**DEVOPS**

**ASSIGNMENT -1**

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

The preferred way of installing **Jenkins** depends on the use case, but generally, **Docker** is the most recommended method.

* **Easy Installation & Cleanup** – No need to install dependencies manually.
* **Portability** – Can run Jenkins on any system with Docker installed.
* **Isolation** – No conflicts with other software.
* **Quick Updates** – Easily upgrade/downgrade Jenkins versions.
* **Consistency** – Ensures the same setup across different environments.

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

**1. Development Stage: Creating the Web Application**

**Step 1: Planning and Requirement Analysis**

* Identify key project requirements, technology stack (e.g., MERN, Django, Spring Boot), and features.
* Set up a version control repository on **GitHub, GitLab, or Bitbucket**.
* Establish a development workflow (Scrum, Agile, or Kanban).

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

* Install essential tools like **Node.js, Python, Docker, and databases (MongoDB, PostgreSQL, MySQL)**.
* Initialize the project using appropriate package managers:
* npm init # For JavaScript-based projects
* pip install -r requirements.txt # For Python-based projects
* Configure development tools such as **VS Code, WebStorm, and necessary frameworks** (React, Angular, Express, Flask).

**Step 3: Writing Code and Implementing Features**

* **Frontend Development:** Build the user interface using **React, Angular, Vue, or HTML/CSS**.
* **Backend Development:** Develop APIs and business logic using **Node.js, Django, Flask, or Spring Boot**.
* **Database Integration:** Connect and configure **MongoDB, PostgreSQL, or MySQL** for data storage.
* **Authentication Setup:** Implement user authentication using **JWT, OAuth, or Firebase**.

**Step 4: Managing Code with Version Control**

* Initialize and maintain code in a **Git repository**:
* git init
* git add .
* git commit -m "Initial project setup"
* git push origin main

**2. Testing Phase: Ensuring Software Quality**

**Step 5: Unit Testing**

* Develop unit tests for individual components using **Jest, Mocha, or PyTest**.
* test('sum of 1 and 2 should be 3', () => {
* expect(1 + 2).toBe(3);
* });

**Step 6: Integration & API Testing**

* Validate API functionality using **Postman, Newman, or Supertest**.
* Automate API tests with **Cypress, Selenium, or JMeter**.

**Step 7: UI & Compatibility Testing**

* Perform **cross-browser testing** to ensure compatibility with **Chrome, Firefox, Edge, and Safari**.
* Check mobile responsiveness across different screen sizes.

**Step 8: Security & Performance Testing**

* Identify vulnerabilities using **OWASP ZAP or Burp Suite**.
* Implement security measures like **SSL, CORS policies, and authentication validation**.

**3. Deployment Phase: Releasing the Application**

**Step 9: Deploying to QA (Testing) Environment**

* Containerize the application with **Docker**:
* docker build -t myapp .
* docker run -d -p 3000:3000 myapp
* Deploy the application to a **QA server** (AWS, DigitalOcean, Azure) for testing.
* Conduct manual and automated testing before production deployment.

**Step 10: Deploying to Production**

* Automate deployment using **CI/CD pipelines** (Jenkins, GitHub Actions, GitLab CI/CD).
* Deploy on production infrastructure like **Kubernetes, AWS Elastic Beanstalk, or Docker Swarm**.
* Perform **load testing** with **JMeter** to assess system performance.
* Implement **zero-downtime deployment strategies** (Blue-Green Deployment).

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

**Step 11: Application Monitoring**

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

**Step 12: Continuous Updates & Fixes**

* Gather user feedback and address issues through regular patches.
* Introduce new features gradually using **feature flagging techniques**.
* Maintain a rollback plan for handling unexpected failures.