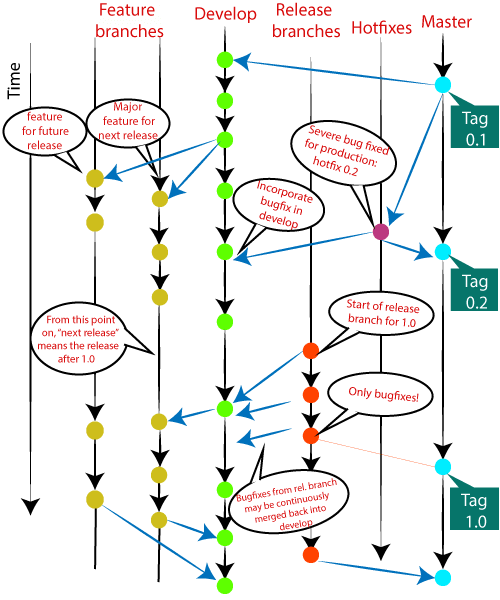
Git Flow / Git Branching Model

Git flow is the set of guidelines that developers can follow when using Git. We cannot say these guidelines as rules. These are not the rules; it is a standard for an ideal project. So that a developer would easily understand the things.

It is referred to as **Branching Model** by the developers and works as a central repository for a project. Developers work and push their work to different branches of the main repository.



There are different types of branches in a project. According to the standard branching strategy and release management, there can be following types of branches:

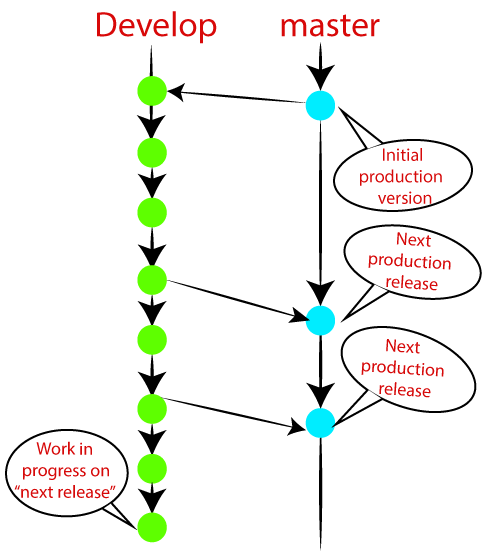
* **Master**
* **Develop**
* **Hotfixes**
* **Release branches**
* **Feature branches**

Every branch has its meaning and standard. Let's understand each branch and its usage.

The Main Branches

Two of the branching model's branches are considered as main branches of the project. These branches are as follows:

* **master**
* **develop**



Master Branch

The master branch is the main branch of the project that contains all the history of final changes. Every developer must be used to the master branch. The master branch contains the source code of HEAD that always reflects a final version of the project.

Your local repository has its master branch that always up to date with the master of a remote repository.

It is suggested not to mess with the master. If you edited the master branch of a group project, your changes would affect everyone else, and very quickly, there will be merge conflicts.

Develop Branch

It is parallel to the master branch. It is also considered as the main branch of the project. This branch contains the latest delivered development changes for the next release. It has the final source code for the release. It is also called as a "**integration branch**."

When the develop branch reaches a stable point and is ready to release, it should be merged with master and tagged with a release version.

Supportive Branches

The development model needs a variety of supporting branches for the parallel development, tracking of features, assist in quick fixing and release, and other problems. These branches have a limited lifetime and are removed after the uses.

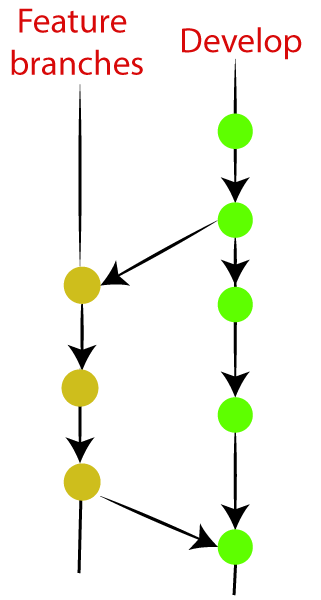
The different types of supportive branches, we may use are as follows:

* **Feature branches**
* **Release branches**
* **Hotfix branches**

Each of these branches is made for a specific purpose and have some merge targets. These branches are significant for a technical perspective.

Feature Branches

Feature branches can be considered as topic branches. It is used to develop a new feature for the next version of the project. The existence of this branch is limited; it is deleted after its feature has been merged with develop branch.



To learn how to create a Feature Branch [**Visit Here**](https://www.javatpoint.com/git-branch).

Release Branches

The release branch is created for the support of a new version release. Senior developers will create a release branch. The release branch will contain the predetermined amount of the feature branch. The release branch should be deployed to a staging server for testing.

Developers are allowed for minor bug fixing and preparing meta-data for a release on this branch. After all these tasks, it can be merged with the develop branch.

When all the targeted features are created, then it can be merged with the develop branch. Some usual standard of the release branch are as follows:

* Generally, senior developers will create a release branch.
* The release branch will contain the predetermined amount of the feature branch.
* The release branch should be deployed to a staging server for testing.
* Any bugs that need to be improved must be addressed at the release branch.
* The release branch must have to be merged back into developing as well as the master branch.
* After merging, the release branch with the develop branch must be tagged with a version number.

To create a release branch, visit [**Git Branching**](https://www.javatpoint.com/git-branch).

To tag branch after merging the release branch, Visit [**Git tag**](https://www.javatpoint.com/git-tags).

Hotfix Branches

Hotfix branches are similar to Release branches; both are created for a new production release.

The hotfix branches arise due to immediate action on the project. In case of a critical bug in a production version, a hotfix branch may branch off in your project. After fixing the bug, this branch can be merged with the master branch with a tag.

