6.1040 Last Recitation!

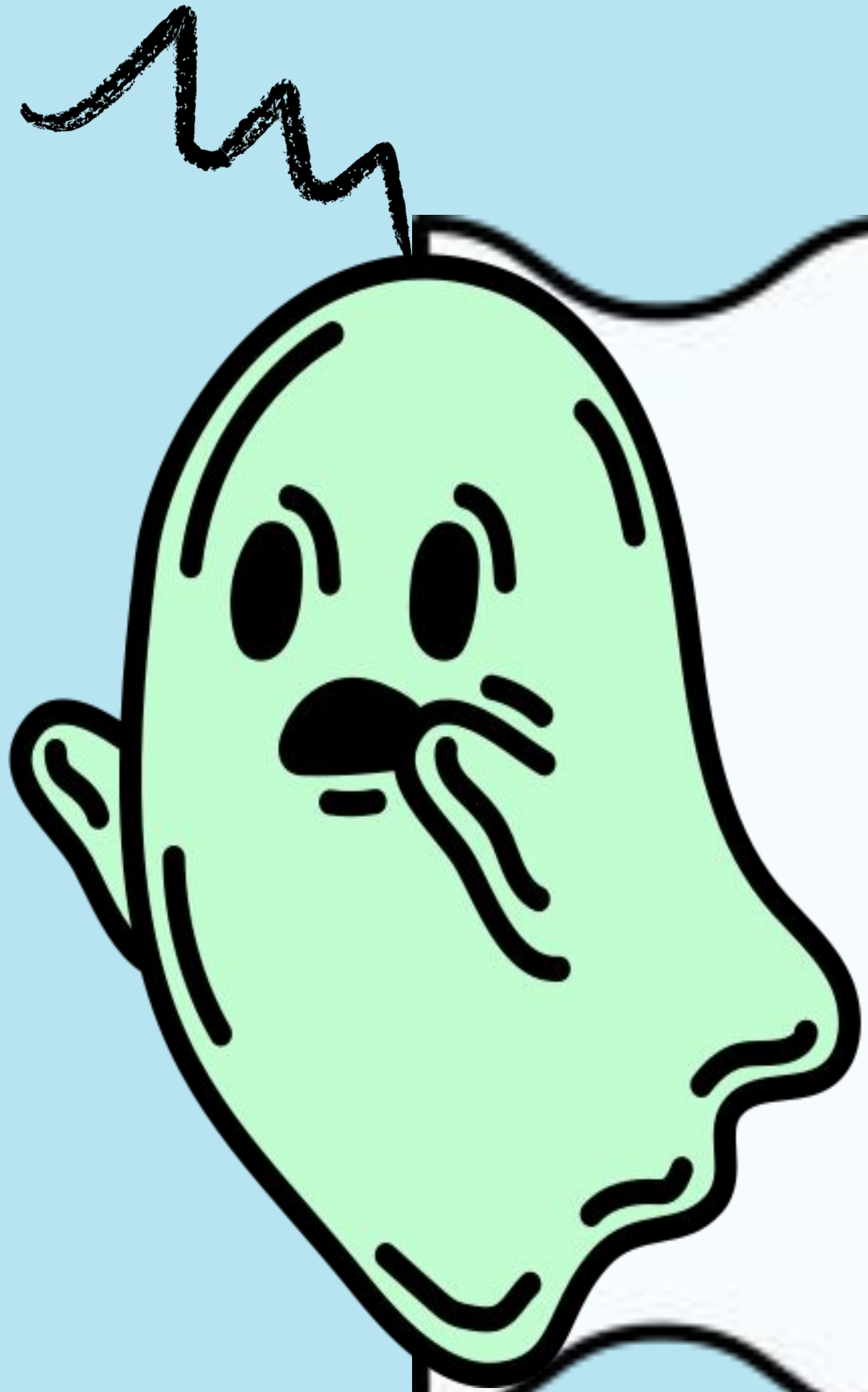
API Syncs + Deployment

Review

Your frontend and backend are two separate programs.

Your Frontend : the **Vue.js app** that runs in your browser.

Your backend : the **Deno server** that actually does the work — talks to the database, runs concept actions, etc.



Your Backend Server (From A4b)

Your Backend Server (From A4b)

Located in `your-backend/src/concept_server.ts`

Walks through your `src/concepts/` directory and looks for subdirectories like:

```
src/concepts/
├── ToDoList/
│   └── ToDoListConcept.ts
├── GiftRegistry/
│   └── GiftRegistryConcept.ts
```

Each folder should contain a `Concept.ts` file defining one concept.

It finds all its method names in each `Concept.ts` file, and automatically creates an API route for it.

API Endpoints

Every concept action is exposed as a POST endpoint.

This means that the backend makes that concept action method publicly accessible through a URL so the frontend can call it.

For example, inside your backend you might have a concept like

```
class ToDoListConcept {
  async addItem({ name }) { ... }
}
```

By exposing it as an endpoint, if someone sends a POST request to `/api/ToDoList/addItem`, the method `ToDoListConcept.addItem()` will be called.

API Endpoint Format

The format of each API endpoint created by your backend is as follows:

```
/api/<conceptName>/<actionName>
```

- <conceptName> = the name of the subfolder under src/concepts/
- <actionName> = the name of a method defined in that concept's class

HTTP Method	Endpoint Path	Description
POST	/api/ToDoList/addItem	Calls ToDoListConcept.addItem()
POST	/api/ToDoList/removeItem	Calls ToDoListConcept.removeItem()

API Endpoint Format

The format of each API endpoint created by your backend is as follows:

```
/api/<conceptName>/<actionName>
```

- <conceptName> = the name of the subfolder under src/concepts/
- <actionName> = the name of a method defined in that concept's class

HTTP Method	Endpoint Path	Description
POST	/api/ToDoList/addItem	Calls ToDoListConcept.addItem()
POST	/api/ToDoList/removeItem	Calls ToDoListConcept.removeItem()

This is the API Base - the starting piece of every API endpoint URL.

API Base

By default, your backend looks for **/api** as the API Base.

The vite.config.js file in the frontend will replace all /api calls to http://localhost:8000/api

your-backend/src/concept-server.ts

```
const flags = parseArgs(Deno.args, {
  string: ["port", "baseUrl"],
  default: {
    port: "8000",
    baseUrl: "/api",
  },
});
```

your-frontend/vite.config.js

```
export default defineConfig({
  plugins: [vue()],
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:8000',
        changeOrigin: true
      }
    }
  }
})
```

Your API endpoint becomes
http://localhost:8000/api/<concept>/<action>

API Base

By default, your backend looks for **/api** as the API Base.

The vite.config.js file in the frontend will replace all /api calls to http://localhost:8000/api

your-backend/src/concept-server.ts

```
const flags = parseArgs(Deno.args, {
  string: ["port", "baseUrl"],
  default: {
    port: "8000",
    baseUrl: "/api",
  },
});
```

your-frontend/vite.config.js

```
export default defineConfig({
  plugins: [vue()],
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:8000',
        changeOrigin: true
      }
    }
  }
})
```

Your API endpoint becomes
http://localhost:8000/api /<concept>/<action>

This is your local development API base.

API Base

With this setup, every time you want to run your app, you have to first run your deno backend so that it starts listening on port 8000, and then you can run your frontend.

When we deploy our app, we will make 2 separate deployments:

- One for your frontend, e.g. https://snoopy-frontend.onrender.com/
- One for your backend, e.g. https://snoopy-backend.onrender.com/

We will configure your frontend to direct all API requests to:

```
https://snoopy-backend.onrender.com/api/<conceptName>/<actionName>
```

API Base

With this setup, every time you want to run your app, you have to first run your deno backend so that it starts listening on port 8000, and then you can run your frontend.

When we deploy our app, we will make 2 separate deployments:

- One for your frontend, e.g. https://snoopy-frontend.onrender.com/
- One for your backend, e.g. https://snoopy-backend.onrender.com/

We will configure your frontend to direct all API requests to:

https://snoopy-backend.onrender.com/api /<conceptName>/<actionName>

This will be your production API base.

API Base

With this setup, every time you want to run your app, you have to first run your deno backend so that it starts listening on port 8000, and then you can run your frontend.

