**git教程**

Git是目前世界上最先进的分布式版本控制系统，能记录每次文件的改动。

Linus在1991年创建了开源的Linux，经过不断发展，已经成为最大的服务器系统软件。

2002年以前，世界各地的志愿者把源代码文件通过diff的方式发给Linus，然后由Linus本人通过手工方式合并代码。Linus坚定地反对CVS和SVN，这些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。有一些商用的版本控制系统，虽然比CVS、SVN好用，但那是付费的，和Linux的开源精神不符。

2002年，代码库之大让Linus很难继续通过手工方式管理，于是Linus选择了一个商业的版本控制系统BitKeeper，BitKeeper的东家BitMover公司出于人道主义精神，授权Linux社区免费使用这个版本控制系统。开发Samba的Andrew试图破解BitKeeper的协议，被BitMover公司发现并收回Linux社区的免费使用权。

于是Linus花了两周时间自己用C写了一个分布式版本控制系统，这就是Git！

2008年，GitHub网站上线了，它为开源项目免费提供Git存储，无数开源项目开始迁移至GitHub，包括jQuery，PHP，Ruby等等。

集中式版本控制系统，版本库是集中存放在中央服务器的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始干活，干完活了，再把自己的活推送给中央服务器。集中式版本控制系统最大的毛病就是必须联网才能工作。

分布式版本控制系统根本没有“中央服务器”，每个人的电脑上都是一个完整的版本库，这样，你工作的时候，就不需要联网了，因为版本库就在你自己的电脑上。分布式版本控制系统通常也有一台充当“中央服务器”的电脑，但这个服务器的作用仅仅是用来方便“交换”大家的修改，没有它大家也一样干活，只是交换修改不方便而已。

和集中式版本控制系统相比，分布式版本控制系统的安全性要高很多，因为每个人电脑里都有完整的版本库，某一个人的电脑坏掉了不要紧，随便从其他人那里复制一个就可以了。而集中式版本控制系统的中央服务器要是出了问题，所有人都没法干活了。Git的优势不单是不必联网这么简单，后面我们还会看到Git极其强大的分支管理。

最早Git是在Linux上开发的，很长一段时间内，Git也只能在Linux和Unix系统上跑，经过移植，Git可以在Linux、Unix、Mac和Windows这几大平台上正常运行。

一、安装git

在linux上安装git (如下图所示):

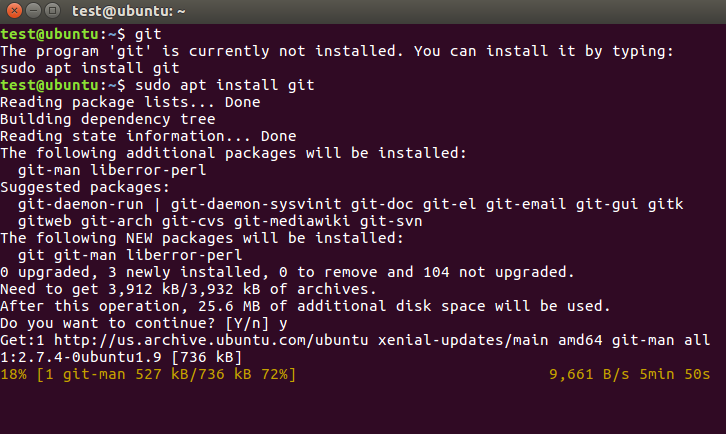
$ sudo apt/apt-get install git

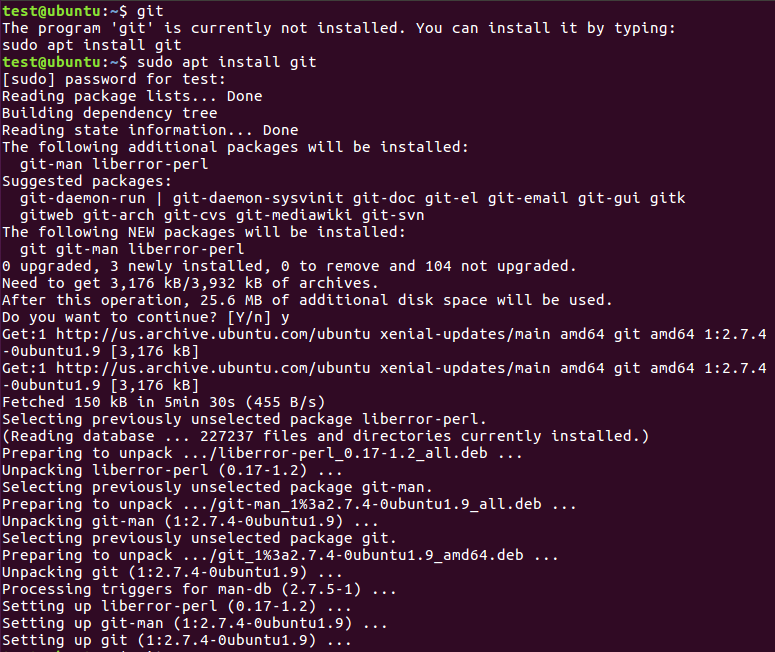
安装完成后在命令行输入：

$ git config --global user.name "yourname"

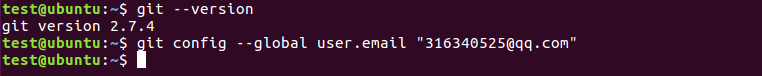
$ git config --global user.email "email@example.com"

