# 一、Git概述

分布式版本控制系统（区别于集中式版本控制系统SVN）

无需联网

内容存储使用的是 SHA-1 哈希算法

# 二、Git安装及配置

1.Ubuntu

#执行安装

$apt-get install git

#查看版本

$git --version

2.windows

直接官网下载，安装

https://git-scm.com/download

3.配置

#查询配置信息

$git config --list

#配置个人信息(--global仅针对当前登录用户)

$git config --global user.name “jiguotong”

$git config --email user.email [1776220977@qq.com](mailto:1776220977@qq.com)

# 三、Git理论原理



workspace：工作区

staging area：暂存区/缓存区

local repository：版本库或本地仓库

remote repository：远程仓库

当对工作区修改（或新增）的文件执行 **git add** 命令时，暂存区的目录树被更新，同时工作区修改（或新增）的文件内容被写入到对象库中的一个新的对象中，而该对象的ID被记录在暂存区的文件索引中。

当执行提交操作（git commit）时，暂存区的目录树写到版本库（对象库）中，master 分支会做相应的更新。即 master 指向的目录树就是提交时暂存区的目录树。

# 四、Git基本使用

1.创建版本库

$mkdir git\_learning

$git init

2.新建文件并提交至版本库

$touch readme.txt

$git add readme.txt

$git commit -m “提交readme文件”

$git add \*.txt #批量添加所有.txt作文后缀的文件

$git add . #批量添加当前目录下所有未被添加的文件

3.查看版本库状态

$git status #可以查看仓库当前的状态

4.查看某个文件的改动部分

$git diff readme.txt

5.查看本仓库的提交日志

$git log

$git log --pretty=oneline

6.回退版本

#head^代表上一个版本，head^^代表上上个版本，head^100代表上一百个版本。

$git reset --hard head^

或者

$git reset --hard [commit id]

如果是不记得commit id，可以用命令git reflog查看，其记录了你的每一个命令。

7.撤销修改

$git checkout -- readme.txt

#将工作区的文件内容撤销到上一次add或者commit的状态，即与暂存区或者版本库同步

8.文件删除与恢复

#删除一个文件

$git rm test.txt

#(只要版本库中有该文件就可恢复)

$git checkout – test.txt

# 五、Git远程仓库

1.Github概念及基础设置

Git是分布式版本控制系统，同一个Git仓库，可以分布到不同的机器上。

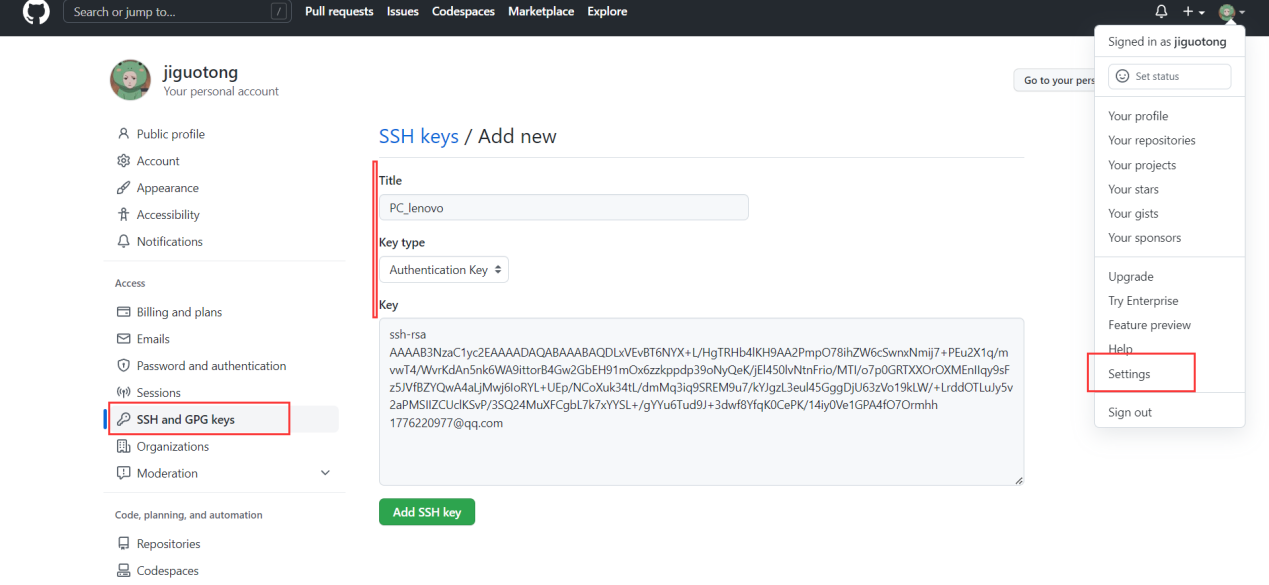
本地Git仓库和GitHub仓库之间的传输是通过SSH加密的，所以需要设置密钥/公钥。

