

Git Best Practice

Git 规范与最佳实践

Michael He

Agenda

- 简介
- 基础命令
- 最佳实践
- 分支管理
- 深入 .git 目录
- 推荐工具

关于技术分享

- 通过分享可以更好的自我审视和学习。（费曼学习法之一）
- 希望大家能够以一种抛转引玉的心态，通过分享去获取新的知识。
- 让跨团队可以在另一个主题下进行沟通和学习，促进跨团队交流。我们不能在一个项目里成为战友，那么可以在一次技术分享会上成为朋友。
- 希望各位，不吝赐教，今后能够踊跃分享技术主题。
- 这就是：字节范

Git 简介

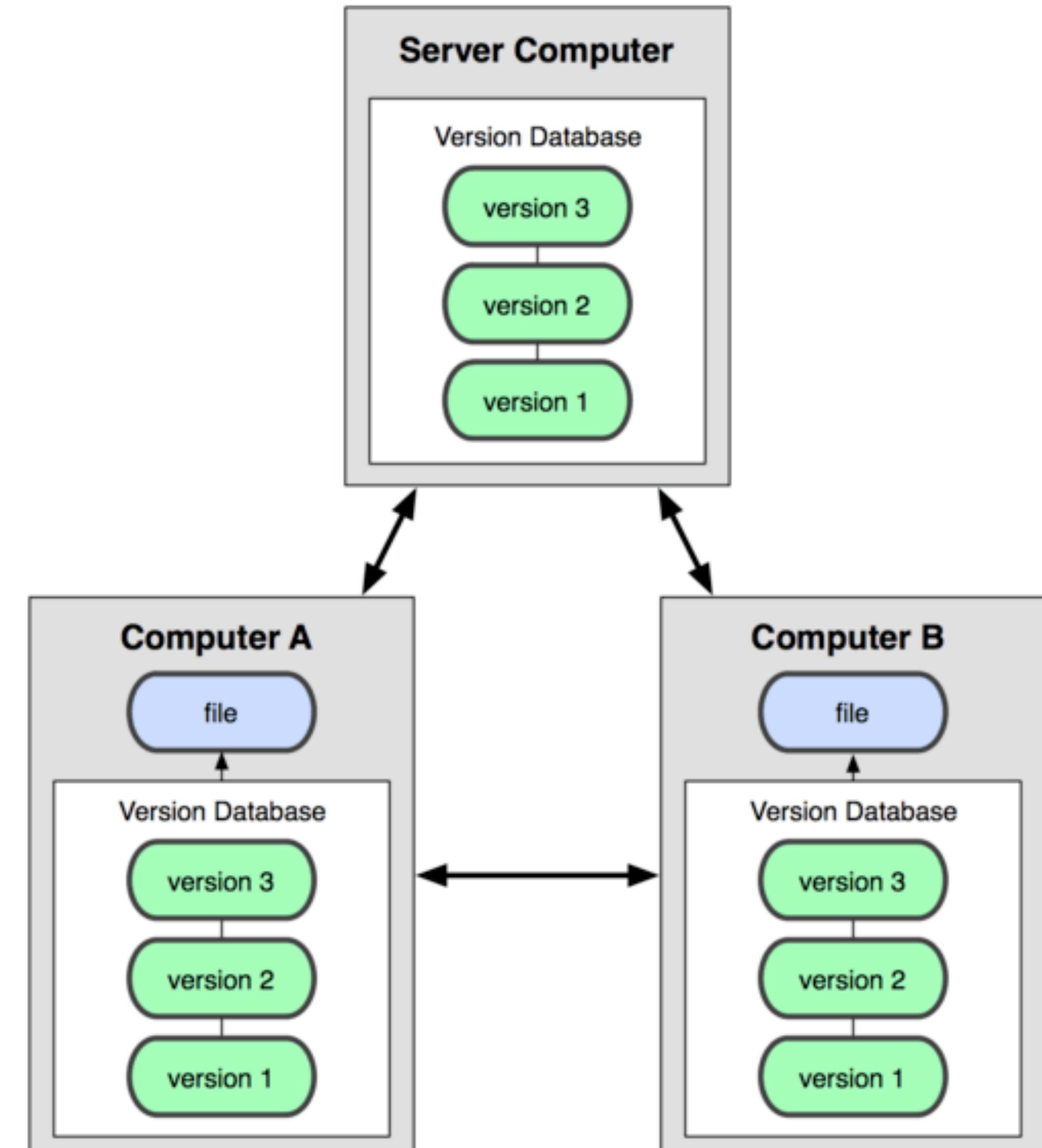
Git /git/ 是由 Linus Torvalds 也就是linux作者开发的。其实Contributer排名第一的是一个日本人叫做濱野純 (From Google) 。

