track跟踪; remote远程; branch分支; config配置; origin起源; clone克隆;

1.不同人修改了不同文件如何处理?

应用场景：两个人维护同一个分支。即有两个git的客户端都工作在相同的分支上面。现在，在同一分支中，两个人修改了不同的文件。如何保证协作的顺畅?

码农A:修改readme.md文件 并push

- 在GitHub的git_learning项目中基于main创建分支`feature/add_git_commands`.
- 将git_learning项目主分支main通过SSH协议从Git服务器克隆到本地!
  - `git clone git@github.com:OnePieceDC/git_learning.git git_learning_02`
  - 若不写git_learning_02,它会在本地自动创建文件夹git_learning,但如果在当前目录有同名的文件夹,会报错说已经存在。
  - 解释：git clone命令会我们拉取所有数据,并为每个分支创建远程跟踪分支（可使用`git branch -r`命令查看）
  - 补充：
    - `git branch -v` 本地所有分支;
    - `git branch -r` 远程所有分支;
    - `git branch -av` 本地和远程所有分支;

```
DCdeMacBook-Air:git_learning_02 One_Piece$ git branch -r
origin/HEAD -> origin/main
origin/feature/add_git_commands
origin/main
origin/master
```

- 进入git_learning_02 `cd git_learning_02`
- 针对本仓库，设置其用户名和邮箱(不加--add 多次设置会覆盖)
  - `git config --add --local user.name 'OnePieceDC'`
  - `git config --add --local user.email '1415806497@qq.com'`
  - `git config --local -l`
  - 若没有达到预期的效果 `vi .git/config` 进行修改!
- 基于远端的origin/feature/add_git_commands分支创建本地分支(跟远端命名一样)

    - 注意: 我们在本地无法直接在clone下来的远程分支上做变更,只能基于远程分支创建本地分支后,才能创建commit.
    - `git_learning_02 One_Piece$ git checkout -b feature/add_git_commands origin/feature/add_git_commands`
    - 基于远端的特定分支创建出一条本地的分支同时还会切换到这条分支上面去
    - set up to track remote branch将本地分支跟远端分支联系起来了.所以后面直接`git push`很顺利。

```
DCdeMacBook-Air:git_learning_02 One_Piece$ git checkout -b
feature/add_git_commands origin/feature/add_git_commands
分支 'feature/add_git_commands' 设置为跟踪来自 'origin' 的远程分支
'feature/add_git_commands'。
切换到一个新分支 'feature/add_git_commands'
```

- 修改readme文件
    - `vi readme.md`添加'We are going to record some git commands here.'
- 添加暂存区并提交变更.
    - `git add -u`
    - `git status`
    - `git commit -m'Add git commands description in readme'`
- 当本地有了变更后,养成好习惯:向远端的服务器做更新,这样相当于在远端做了一个备份,同时也可以让团队的其他成员及时的拉取到我们的变更以便做集成开发。
    - 因为缺省的remote就是origin. 所以直接`git push`
    - 等同于 git push <远程主机名> <本地分支名>:<远程分支名>
    - git push origin feature/add_git_commands:feature/add_git_commands
    - 打开GitHub网页,进入此分支,你会发现readme.md文件已经改变。
- 补充！！这个很重要！fast-forward什么意思?
    - 举个栗子:本地分支往远端分支做push操作,如果远端分支不是这个本地分支的祖先,那它两就不是fast- forward关系。反之异然。
    - 若远端分支不是本地分支的祖先有什么影响？本地分支直接push到对应的远端分支，报非fast-forward的错误！
    - 我们把fast-forward通俗地说：commit和commit之间，子commit有一个箭头指向父commit，用这种方式，版本演变的历史就会形成一幅带方向的图。把commit比做人的话，孙子和爷爷之间，儿子和爸爸之间就是fastforward的关系，而堂兄之间就不是fastforward的关系了。

