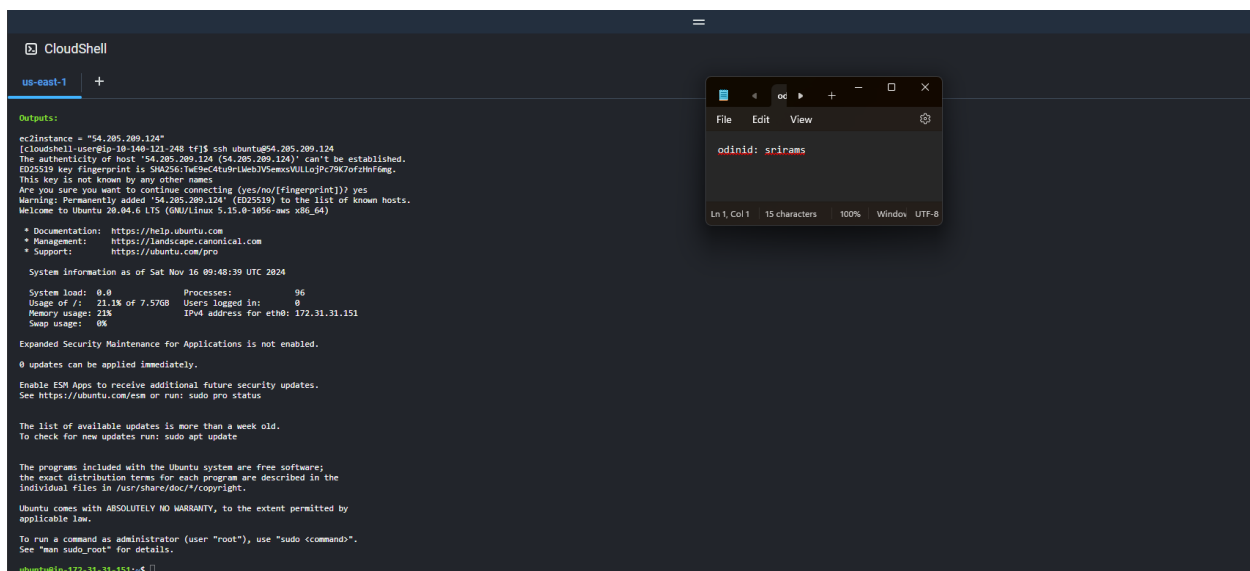
- Take a screenshot that includes the VM's IP addresses



## 7.1.5 Adding network access

## 7.1.6 Adding ssh access

- Take a screenshot of the successful ssh login from Cloud Shell.



## 7.1.7 Adding the Guestbook application

- Take a screenshot of the output of the command that includes the IP address of the instance

```
CloudShell
us-east-1 +
}
{
  cidr_blocks = [
    "0.0.0.0/0",
  ]
  from_port = 80
  to_port = 80
  protocol = "tcp"
  security_groups = [
    sg-0b0e0c9f,
  ]
  self = false
  to_port = 80
  # (s unchanged attribute hidden)
}
}
name = "Guestbook-50"
name_prefix = (known after apply)
owner_id = (known after apply)
remove_rules_on_delete = false
tags_all = (known after apply)
vpc_id = (known after apply)
}

Plan: 3 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  ec2instance = (known after apply)

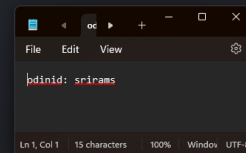
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_key_pair.kp: Creating...
aws_security_group.sg.guestbook: Creating...
aws_key_pair.kp: Creation complete after 0s [id=guestbook-key]
aws_security_group.sg.guestbook: Creation complete after 2s [id=sg-0b0e0c9f]
aws_instance.guestbook: Creating...
aws_instance.guestbook: Still creating... [10s elapsed]
aws_instance.guestbook: Creation complete after 13s [id=i-0c185390f0683ca97]

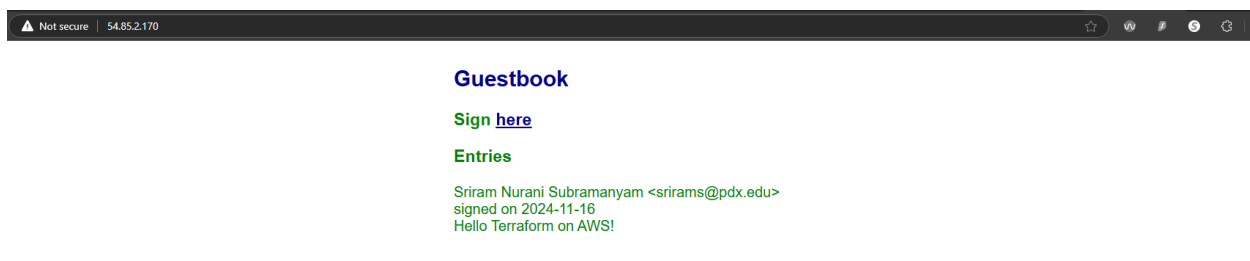
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:
ec2instance = "54.85.2.170"
```



## 7.1.8 View the Guestbook

- Take a screenshot of the Guestbook including the URL with the entry in it.