Git 源于英语俚语，意思是 The stupid content tracker, 愚蠢的内容追踪器。

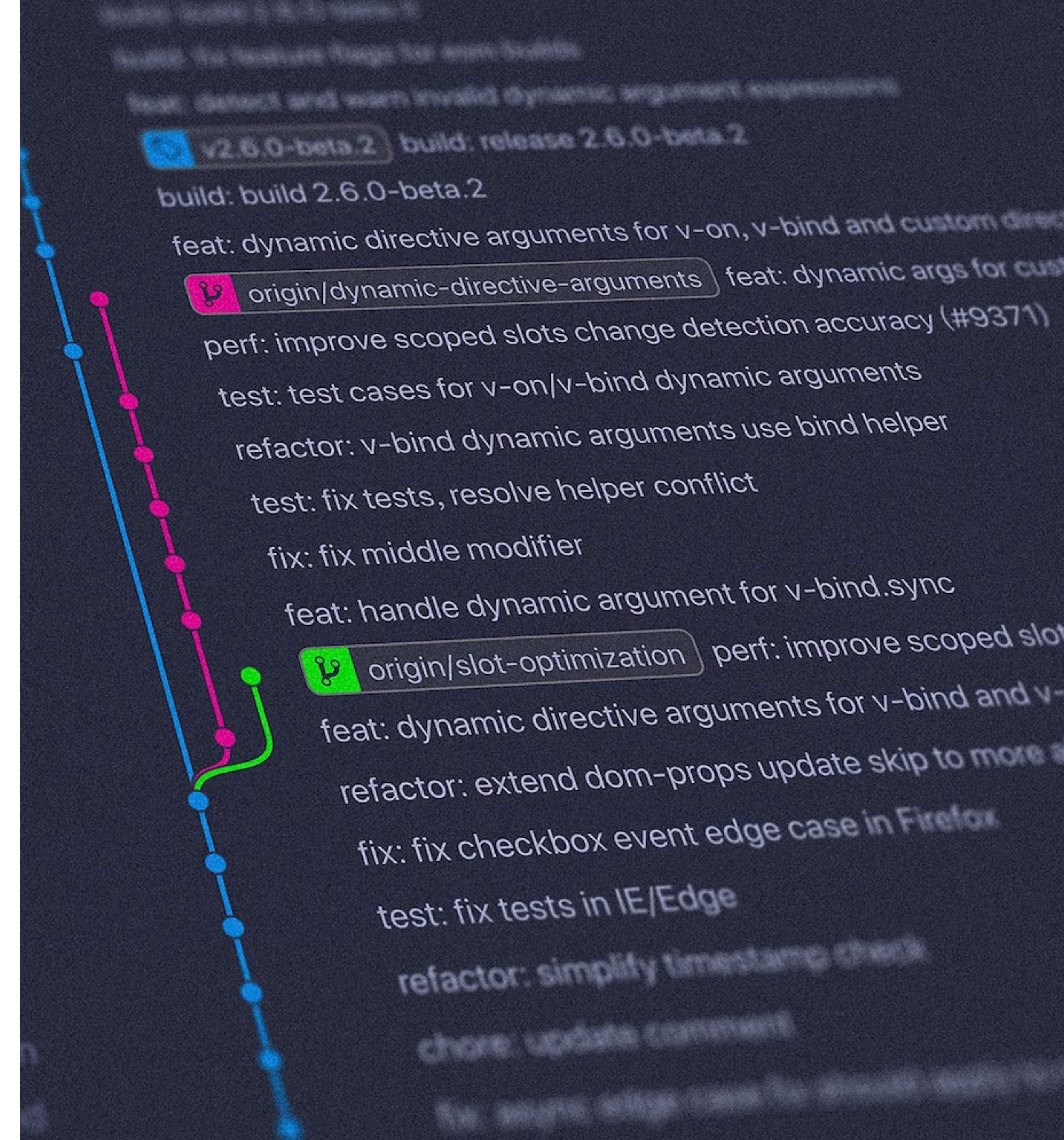
最开始被开发的时候主要是用于 Linux 内核开发的版本控制工具。

他跟以前的CVS, SVN最大的区别就是，他是分布式的管理代码库。

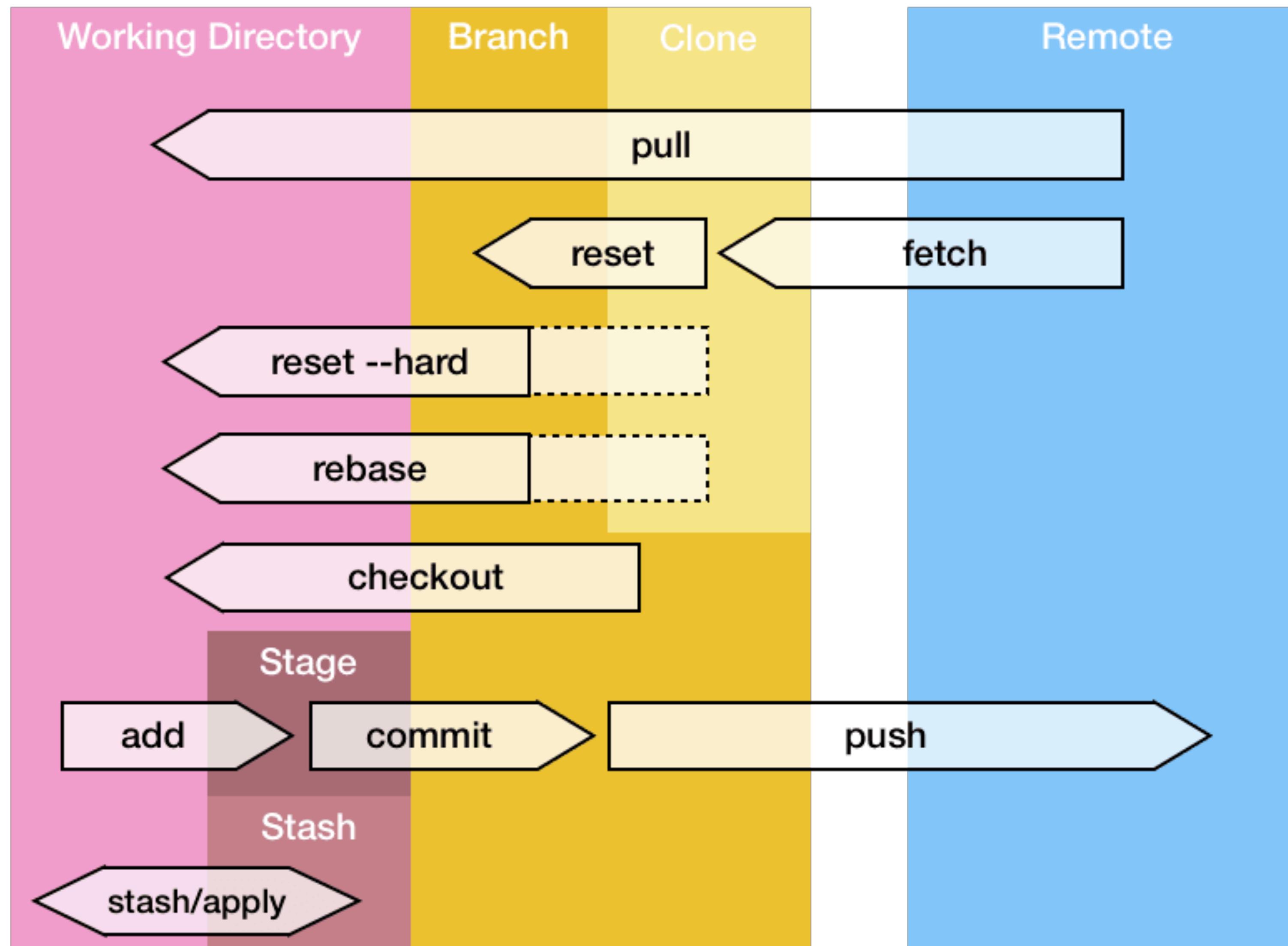
每个commit都可以在本地进行，之后同步服务器，极大增进了开发体验。



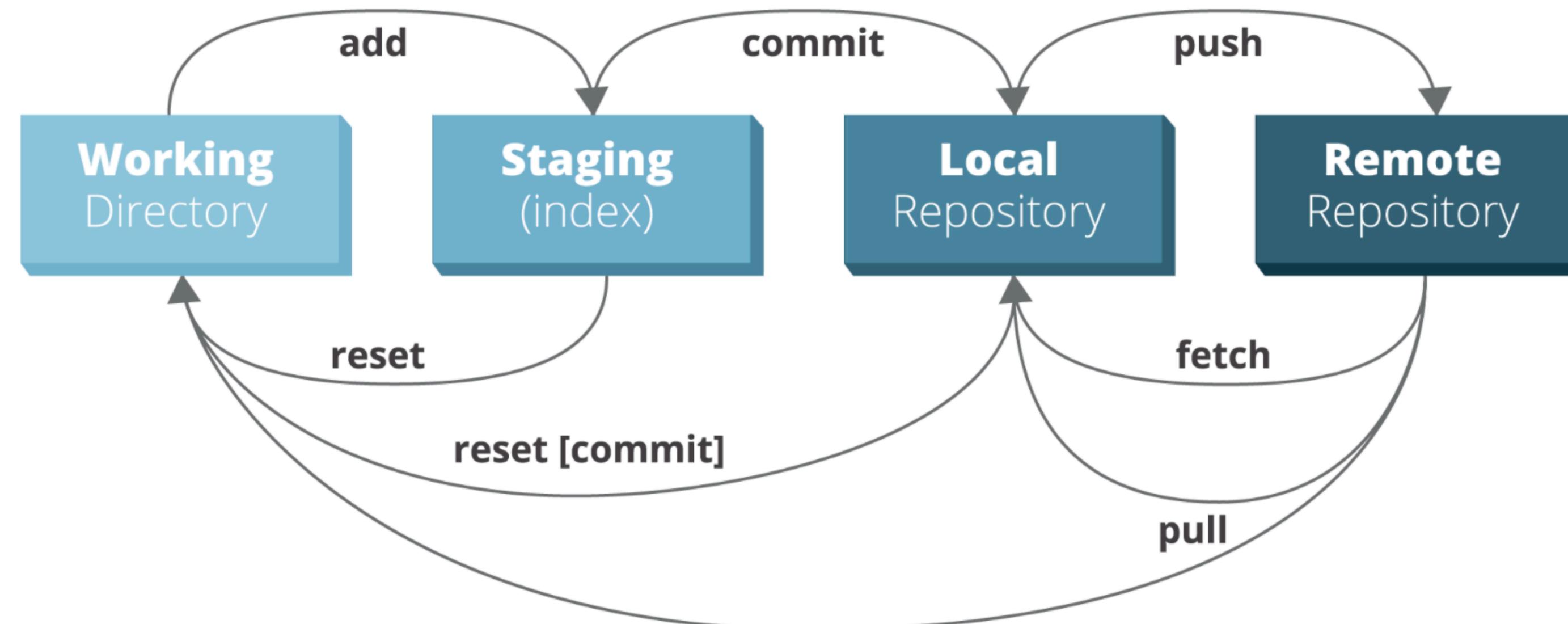
基础命令



Git基础命令



Git Stages 状态流程图



Git 创建

- Git init
- Git clone [url]
注意 https 和 ssh 的区别。
https 使用 http basic auth
所有会要求每次都输入密码。
- 学会查看 man
 1. man git-[subcmd]
 2. git cmd --help
 3. git help cmd

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) [git@github.com:hlxwell/gitflow-example.git](#)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# gitflow-example" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:hlxwell/gitflow-example.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:hlxwell/gitflow-example.git
git branch -M main
git push -u origin main
```

Git Remote 管理

- Git fetch
仅仅把 remote 服务器数据同步到本地的 remotes 下的分支。
git fetch -p 可以清理远程已删除的本地分支。
- Git pull [remote] [branch]
不加参数是把服务器上的所有分支下载到本地，加branch他会把 remote branch merge 到本地的branch。
git fetch && git merge branch origin/branch
- Git pull --rebase
加了rebase以后，就等于 **git fetch && git rebase branch origin/branch**
- Git push [--force]
把本地所有的分支 push 到 remote. 等于 **git push remote {branch1, branch2, ...}**
--force 的意思是，覆盖远程的分支，谨慎使用。
- Git push [remote] :branch
删除远程分支。
- Git remote add new-remote-name [url]
添加一个新的remote，常用于同步两个代码库的内容。

Git Local 查看

- Git status

查看当前working directory状态。包括增删改的文件。

git status --sb 简易版 status 并且显示分支。

- Git branch [-all]

查看本地所有的branch，如果加了--all 可以看到 remotes 的分支。

- Git diff

默认是查看 working directory 和 working branch 的 difference。也可以查看任意两个分支或者commit之间的差

别。生成 Patch: **git diff > xxx.patch, git apply xxx.patch**

- Git blame

查看哪行代码是谁在哪个commit里面写的。

- Git show [commit]:[file]

查看一个 commit 内容，或者 commit 下某个文件的变更。

- Git log [file|folder]

查看commit log 或者具体文件夹或者文件的log。

可以利用 git log branch1..branch2 去查看两个分支的差别。

- Git shortlog -sn

查看建议压缩版的log，经常用于查看 contributer 提交数量。

Git commit 管理

- Git add [.|file]
添加文件或者changes到 **Stage**。
- Git revert [commit-num]
把以前的commit内容去掉，并且生成一个新的 **commit**。
- Git cherry-pick [commit-num]
把任意一个commit给pick到当前分支。
- Git commit -m|-am “message”
提交stage到local branch
- Git reset [-soft|-hard] [commit] [file]
Unstage file, 把local branch 的 commit 删掉，删掉的内容有soft和hard两种处理方式。
-soft 的意思是把删掉commit放到stage，你可以重新整理再提交。
-hard 的意思是把删掉的commit内容全部从stage也删除，就是不要了。
Tip: **git checkout -- .** 等于 **git reset --hard**可以把stage的内容全部清除。
- Git clean -fx
删除 untracked files. (什么叫 untracked file?)
<https://www.jianshu.com/p/0b05ef199749>



Git Branch 管理

- Git checkout xxx-branch
切换分支
- Git checkout -b new-branch-name [start-from-branch]
创建新的local branch
- Git branch -d branch-name-to-delete
删除本地分支
- Git push remote :**branch-name-to-delete**
删除远程分支
- Git merge [—squash]
merge 分支, 默认是保留分支所有的commit。
加了 squash 会合并所有的 commits, 目的是
去掉那些没有意义的中间修改commit。
- Git merge --no-ff
merge 分支, 并且不采用 **fast-forward** 的方式。
为了创建一个merge commit, 在历史记录里清晰
表明merge过程。
- Git rebase [branch]
Rebase 当前 branch 到其他branch。
- Git rebase –onto NEW-BASE FROM-COMMIT TO-COMMIT
选择任意片段merge到任意commit。
- Git mergetool
一个处理conflict的工具

Git Stash 管理

- Git stash
Push所有work directory和stage的东西到stash。
- Git stash apply & Git stash pop
把最近一个stash恢复到work directory和stage, pop会删除stash, apply则不会。
- Git stash show [stash_id]
查看具体一个stash内容。
- Git stash list
查看历史stash。
- Git stash clear
删除所有stash。

Git Tag 管理

- Tag 主要用途就是为了发布版本
- Git tag new-tag-name
添加一个tag
- Git tag -l "v1.*"
通过regexp来列出 tags
- Git push origin –tags
上传所有的tags。
- Git push origin v1.2
上传单个tag。
- Git tag -d v1.2
删除tag, 怎么删除远程tag? (查看branch)

Git Revision

修订版本

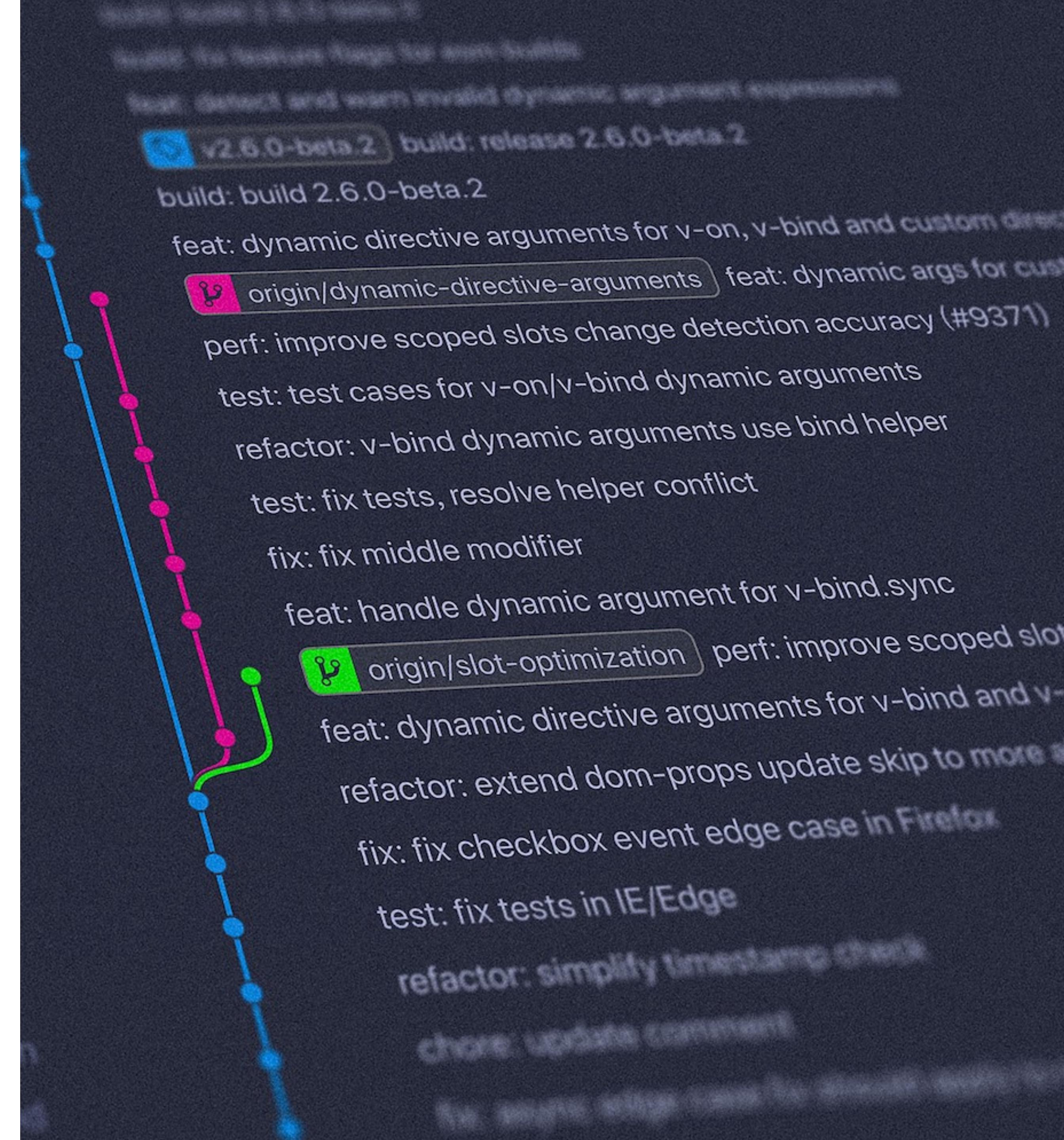
- HEAD
当前本地分支的最新一个 commit.
- HEAD[^]
HEAD的上一个commit
- HEAD^{^^^} == HEAD^{^3}
上上上一个版本
- HEAD^{~1}
当前本地分支的最新一个 commit.
- HEAD^{~3}
上上上一个版本

实战小任务

Are You Ready?

- 创建新的 git repo
- 创建 README.md
- 把 [自己名字.md] 放到项目里面来
- 在你名字的文件里面加上你的兴趣爱好
- 然后把自己的名字放到 README.md
- [老板说在README加个老板名字]
stash 一下，去改 README，然后回来恢复stash
- 合并分支
- 查看某个分支上面发生的所有事情
- 哥们A添加一个密码到自己.md文件
- 然后我们来帮忙把这个密码给销声匿迹
- 老王辞职了，去掉“老王”分支的所有变更

最佳实践



- 一个 Commit 一个目的
- Commit Message 必须有意义
- 使用 .gitignore 避免系统生成的文件和local config提交到repo。
- 不要直接 commit 到 master, 要常开分支
- 尽量不要改写 master 的 history. (Push –force)
- 在分支上开发的时候用 rebase 而不是 merge (pull --rebase)
- develop 到 master 需要用 merge 压缩(squash) commits
- 分支到 develop 需要用 merge without fast forward
- 使用 git-flow 来规范团队协作
- 使用 ~/.gitconfig 配置去简化操作

.gitconfig 配置

- 几大 section

- Core 基础配置，如编辑器，显示参数

- Alias 可以自定义很多快捷命令

git st => git status

git bd xxx => git branch -d xxx

git prom => git pull --rebase origin master

git pushod => git push origin develop

- User 提交时的用户信息

- Color 各种显示颜色

- 参考规范

<https://gist.github.com/pksunkara/988716>

