

Author Name

该文档原作者为 陆俊宇，实际参考时请忽略本行

一个有趣的可视化学习网站：<https://learngitbranching.js.org/>

VS Code中好用的git插件：

- GitLens: 可视化的git操作，免去命令行输入
- Git Graph: 仓库提交历史的可视化，可直接在分支节点上进行操作

安装部分略

SSH

- 生成ssh：

```
1 ssh-keygen -t ed25519 -C "username@example.com"
```

```
allen@LAPTOP-R5T3D05Q:/mnt/e/ssh$ ssh-keygen -t ed25519 -C "1967414827@qq.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/allen/.ssh/id_ed25519): _
```

- 添加到ssh-agent:

首先必须启动的ssh-agent:

```
1 eval `ssh-agent -s`
```

或

```
1 eval "$(ssh-agent -s)"
```

有趣的是，你不能直接输入命令

```
1 ssh-agent -s
```

因为：

SSH needs two things in order to use ssh-agent: an ssh-agent instance running in the background, and an environment variable set that tells SSH which socket it should use to connect to the agent (`SSH_AUTH_SOCK` IIRC). If you just run

`ssh-agent` then the agent will start, but SSH will have no idea where to find it.

也不能在eval后用'或"直接包裹命令：

```
1 eval "ssh-agent -s"
```

而必须使用反引号`，因为：

The command inside backquotes is executed in a new shell, and the output is sent to eval. If you execute ssh-agent, and copy/paste the output, then execute it, it will work as if you did use backquotes.

之后将之前生成的私钥添加到ssh-agent：

```
1 ssh-add '~/.ssh/id_ed25519'
```

即可。

- 添加到账户：

首先查看公钥：

```
1 cat ~/.ssh/id_ed25519.pub
```

```
allen@LAPTOP-R5T3D05Q:/mnt/e/ssh$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKsdbHywe3DRRPjbjfVZ15PDGAPV6025mezMfwouzJhr 1967414827@qq.com
allen@LAPTOP-R5T3D05Q:/mnt/e/ssh$
```

然后将显示内容粘贴到 github/gitee/whatever 的公钥设置项中：

SSH keys / Add new

Title

PC

Key

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKsdbHywe3DRRPjbjfVZ15PDGAPV6025mezMfwouzJhr  
1967414827@qq.com
```

Add SSH key

- 测试连接：

```
1 ssh -T git@example.com
```

在linux shell应该可以成功，因为ssh-agent在会话之初就将自启动：

```
allen@LAPTOP-R5T3D05Q:/mnt/c/Users/lenovo$ ssh -T git@github.com  
Hi AllenLuuu! You've successfully authenticated, but GitHub does not provide shell access.
```

在powershell/cmd则会失败：

```
E:\code\Web_Project\React\test>ssh -T git@github.com  
git@github.com: Permission denied (publickey).
```

需要进入

计算机管理 >

服务和应用程序 >

服务 将

OpenSSH Authentication Agent 设置为自动启动。

再次测试即可成功：

```
E:\code\Web_Project\React\test>ssh -T git@github.com  
Hi AllenLuuu! You've successfully authenticated, but GitHub does not provide shell access.
```

(注意：linux和windows的ssh key并不互通，切换系统时需要重新生成并再次添加到github账户和ssh-agent)

创建仓库

- 初始化仓库：

进入项目文件夹的根目录，使用：

```
1 git init
```

- 克隆仓库：

使用

```
1 git clone <repo url>
```

保存更改

- 添加到暂存区：

使用

```
1 git add <file/directory>
```

将更改添加到暂存区。

特别地：

```
1 git add -A
```

将暂存所有更改；

```
1 git add .
```

可将当前目录下所有更改放入暂存区；

```
1 git add -p
```

将提示你手动选择是否添加每一处更改。

- 提交

使用

```
1 git commit -m "message"
```

命令进行提交。每次提交必须有相应的描述。如省略-m参数，则将弹出文本框询问提交信息。

```
E:\code\Web_Project\React\test>git commit -m "first commit"
[master 71fb922] first commit
 8 files changed, 201 insertions(+), 120 deletions(-)
 delete mode 100644 src/App.css
 delete mode 100644 src/App.js
 delete mode 100644 src/App.test.js
 rewrite src/index.css (96%)
 rewrite src/index.js (82%)
 delete mode 100644 src/logo.svg
 delete mode 100644 src/reportWebVitals.js
 delete mode 100644 src/setupTests.js
```

附：git 提交风格指南：

<https://gist.github.com/robertpainsi/b632364184e70900af4ab688decf6f53>

若要修改最后一次提交，再次git add之后使用：

