



git

For Beginners

Chao Fang

ICAIS Lab, Nanjing University





Outline

- Git 的使用需求
- Git 的基本概念
- Git 的使用方法
 - 本地仓库
 - 远程仓库
- Git 其他学习资源



Outline

- Git 的使用需求
- Git 的基本概念
- Git 的使用方法
 - 本地仓库
 - 远程仓库
- Git 其他学习资源



Motivations

先从一个故事讲起 ...

- 1991年，Linus 创造了开源 Linux 操作系统；
- **2002年之前，Linus 手工合并世界各地开发者提交的 Linux 升级代码；**
- 2002-2005年之间，Linus 使用 BitKeeper 工具进行项目管理；
- 2005年，Linus 在一个月之内基于 C 语言开发了 Git 工具；
- 2008年，GitHub 上线，并成为了当前最流行的代码托管平台；

更多八卦，详见[这里](#)



Motivations

- Git — 快速、可拓展的**分布式**版本控制系统
- 它主要解决了以下两个需求
 - 版本控制
 - 多人协同

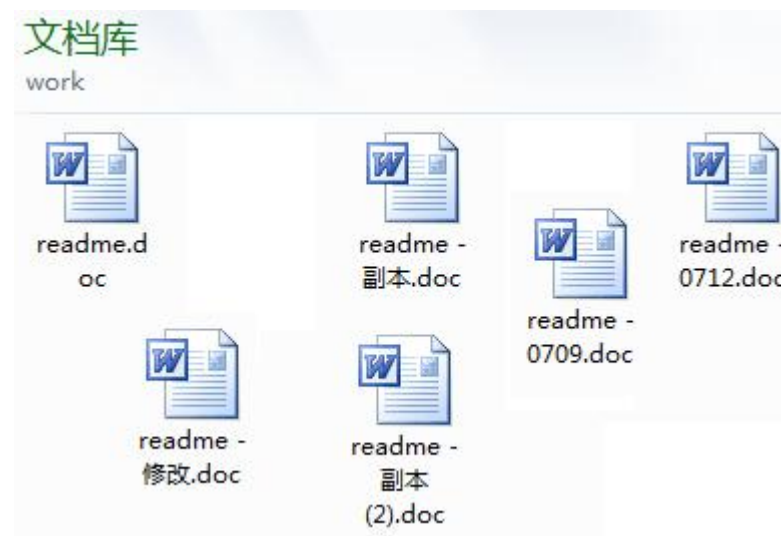
“分布式”意味着团队中的所有开发人员都拥有项目的完整版本。





Motivations

- 怎么理解 **版本控制**?
- 原始的版本控制如右图所示
 - 修改或者删除 readme.doc 的一个段落
 - 但是又担心旧的 readme.doc 文档可能有用
 - 那就全部都保留下来吧!
- 缺点
 - 手工管理
 - 保留文件过于冗余
- 能否有一个工具，自动记录每次文件改动?





Motivations

没有 Git 之前

Ver 1.0

Ver 1.2

Ver 2.0

Ver 3.0



起步开发，
还是个靓仔

开发三个月，
情绪不对劲

情绪逐渐失控

有了 Git 之后

- 跟踪项目的文件
- 记录项目文件中的更改（新建/修改/删除）
- 还原项目中文件的先前版本
- 代码比较和代码分析
- 合并来自不同计算机和不同团队成员的代码

一日靓仔，终身靓仔

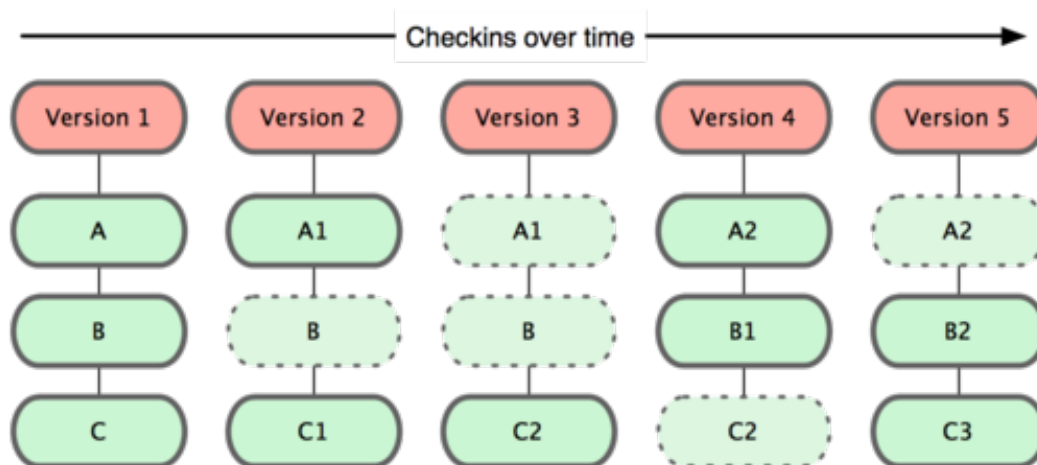
Ver 1.0 **Ver 1.2** **Ver 2.0** **Ver 3.0**





