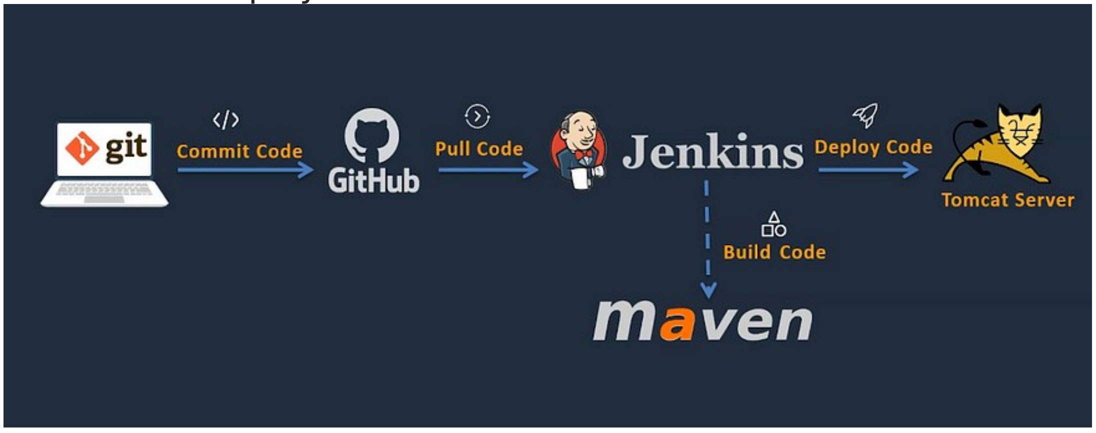
In this blog, we are going to deploy a Java Web app using Maven on a remote Tomcat Server built on an EC2 Instance through the use of Jenkins.

**Agenda:**

- Setup Jenkins
- Setup & Configure Maven and Git
- Setup Tomcat Server
- Integrating GitHub, Maven, and Tomcat Server with Jenkins
- Create a CI and CD job
- Test the deployment



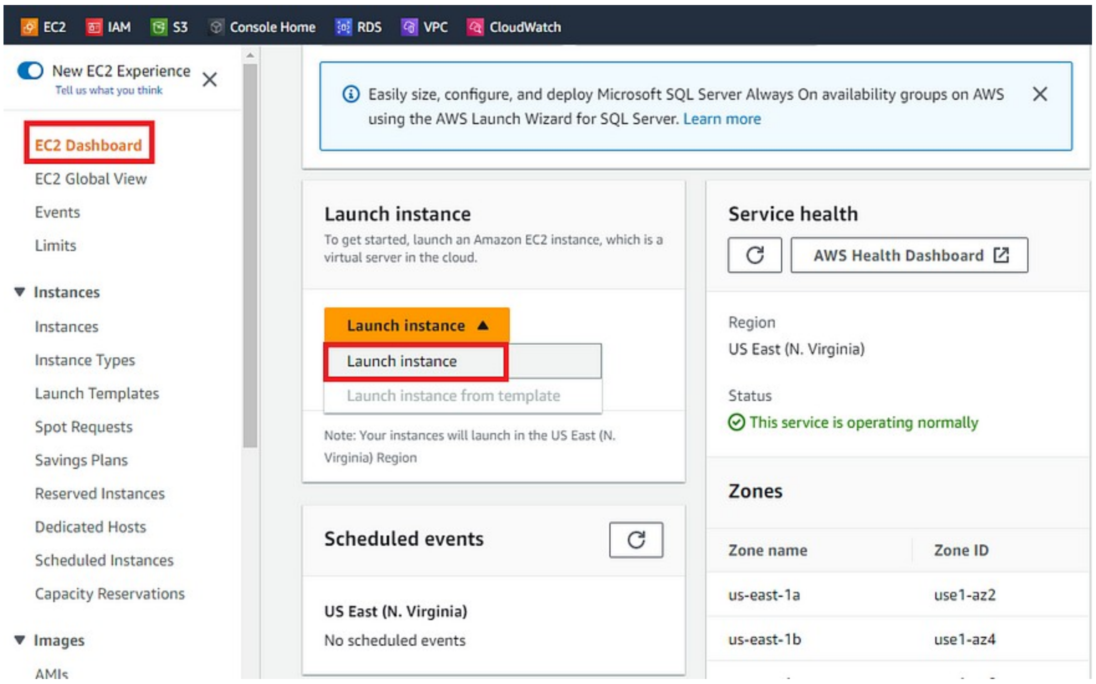
**Prerequisites:**

- AWS Account
- Git/ Github Account with the Source Code
- A local machine with CLI Access

Step 1: Setup Jenkins Server on AWS EC2 Instance

- Setup a Linux EC2 Instance
- Install Java
- Install Jenkins
- Start Jenkins
- Access Web UI on port 8080

Log in to the Amazon management console, open EC2 Dashboard, click on the Launch Instance drop-down list, and click on Launch Instance as shown below:



Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the US East (N. Virginia) Region

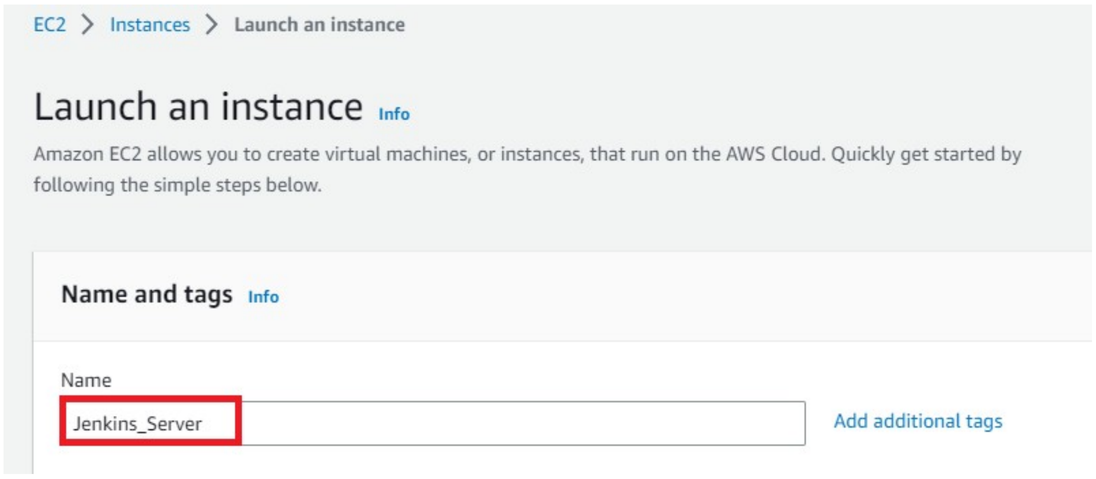
Region: US East (N. Virginia)

Status: This service is operating normally

US East (N. Virginia)
No scheduled events

| Zone name | Zone ID |
|---|---|
| us-east-1a | use1-az2 |
| us-east-1b | use1-az4 |

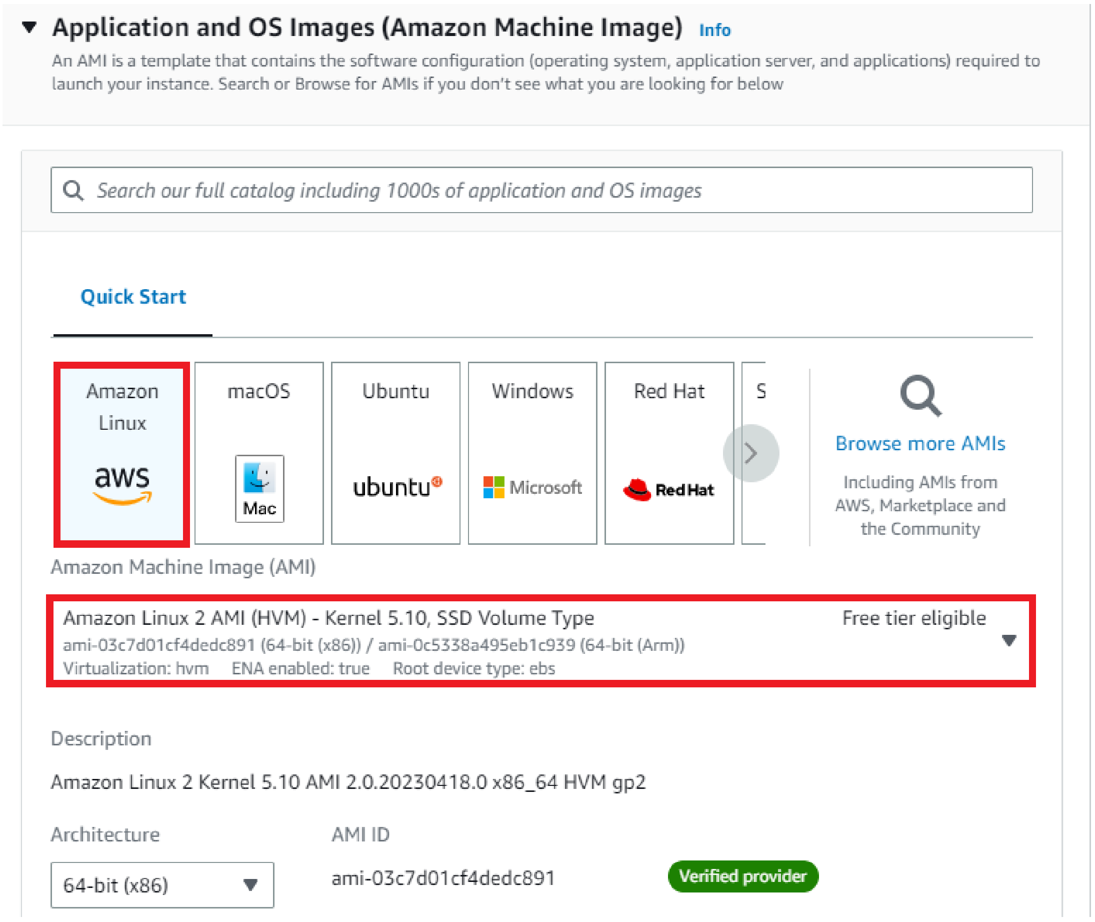
Once the **Launch an instance** window opens, provide the name of your EC2 Instance:



Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name: Jenkins_Server

For this demo, we will select Amazon Linux 2 AMI which is free tier eligible.


Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-03c7d01cf4dedc891 (64-bit (x86)) / ami-0c5338a495eb1c939 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

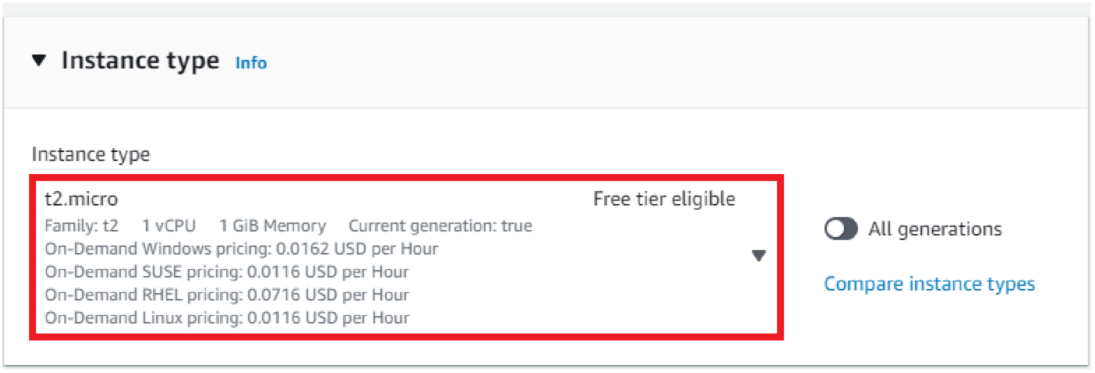
Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20230418.0 x86_64 HVM gp2

Architecture: 64-bit (x86)

AMI ID: ami-03c7d01cf4dedc891 — Verified provider

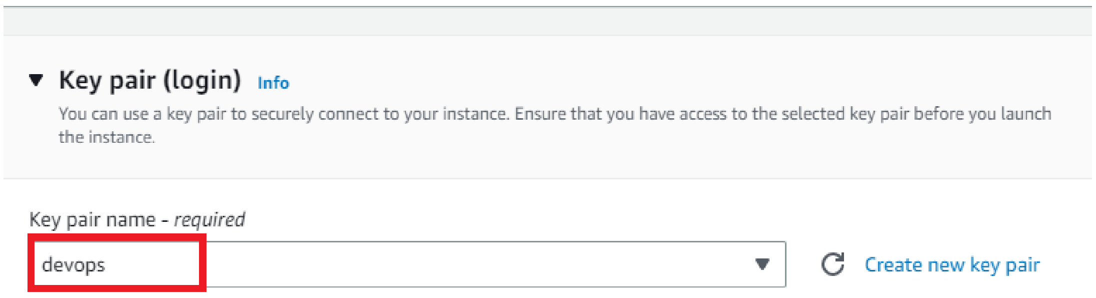
Choose an Instance Type. Here you can select the type of machine, number of vCPUs, and memory that you want to have. Select **t2.micro** which is free-tier eligible.


Instance type Info

Instance type

t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows pricing: 0.0162 USD per Hour
On-Demand SUSE pricing: 0.0116 USD per Hour
On-Demand RHEL pricing: 0.0716 USD per Hour
On-Demand Linux pricing: 0.0116 USD per Hour

For this demo, we will select an already existing key pair. You can create new key pair if you don’t have:

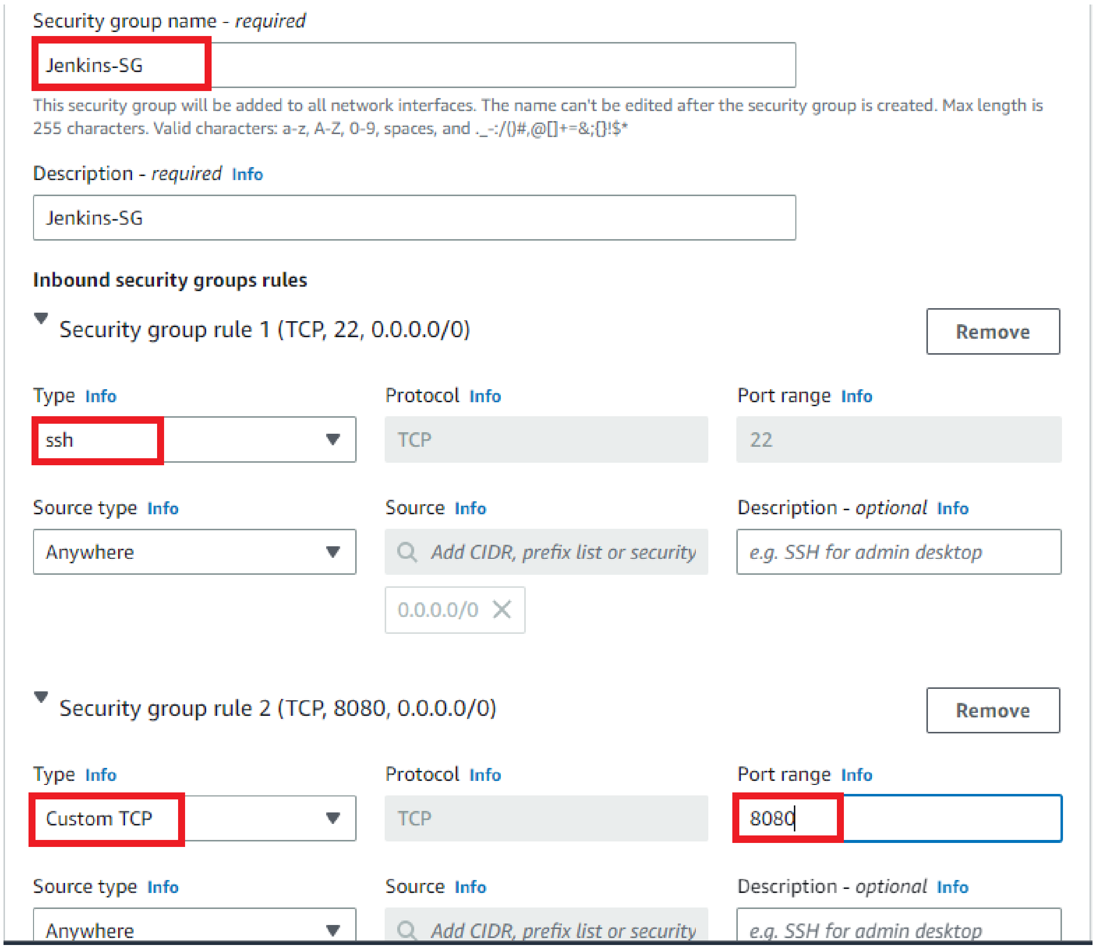

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

devops

Now under **Network Settings**, Choose the default VPC with Auto-assign public IP in enable mode. Create a new Security Group, provide a name for your security group, allow ssh traffic, and custom default TCP port of 8080 which is used by Jenkins.



Security group name - *required*

Jenkins-SG

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* Info

Jenkins-SG

**Inbound security groups rules**

Security group rule 1 (TCP, 22, 0.0.0.0/0)

| Type | Protocol | Port range |
|---|---|---|
| ssh | TCP | 22 |

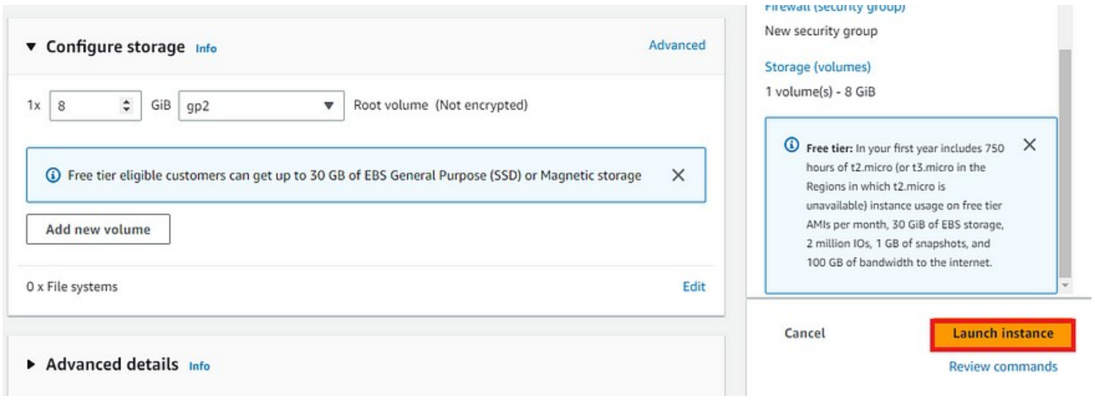
| Source type | Source | Description - *optional* |
|---|---|---|
| Anywhere | *Add CIDR, prefix list or security* 0.0.0.0/0 | *e.g. SSH for admin desktop* |

Security group rule 2 (TCP, 8080, 0.0.0.0/0)

| Type | Protocol | Port range |
|---|---|---|
| Custom TCP | TCP | 8080 |

| Source type | Source | Description - *optional* |
|---|---|---|
| Anywhere | *Add CIDR, prefix list or security* | *e.g. SSH for admin desktop* |

Rest of the settings we will keep them at default and go ahead and click on **Launch Instance**



**Configure storage** Info

1x 8 GiB gp2 Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

0 x File systems

**Advanced details** Info

Firewall (security group)

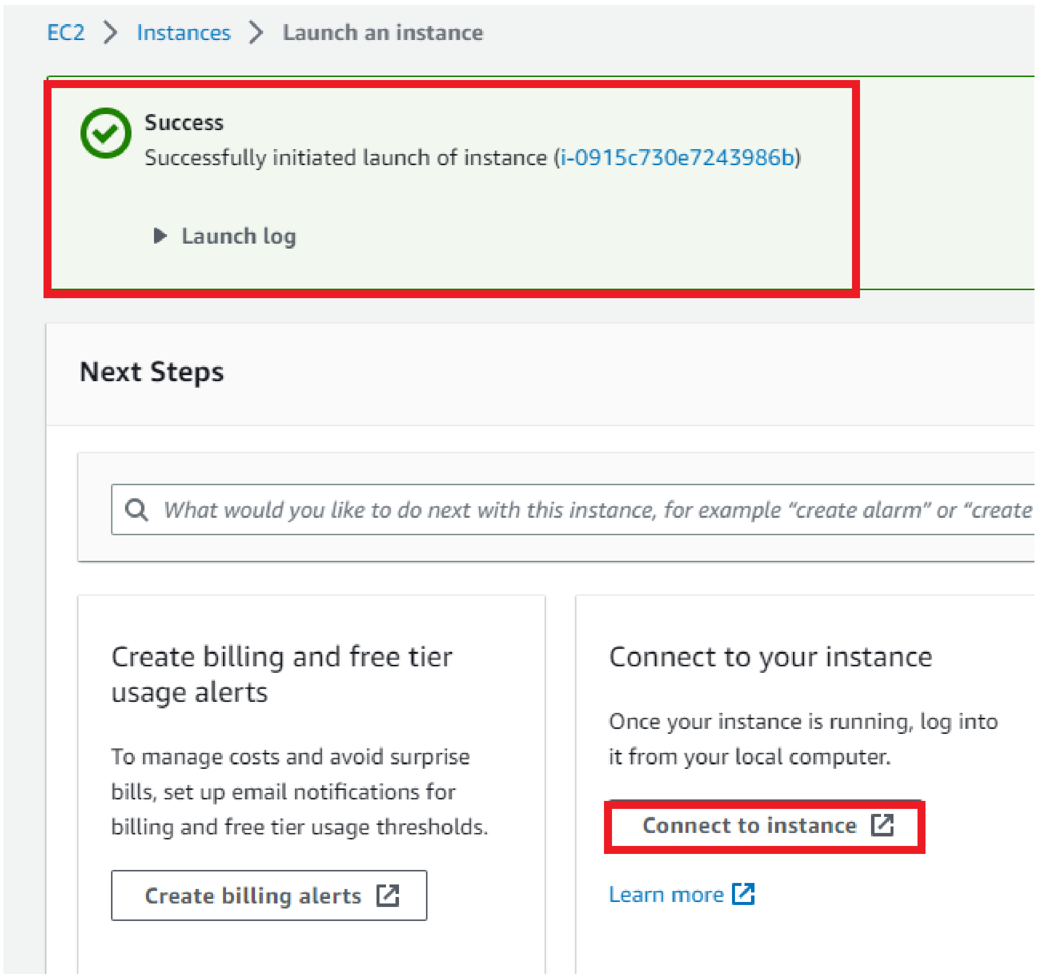
New security group

Storage (volumes)

1 volume(s) - 8 GiB

**Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

On the next screen you can see a success message after the successful creation of the EC2 instance, click on Connect to instance button:


EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-0915c730e7243986b)

Launch log

Next Steps

What would you like to do next with this instance, for example "create alarm" or "create

Create billing and free tier usage alerts

To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.

