**Git学习之旅**

**讲师：Jack QQ：3434211960 Git学习交流群：699242826**

**咕泡学院官网:www.gupaoedu.com**

# 1 Git简介

学习一门技术老师更加倾向于看官网的。

度娘看完了，官网看完了，大家还是很懵逼

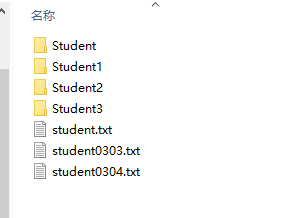
学生成绩管理系统

登录模块 3.2

登录模块进一步完善 缺一个验证码的功能 3.3

登录模块还有一些小bug需要处理 3.4

学生的添加模块 3.5



这绝对不是一个优秀的程序员该干出来的事情！

多人开发的时候（甲乙丙）

合作去开发一个购物网站

甲：订单处理模块

乙：商品管理模块

丙：其他模块

每个人单独在自己的系统中进行开发，这时候最终需要把单个分散的系统整合成一个完成的，怎么办？

Copy到一个人的机器上，大家眼睛都盯着这个电脑，然后啪啦啪啦copy。如果说在整合的时候，你发现哪个哥们他写的模块有问题。数据库的整合，订单模块里面的内容需要调用到商品模块的内容。

当你中招了之后，这是好事，犯了错，才能解决嘛

纵观整个软件开发历史，也有几十年的历史了吧，当你现在的遇到的一些问题，我们前辈们肯定遇到的。

既然前辈们遇到了，他们肯定有义务（想要）去解决。

**大纲并没有去罗列出来**

之前老师的Maven视频有18小节

两者切换起来大家看的时候比较不专注

# 2 Git官网

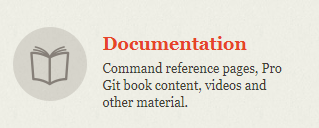
对于官网（尤其是这种英文官网，比较害怕），看了比较痛苦，主要是看不懂

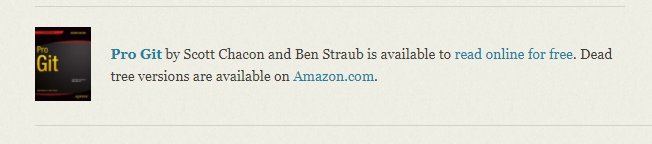
心里也知道看官网是最权威的，任何的其他途径的中文文档都是翻译过来，翻译就有一个麻烦的地方，有时候不准确，译者他翻译的结果达不到你自己内容的一个期望。

