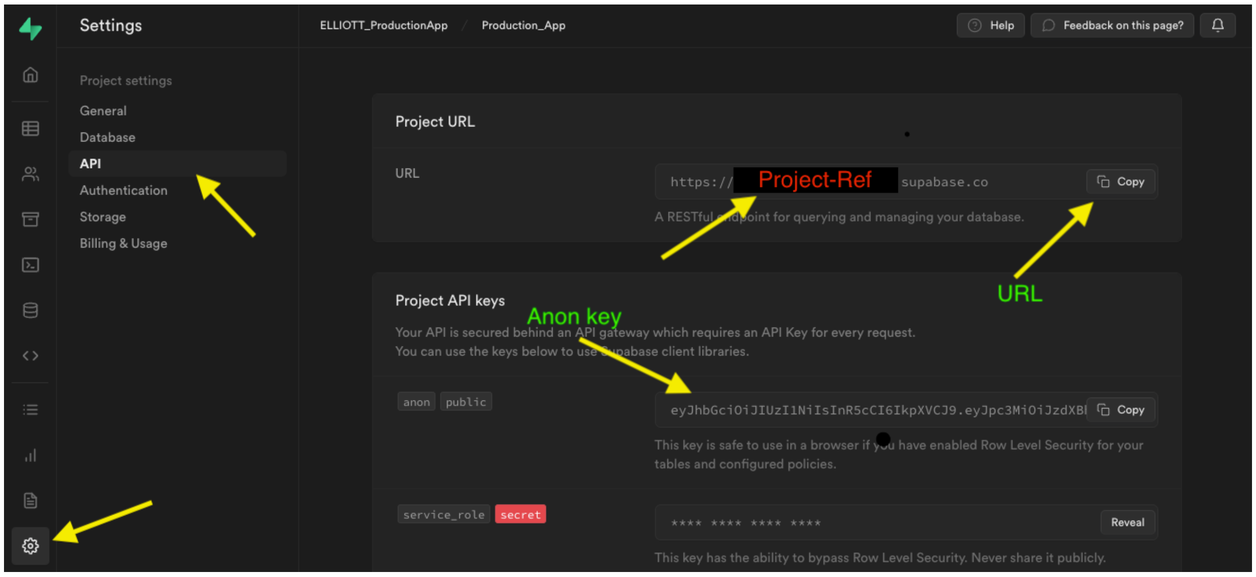
HAPPY DAYS README GUIDE

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)



A RESTful endpoint for querying and managing your database.

Your API is secured behind an API gateway which requires an API Key for every request.
You can use the keys below to use Supabase client libraries.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBl
```

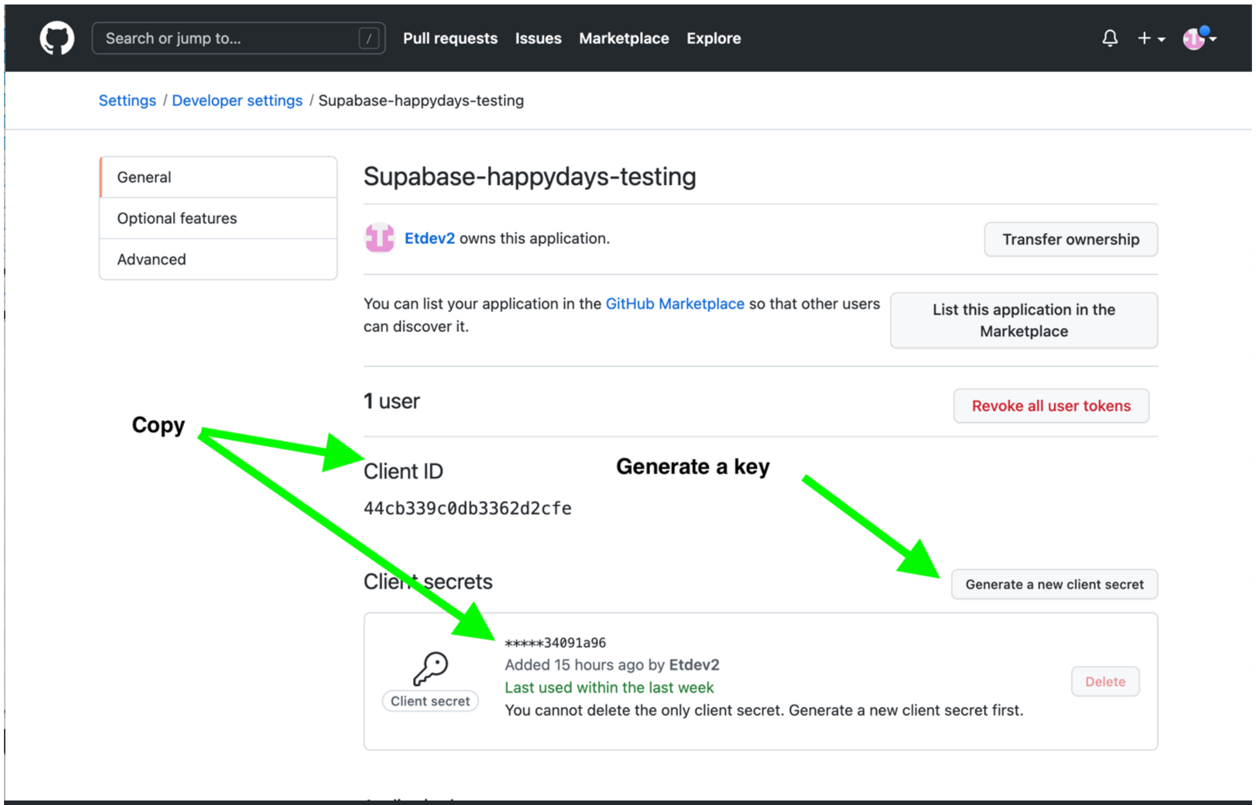
This key is safe to use in a browser if you have enabled Row Level Security for your tables and configured policies.

