

# Jenkins Configuration

1. SonarQube
2. Jfrog
3. Maven
4. Active Directory

## **Prerequisite**

1. We need to have installed jenkins for this.
2. Sonarqube endpoint
3. Jfrog endpoint and credentials

# SonarQube

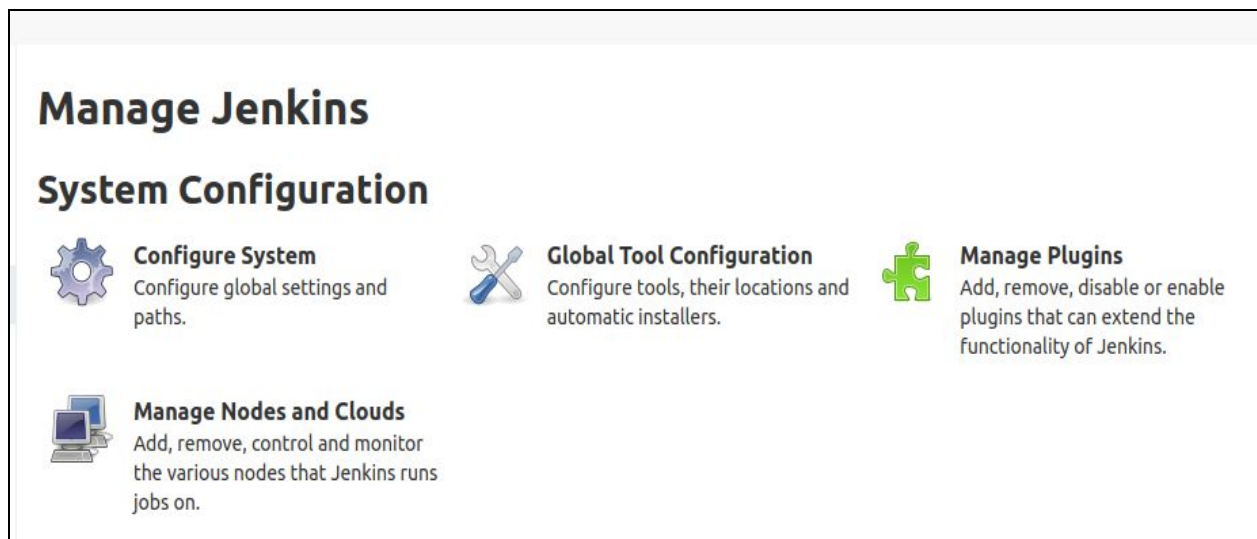
Our Endpoint is:-

<http://34.72.39.32:9000>

**Step-1** Select Manage Jenkins in your Dashboard



**Step-2** In Manage Jenkins Select Configuration System



### Step-3 Scroll until section Sonarqube servers appears

## 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

Add SonarQube

List of SonarQube installations

### Step-4 Click on Add Sonarqube button

Dashboard > configuration

☐ Disable deferred wipeout on this node

☐ Environment variables

☐ Tool Locations

### 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

Server URL

Server authentication token

- none -

Add

Default is http://localhost:9000

SonarQube authentication token. Mandatory when anonymous access is disabled.

Advanced...