--global参数，表示你这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。





安装成功并查看版本如下图所示



1. 创建版本库（本地仓库）以及管理修改

版本库又名仓库，英文名repository，你可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等等，Git也不例外。版本控制系统可以告诉你每次的改动，而图片、视频这些二进制文件，虽然也能由版本控制系统管理，但没法跟踪文件的变化，只能把二进制文件每次改动串起来，也就是只知道图片从100KB改成了120KB，但到底改了啥，版本控制系统不知道，也没法知道。

有关文本的编码强烈建议使用标准的UTF-8编码，所有语言使用同一种编码，既没有冲突，又被所有平台所支持。

1. 创建版本库：

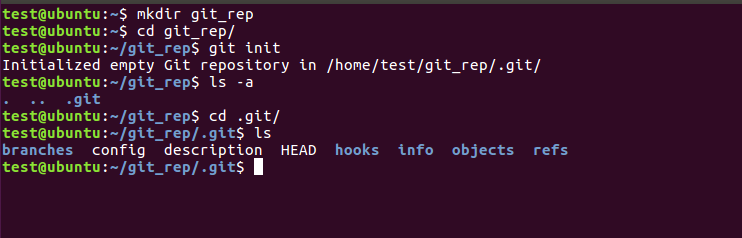
$ git init

可以发现在当前目录下有一个隐藏文件为.git，此文件是用来跟踪管理版本库的。

当前目录被称为工作区

.git被称为版本库，其中有很多内容，最重要的内容是以下三个：

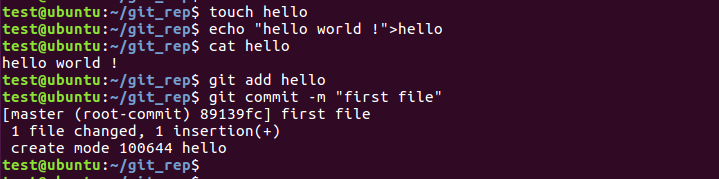
1. 被称为stage/index的暂存区
2. git为我们自动创建的一个主分支master
3. 指向master的一个被称为HEAD的指针



1. 添加文件到版本库：

git add hello 将hello文件修改从工作区添加到暂存区

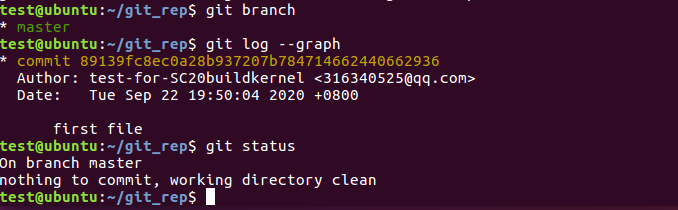
git commit -m "hello" 将hello从暂存区(其实是将暂存区中所有内容)提交到当前分支



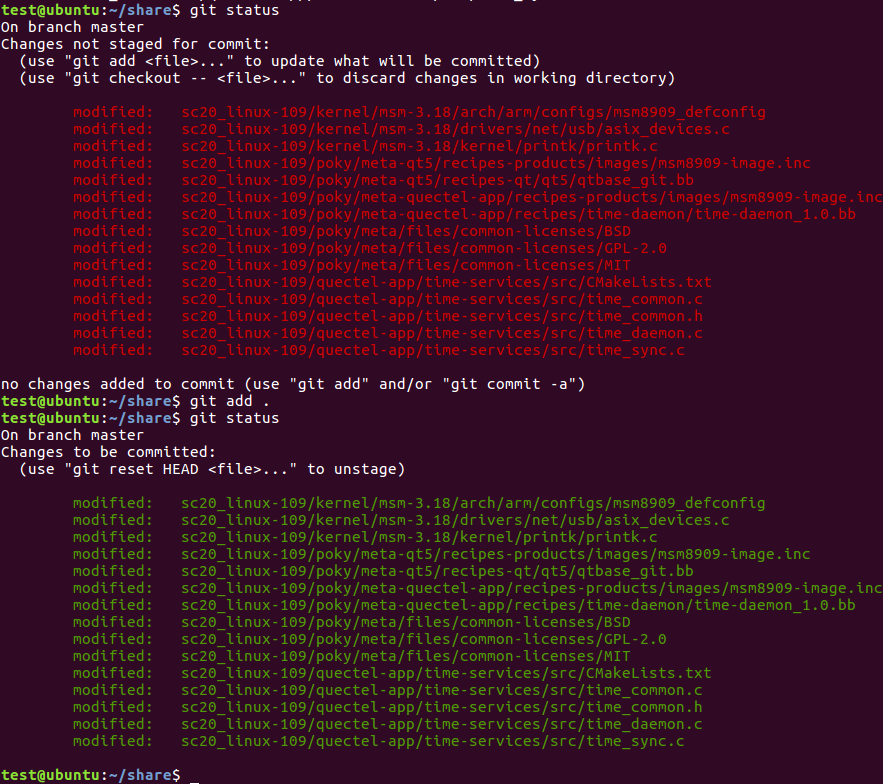
git branch 查看当前所在分支

git log --graph 查看分支

git status 查看当前所在分支状态



git add . 表示添加工作区中的所有修改内容(注意后面的点)



1. 版本回退：

git diff hello 比较文件的修改(比较工作区与暂存区中的文件)

当暂存区没有文件时，工作区与最后一次从暂存区提交到仓库的文件比较

git diff --cached 比较文件的修改(比较暂存区与最后一次提交的文件)

git diff HEAD 比较文件的修改(比较工作区与最后一次提交的文件)

其中

index a95200f . . 20c481d 100644

表示两个文件的hash值(index区域的a95200f对象与工作区的20c481d对象)

