实验一 Git和Markdown基础

班级： 22物联1

学号： B20220305112

姓名： 张利文

Github地址：https://github.com/yourusername/python_course

实验目的

1. Git基础，使用Git进行版本控制
2. Markdown基础，使用Markdown进行文档编辑
3. vscode的基本使用

实验环境

1. Git
2. VSCode
3. VSCode插件

实验过程记录

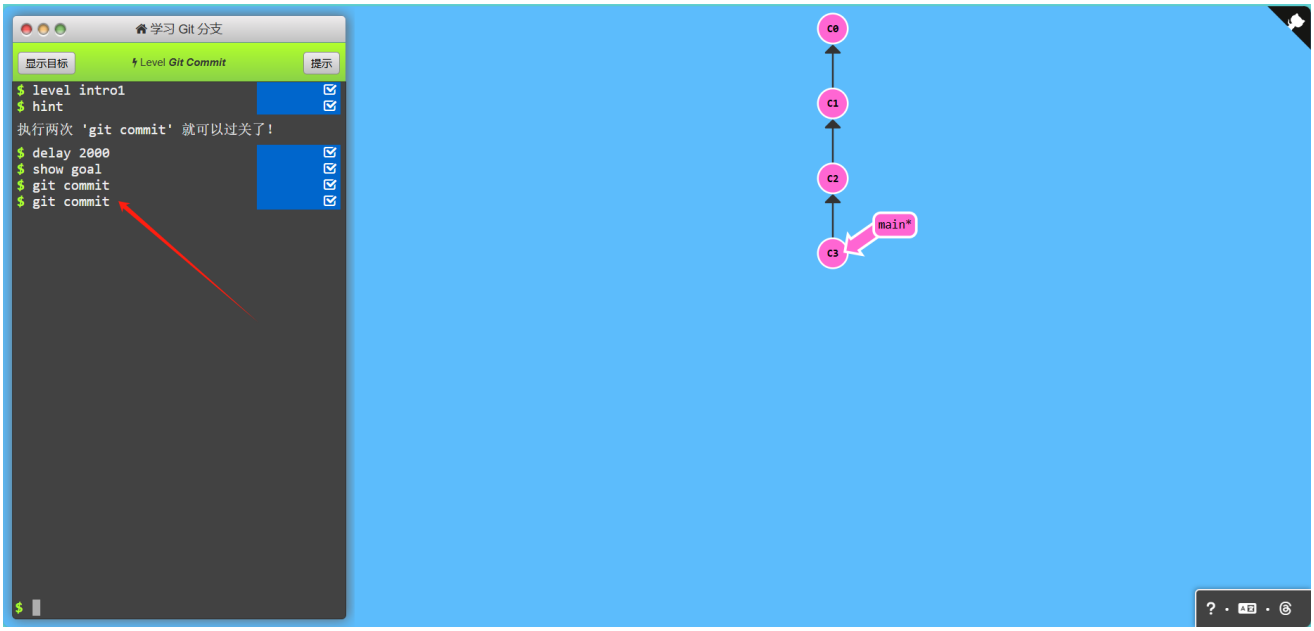
Git实验过程的记录请参考Learning Git Branch Tutorial，请记录下每个git小实验的：

- 动机：为什么要使用这个git命令，或者这个git命令的作用是什么？
- 使用的git命令：使用markdown代码块的形式记录下你使用的git命令，例如：

- git命令的解释：使用自然语言解释你使用的git命令

基础篇

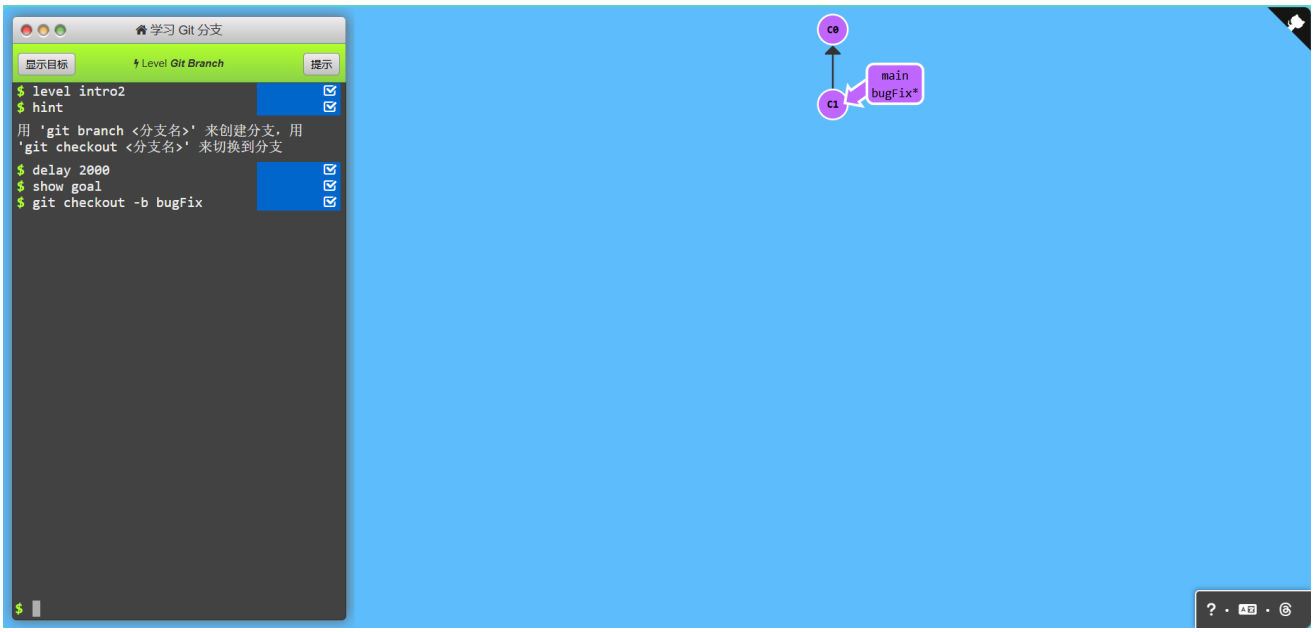
1.



git commit

`git commit` 是一个非常重要的命令，它帮助你保存你的工作进度，并给你一个机会去记录下每次更改的要点。这样，你就可以在未来任何时候回过头来，理解你为什么要做出这些更改。

