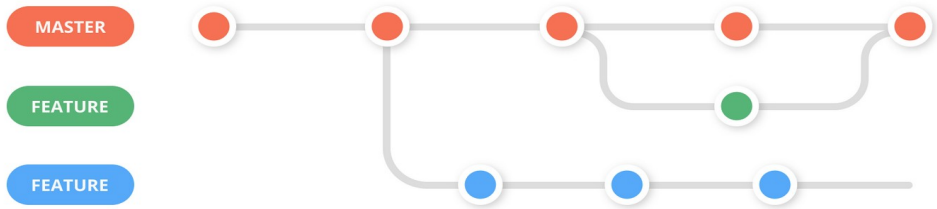
1. Basic Git Workflow

The most basic git workflow is the one where there is only one branch — the master branch. Developers commit directly into it and use it to deploy to the staging and production environment.

This workflow isn’t usually recommended unless you’re working on a side project and you’re looking to get started quickly.
Since there is only one branch, there really is no process over here. This makes it effortless to get started with Git. However, some cons you need to keep in mind when using this workflow are:
1.Collaborating on code will lead to multiple conflicts.
2.Chances of shipping buggy software to production is higher.
3.Maintaining clean code is harder.

2. Git Feature Branch Workflow

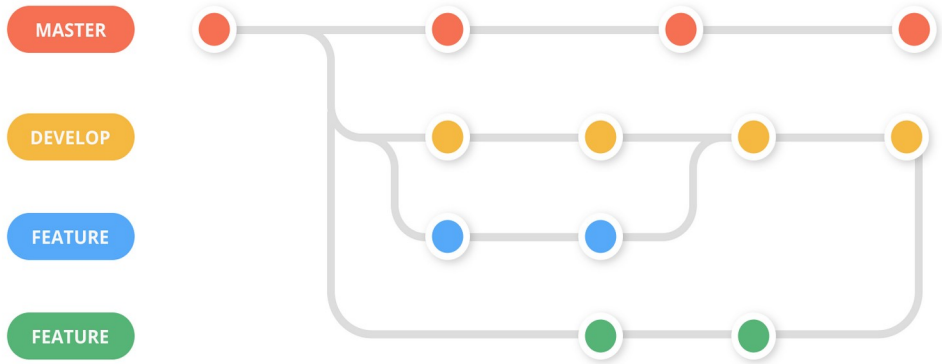
The Git Feature Branch workflow becomes a must have when you have more than one developer working on the same codebase.
Imagine you have one developer who is working on a new feature. And another developer working on a second feature. Now, if both the developers work from the same branch and add commits to them, it would make the codebase a huge mess with plenty of conflicts.



To avoid this, the two developers can create two separate branches from the master branch and work on their features individually. When they're done with their feature, they can then merge their respective branch to the master branch, and deploy without having to wait for the second feature to be completed.
The Pros of using this workflow is, the git feature branch workflow allows you to collaborate on code without having to worry about code conflicts.

3. Git Feature Workflow with Develop Branch

This workflow is one of the more popular workflows among developer teams. It's similar to the Git Feature Branch workflow with a develop branch that is added in parallel to the master branch.
In this workflow, the master branch always reflects a production-ready state. Whenever the team wants to deploy to production they deploy it from the master branch.
The develop branch reflects the state with the latest development changes for the next release. Developers create branches from the develop branch and work on new features. Once the feature is ready, it is tested, merged with develop branch, tested with the develop branch's code in case there was a prior merge, and then merged with master.



The advantage of this workflow is, it allows teams to consistently merge new features, test them in staging, and deploy to production. While maintaining code is easier, it can get a little tiresome for some teams since it can feel like going through a tedious process.