1 . Installing Jenkins in amazon linux in ec2

2. converting pem to ppk by load in putty gen and then save privatekey , open putty and add public ip of ec2 instance ..to get the public ip go to actions and sleect connect

3. after logging in create user Jenkins and give sudo permissions

4. sudo yum install docker

Docker service start

systemctl enable docker.service

or sudo chkconfig docker on

sudo yum install -y git

sudo reboot

**installing java and setting env variables**

**worked for centos vm oct5 2020-10-04**

# Set JAVA\_HOME on CentOS / RHEL / Fedora

I’ve seen many questions on how to set JAVA\_HOME on CentOS / Fedora / RHEL Linux distributions. JAVA\_HOME is used to set the path of Java installation on a Linux or Windows system. JAVA\_HOME is just a convention and it is usually used by Java EE and Tomcat servers and build tools such as Gradle, Ant and Maven to find where Java is installed.

In this guide I’ll show you an easy and recommended way of setting JAVA\_HOME on CentOS / Fedora / RHEL Linux system. We assume you already have Java installed before you can set JAVA\_HOME.

Install Java on [CentOS 7](https://computingforgeeks.com/how-to-install-java-11-on-centos-7-fedora-29-fedora-28/), [Fedora](https://computingforgeeks.com/how-to-install-java-11-on-centos-7-fedora-29-fedora-28/), [RHEL/ CentOS 8](https://computingforgeeks.com/how-to-install-java-11-openjdk-11-on-rhel-8/).

## Set JAVA\_HOME on CentOS / Fedora / RHEL

If you have more than one version of Java installed, you may want to set default version before you configure JAVA\_HOME on CentOS / Fedora / RHEL system. For this, use the command below.

sudo alternatives --config java

This will give you a prompt to confirm the default Java version you want to set.

There are 2 programs which provide 'java'.

Selection Command

-----------------------------------------------

\* 1 java-1.8.0-openjdk.x86\_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-2.el7\_6.x86\_64/jre/bin/java)

+ 2 java-1.7.0-openjdk.x86\_64 (/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.211-2.6.17.1.el7\_6.x86\_64/jre/bin/java)

Enter to keep the current selection[+], or type selection number: 1

You can set JAVA\_HOME in .bash\_profile, .bashrc file or for all Global users in /etc/profile or as bash function inside /etc/profile.d/ directory.

Add below line to any of bash dotfiles mentioned above.

export JAVA\_HOME=$(dirname $(dirname $(readlink $(readlink $(which javac)))))

Then source the file. Suppose you added this to ~/.bashrc, you’ll run:

source ~/.bashrc

Confirm Environment variable value.

$ **echo $JAVA\_HOME**  
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-2.el7\_6.x86\_64

You also need to add Java /bin directory to your PATH

export PATH=$PATH:$JAVA\_HOME/bin

Java CLASSPATH can be set using:

export CLASSPATH=.:$JAVA\_HOME/jre/lib:$JAVA\_HOME/lib:$JAVA\_HOME/lib/tools.jar

So your complete setting will have the lines:

export JAVA\_HOME=$(dirname $(dirname $(readlink $(readlink $(which javac)))))  
export PATH=$PATH:$JAVA\_HOME/bin  
export CLASSPATH=.:$JAVA\_HOME/jre/lib:$JAVA\_HOME/lib:$JAVA\_HOME/lib/tools.jar

Here is my screenshot.

Don’t forget to source the file or logout and back in.

**# Examples**$ source ~/.bashrc  
$ source ~/.bash\_profile  
$ source /etc/profile  
$ source /etc/profile.d/java.sh

Then confirm:

$ echo $JAVA\_HOME  
$ echo $PATH  
$ echo $CLASSPATH

And that’s all. You application should locate the Java installation directory.

**method1**

**SIMPLE STEPS.**

**First, check the version of JDK**

**Step 1**:**Ensure you have right version of JAVA installed or upgrade if necessary** (Optional)

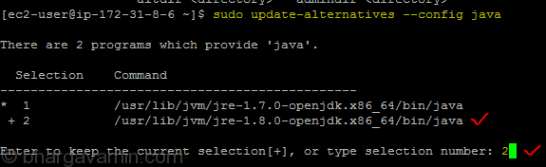
The latest version of JAVA is 1.8, by default Amazon Linux has JAVA version 1.7 so you would directly want to upgrade it use below command :

sudo yum install java-1.8.0-openjdk.x86\_64

If you already have JAVA installed you can change/check the JAVA version using below command.

$ sudo update-alternatives --config java

Select an option as shown in the image below:



Note: It is advisable to remove the previous version so that it doesn’t switch back.

Command to check package is installed or not

$ rpm -qa | grep nano nanao is the package name

Also, you can use Yum command like below.

$ yum list installed|grep 'nano'

**Step 2:** **Find out where JAVA is!**

For Linux systems, you can recursively run the commandfile followed by whichcommand to find the JAVA installation location as shown in the image below.

$ file $(which java)

/usr/bin/java: symbolic link to `/etc/alternatives/java'

The above output shows that java is pointing to a /etc/alternatives/java file but that not the actual location of JAVA hence you will need to dig in more to fetch its actual path.

**Step 3** : **Follow the lead!**

In the previous step, we located /etc/alternatives/java file this file will get us to the actual location where JAVA config files are.

Run the file command on that location /etc/alternatives/java.

$ file /etc/alternatives/java

/etc/alternatives/java: symbolic link to `/usr/lib/jvm/java-8-openjdk.x86\_64/bin/java'

There you go… You’ve now located JAVA config file location which we will use in below steps to set JAVA environment variable

You can re-affirm the location running  file command on the symbolic path:

$ file /usr/lib/jvm/java-8-openjdk.x86\_64/bin/java

/usr/lib/jvm/java-8-openjdk.x86\_64/bin/java: ELF 64-bit LSB executable...

This means that the JAVA is installed perfectly, Now go ahead and copy the path of above output

/usr/lib/jvm/jdk-1.8.0-openjdk.x86\_64/bin

**Step 4:  Set JAVA environment variable**

To set the JAVA\_HOME environment variables on Linux/Unix go to .baschrc file.

**Note:** .bashrc file is different for each user in Linux, hence you will need to update the same file for every user you want to set environment variable for.

Copy paste below two lines in the .bashrc file found in home the directory of ec2-user and root user:

 export JAVA\_HOME="/usr/lib/jvm/jdk-1.8.0-openjdk.x86\_64"

 PATH=$JAVA\_HOME/bin:$PATH

Save the file.

