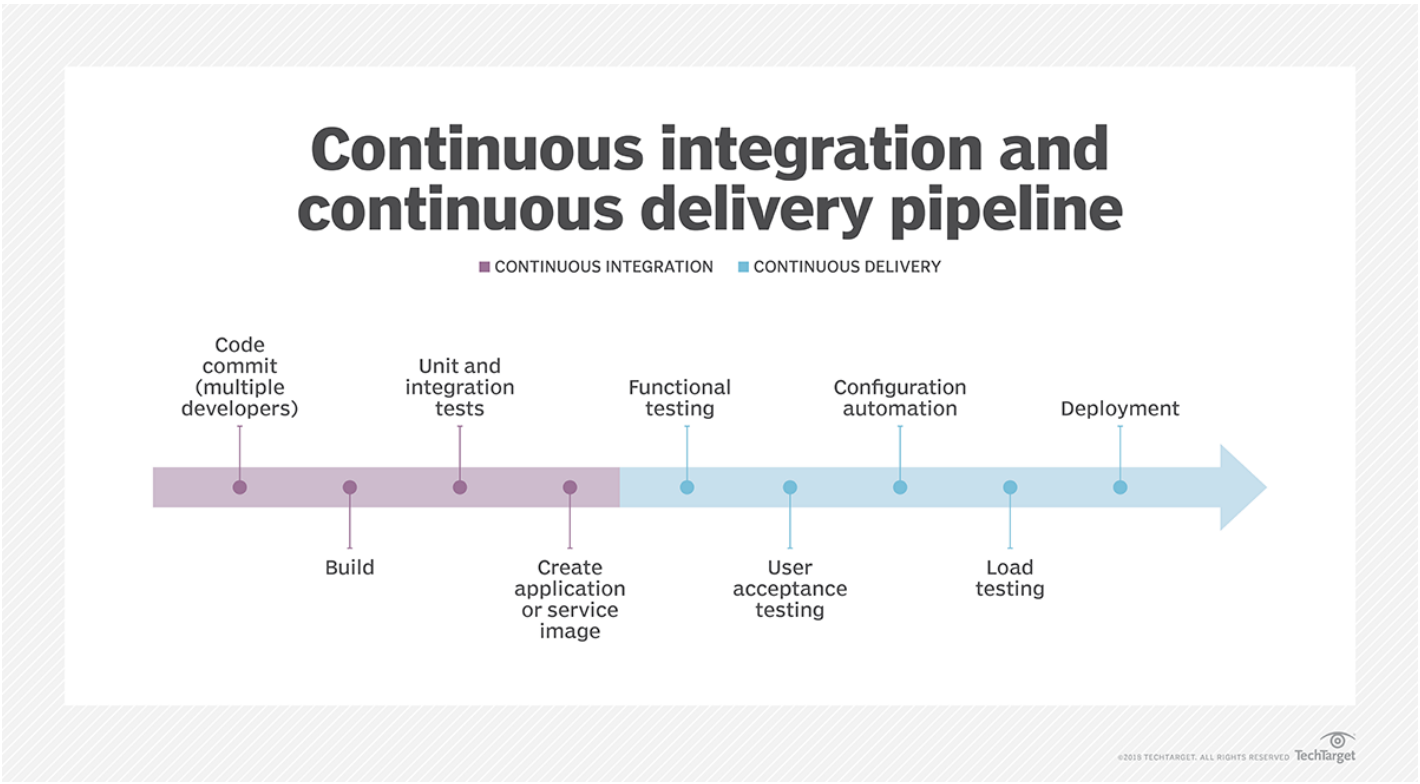
Jenkins

Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used to implement CI/CD workflows called pipelines.

CI/CD pipelines automate testing and reporting on isolated changes in a larger codebase in real time. They also facilitate the integration of disparate branches of the code into a main branch. Pipelines rapidly detect defects in a codebase, build the software, automate testing of builds, prepare the codebase for deployment and delivery, and ultimately deploy code to containers and virtual machines (VMs), as well as to bare-metal and cloud servers.

There are several commercial versions of Jenkins; however, this definition focuses on the upstream open source Jenkins project.



Jenkins and CI/CD

Jenkins uses the CI/CD pipeline methodology to streamline the work of DevOps teams. This pipeline can be broken down into two parts:

- **Continuous integration.** With the initial CI process, developers typically add new code on a regular basis to improve a software project. Continuous integration detects bugs and defects before integrating new code into an existing codebase.
- **Continuous delivery and deployment.** Continuous delivery comes after CI. It automates the building and packaging of code for deployment to test, staging and production environments. Finally, continuous deployment automates software delivery processes, deploying the code to its destination.

  With Jenkins, in both CI and CD, automation reduces the number of errors that occur because the correct steps and best practices are encoded into software. Jenkins describes a desired state, and the automation server ensures it happens.

  Jenkins' CI/CD capabilities benefit other software projects through integrations. For example, Jenkins integrates with Docker to build software as containers and with Kubernetes to orchestrate these Docker containers into apps. Jenkins streamlines development through continuous building, testing and deployment. This workflow makes the work of developers more efficient.

How Jenkins works

Jenkins runs as a server on a variety of platforms including Windows, macOS, Unix variants and especially Linux. **It requires a Java 8 VM or above**, and is run on the Oracle Java Runtime Environment

or OpenJDK. Usually, Jenkins runs as a Java servlet within a Jetty application server. It can run on other Java application servers such as Apache Tomcat.

Once Jenkins is installed on a server, all its projects share a common goal: to aid developers with a repeatable workflow that builds and tests new code, then deploys and releases new software into production. This workflow takes the form of a pipeline customized for each individual project. The pipeline is set up once and then used repeatedly.

A Jenkins pipeline automates various processes, including code testing, conducting a staged deployment to test how software features do in a production environment and deploying to a production environment.

The need for Jenkins becomes especially acute when deploying software to microservices architectures. Jenkins can integrate with Docker for containerization and Kubernetes for container orchestration. Since one of the goals of microservices is to frequently update applications and services, the ability to do so can't be limited by release bandwidth. Jenkins provides the automation needed for more and smaller services with faster update intervals.

Jenkins pipeline

Once an organization installs Jenkins, the DevOps team creates a pipeline as a series of steps the Jenkins server will take to perform the required tasks of the CI/CD process. There are two types of Jenkins pipelines: scripted and declarative. A scripted pipeline gives developers more control over the script so that new code can be injected into the pipeline at any time to update and change tasks. A declarative pipeline uses predefined constructs for simpler coding, but that also makes it more rigid than scripted pipelines.

The contents of both declarative and scripted pipelines are stored in plain text Jenkinsfiles. A Jenkinsfile is a text file that describes a pipeline in the form of code, explaining its purpose and how it's configured to suit a specific project's needs. Developers usually code these directly. A Jenkinsfile uses a Groovy-based syntax so that the Jenkins server can read it and execute its commands.