**Social Echo: Containerized Full-Stack social media (Three-tier) App with CI/CD and Docker Swarm Deployment**

**Author: Nitesh Nepali**

**DevOps Engineer/ Cyber Security Enthusiast: (CCNA, AWS-SAA, Linux (RHCSA), DevSecOps)**

**Email:** [**pingnitesh07@gmail.com**](mailto:pingnitesh07@gmail.com)

**1. Project Overview**

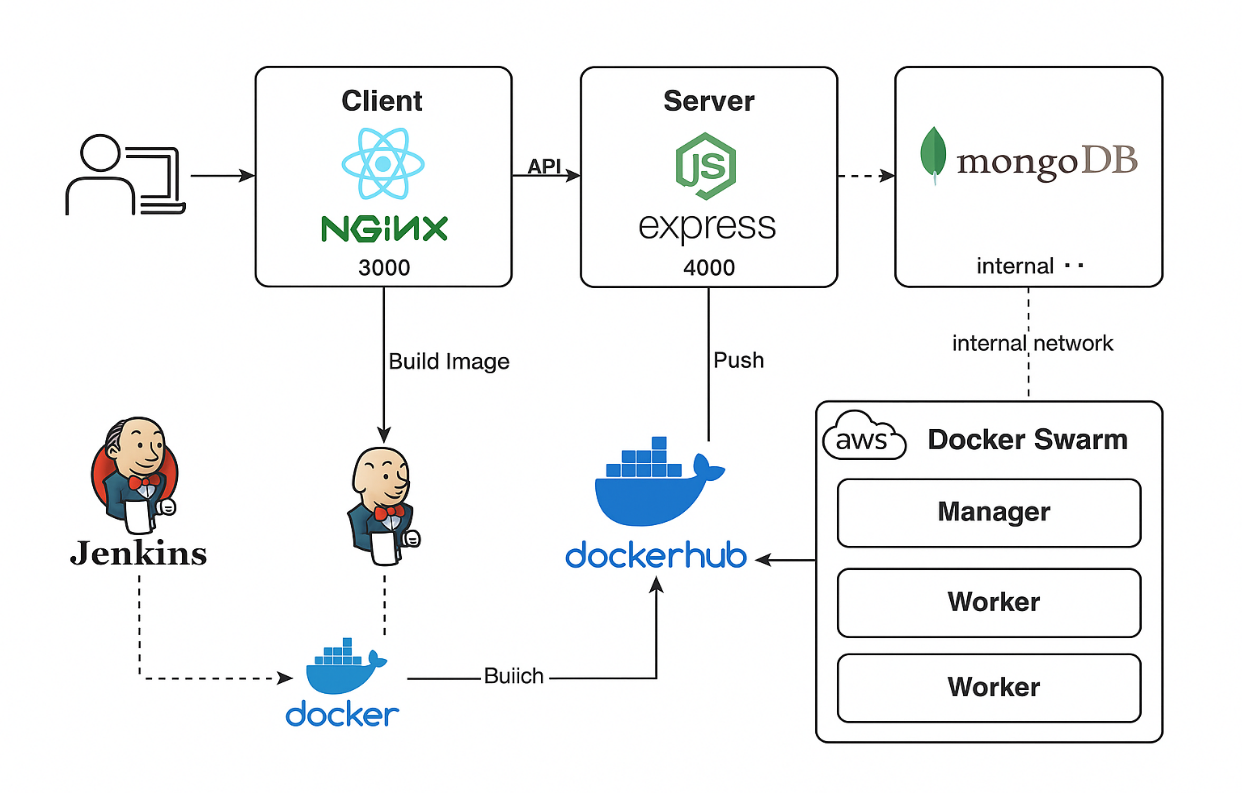
Social Echo is a **multi-tier(three-tier)** social media application designed for seamless content sharing and interaction. This project implements a complete **DevOps lifecycle** — from development to deployment — using **Docker Swarm**, **Jenkins CI/CD pipelines**, and **environment-specific configurations**.

The application is split into three core services:

* **Client** – Frontend built with modern JavaScript framework (React/Vue).
* **Server** – Backend API with authentication, content moderation, and messaging features.
* **Database** – MongoDB as the persistent data store.