Delete SonarQube

Add SonarQube

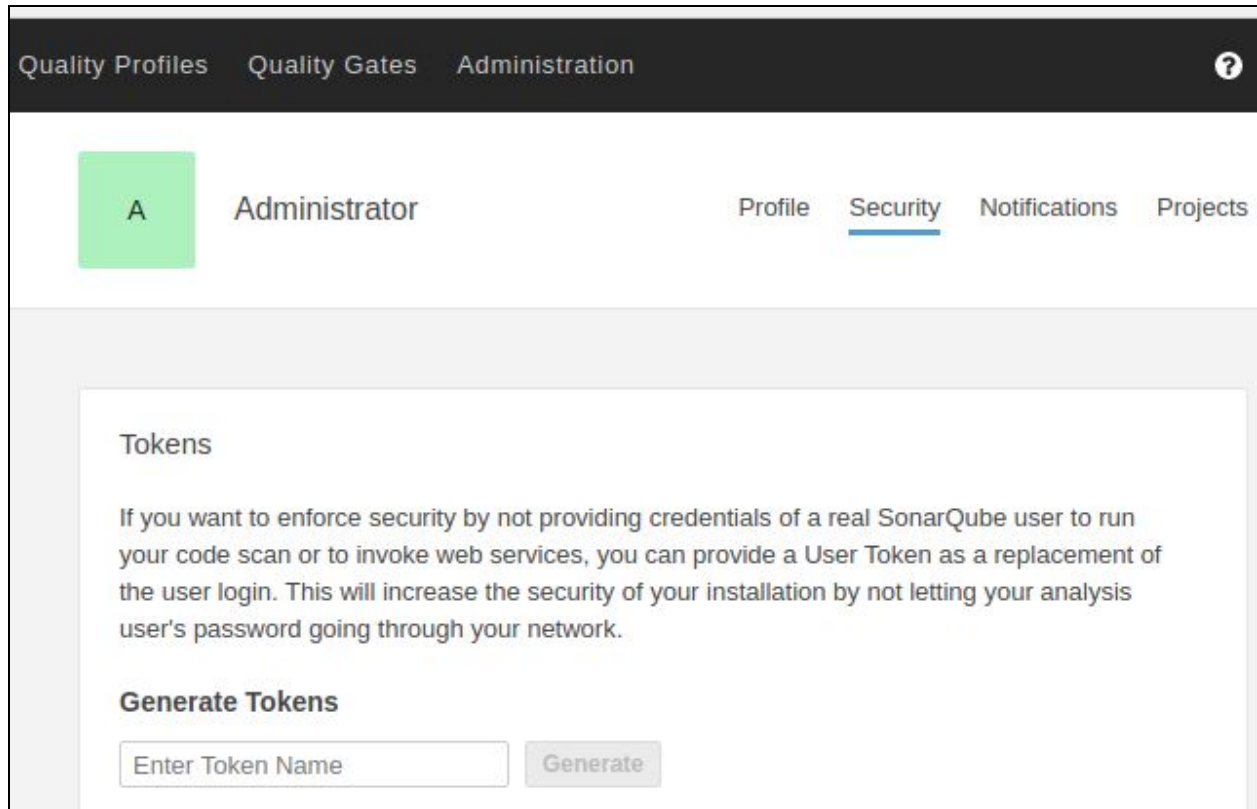
List of SonarQube installations

Save

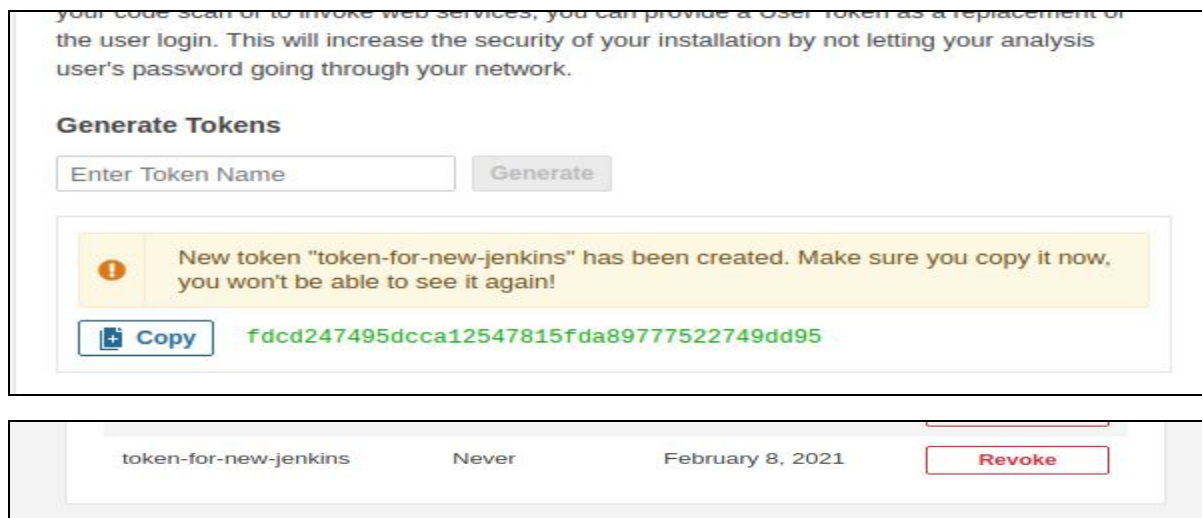
Apply

## Step-5 First create a token for SonarQube in SonarQube

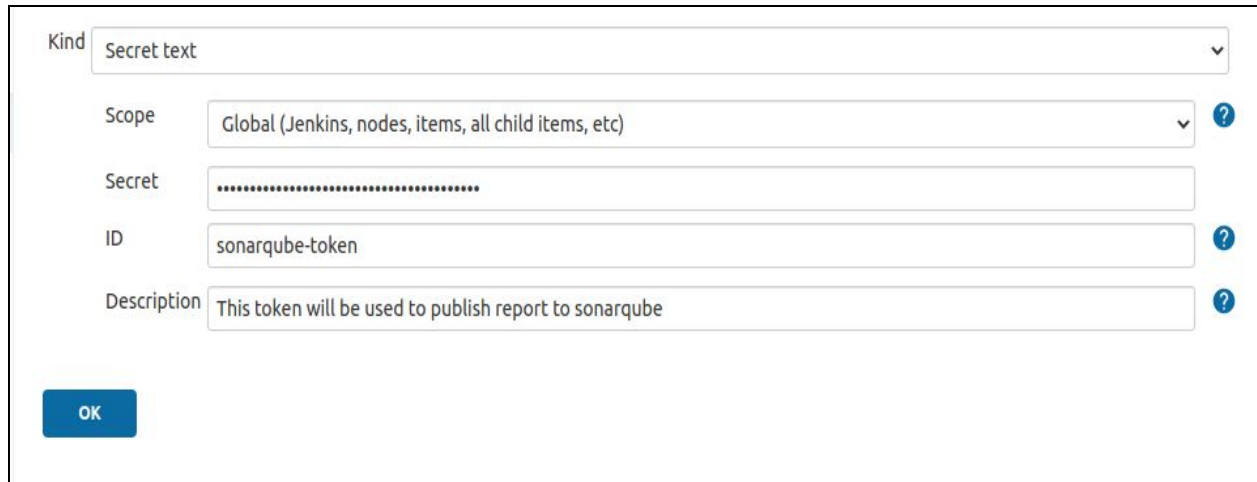
Use your SonarQube and select Administration and in the Security Section you will see Tokens. Here we can create a new token for our new Jenkins.



