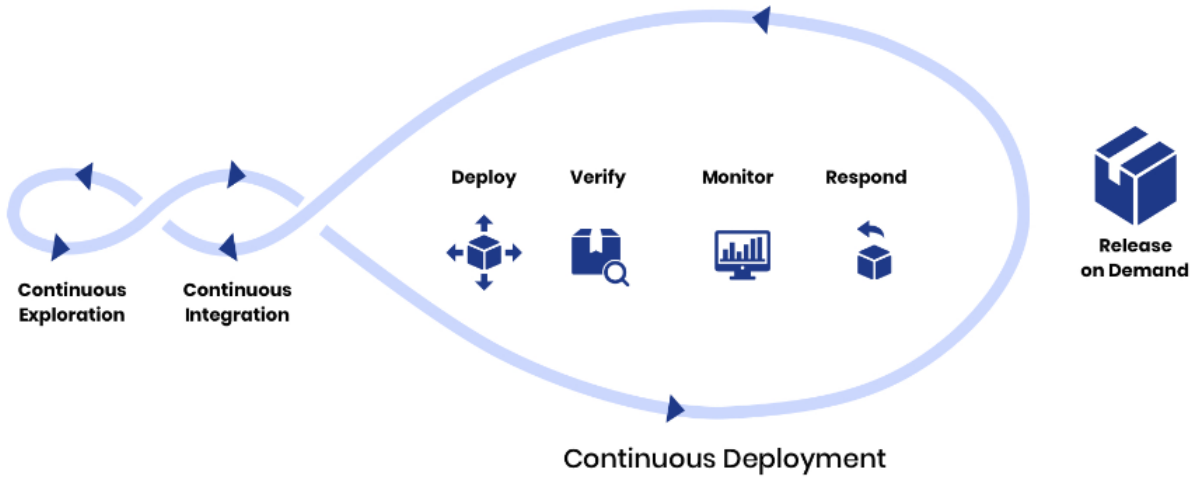
Deployment Pipelines

Lesson Plan

PW SKILLS

Deployment pipelines in Continuous Deployment (CD) are automated processes that ensure code changes are reliably tested, integrated, and deployed to production environments. Here's an overview of a typical CD pipeline, along with key stages:



1. Source Code Management (SCM)

- **Description:** The pipeline starts when developers push code changes to a version control system like Git.
- **Steps:** Code commits trigger the pipeline, initiating automated tests and builds.
- Tools: GitHub, GitLab, Bitbucket.

2. Continuous Integration (CI)

- **Description:** The CI stage automatically compiles the code, runs tests, and generates build artifacts.
- **Steps:**
  - **Code Compilation:** The pipeline compiles the application.
  - **Automated Testing:** Unit, integration, and end-to-end tests are run to validate the code.
  - **Build Artifacts:** Successful builds are packaged (e.g., into Docker images).
- **Tools:** Jenkins, GitLab CI, CircleCI, Travis CI.

3. Artifact Repository

- **Description:** The built artifacts are stored in a repository, making them available for deployment.
- **Steps:**
  - **Storing Artifacts:** Built images, binaries, or packages are saved.
  - **Versioning:** Artifacts are versioned to track changes over time.
- **Tools:** Nexus, Artifactory, Docker Hub.

4. Automated Deployment to Staging

- **Description:** The artifact is deployed to a staging environment for further testing.
- **Steps:**
  - **Deploy:** The pipeline automatically deploys the artifact to a staging server.
  - **Smoke Testing:** Basic tests are run to ensure the deployment was successful.
- **Tools:** Kubernetes, Docker, Ansible.

5. Acceptance Testing

- **Description:** The application undergoes acceptance tests in the staging environment to ensure it meets business requirements.
- **Steps:**
  - **User Acceptance Testing (UAT):** Automated or manual tests to verify functionality.
  - **Performance Testing:** Ensure the application performs under expected loads.
- **Tools:** Selenium, JMeter, LoadRunner.

6. Approval Gate (Optional)

- **Description:** Some pipelines include a manual approval step before deploying to production.
- **Steps:**
  - **Manual Approval:** A human reviews the staging results and approves or rejects the deployment.
- **Tools:** Jira, ServiceNow.

7. Automated Deployment to Production

- **Description:** If all tests pass and approval is granted, the artifact is automatically deployed to the production environment.
- **Steps:**
  - **Rolling Deployment:** Gradually replaces old versions with new ones to minimize downtime.
  - **Canary Deployment:** Deploys to a small subset of users first to ensure stability.
  - **Blue-Green Deployment:** Keeps two environments (blue and green) and switches traffic to the new version.
- **Tools:** Kubernetes, Terraform, AWS CodeDeploy.

8. Monitoring and Logging

- **Description:** After deployment, monitoring tools track the application's performance and health.
- **Steps:**
  - **Real-Time Monitoring:** Observing metrics like uptime, response times, and errors.
  - **Logging:** Collecting and analyzing logs for any issues.
- **Tools:** Prometheus, Grafana, ELK Stack.

9. Rollback Mechanism

- **Description:** If issues are detected post-deployment, the pipeline can automatically or manually roll back to the previous stable version.
- **Steps:**
  - **Trigger Rollback:** Reverts to the previous version if errors are detected.
  - **Post-Mortem Analysis:** Reviews what went wrong to prevent future occurrences.
- **Tools:** Git, Kubernetes, Terraform.

10. Feedback Loop

- **Description:** Feedback from monitoring is used to improve the pipeline, fix issues, and optimize future deployments.
- **Steps:**
  - **Continuous Improvement:** Teams analyze feedback and make adjustments to the pipeline.
  - **Issue Tracking:** Problems are logged and resolved in subsequent sprints.
- **Tools:** Jira, GitHub Issues.