

What Do You Learn

- How a Developer use Maven
- Activities of a DevOps Engineer on Maven
- Build projects on Maven
- Setup and Integration of Maven with Jenkins



Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

Developer activities

- Write code in favorite IDE
- Compile the code locally
- Test it
- Create a package
- Deploy it local application server
- Push the working code on to source code repository

Maven downloads third party dependencies we just need to mention in maven file . Maven will build the application with all dependencies .

How to do?



Maven is used to download dependency , build , test ,deploy .

Topics Covered

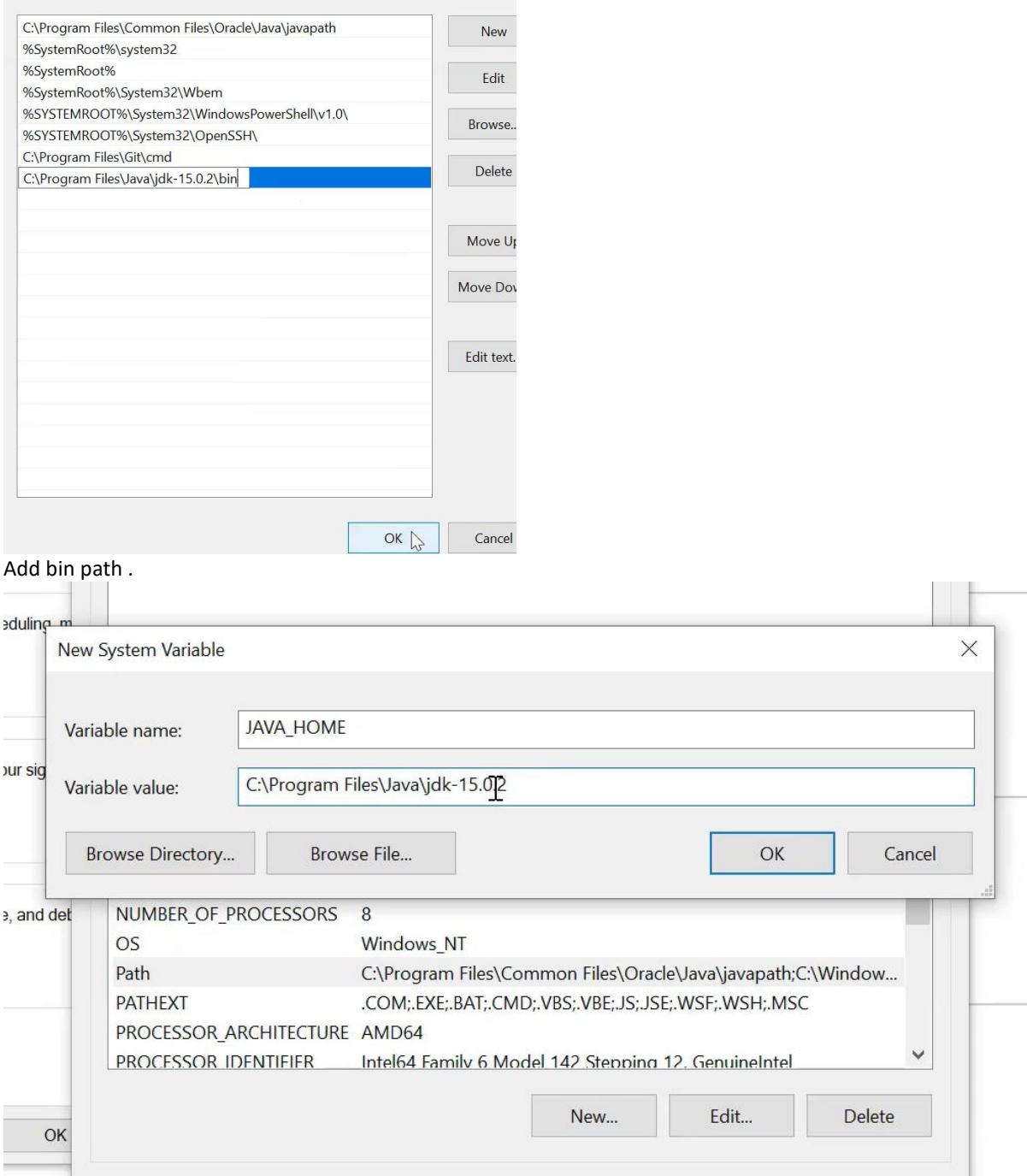
Section: Prepare Maven Environment on Windows

- Install Java
- Install Eclipse
- Create 1st Maven Project
- Maven Coordinates
- Default Directory Structure
- Steps Involved in Building a Java Project
- Maven Goals
- Build 1st Maven Project

Java jdk install > oracle.com – jdk download .

This PC > Windows (C:) > Program Files > Java > jdk-15.0.2 >		
oss	Name	Date modified
ads	bin	08-03-2021
nts	conf	08-03-2021
	include	08-03-2021
	jmods	08-03-2021
	legal	08-03-2021
edia banner	lib	08-03-2021
	COPYRIGHT	07-12-2020
	release	08-03-2021
ebsite		

Set env variable for java – so that java will be accessible from over all system .



Add system variable .

Java –version : to check java version .

Javac – compilation of java program .

Maven coordinates :

/src/main/java – actual production code .

/src/test/java – test cases .

Pom.xml – dependency for our project .

Group id , artifact id – gets stored .

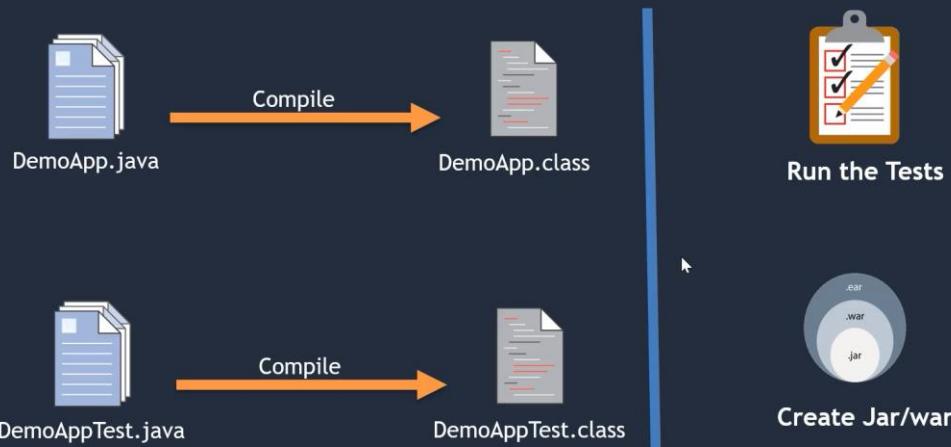
Build an Application

- Compiling source code
 - App.java → App.class
 - AppTest.java → AppTest.class
- Run the tests
- Packaging jar/ear/war files
- Deploying to server (may not be the primary responsibility of maven)