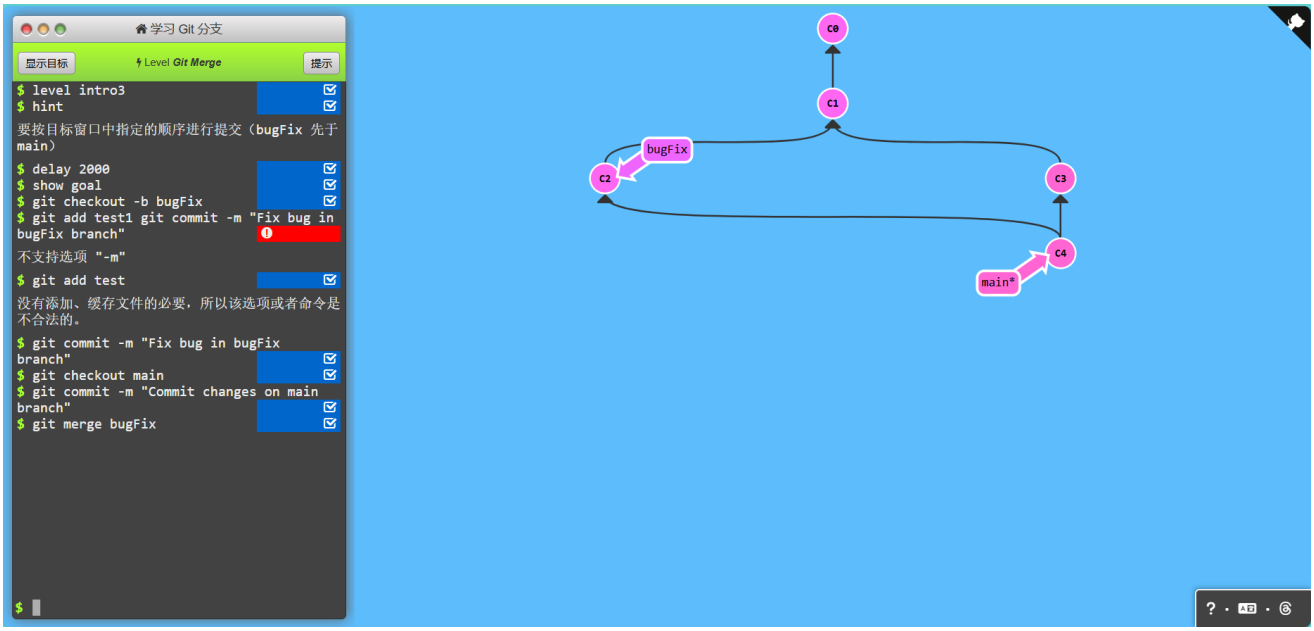
2.



git checkout -b bugFix

这个命令通常用于开始一个新的开发任务，比如修复一个 bug。通过创建一个新的分支，你可以专注于修复这个问题，而不用担心影响到其他正在进行的工作

3



git checkout -b bugFix

创建并检出（切换）到一个新分支，分支名为 `bugFix`

git commit -m "Fix bug in bugFix branch"

`git commit` 提交这些更改`-m` 后面跟着的是你的提交信息，描述了这次提交的目的。

git checkout main

切换到 `main` 分支

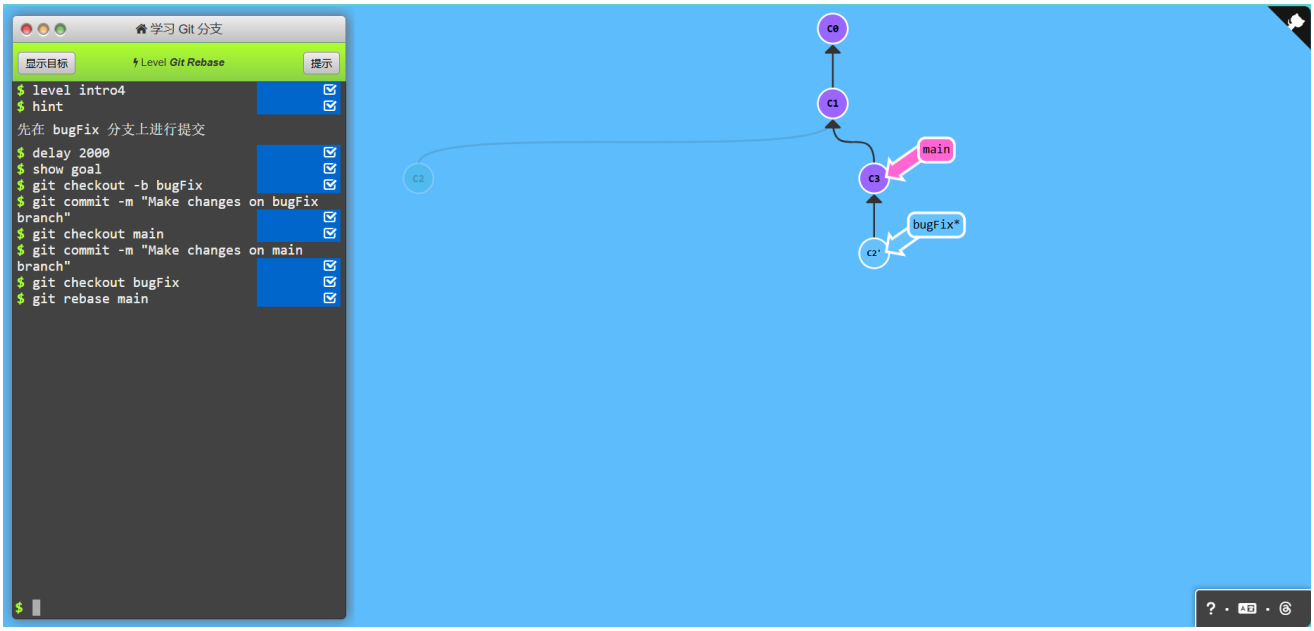
git commit -m "Commit changes on main branch"

在 `main` 分支上，你可能有一些其他的更改需要提交。同样地，然后使用 `git commit` 提交这些更改。

git merge bugFix

将 `bugFix` 分支合并到当前分支（这里是 `main` 分支）

4



git checkout -b bugFix

这个命令创建了一个新的分支 `bugFix` 并自动切换到该分支。

git commit -m "Make changes on bugFix branch"

在 `bugFix` 分支上进行一些更改后，使用 git commit命令将更改提交到bugFix` 分支。

git checkout main

使用 `git checkout main` 命令切换回 `main` 分支

git commit -m "Make changes on main branch"

