DevOps Shack

200 Maven, NPM Interview Questions and Answers

1. What happens if you remove the <scope> tag from a dependency in pom.?

Answer:
If the <scope> tag is removed, the dependency defaults to the compile scope. This means the dependency will be available at compile-time and runtime and will be included in the final package.

2. How does Maven handle transitive dependencies?

Answer:
Maven automatically resolves dependencies for libraries that your project depends on. If a library A depends on another library B, Maven will download and include B as well. However, it follows the "nearest definition" rule, where the closest version of a dependency is used in case of conflicts.

3. What is the difference between mvn clean install and mvn package?

Answer:

- mvn clean install: Deletes previous builds, compiles code, runs tests, and installs the packaged artifact in the local repository.

- `mvn package`: Only compiles and packages the application but does not install it in the local repository.

4. What is the role of a `settings.` file in Maven?

Answer:
The `settings.` file contains user-specific configurations like repository locations, authentication credentials, and proxy settings. It is located in the `.m2` directory of the user's home folder.

5. How can you exclude a transitive dependency in Maven?

Answer:
You can exclude a transitive dependency using the `<exclusions>` tag inside the dependency declaration:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
        </exclusion>
```

```
    </exclusions>
</dependency>
```

6. What happens if two dependencies in Maven bring different versions of the same transitive dependency?

Answer:
Maven follows the "nearest definition" rule and selects the version closest to your project in the dependency tree. If needed, you can override this by explicitly specifying a version in your `pom.`.

7. What is the purpose of the `dependencyManagement` tag in Maven?

Answer:
The `dependencyManagement` tag allows you to specify versions of dependencies centrally without adding them as direct dependencies in all modules. It is mainly used in multi-module projects.

8. How does the Maven lifecycle work?

Answer:
Maven has three main lifecycles:

- Clean: `mvn clean` deletes previous build artifacts.
- Default: Includes phases like compile, test, package, install, and deploy.
- Site: Generates documentation.

9. What is the difference between `provided` and `compile` scope in Maven?

Answer:

- **Compile Scope: The dependency is available in all build phases and is included in the final package.**
- **Provided Scope: The dependency is required for compilation but is not included in the final package (e.g., `javax.servlet` API, which is provided by a servlet container).**

10. How do you create a Maven project without an internet connection?

Answer:
Use the `-o` (offline) flag:

```
mvn -o package
```

This will use the locally cached dependencies.

11. What is the difference between `dependencies` and `devDependencies` in `package.`?

Answer:

- **`dependencies`: Required for the application to run.**
- **`devDependencies`: Required only for development (e.g., testing libraries, build tools).**

12. What happens when you run `npm install`?

Answer:

- **Installs all dependencies listed in `package.`.**
- **If a `package-lock.` exists, it ensures consistent versions.**
- **Creates a `node_modules` folder (if not already present).**

13. How do you install a specific version of an NPM package?

Answer:
Use:

```
npm install package-name@version
```

Example:

```
npm install express@4.17.1
```

14. What is the difference between `npm install -g` and `npm install`?

Answer:

- **`npm install -g` installs the package globally, making it available system-wide.**
- **`npm install` installs the package locally in `node_modules`.**

15. How do you update all outdated packages in `package.`?

Answer:
Run:

```
npm update
```

16. What is the purpose of `npx`?

Answer:
`npx` runs packages without installing them globally.
Example:

```
npx create-react-app my-app
```

This ensures you always use the latest version.

17. What is the difference between `npm ci` and `npm install`?

Answer:

- **`npm install` updates `package-lock.` if there are changes.**
- **`npm ci` installs dependencies strictly according to `package-lock.`, ensuring consistency.**

18. How do you check the installed versions of NPM packages?

Answer:
Run:

```
npm list
```

For globally installed packages:

```
npm list -g --depth=0
```

19. What is the purpose of the peerDependencies field in package.?

Answer:
It specifies packages that the project expects to be installed by the consumer of the package, rather than being installed automatically.

20. How do you remove a package from package. and node_modules?

Answer:
Use:

```
npm uninstall package-name
```

21. How do you deploy a Maven project to an internal Nexus repository?

Answer:
Use the distributionManagement section in pom.:

```
<distributionManagement>
    <repository>
        <id>internal-repo</id>
<url>http://your-nexus-repo/repository/maven-releases/</
/url>
    </repository>
```

```
</distributionManagement>
```

Then run:

```
mvn deploy
```

22. What will happen if you use `mvn install` without first running `mvn clean`?

Answer:
It will build and install the project, but old artifacts or compiled files may still exist in the target directory, potentially causing unexpected behavior.

23. How do you skip tests while building a Maven project?

Answer:
Use the `-DskipTests` flag:

```
mvn clean install -DskipTests
```

Alternatively, in pom.:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
        <skipTests>true</skipTests>
```

```
    </configuration>
</plugin>
```

24. How do you force Maven to update dependencies?

Answer:
Use the `-U` flag:

```
mvn clean install -U
```

This forces Maven to check for updated dependencies.

