DevOps

Caltech | Center for Technology & Management Education

simplilearn

Caltech | Center for Technology & Management Education

CI/CD Pipeline with Jenkins

simplilearn

Jenkins Pipeline

Learning Objectives

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

- Give an overview of Jenkins Pipeline
- List the benefits of Jenkins Pipeline
- Present the similarities and differences between Scripted and Declarative Jenkins Pipelines
- Discuss the functionalities of JenkinsFile
- Configure a Jenkins Pipeline job

Jenkins Pipeline

Jenkins Pipeline

- Jenkins Pipeline is a suite of sequenced and interlinked plugins. It supports Continuous Integration, Continuous Delivery, and Continuous Deployment.

Pipeline

Has an automation server for creating delivery pipelines as code

Uses JenkinsFile as the pipeline script to automate Builds, test cases, and deployment across applications

Note

Jenkins uses the Pipeline plugin to support JenkinsFile-based CI/CD pipelines.

Jenkins Pipeline

- The Jenkins Pipeline code is stored as text files in the source code repository.

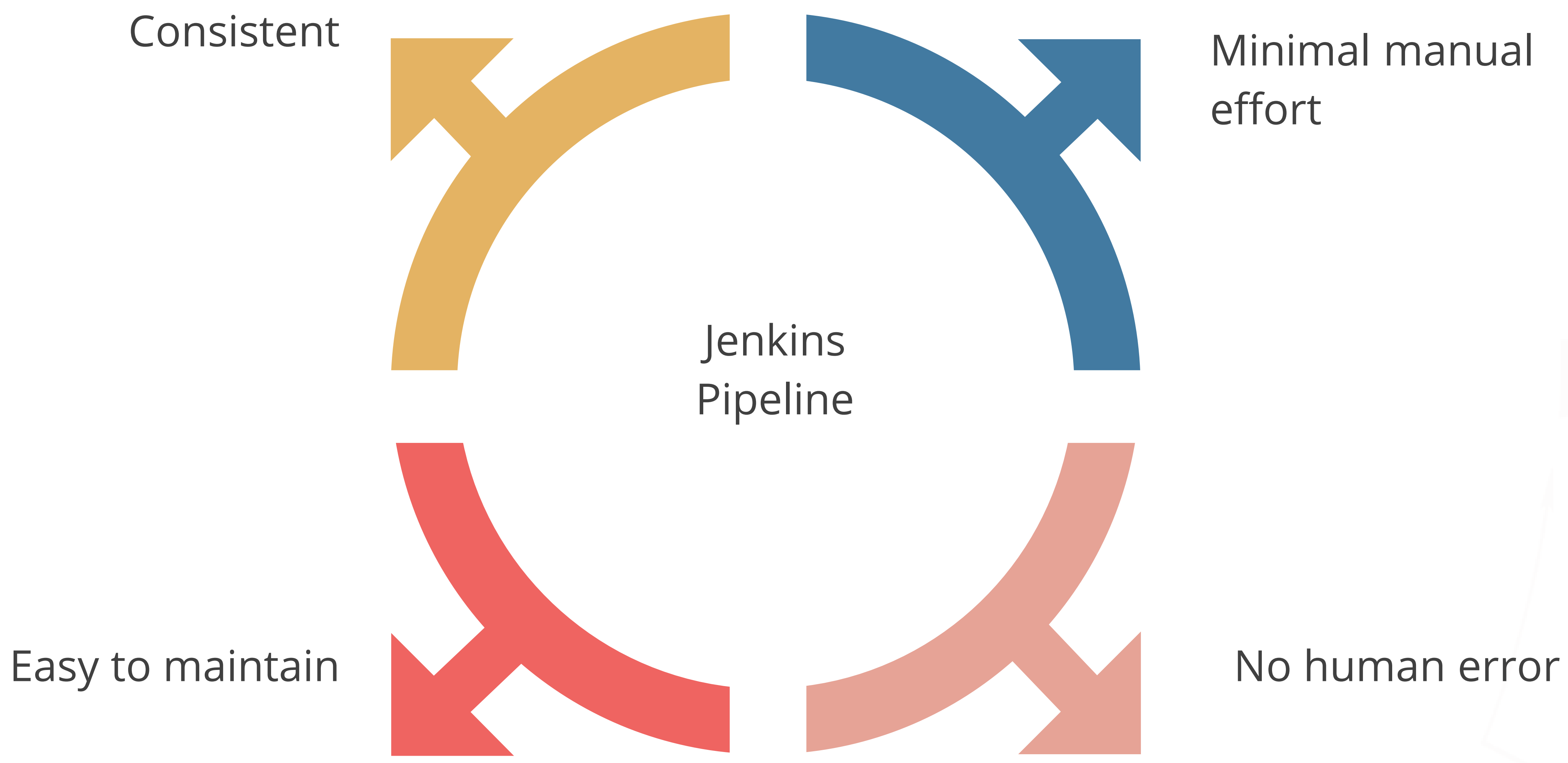
- Pipelines require one-time configuration; any further changes can be made using the pipeline script.
- Separate pipelines can be used for each branch in the source code repository.
- Pipelines are developed using Domain-Specific Language (DSL).

Note

Jenkins provides a Replay option that can be used by developers to troubleshoot any pipeline failures.

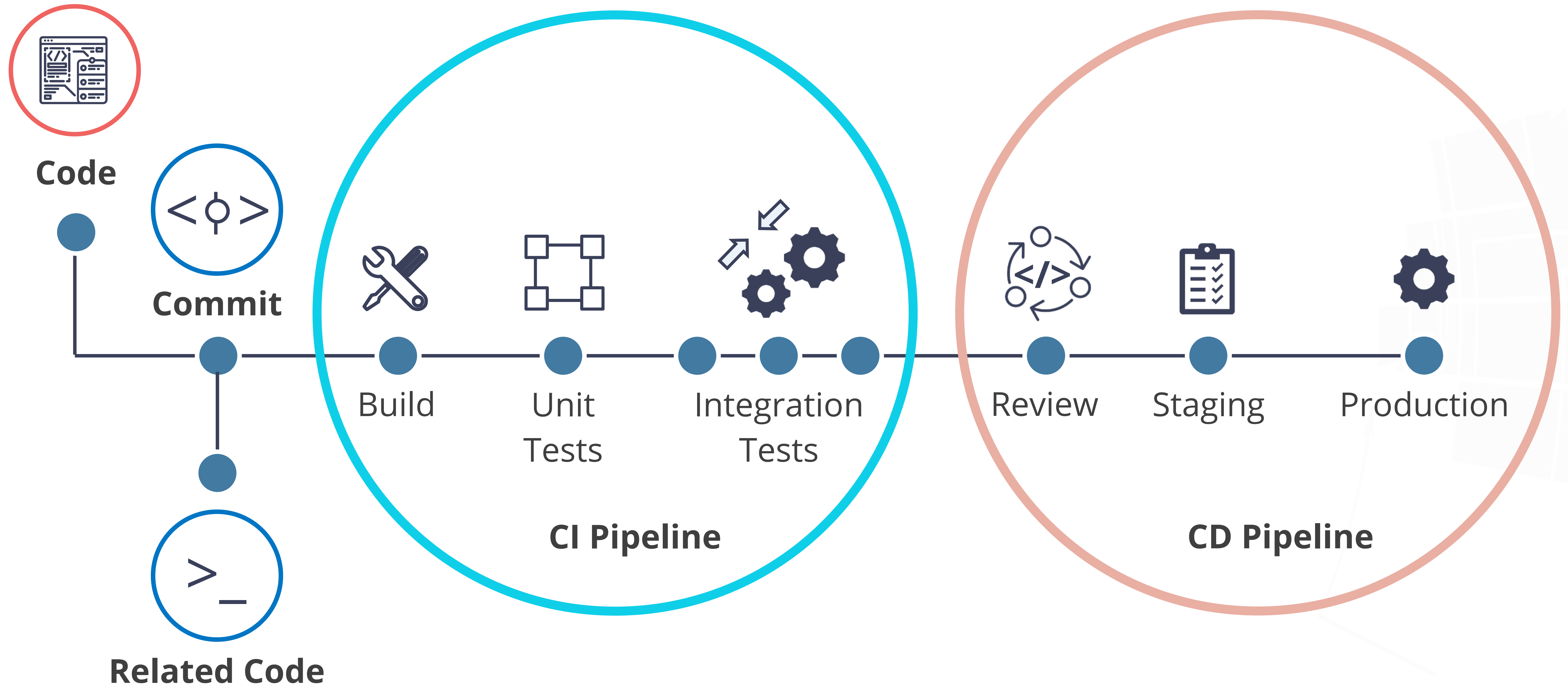
Benefits of Jenkins Pipeline

- Jenkins Pipeline offers the following benefits:



