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.

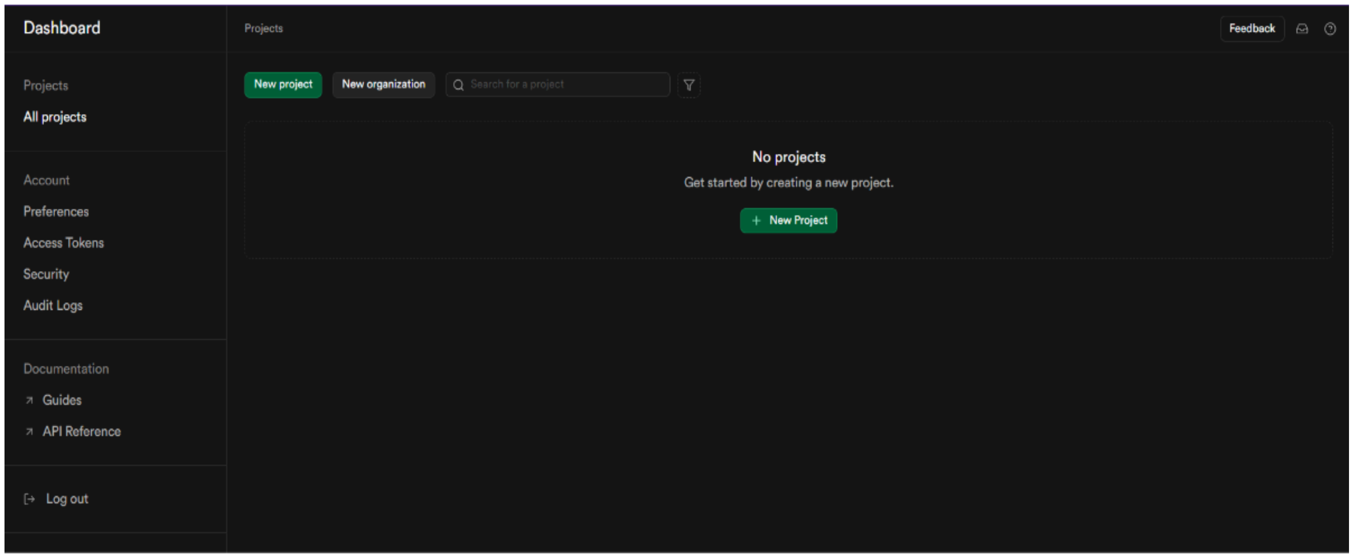
Saas Software as a service
Here they directly provide you the service, no need to worry about configuring anything. Just directly use it. Good example would be your gmail, google drive etc. Sign up on

Practice Time…

A) IAAS - Infrastructure Backend as a service

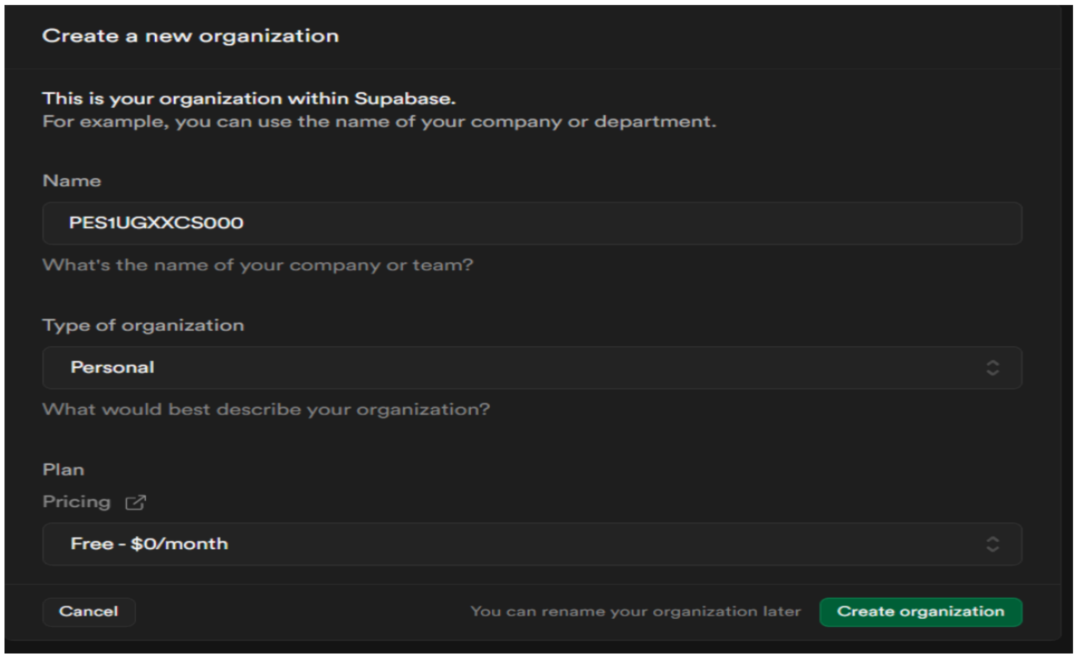
One of the widely used cloud services is object store, so unlike your traditional file system where data is stored as files and folders here you store data as blobs or objects, these objects are associated with a variable amount of metadata, and a globally unique identifier. Object storage systems allow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



No projects
Get started by creating a new project.

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.



Create a new organization

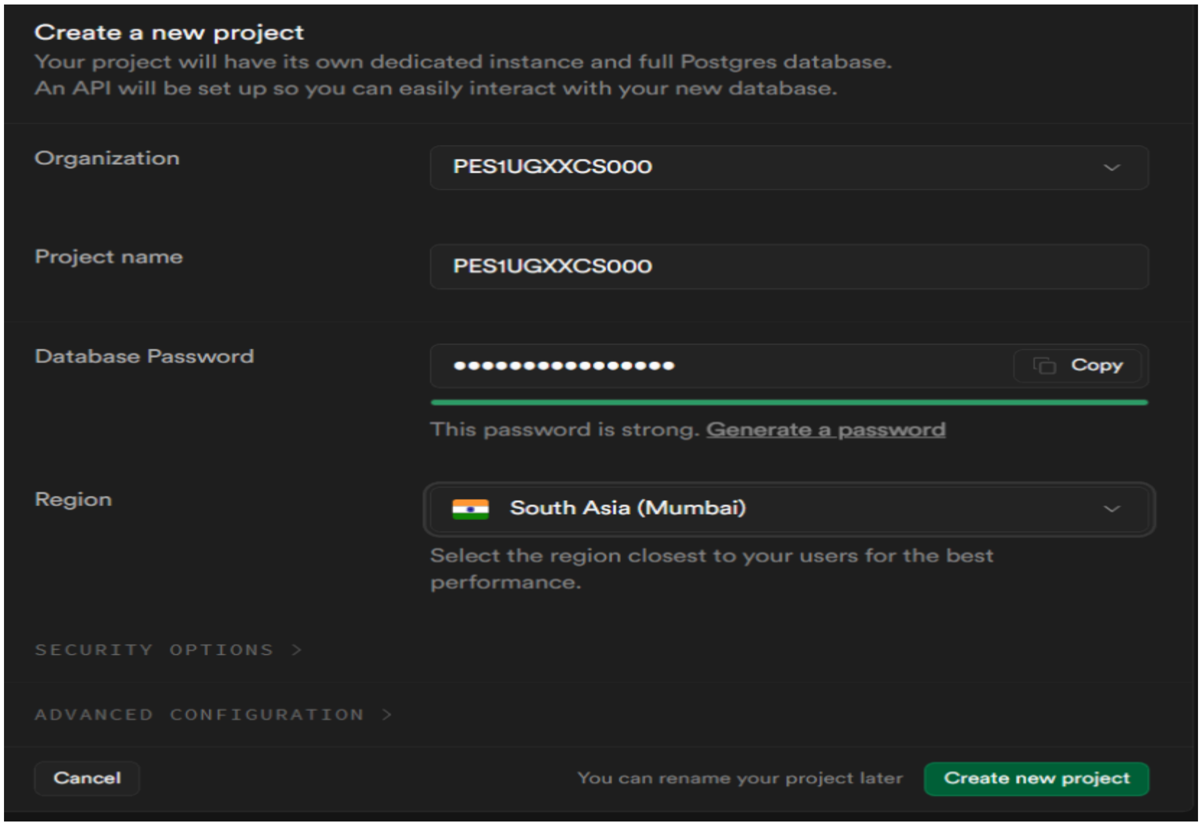
This is your organization within Supabase.
For example, you can use the name of your company or department.

Name
PES1UGXXCS000
What's the name of your company or team?

Type of organization
Personal
What would best describe your organization?

Plan
Pricing
Free - $0/month

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)



Create a new project

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.

Organization: PES1UGXXCS000

Project name: PES1UGXXCS000

Database Password: •••••••••••••••••

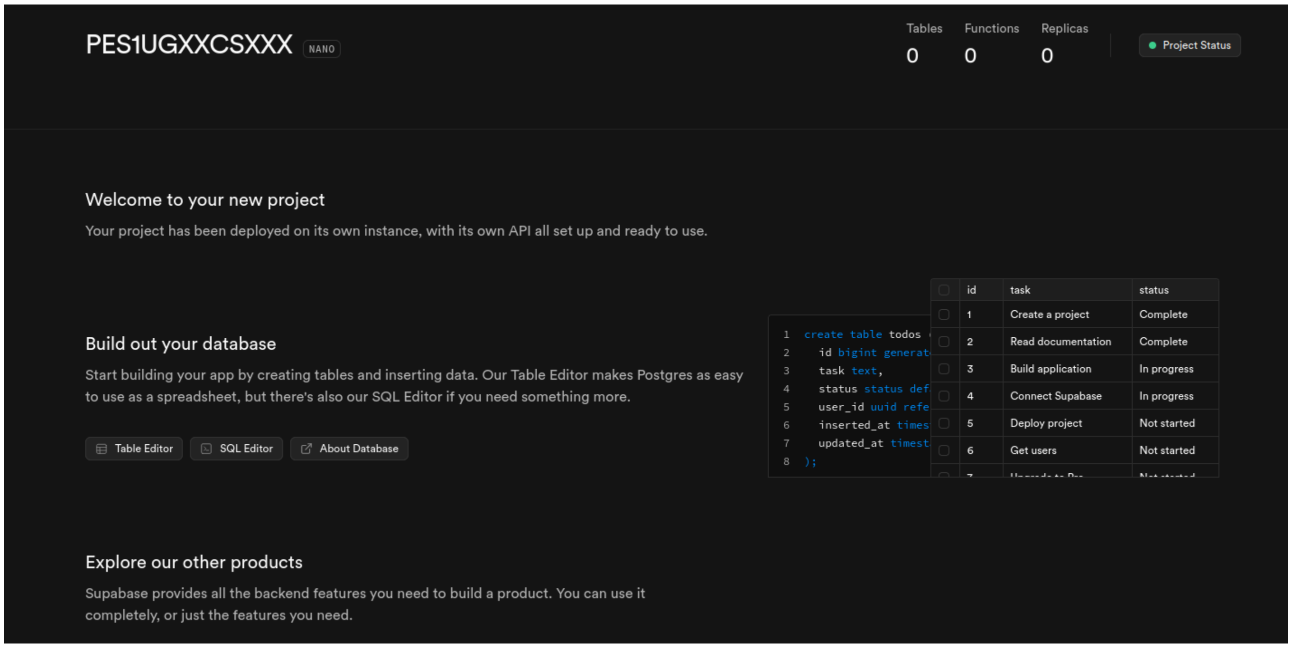
This password is strong. Generate a password

Region: South Asia (Mumbai)

Select the region closest to your users for the best performance.

