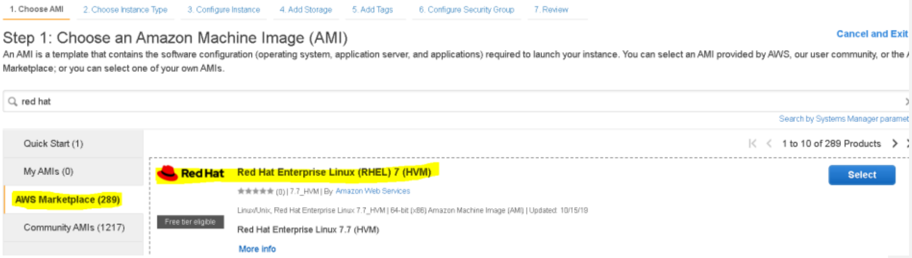
Configuration Management Using Ansible

Module 5: Automating Continuous Integration

Demo Document - 4

Jenkins-Server Installation:

1. On the AWS Management Console, click launch instance, and choose **Red Hat Enterprise Linux (RHEL) 7 (HVM)** AMI:



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the
Marketplace; or you can select one of your own AMIs.

red hat

Red Hat Enterprise Linux (RHEL) 7 (HVM)

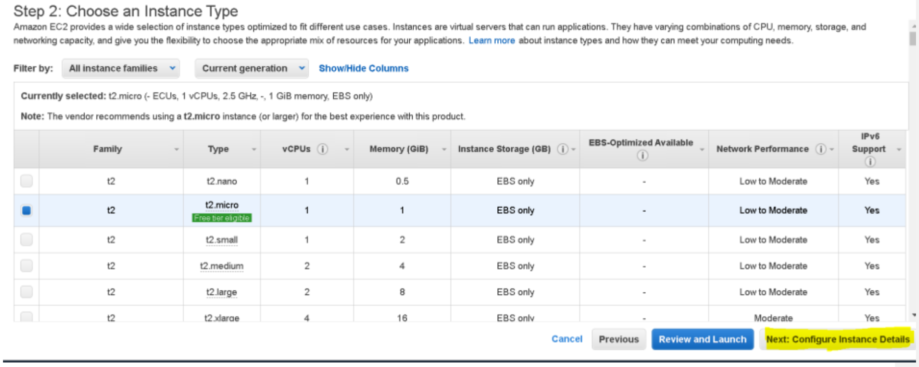
★★★★★ (0) | 7.7_HVM | By Amazon Web Services

Linux/Unix, Red Hat Enterprise Linux 7.7_HVM | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 10/15/19

Red Hat Enterprise Linux 7.7 (HVM)

More info

2. Keep clicking "Next: Configure Instance Details":



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

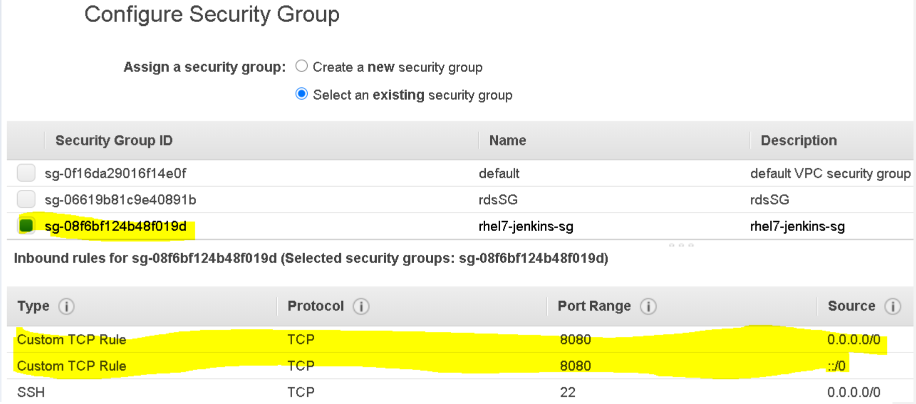
Filter by: All instance families | Current generation | Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

Note: The vendor recommends using a **t2.micro** instance (or larger) for the best experience with this product.

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
□	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
□	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
□	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
□	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
□	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes

Note: Make sure port 8080 is allowed on the Security Group inbound, refer screen shot below:



Configure Security Group

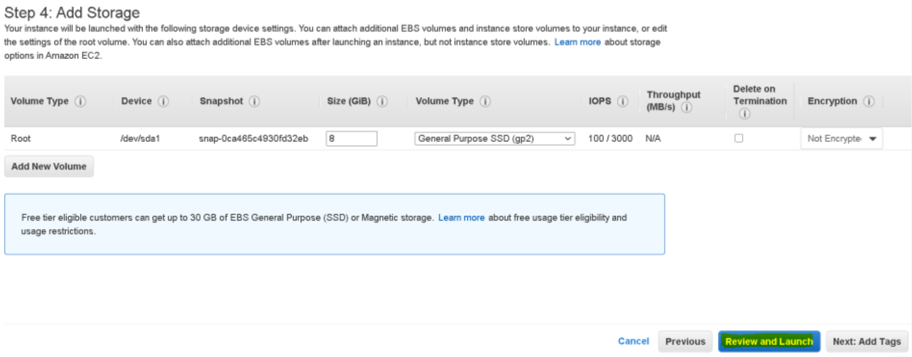
Assign a security group: ○ Create a **new** security group
● Select an **existing** security group

	Security Group ID	Name	Description
□	sg-0f16da29016f14e0f	default	default VPC security group
□	sg-06619b81c9e40891b	rdsSG	rdsSG
■	sg-08f6bf124b48f019d	rhel7-jenkins-sg	rhel7-jenkins-sg

Inbound rules for sg-08f6bf124b48f019d (Selected security groups: sg-08f6bf124b48f019d)

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	8080	0.0.0.0/0
Custom TCP Rule	TCP	8080	::/0
SSH	TCP	22	0.0.0.0/0

3. Then click "Review and Launch" and then finally click "Launch":



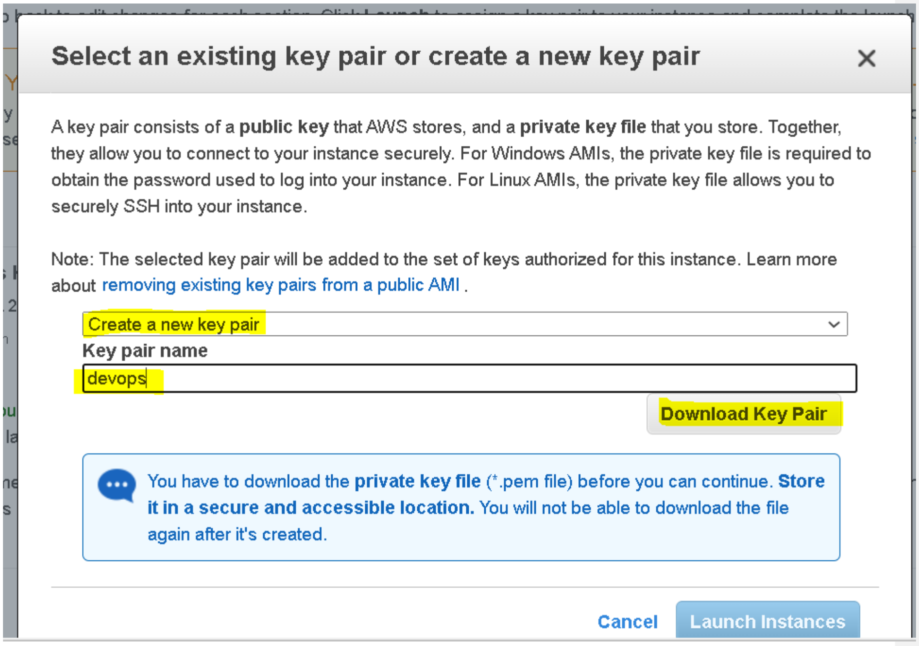
Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0ca465c4930fd32eb	8	General Purpose SSD (gp2)	100 / 3000	N/A	□	Not Encrypte

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

4. Create a New key pair and save the public key in your local system:



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

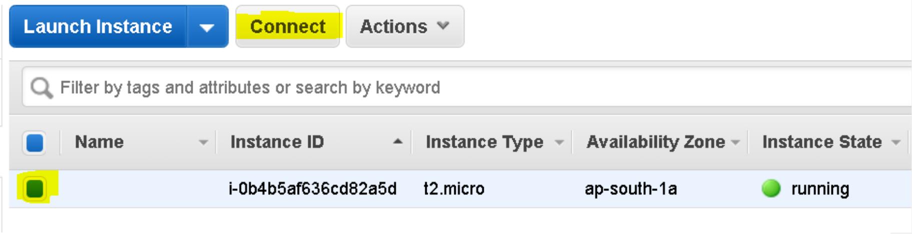
Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .

Create a new key pair

Key pair name: devops

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

5. Then choose the instance and click on connect to SSH into the server:



☐	Name	Instance ID	Instance Type	Availability Zone	Instance State
■		i-0b4b5af636cd82a5d	t2.micro	ap-south-1a	● running

6. After you have logged in to the server, run the following commands in sequence.

#become “root” user

sudo su –

#update all packages on the server

apt-get update –y

#install java

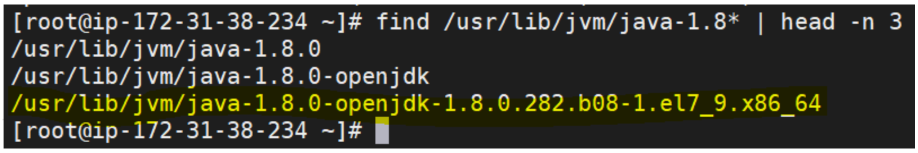
apt-get -y install java-1.8*

#check default java version

java –version

#check the java path to be added to user profile

find /usr/lib/jvm/java-1.8* | head -n 3

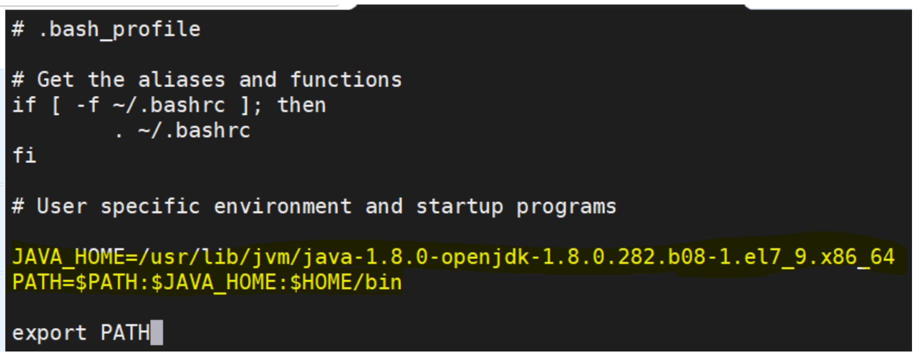


```
[root@ip-172-31-38-234 ~]# find /usr/lib/jvm/java-1.8* | head -n 3
/usr/lib/jvm/java-1.8.0
/usr/lib/jvm/java-1.8.0-openjdk
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.el7_9.x86_64
[root@ip-172-31-38-234 ~]#
```

export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.el7_9.x86_64

#to make the change persistent across reboots

vi .bash_profile



```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.el7_9.x86_64
PATH=$PATH:$JAVA_HOME:$HOME/bin

export PATH
```

#save the .bash_profile with the above changes

source .bash_profile

#verify the updated path

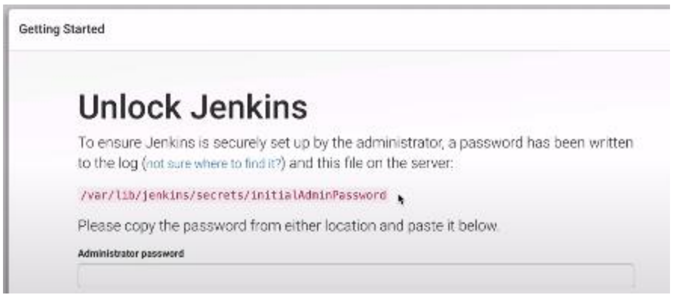
```
echo $PATH
apt-get -y install wget
wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
apt-get -y install jenkins
systemctl enable jenkins
systemctl start Jenkins
```

Now, follow the below steps in sequence to setup Jenkins UI:-

- login to Jenkins UI: http://jenkins-server-public-ip:8080

  Note: Skip installing any plugins.

We need to show the password for the admin user to log in to our Jenkins web interface:



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

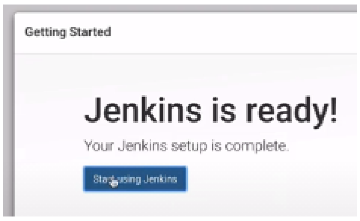
Please copy the password from either location and paste it below.

Administrator password

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

Copy the string that is output and paste it into the *Administrator password* field in your browser. Click **Continue**.

Click **Save and continue**. Next, click **Start using Jenkins**.



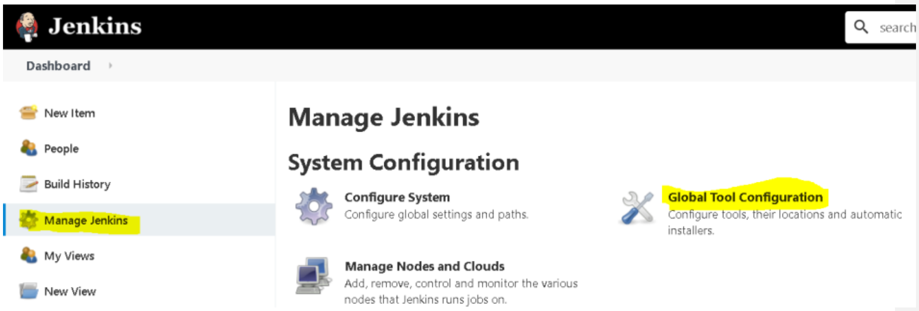
Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

#Java configuration on the Jenkins UI

- Click on “Global Tool Configuration” under “Manage Jenkins”.



Manage Jenkins

System Configuration

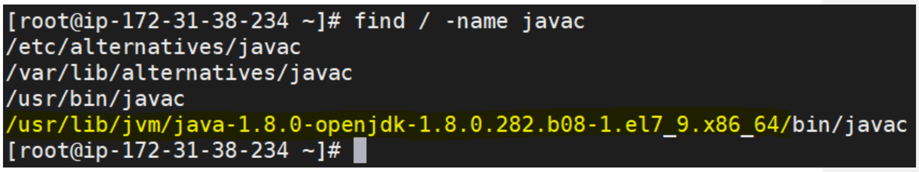
Configure System
Configure global settings and paths.

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.

