



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`)

DNS management for mondaylunch.club					
All changes made in the edit drawer are implemented once saved.					
Search DNS Records ▼ Add filter Search + Add record					
Type ▲	Name	Content	Proxy status	TTL	Actions
A	api	123.123.123.123	 Proxied	Auto	Edit ▶
A	mondaylunch.club	123.123.123.123	 Proxied	Auto	Edit ▶

2. Configure Auth0

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



Start building with your free plan

No credit card required.

© 2023 Okta, Inc. All Rights Reserved.



Sign up

Email

By continuing, you agree to the [Self Service PSS](#) and [Privacy Policy](#).

Continue

OR



Continue with GitHub



Continue with Google



Continue with Microsoft

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

The screenshot shows the Auth0 dashboard interface. At the top, there's a dark header with the Auth0 logo, a user profile dropdown for 'mondaylunch Development', a search bar, and links for 'Discuss your needs', 'Docs', and a notification bell. The left sidebar contains a navigation menu with items like 'Getting Started', 'Activity', 'Applications' (highlighted), 'APIs', 'SSO Integrations', 'Authentication', 'Organizations', 'User Management', 'Branding', 'Security', 'Actions', 'Auth Pipeline', 'Monitoring', 'Marketplace', 'Extensions', and 'Settings'. The main content area is titled 'Applications' and includes a '+ Create Application' button. Below the title, there's a sub-header 'Setup a mobile, web or IoT application to use Auth0 for Authentication. [Learn more](#)'. A table lists the applications, showing one entry: 'Default App' with a 'Generic' type and a 'Client ID' of '7Y8uIx2XhUMNqclVRq2geKyHsTSVoMhy'. A three-dot menu icon is visible next to the Client ID.

Select "Regular Web Application":

Create application



Name *

Gatos

You can change the application name later in the application settings.

Choose an application type



Native

Mobile, desktop,
CLI and smart
device apps
running natively.

e.g.: iOS,
Electron, Apple
TV apps



Single Page Web Applications

A JavaScript
front-end app
that uses an API.

e.g.: Angular,
React, Vue



Regular Web Applications

Traditional web
app using
redirects.

e.g.: Node.js
Express,
ASP.NET, Java,
PHP



Machine to Machine Applications

CLIs, daemons or
services running
on your backend.

e.g.: Shell script

Cancel

Create

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:

[← Back to Applications](#)



Gatos

Regular Web Application

Client ID `sLbeZ6xzoR3SB6q3l3yZRTTr4SPq1JxT9`

[Quick Start](#)

[Settings](#)

[Addons](#)

[Connections](#)

[Organizations](#)

Basic Information

Name *

Gatos

Domain

mondaylunch.uk.auth0.com

Client ID

sLbeZ6xzoR3SB6q3l3yZRTTr4SPq1JxT9

Client Secret

.....



The Client Secret is not base64 encoded.

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

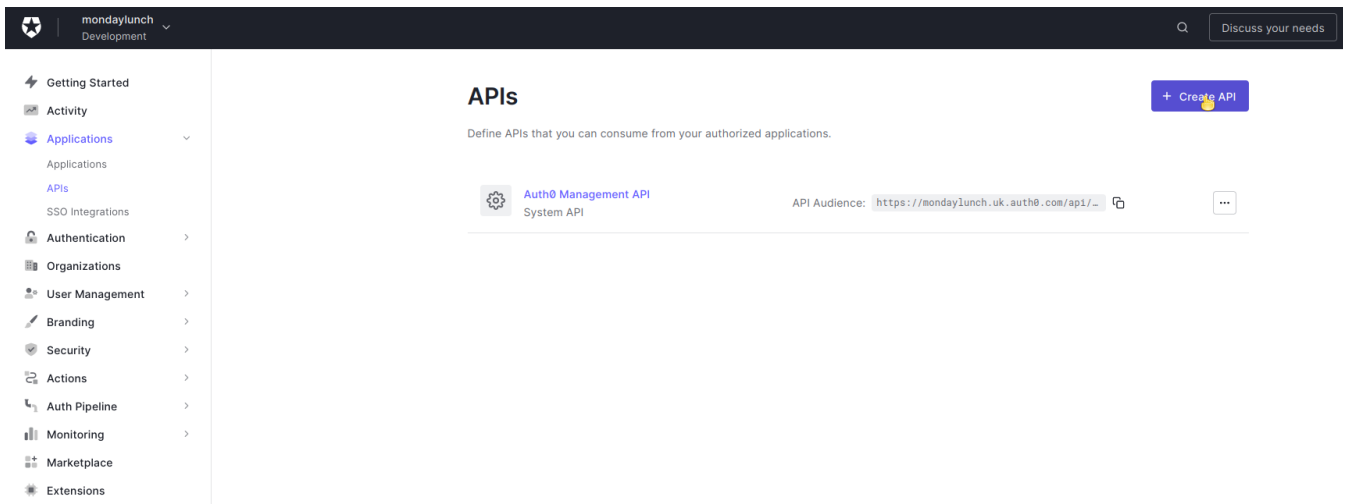
```
https://<backend origin>/login/oauth2/code/auth0, https://<frontend origin>/api/auth/callback/auth0
```

