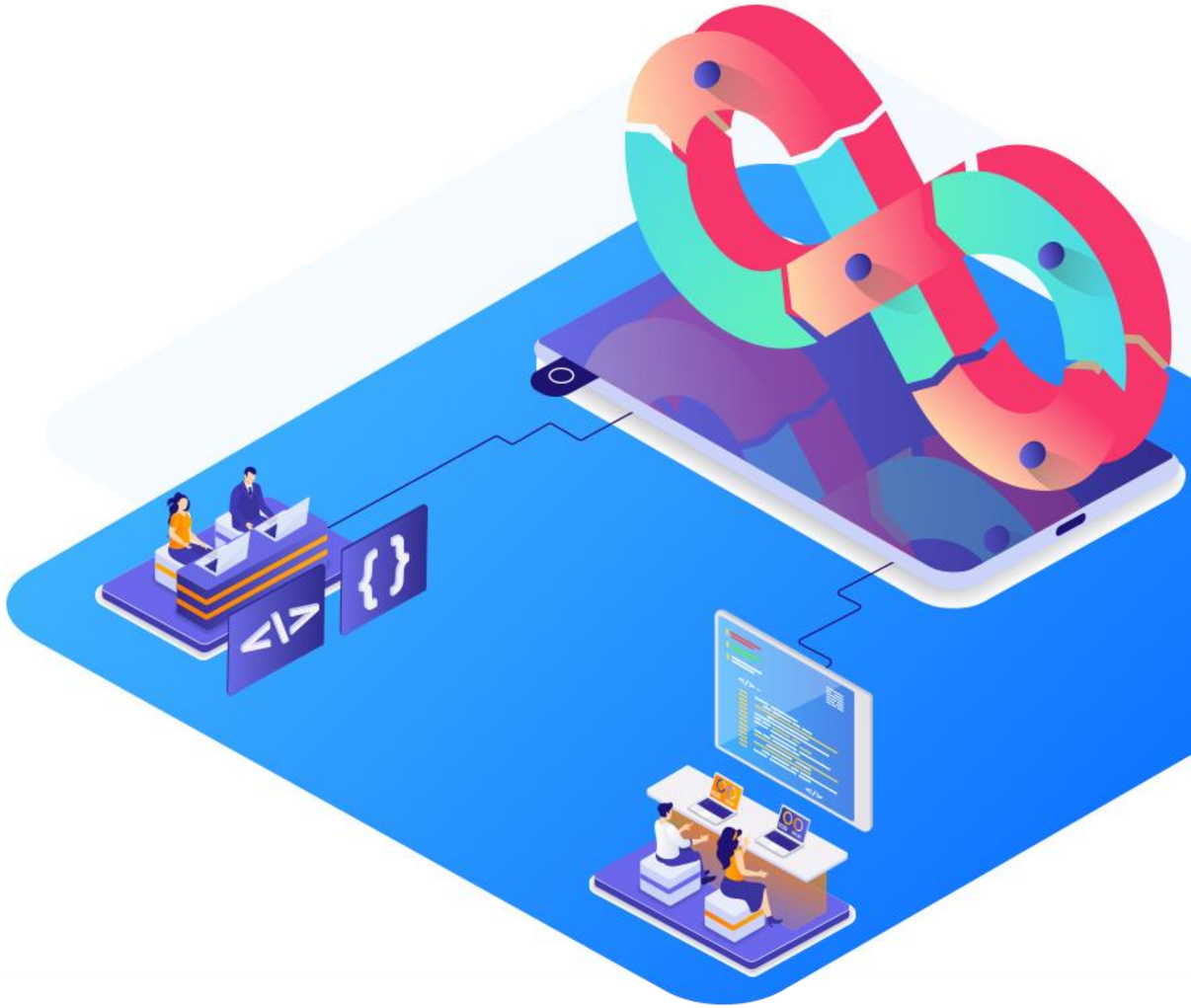
DevOps Foundations: Version Control and CI/CD with Jenkins



CI/CD Pipeline with Jenkins

Learning Objectives

By the end of this lesson, you will be able to:

- Outline the fundamental concepts of CI/CD to deliver high-quality software consistently, efficiently, and with reduced risk
- Identify the appropriate CI/CD tools for optimizing the software development process, ensuring compatibility, scalability, and efficiency
- Define the features and benefits of Jenkins for optimizing development process and achieving reliable software delivery
- Illustrate the Jenkins architecture and its support for various platforms for optimizing resource management in CI/CD pipelines

Learning Objectives

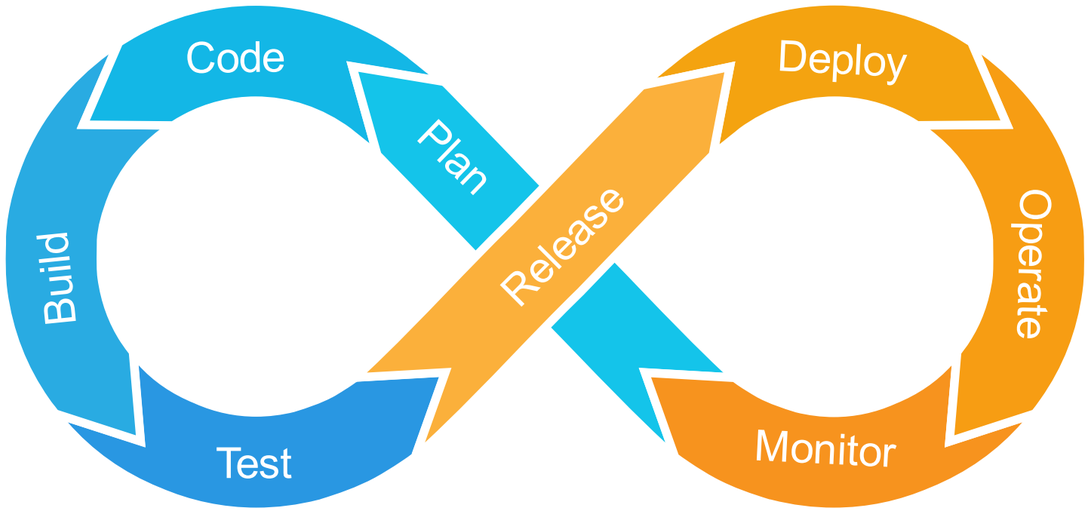
By the end of this lesson, you will be able to:

- Illustrate how to create new users in Jenkins to enhance security, enable customization, and promote collaboration among team members
- Explore the Jenkins UI to effectively utilize its features and administer user accounts and permissions

Getting Started with CI/CD Pipeline

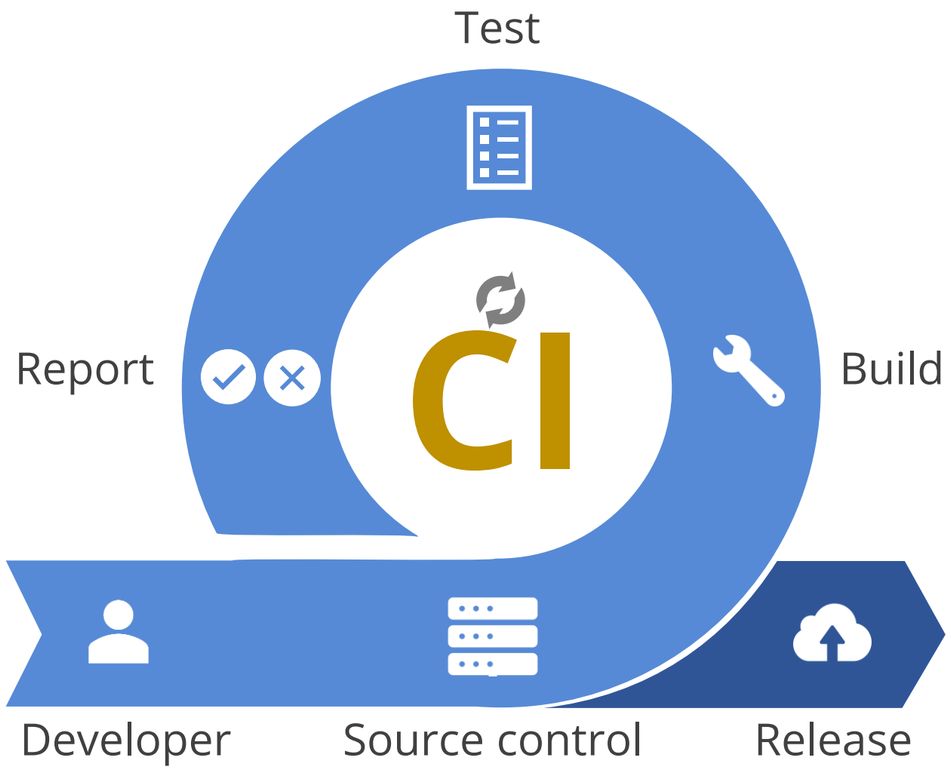
What Is a CI/CD Pipeline?

CI/CD pipeline stands for continuous integration and continuous delivery pipeline. It is essentially an automated workflow that streamlines the software development process by automating various stages.



Continuous Integration

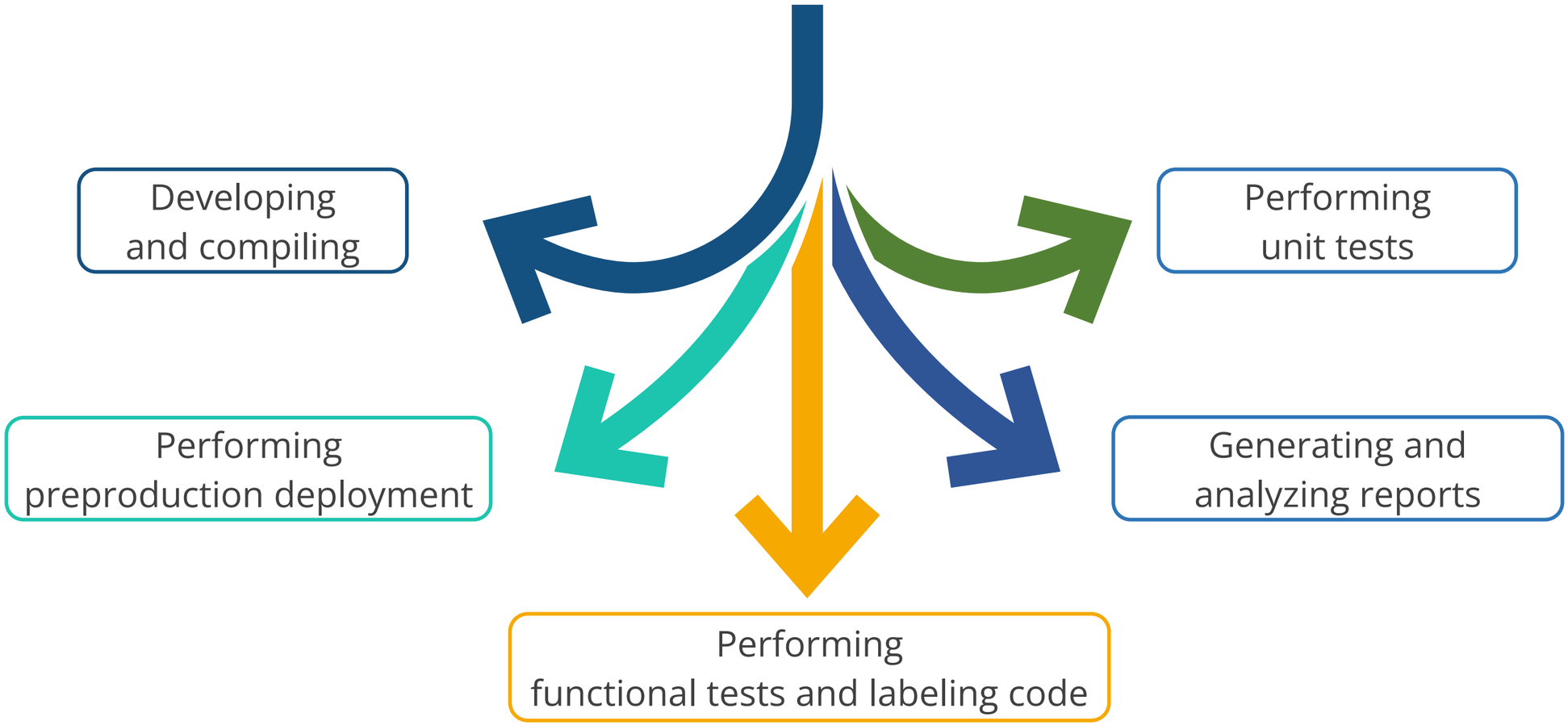
Continuous Integration (CI) is a development practice in which all development work is integrated as early as possible. It emphasizes frequent integration of code changes into a central repository.