25. What is the difference between `mvn dependency:tree` and `mvn dependency:list`?

Answer:

- `mvn dependency:tree` ows the hierarchy of dependencies.
- `mvn dependency:list` ows all dependencies in a flat list.

26. What is a Maven BOM (Bill of Materials)?

Answer:
A BOM is a special POM that manages dependency versions centrally using `dependencyManagement`. Example:

```
<dependencyManagement>
    <dependencies>
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-dependencies</artifactId>

            <version>2.5.0</version>

            <type>pom</type>

            <scope>import</scope>

        </dependency>

    </dependencies>

</dependencyManagement>
```

27. How do you configure a Maven project to use a specific Java version?

Answer:
Use the maven-compiler-plugin:

```
<plugin>

    <groupId>org.apache.maven.plugins</groupId>

    <artifactId>maven-compiler-plugin</artifactId>

    <version>3.8.1</version>

    <configuration>

        <source>11</source>
```

```
        <target>11</target>
    </configuration>
</plugin>
```

28. What does the `-pl` flag do in Maven?

Answer:
The `-pl` flag allows you to build only specific modules in a multi-module project:

```
mvn clean install -pl module-name
```

29. How do you generate a dependency report in Maven?

Answer:
Run:

```
mvn dependency:tree > dependencies.txt
```

30. How do you resolve "Could not find or load main class" after building a Maven project?

Answer:

- **Ensure the `mainClass` is correctly specified in `pom.`:**

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <configuration>
```

```
        <archive>
            <manifest>
<mainClass>com.example.MainClass</mainClass>
            </manifest>
        </archive>
    </configuration>
</plugin>
```

- Run:

```
mvn clean package
java -jar target/app.jar
```

31. What is the purpose of `package-lock.` in NPM?

Answer:
`package-lock.` ensures consistent dependency versions across different environments by locking exact versions.

32. What happens if `node_modules` is deleted and you run `npm install` again?

Answer:
NPM will reinstall all dependencies as per `package.` and `package-lock.`.

33. How do you install an NPM package from a GitHub repository?

Answer:
Use:

```
npm install github:username/repository
```

34. How do you list outdated dependencies in an NPM project?

Answer:
Run:

```
npm outdated
```

35. How do you install an NPM package only for a single project (without adding it to package.)?

Answer:
Use the `--no-save` flag:

```
npm install package-name --no-save
```

36. How do you globally install an NPM package and then use it?

Answer:
Use:

```
npm install -g package-name
```

Example:

```
npm install -g nodemon

nodemon app.js
```

37. How do you completely remove an NPM package and all its dependencies?

Answer:
Run:

```
npm uninstall package-name

rm -rf node_modules

npm install
```

38. What happens if you install a package without specifying a version?

Answer:
The latest version will be installed.

39. How do you execute a script defined in `package.`?

Answer:
Use:

```
npm run script-name
```

Example:

```
"scripts": {

    "start": "node index.js"

}
```

Run:

```
npm run start
```

40. What is the difference between `npm prune` and `npm dedupe`?

Answer:

- `npm prune`: Removes unnecessary dependencies that are not listed in `package..`
- `npm dedupe`: Removes duplicate dependencies to optimize `node_modules`.

41. How do you handle a Maven build failure due to a missing dependency?

Answer:

- Check if the dependency is available in the Maven central repository.

Run:

```
mvn dependency:resolve
```

If missing, manually install it in the local repository:

```
mvn install:install-file -Dfile=path-to-jar
-DgroupId=com.example -DartifactId=my-lib -Dversion=1.0
-Dpackaging=jar
```

42. What is the difference between `mvn verify` and `mvn test`?

Answer:

- `mvn test` runs only unit tests.
- `mvn verify` runs integration tests along with unit tests.

43. How do you analyze Maven dependencies to detect conflicts?

Answer:
Run:

```
mvn dependency:tree -Dverbose
```

or

```
mvn dependency:analyze
```

44. What is the difference between a parent POM and a BOM in Maven?

Answer:

- Parent POM: Defines common configurations, properties, and plugins for child projects.
- BOM (Bill of Materials): Manages dependency versions using `dependencyManagement` without enforcing inheritance.

45. How do you repackage a Spring Boot application as an executable JAR?

Answer:
Ensure the `spring-boot-maven-plugin` is configured:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Then run:

```
mvn clean package
java -jar target/app.jar
```

46. How do you create a custom Maven plugin?

Answer:

- Create a Maven project with `maven-plugin` packaging.
- Implement the `org.apache.maven.plugin.AbstractMojo` class.
- Define execution goals in `META-INF/maven/plugin.`.

47. What are Maven SNAPOT versions, and how do they work?

Answer:

A SNAPOT version (e.g., 1.0-SNAPOT) is a development version that changes frequently. Maven fetches the latest version from the repository unless cached.

48. How do you publi a Maven artifact to a remote repository?

Answer:
Ensure distributionManagement is set, then run:

```
mvn deploy
```

If using credentials, store them in settings.:

```
<servers>
    <server>
        <id>my-repo</id>
        <username>user</username>
        <password>password</password>
    </server>
</servers>
```

49. How do you generate project documentation using Maven?

Answer:
Run:

```
mvn site
```

This generates a site in target/site.

50. How do you include resources in a Maven JAR?

Answer:
Place resources in src/main/resources or configure maven-resources-plugin:

```
<build>
    <resources>
        <resource>
            <directory>src/main/resources</directory>
        </resource>
    </resources>
</build>
```

51. What is the difference between npm rebuild and npm install?

Answer:

- npm install installs missing dependencies.
- npm rebuild recompiles native modules.

52. How do you install a package from a private NPM registry?

Answer:

Run:

```
npm set registry https://custom-registry.com

npm install package-name
```

Or configure .npmrc:

```
registry=https://custom-registry.com

_authToken=your-token
```

53. How do you list globally installed NPM packages?

Answer:
Run:

```
npm list -g --depth=0
```

54. How do you find which package introduced a vulnerable dependency?

Answer:
Use:

```
npm audit
```

55. How do you run multiple NPM scripts sequentially?

Answer:
Use &&:

```
npm run build && npm run test
```

56. How do you run multiple NPM scripts in parallel?

Answer:
Use `&`:

```
npm run lint & npm run test
```

57. What is the purpose of `npm rinkwrap`?

Answer:
It locks exact dependency versions similar to `package-lock.` but is used for publiing libraries.

58. How do you install only `devDependencies` without `dependencies`?

Answer:
Run:

```
npm install --only=dev
```

59. How do you create a new package using `npm init` with defaults?

Answer:
Use:

```
npm init -y
```

60. How do you set environment variables while running an NPM script?

Answer:
Use:

```
NODE_ENV=production npm run start
```

61. How do you configure multi-module Maven projects?

Answer:
Define a parent POM and include submodules:

```
<modules>
    <module>module1</module>
    <module>module2</module>
</modules>
```

62. How do you cache NPM dependencies in a CI/CD pipeline?

Answer:
For GitHub Actions:

yml

```
- name: Cache NPM dependencies
  uses: actions/cache@v2
  with:
    path: ~/.npm
    key: ${{ runner.os }}-npm-${{
haFiles('**/package-lock.') }}
    restore-keys: |
      ${{ runner.os }}-npm-
```

63. How do you create a monorepo using NPM workspaces?

Answer:
Enable workspaces in package.:

```
{
  "workspaces": ["packages/*"]
}
```

Then install dependencies:

```
npm install
```

64. How do you migrate from NPM to Yarn?

Answer:
Run:

```
yarn import
```

65. How do you resolve version conflicts in an NPM project?

Answer:
Run:

```
npm dedupe
```

or manually update package..

66. How do you handle cross-platform scripts in NPM?

Answer:
Use `cross-env`:

```
npm install cross-env --save-dev
```

Then:

```
cross-env NODE_ENV=production npm run build
```

67. What happens if an NPM package has a `peerDependency` that is missing?

Answer:
A warning is own, but installation continues.

68. How do you publi an NPM package?

Answer:
Run:

```
npm login
```

```
npm publi
```

69. How do you force remove all NPM caches?

Answer:
Run:

```
npm cache clean --force
```

70. How do you run an NPM script conditionally based on the OS?

Answer:

Use:

```
"scripts": {
  "start:win": "set NODE_ENV=production && node app.js",
  "start:unix": "NODE_ENV=production node app.js"
}
```

71. How do you resolve a Maven build error due to a missing artifact?

Answer:

- **Ensure the dependency is available in the Maven Central repository.**

Run:

```
mvn dependency:resolve
```

If using a private repository (like Nexus), verify the repository URL in pom.:

```
<repositories>
    <repository>
        <id>my-repo</id>
<url>https://my-nexus-repo/repository/maven-public/</url>
```

```
    </repository>
</repositories>
```

If the artifact is not available, install it manually:

```
mvn install:install-file -Dfile=path-to-jar
-DgroupId=com.example -DartifactId=my-lib -Dversion=1.0
-Dpackaging=jar
```

72. How do you add multiple repositories in Maven?

Answer:
In pom., use:

```
<repositories>
    <repository>
        <id>central</id>
        <url>https://repo.maven.apache.org/maven2</url>
    </repository>
    <repository>
        <id>internal-repo</id>

<url>https://mycompany.com/repository/maven-releases/</
url>
    </repository>
</repositories>
```

73. How do you run Maven in debug mode?

Answer:
Run:

```
mvn clean install -X
```

This provides detailed logs for debugging.

74. How do you configure a Maven project to generate both JAR and WAR artifacts?

Answer:
Use the `build` section:

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

```
    </plugins>
</build>
```

Run:

```
mvn package
```

75. How do you set a system property while running a Maven build?

Answer:
Use:

```
mvn clean install -DmyProperty=value
```

Or in pom.:

```
<properties>
    <myProperty>value</myProperty>
</properties>
```

76. How do you configure Maven to build a Java 17 project?

Answer:
Use:

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

```
    <configuration>
        <source>17</source>
        <target>17</target>
    </configuration>
</plugin>
```

77. How do you skip a specific Maven plugin during a build?

Answer:
Use:

```
mvn clean install -Dplugin.name.skip=true
```

For example, to skip the Surefire plugin:

```
mvn clean install -DskipTests
```

78. What is the difference between `mvn clean compile` and `mvn clean install`?

Answer:

- `mvn clean compile` compiles the project but does not package or install artifacts.
- `mvn clean install` compiles, packages, and installs the artifact into the local repository.

79. How do you enforce dependency versioning across multiple Maven projects?

Answer:
Use a BOM (Bill of Materials):

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-dependencies</artifactId>
            <version>2.5.0</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

80. How do you generate a Maven dependency report?

Answer:
Run:

```
mvn dependency:tree > dependency-report.txt
```

81. How do you check for outdated dependencies in an NPM project?

Answer:
Run:

```
npm outdated
```

It ows outdated dependencies and available updates.

82. How do you update an NPM package to the latest version?

Answer:
Run:

```
npm update package-name
```

83. What is the difference between `npm update` and `npm upgrade`?

Answer:

- **`npm update` updates dependencies to the latest compatible versions as per `package.`.**
- **`npm upgrade` updates dependencies to the latest versions available.**

84. How do you install a package as a peer dependency?

Answer:
Run:

```
npm install package-name --save-peer
```

85. How do you fix vulnerabilities in NPM dependencies?

Answer:
Run:

```
npm audit fix
```

For major changes:

```
npm audit fix --force
```

86. How do you create a global alias for an NPM script?

Answer:
In package.:

```
"scripts": {
  "dev": "node server.js"
}
```

Run:

```
npm run dev
```

87. How do you set an environment variable inside an NPM script?

Answer:
Use cross-env:

```
npm install cross-env --save-dev
```

Then:

```
"scripts": {
```

```
  "start": "cross-env NODE_ENV=production node
server.js"

}
```

88. How do you list all installed dependencies and their versions?

Answer:
Run:

```
npm list --depth=0
```

89. How do you uninstall an NPM package completely?

Answer:
Run:

```
npm uninstall package-name
```

90. How do you install a package from a different registry?

Answer:
Run:

```
npm install package-name
--registry=https://custom-registry.com
```

91. How do you speed up Maven builds?

Answer:

Use:

```
mvn clean install -T 4
```

- (Runs in parallel with 4 threads)

Enable incremental builds:

```
mvn clean install -Dmaven.test.skip=true
```

- Use a local repository manager like Nexus.

92. How do you manage different NPM package versions in a monorepo?

Answer:
Use NPM Workspaces:

```
{
  "workspaces": ["packages/*"]
}
```

93. How do you create a custom NPM package?

Answer:
Run:

```
npm init
```

Then publi:

```
npm publi
```

94. How do you ensure all dependencies in an NPM

project are using the same versions?

Answer:
Use:

```
npm dedupe
```

95. How do you clear the NPM cache?

Answer:
Run:

```
npm cache clean --force
```

96. How do you skip a specific test class while running `mvn test`?

Answer:
Use the `-Dtest` flag:

```
mvn test -Dtest=TestClassName
```

Or exclude it in `pom.`:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
        <excludes>
            <exclude>**/TestClassName.java</exclude>
```

```
        </excludes>
    </configuration>
</plugin>
```

97. How do you build a Maven project without running tests?

Answer:
Run:

```
mvn clean install -DskipTests
```

or

```
mvn clean install -Dmaven.test.skip=true
```

The second option skips test compilation as well.