Enter the name of your token and click on generate token. This is just to show you we have deleted this token.





**Step-6** Enter the token and id and description about your token. It will be used everytime Jenkins wants to access sonarqube.

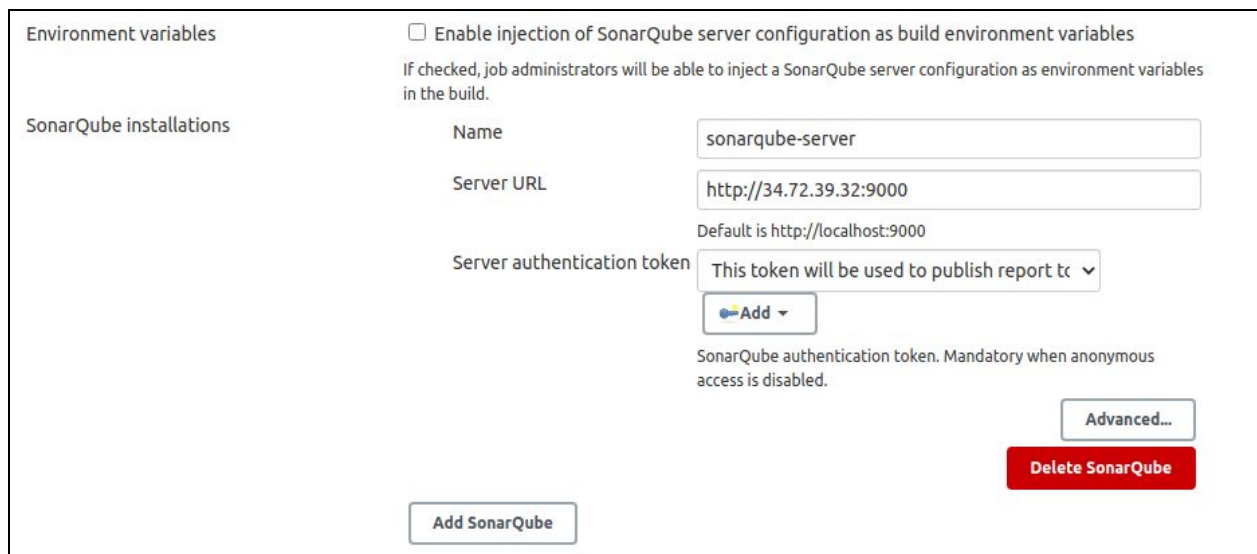


The screenshot shows the 'Add Credentials' form in Jenkins. The 'Kind' is set to 'Secret text'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Secret' field is masked with dots. The 'ID' is 'sonarqube-token'. The 'Description' is 'This token will be used to publish report to sonarqube'. There is an 'OK' button at the bottom left.

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 sonarqube-token	This token will be used to publish report to sonarqube	Secret text	This token will be used to publish report to sonarqube 

**Step-7** Enter the details like Name of server, it's address and key through which we will access it.



The screenshot shows the 'SonarQube installations' configuration page. Under 'Environment variables', there is a checkbox 'Enable injection of SonarQube server configuration as build environment variables'. Below it, a note states: 'If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.' The 'Name' field is 'sonarqube-server'. The 'Server URL' is 'http://34.72.39.32:9000'. A note below says 'Default is http://localhost:9000'. The 'Server authentication token' dropdown is set to 'This token will be used to publish report to'. There is an 'Add' button with a key icon. A note at the bottom says 'SonarQube authentication token. Mandatory when anonymous access is disabled.' At the bottom right, there are 'Advanced...' and 'Delete SonarQube' buttons. At the bottom center, there is an 'Add SonarQube' button.

