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`

11/12/2020

Step 1 – Install Java on Amazon Linux

The OpenJDK 8 is available under default yum repositories and OpenJDK 11 is available under Amazon Linux 2 extras repositories. You can simply install Java 11 or Java 8 on the Amazon Linux system using the following commands.

- Run below commands to **install Java 11 on Amazon Linux**:

```
sudo amazon-linux-extras install java-openjdk11
```

- Run below commands to **install Java 8 on Amazon Linux**:

```
sudo yum install java-1.8.0-openjdk
```

Step 2 – Check Active Java Version

After successfully installing Java on Amazon Linux using the above steps, Let's verify the installed version using the following command.

```
java -version


openjdk version "1.8.0_222"

OpenJDK Runtime Environment (build 1.8.0_222-8u222-b10-1ubuntu1~18.04.1-b10)

OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
```

Step 3 – Switch Java Version

Use alternatives command-line utility to switch active Java version on your Amazon Linux system. Run below command from the command line and select the appropriate Java version to make it default.

```
alternatives --config java
```

```
[root@tecadmin ~]# alternatives --config java

There are 2 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
   1           java-11-openjdk.x86_64 (/usr/lib/jvm/java-11-openjdk-11.0.2.7
*+ 2           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.

Enter to keep the current selection[+], or type selection number: 1
[root@tecadmin ~]#
```

After switching let's check again active Java version:

```
java -version


openjdk version "11.0.7" 2020-04-14 LTS

OpenJDK Runtime Environment 18.9 (build 11.0.7+10-LTS)

OpenJDK 64-Bit Server VM 18.9 (build 11.0.7+10-LTS, mixed mode, sharing)
```

java8-centos-amazon-linux.md

```
# Remove java 7
sudo yum remove -y java

# Install basic packages
sudo yum install -y git

# Download and install java 8
wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F;
oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u131-
b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz"
tar -xzvf jdk-8u131-linux-x64.tar.gz
rm -rf jdk-8u131-linux-x64.tar.gz

# Configure JAVA_HOME
sudo vim ~/.bashrc
alias cls='clear'

export JAVA_HOME=~/jdk1.8.0_131
export JRE_HOME=~/jdk1.8.0_131/jre
export PATH=$PATH:~/jdk1.8.0_131/bin:/~/jdk1.8.0_131/jre/bin
source ~/.bashrc
java -version
```

installing java and setting env variables

worked for centos vm oct5 2020-10-04

How to Install Java 8 in CentOS / Amazon Linux?

Raw

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, Fedora, RHEL/ CentOS 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/t
ools.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/t
ools.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)

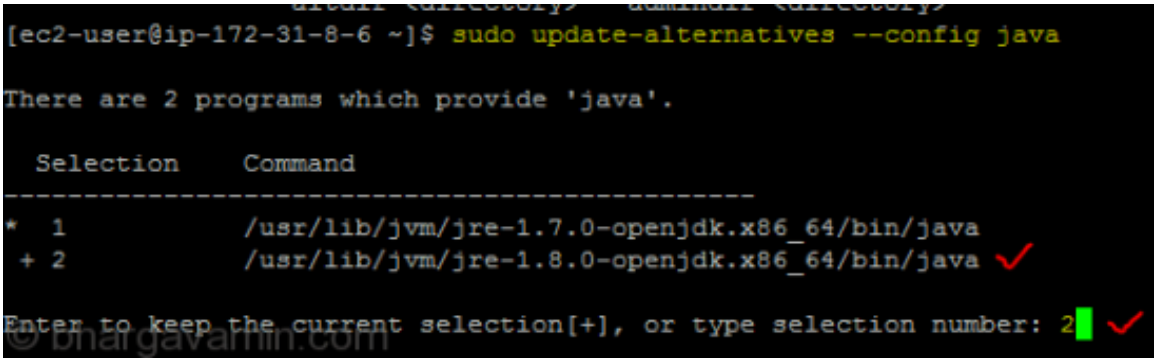
The latest version of JAVA is 1.8, by default Amazon Linux has JAVA version 1.7 so you would directly want to upgrade it use below command :

```
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 command`file` followed by `which`command 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`

2. 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
   ```

3. 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.

