**What is CI?**

* **CI (Continuous Integration)** means:  
  Every time a developer changes the code and pushes it to GitHub, the computer automatically:
  1. Downloads the code
  2. Builds it (makes sure it compiles or runs)
  3. Runs tests (checks if it works properly)

👉 If something is broken, it tells you immediately.  
This way, the **main code** always stays healthy.

**Continuous Integration (CI)** is the practice of frequently merging code changes into a shared main branch and automatically verifying each change with checks like build, unit tests, linting, and security scans.

* **Goal:** Catch problems early (at commit/PR time), keep the codebase always buildable.
* **Core loop:** *Commit/PR → Build → Test → Report status (pass/fail).*
* **Benefits:** Fewer merge conflicts, fast feedback, higher code quality, stable main branch.

**What is CD?**

* **CD** comes after CI. It’s about **getting the code to users**.

There are **two flavors** of CD:

1. **Continuous Delivery**
   * The code is always ready to release.
   * But someone (a human) must **press the button** to release it to real users.
2. **Continuous Deployment**
   * The code is not only ready, it is **automatically released** to real users.
   * No human approval needed.

👉 Delivery = “ready to ship, but waiting for your OK.”  
👉 Deployment = “shipped automatically.”

**CD** has two common expansions. Both start **after** CI has produced a good build.

1. **Continuous Delivery**
   * Every change that passes CI is **packaged and ready to release** (e.g., pushed to a staging environment or stored as a versioned artifact).
   * A **human approval** (or change window) typically triggers the final push to production.
2. **Continuous Deployment**
   * Every change that passes CI is **automatically released to production** with **no manual step**.
   * Requires strong automated tests, safe rollout strategies (feature flags, canaries), and rapid rollback.

Continuous integration (CI) is practice that involves developers making small changes and checks to their code. Due to the scale of requirements and the number of steps involved, this process is automated to ensure that teams can build, test, and package their applications in a reliable and repeatable way. CI helps streamline code changes, thereby increasing time for developers to make changes and contribute to improved software.

Continuous delivery (CD) is the automated delivery of completed code to environments like testing and development. CD provides an automated and consistent way for code to be delivered to these environments.

Continuous deployment is the next step of continuous delivery. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

In short, CI is a set of practices performed *as developers are writing* code, and CD is a set of practices performed *after* the code is completed.

A diagram of a process

AI-generated content may be incorrect.

**How does CI/CD relate to DevOps?**

In simple terms:

* DevOps is the CULTURE and PROCESS of teams working together to build and deliver software quickly and reliably.
* CI/CD is the TOOLSET and AUTOMATION that makes that culture possible. It's the engine that takes the theory of DevOps and makes it a reality.

You can't have a successful, fast-moving DevOps culture without automation like CI/CD. And CI/CD pipelines are built for the main purpose of enabling a DevOps way of working.

Think of it like a pizza restaurant:

* **DevOps is the entire philosophy** of the restaurant. It's the belief that the chefs (developers) and the waitstaff (operations team) should work together closely. They communicate, share the same goal (happy customers), and constantly try to improve the entire process from taking an order to delivering the pizza.
* **CI/CD is the automated pizza-making machine** they built to make that philosophy work.
  + **CI (Continuous Integration)** is the part where every time a chef adds a new topping (a code change), the machine automatically checks that it doesn't ruin the pizza (automated testing). Everyone's changes are integrated smoothly and tested constantly.
  + **CD (Continuous Delivery/Deployment)** is the part where the machine then automatically boxes up the perfect pizza and gets it ready for a server to deliver it. In some cases, the machine can even send the pizza out to the customer's table by itself (automatic deployment).