```
**** **** **** ****
```

This key has the ability to bypass Row Level Security. Never share it publicly.

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



Settings / Developer settings / Supabase-happydays-testing

Supabase-happydays-testing

Etdev2 owns this application.

You can list your application in the GitHub Marketplace so that other users can discover it.

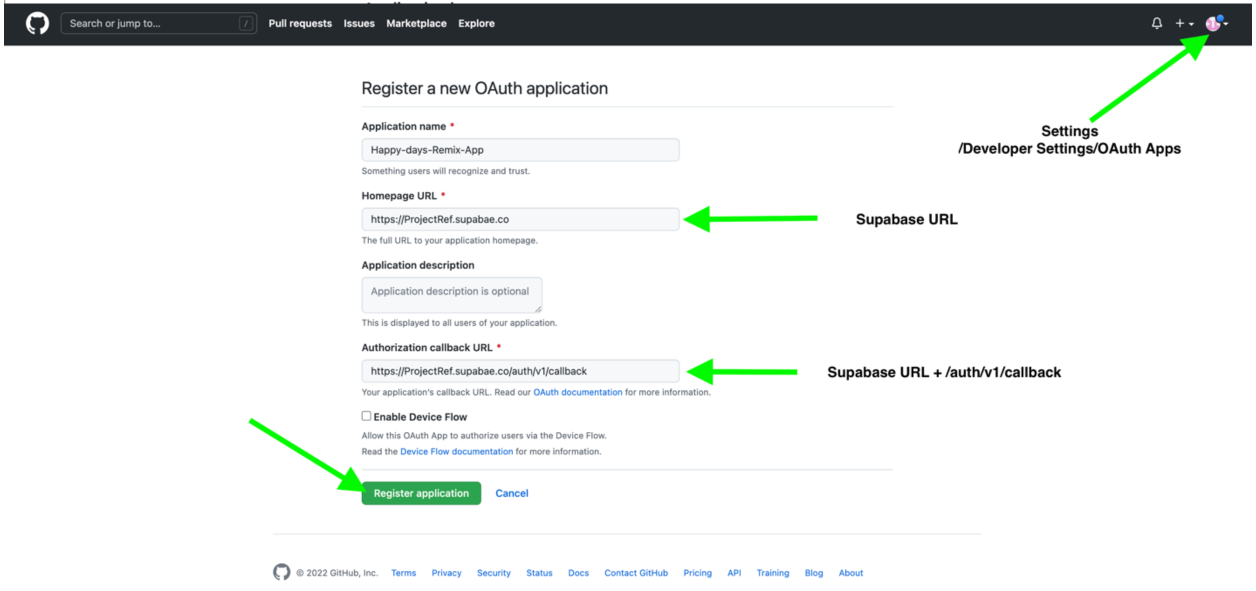
1 user

Client ID

44cb339c0db3362d2cfe

Client secrets

*****34091a96
Added 15 hours ago by Etdev2
Last used within the last week
You cannot delete the only client secret. Generate a new client secret first.



Register a new OAuth application

Application name *
Happy-days-Remix-App
Something users will recognize and trust.

Homepage URL *
https://ProjectRef.supabae.co
The full URL to your application homepage.

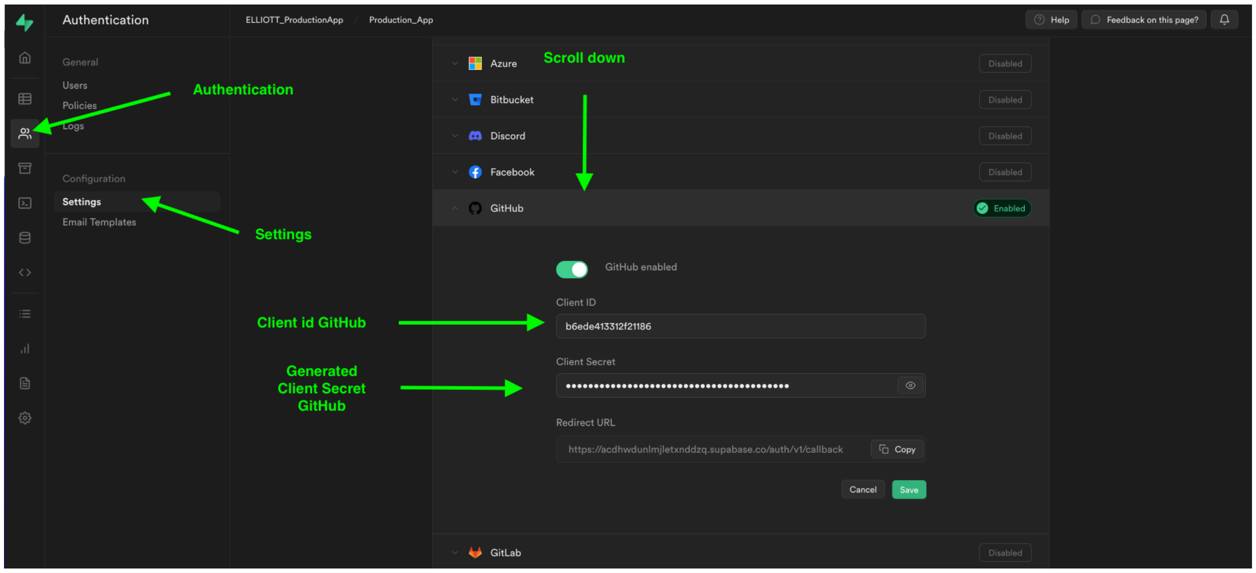
Application description