## 7.1.9 Clean Up

## 7.1g: Terraform GCP Guestbook

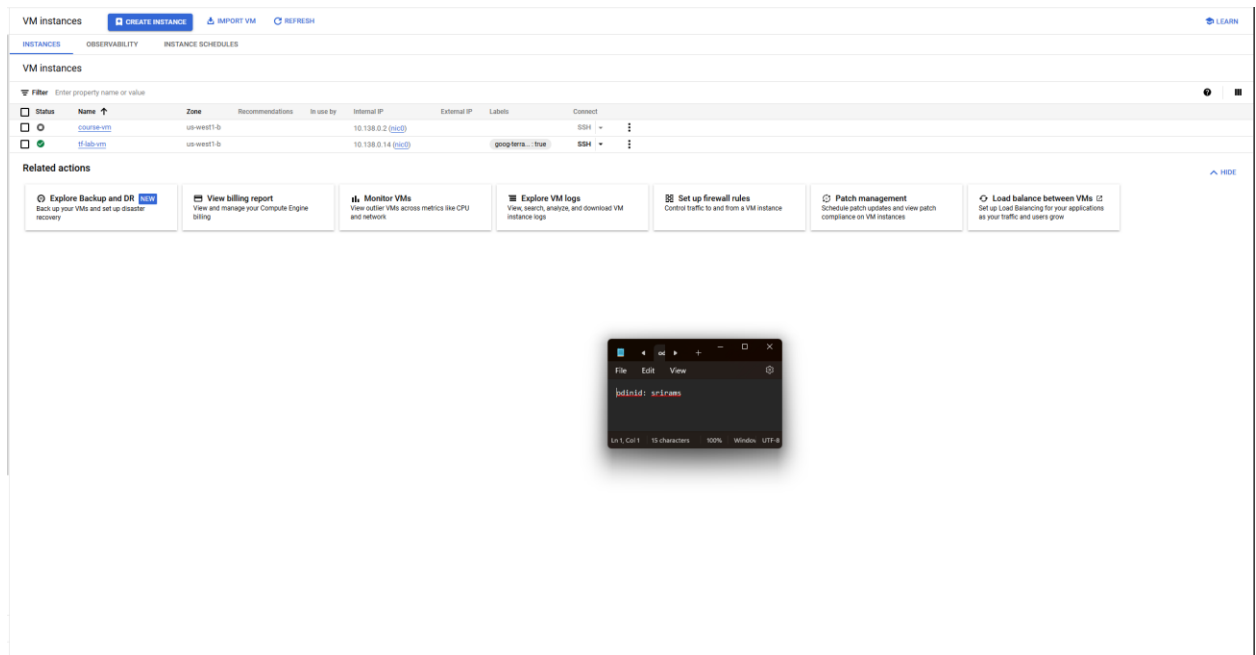
## 7.1.1 Terraform

## 7.1.2 Setup

## 7.1.3 Initial configuration

## 7.1.4 Launching configuration

- Take a screenshot that includes the VM's IP addresses



## 7.1.5 Adding an external IP address

- Take a screenshot showing the completion of the command including its output

```

offensive_label = {
    "group-terraform-provider-name" = "true"
}

id = (known after apply)
label_fingerprint = none
name = (known after apply)
network_tier = (known after apply)
profile_image = (known after apply)
project = "cloud-opensource"
region = (known after apply)
self_link = (known after apply)
self_link_short = (known after apply)
terraform_label = {
    "group-terraform-provider-name" = "true"
}

name = (known after apply)

# google_compute_instance_default will be updated in-place
resource "google_compute_instance" "terraform" {
  id = "projects/cloud-opensource-vm-images/zone/us-west-1/instances/tf-lab-vm"
  name = "TF-Lab-vm"
  tags = ["TF-Lab-vm"]

  # [OPTIONAL] Network interface
  network_interface {
    name = "eth0"

    # [OPTIONAL] Network address
    address_aliases {
      nat_ip = (known after apply)
    }
  }
}

Plan: 1 to add, 1 to change, 0 to destroy.
```

Change in Output:

```

id = (known after apply)
```

So you want to perform these actions?

Therefore will perform the actions described above.  
 Only "yes" will be accepted to agree.

Enter a value: yes

```

google_compute_addresses.static: Creating...
google_compute_addresses.static: Still creating... [10s elapsed]
google_compute_addresses.static: Creation complete after 11s [10s-joint/cloud-opensource-vm-images/zone/us-west-1/addresses/tf-lab-vm]
google_compute_instance.terraform: Modifying... [10s elapsed]
google_compute_instance.terraform: Modifying complete after 11s [10s-joint/cloud-opensource-vm-images/zone/us-west-1/instances/tf-lab-vm]
Apply complete! Resources: 1 added, 1 changed, 0 destroyed.
```

Outputs:

```

ip = "35.145.125.88"
external_ip_address = "35.145.125.88" [cloud-opensource-vm-images]
```

- **Take a screenshot that includes the VM's IP addresses**

INSTANCES

OBSERVABILITY

INSTANCE SCHEDULES

### VM instances

Filter

Enter property name or value

Status

Name

Zone

Recommendations

In use by

Internal IP

External IP

Labels

Connect

○

○

●

course-vm

tf-lab-vm

us-west1-b

us-west1-b

10.138.0.2

10.138.0.14

34.143.125.88

gcp-terra-...

gcp-terra-...

SSH

SSH

### Related actions

Explore Backup and DR

Back up your VMs and set up disaster recovery

View billing report

View and manage your Compute Engine billing

Monitor VMs

View outlier VMs across metrics like CPU and network

Explore VM logs

View, search, analyze, and download VM instance logs

Set up firewall rules

Control traffic to and from a VM instance

Patch management

Schedule patch updates and view patch compliance on VM instances

Load balance between VMs

Set up Load Balancing for your applications as your traffic and users grow

### 7.1.6 Adding ssh access

- **Take a screenshot of the successful ssh login from Cloud Shell.**

```

srirama@cloudshell: /tf (cloud-nirani-srirama) $ ssh 34.145.125.88
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-1071-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Mon Nov 18 08:31:19 UTC 2024

System load: 0.0          Processes: 102
Usage of /: 19.8% of 9.51GB   Users logged in: 0
Memory usage: 5%           IPv4 address for ens4: 10.138.0.14
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software:
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

srirama@tf-lab-vml:~$

```

## 7.1.7 Adding the Guestbook application

- **What resources are being added, changed, or destroyed?**

From the provided Terraform plan:


### Resources being destroyed and recreated:

- google\_compute\_instance.default will be replaced. This is indicated by the symbol -/+ (destroy and then create replacement).
- Changes include:
  - Metadata startup script is being added.
  - Other computed values like cpu\_platform, creation\_timestamp, current\_status, metadata\_fingerprint, and others are marked as (known after apply).

- **What part of the configuration forces a replacement to occur?**

The replacement is triggered because of the addition of a metadata\_startup\_script. This is indicated in the plan by:

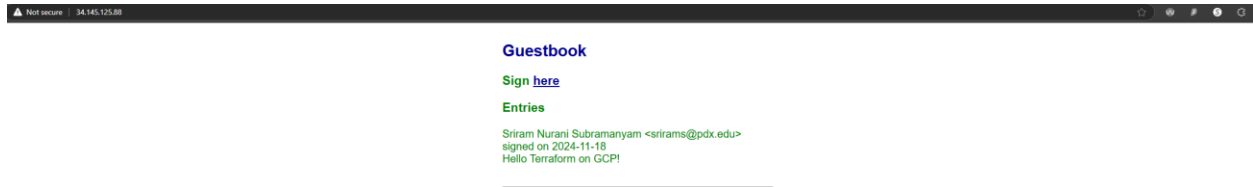




The addition of this startup script to the metadata block requires a full replacement of the `google_compute_instance.default` resource because metadata changes are not update-in-place for Compute Engine instances in GCP. Instead, Terraform destroys the instance and recreates it with the updated configuration.

### 7.1.8 View the Guestbook

- **Take a screenshot of the Guestbook including the URL with the entry in it.**



## 7.1.9 Clean Up

# 7.2: Kubernetes Guestbook

## 7.2.1 Kubernetes


## 7.2.2 Setup

## 7.2.3 Assigning privileges


## 7.2.4 Create Kubernetes cluster

- **What is the name of the Instance Template dynamically generated to create the two nodes (VMs)?**

gke-guestbook-default-pool-eee11d7f


A screenshot of a code editor window. The menu bar shows 'File', 'Edit', and 'View'. The text 'odinid: scirams' is entered in the editor area. The status bar at the bottom shows 'Ln 1, Col 1', '15 characters', '100%', 'Window', and 'UTF-8'.