The entire stack is containerized, orchestrated on a **multi-node Docker Swarm cluster**, and automatically deployed using Jenkins pipelines triggered by GitHub webhooks.

**2. Architectural Diagram**

****

**4. Technologies Used**

* **Frontend:**  React (served via Nginx)
* **Backend:** Node.js, Express.js
* **Database:**  MongoDB
* **Container:** Docker
* **Orchestration:** Docker Swarm (multi-node)
* **CI/CD:** Jenkins (multi-pipeline setup)
* **Version Control:** GitHub
* **Secrets Management:** .env files per service
* **Networking:** **Docker overlay network**
* **Cloud:** Aws

I have divided the Deployment process into 4 phases.

**Phase 1: Testing the Project Locally**  
Testing the app locally on dev environment

**Phase 2: Dockerizing the Application**  
Containerize the application using Docker by creating Dockerfiles and building Docker images.

**Phase 3: Creating Jenkins Pipeline**  
Set up a Jenkins pipeline to automate building, testing, and deploying the Dockerized application.

**Phase 4: Deploying to Docker Swarm**  
Deploy the Docker containers to a Docker Swarm cluster for scalable and fault-tolerant orchestration.

**Phase 1: Testing the project Locally**

**In this we test the app to run locally.**

**First, we have to install node js, MongoDB and initialize the Mongo DB before running the app and follow the below process.**

**git clone https://github.com/nz-m/SocialEcho.git**

**cd SocialEcho/server**

**npm install**

**# Create. env with MongoDB URI, JWT secret as stated in .env.example file in the rep of this project**

**npm start**

**# In a new terminal**

**cd SocialEcho/client**

**npm install**

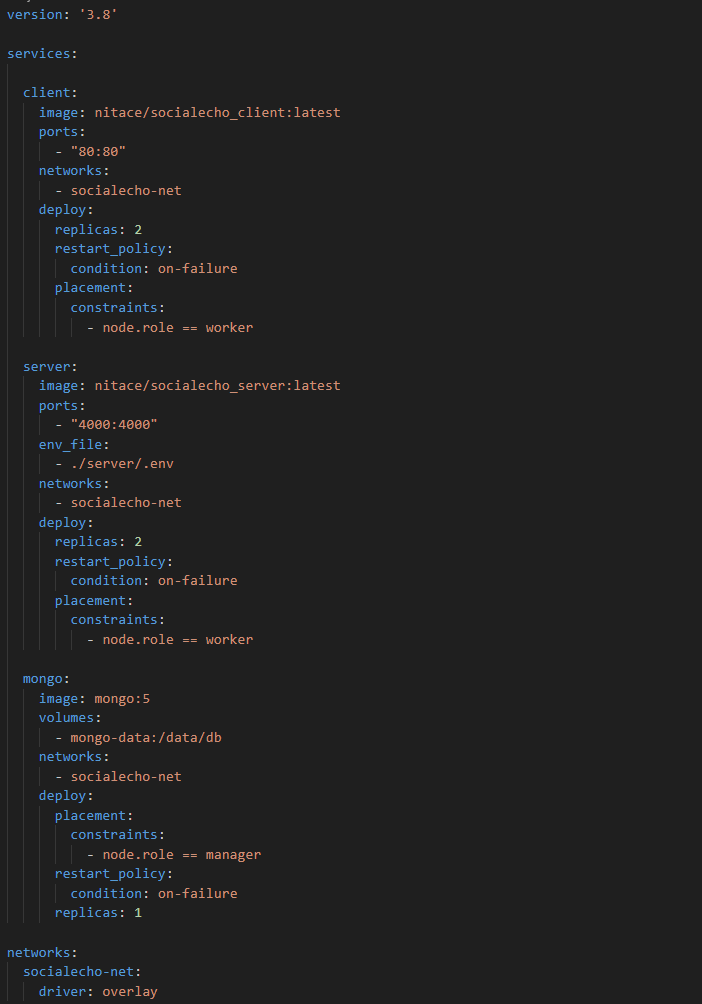
**# Create. env with REACT\_APP\_API\_URL as stated in .env.example file in the rep of this project**

**npm start**

**Phase 4: Deploying to Docker Swarm**  
In this phase, Docker Swarm Cluster is setup and docker-stack.yaml file I created for the deployment.