在 `main` 分支上进行一些更改，并提交这些更改。

git checkout bugFix

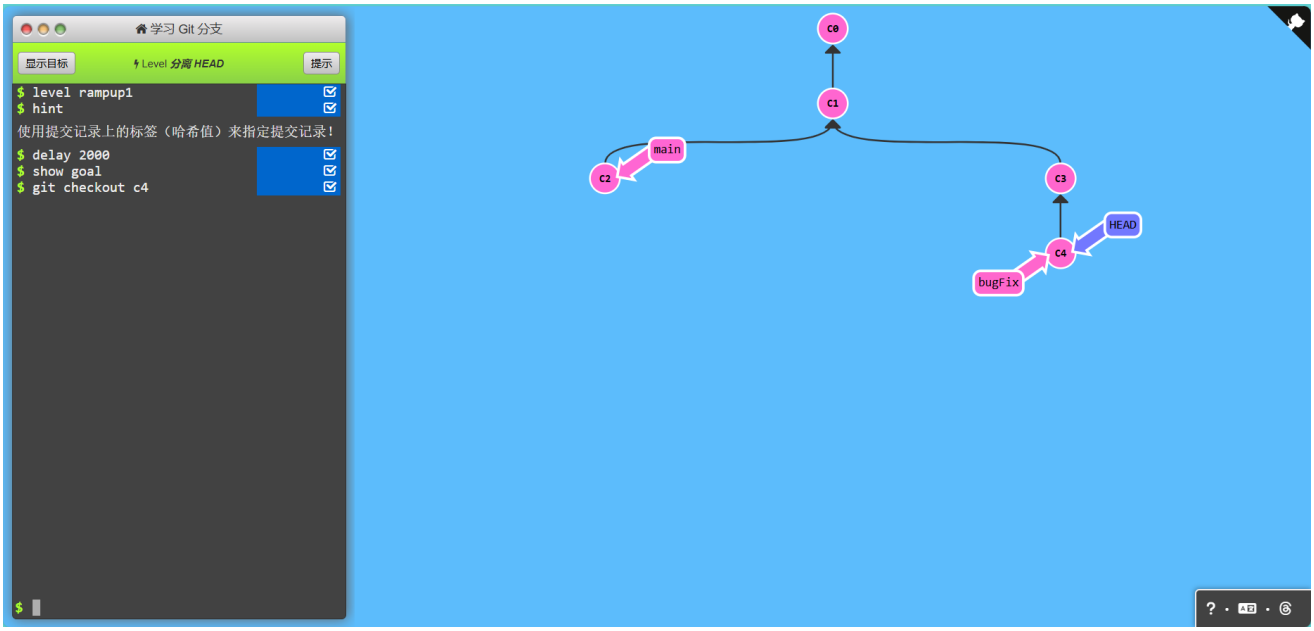
使用 `git checkout bugFix` 命令切换回 `bugFix` 分支。

git rebase main

将 `bugFix` 分支上的更改重新基于 `main` 分支的最新更改进行应用

高级篇

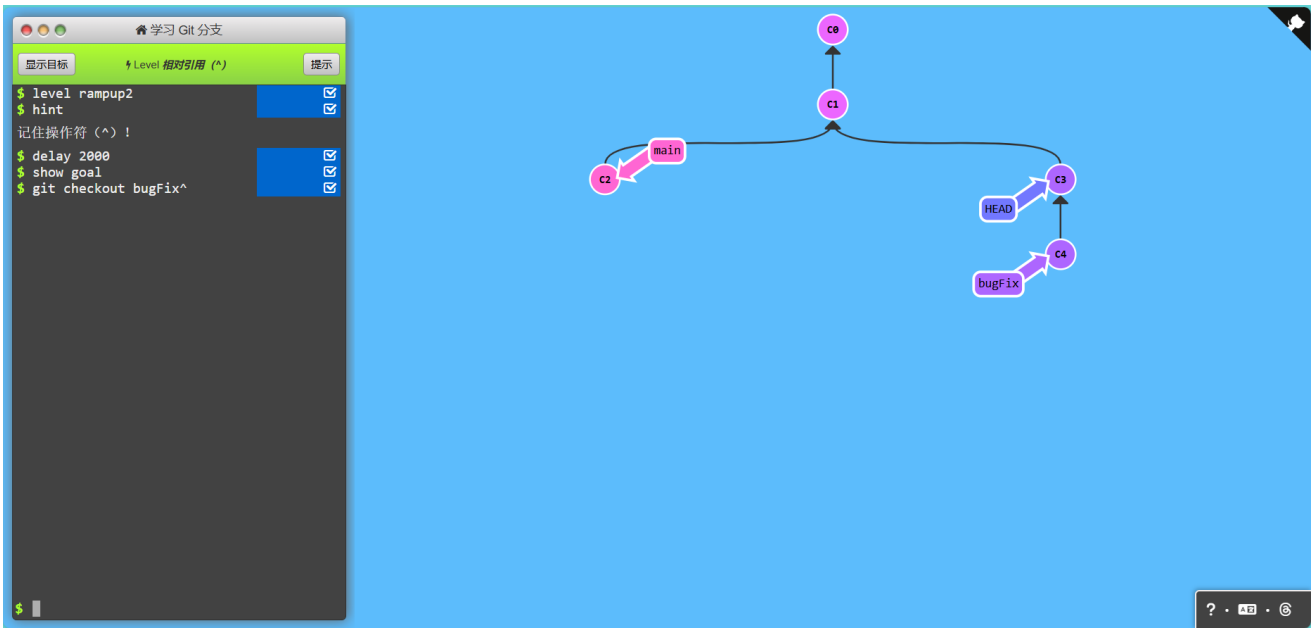
1.



git checkout c4

当你在 Git 中执行 `git checkout c4` 命令时，你实际上是在告诉 Git 将 `HEAD` 指向特定的提交。这里的 `c4` 应该被替换为一个更长的提交哈希值

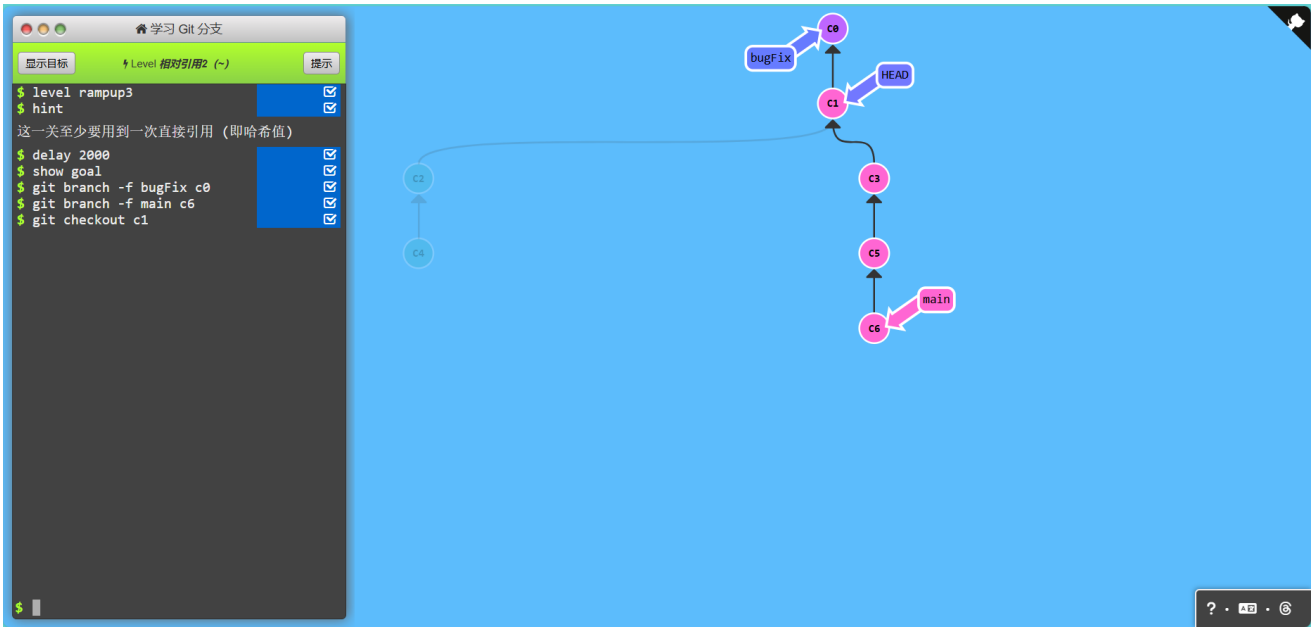
2.



