Integrating Jenkins CI/CD with Docker Compose Deployment

1. Prerequisites

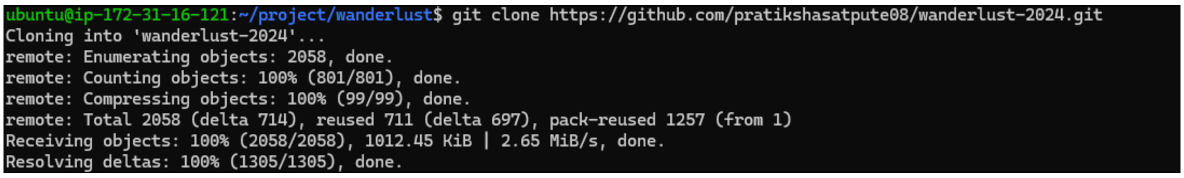
Ensure you have the following installed:

- **Jenkins:** A running Jenkins server.
- **Git:** For source code management.
- **Docker & Docker Compose:** To handle containerized applications.
- **Jenkins Plugins:** Ensure you have the following plugins installed:
  - Docker Pipeline
  - Git Plugin
  - Pipeline Plugin

2. Clone the Repository

Begin by cloning the repository that contains the Docker Compose project. Open your terminal and run:

```
git clone <repository_url>
```

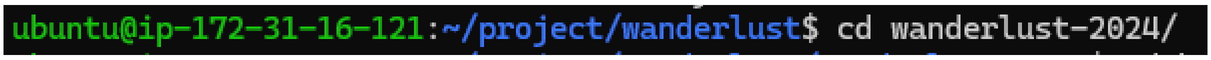


```
ubuntu@ip-172-31-16-121:~/project/wanderlust$ git clone https://github.com/pratikshasatpute08/wanderlust-2024.git
Cloning into 'wanderlust-2024'...
remote: Enumerating objects: 2058, done.
remote: Counting objects: 100% (801/801), done.
remote: Compressing objects: 100% (99/99), done.
remote: Total 2058 (delta 714), reused 711 (delta 697), pack-reused 1257 (from 1)
Receiving objects: 100% (2058/2058), 1012.45 KiB | 2.65 MiB/s, done.
Resolving deltas: 100% (1305/1305), done.
```

3. Navigate to the Project Directory

After cloning the repository, navigate to the project's root directory:

```
cd  <project_directory>
```



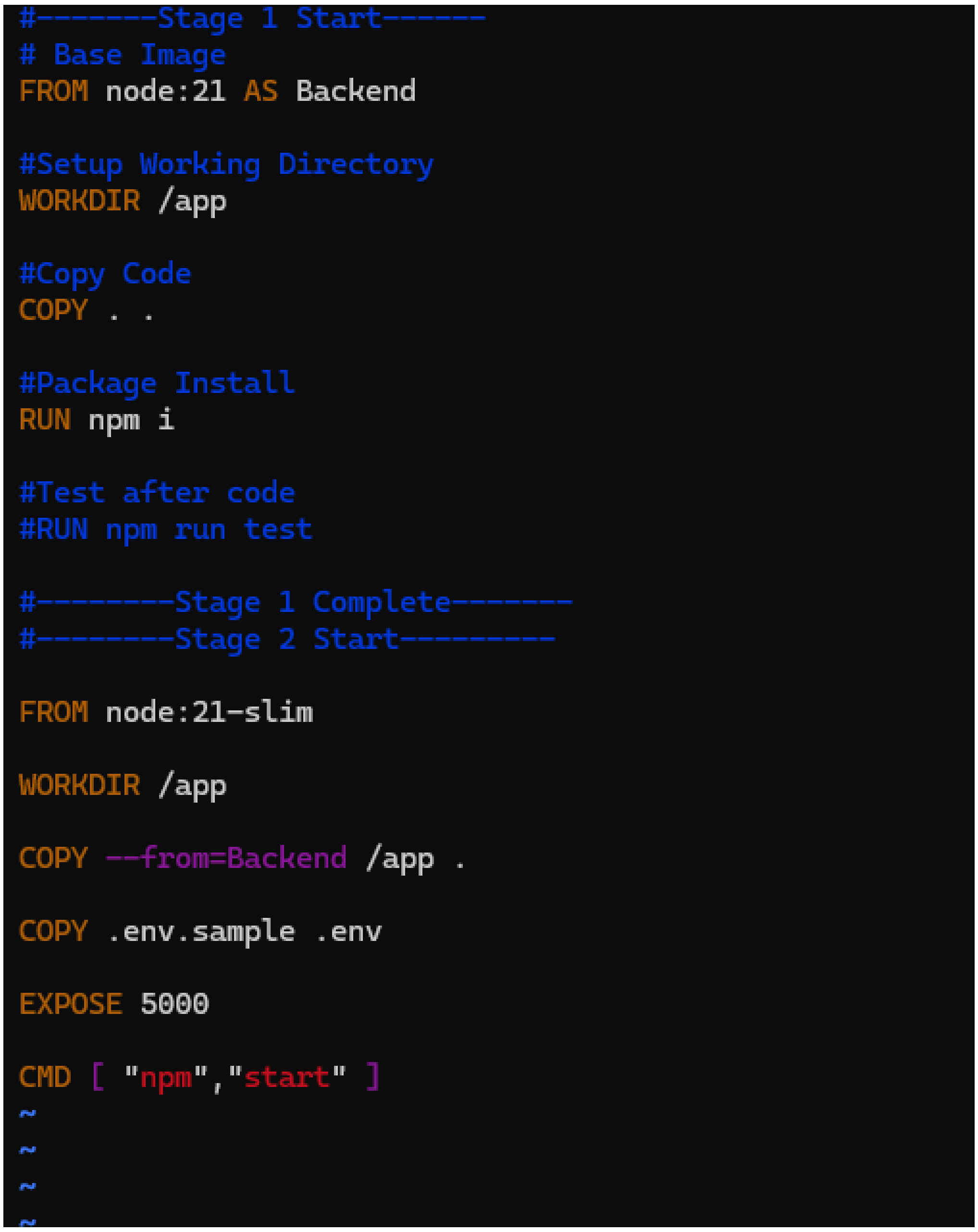
```
ubuntu@ip-172-31-16-121:~/project/wanderlust$ cd wanderlust-2024/
```