01:20 / 02:25

01:20 / 02:25

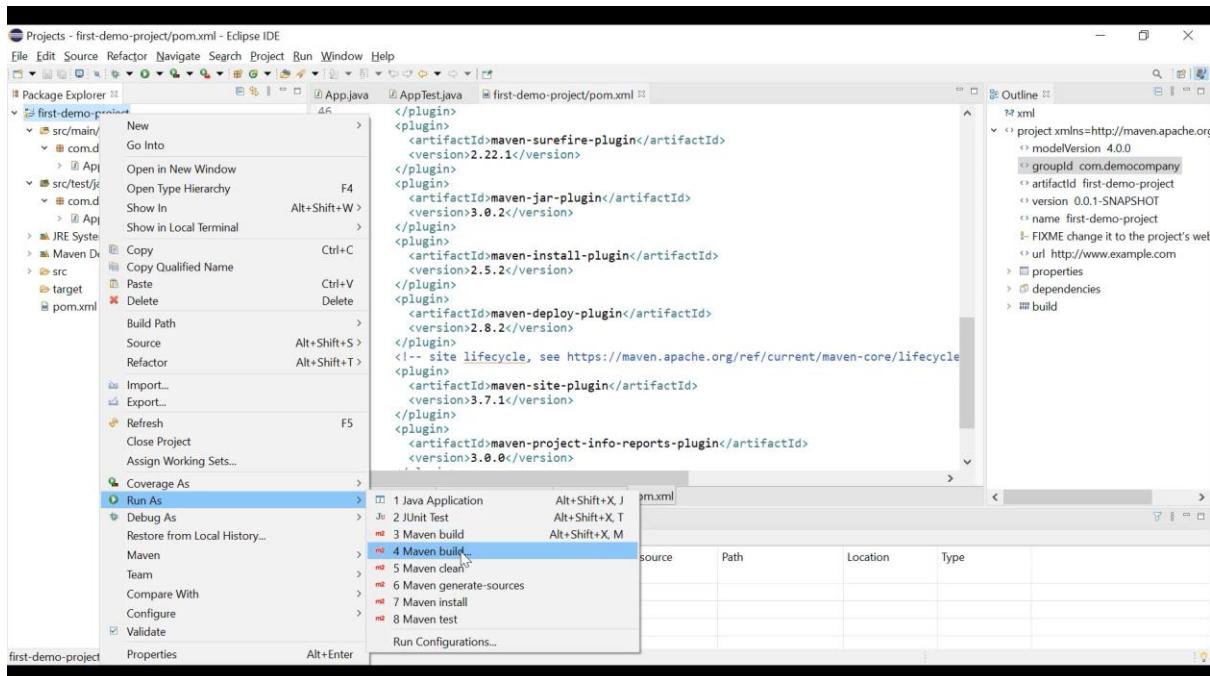
Build an Application



01:40 / 02:25

01:40 / 02:25

To build application .



We can do maven build .

To build maven project we need goals .

There are three built-in build lifecycles: default, clean and site. The `default` lifecycle handles your project deployment, the `clean` lifecycle handles project cleaning, while the `site` lifecycle handles the creation of your project's site documentation.

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](#)):

- `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.

Maven Goals

<code>clean</code>	- remove all files generated by the previous build
<code>validate</code>	- validate the project is correct and all necessary information is available.
<code>compile</code>	- compile the source code of the project.
<code>test-compile</code>	- compile the test source code into the test destination directory
<code>test</code>	- run tests using a suitable unit testing framework.
<code>package</code>	- take the compiled code and package it in its distributable format, such as a JAR.
<code>verify</code>	- run any checks to verify the package is valid and meets quality criteria.
<code>install</code>	- install the package into the local repository, for use as a dependency in other projects locally.
<code>deploy</code>	- copies the final package to the remote repository for sharing with other projects

Maven goals are executed in sequential manner . If we run any goal then previous goal will automatically be run .

The following lists all build phases of the `default`, `clean` and `site` lifecycles, which are executed in the order given up to the point of the one specified.

Clean Lifecycle

Phase	Description
<code>pre-clean</code>	execute processes needed prior to the actual project cleaning
<code>clean</code>	remove all files generated by the previous build
<code>post-clean</code>	execute processes needed to finalize the project cleaning

Default Lifecycle

Phase	Description
<code>validate</code>	validate the project is correct and all necessary information is available.
<code>initialize</code>	initialize build state, e.g. set properties or create directories.
<code>generate-sources</code>	generate any source code for inclusion in compilation.
<code>process-sources</code>	process the source code, for example to filter any values.
<code>generate-resources</code>	generate resources for inclusion in the package.
<code>process-resources</code>	copy and process the resources into the destination directory, ready for packaging.
<code>compile</code>	compile the source code of the project.

Site Lifecycle

Phase	Description
<code>pre-site</code>	execute processes needed prior to the actual project site generation
<code>site</code>	generate the project's site documentation
<code>post-site</code>	execute processes needed to finalize the site generation, and to prepare for site deployment
<code>site-deploy</code>	deploy the generated site documentation to the specified web server

If we want to create documentation of our project then we can use site lifecycle .

We can compile our source code , once compilation done our target directory will contain all class files .

If we run goal test-compile – it will validate , compile , and test , Whenever we run package we should clean as default goal if not done so it will use olders files for package .

Once package goal is successful – it will package the jar file . Package will get stored in target . artifact id is used while we package our jar file .

What is a POM?

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Examples for this is the build directory, which is `target`; the source directory, which is `src/main/java`; the test source directory, which is `src/test/java`; and so on. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

```
<groupId>com.democompany</groupId>
<artifactId>first-demo-project</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

If we don't specify any package parameter then it will treat it as jar .

```

28    </dependencies>
29
30  <build>
31    <pluginManagement><!-- lock down plugins versions to avoid
32      <plugins>
33        <!-- clean lifecycle, see https://maven.apache.org/re-
34        <plugin>
35          <artifactId>maven-clean-plugin</artifactId>
36          <version>3.1.0</version>
37        </plugin>
38        <!-- default lifecycle, jar packaging: see https://ma-
39        <plugin>
40          <artifactId>maven-resources-plugin</artifactId>
41          <version>3.0.2</version>
42        </plugin>

```

```

<name>first-demo-project</name>
<!-- FIXME change it to the project's website -->
<url>http://www.example.com</url>

<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.7</maven.compiler.source>
  <maven.compiler.target>1.7</maven.compiler.target>
</properties>

```

In above we have given name of project , compiler to be used in project .

```

13   <url>http://www.example.com</url>
14
15  <properties>
16    <project.build.sourceEncoding>UTF-8</project.build.sou
17    <maven.compiler.source>1.7</maven.compiler.source>
18    <maven.compiler.target>1.7</maven.compiler.target>
19  </properties>
20
21  <dependencies>
22    <dependency>
23      <groupId>junit</groupId>
24      <artifactId>junit</artifactId>
25      <version>4.11</version>
26      <scope>test</scope>
27    </dependency>
28  </dependencies>
29
30  <build>

```

We need to specify dependency . What all package required to build this application .

It will download junit package as dependency using maven .

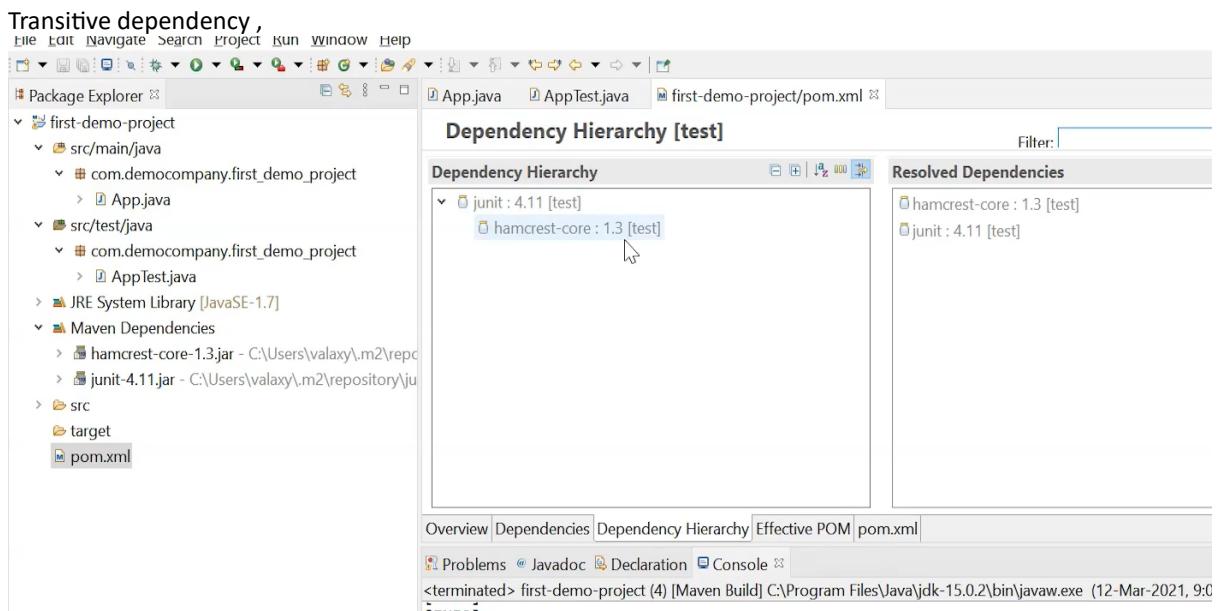
We can also add third party dependencies that can be added .

