Se iniciara con crear un servidor de Python

Copy code

```
pip install flask
```

Copy code

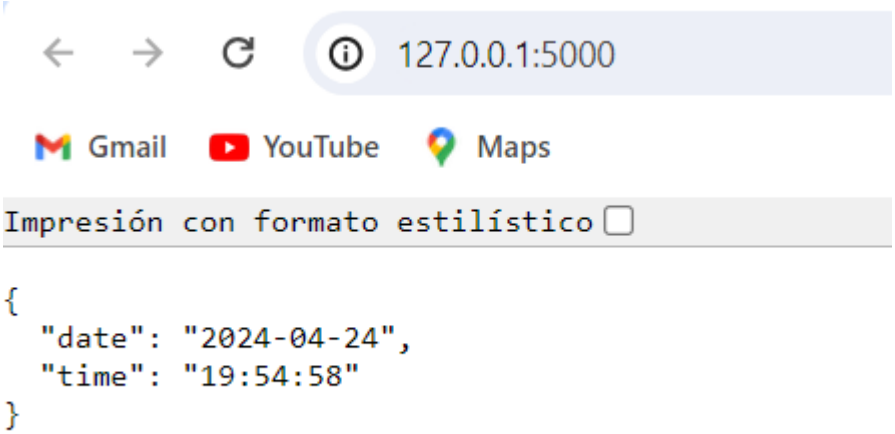
```
pip install pytz
```

Primero instalamos las librerías que necesitamos

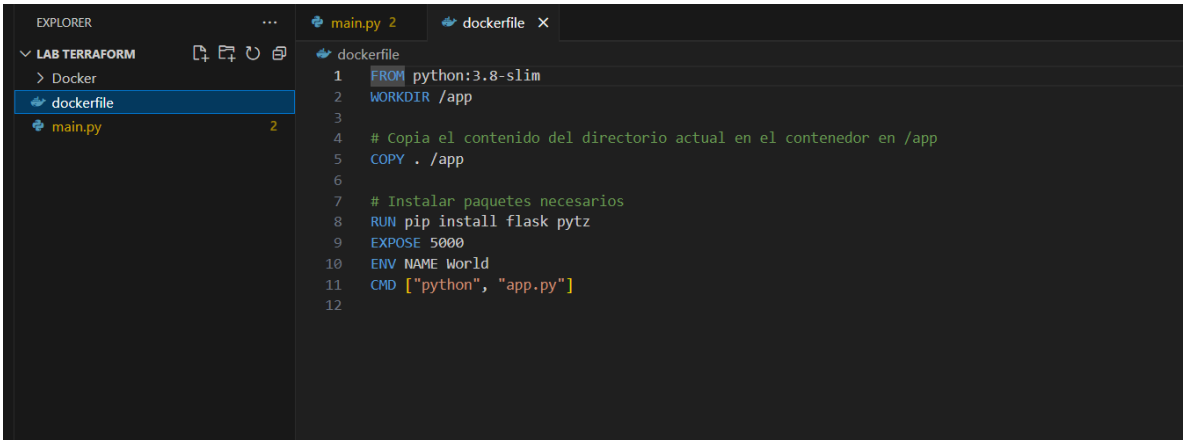
```
main.py > get_datetime
from flask import Flask, jsonify
from datetime import datetime
import pytz

app = Flask(__name__)

@app.route('/')
def get_datetime():
    now = datetime.now(pytz.timezone('America/Guatemala'))
    return jsonify({
        'date': now.strftime("%Y-%m-%d"),
        'time': now.strftime("%H:%M:%S"),
    })

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

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

 * 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://10.203.145.164:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 596-497-820
127.0.0.1 - - [24/Apr/2024 19:54:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Apr/2024 19:55:00] "GET /favicon.ico HTTP/1.1" 404 -
```

