

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...

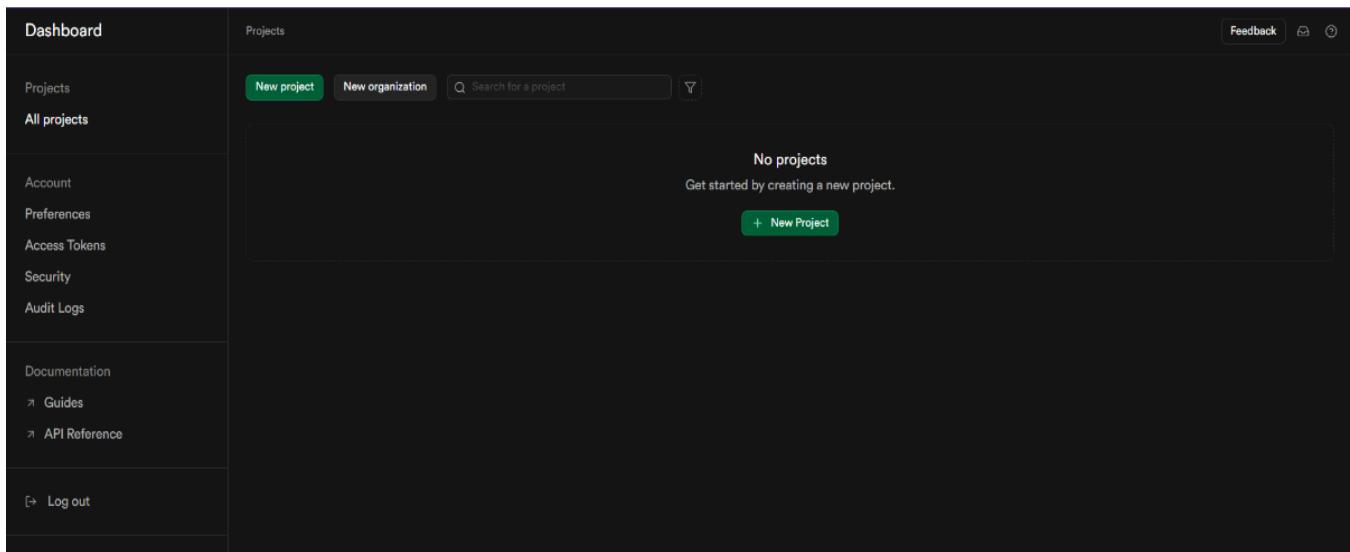
A) IAAS - Infrastructure Backend as a service

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.

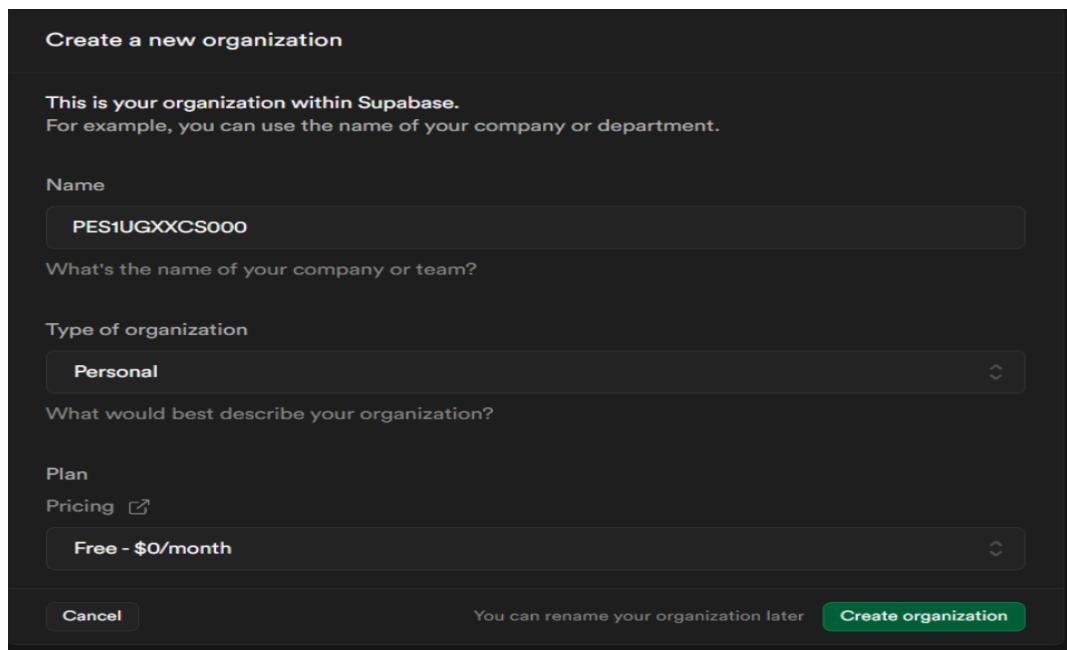
1. Sign up for Supabase account <https://supabase.com/>

Once you do, you should see a page like this



The screenshot shows the Supabase dashboard. On the left, there's a sidebar with links for Dashboard, Projects (highlighted), All projects, Account, Preferences, Access Tokens, Security, Audit Logs, Documentation (with Guides and API Reference), and Log out. The main area is titled 'Projects' and shows a button 'New project'. Below it, a message says 'No projects' and 'Get started by creating a new project.' with a 'New Project' button.

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.



This screenshot shows the 'Create a new organization' dialog box. It has fields for Name (containing 'PES1UGXXCS000'), Company name, Type of organization (set to 'Personal'), Organization description, Plan (set to 'Free - \$0/month'), and Pricing (checkbox checked). At the bottom, there are 'Cancel' and 'Create organization' buttons.