```

19   </properties>
20
21  <dependencies>
22    <dependency>
23      <groupId>junit</groupId>
24      <artifactId>junit</artifactId>
25      <version>4.11</version>
26      <scope>test</scope>
27    </dependency>
28    <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring
29<dependency>
30    <groupId>org.springframework.boot</groupId>
31    <artifactId>spring-boot-starter-test</artifactId>
32    <version>2.4.3</version>
33    <scope>test</scope>
34 </dependency>
35

```

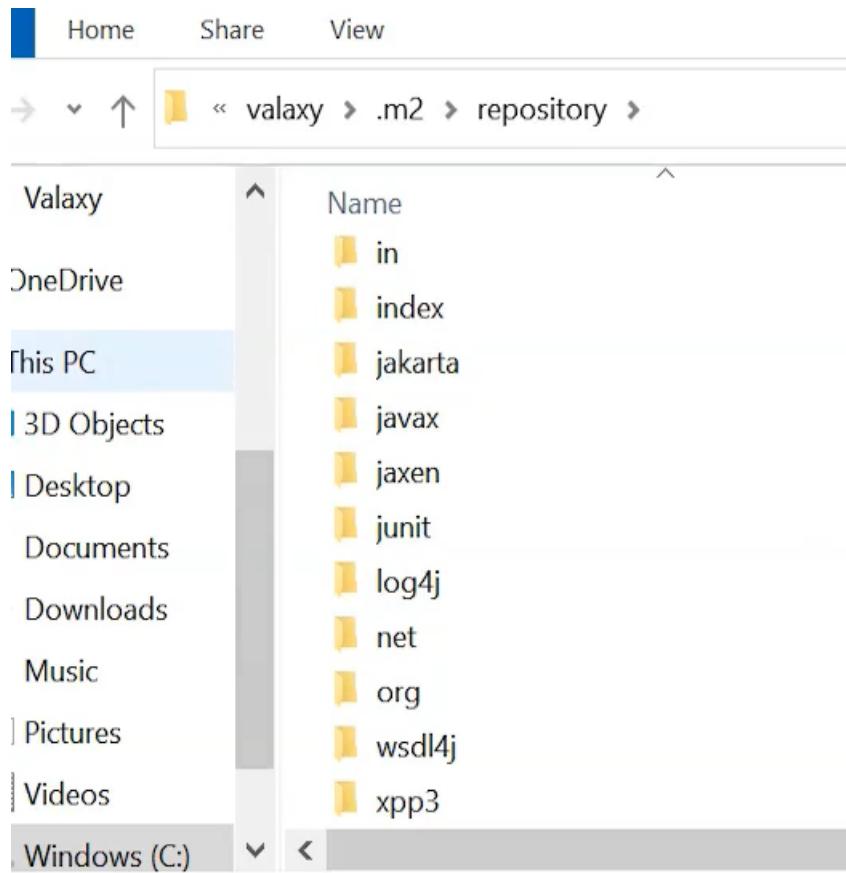
Overview Dependencies Dependency Hierarchy Effective POM pom.xml



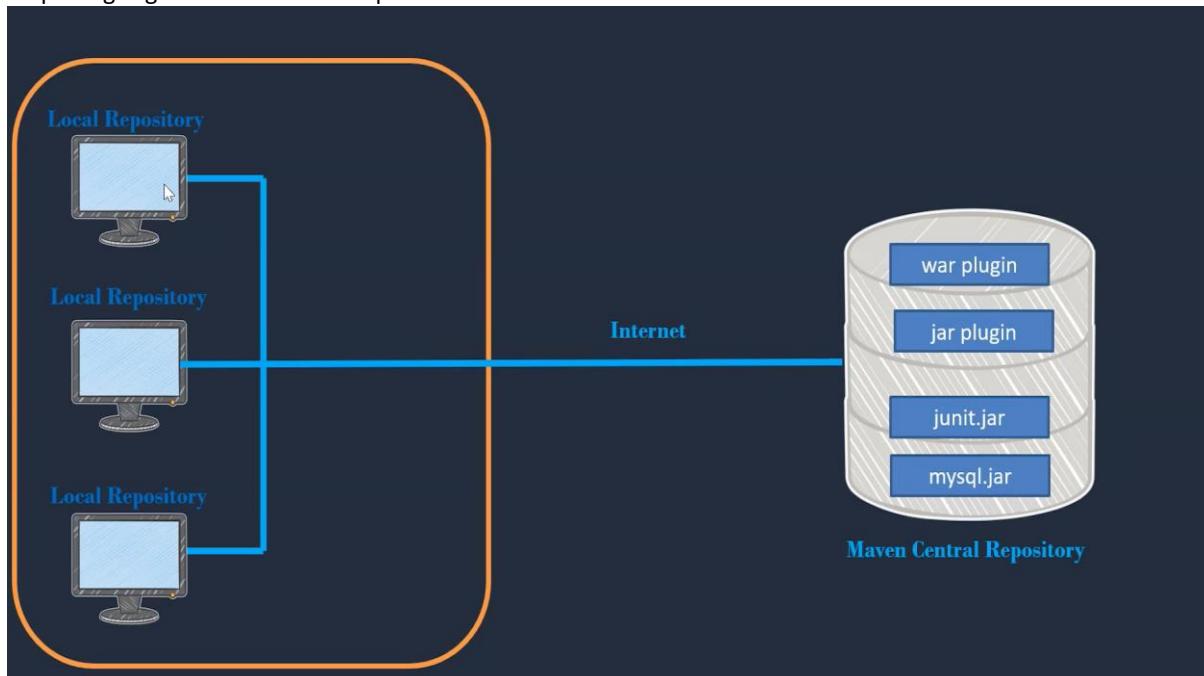
Some dependency requires other dependency this is known as transitive dependency .

Local repo and remote repo in maven :

There is local repo in our system . Whenevr we run maven project local repo gets created with .m2 .



All packages gets stored in local repo .



When we use clean goal – target repo will not be visible .

Install will run all goal .

generate-test-resources	create resources for testing.
process-test-resources	copy and process the resources into the test destination directory.
test-compile	compile the test source code into the test destination directory
process-test-classes	post-process the generated files from test compilation, for example to do bytecode enhancement on Java classes.
test	run tests using a suitable unit testing framework. These tests should not require the code be packaged or deployed.
prepare-package	perform any operations necessary to prepare a package before the actual packaging. This often results in an unpacked, processed version of the package.
package	take the compiled code and package it in its distributable format, such as a JAR.
pre-integration-test	perform actions required before integration tests are executed. This may involve things such as setting up the required environment.
integration-test	process and deploy the package if necessary into an environment where integration tests can be run.
post-integration-test	perform actions required after integration tests have been executed. This may include cleaning up the environment.
verify	run any checks to verify the package is valid and meets quality criteria.
install	install the package into the local repository, for use as a dependency in other projects locally.
deploy	done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

Deploy goal – copy

Src and pom.xml are sufficient to build the code .

Setting up maven server :

Install jdk in linux server .

Maven.apache.org – we will get url for maven tar package to download in server .

Yum install java-1.8*

Java -version

Set java home path :

Check home path for java : whereis java .