Motivations

- Git 如何实现 版本控制?





Outline

- Git 的使用需求
- **Git 的基本概念**
- Git 的使用方法
 - 本地仓库
 - 远程仓库
- Git 其他学习资源



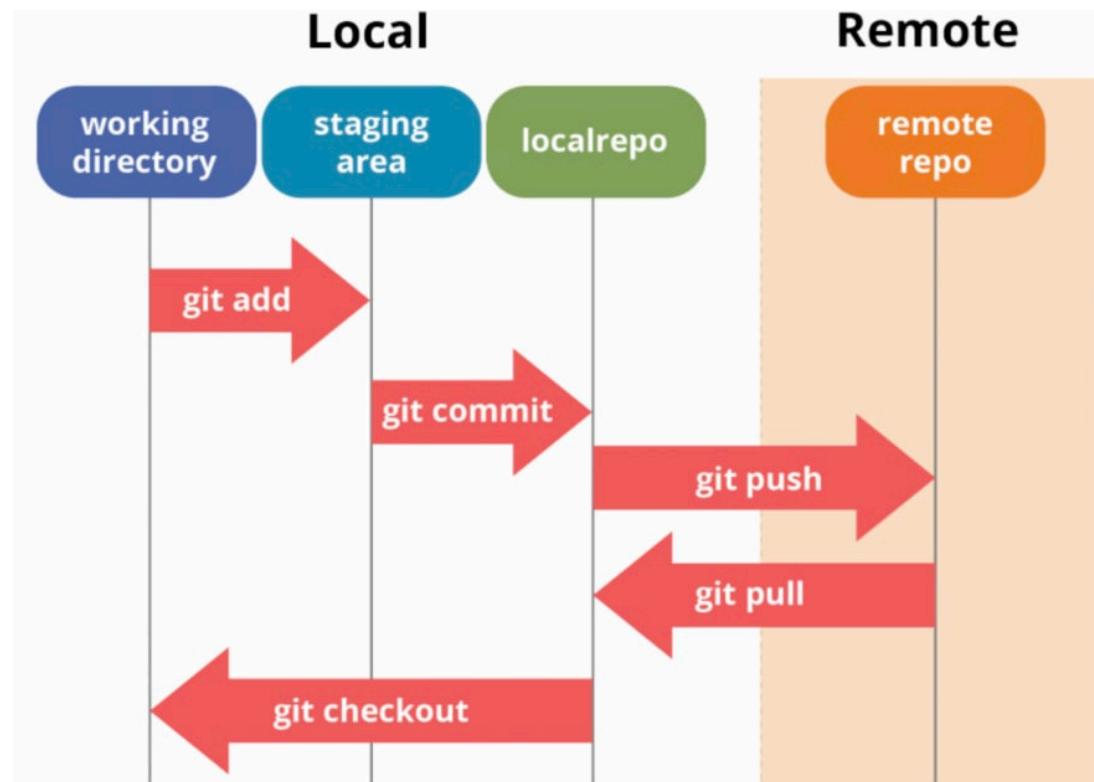
Basic Concepts

- Git 的基本概念

- 工作区 (working directory)
- 暂存区 (staging area, index)
- 本地仓库 (local repo)
- 远程仓库 (remote repo)

