

Deploy App Engine Standard Service

Google App Engine is a Platform-as-a-Service (PaaS) provided by Google Cloud that allows developers to build and deploy applications without managing the underlying infrastructure. It abstracts away server management tasks such as provisioning, scaling, and patching, allowing developers to focus on writing code.

Use Cases

1. Web Applications

- Hosting scalable and resilient web applications using frameworks like Django, Flask, Express, or Spring.

2. APIs and Backend Services

- Running RESTful APIs and microservices for mobile apps or client-facing web apps.

3. Startups and MVPs

- Rapidly prototyping and deploying applications with minimal setup and operational overhead.

4. Enterprise Applications

- Deploying internal tools or business applications with high availability and auto-scaling.

5. Scheduled Tasks and Background Jobs

- Running cron jobs, asynchronous processing, or event-driven tasks using built-in services.

6. Education and Training

- Quick deployment of student projects or lab environments without infrastructure concerns.

Properties

- **Fully Managed:** No need to manage VMs, networking, or OS updates.
- **Automatic Scaling:** Applications scale up or down automatically based on demand.
- **Multiple Language Support:** Supports Python, Java, Node.js, Go, PHP, Ruby, and custom runtimes via Docker.
- **Two Environments:**

- **Standard Environment:** Lightweight, sandboxed, supports rapid scaling.
- **Flexible Environment:** Based on Docker containers, offers more control and customization.
- **Built-in Services:** Includes traffic splitting, version management, logging, monitoring, and task queues.
- **Integration with GCP:** Seamless integration with other Google Cloud services like Cloud SQL, Firestore, and Pub/Sub.

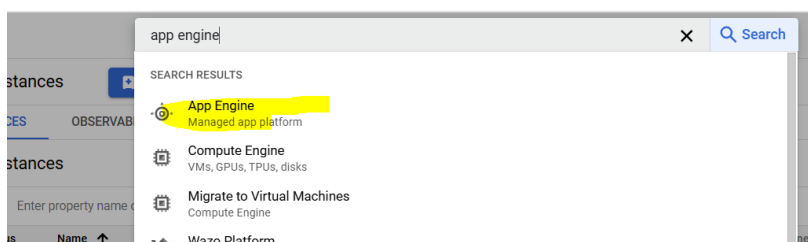
Benefits

- **Speed of Development:** Faster time to market with less operational complexity.
- **Reduced Operational Overhead:** No need to manage infrastructure or system-level concerns.
- **Scalability:** Automatically handles increases or decreases in traffic.
- **Cost Efficiency:** Pay only for what you use, with a free tier available.
- **Reliability and Security:** Google manages updates, security patches, and infrastructure reliability.
- **Flexibility:** Suitable for a range of applications, from simple websites to complex backends.

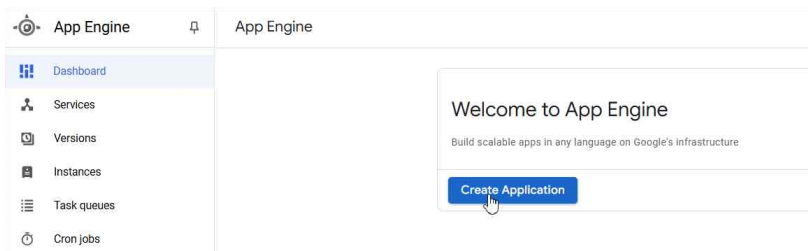
Google App Engine is well-suited for developers and teams looking to deploy applications quickly and scale efficiently without the burden of infrastructure management.

