实验一 Git和Markdown基础

班级： 21计科1

学号： B20210302128

姓名： 肖锟

Github地址：https://github.com/xiaokun8888/python.git

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

实验环境

1. Git
2. VSCode
3. VSCode插件

实验内容和步骤

第一部分 实验环境的安装

1. 安装git，从git官网下载后直接点击可以安装：git官网地址
2. 从Github克隆课程的仓库：课程的仓库地址，运行git bash应用（该应用包含在git安装包内），在命令行输入下面的命令（命令运行成功后，课程仓库会默认存放在Windows的用户文件夹下）

```
git clone https://github.com/zhoujing204/python_course.git
```

如果你在使用`git clone`命令时遇到SSL错误，请运行下面的git命令(这里假设你的Git使用了默认安装目录)：

```
git config --global http.sslCAInfo "C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt"
```

或者运行下面的命令:

```
git config --global http.sslVerify false
```

该仓库的课程材料后续会有更新，如果需要更新课程材料，可以在本地课程仓库的目录下运行下面的命令：

```
git pull
```

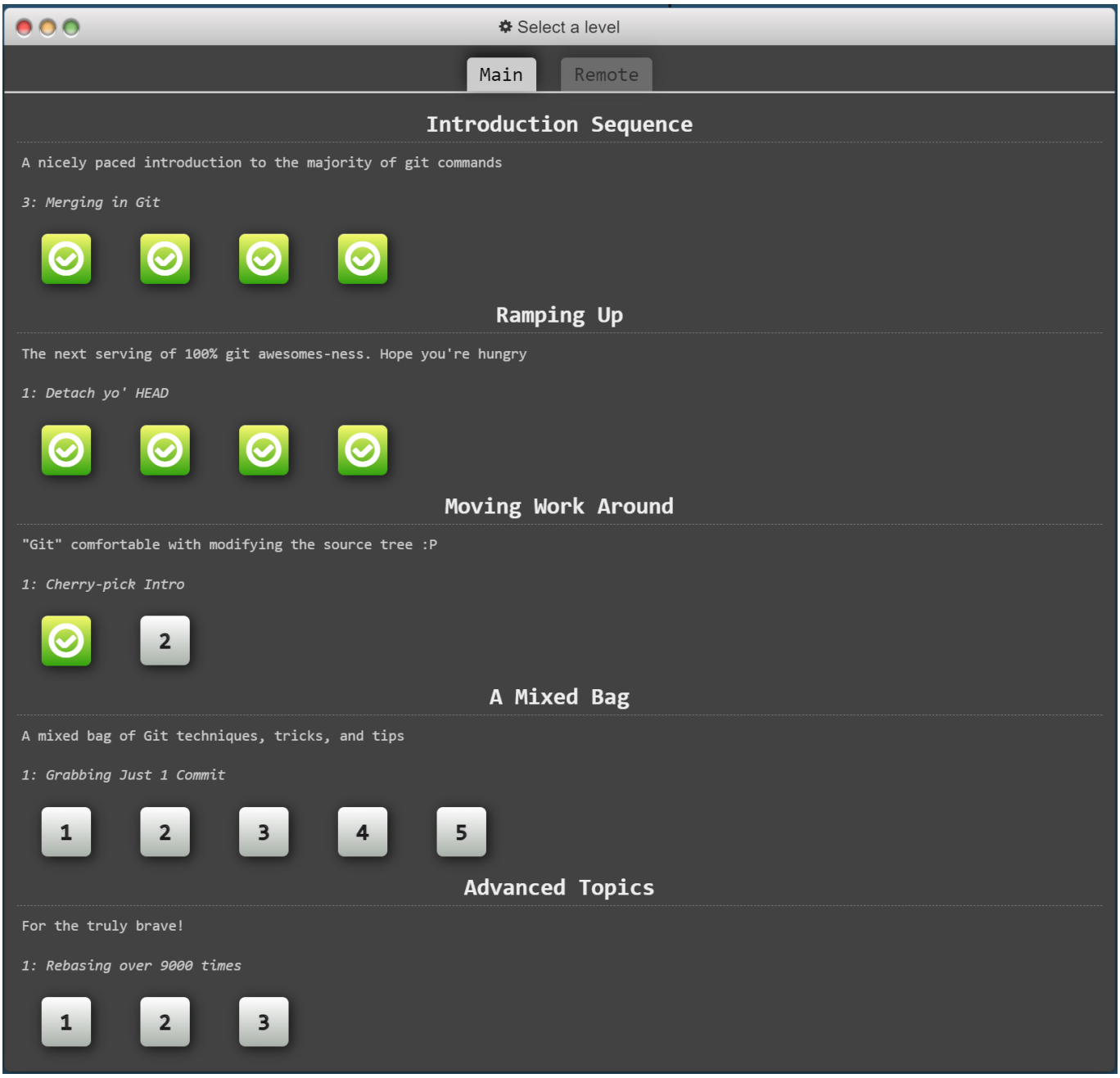
3. 注册Github账号，创建一个新的仓库，用于存放实验报告和实验代码。
4. 安装VScode，下载地址：Visual Studio Code
5. 安装下列VScode插件
    - GitLens
    - Git Graph
    - Git History
    - Markdown All in One
    - Markdown Preview Enhanced
    - Markdown PDF
    - Auto-Open Markdown Preview
    - Paste Image
    - markdownlint

第二部分 Git基础

教材《Python编程从入门到实践》P440附录D：使用Git进行版本控制，按照教材的步骤，完成Git基础的学习。

第三部分 learngitbranching.js.org

访问learngitbranching.js.org，如下图所示完成Main部分的Introduction Sequence和Ramping Up两个小节的学习。



上面你学习到的git命令基本上可以应付百分之九十以上的日常使用，如果你想继续深入学习git，可以：

- 继续学习learngitbranching.js.org后面的几个小节（包括Main和Remote）
- 在日常的开发中使用git来管理你的代码和文档，用得越多，记得越牢
- 在git使用过程中，如果遇到任何问题，例如：错误删除了某个分支、从错误的分支拉取了内容等等，请查询git-flight-rules