- 高频命令

- \$ git add
- \$ git commit
- \$ git push
- \$ git pull
- \$ git checkout





Outline

- Git 的使用需求
- Git 的基本概念
- **Git 的使用方法**
 - 本地仓库
 - 远程仓库
- Git 其他学习资源

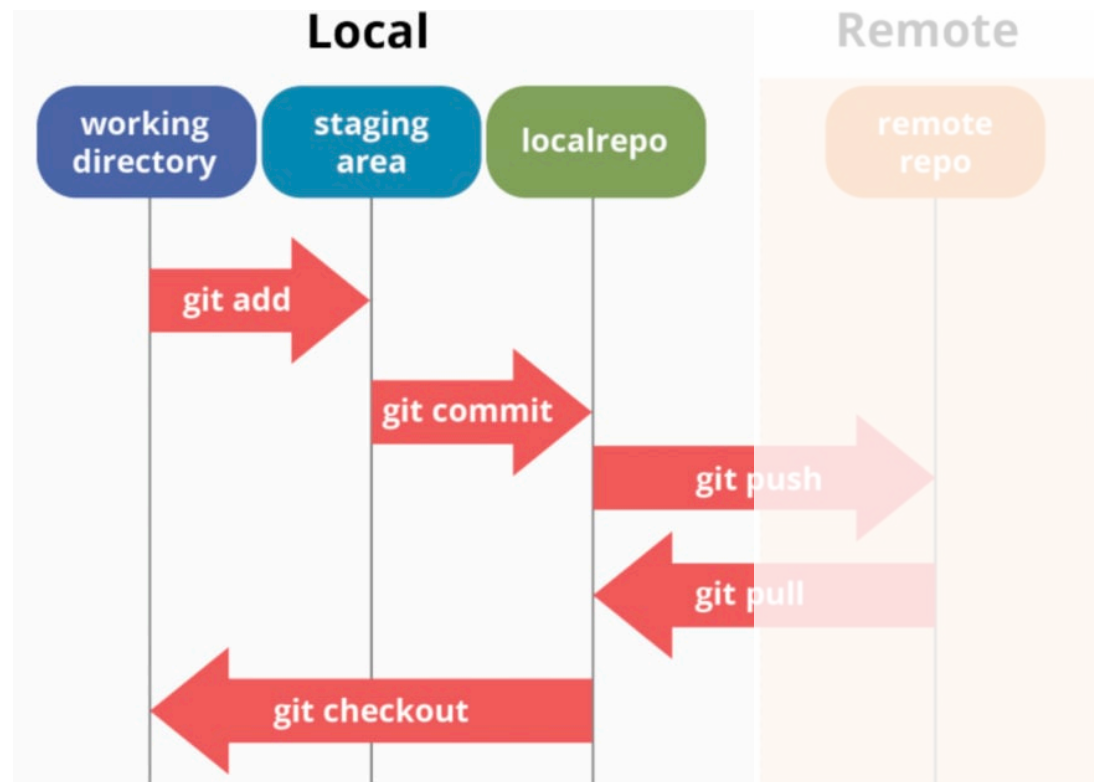
Local Repo

- Git 的基本概念

- 工作区 (working directory)
- 暂存区 (staging area, index)
- 本地仓库 (local repo)
- 远程仓库 (remote repo)

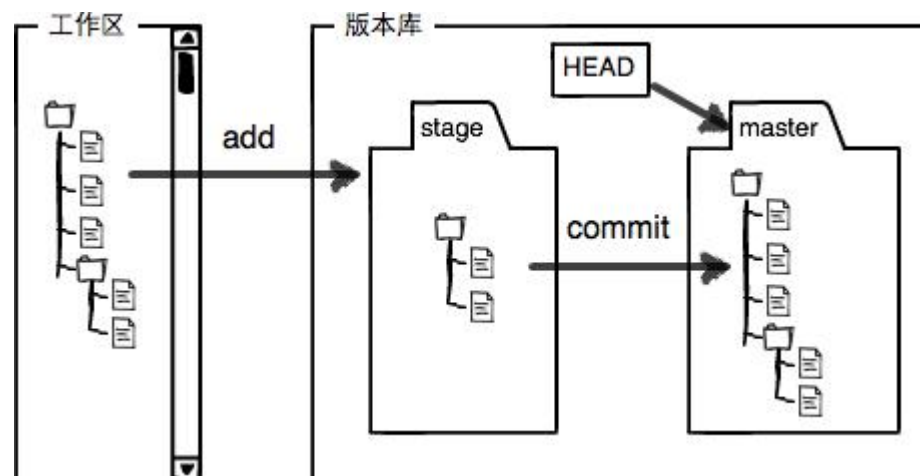
- 高频命令

- `$ git add`
- `$ git commit`
- `$ git push`
- `$ git pull`
- `$ git checkout`



Local Repo

- 一个完整的本地仓库工作流程
 - 在根目录或子目录中创建新文件，或更新现有文件。
 - 对修改后的文件进行快照，然后保存到版本库的暂存区 (stage)
 - 提交更新 (commit)，将保存在暂存区域的文件快照永久转储到 Git 目录中
- 举一个醋溜砖头菜谱的栗子





