

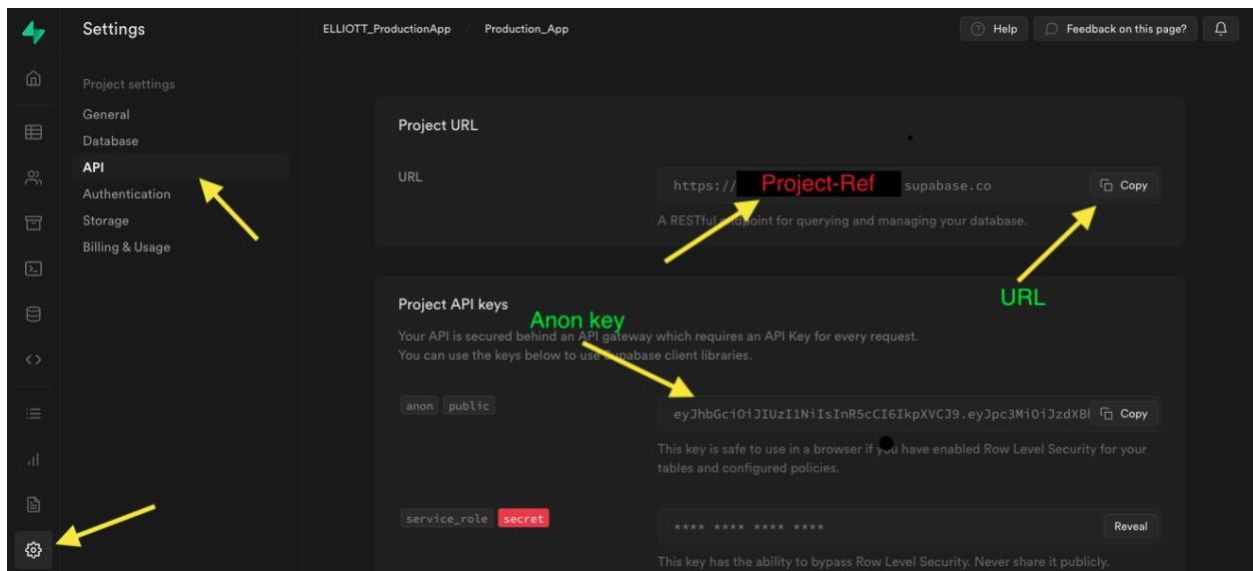
# HAPPY DAYS README GUIDE

## ACCOUNTS REQUIREMENTS

- Supabase free account
- Cloudflare workers
- GitHub

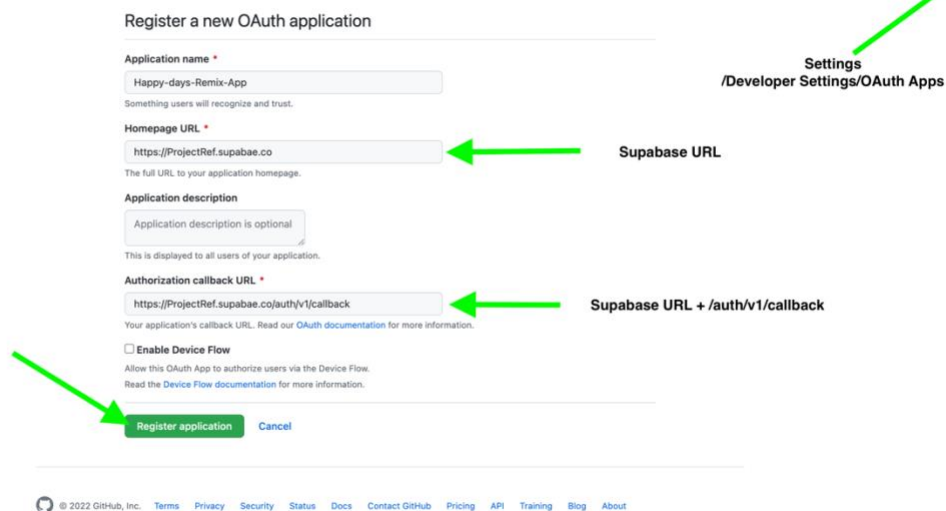
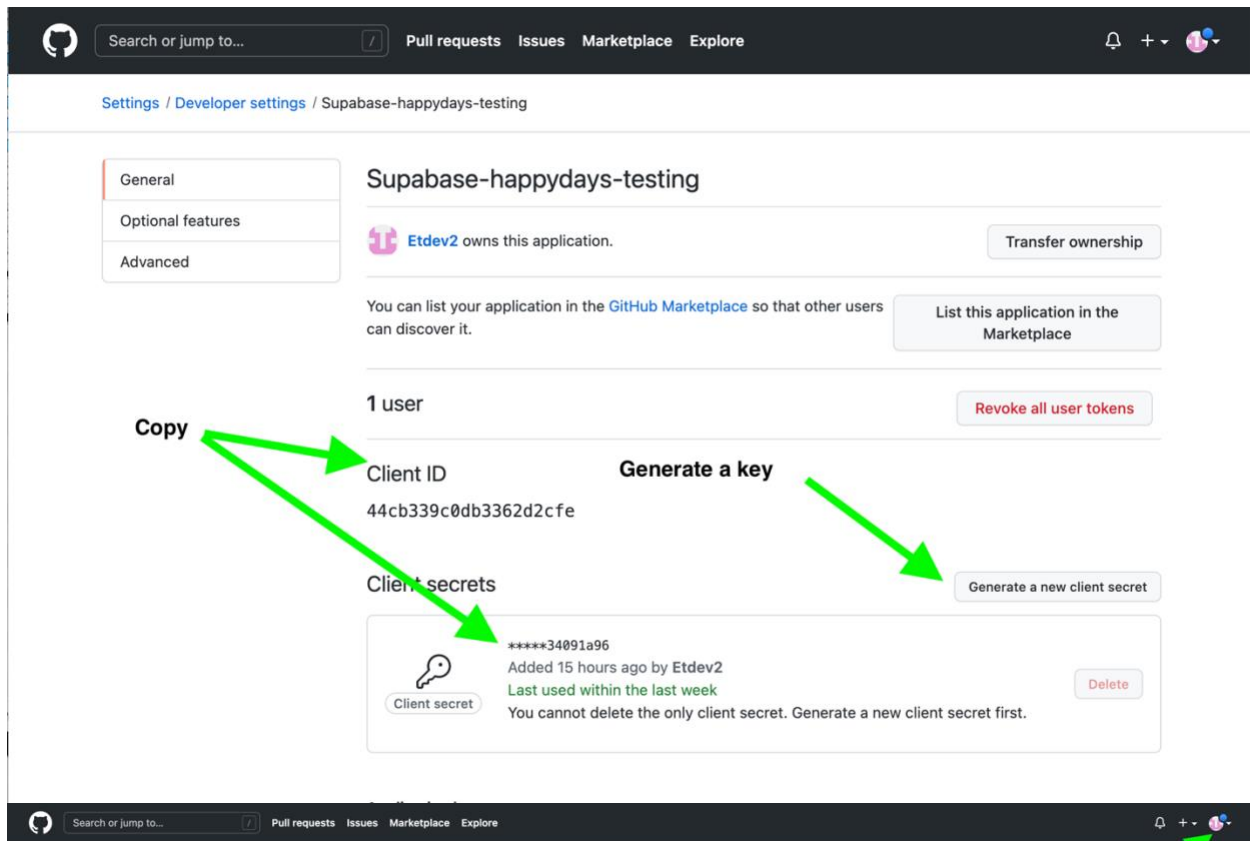
**SUPABASE REFERENCES** (Copy these to note pad will be used many