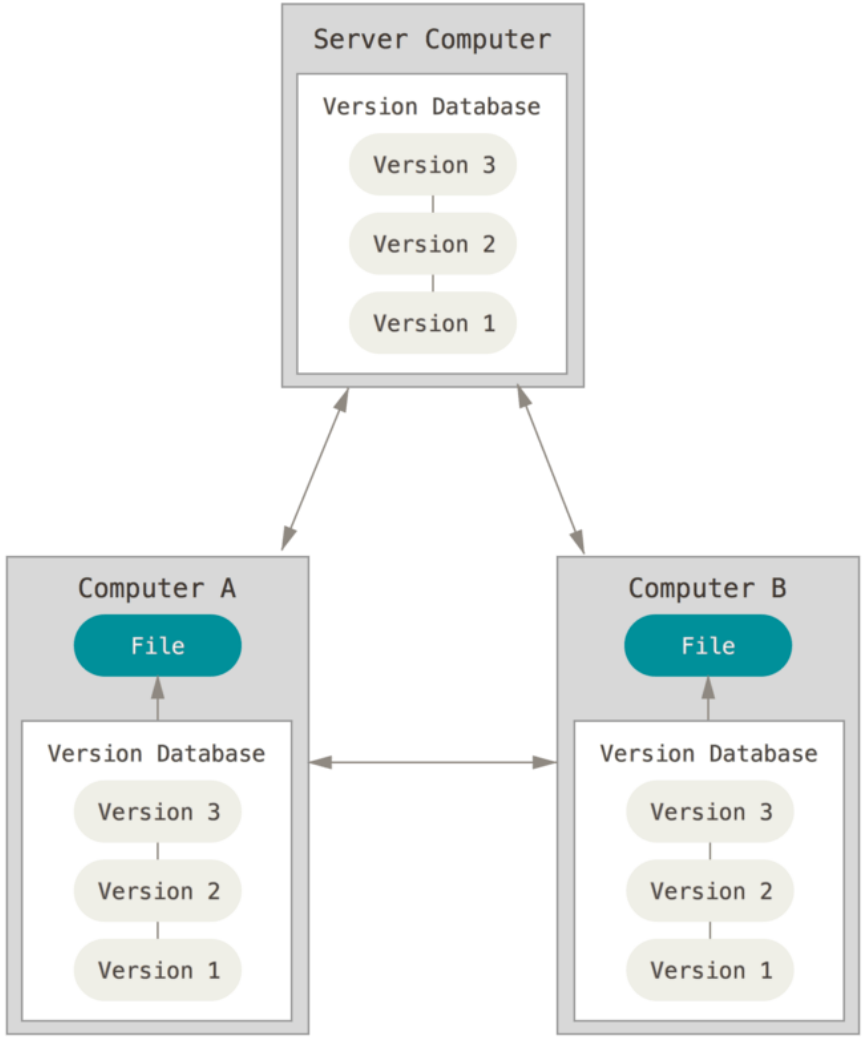
Git使用入门

陈俊达
北京大学计算中心
2022/1/8

目录

Git是什么？用来干什么？

- Git：分布式版本控制系统
- **版本**：每完成一个任务后的文件状态
- **Commit**：一次文件修改，产生一个新版本
- **版本控制**：记录所有版本，以供将来查阅
- **分布式**：每个仓库都有项目所有历史版本
- 作用
  - **组织修改**：一个版本=一个功能/bugfix
  - **多人合作**：记录、合并每个人的修改



