What is CI/CD?

Your All-in-One Learning Portal: GeeksforGeeks is a comprehensive educational platform that empowers learners across domains-spanning computer science and programming, school education, upskilling, commerce, software tools, competitive exams, and more.

By GeeksforGeeks

7 min. read · View original

CI/CD is the practice of automating the integration of code changes from multiple developers into a single codebase. It is a software development practice where the developers commit their work frequently to the central code repository (Github or Stash). Then there are automated tools that build the newly committed code and do a code review, etc as required upon integration.

The key goals of Continuous Integration are to find and address bugs quicker, make the process of integrating code across a team of developers

easier, improve software quality, and reduce the time it takes to release new feature updates. Some popular CI tools are **Jenkins, TeamCity, and Bamboo**.

Continuous Integration

There could be scenarios when developers in a team work in isolation for an extended period and only merge their changes to the master branch once their work is completed. This not only makes the merging of code very difficult, prone to conflicts, and time-consuming but also results in bugs accumulating for a long time which are only identified in later stages of development. These factors make it harder to deliver updates to customers quickly.

With **Continuous Integration**, developers frequently commit to a shared common repository using a version control system such as Git. A continuous integration pipeline can automatically run builds, store the artifacts, run unit tests, and even conduct code reviews using tools like Sonar. We can configure the CI pipeline to be triggered every time there is a commit/merge in the codebase.

Continuous Delivery

Continuous delivery helps developers test their code in a production-similar environment, hence preventing any last-moment or post-production

surprises. These tests may include UI testing, load testing, integration testing, etc. It helps developers discover and resolve bugs preemptively.

By automating the software release process, CD contributes to low-risk releases, lower costs, better software quality, improved productivity levels, and most importantly, it helps us deliver updates to customers faster and more frequently. If Continuous Delivery is implemented properly, we will always have a deployment-ready code that has passed through a standardized test process.

CDorContinuous Delivery is carried out after Continuous Integration to make sure that we can release new changes to our customers quickly in an error-free way. This includes running integration and regression tests in the staging area (similar to the production environment) so that the final release is not broken in production. It ensures automation of the release process so that we have a release-ready product at all times and we can deploy our application at any point in time.

Continuous Delivery automates the entire software release process. The final decision to deploy to a live production environment can be triggered by the developer/project lead as required. Some

popular CD tools are AWS CodeDeploy, Jenkins, and GitLab.

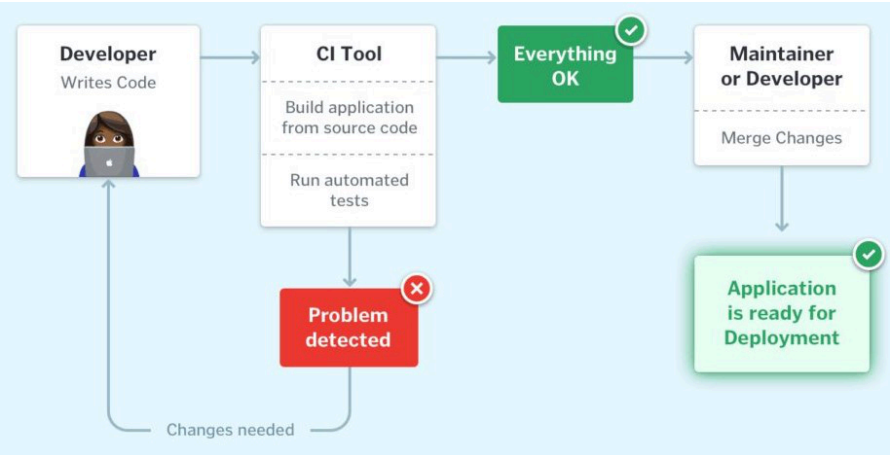
Continuous Deployment

The final stage of CI and CD will be continuous deployment. It's an extension of continuous delivery, which automate the proper code to the code repository, continuous deployment will automate the related app for production purpose because there is not having any manual gate at the stage of the pipeline before production, continuous deployment relies on high automation.

in simple language, it is a change of application that goes through the cloud which is carried by the developer and it will live within a few minutes of writing pass with the automated testing.

CI Workflow

Below is a pictorial representation of a **CI pipeline**– the workflow from developers checking in their code to its automated build, test, and final notification of the build status.



Once the developer commits their code to a version control system like Git, it triggers the CI pipeline which fetches the changes and runs automated build and unit tests. Based on the status of the step, the server then notifies the concerned developer whether the integration of the new code to the existing code base was a success or a failure.

This helps in finding and addressing the bugs much more quickly, makes the team more productive by freeing the developers from manual tasks, and helps teams deliver updates to their customers more frequently. It has been found that integrating the entire development cycle can reduce the developer's time involved by ~25 – 30%.

CI and CD Workflow

The below image describes how Continuous Integration combined with Continuous Delivery helps quicken the software delivery process with lower risks and improved quality.

CI / CD workflow

We have seen how Continuous Integration automates the process of building, testing, and packaging the source code as soon as it is committed to the code repository by the developers. Once the CI step is completed, the

code is deployed to the staging environment where it undergoes further automated testing (like Acceptance testing, Regression testing, etc.). Finally, it is deployed to the production environment for the final release of the product.

