## **What Is GitFlow?**

[GitFlow](http://nvie.com/posts/a-successful-git-branching-model/) is a branching model for Git, created by Vincent Driessen. It has attracted a lot of attention because it is very well suited to collaboration and scaling the development team.

## **Key Benefits**

### **Parallel Development**

One of the great things about GitFlow is that it makes parallel development very easy, by isolating new development from finished work. New development (such as features and non-emergency bug fixes) is done in **feature branches**, and is only merged back into main body of code when the developer(s) is happy that the code is ready for release.

Although interruptions are a BadThing(tm), if you are asked to switch from one task to another, all you need to do is commit your changes and then create a new feature branch for your new task. When that task is done, just checkout your original feature branch and you can continue where you left off.

### **Collaboration**

Feature branches also make it easier for two or more developers to collaborate on the same feature, because each feature branch is a sandbox where the only changes are the changes necessary to get the new feature working. That makes it very easy to see and follow what each collaborator is doing.

### **Release Staging Area**

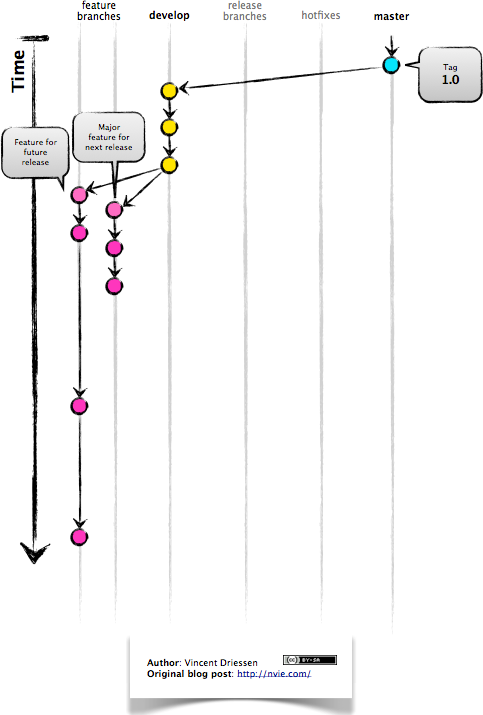
As new development is completed, it gets merged back into the **develop branch**, which is a staging area for all completed features that haven’t yet been released. So when the next release is branched off of **develop**, it will automatically contain all of the new stuff that has been finished.

### **Support For Emergency Fixes**

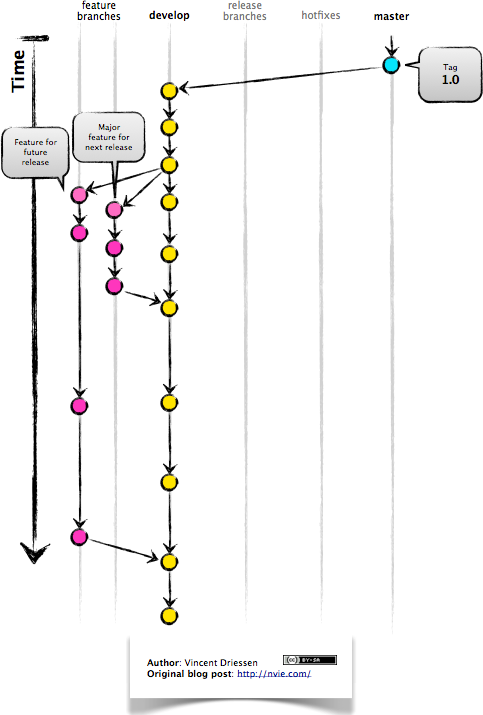
GitFlow supports **hotfix branches** - branches made from a tagged release. You can use these to make an emergency change, safe in the knowledge that the hotfix will only contain your emergency fix. There’s no risk that you’ll accidentally merge in new development at the same time.

