Training Course

Jenkins

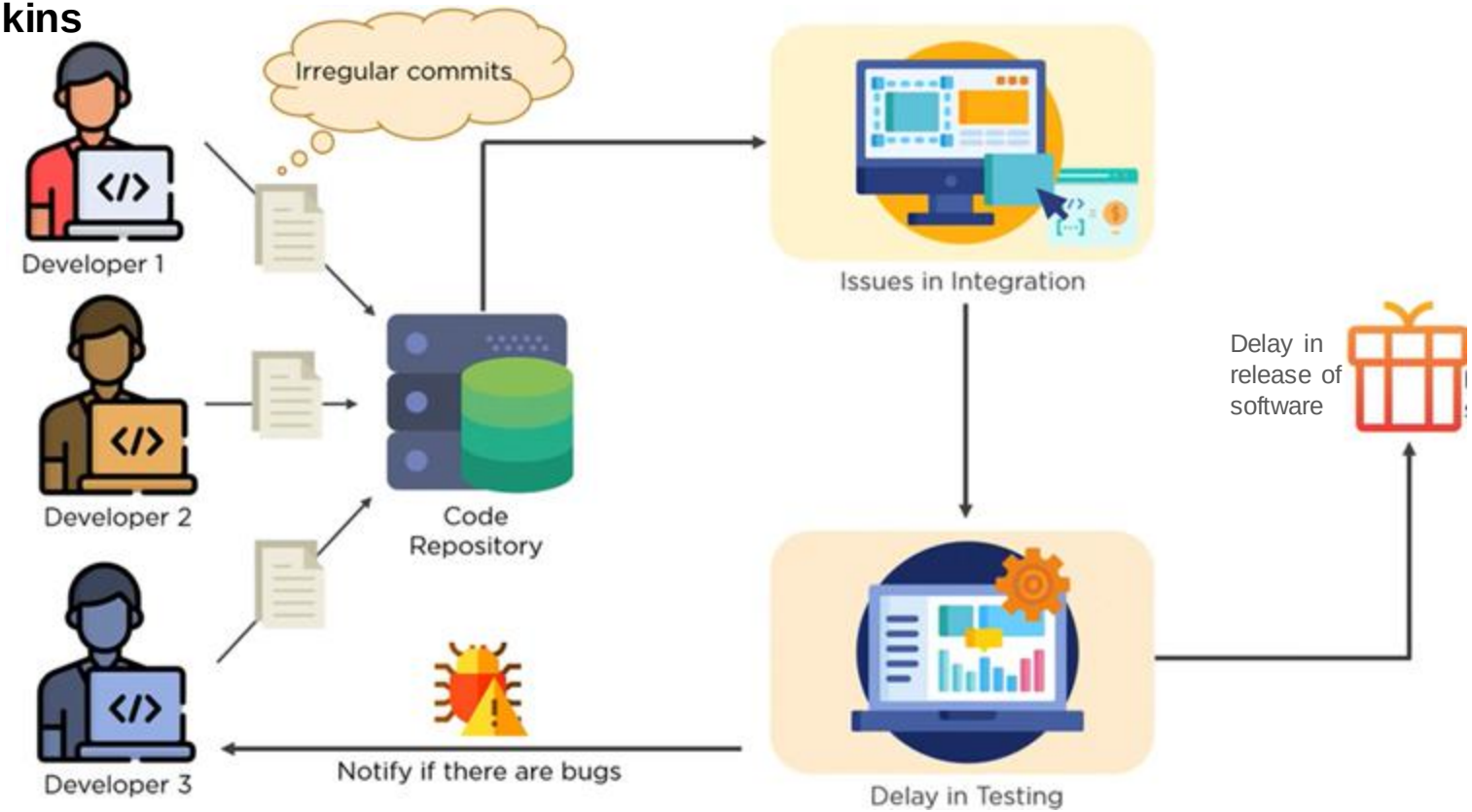
Course Overview

- Jenkins Introduction
- Jenkins credentials
- Jenkins pipeline
- Jenkins pipeline syntax
- Jenkins plugins
- Jenkins Master and Slave
- Lab

Jenkins Introduction

Jenkins Introduction

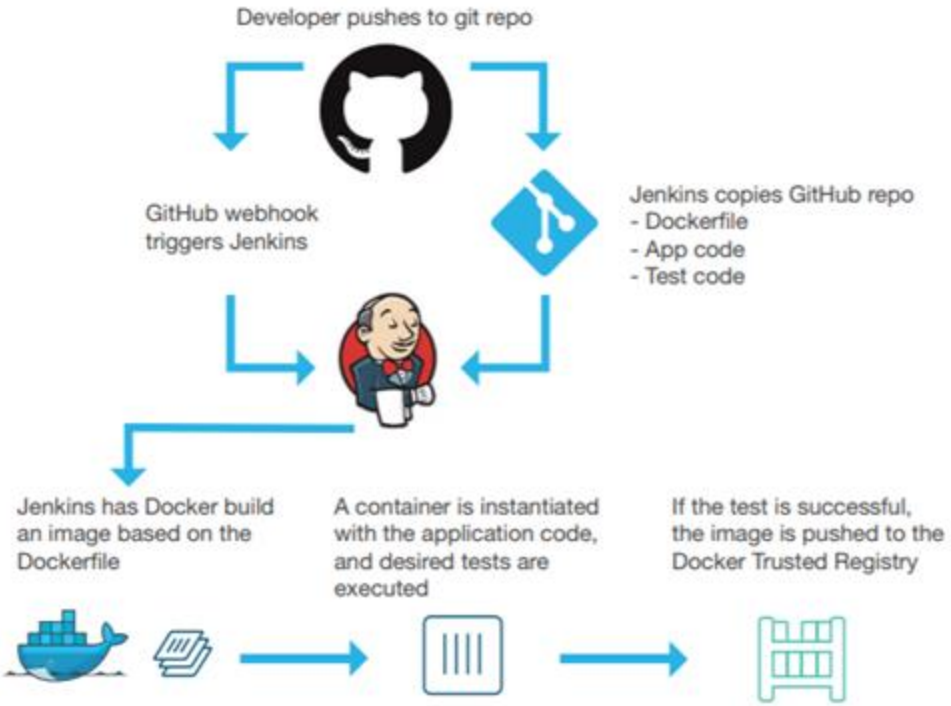
Before Jenkins



Jenkins Introduction

What is Jenkins?