Application description is optional
This is displayed to all users of your application.

Authorization callback URL *
https://ProjectRef.supabae.co/auth/v1/callback
Your application's callback URL. Read our OAuth documentation for more information.

☐ Enable Device Flow
Allow this OAuth App to authorize users via the Device Flow.
Read the Device Flow documentation for more information.

<u>SUPABASE GITHUB AUTH PROVIDER (SUPABASE SETUP)</u>

- After setting up GitHub OAuth copy Client ID AND copy the generated client secrete
- 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



Provider	Status
Azure	Disabled
Bitbucket	Disabled
Discord	Disabled
Facebook	Disabled
GitHub	Enabled
GitLab	Disabled

GitHub enabled

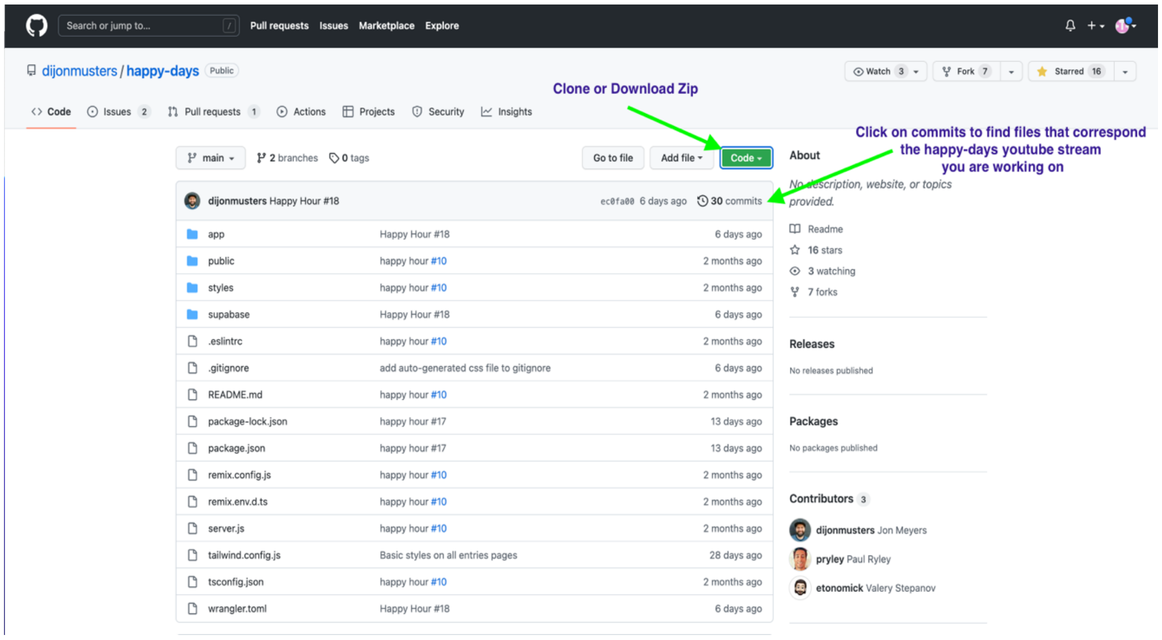
Client ID: b6ede413312f21186

Client Secret: ••

Redirect URL: https://acdhwdunlmjletxnddzq.supabase.co/auth/v1/callback

CLONING PROJECT FORM GITHUB

