

# DevOps Foundations: Version Control and CI/CD with Jenkins



## Jenkins Build Tools and Build Triggers



# Learning Objectives

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

- Outline how Jenkins integrates with various build tools to automate build processes
- Utilize the Maven build tool in Jenkins to automate the compilation, testing, and packaging of the Java projects efficiently
- Apply knowledge of unit testing concepts to create effective test suites in Jenkins
- Evaluate the benefits of using artifacts and fingerprints in a CI/CD pipeline for tracking changes, ensuring build integrity and facilitating faster rollbacks
- Configure webhook integration in Jenkins to automate the triggering of events from external systems

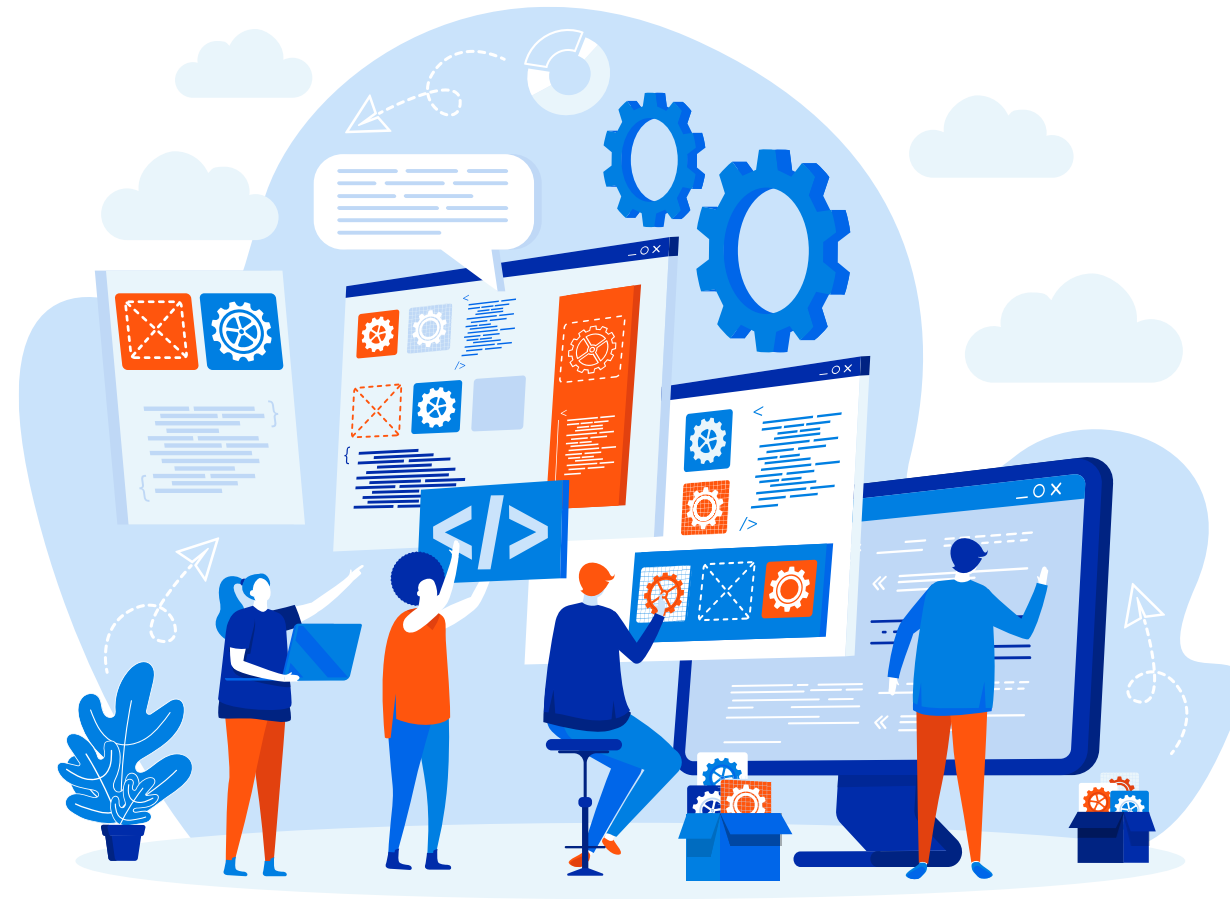




# Build Tools Integration

# Jenkins Build Tools

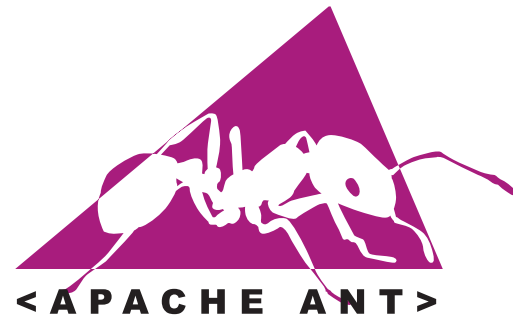
It automates the conversion of source code into deployable formats, enhancing SDLC phases and serving as a vital component in software development.



Build tools like Ant, Maven, and Gradle can be used to perform build automation and IDE integration.

# Popular Build Tools

Below are some popular build tools available in Jenkins:



## Apache Ant

A Java tool that automates building software, simplifying tasks like compiling, packaging, and deploying, primarily using XML configuration.



## Maven

It is a popular build automation tool that has found its niche in Java development. It controls construction via XML-based Project Object Model (POM) files

# Popular Build Tools



## Gradle

A flexible and fast open-source tool for automating builds, it uses Groovy-based scripts to define tasks, focusing on adaptable and efficient development processes.



## Travis CI

It is the preferred cloud solution for automating build and testing on GitHub, due to its easy-to-use interface and cloud-based setup.

## Assisted Practice



### Setting up Git configuration in Jenkins job

Duration: 15 Min.

#### Problem statement:

You have been assigned a task to set up Git configuration in Jenkins for streamlined version control and automated build processes.

#### Outcome:

By the end of this demo, you will be able to set up Git configuration in Jenkins for efficient version control and automated build processes.