4.1 Creating Docker Swarm Cluster

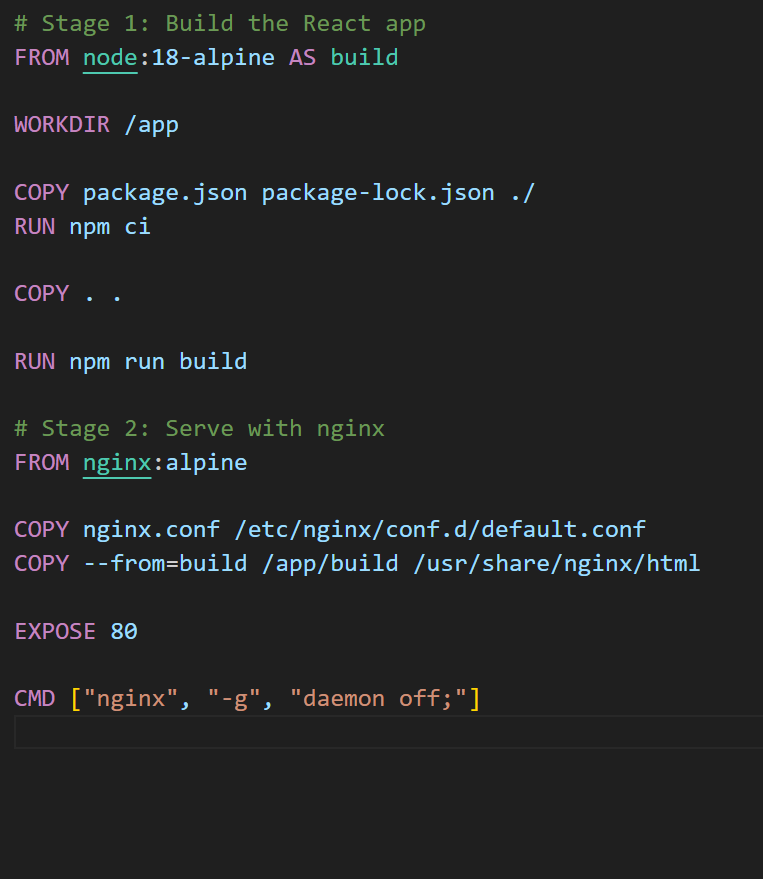
4.2 docker-stack.yaml file for deployment



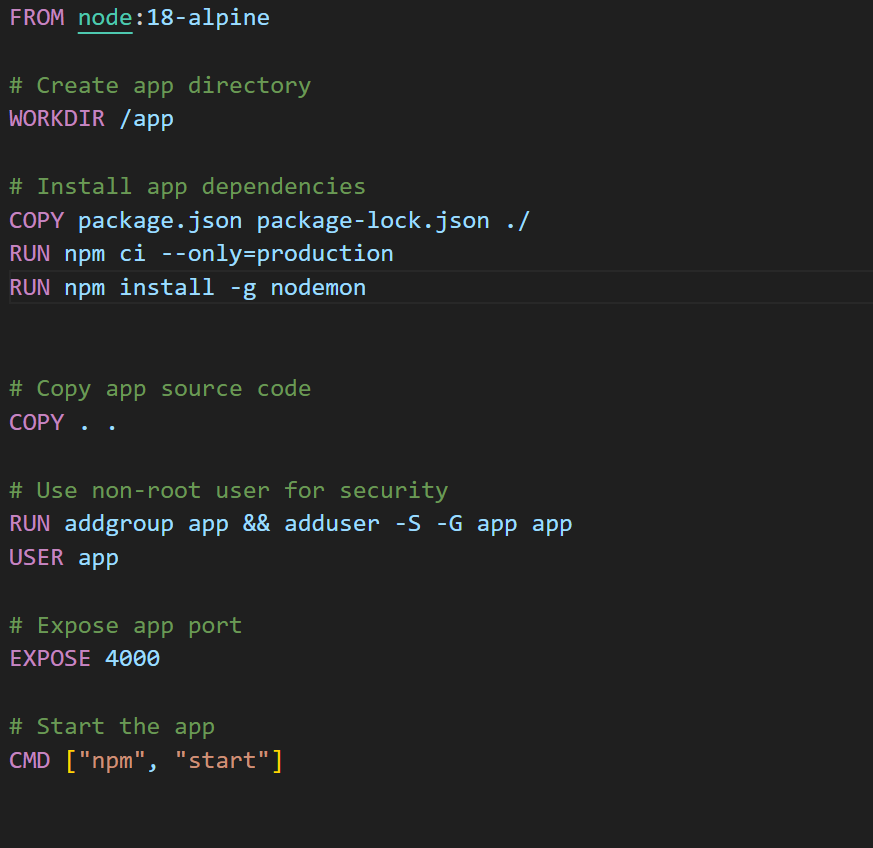
**Phase 2: Dockerizing the Application**