4. Restart system and login Log on again.

MAVNE INSTALLATION WORKED

How to Install Apache Maven on CentOS 7

Ravi SaiveAugust 10, 2018 CategoriesCentOS 6 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 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.

Installing Maven on Linux

To install Maven on the Linux operating system, download the latest version from the Apache Maven site, select the Maven binary tar.gz file, for example: apache-maven-3.3.9-bin.tar.gz.

Extract the archive to your desired location.

3.1. Adding Maven to the Environment Path

Open the command terminal and run the following commands to set the environment variables:

```
$ export M2_HOME=/usr/local/apache-maven/apache-maven-3.3.9

$ export M2=$M2_HOME/bin

$ export MAVEN_OPTS=-Xms256m -Xmx512m
```

with *M2_Home* path corresponding with the location of your extracted Maven files.

Now append the *M2* variable to the system path:

```
$ export PATH=$M2:$PATH
```

Finally, verify if Maven has been added by running:

```
$ mvn -version
```

The output should be as follows:

```
Apache Maven 3.3.3 (7994120775791599e205a5524ec3e0dfe41d4a06; 2016-12-03T17:27:37+05:30)

Maven home: /usr/local/apache-maven/apache-maven-3.3.9

Java version: 1.8.0_75, vendor: Oracle Corporation

Java home: /usr/local/java-current/jdk1.8.0_75/jre
```

You have successfully installed Maven on your Linux system.

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 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
```

```
[root@tecmint-centos8:~]# dnf install java-11-openjdk-devel
Last metadata expiration check: 14:30:11 ago on Wed 06 Nov 2019 12:58:15 AM E
Dependencies resolved.
================================================================================
 Package                   Arch   Version                        Repository
                                                                            Si
================================================================================
Installing:
 java-11-openjdk-devel     x86_64 1:11.0.5.10-0.el8_0            AppStream 3.4
Installing dependencies:
 copy-jdk-configs          noarch 3.7-1.el8                      AppStream  27
 java-11-openjdk           x86_64 1:11.0.5.10-0.el8_0            AppStream 228
 java-11-openjdk-headless  x86_64 1:11.0.5.10-0.el8_0            AppStream  39
 javapackages-filesystem   noarch 5.3.0-1.module_el8.0.0+11+5b8c10bd
                                                                 AppStream  30
 ttmkfdir                  x86_64 3.0.9-54.el8                   AppStream  62
 tzdata-java               noarch 2019a-1.el8                    AppStream 188
 xorg-x11-fonts-Type1      noarch 7.5-19.el8                     AppStream 522
 lksctp-tools              x86_64 1.0.18-3.el8                   BaseOS    100
Enabling module streams:
 javapackages-runtime             201801

Transaction Summary
================================================================================
Install  9 Packages
```

Install Java on CentOS 8

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

```
# java --version
```

```
[root@tecmint-centos8:~]#
[root@tecmint-centos8:~]# java --version
openjdk 11.0.5 2019-10-15 LTS
OpenJDK Runtime Environment 18.9 (build 11.0.5+10-LTS)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.5+10-LTS, mixed mode, sharing)
[root@tecmint-centos8:~]#
```

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
```

```
[root@tecmint-centos8:/etc/yum.repos.d]# curl -O https://pkg.jenkins.io/redha
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curre
                                 Dload  Upload   Total   Spent    Left  Speed
100    85  100    85    0     0     69      0  0:00:01  0:00:01 --:--:--    6
[root@tecmint-centos8:/etc/yum.repos.d]#
```

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.

```
# yum install jenkins
```

```
[root@tecmint-centos8:/etc/yum.repos.d]# dnf install jenkins
Jenkins-stable                                                    5.0 kB/
Dependencies resolved.
================================================================================
 Package                Arch                Version                     Repo
================================================================================
Installing:
 jenkins                noarch              2.190.2-1.1                 jenk

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

Total download size: 74 M
Installed size: 75 M
Is this ok [y/N]: y
```

