Maven & Git IQ

1. **How can you make sure that all team members are using the same selenium version?**

   We can use tools like Maven that give version and build control for team perspective we have same version in pom.xml

2. **What is Maven?**

   Maven is a build automation tool or a project management tool. In Maven we have many inbuilt templates. These templates are called archetypes.
   As an automation engineers using Maven tool, we can create project structure, manage project required dependencies and execute script from command line

3. **What is POM.XML?**

   POM stands for Project Object Model. The pom.xml file contains information of project and project configuration information for maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc. Maven reads the pom.xml file, then executes the goal.

4. **What is Maven Repository? What are their types?**

   Maven repository is a location where all the project jars, library jars, plugins or any other particular project related artifacts are stored and can be easily used by Maven.

   Maven repositories:
   - local - repository inside your computer (.M2)
   - central (https://mvnrepository.com/)
   - remote - developer's own custom repository containing required libraries or other project jars. For example, in your corporate office there may be projects or modules specific to organization only. In this case, organization can create remote repository and deploy these private artifacts. This remote repository will be accessible only inside organization.

5. **What is dependency , how to handle dependencies in framework?**

In maven we are having this dependencies ,dependency is just a Jar file which will be added to the classpath while executing the tasks we are going to update the dependencies in the POM.xml which is heart of maven project.
We are using dependencies in our framework because of compatibility issues as we work in the team. For example if the selenium webdriver version which we are working is not supporting we can simply update the dependency file as given below

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
   <groupId>org.seleniumhq.selenium</groupId>
   <artifactId>selenium-java</artifactId>
   <version>3.9.1</version>
</dependency>
```

Here we can simply change the version according the compatibility

6. How do you submit plugins in Maven? What are they? Version?

Maven Plugins are generally used to:

- Maven compiler plugin - compile code files
- Maven surefire plugin - is used during the test phase of the build lifecycle to execute the unit tests of an application

A plugin provides a set of goals, which can be executed using the following syntax

mvn [plugin-name]:[goal-name]