**What is CI? (Continuous Integration)**

**CI is like your team's super-organized writing and editing process.**

* **The Problem:** In the old days, one writer would finish their article and just throw it on a pile. Another writer would do the same. At the end of the week, someone would try to stitch all the articles together, and it would be a mess. The titles were different fonts, two articles were about the same thing, and someone spilled coffee on a page.
* **The CI Solution:** Now, you have a central rule: **"Every time a writer finishes a page, they immediately put it into a special copying machine."**
  + This machine **automatically** makes sure the page fits with the others (\*\*\*\*Integration\*\*).
  + It checks for spelling mistakes (**Automated Testing**).
  + It makes sure the pictures are the right size (**Code Quality Checks**).
  + If anything is wrong, the machine instantly tells the writer to fix it right away.

**In short: CI is the process of automatically testing and combining small pieces of code as they are finished, so you never have a big, messy pile.**

**What is CD? (Continuous Delivery/Deployment)**

**CD is what happens after the magazine is edited and ready to go. It's your automated printing and delivery system.**

There are two slightly different ideas here:

1. **Continuous Delivery:** The magazine is **always ready to be shipped**. The copying machine automatically bundles up the entire, perfect magazine and places it on a loading dock, ready for a human to just press a button to send it to the printing press. (You are **able** to release at any time).
2. **Continuous Deployment:** This goes one step further. The process is so reliable that you **automatically send** every finished magazine straight from the copying machine to the printing press and then to the reader's mailbox, without needing a person to press the button. (Every good change is **automatically released** to users).

**In short: CD is the automated process of packaging up your tested code and delivering it to your users quickly and safely.**

**Putting It All Together (The CI/CD Pipeline)**

This whole automated system—from the writer adding a page to the magazine arriving at a reader's door—is called a **CI/CD Pipeline**. It's one big, automatic assembly line for your software.

**Why is this so great?**

* **Faster Releases:** You can get new features to your readers every day instead of every month.
* **Fewer Errors:** The automatic checking machine catches spelling mistakes early, so the final magazine is high-quality.
* **Less Stress:** The process is automated and reliable, so no one is up all night trying to stitch a magazine together by hand.

**So, in simple terms:**

* **CI** is automatically **building and testing** your application constantly.
* **CD** is automatically **delivering and releasing** that application.

**🏏 Cricket Match Example**

* Players practice batting, bowling, and fielding.
* **CI** → After each practice session, the coach checks: “Is batting form correct? Are bowlers not throwing wide balls?”
* **CD** → Once the team is ready:
  + Delivery = Team is dressed and waiting in the pavilion.
  + Deployment = Team directly walks to the ground and starts the match.

**🔧 Tools used in CI/CD**

There are many tools that help with automation. Think of them as **robots** in the factory assembly line.

**1. Version Control (Git, GitHub, GitLab, Bitbucket)**

* **Think of this like Google Docs for code.**
* **It saves your code, remembers every change, and lets many people work together without losing progress.**

**2. CI/CD Tools (Jenkins, GitHub Actions, GitLab CI, CircleCI)**

* **These are like robots.**
* **Whenever you add or change code, the robot wakes up and checks:**
  + **Does the code build?**
  + **Do all tests pass?**
  + **Can this code be safely delivered to users?**

**3. Build Tools (npm, Maven, pip, etc.)**

* **These are like cooking tools.**
* **They prepare your raw code into something usable (like turning ingredients into a dish).**
* **Example:**
  + **For Python → pip installs your needed packages.**
  + **For Java → Maven/Gradle build your project.**

**4. Testing Tools (pytest, JUnit, Jest, etc.)**

* **These are like quality checkers in a factory.**
* **They test small parts of your program to see if it works correctly.**
* **Example: If you made a calculator, a test checks: *Does 2 + 2 really give 4?***

**5. Deployment Tools (Docker, Kubernetes, Ansible)**

* **These help you send your code to servers so real users can use it.**
* **Docker puts your app into a box (container) so it runs the same everywhere.**
* **Kubernetes helps you run many such boxes together.**
* **Ansible/Terraform help you set up machines automatically.**