- gke-guestbook-default-pool-eee11d7f-grp



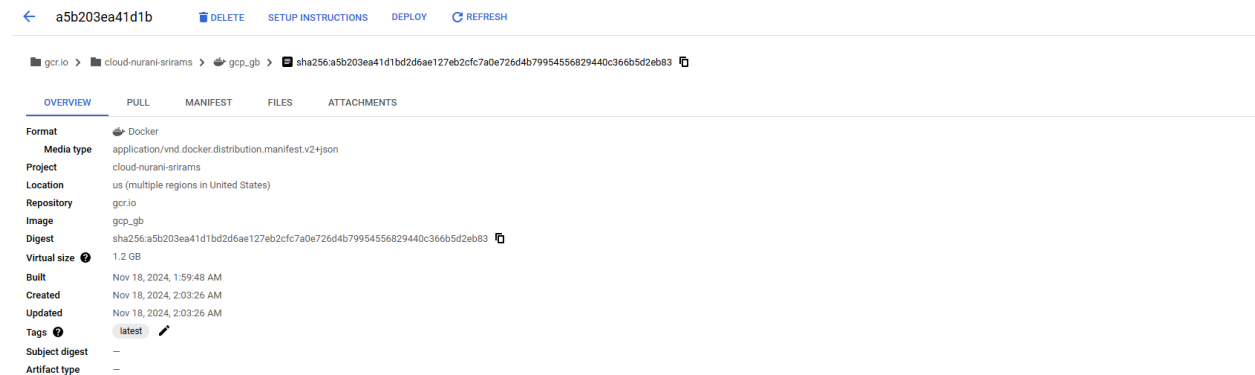
A screenshot of a Windows Notepad window. The title bar shows the Notepad icon and the text 'Notepad'. The menu bar includes 'File', 'Edit', and 'View'. The text area contains the command 'jodinid: srlrsm' in a monospaced font. The status bar at the bottom indicates 'Ln 1, Col 1', '15 characters', '100%', 'Window', and 'UTF-8'.

- gke-guestbook-default-pool-eee11d7f-vvbz

A screenshot of a Windows Notepad application window. The title bar shows the Notepad icon, the text 'od', and standard window controls (minimize, maximize, close). The menu bar includes 'File', 'Edit', and 'View'. The text area contains the string 'odinid: srxnames' in a monospaced font. The status bar at the bottom indicates 'Ln 1, Col 1', '15 characters', '100%', 'Window', and 'UTF-8'.

## 7.2.5 Prepare a container image

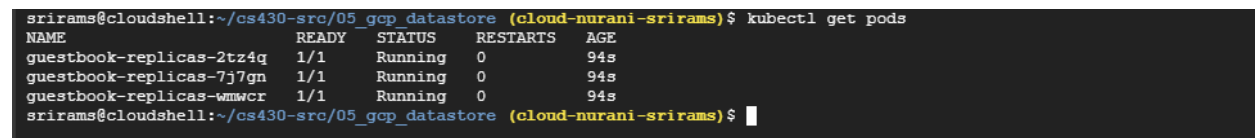
- Take a screenshot of the container image created



## 7.2.6 Kubernetes.yaml

## 7.2.7 Deploy the configuration

- Take a screenshot of the output of the following command when all 3 replicas reach a "Running" state.

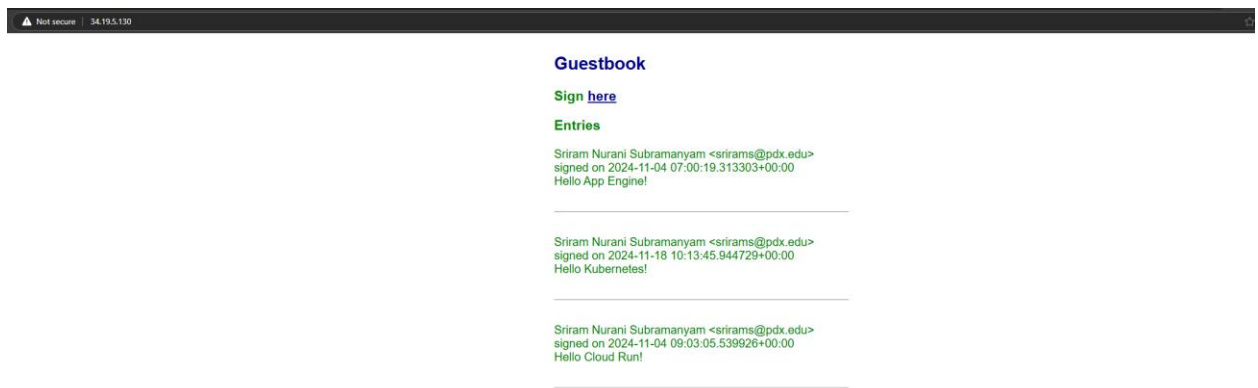


- Take a screenshot of listing services with LoadBalancer indicating an external IP address that is ready for access.

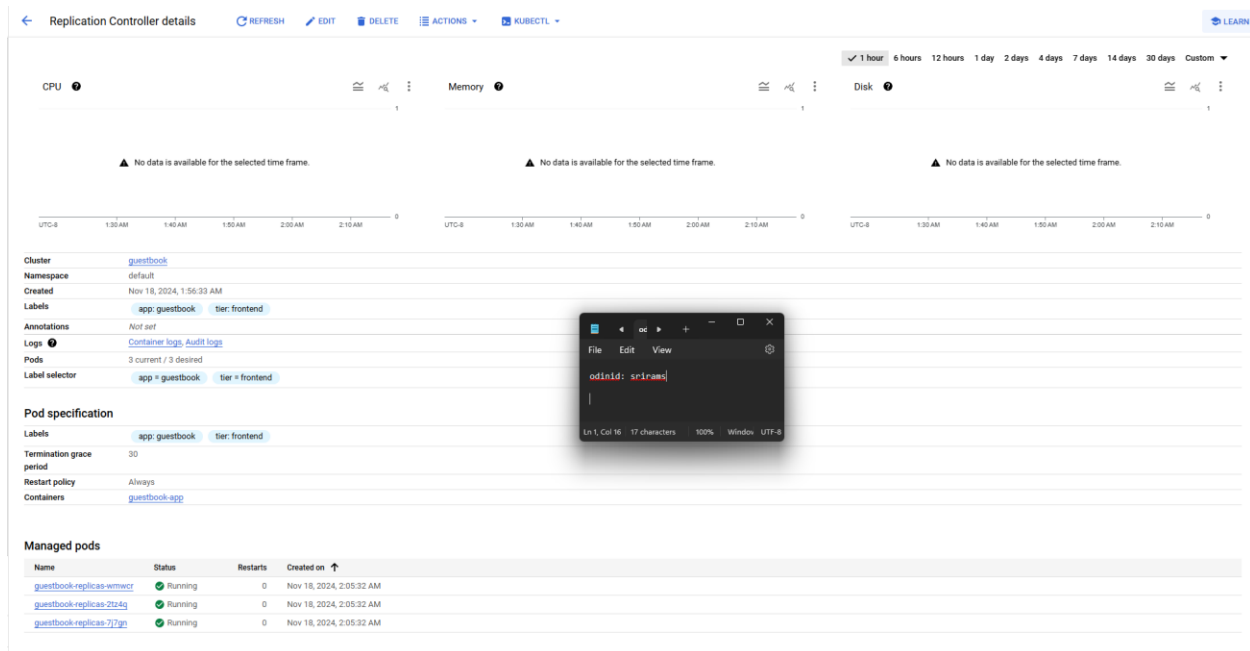
```
srirams@cloudshell:~/cs430-src/05_gcp_datastore (cloud-nurani-srirams)$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
guestbook-lb  LoadBalancer  34.118.236.74  34.19.5.130    80:30840/TCP     13m
kubernetes    ClusterIP     34.118.224.1  <none>         443/TCP          30m
srirams@cloudshell:~/cs430-src/05_gcp_datastore (cloud-nurani-srirams)$
```

