

# How to - Install jenkins and configuration on Cloud

## Pre-Requisites:

1. Operating system : RHEL/CentOS/Fedora/Ubuntu/Debian
2. JDK 1.7 or above
3. Docker and git should be installed.

## Jenkins installation –RHEL7

1. To install Jenkins on AWS/Gcloud Downloaded Docker Jenkins image through docker command.
- 2.

```
#####
```

Command

```
Docker pull Jenkins:latest
```

Run docker

```
docker run -p 8080:8080 -p 50000:50000 -v  
/your/home:/var/jenkins_home/Jenkins jenkins
```

```
#####
```

This will store the Jenkins data in /your/home on the host. Ensure that /your/home is accessible by the Jenkins user in container

3. docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
fb8822aea816	jenkins	"/bin/tini -- /usr/l..."	About a minute ago	Up About a minute	0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp	elastic_marguli

4. login into container

CMD : docker exec -it {container id} /bin/bash

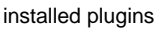
5. Once Jenkins is up and running, we can open its UI in a browser.

open: <http://localhost>

## Installing Plugins in Jenkins

Once the user has started the jenkins and is able to see the home page, the first thing is to install the plugins.

Go to Jenkins --> Manage Jenkins --> Manage Plugins web link as shown below:-



Go to manage Jenkins-->global tool configuration

Go to JDK, Ant installation and click on the add button and enter the respective installation path as shown below.

JDK

831px



**Maven**

Maven installations

[Add Maven](#)

☐ Maven

Name

☒ Install automatically

☐ Install from Apache

Version

[Delete installer](#)

[Add installer](#)

**Git**

List of Docker installations on this system

Git installations

[Add Git](#)

☐ Git

Name

Git launch arguments

[Save](#) [Apply](#)

## Creating pipeline job in Jenkins:

To create a pipeline in Jenkins follow under mentioned steps:

1.Create on New Item and select the job type as pipeline. This is shown in below screenshot:

Jenkins > All >

\* Required field

- Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- External Job**  
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.
- Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Bitbucket Team/Project**  
Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.
- Folder**  
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 space, so you can have multiple things of the same name as long as they are in different folders.

## 2.Configure job

Jenkins > Nov-demo >

General Outgoing Triggering Build Triggers Advanced Project Options **Pipeline** [Advanced...](#)

**Pipeline**

Definition

SCM

Repositories

Repository URL

Credentials

[Add Repository](#)

Branches to build

Branch Specifier (blank for 'any')

[Add Branch](#)

Repository browser

Additional Behaviours [Add](#)

Script Path

Lightweight checkout ☒

[Pipeline Syntax](#)

[Save](#) [Apply](#)

## 3.Jenkins file stage

### a. Build stage

```
#####
```

Build Stage

```
#####
```

In this stage its build the scala code and compile the packages

```
stage('Build') {  
    steps {  
        checkout scm  
        echo 'Start Compiling'  
        build 'Scala'  
    }  
}
```

#### b. Sonar test

```
#####
```

Sonar Test

```
#####
```

In this stage sonar code quality tests are executed on the scala code which was checked out from repository

```
stage('Sonar Test') {  
    steps {  
        echo 'Start Sonar Test'  
        sonar 'Scala'  
    }  
}
```

#### c. Publish

#####

Publish the code to  
nexus/artifactory

#####

In this stage it publish the artifacts in to nexus repository

```
script {  
    withCredentials([usernamePassword(credentialsId:  
'artifactory', passwordVariable: 'pass', usernameVariable: 'user')]) {  
        version = nextVersionFromGit()  
        sh "sed -i '/ThisBuild \\\\/ version/c\\ThisBuild  
\\\\\\\/ version      :=\"${version}\"' build.sbt"  
        sh "${tool name: 'sbt',  
type:'org.jvnet.hudson.plugins.SbtPluginBuilder$SbtInstallation'}/bin/sb  
t package"  
        echo "Deploy  
target/scala-2.12/hello-world_2.12-${version}.jar "  
        sh "curl -u ${user}:${pass} -T  
target/scala-2.12/hello-world_2.12-${version}.jar  
'http://novartis.devops.altimetrik.io:8081/artifactory/scala-project/hel  
lo-world_${version}.jar'"  
    }  
}
```