4. crate docker file

Dockerfiles for both the backend and frontend components, as well as a docker-compose.yml file to orchestrate the deployment.

Create the Backend Dockerfile

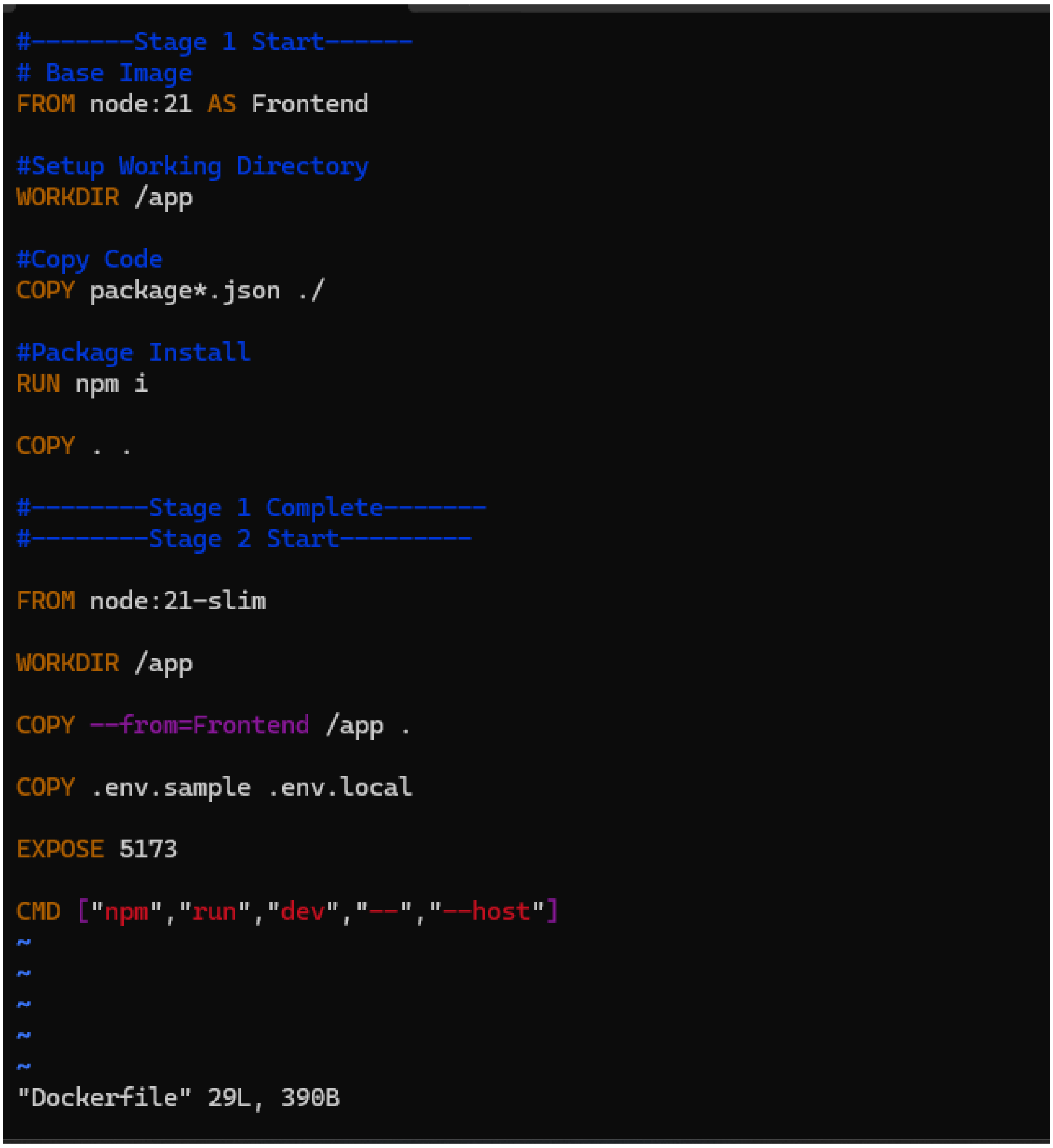
In your project directory, create a Dockerfile for the backend service. This file should be located in the backend directory (e.g., backend/Dockerfile).



```
#--------Stage 1 Start------
# Base Image
FROM node:21 AS Backend

#Setup Working Directory
WORKDIR /app

#Copy Code
COPY . .

#Package Install
RUN npm i

#Test after code
#RUN npm run test

#---------Stage 1 Complete-------
#---------Stage 2 Start---------

FROM node:21-slim

WORKDIR /app

COPY --from=Backend /app .

COPY .env.sample .env

EXPOSE 5000

CMD [ "npm","start" ]
~
~
~
~
```

