**git**

**git Branch**

A branch is an independent development route, which can later be connected back to the main branch. We use git branch command to **create**, **list**, **rename** and **delete**branches.

**When do we use branches in git?**

1. When adding a small or big feature
2. When testing some logic and do not want bad code to appear in the master branch.
3. To separate releases
4. When fixing bugs or assigned issues etc.

**Branch types**

1. Master branch: This branch holds the latest version of fully working code, which is the current code in production - simply, it only tracks released code only. You can see that the only merges to master come from release and hotfixes branches.
2. Develop branch: This branch is created from the master branch - it complements the default master branch. It is the integration branch for features.
3. Feature branch: When you begin a feature, you create a branch off the develop branch, and once the feature is done, you merge back to develop.
4. Release branch: A release branch is created off the develop branch after several features have been completed and ready for release.
5. Hotfixes branch: If an issue is detected on master, a hotfix branch is created off the master branch. Once the hotfix is complete, it is merged to both develop and master branches.
6. Bugfix/issue: If some issues were detected in a release branch while testing, small branches are created to fix the issues. Such branches are merged back the release branch.

**git Tagging**

The purpose of tagging is usually to take note of specific points in the repository history as being important. It is mostly used to mark release points.

It can be done as follows:

* Checkout to master
* Merge release to master
* Tag the release for future reference.

You can checkout to a **tag**just the way you do to a branch. This will allow you to see the status of your source code at a point when that tag was created.

This also puts your repository in a detached HEAD state, which means you can play around with the code in a tag branch without affecting your real code.

**git hot fix**

If you discover a bug in the production code, a hot fix will have to be created to solve the problem as follows:

* Create a hotfix/1.0.1
* Fix the bug and commit changes.
* Merge to master and develop branches. Remember to tag when you merge to master.

**Handling git merge conflict**

git conflicts can occur when you:

* Pull from a remote repository.
* Merge a branch.

When a git conflict happens, git will show you the **2 sections**, from which you need to choose the correct one.  It is up to you to decide what to keep and what to discard.

Resolving a conflict can take from just a minute to days. To avoid conflict resolutions taking a long time, it is advisable to commit, pull, push and merge more often.