还是要尝试去看官网。

可以先有一个整体的认识，采用总-分-总的原则

开源-开放源代码 所有人都可以参与其中的开发，更加有利于一款软件或者技术的发展。

官网最权威的一个文档提供给大家学习



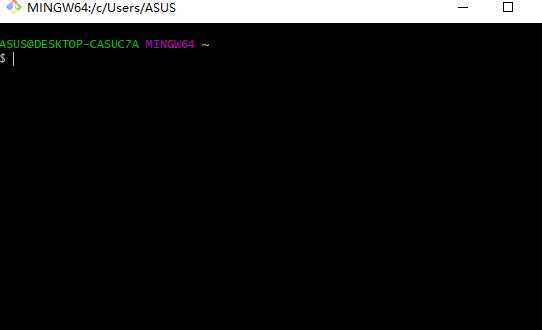
其实对于英文文档也不是那么难看懂嘛，实在遇不到不会的单词你查。

# 3 下载与安装

先从Windows入手，考虑到大家手上有的操作系统。

选择自己对应的操作系统，然后进行傻瓜式安装即可。

Windows



Linux中的安装

yum install git 啪啦啪啦进行安装嘛

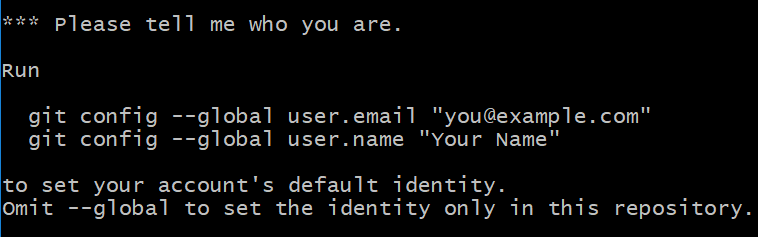
接下来的课程中我们会进行windows和linux中的命令使用。

# 4 Git初体验

通过前面三个小节的学习，我们已经基本上知道了git是干什么的，以及git的下载和安装等。

我们还想知道，git到底怎么去进行版本控制。

需求是：比如有一份文本文件，想要操作这个文本文件。我去不断的修改这个文件的内容，每次修改之后，我需要记录修改的历史记录。

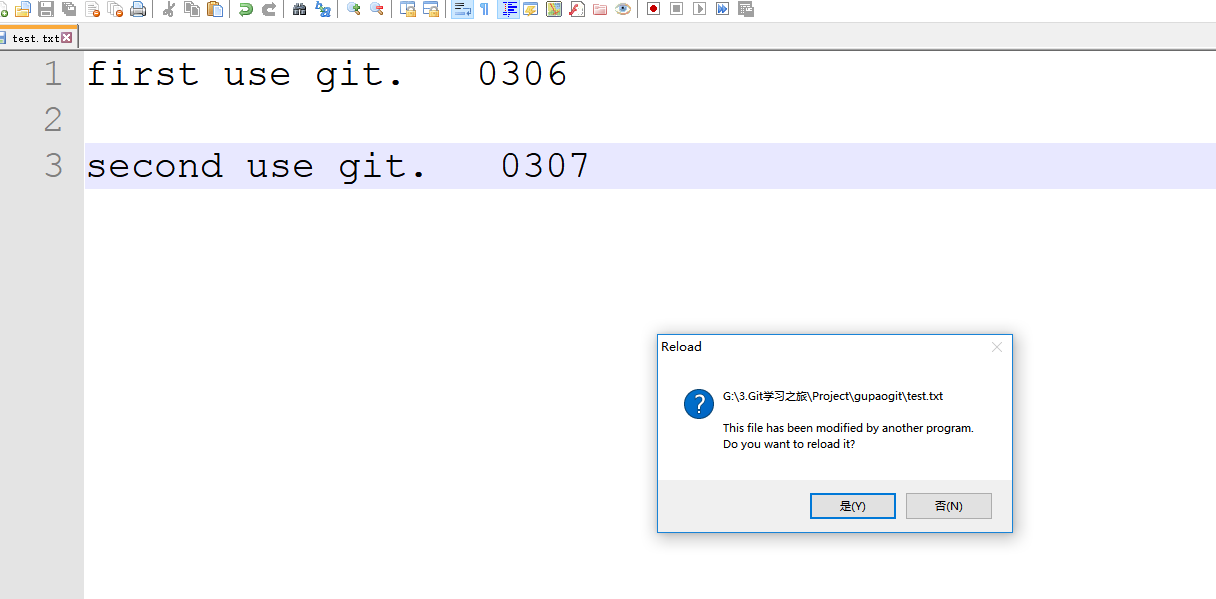


也就是说，在使用git进行版本管理之前，需要进行一个配置，这个配置是告诉git你的用户名以及你的邮件账号

小技巧：忘记命令的具体写法时，可以进行Tab键进行提示。

git add git commit 就可以对当前的一个版本进行控制