Lo corremos y en este puerto podemos ver lo que devuelve




Generamos el archivo Docker



Y lo corremos (teniendo Docker instalado y con powershell)

```
Administrador: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\WINDOWS\system32> cd "C:\Users\DELL\Desktop\U\Lab Terraform"
PS C:\Users\DELL\Desktop\U\Lab Terraform> docker build -t app_flask .
ERROR: error during connect: in the default daemon configuration on Windows, the docker client must be run with elevated privileges to connect: Get "http://%2F%2F.%2Fpipe%2Fdocker_engine/_ping": open //./pipe/do
cker_engine: The system cannot find the file specified.
PS C:\Users\DELL\Desktop\U\Lab Terraform> docker build -t app_flask .
[+] Building 22.8s (9/9) FINISHED                                                                                                                docker:default
 => [internal] load build definition from dockerfile                                                                                                        0.0s
 => => transferring dockerfile: 276B                                                                                                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                                                                                          1.8s
 => [internal] load .dockerignore                                                                                                                           0.0s
 => => transferring context: 2B                                                                                                                             0.0s
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:2f911e2866173a52104dc16b5e42b7069c2eba05eb78556d18b1ca665d0dc445                                   12.8s
 => => resolve docker.io/library/python:3.8-slim@sha256:2f911e2866173a52104dc16b5e42b7069c2eba05eb78556d18b1ca665d0dc445                                    0.0s
 => => sha256:2f911e2866173a52104dc16b5e42b7069c2eba05eb78556d18b1ca665d0dc445 1.86kB / 1.86kB                                                              0.0s
 => => sha256:11c49f621bb96ca802f21cb54a4940088f0bf85e5948db600433a11da779c774 1.37kB / 1.37kB                                                             0.0s
 => => sha256:5b29acce18dd86b702bc6a011e25b3d631090f7ffe35fe2a4977abf53eb3954 6.95kB / 6.95kB                                                              0.0s
 => => sha256:b0a0cf830b12453b7e15359a804215a7bcccd3788e2bcecff2a03af64bbd4df7 29.15MB / 29.15MB                                                            0.9s
 => => sha256:72914424168c8ebb0dbb3d0e08eb1d3b5b2a64cc51745bd65caf29c335b31dc7 3.51MB / 3.51MB                                                              1.1s
 => => sha256:545ebfaa75064d83d4862d6b0ca34e531e2e90d431f42a8e59968cc372099695 11.68MB / 11.68MB                                                            2.5s
 => => sha256:80ee918b20840648abeedaab21c9dd7a6b03105ec8f362d48d55195e0402ebfb 243B / 243B                                                                  1.2s
 => => sha256:d361726ad66f2bc2e1928c9b5ddaf7f33b10226d544315ca2408b7d9e2dacd16 3.14MB / 3.14MB                                                              2.1s
 => => extracting sha256:b0a0cf830b12453b7e15359a804215a7bcccd3788e2bcecff2a03af64bbd4df7                                                                    3.0s
 => => extracting sha256:72914424168c8ebb0dbb3d0e08eb1d3b5b2a64cc51745bd65caf29c335b31dc7                                                                    0.2s
 => => extracting sha256:545ebfaa75064d83d4862d6b0ca34e531e2e90d431f42a8e59968cc372099695                                                                    0.7s
 => => extracting sha256:80ee918b20840648abeedaab21c9dd7a6b03105ec8f362d48d55195e0402ebfb                                                                    0.0s
 => => extracting sha256:d361726ad66f2bc2e1928c9b5ddaf7f33b10226d544315ca2408b7d9e2dacd16                                                                    1.2s
 => [internal] load build context                                                                                                                           0.0s
 => => transferring context: 1.30kB                                                                                                                         0.0s
 => [2/4] WORKDIR /app                                                                                                                                      0.6s
 => [3/4] COPY . /app                                                                                                                                       0.0s
 => [4/4] RUN pip install flask pytz                                                                                                                        7.2s
 => exporting to image                                                                                                                                      0.3s
 => => exporting layers                                                                                                                                     0.2s
 => => writing image sha256:1a7f38eb7c6e064fda9a86d10b30e37a38ffc1b63c367aef108d21b338757df5                                                                0.0s
 => => naming to docker.io/library/app_flask                                                                                                                0.0s

View build details: docker-desktop://dashboard/build/default/default/iivxw8acjt1kxuyz2009gfzia

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\DELL\Desktop\U\Lab Terraform>
```

Lo corrremos y podemos ver que con las configuraciones también nos levanta en el mismo puerto.

```
View build details: docker-desktop://dashboard/build/default/default/q7evu1uv6962xf2xx6xmks18j

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\DELL\Desktop\U\Lab Terraform> docker run -p 5000:5000 app_flask
 * Serving Flask app 'app'
 * 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 stat
 * Debugger is active!
 * Debugger PIN: 263-818-055
```

