Deployment

This guide explains everything required to run Gatos in production.

Prerequisites

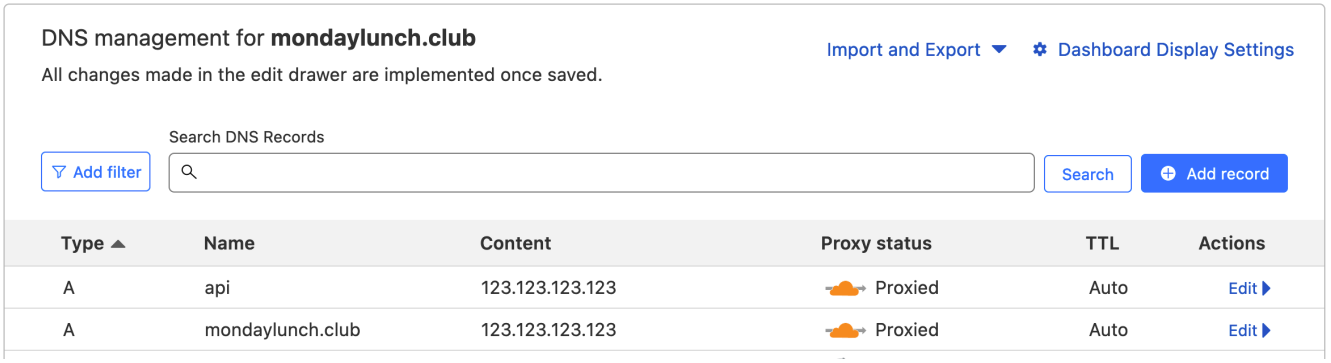
- Install Docker
- Install Docker Compose v3+

1. Configure server and DNS

To begin, find a server to deploy to, any VPS with a public IP works.

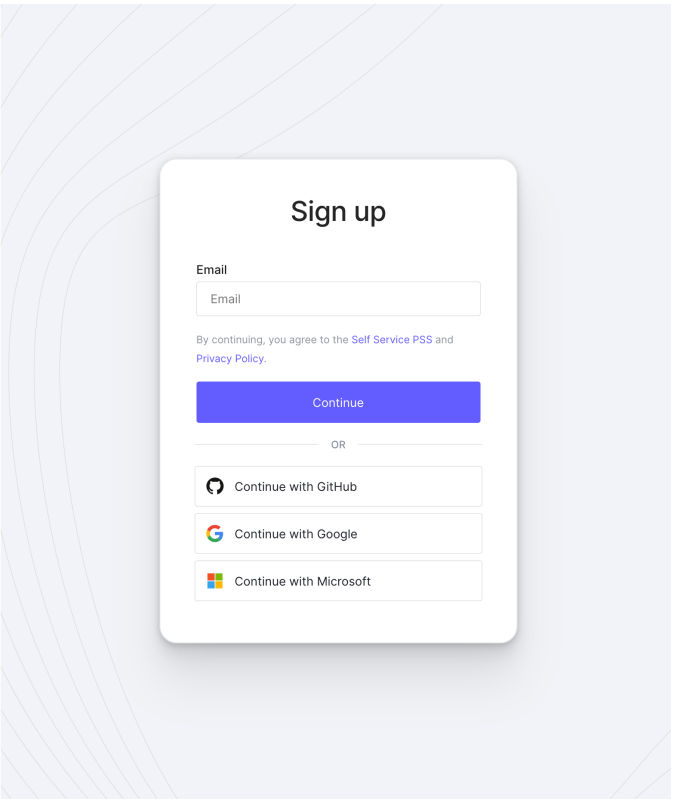
Configure two DNS records to point to your server:

- `A your.domain` to your IP (this will be referred to as `<frontend origin>`, in the example shown this is `mondaylunch.club`)
- `A api.your.domain` to your IP (this will be referred to as `<backend origin>`, in the example shown this is `api.mondaylunch.club`)