Create billing alerts

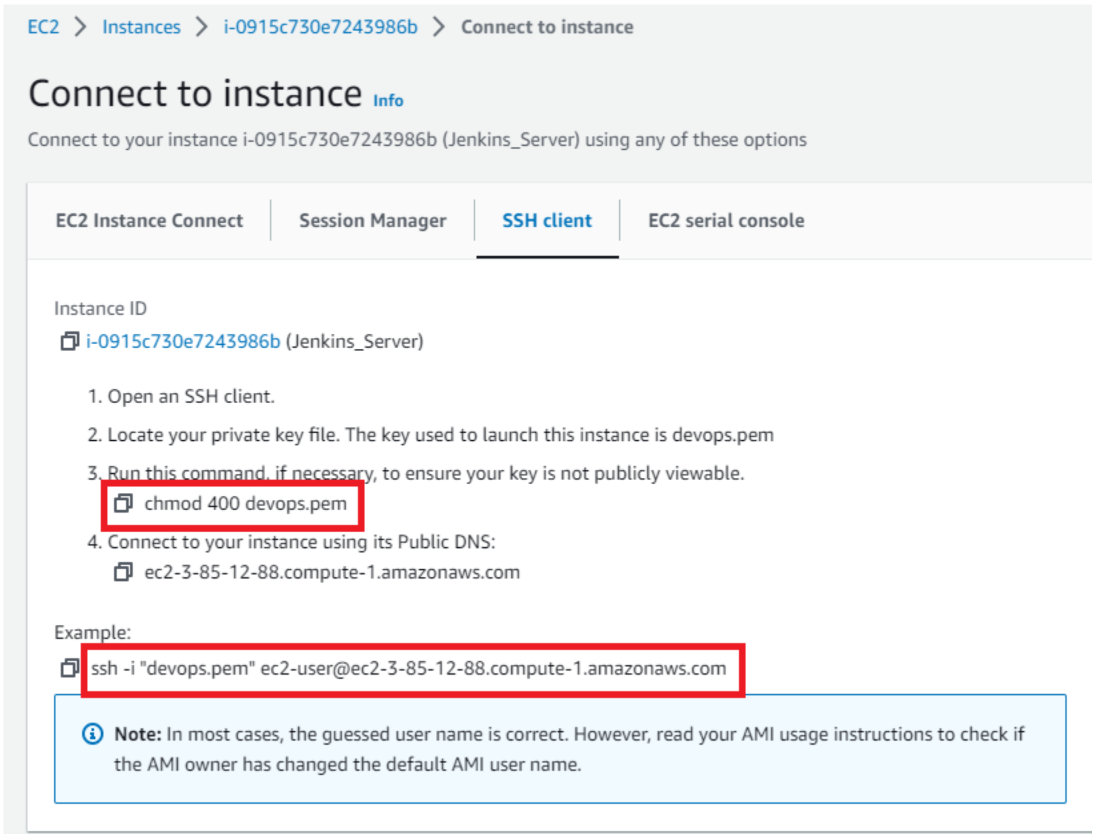
Connect to your instance

Once your instance is running, log into it from your local computer.

Connect to instance

Learn more

Now **connect to instance** wizard will open, go to SSH client tab and copy the provided chmod and SSH command:


EC2 > Instances > i-0915c730e7243986b > Connect to instance

Connect to instance Info

Connect to your instance i-0915c730e7243986b (Jenkins_Server) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID

i-0915c730e7243986b (Jenkins_Server)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is devops.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 devops.pem
```

4. Connect to your instance using its Public DNS:

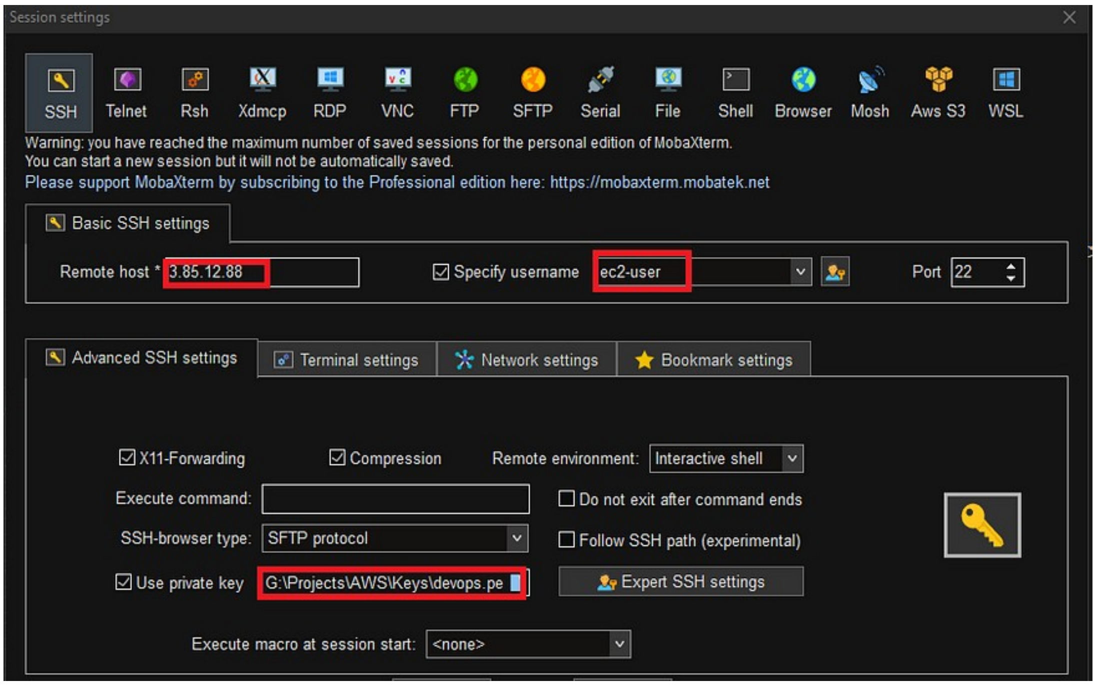
```
ec2-3-85-12-88.compute-1.amazonaws.com
```

Example:

```
ssh -i "devops.pem" ec2-user@ec2-3-85-12-88.compute-1.amazonaws.com
```

**Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Open any SSH Client in your local machine, take the public IP of your EC2 Instance, and add the pem key and you will be able to access your EC2 machine in my case I am using MobaXterm on Windows:



Warning: you have reached the maximum number of saved sessions for the personal edition of MobaXterm.
You can start a new session but it will not be automatically saved.
Please support MobaXterm by subscribing to the Professional edition here: https://mobaxterm.mobatek.net

Basic SSH settings

Remote host * 3.85.12.88 ☑ Specify username ec2-user Port 22

Advanced SSH settings | Terminal settings | Network settings | Bookmark settings

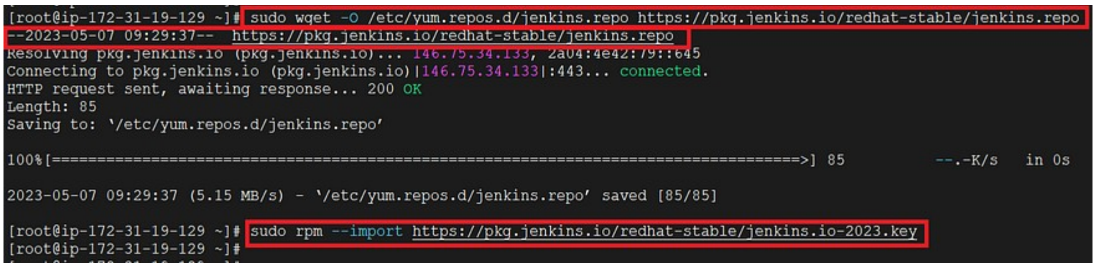
☑ X11-Forwarding ☑ Compression Remote environment: Interactive shell
Execute command: ☐ Do not exit after command ends
SSH-browser type: SFTP protocol ☐ Follow SSH path (experimental)
☑ Use private key G:\Projects\AWS\Keys\devops.pe Expert SSH settings
Execute macro at session start: <none>

After logging in to our EC2 machine we will install Jenkins following the instructions from the official Jenkins website:
https://pkg.jenkins.io/redhat-stable/

To use this repository, run the following command:

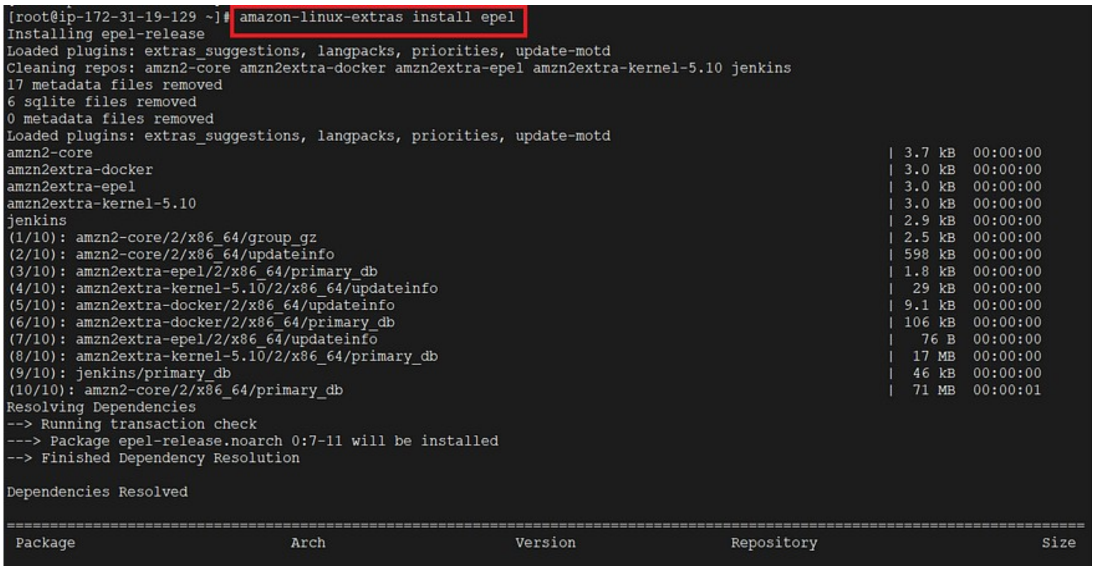
```
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

Output:



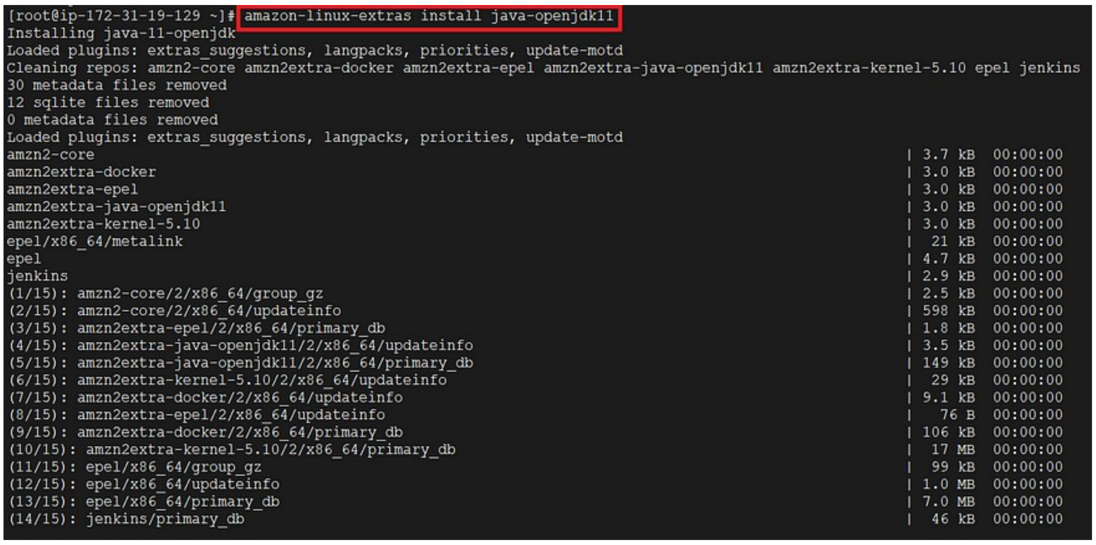
```
[root@ip-172-31-19-129 ~]# sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2023-05-07 09:29:37--  https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.34.133, 2a04:4e42:79::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|146.75.34.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

100%[=====================================================================================>] 85          --.-K/s   in 0s

2023-05-07 09:29:37 (5.15 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[root@ip-172-31-19-129 ~]# sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[root@ip-172-31-19-129 ~]#
```

Now let’s install epel packages for Amazon Linux AMI:



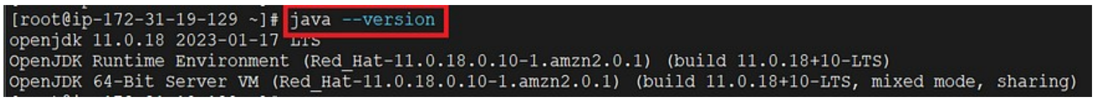
```
[root@ip-172-31-19-129 ~]# amazon-linux-extras install epel
Installing epel-release
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-epel amzn2extra-kernel-5.10 jenkins
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                                    | 3.7 kB  00:00:00
amzn2extra-docker                                                             | 3.0 kB  00:00:00
amzn2extra-epel                                                               | 3.0 kB  00:00:00
amzn2extra-kernel-5.10                                                        | 3.0 kB  00:00:00
jenkins                                                                       | 2.9 kB  00:00:00
(1/10): amzn2-core/2/x86_64/group_gz                                          | 2.5 kB  00:00:00
(2/10): amzn2-core/2/x86_64/updateinfo                                        | 598 kB  00:00:00
(3/10): amzn2extra-epel/2/x86_64/primary_db                                   | 1.8 kB  00:00:00
(4/10): amzn2extra-kernel-5.10/2/x86_64/updateinfo                            |  29 kB  00:00:00
(5/10): amzn2extra-docker/2/x86_64/updateinfo                                 | 9.1 kB  00:00:00
(6/10): amzn2extra-docker/2/x86_64/primary_db                                 | 106 kB  00:00:00
(7/10): amzn2extra-epel/2/x86_64/updateinfo                                   |   76 B  00:00:00
(8/10): amzn2extra-kernel-5.10/2/x86_64/primary_db                            |  17 MB  00:00:00
(9/10): jenkins/primary_db                                                    |  46 kB  00:00:00
(10/10): amzn2-core/2/x86_64/primary_db                                       |  71 MB  00:00:01
Resolving Dependencies
--> Running transaction check
---> Package epel-release.noarch 0:7-11 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================================================
 Package                        Arch                     Version                  Repository          Size
```

After installing epel packages, let’s install java-openjdk11:



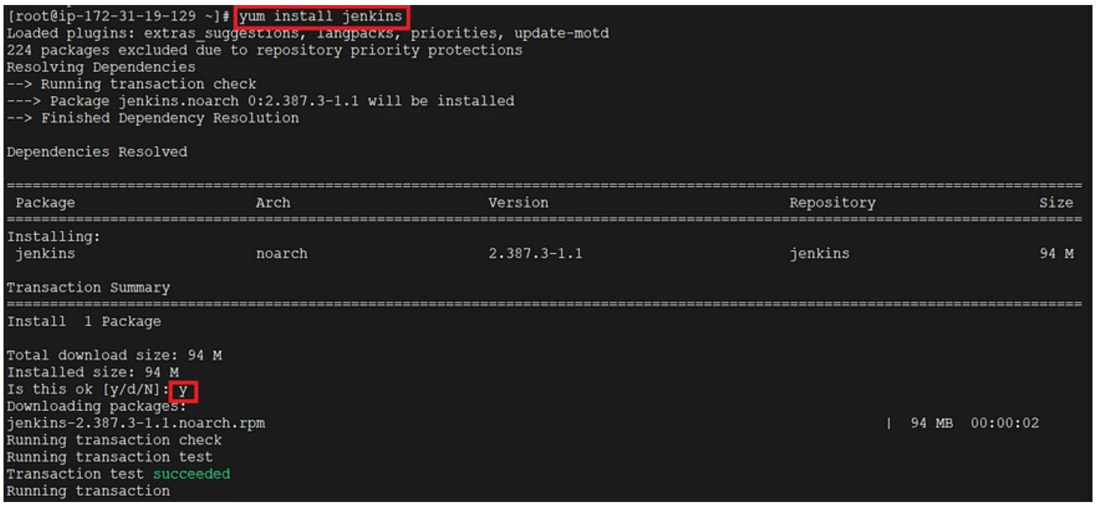
```
[root@ip-172-31-19-129 ~]# amazon-linux-extras install java-openjdk11
Installing java-11-openjdk
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-epel amzn2extra-java-openjdk11 amzn2extra-kernel-5.10 epel jenkins
30 metadata files removed
12 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                                      | 3.7 kB  00:00:00
amzn2extra-docker                                                               | 3.0 kB  00:00:00
amzn2extra-epel                                                                 | 3.0 kB  00:00:00
amzn2extra-java-openjdk11                                                       | 3.0 kB  00:00:00
amzn2extra-kernel-5.10                                                          | 3.0 kB  00:00:00
epel/x86_64/metalink                                                            |  21 kB  00:00:00
epel                                                                            | 4.7 kB  00:00:00
jenkins                                                                         | 2.9 kB  00:00:00
(1/15): amzn2-core/2/x86_64/group_gz                                            | 2.5 kB  00:00:00
(2/15): amzn2-core/2/x86_64/updateinfo                                          | 598 kB  00:00:00
(3/15): amzn2extra-epel/2/x86_64/primary_db                                     | 1.8 kB  00:00:00
(4/15): amzn2extra-java-openjdk11/2/x86_64/updateinfo                           | 3.5 kB  00:00:00
(5/15): amzn2extra-java-openjdk11/2/x86_64/primary_db                           | 149 kB  00:00:00
(6/15): amzn2extra-kernel-5.10/2/x86_64/updateinfo                              |  29 kB  00:00:00
(7/15): amzn2extra-docker/2/x86_64/updateinfo                                   | 9.1 kB  00:00:00
(8/15): amzn2extra-epel/2/x86_64/updateinfo                                     |   76 B  00:00:00
(9/15): amzn2extra-docker/2/x86_64/primary_db                                   | 106 kB  00:00:00
(10/15): amzn2extra-kernel-5.10/2/x86_64/primary_db                             |  17 MB  00:00:00
(11/15): epel/x86_64/group_gz                                                   |  99 kB  00:00:00
(12/15): epel/x86_64/updateinfo                                                 | 1.0 MB  00:00:00
(13/15): epel/x86_64/primary_db                                                 | 7.0 MB  00:00:00
(14/15): jenkins/primary_db                                                     |  46 kB  00:00:00
```

Let’s check the version of Java now:



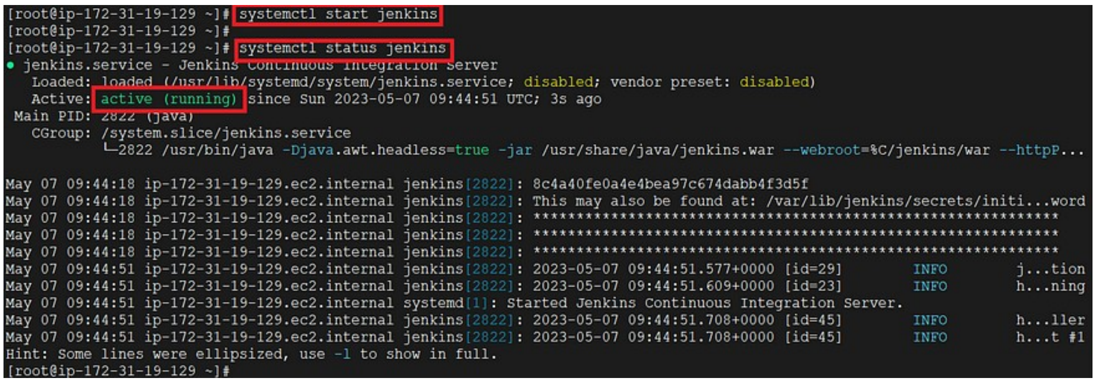
```
[root@ip-172-31-19-129 ~]# java --version
openjdk 11.0.18 2023-01-17 LTS
OpenJDK Runtime Environment (Red_Hat-11.0.18.0.10-1.amzn2.0.1) (build 11.0.18+10-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.18.0.10-1.amzn2.0.1) (build 11.0.18+10-LTS, mixed mode, sharing)
```

Now let's install Jenkins with the below command as shown in the output:



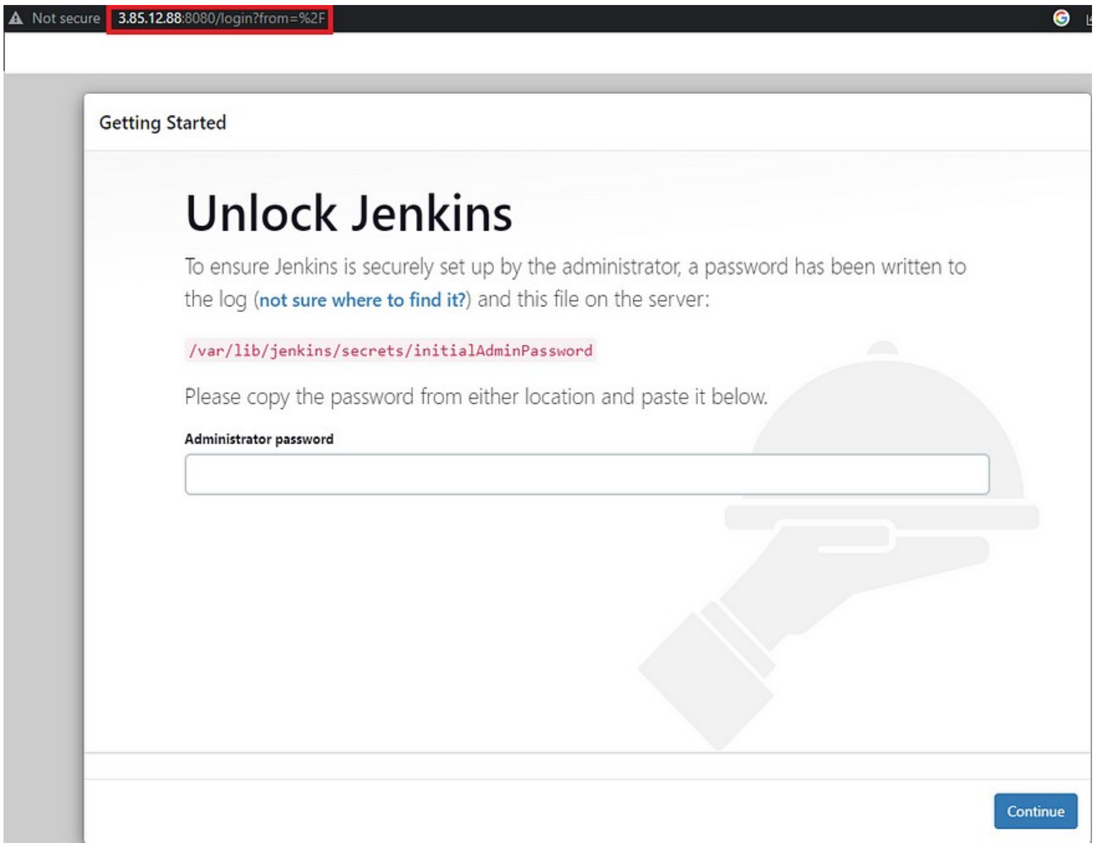
```
[root@ip-172-31-19-129 ~]# yum install jenkins
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
224 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package jenkins.noarch 0:2.387.3-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

===================================================================================================================
 Package                    Arch                         Version                         Repository            Size
===================================================================================================================
Installing:
 jenkins                    noarch                       2.387.3-1.1                     jenkins               94 M

Transaction Summary
===================================================================================================================
Install  1 Package

Total download size: 94 M
Installed size: 94 M
Is this ok [y/d/N]: y
Downloading packages:
jenkins-2.387.3-1.1.noarch.rpm                                                                |  94 MB  00:00:02
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
```

After successful installation Let's enable and start Jenkins service in our EC2 Instance:



```
[root@ip-172-31-19-129 ~]# systemctl start jenkins
[root@ip-172-31-19-129 ~]#
[root@ip-172-31-19-129 ~]# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; vendor preset: disabled)
   Active: active (running) since Sun 2023-05-07 09:44:51 UTC; 3s ago
 Main PID: 2822 (java)
   CGroup: /system.slice/jenkins.service
           └─2822 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpP...

May 07 09:44:18 ip-172-31-19-129.ec2.internal jenkins[2822]: 8c4a40fe0a4e4bea97c674dabb4f3d5f
May 07 09:44:18 ip-172-31-19-129.ec2.internal jenkins[2822]: This may also be found at: /var/lib/jenkins/secrets/initi...word
May 07 09:44:18 ip-172-31-19-129.ec2.internal jenkins[2822]: *************************************************************
May 07 09:44:18 ip-172-31-19-129.ec2.internal jenkins[2822]: *************************************************************
May 07 09:44:18 ip-172-31-19-129.ec2.internal jenkins[2822]: *************************************************************
May 07 09:44:51 ip-172-31-19-129.ec2.internal jenkins[2822]: 2023-05-07 09:44:51.577+0000 [id=29]        INFO       j...tion
May 07 09:44:51 ip-172-31-19-129.ec2.internal jenkins[2822]: 2023-05-07 09:44:51.609+0000 [id=23]        INFO       h...ning
May 07 09:44:51 ip-172-31-19-129.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
May 07 09:44:51 ip-172-31-19-129.ec2.internal jenkins[2822]: 2023-05-07 09:44:51.708+0000 [id=45]        INFO       h...ller
May 07 09:44:51 ip-172-31-19-129.ec2.internal jenkins[2822]: 2023-05-07 09:44:51.708+0000 [id=45]        INFO       h...t #1
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-172-31-19-129 ~]#
```

Now let’s try to access the Jenkins server through our browser. For that take the public IP of your EC2 instance and paste it into your favorite browser and should see something like this:


Not secure 3.85.12.88:8080/login?from=%2F

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

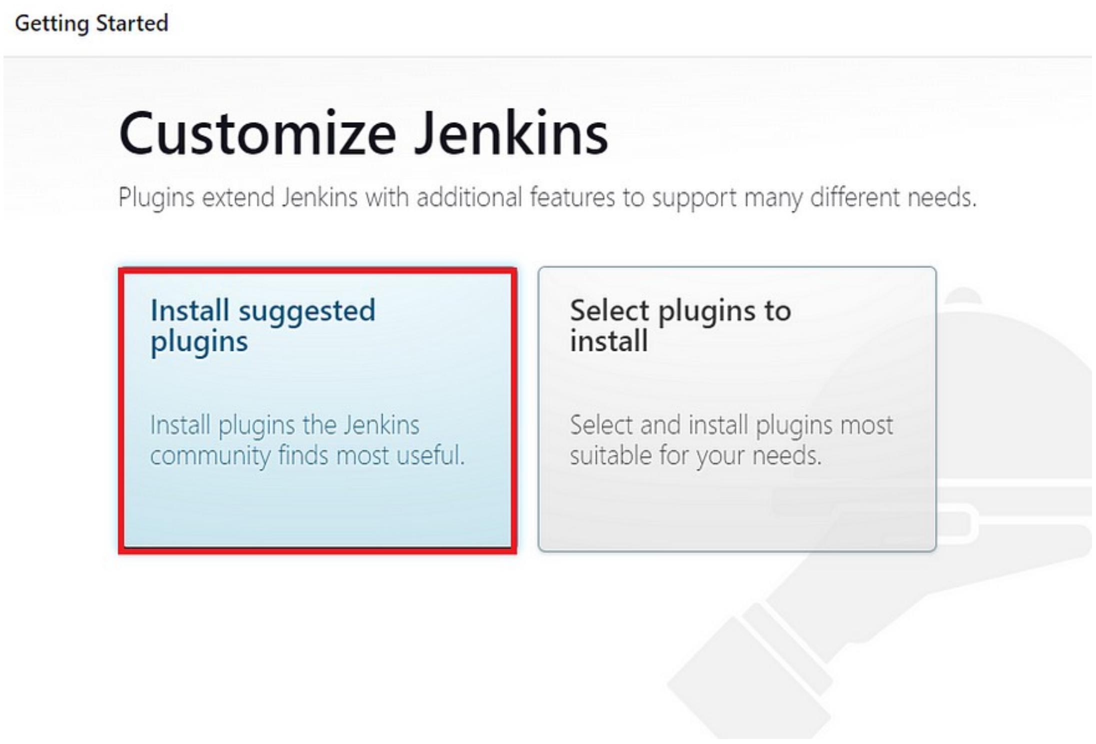
Please copy the password from either location and paste it below.

Administrator password

To unlock Jenkins we need to go to the path /var/lib/jenkins/secrets/initialAdminPassword and fetch the admin password to proceed further:

```
[root@ip-172-31-19-129 ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
8c4a40fe0a4e4bea97c674dabb4f3d5f
```

Now on the Customize Jenkins page, we can go ahead and install the suggested plugins:


Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

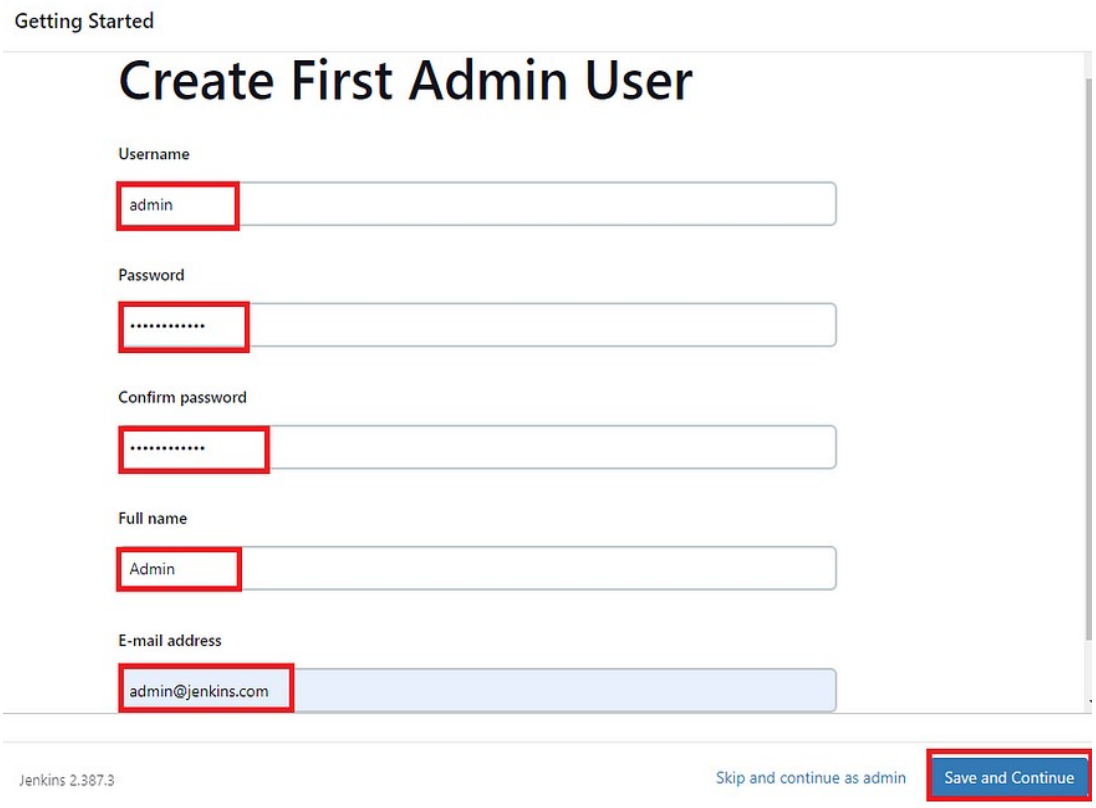
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Now we can create our first Admin user, provide all the required data and proceed to save and continue.


Getting Started

Create First Admin User

Username: admin

Password: ••••••••••••

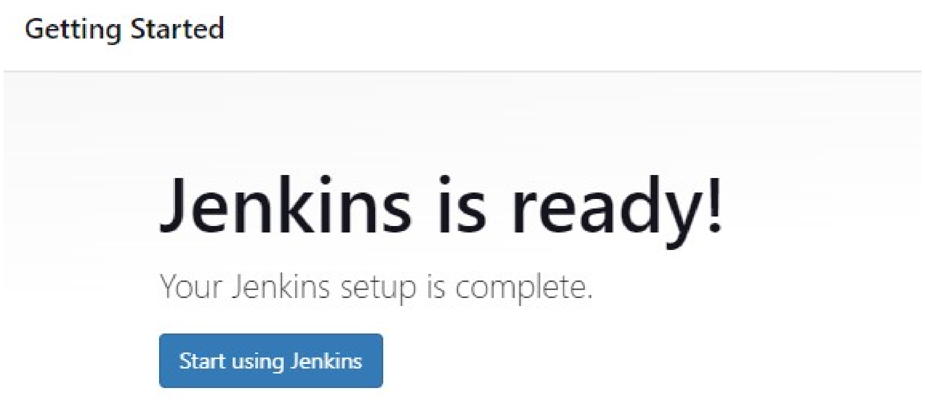
Confirm password: ••••••••••••

Full name: Admin

E-mail address: admin@jenkins.com

Jenkins 2.387.3

Now we are ready to use our Jenkins Server.


Getting Started

Jenkins is ready!

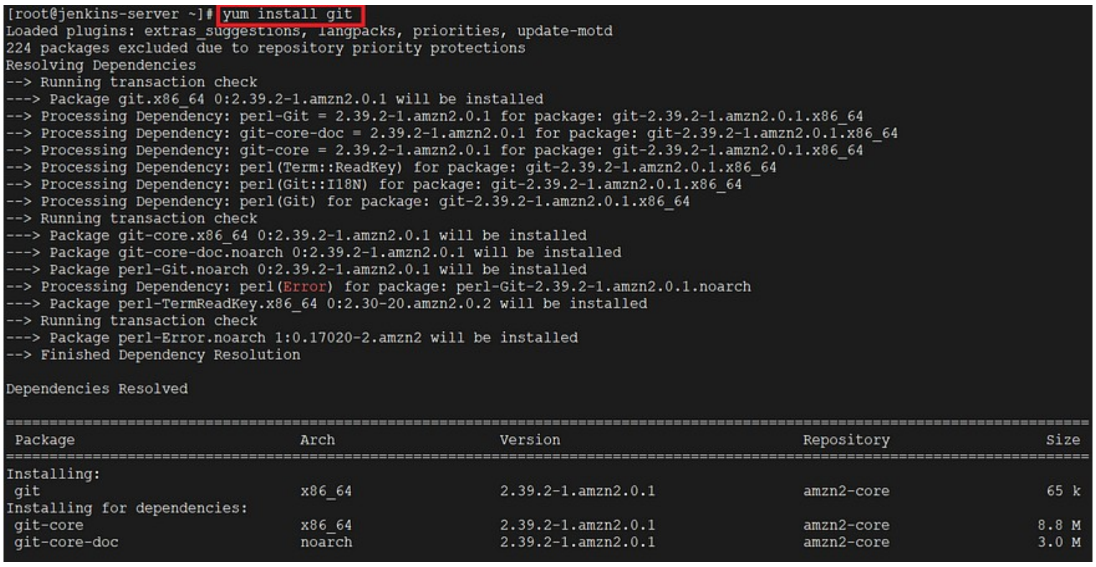
Your Jenkins setup is complete.

Start using Jenkins

Step 2: Integrate GitHub with Jenkins

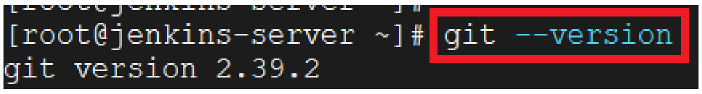
- •Install Git on Jenkins Instance
- •Install Github Plugin on Jenkins GUI
- •Configure Git on Jenkins GUI

Let’s first install Git on our EC2 instance with the below command:



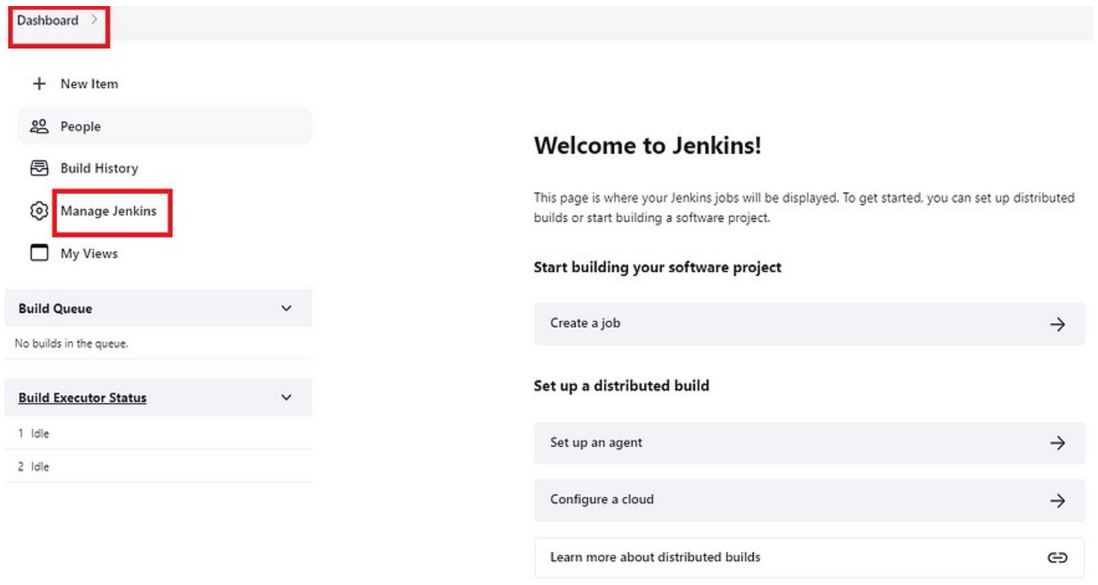
```
[root@jenkins-server ~]# yum install git
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
224 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package git.x86_64 0:2.39.2-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-Git = 2.39.2-1.amzn2.0.1 for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core-doc = 2.39.2-1.amzn2.0.1 for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core = 2.39.2-1.amzn2.0.1 for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: perl(Term::ReadKey) for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: perl(Git::I18N) for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: perl(Git) for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Running transaction check
---> Package git-core.x86_64 0:2.39.2-1.amzn2.0.1 will be installed
---> Package git-core-doc.noarch 0:2.39.2-1.amzn2.0.1 will be installed
---> Package perl-Git.noarch 0:2.39.2-1.amzn2.0.1 will be installed
--> Processing Dependency: perl(Error) for package: perl-Git-2.39.2-1.amzn2.0.1.noarch
---> Package perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2 will be installed
--> Running transaction check
---> Package perl-Error.noarch 1:0.17020-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================
 Package                     Arch               Version                          Repository                Size
================================================================================================================
Installing:
 git                         x86_64             2.39.2-1.amzn2.0.1               amzn2-core                65 k
Installing for dependencies:
 git-core                    x86_64             2.39.2-1.amzn2.0.1               amzn2-core               8.8 M
 git-core-doc                noarch             2.39.2-1.amzn2.0.1               amzn2-core               3.0 M
```

We can check the version as shown in the below screenshot:



```
[root@jenkins-server ~]# git --version
git version 2.39.2
```

To install the GitHub plugin lets go to our Jenkins Dashboard and click on manage Jenkins as shown:



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

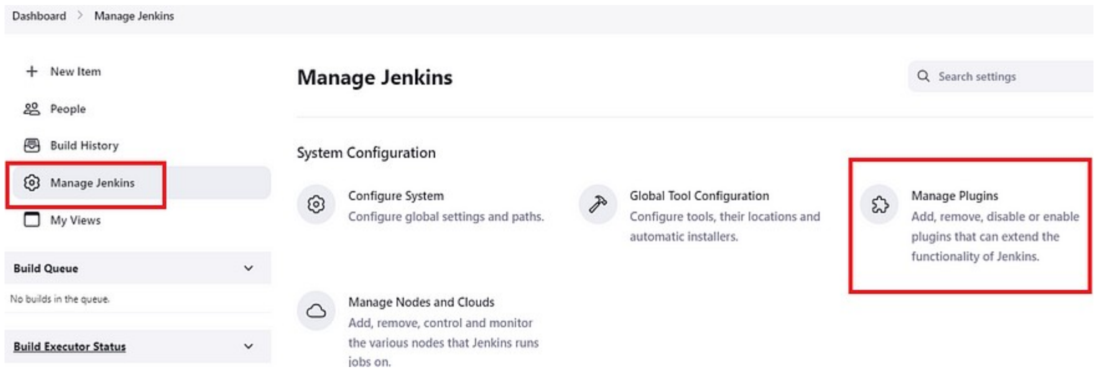
Set up a distributed build

No builds in the queue.

1 Idle

2 Idle

On the next page, click on manage plugins:



Manage Jenkins

System Configuration

Configure System
Configure global settings and paths.

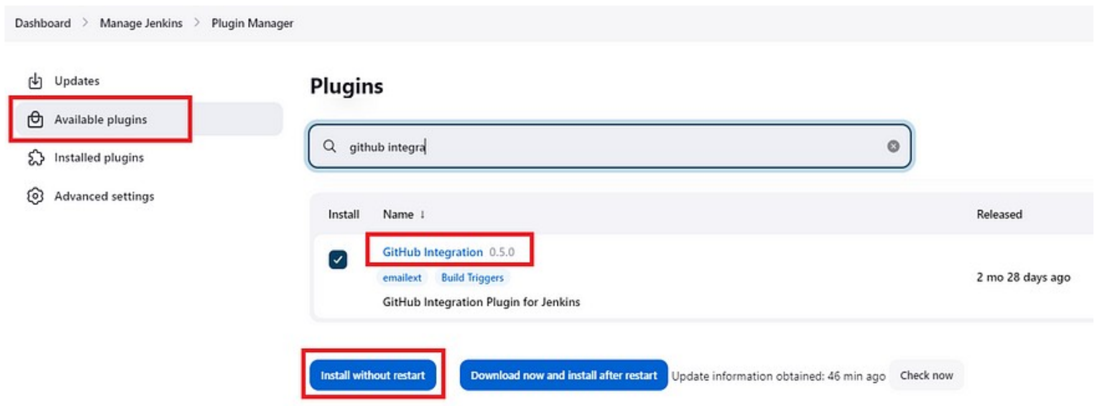
Global Tool Configuration
Configure tools, their locations and automatic installers.

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.

No builds in the queue.