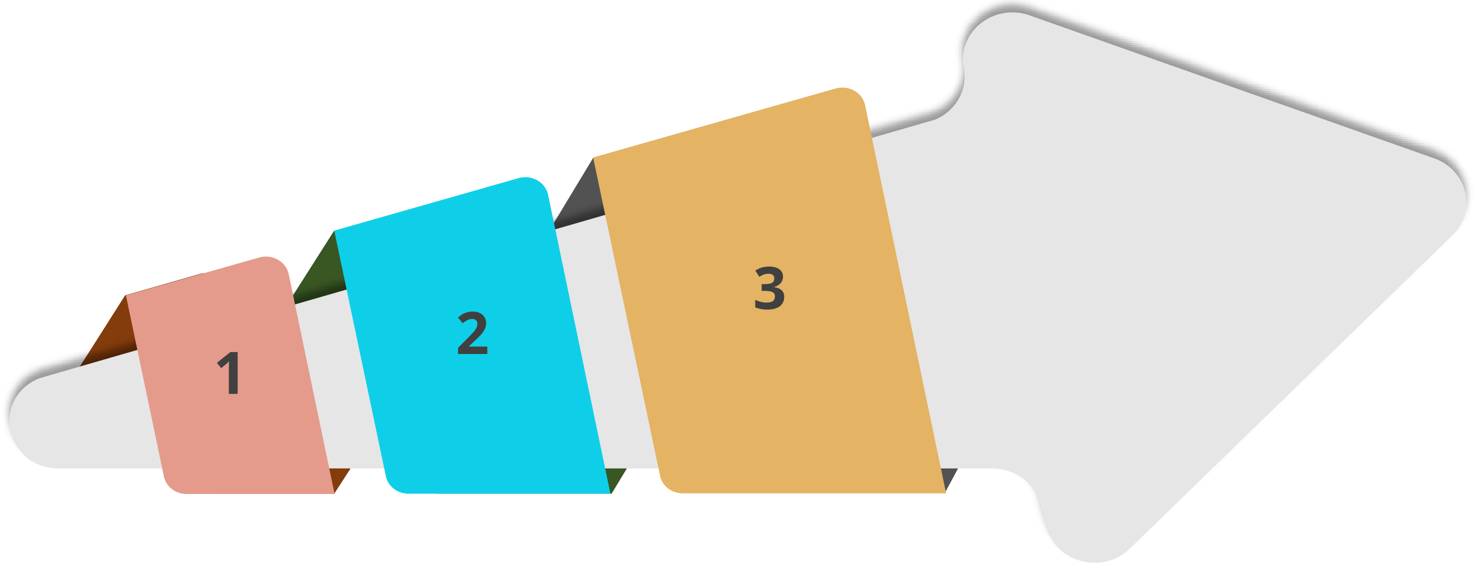
Jenkins CI/CD Pipeline

- A fully automated Jenkins CI/CD pipeline is shown below:



Process to Build a Jenkins CI/CD Pipeline

- To build a CI/CD pipeline with Jenkins, there is a three-step process involved:



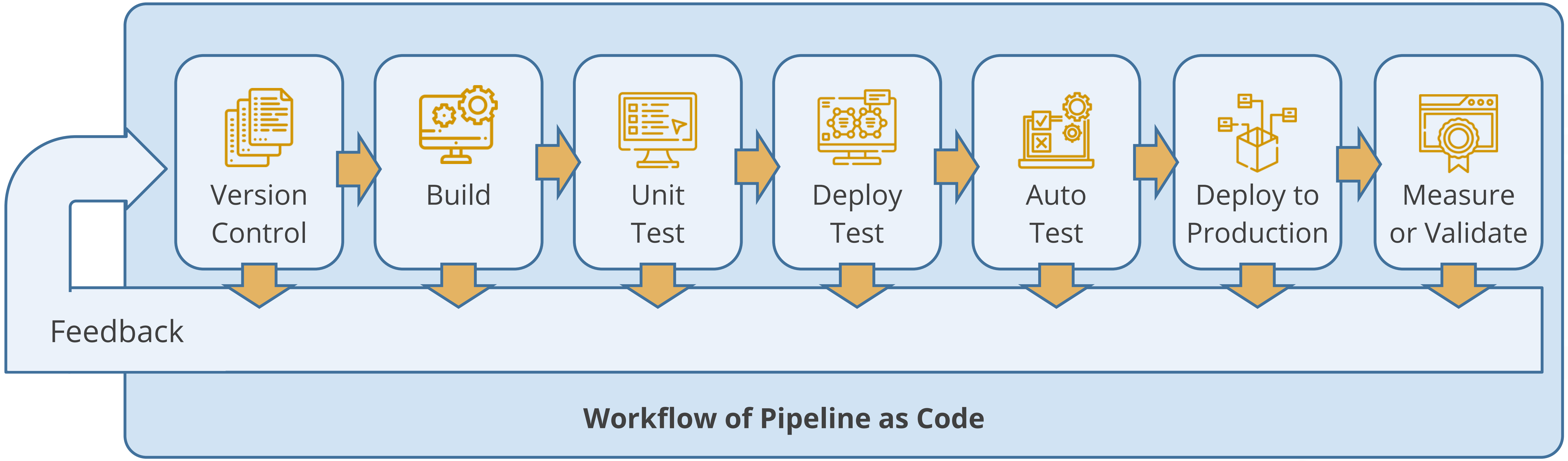
1. Download Jenkins and execute it as Java Binary

2. Create a Jenkins Pipeline job

3. Configure and execute the Pipeline job using a direct script or through source code management

Pipeline as Code

- Pipeline as Code allows developers to automate Builds and test executions, and deploy them using Groovy or YAML-specific pipeline scripts.



Workflow of Pipeline as Code

Pipeline as Code

- Pipeline as Code helps eliminate the need to create and manage jobs manually.

- It is stored in version control system.
- It enables Jenkins to discover, manage, and run jobs for multiple source repositories.
- It is a part of 'as code' movement that started with Infrastructure as code.

Jenkins Pipeline Job

- A Jenkins Pipeline job is used to create CI/CD automation within Jenkins. To configure Pipeline jobs, developers must first download the Pipeline plugin.

To download and use the Pipeline plugin in Jenkins:

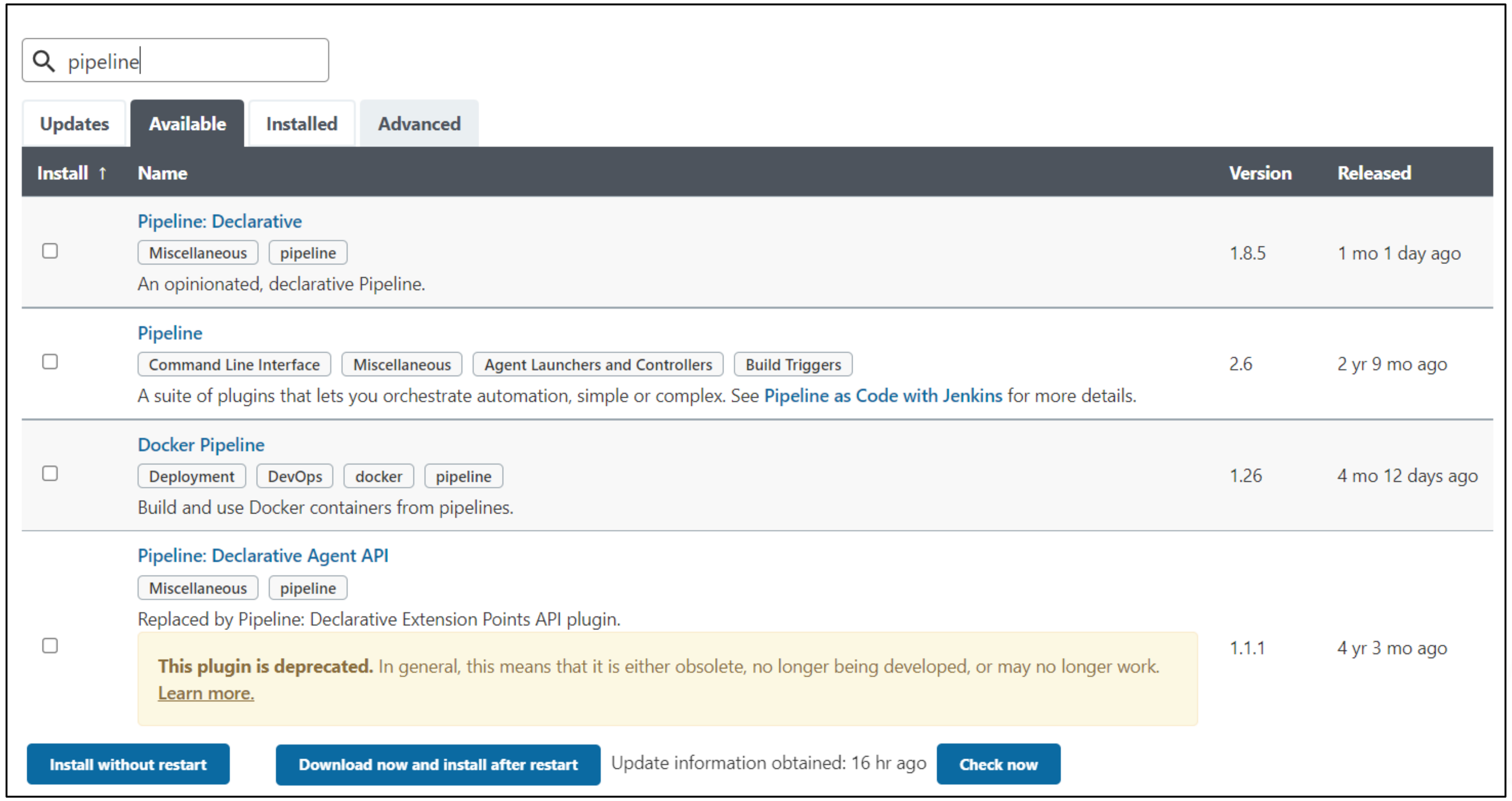
- Login to Jenkins
- Navigate to Manage Jenkins
- Access plugin manager

