Date: 07.07.2025

CI/CD

Contents

1. What is CI/CD? 2
2. Benefits of CI/CD 2
3. Key Concepts 3
   a. Continuous Integration (CI) 3
   b. Continuous Delivery (CD) 3
   c. Continuous Deployment (CDep) 3
4. Stages of CI/CD 4
   a. Source 4
   b. Build 4
   c. Test 4
   d. Deploy 4
5. Tools for CI/CD 5
   General Purpose CI/CD platforms 5
   Specialized Continuous Delivery (CD) Tools 6
   Built-in CI/CD Solutions from Hosting Platforms 6
6. Resources 7

1. What is CI/CD?

- CI/CD, or Continuous Integration/Continuous Delivery, is a software development practice that automates the process of building, testing, and deploying code.
- It is a software development approach that aims to improve the speed, efficiency, and reliability of software delivery.

CI/CD is made up of two main parts:

- Continuous Integration (CI): Developers merge code changes into a central repository as often as possible. This practice helps reduce testing costs and the number of bugs that get shipped to production.
- Continuous Delivery or Continuous Deployment (CD): The next step in the process, where code changes are tested and deployed fully automatically (with or without manual approval steps).

2. 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.

3. Key Concepts

a. Continuous Integration (CI)

- CI addresses the issues of traditional software development where code was only integrated at the end of a release, leading to bugs and delays.
- In CI, developers frequently merge changes into the master branch, and automated tests run immediately to catch issues early.
- A build server handles code integration, testing, and build generation.
- CI improves software quality and enables faster, more frequent releases.

b. Continuous Delivery (CD)

- CD automates the deployment process, reducing the risks and manual work of traditional deployments.
- Once code passes CI, it is packaged, tested in a staging environment, and can be deployed to production with minimal effort—often just a button press.
- CD ensures that software is always in a deployable state, supporting faster, safer releases.

c. Continuous Deployment (CDep)

- CDep automates the entire pipeline, including deployment to production.
- Once code passes all tests, it is automatically released to users without manual approval.
- This eliminates release schedules, speeds up feedback, and promotes a rapid, stress-free development cycle.
- It requires strong automation, testing, and monitoring systems.

4. Stages of CI/CD

a. Source

In the source phase, developers write code using frameworks (e.g., Java, .NET, PHP) and IDEs. Tools like linters, vulnerability scanners, and version control systems (e.g., Git) ensure code quality and manage changes.

b. Build

The build phase compiles code from the repository, resolves dependencies, and generates executable files. Tools may also package builds into environments like VMs or Docker containers and provide error logs and notifications.

c. Test

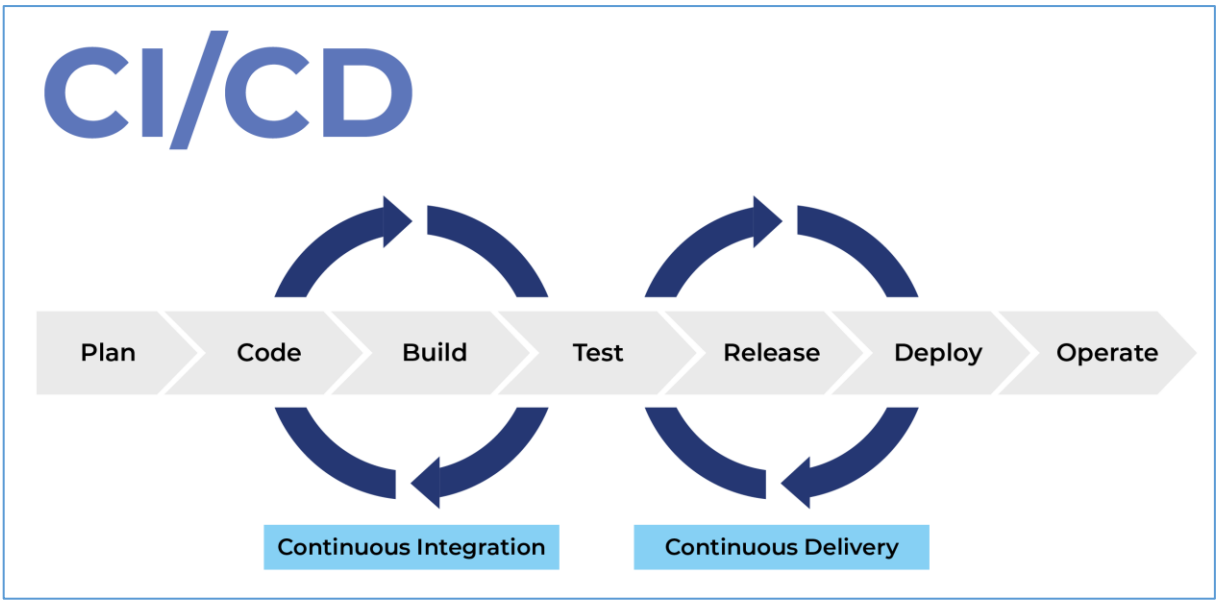
After building, code undergoes dynamic testing, including unit, regression, integration, and performance tests. Automated testing ensures accuracy and efficiency. Test failures are looped back to developers for fixes.