When we deploy our app, we will make 2 separate deployments:

- One for your frontend, e.g. https://snoopy-frontend.onrender.com/
- One for your backend, e.g. https://snoopy-backend.onrender.com/

We will configure your frontend to direct all API requests to:

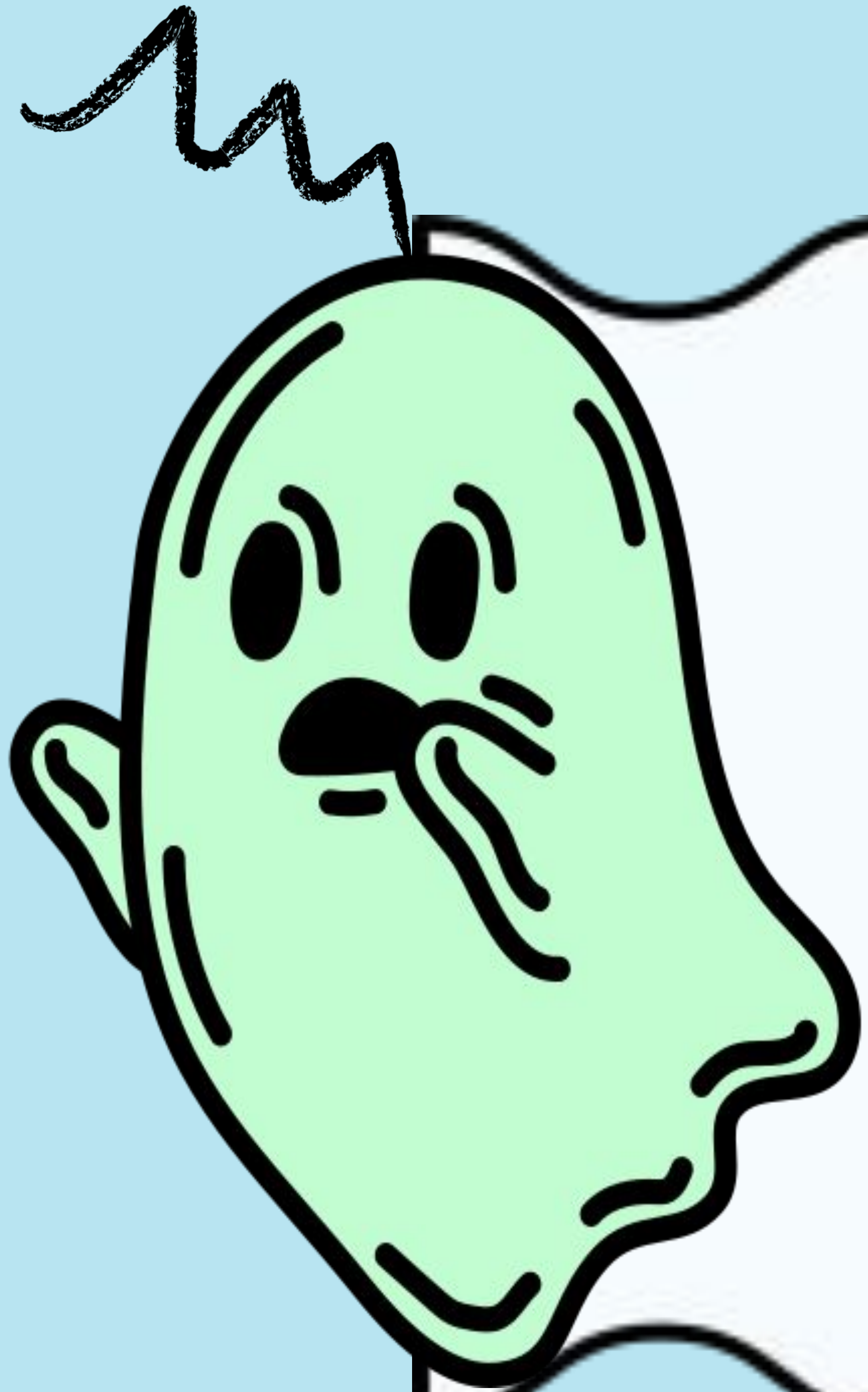
```
https://snoopy-backend.onrender.com/api    /<conceptName>/<actionName>
```

This will be your production API base.

Now, your deployed backend is always running, and you don't have to separately start the backend server anytime you want to use your app.

Your ***New*** Backend Server (From A4c)

with syncs!



Your New Backend Server

Entry point located in `your-backend/src/`**`main.ts`**

This is the new entry point for your backend server that runs your backend with

- Logging
- Requesting concept
- Syncs

Logging

your-backend/src/**main.ts**

```
/**
 * Available logging levels:
 *   Logging.OFF
 *   Logging.TRACE - display a trace of the actions.
 *   Logging.VERBOSE - display full record of synchronization.
 */
Engine.logging = Logging.TRACE;
```

By default, your backend will print out a trace of every action (so you can see what syncs are firing in the console).

We will ask you to submit the trace generated from your demo video.

Requesting Concept

The Requesting Concept encapsulates the API request server

- located in your-backend/src/concepts/Requesting/**RequestingConcept.ts**

1. Your frontend sends a POST request like /api/**concept/action** with a JSON body of **inputs**
2. The backend determine whether this route is a passthrough or excluded

*(you should define these in your-backend/src/concepts/Requesting/**passthrough.ts**)*

If the route is in **inclusions**, it's a passthrough:

- The server directly executes the concept action, e.g. Concept.action({ **inputs** })
- The return value of that action is converted to JSON and sent back to the frontend.

If the route is in **exclusions**:

- The Requesting.request({ **path**, **inputs** }) action is executed.
- The corresponding **Request Sync** is executed
- The corresponding **Respond Sync** is executed
- The response JSON is returned to frontend

Syncs

Syncs are written in your-backend/src/syncs/**syncs.ts**

For each excluded API request /api/**concept/action**, there should be 2 Syncs

The **Request Sync**

- "when" clause listens for Requesting.request({ **path**, **inputs** })
- "then" clause fires some **concept actions**

The **Response Sync**

- "when" clause listens for Requesting.request({ **path**, **inputs** }) and the **concept actions** of the Request Sync
- "then" clause fires Requesting.respond().

Run Your Backend

Run commands are defined in `your-backend/`**`deno.json`**

To start your backend with syncs and the Requesting concept:

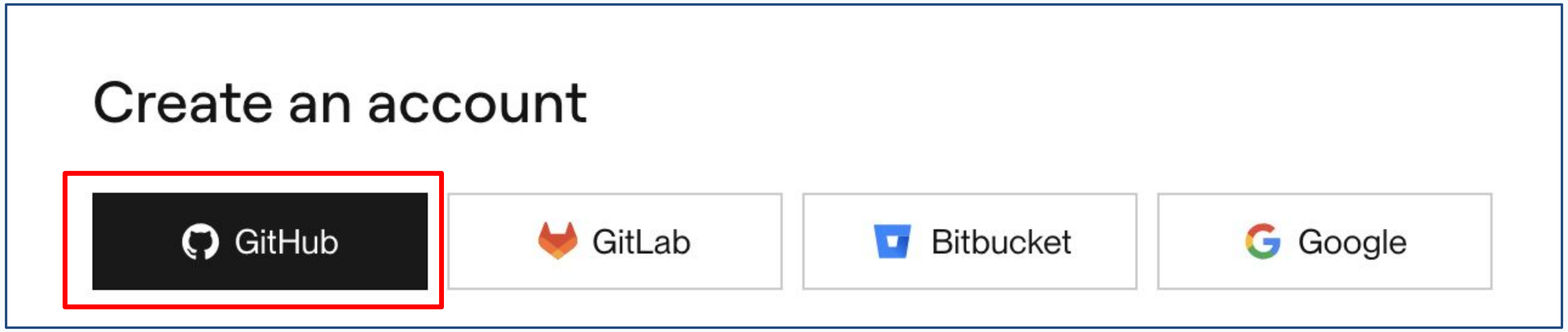
```
deno task start
```

To start your old backend server without the syncs and Requesting concept:

```
deno task concepts
```

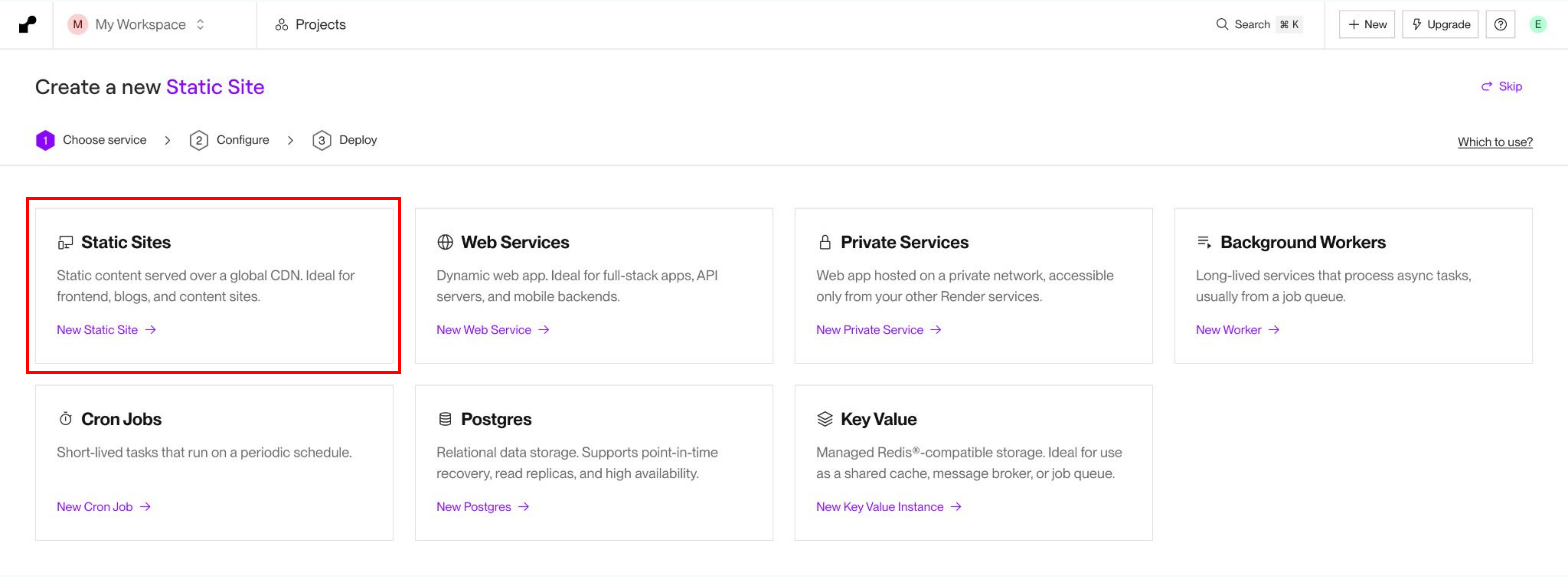
Now, we will be using Render to deploy your app!

