

# Git简介

周君栋

2022.7.13

# 内容

- 版本控制
- Git基本概念
- Git指令

# 版本控制

- 什么是版本控制？
- 为什么要版本控制？

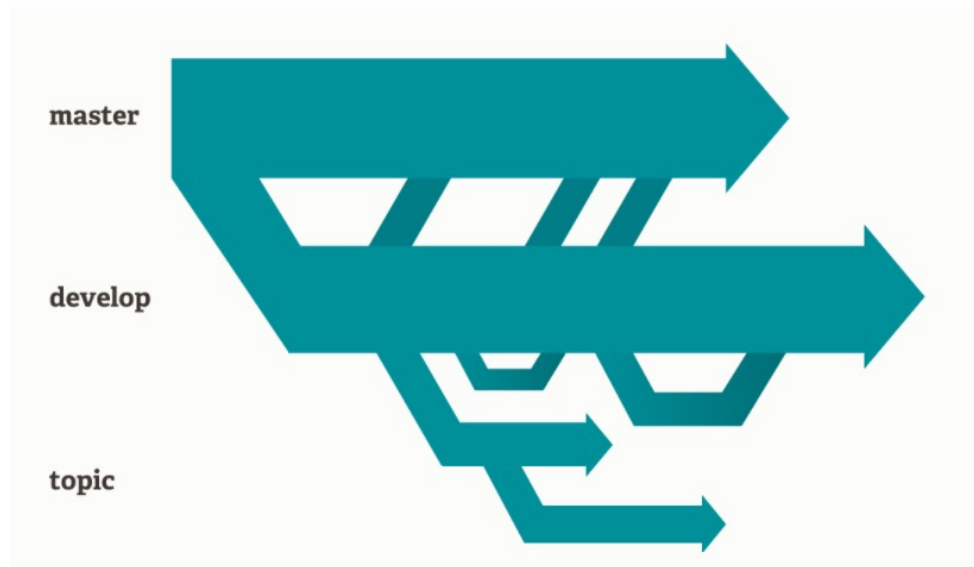
# 开发困境

- E.g. 论文 **version1.7.2**……version 3.2.1——版本记录
- E.g. 代码加功能，改着改着之前的功能出问题——代码回退
- E.g. 多人协作大作业，功能A+功能B=Nothing——合并提交与解决冲突
- E.g. 大型商业项目，线上稳定版本VS内部开发版——分支管理

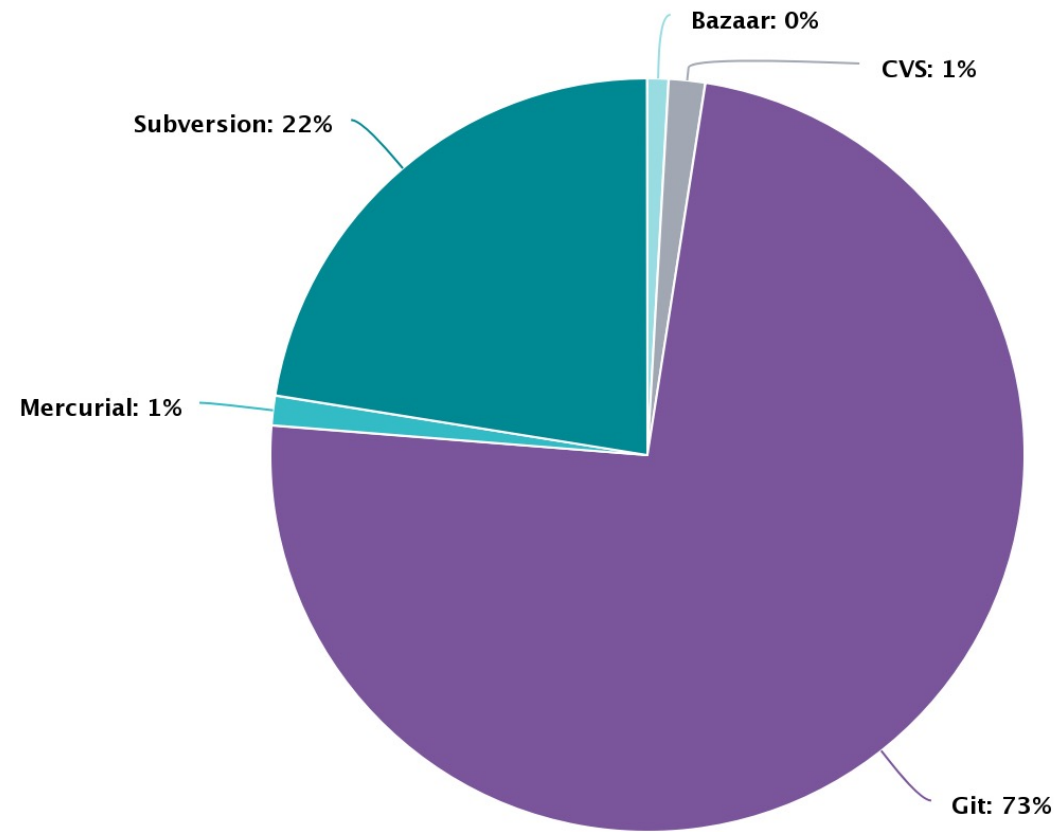
<input type="checkbox"/>			自然辩证法大作业v6.docx
<input type="checkbox"/>			自然辩证法大作业v5.docx
<input type="checkbox"/>			自然辩证法大作业v4.docx
<input type="checkbox"/>			自然辩证法大作业v3.docx
<input type="checkbox"/>			自然辩证法大作业v2.docx
<input type="checkbox"/>			自然辩证法大作业v1.docx