- Click Manage plugin
- Navigate to Available tab
- Search for Pipeline plugin

- Click Download Now
- Install the plugin after restart
- Click Restart to bounce Jenkins

Jenkins Pipeline Job

- This is a screenshot of the Jenkins dashboard, which lists all the available plugins:



pipeline

Updates | Available | Installed | Advanced

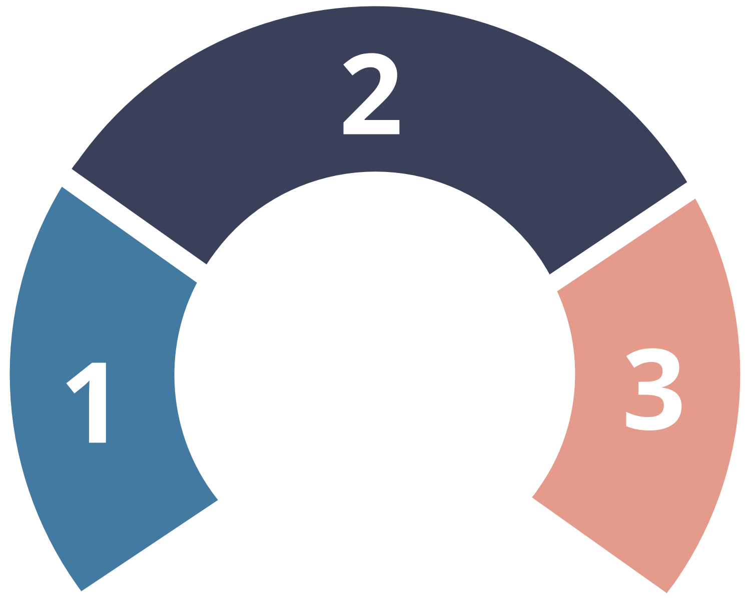
Install ↑	Name	Version	Released
☐	**Pipeline: Declarative** Miscellaneous · pipeline An opinionated, declarative Pipeline.	1.8.5	1 mo 1 day ago
☐	**Pipeline** Command Line Interface · Miscellaneous · Agent Launchers and Controllers · Build Triggers A suite of plugins that lets you orchestrate automation, simple or complex. See Pipeline as Code with Jenkins for more details.	2.6	2 yr 9 mo ago
☐	**Docker Pipeline** Deployment · DevOps · docker · pipeline Build and use Docker containers from pipelines.	1.26	4 mo 12 days ago
☐	**Pipeline: Declarative Agent API** Miscellaneous · pipeline Replaced by Pipeline: Declarative Extension Points API plugin. **This plugin is deprecated.** In general, this means that it is either obsolete, no longer being developed, or may no longer work. Learn more.	1.1.1	4 yr 3 mo ago

Install without restart | Download now and install after restart | Update information obtained: 16 hr ago | Check now

Jenkins Pipeline Job: Creation

- Jenkins supports creation of Pipeline and multi-branch Build jobs. To create a Pipeline job, follow the steps given below.

Enter the name of the job; click Pipeline Job type



Login to Jenkins; navigate to New Item

Configure the job created and click Save

Difference Between Freestyle and Pipeline Jobs

- The table below lists the differences between Jenkins Freestyle and Pipeline jobs:

Freestyle Jenkins Job	Pipeline Jenkins Job
Configurations stored inside Jenkins	Configurations stored in the code repository
Dependency on CI team to update configurations in Jenkins	No dependency on CI Team to update configurations
Difficult to manually handle multiple branches	Supports multi-branch pipelines that allow one-time configuration
CI/CD automation is a combination of multiple Jenkins jobs	CI/CD automation is flexible and easy; provides a stage view

Groovy Concepts

Groovy DSL

- Apache Groovy is an object-oriented programming language that provides a lot of features to orchestrate a Jenkins pipeline. Jenkins Pipeline job also provides an inline Groovy editor.

Groovy DSL code is written in Groovy language.

Compared to traditional programming languages, Groovy is easy to understand and manage.

It can interface seamlessly with Java.

Note

A Domain-Specific Language (DSL) is a programming language dedicated to a particular domain, or a particular problem/solution technique.

Blue Ocean Editor

- Blue Ocean Editor provides the simplest way to develop a Jenkins Pipeline. The editor provides an intuitive and visual process to help developers create a pipeline.

- Has an interface to add stages, steps, and commands that must be executed during the Build process
- Saves the pipeline contents to JenkinsFile in the repository once the changes are done
- Gets launched when the New Pipeline button is clicked

Jenkins Pipeline

- The Jenkins Pipeline execution supports two types of syntaxes, namely Scripted and Declarative pipelines.

Scripted Jenkins Pipeline

- Supports coding of the pipelines using Groovy DSL

Declarative Jenkins Pipeline

- Eliminates the necessity for Groovy DSL; provides a set of predefined structure and model for pipeline creation

Note

Scripted and Declarative pipelines use their own set of instructions and totally differ from each other from the perspective of syntax.

Scripted Jenkins Pipeline

- Scripted pipeline is a traditional way of developing Jenkins pipelines. It supports configuration of large and complex pipelines. To create a scripted pipeline, developers must start with the node elements within the node block.

It gets executed on the Jenkins Master.

It does not provide a formal structure. It is, thus, difficult to process and interpret pipeline functionalities.

It uses less resources to transform pipeline instructions into atomic commands.

It follows Groovy DSL rules strictly.

Scripted Jenkins Pipeline

- The following code snippet is an example of a Scripted Jenkins pipeline:

Demo-1

```
node('master') {
stage ('checkout code'){
checkout scm
}
stage ('build'){
sh "${MVNHOME}/bin/mvn clean install"
}
stage ('Test Cases Execution'){
sh “mvn clean test"
}
stage ('Production Deployment'){
sh 'cp target/*.war /opt/tomcat8/webapps'
}
}
```

Declarative Jenkins Pipeline

- Declarative Pipelines are relatively easy to develop and manage. Developers have complete control over all the aspects of pipeline execution.

- It contains a predefined hierarchy for creating pipelines.
- It offers a simple and straight-forward syntax.
- It facilitates clean, simple, and smooth integration with the Blue Ocean GUI component.
- It does not support some of the Jenkins plugins.

Declarative Jenkins Pipeline

- The following code snippet is an example of a Declarative Jenkins Pipeline:

Demo-1

```
pipeline {
    agent any
    stages {
        stage("Build") {
            steps {
                sh "${MVNHOME}/bin/mvn
clean install"
            }
        }
        stage("Test") {
            steps {
                sh "mvn clean test"
            }
        }
    }
}
```

Scripted vs. Declarative Jenkins Pipeline

- The table below lists the similarities and differences between Scripted and Declarative Jenkins pipelines:

Scripted Jenkins Pipeline	Declarative Jenkins Pipeline
Is a traditional approach to developing pipelines	Is a newer approach to developing pipelines
Follows a strict Groovy DSL syntax	Provides flexibility during pipeline integration
Does not support integration with Blue Ocean GUI	Supports integration with Blue Ocean GUI
Does not support restart from a specific stage	Supports restart from a specific stage

Jenkins Pipeline: Keywords

Pipeline	A set of instructions (defined as source code) for implementing CI and CD. It helps configure Build, test, and deployment automations.
Agent	Specifies the server or the machine where the Build execution will take place. This configuration defines a label with which an agent is deployed in Jenkins Master.
Stage	Used to configure similar pipeline processes. It comprises one or more Build steps.
Trigger	Defines automated ways to trigger a pipeline. It uses a cron job for time-based scheduling and polls SCM to check the repository for changes.

Jenkins Pipeline: Keywords

Options	Specify pipeline-specific options including timeout or retry
Environment	Defines a set of key-values used as environment variables during a Build
Parameters	Define a set of user-input parameters
Step	Is an individual task that gets executed as part of Jenkins pipeline

Jenkins Pipeline: Keywords

When	Is used to control the execution of a stage depending on a specific condition
sh	Is used to execute shell commands and shell scripts; also used to configure custom shell commands and scripts
Script	Executes a block of Groovy-based code that can be used for non-trivial scenarios that require flow control

JenkinsFile Pipeline

JenkinsFile Basics

JenkinsFile helps implement the Pipeline as a code functionality in Jenkins.