times in the guide)  
(See screenshots below as a visual reference to locate)

- Supabase URL (supabase/settings/api)
- Supabase Anon Key (Supabase/setting/api)
- Supabase Project Ref (supabase/settings/api)



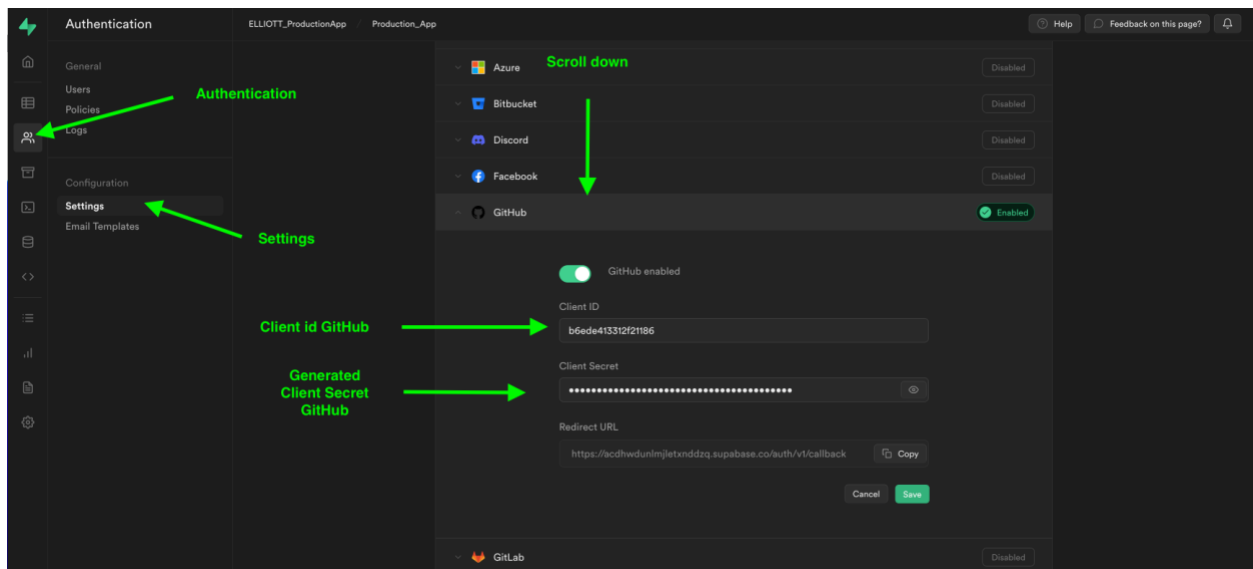
## SUPABASE GITHUB AUTH PROVIDER (GITHUB SETUP)