```
<plugin>
    <groupId>net.masterthought</groupId>
    <artifactId>maven-cucumber-reporting</artifactId>
    <version>3.15.0</version>
    <executions>
        <execution>
            <id>execution</id>
            <phase>verify</phase>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <projectName>OrangeHRM</projectName>
                <!-- path to the report that will be generated -->
                <outputDirectory>${project.build.directory}/cucumber-html-reports</outputDirectory>
                <!-- path to the json file that will be used to genetrate report -->
                <!-- it has to match the one on the testrunner -->
                <cucumberOutput>${project.build.directory}/cucumber.json</cucumberOutput>
            </configuration>
        </execution>
    </executions>
</plugin>
```

7. List out some Maven build phases?

Following are the phases:

- **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.
- **integration-test** − process and deploy the package if necessary into an environment where integration tests can be run.
- **verify** − run any checks to verify the package is valid and meets quality criteria.
- install − install the package into the local repository, for use as a dependency in other projects locally.
- **deploy** − done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

8. What type of version control do you use?

Version Control is essential to development, even if you're working by yourself because it protects you from yourself. If you make a mistake, it's a simple matter to rollback to a

previous version of your code that you know works. This also frees you to explore and experiment with your code because you're free of having to worry about whether what you're doing is reversible or not.

We have used maven for build generation and git for version control.

9. What is GIT?

GIT is a distributed version control system and source code management (SCM) system with an emphasis to handle small and large projects with speed and efficiency.

10. What is the function of git clone?

The git clone command creates a copy of an existing Git repository. To get a copy of a central repository, 'cloning' is the most common way used by programmers.

11. What is the function of 'git config'?

The 'git config' command is a convenient way to set configuration options for your Git installation. Behaviour of a repository, user info, preferences etc. can be defined through this command.

12. How can you create a repository in Git?

In Git, to create a repository, create a directory for the project if it does not exist, and then run command "git init". By running this command .git directory will be created in the project directory, the directory does not need to be empty.

13. What is a 'conflict' in git?

A 'conflict' arises when the commit that has to be merged has some change in one place, and the current commit also has a change at the same place. Git will not be able to predict which change should take precedence.

14. What is 'git status' is used for?

As 'Git Status' shows you the difference between the working directory and the index, it is helpful in understanding a git more comprehensively.

15. Explain GitHub Workflow?

GIT provides three key areas that are uniquely designed, to give developers lots of control over workflow:
1.Working directory: It contains all the current states of files. Numerous developers can access directory when they are logged in, so collaboration is extremely easy.

2.Staging Area: It indexes everything for the next commit and any files that have been added or edited since the previous save.

3.GIT repository is a dedicated space where new commits are added:GIT repository maintains all the metadata, the files, and a dedicated database that tracks versions of the project.

16. What are the advantages of using GIT?

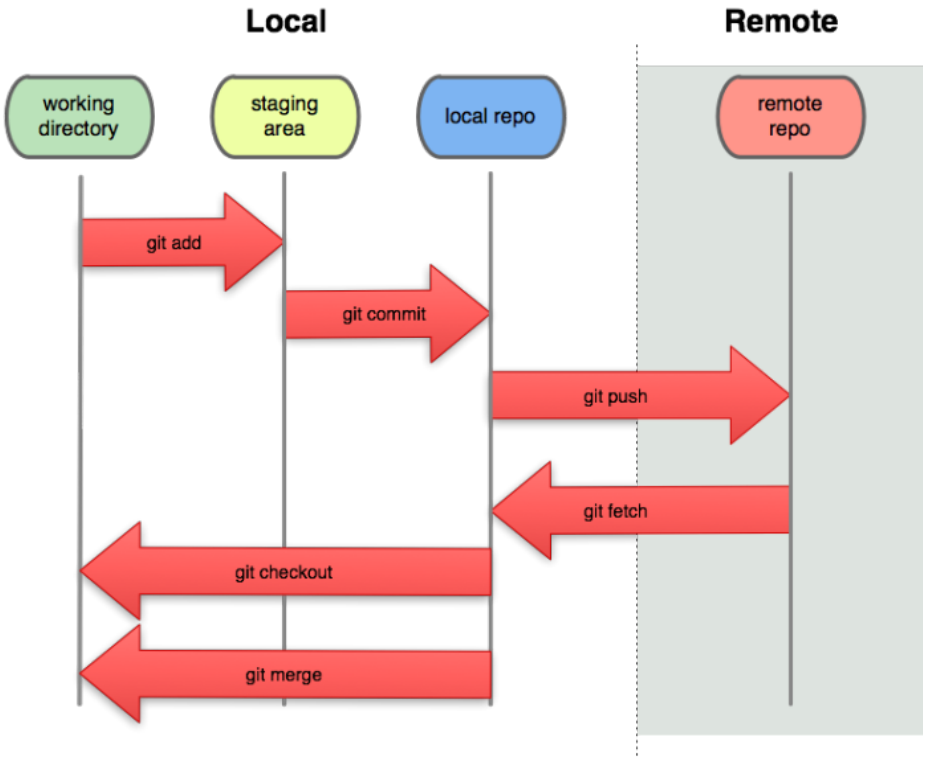
The Advantages of using GIT are:

- Data redundancy and replication
- Any sort of projects can use GIT
- High availability
- •Only one .git directory per repository
- Superior disk utilization and network performance
- Collaboration friendly

17. What are the commands for your version control?

- git add
- git status
- git commit
- git push
- git pull

18. What is the difference between local vs remote repositories? (Git vs GitHub)



19. What is the difference between commit and push?

git commit "records changes to the repository" while git push “updates remote repository”

20. How do you see who updated the class file using GIT?

Using “git log” command

21. What is staging in GIT?

The staging area is a file, generally contained in your Git directory that stores information about what will go into your next commit. Its technical name in Git parlance is the “index”, but the phrase “staging area” works just as well.

The basic Git workflow goes something like this:

1. You modify files in your working tree.
2. You selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

22. Simple rules for less merge conflicts

Updating your master branch

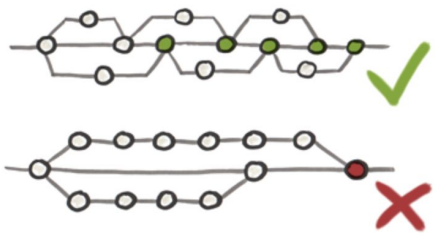
By simply updating your master branch, every morning using the command “git pull” you will insure your master branch has the most updated changes from the web repositories.

Updating your branch with master branch

Merging master branch into your active branch insures, your active branch has the latest changes from the master branch.

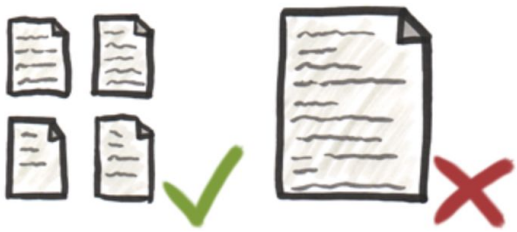
Short-Living Branches

Many big merge conflicts are results of long-living branches. If you work in your own branch for several days, or even weeks, the risk is high that someone else changes parts of the code you have touched in another branch. That would lead to merge conflicts. On the other hand, if you have a short-living branch, meaning it is merged back into the trunk or master branch after a few hours, there will probably be no merge conflict. Even if there is a conflict, it can usually be resolved very quickly because there is much less code to think about. The following figures both show a master branch (the line in the middle) and some kind of feature branches. The difference between those figures is that the first one contains many short-living branches while the second one has only two long-living feature branches. The merge commits in the first figure are always green because the short branches have not caused any merge conflicts. In the second figure we have a red dot at the end which represents a merge conflict that was caused by two of the many commits in the long-living branches.



Small Modules

The Single Responsibility Principle (which is part of the SOLID principles of software design) says that “a class should have one, and only one, reason to change“. If we take this approach serious, there should never be a situation in which two developers change the same code at the same time because that would mean they are working on the same task. You see, a modular architecture and small classes do not only help to increase the quality of the code design but also decrease the chance for merge conflicts.



Strong Communication

“Communication is the key” – We have heard this phrase so many times… and it’s true! If all your teammates know what you are working on and which parts of the code you are touching they will try not to change the same code at the same time. If another developer wants to change a part of the code base which you are working on then it would probably be a good idea to work together