**GIT New Branching Strategy and Industry Practice**

The main branches

The Central repo holds two Main branches with an infinite life time:

1. Master
2. Development

Master:

CM team considered that, origin/master to be the main branch where the source code of HEAD always reflects a production-ready state. No development is done on this branch. It should always be stable and contains the all history of the project with tags. Every production release should be tagged. Master should be blocked by other developer commits except merges from the CM team.

Development:

CM team considered that, origin/development to be the main branch where the source code of HEAD always reflects a state with the latest delivered development changes for the next release.

Supporting branches

These branches have the limited life time, can be deleted once work to be done.

1. Feature
2. Release (IV&V branch)
3. Hotfix

Feature:

This branch cut off from development and merge back into development branch and naming convention should be feature/CQ\_ID. Feature branches (or sometimes called topic branches) are used to develop new features for the upcoming or a distant future release. When starting development of a feature, the target release in which this feature will be incorporated may well be unknown at that point. a feature branch is, it exists as long as the feature is in development, but will eventually be merged back into development. Feature branches typically exist in developer repos only, not in origin, we can maintain them in origin too if we want too.

Release (IV&V branch):

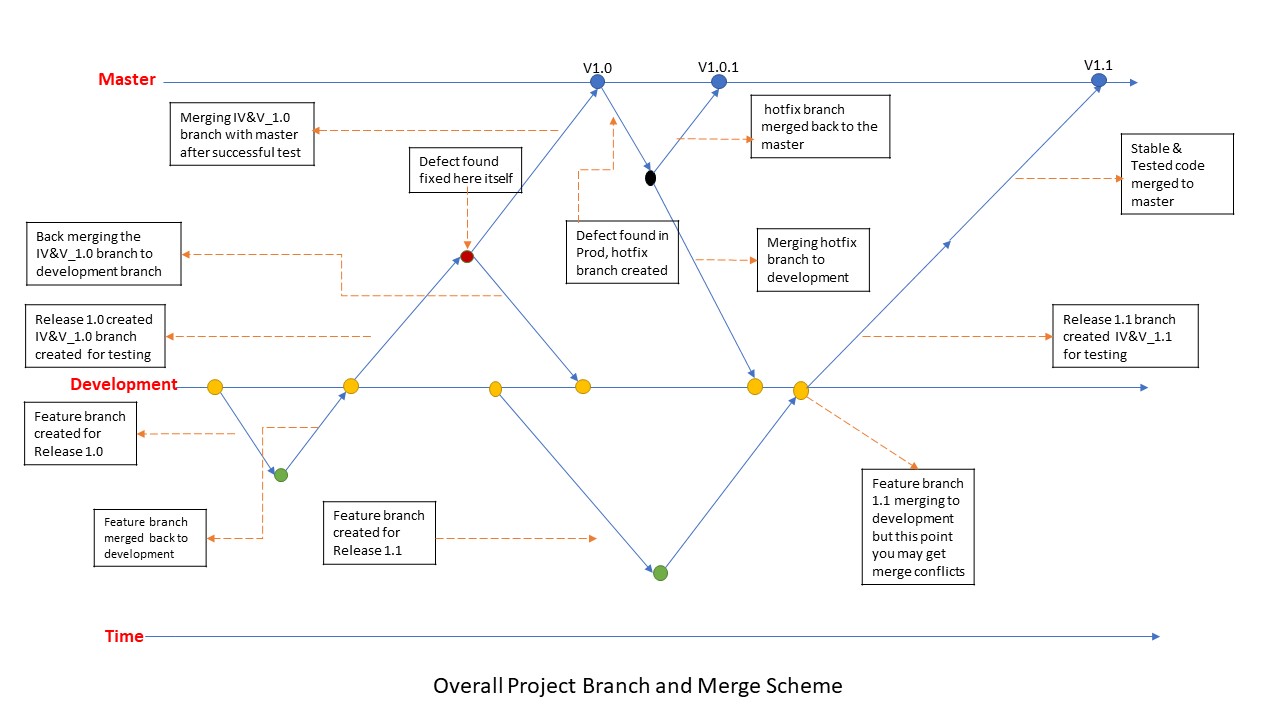
This branch cut off from development and merge back into development and master as well (after IV&V pass) and naming convention should be release-major release number.minor rlease number (Eg.1.1). (Current naming convention for Platform release branches are IVV-PLT-YEAE.DATA.TIME e.g., IVV-PLT-2020.0316.1200)Release branches support preparation of a new production release, they allow for minor bug fixes and preparing meta-data for a release. By doing all of this work on a release branch, the development branch is cleared to receive features for the next big release. The key moment to branch off a new release branch from development is when develop (almost) reflects the desired state of the new release. At least all features that are targeted for the release-to-be-built must be merged in to develop at this point in time. All features targeted at future releases may not, they must wait until after the release branch is branched off.