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

Tables	Functions	Replicas
0	0	0

● 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

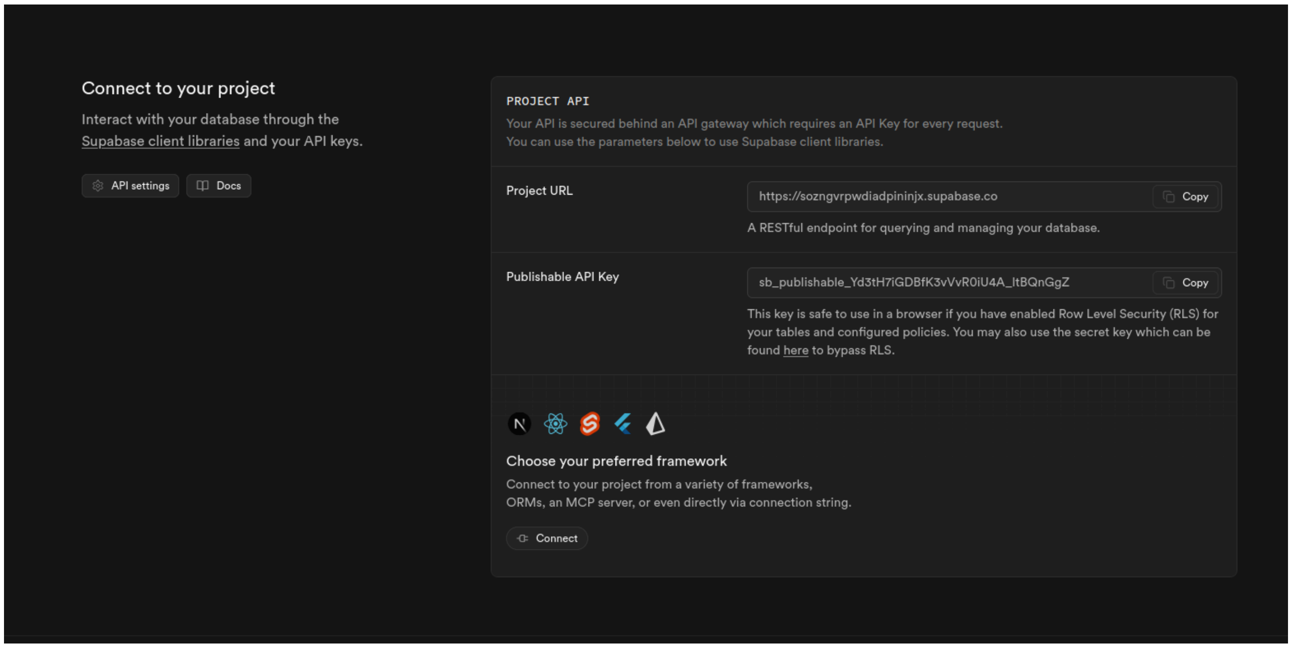
Start building your app by creating tables and inserting data. Our Table Editor makes Postgres as easy to use as a spreadsheet, but there's also our SQL Editor if you need something more.

```
create table todos (
  id bigint generat
  task text,
  status status def
  user_id uuid refe
  inserted_at times
  updated_at timest
);
```

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

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.



Connect to your project

Interact with your database through the Supabase client libraries and your API keys.

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://sozngvrpwdiadpininjx.supabase.co

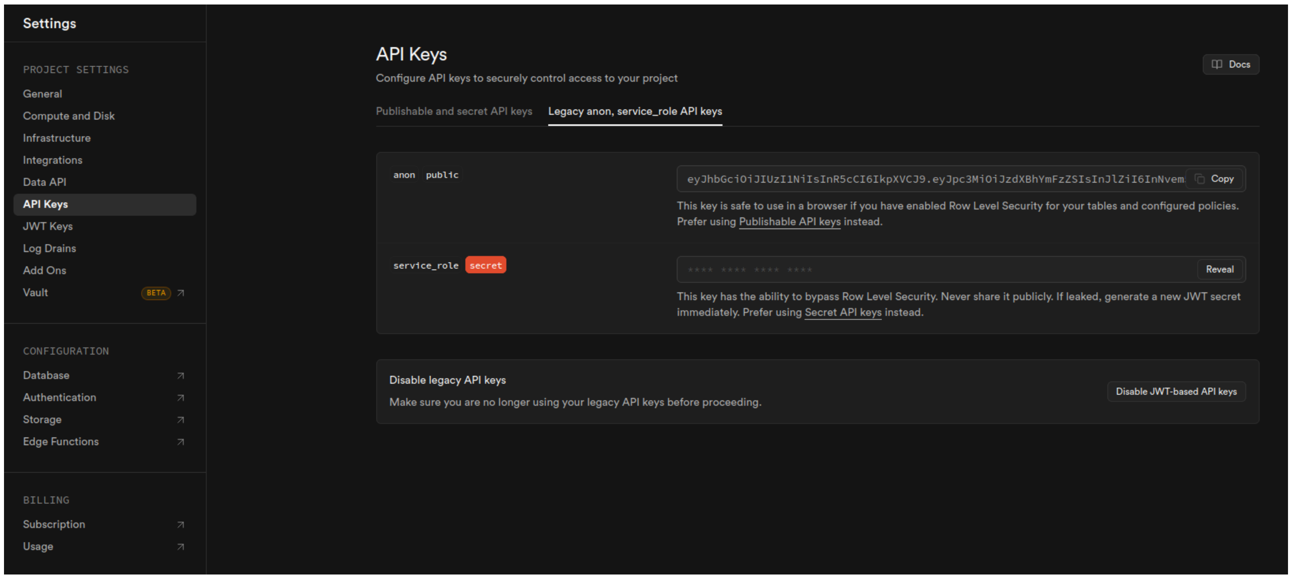
A RESTful endpoint for querying and managing your database.

Publishable API Key: sb_publishable_Yd3tH7iGDBfK3vVvR0iU4A_ltBQnGgZ

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.

Choose your preferred framework

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



API Keys

Configure API keys to securely control access to your project

anon public

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InNvem
```

This key is safe to use in a browser if you have enabled Row Level Security for your tables and configured policies. Prefer using Publishable API keys instead.

service_role secret

```
**** **** **** ****
```

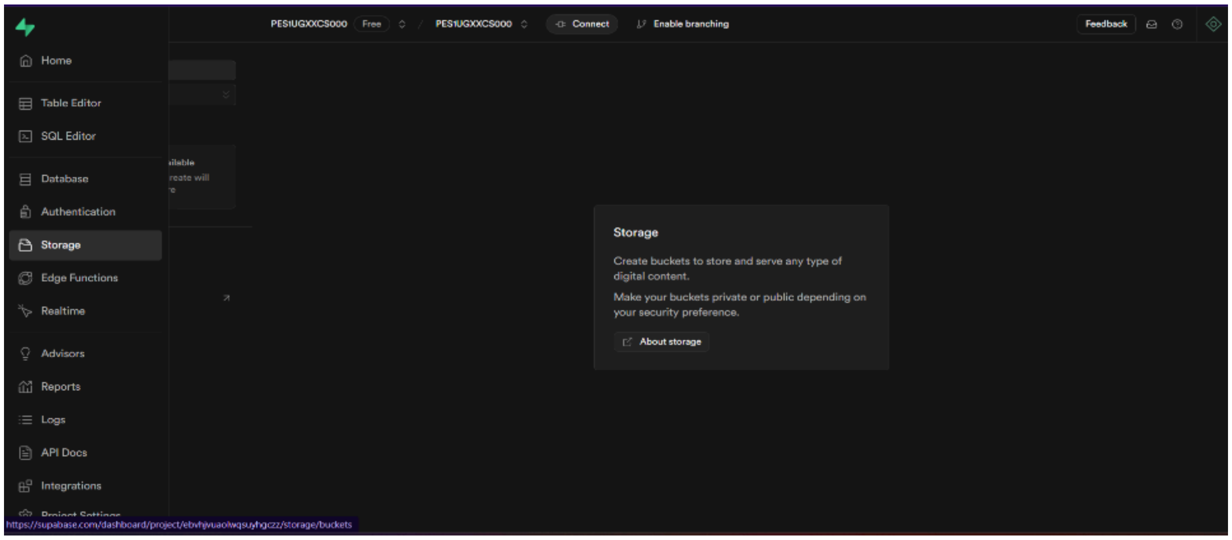
This key has the ability to bypass Row Level Security. Never share it publicly. If leaked, generate a new JWT secret immediately. Prefer using Secret API keys instead.

Disable legacy API keys

Make sure you are no longer using your legacy API keys before proceeding.

(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.



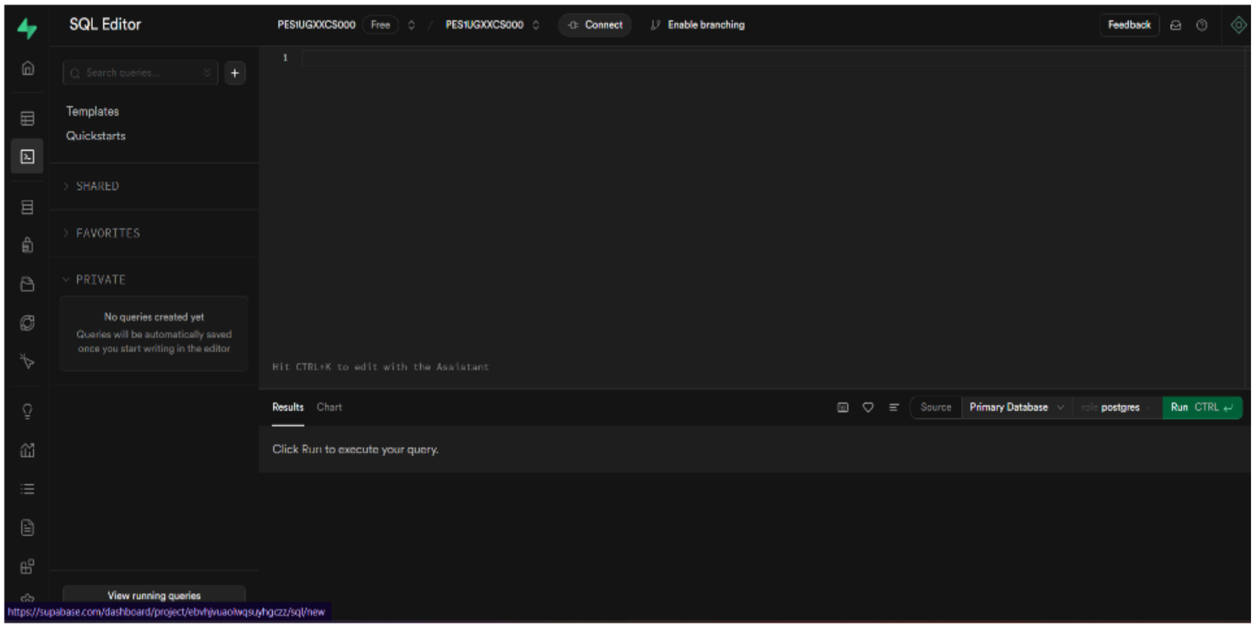
Storage

Create buckets to store and serve any type of digital content.

Make your buckets private or public depending on your security preference.

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.



No queries created yet
Queries will be automatically saved once you start writing in the editor

Hit CTRL+K to edit with the Assistant

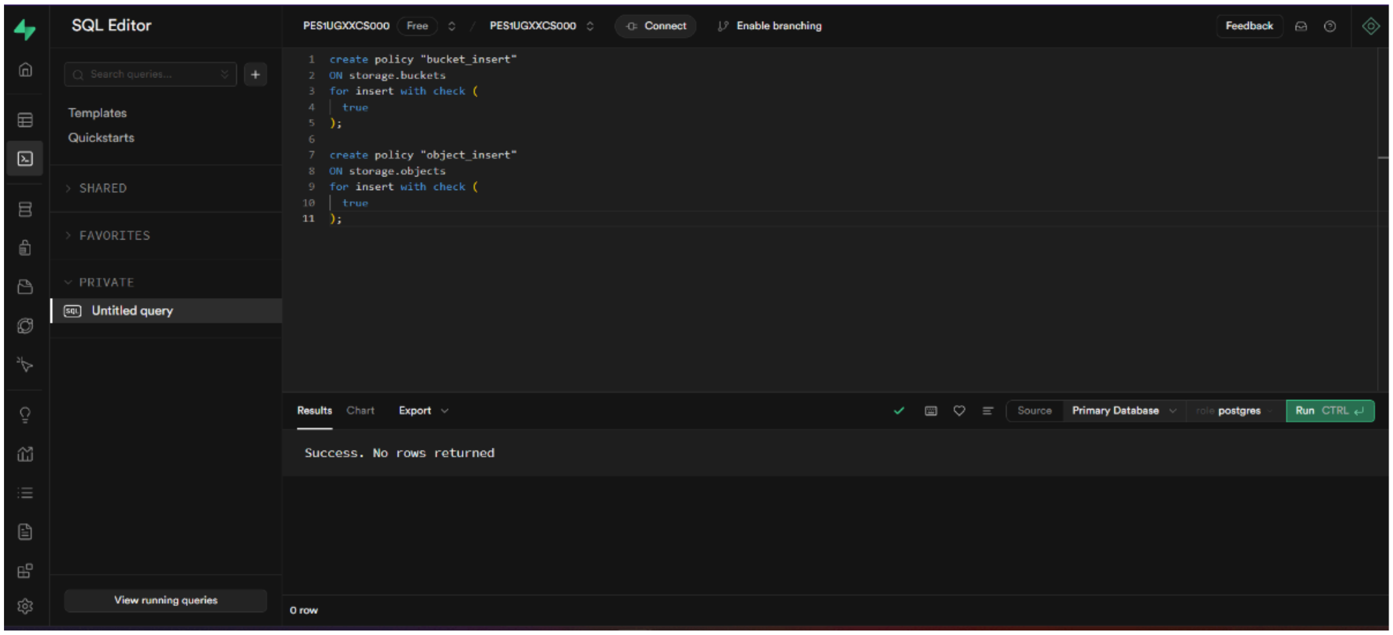
Click Run to execute your query.

https://supabase.com/dashboard/project/ebvhjvuaolwqsuyhgczz/sql/new

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.]



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

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