Now in order to install any plugin we need to select Available Plugins, search for Github Integration, select the plugin, and finally click on **Install without restart** as shown below:



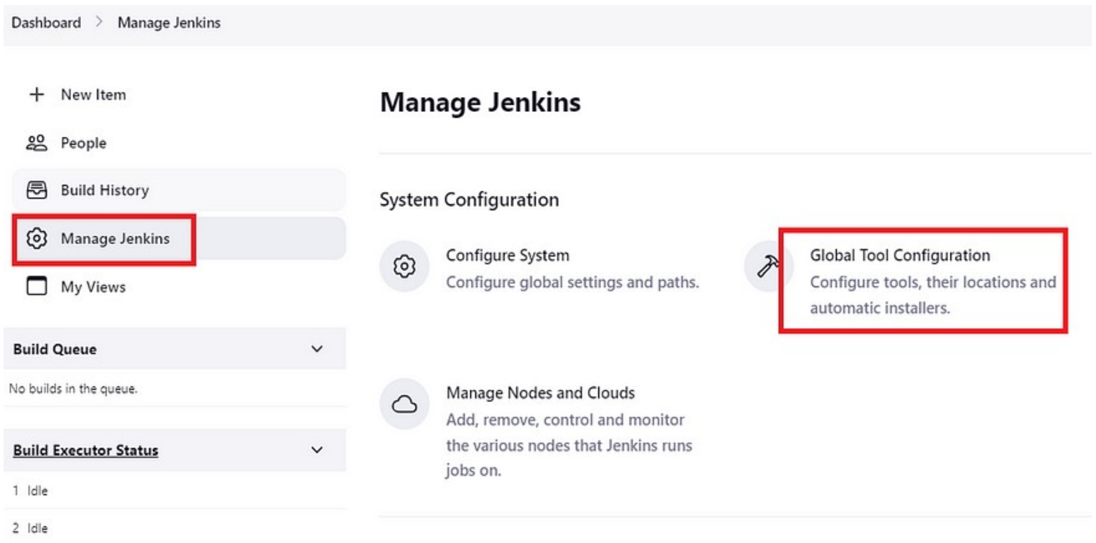
Plugins

github integra

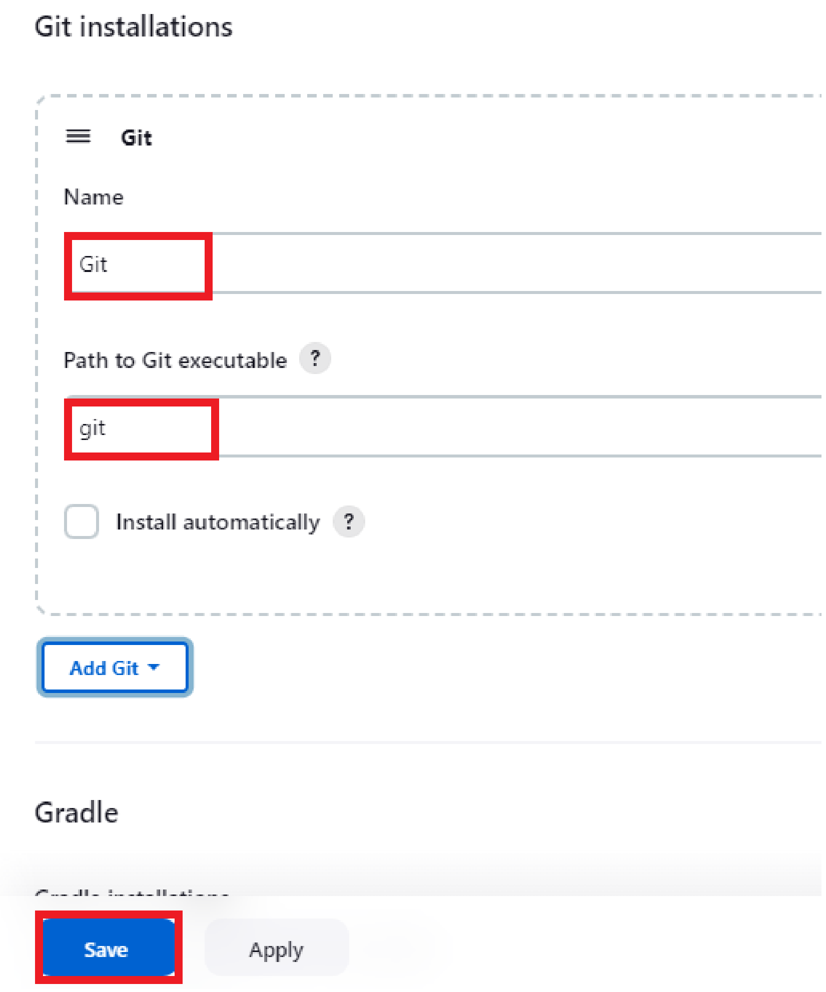
| Install | Name ↓ | Released |
|---|---|---|
| ✓ | GitHub Integration 0.5.0<br>emailext Build Triggers<br>GitHub Integration Plugin for Jenkins | 2 mo 28 days ago |

Update information obtained: 46 min ago

Now let’s configure Git on Jenkins. Go to **Manage Jenkins, and** click on **Global Tool Configuration.**


Dashboard > Manage Jenkins
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Manage Jenkins
System Configuration
Configure System
Configure global settings and paths.
Global Tool Configuration
Configure tools, their locations and automatic installers.
Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Under **Git installations**, provide the name Git, and under **Path,** we can either provide the complete path where our Git is installed on the Jenkins machine or just put any name, in my case I put Git to allow Jenkins to automatically search for Git. Then click on Save to complete the installation.



Git installations

Git

Name: Git

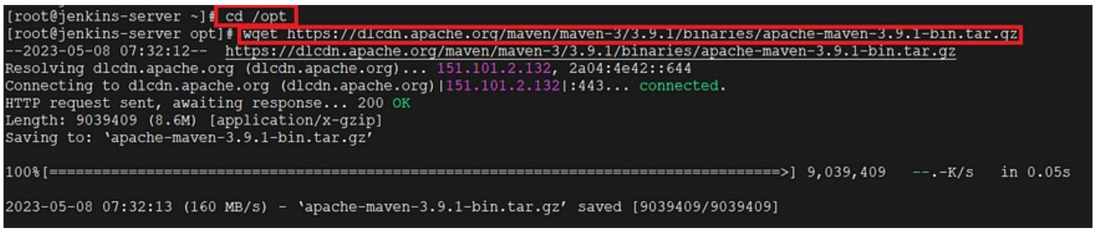
Path to Git executable: git

Install automatically: [ ]

Step 3: Integrate Maven with Jenkins

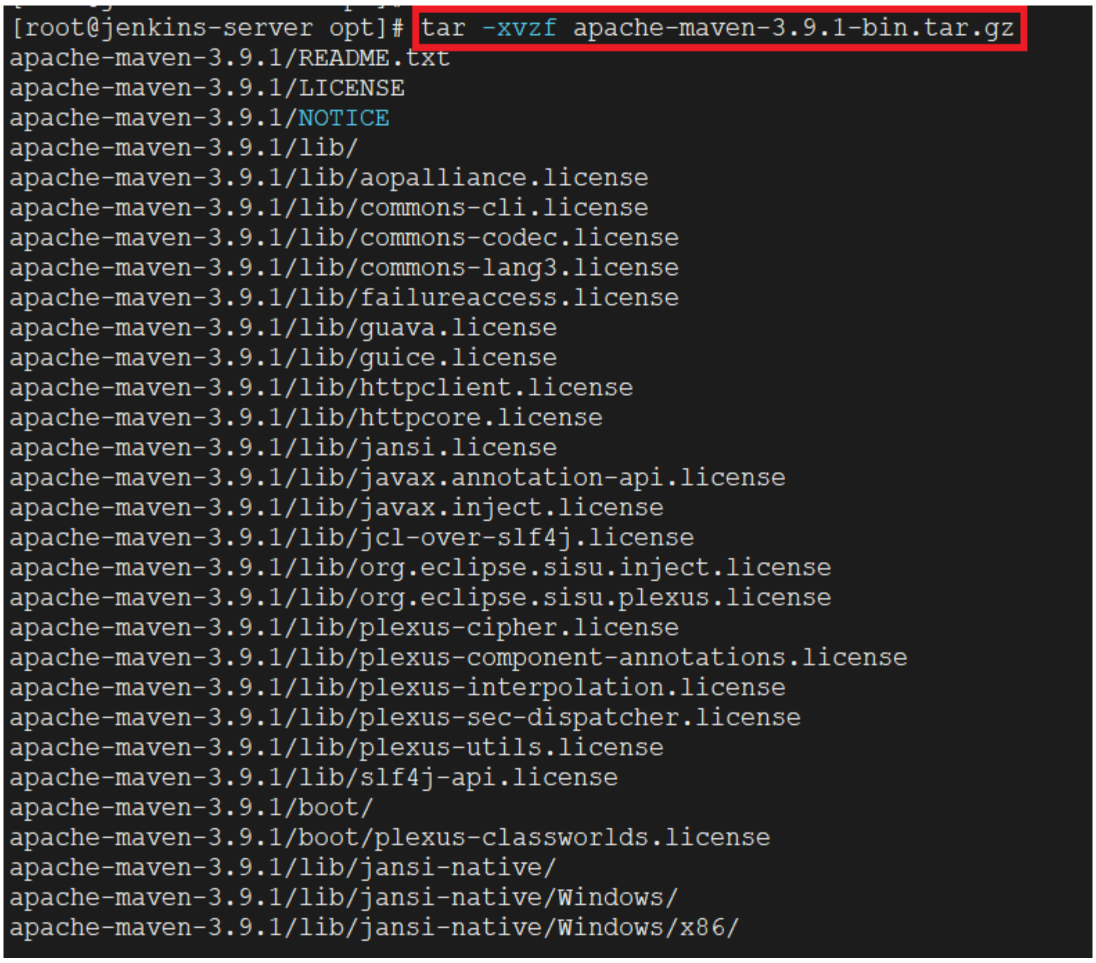
- •Setup Maven on Jenkins Server
- •Setup Environment Variables JAVA_HOME,M2,M2_HOME
- •Install Maven Plugin
- •Configure Maven and Java

To install Maven on our Jenkins Server we will switch to the /opt directory and download the Maven package:



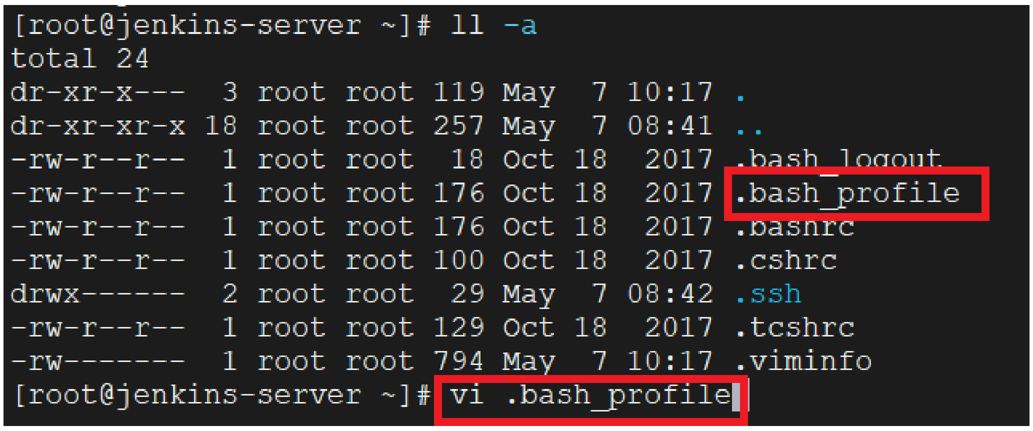
```
[root@jenkins-server ~]# cd /opt
[root@jenkins-server opt]# wget https://dlcdn.apache.org/maven/maven-3/3.9.1/binaries/apache-maven-3.9.1-bin.tar.gz
--2023-05-08 07:32:12--  https://dlcdn.apache.org/maven/maven-3/3.9.1/binaries/apache-maven-3.9.1-bin.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9039409 (8.6M) [application/x-gzip]
Saving to: ‘apache-maven-3.9.1-bin.tar.gz’

100%[==================================================================================>] 9,039,409   --.-K/s   in 0.05s

2023-05-08 07:32:13 (160 MB/s) - ‘apache-maven-3.9.1-bin.tar.gz’ saved [9039409/9039409]
```

Now we will extract the tar.gz file:



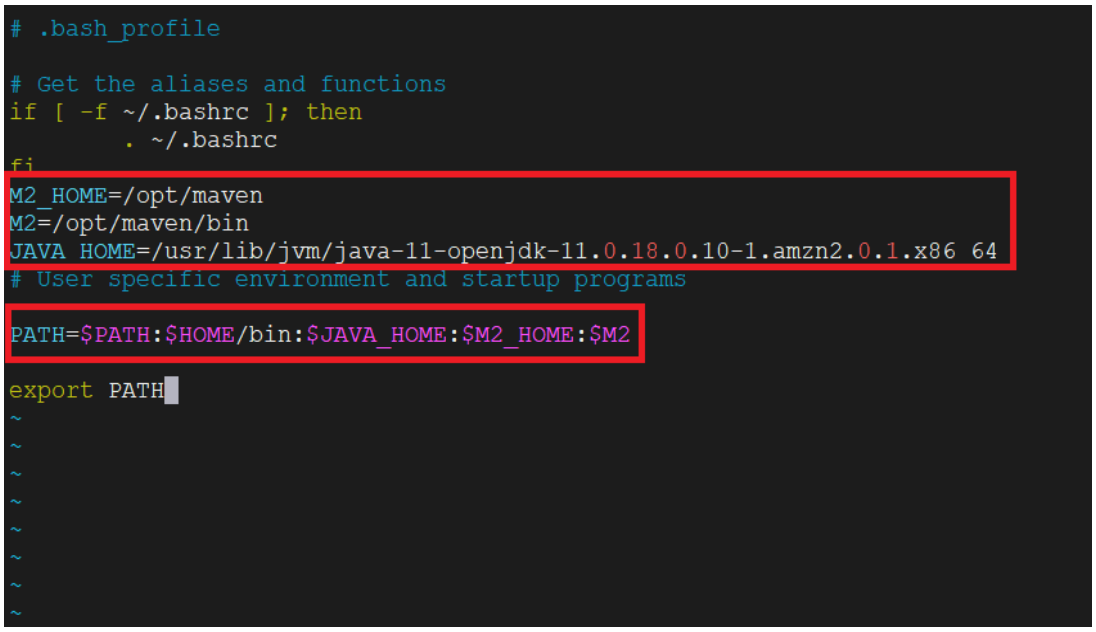
```
[root@jenkins-server opt]# tar -xvzf apache-maven-3.9.1-bin.tar.gz
apache-maven-3.9.1/README.txt
apache-maven-3.9.1/LICENSE
apache-maven-3.9.1/NOTICE
apache-maven-3.9.1/lib/
apache-maven-3.9.1/lib/aopalliance.license
apache-maven-3.9.1/lib/commons-cli.license
apache-maven-3.9.1/lib/commons-codec.license
apache-maven-3.9.1/lib/commons-lang3.license
apache-maven-3.9.1/lib/failureaccess.license
apache-maven-3.9.1/lib/guava.license
apache-maven-3.9.1/lib/guice.license
apache-maven-3.9.1/lib/httpclient.license
apache-maven-3.9.1/lib/httpcore.license
apache-maven-3.9.1/lib/jansi.license
apache-maven-3.9.1/lib/javax.annotation-api.license
apache-maven-3.9.1/lib/javax.inject.license
apache-maven-3.9.1/lib/jcl-over-slf4j.license
apache-maven-3.9.1/lib/org.eclipse.sisu.inject.license
apache-maven-3.9.1/lib/org.eclipse.sisu.plexus.license
apache-maven-3.9.1/lib/plexus-cipher.license
apache-maven-3.9.1/lib/plexus-component-annotations.license
apache-maven-3.9.1/lib/plexus-interpolation.license
apache-maven-3.9.1/lib/plexus-sec-dispatcher.license
apache-maven-3.9.1/lib/plexus-utils.license
apache-maven-3.9.1/lib/slf4j-api.license
apache-maven-3.9.1/boot/
apache-maven-3.9.1/boot/plexus-classworlds.license
apache-maven-3.9.1/lib/jansi-native/
apache-maven-3.9.1/lib/jansi-native/Windows/
apache-maven-3.9.1/lib/jansi-native/Windows/x86/
```

Now we will set up Environment Variables for our root user in **bash_profile** in order to access Maven from any location in our Server Go to the home directory of your Jenkins server and edit the bash_profile file as shown in the below steps:



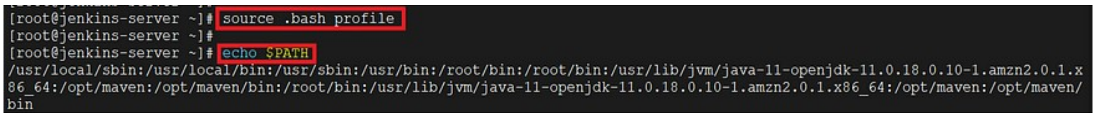
```
[root@jenkins-server ~]# ll -a
total 24
dr-xr-x---   3 root root 119 May  7 10:17 .
dr-xr-xr-x  18 root root 257 May  7 08:41 ..
-rw-r--r--   1 root root  18 Oct 18  2017 .bash_logout
-rw-r--r--   1 root root 176 Oct 18  2017 .bash_profile
-rw-r--r--   1 root root 176 Oct 18  2017 .bashrc
-rw-r--r--   1 root root 100 Oct 18  2017 .cshrc
drwx------   2 root root  29 May  7 08:42 .ssh
-rw-r--r--   1 root root 129 Oct 18  2017 .tcshrc
-rw-------   1 root root 794 May  7 10:17 .viminfo
[root@jenkins-server ~]# vi .bash_profile
```

In the **.bash_profile file**, we need to add Maven and Java paths and load these values.



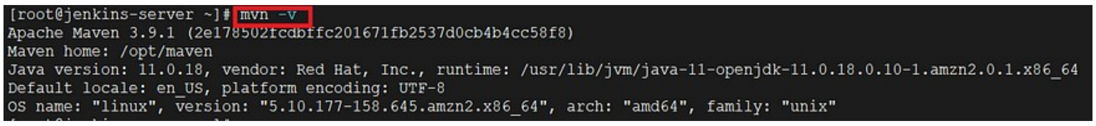
```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
M2_HOME=/opt/maven
M2=/opt/maven/bin
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64
# User specific environment and startup programs

PATH=$PATH:$HOME/bin:$JAVA_HOME:$M2_HOME:$M2

export PATH
~
~
~
~
~
~
~
~
```

To verify follow the below steps:



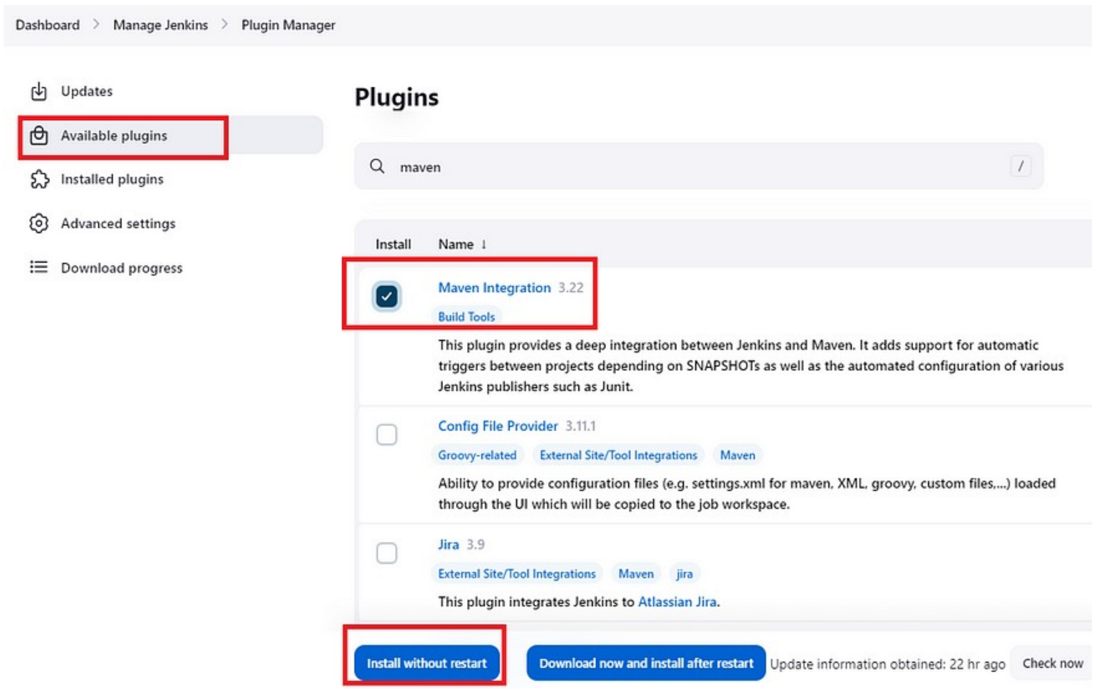
```
[root@jenkins-server ~]# source .bash_profile
[root@jenkins-server ~]#
[root@jenkins-server ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/root/bin:/usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x
86_64:/opt/maven:/opt/maven/bin:/root/bin:/usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64:/opt/maven:/opt/maven/
bin
```

With this setup, we can execute maven commands from anywhere on the server:



```
[root@jenkins-server ~]# mvn -v
Apache Maven 3.9.1 (2e178502fcdbffc201671fb2537d0cb4b4cc58f8)
Maven home: /opt/maven
Java version: 11.0.18, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.10.177-158.645.amzn2.x86_64", arch: "amd64", family: "unix"
```

Now we need to update the paths where Java and Maven have been installed in the Jenkins UI. We will first install the Maven Integration Plugin as shown below:


Dashboard > Manage Jenkins > Plugin Manager

Updates
Available plugins
Installed plugins
Advanced settings
Download progress

Plugins

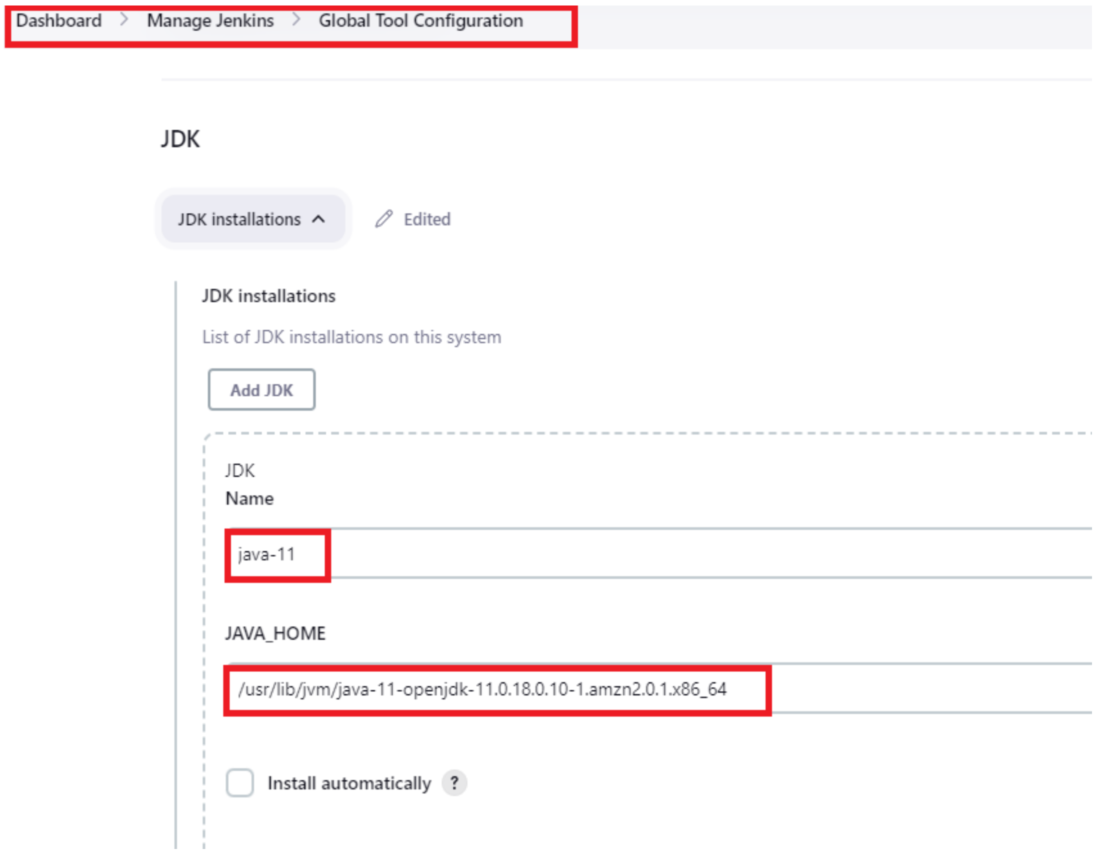
maven

| Install | Name ↓ |
|---|---|
| ✓ | Maven Integration 3.22<br>Build Tools<br>This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit. |
|  | Config File Provider 3.11.1<br>Groovy-related External Site/Tool Integrations Maven<br>Ability to provide configuration files (e.g. settings.xml for maven, XML, groovy, custom files,...) loaded through the UI which will be copied to the job workspace. |
|  | Jira 3.9<br>External Site/Tool Integrations Maven jira<br>This plugin integrates Jenkins to Atlassian Jira. |

Install without restart | Download now and install after restart | Update information obtained: 22 hr ago | Check now

After clicking on **Install without restart**, go again to manage Jenkins and select **Global Tool configuration** to set the paths for Java and Maven.

