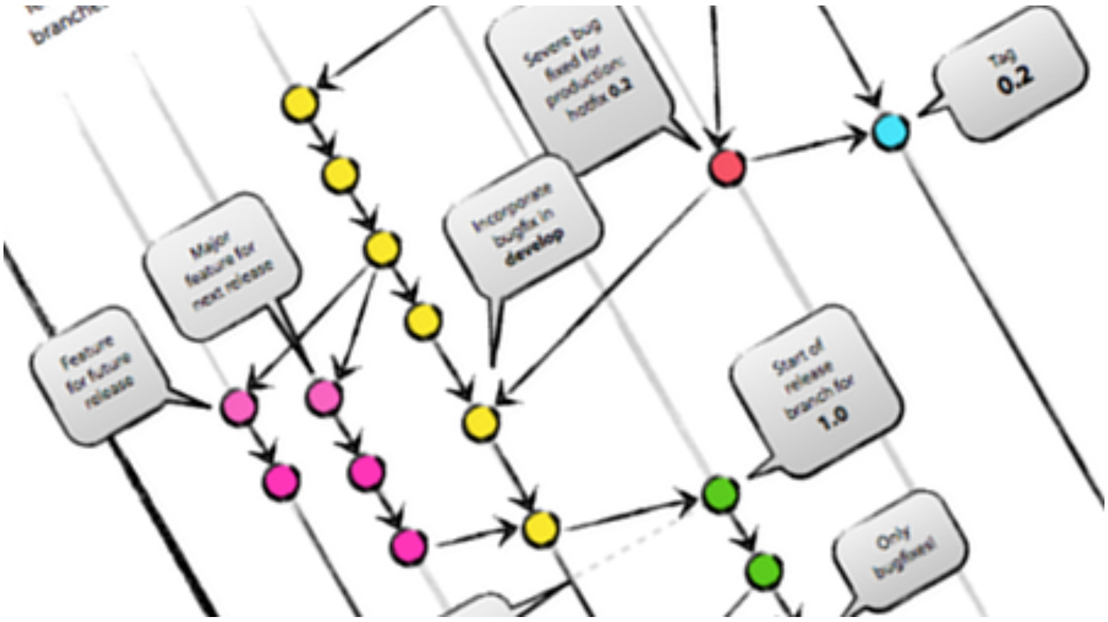
WHY AREN'T YOU USING GIT-FLOW?

2010-08-19

In January of this year, **@nvie** published **"A successful Git branching model"**, in which he explained how he keeps his Git repositories nice and tidy. In addition to that, he released **git-flow**; a bunch of Git extensions to make following this model extremely easy.

I'm astounded that some people never heard of it before, so in this article I'll try to tell you why it can make you happy and cheerful all day.


Severe bug fixed for production: hotfix 0.2
Tag 0.2
Incorporate bugfix in develop
Major feature for next release
Feature for future release
Start of release branch for 1.0
Only bugfixes!

After installing git-flow, you can start a new repository in the current directory or convert an existing one to the new branch structure:

```
$ git flow init
```

It will ask you a bunch of questions, but you probably want to accept the default values:

```
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]
How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
```

After you’ve answered the questions, git flow sets your default branch to `develop` (or whatever you named it) automatically, since that’s the one you’ll be working in.

Now, simply use Git like you’re used to, but only work on some small features in the `develop` branch. If you need to work on a bigger feature, just create a feature branch based on `develop`. Let’s say you want to add a login page:

```
$ git flow feature start login
```

This will create a new branch called `feature/login`, based on our `develop` branch and switches to it. Commit away and after you finish working on the login page, simply finish it:

```
$ git flow feature finish login
```

It’ll merge `feature/login` back to `develop` and delete the feature branch.

When you’re feature complete, simply start a release branch — again, based on `develop` — to bump the version number and fix the last bugs before releasing:

```
$ git flow release start v0.1.0
```

When you finish a release branch, it’ll merge your changes to `master` *and* back to `develop`, so you don’t have to worry about your `master` being ahead of `develop`.

The last thing that makes git-flow awesome is its ability to handle hotfixes. You start and finish a hotfix branch like anything else, but it’s based on `master` so you can quickly fix it when something’s broken production and merge it back to `master` and `develop` using `finish`.

Awesome, right? Now, what are you waiting for?