98. How do you force Maven to always check for updated dependencies?

Answer:
Use:

```
mvn clean install -U
```

This forces updates for SNAPOT dependencies.

99. How do you resolve dependency conflicts in a multi-module Maven project?

Answer:
Use:

```
mvn dependency:tree
```

To enforce a specific version, define it in dependencyManagement:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.12.0</version>
        </dependency>
    </dependencies>
</dependencyManagement>
```

100. How do you profile a Maven build?

Answer:
Use:

```
mvn clean install -P profile-name
```

Define profiles in pom.:

```
<profiles>
    <profile>
```

```
        <id>dev</id>
        <properties>
            <env>development</env>
        </properties>
    </profile>
</profiles>
```

101. How do you analyze the build time of each phase in Maven?

Answer:
Use:

```
mvn clean install -X
```

or

```
mvn clean install -DbuildTimeSummary=true
```

102. How do you configure Maven to build different environments like dev, staging, and production?

Answer:
Use profiles in pom.:

```
<profiles>
    <profile>
        <id>dev</id>
```

```
        <activation>
            <activeByDefault>true</activeByDefault>
        </activation>
        <properties>
            <env>development</env>
        </properties>
    </profile>
    <profile>
        <id>prod</id>
        <properties>
            <env>production</env>
        </properties>
    </profile>
</profiles>
```

Activate with:

```
mvn clean package -P prod
```

103. What is the difference between `mvn dependency:copy-dependencies` and `mvn dependency:go-offline`?

Answer:

- `mvn dependency:copy-dependencies`: Copies dependencies to the `target/dependency` folder.
- `mvn dependency:go-offline`: Downloads all dependencies to the local repository for offline use.

104. How do you check the effective POM after inheritance and profiles are applied?

Answer:
Run:

```
mvn help:effective-pom
```

105. How do you create a custom Maven lifecycle?

Answer:
Define a plugin that binds to new phases in pom.:

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>build-helper-maven-plugin</artifactId>
    <executions>
        <execution>
            <goals>
                <goal>add-source</goal>
```

```
            </goals>
        </execution>
    </executions>
</plugin>
```

106. How do you install only production dependencies using NPM?

Answer:
Run:

```
npm install --production
```

This excludes devDependencies.

107. How do you check the latest available version of an NPM package?

Answer:
Run:

```
npm ow package-name version
```

or

```
npm info package-name
```

108. How do you list the top-level dependencies of an NPM project?

Answer:
Run:

```
npm list --depth=0
```

109. How do you prevent breaking changes when updating NPM packages?

Answer:

Use semantic versioning in package.:

```
"dependencies": {
    "express": "^4.17.1"
}
```

Lock versions using:

```
npm ci
```

110. How do you create a private NPM package?

Answer:

Add "private": true to package.:

```
{
    "name": "my-package",
    "version": "1.0.0",
    "private": true
}
```

Publi to a private registry:

```
npm publi --registry=https://my-private-npm.com
```

111. How do you test an unpublied NPM package locally?

Answer:
Run:

```
npm link
```

In another project:

```
npm link package-name
```

112. How do you generate a dependency graph in NPM?

Answer:
Run:

```
npm list
```

or use:

```
npm ls --
```

113. How do you downgrade an NPM package to a specific version?

Answer:
Run:

```
npm install package-name@version
```

Example:

```
npm install express@4.16.3
```

114. How do you install an NPM package without modifying package.?

Answer:
Use:

```
npm install package-name --no-save
```

115. How do you remove all installed NPM packages from a project?

Answer:
Run:

```
rm -rf node_modules
```

```
npm cache clean --force
```

116. How do you debug NPM scripts?

Answer:
Use:

```
npm run script-name --verbose
```

117. How do you publi a scoped NPM package?

Answer:
Run:

```
npm publi --access=public
```

118. How do you find which package introduced a specific dependency?

Answer:
Run:

```
npm why package-name
```

119. How do you clean the local Maven repository?

Answer:
Run:

```
rm -rf ~/.m2/repository
```

or

```
mvn dependency:purge-local-repository
```

120. How do you verify if an NPM package is installed globally?

Answer:
Run:

```
npm list -g --depth=0
```

121. How do you deploy a Maven project to a remote Nexus repository using credentials?

Answer:

Add the repository in pom.:

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

Store credentials in ~/.m2/settings.:

```
<servers>
    <server>
        <id>nexus-repo</id>
        <username>myuser</username>
        <password>mypassword</password>
    </server>
</servers>
```

Deploy with:

```
mvn deploy
```

122. What is the difference between `mvn site` and `mvn javadoc:javadoc`?

Answer:

- `mvn site` generates a project documentation site with reports.
- `mvn javadoc:javadoc` generates only Javadoc documentation.