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

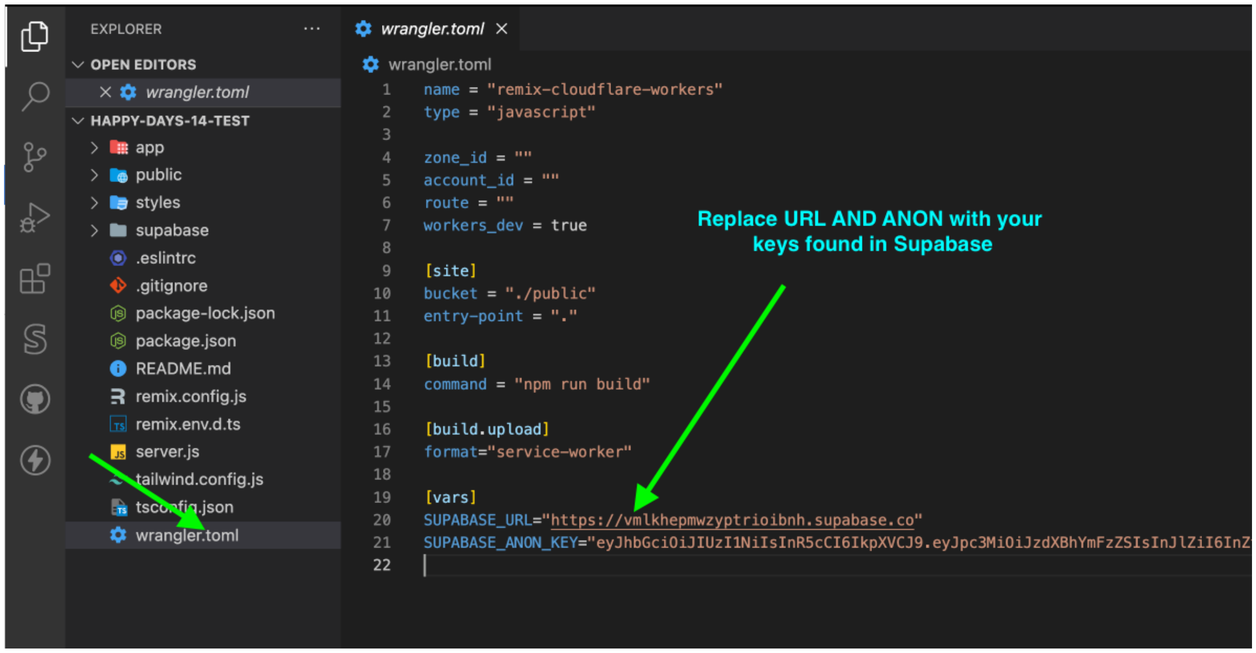


dijonmusters Happy Hour #18 — ec0fa00 6 days ago — 30 commits

app	Happy Hour #18	6 days ago
public	happy hour #10	2 months ago
styles	happy hour #10	2 months ago
supabase	Happy Hour #18	6 days ago
.eslintrc	happy hour #10	2 months ago
.gitignore	add auto-generated css file to gitignore	6 days ago
README.md	happy hour #10	2 months ago
package-lock.json	happy hour #17	13 days ago
package.json	happy hour #17	13 days ago
remix.config.js	happy hour #10	2 months ago
remix.env.d.ts	happy hour #10	2 months ago
server.js	happy hour #10	2 months ago
tailwind.config.js	Basic styles on all entries pages	28 days ago
tsconfig.json	happy hour #10	2 months ago
wrangler.toml	Happy Hour #18	6 days ago

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.



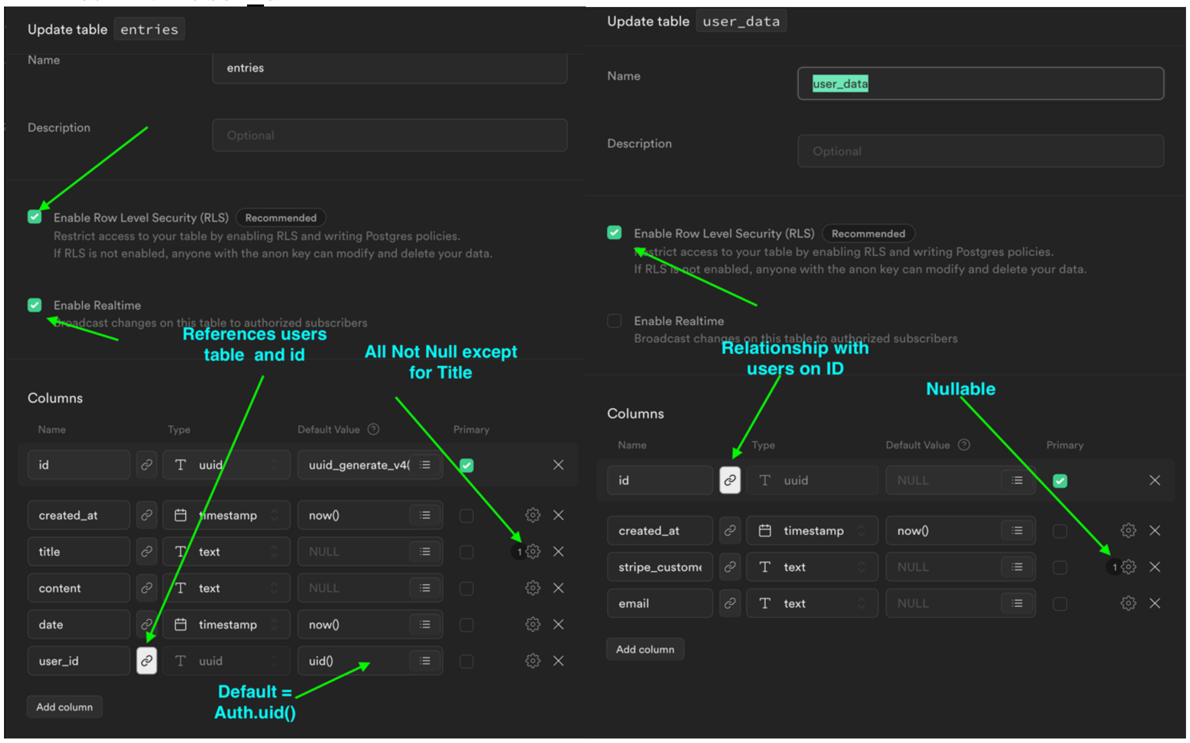
wrangler.toml