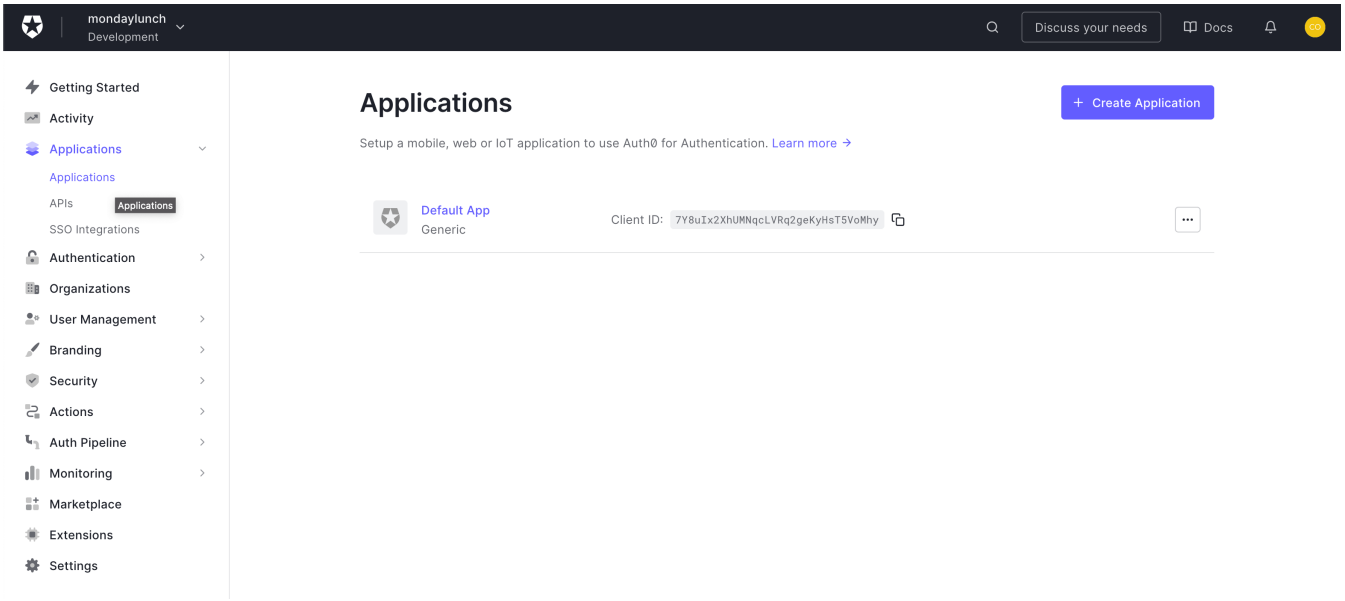


2. Configure Auth0

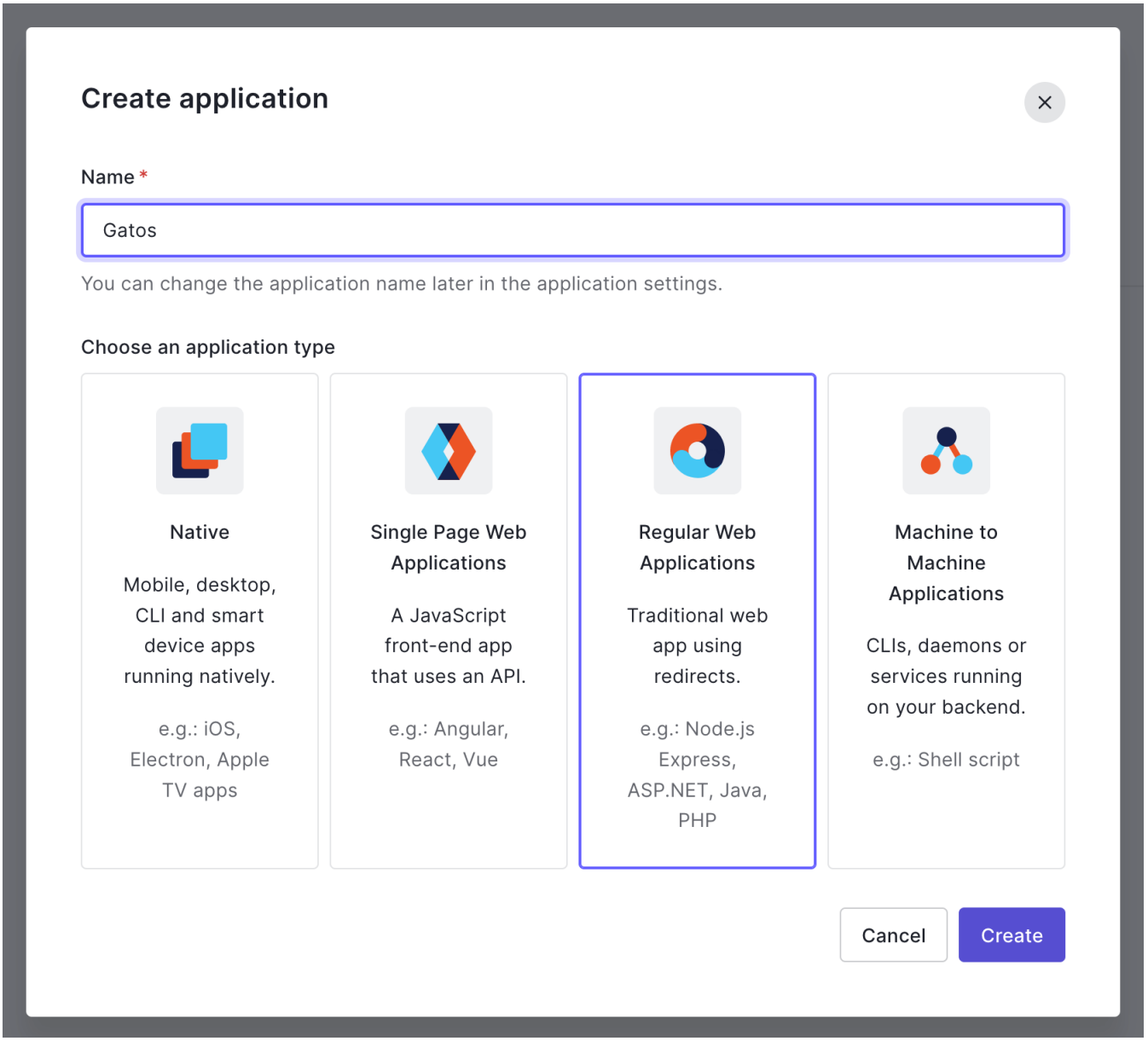
Go to https://auth0.com and create a new tenant by signing up.



Once logged in, go to "Applications" and select "+ Create Application":

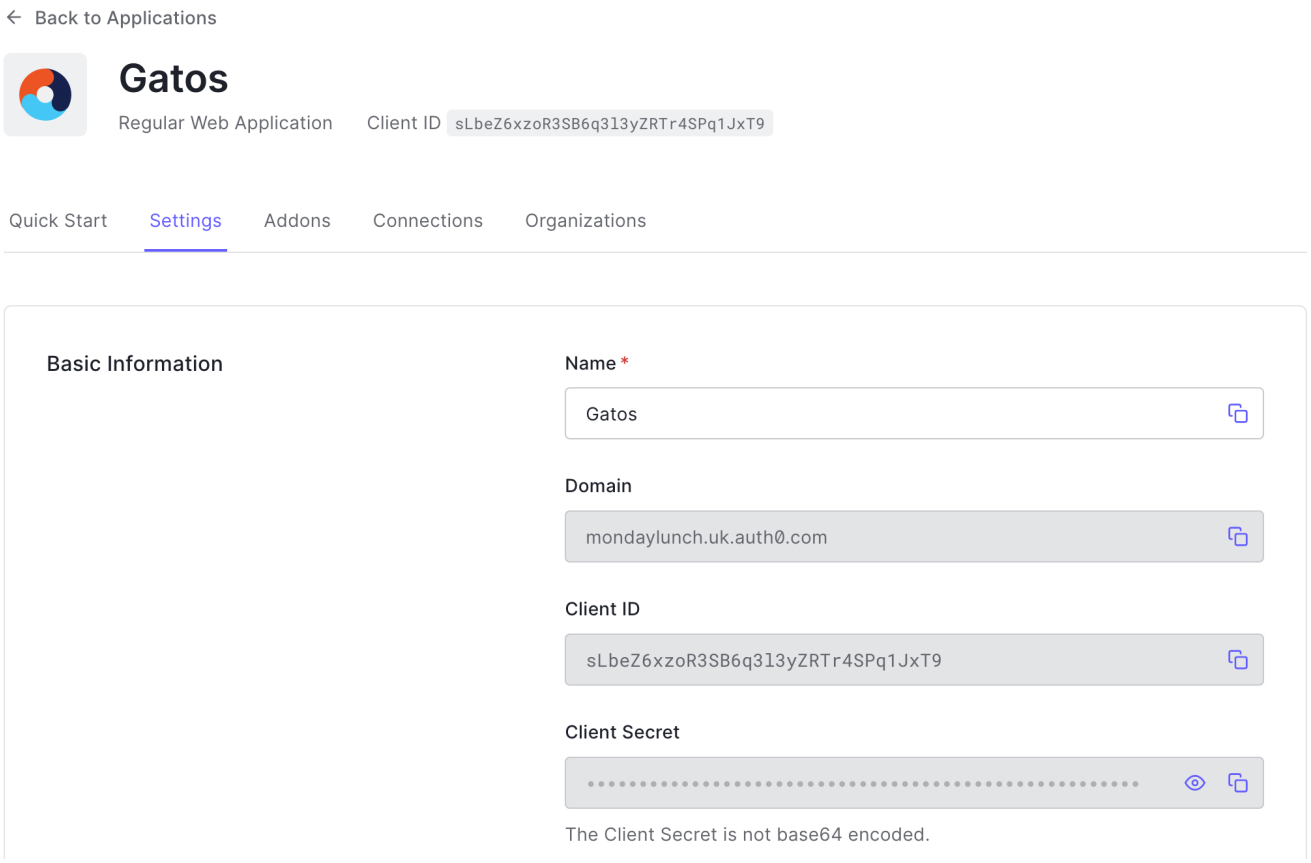


Select "Regular Web Application":



We will refer to this as the "frontend application".

Open settings and make note of the domain (referred to as `<your tenant>`), client ID, and client secret:



Scroll down to "Allowed Callback URLs" and set (replacing angled brackets with URLs from earlier):

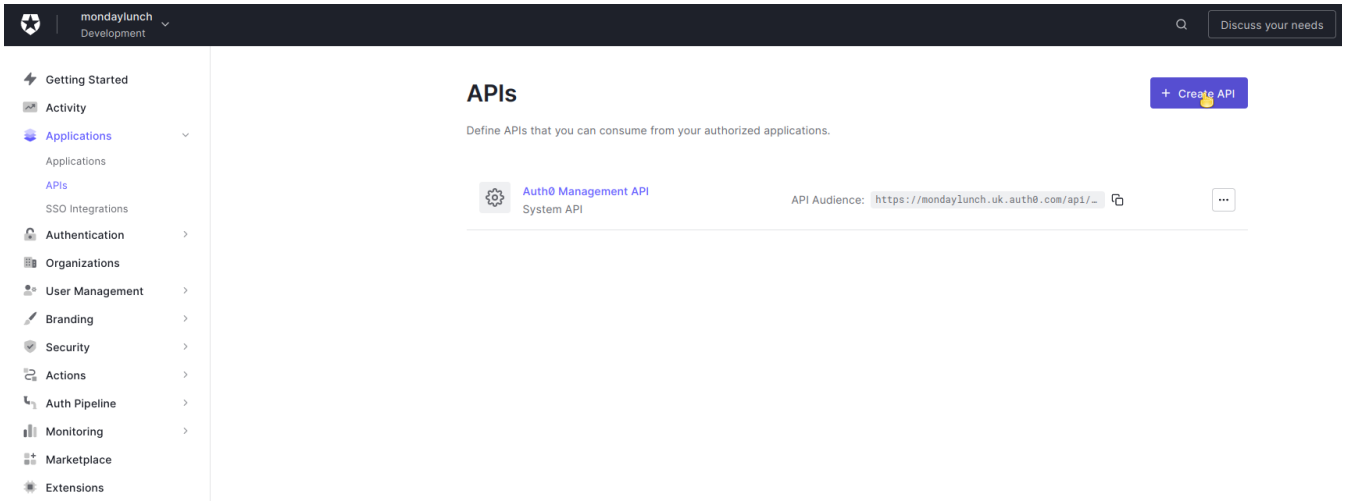
```
https://<backend origin>/login/oauth2/code/auth0, https://<frontend origin>/api/auth/callback/auth0
```

Also change "Allowed Logout URLs":

