Deploying and Monitoring a Containerized Application Using AWS, Docker, and GitHub Actions

Racheal Kuranchie

1. Introduction

In this project, I deployed and monitored a containerized web application on AWS using Docker and EC2. To streamline deployments, I implemented a CI/CD pipeline with GitHub Actions and configured monitoring with Uptime Kuma. This project demonstrates my skills in DevOps practices, including containerization, CI/CD, and system monitoring.

2. Project Overview

Objective

- Deploy a web application using Docker on an AWS EC2 instance.
- Automate deployments using GitHub Actions.
- Monitor application health and performance with Uptime Kuma.

Tools Used

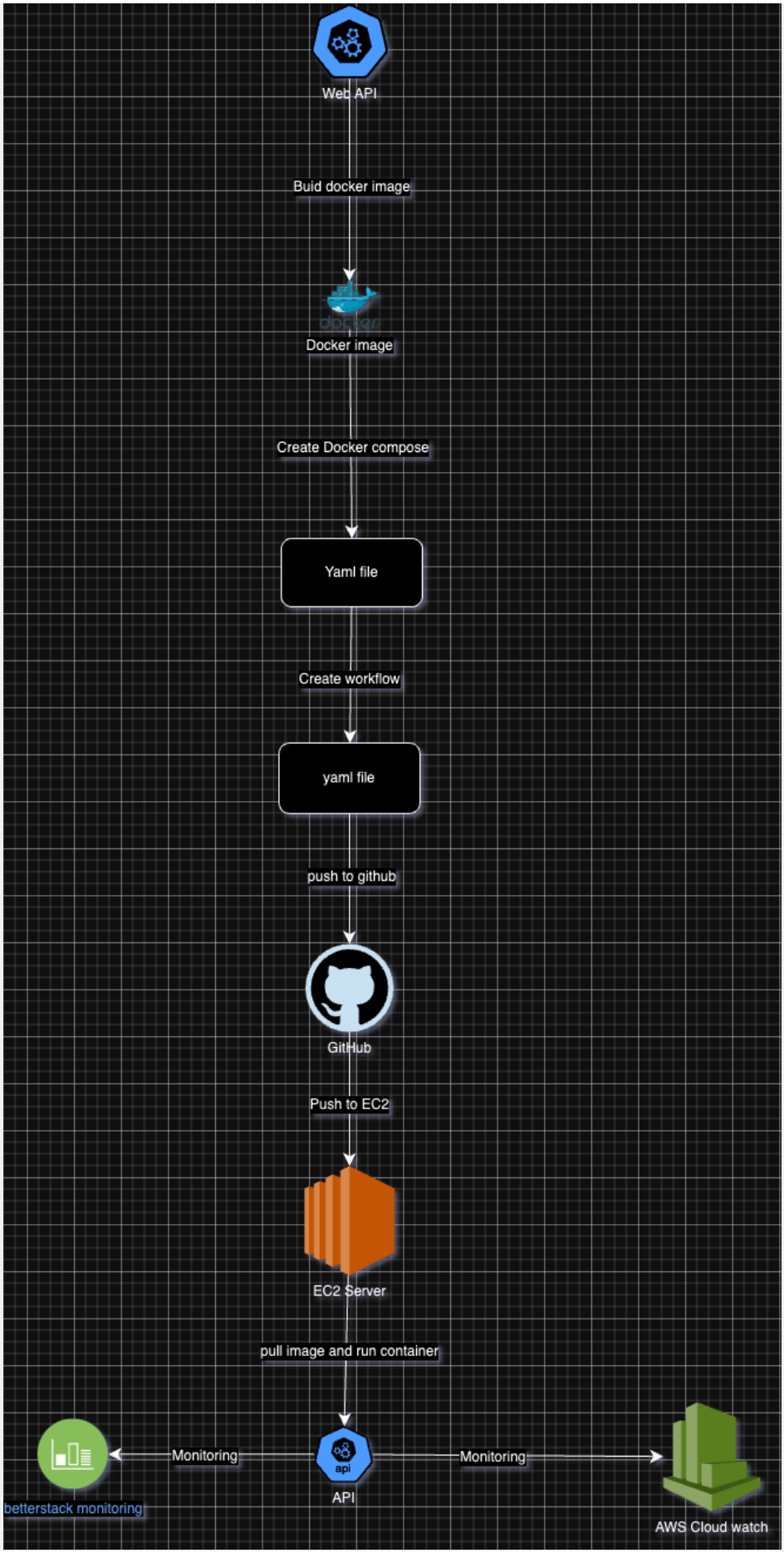
- AWS: EC2,
- Docker: Containerization
- GitHub Actions: CI/CD automation
- Uptime Kuma: Monitoring and alerting