git checkout bugFix^

这个命令会将 `HEAD` 移动到 `bugFix` 分支的最新提交的父节点，并且你将处于分离 HEAD 状态。

3.



git branch -f bugFix c0

这个命令会强制 (`-f`) 将 `bugFix` 分支的指针移动到提交 `c0`。

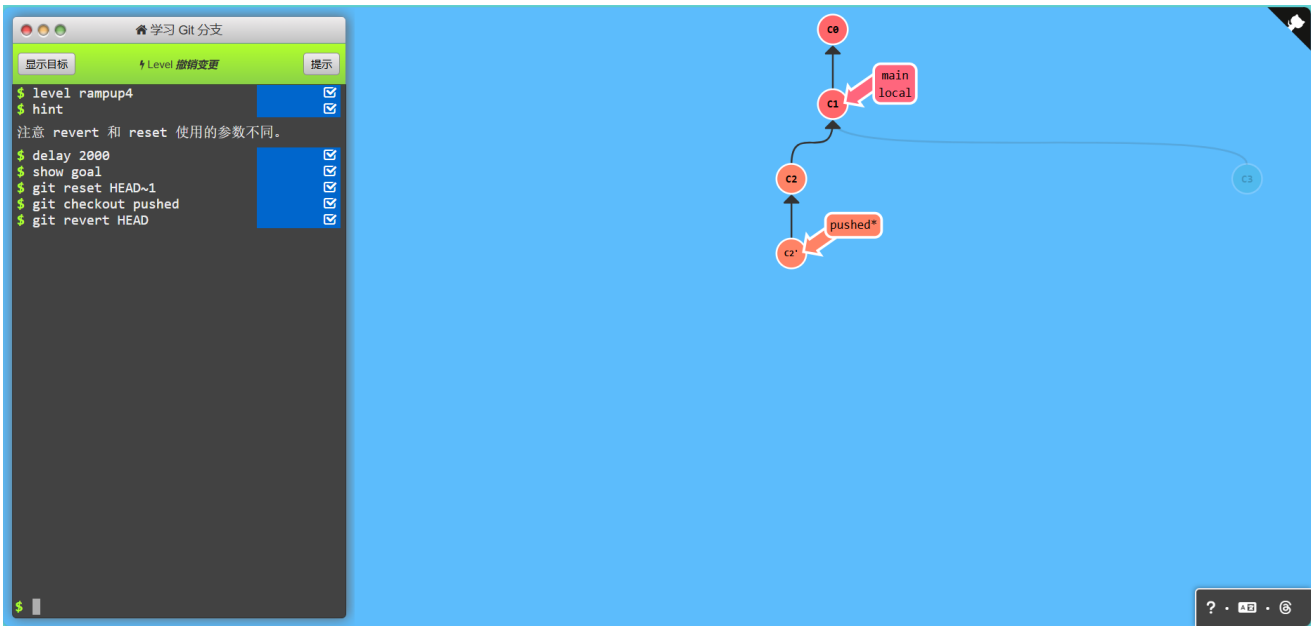
git checkout c1

这会将 `HEAD` 移动到提交 `c1`，但不会改变任何分支的指针。如果您想要将 `bugFix` 分支设置为 `c1`，那么您已经在上一步完成了这个操作。

git branch -f main c6

这个命令会强制 (`-f`) 将 `main` 分支的指针移动到提交 `c6`。

4.