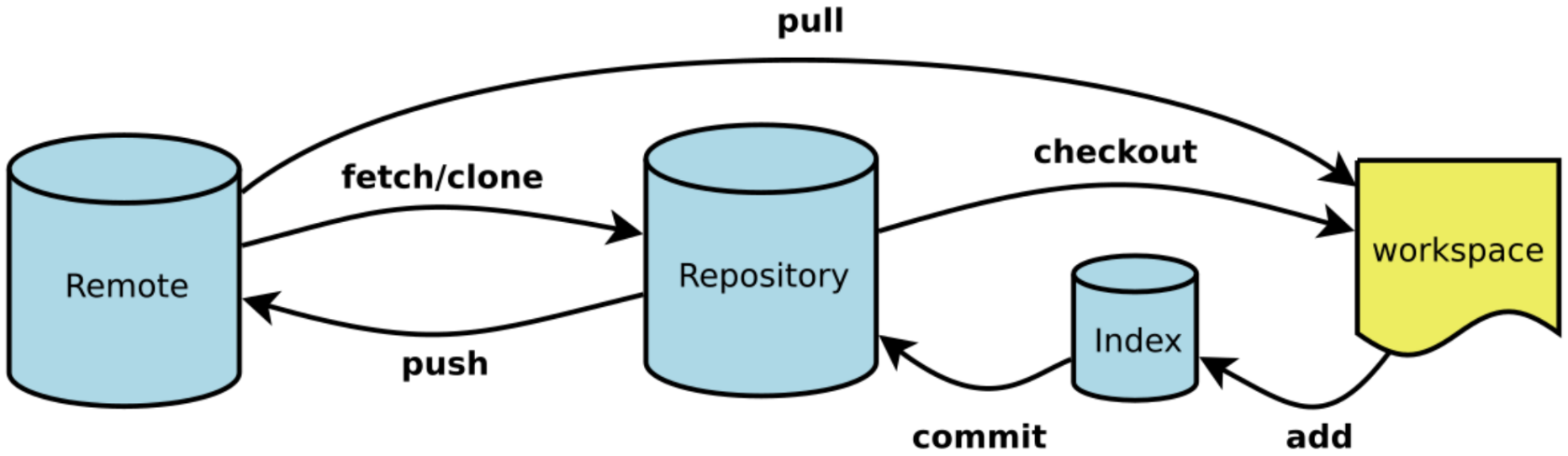
https://git-scm.com/book/zh/v2/%E8%B5%B7%E6%AD%A5-%E5%85%B3%E4%BA%8E%E7%89%88%E6%9C%AC%E6%8E%A7%E5%88%B6

示例：Git入门

- 安装git：https://git-scm.com/
- 声明自己的姓名和email
  - `git config --global user.email ddadaal@outlook.com`
  - `git config --global user.name “Chen Junda”`
- 创建一个目录：`mkdir gittest && cd gittest`
- 初始化git仓库（并同时创建一个master分支）：`git init`
- 新建一个文件：`echo test > test.txt`
- 将文件加入**暂存区**：`git add .`
- 新建一个commit：`git commit -m “first commit”`
- 查看历史：`git log`

Git基本概念

- **Commit**：提交，一系列文件修改，git树基本单位，每个commit有个唯一的ID
- **Index/Stage**：暂存区，放进暂存区的文件称为**staged**
- **Repository**：仓库，保存有所有的commit的内容和之间的关系
- **Workspace**：磁盘上的当前工作目录
- **Remote**：同样是一个仓库，和本地仓库具有相同的内容，只是保存在其他地方



（续）示例：远程仓库

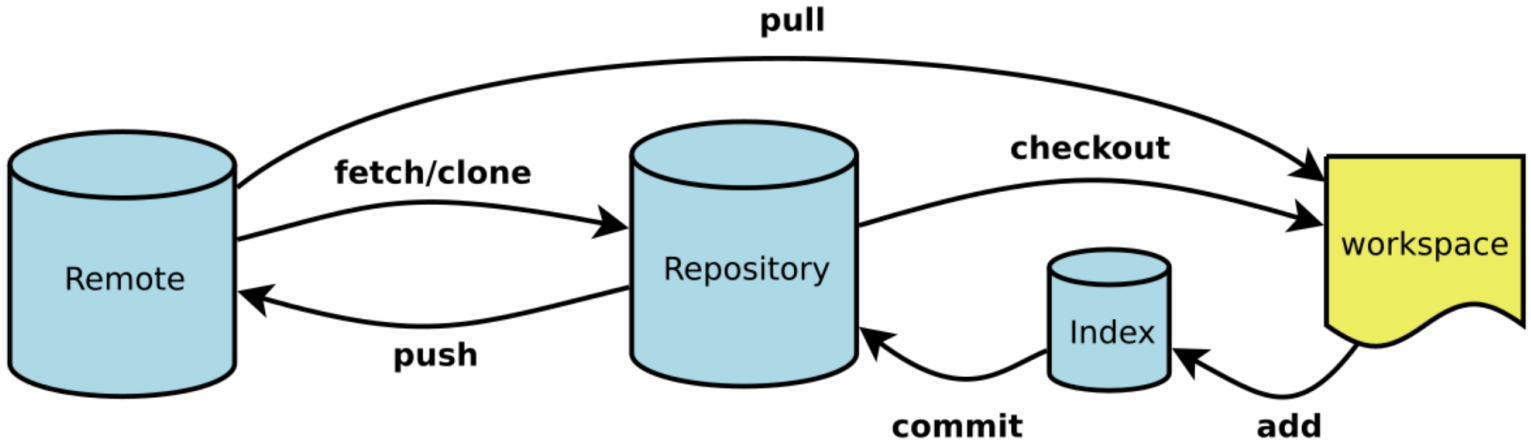
- 去github上创建一个repo，取名为gittest
- 将这个repo设定为一个remote仓库，名字为origin
  - `git remote add origin https://github.com/ddadaal/gittest`
- 将当前分支push到remote仓库origin的master分支
  - `git push origin master`
- 打开github.com/ddadaal/gittest，查看文件
- 在github上修改文件：创建了一个新的commit
- 把远程仓库中的commit pull到本地：`git pull origin master`
- 查看本地文件和提交历史：`cat test && git log`
- Git的commit记录父commit，形成链状结构