第四部分 Markdown基础

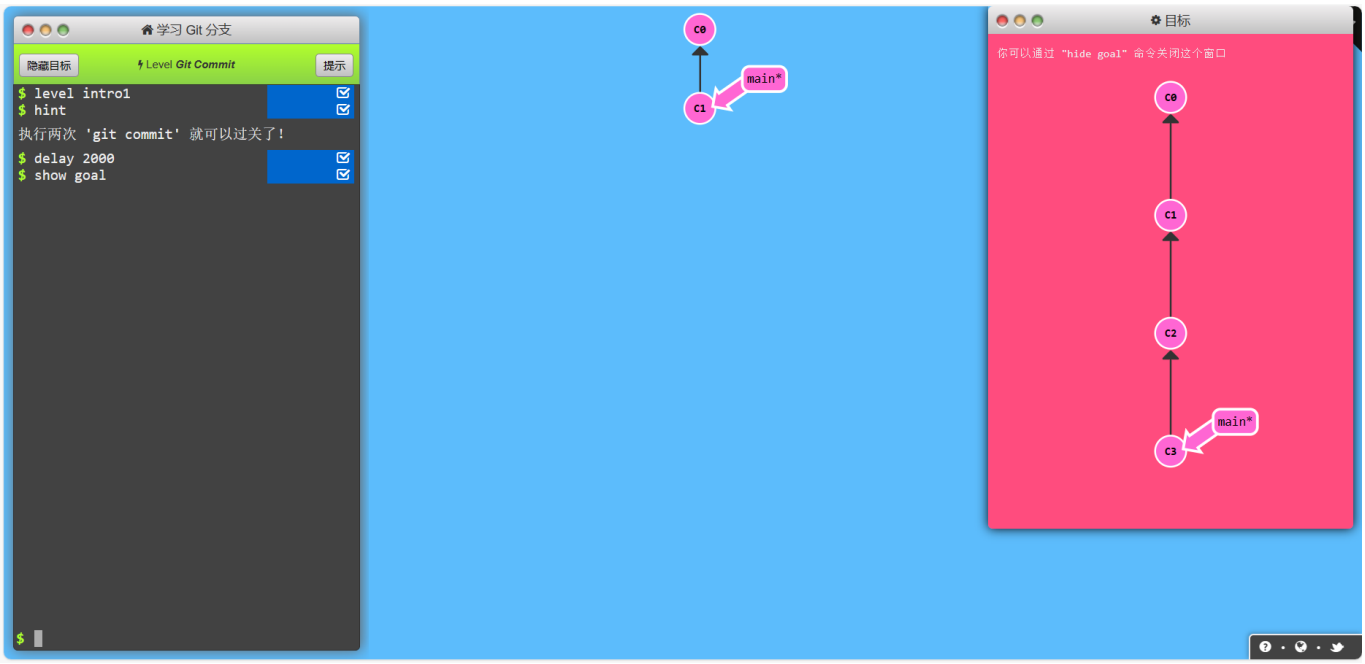
查看Markdown cheat-sheet，学习Markdown的基础语法

使用Markdown编辑器（例如VScode）编写本次实验的实验报告，包括实验过程与结果、实验考查和实验总结，并将其导出为 **PDF格式** 来提交。

实验过程与结果

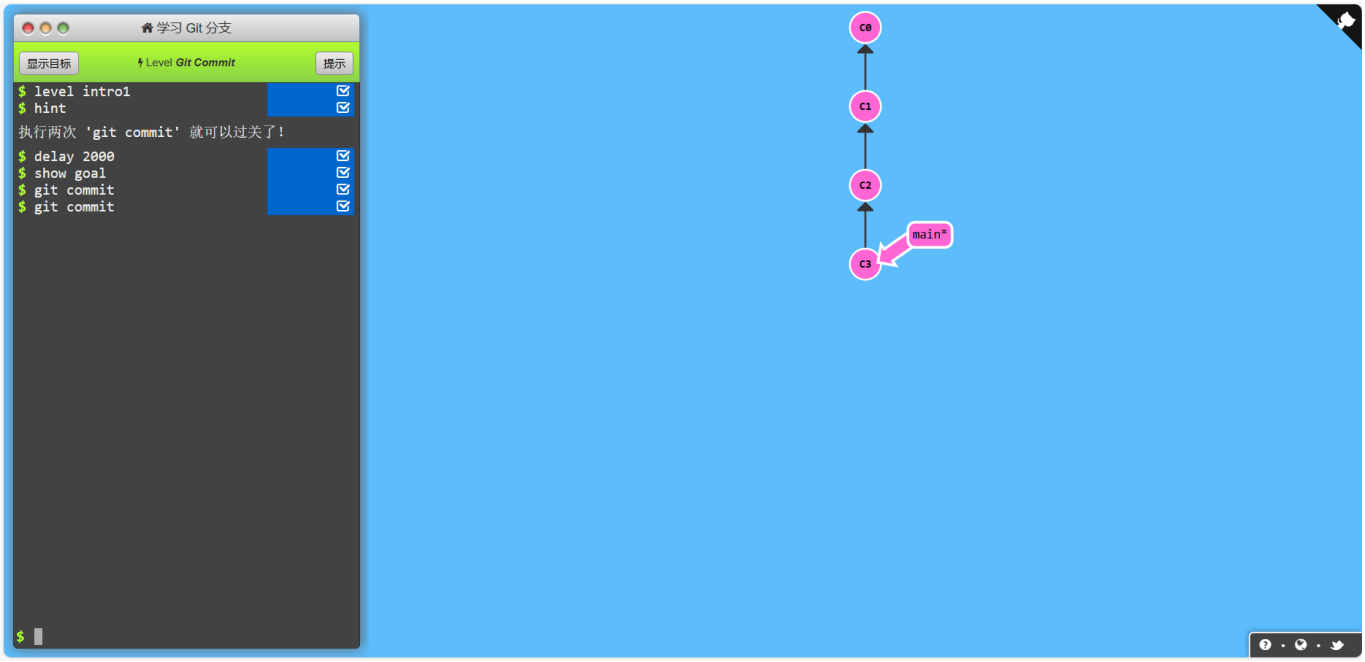
基础篇

1.执行两次 'git commit'