```
68 [alias]
69   a = add --all
70   ai = add -i
71   #####
72   ap = apply
73   as = apply --stat
74   ac = apply --check
75   #####
76   ama = am --abort
77   amr = am --resolved
78   ams = am --skip
79   #####
80   b = branch
81   ba = branch -a
82   bd = branch -d
83   bdd = branch -D
84   br = branch -r
85   bc = rev-parse --abbrev-ref HEAD
86   bu = !git rev-parse --abbrev-ref --symbolic-full-name "@{u}"
87   bs = !git-branch-status
88   #####
89   c = commit
90   ca = commit -a
91   cm = commit -m
92   cam = commit -am
93   cem = commit --allow-empty -m
94   cd = commit --amend
95   cad = commit -a --amend
96   ced = commit --allow-empty --amend
97   #####
98   cl = clone
99   cld = clone --depth 1
100  clg = !sh -c 'git clone git://github.com/$1 $(basename $1)' -
101  clgp = !sh -c 'git clone git@github.com:$1 $(basename $1)' -
102  clgu = !sh -c 'git clone git@github.com:$(git config --get user.username)/$1 $1' -
```

.gitignore 配置

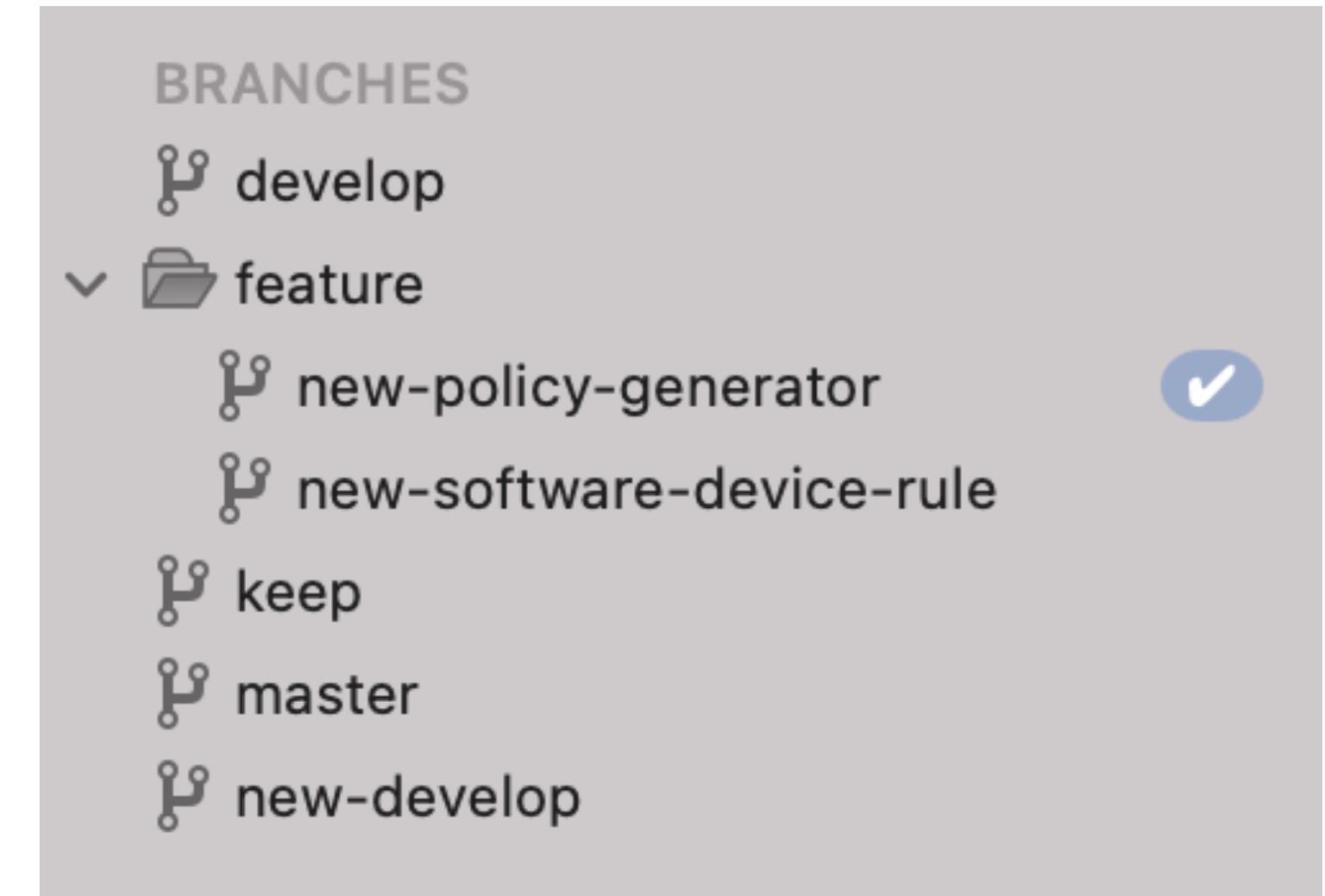
- 很多示例 project 的 .gitignore
<https://github.com/github/gitignore>
- 可以ignore
 - 本地配置文件
 - 临时文件
 - 日志文件
 - 本地编辑器配置文件 .vscode
 - 第三方的库本地缓存
 - 本地 build 出来的文件
 - 可以用 .keep 文件去保留文件夹。

```
69 lines (55 sloc) | 1.32 KB

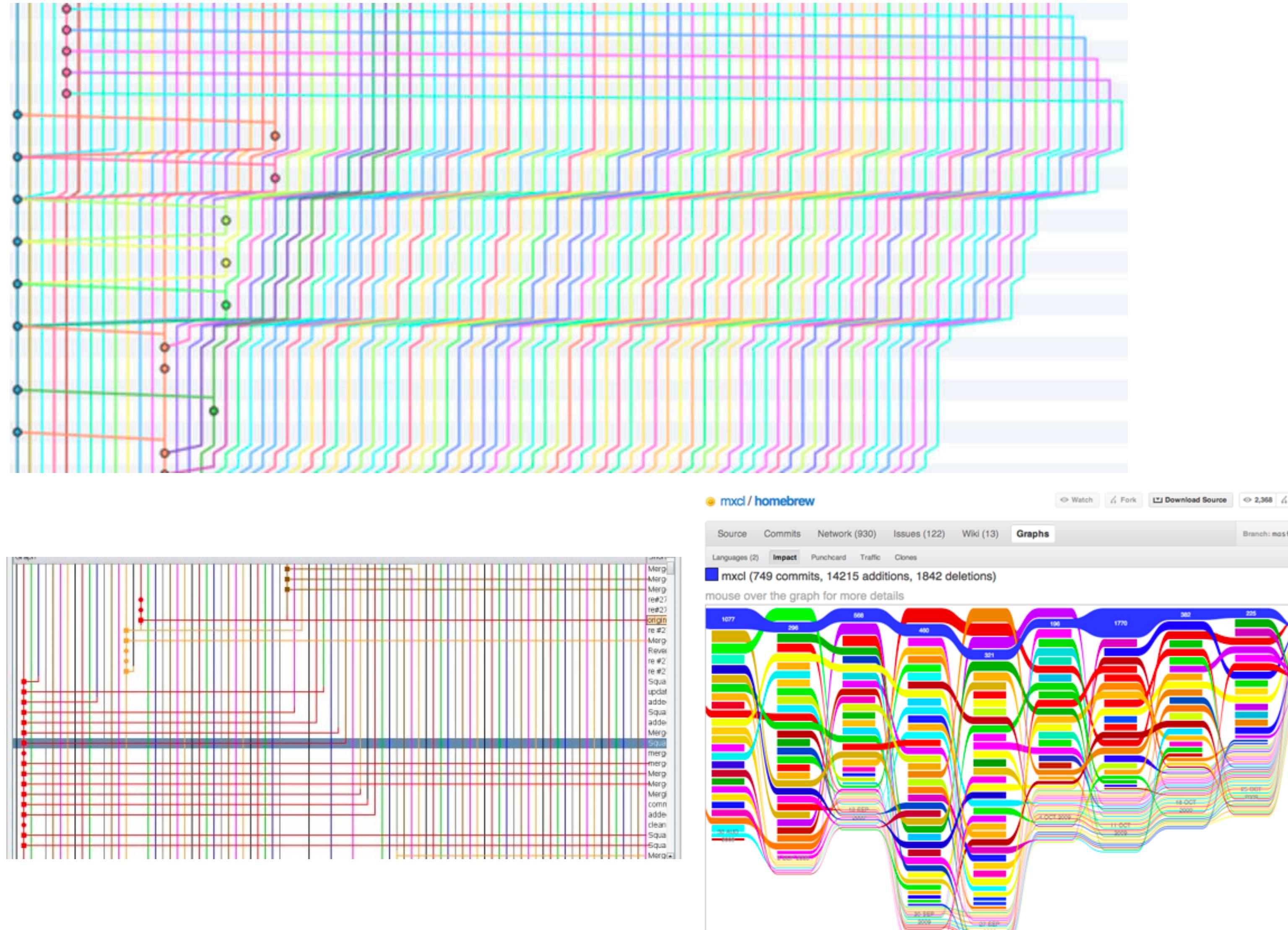
1 *.rbc
2 capybara-*.html
3 .rspec
4 /db/*.sqlite3
5 /db/*.sqlite3-journal
6 /db/*.sqlite3-[0-9]*
7 /public/system
8 /coverage/
9 /spec/tmp
10 *.orig
11 rerun.txt
12 pickle-email-*.html
13
14 # Ignore all logfiles and tempfiles.
15 /log/*
16 /tmp/*
17 !/log/.keep
18 !/tmp/.keep
19
20 # TODO Comment out this rule if you are OK with secrets being uploaded to the repo
21 config/initializers/secret_token.rb
22 config/master.key
23
24 # Only include if you have production secrets in this file, which is no longer a Rails default
25 # config/secrets.yml
26
27 # dotenv, dotenv-rails
28 # TODO Comment out these rules if environment variables can be committed
29 .env
30 .env.*
31
32 ## Environment normalization:
33 /.bundle
34 /vendor/bundle
35
```

如何命名分支

- 以一个 group 起头
 - `feature/your-feature`
 - `hotfix/your-hotfix`
 - `fix/your-fix`
 - `chore/your-chore-work`
 - `wip/refactory-everything`
- [坑] 基于 **feature/xxx** 再建立子分支叫做 **feature/xxx/aaa** 为什么不行?
- 如果使用 issue tracker 比如gitlab issue, 那么可以使用 issue ID
fix/1928-fix-popup-js-bug
- 使用 **- or _** 作为分隔字符



To avoid the chaos...