If the deployment to production is a manual step. In that case, the process is called Continuous Delivery whereas if the deployment to the production environment is automated, it is referred to as Continuous Deployment.

Why is CI/CD Important?

CI/CD tool helps in the automaton of software delivery process from writing code step to deploying it in live production environment. When developers push code, CI/CD tools like Jenkins, Github Actions etc automatically build the application, run tests to catch the bugs in the early stage and deploy the code to servers in a consistent and authentic way. This automation enables faster feedback, reduces manual errors and speeds up the release cycle. As a result, teams can deliver high quality software faster and more efficient way after this.

What are the Benefits of CI/CD?

The following are the benefits of CI/CD:

- **Improves Software Quality & Security:** Automated testing helps catch bugs early, making the software more reliable and secure.

- **Makes Code More Profitable:** By reducing errors, companies spend less time fixing issues and more time delivering value.
- **Speeds Up Product Releases:** CI/CD pipelines help roll out new features faster, keeping customers happy.
- **Reduces Developer Workload:** Automation takes care of repetitive tasks, allowing developers to focus on innovation.
- **Gives a Competitive Edge:** Faster updates help businesses stay ahead of their competitors.
- **Attracts Skilled Professionals:** A well-structured CI/CD process creates an efficient work environment that appeals to top talent.
- **Eliminates Bottlenecks:** By moving away from slow, step-by-step processes, teams can work more efficiently.

What are Some Common CI/CD Tools?

CI and CD tools can help team in the development, deployment, and testing, some of the tools are highly recommended for the integration part and some are for the development and management of the testing and related functionality.

Most of the famous tools for the CI and CD is Jenkins. It is open source and it will help to handle all types of work and design a simple CI server to complete the CD hub.

Apart from the Jenkins, many more sources are available for the proper way of managing CI and CD which are listed below:

- Concourse: It is an open-source tool to build the mechanics of CI and CD.
- GoCD: it's used for the modeling and visualization.
- Screwdriver is a building platform for CD.
- Spinnaker: it's a CD platform used to build a multi-cloud environment.

What is CI/CD Pipelines?

People often get confused between CI/CD and CI/CD Pipeline, but they are not same. So before starting discussion on CI/CD pipelines let's discuss the key difference CI/CD and CI/CD pipeline:

Continuous Integration and Continuous Delivery(CI/CD) refers to the set of processes where you can made multiple changes to the code simultaneously to improve software delivery while a CI/CD pipeline is a process that takes your code from development to production automatically and smoothly. While,

CI/CD pipeline is the combination of automated steps that helps software development teams to deliver code in quick, secure and efficient way. In simple words we can CI/CD pipeline is a process of taking your code from development to production environment automatically and smoothly. Let's try to understand it by a simple example:

The following are the CI/CD pipeline steps:

1. Source Stage

Riya is a developer who writes code using any IDEs like VSCode or Pycharm. After writing the complete code she pushes that code to shared repository(like Github,Gitlab). The Github stores everything like Code, test scripts, documentation and even build files.

2. Build Stage

Once the code is pushed,a CI tool like Github Actions or Jenkins is triggered automatically. The pipeline pulls the code, install all the dependencies, compiles the code and packages it.

3. Test Stage

After the build is ready, a series of automated tests are run:

- **Unit tests** to check small pieces of code
- **Integration tests** to check how components work together
- **Static code analysis** by using Synk or SonarQube for security issues.
- **User Acceptance Tests (UAT)** by using tools like Selenium to validate functionality.

If all tests pass, the pipeline continues.

If any tests fails, the pipeline will stop working and report the issue. The test reports are integrated into Github Actions or Jenkins.

4. Deployment Stage

If tests pass, the pipeline deploys the app:

- First, it is deployed to a staging server for final checks.
- After approval, it automatically goes to production server where users can use it.

For example:

- The app is packaged into a **Docker image** and stored in **Docker Hub**
- Then it's deployed using **Ansible**, which sets up the server and runs the app

With this CI/CD pipeline:

- Every time Riya pushes code, the app is automatically built, tested, and deployed.
- Teams catch bugs early, reduce manual work, and release features faster with more confidence.

What is CI/CD Security?

CI/CD security is all about **protecting software development pipelines** from hackers, data leaks, and security flaws. By adding security checks at every stage of development, teams can **catch problems early** and prevent attacks.

Common Security Risks in CI/CD Pipelines

- **Code & Dependencies:** Using outdated or unsafe libraries can introduce vulnerabilities.
- **Pipeline Access:** Weak passwords or excessive access can let attackers in.
- **Container & Artifact Security:** Running unverified images may lead to security breaches.
- **Infrastructure Security:** Poorly configured cloud settings can expose data or allow unauthorized access.

Conclusion

CI/CD has revolutionized the way software is developed, tested, and deployed. By automating key processes, teams can deliver updates faster, with fewer bugs and security risks. A well-implemented CI/CD pipeline ensures a smooth workflow, reduces manual effort, and keeps software in a ready-to-deploy state at all times. It also enhances security by catching vulnerabilities early in the development cycle. Businesses that embrace CI/CD gain a competitive advantage by improving software quality, reducing release cycles, and maintaining a seamless development experience.