What is CI/CD?

Updated May 11, 2022 • 5-minute read

Copy URL

Overview

CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment. CI/CD is a solution to the problems integrating new code can cause for development and operations teams (AKA "integration hell").

Specifically, CI/CD introduces ongoing automation and continuous monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment. Taken together, these connected practices are often referred to as a "CI/CD pipeline" and are supported by development and operations teams working together in an agile way with either a DevOps or site reliability engineering (SRE) approach.

Learn how automation supports CI/CD pipelines

What's the difference between CI and CD (and the other CD)?

The acronym CI/CD has a few different meanings. The "CI" in CI/CD always refers to continuous integration, which is an automation process for developers. Successful CI

means new code changes to an app are regularly built, tested, and merged to a shared repository. It's a solution to the problem of having too many branches of an app in development at once that might conflict with each other.

The "CD" in CI/CD refers to continuous delivery and/or continuous deployment, which are related concepts that sometimes get used interchangeably. Both are about automating further stages of the pipeline, but they're sometimes used separately to illustrate just how much automation is happening.

Continuous delivery usually means a developer's changes to an application are automatically bug tested and uploaded to a repository (like GitHub or a container registry), where they can then be deployed to a live production environment by the operations team. It's an answer to the problem of poor visibility and communication between dev and business teams. To that end, the purpose of continuous delivery is to ensure that it takes minimal effort to deploy new code.

Continuous deployment (the other possible "CD") can refer to automatically releasing a developer's changes from the repository to production, where it is usable by customers. It addresses the problem of overloading operations teams with manual processes that slow down app delivery. It builds on the benefits of continuous delivery by automating the next stage in the pipeline.

It's possible for CI/CD to specify just the connected practices of continuous integration and continuous delivery, or it can also mean all 3 connected practices of continuous integration, continuous delivery, and continuous deployment. To make it more complicated, sometimes "continuous delivery" is used in a way that encompasses the processes of continuous deployment as well.

In the end, it's probably not worth your time to get bogged down in these semantics—just remember that CI/CD is really a process, often visualized as a pipeline, that involves adding a high degree of ongoing automation and continuous monitoring to app development.

Case-by-case, what the terms refer to depends on how much automation has been built into the CI/CD pipeline. Many enterprises start by adding CI, and then work their way towards automating delivery and deployment down the road, for instance as part of cloud-native apps.

Our experts can help your organization develop the practices, tools, and culture needed to more efficiently modernize existing applications and to build new ones.
Get expert help on your cloud-native development journey

Continuous integration

In modern application development, the goal is to have multiple developers working simultaneously on different features of the same app. However, if an organization is set up to merge all branching source code together on one day (known as "merge day"), the resulting work can be tedious, manual, and time-intensive. That's because when a developer working in isolation makes a change to an application, there's a chance it will conflict with different changes being simultaneously made by other developers. This problem can be further compounded if each developer has customized their own local integrated development environment (IDE), rather than the team agreeing on one cloud-based IDE.

Continuous integration (CI) helps developers merge their code changes back to a shared branch, or "trunk," more frequently—sometimes even daily. Once a developer's changes to an application are merged, those changes are validated by automatically building the application and running different levels of automated testing, typically unit and integration tests, to ensure the changes haven't broken the app. This means testing everything from classes and function to the different modules that comprise the entire app. If automated testing discovers a conflict between new and existing code, CI makes it easier to fix those bugs quickly and often.

Let's get even more technical

Continuous delivery

Following the automation of builds and unit and integration testing in CI, continuous delivery automates the release of that validated code to a repository. So, in order to have an effective continuous delivery process, it's important that CI is already built

into your development pipeline. The goal of continuous delivery is to have a codebase that is always ready for deployment to a production environment.

In continuous delivery, every stage—from the merger of code changes to the delivery of production-ready builds—involves test automation and code release automation. At the end of that process, the operations team is able to deploy an app to production quickly and easily.

You can also automate these deployments

Continuous deployment

The final stage of a mature CI/CD pipeline is continuous deployment. As an extension of continuous delivery, which automates the release of a production-ready build to a code repository, continuous deployment automates releasing an app to production. Because there is no manual gate at the stage of the pipeline before production, continuous deployment relies heavily on well-designed test automation.

