Pre-Assignment

Jenkins: Build a simple deployment pipeline

Set Jenkins up in an AWS EC2 server

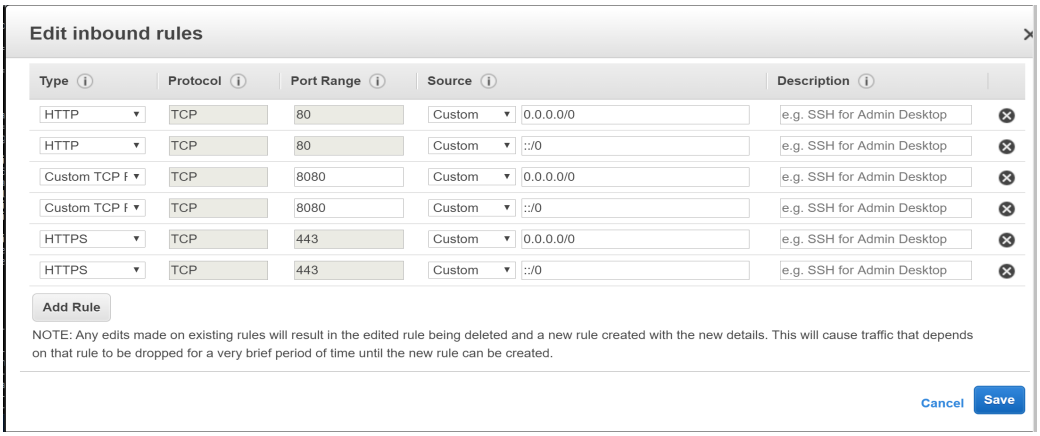
Jenkins can run on your local machine, but it is more practical to host it on a server. By running Jenkins on a server, tests do not use any of your developing systems resources. For this tutorial we will run Jenkins on an AWS EC2 server.
EC2 servers by default have Java 7 installed. ***In order to run Jenkins on your EC2 server, you will need to remove Java 7 and install java 8****. You will also need to install tools that Jenkins will need to act as a continuous delivery environment.*

Note: *There can be a cost associated with using EC2. AWS allows certain services to be run within a "free tier" for the first year of use of your AWS account, so most students can run an EC2 instance for free. However, if you have used up your free tier eligibility (by having created an AWS account more than a year ago, possibly for CS 260 or another class) you could be charged. However, the charges are small. If you create a EC2 t2.micro server for use in this lab, and leave it running for an entire week, you will be charged approximately $1.95 if you are not still eligible for the free tier. We recommend creating an EC2 t2.micro instance as you do this pre-assignment and then stopping (but not terminating) your instance when you are done. You can then restart it when you need for the next two weeks when you do the two Jenkins tutorials for this class.*

1. Create an AWS EC2 Linux instance
    a. If you need help, look at the previous AWS Web Server Tutorial
    b. Make sure it is type "**Amazon Linux 2 AMI (HVM), SSD Volume Type**"
2. Connect to the AWS server via SSH : `ssh -i path\To\KeyPair\File ec2-user@IPAddressOfEC2`
3. Install updates : `sudo yum update`
4. Install Git `sudo yum install git -y`
5. Remove java 7 and download Java 8 and Jenkins using these commands in your EC2 terminal :
    a. `sudo yum remove java-1.7.0-openjdk`
    b. `sudo yum install java-1.8.0`
    c. `sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo`
    d. `sudo rpm --import http://pkg.jenkins.io/redhat-stable/jenkins.io.key`
    e. `sudo yum install jenkins -y`
        i. If you run into an error because you lack the Daemonize package run the following commands
            1. `sudo amazon-linux-extras install epel -y`
            2. `sudo yum update -y`
            3. `sudo yum install jenkins java-1.8.0-openjdk-devel -y`