```
[root@ip-172-31-35-183 ~]# whereis java
java: /usr/bin/java /usr/lib/java /etc/java /usr/share/java /usr/share/man/man1/java.1.gz
[root@ip-172-31-35-183 ~]# find /usr/lib/jvm/java-1.8* | head -n 3
/usr/lib/jvm/java-1.8.0
/usr/lib/jvm/java-1.8.0-openjdk
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64
[root@ip-172-31-35-183 ~]#
```

Set java home path in bashrc profile .

```
[root@ip-172-31-35-183 ~]# vi ~/.bash_profile
[root@ip-172-31-35-183 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
```

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64
# User specific environment and startup programs
PATH=$PATH:$HOME/bin:$JAVA_HOME
export PATH
~
```

-- INSERT --

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

```
/root
[root@ip-172-31-35-183 ~]# cd /opt
[root@ip-172-31-35-183 opt]# wget https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
--2021-03-12 11:57:27-- https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
Resolving mirrors.estointernet.in (mirrors.estointernet.in)... 43.255.166.254, 2403:8940:3:1::f
Connecting to mirrors.estointernet.in (mirrors.estointernet.in)|43.255.166.254|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9506321 (9.1M) [application/octet-stream]
Saving to: 'apache-maven-3.6.3-bin.tar.gz'

100%[=====] 9,506,321  23.7MB/s  in 0.4s
2021-03-12 11:57:28 (23.7 MB/s) - 'apache-maven-3.6.3-bin.tar.gz' saved [9506321/9506321]

[root@ip-172-31-35-183 opt]# ls
apache-maven-3.6.3-bin.tar.gz  aws  rh
[root@ip-172-31-35-183 opt]# tar -xvzf apache-maven-3.6.3-bin.tar.gz
```

Wget is used to download the package . we unarchive the package using tar .

```
apache-maven-3.6.3-bin.tar.gz
[root@ip-172-31-35-183 opt]# ls
apache-maven-3.6.3 apache-maven-3.6.3-bin.tar.gz  aws  rh
[root@ip-172-31-35-183 opt]# cd apache-maven-3.6.3
[root@ip-172-31-35-183 apache-maven-3.6.3]# ls
bin  boot  conf  lib  LICENSE  NOTICE  README.txt
[root@ip-172-31-35-183 apache-maven-3.6.3]# pwd
/opt/apache-maven-3.6.3
[root@ip-172-31-35-183 apache-maven-3.6.3]#
```

Now we have to set maven home path also .

The terminal window shows the .bash_profile file being edited. The code sets environment variables for Java and Maven, and defines a PATH variable. The export command is highlighted.

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64
M2_HOME=/opt/apache-maven-3.6.3
M2=/opt/apache-maven-3.6.3/bin

# User specific environment and startup programs

PATH=$PATH:$HOME/bin:$JAVA_HOME:$M2:$M2_HOME
export PATH
~
```

Edit in bash_profile .

Mvn –version : to check maven version .

Install git in linux server : yum install git .

Once we clone our code we can execute maven goals in our project .

The terminal window shows the mvn validate command being run. It scans for projects, builds the first-demo-project, and concludes with a BUILD SUCCESS message. The pom.xml and src directories are listed in the directory structure.

```
[root@ip-172-31-35-183 ~]# cd first-demo-project/
[root@ip-172-31-35-183 first-demo-project]# ls
pom.xml  src
[root@ip-172-31-35-183 first-demo-project]# mvn validate
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.democompany:first-demo-project >-----
[INFO] Building first-demo-project 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time:  0.215 s
[INFO] Finished at: 2021-03-13T00:19:56Z
[INFO]
[INFO] -----
[root@ip-172-31-35-183 first-demo-project]# ls
pom.xml  src
[root@ip-172-31-35-183 first-demo-project]#
```

Mvn compile – it will compile the code and create jar file .

The terminal window shows the mvn compile command being run. It lists the pom.xml and src directories and creates a target directory.

```
[INFO] -----
[root@ip-172-31-35-183 first-demo-project]# ls
pom.xml  src  target
[root@ip-172-31-35-183 first-demo-project]#
```

Target gets created when we compile the code .

Mvn test-compile – along with source code test cases will also be compiled .

The terminal window shows the mvn test-compile command being run. It scans for projects, builds the first-demo-project, and downloads JUnit dependencies from central repository. Progress bar indicates the download progress.

```
[root@ip-172-31-35-183 first-demo-project]# mvn test-compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.democompany:first-demo-project >-----
[INFO] Building first-demo-project 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/junit/junit/4.11/junit
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/hamcrest/hamcrest-
Progress (2): 112/245 kB | 45 kB
```

```
complete:  
[root@ip-172-31-35-183 target]# tree test-classes  
test-classes  
└── com  
    └── democompany  
        └── first_demo_project  
            └── AppTest.class  
  
3 directories, 1 file
```

To view directory structure : tree

mvn test : to test source code .

mvn package : to package our artifact .

mvn install install package in local repo .

lets find where our local repo for maven is present .

```
[INFO]  
[root@ip-172-31-35-183 first-demo-project]# find / -name .m2  
/root/.m2  
[root@ip-172-31-35-183 first-demo-project]#
```

The command "find / -name .m2" searches for a directory named ".m2" starting from the root directory ("/") in a Linux or Unix-like operating system. This command is commonly used to locate the ".m2" directory, which is a default directory used by Apache Maven for storing local repository data.

mvn install – install packages in local repo in com directory .

```
pom.xml  src  target  
[root@ip-172-31-35-183 first-demo-project]# mvn install  
[INFO] Scanning for projects...  
[INFO]  
[INFO] -----< com.democompany:first-demo-project >-----  
[INFO] Building first-demo-project 0.0.1-SNAPSHOT  
[INFO] -----[ jar ]-----  
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-jar-plugin/3.1.1/maven-jar-plugin-3.1.1.pom  
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-jar-plugin/3.1.1/maven-jar-plugin-3.1.1.pom (13 kB at 1 kB/s)
```

```
first-demo-project second-demo-project
[root@ip-172-31-35-183 ~]# cd second-demo-project/
[root@ip-172-31-35-183 second-demo-project]# ls
pom.xml  src
[root@ip-172-31-35-183 second-demo-project]# tree .
.
└── pom.xml
    └── src
        └── main
            └── webapp
                ├── index.jsp
                └── WEB-INF
                    └── web.xml
```

4 directories, 3 files

```
[root@ip-172-31-35-183 second-demo-project]#
```

```
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

Pom.xml version that is 4.0.0 ,

```
<groupId>com.democompaly</groupId>
<artifactId>second-demo-project</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>war</packaging>

<name>second-demo-project Maven Webapp</name>
<!-- FIXME change it to the project's website --&gt;
&lt;url&gt;http://www.example.com&lt;/url&gt;</pre>
```

Name and url of project .