**Sonarqube is successfully configured.**

# JFROG

Our Endpoint is

<https://raja1234.jfrog.io/ui/admin/management/users>


**Step-1** Select Manage Jenkins in your Dashboard





## Step-2 In Manage Jenkins Select Manage Plugins


# Manage Jenkins

## System Configuration

**Configure System**  
Configure global settings and paths.

**Global Tool Configuration**  
Configure tools, their locations and automatic installers.

**Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

**Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

## Step-3 Search for Artifactory Plugin and install it.

Updates

Available

Installed

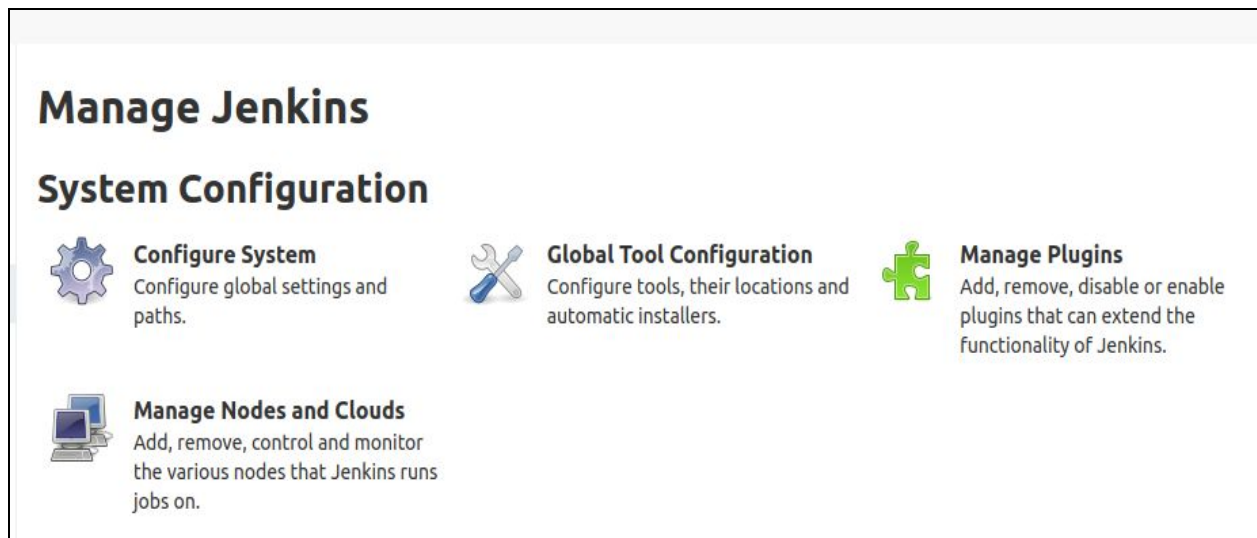
Advanced

Install	Name	Version	Released
<input checked="" type="checkbox"/>	<b>Artifactory</b> pipeline This plugin allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, and then have them linked to the build job that created them. The plugin includes a vast collection of features, including a rich pipeline API library and release management for Maven and Gradle builds with Staging and Promotion.	3.10.4	14 days ago
<input type="checkbox"/>	<b>lambdatest-automation</b> Artifactory auto generated POM	1.19	2 mo 14 days ago

**Step-4** Again in your dashboard. Click on section Manage Jenkins



**Step-5** In Manage Jenkins Select Configuration System





## Step-6 Scroll until section Jfrog appears

### JFrog

☐ Use the Credentials Plugin

Artifactory servers

Add Artifactory Server

List of Artifactory servers that projects will want to deploy artifacts and build info to

JFrog Pipelines server

Integration URL

Advanced...

### Credentials

Credentials

- none -

Add

Test Connection

Report job results to JFrog Pipelines

Save

Apply

## Step-7 Click on Add Artifactory Servers

### JFrog

☐ Use the Credentials Plugin

Artifactory servers

Artifactory

Server ID

Please set server ID

URL

Advanced...

## Step-8 Click on Add Artifactory Servers

**JFrog**

☐ Use the Credentials Plugin

Artifactory servers

Artifactory

Server ID

URL

Advanced...