**Note:** Refer to the demo document for detailed steps:



# Assisted Practice: Guidelines

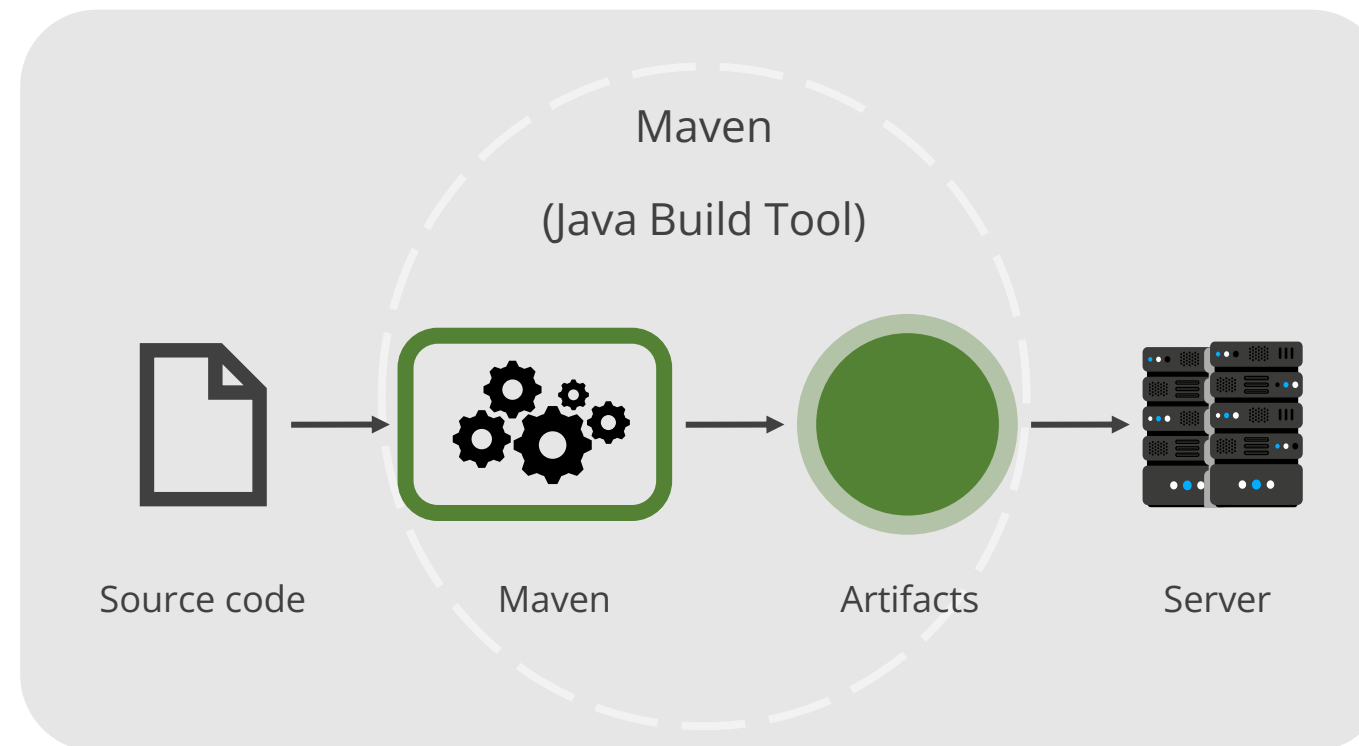


Steps to be followed:

1. Create new credentials in Jenkins for Git configuration
2. Create a Jenkins freestyle project
3. Configure Git in Jenkins

# Maven Build Tool

It is an open-source build tool for Java projects that simplifies the build process, ensuring consistency and streamlined project management.



It supports building and managing projects in various languages like C#, Ruby, and Scala. It's based on a Project Object Model (POM) for managing builds.

# Maven Build Tool

Some of the key functionalities of Maven are:

## **Project structure and setup:**

Ensures a consistent project structure and simplifies project navigation

## **Dependency management:**

Downloads and manages external libraries, ensuring compatibility and eliminating manual downloads for project

## **Build automation:**

Automates tasks like compiling, packaging, and testing, Maven frees developers and ensures consistent builds across environments

# Maven Build Tool

Some of the key functionalities of Maven are:

## **Reporting and documentation:**

Generates reports on project aspects and automates project documentation based on metadata

## **Extensibility:**

Enables custom plugin creation in Java or scripting languages for specific project needs

## **Integration with other tools:**

Integrates with CI systems for automated builds and deployments alongside other development tools.

## Assisted Practice



### Integrating Maven with Jenkins

Duration: 10 Min.

#### Problem statement:

You have been assigned a task to install the Maven plugin in Jenkins for smooth integration and automation of Maven-based build processes within the Jenkins environment.

#### Outcome:

By the end of this demo, you will be able to install the Maven plugin in Jenkins for seamless integration and automation of Maven-based build processes.

**Note:** Refer to the demo document for detailed steps:

# Assisted Practice: Guidelines



Steps to be followed:

1. Install the Maven plugin
2. Set up Global Tool Configuration
3. Fork a sample repository
4. Integrate Maven with Jenkins


# Build Automation Using Jenkins

Build automation refers to the process of automating the repetitive tasks involved in building software.




# Build Automation Using Jenkins


Below are some of the tasks:




**Retrieving code from VCS:** Retrieves the source code from version control systems such as Git or SVN to begin the build process



**Compiling code:** Compiles the source code into executable files or libraries based on the project configuration and dependencies



**Running unit tests:** Executes unit tests to ensure the code functions correctly and meets the specified requirements



**Packaging the final application:** Packages the compiled code, resources, and dependencies into a distributable format like JAR or WAR for deployment



# Build Automation Using Jenkins: Benefits

Some advantages of using Jenkins for build automation are:

- Frees developers from repetitive tasks
- Ensures consistent builds across environments
- Improves build quality through automated testing
- Speeds up builds through parallelization
- Enhances collaboration with a central build management hub

## Assisted Practice



### Setting up Maven build job in Jenkins

Duration: 10 Min.

#### Problem statement:

You have been assigned a task to set up a Maven build job in Jenkins for automating the build process, enabling continuous integration to enhance the software development lifecycle.

#### Outcome:

By the end of this demo, you will be able to set up a Maven build job in Jenkins to automate the build process and enabling continuous integration in your software development lifecycle.