```
Merge branch 'devel' into 3.10
<3.1>
Me
BL
I3
NE
<3>
FI
<3>
Merge branch 'devel' into 3.9
o FIX. add a case for gender is not "m" and
<3.9.16> BLD. 3.9.16
Merge branch 'devel' into 3.9
<3.9.15> BLD. 3.9.15
Merge branch 'devel' into 3.9
<3.9.14> BLD. 3.9.14
Merge branch 'devel' into 3.9
<3.9.13> BLD. 3.9.13
Merge branch 'devel' into 3.9
<3.9.12> BLD. 3.9.12
Merge branch 'devel' into 3.9
<3.9.11> BLD. 3.9.11
Merge branch 'devel' into 3.9
<3.9.10> BLD. 3.9.10
Merge branch 'devel' into 3.9
<3.9.9> BLD. 3.9.9
Merge branch 'devel' into
<3.9.8> BLD. 3.9.8
FIX. check EmailAddress is
<3.9.7> BLD 3.9.7
Merge branch 'devel' into
<3.9.6> BLD 3.9.6
Merge branch 'devel' into
<3.9.5> BLD 3.9.5
Merge branch 'devel' into
<3.9.4> BLD. 3.9.4
Merge branch 'deve
<3.9.3> BLD 3.9.3
Merge branch 'de
<3.9.2> BLD 3.9.
Merge branch '
<3.9.1> BLD 3.
Revert "BLD 3.
BLD 3.8.1
FIX. i
FIX. f
FIX. f
UPD. F
UPD. u
UPD. c
UPD. F
UPD. r
UPD. f
FIX. f
FIX. f
FIX. f
FIX. f
UPD. d
UPD. d
UPD. g
UPD. u
```

2014-11-18 08:00 t

[main] 4f2c7640edeb14c1d471b77c59148008644eae7d - commit 283 of 2658

如何提交流程

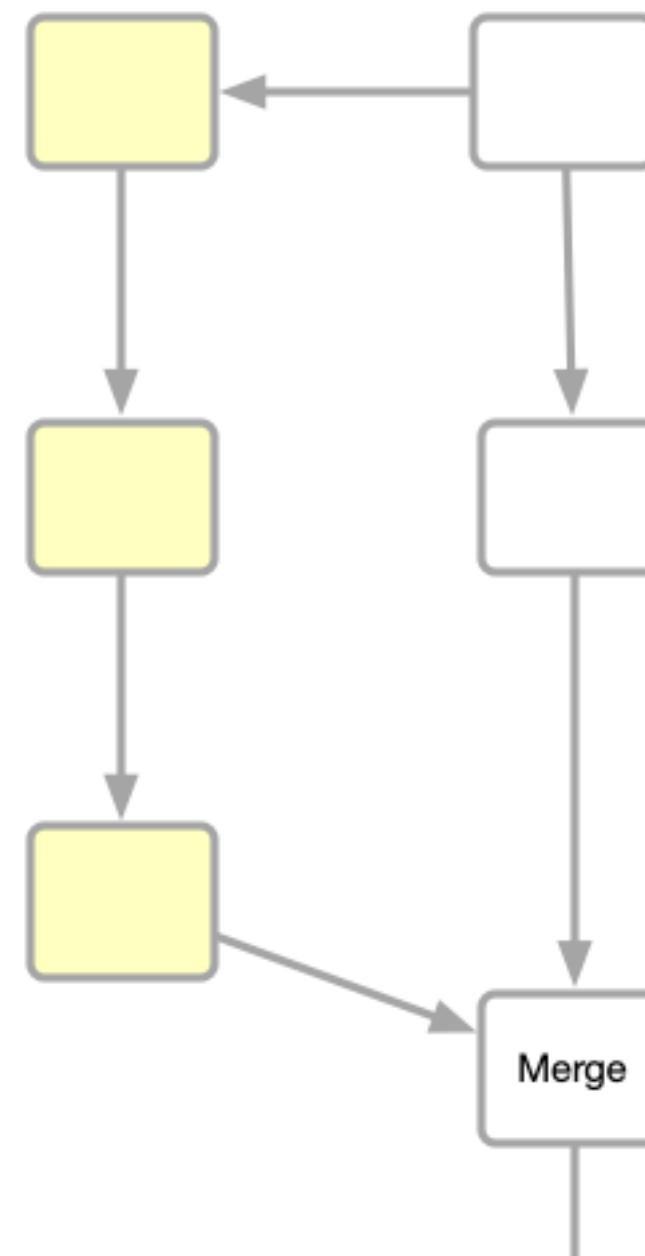
- 首先你需要更新本地分支。
git pull origin --rebase feature/your-amazing-feature
- 如果不加 --rebase 会出现一些merge。



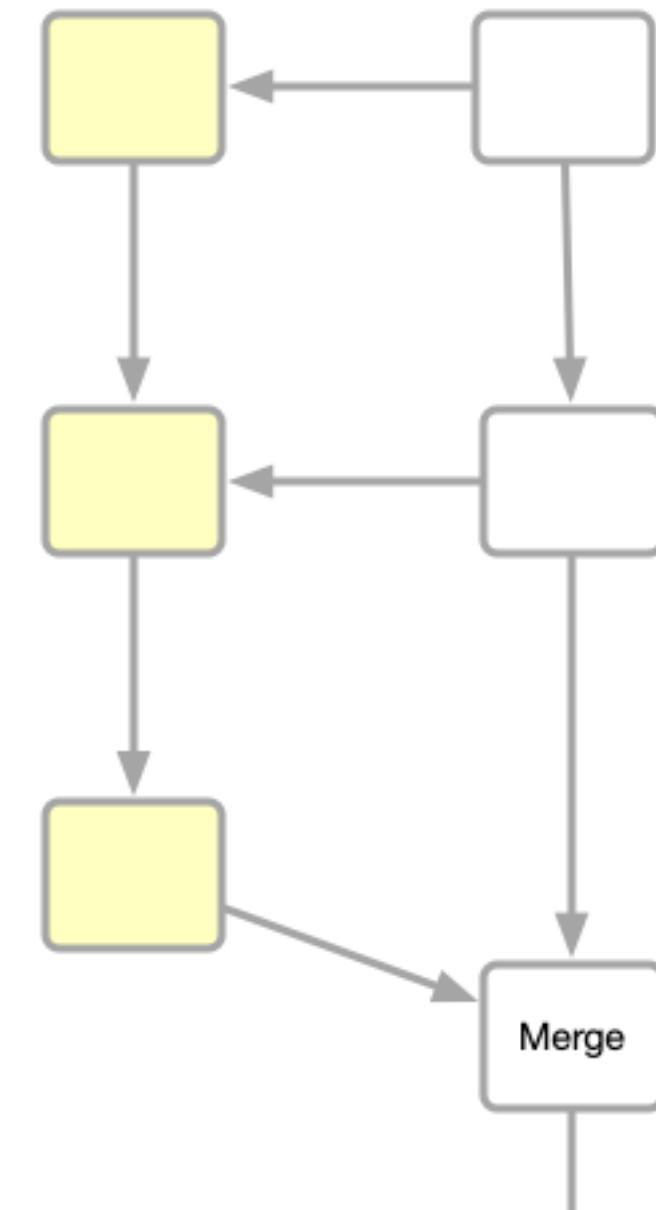
右图是期待分支，以及合并后的样子。

原因是，为了分支管理的时候便于随时去掉某个分支的内容，最重容易管理。

- 在分支上面开发，尽量保持一条直线，除非是一个分支又有开叉，比如子功能。
- 然后 rebase 一下 develop 分支并且解决冲突。
- 解决完后，即可 push 回 origin。
- 别merge，看下面。(而且一般情况下你没权限。)



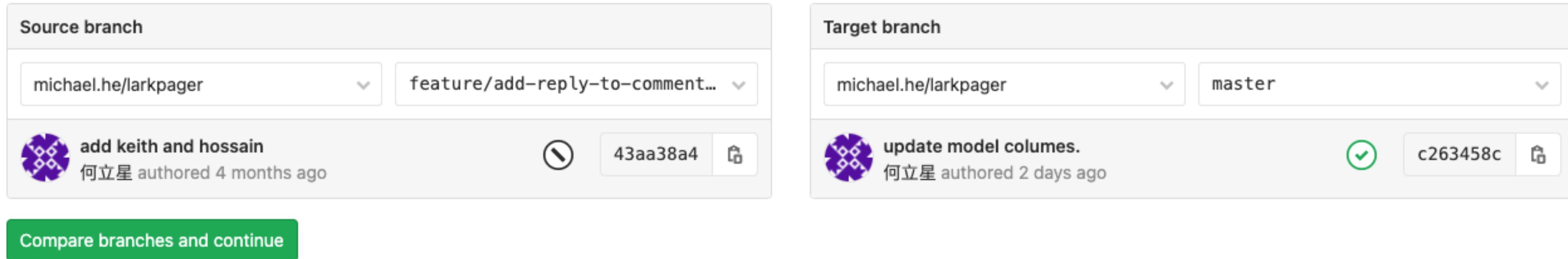
期待分支的样式



而不是分支的样式

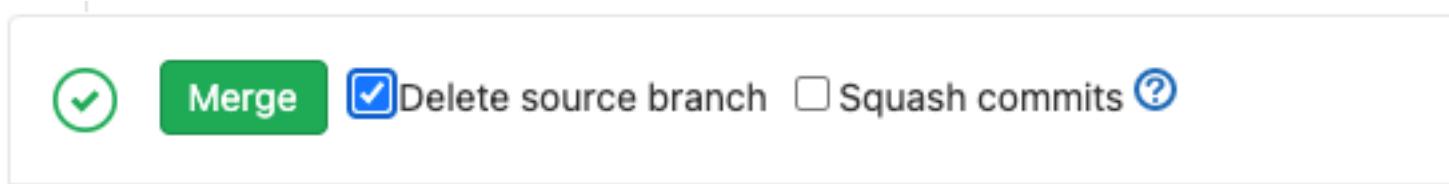
如何合并分支

- 在 gitlab 上面点击 Create Merge Request。
- 然后选择你的分支和目标合并分支。



The screenshot shows the GitLab Merge Request interface. On the left, the 'Source branch' section shows a dropdown for the source repository 'michael.he/larkpager' and a dropdown for the source branch 'feature/add-reply-to-comment...'. Below this, a commit card for 'add keith and hossain' by '何立星' is shown, with a purple profile picture and a timestamp of 'authored 4 months ago'. On the right, the 'Target branch' section shows a dropdown for the target repository 'michael.he/larkpager' and a dropdown for the target branch 'master'. Below this, a commit card for 'update model columes.' by '何立星' is shown, with a purple profile picture and a timestamp of 'authored 2 days ago'. A green checkmark icon is next to the commit. At the bottom left, a green button says 'Compare branches and continue'.

- 然后让团队成员 review 一下你的 merge request 内容。
- 最后让有权限的人来合并一下即可。
- 合并的时候可以选择删除分支，大部分情况不会用了。



squash commits 的用途是在你的分支做了很多碎小的没意义commits时，你可以squash。(比如，加了行代码，又去掉了，又加回来了，又去掉了.....)

解决冲突流程

- 搜索所有的“<<<<<<<”就能找到所有的冲突的地方。
- 需要对本身修改的代码有所了解才能修复冲突。
- 修复完后最好要跑一下测试。
- 如果发现不明白的地方最好找一下那个分支的作者，一些商讨一下。

git mergetool

```
1 # Gitflow Example
2
3 Just for sharing the knowledge within the team.
4
5 - hello jet.

1 # Gitflow Example
2
3 Just for sharing.
4
5 ## Steps
6
7 - basic command lines
8 - add git push

feature/add-test > README_LOCAL_34681.md <2 | feature/add-test > README_BASE_34681.md <17 | feature/add-test > README_REMOTE_34681.md <

