Continuous Integration / Continuous Deployment (CI / CD) Pipelines

CI / CD falls under DevOps and combines the practices of continuous integration and continuous delivery. CI / CD automates much or all manual human intervention traditionally needed to get new code from a commit to production, encompassing the build, test and deploy as well as infrastructure provisioning.

With a CI/CD pipeline, development teams can make changes to code that are then automatically tested and pushed out for delivery and deployment. Get CI/CD right and downtime is minimized and code releases happen faster.

This automation means teams can release new features and fixes faster and more frequently, enhancing the product's responsiveness to user needs. By continuously integrating and deploying, errors are detected sooner, reducing downtime and improving software quality.

CI /CD Concepts :-

Continuous Integration (CI) –

Continuous Integration is process of integrating all your code changes into the main branch of a shared source code repository early and often, automatically testing each change when you commit or merge them, and automatically kicking off a build. With continuous integration, errors and security issues can be identified and fixed more easily, and much earlier in the development process.

Continuous Delivery (CD)-

Continuous delivery is a software development practice that works with CI to automate the infrastructure provisioning and application release process. CD takes over during the final stages after code has been tested in CI process to ensure it's packaged with everything it needs to deploy to any environment at any time. CD can cover everything from provisioning the infrastructure to deploying the application to the testing or production environment.With CD, the software is built so that it can be deployed to production at any time. Then you can trigger the deployments manually or move to continuous deployment, where deployments are automated as well.

Continuous Deployment –

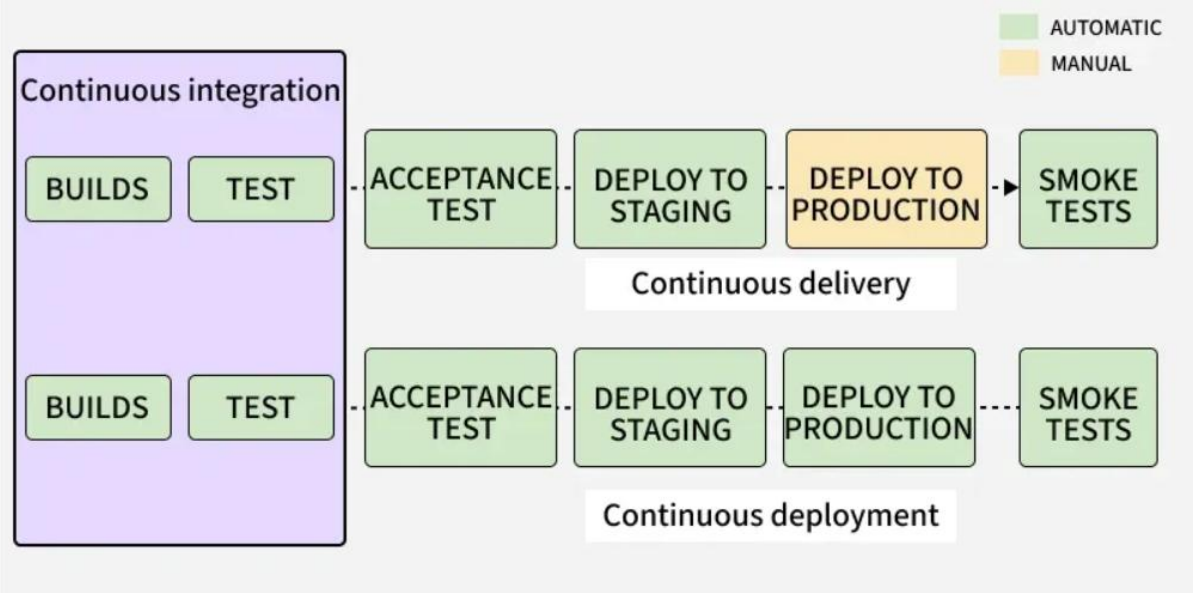
With continuous deployment, DevOps teams set the criteria for code releases ahead of time and when those criteria are met and validated, the code is deployed into the production environment. This allows organizations to be more nimble and get new features into the hands of users faster.

CI / CD Fundamentals :-

1. **Single source repository** - Source code management (SCM) that houses all necessary files and scripts to create builds is critical. This includes source code, database structure, libraries, properties files, and version control. It should also contain test scripts and scripts to build applications.
2. **Frequent check-ins to main branch** - Integrate code in your trunk, mainline or master branch i.e., trunk-based development early and often. Avoid sub-branches and work with the main branch only.
3. **Automated builds** - Scripts should include everything you need to build from a single command. This includes web server files, database scripts, and application software. The CI processes should automatically package and compile the code into a usable application.
4. **Self-Testing Builds -** Builds must be **self-verifying**. Use automated testing. If tests fail → build fails.Prevents bad code from entering production.
5. **Frequent Iterations** - Commit **small, incremental changes** instead of big releases. Benefits are Fewer bugs per iteration, Easier rollbacks if something breaks, Faster feedback loop. Supports Agile + DevOps principles.
6. **Stable Testing Environments** - Test in **production-like environments** (staging). Never test directly on live production. Use cloned environments with the same OS, DB, dependencies, configs. Ensures bugs caught in staging = bugs that would occur in production.
7. **Maximum Visibility** - Everyone in the team should see: Latest builds, Test results, Deployment status. Version control ensures clarity. This Improves collaboration & transparency.
8. **Predictable Deployments Anytime** - Deployments should be **routine, low-risk, and automated**. CI/CD process ensures: Confidence in quality (via testing + automation), Frequent small releases instead of risky big releases, Easy rollbacks if issues occur.

CI/CD Pipeline Stages :-

1. **Commit** → Developer pushes code to repo.
2. **Build** → Code is compiled, dependencies installed.
3. **Test** → Automated tests run (unit, integration, etc.).
4. **Package** → Artifact is created (JAR, Docker image, etc.).
5. **Deploy** → Application deployed to staging/prod.
6. **Monitor** → Performance/errors tracked, feedback loop created.



CI / CD Workflow