

UE23CS351B - CLOUD COMPUTING

LAB 1

What is cloud computing(CC)?

In simple terms, it is like having your computer on the internet. Whatever you can do on your PC can be done on the cloud, unlocking a treasure chest of possibilities.

If cloud computing is the same as having a personal computer then why use cloud computing?

The PC you use often will have 8 to 16 GB's of RAM , possibly more and a single processor. This works well for personal use, but fails when you have to scale and serve millions of users . In order to do that you would need a Chonky System , Powerful processors , High capacity Ram , High speed storage , GPU's etc. Well building it yourself would be good if you were a crypto bro and earning millions but realistically speaking it's not feasible for a single person to do and maintain it , hence we offload all our troubles to cloud providers which massively helps us .Along the course you will learn the advantages and disadvantages of using cloud services .

This document provides an overview of what can be achieved with cloud computing. Before diving into practical examples, we will first explore the different models of cloud services Prerequisite for this lab is a [github](#) account and some Coffee.

Services available to us

Iaas Infrastructure as a service

Here the cloud providers offer the infrastructure for you to work on. You will need to configure your resources such as ram, cpus, operating system [maybe more]. Basically you will get to customize your PC on the internet and use it however you want. A good example for that is an EC2 instance of AWS.

Paas Platform as a service

Here the cloud providers hide away the hardware configuration and most of the other stuff, and just provide the service that you would want, for example you would need a database. Well, here you don't have to worry about allocating ram, disk space, or networking. You would need to worry about the schema of the tables, the constraints and stuff.

Saas Software as a service

Here they directly provide you the service, no need to worry about configuring anything. Just directly use it. Good example would be your gmail, google drive etc. Sign up on

Practice Time...

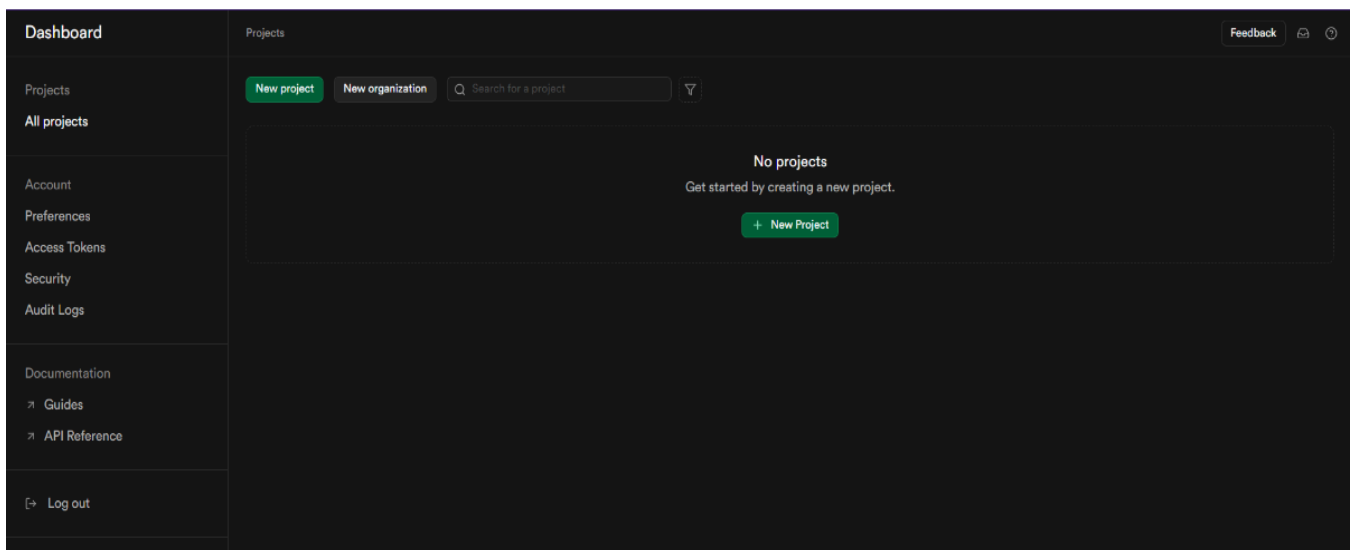
A) IAAS - Infrastructure Backend as a service

One of the widely used cloud services is object store, so unlike your traditional file system where data is stored as files and folders here you store data as blobs or objects, these objects are associated with a variable amount of metadata, and a globally unique identifier. Object storage systems allow retention of massive amounts of unstructured data in which data is written and read.

Here for object store we are going to use Supabase, supabase is open source baas (backend as a service), it provides a lot of free services such as Authentication, postgresql database, realtime database and object store.

1. Sign up for Supabase account <https://supabase.com/>

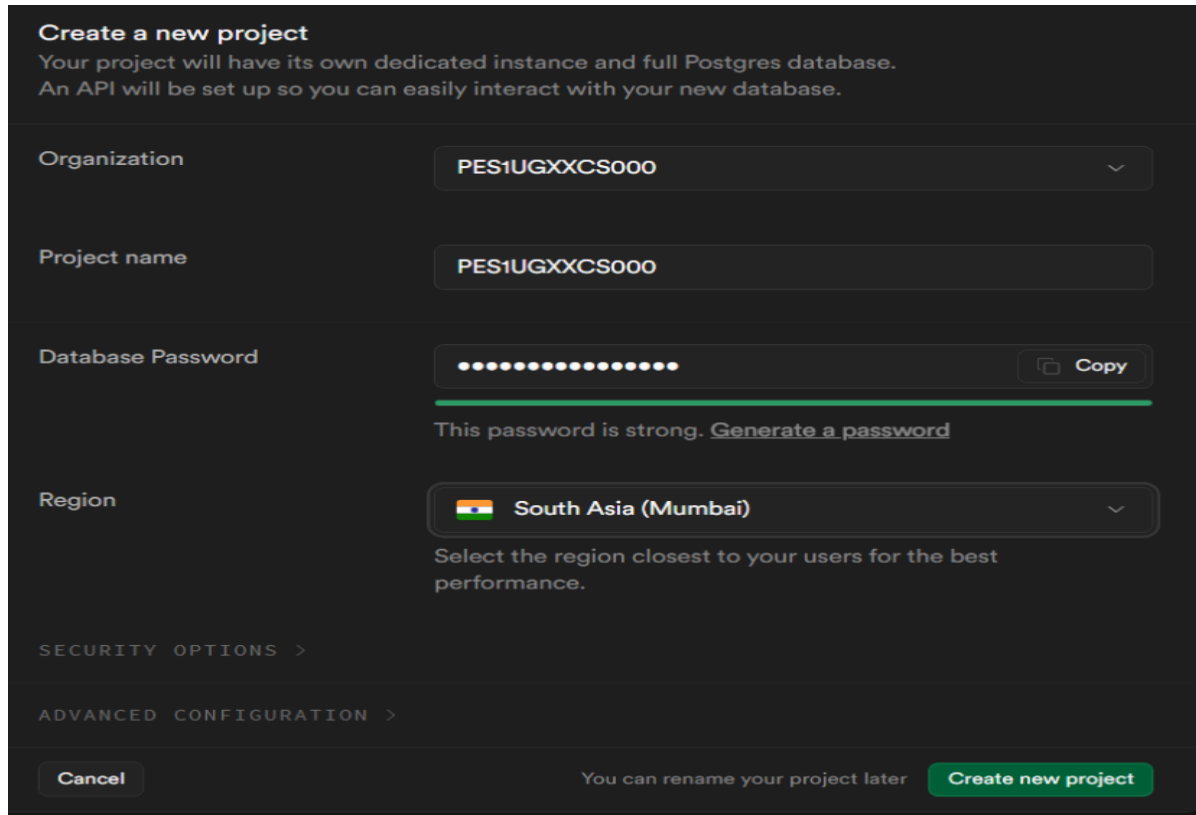
Once you do, you should see a page like this



2. When you create a new project for the first time it will ask you to create an organization, go ahead and do it under personal and free tier.

The image shows the 'Create a new organization' form. It includes a title 'Create a new organization' and a subtitle 'This is your organization within Supabase. For example, you can use the name of your company or department.' The form has three main sections: 'Name' with a text input containing 'PES1UGXXCS000' and a label 'What's the name of your company or team?'; 'Type of organization' with a dropdown menu set to 'Personal' and a label 'What would best describe your organization?'; and 'Plan' with a dropdown menu set to 'Free - \$0/month' and a label 'Pricing' with an external link icon. At the bottom, there is a 'Cancel' button, a note 'You can rename your organization later', and a 'Create organization' button.

3. While creating new project enter the database password [ideally its is best to generate one and keep it safe], and select region as South Asia (Mumbai)



The screenshot shows the 'Create a new project' form in Supabase. At the top, it says 'Create a new project' and provides context: 'Your project will have its own dedicated instance and full Postgres database. An API will be set up so you can easily interact with your new database.' The form has several fields: 'Organization' with a dropdown menu showing 'PES1UGXXCS000'; 'Project name' with a text input field containing 'PES1UGXXCS000'; 'Database Password' with a masked password field (dots) and a 'Copy' button; a feedback message 'This password is strong. [Generate a password](#)'; 'Region' with a dropdown menu showing 'South Asia (Mumbai)' and a note 'Select the region closest to your users for the best performance.' Below these are links for 'SECURITY OPTIONS >' and 'ADVANCED CONFIGURATION >'. At the bottom, there is a 'Cancel' button, a note 'You can rename your project later', and a green 'Create new project' button.

4. Once you create the new project your screen will display something similar to the first image below. This would be the first **Screenshot(SS1) required**.

Once you create the new project you will also get the Project api Keys and Project URL [Scroll down a bit]. Copy ONLY Project URL and then go to the API Settings next API keys and then copy the SERVICE KEY paste both URL and SERVICE KEY in the `supabase_object_store.py`

PES1UGXXCSXXX

NANO

Tables0

Functions0

Replicas0

Project Status

Welcome to your new project

Your project has been deployed on its own instance, with its own API all set up and ready to use.