123. How do you exclude a transitive dependency in Maven?

Answer:
Use the `<exclusions>` tag:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

124. How do you package a multi-module Maven project into a single JAR?

Answer:

Use the Maven ade Plugin:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-ade-plugin</artifactId>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>ade</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Then run:

```
mvn clean package
```

125. How do you find unused dependencies in a Maven project?

Answer:
Run:

```
mvn dependency:analyze
```

126. How do you run only integration tests and skip unit tests in Maven?

Answer:
Run:

```
mvn verify -DskipUnitTests
```

Define in pom.:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
        <excludes>
            <exclude>**/UnitTest*</exclude>
```

```
        </excludes>
    </configuration>
</plugin>
```

127. How do you use a specific Maven profile in Jenkins?

Answer:
In the Jenkins build step, add:

```
mvn clean package -Pprofile-name
```

128. How do you override properties in a Maven build?

Answer:
Run:

```
mvn clean install -DpropertyName=value
```

129. What is the difference between `mvn compile` and `mvn package`?

Answer:

- **`mvn compile`: Compiles Java source files but does not package them.**
- **`mvn package`: Compiles and packages them into a JAR/WAR.**

130. How do you generate a JAR with dependencies in Maven?

Answer:
Use the Maven Assembly Plugin:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <executions>
        <execution>
            <id>make-assembly</id>
            <phase>package</phase>
            <goals>
                <goal>single</goal>
            </goals>
            <configuration>
                <descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
        </execution>
    </executions>
</plugin>
```

131. How do you uninstall a global NPM package?

Answer:
Run:

```
npm uninstall -g package-name
```

132. How do you install an NPM package without adding it to package.?

Answer:
Use:

```
npm install package-name --no-save
```

133. How do you create an NPM package and publi it?

Answer:

Initialize the package:

```
npm init
```

Publi it:

```
npm publi
```

134. How do you install an NPM package from a GitHub repository?

Answer:
Run:

```
npm install github:username/repository
```

135. How do you check the versions of installed NPM packages?

Answer:
Run:

```
npm list --depth=0
```

136. How do you upgrade all outdated NPM dependencies?

Answer:
Run:

```
npm update
```

137. What is the difference between `npm cache clean` and `npm cache verify`?

Answer:

- `npm cache clean --force`: Deletes all cached packages.
- `npm cache verify`: Checks the integrity of the NPM cache.

138. How do you prevent an NPM package from being accidentally publied?

Answer:
Add `"private": true` to `package.`:

```
{
  "name": "my-package",
  "version": "1.0.0",
  "private": true
}
```

139. How do you add an NPM package to devDependencies?

Answer:
Run:

```
npm install package-name --save-dev
```

140. How do you globally install multiple NPM packages at once?

Answer:
Run:

```
npm install -g package1 package2 package3
```

141. How do you enforce a specific Node.js version in an NPM project?

Answer:
Specify in package.:

```
"engines": {
  "node": ">=14.0.0"
```

```
}
```

Then use:

```
nvm use 14
```

142. How do you prevent installing new NPM packages but allow updates?

Answer:
Run:

```
npm ci
```

143. How do you force NPM to install dependencies from package-lock.?

Answer:
Use:

```
npm ci
```

144. How do you list all globally installed NPM packages?

Answer:
Run:

```
npm list -g --depth=0
```

145. How do you verify the integrity of installed NPM packages?

Answer:
Run:

```
npm audit
```

146. How do you find which package introduced a security vulnerability?

Answer:
Run:

```
npm audit
```

147. How do you clean up unnecessary dependencies in NPM?

Answer:
Run: `npm prune`

148. How do you fix a package version in NPM?

Answer:
Run: `npm install package-name@version --save-exact`

149. How do you list all available Maven goals for a project?

Answer:
Run:

```
mvn help:describe -Dcmd=compile
```

or

```
mvn help:effective-pom
```

to see all project settings.

150. How do you generate a Maven dependency report in a structured format?

Answer:
Run:

```
mvn dependency:tree -DoutputFile=dependency-tree.txt
```

To generate an HTML report:

```
mvn project-info-reports:dependencies
```

151. How do you generate a Maven build report for performance analysis?

Answer:
Run:

```
mvn clean install -DbuildTimeSummary=true
```

or

```
mvn clean install -X > build.log
```

Then analyze `build.log`.