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

Project name

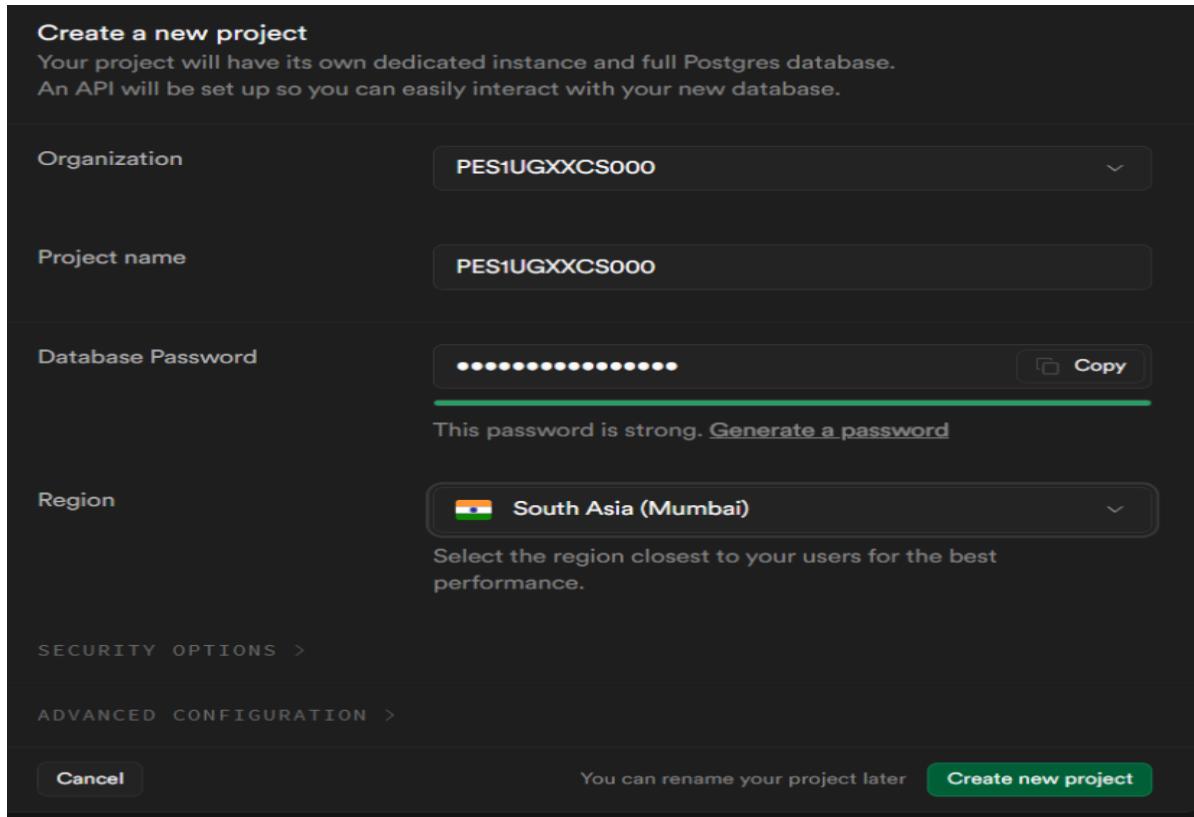
Database Password Copy
This password is strong. [Generate a password](#)

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

SECURITY OPTIONS >

ADVANCED CONFIGURATION >

Cancel You can rename your project later Create new project



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 0 Functions 0 Replicas 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.

[Table Editor](#) [SQL Editor](#) [About Database](#)

	id	task	status
1	1	Create a project	Complete
2	2	Read documentation	Complete
3	3	Build application	In progress
4	4	Connect Supabase	In progress
5	5	Deploy project	Not started
6	6	Get users	Not started
7			
8			

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.

[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://sozngvrpwdiadinjx.supabase.co> [Copy](#)

A RESTful endpoint for querying and managing your database.

Publishable API Key sb_publisheable_Yd3tH7iGDBfK3vVvR0iU4A_ItBQnGgZ [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.

Choose your preferred framework

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

[Connect](#)

The screenshot shows the 'API Keys' section of the Supabase dashboard. On the left, there's a sidebar with 'PROJECT SETTINGS' (General, Compute and Disk, Infrastructure, Integrations, Data API, API Keys, JWT Keys, Log Drains, Add Ons, Vault), 'CONFIGURATION' (Database, Authentication, Storage, Edge Functions), and 'BILLING' (Subscription, Usage). The 'API Keys' section is currently selected. The main area displays two API keys: 'anon public' and 'service_role secret'. The 'service_role' key is highlighted with a red border. A note says: '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.' There are 'Copy' and 'Reveal' buttons for each key.