- 官网解释：fast-forward是merge的一种特殊类型，你把A分支合入到B分支的时候，恰好B分支指向的commit是A分支的祖先，在这种情况下，我们无需额外创建一个merge的commit，而只需把B分支指向A分支对应的commit就行。

码农B: 修改index.html文件 暂停push

- 进入前面学习建立的git_learning仓库
- 查看此仓库的用户名: `git config --local -l`
  - 其用户名为'suling',邮箱为'suling@163.com'
  - 所以基于这个仓库产生的author应该是'suling'
- `git fetch github`
  - 在前面一大章已经添加了名为github的远程仓库地址,回顾下:
    - `git remote add github git@github.com:OnePieceDC/git_learning.git`
    - 查看现有的远程仓库地址 `git remote -v`
  - 这里的操作有些许不同.码农A操作的那个仓库是新建克隆下来的,直接将远端的所有分支拉到本地了.码农B操作的仓库里并没有远端的fecture/add_git_commands分支.所有需要先将远程主机的最新内容拉取到本地。
  - 再简单解释：fetch只是把本地的远端分支和服务器对应的远端分支保持一致!
  - `git branch -av` 会发现本地多了fecture的远程分支

```
DCdeMacBook-Air:git_learning One_Piece$ git branch -r
github/feature/add_git_commands
github/main
github/master
github/temp
```

- 同样的,创建一个本地分支(建议名字与远程分支保持一致 当然不一样也可以)
  - `git checkout -b feature/add_git_commands github/feature/add_git_commands`
  - 注意比较：码农A的操作是对缺省默认的origin操作,而码农B是对我们自定义命名的github操作。
  - 你会发现,本地和远端的feature分支的hash值都是一样的,因而它们指向同一个东西。

- 修改index.html文件
  - `vi index.html` 添加 `<li>add</li>`
- 将变更的文件添加到暂存区并创建一个新的commit
  - `git add -u`
  - `git commit -m'Add git add command in index'`
  - 这时候查看 `gitk --all` 你会发现本地跟远程的分支是fastforward的关系! 意味着push会很顺利！！
- 走到这一步,暂停,不急着将其push到远端。这时候,码农A在远端分支上修改些东西后,码农B再执行push操作,看会出现什么问题!

码农A: 第二次 继续修改readme.md文件 并push

- `cd git_learning_02`; `vi readme.md` 末尾添加'eg add and so on.'
- `git commit -am 'Fix readme'`
- `git push`

码农B: 开始push

- `git status`;`git push github`
  - 看报错->拒绝的原因：本地跟远端的分支不是fast-forwards。

```
DCdeMacBook-Air:git_learning One_Piece$ git status
位于分支 feature/add_git_commands
您的分支领先 'github/feature/add_git_commands' 共 1 个提交。
  （使用 "git push" 来发布您的本地提交）

无文件要提交，干净的工作区
DCdeMacBook-Air:git_learning One_Piece$ git push
To github.com:OnePieceDC/git_learning.git
 ! [rejected]        feature/add_git_commands ->
feature/add_git_commands (fetch first)
error: 推送一些引用到 'github.com:OnePieceDC/git_learning.git' 失败
提示：更新被拒绝，因为远程仓库包含您本地尚不存在的提交。这通常是因为另外
提示：一个仓库已向该引用进行了推送。再次推送前，您可能需要先整合远程变更
提示：（如 'git pull ...'）。
提示：详见 'git push --help' 中的 'Note about fast-forwards' 小节。
```

- `git fetch github`
  - 执行完后看最后一行 在本地更新远端分支的hash值 从91dd152变成了157a593

```
91dd152..157a593  feature/add_git_commands ->
github/feature/add_git_commands
```

- `git branch -av`
  - 执行完后,细品这行反馈 [ahead 1,behind 1]
  - 意思是:本地有一个commit是比远端新的,远端有一个commit本地没有。
  - 解决方案：如果团队不要求线性的分支模式的话,一般会采用merge命令将本地跟远端的进行合并
    - 这里的合并会很顺利 因为码农A码农B修改的是不同文件
    - 提一句,gitk --all 你会发现此commit会有两个父类
    - `git merge github/feature/add_git_commands` 修改后本地和远端分支的hash值都变成了860b33f,也就具备了fast-forwards的关系。
    - `git push github`

```
* feature/add_git_commands                  8cec038 [领先 1，落后
1] Add git add command in index
```

没错 这是分割线～

2.不同人修改了同文件的不同区域如何处理?