Build plugin – which goal we will execute all plugins are mentioned .

Junit dependency is necessary only if we have test cases in our code .

If test cases are not mentioned then we can remove dependency .

Junit dependency .

Setting.xml in maven :

Pom.xml is associated to specific project . setting.xml is global configuration .

The `settings` element in the `settings.xml` file contains elements used to define values which configure Maven execution in various ways, like the `pom.xml`, but should not be bundled to any specific project, or distributed to an audience. These include values such as the local repository location, alternate remote repository servers, and authentication information.

There are two locations where a `settings.xml` file may live:

- The Maven install: `#{maven.home}/conf/settings.xml`
- A user's install: `#{user.home}/.m2/settings.xml`

The screenshot shows the Maven Settings Reference page with the URL maven.apache.org/settings.html. The left sidebar contains links like 'Plugin Testing', 'Plugin Tools', 'Resource Bundles', 'SCM', 'Shared Components', 'Skins', 'Surefire', 'Wagon', 'ASF', 'How Apache Works', 'Foundation', 'Sponsoring Apache', and 'Thanks'. The main content area is titled 'Plugin Groups' and describes how it contains a list of `<pluginGroup>` elements. Below this is a code snippet example:

```

1. <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2.   xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 https://maven.apache.org/xsd/settings-1.0.0.xsd"
3. ...
4. <pluginGroups>
5.   <pluginGroup>org.eclipse.jetty</pluginGroup>
6. </pluginGroups>
7. ...
8. </settings>

```

Below the code snippet, there is a note: "For example, given the above settings the Maven command line may execute `org.eclipse.jetty:jetty-maven-plugin:run` with the truncated command:" followed by a truncated command line entry.

We will deploy application from maven to tomcat .

Deploy an application on tomcat using maven .

https://github.com/ravdy/maven/blob/master/tomcat_installation.MD

The screenshot shows a terminal session in MobaXterm. The title bar says "13.233.2.118 (ec2-user)". The terminal window displays the following command-line session:

```

[root@ip-172-31-42-165 ~]# cd /opt
[root@ip-172-31-42-165 opt]# ls
apache-tomcat-8.5.58 apache-tomcat-8.5.58.tar.gz aws comasetest containerd docker-compose-lamp nginx nodejs project rh
[root@ip-172-31-42-165 opt]# cd apache-tomcat-8.5.58
[root@ip-172-31-42-165 apache-tomcat-8.5.58]# ls
bin BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps work
[root@ip-172-31-42-165 apache-tomcat-8.5.58]# cat conf/tomcat-users.xml

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Lets deploy our application to /opt/tomcat/webapp/.

https://github.com/ravdy/maven/blob/master/deploy_warfile_on_tomcat_server_using_maven.md

We will deploy second app

```

[root@maven ~]#
[root@maven ~]# pwd
/root
[root@maven ~]# ls
first-demo-project second-demo-project
[root@maven ~]# cd second-demo-project/
[root@maven second-demo-project]# ls
pom.xml src target
[root@maven second-demo-project]# vi

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Add tomcat plugin to pom.xml ,

```

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
</properties>

<build>
    <finalName>second-demo-project</finalName>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <version>2.2</version>
                <configuration>
                    <url>http://13.233.2.118:8080/manager/text</url>
                    <server>TomcatServer</server>
                    <path>/second-maven-project</path>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-clean-plugin</artifactId>
                <version>3.1.0</version>
            </plugin>
        </plugins>
    </pluginManagement>
</build>

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Lets create settings.xml ,

The screenshot shows a terminal window in MobaXterm with two tabs open. The left tab is titled '15.206.94.219 (ec2-user)' and contains the following XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings ...>
    <servers>
        <server>
            <id>TomcatServer</id>
            <username>admin</username>
            <password>admin</password>
        </server>
    </servers>
</settings>
```

The right tab is titled '5.13.233.2.118 (ec2-user)'. A vertical sidebar on the left has icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, and Help. A status bar at the bottom says 'UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net'.

Lets deploy application :

The screenshot shows a terminal window in MobaXterm with two tabs open. The left tab is titled '15.206.94.219 (ec2-user)' and contains the following Maven deployment configuration:

```
<!!-- FIXME change it to the project's website -->
<url>http://www.example.com</url>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
</properties>

<build>
    <finalName>second-demo-project</finalName>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <version>2.2</version>
                <configuration>
                    <url>http://13.233.2.118:8080/manager/text</url>
                    <server>TomcatServer</server>
                    <path>/second-demo-project</path>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-clean-plugin</artifactId>
                <version>3.1.0</version>
            </plugin>
        </plugins>
    </pluginManagement>
</build>
```

The right tab is titled '5.13.233.2.118 (ec2-user)'. A vertical sidebar on the left has icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, and Help. A status bar at the bottom says '04:53 / 09:40 UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net'.

Path should be same as our project name .

Mvn tomcat7:deploy to deploy our tomcat application .

Once deploy done we would be able to see our application deployed in webapps .

```

13.233.2.118 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect... 4 15.206.94.219 (ec2-user) 5 13.233.2.118 (ec2-user)
ciphers.bat configtest.sh setclasspath.sh tomcat-juli.jar version.sh
[root@tomcat bin]# cd ..
[root@tomcat apache-tomcat-8.5.58]# pwd
/opt/apache-tomcat-8.5.58
[root@tomcat apache-tomcat-8.5.58]# ls -l
total 128
drwxr-x--- 2 root root 4096 Sep 19 11:15 bin
-rw-r----- 1 root root 19318 Sep 10 2020 BUILDING.txt
drwx----- 3 root root 254 Sep 19 11:19 conf
-rw-r----- 1 root root 5408 Sep 10 2020 CONTRIBUTING.md
drwxr-x--- 2 root root 4096 Sep 19 11:15 lib
-rw-r----- 1 root root 57011 Sep 10 2020 LICENSE
drwxr-x--- 2 root root 4096 Mar 13 05:06 logs
-rw-r----- 1 root root 1726 Sep 10 2020 NOTICE
-rw-r----- 1 root root 3257 Sep 10 2020 README.md
-rw-r----- 1 root root 7136 Sep 10 2020 RELEASE-NOTES
-rw-r----- 1 root root 16262 Sep 10 2020 RUNNING.txt
drwxr-x--- 2 root root 30 Sep 19 11:15 temp
drwxr-x--- 7 root root 81 Mar 13 06:19 webapps
drwxr-x--- 3 root root 22 Sep 19 11:19 work
[root@tomcat apache-tomcat-8.5.58]# cd webapps
[root@tomcat webapps]# ls
docs examples host-manager manager ROOT
[root@tomcat webapps]# pwd
/opt/apache-tomcat-8.5.58/webapps
[root@tomcat webapps]# ls
docs examples host-manager manager ROOT second-demo-project second-demo-project.war
[root@tomcat webapps]# [REDACTED]
06.03 / 09:40 UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
15.206.94.219 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect... 4 15.206.94.219 (ec2-user) 5 13.233.2.118 (ec2-user)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/reporting/maven-reporting-api/2.2.1/maven-reporting-api-2.2.1.jar (9.8 kB at 2.9 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/slf4j/jcl-over-slf4j/1.7.5/jcl-over-slf4j-1.7.5.jar
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/doxia/doxia-sink-api/1.1/doxia-sink-api-1.1.jar (13 kB at 3.7 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/slf4j/slf4j-api/1.7.5/slf4j-api-1.7.5.jar
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-logging-api/1.1/doxia-logging-api-1.1.jar (11 kB at 3.2 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/sonatype/plexus/plexus-build-api/0.0.4/plexus-build-api-0.0.4.jar (6.8 kB at 1.9 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interactivity-api/1.0-alpha-4/plexus-interactivity-api-1.0-alpha-4.jar (13 kB at 3.8 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/slf4j/jcl-over-slf4j/1.7.5/jcl-over-slf4j-1.7.5.jar (17 kB at 4.6 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/slf4j/slf4j-api/1.7.5/slf4j-api-1.7.5.jar (26 kB at 7.0 kB/s)
[INFO] Deploying war to http://13.233.2.118:8080/second-demo-project
Uploading: http://13.233.2.118:8080/manager/text/deploy?path=%2Fsecond-demo-project
Uploaded: http://13.233.2.118:8080/manager/text/deploy?path=%2Fsecond-demo-project (3 kB at 2142.6 kB/sec)

[INFO] tomcatManager status code:200, ReasonPhrase:
[INFO] OK - Deployed application at context path [/second-demo-project]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 37.786 s
[INFO] Finished at: 2021-03-13T06:29:37Z
[INFO] -----
[root@maven second-demo-project]# [REDACTED]
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```

In maven server :