# Git是什么？

- Git是一个分布式版本控制软件（类似的工具还有SVN等）
- Linus Torvalds



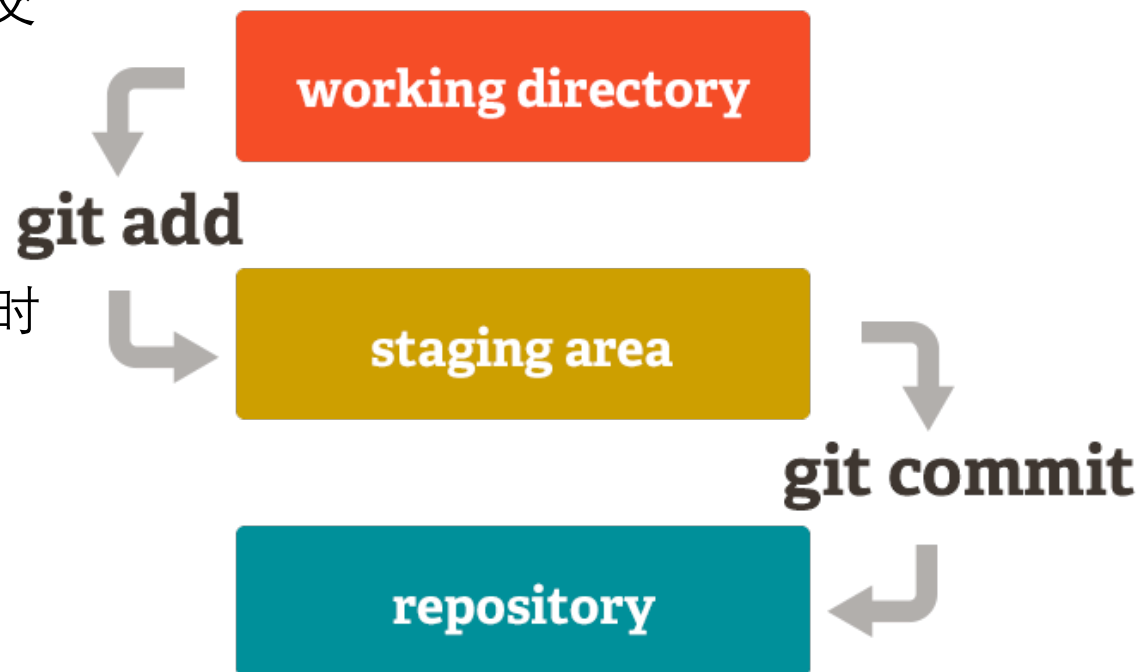
# Why Git



[Compare Repositories - Open Hub](#)