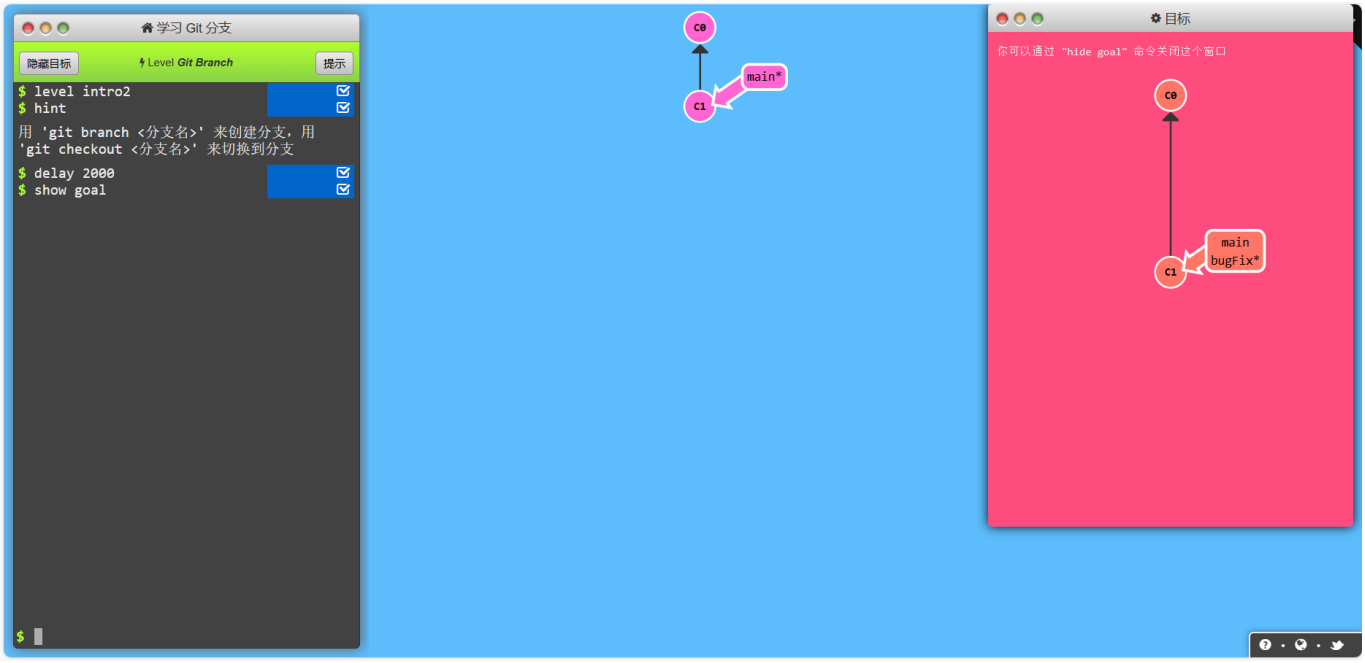



```
git commit
git commit
```

运行结果

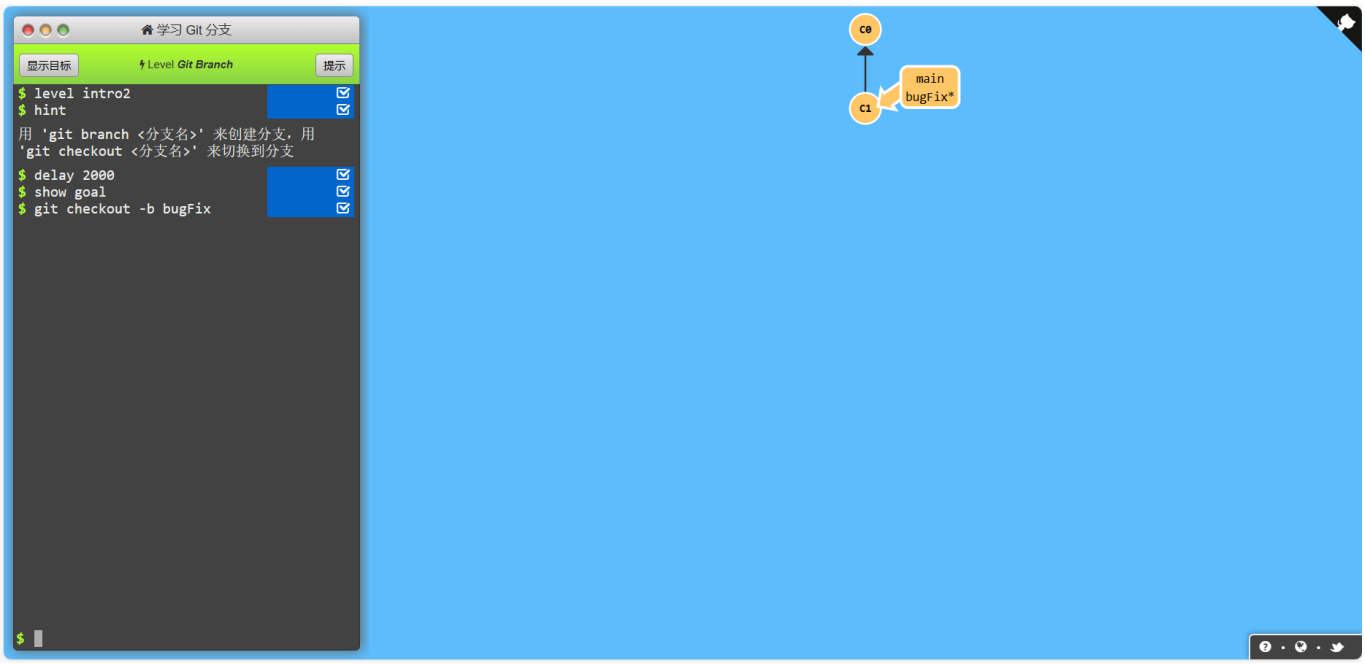


2.用 'git branch <分支名>' 来创建分支，用 'git checkout <分支名>' 来切换到分支

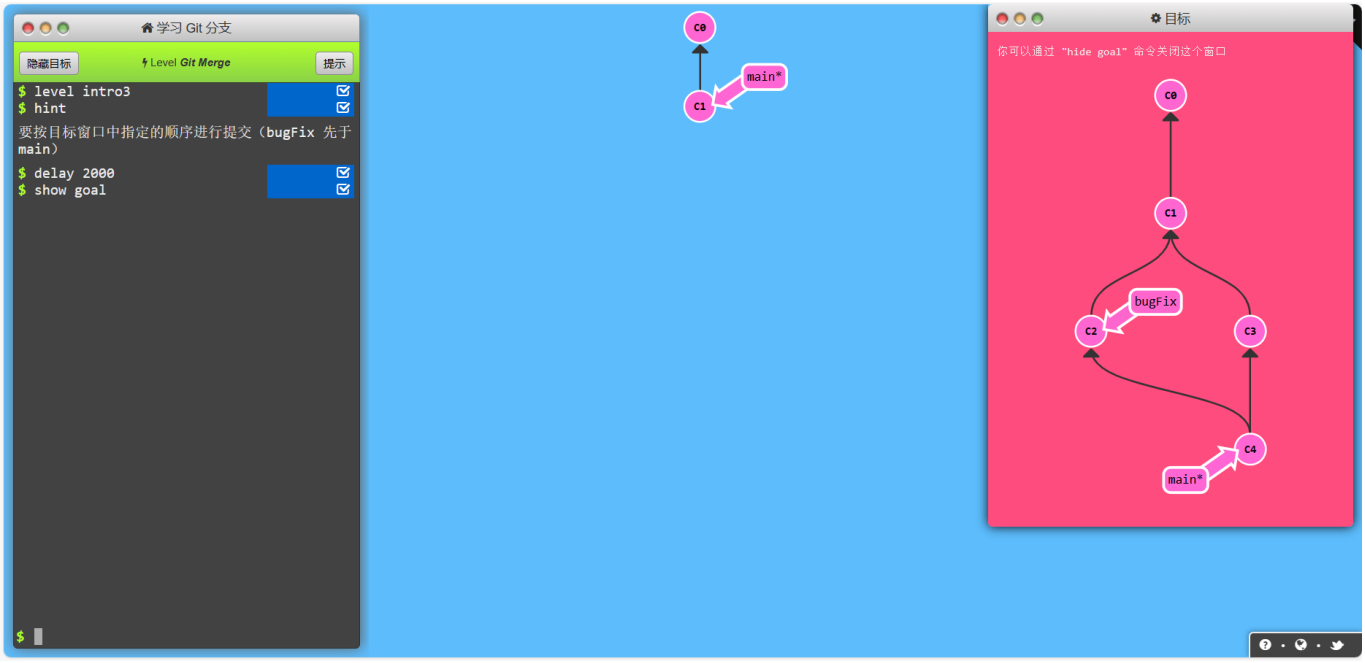



```
git checkout -b bugFix
```

运行结果