应用场景：两个人开发同一个分支,修改了相同文件的不同区域。同样的,码农A OnePieceDC同学 -> git_learning_02 ;码农B suling同学-> git_learning.

- 在开发之前,养成好习惯: 将本地环境跟远端做一个同步.`git pull`
  - 这里要做一个说明,同步后,AB码农本地和远端的feature分支的Hash值都变成了860b33f
  - 当本地分支有文件变更,本地分支hash会立刻发生变化,远程分支的不变
  - push成功后 本地和远程的hash会相同
- 修改index.html文件的不同区域
  - 码农A
    - `vi index.html` 第一个无序列表添加 `<li>non fast-forward</li>`
    - `git commit -am 'Add non fast-forward'`
  - 码农B
    - `vi index.html` 第二个无序列表添加 `<li>commit</li>`
    - `git commit -am 'Add commit command'`
  - 所以这样的操作 基于远端的分支做了两个彼此之间不具备'fast-forwards'的commit。
- 码农B先push到远端 `git push github` 成功！hash值-44bc3a5
- 码农A执行`git push` 被无情拒绝！报错: 'Note about fast-forwards'
  - 解决方案1： `git pull` 将远端的commit做一个变更并与本地的做一个marge
  - 解决方案2：
    - `git fetch` 将远端的commit做一个变更
    - `git merge 44bc3a5` 等同于 `git merge origin/feature/add_git_commands`
    - 验证 `cat index.html`
    - 说明: merge操作将本地与远程分支合并 那么新的commit会有两个parent父亲。在这里,一个是码农B suling同学刚成功push发布的commit'Add commit command';另一个就是码农A OnePieceDC同学在本地的分支'Add non fast-forward'
  - `git push` 这一操作后,远端分支和码农A刚发布的这一commit的Hash相同.

码农A(OnePieceDC)	本地A	本地远端	码农B(suling)	本地B	本地远端
–	35dcc99	35dcc99	–	35dcc99	35dcc99
vi index.html	35dcc99	35dcc99	vi index.html	35dcc99	35dcc99
git add -u	35dcc99	35dcc99	git add -u	35dcc99	35dcc99
git commit -m 'warehouse'	84c813a [领先 1]	35dcc99	git commit -m'status'	0e75f71 [领先 1]	35dcc99
–	84c813a [领先 1]	35dcc99	git push github	0e75f71	0e75f71
git fetch	84c813a [领先 1，落后 1]	0e75f71	–	0e75f71	0e75f71
git merge 0e75f71	5836caa [领先 2]	0e75f71	–	0e75f71	0e75f71
git push	5836caa	5836caa	–	0e75f71	0e75f71
git pull	5836caa	5836caa	git pull	5836caa	5836caa

没错 这是分割线～

3.不同人修改了同文件的同一区域如何处理?

应用场景：两个人修改了同一个文件的同一个区域,其中一个人先提交,另一个将远端拉到本地的时候,出现了冲突.

- 同样的,码农A码农B先`git pull`
- 码农A:
  - 修改index.html 第二个无序列表第5第6位 添加内容stash和log
  - `git commit –am'Add stash and log commands'`
  - `git push`
- 码农B:
  - 修改index.html 第二个无序列表第5第6位 添加内容mv和rm
  - `git commit –am'Add mv and rm commands'`
  - `git pull` 会先成功fetch,然后自动marge合并的时候,合并失败。

```
自动合并 index.html
冲突（内容）：合并冲突于 index.html
自动合并失败，修正冲突然后提交修正的结果。
```

