**DEVOPS**

**ASSIGNMENT -1**

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

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

* Easy to set up and run in a containerized environment.
* No need to install dependencies manually.
* Can be easily deployed on any platform that supports Docker.
* Allows quick updates and rollbacks with different Jenkins versions.

1. **Write down the steps involved in building a web app, testing it and deployed to QA and Production.**

## **1. Planning & Requirement Analysis**

* Define the features and functionalities.
* Choose the tech stack (Frontend: React/Angular, Backend: Node.js/Python, Database: MySQL/MongoDB).
* Design wireframes and UI/UX.

## **2. Development**

### **Frontend Development**

* Set up the framework (e.g., React, Angular, Vue).
* Create UI components and integrate APIs.
* Ensure responsive design with CSS frameworks like Tailwind or Bootstrap.

### **Backend Development**

* Set up a server using Node.js, Django, or Flask.
* Develop REST APIs or GraphQL endpoints.
* Implement authentication and authorization.
* Integrate the database and optimize queries.

### **Database Setup**

* Design the database schema.
* Write SQL or NoSQL queries for CRUD operations.

## **3. Testing the Web App**

### **Unit Testing** (Ensures individual components work correctly)

* Use Jest, Mocha, PyTest for backend testing.
* Use React Testing Library or Cypress for frontend testing.

### **Integration Testing** (Verifies API connections and database interactions)

* Test API endpoints using Postman or Newman.
* Mock database responses and verify API correctness.

### **End-to-End (E2E) Testing** (Ensures the entire flow works)

* Use Selenium, Cypress, or Playwright to automate UI testing.

### **Security Testing**

* Validate authentication and authorization.
* Check for vulnerabilities using OWASP ZAP.

## **4. Deployment to QA (Testing Environment)**

### **Build and Package the App**

* Run npm build or yarn build (Frontend).
* Use Docker or a cloud environment (AWS, Azure, GCP) for backend.

### **Deploy to QA Server**

* Push the code to a QA branch in Git.
* Use CI/CD (GitHub Actions, Jenkins, GitLab CI/CD) to automate deployment.
* Deploy on a staging environment using **Netlify, Vercel (Frontend)** or **AWS EC2, DigitalOcean, or Kubernetes (Backend)**.

### **QA Testing**

* QA team tests all features and logs defects.
* Fix reported bugs and retest.

## **5. Deployment to Production**

### **Final Testing in Staging Environment**

* Perform load testing using JMeter.
* Ensure all services work properly.

### **Deploy to Production Server**

* Merge QA-approved code to the main branch.
* Deploy using CI/CD pipelines (Jenkins, GitHub Actions).
* Configure database migrations if needed.
* Set up monitoring tools (New Relic, Datadog).

### **Post-Deployment Checks**

* Verify logs and performance metrics.
* Run a smoke test to check if core features work.
* Monitor user feedback and fix issues quickly.