• 接下来先讨论本地仓库操作

- 配置 Git
- 创建新的本地仓库
- 使用 Git 暂存项目文件
- 向 Git 本地仓库提交阶段更改
- Git 仓库版本回退
- 放弃对项目内特定文件的追踪
- 编写 .gitignore 文件



Configuring Git

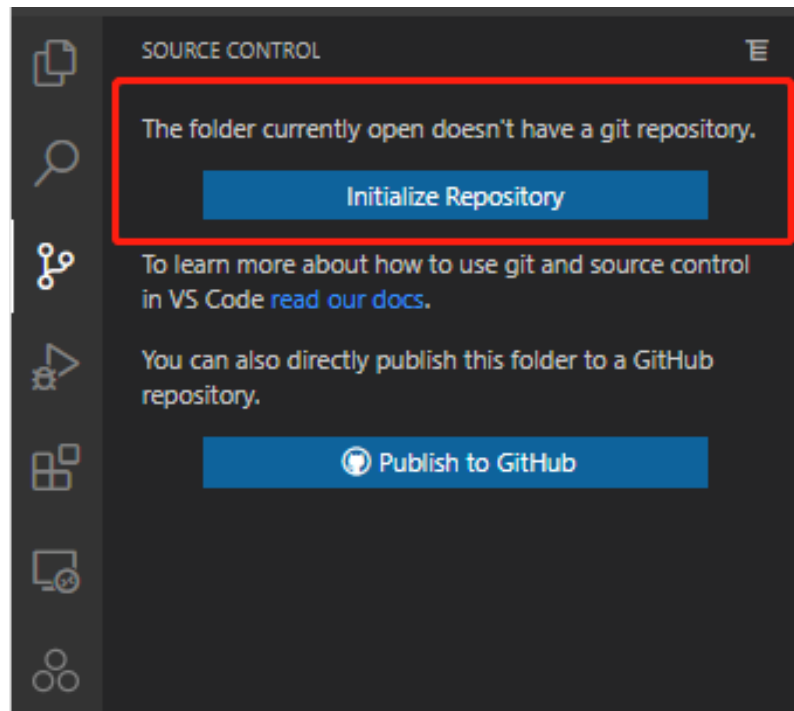
- 配置 Git 本地用户信息
 - `$ git config --global user.name "YOUR NAME"`
 - `$ git config --global user.email "YOUR EMAIL"`
- 配置 Git 高亮文本
 - `$ git config --global color.ui true`
- 查看 Git 配置
 - `$ git config --list`



Starting a New Local Repository with Git

- 初始化 Git 仓库
 - `$ git init`
- 查看仓库状态
 - `$ git status`

在项目文件夹下得到了
.git 隐藏文件夹。
这个就是你的 Git 仓库。





Staging Files with Git

- 将修改文件放入暂存区

- `$ git add <my_new_file>` # 将单个文件放入暂存区
- `$ git add .` # 添加当前目录的所有文件到暂存区
- `$ git add -all` # 把所有修改过的文件放入暂存区