```toml
name = "remix-cloudflare-workers"
type = "javascript"

zone_id = ""
account_id = ""
route = ""
workers_dev = true

[site]
bucket = "./public"
entry-point = "."

[build]
command = "npm run build"

[build.upload]
format="service-worker"

[vars]
SUPABASE_URL="https://vmlkhepmwzyptrioibnh.supabase.co"
SUPABASE_ANON_KEY="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InZ
```

SETTING UP DATABASE

- Create an entries table and user_data table (Manually or with the SQL code bellow)

Entries AND user_data


References users table and id
All Not Null except for Title
Default = Auth.uid()
Relationship with users on ID
Nullable
Add column

Update table `entries`

Name: entries

Description: Optional

✓ Enable Row Level Security (RLS) Recommended
Restrict access to your table by enabling RLS and writing Postgres policies.
If RLS is not enabled, anyone with the anon key can modify and delete your data.

✓ Enable Realtime
Broadcast changes on this table to authorized subscribers

Columns

Name	Type	Default Value	Primary
id	uuid	uuid_generate_v4(	✓
created_at	timestamp	now()	
title	text	NULL	
content	text	NULL	
date	timestamp	now()	
user_id	uuid	uid()	

Update table `user_data`

Name: user_data

Description: Optional

✓ Enable Row Level Security (RLS) Recommended
Restrict access to your table by enabling RLS and writing Postgres policies.
If RLS is not enabled, anyone with the anon key can modify and delete your data.

Enable Realtime
Broadcast changes on this table to authorized subscribers

Columns

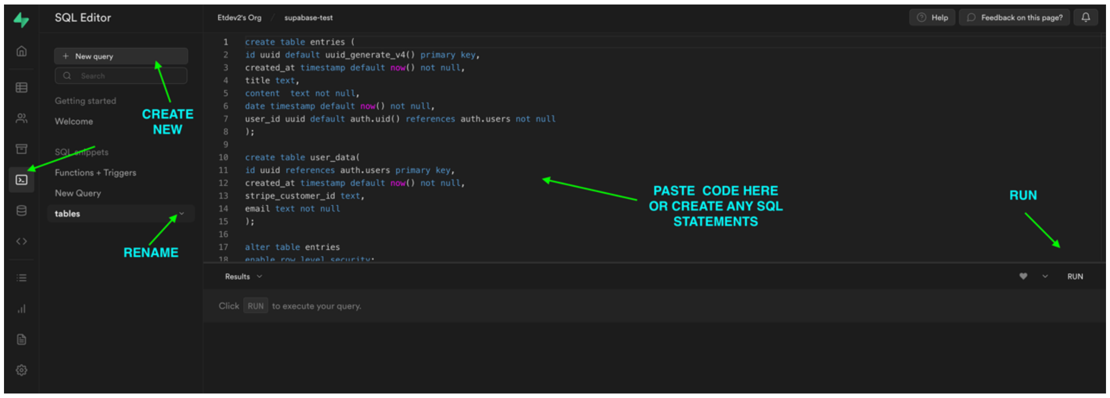
Name	Type	Default Value	Primary
id	uuid	NULL	✓
created_at	timestamp	now()	
stripe_custome	text	NULL	
email	text	NULL	

SETUP SUPABASE WITH THE SQL CODE BELLOW:(entries,user_data)

```sql
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



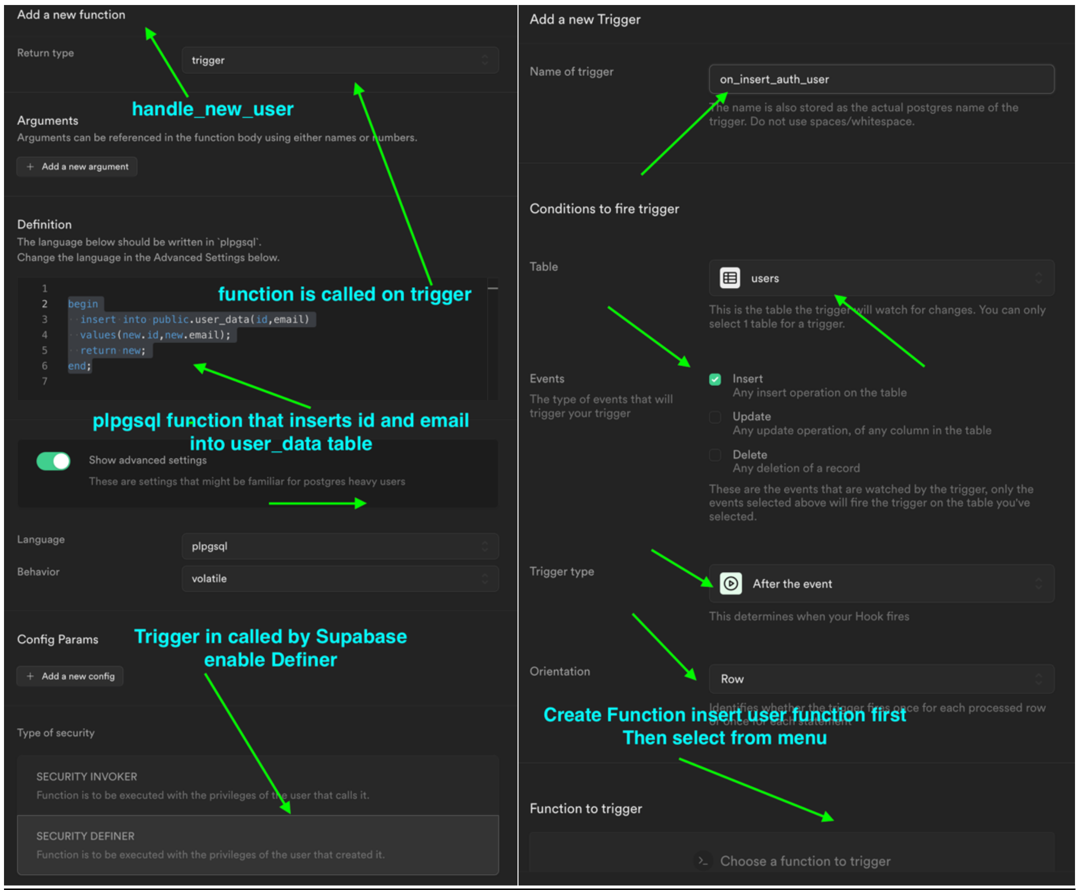
```sql
create table entries (
id uuid default uuid_generate_v4() primary key,
created_at timestamp default now() not null,
title text,
content  text not null,
date timestamp default now() not null,
user_id uuid default auth.uid() references auth.users not null
);

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;
```

<u>Finally, time to Remix!</u>

- If you have not cloned the repository fallow the directions above.
- Before starting make sure you competed (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, trigged 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.


handle_new_user
function is called on trigger
plpgsql function that inserts id and email into user_data table
Trigger in called by Supabase enable Definer
Create Function insert user function first Then select from menu