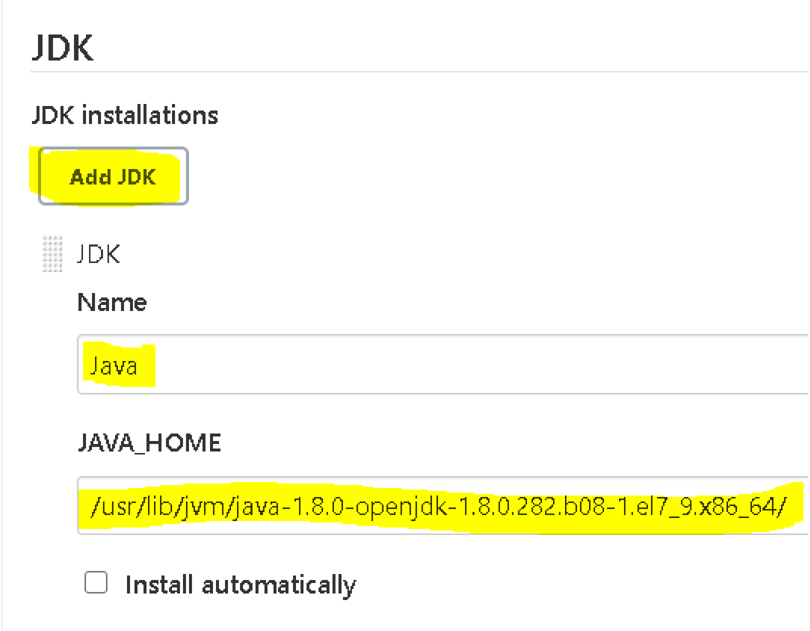
- Click on “Add JDK” and enter the details.
  For JAVA_HOME, run find / -name javac on the CLI

  CLI screen shot:



```
[root@ip-172-31-38-234 ~]# find / -name javac
/etc/alternatives/javac
/var/lib/alternatives/javac
/usr/bin/javac
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.el7_9.x86_64/bin/javac
[root@ip-172-31-38-234 ~]#
```

Jenkins UI screen shot:



JDK

JDK installations

JDK

Name

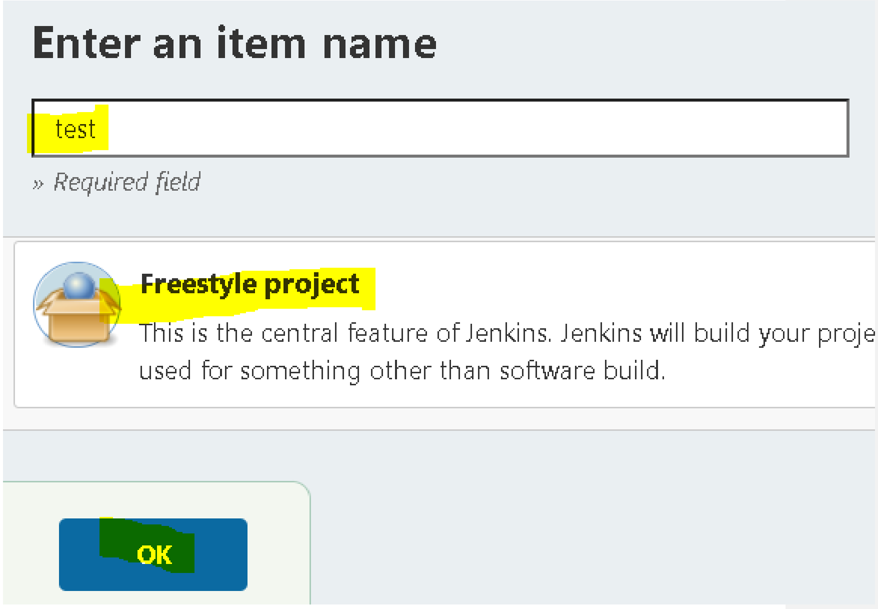
Java

JAVA_HOME

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.el7_9.x86_64/

□ Install automatically

- Click on apply.
- Now, we can test Jenkins functionality.
  a. On the Jenkins UI, click “New Item”, enter a name “test”, choose “Freestyle Project” and click “ok”.



Enter an item name

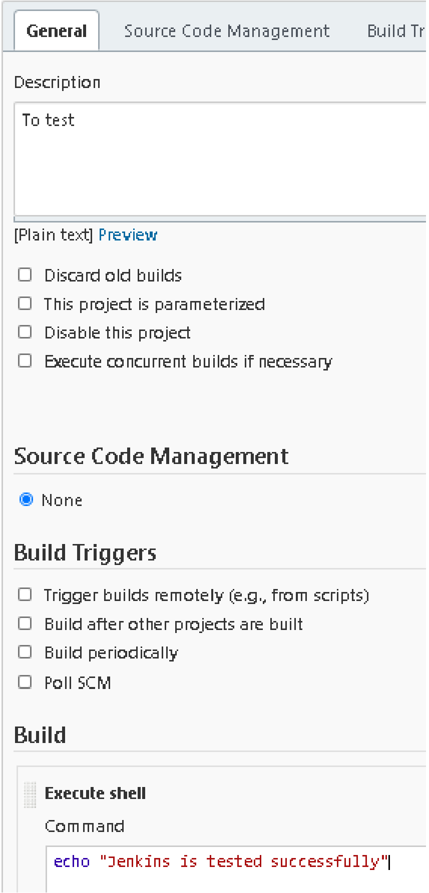
test

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your proje
used for something other than software build.

b. Enter the details as below.



Description

To test

[Plain text] Preview

□ Discard old builds
□ This project is parameterized
□ Disable this project
□ Execute concurrent builds if necessary

Source Code Management

● None

Build Triggers

□ Trigger builds remotely (e.g., from scripts)
□ Build after other projects are built
□ Build periodically
□ Poll SCM

Build

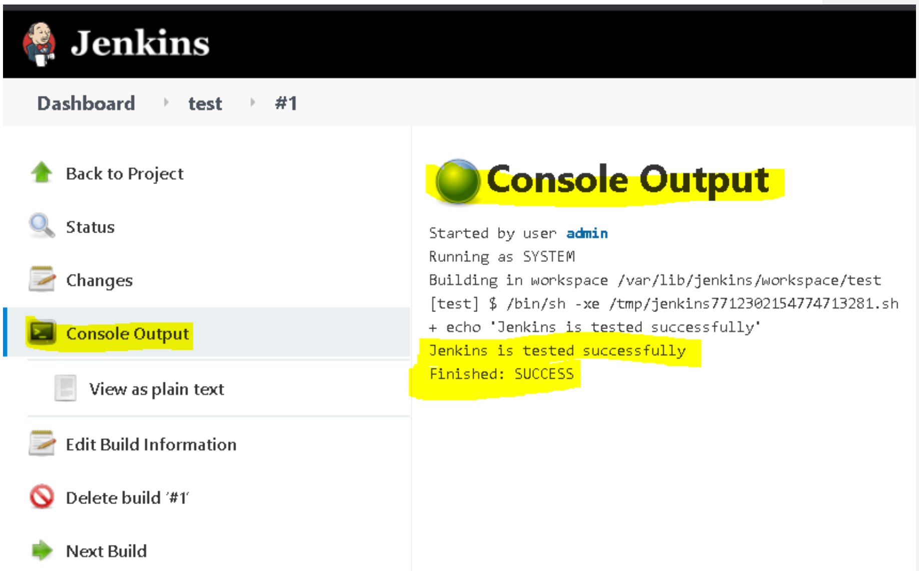
Execute shell

Command

```
echo "Jenkins is tested successfully"
```

c. Now go to home page and click on build to run the test job.

d. Check the console output if the job ran successfully.



```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test
[test] $ /bin/sh -xe /tmp/jenkins7712302154774713281.sh
+ echo 'Jenkins is tested successfully'
Jenkins is tested successfully
Finished: SUCCESS
```

Configure Maven Build Tool on Jenkins server

```
mkdir /opt/maven
```

```
cd /opt/maven
```

```
wget https://mirrors.estointernet.in/apache/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.tar.gz
```

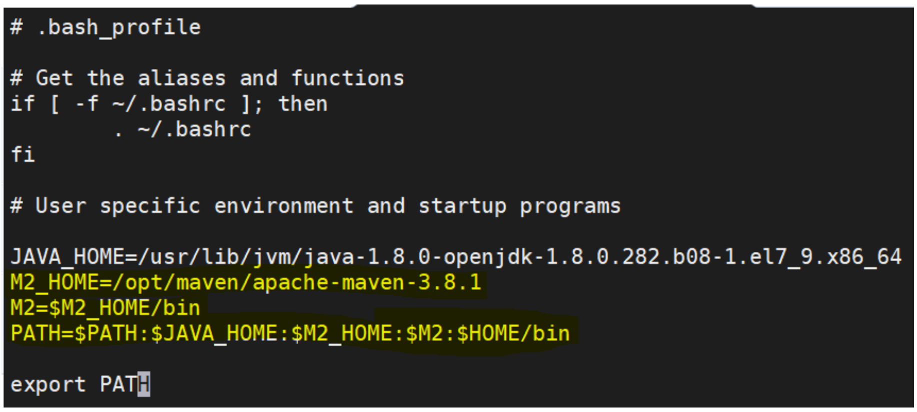
```
tar -xvzf apache-maven-3.8.1-bin.tar.gz
```

```
cd apache-maven-3.8.1
```

```
pwd     #copy the path
```

```
vi /root/.bash_profile
```

#make the following changes and save the file

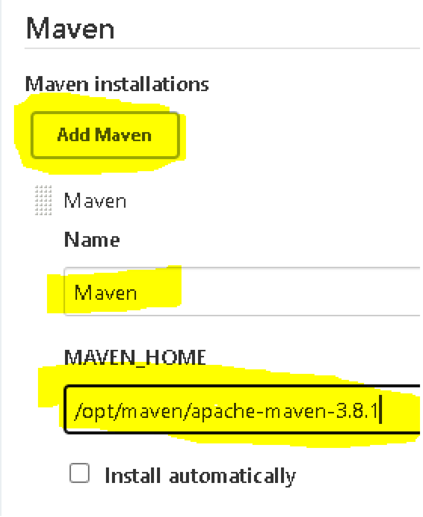


```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.el7_9.x86_64
M2_HOME=/opt/maven/apache-maven-3.8.1
M2=$M2_HOME/bin
PATH=$PATH:$JAVA_HOME:$M2_HOME:$M2:$HOME/bin

export PATH
```

source /root/.bash_profile   *#if this does not apply changes, then logout and log back in*

a. Now go back to the Jenkins console and click on: Manage Jenkins >> Manage Plugins.
b. Under Available plugins, search for “maven invoker”, then select it and choose “Install without restart”.
c. Under Available plugins, search for “maven integration plugin”, then select it and choose “Install without restart”.
d. Under Available plugins again, search for “github”, then select it and choose “Install without restart”.
e. Under Available plugins again, search for “deploy to container”, then select it and choose “Install without restart”.
f. Under Available plugins again, search for “publish over ssh”, then select it and choose “Install without restart”.
g. Now, go to: Manage Jenkins >> Global Tool Configuration, add maven configuration, apply and save:

Commented [C1]: Could not find Github plugin



Maven

Maven installations

Add Maven

Maven

Name

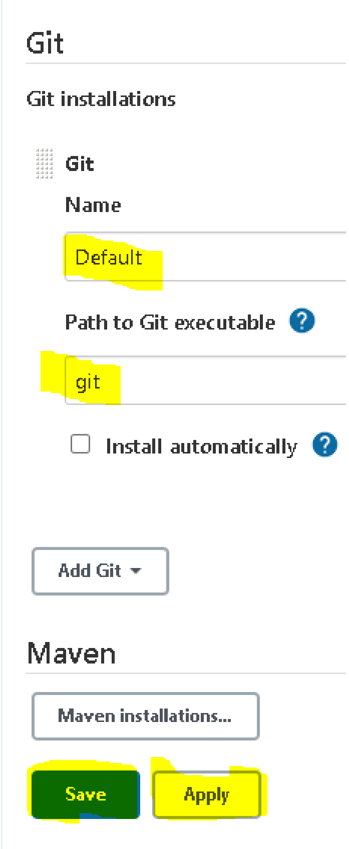
Maven

MAVEN_HOME

/opt/maven/apache-maven-3.8.1

□ Install automatically

h. Now again, go to: Manage Jenkins >> Global Tool Configuration, verify git configuration as per screen shot below, apply and save:



Git

Git installations

Git

Name

Default

Path to Git executable

git

□ Install automatically

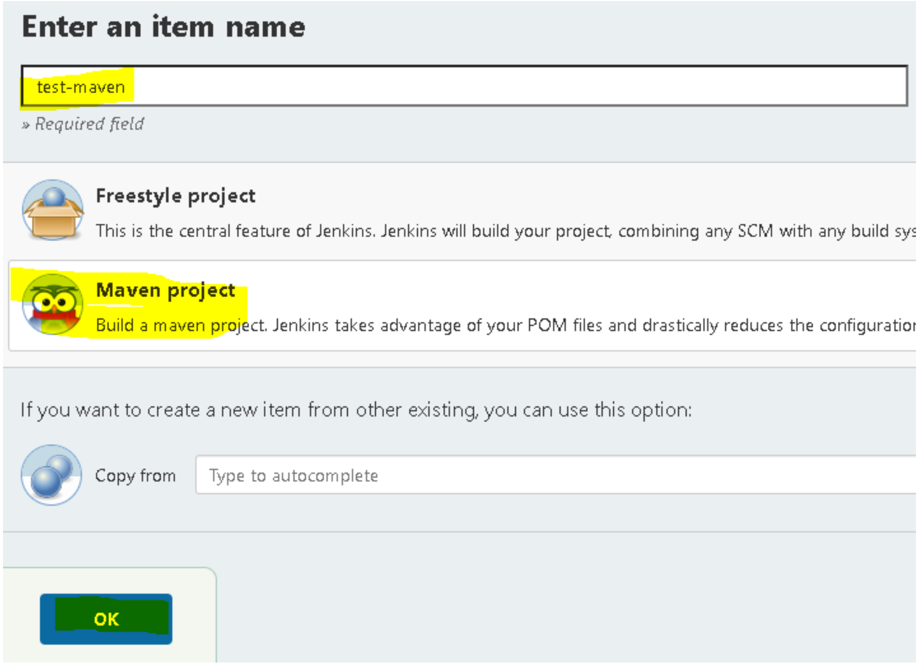
Configure git on Jenkins server

#On the CLI, run these commands

apt-get -y install git

Test your new Jenkins, Git, and Maven configuration

Click “new item” and follow the steps as per screen shots below:



Enter an item name

test-maven

» Required field

Freestyle project

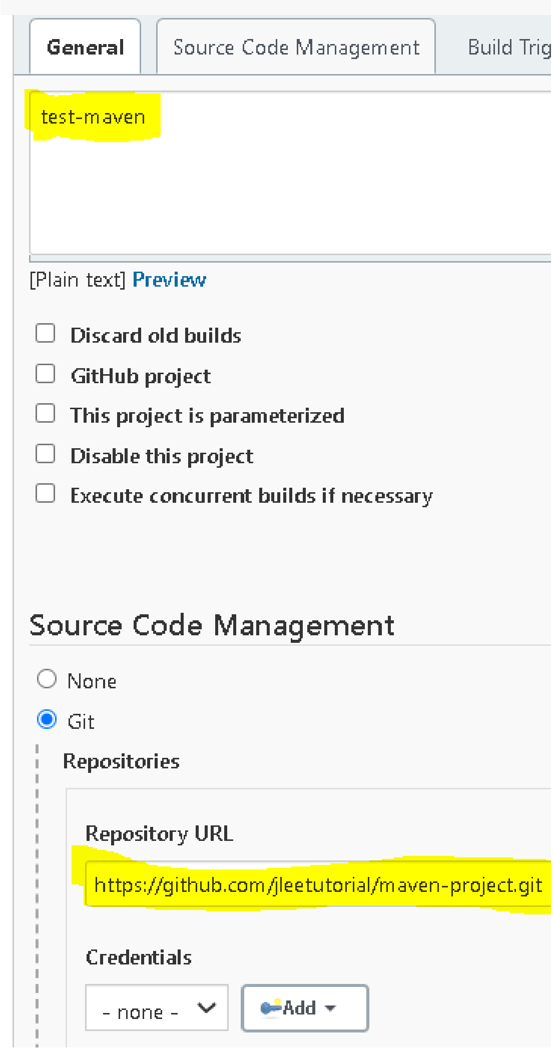
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build sys

