**DEVOPS ASSIGNMENT - 1**

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

I would choose to install Jenkins using **Docker** due to the following advantages:

* **Quick Setup:** Eliminates the need for manually installing Java or other dependencies.
* **Simplified Cleanup:** Easily remove the container when no longer needed, reducing clutter.
* **Portability:** Consistent behavior across multiple operating systems.
* **No System Pollution:** Prevents unnecessary software installation on the host machine.
* **Easy Upgrades:** Effortlessly update to the latest Jenkins image by pulling from the Docker registry.

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

**Phase 1: Development – Crafting the Web Application**

**Step 1: Gathering Requirements & Planning**

* Define the project's scope, core features, and technology stack (e.g., MERN, Django, etc.).
* Set up a version control system (e.g., GitHub, GitLab, Bitbucket).
* Determine the development process (Agile, Scrum, Kanban).

**Step 2: Environment Setup**

* Install necessary tools (e.g., Node.js, Python, Docker, databases).
* Initialize the project with package managers (e.g., npm init, pip install).
* Choose and configure your development environment (VS Code, WebStorm).

**Step 3: Coding the Application**

* Develop the frontend (React, Angular, Vue, HTML/CSS).
* Build the backend (Node.js, Django, Flask, Spring Boot).
* Integrate databases (MongoDB, PostgreSQL, MySQL).
* Implement user authentication (JWT, OAuth, Firebase).

**Step 4: Version Control & Collaboration**

* Set up a Git repository and push initial code changes:

bash

Copy

git init

git add .

git commit -m "Initial commit"

git push origin main

**Phase 2: Testing – Ensuring Quality and Reliability**

**Step 5: Unit Testing**

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

javascript

Copy

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

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

});

**Step 6: API & Integration Testing**

* Use Postman, Newman, or Supertest to test APIs.
* Automate API testing with tools like Cypress or Selenium.

**Step 7: UI/UX Testing**

* Verify cross-browser compatibility (Chrome, Firefox, Edge).
* Ensure mobile responsiveness across various devices.

**Step 8: Security Testing**

* Run vulnerability scans with tools like OWASP ZAP or Burp Suite.
* Implement best practices such as SSL, CORS policies, and authentication checks.

**Phase 3: Deployment – From QA to Production**

**Step 9: Deploying to QA Environment**

* Containerize the app using Docker:

bash

Copy

docker build -t myapp .

docker run -d -p 3000:3000 myapp

* Deploy the application on a QA server (AWS, DigitalOcean, Azure).
* The QA team conducts manual and automated testing.

**Step 10: Production Deployment**

* Automate deployment using CI/CD pipelines (GitHub Actions, Jenkins, GitLab CI/CD).
* Deploy with orchestration tools like Kubernetes, Docker Swarm, or AWS Elastic Beanstalk.
* Perform load testing (JMeter) before the official production launch.
* Implement **Blue-Green Deployment** to ensure zero downtime during the release.

**Phase 4: Post-Deployment – Monitoring and Maintenance**

**Step 11: Continuous Monitoring**

* Use monitoring tools such as Prometheus, Grafana, or Datadog for real-time insights.
* Set up log management with the ELK Stack (Elasticsearch, Logstash, Kibana).

**Step 12: Bug Fixes and Continuous Updates**

* Collect user feedback and resolve bugs as they arise.
* Release new features using techniques like feature flagging.
* Maintain robust rollback procedures to revert changes if needed.