DEVOPS ASSIGNMENT – 1

1. Preferred Jenkins Installation Method

I prefer installing Jenkins using Docker because it is fast, easy to manage, avoids dependency issues, and provides an isolated environment, making upgrades and rollbacks hassle-free.

✅ Quick Setup – No need to manually install Java or other dependencies.
✅ Easy Cleanup – Simply remove the container when no longer needed.
✅ Portability – Ensures consistent performance across different OS environments.
✅ No System Pollution – Does not install additional packages on the host machine.
✅ Easy Upgrades – Upgrading is as simple as pulling the latest Jenkins image.

---

2. Steps for Building, Testing, and Deploying a Web App

1. Development Phase: Building the Web App

Step 1: Requirement Gathering & Planning

- Define project scope, features, and technology stack (e.g., MERN, Django, etc.).
- Set up a repository (GitHub/GitLab/Bitbucket) for version control.
- Establish a development workflow (Kanban, Agile, Scrum).

Step 2: Setting Up the Development Environment

- Install required software (Node.js, Python, Docker, databases, etc.).
- Initialize the project using package managers.
- Configure development tools like VS Code, WebStorm, and frameworks.

Step 3: Writing Code

- Develop the frontend using React, Angular, Vue, or HTML/CSS.
- Implement the backend using Node.js, Django, Flask, or Spring Boot.
- Integrate a database (MongoDB, PostgreSQL, MySQL).
- Implement authentication mechanisms.

Step 4: Version Control

- Use Git for version control and push the code to a repository.

---

2. Testing Phase: Ensuring Quality

Step 5: Unit Testing

- Write unit tests for individual components.

Step 6: Integration & API Testing

- Test API endpoints with tools like Postman and automate API tests.

Step 7: UI/UX Testing

- Ensure cross-browser compatibility.
- Conduct mobile responsiveness testing.

Step 8: Security Testing

- Perform vulnerability scans.
- Implement security measures like SSL, CORS, and authentication checks.

---

3. Deployment Phase: QA & Production

Step 9: Deploy to QA Environment

- Containerize the application using Docker.
- Deploy to a QA server.
- QA team executes manual and automated tests.

Step 10: Deploy to Production

- Implement CI/CD pipelines.
- Deploy using Kubernetes, Docker Swarm, or AWS Elastic Beanstalk.
- Perform load testing before the final launch.
- Use zero-downtime deployment strategies.

---

4. Post-Deployment: Monitoring & Maintenance

Step 11: Monitoring

- Utilize monitoring tools for performance tracking.

- Set up logging for debugging and analysis.

Step 12: Bug Fixes & Updates

- Gather user feedback and address issues promptly.
- Deploy new features using feature flagging techniques.
- Maintain rollback strategies in case of failures.

This structured approach ensures a smooth workflow from development to deployment, maintaining quality and efficiency at every stage.