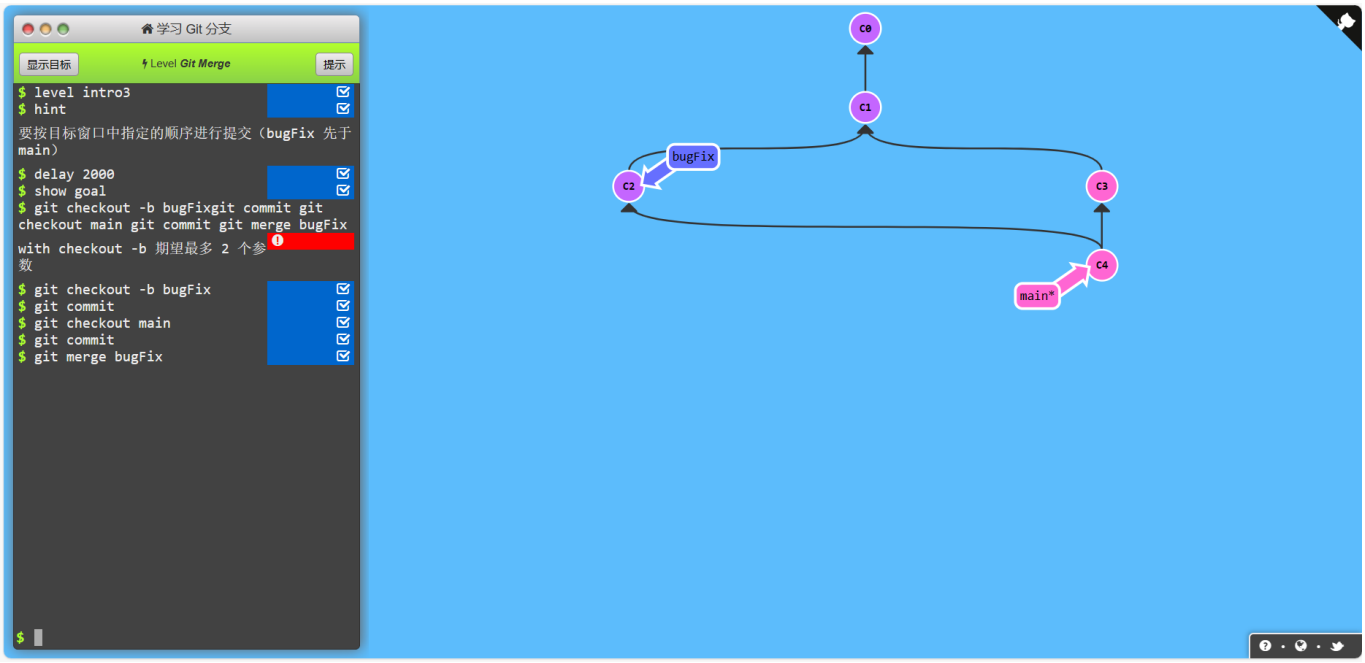


3.创建新分支 bugFix,用 git checkout bugFix 命令切换到该分支,提交一次,用 git checkout main 切换回 main,再提交一次用 ,git merge 把 bugFix 合并到 main

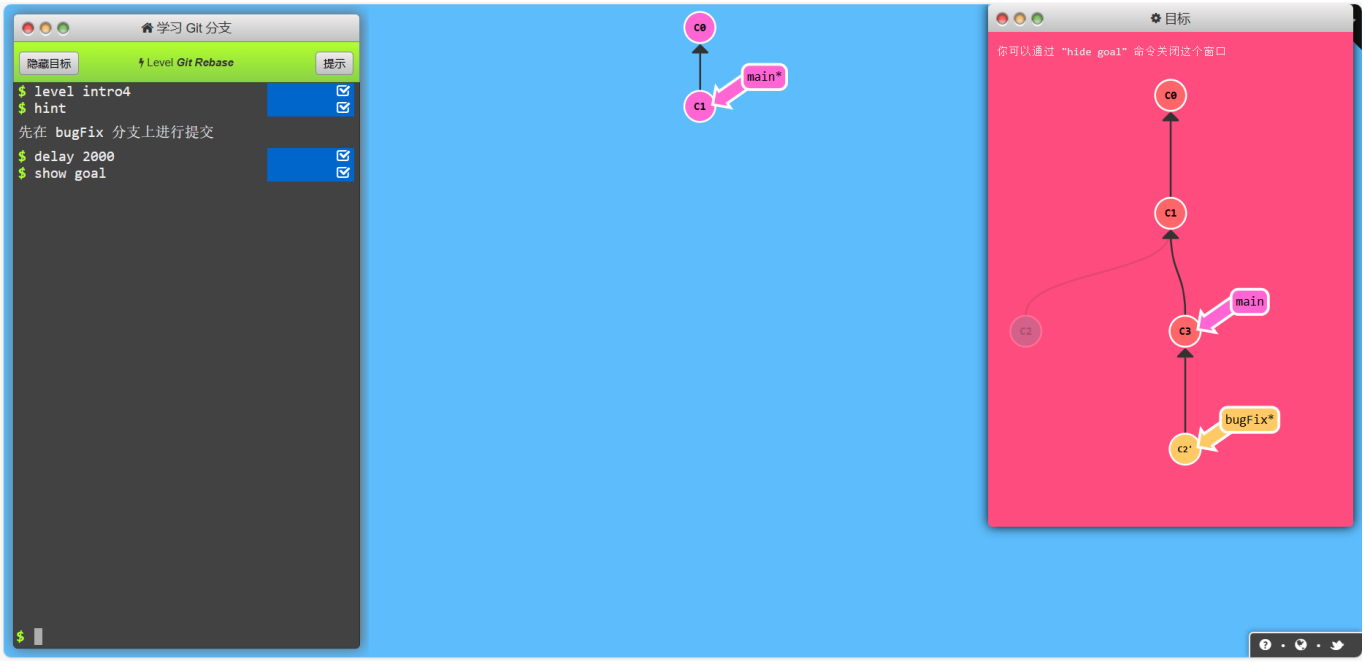



```
git checkout -b bugFix
git commit
git checkout main
git commit
git merge bugFix
```