Add a new function

Return type: trigger

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

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

```
begin
  insert into public.user_data(id,email)
  values(new.id,new.email);
  return new;
end;
```

Show advanced settings
These are settings that might be familiar for postgres heavy users

Language: plpgsql
Behavior: volatile

Config Params

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 each statement

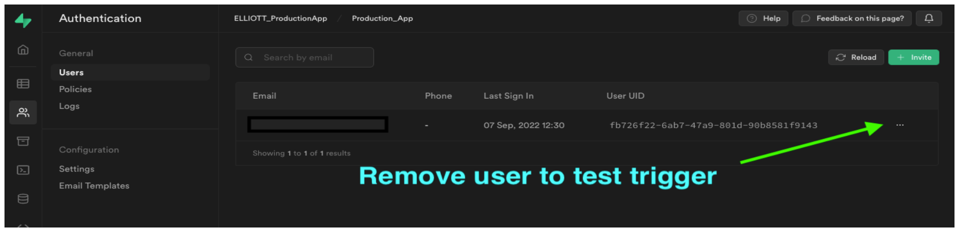
Function to trigger: Choose a function to trigger

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 VSCODE IN TERMAL

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

  Authentication | ELLIOTT_ProductionApp / Production_App

Email	Phone	Last Sign In	User UID
[redacted]	-	07 Sep, 2022 12:30	fb726f22-6ab7-47a9-801d-90b8581f9143

  Showing 1 to 1 of 1 results

  Remove user to test 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.

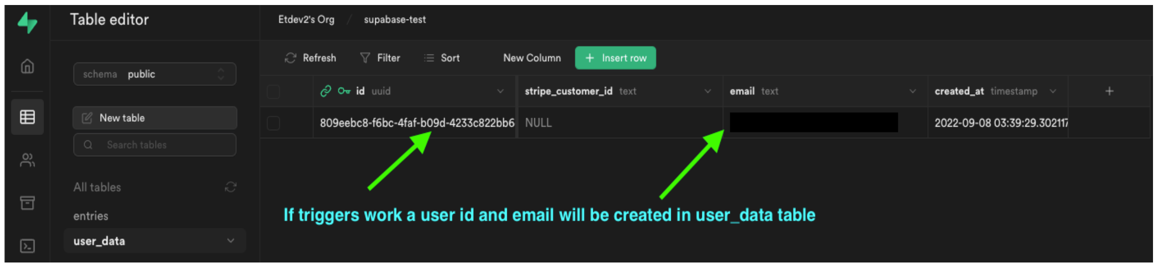
- 

  Table editor | Etdev2's Org / supabase-test

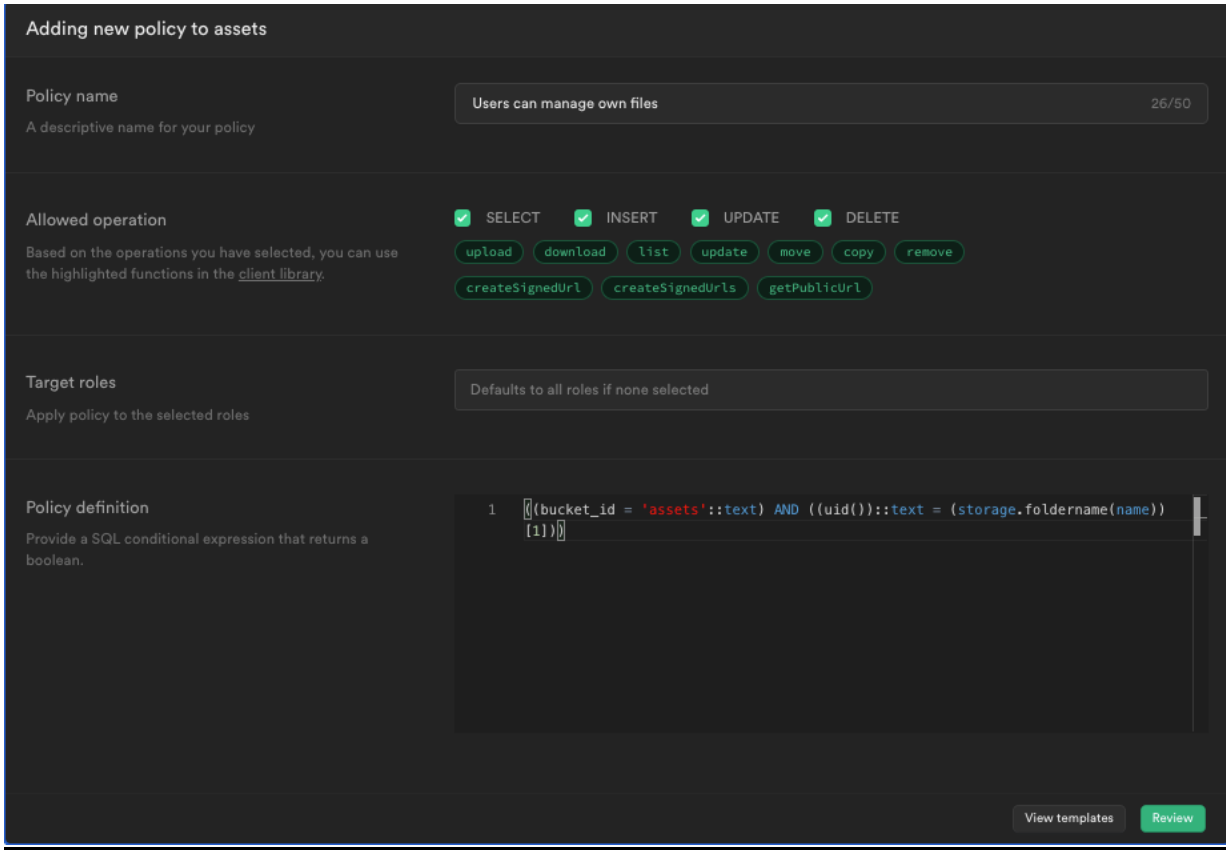
id uuid	stripe_customer_id text	email text	created_at timestamp
809eebc8-f6bc-4faf-b09d-4233c822bb6	NULL	[redacted]	2022-09-08 03:39:29.302117

  If triggers work a user id and email will be created in user_data table

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
A descriptive name for your policy
Users can manage own files 26/50

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
Apply policy to the selected roles
Defaults to all roles if none selected

Policy definition
Provide a SQL conditional expression that returns a boolean.