## Step-9 Add your credentials here.

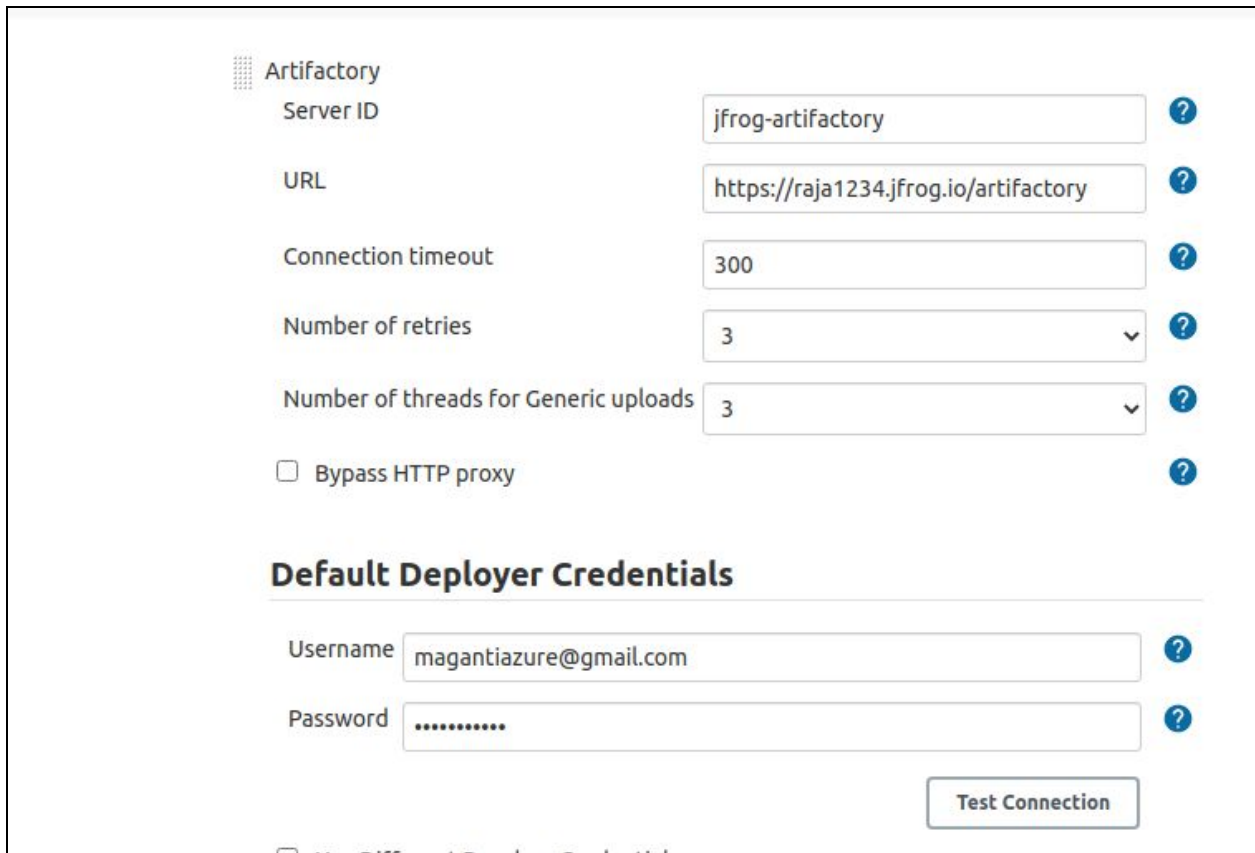
**Default Deployer Credentials**

Username

Password

Test Connection

**Step-10** It will look something like this



The screenshot shows the JFrog configuration interface. Under the 'Artifactory' section, there are several input fields: 'Server ID' with the value 'jfrog-artifactory', 'URL' with the value 'https://raja1234.jfrog.io/artifactory', 'Connection timeout' with the value '300', 'Number of retries' with a dropdown menu showing '3', and 'Number of threads for Generic uploads' with a dropdown menu showing '3'. There is also a checkbox labeled 'Bypass HTTP proxy'. Below this section is a section titled 'Default Deployer Credentials' which contains a 'Username' field with the value 'magantiazure@gmail.com' and a 'Password' field with masked characters. A 'Test Connection' button is located at the bottom right of the 'Default Deployer Credentials' section.

**Step-11** Click on Test Connection and it will show if an artifactory was found or not.



The screenshot shows the JFrog configuration interface after clicking the 'Test Connection' button. The 'Default Deployer Credentials' section is still visible, but the 'Test Connection' button is now disabled. Below the 'Test Connection' button, the text 'Found Artifactory 7.13.2' is displayed, indicating that the connection was successful. There is also a checkbox labeled 'Use Different Resolver Credentials' at the bottom left.

**JFrog is successfully configured.**

# MAVEN

It is required to install maven in the Jenkins instance and then in configuration system we will point where maven is i.e /usr/share/jenkins.

## Step-1 Installing maven.

First use yum update to update repository

```
[ec2-user@ip-172-31-36-177 ~]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.7 kB 00:00:00
amzn2extra-docker | 3.0 kB 00:00:00
jenkins | 2.9 kB 00:00:00
No packages marked for update
[ec2-user@ip-172-31-36-177 ~]$
```

Then use **sudo yum install maven**. This will fetch packages and then proceed for installation

```
[ec2-user@ip-172-31-36-177 ~]$ sudo yum install maven
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package maven.noarch 0:3.0.5-17.amzn2 will be installed
--> Processing Dependency: sisu-inject-plexus for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: sisu-inject-bean for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: plexus-utils for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: plexus-sec-dispatcher for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: plexus-interpolation for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: plexus-containers-component-annotations for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: plexus-cipher for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: objectweb-asm for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.sonatype.sisu:sisu-inject-plexus) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.sonatype.plexus:plexus-sec-dispatcher) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.sonatype.plexus:plexus-cipher) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.sonatype.aether:aether-util) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.sonatype.aether:aether-spi) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.sonatype.aether:aether-impl) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.sonatype.aether:aether-api) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.codehaus.plexus:plexus-utils) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.codehaus.plexus:plexus-interpolation) for package: maven-3.0.5-17.amzn2.noarch
--> Processing Dependency: mvn(org.codehaus.plexus:plexus-container-default) for package: maven-3.0.5-17.amzn2.noarch
```

Enter option Y to install maven.

```
plexus-sec-dispatcher      noarch      1.4-13.amzn2      amzn2-core      29 k
plexus-utils               noarch      3.0.9-9.amzn2     amzn2-core      225 k
qdox                      noarch      1.12.1-10.amzn2   amzn2-core      170 k
regex                      noarch      1.5-13.amzn2      amzn2-core      47 k
sisu-inject-bean          noarch      2.3.0-11.amzn2    amzn2-core      181 k
sisu-inject-plexus        noarch      2.3.0-11.amzn2    amzn2-core      179 k
slf4j                     noarch      1.7.4-4.amzn2     amzn2-core      170 k
tomcat-servlet-3.0-api    noarch      7.0.76-10.amzn2.0.2 amzn2-core      213 k
xalan-j2                  noarch      2.7.1-23.1.amzn2  amzn2-core      1.9 M
xbean                     noarch      3.13-6.amzn2      amzn2-core      376 k
xerces-j2                 noarch      2.11.0-17.amzn2   amzn2-core      1.1 M
xml-commons-apis          noarch      1.4.01-16.amzn2   amzn2-core      227 k
xml-commons-resolver      noarch      1.2-15.amzn2      amzn2-core      108 k

Transaction Summary
=====
Install 1 Package (+66 Dependent packages)

Total download size: 16 M
Installed size: 21 M
Is this ok [y/d/N]: █
```

