Project Report

Introduction

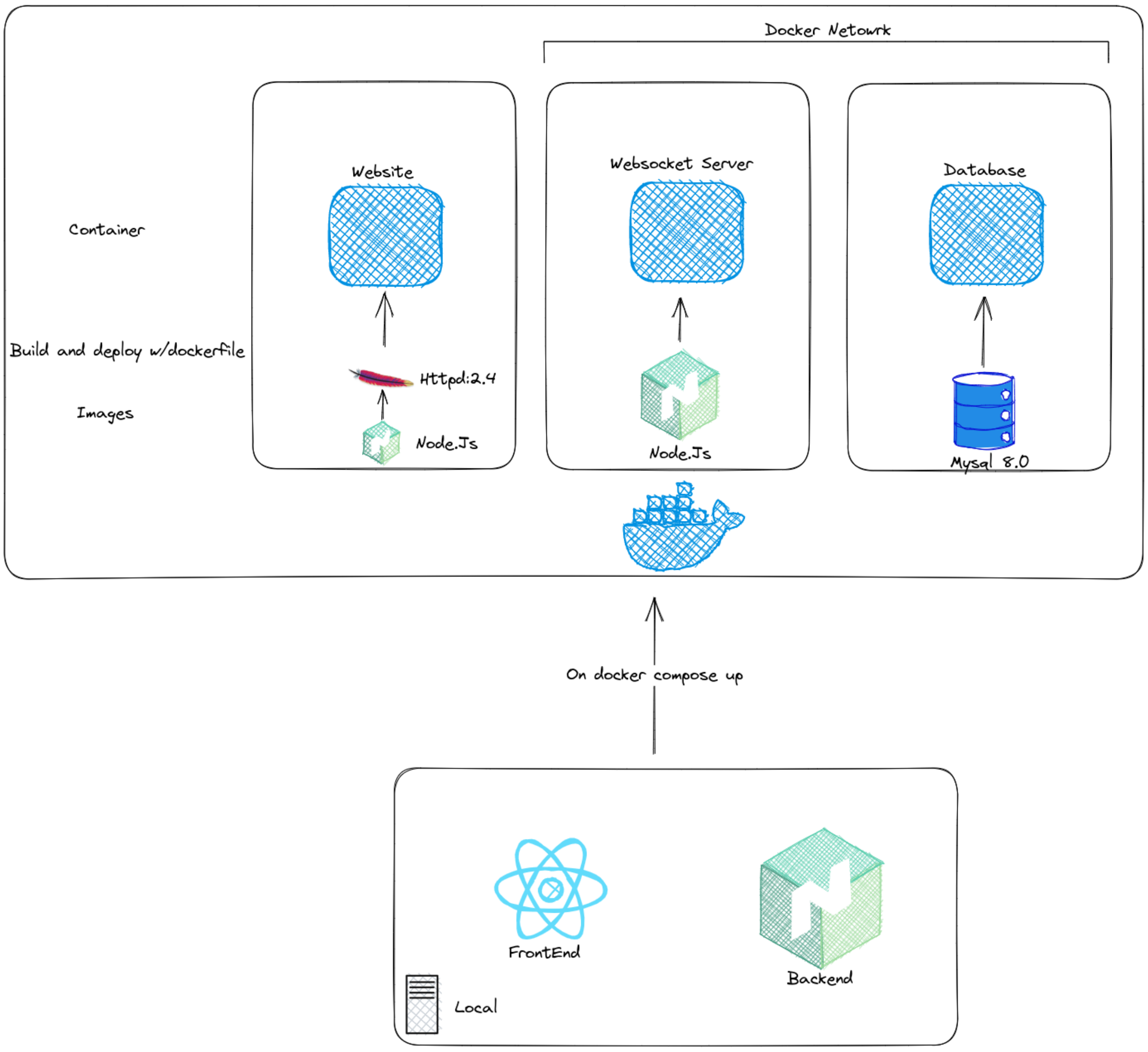
The projects consists of a platform that receives information via websockets and distributes them to the correct connected clients. To visualize the entire process there is a web platform that receives all the information. To log all of the communication there is a database to save everything.

The mais focus of the website and backend is to be fast while the database is to be reliable.

