Deploy Guide Doc

disclaimer1: this guide is written in 26/10/2025 so there may be some more or less significant change to the step of obtaining the api key in the section 2. since the UI of supabase has changed from the past and could be change in the unknown future so there is a possibility that this guide will be outdate.

disclamer2: the writer of the guide use ubuntu and hence all the command line here written in that environment perspective, so if you are using window CMD be sure to look for the right command such as navigating yourself to the correct path, but ideally if you're using window please use wsl if possible.

1. Docker deployment (for testing on local machine)

1.1 firstly clone the repo from https://github.com/thanyakwaonueng/project_se via command

$git clone https://github.com/thanyakwaonueng/project_se GitHub - thanyakwaonueng/project_se

1.2 once the project is clone to your local machine navigate to the root of your project from your terminal then excute the build_docker.sh script with $./build_docker.sh, if the script is not excutable yet do $chmod + x build_docker.sh to make script excutable then $./build_docker.sh

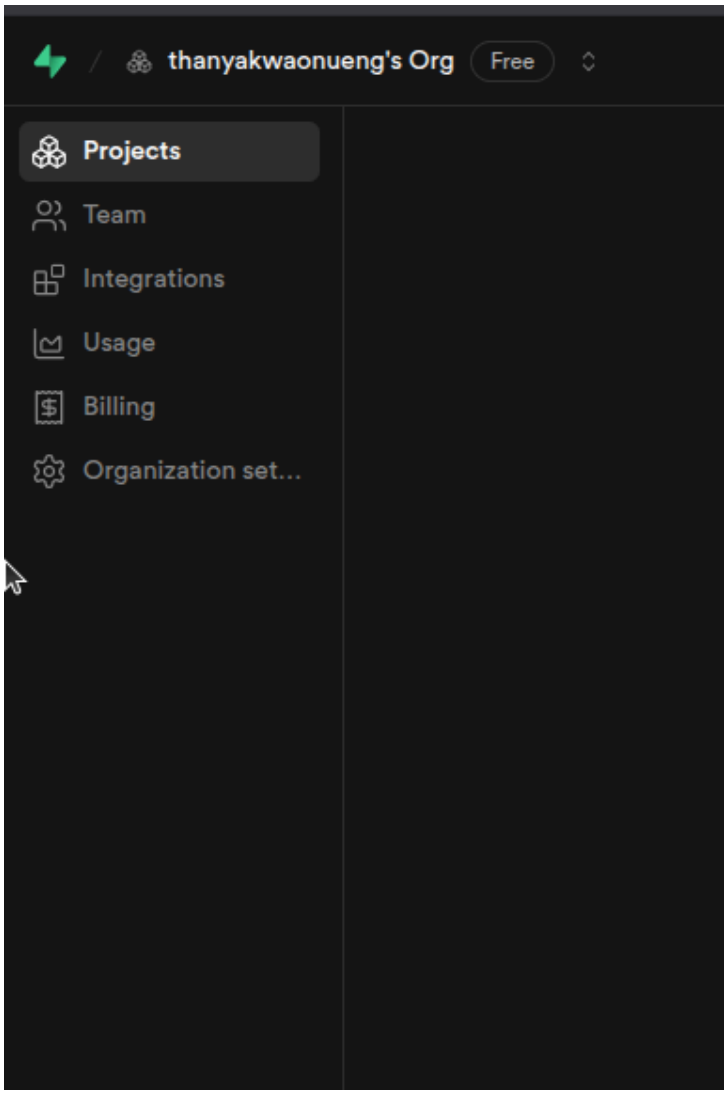
1.3 after this the application should be correctly deploy to docker container, which consist of 3 container

2. Production deployment

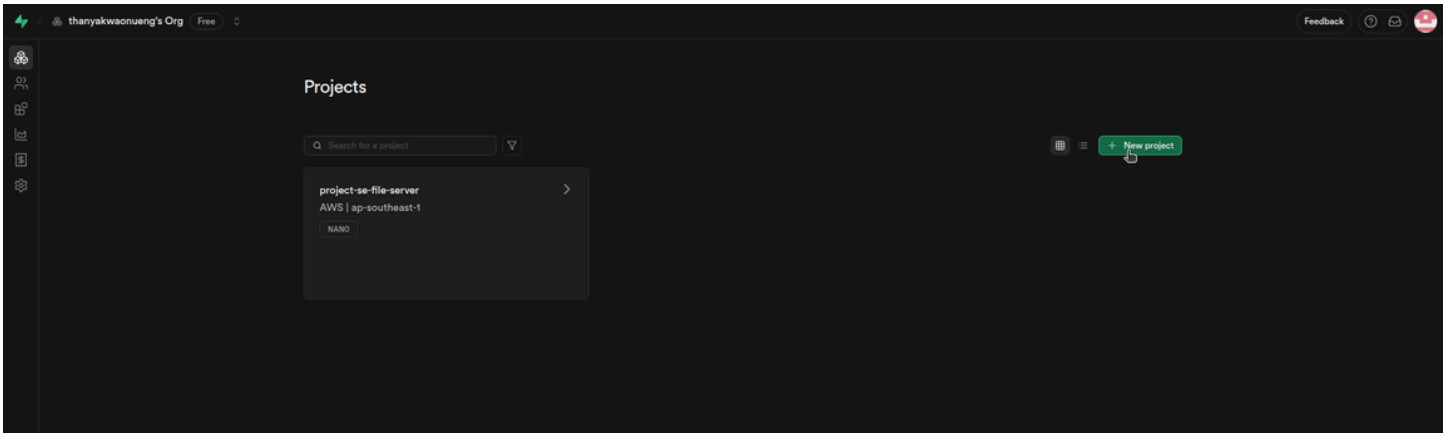
2.1 Setting Up the Database in Supabase

2.1.1 go to https://supabase.com, then signup/signin, then supabase will ask you to create the organization which is like workspace please proceed to do so

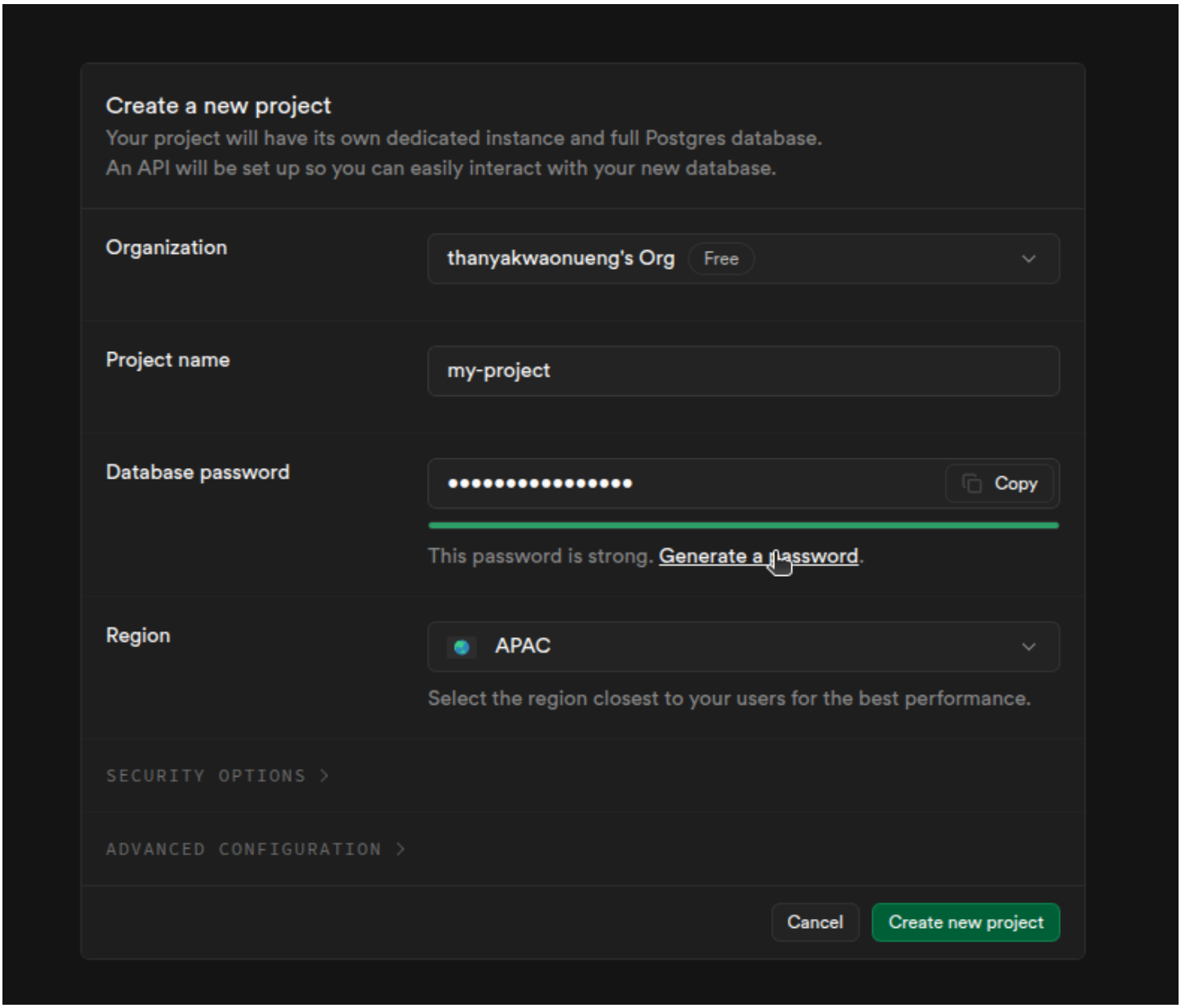
2.1.2 next navigate to Projects in the left navbar



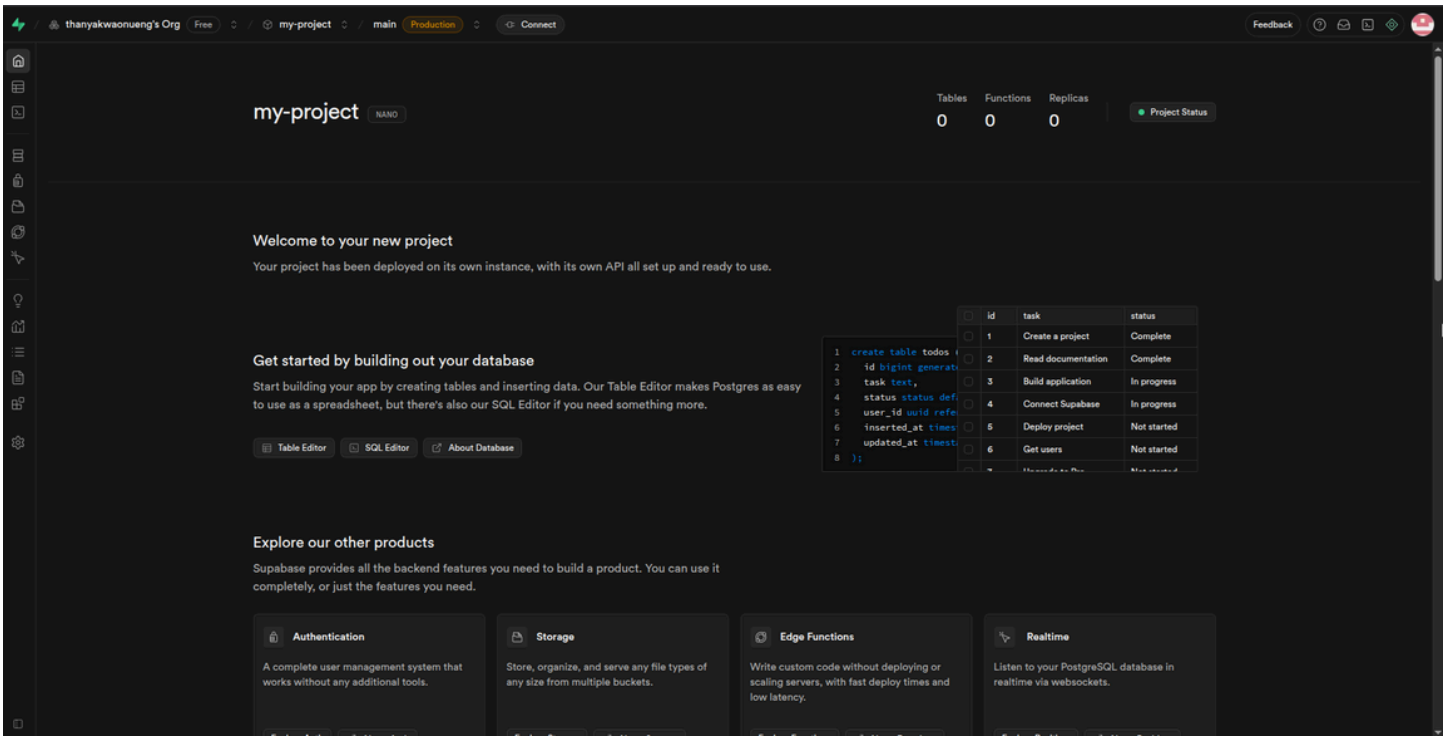
2.1.3 then click the green button that cursor is pointing in the image, + New project button



2.1.4 then fill in the project name, choosing region, then database password, **noting that database password will be include in the credential in the env of production deployment be sure to copy it on clipboard and write it down somewhere so you not forget, also you can use the generate a password button for it in case you don’t come up with any sensible one

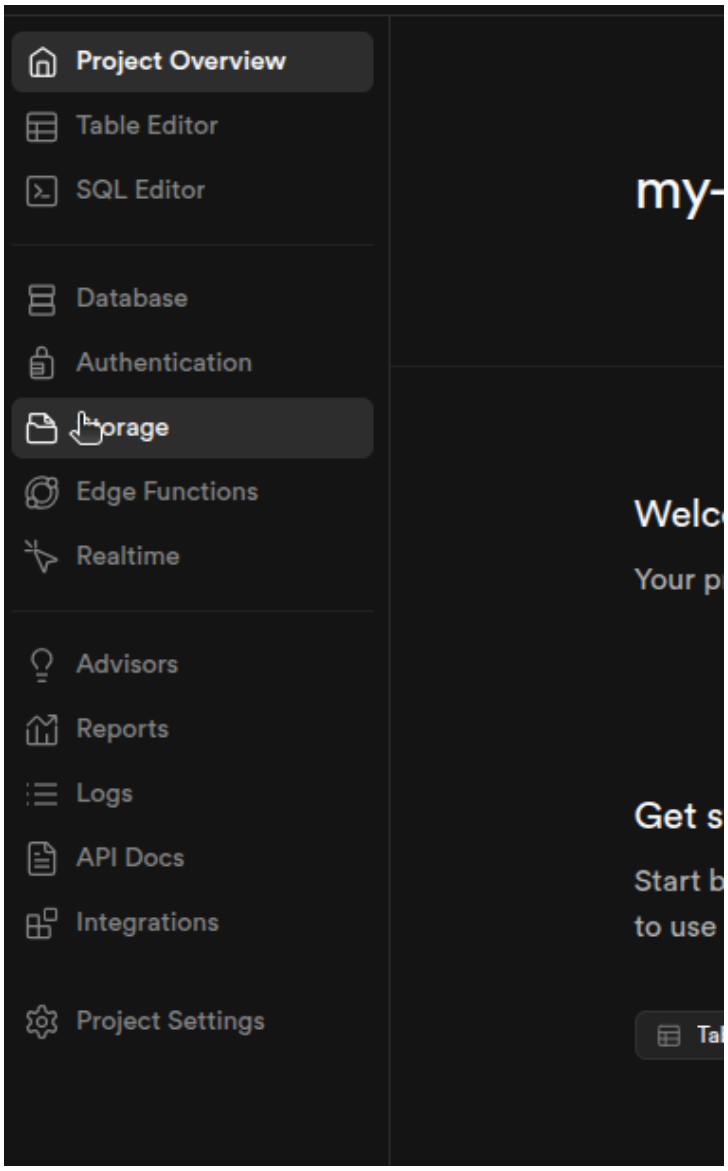


2.1.5 once project created supabase also automatically creates a PostgreSQL database for you too,