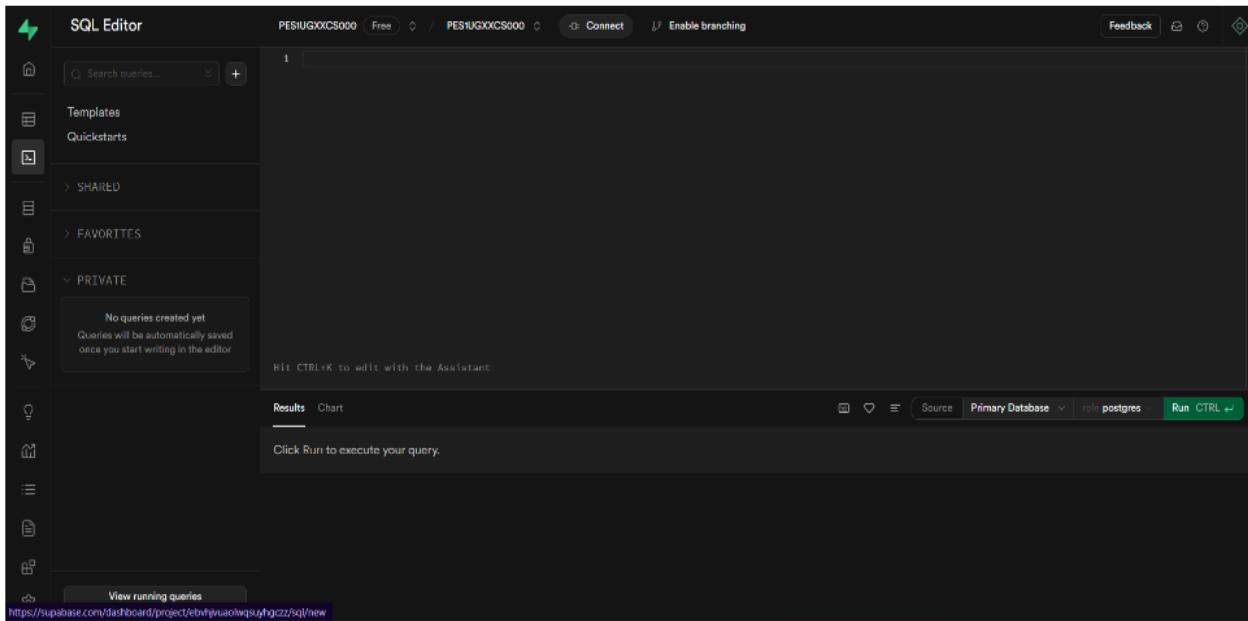
(You don't need to copy the shown api key and url, by the time you see this docs the project would have been deleted XD)

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

The screenshot shows the 'Storage' section of the Supabase dashboard. The left sidebar includes 'Table Editor', 'SQL Editor', 'Database', 'Authentication', 'Storage' (which is selected and highlighted in grey), 'Edge Functions', 'Realtime', 'Advisors', 'Reports', 'Logs', 'API Docs', 'Integrations', and 'Devant Settings'. The main content area is titled 'Storage' and contains the text: 'Create buckets to store and serve any type of digital content. Make your buckets private or public depending on your security preference.' There is a 'About storage' link at the bottom.

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.]

The screenshot shows the Supabase SQL Editor interface. On the left, there's a sidebar with icons for file operations, templates, quickstarts, and database connections. The main area is titled "SQL Editor" and shows a query being run:

```
1 create policy "bucket_insert"
2 ON storage.buckets
3 for insert with check (
4 | true
5 );
6
7 create policy "object_insert"
8 ON storage.objects
9 for insert with check (
10 | true
11 );
```

The results section below shows the message "Success. No rows returned" and indicates "0 row". The top right of the interface includes buttons for "Feedback", "Connect", "Enable branching", and other UI elements.

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.

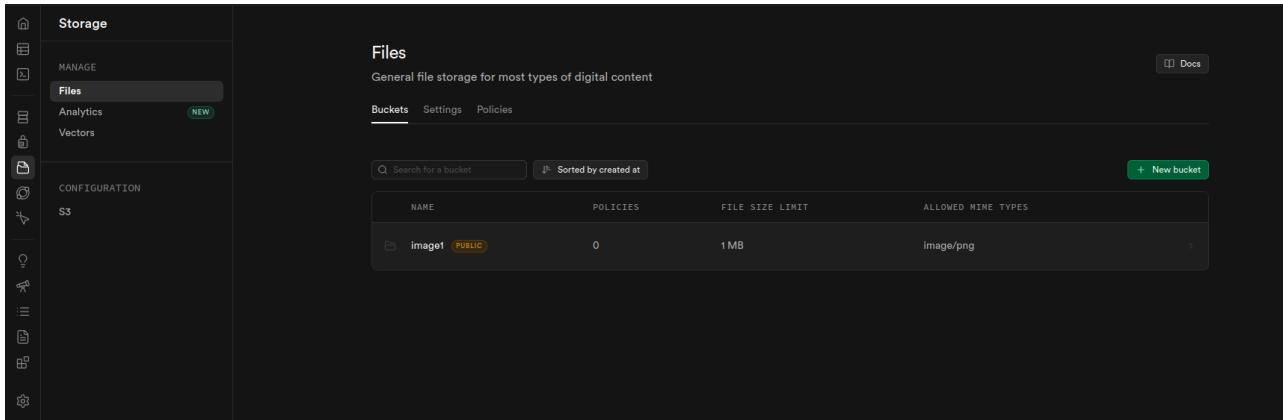
10. Upload your image in jpeg format in the same folder where you have all other files and rename your image to your PES1UGXXCSXXX.jpeg

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)

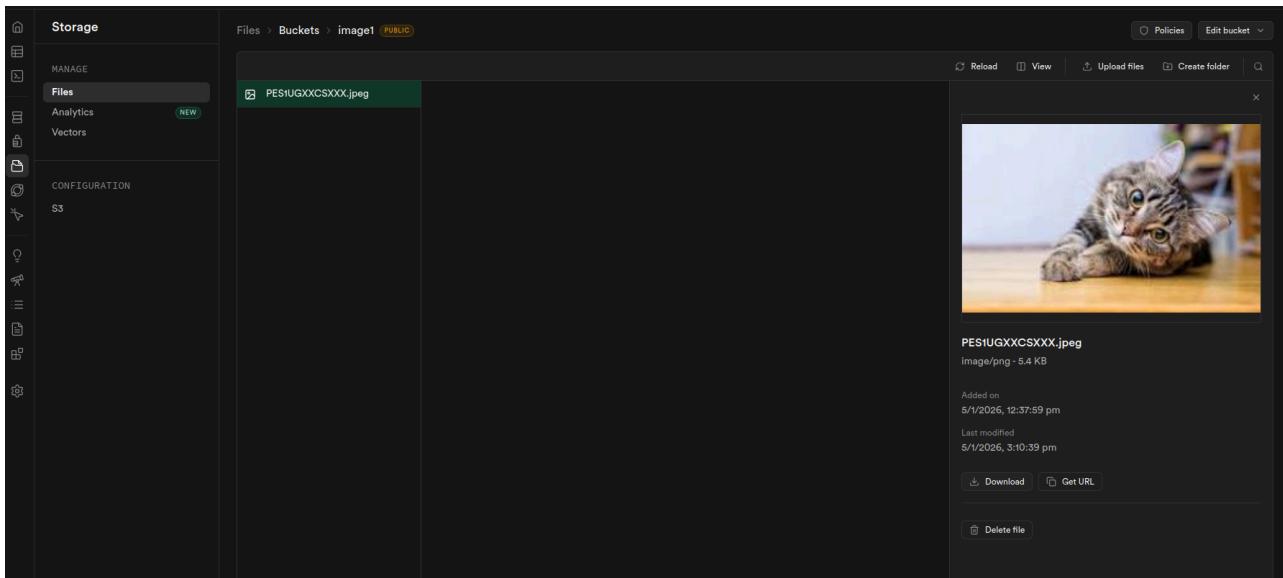
```
ccbd@ccbd-ThinkStation-P620:~/Desktop/cc26/CC-Lab1$ python3 supabase_object_store.py
Storage endpoint URL should have a trailing slash.
Bucket 'image1' created successfully.
ccbd@ccbd-ThinkStation-P620:~/Desktop/cc26/CC-Lab1$ 
```



