Comandos Maven.

1. Comandos Básicos.

- ***mvn clean****
  Elimina archivos generados ya compilados y empaquetados previamente para empezar una proyecto desde cero.
  1. Ejecuta el plugin ***maven-clean-plugin*** (asociado a la fase clean).
  2. Elimina el directorio ***target/***:
     - El directorio contiene lo ya generado por Maven (*Clases* compiladas, *JAR/WAR*, informes de pruebas, etc.).
     - Si el proyecto es multi módulo, limpia el *target/* de cada submódulo.
  3. No afecta el repositorio local ***~/.m2/repository/***.

- ***mvn validate***
  Verifica que el proyecto esté correctamente configurado (por ejemplo, que el pom.xml sea válido y todas las dependencias estén disponibles).

- ***mvn compile****
  Compila el código fuente del proyecto (ubicado en *src/main/java*) y lo guarda en *target/classes/*:
  1. Ejecuta el plugin ***maven-compiler-plugin*** (configurado en el *pom.xml*).
  2. Compila el código en ***src/main/java/:***
     - Usa la versión de Java definida en **<source>** y **<target>** (por defecto, Java 1.7 si no se especifica).
     - Los archivos ***.class*** resultantes se guardan en ***target/classes/***.
  3. Procesa recursos (archivos en *src/main/resources/*):
     - Copia archivos como ***.properties*** o ***.xml*** a *target/classes/*.

- ***mvn test****
  Ejecuta pruebas unitarias *(src/test/java)* con *Frameworks* como *JUnit5*; crea informes en *target/surefire-reports/*.
  1. Ejecuta automáticamente ***compile*** (por dependencia del ciclo de vida).
  2. Compila el código de pruebas en ***src/test/java/*** usando ***maven-compiler-plugin***:
     - Los archivos *.class* de pruebas se guardan en ***target/test-classes/***.
  3. Ejecuta las pruebas con el plugin ***maven-surefire-plugin***:
     - Busca *Clases* que coincidan con los patrones *****/Test*.java***, *****/*Test.java***, o *****/*TestCase.java***.
     - Genera informes en ***target/surefire-reports/*** *(formato .txt y .xml)*.
  4. Si una prueba falla, el build se detiene *(a menos que se use* ***-DtestFailureIgnore=true****)*.

- ***mvn package****
  Empaqueta el proyecto en un ejecutable (*JAR*, *WAR*, etc.) según la configuración del *pom.xml* y lo guarda en target/.
  1. Ejecuta todas las fases anteriores (*validate*, *compile*, *test*, etc.).
  2. Empaqueta según el ***<packaging>*** del *pom.xml*:
     - ***JAR****:* Usa ***maven-jar-plugin*** para crear un *JAR* en *target/*.
     - ***WAR****:* Usa ***maven-war-plugin*** para empaquetar el proyecto web *(incluyendo WEB-INF/)*.
  3. Ejecuta plugins de empaquetado adicionales:
     - Por ejemplo, ***maven-shade-plugin*** para *JARs* con dependencias incluidas *("fat JAR")*.
  4. El artefacto se guarda en ***target/*** *(ej: mi-proyecto-1.0.jar)*.

- ***mvn verify***
  Ejecuta pruebas de integración y verifica que se cumplan criterios de calidad (por ejemplo, con plugins como checkstyle o failsafe).
  ****NOTA:*** Ideal para integración continua *(incluye pruebas de integración con* ***maven-failsafe-plugin****)*.

- ***mvn install****
  Instala el artefacto *JAR/WAR* en el repo local *(~/.m2/repository/)* y ponerlo a disposición como dependencia en más proyectos.
  1. Ejecuta todas las fases anteriores (incluyendo *package*).
  2. Copia el artefacto (*JAR/WAR*) al repositorio local (***~/.m2/repository/***):
     - La ruta sigue el formato: ***~/.m2/repository/groupId/artifactId/version/artifactId-version.package***
       p.e.: ~/.m2/repository/com/example/mi-proyecto/1.0/mi-proyecto-1.0.jar.
  3. Instala también el ***pom.xml*** (para manejar dependencias transitivas).

  ****NOTA: mvn clean install*** es la combinación más usada para reconstruir todo e instalar localmente.

- ***mvn deploy****
  Sube el artefacto a un repositorio remoto (como *Nexus* o *Artifactory*), configurado en el *pom.xml* o *settings.xml*.
  1. Ejecuta todas las fases anteriores (incluyendo *install*).
  2. Requiere configuración previa en *pom.xml* o *settings.xml*:
     - Se debe definir el ***<distributionManagement>*** con las URLs del repositorio remoto.

```
<distributionManagement>
  <repository>
    <id>my-repo</id>
    <url>https://repo.example.com/releases</url>
  </repository>
</distributionManagement>
```

  3. Usa el plugin ***maven-deploy-plugin*** para subir:
     - El artefacto (*JAR/WAR*).
     - El *pom.xml*.
     - Archivos adicionales como fuentes o *javadoc* (si están configurados).

  ***NOTA:** Solo se usa en entornos de producción o *CI/CD* para publicar artefactos.

2. Comandos para Dependencias.

Gestionan y analizan dependencias del proyecto.

- ***mvn dependency:tree****
  Muestra el árbol de dependencias del proyecto, incluyendo dependencias transitivas. Útil para detectar conflictos de versiones.
- ***mvn dependency:analyze***
  Identifica dependencias declaradas en el pom.xml que no se usan en el código, o dependencias usadas pero no declaradas.

3. Comandos para Generación de Proyectos y Documentación.

- ***mvn archetype:generate****
  Crea un nuevo proyecto Maven usando una plantilla (arquetipo). Por ejemplo:
  mvn archetype:generate -DgroupId=com.example -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart
- ***mvn site:site***
  Genera documentación del proyecto (informes de pruebas, cobertura, etc.) en *target/site/*.

4. Opciones Adicionales de Maven.

Modifican el comportamiento de los comandos anteriores.

- ***mvn -f dir/pom.xml package***
  Especifica una ruta alternativa al archivo *pom.xml* (útil para proyectos con múltiples módulos).
- ***mvn -o package***
  Ejecuta Maven en modo offline (sin descargar dependencias de Internet). Usa solo el repositorio local.
- ***mvn -q package***
  Modo silencioso: solo muestra errores y resultados de pruebas.
- ***mvn -X package***
  Modo debug: muestra información detallada del proceso de construcción.
- ***mvn -V o mvn -V package***
  Muestra la versión de Maven (-V solo muestra la versión; -V package la muestra y luego ejecuta el comando).
- ***mvn -DskipTests package***
  Omite la ejecución de pruebas unitarias. Equivalente a *-Dmaven.test.skip=true*.
- ***mvn -T 4 clean install***
  Ejecuta el build en paralelo con 4 hilos (acelera la construcción en proyectos multimódulo).

5. Ayuda y Versión.

- ***mvn -help***
  Muestra la ayuda de Maven, incluyendo todas las opciones disponibles.
- ***Ejemplo de Uso Combinado***
  Limpiar, compilar, saltar pruebas y empaquetar en modo paralelo:
  mvn clean package -DskipTests -T 4