A Pipeline is a combination of a series of Build steps executed all together in a sequence.

- Stored within a version control system with source code
- Written in Groovy language

JenkinsFile Pipeline Job Configuration

To configure a Pipeline project, developers must configure the Pipeline and General and Build sections of JenkinsFile.

General and Build Triggers Section

Similar to Freestyle job configuration

Pipeline Section

Pipeline code or Git repository can be provided

Pros and Cons of Configuring Pipeline Code in Pipeline Job

The benefits and drawbacks of configuring Pipeline code in a Pipeline job are:

Benefits:

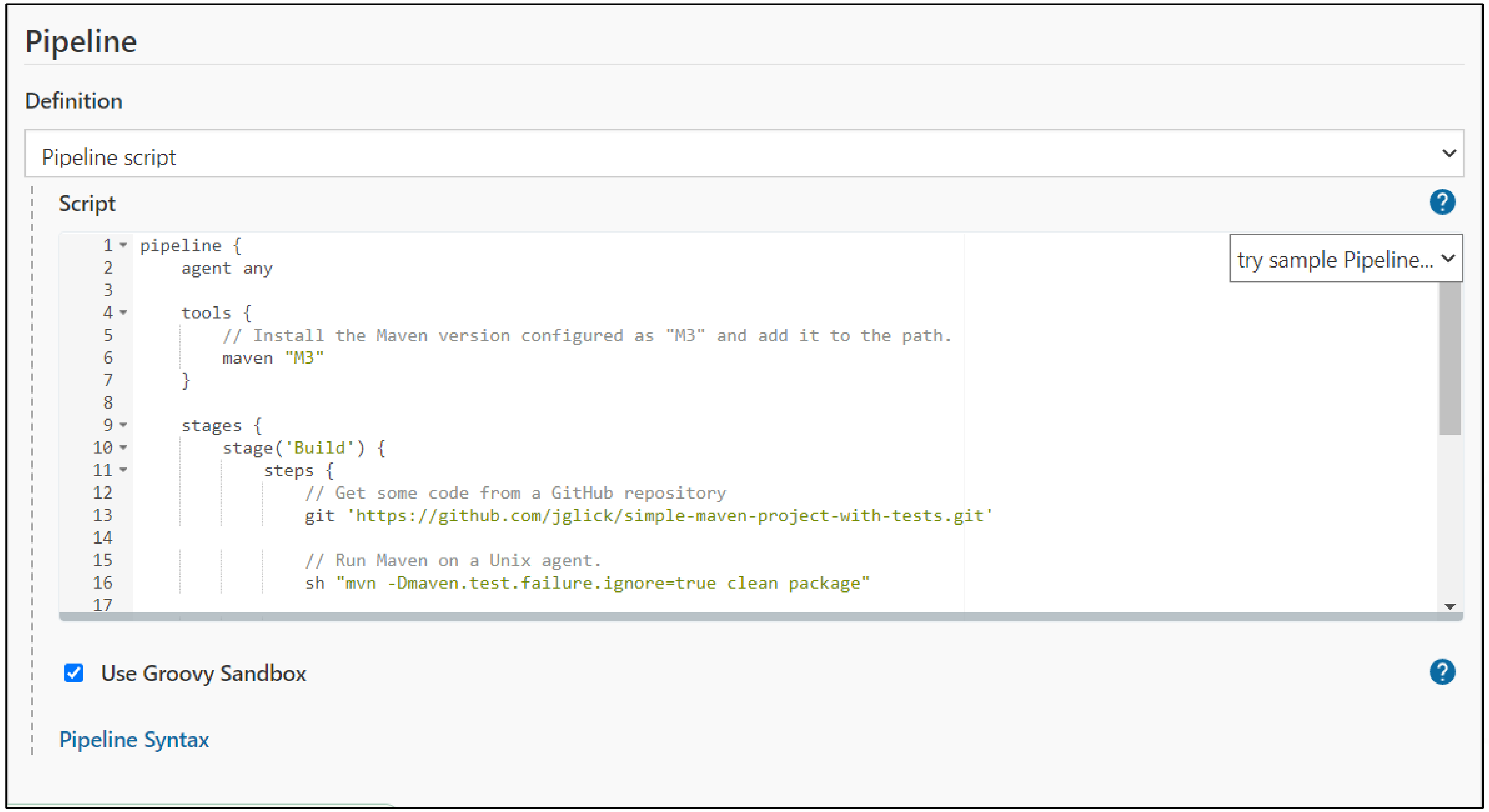
- The Pipeline code can be stored directly in Jenkins job.
- The Pipeline code can be changed.

Drawback:

- A login to Jenkins is required each time the Pipeline contents need to be modified.

Pipeline Code in Pipeline Job: Configuring

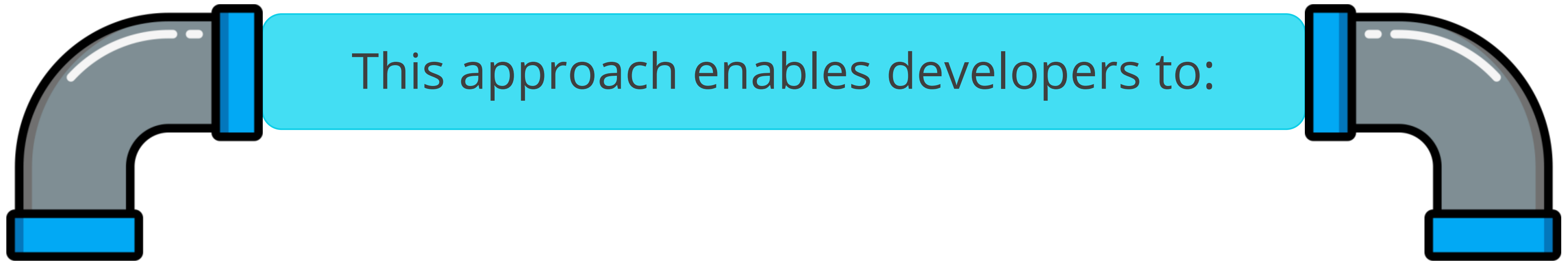
The Pipeline script specifies the steps needed to execute the job. Shown below is a screenshot of a sample Pipeline script:



```
pipeline {
    agent any

    tools {
        // Install the Maven version configured as "M3" and add it to the path.
        maven "M3"
    }

    stages {
        stage('Build') {
            steps {
                // Get some code from a GitHub repository
                git 'https://github.com/jglick/simple-maven-project-with-tests.git'

                // Run Maven on a Unix agent.
                sh "mvn -Dmaven.test.failure.ignore=true clean package"

```

Configure JenkinsFile Pipeline from Git Version Control System

This is a better option as compared to configuring the pipeline code in Jenkins itself.



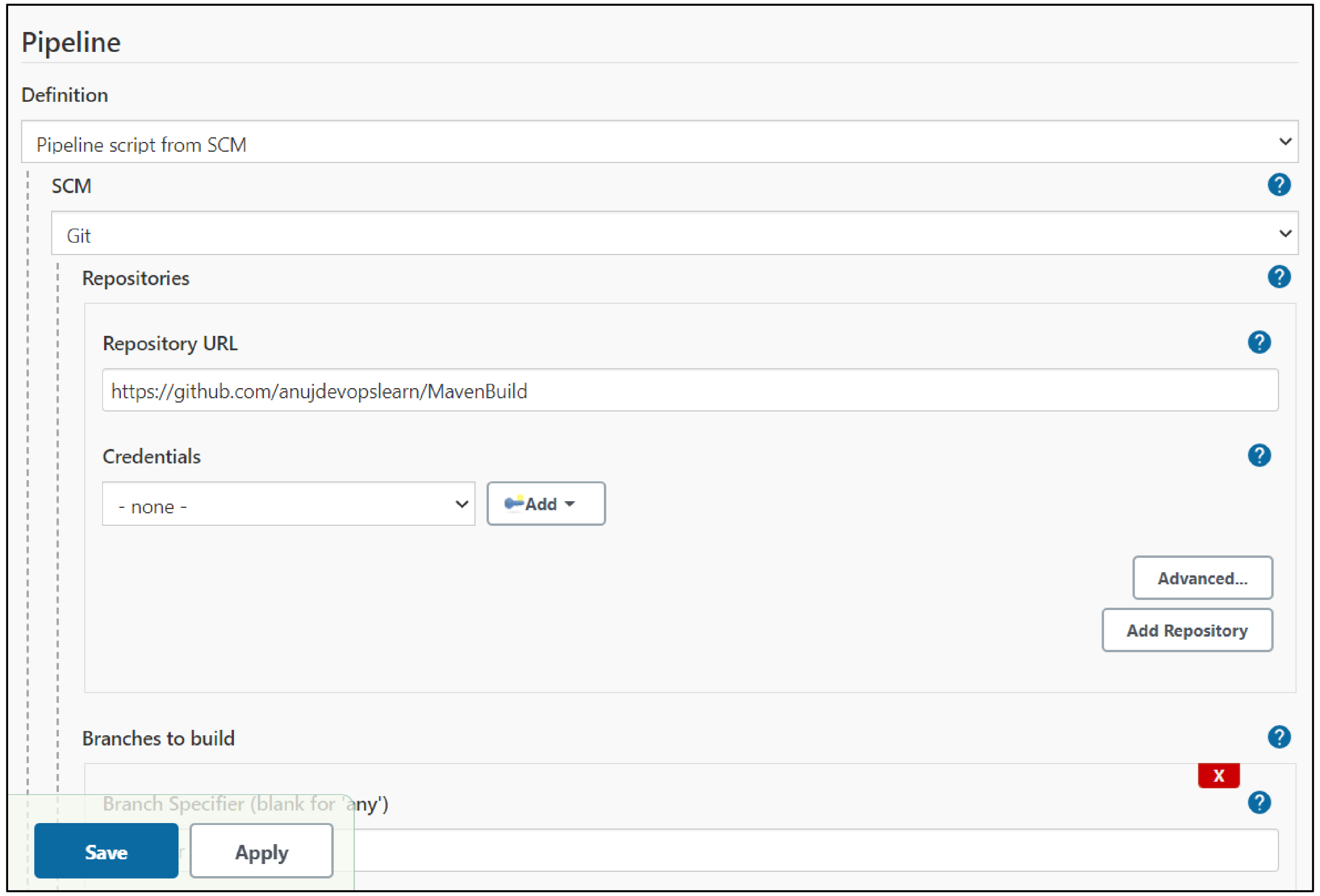
This approach enables developers to:

Configure source control tool

Control Pipeline changes from Git repository

Configure JenkinsFile Pipeline from Git Version Control System

A fundamental step of CI/CD implementation is integrating the SCM tool with the CI tool. The screenshot below represents how this can be done:



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/anujdevopslearn/MavenBuild

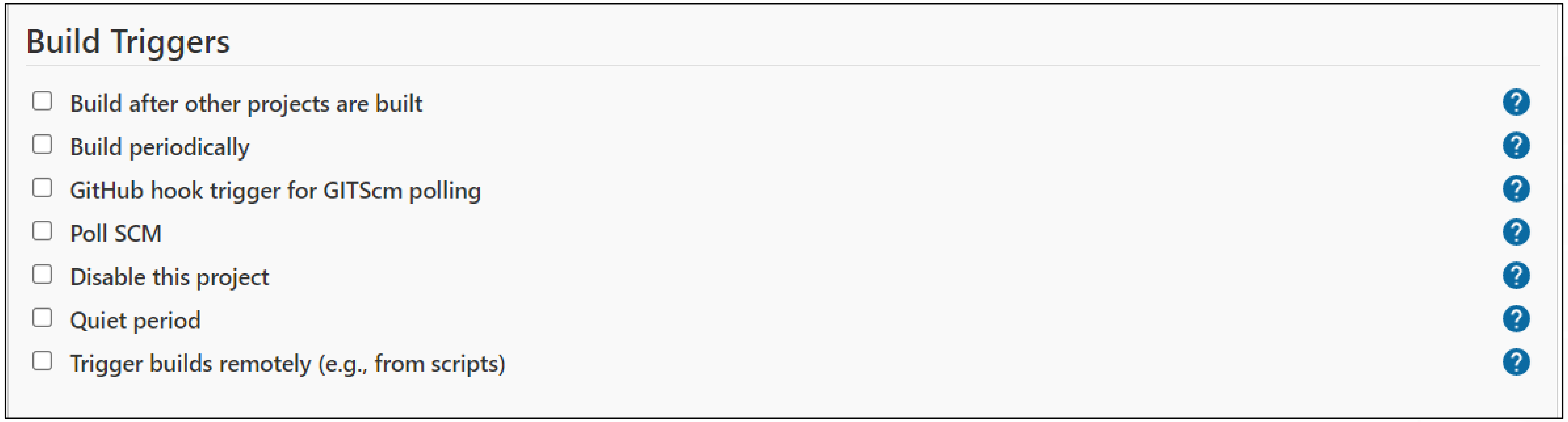
Credentials: - none -

Branches to build

Branch Specifier (blank for 'any')

Pipeline Build Triggers

While configuring Jenkins job, the following triggers can be configured:



Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM
- ☐ Disable this project
- ☐ Quiet period
- ☐ Trigger builds remotely (e.g., from scripts)

Pipeline Build Triggers

Some of the uses of Pipeline Build triggers are:

Polling can be performed to the version control system.

Automated Builds can be triggered by configuring a web hook.

Jenkins job can be scheduled using crontab like syntax.

The upstream Jenkins job can be configured in the original job.

Jenkins Builds can be run remotely using shell command and REST API.

Assisted Practice

Set up Jenkins Pipeline Job from Git Version Control System

Duration: 20 min

Problem Statement:

Set up a Pipeline job on Jenkins from the Git version control system.

Assisted Practice: Guidelines

Steps to set up a Pipeline job on Jenkins from the Git version control system:

1. Login to Jenkins CI tool with admin user and click on New Item to create a new Jenkins's job.
2. Select Pipeline project while creating Jenkins's job and provide a custom job name.
3. Click on Ok button to save the Jenkins's job and a new configuration screen would be opened.
4. Access the Pipeline tab to provide Git repository configurations from where you want to checkout pipeline.
5. In case you are using a different Pipeline script you can configure Script path in Pipeline project.
6. Once the Git repository configurations are configured in Jenkins's job, you can trigger Jenkins's job build.

Jenkins File Advanced

Jenkins Pipeline Variables

By using Pipeline variables, hardcoding can be avoided, and sensitive information can be moved around the entire Pipeline script.

- Boolean variables
- Environment variables
- Custom variables

Variables in Pipeline scripts are defined using the keyword “def”.

Example: def variable = “value"

Environment Variables

Jenkins environment variables are a set of variables exposed by Jenkins through env variable.

Using env variable, any variable can be referred anywhere within the Pipeline.

These variables hold string values and can be easily initialized at the top, or within a stage or script block.

Jenkins environment variables can be accessed using <JENKINS_URL>/env-vars.html

Environment Variables

These variables are available to shell scripts:

- BRANCH_NAME
- BRANCH_IS_PRIMARY
- CHANGE_ID
- CHANGE_URL
- CHANGE_TITLE
- CHANGE_AUTHOR
- CHANGE_AUTHOR_DISPLAY_NAME
- CHANGE_AUTHOR_EMAIL
- CHANGE_TARGET
- CHANGE_BRANCH

Variables: Defining

Scripted Pipeline

```
node{
   stage(Deployment') {
       withEnv([“PERFORM_DEPLOY=true"]) {
       }
   }
 }
```

Variables: Defining

Declarative Pipeline

```
environment {
        PERFORM_DEPLOY = 'true'
}
```

Variables: Fetching

Scripted Pipeline

```
node{
   stage('Build') {
       withEnv([" PERFORM_DEPLOY=true"]) {
         echo env. PERFORM_DEPLOY
       }
   }
 }
```

Variables: Fetching

Declarative Pipeline

Declarative Pipeline

```
Pipeline {
   agent any
   environment {
       PERFORM_DEPLOY= 'true’
    }
   stages {
       stage(“Build”) {
           steps {
               echo env. PERFORM_DEPLOY
           }
       }
   }
}
```

Global Environment Variables Configuration

Global environment variables can be used in any Jenkins Pipeline job. They can be configured within the Jenkins console.

Step 1

Login to Jenkins and access Manage Jenkins.

Step 2

In Manage Jenkins, click on Configure system and then go to the Global properties section.