**Method2**

**OpenJDK Installation**

sudo yum install wget

yum install java-1.8.0-openjdk

**Procedure**

1. Find the JRE installation home directory.

Example: /usr/lib/jvm/jre1.8.0\_65

1. Export it in the JAVA\_HOME environment variable.

Example:

export JAVA\_HOME=/usr/lib/jvm/jre1.8.0\_65

export PATH=$JAVA\_HOME/bin:$PATH

1. Add these lines at the end of the user profiles in the ~/.bash\_profile file or, as a superuser, at the end of the global profiles in the /etc/profile file.
2. Restart system and login Log on again.

**MAVNE INSTALLATION WORKED**

# How to Install Apache Maven on CentOS 7

[**Ravi Saive**](https://www.tecmint.com/author/admin/)**August 10, 2018 Categories**[**CentOS**](https://www.tecmint.com/category/linux-distros/centos/)[**6 Comments**](https://www.tecmint.com/install-apache-maven-on-centos-7/#comments)

**Apache Maven** is a open source software project management and build automation tool, that is based on the conception of a project object model (**POM**), which is primarily used for deploying Java-based applications, but can also be used on projects written in **C#**, **Ruby** and other programming languages.

In this article, I will explain how to install and configure latest version of **Apache Maven** on a **CentOS 7** system (the given instructions also works on **RHEL** and **Fedora** distribution).

#### Prerequisites

* A newly deployed or existing CentOS 7 server instance.
* **Java Development Kit (JDK)** – Maven 3.3+ require JDK 1.7 or above to execute.

### Install OpenJDK 8 in CentOS 7

**Java Development Kit (JDK)** is a primary requirement to install **Apache Maven**, so first install Java on CentOS 7 system from the default repository and verify the version using following commands.

# yum install -y java-1.8.0-openjdk-devel

# java -version

If installation went well, you see the following output.

openjdk version "1.8.0\_141"

OpenJDK Runtime Environment (build 1.8.0\_141-b16)

OpenJDK 64-Bit Server VM (build 25.141-b16, mixed mode)

### Install Apache Maven in CentOS 7

Next, go to the [official Apache Maven download](https://maven.apache.org/download.cgi) page and grab the latest version or use the following wget command to download it under the maven home directory **‘/usr/local/src’**.

# cd /usr/local/src

# wget http://www-us.apache.org/dist/maven/maven-3/3.5.4/binaries/apache-maven-3.5.4-bin.tar.gz

Extract the downloaded archive file, and rename it using following commands.

# tar -xf apache-maven-3.5.4-bin.tar.gz

# mv apache-maven-3.5.4/ apache-maven/

### Configure Apache Maven Environment

Now we need to configure the environments variables to pre-compiled Apache Maven files on our system by creating a configuration file **‘maven.sh’** in the **‘/etc/profile.d’** directory.

# cd /etc/profile.d/

# vim maven.sh

Add the following configuration in **‘maven.sh’** configuration file.

# Apache Maven Environment Variables

# MAVEN\_HOME for Maven 1 - M2\_HOME for Maven 2

export M2\_HOME=/usr/local/src/apache-maven

export PATH=${M2\_HOME}/bin:${PATH}

Now make the **‘maven.sh’** configuration file executable and then load the configuration by running the **‘source’** command.

# chmod +x maven.sh

# source /etc/profile.d/maven.sh

### Check Apache Maven Version

To verify Apache Maven installation, run the following **maven** command.

# mvn --version

And you should get a output similar to the following:

Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T19:33:14+01:00)

Maven home: /usr/local/src/apache-maven

Java version: 9.0.4, vendor: Oracle Corporation, runtime: /opt/java/jdk-9.0.4

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "4.17.6-1.el7.elrepo.x86\_64", arch: "amd64", family: "unix"

That’s It! You have successfully installed **Apache Maven 3.5.4** on your **CentOS 7** system. If you have any problems related to installation, do share with us in the comment section.

**OFFICIAL WEBSITE FOR MAVEN**

<https://maven.apache.org/download.cgi>

Create directory in /opt a s maven (u can create direcrory anywhere)

Download the tar for linux by using

wget https://www-us.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz

Extract the gzip file

Tar –zxvf

And go to /opt/maven/apachemaven/

Now set the maven path

export M2\_HOME=/opt/maven/

export PATH=/opt/maven/bin:$PATH

**TO MAKE PATH PERMANENT GO TO USER HOME DIRECTORY AND EDIT ls –a**

**.BASH\_PROFILE AND PALCE THIS STUFF**

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.191.b12-1.el7\_6.x86\_64/

export M2\_HOME=/opt/maven/apache-maven-3.6.0/

export PATH=$PATH:$JAVA\_HOME/bin:$M2\_HOME/bin:$HOME/bin

**INATALLING JENKINS USING Repos in linux or centos**

Previously during software development, developers would submit their code to a code repository like [GitHub or Git Lab](https://www.tecmint.com/github-alternatives-to-host-open-source-projects/) usually, the source code would be fraught with bugs and errors. To make it even worse, developers would have to wait until the entire source code was built & tested to check for errors. This was tedious, time-consuming and frustrating. There was no iterative improvement of code, and overall, the software delivery process was slow. Then came **Jenkins**.

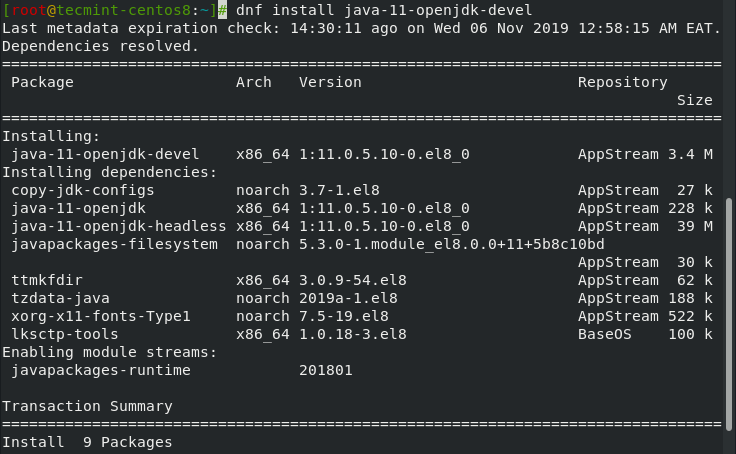
**Jenkins** is a free and opensource continuous integration tool written in **Java** that allows developers to continuously develop, test and deploy code in a simple and effective way. It automates tasks thereby saving time and takes away the stressful part of the software development process.

In this article, we demonstrate how you can install **Jenkins** on **CentOS 8** Linux.