Maven project

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

If you want to create a new item from other existing, you can use this option:

Copy from



test-maven

[Plain text] Preview

□ Discard old builds

□ GitHub project

□ This project is parameterized

□ Disable this project

□ Execute concurrent builds if necessary

Source Code Management

○ None

● Git

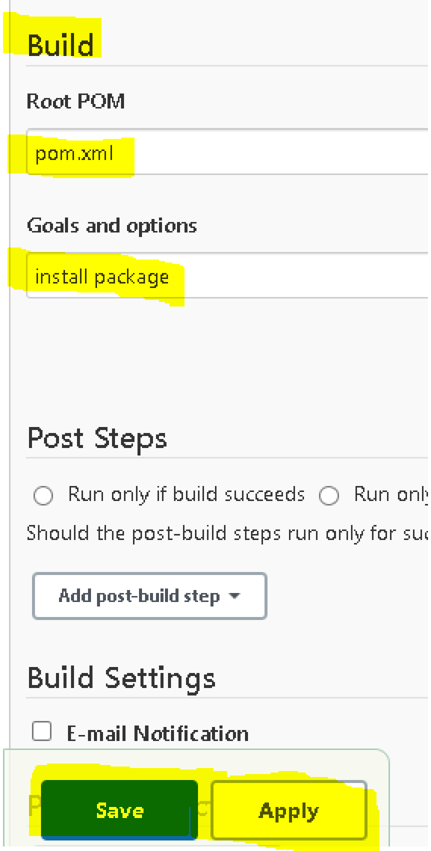
Repositories

Repository URL

https://github.com/jleetutorial/maven-project.git

Credentials

- none -



Build

Root POM

pom.xml

Goals and options

install package

Post Steps

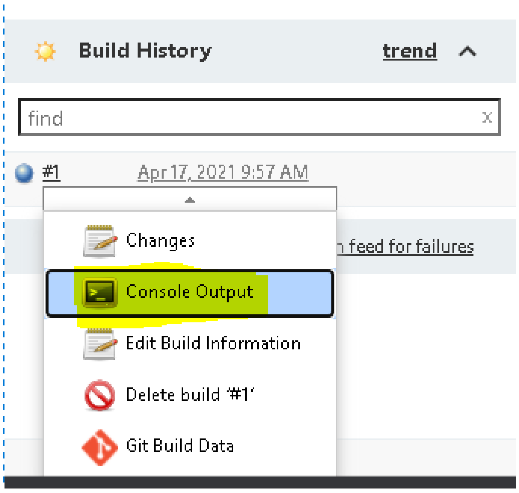
○ Run only if build succeeds ○ Run onl

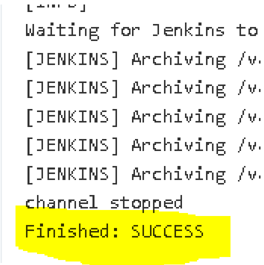
Should the post-build steps run only for suc

Build Settings

□ E-mail Notification

Now, click on your project and go to “Console Output” and check for SUCCESS message at the end of the output.

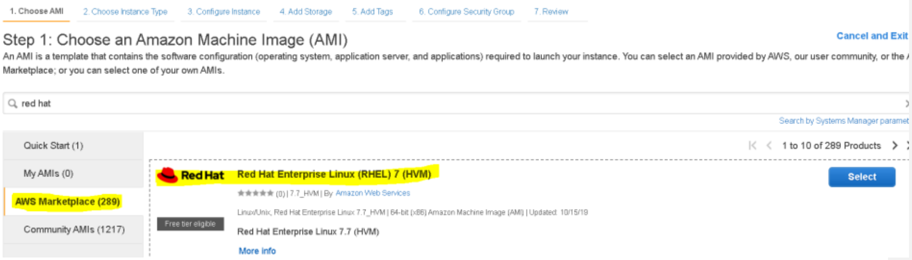




```
Waiting for Jenkins to
[JENKINS] Archiving /v
[JENKINS] Archiving /v
[JENKINS] Archiving /v
[JENKINS] Archiving /v
[JENKINS] Archiving /v
channel stopped
Finished: SUCCESS
```

Configure webserver

1. On the AWS Management Console, click launch instance, and choose **Red Hat Enterprise Linux (RHEL) 7 (HVM)** AMI:



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the Marketplace; or you can select one of your own AMIs.

red hat

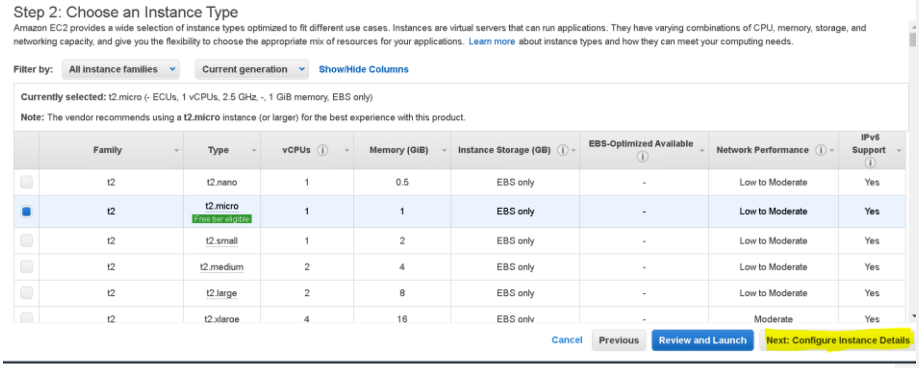
Red Hat Enterprise Linux (RHEL) 7 (HVM)

★★★★★ (0) | 7.7_HVM | By Amazon Web Services

Linux/Unix, Red Hat Enterprise Linux 7.7_HVM | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 10/15/19

Red Hat Enterprise Linux 7.7 (HVM)

2. Keep clicking “Next: Configure Instance Details”:



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

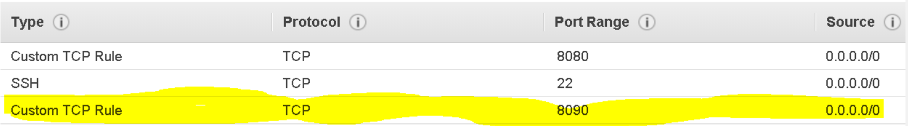
Filter by: All instance families | Current generation | Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

Note: The vendor recommends using a **t2.micro** instance (or larger) for the best experience with this product.

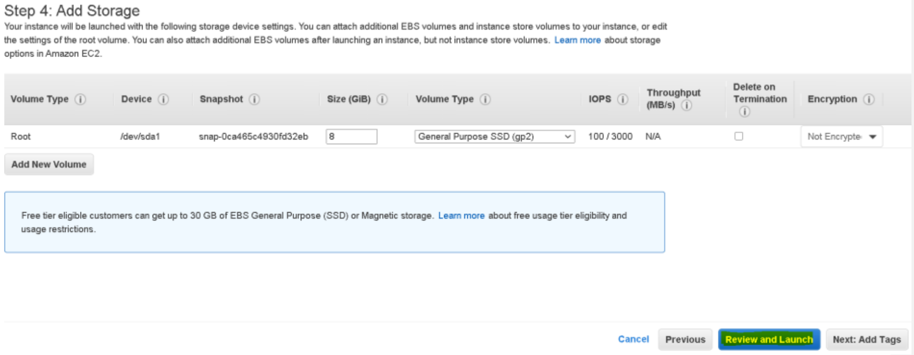
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
□	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
□	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
□	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
□	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
□	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes

Note: Make sure port 8090 is allowed on the Security Group inbound, refer screen shot below:



Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	8080	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0
Custom TCP Rule	TCP	8090	0.0.0.0/0

3. Then click “Review and Launch” and then finally click “Launch”:



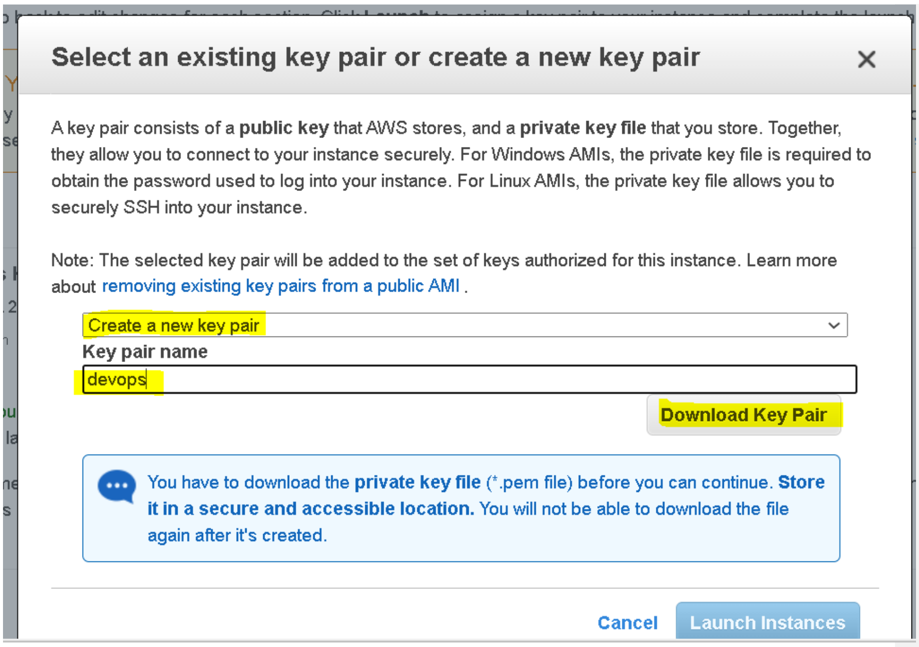
Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0ca465c4930fd32eb	8	General Purpose SSD (gp2)	100 / 3000	N/A	□	Not Encrypte

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

4. Create a New key pair and save the public key in your local system:



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .

Create a new key pair

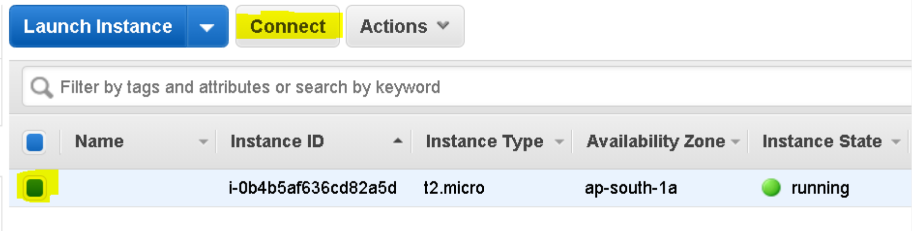
Key pair name

devops

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

5. Then choose the instance and click on connect to SSH into the webserver:



☐	Name	Instance ID	Instance Type	Availability Zone	Instance State
■		i-0b4b5af636cd82a5d	t2.micro	ap-south-1a	● running

6. After you have logged in to the server, run the following commands in sequence.

#become "root" user

sudo su –

#update all packages on the server

apt-get update -y

#install java

apt-get -y install java-1.8*

#check default java version

java –version

cd /opt

apt-get -y install wget

wget https://mirrors.estointernet.in/apache/tomcat/tomcat-8/v8.5.65/bin/apache-tomcat-8.5.65.tar.gz

tar -xvzf apache-tomcat-8.5.65.tar.gz

cd apache-tomcat-8.5.65

cd bin

chmod +x startup.sh

chmod +x shutdown.sh

echo $PATH *#to copy the command directory*

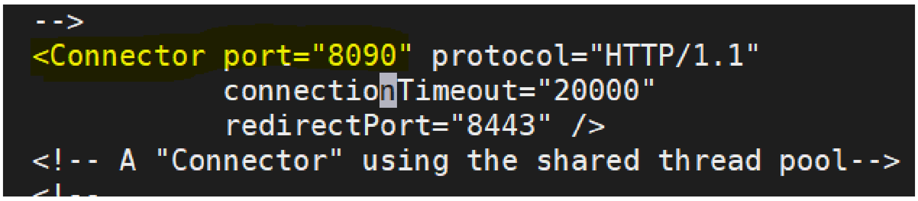
ln -s /opt/apache-tomcat-8.5.65/bin/startup.sh /usr/local/bin/tomcatup

ln -s /opt/apache-tomcat-8.5.65/bin/shutdown.sh /usr/local/bin/tomcatdown

tomcatup

ps -ef | grep -i tomcat

vi /opt/apache-tomcat-8.5.65/conf/server.xml *#search for “Connector port” and change it to 8090*



```
  -->
  <Connector port="8090" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8443" />
  <!-- A "Connector" using the shared thread pool-->
```

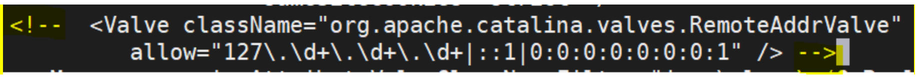
tomcatdown

tomcatup

#tomcat server is now accessible on: <public-ip-of-server:8090>

vi /opt/apache-tomcat-8.5.65/webapps/host-manager/META-INF/context.xml

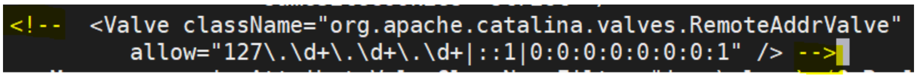
#<!-- and --> is used to comment lines in this file



```
<!--  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
```

vi /opt/apache-tomcat-8.5.65/webapps/manager/META-INF/context.xml

#<!-- and --> is used to comment lines in this file



```
<!--  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
```

tomcatdown

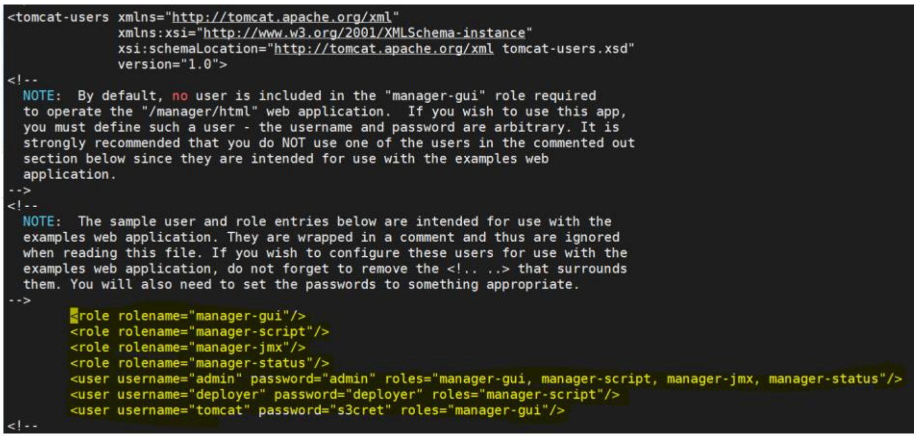
tomcatup

#we need to add users and roles to login to tomcat server on the browser

vi /opt/apache-tomcat-8.5.65/conf/tomcat-users.xml

```
        <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"/>
```