For JAVA:


Dashboard > Manage Jenkins > Global Tool Configuration

JDK

JDK installations ^ Edited

JDK installations

List of JDK installations on this system

Add JDK

JDK
Name

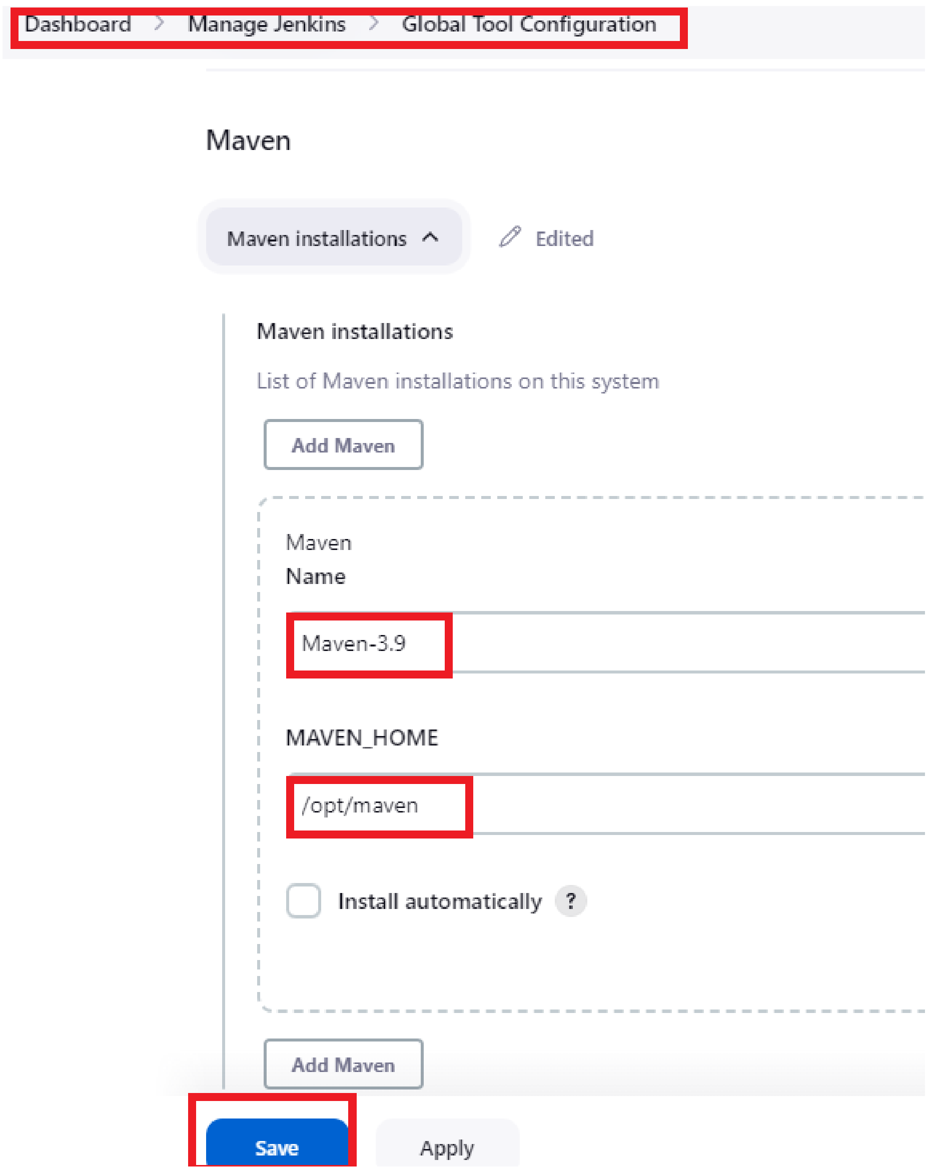
java-11

JAVA_HOME

/usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64

☐ Install automatically ?

For MAVEN:


Dashboard > Manage Jenkins > Global Tool Configuration

Maven

Maven installations — Edited

Maven installations

List of Maven installations on this system

Maven

Name: Maven-3.9

MAVEN_HOME: /opt/maven

Install automatically: ☐

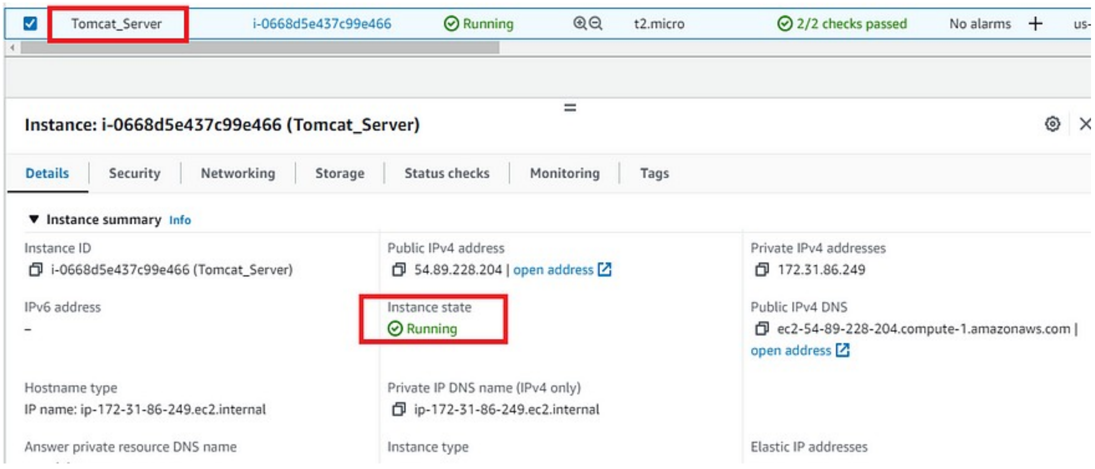
Click on save and hence we have successfully Integrated Java and Maven with Jenkins.

Step 4: Setup Tomcat Server

- Setup a Linux EC2 Instance
- Install Java
- Configure Tomcat
- Start Tomcat Server
- Access Web UI on port 8080

Let’s first create the Amazon Linux 2 EC2 Instance. Here we will skip the steps as we have already seen the creation of EC2 in the earlier steps.

Below is the screenshot of the EC2 Instance:



Tomcat_Server | i-0668d5e437c99e466 | Running | t2.micro | 2/2 checks passed | No alarms | us-

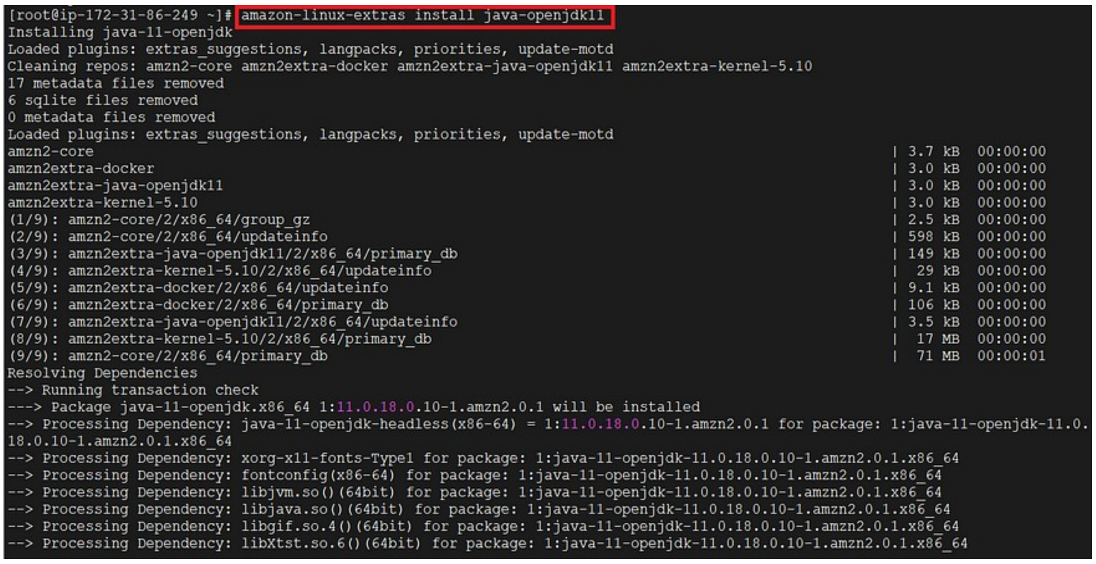
**Instance: i-0668d5e437c99e466 (Tomcat_Server)**

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

**Instance summary** Info

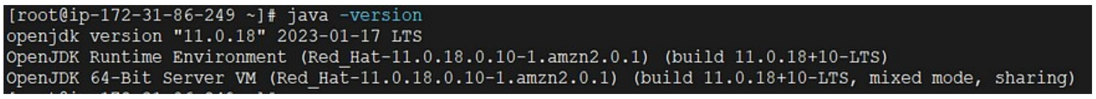
| | | |
|---|---|---|
| Instance ID<br>i-0668d5e437c99e466 (Tomcat_Server) | Public IPv4 address<br>54.89.228.204 \| open address | Private IPv4 addresses<br>172.31.86.249 |
| IPv6 address<br>– | Instance state<br>Running | Public IPv4 DNS<br>ec2-54-89-228-204.compute-1.amazonaws.com \| open address |
| Hostname type<br>IP name: ip-172-31-86-249.ec2.internal | Private IP DNS name (IPv4 only)<br>ip-172-31-86-249.ec2.internal | |
| Answer private resource DNS name | Instance type | Elastic IP addresses |

Let’s first install Java on the Tomcat Server.



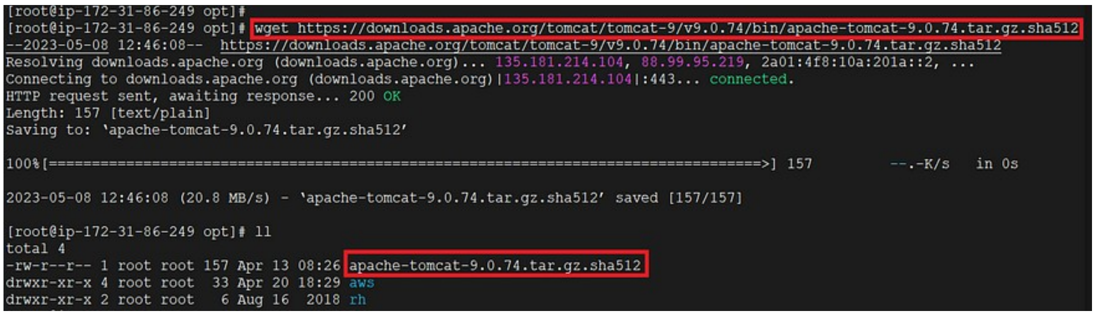
```
[root@ip-172-31-86-249 ~]# amazon-linux-extras install java-openjdk11
Installing java-11-openjdk
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-java-openjdk11 amzn2extra-kernel-5.10
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                                                 | 3.7 kB  00:00:00
amzn2extra-docker                                                                          | 3.0 kB  00:00:00
amzn2extra-java-openjdk11                                                                  | 3.0 kB  00:00:00
amzn2extra-kernel-5.10                                                                     | 3.0 kB  00:00:00
(1/9): amzn2-core/2/x86_64/group_gz                                                        | 2.5 kB  00:00:00
(2/9): amzn2-core/2/x86_64/updateinfo                                                      | 598 kB  00:00:00
(3/9): amzn2extra-java-openjdk11/2/x86_64/primary_db                                       | 149 kB  00:00:00
(4/9): amzn2extra-kernel-5.10/2/x86_64/updateinfo                                          |  29 kB  00:00:00
(5/9): amzn2extra-docker/2/x86_64/updateinfo                                               | 9.1 kB  00:00:00
(6/9): amzn2extra-docker/2/x86_64/primary_db                                               | 106 kB  00:00:00
(7/9): amzn2extra-java-openjdk11/2/x86_64/updateinfo                                       | 3.5 kB  00:00:00
(8/9): amzn2extra-kernel-5.10/2/x86_64/primary_db                                          |  17 MB  00:00:00
(9/9): amzn2-core/2/x86_64/primary_db                                                      |  71 MB  00:00:01
Resolving Dependencies
--> Running transaction check
---> Package java-11-openjdk.x86_64 1:11.0.18.0.10-1.amzn2.0.1 will be installed
--> Processing Dependency: java-11-openjdk-headless(x86-64) = 1:11.0.18.0.10-1.amzn2.0.1 for package: 1:java-11-openjdk-11.0.
18.0.10-1.amzn2.0.1.x86_64
--> Processing Dependency: xorg-x11-fonts-Type1 for package: 1:java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64
--> Processing Dependency: fontconfig(x86-64) for package: 1:java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64
--> Processing Dependency: libjvm.so()(64bit) for package: 1:java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64
--> Processing Dependency: libjava.so()(64bit) for package: 1:java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64
--> Processing Dependency: libgif.so.4()(64bit) for package: 1:java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64
--> Processing Dependency: libXtst.so.6()(64bit) for package: 1:java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64
```

Verify the version of Java:



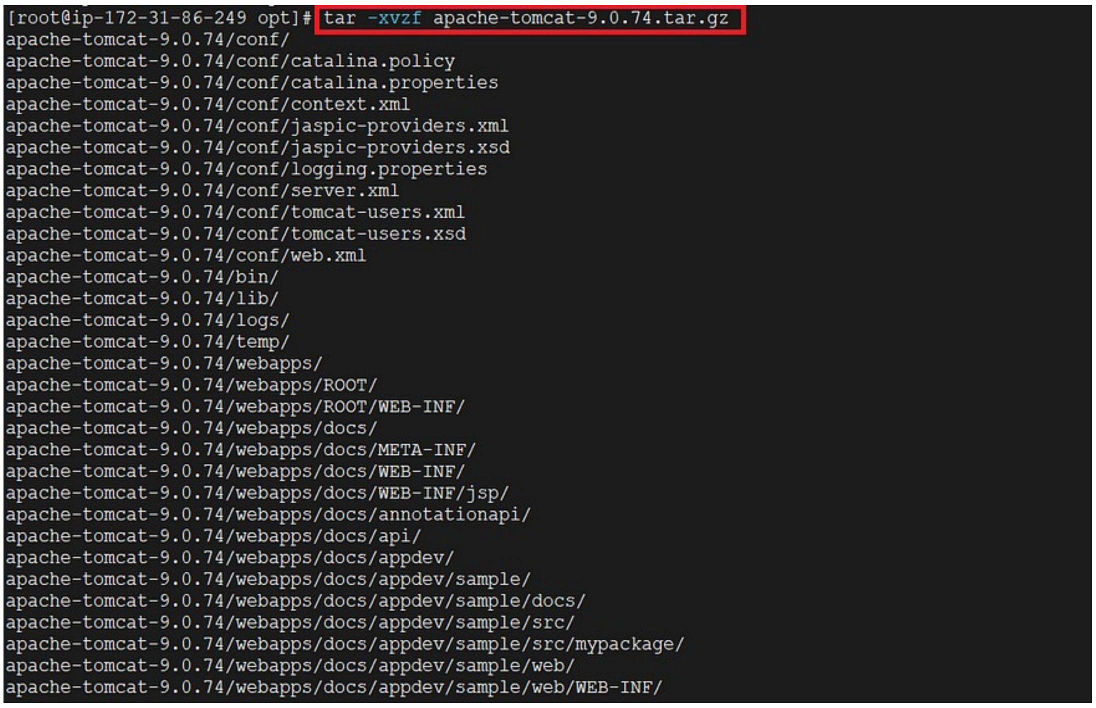
```
[root@ip-172-31-86-249 ~]# java -version
openjdk version "11.0.18" 2023-01-17 LTS
OpenJDK Runtime Environment (Red_Hat-11.0.18.0.10-1.amzn2.0.1) (build 11.0.18+10-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.18.0.10-1.amzn2.0.1) (build 11.0.18+10-LTS, mixed mode, sharing)
```

Now let’s first download the Tomcat Server and then install it in the /opt directory:



```
[root@ip-172-31-86-249 opt]#
[root@ip-172-31-86-249 opt]# wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.74/bin/apache-tomcat-9.0.74.tar.gz.sha512
--2023-05-08 12:46:08--  https://downloads.apache.org/tomcat/tomcat-9/v9.0.74/bin/apache-tomcat-9.0.74.tar.gz.sha512
Resolving downloads.apache.org (downloads.apache.org)... 135.181.214.104, 88.99.95.219, 2a01:4f8:10a:201a::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135.181.214.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 157 [text/plain]
Saving to: ‘apache-tomcat-9.0.74.tar.gz.sha512’

100%[=================================================================================>] 157         --.-K/s   in 0s

2023-05-08 12:46:08 (20.8 MB/s) - ‘apache-tomcat-9.0.74.tar.gz.sha512’ saved [157/157]

[root@ip-172-31-86-249 opt]# ll
total 4
-rw-r--r-- 1 root root 157 Apr 13 08:26 apache-tomcat-9.0.74.tar.gz.sha512
drwxr-xr-x 4 root root  33 Apr 20 18:29 aws
drwxr-xr-x 2 root root   6 Aug 16  2018 rh
```

Now extract the file as:

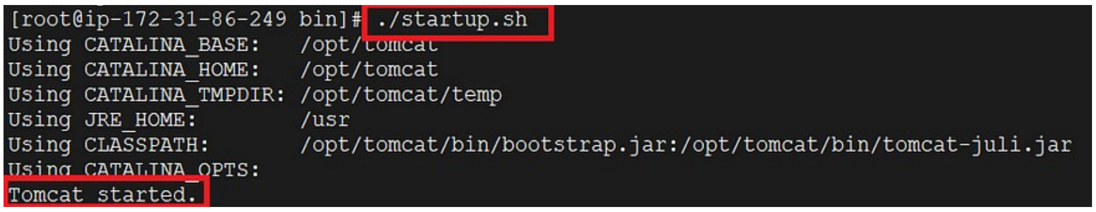


```
[root@ip-172-31-86-249 opt]# tar -xvzf apache-tomcat-9.0.74.tar.gz
apache-tomcat-9.0.74/conf/
apache-tomcat-9.0.74/conf/catalina.policy
apache-tomcat-9.0.74/conf/catalina.properties
apache-tomcat-9.0.74/conf/context.xml
apache-tomcat-9.0.74/conf/jaspic-providers.xml
apache-tomcat-9.0.74/conf/jaspic-providers.xsd
apache-tomcat-9.0.74/conf/logging.properties
apache-tomcat-9.0.74/conf/server.xml
apache-tomcat-9.0.74/conf/tomcat-users.xml
apache-tomcat-9.0.74/conf/tomcat-users.xsd
apache-tomcat-9.0.74/conf/web.xml
apache-tomcat-9.0.74/bin/
apache-tomcat-9.0.74/lib/
apache-tomcat-9.0.74/logs/
apache-tomcat-9.0.74/temp/
apache-tomcat-9.0.74/webapps/
apache-tomcat-9.0.74/webapps/ROOT/
apache-tomcat-9.0.74/webapps/ROOT/WEB-INF/
apache-tomcat-9.0.74/webapps/docs/
apache-tomcat-9.0.74/webapps/docs/META-INF/
apache-tomcat-9.0.74/webapps/docs/WEB-INF/
apache-tomcat-9.0.74/webapps/docs/WEB-INF/jsp/
apache-tomcat-9.0.74/webapps/docs/annotationapi/
apache-tomcat-9.0.74/webapps/docs/api/
apache-tomcat-9.0.74/webapps/docs/appdev/
apache-tomcat-9.0.74/webapps/docs/appdev/sample/
apache-tomcat-9.0.74/webapps/docs/appdev/sample/docs/
apache-tomcat-9.0.74/webapps/docs/appdev/sample/src/
apache-tomcat-9.0.74/webapps/docs/appdev/sample/src/mypackage/
apache-tomcat-9.0.74/webapps/docs/appdev/sample/web/
apache-tomcat-9.0.74/webapps/docs/appdev/sample/web/WEB-INF/
```

After extracting, let’s rename the folder as tomcat to make things simpler.