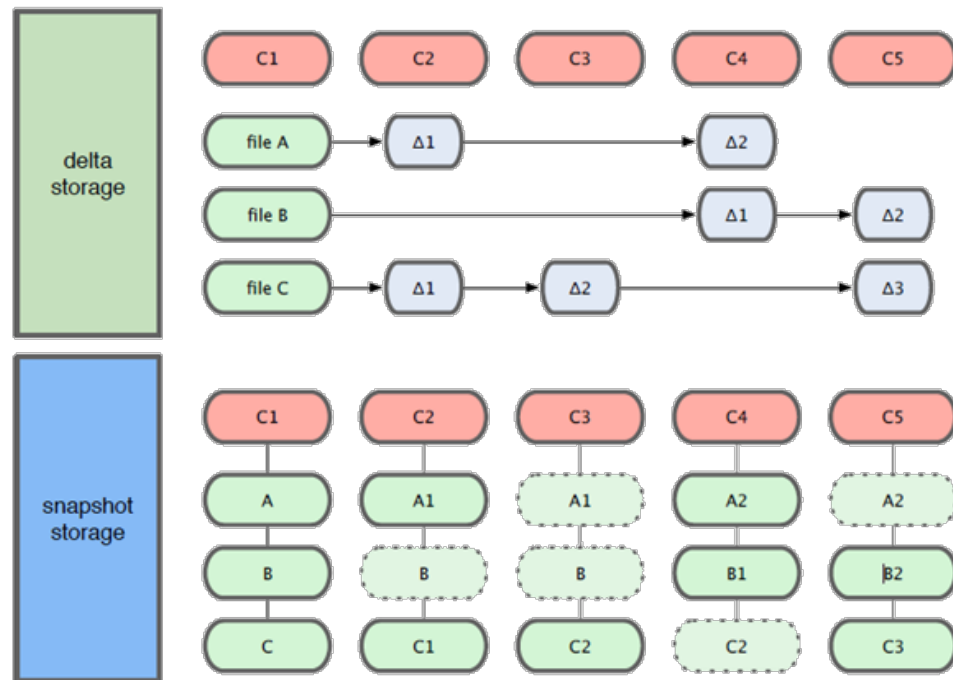
# Git基本概念

- 工作区(Workspace)-与文件系统保持一致
- 暂存区(Index/Stage)-告诉Git哪些修改要被正式提交到记录中
- 本地仓库(Repository)-Git存档的提交记录内容
- 远程(Remote)-位于远端服务器的存储仓库
- 仓库中的正式提交-拥有唯一的hash值, 可在任意时间点回退
- .git隐藏目录-存储相关信息的Git版本库