```
Success. No rows returned
```

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

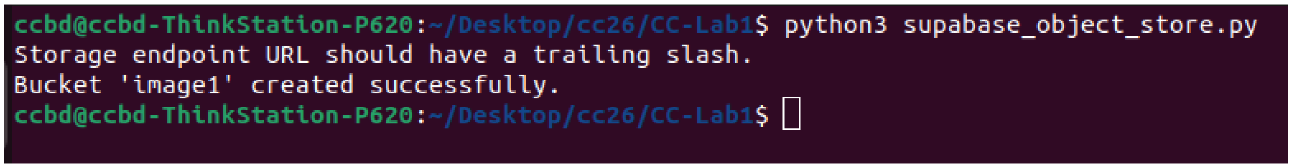
9. Next open up the **supabase_object_store.py** file, make sure you added servicei key and project url. You can change the bucket name or the image path.

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

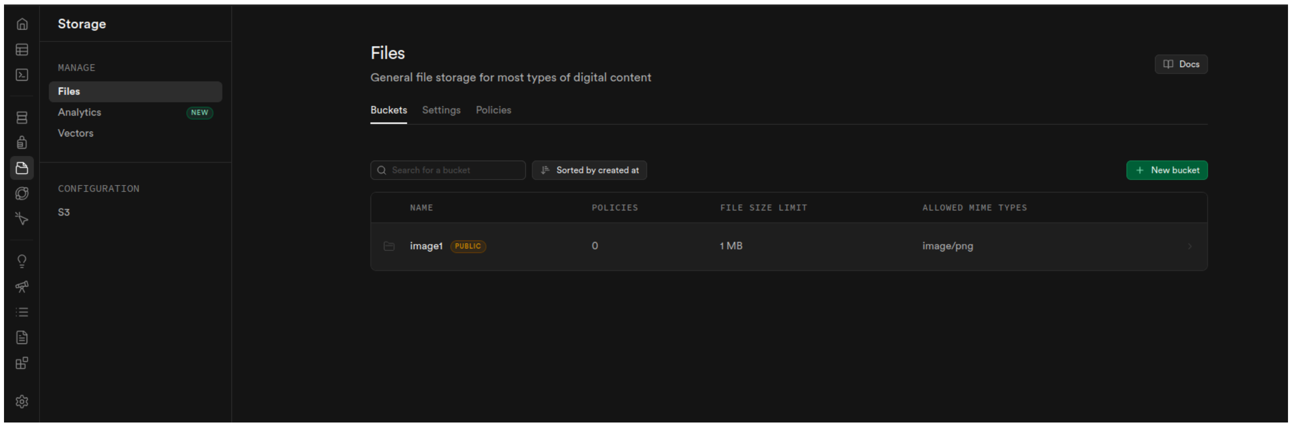
11. Run each step individually by commenting and uncommenting the other step. Code is extensively commented so that you understand what each part of the code does. At the end you should see a bucket created, your image uploaded, and a public url to view the image. (Note you can upload the portfolio image 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$ 
```



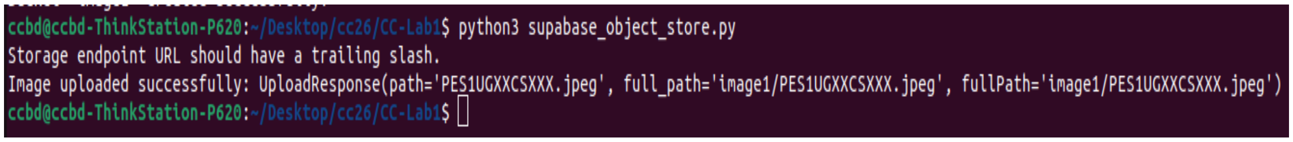
Files

General file storage for most types of digital content

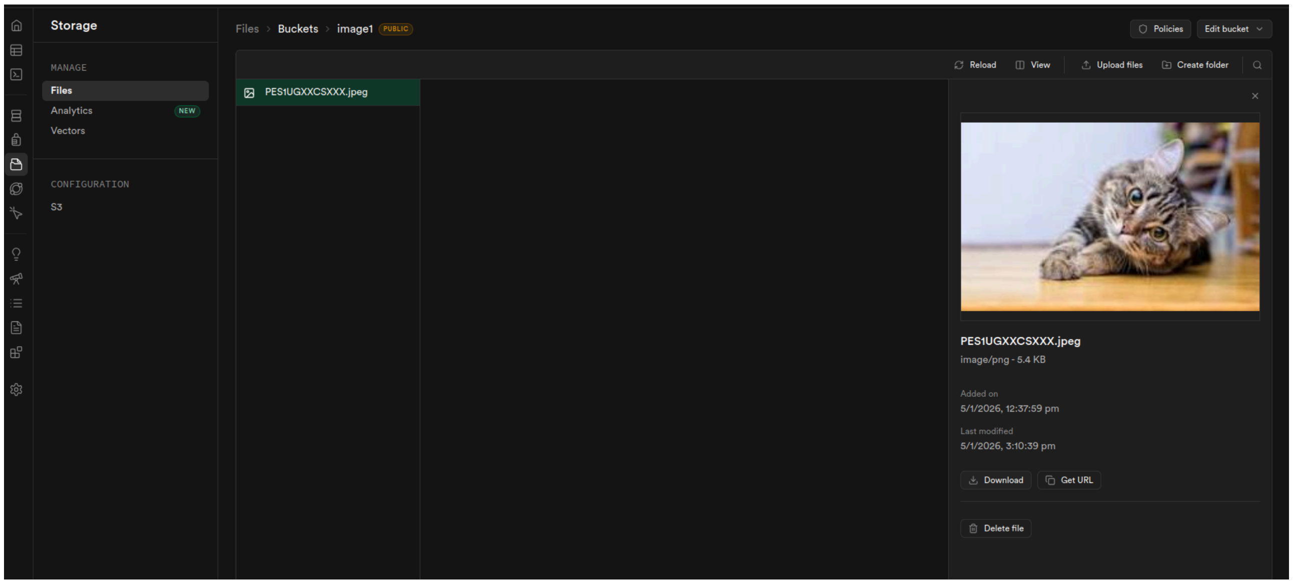
NAME	POLICIES	FILE SIZE LIMIT	ALLOWED MIME TYPES
image1 PUBLIC	0	1 MB	image/png