Create the Frontend Dockerfile

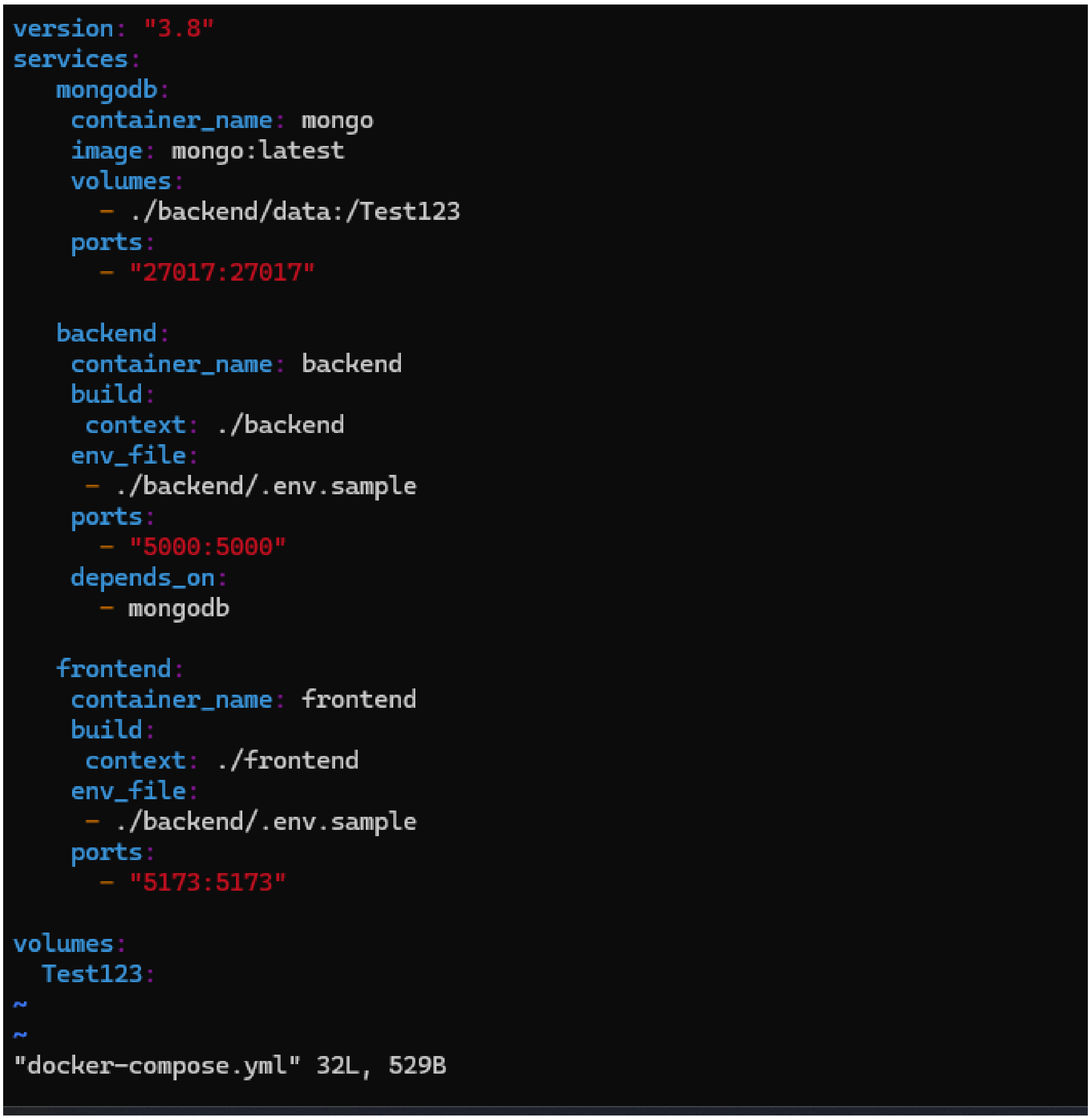
Next, create a Dockerfile for the frontend service, typically located in the frontend directory (e.g., frontend/Dockerfile).



```
#--------Stage 1 Start-------
# Base Image
FROM node:21 AS Frontend

#Setup Working Directory
WORKDIR /app

#Copy Code
COPY package*.json ./

#Package Install
RUN npm i

COPY . .

#---------Stage 1 Complete--------
#---------Stage 2 Start----------

FROM node:21-slim

WORKDIR /app

COPY --from=Frontend /app .

COPY .env.sample .env.local

EXPOSE 5173

CMD ["npm","run","dev","--","--host"]
~
~
~
~
~
"Dockerfile" 29L, 390B
```

5. Create the Docker Compose File

The docker-compose.yml file is crucial as it defines how the backend and frontend services will interact. Place this file at the root of your project directory.



```
version: "3.8"
services:
   mongodb:
    container_name: mongo
    image: mongo:latest
    volumes:
      - ./backend/data:/Test123
    ports:
      - "27017:27017"

   backend:
    container_name: backend
    build:
     context: ./backend
    env_file:
     - ./backend/.env.sample
    ports:
      - "5000:5000"
    depends_on:
      - mongodb

   frontend:
    container_name: frontend
    build:
     context: ./frontend
    env_file:
     - ./backend/.env.sample
    ports:
      - "5173:5173"

volumes:
  Test123:
~
~
"docker-compose.yml" 32L, 529B
```

Explanation of docker-compose.yml:

version: Specifies the version of the Docker Compose file format. Version 3.8 is one of the latest, offering a balance between features and compatibility.

Services

The services section defines each container that Docker Compose will manage. In this case, you have three services: mongodb, backend, and frontend.

▯ **container_name:** Specifies the name of the container, which will be mongo.