Jenkins is a open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

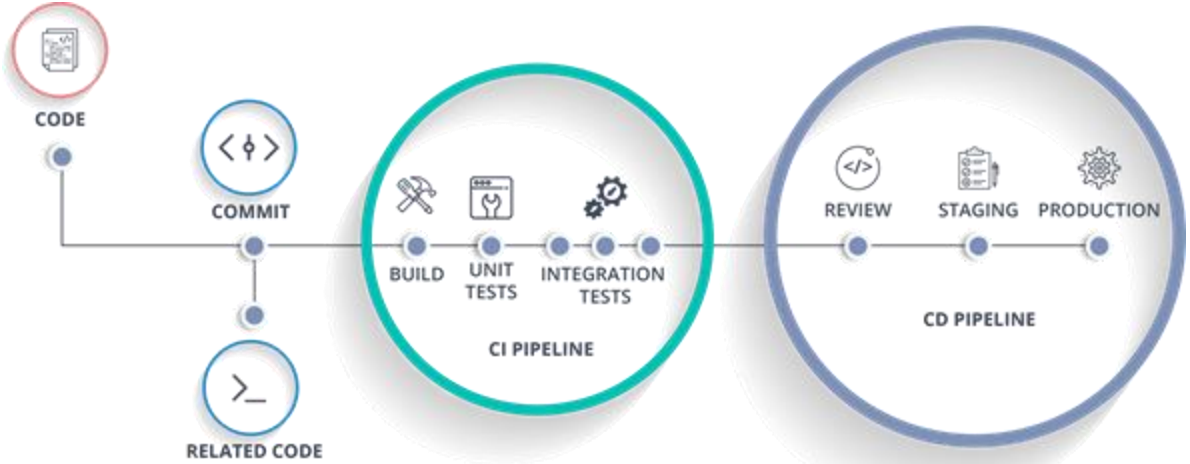


Jenkins Introduction

What is CI/CD?

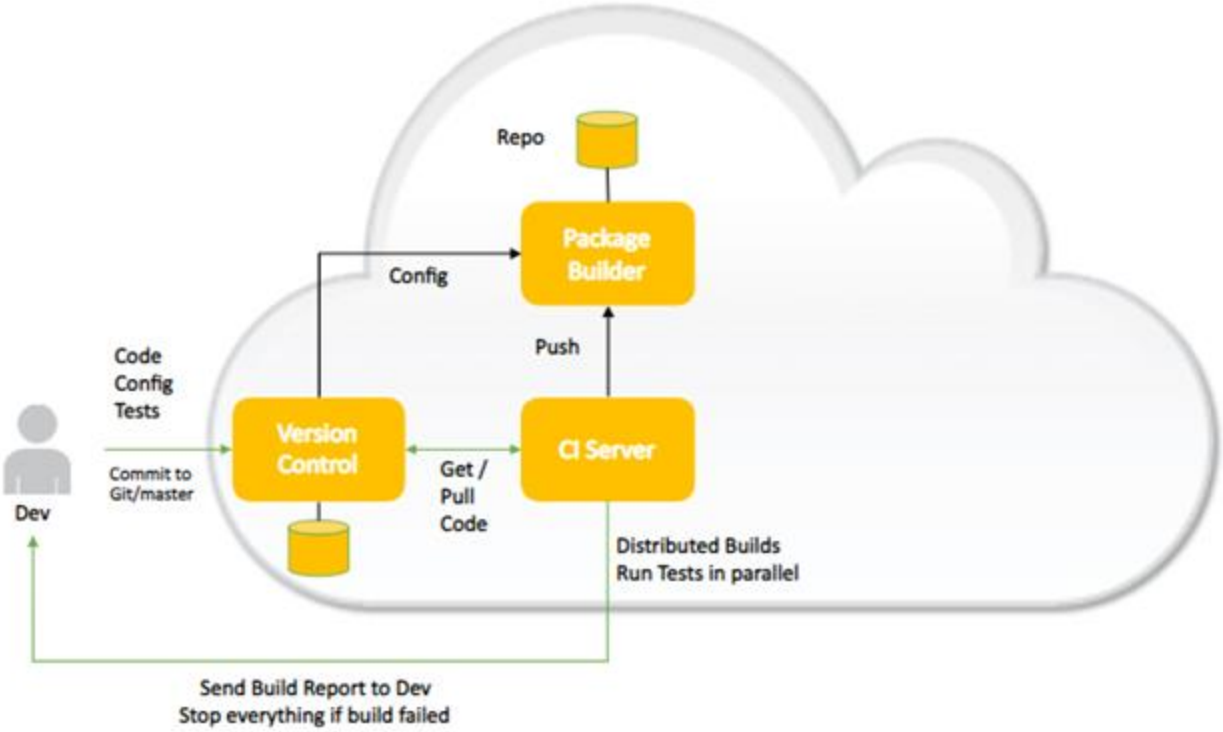
- Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project.

- A *continuous delivery (CD) pipeline* is an automated expression of your process for getting software from version control right through to your users and customers.
- Every change to your software (committed in source control) goes through a complex process on its way to being released.