1. Sign up for Render at https://render.com/
    - Create an account with your GitHub.



Create a new Static Site on Render for your frontend.

Create a new Static Site.



My Workspace

Projects

Search ⌘ K

New

Upgrade

Create a new Static Site

Skip

1 Choose service > 2 Configure > 3 Deploy

Which to use?

Static Sites

Static content served over a global CDN. Ideal for frontend, blogs, and content sites.

New Static Site →

Web Services

Dynamic web app. Ideal for full-stack apps, API servers, and mobile backends.

New Web Service →

Private Services

Web app hosted on a private network, accessible only from your other Render services.

New Private Service →

Background Workers

Long-lived services that process async tasks, usually from a job queue.

New Worker →

Cron Jobs

Short-lived tasks that run on a periodic schedule.

New Cron Job →

Postgres

Relational data storage. Supports point-in-time recovery, read replicas, and high availability.

New Postgres →

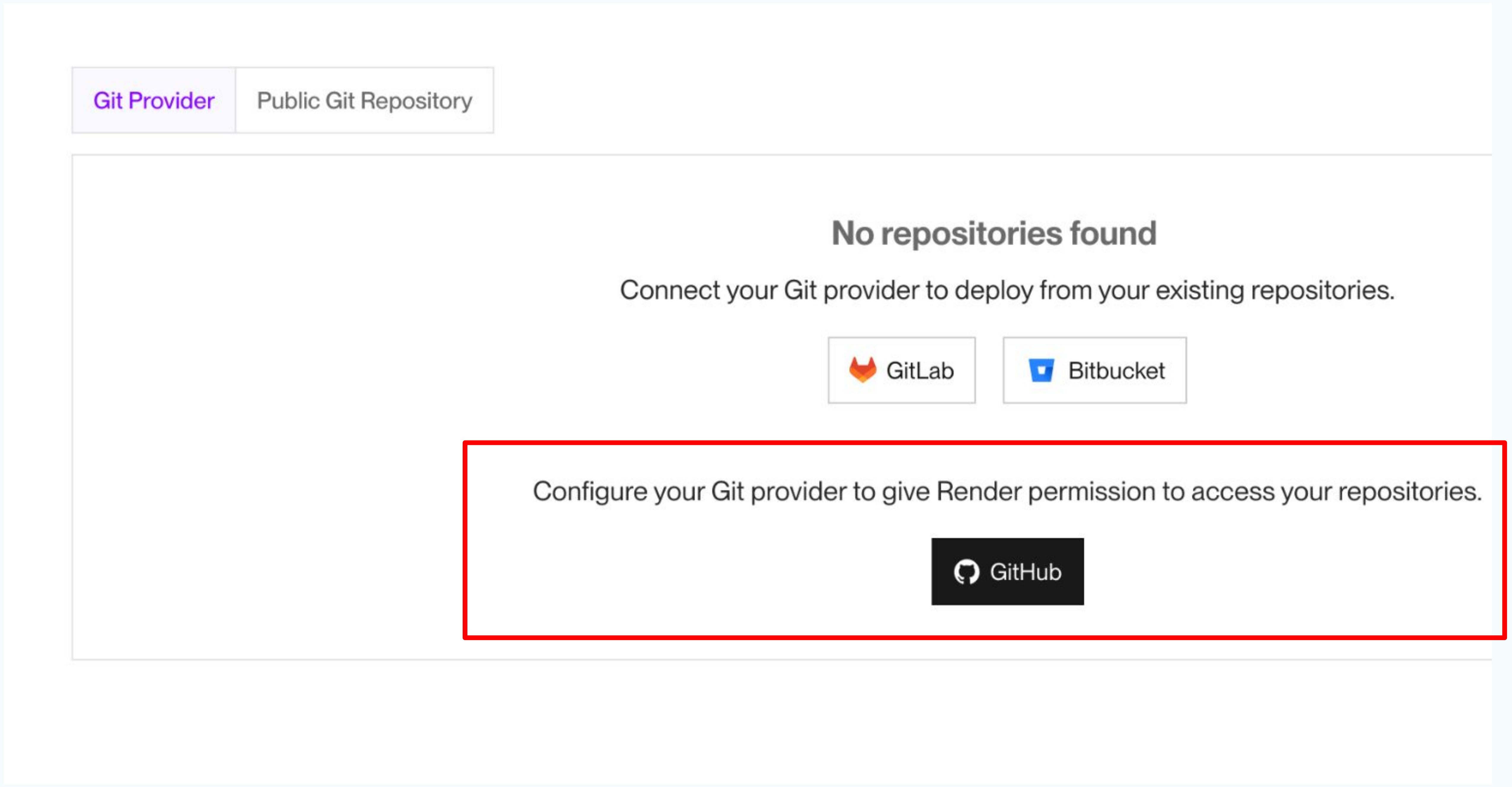
Key Value

Managed Redis®-compatible storage. Ideal for use as a shared cache, message broker, or job queue.

New Key Value Instance →

Configure your Git provider to give Render permission to access your repositories.

- This will redirect you to GitHub
- Select the account under which you host your repos for this class.
- Authorize Render to all repositories. This will allow Render to automatically redeploy your site every time you commit and push to the selected repo.



Git Provider | Public Git Repository

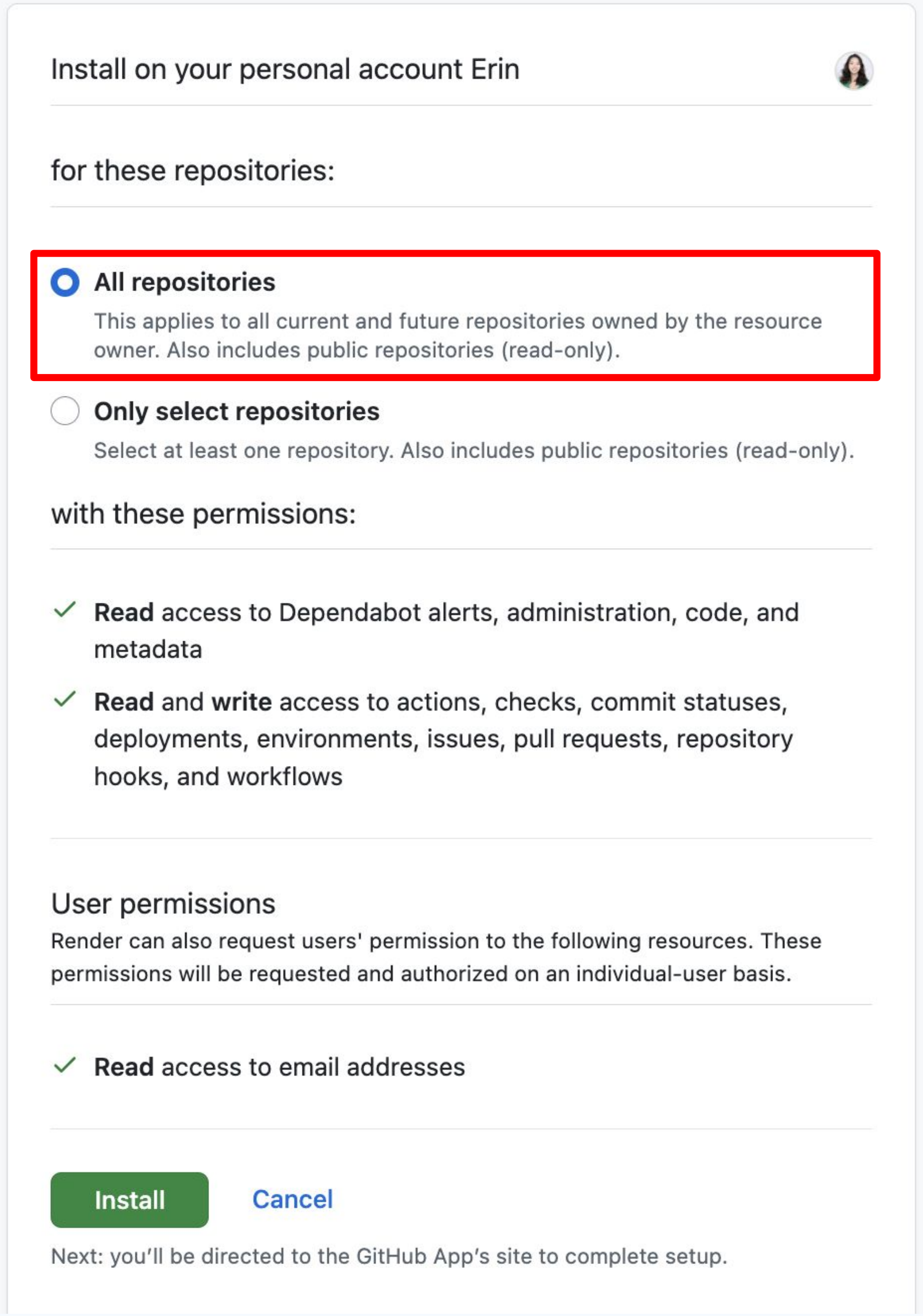
No repositories found

Connect your Git provider to deploy from your existing repositories.

