Name – Pavan A V

USN – 4NI19IS118

SECTION – 'A'

DEVOPS

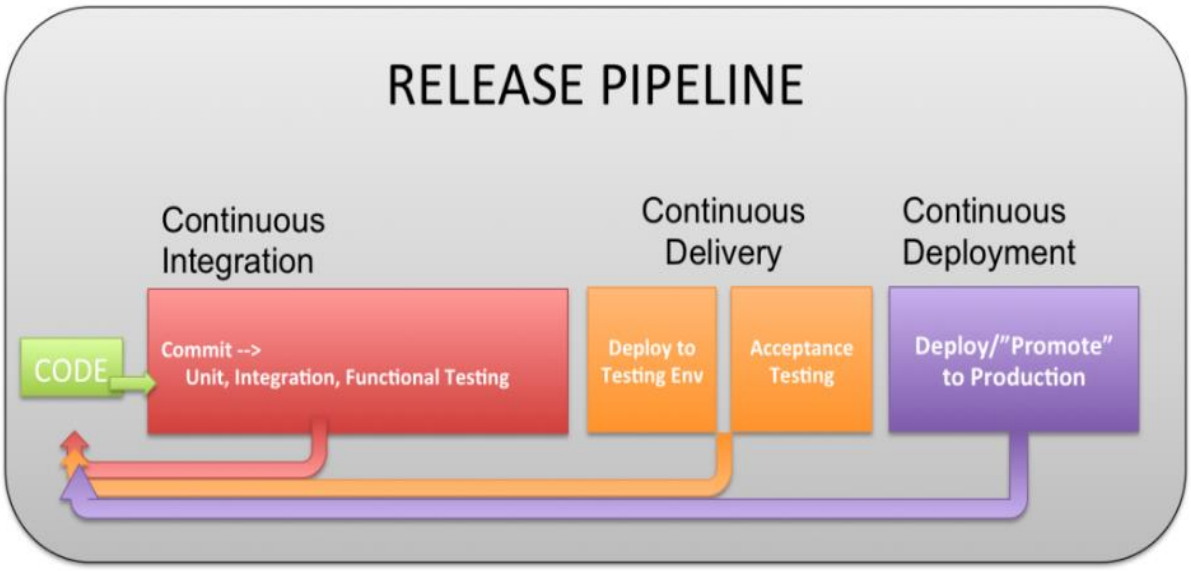
ASSIGNMENT-1

Continuous Integration and Continuous Deployment (CI/CD)?

In the 5G world, communication service providers (CSP) are accelerating the launch of new innovative services in ever-increasing numbers. For their networks to keep pace with service innovation, traditional waterfall upgrade processes with frequent manual interventions are insufficient. Faster time-to-market requirements demand the adoption of **DelOps** (delivery and operations) and **CI/CD** (continuous integration, continuous delivery). Together, they speed up the rollout of new services while reducing the risk to highly complex telecom operational networks.

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.



This allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method.

Difference between CI and CD

Continuous integration (CI) is practice that involves developers making small changes and checks to their code. Due to the scale of requirements and the number of steps involved, this process is automated to ensure that teams can build, test, and package their applications in a reliable and repeatable way. CI helps streamline code changes, thereby increasing time for developers to make changes and contribute to improved software.

Continous Delivery (CD) is the automated delivery of completed code to environments like testing and development. CD provides an automated and consistent way for code to be delivered to these environments.

Continous Deployment is the next step of continuous delivery. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

How CI/CD relate to DevOps

DevOps is a set of practices and tools designed to increase an organization's ability to deliver applications and services faster than traditional software development processes. The increased speed of DevOps helps an organization serve its customers more successfully and be more competitive in the market. In a DevOps environment, successful organizations "bake security in" to all phases of the development life cycle, a practice called DevSecOps.