- In your GitHub profile, on the right side of the nav bar, go to Settings / Developer Settings / OAuth apps
- Register a New OAuth application
- Name: your choice
- Homepage URL = Supabase URL
- Authorization Callback = Supabase URL + / auth/v1/callback



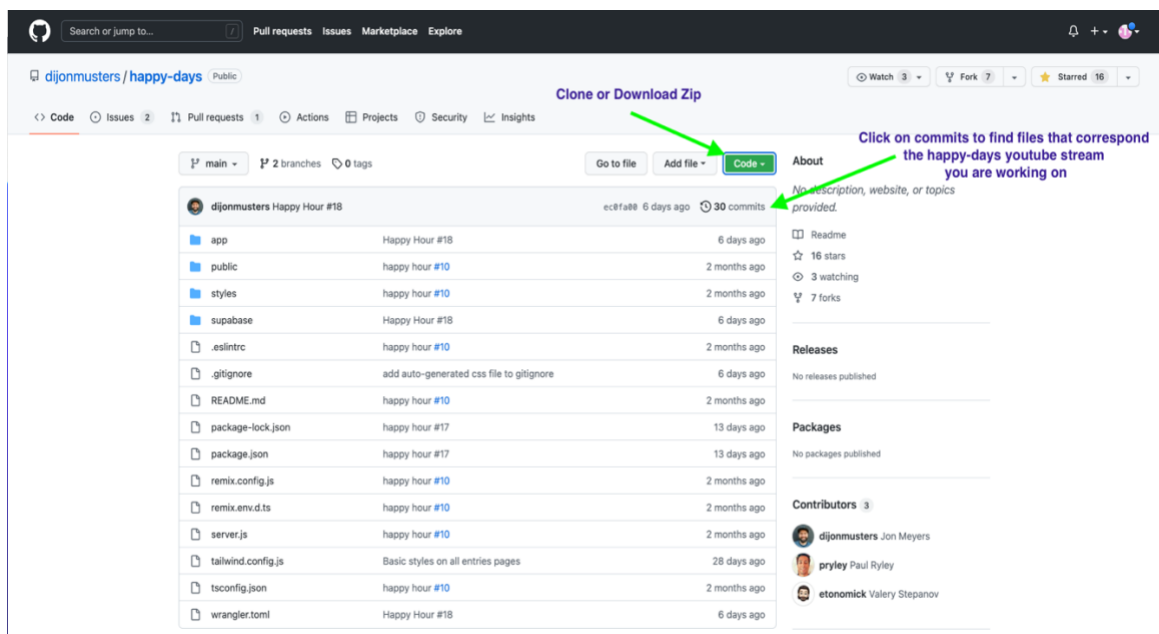
## SUPABASE GITHUB AUTH PROVIDER (SUPABASE SETUP)

- After setting up GitHub OAuth copy Client ID AND copy the generated client secret
- Got to Supabase Dashboard and got to pages Supabase/Authentication/Settings/ and scroll down to Auth Providers
- Enable GitHub, Paste in Client ID, and Client secret and save