It involves automating the build, testing, and verifying the code whenever a change is made.

Tasks Involved in Continuous Integration

Following is a description of the tasks involved in continuous integration:



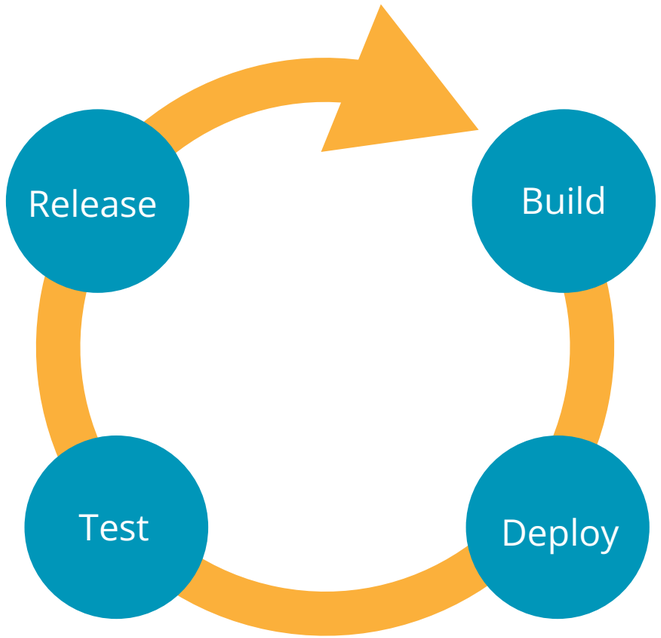
Advantages of Continuous Integration

Continuous integration offers several advantages, including:

- It gives a clear snapshot of the ongoing development work when automated end-to-end acceptance tests are conducted.
- It results in fewer bugs and quicker delivery when CI tools are used, as they are designed to identify and fix integration and regression issues faster.
- It is the automation of the deployment process that provides testers and end users with quick access to the software.
- It simplifies and accelerates delivery when the deployment process is automated by CI.

Continuous Delivery

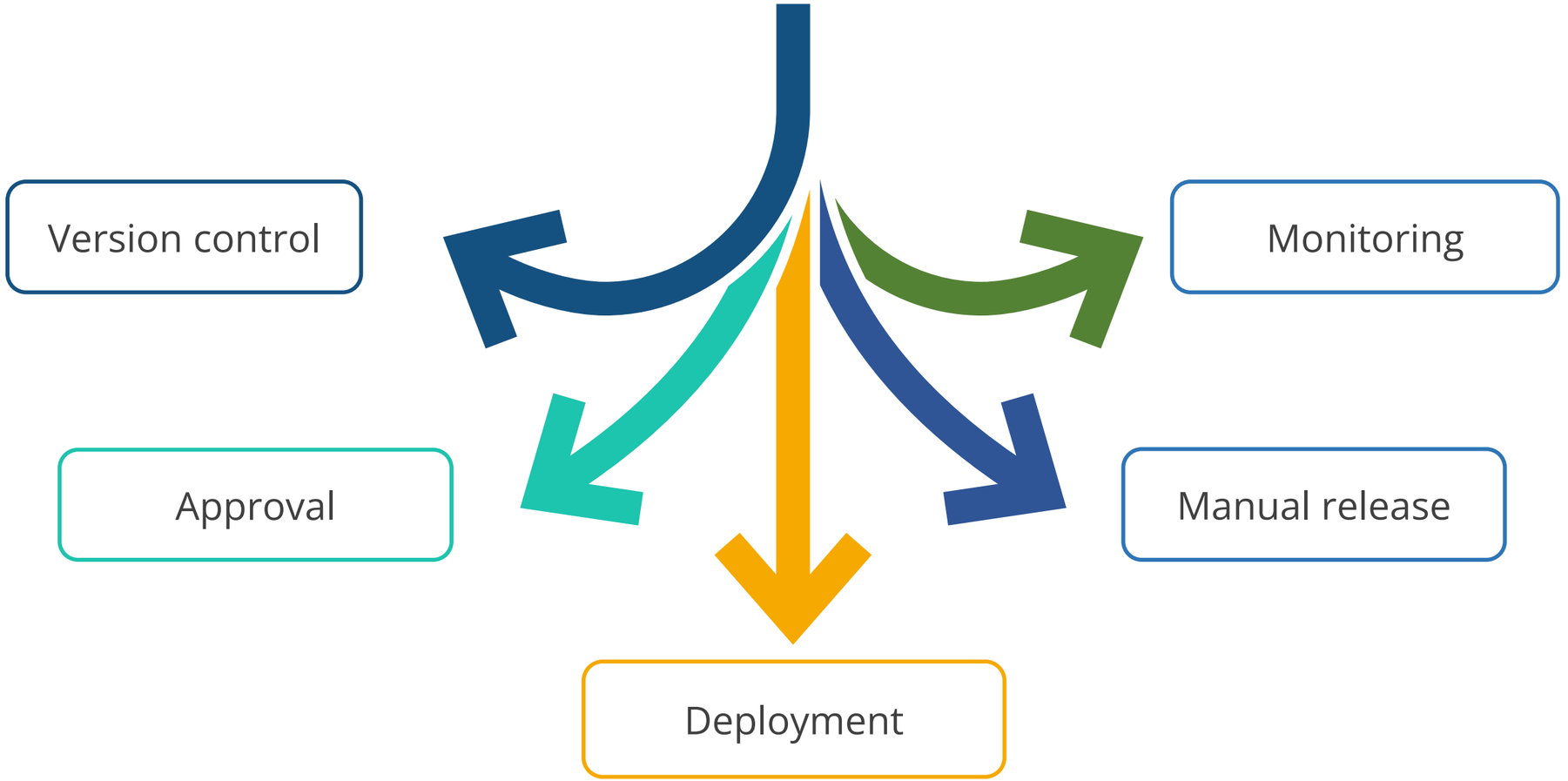
Continuous Delivery (CD) is an extension of continuous integration. It automates the entire software release process ensuring that the code changes are always ready for deployment to production.



