3-Tier Application Deployment with Docker-Compose and GitHub Actions

This repository contains a 3-tier web application that uses a React frontend, a Node.js backend, and a MongoDB database. The application is containerized using Docker-Compose and automatically deployed to an AWS EC2 instance via GitHub Actions.

Github Repo : - https:/github.com/nkdkd/Containerized-mern-app.git

Prerequisites

Before setting up the deployment, make sure the following prerequisites are met:

EC2 Instance Setup

1. **AWS Account**: You need an active AWS account to create an EC2 instance.
2. **EC2 Instance**: Ensure you have an Ubuntu EC2 instance running. Make sure the instance has:
    - Ports 22 (SSH), 80 (HTTP), 443 (HTTPS) , and 5000open in the security group.
    - Docker and Docker Compose installed on the EC2 instance.
    - Public and private key pairs generated for SSH access.
3. **Elastic IP**: Associate an elastic IP to the instance to keep the IP static.(optional)
4. **Environment Variables**: Ensure sensitive information such as database passwords, API keys, etc., are handled using environment variables.
5. **SSH Access**: Ensure that your SSH key pair is set up for the EC2 instance and has been added to your GitHub secrets as `SSH_PRIVATE_KEY`.

Dependencies on EC2

Install the following on your EC2 instance:

```
bash
# Update the package manager
sudo apt-get update

# Install Docker
sudo apt-get install docker.io

# Install Docker Compose
sudo apt-get install docker-compose

# Add the current user to the docker group (optional)
sudo usermod -aG docker $USER
```

```
# give this permission
sudo chmod 777 /var/run/docker.sock
```

Make sure to restart your terminal or run `newgrp docker` for this to take effect.

Project Setup

Step 1: Docker-Compose File

The `docker-compose.yml` file orchestrates the 3-tier application. It defines services for the React frontend, Node.js backend, and MongoDB database.

```
version: '3'

services:
  backend:
    build:
      context: ./Backend
    ports:
      - "5000:5000"
    environment:
      - MONGO_URI=${MONGO_URI}
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_INITDB_ROOT_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_INITDB_ROOT_PASSWORD}
    depends_on:
      - mongo

  frontend:
    build:
      context: ./Frontend
    ports:
      - "80:80"

  mongo:
    image: mongo:latest
    environment:
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_INITDB_ROOT_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_INITDB_ROOT_PASSWORD}
    volumes:
      - mongo_data:/data/db
    ports:
      - "27017:27017"

volumes:
  mongo_data:
```

Important Notes:

1. **Environment Variables**:
    - Ensure that the `.env` file contains the required environment variables such as `MONGO_URI`, `MONGO_INITDB_ROOT_USERNAME`, and `MONGO_INITDB_ROOT_PASSWORD`.

Step 2: GitHub Actions Workflow

```
name: Deploy to EC2

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up SSH agent
      uses: webfactory/ssh-agent@v0.5.3
      with:
        ssh-private-key: ${{ secrets.SSH_PRIVATE_KEY }}

    - name: Deploy to EC2
      run: |
        ssh -o StrictHostKeyChecking=no ${{ secrets.EC2_HOST }} << 'EOF'
        cd /home/ubuntu/my-app
        cd Containerized-mern-app/Dockerized-mern-app/
        git pull origin main
        docker-compose down
        docker-compose up -d --build
        EOF
```

Step 3: Steps to configure:

1. **GitHub Secrets**:
   - Go to your GitHub repository > Settings > Secrets > Actions.
   - Add the following secrets:
     - `SSH_PRIVATE_KEY`: Your SSH private key to authenticate the connection.
     - `EC2_HOST`: The EC2 instance address, e.g., `ubuntu@ec2-XX-XX-XX-XX.compute-1.amazonaws.com`.
2. **EC2 Setup**:
   - Ensure your EC2 instance has Docker and Docker Compose installed.
   - Make sure the necessary ports (80, 5000, 27017, etc.) are open for communication.
   - Ensure your repository is cloned on the EC2 instance in `/home/ubuntu/my-app/Containerized-mern-app/Dockerized-mern-app/`.

The `deploy.yml` file automates the deployment of the Docker-Compose file to your EC2 instance using GitHub Actions. This setup uses SSH for remote deployment.

To add GitHub secrets:

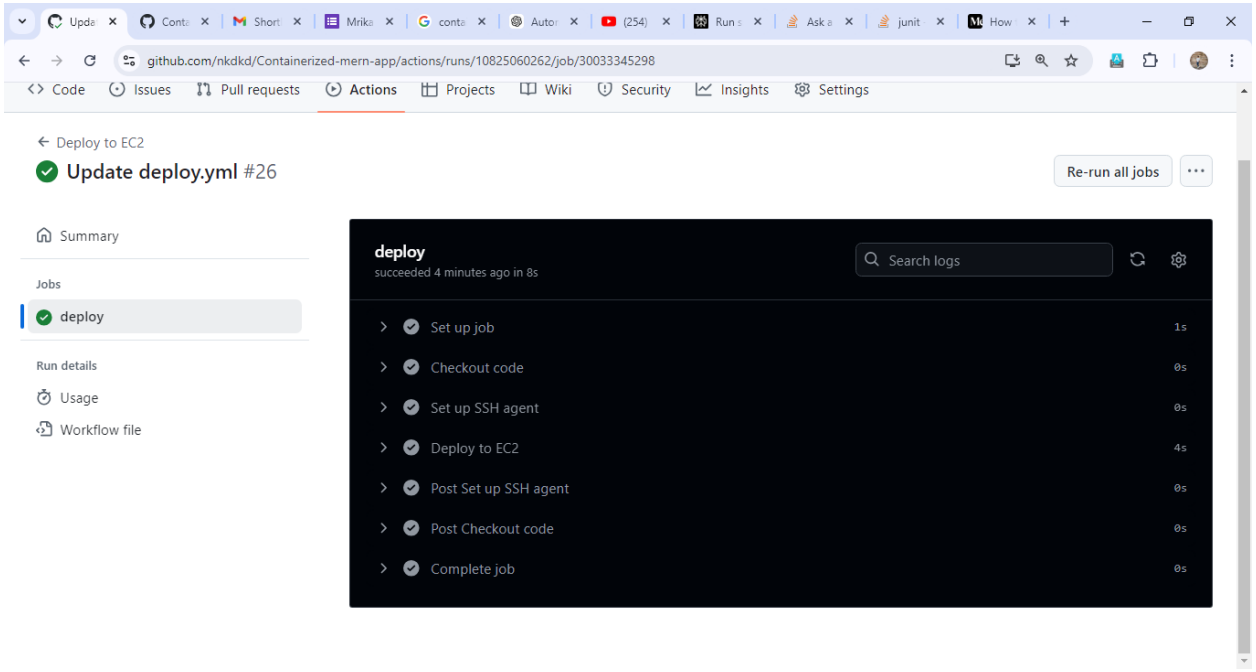
- Go to your repository's "Settings" > "Secrets and Variables" > "Actions."
- Click "New repository secret" and add your secrets.

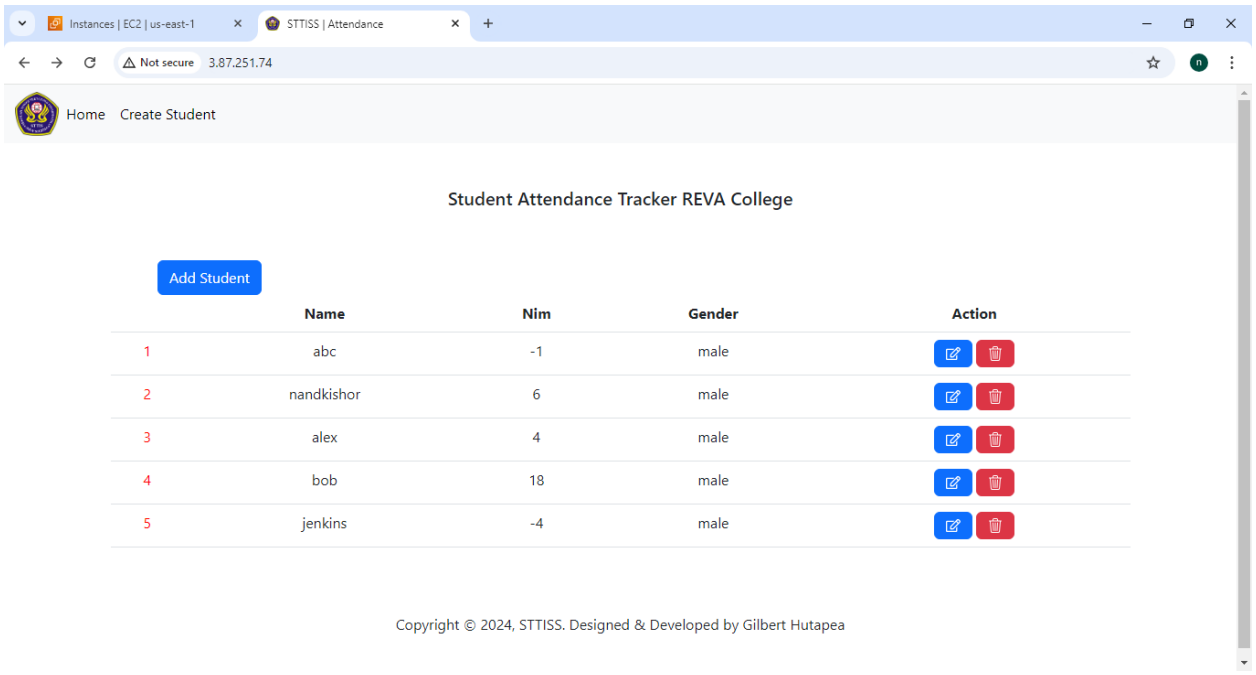
Step 4: Deployment

Once everything is set up, every time you push code to the `main` branch, the GitHub Actions workflow will automatically:

- Install Docker and Docker Compose on the EC2 instance if not installed.
- Pull the latest code from the GitHub repository on the EC2 instance.
- Rebuild and restart the Docker containers using Docker Compose.

Screenshots




Instances | EC2 | us-east-1
STTISS | Attendance
Not secure
3.87.251.74
Home
Create Student
Student Attendance Tracker REVA College
Add Student
Name
Nim
Gender
Action
1 abc -1 male
2 nandkishor 6 male
3 alex 4 male
4 bob 18 male
5 jenkins -4 male
Copyright © 2024, STTISS. Designed & Developed by Gilbert Hutapea