- 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$
```



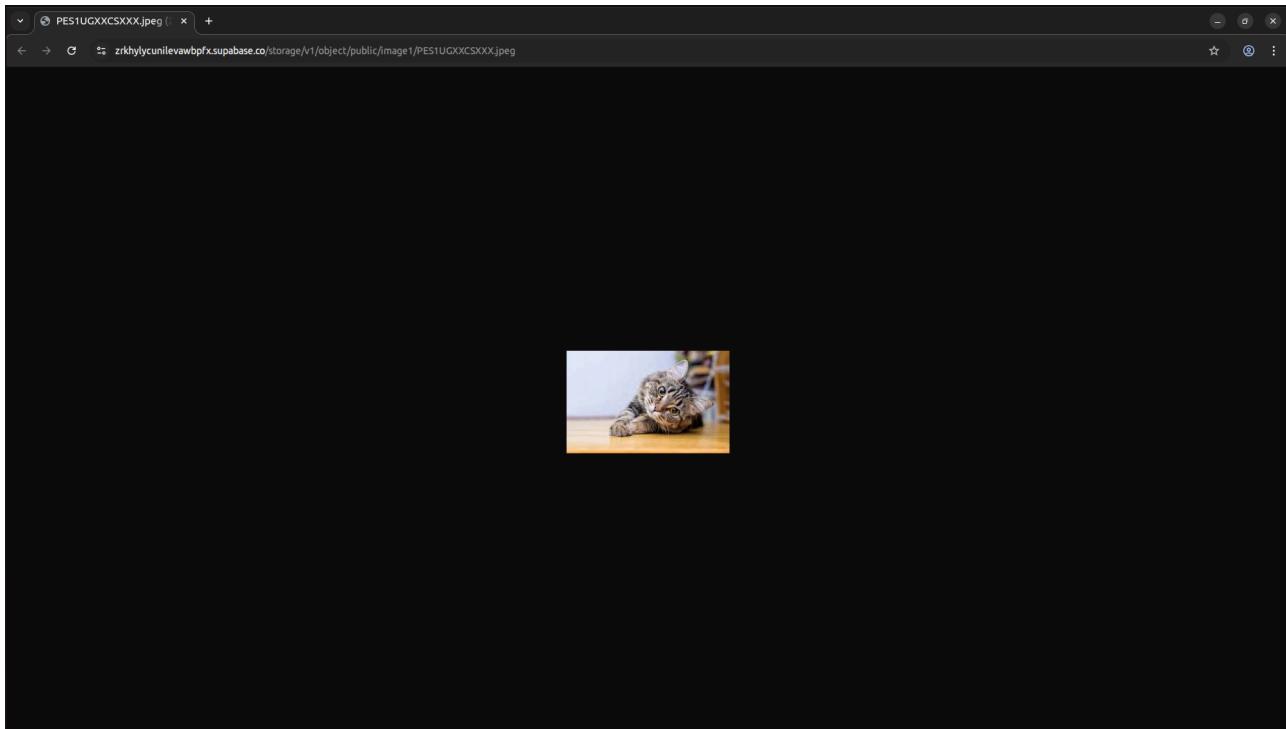
(uncomment step 3 and comment step 1 and 2)

```
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 'Beginner: Free!' plan box with a yellow header. It describes it as 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!'. Below this is a 'Create a Beginner account' button. To the right is an 'Education accounts' section with a yellow header, stating '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.' At the bottom of the page, there are four main plan categories: 'Hacker' (\$5/month), 'Web dev' (\$12/month), 'Startup' (\$99/month), and 'Custom' (\$5 to \$500/month). Each category has a detailed description of its features and limitations.

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	A Python IDE in your browser with unlimited Python/bash consoles	Up to 2 web apps on custom domains or your-username.pythonanywhere.com	Start a business and don't worry about having to scale to handle traffic spikes	A Python IDE in your browser with unlimited Python/bash consoles	A Python IDE in your browser with unlimited Python/bash consoles	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	One web app on a custom domain 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)	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	As many web workers as you need to scale your site's capacity. (more info)
Enough power to run a typical 100,000 hit/day website (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)	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.**

The screenshot shows the PythonAnywhere dashboard for user PES1UG23CSXXX. At the top, it displays CPU Usage (4% used) and File storage (0% full). Below this, there are sections for Recent Consoles, Recent Files, Recent Notebooks, and All Web apps. The Recent Notebooks section notes that the account does not support Jupyter Notebooks and suggests upgrading. The All Web apps section shows a single entry: PES1UG23CSXXX.pythonanywhere.com. On the left, there's a 'New console' dropdown with options like '\$ Bash', '>>> Python', and 'More...'. The bottom of the dashboard features a navigation bar with links for Send feedback, Forums, Help, Blog, Account, and Log out.