git reset HEAD~1

撤销 local 分支的提交

git checkout pushed

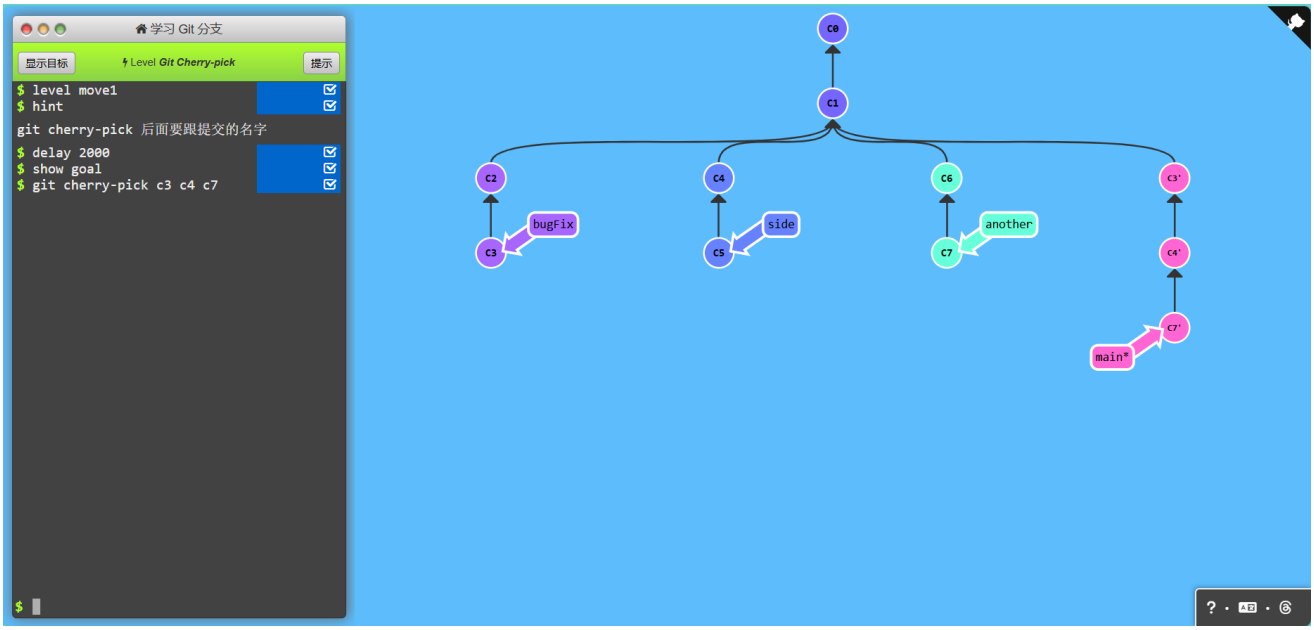
切换到 pushed 分支

git revert HEAD

撤销本地分支（local）上的最近一次提交

移动提交记录

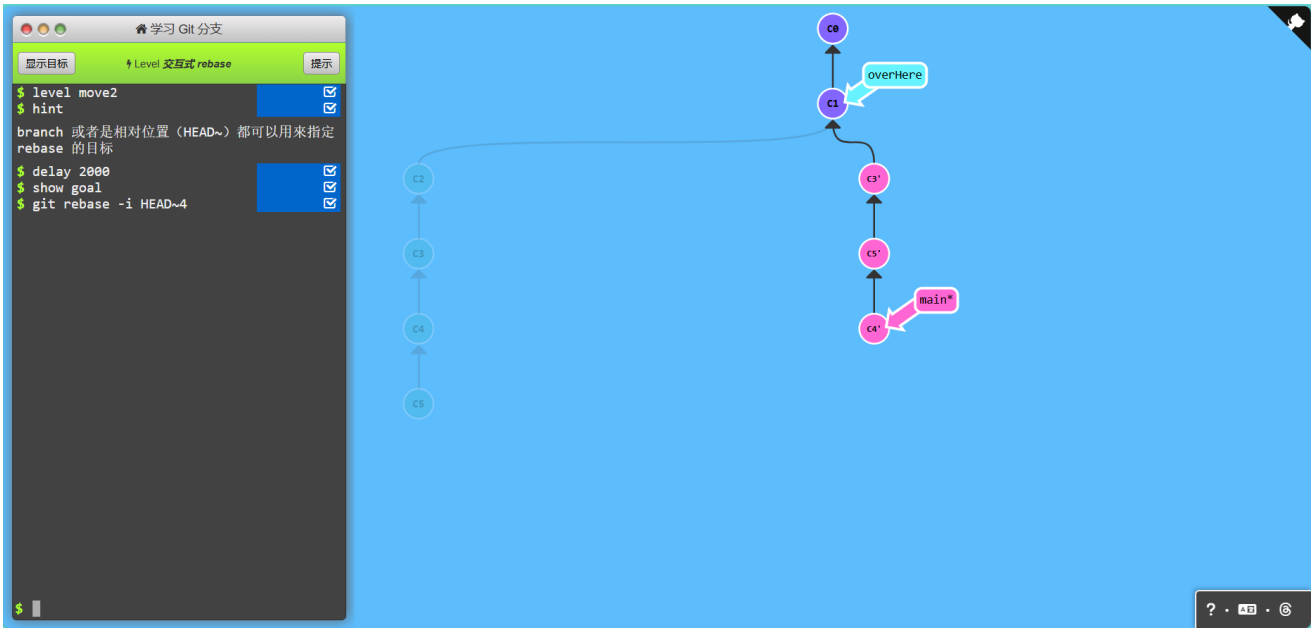
1



git cherry-pick c3 c4 c7

当你执行 `git cherry-pick c3 c4 c7` 命令时，Git 会尝试将提交 `c3`、`c4` 和 `c7`（这些提交的哈希值）从一个分支应用到你当前所在的分支上。

2.