### Step 1: Install Java on CentOS 8

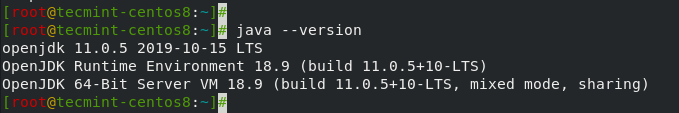
For **Jenkins** to function, you need to install either **Java JRE 8** or **Java 11**. In the example below, we decided to go with the installation of **Java 11**. Therefore, to install **Java 11**, run the command.

# dnf install java-11-openjdk-devel

[](https://www.tecmint.com/wp-content/uploads/2019/11/Install-Java-on-CentOS-8.png)Install Java on CentOS 8

To verify the installation of **Java 11**, run the command.

# java --version

[](https://www.tecmint.com/wp-content/uploads/2019/11/Check-Java-Version.png)Check Java Version

The output confirms that **Java 11** has been successfully installed.

### Step 2: Add Jenkins Repository on CentOS 8

Since **Jenkins** is not available in **CentOS 8** repositories, therefore we are going to add **Jenkins Repository** manually to the system.

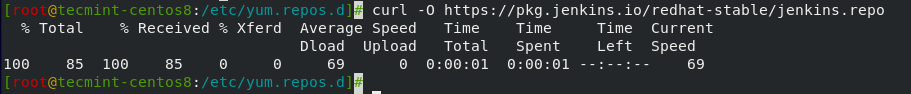
Begin by adding **Jenkins Key** as shown.

# rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

Now append Jenkin’s repository to **CentOS 8**.

# cd /etc/yum/repos.d/

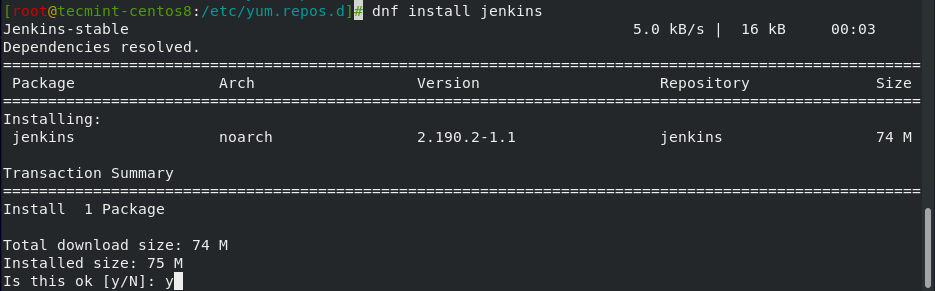
# curl -O https://pkg.jenkins.io/redhat-stable/jenkins.repo

[](https://www.tecmint.com/wp-content/uploads/2019/11/Add-Jenkins-Repository-on-CentOS-8.png)Add Jenkins Repository on CentOS 8

### Step 3: Install Jenkins on CentOS 8

Having successfully added **Jenkins** repository, you can proceed to install **Jenkins** by running.

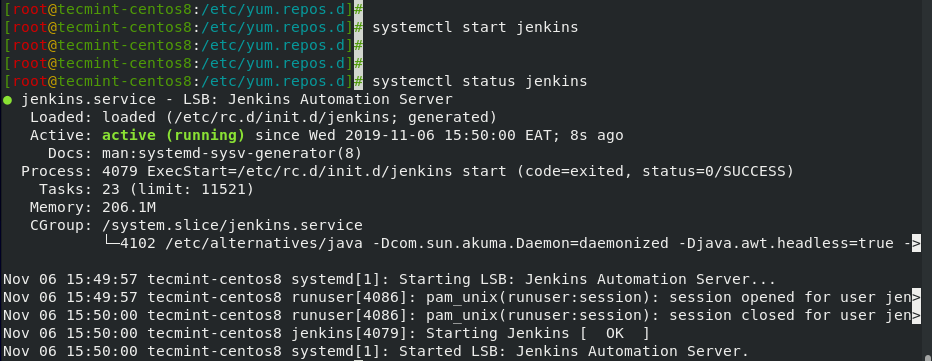
# dnf install jenkins

[](https://www.tecmint.com/wp-content/uploads/2019/11/Install-Jenkins-on-CentOS-8.png)Install Jenkins on CentOS 8

Once installed, start and verify the status of **Jenkins** by executing the commands.

# systemctl start jenkins

# systemctl status jenkins

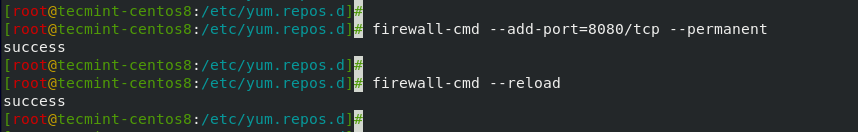
[](https://www.tecmint.com/wp-content/uploads/2019/11/Start-and-Verify-Jenkins-Status.png)Start and Verify Jenkins Status

The output above shows that Jenkins is up and running.

Next, you need to configure the firewall to allow access to port **8080** which is used by **Jenkins**. To open the port on the firewall, run the commands.

# firewall-cmd --add-port=8080/tcp --permanent

# firewall-cmd --reload

[](https://www.tecmint.com/wp-content/uploads/2019/11/Open-Jenkins-Port-on-Firewall.png)Open Jenkins Port on Firewall

### Step 4: Setting up Jenkins on CentOS 8

With the initial configurations done, the only remaining part is setting up **Jenkins** on a web browser. To achieve this, browse your server’s IP address as shown:

http://server-IP:8080

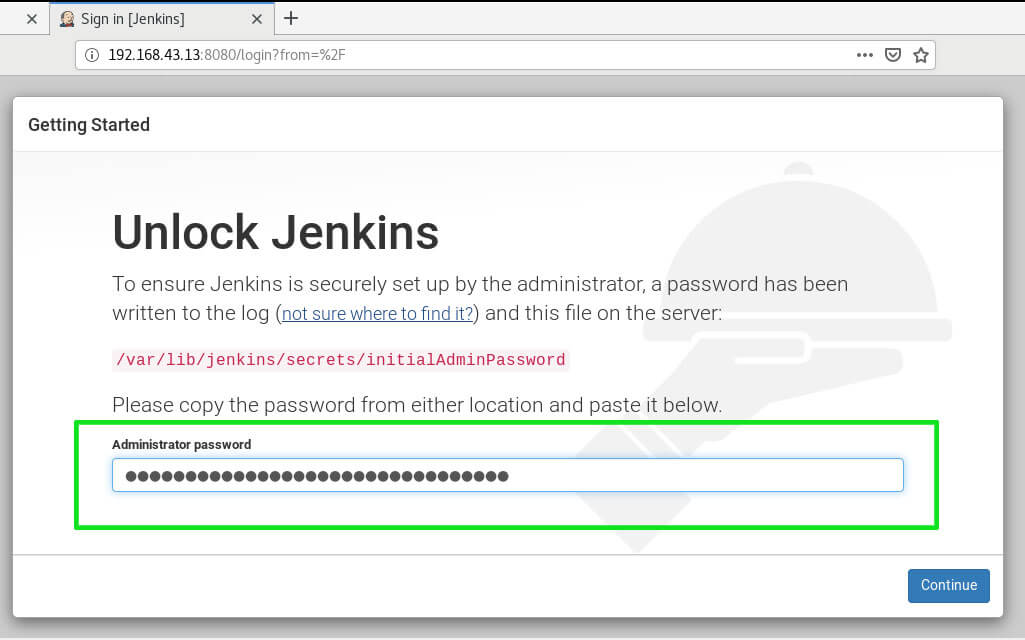
The first section requires you to unlock **Jenkins** using a password. This password is placed in the file **/var/lib/Jenkins/secrets/initialAdminPassword** file.