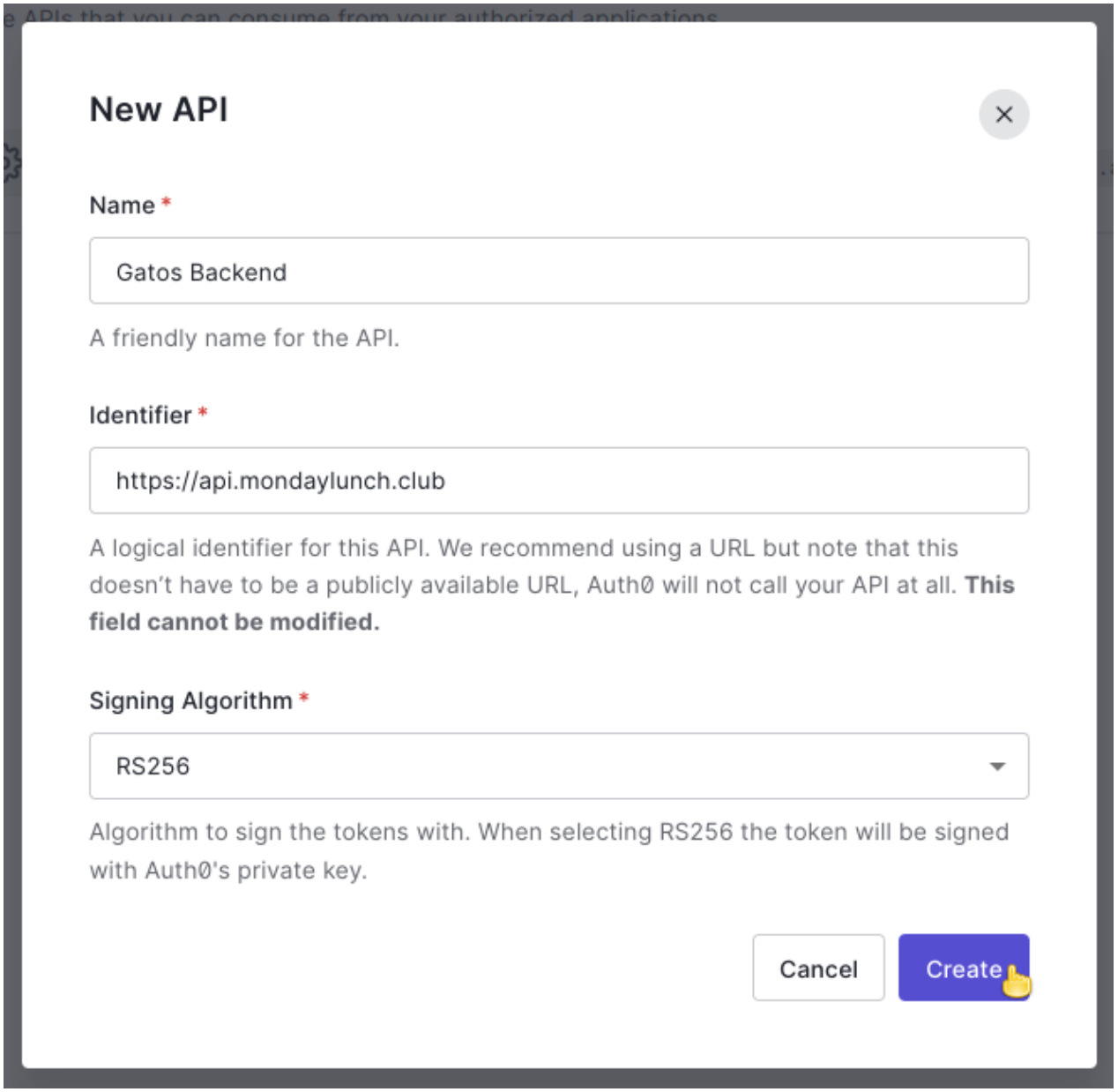
```
https://<frontend origin>
```

Then scroll to the end and save settings.

Now go to "APIs", and select "+ Create API":

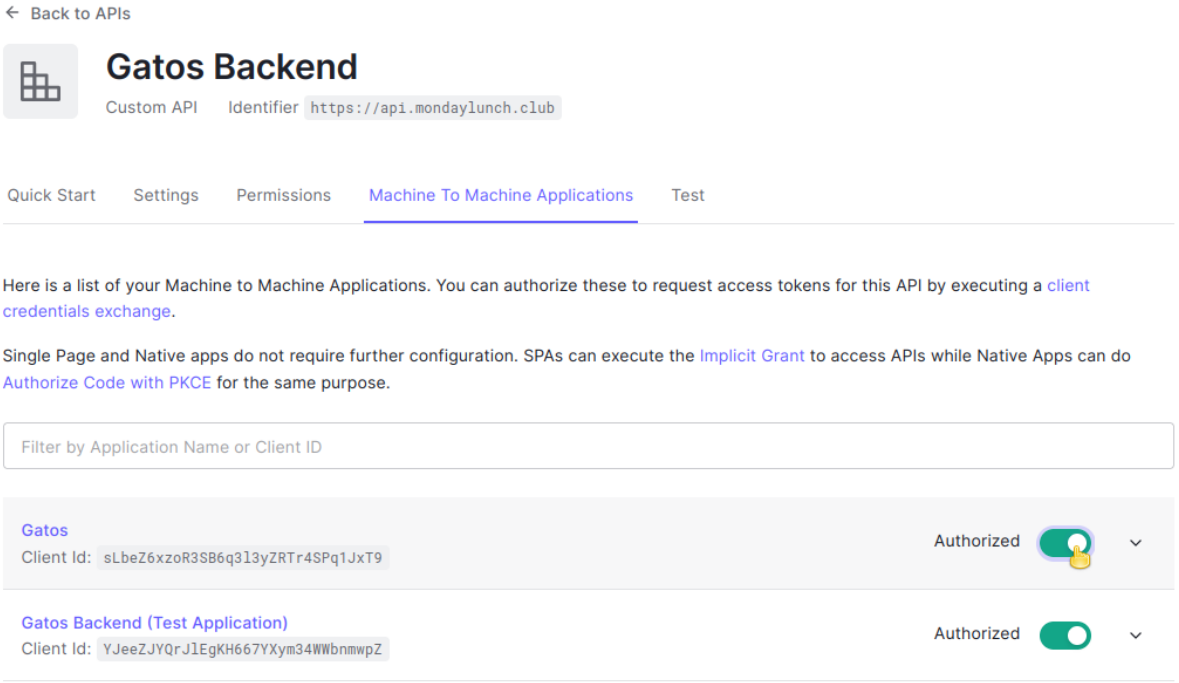


Fill out the details with a name and identifier (just set this to API origin, this will be the `audience url` later):

