Canva Clone

Project Setup Documentation

Getting Started:

Before we can start development. We need to make a copy of the project started files into our machine. The project's source code contains all the scripts and references to the third-party libraries that we will need to develop our web application.

To clone the code to your local machine, open the vs code, navigate to the directory where you keep your projects.

Example:

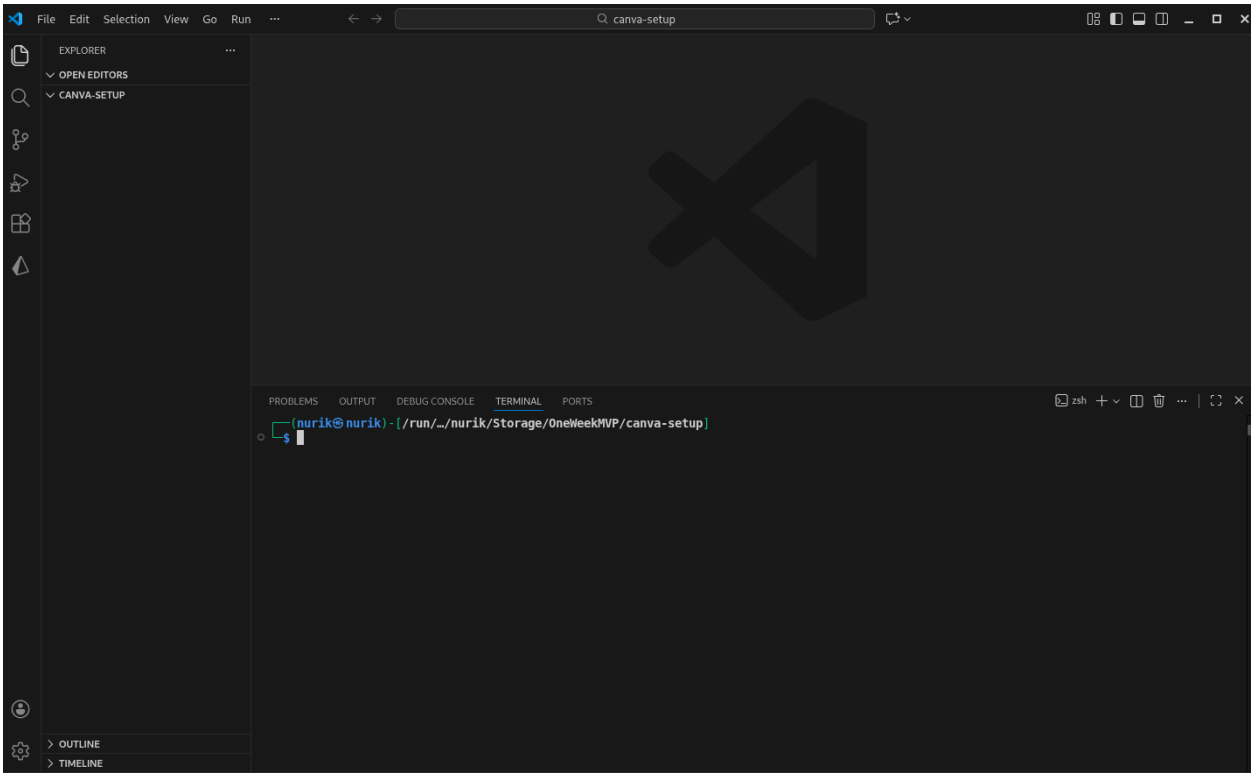


Fig.1: Create a project folder.

Cloning Setup:

- Go to: https://github.com/OneWeekMVP/canva-clone
- Paste: git clone https://github.com/OneWeekMVP/canva-clone.git
- Change directory: cd canva-clone

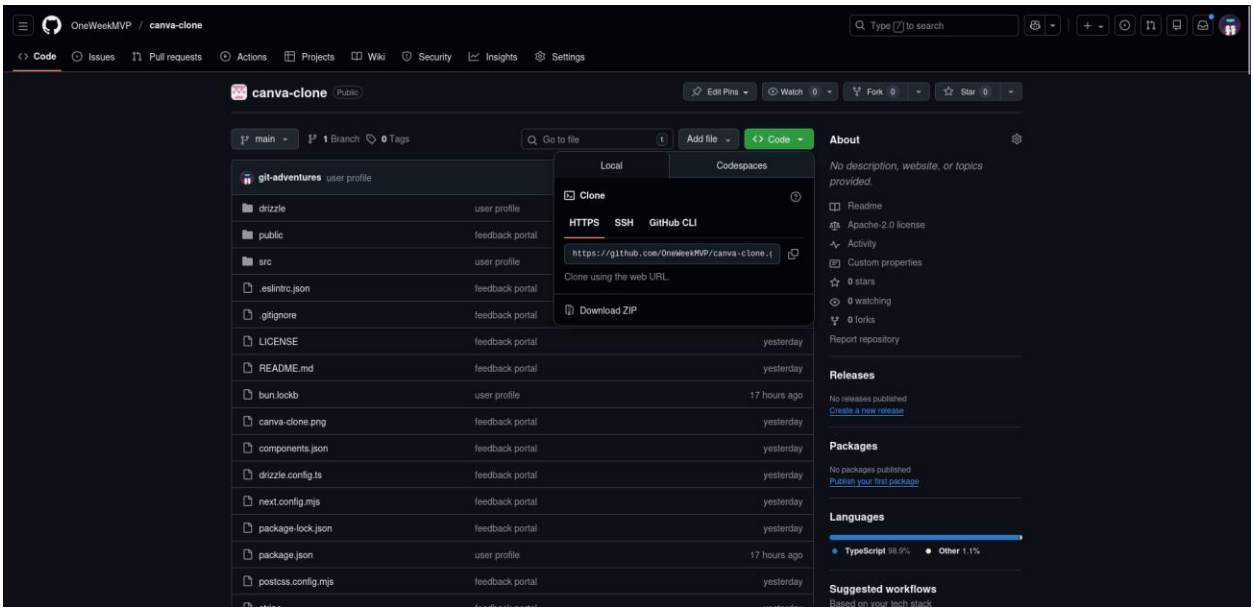


Fig.2: Grab the HTTPS URL.

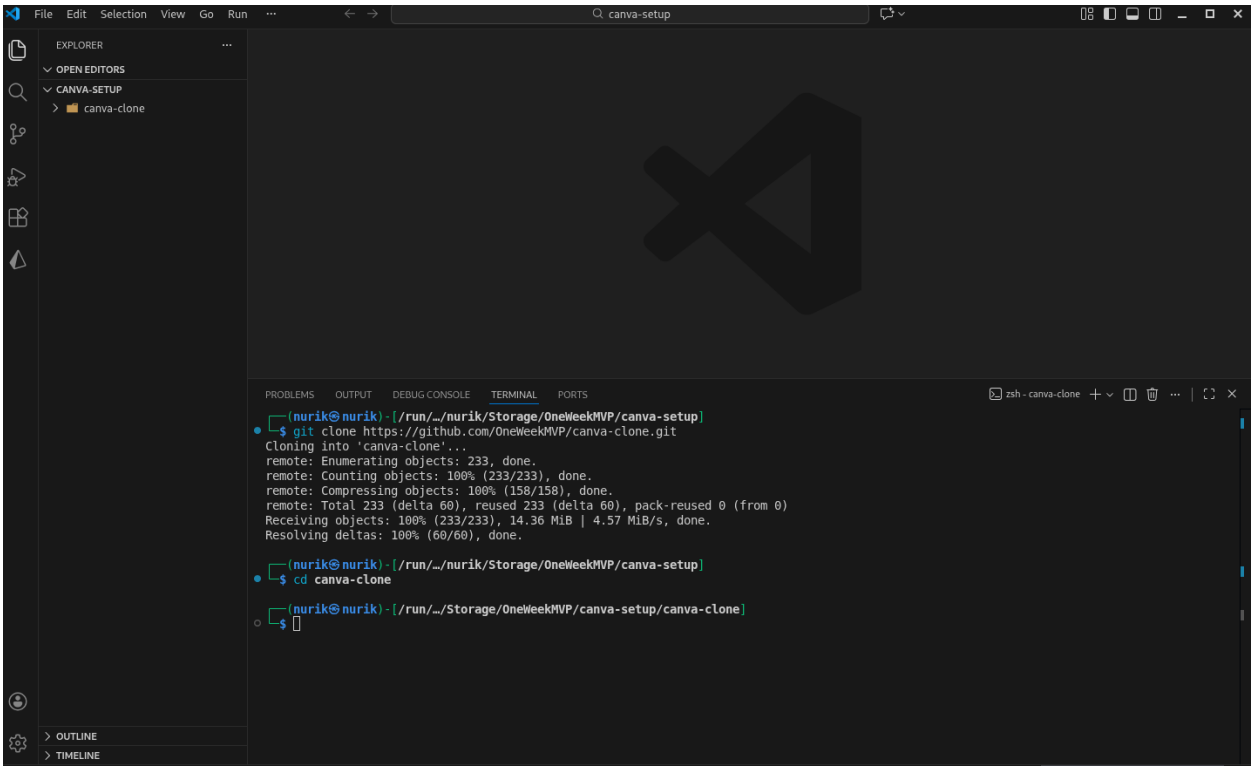


Fig.3: Clone and change the directory.

Once you are in, check for your bun, bunx versions since I will be using bun, you don't have to if you want to continue with npm, npx:

$ bun --version

1.3.5

$ bunx --verison

1.3.5

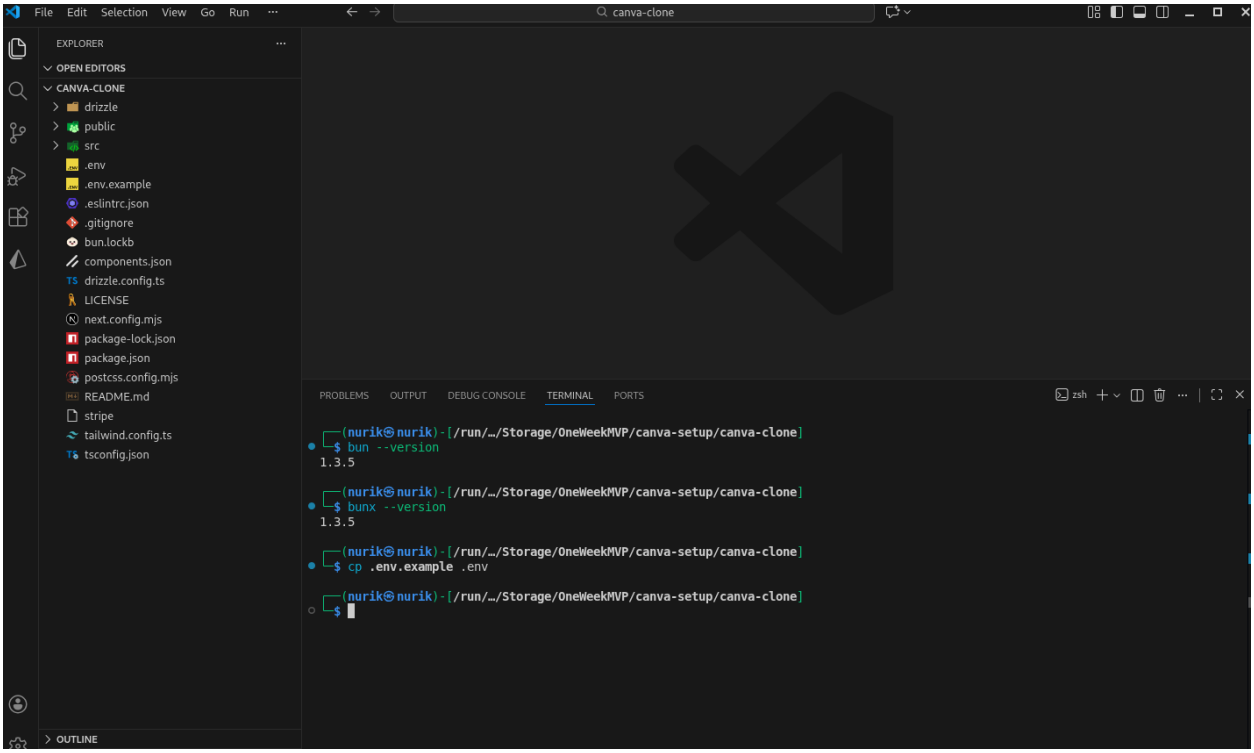


Fig.4: Check for versions and copy .env.example

Note: We are doing this inside the canva-clone directory, so you should navigate your terminal to it.

Now, run:

$ bun install

*bun install, it downloads all the necessary packages for the project and saves them within node_modules.

$ bun dev

Now, if you run it, you will encounter an error telling you:

Error: No database connection string was provided to `neon()`. Perhaps an environment variable has not been set?

It means **the database URL is missing** — the environment variable for your database connection (e.g. DATABASE_URL) is **not set or not loaded**.

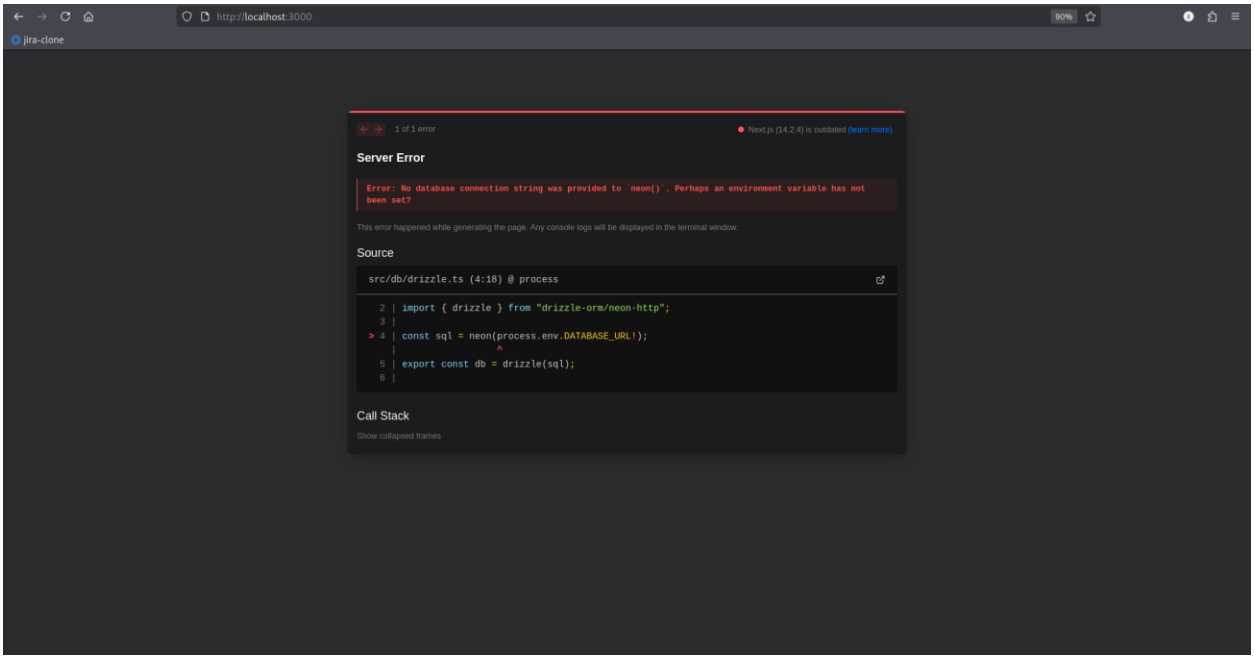


Fig.5: Database URL, missing it in .env file

Now, we need to establish a database connection with Neon DB:

- Go to: https://neon.com
- Create an account, come up with organization name, and finish the common procedure.

Once you are in, you will see a project's dashboard where it stores your projects. In my case, I have had sever of them. You should see blank.

- Go and press on 'New project'
- Name your project anyhow you like and press create

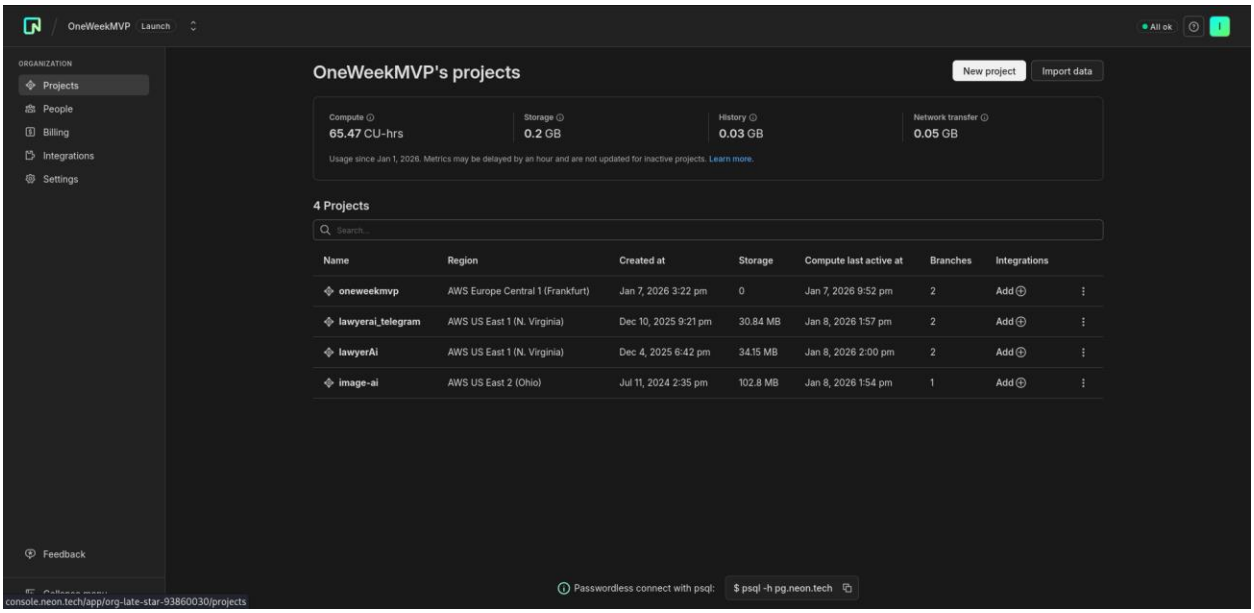


Fig.6: Neon database

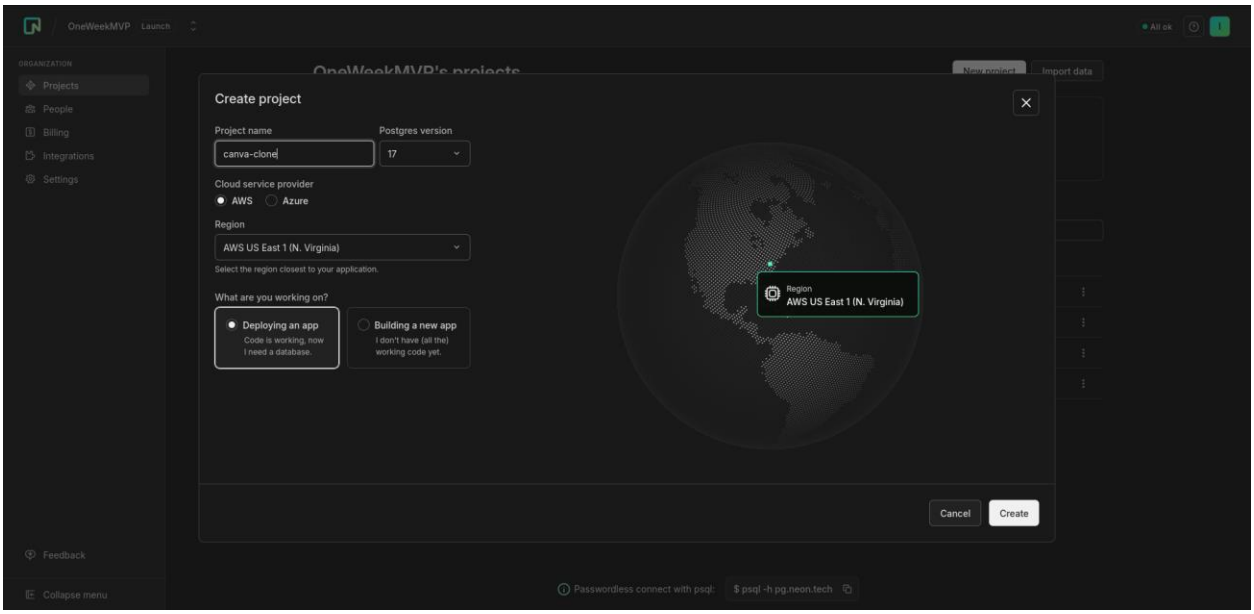


Fig.7: Database creatoin

From there, you will be redirected to your project, where you will now need to connect your psql (PostgreSQL).

- Press on ‘Connect’
- Copy your connection string

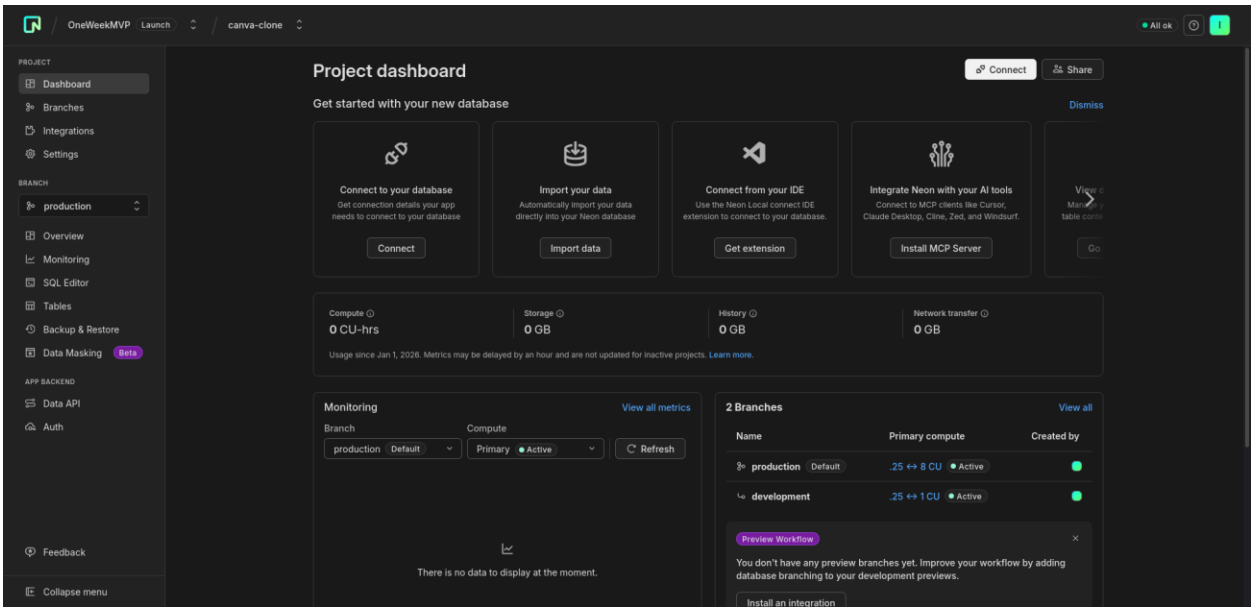


Fig.8: Project Dashboard

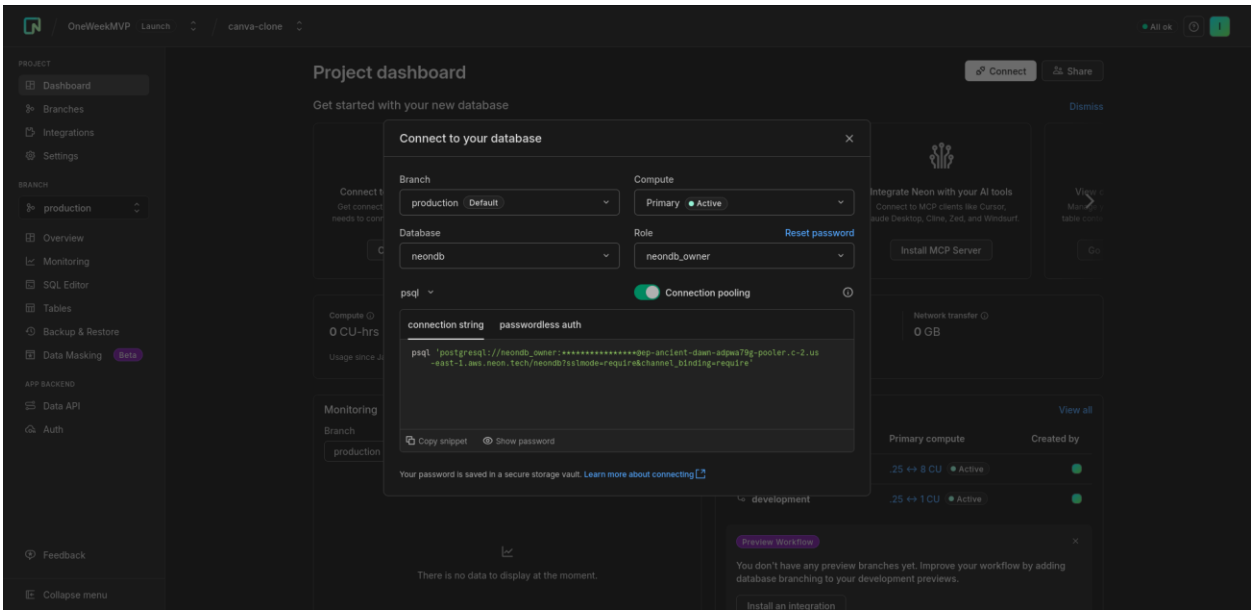


Fig.9: Copy your connection string

Pass it to your .env file, next to **DATABASE_URL=**

- Remove the last '&channel_binding=require' string
- Save

While we are at it, let's also create **AUTH_SECRET=**

$ openssl rand -base64 32

* **AUTH_SECRET** - Used to encrypt and sign JWT tokens, cookies, and session data for NextAuth.js (Auth.js). Without it, authentication won't work.

openssl rand -base64 32 - Generates a cryptographically secure random 32-byte string encoded in base64 format. It's a secure way to create secret keys.