In practice, continuous deployment means that a developer's change to a cloud application could go live within minutes of writing it (assuming it passes automated testing). This makes it much easier to continuously receive and incorporate user feedback. Taken together, all of these connected CI/CD practices make deployment of an application less risky, whereby it's easier to release changes to apps in small pieces, rather than all at once. There's also a lot of upfront investment, though, since automated tests will need to be written to accommodate a variety of testing and release stages in the CI/CD pipeline.

What does this really look like?

What are some common CI/CD tools?

CI/CD tools can help a team automate their development, deployment, and testing. Some tools specifically handle the integration (CI) side, some manage development and deployment (CD), while others specialize in continuous testing or related functions.

One of the best known open source tools for CI/CD is the automation server Jenkins. Jenkins is designed to handle anything from a simple CI server to a complete CD hub.

Deploying Jenkins on Red Hat OpenShift

Tekton Pipelines is a CI/CD framework for Kubernetes platforms that provides a standard cloud-native CI/CD experience with containers.

Deploying Jenkins on Red Hat OpenShift

Beyond Jenkins and Tekton Pipelines, other open source CI/CD tools you may wish to investigate include:

- Spinnaker, a CD platform built for multicloud environments.
- GoCD, a CI/CD server with an emphasis on modeling and visualization.
- Concourse, "an open-source continuous thing-doer."
- Screwdriver, a build platform designed for CD.

Teams may also want to consider managed CI/CD tools, which are available from a variety of vendors. The major public cloud providers all offer CI/CD solutions, along with GitLab, CircleCI, Travis CI, Atlassian Bamboo, and many others.

Additionally, any tool that's foundational to DevOps is likely to be part of a CI/CD process. Tools for configuration automation (such as Ansible, Chef, and Puppet), container runtimes (such as Docker, rkt, and cri-o), and container orchestration (Kubernetes) aren't strictly CI/CD tools, but they'll show up in many CI/CD workflows.

Keep reading

ARTICLE

Who is a DevOps engineer?

A DevOps engineer has a unique combination of skills and expertise that enables collaboration, innovation, and cultural shifts within an organization.

Read more

ARTICLE

What is CI/CD?

CI/CD introduces ongoing automation and continuous monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment.

Read more

ARTICLE

What is DevSecOps?

If you want to take full advantage of the agility and responsiveness of DevOps, IT security must play a role in the full life cycle of your apps.

Read more

More about DevOps

Products | Related articles | Resources

Red Hat
Open Innovation Labs

An intensive, highly focused residency with Red Hat experts where you learn to use an agile

Red Hat
Consulting

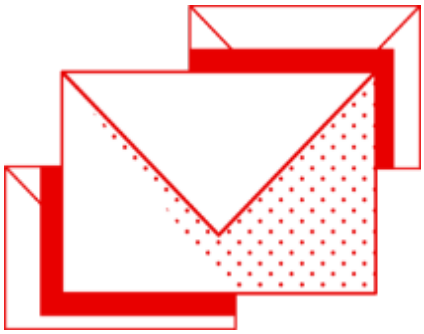
Engagements with our strategic advisers who take a big-picture view of your organization, analyze your challenges, and help you

methodology and open source tools to work on your enterprise's business problems.

Learn more

overcome them with comprehensive, cost-effective solutions.

Learn more



Get more content like this

Sign up for our free newsletter, Red Hat Shares.

Continue

ABOUT

We're the world's leading provider of enterprise open source solutions, using a community-powered approach to deliver high-performing Linux, cloud, container, and Kubernetes technologies. We help you standardize across environments, develop cloud-native applications, and integrate, automate, secure, and manage complex environments with award-winning support, training, and consulting services.

Company information

Jobs

Locations

Development model

Events

Newsroom

Blog

Cool Stuff Store

Diversity, equity, and inclusion

PRODUCTS

Red Hat Enterprise Linux

Red Hat OpenShift

Red Hat Ansible Automation Platform

Cloud services

See all products

TOOLS

My account

Console

Customer support

Partner resources

Developer resources

Training and certification

Learning community

Red Hat Ecosystem Catalog

Resource library

TRY, BUY, SELL

Product trial center

Red Hat Store

Red Hat Marketplace

Find a partner

Contact sales

Contact training

Contact consulting

COMMUNICATE

Contact us

Feedback

Social

Red Hat newsletter