两天进行了修改，并且把每一次修改提价到了版本库，然后我的需求是，想要回退到3.6写的内容



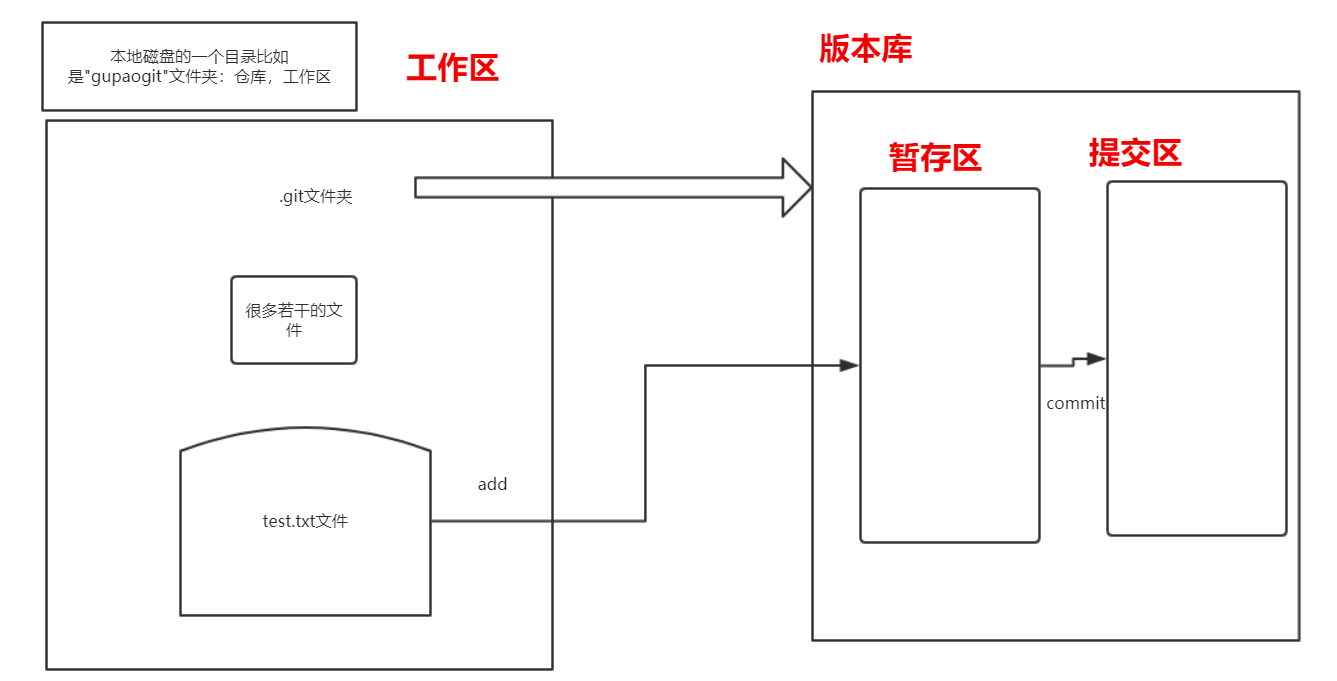
我利用这种思想，可以一个文件走天的。

时光穿梭，回退以前可以

那假如我想到“当下”，这时候怎么办呢？

# 5 Git学前知识

不妨先来画图，画图是老师的一个绝招，老师要用绝招给大家展示一些东西。



工作区就是你当前使用的一个目录，比如“gupaogit”文件夹

版本库就是add以及commit之后的乱七八糟的东西

使用git，步骤：

1. 下载与安装
2. git config --global user.email/name
3. git init 对一个文件夹目录进行git初始化操作，这个初始化就是为了让git对这个文件夹中的文件进行版本管理。

初始化完成之后，在当前目录下会有一个.git文件夹，这个文件夹就是我们所说的版本库。

现在就可以用git对其下的文件进行一个版本的控制，就是可以对它进行一个版本的回退，回到当前等等。

1. 接下来你的操作就是

无非在这个工作区或者仓库中创建文件，然后对文件进行一系列修改操作之类的。

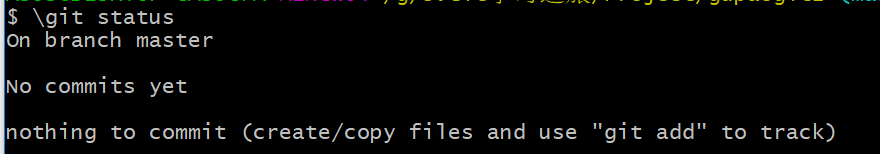
Git的要求是，你要让git帮你进行的版本的管理，必须先有一个文件夹，这个文件夹要被git进行初始化操作，这样才能进行一个git的版本管理。