100644 表示文件的属性(st\_mode)，权限(权限为644)

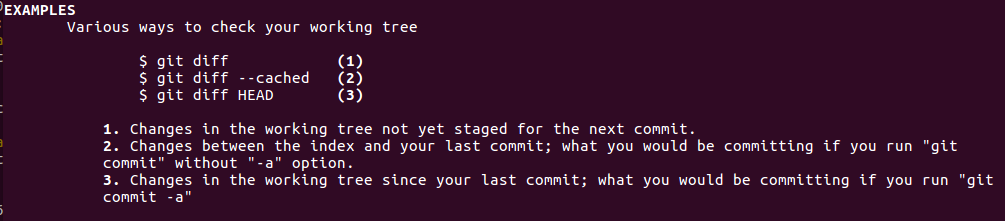
--- 旧文件

+++ 新文件

@@ -1 +1，2 @@

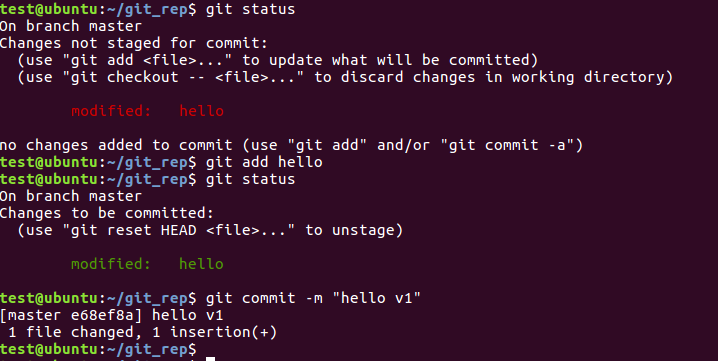
表示旧文件的内容从第一行开始向后偏移0行，

新文件的内容从第一行开始向后偏移2行





查看当前分支状态，将修改过的hello文件添加并提交到本地仓库



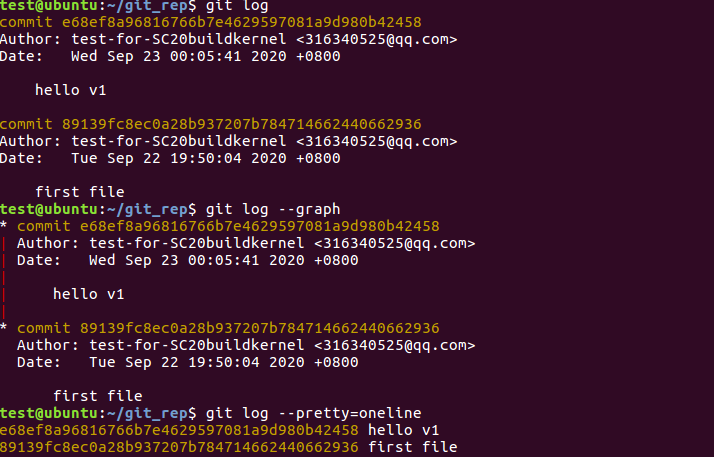
git log 查看历史版本，即从最近到最远的提交日志

git log --graph 查看当前分支

git log --pretty=oneline 查看历史版本的简略信息，第一个为HEAD指向的

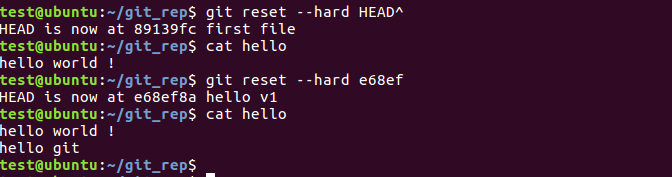
其中commit 后面的是版本号

git中HEAD指向是当前版本，上一个版本是HEAD^，上上一个版本是HEAD^^,往上 n 个版本表示为HEAD~100。

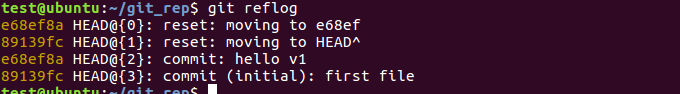


git reset --hard HEAD^ 回到上一个版本

git reset --hard commit\_ID 回到某个版本

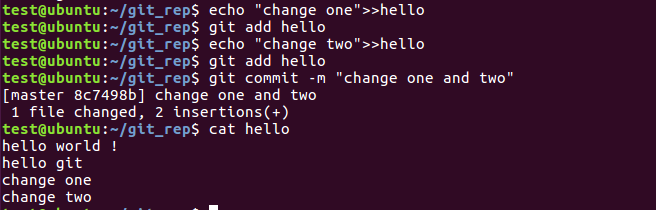


git reflog 记录每一次命令



1. 管理修改

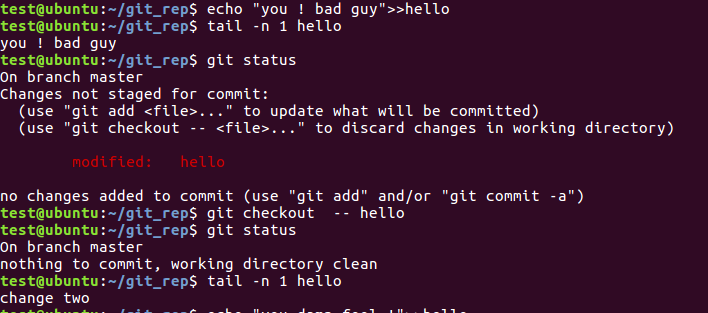
修改hello 文件两次，添加两次，一次提交即可合并前两次的修改



1. 撤销修改

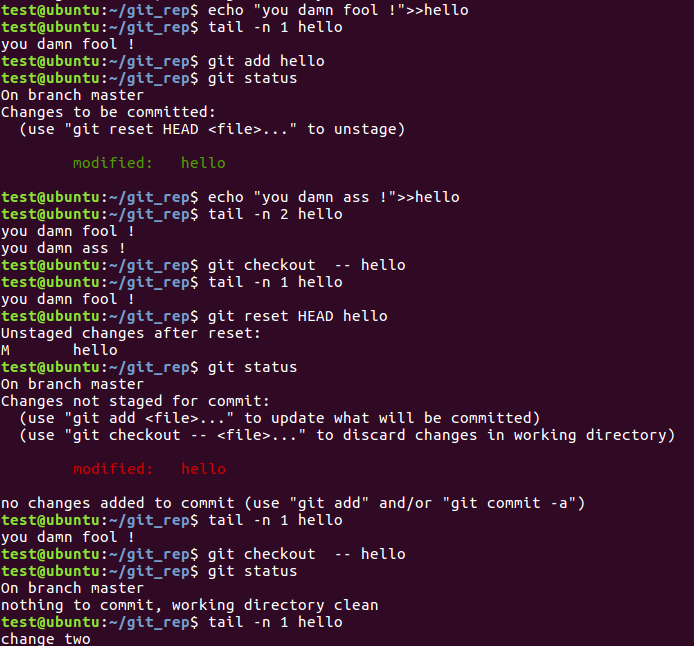
第一种：

工作区中有修改，未添加未提交，使用git checkout -- hello即可撤销修改。



第二种：

已添加，未提交，工作区又有修改，可参考第一种撤销修改，想要撤销已添加的修改，可使用 git reset HEAD hello 撤销暂存区的修改并放回工作区，此时又回到的第一种情况。



