MAVEN:

At first glance Maven can appear to be many things, but in a nutshell Maven is an attempt *to apply patterns to a project's build infrastructure in order to promote comprehension and productivity by providing a clear path in the use of best practices*. Maven is essentially a project management and comprehension tool and as such provides a way to help with managing:

- Builds
- Documentation
- Reporting
- Dependencies
- SCMs
- Releases
- Distribution

1. **How does Maven manage dependencies and transitive dependencies?** Ans: Maven manages dependencies and transitive dependencies through the use of a Project Object Model (POM) file. This file contains information about the project, including a list of dependencies and their versions. When a project depends on another project, Maven will automatically include that project's dependencies (transitive dependencies) in the final build, eliminating the need for manual management of these dependencies. Maven uses a repository to store these dependencies, and it will automatically download them from the repository when they are needed. Additionally, Maven uses a dependency hierarchy to resolve conflicts between different versions of the same dependency, using the "nearest definition" strategy.

2. **How to install Maven on a Linux machine?**
   Ans: **Click Here for Answer with code & Screenshots**

3. **What is a SNAPSHOT version?**

Ans:

Notice the value of the **version** tag in the `pom.xml` file shown below has the suffix: `-SNAPSHOT`.

```
1. <project xmlns="http://maven.apache.org/POM/4.0.0"
2.   ...
3.   <groupId>...</groupId>
4.   <artifactId>my-app</artifactId>
5.   ...
```

```
6.  <version>1.0-SNAPSHOT</version>
7.  <name>Maven Quick Start Archetype</name>
8.  ...
```

The `SNAPSHOT` value refers to the 'latest' code along with a development branch, and provides no guarantee the code is stable or unchanged. Conversely, the code in a 'release' version (any version value without the suffix `SNAPSHOT`) is unchanging.

In other words, a SNAPSHOT version is the 'development' version before the final 'release' version. The SNAPSHOT is "older" than its release.

During the release process, a version of **x.y-SNAPSHOT** changes to **x.y**.

The release process also increments the development version to **x.(y+1)-SNAPSHOT**. For example, version **1.0-SNAPSHOT** is released as version **1.0**, and the new development version is version **1.1-SNAPSHOT**.

3.What is the purpose of mvn clean command?

Ans: This command removes the target directory before the starting of a build process.

4. **How to integrate Maven pom.xml with Jenkins job?**

Click Here For Steps with Screenshot

5. List out the dependency scope in Maven?

The various dependency scopes used in Maven are:

• Compile: It is the default scope, and it indicates what dependency is available in the classpath of the project
• Provided: It indicates that the dependency is provided by JDK or web server or container at runtime
• Runtime: This says that the dependency is not needed for compilation but is required during execution
• Test: It says dependency is available only for the test compilation and execution phases
• System: It indicates you have to provide the system path
• Import: This indicates that the identified or specified POM should be replaced with the dependencies in that POM's section

6. In Maven, what are the two settings files called and what are their locations?

In Maven, the settings files are called settings.xml, and the two settings files are located at

• Maven installation directory: $M2_Home/conf/settings.xml
• User's home directory: ${ user.home }/ .m2 / settings.xml

7. Maven dependencies with pom.xml for Automation Framework design? How to add selenium, testng,cucumber dependency in Maven?

Click Here For POM.xml

This has the entire list dependencies required to design a Selenium Automation Framework from scratch.

8. **Dependency vs plugin**

Both plugins and dependencies are Jar files.

But the difference between them is, most of the work in maven is done using plugins; whereas dependency is just a Jar file which will be added to the classpath while executing the tasks.

So, we can say, plugin is a Jar file which executes the task, and dependency is a Jar which provides the class files to execute the task.

9. Group id vs artifact id

groupId will identify your project uniquely across all projects, so we need to enforce a naming schema. You have to follow the package name rules, which means that it has to be at least as a domain name you control, and you can create as many subgroups as you want.
eg. org.apache.maven, org.apache.commons

artifactId is the name of the jar without version. If you created it then you can choose whatever name you want with lowercase letters and no strange symbols. If it's a third party jar you have to take the name of the jar as it's distributed.
eg. maven, commons-math

10. What is the default location for your local repository?

Answer: ~ / M2 / repository.

11. Maven plugins?

1. surefire plugin
2. compiler plugin
3. resource plugin

12. What is the default scope that the maven uses if we do not specify the scope element in pom.xml?

Answer: Compile

How do I create documentation?

To get you jump started with Maven's documentation system you can use the archetype mechanism to generate a site for your existing project using the following command:

```
mvn archetype:generate \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DarchetypeArtifactId=maven-archetype-site \
  -DgroupId=com.mycompany.app \
  -DartifactId=my-app-site
```

13. What scope can be used to make certain dependencies available only while compiling and running tests?

Answer: Test

14. What is archetype?

Answer: We will use maven archetype to generate the maven folder structure based on the inputs.

Ex: mvn archetype:generate.