# 6 Git基础操作

git status git add git commit

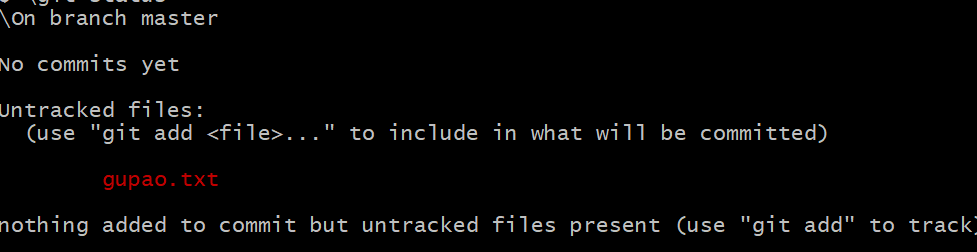
需求：在git工作区中新建一个文件，名称为“gupao.txt”。

git status:表示查看git工作区（仓库）中的一个状态

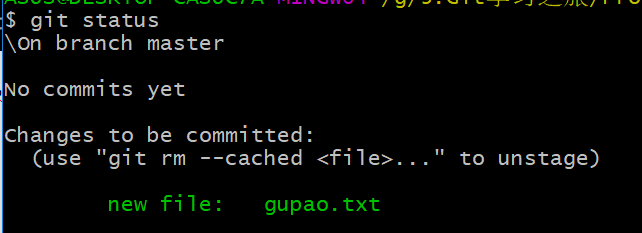
(1)nothing to commit

(2)当你新建了一个文件之后

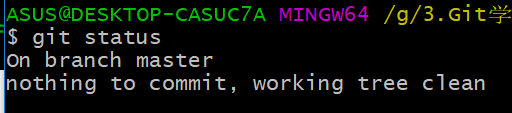
Untracked files: 未被追踪的文件



(3)当我们进行git add gupao.txt之后

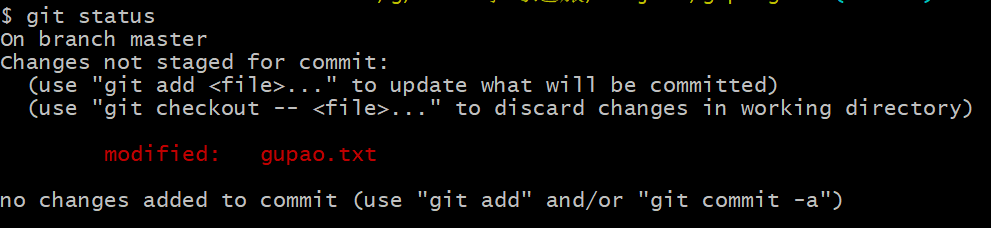


(4)当我们进行git commit -m “gupao first”