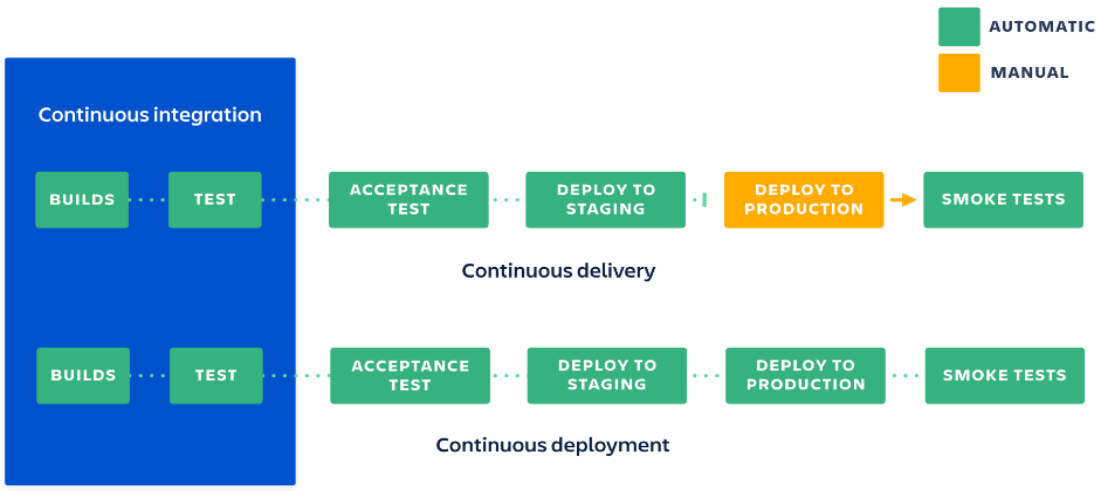
The CI/CD pipeline is part of the broader DevOps/DevSecOps framework. In order to successfully implement and run a CI/CD pipeline, organizations need tools to prevent points of friction that slow down integration and delivery. Teams require an integrated toolchain of technologies to facilitate collaborative and unimpeded development efforts.

Stages of CI/CD Pipeline

The continuous integration/continuous delivery (CI/CD) pipeline is an agile DevOps workflow focused on a frequent and reliable software delivery process. The methodology is iterative, rather than linear, which allows DevOps teams to write code, integrate it, run tests, deliver releases and deploy changes to the software collaboratively and in real-time.

A key characteristic of the CI/CD pipeline is the use of automation to ensure code quality. As the software changes progress through the pipeline, test automation is used to identify dependencies and other issues earlier, push code changes to different environments and deliver applications to production environments

The ability to automate various phases of the CI/CD pipeline helps development teams improve quality, work faster and improve other DevOps metrics.



Developers practicing continuous integration merge their changes back to the main branch as often as possible. The developer's changes are validated by creating a build and running automated tests against the build. By doing so, you avoid integration challenges that can happen when waiting for release day to merge changes into the release branch.

Continuous delivery is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage. This means that on top of automated testing, you have an automated release process and you can deploy your application any time by clicking a button.

Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.

CI/CD pipeline phases

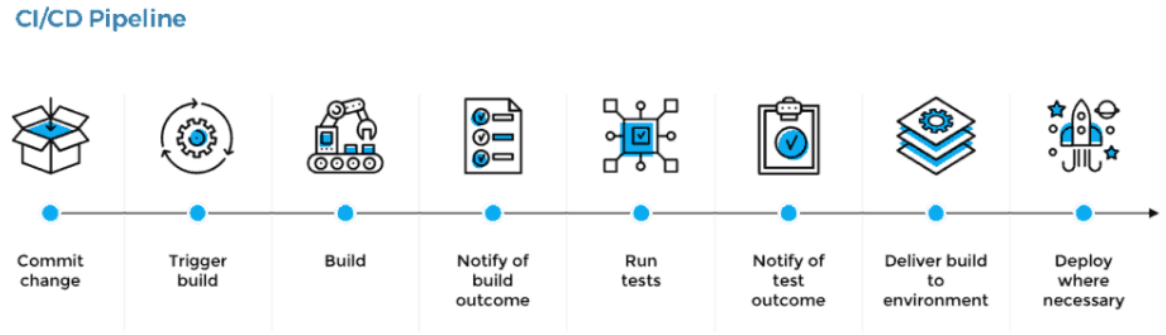
In a nutshell, the CI/CD is an automated pipeline for the software delivery process. A CI/CD pipeline is one of the best modern practices that has evolved to enable a DevOps team deliver quality software in a sustainable manner.The CI/CD pipeline has become imperative if an organization desires to ship bug-free code at high-velocity. In addition to eliminating manual errors due to automation, it provides a standardized feedback loop to the DevOps that produces faster iterations.