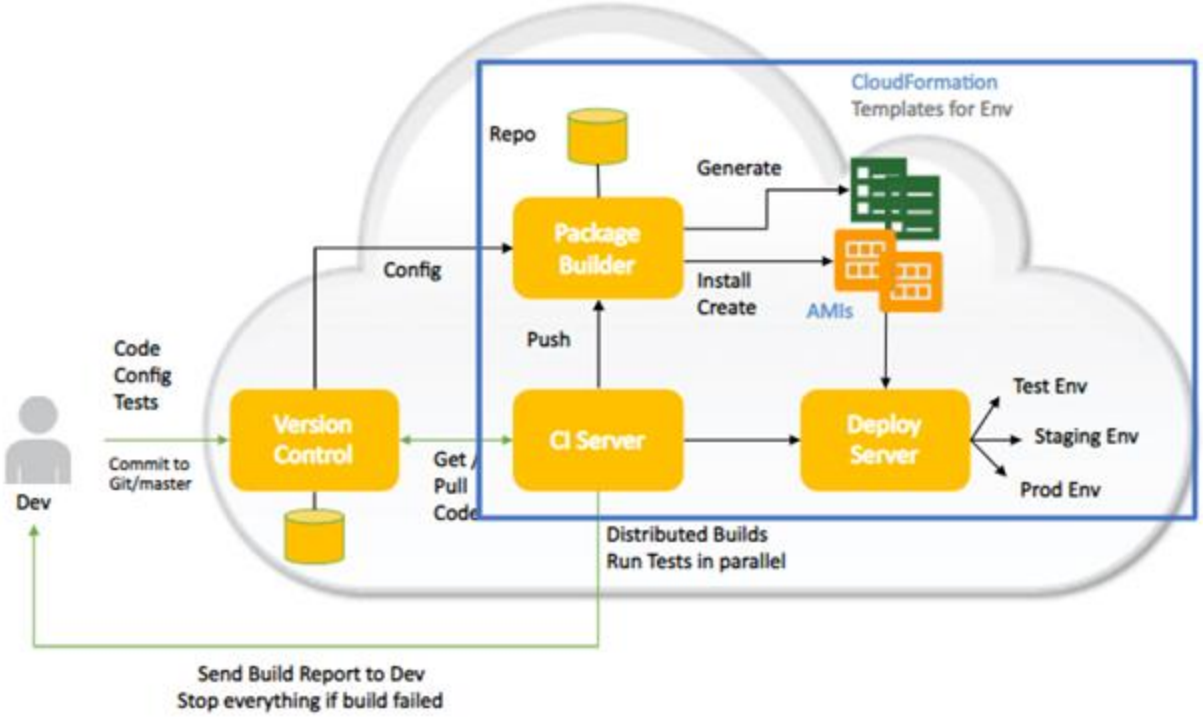
Jenkins Introduction

Continuous Integration

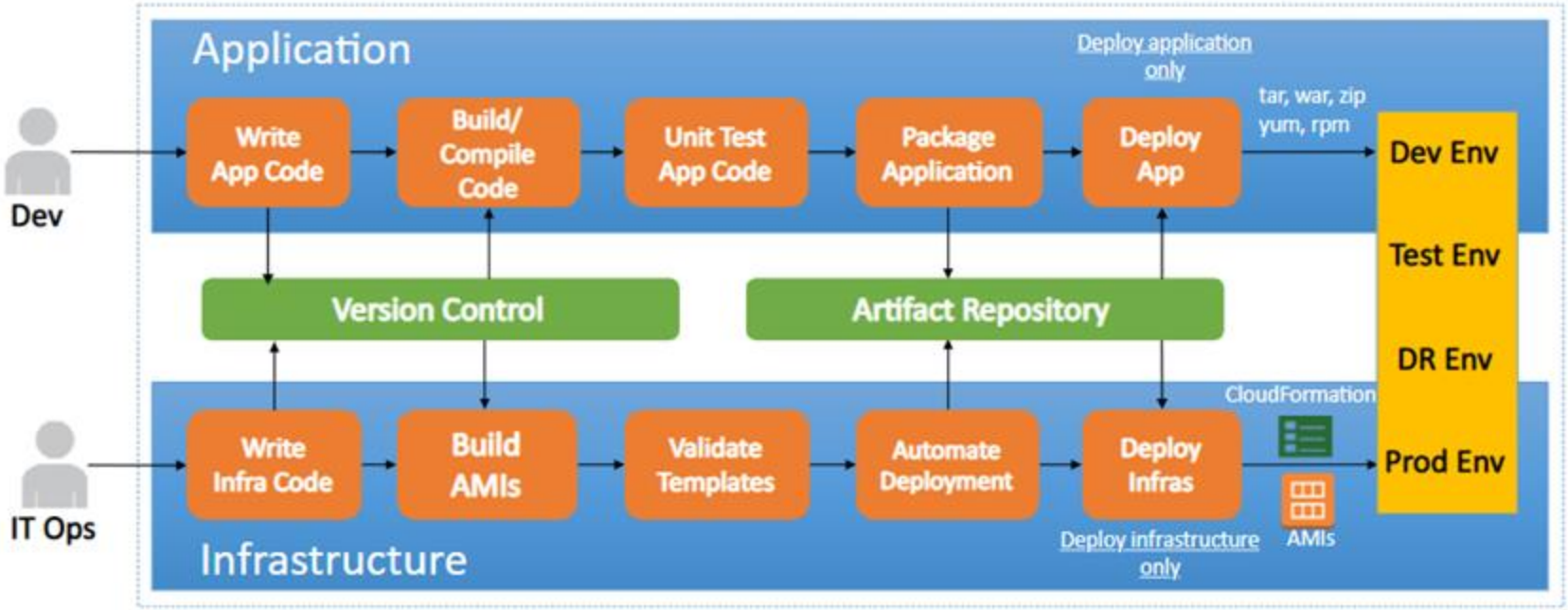


Jenkins Introduction

Continuous Delivery

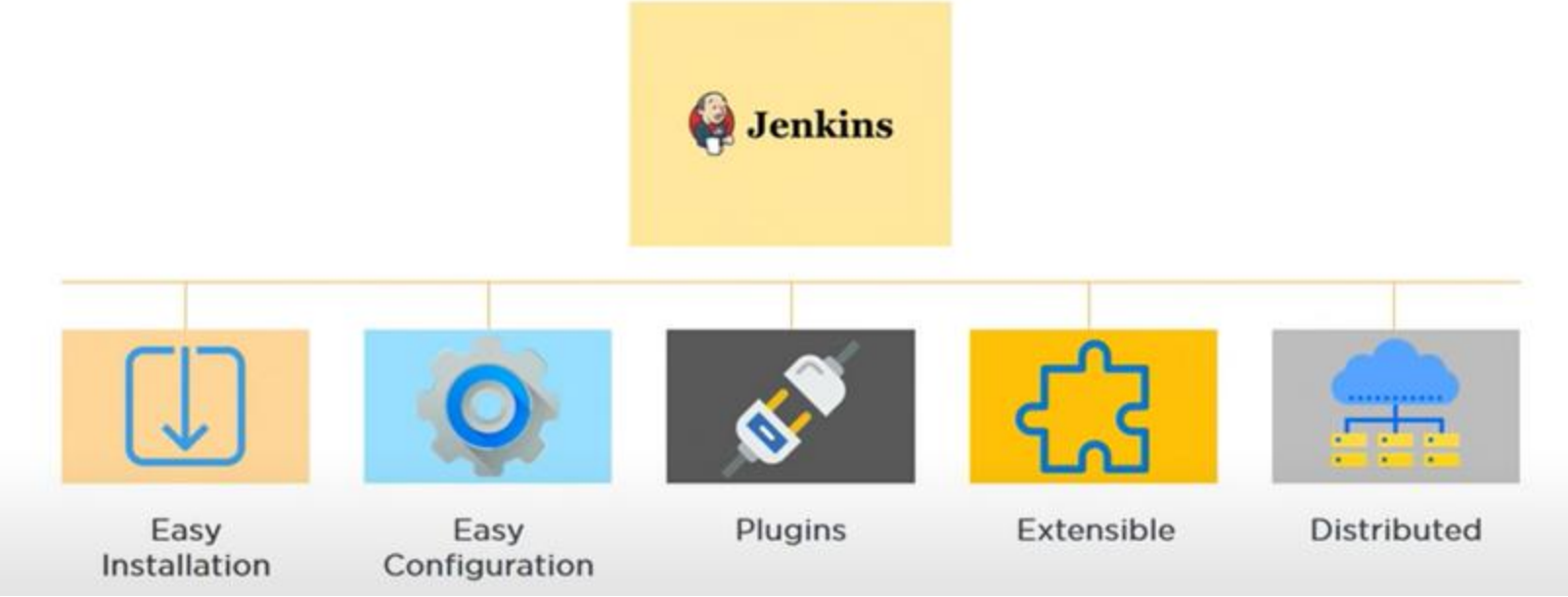


Example CI/CD pipeline



Jenkins Introduction

Features of Jenkins

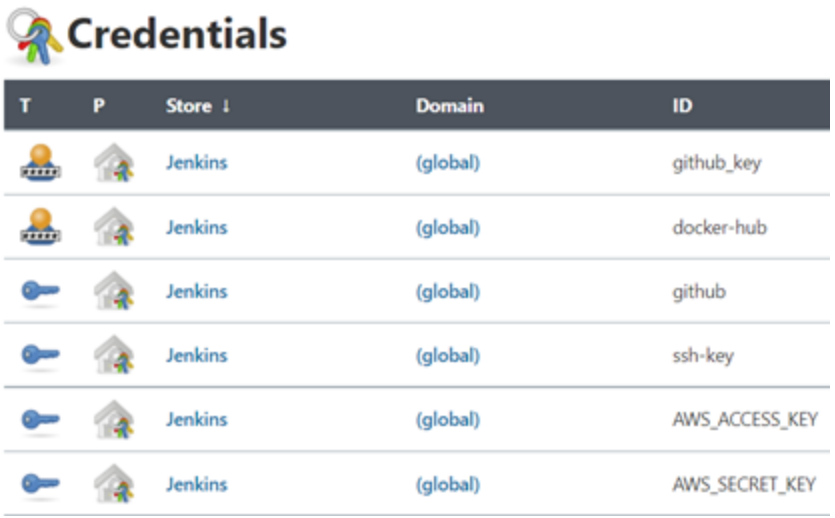


Jenkins Credentials

Jenkins Credentials

Credentials stored in Jenkins can be used:

- anywhere applicable throughout Jenkins (i.e. global credentials)
- by a specific Pipeline project/item (read more about this in the Handling credentials section of Using a Jenkinsfile)
- by a specific Jenkins user (as is the case for Pipeline projects created in Blue Ocean).



Credentials

T	P	Store ↓	Domain	ID
		Jenkins	(global)	github_key
		Jenkins	(global)	docker-hub
		Jenkins	(global)	github
		Jenkins	(global)	ssh-key
		Jenkins	(global)	AWS_ACCESS_KEY
		Jenkins	(global)	AWS_SECRET_KEY

Jenkins Credentials

Jenkins can store the following types of credentials:

- **Secret text** - a token such as an API token (e.g. a GitHub personal access token),
- **Username and password** - which could be handled as separate components or as a colon separated string in the format username:password (read more about this in Handling credentials),
- **Secret file** - which is essentially secret content in a file,
- **SSH Username with private key** - an SSH public/private key pair,
- **Certificate** - a PKCS#12 certificate file and optional password, or
- **Docker Host Certificate Authentication** credentials.

Jenkins Pepiline

Jenkins Pipeline

What is Jenkins Pipeline?

- Jenkins Pipeline is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.