To read the password, simply use the [cat command](https://www.tecmint.com/13-basic-cat-command-examples-in-linux/) as shown.

# cat /var/lib/Jenkins/secrets/initialAdminPassword

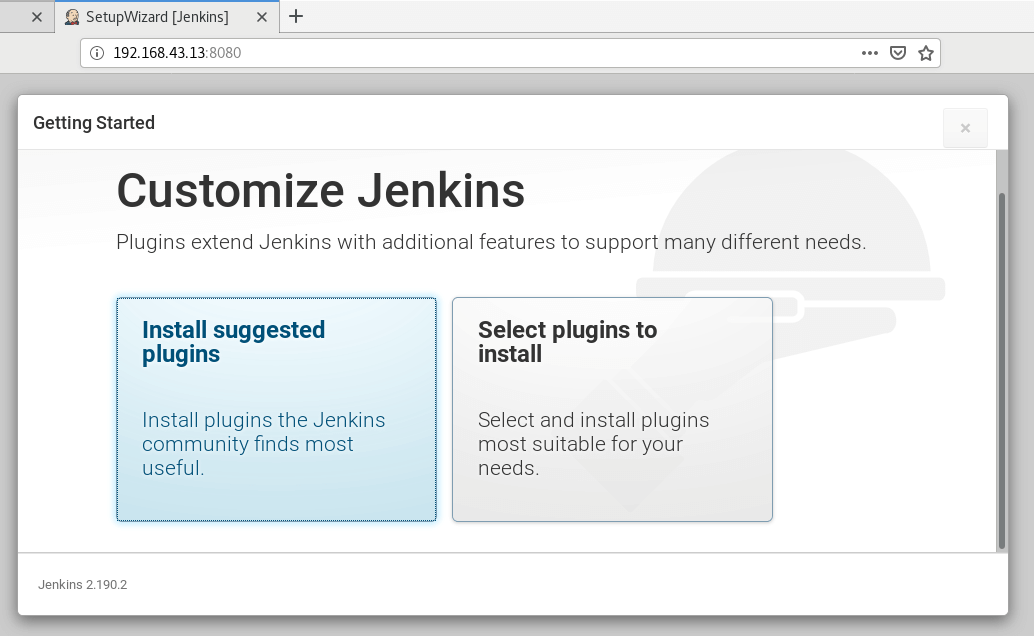
[](https://www.tecmint.com/wp-content/uploads/2019/11/View-Jenkins-Admin-Password.png)View Jenkins Admin Password

Copy & paste the password in the Administrator password text field & click ‘**Continue**‘.

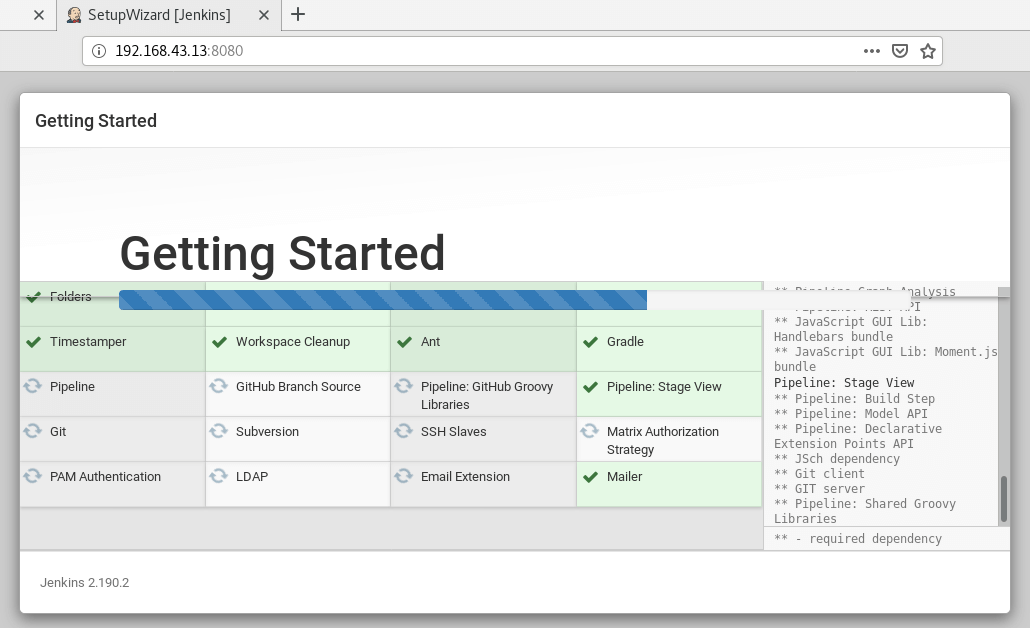
[](https://www.tecmint.com/wp-content/uploads/2019/11/Unlock-Jenkins.jpg)Unlock Jenkins

In the second stage, you will be presented with 2 options: ‘**Install using suggested plugins**‘ or ‘**Select plugins to install**‘.