What is Apache Maven, and why is it used?

> Apache Maven is a build automation tool used for managing and building projects. It simplifies project configuration, dependencies, and builds.

What is a POM file in Maven?

POM (Project Object Model) is an XML file that contains project configuration details, including dependencies, plugins, and build instructions.

How do you create a new Maven project using the command line?

To create a new Maven project, use the following command:
arduino

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

What is a Maven repository, and how does it work?

A Maven repository is a directory where Maven stores project dependencies. There are local repositories on your machine and remote repositories on the internet.

How can you specify dependencies in a Maven POM file?

You can specify dependencies in the `<dependencies>` section of the POM file. For example:

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

What is a Maven plugin, and how is it used?

A Maven plugin is an extension that provides additional functionality to Maven. Plugins are configured in the POM file and executed during the build process.

How do you build a Maven project from the command line?

To build a Maven project, use the following command:

```
mvn clean install
```

Explain the Maven build phases and their order.

Maven build phases include `validate`, `compile`, `test`, `package`, `verify`, `install`, and `deploy`, in that order. Each phase builds upon the previous one.

How do you skip a specific Maven build phase?

To skip a specific build phase, use the `-Dskip.phaseName` or `-Dmaven.test.skip=true` option. For example:

```
mvn clean install -DskipTests
```

What is the purpose of the `settings.xml` file in Maven?

The `settings.xml` file is used to configure Maven settings, such as repository locations, profiles, and authentication credentials.

What is a Maven profile, and how do you activate it?

A Maven profile is a set of configuration settings that can be activated based on conditions. You can activate a profile using the `-P` option. For example:

```
mvn clean install -PprofileName
```

What is the purpose of the `mvn clean` command, and when should it be used?

The `mvn clean` command removes the target directory and any compiled artifacts. It's useful when you want to start fresh or clean up your project.

How can you create a custom Maven archetype?

To create a custom archetype, you can use the `mvn archetype:create-from-project` command and then package it as a new archetype project.

Explain the difference between the `compile` and `test` scopes for dependencies.

Dependencies with the `compile` scope are needed for compilation and runtime, while those with the `test` scope are only needed for testing.

What is the Maven Central Repository, and why is it important?

The Maven Central Repository is a global repository of open-source artifacts. It's essential for sharing and accessing commonly used libraries.

How do you handle dependency conflicts in Maven?

> You can use the `<exclusions>` element to exclude specific transitive dependencies or use the `<dependencyManagement>` section to specify preferred versions.

What is the purpose of the `maven-surefire-plugin` in Maven?

> The `maven-surefire-plugin` is used for executing unit tests. It automatically runs test classes whose names match the `*Test` pattern.

How do you generate a Maven project's site documentation?

> To generate site documentation, use the `mvn site` command. It generates HTML-based documentation in the `target/site` directory.

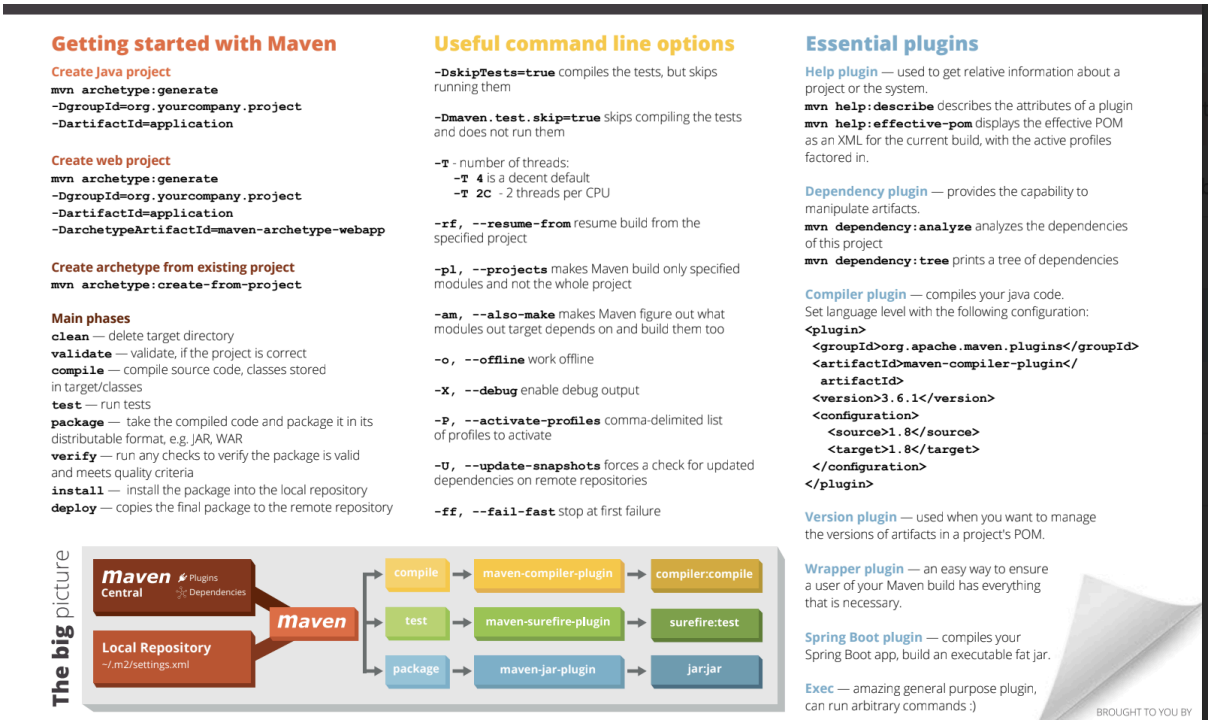
What is the Maven assembly plugin, and how is it used?