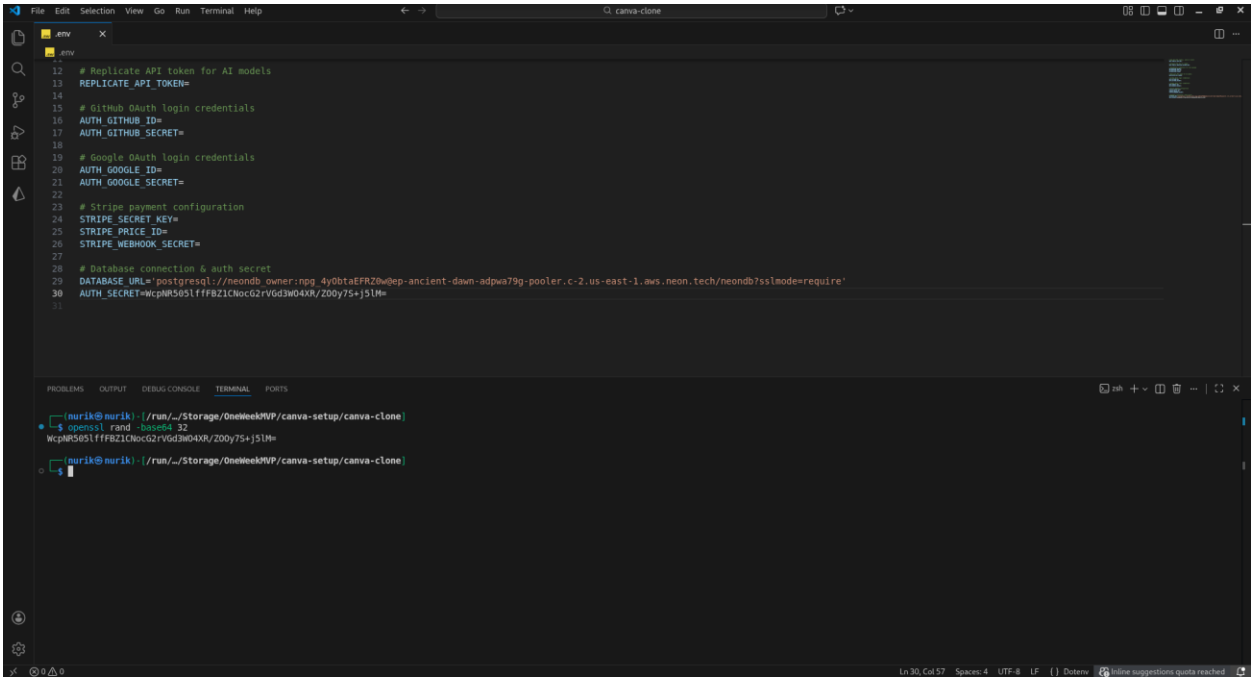


Fig.10: **DATABASE_URL** && **AUTH_SECRET**

Now, after this, run the following commands in a row:

$ bun run db:generate

$ bun run db:migrate

What do they do?

- bun run db:generate → generates database schema files from your Drizzle config
- bun run db:migrate → runs database migrations (applies schema changes)

These commands are short-cuts; you can find them in package.json

"db:generate": "bunx drizzle-kit generate",

"db:migrate": "bunx drizzle-kit migrate",

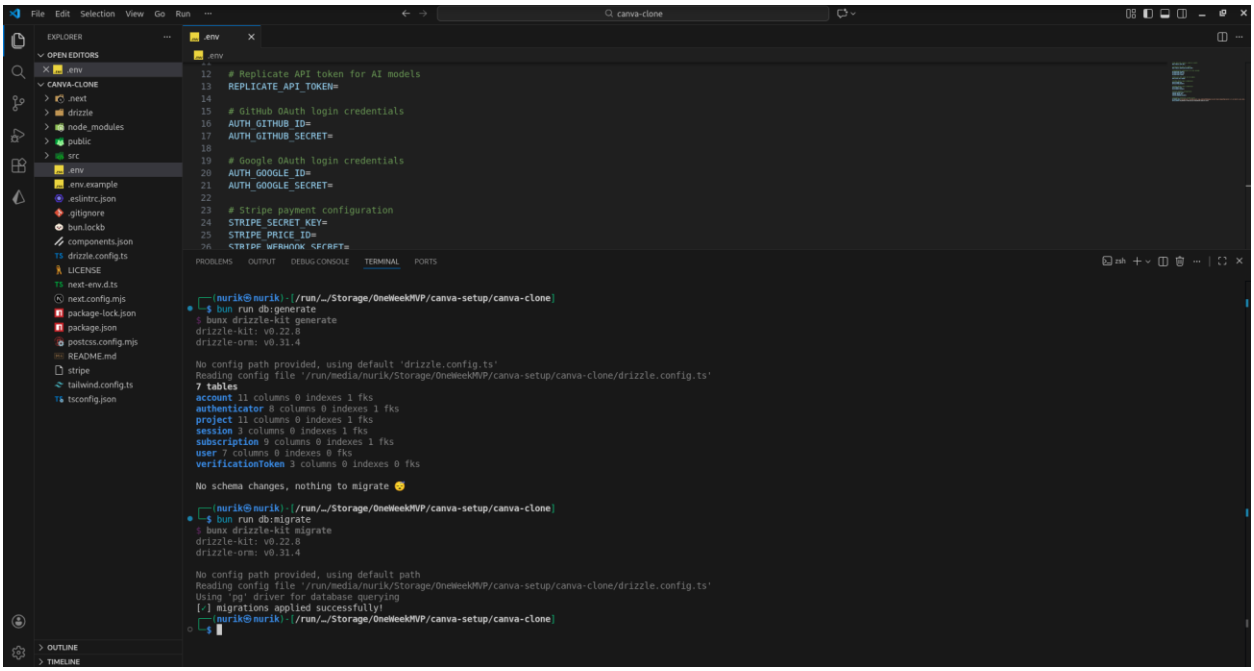


Fig.11: Database schema && migration process

After that, you can also run:

$ bun run db:studio

- bun run db:studio → opens Drizzle Studio (GUI to view/edit the database)

Drizzle Studio is up and running on https://local.drizzle.studio, or you can also see the changes from your Neon database.

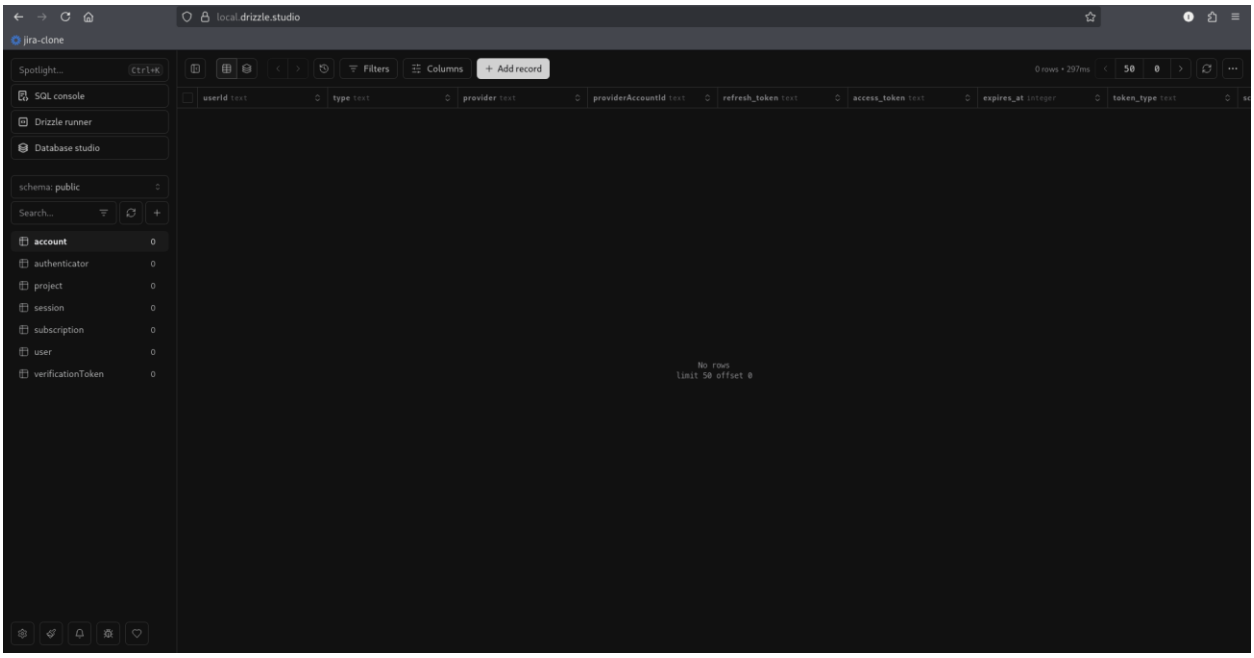


Fig.12: Drizzle studio, where your data is saved

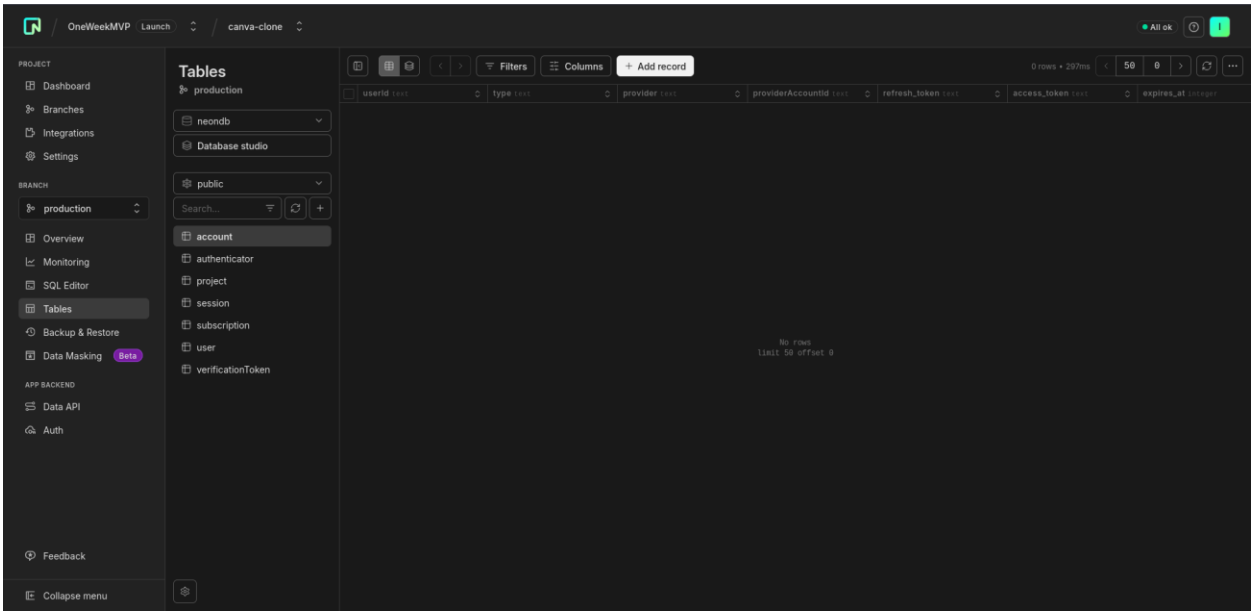
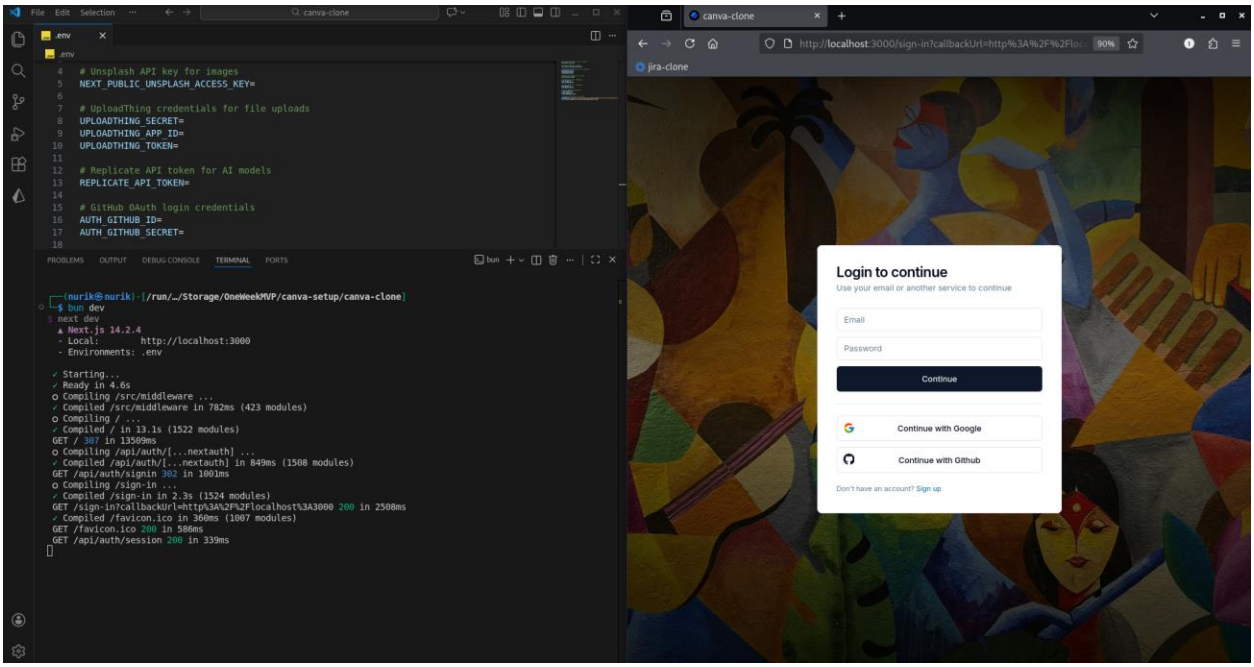


Fig. 13: Drizzle via Neon db

And now if you can run bun dev, you will have it working on your localhost:3000

$ bun dev



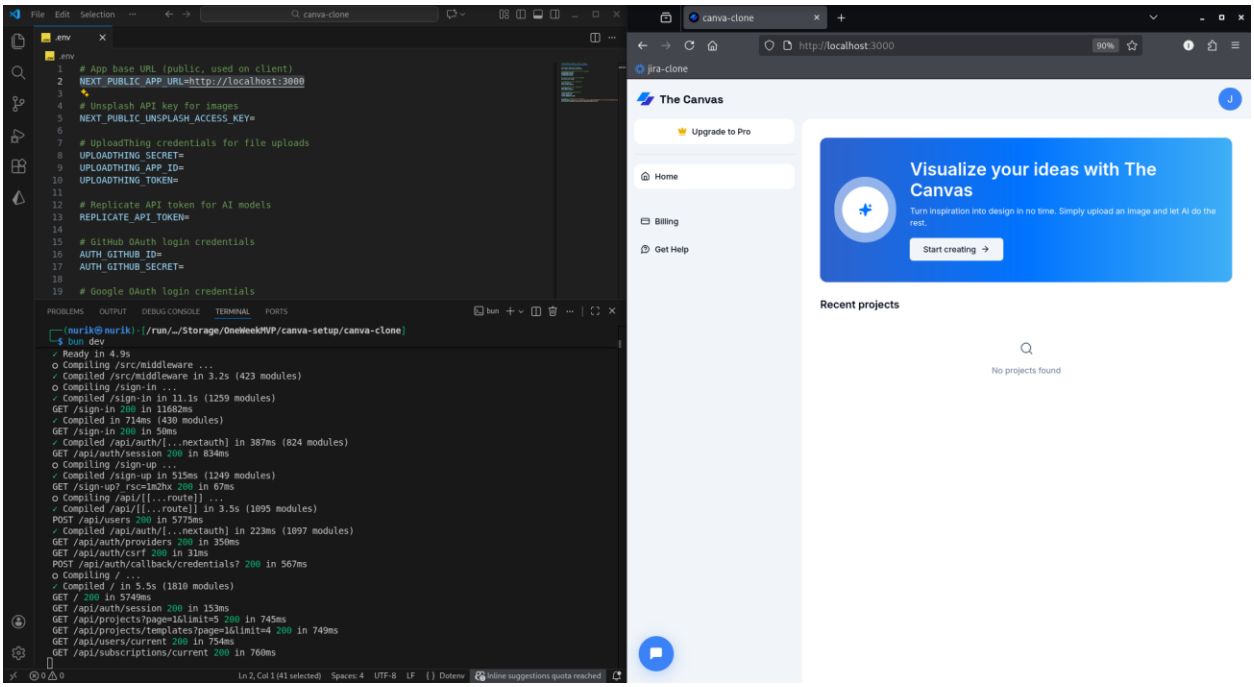
Fig,14: Now it is working on localhost:3000

Now fill your **NEXT_PUBLIC_APP_UR**L with your localhost:3000

Like this: **NEXT_PUBLIC_APP_URL=http://localhost:3000**

We will later change the URL after deployment to its production name.

Go ahead and create an account, you should be able to create one, and you will be redirected to your dashboard page.



Fir.15: Dashboard Page

It is working! Pow, pow. Wow.

Now, go ahead and create a project. A project will be created. If you look at the left side panel, you will notice several tools or call them features. Select the ‘Image’ section. Because now, we enable images using Unsplash API and Uploadthing API.

- Unsplash API → used to fetch high-quality stock photos programmatically
- UploadThing API → used to upload, store, and manage files (images, videos, etc.) easily

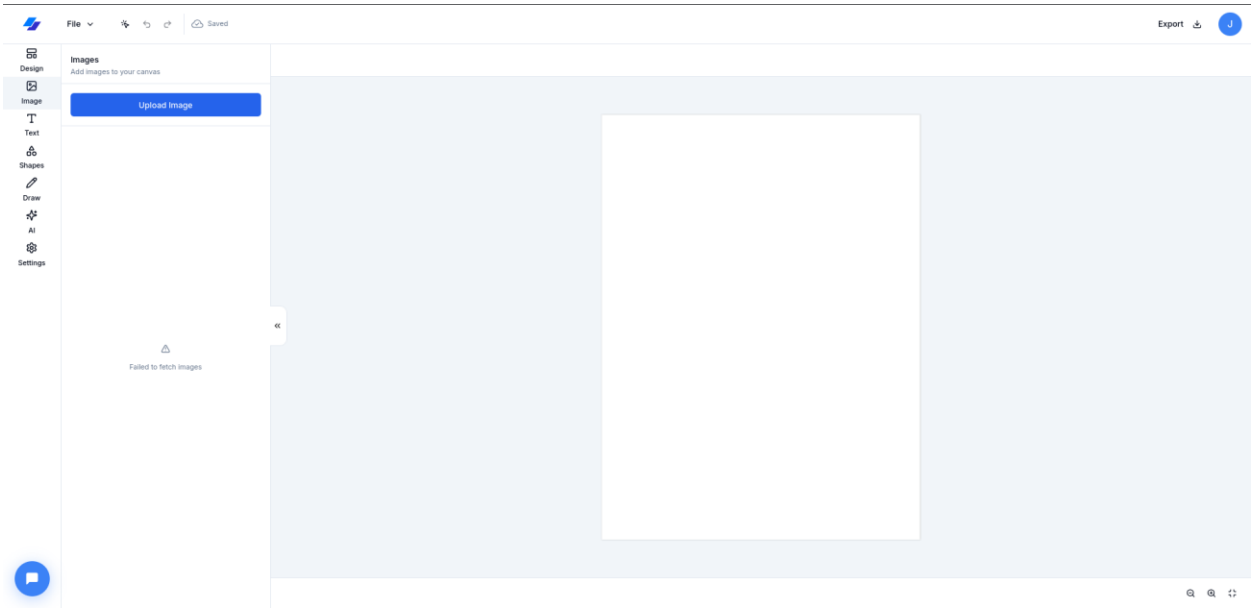


Fig.16: Before the integration with Unplash and Uploadthing APIs

Go to: https://unsplash.com

Log in or create an account, go with the procedure.

- Once you logged in, find developer's api in 'Product' section
- Click on 'Your apps'

Note: If your browser is timed out or for some reason you can't seem to get in. Just download Tor browser like I did and use Tor. You can find it in here: https://www.torproject.org/download

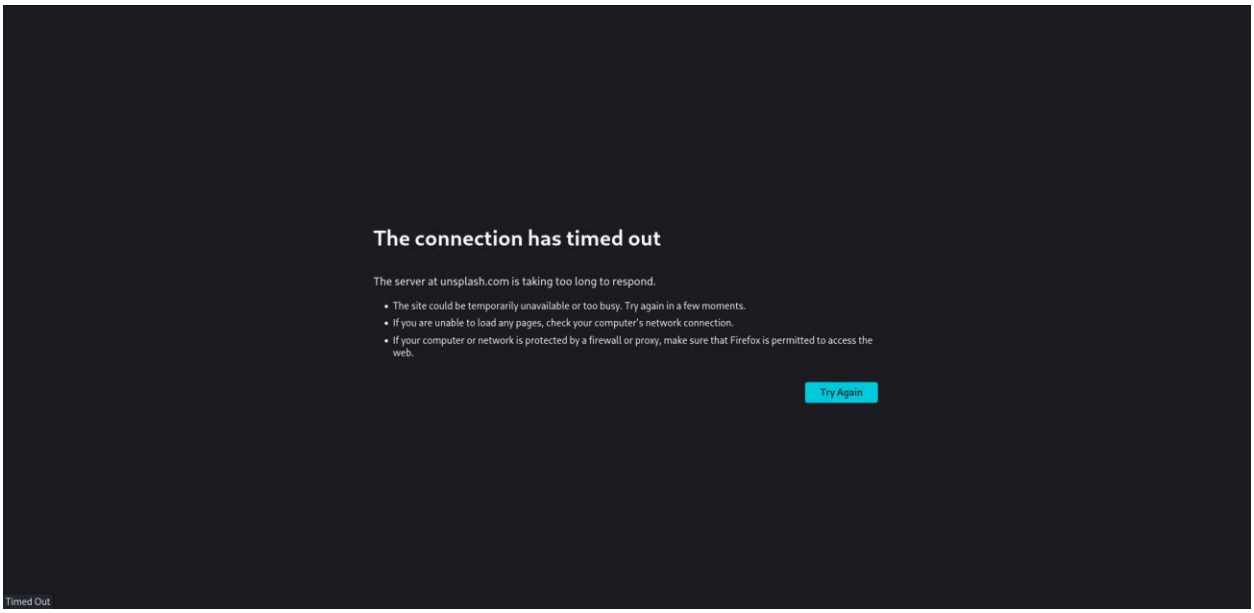


Fig.16.1: Connection Error

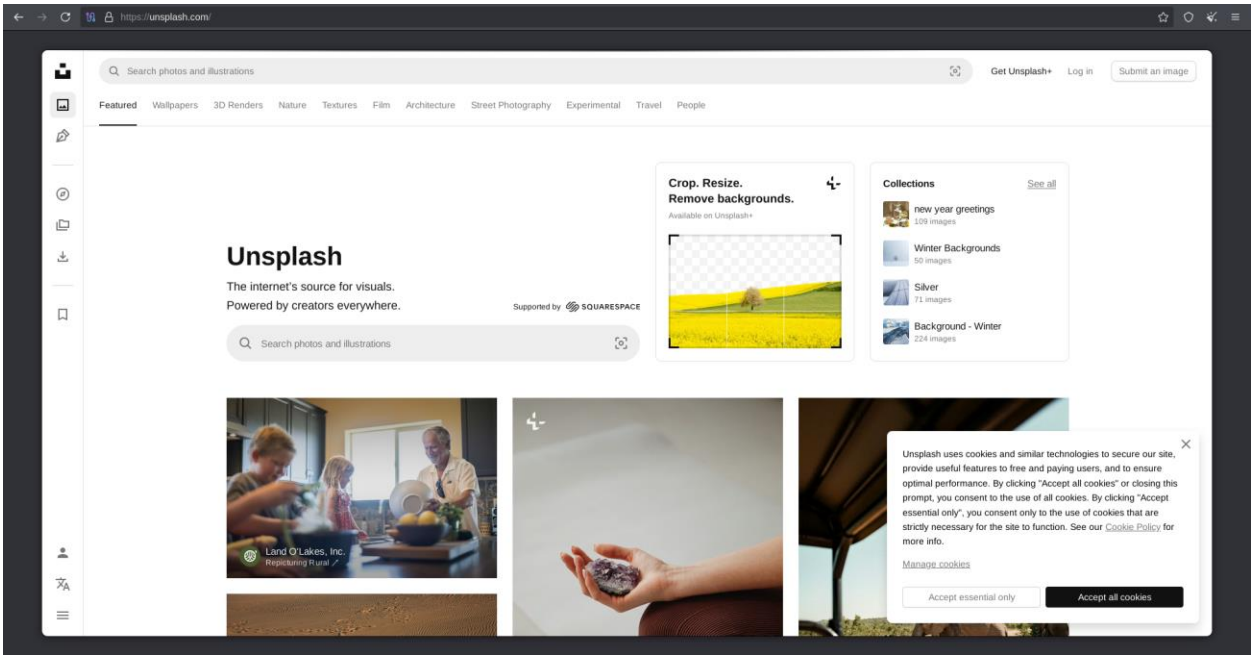


Fig.17: Unsplash via Tor

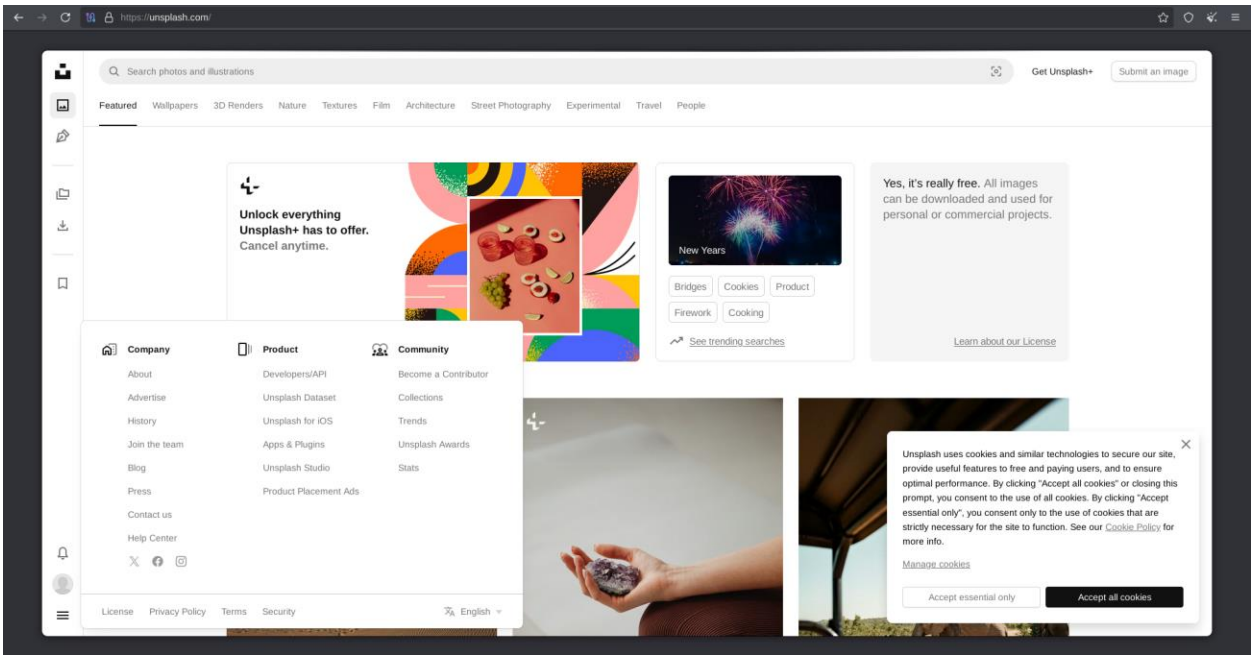


Fig.18: Select Developers API

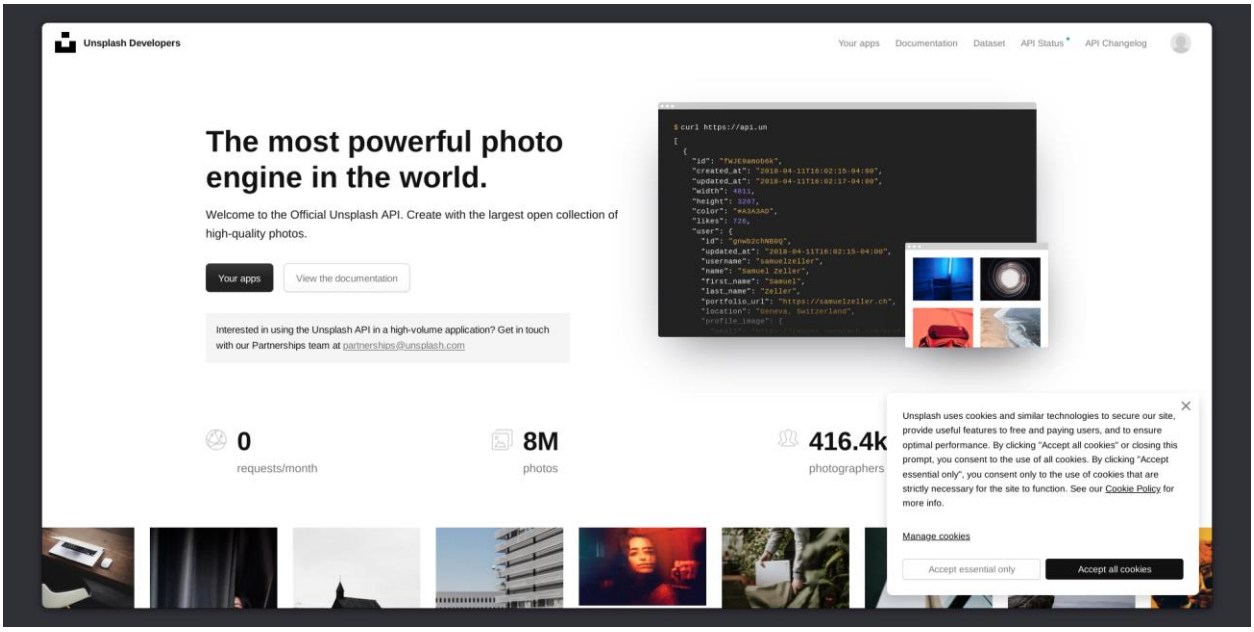


Fig: 19: Select ‘Your apps’

