Git分支操作：利用新分支备份状态，实现多分支版本管理

最近在改代码，因为偷懒，所以试了一下copilot，结果发现有点太乱了，所以想回退到之前的版本重新看一下，于是想到可以用另一个分支备份一下当前的成果，紧急学习了一下一些基本的分支操作，分享一下。

检查当前分支

```
git branch
```

此时我们会看到自己处于 `main` 或者 `master` 分支，之后按下 `q` 回到命令行。接下来的工作就是创建一个新的分支，用于储存你最新的更改。

创建新的分支

执行命令：

```
git branch <branch_name>
```

这样会创建一个新的名字为 `branch_name` 的分支。比如 `git branch copilot` 就会创建一个叫 `copilot` 的分支，此时你执行 `git branch` 就能看到新的分支，只不过 `*` 符号仍然指向你的原本的分支，表明你仍然处于旧的分支。

切换分支

使用命令：

```
git checkout <branch_name>
```

可以切换到你想要的另一个分支，执行后你使用 `git branch` 查看就可以发现 `*` 指向了你刚刚指定的分支，这就说明我们处于另一个分支了。

推送最新更改，完成备份

我们这时需要把你目前的状态保存到这个分支上。执行以下命令：

```
git add . # add all changes to staged status
git status # check if any file has yet to be staged
```

```
git commit -m "your commit message" # or use git commit -a -m "xxx" to mix
add and commit commands together.
git push origin <new_branch_name>
```

只要你成功把更改推送到这个分支上，这个分支会一并将你之前本地的`commit`历史都记录在当前分支的提交记录里，正好像一条河流的河水，从某一刻起脱离了主流（你的`main`/`master`分支），成为了一条真正的支流，所以这个“branch”也是很形象了。

回滚原分支状态

现在我们需要把原分支恢复到你之前的某一次提交后的状态。这需要实现两点：

- 将本地git的提交历史回滚到你想要的状态；
- 将本地git的提交历史/流同步到你的远程仓库
  我们分步来说这件事

1. 首先，**切换**回原先分支

```
git checkout master
```

2. 查看日志，根据你的提交信息确定你需要回滚到哪一次提交。

```
git log

commit fabf996d953095f596a9ea63d9a648ab6709c8cc
Author: ziyuliu258 <xxx@xxx.com>
Date:   Fri Oct 3 21:57:05 2025 +0800

    Read Part of xxx Codes

commit 707fdf2874be1c9ea7eec483ac6346afc0076fc3
Author: ziyuliu258 <xxx@xxx.com>
Date:   Mon Sep 29 11:14:40 2025 +0800

    add xxx score mechanism targeting xxx

commit 4a6307f834bd7a026a227b5717ecdab5475b1499
Author: ziyuliu258 <xxx@xxx.com>
Date:   Fri Sep 26 15:17:16 2025 +0800

    after running comparison with old model

......
```

我们记住我们要回滚到的那个提交的`commit ID`。
3. 强制回滚

```
git reset --hard fabf996d953095f596a9ea63d9a648ab6709c8cc
```

这样就能让你的本地`git`提交历史和文件状态都回到你指定的这个提交的位置。但是此时任务还没有完成，如果你没有把这个状态更新到远程仓库，那你仍然无法完成远程分支历史的更改。
4. 本地回滚推送回远程原分支
此时我们执行

```
git push origin master -f
```

来将提交状态强行推送到原本的`master`分支上，其中`-f`就是强制（force）推送的意思。
之后执行`git status`，就能看到你的本地状态和远程仓库的`master`分支一样了！