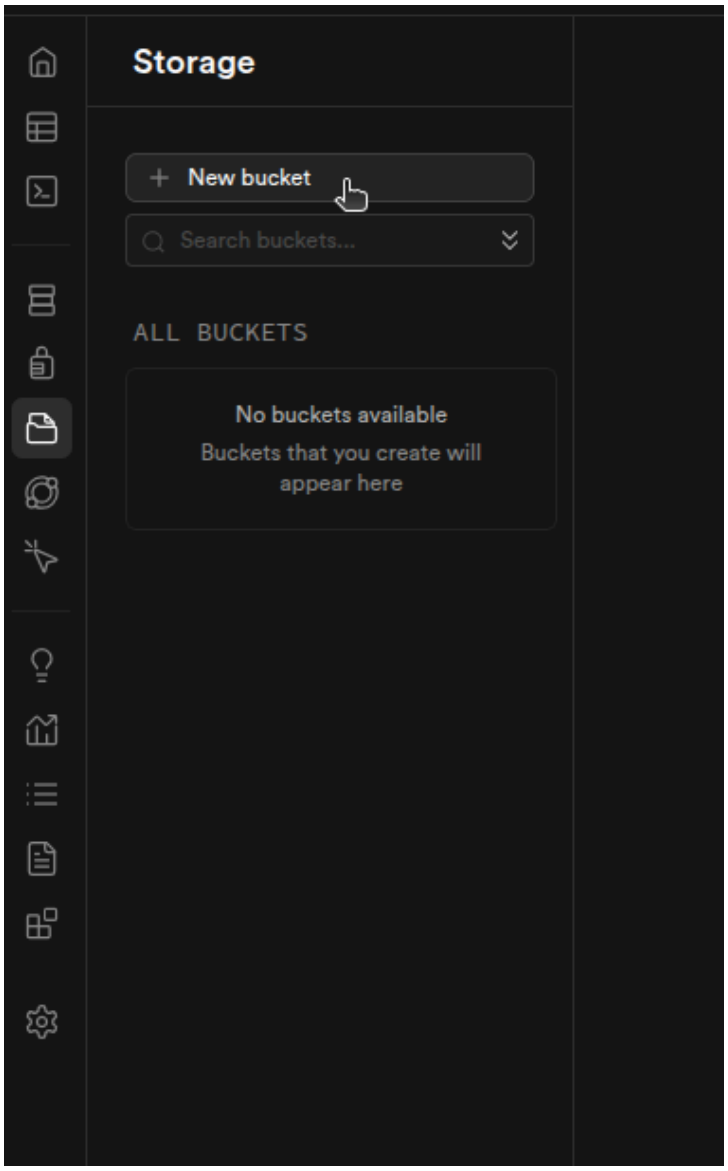


2.2 Setting Up the File Server(Bucket) in Supabase

2.2.1 navigate to the left navbar then click on storage

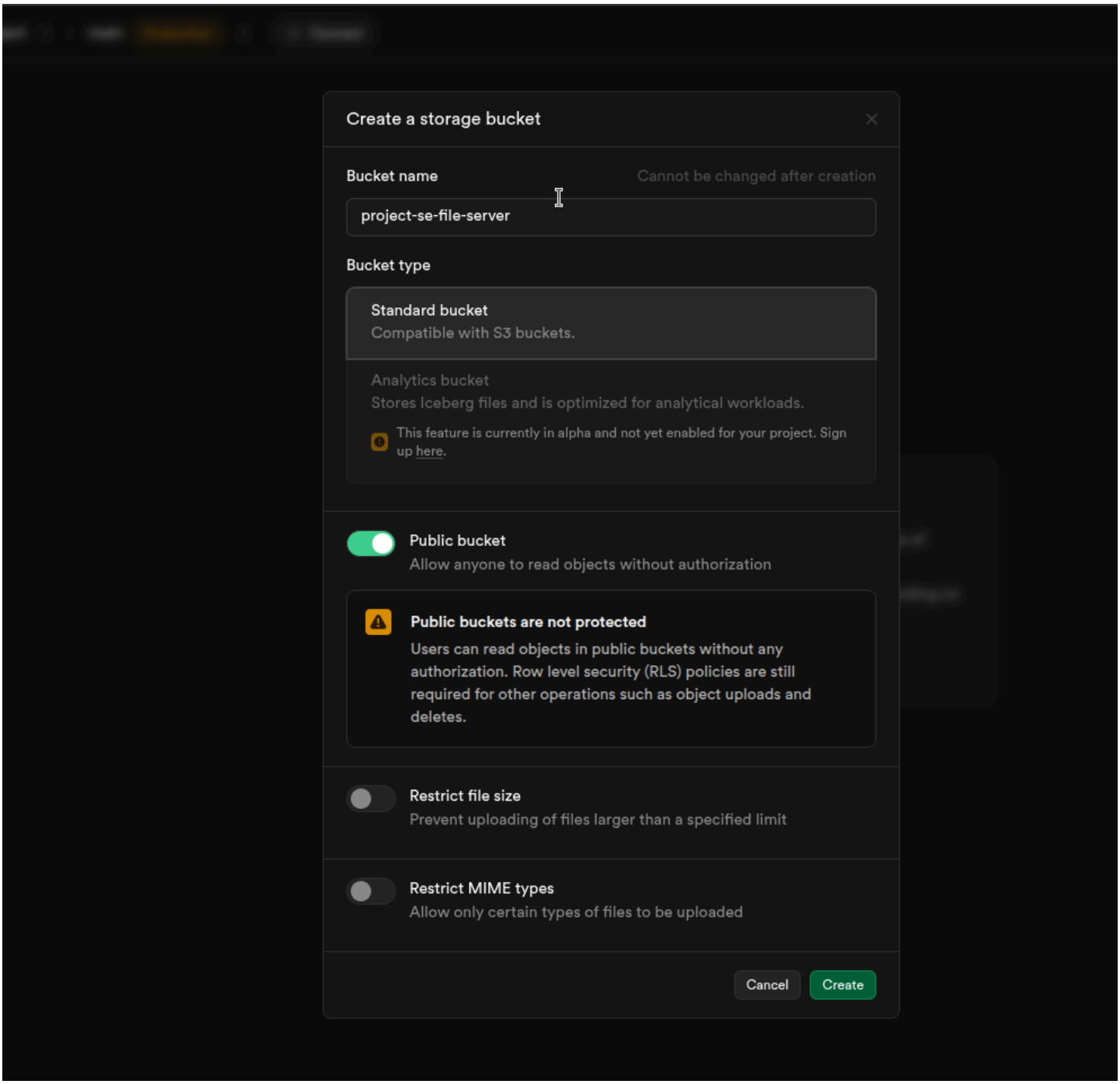


2.2.2 click on + New bucket

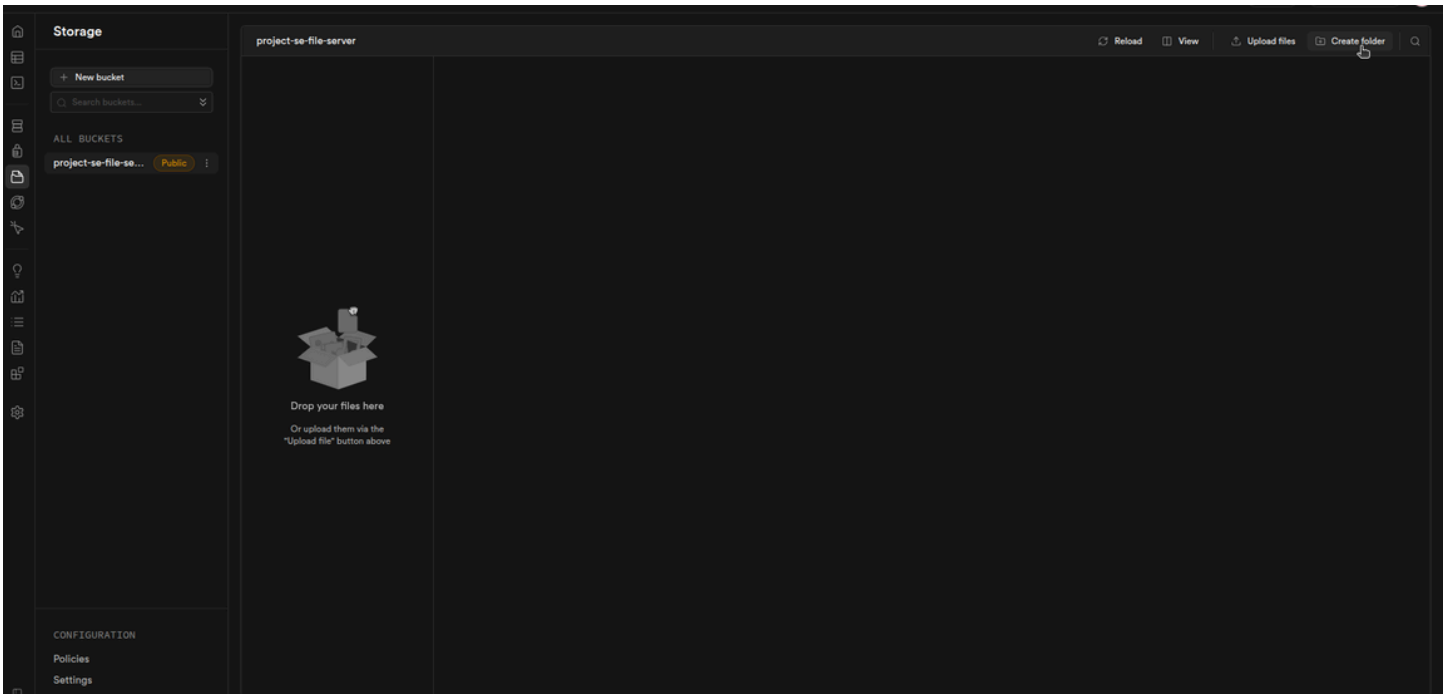


2.2.3 for simplicity of following the guide the bucket policy setting up in the tutorial here may be weak and therefore vulnerable, so be sure to set up the bucket policy yourself as you see appropiate for the real use in production, **also due to the technical reason which is the code in almost last line responsible for uploading file to the bucket has predetermined name which is “project-se-file-server” so again for sake of simplicity here please name it project-se-file server or modify code in the line 3614 of project_se/backend/index.js which responsible for bucket name to be the name you more desired.

then you click the create button.

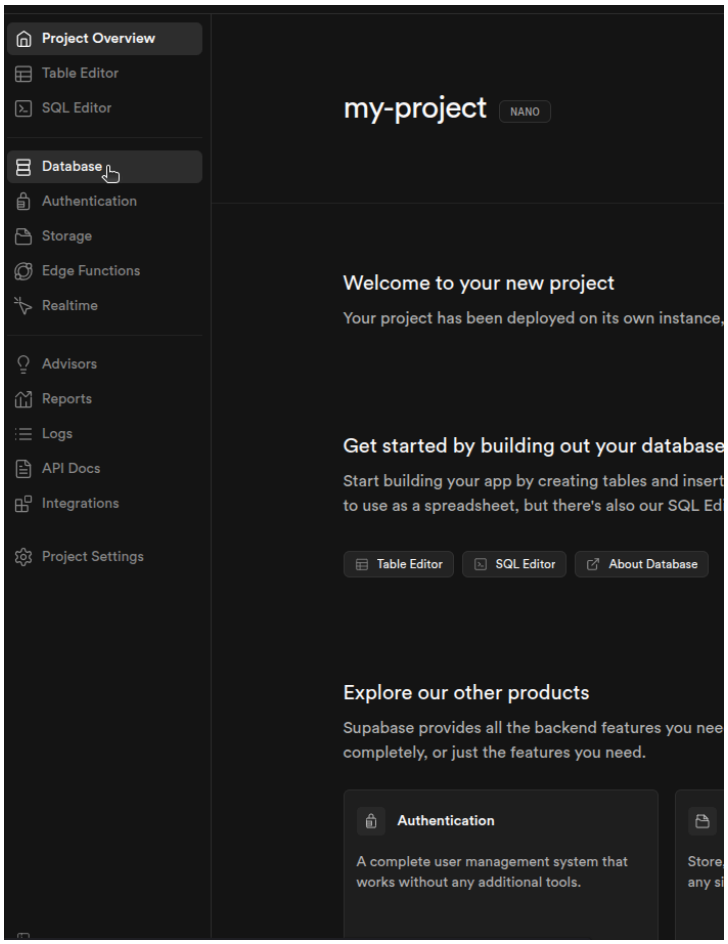


2.2.4 at the right corner where the cursor pointing to click the create folder button, and again due to the technical reason you have to name it “user-uploads” otherwise the whole thing will not work, nevertheless you can change it by modify the line 3615 in project_se/backend/index.js to be your more desired name.

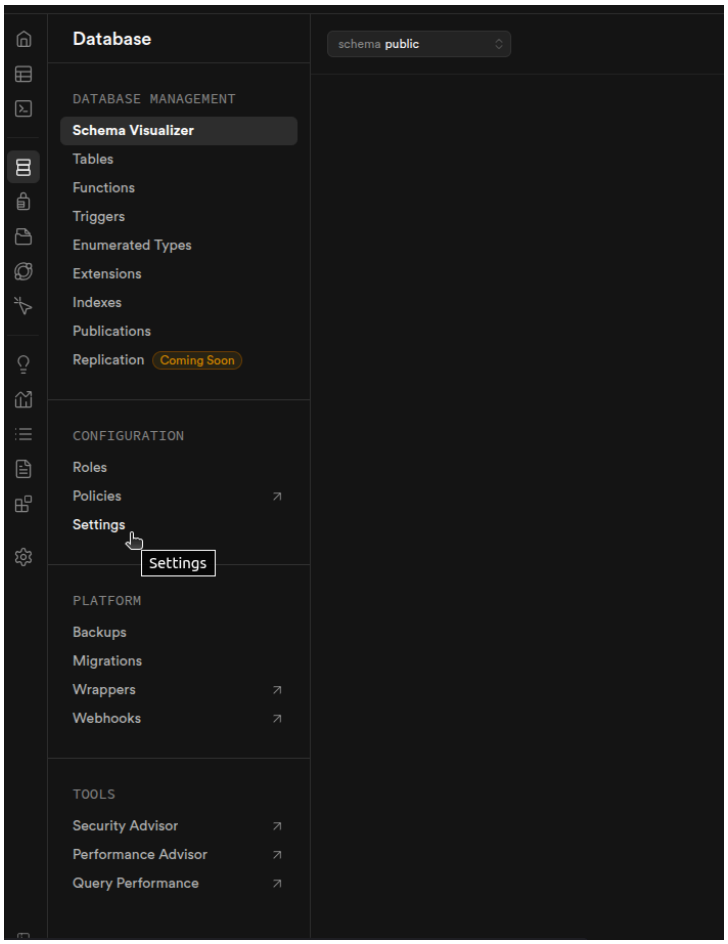


2.3 saving all credential for using in production env

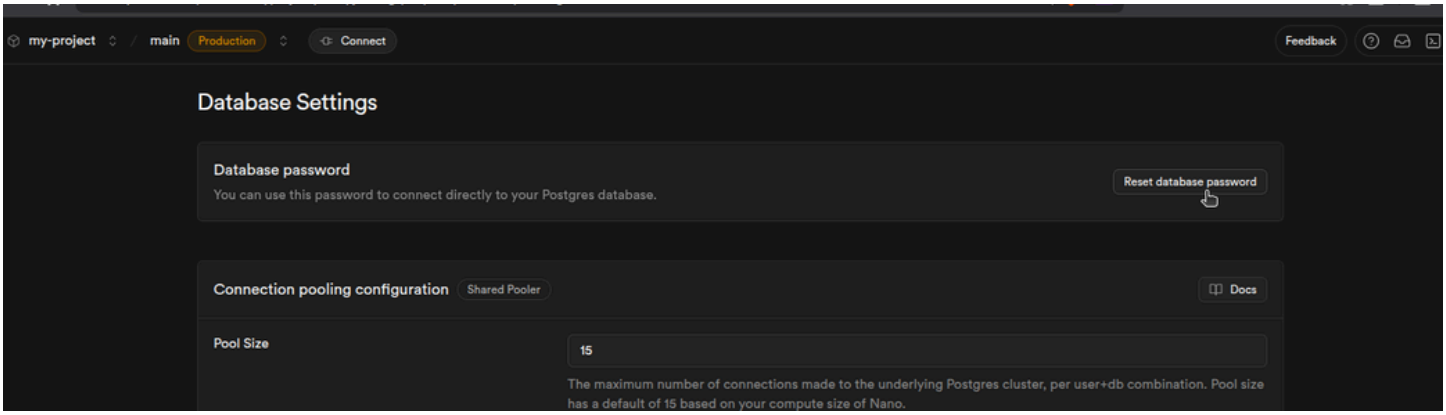
2.3.1 as mentioned earlier you must have save your database password now if you already did skip to step 2.3.4, but in case you forgot you can always reset the database password by navigating to database in left navbar here



2.3.2 then click on setting



2.3.3 then click on reset database password here where my cursor pointing to, then proceed to generate your new database password



2.3.4 all the credential necessary for setting up application consist of

JWT_SECRET, SUPABASE_URL, SUPABASE_SERVICE_KEY, DATABASE_URL, DIRECT_URL, GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET, EMAIL_USER, EMAIL_PASS, VITE_GOOGLE_CLIENT_ID

which I will show here step by step on how to get each one of these, make sure to save/write it down somewhere so you not forget

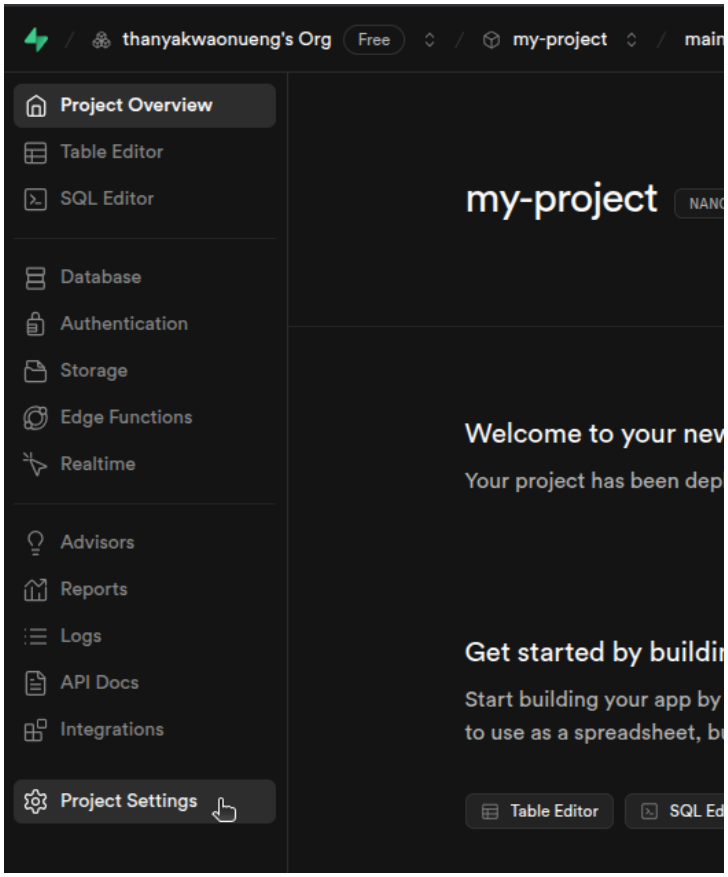
2.3.4.1 JWT_SECRET

you can generate a random jwt_secret from your termial you can do it many ways but here we are using OpenSSL, navigate to your terminal, run command $openssl rand -hex 32

