

# 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:

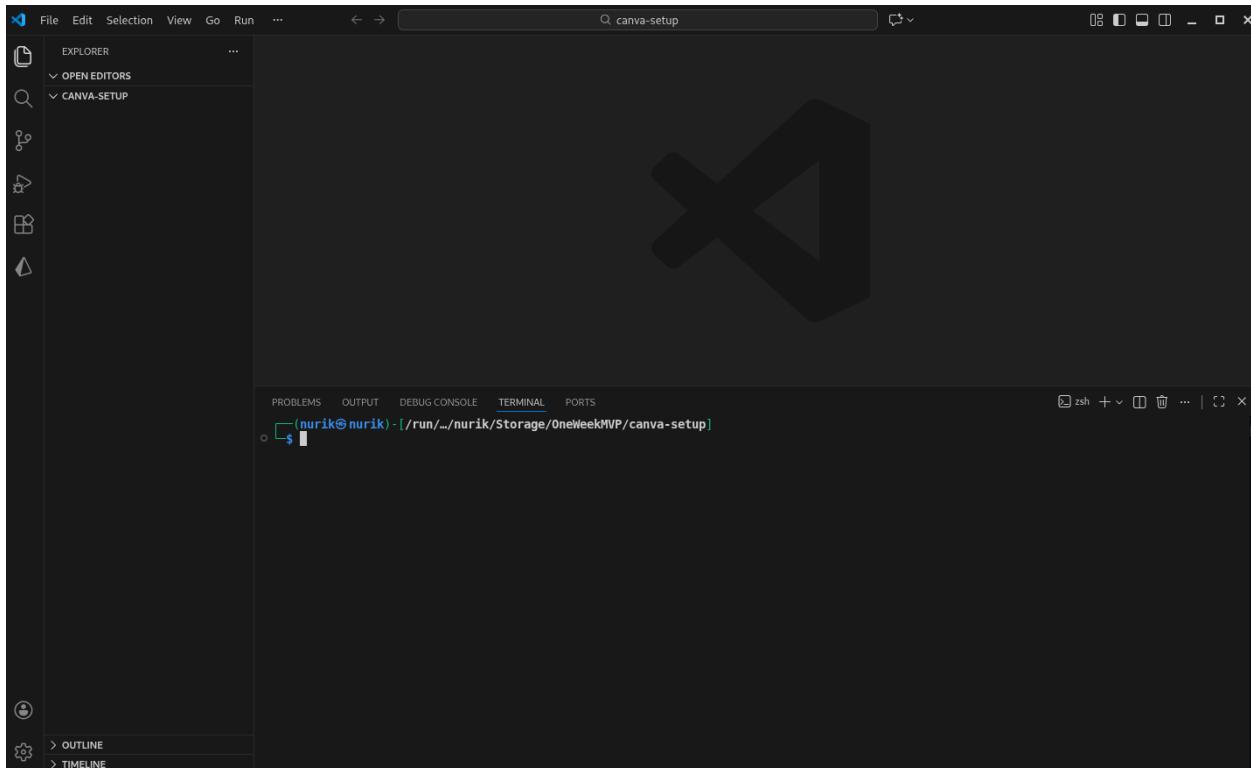


Fig.1: Create a project folder.

### 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

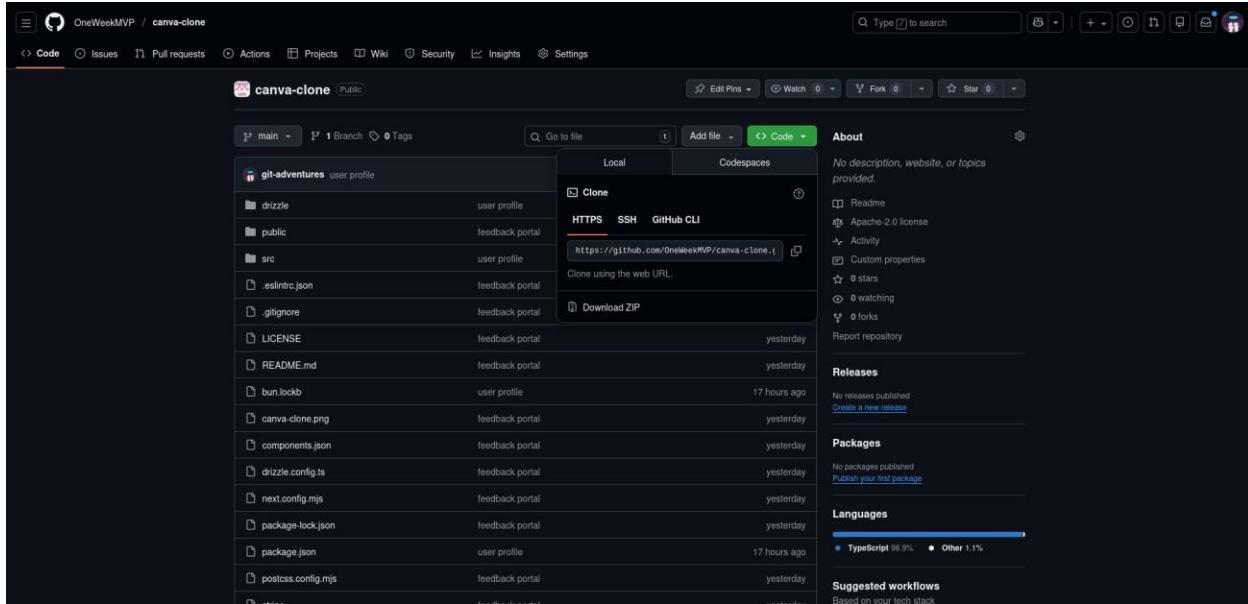


Fig.2: Grab the HTTPS URL.

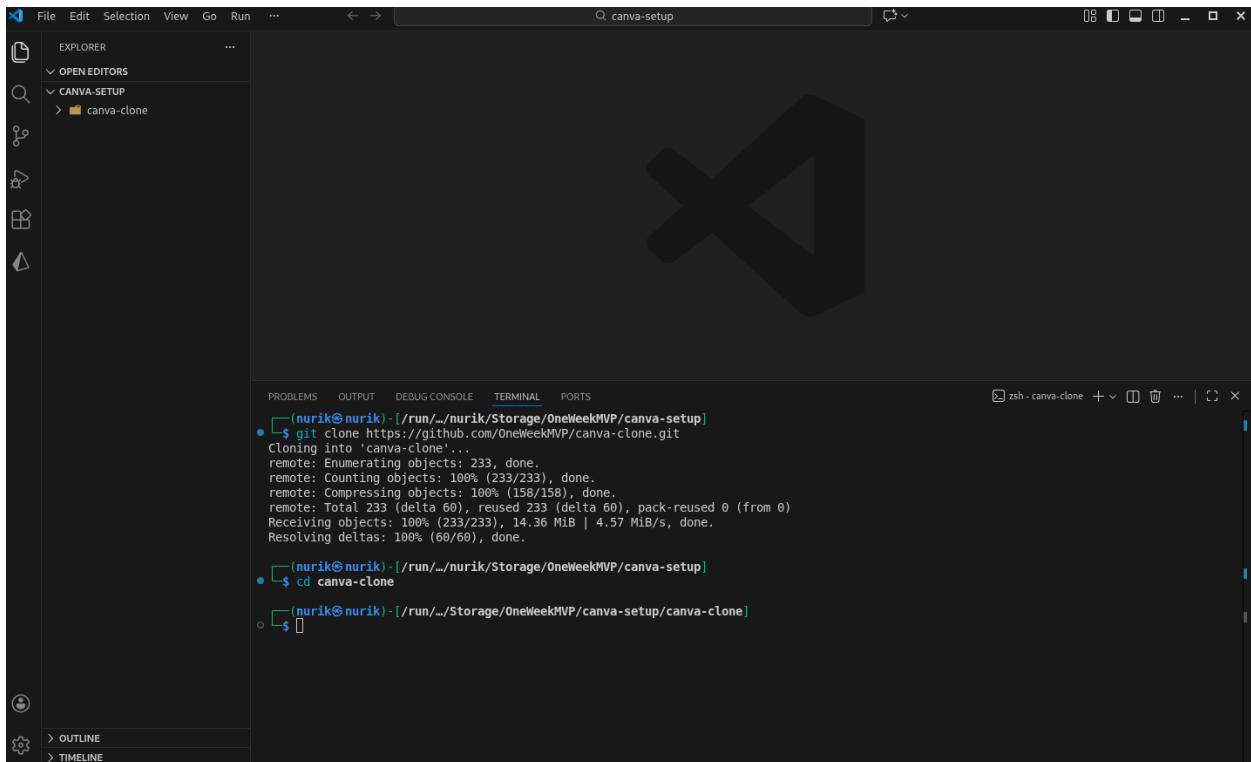


Fig.3: Clone and change the directory.

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

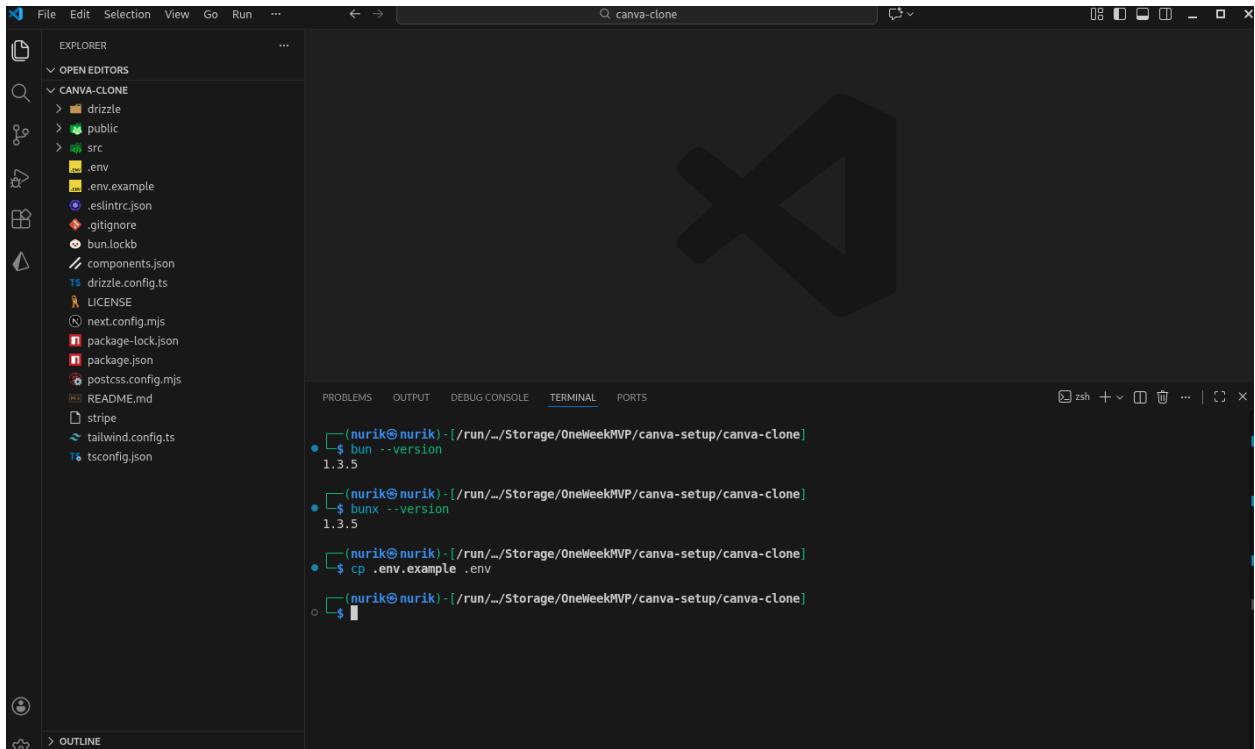


Fig.4: Check for versions and copy .env.example

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**.

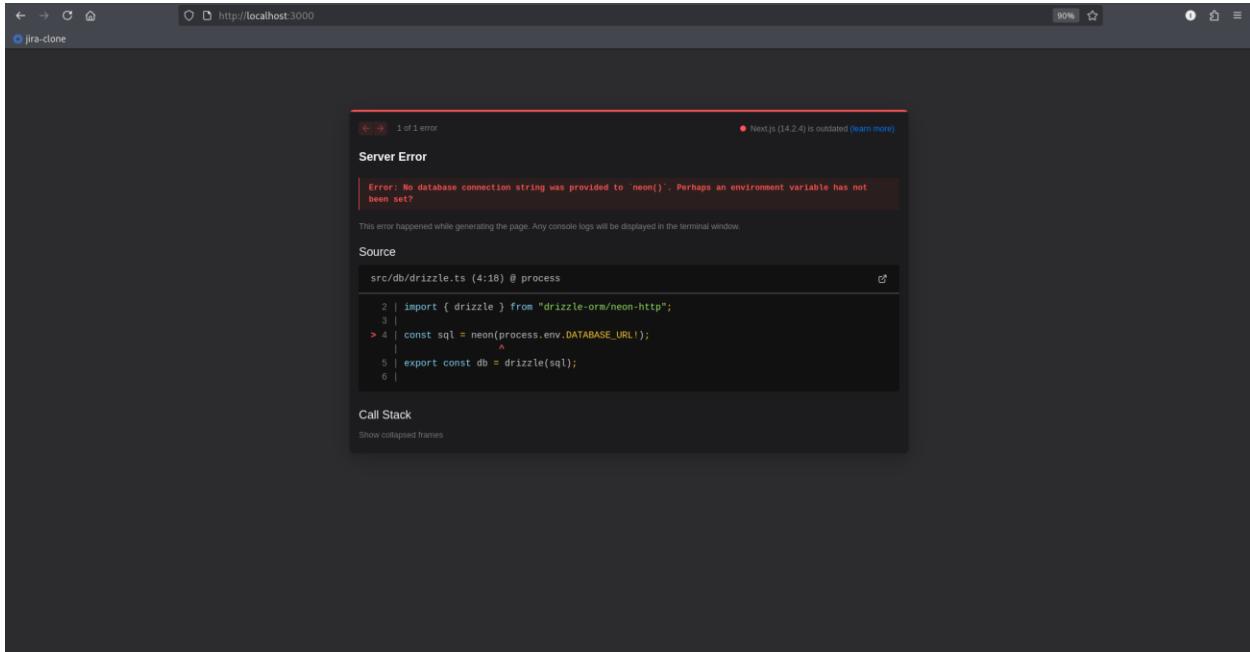


Fig.5: Database URL, missing it in .env file

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

OneWeekMVP's projects

Name	Region	Created at	Storage	Compute last active at	Branches	Integrations
oneweekmvp	AWS Europe Central 1 (Frankfurt)	Jan 7, 2026 3:22 pm	0	Jan 7, 2026 9:52 pm	2	Add ⚙️
lawyerai_telegram	AWS US East 1 (N. Virginia)	Dec 10, 2025 9:21 pm	30.84 MB	Jan 8, 2026 1:57 pm	2	Add ⚙️
lawyerAi	AWS US East 1 (N. Virginia)	Dec 4, 2025 6:42 pm	34.15 MB	Jan 8, 2026 2:00 pm	2	Add ⚙️
image-ai	AWS US East 2 (Ohio)	Jul 11, 2024 2:35 pm	102.8 MB	Jan 8, 2026 1:54 pm	1	Add ⚙️

Fig.6: Neon database

Create project

Project name	Postgres version
canva-clone	17

Cloud service provider

AWS  Azure

Region

AWS US East 1 (N. Virginia)

Select the region closest to your application.

What are you working on?

Deploying an app  
Code is working, now I need a database.

Building a new app  
I don't have (all the) working code yet.

Cancel Create

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

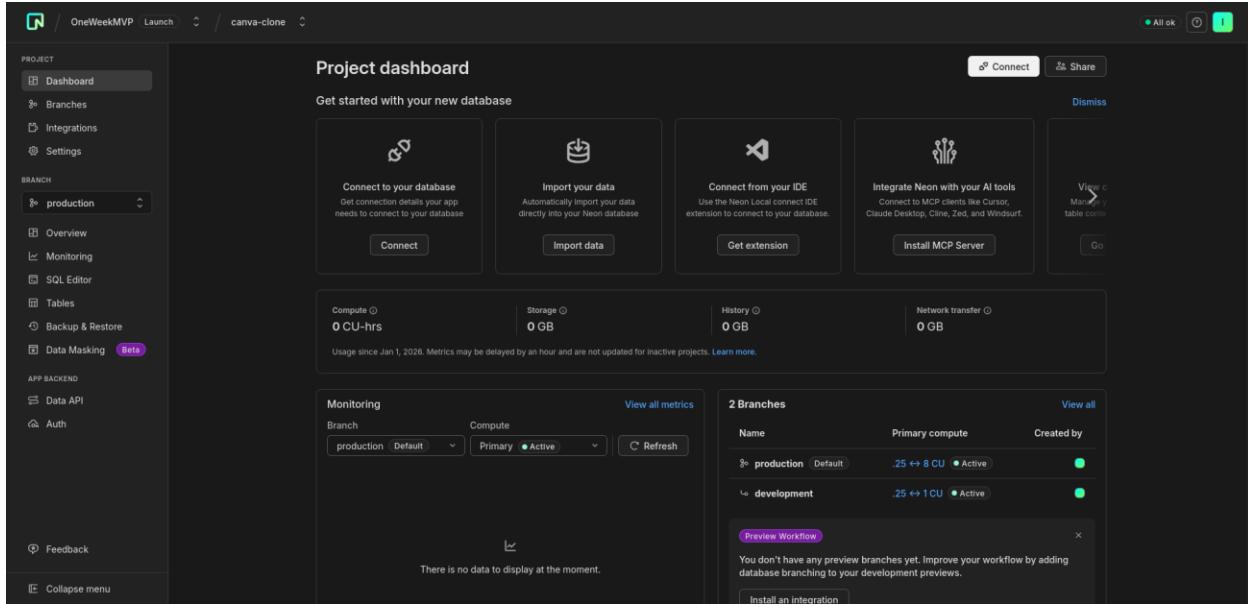


Fig.8: Project Dashboard

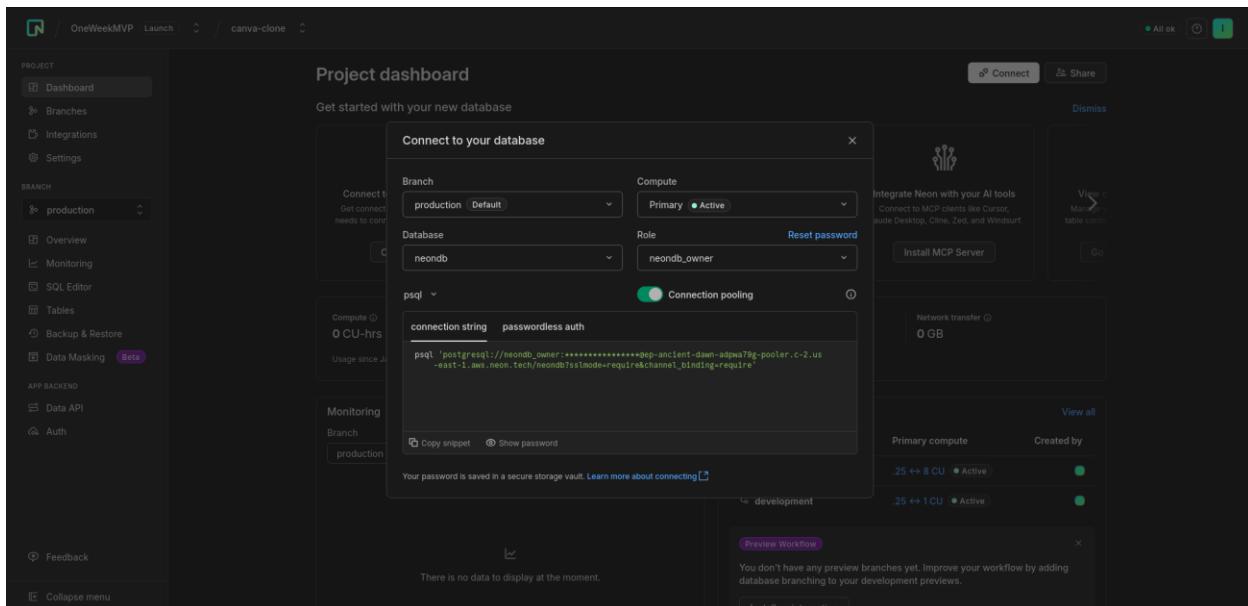


Fig.9: Copy your connection string

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.

