1.如何删除不需要的分支?

`git branch -d [分支名称]`；`git branch -D [分支名称]`

```
DCdeMacBook-Air:git_learning One_Piece$ git branch -d fix_css
error: 分支 'fix_css' 没有完全合并。
如果您确认要删除它，执行 'git branch -D fix_css'。
DCdeMacBook-Air:git_learning One_Piece$ git branch -D fix_css
已删除分支 fix_css（曾为 8915f95）。
```

2.怎么修改commit的message?

- 最近最新的一次提交 `git log -1` , **`git commit --amend`**
- 老旧commit的message

```
DCdeMacBook-Air:git_learning One_Piece$ git log -3 --oneline
1c47de6 (HEAD -> master) Move filename readme to readme.md
93a8a55 Add refering projects
a5cdf21 Add js
```

  - 需求：将message 'Add refering projects'变为'Add a refering project'
  - 此message作为commit数据结构当中的一个属性 对message进行了变更，其commit的id号肯定会变
  - <rebase> 变基 '基'要选择被变commit的父级
    - `git rebase -i a5cdf21` i 交互式的变更
    - pick 93a8a55 Add refering projects 改变其策略为 r 93a8a55 Add refering projects
    - `r` commit变更文件的内容是不动的 只改变message

```
DCdeMacBook-Air:git_learning One_Piece$ git rebase -i a5cdf21
[分离头指针 00f3add] Add a refering project
 Date: Thu Sep 9 10:55:08 2021 +0800
 2 files changed, 2 insertions(+)
成功变基并更新 refs/heads/master。

DCdeMacBook-Air:git_learning One_Piece$ git log -n3 --oneline --
graph
* 0787166 (HEAD -> master) Move filename readme to readme.md
* 00f3add Add a refering project
* a5cdf21 Add js
```

  - 注意：变基操作是离不开分离头指针的. 仔细观察 master(因为其parent父级变了)和我们需要变更message的commit，它们的ID都变了。
  - **特别注意**：此变基行为仅适用于在自己的分支上面做变更，还没有贡献到团队的集成分支上时。 假如已经贡献到集成分支上,就不能这样轻易的变基了！！现目前的操作，都是我们在维护自己的分支，还没有分享出去。

3.把连续的多个commit整理成1个

```
DCdeMacBook-Air:git_learning One_Piece$ git log --graph --oneline
* 0787166 (HEAD -> master) Move filename readme to readme.md
* 00f3add Add a refering project
* a5cdf21 Add js
* 0445492 Add style
* 47183d8 Add index + logo
* 73ffe70 Add readme
```

- 我们想将中间四个commit变更为一个。`git rebase -i 73ffe70`
  - `s` 合并到前一个提交

```
pick 47183d8 Add index + logo
s 0445492 Add style
s a5cdf21 Add js
s 00f3add Add a refering project
pick 0787166 Move filename readme to readme.md
```

```
# 这是一个 4 个提交的组合。
Create a complete web page
# 这是第一个提交说明:
Add index + logo
# 这是提交说明 #2:
Add style
# 这是提交说明 #3:
Add js
# 这是提交说明 #4:
Add a refering project
```

```
DCdeMacBook-Air:git_learning One_Piece$ git rebase -i 73ffe70
[分离头指针 b638011] Create a complete web page Add index + logo
Add style Add js Add a refering project
 Date: Thu Sep 9 10:45:29 2021 +0800
 4 files changed, 116 insertions(+)
 create mode 100644 images/git-logo.png
 create mode 100644 index.html
 create mode 100644 js/script.js
 create mode 100644 styles/style.css
成功变基并更新 refs/heads/master。
```

  - 注意观察master分支和全部分支

```
DCdeMacBook-Air:git_learning One_Piece$ git log --graph --
oneline
* b387b6e (HEAD -> master) Move filename readme to readme.md
* b638011 Create a complete web page Add index + logo Add style
Add js Add a refering project
* 73ffe70 Add readme
DCdeMacBook-Air:git_learning One_Piece$ git log --graph --all --
oneline
* b387b6e (HEAD -> master) Move filename readme to readme.md
* b638011 Create a complete web page Add index + logo Add style
Add js Add a refering project
| * 8f89a6d (temp) Add test
| * a5cdf21 Add js
| * 0445492 Add style
| * 47183d8 Add index + logo
|/
* 73ffe70 Add readme
```