This DevOps practice significantly reduces the risk and time associated with the traditional deployments.

Tasks Involved in Continuous Delivery

The following automated tasks are carried out under continuous delivery that streamlines the software release process:



Advantages of Continuous Delivery

CD boosts your team’s productivity and code quality by automating processes and delivering faster updates to customers. Here are some advantages of continuous delivery:

- It allows developers to develop and deploy high-quality software at a fast pace.
- It helps proactively resolve issues and ensure more stable production releases.
- It makes release processes as efficient and repeatable as possible.
- It boosts DevOps return on investment by making manual processes easily repeatable.

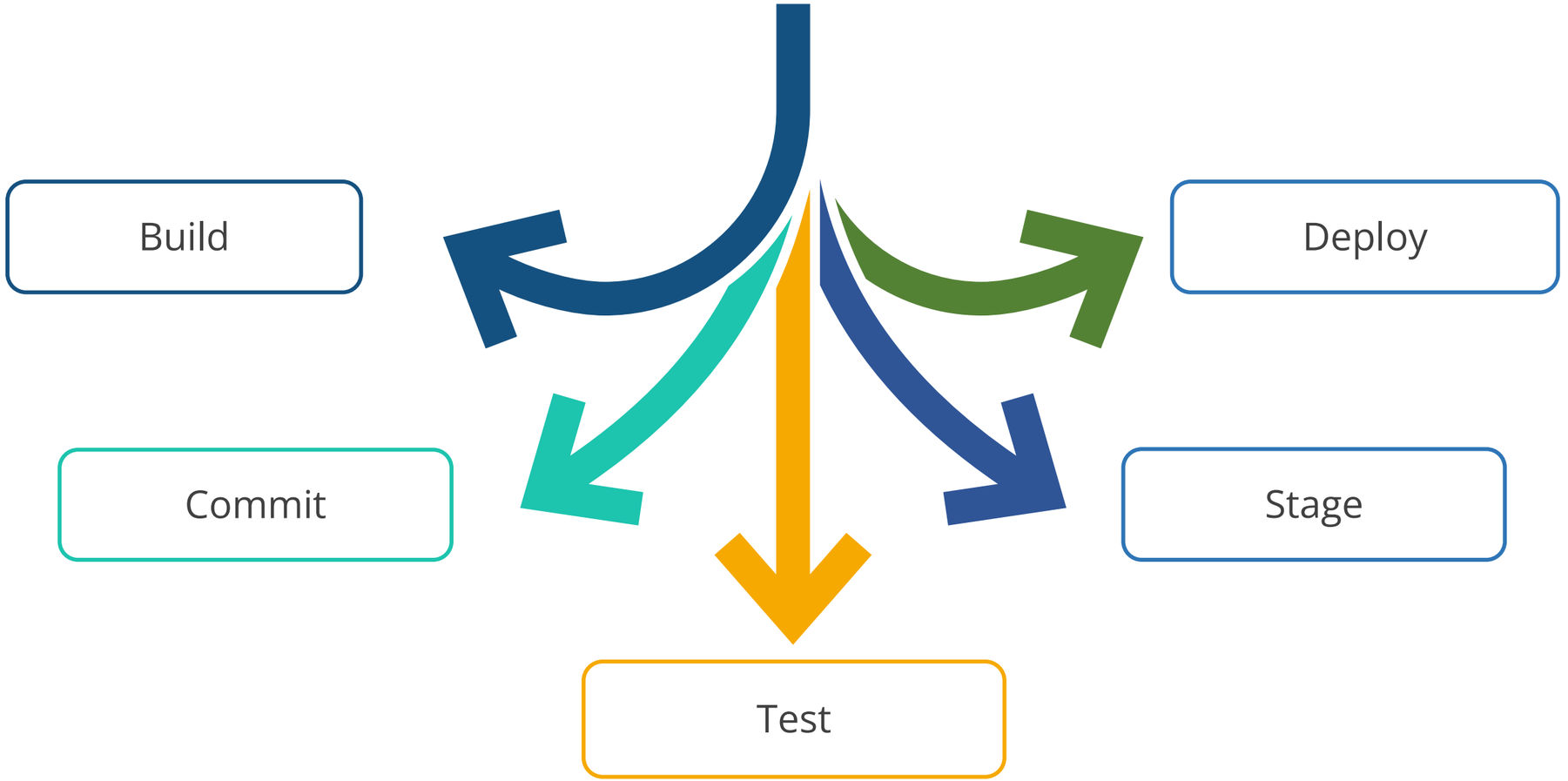
Continuous Deployment

Continuous Deployment (CD) is an extension of continuous integration. It aims to reduce the time between the development team writing a new line of code and its use in production.

It essentially advances continuous delivery by removing the requirement for manual approval before updates are pushed live.

Tasks Involved in Continuous Deployment

The following tasks are carried out to build a robust and automated pipeline with a strong focus on testing and monitoring to ensure smooth and reliable delivery of code changes to production:



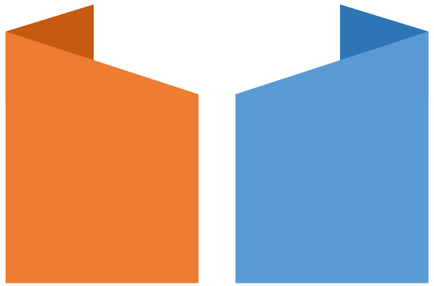
Advantages of Continuous Deployment

Continuous deployment simplifies the release process and offers the following advantages:

Fast delivery
It automatically deploys the changes right after development.

Fast feedback cycle
It maintains stability by quickly identifying and resolving bugs.

Low-risk releases
It becomes safer and repeatable with daily releases.