Step 3

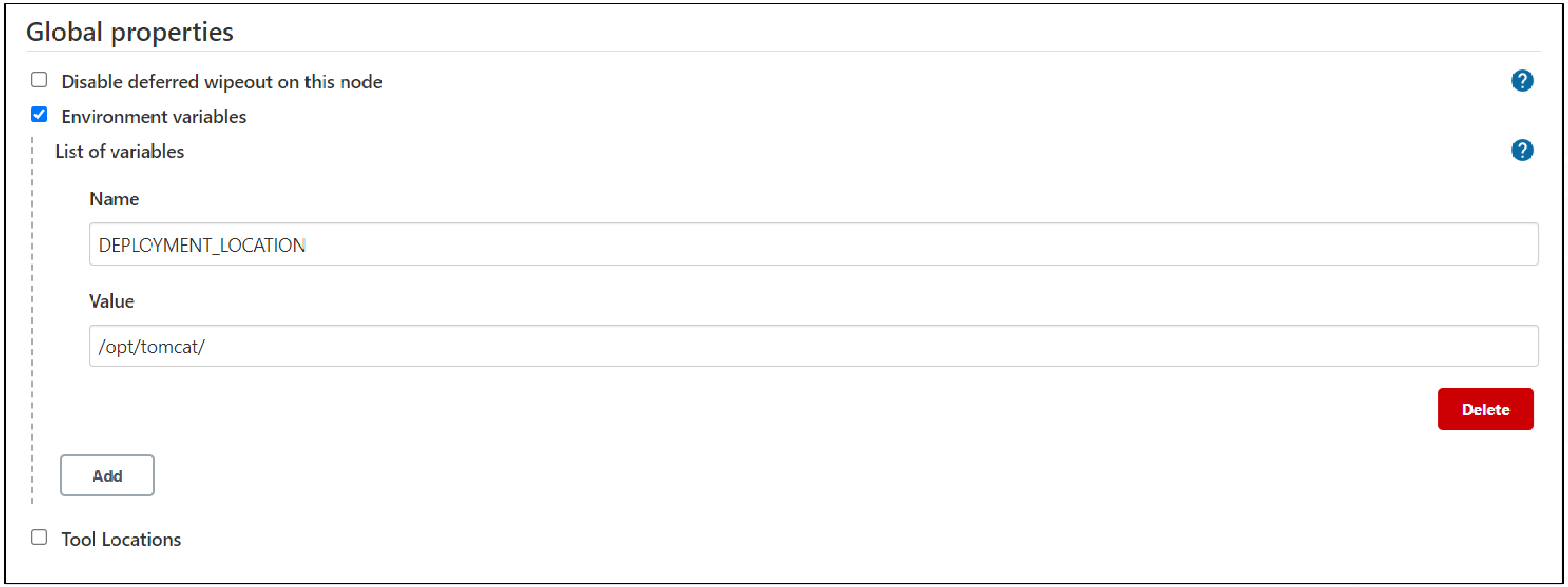
In the Global properties section, tick the box for environment variables.

Step 4

Click on Add. Add values for variable Name and Value.

Global Environment Variables: Defining

The screenshot below shows the Global properties tab of the Jenkins dashboard used for setting environment variables:



Global properties

- [ ] Disable deferred wipeout on this node
- [x] Environment variables

List of variables

Name: DEPLOYMENT_LOCATION

Value: /opt/tomcat/

- [ ] Tool Locations

Jenkins Credentials

The Jenkins credentials plugin provides a default internal credentials store for storing sensitive information.

- All the passwords and configurations used within Jenkins are primarily stored in Jenkins credentials.
- Both freestyle and Pipeline-based projects have this component.
- However, there are some vulnerabilities or hacks that make this mechanism unsafe.

Jenkins Credentials

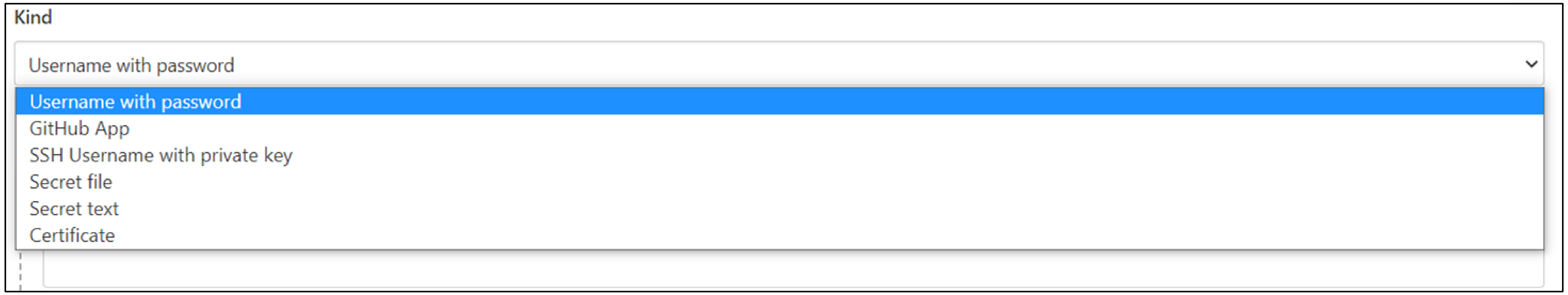
Follow these steps to create or manage Jenkins credentials:

Step 1

Login to Jenkins and navigate to Manage Jenkins. Manage credentials to Manage Jenkins credentials.

Step 2

Click on Global domain. It will move to a screen where you can Add or Manage credentials.



Kind

Username with password

- Username with password
- GitHub App
- SSH Username with private key
- Secret file
- Secret text
- Certificate

Jenkins Credentials Scope

Jenkins credentials supports two types of credentials:

Global Scope

- Default scope
- Can be accessed by all jobs
- Primarily used for storing VCS credentials, AWS credentials, Secret token, and Secret text

System Scope

- Only used internally by Jenkins system
- Not available to any other Jenkins job
- Restricts the usage of credentials and provides a high degree of confidentiality

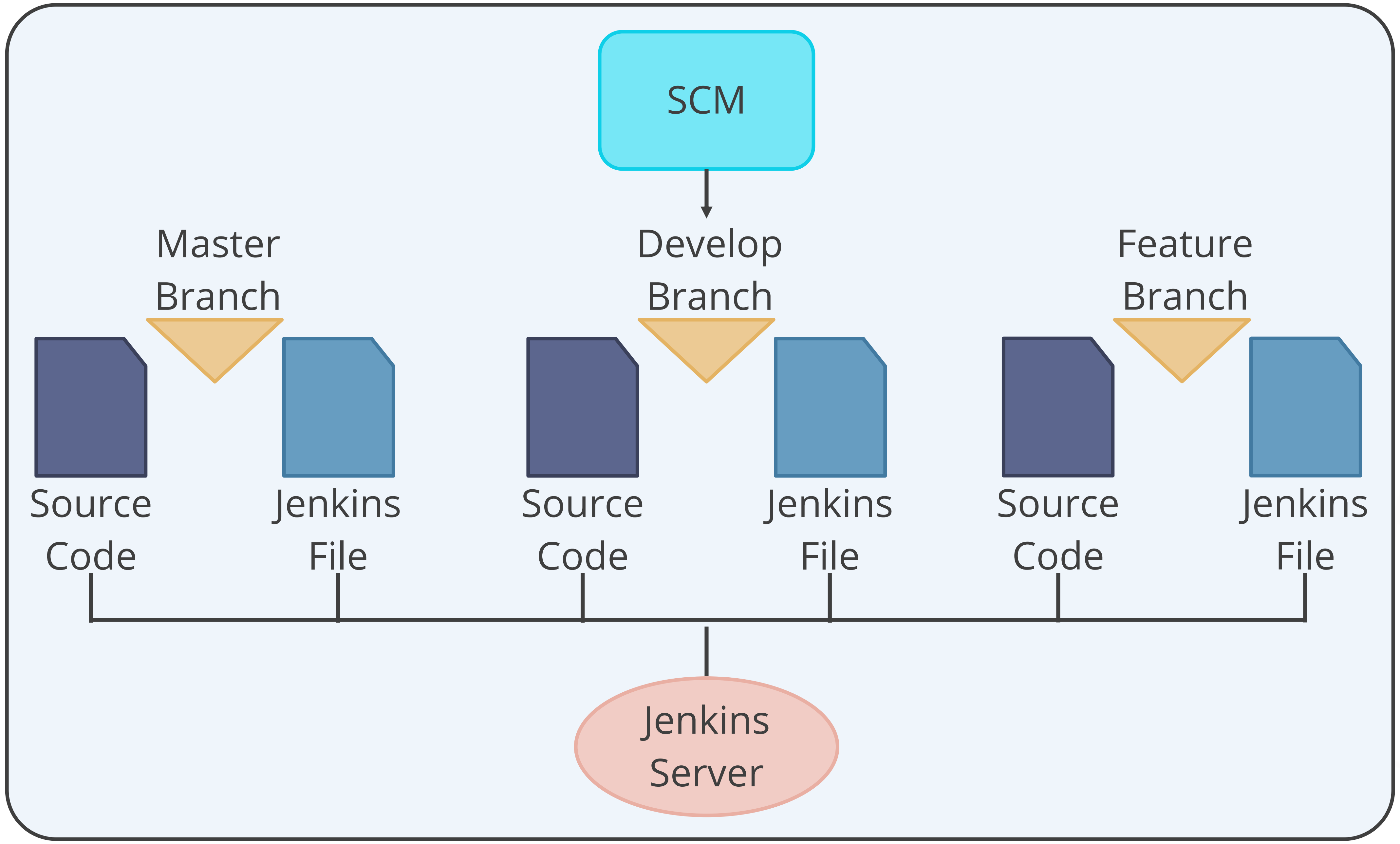
Multibranch Pipeline

The Multibranch Pipeline is an extension to the Pipeline job in Jenkins.

- Helps developers to configure multiple branch configurations for a single project
- Automatically scans for new branches
- Allows developers to manage environment-specific configurations inside Jenkins Pipeline scripts

Multibranch Pipeline: Example

The SCM repository has three branches, each having a source code along with the Jenkins file.