3. Architecture Diagram

Overview

The architecture involves:

- A Dockerized web application hosted on an EC2 instance.
- GitHub Actions for automating builds and deployments.
- Uptime Kuma for monitoring application uptime and performance.


Web API
Buid docker image
Docker image
Create Docker compose
Yaml file
Create workflow
yaml file
push to github
GitHub
Push to EC2
EC2 Server
pull image and run container
Monitoring
Monitoring
api
API
betterstack monitoring
AWS Cloud watch

Architecture

Setting Up the Foundation

Provisioning EC2 Instance

1. Launch an EC2 instance from the AWS Management Console.
2. Install Docker and Docker Compose:
3. `sudo apt update`
4. `sudo apt install docker.io docker-compose`

Containerizing the Web Application

1. Create a simple Node.js application.
2. Write a Dockerfile:
3. Build and run the Docker image locally.

Deploying the Application

1. Push the Docker image to Docker Hub:
2. docker tag app:latest your-dockerhub-repo/app:latest
3. docker push your-dockerhub-repo/app:latest
4. Run the container on the EC2 instance:
5. docker run -d -p 80:3000 your-dockerhub-repo/app:latest

CI/CD Pipeline with GitHub Actions

Setup

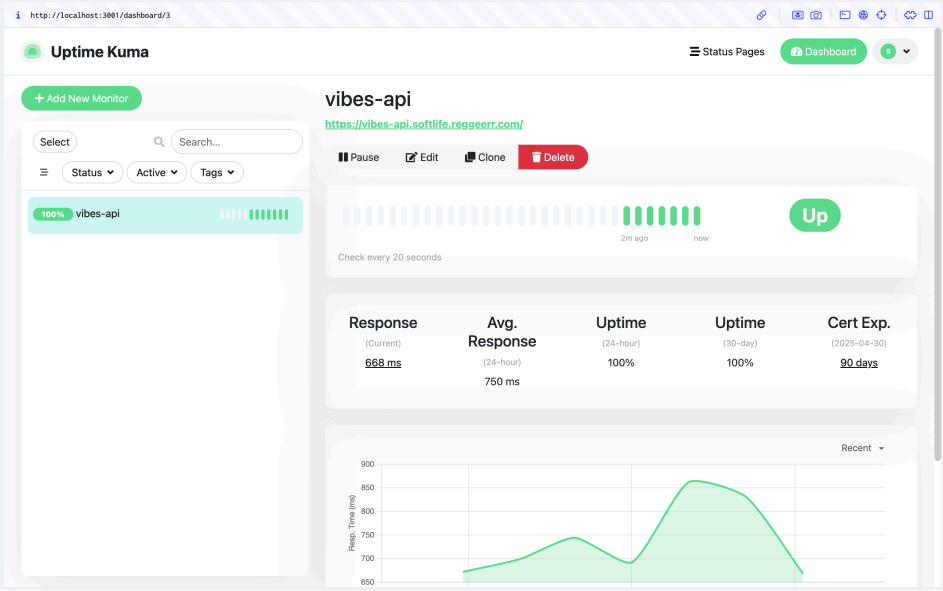
1. Add a GitHub Actions workflow file (.github/workflows/main.yml):