# Git基本概念

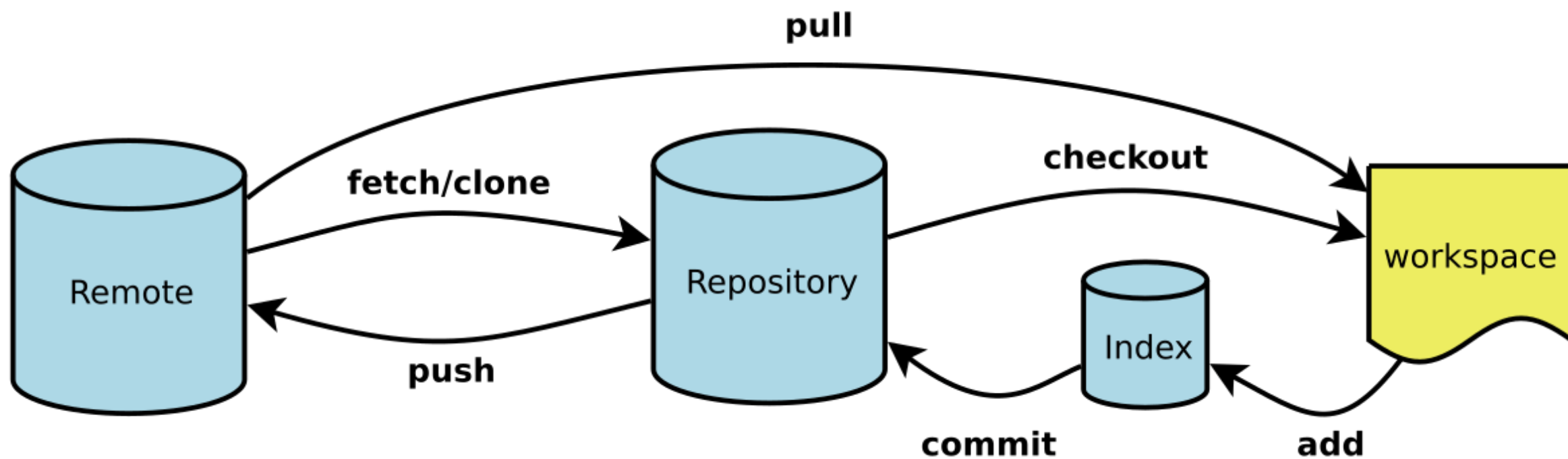
- 快照记录变动



- 以行为单位对文件进行比对及合并→对二进制文件的支持较差



# Git基本操作

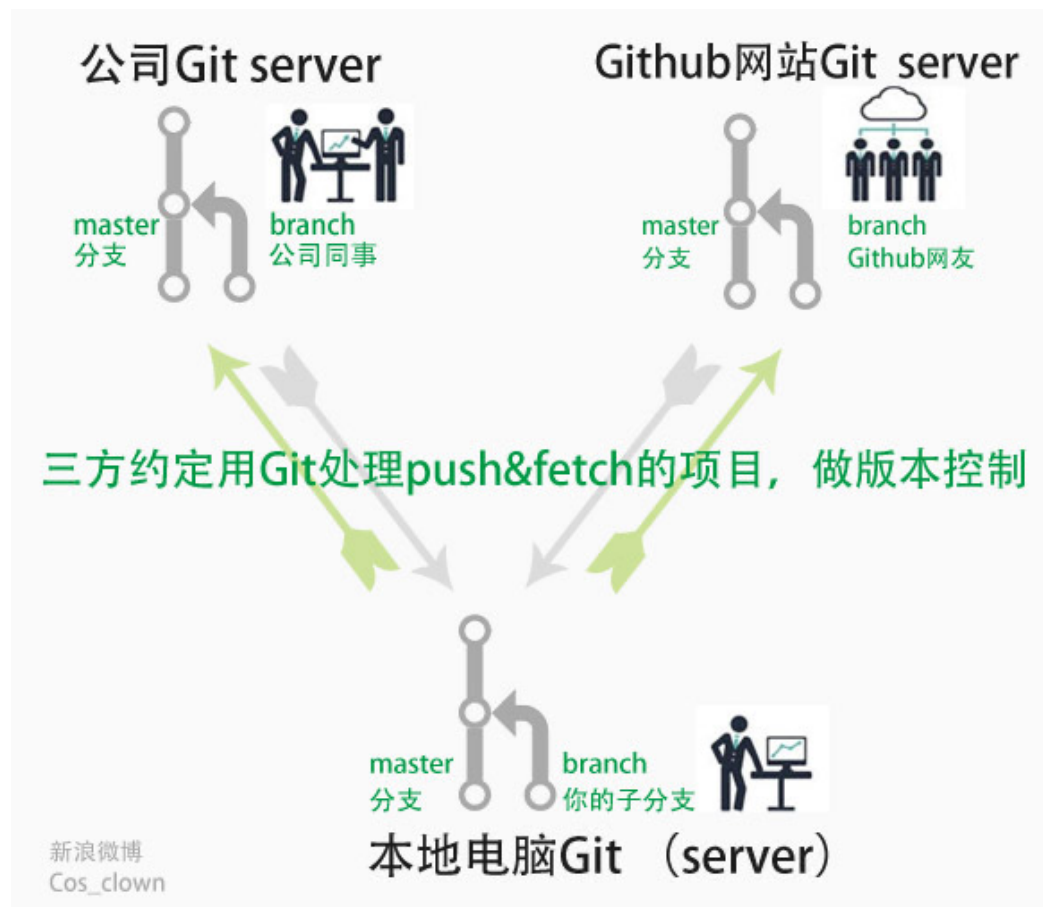


# GitHub&GitLab是什么？

- 底层通过Git进行版本控制的代码托管平台
- ~~GitHub 全球最大交友网站~~
- Git与GitHub的关系：
  - Git 是一套命令行工具，只能在本地使用，无法与他人协作
  - GitHub 依托 Git，充当协作中转站
- 常用 Git 开源平台：
  - GitHub (<https://github.com/>)
  - THU GitLab (<https://git.tsinghua.edu.cn/>)
  - Gitee 码云 (<https://gitee.com/>)

# GitHub&GitLab是什么？

- 理论上也可以自己通过Git搭建内部Server




# GitHub&GitLab 使用示例

- 远端新建仓库


Github

---

Language ▼ Sort ▼  New

---

GitLab

 New project

---

Name ▼

---

---

# GitHub&GitLab 使用示例

- 远端新建仓库

## Create new project



### Create blank project

Create a blank project to house your files, plan your work, and collaborate on code, among other things.



### Create from template

Create a project pre-populated with the necessary files to get you started quickly.



### Import project

Migrate your data from an external source like GitHub, Bitbucket, or another instance of GitLab.

## Create a new repository

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

Owner \*

HugoAhoy ▾

Repository name \*

test



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

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.](#)

.gitignore template: None ▾

### Choose a license

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

License: None ▾