To begin with the Lab

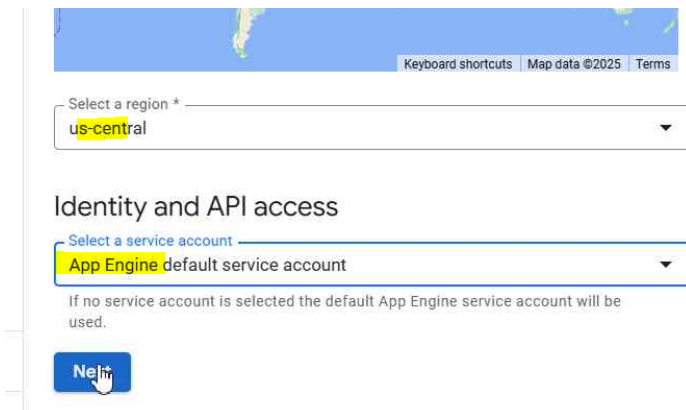
- 1) Go to the GCP console, search for App Engine



- 2) Click on create application

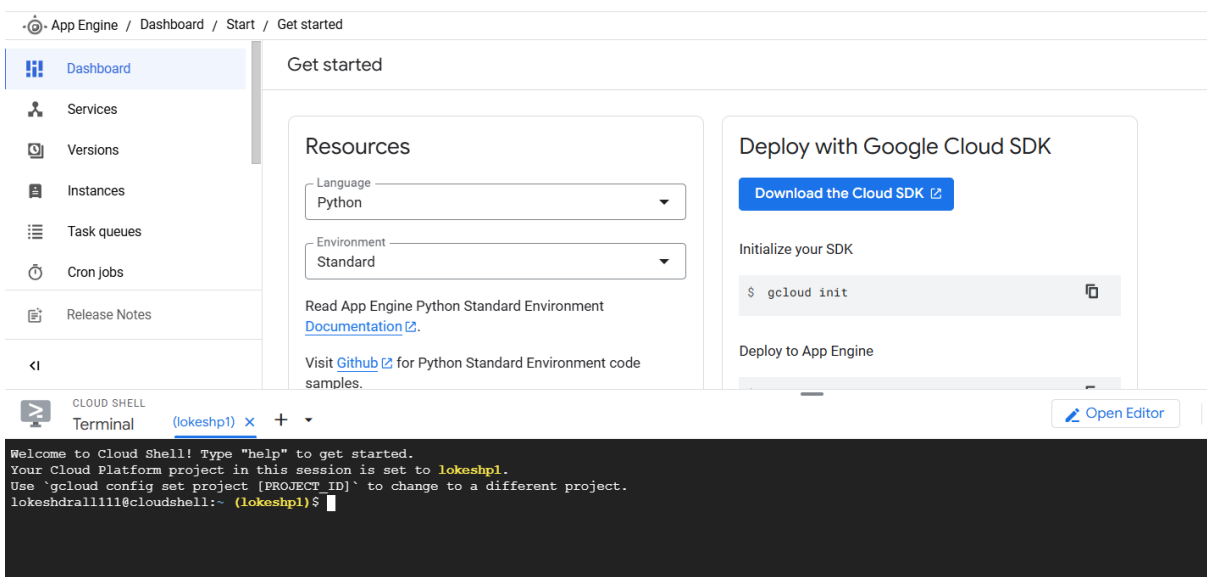


3) Select region and “App Engine default service account



After clicking Next, it will take some time, and once it reaches the next screen, go to next step from there on

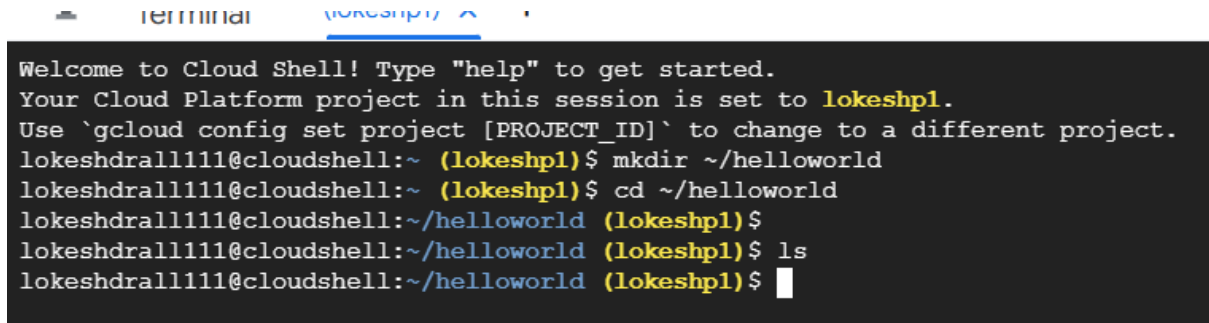
- 4) Now extract files from the "hello standardappengine" zip file attached in this module folder.
- 5) Do not change anything else on the screen, just open Cloud Shell in GCP.