```
1 git commit --amend
```

- 比较差异

git diff命令可以比较更改前后文件的差异。只需运行：

```
1 git diff
```

即可看到输出：

```
diff --git a/src/setupTests.js b/src/setupTests.js
index 8f2609b..0000000
--- a/src/setupTests.js
+++ /dev/null
deleted file mode 100644
@@ -1,5 +0,0 @@
-// jest-dom adds custom jest matchers for asserting on DOM nodes.
-// allows you to do things like:
-// expect(element).toHaveTextContent(/react/i)
-// learn more: https://github.com/testing-library/jest-dom
-import '@testing-library/jest-dom';
```

输出的具体涵义可见：<https://www.cloudbees.com/blog/git-diff-a-complete-comparison-tutorial-for-git>

- 忽略文件

.gitignore文件可以忽略特定文件/文件夹，使其的更改不被追踪。

文件格式详见：<https://www.atlassian.com/git/tutorials/saving-changes/gitignore>


文件忽略规则详见：<https://www.liaoxuefeng.com/wiki/896043488029600/900004590234208>

远程同步

- 连接至远程仓库：


首先创建一个仓库或进入一个已有仓库，找到仓库链接：


 陆俊宇 / test

 代码

 Issues 0

 Pull Requests 0

 Wiki



快速设置— 如果你知道该怎么操作，直接使用下面的地址

HTTPS

SSH

git@gitee.com:Allen_Luuu/test.git



我们强烈建议所有的git仓库都有一个 `README` , `LICENSE` , `.gitignore` 文件

初始化 readme 文件

Git入门? 查看 帮助 , Visual Studio / TortoiseGit / Eclipse / Xcode 下如何连接本站, 如何导入仓库

复制后使用：

```
1 git remote add <name> <url>
```

即可

移除远程仓库可使用

```
1 git remote rm <name>
```

重命名可使用：

```
1 git remote rename <old-name> <new-name>
```

- 推送更改

使用

```
1 git push <remote> <branch>
```

命令向指定分支推送更改。

需要注意的是，每次在分支上推送本地更改时应首先拉取远程仓库的最新版本。

```
E:\code\Web_Project\React\test>git push origin master
The authenticity of host 'gitee.com (212.64.62.183)' can't be established.
ECDSA key fingerprint is SHA256:FQGC9Kn/eye1W8icdBgrOp+KkGYoFgbVr17bmjey0Wc.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added 'gitee.com,212.64.62.183' (ECDSA) to the list of known hosts.
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 16 threads
Compressing objects: 100% (27/27), done.
Writing objects: 100% (27/27), 287.80 KiB | 7.78 MiB/s, done.
Total 27 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Powered by GITEE.COM [GNK-6.3]
To gitee.com:Allen_Luuu/test.git
 * [new branch]      master -> master
```

之后即可在仓库看到推送的文件：

陆俊宇 first commit 79c2f6a 1小时前	2 次提交
public	Initialize project using Create React App 7天前
src	first commit 1小时前
.gitignore	Initialize project using Create React App 7天前
README.md	Initialize project using Create React App 7天前
package-lock.json	Initialize project using Create React App 7天前
package.json	Initialize project using Create React App 7天前

- 拉取内容

使用

```
1 git pull <remote>
```

拉取特定分支的最新版本。

检查仓库状态

- 列出暂存、未暂存和未跟踪的文件

使用

```
1 git status
```

即可。

- 查看提交历史

使用

```
1 git log
```

查看提交历史：


```
E:\code\Web_Project\React\test>git log
commit 79c2f6a3efac96c6868986fb82e7f4f37979c440 (HEAD -> master, origin/master)
Author: AllenLu <1967414827@qq.com>
Date: Thu May 19 14:55:19 2022 +0800

    first commit

commit 9b41e5a7f2684746f92ee991f63946511c63ee0f
Author: AllenLu <1967414827@qq.com>
Date: Thu May 12 19:21:36 2022 +0800