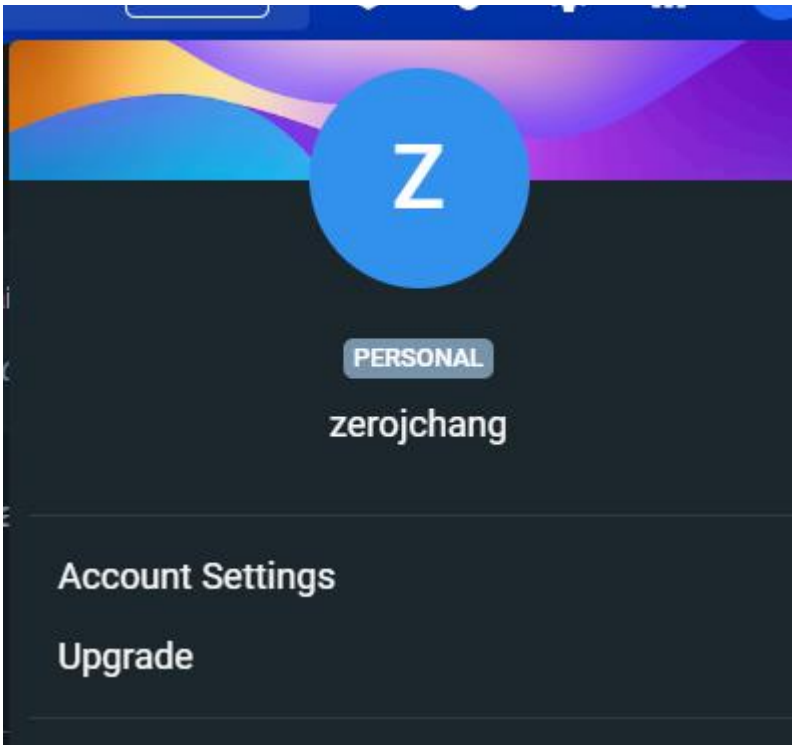
127.0.0.1:5000

Gmail YouTube Maps

Impresión con formato estilístico


```
{
  "date": "2024-04-24",
  "time": "19:54:58"
}
```

Ahora para levantar en la nube necesitamos correr nuestro usuario.




```
Administrador: Windows PowerShell
 * Debugger is active!
 * Debugger PIN: 263-818-055
172.17.0.1 - - [25/Apr/2024 02:04:05] "GET / HTTP/1.1" 200 -
PS C:\Users\DELL\Desktop\U\Lab Terraform> ^C
PS C:\Users\DELL\Desktop\U\Lab Terraform>  docker build -t zerojchang/flask-app-nube:latest .
[+] Building 0.0s (0/0)  docker:default
2024/04/24 20:08:24 http2: server: error reading preface from client //./pipe/docker_engine: file has already been close
[+] Building 0.9s (10/10) FINISHED                                                                      docker:default
 => [internal] load build definition from dockerfile                                                            0.0s
 => => transferring dockerfile: 276B                                                                            0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                                              0.8s
 => [auth] library/python:pull token for registry-1.docker.io                                                   0.0s
 => [internal] load .dockerignore                                                                               0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:2f911e2866173a52104dc16b5e42b7069c2eba05eb78556d18b1ca665  0.0s
 => [internal] load build context                                                                               0.0s
 => => transferring context: 148B                                                                               0.0s
 => CACHED [2/4] WORKDIR /app                                                                                   0.0s
 => CACHED [3/4] COPY . /app                                                                                    0.0s
 => CACHED [4/4] RUN pip install flask pytz                                                                     0.0s
 => exporting to image                                                                                          0.0s
 => => exporting layers                                                                                         0.0s
 => => writing image sha256:51c0f59c3563c3d68d4f0860a83fc3d5ba697e6c6ef04b939b1c0c9be55d7db7                    0.0s
 => => naming to docker.io/zerojchang/flask-app-nube:latest                                                     0.0s

View build details: docker-desktop://dashboard/build/default/default/ie1h7uwr7leiuwb3b25eg3x12

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\DELL\Desktop\U\Lab Terraform>
```

```
See 'docker run --help'.
PS C:\Users\DELL\Desktop\U\Lab Terraform> docker run -d -p 5050:5000 zerojchang/flask-app-nube:latest
7e5e83ce8315dabe3b2cd8fd556ef7fe011bad207dca72bf9f832d817e60642e
PS C:\Users\DELL\Desktop\U\Lab Terraform>
```

Para usar una nube debemos logueranos en este caso Google cloud (se escogió este debido a que por el curso Ingenieria de Software 2 tenemos 1 mes de prueba).