- 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$ 
```



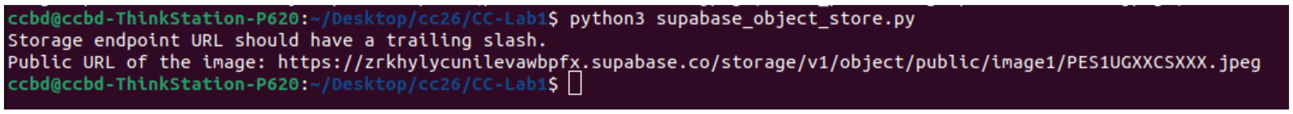
PES1UGXXCSXXX.jpeg

image/png - 5.4 KB

Added on
5/1/2026, 12:37:59 pm

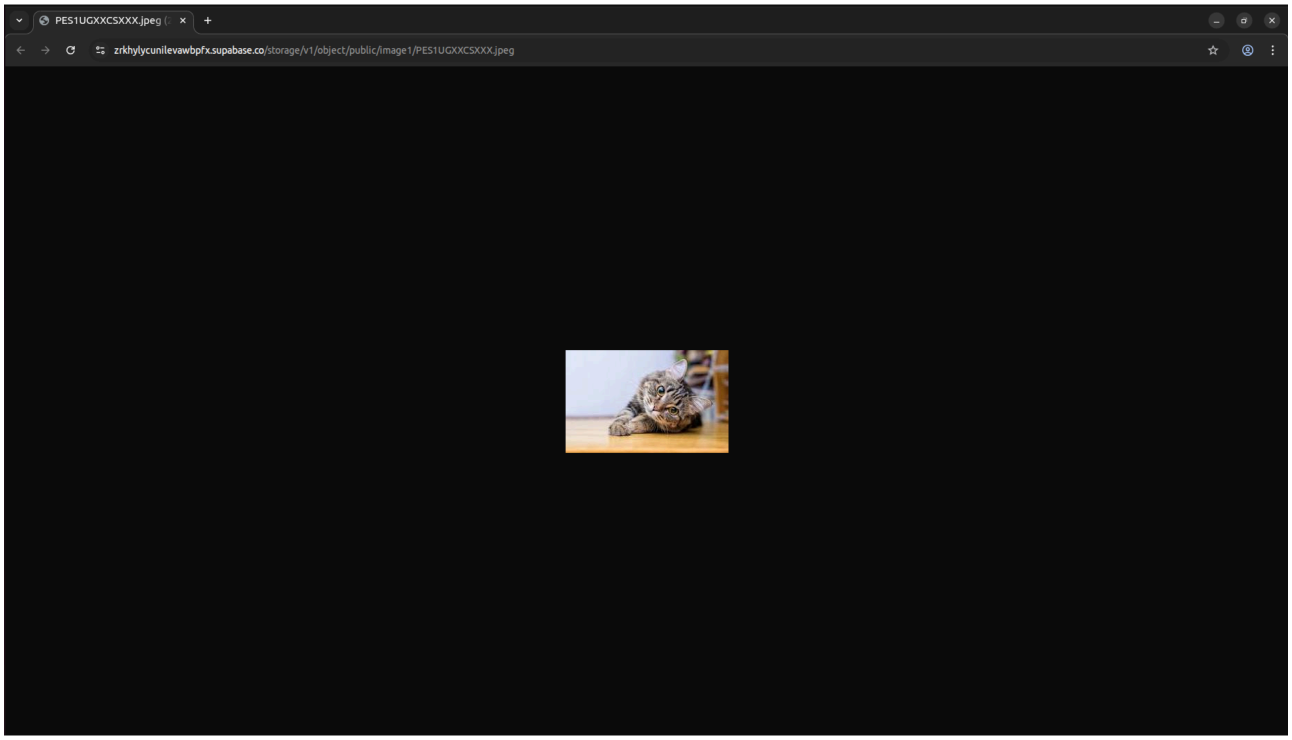
Last modified
5/1/2026, 3:10:39 pm

(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://lvxhhflqjupwgezviuqh.supabase.co/storage/v1/object/public/images/cat.jpeg`



```
PES1UGXXCSXXX.jpeg
zrkhylycunilevawbpfx.supabase.co/storage/v1/object/public/image1/PES1UGXXCSXXX.jpeg
```

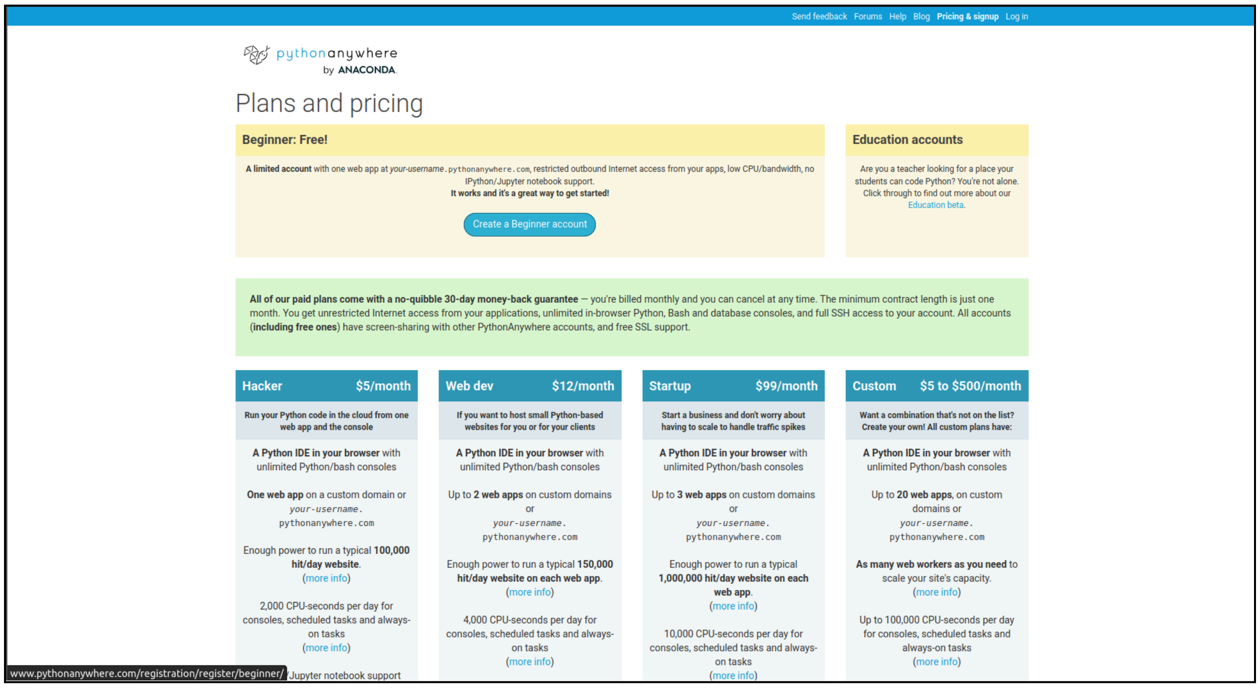
B) PaaS - Platform as a Service - Deployment of a Flask Application using PythonAnywhere

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/


Send feedback Forums Help Blog Pricing & signup Log in
pythonanywhere by ANACONDA
Create a Beginner account
www.pythonanywhere.com/registration/register/beginner/

Plans and pricing

Beginner: Free!

A limited account with one web app at *your-username*.`pythonanywhere.com`, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython/Jupyter notebook support.
It works and it's a great way to get started!

Education accounts

Are you a teacher looking for a place your students can code Python? You're not alone. Click through to find out more about our Education beta.

All of our paid plans come with a no-quibble 30-day money-back guarantee — you're billed monthly and you can cancel at any time. The minimum contract length is just one month. You get unrestricted Internet access from your applications, unlimited in-browser Python, Bash and database consoles, and full SSH access to your account. All accounts (**including free ones**) have screen-sharing with other PythonAnywhere accounts, and free SSL support.

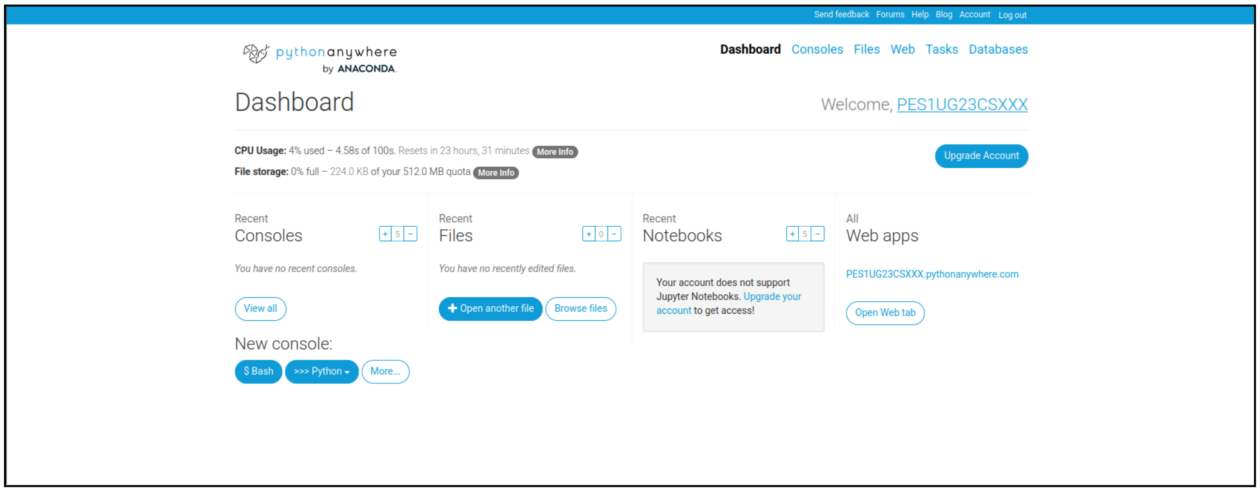
Hacker $5/month	Web dev $12/month	Startup $99/month	Custom $5 to $500/month
Run your Python code in the cloud from one web app and the console	**If you want to host small Python-based websites for you or for your clients**	**Start a business and don't worry about having to scale to handle traffic spikes**	**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)
Jupyter notebook support			

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**.



Dashboard

Welcome, PES1UG23CSXXX

CPU Usage: 4% used – 4.58s of 100s. Resets in 23 hours, 31 minutes More Info

File storage: 0% full – 224.0 KB of your 512.0 MB quota More Info

You have no recent consoles.

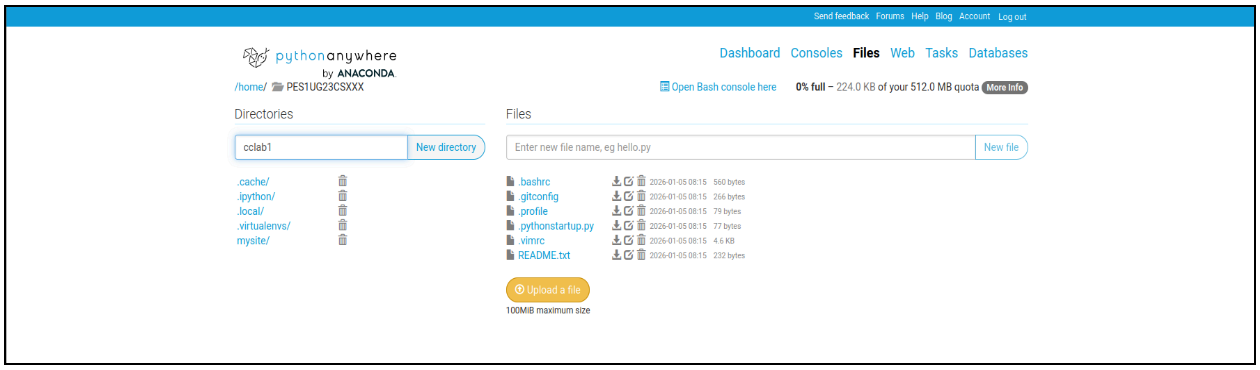
You have no recently edited files.

Your account does not support Jupyter Notebooks. Upgrade your account to get access!

PES1UG23CSXXX.pythonanywhere.com

Step 3: Create Project Directory

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



/home/ PES1UG23CSXXX

0% full – 224.0 KB of your 512.0 MB quota More Info

cclab1

Enter new file name, eg hello.py

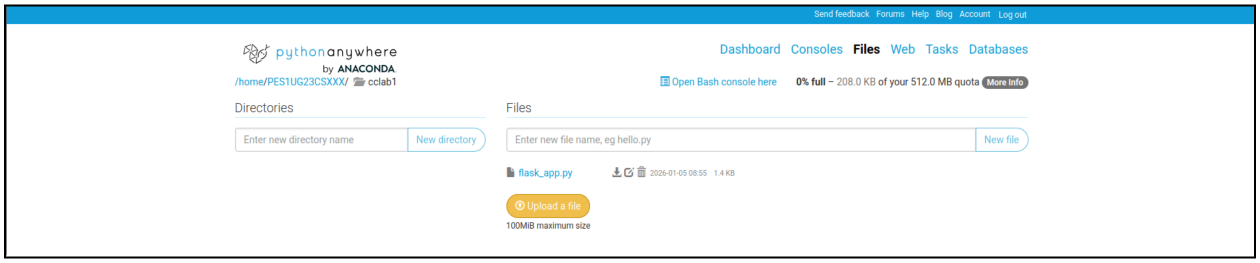
.cache/
.ipython/
.local/
.virtualenvs/
mysite/

