

**Qwitter**

**Qwitter**

**DevOps Phase2**

**Prepared By**

**Fares Atef**

**Qwitter DevOps Engineer**

**13/11/2023**

Contents

[Introduction 2](#_Toc150783651)

[1) Docker Hub 3](#_Toc150783652)

[1) Write/modify my code 4](#_Toc150783653)

[2) Build the images on my local machine 4](#_Toc150783654)

[3) Push the images to Docker Hub 4](#_Toc150783655)

[4) Pull the images on the server 5](#_Toc150783656)

[5) Run the docker-compose.yml 5](#_Toc150783657)

[2) Free Domain Name 6](#_Toc150783658)

[3) Deploy Backend 7](#_Toc150783659)

[4) Deploy Frontend 8](#_Toc150783660)

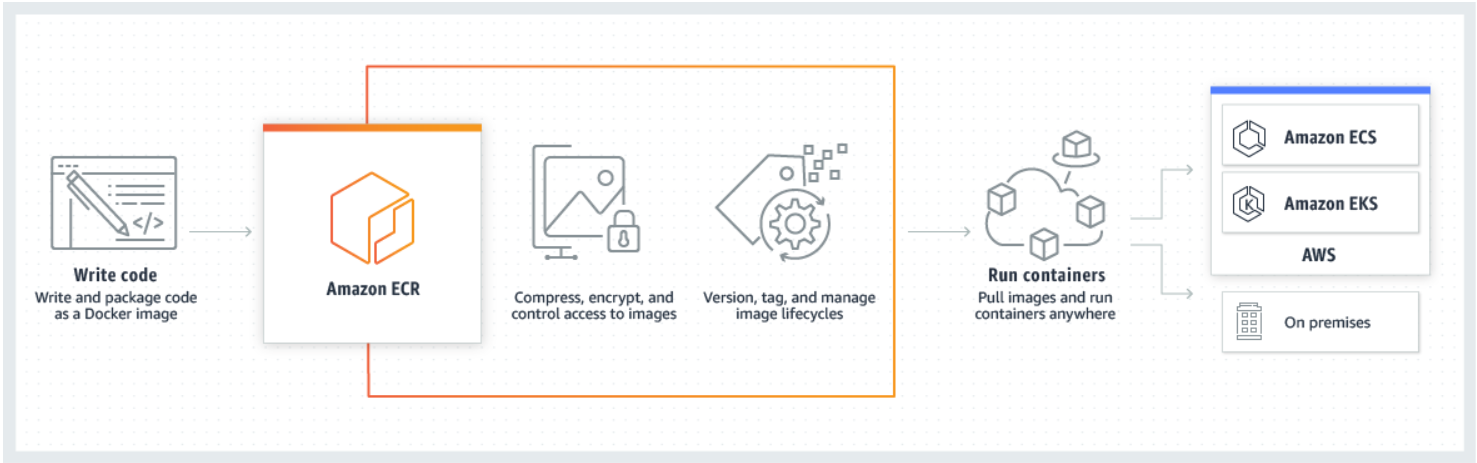
# Introduction

In this phase, I will learn how to use Docker Hub to push and pull my Docker images, rather than building them on the server to conserve its resources. Additionally, I will deploy Qwitter on AWS, using a domain name instead of the IP, and gain an introduction to Jenkins.

# 1) Docker Hub

In the previous phase, I made 4 projects and dockerized them, then I deployed them in AWS, but I made something which is not a good practice, I was building the docker images inside the server, this consumes the server resources, so instead we should build the images in another place and the server has to pull these ready images and run them.

This service can be achieved by different methods like Amazon ECR and Docker Hub.