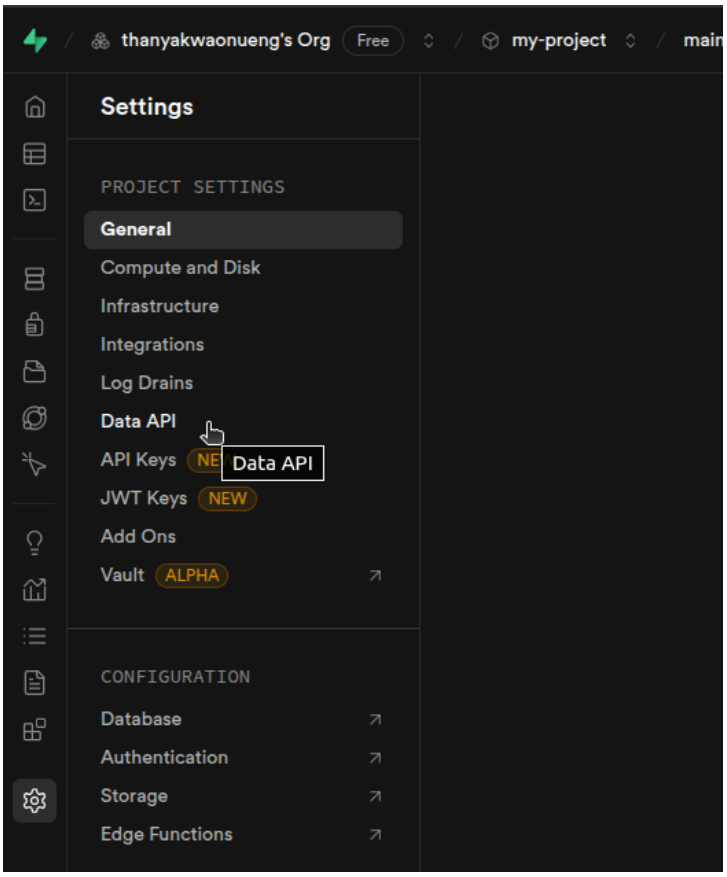
then save that string as your JWT_SECRET

2.3.4.2 SUPABASE_URL

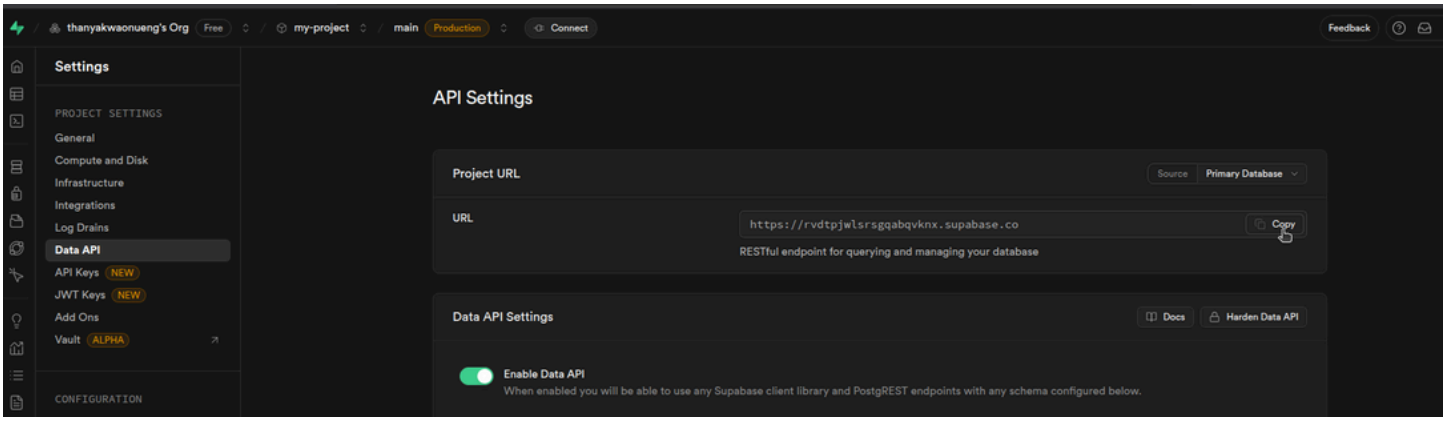
navigate to project settings in your left navbar in supabase



then navigate to Data API

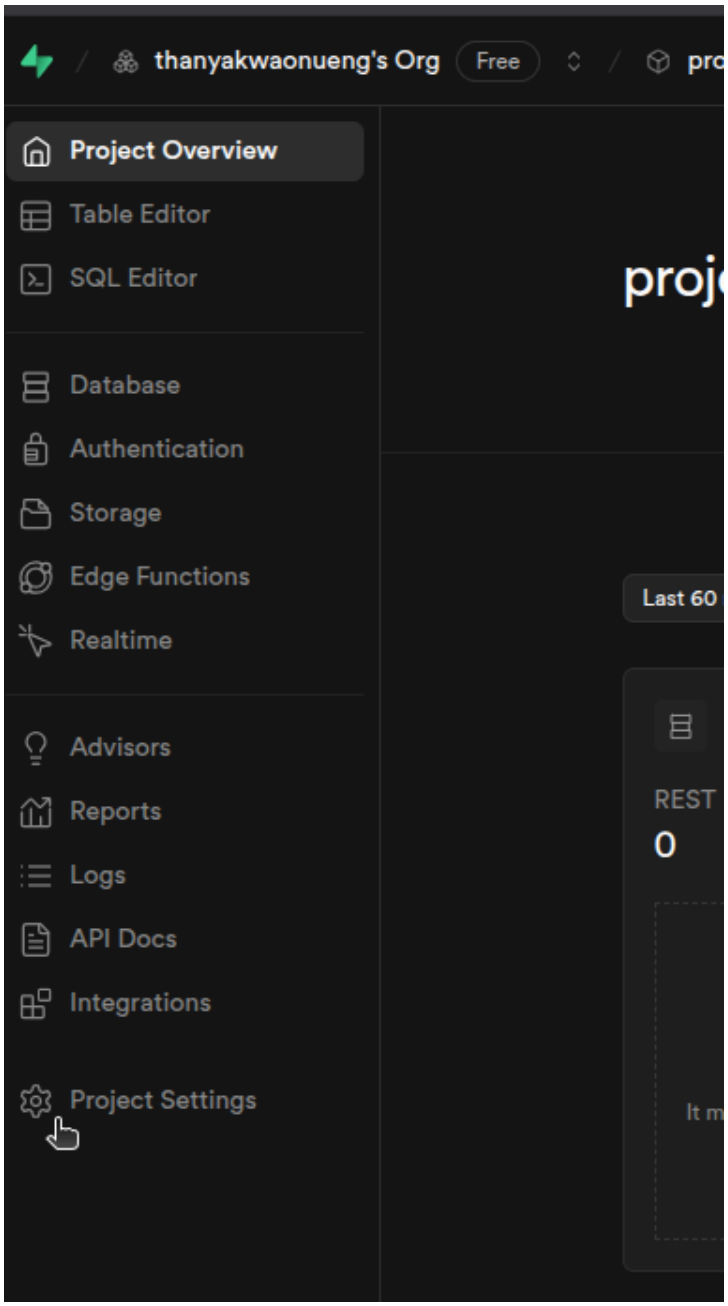


then copy your project url, thats gonna be your SUPABASE_URL credential

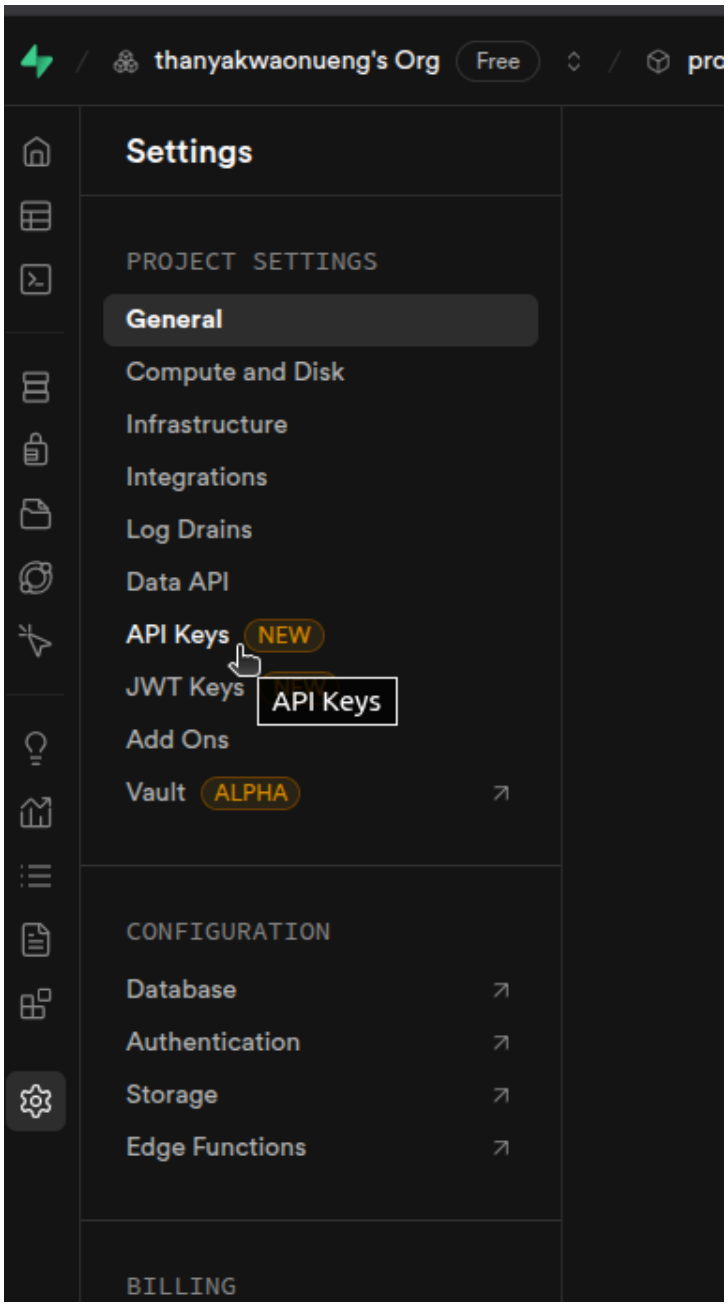


2.3.4.3 SUPABASE_SERVICE_KEY

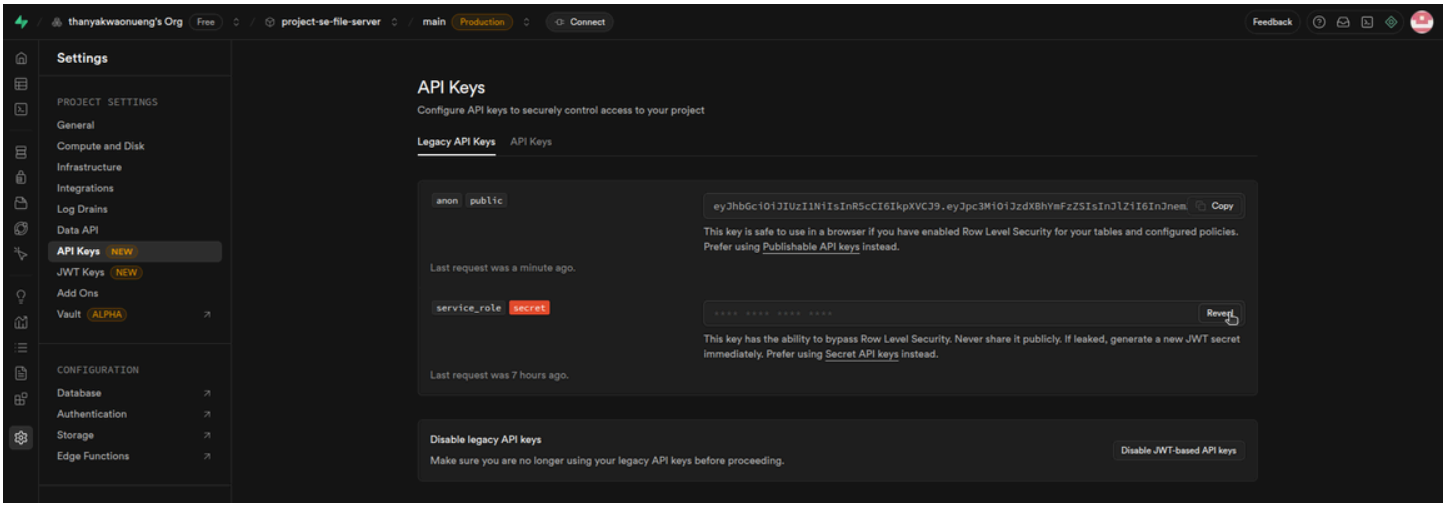
to left navbar navigate to project settings



then navigate to API keys section

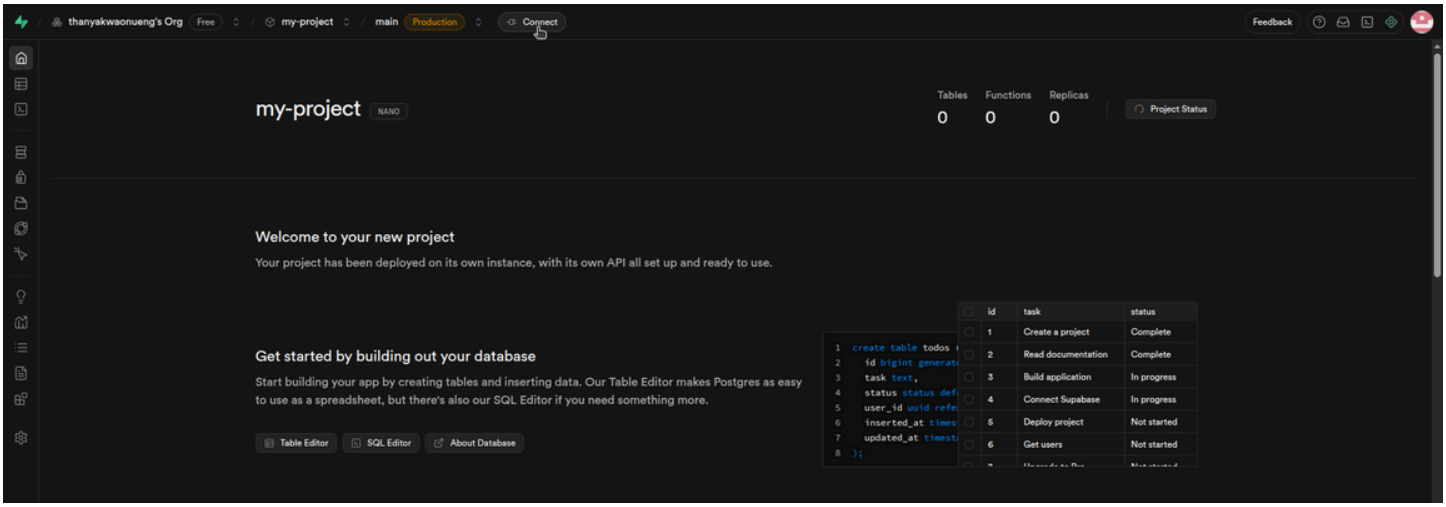


here at the service_role click reveal then copy the string, thereby we use that as SUPABASE_SERVICE_KEY

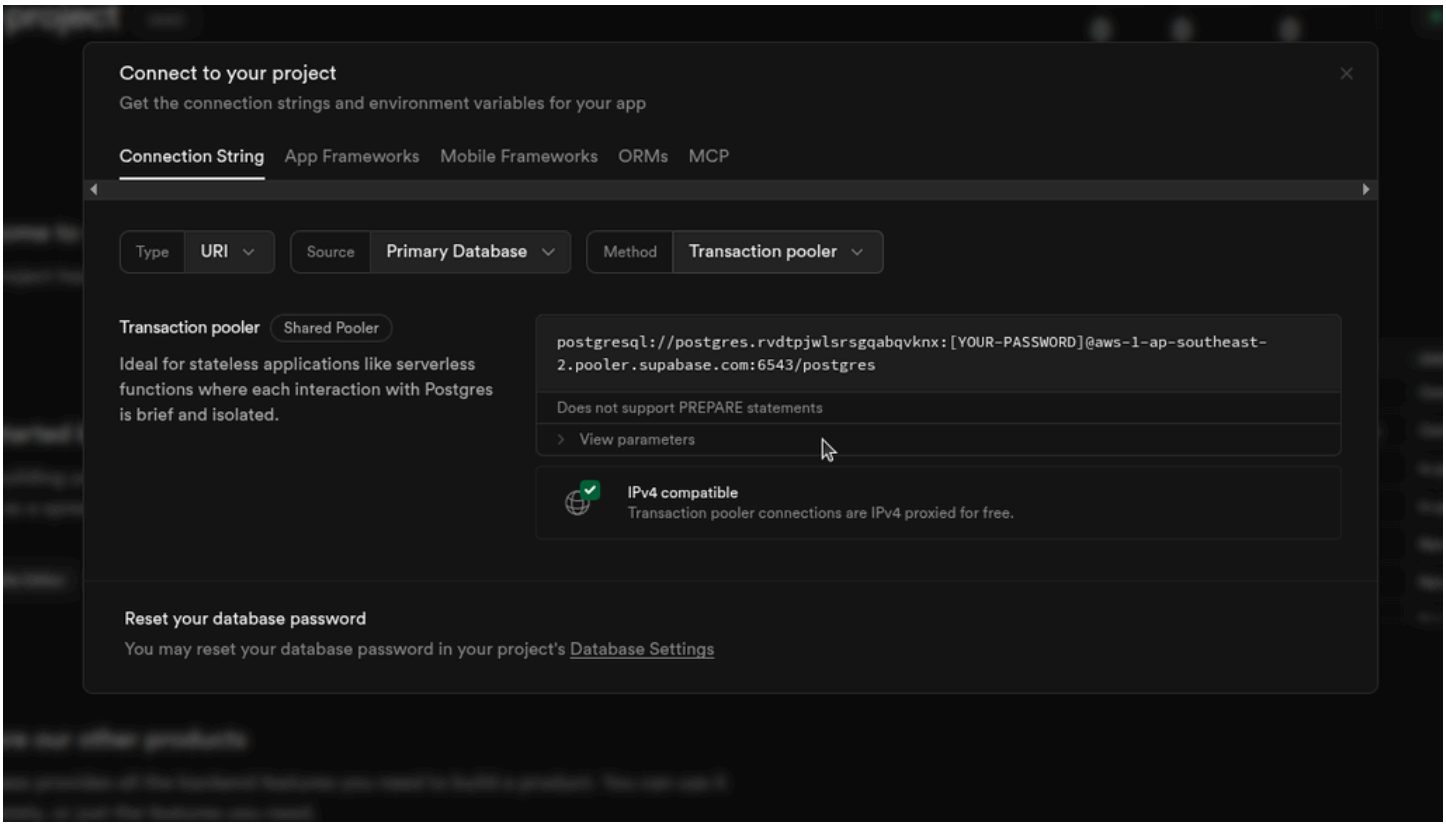


2.3.4.4 DATABASE_URL

on top navbar click connect



choose the method as transaction pooler, copy the string then replace [YOUR-PASSWORD] with your database password mentioned in 2.3.1, now this is your DATABASE_URL