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):

The screenshot shows the 'New API' form. It has a title 'New API' and a close button (X). The form contains three fields: 'Name' with the value 'Gatos Backend', 'Identifier' with the value 'https://api.mondaylunch.club', and 'Signing Algorithm' with the value 'RS256'. Below the 'Identifier' field is a note: 'A logical identifier for this API. We recommend using a URL but note that this doesn't have to be a publicly available URL, Auth0 will not call your API at all. This field cannot be modified.' At the bottom right are 'Cancel' and 'Create' buttons.

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

authorize the app created earlier.

[← Back to APIs](#)



Gatos Backend

Custom API Identifier `https://api.mondaylunch.club`

[Quick Start](#) [Settings](#) [Permissions](#) [Machine To Machine Applications](#) [Test](#)

Here is a list of your Machine to Machine Applications. You can authorize these to request access tokens for this API by executing a [client credentials exchange](#).

Single Page and Native apps do not require further configuration. SPAs can execute the [Implicit Grant](#) to access APIs while Native Apps can do [Authorize Code with PKCE](#) for the same purpose.

Gatos Client Id: <code>sLbeZ6xzoR3SB6q3l3yZRT4SPq1JxT9</code>	Authorized 	▼
Gatos Backend (Test Application) Client Id: <code>YJeeZJYQrJlEgKH667YXym34WWbnmwpZ</code>	Authorized 	▼

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

Create application



Name *

Gatos Backend

You can change the application name later in the application settings.

Choose an application type



Native

Mobile, desktop, CLI and smart device apps running natively.

e.g.: iOS, Electron, Apple TV apps



Single Page Web Applications

A JavaScript front-end app that uses an API.

e.g.: Angular, React, Vue



Regular Web Applications

Traditional web app using redirects.

e.g.: Node.js Express, ASP.NET, Java, PHP



Machine to Machine Applications

CLIs, daemons or services running on your backend.

e.g.: Shell script

Cancel

Create

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

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

Authorize Machine to Machine Application



Machine to Machine Applications require at least one authorized API. Please select an API you want to authorize for invocation from your application.

Auth0 Management API

Permissions

Select: [All](#) | [None](#)

☐ read:client_grants

☐ create:client_grants

☐ delete:client_grants

☐ update:client_grants

☒ read:users

☐ update:users

☐ delete:users

☐ create:users

☐ read:users_app_metadata

☐ update:users_app_metadata

[Cancel and create a new API first.](#)

Back

Authorize

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

← Back to Applications



Gatos Backend

Machine to Machine

Client ID ytPnfaIjBnVrJysGJaC3rDPhHu96q9Hz

Quick Start

Settings

APIs

⚠ This feature is included in your current plan for up to 1,000 tokens per month. [Upgrade your subscription](#) or [contact enterprise sales](#) for additional monthly quota.

Basic Information

Name *

Gatos Backend

Domain

mondaylunch.uk.auth0.com

Client ID

ytPnfaIjBnVrJysGJaC3rDPhHu96q9Hz

Client Secret

.....



The Client Secret is not base64 encoded.

Description

Add a description in less than 140 characters

A free text description of the application. Max character count is 140.

Auth0 is now set up.

3. Configure Discord

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

CREATE AN APPLICATION

Are you a game dev? We may already have your app in our database. Reach out to our **Dev Support** for more info and to claim your game!

NAME *

Gatos

TEAM

 Personal

☒ By clicking Create, you agree to the Discord [Developer Terms of Service](#) and [Developer Policy](#).

Cancel Create

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>`):

← Back to Applications

SELECTED APP

Gatos

SETTINGS

General Information

OAuth2

General

URL Generator

Bot

Rich Presence

App Testers

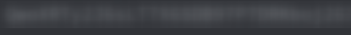
OAuth2

Use Discord as an authorization system or use our API on behalf of your users. Add a redirect scopes, roll a D20 for good luck, and go!

[Learn more about OAuth2](#)

A new token was generated! Be sure to copy it as it will not be shown to you again.