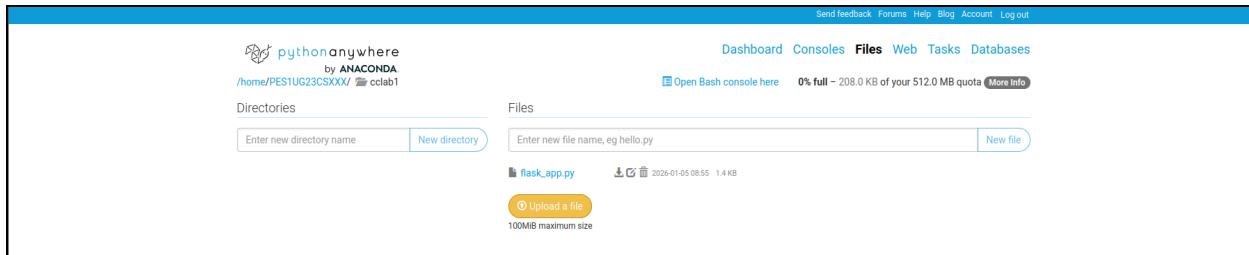
Step 3: Create Project Directory

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

The screenshot shows the PythonAnywhere Files interface. The user is in the /home/PES1UG23CSXXX directory. A new directory named 'cclab1' has been created and is selected. The 'Files' tab is active, showing a list of files including .bashrc, .gitconfig, .profile, .pythonstartup.py, .vimrc, and README.txt. There is also a 'New file' button and an 'Upload a file' button with a 100MB maximum size limit. The top navigation bar includes links for Dashboard, Consoles, Files, Web, Tasks, Databases, and a note about file storage (0% full).

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.**



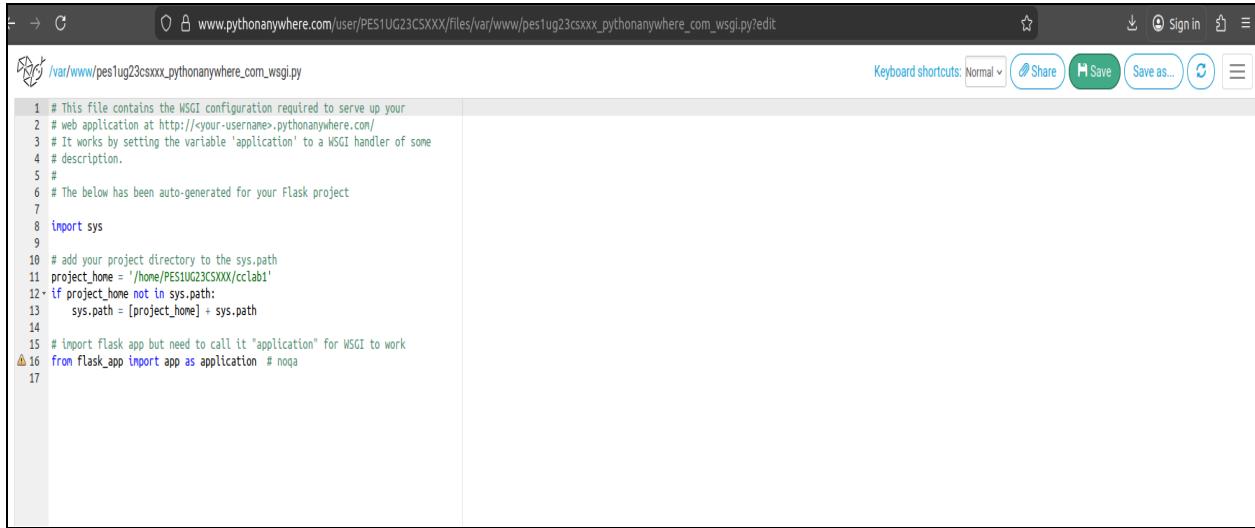
Step 5: Configure Web App Source Code Path

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/cclab1`

A screenshot of the PythonAnywhere Web configuration page. It starts with a 'Traffic:' section showing 'How busy is your site?'. Below that are statistics for 'This month (previous month)', 'Today (yesterday)', and 'Hour (previous hour)'. Then there's a note about getting pretty charts for paying accounts. Next is a 'Code:' section with 'What your site is running.' It lists 'Source code' as '/PES1UG23CSXXX/cclab1', 'Working directory' as '/home/PES1UG23CSXXX/', 'WSGI configuration file' as '/var/www/pes1ug23csxxx_pythonanywhere_com_wsgi.py', and 'Python version' as '3.12'. To the right of these fields are 'Go to directory' buttons. Below this is a 'Virtualenv:' section with a note about using a virtualenv for different software versions. It includes a text input field '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'`