**Note:** Refer to the demo document for detailed steps:

# Assisted Practice: Guidelines

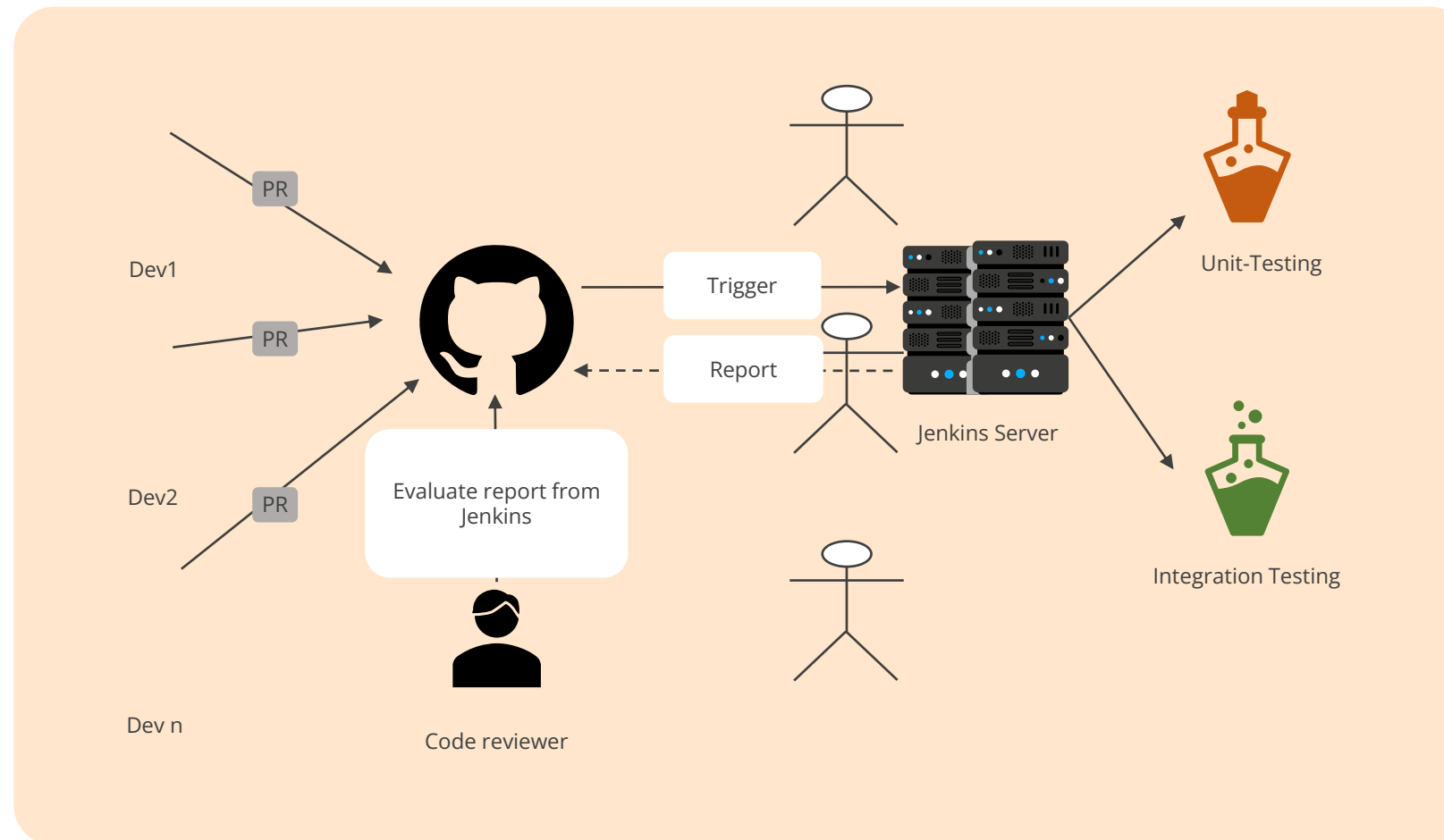


Steps to be followed:

1. Log in to Jenkins CI tool and configure Maven freestyle job

# Configuring Unit Test Cases

Jenkins plugins automate unit testing and improve code quality with clear pass or fail results.



Configure Jenkins by installing **Testing** plugins, specifying the test file location, and enabling automatic execution of unit tests.

# Configuring Unit Test Cases

Key steps to configure Jenkins for running unit tests:

## 1. Install necessary plugins:

- Install unit test framework plugins in Jenkins
- Choose plugins based on the test framework (for example: JUnit for Java tests, NUnit for .NET tests)
- Interpret and present the test results effectively by utilizing these plugins and allowing Jenkins to act as translators

# Configuring Unit Test Cases

## 2. Configure build steps:

- Configure how Jenkins executes tests after installing the right plugins
- Setting up build steps within the project configuration typically involves the below processes:
  - **Invoke Ant:** For Java projects, use Ant build scripts to run tests
  - **Build Maven:** For Maven projects, leverage Maven goals to execute tests

# Configuring Unit Test Cases

## 3. Run tests and view results:

- Execute unit tests automatically with the configured plugins and build steps
- Trigger a build and let Jenkins handle the execution of unit tests automatically
- Use plugins to show clear pass or fail results and errors, providing insights into code health and status

# Generating Test Reports in Jenkins

It involves automatically creating detailed summaries and analyses of test results following the execution of automated tests as part of a build process.



These reports offer insights into test outcomes (passed or failed), indicating the number of tests run along with any encountered errors or issues during testing.



# Generating Test Reports in Jenkins

Key steps in generating test reports Jenkins include:



# Generating Test Reports in Jenkins

## Benefits of test reports:

### Detailed analysis:

Identify failing tests by analyzing individual test cases using reports

### Improved collaboration:

Share testing results effectively with the team using clear and shareable reports



### Tracked trends:

Monitor reports over time to track code quality trends and identify areas for improvement

## Assisted Practice



### Publishing test cases reports in Jenkins

Duration: 15 Min.

#### Problem statement:

You have been assigned a task to publish test case reports in Jenkins to enhance transparency, accountability, and quality in the software development process

#### Outcome:

By the end of this demo, you will be able to configure Jenkins to publish test case reports, promoting transparency, accountability, and quality in your software development process.

**Note:** Refer to the demo document for detailed steps:

# Assisted Practice: Guidelines



Steps to be followed:

1. Log in to the Jenkins CI tool and publish test case reports

## Quick Check

Imagine working in a company where you need to track code quality trends and identify areas for improvement over time. Which feature of Jenkins should you utilize?

