**7 Stages of DevOps - Lecture Notes**

**Introduction**

DevOps is a software development approach that bridges the gap between development and operations teams, ensuring faster and more reliable software delivery. It consists of **7 key stages**, each involving various tools and best practices for automation and efficiency.

**1️. Plan**

The first stage involves planning and defining the objectives of the project.

**Key Activities:**

* Gathering requirements.
* Defining goals and roadmaps.
* Choosing the right DevOps strategy.

**Common Tools:**

* **Jira, Trello** - Agile project management.
* **Confluence** - Documentation.
* **GitHub Projects, GitLab Issues** - Tracking tasks.

**Example:**

A software development team uses **Jira** to manage sprint planning and backlog items, ensuring clear objectives and task tracking.

**2️. Develop**

The development phase includes writing, reviewing, and maintaining code.

**Key Activities:**

* Writing and testing code.
* Code versioning and collaboration.
* Using feature branching.

**Common Tools:**

* **Git, GitHub, GitLab, Bitbucket** - Version control.
* **Visual Studio Code, IntelliJ IDEA** - IDEs for coding.
* **SonarQube** - Code quality analysis.

**Example:**

Developers use **GitHub** for version control, ensuring all team members can collaborate efficiently.

**3️. Build**

Code is compiled and transformed into executable artifacts.

**Key Activities:**

* Automating builds.
* Managing dependencies.
* Running pre-build tests.

**Common Tools:**

* **Maven, Gradle** - Build automation for Java projects.
* **Webpack, NPM** - Frontend build tools.
* **Jenkins, GitHub Actions** - Continuous integration (CI).

**Example:**

A Java project uses **Maven** to compile code and **Jenkins** to automate build processes.

**4️. Test**

Automated testing ensures code quality and reliability.

**Key Activities:**

* Unit, integration, and functional testing.
* Security and performance testing.
* Automated test execution.

**Common Tools:**

* **JUnit, Selenium** - Automated testing frameworks.
* **Postman** - API testing.
* **JMeter** - Performance testing.

**Example:**

A web application team uses **Selenium** for UI testing and **JUnit** for unit tests.

**5️. Release**

Code is prepared for deployment in different environments.

**Key Activities:**

* Versioning and tagging.
* Approving and merging stable code.
* Release management.

**Common Tools:**

* **GitHub Actions, Jenkins** - Automated release pipelines.
* **Helm** - Kubernetes package management.
* **ArgoCD** - GitOps-based deployment.

**Example:**

A DevOps team automates releases using **Jenkins**, which pushes tested code to a production branch.

**6️. Deploy**

The software is deployed to production or staging environments.

**Key Activities:**

* Automating deployments.
* Rolling updates and rollback strategies.
* Infrastructure provisioning.

**Common Tools:**

* **Docker, Kubernetes** - Containerization and orchestration.
* **Ansible, Terraform** - Infrastructure as Code (IaC).
* **AWS CodeDeploy, Azure DevOps** - Cloud deployments.

**Example:**

A cloud-based application is deployed using **Kubernetes** with rolling updates.

**7️. Operate & Monitor**

Ensuring system stability, performance, and security.

**Key Activities:**

* Monitoring application performance.
* Logging and alerting.
* Incident management and root cause analysis.

**Common Tools:**

* **Prometheus, Grafana** - Monitoring and visualization.
* **ELK Stack (Elasticsearch, Logstash, Kibana)** - Log management.
* **PagerDuty** - Incident response.

**Example:**

A SaaS company uses **Prometheus and Grafana** to monitor application performance and set up alerts for server issues.

The **7 Stages of DevOps** provide a structured framework for efficient software delivery. By integrating automation and the right tools, DevOps accelerates development cycles and enhances operational efficiency. Organizations such as **Netflix, Amazon, and Google** leverage DevOps to achieve high reliability and scalability.By mastering these stages, IT teams can improve software quality, deployment speed, and system resilience.

### **STAGES UNDER CONTINUOUS INTEGRATION (CI), CONTINUOUS DEPLOYMENT (CD), AND CONTINUOUS TESTING (CT)**

**1️. Continuous Integration (CI) Stages**

These stages focus on automating code integration, builds, and early-stage testing.

1. **Code Commit & Version Control**
   * Developers push code to a **Git repository** (e.g., GitHub, GitLab, Bitbucket).
   * Version control tools (e.g., Git) track changes and manage branches.
2. **Automated Build Process**
   * Tools like **Jenkins, GitHub Actions, or GitLab CI/CD** compile the code.
   * Ensures all dependencies are met, and code compiles without errors.
3. **Static Code Analysis & Linting**
   * Tools like **SonarQube, ESLint, or Pylint** check code quality, security vulnerabilities, and best practices.
4. **Unit Testing**
   * Automated unit tests using **JUnit, NUnit, PyTest, or Mocha** ensure code correctness at the function level.
   * Helps catch errors before merging into the main branch.

**2️. Continuous Testing (CT) Stages**

These stages ensure the application is thoroughly tested before deployment.

1. **Automated Functional Testing**
   * Tools: **Selenium, Cypress, TestNG**
   * Verifies if the application works as expected.
2. **Integration Testing**
   * Tools: **Postman, JUnit, TestContainers**
   * Checks if different modules of the system work together.
3. **Performance & Load Testing**
   * Tools: **JMeter, Gatling, Locust**
   * Simulates user traffic and analyzes response times.
4. **Security Testing**
   * Tools: **OWASP ZAP, SonarQube, Snyk**
   * Identifies security vulnerabilities before deployment.

**3️. Continuous Deployment (CD) Stages**

These stages ensure automated release and deployment of applications.

1. **Artifact Packaging & Repository Storage**
   * Tools: **Docker, Nexus, JFrog Artifactory**
   * The built application is stored for deployment.
2. **Infrastructure Provisioning & Configuration**
   * Tools: **Terraform, Ansible, AWS CloudFormation**
   * Ensures infrastructure is set up correctly for deployment.
3. **Containerization & Orchestration**
   * Tools: **Docker, Kubernetes, Helm, OpenShift**
   * Applications are deployed as containers with orchestration.
4. **Deployment Automation & Monitoring**
   * Tools: **ArgoCD, Spinnaker, AWS CodeDeploy**
   * Deploys applications to production and ensures reliability.

**CONCLUSION**

**1. Continuous Integration (CI)**

* CI automates the integration of code from multiple developers into a shared repository.
* It includes **automated builds, linting, static code analysis, and unit testing**.
* Tools: **Jenkins, GitHub Actions, GitLab CI/CD**.

**2. Continuous Testing (CT)**

* Ensures the application works correctly through different levels of automated testing.
* It includes **functional, integration, performance, and security testing**.
* Tools: **Selenium, JMeter, Postman, SonarQube**.

**3. Continuous Deployment (CD)**

* Automates the deployment process, making new versions of applications available quickly.
* It includes **artifact storage, infrastructure provisioning, containerization, and monitoring**.
* Tools: **Docker, Kubernetes, Terraform, ArgoCD**.