## **How It Works**

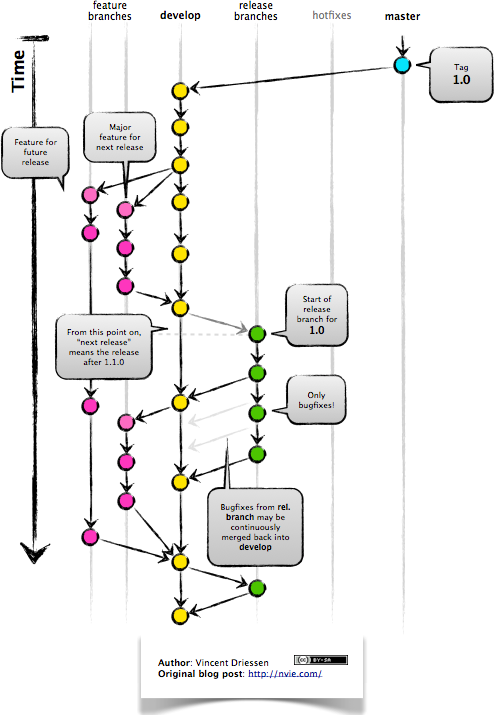
New development (new features, non-emergency bug fixes) are built in **feature branches**:



Feature branches are branched off of the **develop branch**, and finished features and fixes are merged back into the **develop branch** when they’re ready for release:

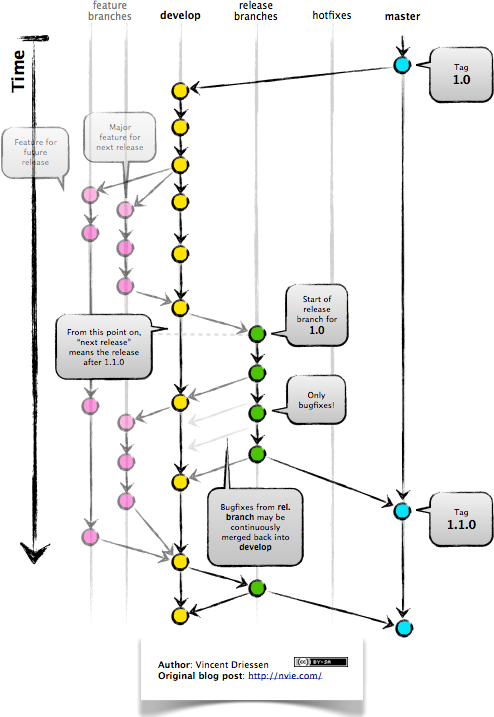


When it is time to make a release, a **release branch** is created off of **develop**:



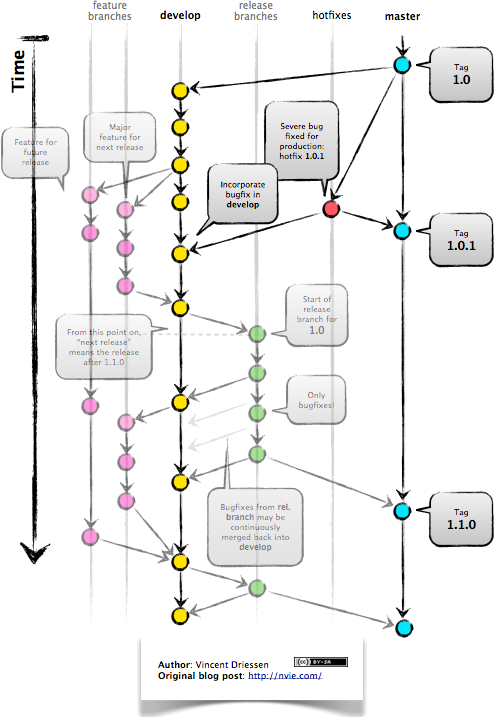
The code in the **release branch** is deployed onto a suitable test environment, tested, and any problems are fixed directly in the release branch. This **deploy -> test -> fix -> redeploy -> retest** cycle continues until you’re happy that the release is good enough to release to customers.

When the release is finished, the **release branch** is merged into **master** **and** into **develop** too, to make sure that any changes made in the **release branch** aren’t accidentally lost by new development.



The **master branch** tracks released code only. The only commits to **master** are merges from **release branches** and **hotfix branches**.

**Hotfix branches** are used to create emergency fixes:



They are branched directly from a tagged release in the **master branch**, and when finished are merged back into both **master** and **develop** to make sure that the hotfix isn’t accidentally lost when the next regular release occurs.