第三种：

已提交修改，此时若想要撤销，可是用版本回退，参考3. 版本回退。

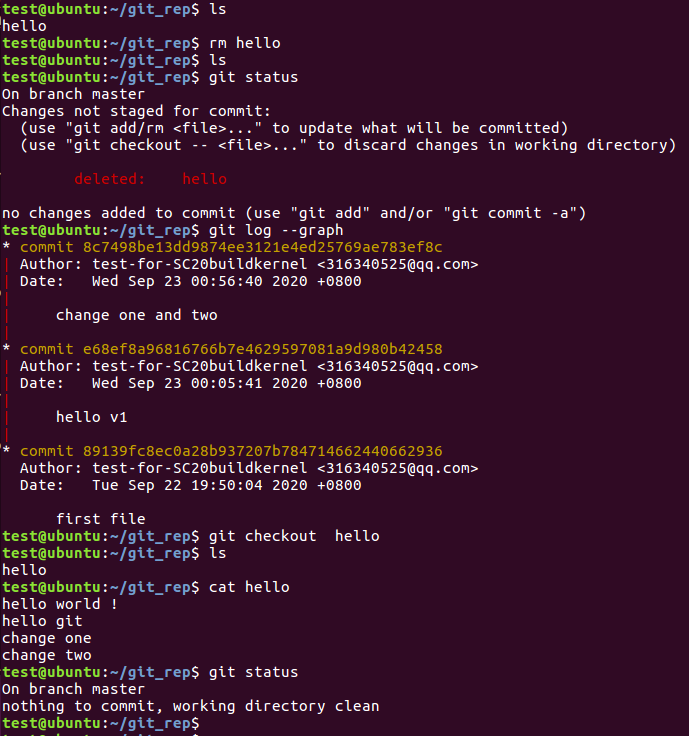
1. 删除修改

rm hello 删除工作区的文件

若版本库中有该文件，则可使用

git checkout hello

来恢复文件

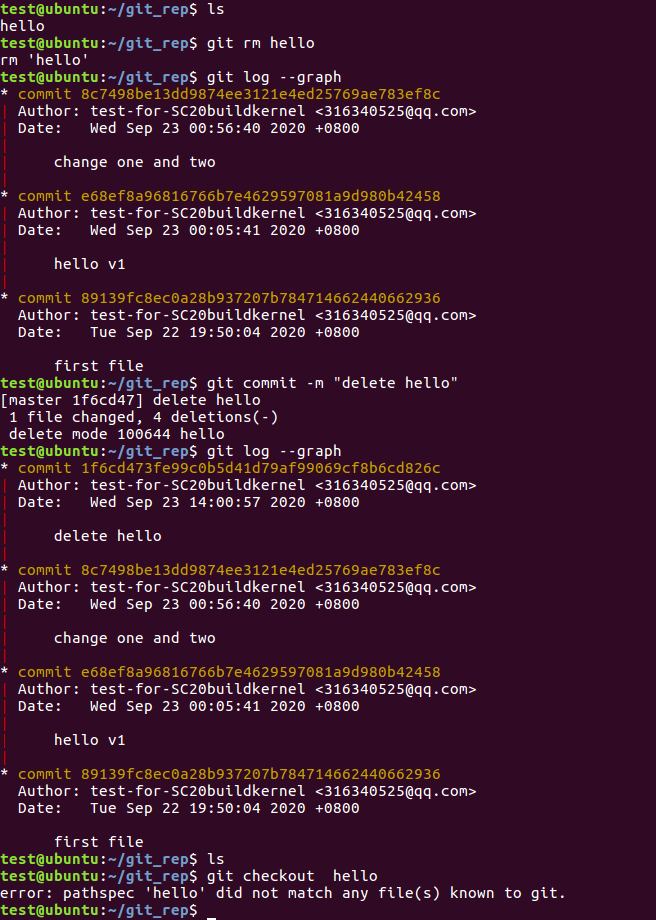


git rm hello 用来删除版本库中的文件，删除后工作区的文件也同时被删除，

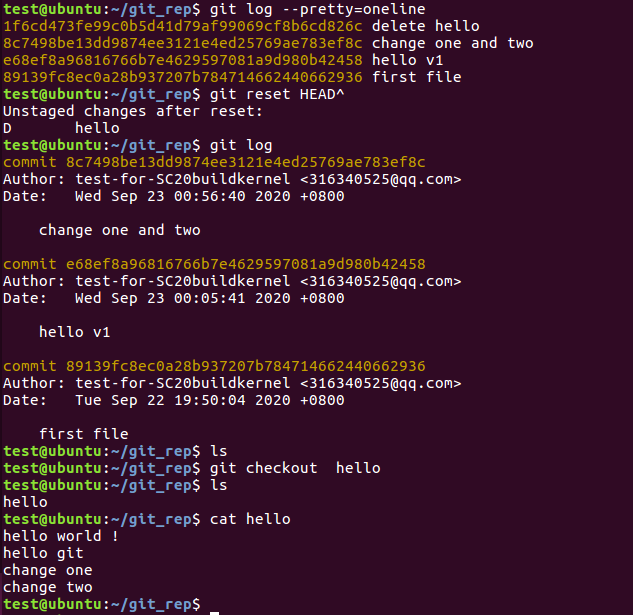
删除后使用git commit -m "delete hello"来提交一下说明已经删除了该文件

无法使用git checkout hello 来从版本库中恢复文件到工作区了。

操作如下图。



若要恢复hello文件，可参考3. 版本回退，操作如下图。



1. 远程仓库

找一台电脑充当服务器的角色，每天24小时开机，其他每个人都从这个“服务器”仓库克隆一份到自己的电脑上，并且各自把各自的提交推送到服务器仓库里，也从服务器仓库中拉取别人的提交。

也可以自己搭建一台运行Git的服务器，目前没必要。