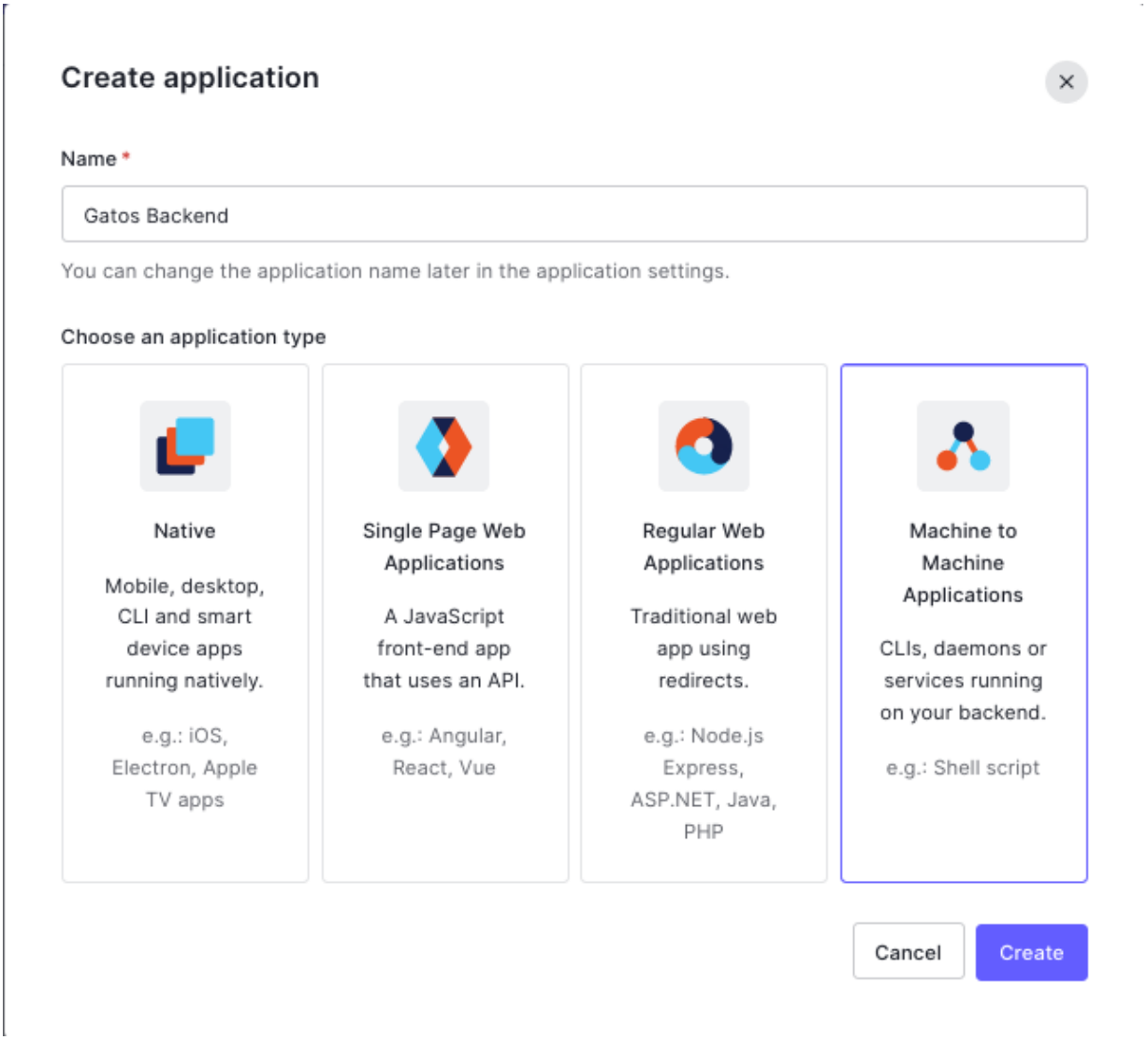


Now select the new API you just created, go to "Machine To Machine Applications" and

authorize the app created earlier.

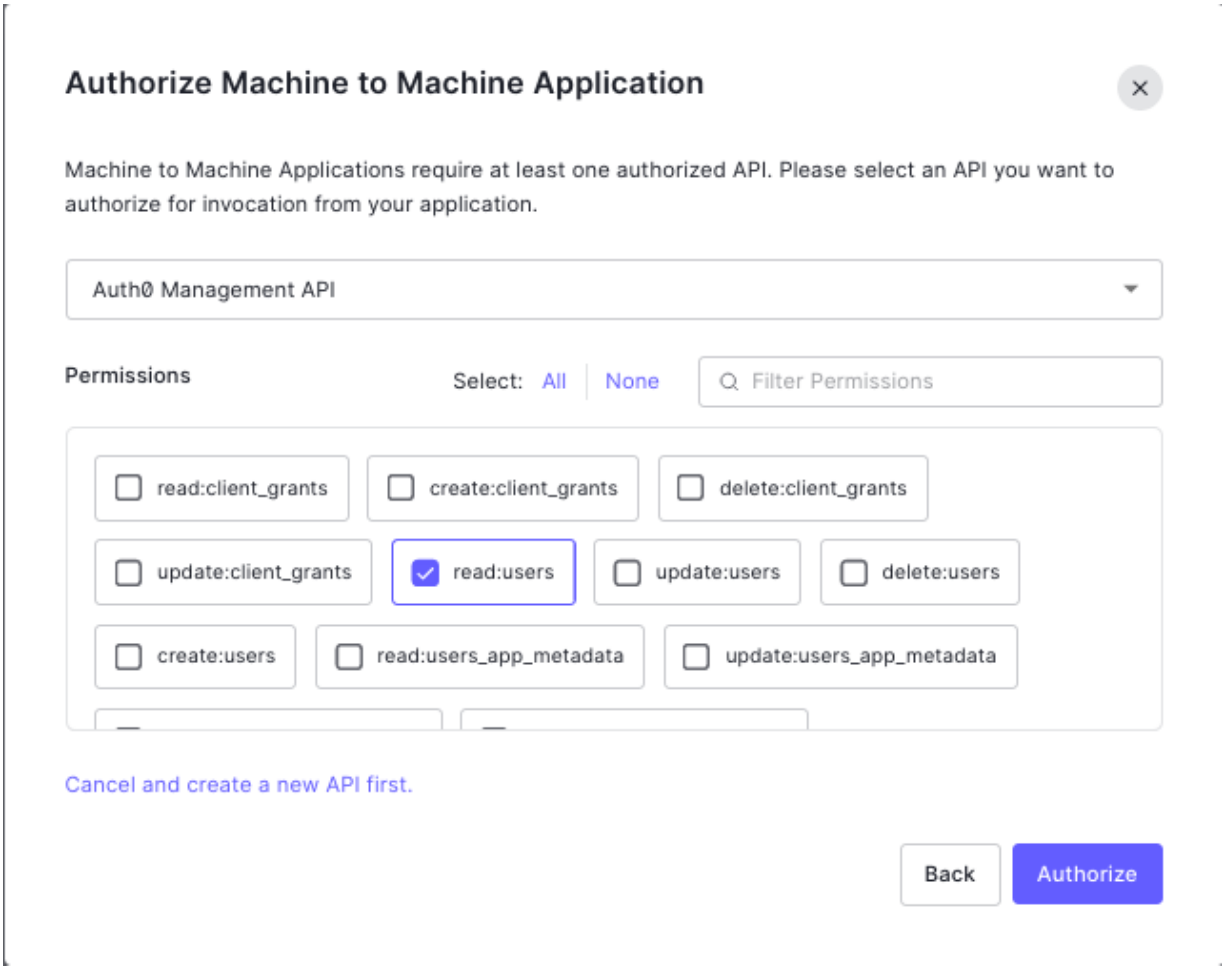


Next, create another Application. This time, select "Machine to Machine Applications":