Install Jenkins on CentOS 8

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

```
# systemctl start jenkins

# systemctl status jenkins
```

```
[root@tecmint-centos8:/etc/yum.repos.d]#
[root@tecmint-centos8:/etc/yum.repos.d]# systemctl start jenkins
[root@tecmint-centos8:/etc/yum.repos.d]#
[root@tecmint-centos8:/etc/yum.repos.d]#
[root@tecmint-centos8:/etc/yum.repos.d]# systemctl status jenkins
● jenkins.service - LSB: Jenkins Automation Server
   Loaded: loaded (/etc/rc.d/init.d/jenkins; generated)
   Active: active (running) since Wed 2019-11-06 15:50:00 EAT; 8s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 4079 ExecStart=/etc/rc.d/init.d/jenkins start (code=exited, status
    Tasks: 23 (limit: 11521)
   Memory: 206.1M
   CGroup: /system.slice/jenkins.service
           └─4102 /etc/alternatives/java -Dcom.sun.akuma.Daemon=daemonized -D

Nov 06 15:49:57 tecmint-centos8 systemd[1]: Starting LSB: Jenkins Automation
Nov 06 15:49:57 tecmint-centos8 runuser[4086]: pam_unix(runuser:session): ses
Nov 06 15:50:00 tecmint-centos8 runuser[4086]: pam_unix(runuser:session): ses
Nov 06 15:50:00 tecmint-centos8 jenkins[4079]: Starting Jenkins [  OK  ]
Nov 06 15:50:00 tecmint-centos8 systemd[1]: Started LSB: Jenkins Automation S
```

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
```

```
[root@tecmint-centos8:/etc/yum.repos.d]#
[root@tecmint-centos8:/etc/yum.repos.d]# firewall-cmd --add-port=8080/tcp --p
success
[root@tecmint-centos8:/etc/yum.repos.d]#
[root@tecmint-centos8:/etc/yum.repos.d]# firewall-cmd --reload
success
[root@tecmint-centos8:/etc/yum.repos.d]#
```

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 as shown.

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

```
[root@tecmint-centos8:~]#
[root@tecmint-centos8:~]# cat /var/lib/jenkins/secrets/initialAdminPassword
5f4b14812ba240d29ada422f5f2794de
[root@tecmint-centos8:~]#
```

View Jenkins Admin Password

Copy & paste the password in the Administrator password text field & click '**Continue**'.
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.
Install Suggested Plugins

Shortly, the installation of the plugins will get underway.

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**'.
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**'.
Jenkins Instance Configuration

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

Jenkins's dashboard is displayed below.

Jenkins Dashboard

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

Find the Community Edition Docker image on Docker Hub.

1. Start the server by running:

```
$ docker run -d --name sonarqube -p 9000:9000 <image_name>
```

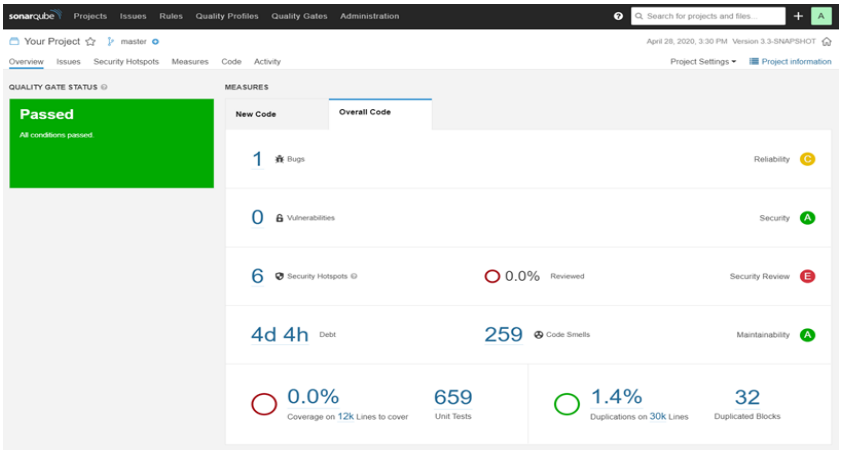
2. Log in to http://localhost:9000 with System Administrator credentials (login=admin, password=admin).

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 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
```

3. *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.