d. Deploy

Once testing is successful, builds are deployed using:

- **Continuous Delivery**: Manual approval before production deployment.
- **Continuous Deployment**: Fully automated deployment to all environments.

Deployment involves moving builds to targets (e.g., servers) and often integrates monitoring and alert systems for post-deployment issues.



5. Tools for CI/CD

General Purpose CI/CD platforms			
Tool	**Description**	**Pros**	**Cons**
Jenkins	Jenkins is an open-source automation server widely used in CI/CD pipelines. Highly customizable and supports many plugins, suitable for various environments.	- Highly customizable with wide plugin range - Integrates with many tools - Detailed reporting and analytics	- Requires technical expertise - Resource-intensive for large projects - No centralized dashboard
GitHub Actions	Powerful CI/CD tool integrated with GitHub repositories to automate workflows, tests, and deployments.	- Integrated with GitHub - Easy to use - Large ecosystem and good documentation	- Limited build minutes - Complex YAML syntax
GitLab CI	Open source CI tool integrated into GitLab, supports testing, building, deploying projects with security and audit features.	- Strong integration with GitLab - Supports multiple languages - Secure project data	- Requires GitLab hosting
CircleCI	CI tool supporting containers, delivery mechanisms, version control like GitHub. Can run complex pipelines with caching and Docker support.	- Supports containers and caching - Runs on cloud and on-premises	- Pricing can be a concern for some
Travis CI	Cloud-based CI/CD with automated testing and deployment supporting many languages and frameworks.	- Easy setup and use - Cloud-based with no infrastructure management	- Limited customization - Not suitable for large, complex projects
Bitbucket Pipelines	Integrated into Bitbucket, allows managing pipelines as code and deploying projects using CD tools.	- Easy Bitbucket integration - Pipeline definitions as code	- Less powerful for complex pipelines
Azure Pipelines	Cloud service to build, test, and ship code for many languages and platforms including VMs, containers, on-prem/cloud.	- Supports many languages - Targets multiple platforms	- Some learning curve
Bamboo	Automation server by Atlassian for CI/CD with automated merging and deployment support.	- Simple UI for CI/CD - Built-in deployment features	- Licensing costs
Codefresh	GitOps CI/CD tool for Kubernetes with enterprise features like unified UI, auditability, single sign-on.	- Kubernetes and GitOps focused - Enterprise-grade security and audit	- Medium-low adoption compared to giants

CloudBees	Enterprise Jenkins-based CI/CD platform adding security, scalability, governance for large orgs.	- Enterprise-grade security and governance - Pipeline automation and analytics	- Higher cost and complexity

Specialized Continuous Delivery (CD) Tools			
Tool	**Description**	**Pros**	**Cons**
Argo CD	Kubernetes-native CD tool for GitOps, stores config in Git, auto-applies to clusters, detects drift, supports blue/green and canary deployments.	- Kubernetes native - Drift detection and rollback - Supports progressive delivery	- Kubernetes focused, less general
AWS CodePipeline	Cloud-based CD service automating release steps with visualization and consistent release processes.	- AWS ecosystem integration - Automates complex release workflows	- Limited outside AWS cloud
Spinnaker	Open source CD platform for multi-cloud, pipeline builder, supports Kubernetes, integrates with monitoring tools like Prometheus, Datadog.	- Multi-cloud support - Pipeline reuse and automation	- Complex setup and maintenance

Built-in CI/CD Solutions from Hosting Platforms			
Tool	**Description**	**Pros**	**Cons**
Built-in CI/CD hosts (Vercel, Netlify)	Hosting providers with built-in CI/CD to deploy sites automatically on repo events.	- Very simple to set up - Direct integration with hosting	- Limited customization

6. Resources

What Is CI/CD? Complete 2025 Guide |
What is CI/CD? - GeeksforGeeks
What is CI/CD? Learn Continuous Integration/Continuous Deployment by Building a Project