Introduction to GitFlow

Nicolas Barrier

UMR MARBEC

nicolas.barrier@ird.fr

March 25, 2019

Introduction

There are several ways to use GIT (we talk about **workflows**).

- *Centralized workflow*: one main branch, everyone commit in the same place.
- *Feature Branch Workflow*: developments are made in dedicated branches (feature branches), which are regularly merged to the master one.
- **Gitflow Workflow**: Strict branching model designed around the project release.

Source: `https://www.atlassian.com/git/tutorials/comparing-workflows`) for more details.

GitFlow branches

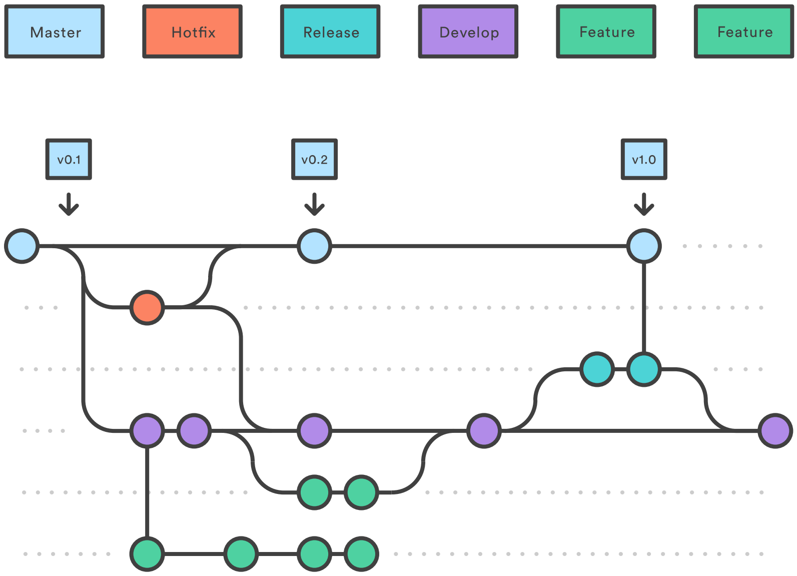
GitFlow workflow contains two main branches:

- *master:* official release history. Branch which is shared to the world!
- *develop:* integration branch for features

It also contains additional temporal branches:

- *feature*: feature branches (one for each new feature to add to the code)
- *release*: branch created when enough features have been added (new version of the code)
- *hotfix*: branch for maintenance and bug correction of the production release

In summary...



How it works in practice

Install the GitFlow extension:

```
sudo apt-get install git-flow
```

Now, from your directory, initialize the GitFlow workflow:

```
git flow init # create the develop branch
git push origin develop # push the develop branch to the
    repository
```

Creating new features

```
# recovers the latest develop branch
git pull origin develop

# create feature branch from develop and switch to it
git flow feature start my-feature

# put the feature branch to the repo (not necessary)
git flow feature publish my-feature

# edit/add files: new functionalities

# commit changes to the my-feature branch
git commit

# merge my-feature branch with develop and remove it
# switch back to develop
git flow feature finish

# push updated develop branch
git push origin develop

# if conflict (someone has already pushed in develop)
git pull origin develop
git push origin develop
```

Creating a new release

```
# recovers the latest develop branch
git pull origin develop

# create a new release branch
git flow release start my-release

# put the release branch to the repo (not necessary)
git flow release publish my-release

# edit/add files: bug fixes + documentation

# commit changes to the my-feature branch
git commit

# finish the release
git flow release finish

# push updated develop/master and tags
git push origin --tags
git push origin develop
git push origin master
```

Creating hotfixes

```
# recovers the latest develop/master branches
git pull origin develop

# create a new hotfix branch from master
# name should be a version name (1.3.2 for instance)
git flow hotfix start hotfix-tag

# edit/add files: bug corrections

# commit changes to the hotfix-tag branch
git commit

# finish the release
# merge hotfix branch to master/develop
git flow hotfix finish

# push updated develop/master and tags
git push origin --tags
git push origin develop
git push origin master
```

References

- https://nvie.com/posts/a-successful-git-branching-model/
- https://www.atlassian.com/git/tutorials/comparing-workflows
- https://danielkummer.github.io/git-flow-cheatsheet/
- https://gist.github.com/JamesMGreene/cdd0ac49f90c987e45ac
- https://blog.xebia.fr/2018/03/28/gitflow-est-il-le-workflow-dont-jai-besoin/