Monitoring and Alerting

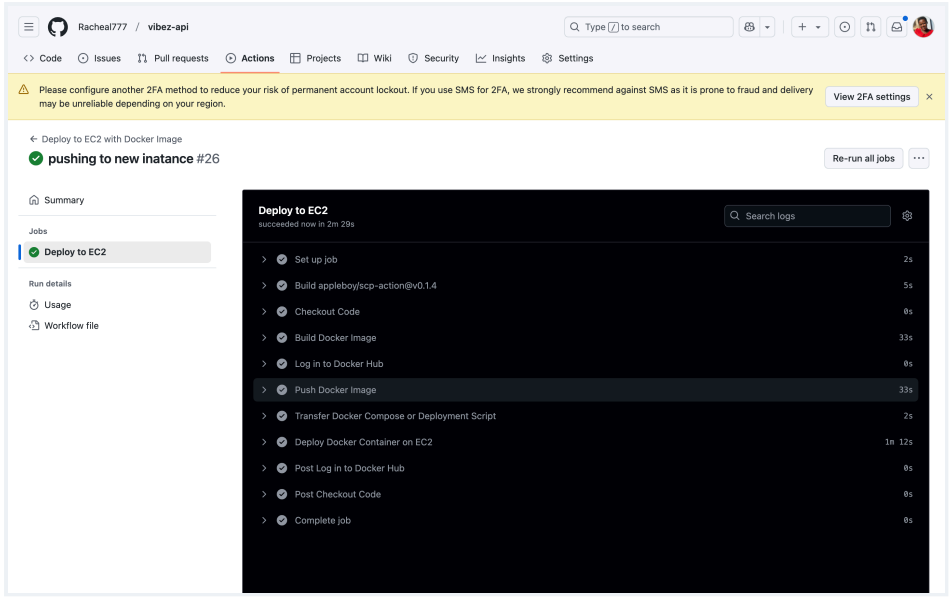
Uptime Kuma

1. Create a monitor in Uptime Kuma.
2. Add your application endpoint for monitoring.
3. Set up alerts for downtime or performance degradation.

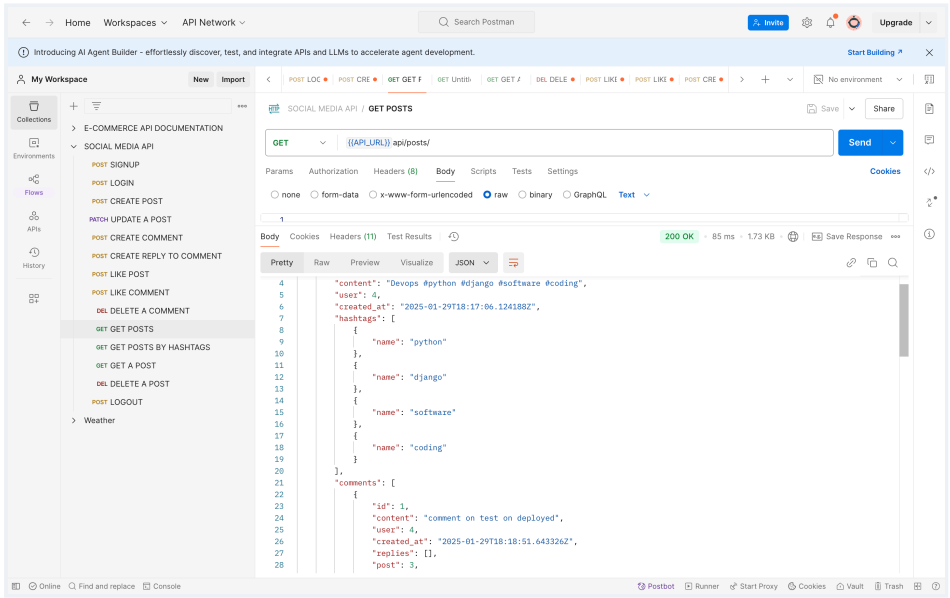
5. Results



Application**Application Monitoring on Uptime Kuma.**



GitHub Actions pipeline execution.



Testing with Postman

6. Challenges Faced and Lessons Learned

- Configuring security groups and networks in AWS EC2 was not easy.
- Learned how to configure Nginx and do SSL certificates.
- Learned how to optimize Docker images for smaller size and faster builds.
- Improved understanding of monitoring tools and alert configuration.

8. Repository and Links

- GitHub Repository: **https://github.com/Racheal777/vibez-api**
- Docker Hub Image: **rachealcodez/vibez:latest**
- References:
  - Docker Documentation
  - AWS EC2 Guide

9. Acknowledgments

Thanks to my tutors, peers, and online communities for guidance and support. Through my errors, the Twitter community has been a great help.