152. How do you exclude a specific plugin from a Maven build?

Answer:
Use:

```
mvn clean install -Dplugin.skip=true
```

For example, to skip the maven-compiler-plugin:

```
mvn clean install -Dmaven.compiler.skip=true
```

153. How do you define a dependency that ould only be used in a specific Maven profile?

Answer:
Define it inside a profile:

```
<profiles>
    <profile>
        <id>test-profile</id>
        <dependencies>
            <dependency>
                <groupId>org.example</groupId>
                <artifactId>test-library</artifactId>
                <version>1.0.0</version>
            </dependency>
        </dependencies>
    </profile>
</profiles>
```

154. How do you run a Maven project with a different JDK version?

Answer:
Run:

```
JAVA_HOME=/path/to/java11 mvn clean install
```

or set it in pom.:

```
<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
</properties>
```

155. How do you generate a site report for a Maven project?

Answer:
Run:

```
mvn site
```

This generates documentation in target/site.

156. How do you ensure that a Maven build fails if a dependency is missing?

Answer:
Set:

```
<failOnMissingWeb>true</failOnMissingWeb>
```

Or run:

```
mvn clean install -e
```

157. How do you enforce Java version compatibility across multiple Maven projects?

Answer:
Define the version in a parent POM:

```
<properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
</properties>
```

158. How do you create a custom Maven lifecycle phase?

Answer:
Create a custom plugin and bind it to a phase in pom.:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-antrun-plugin</artifactId>
    <executions>
        <execution>
            <phase>validate</phase>
```

```
            <goals>
                <goal>run</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

159. How do you use Maven profiles in Jenkins?

Answer:
In the Jenkins build step, use:

```
mvn clean package -Pprod
```

160. How do you make a Maven project compile faster?

Answer:
Use parallel builds:

```
mvn clean install -T 4
```

Skip tests:

```
mvn clean install -DskipTests
```

Use incremental compilation:

```
mvn clean package -Dcompile.incremental=true
```

161. How do you install a package as a peer dependency in NPM?

Answer:
Run:

```
npm install package-name --save-peer
```

162. How do you install multiple versions of the same package in an NPM project?

Answer:
Use npx:

```
npx create-react-app@4.0.0 my-app

npx create-react-app@5.0.0 my-app2
```

Or use npm alias:

```
"dependencies": {

    "react-legacy": "npm:react@16.8.0",

    "react-latest": "npm:react@18.0.0"

}
```

163. How do you generate an NPM dependency graph?

Answer:
Run:

```
npm list -- > dependency-tree.
```

164. How do you install an NPM package from a private registry?

Answer:

Run:

```
npm install package-name
--registry=https://custom-registry.com
```

Or configure .npmrc:

ini

```
registry=https://custom-registry.com

_authToken=your-token
```

165. How do you upgrade a specific dependency in NPM?

Answer:
Run:

```
npm update package-name
```

166. How do you downgrade an NPM package?

Answer:
Run:

```
npm install package-name@version
```

167. How do you link a local package in NPM for development?

Answer:
Run inside the package directory:

```
npm link
```

Then, in another project:

```
npm link package-name
```

168. How do you verify the integrity of installed NPM packages?

Answer:
Run:

```
npm audit
```

169. How do you delete all NPM dependencies and reinstall them?

Answer:
Run:

```
rm -rf node_modules package-lock.
npm install
```

170. How do you run multiple NPM scripts in sequence?

Answer:
Use `&&`:

```
npm run lint && npm run build && npm run test
```

171. How do you create an NPM workspace for managing multiple packages?

Answer:

Define workspaces in `package.`:

```
{
    "workspaces": ["packages/*"]
}
```

Install dependencies:

```
npm install
```

172. How do you completely reset NPM to its default state?

Answer:
Run:

```
npm cache clean --force
rm -rf node_modules package-lock.
npm install
```

173. How do you publi an NPM package with a specific tag?

Answer:
Run:

```
npm publi --tag beta
```

174. How do you enforce Node.js version in an NPM project?

Answer:
Specify in package.:

```
"engines": {
  "node": ">=16.0.0"
}
```

Run:

```
nvm use 16
```

175. How do you prevent breaking changes when updating dependencies?

Answer:
Use:

```
npm audit fix --force
```

or manually update versions in package..

