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.  
  
✅Quick Setup – No need to manually install Java or other dependencies.  
✅ Easy Cleanup – Just remove the container when needed.  
✅ Portability – Works the same across different OS environments.  
✅ No System Pollution – Does not install extra packages on your machine.  
✅ Easy Upgrades – Just pull the latest Jenkins image.

### Alternative Answer:

I prefer installing Jenkins on a Linux server directly for more control over configuration and plugins.  
  
✅ Full Control – Customize configurations and environment.  
✅ System Integration – Can integrate with system services like systemd.  
✅ Persistent Data – No need to manage Docker volumes.

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

### 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.  
 Create 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 with package managers (e.g., npm init, pip install).  
 Set up a code editor (VS Code, WebStorm) and frameworks (React, Express, etc.).

#### Step 3: Writing Code

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

#### Step 4: Version Control

Create a Git repository and push code.  
  
git init  
git add .  
git commit -m "Initial commit"  
git push origin main

### Alternative Answer:

Step 4: Version Control with Docker Compose  
Create a Docker Compose setup to manage frontend, backend, and database:  
  
docker-compose up

### 2. Testing Phase: Ensuring Quality

#### Step 5: Unit Testing

Write unit tests for individual components 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 with Postman, Newman, or Supertest.  
 Automate API tests using Cypress or Selenium.

#### Step 7: UI/UX Testing

 Check cross-browser compatibility (Chrome, Firefox, Edge).  
 Perform mobile responsiveness testing.

#### Step 8: Security Testing

 Run vulnerability scans with OWASP ZAP or Burp Suite.  
 Implement SSL, CORS, and authentication checks.

### Alternative Answer:

Step 8: Security Testing with Snyk  
 Use Snyk to detect vulnerabilities in dependencies.  
 Implement Content Security Policy (CSP) headers.

### 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).  
QA team runs manual and automated tests.

#### Step 10: Deploy to Production

 Use CI/CD Pipelines (GitHub Actions, Jenkins, GitLab CI/CD).  
 Deploy using Kubernetes, Docker Swarm, or AWS Elastic Beanstalk.  
 Run load testing (JMeter) before final launch.  
 Deploy with zero downtime (Blue-Green Deployment).

### Alternative Answer:

Step 10: Deploy to Production with Canary Deployment  
 Deploy to a small percentage of users first.  
 Gradually roll out the update while monitoring errors.

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

#### Step 11: Monitoring

 Use Prometheus, Grafana, Datadog for app monitoring.  
 Set up logging with ELK Stack (Elasticsearch, Logstash, Kibana).

#### Step 12: Bug Fixes & Updates

 Gather user feedback and fix issues.  
 Deploy new features via feature flagging.  
 Maintain rollback plans in case of failure.

### Alternative Answer:

Step 12: Bug Fixes & Updates with Feature Toggles  
 Use LaunchDarkly or Unleash to control feature rollouts.  
 Rollback instantly by toggling off a feature.

## Final Note:

The approach depends on project size, team expertise, and infrastructure. Docker, CI/CD, and automation ensure faster, reliable delivery — but traditional setups offer more control when needed.