## 7.2.8 View the Guestbook

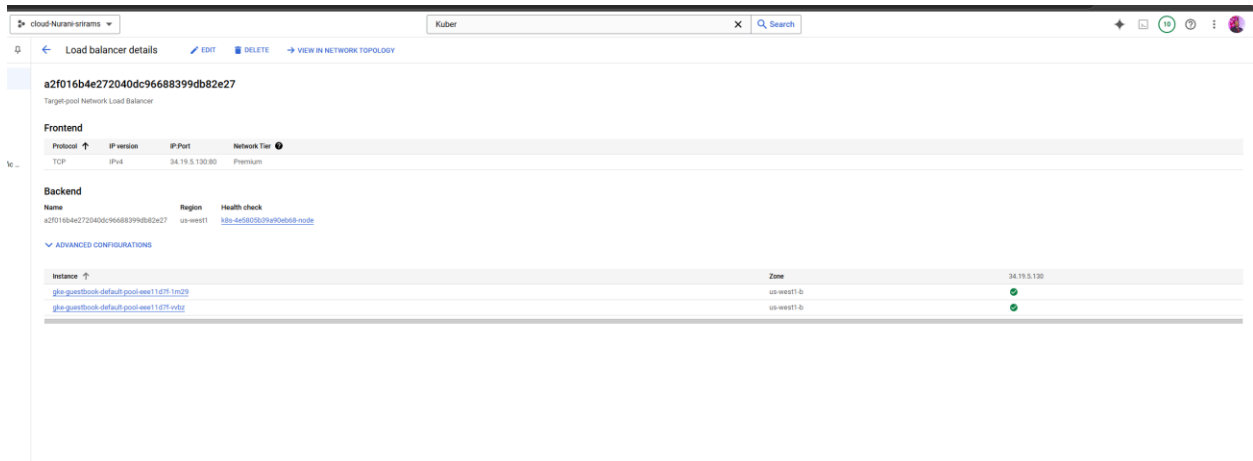
- Take a screenshot of the Guestbook including the URL with the entry in it.



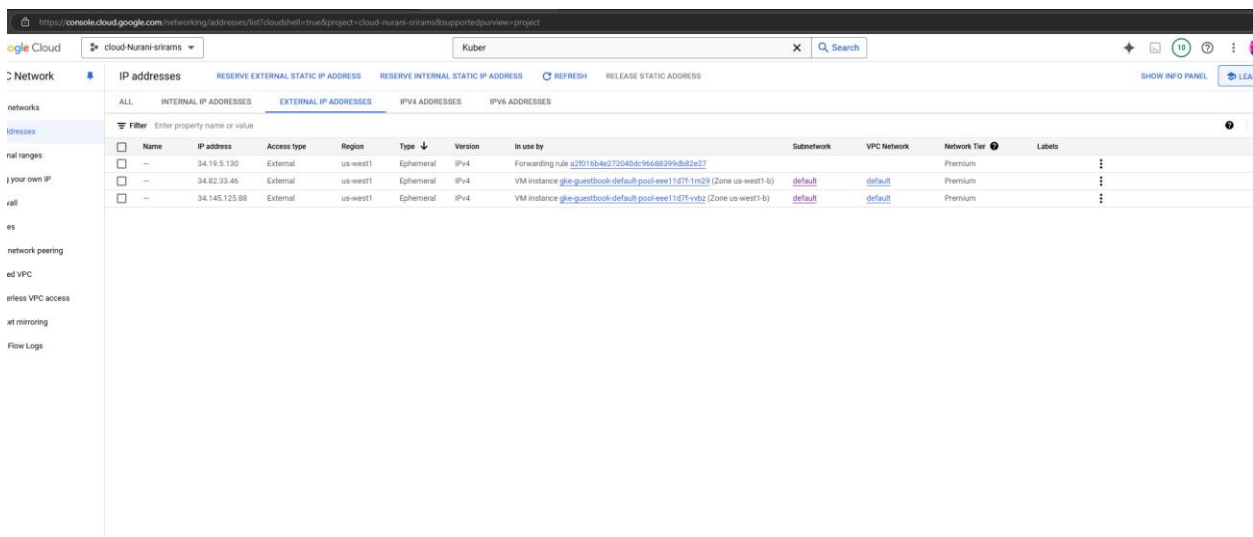
- Take a screenshot of the managed guestbook pods and the service being exposed.



- Take a screenshot of the load balancer and its details



- Take a screenshot of the addresses allocated and indicate the ones associated with nodes versus the one associated with the load balancer.



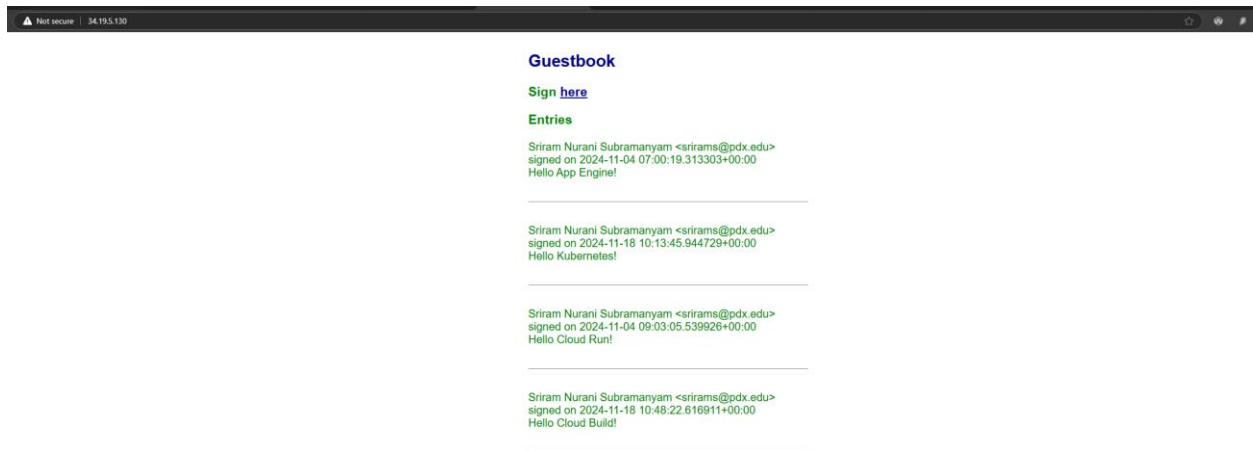
## 7.2.9 Delete workload and service

## 7.2.10 CI/CD build automation

## 7.2.11 Configure build automation

## 7.2.12 Deploy and view application

- Take a screenshot of the Guestbook including the URL with the entry in it.



## 7.2.13 Clean up

## 7.3: APIs (Slack, Knowledge Graph)

### 7.3.1 Slack and Knowledge Graph integration

### 7.3.2 Code

- **Does Google provide a Python package specifically for accessing the Knowledge Graph API?**

No, Google does not provide a specific Python package exclusively for accessing the Knowledge Graph API. However, you can use the Google API Client Library for Python to interact with Google's APIs, including the Knowledge Graph API. This library provides tools to authenticate and make requests to various Google APIs.

Alternatively, you can use generic HTTP libraries like requests to access the Knowledge Graph API by making RESTful API calls directly, as long as you have the API key.

### 7.3.3 Code

- **Show the source line that constructs the query we wish to send to the Knowledge Graph API.**

```
kg_search_response = make_search_request(request.form['text'])
```

This line passes the query (from request.form['text']) to the make\_search\_request function, which constructs the query.