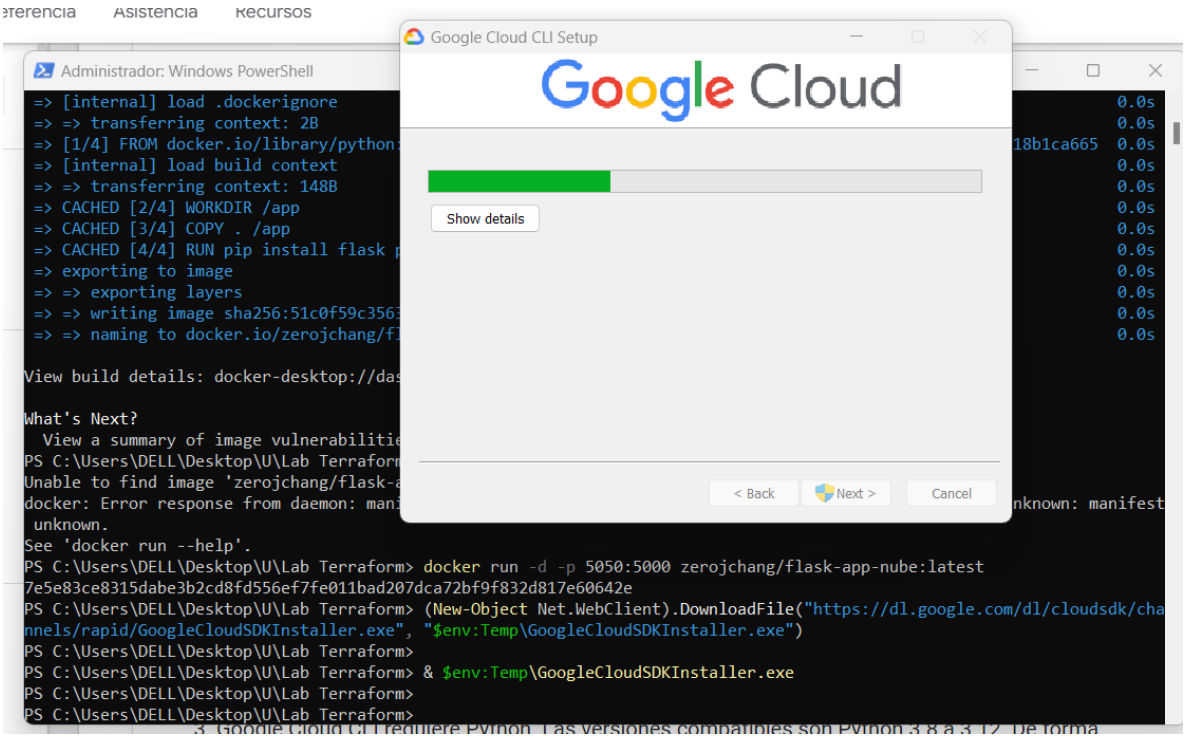
posteriores.

