**🚀 DevOps Tools Used in Travel Diaries MERN Project**

This document explains the DevOps tools integrated into the project, including Docker, Docker Compose, Jenkins, and Ansible, along with the code snippets used for containerization and deployment.

**1. Docker**

Docker is used to containerize both the frontend and backend of the MERN stack application. This ensures the application behaves the same way across different environments.

* Frontend Docker Image: prathamdandale/frontend-image:latest
* Backend Docker Image: prathamdandale/backend-image:latest

These images are built and later deployed using Docker Compose and Jenkins.

**2. Docker Compose**

docker-compose.yml simplifies managing multi-container applications. It defines how the frontend and backend services are run.

**docker-compose.yml**

version: "1"

services:

frontend:

image: prathamdandale/frontend-image:latest

container\_name: frontend-container

ports:

- "3000:3000"

backend:

image: prathamdandale/backend-image:latest

container\_name: backend-container

ports:

- "8000:8000"

networks:

frontend\_network:

external: true

backend\_network:

external: true

**Key Highlights:**

* services: defines frontend and backend containers.
* ports: exposes the app to the host machine (frontend on 3000, backend on 8000).
* networks: connects the containers via Docker's networking.

**3. Jenkins (CI/CD Pipeline)**

Jenkins automates the entire CI/CD pipeline, from code checkout to deployment.

**Jenkinsfile**

pipeline {

environment {

fe\_image = ""

be\_image = ""

}

agent any

stages {

stage('Stage 1: Pull code from Github') {

steps {

git branch: 'main', url: 'https://github.com/pratham0609/SPE-TravelDiaries.git'

}

}

stage('Stage 2: Build frontend docker image') {

steps {

echo "Build frontend docker image"

sh "docker build -t prathamdandale/frontend-image:latest frontend/"

}

}

stage('Stage 3: Build backend docker image') {

steps {

echo "Build backend docker image"

sh "docker build -t prathamdandale/backend-image:latest backend/"

}

}

stage('Stage 4: Push frontend & backend images to Docker Hub') {

steps {

script {

docker.withRegistry('', 'DockerHubCredentials') {

echo "Push frontend Docker Image to Docker Hub"

sh "docker push prathamdandale/frontend-image:latest"

echo "Push backend Docker Image to Docker Hub"

sh "docker push prathamdandale/backend-image:latest"

}

}

}

}

stage('Stage 5: Remove dangling images') {

steps {

script {

sh 'docker image prune -f'

}

}

}

stage('Stage 6: Deploy containers on target machines using Ansible') {

steps {

ansiblePlaybook installation: 'Ansible',

playbook: 'ansible/deploy\_containers.yml',

inventory: 'ansible/inventory',

credentialsId: 'LocalhostUserCredentials'

}

}

}

}

**What It Does:**

* Pulls code from GitHub
* Builds Docker images for frontend and backend
* Pushes them to Docker Hub
* Triggers Ansible for deployment
* Cleans up dangling images to save disk space

**4. Ansible (Container Deployment)**

Ansible automates the process of deploying the application on the target machine.

**ansible/inventory**

[localhost]

127.0.0.1 ansible\_connection=local ansible\_user=pratham

**ansible/deploy\_containers.yml**

---

- name: Deploy and run frontend and backend containers on all target machines

hosts: all

vars:

ansible\_python\_interpreter: /usr/bin/python3

tasks:

- name: Start docker service

service:

name: docker

state: started

- name: Copy Docker compose file from host machine to target machine

copy:

src: ../docker-compose.yml

dest: ./

- name: Remove containers from target machine if they already exist

shell:

cmd: |

if [ "$(docker ps -a -q -f name=frontend-container)" ]; then

docker stop frontend-container

docker rm frontend-container

fi

if [ "$(docker ps -a -q -f name=backend-container)" ]; then

docker stop backend-container

docker rm backend-container

fi

- name: Run the containers using docker compose

shell: docker compose up -d

**How It Works:**

* Starts Docker service if it's not running
* Copies docker-compose.yml to the target machine
* Stops and removes old containers if they already exist
* Deploys containers using docker compose up -d

**✅ Summary: DevOps Workflow**

GitHub → Jenkins → Docker → Docker Hub → Ansible → Running Containers

1. Code is pushed to GitHub.
2. Jenkins pulls the code and builds images.
3. Docker images are created and pushed to Docker Hub.
4. Ansible deploys these images to the target system using Docker Compose.

This pipeline ensures automation, consistency, and fast delivery — essential for modern software development practices.