176. How do you configure Maven to store dependencies in a custom directory instead of ~/.m2/repository?

Answer:
Use the localRepository setting in settings.:

```
<localRepository>/custom/maven/repository</localRepository>
```

Or specify it in the command line:

```
mvn clean install
-Dmaven.repo.local=/custom/maven/repository
```

177. How do you enforce a specific dependency version across all modules in a multi-module Maven project?

Answer:
Define it in dependencyManagement in the parent POM:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.12.0</version>
        </dependency>
    </dependencies>
</dependencyManagement>
```

178. How do you deploy a Maven-built JAR to an AWS S3 bucket?

Answer:
Use the Maven Wagon Plugin:

```
<distributionManagement>
    <repository>
        <id>aws-s3</id>
```

```
        <url>s3://my-bucket/releases/</url>
    </repository>
</distributionManagement>
```

Then, deploy with:

```
mvn deploy
```

179. How do you resolve `Could not resolve dependencies` error in Maven?

Answer:

Run:

```
mvn dependency:tree
```

- Manually add the correct version in `dependencyManagement`.

Run:

```
mvn clean install -U
```

180. How do you skip Maven enforcer plugin rules?

Answer:
Run:

```
mvn clean install -Denforcer.skip=true
```

181. How do you configure a Maven project to build a Docker image?

Answer:
Use Jib Plugin:

```
<plugin>
    <groupId>com.google.cloud.tools</groupId>
    <artifactId>jib-maven-plugin</artifactId>
    <configuration>
        <to>
            <image>my-docker-repo/my-app:latest</image>
        </to>
    </configuration>
</plugin>
```

Run:

```
mvn compile jib:build
```

182. How do you use a different settings. file for a Maven build?

Answer:
Run:

```
mvn clean install --settings /path/to/custom-settings.
```

183. How do you generate a custom Maven archetype?

Answer:

Run:

```
mvn archetype:create-from-project

cd target/generated-sources/archetype

mvn install
```

184. How do you configure Maven for offline builds?

Answer:
Run:

```
mvn clean install -o
```

185. How do you find the effective configuration of a Maven project?

Answer:
Run:

```
mvn help:effective-pom
```

or

```
mvn help:effective-settings
```

186. How do you install an NPM package without executing its post-install scripts?

Answer:
Run:

```
npm install package-name --ignore-scripts
```

187. How do you check if an NPM package version exists?

Answer:
Run:

```
npm ow package-name versions
```

188. How do you use a different `.npmrc` file for different projects?

Answer:
Run:

```
npm install --userconfig /path/to/custom/.npmrc
```

189. How do you remove an NPM package from `dependencies` but keep it in `devDependencies`?

Answer:
Run:

```
npm uninstall package-name --save-prod
```

190. How do you lock dependency versions in an NPM project?

Answer:
Run:

```
npm ci
```

191. How do you install a package but prevent automatic dependency resolution?

Answer:
Run:

```
npm install package-name --no-save --legacy-peer-deps
```

192. How do you determine which dependency installed another package in NPM?

Answer:
Run:

```
npm ls package-name
```

193. How do you automatically update all dependencies to their latest versions?

Answer:
Run:

```
npx npm-check-updates -u
```

```
npm install
```

194. How do you create an NPM package with TypeScript?

Answer:

Initialize a project:

```
npm init -y
```

Install TypeScript:

```
npm install typescript --save-dev
```

Generate tsconfig.:

```
npx tsc --init
```

195. How do you list all unused dependencies in an NPM project?

Answer:
Run:

```
npx depcheck
```

196. How do you install an NPM package globally for a specific user?

Answer:
Run:

```
npm install -g package-name --prefix /home/user/.npm-global
```

Then add this directory to PATH:

```
export PATH=/home/user/.npm-global/bin:$PATH
```

197. How do you debug an NPM package installation failure?

Answer:
Run:

```
npm install package-name --verbose
```

or check logs:

```
npm install package-name 2> error.log
```

198. How do you publi an NPM package to a custom registry?

Answer:
Set the registry:

```
npm set registry https://custom-registry.com
```

Then publi:

```
npm publi --registry=https://custom-registry.com
```

199. How do you prevent an NPM package from being automatically updated?

Answer:
Use:

```
"dependencies": {
  "package-name": "1.2.3"
}
```

or add:

```
"package-name": {
  "version": "1.2.3",
  "resolved":
```

```
"https://registry.npmjs.org/package-name/-/package-name
-1.2.3.tgz",
  "integrity": "a512-abc123..."
}
```

in package-lock..

200. How do you automatically fix security vulnerabilities in NPM packages?

Answer:
Run:

```
npm audit fix
```

For forced updates:

```
npm audit fix --force
```