▯ **image:** Specifies the Docker image to use for this service. mongo:latest pulls the latest version of MongoDB from Docker Hub.

▯ **volumes:**

- Maps a directory on the host (./backend/data) to a directory inside the container (/Test123).
- This is important for persisting MongoDB data. Even if the container is stopped or removed, the data remains intact on the host machine.
- Test123 is the name of the volume inside the container, allowing MongoDB to store its data there.

ports:

- Maps port 27017 on the container to port 27017 on the host.
- Port 27017 is the default port MongoDB listens on.

▯ **container_name:** The name of the container will be backend.

▯ **build:**

- **context:** Specifies the build context, which is the ./backend directory. Docker will look for a Dockerfile in this directory to build the backend image.

▯ **env_file:**

- Points to an environment file (.env.sample) located in the backend directory.
- This file contains environment variables that the backend service will use, such as database credentials, API keys, or configuration settings.

▯ **ports:**

- Maps port 5000 on the container to port 5000 on the host.
- The backend service will be accessible via http://localhost:5000.

depends_on:

- Specifies that the backend service depends on the mongodb service.
- Docker Compose will ensure that the MongoDB service is started before the backend service.

▯ **container_name:** The name of the container will be frontend.

� **build:**

- **context:** Specifies the build context as the ./frontend directory. Docker will look for a Dockerfile in this directory to build the frontend image.

� **env_file:**

- Points to the same environment file (.env.sample) as the backend.
- This is likely used to share some configuration, such as API endpoints, between the backend and frontend.

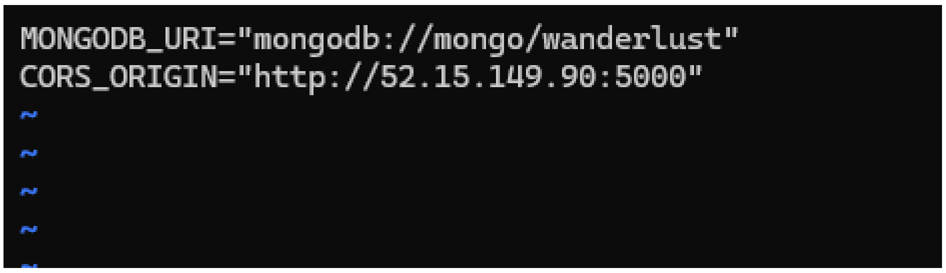
� **ports:**

- Maps port 5173 on the container to port 5173 on the host.
- The frontend service will be accessible via http://localhost:5173.

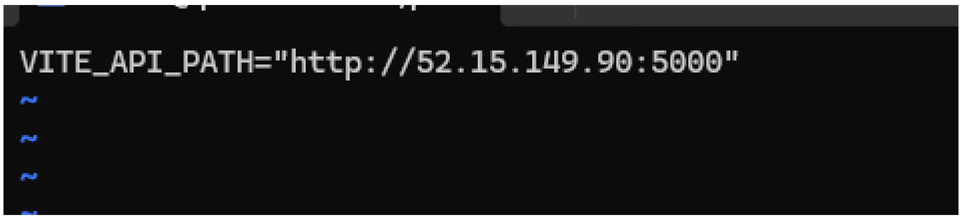
volumes:

- Defines a named volume Test123, which is being mapped to ./backend/data in the MongoDB service.
- **Named volumes** like Test123 ensure that data is persisted even when containers are stopped or removed.
- The named volume Test123 is mounted inside the MongoDB container at /Test123, enabling MongoDB to store its database files in this location.

6. Add you ec2 instance port on both frontend and backend .env.sample file



```
MONGODB_URI="mongodb://mongo/wanderlust"
CORS_ORIGIN="http://52.15.149.90:5000"
~
~
~
~
```



```
VITE_API_PATH="http://52.15.149.90:5000"
~
~
~
```

7. Create a Jenkins Pipeline Job

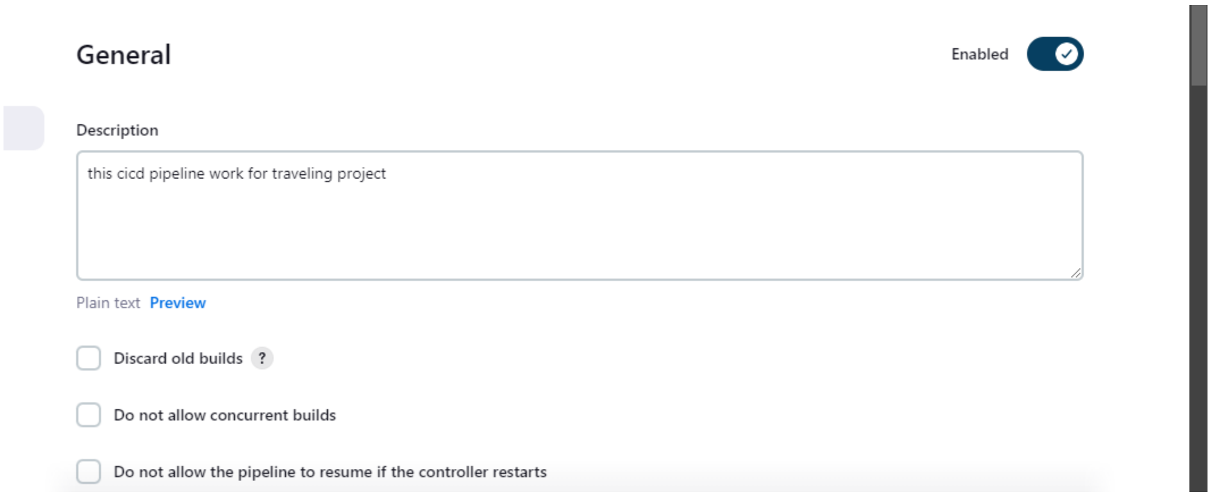
 Log in to Jenkins:

- **Open your Jenkins dashboard.**

 Create a New Pipeline:

- **Click on New Item in the Jenkins dashboard.**
- **Enter a name for your pipeline.**
- **Select Pipeline and click OK.**

8.Configure the Pipeline


General

Enabled ✓

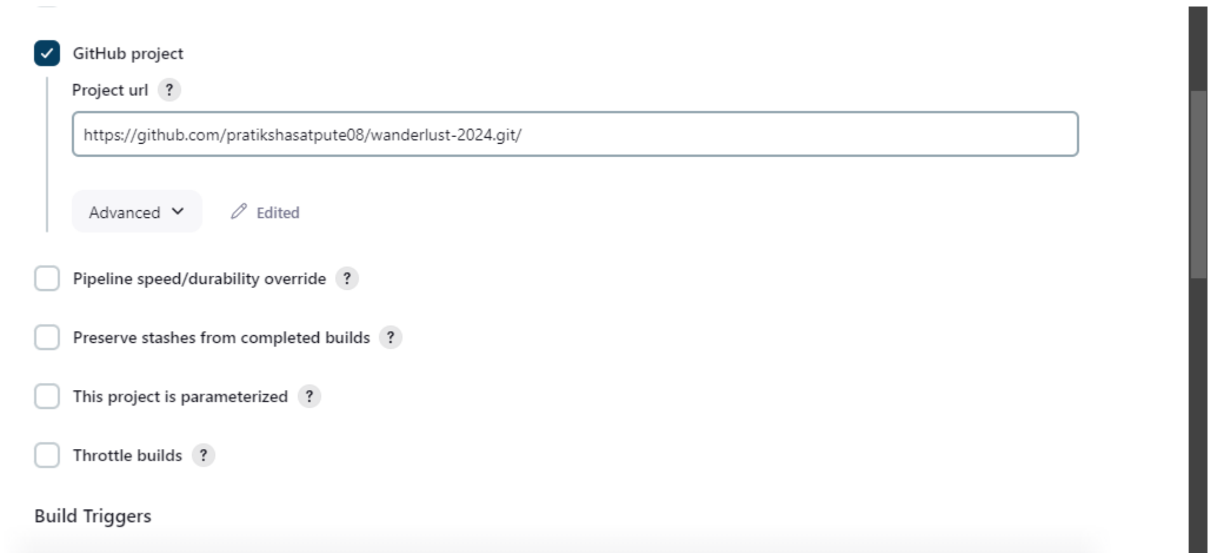
Description

this cicd pipeline work for traveling project

Plain text Preview

- [ ] Discard old builds ?
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts

9.Add GitHub repo URL(Enter the repository URL in the Repository URL field.) :


- [x] GitHub project

Project url ?

https://github.com/pratikshasatpute08/wanderlust-2024.git/

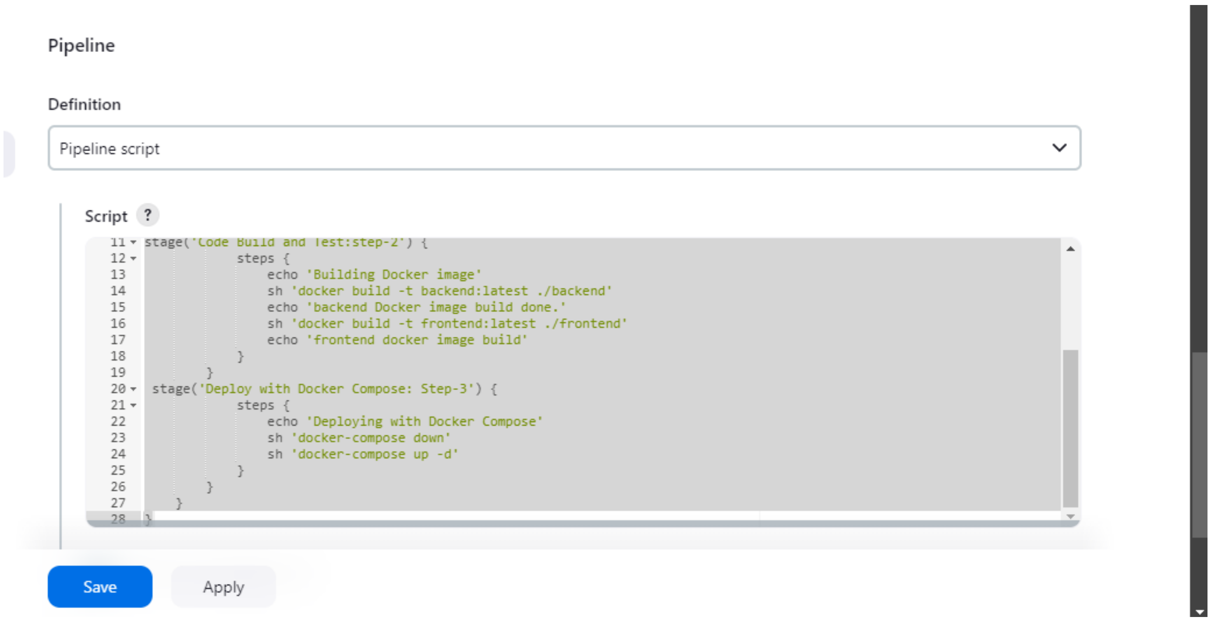
Advanced Edited

- [ ] Pipeline speed/durability override ?
- [ ] Preserve stashes from completed builds ?
- [ ] This project is parameterized ?
- [ ] Throttle builds ?