- A. Building automation using Jenkins
- B. Configuring unit test cases
- C. Generating test reports in Jenkins
- D. Configuring Git in Jenkins job



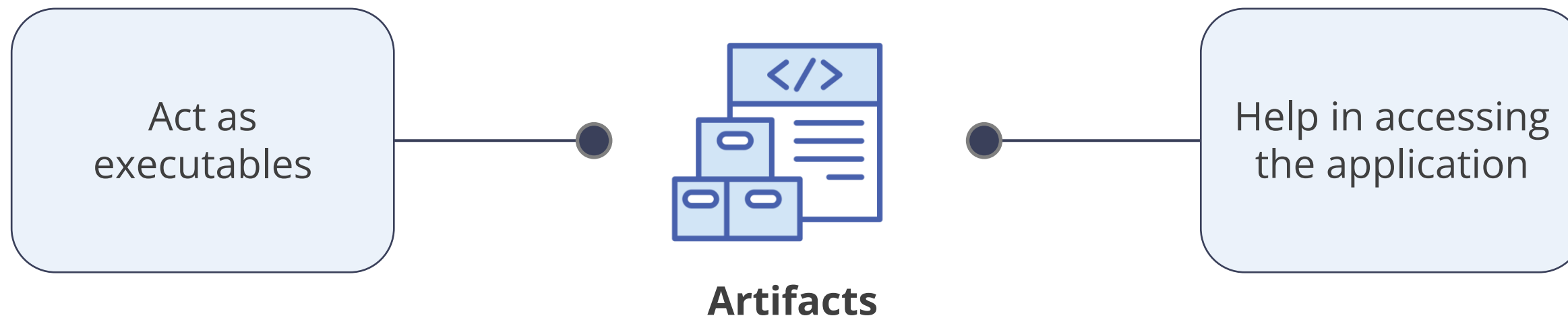


## Build Triggers

# Artifacts and Fingerprints

**Artifacts** encompass compiled code, documentation, reports, and binaries created during builds.

Artifacts are the files produced by a build.



These are vital for deployment or testing and are retained for later use or distribution.

# Artifacts and Fingerprints

**Artifacts** can be of different file types depending on the project setup, such as:

**Compiled code files:** Produce the final executable code after compilation, such as .exe for Windows or .jar for Java applications

**Test reports:** Provide summaries of unit test executions, showing pass or fail rates and errors, often generated using tools like JUnit

**Package deployment files:** Create compressed files containing everything needed to deploy an application to a server

**Record logs:** Capture build details such as console output and errors during the process using the text files



# Artifacts and Fingerprints

**Fingerprints** in Jenkins are unique identifiers assigned to each artifact, helping track their usage dependencies and ensuring traceability for effective version and dependency management.

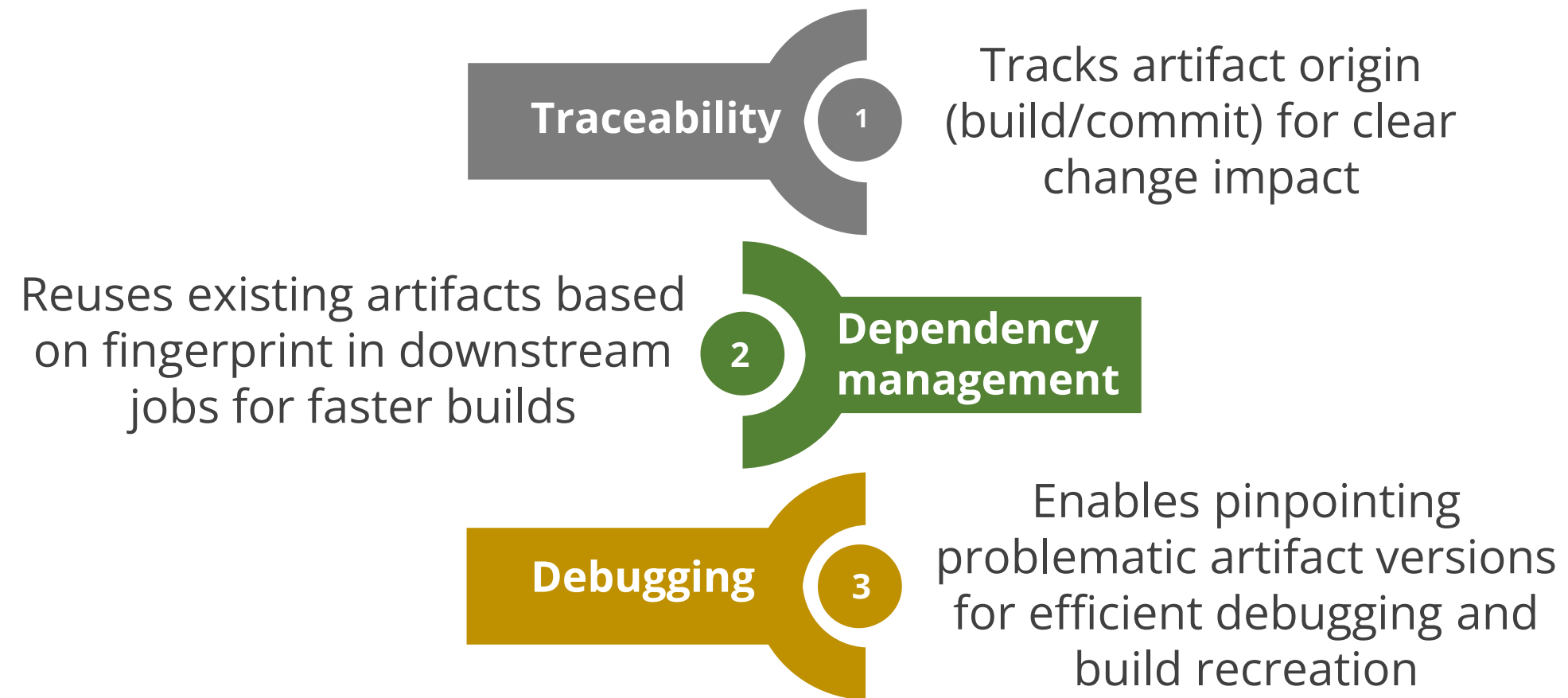


**Jenkins fingerprint**

Jenkins stores fingerprints in its database along with artifact details and build information.