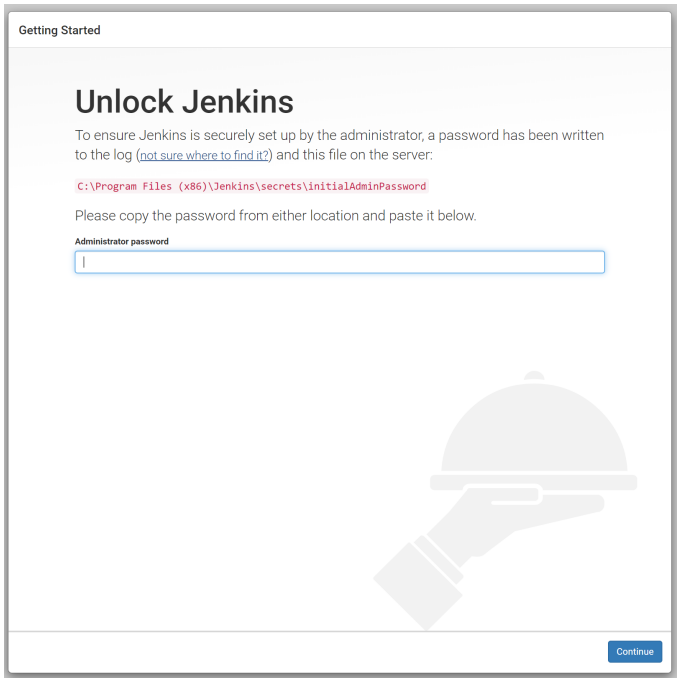
f. `sudo service jenkins start`
g. If you want to understand these steps better or need further help look at this tutorial : [Install and Configure Jenkins on Amazon AMI (Part II)]

6. To connect to Jenkins AWS server, you need to have the correct security rules set up
    a. Make sure that you allow inbound traffic from ports 80, 8080, and 443 (HTTP, Custom TCP, and HTTPS)
        i. **If you need help configuring these rules, refer to this mini-tutorial:** [How to set-up an AWS security group for Jenkins]
    b. If you set up everything correctly, you should be able to connect to the Jenkins Dashboard by typing in "*your_EC2_IP_ADDRESS:8080*" into the URL field of a web browser. The host address can be found on the "Description" tab for your EC2 instance. It is the value for the "Public DNS (IPv4)" field.
    c. Here is a screen shot of the AWS security group I made for this project