Build out your database

Start building your app by creating tables and inserting data. Our Table Editor makes Postgres as easy to use as a spreadsheet, but there's also our SQL Editor if you need something more.

Table Editor

SQL Editor

About Database

Explore our other products

Supabase provides all the backend features you need to build a product. You can use it completely, or just the features you need.

1

create table todos

2

id bigint generated by default as identity

3

task text,

4

status status default 'pending'

5

user_id uuid references users (id)

6

inserted_at timestamp without time zone default now()

7

updated_at timestamp without time zone default now()

8

);

	id	task	status
	1	Create a project	Complete
	2	Read documentation	Complete
	3	Build application	In progress
	4	Connect Supabase	In progress
	5	Deploy project	Not started
	6	Get users	Not started
	7	Handle errors	Not started

Connect to your project

Interact with your database through the [Supabase client libraries](#) and your API keys.

API settings

Docs

PROJECT API

Your API is secured behind an API gateway which requires an API Key for every request. You can use the parameters below to use Supabase client libraries.

Project URL

https://sozngvrpwwdiadpininix.supabase.co

Copy

A RESTful endpoint for querying and managing your database.

Publishable API Key

sb_publishable_Yd3tH7GDBfK3vVvR0IU4A_tBQnGgZ

Copy

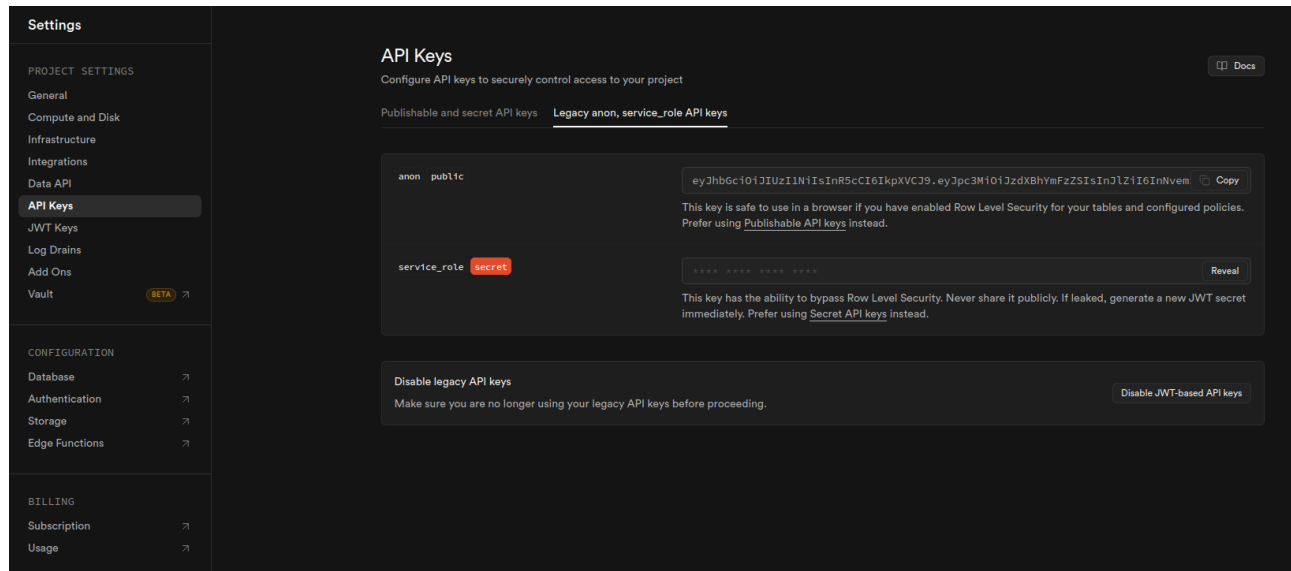
This key is safe to use in a browser if you have enabled Row Level Security (RLS) for your tables and configured policies. You may also use the secret key which can be found [here](#) to bypass RLS.

N

Choose your preferred framework

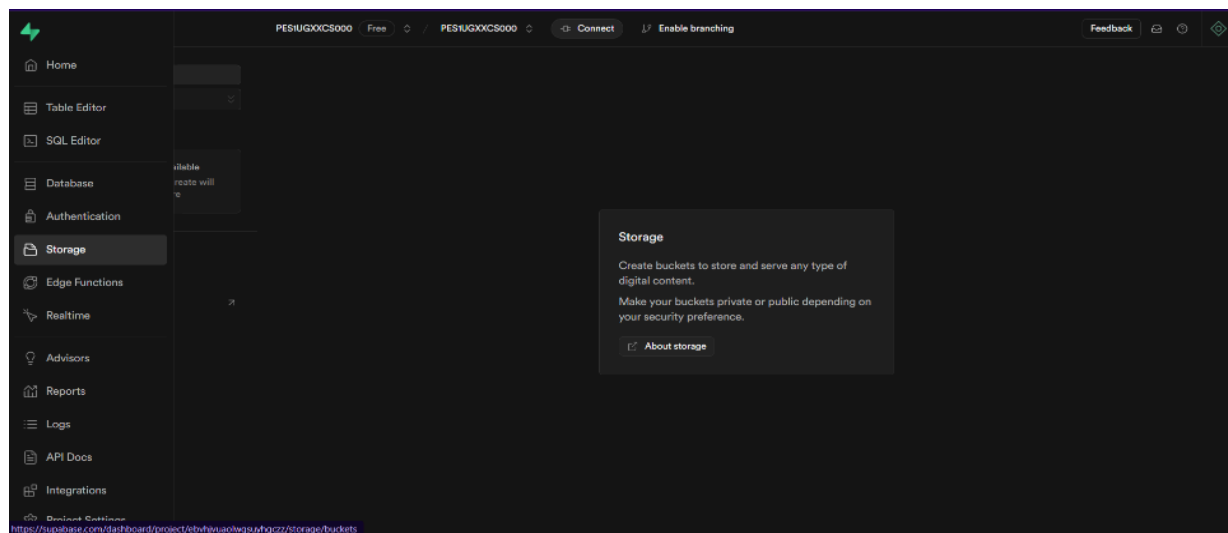
Connect to your project from a variety of frameworks, ORMs, an MCP server, or even directly via connection string.

Connect



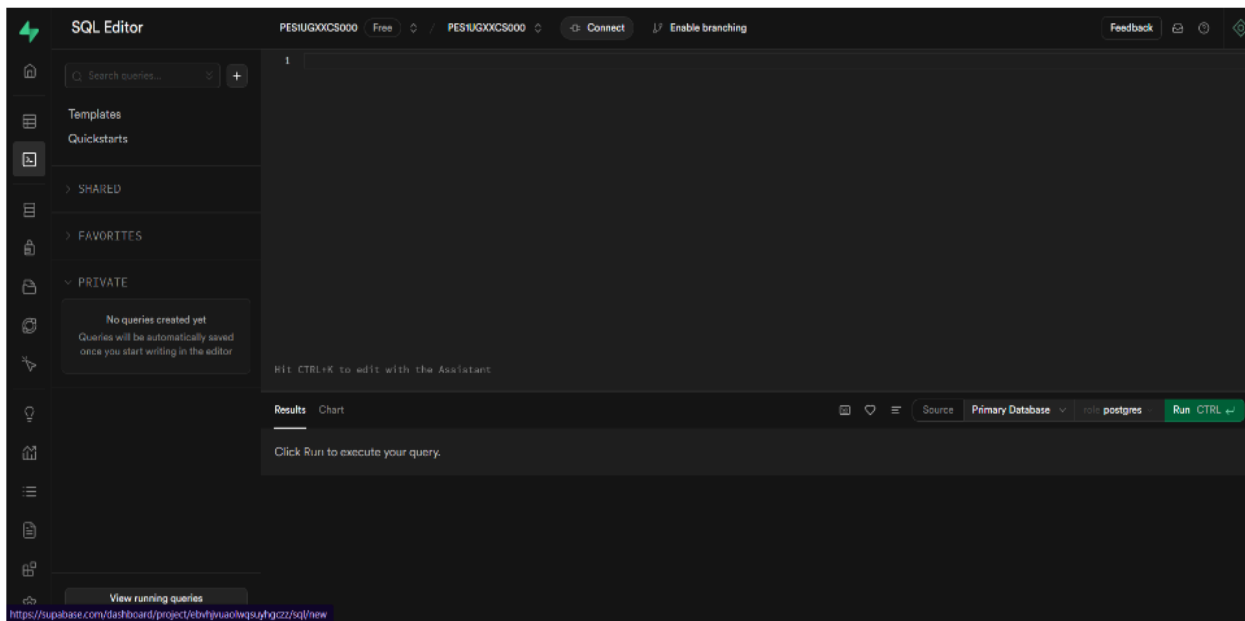
(You don't need to copy the shown api key and url, by the time you see this docs the project would have been deleted XD)

5. You can explore any services, for now we will focus on object store.



Well here you can create a bucket, and store data by uploading them by using the ui, but since that is not suitable for large projects, We will do it through automated script.