GitLab | Bitbucket

Configure your Git provider to give Render permission to access your repositories.

GitHub



Install on your personal account Erin

for these repositories:

(●) **All repositories**
This applies to all current and future repositories owned by the resource owner. Also includes public repositories (read-only).

( ) **Only select repositories**
Select at least one repository. Also includes public repositories (read-only).

with these permissions:

✓ **Read** access to Dependabot alerts, administration, code, and metadata

✓ **Read** and **write** access to actions, checks, commit statuses, deployments, environments, issues, pull requests, repository hooks, and workflows

User permissions

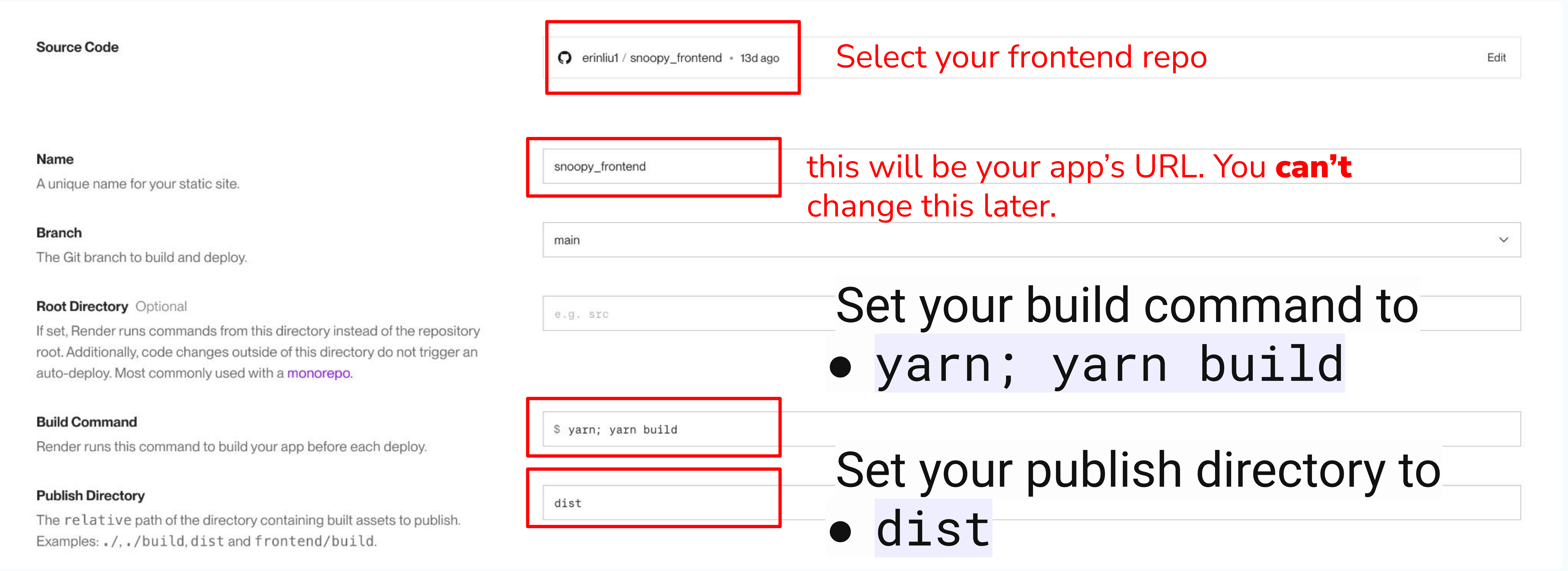
Render can also request users' permission to the following resources. These permissions will be requested and authorized on an individual-user basis.

✓ **Read** access to email addresses

Install | Cancel

Next: you'll be directed to the GitHub App's site to complete setup.

Deploy your Static Site



Source Code

erinliu1 / snoopy_frontend · 13d ago

Select your frontend repo

Edit

Name
A unique name for your static site.

snoopy_frontend

this will be your app's URL. You **can't** change this later.

Branch
The Git branch to build and deploy.

main

Root Directory Optional
If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a monorepo.

e.g. src

Set your build command to

- `yarn; yarn build`

Build Command
Render runs this command to build your app before each deploy.

```
$ yarn; yarn build
```

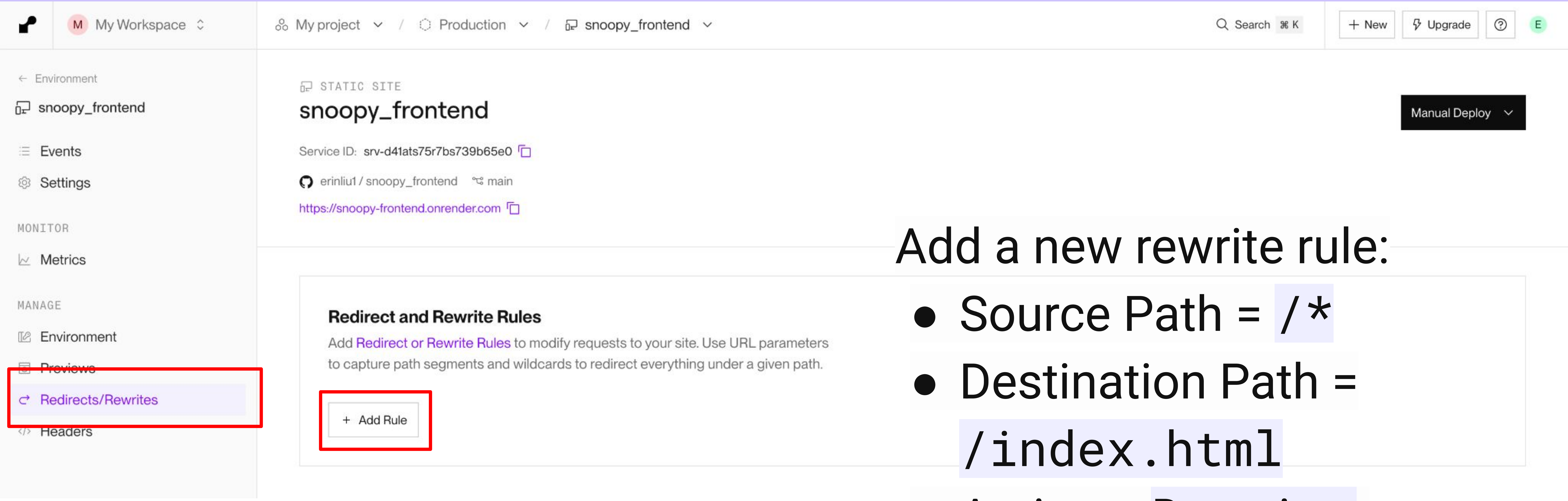
Publish Directory
The `relative` path of the directory containing built assets to publish. Examples: `./`, `./build`, `dist` and `frontend/build`.

```
dist
```

Set your publish directory to

- `dist`

This is required for any Vue app



snoopy_frontend

Service ID: srv-d41ats75r7bs739b65e0

erinliu1 / snoopy_frontend main

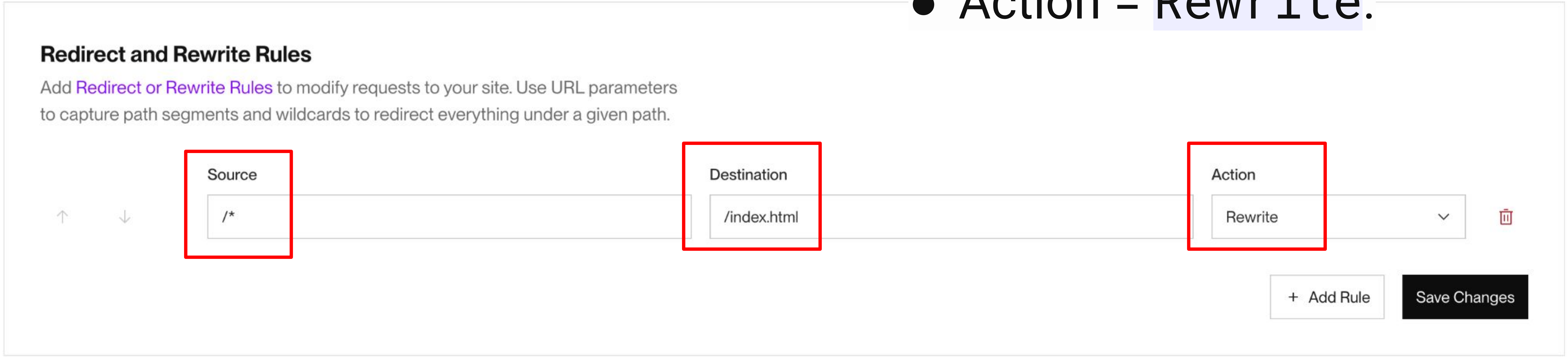
https://snoopy-frontend.onrender.com

Redirect and Rewrite Rules

Add Redirect or Rewrite Rules to modify requests to your site. Use URL parameters to capture path segments and wildcards to redirect everything under a given path.

