**CI & CD PROCESS**

**Continous integration:**

Continuous Integration refers to the practice of automatically integrating code changes from multiple contributors into a shared repository frequently (often multiple times a day). The key components of the CI process are:

**Steps in Continuous Integration:**

1. **Developers Write Code:**
   * Developers work on new features, bug fixes, or enhancements in isolated branches.
2. **Push Code to Version Control:**
   * Once a developer finishes a piece of work, they push their changes to the version control system (e.g., Git). This might be a feature branch or a bugfix branch.
3. **Automatic Build and Test:**
   * A CI server (e.g., Jenkins, GitLab CI, Circle CI, Travis CI) monitors the version control system for changes.
   * When a new change is detected (e.g., a new commit or pull request), the CI system automatically triggers a build process.
   * The build process includes:
     + Compiling the code.
     + Running automated tests (unit tests, static code analysis, code quality, automations, reports & deployments) to ensure that the new code does not break existing functionality.
     + Code linting and analysis to ensure code quality.
4. **Feedback:**
   * The CI server provides feedback to the developers. This feedback may include:
     + Build success or failure status.
     + Test results (pass/fail).
     + Code quality issues.
   * If any errors or failures occur, the developers are notified, and they can fix the issues immediately. This helps avoid issues being pushed down the line, where they might become more expensive to fix.
5. **Merge into the Main Branch:**
   * Once the build passes and tests are successful, the changes are merged into the main branch (often called main or master).

**Benefits of CI:**

* Faster Bug Detection: Errors are detected early, making them easier and cheaper to fix.
* Consistent Builds: Every commit is tested in a clean environment, ensuring consistent builds.
* Higher Code Quality: Continuous code quality checks (linting, static analysis) improve the maintainability of the codebase.

**Key terms**

* Unit testing is nothing but testing code for a block or specific functionality.
* Static code analysis is to verify format, all indentation is right, syntax is right, make sure u did not declare any unnecessary variables.

* Code quality/ vulnerability testing is done to check for vulnerability, that can be misused by hackers
* Automation testing/functional testing /end-to-end testing, in this you verify your application in end-to-end manner such that one functionality does not affect other functionality.
* Reports - reports for how many units test passed, what is code quality etc.
* Deployment stage- deploying your application on a platform so that your customer can access your application.

**Continuous delivery:**

Continuous Delivery refers to the practice of keeping your codebase in a state where it is always ready for deployment to production. While CI automates code integration, CD automates the delivery process, making sure the software can be deployed at any time.

**Steps in Continuous Delivery:**

1. **Build and Test:**
   * After the code passes CI (including unit tests), the next step is to deploy it to a staging environment for further testing.
   * This includes additional tests, such as:
     + User Acceptance Testing (UAT).
     + End-to-End Testing.
     + Performance Testing.
2. **Deployment to Staging Environment:**
   * The code is automatically deployed to a staging environment that closely mimics the production environment.
   * It allows teams to test the application under realistic conditions.
3. **Approval Process (optional**):
   * In some cases, before deploying to production, a manual approval step is introduced. This is common in industries with strict compliance requirements.
   * The QA team or product managers might review the build, ensuring everything is working as expected.
4. **Release Candidate:**
   * If all tests pass successfully, the code is considered a "release candidate" and can be deployed at any time.

**Benefits of Continuous Delivery:**

* Quick Rollbacks: Since deployments are automated, rolling back changes to a previous version is also automated.
* Faster Time to Market: New features, bug fixes, and improvements are delivered quickly to end users.
* Reduced Risk: Smaller, more frequent releases reduce the risk of large-scale failures.

**Continous deployement:**

Continuous Deployment extends Continuous Delivery by fully automating the deployment process. In CD, every change that passes automated testing is deployed directly to the production environment, without any manual intervention.

**Steps in Continuous Deployment:**

1. **Automatic Testing**:
   * Just like in Continuous Delivery, after the code passes CI (including unit and integration tests), it proceeds through automated quality gates, including functional and performance tests.
2. **Deployment to Production**:
   * If the code passes all the necessary tests, it is automatically deployed to the production environment. This deployment happens without human intervention, ensuring that the code is always live for end users.
3. **Monitoring and Rollbacks**:
   * Once deployed, the system should be monitored for any issues (e.g., using logging and monitoring tools like Prometheus or Grafana).
   * If an issue arises, automated rollback mechanisms should be in place to revert to a stable version.

**Benefits of Continuous Deployment:**

* **Instant Delivery**: Every change that passes testing is immediately available to users, reducing the time between code creation and user access.
* **Highly Efficient**: Automation removes manual intervention, speeding up the entire development and release process.
* **High Product Quality**: Since only changes that pass all tests reach production, the quality of the product remains high.

**CI & CD INTERVIEW QUESTIONS:**

**What tools are commonly used for CI/CD?**

**Answer:**

* CI Tools: Jenkins, GitLab CI, Travis CI, CircleCI, GitHub Actions, Bamboo.
* Version Control: Git, GitHub, GitLab, Bitbucket.
* Build Tools: Maven, Gradle, Ant, npm.
* Deployment Tools: Kubernetes, Docker, Ansible, Helm, Terraform.
* Monitoring Tools: Prometheus, Grafana, ELK Stack.

**What is the difference between Continuous Integration and Continuous Delivery?**

**Answer:**

* **Continuous Integration (CI)** focuses on automatically integrating code changes into a shared repository, triggering automated tests to detect integration issues.
* **Continuous Delivery (CD)** is the practice of keeping your codebase deployable at any point. It ensures that new features, bug fixes, and improvements can be released into production with minimal manual intervention. Continuous Deployment, a form of CD, automatically deploys every change that passes tests.

**What is a build pipeline in CI/CD?**

**Answer:**

A build pipeline is a set of automated processes that allow developers to compile, test, and deploy code. It typically involves stages like:

* **Build**: Compiling the code and generating executable artifacts.
* **Test**: Running automated unit, integration, and other tests.
* **Deploy**: Deploying the code to a staging or production environment.

**What is a “Rolling Deployment” in CI/CD?**

**Answer:**

A **Rolling Deployment** is a deployment strategy where new application versions are gradually rolled out to a subset of servers, reducing the risk of affecting all users. This allows the system to stay available throughout the process, while any issues can be detected early on.

**What is a canary release?**

**Answer:**

A **canary release** is a deployment strategy where a new version of an application is rolled out to a small subset of users first (canary group) to monitor performance and catch issues before it reaches the full user base. If successful, the release is gradually rolled out to all users.

**What is the concept of "Blue/Green Deployment"?**

**Answer:**

**Blue/Green Deployment** is a strategy that reduces downtime and risk by running two identical production environments, referred to as the **blue** (current) and **green** (new) environments. New code is deployed to the green environment, and once it's verified, traffic is switched from blue to green. If issues arise, the traffic can be switched back to the blue environment.

**What are the benefits of CI/CD?**

**Answer:**

* **Faster Development Cycle**: Automating testing and deployment speeds up development.
* **Early Bug Detection**: Frequent integrations and automated tests catch issues early in the process.
* **Improved Code Quality**: Consistent code testing and delivery ensure the codebase remains high quality.
* **Faster Time to Market**: Continuous Delivery ensures that updates and features can be pushed to production more frequently.
* **Reduced Risk**: Smaller, incremental updates are easier to test, monitor, and roll back if needed.