# Lecture Notes: Branching Strategies

## 2.1 Introduction to Branching Strategy

Branching strategies are defined as a set of rules that describes the various guidelines associated with the branches. For example, proper naming convention should be used to identify the work that takes place in a particular branch. These strategies play a very important role in any software development process. Whenever working on a project, using the right branching strategy ensures that our codebase evolves in a logical and consistent manner. Branching strategy answers the following questions:

- When should a developer branch?
- From which branch they should branch off?
- Over which branch should the merge operation be performed?

However, it is very important that the right branching strategy should be adopted by a team. Adopting a right branching strategy enhances the quality of the product being developed and also facilitates faster development. Branching strategy also helps us understand how work flows among the different branches that are being used in the organisation.

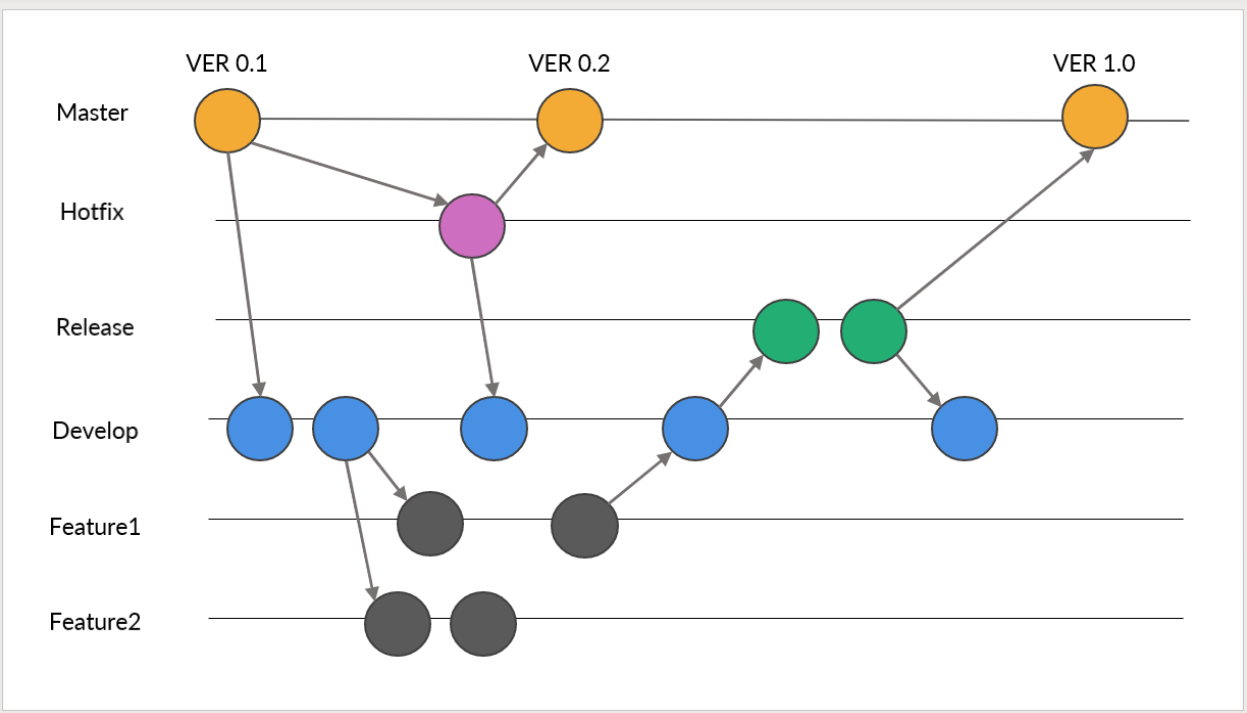
**Need for branching strategy**

Branching strategies ease the task of a developer, as they reduce a lot of complexities faced by the teams while managing large projects. Generally, when a team comprises a large number of developers, then, at that time, branching and merging becomes a very difficult operation. Merge conflicts and late-stage integrations result in a delay in the future work or releases. Therefore, using the right branching strategy helps organise the pipeline in a far more better and convenient way so that the teams are able to complete their work without compromising with quality for speed. However, it is important that the branching strategy chosen by a team should aim to achieve the following objectives:

- It should ensure parallel development.
- Developers should be able to work in isolated environments, so that there is no overlapping and no overwriting of work.
- It should evolve to accommodate the changes easily.
- It should always focus on optimising productivity.
- Proper deadlines must be defined for structured releases.

## 2.2 Understanding Gitflow

**Git flow**




Gitflow Workflow was first proposed and published by Vincent Driessen. This workflow is one of the widely used workflows in software development and implementation of DevOps practices. It is ideally suited for projects that have scheduled release and require continuous delivery. This workflow is also suitable for managing large projects that comprises many developers working together because of its robust branching nature.

This branching strategy employs separate branches for performing each task, such as feature development, release management, production management and bug fixing. Using separate branches for each task facilitates parallel development and ensures that, in case of failure of any branch, other branches still carry their work. Besides this, there are also guidelines regarding how and when the different branches are going to interact with each other. The description of various branches used in gitflow are as follows:

- **Master:** This is the main branch that contains the latest release version of the source code that is live in production. Master branch is highly stable and has an infinite lifetime.

- **Develop:** This is the second most important branch after the master branch. Usually, we do not merge the feature branches to the master branch because we do not want to disturb code that is already live in production. Therefore, the develop branch acts as an integration branch where the features that are planned for an upcoming release are integrated.

- **Feature:** Whenever any new feature needs to be developed, then feature branches are created. They branch off from the develop branch, and as soon as the work on a feature branch is completed, it again merges to the develop branch and is deleted.

- **Release:** Release branches are meant to perform release-oriented tasks, such as documentation, bug fixing and testing the feature. Whenever the code present in the develop branch is ready to be released, the release branch branches out. Once all the activities are performed, this branch is merged with the master branch, which is tagged with a version number, and also with the develop branch.

- **Hotfix:** These branches branch off the master branch. Whenever any bug is identified in the code present at the production stage, the hotfix branches are quickly forked out in order to fix the bug. Once the error is resolved, the hotfix branch merges back to both the master and the develop branches.