  - vi index.html
    - 有两个约束(marge时产生的提示信息) 一个是远端的,一个是本地当前工作区的分支
    - 解决: 商量后决定四个都保留(调整下顺序都是可以的),同时删除约束的符号。

```
                <li>config</li>
                <li>add</li>
                <li>commit</li>
                <li>status</li>
<<<<<<< HEAD
                <li>mv</li>
                <li>rm</li>
=======
                <li>stash</li>
                <li>log</li>
>>>>>>> e5d6926e0a339acfdf191f067dffa4b5214ab29f
                <li></li>
```

- 运行 `git status` 查看marge是否完成
  - 结果显示没有,还有Unmerged path.
  - 若不想让两个分支进行merge了. `git merge --abort` 恢复merge之前的状态
  - 这里merge结果是需要保留的.所以根据提示信息生成一个新的commit.

```
DCdeMacBook-Air:git_learning One_Piece$ git status
位于分支 feature/add_git_commands
您的分支和 'github/feature/add_git_commands' 出现了偏离,
并且分别有 1 和 1 处不同的提交。
  (使用 "git pull" 来合并远程分支)

您有尚未合并的路径。
  (解决冲突并运行 "git commit")
  (使用 "git merge --abort" 终止合并)

未合并的路径:
  (使用 "git add <文件>..." 标记解决方案)
    双方修改:    index.html

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

- `git commit -am 'Resolved conflict by hand with 4 git commands'`
- `git status`
  - 此时工作路径是干净的,没有什么需要commit的。
  - 证明解决了冲突并且生成了一个新的commit
- `git push github`

没错 这是分割线～

4.同时变更了文件名和文件内容如何处理?

应用场景: 多人协作下,有人将一个文件的名字变更了,其他人不晓得,还是基于原来的文件名将内容做了变更。

- 同样的,先同步更新
- 码农B(suling)
  - `ls -al` ; `git mv index.html index.htm`
  - `git status` ; `git commit -am'Mv index.html to index.htm'`
  - 先提交变更 `git push github`
- 码农A(OnePieceDC)
  - `vi index.html` 简单修改内容
  - `git commit -am 'Improve index head'`
  - 这时git push会提示'Note about fast-forwards'
    - 解决方案: `git pull` 智能的处理！！
    - 会有神奇的事情发生,git能够感知到文件名的变化,然后不受到影响,而且文件内容的变更顺理成章的添加进去了。自动将html -> htm。
  - `git push`

没错 这是分割线～

5.把同一文件改成了不同文件名如何处理?

应用场景: 多人修改了同一文件的文件名,有人先提交有人后提交。

- 同样的,先同步更新
- 码农B(suling)
  - `git mv index.htm index1.htm`
  - `git commit -am'Mv index to index1'`
  - 先提交上去了 `git push github`
- 码农A(OnePieceDC)
  - `git mv index.htm index1.htm`
  - `git commit -am'Mv index to index1'`
  - `git push` 报错'Note about fast-forwards'
    - `git pull` 报错,并且 `ls -al` 出现了index1.htm和index2.htm两个文件
    - `diff index1.htm index2.htm` 表明两文件的内容是一样的

```
冲突（重命名/重命名）：在分支 "HEAD" 中重命名 "index.htm"->"index1.htm"，在分支 "de709cd050c75fc72cfe43e526ef5ee0bb47ee9d" 中重命名 "index.htm"->"index2.htm"
自动合并失败，修正冲突然后提交修正的结果。
```

  - `git status` 商量后决定保留index1.htm
    - `git rm index.htm`
    - `git rm index2.htm`
    - `git add index1.htm`

```
未合并的路径：
（酌情使用 "git add/rm <文件>..." 标记解决方案）
双方删除：    index.htm
由我们添加：  index1.htm
由他们添加：  index2.htm
```

  - `git status`
    - `git commit -am'Decide to mv index to index1'`

```
所有冲突已解决但您仍处于合并中。
（使用 "git commit" 结束合并）
```

  - `git push`