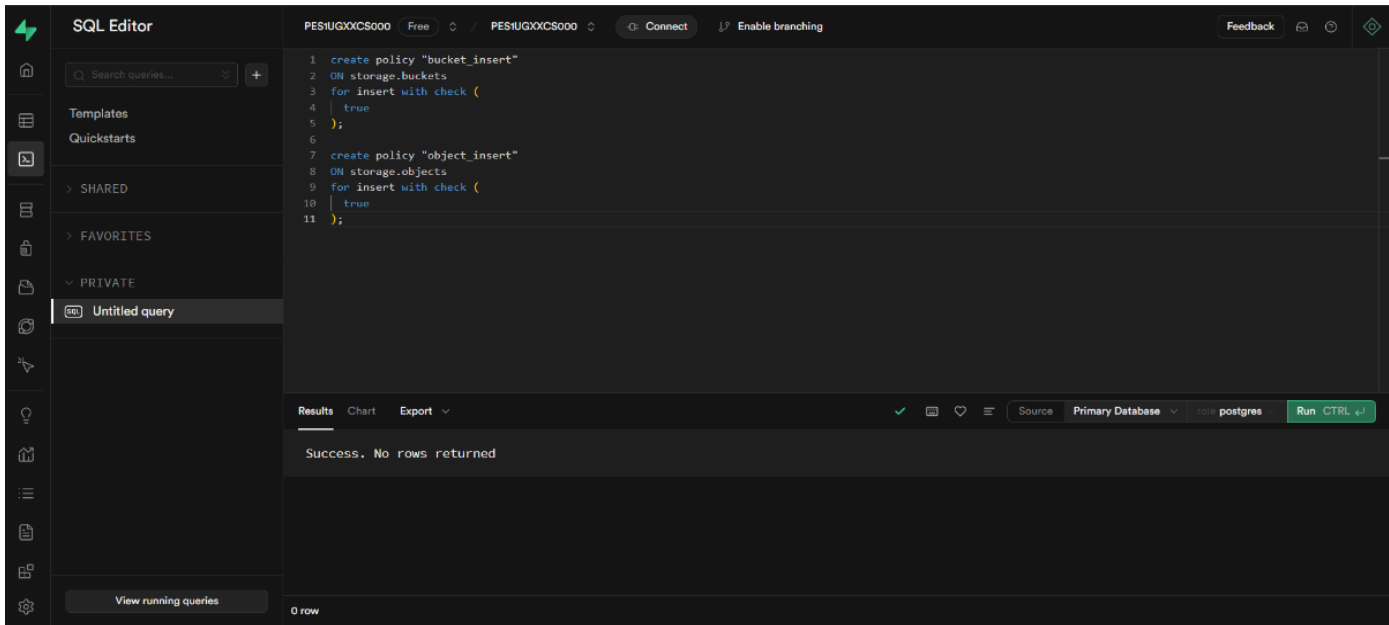
6. Before we jump to python script there is one thing we need to do, that is we need to tell supabase to allow api calls to allow operations on object storage. For that we need to move to SQL Editor.



7. On this you need to run two sql queries. Once you click Run, take the **Screenshot(SS2, Similar to the second image under this step)** indicating the successful execution of those 2 queries.

```
create policy "bucket_insert"
ON storage.buckets
for insert with check (
true
);
create policy "object_insert"
ON storage.objects
for insert with check (
true
);
```

[Well you see most of the permission handling on supabase is done through postgresql database, these 2 queries modify the table to allow insert operations on object store by api calls.]



8. For the supabase api, you would need to install the supabase module for python.

```
pip install supabase
```

9. Next open up the **supabase_object_store.py** file, make sure you added servicei key and project url. You can change the bucket name or the image path.

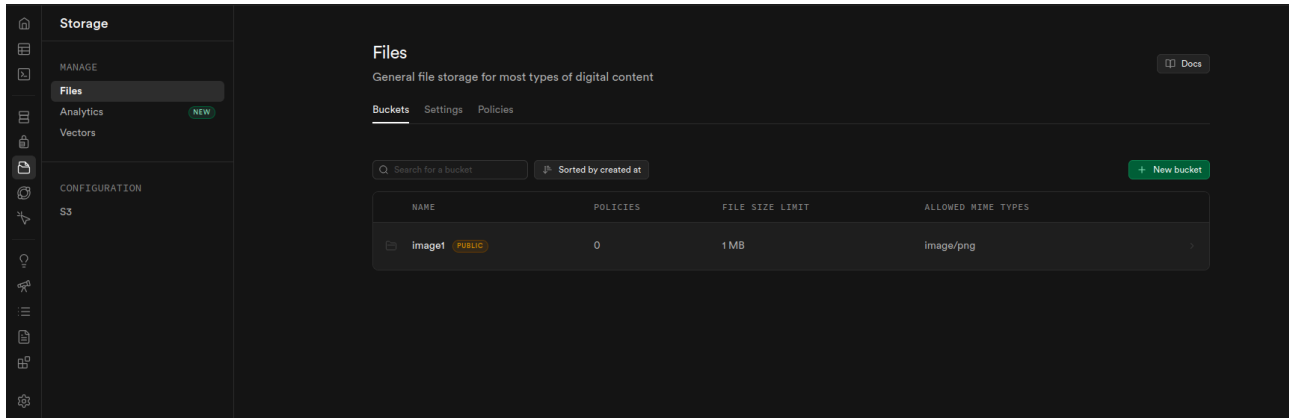
10. Upload your image in jpeg format in the same folder where you have all other files and rename your image to your PES1UGXXCSXXX.jpeg

11. Run each step individually by commenting and uncommenting the other step. Code is extensively commented so that you understand what each part of the code does. At the end you should see a bucket created, your image uploaded, and a public url to view the image. (Note you can upload the portfolio image here, and can use the public url for the upcoming practical . The **Screenshots** and **Links** required for evaluation are mentioned below.

- SS3 - Successful bucket creation

(uncomment step 1 and comment step 2 and 3)

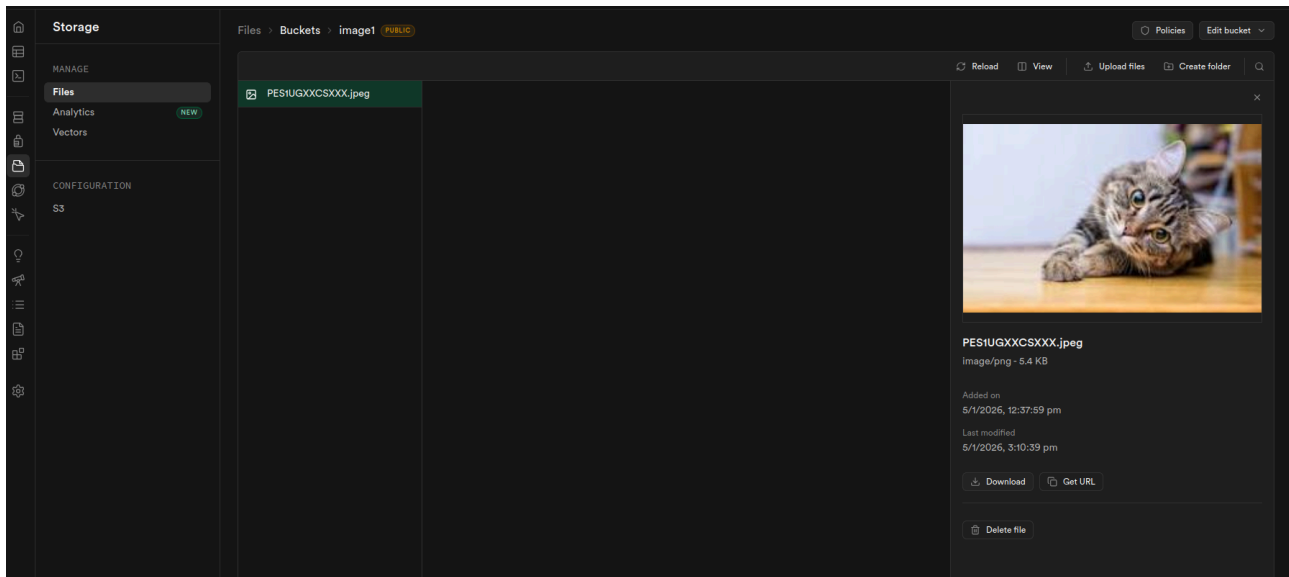
```
ccbd@ccbd-ThinkStation-P620:~/Desktop/cc26/CC-Lab1$ python3 supabase_object_store.py
Storage endpoint URL should have a trailing slash.
Bucket 'image1' created successfully.
ccbd@ccbd-ThinkStation-P620:~/Desktop/cc26/CC-Lab1$
```



- SS4 - Successful image upload

(uncomment step 2 and comment step 1 and 3)

```
ccbd@ccbd-ThinkStation-P620:~/Desktop/cc26/CC-Lab1$ python3 supabase_object_store.py
Storage endpoint URL should have a trailing slash.
Image uploaded successfully: UploadResponse(path='PES1UGXXCSXXX.jpeg', full_path='image1/PES1UGXXCSXXX.jpeg', fullPath='image1/PES1UGXXCSXXX.jpeg')
ccbd@ccbd-ThinkStation-P620:~/Desktop/cc26/CC-Lab1$
```