- **Show the source line that then executes the query and saves the response. What is the name of the method that sends the query to the Knowledge Graph API?**

```
# [START functions_slack_request]
def make_search_request(query):
    req = kgsearch.entities().search(query=query, limit=1)
    res = req.execute()
    return format_slack_message(query, res)
```

The source line that executes the query to the Knowledge Graph API and saves the response is res=req.execute()

The name of the method that sends the query to the Knowledge Graph API is execute(). This method is used after constructing the request with the desired parameters for the Knowledge Graph Search API. The execute() method sends the request to the API and retrieves the response.

- **What is the Python data type that is used to represent the formatted message?**

```
message = {
    "response_type": "in_channel",
    "text": f"Query: {query}",
    "attachments": [],
}
```

The Python data type used to represent the formatted message is a dictionary (dict). The message variable is a dictionary with keys such as "response\_type", "text", and "attachments", where "attachments" is a list that can contain more dictionaries representing individual attachments.

The text attribute within this dictionary is a string (str) that contains the main textual content of the Slack message. Therefore, while the text attribute is indeed text (a string in



Python), the entire formatted message that includes this text and other information is structured as a dictionary.

- **What are the three main attributes of the formatted message passed back to Slack?**

The three main attributes of the formatted message passed back to Slack, as constructed in the `format_slack_message` function, are "response\_type", "text", and "attachments".

## 7.3.4 Knowledge Graph setup

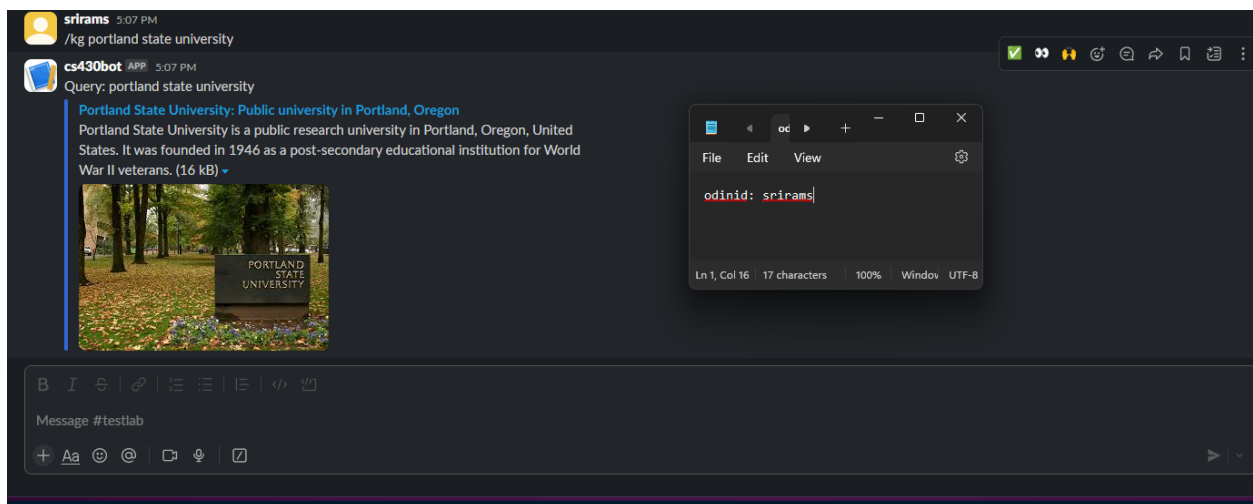
## 7.3.5 Create a Slack workspace

## 7.3.6 Configure and Deploy

## 7.3.7 Create Slack command

## 7.3.8 Test the command

- **Take a screenshot of its response for your lab notebook.**



# 7.4: ML APIs

## 7.4.1 APIs #1 (Vision, Speech, Translate, Natural Language APIs)

## 7.4.2 IAM service account setup

## 7.4.3 Vision

- **Show the output for your lab notebook**

```
(env) srirams@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-murani-srirams)$ python detect.py labels-uri gs://cloud-samples-data/ml-api-codelab/birds.jpg
Labels:
Bird
Ratite
Cloud
Sky
Beak
Plant
Green
Neck
Ostrich
Casuariiformes
(env) srirams@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-murani-srirams)$
```

- **What is the name of the function?**

```
# [END vision_label_detection]

# [START vision_label_detection_gcs]
def detect_labels_uri(uri):
    """Detects labels in the file located in Google Cloud Storage or on the
    Web."""
    from google.cloud import vision

    client = vision.ImageAnnotatorClient()
    image = vision.Image()
    image.source.image_uri = uri

    response = client.label_detection(image=image)
    labels = response.label_annotations
    print("Labels:")

    for label in labels:
        print(label.description)

    if response.error.message:
        raise Exception(
            "{}\nFor more info on error messages, check: "
            "https://cloud.google.com/apis/design/errors".format(response.error.message)
        )

# [END vision_label_detection_gcs]
```

The function that is called given the arguments of the command `python detect.py labels-uri gs://cloud-samples-data/ml-api-codelab/birds.jpg` is “`detect_labels_uri`”. This function is designed to detect labels in the file located in Google Cloud Storage or on the Web, as indicated by the command argument `labels-uri`. It takes a URI as its parameter, which points to the image file for which the labels need to be detected

- **What type of Vision client is instantiated in it?**

In the `detect_labels_uri` function, the type of Vision client instantiated is a `google.cloud.vision.ImageAnnotatorClient` i.e In the `detect_labels_uri` function, a `vision.ImageAnnotatorClient` from the Google Cloud Vision API is instantiated. This client is used to perform label detection and other vision-related tasks on images, including those stored in Google Cloud Storage or accessible via a URI on the web.

- **What method is invoked in the Vision client to perform the detection?**

In the `detect_labels_uri` function, the method invoked on the `vision.ImageAnnotatorClient` to perform the label detection is `label_detection`. This

method is called with an image parameter, which is an instance of vision.Image configured with the source image URI for the detection task

- **What is the name of the attribute in the response object that contains the results we seek?**

The name of the attribute in the response object that contains the results sought after label detection is performed using the ImageAnnotatorClient's label\_detection method is “label\_annotations”. This attribute contains a list of detected labels (objects) in the image, where each label provides information such as the label description, score, and other relevant details.

