DevOps Shack

Maven Comprehensive Guide with Examples and Sample Commands

[Click Here To Enrol To Batch-5 | DevOps & Cloud DevOps]

Table of Contents

1. Setup and Installation

Install Maven

1. **Download Maven** from the official website.
2. **Extract the archive**:

   ```
   tar xzvf apache-maven-3.x.x-bin.tar.gz
   ```

3. **Move to the desired directory**:

   ```
   sudo mv apache-maven-3.x.x /usr/local/apache-maven
   ```

4. **Set environment variables**:
5. ```
   export M2_HOME=/usr/local/apache-maven
   ```
6. ```
   export M2=$M2_HOME/bin
   export PATH=$M2:$PATH
   ```

7. **Verify installation**:

   ```
   mvn -version
   ```

---

2. Basic Commands

Create a New Maven Project

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

Clean Project

```
mvn clean
```

Compile Project

```
mvn compile
```

Package Project

```
mvn package
```

Install Project to Local Repository

```
mvn install
```

Deploy Project

```
mvn deploy
```

3. Project Creation and Structure

Create a Simple Maven Project

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

Standard Directory Layout

```
my-app
|-- src
|   |-- main
|   |   |-- java
|   |   |   `-- com
|   |   |       `-- example
|   |   |           `-- App.java
|   |   `-- resources
|   `-- test
|       |-- java
|       |   `-- com
|       |       `-- example
|       |           `-- AppTest.java
|       `-- resources
|-- pom.xml
```

Sample `pom.xml` File

```
<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>
    <groupId>com.example</groupId>
    <artifactId>my-app</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

4. Dependencies Management

Adding Dependencies

Update the `dependencies` section in your `pom.xml`:

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.2.8.RELEASE</version>
</dependency>
```

Excluding Transitive Dependencies

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.4.21.Final</version>
    <exclusions>
        <exclusion>
            <groupId>org.jboss.logging</groupId>
            <artifactId>jboss-logging</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Using Dependency Management

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
    </dependencies>
</dependencyManagement>
```

5. Build Lifecycle

Validate

```
mvn validate
```

Compile

```
mvn compile
```

Test

```
mvn test
```

Package

```
mvn package
```

Install

```
mvn install
```

Deploy

```
mvn deploy
```

---

6. Plugins

Adding a Plugin

Update the `plugins` section in your `pom.xml`:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Running Plugin Goals

```
mvn compiler:compile
```

Creating a Custom Plugin

Create a new project and define the plugin:

```
<project>
    ...
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-plugin-plugin</artifactId>
                <version>3.5</version>
            </plugin>
        </plugins>
    </build>
    ...
</project>
```

---

7. Profiles

Creating Profiles

```
<profiles>
    <profile>
        <id>production</id>
        <properties>
            <environment>production</environment>
        </properties>
    </profile>
    <profile>
        <id>development</id>
        <properties>
            <environment>development</environment>
        </properties>
    </profile>
</profiles>
```

Activating Profiles

```
mvn clean install -P production
```

Conditional Configuration

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
```

```
            <configuration>
                <skipTests>${skipTests}</skipTests>
            </configuration>
        </plugin>
    </plugins>
</build>
```

---

8. Testing

Using Surefire Plugin

Add the Surefire plugin to your `pom.xml`:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.2</version>
            <configuration>
                <includes>
                    <include>**/*Test.java</include>
                </includes>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Running Tests

```
mvn test
```

Skipping Tests

```
mvn install -DskipTests
```

---

9. Site Generation

Generating a Site

Add the site plugin to your `pom.xml`:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-plugin</artifactId>
            <version>3.9.1</version>
        </plugin>
    </plugins>
```

```
</build>
```

Run Site Generation

```
mvn site
```

---

10. Deployment

Deploy to a Remote Repository

Update your `pom.xml` with distribution management:

```
<distributionManagement>
    <repository>
        <id>releases</id>
        <url>http://myrepo/releases</url>
    </repository>
    <snapshotRepository>
        <id>snapshots</id>
        <url>http://myrepo/snapshots</url>
    </snapshotRepository>
</distributionManagement>
```

Run Deploy Command

```
mvn deploy
```

---

11. Integration with CI/CD

Jenkins Pipeline

Create a Jenkinsfile in the root of your project:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'mvn clean install'
            }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                sh 'mvn
```

```
 deploy'
            }
        }
    }
}
```

GitLab CI/CD

Create a `.gitlab-ci.yml` file in the root of your project:

```
image: maven:3.6.3-jdk-8

stages:
  - build
  - test
  - deploy

build:
  stage: build
  script:
    - mvn clean install

test:
  stage: test
  script:
    - mvn test

deploy:
  stage: deploy
  script:
    - mvn deploy
```

---

12. Advanced Examples

Docker Integration

Create a Docker image as part of the Maven build process using the `dockerfile-maven-plugin`:

```
<plugin>
    <groupId>com.spotify</groupId>
    <artifactId>dockerfile-maven-plugin</artifactId>
    <version>1.4.10</version>
    <executions>
        <execution>
            <goals>
                <goal>build</goal>
                <goal>push</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <repository>my-docker-repo/my-app</repository>
        <tag>${project.version}</tag>
    </configuration>
```

```
</plugin>
```

Using Properties

Define properties in `pom.xml`:

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

Use properties in your configuration:

```
<configuration>
    <source>${maven.compiler.source}</source>
    <target>${maven.compiler.target}</target>
</configuration>
```

Multi-module Projects

Define a parent `pom.xml`:

```
<modules>
    <module>module-a</module>
    <module>module-b</module>
</modules>
```

Each module will have its own `pom.xml`:

```
<parent>
    <groupId>com.example</groupId>
    <artifactId>my-app</artifactId>
    <version>1.0-SNAPSHOT</version>
</parent>
```