第1步：创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

$ ssh-keygen -t rsa -C [1776220977@qq.com](mailto:1776220977@qq.com)

一路回车，完成后产生id\_ras和id\_rsa.pub两个文件，id\_ras是私钥， id\_rsa.pub是公钥。

第2步：打开Github，进行公钥添加



2.Github基本操作

（1）新建库

（2）本地同步到远程仓库

#建立关联

$ git remote add origin git@github.com:jiguotong/gitlearning.git

#本地仓库push到远程仓库

$ git push <远程主机名> <本地分支名>:<远程分支名>

$ git push -u oringin master

#查看关联

$ git remote -v

#断开关联

$ git remote rm origin

（3）远程仓库克隆到本地仓库

$ git clone [url]

（4）远程仓库更新到本地

①直接pull，git pull <远程主机名> <远程分支名>:<本地分支名>

$ git pull origin master

②先fetch，后merge

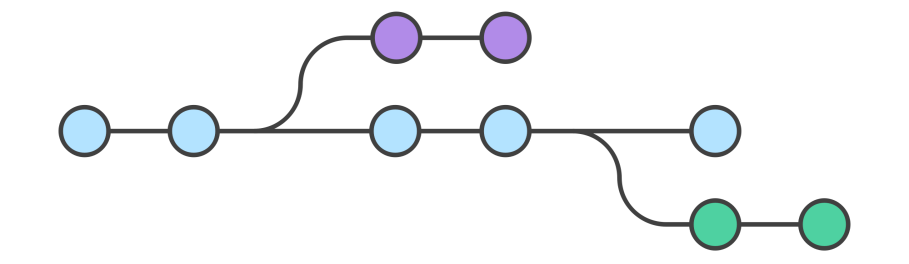
$ git fetch origin

$ git merge origin/master

# 六、Git分支管理

1.分支基本概念

一个分支代表一条独立的开发线，使用分支意味着从开发主线上分离开来。





2.分支相关命令

#列出分支（当前分支会显示\*号）

$ git branch

# 创建分支

$ git branch ji

#切换分支

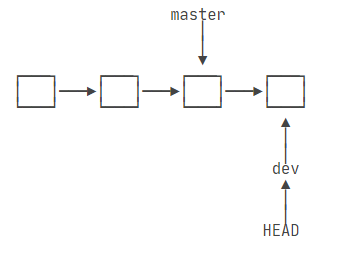
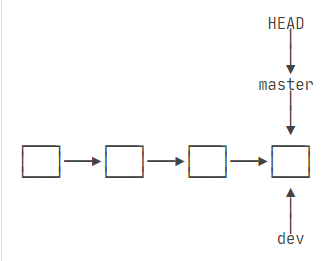
$ git checkout ji

#创建并切换到该分支

$ git checkout -b ji

#合并分支（合并指定分支到当前分支）

$ git merge ji #合并ji到master，将master指向ji的当前提交

→

#删除分支（分支已被合并）

$ git branch -d ji

#删除分支（分支未被合并，强制删除）

$ git branch -D ji

3.冲突合并

进行分支合并的时候可能会出现冲突，这时候需要进行手动解决冲突。

然后对文件进行一下add

然后再commit

最后删除分支

# 七、Git标签管理

1.发布一个版本时，我们通常先在版本库中打一个标签（tag），这样就唯一确定了打标签时刻的版本.

2.标签相关命令

#查看标签

$ git tag

#创建标签 git tag <name>(默认标签是打在最新提交的commit上)

$ git tag V1.0

#给指定commit添加标签

$ git log --pretty=oneline --abbrev-commit #查看所有提交

$ git tag v0.9 fa9949d

#创建带有说明的标签

$ git tag -a V1.0 -m “This is the first tag!”

#查看标签所指的commit信息

$ git show V1.0

#推送标签到远程 git push <远程主机> <标签名字>

$ git push origin V1.0

#一次性推送全部尚未推送到远程的本地标签

$ git push origin --tags

#删除本地标签

$ git tag -d V1.0

#删除远程标签

$ git push origin :refs/tags/V1.0

# 八、Git其他配置

1、配置别名（开发神器）

git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

2.

# 八、Git问题汇总

1、小乌龟push的时候出现git did not exit cleanly exit code 128

解决方法：“TortoiseGit” --> "Settings" --> "Network"

将C:\Program Files\TortoiseGit\bin\TortoiseGitPlink.exe改为

D:\Git\usr\bin\ssh.exe即可(D:\Git是我的安装目录)

2、

<https://www.liaoxuefeng.com/wiki/896043488029600/899998870925664>

https://www.runoob.com/git/git-server.html