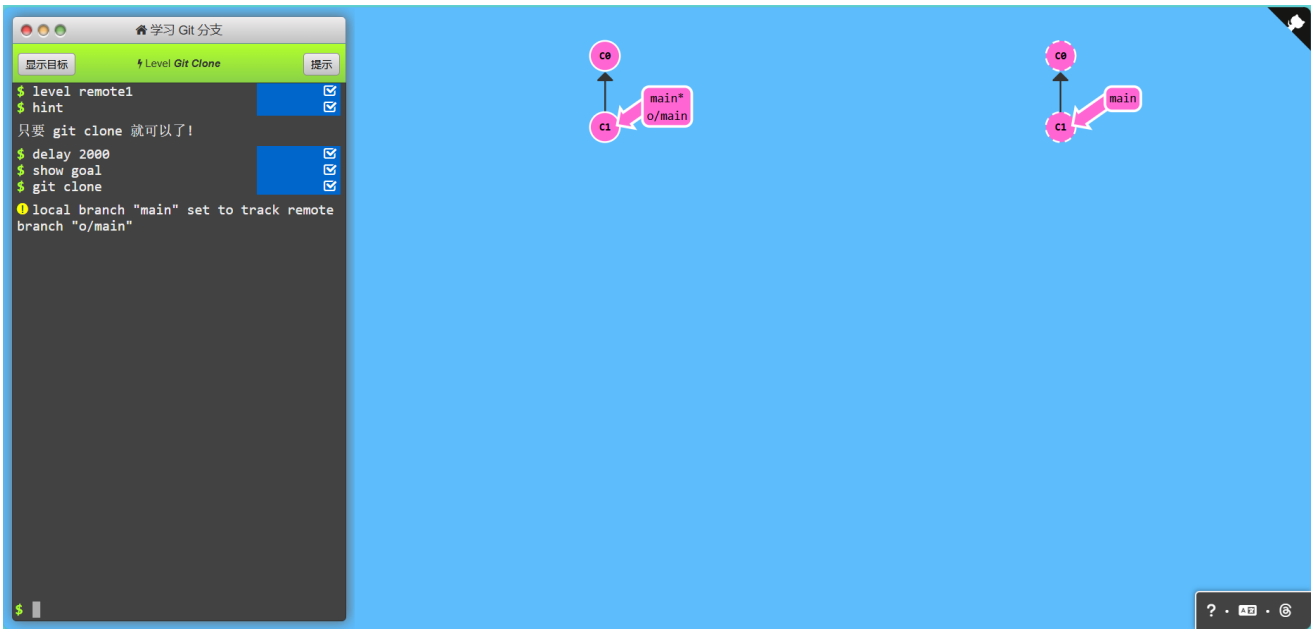


git rebase -i HEAD~4

`git rebase -i HEAD~4` 是一个 Git 命令，它允许你以交互的方式重新基线化（rebase）当前分支上最近的4个提交

远程

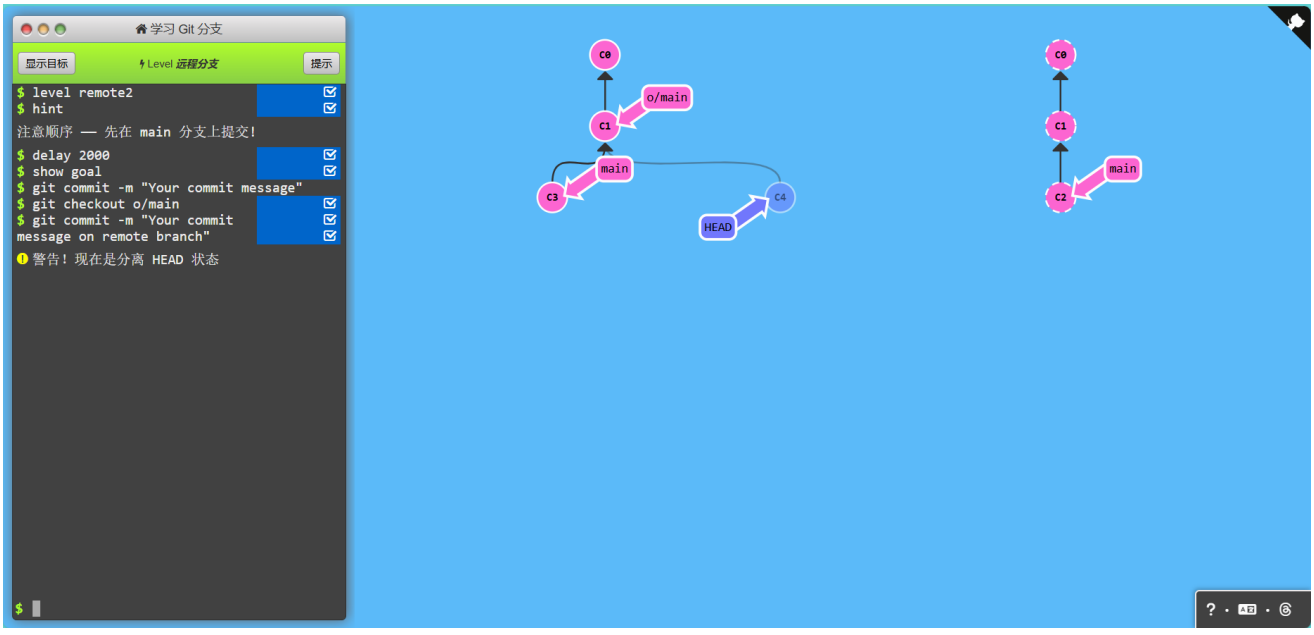
1.



git clone

`git clone` 命令在 Git 中用于从远程仓库复制一个 Git 仓库到本地计算机，并创建一个新的本地仓库。这个命令非常常用，当你需要开始在本地工作或创建一个现有项目的副本时，通常会用到它

2.



git commit -m "Your commit message"

提交更改

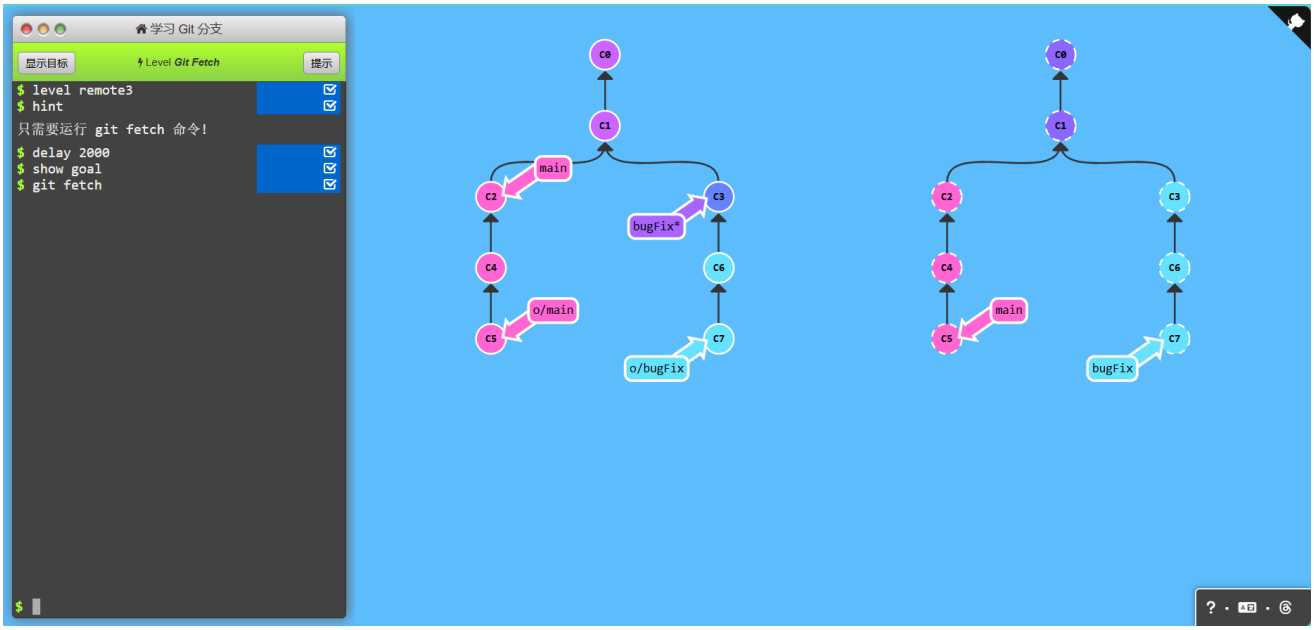
git push origin main

推送到远程 `main` 分支：

git commit -m "Your commit message on remote branch"

提交更改

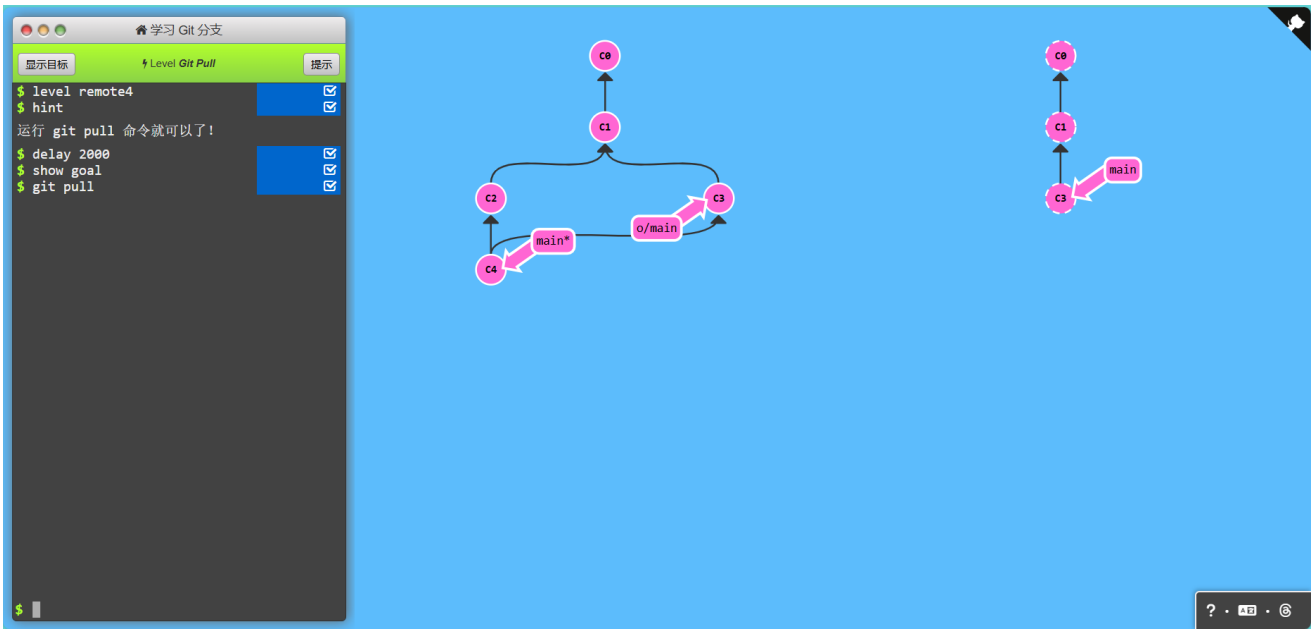
3.