## Maven is successfully installed

```
log4j.noarch 0:1.2.17-16.amzn2
maven-wagon.noarch 0:2.4-3.amzn2
nekohtml.noarch 0:1.9.14-13.amzn2
objectweb-asn1.noarch 0:3.3.1-9.amzn2
plexus-cipher.noarch 0:1.7-5.amzn2
plexus-classworlds.noarch 0:2.4.2-8.amzn2
plexus-component-api.noarch 0:1.0-0.16.alpha15.amzn2
plexus-containers-component-annotations.noarch 0:1.5.5-14.amzn2
plexus-containers-container-default.noarch 0:1.5.5-14.amzn2
plexus-interactivity.noarch 0:1.0-0.14.alpha6.amzn2
plexus-interpolation.noarch 0:1.15-8.amzn2
plexus-sec-dispatcher.noarch 0:1.4-13.amzn2
plexus-utils.noarch 0:3.0.9-9.amzn2
qdox.noarch 0:1.12.1-10.amzn2
regex.noarch 0:1.5-13.amzn2
sisu-inject-bean.noarch 0:2.3.0-11.amzn2
sisu-inject-plexus.noarch 0:2.3.0-11.amzn2
slf4j.noarch 0:1.7.4-4.amzn2
tomcat-servlet-3.0-api.noarch 0:7.0.76-10.amzn2.0.2
xalan-j2.noarch 0:2.7.1-23.1.amzn2
xbean.noarch 0:3.13-6.amzn2
xerces-j2.noarch 0:2.11.0-17.amzn2
xml-commons-apis.noarch 0:1.4.01-16.amzn2
xml-commons-resolver.noarch 0:1.2-15.amzn2

Complete!
[ec2-user@ip-172-31-36-177 ~]$ █
```

## Step-2 Fetch Maven location in instance

Use command `whereis maven`.

**/usr/share/maven** is a directory where our whole binaries and configuration exist.

```
[ec2-user@ip-172-31-36-177 ~]$ whereis maven
maven: /etc/maven /usr/share/maven
[ec2-user@ip-172-31-36-177 ~]$
```


## Step-3 Select Manage Jenkins in your Dashboard





## Step-4 In Manage Jenkins click on Global Tool Configuration


### Manage Jenkins

#### System Configuration

**Configure System**  
Configure global settings and paths.

**Global Tool Configuration**  
Configure tools, their locations and automatic installers.

**Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

**Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

## Step-5 Scroll down until Maven appears

List of SonarQube Scanner installations on this system

**Ant**

Ant installations

List of Ant installations on this system

**Maven**

**Docker**

Docker installations

List of Docker installations on this system

## Step-6 Unselect Install automatically

Dashboard > Global Tool Configuration

Maven

Maven installations

Add Maven

Maven

Name

Required

☒ Install automatically

Install from Apache

Version

3.6.3

Add Installer

Add Maven

Save

Apply

List of Maven installations on this system

Delete Installer

Delete Maven



**Step-7** Here add Name and path of maven as discussed in previous step.

### Maven

Maven installations

Add Maven

Maven

Name

maven

MAVEN\_HOME

/usr/share/maven

☐ Install automatically

?