The screenshot shows a code editor with an .env file open. The file contains environment variables for various authentication providers and a database connection. Below the editor is a terminal window showing the command `openssl rand -base64 32` being run, which generates a long base64-encoded string. The terminal also shows the `DATABASE_URL` and `AUTH_SECRET` variables being set.

```
File Edit Selection View Go Run Terminal Help
.env
12 # Replicate API token for AI models
13 AUTH_GITHUB_ID=
14 AUTH_GITHUB_SECRET=
15 # GitHub OAuth login credentials
16 AUTH_GITHUB_ID=
17 AUTH_GITHUB_SECRET=
18 # Google OAuth login credentials
19 AUTH_GOOGLE_ID=
20 AUTH_GOOGLE_SECRET=
21
22 # Stripe payment configuration
23 STRIPE_SECRET_KEY=
24 STRIPE_PRICE_ID=
25 STRIPE_WEBHOOK_SECRET=
26
27 # Database connection & auth secret
28 DATABASE_URL="postgres://neondb_owner:npq_4y0btaEFR20w@ep-ancient-dawn-adpwa79g-pooler.c-2.us-east-1.aws.neon.tech/neondb?sslmode=require"
29 AUTH_SECRET=WcpNR5051ffFBZ1NocGzrVGd3W04XR/Z00y7s+j5IM=
30
31

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
nurik@nurik:~/Storage/OneWeekWP/canca-setup/canca-clone$ openssl rand -base64 32
WcpNR5051ffFBZ1NocGzrVGd3W04XR/Z00y7s+j5IM=
nurik@nurik:~/Storage/OneWeekWP/canca-setup/canca-clone$ $
```

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",
```

The screenshot shows a terminal window within a code editor interface. The terminal is running a series of commands to set up a database. It starts by generating environment variables from a .env file:

```

12 # Replicate API token for AI models
13 REPLICATE_API_TOKEN=
14
15 # GitHub OAuth login credentials
16 AUTH_GITHUB_ID=
17 AUTH_GITHUB_SECRET=
18
19 # Google OAuth login credentials
20 AUTH_GOOGLE_ID=
21 AUTH_GOOGLE_SECRET=
22
23 # Stripe payment configuration
24 STRIPE_SECRET_KEY=
25 STRIPE_PRICE_ID=
26 STRIPE_WRAPIKEY_SECRET=

```

Then it runs a migration command:

```

$ bun run db:migrate

```

Output shows the migration process:

```

i bunx drizzle-kit generate
drizzle-kit: v0.22.8
drizzle-orm: v0.31.4

No schema changes, nothing to migrate 😊

```

Finally, it reads a config file and runs migrations:

```

$ bun run db:migration

```

Output shows the migration process:

```

i bunx drizzle-kit migrate
drizzle-kit: v0.22.8
drizzle-orm: v0.31.4

No config path provided, using default path
Reading config file '/run/media/nurik/Storage/OneWeekMVP/canva-setup/canva-clone/drizzle.config.ts'
Using 'pg' driver for database querying
[-] migrations applied successfully!

```

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.

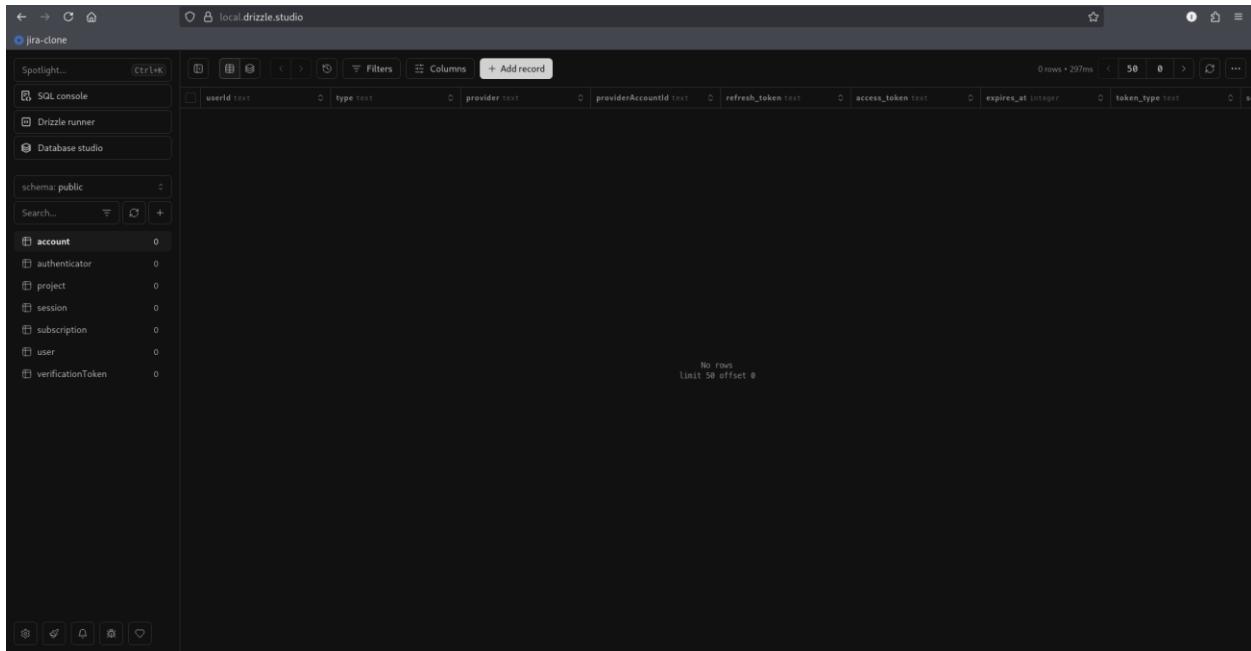


Fig.12: Drizzle studio, where your data is saved

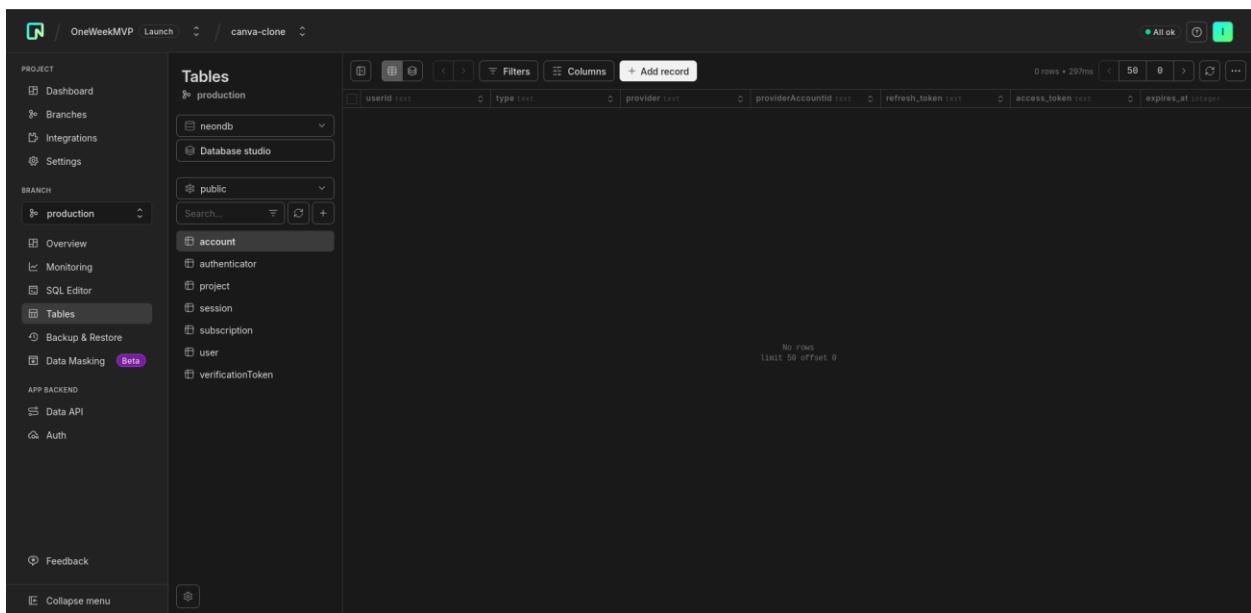


Fig. 13: Drizzle via Neon db

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

```
$ bun dev
```

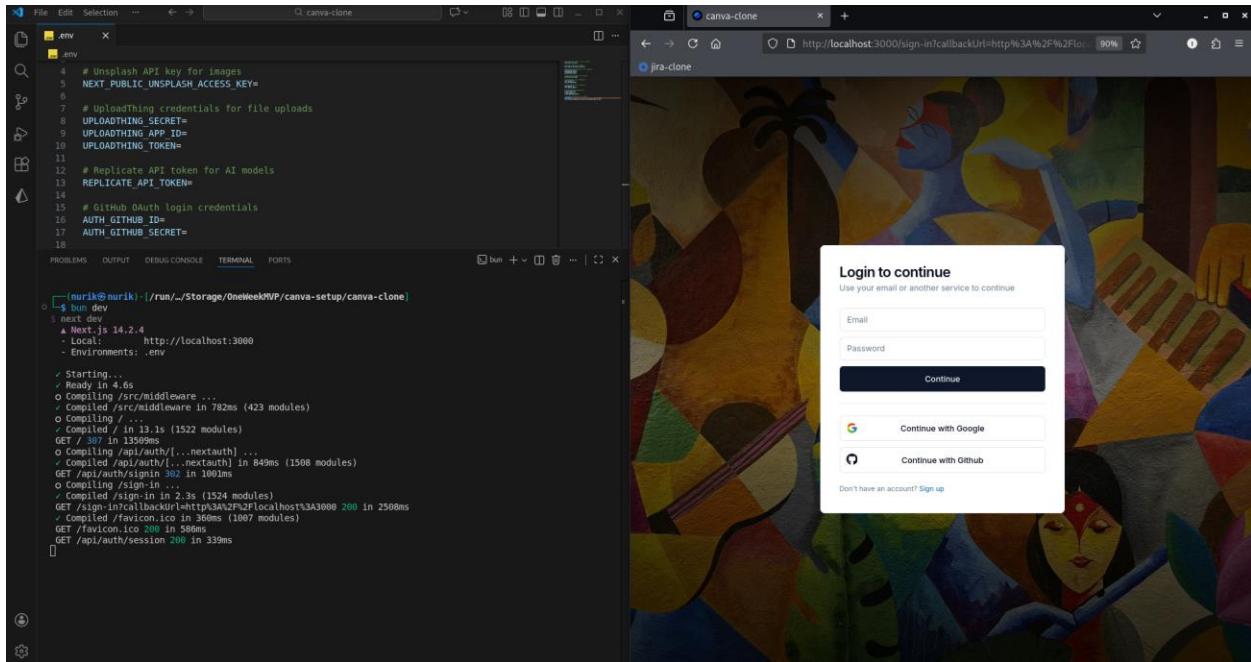


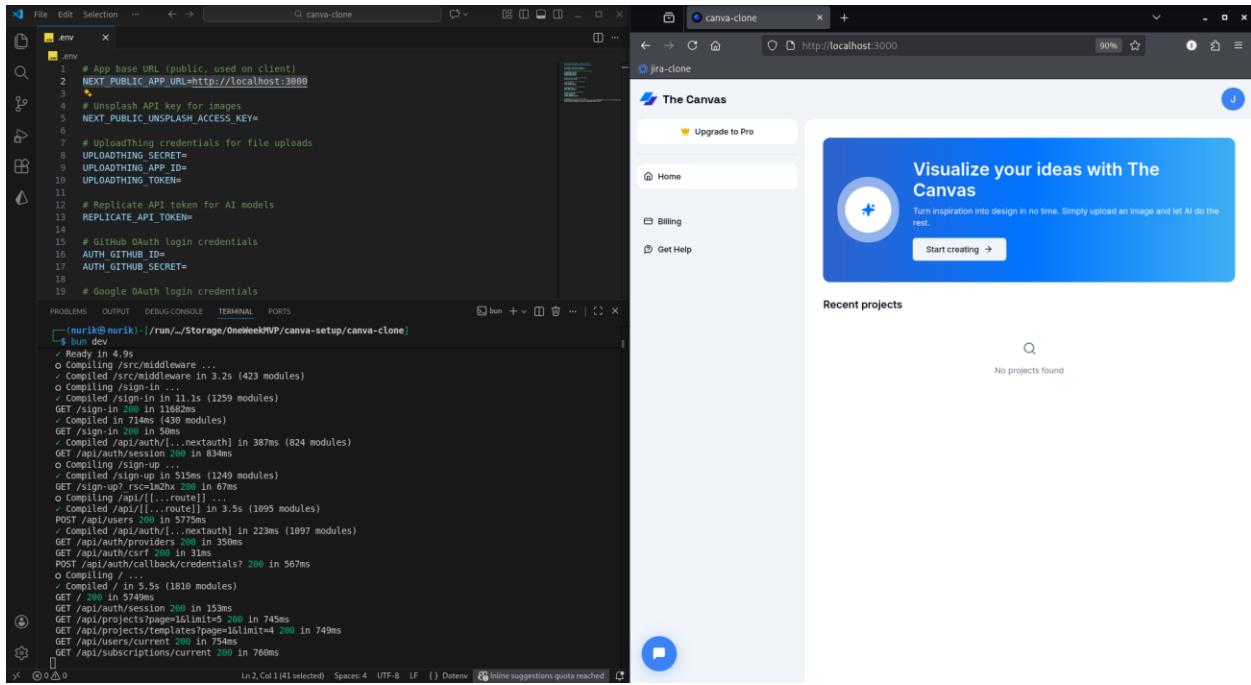
Fig.14: Now it is working on localhost:3000

Now fill your **NEXT\_PUBLIC\_APP\_URL** 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

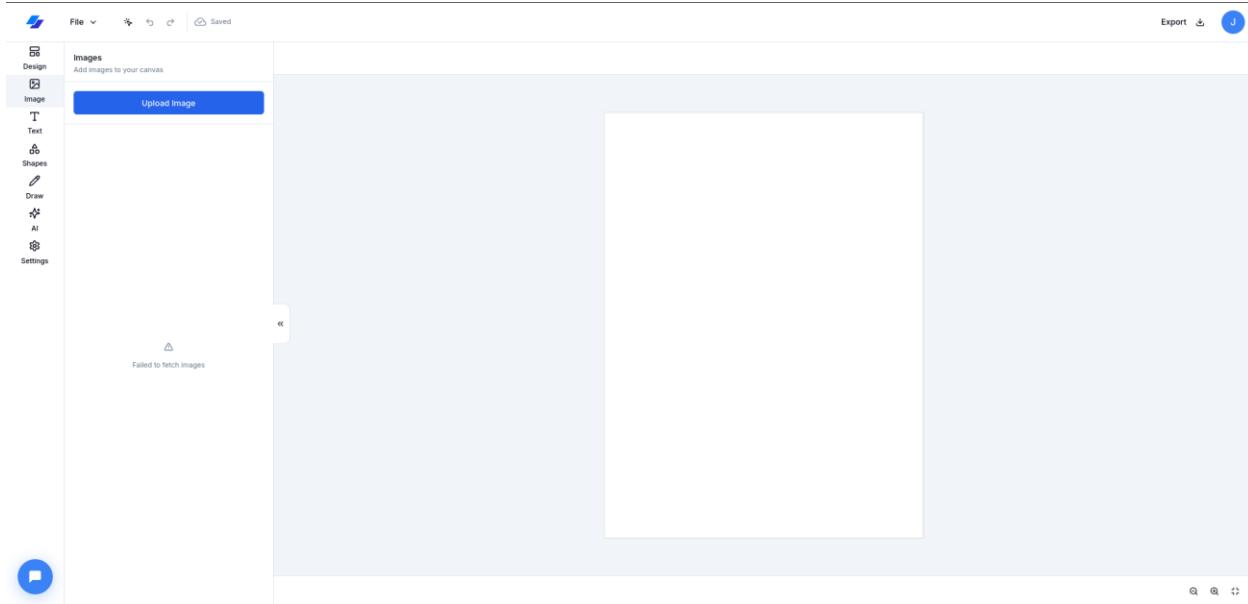


Fig.16: Before the integration with Unsplash and Uploadthing APIs

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>

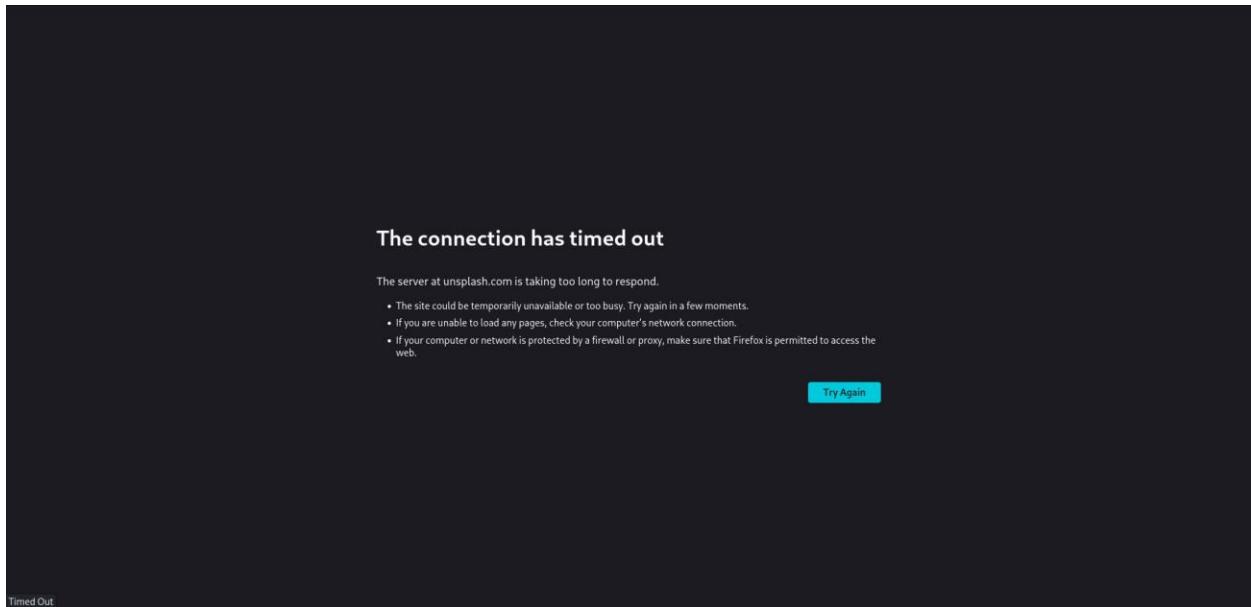


Fig.16.1: Connection Error

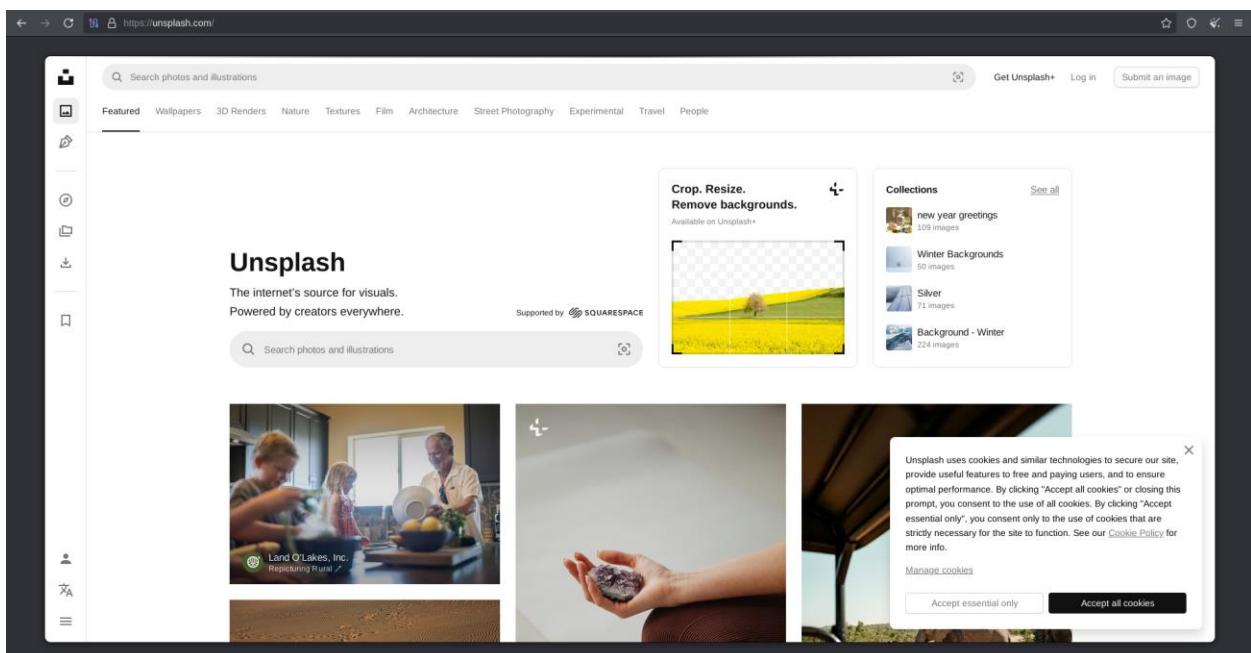


Fig.17: Unsplash via Tor

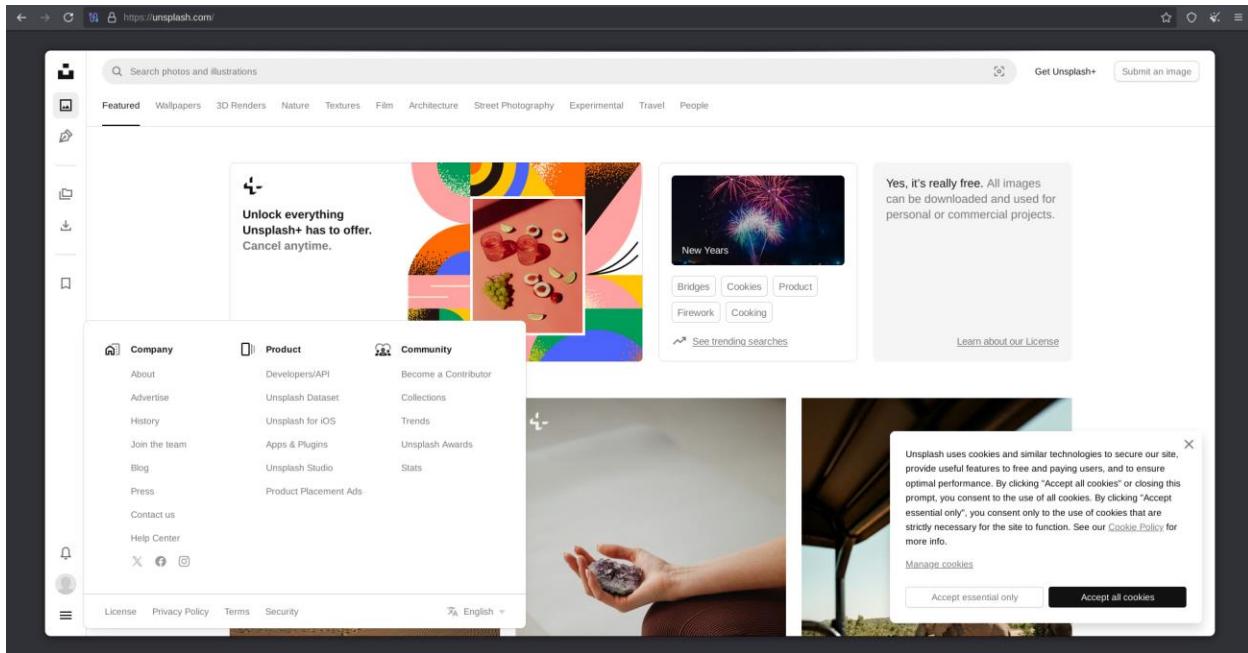


Fig.18: Select Developers API

The screenshot shows the Unsplash Developers page. It features a main heading 'The most powerful photo engine in the world.' and a sub-section for 'Your apps'. Below this, there's a callout for partnerships and a 'View the documentation' button. On the right, there's a preview of a JSON response from the API and a 'Cookie consent' dialog. At the bottom, there are sections for '0 requests/month', '8M photos', and '416.4K photographers', each accompanied by a small image thumbnail.

Fig: 19: Select 'Your apps'

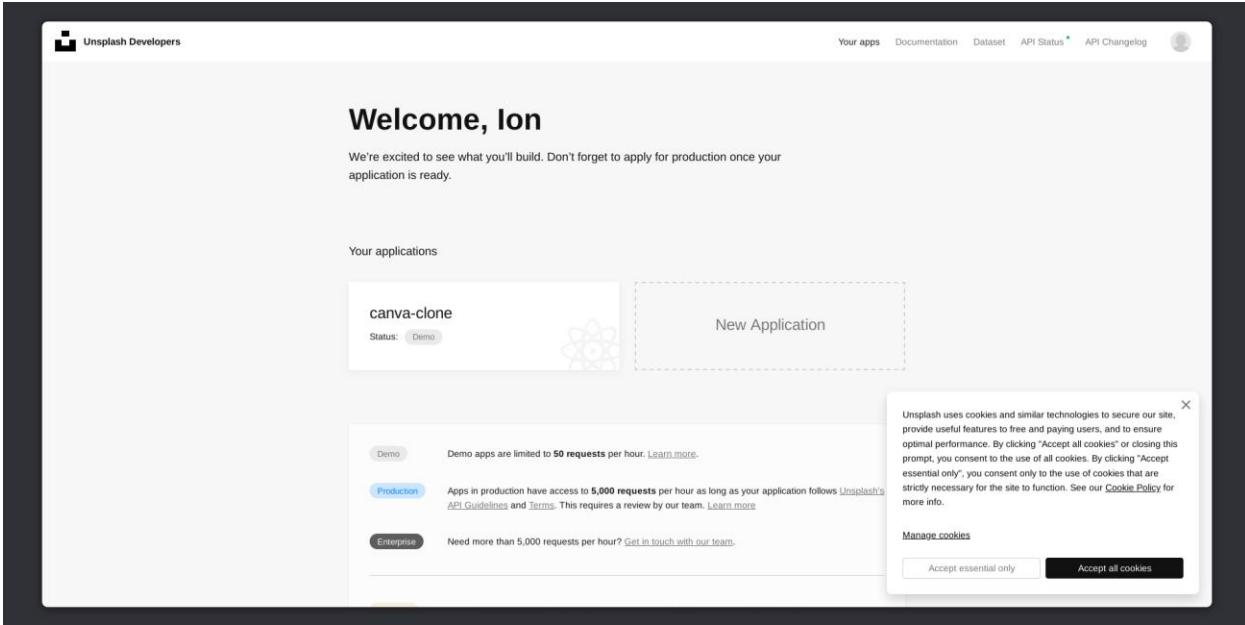


Fig.20: Create an app

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

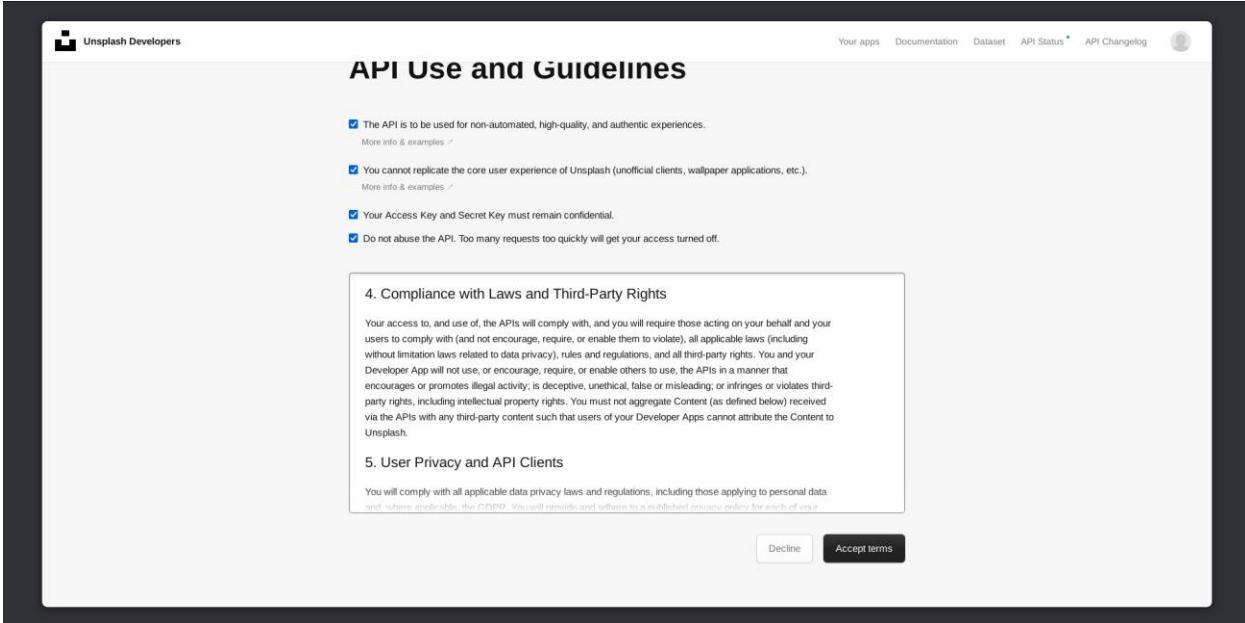


Fig.21: Accept terms

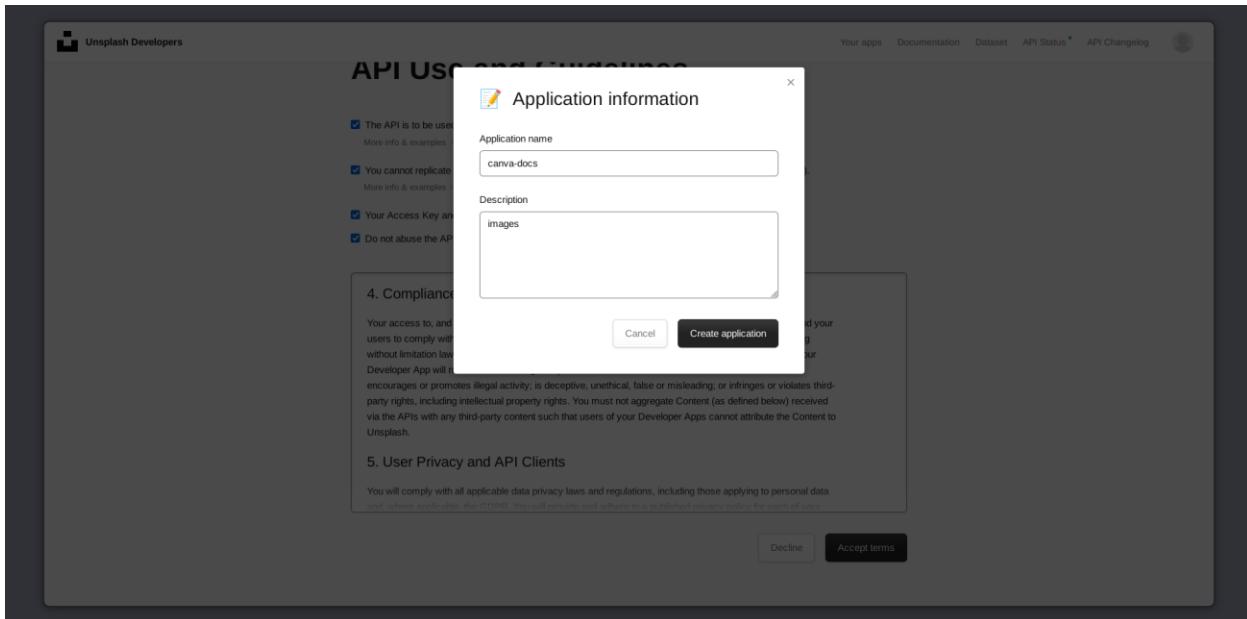


Fig.22: Name your app with a few words