    Initialize project using Create React App
```

```
1 git log -n <limit>
```

可只显示limit次提交；

```
1 git log --author="<pattern>"
```

可显示特定作者的提交历史；pattern可为正则表达式；

```
1 git log --grep="<pattern>"
```

可显示含有匹配pattern的提交信息的提交；

```
1 git log <file>
```

可显示包含特定文件的提交。

分支

- 创建新分支

```
1 git branch <branch>
```

- 删除指定分支

```
1 git branch -d <branch>
```

如果该分支有未合并更改则不会执行。

- 强制删除指定分支

```
1 git branch -D <branch>
```

- 重命名当前分支

```
1 git branch -m <branch>
```

撤销更改

- git checkout

使用

```
1 git checkout <commit>
```

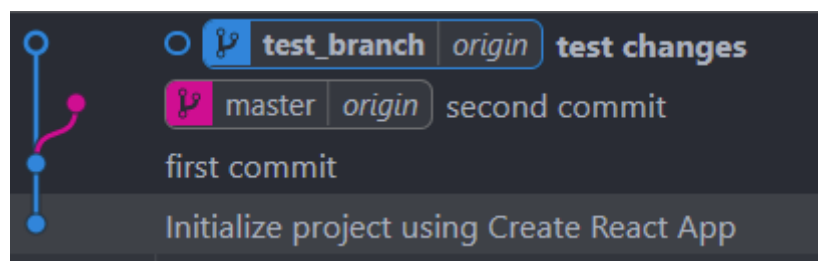
命令可移动到某一次特定的提交。

如果想要在此提交上继续进行，先使用

```
1 git switch -c <new-branch-name>
```

命令创建新的分支，并在此分支上进行新的工作。

效果如下：



- git revert

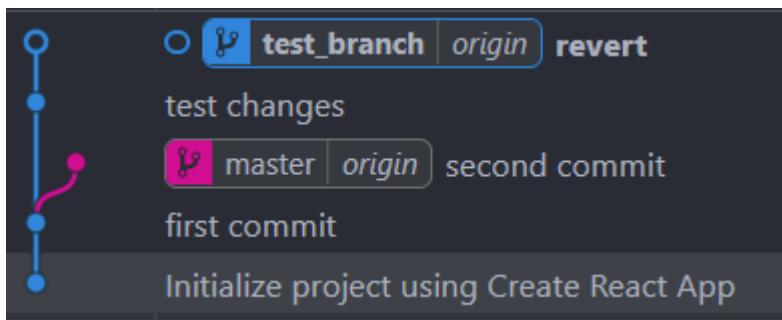
使用

```
1 git revert <commit>
```

来恢复某个指定的commit。

此命令并不会撤销指定commit之后的更改，而是新建一个提交，将代码恢复至指定commit处的形态。

执行后效果如下：

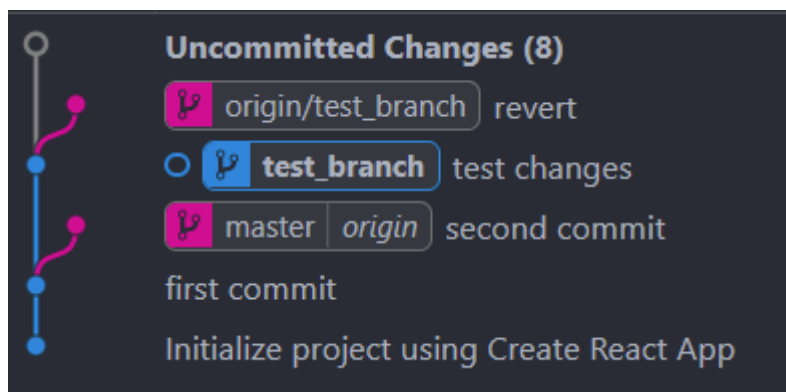


- git reset

与

`git revert` 不同，此命令将使分支指针指向之前的节点以完成撤销的效果。执行commit之后的节点仍可通过哈希值访问。

效果如下：



- git rm

使用