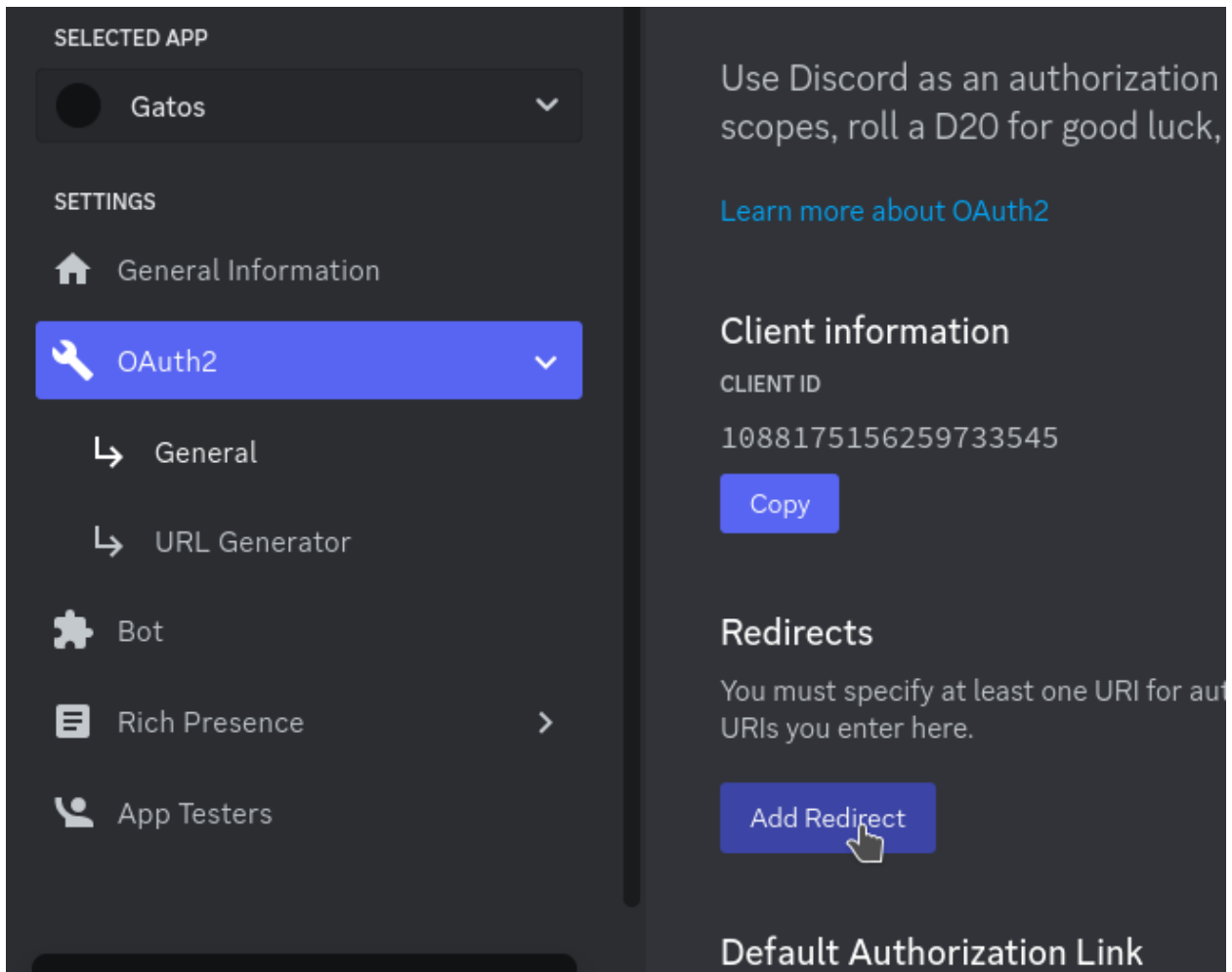
Client information

CLIENT ID	CLIENT SECRET
1088175156259733545	
Copy	Copy Reset Secret

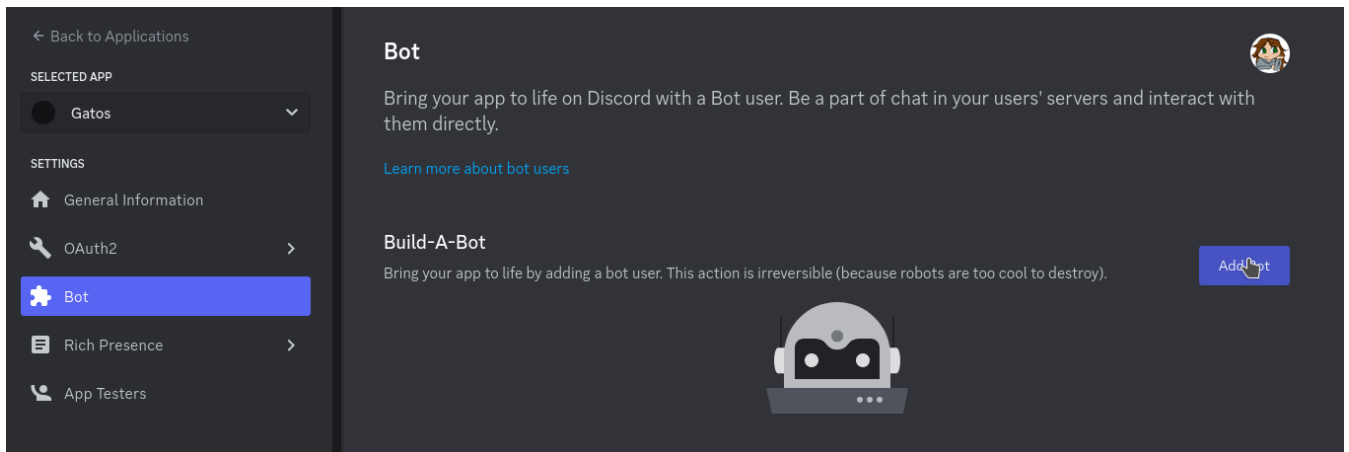
Redirects

You must specify at least one URI for authentication to work. If you pass a URI in an OAuth request, it must exactly match one of the URIs you enter here.

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



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

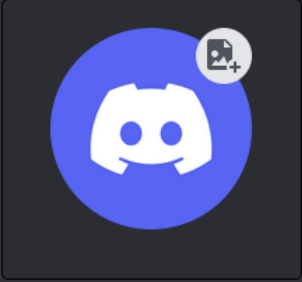


Make a note of the Discord bot token (referred to as `<Discord bot token>`):

Build-A-Bot

Bring your app to life by adding a bot user. This action is irreversible (because robots are too cool to destroy).

ICON



USERNAME

Gatos

#9125

TOKEN

For security purposes, tokens can only be viewed once, when created. If you forgot or lost access to your token, please regenerate a new one.

CopyReset Token

Authorization Flow

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

SERVER MEMBERS INTENT

Required for your bot to receive events listed under [GUILD_MEMBERS](#).

NOTE: Once your bot reaches 100 or more servers, this will require verification and approval. [Read more here](#)

MESSAGE CONTENT INTENT

Required for your bot to receive [message content](#) in most messages.

NOTE: Once your bot reaches 100 or more servers, this will require verification and approval. [Read more here](#)

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.

← Back to Applications

SELECTED APP

Gatos

SETTINGS

General Information

OAuth2

General

URL Generator

Bot

Rich Presence

App Testers

SURVEY

SCOPES

- ☐ identify
- ☐ email
- ☐ connections
- ☐ guilds
- ☐ guilds.join
- ☐ guilds.members.read
- ☐ gdm.join
- ☐ rpc
- ☐ rpc.notifications.read
- ☐ rpc.voice.read
- ☐ rpc.voice.write
- ☐ rpc.activities.write
- ☒ bot
- ☐ webhook.incoming
- ☐ messages.read
- ☐ applications.builds.upload
- ☐ applications.builds.read