Build Triggers

10.Define the Pipeline Script

In your project repository, create a Jenkinsfile at the root level. This file will define the CI/CD pipeline.



```
stage('Code Build and Test:step-2') {
            steps {
                echo 'Building Docker image'
                sh 'docker build -t backend:latest ./backend'
                echo 'backend Docker image build done.'
                sh 'docker build -t frontend:latest ./frontend'
                echo 'frontend docker image build'
            }
        }
 stage('Deploy with Docker Compose: Step-3') {
            steps {
                echo 'Deploying with Docker Compose'
                sh 'docker-compose down'
                sh 'docker-compose up -d'
            }
        }
    }
}
```

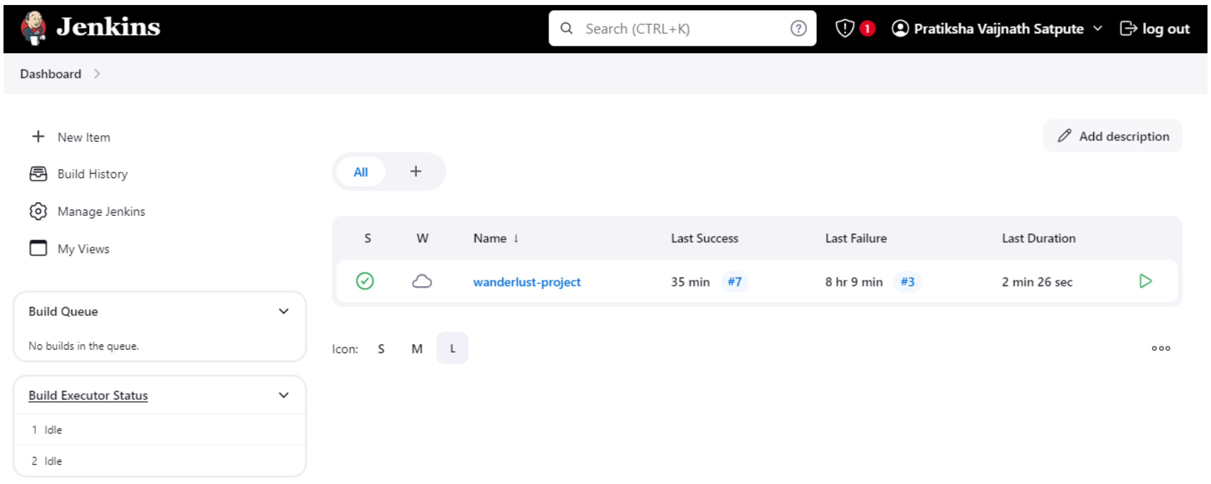
```
pipeline {
   agent any

   stages {
      stage('Code Clone from GitHub:step-1') {
         steps {
            echo 'Cloning code from GitHub'
            git url: 'https://github.com/pratikshasatpute08/wanderlust-2024.git', branch: 'main'
         }
      }
stage('Code Build and Test:step-2') {
         steps {
            echo 'Building Docker image'
            sh 'docker build -t backend:latest ./backend'
            echo 'backend Docker image build done.'
```

```
                sh 'docker build -t frontend:latest ./frontend'
                echo 'frontend docker image build'
            }
        }
    stage('Deploy with Docker Compose: Step-3') {
            steps {
                echo 'Deploying with Docker Compose'
                sh 'docker-compose down'
                sh 'docker-compose up -d'
            }
        }
    }
}
```

11.Run the Jenkins Pipeline

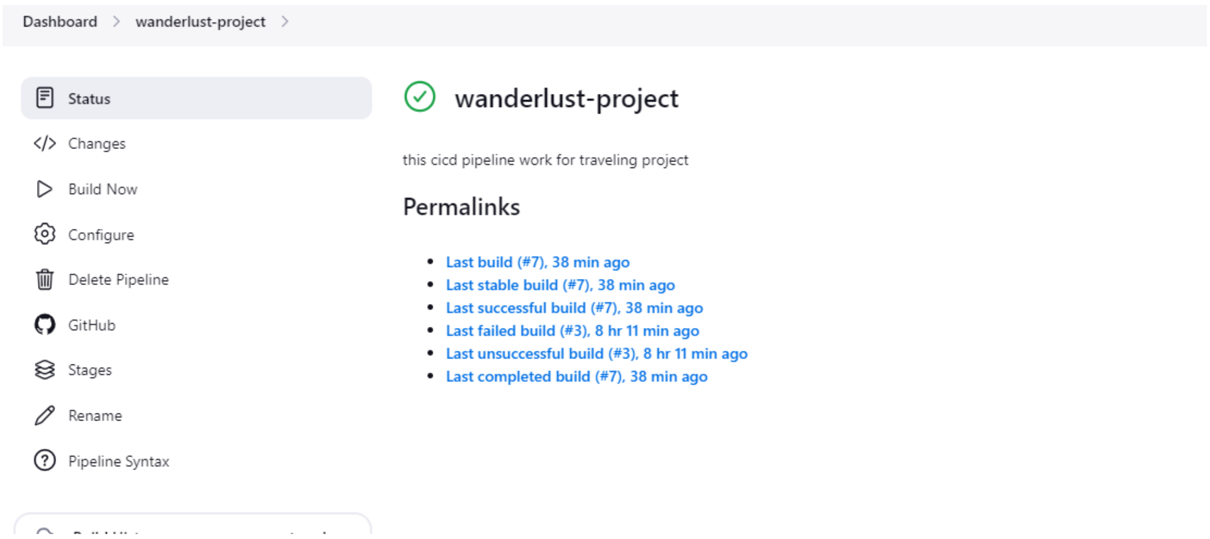
1. **Save the Pipeline Configuration:**
   - Click Save after configuring the pipeline.