2.3.4.5 DIRECT_URL

do the same with 2.3.4.4 by clicking connect but at the method choose session pooler, copy the string then replace [YOUR-PASSWORD] with your database password mentioned in 2.3.1, now this is your DIRECT_URL

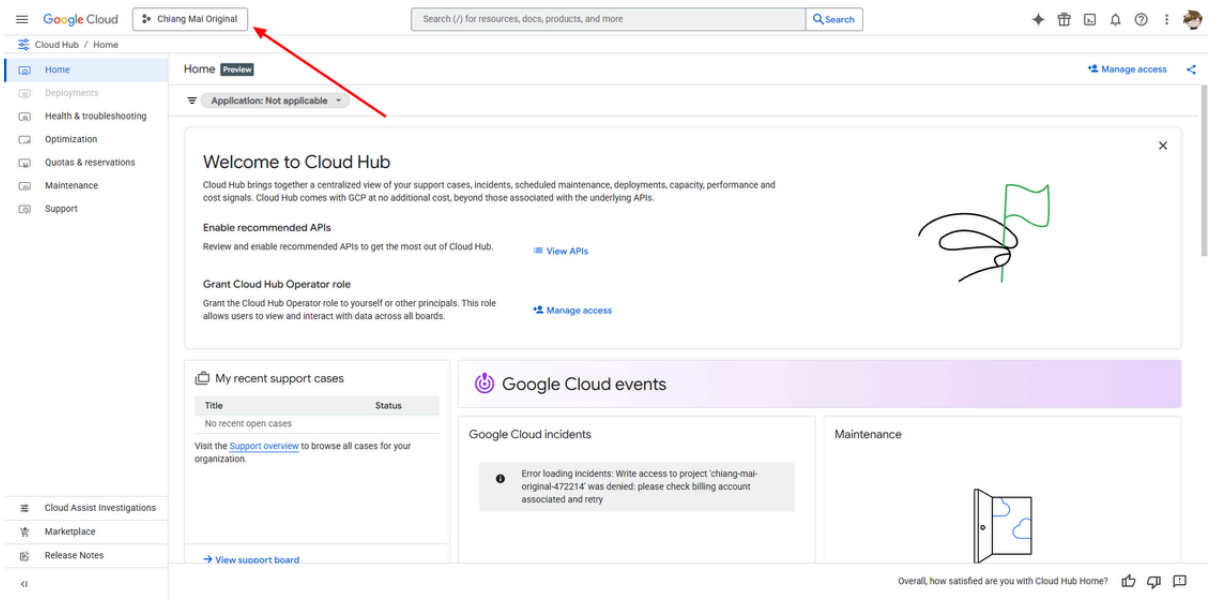
**note if you do not care about Oauth part yet and wish to see if the deployment setup will work please proceed to section 3.

2.3.4.6 GOOGLE_CLIENT_ID

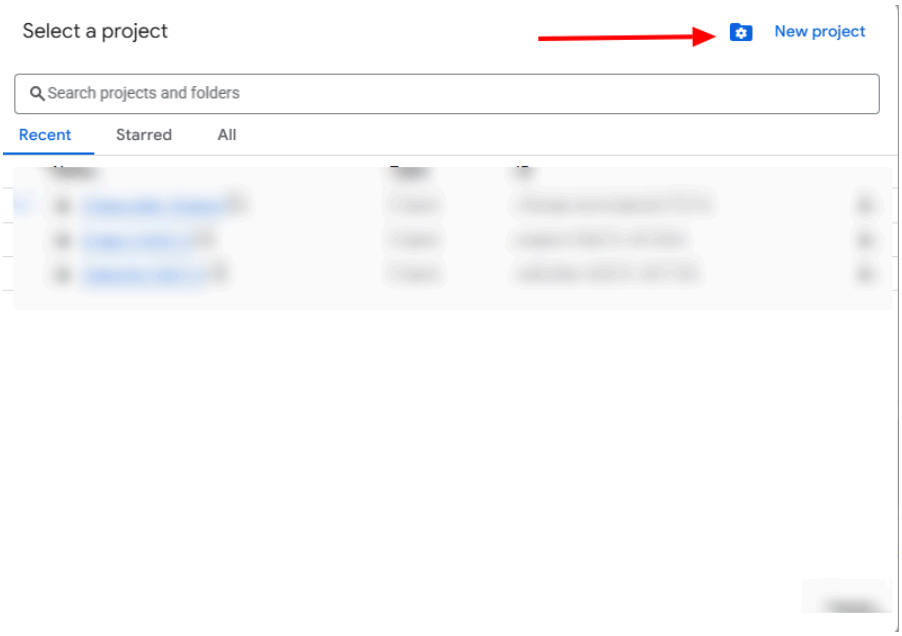
1. Go to Google Cloud Console and login with your Google account
    - Google Cloud Console link: https://console.cloud.google.com

2. Create a new project

2.1) Click the project selector menu (at the top-left).



2.2) Click **"New Project"**

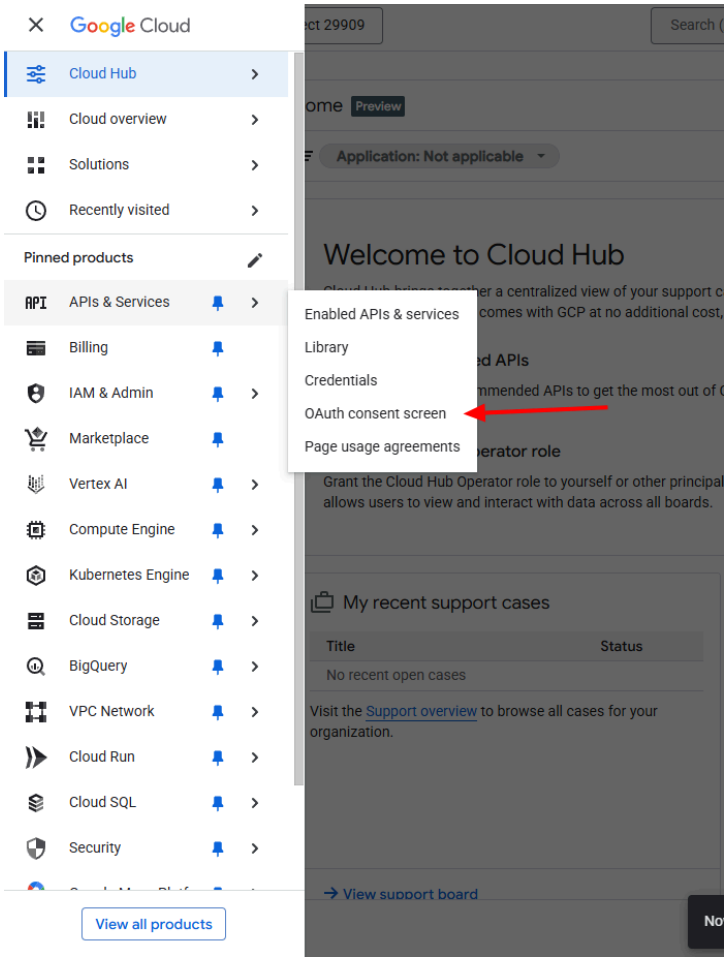


2.3) Give your project a name (e.g., "My Web App Auth") and click **"Create"**

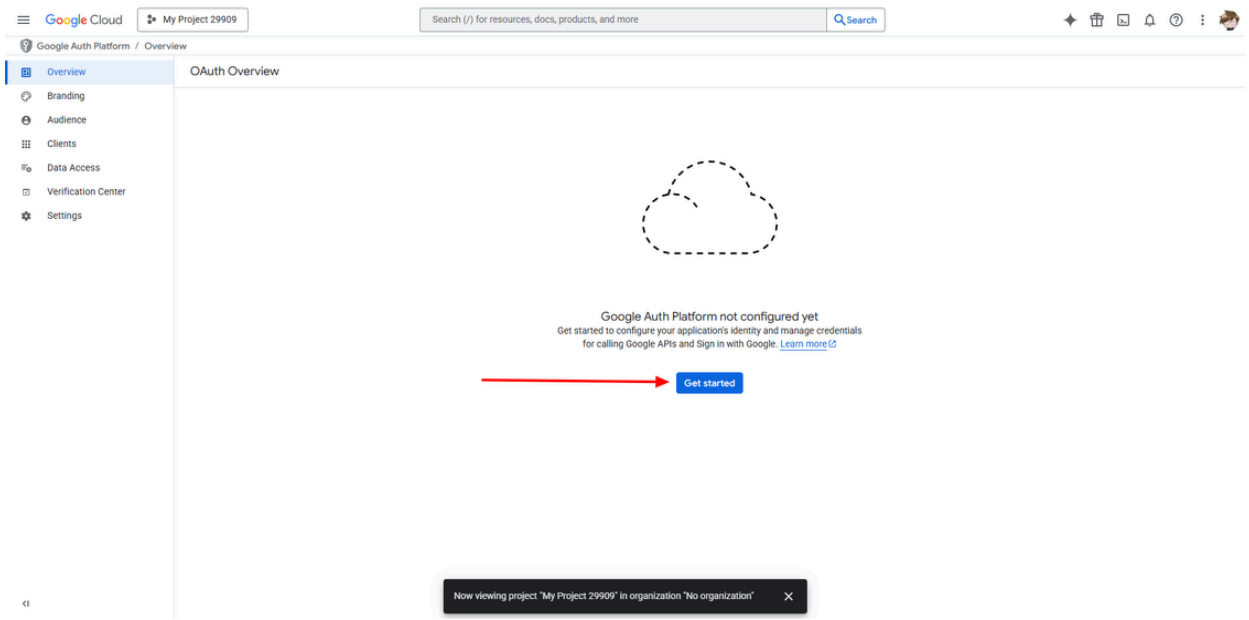
3. Configure the OAuth Consent Screen

Before you can create a Client ID, you must tell Google what your app is and what permissions it will request from users.

3.1) In the left-hand navigation menu, go to **APIs & Services** > **OAuth consent screen**.

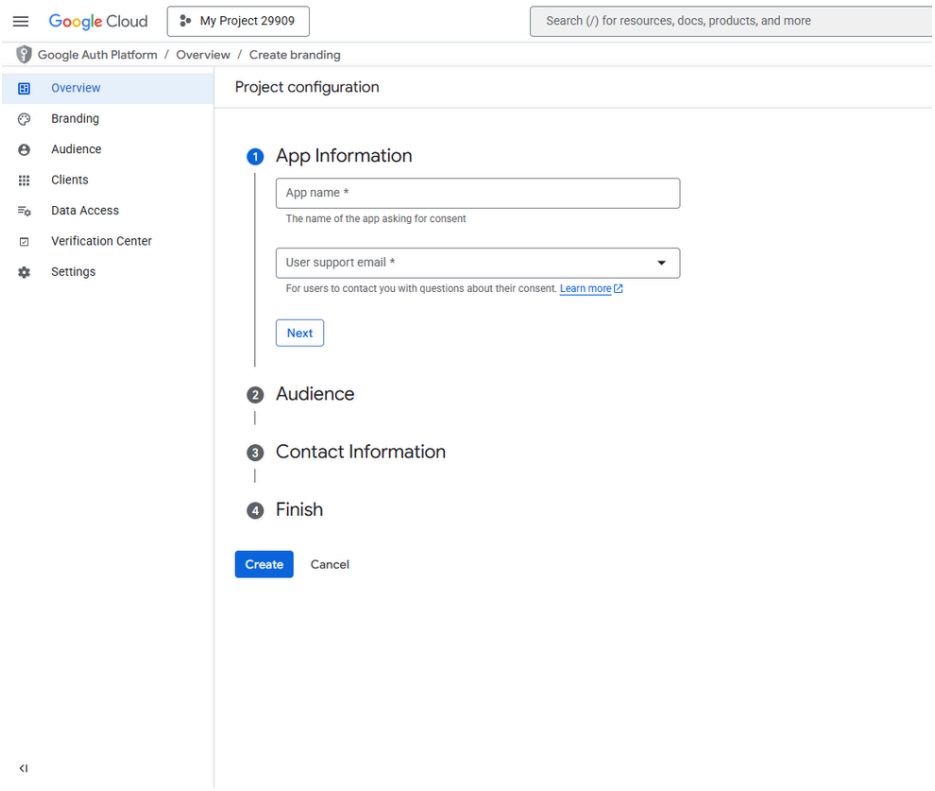


3.2) Click **“Get Started”** to start configuring the OAuth Consent screen.



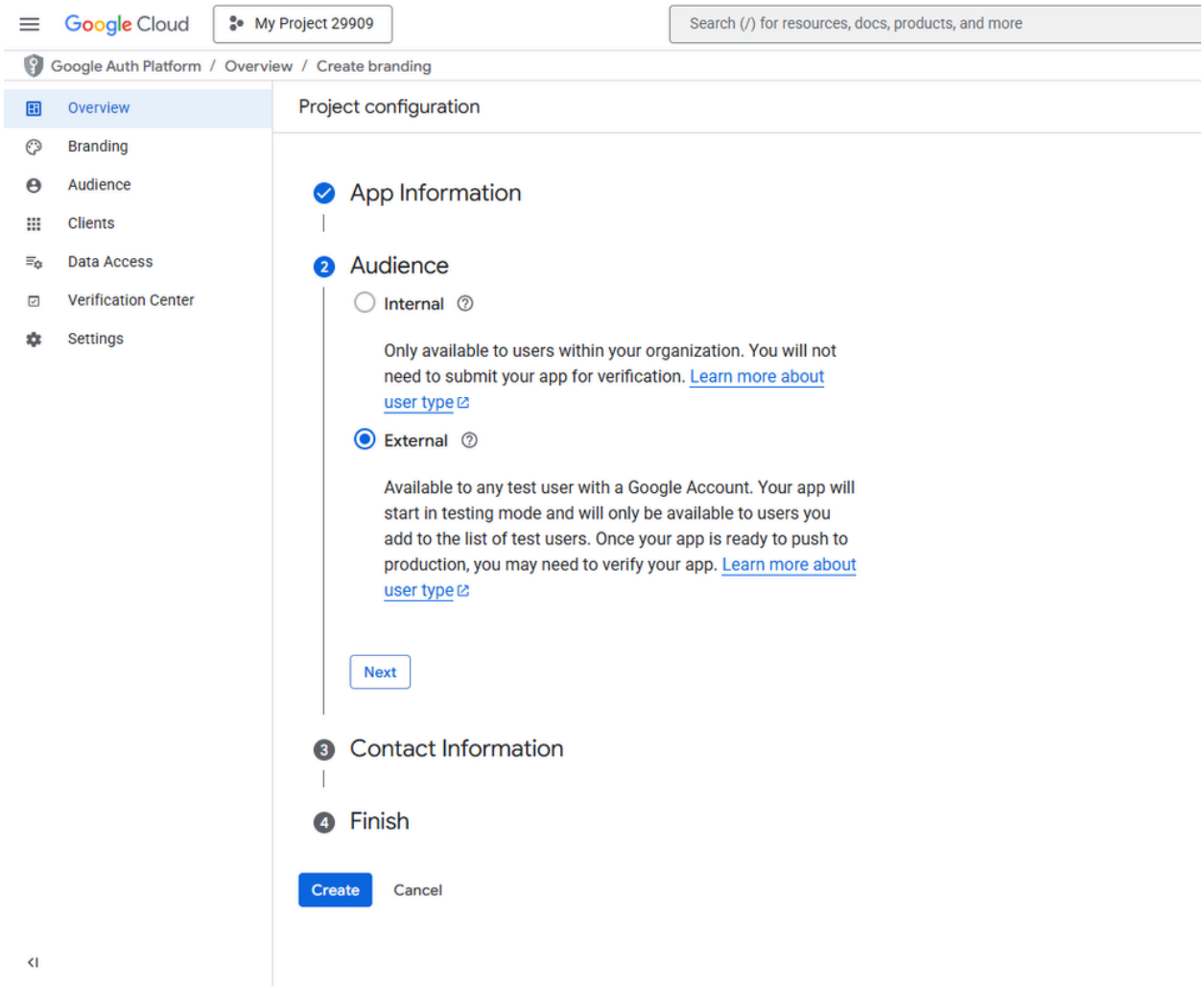
3.3) Fill in the required information:

- **App name:** The name of your app that users will see.
- **User support email:** Your email address for users to contact.