```
.bashrc            2026-01-05 08:15   560 bytes
.gitconfig         2026-01-05 08:15   266 bytes
.profile           2026-01-05 08:15   79 bytes
.pythonstartup.py  2026-01-05 08:15   77 bytes
.vimrc             2026-01-05 08:15   4.6 KB
README.txt         2026-01-05 08:15   232 bytes
```

100MiB maximum size

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**.


Send feedback Forums Help Blog Account Log out
pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases
Open Bash console here
More Info
New directory
New file
Upload a file

/home/PES1UG23CSXXX/ cclab1

0% full – 208.0 KB of your 512.0 MB quota

Directories

Enter new directory name

Files

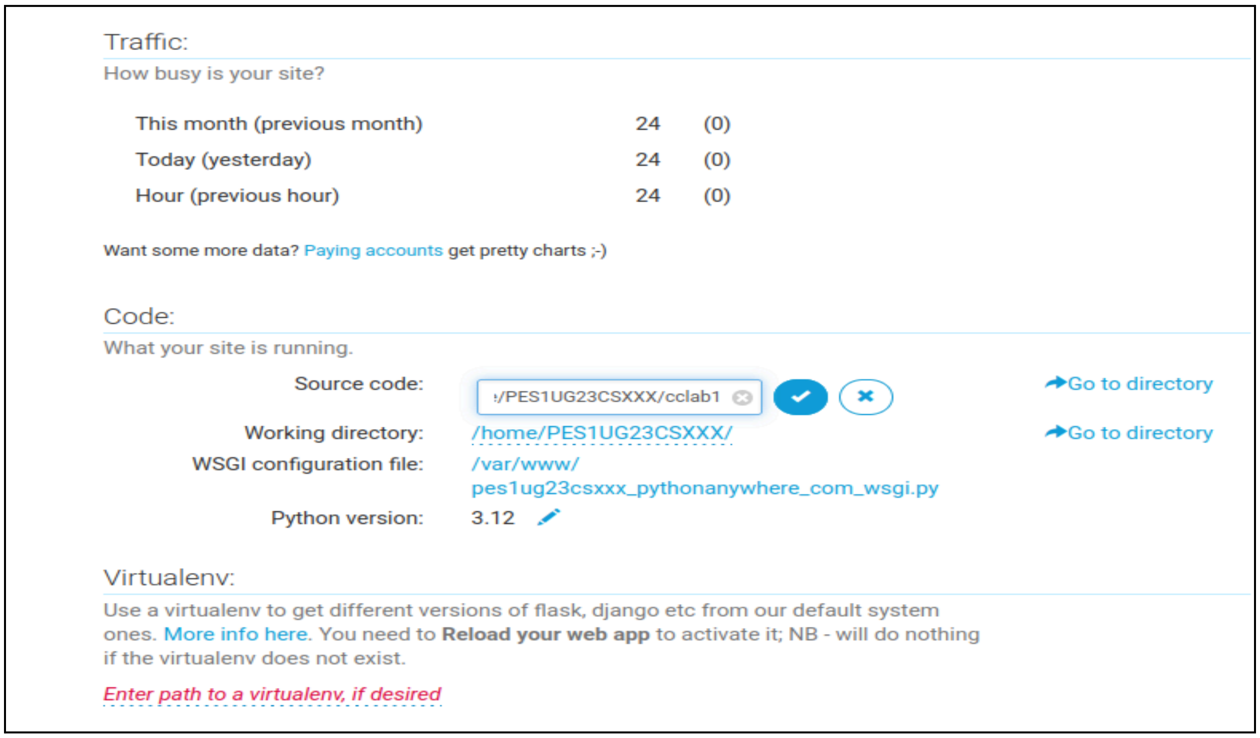
Enter new file name, eg hello.py

flask_app.py 2026-01-05 08:55 1.4 KB

100MiB maximum size

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/cclab1`



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: :/PES1UG23CSXXX/cclab1

Working directory: /home/PES1UG23CSXXX/

WSGI configuration file: /var/www/pes1ug23csxxx_pythonanywhere_com_wsgi.py

Python version: 3.12

Virtualenv:

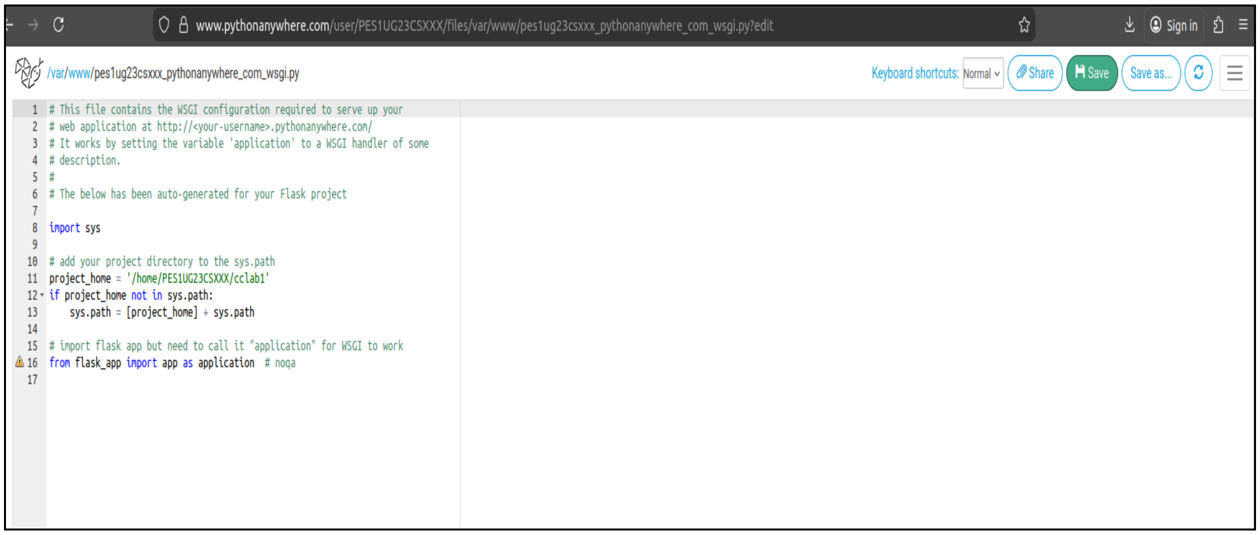
Use a virtualenv to get different versions of flask, django etc from our default system ones. More info here. You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

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/cclab1'
```



www.pythonanywhere.com/user/PES1UG23CSXXX/files/var/www/pes1ug23csxxx_pythonanywhere_com_wsgi.py?edit

/var/www/pes1ug23csxxx_pythonanywhere_com_wsgi.py

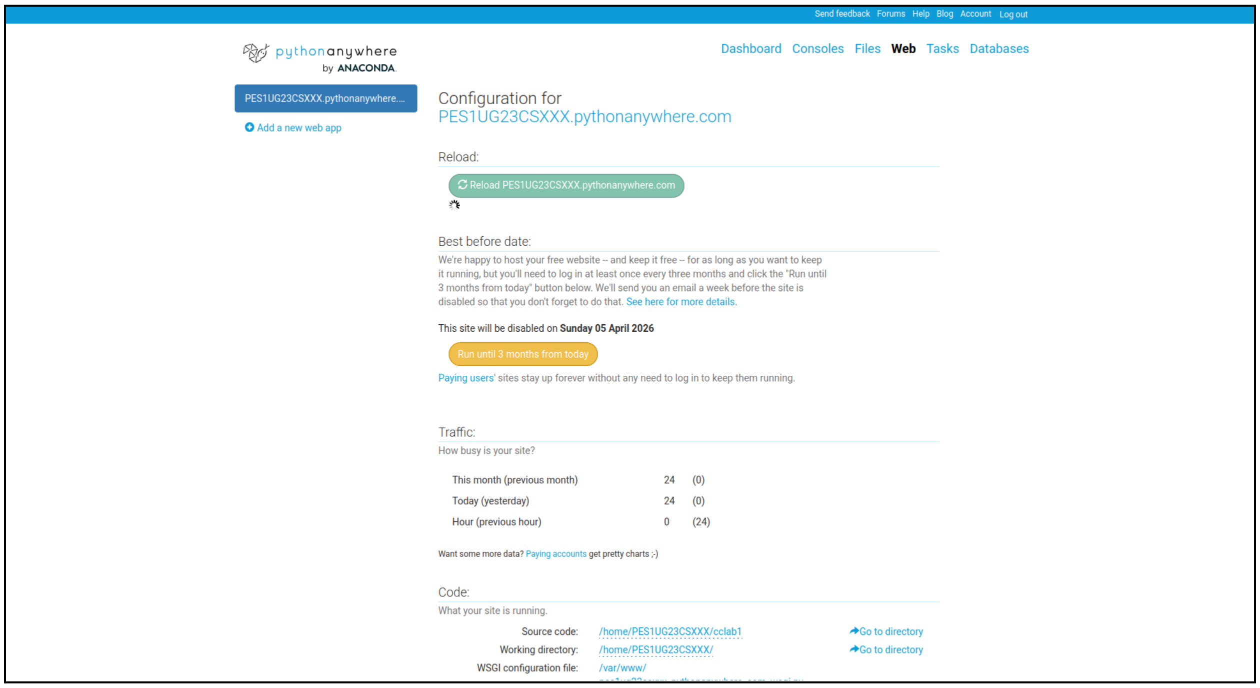
```python
# This file contains the WSGI configuration required to serve up your
# web application at http://<your-username>.pythonanywhere.com/
# It works by setting the variable 'application' to a WSGI handler of some
# description.
#
# The below has been auto-generated for your Flask project

import sys

# add your project directory to the sys.path
project_home = '/home/PES1UG23CSXXX/cclab1'
if project_home not in sys.path:
    sys.path = [project_home] + sys.path

# import flask app but need to call it "application" for WSGI to work
from flask_app import app as application  # noqa
```

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



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 ;-)

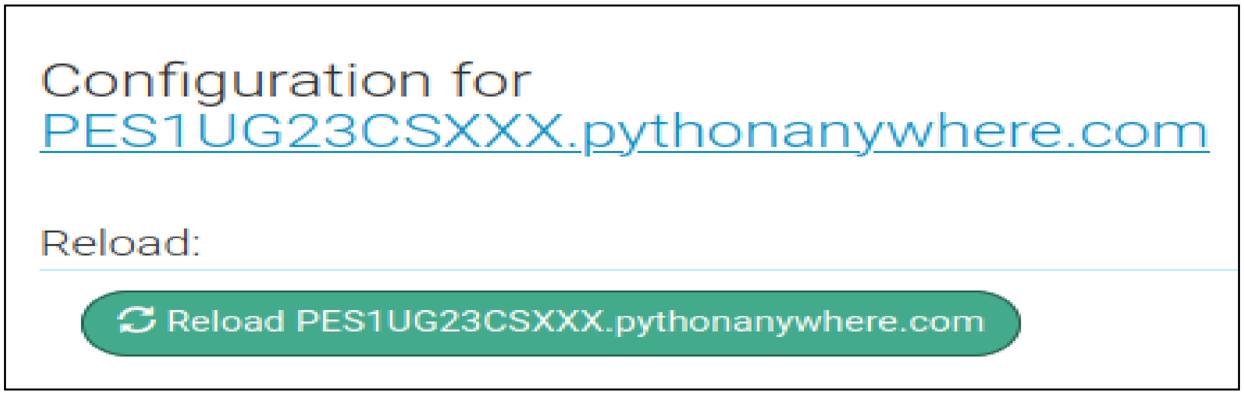
Code:

What your site is running.

