

# Pre-class Assignment

Jenkins: Build a simple deployment pipeline

This pre-class assignment will prepare you for the upcoming in-class assignment. There will also be a quiz about the readings and the set-up with a question asking if you did all of the following.

## Overview

In this pre-class assignment you will:

1. Learn about what Jenkins is and how it is useful
2. Launch an AWS EC2 instance
3. Install Jenkins, Git, Java, and Maven on the AWS EC2 instance
4. Verify Jenkins was set up correctly to be used in the tutorial

## Learn The Basics of Jenkins

Read the following articles and answer the questions found below

10 min - <https://dev.to/bugfenderapp/what-is-jenkins-and-why-should-you-be-using-it-2pe>

5 min - <https://www.guru99.com/jenkins-pipeline-tutorial.html>

- Stop after reading the section “**Jenkins Pipeline Concepts**”. You will **NOT** be doing the tutorial in the reading.

## What to learn

- What does Jenkins do?
- How is the base functionality of Jenkins extended?
- How does Jenkins help code development?
- What is a Jenkins pipeline?
- What is a JenkinsFile?
- What is a CI environment?
- What is a CD environment?

## 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-class 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 Jenkins EC2 instance - <https://aws.amazon.com/ec2/>
  - a. If you need help, look at the previous AWS Web Server Tutorial
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-ci.org/redhat/jenkins-ci.org.key`
  - e. `sudo yum install jenkins -y`
  - f. `sudo service jenkins start`
  - g. If you want to understand these steps better or need further help look at this tutorial : <https://medium.com/@mohan08p/install-and-configure-jenkins-on-amazon-ami-8617f0816444>
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: [Link](#)**
  - 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

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom ::0	e.g. SSH for Admin Desktop
Custom TCP	TCP	8080	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP	TCP	8080	Custom ::0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom ::0	e.g. SSH for Admin Desktop

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Buttons: Add Rule, Cancel, Save

## 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

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:

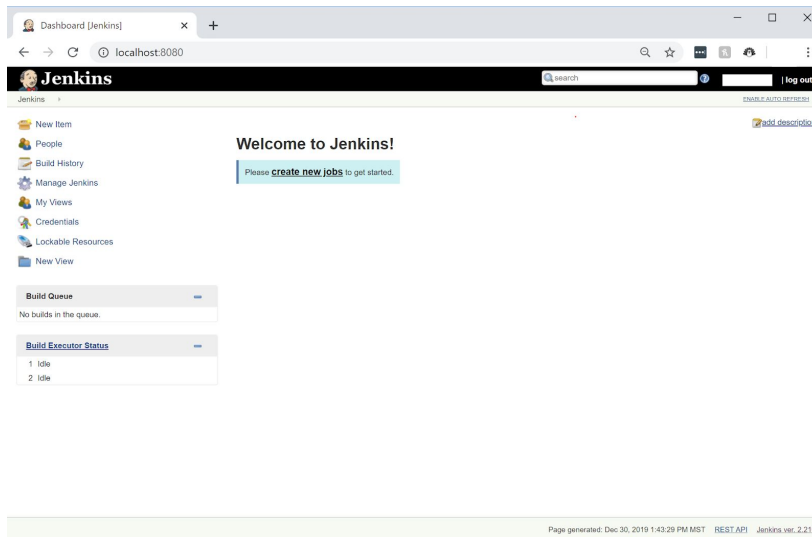
`C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword`

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

Administrator password

Continue

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**
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**
7. If everything went well, your screen should look like the screenshot below



## Install Jenkins Plugins

*Plugins extend the base functionality of Jenkins*

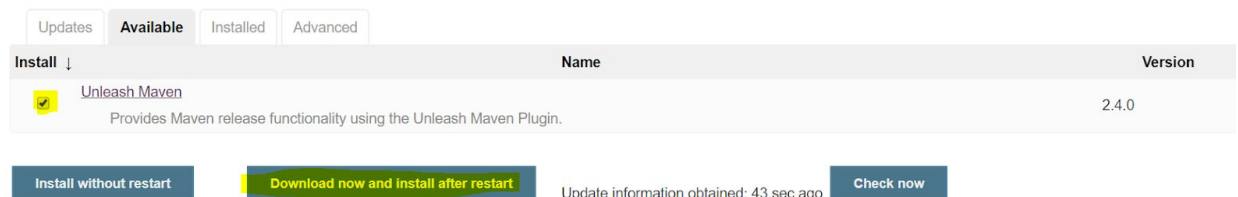
1. Go to the main Jenkins Dashboard
2. Click “Manage Jenkins”



### Welcome to Jenkins!

Please [create new jobs](#) to get started.