The 7 essential stages of a CI/CD pipeline

- The trigger.
- Code checkout.
- Compile the code.
- Run unit tests.
- Package the code.
- Run acceptance tests.
- Delivery or Deployment.

    - **Build:** This phase is part of the continuous integration process and involves the creation and compiling of code. Teams build off of source code collaboratively and integrate new code while quickly determining any issues or conflicts.
    - **Test:** At this stage, teams test the code. Automated tests happen in both continuous delivery and deployment. These tests could include integration tests, unit tests, and regression tests.
    - **Deliver:** Here, an approved codebase is sent to a production environment. This stage is automated in continuous deployment and is only automated in continuous delivery after developer approval.
    - **Deploy:** Lastly, the changes are deployed and the final product moves into production. In continuous delivery, products or code are sent to repositories and then moved into

production or deployment by human approval. In continuous deployment, this step is automated.



The trigger

The best pipelines are triggered automatically when new code is committed to the repository.You can either configure your CI/CD tool to *poll* for changes to a Git repository, or you can set up a Webhook to notify your CI/CD tool whenever a developer makes a *push*. The main reason for doing this is to make running the pipeline automatic, and friction-free.

Code checkout

In this first stage, the CI server will check out the code from the source code repository, such as GitHub or Bitbucket. The CI/CD tool usually receives information from a poll, or a webhook, which says which specific commit triggered the pipeline.The pipeline then checks out the source code at a given commit point, and starts the process.

Compile the code

If you're developing in a compiled language like Java, the first thing you'll probably need to do is compile your program. This means that your CI tool needs to have access to whatever build tools you need to compile your app. For example, if it's Java, you'll use something like Maven or Gradle.

Run unit tests

The next key element of your CI/CD pipeline is unit testing. This is the stage where you configure your CI/CD tool to execute the tests that are in your codebase. You might use Maven or Gradle to do this in Java, or Jest in JavaScript. The aim at this point is not only to verify that all the unit tests pass, but that the tests are being maintained and enhanced as the code base grows.

Package the code

Once all of the tests are passing, you can now move on to packaging the code. Exactly how you package your application depends on your programming language and target environment. If you're using Java, you might build a JAR file. If you're using Docker containers, you might build a Docker image.

Run acceptance tests

Now comes the time to perform *acceptance testing* on your application. Acceptance tests are a way of ensuring that your software does what it is meant to do, and that it meets the original requirements.

Implementation of CI/CD pipeline stages

Plan

Propose a feature that needs to be built, or outline an issue that warrants change.

Code

Turn use case suggestions and flowcharts into code, peer-review code changes, and obtain design feedback.

Test

Verify code changes through testing, preferably automated testing. At this point, unit testing will usually suffice.

Repository

Push code to a shared repository like Github that uses version control software to keep track of changes made. Some consider this as the first phase of the CI/CD pipeline.

Set up an Integration Testing Service

For example, Travis CI to continuously run automated tests, such as regression or integration tests, on software hosted on services like Github and BitBucket.

Set up a Service To Test Code Quality

Better Code Hub can help continuously check code for quality in CI/CD pipeline. This will enable the software development team to spend less time fixing bugs. Better Code Hub uses 10 guidelines to gauge quality and maintainability in order to future proof the code.

Build

This might overlap with compilation and containerization. Assuming the code is written in a programming language like Java, it'll need to be compiled before execution. Therefore, from the version control repository, it'll go to the build phase where it is compiled. Some pipelines prefer to use Kubernetes for their containerization.

Testing Phase

Smoke testing might be done at this stage so that a badly broken build can be identified and rejected, before further time is wasted installing and testing it. Push the container (Docker) image created to a Docker hub: This makes it easy to share your containers in different platforms or environments or even go back to an earlier version.

Deploy the App, Optionally To a Cloud-Based Platform

The cloud is where most organizations are deploying their applications. Heroku is an example of a relatively cheap cloud platform. Others might prefer Microsoft Azure or Amazon Web Service (AWS).

Advantages of CI/CD pipelines

- Improves flexibility and has the ability to ship new functionalities.
- CI/CD pipeline can streamline communication.
- Helps you to achieve faster customer feedback.
- It enables you to remove manual errors.
- Reduces costs and labour.
- CI/CD pipelines can make the software development lifecycle faster.
- A CD pipeline gives a rapid feedback loop starting from developer to client.
- Improves communications between organization employees.