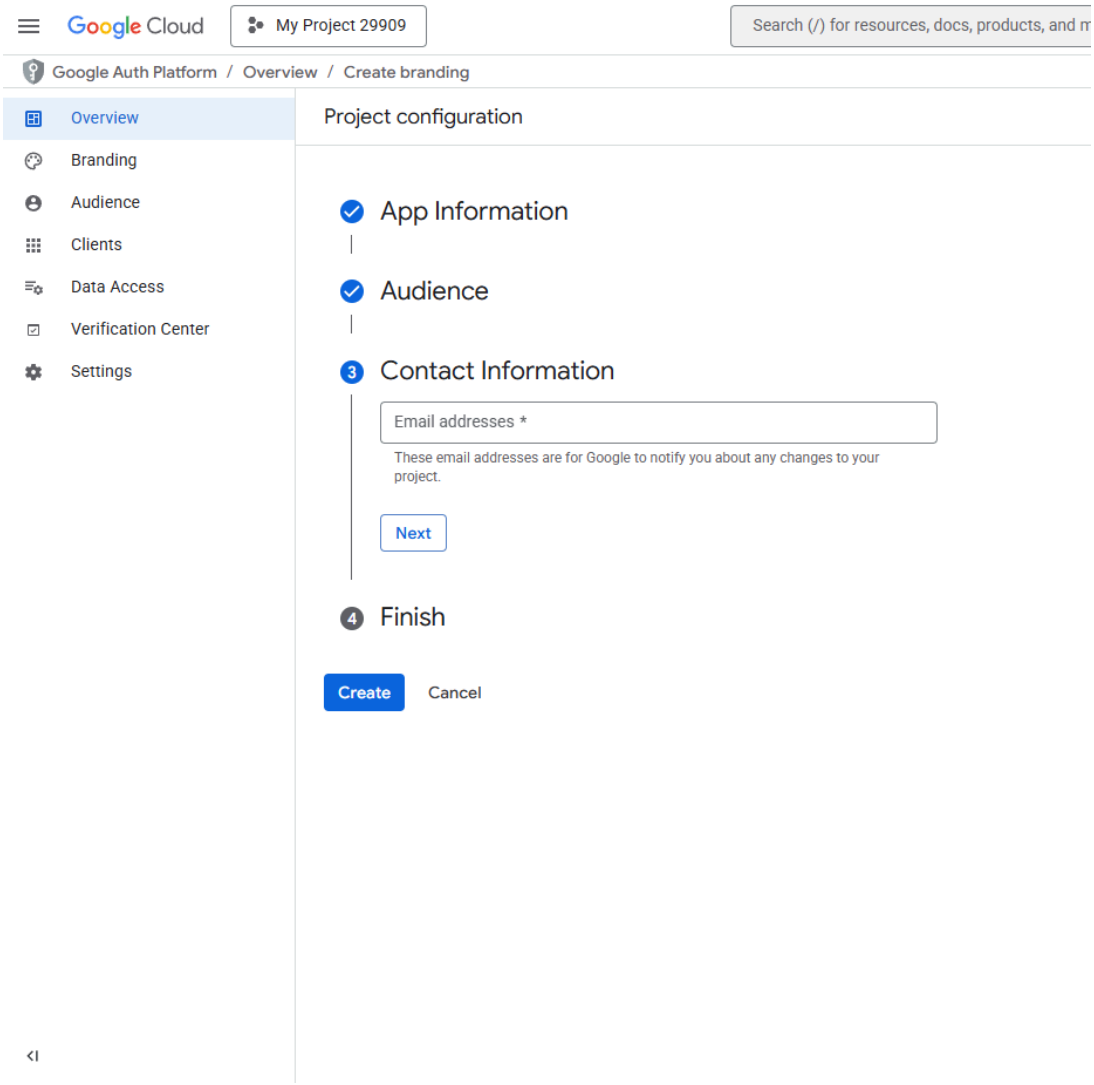


3.4) Choose the **User Type**: **(Choose External)**

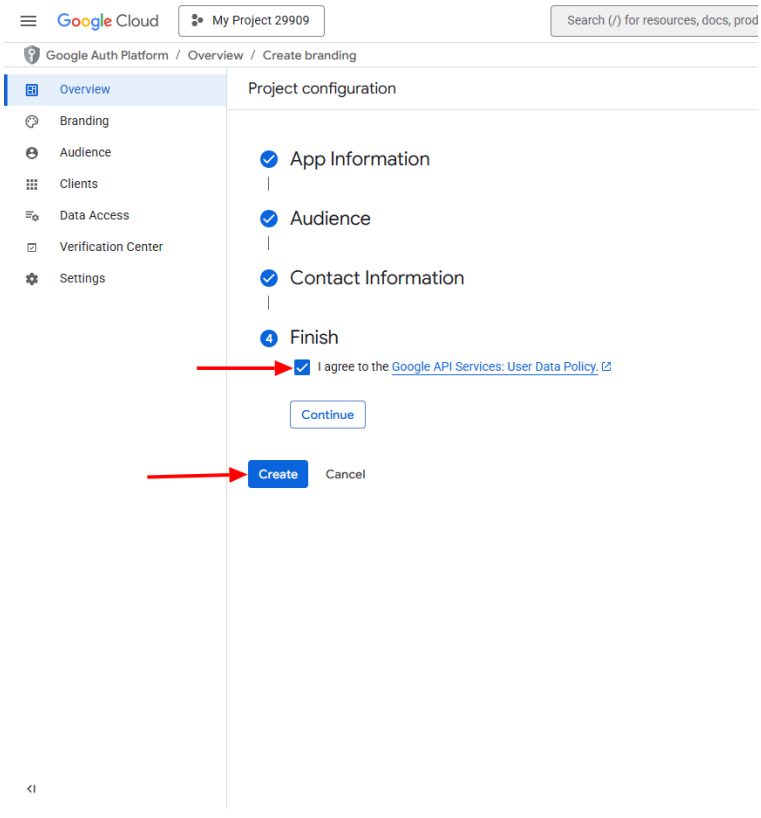
- **Internal:** Only for users within your Google Workspace organization (e.g., company employees).
- **External:** For any user with a Google account (this is the most common choice for a public web app).



3.5) Fill in contact information with your email.

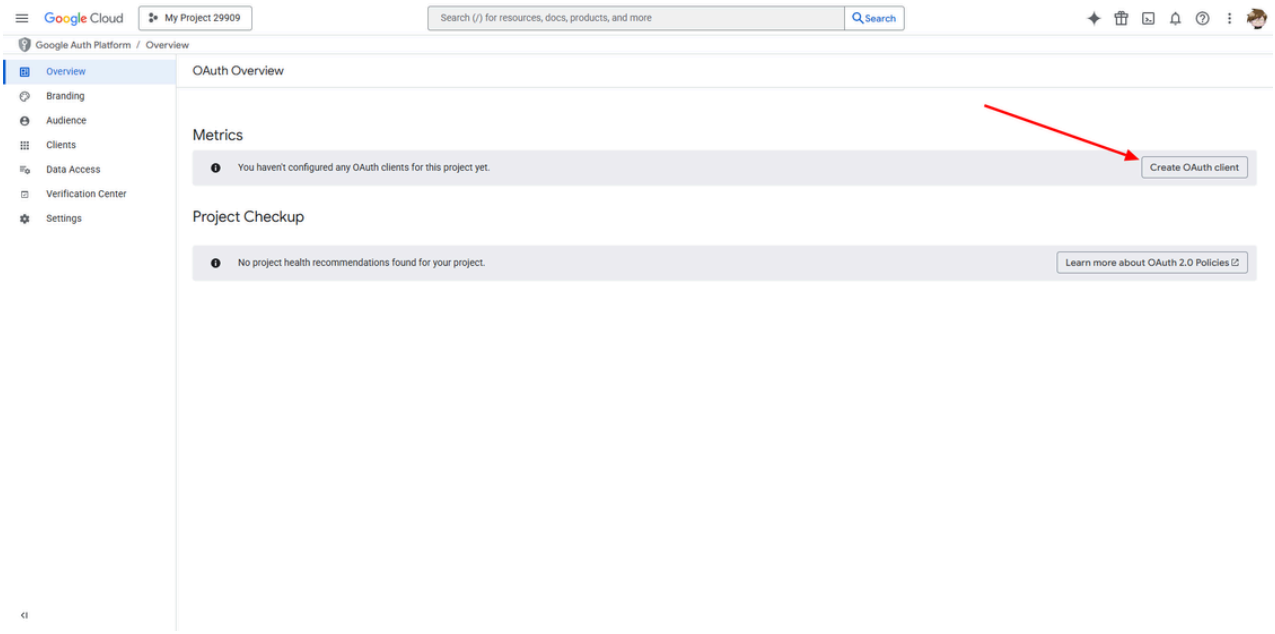


3.6) Click **“I agree to the Google API Services: User Data Policy”** and **“Create”.**

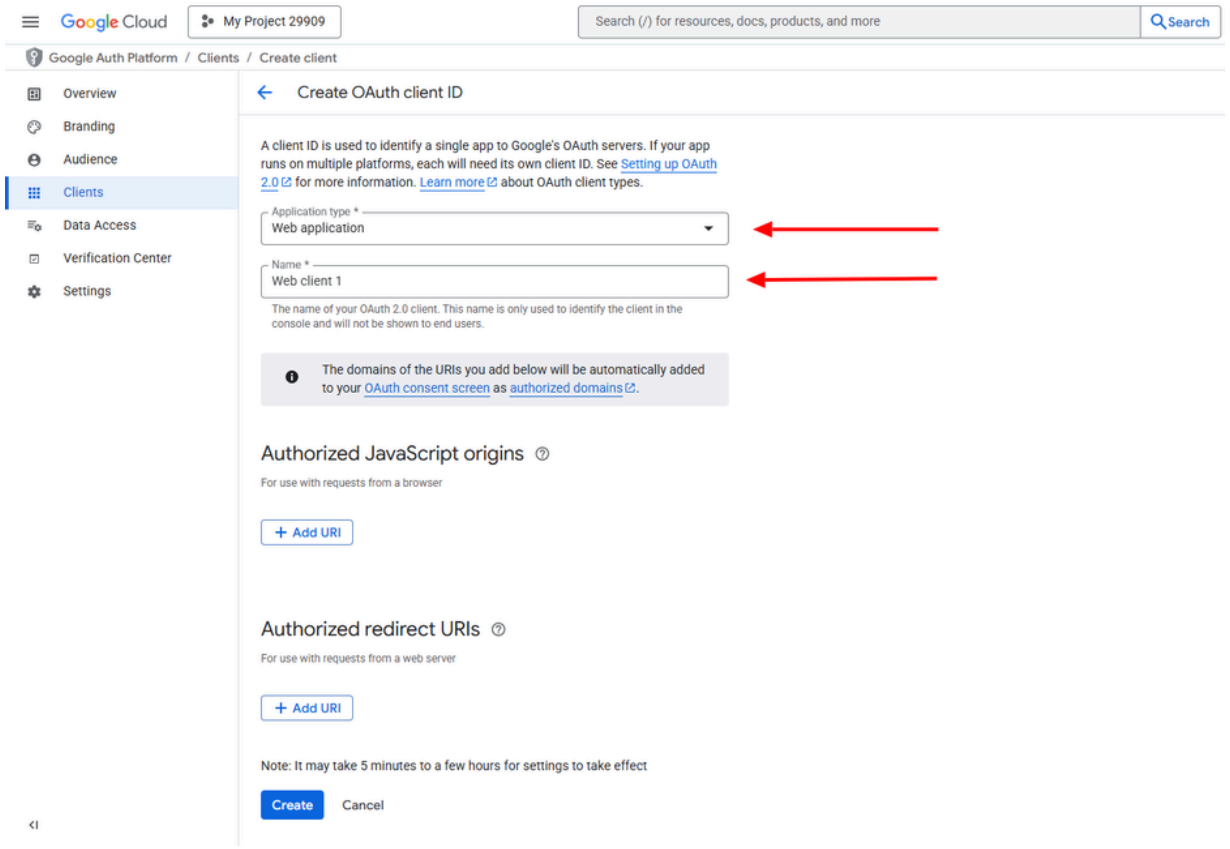


4. Create Your Credentials

4.1) Click **“Create OAuth Client”**

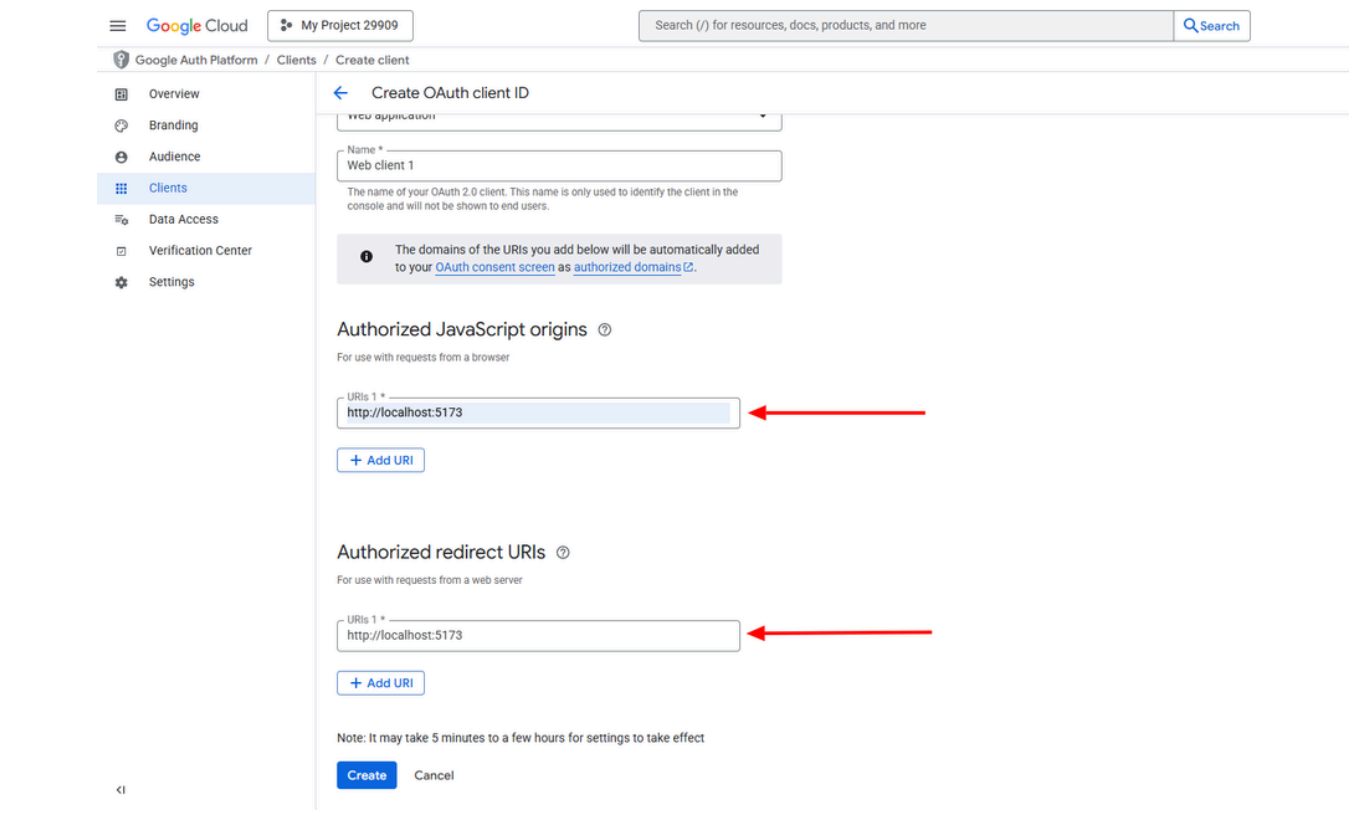


4.2) On the next page, set the **Application type** to **Web application** and give this Client ID a **Name** (e.g., "Web App Client").



4.3) Set the URIs **(Very Important)**:

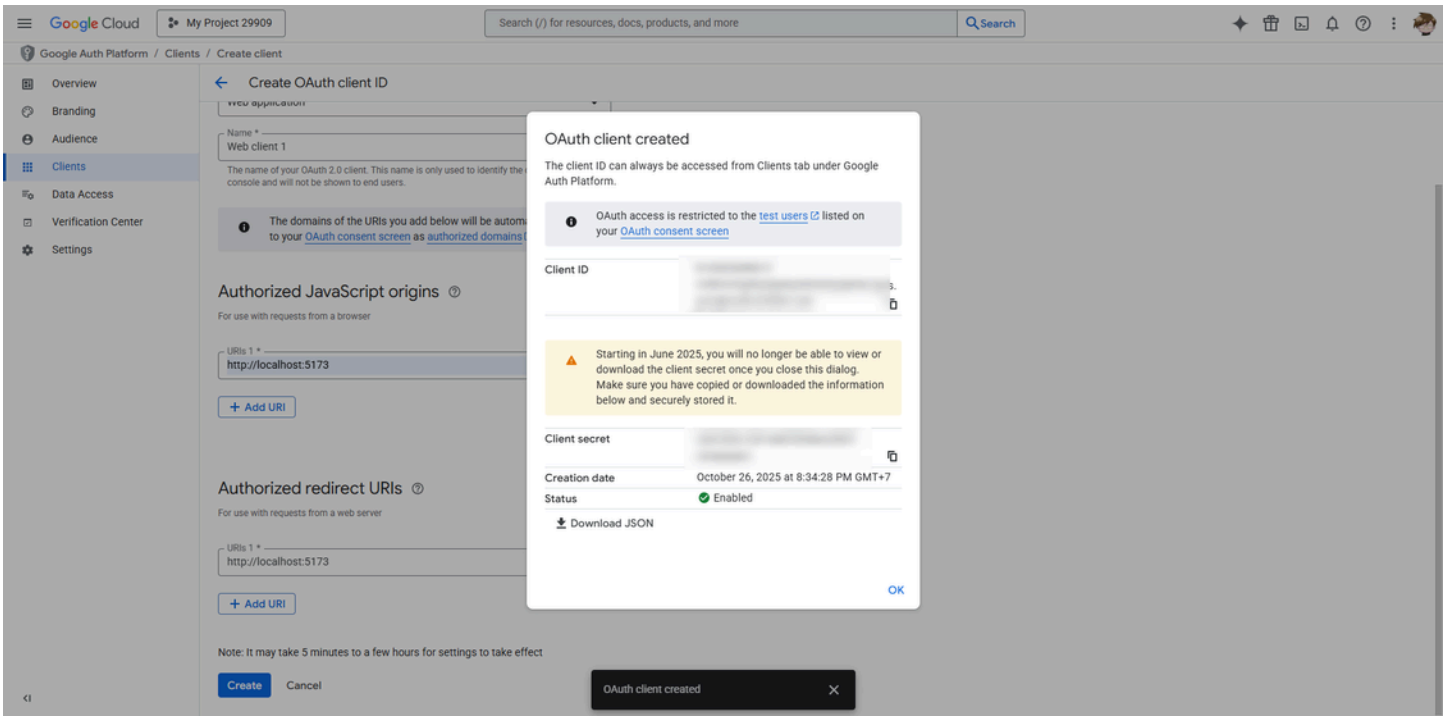
- **Authorized JavaScript origins:** The URL origins where your web app is hosted. *Do not include paths.* (e.g., https://www.your-domain.com or http://localhost:3000 for testing).
- **Authorized redirect URIs:** The specific URL endpoint where Google will send the user back to (with an authorization code) after they log in. *This must include the full path or use the same URL with* **Authorized JavaScript origins**. (e.g., https://www.yoyouromain.com/auth/google/callback or http://localhost:3000/callback).



4.4) Click **CREATE**

5. Copy your Client ID and Client Secret or download with JSON then keep them somewhere safely.

- GOOGLE_CLIENT_ID: **Client ID**
- GOOGLE_CLIENT_SECRET: **Client secret**



2.3.4.7 GOOGLE_CLIENT_SECRET

- Do the same with 2.3.4.6 by copying the Client secret.

2.3.4.8 EMAIL_USER

- Email you want to use for sending OTP to the user **(Must be Gmail only!)**.

