

# 格蒂Git 使用规范

---

团队开发中，遵循一个合理、清晰的 Git 使用流程，是非常重要的。

否则，各种不清晰的分支结构，后续产品迭代或维护都会让人很头疼，再如果每个程序员都提交一堆杂乱无章的commit，后续的快速查找定位问题只能通过阅读代码，也是很低效的。

## 分支规范

---

几乎所有的版本控制系统都以某种形式支持分支。使用分支意味着你可以把你的工作从开发主线上分离开来，以免影响开发主线。有人把 Git 的分支模型称为它的“必杀技特性”，因为基于指针的实现使其足够轻量。

Git 鼓励在工作流程中频繁地使用分支与合并，哪怕一天之内进行许多次，但仍要遵循一定的规范

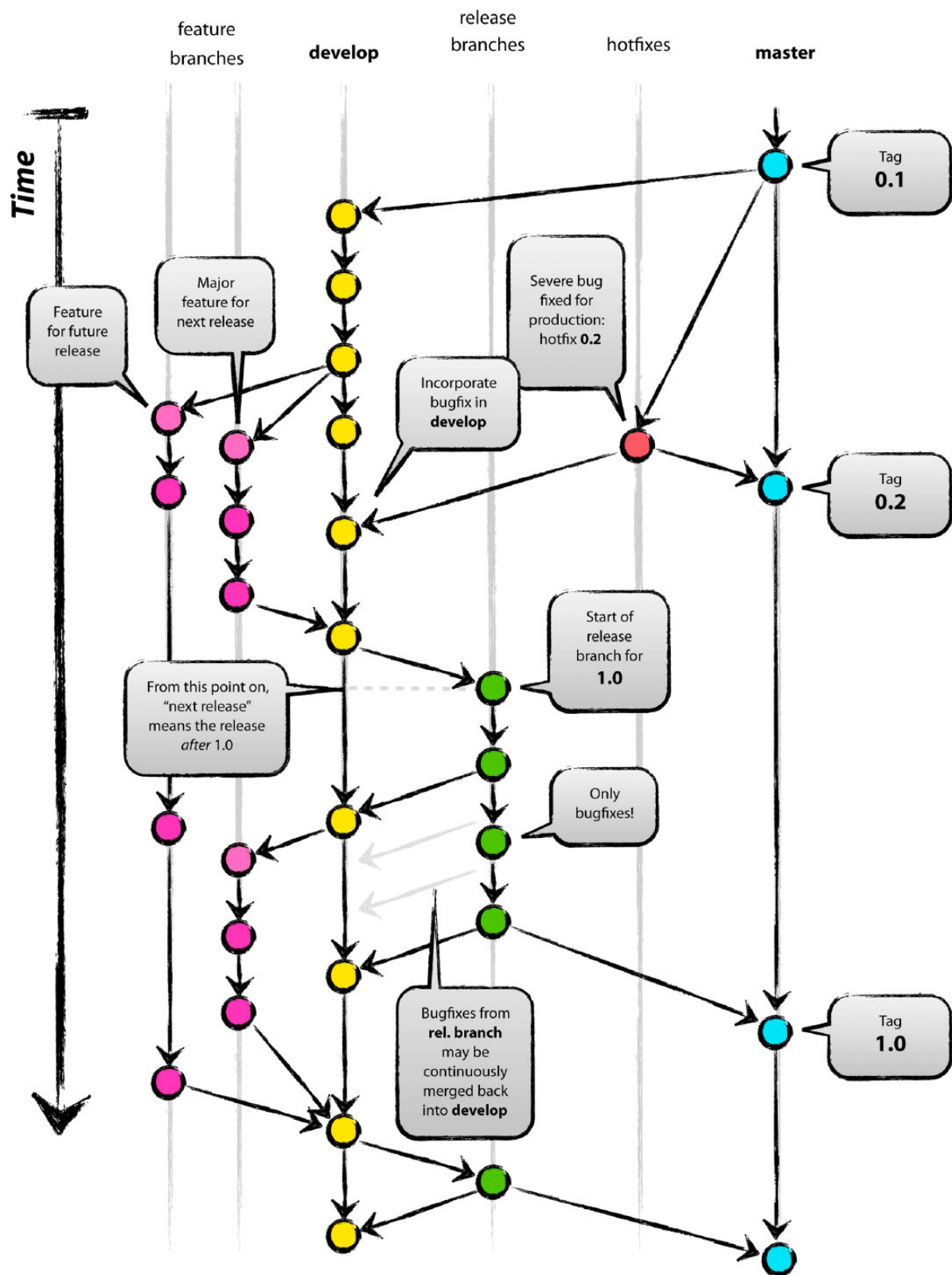
## 分支命名

- **master 分支**
  - master 为主分支，也是用于部署生产环境的分支，master 分支要确保稳定性
  - master 分支一般由 develop 以及 hotfix 分支合并，任何时间都不能直接修改代码
- **develop 分支**
  - develop 为开发分支，始终保持最新完成以及bug修复后的代码
  - 一般开发新功能时，feature 分支都是基于 develop 分支下创建的
- **feature 分支**
  - 开发新功能时，以 develop 分支为基础创建 feature 分支
  - 分支命名: feature/ 开头的为特性分支，命名规则: feature/user\_module、feature/cart\_module
- **release 分支**
  - release 为预上线分支，发布提测阶段，以 release 分支代码为基准提测
- **hotfix 分支**
  - 分支命名: hotfix/ 开头的为修复分支，它的命名规则与 feature 分支类似
  - 线上出现紧急问题时，需要及时修复，以 master 分支为基线，创建 hotfix 分支，修复完成后，需要合并到 master 分支和 develop 分支

当有一组 feature 开发完成，首先会合并到 develop 分支，进入提测时，会创建 release 分支。

如果测试过程中存在 bug 需要修复，则直接由开发者在 release 分支修复并提交。

当测试完成之后，合并 release 分支到 master 和 develop 分支，此时 master 为最新代码，用作上线。



以上规范不一定是必须的，一般是根据实际情况来的，但是要在上边的规范下实行

- 自己的分支一定要自测，切记不要提交后，影响到其他代码，更别说别人拉下代码还报错这种低级错误
- 本地分支要做到勤提交，分小功能提交，一次提交一大堆各种功能的做法也要杜绝
- 每天第一件事就是更新 **develop** 分支内容到本地分支，避免大规模 merge，太容易出错了
- 迭代新版本时，一定要保证当前开发分支和线上分支一样

## 提交规范

我们都知道，Git 每次提交代码，都要写 Commit message（提交说明），否则就不允许提交，这其实就是规范。

```
$ git commit -m "hello world"
```