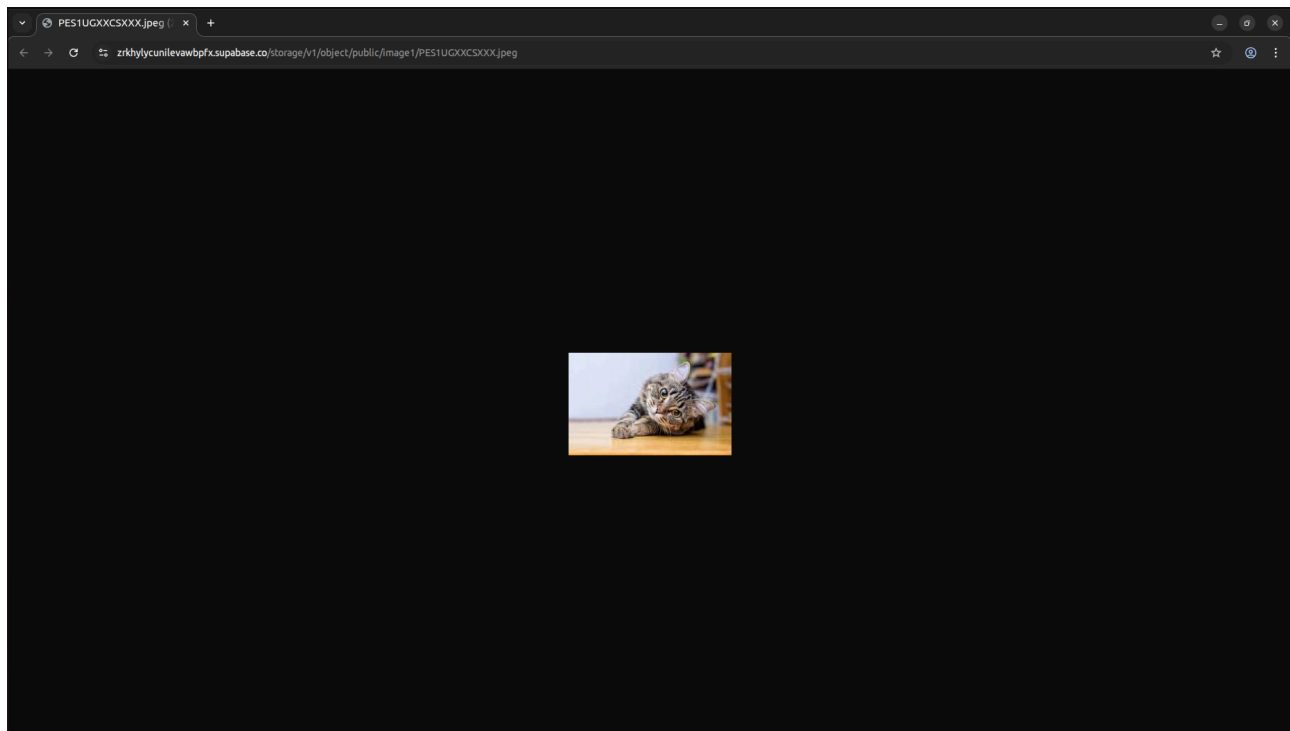
(uncomment step 3 and comment step 1 and 2)

```
ccbd@ccbd-ThinkStation-P620:~/Desktop/cc26/CC-Lab1$ python3 supabase_object_store.py
Storage endpoint URL should have a trailing slash.
Public URL of the image: https://zrkhylycunilevawbpfx.supabase.co/storage/v1/object/public/image1/PES1UGXXCSXXX.jpeg
ccbd@ccbd-ThinkStation-P620:~/Desktop/cc26/CC-Lab1$
```

- Paste the public URL of the image obtained from running step4 of the python file given to you.

- Example :

`https://lvxhhflqjupwgezviugh.supabase.co/storage/v1/object/public/images/cat.jpeg`



B) PaaS - Platform as a Service - Deployment of a Flask Application using PythonAnywhere

Platform as a Service (PaaS) provides a cloud-based environment where users can develop, deploy, and run applications without managing the underlying infrastructure. The cloud provider handles servers, storage, and runtime configuration, allowing developers to focus only on application code.

In this experiment, **PythonAnywhere** is used as the PaaS platform to deploy a simple **Flask-based quote generator application** directly from the browser. This activity demonstrates how applications can be hosted and executed on the cloud using Platform as a Service.

Step 1: Create a PythonAnywhere Account

1. Open the PythonAnywhere website using the following link:
<https://www.pythonanywhere.com/>

The screenshot shows the PythonAnywhere website's 'Plans and pricing' page. At the top, there's a navigation bar with links for 'Send feedback', 'Forums', 'Help', 'Blog', 'Pricing & signup', and 'Log in'. The main heading is 'Plans and pricing'. Below this, there are two highlighted boxes: 'Beginner: Free!' and 'Education accounts'. The 'Beginner: Free!' box describes a limited account with one web app and a 'Create a Beginner account' button. The 'Education accounts' box asks if the user is a teacher and provides a link to 'Education beta'. Below these, a green box states that all paid plans come with a 30-day money-back guarantee. At the bottom, there is a table with four columns representing different plans: Hacker (\$5/month), Web dev (\$12/month), Startup (\$99/month), and Custom (\$5 to \$500/month). Each plan column lists its features, such as the number of web apps, CPU-seconds per day, and console access. The URL 'www.pythonanywhere.com/registration/register/beginner/' is visible at the bottom left of the page.

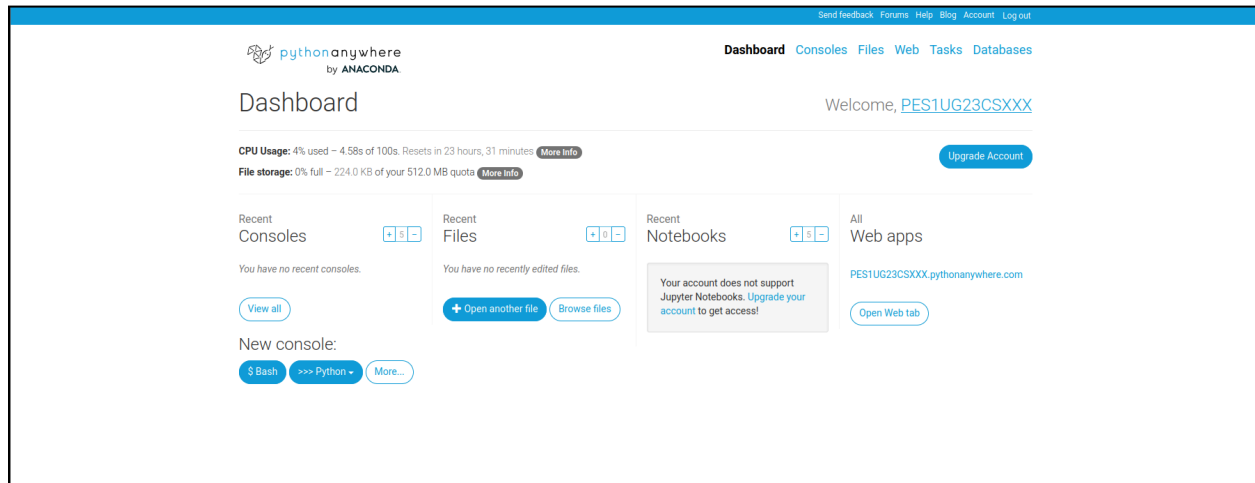
Hacker	\$5/month	Web dev	\$12/month	Startup	\$99/month	Custom	\$5 to \$500/month
Run your Python code in the cloud from one web app and the console		If you want to host small Python-based websites for you or for your clients		Start a business and don't worry about having to scale to handle traffic spikes		Want a combination that's not on the list? Create your own! All custom plans have:	
A Python IDE in your browser with unlimited Python/bash consoles		A Python IDE in your browser with unlimited Python/bash consoles		A Python IDE in your browser with unlimited Python/bash consoles		A Python IDE in your browser with unlimited Python/bash consoles	
One web app on a custom domain or your-username.pythonanywhere.com		Up to 2 web apps on custom domains or your-username.pythonanywhere.com		Up to 3 web apps on custom domains or your-username.pythonanywhere.com		Up to 20 web apps, on custom domains or your-username.pythonanywhere.com	
Enough power to run a typical 100,000 hit/day website. (more info)		Enough power to run a typical 150,000 hit/day website on each web app. (more info)		Enough power to run a typical 1,000,000 hit/day website on each web app. (more info)		As many web workers as you need to scale your site's capacity. (more info)	
2,000 CPU-seconds per day for consoles, scheduled tasks and always-on tasks (more info)		4,000 CPU-seconds per day for consoles, scheduled tasks and always-on tasks (more info)		10,000 CPU-seconds per day for consoles, scheduled tasks and always-on tasks (more info)		Up to 100,000 CPU-seconds per day for consoles, scheduled tasks and always-on tasks (more info)	

2. Click on **Create a new account**.
3. Use your **SRN (e.g., PES1UG23CSXXX)** as the **username**.
4. Enter your email ID and set a password of your choice.

Step 2: Verify Email and Access Dashboard

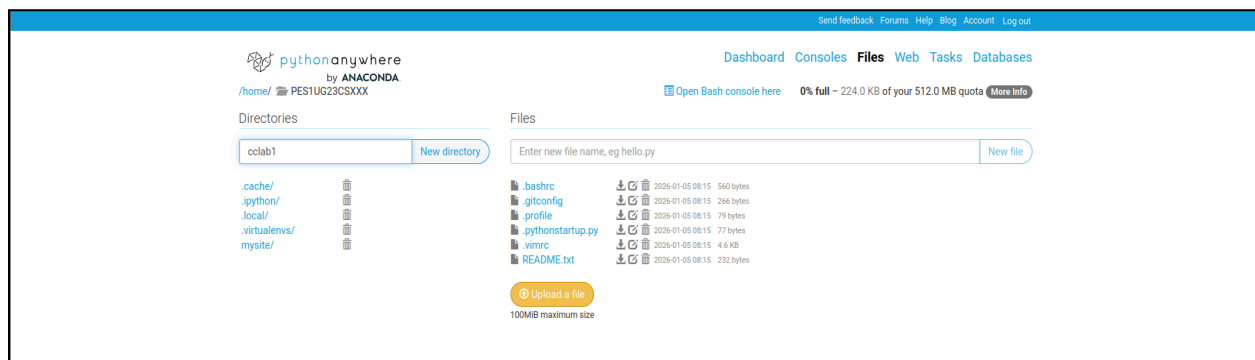
1. Open your registered email inbox.
2. Verify your email ID by clicking on the verification link sent by PythonAnywhere.