# Artifacts and Fingerprints

Benefits of using artifacts and fingerprints:

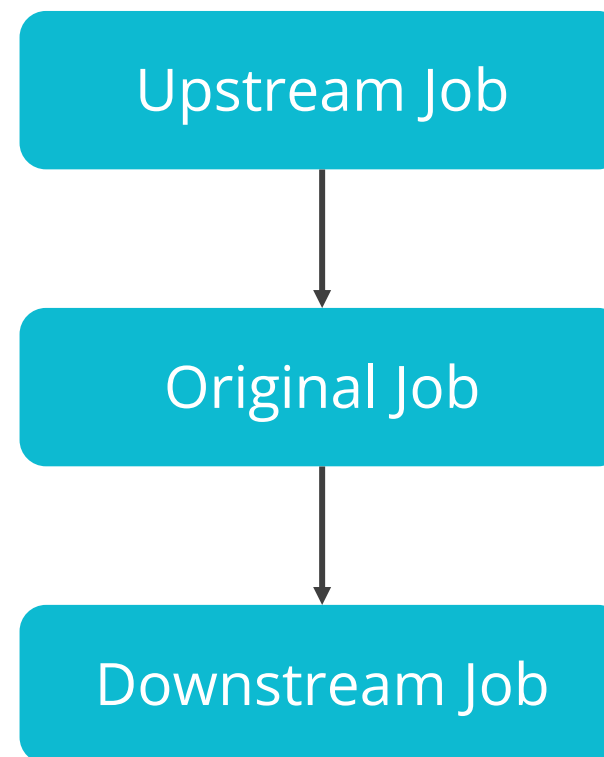


# Upstream and Downstream Jenkins Job

An **Upstream job** is a configured project that triggers a project as part of its execution.

A **Downstream job** is a configured project that is triggered as part of an execution of pipeline.

Given below is the order of job triggers:



# Upstream and Downstream Jenkins Job

Developers can interlink different Jenkins jobs in CI/CD automation using upstream and downstream configurations. Below is a comparison of how these jobs are configured:

## Upstream Job

---

- It is triggered before the original job is triggered.
- Configure execution of the original job depending on upstream job Build status

## Downstream Job

---

- It is triggered after the original job is triggered.
- Configure downstream job execution depending on the Build status of the original job

# Configuring Upstream and Downstream Jenkins Job

To configure the build condition for an upstream Jenkins job, select the appropriate condition in the Build Triggers tab as shown:

### Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☒ Build after other projects are built ?

Projects to watch

UpStreamJob,

☒ Trigger only if build is stable  
☐ Trigger even if the build is unstable  
☐ Trigger even if the build fails

☐ Build periodically ?

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

# Configuring Upstream and Downstream Jenkins Job

To configure Build condition for a downstream Jenkins job, select the appropriate condition in the Post-build Actions tab as shown:

### Post-build Actions

**Build other projects**

X

?

Projects to build

DownStreamJob

☒ Trigger only if build is stable





☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

Add post-build action ▾

# Configuring Upstream and Downstream Jenkins Job

To understand the configuration of upstream and downstream Jenkins jobs based on trigger conditions:

-  **Build success:** Triggers are set up so that the downstream job only initiates when the upstream job finishes successfully, which is the default option
-  **Build failure:** Initiates even if the upstream job fails, the downstream job triggers, which can be useful for notification or cleanup purposes
-  **Always:** Activates regardless of the upstream job's outcome, the downstream job triggers, although this is rarely used
-  **Custom triggers:** Utilizes pipeline scripting for more advanced trigger conditions based on specific build parameters or artifacts

# Build Periodically

Configure a Jenkins job that involves setting it to build periodically using a cron expression, which defines the desired schedule for the job's execution.

**Build Triggers**

☒ Build after other projects are built ?

Projects to watch

☐ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

☐ Always trigger, even if the build is aborted

☐ Build periodically ?

☒ Build periodically with parameters ?

Schedule ?



# Build Periodically

Benefits of using periodic builds:

## Improved efficiency

Automate repetitive build tasks and freeing developers for other activities

## Enhanced consistency

Ensure builds are running regularly and minimizing the risk of regressions

## Early detection of issues

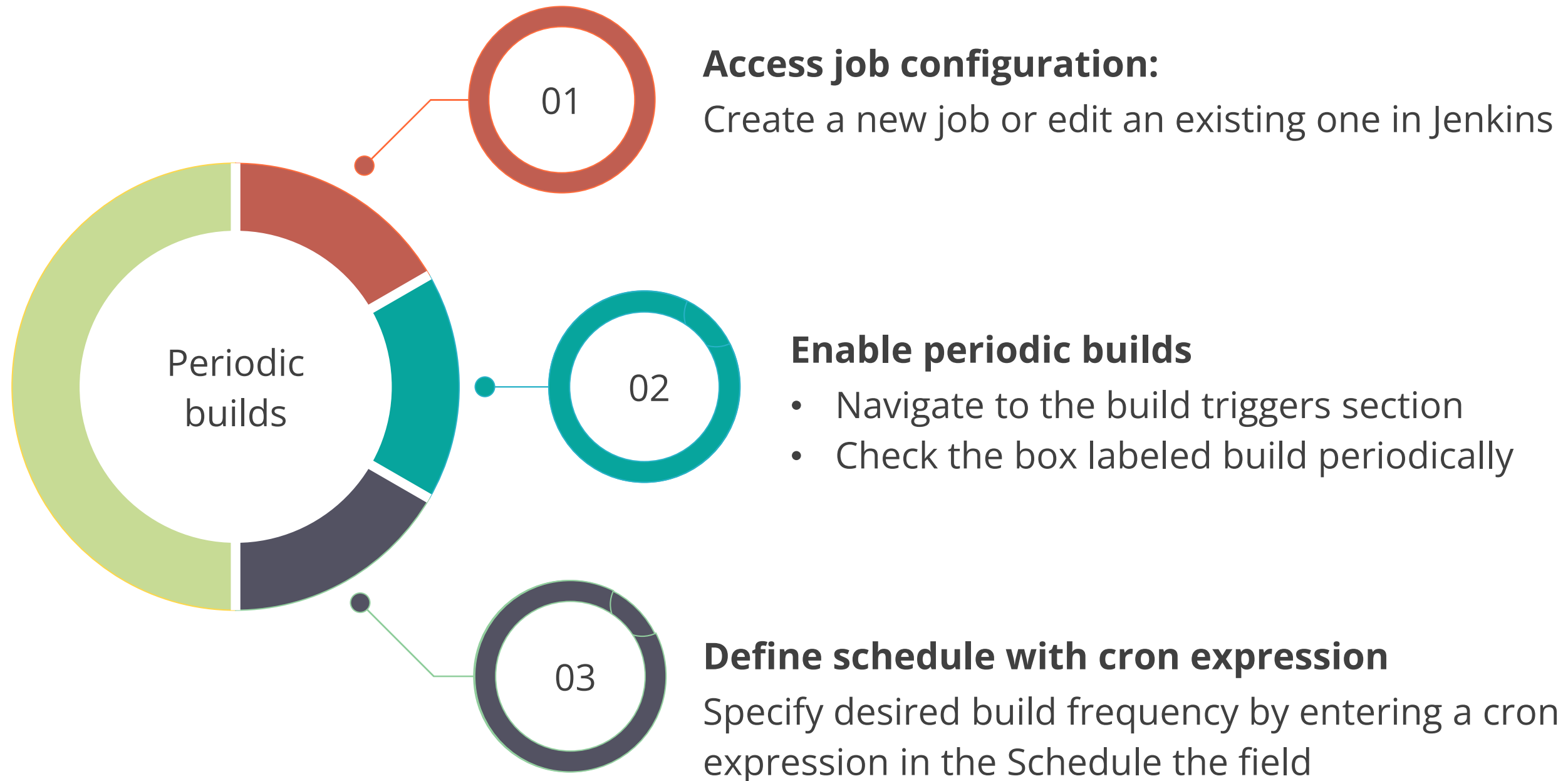
Mitigate potential problems early by conducting frequent builds and receiving feedback