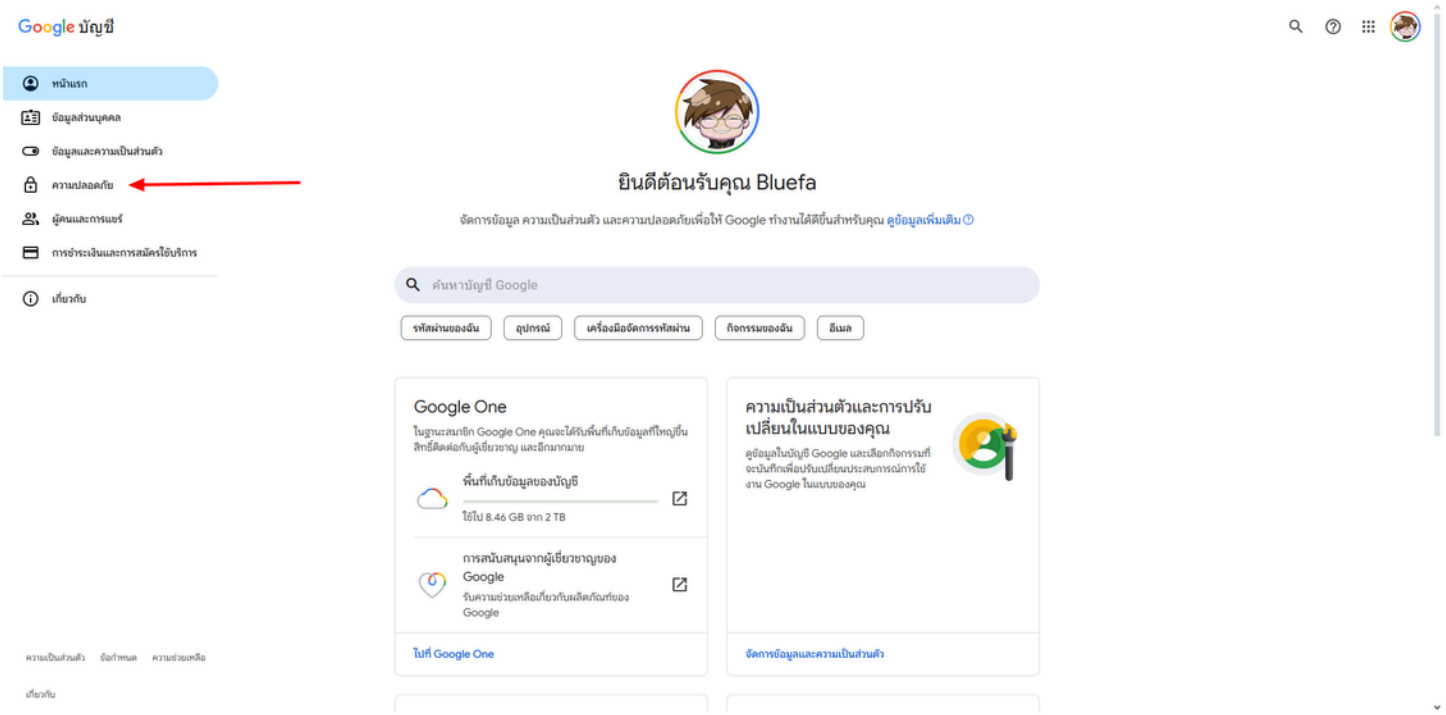
2.3.4.9 EMAIL_PASS (a.k.a. App password)

1. Enable 2-Step Verification **(If not already on)**

   If you haven't enabled 2FA, you must do this first:

   1.1) Go to your Google Account settings: https://myaccount.google.com

   1.2) On the left, click the **"Security"** tab.

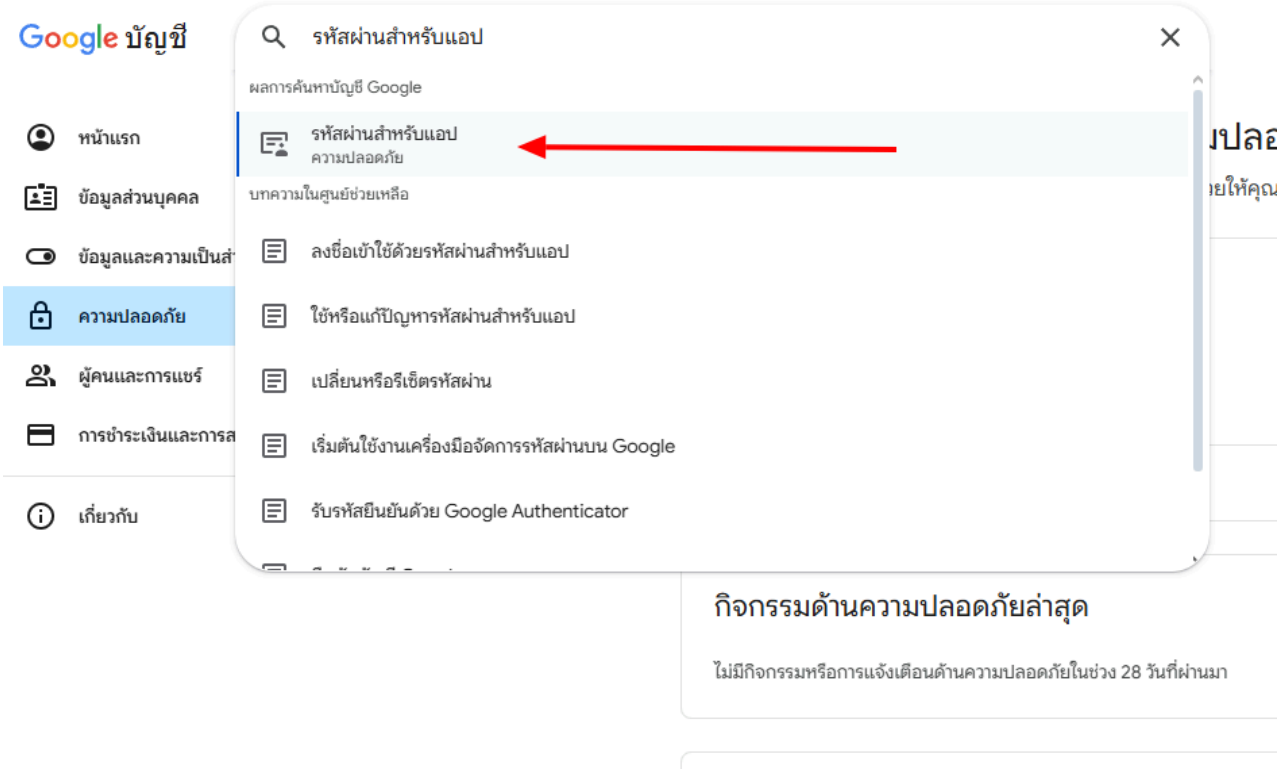


1.3) Under the "How you sign in to Google" section, click on **"2-Step Verification"**

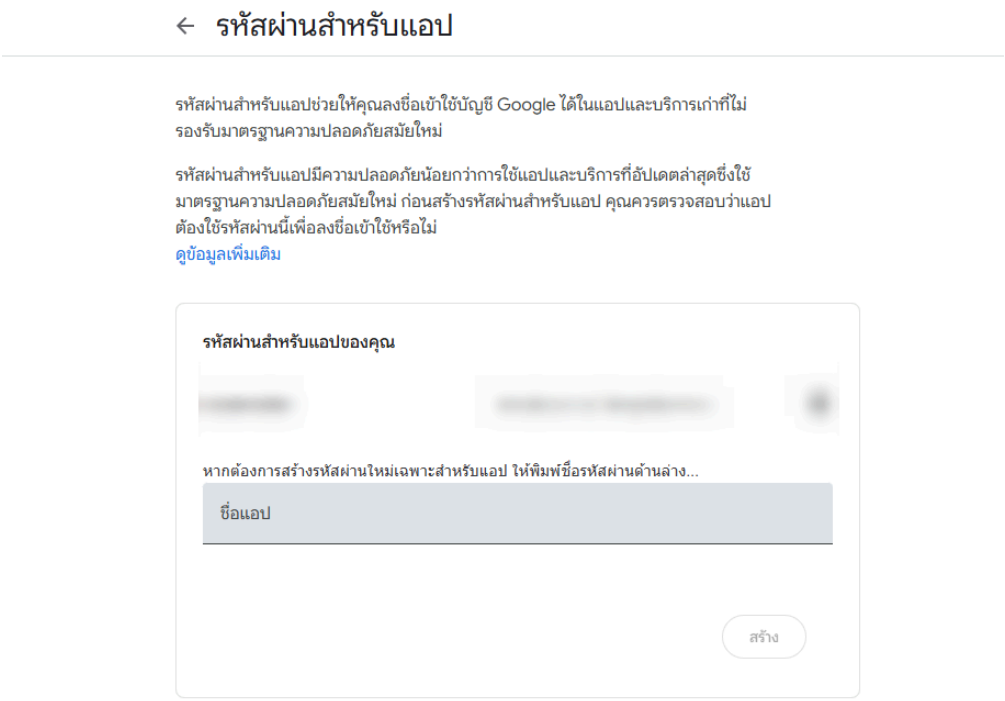
1.4) Click **"Get Started"** and follow the on-screen steps to verify your phone number.

2. Generate the App Password

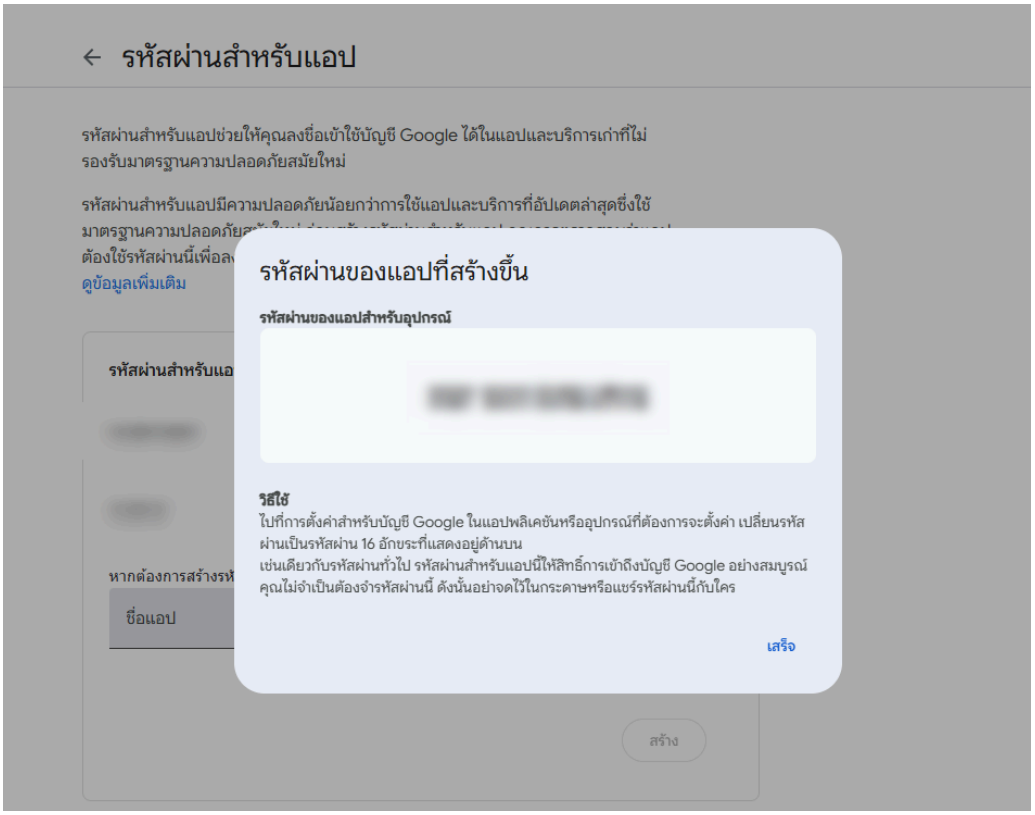
   2.1) Search **“รหัสผ่านสำหรับแอป”** (or **“App Passwords”** in English) and click it**.**



2.2)Fill your app name



2.3) Save your password somewhere safe. Your password should be this format **“xxxx xxxx xxxx xxxx”**



2.4) Save variable as a string (EMAIL_PASS= **“xxxx xxxx xxxx xxxx”**)

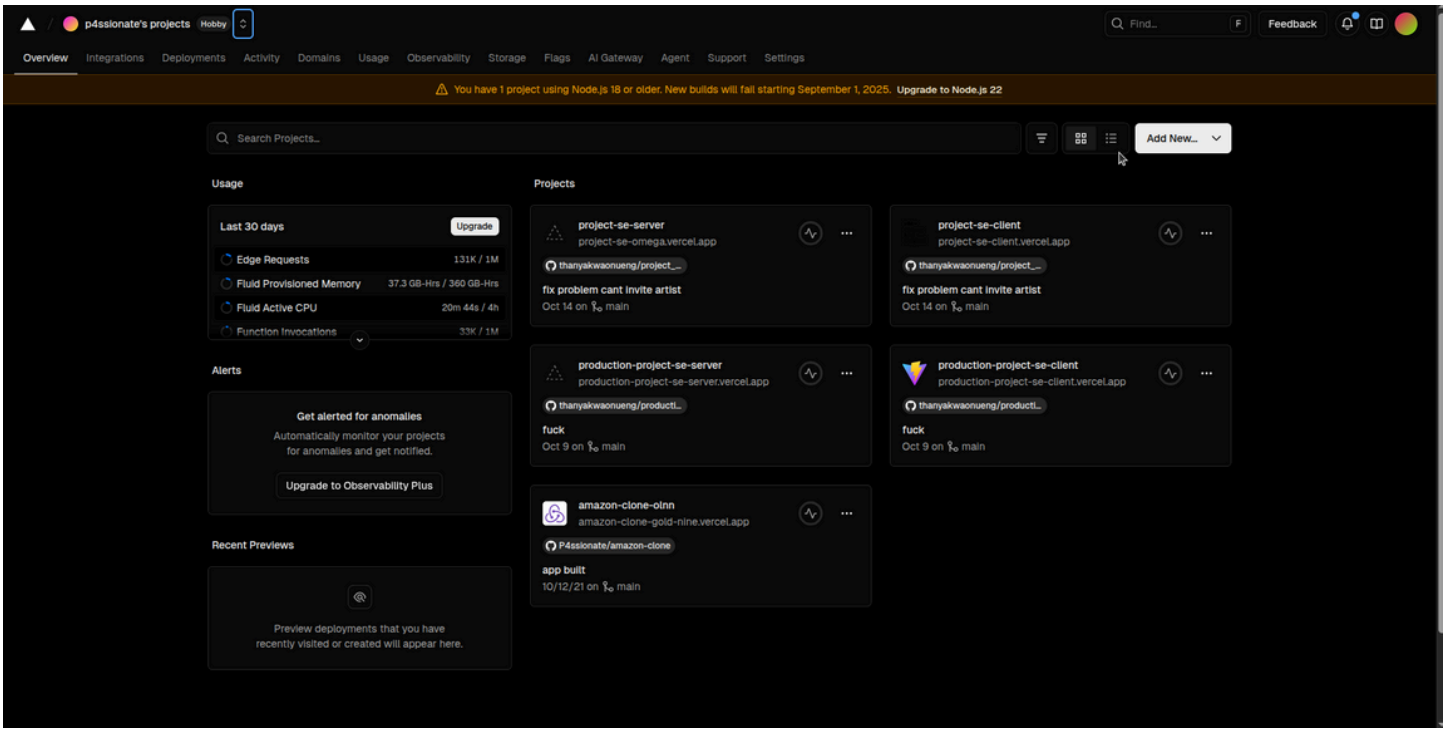
2.3.4.10 VITE_GOOGLE_CLIENT_ID

- Same value as GOOGLE_CLIENT_ID but used for the frontend library.