3. After verification, you will be redirected to the PythonAnywhere dashboard.
4. **Upload a screenshot (SS5) of your dashboard with your SRN clearly visible.**



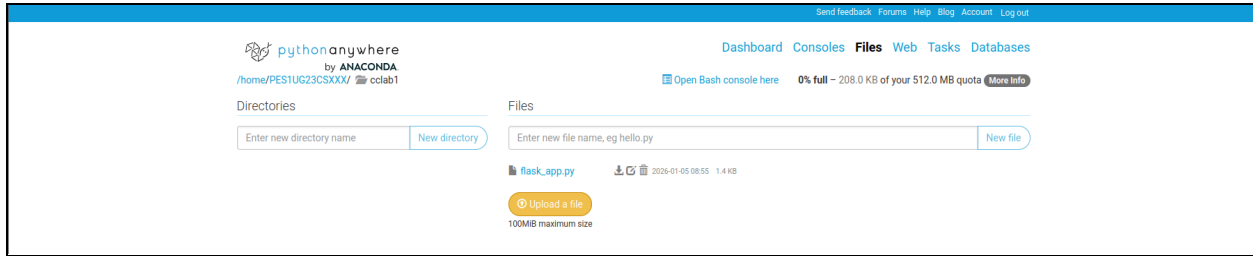
Step 3: Create Project Directory

1. From the dashboard, click on the **Files** tab.
2. Create a new directory named **cclab1**.



Step 4: Upload Flask Application File

1. Open the **cclab1** directory.
2. Upload the file **flask_app.py** provided to you.
3. Ensure your **SRN is visible on the top-left corner** of the screen.
4. **Upload a screenshot (SS6) as shown below with your SRN clearly visible.**



Step 5: Configure Web App Source Code Path

1. Click on the **Web** tab located at the top-right corner.
2. Scroll down to the **Code** section as shown below.
3. Update the **Source code** path to:
`/home/PES1UG23CSXXX/cc1ab1`

Traffic:
How busy is your site?

This month (previous month)	24	(0)
Today (yesterday)	24	(0)
Hour (previous hour)	24	(0)

Want some more data? [Paying accounts](#) get pretty charts ;-)

Code:
What your site is running.

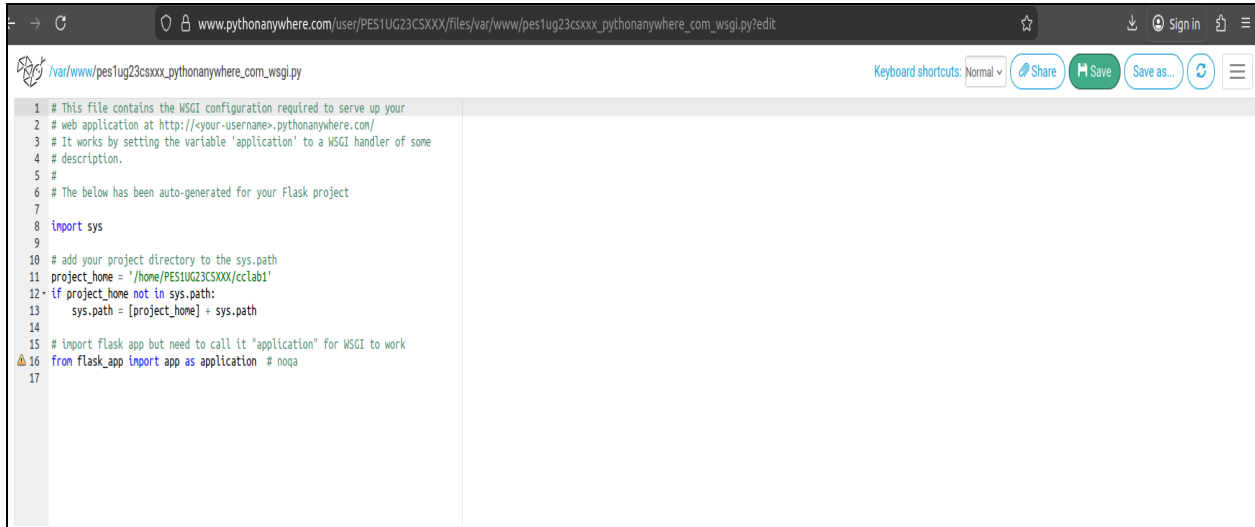
Source code:	<input type="text" value="/PES1UG23CSXXX/cc1ab1"/>	<input checked="" type="radio"/> <input type="radio"/>	Go to directory
Working directory:	/home/PES1UG23CSXXX/		Go to directory
WSGI configuration file:	/var/www/pes1ug23csxxx-pythonanywhere_com_wsgi.py		
Python version:	3.12		

Virtualenv:
Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

[Enter path to a virtualenv, if desired](#)

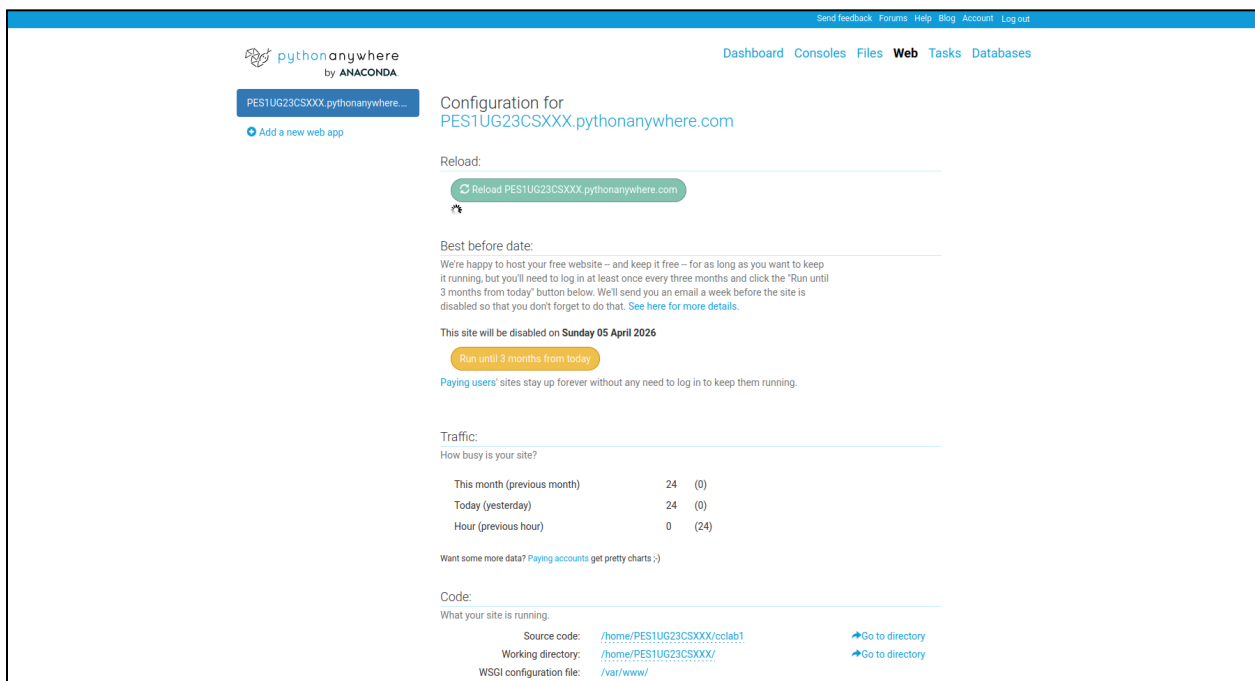
Step 6: Update WSGI Configuration File

1. Go to the **WSGI configuration file** and click on the link to edit it:
`/var/www/pes1ug23csxxx-pythonanywhere_com_wsgi.py`
2. Click on the file to edit it.
3. Modify the `project_home` path as shown below:
`project_home = '/home/PES1UG23CSXXX/cc1ab1'`



```
1 # This file contains the WSGI configuration required to serve up your
2 # web application at http://<your-username>.pythonanywhere.com/
3 # It works by setting the variable 'application' to a WSGI handler of some
4 # description.
5 #
6 # The below has been auto-generated for your Flask project
7
8 import sys
9
10 # add your project directory to the sys.path
11 project_home = '/home/PES1UG23CSXXX/cc1ab1'
12 if project_home not in sys.path:
13     sys.path = [project_home] + sys.path
14
15 # import flask app but need to call it "application" for WSGI to work
16 from flask_app import app as application # noqa
17
```

4. Save the file.
5. Go back to the **Web** tab and click on **Reload**.



pythonanywhere by ANACONDA

Dashboard Consoles Files **Web** Tasks Databases

PES1UG23CSXXX.pythonanywhere.com

Add a new web app

Configuration for PES1UG23CSXXX.pythonanywhere.com

Reload:

Reload PES1UG23CSXXX.pythonanywhere.com

Best before date:

We're happy to host your free website – and keep it free – for as long as you want to keep it running, but you'll need to log in at least once every three months and click the 'Run until 3 months from today' button below. We'll send you an email a week before the site is disabled so that you don't forget to do that. [See here for more details.](#)

This site will be disabled on **Sunday 05 April 2026**

Run until 3 months from today

Paying users' sites stay up forever without any need to log in to keep them running.

Traffic:

How busy is your site?

This month (previous month)	24	(0)
Today (yesterday)	24	(0)
Hour (previous hour)	0	(24)

Want some more data? [Paying accounts](#) get pretty charts :)

Code:

What your site is running.

Source code:	/home/PES1UG23CSXXX/cc1ab1	Go to directory
Working directory:	/home/PES1UG23CSXXX/	Go to directory
WSGI configuration file:	/var/www/	

Step 7: Access the Deployed Application

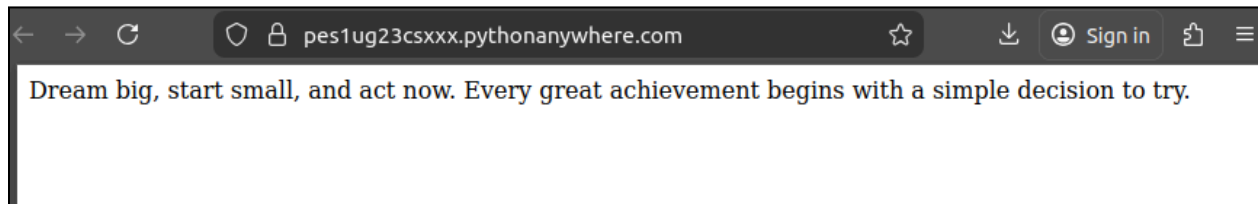
1. Click on the application link displayed under the **Web** tab configuration section.