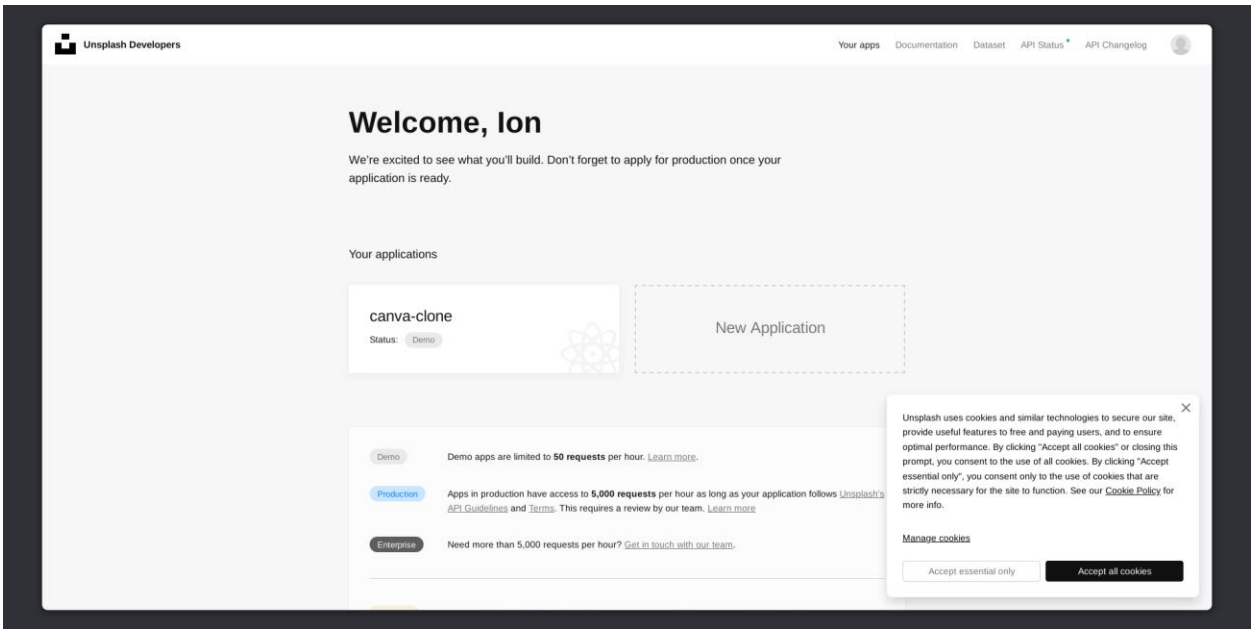


Fig.20: Create an app

Go ahead and create a new app since I used it before this page is not blank for me.

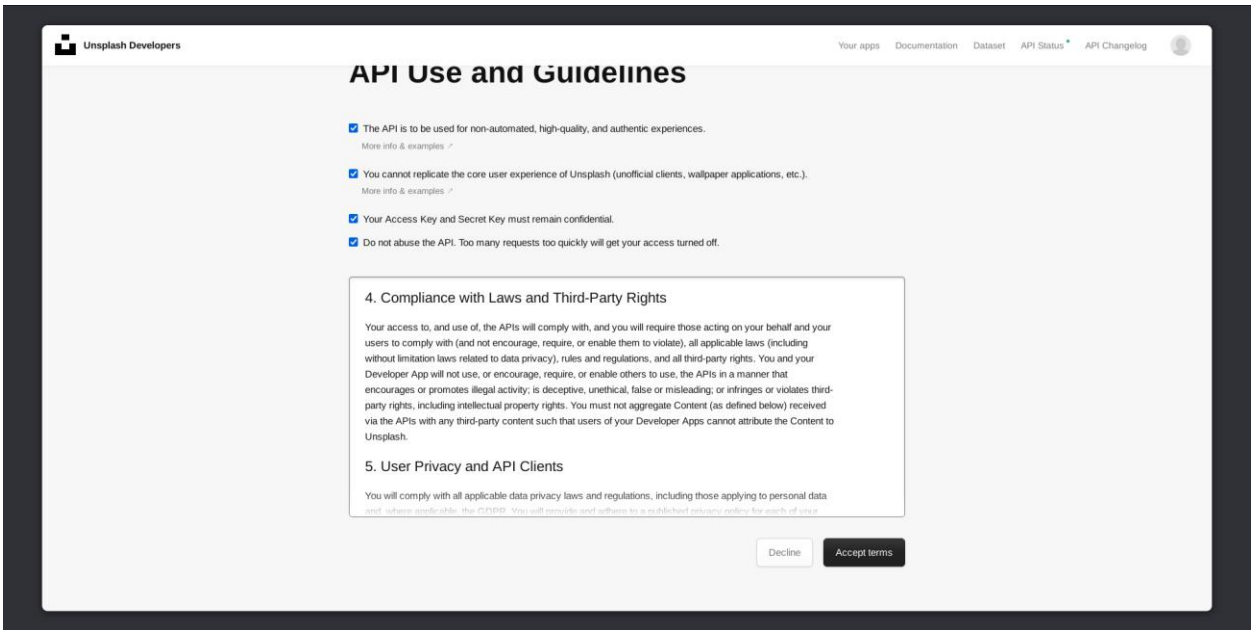


Fig.21: Accept terms

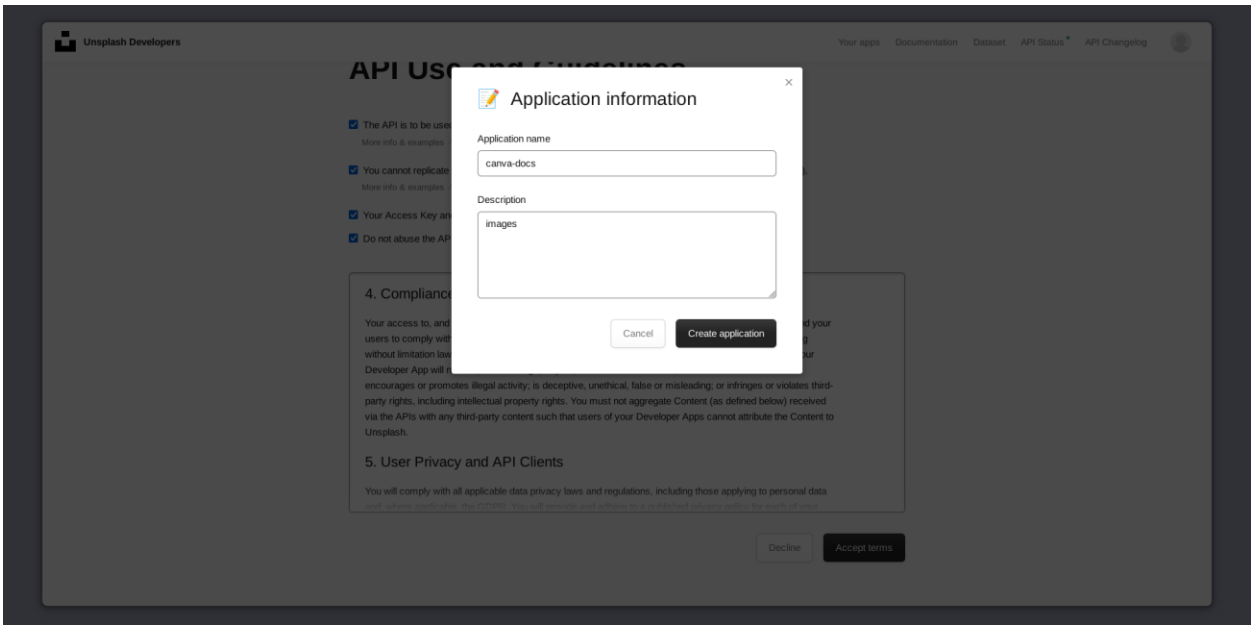


Fig.22: Name your app with a few words

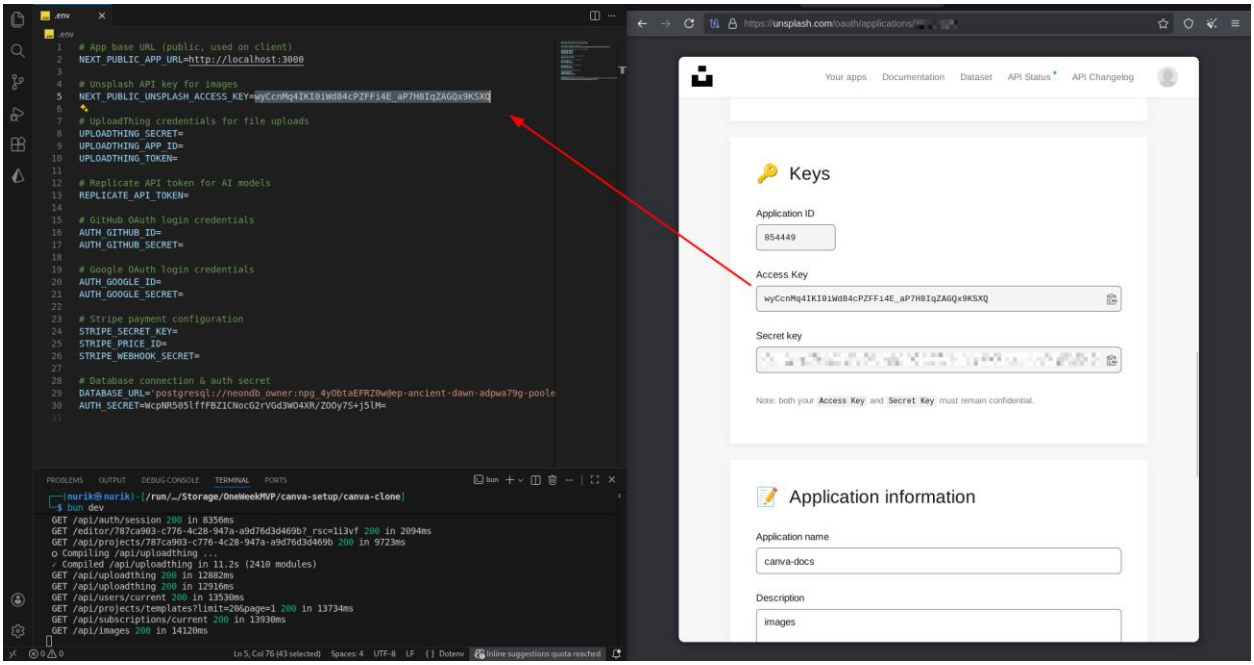


Fig.23: Grab your Access Key

Get your access key and put it in the environment file.

- **NEXT_PUBLIC_UNSPLASH_ACCESS_KEY=**
- Restart your localhost:3000 and see again!

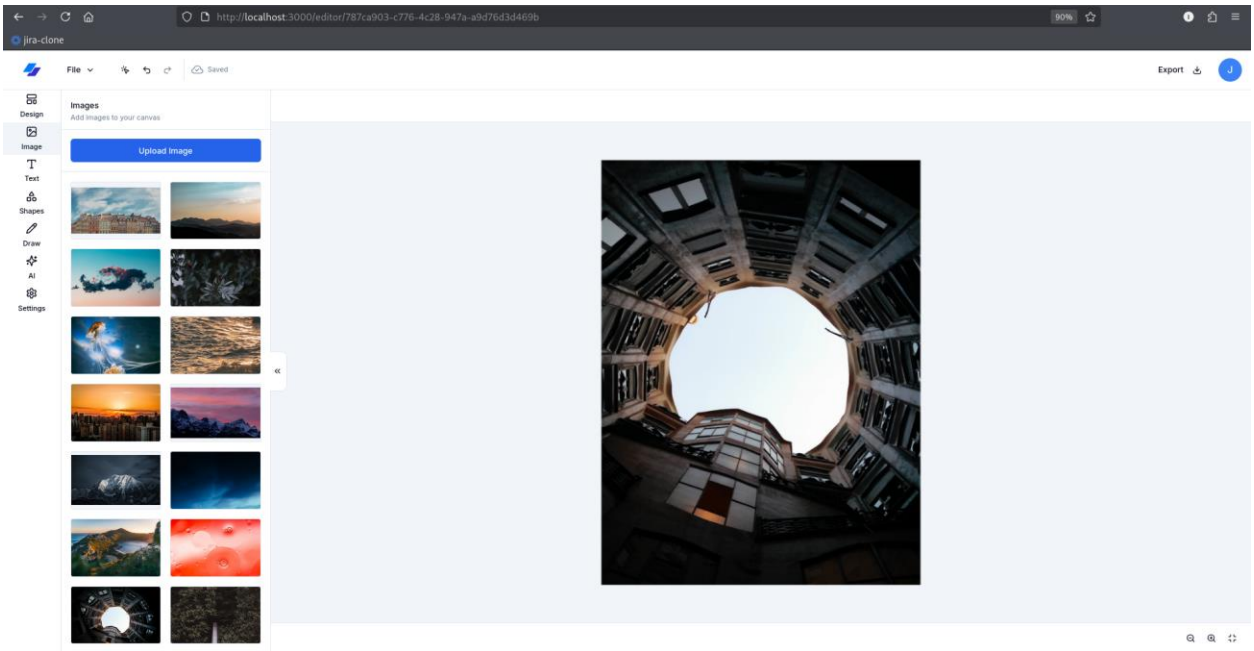


Fig.24: Unsplash is set

Now we need to enable the Upload Image button using Upload thing, go to: https://uploadthing.com

- Get Started for Free
- Log in or create an account

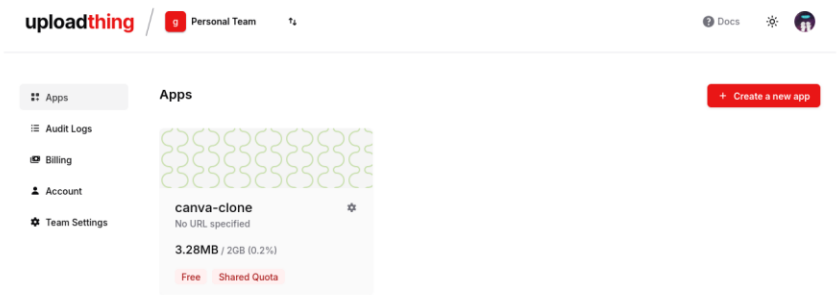


Fig.25: Projects Dashboard

Go and create a new app:

- Just give a name for your app
- Select Free option
- And press on ‘create app’

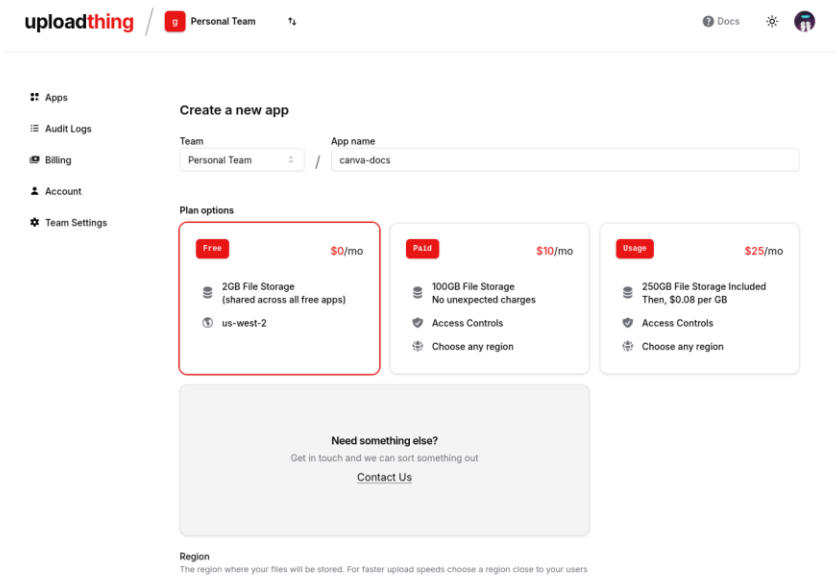


Fig.26: Name && Select Free

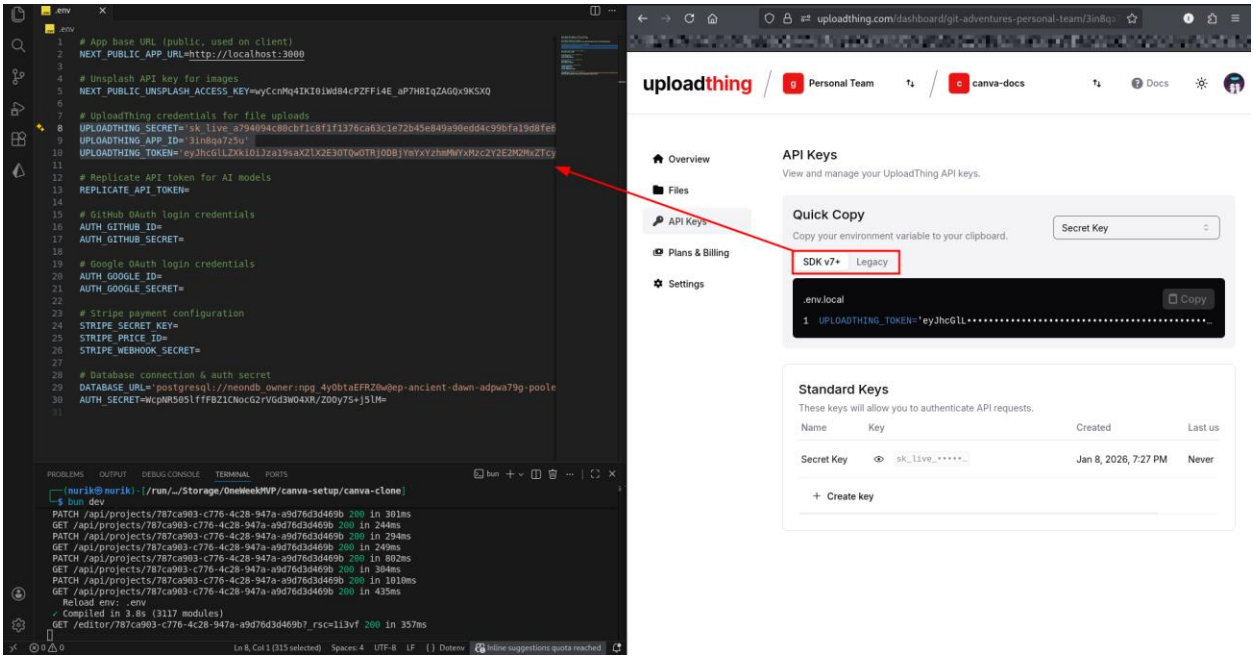


Fig. 27: SDK v7+ && Legacy

Select both your SDK and Legacy keys and pass them in the environment file.

UPLOADTHING_SECRET=

UPLOADTHING_APP_ID=

UPLOADTHING_TOKEN=

Once you save it, go to your localhost:3000 and refresh. And try to upload an image

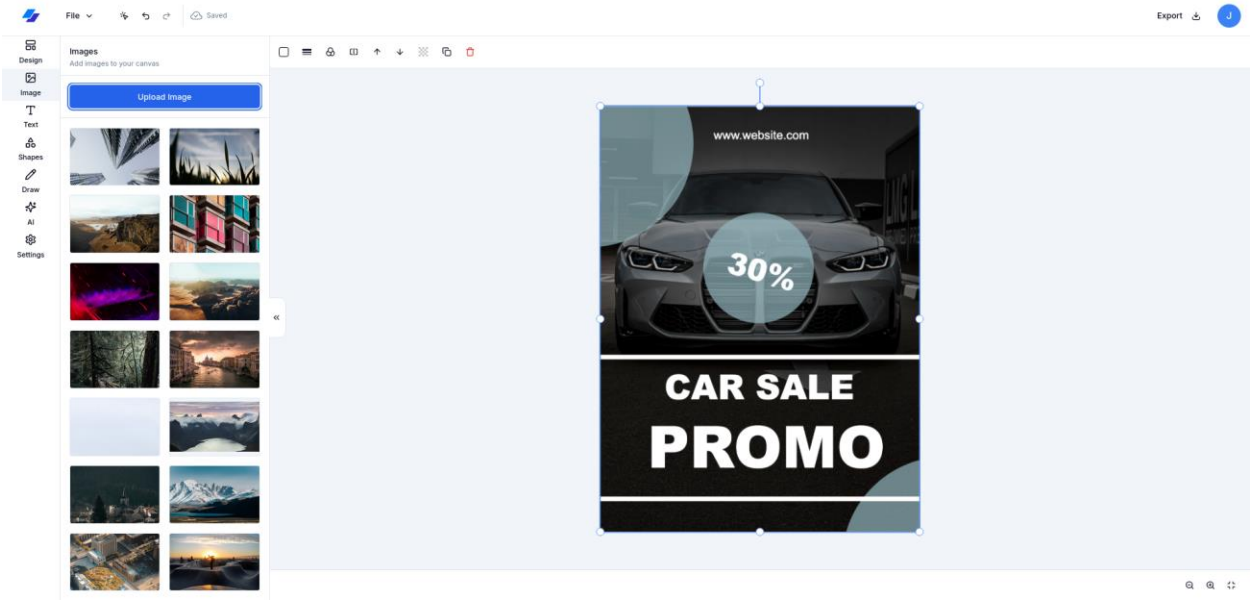


Fig.28: Upload Image

It works! Pow, pow. Wow.

And now, we need to revive our AI image generator.

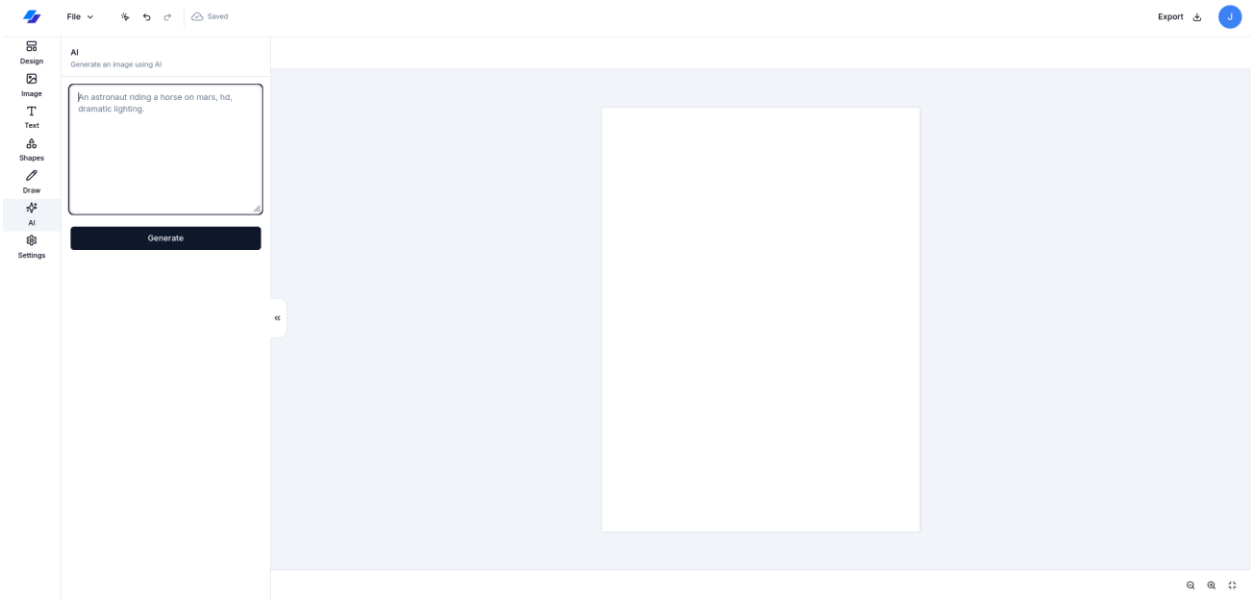


Fig.29: Before AI

Now, we should get our AI token. Go to https://replicate.com/account/billing

As always, go through account creation procedures. Once you are in, well, this one requires adding billing credentials since it generates high quality images. You should have at least $2 in your replicate account. Once you have some credentials to continue, go to API tokens and create a token.

