**Git Branching Strategy**

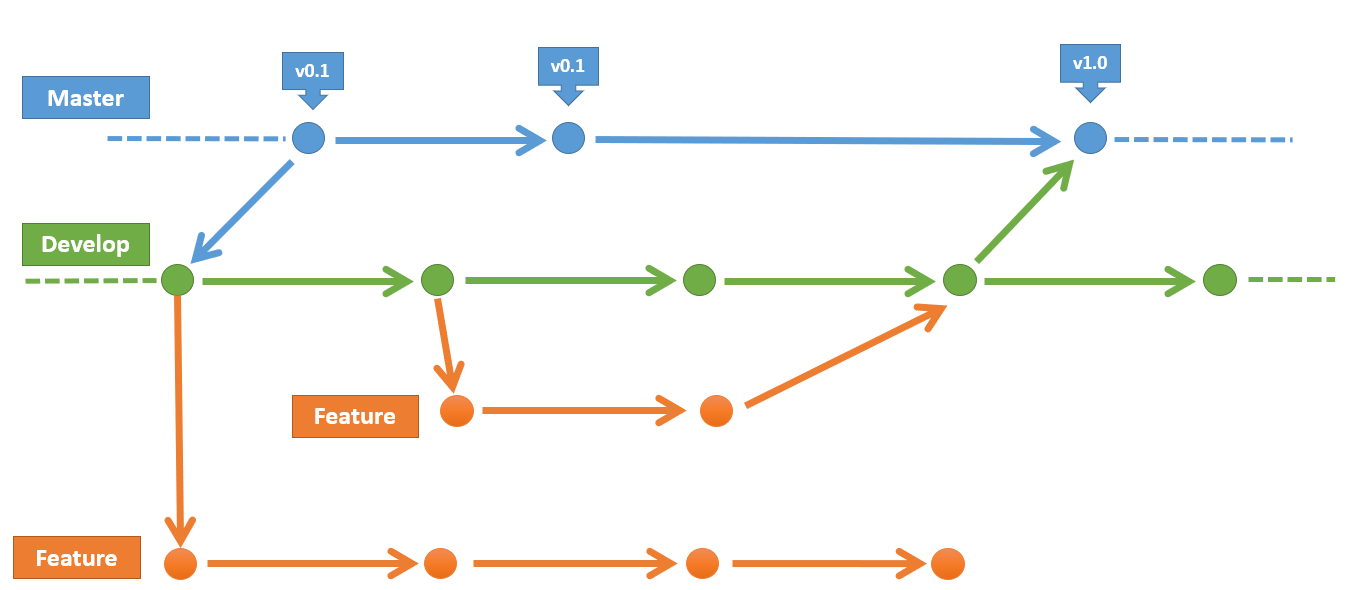
Next to the main branches master and develop, our development model uses a variety of supporting branches to aid parallel development between team members, ease tracking of features, prepare for production releases and to assist in quickly fixing live production problems. Unlike the main branches, these branches always have a limited life time, since they will be removed eventually.

The different types of branches we may use are:

* Feature branches
* Release branches
* Hotfix branches

Each of these branches have a specific purpose and are bound to strict rules as to which branches may be their originating branch and which branches must be their merge targets. We will walk through them in a minute.

Below mentioned is one of the most common and efficient Git workflows. With this one we can easily manage feature, develop and master branches all together in a particular flow. Changes can easily move from one branch to another with merge and rebase features.



**Git Branching**

Git branching helps us to manage the branches in Git repositories. We can create, switch and delete these branches according to our requirements.

**List Branch:** We can list branches in an existing repository using command:

git branch –l

**Create Branch:** We can create branch in Git to support parallel and independent programming. Use below command to create branch.

git branch <branch\_name>

**Delete Branch:** Any time we can delete obsolete branches from our repository so that our repo can be light weight.

git branch –D <branch\_name>

**Switch Branch:** We can switch between branches so that we can work on different streams of source code. It helps us to move from one branch to another.

git checkout <branch\_name>

**Creating a feature/story branch**



Feature branches (or sometimes called story 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.

When starting work on a new story, branch off from the develop branch.

**$ git checkout -b story1 develop**

**Switched to a new branch "story1"**

Now Developer will be working on this individual story branches and performs

Check-ins to these respective branches as per below flow:

**$ git checkout develop**

**Switched to branch 'develop'**

**$ git merge --no-ff story1**

**Updating ea1b82a..05e9557**

**(Summary of changes)**

**$ git branch -d story1**

**Deleted branch story1 (was 05e9557).**

**$ git push origin develop**

The **--no-ff** flag causes the merge to always create a new commit object, even if the merge could be performed with a fast-forward. This avoids losing information about the historical existence of a feature branch and groups together all commits that together added the feature. Compare:



**Creating a Hotfix branch**

Hotfix branches are very much like release branches in that they are also meant to prepare for a new production release, albeit unplanned. 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.

**$ git checkout -b hotfix-1.2.1 master**

**Switched to a new branch "hotfix-1.2.1"**

**$ git commit -m "Fixed severe production problem"**

**[hotfix-1.2.1 abbe5d6] Fixed severe production problem**

**5 files changed, 32 insertions(+), 17 deletions(-)**

**NOTE: When finished, the bugfix needs to be merged back into master, but also needs to be merged back into develop, in order to safeguard that the bugfix is included in the next release as well.**

**$ git checkout master**

**Switched to branch 'master'**

**$ git merge --no-ff hotfix-1.2.1**

**Merge made by recursive.**

**(Summary of changes)**

**$ git tag -a 1.2.1**

Next, include the bugfix in develop, too:

**$ git checkout develop**

**Switched to branch 'develop'**

**$ git merge --no-ff hotfix-1.2.1**

**Merge made by recursive.**

**(Summary of changes)**