Commented [C2]: Why adding 3 users

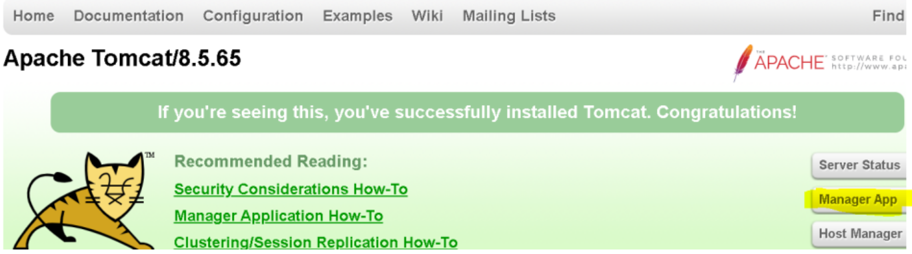


```
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary. It is
  strongly recommended that you do NOT use one of the users in the commented out
  section below since they are intended for use with the examples web
  application.
-->
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  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"/>
<!--
```

tomcatdown

tomcatup

#now, browse to <public-ip-of-web-server:8090> and click on “Manager App”

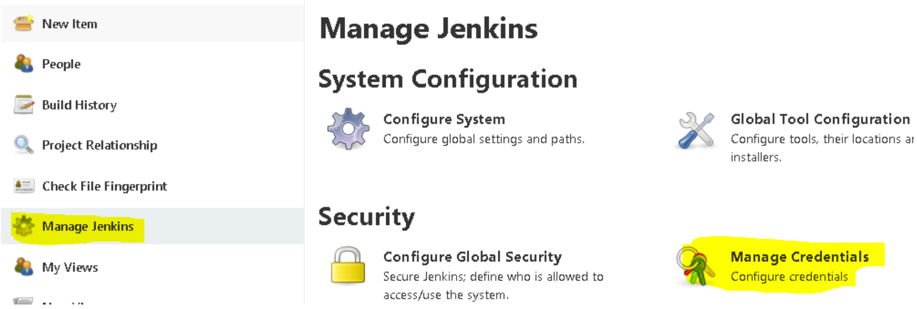

Home
Documentation
Configuration
Examples
Wiki
Mailing Lists
Find
Apache Tomcat/8.5.65
APACHE
If you're seeing this, you've successfully installed Tomcat. Congratulations!
Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To
Server Status
Manager App
Host Manager

#use the below ID and password to login:

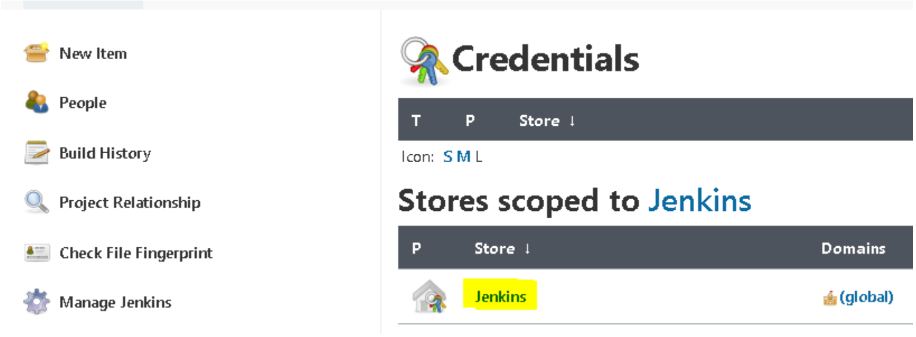
#username: tomcat

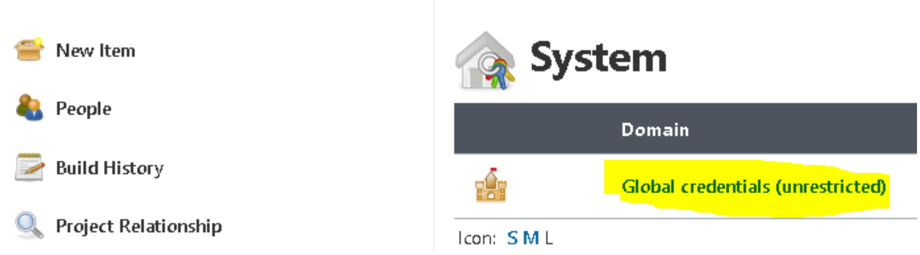
#password: s3cret

Now on the Jenkins UI, go to: Manage Jenkins > Manage Credentials.



Click on Jenkins > Global Credentials > Add Credentials as shown below:

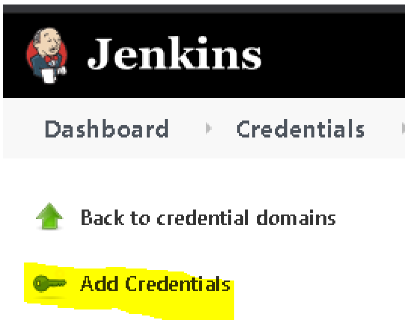




System

Domain
Global credentials (unrestricted)

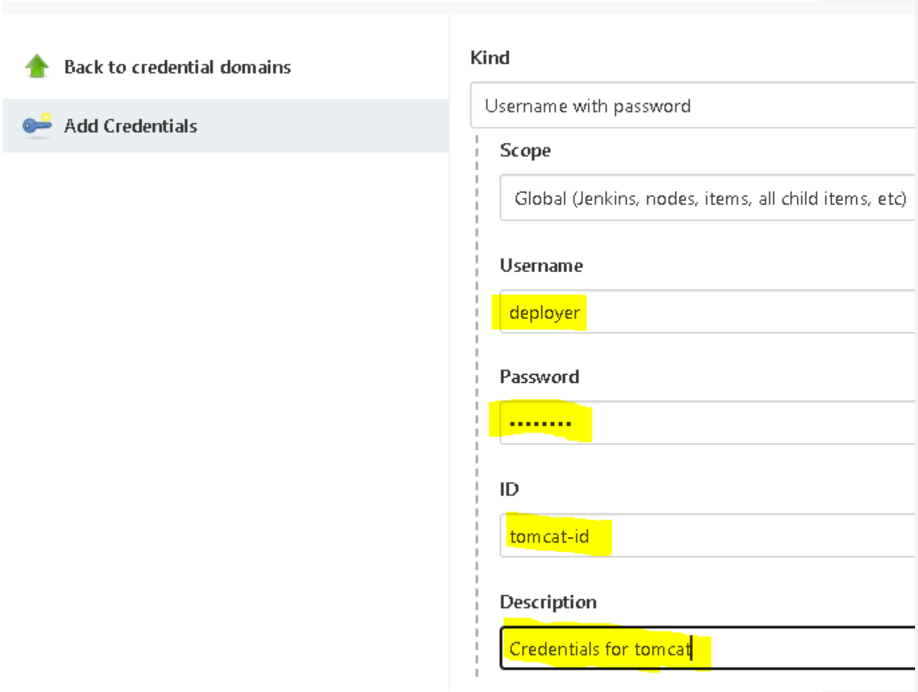
Icon: S M L



Back to credential domains

Add Credentials

Now, under add credentials, add “deployer” as the username and password.



Kind: Username with password

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

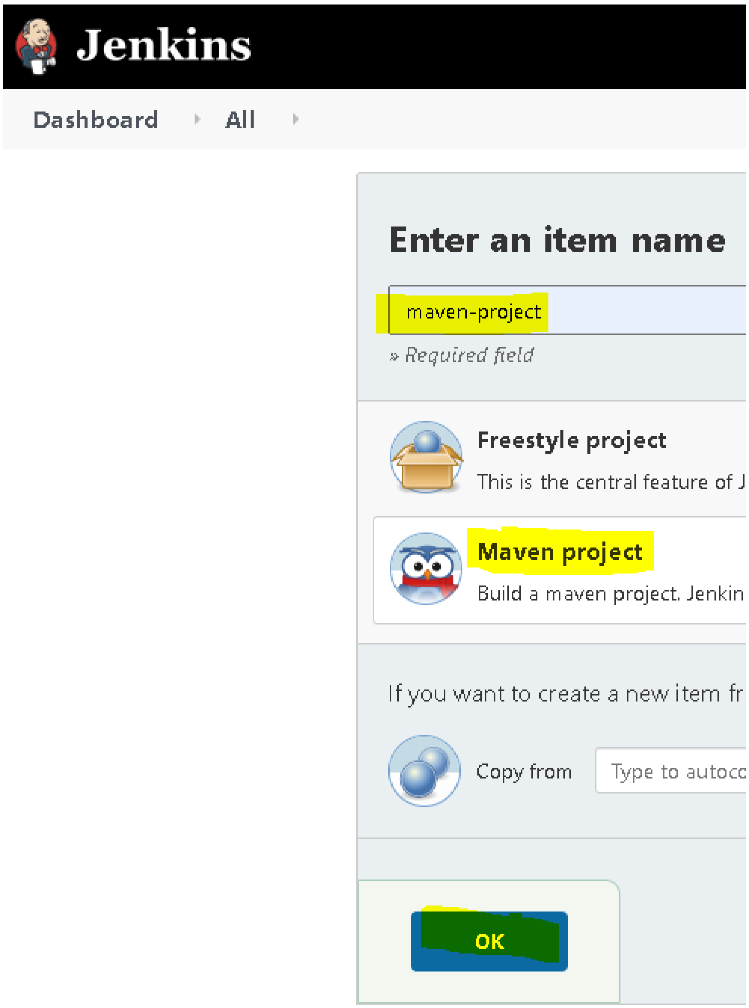
Username: deployer

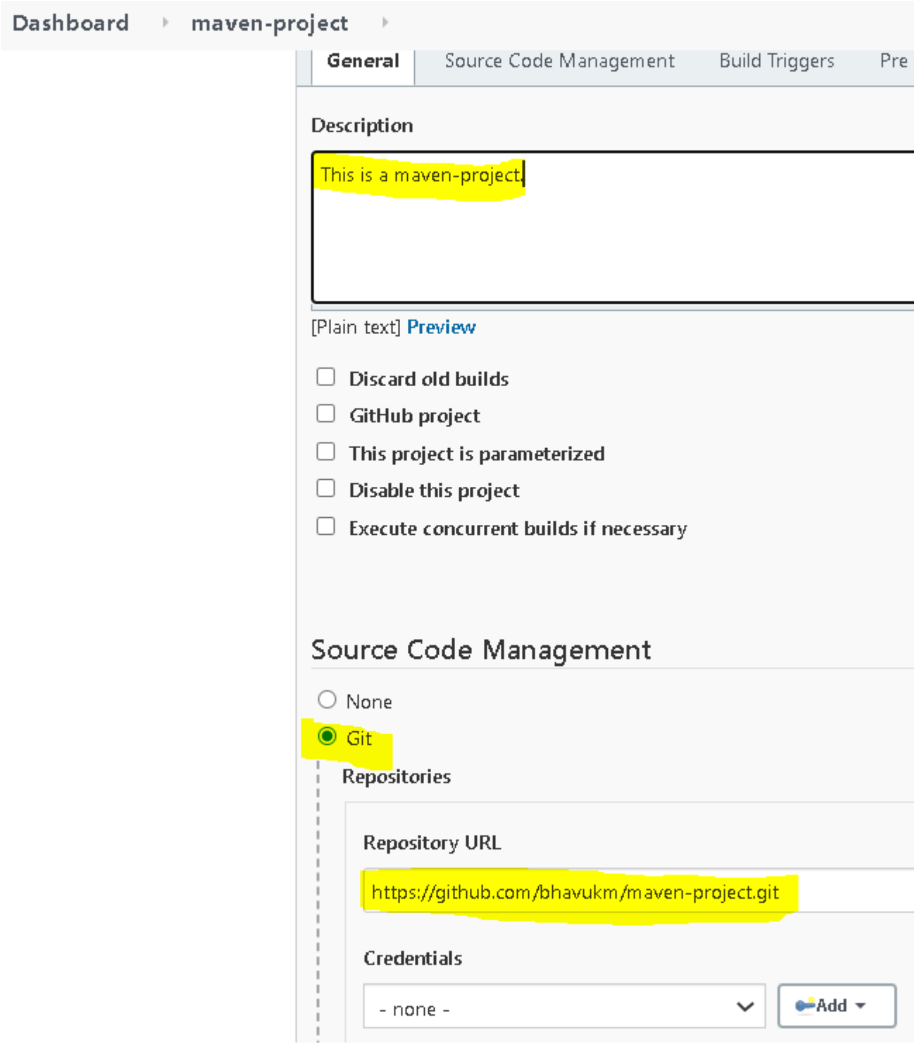
Password: ••••••••

ID: tomcat-id

Description: Credentials for tomcat

Now, create a “New Item” called “maven-project” to test the configuration.





Description

This is a maven-project.

[Plain text] Preview

□ Discard old builds

□ GitHub project

□ This project is parameterized

□ Disable this project

□ Execute concurrent builds if necessary

Source Code Management

○ None

● Git

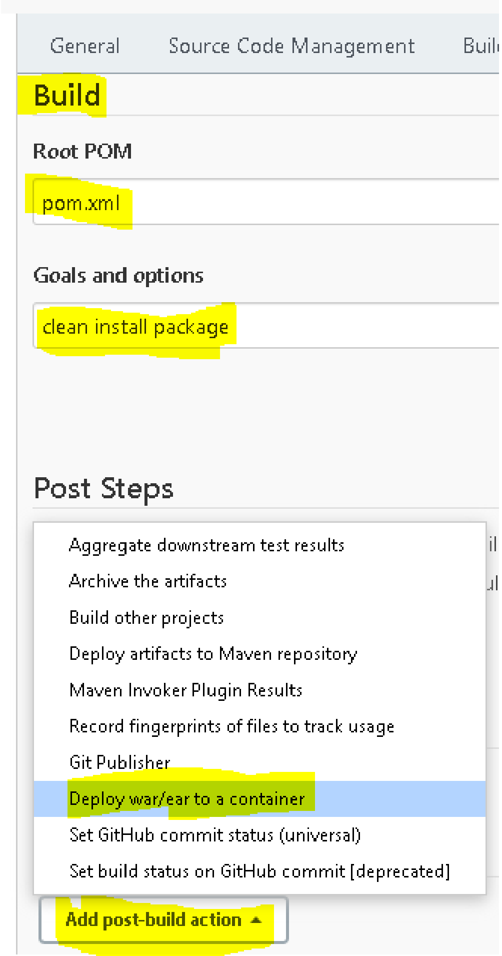
Repositories

Repository URL

https://github.com/bhavukm/maven-project.git

Credentials

- none -



General Source Code Management Buil

Build

Root POM

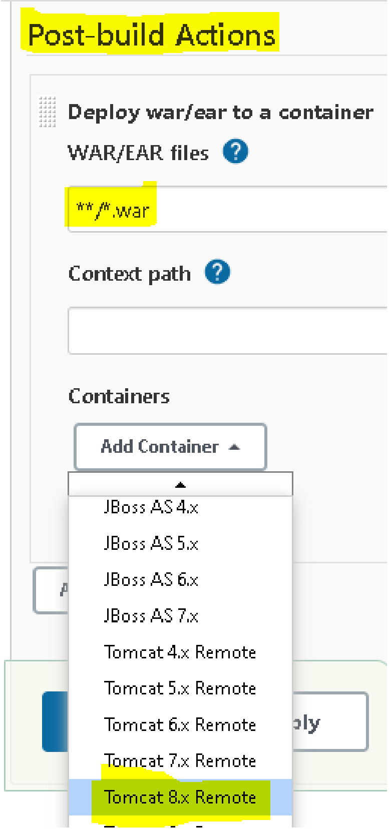
pom.xml

Goals and options

clean install package

Post Steps

Aggregate downstream test results
Archive the artifacts
Build other projects
Deploy artifacts to Maven repository
Maven Invoker Plugin Results
Record fingerprints of files to track usage
Git Publisher
Deploy war/ear to a container
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Add post-build action



