***What is Maven Life Cycle?***

The default Maven lifecycle consists of 8 major steps or phases for compiling, testing, building and installing a given Java project as specified below:

1. **Validate:** This step validates if the project structure is correct. For example – It checks if all the dependencies have been downloaded and are available in the local repository.
2. **Compile:** It compiles the source code, converts the .java files to .class and stores the classes in target/classes folder.
3. **Test:** It runs unit tests for the project.
4. **Package:** This step packages the compiled code in distributable format like JAR or WAR.
5. **Integration test:** It runs the integration tests for the project.
6. **Verify:** This step runs checks to verify that the project is valid and meets the quality standards.
7. **Install:** This step installs the packaged code to the local Maven repository.
8. **Deploy:** It copies the packaged code to the remote repository for sharing it with other developers.

***Maven Commands:***

* **mvn clean:** Cleans the project and removes all files generated by the previous build.
* **mvn compile:** Compiles source code of the project.
* **mvn test-compile:** Compiles the test source code.
* **mvn test:** Runs tests for the project.
* **mvn package:** Creates JAR or WAR file for the project to convert it into a distributable format.
* **mvn install:** Deploys the packaged JAR/ WAR file to the local repository.
* **mvn deploy:** Copies the packaged JAR/ WAR file to the remote repository after compiling, running tests and building the project.

***Describe:***

* 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.

***What is Web Services?***

***A web service is a software system that supports interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically, web Service Definition Language, or WSDL). web services fulfill a specific task or a set of tasks.***

***What is rest controller?***

RestController is a Spring annotation that is used to build REST API in a declarative way. RestController annotation is applied to a class to mark it as a request handler, and Spring will do the building and provide the RESTful web service at runtime

***.Web Methods:***

***1: GET*** *:* It requests are the most common and widely used methods in APIs and websites. Simply put, the GET method is used to **retreive data from a server at the specified resource**.

***2: POST:*** In web services, POST requests are used to **send data to the API server** to create or update a resource. The data sent to the server is stored in the [request body](https://stackoverflow.com/questions/22034144/what-does-it-mean-http-request-body) of the HTTP request.

***3: PUT:***  Simlar to POST, PUT requests are used to send data to the API to **update or create a resource**. The difference is that **[PUT requests are idempotent](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT)**. That is, calling the same PUT request multiple times **will always produce the same result**. In contrast, calling a POST request repeatedly make have side effects of creating the same resource multiple times.

***4:DELETE:*** The DELETE method is exactly as it sounds: **delete the resource at the specified URL**. This method is one of the more common in RESTful APIs so it's good to know how it works.

***GIT Life Cycle:***

* ***In Step – 1****, We first clone any of the code residing in the remote repository to make our own local repository.*
* ***In Step-2****we edit the files that we have cloned in our local repository and make the necessary changes in it.*
* ***In Step-3****we commit our changes by first adding them to our staging area and committing them with a commit message.*
* ***In Step – 4 and Step-5****we first check whether there are any of the changes done in the remote repository by some other users and we first pull that changes.*
* *If there are no changes we directly proceed with****Step – 6****in which we push our changes to the remote repository and we are done with our work.*

***Describe the git commands***:

**. Git clone**

Git clone is a command for downloading existing source code from a remote repository (like Github, for example). In other words, Git clone basically makes an identical copy of the latest version of a project in a repository and saves it to your computer.

## . Git branch

Branches are highly important in the git world. By using branches, several developers are able to work in parallel on the same project simultaneously. We can use the git branch command for creating, listing and deleting branches.

## . Git checkout

This is also one of the most used Git commands. To work in a branch, first you need to switch to it. We use **git checkout** mostly for switching from one branch to another. We can also use it for checking out files and commits.

## . Git add

When we create, modify or delete a file, these changes will happen in our local and won't be included in the next commit (unless we change the configurations).

## . Git commit

This is maybe the most-used command of Git. Once we reach a certain point in development, we want to save our changes (maybe after a specific task or issue).

## Git push

After committing your changes, the next thing you want to do is send your changes to the remote server. Git push uploads your commits to the remote repository.

## Git pull

The **git pull**command is used to get updates from the remote repo. This command is a combination of **git fetch** and **git merge** which means that, when we use git pull, it gets the updates from remote repository (git fetch) and immediately applies the latest changes in your local (git merge).

## Git revert

Sometimes we need to undo the changes that we've made. There are various ways to undo our changes locally or remotely (depends on what we need), but we must carefully use these commands to avoid unwanted deletions.

## Git merge

When you've completed development in your branch and everything works fine, the final step is merging the branch with the parent branch (dev or master). This is done with the git merge command.

**Git Init :** Initializes a new Git repository. If you want to place a project under revision control, this is the first command you need to learn.

**Git Rebase:** The -i flag is used to begin an interactive rebasing session. This provides all the benefits of a normal rebase, but gives you the opportunity to add, edit, or delete commits along the way.

**Git Reset** : Undoes changes to files in the working directory. Resetting lets you clean up or completely remove changes that have not been pushed to a public repository.

***Centralized Vs Distributed version control:***

The **main difference** between centralized and distributed version control is that, **in centralized version control, the versions are saved in the remote repository, while in distributed version control, versions can be saved in the remote repository as well as in local repositories of the local machines.**