```

.env
1 # App base URL (public, used on client)
2 NEXT_PUBLIC_APP_URL=http://localhost:3000
3
4 # Unsplash API key for images
5 NEXT_PUBLIC_UNSPASH_ACCESS_KEY=wyCcnMq4IK101w84cPZFF14E_aP7H8IqZAQx9KSXQ
6
7 # UploadThing credentials for file uploads
8 UPLOADTHING_SECRET=
9 UPLOADTHING_APP_ID=
10 UPLOADTHING_TOKEN=
11
12 # Replicate API token for AI models
13 REPLICATE_API_TOKEN=
14
15 # GitHub OAuth login credentials
16 AUTH_GITHUB_ID=
17 AUTH_GITHUB_SECRET=
18
19 # Google OAuth login credentials
20 AUTH_GOOGLE_ID=
21 AUTH_GOOGLE_SECRET=
22
23 # Stripe payment configuration
24 STRIPE_SECRET_KEY=
25 STRIPE_PRICE_ID=
26 STRIPE_WEBHOOK_SECRET=
27
28 # Database connection & auth secret
29 DATABASE_URL="postgresql://neondb_owner:npq_4yobtaFR20@ep-ancient-dawn-adpwa79g-pool
30 AUTH_SECRET=WcpNR5051ffFBZ1CNoGz2VGd3W04XR/Z00y75+j51M="

PROBLEMS OUTPUT DEBUG-CONSOLE TERMINAL PORTS
$ bun dev
[...]

```

Fig.23: Grab your Access Key

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

- **NEXT\_PUBLIC\_UNSPASH\_ACCESS\_KEY=**
- Restart your localhost:3000 and see again!

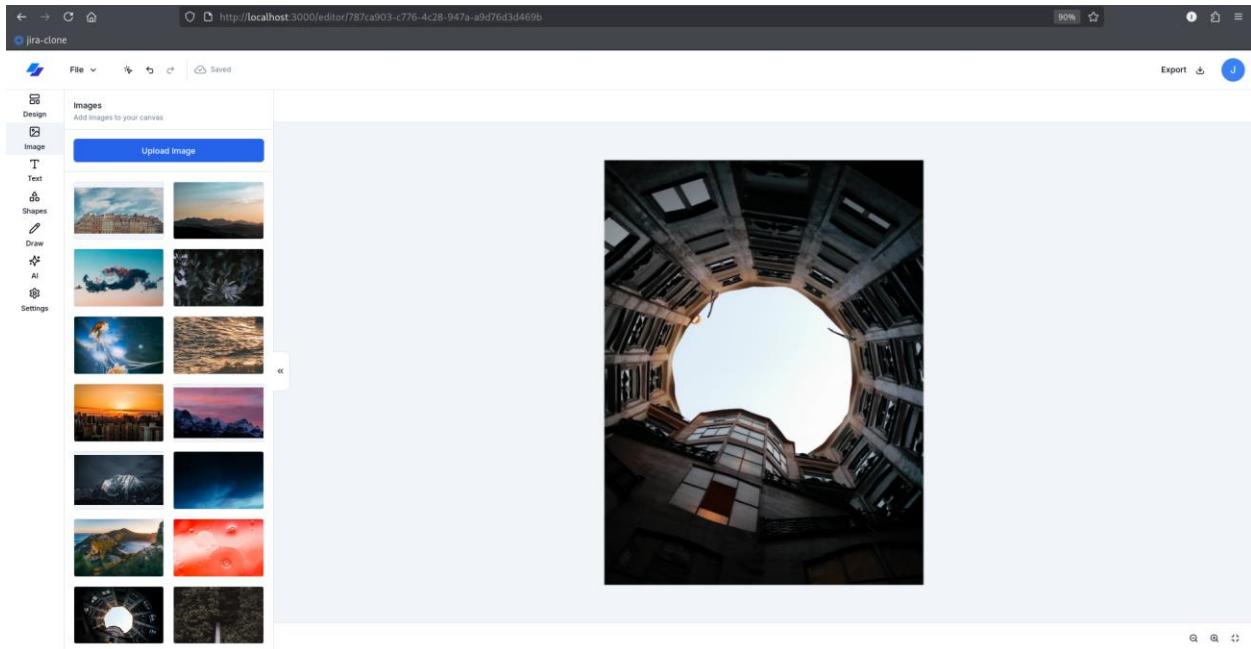


Fig.24: Unsplash is set

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

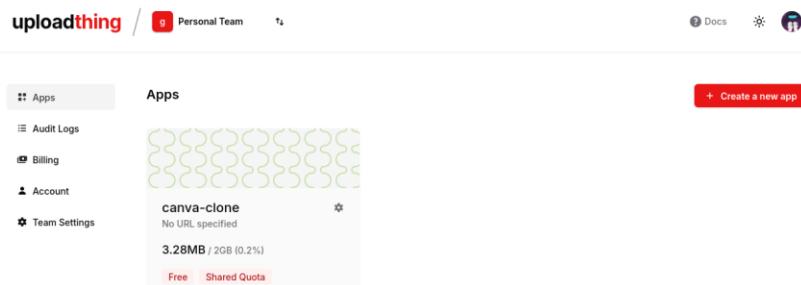


Fig.25: Projects Dashboard

Go and create a new app:

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

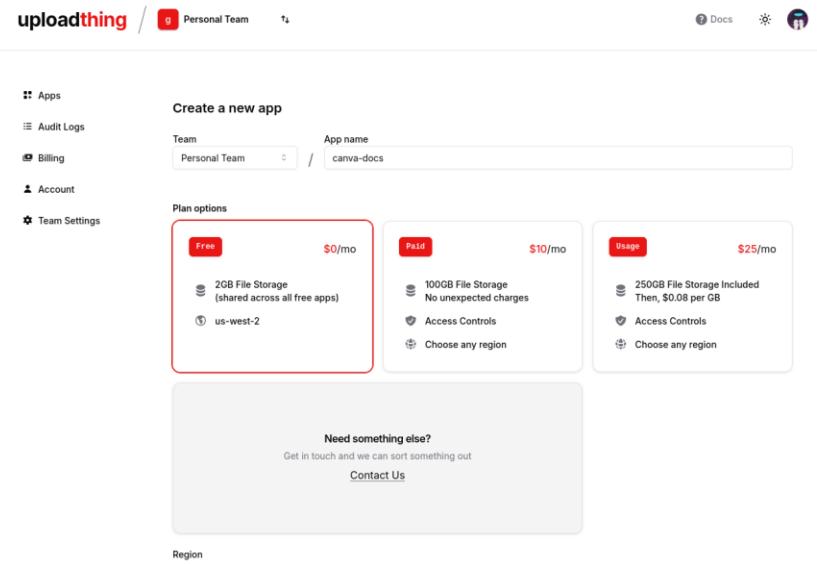


Fig.26: Name && Select Free

The screenshot illustrates a developer's workflow across three main windows:

- Code Editor:** Shows an `.env` file with environment variables. A red arrow points from the `UPLOADTHING_TOKEN` variable in the code to the corresponding value in the API Keys section of the UploadThing dashboard.
- Terminal:** Shows the developer running `npm run dev` in a project directory. The terminal output includes logs for various API requests and responses, such as PATCH and GET requests to the `/api/projects` endpoint.
- UploadThing Dashboard:** Shows the `API Keys` section where the `UPLOADTHING_TOKEN` is listed under the `SDK v7+` tab. The token value is `eyJhbGk...ZTc`.

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

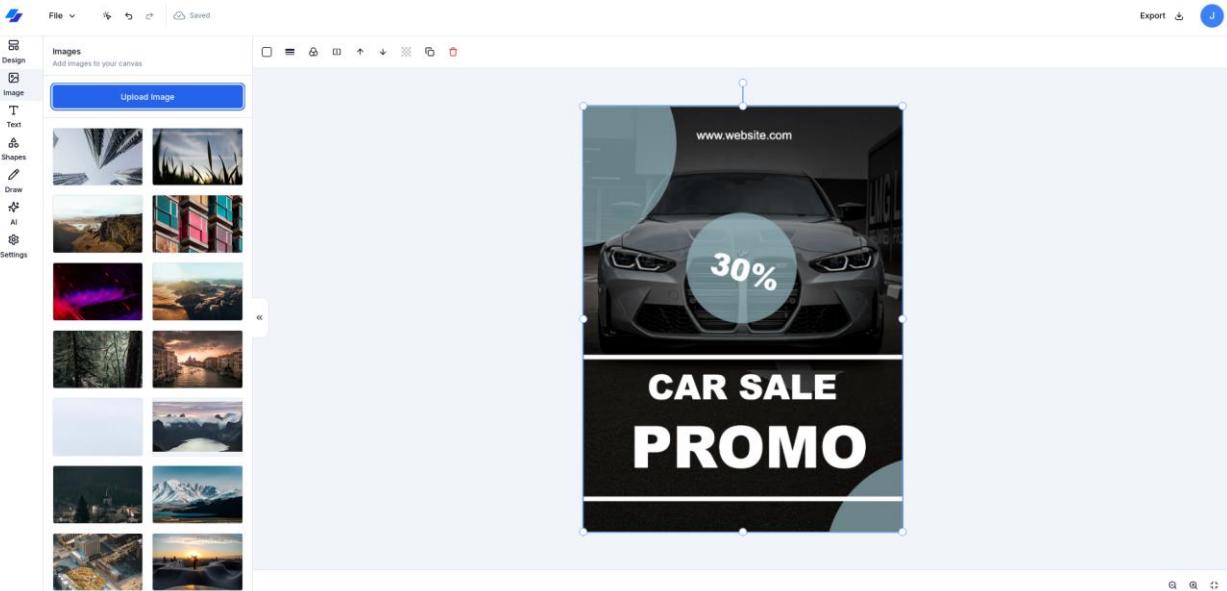


Fig.28: Upload Image

*It works! Pow, pow. Wow.*

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

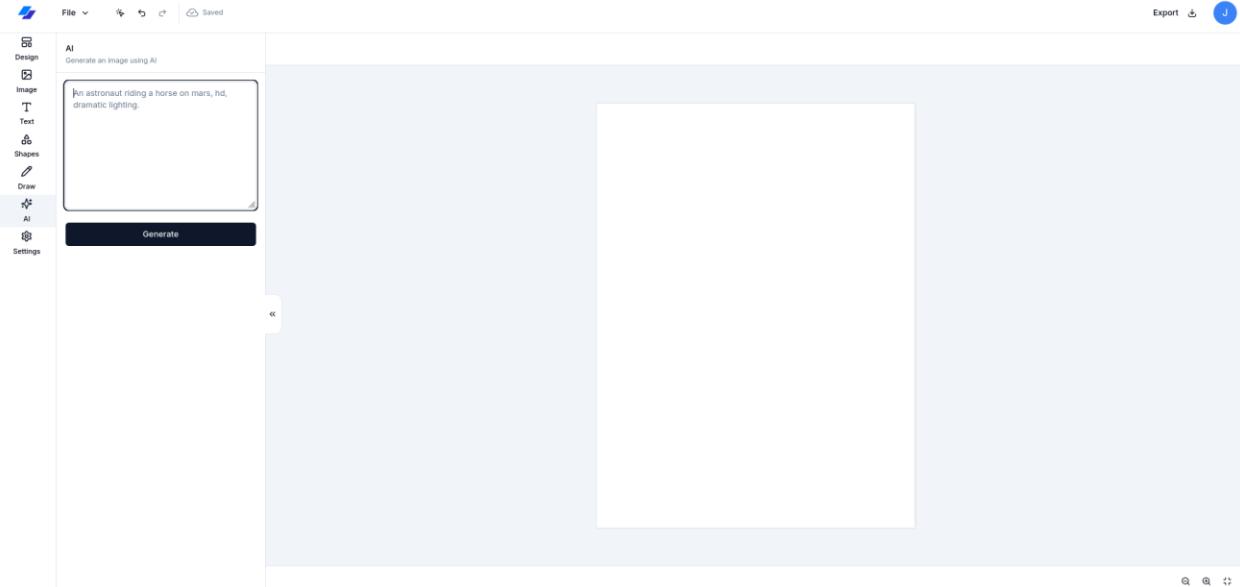


Fig.29: Before AI

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.

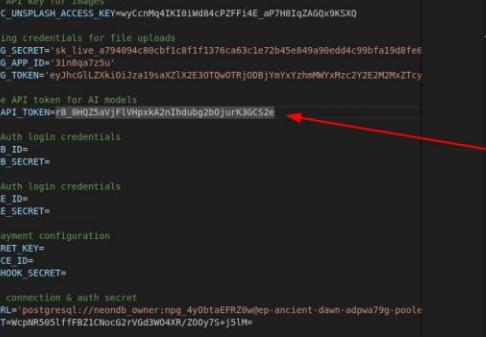
Outstanding balance this month	Usage this month	Credit remaining
\$0.00	\$0.28	\$1.72

Month	Status	Total
January 2026	In progress	\$0.28

Date	Status	Total
1/6/2026	Succeeded	\$2.00

All services are online

Fig.30: Replicate Billing Page