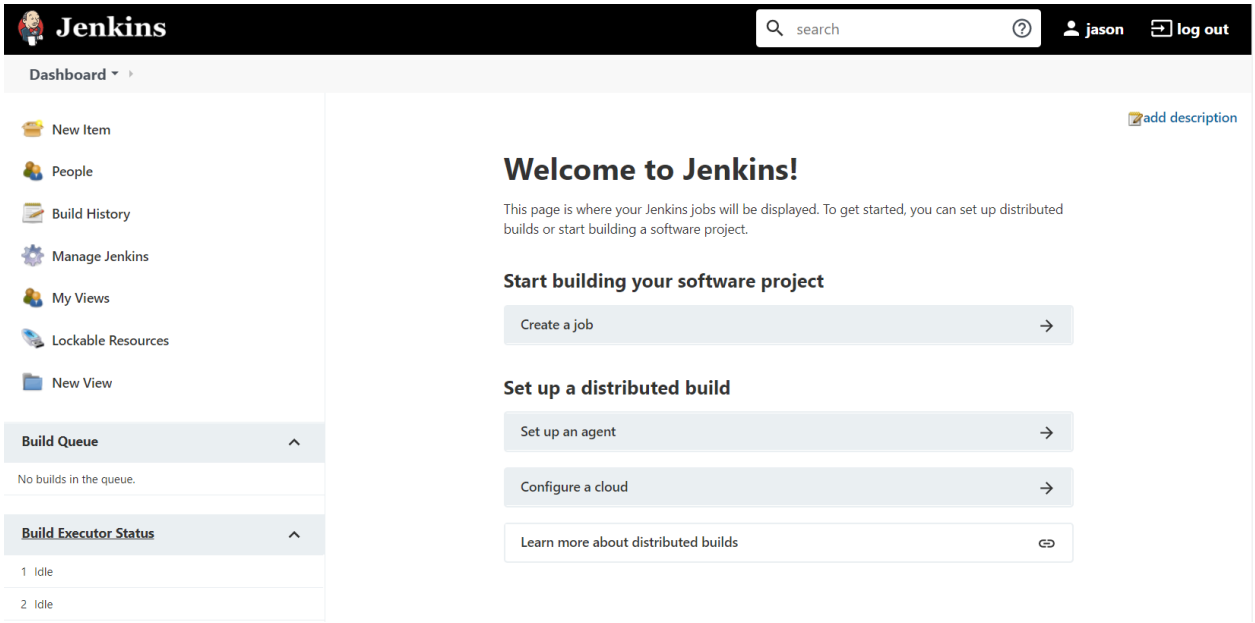


Set up Jenkins

1. Navigate to "*your_EC2_IP_ADDRESS:8080*"
2. Remain on the "*your_EC2_IP_ADDRESS:8080*" until the screen below appears



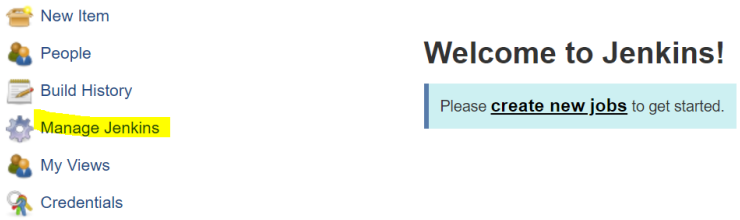
3. You will need to get the initial admin password generated by Jenkins
    a. On your EC2 AWS server retrieve the password by typing in the following command:
        i. `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
            1. **You may have to wait for Jenkins to create this file before you can open it**
4. After inputting the Admin password, you will be prompted to install plugins for Jenkins; **install suggested plugins**
    a. **This will install the git and Pipeline plugin that are required**
5. After the plugins install you will be prompted to create a new Administrator User
6. Create one **with credentials that you will not forget**
    a. If you forget you ill have to **uninstall and re-install Jenkins for the next lab**
7. If everything went well, your screen should look like the screenshot below