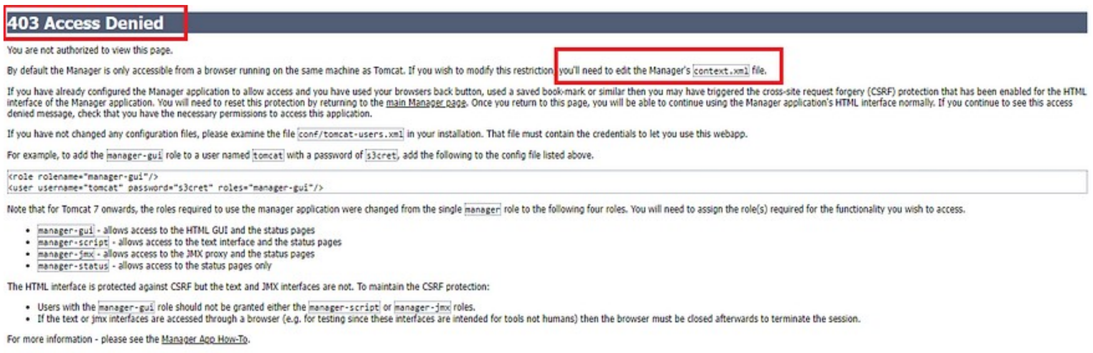
```
mv apache-tomcat-9.0.74 tomcat
```

Now move into the tomcat directory, then to /bin directory there we need to run the **startup.sh** script to run the tomcat services on our Server.



```
[root@ip-172-31-86-249 bin]# ./startup.sh
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
```

Now in order to make sure that tomcat server **Manager App** is accessible from anywhere we need to make some changes in a couple of files as initially after the installation the tomcat service Manager App service would be accessible only on the local host on which it is installed:



403 Access Denied

You are not authorized to view this page.

By default the Manager is only accessible from a browser running on the same machine as Tomcat. If you wish to modify this restriction you'll need to edit the Manager's `context.xml` file.

If you have already configured the Manager application to allow access and you have used your browsers back button, used a saved book-mark or similar then you may have triggered the cross-site request forgery (CSRF) protection that has been enabled for the HTML interface of the Manager application. You will need to reset this protection by returning to the main Manager page. Once you return to this page, you will be able to continue using the Manager application's HTML interface normally. If you continue to see this access denied message, check that you have the necessary permissions to access this application.

If you have not changed any configuration files, please examine the file `conf/tomcat-users.xml` in your installation. That file must contain the credentials to let you use this webapp.

For example, to add the `manager-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

```
<role rolename="manager-gui"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
```

Note that for Tomcat 7 onwards, the roles required to use the manager application were changed from the single `manager` role to the following four roles. You will need to assign the role(s) required for the functionality you wish to access.

- `manager-gui` - allows access to the HTML GUI and the status pages
- `manager-script` - allows access to the text interface and the status pages
- `manager-jmx` - allows access to the JMX proxy and the status pages
- `manager-status` - allows access to the status pages only

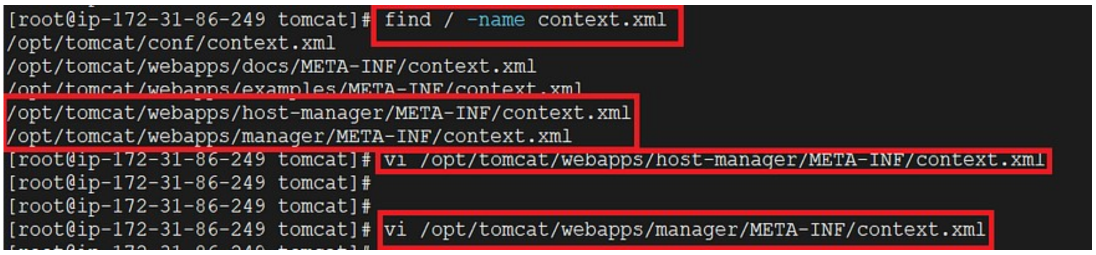
The HTML interface is protected against CSRF but the text and JMX interfaces are not. To maintain the CSRF protection:

- Users with the `manager-gui` role should not be granted either the `manager-script` or `manager-jmx` roles.
- If the text or jmx interfaces are accessed through a browser (e.g. for testing since these interfaces are intended for tools not humans) then the browser must be closed afterwards to terminate the session.

For more information - please see the Manager App How-To.

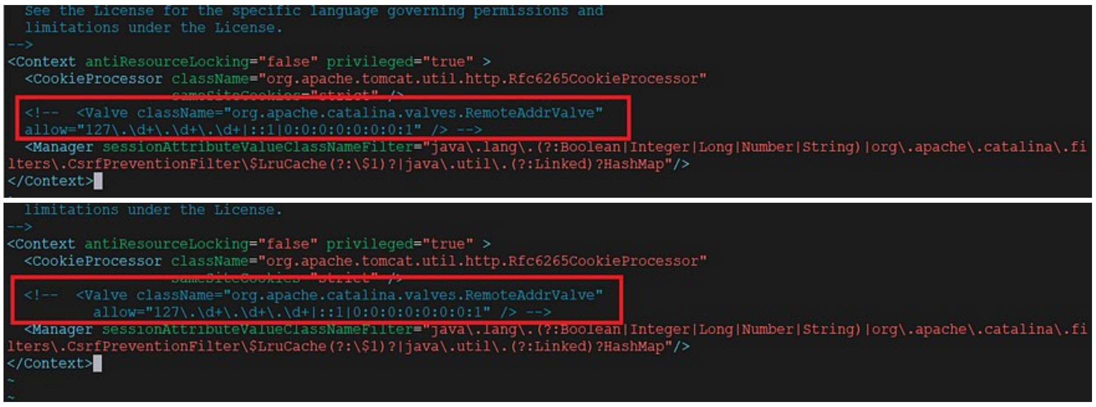
Now we will update the **context.xml** file to allow access to Tomcat Server from anywhere apart from the localhost.

First, we will search for the context.xml file in the tomcat directory which is present twice as shown below:



```
[root@ip-172-31-86-249 tomcat]# find / -name context.xml
/opt/tomcat/conf/context.xml
/opt/tomcat/webapps/docs/META-INF/context.xml
/opt/tomcat/webapps/examples/META-INF/context.xml
/opt/tomcat/webapps/host-manager/META-INF/context.xml
/opt/tomcat/webapps/manager/META-INF/context.xml
[root@ip-172-31-86-249 tomcat]# vi /opt/tomcat/webapps/host-manager/META-INF/context.xml
[root@ip-172-31-86-249 tomcat]#
[root@ip-172-31-86-249 tomcat]#
[root@ip-172-31-86-249 tomcat]# vi /opt/tomcat/webapps/manager/META-INF/context.xml
```

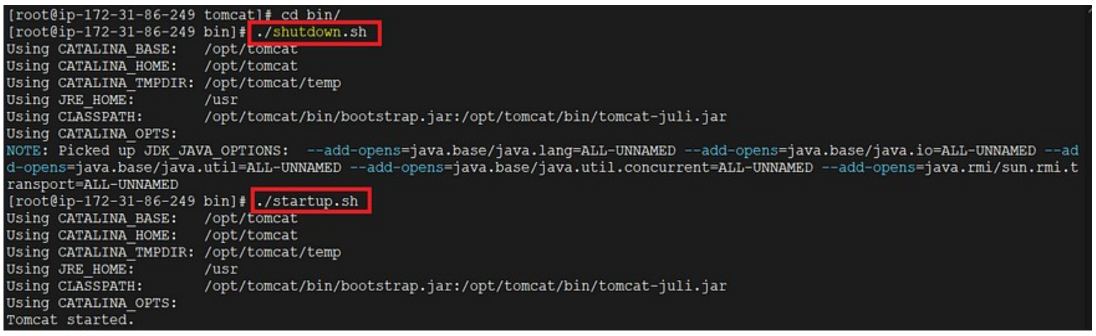
Then after opening the files, we need to comment out a line as shown in the below screenshot:



```
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
                   sameSiteCookies="strict" />
  <!--  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
  allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.catalina\.fi
lters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
```

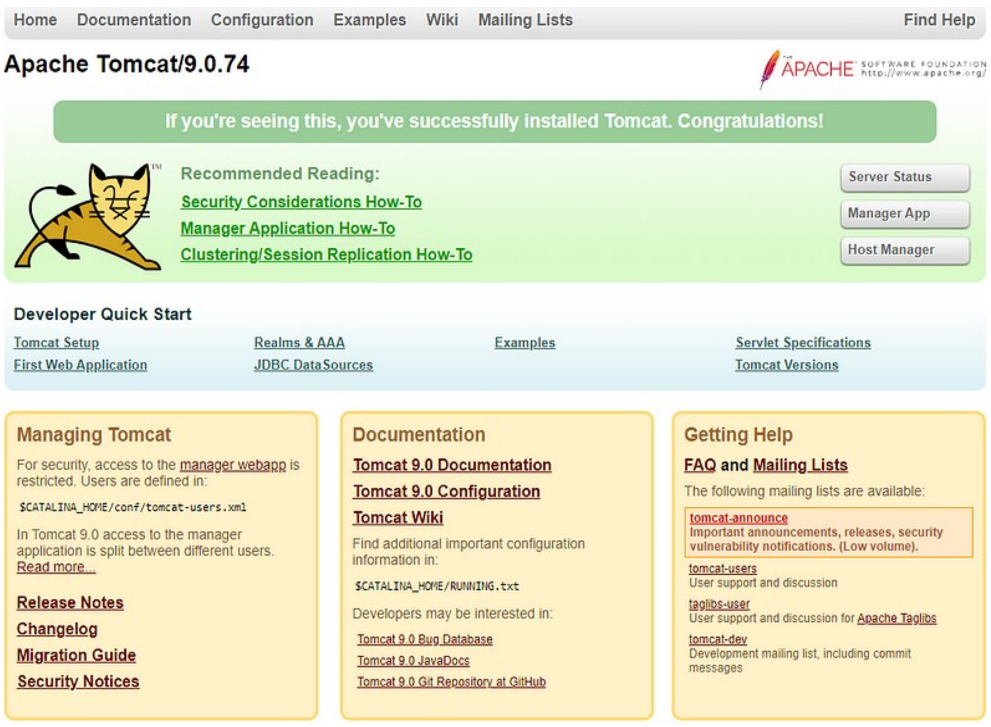
```
  limitations under the License.
-->
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
                   sameSiteCookies="strict" />
  <!--  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.catalina\.fi
lters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
~
~
```

After making the changes we need to restart the tomcat services:



```
[root@ip-172-31-86-249 tomcat]# cd bin/
[root@ip-172-31-86-249 bin]# ./shutdown.sh
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
NOTE: Picked up JDK_JAVA_OPTIONS:  --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --ad
d-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.t
ransport=ALL-UNNAMED
[root@ip-172-31-86-249 bin]# ./startup.sh
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
```

Now we can access the tomcat server from our browser:



Apache Tomcat/9.0.74

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To

Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 9.0 access to the manager application is split between different users. Read more...
Release Notes
Changelog
Migration Guide
Security Notices

Documentation
Tomcat 9.0 Documentation
Tomcat 9.0 Configuration
Tomcat Wiki
Find additional important configuration information in:
$CATALINA_HOME/RUNNING.txt
Developers may be interested in:
Tomcat 9.0 Bug Database
Tomcat 9.0 JavaDocs
Tomcat 9.0 Git Repository at GitHub

Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).
tomcat-users
User support and discussion
taglibs-user
User support and discussion for Apache Taglibs
tomcat-dev
Development mailing list, including commit messages

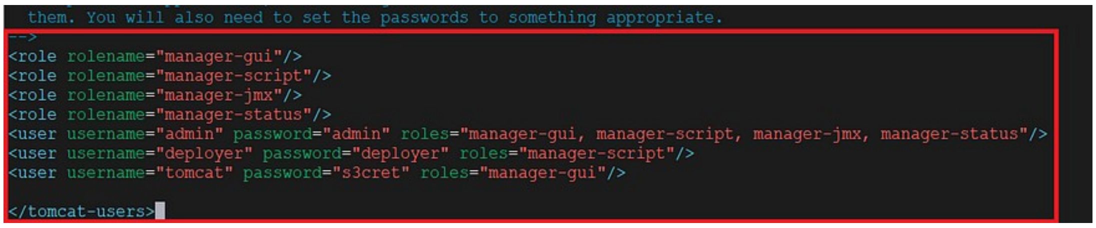
In order to access the **Manager App** found on the home page of the tomcat server we need to provide credentials. For this, we need to add some Users in the **conf/tomcat-users.xml** file.

Update the user's information in the tomcat-users.xml file goto tomcat home directory and Add the below users to conf/tomcat-users.xml file:

```
  <role rolename="manager-gui"/>


  <role rolename="manager-script"/>


  <role rolename="manager-jmx"/>


  <role rolename="manager-status"/>


  <user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx,
manager-status"/>
```

```
<user username="deployer" password="deployer" roles="manager-script"/>


<user username="tomcat" password="s3cret" roles="manager-gui"/>
```

Output:



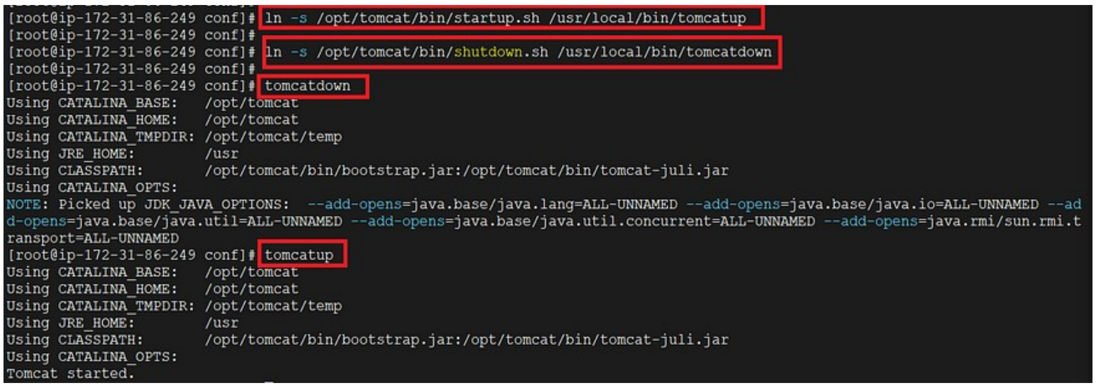
```
  them. You will also need to set the passwords to something appropriate.
-->
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx, manager-status"/>
<user username="deployer" password="deployer" roles="manager-script"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>

</tomcat-users>
```

Now we again to restart the services, to make things easier let’s create link files for tomcat startup.sh and shutdown.sh

For that add the below lines and restart the tomcat service:

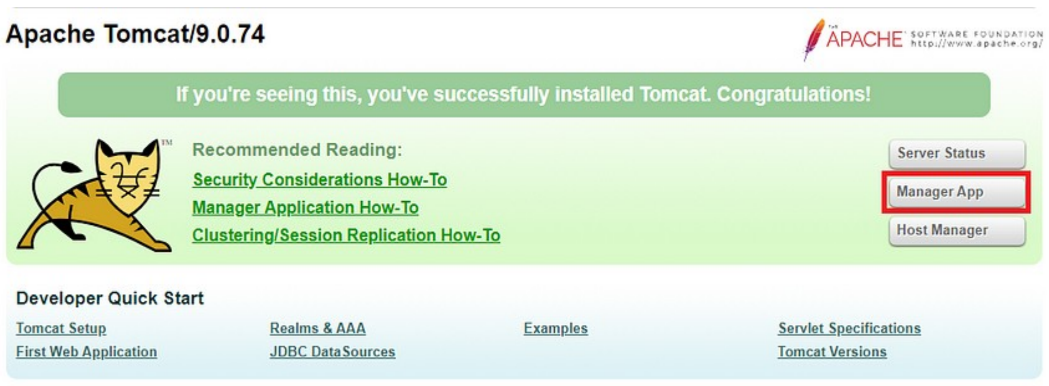
```
ln -s /opt/tomcat/bin/startup.sh /usr/local/bin/tomcatup


ln -s /opt/tomcat/bin/shutdown.sh /usr/local/bin/tomcatdown
```

Output:



```
[root@ip-172-31-86-249 conf]# ln -s /opt/tomcat/bin/startup.sh /usr/local/bin/tomcatup
[root@ip-172-31-86-249 conf]#
[root@ip-172-31-86-249 conf]# ln -s /opt/tomcat/bin/shutdown.sh /usr/local/bin/tomcatdown
[root@ip-172-31-86-249 conf]#
[root@ip-172-31-86-249 conf]# tomcatdown
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
NOTE: Picked up JDK_JAVA_OPTIONS:  --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --ad
d-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.t
ransport=ALL-UNNAMED
[root@ip-172-31-86-249 conf]# tomcatup
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
```

Now we can access the Manager App on the tomcat server by providing a username: tomcat and password as s3cret


Apache Tomcat/9.0.74

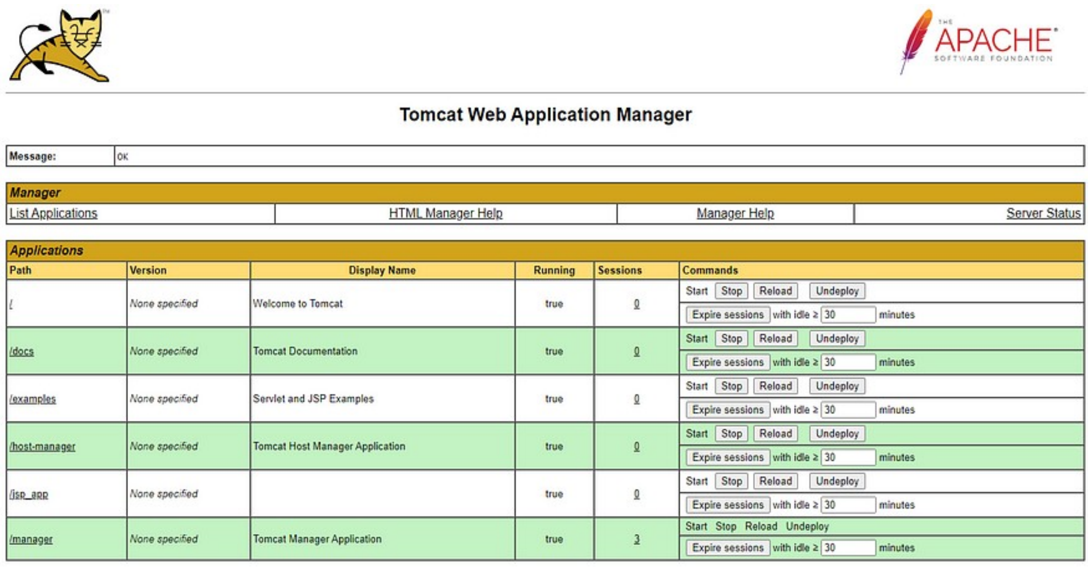
If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To

Developer Quick Start

Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

After clicking on the Manager App, provide the credentials and we would be able to see the page below:


Tomcat Web Application Manager

| | |
|---|---|
| Message: | OK |

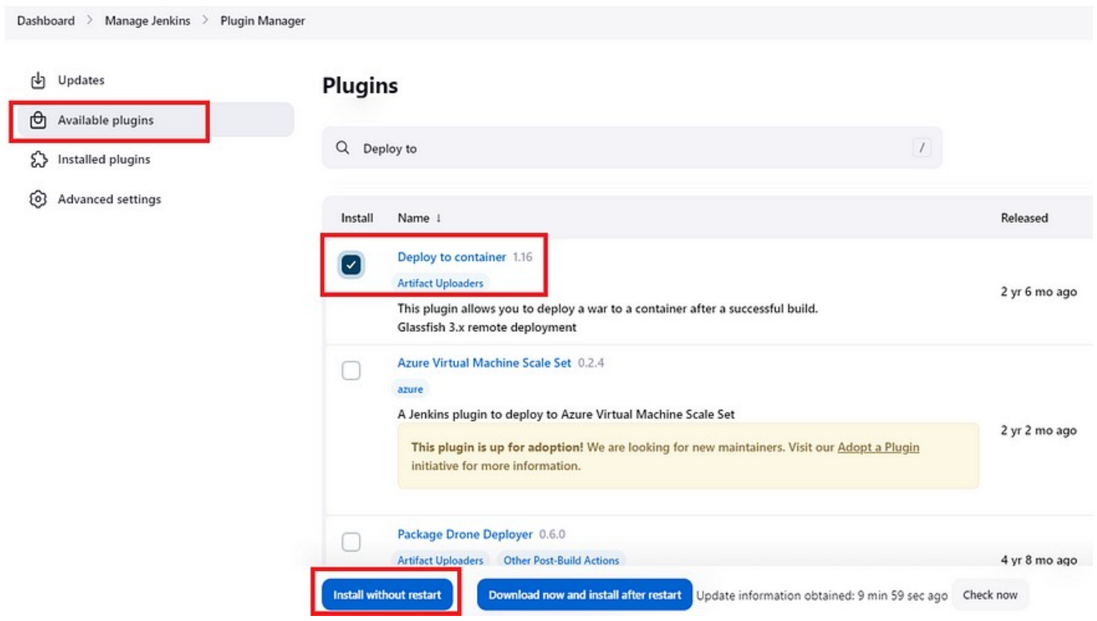
| Manager | | | |
|---|---|---|---|
| List Applications | HTML Manager Help | Manager Help | Server Status |

| Path | Version | Display Name | Running | Sessions | Commands |
|---|---|---|---|---|---|
| / | None specified | Welcome to Tomcat | true | 0 | Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes |
| /docs | None specified | Tomcat Documentation | true | 0 | Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes |
| /examples | None specified | Servlet and JSP Examples | true | 0 | Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes |
| /host-manager | None specified | Tomcat Host Manager Application | true | 0 | Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes |
| /jsp_app | None specified | | true | 0 | Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes |
| /manager | None specified | Tomcat Manager Application | true | 3 | Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes |

Step 5: Integrate Tomcat with Jenkins

- •Install “Deploy to container” plugin on Jenkins UI
- •Configure the tomcat server with credentials

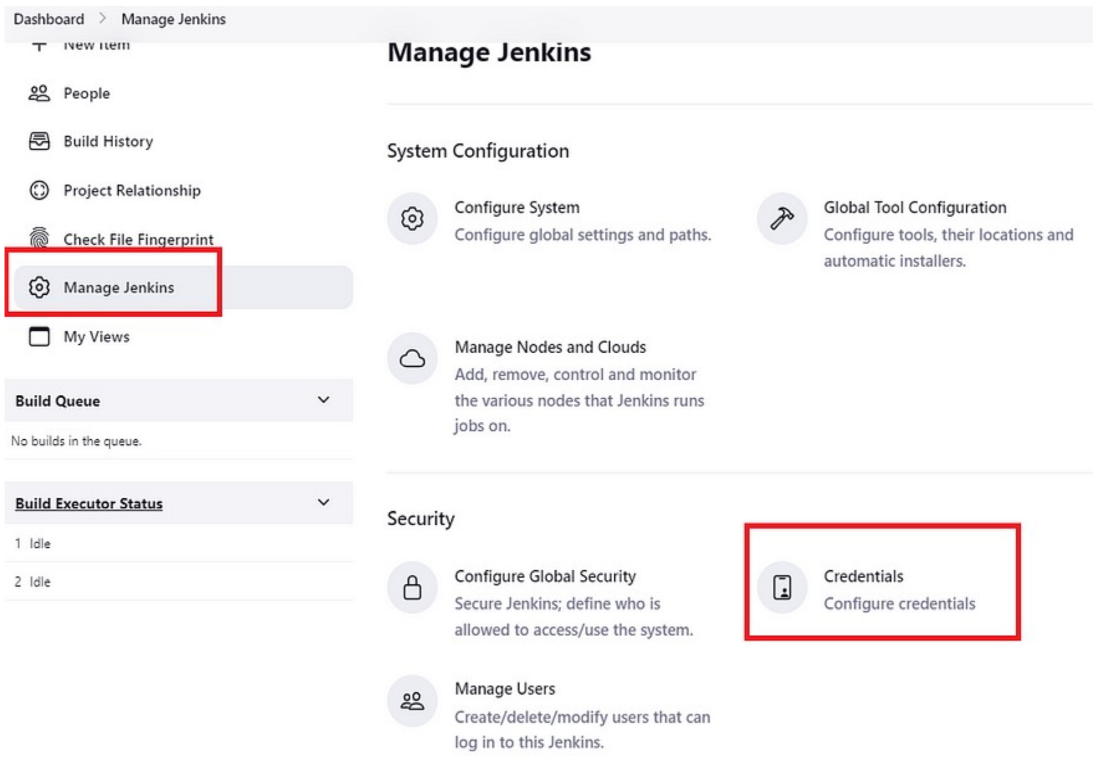
Now let’s first install the Deploy to Container plugin. Go to manage Jenkins > Manage Plugins:



| Install | Name ↓ | Released |
|---|---|---|
| ✓ | Deploy to container 1.16<br>Artifact Uploaders<br>This plugin allows you to deploy a war to a container after a successful build.<br>Glassfish 3.x remote deployment | 2 yr 6 mo ago |
|  | Azure Virtual Machine Scale Set 0.2.4<br>azure<br>A Jenkins plugin to deploy to Azure Virtual Machine Scale Set<br>**This plugin is up for adoption!** We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information. | 2 yr 2 mo ago |
|  | Package Drone Deployer 0.6.0<br>Artifact Uploaders  Other Post-Build Actions | 4 yr 8 mo ago |

Update information obtained: 9 min 59 sec ago

Now let’s configure Tomcat with credentials. For that go to Manage Jenkins and under security select Credentials



No builds in the queue.