1 # Gitflow Example
2
3 Just for sharing the knowledge within the team.
4
5 Just for sharing.
6
7 ## Steps
8
9 - hello jet.
10 - basic command lines
11 - add git push
```

NORMAL > feature/add-test > README.md +

1 more line; before #4 2 seconds ago

unix < utf-8 < markdown < 9% < 1:1

git mergetool

1 README.md >  # Gitflow Example >  ## Steps

2 You, 4 minutes ago | 2 authors (You and others)

3 **# Gitflow Example**

4

5 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

6 **<<<<< HEAD (Current Change)**

7 Just for sharing the knowledge within the team.

8

9 – hello jet.

10 =====

11 Just for sharing.

12

13 You, 4 minutes ago | 1 author (You)

14 **## Steps**

15

16 – basic command lines

17 – add git push

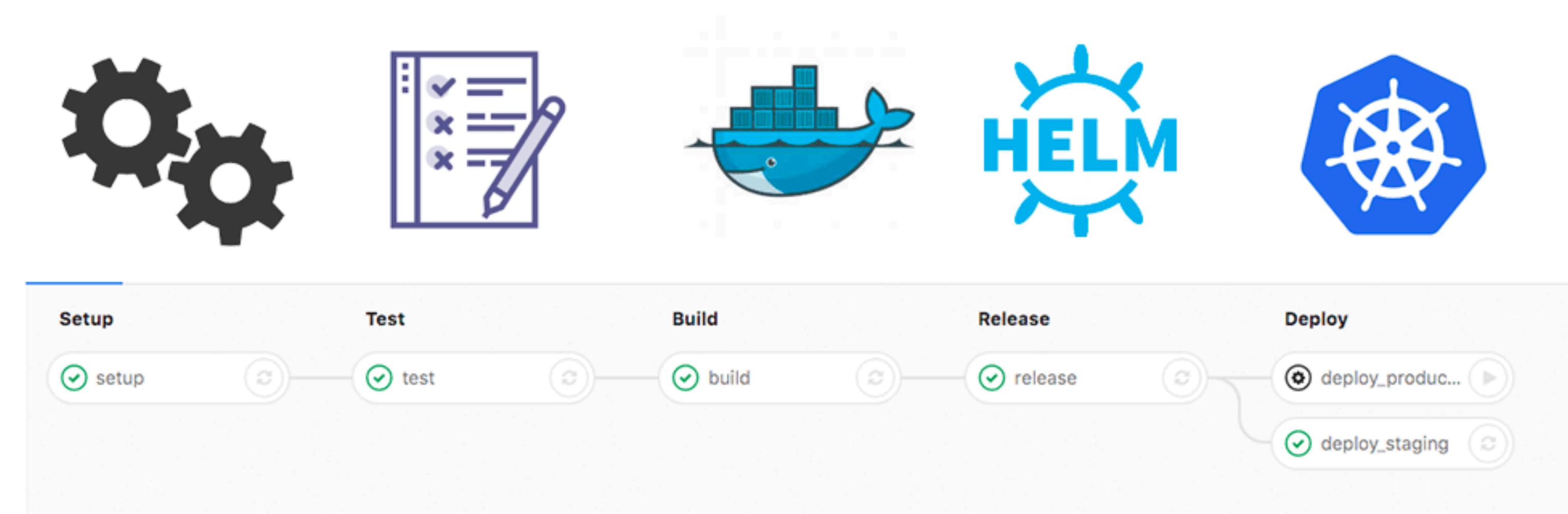
18 **>>>>> develop (Incoming Change)**

19

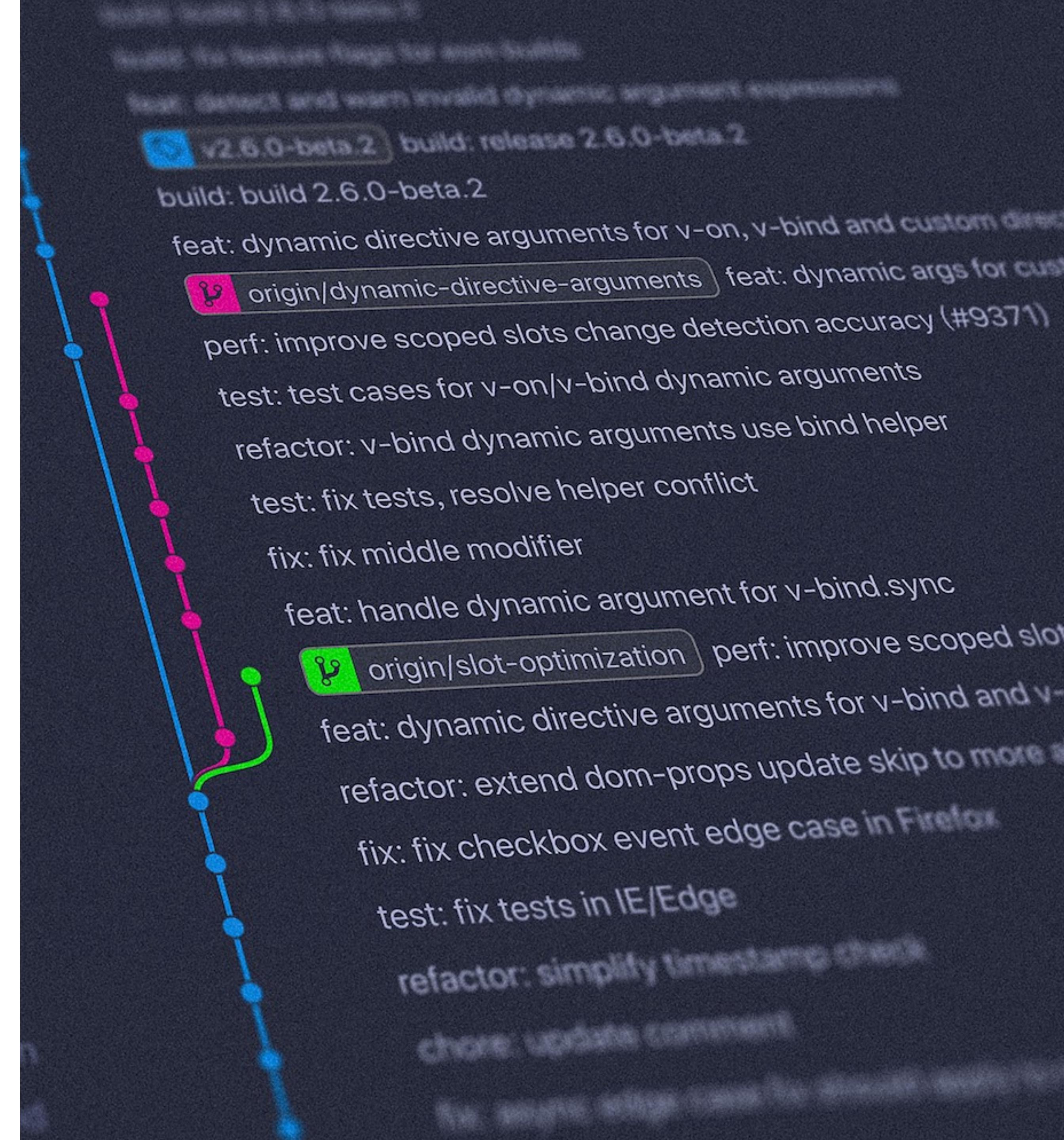
Vscode

如何发布

- 一般会通过 Gitlab CI/CD 进行测试，打包，并且发布。
- 在 gitlab CI 配置文件里面可以设置 当检测到新tag的时候，触发部署到production流程。
- 整个Pipeline的配置和设计是另外一个分享内容。