- 6) Now, first we will test our code which we extracted from the folder, for that follow below
- 7) (Optional) Open the folder with these files in vscode, only go through the code

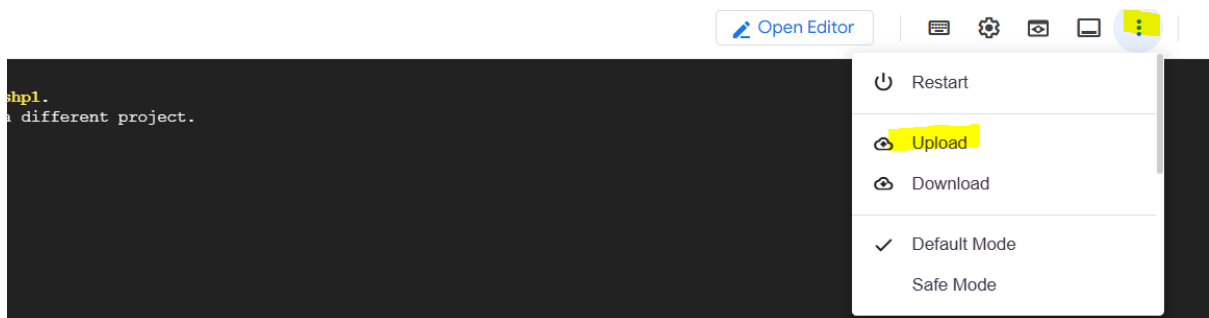
- 8) In CLI, make a new directory, type `~/helloworld`, then go to this directory by typing `cd ~/helloworld`

Now, if you do `ls` (listing of objects) there, it will not show anything because it is a blank directory.

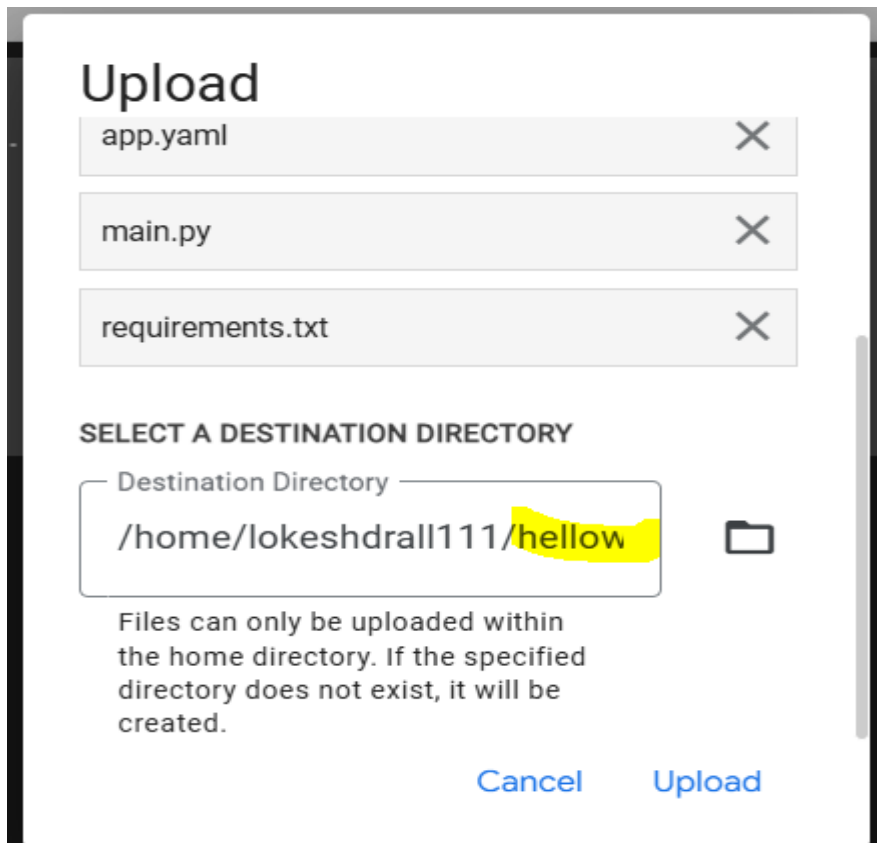


```
terminal (lokeshp1)
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to lokeshp1.
Use `gcloud config set project [PROJECT_ID]` to change to a different project.
lokeshdrall111@cloudshell:~ (lokeshp1)$ mkdir ~/helloworld
lokeshdrall111@cloudshell:~ (lokeshp1)$ cd ~/helloworld
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$ ls
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$
```