[GitHub](https://github.com/" \t "https://www.liaoxuefeng.com/wiki/896043488029600/_blank)、[Gitlab](https://github.com/" \t "https://www.liaoxuefeng.com/wiki/896043488029600/_blank)等网站，是提供Git仓库托管服务的，所以，只要注册一个账号，就可以免费获得Git远程仓库。

git 支持多种协议，例如https、ssh等等，但是ssh协议是速度最快的。

本地Git仓库和[GitHub](https://github.com/" \t "https://www.liaoxuefeng.com/wiki/896043488029600/_blank)仓库之间的传输是通过SSH加密的，故需要设置SSH key。

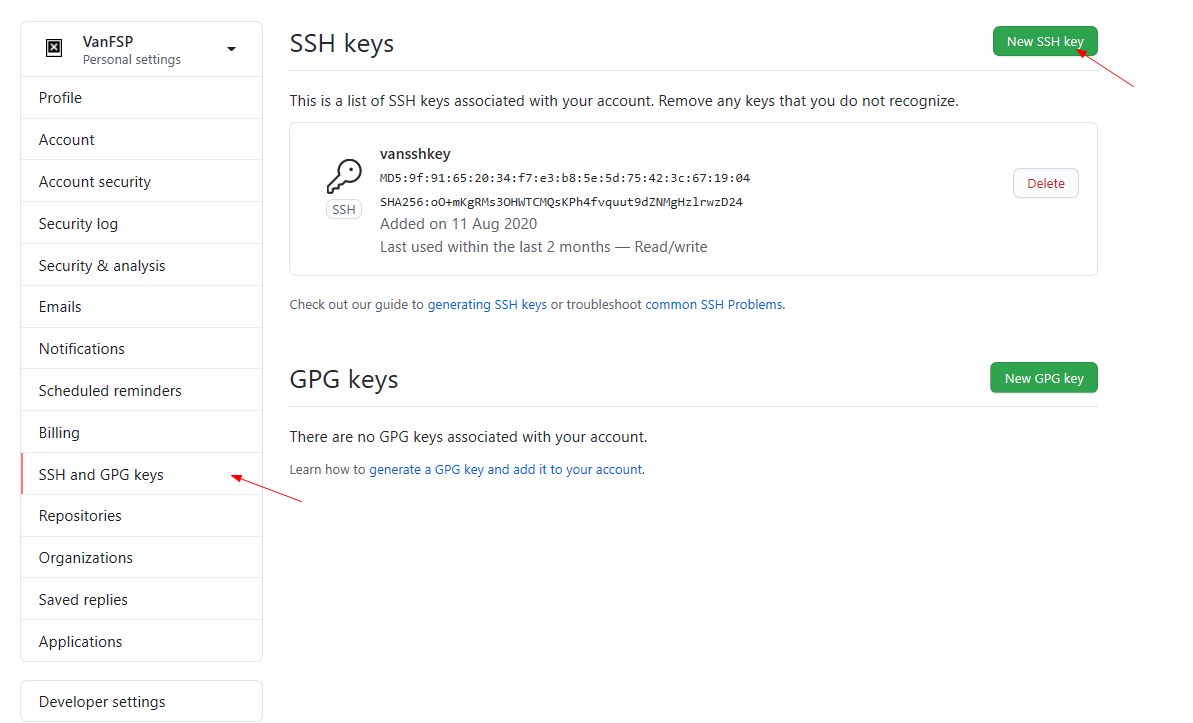
在家目录下执行

ssh-keygen -t rsa -C ["email@example.com"](mailto:\"email@example.com\")