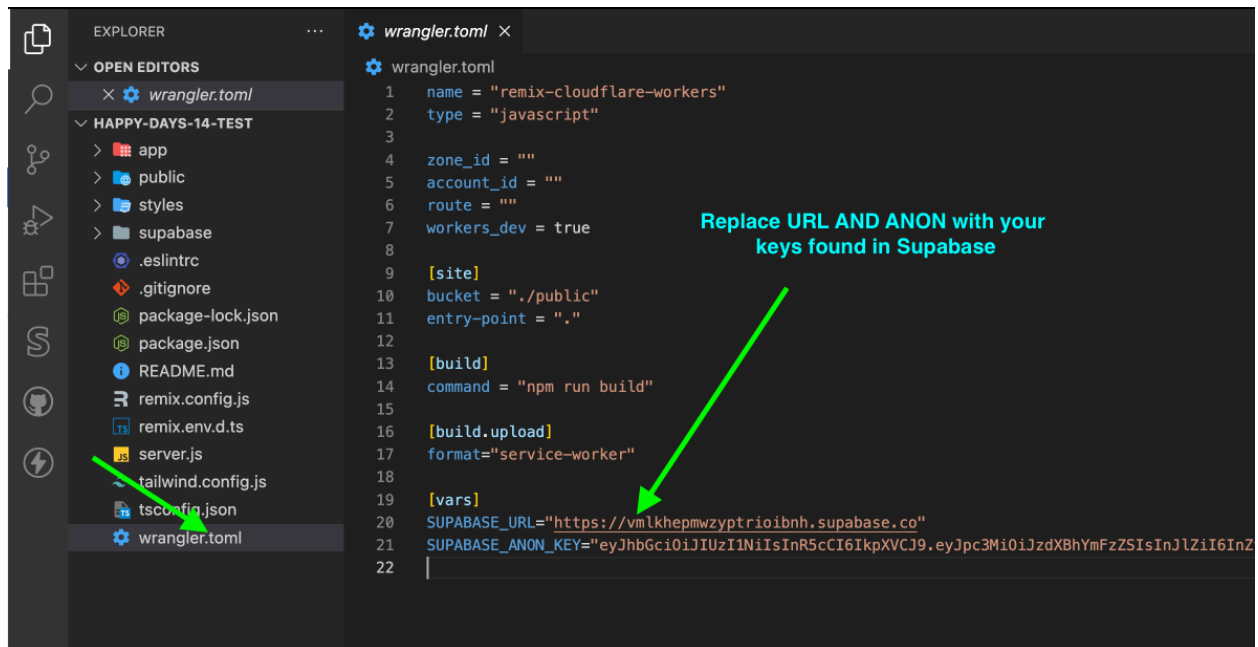
## CLONING PROJECT FROM GITHUB

- <https://github.com/dijonmusters/happy-days>



## OPEN IN VSCODE

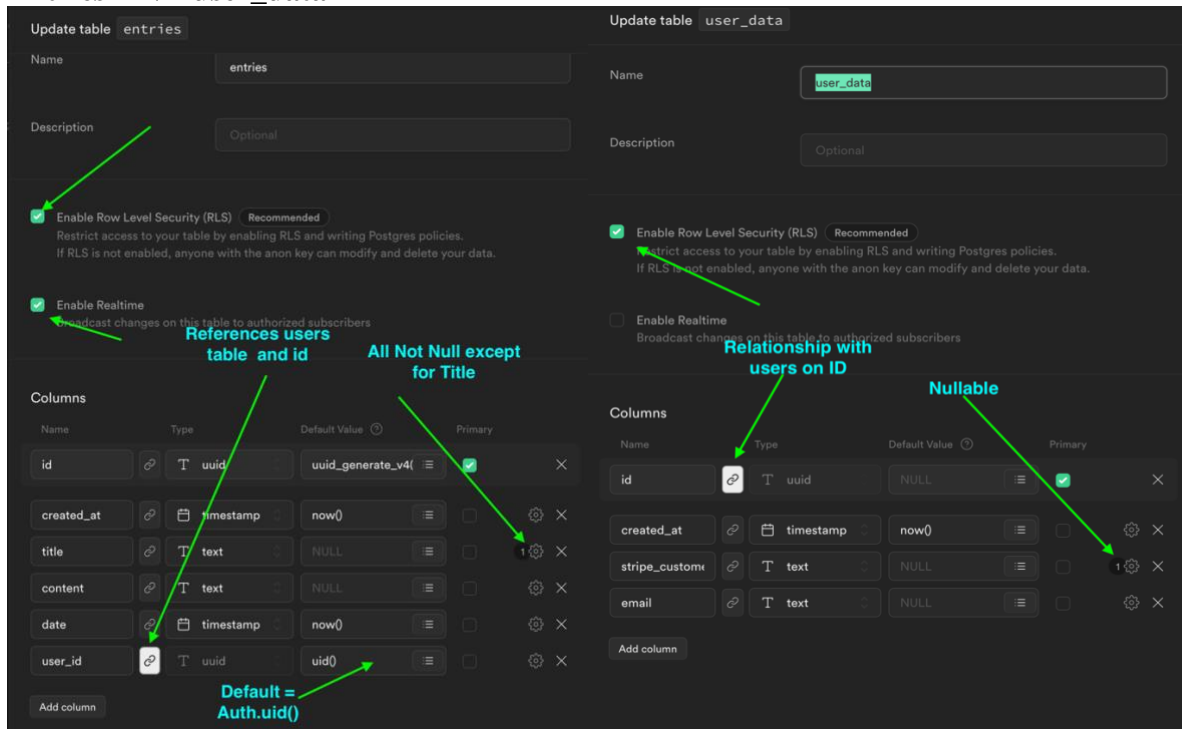
- In wrangler.toml under vars replace SUPABASE\_URL and SUPABASE\_ANON\_KEY with your superbase keys found in the dashboard.
- Continue with VS code and remix after the database is configured.