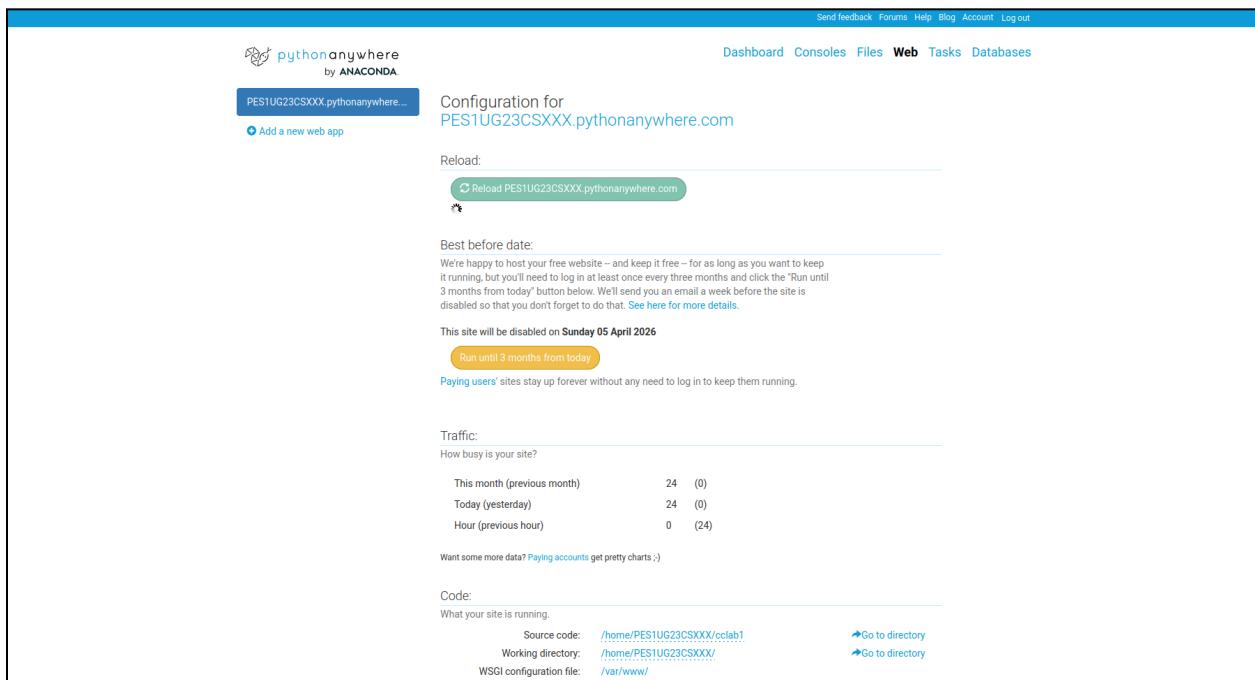


```

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/cclab1'
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**.



Configuration for **PESTUG23CSXXX.pythonanywhere.com**

Reload: [Reload PESTUG23CSXXX.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](#)

Traffic:
How busy is your site?

	24 (0)
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/wsgi.py	Go to directory

Step 7: Access the Deployed Application

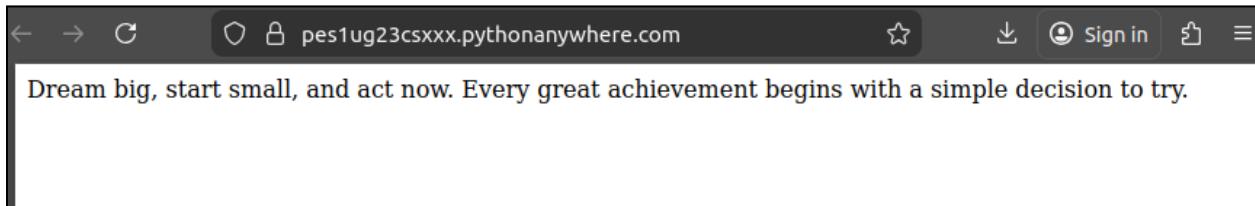
1. Click on the application link displayed under the **Web** tab configuration section.

Configuration for PES1UG23CSXXX.pythonanywhere.com

Reload:

 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.



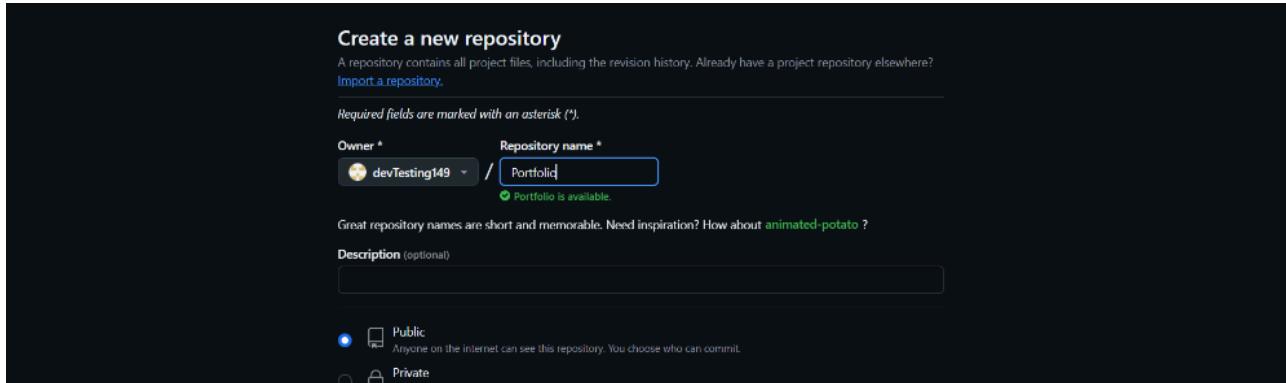
Result

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).**

A screenshot of a GitHub repository named 'Portfolio'. The repository is public. It contains one branch ('main') and no tags. There is one commit by 'dev-testing123' with the message 'PESXUGXXCSXXX'. The commit was made 'now' and has 2 commits. The repository has 0 stars, 0 forks, and 1 watching. It also has 0 releases and 0 packages.

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

The screenshot shows the 'Account Settings' page in the Vercel dashboard. On the left, there's a sidebar with links like 'General', 'Authentication', 'Sign in with Vercel', 'Billing', 'Invoices', and 'Tokens'. The main area is titled 'Authentication' with the sub-instruction 'Connect your Vercel Account with a third-party service to use it for login, or create a passkey.' Below this is a 'Add New' button and three service buttons: GitHub, GitLab, and Bitbucket. A 'Passkey' button is also present.