```

[root@maven second-demo-project]# tree
.
+-- pom.xml
+-- src
|   +-- main
|   |   +-- webapp
|   |   |   +-- index.jsp
|   |   |   +-- WEB-INF
|   |   |       +-- web.xml
+-- target
    +-- maven-archiver
    |   +-- pom.properties
    +-- second-demo-project
        +-- index.jsp
        +-- META-INF
        +-- WEB-INF
            +-- classes
            +-- web.xml
    +-- second-demo-project.war

10 directories, 7 files
[root@maven second-demo-project]# cat src/main/webapp/index.jsp
<html>
<body>
<h2>Hello World!</h2>
</body>
</html>

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Same application is deployed in index.jsp .

If we edit index.jsp then we have to again deploy the application .

```

</html>
[root@maven second-demo-project]# vi src/main/webapp/index.jsp
[root@maven second-demo-project]# mvn tomcat7:deploy

```

Add maven build server as an agent to Jenkins .

Name	Version	Released
Maven Integration	3.10	8 days 7 hr ago
Maven Info	0.2.0	7 yr 0 mo ago
Maven Invoker	2.4	2 yr 1 mo ago

install maven invoker and maven integration plugin .

We are adding maven build server as a slave to Jenkins .

System Configuration

istory

e Jenkins

NS

le Resources

ew



Configure System

Configure global settings and paths.



Manage Nodes and Clouds

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



Back to Dashboard



Manage Jenkins



New Node



Configure Clouds



Node Monitoring

BUILD QUEUES

Node name

maven_build_server

Permanent Agent

Adds a plain, permanent agent to Jenkins agents, such as dynamic provisioning. See computer, virtual machines managed ou



Executor – number of parallel build it can run .

For remote root directory as a best practice create new user on maven system .

```
[root@maven second-demo-project]# useradd jenkins
[root@maven second-demo-project]# passwd jenkins
Changing password for user jenkins.
New password:
BAD PASSWORD: The password contains the user name in some form
Retype new password:
passwd: all authentication tokens updated successfully.
[root@maven second-demo-project]#
```

Also add this user to sudo file .

Visudo – added Jenkins as sudo user .

```
## Allow root to run any commands anywhere
root    ALL=(ALL)          ALL
jenkins ALL=(ALL)          NOPASSWD: ALL
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)          ALL

## Same thing without a password
# %wheel      ALL=(ALL)      NOPASSWD: ALL
## Allows members of the users group to mount and unmount the
## cdrom as root
# %users   ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users   localhost=/sbin/shutdown -h now
```

By adding Jenkins as above we are telling while executing sudo don't ask for password .

We need to enable password based connection for user .

Edit /etc/ssh/sshd_config

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
# PasswordAuthentication no
```

Once password authentication enabled , reload the server – service sshd reload

We have enabled password based authentication on maven server . Now use Jenkins to connect to maven server .

Adding maven as slave to Jenkins .

Remote root directory - /home/Jenkins

Launch method

Launch agent by connecting it to the master

Launch agent by connecting it to the master

Launch agent via execution of command on the master

Launch agents via SSH

Custom WORKDIR path

In host – maven server ip , we have to add creds also .

usage

Use this node as much as possible

Launch method

Launch agents via SSH

Host

172.31.35.183

Credentials

jenkins/*********

Add ▾

Host Key Verification Strategy

Known hosts file Verification Strategy

Availability

Verification strategy – select non verifying

Dashboard > Nodes > maven_build_server

```

XDG_RUNTIME_DIR=/run/user/1001
XDG_SESSION_ID=306
_=etc/bashrc
[03/13/21 07:20:26] [SSH] Checking java version of /home/jenkins/jdk/bin/java
Couldn't figure out the Java version of /home/jenkins/jdk/bin/java
bash: /home/jenkins/jdk/bin/java: No such file or directory

[03/13/21 07:20:26] [SSH] Checking java version of java
[03/13/21 07:20:27] [SSH] java -version returned 1.8.0_272.
[03/13/21 07:20:27] [SSH] Starting sftp client.
[03/13/21 07:20:27] [SSH] Copying latest remoting.jar...
[03/13/21 07:20:27] [SSH] Copied 1,506,923 bytes.
Expanded the channel window size to 4MB
[03/13/21 07:20:27] [SSH] Starting agent process: cd "/home/jenkins" && java -jar remoting.jar -w
cache /home/jenkins/remoting/jarCache
Mar 13, 2021 7:20:27 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/jenkins/remoting as a remoting work directory
Mar 13, 2021 7:20:27 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/jenkins/remoting
<===[JENKINS REMOTING CAPACITY]==>channel started
Remoting version: 4.6
This is a Unix agent
Evacuated stdout
Agent successfully connected and online

```

Now agent will connect to maven server .

New item > maven project > restrict to run on maven server only .

- Restrict where this project can be run

Label Expression

maven build server

Label maven build server matches 1 node. Permissions or othe

Add git url to build section .

Build

Maven Version

Jenkins needs to know where your Maven is installed.
Please do so from the tool configuration.

Root POM

pom.xml



Goals and options

clean install



Advanced...

For above error we have to add in global tool configuration .

 Jenkins

Dashboard >

- New Item
- People
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins**
- My Views
- Lockable Resources

Manage Jenkins

System Configuration

 **Configure System**
Configure global settings and paths.

 **Global Tool Configuration**
Configure [tools](#), their locations and automatic installations.

 **Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

```
/usr/local/bin./usr/bin./usr/local/sbin./usr/sbin./home/jenkins/.local/bin/maven [jenkins@maven ~]$ sudo cat /root/.bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64
M2_HOME=/opt/apache-maven-3.6.3
M2=/opt/apache-maven-3.6.3/bin

# User specific environment and startup programs

PATH=$PATH:$HOME/bin:$JAVA_HOME:$M2:$M2_HOME

export PATH
[jenkins@maven ~]$
```

Add JAVA_HOME path

JDK

Name

JAVA_1.8

JAVA_HOME

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64

Git

Git installations

 **Git**

Name

Path to Git executable


```
export PATH
[jenkins@maven ~]$ whereis git
git: /usr/bin/git /usr/share/man/man1/git.1.gz
[jenkins@maven ~]$ cd /opt
[jenkins@maven opt]$ ls
apache-maven-3.6.3 apache-maven-3.6.3-bin.tar.gz aws jenkins rh slave-agent.jnl
[jenkins@maven opt]$ cd apache-maven-3.6.3
[jenkins@maven apache-maven-3.6.3]$ pwd
/opt/apache-maven-3.6.3
[jenkins@maven apache-maven-3.6.3]$
```

Add maven path also to Jenkins .

