**Maven**

Example of build tools are Maven, Ant and Gradle.

The build tools are responsible for validate the dependencies, compile the code, test it and then create the artifacts.

Maven is a build automation tool primarily used for java based application.

Artifact: package or bundle of complied code available in war or jar format.

JDK (java development kit) needs to be installed to run the maven.

Definition of Maven: Maven is a build automation tool used primarily for Java projects. Maven addressed the two aspects of building software:

First, it describes how software is built, and second, it describes its dependencies

Step1: Create one RHEL machine from AWS management console

Login to machine using .pem file

Switch user to root user as below

**sudo -i**

Execute below commands to install all these applications in one go.

sudo yum install wget tree unzip git -y

Post installation go to ‘/opt’ dir

cd /opt/

Install the JDK, download latest jdk file from internet and install as below:

\*\*\*\*\*Download java jdk file from internet and extract it.

wget --no-cookies --no-check-certificate --header "Cookie: oraclelicense=accept-securebackup-cookie" https://github.com/frekele/oracle-java/releases/download/8u212-b10/jdk-8u212-linux-x64.tar.gz

To check the file downloaded or not

ls -lrt

# to extract the file, execute below command

tar -xvzf jdk-8u212-linux-x64.tar.gz

go to below path

cd /opt/jdk1.8.0\_212/bin

Check the java version

java -version

[root@maven bin]# java version

-bash: java: command not found

[root@maven bin]#

===============================================

alternatives --install /usr/bin/java java/opt/jdk1.8.0\_21 =====>

alternatives --config java

There are 1 programs which provide 'java'.

Selection Command

-----------------------------------------------

\* 1 /opt/cd /opt/jdk1.8.0\_212/bin/java

======================================================

Enter to keep the current selection[+], or type selection number: 1

===========================================================

to set the java path permanently in bash profile

vi /root/.bash\_profile

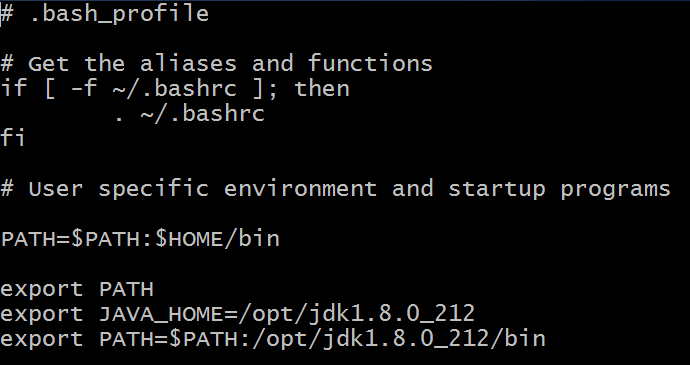
Here add the export cmds

[root@maven bin]# vi /root/.bash\_profile

Add below lines(see the screenshot)

export JAVA\_HOME=/opt/jdk1.8.0\_212

export PATH=$PATH:/opt/jdk1.8.0\_212/bin

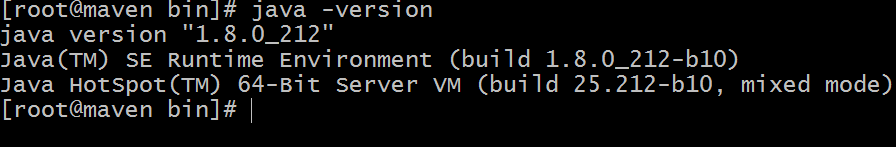


In VI editor to save and quit by :wq

To update/effect the path/environment variable paths execute below command

source /root/.bash\_profile

Then, where ever you check the java version, you will get java -version



**Install Maven**

================

Go to internet and download latest maven package in below folder.

cd /opt

wget http://mirrors.estointernet.in/apache/maven/maven-3/3.6.1/binaries/apache-maven-3.6.1-bin.zip

unzip apache-maven-3.6.1-bin.zip ===> extract the zip file

=======================================================

[root@maven opt]# ls -lrt

total 199412

drwxr-xr-x. 7 10 143 245 Apr 2 04:49 jdk1.8.0\_212

-rw-r--r--. 1 root root 9181276 Apr 4 19:30 apache-maven-3.6.1-bin.zip

drwxr-xr-x. 6 root root 99 Apr 4 21:02 apache-maven-3.6.1

-rw-r--r--. 1 root root 195013152 Apr 20 06:02 jdk-8u212-linux-x64.tar.gz

[root@maven opt]# mvn -v

-bash: mvn: command not found

[root@maven opt]# cd apache-maven-3.6.1

[root@maven apache-maven-3.6.1]# mvn -v

-bash: mvn: command not found

[root@maven apache-maven-3.6.1]# cd bin/

[root@maven bin]# ls -lrt

total 32

-rwxr-xr-x. 1 root root 1532 Apr 4 20:56 mvnyjp

-rw-r--r--. 1 root root 1668 Apr 4 20:56 mvnDebug.cmd

-rwxr-xr-x. 1 root root 1485 Apr 4 20:56 mvnDebug

-rw-r--r--. 1 root root 6349 Apr 4 20:56 mvn.cmd

-rwxr-xr-x. 1 root root 5741 Apr 4 20:56 mvn

-rw-r--r--. 1 root root 228 Apr 4 20:56 m2.conf

[root@maven bin]# ./mvn -v

Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04T19:00:29Z)