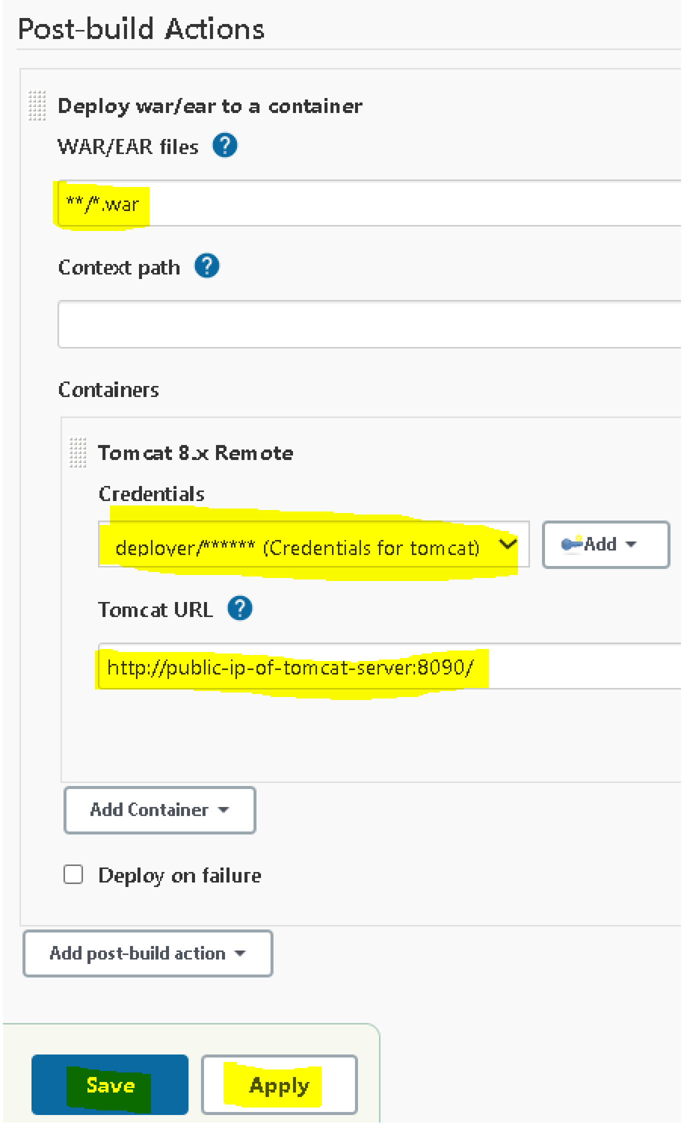
Post-build Actions

Deploy war/ear to a container

WAR/EAR files: **/*.war

Context path:

Containers



Post-build Actions

Deploy war/ear to a container

WAR/EAR files: **/*.war

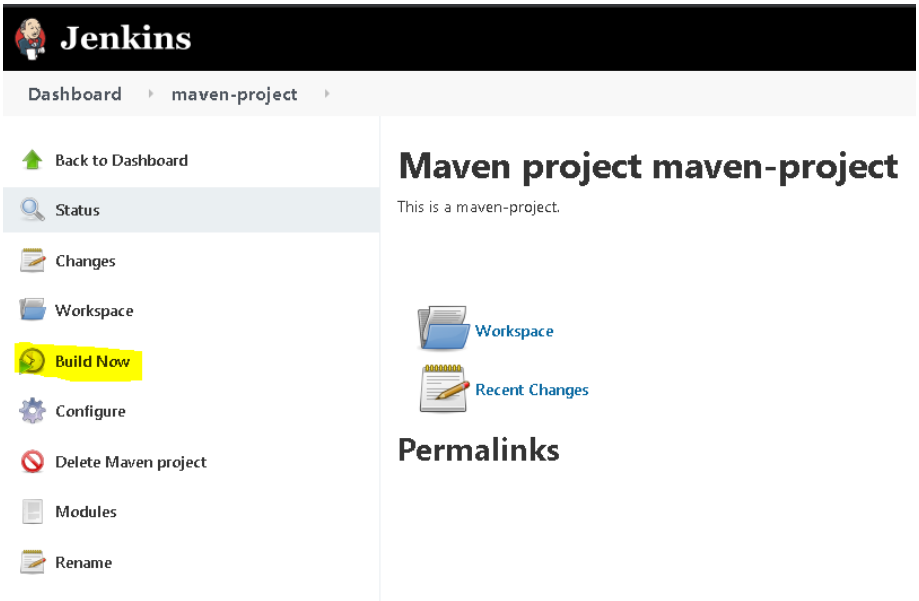
Context path:

Containers

Tomcat 8.x Remote

Credentials: deplover/****** (Credentials for tomcat)

Tomcat URL: http://public-ip-of-tomcat-server:8090/

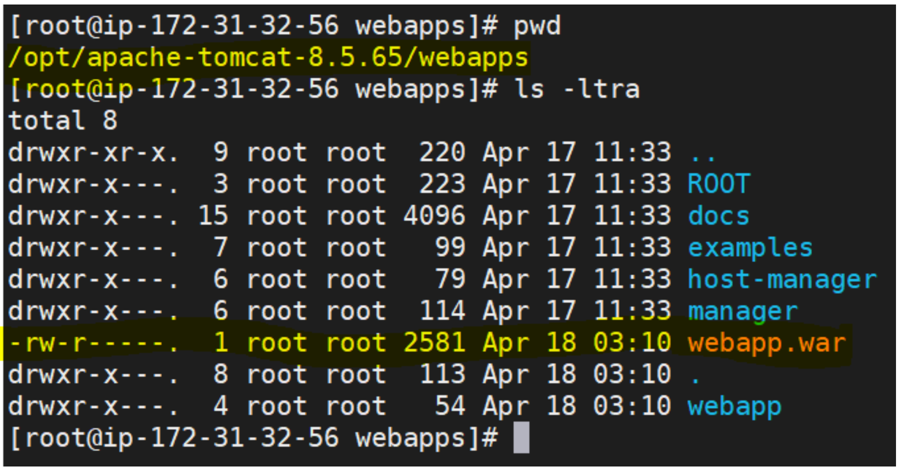


Maven project maven-project

This is a maven-project.

Permalinks

On the Tomcat server CLI, you can find the compiled "webapp.war" file.

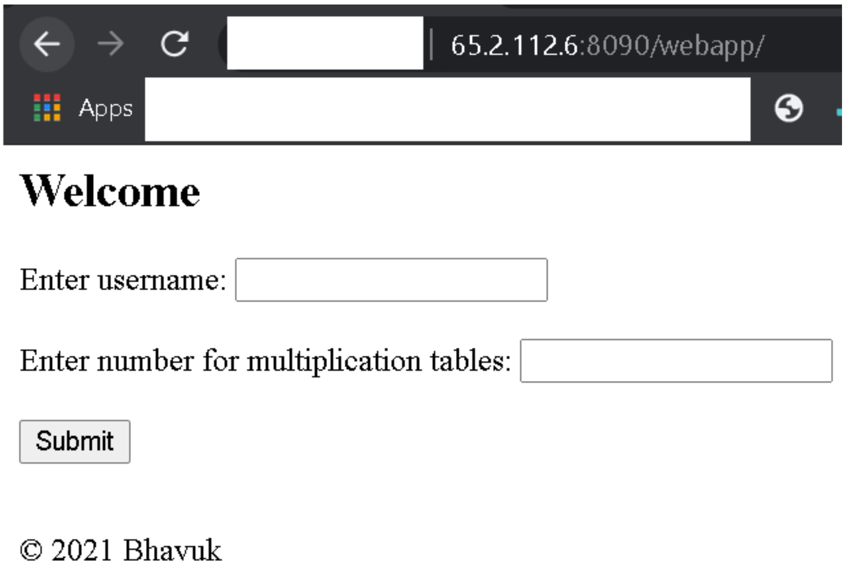


```
[root@ip-172-31-32-56 webapps]# pwd
/opt/apache-tomcat-8.5.65/webapps
[root@ip-172-31-32-56 webapps]# ls -ltra
total 8
drwxr-xr-x.  9 root root  220 Apr 17 11:33 ..
drwxr-x---.  3 root root  223 Apr 17 11:33 ROOT
drwxr-x---. 15 root root 4096 Apr 17 11:33 docs
drwxr-x---.  7 root root   99 Apr 17 11:33 examples
drwxr-x---.  6 root root   79 Apr 17 11:33 host-manager
drwxr-x---.  6 root root  114 Apr 17 11:33 manager
-rw-r-----.  1 root root 2581 Apr 18 03:10 webapp.war
drwxr-x---.  8 root root  113 Apr 18 03:10 .
drwxr-x---.  4 root root   54 Apr 18 03:10 webapp
[root@ip-172-31-32-56 webapps]# 
```

To access the web application, use the following URL on your browser:

http://public-ip-of-tomcat-server:8090/webapp

Reference screen shot:



Welcome

Enter username:

Enter number for multiplication tables:

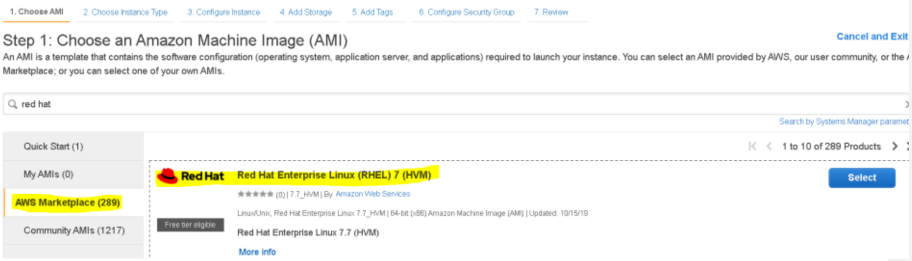
Submit

© 2021 Bhavuk

Configuring Ansible servers (Master and Slave)

Ansible-master configuration:

1. On the AWS Management Console, click launch instance, and choose **Red Hat Enterprise Linux (RHEL) 7 (HVM)** AMI:



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the Marketplace; or you can select one of your own AMIs.

red hat

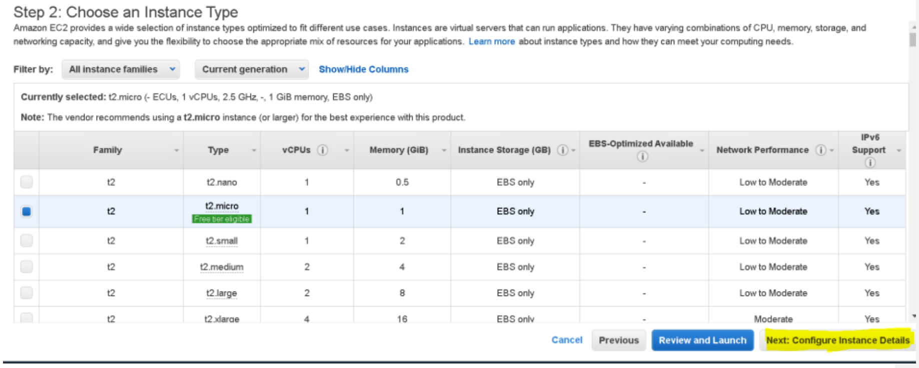
Red Hat Enterprise Linux (RHEL) 7 (HVM)

★★★★★ (0) | 7.7_HVM | By Amazon Web Services

Linux/Unix, Red Hat Enterprise Linux 7.7_HVM | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 10/15/19

Red Hat Enterprise Linux 7.7 (HVM)

2. Keep clicking “Next: Configure Instance Details”:



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

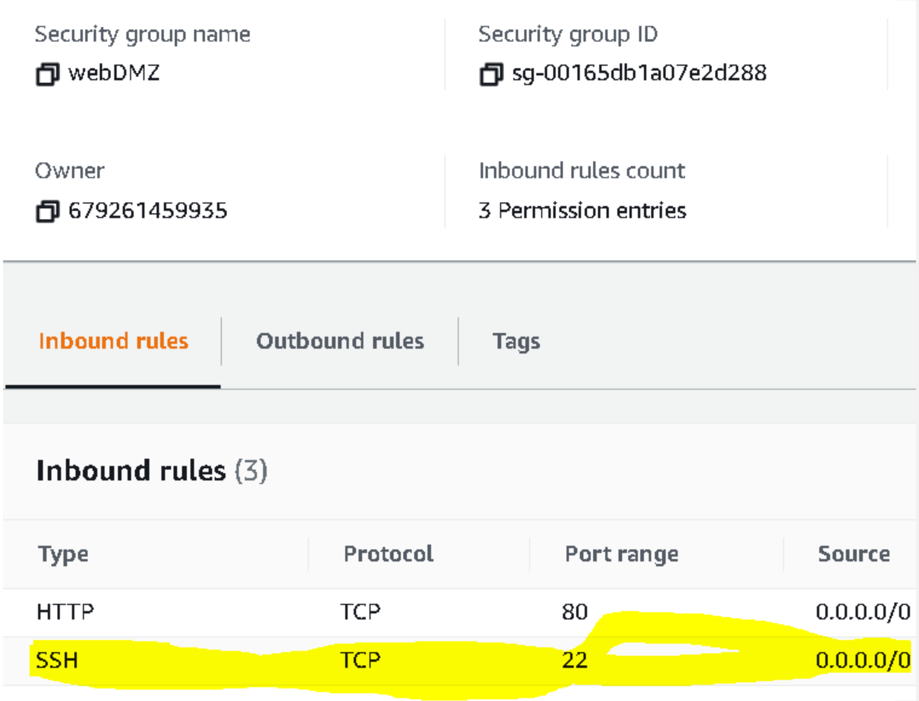
Filter by: All instance families | Current generation | Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

Note: The vendor recommends using a **t2.micro** instance (or larger) for the best experience with this product.