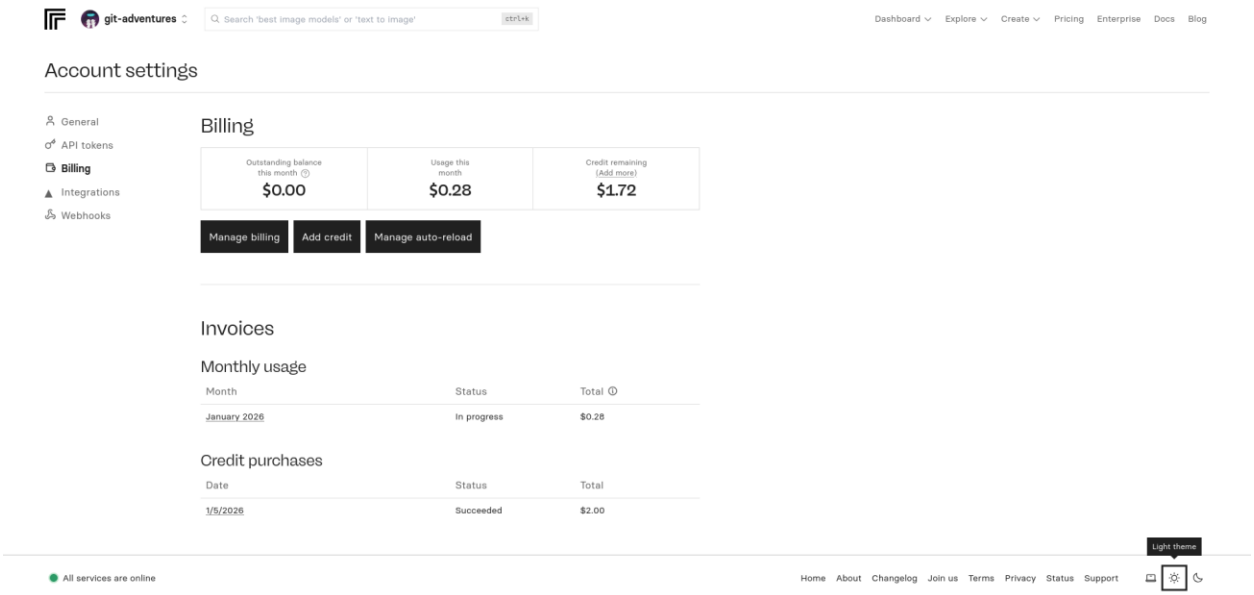


Fig.30: Replicate Billing Page

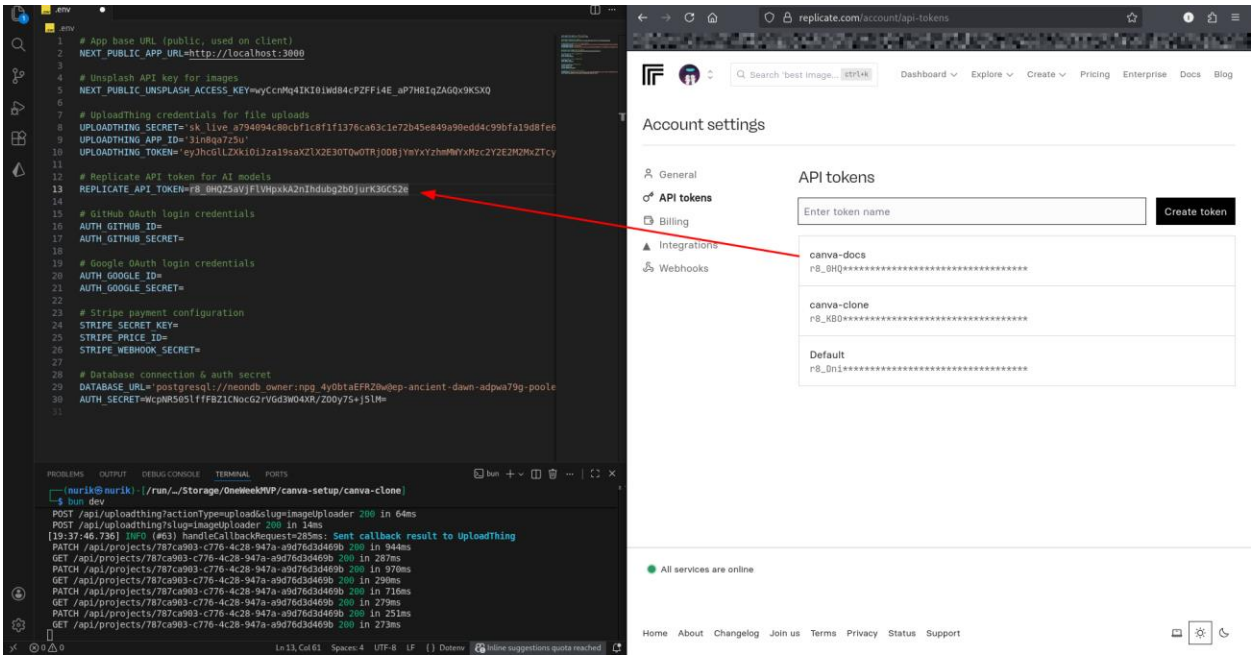


Fig.31: Replicate token

Save it. And go to your localhost:3000, one hard refresh and you are good to go.

REPLICATE_API_TOKEN=your_replicate_token

Guess what? I have forgotten about the Paid Plan. However, I am 100% sure that it works. Trust me. Next, we will connect the Stripe, so that we can enjoy using paid plan features.

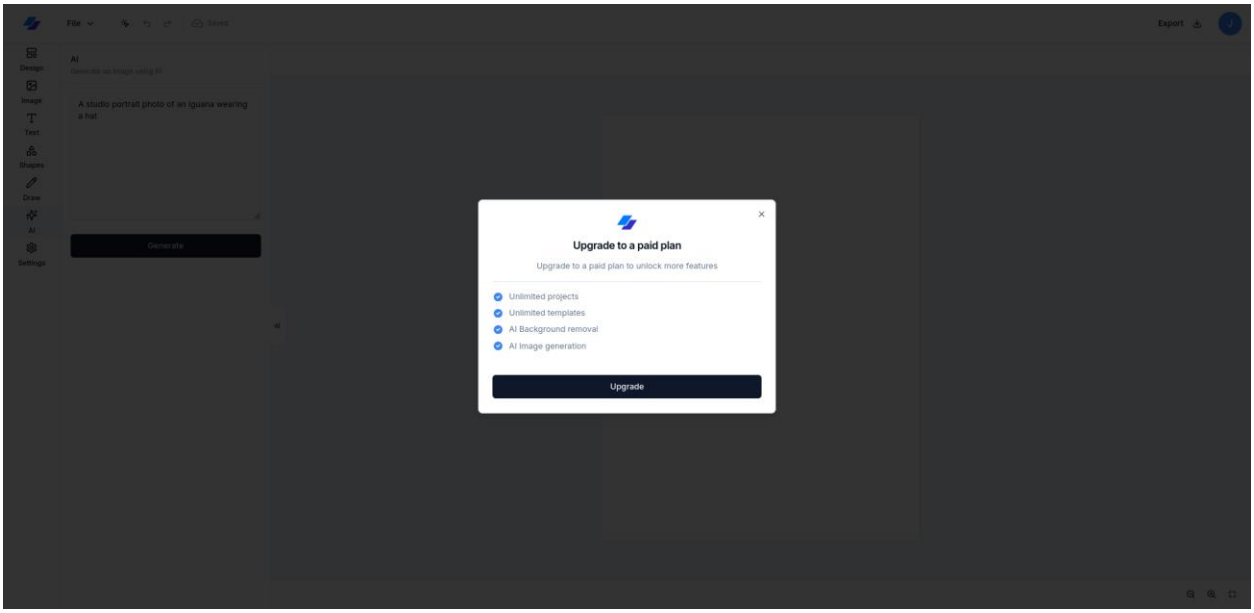


Fig.32: Paidwall && Next is Stripe

Go take some air, drink tea or coffee, your favorite, just relax. Because this time with Stripe, it is a bit complex. But I guarantee that it will be worth it.

Go to https://dashboard.stripe.com and create a Stripe account if you don't have.

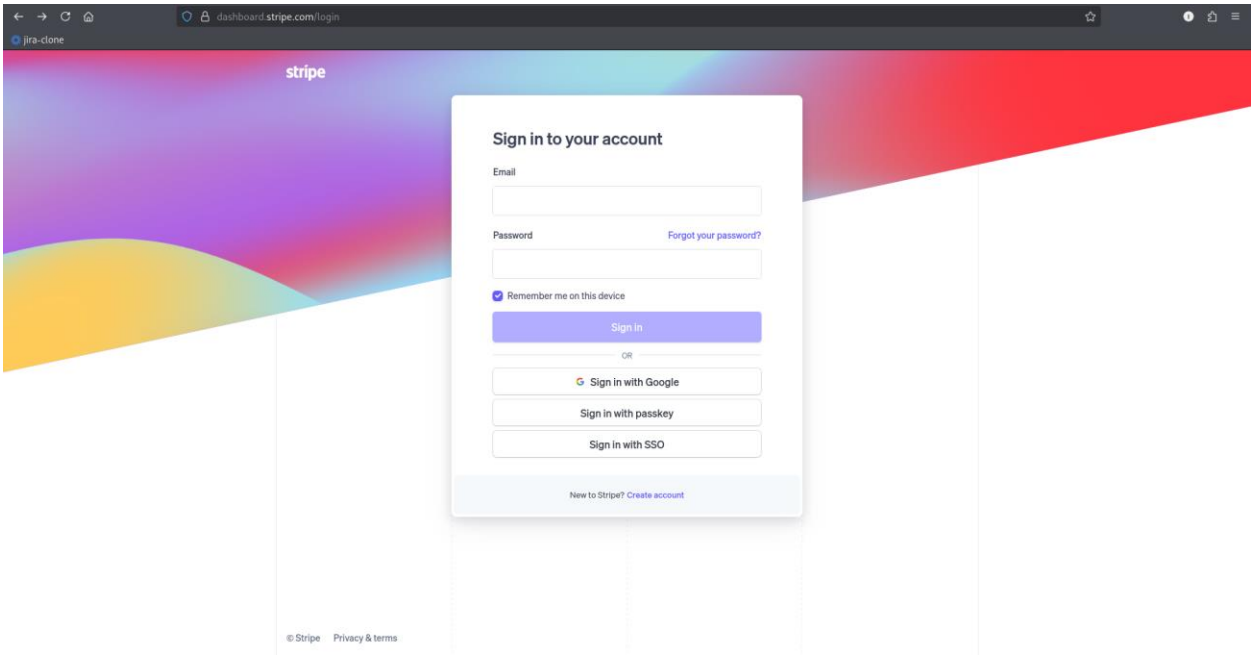


Fig.33: Stripe Dashboard

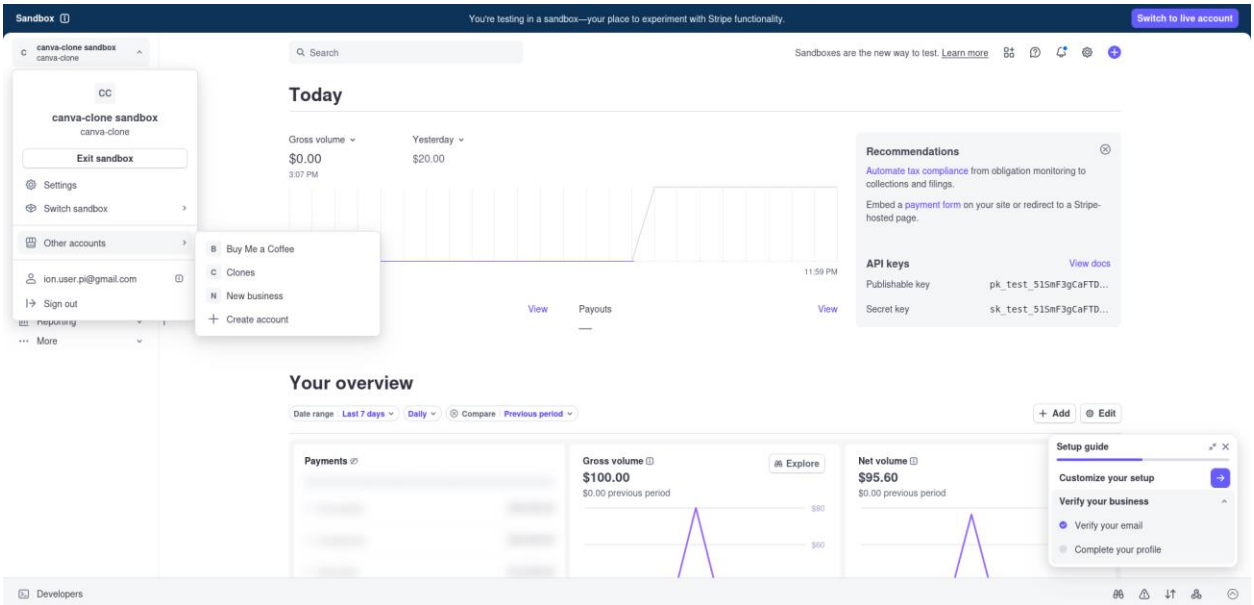


Fig.34: Create a new account

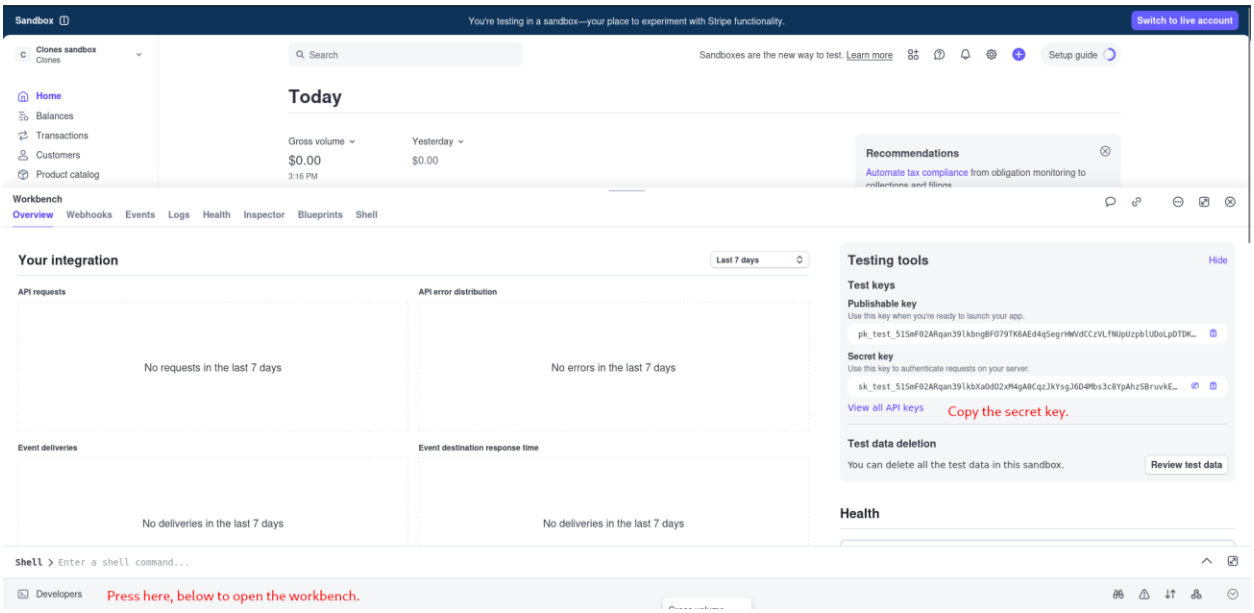


Fig.35: Secrete key

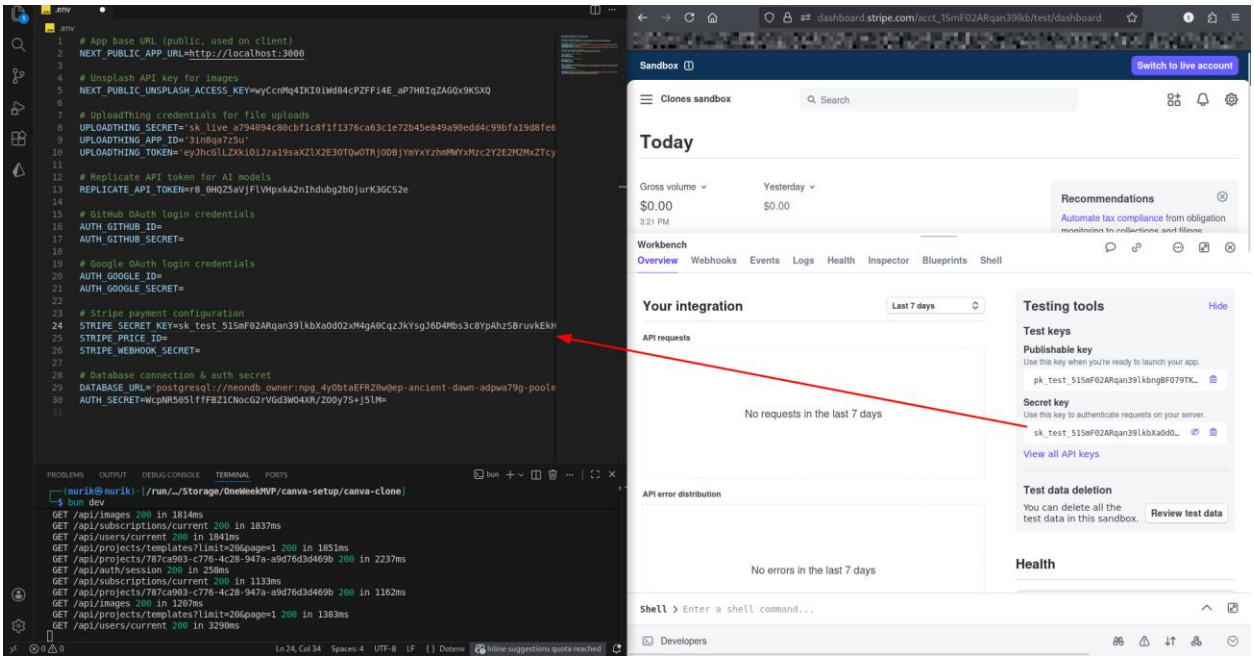


Fig.35.1: Fill the Stripe Secret Key in .env file

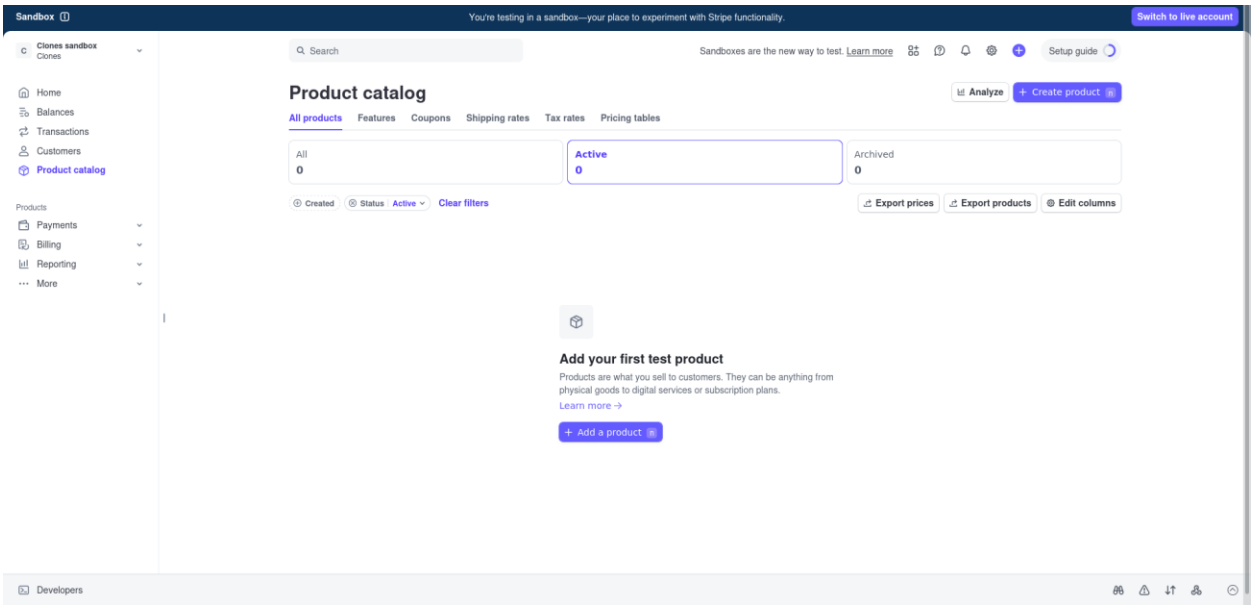


Fig.36: Adding product

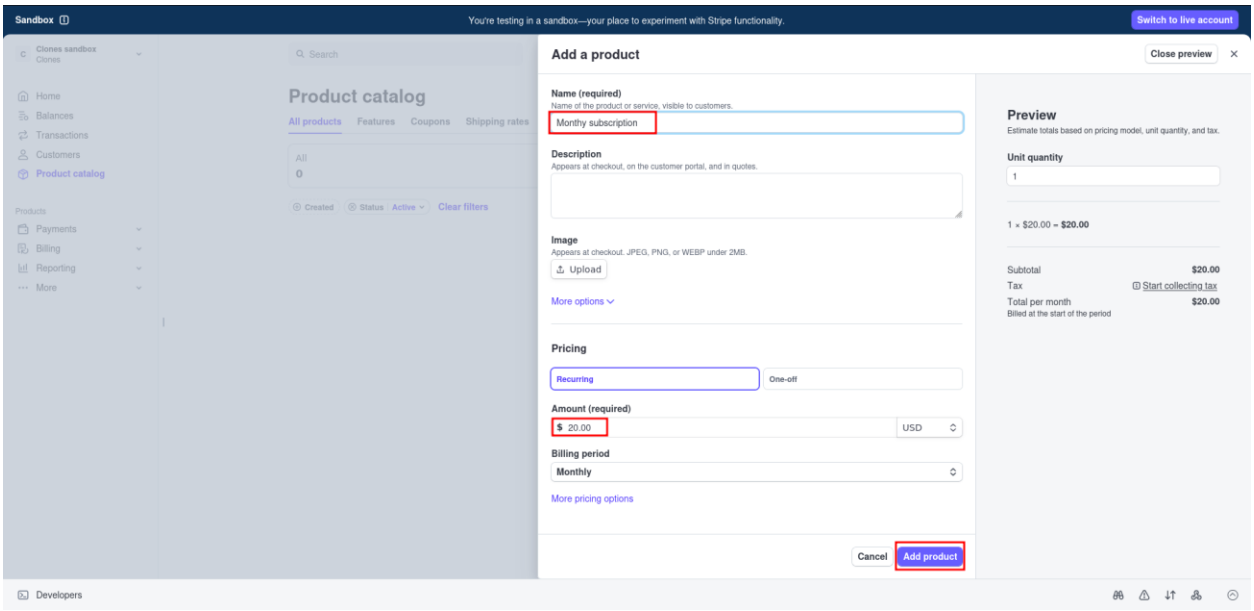


Fig.37: Product

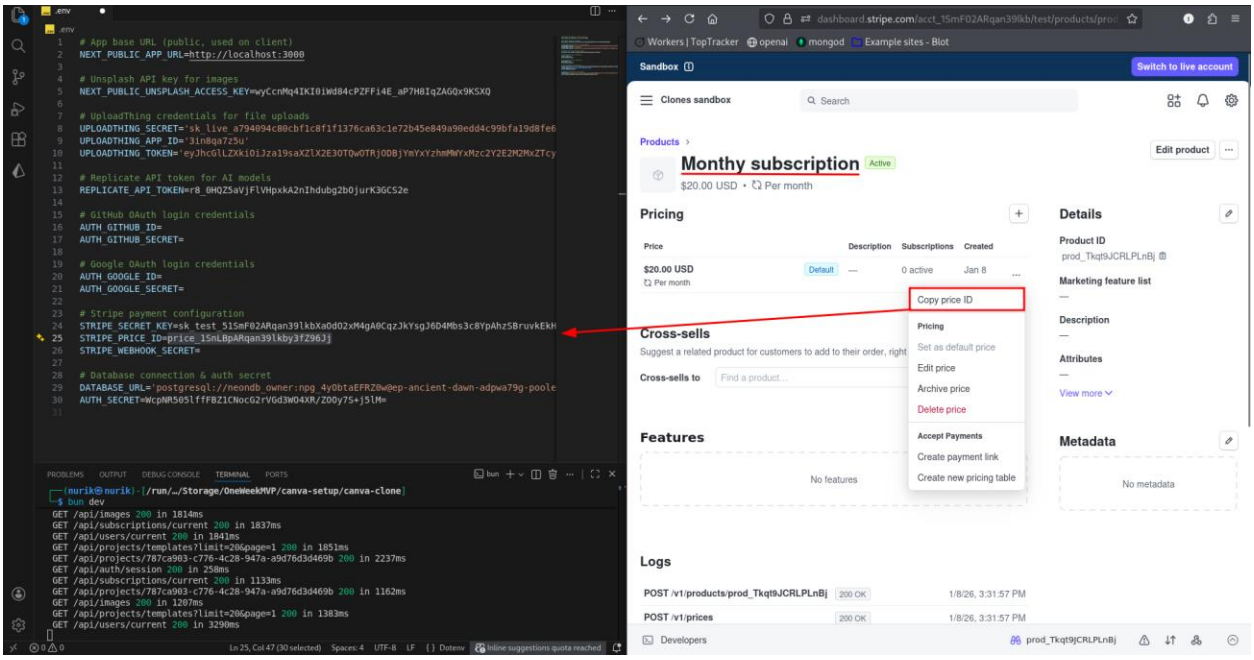


Fig.38: Copy price ID

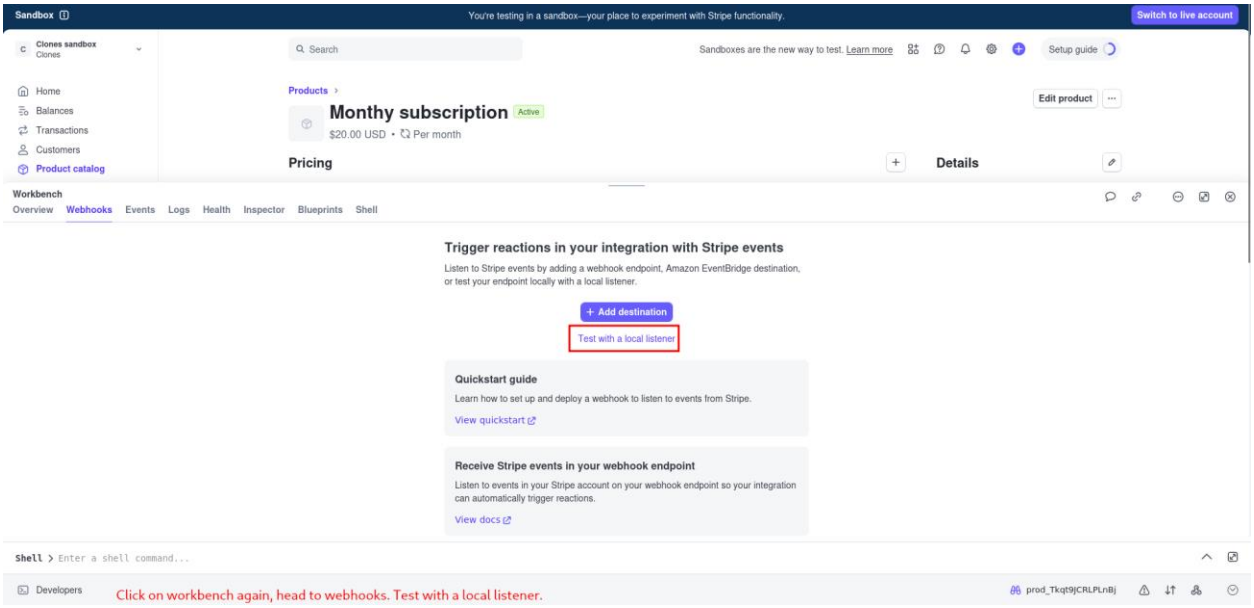


Fig.39: Test with a local listener

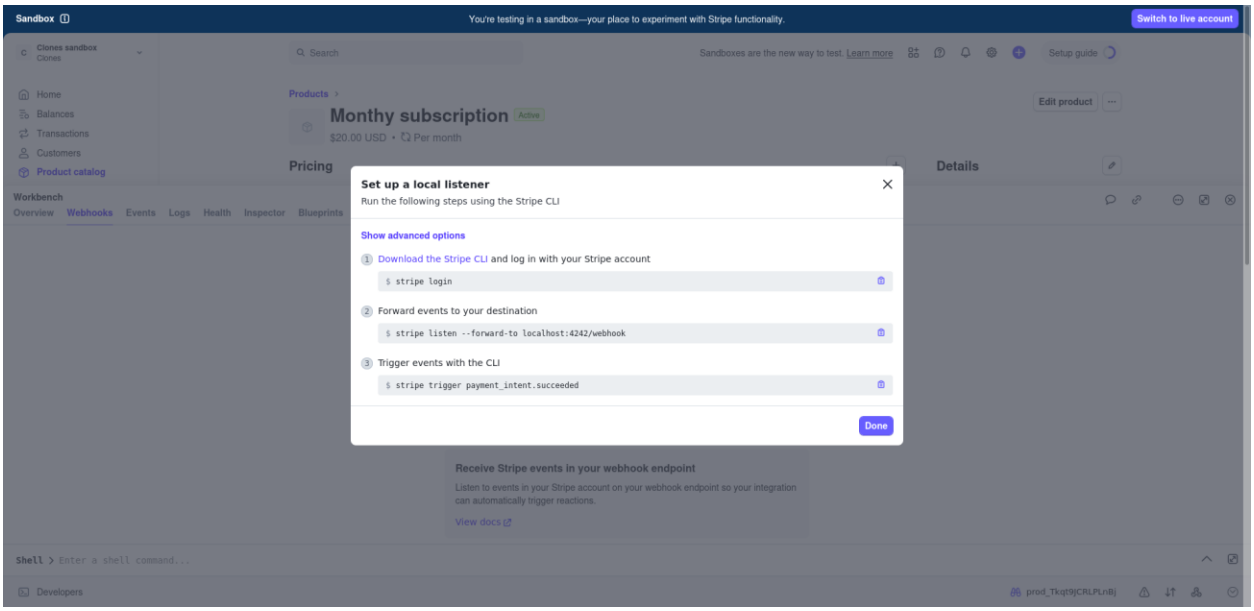


Fig.40: Local listener setup

Since I have already downloaded Stripe CLI for Linux users, they can use it, but you have to choose your laptop compatible one: https://docs.stripe.com/stripe-cli/install and install it in your root project folder.