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

5. Now we will import project from github and deploy

The screenshot shows the 'Account Overview' page. On the left, there are sections for 'Teams' and 'Domains'. The main area is titled 'Teams' with the sub-instruction 'The teams that are associated with your Vercel account.' It includes a search bar 'Search for a team...' and a list item for 'PESIUGXCS000's projects Hobby Owner'. On the right, there's a sidebar with options like 'Dashboard', 'Account Settings', 'Create Team', 'Command Menu', 'Theme', 'Home Page', 'Log Out', and an 'Upgrade to Pro' button. The 'Create Team' button is highlighted with a red box.

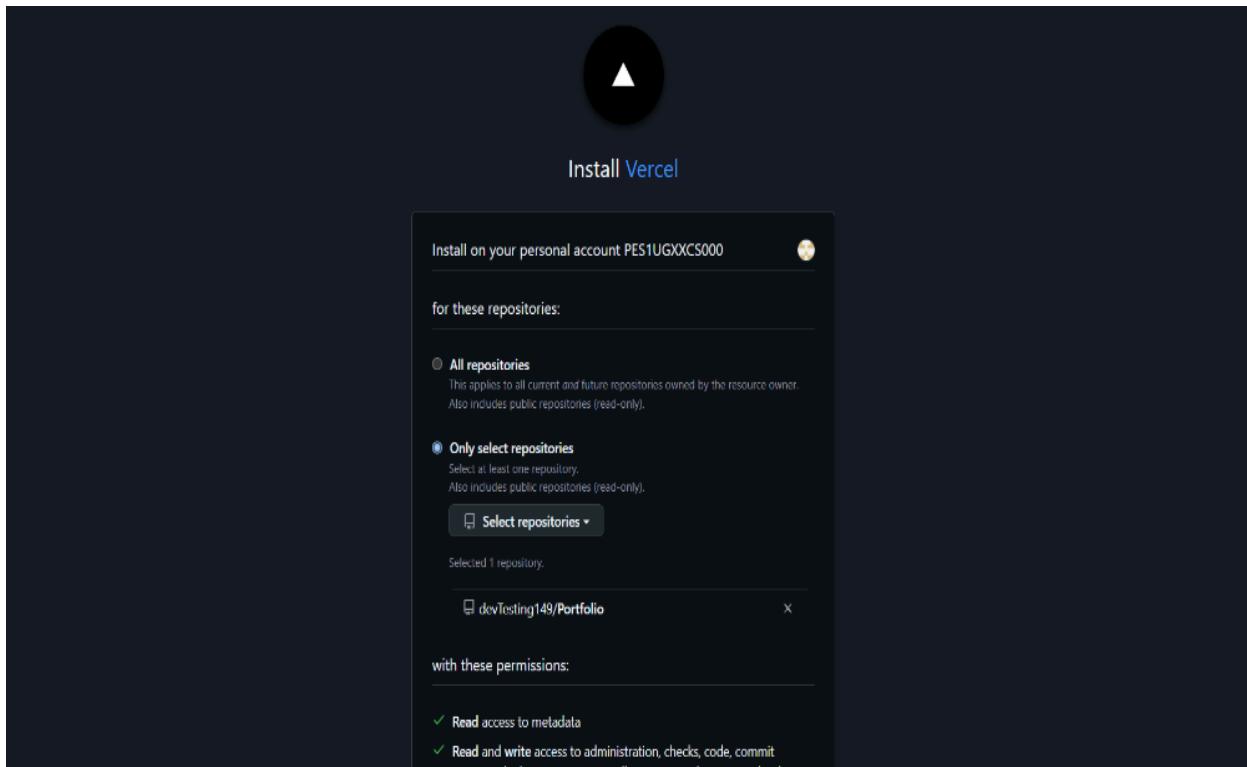
Go to dashboard

The screenshot shows the Vercel dashboard with the 'Overview' tab selected. At the top, there's a navigation bar with links for Overview, Integrations, Activity, Domains, Usage, Monitoring, Observability, Storage, AI, Support, and Settings. Below the navigation, a section titled 'Deploy your first project' encourages users to start with one of their templates or create something new. It features five cards: 'Import Project' (with an Import button), 'Next.js Boilerplate' (with a Deploy button), 'AI Chatbot' (with a Deploy button), 'Commerce' (with a Deploy button), and 'Vite + React Starter' (with a Deploy button). At the bottom of this section is a 'Browse Templates' button with icons for Next.js, SvelteKit, Vite, and Tailwind CSS.