> The Maven assembly plugin is used to create distributions or assemblies of your project, such as JARs with dependencies or ZIP files.

How do you deploy a Maven project to a remote repository?

> To deploy a Maven project to a remote repository, use the `mvn deploy` command. Ensure that your `settings.xml` file has the necessary credentials and repository configuration.

CHEATSHEET



Getting started with Maven

Create Java project

```
mvn archetype:generate
-DgroupId=org.yourcompany.project
-DartifactId=application
```

Create web project

```
mvn archetype:generate
-DgroupId=org.yourcompany.project
-DartifactId=application
-DarchetypeArtifactId=maven-archetype-webapp
```

Create archetype from existing project

```
mvn archetype:create-from-project
```

Main phases

clean — delete target directory
validate — validate, if the project is correct
compile — compile source code, classes stored in target/classes
test — run tests
package — take the compiled code and package it in its distributable format, e.g. JAR, WAR
verify — run any checks to verify the package is valid and meets quality criteria
install — install the package into the local repository
deploy — copies the final package to the remote repository

Useful command line options

-DskipTests=true compiles the tests, but skips running them

-Dmaven.test.skip=true skips compiling the tests and does not run them

-T - number of threads:
-T 4 is a decent default
-T 2C - 2 threads per CPU

-rf, --resume-from resume build from the specified project

-pl, --projects makes Maven build only specified modules and not the whole project

-am, --also-make makes Maven figure out what modules out target depends on and build them too

-o, --offline work offline

-X, --debug enable debug output

-P, --activate-profiles comma-delimited list of profiles to activate

-U, --update-snapshots forces a check for updated dependencies on remote repositories

-ff, --fail-fast stop at first failure

Essential plugins

Help plugin — used to get relative information about a project or the system.
mvn help:describe describes the attributes of a plugin
mvn help:effective-pom displays the effective POM as an XML for the current build, with the active profiles factored in.

Dependency plugin — provides the capability to manipulate artifacts.
mvn dependency:analyze analyzes the dependencies of this project
mvn dependency:tree prints a tree of dependencies

Compiler plugin — compiles your java code.
Set language level with the following configuration:

```
<plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</
  artifactId>
 <version>3.6.1</version>
 <configuration>
   <source>1.8</source>
   <target>1.8</target>
 </configuration>
</plugin>
```

Version plugin — used when you want to manage the versions of artifacts in a project's POM.

Wrapper plugin — an easy way to ensure a user of your Maven build has everything that is necessary.

Spring Boot plugin — compiles your Spring Boot app, build an executable fat jar.

Exec — amazing general purpose plugin, can run arbitrary commands :)

SCENARIO BASED REAL TIME

- **Question:** How do you resolve dependency conflicts in a Maven project?

  Answer: Maven uses the nearest definition strategy, where the version of the dependency declared closest to the project in the dependency tree takes precedence. Understanding the dependency hierarchy and utilizing the `dependency:tree` command helps identify and resolve conflicts.

- **Question:** Explain how you would perform an offline build using Maven.

  Answer: Maven allows offline builds by using the `-o` or `--offline` option. This prevents Maven from attempting to connect to remote repositories, relying on the locally cached dependencies. This is useful in scenarios with limited or no internet connectivity.

- **Question:** Can you customize the Maven build lifecycle, and if so, provide an example?

  Answer: Yes, the build lifecycle can be customized using plugins. For instance, the `exec-maven-plugin` allows executing arbitrary commands during the build process. Configuring it in the `<build>` section of the POM file enables the execution of custom scripts or commands.

- **Question:** What are Maven snapshot releases, and when should they be used?

  Answer: Snapshot releases are versions of a project that are in development. They have a unique version number with the "-SNAPSHOT" suffix. Snapshots are useful during development to allow continuous integration systems to pick up the latest changes. However, they should be avoided in release versions.

- **Question:** How do you handle a multi-module Maven project, and what benefits does it offer?

  Answer: In a multi-module project, Maven can build multiple projects within the same hierarchy. Each module represents a different subproject, and the parent POM coordinates the build process. This structure promotes code organization, reuse, and simplifies the management of complex projects.