Introduction: What does CI/CD stands for ?

CI and CD stand for Continuous Integration and Continuous Delivery/Continuous Deployment.

- **Continuous Integration**: Is the practice of merging all developer's working copies into a central repository, after which automated builds and tests are run. CI handles everything has to do with coding.

- **Continuous Delivery**: Is an engineering paradigm in which teams produce and release value in a short cycles. CD enhances the practices of Continuous Integration and Continuous Delivery.

- **Continuous Deployment**: A software engineering approach in which value is delivered through automated deployments. Continuous Deployment system deploys every change that passes all stages of production pipeline automatically without human intervention.

Continuous Integration + Continuous Deployment = Continuous Delivery

Why we need CI/CD ?

Our current pain points:

i.   **Code Conflit**

Solution with CI: We need a continuous integration server that can monitor the main repository and run the tests automatically for every new commits pushed. Developers are alerted as soon as they break the build and can work on fixing it before they move to another task.

ii.   **Unit test takes a lot of time**

CI server can run hundreds of tests in a matter of seconds.

iii.   **Human error and Inconsistency**

Eliminate manual processes throughout the development and deployment cycles with automated build, test, artifact management, and deployment tools.

Benefits of CI/CD

Technical Language	Translation	Value
Catch Compile Errors After Merge	Less developer time on issues from new developer code	Reduce Cost
Catch Unit Test Failures	Less bugs in production and less time in testing	Avoid Cost
Automate Infrastructure Creation	Less human error, Faster deployments	Avoid Cost
Deploy to Production Without Manual Checks	Less time to market	Increase Revenue
Faster and More Frequent Production Deployments	New value-generating features released more quickly	Increase Revenue
Automated Rollback Triggered by Job Failure	Quick undo to return production to working state	Protect Revenue
Detect Security Vulnerabilities	Prevent embarrassing or costly security holes	Avoid Cost