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
□	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
□	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
□	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
□	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
□	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes

Note: Make sure port 22 is allowed on the Security Group inbound, refer screen shot below:



Security group name
webDMZ

Security group ID
sg-00165db1a07e2d288

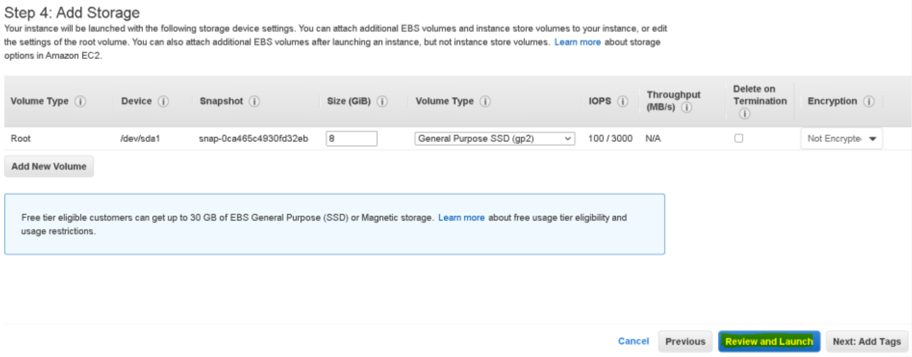
Owner
679261459935

Inbound rules count
3 Permission entries

Inbound rules (3)

Type	Protocol	Port range	Source
HTTP	TCP	80	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0

3. Then click “Review and Launch” and then finally click “Launch”:



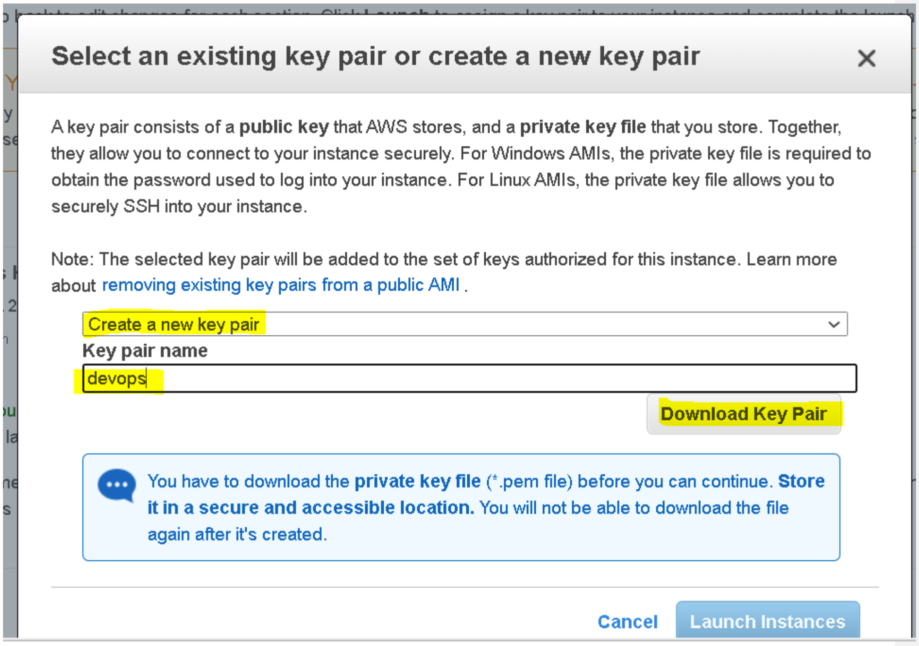
Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0ca465c4930fd32eb	8	General Purpose SSD (gp2)	100 / 3000	N/A	□	Not Encrypte

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

4. Create a New key pair and save the public key in your local system:



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

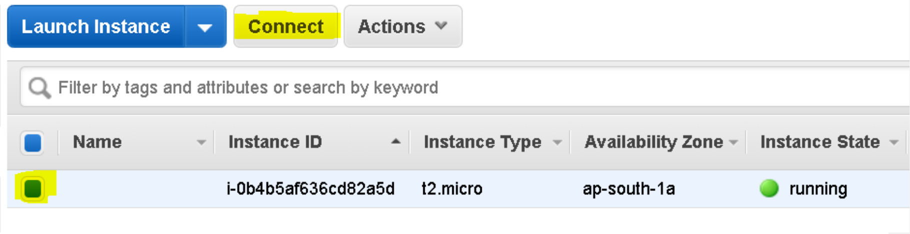
Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .

Create a new key pair

Key pair name: devops

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

5. Then choose the instance and click on connect to SSH into the server:



□	Name	Instance ID	Instance Type	Availability Zone	Instance State
☑		i-0b4b5af636cd82a5d	t2.micro	ap-south-1a	● running

6. After you have logged in to the servers, run the following commands in sequence.

#On Ansible-Master and Ansible-Slave (we are taking Tomcat-Webserver as Ansible-Slave)

#become “root” user

sudo su -

#update all packages on the server

apt-get update -y

#On Ansible-Master

apt install rpm

rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

apt-get install ansible -y

ansible --version

#On Ansible-Master and Ansible-Slave

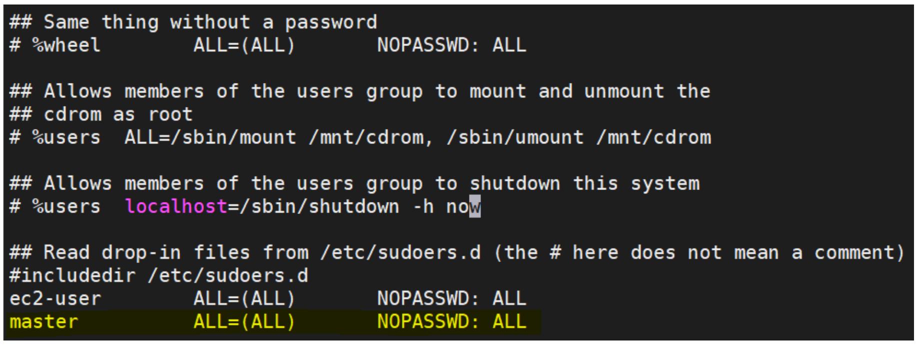
useradd master

passwd master

visudo

#scroll to the end of the file (shift + G) and type:

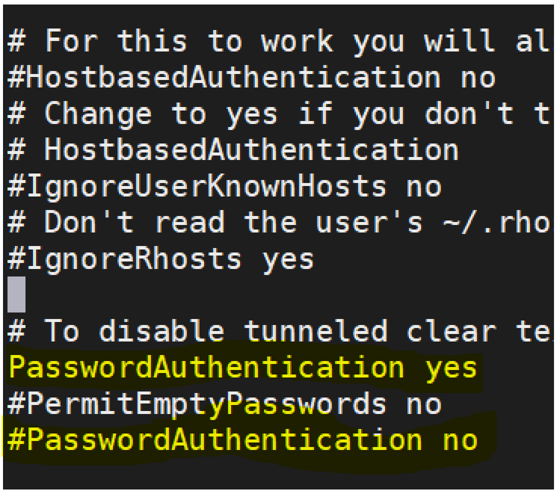
master ALL=(ALL) NOPASSWD: ALL



```
## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users  localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
ec2-user        ALL=(ALL)       NOPASSWD: ALL
master          ALL=(ALL)       NOPASSWD: ALL
```

#enable Password Authentication

vi /etc/ssh/sshd_config



```
# For this to work you will al
#HostbasedAuthentication no
# Change to yes if you don't t
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rho
#IgnoreRhosts yes

# To disable tunneled clear te
PasswordAuthentication yes
#PermitEmptyPasswords no
#PasswordAuthentication no
```

systemctl restart sshd

#now, login as user "master" on Ansible-server

su - master

ssh-keygen *#keep pressing enter until the prompt ($) comes back*

ssh-copy-id <private-ip-address-of-Ansible-slave>

ssh <private-ip-address-of-Ansible-slave> *#to test password-less authentication after copying the keys*

#to add any slave machines as host on Ansible-server, login as "root" on Ansible-server, then run the following commands

chown master:master /etc/ansible/hosts

vi /etc/Ansible/hosts

#delete all the lines using "dd" and then enter slave-machine's private ip, save and quit

#to test if Ansible-server is able to ping Ansible-slave

ansible all -m ping

#make sure you are still logged in as "root" user on Ansible-server

mkdir /opt/playbooks

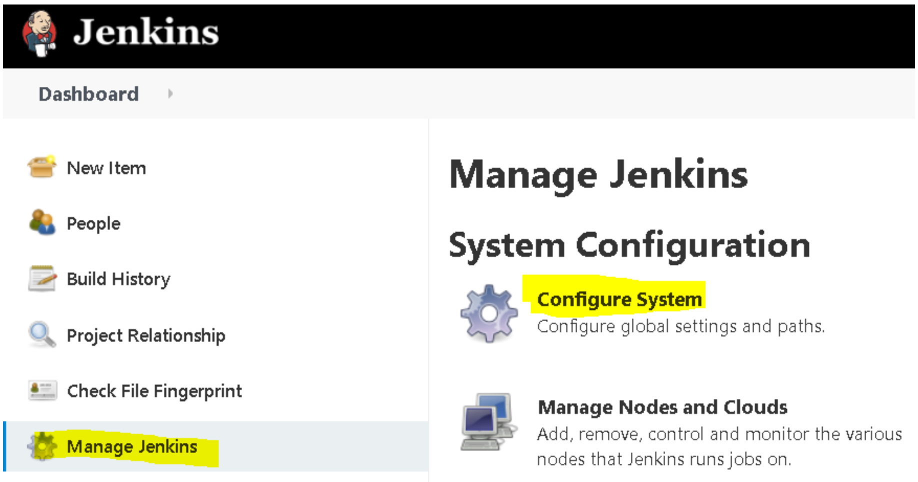
chown -R master:master /opt/playbooks

#login as "master" user

```
su - master
vi /opt/playbooks/copyfile.yml
---
- hosts: all
  become: true
  tasks:
    - name: copy war file
        copy:
          src: /opt/playbooks/webapp/target/webapp.war
          dest: /opt/apache-tomcat-8.5.65/webapp

vi /opt/playbooks/debian.yaml
- name: Install curl package
  ansible.builtin.apt:
    name: "curl"
    state: present
vi /opt/playbooks/redhat.yaml
- name: Install curl package
  ansible.builtin.yum:
    name: "curl"
    state: present
vi /opt/playbooks/ansible-role.yaml
---
- hosts: all
  become: true
  tasks:
    - name: Install "curl" to test the website from CLI on Redhat
      import_tasks: redhat.yaml
      when: ansible_facts['os_family']|lower == 'redhat'

    - name: Install "curl" to test the website from CLI on Debian
      import_tasks: debian.yaml
    when: ansible_facts['os_family']|lower == 'debian'
```

#Configure final Infrastructure for CI (Continuous Integration)

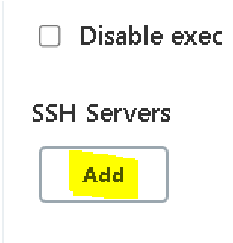
On the Jenkins UI, click on: Manage Jenkins > Configure System.

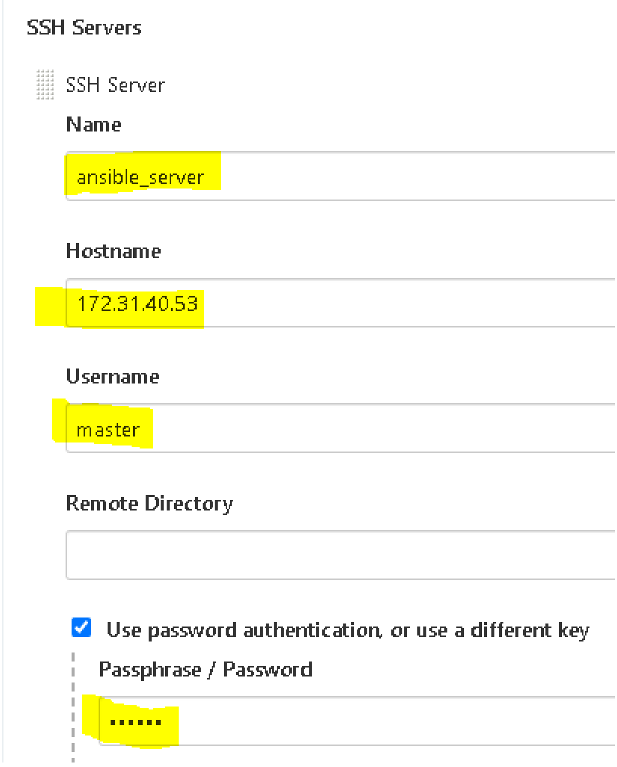


scroll to the end until "SSH Servers" section, add servers (Ansible and Tomcat) using the following details:

Name: Ansible_server ; Hostname: private-ip-address-of-Ansible-Server ; Username: master ; Password: password-you-set-for-master-user

Name: Tomcat ; Hostname: private-ip-address-of-Tomcat-Server ; Username: master ; Password: password-you-set-for-master-user





SSH Servers

SSH Server

Name

ansible_server

Hostname

172.31.40.53

Username

master

Remote Directory

☑ Use password authentication, or use a different key

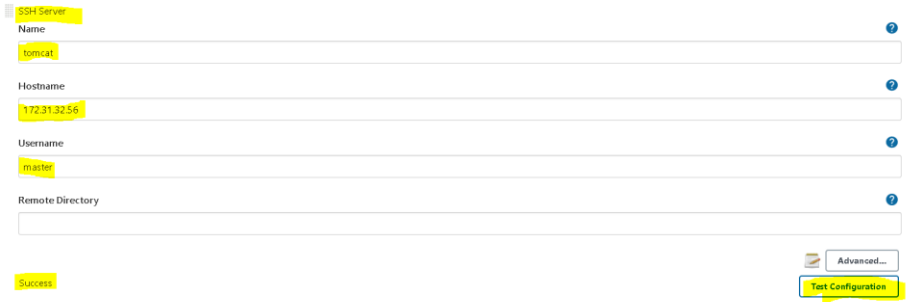
Passphrase / Password

••••••

Now, click on “Test Configuration” and look for “Success” message. Then click on “Apply” and “Save”.

Success

Test Configuration



SSH Server

Name

tomcat

Hostname

172.31.32.56

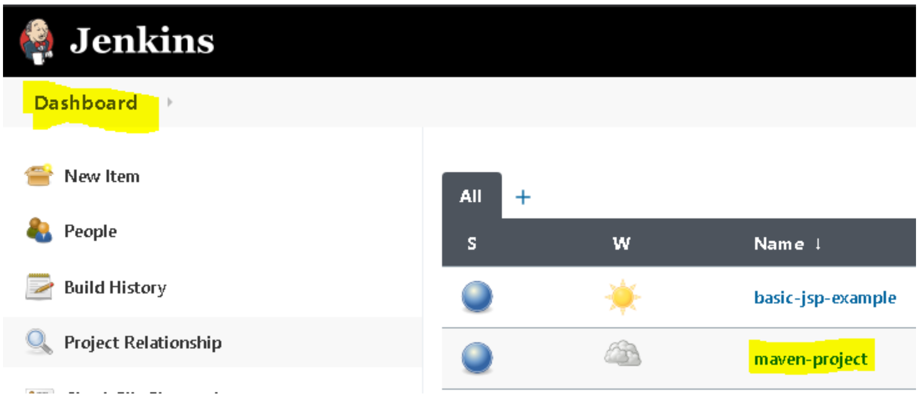
Username

master

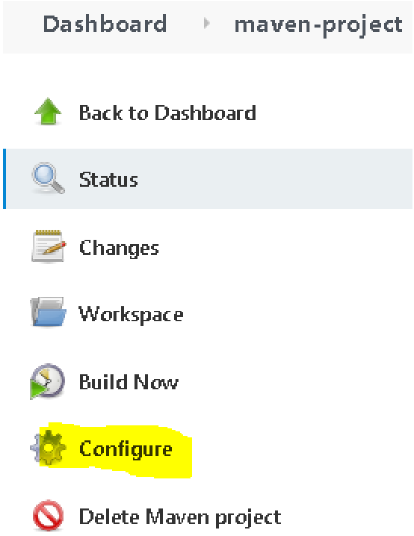
Remote Directory

Success

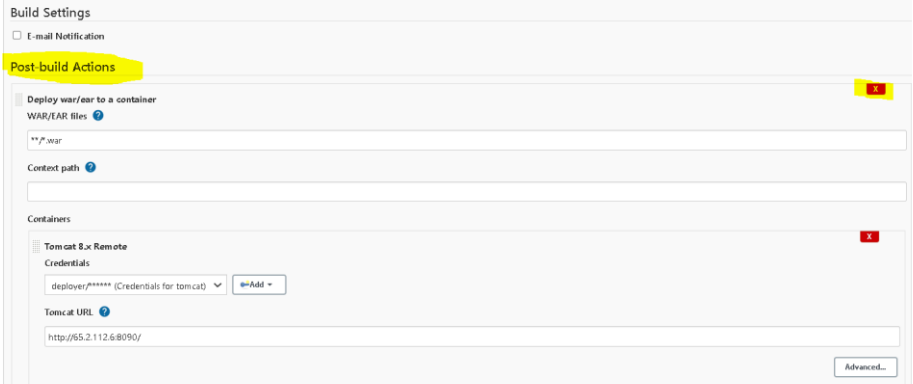
Click on “Dashboard” and then “maven-project” job.