## SETTING UP DATABASE

- Create an **entries** table and **user\_data** table (Manually or with the SQL code bellow)

## Entries AND user\_data

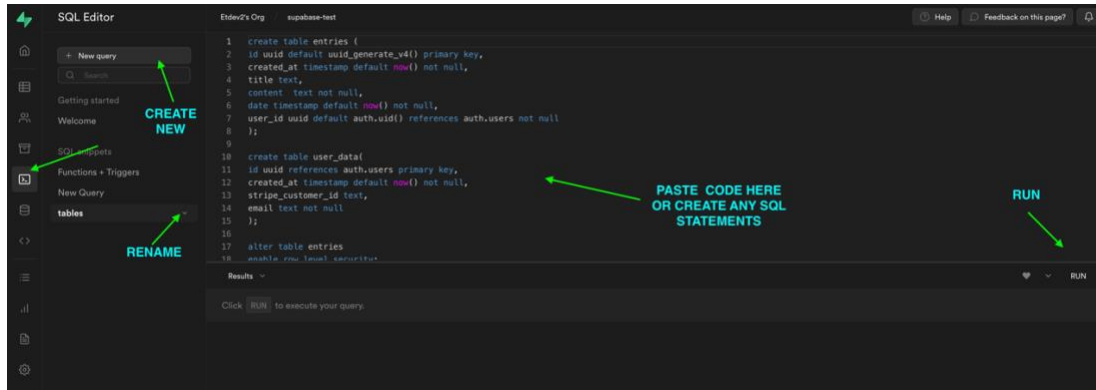


## SETUP SUPABASE WITH THE SQL CODE BELOW:(entries,user\_data)

```
create table entries (  
id uuid default uuid_generate_v4() primary key,  
created_at timestamp default now() not null,  
title text,  
content text not null,  
date timestamp default now() not null,  
user_id uuid default auth.uid() references auth.users not null  
);  
  
create table user_data(  
id uuid references auth.users primary key,  
created_at timestamp default now() not null,  
stripe_customer_id text,  
email text not null  
);  
  
alter table entries  
enable row level security;  
  
CREATE POLICY "Authenticated users can see their own entries" ON "public"."entries"  
AS PERMISSIVE FOR SELECT  
TO authenticated  
USING (user_id = auth.uid());  
  
CREATE POLICY "users can update their entry" ON "public"."entries"  
AS PERMISSIVE FOR UPDATE  
TO authenticated  
USING (user_id = auth.uid())  
WITH CHECK (user_id = auth.uid());  
  
CREATE POLICY "Authenticated users can insert own data" ON "public"."entries"  
AS PERMISSIVE FOR INSERT  
TO authenticated  
WITH CHECK (user_id = auth.uid());  
  
alter table user_data  
enable row level security;
```

## ENTERING CODE IN SQL EDITOR IN SUPABASE DASHBOARD

- Paste entries and user\_data into SQL code editor



## Finally, time to Remix!

- If you have not cloned the repository follow the directions above.
- Before starting make sure you completed (OAuth, Created tables (entries,user\_data, RLS)
- From the cloned repository replace SUPABASE\_KEY and Anon\_key in wangler.toml.
- RUN npm install to install packages, npm audit fix to install dependencies
- RUN **npm run dev to check if OAuth works**
- If a user is created the user will show up in the database in supabase/authentication/users.
- If the user is created continue to set up a function, that creates a user in the user\_data table, triggered by the login.

## SETTING UP FUNCTIONS AND TRIGGERS (MANUALLY )

- In the supabase dashboard go to Supabase/ Database / In the menu below, you will see Triggers / Functions and Database Webhooks.
- Create the **handle\_new\_users** function **first**
- After the function is created you can connect the **handle\_new\_users** function to the **on\_insert\_auth\_user** trigger. To connect the trigger-function use the drop-down menu “functions to trigger” in the trigger and you will see the function handle\_new\_users.

Add a new function

Return type

trigger

Arguments

Arguments can be referenced in the function body using either names or numbers.

+ Add a new argument

Definition

The language below should be written in 'plpgsql'.  
Change the language in the Advanced Settings below.

```

1
2
3
4
5
6
7
begin
insert into public.user_data(id,email)
values(new.id,new.email);
return new;
end;

```

plpgsql function that inserts id and email into user\_data table

Show advanced settings

These are settings that might be familiar for postgres heavy users

Language

plpgsql

Behavior

volatile

Config Params

+ Add a new config

Type of security

SECURITY INVOKER

Function is to be executed with the privileges of the user that calls it.

SECURITY DEFINER

Function is to be executed with the privileges of the user that created it.