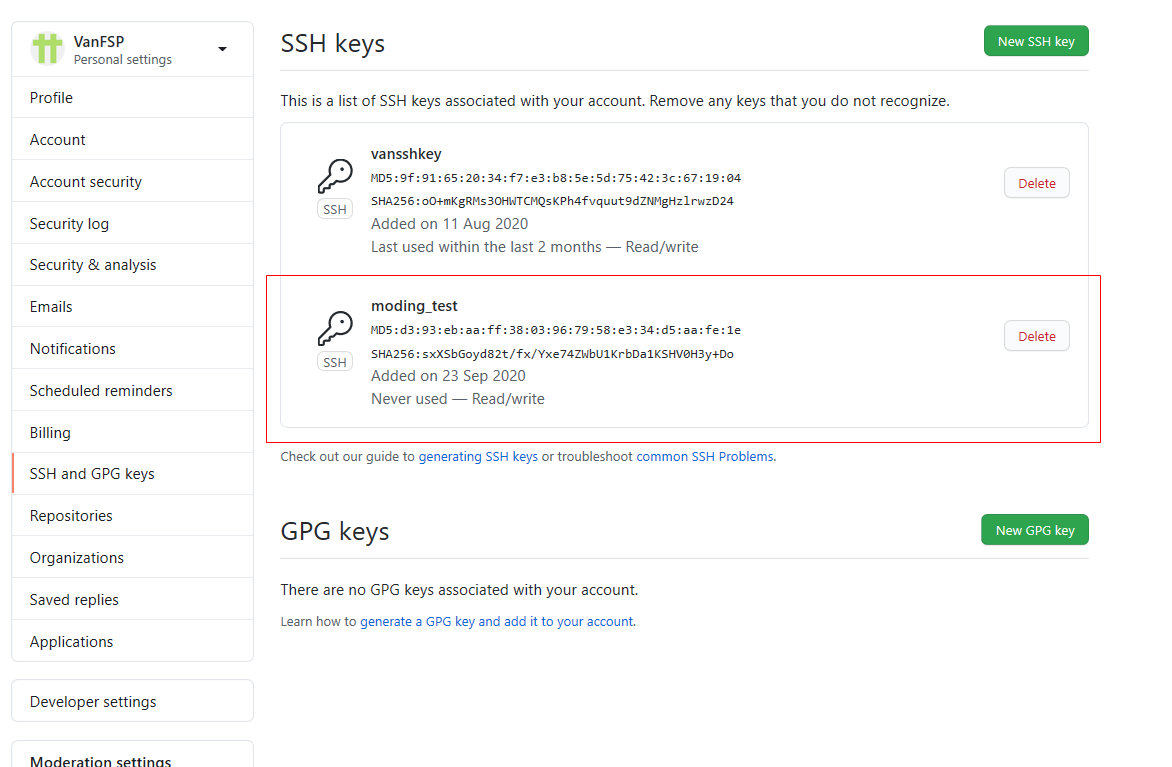
一直回车(即默认选项)即可。

之后会生成一个.ssh目录，进入目录看到两个密匙，将公匙复制即可。





添加完成后如下图所示



在GitHub上免费托管的Git仓库，任何人都可以看到喔（但只有你自己才能改）。

1. 添加远程仓库并推送

通过

git remote add origin [git@github.com:VanFSP/sc20\_linux.git](mailto:git@github.com:VanFSP/sc20_linux.git)

或

git remote add origin <https://github.com/VanFSP/sc20_linux.git>

关联远程仓库，origin 是远程仓库的默认名字

VanFSP 是github账号的用户名

sc20\_linux 是你所建远程仓库

git push -u origin master 将本地仓库master主分支上的所有内容推到远程

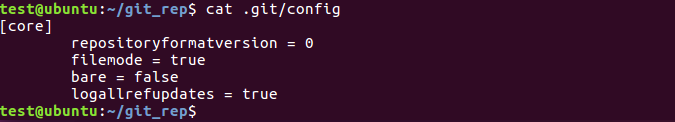
-u 参数表明git不但会把本地的master分支内容推送的远程新的master分支， 还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取 时就可以简化命令，即

git push origin master

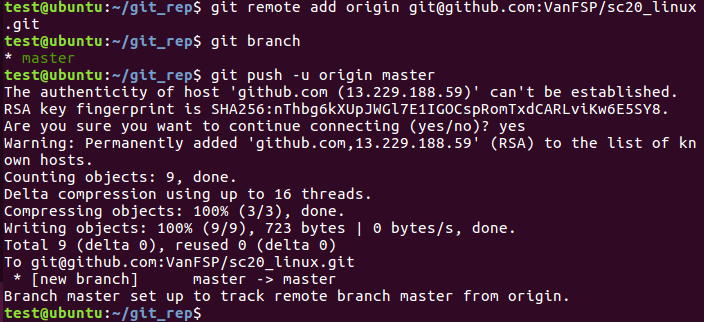
可把本地master分支的最新修改推送到远程仓库。

当你第一次使用git的clone或者push命令连接GitHub时，会得到一个警告，git 使用SSH连接，而SSH连接在第一次验证GitHub服务器的Key时，需要你确认 GitHub的Key的指纹信息是否真的来自GitHub的服务器，输入yes回车即可。