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

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

STEPS TO THE EDGE

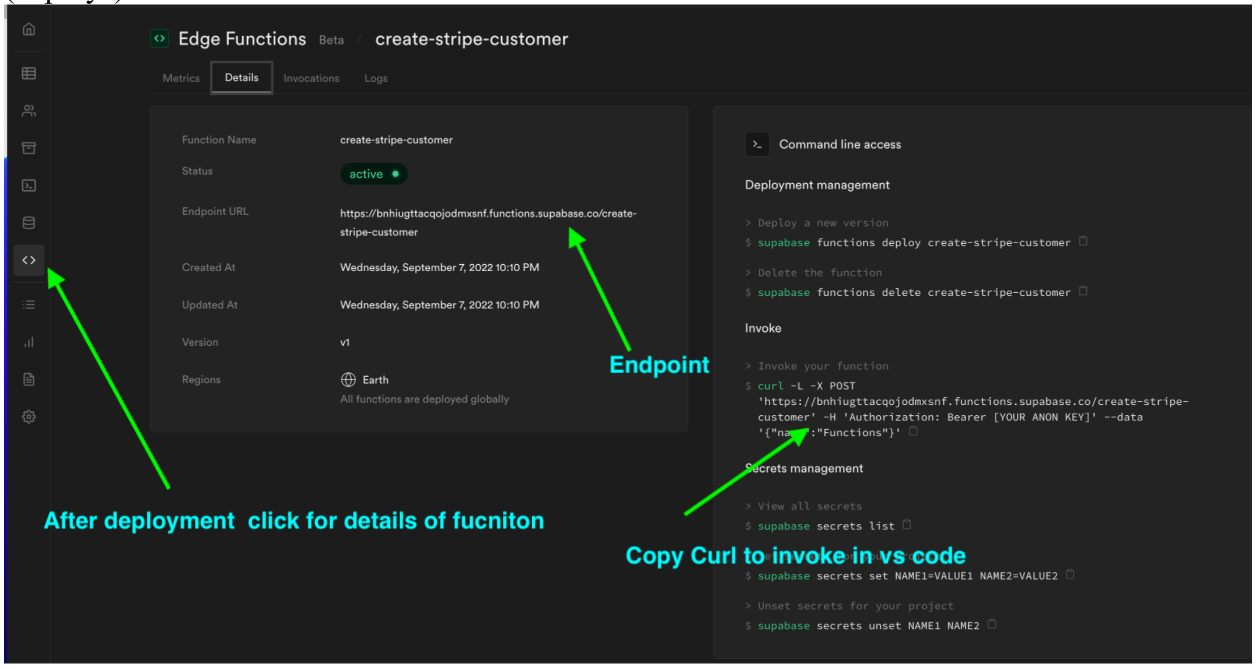
Follow steps below is more detailed may have messed around a few time to get everything working, It took me a few times to make sure everything was right and spelled correctly

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 everythning is working continue and remove all user data from auth and user_data table**
10. **Kill server**
11. **supabase secretes list**
12. **supabase secrets set STRIPE_KEY = sk_13234**
13. **supabase secretes 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 temperately 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)


Edge Functions
Beta
Metrics
Details
Invocations
Logs
Endpoint
After deployment click for details of fucniton
Copy Curl to invoke in vs code

create-stripe-customer

Function Name	create-stripe-customer
Status	active ●
Endpoint URL	https://bnhiugttacqojodmxsnf.functions.supabase.co/create-stripe-customer
Created At	Wednesday, September 7, 2022 10:10 PM
Updated At	Wednesday, September 7, 2022 10:10 PM
Version	v1
Regions	Earth All functions are deployed globally

Command line access

Deployment management

```
> Deploy a new version
$ supabase functions deploy create-stripe-customer

> Delete the function
$ supabase functions delete create-stripe-customer
```

Invoke

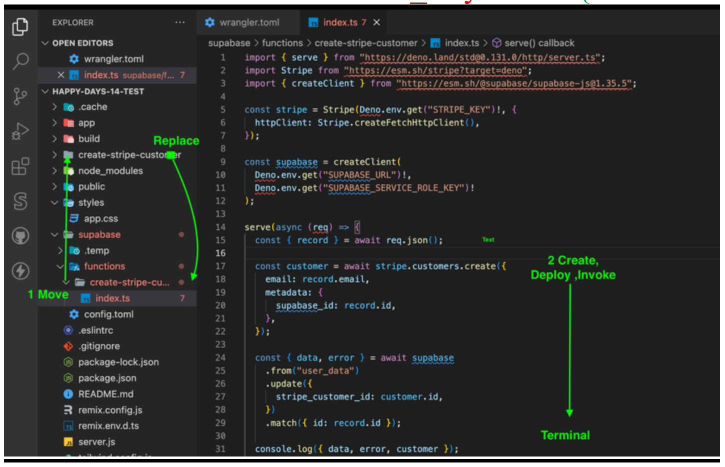
```
> Invoke your function
$ curl -L -X POST
  'https://bnhiugttacqojodmxsnf.functions.supabase.co/create-stripe-
  customer' -H 'Authorization: Bearer [YOUR ANON KEY]' --data
  '{"name":"Functions"}'
```