git fetch

`git fetch` 命令用于从远程仓库获取数据并合并到你的本地仓库。这个命令主要用于获取远程仓库的最新更改，以便你可以在本地查看或合并这些更改。

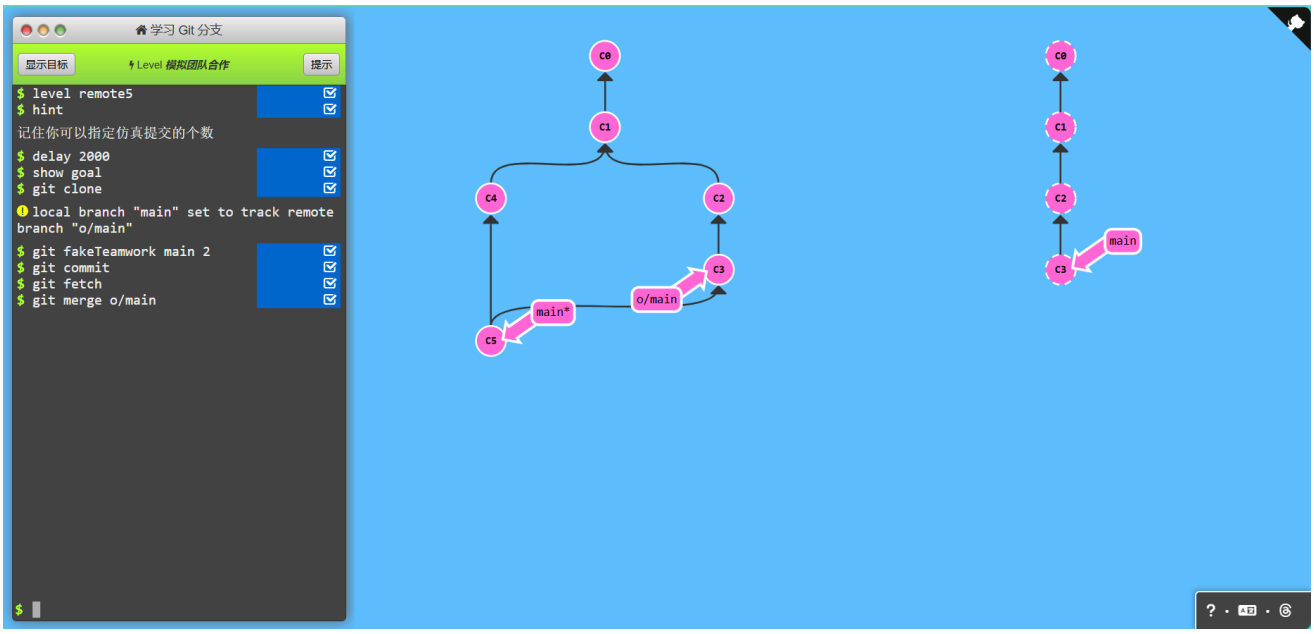
4.



git pull

`git pull` 命令是 Git 中用来从远程仓库拉取内容并尝试将其合并到你的当前本地分支的命令。这个命令实际上是两个 Git 操作的组合：`git fetch` 和 `git merge`。

5.



git clone

`git clone`命令克隆了远程仓库，并且本地的`master`分支被设置为跟踪远程的`origin/master`分支。

git fakeTeamwork main 2

模拟团队合作的简化命令

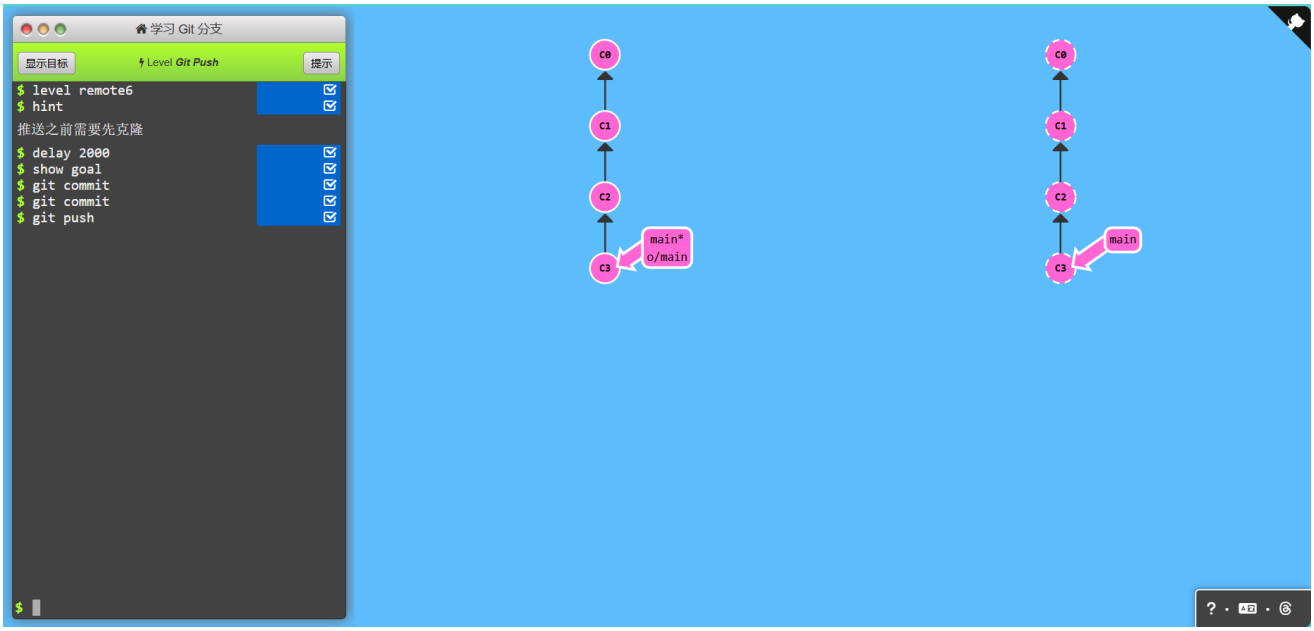
git commit -m "Your commit message on remote branch"

进行提交

git pullg

`git pull` 命令是 Git 中的一个常用命令，它用于从远程仓库拉取内容并尝试将其合并到当前检出（checked out）的分支。`git pull` 命令实际上是两个命令的组合：`git fetch` 和 `git merge`。