Then, configure.



Remove “Post-Build Actions”, click on red-cross.



Build Settings

☐ E-mail Notification

Post-build Actions

Deploy war/ear to a container

WAR/EAR files: **/*.war

Context path:

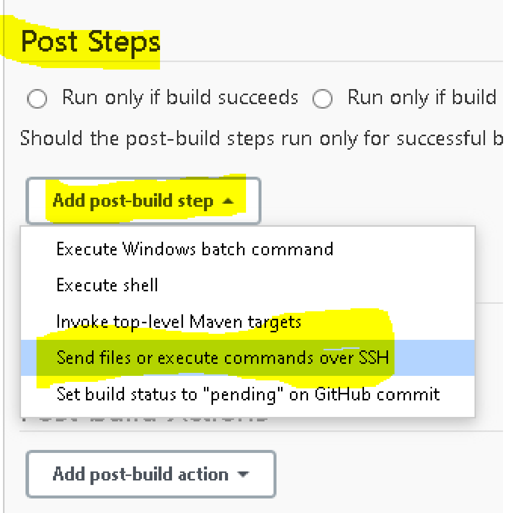
Containers

Tomcat 8.x Remote

Credentials: deployer/****** (Credentials for tomcat)

Tomcat URL: http://65.2.112.6:8090/

Choose post-build actions as follows:

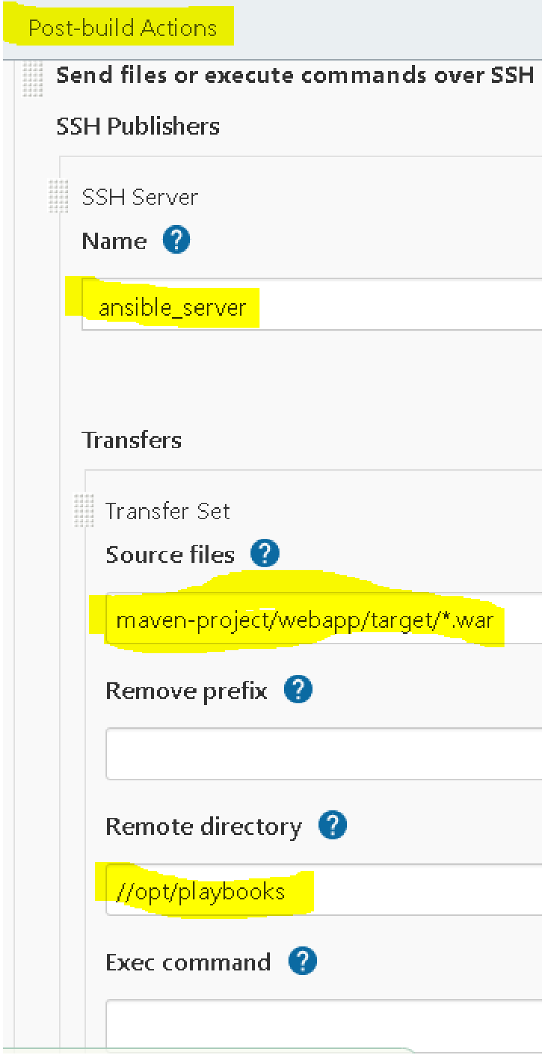


Post Steps

○ Run only if build succeeds ○ Run only if build

Should the post-build steps run only for successful b

Add post-build step
Execute Windows batch command
Execute shell
Invoke top-level Maven targets
Send files or execute commands over SSH
Set build status to "pending" on GitHub commit
Add post-build action



Post-build Actions

Send files or execute commands over SSH

SSH Publishers

SSH Server

Name: ansible_server

Transfers

Transfer Set

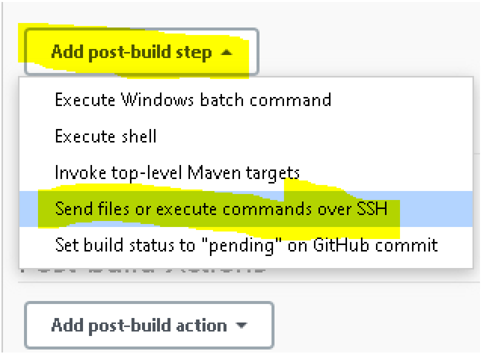
Source files: maven-project/webapp/target/*.war

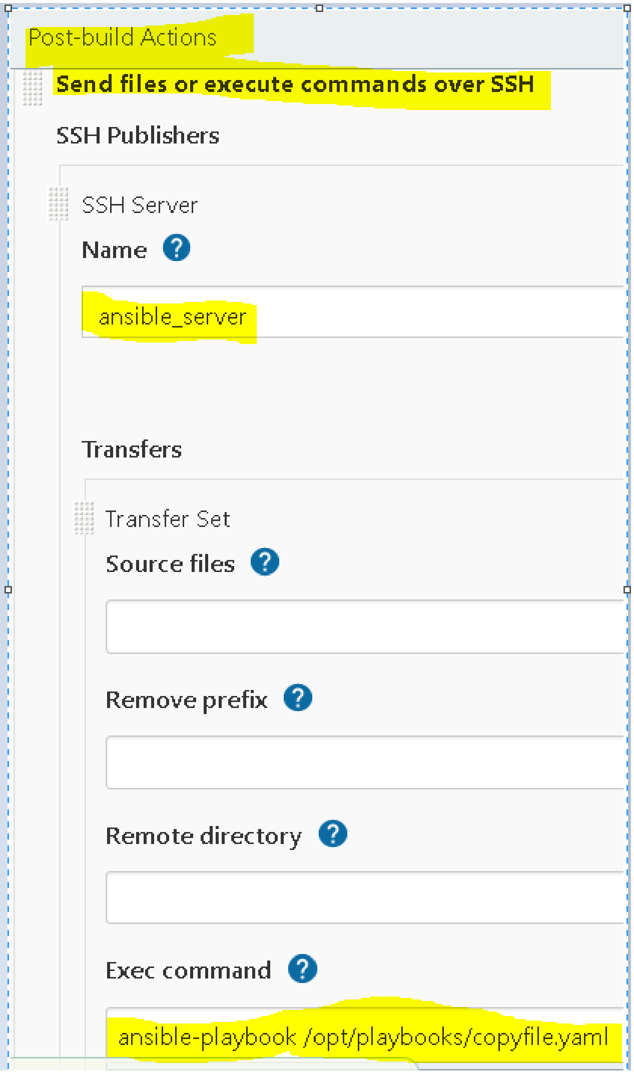
Remove prefix:

Remote directory: //opt/playbooks

Exec command:

Add another post-build step as follows.


Add post-build step
Execute Windows batch command
Execute shell
Invoke top-level Maven targets
Send files or execute commands over SSH
Set build status to "pending" on GitHub commit
Add post-build action



Post-build Actions

Send files or execute commands over SSH

SSH Publishers

SSH Server

Name: ansible_server

Transfers

Transfer Set

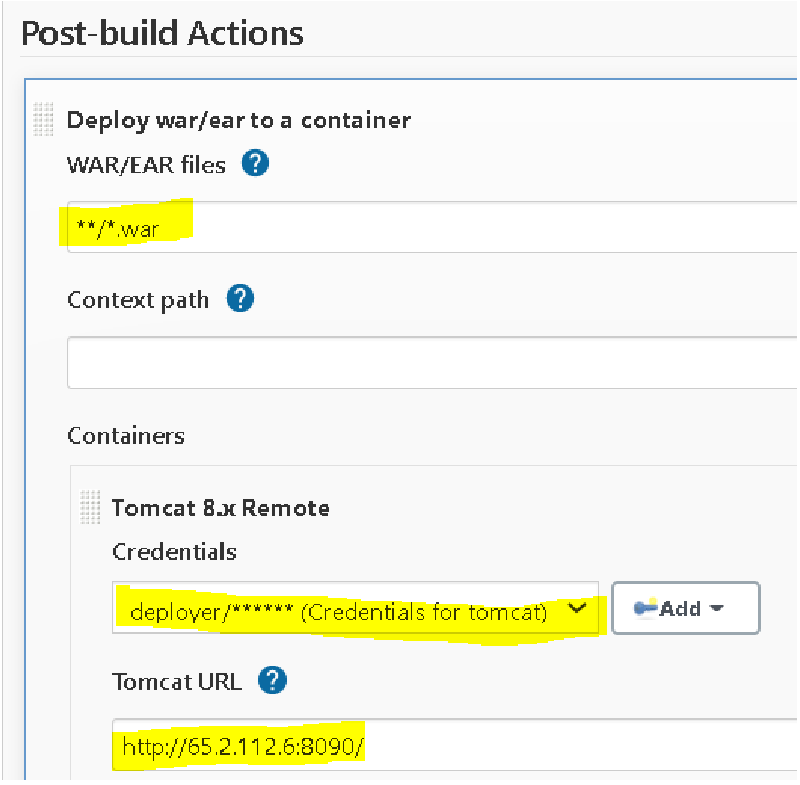
Source files:

Remove prefix:

Remote directory:

Exec command: ansible-playbook /opt/playbooks/copyfile.yaml

Add one more Post-Build step to configure Tomcat server



Post-build Actions

Deploy war/ear to a container

WAR/EAR files: `**/*.war`

Context path:

Containers

Tomcat 8.x Remote

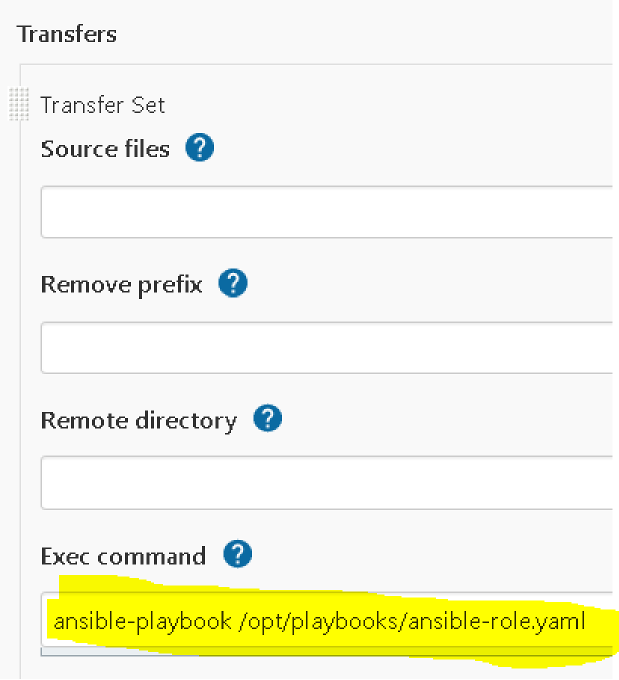
Credentials: deployer/****** (Credentials for tomcat)

Tomcat URL: http://65.2.112.6:8090/

Add another post-build step to work with Ansible roles. We will install curl package on the Ansible server to test if we are able to reach Apache webserver using the command:

curl http://public-ip-of-apache-webserver:8090

Note: This Ansible role will identify the OS of the server and accordingly run the appropriate command to install "curl" package.



Transfers

Transfer Set

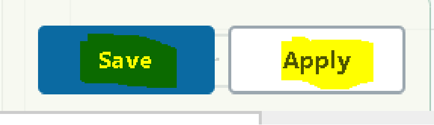
Source files

Remove prefix

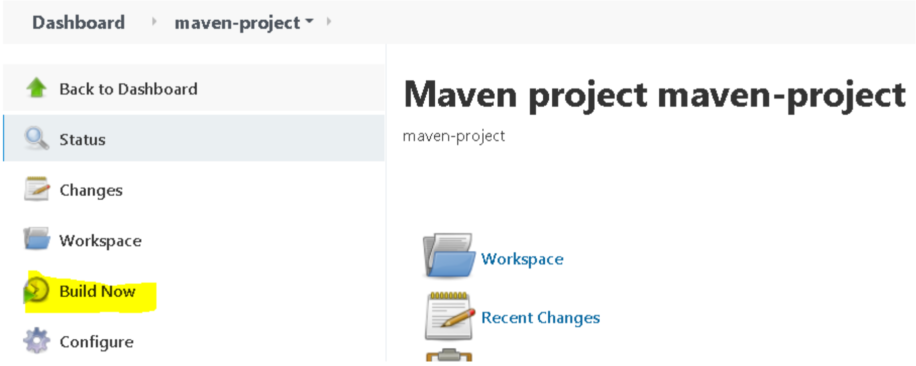
Remote directory

Exec command

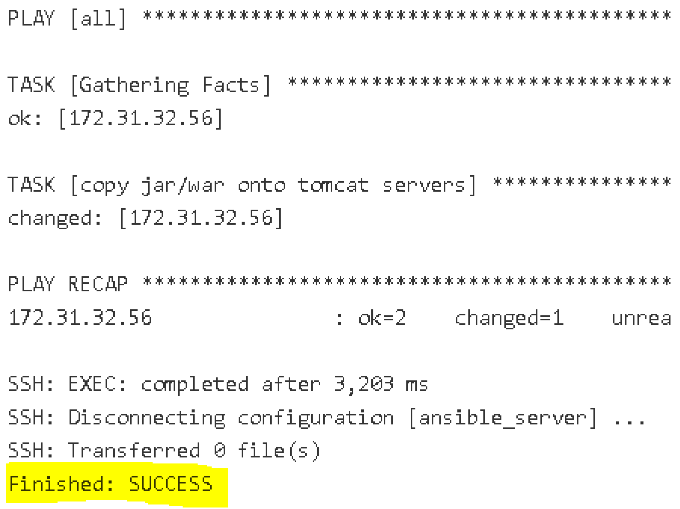
```
ansible-playbook /opt/playbooks/ansible-role.yaml
```



Build the job now to test your new configuration.



check the console output for ‘SUCCESS” message.