1 Idle

2 Idle

Configure System
Configure global settings and paths.

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

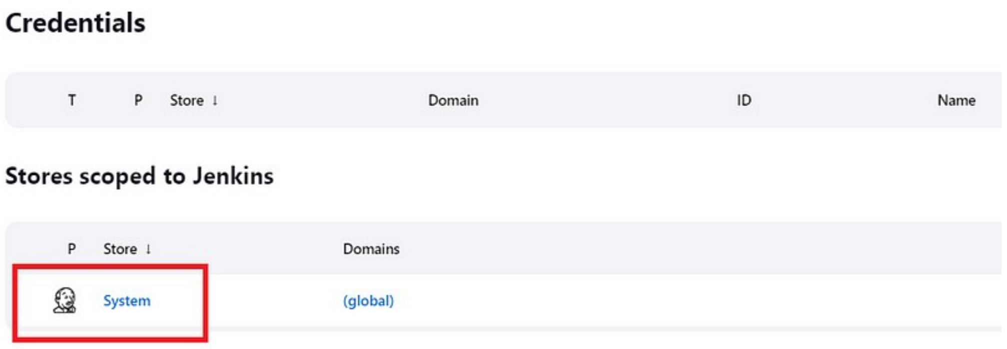
Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Credentials
Configure credentials

Manage Users
Create/delete/modify users that can log in to this Jenkins.

Click on System:

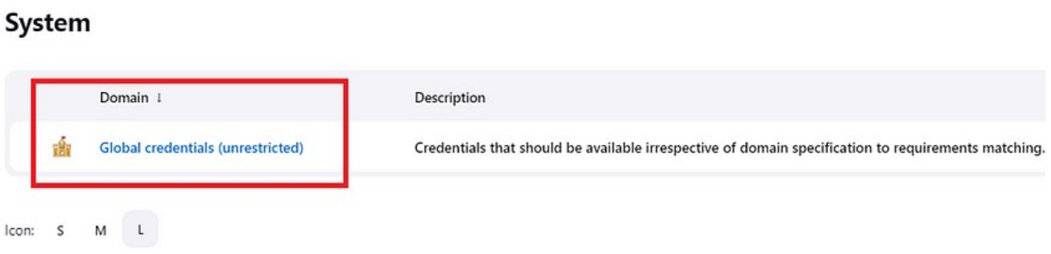

Credentials

| T | P | Store ↓ | Domain | ID | Name |
|---|---|---|---|---|---|

Stores scoped to Jenkins

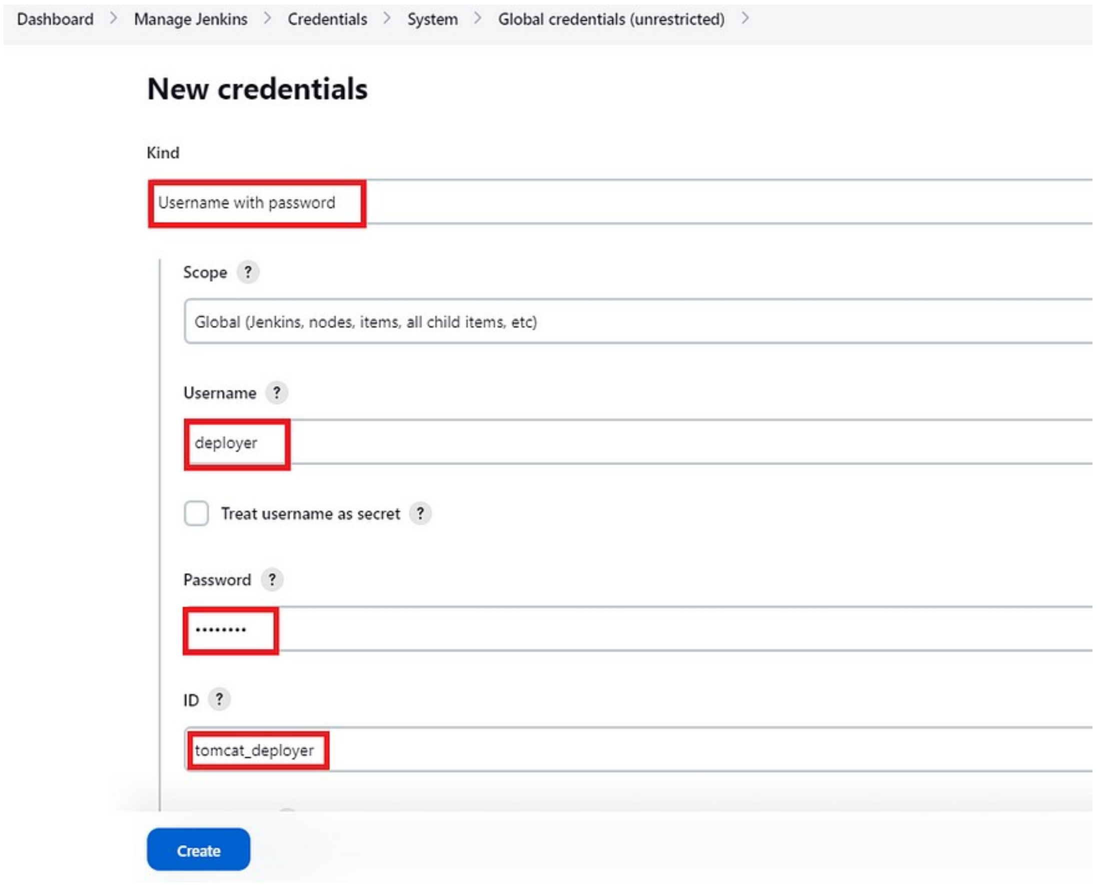
| P | Store ↓ | Domains |
|---|---|---|
| | System | (global) |

Then select Global credentials (unrestricted):


System

| Domain ↓ | Description |
|---|---|
| Global credentials (unrestricted) | Credentials that should be available irrespective of domain specification to requirements matching. |

On the next screen, Provide the required information, for example under **kind** select Username and password, etc and select **Create** to proceed:


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: deployer

Treat username as secret: ☐

Password: ••••••••

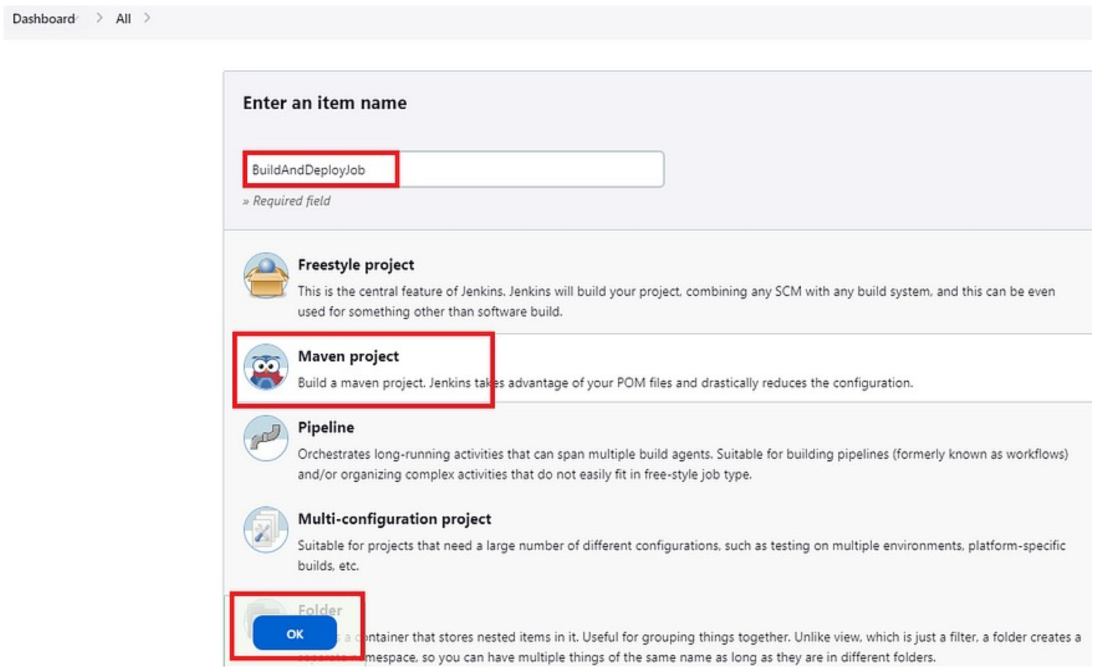
ID: tomcat_deployer

Hence this completes the successful integration of Tomcat with Jenkins.

Step 6: Deploy a Java application on a remote Tomcat Server

In this step we would first create a new job in Jenkins, provide the URL of our GitHub repository from where the source code would be pulled, then Maven will be used to build the project and finally, we will deploy the project on the tomcat server all using the CI server named Jenkins.

Here let’s create a new job from scratch:



Dashboard > All >

Enter an item name

BuildAndDeployJob

» 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.

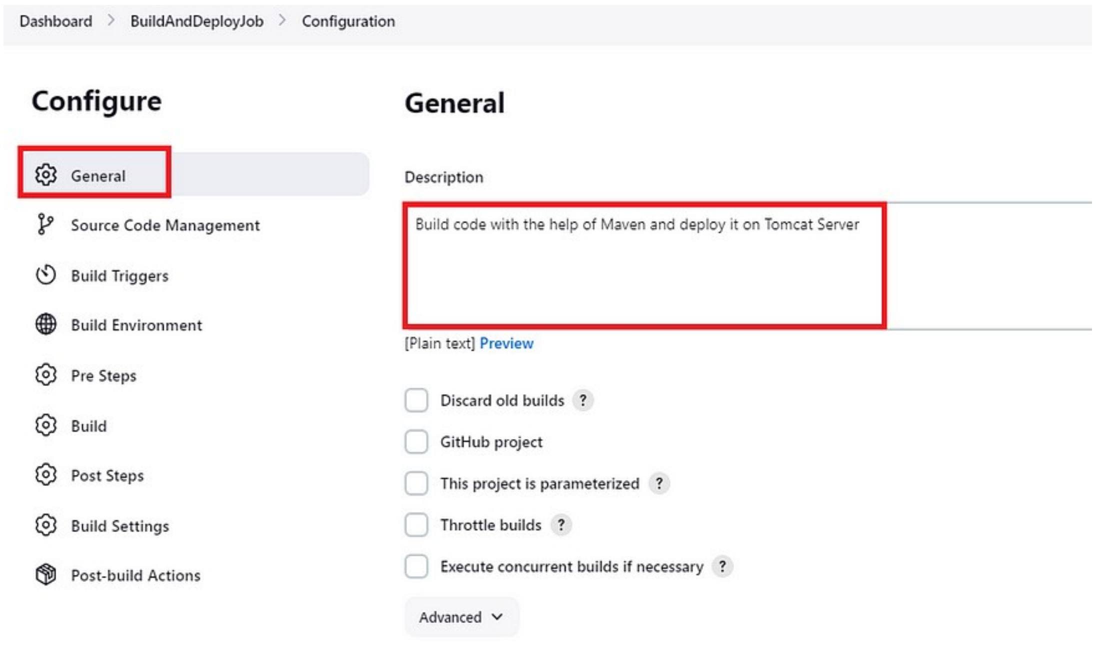
**Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**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.

**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**
...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.

Now let’s configure our new job. Under General provide the description of your choice:



Dashboard > BuildAndDeployJob > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Pre Steps
- Build
- Post Steps
- Build Settings
- Post-build Actions

General

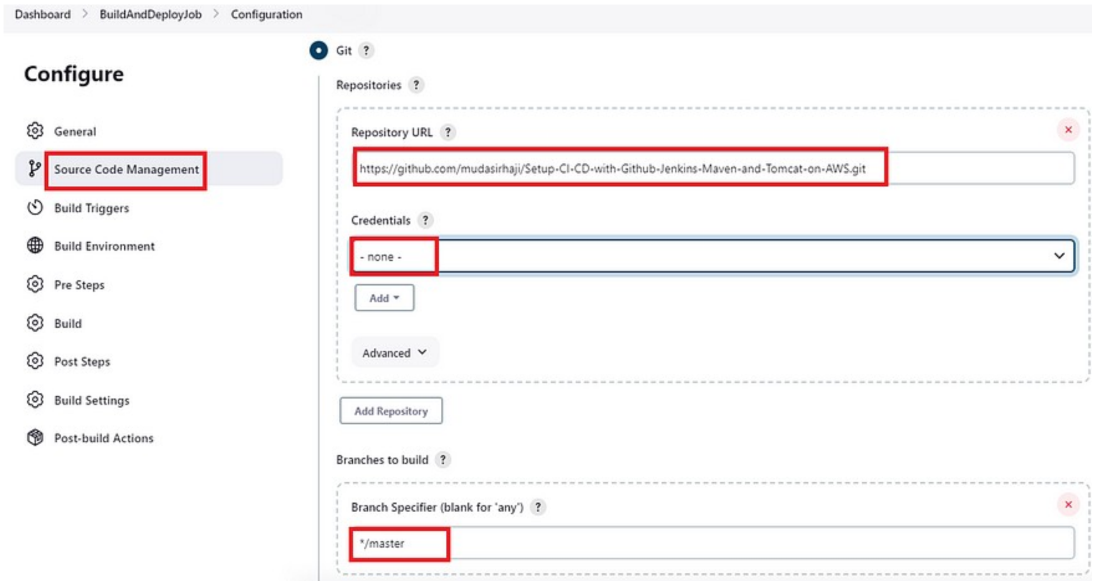
Description

Build code with the help of Maven and deploy it on Tomcat Server

[Plain text] Preview

- [ ] Discard old builds ?
- [ ] GitHub project
- [ ] This project is parameterized ?
- [ ] Throttle builds ?
- [ ] Execute concurrent builds if necessary ?

Under **Source Code Management**, paste the URL of your GitHub code repository, you can leave the credentials as blank as this is a public repository and mention the branch of your repo, in my case it's Master.

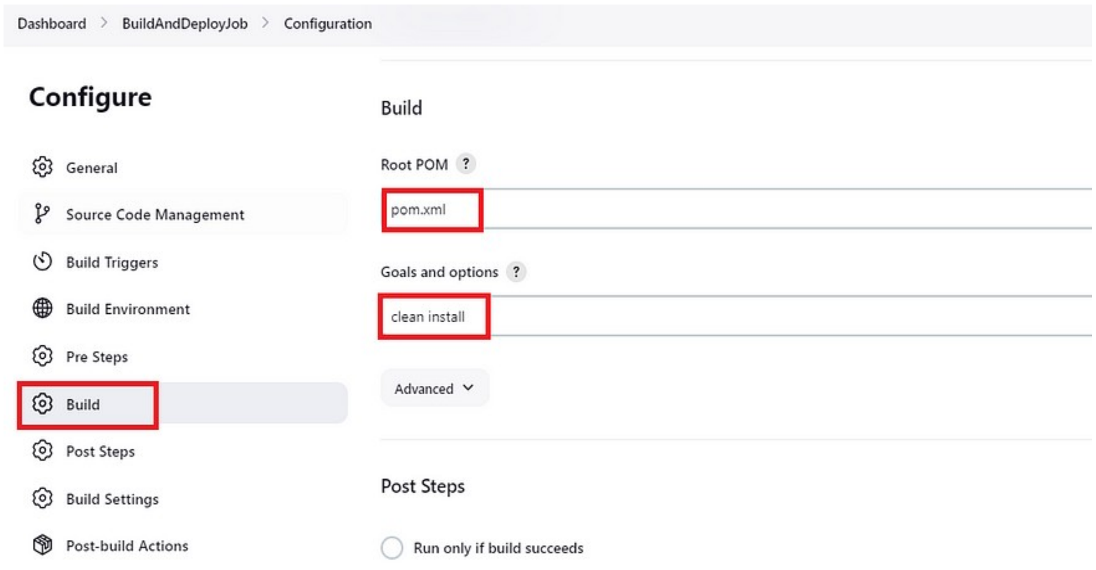


Repository URL: https://github.com/mudasirhaji/Setup-CI-CD-with-Github-Jenkins-Maven-and-Tomcat-on-AWS.git

Credentials: - none -

Branch Specifier (blank for 'any'): */master

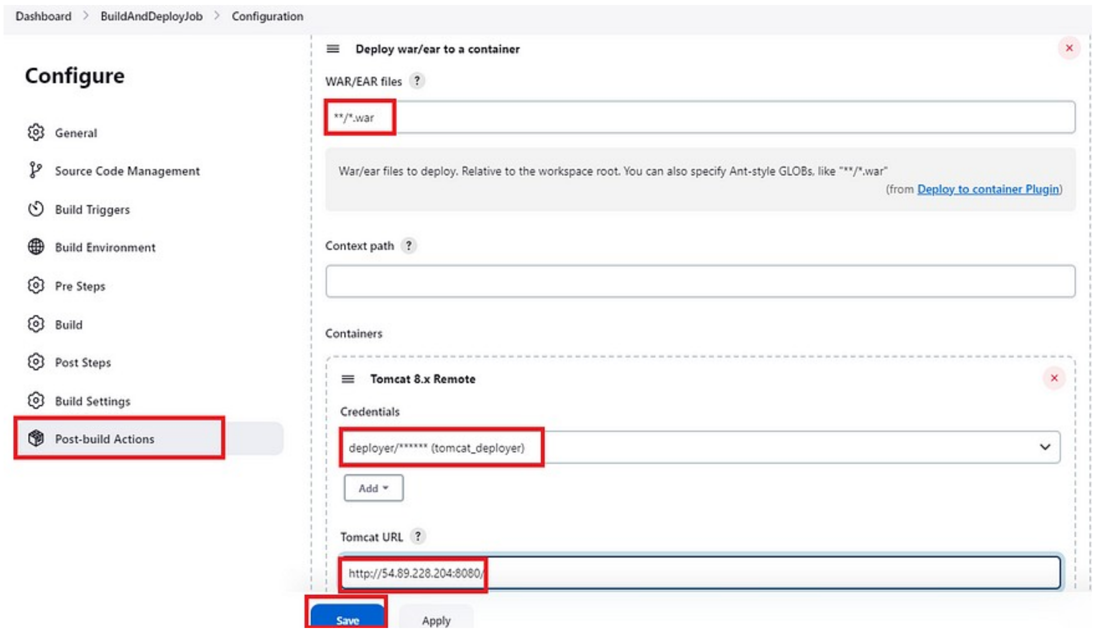
Under **Build** Settings**,** under **Root POM** mention the pom.xml which should be present in our Git code repository. Under **Goals and Options**. provide **clean install** package name which will install the necessary packages and install them in our local repository.



