



Git good

Git tips and workflow

Viktor Holmgren

February 27, 2019

LiThe kod

Table of contents

Match that flow

Master and develop

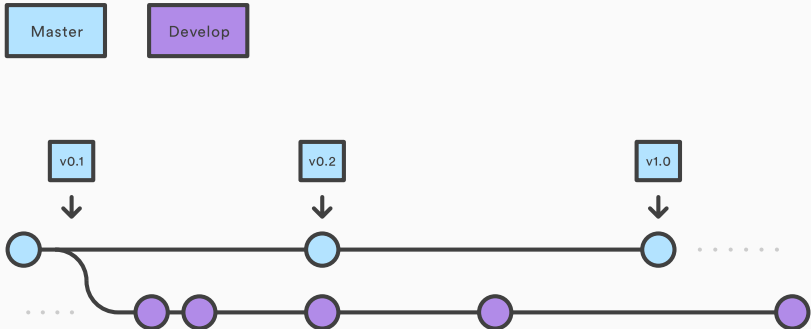
Git Flow workflow typically recommends *two main branches*;

master: official releases

develop: integration branch for features

```
git branch develop
```

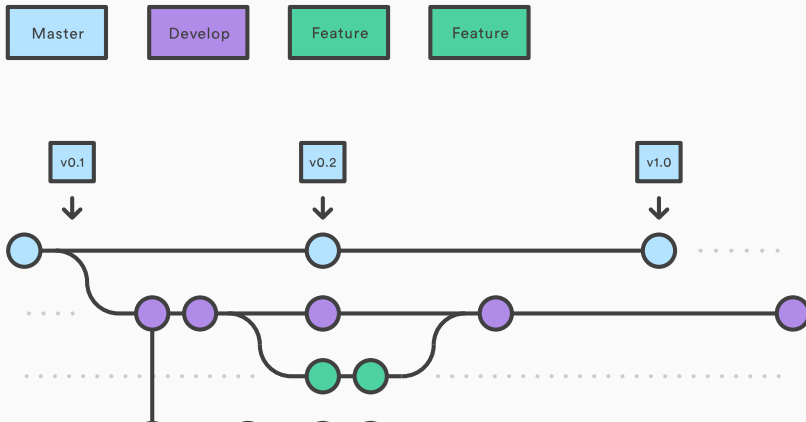
```
git push -u origin develop
```



Use them branches yo

Using Topic branches (feature/bugfix) is an extremely good practice!

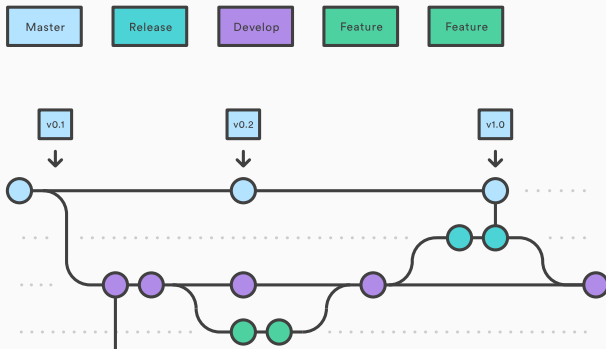
- Create a new branch from develop for each new *feature*
- Develop your feature
- Interactive rebase if needed to clean up code
- Merge into develop using CI



Lets ship it!

- When develop has enough features, or release is approaching
- Fork new branch from develop
- Continue with bugfixes, documentation but **no features**
- When ready merge *release* into *master* and tag, merge *master* into *develop*

Good: Work on release and develop in parallel, easy to see what release is in process

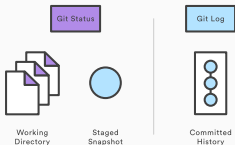


Making it look nice af

Patch it up baby

Git stores changes in three separate states:

- Working directory: Your files locally, currently.
- Staged/Index: Changes to be committed.
- Committed: Changes stored in the history.



Use *git status* and git will tell you what to do!

- *git add* add files to index, *git add -p* adds *parts* of a file
- *git reset* removes files from index, *git reset -p* removes *parts* of a file
- *git reset --soft* removes commits and moves changes to index
- *git reset --hard* removes commits and discards changes.

Stashing them goodies

Sometimes you need to switch to work on something else, or store an attempt. Either use a *Work in progress* branch or stash your changes.

- WIP: `git checkout -b WIP-<something> && git add . && git commit -m 'WIP'`
- Stash: `git stash` then later `git stash pop`

What the rebase?

Rebasing is an alternative to *merging*.