Add a new rewrite rule:

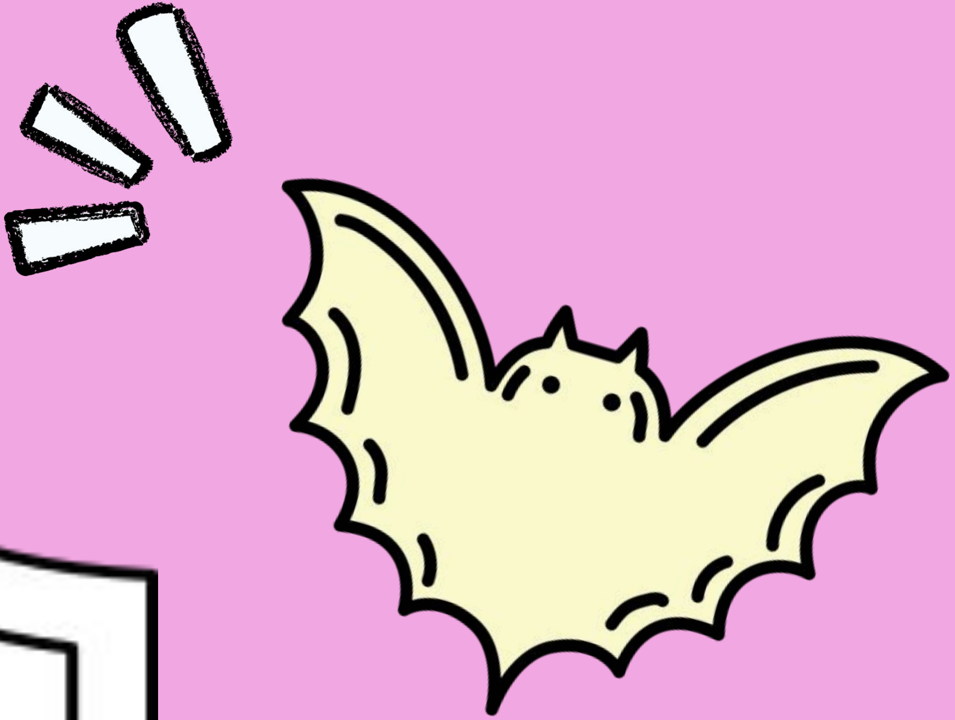
- Source Path = `/*`
- Destination Path = `/index.html`
- Action = `Rewrite`.



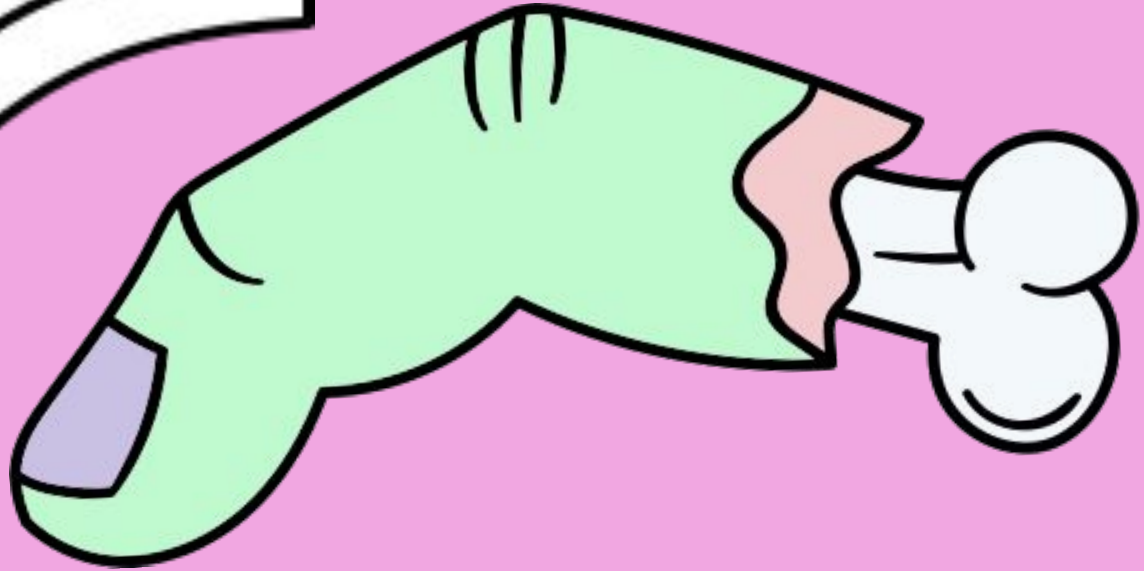
Redirect and Rewrite Rules

Add Redirect or Rewrite Rules to modify requests to your site. Use URL parameters to capture path segments and wildcards to redirect everything under a given path.

Source	Destination	Action
/*	/index.html	Rewrite



Sync the changes from the concept-backend GitHub repo



credits to Xuan for figuring this out :)

In your backend repo:

1. Check your current remotes

```
git remote -v
```

You should see

```
origin  https://github.com/<your-username>/<your_backend>.git (fetch)
origin  https://github.com/<your-username>/<your_backend>.git (push)
```

If you see any existing upstream repos, do `git remote remove upstream`. You should see only your origin now.

2. Add the original repo as upstream

```
git remote add upstream https://github.com/61040-fa25/concept_backend.git
```

Then confirm with git remote -v

```
origin  https://github.com/<your-username>/<your_backend>.git (fetch)
origin  https://github.com/<your-username>/<your_backend>.git (push)
upstream  https://github.com/61040-fa25/concept_backend.git (fetch)
upstream  https://github.com/61040-fa25/concept_backend.git (push)
```

credits to Xuan for figuring this out :)

Now, make sure you're in the main branch of your backend repo:

```
git checkout main
```

Fetch the latest changes from the original repo

```
git fetch upstream
```

Rebase your local main branch on top of the upstream main

```
git rebase upstream/main
```

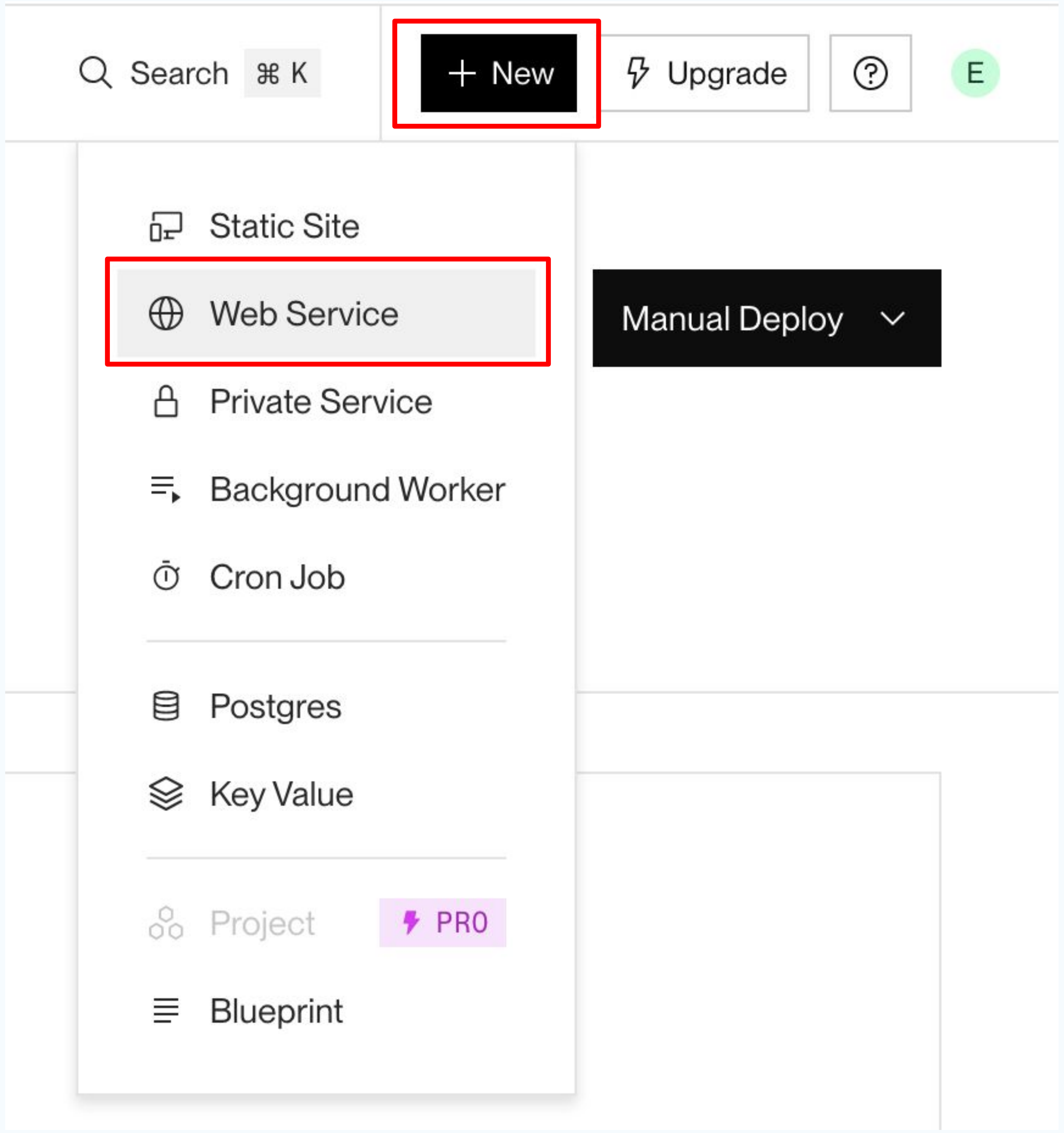
You must manually fix any merge conflicts that arise.