Configuration for [PES1UG23CSXXX.pythonanywhere.com](https://pes1ug23csxxx.pythonanywhere.com)

Reload:

 Reload PES1UG23CSXXX.pythonanywhere.com

2. The link will open a new page displaying a quote.
3. Reload the page multiple times to observe different quotes being generated dynamically.
4. **Upload a screenshot (SS7)** as shown below **with the URL** showing a quote displayed on the webpage.



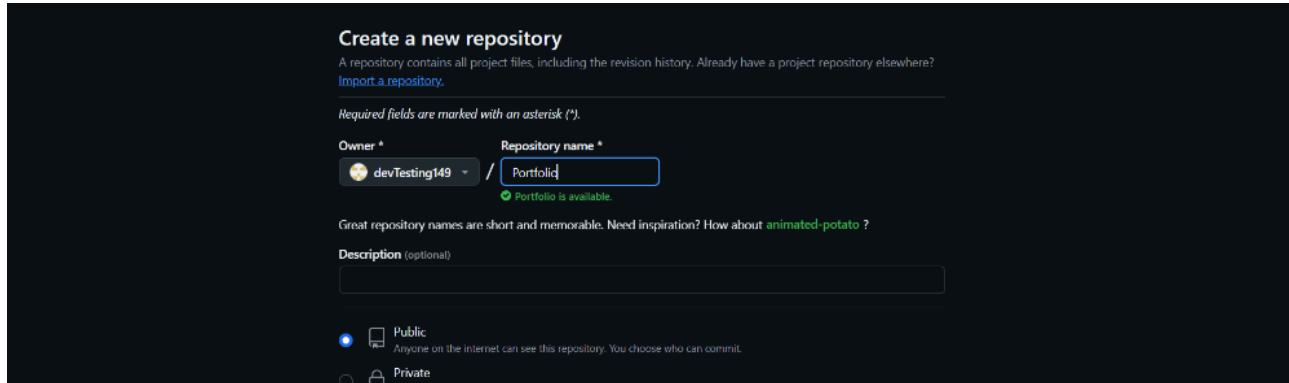
Result

The Flask application has been successfully deployed using PythonAnywhere, demonstrating **Platform as a Service (PAAS)** by hosting and running a web application without managing the underlying server infrastructure.

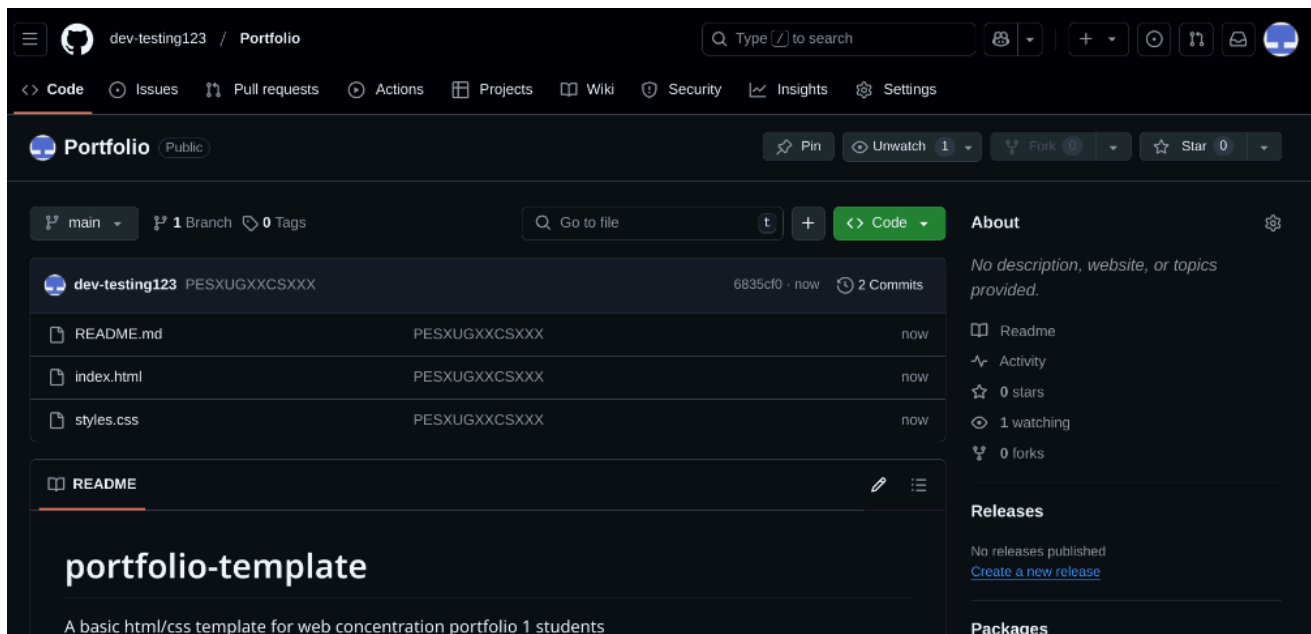
C) Software Application deployment

One of the most important uses of the cloud is to deploy an application so it reaches millions of users. For this we will deploy a portfolio website.

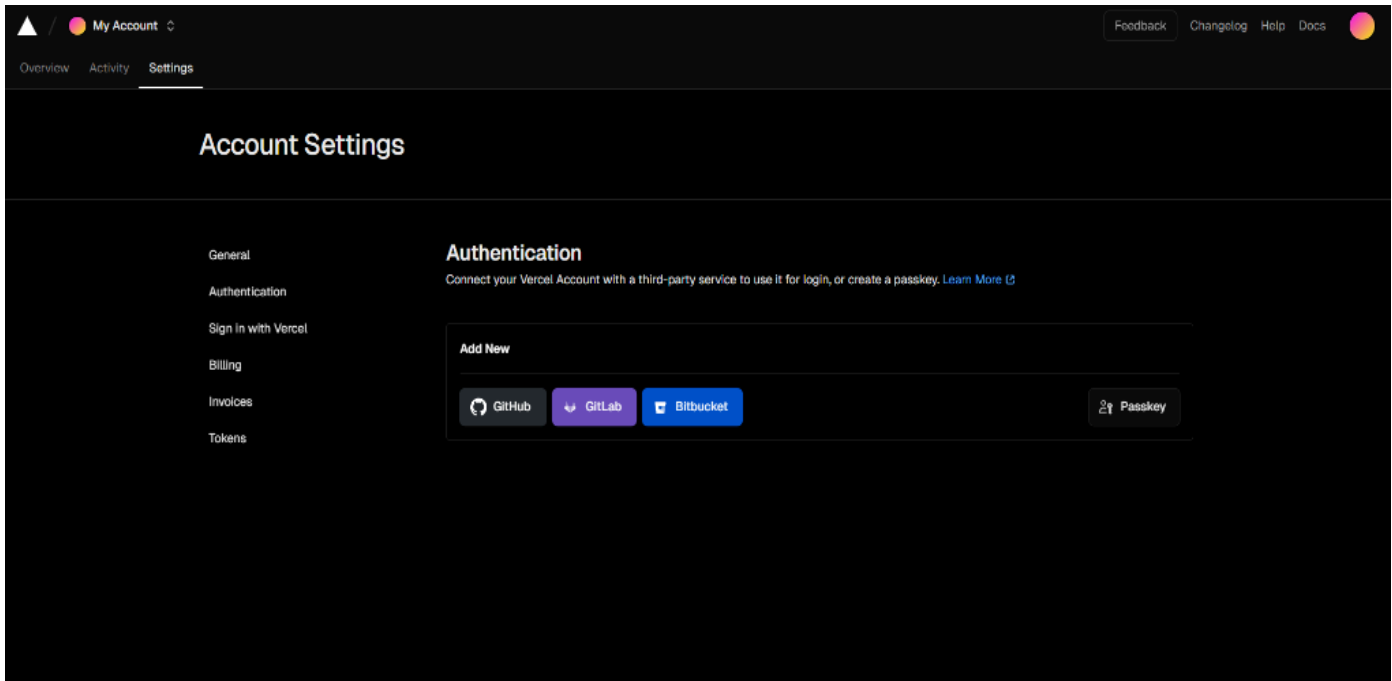
1. Customise the **portfolio-template/index.html** file on your liking (you can use the public url of the image here)
2. Once done make a github repository and keep your repository public.



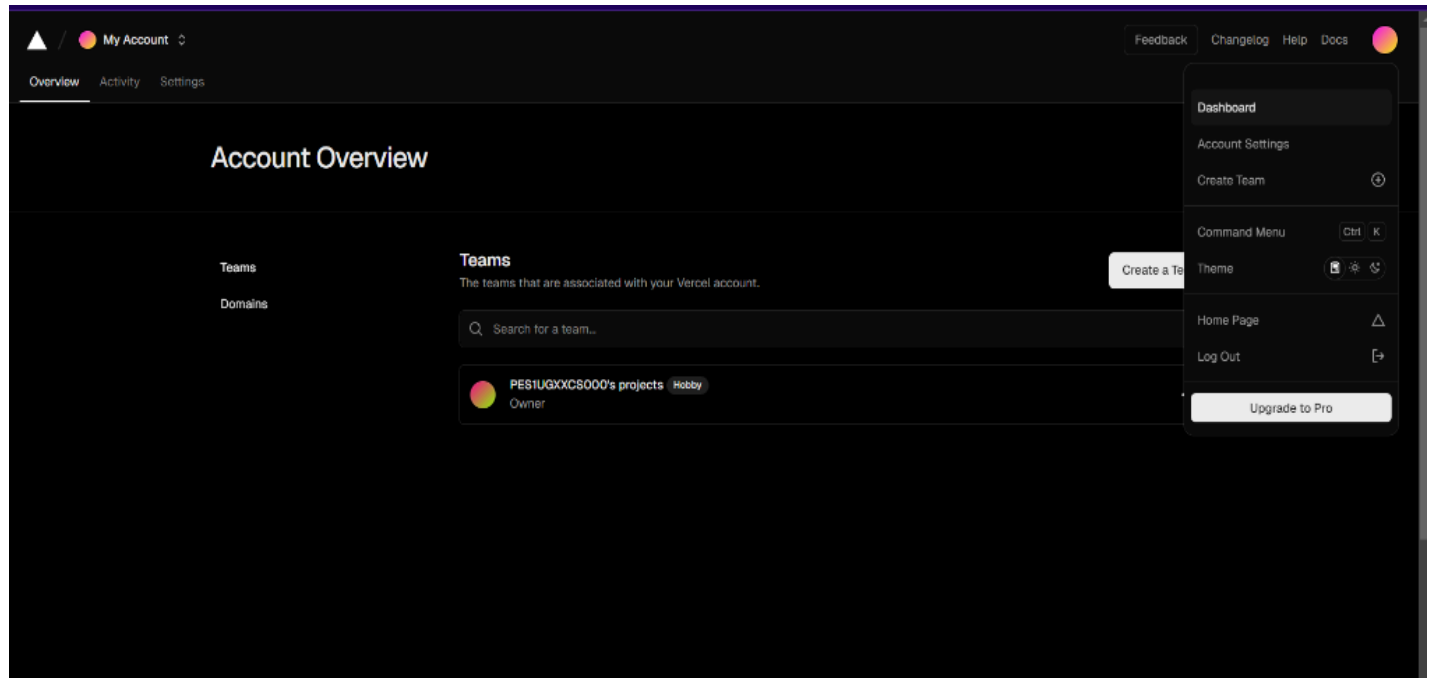
3. Upload your portfolio code to this repository using git or manually uploading them. **NOTE MAKE SURE THE MESSAGE ON THE COMMIT IS YOUR SRN.** Take a Screenshot(SS8).