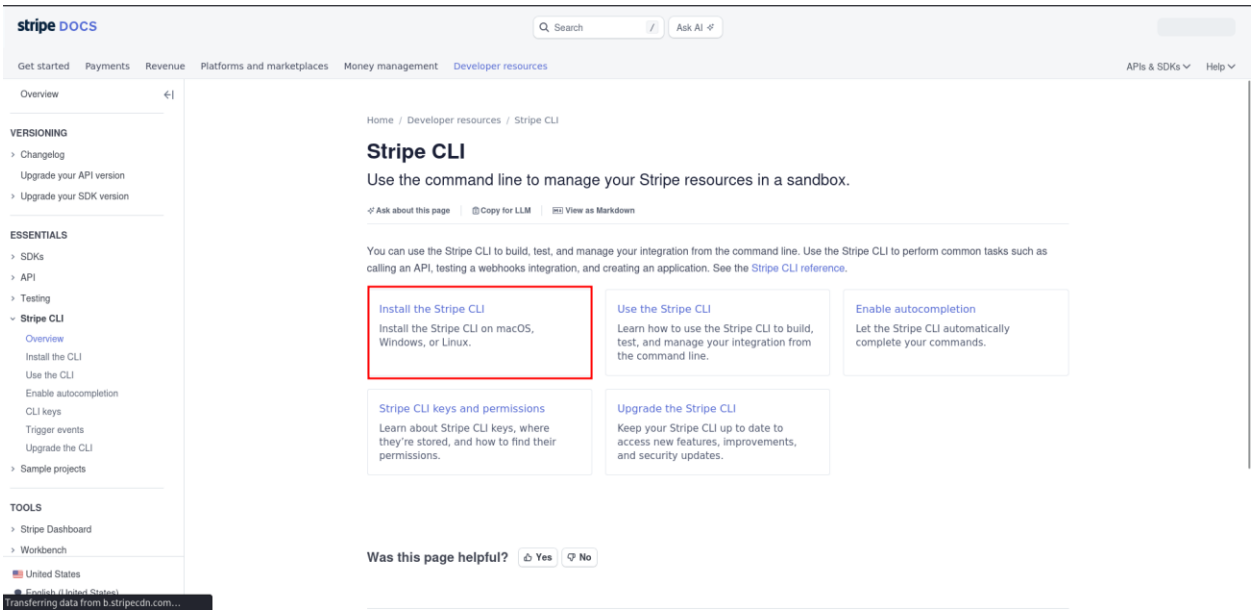


Fig.41: Stripe CLI

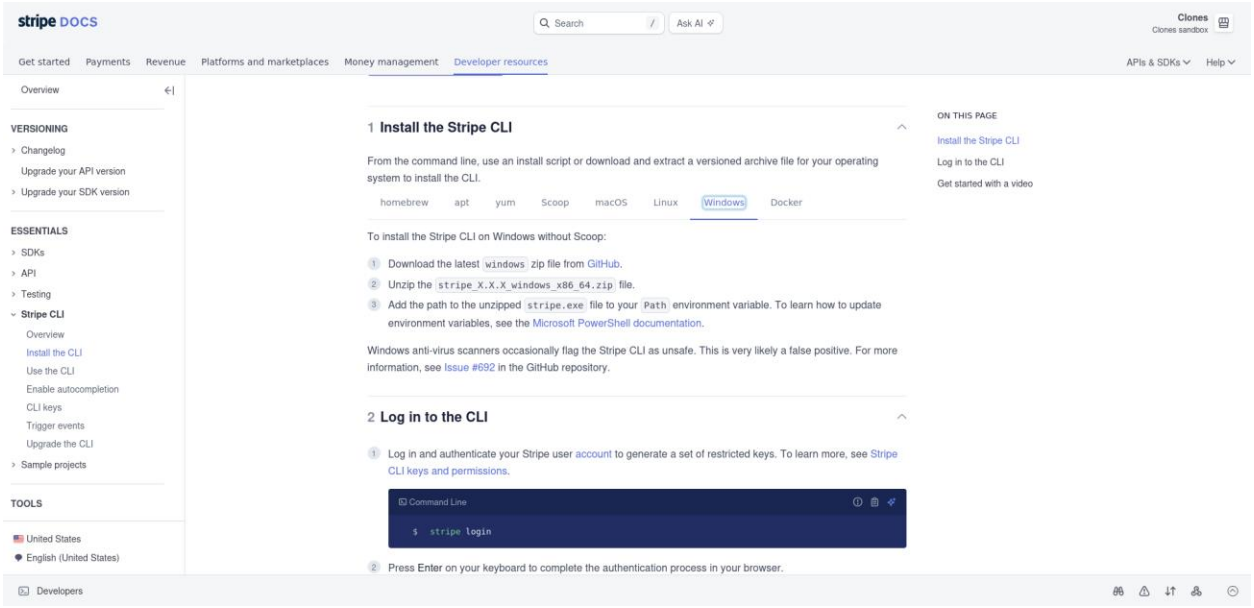


Fig.42: Go with the flow

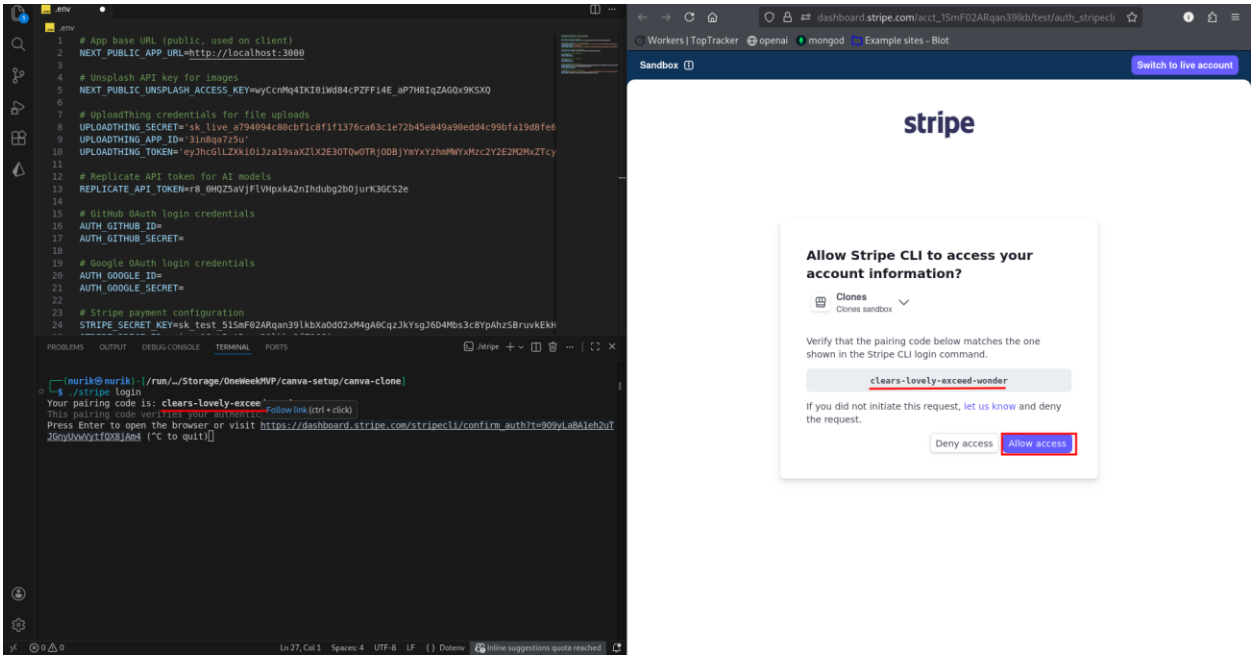


Fig.43: Allow access

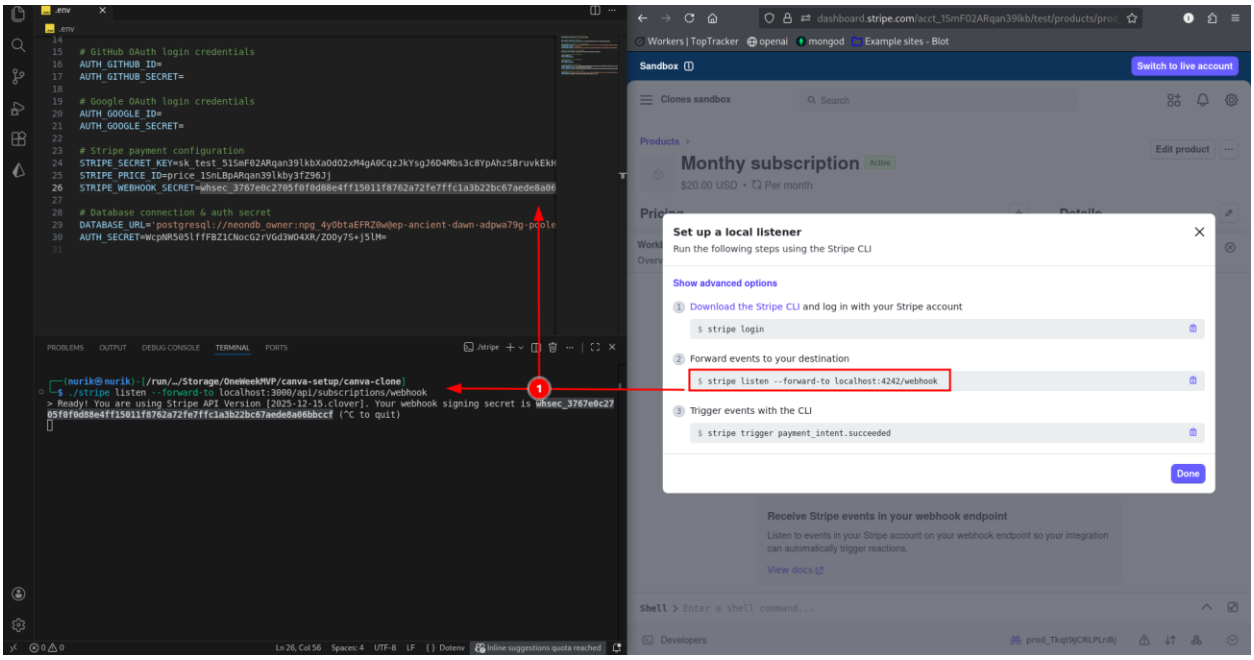


Fig.44: Stripe listen on localhost:3000

Do not close this terminal!

Open a new terminal for commands. And run:

$ bun dev

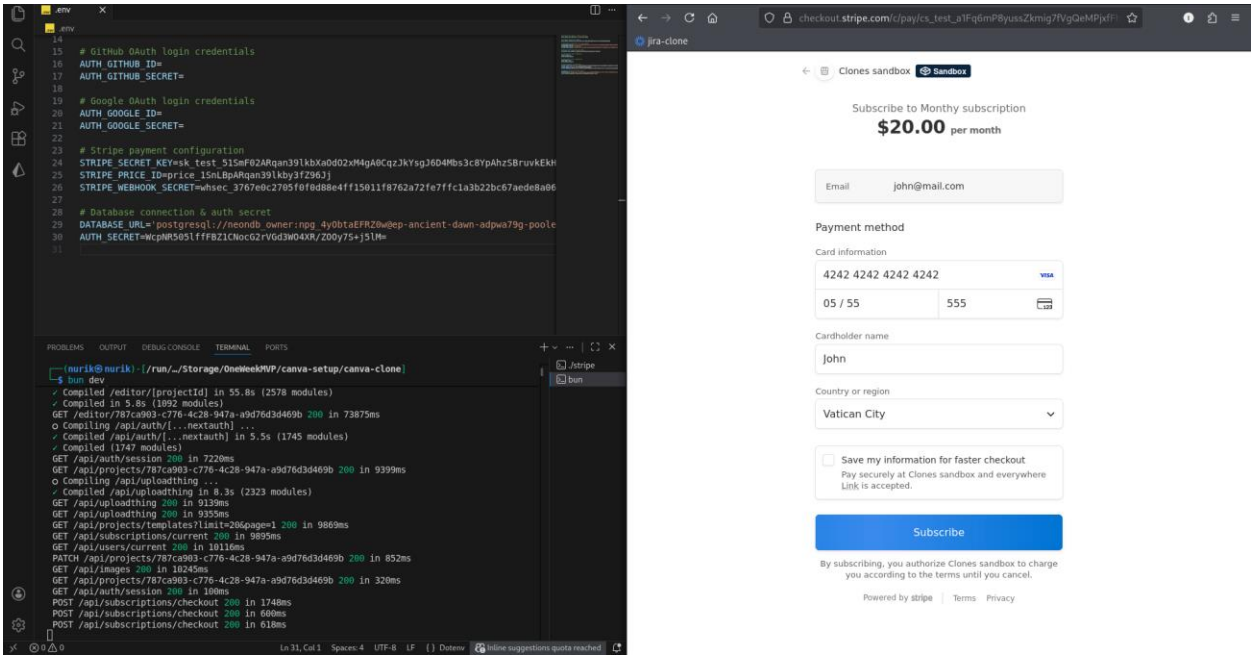


Fig.45: Dummy info && Subscribe

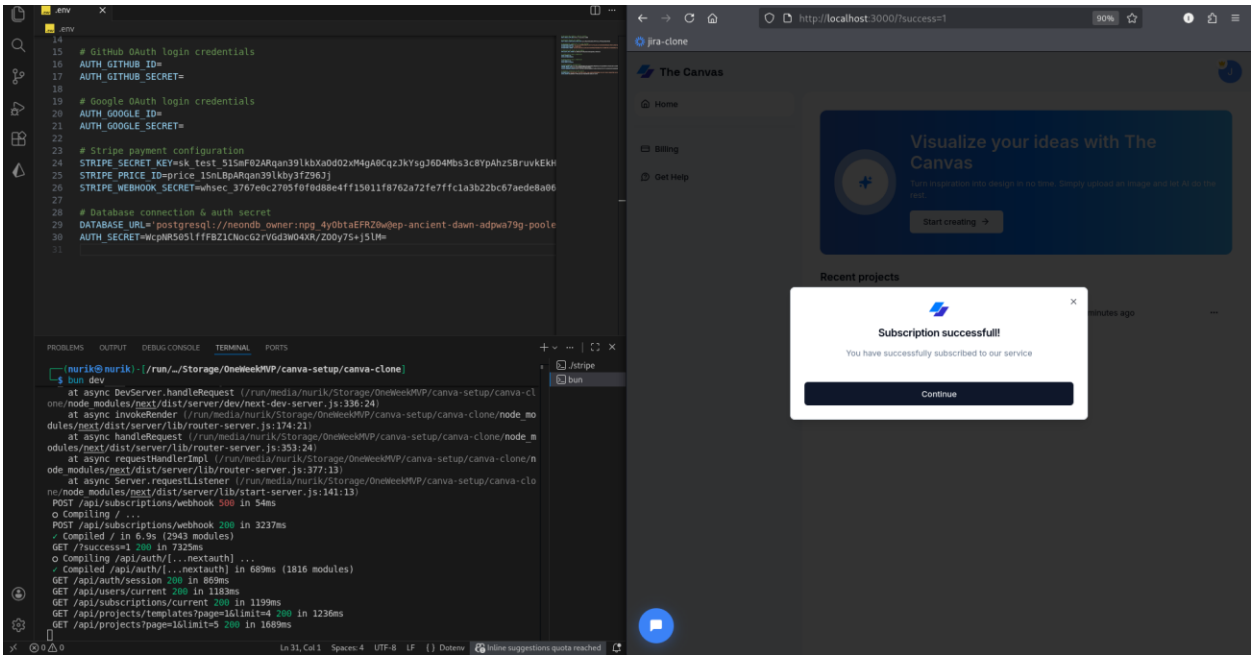


Fig.46: Successful

Remember? We were prompting an AI-generated image. It’s time now!

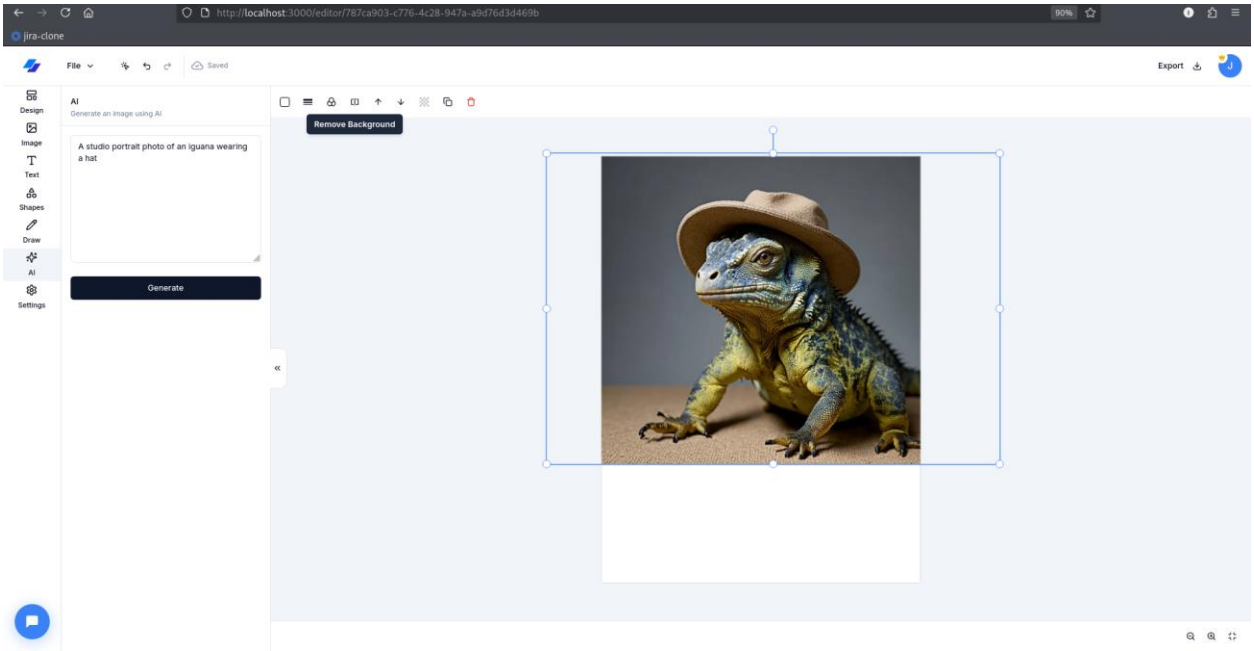


Fig.47: Image Generation

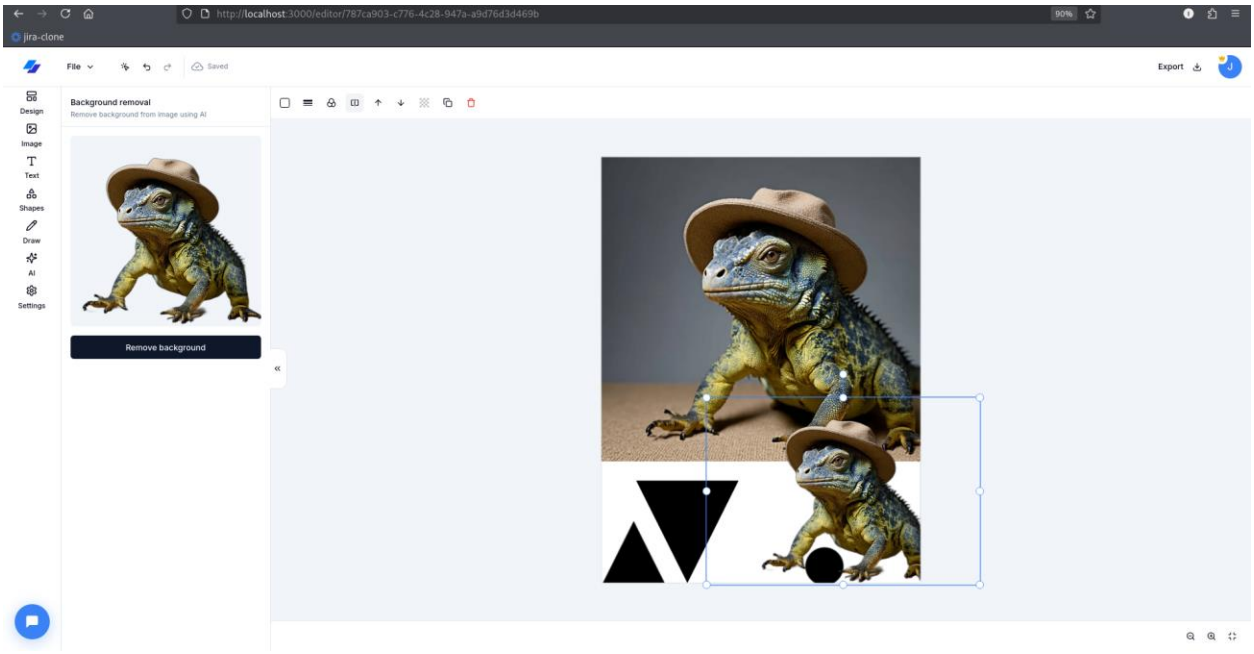


Fig.48: Background Removed

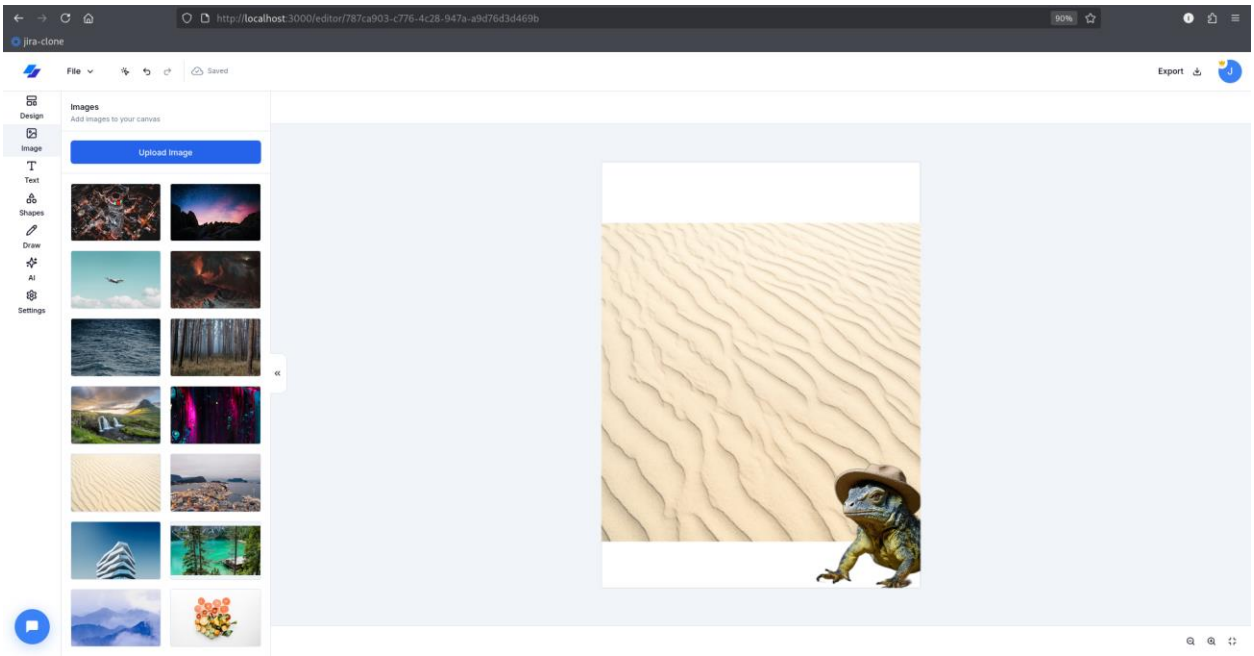


Fig.49: Go ahead and play

Now, let’s create some templates.