```
1 git rm <file>
```

命令可删除特定文件

合并分支

- merge

使用

```
1 git merge <branch>
```

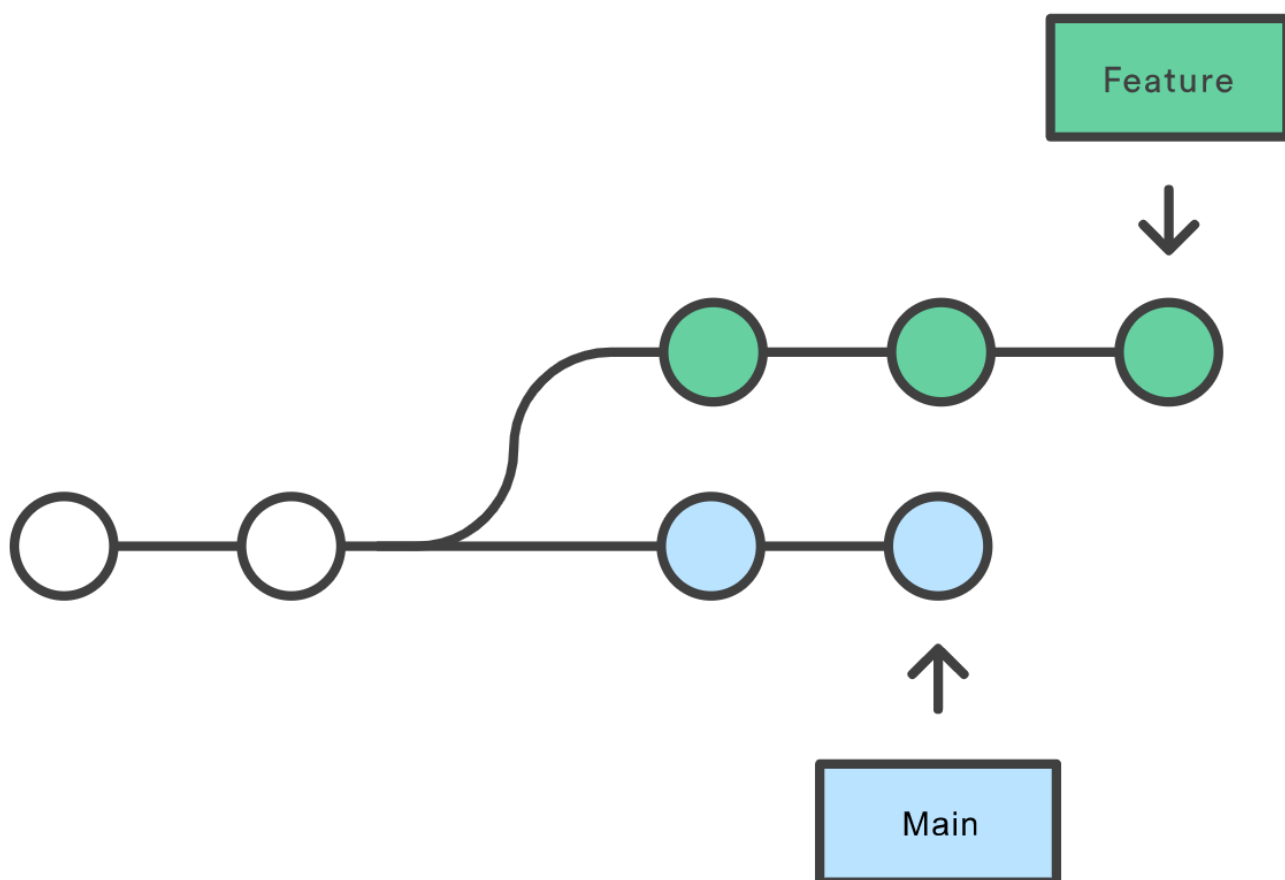
可使当前分支与branch分支合并，合并后当前分支会前进到合并后的节点，而branch分支不会。
也可使用