```
# app base URL (public, used on client)
1 NEXT_PUBLIC_APP_URL=http://localhost:3000
2
3 # Unsplash API key for images
4 NEXT_PUBLIC_UNSPLASH_ACCESS_KEY=wyCcmMq4IK10iwb84cPZFfI4E_aP7HB1qZAG0x9KSQ
5
6 # UploadThing credentials for file uploads
7 UPLOADTHING_SECRET=ek_live_a79494c80cb1c18ff137e63c63c1e72b4e8a99eddc499bf1a9d8f6
8 UPLOADTHING_APP_ID=sin8q0t7zsu
9 UPLOADTHING_TOKEN=eyJhcGlfLzXkIjoiJz19saXZtX2E30TQwOTRj00B8YmYxzHmMyxzCzY2E2M2NxTcy
10
11 # Replicate API token for AI models
12 REPLICATE_API_TOKEN=r8_0HhZ5avFLVhpxA2n1hdubg2b0jurK3cCS2e
13
14 # GitHub OAuth login credentials
15 AUTH_GITHUB_ID=
16 AUTH_GITHUB_SECRET=
17
18 # Google OAuth login credentials
19 AUTH_GOOGLE_ID=
20 AUTH_GOOGLE_SECRET=
21
22 # Stripe payment configuration
23 STRIPE_SECRET_KEY=
24 STRIPE_PRICE_ID=
25 STRIPE_WEBHOOK_SECRET=
26
27 # Database connection & auth secret
28 DATABASE_URL='postgresql://neonb_owner:np_y0btaERFZowep-ancient-dawn-adpwa7g-pool
29 AUTH_SECRET=NcpnR50$1ffFBz1CnC62vGd3W0XR/200y7S+j5IM'
30
31
```

The screenshot shows a terminal window on the left and a browser window on the right. The terminal window displays a POST request to upload an image to a cloud storage service, resulting in a success message with a file ID. A red arrow points from the terminal's file ID back to the browser's API tokens list, where a token named 'canva-docs' is shown with a similar file ID.

replicate.com/account/api-tokens

Account settings

General

API tokens

Billing

Integrations

Webhooks

Enter token name

Create token

canva-docs  
r8\_6HQ\*\*\*\*\*

canva-clone  
r8\_K80\*\*\*\*\*

Default  
r8\_Bni\*\*\*\*\*

All services are online

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.

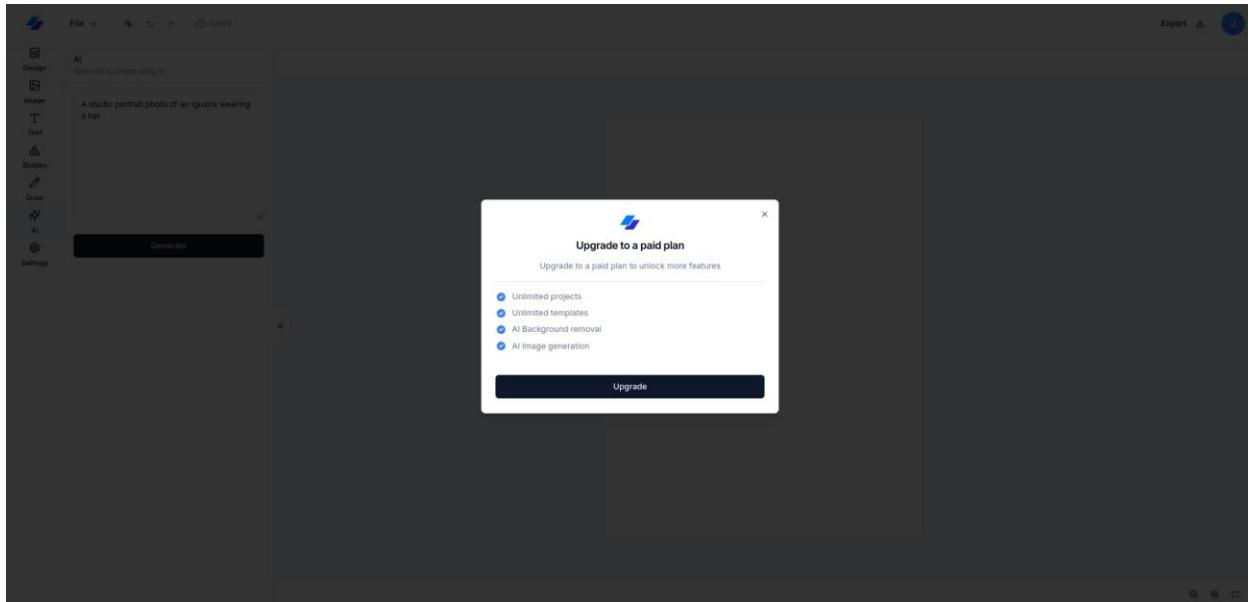


Fig.32: Paidwall && Next is Stripe

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.

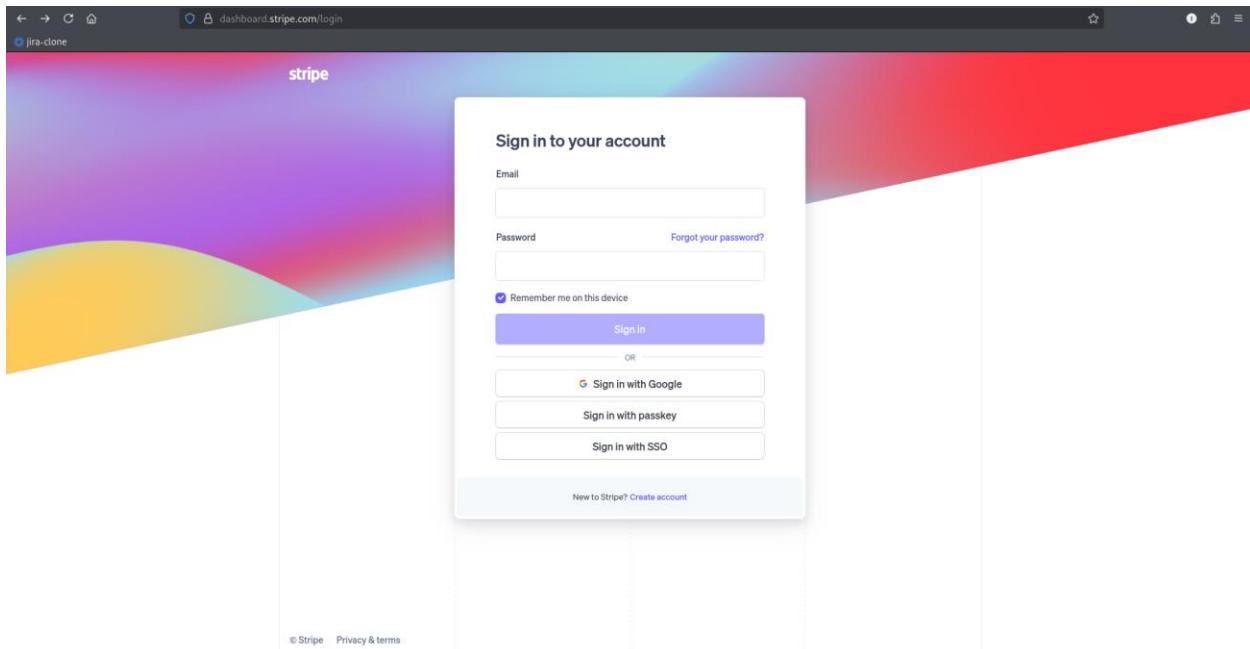


Fig.33: Stripe Dashboard

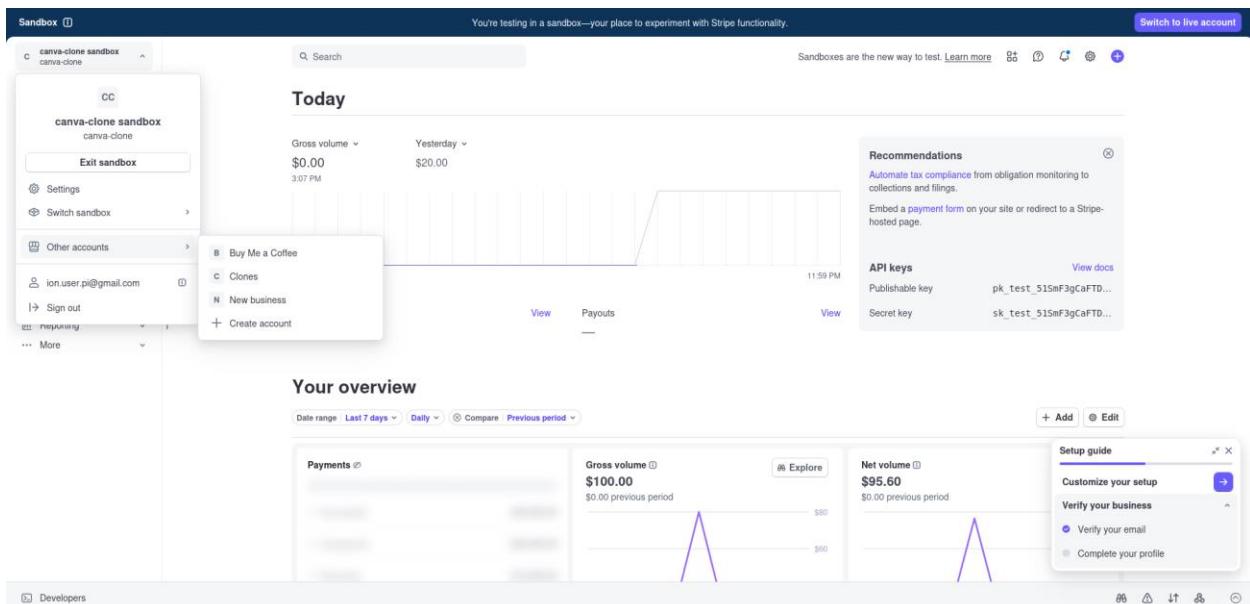


Fig.34: Create a new account

The screenshot shows the Stripe Sandbox interface. At the top, it says "Sandbox" and "You're testing in a sandbox—your place to experiment with Stripe functionality." There's a "Switch to live account" button. The main area has a sidebar with "Home", "Balances", "Transactions", "Customers", and "Product catalog". Below that is the "Workbench" section with tabs: Overview (selected), Webhooks, Events, Logs, Health, Inspector, Blueprints, and Shell. The "Your integration" section shows "API requests" and "Event deliveries", both with "No requests in the last 7 days". The "Testing tools" section includes "Test keys" (with a "Publishable key" and "Secret key" shown), "Test data deletion" (with a "Review test data" button), and a "Health" section. A "Shell" terminal window at the bottom shows a .env file with a Stripe secret key.

Fig.35: Secret key

This screenshot shows a terminal window with a .env file open. The file contains various environment variables, including the Stripe secret key: `STRIPE_SECRET="sk_test_51...".` A red arrow points from this line to the Stripe dashboard window above, indicating where the key should be pasted. The terminal also shows some log output from a command like `npm run dev`.

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

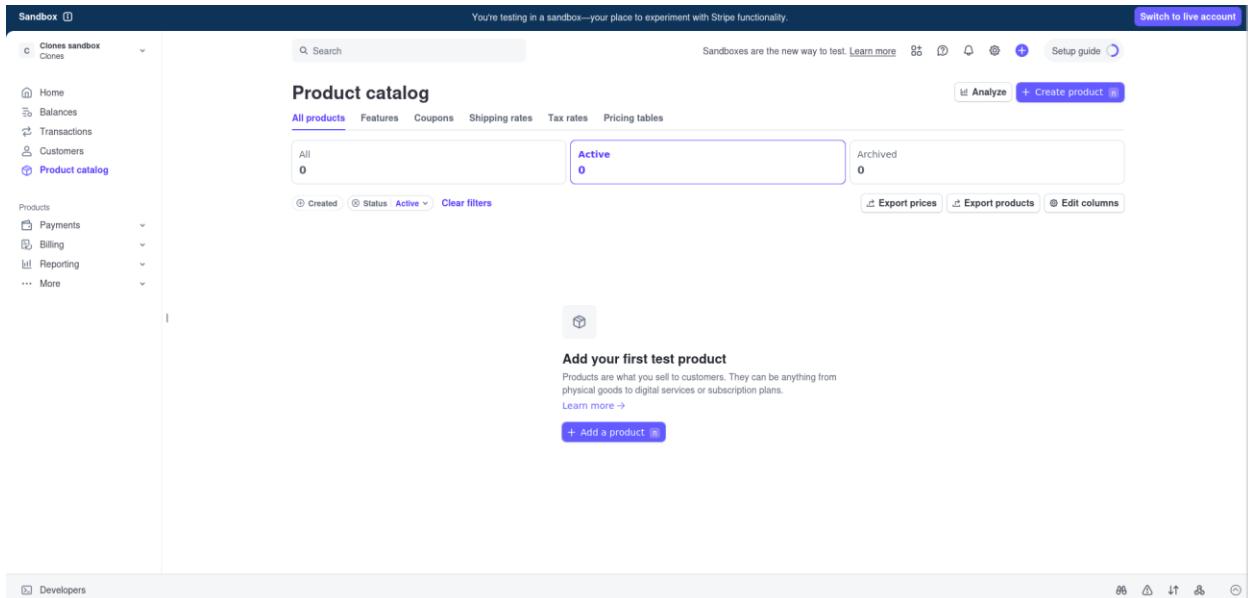


Fig.36: Adding product

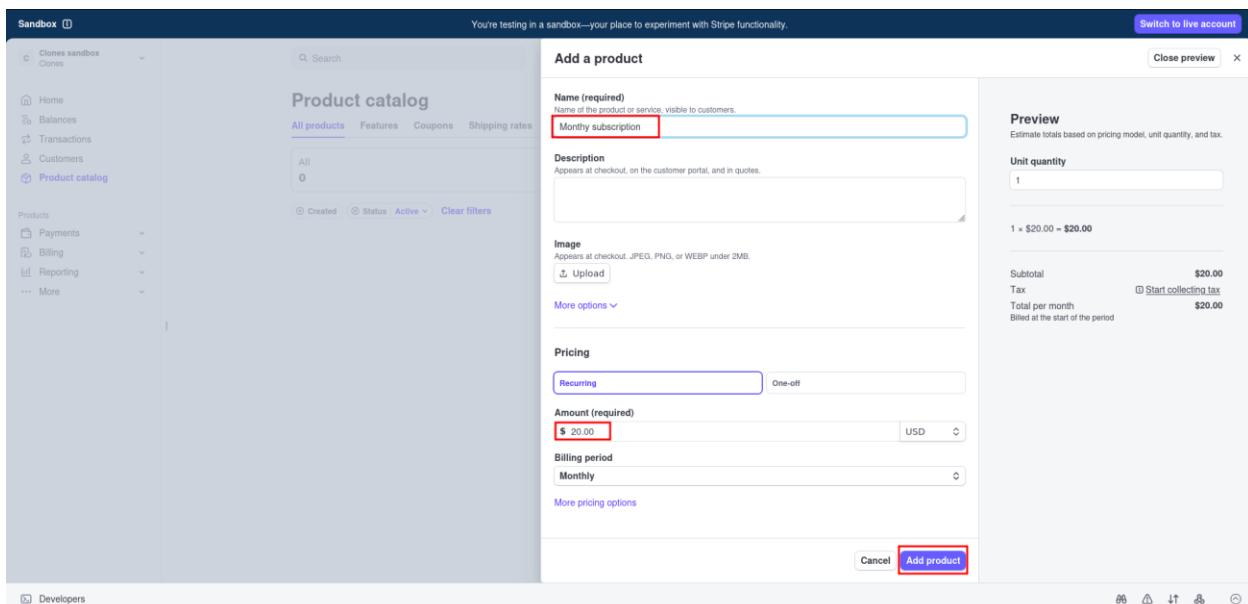


Fig.37: Product

The terminal window shows the following environment variables:

```

1 # App base URL (public, used on client)
2 NEXT_PUBLIC_APP_URL=http://localhost:3000
3
4 # Unsplash API key for images
5 NEXT_PUBLIC_UNSPLASH_ACCESS_KEY=veyCnnMq4IK101wd84cPZFF14E_aP7H8IqZAG0x9KSX0
6
7 # UploadThing credentials for file uploads
8 UPLOADTHING_SECRET=sk_live_a794949480cb1cf1f137ca63c1e72b45e849a90edd4c99bfa19d8fe6
9 UPLOADTHING_APP_ID=3ln8q7z5U
10 UPLOADTHING_TOKEN=eyJhbGciOiJzIiJ9.a2195axZLXZE30TQw0THJ0D8jYmYXzHMMYXzL2YzE2MzRzZTc
11
12 # Replicate API token for AT models
13 REPLICATE_API_TOKEN=r8_0HQZ5avJFlVhpXkA2n1hdubg2b0jurK36CS2e
14
15 # GitHub OAuth login credentials
16 AUTH_GITHUB_ID=
17 AUTH_GITHUB_SECRET=
18
19 # Google OAuth login credentials
20 AUTH_GOOGLE_ID=
21 AUTH_GOOGLE_SECRET=
22
23 # Stripe payment configuration
24 STRIPE_SECRET=sk_test_51mF02ARqan391kbXa0d02X4gA0CqJkYsgJ6D4Mb53c8yPhzSBrvkEkh
25 STRIPE_PRICE_ID=price_1SnL8pARqan391kbv3f296J
26 STRIPE_WEBHOOK_SECRET=
27
28 # Database connection & auth secret
29 DATABASE_URL='postgres://neondb:owner:rnp_4y0btaERZ@sep-ancient-dawn-adpwa79g-pool
30 AUTH_SECRET=WcpNR5051ffFB21CNoG2rVGd3W04XR_Z0oy75+j51M
31

```

The browser window shows the Stripe dashboard under the 'Sandbox' tab. A red arrow points to the 'Copy price ID' button in the 'Pricing' section of the 'Monthly subscription' product page.

Fig.38: Copy price ID

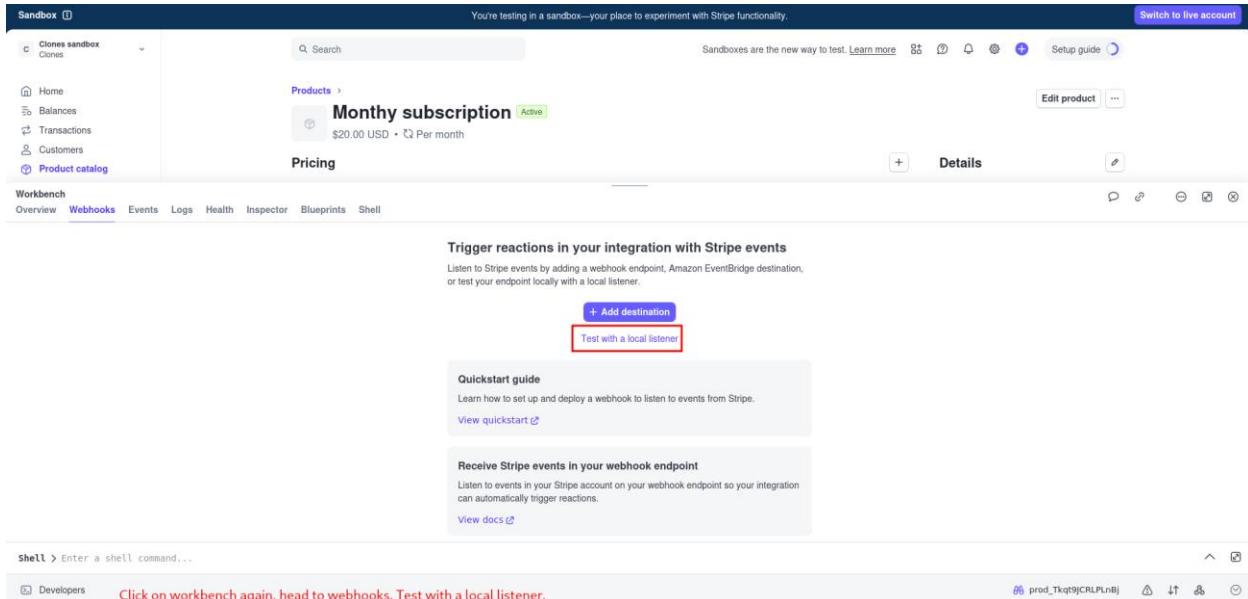


Fig.39: Test with a local listener

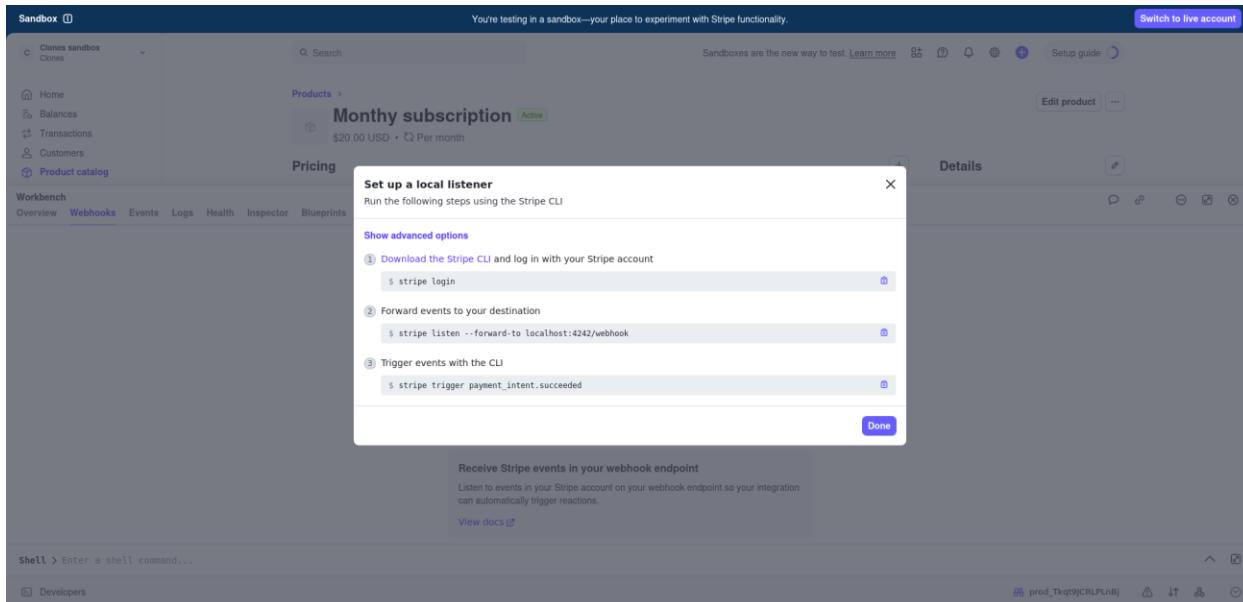


Fig.40: Local listener setup

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

**stripe DOCS**

Get started Payments Revenue Platforms and marketplaces Money management Developer resources

Overview

**VERSIONING**

- > Changelog
- Upgrade your API version
- Upgrade your SDK version

**ESSENTIALS**

- > SDKs
- > API
- > Testing
- > Stripe CLI
  - Overview
  - Install the CLI
  - Use the CLI
  - Enable autocompletion
  - CLI keys
  - Trigger events
  - Upgrade the CLI
- > Sample projects

**TOOLS**

- United States
- English (United States)

Developers

**ON THIS PAGE**

- Install the Stripe CLI
- Log in to the CLI
- Get started with a video

**1 Install the Stripe CLI**

From the command line, use an install script or download and extract a versioned archive file for your operating system to install the CLI.

homebrew apt yum Scoop macOS Linux **Windows** Docker

To install the Stripe CLI on Windows without Scoop:

- Download the latest windows zip file from GitHub.
- Unzip the stripe-X.X.X-windows-x86\_64.zip file.
- Add the path to the unzipped stripe.exe file to your Path environment variable. To learn how to update environment variables, see the Microsoft PowerShell documentation.

Windows anti-virus scanners occasionally flag the Stripe CLI as unsafe. This is very likely a false positive. For more information, see Issue #692 in the GitHub repository.

**2 Log in to the CLI**

Log in and authenticate your Stripe user account to generate a set of restricted keys. To learn more, see Stripe CLI keys and permissions.

Command Line

```
$ stripe login
```

Press Enter on your keyboard to complete the authentication process in your browser.

Fig.42: Go with the flow

ENV

```

1 # App base URL (public, used on client)
2 NEXT_PUBLIC_APP_URL=http://localhost:3000
3
4 # Unsplash API key for images
5 NEXT_PUBLIC_UNSPLASH_ACCESS_KEY=yourUnsplashAPIKey
6
7 # UploadThing credentials for file uploads
8 UPLOADTHING_SECRET='sk_live_a794094c80cb1cf1f1376ca33ce72b45e849a9eddc99bf19d8fe6
9 UPLOADTHING_APP_ID='3in8q07z5u'
10 UPLOADTHING_TOKEN='eyJhbGciLzKkIjOj2a19axZLX2e30T0wOTRj00BjYmYxzhMwYxMzc2Y2E2M2hXZTcy
11
12 # Replicate API token for AI models
13 REPLICATE_API_TOKEN='r_0H0Z5avF1VpoxA2nIhdubg2b0jurK3Gc52e
14
15 # GitHub OAuth login credentials
16 AUTH_GITHUB_ID=
17 AUTH_GITHUB_SECRET=
18
19 # Google OAuth login credentials
20 AUTH_GOOGLE_ID=
21 AUTH_GOOGLE_SECRET=
22
23 # Stripe payment configuration
24 STRIPE_SECRET_KEY='sk_test_515mF02ARgan391kbKa0d02xM4gA0CqZkYsg16dMbs3c8yphz5BrvukEk'

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

nurik@nurik-[~/.Storage/OneWeekMVP/canca-setup/canca-clone]

```

o $ ./stripe login
Your pairing code is: clears-lovely-excede-wonder. Follow link (ctrl + click)
This pairing code verifies you're automatically authorized to access your account. If you didn't request this pairing code, or if you believe it was sent to the wrong person, visit https://dashboard.stripe.com/stripecli/confirm_auth?t=9092laBA1ehzuJGryDwvlytfoX&[redacted] (^c to quit)

```

Sandbox

stripe

**Allow Stripe CLI to access your account information?**

Verify that the pairing code below matches the one shown in the Stripe CLI login command.

**clears-lovely-excede-wonder**

If you did not initiate this request, let us know and deny the request.

**Deny access** **Allow access**

Fig.43: Allow access

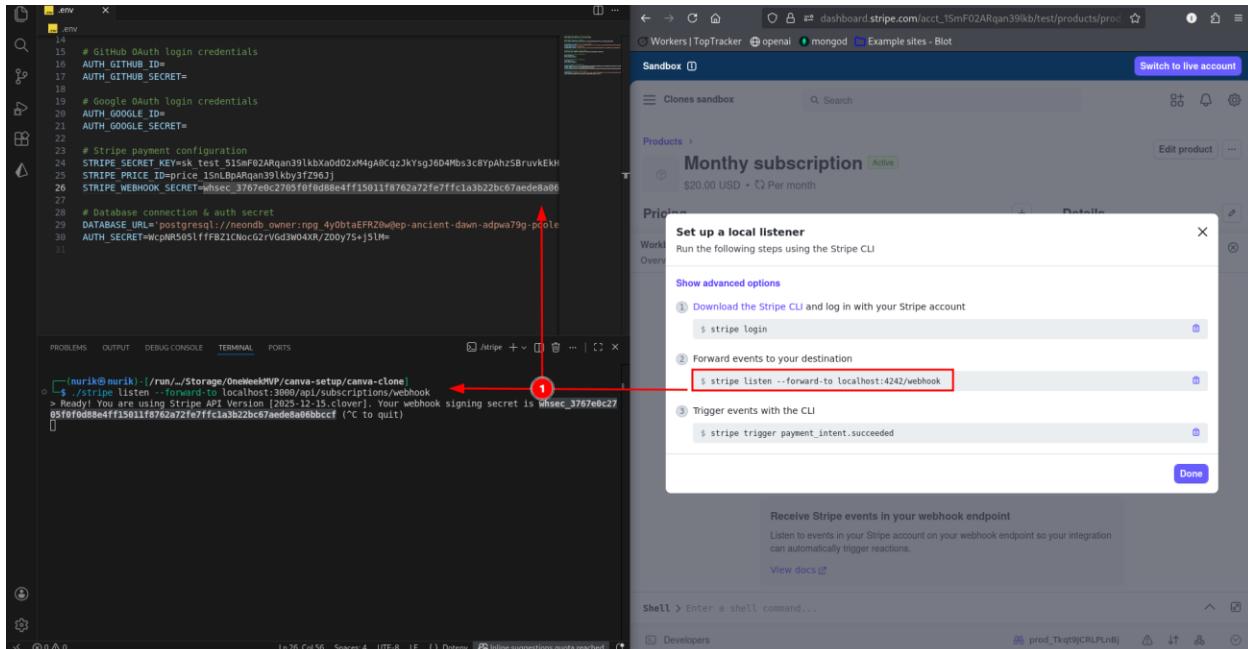


Fig.44: Stripe listen on localhost:3000

Do not close this terminal!

Open a new terminal for commands. And run:

\$ bun dev

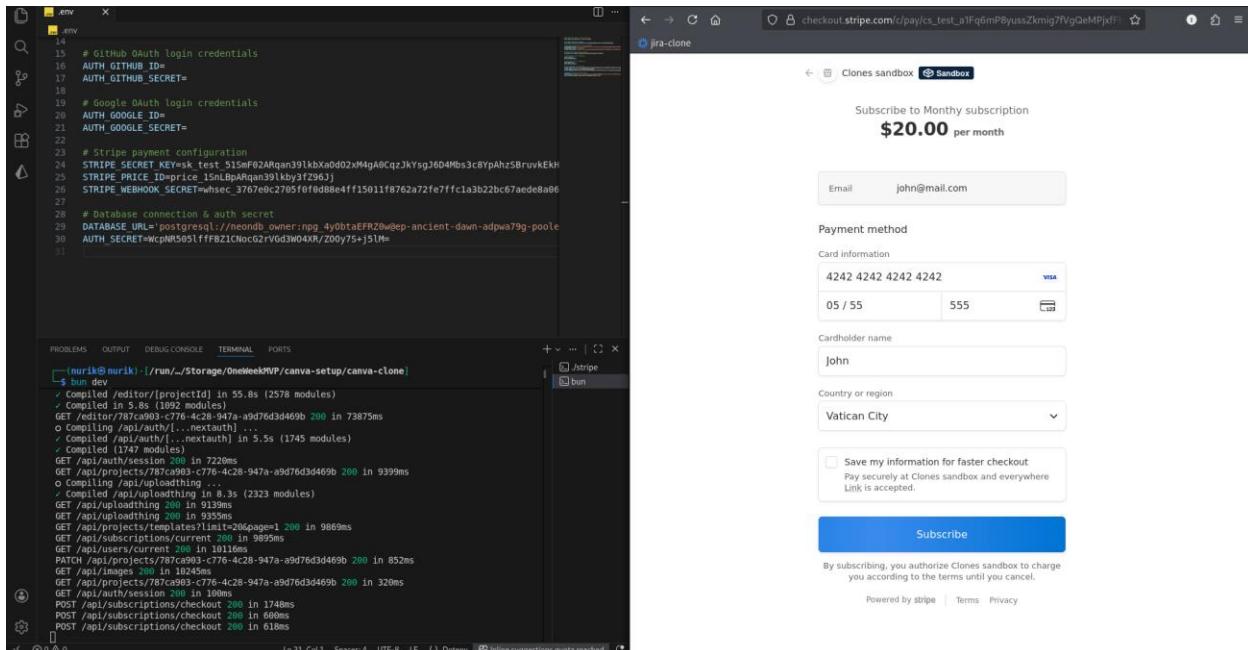


Fig.45: Dummy info && Subscribe

The screenshot shows a terminal window with several tabs open. One tab displays a configuration file with environment variables like GitHub and Google OAuth credentials, Stripe payment configuration, and database connection details. Another tab shows the command `curl -X POST https://api.thecanvas.com/v1/subscriptions/webhook -H "Content-Type: application/json" -d '{\"url\": \"http://localhost:3000/editor/787ca903-c776-4c28-947a-a9d76d3d469b\", \"secret\": \"whsec\_3767e0c2705f6f0d88e4ff15011f8762a72fe7ffc1a3b2bc67ae8a06\"}'` being run. A third tab shows the response from the API, indicating a successful subscription with the message "Subscription successful!". The terminal also shows logs for a Next.js application running on port 3000.

Fig.46: Successful

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

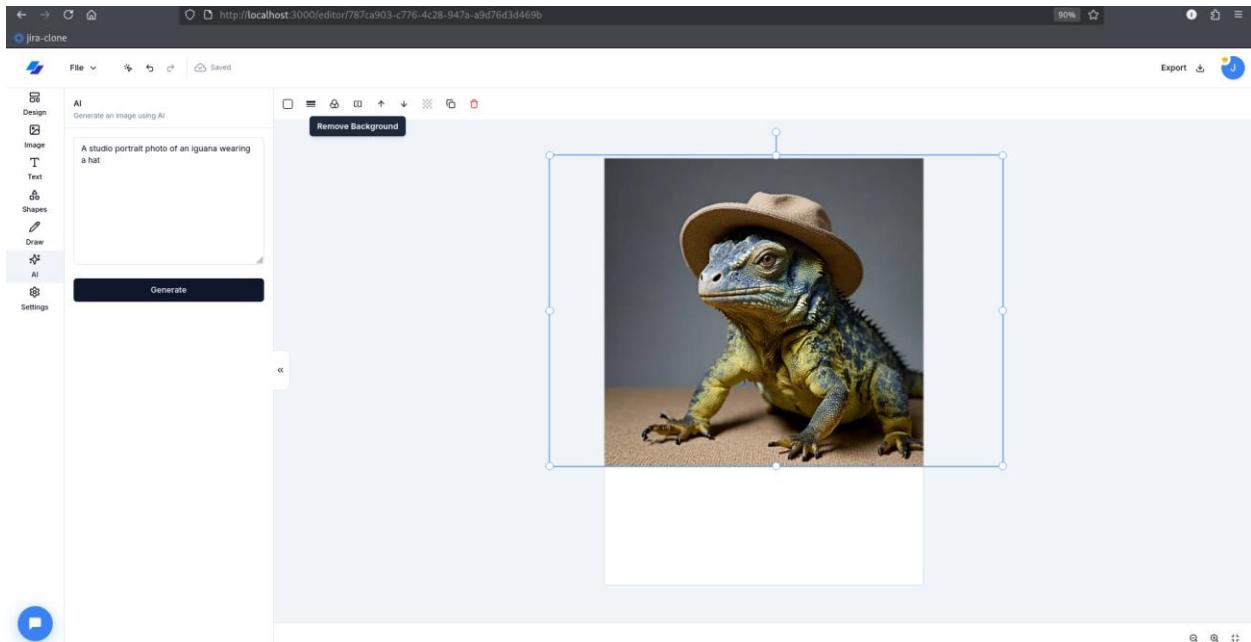


Fig.47: Image Generation

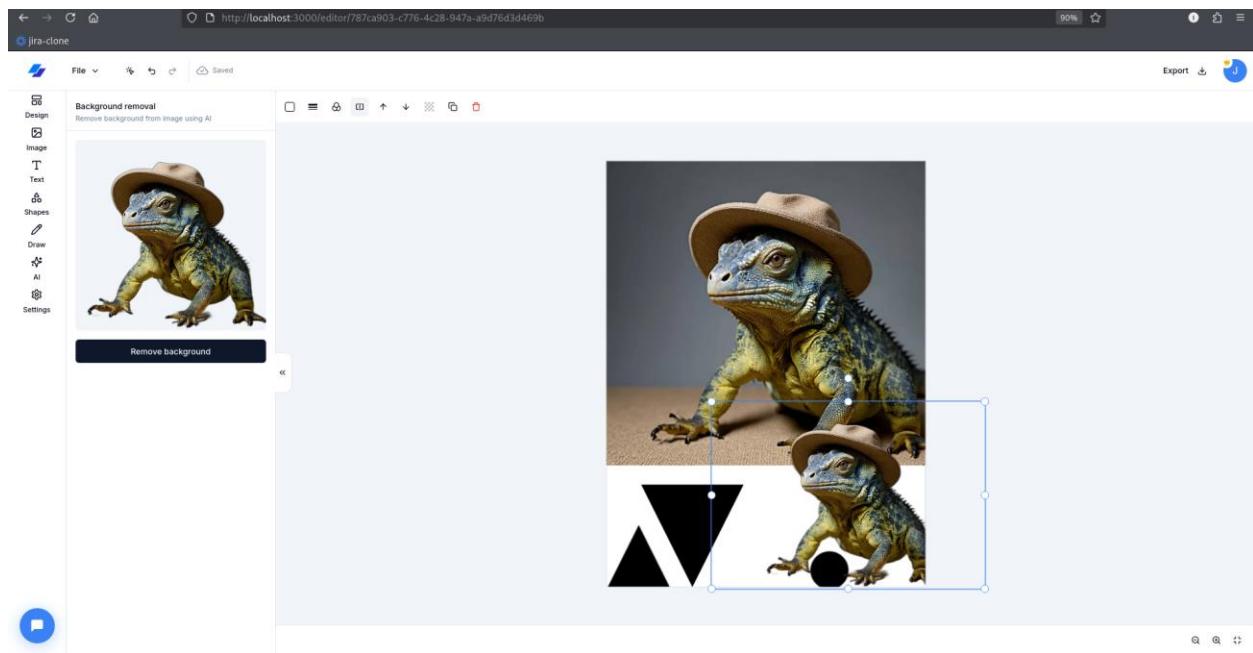


Fig.48: Background Removed

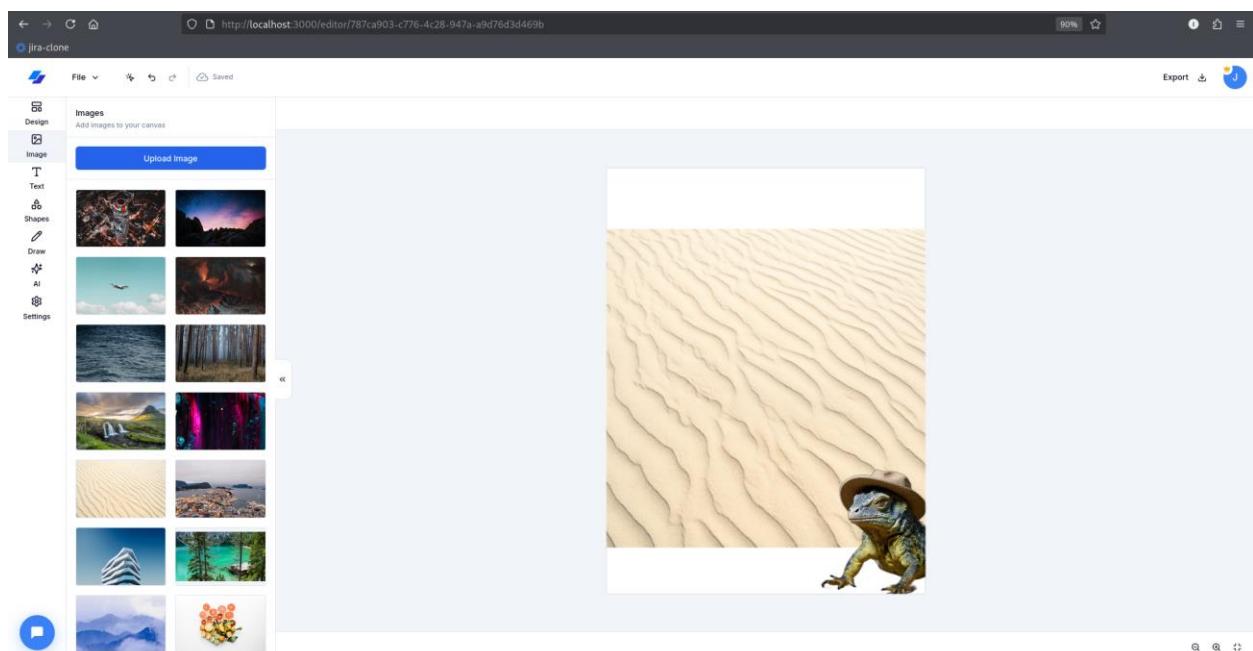


Fig.49: Go ahead and play

Now, let's create some templates.