```
first-demo-project
[jenkins@maven workspace]$ cd first-demo-project
[jenkins@maven first-demo-project]$ ls
pom.xml src target
[jenkins@maven first-demo-project]$
```

Once we run pipeline our project will be run in workspace folder and target folder will be created as compile goal is also run .

Build war file on our maven build server node .

<https://github.com/ravdy/second-demo-project.git>

New item > maven project > restrict to run in maven build server > add git repo >

Build

Root POM

Goals and options

Build the pipeline .

The screenshot shows a MobaXterm window with two sessions. The left session (port 22) displays a terminal window with the following commands and output:

```
[root@maven jenkins]# cd workspace/
[root@maven workspace]# ls
first-demo-project second-demo-project
[root@maven workspace]# cd second-demo-project/
[root@maven second-demo-project]# ls
pom.xml src target
[root@maven second-demo-project]# tree
.
+-- pom.xml
+-- src
|   +-- main
|   |   +-- webapp
|   |   |   +-- index.jsp
|   |   |   +-- WEB-INF
|   |   |   +-- web.xml
+-- target
    +-- maven-archiver
    |   +-- pom.properties
    +-- second-demo-project
        +-- index.jsp
        +-- META-INF
        +-- WEB-INF
        |   +-- classes
        |   +-- web.xml
    +-- second-demo-project.war
10 directories, 7 files
[root@maven second-demo-project]#
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Now the war file is deployed for second project .

Sonarqube installation :

<https://github.com/ravdy/DevOps/tree/master/sonarqube>

1. A small-scale (individual or small team) instance of the SonarQube server requires at least 2GB of RAM to run efficiently and 1GB of free RAM for the OS. If you are installing an instance for a large teams or Enterprise, please consider the additional recommendations below.

SonarQube is an open-source static testing analysis software, it is used by developers to manage source code quality and consistency.

Prerequisites

Source: <https://docs.sonarqube.org/latest/requirements/requirements/>

1. An EC2 instance with a minimum of 2 GB RAM (t2.small)
2. Java 11 installation

```
amazon-linux-extras list  
amazon-linux-extras install java-openjdk11
```

3. SonarQube cannot be run as root on Unix-based systems, so create a dedicated user account for SonarQube if necessary.

Installation steps

1. Download SonarQube latest verions on to EC2 instace

```
cd /opt  
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-x.x.zip
```

2. extract packages

```
unzip /opt/sonarqube-x.x.zip
```

3. Change ownership to the user and Switch to Linux binaries directory to start service

```
chown -R <sonar_user>:<sonar_user_group> /opt/sonarqube-x.x  
cd /opt/sonarqube-x.x/bin/linux-x86-64  
.sonar.sh start  
[root@ip-172-31-2-255 ~]# java -version  
-bash: java: command not found  
[root@ip-172-31-2-255 ~]# yum list | grep openjdk  
java-1.7.0-openjdk.x86_64           1:1.7.0.261-2.6.22.2.amzn2.0.1 amzn  
java-1.7.0-openjdk-accessibility.x86_64  
java-1.7.0-openjdk-demo.x86_64      1:1.7.0.261-2.6.22.2.amzn2.0.1 amzn  
java-1.7.0-openjdk-devel.x86_64     1:1.7.0.261-2.6.22.2.amzn2.0.1 amzn  
java-1.7.0-openjdk-headless.x86_64   1:1.7.0.261-2.6.22.2.amzn2.0.1 amzn  
java-1.7.0-openjdk-javadoc.noarch    1:1.7.0.261-2.6.22.2.amzn2.0.1 amzn  
java-1.7.0-openjdk-src.x86_64       1:1.7.0.261-2.6.22.2.amzn2.0.1 amzn  
java-1.8.0-openjdk.x86_64          1:1.8.0.302.b08-0.amzn2.0.1 amzn  
java-1.8.0-openjdk-accessibility.x86_64  
java-1.8.0-openjdk-accessibility-debug.x86_64  
java-1.8.0-openjdk-debug.x86_64      1:1.8.0.302.b08-0.amzn2.0.1 amzn  
java-1.8.0-openjdk-demo.x86_64      1:1.8.0.302.b08-0.amzn2.0.1 amzn  
java-1.8.0-openjdk-demo-debug.x86_64 1:1.8.0.302.b08-0.amzn2.0.1 amzn  
java-1.8.0-openjdk-devel.x86_64     1:1.8.0.302.b08-0.amzn2.0.1 amzn  
java-1.8.0-openjdk-devel-debug.x86_64 1:1.8.0.302.b08-0.amzn2.0.1 amzn  
java-1.8.0-openjdk-headless.x86_64   1:1.8.0.302.b08-0.amzn2.0.1 amzn  
java-1.8.0-openjdk-headless-debug.x86_64
```

Install sonarqube binary package ,

```
[root@ip-172-31-2-255 ~]# cd /opt
[root@ip-172-31-2-255 opt]# ll
total 0
drwxr-xr-x 4 root root 33 Aug 24 19:06 aws
drwxr-xr-x 2 root root 6 Aug 16 2018 rh
[root@ip-172-31-2-255 opt]# wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.9.2.46101.zip
--2021-09-07 07:45:32-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.9.2.46101.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 91.134.125.245
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|91.134.125.245|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 276400757 (264M) [application/zip]
Saving to: 'sonarqube-8.9.2.46101.zip'

100%[=====] 276,400,757 11.5MB/s

2021-09-07 07:45:57 (11.2 MB/s) - 'sonarqube-8.9.2.46101.zip' saved [276400757/276400757]

[root@ip-172-31-2-255 opt]# ll
total 269924
drwxr-xr-x 4 root root 33 Aug 24 19:06 aws
drwxr-xr-x 2 root root 6 Aug 16 2018 rh
-rw-r--r-- 1 root root 276400757 Jul 27 07:55 sonarqube-8.9.2.46101.zip
[root@ip-172-31-2-255 opt]#
```

Unzip sonarqube.zip

```
 -rw-r--r-- 1 root root 276400757 Jul 27 07:55 sonarqube-8.9.2.46101
 [root@ip-172-31-2-255 opt]# cd sonarqube-8.9.2.46101/
 [root@ip-172-31-2-255 sonarqube-8.9.2.46101]# ll
total 12
drwxr-xr-x 6 root root 94 Jul 27 06:27 bin
drwxr-xr-x 2 root root 50 Jul 27 06:27 conf
-rw-r--r-- 1 root root 7651 Jul 27 06:27 COPYING
drwxr-xr-x 2 root root 24 Jul 27 06:27 data
drwxr-xr-x 7 root root 132 Jul 27 06:27 elasticsearch
drwxr-xr-x 4 root root 40 Jul 27 06:27 extensions
drwxr-xr-x 6 root root 143 Jul 27 06:41 lib
drwxr-xr-x 2 root root 24 Jul 27 06:27 logs
drwxr-xr-x 2 root root 24 Jul 27 06:27 temp
drwxr-xr-x 6 root root 4096 Jul 27 06:41 web
[root@ip-172-31-2-255 sonarqube-8.9.2.46101]# cd conf
[root@ip-172-31-2-255 conf]# l
-bash: l: command not found
[root@ip-172-31-2-255 conf]# ll
total 28
-rw-r--r-- 1 root root 20529 Jul 27 06:27 sonar.properties
-rw-r--r-- 1 root root 3315 Jul 27 06:27 wrapper.conf
[root@ip-172-31-2-255 conf]# cat sonar.properties
```

We can update sonar.properties as per our requirement .

```
# The schema must be created first.
#sonar.jdbc.username=
#sonar.jdbc.password=
```

To connect to database we need to specify .