运行结果

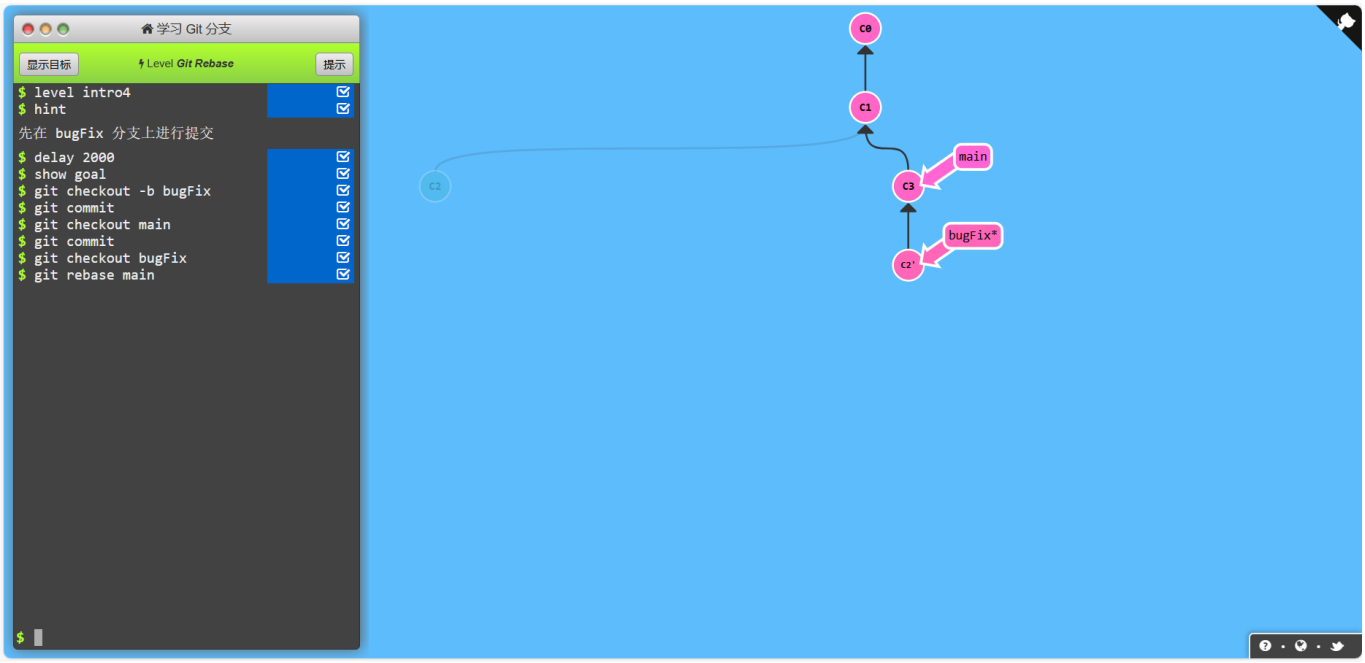


4.要完成此关，执行以下操作：新建并切换到 bugFix 分支，提交一次，切换回 main 分支再提交一次，再次切换到 bugFix 分支，rebase 到 main 上



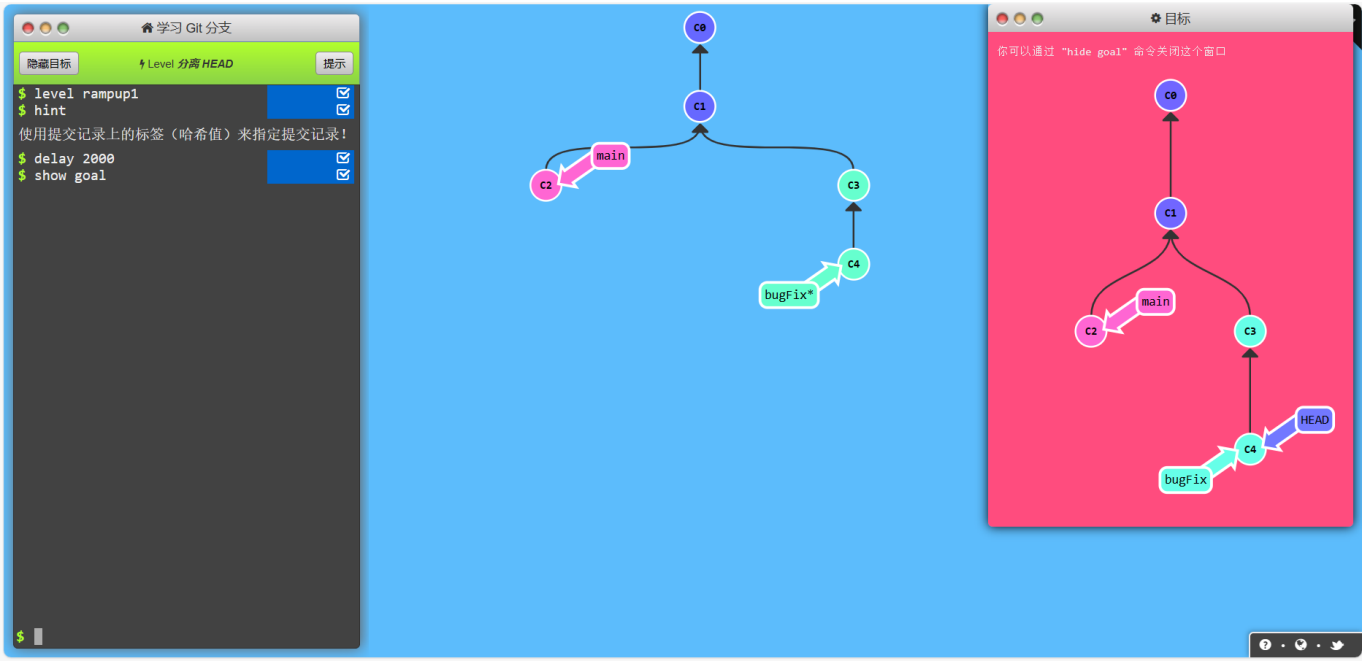

```
git checkout -b bugFix
git commit
git checkout main
git commit
git checkout bugFix
git rebase main
```