## Streamlined workflows

Integrate periodic builds into the CI/CD pipeline for a smooth development process

# Build Periodically

Below are the steps to configure periodic builds in Jenkins:



# Build Periodically

Cron expression examples:

@hourly

Runs the job every hour

H/15

Initiates the job every 15 minutes

0 0 \* \* \*

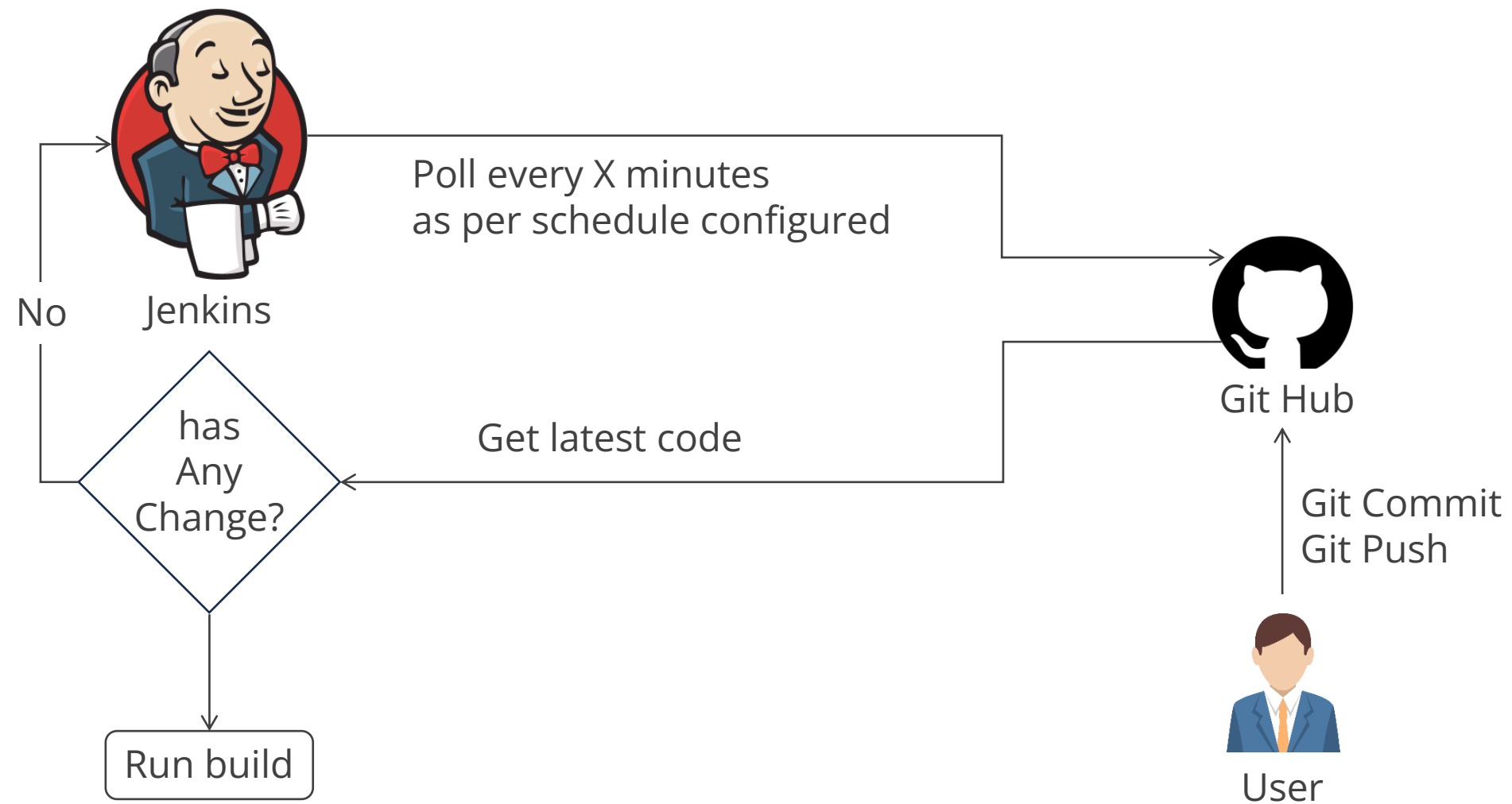
Triggers the job daily at midnight

30 17 \* \* FRI