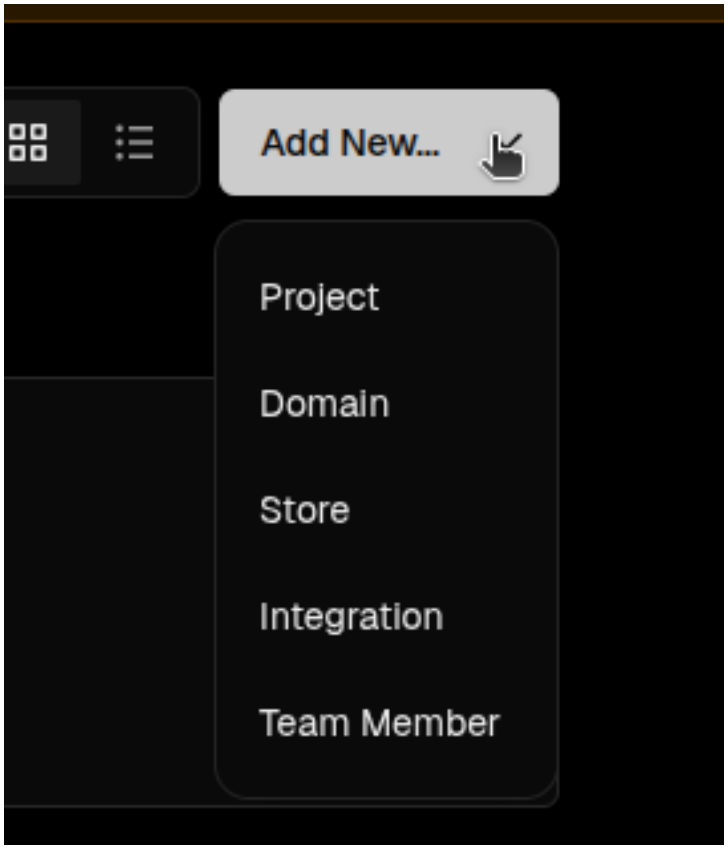
3.deploy frontend and backend service to vercel

3.1 firstly, fork https://github.com/thanyakwaonueng/project_se as your github repository

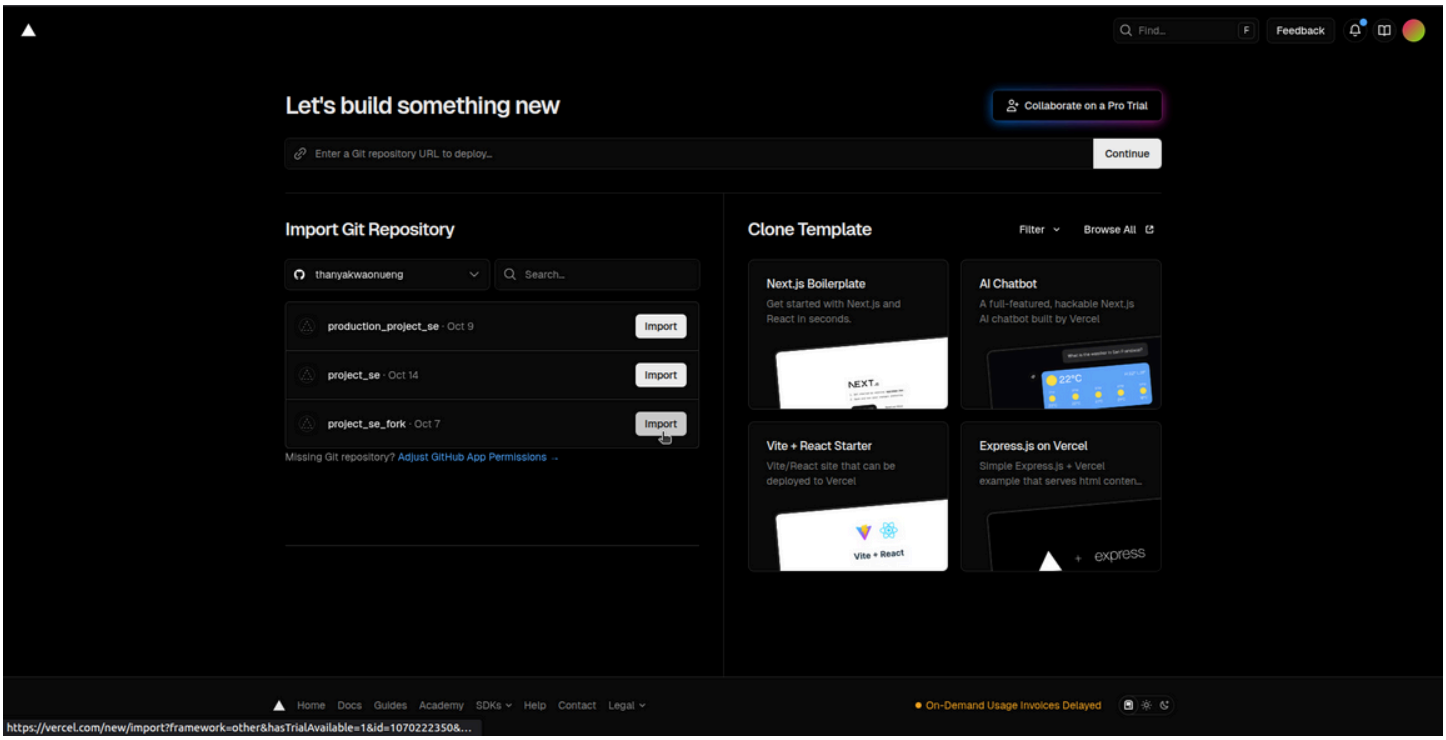
3.2 signup/signin vercel then navigate to Overview from top navbar



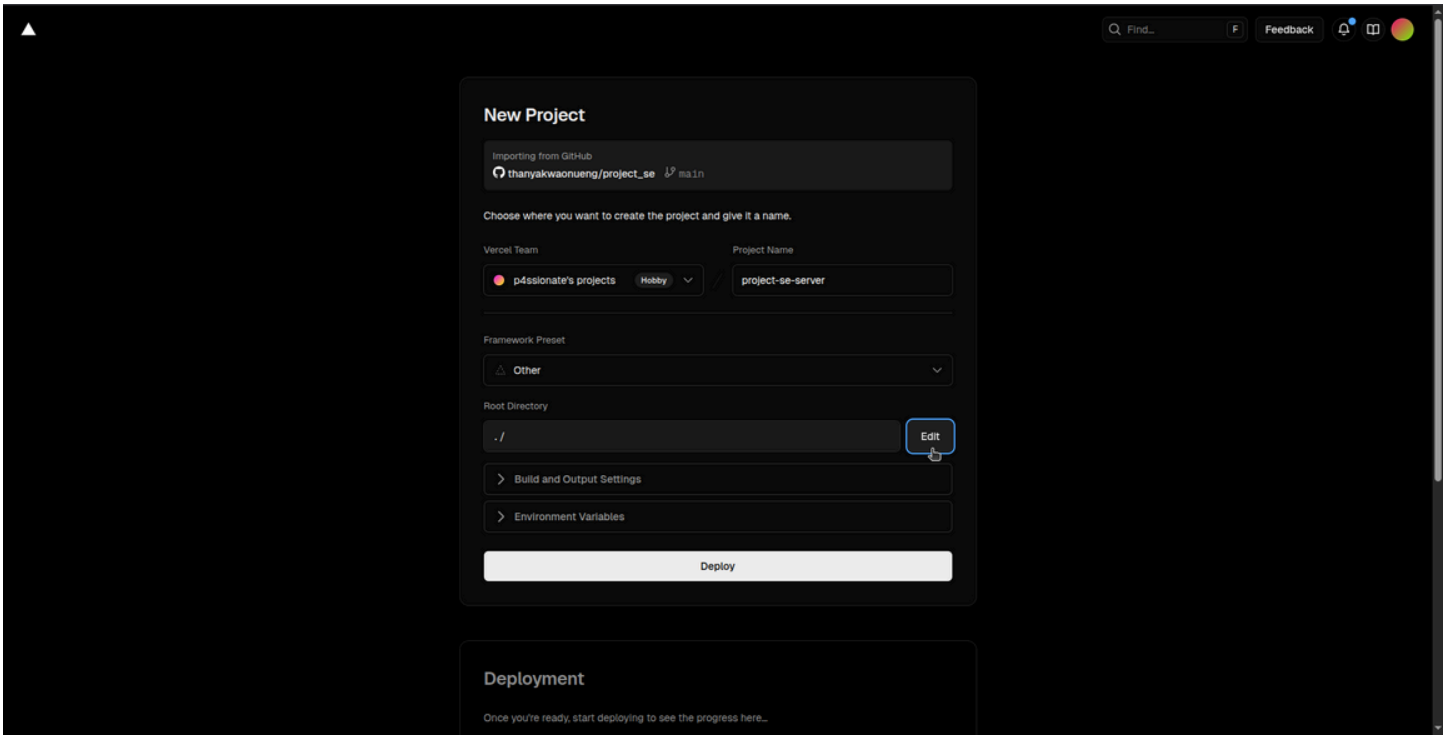
3.3 click Add new → project



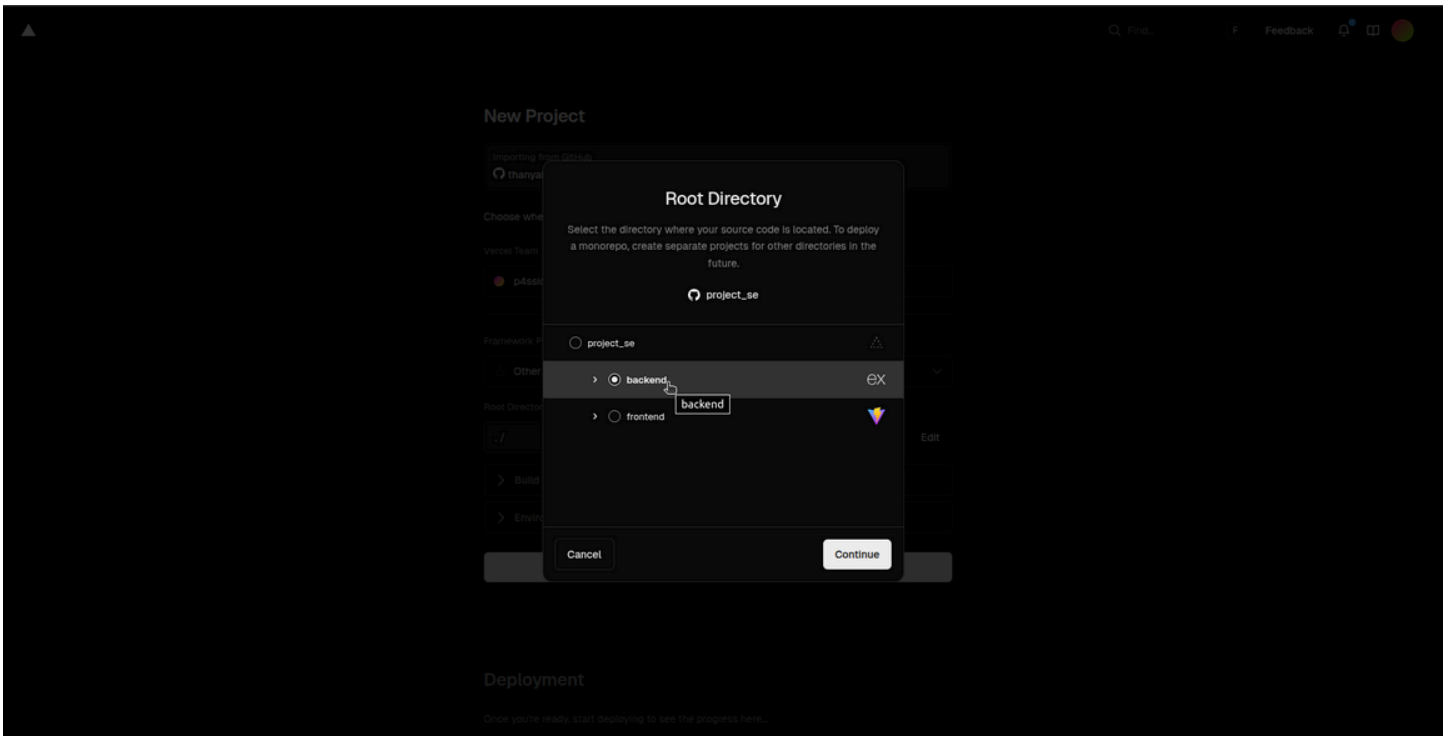
3.4 import the forked repository



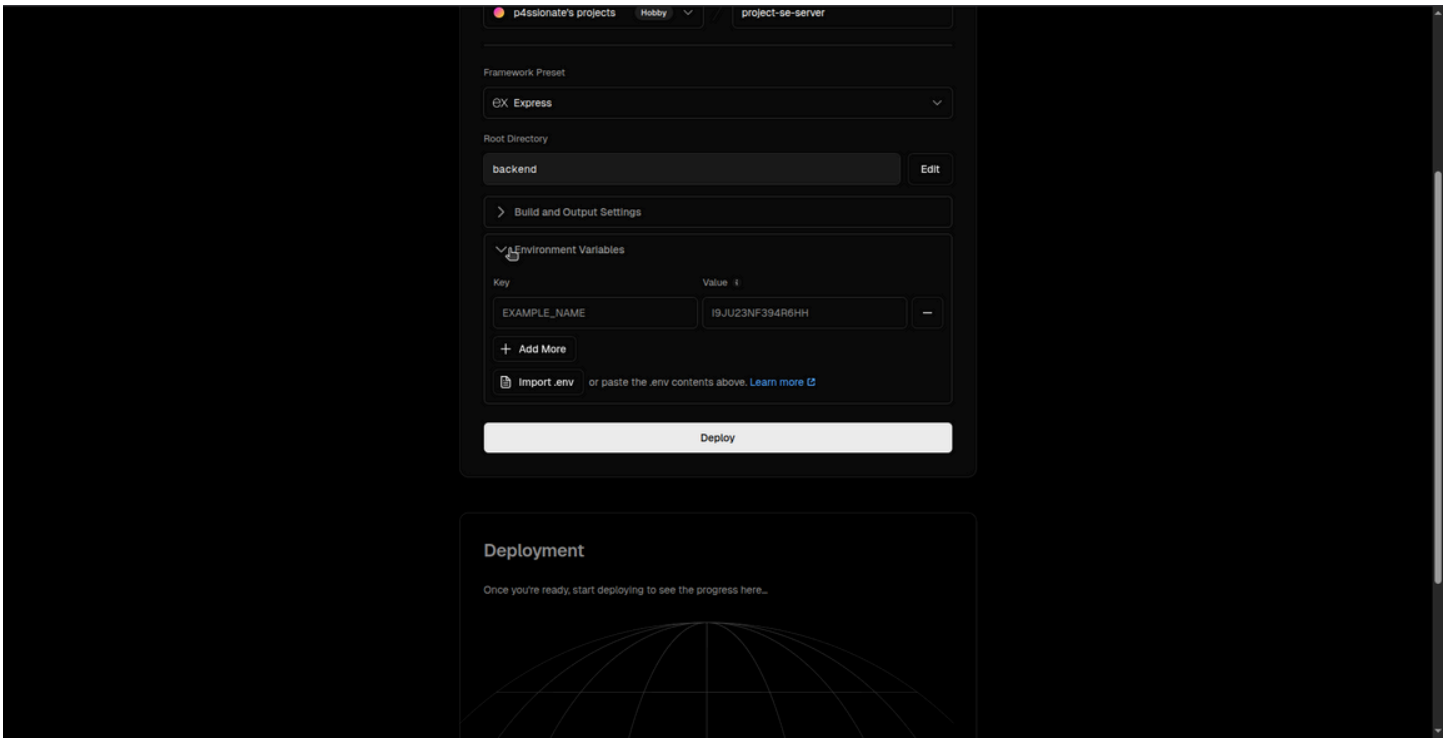
3.5 you'll be in this page name your project as something for example my-project-server, then click Edit



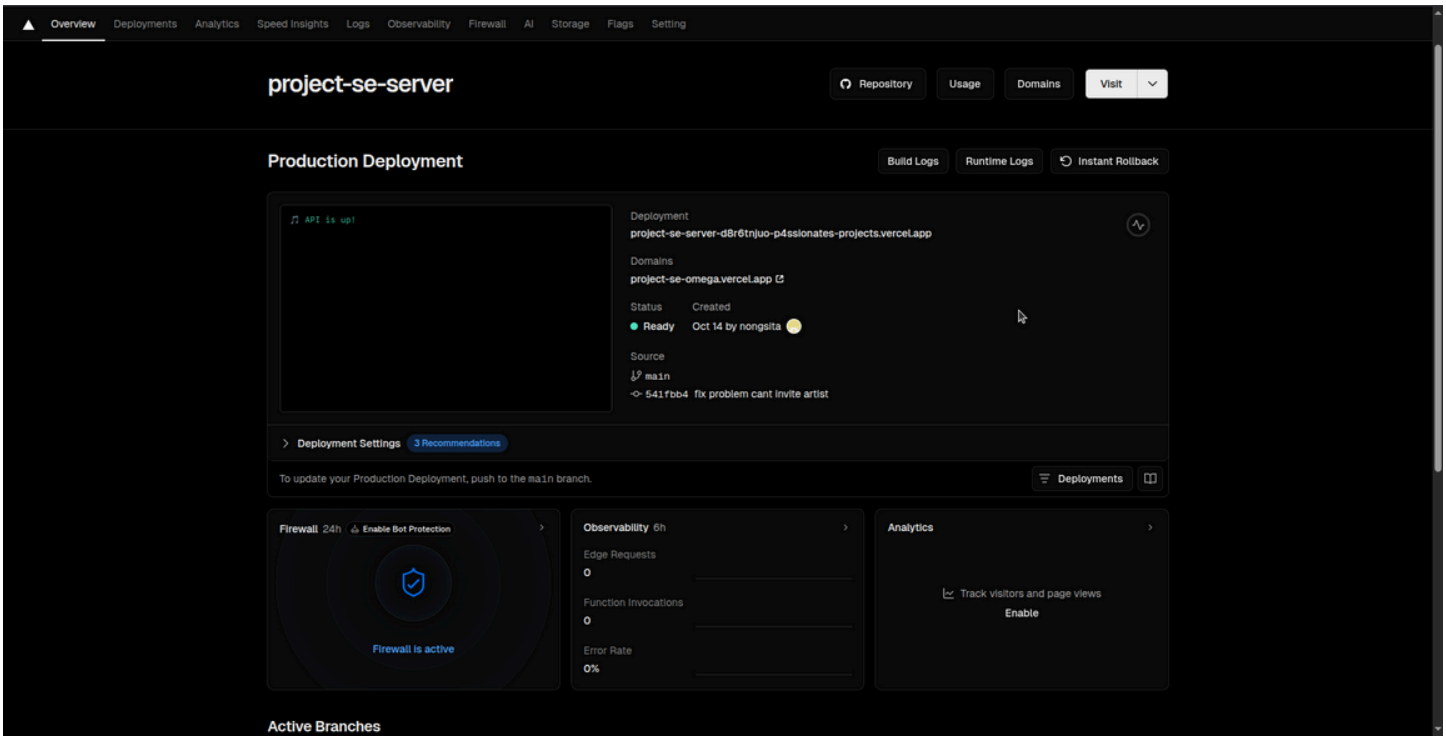
3.6 choose backend, then click continue



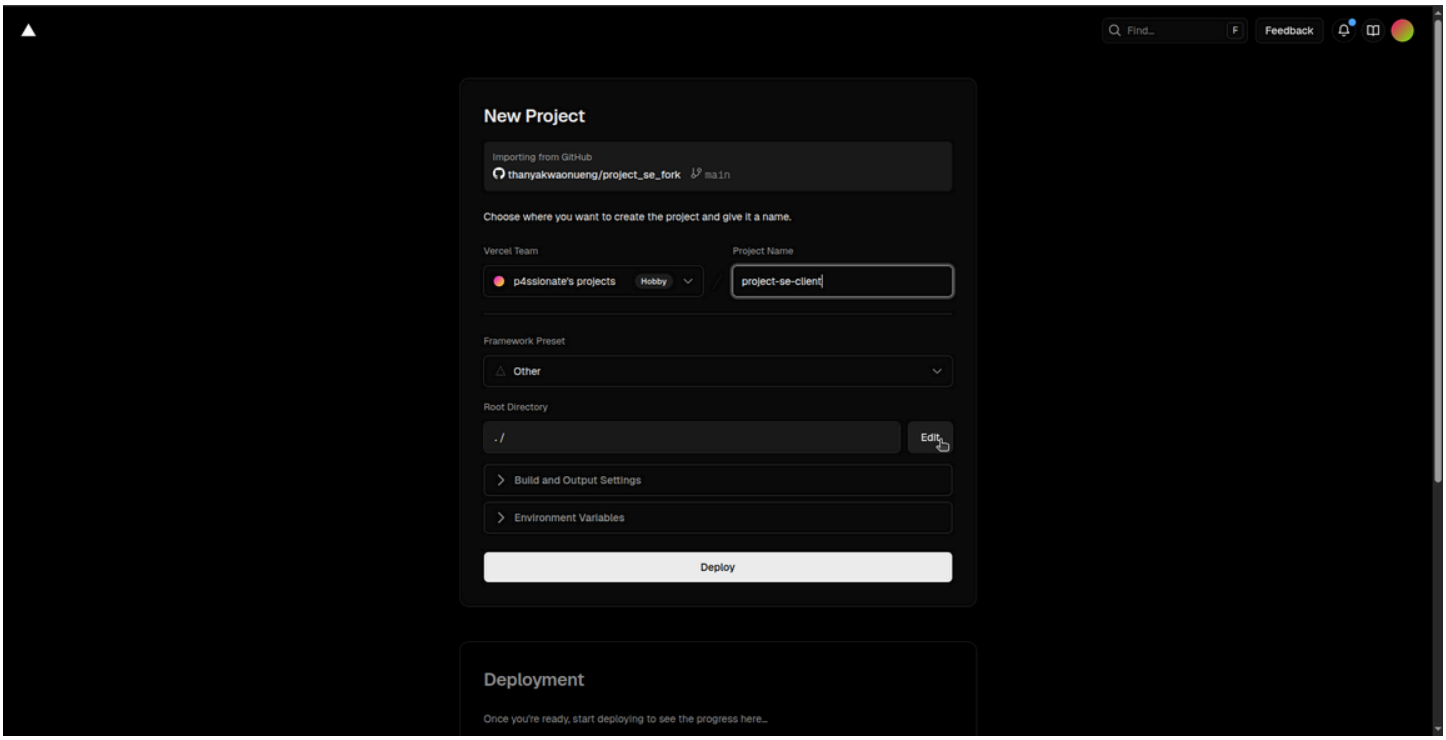
3.7 select environment variable section then add all env credential in the step 2.3.4, then click deploy