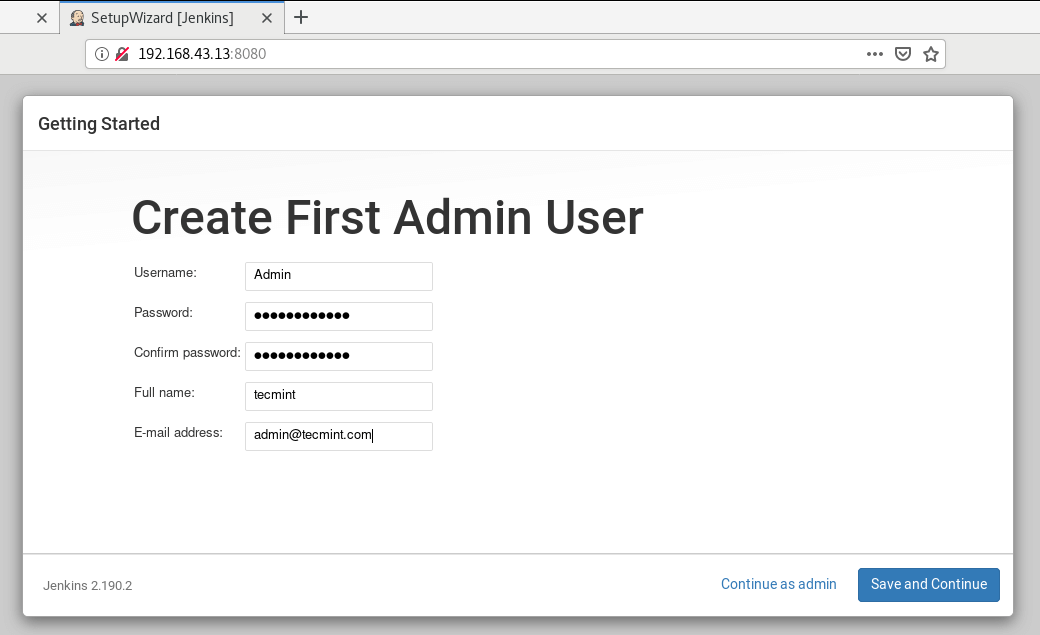
For now, click on ‘**Install using suggested plugins**‘ to install essential plugins for our setup.

[](https://www.tecmint.com/wp-content/uploads/2019/11/Install-Suggested-Plugins.png)Install Suggested Plugins

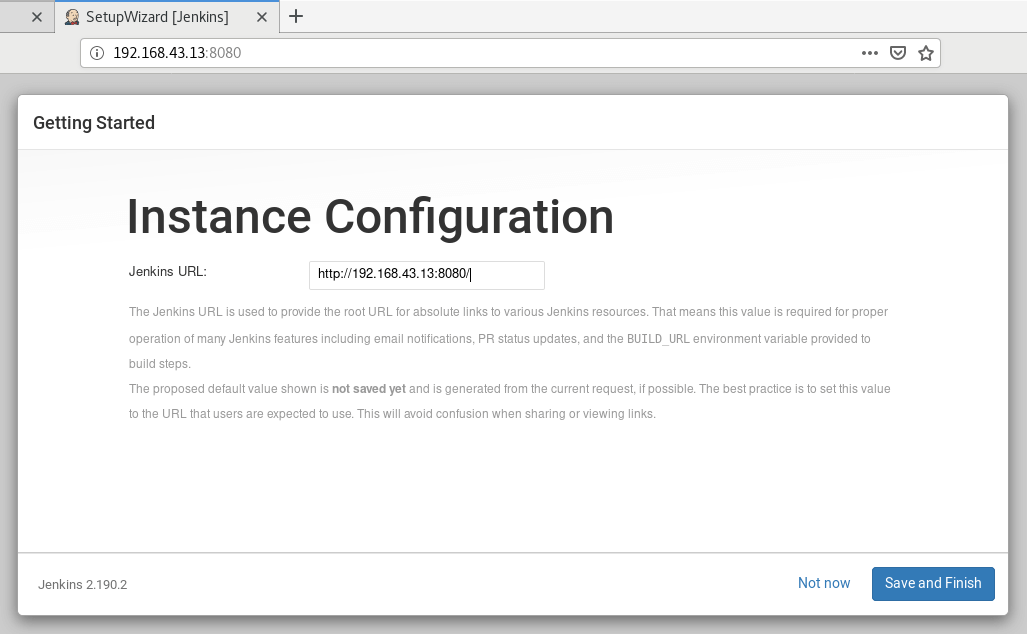
Shortly, the installation of the plugins will get underway.

[](https://www.tecmint.com/wp-content/uploads/2019/11/Jenkins-Plugin-Installation.png)Jenkins Plugin Installation

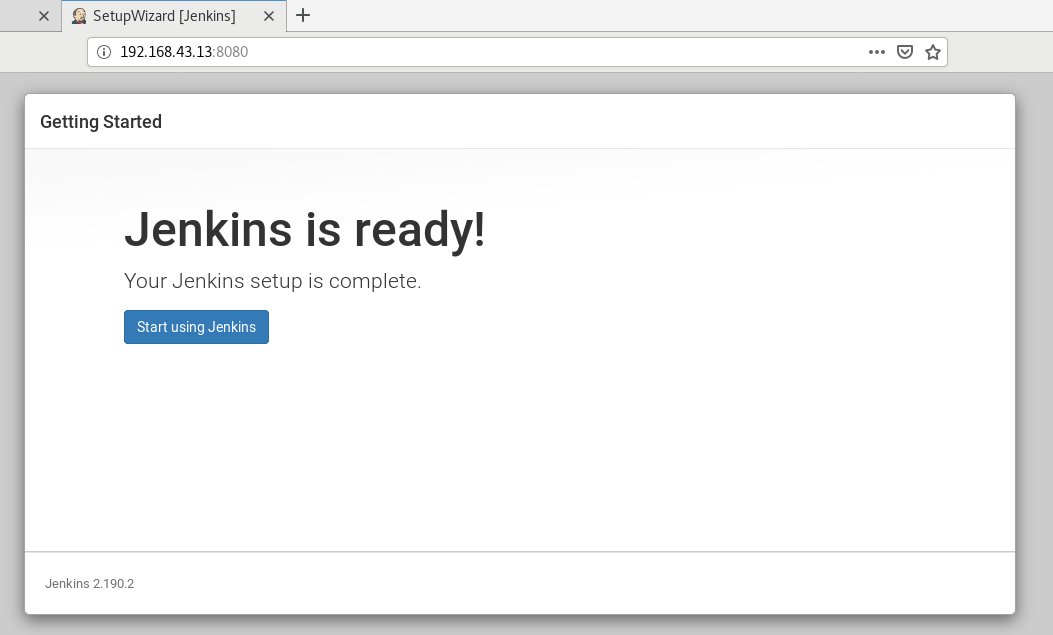
In the next section, fill out the fields in order to create the **First Admin** user. After you are done, click on ‘**Save and continue**‘.