Scenarios

This project consists of two scenario, one for local deployment that can also be used for development, and one that allows the project to be deployed on the cloud more focused in the production environment.

Local



The local scenario is possible due to a docker-compose that build all the images, deploys containers the containers and associate them to a docker network. The this setup requires the backend.

Frontend(website)

The first service This service exposes port 80 and maps it to the host machine's port 80. It build the image in a node image and then it copies the static file to a httpd containers that hosts the files. By doing it this way and not in a node container we can that test the website as if it was in production.

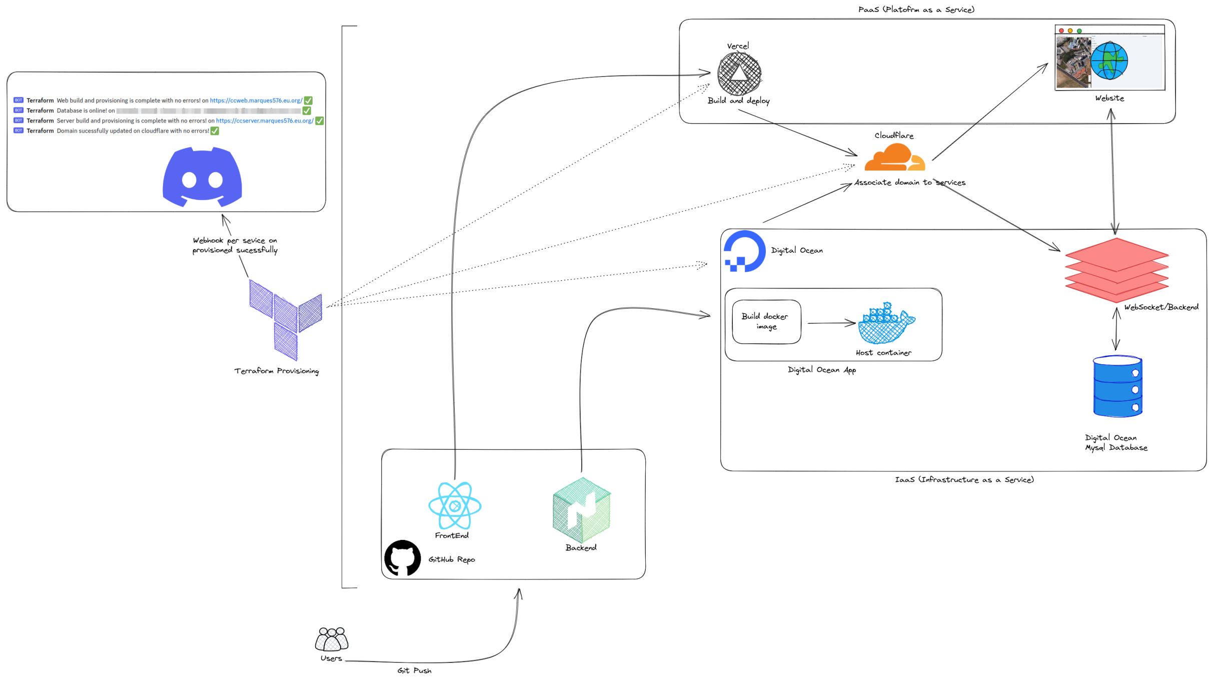
Backend(socket server)

The backend also run a Dockerfile and some environment variables used to configure the service. This service runs in a node container and mapps the 8080 port of the host to the 8080 of the container respectively. It waits before the database is launched to launch itself.

Database

The third service uses the latest version of the MySQL Docker image. It maps port 3306 to the host machine's port 3306.

Cloud


BOT **Terraform** Web build and provisioning is complete with no errors! on https://ccweb.marques576.eu.org/ ✅
BOT **Terraform** Database is online! on [redacted] ✅
BOT **Terraform** Server build and provisioning is complete with no errors! on https://ccserver.marques576.eu.org/ ✅
BOT **Terraform** Domain sucessfully updated on cloudflare with no errors! ✅

This code is written in Terraform, a tool for building, changing, and versioning infrastructure safely and efficiently. In this particular code, several resources are being declared using providers for different cloud services.