Add a new Trigger

Name of trigger

on\_insert\_auth\_user

The name is also stored as the actual postgres name of the trigger. Do not use spaces/whitespace.

Conditions to fire trigger

Table

users

This is the table the trigger will watch for changes. You can only select 1 table for a trigger.

Events

The type of events that will trigger your trigger

☒ Insert  
Any insert operation on the table
 ☐ Update  
Any update operation, of any column in the table
 ☐ Delete  
Any deletion of a record

These are the events that are watched by the trigger, only the events selected above will fire the trigger on the table you've selected.

Trigger type

After the event

This determines when your Hook fires

Orientation

Row

Identifies whether the trigger fires once for each processed row or once for the entire table.

Function to trigger

> Choose a function to trigger

## SETTING UP FUNCTION + TIGGER IN SQL EDITOR

```

create function public.handle_new_user()
returns trigger as
$$
begin
insert into public.user_data(id,email)
values(new.id,new.email);
return new;
end;

$$

language plpgsql security definer;

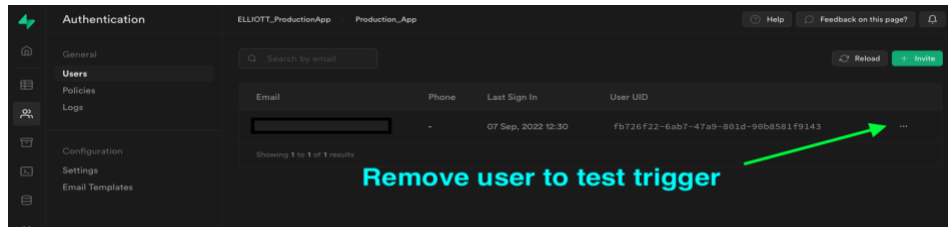
create trigger on_insert_auth_user
after insert on auth.users
for each row
execute procedure public.handle_new_user();

```

## IN VS CODE IN TERMAL

- **REMOVE** all the user data from the supabase including the user before testing the trigger

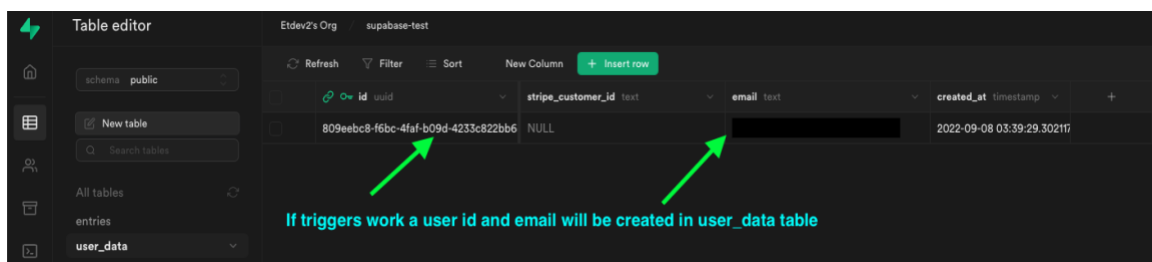
- 



- 

- **RUN** `npm run dev`

- If everything is working properly after signing in to the remix app. A user with `user_id` will be written into the `user_data` base with `stripe_customer` null.



- 

## SUPABASE EDGE FUNCTIONS AND WEBHOOKS

### SETUP STORAGE

- Create a new storage bucket named assets
- Allow all operations
- Added the below definitions
- `((bucket_id = 'assets'::text) AND ((uid())::text = (storage.foldername(name))[1]))`



### Adding new policy to assets

Policy name

Users can manage own files
26/50

A descriptive name for your policy

Allowed operation

Based on the operations you have selected, you can use the highlighted functions in the [client library](#).

☒ SELECT
☒ INSERT
☒ UPDATE
☒ DELETE

upload

download

list

update

move

copy

remove

createSignedUrl

createSignedUrls

getPublicUrl

Target roles

Defaults to all roles if none selected

Apply policy to the selected roles

Policy definition

Provide a SQL conditional expression that returns a boolean.

```

1  [(bucket_id = 'assets'::text) AND ((uid())::text = (storage.foldername(name))
    {1})]