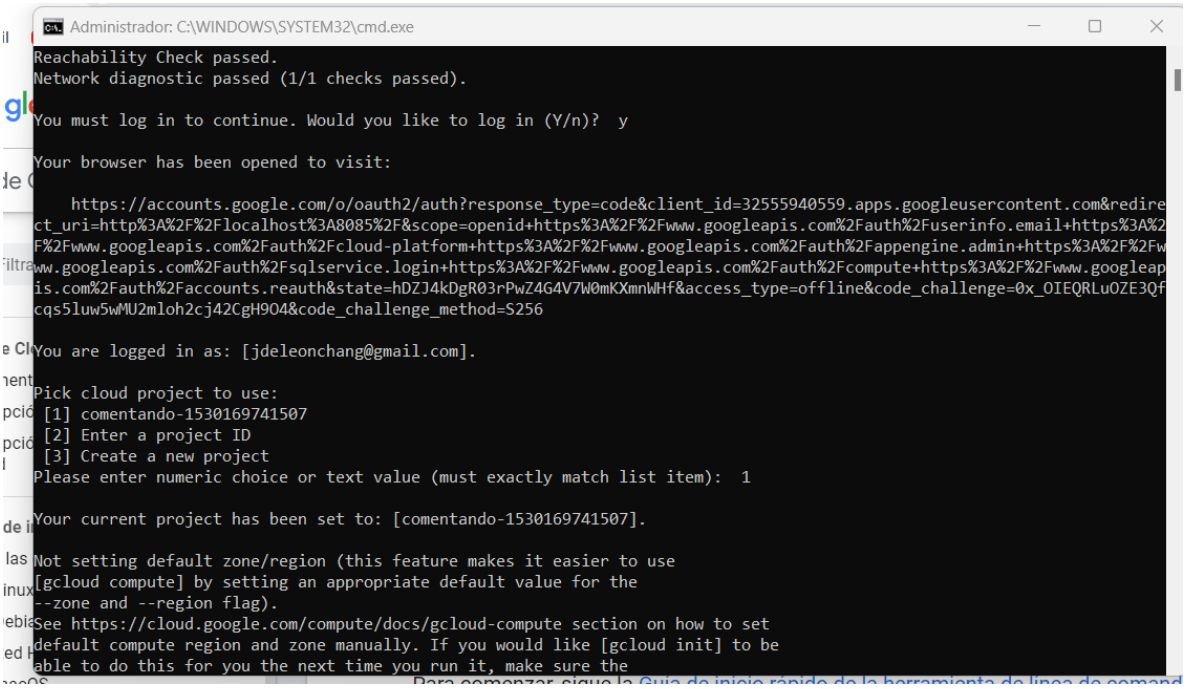
1. Descarga el instalador de la Google Cloud CLI.

   Como alternativa, abre una terminal de PowerShell y ejecuta los siguientes comandos:

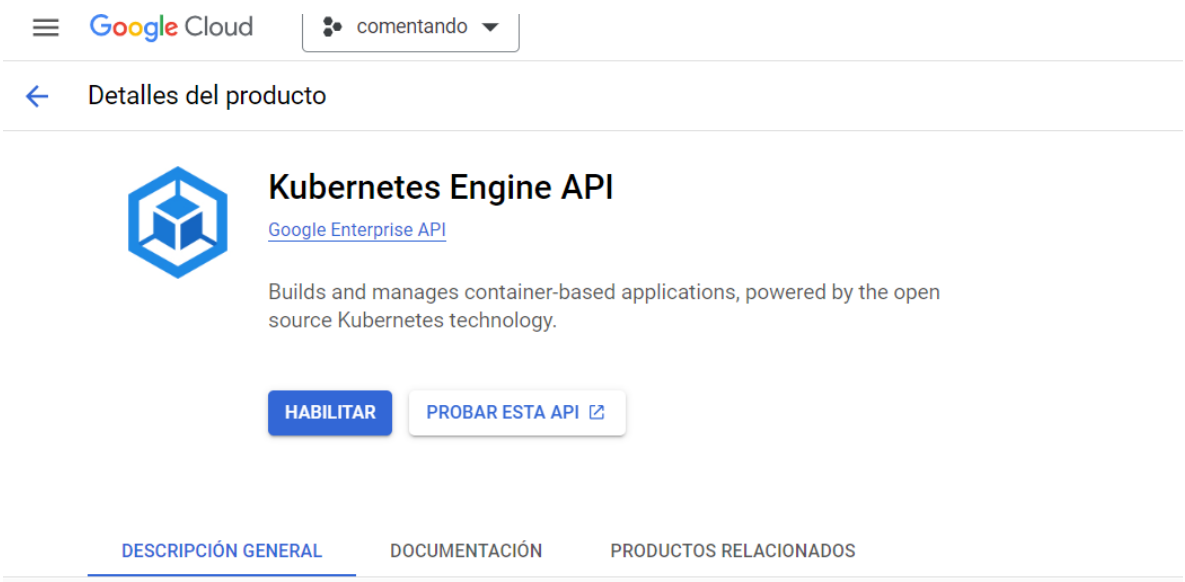
```
(New-Object Net.WebClient).DownloadFile("https://dl.google.com/dl/cloudsdk/channels/rapic

& $env:Temp\GoogleCloudSDKInstaller.exe
```

