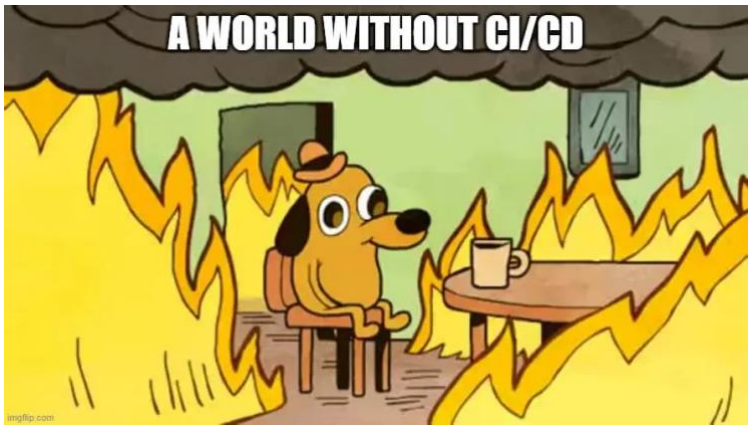
INTRODUCTION TO CICD

Say you make a website and you want users to be able to access it so you copied the files to your production server but did not test the code on the server at all, or forgot to edit access. What will happen? you and your users will have to face downtime and your users will be 😔.

Now what if you and your coworker want to edit some files at the same time. What will happen? your server might take half the edits that you made and half of the files your coworker made and then the code will not run properly and your users will face downtime == 😭



The Solution?

“ENTER GITHUB.” We need a repository where you can store the code in one place where everyone can check the code and the repository itself will only take one person's changes at a time and if there’s a problem then the repository itself will report an ERROR.

CI: Continous Integration (Aap code update karte raho)
CD: Continous Development (Hum Develop karte rahenge)

CI/CD pipeline is an automated DevOps workflow that streamlines the delivery process. It is the use of automation to help ensure code quality. As the software changes progress through the pipeline, automated tests identify dependencies and other issues earlier, push code changes to different environments and deliver applications to production environments.

This gives teams a single repository for storing work and automation for storing work and automation tools to combine and test the code to help ensure it works consistently.

Benefits of CICD:

1. Quicker time to market
2. Higher software quality
3. Reduced Risk
4. Collaboration
5. Decreased costs
6. Continous feedback loops

What is continuous integration?

Developers commit their code to central repositories managed by version control systems (VCSs). CI allows developers to work independently, creating their own coding "branch" to implement small changes.
As the developer works, they can take snapshots of the source code, typically within a versioning tool like Git. The developer is then free to work on new features; if a problem comes up, Git can quickly revert the codebase to its previous state.
The work of individuals is then pushed into an automated system that uses scripts to build and test the code changes. After the automated build stage, a CI server compiles the source code changes into the main branch code or "trunk" of the shared source code repository.

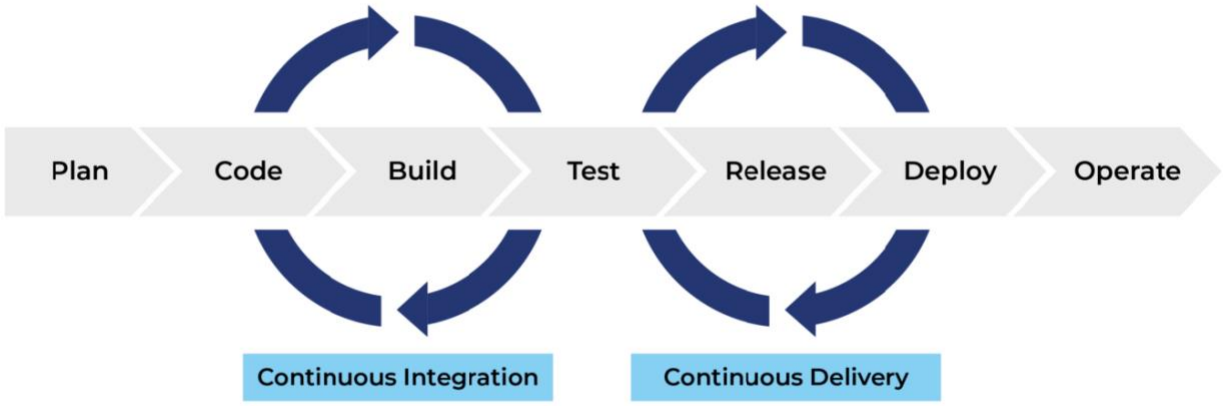
What is continuous delivery?

Now, CD puts the validated code changes made in continuous integration into select environments or code repositories, such as GitHub. Here, the operations team can deploy them to a live production environment.

The software and APIs are tested, and errors are resolved through an automated process. In the final step of the CD process, the DevOps team receives a notification about the latest build, and they manually send it to the deployment stage.

What is continuous deployment?

Continuous deployment automatically releases code changes to end-users after passing a series of predefined tests, such as integration tests that test code in a copycat environment to help ensure code integrity.



Example of these tools:

Jenkins-Jenkins automates tasks by executing jobs triggered by events like new commits, branches, and pull requests. It can also distribute builds across multiple machines to reduce build times.

Gitlab-It includes tools for version control, project planning, testing, and more. GitLab is built around Git, a distributed version control system.

Open Source: Travis CI, Azure, Amazon Web Services etc.