- 将修改文件移出暂存区

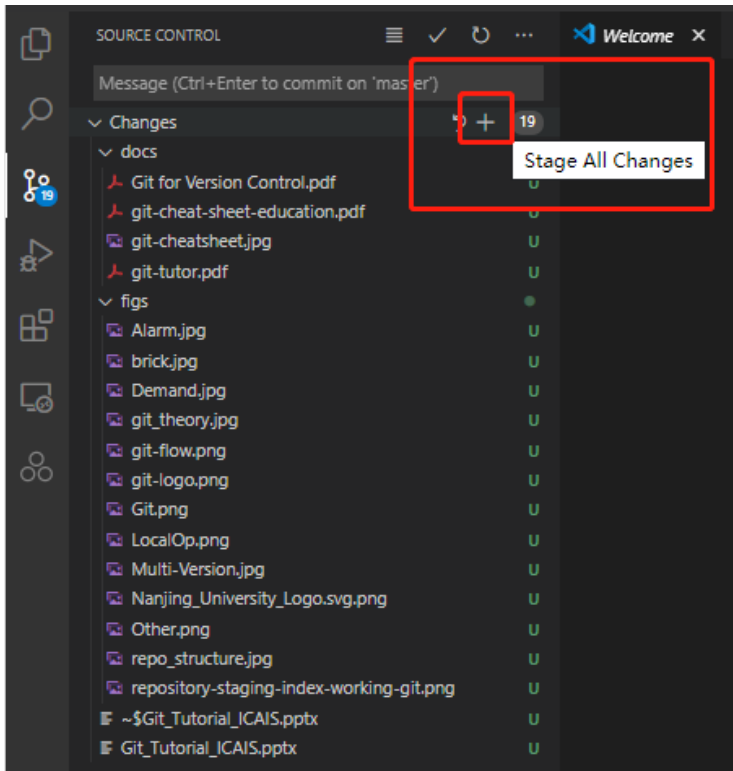
- `$ git rm --cached <my_file>` # --cached 选项，停止追踪文件，但该文件保留在工作区
- `$ git reset <my_file>` # 等价于上一条命令
- `$ git rm <my_file>` # 删除工作区文件，并且将这次删除放入暂存区

- 不妨将暂存区看成一个购物车

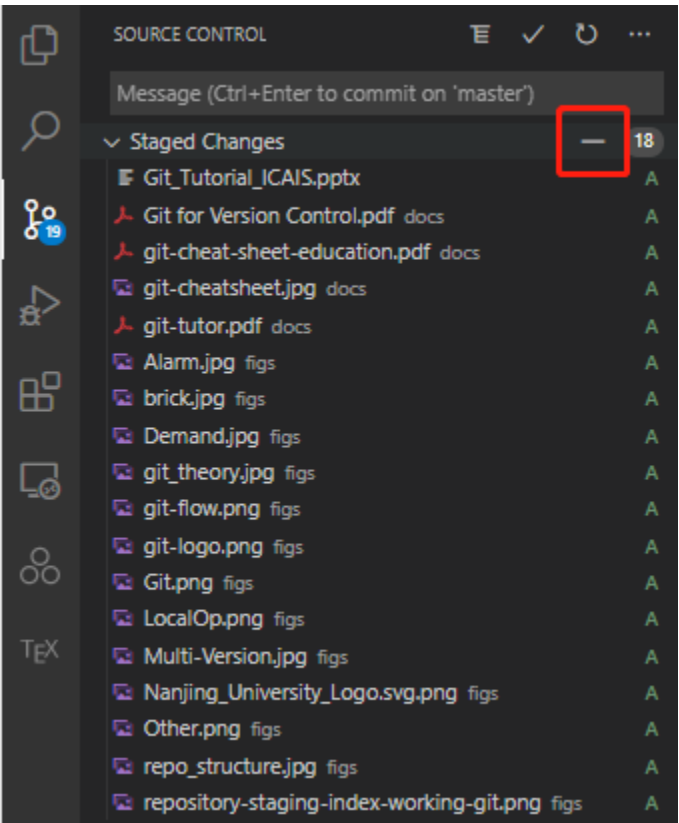


Staging Files with Git

将修改文件放入暂存区 (VS Code)



将修改文件移出暂存区 (VS Code)





Committing Changes to Git

- 完成一次提交

- `$ git commit -m "<Commit Message>"` # 把暂存区文件在本地仓库更新
- `$ git commit -a -m "<Commit Message>"` # 把所有修改文件提交到暂存区并提交
- 注：Commit Message 非常重要。这将是版本回退的重要依据。请按照规范填写。详见[这里](#)。