4.把间隔的几个commit整理成1个

```
DCdeMacBook-Air:git_learning One_Piece$ git branch -av
* master b387b6e Move filename readme to readme.md
  temp   8f89a6d Add test
DCdeMacBook-Air:git_learning One_Piece$ git log --oneline
b387b6e (HEAD -> master) Move filename readme to readme.md
b638011 Create a complete web page Add index + logo Add style Add
js Add a refering project
73ffe70 Add readme
```

`git rebase -i 73ffe70`

- 可以发现master分支第一个和第三个commit都是关于readme文件的，我们需要将它两进行合并。
- 因为73ffe70commit已经是老祖宗了。将最古老的这一个commit添加到最前面。`pick 73ffe70`
  - 注意其顺序 73ffe70pick不变；将b387b6e合并到73ffe70中(**合并到旧的commit中**)；b638011是一个完整的功能pick不变。

```
pick 73ffe70
s b387b6e Move filename readme to readme.md
pick b638011 Create a complete web page Add index + logo Add
style Add js Add a refering project
```

```
git status
git rebase --continue
```

```
# 这是一个 2 个提交的组合。
Add readme.md
# 这是第一个提交说明:
Add readme
# 这是提交说明 #2:
Move filename readme to readme.md
```

  - 一通操作下来，现在master分支有且仅有两个commit。一个是关于添加readme.md文件的，一个是关于静态页面的。

```
DCdeMacBook-Air:git_learning One_Piece$ git log --graph --
oneline
* 016b653 (HEAD -> master) Create a complete web page Add index
+ logo Add style Add js Add a refering project
* 82cc036 Add readme.md Add readme Move filename readme to
readme.md
```

  - 所有分支,你会惊奇的发现。此时出现了两颗独立的树！

5.如何比较暂存区和HEAD所含文件的差异?

养成一个好习惯：在暂存区的东西是否可以作为正式的提交？用diff命令把暂存区变更的文件跟我们当前分支最近的一次commit做一个比较，看看两者的差异。

HEAD指向的是master分支最新的提交。

!!!!未执行git add之前，暂存区的内容是HEAD对应的内容或分支最新commit的内容。

```
# 当前处于master分支
vi index.html 更改内容 <li>add</li>
git add index.html
git status
# 比较暂存区与HEAD之间的差异 --cached! !
git diff --cached

# 重新更改内容
vi index.html 更改内容 <li>config</li>
git add index.html
git diff --cached
git commit -m'Add the first git command with config'
```

注意：`git diff --staged`等同于`git diff --cached`;命令后面可以指定文件名，不加就是比对所有文件差异。

6.如何比较工作区和暂存区所含文件的差异?

git diff 默认比较的就是工作区和暂存区的区别。

```
# 暂存区
vi index.html 更改另一个li标签 <li>bare repository</li>
git add index.html

# 工作区
vi styles/style.css 更改字体颜色白色变为黑色
vi readme.md 添加一行内容'say hello'

# 比较工作区与暂存区
# 所有文件
git diff
# 指定文件
git diff style/style.css
git diff readme.md
# 多个文件
git diff style/style.css readme.md
```

7.如何让暂存区恢复成HEAD的一样?

```
git status
# 有点将暂存区清空的意思
git reset HEAD
git status
git diff --cached
```

- PS: 取消暂存区部分文件的变更? `git reset HEAD [filename1] [filename2]`

8.如何让工作区的文件恢复为和暂存区一样?

