**DEVOPS**

**ASSIGNMENT – 1**

**1. Preferred Method of Installing Jenkins and Justification**

I prefer installing Jenkins using Docker due to its efficiency, ease of management, and ability to provide an isolated environment. This method eliminates dependency issues, streamlines upgrades, and simplifies rollbacks.

**Advantages of Using Docker for Jenkins Installation:**

* **Quick Setup** – Eliminates the need for manual installation of Java and other dependencies.
* **Easy Cleanup** – Containers can be removed easily when no longer needed.
* **Portability** – Ensures consistency across different operating systems.
* **No System Pollution** – Prevents installation of unnecessary packages on the host machine.
* **Easy Upgrades** – Allows seamless updates by pulling the latest Jenkins image.

**2. Steps for Building, Testing, and Deploying a Web Application**

**1. Development Phase: Building the Web App**

**Step 1: Requirement Gathering & Planning**

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

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

* Install necessary software (Node.js, Python, Docker, databases).
* Initialize the project using package managers (e.g., npm init, pip install).
* Configure a code editor (VS Code, WebStorm) and frameworks (React, Express).

**Step 3: Writing Code**

* Develop the frontend using React, Angular, or Vue.
* Implement backend functionality using Node.js, Django, Flask, or Spring Boot.
* Integrate a database such as MongoDB, PostgreSQL, or MySQL.
* Implement authentication mechanisms (JWT, OAuth, Firebase).

**Step 4: Version Control**

* Initialize and commit code to a Git repository:
* 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 using Jest, Mocha, or PyTest.
* test('adds 1 + 2 to equal 3', () => {
* expect(1 + 2).toBe(3);
* });

**Step 6: Integration & API Testing**

* Test API endpoints using Postman, Newman, or Supertest.
* Automate API testing with Cypress or Selenium.

**Step 7: UI/UX Testing**

* Verify cross-browser compatibility (Chrome, Firefox, Edge).
* Perform mobile responsiveness testing.

**Step 8: Security Testing**

* Conduct vulnerability scans using OWASP ZAP or Burp Suite.
* Implement security best practices such as SSL, CORS policies, and authentication checks.

**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).
* Perform manual and automated testing in the QA environment.

**Step 10: Deploy to Production**

* Set up CI/CD pipelines using GitHub Actions, Jenkins, or GitLab CI/CD.
* Deploy using Kubernetes, Docker Swarm, or AWS Elastic Beanstalk.
* Conduct load testing with JMeter before production release.
* Implement zero-downtime deployment strategies (e.g., Blue-Green Deployment).

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

**Step 11: Monitoring**

* Monitor application performance using Prometheus, Grafana, or Datadog.
* Set up centralized logging with the ELK Stack (Elasticsearch, Logstash, Kibana).

**Step 12: Bug Fixes & Updates**

* Collect user feedback and resolve reported issues.
* Deploy new features using feature flagging.
* Maintain rollback strategies for emergency fixes.