（续）示例：远程仓库

- 把仓库clone到本地gittest2目录下
  - `git clone https://github.com/ddadaal/gittest gittest2`
- 和远程和本地仓库对比：`git log`
- gittest、gittest2和远程仓库**具有相同的内容**!

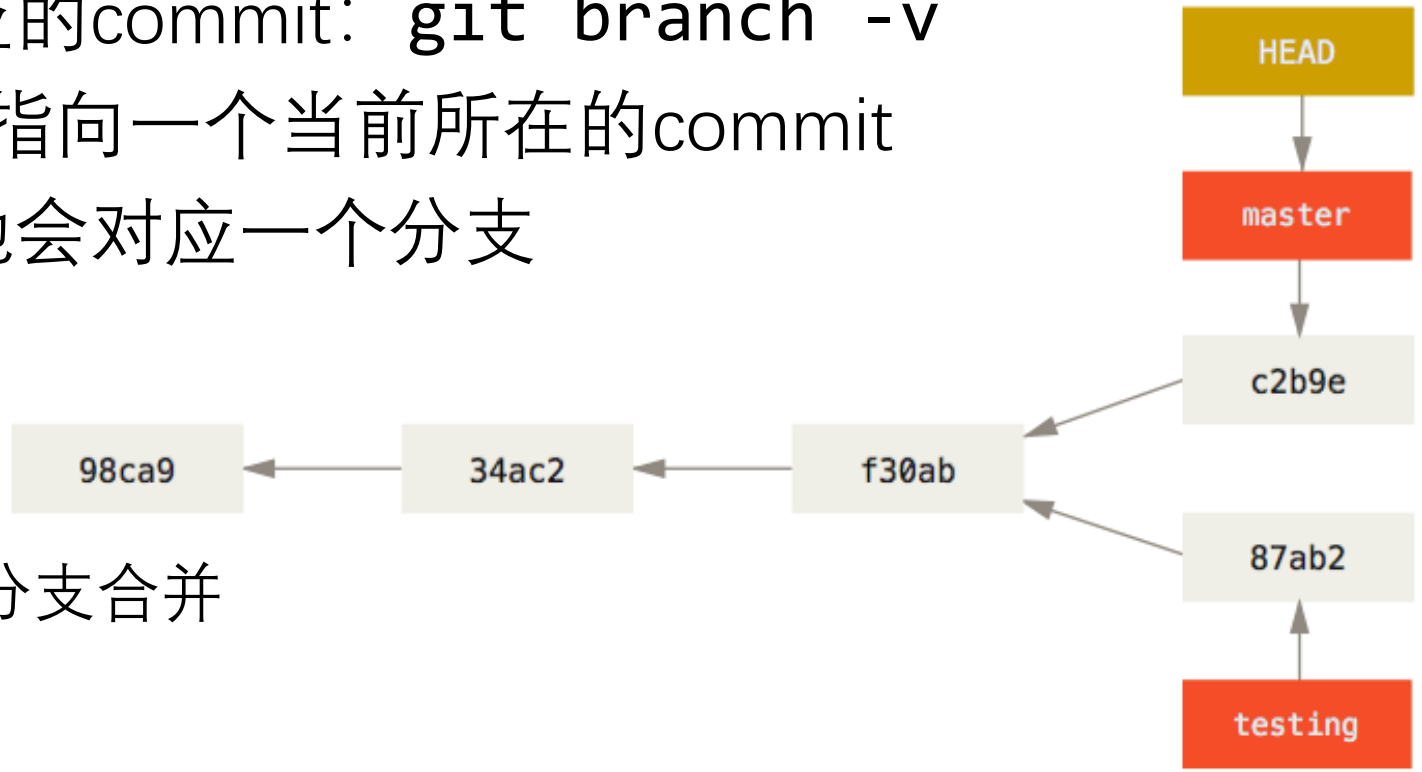
复习：基础命令

- `git add`：把文件加入暂存区
- `git commit`：把当前暂存区文件作为一个commit提交
- `git clone`：从远程克隆一个仓库到本地
- `git pull`：拉取远程仓库的commit到本地，并应用到本地分支
- `git push`：把本地分支更新的commit上传到远程仓库



Git分支

- Git分支：指向某个commit的指针
- 查看所有本地分支以及对应的commit：`git branch -v`
- HEAD：一个特殊的指针，指向一个当前所在的commit
- 远程仓库的每个分支在本地会对应一个分支
  - `git branch -v -a`
- Pull的工作：
  - Fetch：拉取远程分支
  - Merge：把远程分支和本地分支合并



（续）示例：分支和合并

- 创建一个指向当前commit的新分支another
  - `git checkout -b another`
- 创建一个新文件，并在another分支创建一个新的commit
  - `echo another > another && git add . && git commit -m “add”`
- 回到原来的master分支：`git checkout master`
- 查看内容，不存在新文件another：`ls`