- 查看提交记录

- `$ git log`

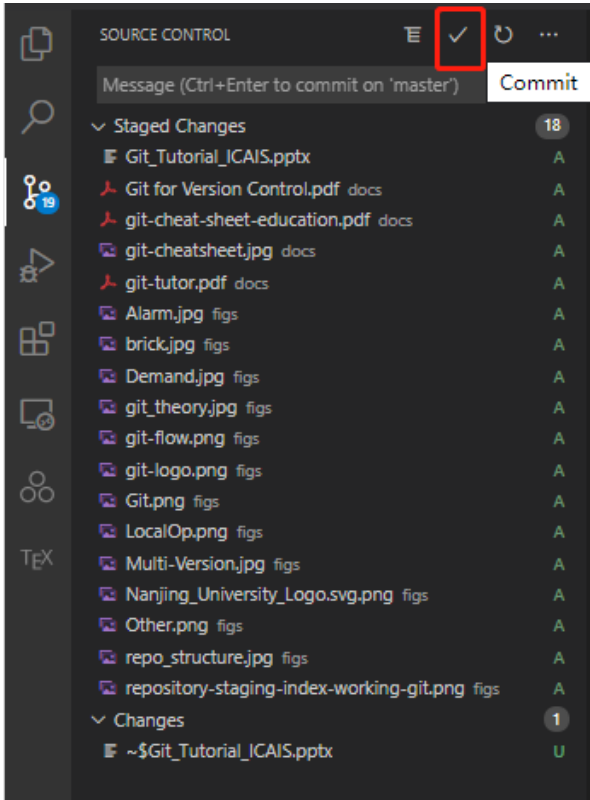
- 撤销提交

- `$ git reset --soft HEAD^` # 撤销上一次提交，^ 代表上一次提交
- `$ git reset --soft <HASH_ID>` # 回退到该 HASH_ID 记录之前，所有文件恢复至提交到 HASH_ID 时的状态