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

- Navigate to the main dashboard of Jenkins
- Navigate to Manage Jenkins -> Global Tool Configuration

## 2. Add JDK

- Go to JDK
- Click JDK installations -> "Add JDK" -> **Check install automatically**
- Set the JDK name to "JDK 8"**
  - This name will identify which JDK Jenkins will use.(if you have multiple JDKs on your system).
- Agree to java SE development Kit License Agreement
- You may need to give Jenkins your oracle Account login information**
  - 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. Feel free to follow this link to their account creation page :  
<https://profile.oracle.com/myprofile/account/create-account.jspx>
- Jenkins now has the ability to automatically download and install the specified Java JDK when it is needed.
- The JDK option should look like the screenshot below

JDK

JDK installations

Add JDK

JDK

Name

☒ Install automatically

Install Oracle Java SE Development Kit from the website

Version

☒ I agree to the Java SE Development Kit License Agreement

Oracle Java SE 11+ is not available for business, commercial or production use without a commercial license.

Public updates for Oracle Java SE 8 released after January 2019 will not be available for business, commercial or production use without a commercial license.

[Oracle Java SE Licensing FAQ](#)

Delete Installer

Add Installer

Delete JDK

## 3. Set up Git path

- 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 should 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

Git

Git installations

Git

Name

Default

Path to Git executable

/usr/bin/git

☐

Install automatically

Delete Git

Add Git

#### 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

Maven

Maven installations

Add Maven

Maven

Name

apache maven 3.6.3

☒

Install automatically

Install from Apache

Version

3.6.3

Delete Installer

Add Installer

Add Maven

List of Maven installations on this system

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


Enter an item name


» Required field


 **Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.


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

 **Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in freestyle job type.


 **Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

 **Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

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

 Copy from

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

**Pipeline**

Definition Pipeline script

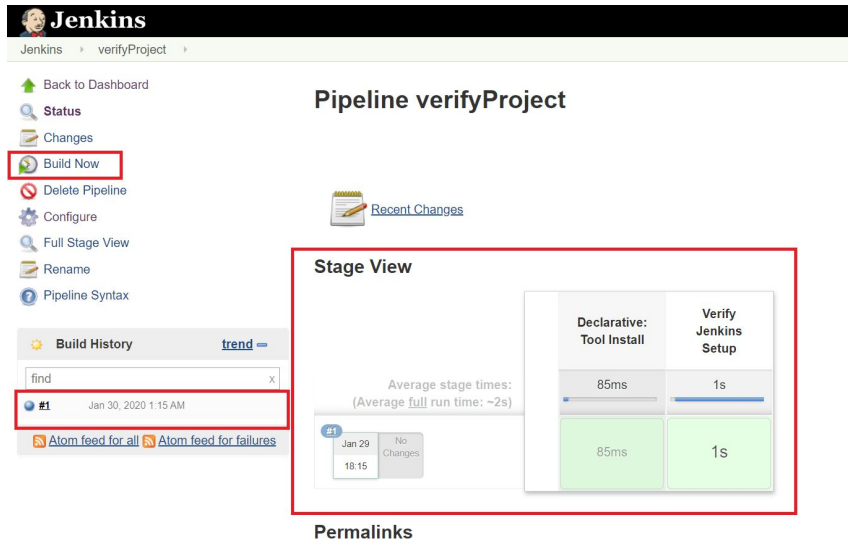
Script

```
1 pipeline {
2   agent any
3   tools {
4     maven 'apache maven 3.6.3'
5     jdk 'JDK 8'
6   }
7   stages {
8     stage ('Verify Jenkins Setup') {
9       steps {
10        sh 'mvn --version'
11        sh 'java -version'
12        sh 'javac -version'
13        sh 'git --version'
14      }
15    }
16  }
17 }
```

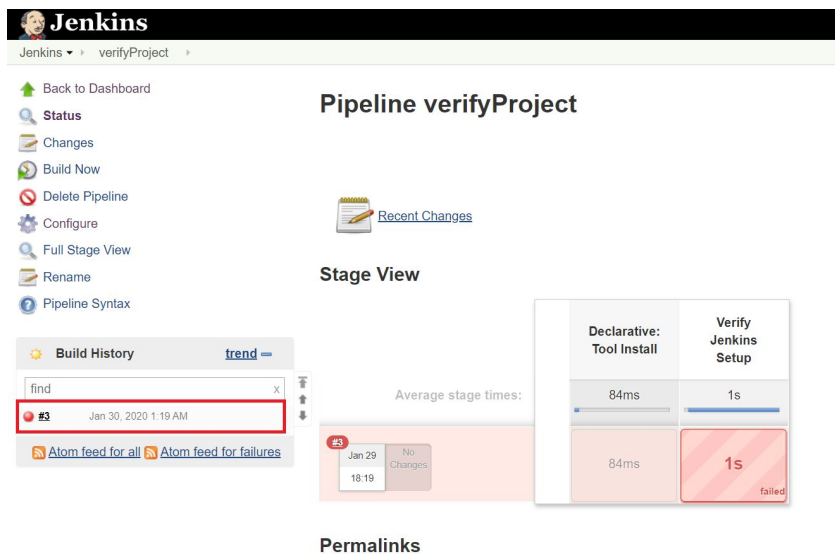
☒ Use Groovy Sandbox

[Pipeline Syntax](#)

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 was 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. Stop your AWS instance - **DO NOT TERMINATE**
2. You are ready for the tutorial!