- **Take a screenshot of the output for the above commands**

```
(env) srirams@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-nurani-srirams)$ wget https://www.pdx.edu/themes/custom/pdxds/psuologo_horiz-spot.svg -O psu.svg
--2024-11-19 02:16:30-- https://www.pdx.edu/themes/custom/pdxds/psuologo_horiz-spot.svg
Resolving www.pdx.edu (www.pdx.edu)... 108.138.94.27, 108.138.94.85, 108.138.94.13, ...
Connecting to www.pdx.edu (www.pdx.edu)[108.138.94.27]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15593 (15K) [image/svg+xml]
Saving to: 'psu.svg'

psu.svg                               100%[=====>] 15.23K --.-KB/s  in 0s

2024-11-19 02:16:31 (519 MB/s) - 'psu.svg' saved [15593/15593]

(env) srirams@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-nurani-srirams)$ ls -ltrh
total 152K
-rw-rw-r-- 1 srirams srirams 16K Oct 31 23:39 psu.svg
-rw-rw-r-- 1 srirams srirams 55 Nov 19 01:43 requirements.txt
-rw-rw-r-- 1 srirams srirams 42 Nov 19 01:43 requirements-test.txt
-rw-rw-r-- 1 srirams srirams 836 Nov 19 01:43 README.rst.in
-rw-rw-r-- 1 srirams srirams 11K Nov 19 01:43 README.rst
-rw-rw-r-- 1 srirams srirams 7.7K Nov 19 01:43 detect_test.py
-rw-rw-r-- 1 srirams srirams 37K Nov 19 01:43 detect.py
-rw-rw-r-- 1 srirams srirams 2.7K Nov 19 01:43 beta_snippets_test.py
-rw-rw-r-- 1 srirams srirams 16K Nov 19 01:43 beta_snippets.py
drwxrwxr-x 3 srirams srirams 4.0K Nov 19 01:43 resources
-rw-rw-r-- 1 srirams srirams 972 Nov 19 01:43 vision_batch_annotate_files_test.py
-rw-rw-r-- 1 srirams srirams 2.3K Nov 19 01:43 vision_batch_annotate_files.py
-rw-rw-r-- 1 srirams srirams 992 Nov 19 01:43 vision_batch_annotate_files_gcs_test.py
-rw-rw-r-- 1 srirams srirams 2.3K Nov 19 01:43 vision_batch_annotate_files_gcs.py
-rw-rw-r-- 1 srirams srirams 749 Nov 19 01:43 set_endpoint_test.py
-rw-rw-r-- 1 srirams srirams 1.6K Nov 19 01:43 set_endpoint.py
-rw-rw-r-- 1 srirams srirams 988 Nov 19 02:10 wget-log
-rw-rw-r-- 1 srirams srirams 992 Nov 19 02:11 wget-log.1
-rw-rw-r-- 1 srirams srirams 992 Nov 19 02:11 wget-log.2
-rw-rw-r-- 1 srirams srirams 992 Nov 19 02:12 wget-log.3
(env) srirams@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-nurani-srirams)$ ls
beta_snippets.py  detect_test.py  README.rst      requirements.txt  set_endpoint_test.py  vision_batch_annotate_files.py  wget-log.1
beta_snippets_test.py  psu.jpg        README.rst.in  resources        vision_batch_annotate_files_gcs.py  vision_batch_annotate_files_test.py  wget-log.2
detect.py         psu.svg        requirements-test.txt  set_endpoint.py  vision_batch_annotate_files_gcs_test.py  wget-log.3
(env) srirams@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-nurani-srirams)$ python detect.py Logos psu.jpg
Logos:
Portland State University
(env) srirams@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-nurani-srirams)$
```

- **What method is invoked in the Vision client to perform the detection?**

The method invoked in the Vision client to perform logo detection on a local file is logo\_detection. This method is called with an image parameter, which is an instance of vision.Image configured with the content of the image file for the detection task. Using the detect.py script to perform logo detection by specifying the logos command and the path to the downloaded image file

```

# [START vision_logo_detection]
def detect_logos(path):
    """Detects logos in the file."""
    from google.cloud import vision

    client = vision.ImageAnnotatorClient()

    # [START vision_python_migration_logo_detection]
    with open(path, "rb") as image_file:
        content = image_file.read()

    image = vision.Image(content=content)

    response = client.logo_detection(image=image)
    logos = response.logo_annotations
    print("Logos:")

    for logo in logos:
        print(logo.description)

    if response.error.message:
        raise Exception(
            "{}\nFor more info on error messages, check: "
            "https://cloud.google.com/apis/design/errors".format(response.error.message)
        )
    # [END vision_python_migration_logo_detection]

# [END vision_logo_detection]

```

## 7.4.4 Speech

- Show the output for your lab notebook

```

(env) srirams@cloudshell:~/python-docs-samples/speech/snippets (cloud-nurani-srirams)$ python transcribe.py resources/audio.raw
Transcript: how old is the Brooklyn Bridge
(env) srirams@cloudshell:~/python-docs-samples/speech/snippets (cloud-nurani-srirams)$

```

- What is the name of the function?

```

def transcribe_file(audio_file: str) -> speech.RecognizeResponse:
    """Transcribe the given audio file.
    Args:
        audio_file (str): Path to the local audio file to be transcribed.
            Example: "resources/audio.wav"
    Returns:
        cloud_speech.RecognizeResponse: The response containing the transcription results
    """
    client = speech.SpeechClient()

    # [START speech_python_migration_sync_request]
    # [START speech_python_migration_config]
    with open(audio_file, "rb") as f:
        audio_content = f.read()

    audio = speech.RecognitionAudio(content=audio_content)
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=16000,
        language_code="en-US",
    )
    # [END speech_python_migration_config]

    # [START speech_python_migration_sync_response]
    response = client.recognize(config=config, audio=audio)

    # [END speech_python_migration_sync_request]
    # Each result is for a consecutive portion of the audio. Iterate through
    # them to get the transcripts for the entire audio file.
    for result in response.results:
        # The first alternative is the most likely one for this portion.
        print(f"Transcript: {result.alternatives[0].transcript}")
    # [END speech_python_migration_sync_response]

    return response

# [END speech_transcribe_sync]

```

Given the command `python transcribe.py resources/audio.raw`, the function called to handle this particular transcription task is: Function Name: `transcribe_file` This function is designed to transcribe audio files that are stored locally.

- What method is invoked in the Speech client to perform the detection?

```
# [START speech_python_migration_sync_response]
response = client.recognize(config=config, audio=audio)
```

Method Invoked in the Speech Client: The method invoked on the `speech.SpeechClient` to perform the transcription is `recognize`. This method takes two key arguments: `config` and `audio`.

- What is the name of the attribute in the response object that contains the results we seek?

```
# [START speech_python_migration_sync_response]
response = client.recognize(config=config, audio=audio)

# [END speech_python_migration_sync_request]
# Each result is for a consecutive portion of the audio. Iterate through
# them to get the transcripts for the entire audio file.
for result in response.results:
    # The first alternative is the most likely one for this portion.
    print(f"Transcript: {result.alternatives[0].transcript}")
# [END speech_python_migration_sync_response]

return response

# [END speech_transcribe_sync]
```

Attribute in the Response Object Containing the Results: The name of the attribute in the response object that contains the transcription results we seek is `results`. The results attribute of the response object is a list where each element represents a consecutive portion of the audio file that has been transcribed

## 7.4.5 Translate

- Show the output for your lab notebook

```
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$ python snippets.py translate-text en '你有没有带外套'
Text: 你有没有带外套
Translation: did you bring a coat
Detected source language: zh-TW
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$
```

- What is the name of the function?

Function Name: translate\_text

```
# [START translate_translate_text]
def translate_text(target: str, text: str) -> dict:
    """Translates text into the target language.

    Target must be an ISO 639-1 language code.
    See https://g.co/cloud/translate/v2/translate-reference#supported_languages
    """
    from google.cloud import translate_v2 as translate

    translate_client = translate.Client()

    if isinstance(text, bytes):
        text = text.decode("utf-8")

    # Text can also be a sequence of strings, in which case this method
    # will return a sequence of results for each text.
    result = translate_client.translate(text, target_language=target)

    print("Text: {}".format(result["input"]))
    print("Translation: {}".format(result["translatedText"]))
    print("Detected source language: {}".format(result["detectedSourceLanguage"]))

    return result

# [END translate_translate_text]
```

- What method is invoked in the Translate client to perform the detection?