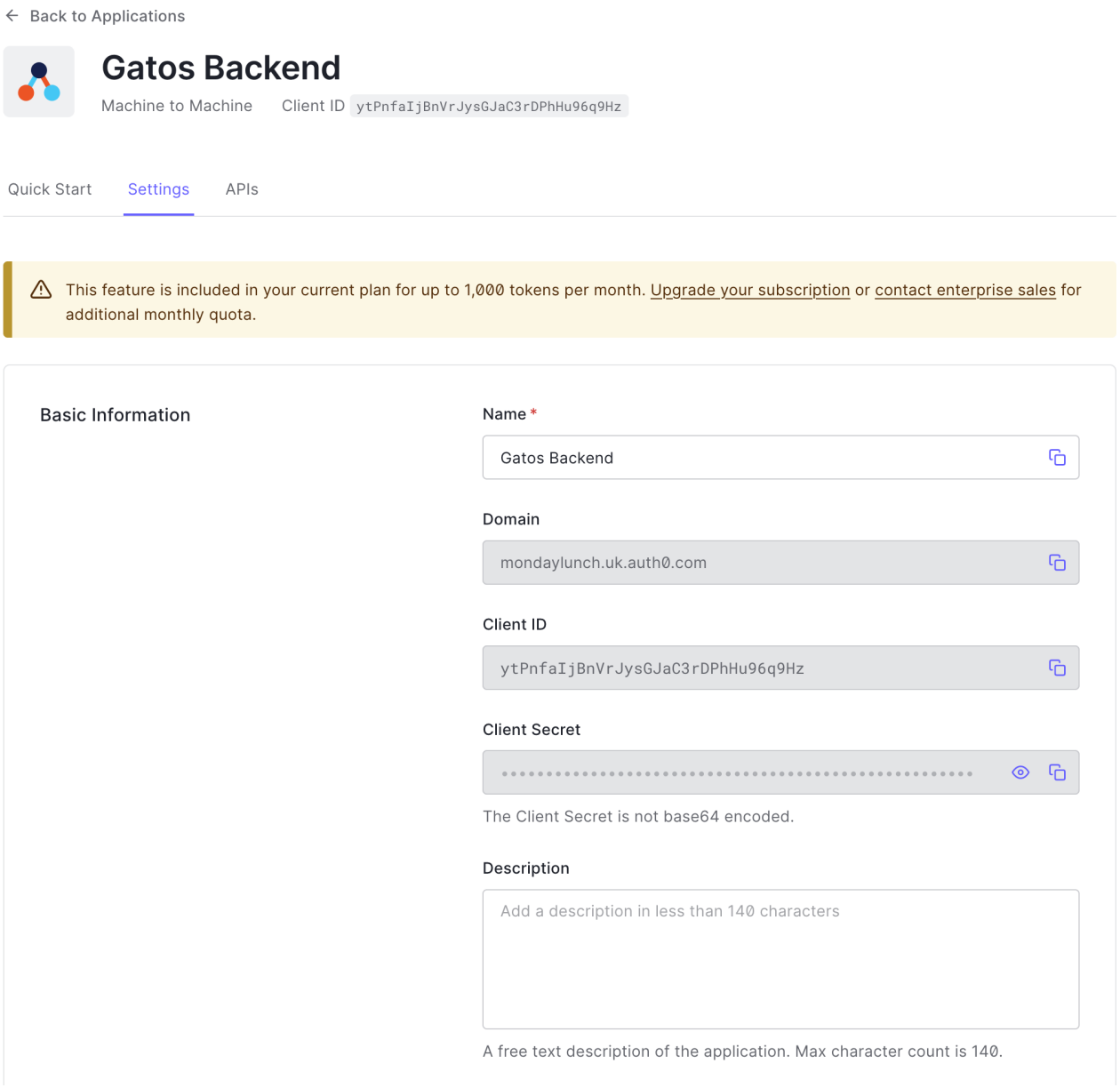


We will refer to this as the "backend application".

Give the backend application access to the `read:users` permission on the Auth0 Management API:



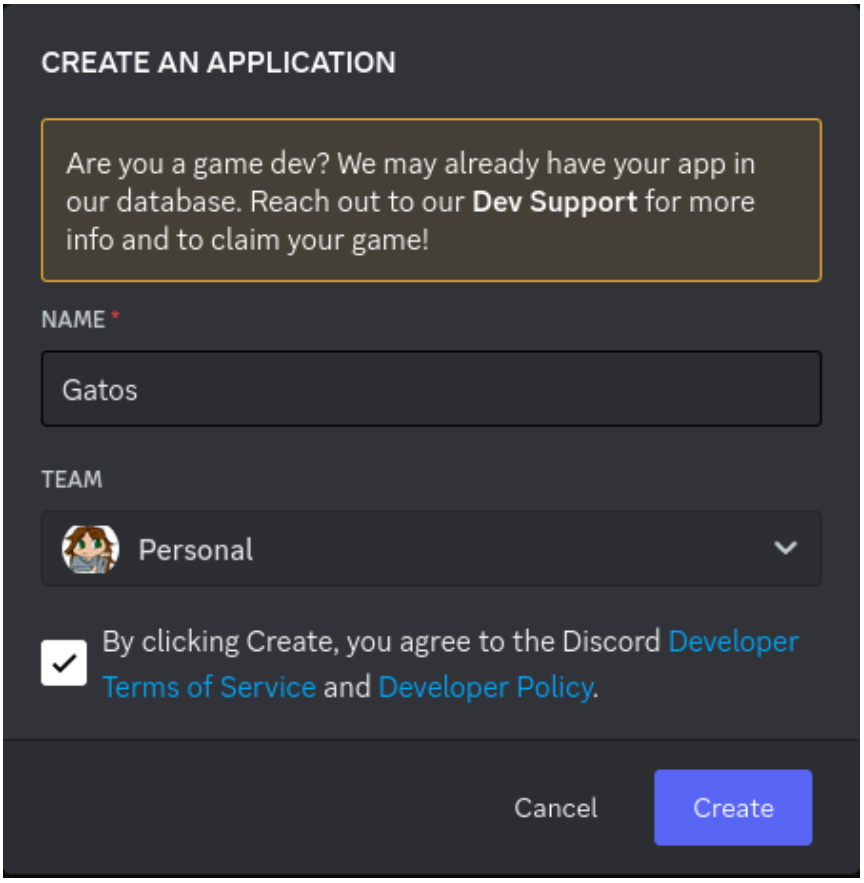
Open settings and make note of the client ID and client secret (this will be the `BACKEND_` prefixed variables later):



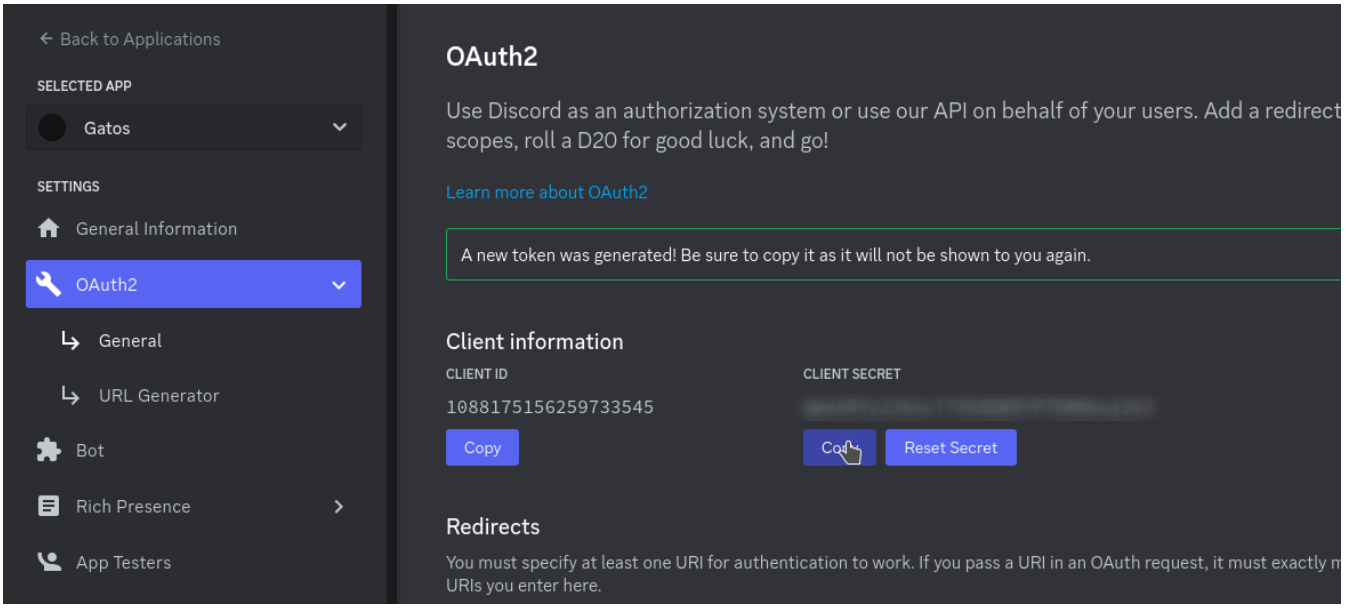
Auth0 is now set up.

