Never Commit to Master

An Introduction to Git Flow

By Oliver Davies / @opdavies

Me

- Oliver Davies
- Developer and Systems Administrator
- @opdavies (Twitter, D.O., IRC)
- http://dgo.to/@opdavies
- PHPer and Drupalista since 2007 (full time since 2010)
- Git user since 2010
- Git Flow user since 2013

precedent.

Git Flow is:

"A collection of Git extensions to provide high-level repository operations for Vincent Driessen's branching model."

From https://github.com/nvie/gitflow

The Branching Model

Branches

- `master`: production code
- `develop`: development code
- `feature/`: a specific task or ticket (multiple)
- `release/`: temporary release branch for testing (single)
- `hotfix/`: temporary branch for emergency fixes
- `suport/`: experimental

Why use Git Flow?

- Separation of production and development code
- Flexibility
- Better code quality
- Encourages collaboration
- Encourages peer code reviews

My rules of Git Flow

1. Never, ever commit code directly to `master`
2. Only commit stable code to `develop`
3. Try not to commit directly to `develop`
4. One feature branch per user story/bug
5. Commit early, commit often, push often

How do I use it?

- CLI
  - `~ $ brew install git-flow`
  - `~ $ apt-get install git-flow`
  - `~ $ yum install gitflow`
- SourceTree (free, cross-platform GUI)

Need Help?

```
~ $ git flow help
# Shows the standard help menu

~ $ git flow {subcommand} help
# Shows the help menu for a specific subcommand
```

Initialise Git Flow

```
~ $ git flow init
```

Create your default branches

```
~ $ git flow init

No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]
```

Configure branch prefixes

```
~ $ git flow init

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? []
```

Tip: Automatically accept the default branch names

```
~ $ git flow init -d
# Accepts the default branch names.
```

Features

```
$ git flow feature
```

- `list`: lists all features
- `checkout`: checks out an existing feature
- `start`: starts a new feature
- `finish`: finishes a feature
- `publish`: pushes a feature into a remote repo
- `pull`: pulls a feature from a remote repo

Start a new feature branch

```
~ $ git flow feature start {name}
```

```
~ $ git flow feature start foo

Switched to a new branch 'feature/foo'

Summary of actions:
- A new branch 'feature/foo' was created, based on 'develop'
- You are now on branch 'feature/foo'

Now, start committing on your feature. When done, use:

git flow feature finish foo
```

Add and commit changes

```
~ $ drush dl admin_menu
~ $ git add sites/all/modules/contrib/admin_menu
~ $ git commit -m "Added admin_menu"
```

Recommended: Rebase Your Feature

Ensure that your feature is up-to-date

```
~ $ git flow feature rebase

Will try to rebase 'foo'...
First, rewinding head to replay your work on top of it...
Applying: Added admin_menu
```

Finish a feature

```
~ $ git flow feature finish {name}
```

```
~ $ git flow feature finish foo

Switched to branch 'develop'
Updating 5c04d5a..6487134
Fast-forward
...
31 files changed, 5051 insertions(+)
Deleted branch feature/foo (was 6487134).

Summary of actions:
- The feature branch 'feature/foo' was merged into 'develop'
- Feature branch 'feature/foo' has been removed
- You are now on branch 'develop'
```

And repeat...

Releases

```
$ git flow release
```

- `list`: lists existing releases
- `start`: starts a new release
- `finish`: finishes a release

Start a new release

```
~ $ git flow release start {version}
```

```
~ $ git flow release start 2014-03-02.0

Switched to a new branch 'release/2014-03-02.0'

Summary of actions:
- A new branch 'release/2014-03-02.0' was created, based on 'develop'
- You are now on branch 'release/2014-03-02.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

     git flow release finish '2014-03-02.0'
```

Finish a release

```
~ $ git flow release finish {version}
```

```
~ $ git flow release finish 2014-03-02.0

  ...

  Deleted branch release/2014-03-02.0 (was f2aee7d).

  Summary of actions:
  - Latest objects have been fetched from 'origin'
  - Release branch has been merged into 'master'
  - The release was tagged '2014-03-02.0'
  - Release branch has been back-merged into 'develop'
  - Release branch 'release/2014-03-02.0' has been deleted
```

Pushing changes remotely

```
~ $ git push --all
# Push the changes to the remote branches.

~ $ git push --tags
# Push the tags.
```

Tip: Finish a release in one command

```
~ $ git flow release finish -pm {message} {version}
# Specify a commit message and automatically push the changes.
```

```
~ $ git flow release finish -pm 2014-03-02.0 2014-03-02.0
```

finish-sprint.sh

~ $ ./finish-sprint.sh 2014-03-02.1

```
#!/bin/bash

DRUPAL_DIR="/path/to/drupal/docroot"
TAG=$1

if [ -z $TAG ]; then
  # A tag must be specified.
  echo 'You must specify a tag.'
fi

# Go into the Drupal directory
cd $DRUPAL_DIR

# Start a new Git Flow release.
git flow release start $TAG -F

# Flush the cache.
drush cc all

# Export the database
drush sql-dump --gzip --result-file=../db/$TAG.sql
```

Hotfixes

```
$ git flow hotfix
```

- `list`: list all hotfixes
- `start`: start a hotfix
- `finish`: finish a hotfix

Create a new hotfix

```
~ $ git flow hotfix start {version}
```

```
~ $ git flow hotfix start 2014-03-02.2

Switched to a new branch 'hotfix/2014-03-02.2'

Summary of actions:
- A new branch 'hotfix/2014-03-02.2' was created, based on 'master'
- You are now on branch 'hotfix/2014-03-02.2'

Follow-up actions:
- Bump the version number now!
- Start committing your hot fixes
- When done, run:

     git flow hotfix finish '2014-03-02.2'
```

Commit your fixes

```
~ $ git ci -am 'Updated .htaccess'
[hotfix/2014-03-02.2 6d04738] Updated .htaccess
 1 file changed, 4 insertions(+), 4 deletions(-)
```

Finish the hotfix

```
~ $ git flow hotfix finish 2014-03-02.2
Switched to branch 'master'
Merge made by the 'recursive' strategy.
 .htaccess | 8 ++++----
 1 file changed, 4 insertions(+), 4 deletions(-)
Switched to branch 'develop'
Merge made by the 'recursive' strategy.
 .htaccess | 8 ++++----
 1 file changed, 4 insertions(+), 4 deletions(-)
Deleted branch hotfix/2014-03-02.2 (was 6d04738).

Summary of actions:
- Latest objects have been fetched from 'origin'
- Hotfix branch has been merged into 'master'
- The hotfix was tagged '2014-03-02.2'
- Hotfix branch has been back-merged into 'develop'
- Hotfix branch 'hotfix/2014-03-02.2' has been deleted
```

Resources

- http://nvie.com/posts/a-successful-git-branching-model/
- http://jeffkreeftmeijer.com/2010/why-arent-you-using-git-flow/
- http://danielkummer.github.io/git-flow-cheatsheet/
- https://github.com/nvie/gitflow
- https://github.com/nvie/gitflow/wiki

Demo

Questions?

Thanks

Feedback appreciated!

- Slides: http://www.oliverdavies.co.uk/git-flow
- Session evaluation: http://2014.drupalcamplondon.co.uk/node/add/session-evaluation?nid=86