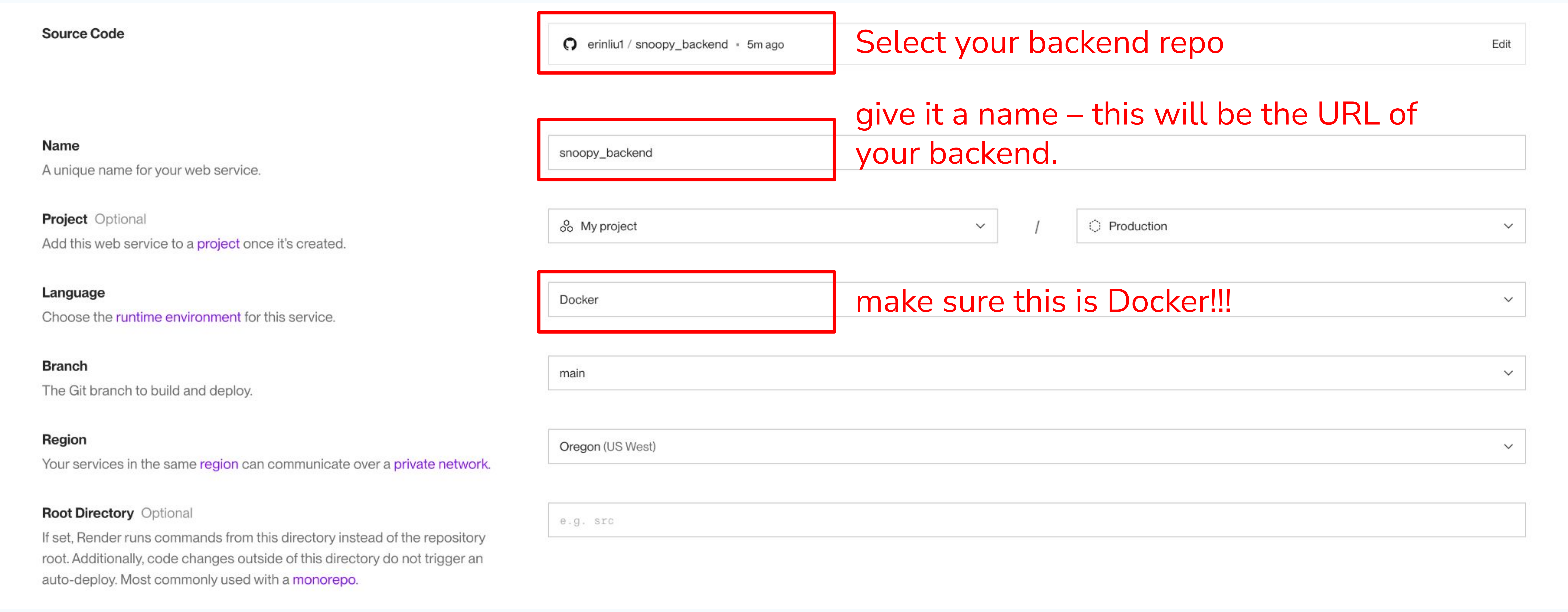
After rebasing, git status will show that branches have diverged. Overwrites your repo's history with the new rebased version

```
git push origin main --force
```

Create a new Web Service on Render for your backend.

Click the +New button on the top right corner, and click **Web Service** .




Source Code

erinliu1 / snoopy_backend • 5m ago

Edit

Select your backend repo

Name

A unique name for your web service.

snoopy_backend

give it a name – this will be the URL of your backend.

Project Optional

Add this web service to a project once it's created.

My project / Production

Language

Choose the runtime environment for this service.

Docker

make sure this is Docker!!!

Branch

The Git branch to build and deploy.

main

Region

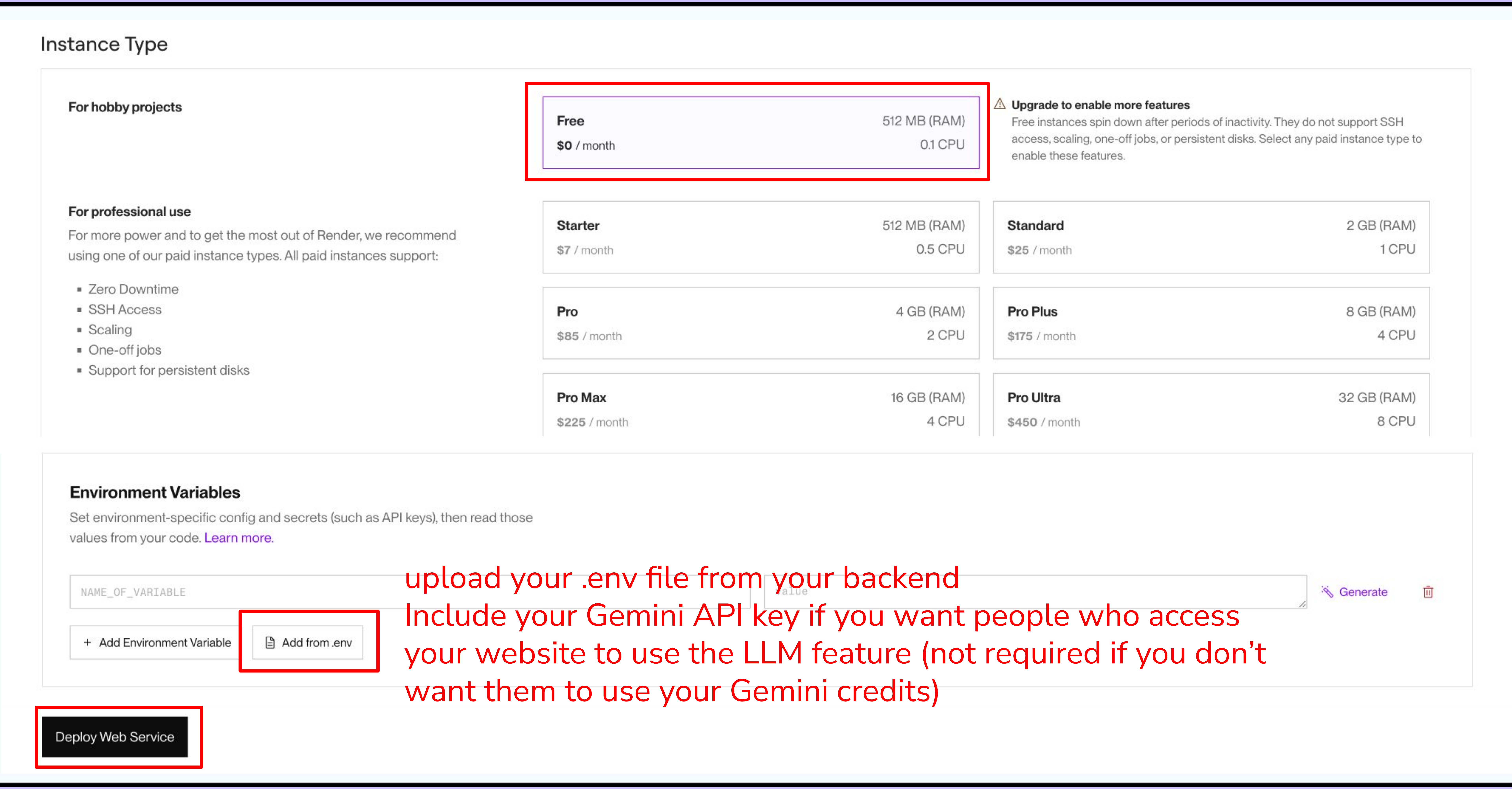
Your services in the same region can communicate over a private network.

Oregon (US West)

Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a monorepo.

e.g. src



Instance Type

For hobby projects

Free	**$0** / month	512 MB (RAM) 0.1 CPU

⚠ **Upgrade to enable more features**
Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.

For professional use
For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:

- Zero Downtime
- SSH Access
- Scaling
- One-off jobs
- Support for persistent disks

Starter	$7 / month	512 MB (RAM)	0.5 CPU
Standard	$25 / month	2 GB (RAM)	1 CPU
Pro	$85 / month	4 GB (RAM)	2 CPU
Pro Plus	$175 / month	8 GB (RAM)	4 CPU
Pro Max	$225 / month	16 GB (RAM)	4 CPU
Pro Ultra	$450 / month	32 GB (RAM)	8 CPU

Environment Variables
Set environment-specific config and secrets (such as API keys), then read those values from your code. Learn more.