3. Configure Discord

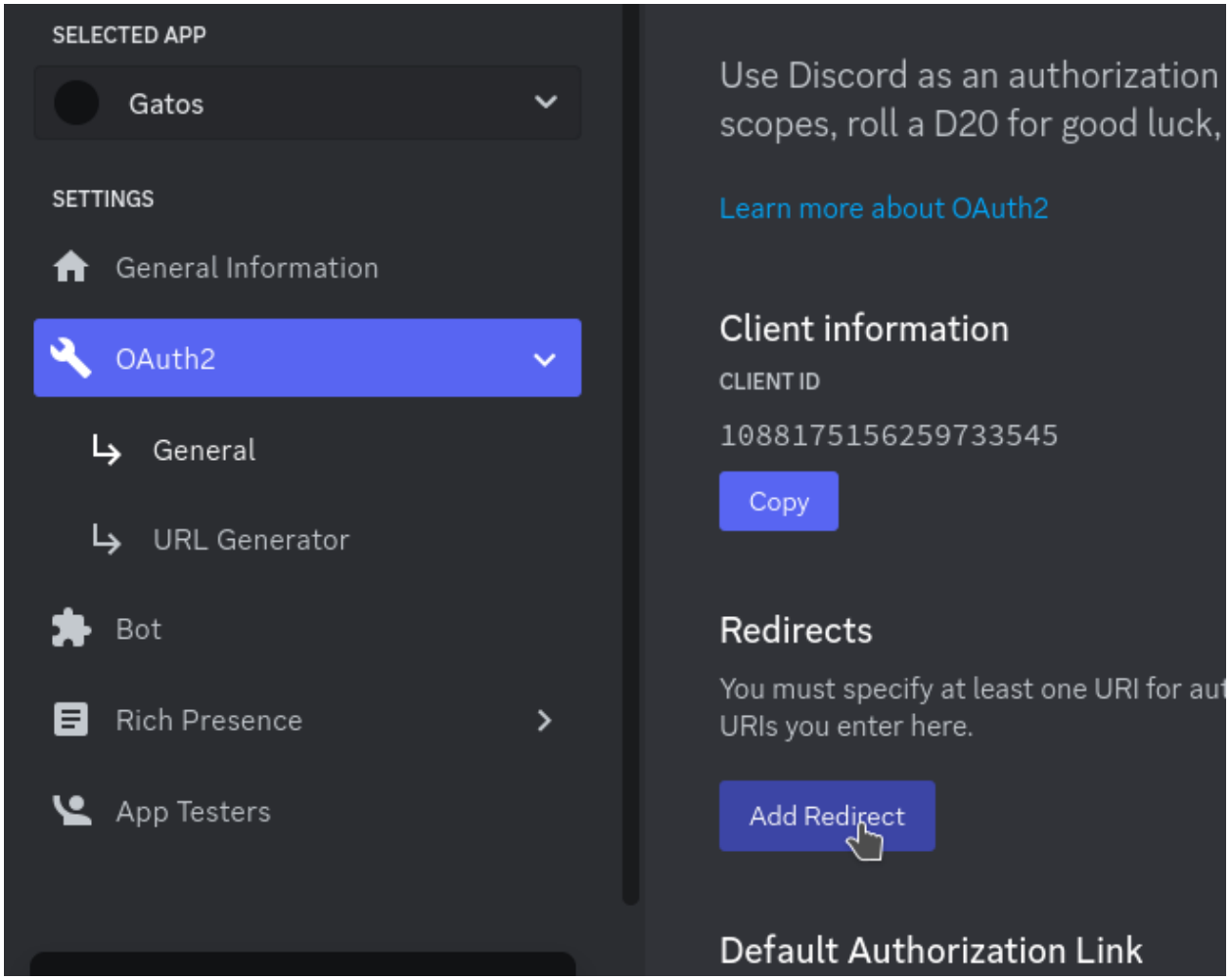
Go to https://discord.com/developers/applications and create a new application:



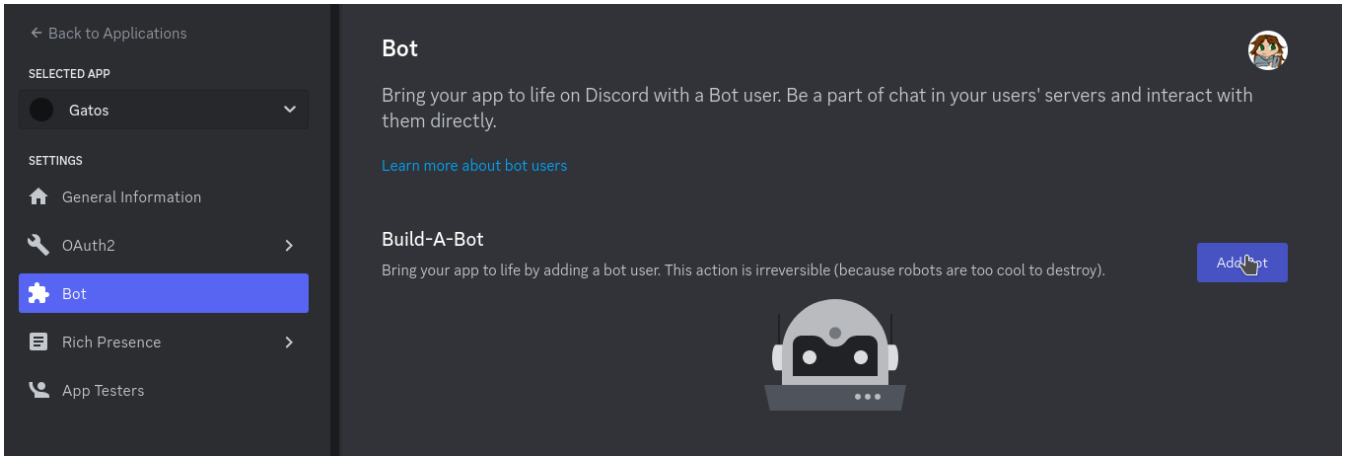
Navigate to the 'OAuth2' tab. Take a note of the client ID and client secret (referred to as `<Discord client ID>` and `<Discord client secret>`):



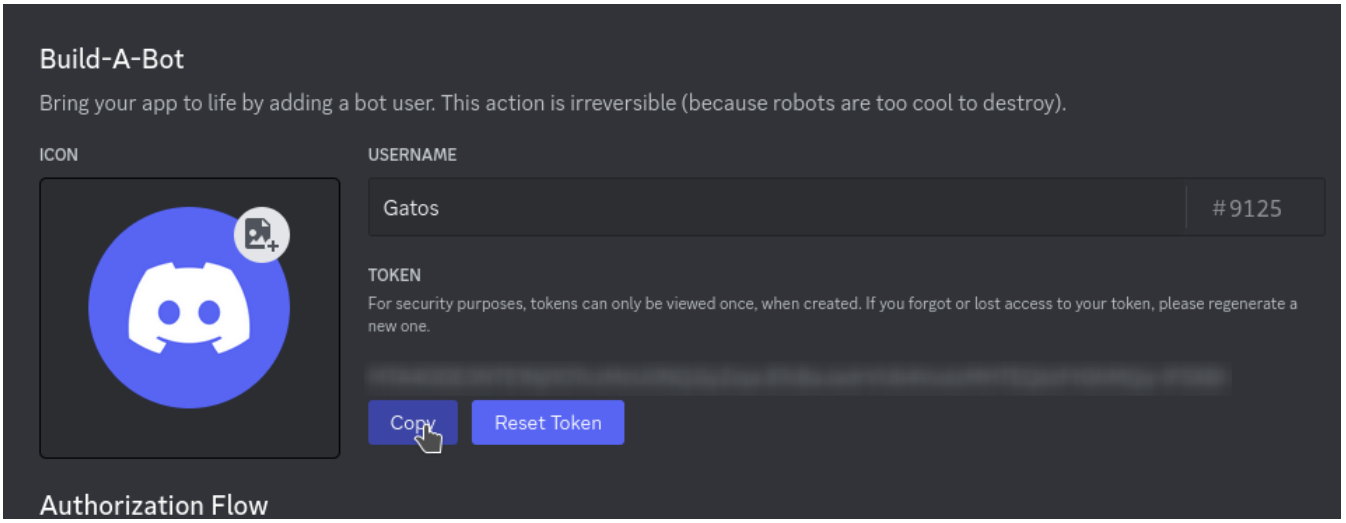
Add a redirect URI, `https://<your tenant>/login/callback`:



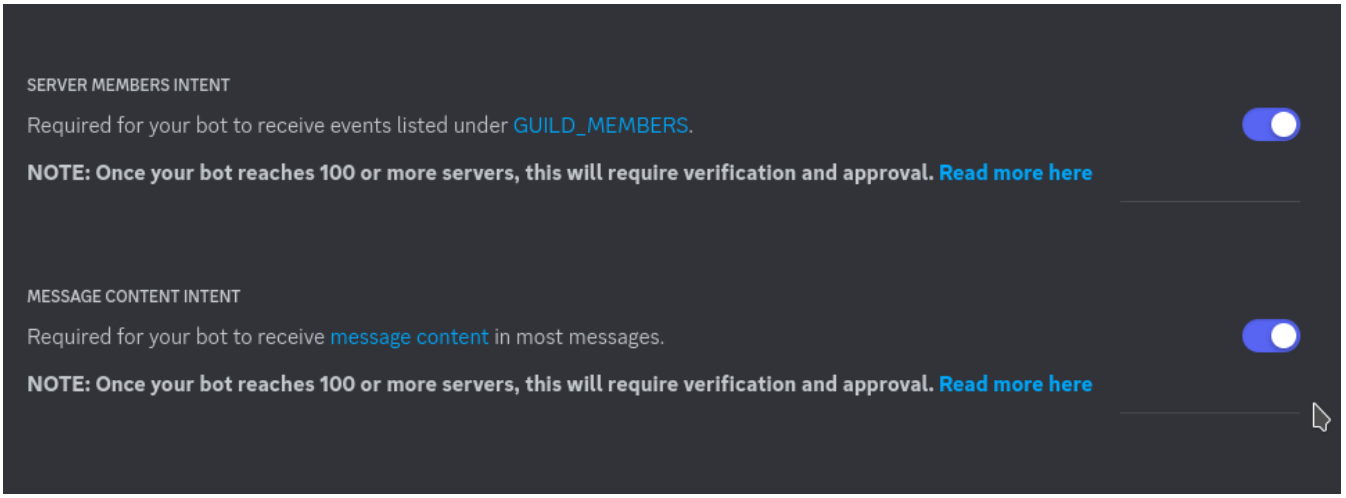
Navigate to the 'Bot' tab and create a new bot for the application:



Make a note of the Discord bot token (referred to as `<Discord bot token>`):

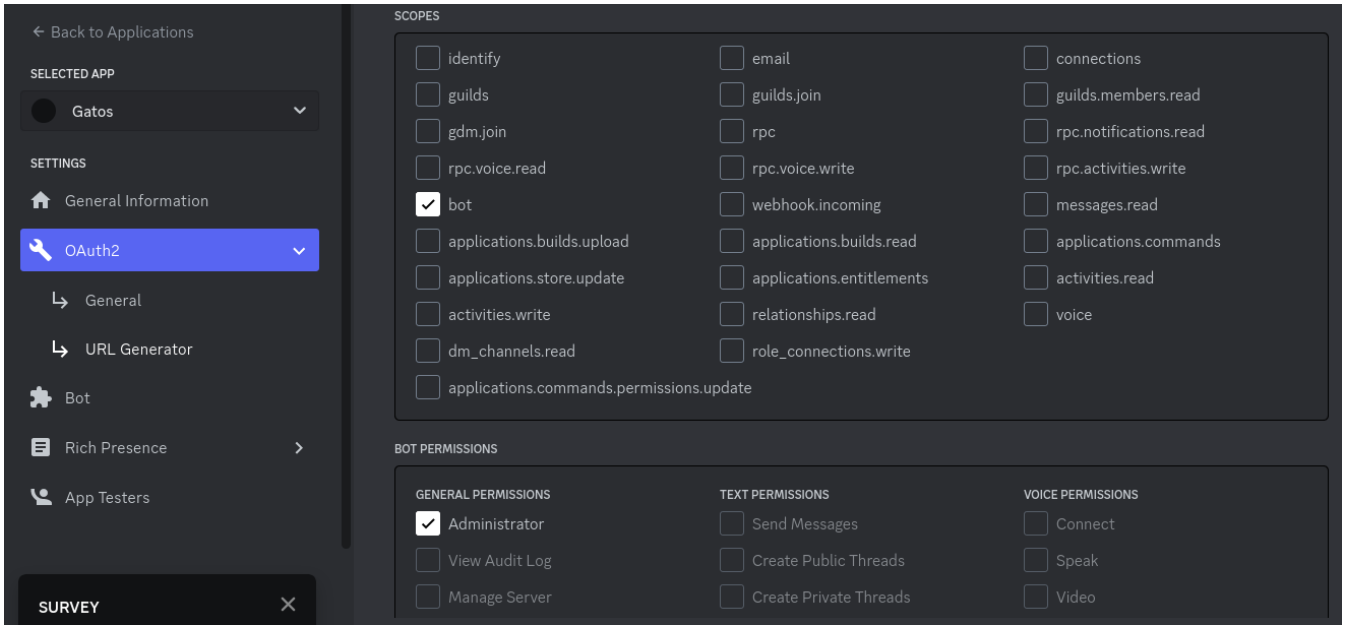


Next, enable the 'Server Members' and 'Message Content' in the 'Privileged Gateway Intents' section:

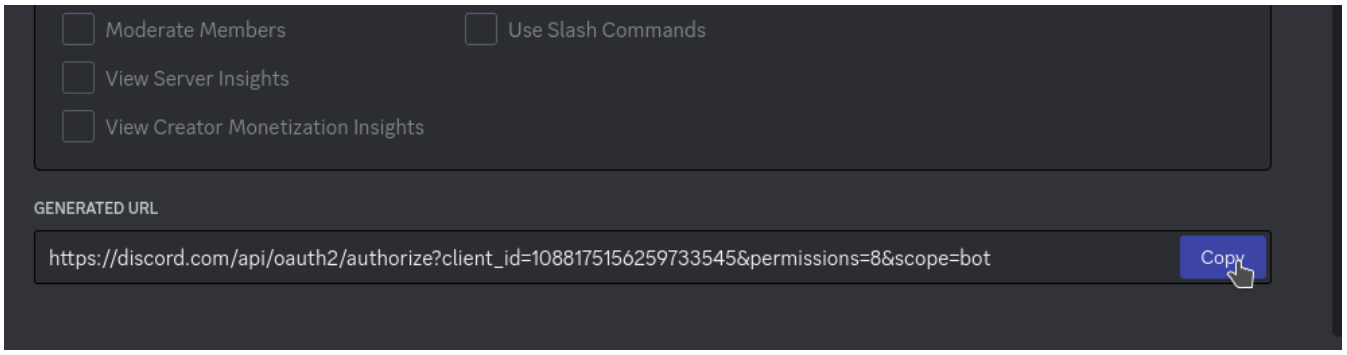


You can now add your bot to your server. Go to the 'OAuth2 -> URL Generator' tab, and select the 'bot' scope, with 'Administrator' permissions:

> **INFO**
>
> You can give the bot more restrictive permissions, but this is the easiest way to set it up.



Scroll down, and copy the generated link:

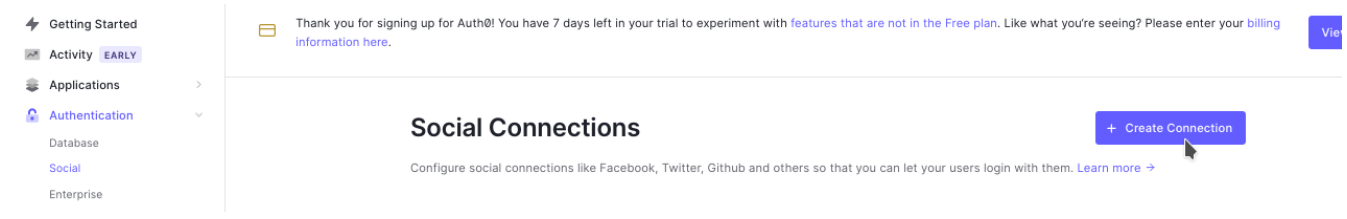


Open this link in a new tab, and select the server you want to add the bot to.

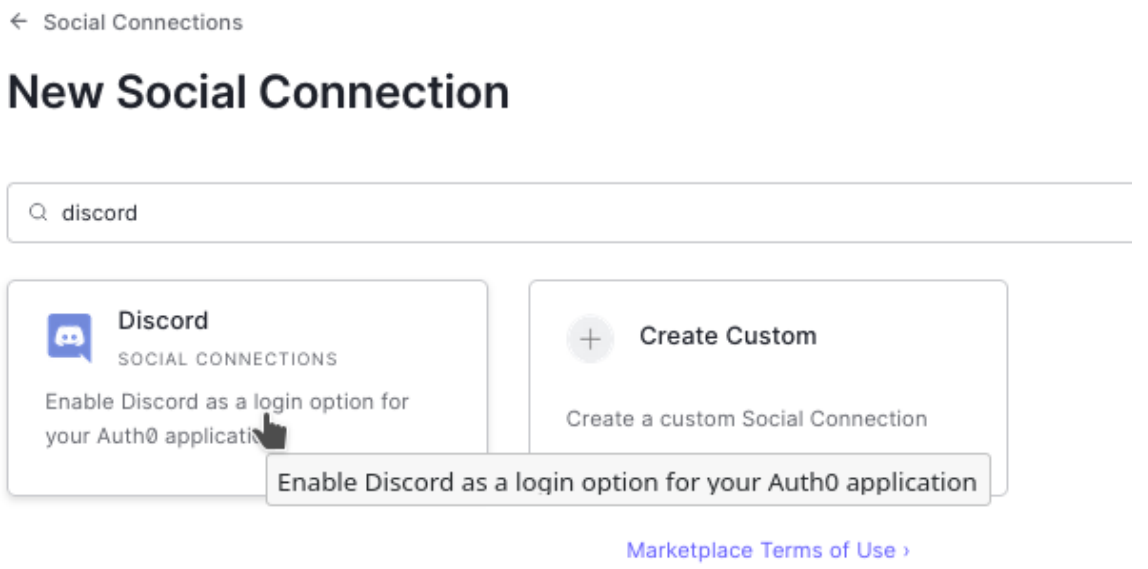
The discord bot is now set up.