4.
```
$ 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 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 repo or 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/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

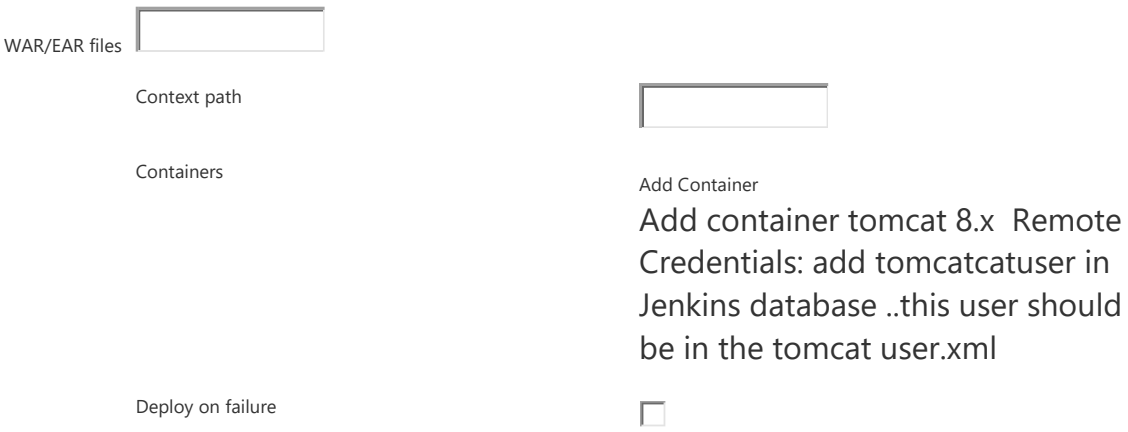
and paste following under second NOTE or from line 36

```
<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**



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

Payload delivery unsuccessful

- ✓ 192.168.0.32 is a private IP on your own network. GitHub needs a public IP to hit, so you'll need to actually enter your public IP there. Depending on how your network is configured, you'll need to configure port forwarding on your router (if it's a home internet connection). You can use something like ngrok for temporary testing.

Jenkins job configuration history plugin

This plugin saves **a copy of the configuration file** of jobs and agents (`config.xml`) for every change made and of the system configuration (`<config-name>.xml`). You can also see what changes have been made by which user if you configured a security policy.

Job Import Plugin

Import jobs from one Jenkins server to another Jenkins instance/server.

Useccase is migrating jobs og Jenkins sever1 to Jenkins server 2

Install the plugin in Jenkins sever 2 and go to configure system and search job import ang give the url of jenkins1 and give the credentials of Jenkins sever1

Now go to job config history plugin in sever2 and start

Jenkins master-slave configuration

Are u going install Jenkins in slave linux instance

Ans: no

We are going install Jenkins slave in another linux server

Prerequisites for master Jenkins and slave is Java jdk

If you are using maven u need to install maven in the slave serves

For improving the performance of master Jenkins we use master-slave configuration

Terraform Installation - Linux Server- Mithun

Terraform Installation

#Login as a root user in ec2 instance
sudo su -

#You will need to upgrade your system and packages
yum update -y

#Install wget and unzip packages
yum install wget unzip vim -y

#Download the terraform software.
#Use https://www.terraform.io/downloads.html to download the terraform software.

wget
https://releases.hashicorp.com/terraform/0.12.21/terraform_0.12.21_linux_amd64.zip

#Extract the terraform software.
unzip terraform_0.12.21_linux_amd64.zip -d /usr/local/bin/

#Check the version
terraform -v (OR) terraform version

#Help
terraform -help