Each provider has a specific purpose for a specific part of the project and after each part de provisioned successfully a webhook is triggered to a discord a discord server. The cloud scenario was also provisioned with push triggers on Github so that if a commit is pushed to the main branch the front and backend are rebuilded.

Frontend(website)

The frontend(website) is hosted on **Vercel.** Vercel is a cloud computing company that provides a platform as a service (**PaaS**) for serverless deployment of web applications specially static websites.

I choose Vercel because:

- It is a tool that is ganning a lot of popularity recently.
- It simplifies a lot of the deployment process.
- The free plan is very generous.
- It has algorithms that optimize the SEO (search engine optimization)

Backend(socket server) and Database

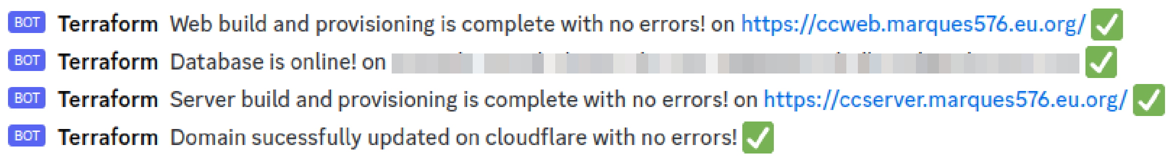
The backend(socket server) and database were deployed in digital ocean. DigitalOcean is a cloud computing company that provides infrastructure as a service (**IaaS**) for developers to deploy and scale applications. While the development of the provisioning I also tested google cloud that is the IaaS provided by Google but found it to be way more confusing and complex. I ended up by choosing digital ocean because like Vercel it simplifies a lot the process.

Cloudflare

Each time the backend is provisioned the digital ocean gives it a random url. And in order to use a custom domain I used the terraform provider to add the digital ocean domain as a CNAME in the cloudflare domain settings. By doing this the front-end can always acess the beackend due to it always having the same url. Due to domain propagation delay the website will not acess the backend by the backend custom domain, it will use the domain that digital ocean generates.

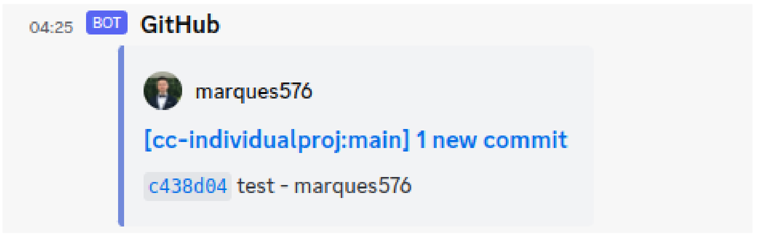
Notifications/Logs

Every time a service provisioned sucessfully with no errors a webhook is triggered so that it is possible to keep logs and at the same time know when services are ready.



BOT **Terraform** Web build and provisioning is complete with no errors! on https://ccweb.marques576.eu.org/ ✅
BOT **Terraform** Database is online! on [redacted] ✅
BOT **Terraform** Server build and provisioning is complete with no errors! on https://ccserver.marques576.eu.org/ ✅
BOT **Terraform** Domain sucessfully updated on cloudflare with no errors! ✅

Every time a commit is made to the main branch a the repo a webhook is also fired.



04:25 BOT **GitHub**
marques576
[cc-individualproj:main] 1 new commit
`c438d04` test - marques576

Discussion

The main difficulties while developing this project came with managing the concurrency in both scenarios due to having multiples services running in parallel.

Environment variables were also a challenges mainly because when a git push is made the projects are rebuilded and the environment variables need to be present as well.

Constantly changing a domains CNAME records means that the most recent record can take time to propagate.

The process of chossing a Iaas was also a challenge because in the first place I wanted to use google cloud because personally I am into the google eco system and wanted to broaden my knowledge in another one of their tools but due to many issues regarding permissions I ended up by choosing digital ocean because it is relatively easier to setup and the documentation of the digital ocean's terraform provider is betters than google cloud's.

Recomendations for other developers:

- Concurrency of the services.
- Try a variety of PaaS and IaaS and use the one that suits your needs.
- Do not double ctrl-c while doing "terraform apply" because it will loose the state of the provisioning and then it requires manual intervention to each service provider to restore the state.