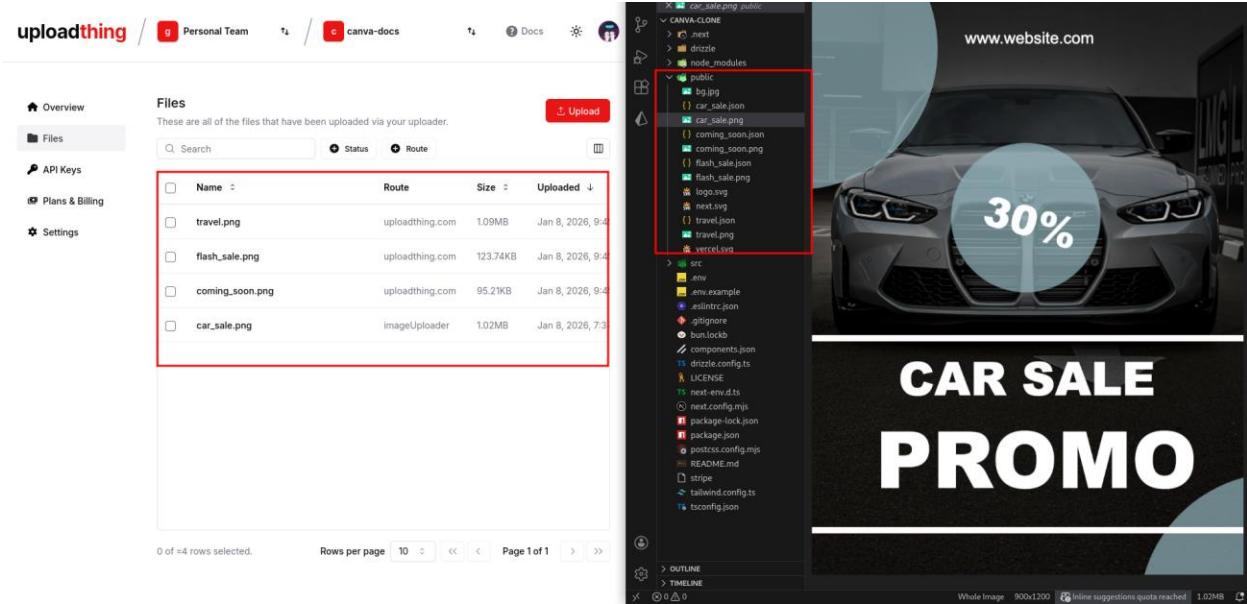


Fig.49: Upload the PNG images from the public

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.

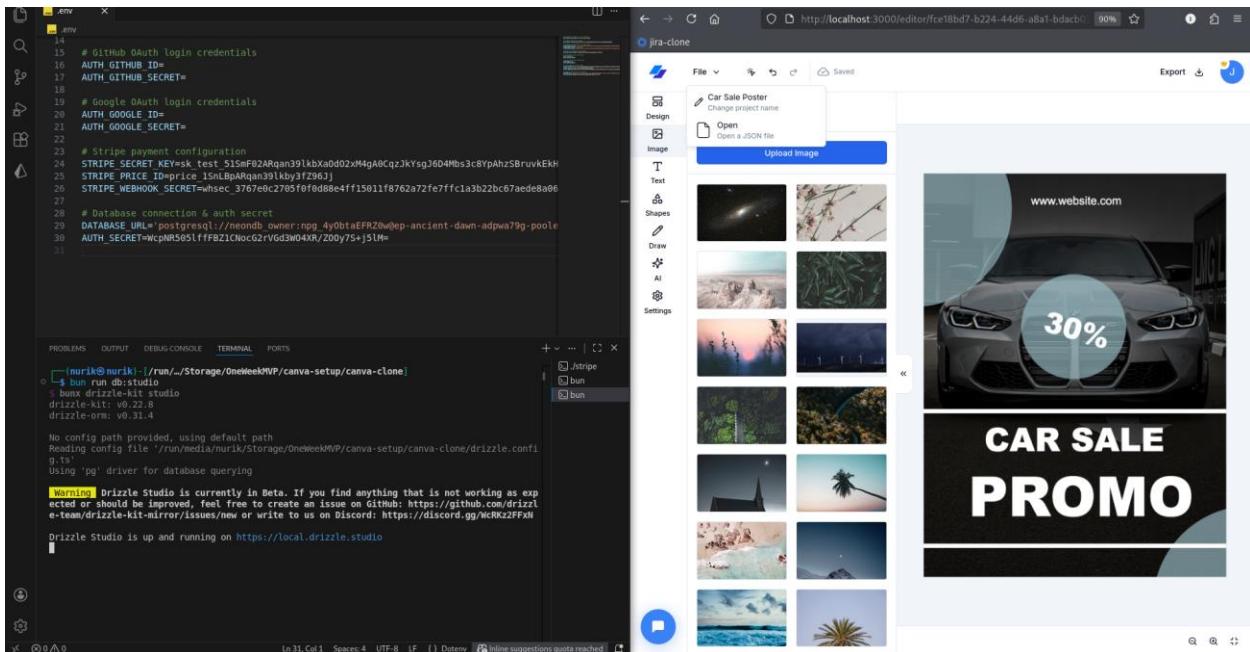


Fig: 50: Upload an image && go to your local drizzle studio

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

The screenshot displays two browser windows side-by-side. The left window is a Jira clone application showing a table with columns: h (integer), thumbnailUrl (text), isTemplate (boolean), isPro (boolean), and createdAt (timestamp). A specific row is highlighted in yellow, containing the URL <https://3in8qa725u.ufs.sh/>, a FALSE value for isTemplate, and a TRUE value for isPro. The right window is an uploadthing.com dashboard showing a list of uploaded files. One file, "car\_sale.png", has its URL (<https://3in8qa725u.ufs.sh/f/YqKnpNMjytG8Q8yhl0SDlBAbqe9tviEz6ZuyJR131MCw>), route ("imageUploader"), size (1.02MB), and upload date (Jan 8, 2026, 9:51) listed. A red arrow points from the highlighted row in the Jira table to the file list in the uploadthing dashboard.

Fig. 51: Make the changes && Save

The screenshot shows a terminal window on the left and a browser window on the right. The terminal window displays a portion of a `next.config.mjs` file with the following code:

```

1 // @type {import('next').NextConfig}
2 const nextConfig = {
3   images: {
4     remotePatterns: [
5       {
6         protocol: "https",
7         hostname: "images.unsplash.com",
8       },
9       {
10         protocol: "https",
11         hostname: "ufs.io",
12       },
13       {
14         protocol: "https",
15         hostname: "replicate.delivery",
16       },
17       {
18         protocol: "https",
19         hostname: "3in8qa725u.ufs.sh",
20       },
21     ],
22   },
23 }
24 export default nextConfig;

```

A red arrow points from the line `hostname: "3in8qa725u.ufs.sh",` in the terminal to the same line in the browser's error message. The browser window shows an 'Unhandled Runtime Error' with the message:

Error: Invalid src prop (<https://3in8qa725u.ufs.sh/f/YqKnpNMjytG8Q8yhl0SDlBAbqe9tviEz6ZuyJR131MCw>) on 'next/image', hostname "<https://3in8qa725u.ufs.sh>" is not configured under images in your 'next.config.js'. See more info: <https://nextjs.org/docs/messages/next-image-unconfigured-host>

The browser also shows a 'Call Stack' section with several entries related to webpack and React.

Fig.51: Error? No Problem!

The terminal window shows the following command and output:

```
UPD: change the hostname to '*.ufs.sh'.
[nurik@nurik:~/run/_Storage/OneWeekMVP/canca-setup/canca-clone] $ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Writing objects: 100% (6/6), 596 bytes | 596.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
remote: GitHub found 26 vulnerabilities on OneWeekMVP/canca-clone's default branch (2 critical, 7 high, 13 moderate, 4 low). To find out more, visit:
remote:   https://github.com/OneWeekMVP/canca-clone/security/dependabot
remote:
To https://github.com/OneWeekMVP/canca-clone.git
 4859922..d597e29  main -> main
[nurik@nurik:~/run/_Storage/OneWeekMVP/canca-setup/canca-clone]
```

The browser screenshot shows the 'The Canvas' app interface with a template card for a 'Car Sale Poster'.

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.

The terminal window shows the following command and output:

```
[nurik@nurik:~/run/_Storage/OneWeekMVP/canca-setup/canca-clone] $ bun dev
✓ Compiled _error in 5.6s (2011 modules)
GET /api/subscriptions/current 200 in 7832ms
GET /api/projects?limit=5 200 in 783ms
GET /api/projects?page=1&limit=5 200 in 7818ms
GET /api/projects/templates?page=1&limit=4 200 in 7818ms
GET /?rscxxzok 200 in 105ms
GET /?rscxxzok 200 in 88ms
GET /api/projects?limit=5 200 in 6525ms
GET /api/users/current 200 in 1460ms
GET /api/projects/templates?page=1&limit=4 200 in 1478ms
GET /api/projects?page=1&limit=5 200 in 1480ms
GET /api/subscriptions/current 200 in 1724ms
GET /api/auth/session 200 in 146ms
GET /api/projects/templates?page=1&limit=4 200 in 884ms
GET /api/subscriptions/current 200 in 890ms
GET /api/users/current 200 in 90ms
GET /api/projects?page=1&limit=5 200 in 912ms
```

The browser screenshot shows the 'The Canvas' app interface with a template card for a 'Car Sale Poster'.

Fig.52: The template is there

Repeat the process for the other images too.

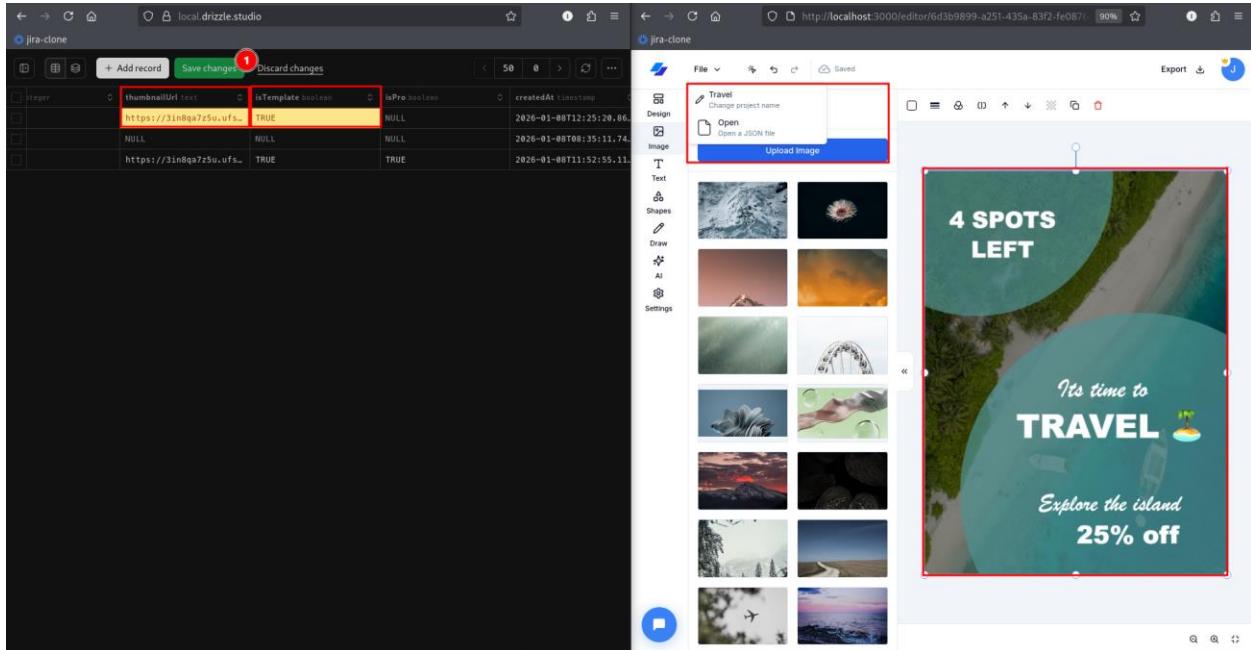


Fig.53: Repeat the process & Templates are set!

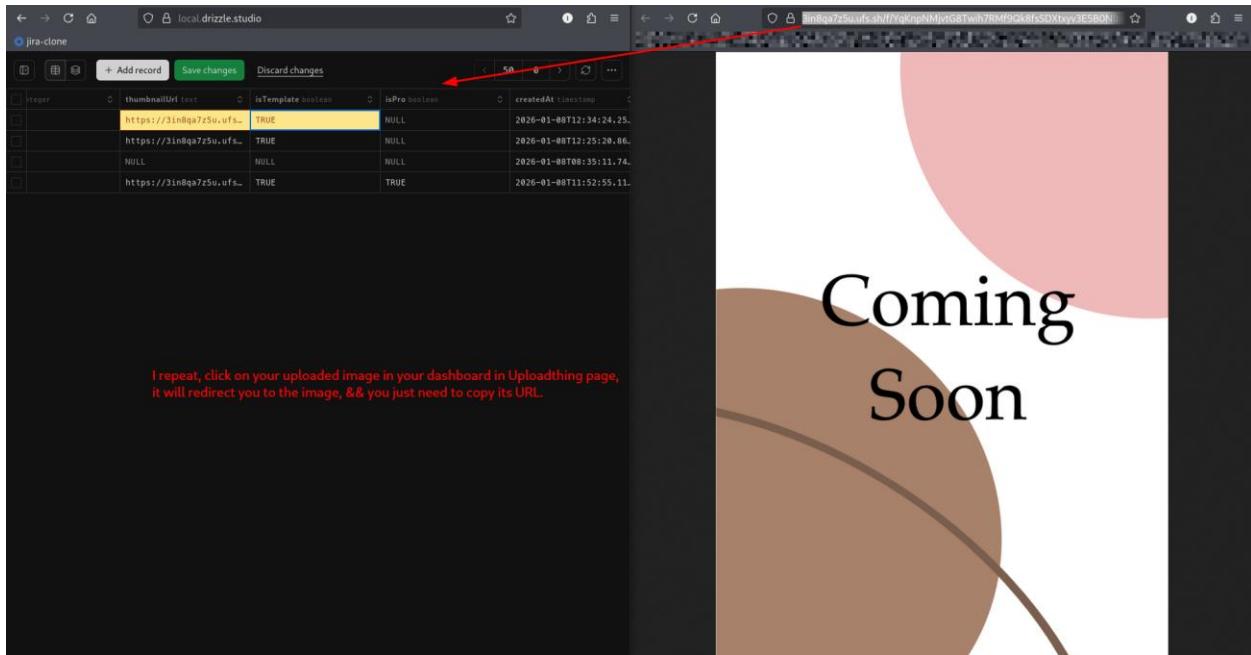


Fig.54: Copy pass the URL by pressing on the image in Uploadthing page

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>

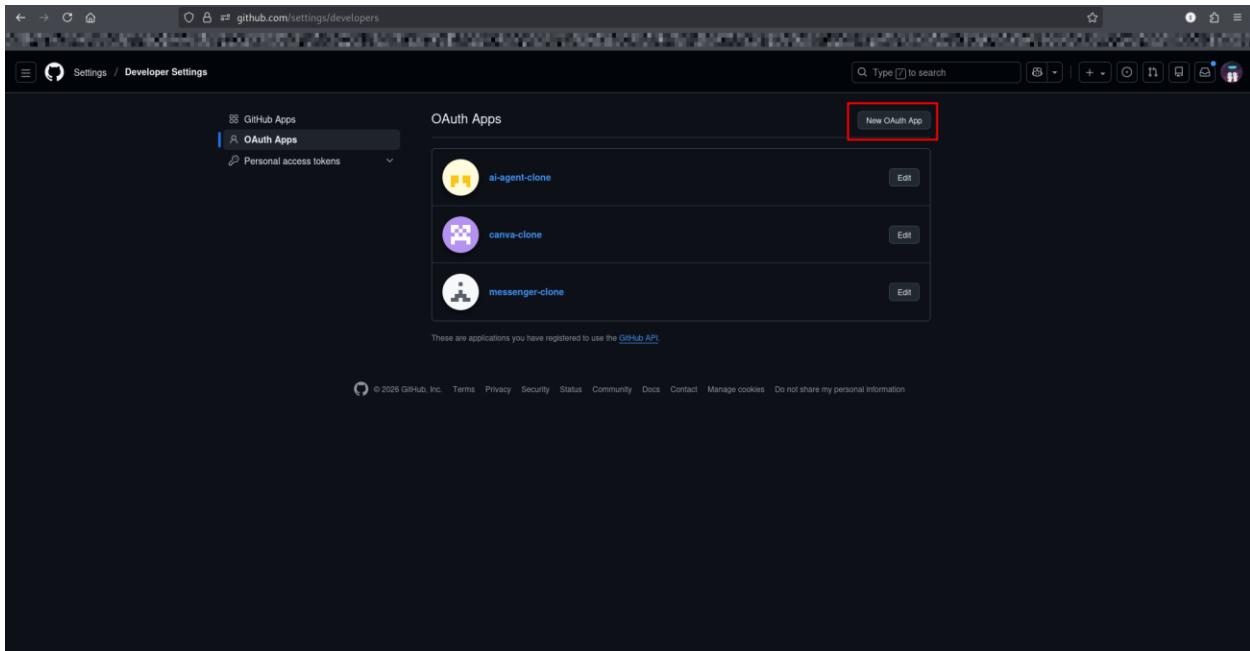


Fig.55: Go to your Github && create a new OAuth App

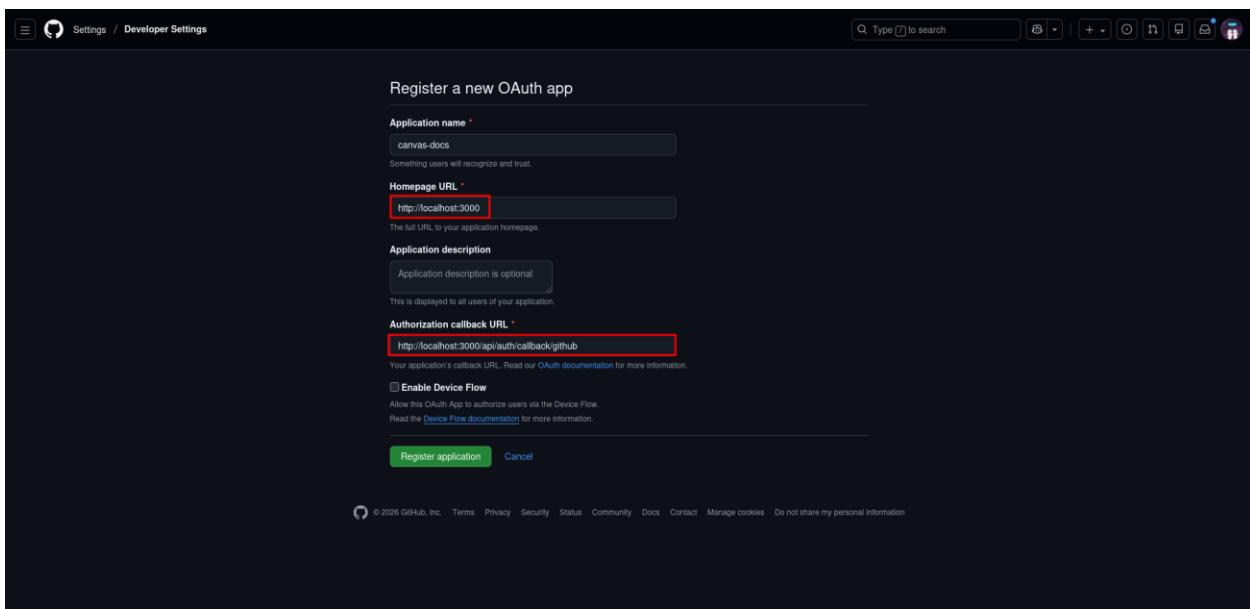


Fig.56: Pass the URL as shown

Homepage URL:

<http://localhost:3000>

Authorization callback URL

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

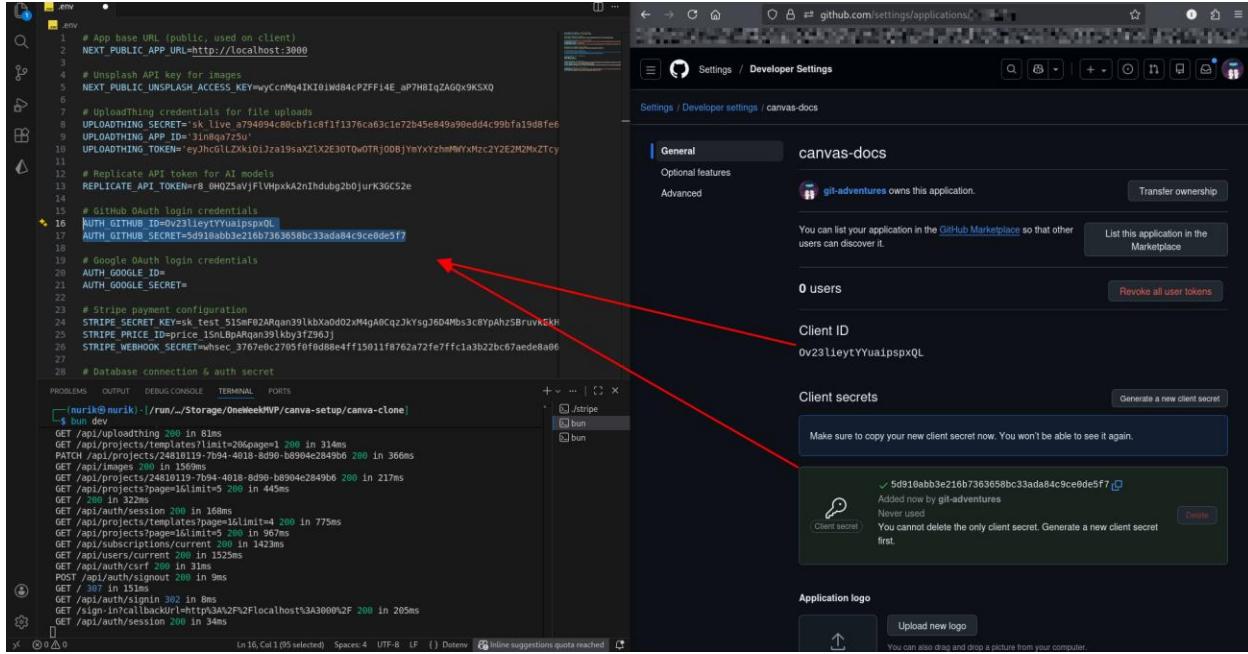


Fig.57: Github is set for localhost:3000

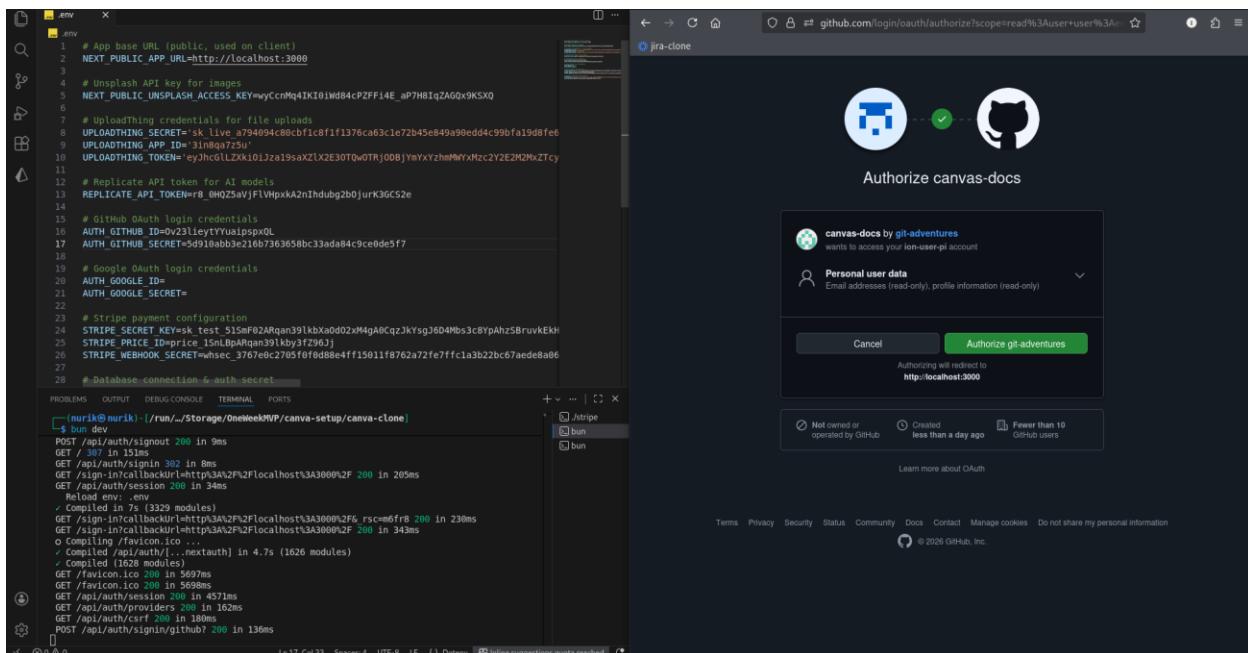


Fig.58: Github works!

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

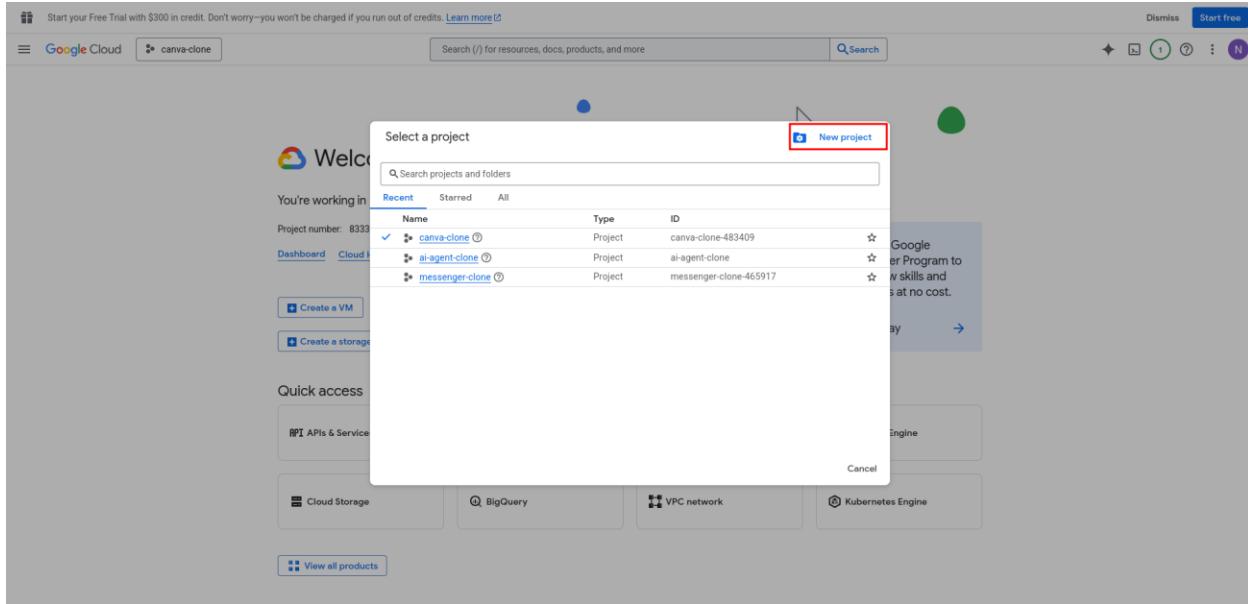


Fig. 59: Project Creation

This screenshot shows the 'New Project' creation process. On the left, a form is filled out with 'Project name' set to 'canva-docs' and 'Location' set to 'No organization'. On the right, the main dashboard shows a 'Notifications' sidebar with a red box highlighting the 'Select Project' link next to a notification about creating a project named 'canva-docs'. The main dashboard also displays the 'Welcome' screen with project details and various quick access links.

Fig. 60: Name your project & Select

The screenshot shows the Google Cloud Platform dashboard for a project named 'canva-docs'. On the left, there's a 'Welcome' section with a 'Create a VM' button. Below it is a 'Quick access' sidebar with several buttons: 'APIs & Services' (which is highlighted with a red box), 'IAM & Admin', 'Billing', 'Compute Engine', 'Cloud Storage', and 'BigQuery'. On the right, the 'APIs & Services' section is open, showing 'Enabled APIs & services' (with 'Library' and 'Credentials' selected), 'OAuth consent screen' (with a yellow warning box containing the text 'Remember to configure the OAuth consent screen with information about your application.' and a 'Configure consent screen' button), 'API Keys', 'OAuth 2.0 Client IDs', and 'Service Accounts'.

Fig. 61: Head to APIs && Configure

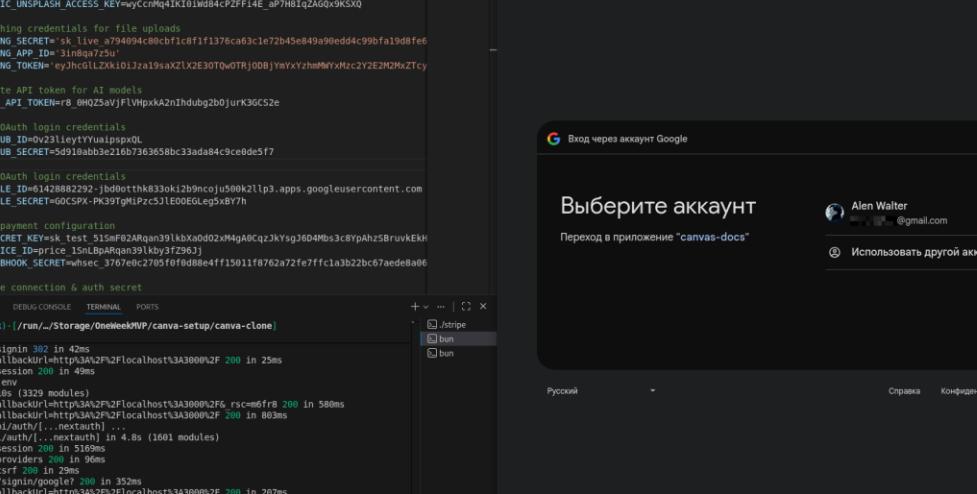
The screenshot shows the 'Google Auth Platform / Overview' page. On the left, there's a sidebar with options: 'Overview' (selected and highlighted with a blue box), 'Branding', 'Audience', 'Clients', 'Data Access', 'Verification Center', and 'Settings'. The main area displays a dashed cloud icon and a message: 'Google Auth Platform not configured yet. Get started to configure your application's identity and manage credentials for calling Google APIs and Sign in with Google. [Learn more](#)'. A 'Get started' button is at the bottom. To the right, there's a 'Create branding' wizard with three steps: 'Project configuration' (selected), 'App Information' (with 'External' selected), and 'Contact Information'. Each step has a 'Next' button. At the bottom, there are 'Create' and 'Cancel' buttons.

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.



```
env
1 # App base URL (public, used on client)
2 NEXT_PUBLIC_APP_URL=http://localhost:3000
3
4 # Unsplash API key for images
5 NEXT_PUBLIC_UNSPLASH_ACCESS_KEY=wyCcnMq4IKI0Iw84cPZFFl4E_aP7H8IqZAG0x9KSQ
6
7 # UploadThing credentials for file uploads
8 UPLOADTHING_SECRET='sk_live_1c8ff1376ca63e1e72b45e849a90edd4c99bf19d8fe6
9 UPLOADTHING_APP_ID='3in8qup7z5u'
10 UPLOADTHING_TOKEN='eyJhbGkIZXK1012ja19saZLX2E30T0wOTRj00BjYmXYZhM0YxMzc2Y2E2M2MxZTc
11
12 # Replicate API token for AI models
13 REPLICATE_API_TOKEN=r_0H0ZsAVfVHpxkA2n1hdubgzb0jurK30CS2e
14
15 # GitHub OAuth login credentials
16 AUTH_GITHUB_ID=0v231leytYuuipsx0L
17 AUTH_GITHUB_SECRET=5a910ab3b2e16b7363658bc33ada84c9ce0de5f7
18
19 # Google OAuth login credentials
20 AUTH_GOOGLE_ID=10d142882292-jbd00tth830ok1b9ncoju500k2llp3.apps.googleusercontent.com
21 AUTH_GOOGLE_SECRET=gCSPx-PK397ghlPz5jLE00EGLeg5y7h
22
23 # Stripe payment configuration
24 STRIPE_SECRET='sk_test_51mF02ARqan39jkXad0d2xM4gA0cQzjkYjs6G4Dms38YpAh2SzvukEkH
25 STRIPE_PRICE_ID=price_1kBy37296j
26 STRIPE_WEBHOOK_SECRET='whsec_3776e027b30f0bd8e4ff15011f8762a72fe7fffc1a3b22b7aeadea86
27
28 # Database connection & auth secret
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
nurik@nurik | /run/_Storage/OneWeekWPF/canca-setup/canca-clone |
+ bun dev
GET /auth/signin 302 in 42ms
GET /sign-in/callbackUrl=https%3A%2F%2Flocalhost%3A3000%2F 200 in 25ms
GET /api/auth/session 200 in 49ms
Reload env: .env
Compiled in 18s (3329 modules)
GET /api/auth/callbackUrl=https%3A%2F%2Flocalhost%3A3000%2F 200 in 580ms
GET /sign-in/callbackUrl=https%3A%2F%2Flocalhost%3A3000%2F 200 in 863ms
o Compiling /api/auth/...nextauth...
✓ compiled /api/auth/...nextauth... in 4.8s (1601 modules)
GET /api/auth/session 200 in 5169ms
GET /api/auth/session 200 in 49ms
GET /api/auth/csrf 200 in 29ms
POST /api/auth/signin/google/ 200 in 352ms
GET /sign-in/callbackUrl=https%3A%2F%2Flocalhost%3A3000%2F 200 in 207ms
GET /api/auth/session 200 in 45ms
GET /api/auth/protection 200 in 40ms
GET /api/auth/csrf 200 in 15ms
POST /api/auth/signin/google/ 200 in 299ms
```

Русский Справка Конфиденциальность Условия

Fig.64: There it is!

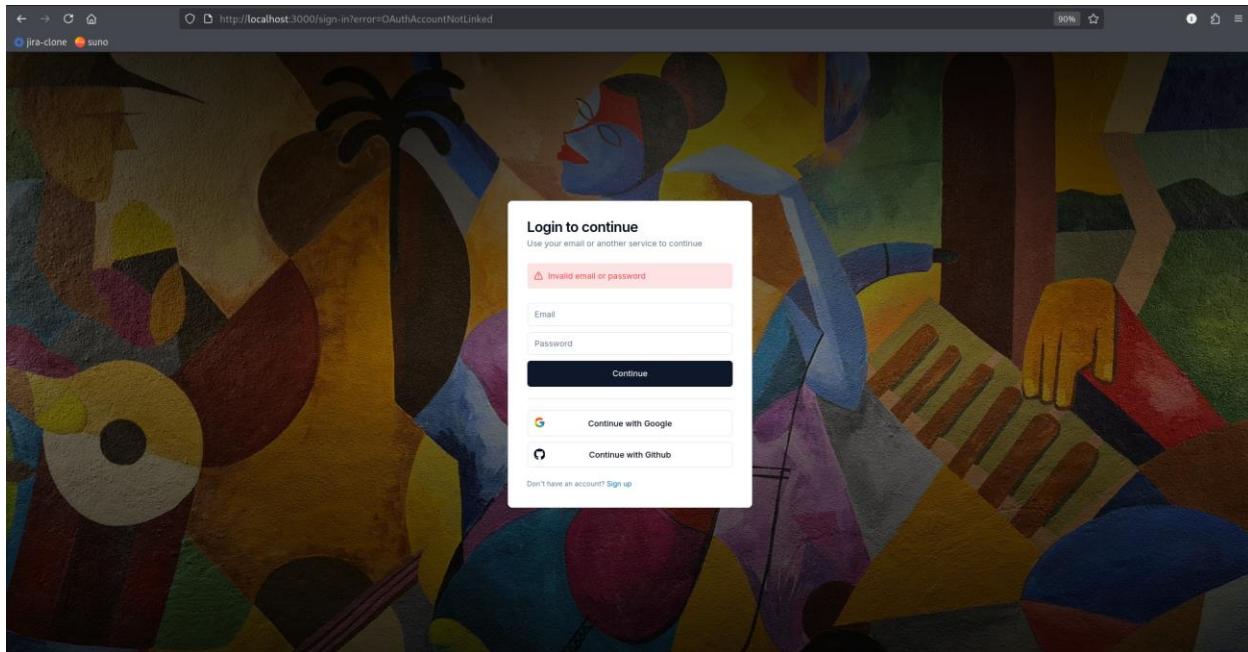


Fig.65: You can't log in with the same account from different OAuths

Finally, let's deploy.