Source code:	/home/PES1UG23CSXXX/cclab1	Go to directory
Working directory:	/home/PES1UG23CSXXX/	Go to directory
WSGI configuration file:	/var/www/	

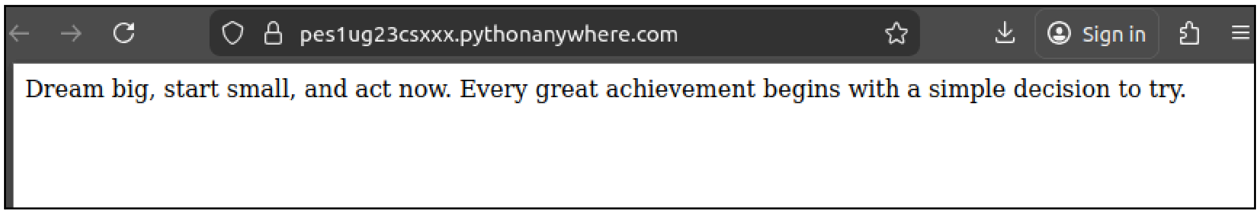
Step 7: Access the Deployed Application

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


Configuration for PES1UG23CSXXX.pythonanywhere.com

Reload:

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.



Dream big, start small, and act now. Every great achievement begins with a simple decision to try.

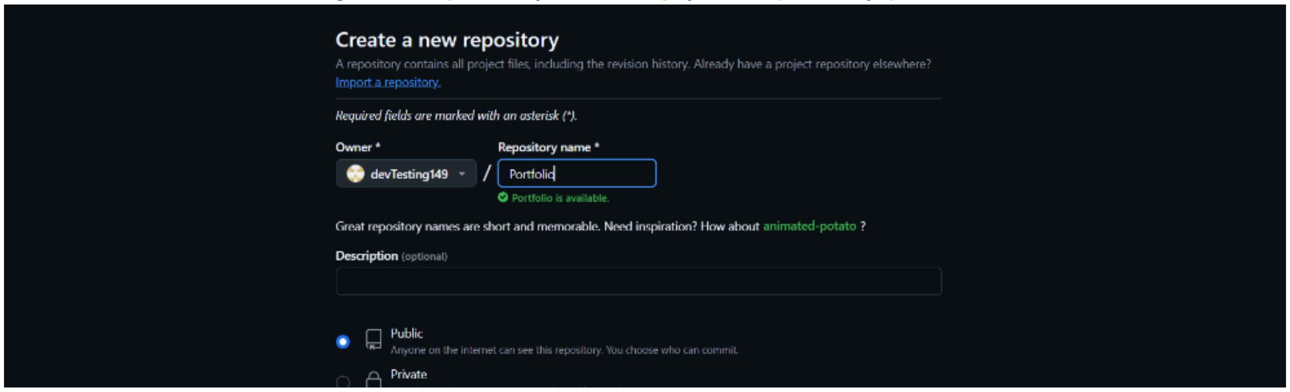
Result

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.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * devTesting149 / Repository name * Portfolio

Portfolio is available.

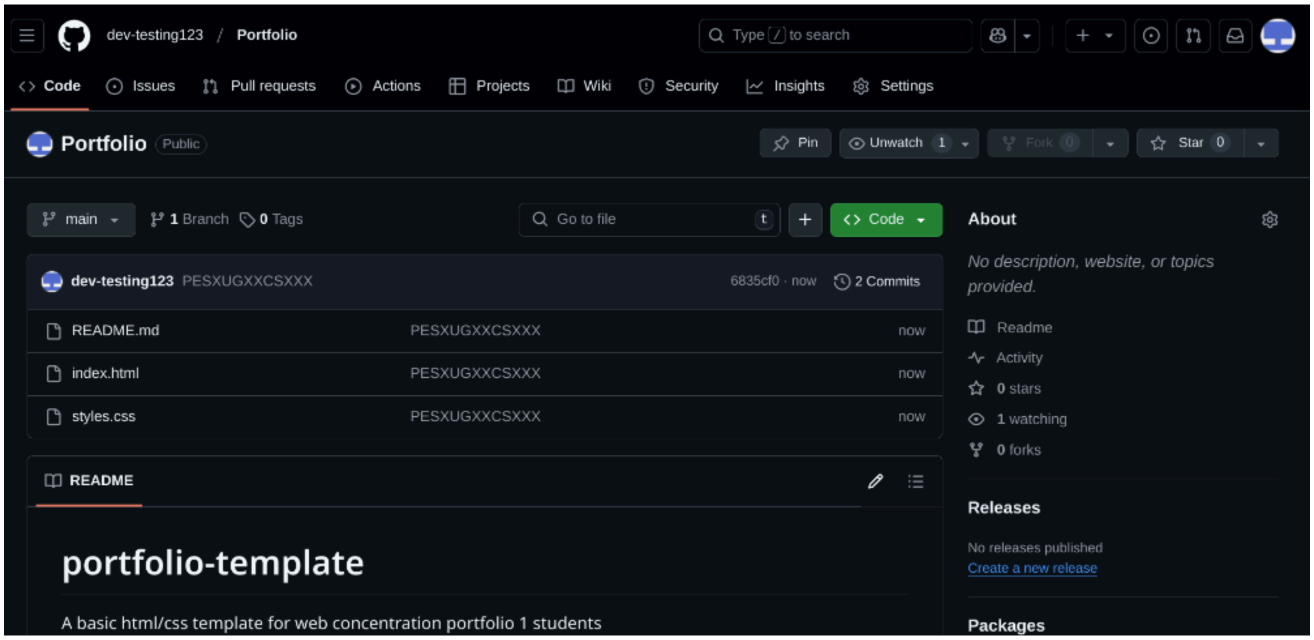
Great repository names are short and memorable. Need inspiration? How about animated-potato ?

Description (optional)

● Public — Anyone on the internet can see this repository. You choose who can commit.

○ Private

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).**



dev-testing123 / Portfolio

Portfolio Public

main · 1 Branch · 0 Tags

dev-testing123 PESXUGXXCSXXX · 6835cf0 · now · 2 Commits

README.md	PESXUGXXCSXXX	now
index.html	PESXUGXXCSXXX	now
styles.css	PESXUGXXCSXXX	now

README

portfolio-template

A basic html/css template for web concentration portfolio 1 students

About

No description, website, or topics provided.