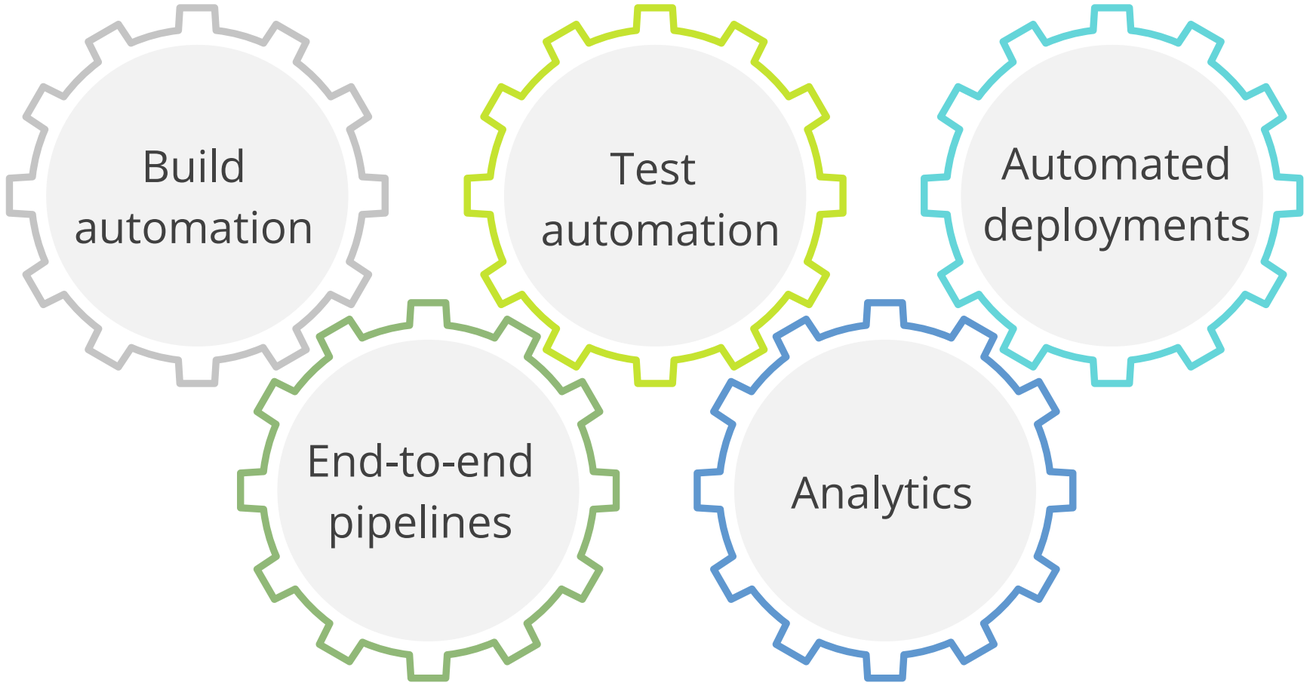


Flexible release options
It enables cost-free, instant software deployment.

Importance of CI/CD Pipeline

CI/CD enables organizations to develop software quickly and efficiently.

Here are the main benefits of implementing CI/CD pipelines:



CI/CD Workflow

A CI/CD workflow is essentially an automated process that includes various stages. Following is an outline of the usual workflow:

Source stage	It entails the organization and preservation of source code in a regulated and versioned way.
Build stage	It is the process where new code is combined with the existing code to create deployable artifacts.
Test stage	It refers to the process where automated tests are executed to verify the code and identify any bugs.
Deploy stage	It refers to the process where the code is deployed to either staging or production environments.

Elements of a CI/CD Pipeline

CI/CD elements can enhance DevOps workflow and software delivery, spanning development to deployment. The following are the fundamental elements of a CI/CD pipeline:

A single source repository	A SCM repository that includes all files and scripts for builds, such as source code, libraries, version control, and build scripts
Frequent check	Frequent merging of small code segments into the main branch, avoiding multiple sub-branches and simultaneous changes
Automated builds	The process of scripting and automating the retrieval of software code from a repository, compiling it into a binary artifact
Self-testing builds	The incorporation of pre-build scripts to validate code integrity before initiating builds

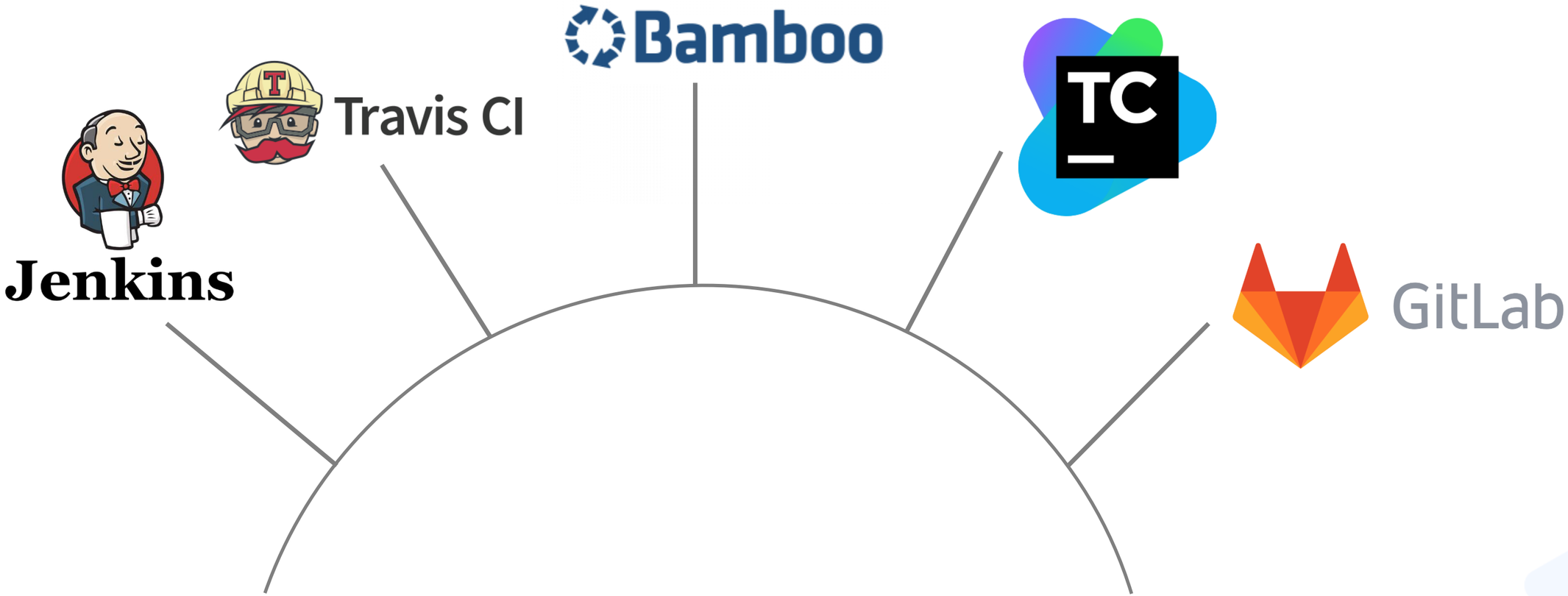
Elements of a CI/CD Pipeline

The following are the fundamental elements of a CI/CD pipeline:

Frequent iterations	Frequent iterations of small commits reduce conflict hideouts. This allows easy rollbacks in case of issues.
Stable testing environments	The emphasis is on testing code in a production clone rather than in live environments to identify any missed bugs before deployment.
Maximum visibility	Developers should access the latest executables and repository changes.
Predictable deployments	Deployments ought to be routine, low-risk, and reliable, allowing for updates at any time.

CI/CD Tools

Following is the list of the popular tools available for building CI/CD pipelines:



Quick Check

As a software development team lead, you are tasked with explaining the concept of a CI/CD pipeline to a new developer joining your team. How would you describe a CI/CD pipeline and its significance in the software development process?

A. A physical pipeline used in construction projects for transporting materials

B. A sequence of automated processes for building, testing, and deploying software changes

C. A method of organizing workflow in a development environment

D. A pipeline transporting oil and gas products across different regions

Introduction to Jenkins

What Is Jenkins?

It is an automation tool that provides a flexible and robust platform for building, testing, analyzing, deploying, and monitoring software changes. Here are some key points about Jenkins:

- It is an open-source project written in the Java programming language that facilitates continuous integration and continuous delivery (CI/CD) practices.
- It supports Windows, macOS, and other Unix-based operating systems.
- It is free, community-supported, and a popular first-choice tool for continuous integration.
- It is primarily used for on-premises deployment, but it can also be run on cloud servers.

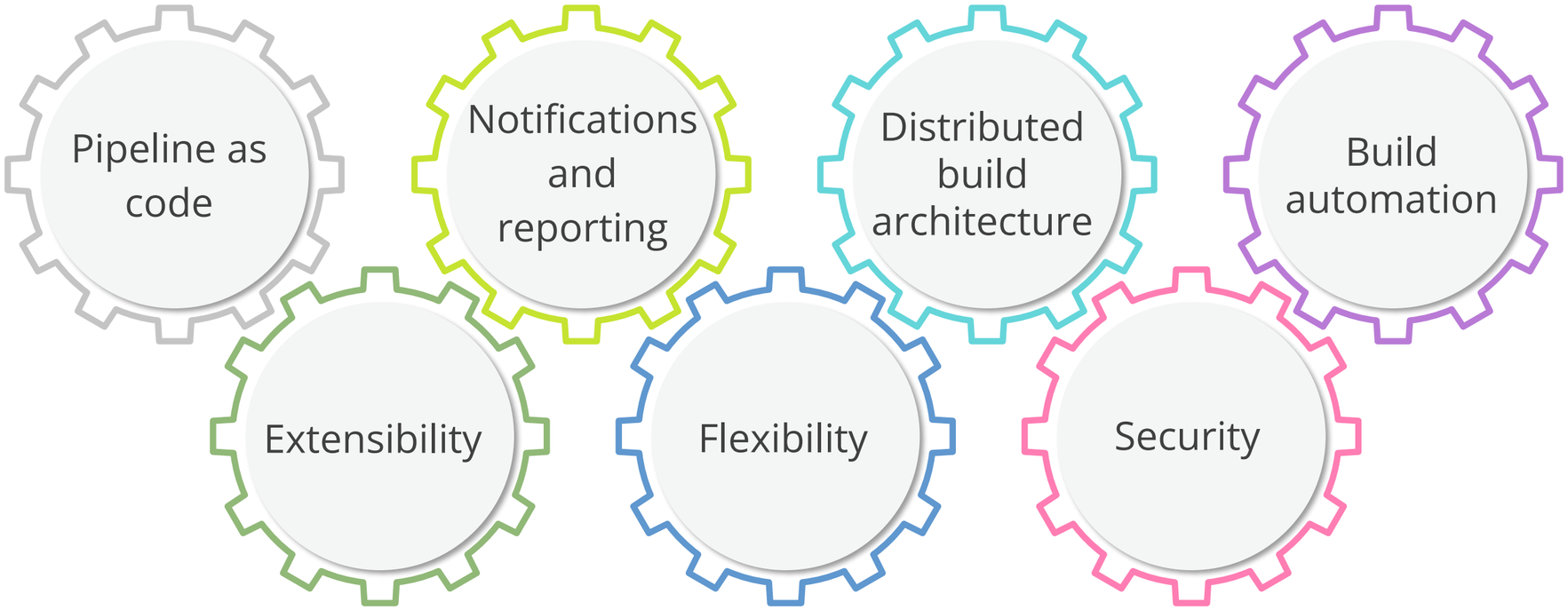
Why Use Jenkins?

It is used for creating and managing CI/CD pipelines, which includes the following:

Continuous integration	It instantly tests Git commits, ensuring release readiness and early bug detection.
Continuous delivery	It simplifies and automates software updates and feature rollouts.
Build and automation	Its plugins and tool integrations automate tasks such as code testing, app packaging, and deployment.
Pipeline orchestration	It manages CI/CD pipelines, simplifying software lifecycle oversight.
Extensibility and customization	Its plugins allow user-specific customizations and tool integrations.