```

View templates

Review

## SUPABASE EDGE FUNCTIONS AND SUPABASE CLI (Run Code In Vscode Termal Exclude **Run** In The Terminal)

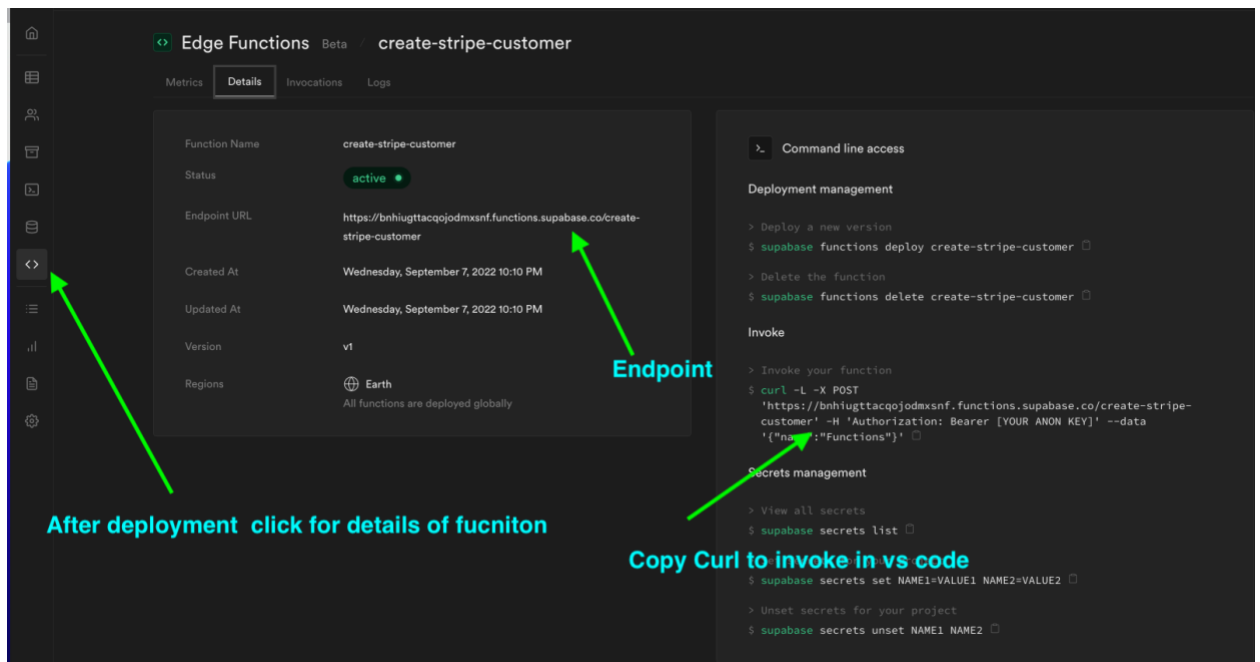
### STEPS TO THE EDGE

Follow steps below is more detailed may have messed around a few time to get everything working, It took me a few times to make sure everything was right and spelled correctly

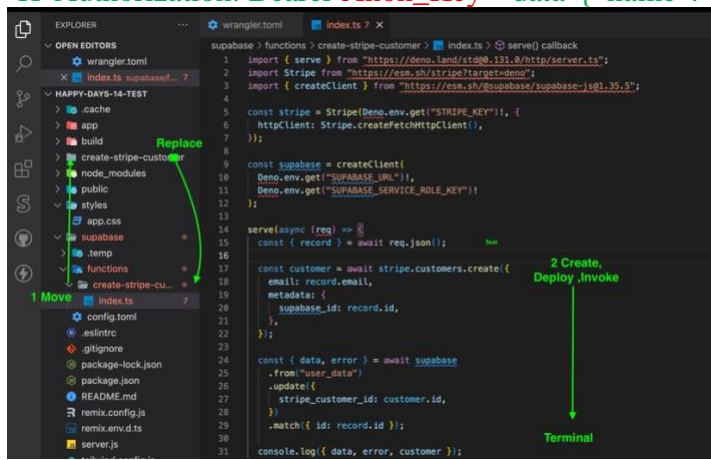
#### 1. Remove supabases file from clone (More detailed below)

2. **supabase init**
3. **supabase link**
4. **supabase functions new create-stripe-customer**
5. **supabase functions deploy create-stripe-customer**
6. **curl -L -X POST** invoke the function
7. **Create database webhook in supabase**
8. **Npm run dev** and login
9. **If user is found in user\_data with stripe customer null everything is working continue and remove all user data from auth and user\_data table**
10. **Kill server**
11. **supabase secrets list**
12. **supabase secrets set STRIPE\_KEY = sk\_13234**
13. **supabase secrets list again and check**
14. **Copy code from code in supabase/functions/index from Happys-14 repo (the first function on top of index starts with `const stripe = stripe(Deno.env.get....)`)**
15. **supabase functions deploy create-stripe-customer**
16. **Finally; npm run dev, start server, login. If no errors everything probably works, check in supabase dashboard in the user\_data table to see if the user was created with a stripe\_customer\_id, if yes, then a customer will be inserted into the stripe dashboard.**
17. **(BELOW IS MORE DETAILED)(FOLLOW THE STEPS)**

- <https://supabase.com/docs/guides/cli>
- **Move temporarily** create-stripe-customer out of the superbase file before creating edge function.
- **RUN** supabase INIT
- **RUN** supabase link --project-ref [Project-ref]
- After linking project you can excluded your project ref when deploying edge function
- **RUN** supabase functions new create-stripe-customer (Creates)
- **RUN** supabase functions deploy create-stripe-customer --project-ref [project-ref] (deploys)



