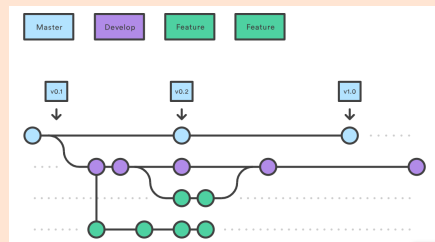


### Feature branches

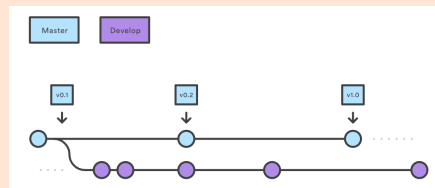


- Each new feature should reside in its own branch
- Pushed to the central repository for collaboration
- When a feature is complete, it gets merged back into develop (Pull request)
- Features should never interact directly with master

### Creating a feature branch

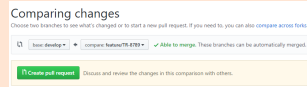
```
git checkout develop
git checkout -b feature/TR-5076
```

### Base Git Flow structure



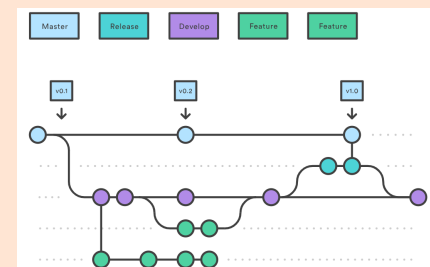
Instead of a single master branch, this workflow uses two branches to record the history of the project. The master branch stores the official release history, and the develop branch serves as an integration branch for features. It's also convenient to tag all commits in the master branch with a version number.

### Creating a pull request



- Create a pull request into **develop** as a base.
- ⚡ **Never use master as your base, except for when working on a hotfix branch**

### Release branches

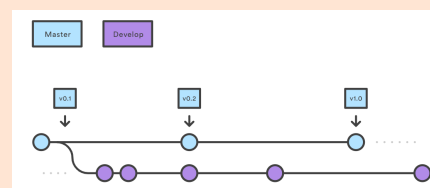


- Fork a release branch off of develop
- No new features can be added after this point
- Only bug fixes, documentation generation, and other release-oriented tasks
- Once it's ready to ship, the release branch gets merged into master and tagged with a version number

### Creating a release branch

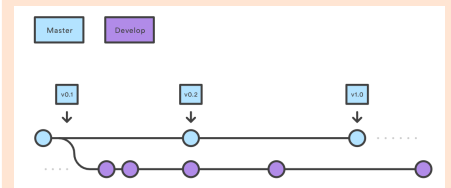
```
git checkout develop
git checkout -b release/3.372
```

### Basic Git Flow structure



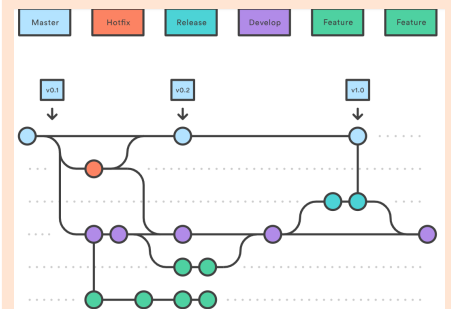
Instead of a single master branch, this workflow uses two branches to record the history of the project. The master branch stores the official release history, and the develop branch serves as an integration branch for features. It's also convenient to tag all commits in the master branch with a version number.

### Base Git Flow structure



Instead of a single master branch, this workflow uses two branches to record the history of the project. The master branch stores the official release history, and the develop branch serves as an integration branch for features. It's also convenient to tag all commits in the master branch with a version number.

### Hotfix branches



- Maintenance or "hotfix" branches are used to quickly patch production releases
- This is the only branch that should fork directly off of master
- As soon as the fix is complete, it should be merged into both master and develop

### Branching conventions

#### Feature branches

- Feature branches are named after the issues they belong to (eg. TR-5076)
- Feature branches can be used for multiple small issues by combining names (eg. TR-5076\_7776\_3921)
- Feature branches should live as short as possible
- Feature branches end their lives when the pull request is approved

#### Release branches

- Release branches are named after the next release (eg. 3.372)
- Release branches may only be merged to *master* by the Release Manager.

#### Hotfix branches

- Hotfixes should be named after the release they are hotfixed on, by adding a number (eg. 3.372.1)
- Hotfixes are created from master



By **CornelvdH**

[cheatography.com/cornelvdh/](https://cheatography.com/cornelvdh/)

Not published yet.

Last updated 8th January, 2019.

Page 2 of 2.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>