- 进行修改，并在master创建一个新的commit
  - `echo haha >> test.txt && git add . && git commit –m “update”`
- 把another分支合并过来：`git merge another`
- 查看提交历史：`git log`

（续）示例：合并冲突

- 切换到another分支：git checkout another
- 在another分支中修改test.txt文件并commit
  - echo haha123 >> test.txt
  - git add . && git commit –m “update”
  - 在master分支中，我们给文件写入的是haha
- 回到master分支：git checkout master
- 尝试merge master分支：git merge another
- 打开文件解决冲突，冲突解决完成后提交文件
  - git add test.txt && git commit –m “resolve conflict”
- 查看刚才的修改：git show HEAD
- 取消merge：git merge --abort

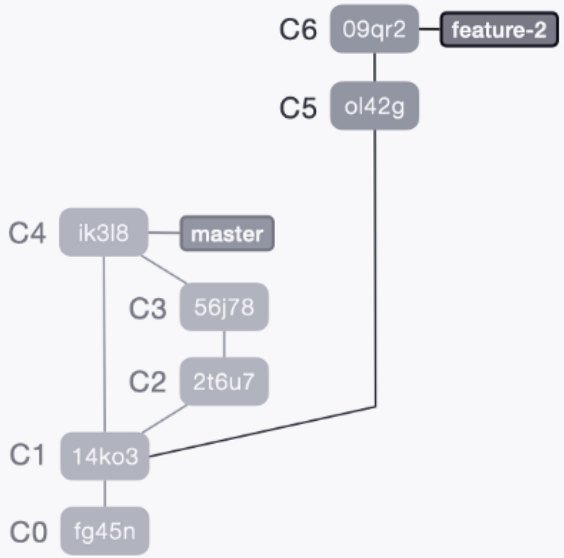
rebase

https://betterprogramming.pub/differences-between-git-merge-and-rebase-and-why-you-should-care-ae41d96237b6

Start case

There are two main ways to integrate changes between branches, **merge** or **rebase**.