- RUN `curl -L -X POST 'https:// [project-ref].functions.supabase.co/create-stripe-customer' -H 'Authorization: Bearer Anon_Key --data '{"name":"Functions"}'` (Invokes)



- Replace the create-stripe-customer folder, that was moved earlier in vs code

## CREATE A STRIPE SECRETE KEY THAT CAN BE CALLED BY FUNCTION

- 
- The screenshot shows the Stripe website's 'Get started with Stripe' section. On the left, there are three cards: 'Share a payment link', 'Send an invoice', and 'Send a payment request'. On the right, there is a sidebar with links: 'Not sure where to start?', 'Explore all products', 'For developers', and 'For mobile'. A green arrow points from the 'Secret key' link in the 'For developers' section to the 'Secret key' field in the 'Send an invoice' form.

- In vscode terminal **Run:** `supabase secrets list`  
 To set stripe key in Supabase **Run:** `supabase`  
 terminal  
**Run:** `supabase secrets list` to check if stripe

## CREATE A SUPABASE DATABASE WEBHOOK

- Create a webhook to call the Edge function that will create a stripe customer in stripe and in supabase when a new user is inserted into the database.
- Delete user\_data run npm run dev, login and if everything works you should have created a stripe customer in user\_data and stripe dash board

Add a new database webhook

Name

create\_stripe\_customer

Conditions to fire hook

Table

user\_data

This is the table the trigger will watch for changes. You can only select 1 table for a trigger.

Events

☒ Insert  
Any insert operation on the table

☐ Update  
Any update operation, of any column in the table

☐ Delete  
Any deletion of a record

Thises are the events that are watched by the function hook, only the events selected above will fire the function hook on the table you've selected.

Type of hook

HTTP Request Alpha  
Send an HTTP request to URL.

Supabase Function Coming soon  
Choose a Supabase Function to run.

Google cloud run Coming soon  
Choose a google cloud function to run

AWS Lambda Coming soon  
Choose an AWS Lambda function to run.

Type of hook

HTTP Request Alpha  
Send an HTTP request to URL.

Supabase Function Coming soon  
Choose a Supabase Function to run.

Google cloud run Coming soon  
Choose a google cloud function to run

AWS Lambda Coming soon  
Choose an AWS Lambda function to run.

Edge function endpoint

Post

Authorization

URLhttps://bnhiugttacqojdmsnfn.functions.supabase.co/create  
URL of the HTTP request. Must include HTTP/HTTPS

HTTP Headers

Content-typeapplication/json

AuthorizationBearer eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp1bmI6ImlkiCklpKp

+ Add a new header

HTTP Params

+ Add a new param

## ADD A NEW PRODUCT IN STRIPE

- **Free/standard/premium**

- **RECURRING MONTHLY**
- **\$0 / \$4.99/ \$9.99**

**To be continued.... This should work up to happy hour #15 more testing is needed, which will save you time! You will need to mess around with things to get it to work properly**

**SQL Should bring database tables to and a trigger + function you need to add database webhooks and supabase Edge functions**

**Run first**

```
create type subscription_tier_test as enum('FREE','STADARD','PREMIUM')
```

```
create table entries (
  id uuid default uuid_generate_v4() primary key,
  created_at timestamp default now() not null,
  title text,
  content text not null,
  date timestamp default now() not null,
  user_id uuid default auth.uid() references auth.users not null,
  asset_urls text [] not null
);
```

```
create table user_data
(
  id uuid references auth.users primary key,
  created_at timestamp default now() not null,
  stripe_customer_id text,
  email text not null,
  subscription_tier subscription_tier default 'FREE' not null
);
```

```
alter table entries
enable row level security;
```

```
CREATE POLICY "Authenticated users can see their own entries" ON "public"."entries"  
AS PERMISSIVE FOR SELECT  
TO authenticated  
USING (user_id = auth.uid());
```

```
CREATE POLICY "users can update their entry" ON "public"."entries"  
AS PERMISSIVE FOR UPDATE  
TO authenticated  
USING (user_id = auth.uid())  
WITH CHECK (user_id = auth.uid());
```

```
CREATE POLICY "Authenticated users can insert own data" ON "public"."entries"  
AS PERMISSIVE FOR INSERT  
TO authenticated  
WITH CHECK (user_id = auth.uid());
```

```
alter table user_data  
enable row level security;
```

```
create function public.handle_new_user()  
returns trigger as  
$$  
begin  
insert into public.user_data(id,email)  
values(new.id,new.email);  
return new;  
end;  
  
$$  
language plpgsql security definer;
```

```
create trigger on_insert_auth_user
after insert on auth.users
for each row
execute procedure public.handle_new_user();
```