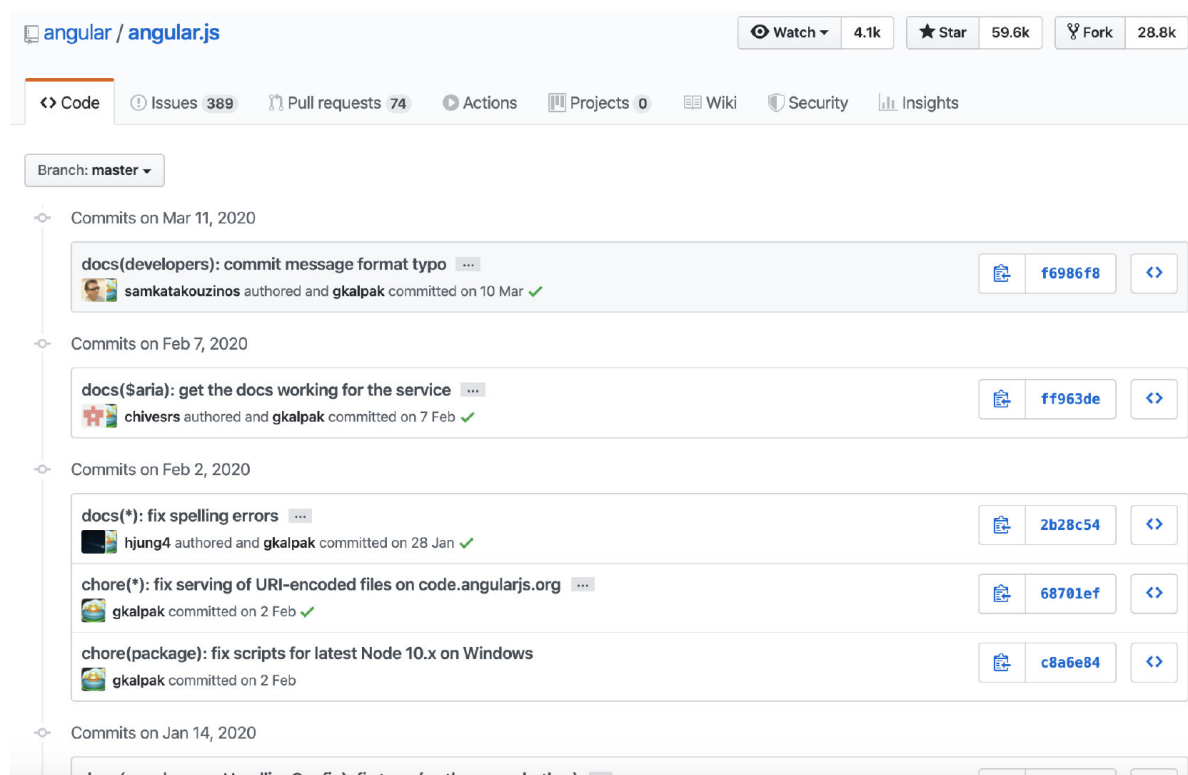
上面代码的 -m 参数，就是用来指定 commit message 的。

如果一行不够，可以只执行git commit，就会跳出文本编辑器，让你写多行。

一般来说，commit message 应该清晰明了，说明本次提交的目的。而且多人协作的时候，有问题也方便查看提交日志。

## 以下部分是大厂使用规范：（仅供参考）

目前，社区有多种 Commit message 的写法规范。来自Angular 规范是目前使用最广的写法，比较合理和系统化。如下图：



每次提交，Commit message 都包括三个部分：Header，Body 和 Footer。

```
<type>(<scope>): <subject>
// 空一行
<body>
// 空一行
<footer>
```

其中，Header 是必需的，Body 和 Footer 可以省略。

不管是哪一个部分，任何一行都不要有太多字符。这是为了避免自动换行影响美观。

## Head

Header部分只有一行，包括三个字段：`type`（必填）、`scope`（影响范围，选填）和 `subject`（必填）。

### type

`type` 用于说明 commit 的类别，只允许使用下面7个标识（或者用对应的 emoji 表情，在前边再加一个 `:` 就会显示了）。

- **feat**: 新功能 (🌟)
- **fix**: 修补bug (🐛)
- **docs**: 修改文档 (📖)
- **style**: 格式化代码结构，没有逻辑上的代码修改 (😄)
- **refactor**: 重构，即不是新增功能，也不是修改bug的代码变动，比如重命名变量 (🔧)
- **test**: 增加测试代码，单元测试一类的，没有生产代码的变更 (🧪)
- **chore**: 构建过程或辅助工具的变动（不会影响代码运行）

### scope

`scope` 用于定义 `type` 影响的范围，比如数据层、控制层、视图层等等，视项目不同而不同。

### subject

`subject` 是 commit 目的的简短描述，不超过50个字符。

## Body

Body 部分是对本次 commit 的详细描述，可以分成多行，每行尽量不超过72个字符。

## Footer

Footer 部分只用于两种情况