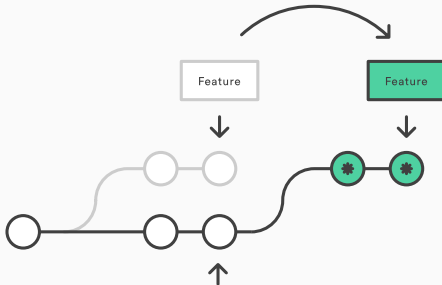
Will move commits from a given branch to the current branch.

```
git rebase <branch>
```

- Results in a linear history
- Will rewrite history (different history results in different commits)

You shallt not rebase any shared changes

With great power comes great responsibility



What kind of sorcery is this?

Select and transform commits arbitrarily

```
git rebase -i <branch>
```

- Remove or add commits at any point
- Change order of commits
- Squash commits together
- Edit commit messages at any point

What kind of sorcery is this?

```
1 pick 8f07e8d Update to use stable clippy
19 pick 73df254 Add ggez dependencies to travis conf
1 pick 88555e2 Add sdl2 deps
2 pick b02247f Start work on layout system
3 pick 9a22697 Implement basic renderable trait
4 pick 05e50a6 Require renderable to be implemented for all components
5 pick ce053c9 Reformat accoring to rustfmt
6 pick d8c230a Add required layout method on views to be able to render them
7 pick c91c1db Make the UI render the current active view
8 pick 6596f68 Attempt to fix travis conf by borrowing install step from ggez
9 pick df92643 Supply ggez context downward instead to avoid having to send DrawParams
10 pick 711b6f7 Make all components renderable
11
12 # Rebase 5f97edf..711b6f7 onto 5f97edf (29 commands)
13 #
14 # Commands:
15 # p, pick <commit> = use commit
16 # r, reword <commit> = use commit, but edit the commit message
17 # e, edit <commit> = use commit, but stop for amending
18 # s, squash <commit> = use commit, but meld into previous commit
19 # f, fixup <commit> = like "squash", but discard this commit's log message
20 # x, exec <command> = run command (the rest of the line) using shell
21 # b, break = stop here (continue rebase later with 'git rebase --continue')
22 # d, drop <commit> = remove commit
```

But why?

Please use before you feature/bugfix branch to remote.

- Removing ulgy: fix <insert minor problem> commits by squash
- Undoing part of the work
- Splitting one large commit into smaller parts
- Combining smaller relevant parts
- Fix spelling mistakes in commit messages
- Correcing styleguide or adding comments in the same commit
- ...

What the hell is happening?!

Logging like a bawse

git log --decorate --oneline --graph

```
Update dependencies (#132)
commit 823a70d7273b51bd1eb42321a3635b776aa49722
Author: Viktor Holmgren <viktor.holmgren@gmail.com>
Date:   Mon Aug 20 22:41:31 2018 +0200

    Remove codecov as its dependencies breaks CI build (#134)

    * Remove codecov as its dependencies breaks CI build

    * Fix option in rustfmt

    * Reformat code according to updated rustfmt

commit 7f190746c7640ab978d62a6fd5c68a58b78c503d
Author: Viktor Holmgren <viktor.holmgren@gmail.com>
Date:   Mon Jul 30 22:50:02 2018 +0200

    Improve galaxy map performance by only iterating over visible systems (#131)

commit 12cc0b7a88ffa12957489cc0facc479fb76a5bf6
Author: Viktor Holmgren <viktor.holmgren@gmail.com>
Date:   Mon Jul 30 22:38:47 2018 +0200

    Minor refactors (#130)
```

Bad log

```
2e20917 Merge pull request #38 from holngr/feature/planet-gen
* 0cf6712 Clean up
* 0922cb3 Improve star generation
* 6297600 Predict planet type using tree model
* 25a0829 Use builders for generation
* c3f6c59 Move system generation complexity to generators instead of new
* c573e57 Move hash function to general module
* d7838ab Separate astronomicals to different files
* 6d8af9c WIP

2c09b82 Merge pull request #37 from holngr/feature/sensible-config
* fb5ec53 Reduce default cluster size, fixes #32

04b2e2a Merge pull request #36 from holngr/bugfix/fallback-names
* 5018d91 Fallback to Unnamed when no name found, fixes #30

6949aa6 Merge pull request #27 from holngr/feature/static-res
* 8f64dbb Replace ResourceHandler with lazy_static

9308d68 Merge pull request #26 from holngr/feature/system-names
* 489089a Add name generation to systems
```

Good log

Show me your secrets

Show the changes made in a given commit:

```
git diff <commit>^!
```

Show the staged changes (what will be committed)

```
git diff --staged
```


Blame it on the new guy

Git blame, aka 'Vem var det som kasta!'