Maven home: /opt/apache-maven-3.6.1

Java version: 1.8.0\_212, vendor: Oracle Corporation, runtime: /opt/jdk1.8.0\_212/jre

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "3.10.0-957.el7.x86\_64", arch: "amd64", family: "unix"

To set the path to execute mvn command anywhere, follow below steps

[root@maven bin]# pwd

/opt/apache-maven-3.6.1/bin

export PATH=$PATH:/opt/apache-maven-3.6.1/bin

# .bash\_profile

# Get the aliases and functions

if [ -f ~/.bashrc ]; then

. ~/.bashrc

fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH

export JAVA\_HOME=/opt/jdk1.8.0\_212

export PATH=$PATH:/opt/jdk1.8.0\_212/bin

export PATH=$PATH:/opt/apache-maven-3.6.1/bin

save the file

to update the path, do the source command as below

[root@maven bin]# source /root/.bash\_profile

[root@maven bin]# mvn -v

[root@maven bin]# mvn -v

Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04T19:00:29Z)

Maven home: /opt/apache-maven-3.6.1

Java version: 1.8.0\_212, vendor: Oracle Corporation, runtime: /opt/jdk1.8.0\_212/jre

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "3.10.0-957.el7.x86\_64", arch: "amd64", family: "unix"

**Day 2:**

create a maven project and work in that, follow the steps

cd /opt

mkdir my\_maven\_projects

cd my\_maven\_projects

==> to generate a mvn sample project, an example provided by maven, execute below complete command

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

[root@maven my\_maven\_projects]# tree

.

└── my-app

├── pom.xml

└── src

├── main

│   └── java

│   └── com

│   └── mycompany

│   └── app

│   └── App.java

└── test

└── java

└── com

└── mycompany

└── app

└── AppTest.java

Note: pom.xml file is called it as definition file

Compile can be triggered in the location where pom.xml file located.

cd my-app

[root@maven my\_maven\_projects]# cd my-app/

[root@maven my-app]# ls -l

total 4

-rw-r--r--. 1 root root 642 Apr 23 04:41 pom.xml

drwxr-xr-x. 4 root root 30 Apr 23 04:41 src

[root@maven my-app]#

mvn compile ====> it will compile source code(which includes multiple java files at once)

[root@maven my-app]# tree

.

├── pom.xml

├── src

│   ├── main

│   │   └── java

│   │   └── com

│   │   └── mycompany

│   │   └── app

│   │   └── App.java

│   └── test

│   └── java

│   └── com

│   └── mycompany

│   └── app

│   └── AppTest.java

└── target

├── classes

│   └── com

│   └── mycompany

│   └── app

│   └── App.class

└── maven-status

└── maven-compiler-plugin

└── compile

└── default-compile

├── createdFiles.lst

└── inputFiles.lst

[root@maven my-app]# mvn compile test

[root@maven my-app]# tree

.

├── pom.xml

├── src

│   ├── main

│   │   └── java

│   │   └── com

│   │   └── mycompany

│   │   └── app

│   │   └── App.java

│   └── test

│   └── java

│   └── com

│   └── mycompany

│   └── app

│   └── AppTest.java

└── target

├── classes

│   └── com

│   └── mycompany

│   └── app

│   └── App.class

├── maven-status

│   └── maven-compiler-plugin

│   ├── compile

│   │   └── default-compile

│   │   ├── createdFiles.lst

│   │   └── inputFiles.lst

│   └── testCompile

│   └── default-testCompile

│   ├── createdFiles.lst

│   └── inputFiles.lst

├── surefire-reports

│   ├── com.mycompany.app.AppTest.txt

│   └── TEST-com.mycompany.app.AppTest.xml

└── test-classes

└── com

└── mycompany

└── app

└── AppTest.class

27 directories, 11 files

To generate artifact:

[root@maven my-app]# mvn package

[INFO] Building jar: /opt/my\_maven\_projects/my-app/target/my-app-1.0-SNAPSHOT.jar

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 4.419 s

[INFO] Finished at: 2019-04-30T02:10:08Z

[INFO] ------------------------------------------------------------------------

tree ===> to check the folder structure

[root@maven my-app]# tree

.

├── pom.xml

├── src

│   ├── main

│   │   └── java

│   │   └── com

│   │   └── mycompany

│   │   └── app

│   │   └── App.java

│   └── test

│   └── java

│   └── com

│   └── mycompany

│   └── app

│   └── AppTest.java

└── target

├── classes

│   └── com

│   └── mycompany

│   └── app

│   └── App.class

├── maven-archiver

│   └── pom.properties

├── maven-status

│   └── maven-compiler-plugin

│   ├── compile

│   │   └── default-compile

│   │   ├── createdFiles.lst

│   │   └── inputFiles.lst

│   └── testCompile

│   └── default-testCompile

│   ├── createdFiles.lst

│   └── inputFiles.lst

├── my-app-1.0-SNAPSHOT.jar

├── surefire-reports

│   ├── com.mycompany.app.AppTest.txt

│   └── TEST-com.mycompany.app.AppTest.xml

└── test-classes

└── com

└── mycompany

└── app

└── AppTest.class

28 directories, 13 files

If you find snapshot keyword in .jar file===> tested in lower environment. Still it is in progress to develop.