Elastic serach work with non root user so we need to start sonaqrube server from sonarqube new user .

```
usage: ./sonar.sh { console | start | stop | force-stop }
[root@ip-172-31-2-255 linux-x86-64]# ./sonar.sh start
Starting SonarQube...
Started SonarQube.
[root@ip-172-31-2-255 linux-x86-64]# ./sonar.sh status
SonarQube is running (3605).
[root@ip-172-31-2-255 linux-x86-64]# ./sonar.sh status
SonarQube is not running.
See: /opt/sonarqube-8.9.2.46101/bin/sonar
uncaught exception in thread [main]
java.lang.RuntimeException: can not run elasticsearch as root
    at org.elasticsearch.bootstrap.Bootstrap.initializeNatives(Bootstrap.java:101)
    at org.elasticsearch.bootstrap.Bootstrap.setup(Bootstrap.java:168)
    at org.elasticsearch.bootstrap.Bootstrap.init(Bootstrap.java:397)
    at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:159)
```

So we need to run sonarqube from non root user.

```
[root@ip-172-31-2-255 logs]# useradd sonaradmin
[root@ip-172-31-2-255 logs]# chown -R sonaradmin:sonaradmin /opt/sonarqube-8.9.2.46101
[root@ip-172-31-2-255 logs]# pwd
/opt/sonarqube-8.9.2.46101/logs
[root@ip-172-31-2-255 logs]# cd ..
[root@ip-172-31-2-255 sonarqube-8.9.2.46101]# ll
total 12
drwxr-xr-x 6 sonaradmin sonaradmin   94 Jul 27 06:27 bin
drwxr-xr-x 2 sonaradmin sonaradmin   50 Jul 27 06:27 conf
-rw-r--r-- 1 sonaradmin sonaradmin 7651 Jul 27 06:27 COPYING
drwxr-xr-x 2 sonaradmin sonaradmin   24 Jul 27 06:27 data
drwxr-xr-x 7 sonaradmin sonaradmin 132 Jul 27 06:27 elasticsearch
drwxr-xr-x 4 sonaradmin sonaradmin   40 Jul 27 06:27 extensions
drwxr-xr-x 6 sonaradmin sonaradmin 143 Jul 27 06:41 lib
drwxr-xr-x 2 sonaradmin sonaradmin   64 Sep  7 07:49 logs
drwxr-xr-x 3 sonaradmin sonaradmin   38 Sep  7 07:48 temp
drwxr-xr-x 6 sonaradmin sonaradmin 4096 Jul 27 06:41 web
[root@ip-172-31-2-255 sonarqube-8.9.2.46101]# su - so
```

Integration of sonarqube with Jenkins .

https://github.com/ravdy/DevOps/blob/master/sonarqube/integrate_sonar_with_jenkins.md

1. a Sonarqube Server [Click here to Setup Sonarqube server](#)
2. A Jenkins server [Click here to set up a Jenkins server](#)

Integration Steps

On Sonarqube server

1. Generate a sonarqube token to authenticate from Jenkins

On Jenkins server

1. Install Sonarqube plugin
2. Configure Sonarqube credentials
3. Add Sonarqube to jenkins "configure system"
4. Install SonarScanner
5. Run Pipeline job

🧹 CleanUp

Stop sonarqube services and delete the instance Stop Jenkins service and delete the instance

🔗 My Profile



Sonarscanner will run analysis on our project .

Pipeline for sonarqube scan .

<https://github.com/ravdy/DevOps/blob/master/sonarqube/Jenkinsfile>

[Code](#)[Blame](#) 29 lines (28 loc) · 721 Bytes

```
1 pipeline{
2     agent any
3     environment {
4         PATH = "$PATH:/opt/apache-maven-3.8.2/bin"
5     }
6     stages{
7         stage('GetCode'){
8             steps{
9                 git 'https://github.com/ravdy/javaloginapp.git'
10            }
11        }
12        stage('Build'){
13            steps{
14                sh 'mvn clean package'
15            }
16        }
17        stage('SonarQube analysis') {
18            def scannerHome = tool 'SonarScanner 4.0';
19            steps{
20                withSonarQubeEnv('sonarqube-8.9') {
21                    // If you have configured more than one global server connection, you can specify its name
22                    sh "${scannerHome}/bin/sonar-scanner"
23                    sh "mvn sonar:sonar"
24                }
25            }
26        }
27    }
28 }
29 }
```

Dashboard ➔ Update Center

[Back to Dashboard](#)[Manage Jenkins](#)[Manage Plugins](#)

Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

SonarQube Scanner

 Installing

Loading plugin extensions

 Pending[Go back to the top page](#)

(you can start using the installed plugins right away)

 Restart Jenkins when installation is complete and no jobs are running

Install sonarqube scanner plugin .

 Back to credential domains

 Add Credentials

Kind

Secret text

Scope

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID

I

Description

Add token as creds for sonarqube .

Manage Jenkins > configure system > add sonarqube server to Jenkins >

SonarQube servers

- Environment variables Enable injection of SonarQube server configuration as build environment variables
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name

sonarqube-8.9.2

Server URL

http://13.127.91.144:9000

Default is http://localhost:9000

Server authentication token

sonarqube-token ▾

 Add ▾

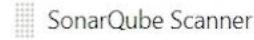


SonarQube authentication token. Mandatory when anonymous access is disabled.

List of SonarQube Scanner installations on this system

SonarQube Scanner

SonarQube Scanner installations

[Add SonarQube Scanner](#)

Name

SonaQubeScanner-4.6.2

Required

 Install automatically

Install from Maven Central

Version

SonarQube Scanner 4.6.2.2472 ▾

[Add Installer ▾](#)[Add SonarQube Scanner](#)

List of SonarQube Scanner installations on this system

[Save](#)[Apply](#)

Install sonarqube scanner .

Lets create pipeline for sonarqube scan .

New item > pipeline job > pipeline script – copy script >

In mvn version we can see maven home path :

```
[root@ip-172-31-34-251 ~]# mvn -version
Apache Maven 3.8.2 (ea98e05a04480131370aa0c110b8c54cf726c06f)
Maven home: /opt/apache-maven-3.8.2
Java version: 11.0.12, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-11.0.12.0.7-0.amzn2.0.2.
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.14.243-185.433.amzn2.x86_64", arch: "amd64", family: "unix"
[root@ip-172-31-34-251 ~]#
```

Copy paste the code and sonarqube scan will be done .

We can create a new quality profile and apply all rules applicable .

Quality gate – here we check bugs velnerabilites based on given value and if it is below the given threshold then our scan will get failed .

Conditions

Conditions on New Code

Metric	Operator	Value
Coverage	is less than	80.0% 
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

Projects

Every project not specifically associated to a quality gate will be associated to this one by default.

Quality gate implementation in sonarqube :

<https://jenkinshero.com/sonarqube-quality-gates-in-jenkins-build-pipeline/>