运行结果



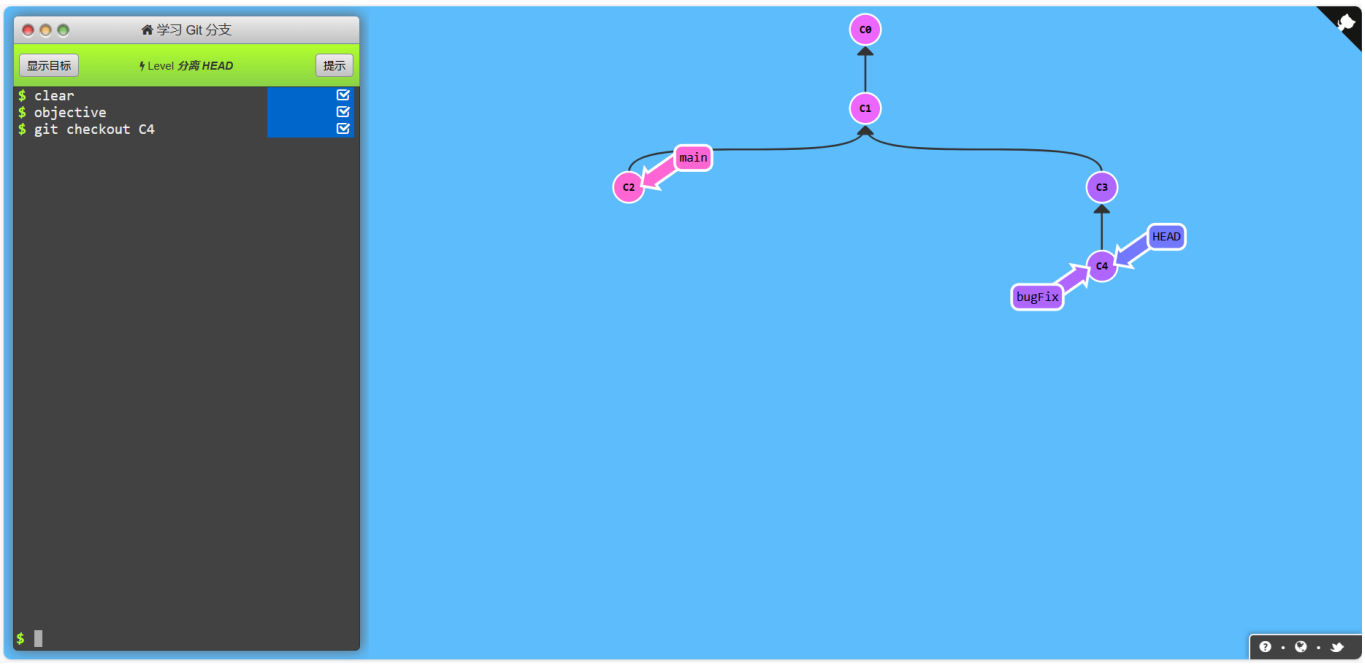
高级篇

5.分离HEAD

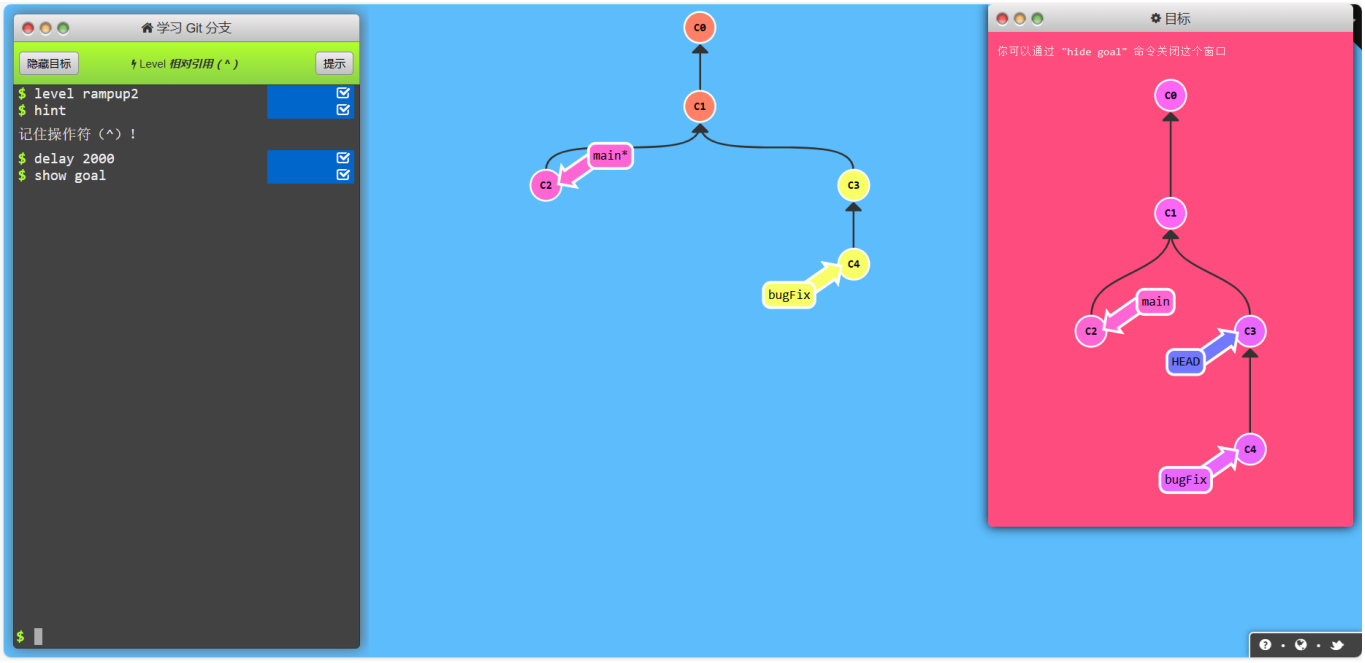



```
git checkout C4
```

运行结果

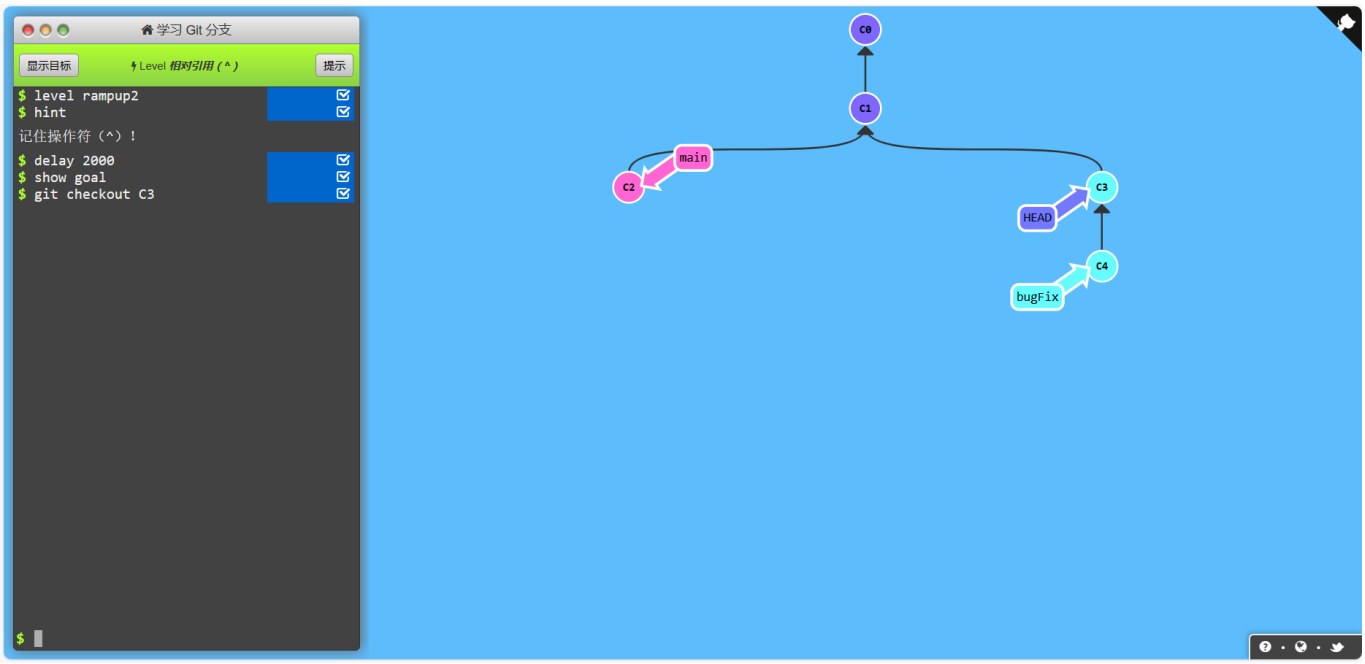


6.相对引用

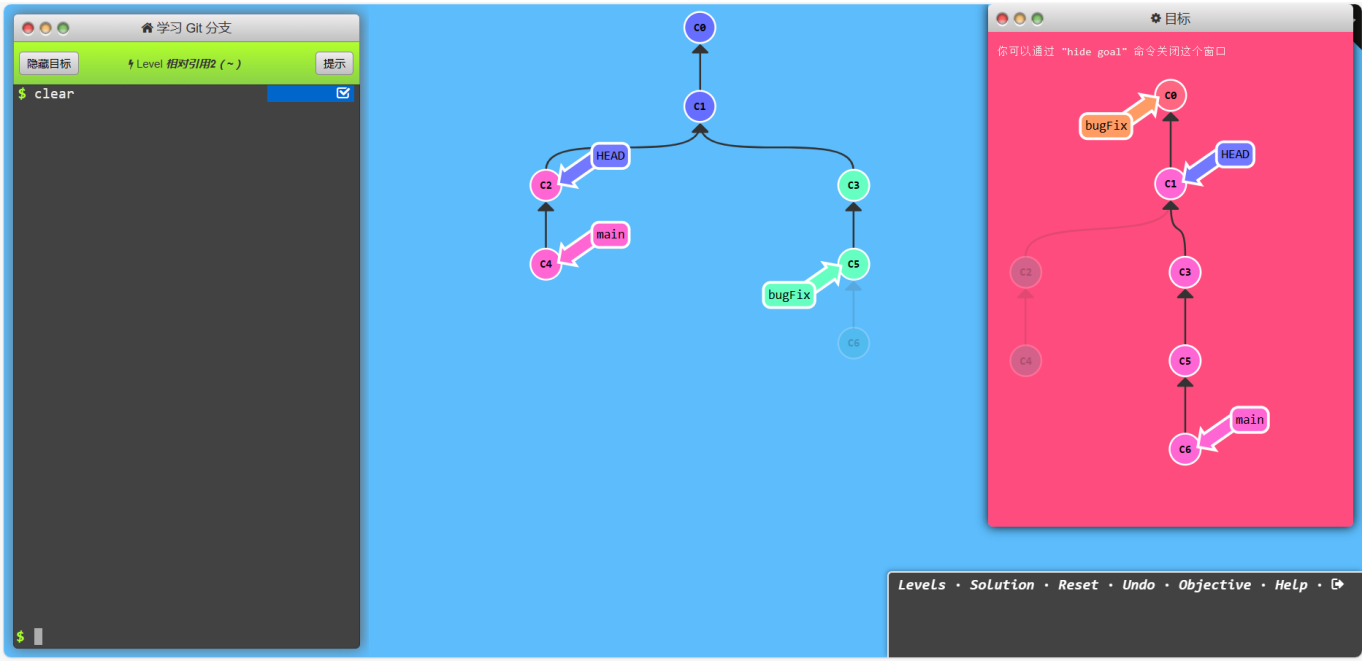



```
git checkout C3
```

运行结果

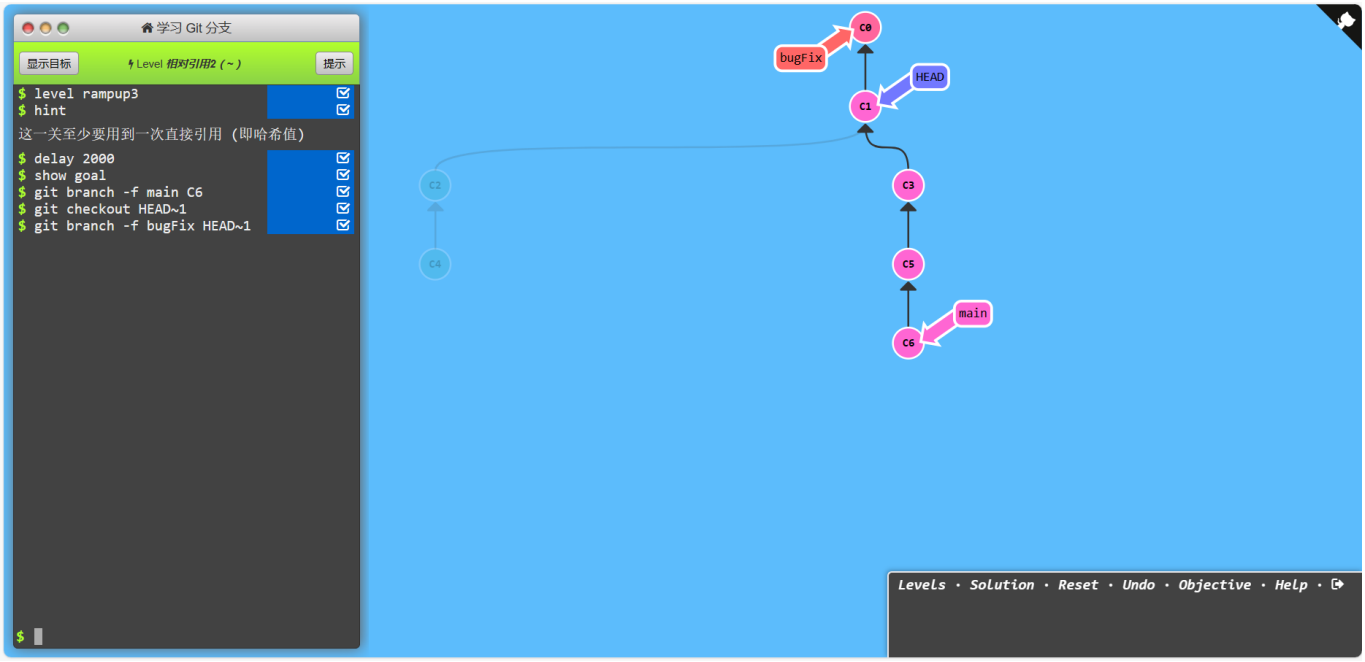


7.相对引用二

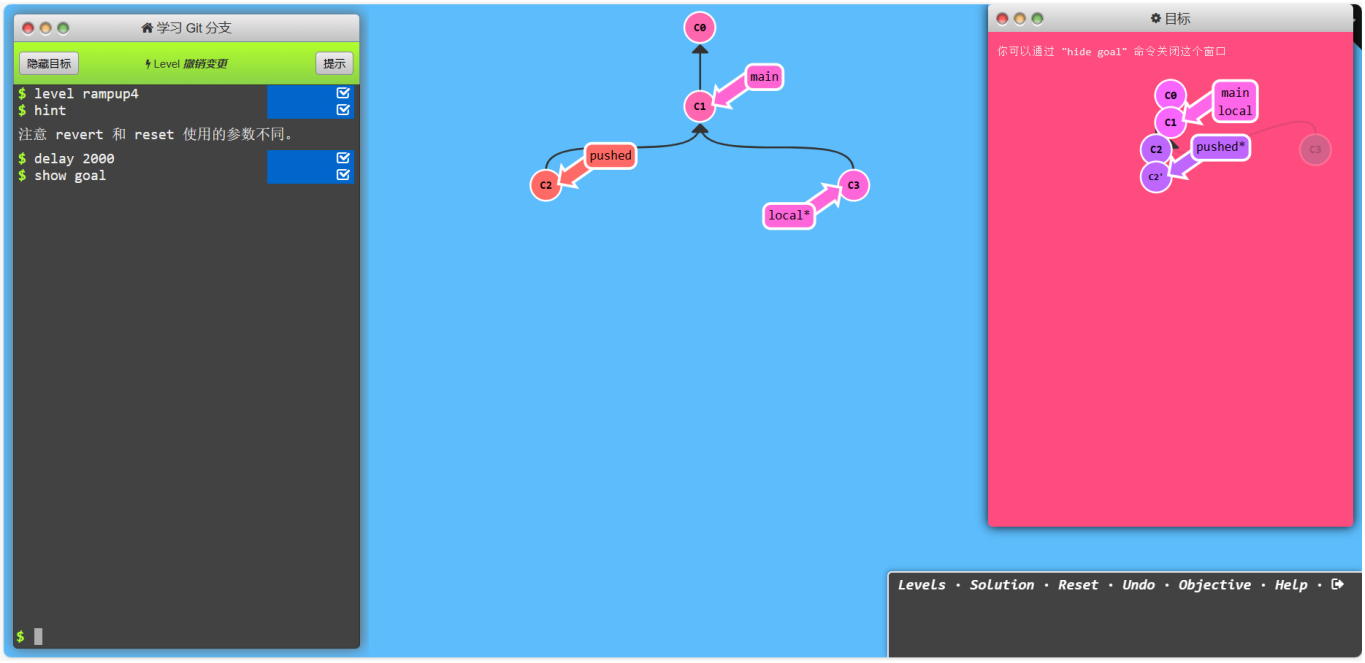



```
git branch -f main C6
git checkout HEAD~1
git branch -f bugFix C0
```

运行结果

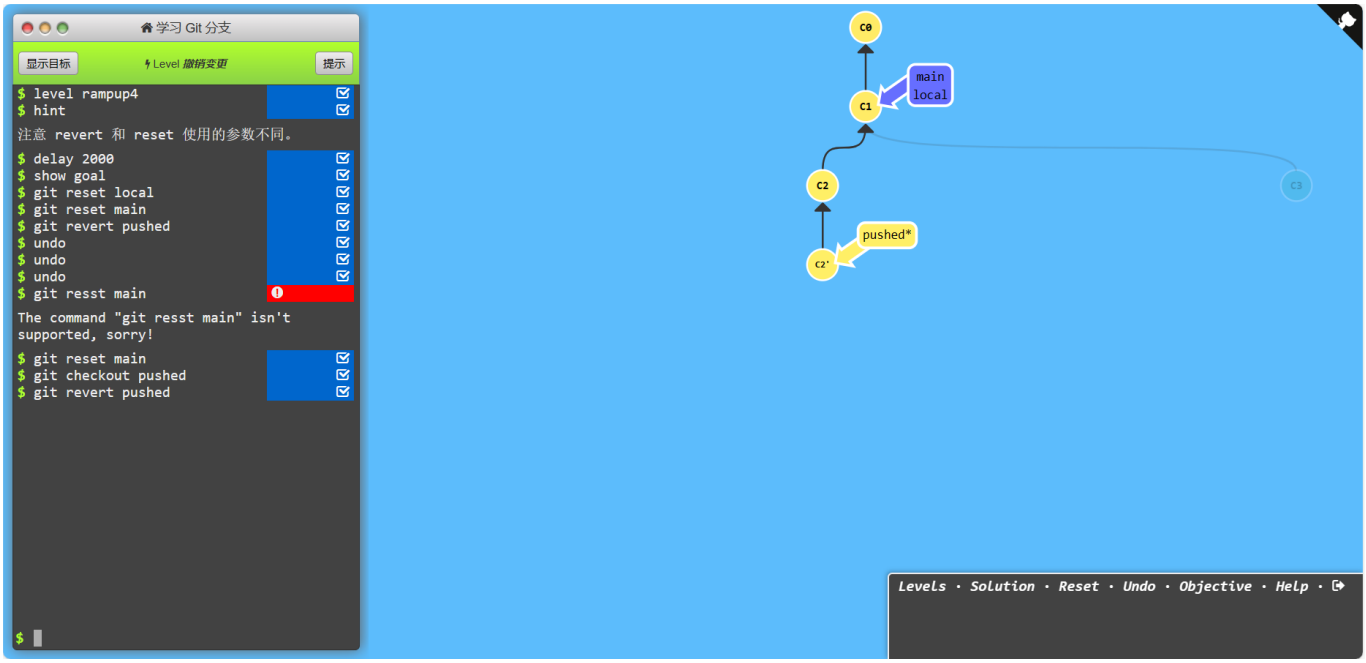