Secrets management

```
> View all secrets
$ supabase secrets list

> Set secrets for your project
$ supabase secrets set NAME1=VALUE1 NAME2=VALUE2

> Unset secrets for your project
$ supabase secrets unset NAME1 NAME2
```

-

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

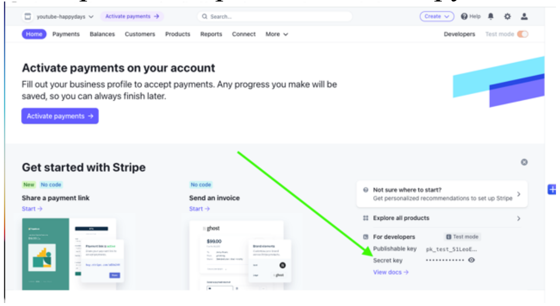


```
import { serve } from "https://deno.land/std@0.131.0/http/server.ts";
import Stripe from "https://esm.sh/stripe?target=deno";
import { createClient } from "https://esm.sh/@supabase/supabase-js@1.35.5";

const stripe = Stripe(Deno.env.get("STRIPE_KEY")!, {
  httpClient: Stripe.createFetchHttpClient(),
});

const supabase = createClient(
  Deno.env.get("SUPABASE_URL")!,
  Deno.env.get("SUPABASE_SERVICE_ROLE_KEY")!
);

serve(async (req) => {
  const { record } = await req.json();

  const customer = await stripe.customers.create({
    email: record.email,
    metadata: {
      supabase_id: record.id,
    },
  });

  const { data, error } = await supabase
    .from("user_data")
    .update({
      stripe_customer_id: customer.id,
    })
    .match({ id: record.id });

  console.log({ data, error, customer });
```

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

CREATE A STRIPE SECRETE KEY THAT CAN BE CALLED BY FUNCTION

- Create a stripe account
- In stripes developer dashboard copy secrete key



youtube-happydays · Activate payments → · Search... · Create · Help

Home · Payments · Balances · Customers · Products · Reports · Connect · More · Developers · Test mode

Activate payments on your account

Fill out your business profile to accept payments. Any progress you make will be saved, so you can always finish later.

Activate payments →

Get started with Stripe

New · No code — Share a payment link — Start →

No code — Send an invoice — Start →

Not sure where to start? Get personalized recommendations to set up Stripe

Explore all products

For developers · Test mode

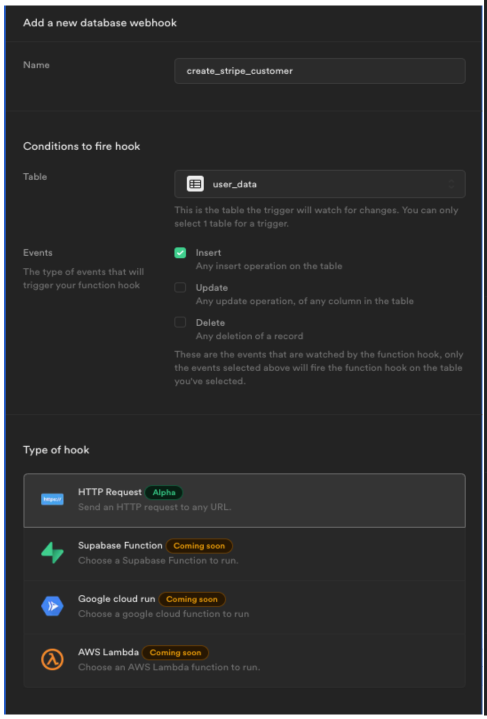
Publishable key	pk_test_51LeoE...
Secret key	••••••••••••

View docs →

-
- In vscode terminal Run: supabase secrets list (get a list of keys stored in supabase)
- To set stripe key in Supabase Run: supabase secrets set STRIPE_KEY =sk_1234 in terminal
- Run: supabase secrets list to check if stripe key was stored

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.

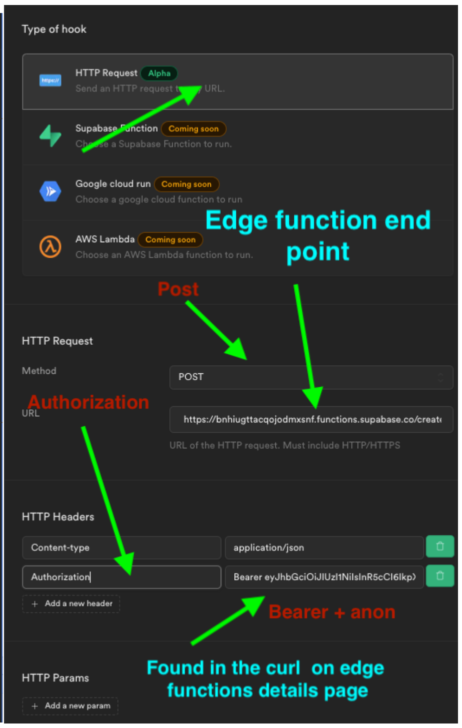
Events — The type of events that will trigger your function hook

- [x] 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 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 any 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 any 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.

HTTP Request

Method: POST

URL: https://bnhiugttacqojodmxsnf.functions.supabase.co/create

URL of the HTTP request. Must include HTTP/HTTPS

HTTP Headers

Content-type	application/json
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp

+ 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 1st

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

Run 2nd.

```

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();
```