只有进入到提交区之后，我们的文件才能被git真正意义上管理起来

1. 对应我们创建一个新文件的时候的状态



git add:就是将工作区中的内容add到暂存区

git commit:将暂存区的内容commit到提交区

要想让git对你的文件内容进行版本管理，必须经过两个步骤，第一个是git add，第二个是git commit。

可以随时随地使用git status进行工作区中状态的查看。

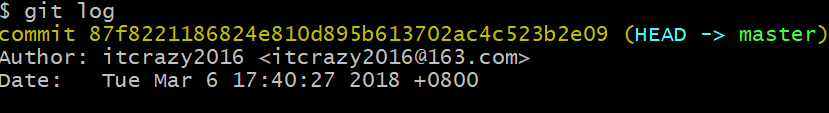
# 7 Git log和版本切换

大家都知道git可以进行版本控制与管理。

如果你想要进行版本的切换的话，你需要知道的就是这个commitid，也就是说你要知道每一次提交的信息才可以。

就要去查看这个每次的commitid是什么

Git log 日志，也就是说查看git的日志信息



Commit

Author :作者 就是你当初配置 git global --- user.name

Date：提交的日期

根据每次的commitid进行一个版本的切换。

git reset --hard commitid（这个id可以根据git log查看日志信息去找到）

我觉得这样打印出日志信息太多了，能不能简化一下呢？

git log --pretty=oneline

补充的知识点：

我觉得我就是想要回退到上一个版本，今天修改了一些东西

直接回退到上一个版本 git reset --hard HEAD^ 回退到上一个版本

Git reset --hard HEAD^^ 回退到上上个版本

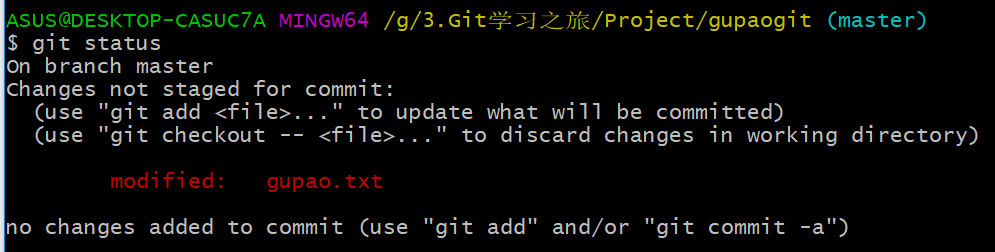
麻烦是，回退到之前的版本之后，最新的commitid没了，如果你的之前的操作窗口有，你复制过来进行

重点是要找到最新的一个版本的commitid即可。 git reflog

如果进行版本回退之后，发现commitid最新的没了，你可以使用git reflog进行所有的commitid的查询。

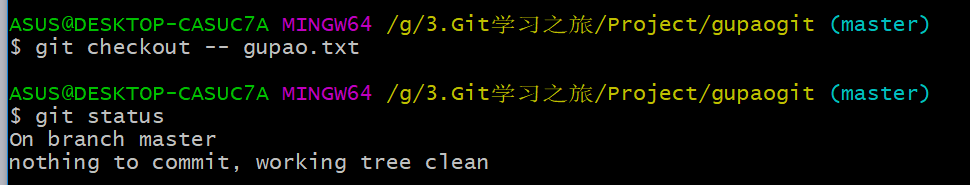
# 8 工作区暂存区和提交区

在工作区进行了修改，然后git status会有一个提示，告诉你下一步需要干嘛



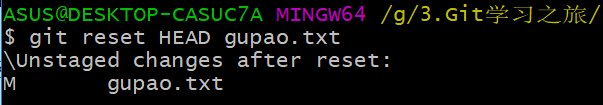
正常的思维，你会进行git add

后悔当前的操作，git checkout -- gupao.txt 会撤销工作区中的一个修改，也就是说你不需要进行git add操作了。



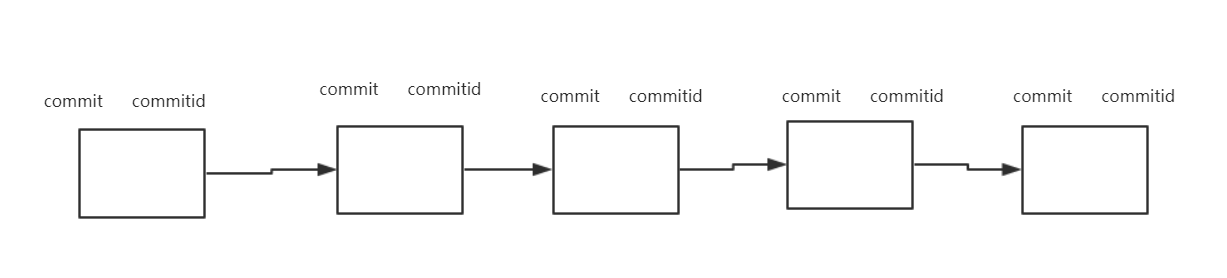
对于已经在暂存区的内容，使用git checkout -- gupao.txt 不可行了

