Project Title

StockTradePro Infra Setup and Deployment Using AWS DevOps Pipeline

Project Objective

To build a robust, scalable, and production-ready infrastructure for the StockTradePro web application using AWS services, containerization, CI/CD automation, and monitoring tools. The objective is to ensure efficient deployment, secure data handling, and high system availability with minimal manual intervention.

Technologies and Tools

- **Cloud Provider**: AWS (EC2, S3, RDS, ECS, IAM, CloudWatch)
- **IaC**: Terraform
- **Containerization**: Docker
- **CI/CD**: GitHub Actions (or CodePipeline)
- **Monitoring**: CloudWatch, Prometheus (optional)
- **Security**: IAM Roles, Security Groups, Secrets Manager
- **VCS**: Git + GitHub
- **Web Hosting**: ECS with Fargate or EC2-based Nginx proxy
- **Database**: AWS RDS (MySQL/PostgreSQL)

Phase 1: Containerization & AWS Deployment (Weeks 1-3)

Deadline: 5/08/2025

Week 1 – Environment & Infrastructure Setup

Goal: Define and provision the base AWS infrastructure and environment.

Tasks:

- Setup VPC, subnets (public/private), route tables.
- Create EC2 instance or ECS Cluster.
- Configure S3 bucket (for static asset storage / backups).
- Setup RDS (MySQL/PostgreSQL) with appropriate security group.
- Initialize Terraform backend with S3 & DynamoDB for state locking.

Week 2 – Dockerization of Frontend & Backend

Goal: Containerize the application for consistent deployment.

Tasks:

- Create Dockerfiles for both frontend and backend.
- Test containers locally using Docker Compose.
- Push Docker images to AWS ECR (Elastic Container Registry).
- Define ECS Task Definitions (if using ECS) or prepare EC2 launch templates.

Week 3 – AWS Deployment

Goal: Deploy the containerized app on AWS infrastructure.

Tasks:

- Setup ECS Fargate or EC2 Auto Scaling for app hosting.
- Configure ALB (Application Load Balancer) or Nginx proxy.
- Connect frontend to backend and backend to RDS.
- Validate full application functionality in a cloud environment.

Phase 2: CI/CD, Monitoring & Security (Weeks 4-6)

Deadline: 5/09/2025

Week 4 – CI/CD Automation

Goal: Enable automated builds, testing, and deployments.

Tasks:

- Setup GitHub Actions workflows for CI (lint, build, test).
- Integrate Docker image build and push to ECR on merge to main.
- Automate ECS deployment on image push or use AWS CodePipeline.
- Add deployment status badges and notifications.

Week 5 – Monitoring and Logging

Goal: Add observability and performance tracking to your stack.

Tasks:

- Enable AWS CloudWatch logs for ECS and RDS.
- Add custom CloudWatch metrics and alarms (CPU, memory, HTTP status).
- (Optional) Setup Prometheus + Grafana stack for detailed metrics.
- Setup alerting via email or Slack.

Week 6 – Security and Optimization

Goal: Harden infrastructure and finalize production readiness.

Tasks:

- Secure environment variables using AWS Secrets Manager.
- Enforce HTTPS (via ACM + ALB).
- Configure IAM roles and least-privilege access policies.
- Enable backups for RDS and lifecycle rules for S3.
- Perform load testing and scale validation.

Final Deliverables & Submission Guidelines

Required Deliverables

The final submission must include the following components:

GitHub Repository

- Complete source code including:
    - `frontend/`
    - `backend/`
    - `infra/`
- Clean and structured commit history.

Technical Documentation

- Deployment guide with environment variables, API usage, infrastructure flow.
- CI/CD pipeline architecture diagram.
- `.env.example` for both frontend and backend.

Screenshots or Video Recording

- ECS/RDS setup screenshots
- GitHub Actions or CodePipeline run
- Live app screenshots or demo video (max 2–3 mins)

Final Presentation Slides

- Project overview and tech stack
- Infrastructure and deployment pipeline
- Challenges & learnings
- Future scope

Submission & Collaboration Guide

- **Version Control (Git & GitHub):**
    - All code and configuration must be managed in the designated GitHub repository.
    - Follow the **GitFlow** or **Feature Branch** workflow. Do not commit directly to the `main` branch.
    - Create new branches for each feature or task (e.g., `feature/backend-dockerization`, `fix/alb-routing-rule`).
    - Use Pull Requests (PRs) to merge changes into the `main` branch. Each PR must be reviewed by at least one other team member before merging.
    - Write clear and descriptive commit messages (e.g., `feat: Add Dockerfile for backend service`, `docs: Update README with setup instructions`).

Evaluation Criteria

Criteria	Weightage
Infrastructure setup (Terraform)	20%
Containerization (Docker)	15%
CI/CD automation	20%
AWS deployment	20%
Monitoring & security	15%
Documentation & collaboration	10%