- **不兼容变动**: 如果当前代码与上一个版本不兼容，则 Footer 部分以 `BREAKING CHANGE` 开头，后面是对变动的描述、以及变动理由和迁移方法。
- **关闭 Issue**: 如果当前 commit 针对某个issue，那么可以在 Footer 部分关闭这个 issue 。

```
Closes #234
Closes #123, #245, #992
```

## Revert

还有一种特殊情况，如果当前 commit 用于撤销以前的 commit，则必须以 `revert:` 开头，后面跟着被撤销 Commit 的 Header。

```
revert: feat(pencil): add 'graphitewidth' option
This reverts commit 667ecc1654a317a13331b17617d973392f415f02.
```

这么多规范有什么用吗，如果项目中只有两三个人开发，其实也不需要严格的规范，只要把提交内容写清楚就行，但是大型项目，开发人员较多，规范提交还是有必要的

## 格式化的Commit message，有几个好处

- 提供更多的历史信息，方便快速浏览

比如，下面的命令显示上次发布后的变动，每个 commit 占据一行。你只看行首，就知道某次 commit 的目的。

```
$ git log <last tag> HEAD --pretty=format:%s
```

- 可以过滤某些commit（比如文档改动），便于快速查找信息

比如，下面的命令仅仅显示本次发布新增加的功能

```
$ git log <last release> HEAD --grep feature
```

- 可以直接从 commit 生成 Change log（Change Log 是发布新版本时，用来说明与上一个版本差异的文档）

最后列出一些 git 提交支持的 emoji 表情，就算是看GitHub 或 GitLab，也很有意思，也是目前大型项目中使用的方式。

/sBin/StyleGuide/Git/CommitMessage <http://slashsbin.com/styleguide-git-c...>

[emoji](#) [emojify](#) [style-guide](#) [gitmoji](#) [styleguide](#) [emojis](#) [commit](#) [commits](#)