Build the Pipeline:



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☁	wanderlust-project	35 min #7	8 hr 9 min #3	2 min 26 sec

- **Trigger the pipeline by clicking on Build Now.**


Dashboard > wanderlust-project >

Status
Changes
Build Now
Configure
Delete Pipeline
GitHub
Stages
Rename
Pipeline Syntax

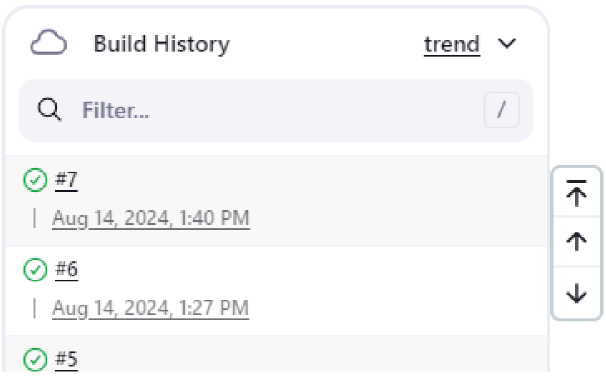
wanderlust-project

this cicd pipeline work for traveling project

Permalinks

- Last build (#7), 38 min ago
- Last stable build (#7), 38 min ago
- Last successful build (#7), 38 min ago
- Last failed build (#3), 8 hr 11 min ago
- Last unsuccessful build (#3), 8 hr 11 min ago
- Last completed build (#7), 38 min ago

-


Build History trend

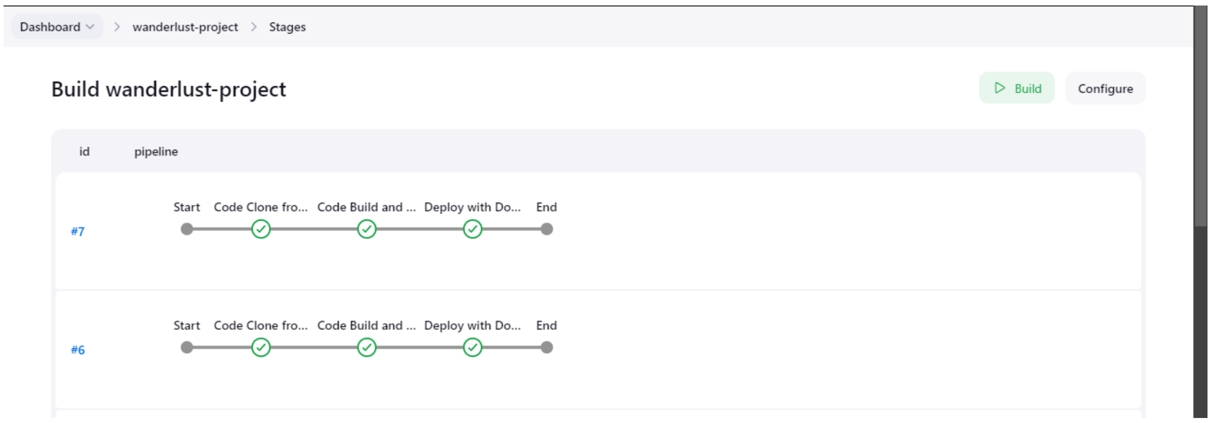
Filter...

#7
| Aug 14, 2024, 1:40 PM

#6
| Aug 14, 2024, 1:27 PM

#5

12.Check the stages of pipeline


Dashboard > wanderlust-project > Stages

Build wanderlust-project

Build Configure

id	pipeline
#7	Start — Code Clone fro... ✓ — Code Build and ... ✓ — Deploy with Do... ✓ — End
#6	Start — Code Clone fro... ✓ — Code Build and ... ✓ — Deploy with Do... ✓ — End

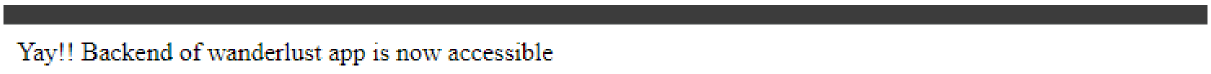
13.Access the Application via Browser

After deploying the application on your EC2 instance, the final step is to verify that everything is working correctly by accessing the frontend and backend via a web browser.

Accessing the Backend

Open your web browser and enter the following URL:

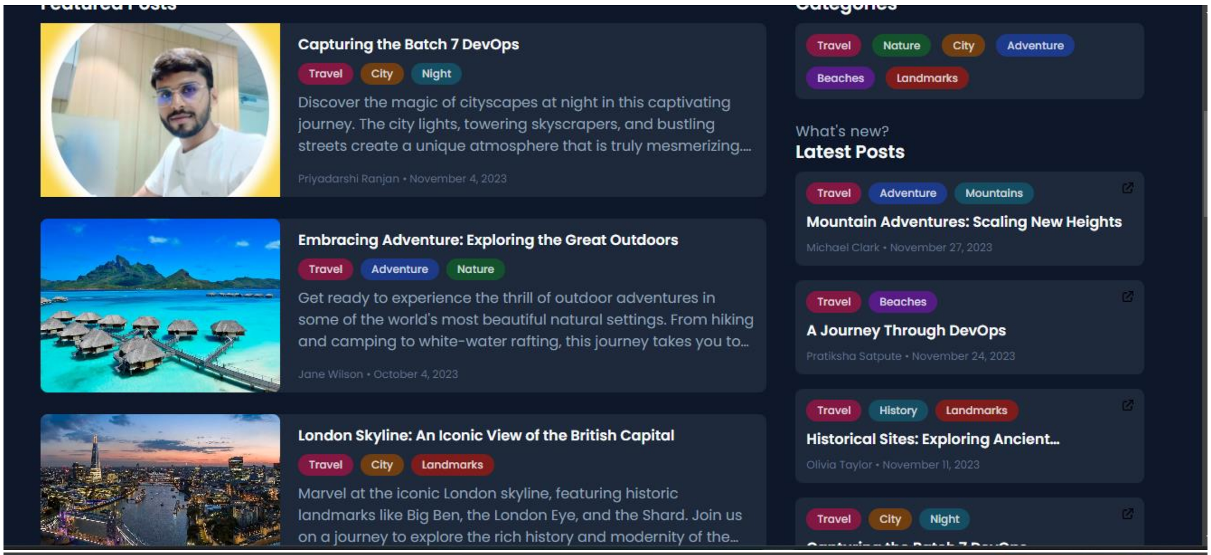
- http://YOUR_EC2_PUBLIC_IP:5000
- This URL allows you to access the backend API directly, which can be useful for testing or debugging.



```
Yay!! Backend of wanderlust app is now accessible
```

Accessing the Frontend

1. **Open your web browser** and enter the following URL:
    - http://YOUR_EC2_PUBLIC_IP:5173
    - Replace YOUR_EC2_PUBLIC_IP with the actual public IP address or DNS name of your EC2 instance.


Capturing the Batch 7 DevOps

Travel City Night

Discover the magic of cityscapes at night in this captivating journey. The city lights, towering skyscrapers, and bustling streets create a unique atmosphere that is truly mesmerizing....

Priyadarshi Ranjan • November 4, 2023

Embracing Adventure: Exploring the Great Outdoors

Travel Adventure Nature

Get ready to experience the thrill of outdoor adventures in some of the world's most beautiful natural settings. From hiking and camping to white-water rafting, this journey takes you to...

Jane Wilson • October 4, 2023

London Skyline: An Iconic View of the British Capital

Travel City Landmarks

Marvel at the iconic London skyline, featuring historic landmarks like Big Ben, the London Eye, and the Shard. Join us on a journey to explore the rich history and modernity of the...

Travel Nature City Adventure Beaches Landmarks

What's new?

Latest Posts

Travel Adventure Mountains

Mountain Adventures: Scaling New Heights

Michael Clark • November 27, 2023

Travel Beaches

A Journey Through DevOps

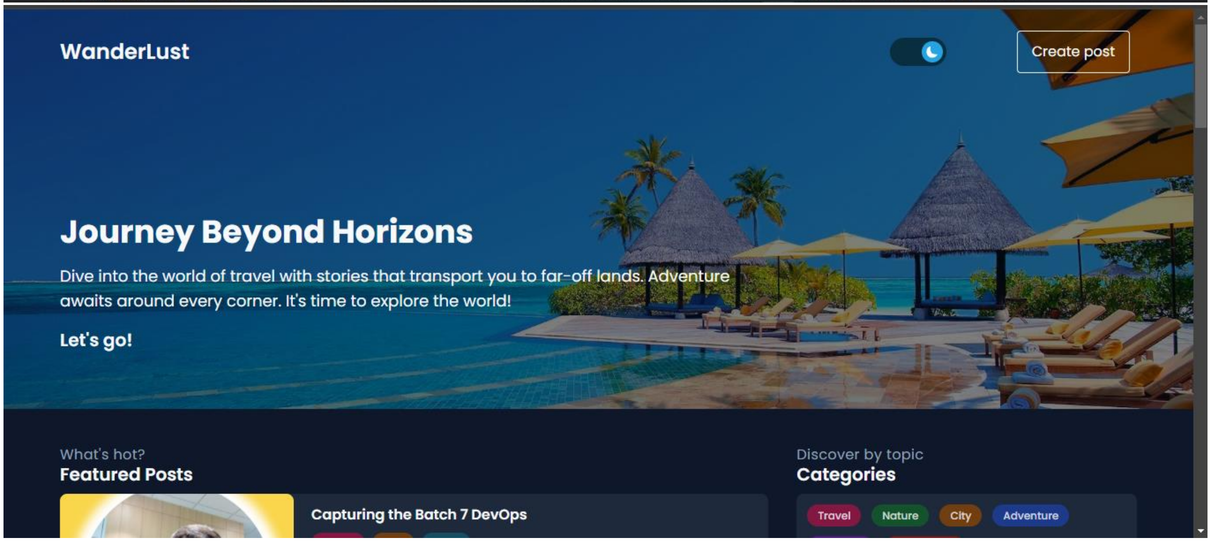
Pratiksha Satpute • November 24, 2023

Travel History Landmarks

Historical Sites: Exploring Ancient...

Olivia Taylor • November 11, 2023

Travel City Night



Journey Beyond Horizons

Dive into the world of travel with stories that transport you to far-off lands. Adventure awaits around every corner. It's time to explore the world!

Let's go!

What's hot?

Featured Posts

Capturing the Batch 7 DevOps

Discover by topic

Categories

Travel Nature City Adventure

Completed all the step.