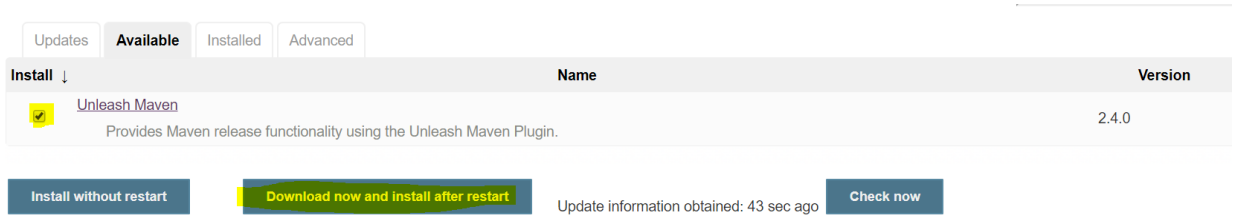
Install Jenkins Plugins

Plugins extend the base functionality of Jenkins

1. Go to the main Jenkins Dashboard
2. Click “Manage Jenkins”



3. Click “Manage Plugins”
4. In available plugins , search for “Unleash Maven” plugin

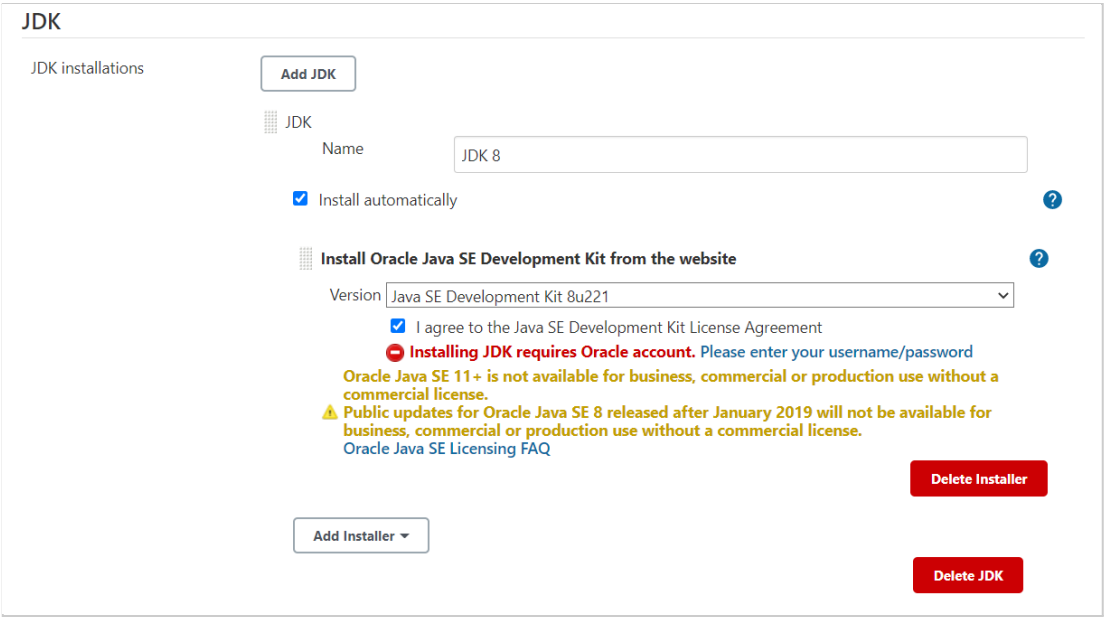


5. Download and restart Jenkins

Configure Jenkins Tools

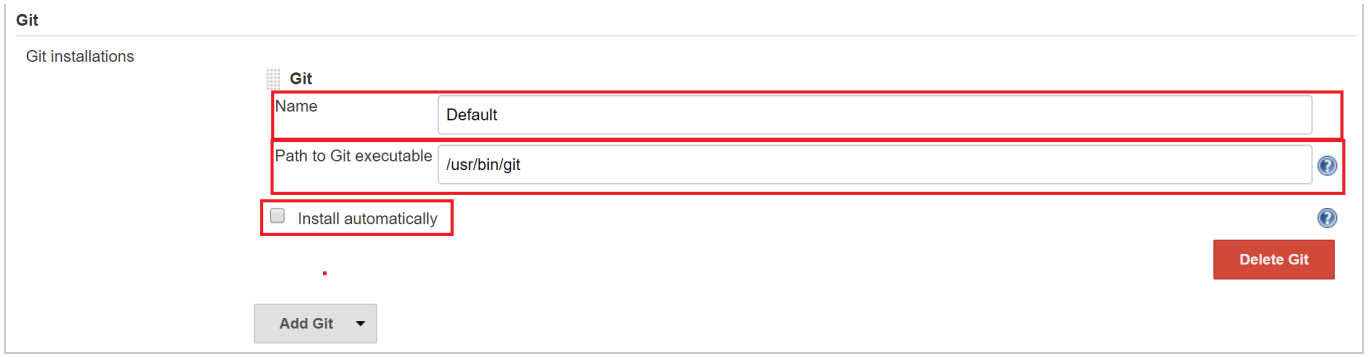
For Jenkins to be able to use Maven, JDK, and Git you must tell it where those tools are in your system or how Jenkins can acquire them.

1. **Navigate to the Global Tool Configuration menu**
    a. Navigate to the main dashboard of Jenkins
    b. Navigate to Manage Jenkins -> Global Tool Configuration
2. **Add JDK**
    a. Go to JDK
    b. Click JDK installations -> “Add JDK” -> **Check install automatically**
    c. Set the JDK name to “JDK 8”
        i. This name will identify which JDK Jenkins will use.(if you have multiple JDKs on your system).
    d. Agree to java SE development Kit License Agreement
    e. **You may need to give Jenkins your oracle Account login information**
        i. Oracle is the current owner of Java. Oracle requires that you have an Oracle account in order to install JDK. **If you do not have an account**, you must create one from their website: [Create an Oracle Account]
    f. Jenkins now has the ability to automatically download and install the specified Java JDK when it is needed.
    g. The JDK option should look like the screenshot below



    h. Make sure to click “Please enter your username/password” and put your oracle account information in it. **Make sure these credentials are correct** or you will have build errors later on. **If you don’t have an oracle account, go create one it is free.**