You are creating a public repository in your personal account.

Create repository

# Github&GitLab 使用示例

- 关联本地仓库

**Quick setup — if you've done this kind of thing before**


 Set up in Desktop

 or 

HTTPS

SSH


git@github.com:HugoAhoy/test.git



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:HugoAhoy/test.git
git push -u origin main
```

**...or push an existing repository from the command line****...or import code from another repository**

Import code

# Git 新建仓库

- 从本地新建仓库
  - `git init`
  - `git init [project-name]`
- 从远端克隆一个仓库
  - `git clone [url]`
- 禁术！！！！
  - `rm -rf .git && git init`

# Git 添加文件

- 添加文件到暂存区
  - `git add [file1] [file2] ...`
  - `git add [dir]`
  - `git add .`
- 查看仓库状态
  - `git status`



# Git 提交文件

- 提交文件到仓库
  - `git commit -m [message]`
  - `git commit [file1] [file2] ... -m [message]`
  - `git commit -a -m [message]`
- 重做 commit
  - `git commit --amend [file1] [file2] ... -m [message]`

# Git 配置

- 基础配置 (用户名&邮箱)
  - `git config --global user.name "Your Name Comes Here"`
  - `git config --global user.email you@yourdomain.example.com`
- 显示当前的Git配置
  - `git config --list`
- 编辑Git配置文件
  - `git config -e [--global]`

# SSH Authentication

- Windows: C:\用户\用户名\.ssh (可通过Git Bash访问)
- MacOS: /Users/用户名/.ssh
- Linux: /home/用户名/.ssh
- ssh-keygen 命令
- 公钥(默认id\_rsa.pub)内容复制到GitHub&GitLab账户设置 (Setting/Profile → SSH)
- ssh -T git@github.com