IV&V test their Integration and validation from the release branch for that particular release, once it is passed we can directly release this package to the production (because it is tested package) and can merge this with master and tag it. Later we can delete this branch. As we do not have any of the actual production releases, so we need not to delete these branches we can keep them for the future references.

Hotfix:

This branch cut off from Master and merge back into development and master as well. Naming convention should be release-major release\_number.minor\_rlease\_number.hotfix\_number number (Eg.1.1.1). They arise from the necessity to act immediately upon an undesired state of a live production version. When a critical bug in a production version must be resolved immediately, a hotfix branch may be branched off from the corresponding tag on the master branch that marks the production version. The essence is that work of team members (on the development branch) can continue, while another person is preparing a quick production fix. Hotfix branch is same as release (IV&V branch) so IV&V team is responsible for testing.

Check the detailed pictures of the git workflow below.



The above picture explains the overall project branch and merge scheme.

The description of the picture is:

1. Feature branch created from the development branch for Release 1.0
2. Feature branch is merged back to development after work done.
3. Now IV&V\_1.0 branch is created for release 1.0
4. Any defect found in IV&V\_1.0 branch during testing the defect is fixed in the IV&V\_1.0 branch and that fix is merged back to development.
5. Merging IV&V\_1.0 branch into Master after all teams signoff
6. Defect found in production, now a hotfix branch is created and fixed. The hotfix branch is merged back to master and development as well.
7. Now another feature branch is created from development for release 1.1 but while merging this feature branch to development you may get merge conflicts because we had merged hotfix branch to the development earlier so developers should be careful and be ready to resolve those merge conflicts. Hotfix defects pushed into development and feature branches too. Developers should be responsible for resolving merge conflicts.

What is the existing process and why we need this new process?

There is no consistency in existing process, some repositories have staging branch some don’t, CM needs all repositories to be ideal with once consistent process. As per our observation, some teams are working on Master branch, but CM team don’t want master branch to be a workable area. We want to freeze this branch and keep Master as a production-ready state.

CM team also removing the single staging branch here (As per old MCA configuration Management plan) instead, we are adding the multiple release branches the reason behind this is we are mixing features at different stages of completion. This will inhibit the ability to release what is ready. If we have the release branches for each release, then that will help to keep the clean track of each release.

The advantage of the multiple release branch situation is that you can easily hot fix or merge fixes into these release branch without having all the additional development code involved in the equation

Why do we need the Master branch?

This is a highly stable branch that is always production-ready and contains the last release version of source code in production. Development takes months and months and different people work on different functionalities, we don’t want the master to be messed up, we are giving development branch to work the teams on their stuff. When we merge all together then test it, then it is called as a product, after that we release the product to the market. There might be an issue in product, there will be some bugs or hotfixes from the customer, we can’t work directly on the Master, because Master will be the single source of truth. If we work directly on the master, we will miss the track, we need a backup to work on those enhancements.