```
1 git merge <source> <target>
```

快速切换到source分支并合并target分支。

例如：

A forked commit history

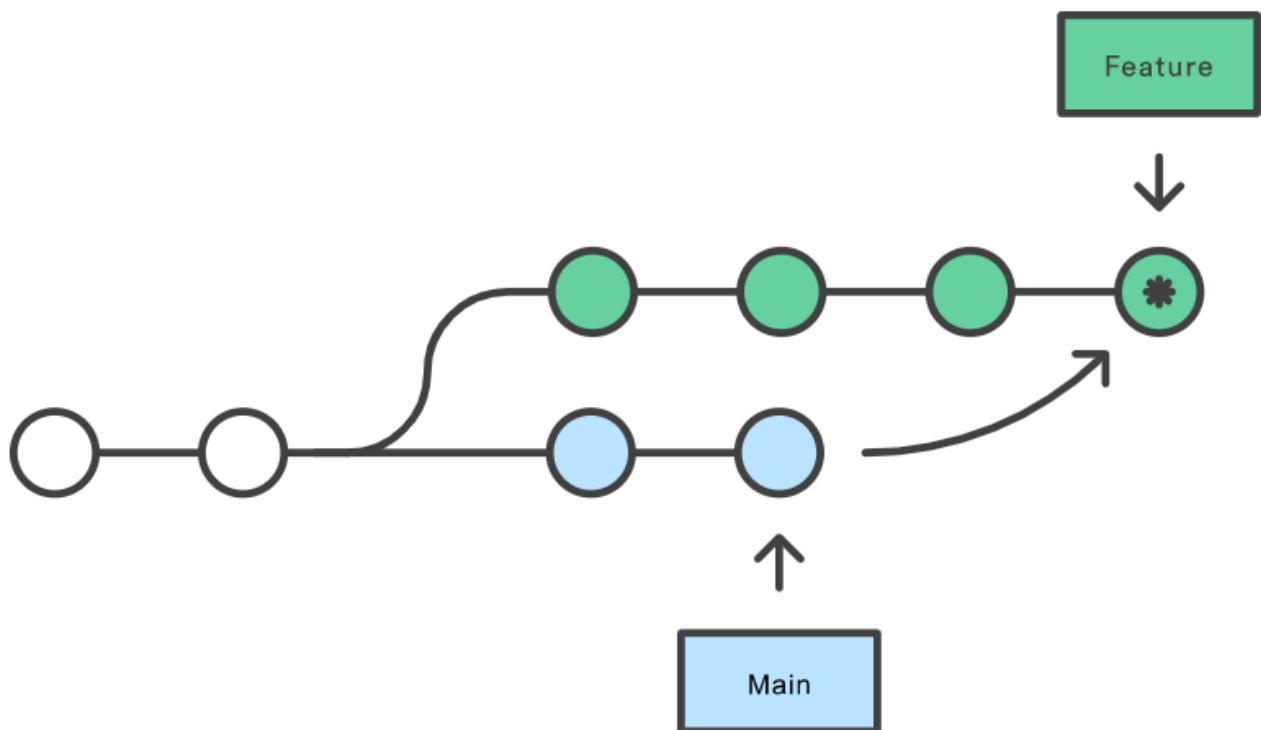


执行