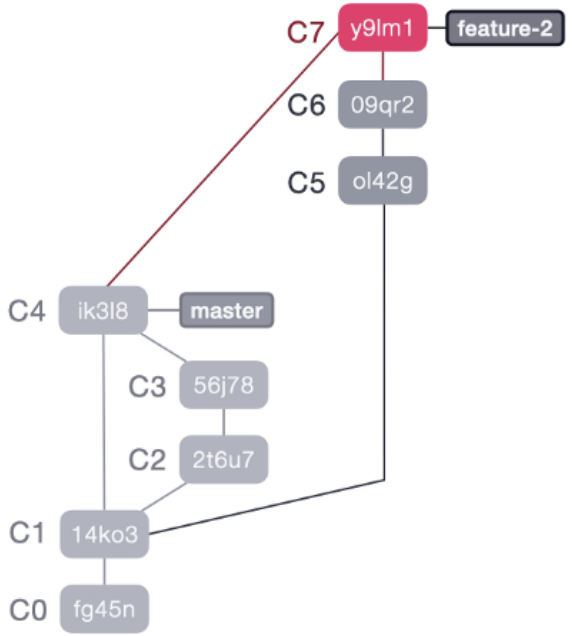
Here, *feature-2* is to be updated with changes from *master*.



Post merge

Merge preserves history as it happened, creating only one new merge commit.

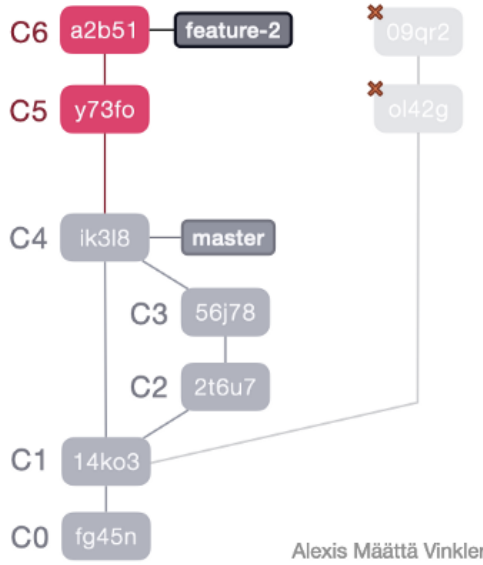
Here, commit **C7** intertwines the two branches – creating a non-linear diamond shaped history.



Post rebase

Rebase rewrites history, reapplying commits on top of another base branch.

Here, commits **C5** and **C6** have been reapplied on top of **C4** – creating a linear history.



Alexis Määttä Vinkler

Git分支组织

- Git的分支速度快，开销小，鼓励多创建
- Master：发布版本，使用tag标记版本；Develop：进行开发
- 每个功能/bugfix一个分支

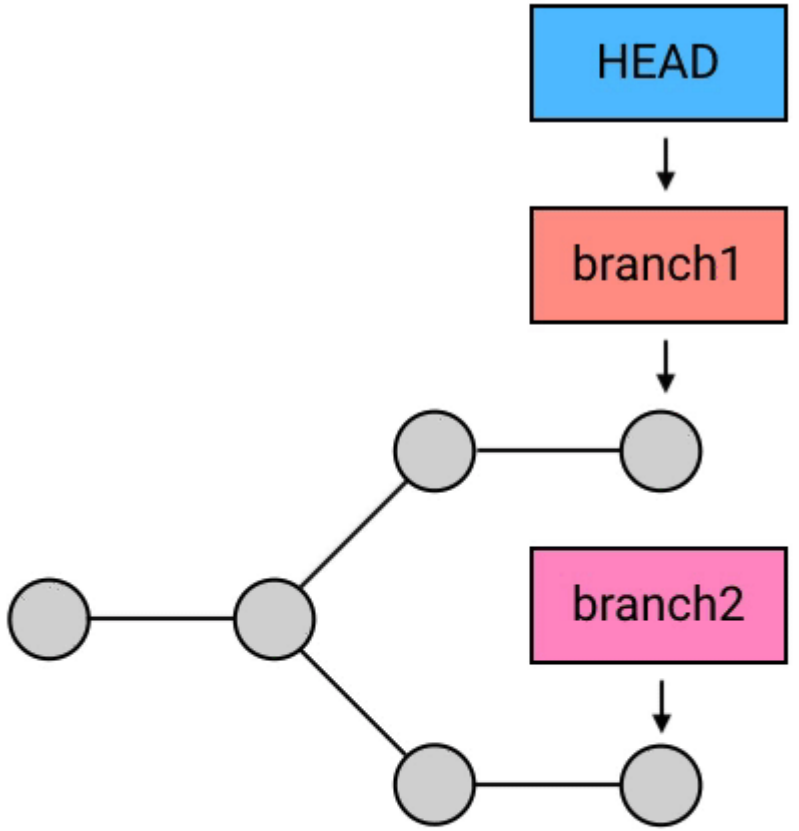
https://blog.51cto.com/u_9291927/2173509