```
➤ ssh -T git@github.com  
Hi Username! You've successfully authenticated, but GitHub does not provide shell access.
```

# Git 仓库同步

- 远端设置

- `git remote -v`
- `git remote add [shortname] [url]`

- 推送到远端

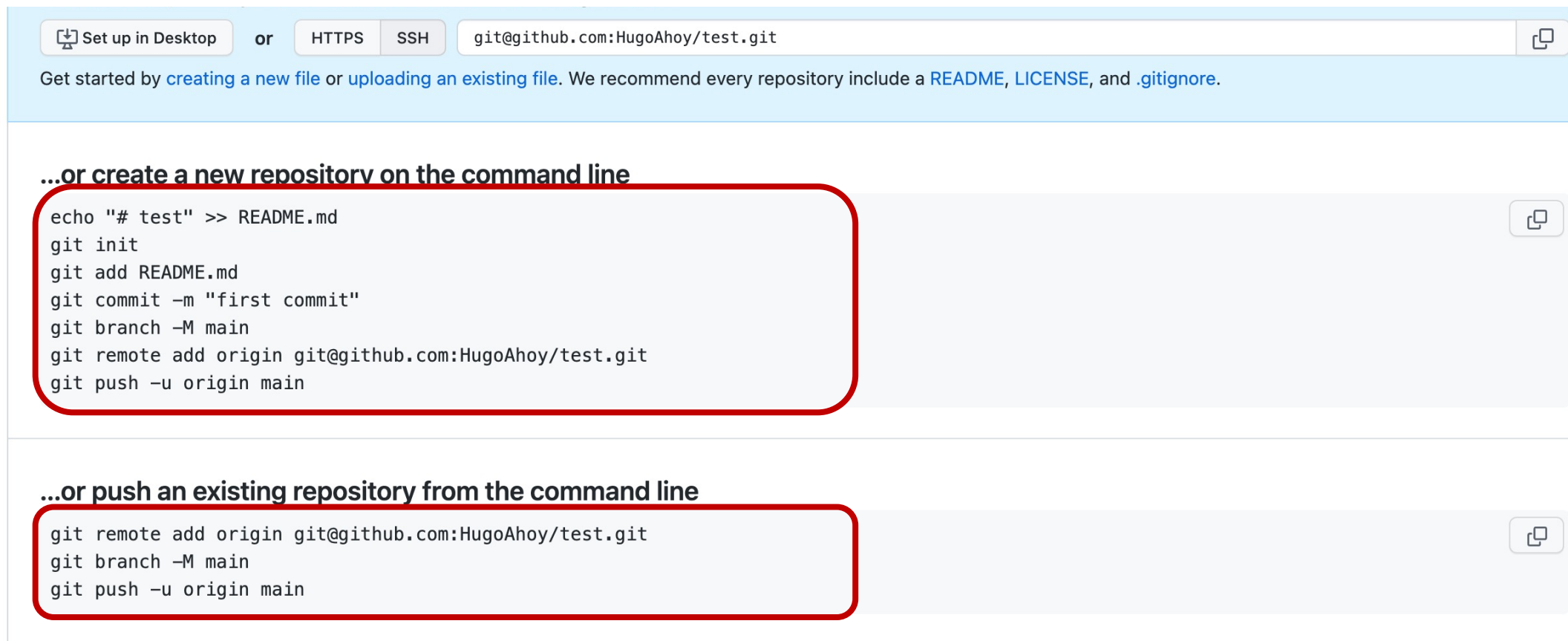
- `git push [remote] [local_branch]:[remote_branch]`
- `git push [remote] --force` ⚠

- 拉取到本地

- `git pull [remote] [remote_branch]`
- `<=>`
- `git fetch && git merge [remote]/[remote_branch]`

# Github&GitLab 使用示例Review

- 关联本地仓库



The screenshot shows the GitHub 'New repository' page. At the top, there are buttons for 'Set up in Desktop', 'HTTPS', and 'SSH', followed by a text input field containing 'git@github.com:HugoAhoy/test.git'. Below this, a light blue banner contains the text: 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).' The main content area has two sections, each with a title and a list of commands in a light gray box with a red border. The first section is titled '...or create a new repository on the command line' and contains the following commands: 

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:HugoAhoy/test.git
git push -u origin main
```

 The second section is titled '...or push an existing repository from the command line' and contains the following commands: 

```
git remote add origin git@github.com:HugoAhoy/test.git
git branch -M main
git push -u origin main
```

Set up in Desktop or HTTPS SSH git@github.com:HugoAhoy/test.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:HugoAhoy/test.git
git push -u origin main
```

**...or push an existing repository from the command line**