3. **Set up Git path**
    a. Navigate to Git in the Global Tool Configuration menu
    b. Leave the name as Default
    c. **Set path to Git** executable to : **/usr/bin/git**
        i. This pathway is where AWS ec2 puts the Git bin when you install it
        ii. If you were to do this on another system, you will need to find the path to the Git Bin
    d. Leave install automatically option **unchecked**
    e. The Git settings should look like the screenshot below



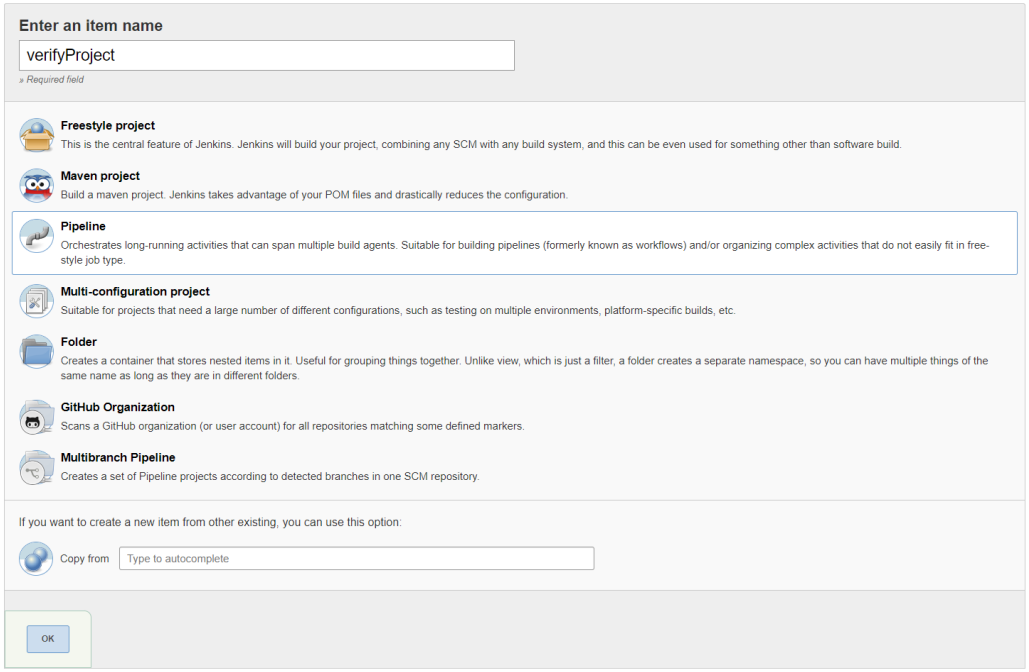
4. **Add Maven**
    a. Scroll down to Maven in the Global Tool configuration menu (NOT Maven Configuration).
    b. Click Maven Installations -> “Add Maven” ->**Check install automatically**
    c. Give Maven the name “apache maven 3.6.3”
    d. Select version 3.6.3
    e. Save and Apply
    f. The Maven options should look like the screenshot below

5. Click **Save**

Verify Set-up

Jenkins should now be able to run maven projects, Use Git, and compile Java files. We are going to verify everything is working correctly by creating a simple Jenkins Job

1. **Navigate to the main Jenkins Dashboard.**
2. On the top left is an option called “new item”
3. **Click “new item”**
4. You should see a dashboard like the one on the screenshot below
5. **Enter “verifyProject”** for the item name and **click Pipeline**
6. Your dashboard should look like the screenshot below