Lo instalamos

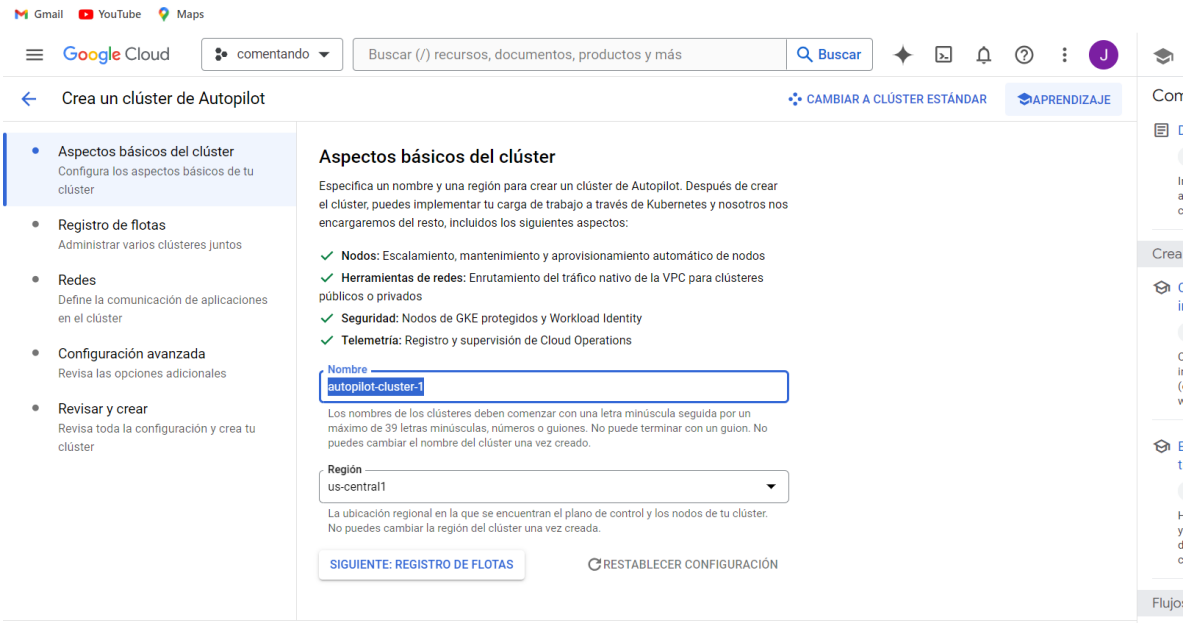




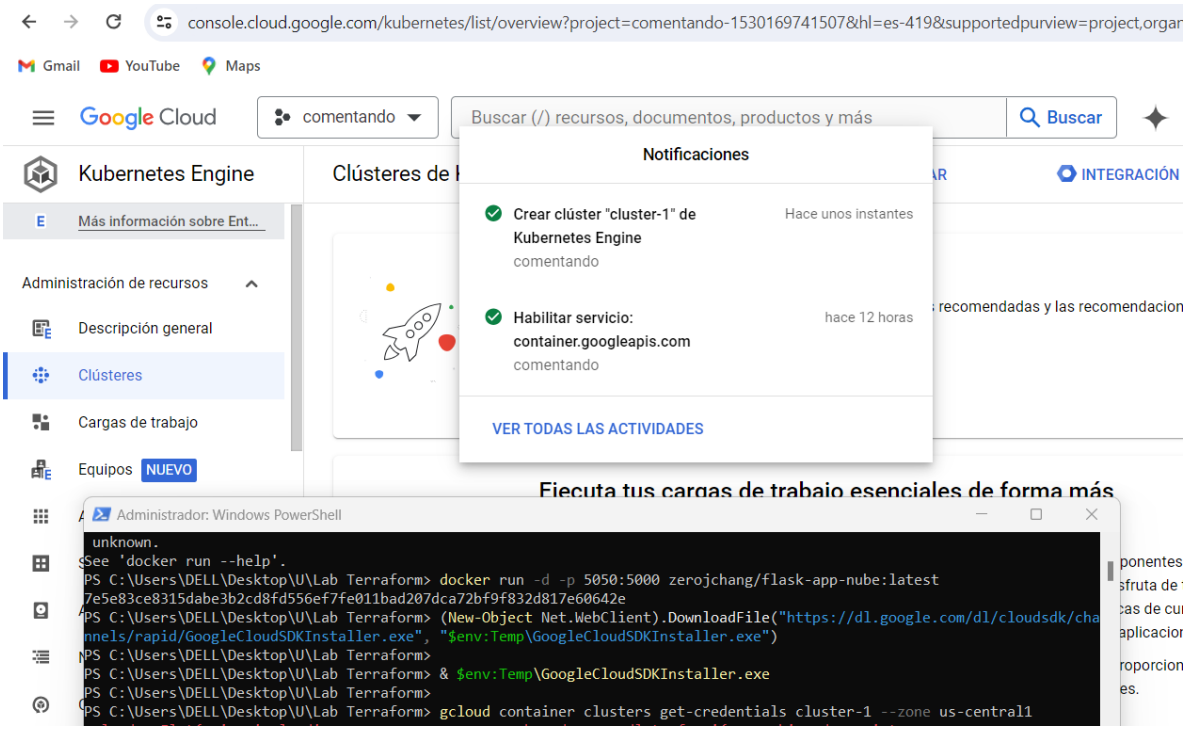

Nos pedirá elegir un proyecto en el cual trabajar.




Habilitamos Kubernetes en nuestro proyecto




Habilitamos un cluster




y ejecutamos

Creamos archivo .yaml

main.tf

```
provider "google"{
    project = "comentando"
    region = "us-central1"
}

resource "google_container_cluster" "cluster-cloud"{
    name = "cluster-1"
    location = "us-central1"
}
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: flask-app-nube
spec:
  selector:
    app: flask-app-nube
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
  type: LoadBalancer
```

Y configuraciones para Docker

Antes de proceder cambiamos el archivo .py para mostrar el dato faltante.

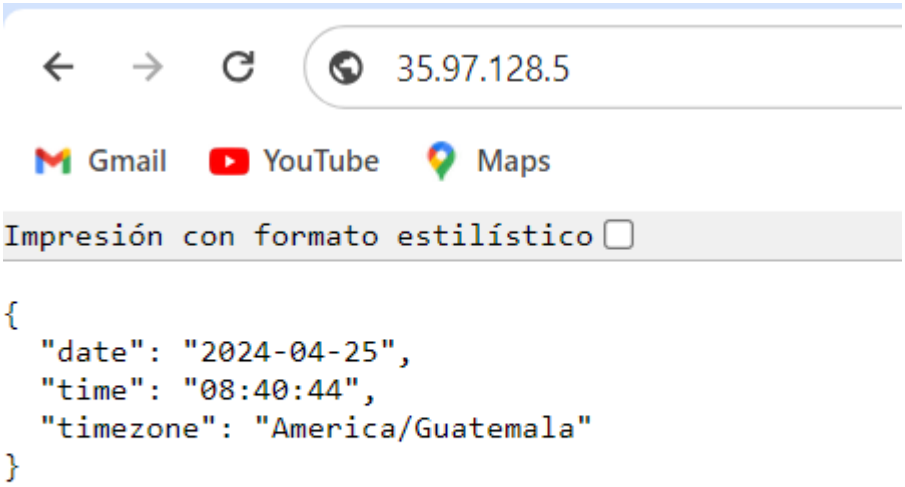
```python
from flask import Flask, jsonify
from datetime import datetime
import pytz

app = Flask(__name__)

@app.route('/')
def get_datetime():
    now = datetime.now(pytz.timezone('America/Guatemala'))
    return jsonify({
        'date': now.strftime("%Y-%m-%d"),
        'time': now.strftime("%H:%M:%S"),
        'timeZone' : now.tzinfo.zone
    })

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

Ejecutaos todos los pasos anteriores

```
PS C:\Users\DELL\Desktop\U\Lab Terraform> kubectl apply -f flask-app-nube-deploy.yaml
```

Corremos el comando para el despliegue

Este nos dará un link para poder acceder a nuestro API endpoint en la nube





```
{
  "date": "2024-04-25",
  "time": "08:40:44",
  "timezone": "America/Guatemala"
}
```


Como podemos ver al acceder al link nos muestra el resultado de nuestro endpoint