Lab 3. Create and build a Java project with Maven

Objectives:

- Install Apache Maven on OL 7
- Create a Java project with the Maven command line and build this project.
- Compile and run the Project with mvn command.

Pre Requisite

- Oracle Linux 7 VM with root access
- Access to Internet

Sequence 1. Install Maven on Oracle Linux 7.

1. Start you Oracle Linux 7 Server and login as ***root***.
2. Open a Terminal Window and Install Java jdk and tree commands.

   # yum install -y java-1.8.0-openjdk-devel tree

3. Download the Apache Maven in the /tmp directory using the wget command:

   # cd /tmp

   # wget \

   https://downloads.apache.org/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.tar.gz

```
[root@server ~]# cd /tmp
[root@server tmp]# wget \
> https://downloads.apache.org/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.tar.gz
--2021-08-07 22:53:50--  https://downloads.apache.org/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 88.99.95.219, 135.181.214.104, 135.181.209.10, ...
Connecting to downloads.apache.org (downloads.apache.org)|88.99.95.219|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9536838 (9.1M) [application/x-gzip]
Saving to: ‘apache-maven-3.8.1-bin.tar.gz’

100%[=============================================================================================================>] 9,536,838

2021-08-07 22:53:53 (4.87 MB/s) - ‘apache-maven-3.8.1-bin.tar.gz’ saved [9536838/9536838]

[root@server tmp]#
```

4. Extract the downloaded archive file, and rename it using following commands.

   # tar -xf apache-maven-3.8.1-bin.tar.gz
   # mv apache-maven-3.8.1/ /opt/apache-maven/

```
[root@server tmp]# tar -xf apache-maven-3.8.1-bin.tar.gz
[root@server tmp]# mv apache-maven-3.8.1/ /opt/apache-maven/
[root@server tmp]# cd
```

5. Back to your home directory, setup environment variables. Open your text editor and create a new file ***maven.sh*** in the ***/etc/profile.d/*** directory and add environment variables to this file.

   # cd ~
   # vi /etc/profile.d/maven.sh
   export JAVA_HOME=/usr/lib/jvm/java-openjdk
   export M2_HOME=/opt/apache-maven
   export MAVEN_HOME=/opt/apache-maven

export PATH=${M2_HOME}/bin:${PATH}

```
File  Edit  View  Search  Terminal  Help
export JAVA_HOME=/usr/lib/jvm/java-openjdk
export M2_HOME=/opt/apache-maven
export MAVEN_HOME=/opt/apache-maven
export PATH=${M2_HOME}/bin:${PATH}
```

6. Save and close the file. Make the script executable by running the ***chmod*** command:

   # chmod +x /etc/profile.d/maven.sh
7. Load the environment variables using the source command:

   # source /etc/profile.d/maven.sh

```
[root@server opt]# vi /etc/profile.d/maven.sh
[root@server opt]# chmod +x /etc/profile.d/maven.sh
[root@server opt]#
```

8. To verify that Maven is installed, use the mvn -version command:
   # mvn -version

```
[root@server ~]# mvn -version
Apache Maven 3.8.1 (05c21c65bdfed0f71a2f2ada8b84da59348c4c5d)
Maven home: /opt/apache-maven
Java version: 1.8.0_302, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-1.8.0-openjdk-1.8
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.4.17-2102.203.6.el7uek.x86_64", arch: "amd64", family: "unix"
[root@server ~]#
```

Sequence 2. Generate a Project with mvn command.

In this project, we use the scaffolding functionality of Maven to create a Java project. To avoid the interactive mode, all required properties are passed directly to the command. Otherwise, Maven asks for all the required parameters.

1. Enter the following into one line in the command shell (**the backslash marks the line breaks. Remove them if you enter the entire command in one line**).

   # mvn archetype:generate -DgroupId=com.example.build.maven.java \
   -DartifactId=com.example.build.maven.java \
   -DarchetypeArtifactId=maven-archetype-quickstart \
   -DinteractiveMode=false

   With this command, Maven generates a Java project.

```
[root@server ~]# mvn archetype:generate -DgroupId=com.example.build.maven.java \
> -DartifactId=com.example.build.maven.java  \
> -DarchetypeArtifactId=maven-archetype-quickstart \
> -DinteractiveMode=false
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/m
-clean-plugin-2.5.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/ma
clean-plugin-2.5.pom (3.9 kB at 2.2 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/m
ns-22.pom
```

2. Review generated project. Install tree command and validate that Maven generated a project on your file system.

cd com.example.build.maven.java
tree

```
[root@server ~]# cd com.example.build.maven.java
[root@server com.example.build.maven.java]# tree
.
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── com
    │           └── example
    │               └── build
    │                   └── maven
    │                       └── java
    │                           └── App.java
    └── test
        └── java
            └── com
                └── example
                    └── build
                        └── maven
                            └── java
                                └── AppTest.java

15 directories, 3 files
[root@server com.example.build.maven.java]#
```

Maven created a ***App.java*** class in the *./src/main/* folder, which is just a simple "Hello World" program. It also created an example test class in *./src/test/*.

3. In the root folder there is a *pom.xml* file.

```
[root@server com.example.build.maven.java]# cat pom.xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example.build.maven.java</groupId>
  <artifactId>com.example.build.maven.java</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>com.example.build.maven.java</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
[root@server com.example.build.maven.java]#
```

4. Now you want to compile your Java sources. For this, use the following Maven command.

mvn compile

```
[root@server com.example.build.maven.java]# mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example.build.maven.java:com.example.build.maven.java >------
[INFO] Building com.example.build.maven.java 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugin
aven-resources-plugin-2.6.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins
ven-resources-plugin-2.6.pom (8.1 kB at 3.4 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugin
aven-resources-plugin-2.6.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins
ven-resources-plugin-2.6.jar (30 kB at 47 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugin
ven-compiler-plugin-3.1.pom
```

Maven now runs through all life cycle phases, which, were required by the compile phase. It resolve dependencies, loads the JUnit artifact and built the sources. It even executed the JUnit test

5. Now you want to create an executable JAR file out of our project. The package goal creates a deployable JAR file.

   # mvn package

6. Instead of running a full build with packaging, it is also possible to just run the test phases of the Maven life cycle.

   # mvn test

7. Final output of this command will be something like

```
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ com.example.build.maven.java ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.324 s
[INFO] Finished at: 2021-01-17T07:19:47-05:00
[INFO] ------------------------------------------------------------------------
[root@server com.example.build.maven.java]#
```

8. You may test the newly compiled and packaged JAR with the following command:

   # java -cp target/com.example.build.maven.java-1.0-SNAPSHOT.jar \
   com.example.build.maven.java.App

```
[root@server com.example.build.maven.java]# java -cp target/com.example.build.maven.java-1.0-SNAPSHOT.jar \
> com.example.build.maven.java.App
Hello World!
[root@server com.example.build.maven.java]#
```

9. Remove all build results / Clean the project. To clean the project and so remove the generated files in the *./target/* folder, run the following command.

   # mvn clean

10. Congratulations. You have successfully completed this lab.
11. Additional Task – Explore and Build the Employee App Downloaded in Lab 2.
    # cd ~/tsp-employee-app/
    # mvn compile
    # mvn package
    # mvn test