Executes the job every Friday at 5:30 PM

# Poll SCM

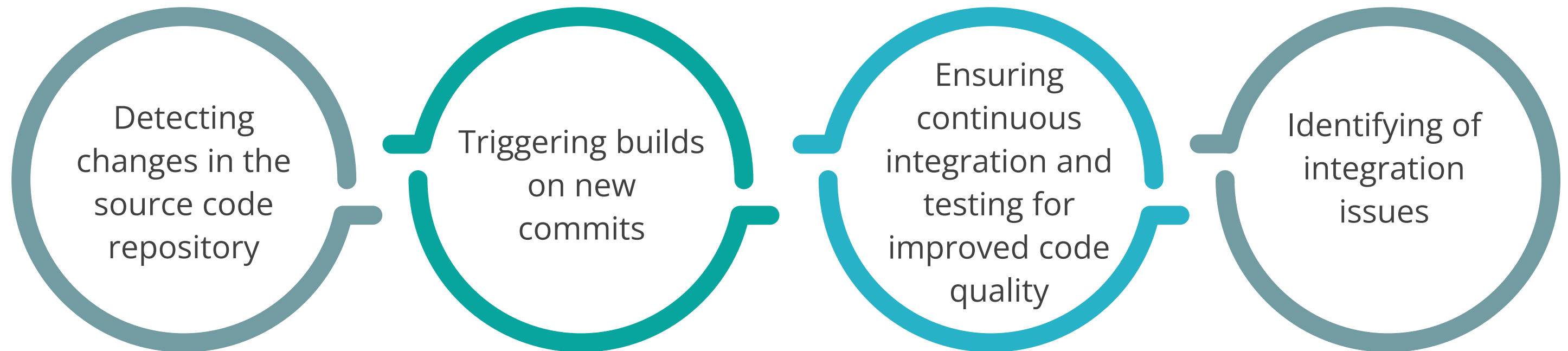
It is a continuous integration technique that involves periodically checking the Source Code Management (SCM) system for any changes or new commits.



This process enables automated builds of the project whenever new commits are detected since the last build.

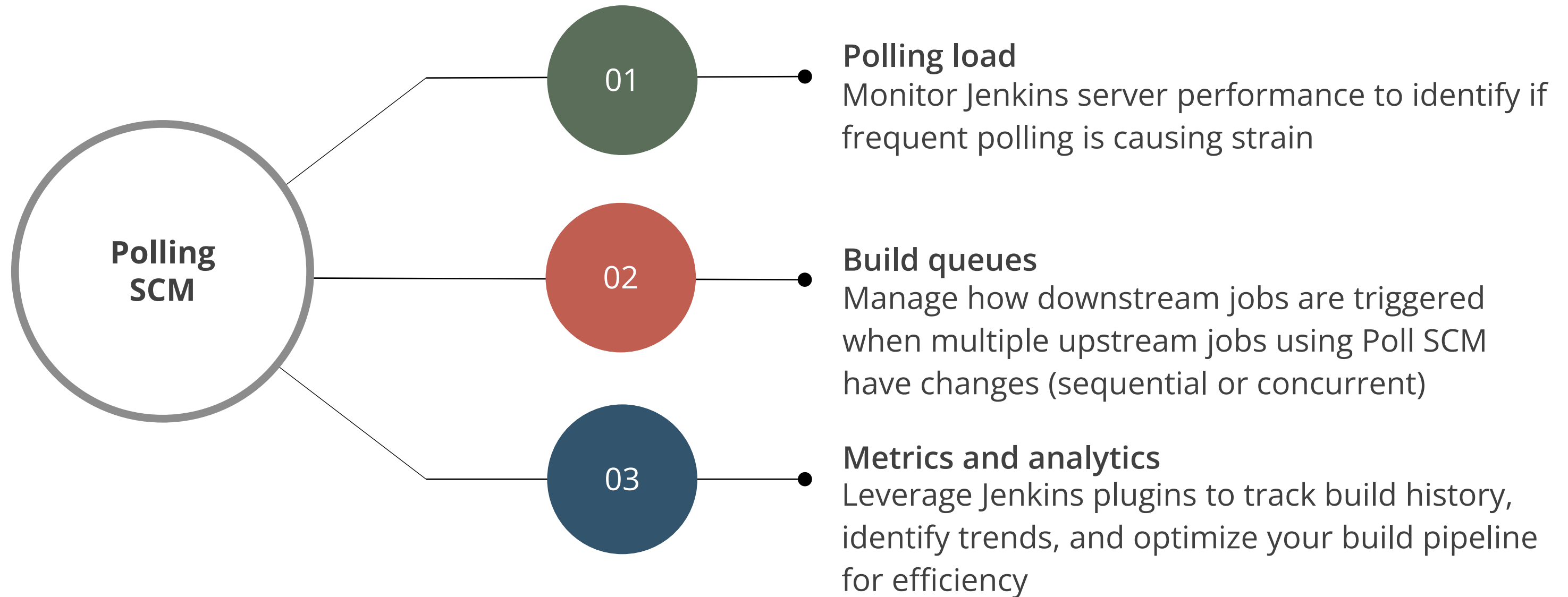
# Poll SCM: Benefits

Benefits of using Poll SCM in Jenkins:



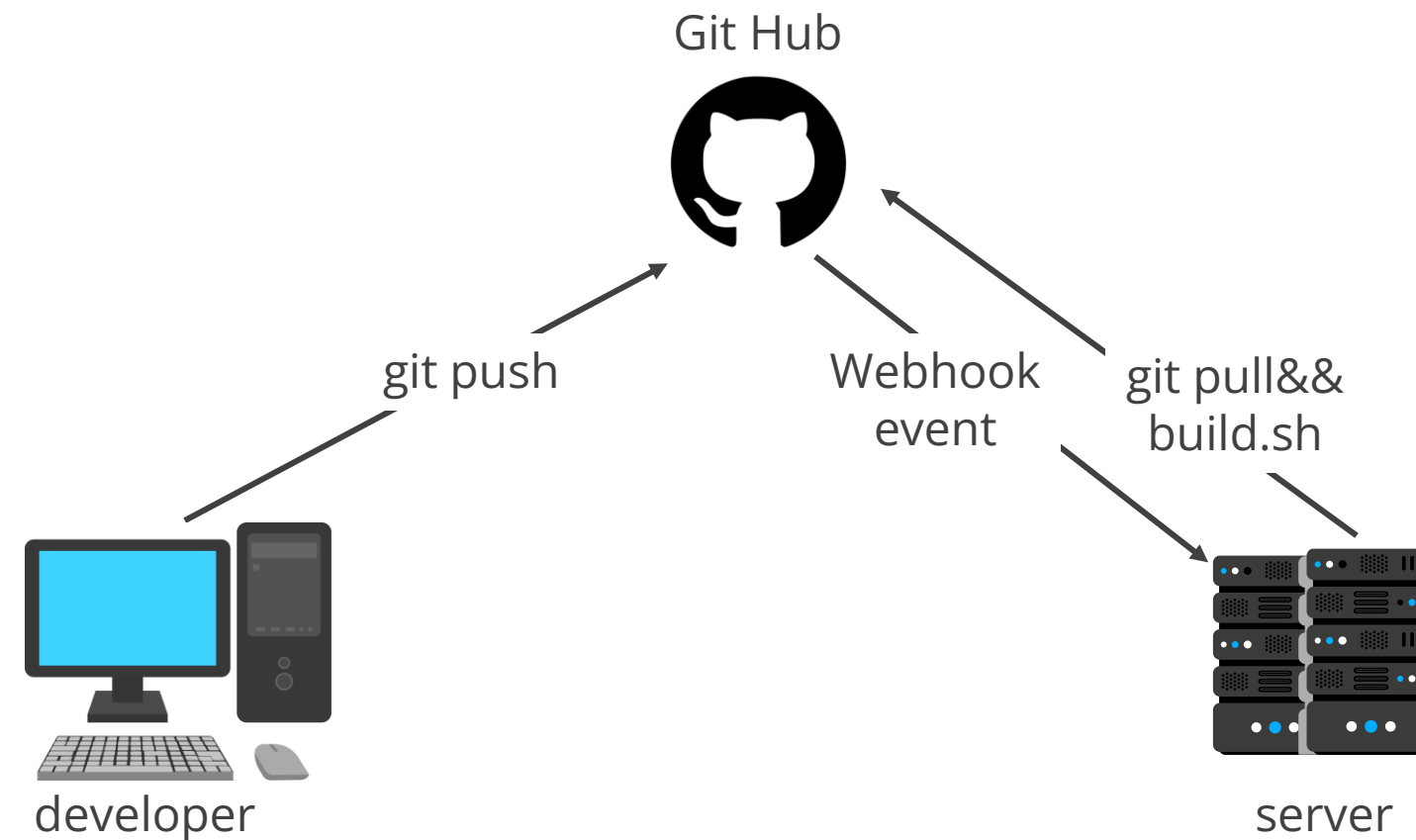
# Poll SCM

Below are the ways to enhance Jenkins efficiency through monitoring, optimization, and analytics:



# Webhook Configuration

Webhooks are real-time messaging mechanisms that enable one application to notify another about specific events.



When a change in a Git repository takes place, a webhook sends a message to a specific web address; this helps different tools and platforms communicate and automate tasks smoothly.

# Webhook Configuration

Use of webhooks in Jenkins:

## Automated build triggers

By automating the build process upon receiving an event notification, Jenkins streamlines the CI/CD pipeline without manual job initiation.

## Enhanced efficiency

By streamlining build triggers, it helps developers to focus on core tasks, thereby improving development cycles.

## Improved collaboration

By enhancing collaboration, Jenkins seamlessly integrates with Git and version control systems, fostering better information flow within the team.



# Webhook Configuration

Steps to configure Webhooks in Jenkins:

## Installing plugins

Install plugins like generic webhook trigger or specific service plugins like GitHub plugin to enable webhook functionality within Jenkins

## Creating a Jenkins job

Create a Jenkins job and define the build tasks to be executed during the build process

## Configuring the webhook trigger

Configure webhook trigger in job settings by selecting the appropriate plugin from the Build Triggers section

## Setting the webhook URL

Set the webhook URL by providing the unique URL generated by Jenkins, which the external service will use to send notifications

# Difference between Poll SCM, Build Periodically, and Webhooks

Poll SCM	Build Periodically	Webhooks
Checks the repositories for changes	Initiates builds on fixed time intervals	Receives direct notifications from the VCS
It has a simple configuration with a polling schedule	It requires setting up of the cron syntax for intervals	It mainly requires setting up webhook URLs
A change in the git repository triggers the builds.	A time-based schedule triggers the builds.	Real-time triggers are based on VCS events.
It is used for projects with infrequent code changes	It is used for scheduled maintenance and integration testing	It is used for real-time integration, and to get quick feedback on code changes

## Assisted Practice



### Setting up Poll SCM configuration in Jenkins

Duration: 15 Min.

#### Problem statement:

You have been assigned a task to set up poll SCM configuration in Jenkins for automating build triggering and optimizing resource usage on Jenkins servers.

#### Outcome:

By the end of this demo, you will be able to configure Jenkins to use poll SCM, automating build triggering and optimizing resource usage on your Jenkins servers.

**Note:** Refer to the demo document for detailed steps:

ASSISTED PRACTICE

# Assisted Practice: Guidelines



Steps to be followed:

1. Log in to Jenkins using the credentials

## Quick Check

You are a developer who needs Jenkins to monitor changes in your version control system and trigger a build only when new commits are detected. What should you use?

- A. Artifacts and fingerprints
- B. Upstream and downstream Jenkins job configuration
- C. Build periodically
- D. Poll SCM



# Key Takeaways

- Build tools automates the conversion of source code into deployable formats enhancing SDLC phases.
- Git configurations in Jenkins job focuses on specifying how Jenkins interacts with the Git repository to retrieve code for the build process.
- Maven build tool is an open-source build tool for Java project that centralizes and simplifies the build process.
- Configuring a Jenkins job involves setting it to build periodically using a Cron expression, which defines the desired schedule for the job's execution.
- Webhooks are real-time messaging mechanisms that enable one application to notify another about specific events.



# Sending Test Reports Using Email Notification

**Duration: 20 Min.**

**Project agenda:** To set up a Jenkins pipeline to compile code, run JUnit tests, and distribute test reports via email notifications

**Description:** Imagine a development team using Jenkins for automated email notifications of test reports, improving code quality and speed. Upon code commits, Jenkins triggers a pipeline that compiles, tests, and emails test results via an SMTP setup. This quick feedback loop is vital for agile practices, ensuring that developers immediately identify and address issues.



# Sending Test Reports Using Email Notification

Duration: 20 Min.

## Perform the following:

1. Create an app password for SMTP configuration
2. Configure SMTP configurations in Jenkins
3. Configure Maven in Jenkins

**Expected deliverables:** A fully configured Jenkins pipeline that compiles code, executes JUnit tests, and sends detailed test reports via email upon each build's completion







**Thank You**