- ☐ applications.commands
- ☐ applications.store.update
- ☐ applications.entitlements
- ☐ activities.read
- ☐ activities.write
- ☐ relationships.read
- ☐ voice
- ☐ dm_channels.read
- ☐ role_connections.write
- ☐ applications.commands.permissions.update

BOT PERMISSIONS

GENERAL PERMISSIONS

- ☒ Administrator
- ☐ View Audit Log
- ☐ Manage Server

TEXT PERMISSIONS

- ☐ Send Messages
- ☐ Create Public Threads
- ☐ Create Private Threads

VOICE PERMISSIONS

- ☐ Connect
- ☐ Speak
- ☐ Video

Scroll down, and copy the generated link:

GENERATED URL

https://discord.com/api/oauth2/authorize?client_id=1088175156259733545&permissions=8&scope=bot [Copy](#)

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:

Getting Started

Activity **EARLY**

Applications

Authentication

Database

Social

Enterprise

Thank you for signing up for Auth0! You have 7 days left in your trial to experiment with features that are not in the Free plan. Like what you're seeing? Please enter your billing information here. [View](#)

Social Connections

Configure social connections like Facebook, Twitter, Github and others so that you can let your users login with them. [Learn more](#)

[+ Create Connection](#)

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

New Social Connection

**Discord**
SOCIAL CONNECTIONS
Enable Discord as a login option for your Auth0 application

**Create Custom**
Create a custom Social Connection

Enable Discord as a login option for your Auth0 application

[Marketplace Terms of Use](#)

Then, enable the connection for your applications:

**discord**
Discord Identifier `con_7bXU08jy3JLsQ43W`

[▶ Try Connection](#)

[📖 Setup Guide](#)

Applications using this connection.

	Gatos Regular Web Application	☒
	Gatos Backend Machine to Machine	☒

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=WRCrcJ4VKL32ZRTTr4SPR33fCgjsd324

# 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  
}
```

 [Edit this page](#)