8.撤销变更




```
git reset main
git checkout pushed
git revert pushed
```

运行结果



实验考查

请使用自己的语言回答下面的问题，这些问题将在实验检查时用于提问和答辩，并要求进行实际的操作。

1. 什么是版本控制？使用Git作为版本控制软件有什么优点？使用Git作为版本控制软件有什么优点？ 版本控制是管理和控制计算机文件的系统，它能记录每个版本的信息； git开源，轻量快速，分布式架构，版本历史记录
2. 如何使用Git撤销还没有Commit的修改？如何使用Git检出（Checkout）已经以前的Commit？（实际操作） git restore; git checkout

3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作） HEAD(大写)是"current branch"(当下的分支)。当你用git checkout切换分支的时候，HEAD 修订版本重新指向新的分支。有的时候HEAD会指向一个没有分支名字的修订版本，这种情况叫”detached HEAD“
4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作） 分支是主线某个状态的一个复制,在不影响主线情况下, 可以有新的变化; git branch 分支名; git checkout 分支名;
5. 如何合并分支？git merge和git rebase的区别在哪里？（实际操作） git merge 分支名,git rebase 合并节点; git merges是将分支更改合并到一个新节点提交 git rabase是将分支更改移动到目标节点的顶部
6. 如何在Markdown格式的文本中使用标题、数字列表、无序列表和超链接？（实际操作） 标题用 # 数字列表用数字，例如1，2 无序列表用 * 超链接用 文本

实验总结

1.本次实验我学习git相关的命令和使用方法，如何复制别人和上传自己 仓库。 2.了解如何使用markdown，掌握了markdown的基本语法。