[](https://www.tecmint.com/wp-content/uploads/2019/11/Create-Jenkins-Admin-User.png)Create Jenkins Admin User

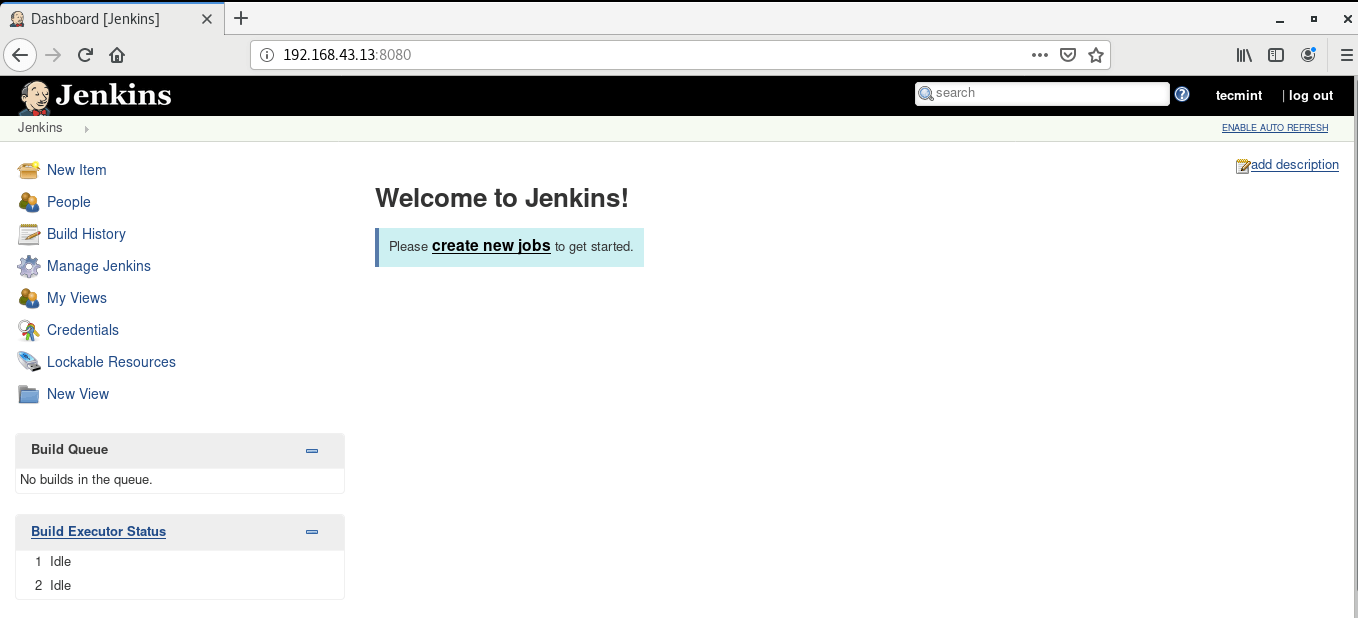
The ‘**Instance Configuration**‘ section will provide you with the default Jenkins URL. For simplicity, it’s recommended to leave it as it is and click ‘**Save and Finish**‘.

[](https://www.tecmint.com/wp-content/uploads/2019/11/Jenkins-Instance-Configuration.png)Jenkins Instance Configuration

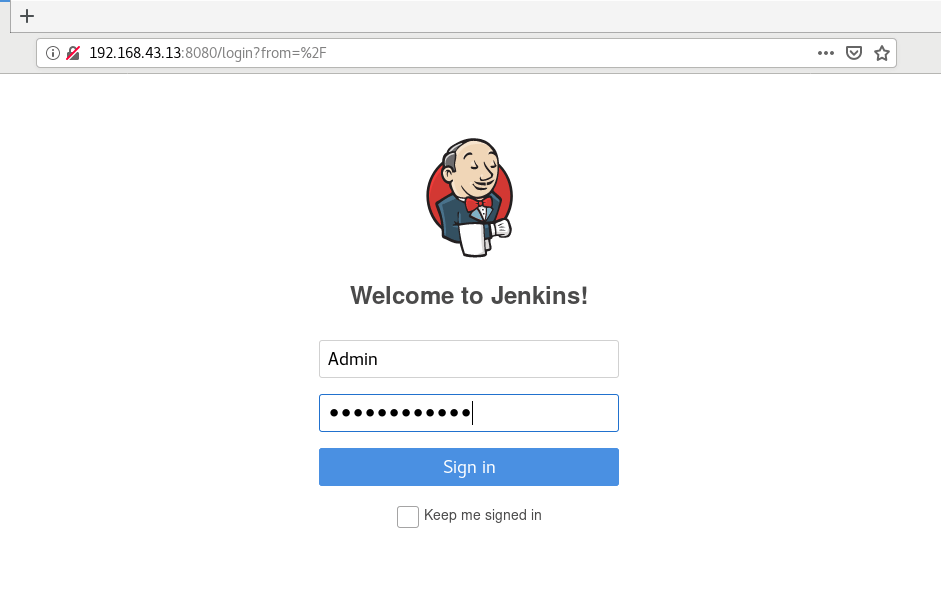
At this point, **Jenkins** setup is now complete. To access the Jenkins dashboard, simply click on ‘**Start using Jenkins**‘.

[](https://www.tecmint.com/wp-content/uploads/2019/11/Jenkins-Installation-Complete.png)Jenkins Installation Complete

Jenkins’s dashboard is displayed below.

[](https://www.tecmint.com/wp-content/uploads/2019/11/Jenkins-Dashboard.png)Jenkins Dashboard

Next time you log into **Jenkins**, simply provide the **Admin** username and the password you specified when creating the Admin user.

[](https://www.tecmint.com/wp-content/uploads/2019/11/Jenkins-Admin-Login.png)Jenkins Admin Login

##### Conclusion

That was a step-by-step procedure of how to install **Jenkins Continuous Integration** tool on **CentOS 8**. To learn more about **Jenkins**. Read [Jenkins Documentation](https://jenkins.io/doc/book/getting-started/). Your feedback on this guide is most welcome.

Tags

**INATALLING JENKINS USING DOCKER**

docker pull jenkins:2.60.3

docker images

**to run a container of Jenkins**

**installing sonar-qube**

[From a Docker image](https://docs.sonarqube.org/latest/setup/get-started-2-minutes/)

Find the Community Edition Docker image on [Docker Hub](https://hub.docker.com/_/sonarqube/).

1. Start the server by running:

$ docker run -d --name sonarqube -p 9000:9000 <image\_name>

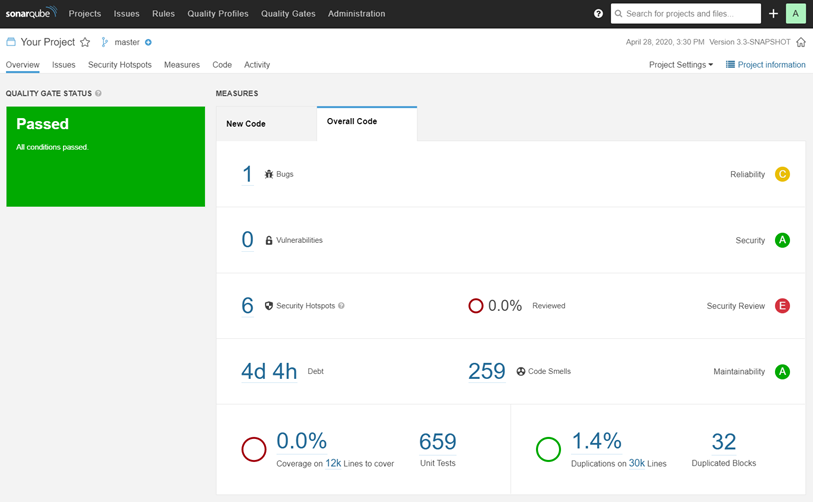
1. Log in to [http://localhost:9000](http://localhost:9000/) with System Administrator credentials (login=admin, password=admin).