需求：做了一些变更并放到了暂存区,接着在工作区继续做变更,但发现工作区的变更还不及暂存区的好,这时候我们希望工作区变为跟暂存区一样的状态。

```
DCdeMacBook-Air:git_learning One_Piece$ git status
位于分支 master
尚未暂存以备提交的变更:
   (使用 "git add <文件>..." 更新要提交的内容)
   (使用 "git restore <文件>..." 丢弃工作区的改动)
     修改:       index.html
     修改:       readme.md
     修改:       styles/style.css

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）

DCdeMacBook-Air:git_learning One_Piece$ git add index.html
DCdeMacBook-Air:git_learning One_Piece$ git diff --cached
diff --git a/index.html b/index.html
index cab702c..2a1ce64 100644
--- a/index.html
+++ b/index.html
@@ -30,7 +30,7 @@
          <div class="accordion"><h1>Basic Commands</h1></div>
              <div class="panel">
                  <ol>
-                     <li></li>
+                     <li>bare repository</li>
                      <li></li>
                      <li></li>
                      <li></li>
```

```
vi index.html现在继续在工作区更改index.html文件.将bare repository改为裸仓库。
# 将暂存区的文件拷贝到工作区
git checkout index.html
```

9.消除最近的几次提交?

```
DCdeMacBook-Air:git_learning One_Piece$ git log --graph --oneline
* 8f89a6d (HEAD -> temp) Add test
* a5cdf21 Add js
* 0445492 Add style
* 47183d8 Add index + logo
* 73ffe70 Add readme
DCdeMacBook-Air:git_learning One_Piece$ git reset --hard 0445492
HEAD 现在位于 0445492 Add style
DCdeMacBook-Air:git_learning One_Piece$ git log --graph --oneline
* 0445492 (HEAD -> temp) Add style
* 47183d8 Add index + logo
* 73ffe70 Add readme
```

理解下 `git reset --soft 47183d8`

```
DCdeMacBook-Air:git_learning One_Piece$ git log --graph --oneline
* 0445492 (HEAD -> temp) Add style
* 47183d8 Add index + logo
* 73ffe70 Add readme
DCdeMacBook-Air:git_learning One_Piece$ git status
位于分支 temp
无文件要提交，干净的工作区
DCdeMacBook-Air:git_learning One_Piece$ git reset --soft 47183d8
DCdeMacBook-Air:git_learning One_Piece$ git status
位于分支 temp
要提交的变更：
  （使用 "git restore --staged <文件>..." 以取消暂存）
    新文件：    styles/style.css
```

10.看看不同提交的指定文件的差异

理解：分支无外乎就是指针，指针就是指向分支的最近的一个commit。

```
git branch -av
git diff 47183d8 47d01ac -- index.html
git diff temp master -- index.html
```

11.开发中临时加塞了紧急任务怎么处理?

应用场景：一部分文件已经放置暂存区，一部分文件还在工作区修改当中，我们正在为某一个功能做开发。突然，测试说对应的分支有bug，要临时的修复线上的bug。

解决方案：将手头的工作先放到某个区域里存起来，修复bug生成新的commit提交后再把以前的工作恢复。

```
DCdeMacBook-Air:git_learning One_Piece$ git status
位于分支 temp
尚未暂存以备提交的变更:
   (使用 "git add <文件>..." 更新要提交的内容)
   (使用 "git restore <文件>..." 丢弃工作区的改动)
    修改:       index.html

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
DCdeMacBook-Air:git_learning One_Piece$ git stash
保存工作目录和索引状态 WIP on temp: 47183d8 Add index + logo
DCdeMacBook-Air:git_learning One_Piece$ git stash list
stash@{0}: WIP on temp: 47183d8 Add index + logo
DCdeMacBook-Air:git_learning One_Piece$ git status
位于分支 temp
无文件要提交，干净的工作区
```

```
# 作用1: 将区域中的变更放到工作区去
# 作用2: stash列表里这个栈还存在 不会被删除 可以反复使用
git stash apply

# 列表中的会丢失
git stash pop
```

12.如何指定不需要Git管理的文件

.gitignore

- *.d 以.d结尾的文件不纳入
- *.dSYM/ 文件夹下的任何文件都不纳入
  - 若有一个a.dSYM文件，git是会管控的
  - 若有名为*.dSYM的文件夹,不管控
- doc 不管是文件还是文件夹都不管控