Readme · Activity · 0 stars · 1 watching · 0 forks

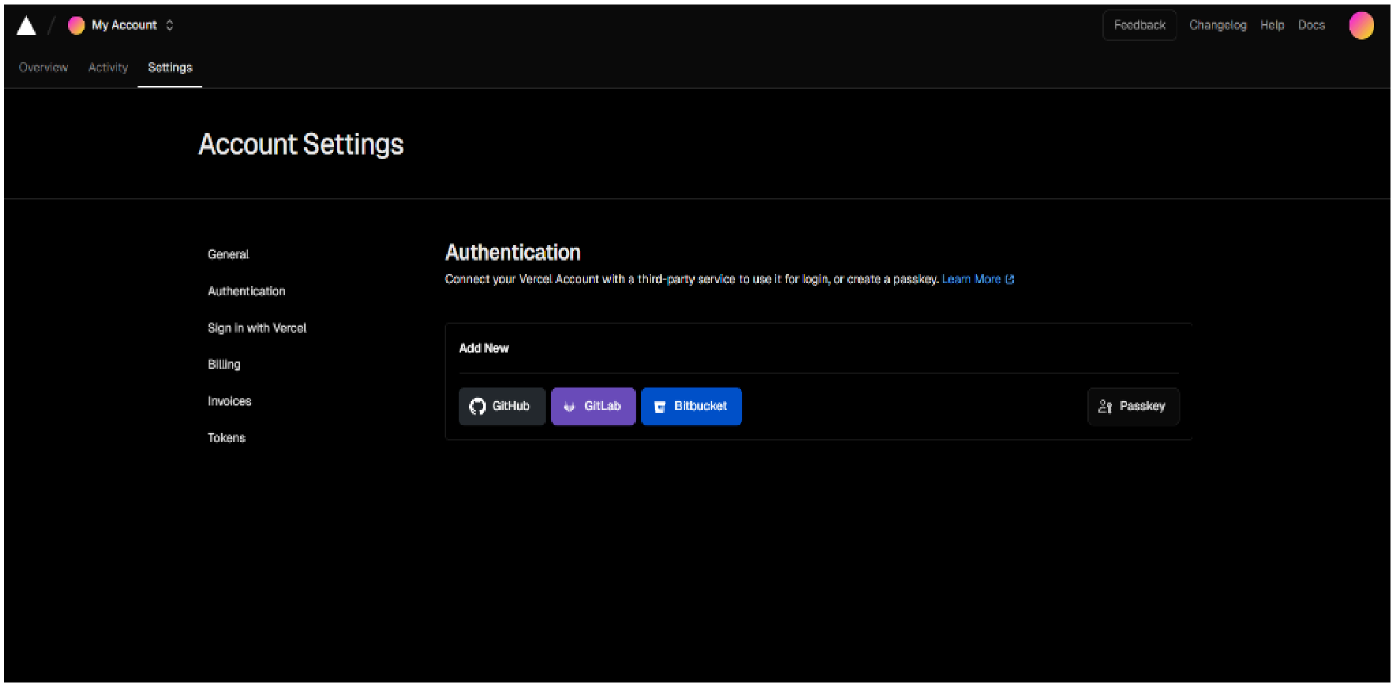
Releases

No releases published

Create a new release

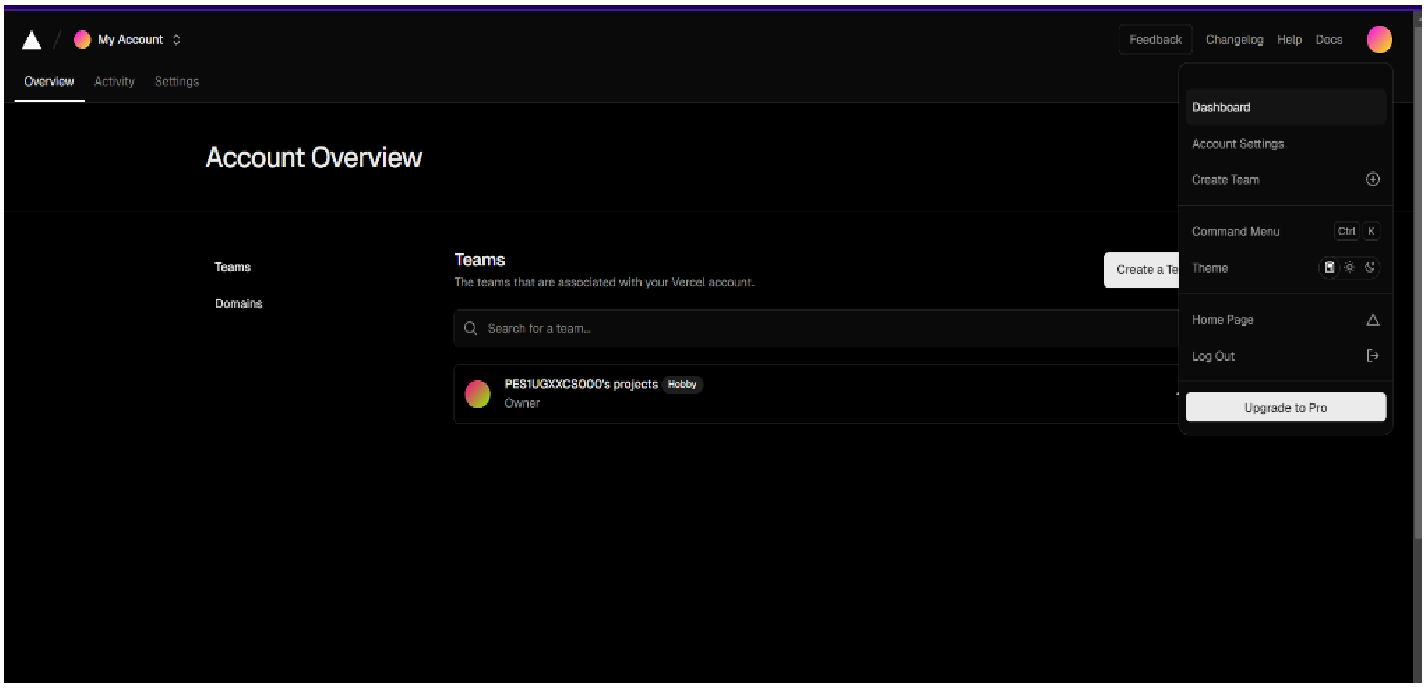
Packages

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

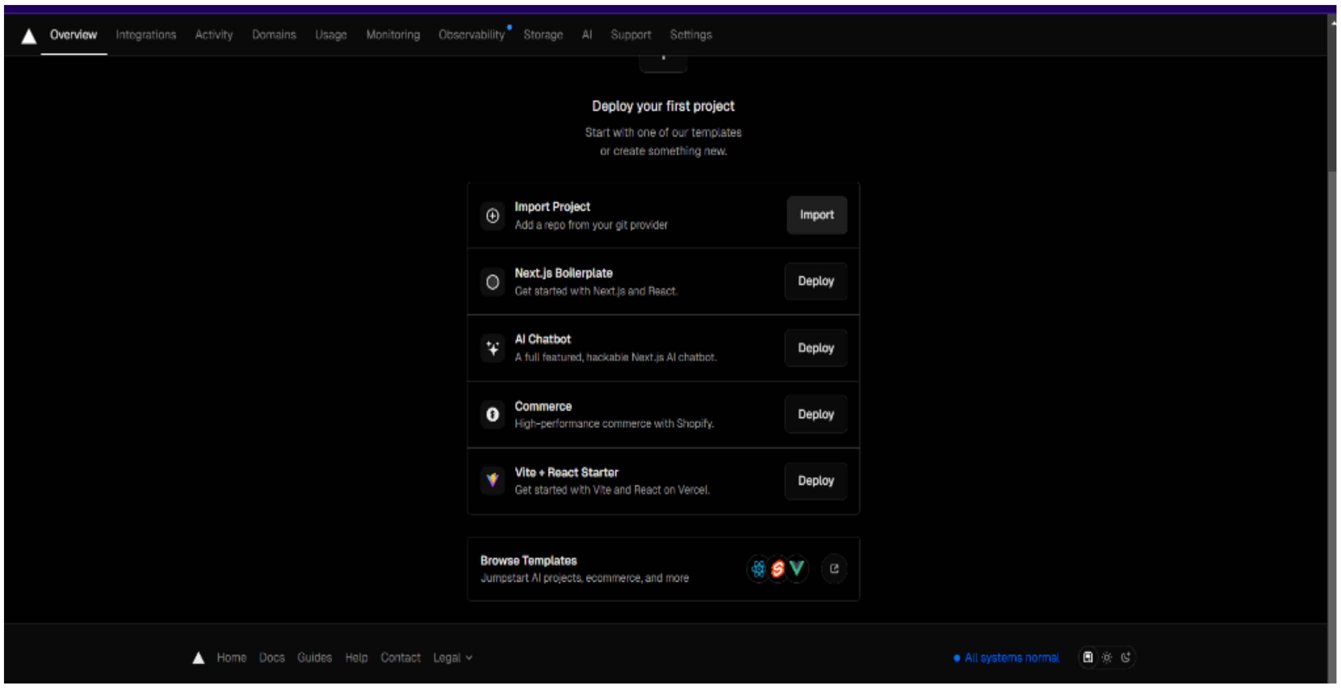


(if you use different email, then you need to connect to the GitHub explicitly)

5. Now we will import project from github and deploy



Go to dashboard



Overview Integrations Activity Domains Usage Monitoring Observability Storage AI Support Settings

Deploy your first project

Start with one of our templates or create something new.

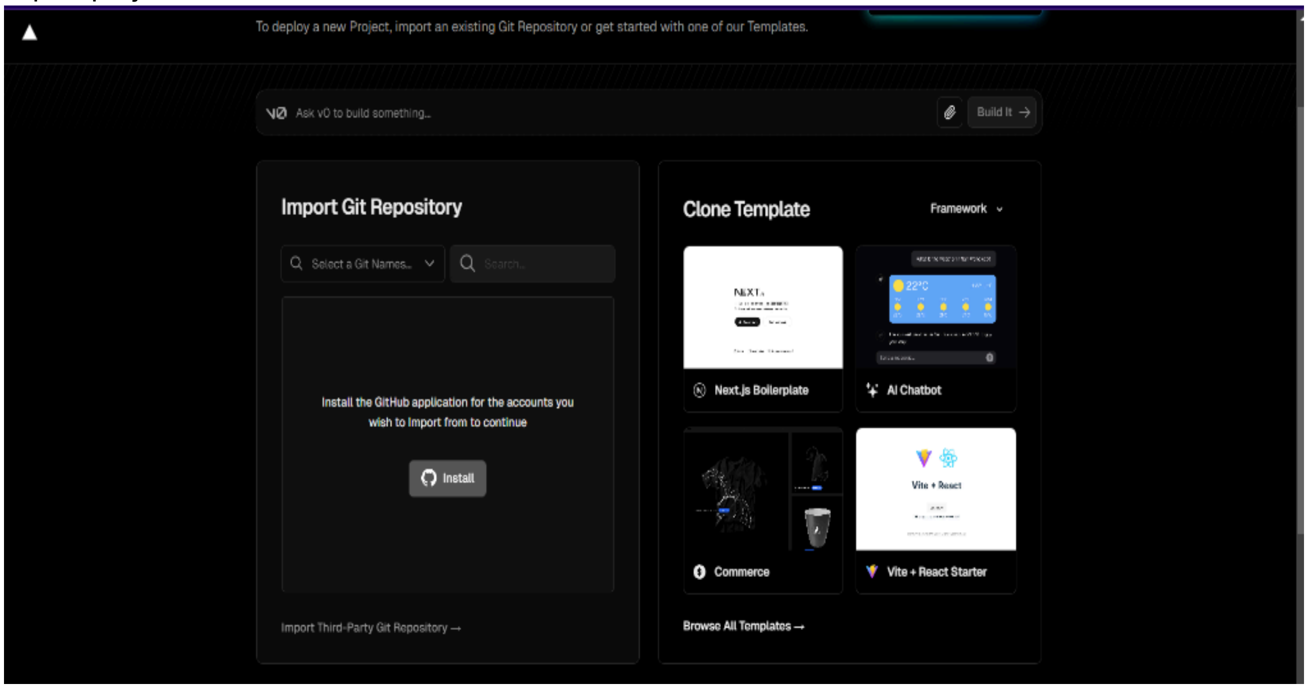
Import Project Add a repo from your git provider	Import
Next.js Boilerplate Get started with Next.js and React.	Deploy
AI Chatbot A full featured, hackable Next.js AI chatbot.	Deploy
Commerce High-performance commerce with Shopify.	Deploy
Vite + React Starter Get started with Vite and React on Vercel.	Deploy

Browse Templates Jumpstart AI projects, ecommerce, and more

Home Docs Guides Help Contact Legal

All systems normal

Import project



To deploy a new Project, import an existing Git Repository or get started with one of our Templates.

Ask v0 to build something... Build It

Import Git Repository

Select a Git Namespace Search...