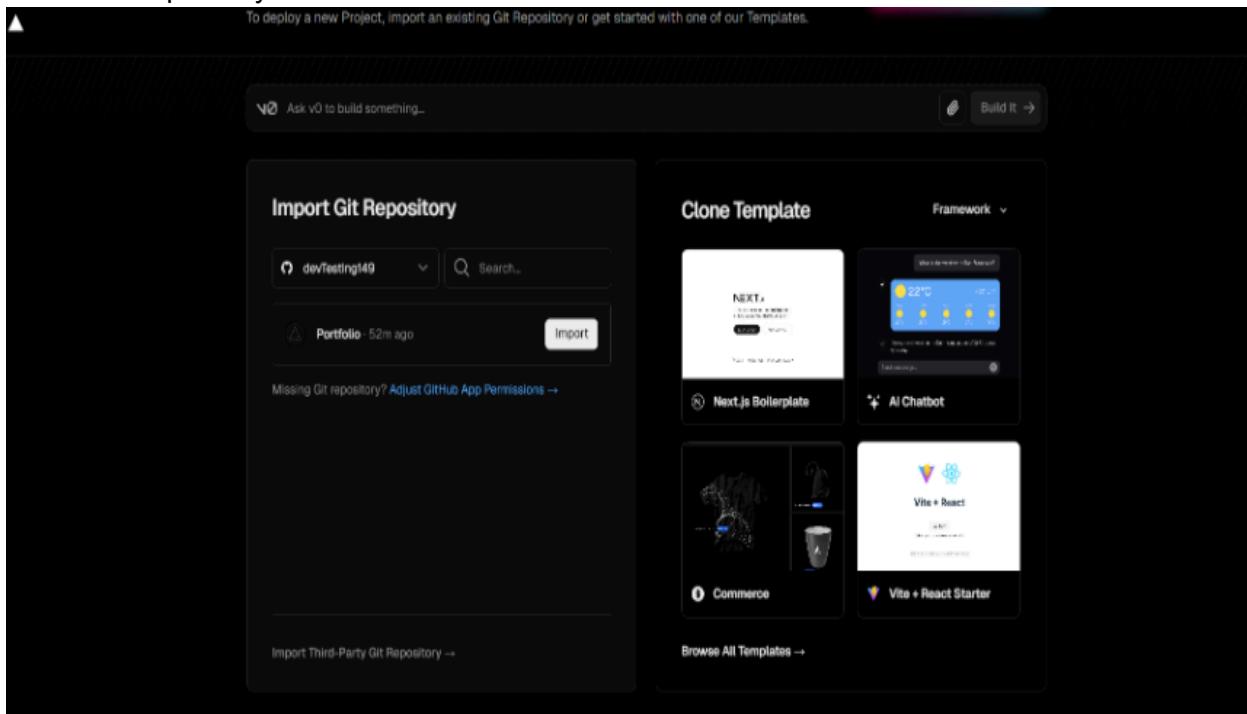
Import project

This screenshot shows the 'Import project' screen. At the top, it says 'To deploy a new Project, import an existing Git Repository or get started with one of our Templates.' Below this, there are two main sections: 'Import Git Repository' and 'Clone Template'. The 'Import Git Repository' section contains a search bar for GitHub names and a note about installing the GitHub application. The 'Clone Template' section displays four template cards: 'Next.js Boilerplate' (a weather app), 'AI Chatbot' (an AI chatbot), 'Commerce' (a commerce app), and 'Vite + React Starter' (a Vite and React starter project). A 'Browse All Templates →' link is at the bottom right of the template section.

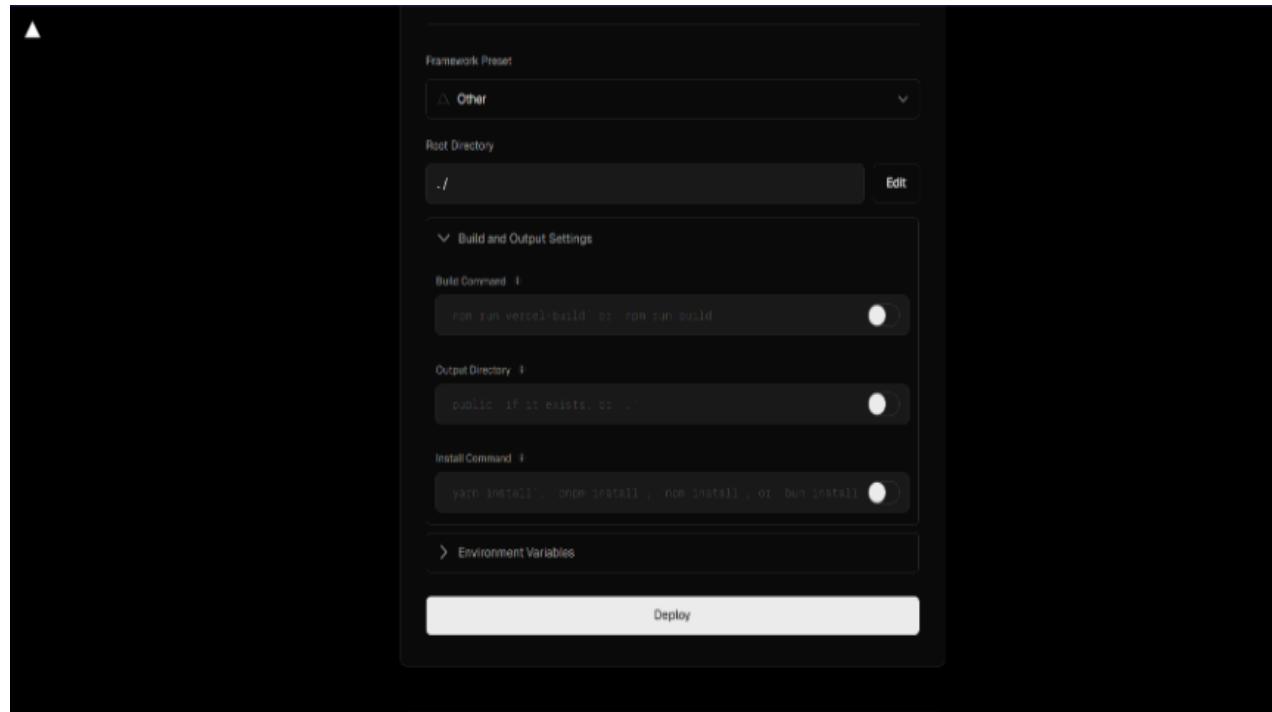
Install



Select the repository



Import the project



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/>

A screenshot of a deployed portfolio website. At the top, it says 'Congratulations!' and 'You just deployed a new project to PES1UGXXCS000's projects.' Below this is a dark header with 'Work | About | Contact' and a placeholder 'Your Name Portfolio' with a logo icon. The main content area has a heading 'Check Out My Best Projects' and a 'Featured Project' section with a challenge statement. Below this are four project cards with titles like 'Project Title'. At the bottom, there's a 'Next Steps' section with 'Instant Previews', 'Add Domain', and 'Enable Speed Insights' options, each with a right-pointing arrow. A 'Continue to Dashboard' button is at the very bottom.

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/#/>