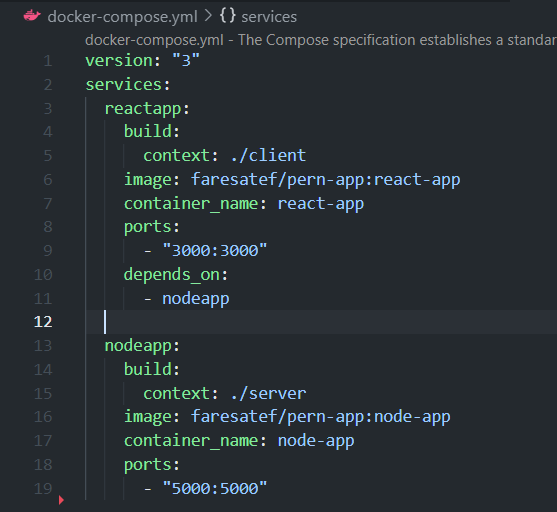


*Fig1: Amazon ECR*

I will use Docker Hub for this purpose, which involves the following steps:

1. Write/modify my code.
2. Build the images on my local machine.
3. Push the images to Docker Hub.
4. Pull the images on the server.
5. Run the docker-compose.yml.

Before proceeding with the aforementioned steps, I created a [Docker Hub repo](https://hub.docker.com/repository/docker/faresatef/pern-app/general) and then modified my docker-compose.yml to include the new built image source.



*Fig2: New docker-compose.yml*

Let’s break down each step:

# 1) Write/modify my code

In this step, I will write or modify my code.

# 2) Build the images on my local machine

After making changes to the code, I will build the new images using the following command:



# 3) Push the images to Docker Hub

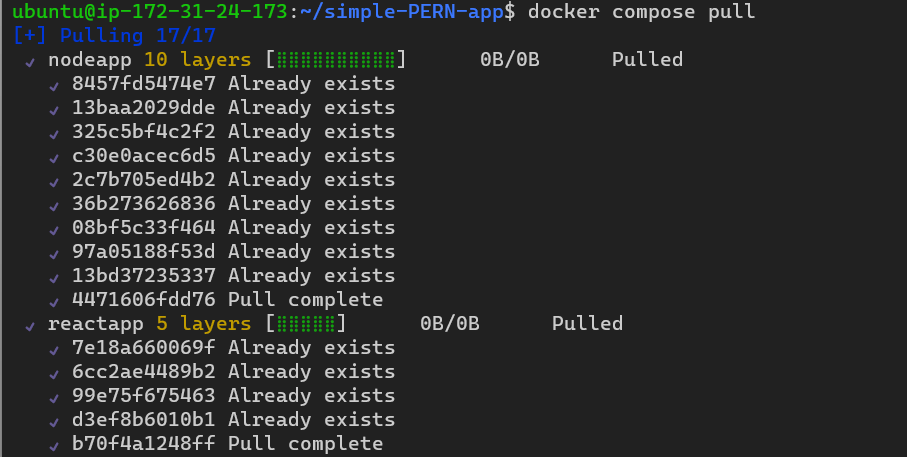
Once the images are ready, I'll push them to my Docker Hub repository using the following command:



# 4) Pull the images on the server

Inside the server, I will run the following command to pull the images:





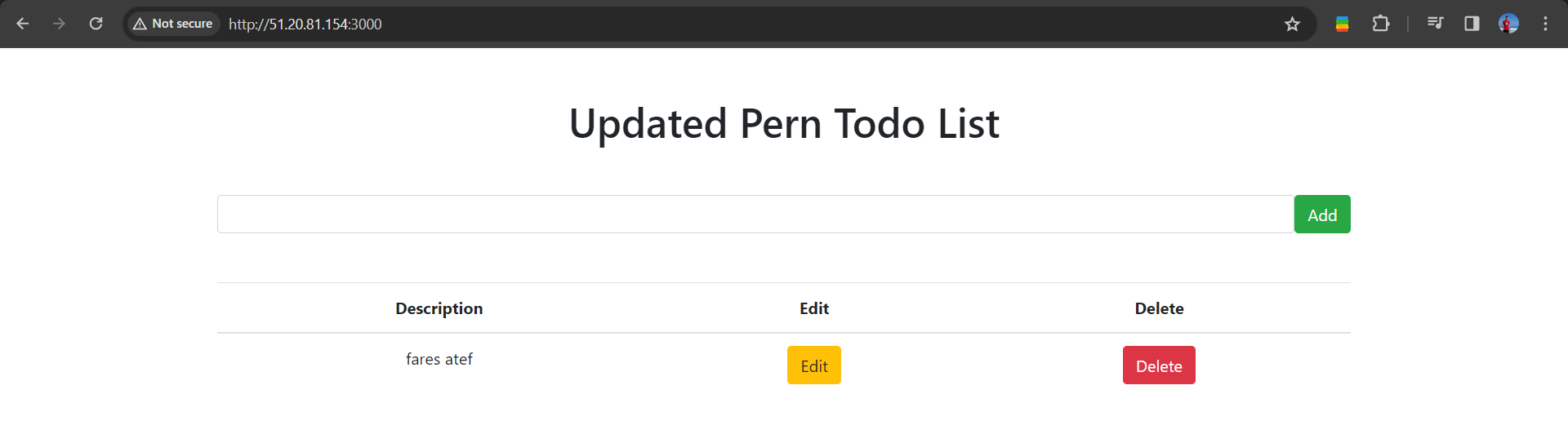
*Fig3: Pulling the images from Docker Hub*

# 5) Run the docker-compose.yml

Now everything is ready, let’s run the containers, this can be done with the following command:



Let’s try to access the new version of the app.

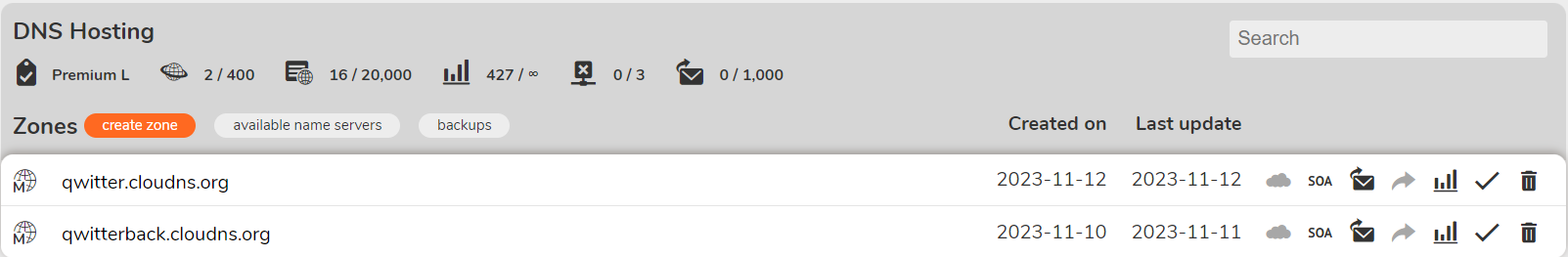


*Fig4: Updated PERN app*

# 2) Free Domain Name

Instead of accessing the website from its IP, we will access it through domain name, for this I will use ClouDNS.

I created a DNS zone for my domain name and added an A record to map the domain name to the IP address of my website.



*Fig5: New zone*

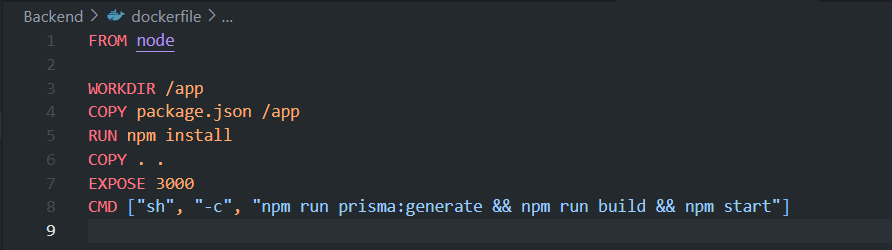


*Fig6: New Record*

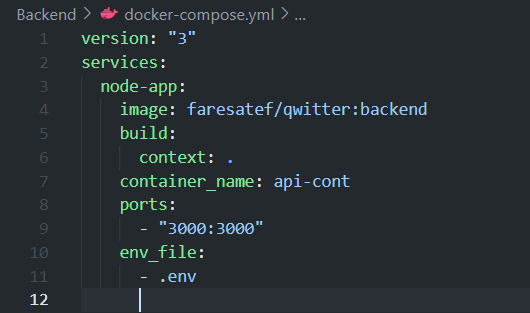
# 3) Deploy Backend

For the backend deployment, I created EC2 instance and RDS, then I did the following steps:

1. Clone the backend code from its repo on GitHub to my local machine.
2. Add the .env file with the endpoint of the RDS.
3. Build the code and then prisma:migrate.
4. Create dockerfile, docker-compose and .dockerignore.
5. Push to Docker Hub.
6. Pull from Docker Hub inside the server.
7. Touch the previous docker-compose and .env files inside the server.
8. Run docker-compose up.



*Fig7: Backend dockerfile*



*Fig8: Backend docker-compose*

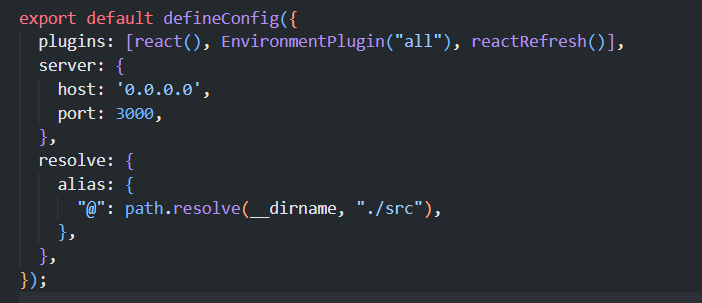
Now, the server is ready and can be accessed from here <http://qwitterback.cloudns.org:3000>.

# 4) Deploy Frontend

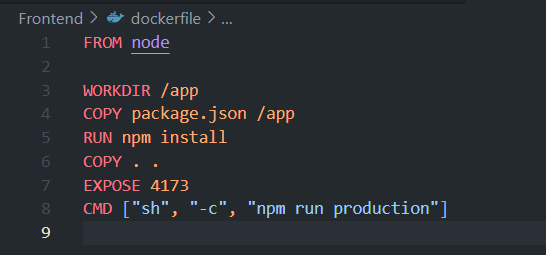
For the frontend deployment, I created EC2 instance, then I did the following steps:

1. Clone the frontend code from its repo on GitHub to my local machine.
2. Add the .env.local file.
3. Build the code.
4. Create dockerfile, docker-compose and .dockerignore.
5. Push to Docker Hub.
6. Pull from Docker Hub inside the server.
7. Touch the previous docker-compose and .env.local files inside the server.
8. Run docker-compose up.

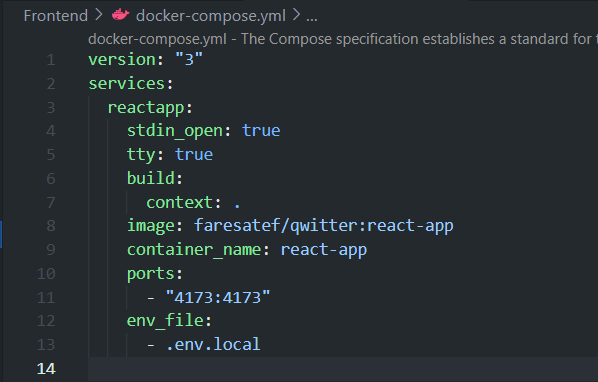
Also, I added this part to the vite configuration so we can expose the port inside the container.



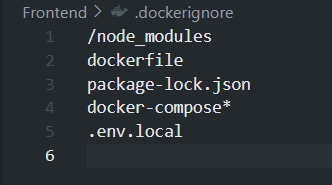
*Fig9: Vite Conf*



*Fig10: Frontend dockerfile*



*Fig11: Frontend dockerfile-compose*

**

*Fig12: Frontend & Backend .dockerignore*

Now, Qwitter can be accessed from here <http://qwitter.cloudns.org:4173>.