Install the GitHub application for the accounts you wish to Import from to continue

Install

Import Third-Party Git Repository →

Clone Template Framework

Next.js Boilerplate

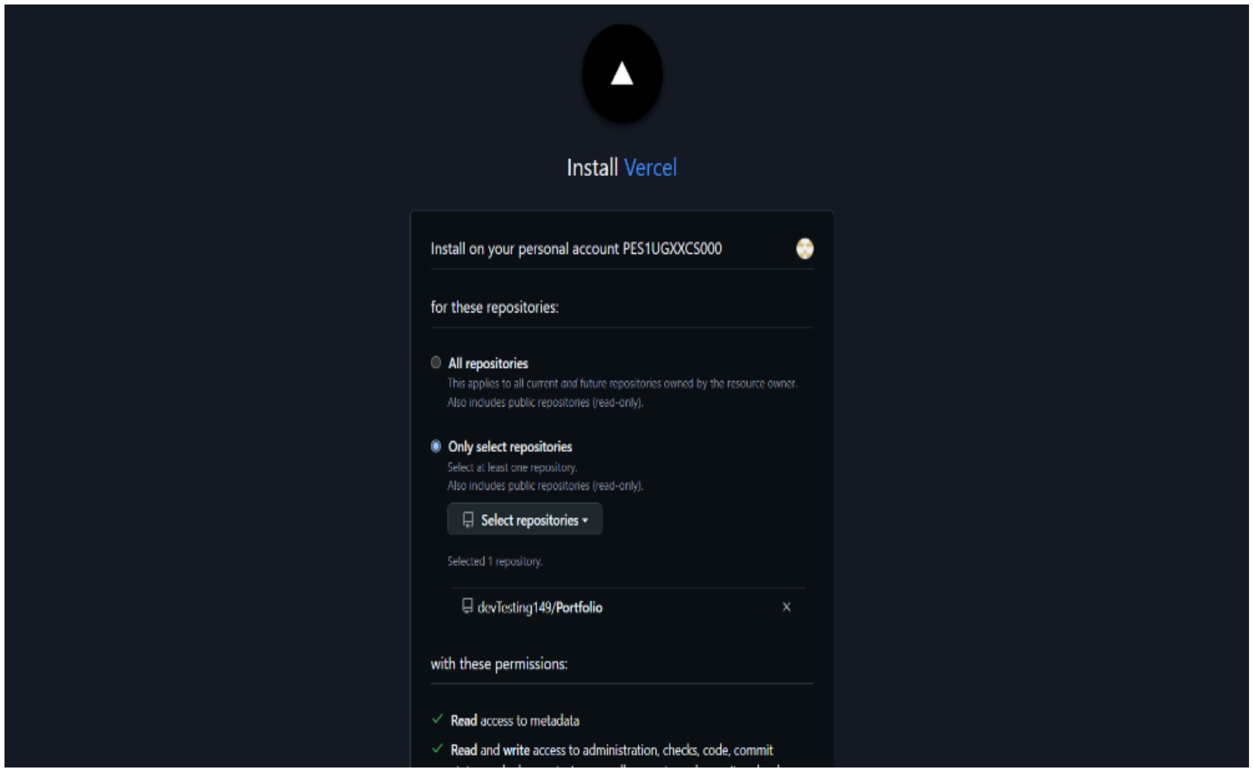
AI Chatbot

Commerce

Vite + React Starter

Browse All Templates →

Install



Install Vercel

Install on your personal account PES1UGXXCS000

for these repositories:

All repositories
This applies to all current and future repositories owned by the resource owner.
Also includes public repositories (read-only).

● Only select repositories
Select at least one repository.
Also includes public repositories (read-only).

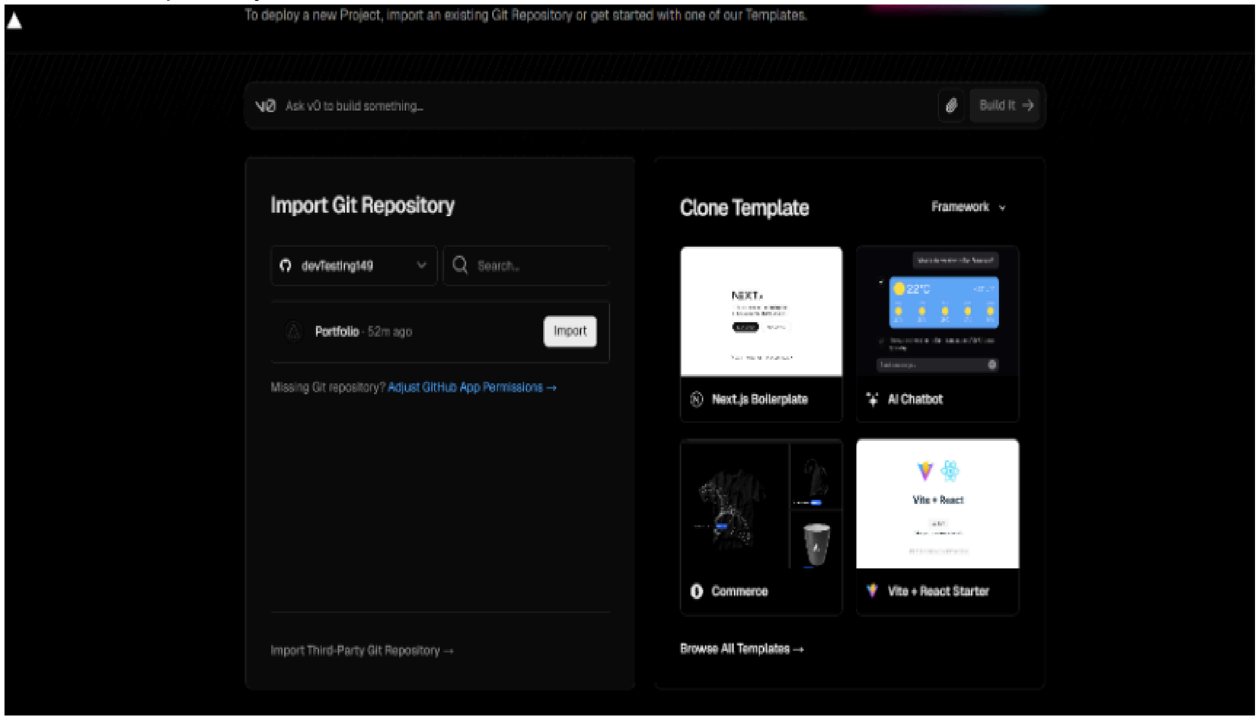
Selected 1 repository.

devTesting149/Portfolio ×

with these permissions:

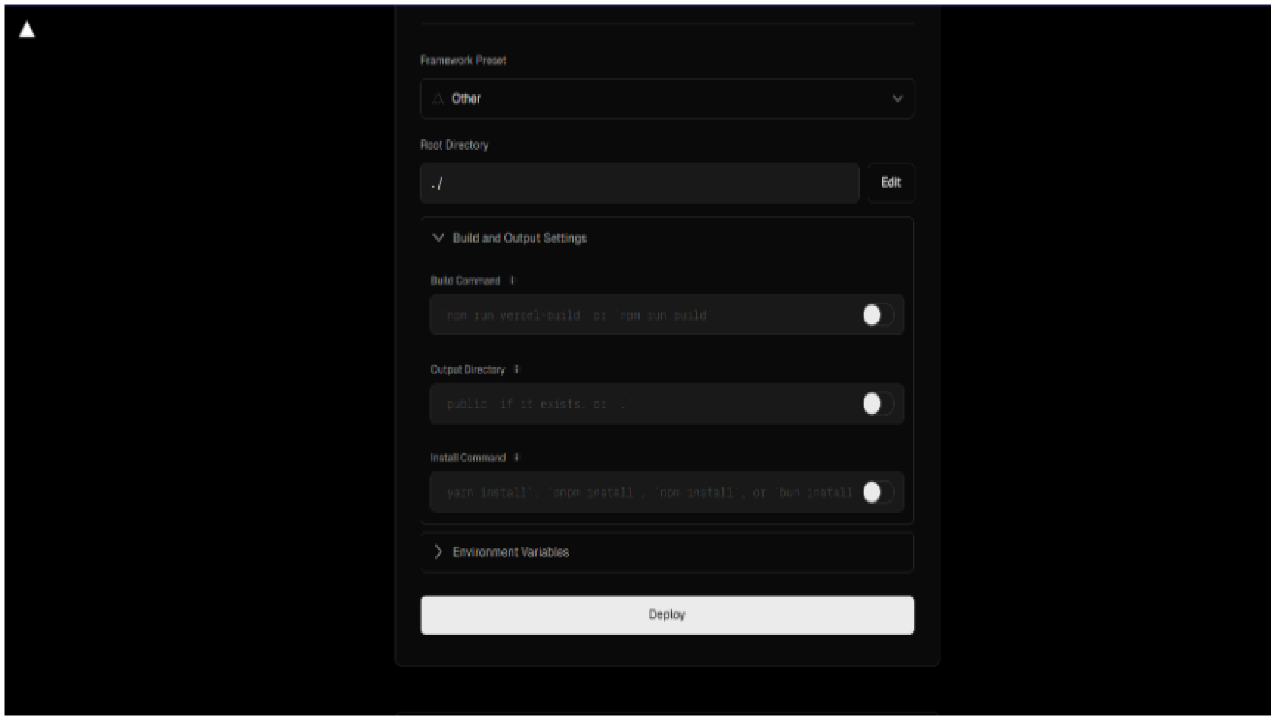
✓ **Read** access to metadata
✓ **Read** and **write** access to administration, checks, code, commit

Select the repository



To deploy a new Project, import an existing Git Repository or get started with one of our Templates.

Import the project


Framework Preset

Other

Root Directory

./

Edit

Build and Output Settings

Build Command

`npm run vercel-build` or `npm run build`

Output Directory

`public` if it exists, or `.`

Install Command

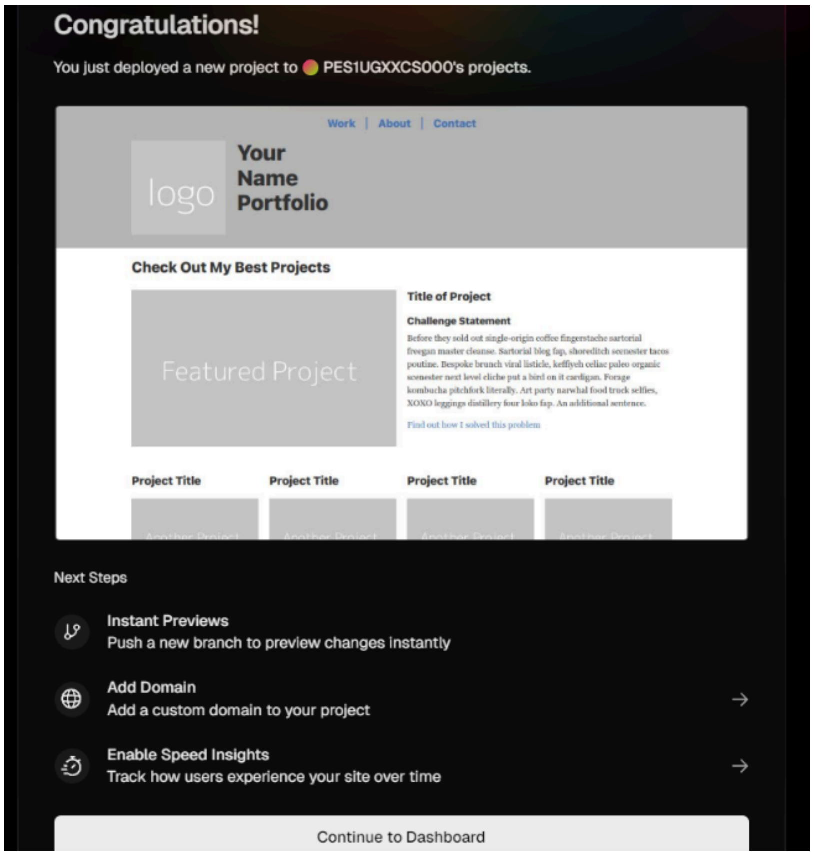
`yarn install`, `pnpm install`, `npm install`, or `bun install`

Environment Variables

Deploy

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!

You just deployed a new project to PES1UGXXCS000's projects.

Next Steps

Instant Previews
Push a new branch to preview changes instantly

Add Domain
Add a custom domain to your project

Enable Speed Insights
Track how users experience your site over time

Continue to Dashboard

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/#/