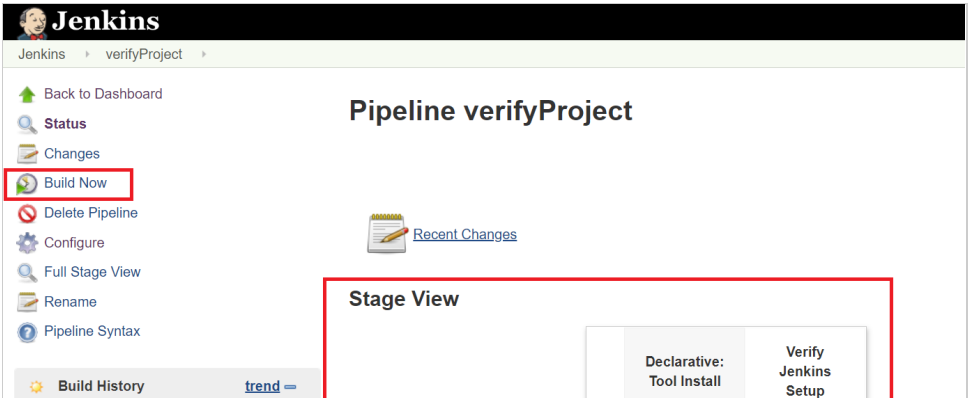
7. **Click OK**
8. There are a lot of options in the next screen, A lot of this will be explained in the tutorial. For now **navigate to the “Pipeline” section**
9. The terminal below accepts Pipeline script to build Jenkins projects
10. **Copy and Paste the following commands** to verify the setup:

```
pipeline {
    agent any
    tools {
        maven 'apache maven 3.6.3'
        jdk 'JDK 8'
    }
    stages {
        stage ('Verify Jenkins Setup') {
```

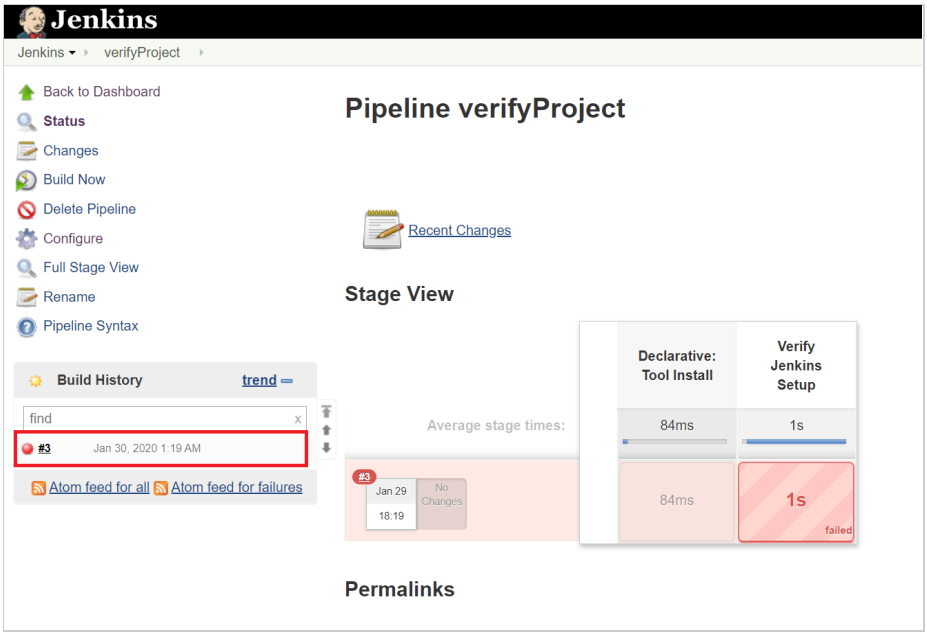
```
            steps {
                sh 'mvn --version'
                sh 'java -version'
                sh 'javac -version'
                sh 'git --version'
            }
        }
    }
}
```

11. Your Pipeline step should look like the screenshot below

12. Click Save
13. You should be brought to the dashboard for the job
14. **Click "Build Now"** on the left side of the dashboard
15. If everything was setup correctly it should look like the screenshot below



16. If something is wrong then the dashboard will look like the screenshot below (pink coloring instead of green)



    a. To check what went wrong, Click on the build number as shown on the screenshot above
        i. Then click “console output” on the left side of the menu
            1. This menu will show you what what outputted to the console, and will tell you why the Jenkins Build failed

Finishing Steps

You are finished setting up Jenkins on an AWS EC2 server. You are ready to use Jenkins. We are going to use this AWS EC2 server for both Jenkins Part 1 and Jenkins Part 2 tutorials, so make sure you **don’t terminate your AWS EC2 instance.** Also make sure you **DO NOT FORGET YOUR JENKINS LOGIN CREDENTIALS.** It can be a pain to retrieve it, and it is usually easier just to remake the whole AWS instance.

1. You are ready for the tutorial!
2. If you are not going to continue to the tutorial today, Stop your AWS instance - **DO NOT TERMINATE**