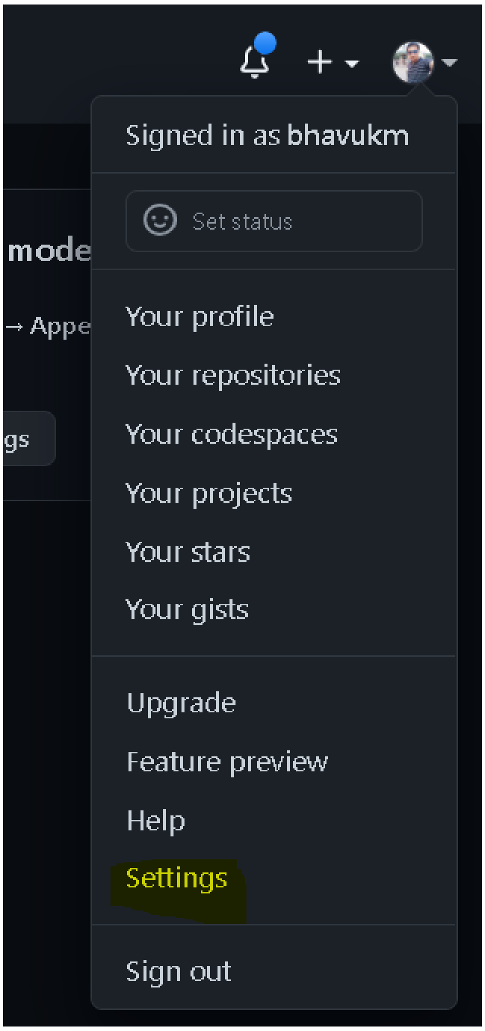


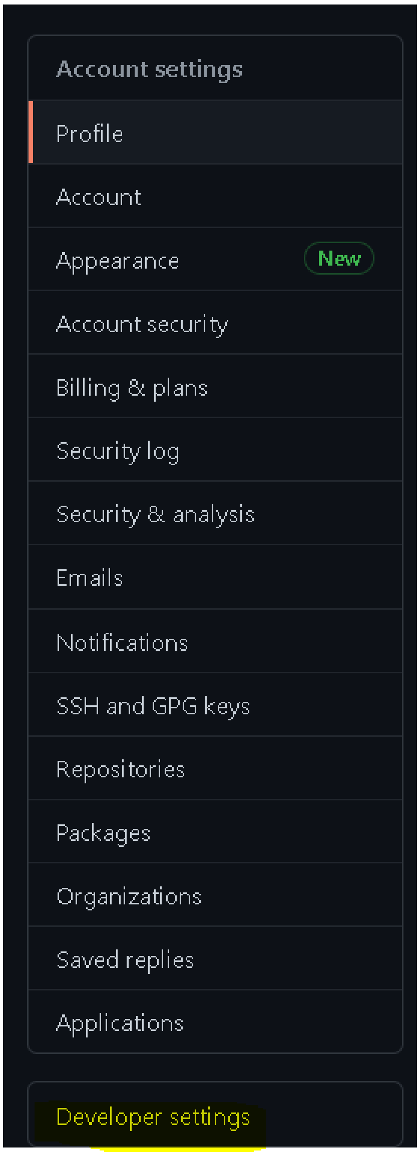
```
PLAY [all] ********************************************

TASK [Gathering Facts] ********************************
ok: [172.31.32.56]

TASK [copy jar/war onto tomcat servers] ***************
changed: [172.31.32.56]

PLAY RECAP ********************************************
172.31.32.56               : ok=2    changed=1    unrea

SSH: EXEC: completed after 3,203 ms
SSH: Disconnecting configuration [ansible_server] ...
SSH: Transferred 0 file(s)
Finished: SUCCESS
```

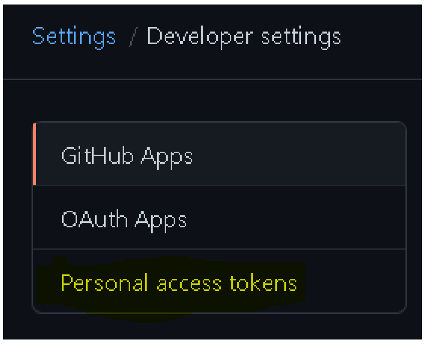
Configuring Webhooks in Jenkins for CI (Continuous Integration)

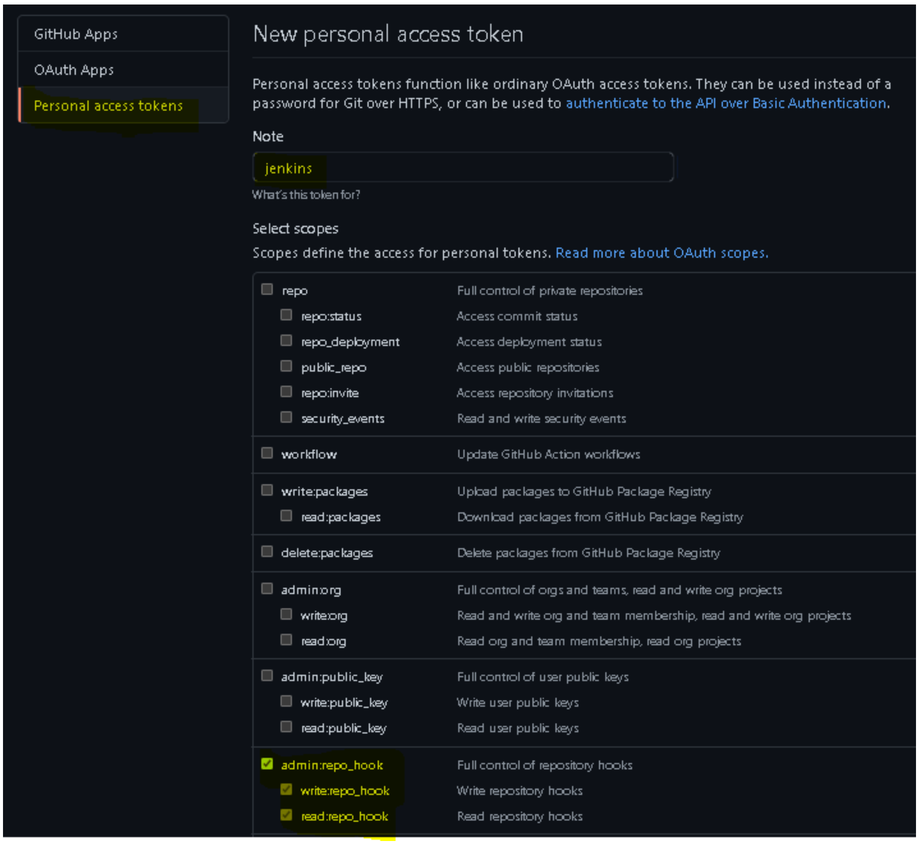
Note: make sure you have forked the below github repo, before generating GitHub token https://github.com/bhavukm/maven-project.git

Login to github.com and follow the screen shots below:




Account settings
Profile
Account
Appearance
New
Account security
Billing & plans
Security log
Security & analysis
Emails
Notifications
SSH and GPG keys
Repositories
Packages
Organizations
Saved replies
Applications
Developer settings





New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

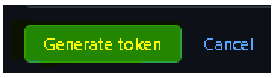
jenkins

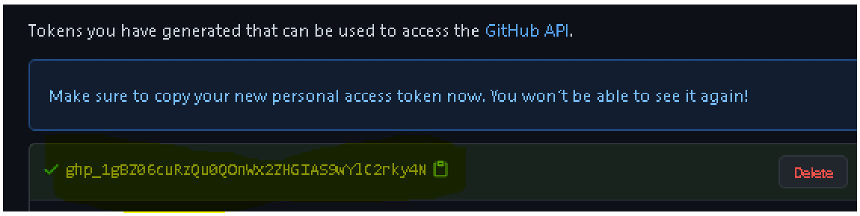
What's this token for?

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

□ repo	Full control of private repositories
□ repo:status	Access commit status
□ repo_deployment	Access deployment status
□ public_repo	Access public repositories
□ repo:invite	Access repository invitations
□ security_events	Read and write security events
□ workflow	Update GitHub Action workflows
□ write:packages	Upload packages to GitHub Package Registry
□ read:packages	Download packages from GitHub Package Registry
□ delete:packages	Delete packages from GitHub Package Registry
□ admin:org	Full control of orgs and teams, read and write org projects
□ write:org	Read and write org and team membership, read and write org projects
□ read:org	Read org and team membership, read org projects
□ admin:public_key	Full control of user public keys
□ write:public_key	Write user public keys
□ read:public_key	Read user public keys
☑ admin:repo_hook	Full control of repository hooks
☑ write:repo_hook	Write repository hooks
☑ read:repo_hook	Read repository hooks



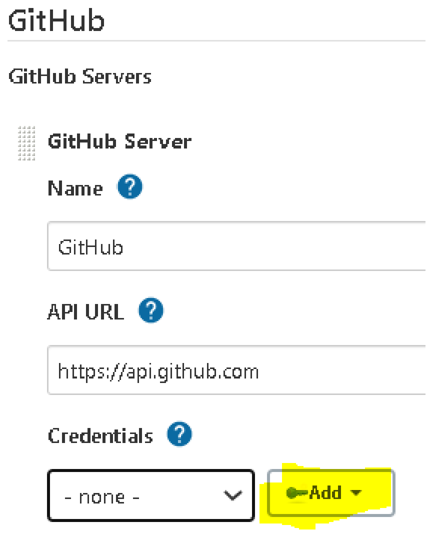


Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ ghp_1gBZ06cuRzQu0QOnWx2ZHGIAS9WY1C2rky4N

Now, go to: Manage Jenkins > Configure System and follow the screen shots below:



GitHub

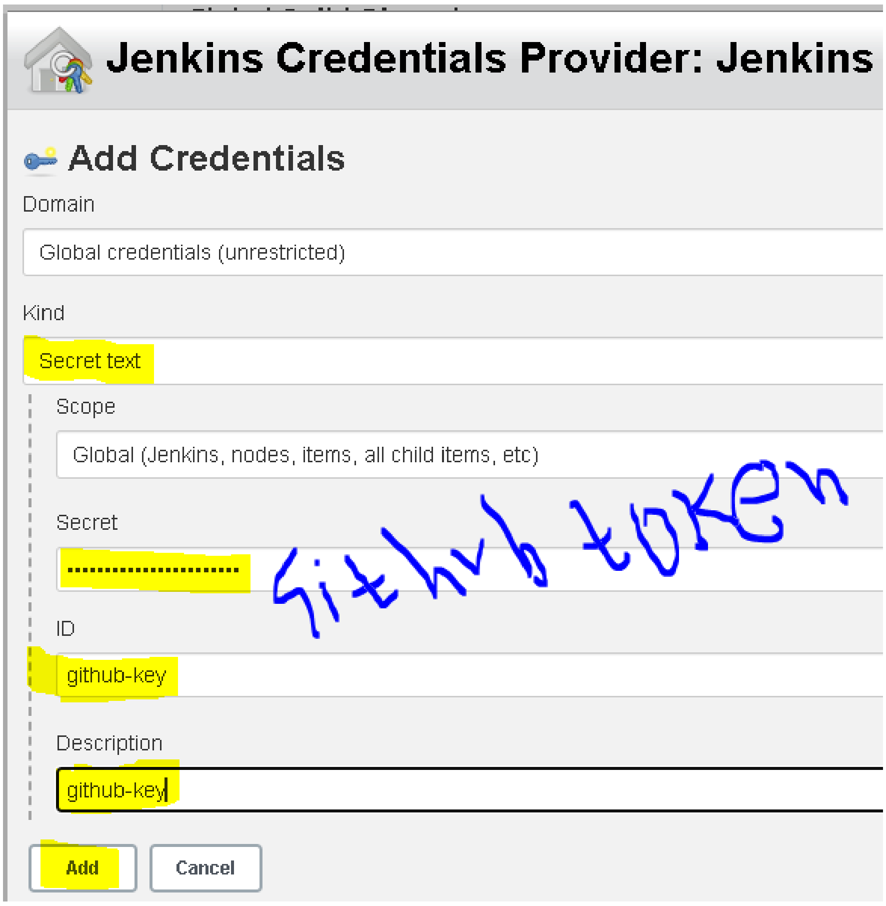
GitHub Servers

GitHub Server

Name: GitHub

API URL: https://api.github.com

Credentials: - none -



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

Secret text

Scope

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

Secret

••••••••••••••••••••••••

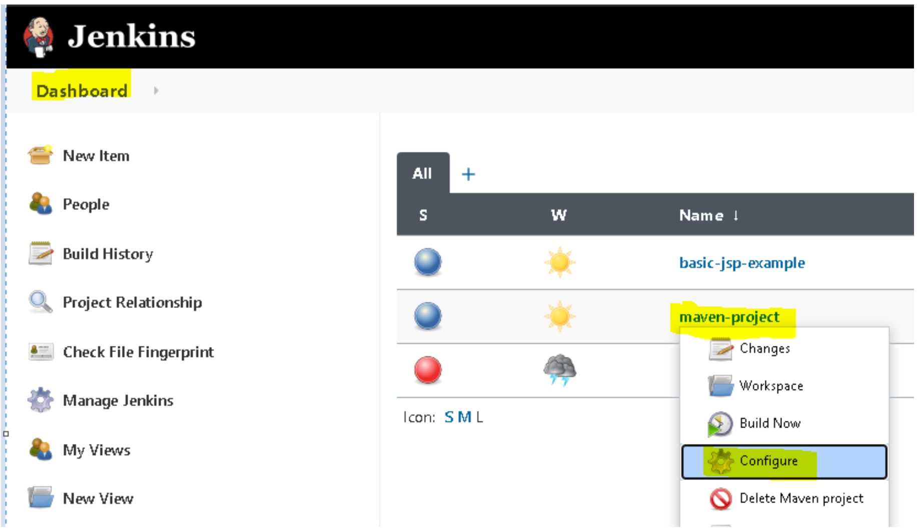
github token

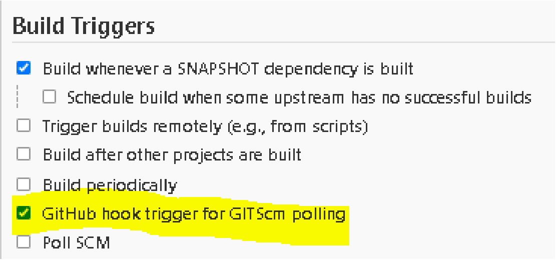
ID

github-key

Description

github-key

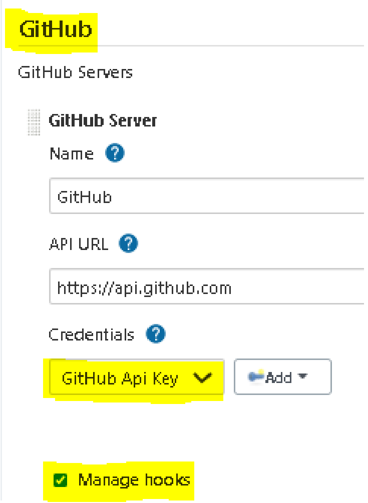




Build Triggers

- ☑ 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 hook trigger for GITScm polling
- □ Poll SCM

Now, again go to: Manage Jenkins > Configure System and follow screen shots:



GitHub

GitHub Servers

GitHub Server

Name ?

GitHub

API URL ?

https://api.github.com

Credentials ?

GitHub Api Key

☑ Manage hooks

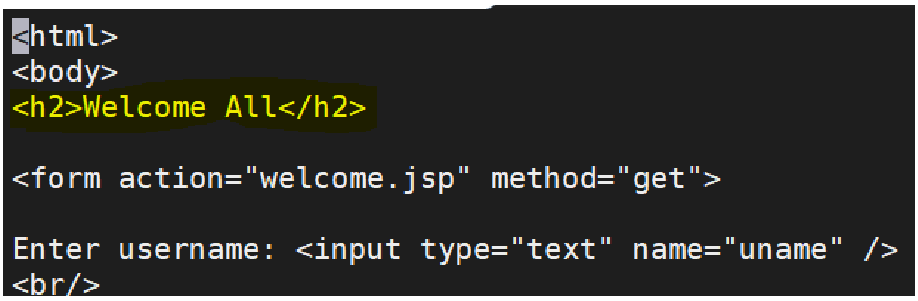
Now, we will test our final CI Configuration:

SSH to your Jenkins server (user: ec2-user) and run commands as given below:

git clone https://github.com/bhavukm/maven-project.git # please use your own forked git repo

vi maven-project/webapp/src/main/webapp/index.jsp

make any text change in <h2> as below and save and exit:



```
<html>
<body>
<h2>Welcome All</h2>

<form action="welcome.jsp" method="get">

Enter username: <input type="text" name="uname" />
<br/>
```

git init

git add .

git commit -m "testing CI"

git remote add repo https://github.com/bhavukm/maven-project.git #use your repo URL

git push repo # enter your GitHub username and password

Now go to Jenkins UI and check for Automated build in the queue.

After the build finishes successfully, go to:

http://public-ip-of-tomcat-server:8090/webapp

and check for updated website page.