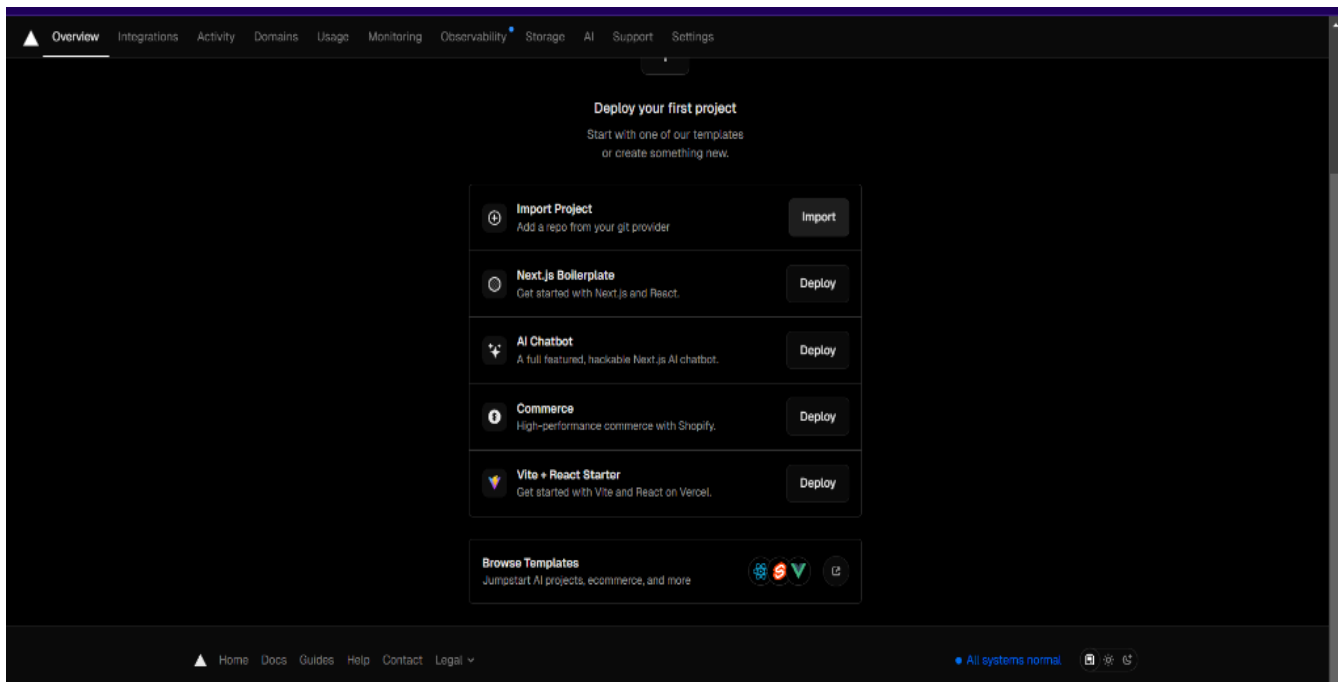
4. Next we are going to signup for vercel <https://vercel.com/login>, ideally use the same github account.



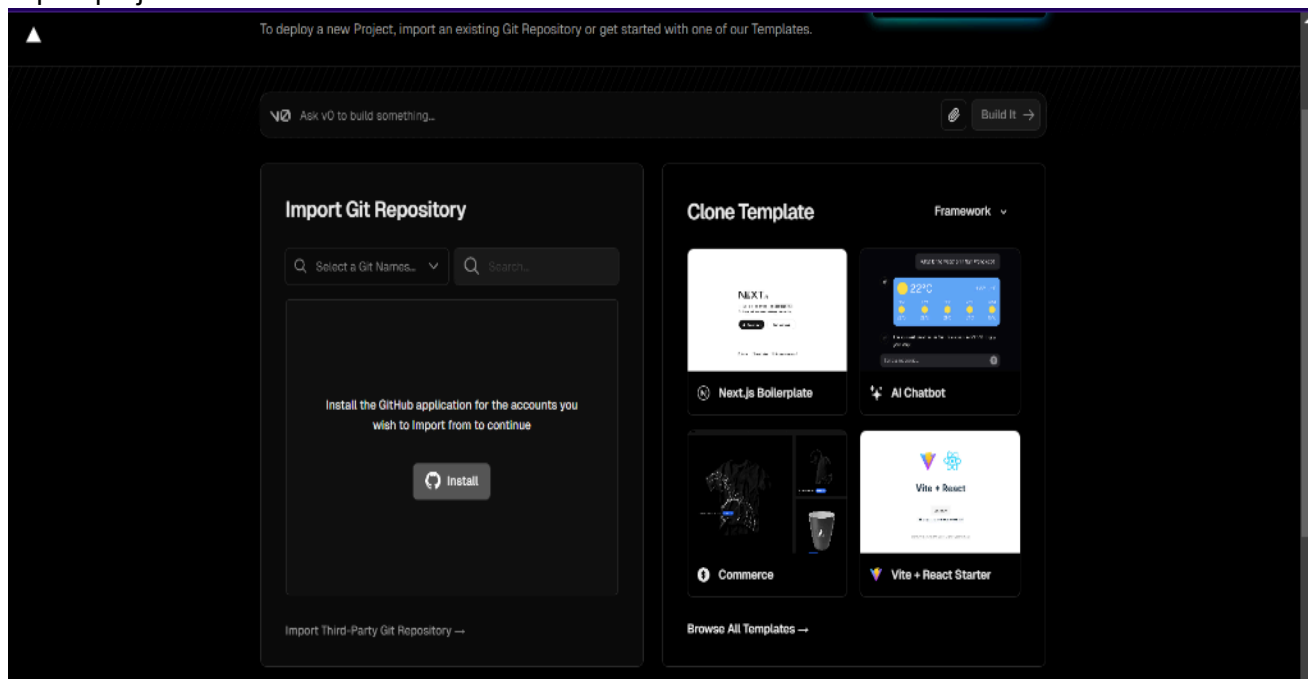
(if you use different email, then you need to connect to the GitHub explicitly)
5. Now we will import project from github and deploy