```
git remote add origin git@github.com:HugoAhoy/test.git
git branch -M main
git push -u origin main
```

# Github&GitLab 使用示例

- 如何在GitHub/GitLab进行代码审核以及合并
  - Github: Pull Request
  - GitLab: Merge Request

# Git分支管理

- 初始默认生成一个 main/master 分支
- `git branch` : 列出所有本地分支及当前所处的分支
- `git branch -r` : 列出所有远程分支
- `git branch -a` : 列出所有分支
- `git branch [branch-name]` : 创建一个新分支但保留在当前分支
- `git branch -d [branch-name]` : 删除名为branch-name的分支

# Git分支管理

- `git checkout [branch-name]` : 切换到 [branch-name]分支
- `git checkout -b [branch-name]` : 新建并切换到 [branch-name]分支
- `git checkout -b [local_branch] [remote]/[remote_branch]` : 新建分支并追踪远程分支
- `git merge [branch-name]` : 将[branch-name]分支合并进当前分支



# Git解决冲突

- pull和merge操作中有可能触发系统不能自动解决的冲突，系统会将冲突的位置进行标识，需要人为解决
- 用下列字符串进行标识
  - <<<<<<<
  - =====
  - >>>>>>>

```
_config.yml
1 # Site settings
2 title: "Jekyll Resume Template"
3 description: "A resume template for Jekyll and GitHub Pages sites."
4
5 # Build settings
6 markdown: kramdown
7 sass:
8   sass_dir: _sass
9   style: compressed
10
11 # Resume settings
12 resume_avatar: "true"
13 <<<<<<< HEAD
14 resume_name: "GitHub Teacher"
15 resume_title: "Professional Trainer"
16 resume_contact_email: "services@github.com"
17 resume_contact_telephone: "1-(877)-448-4820"
18 resume_contact_address: "San Francisco, California"
19 resume_header_contact_info: "San Francisco, California | 1-(877)-448-4820 | services@github.com"
20 resume_header_intro: "<p>Charting the knowledge of the Internet, just like Galileo charted the stars.</p>"
21 =====
22 resume_name: "Surftocat"
23 resume_title: "Professional Surfer"
24 resume_contact_email: "surftocat@github.com"
25 resume_contact_telephone: "1-(877)-448-4820"
26 resume_contact_address: "San Diego, California"
27 resume_header_contact_info: "San Diego, California | 1-(877)-448-4820 | surftocat@github.com"
28 resume_header_intro: "<p>Professional Surfer, 'nuff said.</p>"
29 >>>>>>> origin/master
30
31 # use "yes" to display the email contact button,
32 # "no" to display an "I'm not looking for work" message,
```

<conflicts/\_config.yml CWD: /Users/kingname/merge-conflicts Line: 2 Column: 1

# 其他常用操作

- `git checkout -- <file>`: 撤销对file在工作区的修改, 回到上一次commit或add的状态
- `git reset <commit_id>`: 回退到commit\_id标识的版本
- `git log`: 查看所有历史提交信息 (包括commit\_id、备注信息、时间、提交人员等)
- `git diff`: 查显示暂存区和工作区的差异
- `git diff --shortstat "@{0 day ago}"`: 查看今天代码量
- `git revert <commit_id>`
- `git cherry-pick <commit_id>`
- `git blame [file]`
- `git rebase`

.....

# Commit id

- **HEAD**: 当前分支的最后一个版本, 即最后一个 commit ( $\Leftrightarrow$  **HEAD~0**)
- **HEAD^**: 当前分支上一个版本 ( $\Leftrightarrow$  **HEAD~1**)
- **HEAD^^**: 当前分支倒数第三个版本 ( $\Leftrightarrow$  **HEAD~2**)

以此类推

\* **git log** 中的hash也是commit\_id/commitHash, 包括short hash 和完整hash

# .gitignore

# Byte-compiled / optimized / DLL files

\_\_pycache\_\_/

\*.py[cod]

\*\$py.class

# C extensions

\*.so

# Distribution / packaging

.Python

build/

develop-eggs/

dist/

[github/gitignore](https://github.com/gitignore): A collection of useful .gitignore templates

# 相关资料

- [Git - Book \(git-scm.com\)](http://git-scm.com)
- [Git 原理入门 - 阮一峰的网络日志 \(ruanyifeng.com\)](http://ruanyifeng.com)
- [Git \(git-scm.com\)](http://git-scm.com)