Git在历史中穿梭

- 刚才我们创建了新的commit，如果我想回退到之前的commit?
- 查看历史，记住对应commit的前几个字符：`git log`
- 回到对应的commit的状态
  - `git reset --hard <commit的前几个字符>`
- `git log`:
  - HEAD和master分支指向此commit，没有最新的commit了!
- 想回去？`git reset --hard <之前的commit的前几个字符>`

Git reset

- `git reset <commit或者指针>`:
  - 将HEAD移动到某个commit或者指针对应的commit上
  - 如果HEAD在一个分支上，把分支指针也指向对应commit
  - HEAD^：HEAD的上一个commit
- **--soft**：工作目录不修改，把重置后的diff放进暂存区
- **--mixed**（默认）：工作目录不修改，重置后的diff放在工作目录
- **--hard**：重置暂存区和工作目录，使工作目录中的状态和新的当前状态一致



https://www.jianshu.com/p/c2ec5f06cf1a

部分Git高级命令

- **checkout**
  - <路径>：取得HEAD指针指向的版本中的对应文件，覆盖当前文件
  - <分支名>：把HEAD指向对应分支
  - <commit>：把HEAD指针指向某个commit，进入detached HEAD状态
- **stash**
  - 保存工作区和暂存区的修改并重置工作区
  - 突然来了一个任务需要完成，把当前文件先放放
- **revert**
  - 创建一个**相反**的commit，抵消上次操作
- **cherry-pick**
  - 把一个commit的修改应用到当前HEAD

Git高级用法

- **Git hook：**在执行一个操作之前/之后，运行一个命令
  - 如Lint：检查代码风格，格式不正确的代码**不允许提交**
  - .git/hooks下写对应hook名的shell脚本
- 和远程仓库平台配合，实现**持续集成、部署**
  - 每进行一次commit，自动测试、构建和发布代码

GitHub/GitLab等协作平台的概念

- **Fork**：Fork仓库到自己的账号中
- **Issue**：提出一个问题
  - https://github.com/fastify/fastify-multipart/issues/150
- **Pull Request** (GitHub)/**Merge Request** (GitLab)
  - 请求把自己的账号的某个分支中的commit merge回原仓库
  - https://github.com/fastify/fastify-multipart/pull/151

参与开源项目

- 发现问题，提交Issue
- 合并自己的代码：
  - **Fork**仓库到自己的账号中
  - 在**自己的仓库**中完成修改
  - 对原仓库发起**Pull Request**
  - 沟通，通过往自己的仓库push来更新pull request
  - 原仓库管理员通过PR，代码进入原仓库

tips

- 远程仓库太大，只想要当前最新的commit？
  - git clone <远程仓库> --depth=1
- 远程仓库在clone、pull、push等太慢，自己有HTTP代理？
  - git config --global http.proxy <代理地址>
  - git config --global https.proxy <代理地址>
  - 只适用于HTTP远程仓库
  - SSH：https://segmentfault.com/a/1190000021998129

tips

- 不使用GitHub等仓库平台？
- 通过SSH，把远程机器的普通目录当作远程仓库
  - 在SSH机器上创建一个新的bare repository：git init –bare
  - Bare repository：没有工作目录，远程仓库必须是bare repository
  - 本地机器上：
    - git remote add origin ssh://<SSH用户名>@<地址>:<端口>/<repo路径>
    - git push –u origin master

Git学习资料和相关工具

- https://learngitbranching.js.org/
  - 强烈推荐！交互式的git教程，直观地看到每条git命令对提交树的修改
- VSCode、IDE等工具的git功能
  - 懒得输入命令，用鼠标点点就可以了

总结

- Git是一个分布式版本控制系统，帮助组织修改、多人协作
- Git的基本概念：commit，工作区，暂存区，仓库，远程仓库
- 各个基本命令clone, add, commit, push，pull
- Git分支，合并，解决合并冲突，分支组织
- git的一些高级命令和用法，开源平台概念和流程
- Git是一个**工具**，不要死记硬背，从**实践**中学习