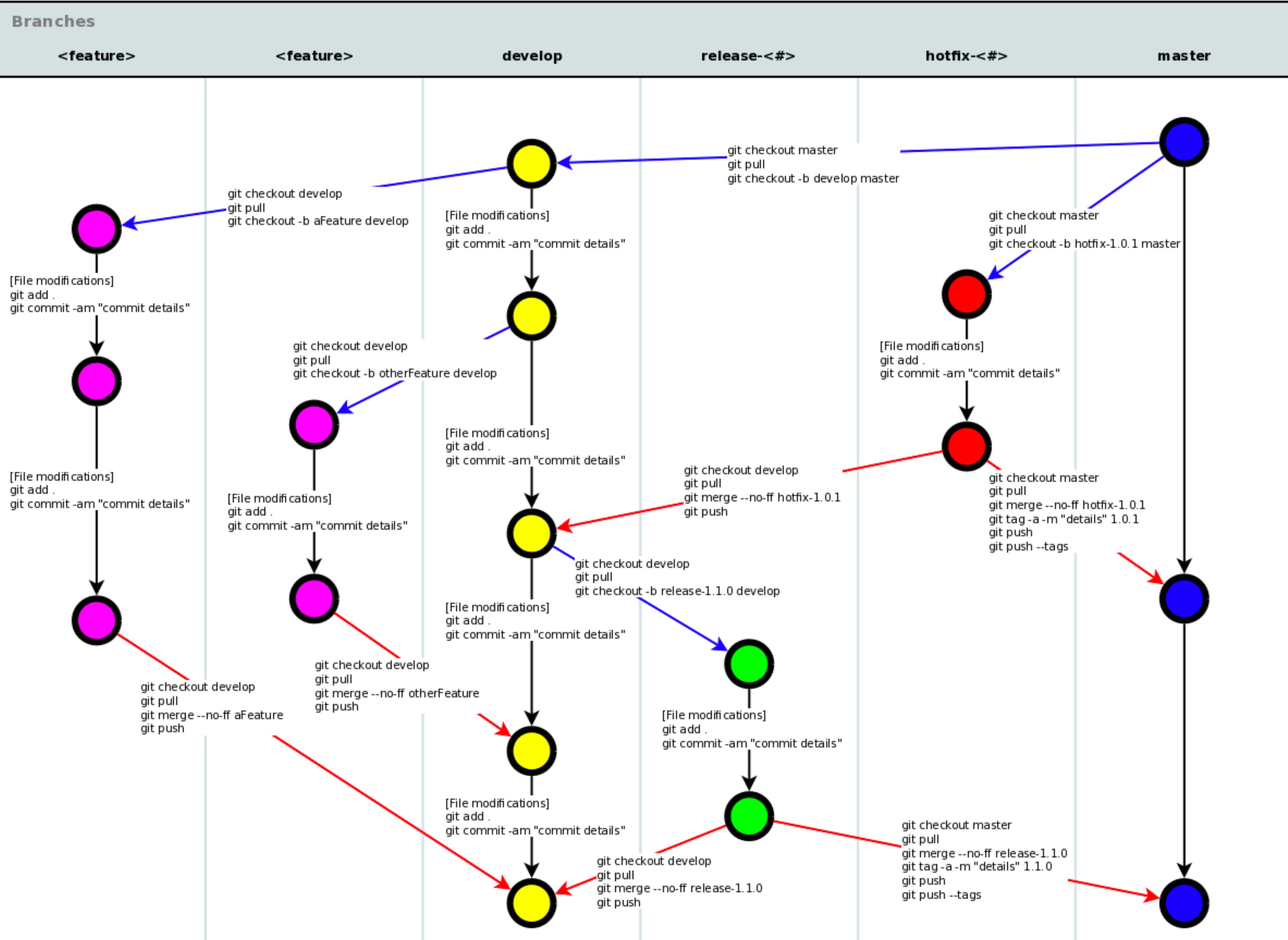


分支管理

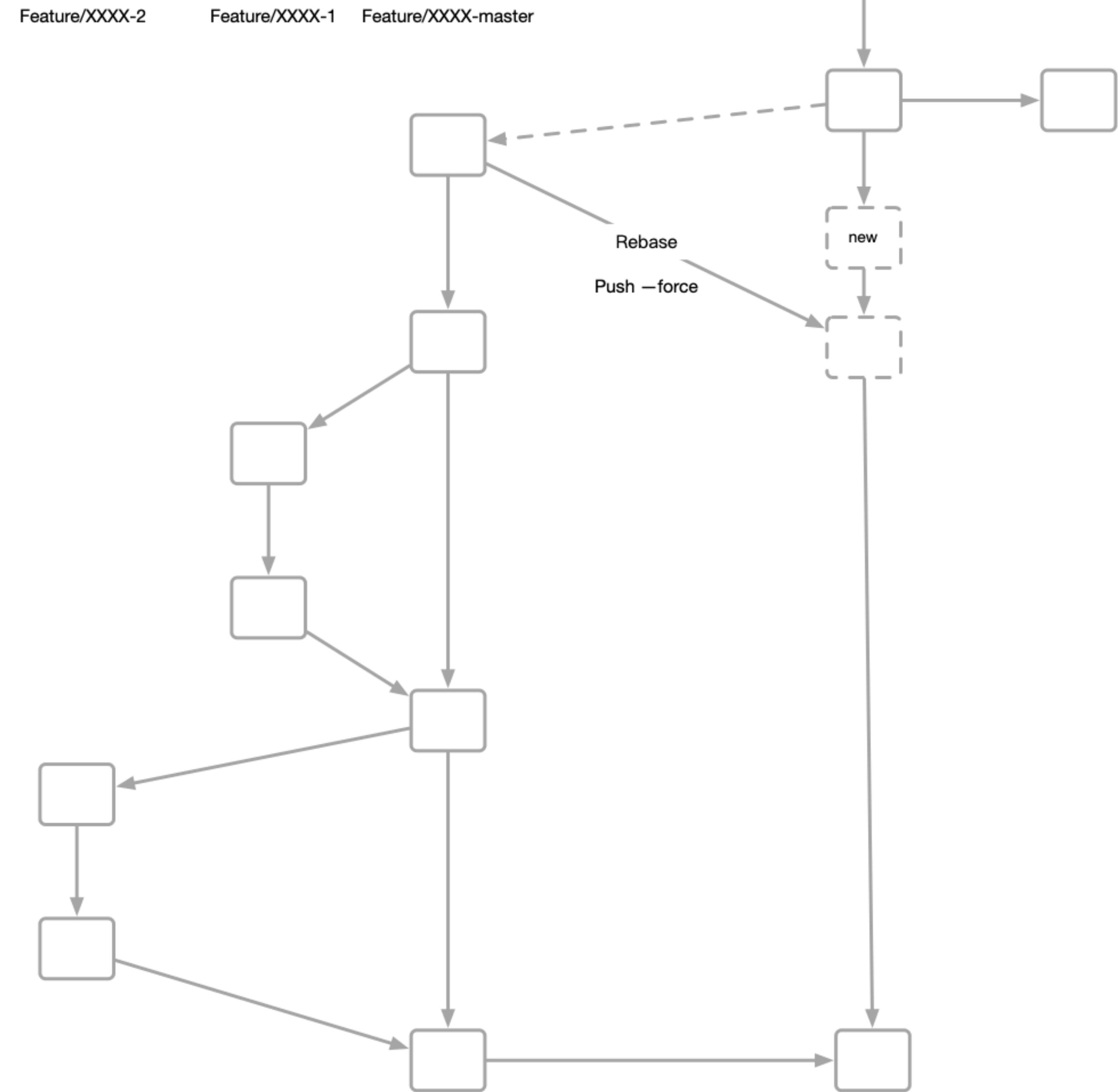


分支类型

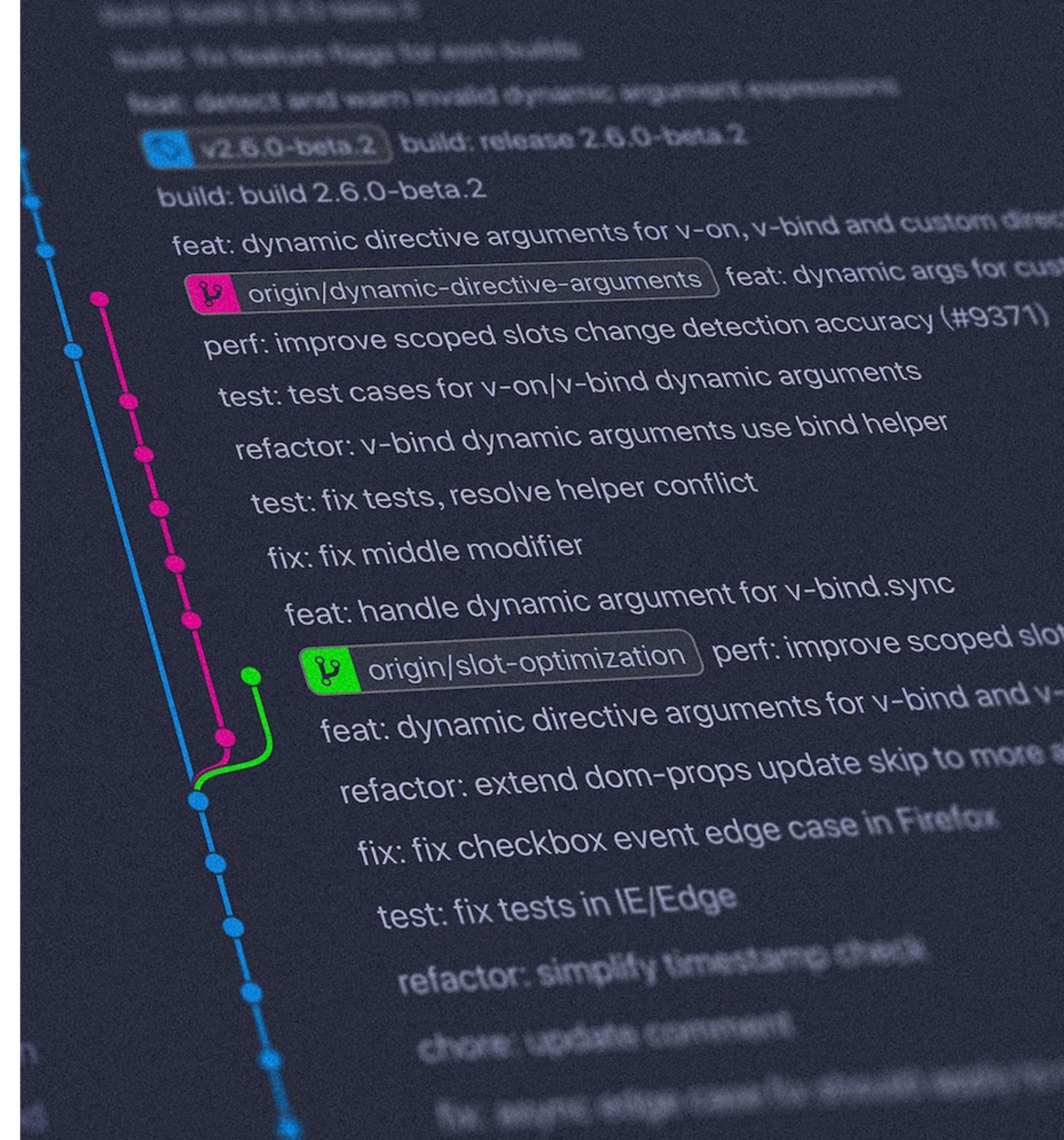
- 长期分支
 - Develop
 - Master
- 任务分支
 - Feature
 - Release
 - Hotfix



- 当多人在同一个feature上协作的时候，可以考虑在feature分支上面再开分支。
- 在合并到 develop 分支之前，需要将所有的子分支合并后再合并到 develop 分支。

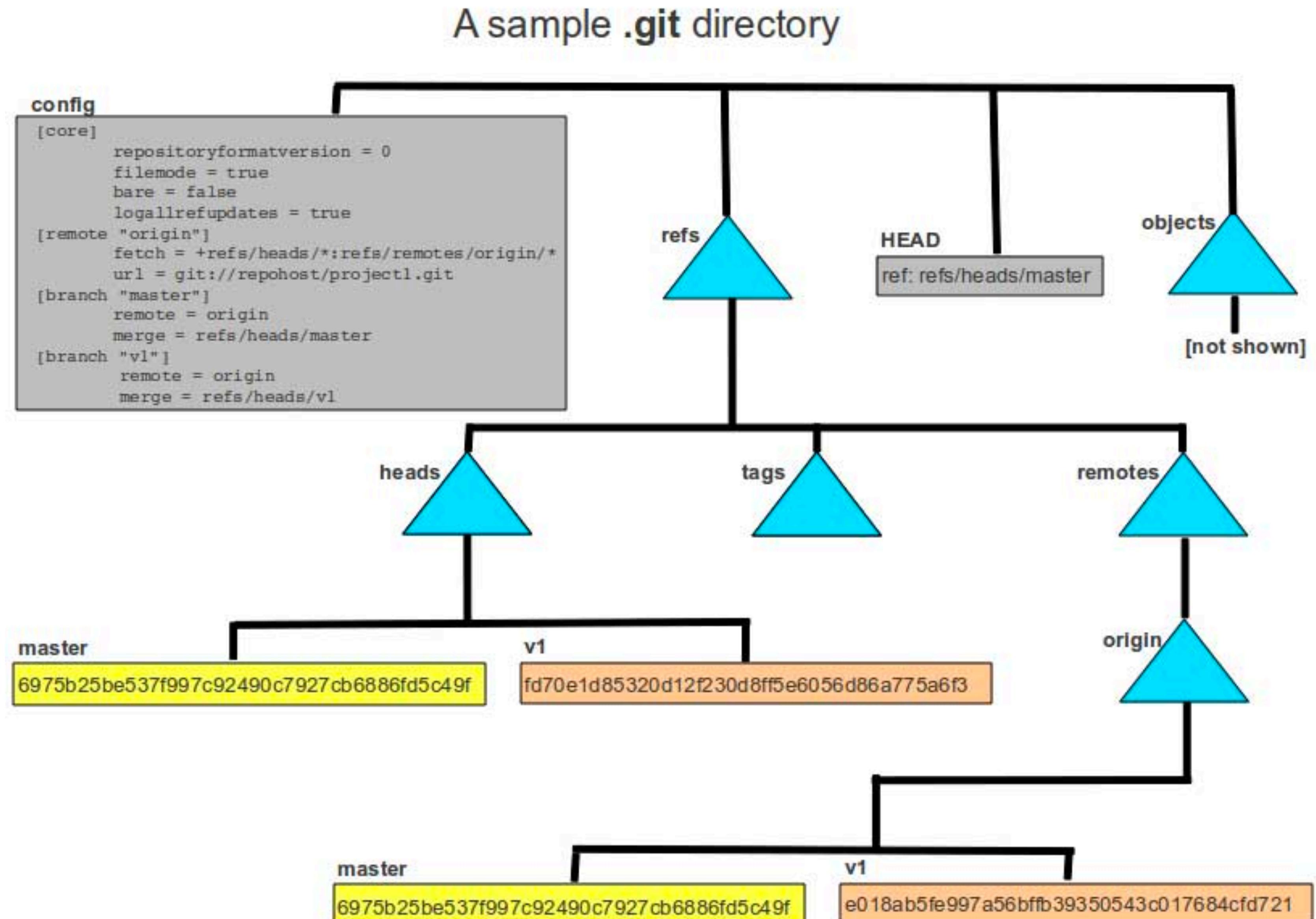


深入.git 目录



文件结构

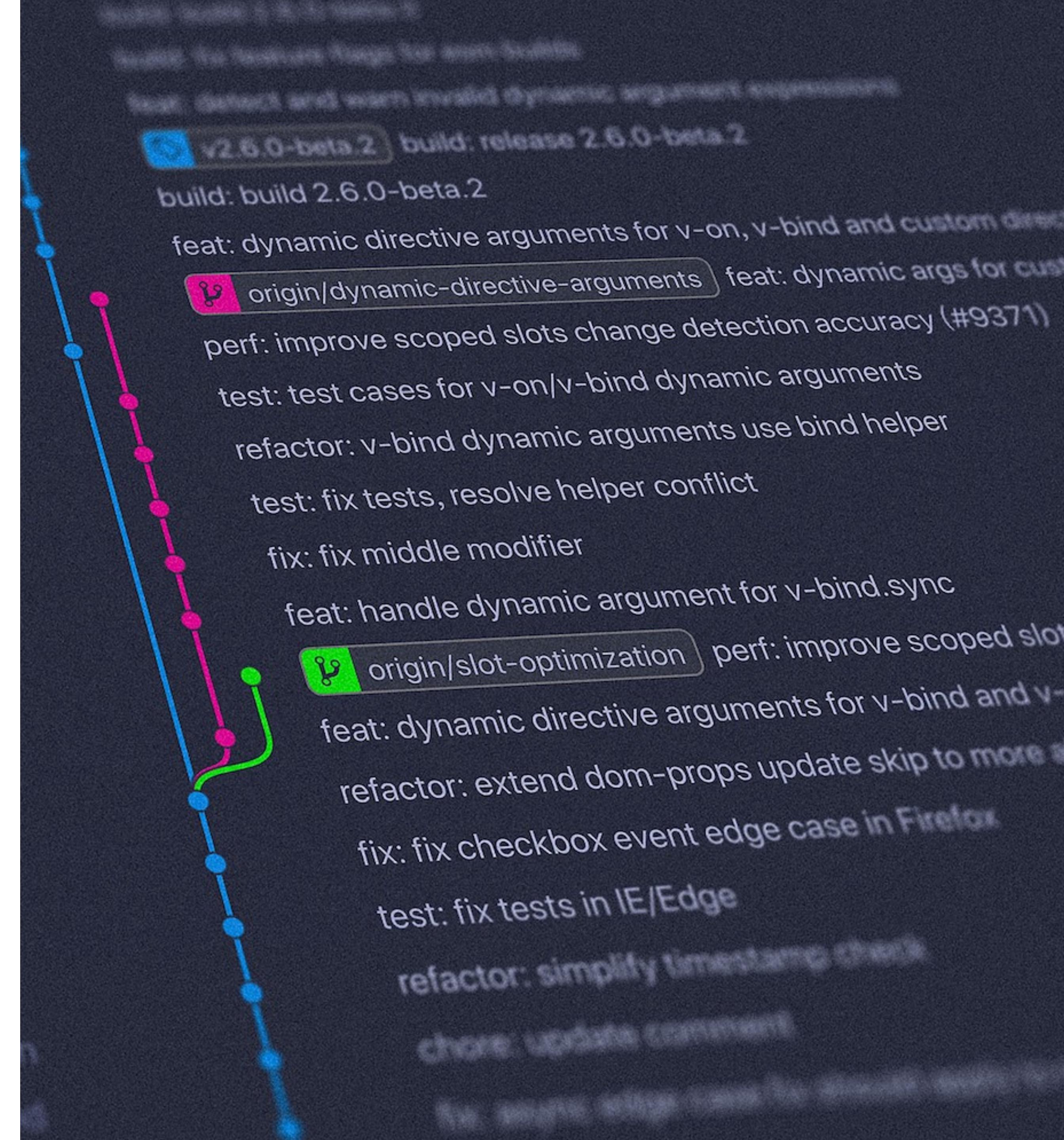
- .git
 - COMMIT_EDITMSG
 - FETCH_HEAD
 - HEAD
 - ORIG_HEAD
 - config
 - description
 - index
 - packed-refs
 - **hooks/ (高级功能)**
<https://segmentfault.com/a/1190000022970270>
 - **info/**
 - **logs/**
 - **objects/**
 - **refs/**
- 听说还能恢复出 git clean 删除的文件呢
<https://www.codenong.com/46257929/>



Bare git repo

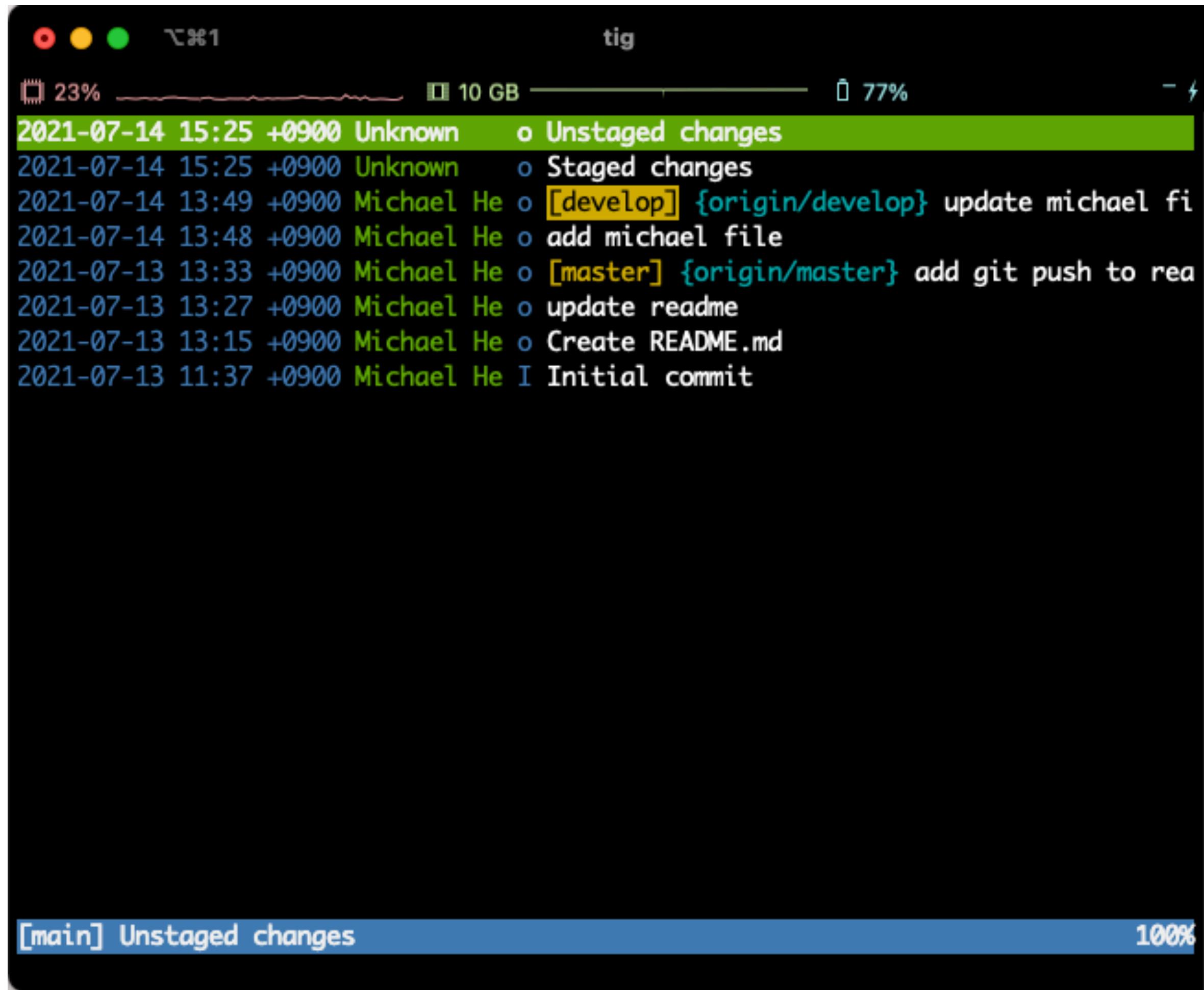
- HEAD
- config
- description
- *COMMIT_EDITMSG*
- *FETCH_HEAD*
- *ORIG_HEAD*
- *index*
- *packed-refs*
- **Hooks/**
- **info/**
- **logs/**
- **objects/**
- **refs/**
- Bare git repo
通过 git init --bare . 创建。