4. Connect Auth0 and Discord

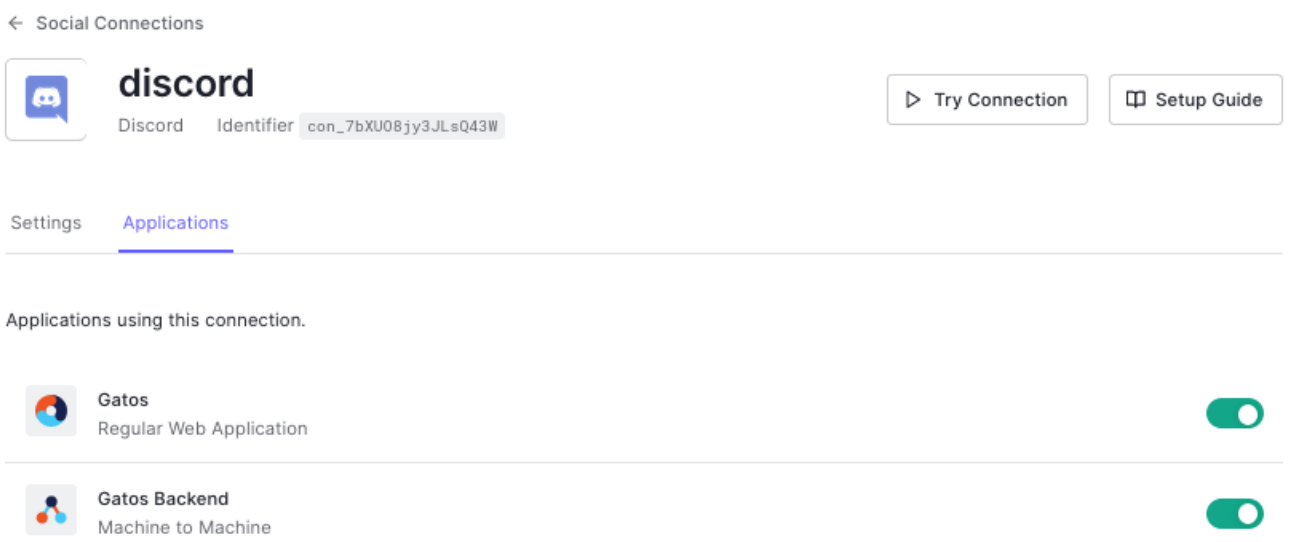
In the Auth0 Dashboard, navigate to the 'Authentication -> Social' tab. Create a new connection:



Select 'Discord' as the provider, and enter the Discord client ID and Discord client secret from earlier:



Then, enable the connection for your applications:



Auth0 will now be able to authenticate users with Discord.

5. Deploy

Create a new directory for the deployment.

Within the directory create a `docker-compose.yml`:

```
version: "3"

services:
  database:
```

```
    image: mongo
    restart: always
    volumes:
      - ./db:/data/db

  frontend:
    image: ghcr.io/mondaylunch/gatos-frontend:master
    env_file: .env
    depends_on:
      - database
    environment:
      - API_URL=http://backend:8080
      - AUTH_TRUST_HOST=1
    restart: always
    ports:
      # change left-hand side assignment if not suitable
      - 3000:3000

  api:
    image: ghcr.io/mondaylunch/gatos-backend-api:master
    env_file: .env
    environment:
      - MONGODB_URI=mongodb://database
    depends_on:
      - database
    restart: always
    ports:
      # change left-hand side assignment if not suitable
      - 8080:8080
```

And create an `.env` file and populate with your keys:

```
# Auth0 Tenant Information
AUTH0_TOKEN_URL=https://<your tenant>.auth0.com/oauth/token
AUTH0_ISSUER=https://<your tenant>.auth0.com/
AUTH0_AUTHORIZATION_LINK=https://<your tenant>.auth0.com/authorize
AUTH0_AUDIENCE=https://<audience url>
AUTH0_MANAGEMENT_AUDIENCE=https://<your tenant>.auth0.com/api/v2/

# Auth0 Client IDs
AUTH0_CLIENT_ID=<frontend application client ID>
BACKEND_AUTH0_CLIENT_ID=<backend application client ID>

# Auth0 Client Secrets
AUTH0_CLIENT_SECRET=<frontend application client secret>
BACKEND_AUTH0_CLIENT_SECRET=<backend application client secret>

# Discord Bot Token
DISCORD_TOKEN=<Discord bot token>
```

```
# Discord Invite URL
VITE_DISCORD_INVITE=https://discord.com/oauth2
/authorize?client_id=00000000000000000000&scope=bot%20applications.commands

# Backend API URL
VITE_API_URL=https://<backend origin>

# JWT Secret
# `openssl rand -base64 32`
# or go to: https://generate-secret.vercel.app/32
AUTH_SECRET=
```

INFO

For example, this might look like:

```
# Auth0 Tenant Information
AUTH0_TOKEN_URL=https://mondaylunch.uk.auth0.com/oauth/token
AUTH0_ISSUER=https://mondaylunch.uk.auth0.com/
AUTH0_AUTHORIZATION_LINK=https://mondaylunch.uk.auth0.com/authorize
AUTH0_AUDIENCE=https://api.mondaylunch.club
AUTH0_MANAGEMENT_AUDIENCE=https://mondaylunch.uk.auth0.com/api/v2/

# Auth0 Client ID
AUTH0_CLIENT_ID=sLbeZ6xzoR3SB6q3l3yZRTr4SPq1JxT9
BACKEND_AUTH0_CLIENT_ID=WRCrcJ4VKL32ZRTr4SPr33fCgjsd324

# Auth0 Client Secret
AUTH0_CLIENT_SECRET=supersecrettoken1234
BACKEND_AUTH0_CLIENT_SECRET=anothersupersecrettoken5678

# Discord Bot Token
DISCORD_TOKEN=1234.abcdef.ghij

# Discord Invite URL
VITE_DISCORD_INVITE=https://discord.com/oauth2
/authorize?client_id=1088446971871768586&permissions=8&
scope=bot%20applications.commands

# Backend API URL
VITE_API_URL=https://api.mondaylunch.club

# JWT Secret
# `openssl rand -base64 32`
# or go to: https://generate-secret.vercel.app/32
AUTH_SECRET=4MODj149L2v5fDv5L2mcJ65gxCTqeCDoDoMa2oVeKxA=
```

You can now start the software by running:

```
docker-compose up -d
```

To update the software, simply run:

```
docker-compose pull
```

And then run the start command again.

> **DANGER**
>
> Please check this page for any specific upgrade information before pulling new containers. Currently there are no special instructions, but some may appear here in the future.

6. Reverse Proxy

You must now reverse proxy:

- `http://<backend origin>` to `http://localhost:8080` or `http://api:8080` (within the container network).
- `http://<frontend origin>` to `http://localhost:3000` or `http://frontend:3000` (within the container network).

For example, you may wish to use Caddy to handle SSL automatically.

Caddy

To configure Caddy, first add the following service to `docker-compose.yml`:

```
caddy:
  image: caddy
  restart: always
  network_mode: host
  volumes:
    - ./Caddyfile:/etc/caddy/Caddyfile
    - ./caddy-data:/data
    - ./caddy-config:/config
```

Then create a corresponding `Caddyfile`:

```
<backend origin> {
  reverse_proxy api:8080
}

<frontend origin> {
  reverse_proxy frontend:3000
}
```

At this point, you can run `docker-compose up -d` again to start Caddy. You may also want to remove port allocations from the API and frontend.

> **INFO**
>
> For example, this might look like:
>
> ```
> https://api.mondaylunch.club {
>   reverse_proxy api:8080
> }
>
> https://mondaylunch.club {
>   reverse_proxy frontend:3000
> }
> ```