- A Jenkinsfile can be written using two types of syntax - Declarative and Scripted.

Declarative

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {

            }
        }
        stage('Test') {
            steps {

            }
        }
        stage('Deploy') {
            steps {

            }
        }
    }
}
```

Scripted

```
node {
    stage('Build') {

    }
    stage('Test') {

    }
    stage('Deploy') {

    }
}
```

Jenkins Pipeline

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any ❶
    stages {
        stage('Build') { ❷
            steps {
                // ❸
            }
        }
        stage('Test') { ❹
            steps {
                // ❺
            }
        }
        stage('Deploy') { ❻
            steps {
                // ❼
            }
        }
    }
}
```

❶ Execute this Pipeline or any of its stages, on any available agent.

❷ Defines the "Build" stage.

❸ Perform some steps related to the "Build" stage.

❹ Defines the "Test" stage.

❺ Perform some steps related to the "Test" stage.

❻ Defines the "Deploy" stage.

❼ Perform some steps related to the "Deploy" stage.

Jenkins Pipeline

Jenkinsfile (Scripted Pipeline)

```
node {  ❶
    stage('Build') { ❷
        // ❸
    }
    stage('Test') { ❹
        // ❺
    }
    stage('Deploy') { ❻
        // ❼
    }
}
```

1. Execute this Pipeline or any of its stages, on any available agent.
2. Defines the "Build" stage.
3. Perform some steps related to the "Build" stage.
4. Defines the "Test" stage.
5. Perform some steps related to the "Test" stage.
6. Defines the "Deploy" stage.
7. Perform some steps related to the "Deploy" stage.

