Summary on Continuous Deployment (CD)

Module Summary

1. Importance of CD

- CD ensures that code changes are delivered to production reliably and quickly. It helps teams deliver features faster, improves product quality, and reduces risks by automating and streamlining the process from code to deployment.

2. Differences Between Continuous Delivery and Continuous Deployment

- Continuous Delivery (CD): Code changes are automatically tested and prepared for release to production. However, manual approval is required before deployment.
- Continuous Deployment: Every change that passes automated tests is automatically deployed to production without human intervention.

3. CD Tools

- Jenkins: An open-source automation server widely used for building, testing, and deploying software.
- GitLab CI: Integrated within GitLab, providing CI/CD pipelines, including testing, code quality checks, and deployments.
- Spinnaker: Multi-cloud deployment tool enabling release management and orchestration for complex pipelines.

4. Deployment Strategies

- Blue/Green Deployment: Two identical environments are set up (Blue and Green). The traffic is switched between environments when releasing new versions, reducing downtime.
- Canary Deployment: A small portion of traffic is routed to the new release, with gradual rollout based on the success of the initial batch.
- Rolling Deployment: New versions are rolled out incrementally across servers, reducing downtime while ensuring continuous availability.

5. Automated Testing and Monitoring

- Automated tests (unit, integration, regression) ensure that code changes are reliable, while monitoring helps track performance, health, and errors in production to ensure smooth operations.

6.Deployment Pipelines

- A deployment pipeline automates the workflow of building, testing, and deploying applications. It typically involves stages like build, test, staging, and production.

7. Object-Oriented Programming (OOP)

OOP is a programming paradigm based on the concept of "objects," which contain data (attributes) and code (methods). It enables modular, reusable, and maintainable code through key principles:

- Encapsulation: Bundling of data and methods within objects.
- Inheritance: Deriving new classes from existing ones.
- Polymorphism: Methods that behave differently based on the object.
- Abstraction: Hiding complex implementation details from the user.

8. Deployment Automation and Orchestration

- Automation ensures code can be deployed with minimal human intervention, while orchestration coordinates and manages complex deployments across multiple environments or servers.

9. CD Best Practices and Optimization

- Use small, incremental changes.
- Ensure code is always in a deployable state.
- Automate tests and deployments.
- Implement monitoring and alerting.
- Optimize pipelines for speed and reliability by parallelizing tasks and reducing manual steps.

10. Release Management and Approvals

- Even with CD, some teams maintain manual gates for critical releases. A well-defined release management process ensures approvals, compliance checks, and feature management before deploying to production.