**DEVOPS**

**ASSIGNMENT -1**

1. **Which way of installing Jenkins would you prefer and why?**

I prefer installing Jenkins using Docker because it is quick, simple to manage, eliminates dependency conflicts, and provides an isolated environment, making updates and rollbacks seamless.

✅ **Rapid Deployment** – No need to set up Java or other prerequisites manually.  
✅ **Effortless Cleanup** – Remove the container anytime without hassle.  
✅ **Consistent Across Platforms** – Runs uniformly across different operating systems.  
✅ **Clean System** – Keeps your machine free from unnecessary installations.  
✅ **Smooth Upgrades** – Easily update by pulling the latest Jenkins image.

1. **Write down the steps involved in building a web app, testing it and deployed to QA and Production.**
2. **Development Phase: Building the Web App**

**Step 1: Requirement Gathering & Planning**

✅ Define project objectives, key features, and the preferred technology stack (e.g., MERN,Django,etc.).  
 ✅ Set up version control using platforms like GitHub, GitLab, or Bitbucket.  
 ✅ Establish a development workflow (Kanban, Agile, Scrum).

**Step 2: Setting Up the Development Environment**

✅ Install necessary tools (Node.js, Python, Docker, databases, etc.).  
✅ Initialize the project using package managers (e.g., npm init, pip install).  
✅ Configure a development environment with an IDE (VS Code, WebStorm) and frameworks (React, Express, etc.).

**Step 3: Writing Code**

✅ Develop the frontend using React, Angular, Vue, or plain HTML/CSS.  
 ✅ Build the backend with Node.js, Django, Flask, or Spring Boot.  
 ✅ Set up and connect a database (MongoDB, PostgreSQL, MySQL).  
 ✅ Implement authentication (JWT, OAuth, Firebase).

**Step 4: Version Control**

Create a Git repository and push code.

git init

git add .

git commit -m "Initial commit"

git push origin main

**2. Testing Phase: Ensuring Quality**

**Step 5: Unit Testing**

Write unit tests for individual components using Jest, Mocha, or PyTest.

test('adds 1 + 2 to equal 3', () => {

expect(1 + 2).toBe(3);

});

**Step 6: Integration & API Testing**

✅ Verify API functionality using Postman, Newman, or Supertest.  
 ✅ Automate API testing with Cypress or Selenium.

**Step 7: UI/UX Testing**

✅ Ensure cross-browser compatibility (Chrome, Firefox, Edge).  
 ✅ Test mobile responsiveness across different screen sizes.

**Step 8: Security Testing**

✅ Run vulnerability scans with OWASP ZAP or Burp Suite.  
 ✅ Implement security measures such as SSL, CORS, and authentication protocols.

**3. Deployment Phase: QA & Production**

**Step 9: Deploy to QA Environment**

Containerize the application using Docker:

docker build -t myapp .

docker run -d -p 3000:3000 myapp

Deploy to a QA server (AWS, DigitalOcean, Azure).

QA team runs manual and automated tests.

**Step 10: Deploy to Production**

✅ Implement CI/CD pipelines with GitHub Actions, Jenkins, or GitLab CI/CD.  
 ✅ Deploy using Kubernetes, Docker Swarm, or AWS Elastic Beanstalk.  
 ✅ Perform load testing using JMeter before launch.  
 ✅ Ensure zero downtime deployment (Blue-Green Deployment).