If you find release keyword in .jar file===> It is ready to deploy in production environment

mvn clean ====> to clean all packages

[root@maven my-app]# mvn clean

[root@maven my-app]# mvn clean

[INFO] Scanning for projects...

[INFO]

[INFO] ----------------------< com.mycompany.app:my-app >----------------------

[INFO] Building my-app 1.0-SNAPSHOT

[INFO] --------------------------------[ jar ]---------------------------------

[INFO]

[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ my-app ---

[INFO] Deleting /opt/my\_maven\_projects/my-app/target

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 0.474 s

[INFO] Finished at: 2019-04-23T04:50:06Z

[INFO] ------------------------------------------------------------------------

[root@maven my-app]#

mvn package

It will do clean first, then compile and create all jar files.

mvn validate

===> to check/scan the pom.xml file if there are any errors

[root@maven my-app]# mvn validate

[INFO] Scanning for projects...

[INFO]

[INFO] ----------------------< com.mycompany.app:my-app >----------------------

[INFO] Building my-app 1.0-SNAPSHOT

[INFO] --------------------------------[ jar ]---------------------------------

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 0.143 s

[INFO] Finished at: 2019-04-30T02:24:15Z

[INFO] ------------------------------------------------------------------------

[root@maven my-app]#

Home exercise:

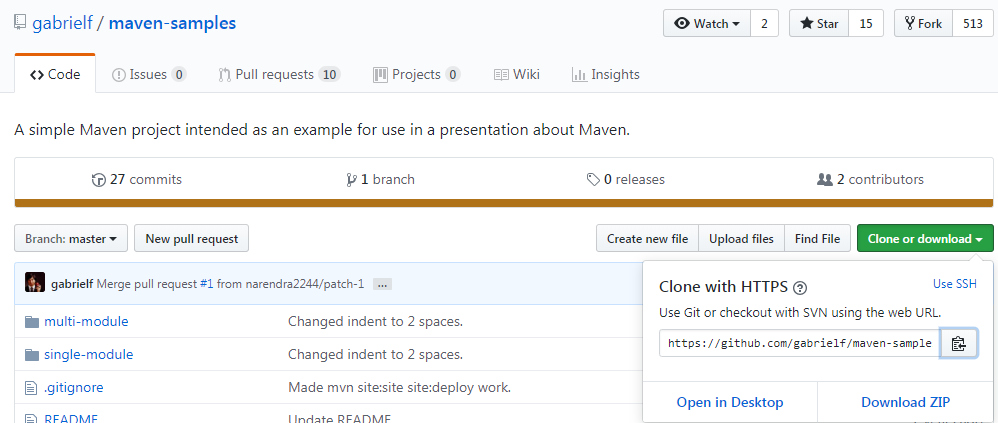
Download the source code available from github and copy in my\_mv=oven\_projects and make it Artifact.

To make artifact

Now, we are going to work on one of the source code developed by the developer and placed in public repo like github.

* Search some sample maven sample project in github

https://github.com/gabrielf/maven-samples.git



Copy the url, then go to

[root@maven my\_maven\_projects]# pwd

/opt/my\_maven\_projects

[root@maven my\_maven\_projects]# git clone https://github.com/gabrielf/maven-samples.git

[root@maven my\_maven\_projects]# ls -lrt

total 0

drwxr-xr-x. 4 root root 46 Apr 23 04:51 my-app

drwxr-xr-x. 5 root root 106 Apr 23 05:08 maven-samples

[root@maven my\_maven\_projects]# cd maven-samples/

[root@maven maven-samples]# ls -lrt

total 8

-rw-r--r--. 1 root root 227 Apr 23 05:08 README

-rw-r--r--. 1 root root 686 Apr 23 05:08 pom.xml

drwxr-xr-x. 5 root root 78 Apr 23 05:08 multi-module

drwxr-xr-x. 3 root root 50 Apr 23 05:08 single-module

[root@maven maven-samples]#

Mvn create the articaft source code.

**Note below info:**

mvn validate

mvn clean

mvn compile

mvn test

mvn package

mvn verify

mvn install

mvn deploy

maven lifecycle goals

<https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>

**A Build Lifecycle is Made Up of Phases**

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the [Lifecycle Reference](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference)):

* validate - validate the project is correct and all necessary information is available
* compile - compile the source code of the project
* test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
* package - take the compiled code and package it in its distributable format, such as a JAR.
* verify - run any checks on results of integration tests to ensure quality criteria are met
* install - install the package into the local repository, for use as a dependency in other projects locally
* deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

===========================================