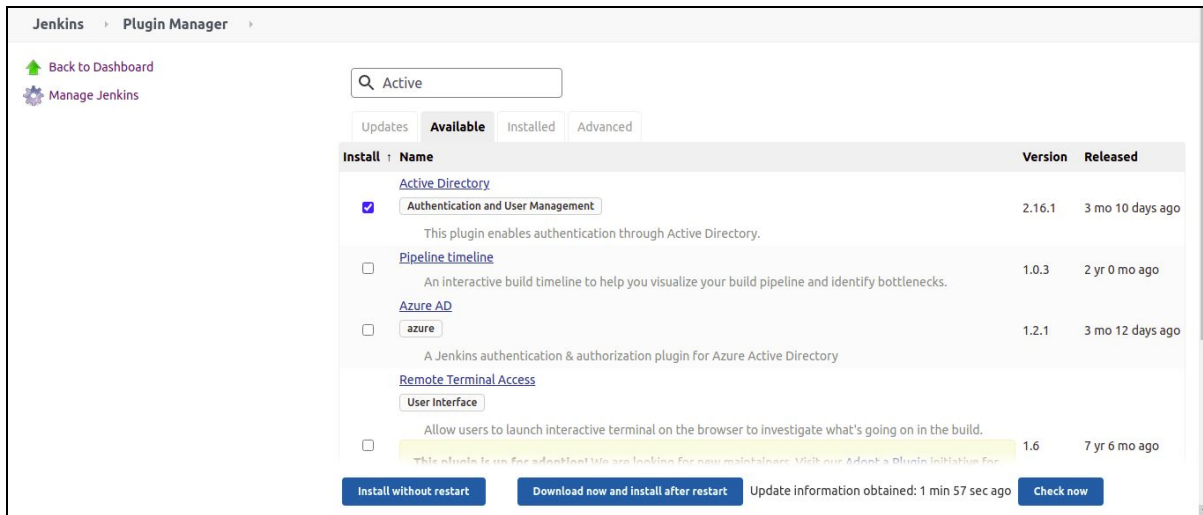
Delete Maven

**Maven is successfully configured.**

# ACTIVE DIRECTORY

## Step-1 Install Plugin Active Directory

Select **Manage Jenkins** on the left panel in Jenkins. Then select **Manage Plugins**. Here we are installing the plugin **Active Directory** in **Available** section it is necessary which will allow Jenkins to use **LDAP protocol** and hence you will be able to authenticate using Jenkins.



The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected, and a search for 'Active' has been performed. The following table lists the available plugins:

Install	Name	Version	Released
<input checked="" type="checkbox"/>	<a href="#">Active Directory</a> Authentication and User Management This plugin enables authentication through Active Directory.	2.16.1	3 mo 10 days ago
<input type="checkbox"/>	<a href="#">Pipeline timeline</a> An interactive build timeline to help you visualize your build pipeline and identify bottlenecks.	1.0.3	2 yr 0 mo ago
<input type="checkbox"/>	<a href="#">Azure AD</a> azure A Jenkins authentication & authorization plugin for Azure Active Directory	1.2.1	3 mo 12 days ago
<input type="checkbox"/>	<a href="#">Remote Terminal Access</a> User Interface Allow users to launch interactive terminal on the browser to investigate what's going on in the build. This plugin is in pre-alpha state. We are looking for new maintainers. Visit our Adopt a Plugin initiative for more details.	1.6	7 yr 6 mo ago

At the bottom, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. The update information is noted as 'Update information obtained: 1 min 57 sec ago'.

## Step-2 Let's setup our security realms for Jenkins to use AD.

Select **Manage Jenkins** on the left panel in Jenkins. In **Security** sections. Then Select **Configure Global Security**.

Here we have various options to select the database which in our case is **Active Directory**.

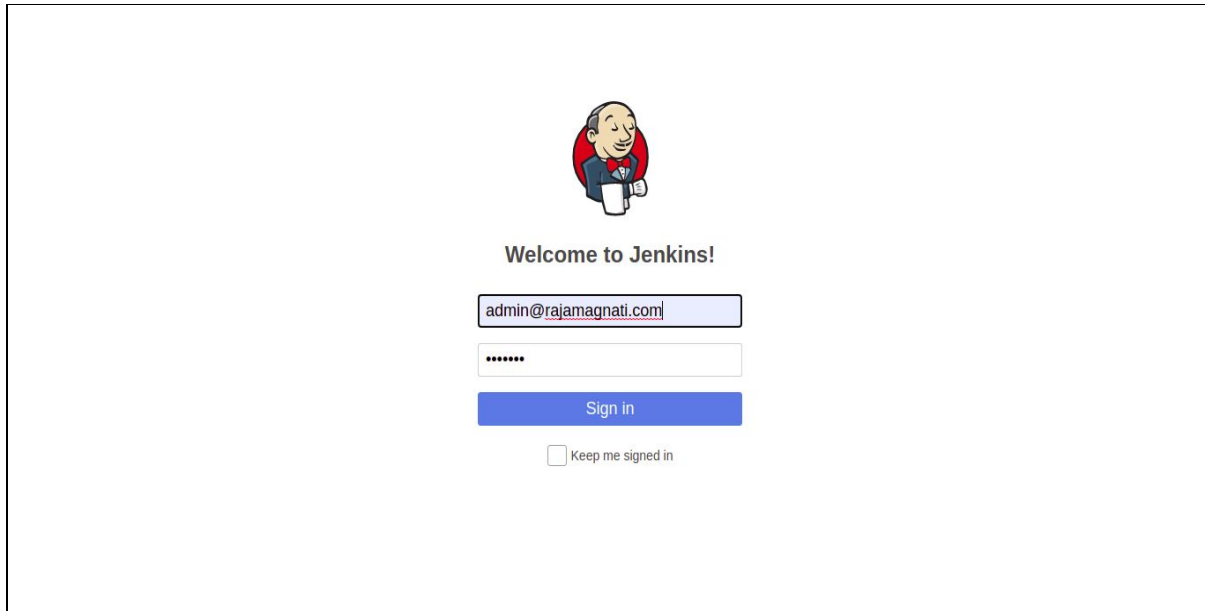
**Domain Name** must be the actual domain which we have used in **Active Directory**.

In the **Domain Controller** we need to specify where our **AD DS** is listening. The **IP Address** of Instance where **AD DS** is installed in our case is **52.66.120.158** and port must be **3268**.

The screenshot shows the 'Jenkins > Configure Global Security' page. Under the 'Authentication' tab, the 'Security Realm' is set to 'Active Directory'. The 'Domains' section contains the following fields: 'Domain Name' (rajamagnati.com), 'Domain controller' (52.66.120.158:3268), 'Site' (empty), 'Bind DN' (empty), and 'Bind Password' (empty). The 'TLS Configuration' dropdown is set to '(Insecure) Trust all Certificates'. A warning message states: 'Leaving blank 'Bind DN' means that any operation performed will use anonymous binding. Keep in mind that this is not recommended as some servers do not allow it by default.' There are 'Test Domain' and 'Delete Domain' buttons. At the bottom, there are 'Save' and 'Apply' buttons.