## Analyzing a Project

Now that you're logged in to your local SonarQube instance, let's analyze a project:

1. Click the **Create new project** button.
2. When asked **How do you want to create your project**, select **Manually**.
3. Give your project a **Project key** and a **Display name** and click the **Set Up** button.
4. Under **Provide a token**, select **Generate a token**. Give your token a name, click the **Generate** button, and click **Continue**.
5. Select your project's main language under **Run analysis on your project**, and follow the instructions to analyze your project. Here you'll download and execute a Scanner on your code (if you're using Maven or Gradle, the Scanner is automatically downloaded).

After successfully analyzing your code, you'll see your first analysis on SonarQube:



**Installing sonatype nexus repo**

**docker pull sonatype/nexus3**

### Persistent Data

There are two general approaches to handling persistent storage requirements with Docker. See [Managing Data in Containers](https://docs.docker.com/engine/tutorials/dockervolumes/) for additional information.

1. Use a docker volume. Since docker volumes are persistent, a volume can be created specifically for this purpose. This is the recommended approach.
2. $ docker volume create --name nexus-data

$ docker run -d -p 8081:8081 --name nexus -v nexus-data:/nexus-data sonatype/nexus3

1. Mount a host directory as the volume. This is not portable, as it relies on the directory existing with correct permissions on the host. However it can be useful in certain situations where this volume needs to be assigned to certain specific underlying storage.
2. $ mkdir /some/dir/nexus-data && chown -R 200 /some/dir/nexus-data

$ docker run -d -p 8081:8081 --name nexus -v /some/dir/nexus-data:/nexus-data sonatype/nexus3

* Default user is admin and the uniquely generated password can be found in the admin.password file inside the volume. See [Persistent Data](https://hub.docker.com/r/sonatype/nexus3#user-content-persistent-data) for information about the volume.
* It can take some time (2-3 minutes) for the service to launch in a new container. You can tail the log to determine once Nexus is ready:

$ docker logs -f nexus

Important note :When stopping, be sure to allow sufficient time for the databases to fully shut down.

docker stop --time=120 <CONTAINER\_NAME>

**For sonarqube anlaysis**

* Genereate a token in myaccount
* And in git repoor branch open pom.xml
* under sonar section give sonarqube server url and under sonar token section give sonar token which u have generated

**Pushing the artifacts to nexus repository u can do it in two ways**

**Method 1**

In the same pom.xml under nexus section paste the nexus server url and give credentials of nexus server

**Giving the credentials of nexus server in maven settings.xml**

1. in maven installation path vi /opt/maven/conf/settings.xml this is path for maven installed path in linux server
2. if you have installed maven automatically in manage Jenkins 🡪gobal tool configurations ,you can find the maven path under Jenkins home path /var/lib/Jenkins/tools/Hudson.taks.maven/maven-3.5./conf/settings.xml
3. In settings.xml under server section copy the server section and paste and edit the credentials nexus server and save the file

Eg: <server>

<id>deploymentRepo</id>

<username>repouser</username>

<password>repopwd</password>

</server>

-->

<!-- Another sample, using keys to authenticate.

<server>

<id>siteServer</id>

<privateKey>/path/to/private/key</privateKey>

<passphrase>optional; leave empty if not used.</passphrase>

</server>

-->

<server>

<id>nexus</id>

<username>admin</username>

<password>admin\_123</password>

</server>

</servers>

# Hosting Maven Repositories

A hosted Maven repository can be used to deploy your own as well as third-party components. A default installation of Nexus Repository Manager includes a two hosted Maven repositories. The maven-releases repository uses a release version policy and the maven-snapshots repository uses a snapshot version policy.

**now create a repository for maven build maven repository**

1. in nexus under settings icon repository –create repo for maven project take m2 group and create
2. now copy the create repo url of releases and paste it in the same pom.xml under distribution management section
3. and copy the created snapshot repo url of snapshots and paste it in the same pom.xml under snapshot repository section
4. and commit
5. and run the job in Jenkins

Method 2

install the nexus artefact plugin and configure it in Jenkins

in manage Jenkins section –configuration 🡪 you will get a nexus link and now

and now create a nexus user in Jenkins and give the credentials in th nexus section

and now copy the url of nexus server and paste here and save

now go to Jenkins job under bulid section select nexus artifacts and configure the details and now run the Jenkins job

Flow-1 Deploying the artifact to tomcat

You need to download apache tomcat 8 and unzip by using tar –xvzf

To start tomcat go to bin dir inside tomcat and ./startup.sh

To check tomcat is up or ps –ef | grep tomcat and check port number 8080

By default tomcat uses 8080 but Jenkins uses 8080 so u nedd to change the port number for tomcat ,u can change it by editing the settings.xml in connector section (line number 69 in file)and change the port number and stop tomcat and start the tomcat

To shutdown ./shutdown.sh

Now give permissions for startup.sh so that any user start up the tomcat ..dont give full permisiions in organization

Chmod +x startup.sh

Chmod +x shutdown.sh

Creating soft link for startup .sh and shutdown.sh

ln -s /opt/tomcat-8.5.49/bin/startup.sh /usr/local/bin/tomcatup ------for starting tomcat

ln -s /opt/tomcat-8.5.49/bin/shutdown.sh /usr/local/bin/tomcatdown ---for stopping tomcat

for tomcat logs /opt/tomcat/logs/catalina.out

tomcatup but page not loading for my centos 7

iptables -I OUTPUT -o -enp0s3 -d 0.0.0.0/0 -j ACCEPT

iptables -I INPUT -i -enp0s3 -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -I INPUT -j ACCEPT

and try to login now 403 acces denied because we didn’t give permmison for tomcat users to login so give permission by doing following things

**cmd :find / -name context.xml**

output :/opt/tomcat-8.5.49/conf/context.xml

/opt/tomcat-8.5.49/webapps/host-manager/META-INF/context.xml

/opt/tomcat-8.5.49/webapps/manager/META-INF/context.xml

vi /opt/tomcat-8.5.49/webapps/host-manager/META-INF/context.xml

comment the line number 19,20 as fallows

19 <!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"