13.git的备份

```
DCdeMacBook-Air:git_learning One_Piece$ git clone --bare
/Users/One_Piece/Desktop/git/git_learning/.git ya.git
克隆到纯仓库 'ya.git'...
完成。
DCdeMacBook-Air:git_learning One_Piece$ git clone --bare
file:///Users/One_Piece/Desktop/git/git_learning/.git zhineng.git
克隆到纯仓库 'zhineng.git'...
remote: 枚举对象中: 20, 完成.
remote: 对象计数中: 100% (20/20), 完成.
remote: 压缩对象中: 100% (15/15), 完成.
remote: 总共 20 (差异 3)，复用 0 (差异 0)，包复用 0
接收对象中: 100% (20/20), 21.54 KiB | 7.18 MiB/s, 完成.
处理 delta 中: 100% (3/3), 完成.
DCdeMacBook-Air:git_learning One_Piece$ ls
doc     index.html  ya.git
images      readme      zhineng.git
```

总结

- 删除某一分支：`git branch -D [分支名称]`
- 修改commit的message：
  - 最新 `git commit --amend`
  - 以前 `git rebase -i [变更提交的父级]` 交互式
    - 修改目标commit的策略 pick -> r
- 将连续多个commit合并为一个:
  - `git rebase -i [连续commit最老的一个的父级]`
    - 策略改变 连续的提交最老的那个为pick 其余连续的为s

- 把间隔的几个commit整理成一个：
  - 若最老祖宗commit'73ffe70'在需要整理的提交中 `git rebase -i 73ffe70`
    - 最前面添加 pick 73ffe70
    - 动态灵活的更改commit的顺序以及其策略为s
    - `git status` ; `git rebase --continue`
- 比较暂存区和HEAD
  - `git diff --cached`
  - 应用场景：工作区更改一些文件后,将它们add到暂存区,在正式提交之前,与HEAD进行比较,看做了哪些更改。
  - **简单理解**：暂存区里文件的更改跟最新一次commit有何差异。
- 比较工作区和暂存区
  - `git diff` ; `git diff [文件1] [文件2]`
  - **简单理解**：还有哪些工作区的文件变更没有加到暂存区中。
- 如何让暂存区恢复成HEAD一样?
  - 通常情况下,如果感觉暂存区的修改没有工作区的好,可以直接将工作区的内容添加到暂存区,那么暂存区会以新添加的文件状态为准。
  - 有这样一个应用场景：已经确定不想保留暂存区的变更,其次工作区还没有改好,想将暂存区的所有变更撤销恢复到HEAD.
    - `git reset HEAD` 暂存区变得跟HEAD一样,工作区文件不变
    - `git reset --hard HEAD` 暂存区和工作区皆变得跟HEAD一样
      - 你会发现 -> 执行 `git diff` 、 `git diff --cached` 皆为空!
  - PS: 怎么取消暂存区部分文件的变更? `git reset HEAD [filename1] [filename2]`
- 如何让工作区的文件恢复为和暂存区一样?
  - 应用场景：工作区修改了文件A,将其添加到暂存区。继续对文件A进行修改,结果不尽人意,需恢复成暂存区的文件A一样的内容。
  - `git checkout [文件A]`
    - 暂存区的文件A依旧存在, `git diff` 为空
- 消除最近几次的提交.
  - `git reset --hard 0445492`
  - 当前分支的头指针指向0445492这个commit,同时暂存区和工作区的内容都要回到跟这个commit一摸一样的状态。
  - 树中0445492后面的几次commit全部丢失！可怕至极。
  - 补充：**reset --soft** 保留工作目录,并把重置HEAD所带来的新的差异放进暂存区。

- 对不同的分支或commit进行差异化比较
  - `git diff commit1_id commit2_id -- filename_path`
- 正确删除文件的方法 `git rm [filename]`
- 开发中临时加塞了紧急任务怎么处理？（先了解stash命令）
  - `git stash`
  - `git stash pop` 或者 `git stash apply`
- .gitignore！！！
- git的备份. 了解哑协议和智能协议。
- 一些常见的linux命令
  - mkdir doc
  - echo 'hi' > doc/readhim
  - rm -rf doc
  - echo 'I am a file' > doc