```
1 git merge feature main
```

之后效果如下：

Merging main into the feature branch



- rebase

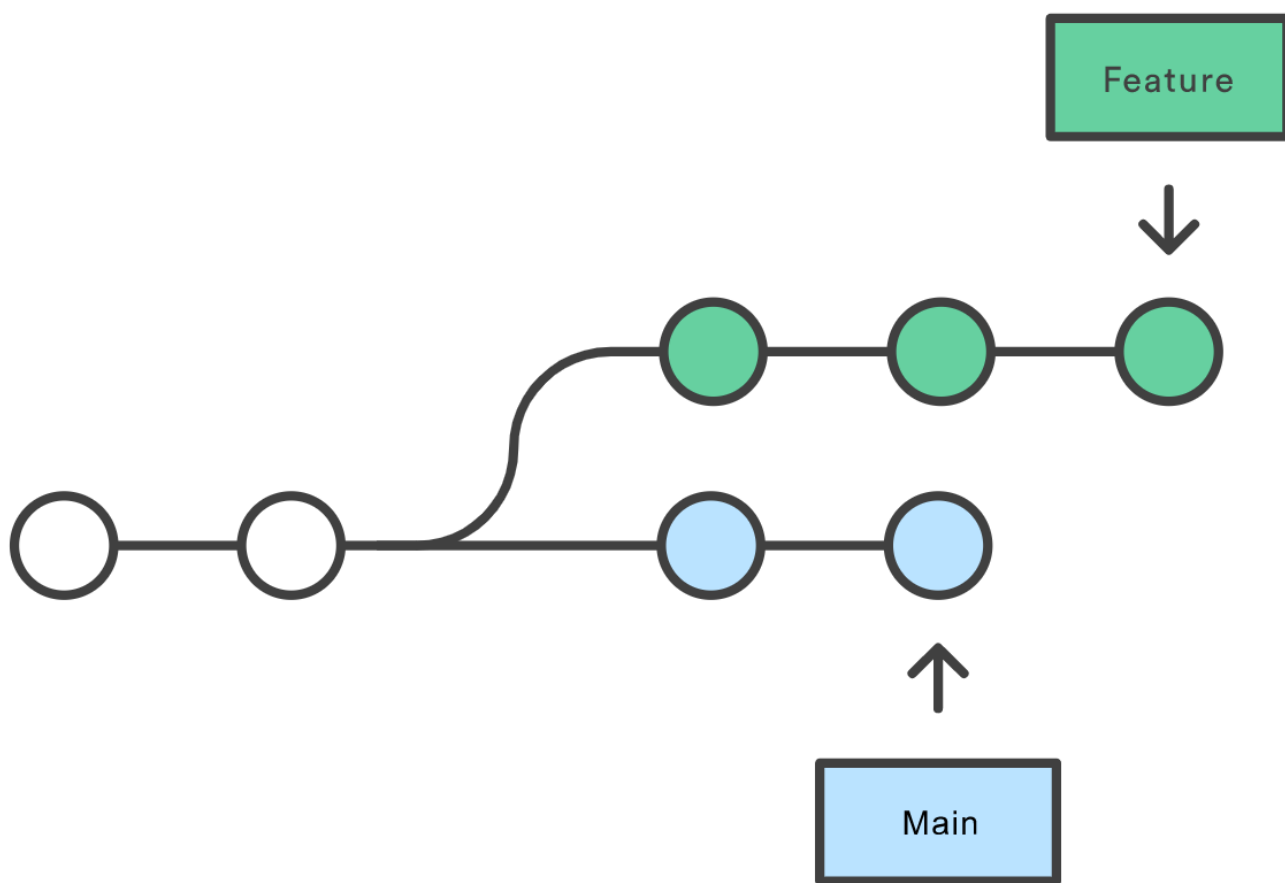
使用