Show the changes made in a given commit:

```
git blame <file>
```

```
notingim@corbett ~ % git blame x.py
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  1) #!/usr/bin/env python
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  2) # Copyright 2016 The Rust P
roject Developers. See the COPYRIGHT
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  3) # file at the top-level dir
ectory of this distribution and at
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  4) # http://rust-lang.org/COPY
RIGHT.
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  5) #
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  6) # Licensed under the Apache
License, Version 2.0 <LICENSE-APACHE or
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  7) # http://www.apache.org/lic
enses/LICENSE-2.0> or the MIT license
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  8) # <LICENSE-MIT or http://op
ensource.org/licenses/MIT>, at your
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700  9) # option. This file may not
be copied, modified, or distributed
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700 10) # except according to those
terms.
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700 11)
04e4d426a16 (Titus Barik        2017-04-30 16:10:31 -0400 12) # This file is only a "syml
ink" to bootstrap.py, all logic should go there.
11adac350b6 (Vadim Petrochenkov 2017-03-03 05:27:07 +0300 13)
a270b8014cb (Alex Crichton      2016-10-21 13:18:09 -0700 14) import os
11adac350b6 (Vadim Petrochenkov 2017-03-03 05:27:07 +0300 15) import sys
11adac350b6 (Vadim Petrochenkov 2017-03-03 05:27:07 +0300 16) rust_dir = os.path.dirname(
os.path.abspath(__file__))
```

Welp that was stupid

Do it right, right a way

- **Use an ignorefile!**, you probably want to set it up before anything. Gitignore.io is really good.
- A global ignorefile can be good also if you work in similar languages or types of projects.

Simply create an ignore file at `/.gitignore_global`

Then run `gitconfig --globalcore.excludesfile /.gitignore_global`

Bisecting FTW

Finding that pesky bug

Git bisect is a general command to binary search for **any** property, but most commonly a bug.

```
> git bisect start
```

```
> git bisect bad
```

```
> git bisect good e7f9b07
```

Bisecting: 13 revisions left to test
after this (roughly 4 steps)

[ea7de...] Implement event forwarding

Then run *git bisect good* if that version works, or *git bisect bad* if it does not work until you are left with the commit you were searching for.








Teamwork makes the dream work

Try to make sense, okay

Writing a good commit message is art. But here are some guidelines:

- Limit the subject line to 50 characters
- Capitalize the subject line
- Do not end the subject line with a period
- Use the imperative mood in the subject line
 - 'Refactor A by moving Y' not
 - 'Refactored A by moving Y'
 - I.e like a command!

<https://chris.beams.io/posts/git-commit/>

| | | |
|---|------------------------------------|---|
|  | Creekyew 04/02/18 8:45 AM | Great googly moogly it's all gone to shit |
|  | bcariborg 04/02/18 8:44 AM | recursive insertion sort working, hell fucking yeah |
|  | nmonemato 04/02/18 8:06 AM | formatting shit |
|  | russellschmidt 04/02/18 7:43 AM | add hello world of shit soundboard |
|  | TRBaldim 04/02/18 7:20 AM | Fixing project shit |
|  | TRBaldim 04/02/18 7:19 AM | Fixing target shit |
|  | doly mood 04/02/18 7:14 AM | responsive and safari shit |

Code reviews, code review, code reviews

Use code reviews for everything, and I mean **everything!**

We used it to great effect in writing reports!

For each topic branch, use some type of code review.

You will:

- Spread knowledge about what is being worked on, and changed (knowledge sharing)
- Catch bugs and faults early!

The screenshot displays a GitHub pull request interface. At the top, a comment from JesperWestell dated May 18, 2017, states: "Främst bindestreck som tagits bort ur texten men kolla gärna igenom om det är några ändringar ni inte håller med om." Below this, a list of commits is shown: "JesperWestell added some commits on May 18, 2017", including "Fix issues adressed by opponents", "Remove indentations after figures", and "Add vertical lines between answers in results". A self-assignment by JesperWestell is also noted. A green checkmark indicates that "yousiftouma approved these changes on May 18, 2017", with a "View changes" link. Below the approval, a comment from yousiftouma reads: "Ett par förslag du kan se över, men inget speciellt att anmärka på, bra!". The main content area shows a diff for the file "thesis/appendix_jesper.tex", marked as "Outdated". The diff includes line numbers 58 and 59, with changes to author and title fields. A comment from yousiftouma dated May 18, 2017, is visible at the bottom, asking for clarification on the use of "studerats" or "studie har gjorts". A "Reply..." input field and a "Resolve conversation" button are also present.

Questions and demo?