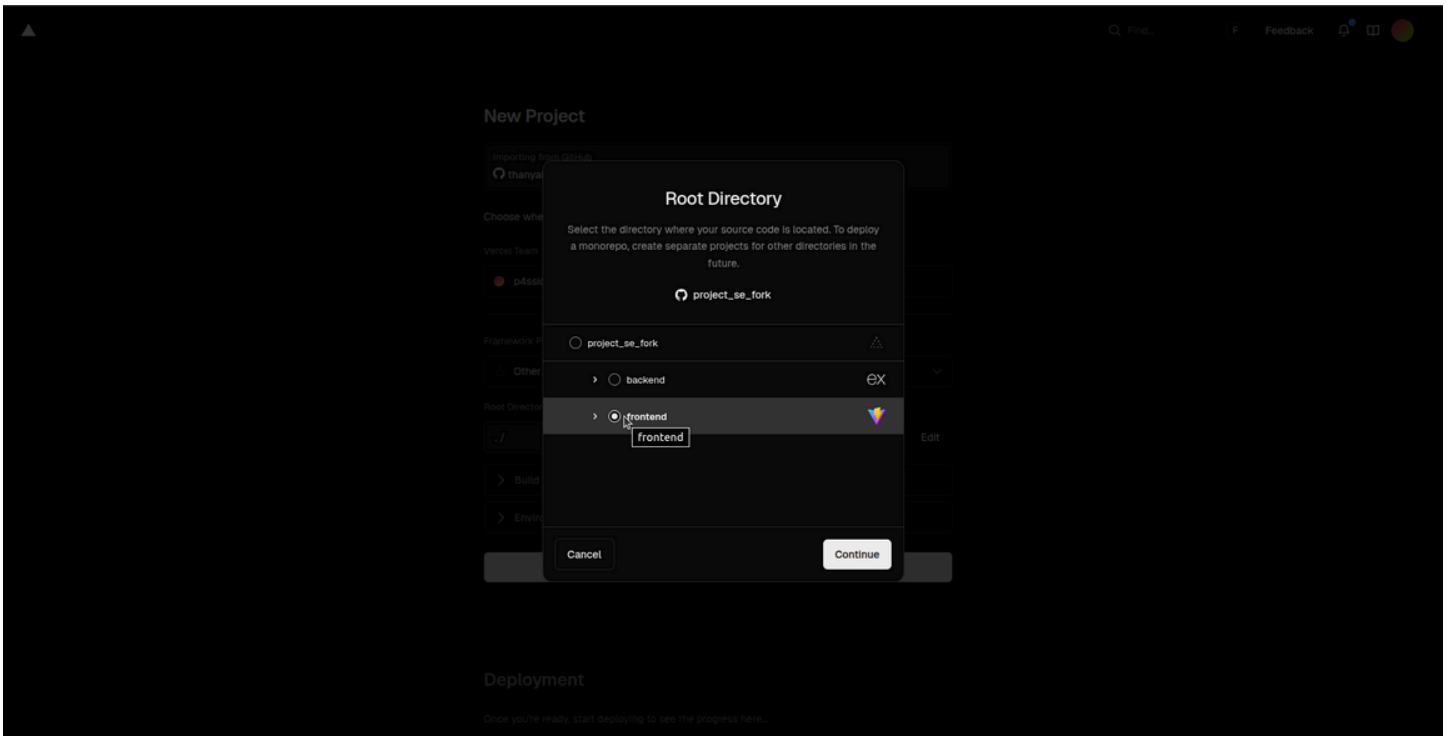
3.8 if built correctly it should be similar to this



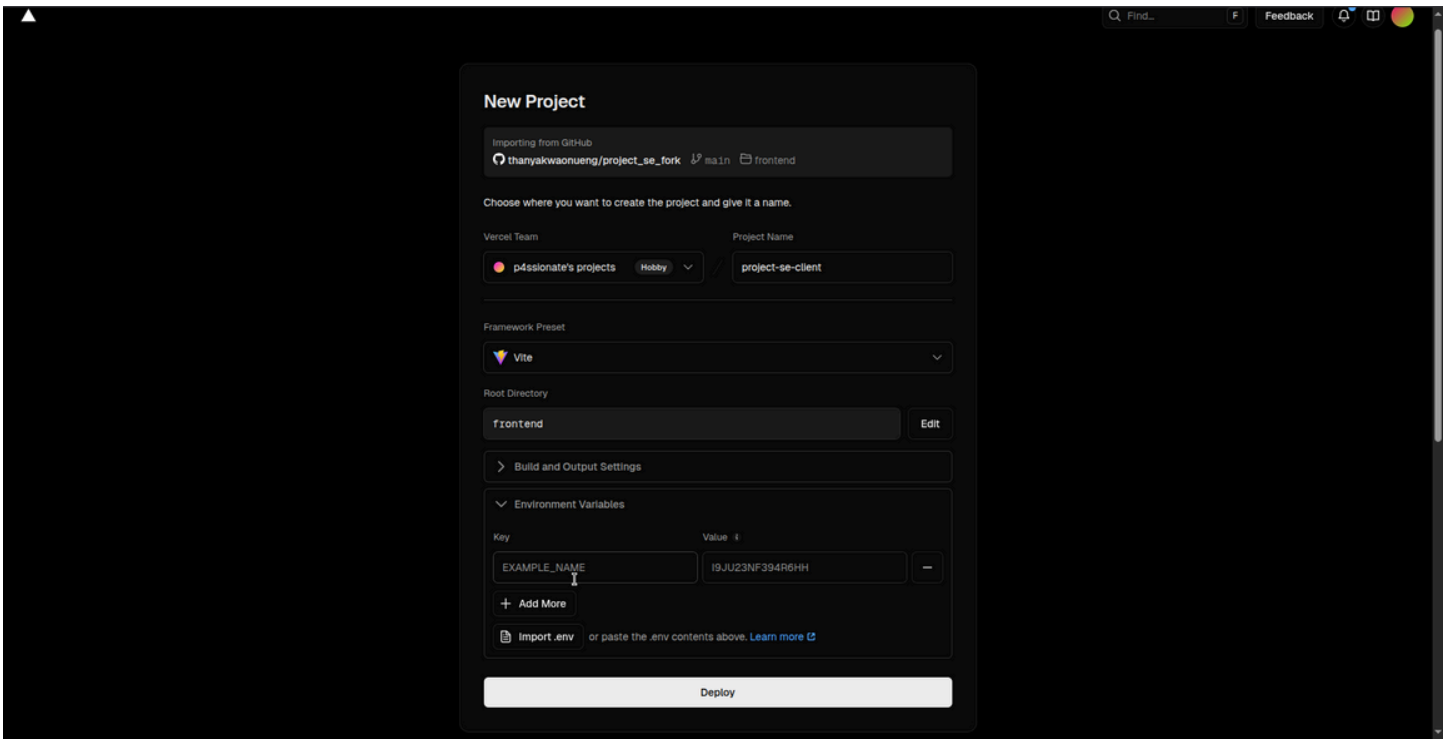
3.9 next to deploy frontend service we do step 3.2 to 3.4 again, i.e. Add new → project, import repo etc., then you will be back to this familar page again name your project something like my-project-client then click Edit



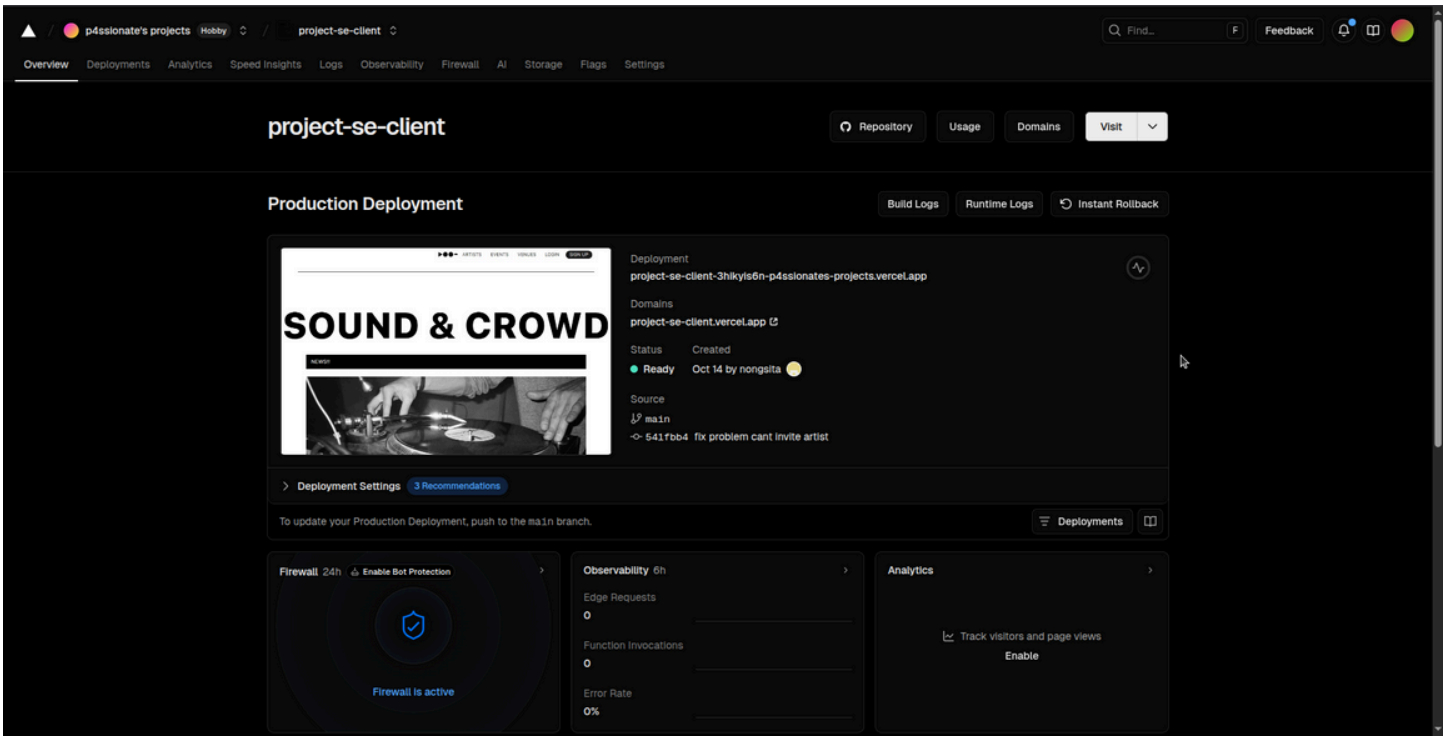
3.10 select frontend, then click continue

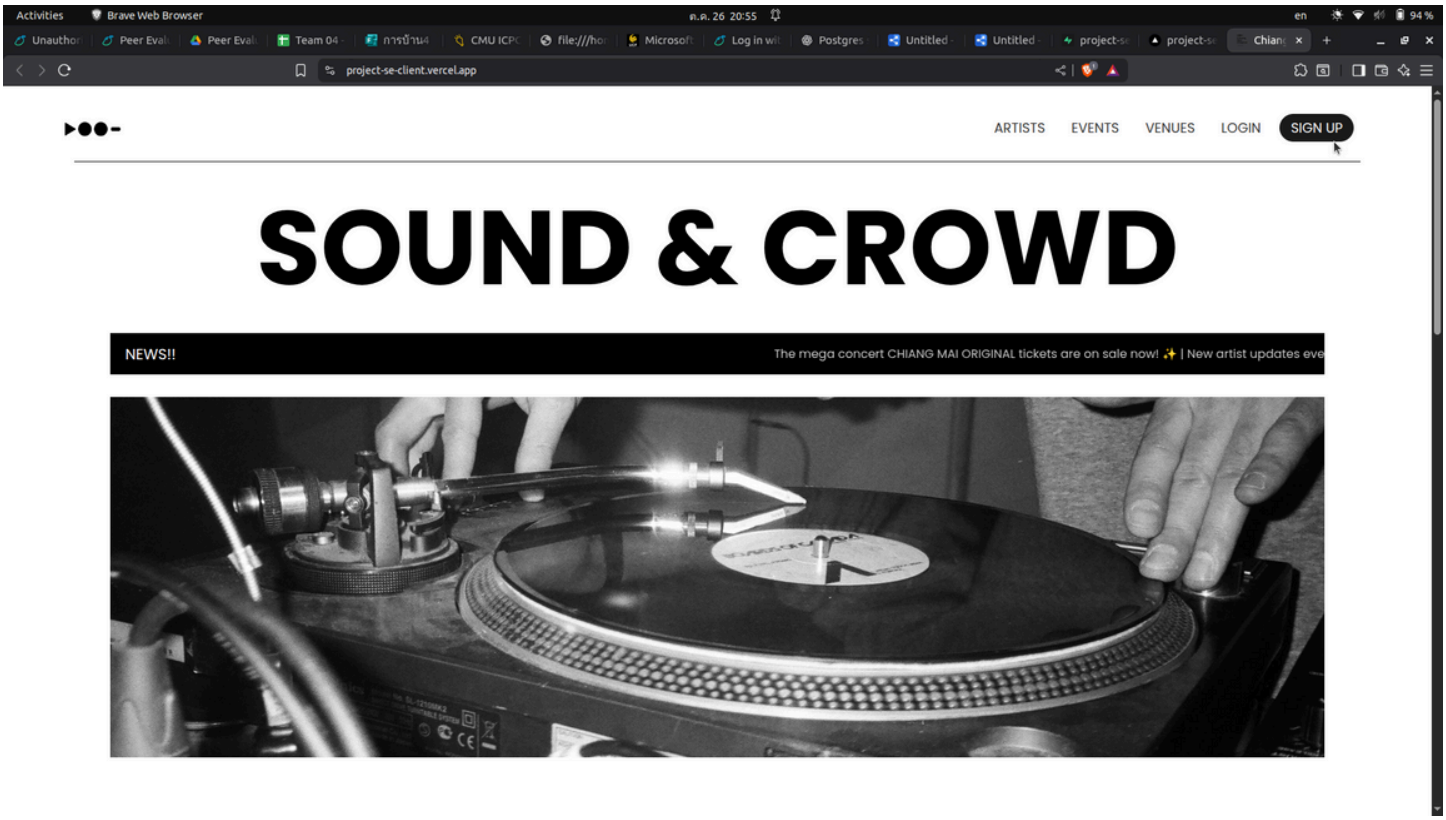


3.11 then again click environment variable and add all credential that we obtain from step 2.3.4 , surely you may notice we don't need all of it here if any at all, the thing is thats the team initial thought too until the Oauth suddenly not working and one of the teamate said that you need to put the credential relate to google Oauth in the frontend service of application too! then from that point on the team just put all env in both service and pretend it as black magic to eliminat the possibility of bug when team do not aware or remember that the frontend service need the credential from this line of .env file or not so wee decide to just put everything there.



3.12 click deploy and after this you should be able to use the web app now


p4ssionate's projects
Hobby
project-se-client
Feedback
Overview
Deployments
Analytics
Speed Insights
Logs
Observability
Firewall
AI
Storage
Flags
Settings
project-se-client
Repository
Usage
Domains
Visit
Production Deployment
Build Logs
Runtime Logs
Instant Rollback
SOUND & CROWD
Deployment
project-se-client-3hikyis6n-p4ssionates-projects.vercel.app
Domains
project-se-client.vercel.app
Status
Ready
Created
Oct 14 by nongsita
Source
main
541fbb4 fix problem cant invite artist
Deployment Settings
3 Recommendations
To update your Production Deployment, push to the main branch.
Deployments
Firewall 24h
Enable Bot Protection
Firewall is active
Observability 6h
Edge Requests
0
Function Invocations
0
Error Rate
0%
Analytics
Track visitors and page views
Enable


Activities
Brave Web Browser
project-se-client.vercel.app
ARTISTS
EVENTS
VENUES
LOGIN
SIGN UP
SOUND & CROWD
NEWS!!
The mega concert CHIANG MAI ORIGINAL tickets are on sale now! ✨ | New artist updates eve