```
1 git checkout <source>
2 git rebase <target>
```

将source分支的更改重现在主分支上。

对于

A forked commit history

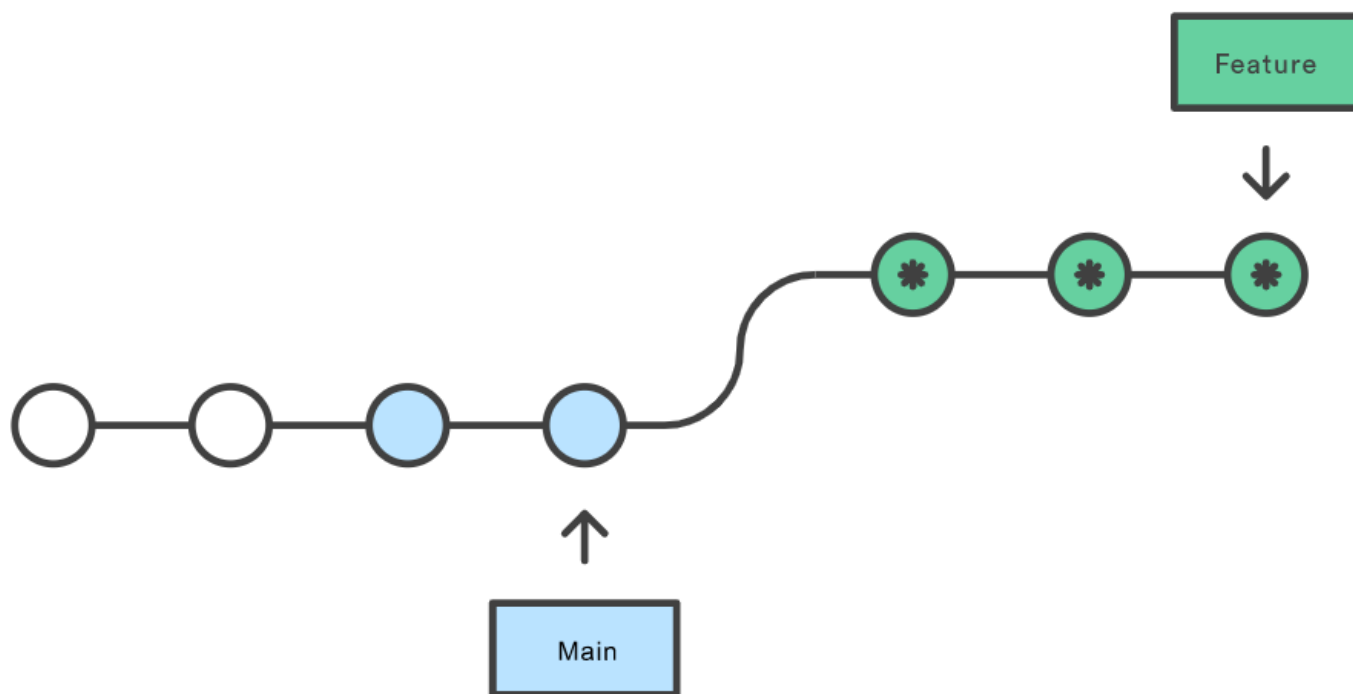


使用

- 1 `git checkout feature`
- 2 `git rebase main`

能够使提交历史变为如下形态：

Rebasing the feature branch onto main

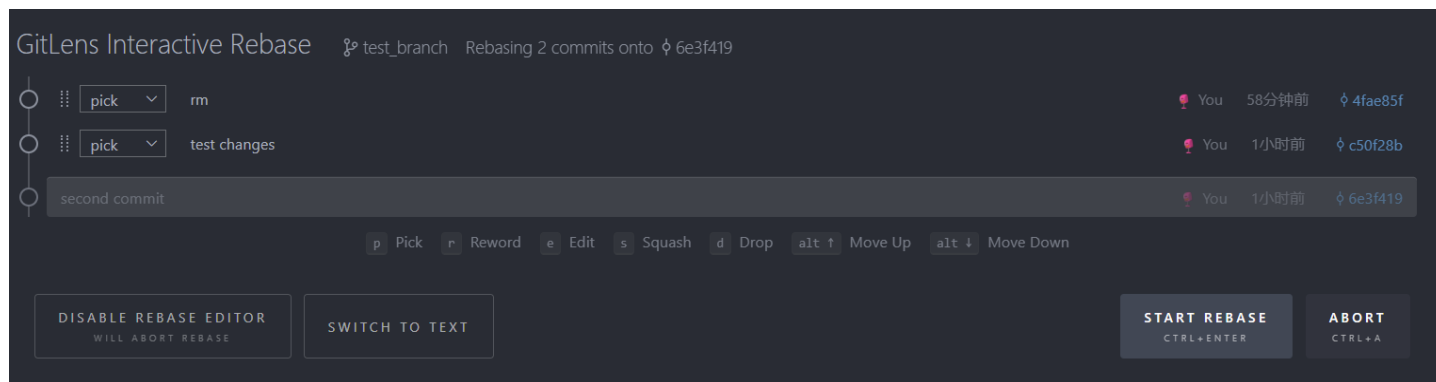


rebase的好处是能够保持**线性**的提交历史。

使用

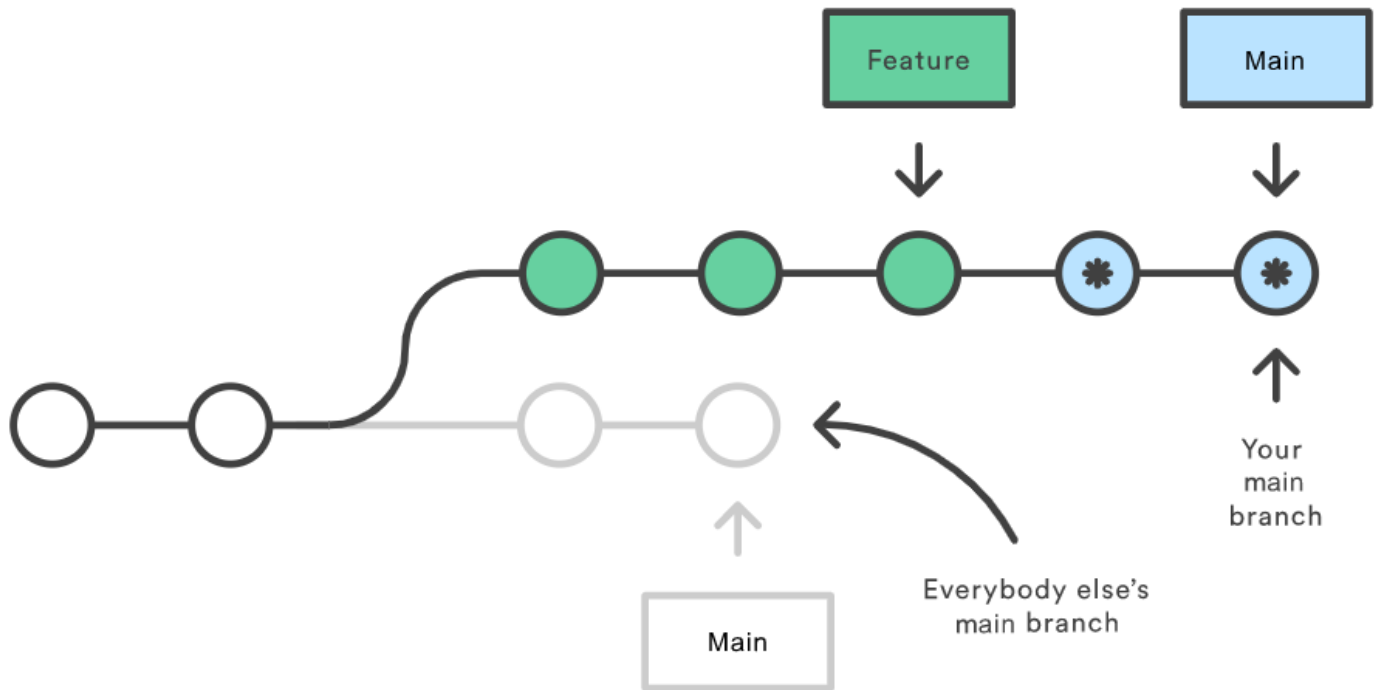
```
1 git rebase -i <branch>
```

命令将执行Interactive Rebasing，弹出对话框显示所有待提交的更改，并允许用户自行修改提交顺序以及合并提交：



注意，rebase的黄金法则是永远不要在主分支上使用rebase，否则将会发生：

Rebasing the main branch



- Cherry Pick

使用

```
1 git cherry-pick <commit1> <commit2> ...
```

可挑选任意提交放入当前的HEAD之后。

Pull Request

- Feature Branch Workflow With Pull Requests

在共享仓库创建功能分支，并提交Pull Request。

如下：

创建 Pull Request

源分支

Allen_Luuu/test

test_branch

→











目标分支

Allen_Luuu/test

master

不可自动合并

test

B I H          

test

之后由管理员审查并决定是否合并。

- Forking Workflow With Pull Requests

在个人仓库更新，并提交合并至主仓库的Pull Request。在个人仓库创建PR, 目标分支选择主仓库的相应分支即可。