**Git Branching**

There a few different branching strategies when it comes to using Git. But the most popular seems to be one known as GitFlow. This strategy was first presented by Vincent Driessen in 2010, http://nvie.com/posts/a-successful-git-branching-model/. Since its introduction GitFlow has been adopted by more projects than just about any other strategy. Because of this you should get to know it.

Here are the basic ideas of GitFlow:

* There are two infinite branches (these names are common but not mandatory)
  + master
    - master branch always represents the code that is in production
  + develop
    - develop branch is similar to master but is ahead by current development
* In addition to the infinite branches you have any number of branches that fall into one of three or more types
  + hot fix (bugs found in prod)
    - hot fixes are branched off of master and merged back into master as well as develop when completed and (optionally) code reviewed.
  + feature (story)
    - features are branched off of develop and merged back into develop after code review
  + release
    - release is branched off develop when enough features are complete to comprise a release, or optionally at the end of a sprint
    - release is merged into master and back into develop when released
    - when release is created this is a code freeze for this particular release, develop then continues towards the next release
    - you may branch off release for bugs found in QA, these branches are merged back into release

In 2015 Adam Ruka proposed a better solution than GitFlow, <http://endoflineblog.com/gitflow-considered-harmful>. He makes some good points but I am not 100% sold that it’s better. And GitFlow is still so popular that I still think it is the one to know.

Here is Driessens original diagram.