- 它的没有working directory, 以及checkout出来的文件。
- 它的用途是作为分享库, 而不是工作库用的。也就是说你可以通过一个 ssh 服务器去自己搭建一个git repo代替使用gitlab。
- 你可以通过下面的命令 clone
git clone ubuntu@127.0.0.1:/path/to/bare-repo

推荐工具



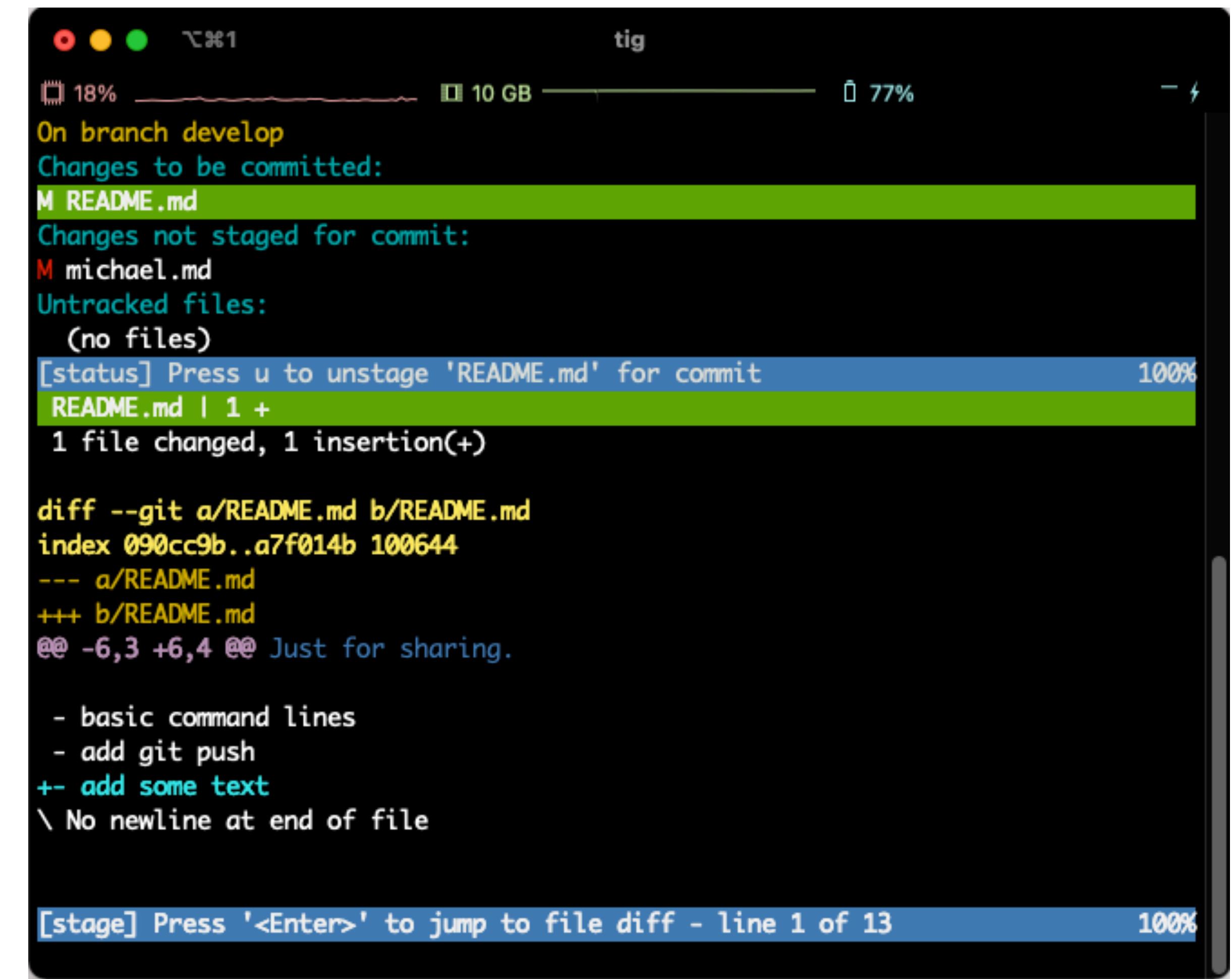
Tig

Command Line Git management tool



2021-07-14 15:25 +0900 Unknown o Unstaged changes
2021-07-14 15:25 +0900 Unknown o Staged changes
2021-07-14 13:49 +0900 Michael He o [develop] {origin/develop} update michael fi
2021-07-14 13:48 +0900 Michael He o add michael file
2021-07-13 13:33 +0900 Michael He o [master] {origin/master} add git push to rea
2021-07-13 13:27 +0900 Michael He o update readme
2021-07-13 13:15 +0900 Michael He o Create README.md
2021-07-13 11:37 +0900 Michael He I Initial commit

[main] Unstaged changes 100%



On branch develop
Changes to be committed:
M README.md
Changes not staged for commit:
M michael.md
Untracked files:
(no files)
[status] Press u to unstage 'README.md' for commit 100%

README.md | 1 +
1 file changed, 1 insertion(+)

