

HenCoder Plus 第 29 课 讲义

Git 深入之三：其他常用指令之二和 Git Flow

add -i 交互式 add

用法：

```
git add -i
```

常用选项：

- p (patch)

选取块时的常用选项：

- y: 选用当前块
- n: 不用当前块
- s: 把当前块做自动切分后再重新询问
- e: 手动选取修改内容

tag

另一种引用类型。

- 和 branch 区别之一：不能改变
- 和 branch 区别之二：不能被 HEAD 指向
- 用处：设置持久标记，例如版本号
- origin/master, origin/feature, origin/HEAD 和 tag 有相似之处：也不能从本地改变位置，也不能被 HEAD 指向

reflog

用法：

```
git reflog <branch>
```

用途：

查看指定的引用（HEAD 或 branch）的移动历史，从而找到之前的某个特定 commit

cherry-pick

用法:

```
git cherry-pick commit1 commit2
```

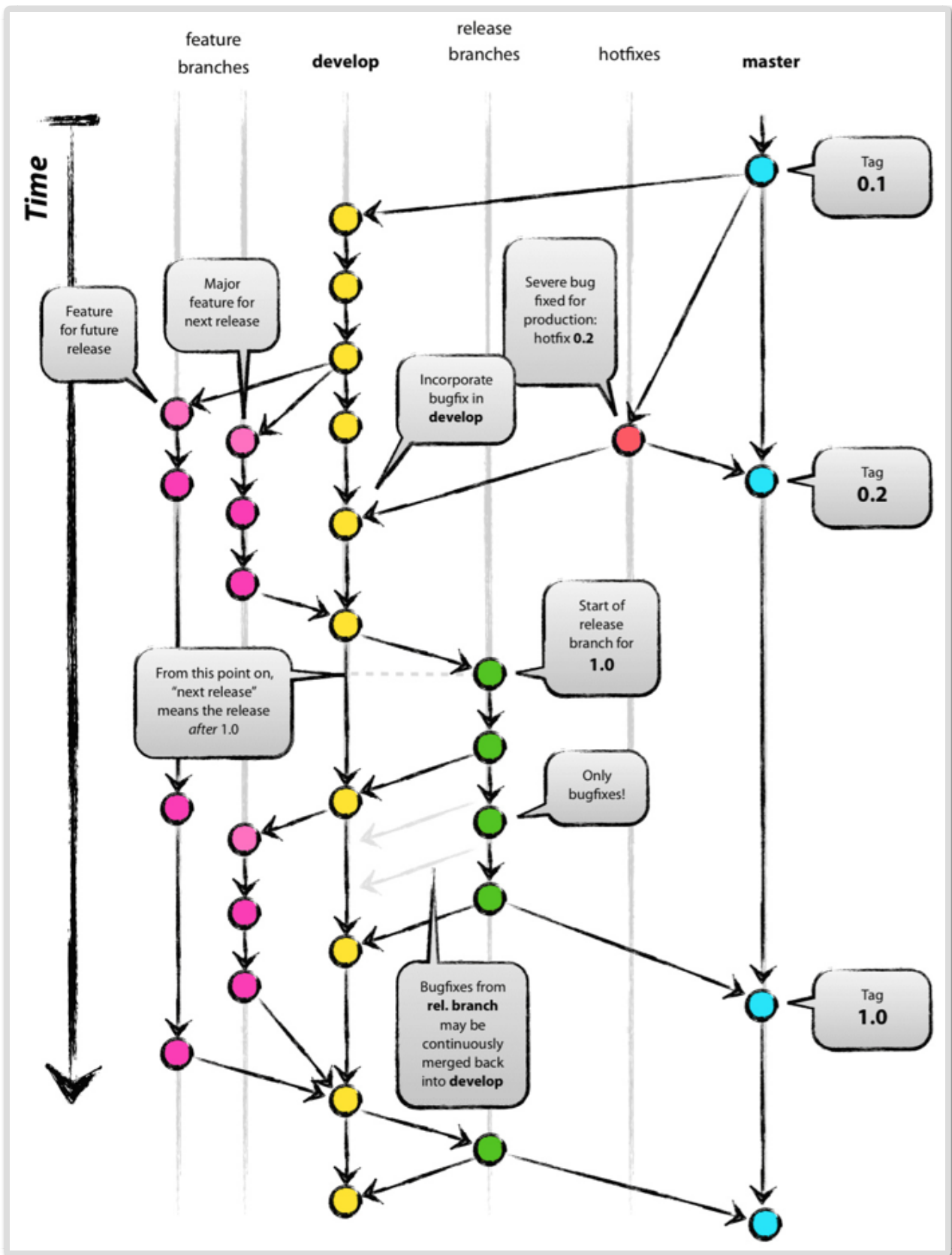
用途: 应用指定的某个或某些 commit 到 HEAD 处

- 比如我想抛弃某个 branch, 这个特性不要了, 但它的某两个 commit 我希望保留
- `git cherry-pick` 这两个 commit

github flow

- 可以有三种 merge 选项:
 1. 直接 merge
 2. branch 上所有 commit 压合成一个后再 merge
 3. branch 上所有 commit rebase 到 master 后再 commit
 - 作用? 保持一条线。
 - 为啥? 为了高质量。
- commit 的时候会自动应用 `--no-ff` 参数, 强行增加一个 merge commit, 用于增加记录信息
- GitHub 的 release 和 tag: tag 是 Git 的功能, Release 是 GitHub 利用 Git 的 tag 功能所提供的额外功能

git flow



两条持久主线：master 和 develop

- master
 - 只用于存放所有的发布
 - 每次有了新的 commit 之后，立即打一个 tag 来记录
- develop
 - 用于存放不稳定版本的发布（例如每夜版）

- develop 并不是直接用于开发 feature 的, 开发 feature 需要专门的 branch
- develop 在第一时间从 master 上分离出来
- 需要开发任何功能的时候, 从 develop 创建出新的 feature branch, 开发完成后合并回 develop (合并的时候使用 --no-ff), 然后删掉 feature branch
- 当下一正式版本需要的所有功能开发完成之后, 从 develop 上创建新的 release branch, 并在 release branch 合并到 master 后合并回 develop (合并的时候用 --no-ff), 然后删掉 release branch
- feature branches
 - 每次开发新功能是从 develop 创建
 - 开发完成后合并到 develop (使用 --no-ff), 然后被删掉
- release branches
 - 每次下一版本的功能开发完毕后, 从 develop 上创建
 - 创建完成后, 更新版本号, 然后单独做一个新的 commit
 - 如果有 bug fix, 直接在 release branch 上创建
 - bug fix 完成后, 合并到 master 和 develop (使用 --no-ff), 然后被删掉
- hotfix branches
 - 已正式发布的产品发现 bug, 直接从 master 或者出问题的 tag 上创建 hotfix branch, 进行紧急修复, 修复完成后合并到 master 和 develop (或 release branch 如果有的话) (使用 --no-ff), 然后被删掉