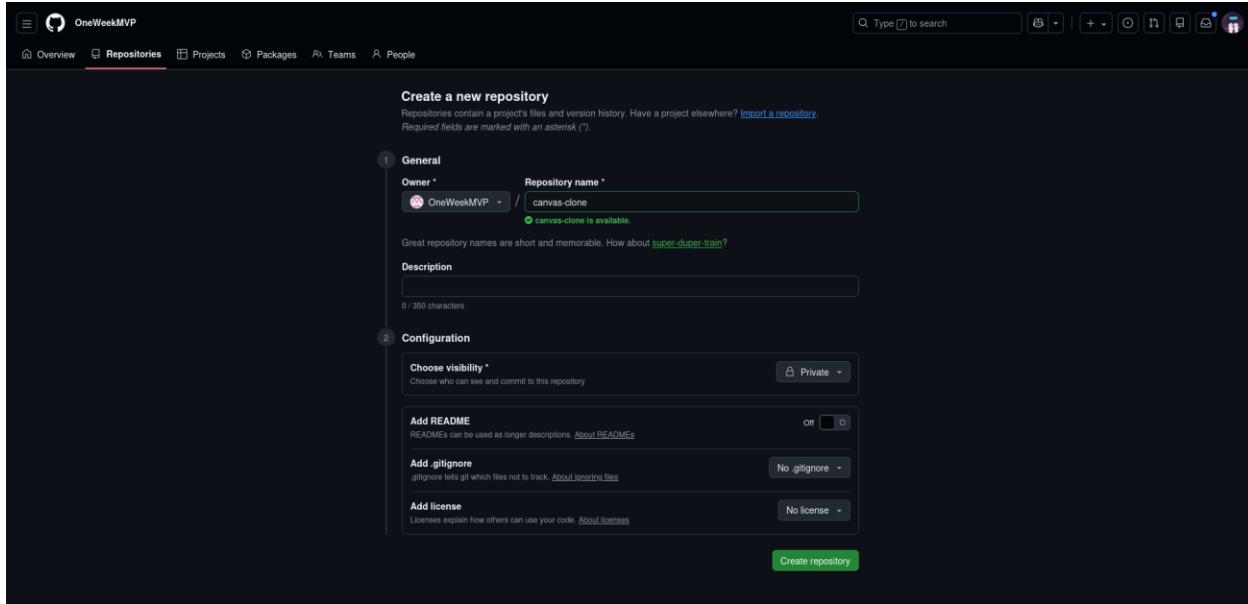


Fig.66: Create a repository

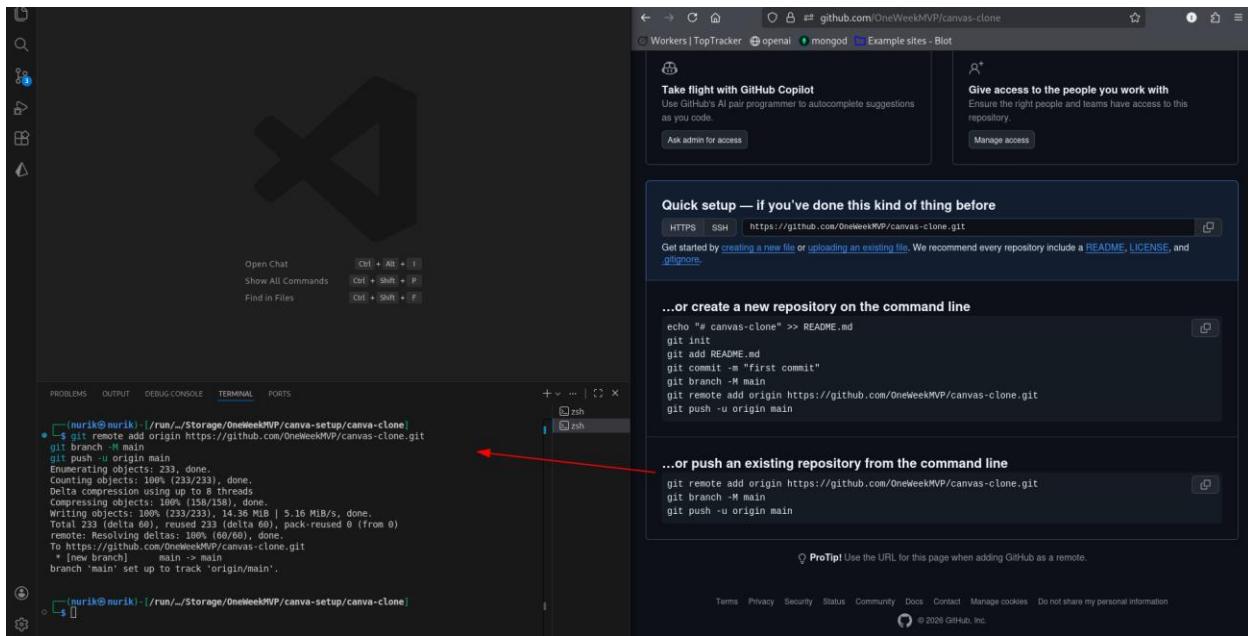


Fig.67: Git push

Refresh your Github to see your repository inside.

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

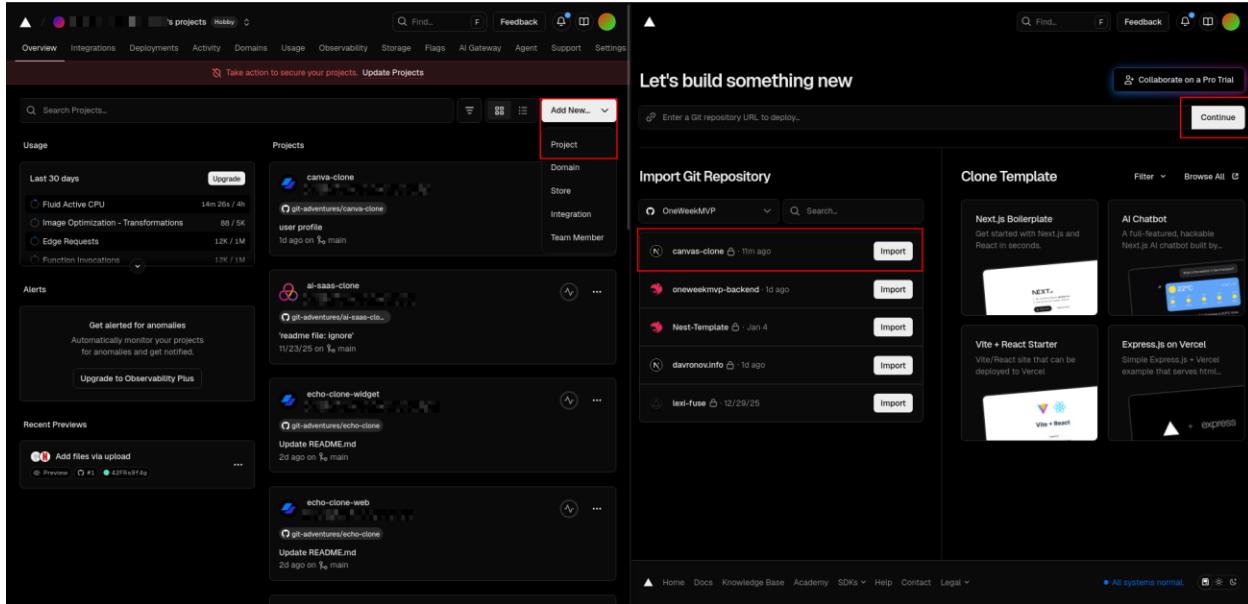


Fig.68: Import your repository from Gihub

This screenshot shows the Vercel deployment interface. On the left, a code editor displays a .env file with various environment variables. On the right, the deployment configuration screen shows a 'Project Name' field set to 'canvas-clone', a 'Framework Preset' of 'Next.js', and an 'Environment Variables' section. One variable, 'EXAMPLE\_NAME', is highlighted with a red box. Below it is a 'Deploy' button.

**.env**

```

1 # App base URL (publicly used on client)
2 NEXT_PUBLIC_APP_URL=http://localhost:3000
3
4 # Unsplash API key for images
5 NEXT_PUBLIC_UNSPLASH_ACCESS_KEY=ycrmq4tX101w84cPZFfI4E_aP7H0tqZAG0x9K5X
6
7 # UploadThing credentials for file uploads
8 UPLOADTHING_SECRET=sk_live_a794094c80cbfc8ff1f376ca3c1e72b5e849a99ed4c99bf1a19d8fe6
9 UPLOADTHING_APP_ID=3in8q07z5u
10 UPLOADTHING_TOKEN=eyJhbGkIZXk10Ijza19saZlX2E3OTw0TRjD8YmYxzhMwYxHzc2YzE2M2HsZTc
11
12 # Replicate API token for AI models
13 REPLICATE_API_TOKEN=r_0H0Z5avfUWpxxA2nIdubg2b0jurK3gC52e
14
15 # GitHub OAuth login credentials
16 AUTH_GITHUB_ID=0v231leytYuippxQL
17 AUTH_GITHUB_SECRET=5d910abb3e216b7363658bc33ada84c9ce0de517
18
19 # Google OAuth login credentials
20 AUTH_GOOGLE_ID=142882292-292-jbd0tthk83ok1i2b9ncoju580k21lp3.apps.googleusercontent.com
21 AUTH_GOOGLE_SECRET=gCSPx-Pk397ghIPz5Jle00EGLeg5xV7h
22
23 # Stripe payment integration
24 STRIPE_SECRET_KEY=sk_test_51mP02Rqan391kbXa0d02xH4gABCqz3kYsgJ6d4Ms3c8YpAhz5BrvukEkH
25 STRIPE_PRICE_ID=price_1SnBpAQRgan391kbj31296J
26 STRIPE_WEBHOOK_SECRET=whsec_37676c2705f0fd88e4ff15011f87e2a721e7ffC1a3b2b67acd886
27
28 # Database connection & auth secret
29 DATABASE_URL="postgres://neondb:owner:npq_4yObtaFRZ0w@ep-ancient-dawn-adpwa79-pool"
30 AUTH_SECRET=WcpNR5051fffB21hocG2rwd3Q0XR/Z0oY7s+51M
31

```

Here, all you need to do is to copy && pass all your .env content into the box in red. Happy deploying!

Fig.69: Happy Deploying!

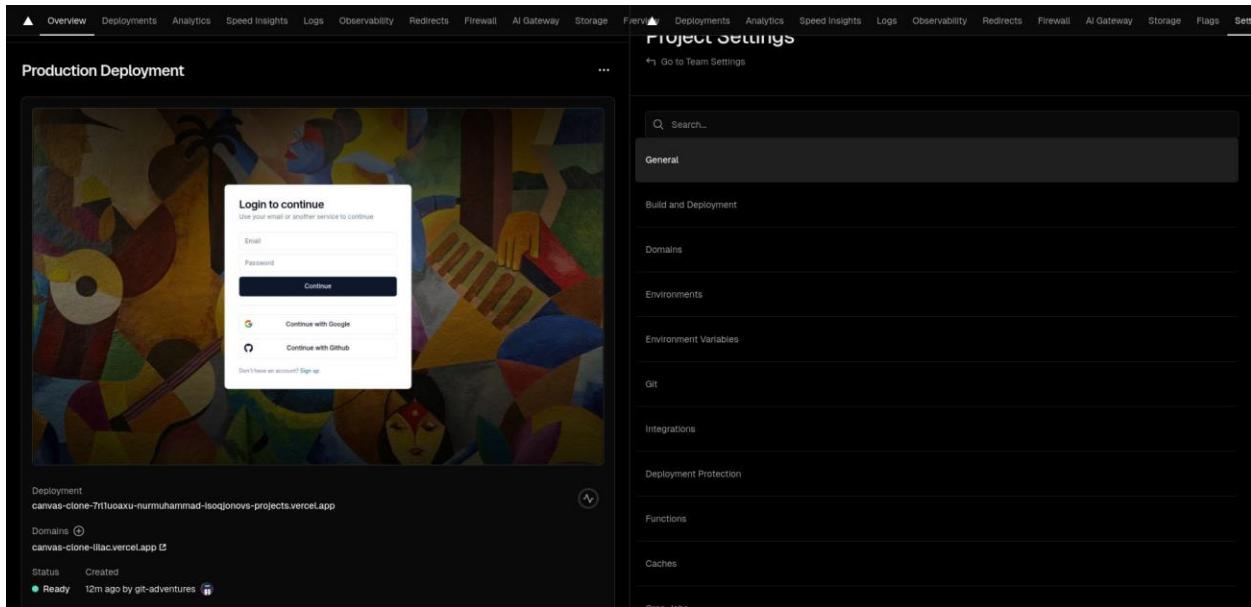


Fig.70: Copy your Domain name (URL) && got to Settings Environment Variables

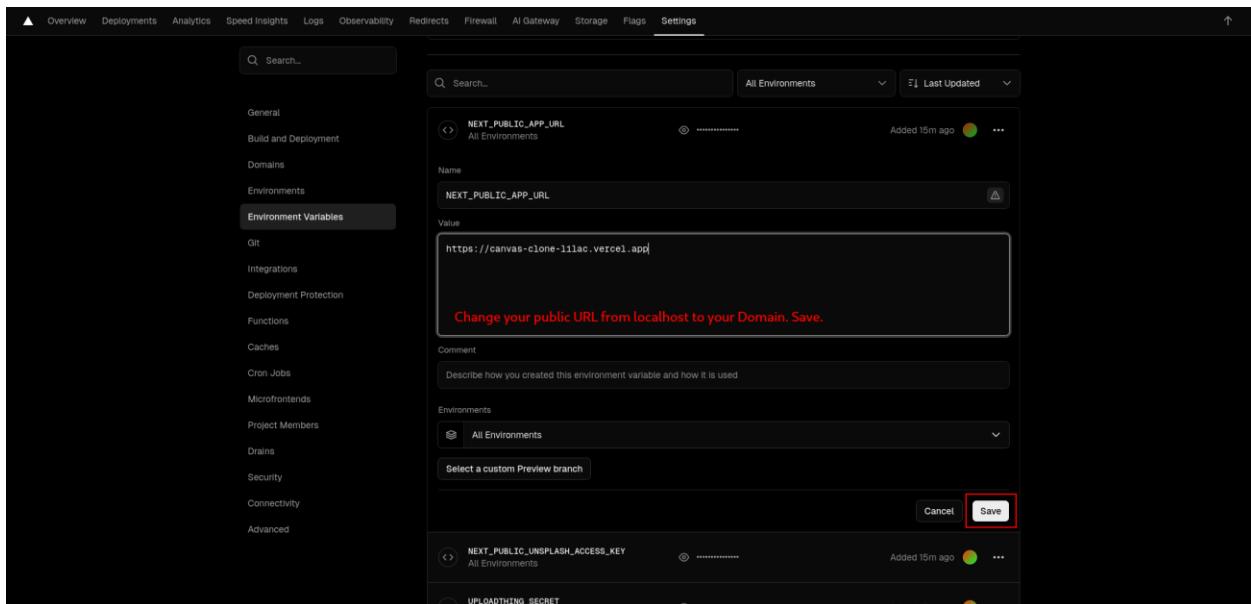


Fig.71: Save

The screenshot shows two side-by-side panels. The left panel is titled 'Monthly subscription' and includes sections for 'Trigger reactions in your integration with Stripe events' (with a red box around the '+ Add destination' button), 'Quickstart guide', and 'Receive Stripe events in your webhook endpoint'. The right panel is titled 'Create an event destination' and shows 'Your account' and 'Connected and v2 accounts' options. It also displays 'API version' (2025-12-15.clover), 'Events' (Selected events), and specific event types like 'Checkout.session.completed' and 'invoice.payment.succeeded' (both with checkboxes checked and red boxes around them). A note at the bottom says: 'In "All events" search for Checkout && Invoice, select the ones shown above and press continue.'

Fig.72: Stripe Events

This screenshot shows two steps of creating a webhook destination. The left step, 'Choose where you want to send events', offers 'Webhook endpoint' (selected) and 'Amazon EventBridge' options. The right step, 'Configure destination', sets 'Events from' to 'Your account', 'Payload style' to 'Snapshot', and 'API version' to '2025-12-15.clover'. It also includes fields for 'Destination name' (whimsical-serenity), 'Endpoint URL' (https://canvas-clone-filac.vercel.app/api/subscriptions/webhook), and 'Description'. A red box highlights the 'Endpoint URL' field.

Fig.73: Webhook Destination

The screenshot shows two main panels. On the left, the Vercel Workbench displays environment variables: `STRIPE_WEBHOOK_SECRET` (Value: `whsec_mJYJZPbysFxn61vfG1sxxxxcd8AFIZ90R`), `NEXT_PUBLIC_APP_URL` (Value: `https://canvas-clone-lilac.vercel.app`), `NEXT_PUBLIC_UNSPLASH_ACCESS_KEY` (Value: `Unset`), and `UPLOADTHING_SECRET` (Value: `Unset`). A blue notification box says "Updated Environment Variable successfully. A new deployment is needed for changes to take effect." On the right, the Sandboxed API interface shows a "whimsical-serenity" webhook with "Event deliveries" and "Destination details". A red arrow points from the "Event deliveries" section to the "Destination details" section.

Fig. 74: Save

The screenshot shows the "Client ID for Web application" configuration page in Google Cloud Platform. It includes sections for "Authorized JavaScript origins" (with `https://canvas-clone-lilac.vercel.app` listed) and "Authorized redirect URIs" (with `https://canvas-clone-lilac.vercel.app/api/auth/callback/google` listed). A note at the bottom says "Now, go back to console.cloud.google.com & github.com, change your localhost to your domain." The right side of the screen shows the "Application logo" and "Application name" fields, along with other configuration options like "Homepage URL" and "Authorization callback URL".

Fig. 75: Change the localhost to your domain name

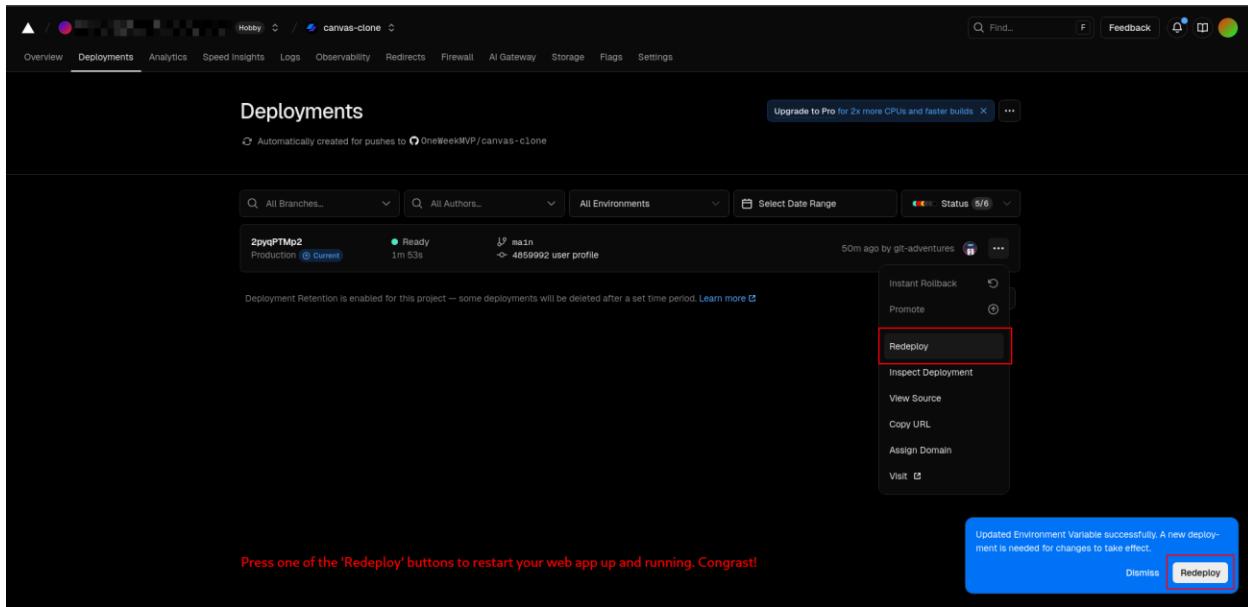


Fig.76: Redeploy

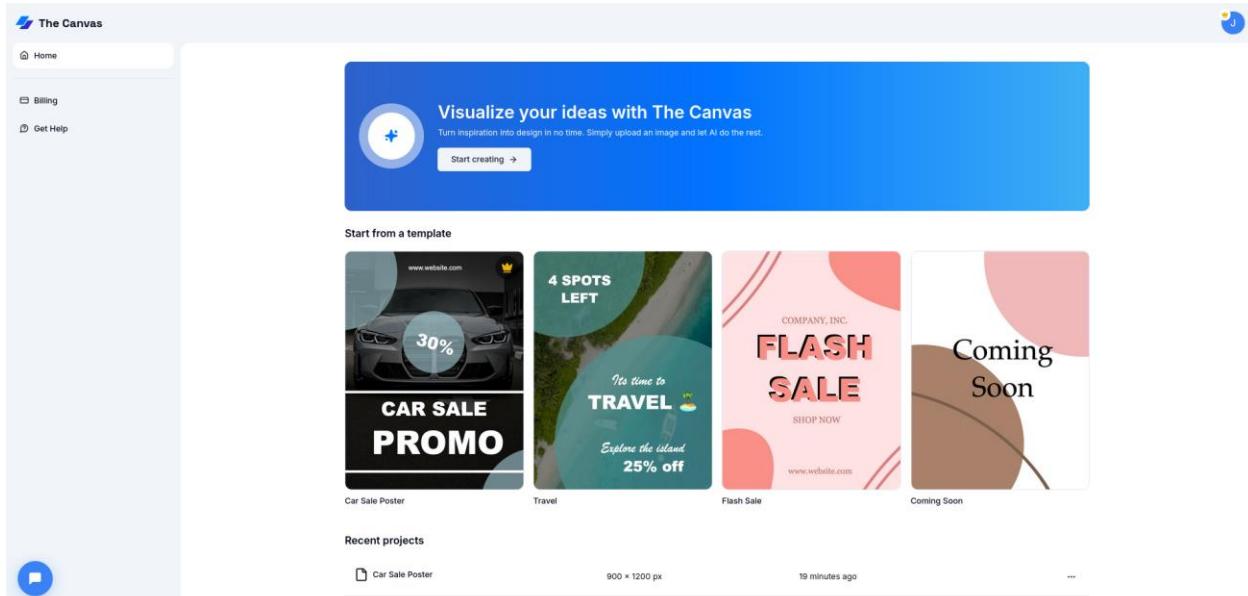


Fig.77: Canva Clone

**Congratulations!** ☐ Your Canva Clone is now fully configured and deployed!

## Reference:

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