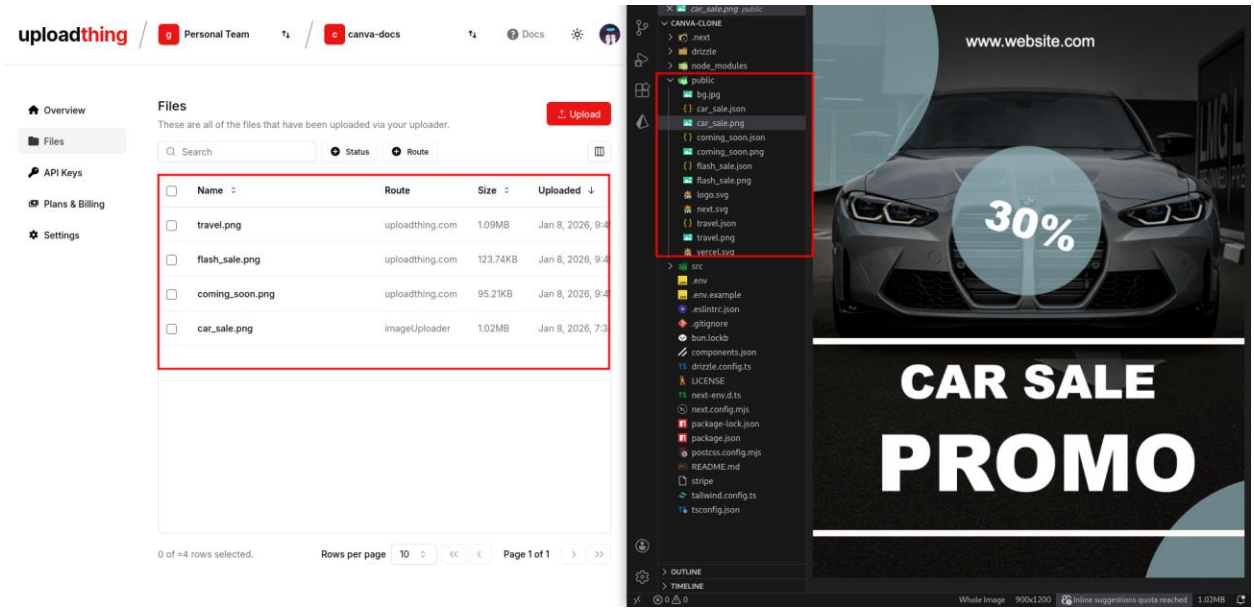


Fig.49: Upload the PNG images from the public

Upload all your images from the public folder to your upload thing. Why? We are going to use their URL for the thumbnail via Drizzle Studio later.

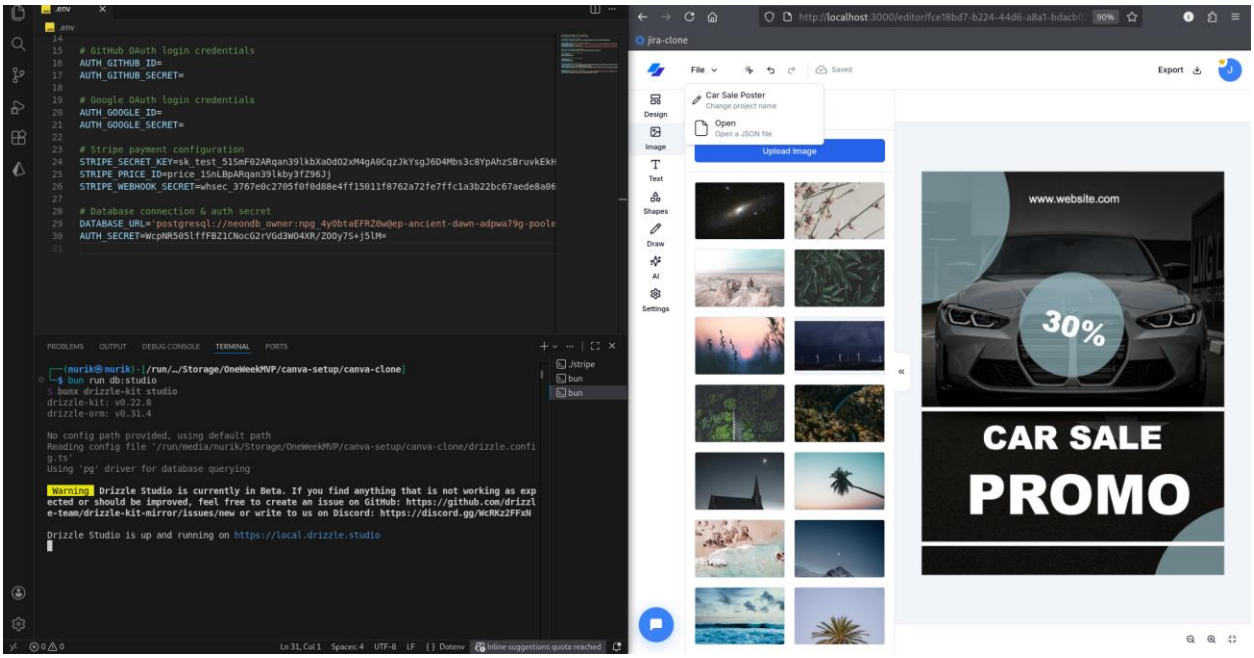


Fig: 50: Upload an image && go to your local drizzle studio

As a super admin aka, the developer who has access to the database. We need some things to modify, like:

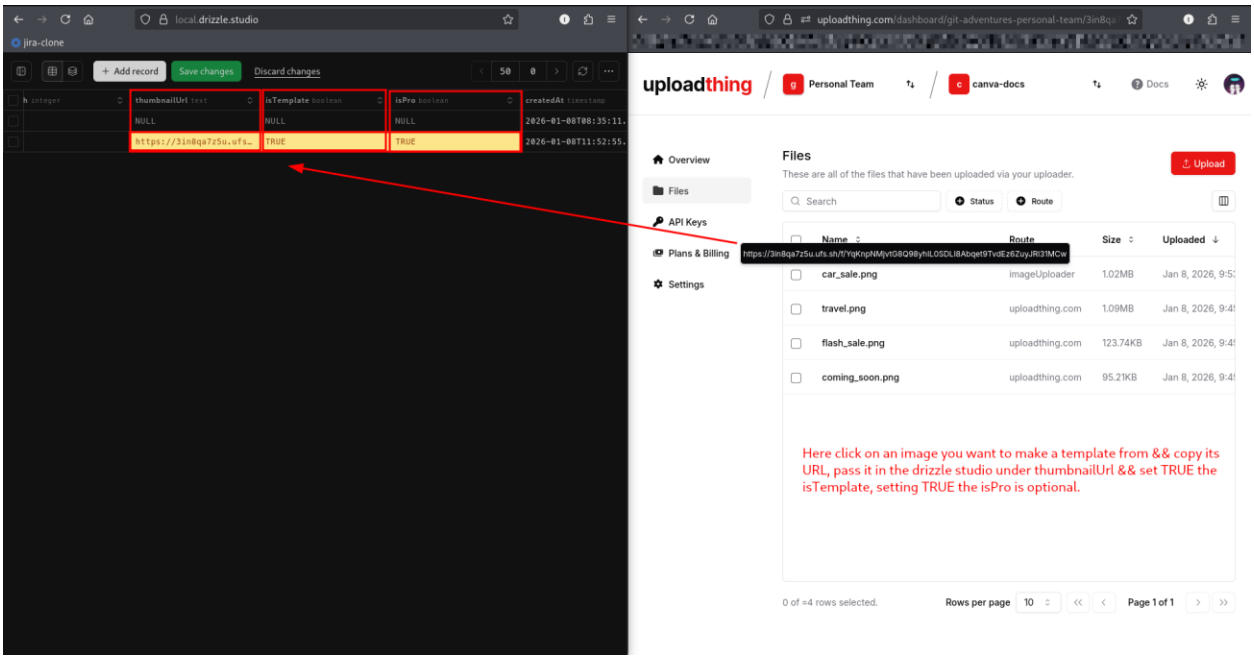


Fig. 51: Make the changes && Save

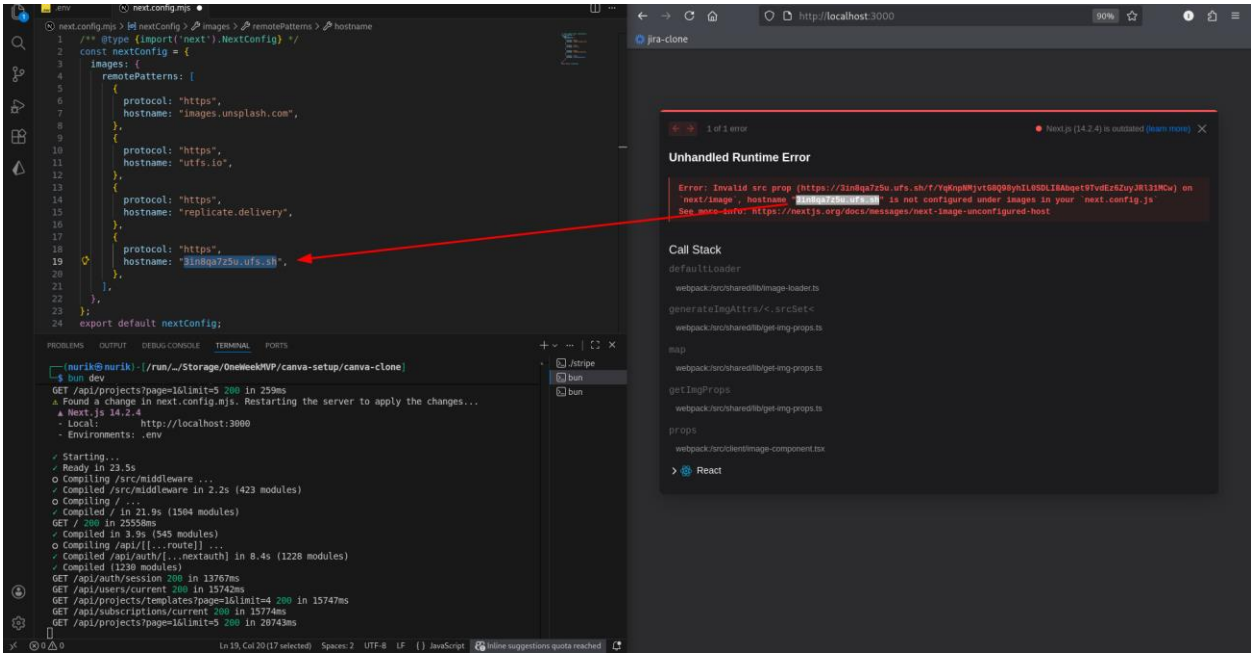


Fig.51: Error? No Problem!

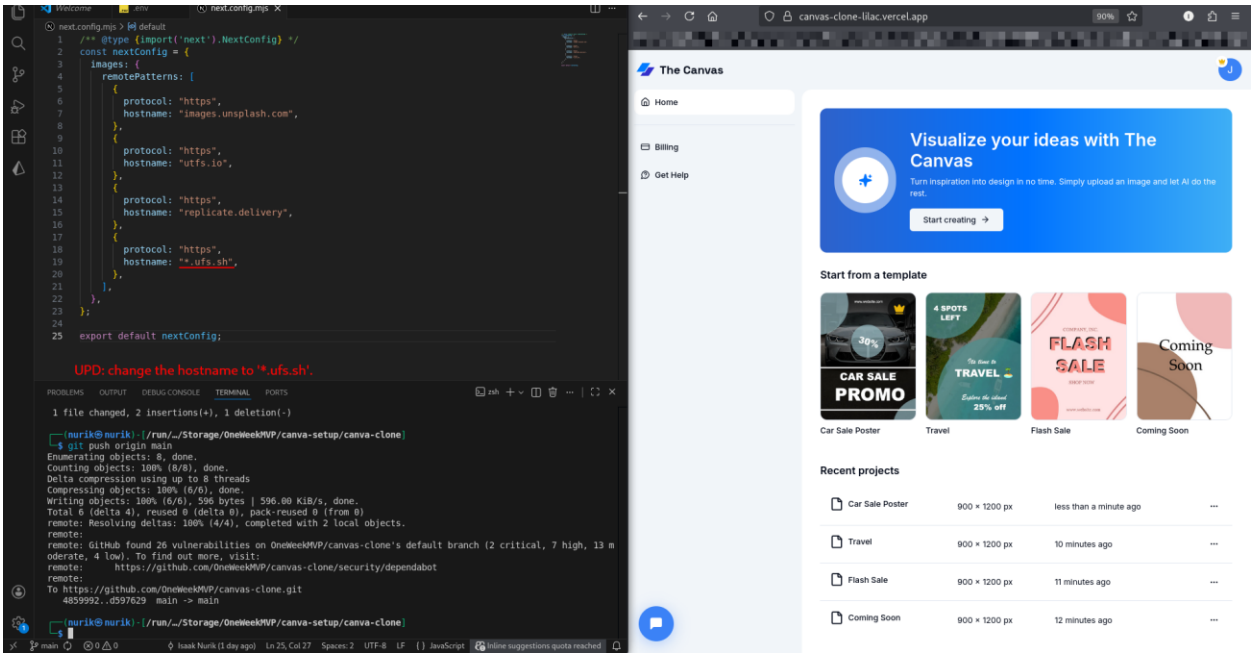


Fig.51.1: Just name the hostname to '***ufs.sh**'

UPD: The issue is that we have the specific subdomain e.g. 3in8qa7z5u.ufs.sh listed, but you need to use a wildcard pattern to match all Uploadthing subdomains since they can change.

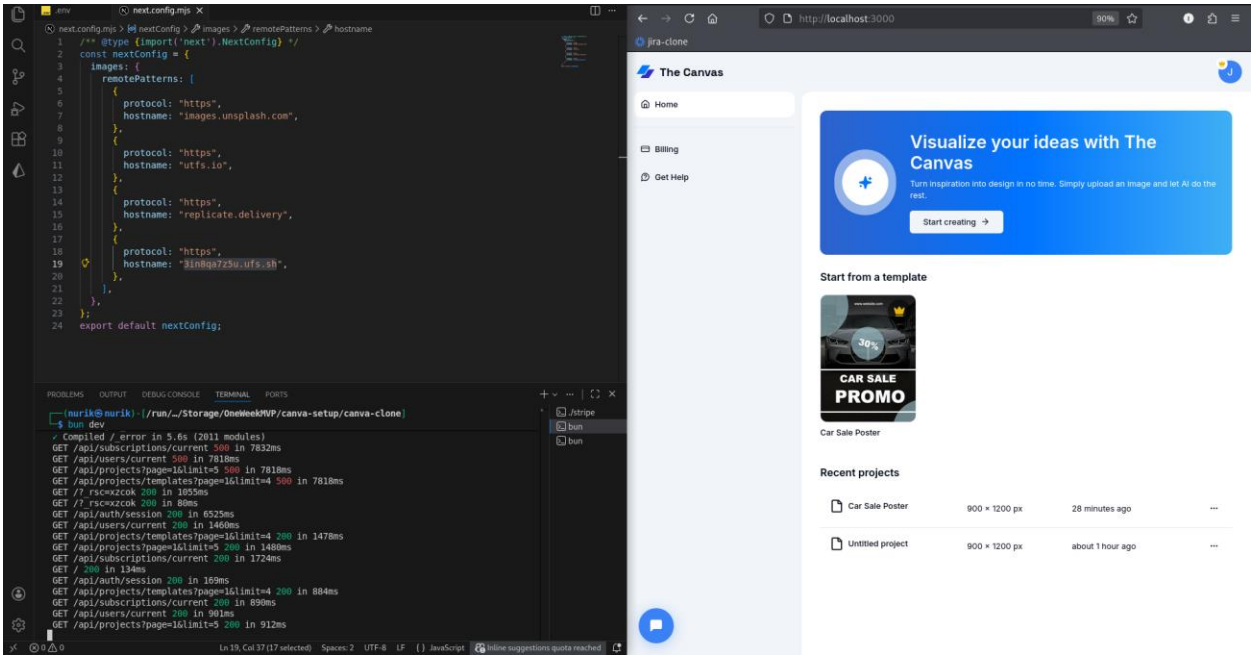


Fig.52: The template is there

Repeat the process for the other images too.

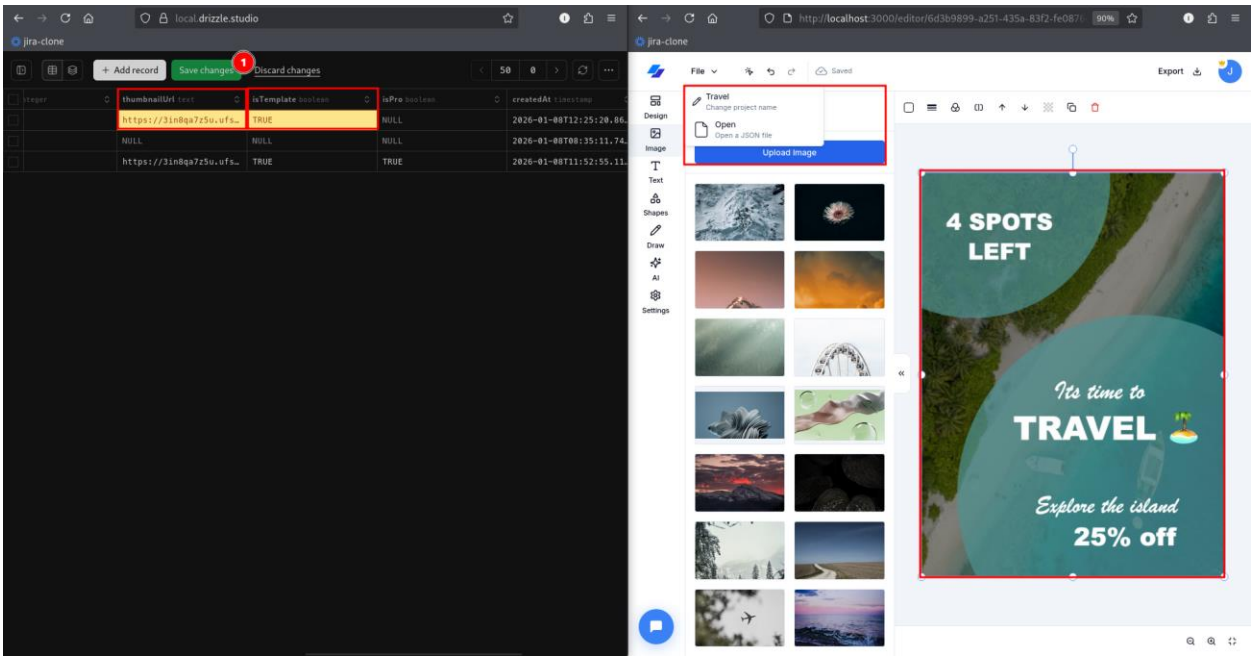


Fig.53: Repeat the process & Templates are set!

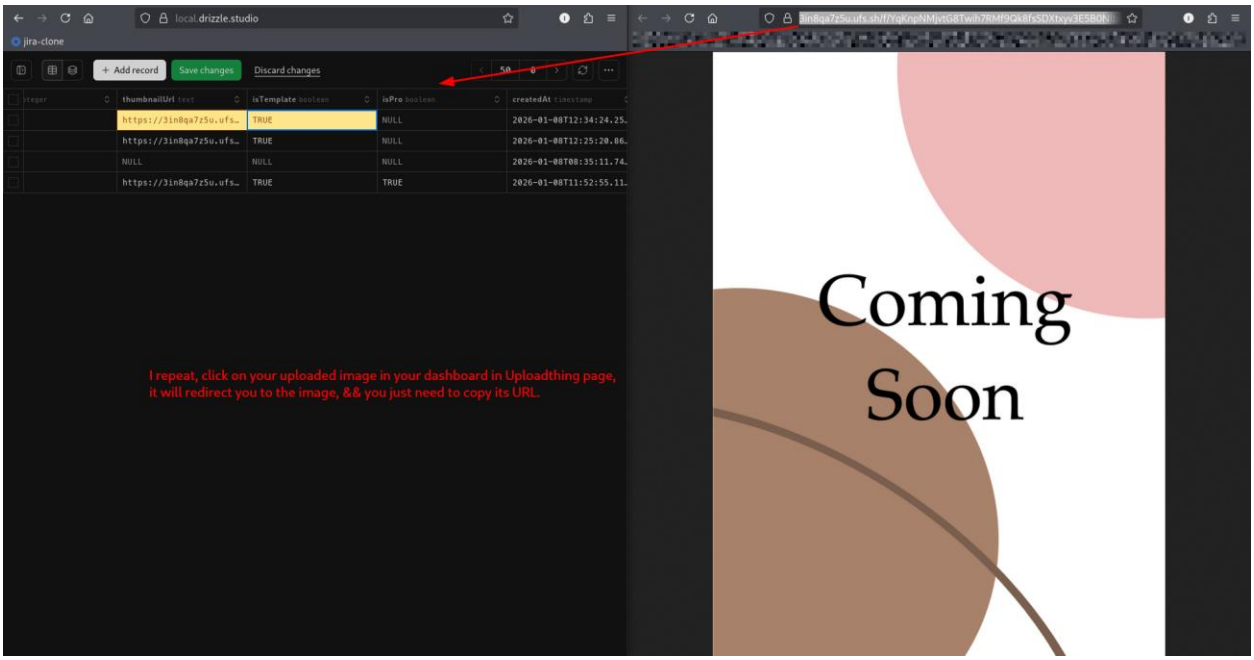


Fig.54: Copy pass the URL by pressing on the image in Uploadthing page

Now, all that's left are Google and GitHub OAuth, which is an easy setup in comparison with stripe.

Go to Github via this link: https://github.com/settings/developers

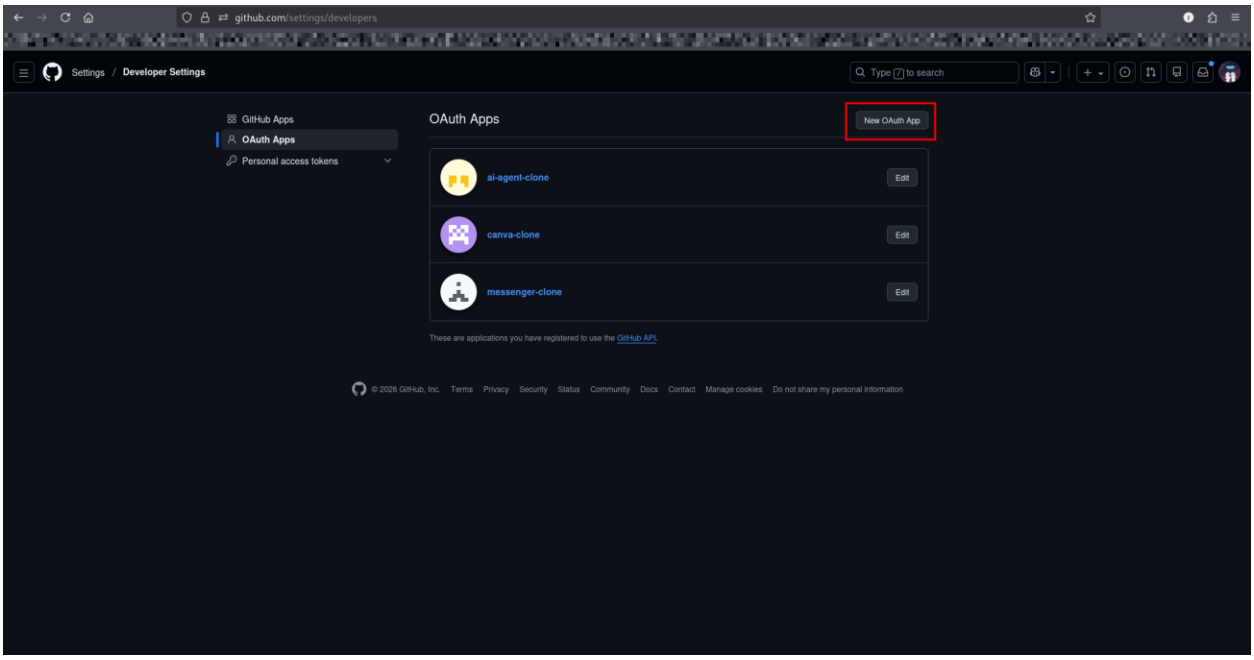


Fig.55: Go to your Github && create a new OAuth App

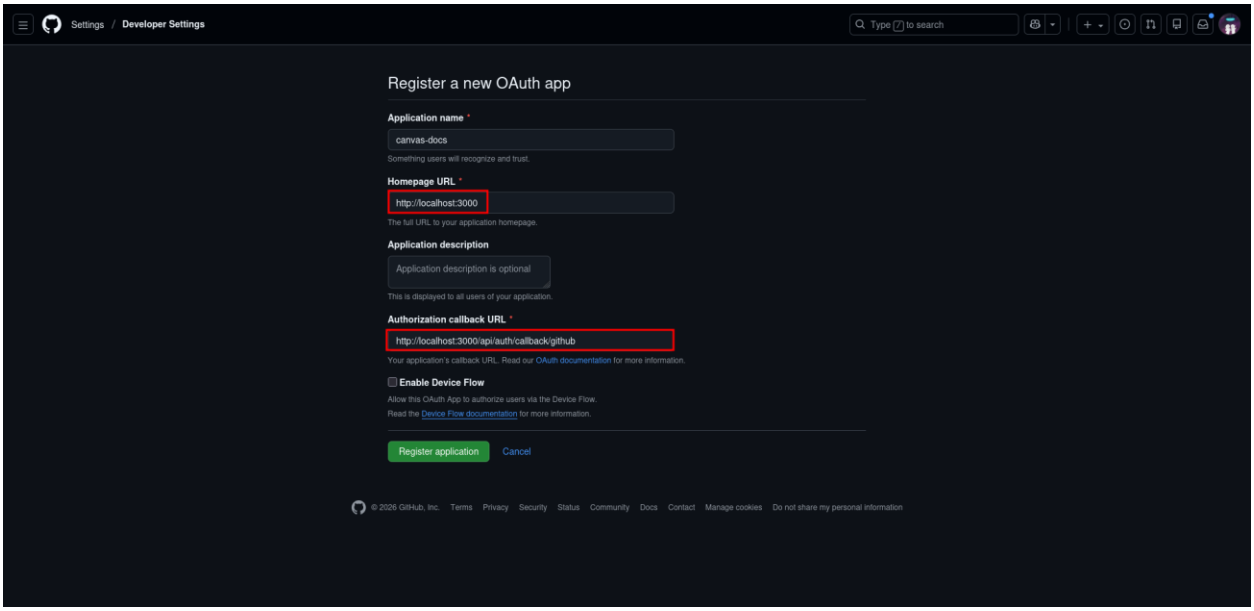


Fig.56: Pass the URL as shown

Homepage URL:

http://localhost:3000

Authorization callback URL

http://localhost:3000/api/auth/callback/github

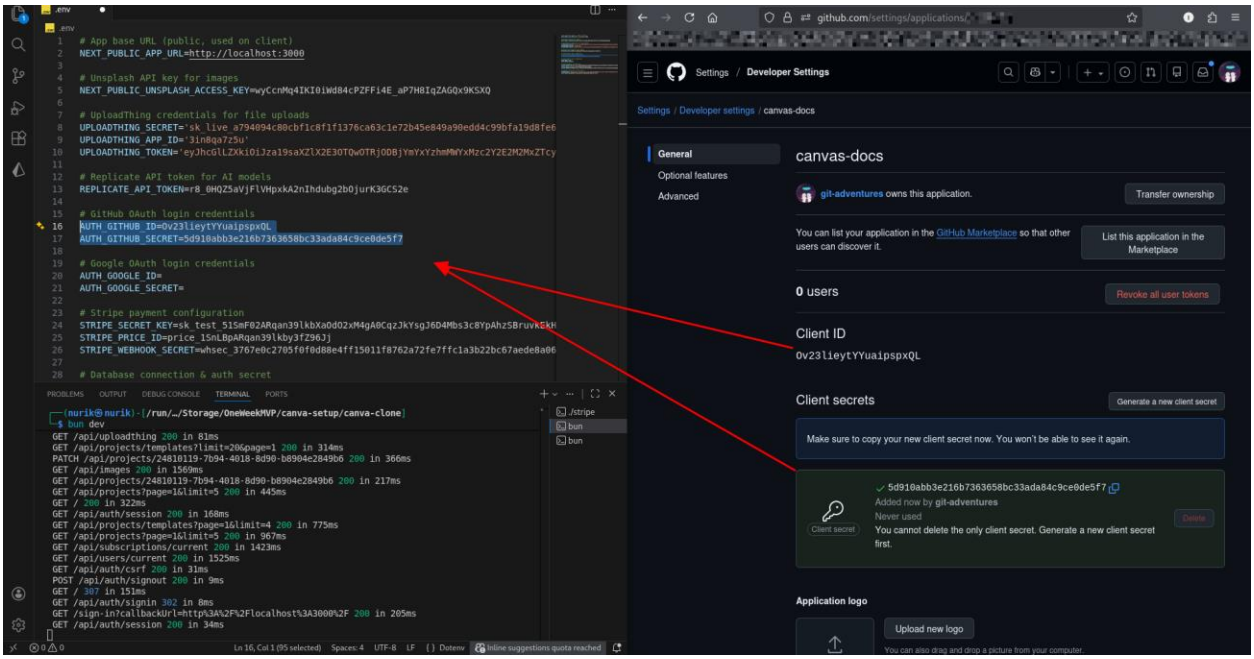


Fig.57: Github is set for localhost:3000

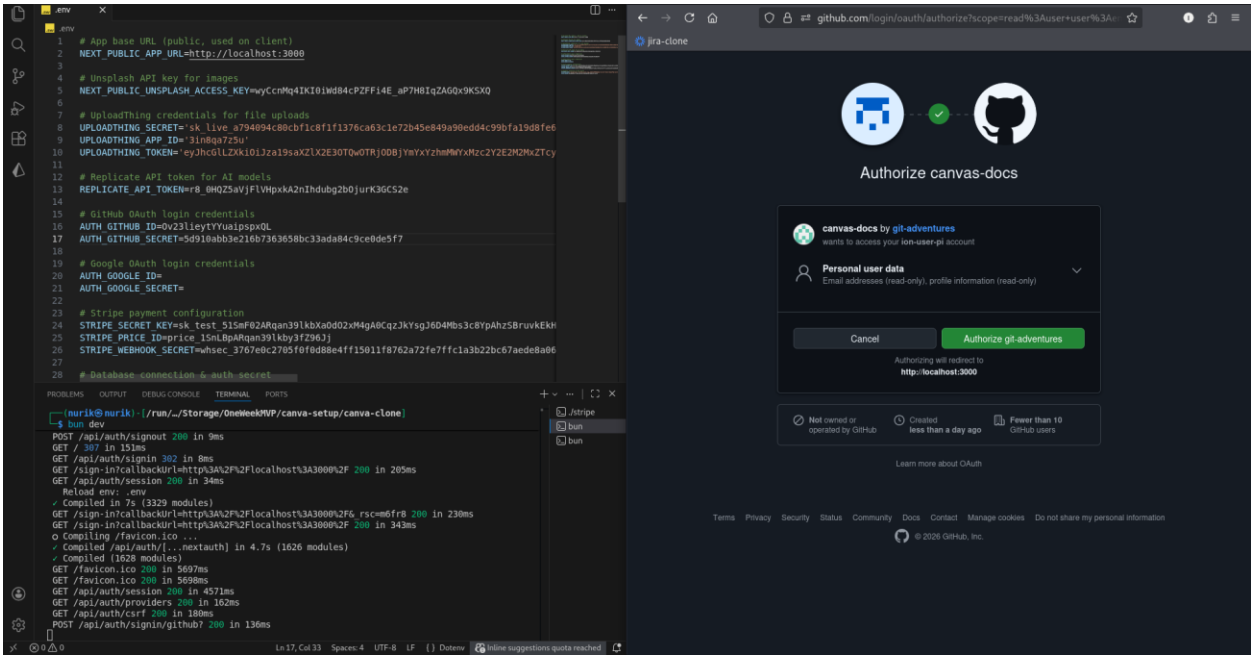


Fig.58: Github works!

Now, let's set up Google. Head to https://console.cloud.google.com

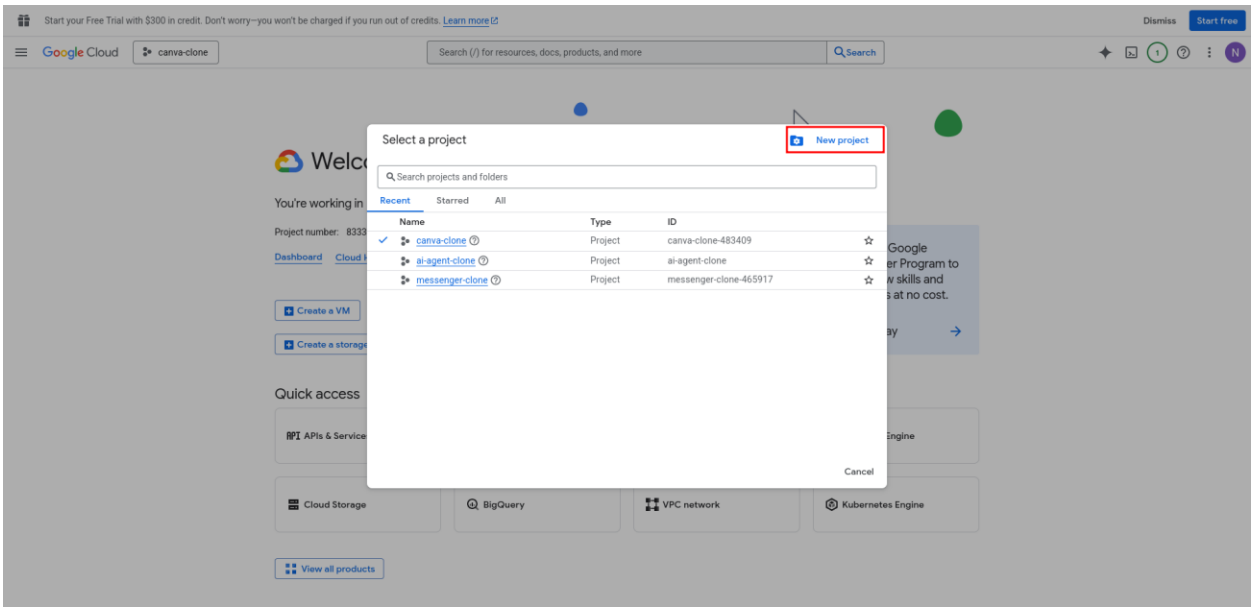


Fig. 59: Project Creation

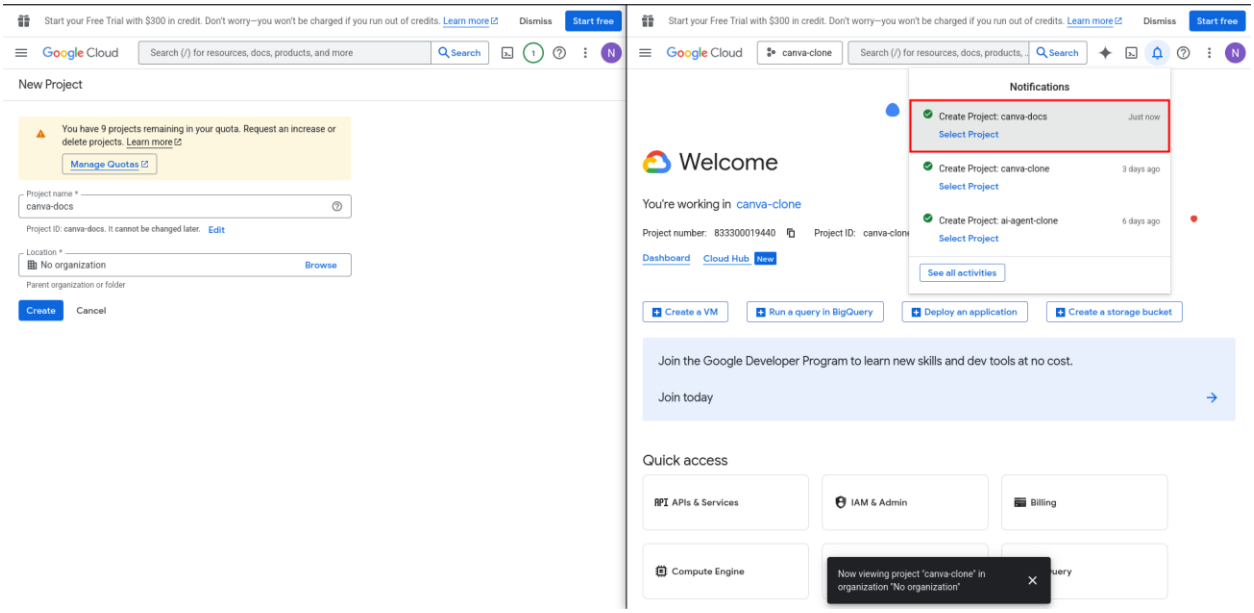


Fig. 60: Name your project && Select

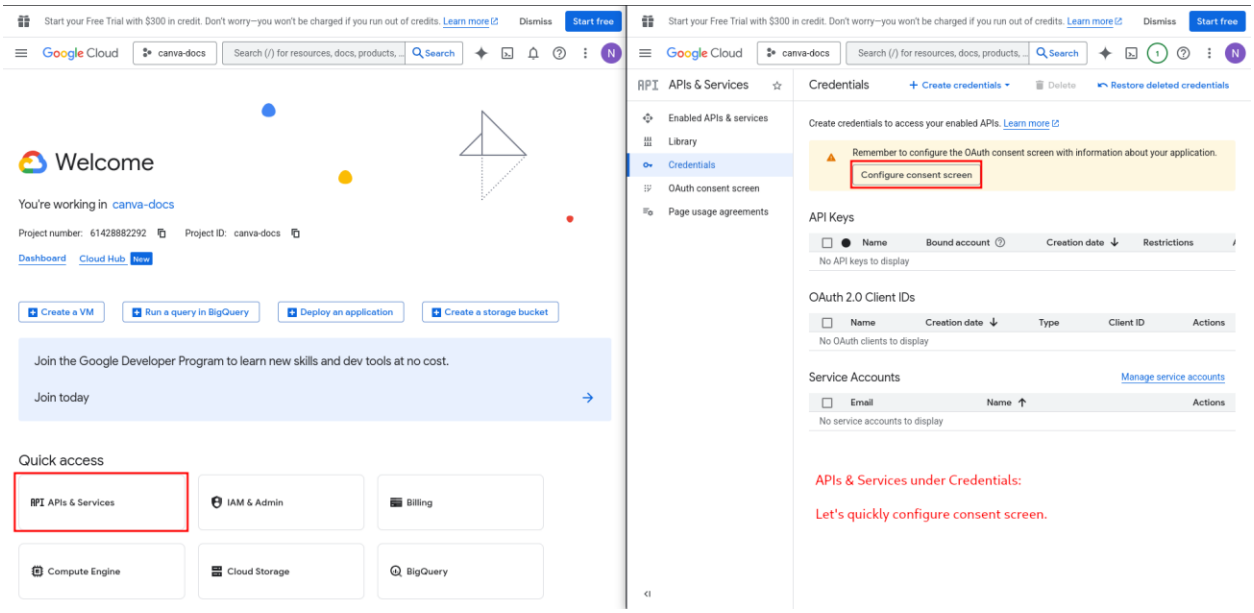


Fig. 61: Head to APIs && Configure

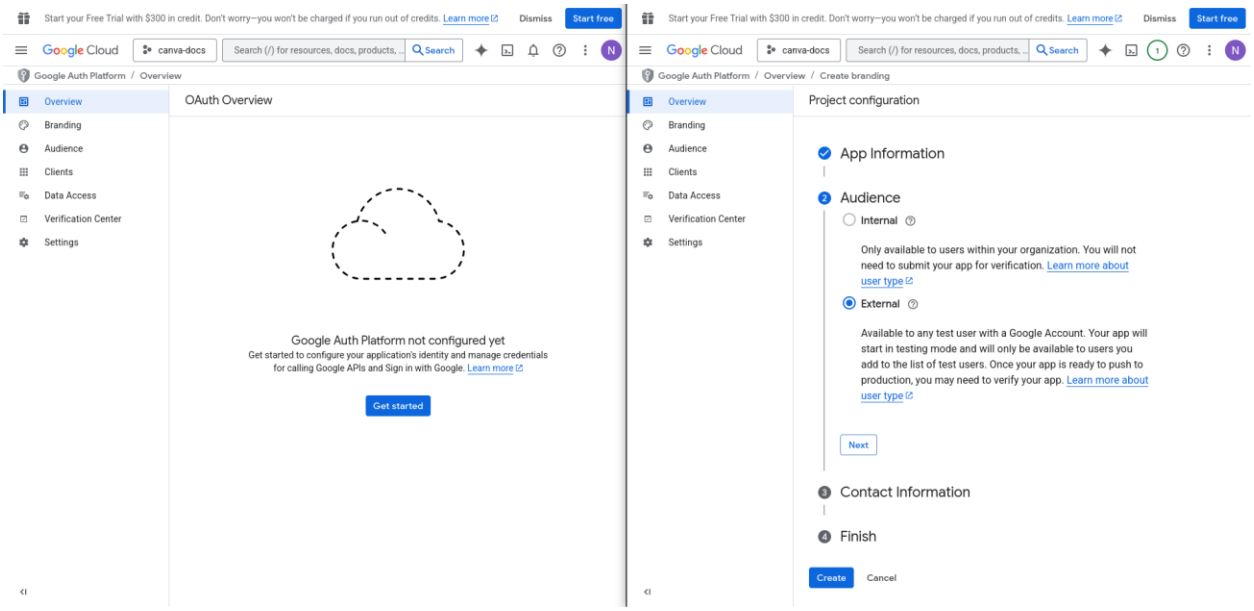


Fig.62: Finish your Google Auth Configuration

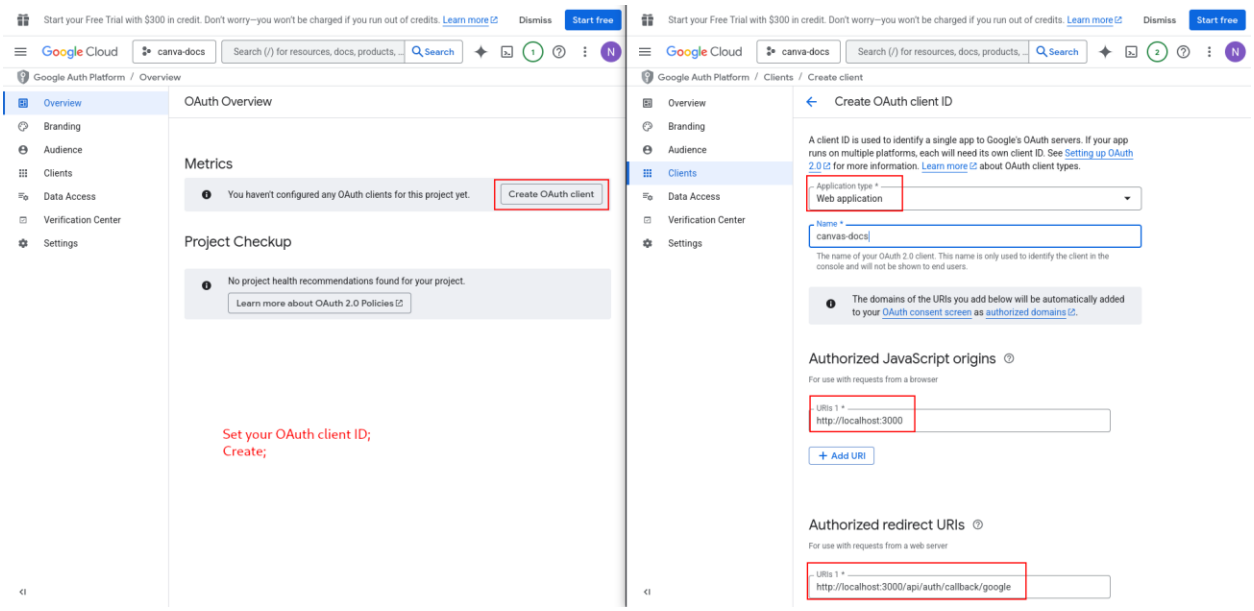


Fig.63: Create OAuth client ID

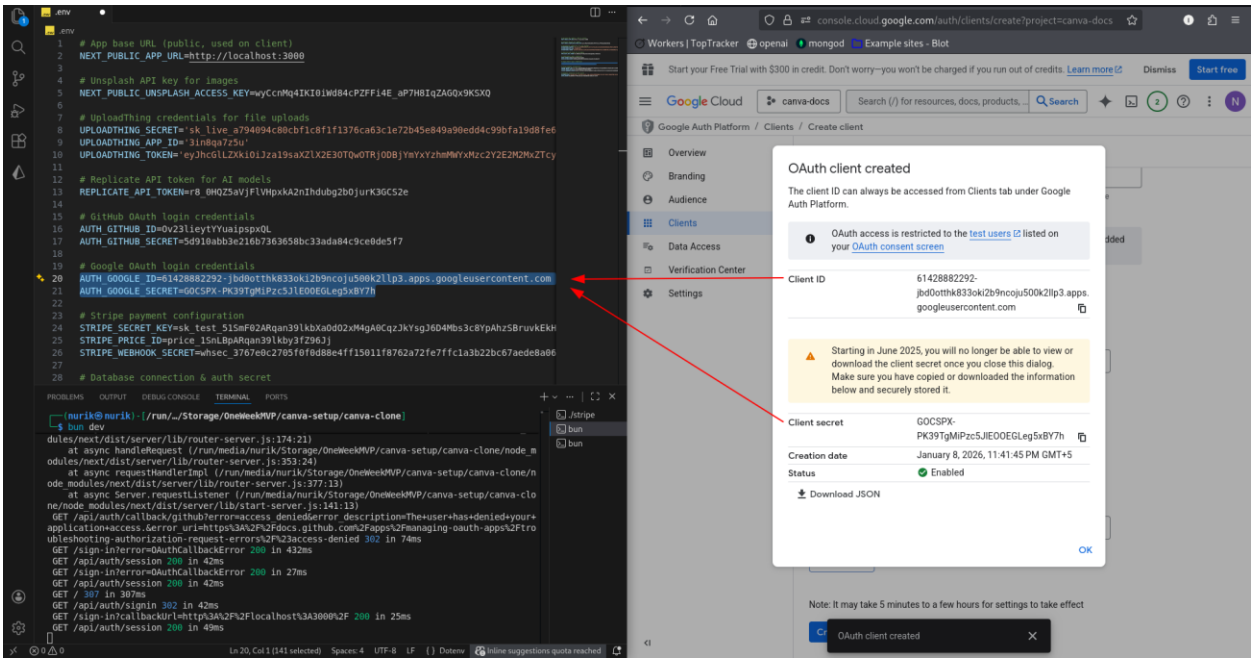


Fig.64: Client ID && Secret

Copy your Client ID and Secret and paste them into your environment file.

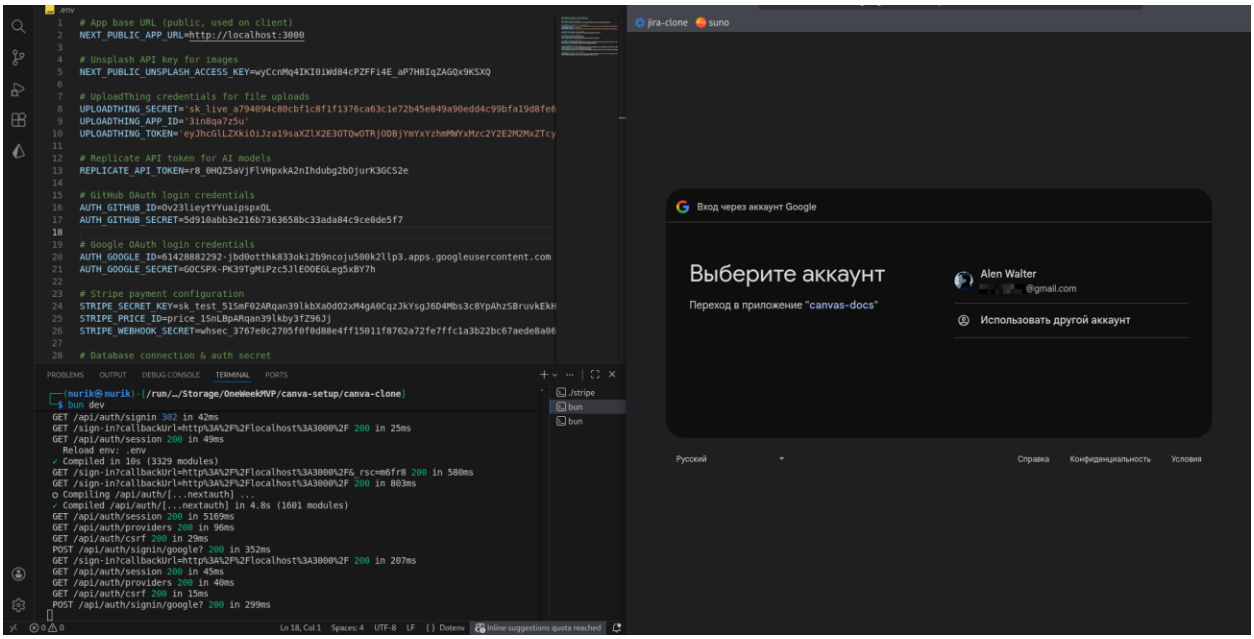


Fig.64: There it is!

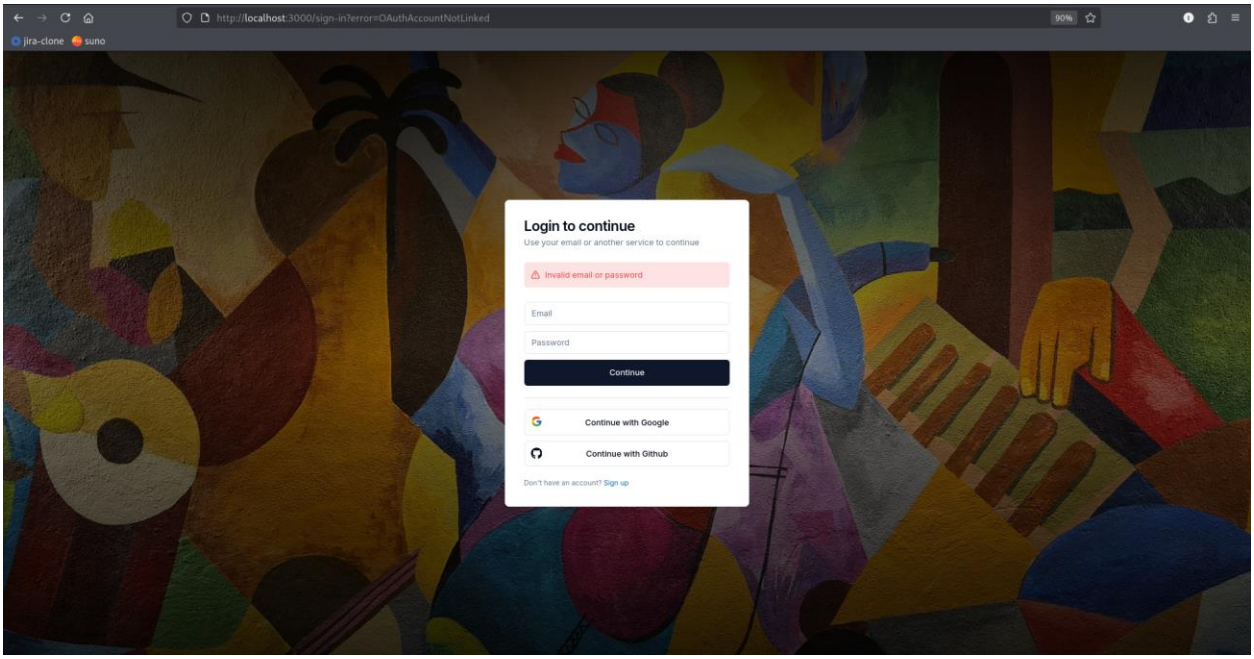


Fig.65: You can’t log in with the same account from different OAuths

Finally, let’s deploy.

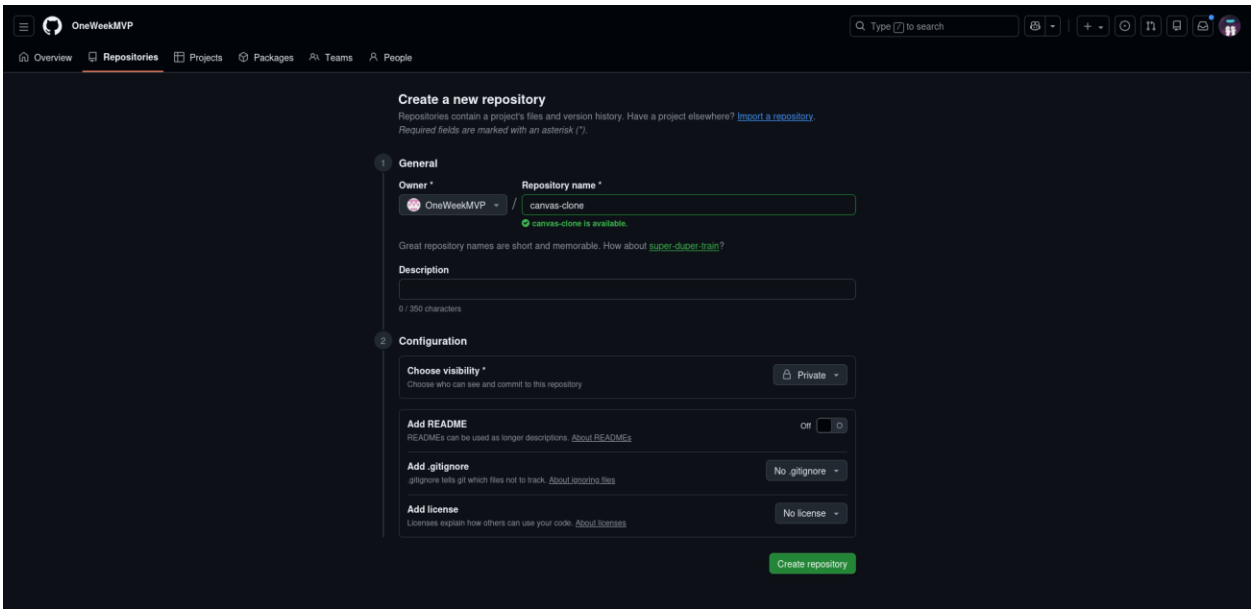


Fig.66: Create a repository

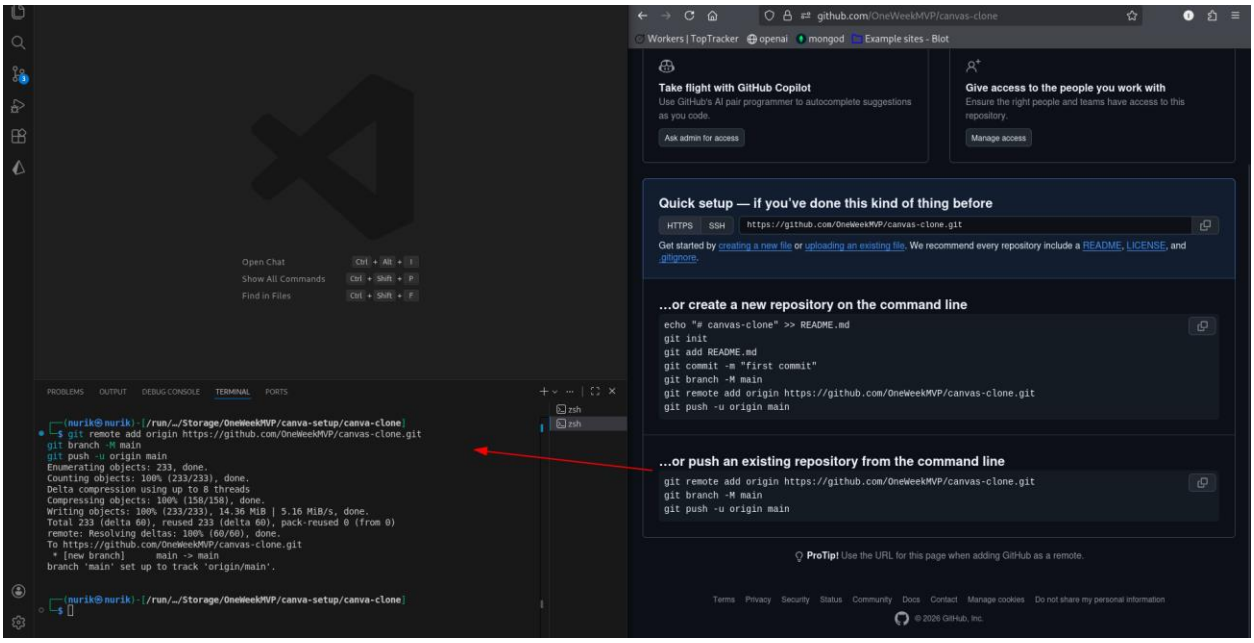


Fig.67: Git push

Refresh your Github to see your repository inside.