In this phase the frontend and backend part of the application is dockize with respective Docker file and env specific files.

**2.1 Dockerfile of the client (front-end part)**



**2.2 Dockerfile of the server(backend) app.**

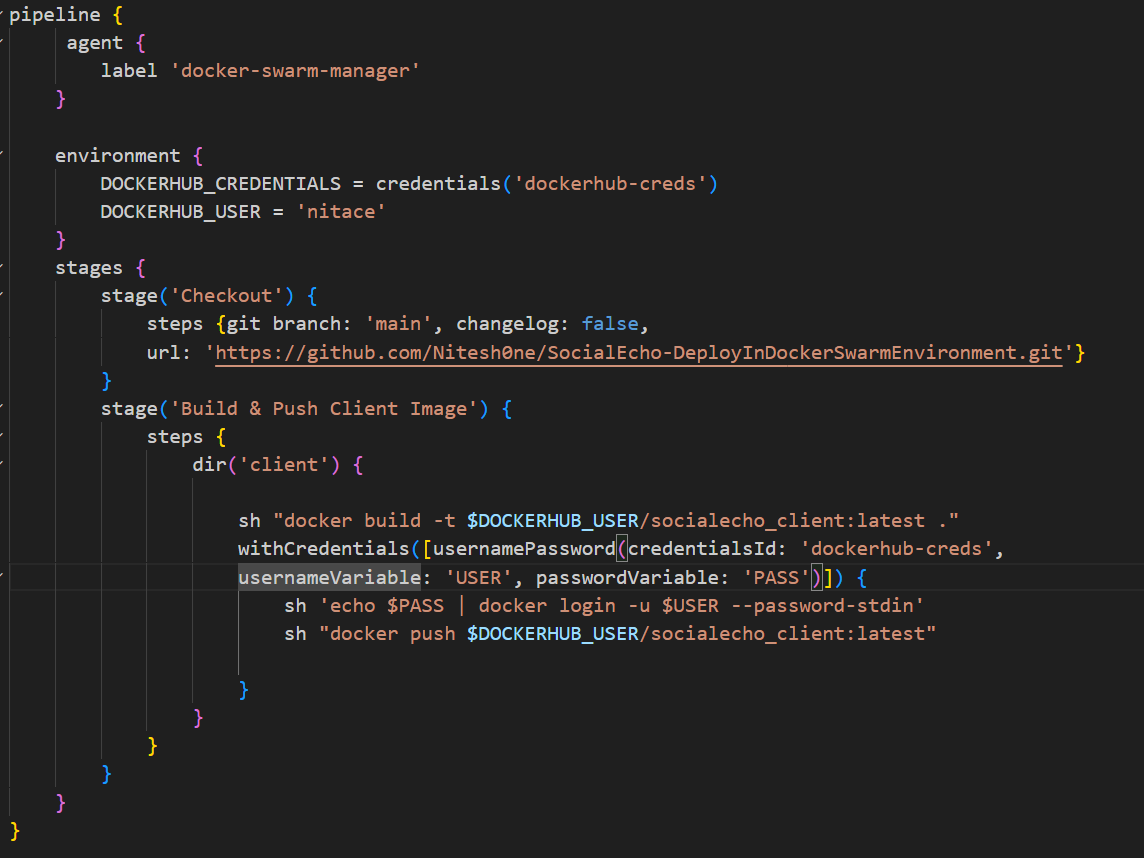


**Phase 3: Creating Jenkins Pipeline**

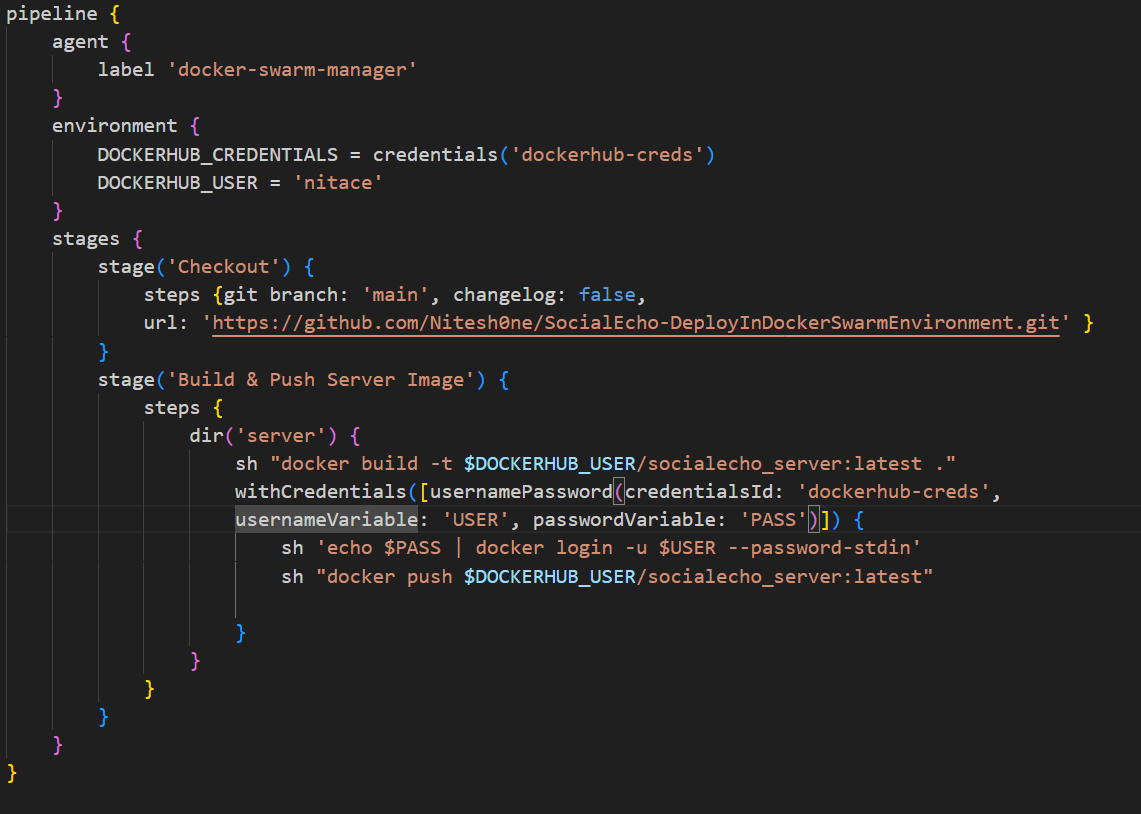
In this phase, **Jenkins Master and Slave** is setup with **dedicated Jenkins Master Server** and **Docker Swarm Manger Node as Slave** for the building image and app deployment.

A separate job is created to build client image, server image and for deployment

**3.1 Jenkinfile for the building Client Image and pushing to the Docker Hub**



**3.2 Jenkinsfile for building the server image and pushing to the Docker Hub**



Note: Docker file and Jenkinsfile should be on respective folder

**Phase 3.3: Jenkins File for the Final app deployment deployment**