20 allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->

vi /opt/tomcat-8.5.49/webapps/manager/META-INF/context.xml

19 <!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"

20 allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->

Now to add user got to /opt/tomcat/conf

And edit tomcat-users.xml

vi tomcat-users.xml

and paste following under second NOTE or from line 36

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<role rolename="manager-jmx"/>

<role rolename="manager-status"/>

<role rolename="admin-gui"/>

<role rolename="admin-script"/>

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

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

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

Now tomcat installation …configuration are all done

Now got Jenkins and manage plugins and in available section search Deplot to conatainer plugin and install it without restart

After restart go to your job configuration and in **post build Actions 🡪select🡪Deploy waror ear to container**

|  |  |  |
| --- | --- | --- |
| WAR/EAR files |  |  |
|  | |  |  |
|  |  | |  |
|  | Context path |  |  |
|  | |  |  |
|  |  | |  |
|  | Containers | Add Container  Add container tomcat 8.x Remote  Credentials: add tomcatcatuser in Jenkins database ..this user should be in the tomcat user.xml |  |
|  | |  |  |
|  | Deploy on failure |  |  |

Save the config of the job and build now the job

And check the war is deployed into tomcat or not by going into manager app

With one click jenkins will get the code form github

Build the pacakage by using maven

Source code analaysis using sonarqube

Copy the .war and snapshot to the nexus repository manager

And deploy the artefact to the dev or stage env into tomcat

And to upadate ur job u need to configure and now add to timestamp to the console output go to build env section and select **add timestamp to the console output**

In build env section select **delete workspace before build starts**

U need to maintain only last 5 build to do that got job config in general section **Discard old builds**

**Days to keep builds**

**Max# no.of builds give 5** here it will store only build information not the packages

**And to store the packages along job infm u need go to**

**advanced section**

**Max# no.of builds to keep with artifacts give 5**

The config done to this job ..sticks only to this job

**Upto now we are doing manual job triggering**

To automate Build trigger we have 3 options

**1.poll scm---- this is crontab format**

**Min Hours Dayofmonth Month Day of week**

**\***

**How Jenkins will know that difference bw previous build and current build**

**Based on the commit ids (if both commitids are same means we don’t have changes in the code )**

**Both Poll SCM and the build takes cron expression as the input. So, How do they actually differ?**

"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, whereas the "build"  shall build the project periodically irrespective to whether or not any changes were made.

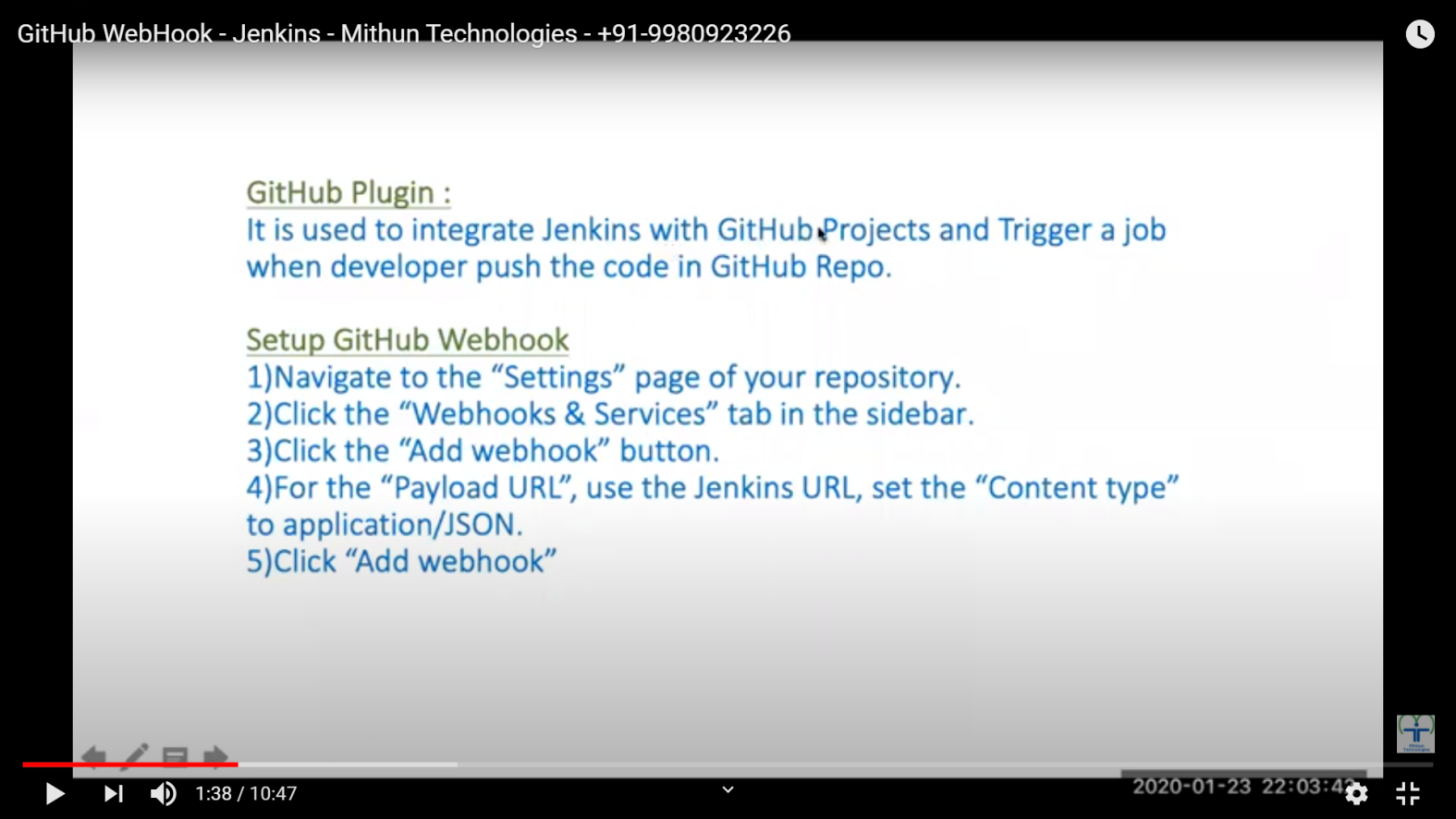
**2.Build periodically**

**this is also crontab format**

**Min Hours Dayofmonth Month Day of week**

**\***

**3.Github webhook**

****

**This is freestyle job …this type of job we are not using much in companies now**

Payload url ----- http://192.168.43.36:8080/github-webhook