Jenkins Pipeline

Environment variable

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any
    environment { ❶
        CC = 'clang'
    }
    stages {
        stage('Example') {
            environment { ❷
                DEBUG_FLAGS = '-g'
            }
            steps {
                sh 'printenv'
            }
        }
    }
}
```

Jenkins Pipeline

Handling credentials

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent {
        // Define agent details here
    }
    environment {
        AWS_ACCESS_KEY_ID     = credentials('jenkins-aws-secret-key-id')
        AWS_SECRET_ACCESS_KEY = credentials('jenkins-aws-secret-access-key')
    }
    stages {
        stage('Example stage 1') {
            steps {
                // ❶
            }
        }
        stage('Example stage 2') {
            steps {
                // ❷
            }
        }
    }
}
```

Jenkins Pipeline

SSH User Private Key

Certificate

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent {
        // define agent details
    }
    stages {
        stage('Example stage 1') {
            steps {
                withCredentials(bindings: [sshUserPrivateKey(credentialsId: 'jenkins-ssh-key-for-abc', \
                                                             keyFileVariable: 'SSH_KEY_FOR_ABC')]) {
                  // ❶
                }
                withCredentials(bindings: [certificate(credentialsId: 'jenkins-certificate-for-xyz', \
                                                       keystoreVariable: 'CERTIFICATE_FOR_XYZ', \
                                                       passwordVariable: 'XYZ-CERTIFICATE-PASSWORD')]) {
                  // ❷
                }
            }
        }
        stage('Example stage 2') {
            steps {
                // ❸
            }
        }
    }
}
```

Jenkins Pipeline

Parameters

Jenkinsfile (Declarative Pipeline)

```
pipeline {
  agent any
  parameters {
    string(name: 'STATEMENT', defaultValue: 'hello; ls /', description: 'What should I say?')
  }
  stages {
    stage('Example') {
      steps {
        /* WRONG! */
        sh("echo ${STATEMENT}")
      }
    }
  }
}
```

Jenkins Pipeline

Handling failure

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                sh 'make check'
            }
        }
    }
    post {
        always {
            junit '**/target/*.xml'
        }
        failure {
            mail to: team@example.com, subject: 'The Pipeline failed :('
        }
    }
}
```

Jenkins Pipeline

Multiple Agents

```
stage('Test on Linux') {
    agent { ❷
        label 'linux'
    }
    steps {
        unstash 'app' ❸
        sh 'make check'
    }
    post {
        always {
            junit '**/target/*.xml'
        }
    }
}
```

```
stage('Test on Windows') {
    agent {
        label 'windows'
    }
    steps {
        unstash 'app'
        bat 'make check' ❹
    }
    post {
        always {
            junit '**/target/*.xml'
        }
    }
}
```

Jenkins Pipeline Syntax

Jenkins Pipeline Syntax

Docker Agents

```
pipeline {
    agent { docker 'maven:3.8.1-adoptopenjdk-11' } ❶
    stages {
        stage('Example Build') {
            steps {
                sh 'mvn -B clean verify'
            }
        }
    }
}
```

Jenkins Pipeline Syntax

Stage-level Agents

```
pipeline {
    agent none ❶
    stages {
        stage('Example Build') {
            agent { docker 'maven:3.8.1-adoptopenjdk-11' } ❷
            steps {
                echo 'Hello, Maven'
                sh 'mvn --version'
            }
        }
        stage('Example Test') {
            agent { docker 'openjdk:8-jre' } ❸
            steps {
                echo 'Hello, JDK'
                sh 'java -version'
            }
        }
    }
}
```

Jenkins Pipeline Syntax

Global Timeout & Stage Timeout

```
pipeline {
    agent any
    options {
        timeout(time: 1, unit: 'HOURS') ❶
    }
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

```
pipeline {
    agent any
    stages {
        stage('Example') {
            options {
                timeout(time: 1, unit: 'HOURS') ❶
            }
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

Jenkins Pipeline Syntax

Post section

```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
    post { ❶
        always { ❷
            echo 'I will always say Hello again!'
        }
    }
}
```

Conditions:

- ✓ always
- ✓ changed
- ✓ failure
- ✓ success
- ✓ unstable
- ✓ unsuccessful
- ✓ cleanup
- ✓ fixed
- ✓ regression
- ✓ aborted

Jenkins Pipeline Syntax

Trigger

```
// Declarative //
pipeline {
    agent any
    triggers {
        cron('H */4 * * 1-5')
    }
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

Jenkins Pipeline Syntax

Tools

```
pipeline {
    agent any
    tools {
        maven 'apache-maven-3.0.1' ❶
    }
    stages {
        stage('Example') {
            steps {
                sh 'mvn --version'
            }
        }
    }
}
```

Jenkins Pipeline Syntax

Condition

```
pipeline {
    agent any
    stages {
        stage('Example Build') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Example Deploy') {
            when {
                branch 'production'
                environment name: 'DEPLOY_TO', value: 'production'
            }
            steps {
                echo 'Deploying'
            }
        }
    }
}
```

Jenkins Pipeline Syntax

Parallel stages

```
stage('Parallel In Sequential') {
    parallel {
        stage('In Parallel 1') {
            steps {
                echo "In Parallel 1"
            }
        }
        stage('In Parallel 2') {
            steps {
                echo "In Parallel 2"
            }
        }
    }
}
```

Jenkins Pipeline Syntax

Matrix

```
matrix {
    axes {
        axis {
            name 'PLATFORM'
            values 'linux', 'mac', 'windows'
        }
    }
    // ...
}
```

```
matrix {
    axes {
        axis {
            name 'PLATFORM'
            values 'linux', 'mac', 'windows'
        }
        axis {
            name 'BROWSER'
            values 'chrome', 'edge', 'firefox', 'safari'
        }
    }
    // ...
}
```

Jenkins Pipeline Syntax

Get value from matrix

```
matrix {
    axes {
        axis {
            name 'PLATFORM'
            values 'linux', 'mac', 'windows'
        }
        axis {
            name 'BROWSER'
            values 'chrome', 'edge', 'firefox', 'safari'
        }
        axis {
            name 'ARCHITECTURE'
            values '32-bit', '64-bit'
        }
    }
```

```
    excludes {
        exclude {
            axis {
                name 'PLATFORM'
                values 'mac'
            }
            axis {
                name 'ARCHITECTURE'
                values '32-bit'
            }
        }
    }
    // ...
}
```

Jenkins Pipeline Syntax

Choice Parameter to run