- 9) Upload our scripts to cli, for this, click on the three dots and click upload



- 10) Select all three files, and in the destination, add the folder name helloworld.
destination will look like `/home/lokeshdrall111/helloworld`



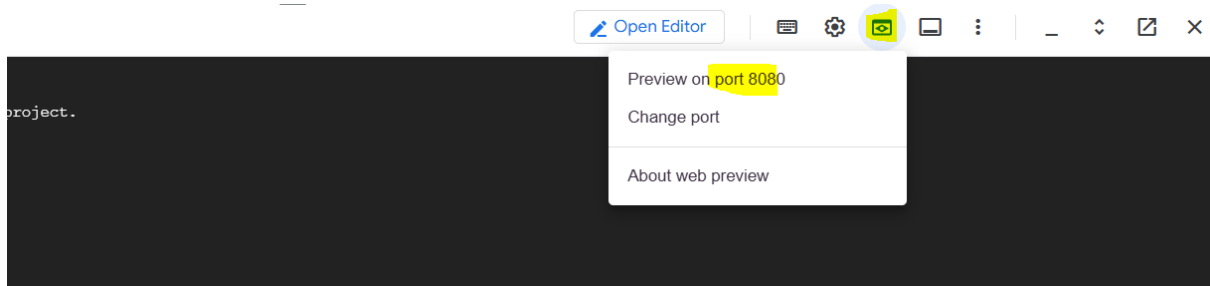
11) Again, do ls and should be able to see the files

```
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$  
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$ ls  
app.yaml  main.py  requirements.txt  
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$
```

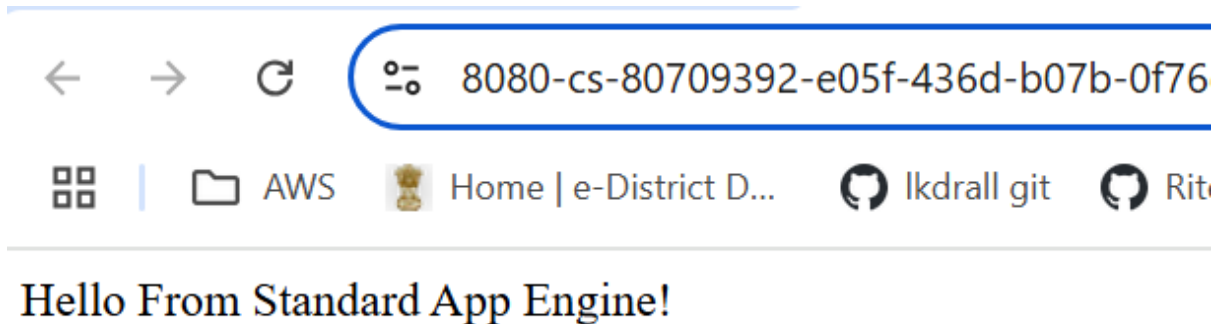
12) Run command python main.py (this should show like below, services running)

```
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$ python main.py  
* Serving Flask app 'main'  
* 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://127.0.0.1:8080  
* Running on http://10.88.0.4:8080  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 851-635-271
```

13) Click the webpreview button and preview on port 8080



14) This will open a new webpage showing the app is running



Note, so far, we have just tested that our code is working fine.