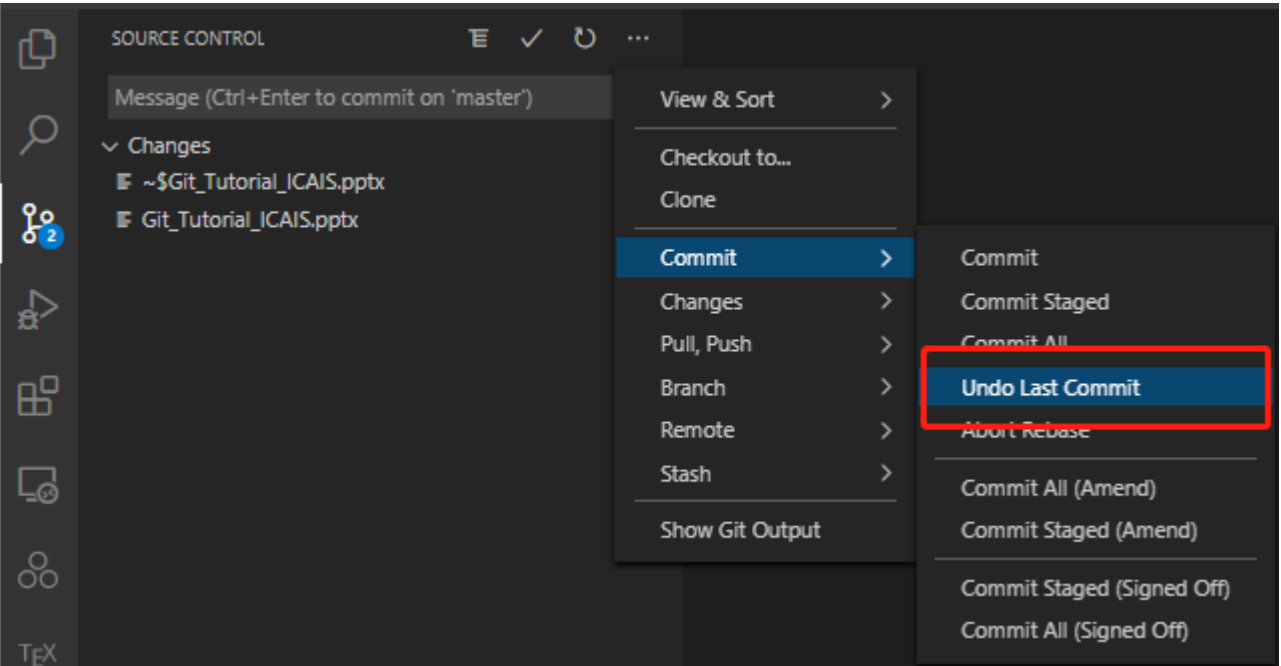


Committing Changes to Git

完成一次提交



撤销一次提交



- 我懒得写了，建议大家直接读[这里](#)。



情绪逐渐失控



Outline

- Git 的使用需求
- Git 的基本概念
- **Git 的使用方法**
 - 本地仓库
 - 远程仓库
- Git 其他学习资源



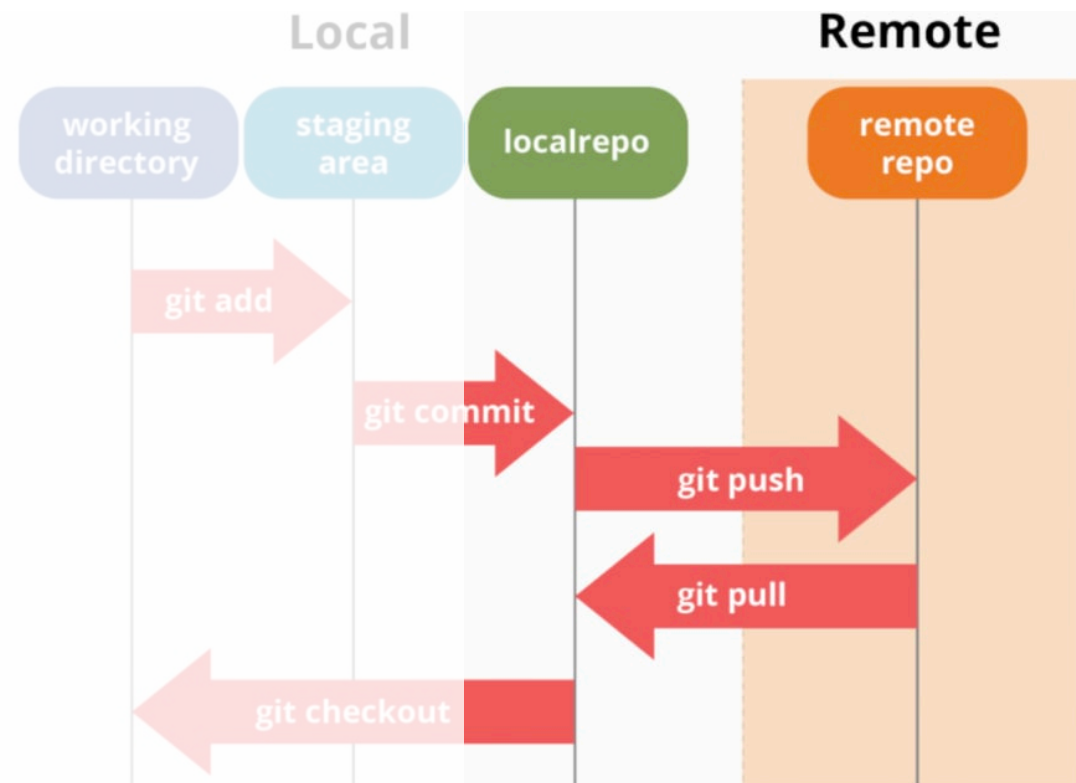
Remote Repo

- Git 的基本概念

- 工作区 (working directory)
- 暂存区 (staging area, index)
- 本地仓库 (local repo)
- 远程仓库 (remote repo)

- 高频命令

- \$ git add
- \$ git commit
- \$ git push
- \$ git pull
- \$ git checkout





- 一个完整的远程协作流程
 - **本地仓库**: 处理一个特性并将文件提交到一个分支(主分支或任何其他分支); 【上一章节已详细叙述该过程。】
 - **本地 -> 远端**: 将历史提交推向(push)远程存储库;
 - **远端 -> 本地**: 其他开发人员将你的提交拉入(pull)到他们的计算机上, 得到该项目的最新版本;