Root POM: pom.xml

Goals and options: clean install

Now we need to deploy our code, so under Build Settings, select **Deploy war/ear to a container,** and then under **Post-build Actions** provide the necessary details like a path to the war file, tomcat server credentials, and URL, as shown in the screenshot:



Deploy war/ear to a container

WAR/EAR files: **/*.war

War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like "**/*.war"
(from Deploy to container Plugin)

Context path

Containers

Tomcat 8.x Remote

Credentials: deployer/****** (tomcat_deployer)

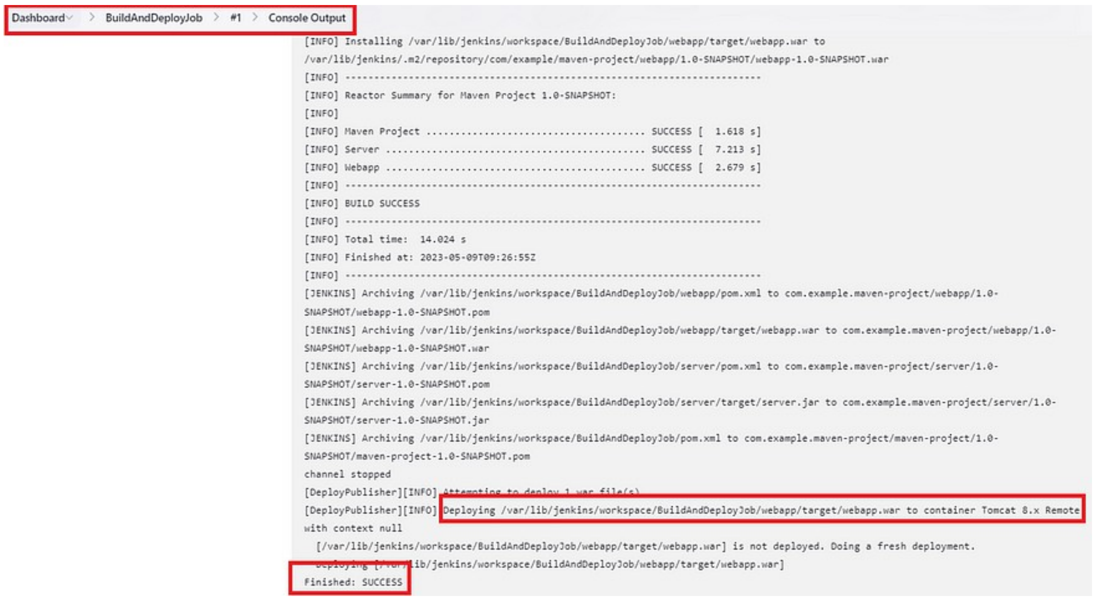
Tomcat URL: http://54.89.228.204:8080/

Finally, click on **Apply** and **Save**.

Now let’s finally Build our code which would eventually copy the Artifacts to the tomcat server.

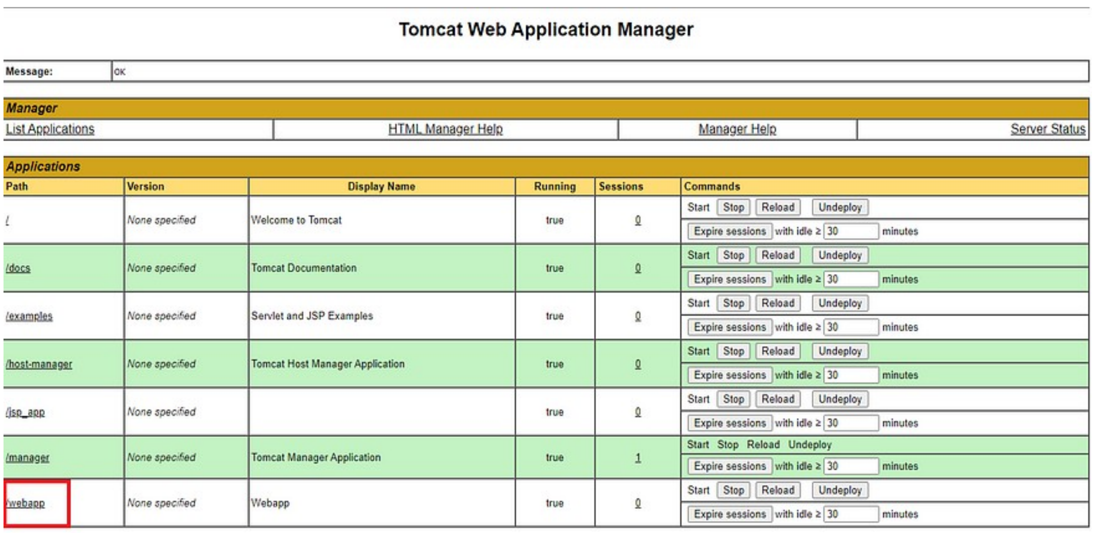
Click on Build now on the Jenkins UI to trigger the build:

After clicking on Build Now if we check the console output we can notice that the Build is successful and Jenkins was successfully able to deploy the WAR file onto the tomcat server as shown below:



```
[INFO] Installing /var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war to
/var/lib/jenkins/.m2/repository/com/example/maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.war
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for Maven Project 1.0-SNAPSHOT:
[INFO]
[INFO] Maven Project ...................................... SUCCESS [  1.618 s]
[INFO] Server ............................................. SUCCESS [  7.213 s]
[INFO] Webapp ............................................. SUCCESS [  2.679 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  14.024 s
[INFO] Finished at: 2023-05-09T09:26:55Z
[INFO] ------------------------------------------------------------------------
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/webapp/pom.xml to com.example.maven-project/webapp/1.0-
SNAPSHOT/webapp-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war to com.example.maven-project/webapp/1.0-
SNAPSHOT/webapp-1.0-SNAPSHOT.war
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/server/pom.xml to com.example.maven-project/server/1.0-
SNAPSHOT/server-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/server/target/server.jar to com.example.maven-project/server/1.0-
SNAPSHOT/server-1.0-SNAPSHOT.jar
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/pom.xml to com.example.maven-project/maven-project/1.0-
SNAPSHOT/maven-project-1.0-SNAPSHOT.pom
channel stopped
[DeployPublisher][INFO] Attempting to deploy 1 war file(s)
[DeployPublisher][INFO] Deploying /var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war to container Tomcat 8.x Remote
with context null
  [/var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war] is not deployed. Doing a fresh deployment.
  Deploying [/var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war]
Finished: SUCCESS
```

If we access our tomcat server Manager App we can notice the presence of a new directory named /webapp:



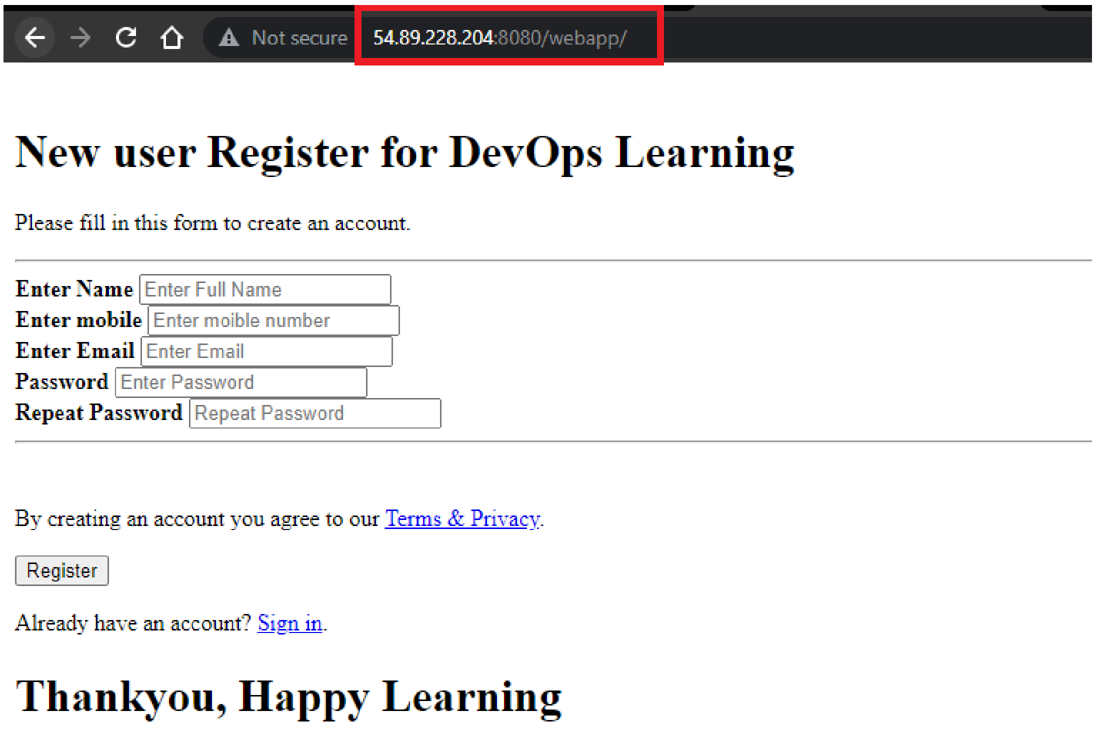
Tomcat Web Application Manager

| | |
|---|---|
| Message: | OK |

| Manager | | | |
|---|---|---|---|
| List Applications | HTML Manager Help | Manager Help | Server Status |

| Applications | | | | | |
|---|---|---|---|---|---|
| Path | Version | Display Name | Running | Sessions | Commands |
| / | None specified | Welcome to Tomcat | true | 0 | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /docs | None specified | Tomcat Documentation | true | 0 | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /examples | None specified | Servlet and JSP Examples | true | 0 | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /host-manager | None specified | Tomcat Host Manager Application | true | 0 | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /jsp_app | None specified | | true | 0 | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /manager | None specified | Tomcat Manager Application | true | 1 | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /webapp | None specified | Webapp | true | 0 | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |

and if we click on the webapp link, it will display the below page:



Not secure 54.89.228.204:8080/webapp/

New user Register for DevOps Learning

Please fill in this form to create an account.

| | |
|---|---|
| Enter Name | Enter Full Name |
| Enter mobile | Enter moible number |
| Enter Email | Enter Email |
| Password | Enter Password |
| Repeat Password | Repeat Password |

By creating an account you agree to our Terms & Privacy.

Register

Already have an account? Sign in.

Thankyou, Happy Learning

Hence we have successfully built and deployed our code. However, in this scenario, we have manually built the code and in case there is any change in our code we need to again manually click on the **Build Now** option in Jenkins and start the process all over again.

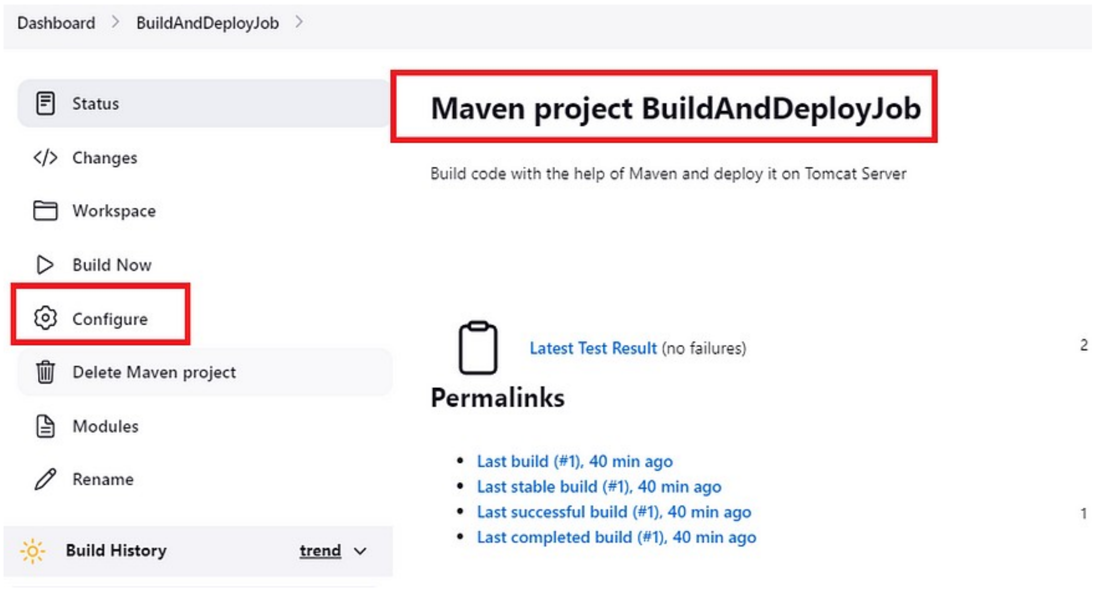
Jenkins has provided many options to automate the build trigger process and one of them is **Poll SCM.**

What is polling the SCM?

“Poll SCM” polls the SCM periodically for checking if any changes/ new commits were made and shall build the project if any new commits were pushed since the last build.

**Configure Poll SCM in Jenkins:**

Go to the previous job we created in the previous steps and click on configure:



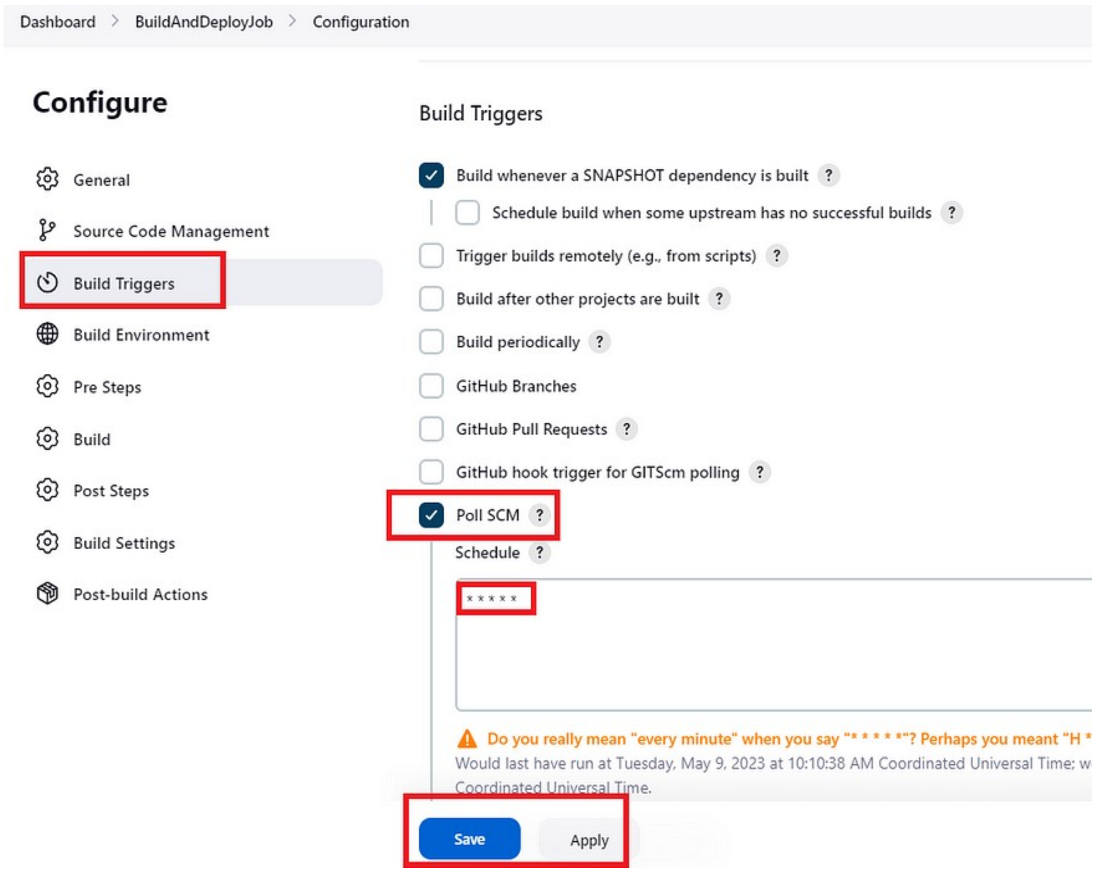
Maven project BuildAndDeployJob

Build code with the help of Maven and deploy it on Tomcat Server

Permalinks

- Last build (#1), 40 min ago
- Last stable build (#1), 40 min ago
- Last successful build (#1), 40 min ago
- Last completed build (#1), 40 min ago

Under **Build Triggers**, Select the **Poll SCM** option and set the schedule for the poll to happen. Here we will select the poll that should check every minute, every hour, every day of the month, month, and every day of the week. Click on Apply and Save to proceed.



Build Triggers

- [✓] Build whenever a SNAPSHOT dependency is built
  - [ ] Schedule build when some upstream has no successful builds
- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub Branches
- [ ] GitHub Pull Requests
- [ ] GitHub hook trigger for GITScm polling
- [✓] Poll SCM

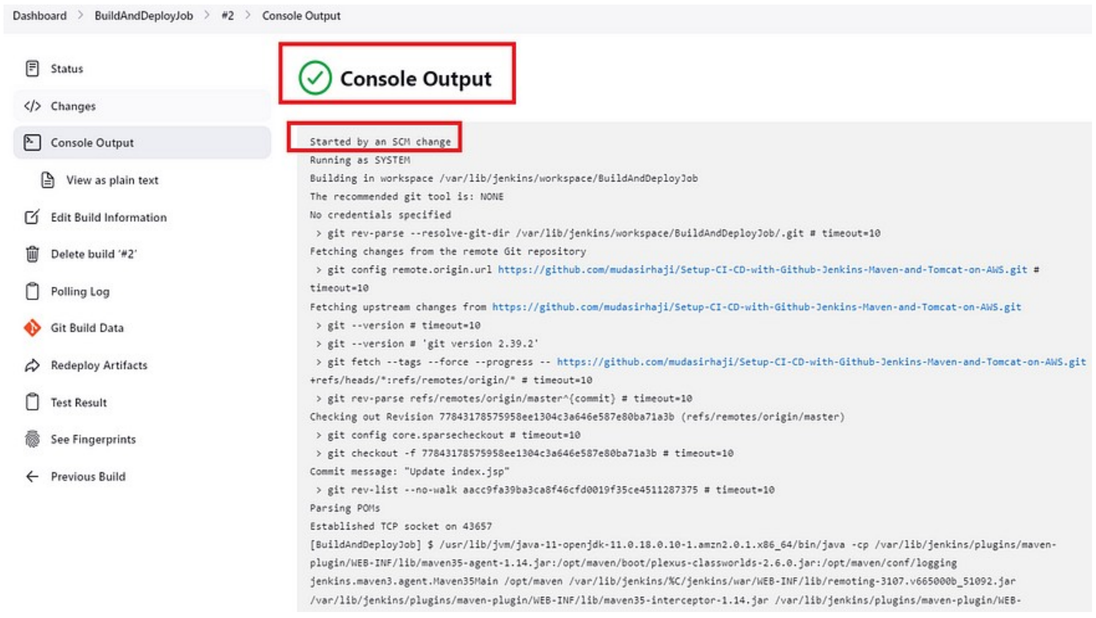
Schedule

```
* * * * *
```

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H *
Would last have run at Tuesday, May 9, 2023 at 10:10:38 AM Coordinated Universal Time; w
Coordinated Universal Time.

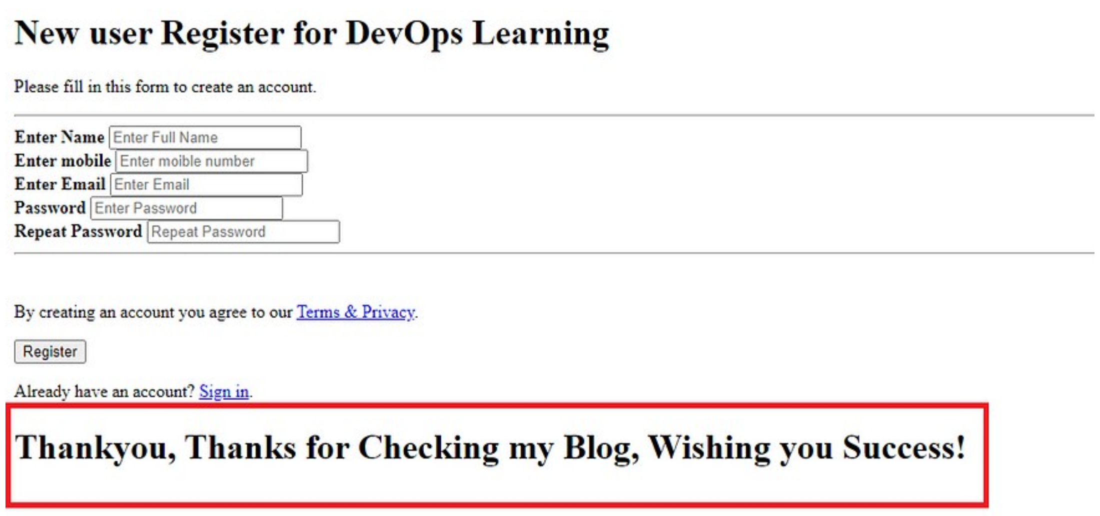
Now do some changes in your source code and without our intervention the Jenkins build process should be triggered automatically and build the code.

After making some changes in my code the build got triggered and checking the console output shows the build has been started by SCM and is successful rather than by the Admin (in an earlier case):



```
Started by an SCM change
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/BuildAndDeployJob
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/BuildAndDeployJob/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/mudasirhaji/Setup-CI-CD-with-Github-Jenkins-Maven-and-Tomcat-on-AWS.git #
timeout=10
Fetching upstream changes from https://github.com/mudasirhaji/Setup-CI-CD-with-Github-Jenkins-Maven-and-Tomcat-on-AWS.git
 > git --version # timeout=10
 > git --version # 'git version 2.39.2'
 > git fetch --tags --force --progress -- https://github.com/mudasirhaji/Setup-CI-CD-with-Github-Jenkins-Maven-and-Tomcat-on-AWS.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 77843178575958ee1304c3a646e587e80ba71a3b (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 77843178575958ee1304c3a646e587e80ba71a3b # timeout=10
Commit message: "Update index.jsp"
 > git rev-list --no-walk aacc9fa39ba3ca8f46cfd0019f35ce4511287375 # timeout=10
Parsing POMs
Established TCP socket on 43657
[BuildAndDeployJob] $ /usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.amzn2.0.1.x86_64/bin/java -cp /var/lib/jenkins/plugins/maven-
plugin/WEB-INF/lib/maven35-agent-1.14.jar:/opt/maven/boot/plexus-classworlds-2.6.0.jar:/opt/maven/conf/logging
jenkins.maven3.agent.Maven35Main /opt/maven /var/lib/jenkins/%C/jenkins/war/WEB-INF/lib/remoting-3107.v665000b_51092.jar
/var/lib/jenkins/plugins/maven-plugin/WEB-INF/lib/maven35-interceptor-1.14.jar /var/lib/jenkins/plugins/maven-plugin/WEB-
```

If we access our Tomcat server from our browser we should see the new changes we did:



New user Register for DevOps Learning

Please fill in this form to create an account.

Enter Name Enter Full Name
Enter mobile Enter moible number
Enter Email Enter Email
Password Enter Password
Repeat Password Repeat Password

By creating an account you agree to our Terms & Privacy.

Register

Already have an account? Sign in.

Thankyou, Thanks for Checking my Blog, Wishing you Success!

## Conclusion

In this blog, we learned how to build a Java web app using GitHub as our SCM, Jenkins as our CI tool, Maven as our build tool, and finally deploying on a remote Tomcat Server.