Now, head to the Vercel: https://vercel.com

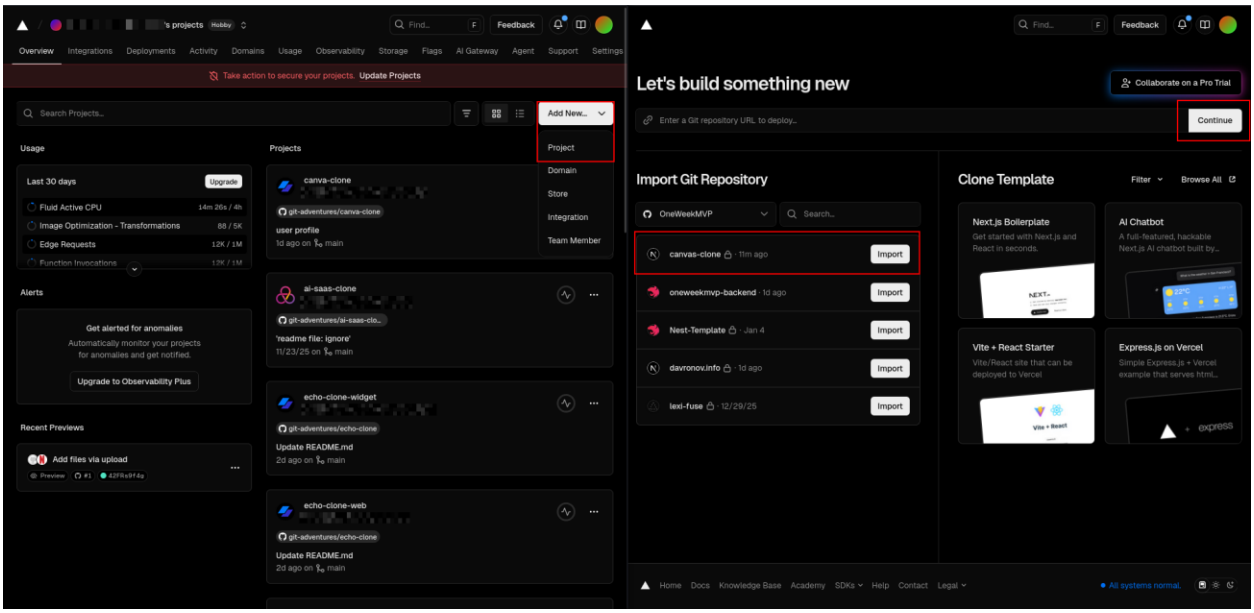


Fig.68: Import your repository from Gihub

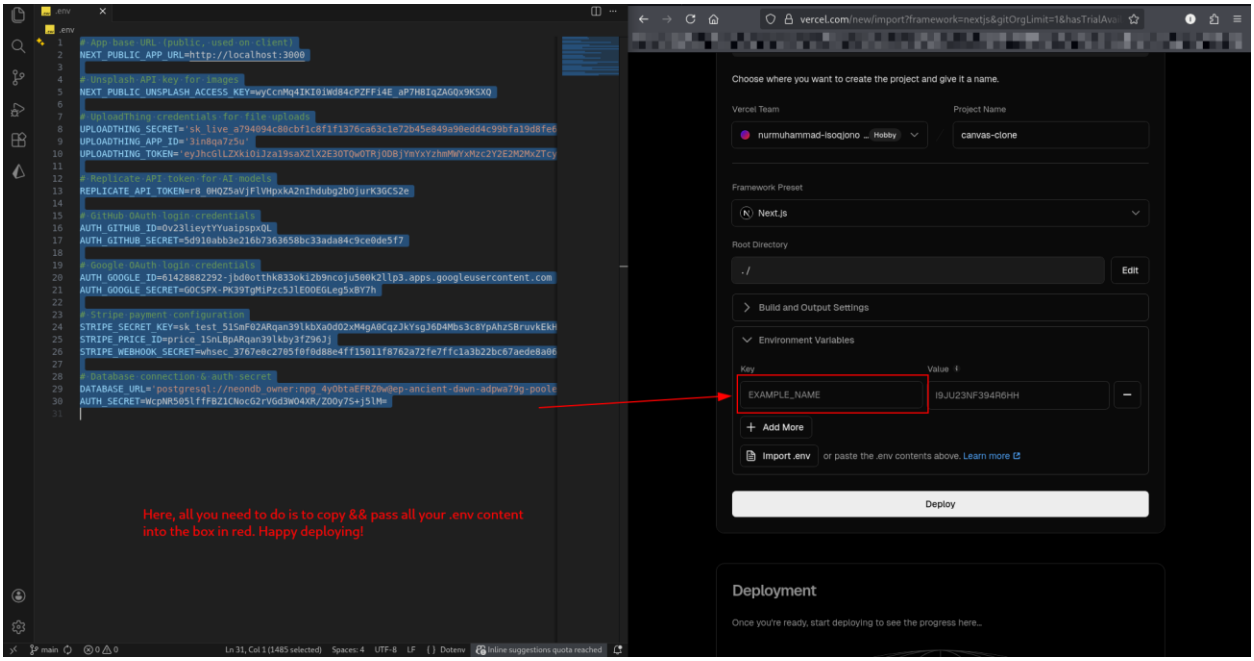


Fig.69: Happy Deploying!

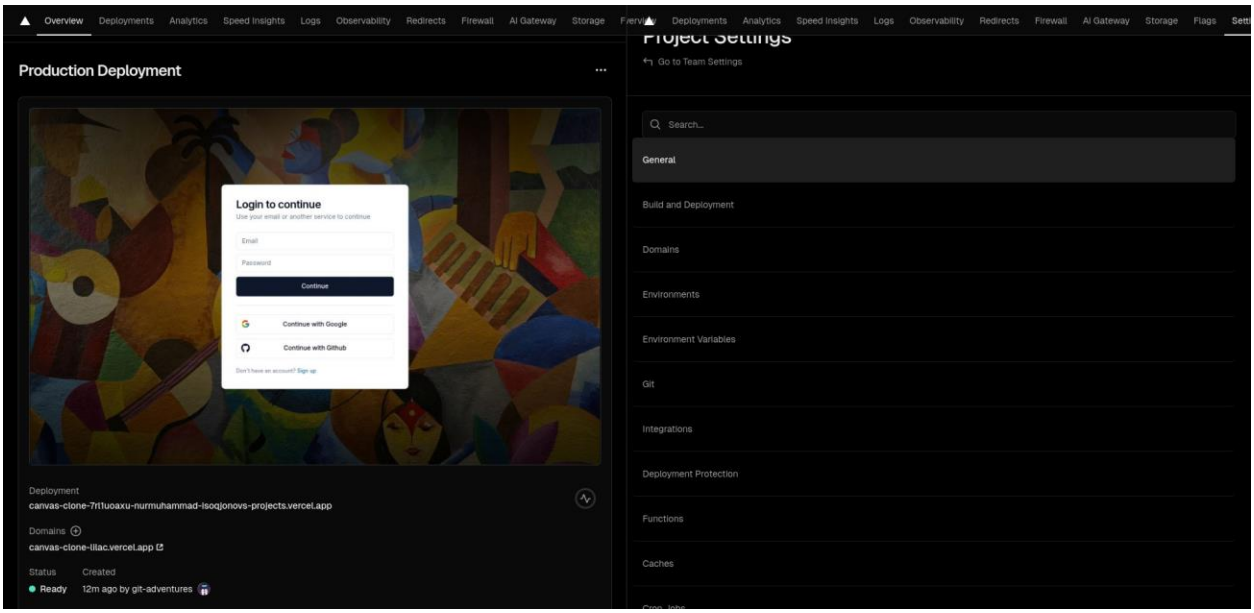


Fig.70: Copy your Domain name (URL) && got to Settings Environment Variables

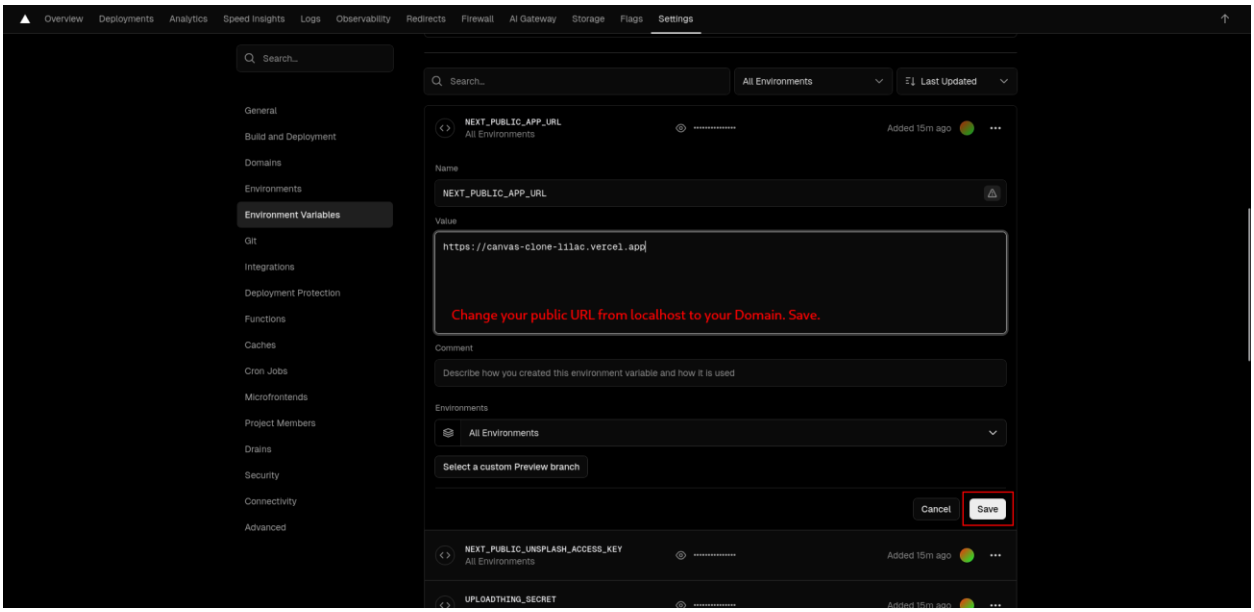


Fig.71: Save

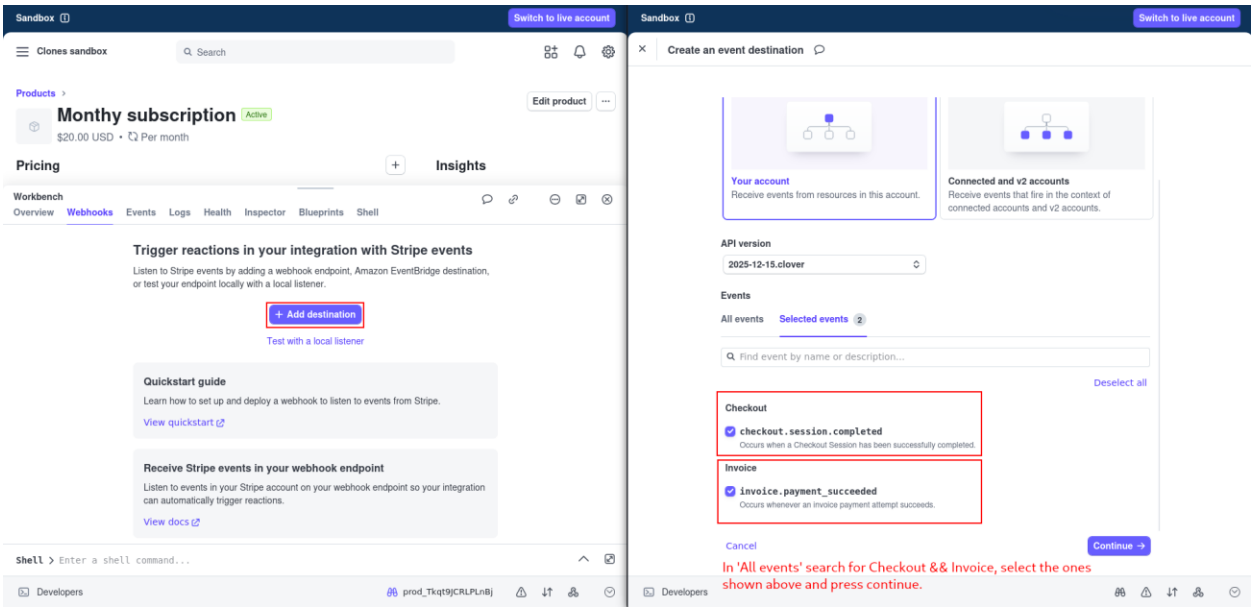


Fig.72: Stripe Events

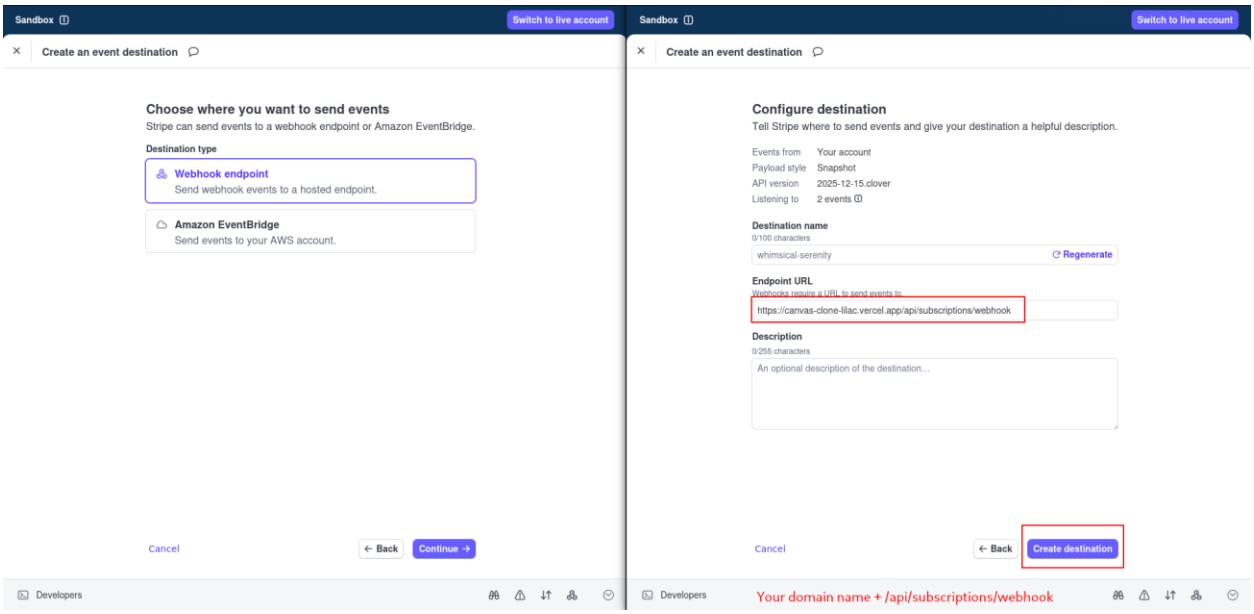


Fig.73: Webhook Destination

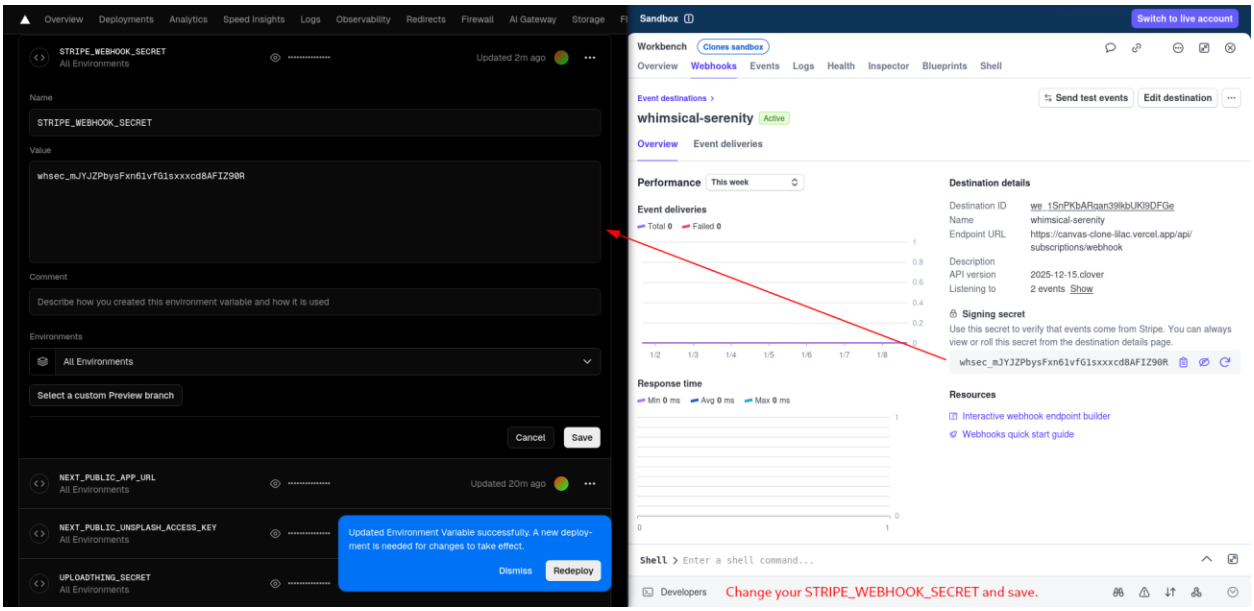


Fig. 74: Save

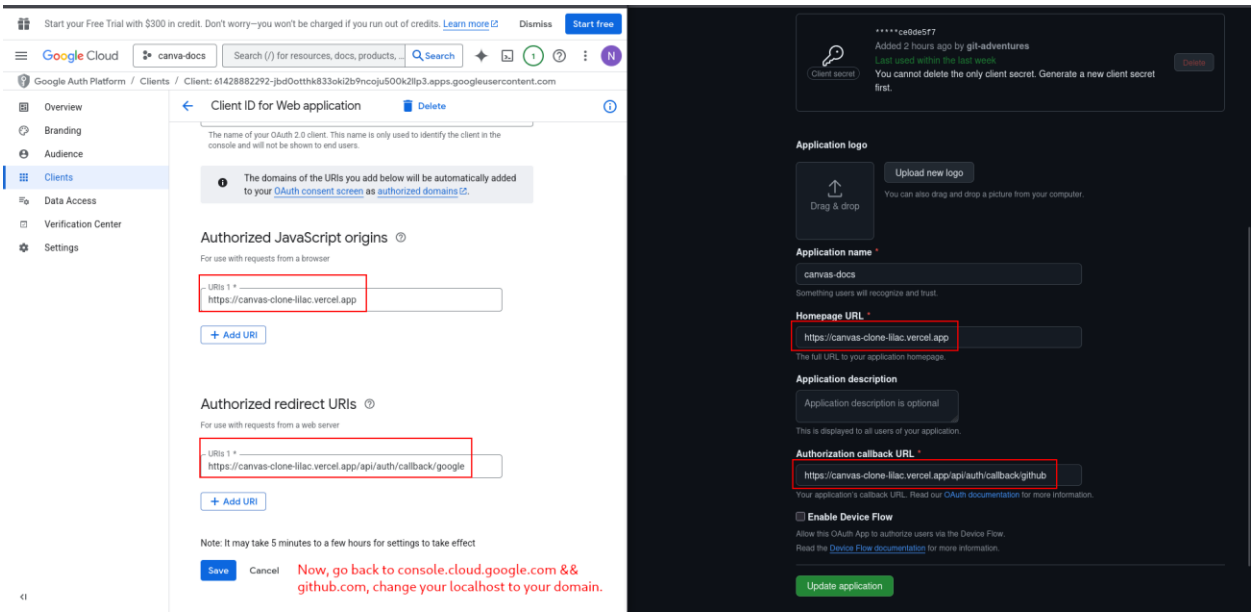


Fig. 75: Change the localhost to your domain name

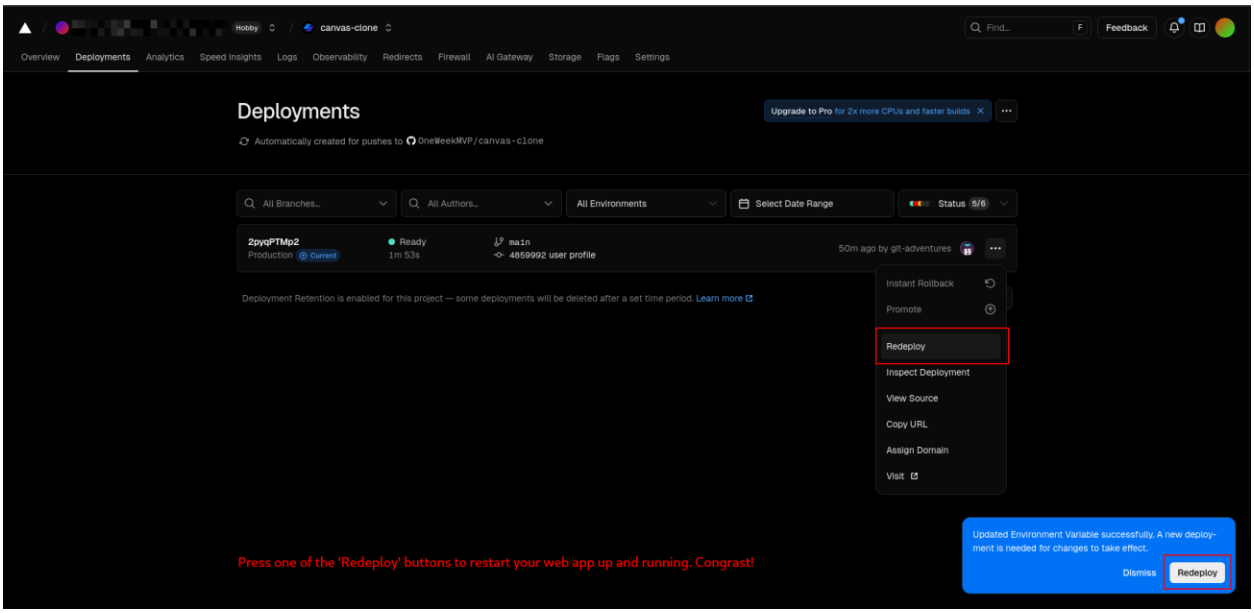


Fig.76: Redeploy

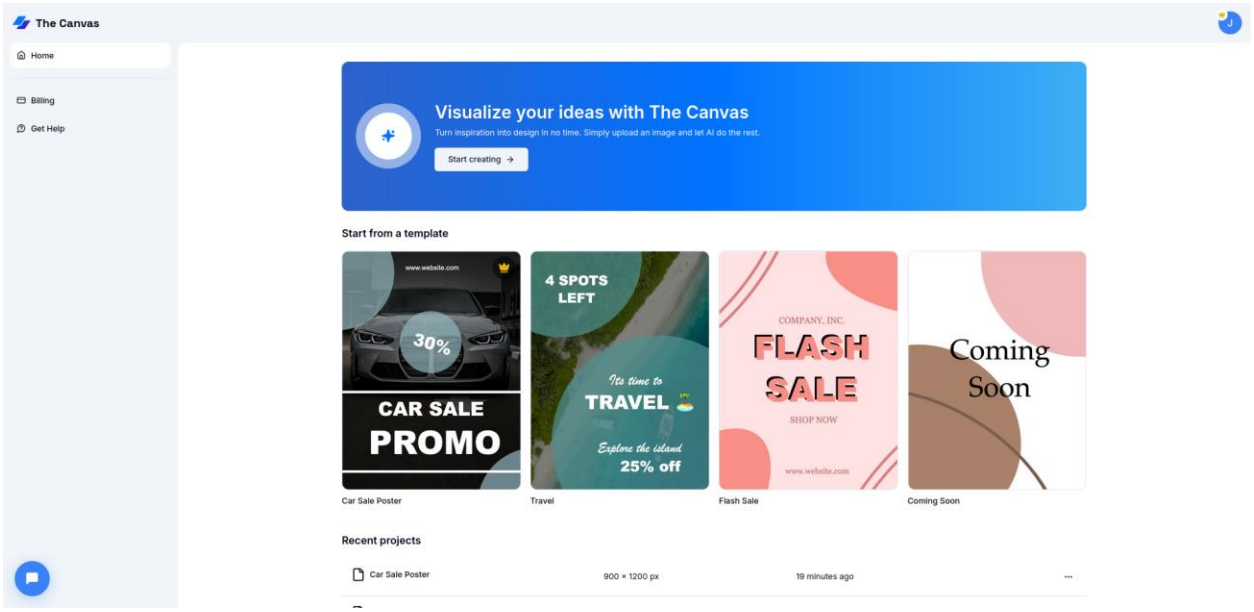


Fig.77: Canva Clone

Congratulations! □ Your Canva Clone is now fully configured and deployed!

Reference:

Video tutorial: https://www.codewithantonio.com/projects/canva-clone