git介绍

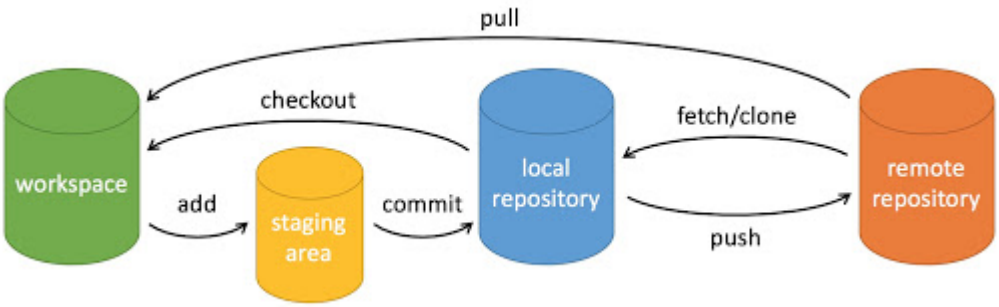
CVS、SVN，集中式的版本控制系统不但速度慢，而且必须联网才能使用，而且有些要付费。

```
sudo apt-get install git-all
```

windows就直接下载安装包安装，电脑中找到“Git”->“Git Bash”运行即可，这是一个mingw64的环境，一个windows模拟出的linux环境，可以使用linux下基本常用的命令。
配置

```
$ git config --global user.name "Your Name"
$ git config --global user.email "email@example.com"
```

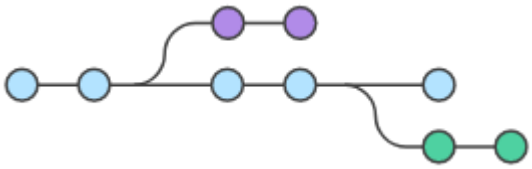
--global是配置全局，对某个仓库使用特定就另外配置。
ssh-keygen -t rsa -C “youremail”
按回车即表示默认这个文件路径名，接着又会提示你输入两次密码（该密码是你push文件的时候要输入的密码，不是github管理者的密码），一般情况都回车，主要是推送代码时不需要输入密码
找到id_rsa.pub文件，复制内容到github中，路径：Settings - SSH and GPG keys - New SSH Key（gitlab类似）。
测试：ssh -T git@github.com （如果其他gitlab就修改对应服务器地址）



git常用命令

git clone、git push、git add 、git commit、git checkout、git pull

- git add .
- 添加文件到暂存区，这个是添加所有文件，添加单个文件需要git add filename
- git commit

    - 将暂存区内容添加到仓库中，后面可以加-m，也可不加-m，不加就会进入vi编辑器添加提交记录
    - git commit --amend 不产生新的commitId
- 恢复
    - git reset --hard commit_id 回退版本，意思是自己的修改都会被删除并恢复到commit_id的版本
    - git reset --hard HEAD^ 这里的HEAD意思就是git指针指向的位置，gitlog查看到的head
    - git checkout file.txt 意思放弃file.txt的修改
- 分支
    - 
    - git branch -a 查看所有分支
    - 查看分支：git branch 查看当前分支
    - 创建分支：`git branch <name>`
    - 切换分支：`git checkout <name>`或者`git switch <name>`
    - 创建+切换分支：`git checkout -b <name>`或者`git switch -c <name>`
    - 合并某分支到当前分支：`git merge <name>`
    - 删除分支：`git branch -d <name>` -D一般是强行删除
- 标签
    - git tag v1.0 这个是标记
    - git tag v0.9 f52c633
    - git tag -d v0.1 删除一个本地标签
    - git push origin v1.0
    - git push origin --tags 可以推送全部未推送过的本地标签
    - git push origin :refs/tags/tagname 可以删除一个远程标签。
- git merge
    - git merge dev 把dev分支合并到默认HEAD（通常是master或者main分支）
- git push
    - git push -u origin master 第一次推送master分支的所有内容，后续推送不需要带-u，这里面还有些其他的用法，有遇到再学3
- git remote
    - git remote -v 远端仓库信息
    - git remote rm origin 删除

git高级命令

- git stash
  git stash save 'message' 作用同上，只是标记信息
  git stash list
  git stash apply <可以跟list对应的id>恢复后，stash内容并不删除，你需要用`git stash drop`来删除
  git stash pop <可以跟list对应的id> 恢复的同时把stash内容也删了
- git cherry-pick
  git cherry-pick commitId 复制一个特定的提交到当前分支，这个commitId一般是其他分支的
- git rebase
  git rebase 一般是回滚基线，意思就是在git log查看的时候时间线上会很清晰
  一般git push之前一般都会git pull，但是这样操作之后会产生一些额外的git log（类似git merge），所以通常会使用 git pull --rebase

github/gitlab的开始使用

- 创建新仓库

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

- 推送一个存在的仓库

```
git remote add origin git@github.com:gavinlook/github-upload.git
git branch -M master
git push -u origin master
```

- 使用已经存在的仓库
  克隆代码到本地——这个是通用的代码操作方式

  git status 修改文件后查看修改
  默认查看全部，查看某文件后面对应文件，可以查看帮助文档 git status --help

```
git clone git@github.com:torvalds/linux.git
touch readme.txt
```

```
git add readme.txt
git commit -m "wrote a readme file"
git push
```

-m后面跟随提交说明，也就是常用git log查看到的提交记录，所以最好简洁明了

git常用问题

- git status提示old mode改动问题
  - 单个项目配置
    git config --add core.filemode false
    忽略掉chmod改动
  - git全局配置：
    git config --global core.filemode false
- git出现换行符^m问题
  - git config --global core.autocrlf true
  - 具体是true还是false看情况，也可配全局的，有问题时候再解决
- git config -list查看配置
- clone代码方式
  - `https`除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放http端口的公司内部就无法使用`ssh`协议而只能用`https`
- git log
  - git log --stat 简略统计信息
  - git log --pretty=oneline 简洁查看
  - git log --pretty=format:"%h - %an, %ar : %s"
  - git log --pretty=format:"%h %s" --graph
  - git log --author=Linus --oneline -5
  - git log --reverse --oneline
  - git log --oneline --before={3.weeks.ago} --after={2010-04-18} --no-merges
  - git log --since=2.weeks
  - git log --oneline --decorate --graph --all你自己提交历史、各个分支的指向以及项目的分支分叉情况
  - http://git-scm.com/docs/git-log

其他软件

- vscode
  - 插件：Git Graph，GitLens。
- tortoise git
  - 这个右键选择git log可以查看所有的记录。

- 目前我的选择
    - 目前代码服务器有些问题，使用tortoise git响应很慢，所以一般使用vscode的Git Graph查看提交记录。另外也可以用SourceTree，这个是

参考

使用教程参考
Pro git2