Method Invoked in the Translate Client: The method invoked on the translate.Client to perform the translation is translate.

```
# Text can also be a sequence of strings, in which case this method
# will return a sequence of results for each text.
result = translate_client.translate(text, target_language=target)
```

- What is the name of the attribute in the response object that contains the results we seek?

The name of the attribute in the response object that contains the translation result we seek is translatedText.

```
print("Text: {}".format(result["input"]))
print("Translation: {}".format(result["translatedText"]))
print("Detected source language: {}".format(result["detectedSourceLanguage"]))
```

## 7.4.6 Natural Language

- Show the output for your lab notebook

```
python can't open file '/home/srirams/python-docs-samples/translate/samples/snippets/language.py': [Errno 2] No such file or directory
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$ nano language.py
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$ python language.py 'homework is awful!'
python language.py 'homework is ok'
python language.py 'homework is awesome?'
python language.py 'homework is awesome!'
python language.py 'The protestors in Oregon put on gas masks and wore yellow t-shirts'
"homework is awful!" has sentiment=-0.800000011920929
"homework is ok" has sentiment=0.30000001192092896
"homework is awesome?" has sentiment=0.4000000059604645
"homework is awesome!" has sentiment=0.8999999761581421
"The protestors in Oregon put on gas masks and wore yellow t-shirts" has sentiment=-0.6000000238418579
Entities are:
name: protestors
name: gas masks
name: Oregon
name: t-shirts
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$
```

## 7.4.7 Integration

## 7.4.8 Code

- What is the name of the function that performs the transcription?

```
def transcribe_gcs(language, gcs_uri):
    """Transcribes the audio file specified by the gcs_uri."""

    # create ImageAnnotatorClient object
    client = speech.SpeechClient()

    # specify location of speech
    audio = speech.RecognitionAudio(uri=gcs_uri) # need to specify speech.types

    # set language to Turkish
    # removed encoding and sample_rate_hertz
    config = speech.RecognitionConfig(language_code=language) # need to specify speech.

    # get response by passing config and audio settings to client
    response = client.recognize(config=config, audio=audio)

    # naive assumption that audio file is short
    return response.results[0].alternatives[0].transcript
```

The name of the function that performs the transcription is `transcribe_gcs`.

- What is the name of the function that performs the translation?

The name of the function that performs the translation is `translate_text`.

```
def translate_text(target, text):
    """Translates text into the target language.

    Target must be an ISO 639-1 language code.
    See https://g.co/cloud/translate/v2/translate-reference#supported_languages
    """

    # create Client object
    translate_client = translate.Client()

    # decode text if it's a binary type
    if isinstance(text, six.binary_type):
        text = text.decode('utf-8')

    # get translation result by passing text and target language to client
    # Text can also be a sequence of strings, in which case this method
    # will return a sequence of results for each text.
    result = translate_client.translate(text, target_language=target)

    # only interested in translated text
    return result['translatedText']
```

- What is the name of the function that performs the entity analysis on the translation?

```
def entities_text(text):
    """Detects entities in the text."""

    # create LanguageServiceClient object
    client = language.LanguageServiceClient()

    # decode text if it's a binary type
    if isinstance(text, six.binary_type):
        text = text.decode('utf-8')

    # Instantiates a plain text document.
    # need to specify language.types
    request = {'document' : {"type_": language.Document.Type.PLAIN_TEXT, "content": text}}
    entities = client.analyze_entities(request).entities

    # we only need the entity names
    entity_names = []
    for entity in entities:
        entity_names.append(entity.name)

    return entity_names
```

The name of the function that performs the entity analysis on the translation is `entities_text`.



- What is the name of the function that performs the entity analysis on the image?

```
def detect_labels_uri(uri):
    """Detects labels in the file located in Google Cloud Storage or on the
    Web."""

    # create ImageAnnotatorClient object
    client = vision.ImageAnnotatorClient()

    # create Image object
    image = vision.Image()

    # specify location of image
    image.source.image_uri = uri

    # get label_detection response by passing image to client
    response = client.label_detection(image=image)

    # get label_annotations portion of response
    labels = response.label_annotations

    # we only need the label descriptions
    label_descriptions = []
    for label in labels:
        label_descriptions.append(label.description)

    return label_descriptions
```

The function that analyzes the image to identify labels, which can be considered as performing entity analysis on the image, is named `detect_labels_uri`.

### 7.4.9 Test integration

```
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$ python solution.py de-DE gs://cloud-samples-data/ml-api-codelab/de-ball.wav gs://cloud-samples-data/ml-api-codelab/football.jpg
Transcription: willst du mit uns Fußball spielen
Translation: do you want to play football with us
Entities: ['football']
Image labels: ['Sports equipment', 'Soccer', 'Football', 'Plant', 'Ball', 'Player', 'Playing sports', 'Soccer ball', 'Ball game', 'Team sport']
The audio and image do not appear to be related.
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$
```

- If the program deems them unrelated, then based on the results from the APIs, what must be changed in the program to address this?

The program successfully identifies relevant entities in both the audio transcription and the image labels but determines they are unrelated. This highlights the need to refine the logic for comparing entities derived from audio and image labels.

**Suggestions for Enhancing Comparison Logic:**

- **Expand Matching Criteria:** Instead of requiring exact matches, incorporate broader criteria that account for synonyms, related concepts (e.g., "Soccer" and "Football"), and hierarchical relationships (e.g., "Soccer ball" as a subset of "Sports equipment").
- **Leverage Semantic Analysis:** Utilize semantic analysis to interpret the meaning of entities and labels more effectively. This could involve using NLP tools or services to analyze the context and establish more nuanced connections between the textual and visual content.

```
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$ python solution.py tr-TR gs://cloud-samples-data/ml-api-codelab/bicycle.jpg gs://cloud-samples-data/ml-api-codelab/tr-bike.wav
Transcription: bisikletimi sokaga birak
Translation: leave my bike on the street
Entities: ['bike', 'street']
Image labels: ['Bicycle', 'Clothing', 'Footwear', 'Tire', 'Wheel', 'Bicycles--Equipment and supplies', 'Land vehicle', 'Shoe', 'Bicycle frame', 'Bicycle wheel']
The audio and image do not appear to be related.
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$
```

- **If the program deems them unrelated, then based on the results from the APIs, what must be changed in the program to address this?**

The program's failure to identify the connection between the audio content ("leave my bike on the street") and the image content (which includes a "Bicycle" among its labels) highlights the need to improve the comparison logic for better understanding and linking the semantic context of entities and labels.

### Suggestions for Improvement:

- **Fuzzy Logic for Matching:** Introduce fuzzy logic to match entities and labels, enabling "close enough" matches based on similarity scores rather than requiring exact matches. This approach can accommodate variations in language or terminology.
- **Adjust Confidence Thresholds:** Modify confidence thresholds for determining relevant matches, incorporating user feedback or leveraging machine learning models to optimize these thresholds over time.

```
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$ python solution.py tr-TR gs://cloud-samples-data/ml-api-codelab/birds.jpg gs://cloud-samples-data/ml-api-codelab/tr-ostrich.wav
Transcription: gok fazla deve kugu var
Translation: There are too many ostriches
Entities: ['ostriches']
Image labels: ['Bird', 'Ratite', 'Cloud', 'Sky', 'Beak', 'Plant', 'Green', 'Beck', 'Ostrich', 'Casuariiformes']
The audio and image do not appear to be related.
(env) srirams@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-murani-srirams)$
```

- **If the program deems them unrelated, then based on the results from the APIs, what must be changed in the program to address this?**

In cases where the program fails to recognize the connection between audio content like "there are too many ostriches" and an image that clearly features ostriches, it is evident that the logic for determining relatedness requires improvement.