### Step-3 Login to Jenkins using AD DS credentials

Here you will use the username and Password of AD when you set during setup of AD DS. We are using these user credentials to log into the system.



The image shows the Jenkins login page. At the top center is the Jenkins logo, a cartoon character with a red bow tie. Below the logo is the text "Welcome to Jenkins!". Underneath is a text input field containing the email address "admin@rajamagnati.com". Below that is a password input field with masked characters "\*\*\*\*\*". A blue "Sign in" button is positioned below the password field. At the bottom, there is a checkbox labeled "Keep me signed in".

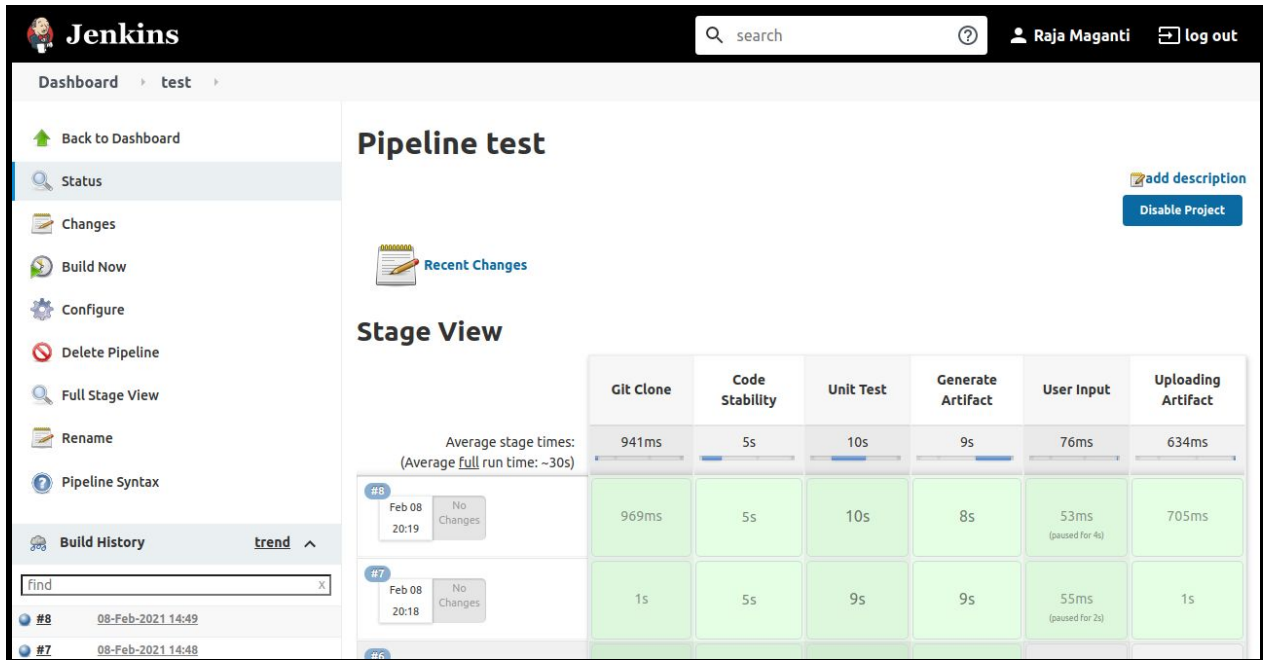
### Step-4 Successful authentication

Now using admin credentials we have accessed jenkins.



# Execution of new job to test Configuration

<http://18.220.129.11:8080/>



The screenshot shows the Jenkins web interface for a pipeline named 'test'. The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main content area is titled 'Pipeline test' and includes a 'Recent Changes' section and a 'Stage View' table. The table displays stage durations for builds #6, #7, and #8. Build #8 is the most recent, showing a full run time of approximately 30 seconds. The 'Disable Project' button is visible in the top right corner.

	Git Clone	Code Stability	Unit Test	Generate Artifact	User Input	Uploading Artifact
Average stage times: (Average full run time: ~30s)	941ms	5s	10s	9s	76ms	634ms
#8 Feb 08 20:19 No Changes	969ms	5s	10s	8s	53ms (paused for 4s)	705ms
#7 Feb 08 20:18 No Changes	1s	5s	9s	9s	55ms (paused for 2s)	1s
#6						

Artifact in jfrog

Happily serving 34 artifacts

Set Me Up








Deploy

Filter repositories

Clear

★ My Favorites ①

Tree View:  

- >  artifactory-build-info
- >  example-repo-local
- >  maven-repo-local
  - >  newjenkins
    - >  BuildArtifact\_8
      - >  java-maven-junit-hellow
  - >  testpoc

 maven-repo-local

Refresh Star Menu

General

Effective Permissions

Properties

Followers

Info

Name: maven-repo-local 

Package Type:  Maven

Repository Path: maven-repo-local/ 