**Jenkins Installation**

Introduction

In this hands-on lab, we will install Java and Jenkins. Once this is done and the Jenkins instance is up and running, take some time to look around the interface and ensure you are familiar with all the menu items and other parts of the interface.

Solution

Log in to the server using the credentials provided:

ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>

Install java-1.8.0-openjdk-devel

1. Install Java:

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

Install the Repo and Key, and Then Install Jenkins

1. Install wget:

sudo yum install -y wget

1. Download the repo:

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

1. Import the required key:

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

1. Install Jenkins:

sudo yum install -y jenkins

1. Enable Jenkins:

sudo systemctl enable jenkins

1. Start Jenkins:

sudo systemctl start jenkins

1. In a new browser tab, navigate to http://<PUBLIC\_IP\_ADDRESS>:8080, replacing <PUBLIC\_IP\_ADDRESS> with the IP address of the cloud server provided on the lab page.
2. We'll be taken to an *Unlock Jenkins* page telling us we need to locate the password. In the terminal, run:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

1. Copy the result of the command, as this is the password we need.
2. Paste the password into the *Administrator password* field on the Jenkins browser page.
3. Click **Continue**.
4. Click **Install suggested plugins**.
5. On the new user creation page, set the following values:
   * *Username*: **student**
   * *Password*: **OmgPassword!**
   * *Full name*: **LAstudent**
   * *E-mail address*: **student@la.com**
6. Click **Save and Continue**.
7. Click **Save and Finish**.
8. Click **Start using Jenkins**.
9. Click **Manage Jenkins** in the left-hand menu, and then look around a bit to get familiar with the items in that area.

Conclusion

Congratulations on successfully completing this hands-on lab!

Ubuntu

apt update

10 apt-get install -y java-1.8.0-openjdk

11 wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -

12 sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

13 sudo apt update

14 sudo apt install jenkins

15 sudo systemctl start jenkins

16 service start jenkins

17 sudo ufw allow 8080

18 sudo ufw status

19 sudo ufw enable

20 sudo apt install jenkins

21 rm /var/lib/dpkg/info/$nomdupaquet\* -f

22 sudo apt install jenkins

23 service start jenkins

24 systemctl start jenkins

25 systemctl status jenkins

26 sudo apt install openjdk-8-jdk

27 sudo apt install jenkins

28 systemctl status jenkins

29 systemctl restart jenkins

30 systemctl status jenkins

31 sudo ufw status

32 ufw allow openssh

33 sudo ufw status

# Building from SCM

## Introduction

In this lab we configure Maven to perform a build. This includes pulling the source code for the build from SCM. At the end of the build process, we also create an artifact for the build.

## Solution

### Configure Maven Installer

**Distributing a Build**

Introduction

In this hands-on lab, we will configure Maven to build a project pulled from SCM — but we will configure a slave node to build the project instead of building the project on the master node.

Solution

Log in to the Jenkins master server using the credentials provided:

ssh cloud\_user@<MASTER\_PUBLIC\_IP\_ADDRESS>

Become root:

sudo su

Configure the Slave Machine for Use with the Jenkins Master

1. Open the /etc/passwd file:

[root@master]$ vim /etc/passwd

1. In the last line in the file (beginning with jenkins), change /bin/false to /bin/bash to allow the jenkins user a shell login.
2. Save and exit the file by pressing **Escape** followed by :x.
3. Change the password for the jenkins user:

[root@master]$ passwd jenkins

1. Enter a password of your choice that you'll easily remember.
2. Switch to jenkins:

[root@master]$ su jenkins

1. Change directory:

[jenkins@master]$ cd ~

1. Generate a public/private RSA key pair:

[jenkins@master]$ ssh-keygen

1. Log in to the slave server:

[jenkins@master]$ ssh cloud\_user@<SLAVE\_PUBLIC\_IP\_ADDRESS>

1. Become root:

[cloud\_user@slave]$ sudo su

1. Create a jenkins user:

[root@slave]$ useradd jenkins

1. Create a password:

[root@slave]$ passwd jenkins

1. Open the sudoers file:

[root@slave]$ visudo

1. In the Defaults section, beneath root, add:

jenkins ALL=(ALL) NOPASSWD: ALL

1. Save and exit the file by pressing **Escape** followed by :x.
2. Exit root:

[root@slave]$ exit

1. See who you're logged in as:

[cloud\_user@slave]$ whoami

You should see you're cloud\_user.

1. Switch to jenkins:

[cloud\_user@slave]$ su jenkins

Enter the password you created.

1. Change directory:

[jenkins@slave]$ cd ~

1. Enter exit *twice* to exit back to the master server.
2. See who you're signed in as:

whoami

You should see you're jenkins.

1. As the jenkins user on the master server, copy the jenkins user's ssh keys to the slave server:

[jenkins@master]$ ssh-copy-id jenkins@<SLAVE\_PUBLIC\_IP\_ADDRESS>

1. Run the following:

cat ./.ssh/id\_rsa

Keep the output listed, as we'll need it for a later step.