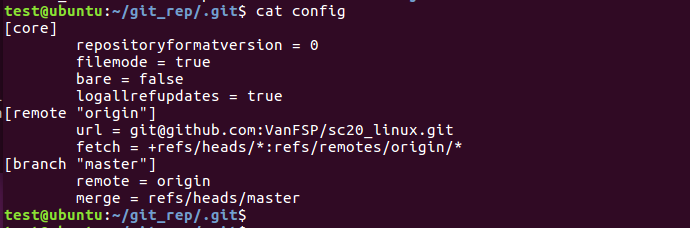
首先查看仓库配置文件如下



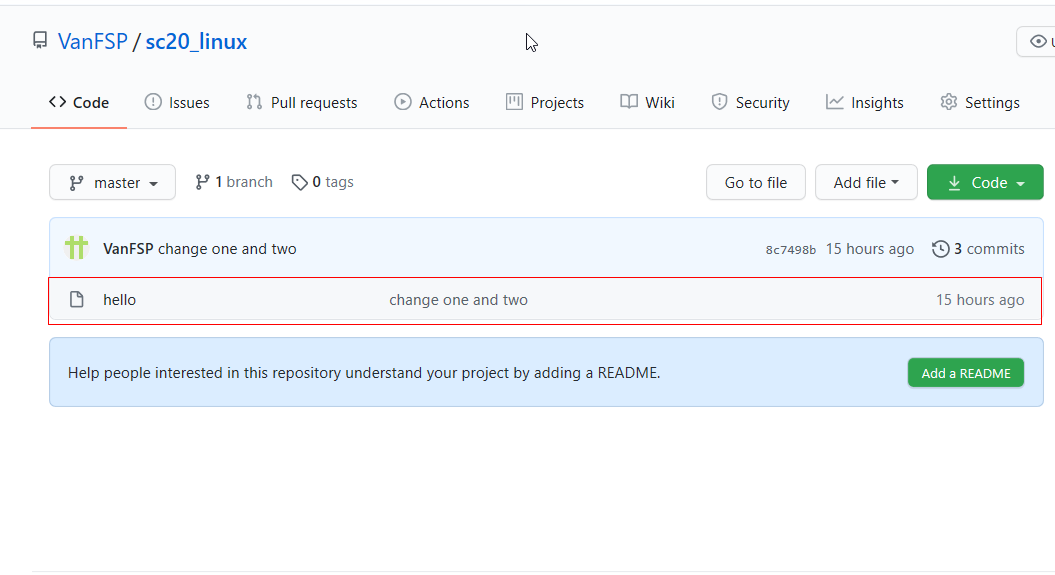
添加远程仓库并推送如下



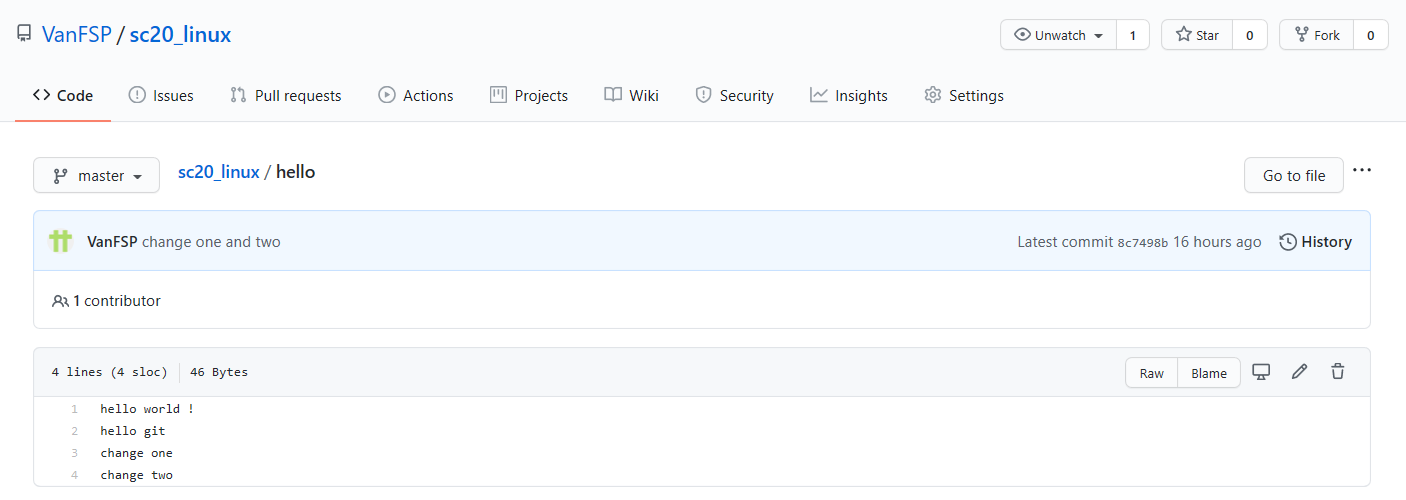
可以通过.git/config 文件看到我们关联的远程仓库以及分支



查看到hello已经推到github的仓库中

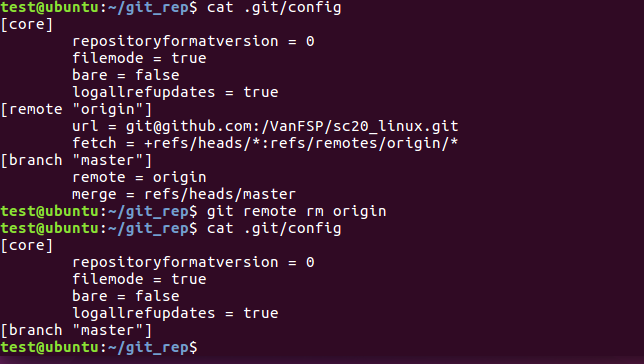


此时查看到hello文件中的内容是与本地仓库一致的



若要删除该配置文件中与远程仓库的关联，通过

git remote rm origin



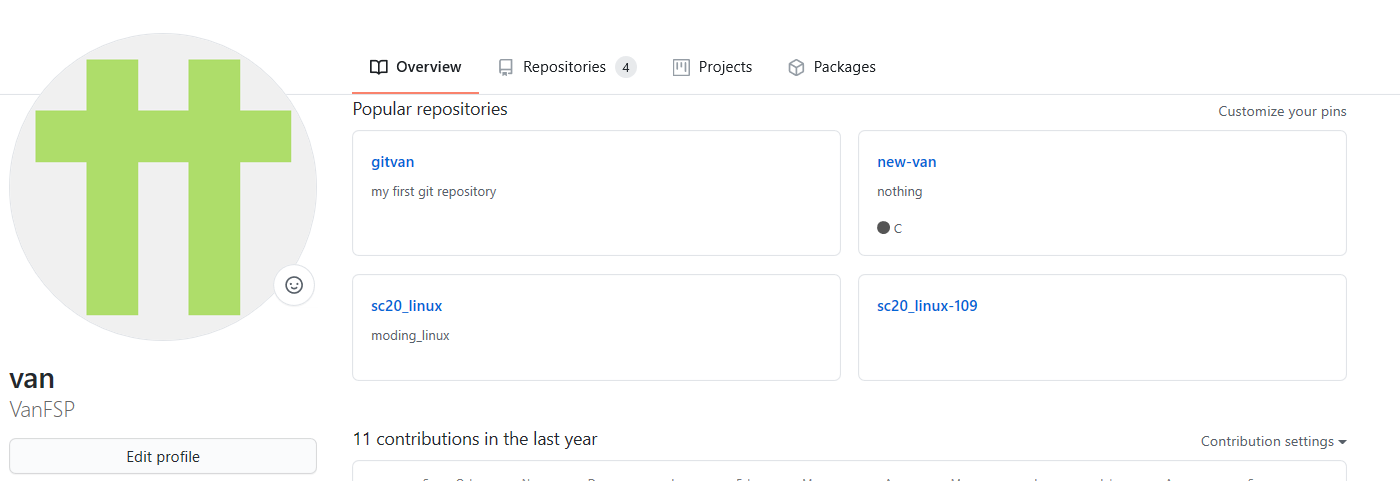
使用

git remote add origin [https://github.com/VanFSP/sc20\_linux-109.git](https://github.com/VanFSP/sc20_linux.git)

关联远程仓库后，在第一次使用git 的push以及clone命令时需要你输入远程仓 库所在用户的用户名以及密码。

1. 从远程仓库克隆到本地

查看到我的远程有四个仓库，但是目前只有其中三个有文件，我们来克隆其中一个。