#### **Suggestions for Refining Comparison Logic:**

- **Contextual and Hierarchical Analysis:** Enhance the comparison logic by incorporating the context in which terms are used and recognizing hierarchical relationships between categories. For example, understanding that an "ostrich" is a type of "Bird" can help establish connections between specific entities and broader labels.
- **Feedback Loops for Continuous Improvement:**
  - **User Feedback:** Introduce a feedback loop that enables users to correct the system when it misjudges the relevance of audio and image content.
  - Use this feedback to train and refine the matching algorithms and comparison logic continually.

### 7.4.10 APIs #2 (Video Intelligence API)

#### 7.4.11 Video setup

#### 7.4.12 Video Intelligence labeling script

#### 7.4.13. Video Intelligence

- **What are the 3 labels with the highest confidence that the Video Intelligence API associates with the video and what are the confidences for each?**

The three labels with the highest confidence associated with the video are:

Sports: Confidence of 0.9218811392784119

Basketball: Confidence of 0.9137870669364929

Player: Confidence of 0.8446521162986755

- **What is the name of the client class in the package that is used?**

```
def analyze_labels(path):
    """ Detects labels given a GCS path. """
    video_client = videointelligence.VideoIntelligenceServiceClient()
    features = [videointelligence.Feature.LABEL_DETECTION]
    operation = video_client.annotate_video(input_uri=path, features=features)
```

The client class used in the package is VideoIntelligenceServiceClient from the google.cloud.videointelligence module.

- **What method is used in that class to perform the annotation?**

The method used in that class to perform the annotation is annotate\_video

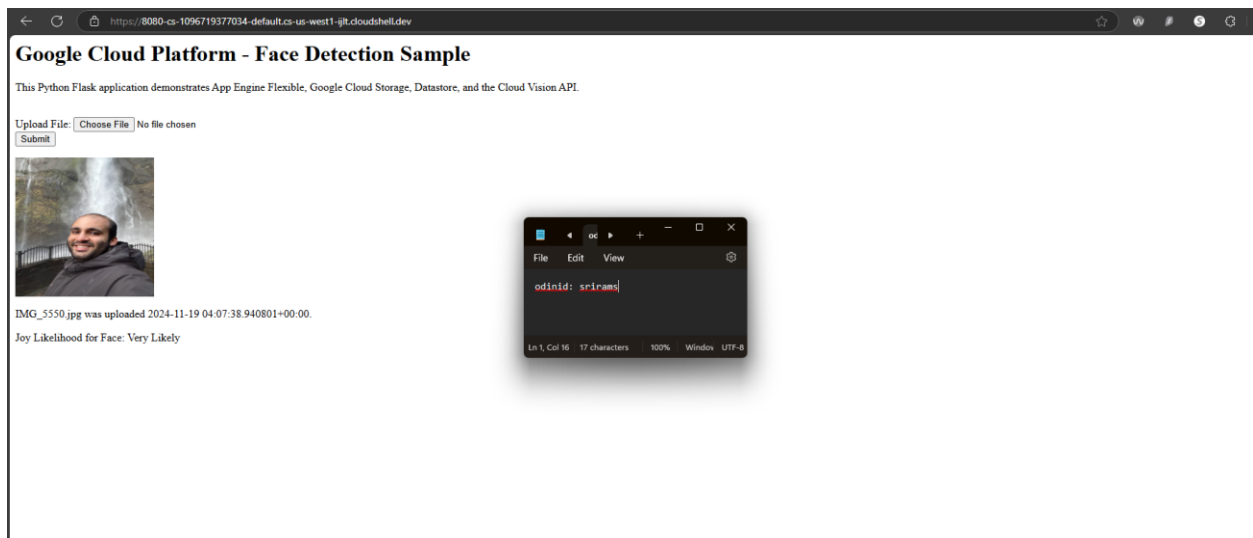
```
def analyze_labels(path):
    """ Detects labels given a GCS path. """
    video_client = videointelligence.VideoIntelligenceServiceClient()
    features = [videointelligence.Feature.LABEL_DETECTION]
    operation = video_client.annotate_video(input_uri=path, features=features)
```

## 7.4.14 APIs #3 (Web site integration)

## 7.4.15 IAM service account setup

## 7.4.16 Application

- **Take a screenshot for your lab notebook that includes the URL.**



## 7.4.17 Code

- **What line of code creates the query for previous detections?**

```
# Use the Cloud Datastore client to fetch information from Datastore about
# each photo.
query = datastore_client.query(kind="Faces")
image_entities = list(query.fetch())
```

```
query = datastore_client.query(kind="Faces")
```

- What line of code sends the query to Cloud Datastore?

```
# Use the Cloud Datastore client to fetch information from Datastore about
# each photo.
query = datastore_client.query(kind="Faces")
image_entities = list(query.fetch())
```

```
image_entities = list(query.fetch())
```

- Show the line that retrieves the name of the storage bucket to use.

```
# Get the bucket that the file will be uploaded to.
bucket = storage_client.get_bucket(CLOUD_STORAGE_BUCKET)
```

```
bucket = storage_client.get_bucket(CLOUD_STORAGE_BUCKET)
```

- What form field is used to specify the uploaded photo?

```
@app.route("/upload_photo", methods=["GET", "POST"])
def upload_photo():
    photo = request.files["file"]
```

```
photo = request.files["file"]
```

- Show the line that copies the photo's contents to the storage bucket.

```
# Create a new blob and upload the file's content.
blob = bucket.blob(photo.filename)
blob.upload_from_string(photo.read(), content_type=photo.content_type)

# Make the blob publicly viewable.
blob.make_public()
```

```
blob.upload_from_string(photo.read(), content_type=photo.content_type)
```

- What method in Vision's annotation client is used to perform the analysis?

```
# Create a Cloud Vision client.
vision_client = vision.ImageAnnotatorClient()

# Use the Cloud Vision client to detect a face for our image.
source_uri = "gs://{}/{}/{}".format(CLOUD_STORAGE_BUCKET, blob.name)
image = vision.Image(source=vision.ImageSource(gcs_image_uri=source_uri))
faces = vision_client.face_detection(image=image).face_annotations
```

faces = vision\_client.face\_detection(image=image).face\_annotations

- What fields are stored in Cloud Datastore for each image?

```
# Construct the new entity using the key. Set dictionary values for entity
# keys blob_name, storage_public_url, timestamp, and joy.
entity = datastore.Entity(key)
entity["blob_name"] = blob.name
entity["image_public_url"] = blob.public_url
entity["timestamp"] = current_datetime
entity["joy"] = face_joy

# Save the new entity to Datastore.
datastore_client.put(entity)

# Redirect to the home page.
return redirect("/")
```

entity["blob\_name"] = blob.name

entity["image\_public\_url"] = blob.public\_url

entity["timestamp"] = current\_datetime

entity["joy"] = face\_joy

- What happens at the end of the `upload_photo` route?

```
# Save the new entity to Datastore.
datastore_client.put(entity)

# Redirect to the home page.
return redirect("/")
```

return redirect("/")

## 7.4.18 Clean up