Multibranch Pipeline

Multibranch Pipeline

Allows to automatically create a Pipeline for each branch of the SCM repository, with the help of JenkinsFile

Enables implementing different JenkinsFile Pipeline scripts for different branches of the same project

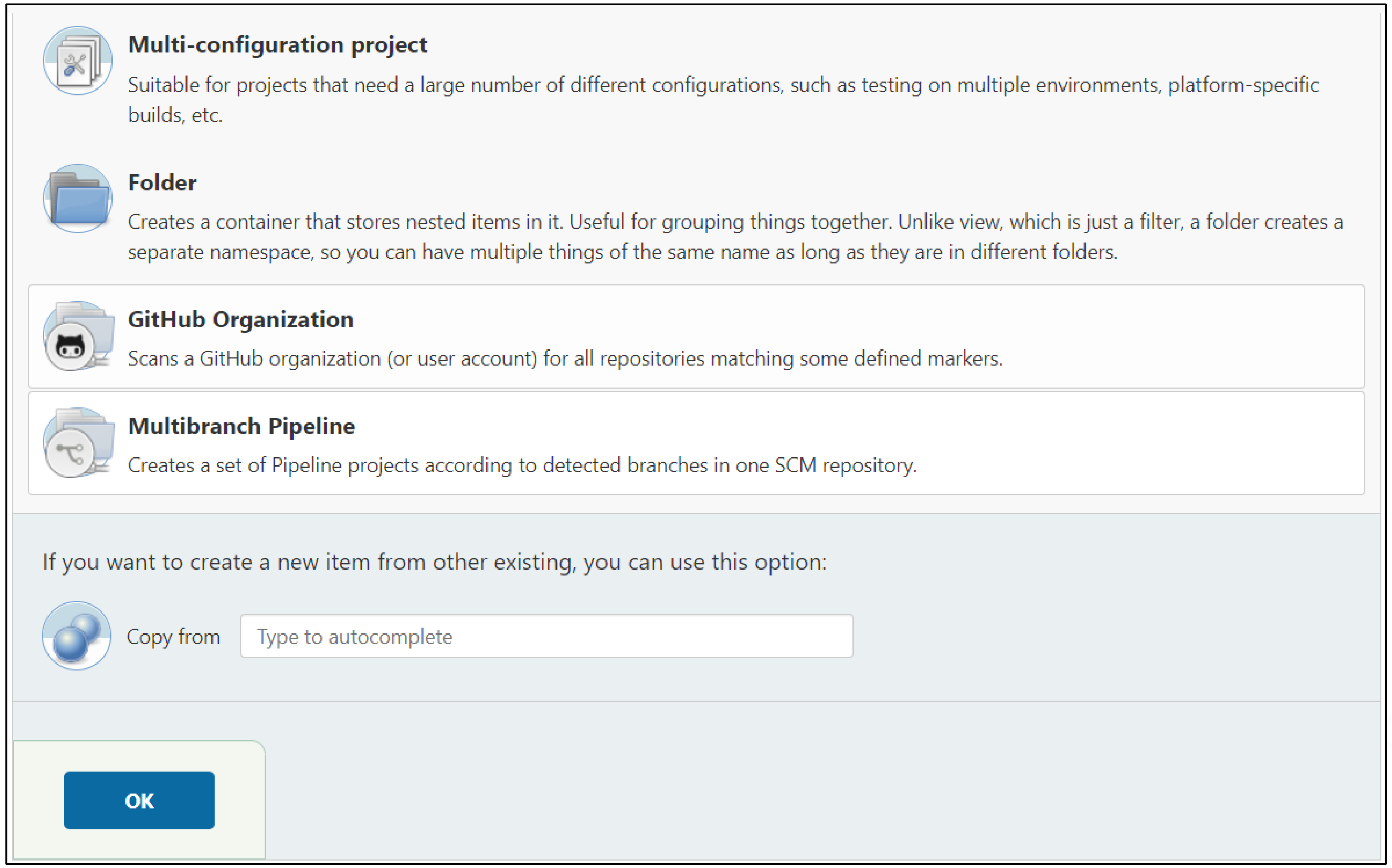
Multibranch Pipeline Job

Branch sources constitute one of the most important configuration sections. When a Multibranch Pipeline job is executed, the following actions happen:

- ✓ The version control system gets configured with all the branches.
- ✓ The job will scan for all the branches that have JenkinsFile inside it.
- ✓ In the event of branch creation or deletion, the job will create or delete the project in Jenkins.
- ✓ Jobs will automatically get removed if a branch is removed from the repository.

Multibranch Pipeline Job

When New Item is selected on the home page, a list of different project types will appear. Select Multibranch Pipeline as shown below:



Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

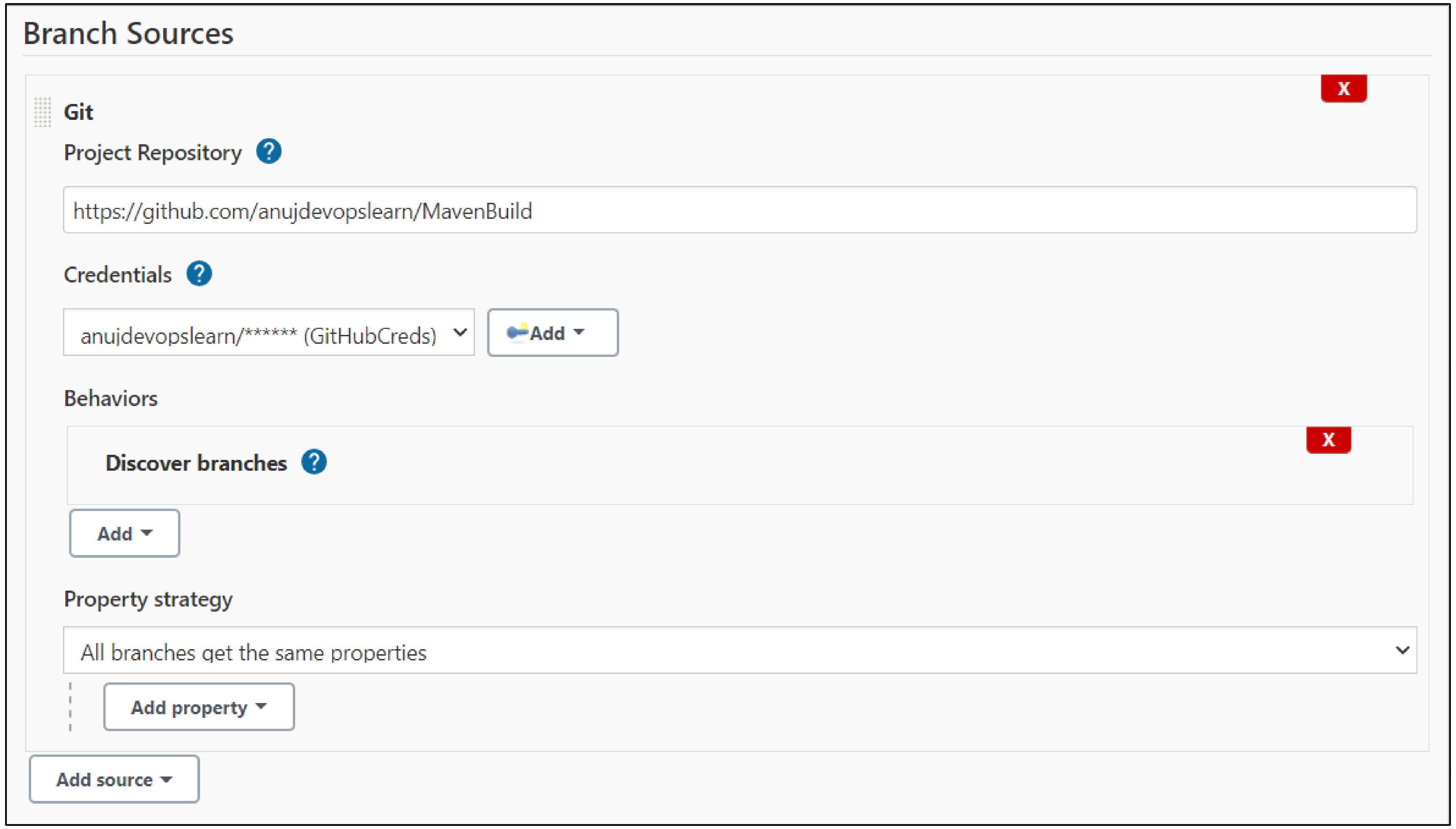
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Copy from Type to autocomplete

Multibranch Pipeline Job

Add a Branch Source (for example, Git) and enter the location of the repository, as shown below:



Branch Sources

Git

Project Repository: https://github.com/anujdevopslearn/MavenBuild

Credentials: anujdevopslearn/****** (GitHubCreds)

Behaviors

Discover branches

Property strategy: All branches get the same properties

Caltech Center for Technology & Management Education

simplilearn

Multibranch Pipeline Job

On completion of SCM configurations, repository scanning starts. A new Pipeline project gets created automatically for each branch.