```
parameters {
  choice(
    choices: ['deployment' , 'cleanup'],
    description: '',
    name: 'REQUESTED_ACTION'
  )
}
```

Jenkins Pipeline Syntax

Scripts

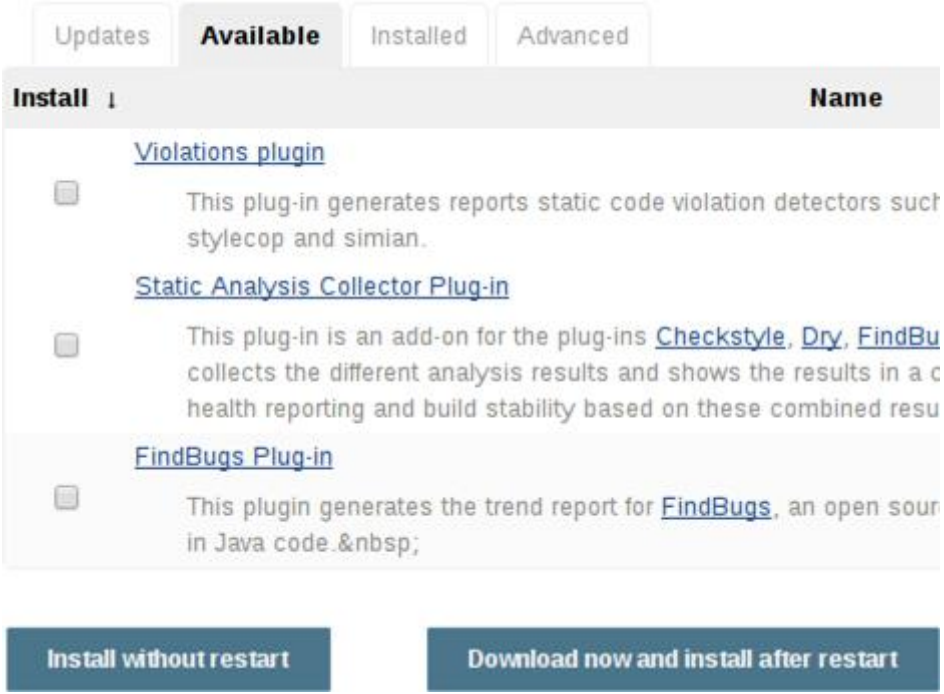
```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'

                script {
                    def browsers = ['chrome', 'firefox']
                    for (int i = 0; i < browsers.size(); ++i) {
                        echo "Testing the ${browsers[i]} browser"
                    }
                }
            }
        }
    }
}
```

Jenkins Plugins

Jenkins Plugins

- Plugins are the primary means of enhancing the functionality of a Jenkins environment to suit organization- or user-specific needs.
- There are over a thousand different plugins which can be installed on a Jenkins controller and to integrate various build tools, cloud providers, analysis tools, and much more.

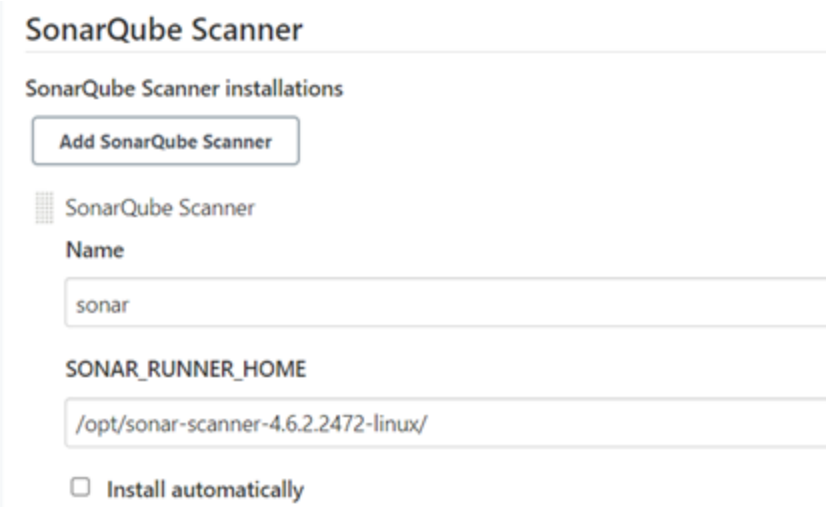


Jenkins Plugins

Example SonarQube Plugin

Install SonarQube in **Manage Plugins**

Config plugin in
Global Tool Configuration



Jenkins Plugins

Example SonarQube Plugin

Configure System

SonarQube servers

☑ **Environment variables** Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name

sonar

Server URL

http://16.162.93.71:9000

Default is http://localhost:9000

Server authentication token

sonar | Add

SonarQube authentication token. Mandatory when anonymous access is disabled.

Jenkins Plugins

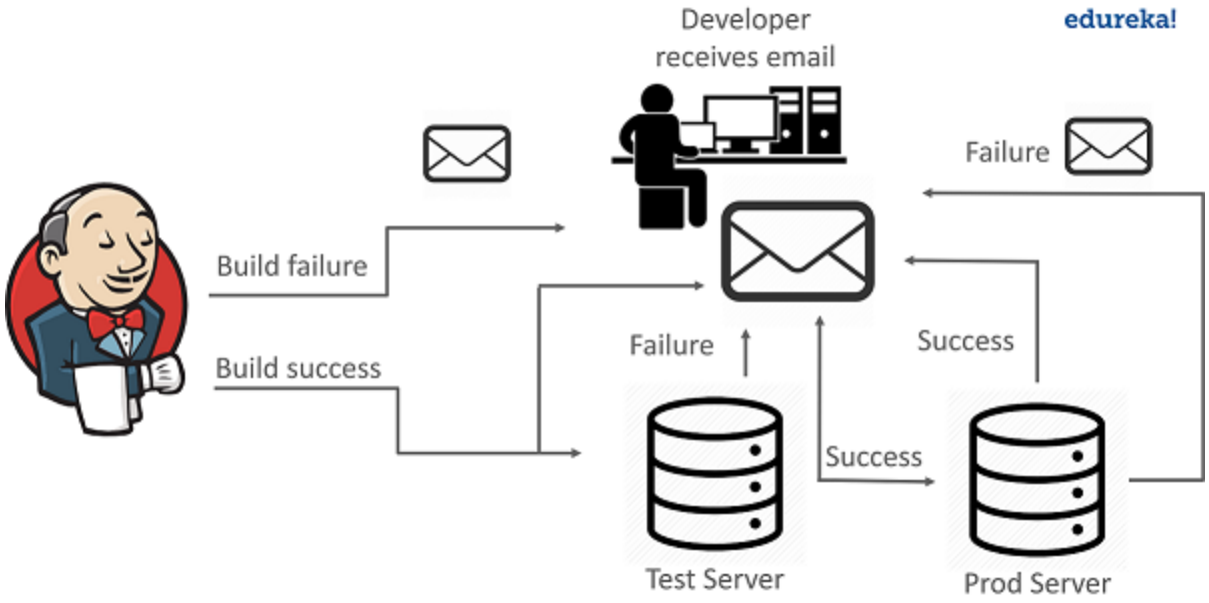
Example SonarQube Plugin

Use plugin in pipeline