6.



git commit

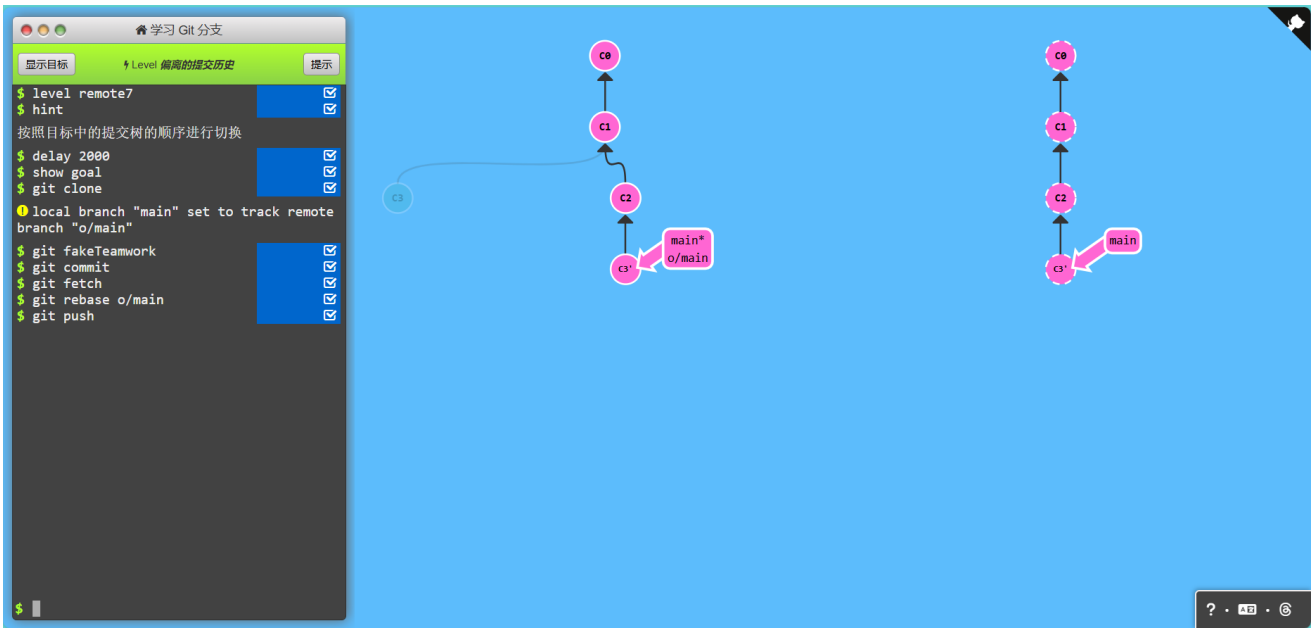
git commit

二次提交

git push

git push 负责将你的变更上传到指定的远程仓库，并在远程仓库上合并你的新提交记录。可以将 git push 想象成发布你成果的命令

7.



git clone

`git clone`命令克隆了远程仓库，并且本地的`master`分支被设置为跟踪远程的`origin/master`分支。

git fakeTeamwork

模拟团队合作的简化命令

git commit

提交

git fetch

使用 `git fetch` 命令来拉取远程仓库的所有分支和标签的信息，但不会自动合并到当前分支

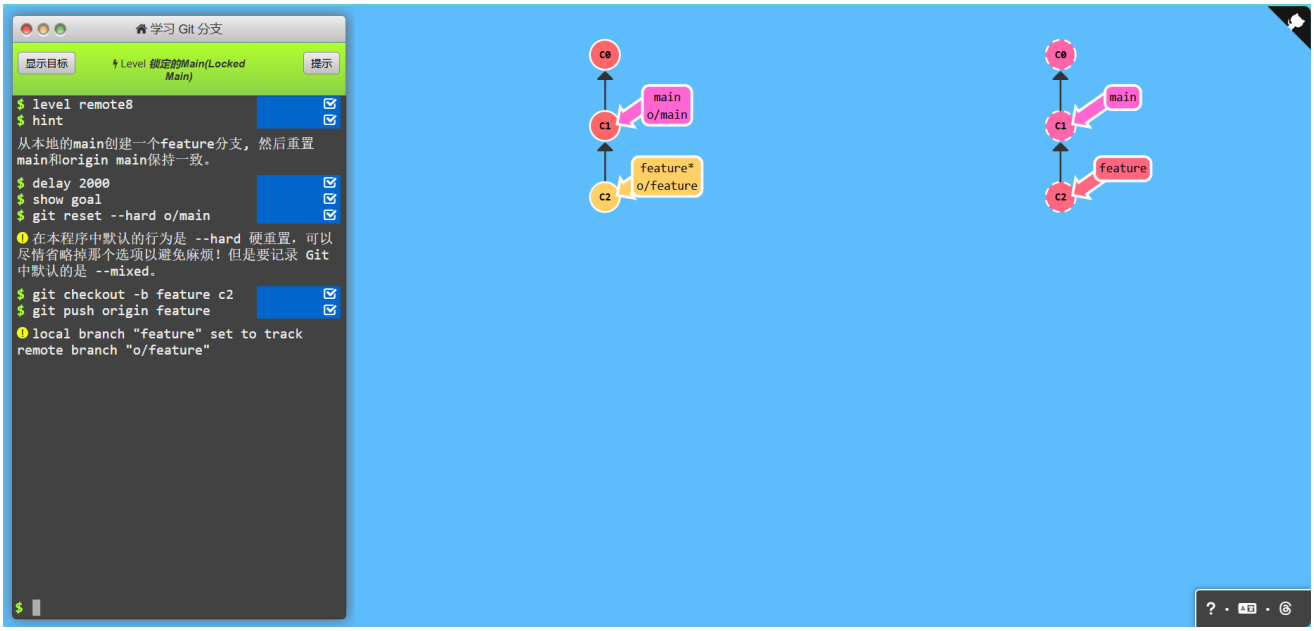
git rebase o/main

使用 `git rebase o/main` 命令将你的本地更改重新应用到远程的 `main` 分支的最新状态之上。

git push

使用 `git push` 命令将你的更改推送到远程仓库。

8.



git reset --hard o/main

使用 `git reset --hard o/main` 命令将你的本地 `main` 分支重置为远程 `origin/main` 分支的状态。这个命令会丢弃本地 `main` 分支上的所有未提交的更改和提交。

git checkout -b feature c2

使用 `git checkout -b feature c2` 命令基于特定的提交（提交哈希为 `c2`）创建并切换到一个新的分支 `feature`

git push origin feature

使用 `git push` 命令将你的新分支 `feature` 推送到远程仓库

实验总结

总结一下这次实验你学习和使用到的知识，例如：编程工具的使用、数据结构、程序语言的语法、算法、编程技巧、编程思想。

Learn Git Branching学习笔记 - 朴素贝叶斯 - 博客园 (cnblogs.com)