Run the Maven Build on the Remote Agent

1. In a new browser tab, navigate to http://<JENKINS\_MASTER\_SERVER\_PUBLIC\_IP>:8080.
2. Log in to Jenkins using the following credentials:
   * *User*: **student**
   * *Password*: **OmgPassword!**
3. Click **Manage Jenkins** in the left-hand menu.
4. Click **Manage Nodes and Clouds**.
5. Click **New Node**.
6. Give it a name of slave1.
7. Select **Permanent Agent**.
8. Click **OK**.
9. For *Remote root directory*, enter /home/jenkins.
10. For *Labels*, enter slave1.
11. For *Host*, enter the slave server's public IP address.
12. Next to *Credentials*, click **Add** > **Jenkins**.
13. Set the following values:
    * *Kind*: **SSH Username with private key**
    * *Username*: **jenkins**
    * *Private Key*: **Enter directly**
      + Copy the *entire* RSA key in the terminal (from dashes to dashes) and paste it into the *Key* window
    * *ID*: **jkey**
    * *Description*: **jenkinsuser**
14. Click **Add**.
15. Set *Credentials* to **jenkins (jenkinsuser)**.
16. Click **Save**.
17. In the upper-left corner, click **Jenkins** > **New Item**.
18. Enter an item name of mavenproject.
19. Select **Freestyle project**.
20. Click **OK**.
21. Set the following values:
    * **General**
      + *Restrict where this project can be run*: Check
      + *Label Expression*: **slave1**
    * **Source Code Management**
      + *Git*: Check
      + *Repository URL*: https://github.com/linuxacademy/content-cje-prebuild.git
        - Click outside the box to make sure the red text goes away.
    * **Build**
      + Click **Add build step** > **Invoke top-level Maven targets**.
        - *Goals*: **clean package**
      + Click **Add build step** > **Execute shell**.
        - *Command*: bin/makeindex
    * **Post-build Actions**
      + Click **Add post-build action** > **Archive the artifacts**.
        - *Files to archive*: index.jsp
      + Click **Advanced...**.
        - *Fingerprint all archived artifacts*: Check
        - Leave other default boxes checked.
22. Click **Save**.
23. In the upper-left corner, click **Jenkins** > **Manage Jenkins** > **Global Tool Configuration**.
24. In the *Maven* section, click **Add Maven**.
25. Give it the name **M3**.
26. Click **Save**.
27. In the upper-left corner, click **Jenkins**.
28. Click **mavenproject**.
29. Click **Configure** in the left-hand menu.
30. In the *Build* section, set *Maven Version* to **M3**.
31. Click **Save**.
32. Click **Build Now** in the left-hand menu.
33. Once the build starts, click the dropdown icon next to *#1* and select **Console Output** and observe its progress.

Conclusion

Congratulations on successfully completing this hands-on lab!

1. Use a browser to navigate to the provided public IP address for the server. Remember to add ":8080" to the end of the IP address to specify the default Jenkins port.
2. Log in with the credentials provided in the lab guide instructions.
3. Click **Manage Jenkins**.
4. Click **Global Tool Configuration**.
5. Under Maven installations, click **Add Maven**.
6. In the Name box, enter "M3".
7. Make sure **Install automatically** is checked.
8. Click **Save**.

### Configure the Build to Use Maven and Make the Index File

1. Click **New Item**.
2. Enter an item name of "mavenproject" in the box provided.
3. Select **Freestyle project**.
4. Click **OK**.
5. Click the **Source Code Management** tab at the top of the screen.
6. Select the option for a **Git** repository.
7. Copy the git repository link from the lab instructions and enter it into the Repository URL box.
8. Click the **Build** tab at the top of the screen.
9. Click **Add build step** and select the **Invoke top-level Maven targets** option.
10. Under Maven Version, select **M3**.
11. In the Goals box, enter "clean package".
12. Click **Add build step** and select the **Execute shell** option.
13. In the Command window, enter "bin/makeindex".
14. Click **Add post-build action** and select the **Archive the artifacts** option.
15. Inside the **Archive the artifacts** box, click **Advanced...**
16. Check the option for **Fingerprint all archived artifacts**.
17. In the Files to archive box, enter "index.jsp".
18. Click **Save**.
19. Click **Build Now**.
20. Refresh the window and click the **View** link next to **index.jsp**. Verify the contents of the index.jsp file.

## Conclusion

Congratulations — you've completed this hands-on lab!

In this video we are going to be looking at the Jenkins rest API. We will investigate how to run a build, copy a job and restart the server. Commands that were used in this lesson: Get crumb CRUMB=$(wget -q --auth-no-challenge --user username --password Password --output-document - 'http://192.168.1.49:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)') 1139d7870b3745018e44079544ac478117 Run a build curl -I -X POST http://darealmc:1193974fe1f27156c1cca4f218d9f0b751@192.168.1.49:8080/job/realmcjobs/job/pipeone/build -H "$CRUMB" Get job info curl -X GET http://darealmc:1193974fe1f27156c1cca4f218d9f0b751@192.168.1.49:8080/job/realmcjobs/job/pipeone/config.xml -H "$CRUMB" -o ./jobconfig.xml Create a job not in folder curl -s -X POST 'http://darealmc:1193974fe1f27156c1cca4f218d9f0b751@192.168.1.49:8080/createItem?name=copyone' --data-binary @jobconfig.xml -H "$CRUMB" -H "Content-Type:text/xml" Create a job in a folder curl -s -XPOST 'http://darealmc:1193974fe1f27156c1cca4f218d9f0b751@192.168.1.49:8080/job/realmcjobs/createItem?name=copyone' --data-binary @jobconfig.xml -H "$CRUMB" -H "Content-Type:text/xml

### How do you feel about this video?