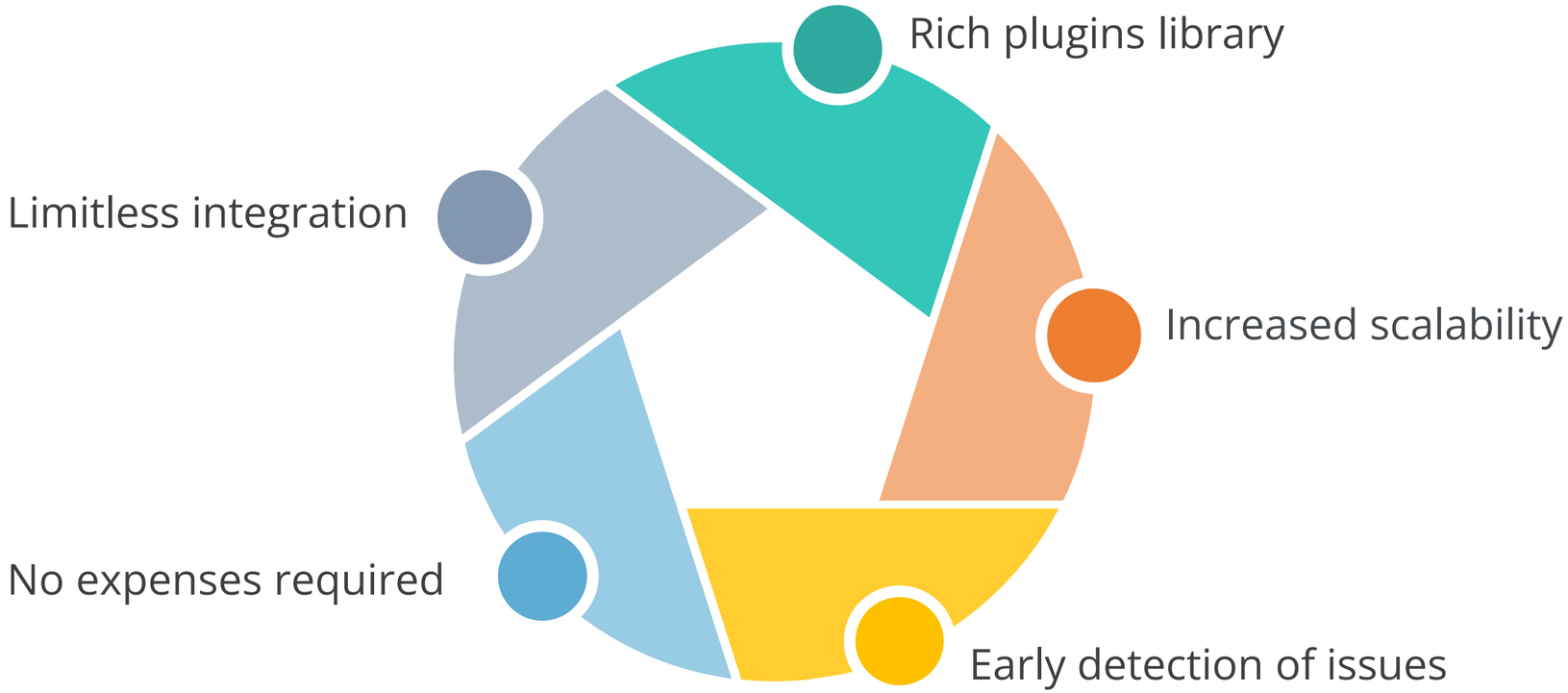
Jenkins: Features

It is a versatile automation tool that offers a wide range of features, including:



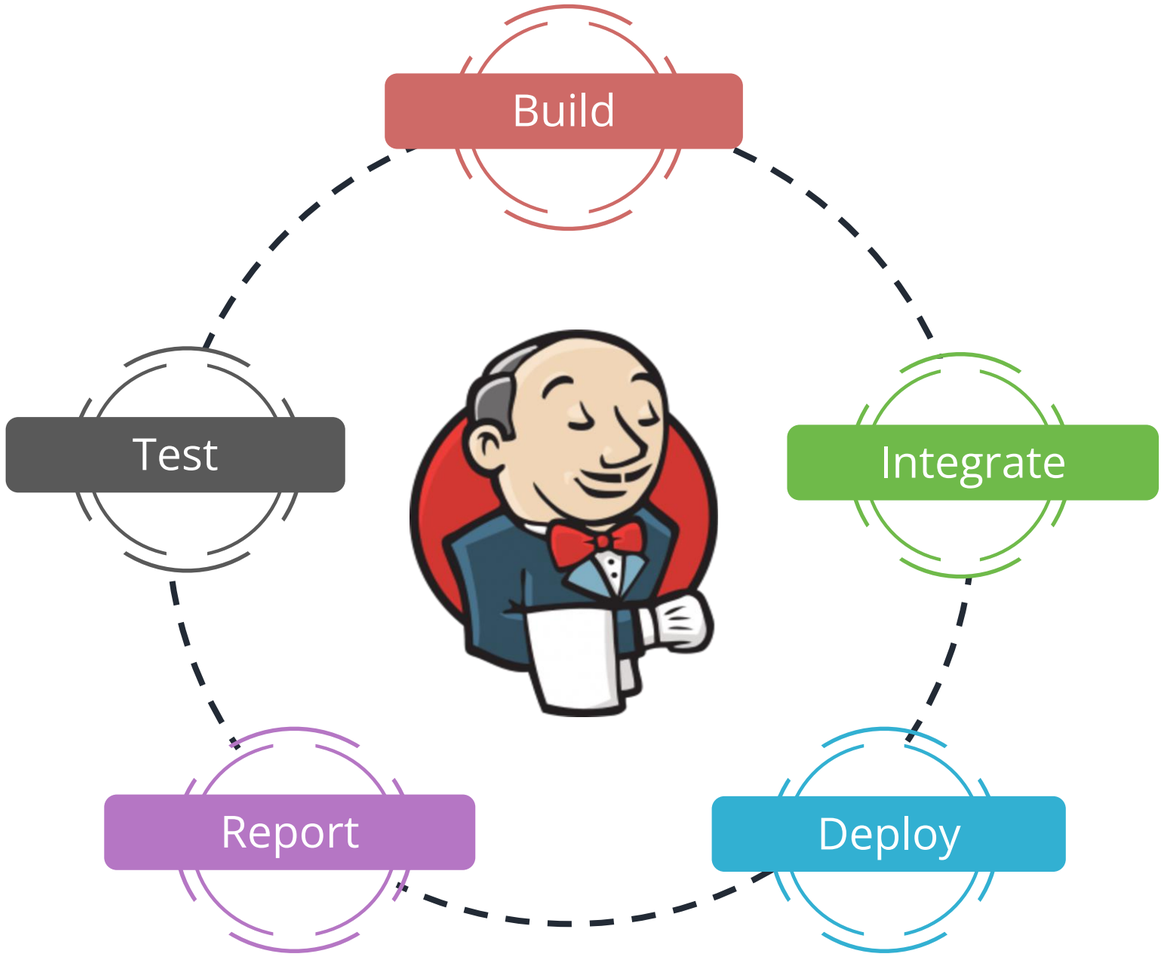
Benefits of Jenkins

Here are some of the benefits of Jenkins:



Role of Jenkins in DevOps

It is used to perform different tasks:

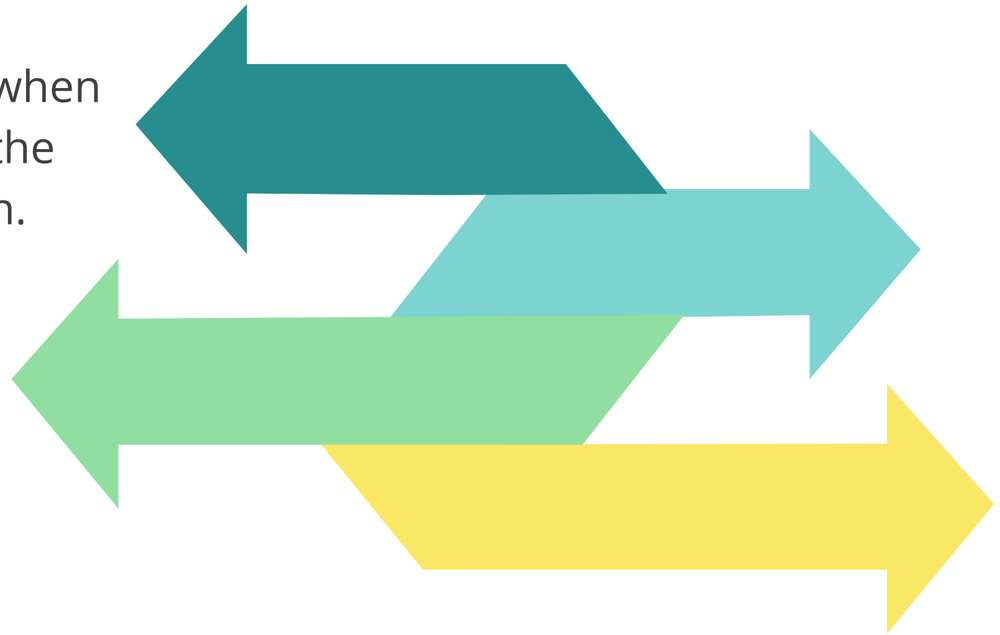


How Does Jenkins Work?

Jenkins working can be disintegrated into the following points:

Triggering builds:
Jenkins triggers a build when a commit is made to the development branch.

Integration testing:
Jenkins automates testing and identifies integration issues.



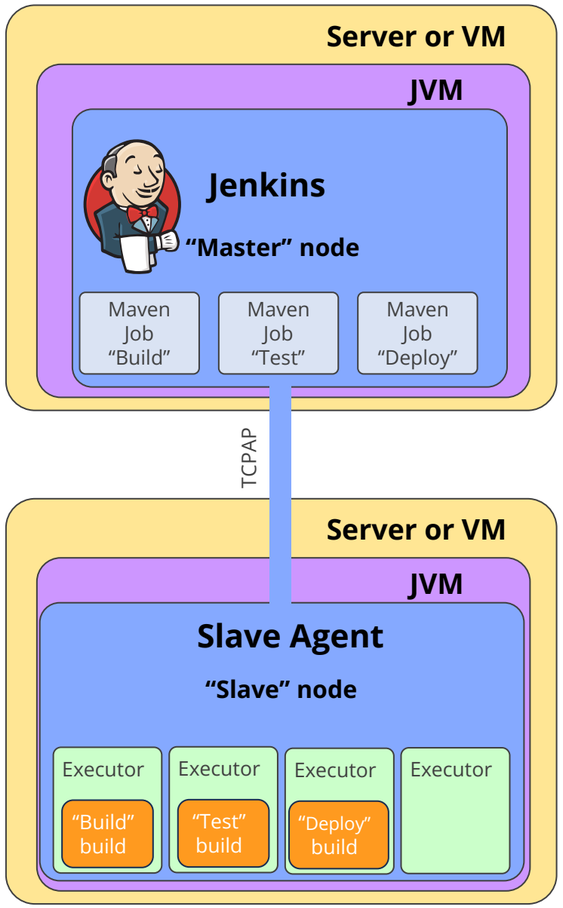
Build steps:
Jenkins tests code, alerts developers for errors, and moves to integration if successful.

User acceptance testing:
Jenkins automates user testing and deploys code that passes these tests.

Jenkins Architecture

Jenkins adheres to the master-slave architecture for managing builds. The slave and master agents communicate via a TCP/IP connection.

Jenkins



Assisted Practice

Creating a new user in Jenkins **Duration: 10 Min.**

Problem statement:

You have been assigned a task to create a new user in Jenkins for managing access and permissions within the Jenkins environment.

Outcome:

By completing this demo, you will be able to create a new user in Jenkins, effectively managing access and permissions within the Jenkins environment.

Note: Refer to the demo document for detailed steps

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed:

1. Log in to the Jenkins CI tool
2. Access the user management section to manage users

ASSISTED PRACTICE

Quick Check

You are part of a DevOps team responsible for streamlining the software development lifecycle. Which aspect of Jenkins would be most beneficial for automating the build and deployment process of your project?

A. Using Jenkins' plugins for integrating various tools and technologies

B. Understanding Jenkins' role in enabling continuous integration and delivery (CI/CD) practices

C. Exploring Jenkins' distributed architecture for efficient resource utilization

D. Using Jenkins across different platforms for seamless integration with cloud environments

Key Takeaways

- Continuous Integration (CI) is a development practice that emphasizes early integration of all development work, promoting frequent integration of code changes into a central repository.
- Continuous Delivery (CD) extends continuous integration, automating the software release process to ensure code changes are always deployment-ready for production.
- Jenkins is an automated tool offering a versatile platform for developing, testing, analyzing, deploying, and monitoring software changes.
- Jenkins follows a master-slave architecture to manage builds, where communication between the master and slave agents occurs over a TCP/IP connection.
- The Jenkins interface allows users to manage builds, set up pipelines, and administer the environment efficiently.

Thank You