**Name - Nabanita Sarkar**

**Emp ID – 815944**

**Java DevOps/CICD Notes**

**CI/CD pipeline:**

A CI/CD pipeline automates the process of software delivery. It builds code, runs tests, and helps you to safely deploy a new version of the software. CI/CD pipeline reduces manual errors, provides feedback to developers, and allows fast product iterations.

CI/CD pipeline introduces automation and continuous monitoring throughout the lifecycle of a software product. It involves from the integration and testing phase to delivery and deployment. These connected practices are referred as CI/CD pipeline.

**Continuous Integration, Continuous Delivery, and Continuous Deployment:**

* Continuous integration is a software development method where members of the team can integrate their work at least once a day. In this method, every integration is checked by an automated build to search the error.
* Continuous delivery is a software engineering method in which a team develops software products in a short cycle. It ensures that software can be easily released at any time.
* Continuous deployment is a software engineering process in which product functionalities are delivered using automatic deployment. It helps testers to validate whether the codebase changes are correct, and it is stable or not.

**Stages of a CI/CD pipeline:**

A CI/CD pipeline is a runnable specification of the steps that any developer should perform to deliver a new version of any software. Failure in each and every stage triggers a notification via email, Slack, or other communication platforms. It enables responsible developers to know about the important issues.

Here are the important Stages of CI/CD pipeline:

**Source Stage**

In the source stage, CI/CD pipeline is triggered by a code repository. Any change in the program triggers a notification to the CI/CD tool that runs an equivalent pipeline. Other common triggers include user-initiated workflows, automated schedules, and the results of other pipelines.

**Build Stage**

This is the second stage of the CI/CD Pipeline in which you merge the source code and its dependencies. It is done mainly to build a runnable instance of software that you can potentially ship to the end-user.Programs that are written in languages like C++, Java, C, or Go language should be compiled. On the other hand, JavaScript, Python, and Ruby programs can work without the build stage.Failure to pass the build stage means there is a fundamental project misconfiguration, so it is better that you address such issue immediately.

**Test Stage**

Test Stage includes the execution of automated tests to validate the correctness of code and the behaviour of the software. This stage prevents easily reproducible bugs from reaching the clients. It is the responsibility of developers to write automated tests.

**Deploy Stage**

This is the last stage where your product goes live. Once the build has successfully passed through all the required test scenarios, it is ready to deploy to live server.

**Example of CI/CD Pipeline**

Here is example of CI/CD pipeline:

* **Source Code Control:** Host code on GitHub as a private repository. This will help you to integrate your application with major services and software.
* **Continuous integration:** Use continuous integration and delivery platform CircleCI and commit every code. When the changes notify, this tool will pull the code available in GitHub and process to build and run the test.
* **Deploy code to UAT:**Configure CircleCI to deploy your code to AWS UAT server.
* **Deploy to production:**You have to reuse continuous integration steps for deploying code to UAT.

**CI/CD pipeline Best Practices**

Here is a CI/CD pipeline best practices:

* Write up the current development process therefore, you can know the procedures that require to change and one that can be easily automated.
* Start off with a small proof of project before going ahead and complete whole development process at once.
* Set up a pipeline with more than one stage in which fast fundamental tests run first.
* Start each workflow from the same, clean, and isolated environment.
* Run open source tools that cover everything from code style to security scanning.
* Setup a better code hub to continuously check the quality of your code by running the standard set of tests against every branch.
* Peer code review each pull request to solve a problem in a collaborative manner.
* You have to define success metrics before you start the transition to CD automation. This will help you to consistently analyze your software, developing progress help refining where needed.

**Advantages of CI/CD pipelines**

Here are the pros/ benefits of CI/CD Pipeline:

* Builds and testing can be easily performed manually.
* It can improve the consistency and quality of code.
* Improves flexibility and has the ability to ship new functionalities.
* CI/CD pipeline can streamline communication.
* It can automate the process of software delivery.
* Helps you to achieve faster customer feedback.
* CI/CD pipeline helps you to increase your product visibility.
* It enables you to remove manual errors.
* Reduces costs and labour.
* CI/CD pipelines can make the software development lifecycle faster.
* It has automated pipeline deployment.
* A CD pipeline gives a rapid feedback loop starting from developer to client.
* Improves communications between organization employees.
* It enables developers to know which changes in the build can turn to the brokerage and to avoid them in the future.
* The automated tests, along with few manual test runs, help to fix any issues that may arise.

**Important CI/CD tools**

Here are the important CI/CD tools:

**Jenkins**

Jenkins is an open-source Continuous Integration server that helps to achieve the Continuous Integration process (and not only) in an automated fashion. Jenkins is free and is entirely written in Java. Jenkins is a widely used application around the world that has around 300k installations and growing day by day.

**Features:**

* Jenkin will build and test code many times during the day.
* Automated build and test process, saving timing, and reducing defects.
* The code is deployed after every successful build and test.
* The development cycle is fast.

**Bamboo**

Bamboo is a continuous integration build server that performs – automatic build, test, and releases in a single place. It works seamlessly with JIRA software and Bitbucket.

**Features:**

* Run parallel batch tests
* Setting up Bamboo is pretty simple
* Per-environment permissions feature allows developers and QA to deploy to their environments
* Built-in Git branching and workflows. It automatically merges the branches.

**Kubernates:**

Kubernetes in an open source container management tool hosted by Cloud Native Computing Foundation (CNCF). This is also known as the enhanced version of Borg which was developed at Google to manage both long running processes and batch jobs, which was earlier handled by separate systems.

Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters. It is capable of creating container centric infrastructure.

**Features of Kubernetes:**

Following are some of the important features of Kubernetes.

* Continues development, integration and deployment
* Containerized infrastructure
* Application-centric management
* Auto-scalable infrastructure
* Environment consistency across development testing and production
* Loosely coupled infrastructure, where each component can act as a separate unit
* Higher density of resource utilization
* Predictable infrastructure which is going to be created