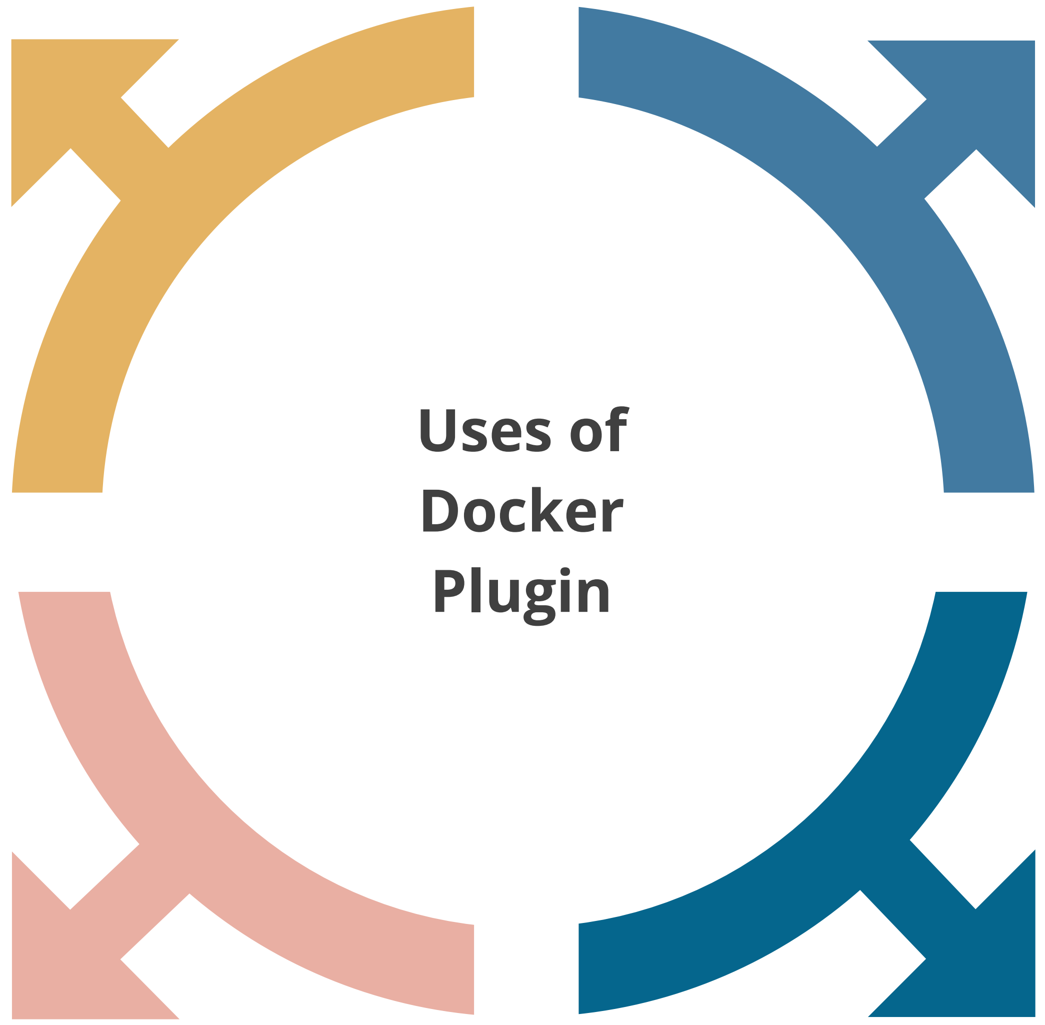
Creation of new Pipeline projects occurs only when Jenkins can read the JenkinsFiles stored in these branches.

If any modifications are applied to a branch, then the Build gets triggered only for that specific branch project.

Scanning should be configured frequently so that Jenkins can process the creation or deletion of new branches from the version control system.

Docker Plugin for Jenkins Pipeline

This plugin supports the Pipeline setup to use Docker images as an execution environment.



Automating the preparation of Docker image

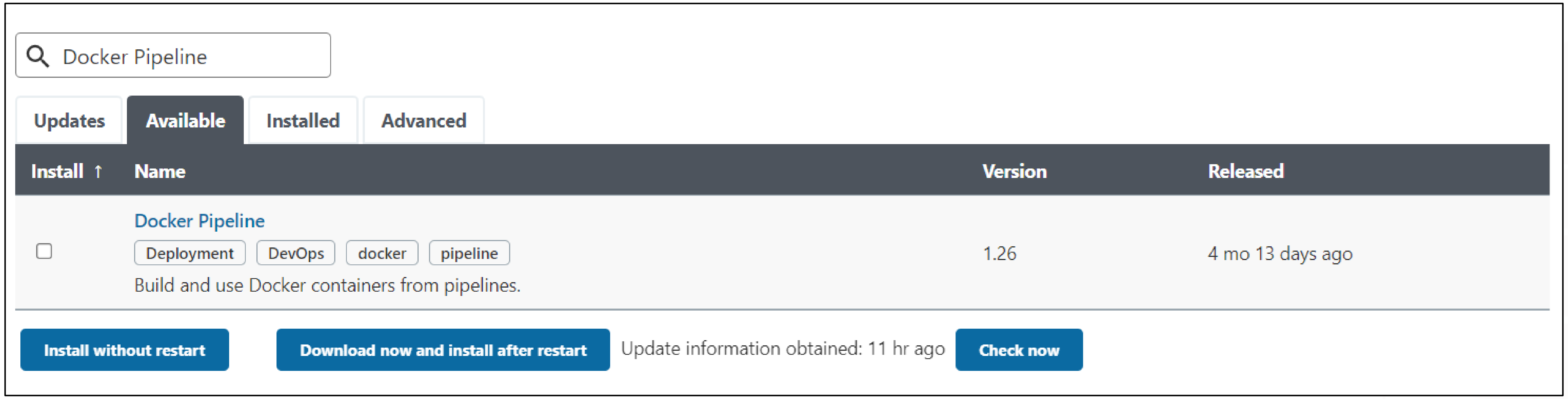
Pushing Docker images to Docker hub

Deploying Docker containers using images

Triggering Build processes inside a Docker container

Docker Plugin for Jenkins Pipeline

Containers can be dynamically provisioned as Jenkins nodes using Docker. The Docker plugin can be downloaded from the Available tab in the Jenkins dashboard.



Install ↑	Name	Version	Released
☐	Docker Pipeline Deployment DevOps docker pipeline Build and use Docker containers from pipelines.	1.26	4 mo 13 days ago

Update information obtained: 11 hr ago

Dockerized Jenkins Pipeline

Dockerized Jenkins Pipeline

```
Pipeline {
    agent {
        docker { image 'node:14-alpine' }
    }
    stages {
        stage('Test') {
            steps {
                sh 'node --version'
            }
        }
    }
}
```

Assisted Practice

Configure Multibranch Jenkins Pipeline Job

Duration: 20 min

Problem Statement:

Configure a Multibranch Pipeline job in Jenkins.

Assisted Practice: Guidelines

Steps to configure a Multibranch Pipeline job in Jenkins:

1. Login to Jenkins CI tool with admin user and click on New Item to create a new Jenkins's job.
2. Select Multi Branch Pipeline project while creating Jenkins's job and provide custom job name.
3. Click on the Ok button to save the Jenkins's job and a new configuration screen would be opened.
4. Access the Pipeline tab to provide Git repository configurations from where you want to checkout pipeline.
5. In case you are using a different Pipeline script, you can configure Script path in the Pipeline project.
6. If you want to periodically scan for branches, you can configure Scan Multibranch Pipeline Triggers.
7. Once the Git repository configurations are configured in Jenkins's job, you can trigger Jenkins's job build.

Assisted Practice

Deploy a Docker Container with Jenkins Pipelines **Duration: 30 min**

Problem Statement:

Deploy a Docker container with Jenkins Pipelines.

Assisted Practice: Guidelines

Steps to Deploy a Docker container with Jenkins Pipelines:

1. Log in to Jenkins CI tool and install Docker and Docker Pipeline Plugin.
2. Navigate to Global Tool Configuration to add Maven and Docker Tool Configuration.
3. Configure Jenkins pipeline and execute build for Docker pipeline

Key Takeaways

- Jenkins Pipeline is one of the latest features offered by Jenkins, using which Continuous Integration, Continuous Delivery and Continuous Deployment can be implemented.
- Jenkins Pipeline scripts are written in Groovy language, which helps in modifying Build configurations as required, even without access to Jenkins.
- JenkinsFile helps to implement the Pipeline as a code functionality in Jenkins.
- A Multibranch Pipeline is an extension to the Pipeline job that implements different Jenkins file Pipeline scripts for different branches of the same project.

Implement a CI/CD Pipeline to Deploy an Application

Lesson-End Project

Problem Statement:

Perform the following:

- Implement a completed CI/CD Pipeline
- Deploy an application to Tomcat Apache

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Thank You