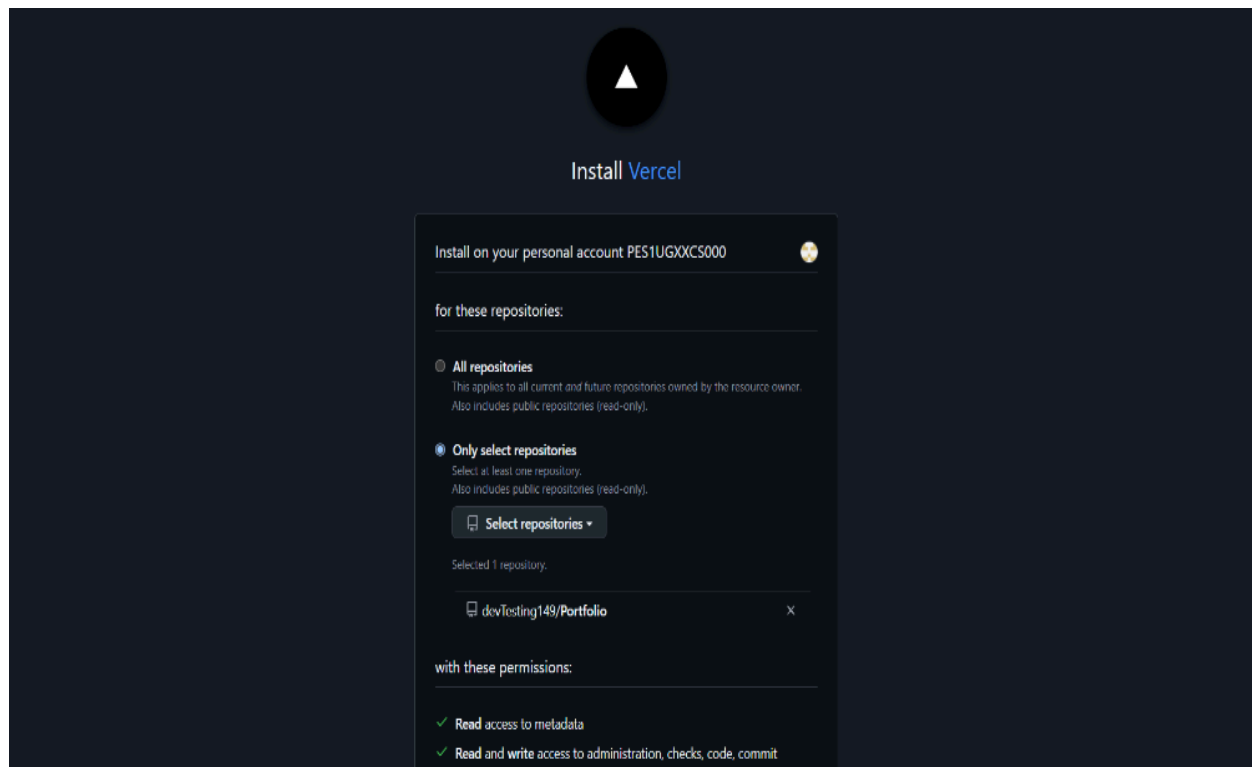
Go to dashboard



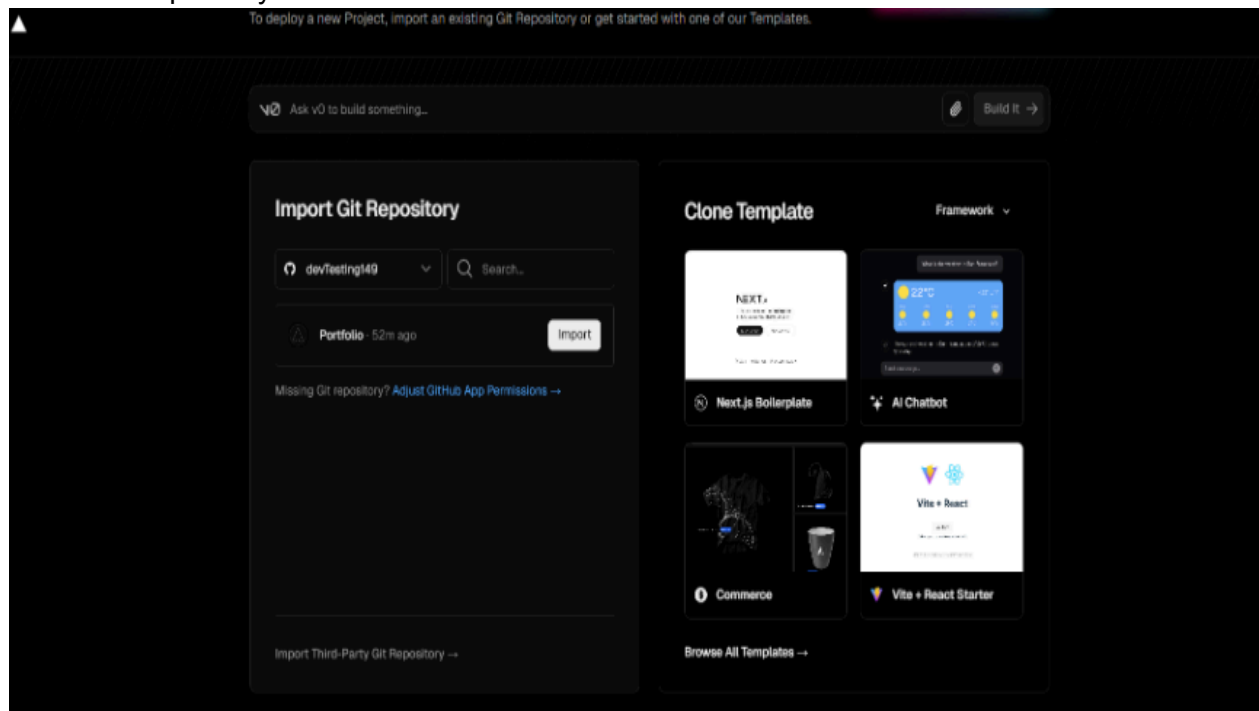
Import project



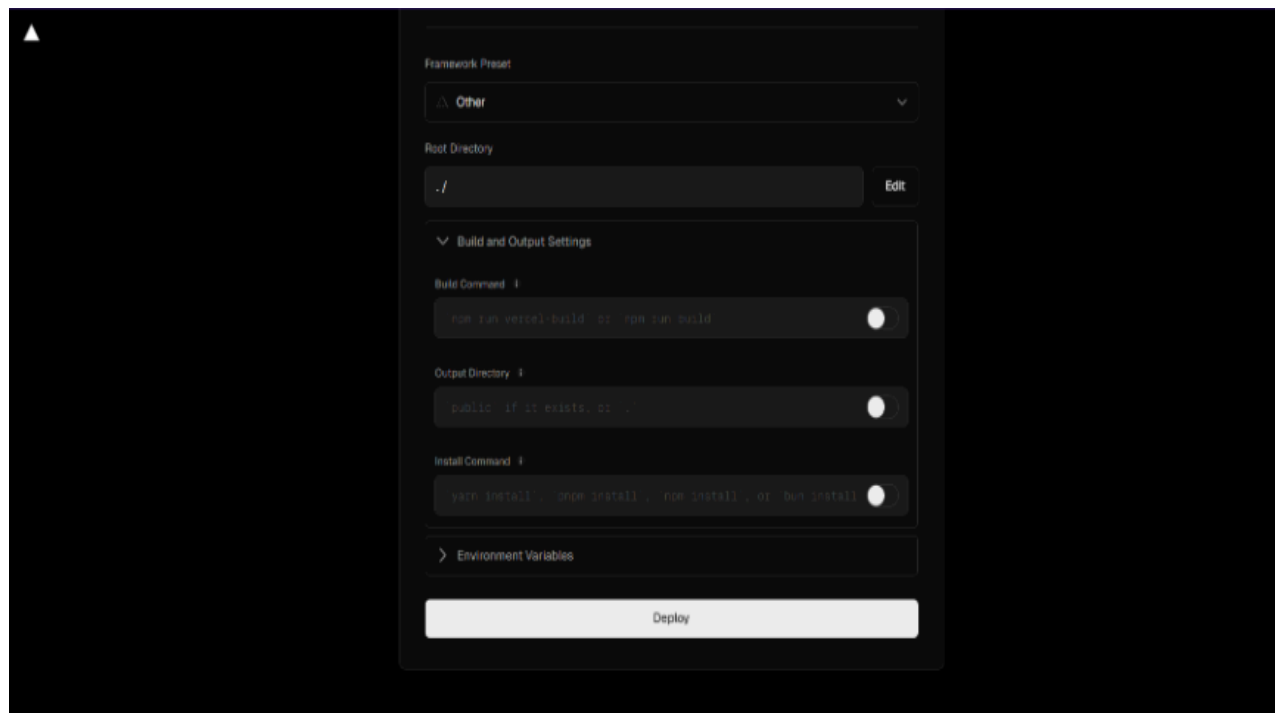
Install



Select the repository



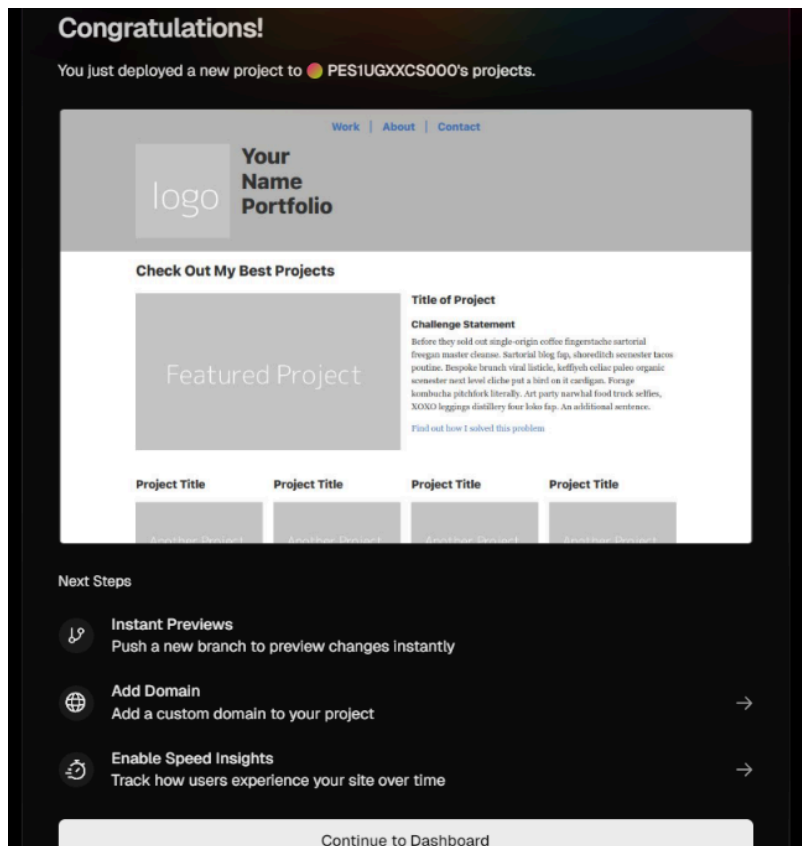
Import the project



The screenshot shows the 'Import Project' dialog in Vercel. It has a dark theme. At the top, there's a 'Framework Preset' dropdown menu with 'Other' selected. Below it is the 'Root Directory' field with './' and an 'Edit' button. A section titled 'Build and Output Settings' is expanded, showing three fields: 'Build Command' with 'npm run vercel-build' or 'npm run build', 'Output Directory' with 'public' if it exists or '.', and 'Install Command' with 'yarn install', 'npm install', or 'bun install'. There's an 'Environment Variables' link below. At the bottom is a large 'Deploy' button.

Deploy the project

6.Finally the project is deployed. This would be the final **Screenshot(SS9)**. Additionally, also paste the **link of the deployed site**. Example : <https://portfolio-ywnr.vercel.app/>



Congratulations on completing your first lab session. Hurray!!!!

If you are interested to explore even more cloud services for free here is the extensive list

<https://free-for.dev/#/>