**4.Post-Deployment: Monitoring & Maintenance**

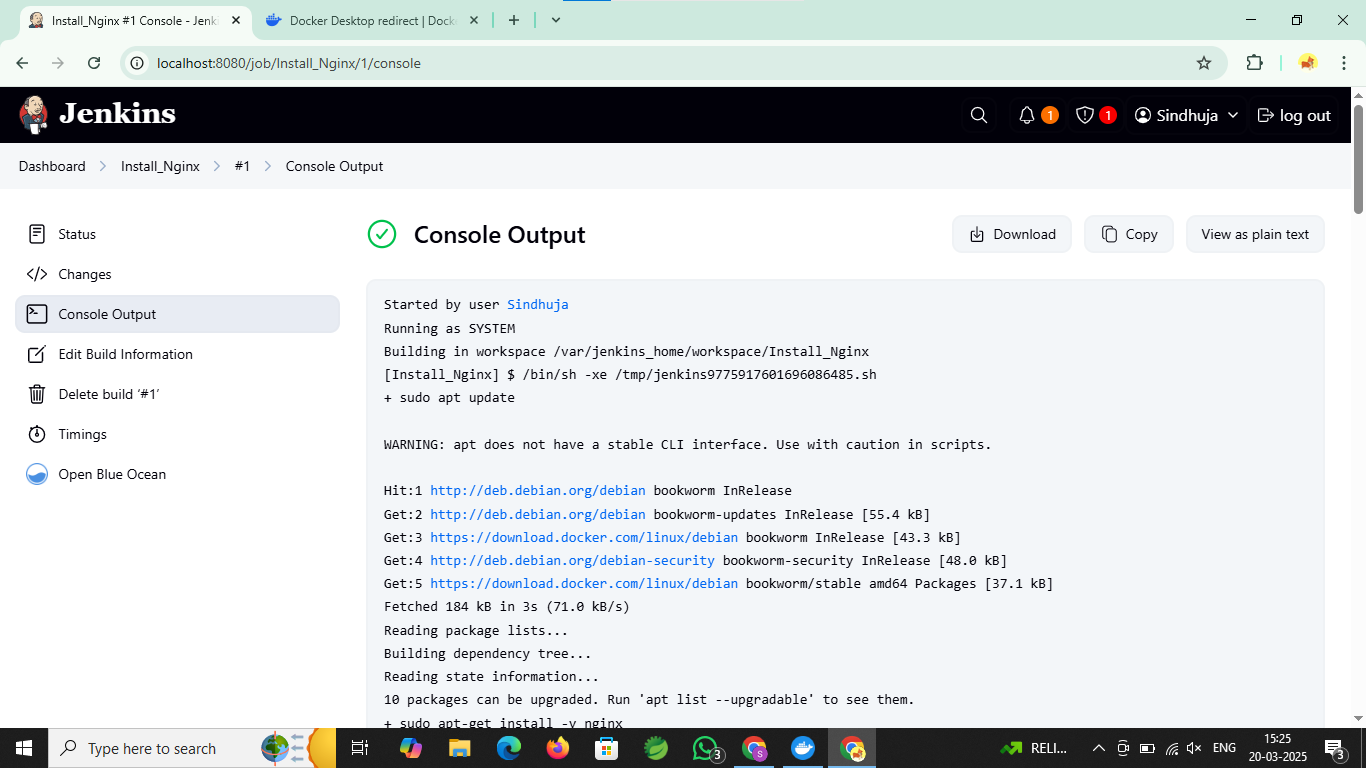
**Step 11: Monitoring**

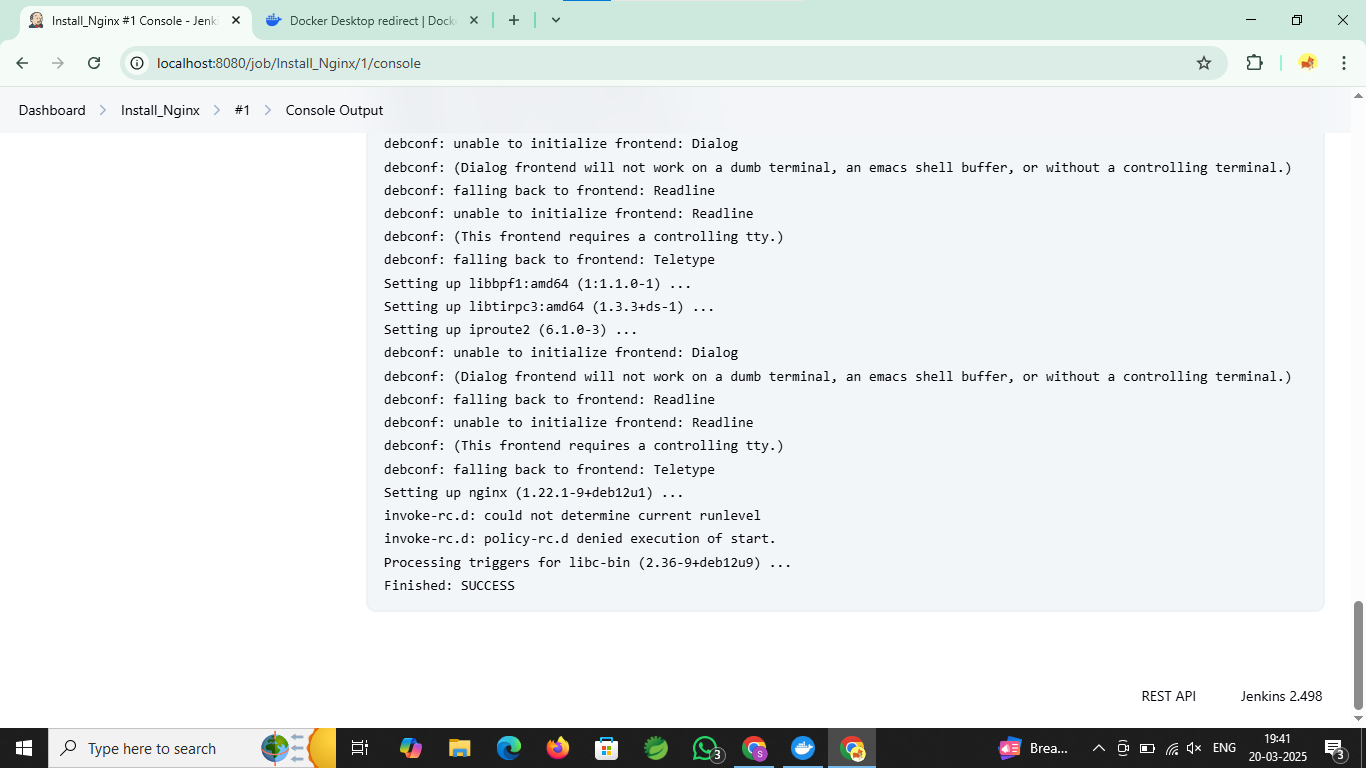
✅ Use monitoring tools like Prometheus, Grafana, or Datadog.  
 ✅ Set up logging with ELK Stack (Elasticsearch, Logstash, Kibana).

**Step 12: Bug Fixes & Updates**

✅ Collect user feedback and resolve issues.  
 ✅ Roll out new features using feature flagging.  
 ✅ Maintain rollback strategies for handling failures efficiently.

**OUTPUT:**

****

****