15) Ctrl+c on CLI to stop the session

16) Now to deploy the app, type gcloud app deploy in cli (make sure you are in helloworld directory)

```
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$ gcloud app deploy
Services to deploy:

descriptor:      [/home/lokeshdrall111/helloworld/app.yaml]
source:          [/home/lokeshdrall111/helloworld]
target project:  [lokeshp1]
target service:  [default]
target version:  [20250114t120637]
target url:      [https://lokeshp1.uc.r.appspot.com]
target service account: [lokeshp1@appspot.gserviceaccount.com]

Do you want to continue (Y/n)?
```

17) Type y and hit enter

18) This will take some time to complete. in case you see any error like not access to bucket, run the same command again

19) Once complete successfully, it will show the deployed service URL.

```

Do you want to continue (Y/n)? y

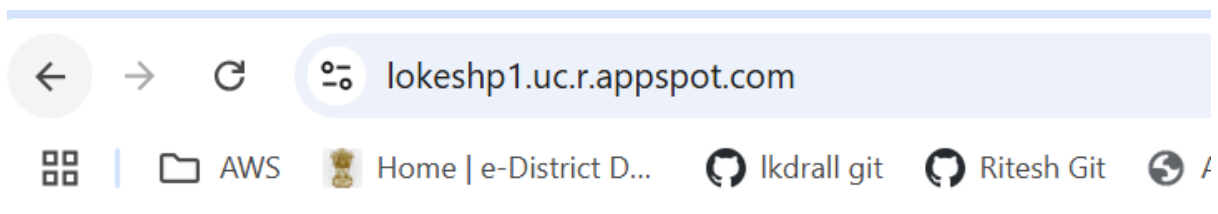
Beginning deployment of service [default]...
Uploading 0 files to Google Cloud Storage
100%
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://lokeshp1.uc.r.appspot.com]

You can stream logs from the command line by running:
$ gcloud app logs tail -s default

To view your application in the web browser run:
$ gcloud app browse
lokeshdrall111@cloudshell:~/helloworld (lokeshp1)$

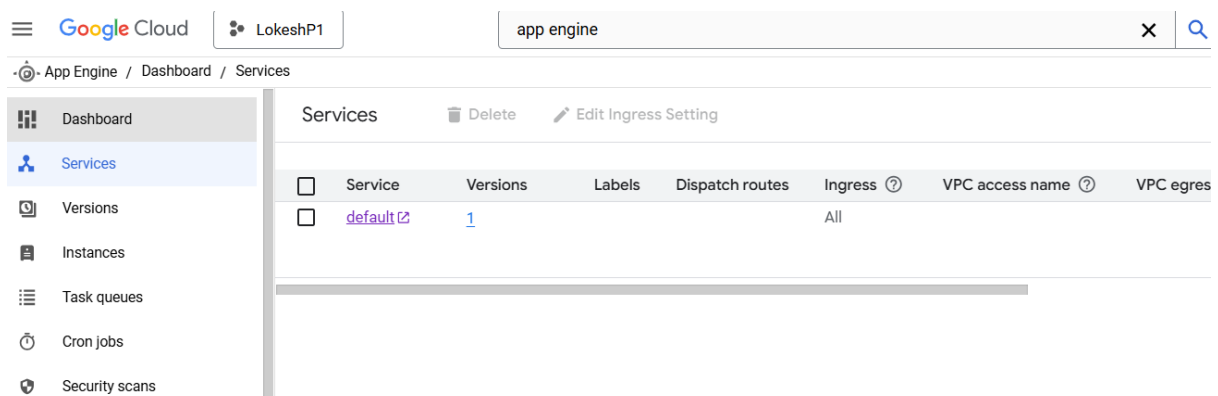
```

20) Copy the URL and open it in a browser, the webpage should open.



Hello From Standard App Engine!

Now just check and explore options under Services and versions got created related to our application.



Google Cloud

LokeshP1

app engine

Search

App Engine / Dashboard / Versions

Dashboard

Services

Versions

Instances

Task queues

Cron jobs

Security scans

Firewall rules

Quotas

Versions

Refresh

Delete

Stop

Start

Migrate traffic

Split traffic

Filter

Filter versions

<input type="checkbox"/>	Version	Status	Traffic Allocation	Instances	Runtime	Runtime Lifecycle	En
<input type="checkbox"/>	20250114t120858	Serving	<div></div> 100%	0	python311	General Availability (GA)	St