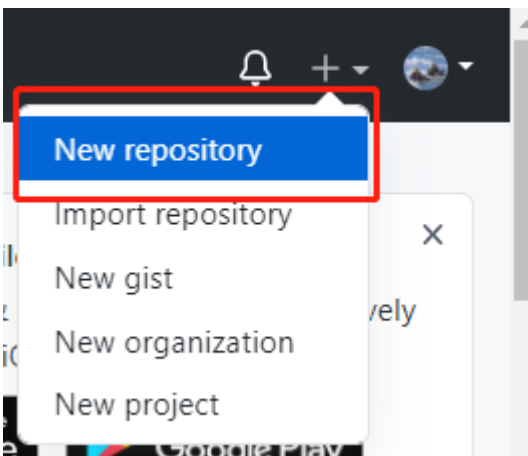
• 最后讨论本地仓库与远端仓库交互的操作

- 创建远程仓库 (GitHub、Gitlab、Gitee)
- 将远程仓库与本地仓库绑定
- 将本地仓库代码更新推至 (push) 远程仓库
- 将远程仓库更新拉入本地 (pull) 仓库
- 克隆 (clone) 远程仓库
- clone 和 pull 的区别




Push and Pull To and From a Remote Repository

• 创建远程仓库




Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner  / Repository name ✓

Great repository names are short and memorable. Need inspiration? How about [bookish-broccoli](#)?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.


☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☒ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  **master** as the default branch. Change the default name in your [settings](#).



Push and Pull

To and From a Remote Repository

- 将远程仓库与本地仓库绑定
 - `$ git remote add origin <remote_repo_HTTP_Link>` # 基于 HTTP 协议
 - `$ git remote add origin <remote_repo_SSH_Link>` # 基于 SSH 协议
- 给远程仓库取一个别名，一般用 `origin`，本地仓库通过别名识别远程仓库
- 基于 HTTP 协议，每次 `push` 都需要输入用户账号和密码
- 基于 SSH 协议，设置好 SSH 公钥 后每次 `push` 无须输入账号密码（具体[详见](#)）



Push and Pull

To and From a Remote Repository

- 将本地仓库代码更新推至 (push) 远程仓库
 - `$ git push <alias_remote_repo> <local_repo_branch>`
 - `$ git push -u origin master` # -u 意味着下次直接敲 `git push` 即可
 - `$ git push --set-upstream origin master` # 等价于上一命令
- 将远程仓库更新拉入本地 (pull) 仓库
 - `$ git pull`
 - `$ git fetch; git merge` # 等价于上一命令
 - `$ git pull --allow-unrelated-histories` # 合并两个不相关的项目



Push and Pull

To and From a Remote Repository

- 克隆 (clone) 远程仓库

- \$ git clone <link>
- \$ git clone <link> <local_proj_name> # 可以给本地仓库取个喜欢的名字



Push and Pull

To and From a Remote Repository

- clone 和 pull 的区别
 - 当 clone 一个远程仓库，Git 将
 - 下载整个项目到指定目录中
 - 在该目录下创建一个名为origin的远程仓库，并将其指向传入的URL
 - 当运行 pull 命令时，Git 将
 - 拉出当前分支中的远程更改到本地的 origin 远程仓库 (git fetch)
 - 把 origin 远程仓库的更新合并至本地仓库 (git merge)



Outline

- Git 的使用需求
- Git 的基本概念
- Git 的使用方法
 - 本地仓库
 - 远程仓库
- **Git 其他学习资源**



Resources

- <https://guides.github.com/activities/hello-world/>
 - GitHub 提供的一个简短概述
- <https://try.github.io/>
 - GitHub 提供的一个在线交互教程
- <https://www.ruanyifeng.com/blog/2015/12/git-cheat-sheet.html>
 - 阮一峰老师总结的 Git Cheat Sheet
- <https://git-scm.com/docs/gittutorial>
- <http://www.vogella.com/tutorials/Git/article.html>
 - 以上两个均是非常详细（长）的教程



Resources

- <https://oschina.gitee.io/learn-git-branching/>
 - 开源中国提供的在线傻瓜教程
- <https://rubygarage.org/blog/most-basic-git-commands-with-examples>
 - 非常好的 Git 短教程
- <https://gitee.com/all-about-git>
 - Gitee 提供的 Git 大全
- **Docs内的其他补充文档也是极好的**

Takeaways

- 介绍了 Git 工具的使用需求
- 介绍了 Git 的基本概念
 - working directory, staging area/index, local repo, remote repo
- 介绍了 Git 的使用方法
 - \$ git add, git commit, git push, git pull, git clone



Takeaways

- 介绍了 Git 工具的使用需求
- 介绍了 Git 的基本概念
 - working directory, staging area/index, local repo, remote repo
- 介绍了 Git 的使用方法
 - \$ git add, git commit, git push, git pull, git clone

• **Easy!**

Fin.