```
stage ("Sonar") {
    steps {
        script {
            def scannerHome = tool name: 'sonar', type: 'hudson.plugins.sonar.SonarRunnerInstallation';
            withSonarQubeEnv("sonar"){
                sh "${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=training -Dsonar.sources=."
            }
        }
    }
}
```

Jenkins Plugins

Example Email Extension Plugin



Jenkins Plugins

Example Email Extension Plugin

Install plugin

Install ↑	Name
☐	Email Extension Template emailext Build Notifiers This plugin allows administrators to create global templat

Use plugin in pipeline

```
success {
    emailext body: 'Pipeline run success!!!',
    recipientProviders: [
        [$class: 'DevelopersRecipientProvider'],
        [$class: 'RequesterRecipientProvider']
    ],
    subject: 'Alert'
}
```

Configure System

Extended E-mail Notification

SMTP server

smtp.gmail.com

SMTP Port

465

SMTP Username

training@fpt.com.vn

SMTP Password

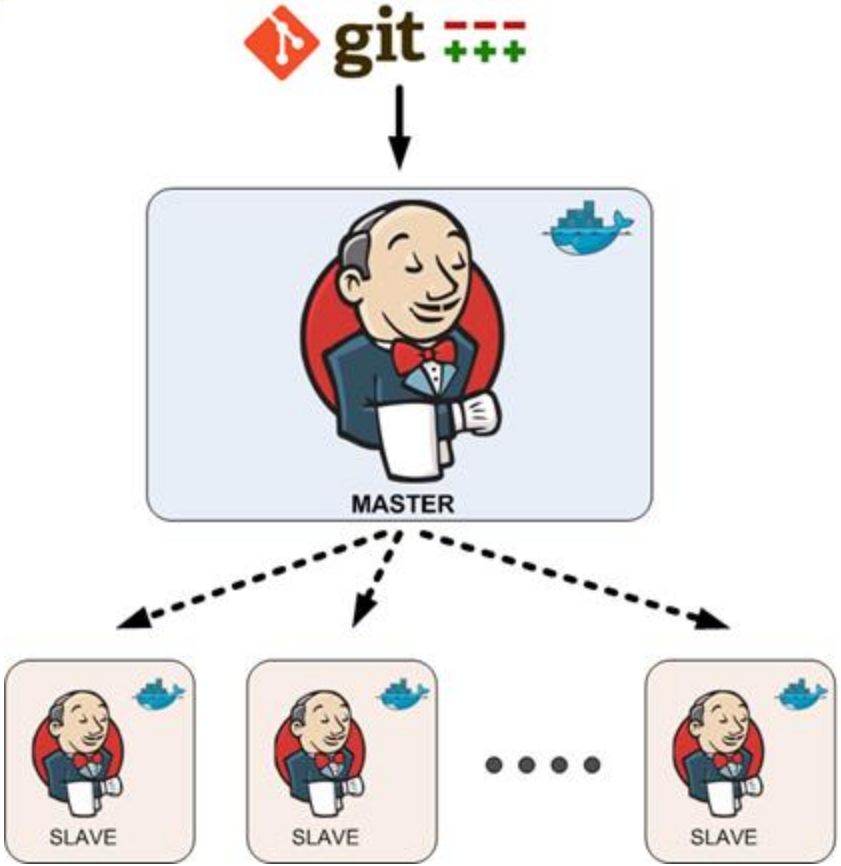
Concealed

☑ Use SSL

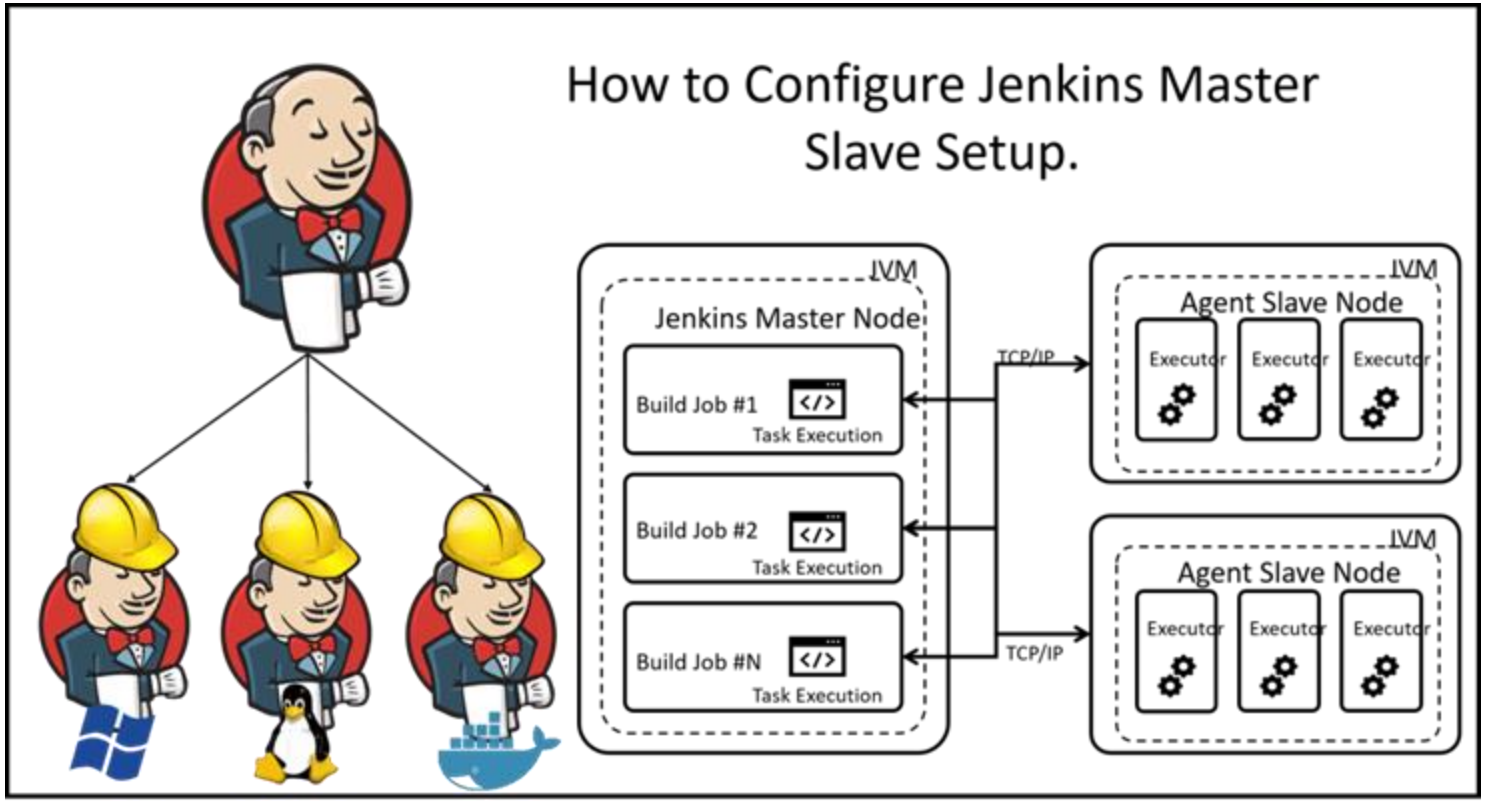
Jenkins Master and Slave

Jenkins Master and Slave

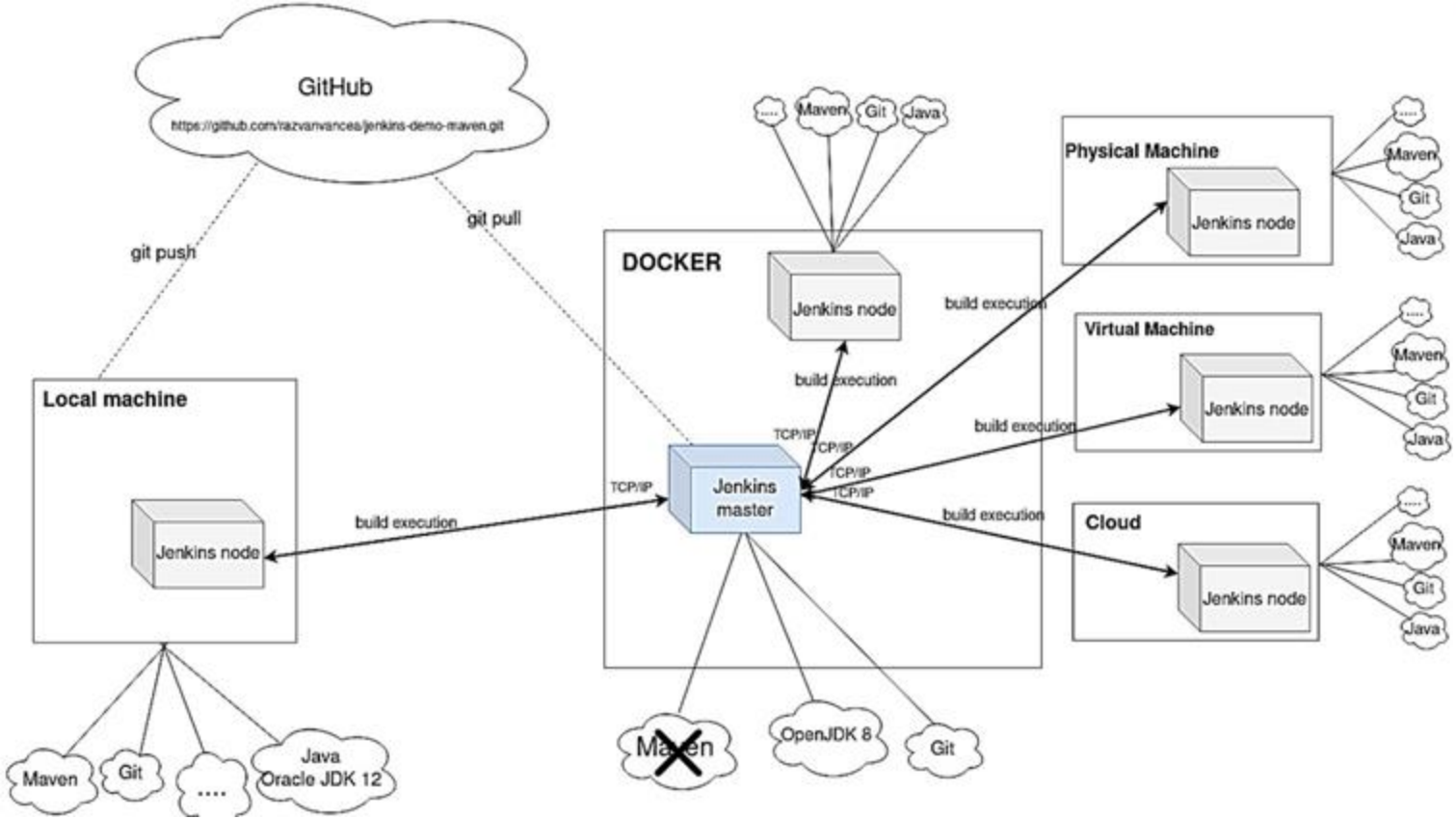
- The Jenkins master acts to schedule the jobs and assign slaves and send builds to slaves to execute the jobs.
- It will also monitor the slave state (offline or online) and getting back the build result responses from slaves and the display build results on the console output.
- The workload of building jobs is delegated to multiple **slaves**.



Jenkins Master and Slave



Jenkins Master and Slave



Lab

- Install Jenkins
- Write a Jenkinsfile and push to GitHub
- Create a pipeline on Jenkins with GitHub repository
- Run this pipeline

Thank you