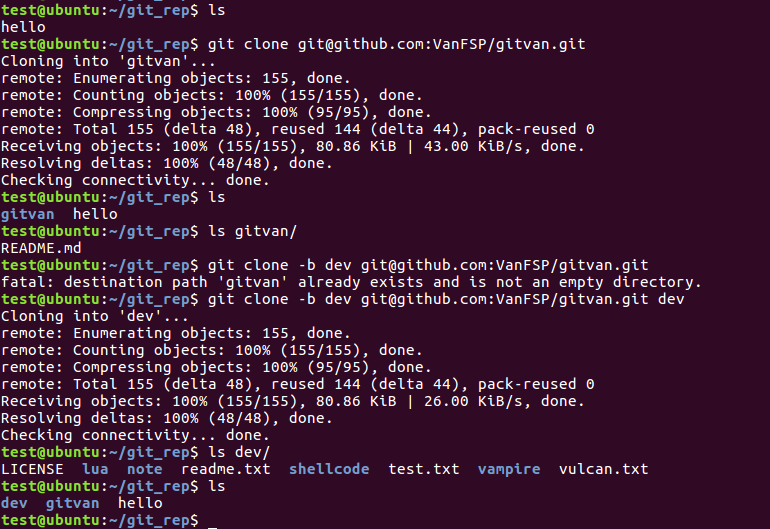
通过

git clone [git@github.com:VanFSP/gitvan.git](mailto:git@github.com:VanFSP/gitvan.git)

默认是将主分支master克隆到本地的工作区，并且文件名为gitvan

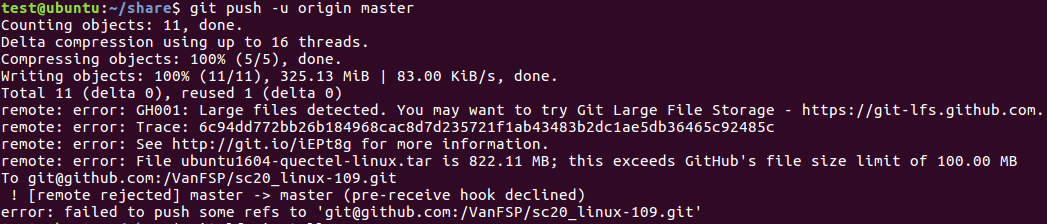
git clone -b dev [git@github.com:VanFSP/gitvan.git](mailto:git@github.com:VanFSP/gitvan.git) dev

将分支dev克隆到本地的工作区，并且文件名设为dev



1. 如何解决github不能push 大于100MB的文件的方案

首先看到一个错误：要推送的文件是822.11MB 远大于100MB。



到官网（https://git-lfs.github.com）查看到如下命令

curl -s

[http://packagecloud.io/install/repositories/github/git-lfs/script.deb.s h](http://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh) | sudo bash