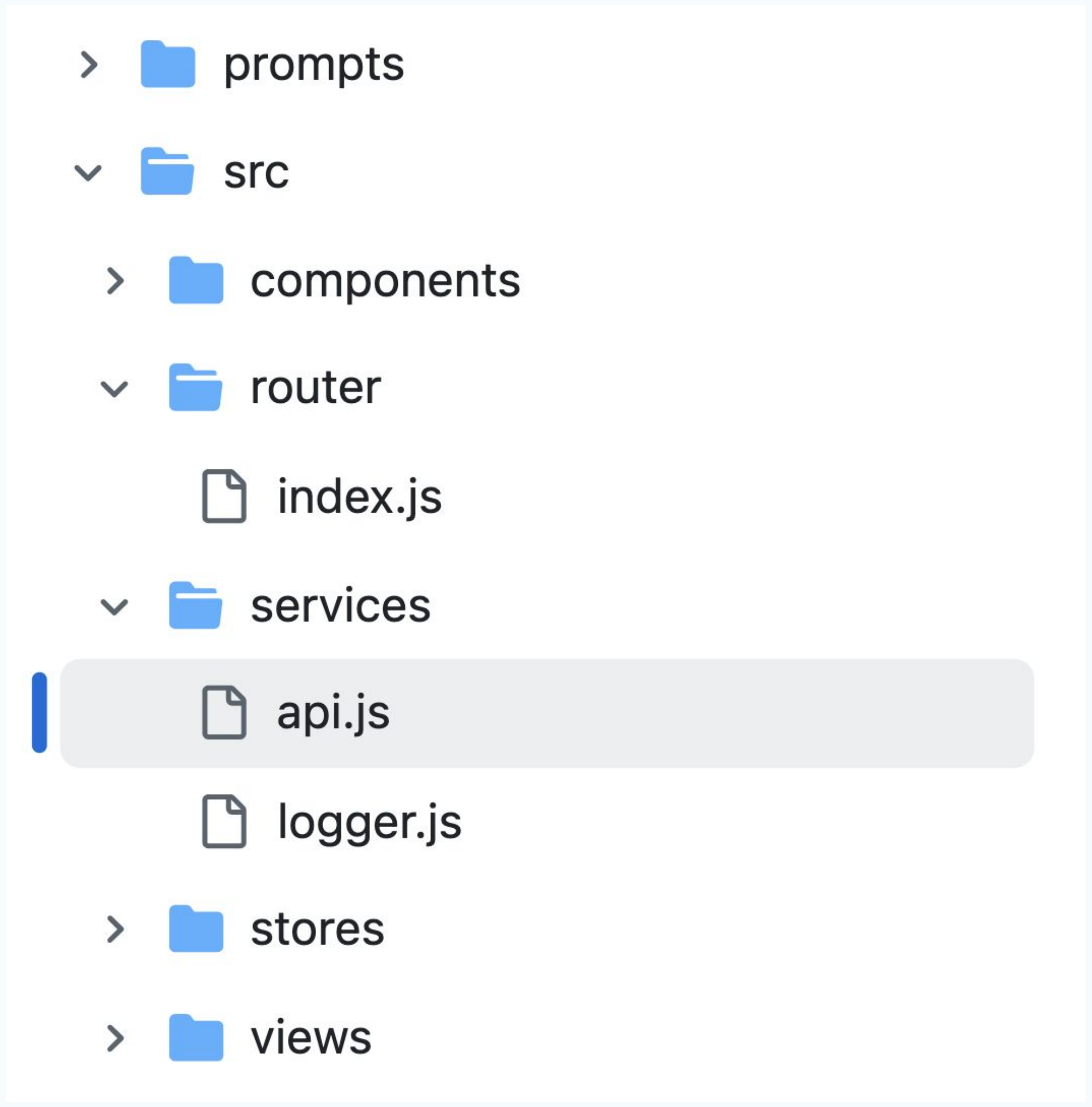
upload your .env file from your backend
Include your Gemini API key if you want people who access your website to use the LLM feature (not required if you don't want them to use your Gemini credits)

Hook up your frontend to your backend.

Go to your frontend codebase.

Open your API file (e.g. api.js)

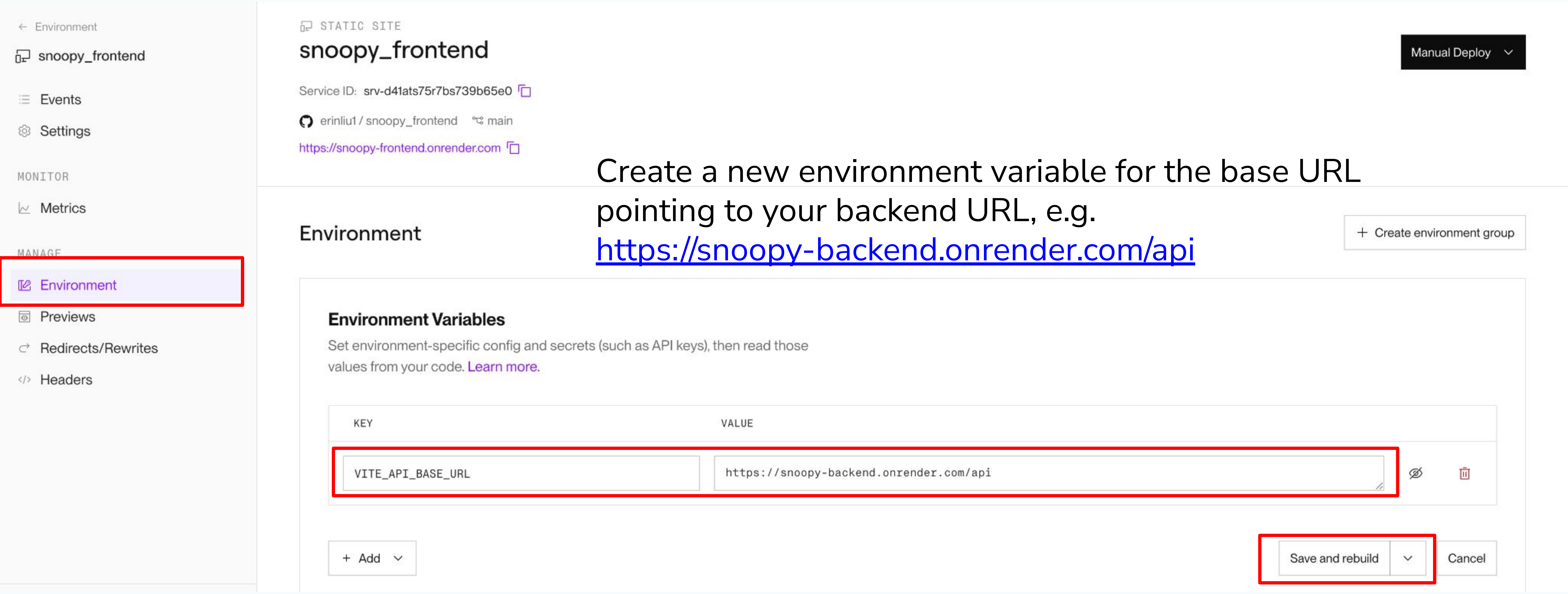
Add this line:



```
const API_BASE = import.meta.env.VITE_API_BASE_URL || '/api'
```

Commit and push your changes.

Go back to your frontend on Render



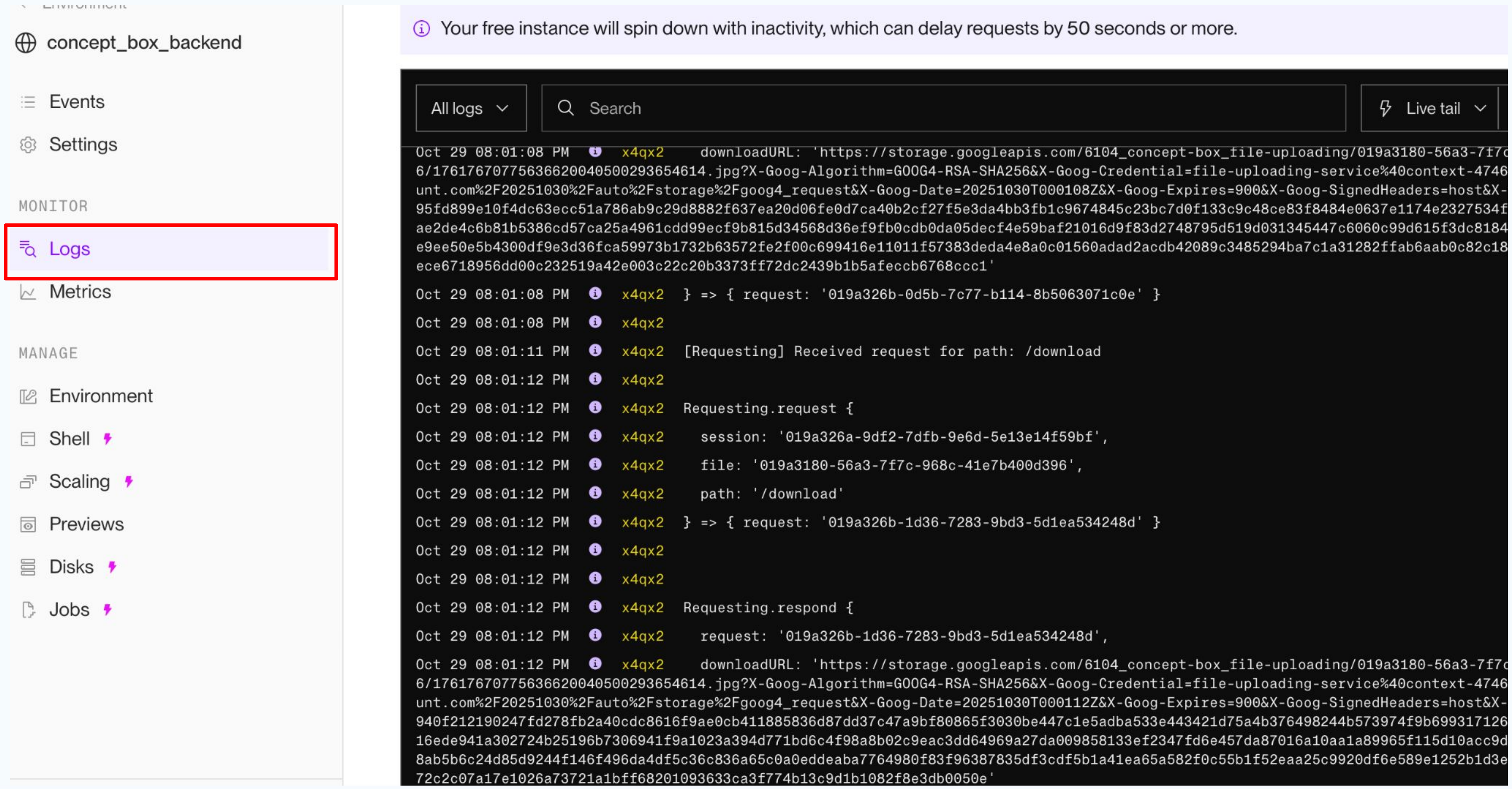
Create a new environment variable for the base URL pointing to your backend URL, e.g. https://snoopy-backend.onrender.com/api

Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. Learn more.

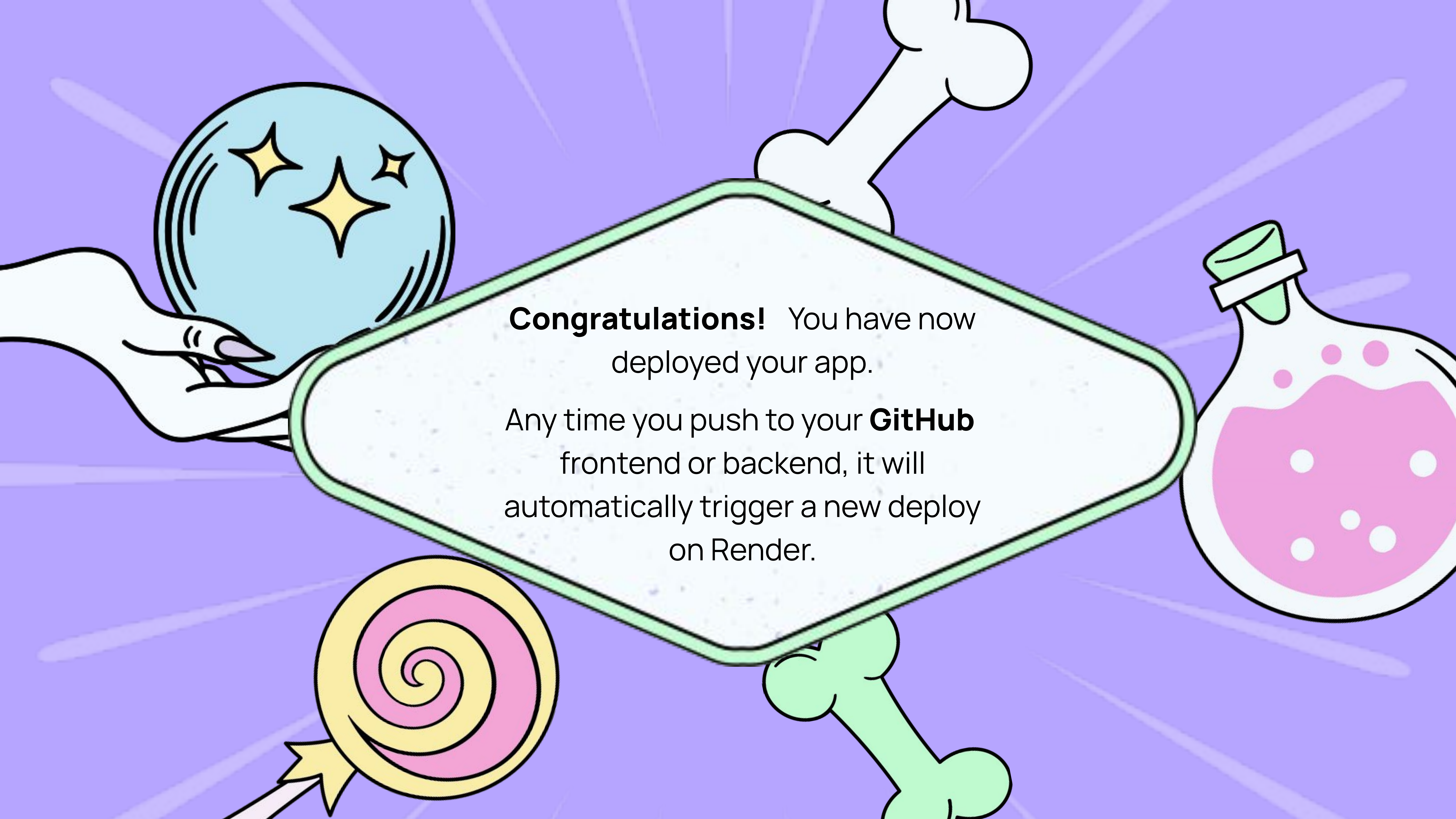
KEY	VALUE
VITE_API_BASE_URL	https://snoopy-backend.onrender.com/api

Render Dashboard: Finding Logs



Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.

```
Oct 29 08:01:08 PM  x4qx2    downloadURL: 'https://storage.googleapis.com/6104_concept-box_file-uploading/019a3180-56a3-7f7c
6/17617670775636620040500293654614.jpg?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=file-uploading-service%40context-4746
unt.com%2F20251030%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20251030T000108Z&X-Goog-Expires=900&X-Goog-SignedHeaders=host&X-
95fd899e10f4dc63ecc51a786ab9c29d8882f637ea20d06fe0d7ca40b2cf27f5e3da4bb3fb1c9674845c23bc7d0f133c9c48ce83f8484e0637e1174e2327534f
ae2de4c6b81b5386cd57ca25a4961cdd99ecf9b815d34568d36ef9fb0cdb0da05decf4e59baf21016d9f83d2748795d519d031345447c6060c99d615f3dc8184
e9ee50e5b4300df9e3d36fca59973b1732b63572fe2f00c699416e11011f57383deda4e8a0c01560adad2acdb42089c3485294ba7c1a31282ffab6aab0c82c18
ece6718956dd00c232519a42e003c22c20b3373ff72dc2439b1b5afeccb6768ccc1'
Oct 29 08:01:08 PM  x4qx2  } => { request: '019a326b-0d5b-7c77-b114-8b5063071c0e' }
Oct 29 08:01:08 PM  x4qx2
Oct 29 08:01:11 PM  x4qx2  [Requesting] Received request for path: /download
Oct 29 08:01:12 PM  x4qx2
Oct 29 08:01:12 PM  x4qx2  Requesting.request {
Oct 29 08:01:12 PM  x4qx2    session: '019a326a-9df2-7dfb-9e6d-5e13e14f59bf',
Oct 29 08:01:12 PM  x4qx2    file: '019a3180-56a3-7f7c-968c-41e7b400d396',
Oct 29 08:01:12 PM  x4qx2    path: '/download'
Oct 29 08:01:12 PM  x4qx2  } => { request: '019a326b-1d36-7283-9bd3-5d1ea534248d' }
Oct 29 08:01:12 PM  x4qx2
Oct 29 08:01:12 PM  x4qx2
Oct 29 08:01:12 PM  x4qx2  Requesting.respond {
Oct 29 08:01:12 PM  x4qx2    request: '019a326b-1d36-7283-9bd3-5d1ea534248d',
Oct 29 08:01:12 PM  x4qx2    downloadURL: 'https://storage.googleapis.com/6104_concept-box_file-uploading/019a3180-56a3-7f7c
6/17617670775636620040500293654614.jpg?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=file-uploading-service%40context-4746
unt.com%2F20251030%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20251030T000112Z&X-Goog-Expires=900&X-Goog-SignedHeaders=host&X-
940f212190247fd278fb2a40cdc8616f9ae0cb411885836d87dd37c47a9bf80865f3030be447c1e5adba533e443421d75a4b376498244b573974f9b699317126
16ede941a302724b25196b7306941f9a1023a394d771bd6c4f98a8b02c9eac3dd64969a27da009858133ef2347fd6e457da87016a10aa1a89965f115d10acc9d
8ab5b6c24d85d9244f146f496da4df5c36c836a65c0a0eddeaba7764980f83f96387835df3cdf5b1a41ea65a582f0c55b1f52eaa25c9920df6e589e1252b1d3e
72c2c07a17e1026a73721a1bff68201093633ca3f774b13c9d1b1082f8e3db0050e'
```



Congratulations! You have now deployed your app.

Any time you push to your **GitHub** frontend or backend, it will automatically trigger a new deploy on Render.