55 commits

1 branch



0 packages















0 releases



























15 contributors






















View license

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

 slashsbin  Update license description ✓ Latest commit 644dead on 7 Oct 2018

 <a href="#">.gitignore</a>	 Add Gems to Build Jekyll Pages Locally	3 years ago
 <a href="#">CODE_OF_CONDUCT.md</a>	 Add Code of Conduct	2 years ago
 <a href="#">CONTRIBUTING.md</a>	 Add Contributing Readme	2 years ago
 <a href="#">Gemfile</a>	 Add Gems to Build Jekyll Pages Locally	3 years ago
 <a href="#">LICENSE</a>	 Update license description	2 years ago
 <a href="#">README.md</a>	 Remove GitHub Hit Count	2 years ago
 <a href="#">_config.yml</a>	 Added Jemoji gem for Github Pages build.	3 years ago

Emoji	Raw Emoji Code	Description
	<code>:art:</code>	when improving the <b>format</b> /structure of the code
	<code>:newspaper:</code>	when creating a <b>new file</b>
	<code>:pencil:</code>	when <b>performing minor changes/fixing</b> the code or language
	<code>:racehorse:</code>	when improving <b>performance</b>
	<code>:books:</code>	when writing <b>docs</b>
	<code>:bug:</code>	when reporting a <b>bug</b> , with <a href="#">@FIXME</a> Comment Tag
	<code>:ambulance:</code>	when fixing a <b>bug</b>
	<code>:penguin:</code>	when fixing something on <b>Linux</b>
	<code>:apple:</code>	when fixing something on <b>Mac OS</b>
	<code>:checkered_flag:</code>	when fixing something on <b>Windows</b>
	<code>:fire:</code>	when <b>removing code</b> or files, <i>maybe</i> with <code>@CHANGED</code> Comment Tag
	<code>:tractor:</code>	when <b>change file structure</b> . Usually together with 
	<code>:hammer:</code>	when <b>refactoring</b> code
	<code>:umbrella:</code>	when adding <b>tests</b>
	<code>:microscope:</code>	when adding <b>code coverage</b>
	<code>:green_heart:</code>	when fixing the <b>CI</b> build
	<code>:lock:</code>	when dealing with <b>security</b>
	<code>:arrow_up:</code>	when upgrading <b>dependencies</b>
	<code>:arrow_down:</code>	when downgrading <b>dependencies</b>
	<code>:fast_forward:</code>	when <b>forward-porting features</b> from an older version/branch
	<code>:rewind:</code>	when <b>backporting features</b> from a newer version/branch
	<code>:shirt:</code>	when removing <b>linter</b> /strict/deprecation warnings
	<code>:lipstick:</code>	when improving <b>UI</b> /Cosmetic
	<code>:wheelchair:</code>	when improving <b>accessibility</b>
	<code>:globe_with_meridians:</code>	when dealing with <b>globalization</b> /internationalization/i18n/g11n

Emoji	Raw Emoji Code	Description
	:construction:	<b>WIP</b> (Work In Progress) Commits, <i>maybe</i> with @REVIEW Comment Tag
	:gem:	New <b>Release</b>
	:egg:	New <b>Release</b> with Python egg
	:ferris_wheel:	New <b>Release</b> with Python wheel package
	:bookmark:	Version <b>Tags</b>
	:tada:	<b>Initial</b> Commit
	:speaker:	when Adding <b>Logging</b>
	:mute:	when Reducing <b>Logging</b>
	:sparkles:	when introducing <b>New</b> Features
	:zap:	when introducing <b>Backward-InCompatible</b> Features, <i>maybe</i> with @CHANGED Comment Tag
	:bulb:	New <b>Idea</b> , with @IDEA Comment Tag
	:snowflake:	changing <b>Configuration</b> , Usually together with 🏠 or 🐛 or 🚀
	:ribbon:	Customer requested application <b>Customization</b> , with @HACK Comment Tag
	:rocket:	Anything related to Deployments/ <b>DevOps</b>
	:elephant:	<b>PostgreSQL</b> Database specific (Migrations, Scripts, Extensions, ...)
	:dolphin:	<b>MySQL</b> Database specific (Migrations, Scripts, Extensions, ...)
	:leaves:	<b>MongoDB</b> Database specific (Migrations, Scripts, Extensions, ...)
	:bank:	<b>Generic Database</b> specific (Migrations, Scripts, Extensions, ...)
	:whale:	<b>Docker</b> Configuration
	:handshake:	when <b>Merge files</b>
	:cherries:	when Commit Arise from one or more <a href="#">Cherry-Pick</a> Commit(s)

项目使用过程中可以根据实际情况选择提交规范。