```
diff --git a/README.md b/README.md
index 090cc9b..a7f014b 100644
--- a/README.md
+++ b/README.md
@@ -6,3 +6,4 @@ Just for sharing.

- basic command lines
- add git push
+- add some text
\ No newline at end of file
```

[stage] Press '<Enter>' to jump to file diff - line 1 of 13 100%

Gitx

Free GUI Git management tool

gitflow-example (branch: develop)

Stage

All Local “develop”

S	Subject	Author	Date
c	develop origin/develop update michael file	Michael He	July 14, 2021 at 13:49
6	add michael file	Michael He	July 14, 2021 at 13:48
2	master origin/master add git push to readme	Michael He	July 13, 2021 at 13:33
4	update readme	Michael He	July 13, 2021 at 13:27
9	feature/add-test update the readme	Michael He	July 13, 2021 at 13:26
2	Create README.md	Michael He	July 13, 2021 at 13:15
4	Initial commit	Michael He	July 13, 2021 at 11:37

Subject: update michael file
Author: Michael He <michael.he@bytedance.com>
Date: Wed Jul 14 2021 13:49:25 GMT+0900 (JST)
SHA: cc68820b98cfdb6961a3c8e83f8cff0ed234ac5d
Refs: develop origin/develop
Parent: 62a62a98829c8ace06eb61ff0b63ba1881c6ff46

update michael file

michael.md

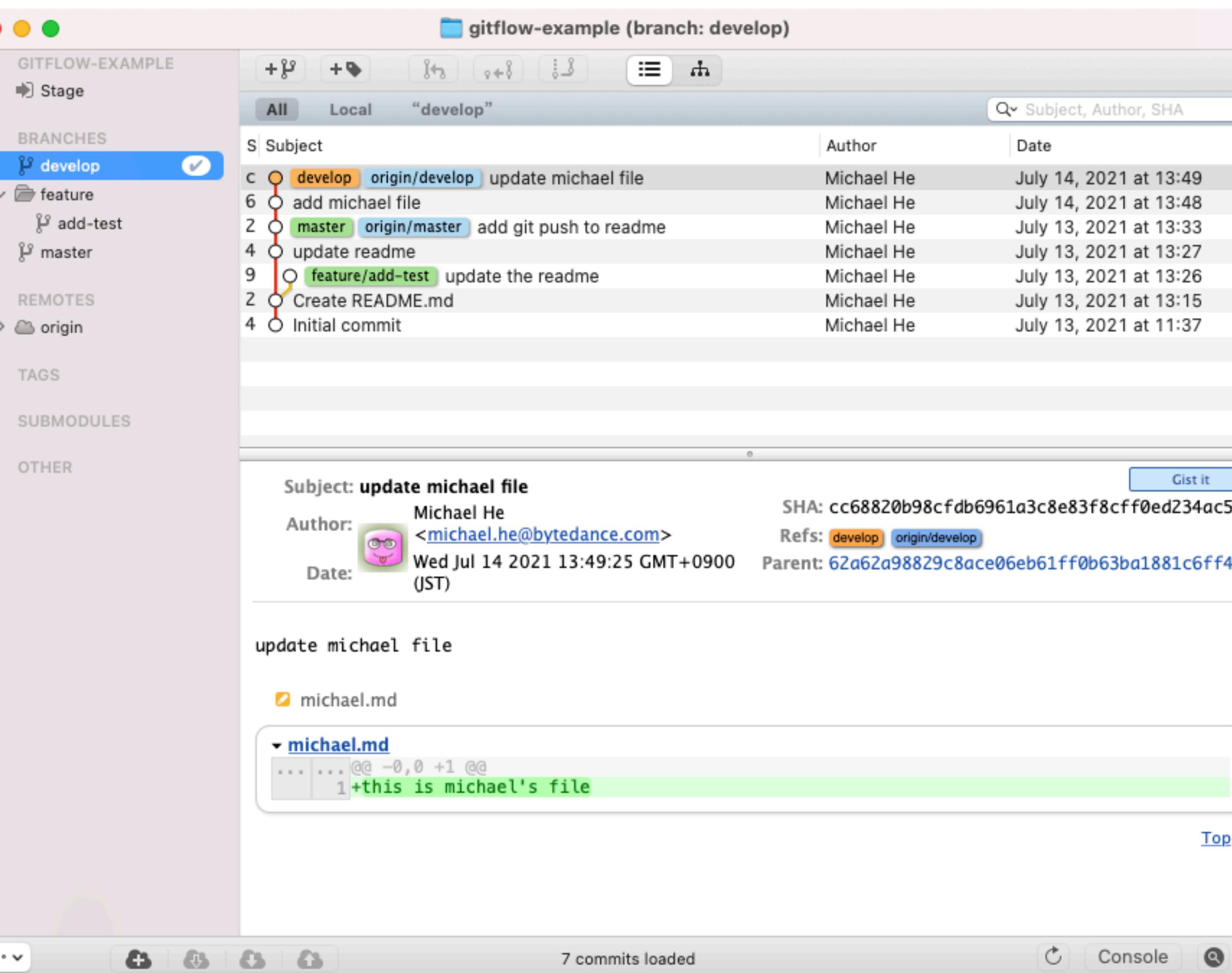
michael.md

... @@ -0,0 +1 @@
1+this is michael's file

Top

7 commits loaded

Console



gitflow-example (branch: develop)

Stage

BRANCHES

develop

feature

add-test

master

REMOTES

origin

TAGS

SUBMODULES

OTHER

Unstaged changes for michael.md

michael.md

... @@ -1 +1,2 @@
1 1 this is michael's file
2 +加一些文字。

Stage Discard

Unstaged Changes Commit Message Staged Changes

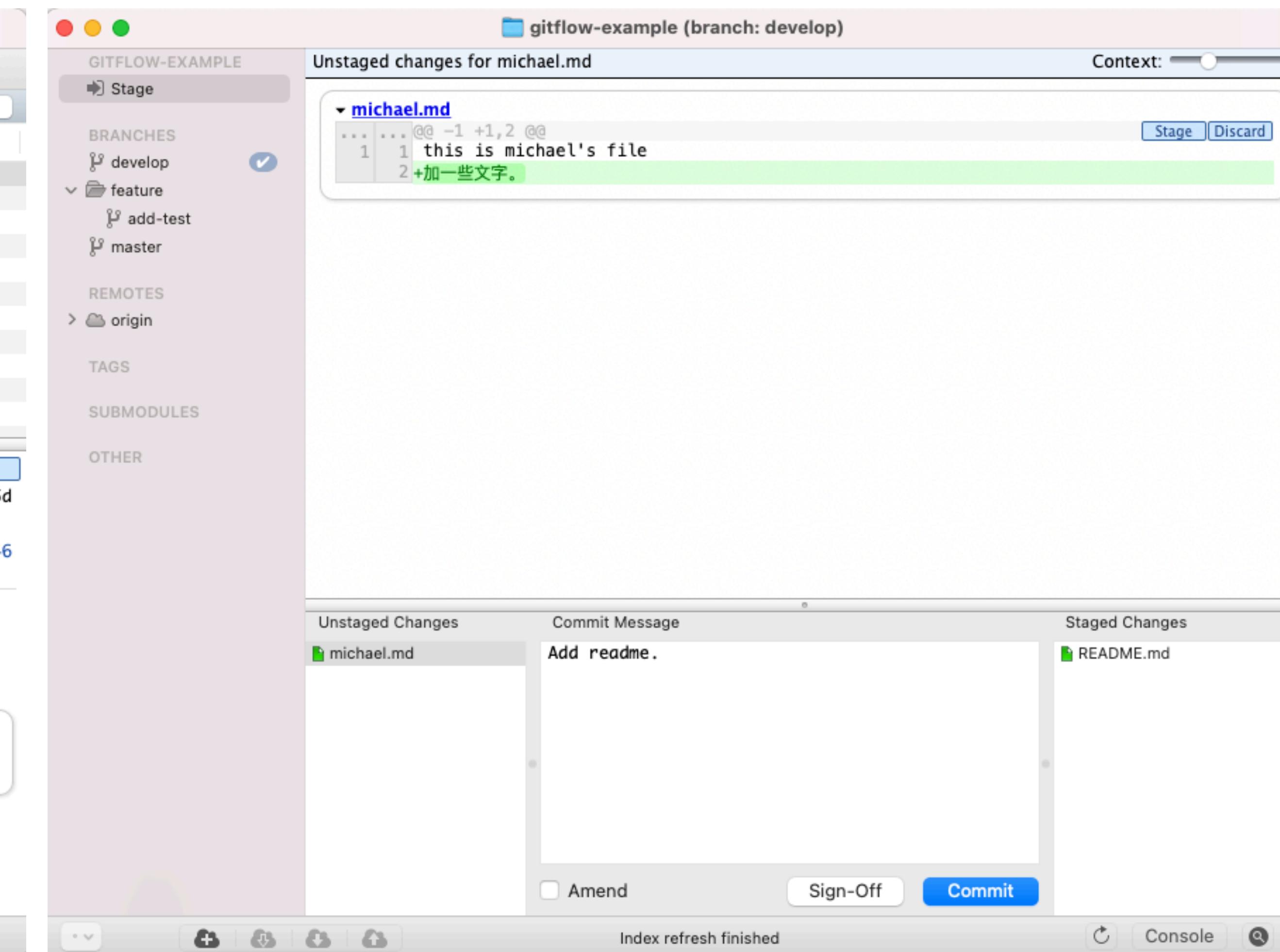
michael.md Add readme.

README.md

Amend Sign-Off Commit

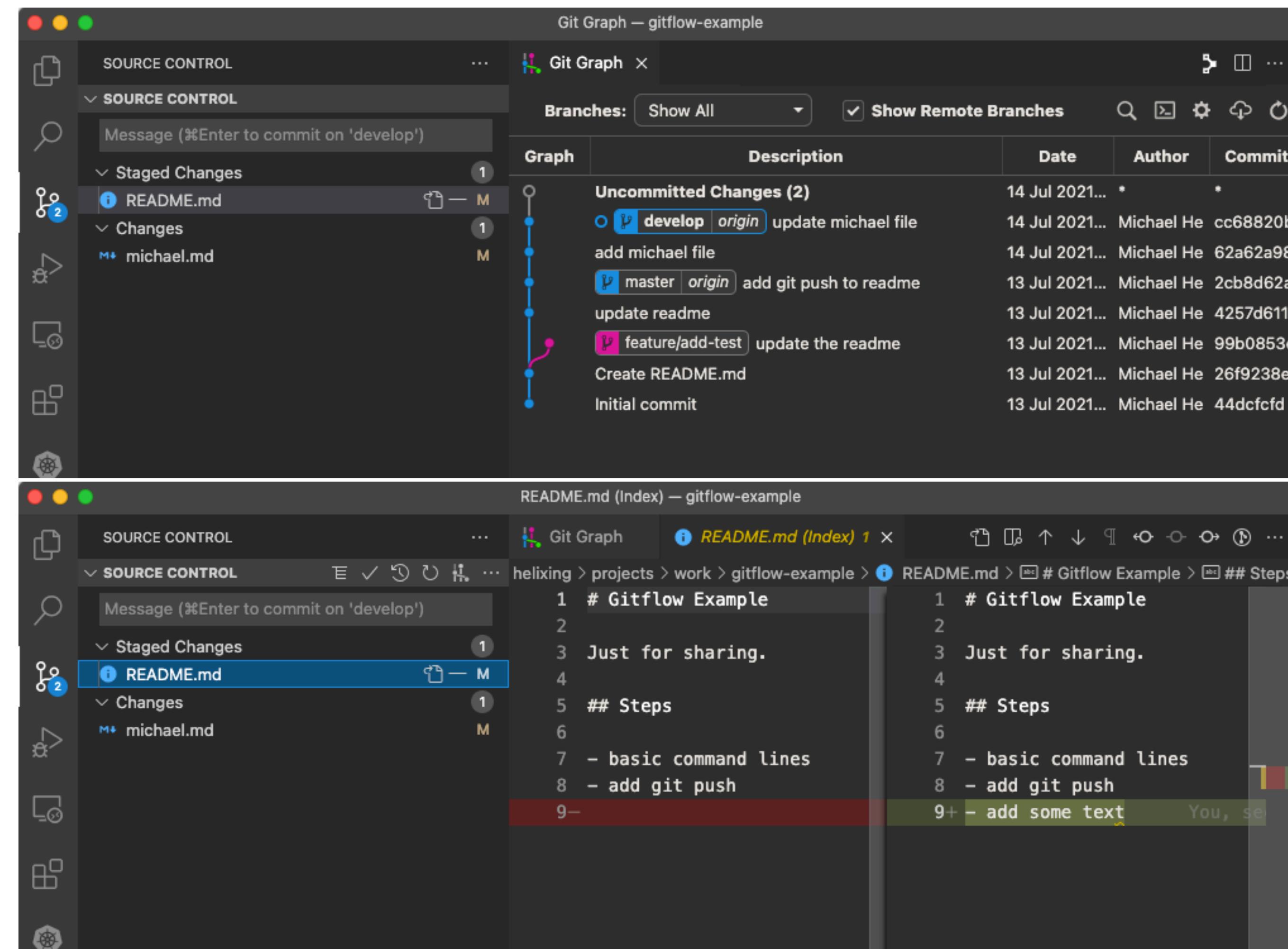
Index refresh finished

Console



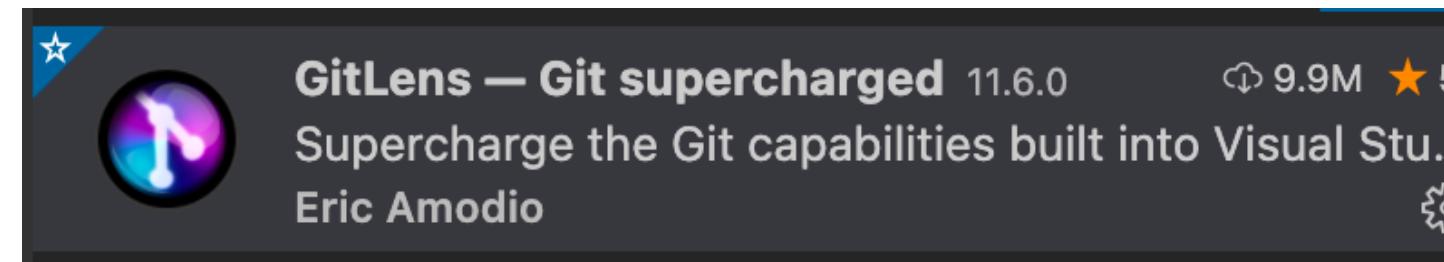
Vscode Plugin

GUI Git management tool



Vscode Plugin

GitLens - Inline Git blame

A screenshot of a Visual Studio Code editor window demonstrating the GitLens extension. The code editor shows a file with the following content:

```
11  ↗ 0593ba4  ↘  |  Team...  |  ...
12  🛡 hechenghan, a day ago (July 19th, 2021 12:27pm)  m()
13
14  生成api.ts
15
16  ↗ 0593ba4  ↘  |  Team...  |  ...
17  for i := 0; i < beforeElem.NumField(); i++ {  hechenghan, a day ago • 生成api.ts
18  |  // 在要修改的结构体中查询有数据结构体中相同属性的字段, 有则修改其值
```

Annotations from the GitLens extension are visible: a blue status bar at the top of the code editor, a blue status bar at the bottom, and a blue background highlight for the line 'for i := 0; i < beforeElem.NumField(); i++ {'. The line 'hechenghan, a day ago • 生成api.ts' is also highlighted in blue.

Q&A