**git reset HEAD gupao.txt 回到提交区中的最新版本**



接下来需要做的就是再次使用git checkout -- gupao.txt，将修改拉回到工作区，把工作区的修改内容清空

# 9 初识分支



把每一次的提交，都用线连起来，你会发现，很连贯。

C/C++ 指针的概念

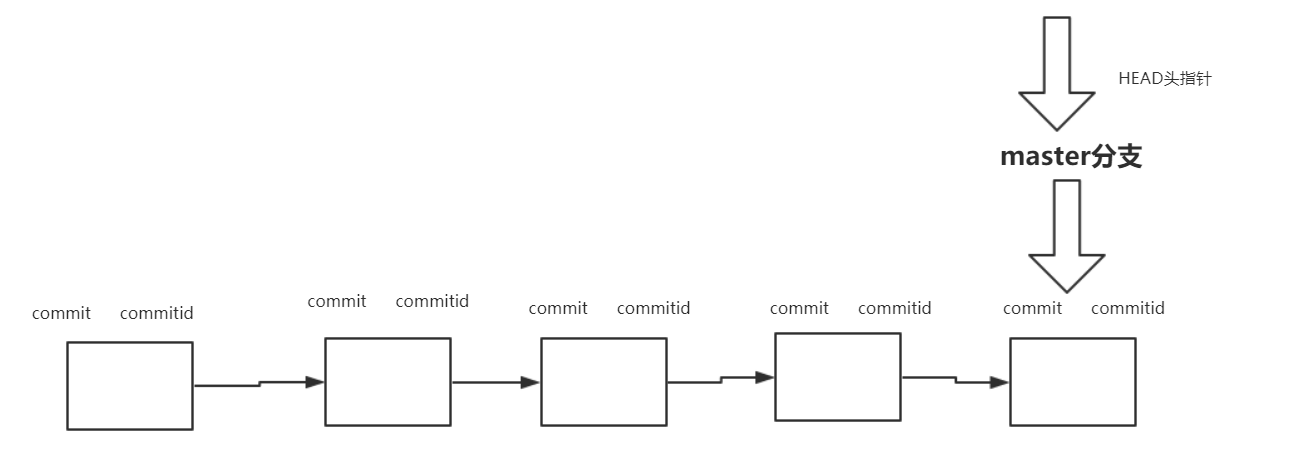
git reset --hard commitid

HEAD 如果说内容已经add到暂存区，此时要想撤销的话，需要先回到最新的一个commitid上 HEAD HEAD^ HEAD^^

HEAD就表示当前最新的版本的commitid，也就是最新的指针指向

Master分支是主分支，也是git为你默认创建的分支

一定就有其他分支



Master分支指向的是最新的提交commitid，头指向指向的是我们的master分支。

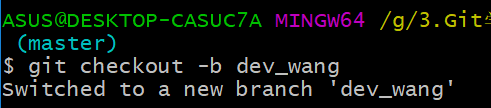
但是git的强大之处远不止如此，而是有一个多人协作开发的概念。

不仅是一个单人开发进行版本控制，同时可以多人协作开发进行一个版本控制。 就是通过git的分支进行的。

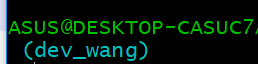
# 10 详解分支

在主分支master上切出一个开发分支，比如叫”dev\_wang”

git checkout -b dev\_wang; 创建一个分支

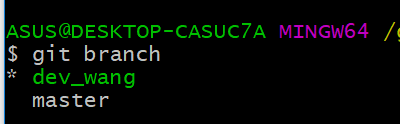


现在你所有的操作都是在一个新的分支上进行。



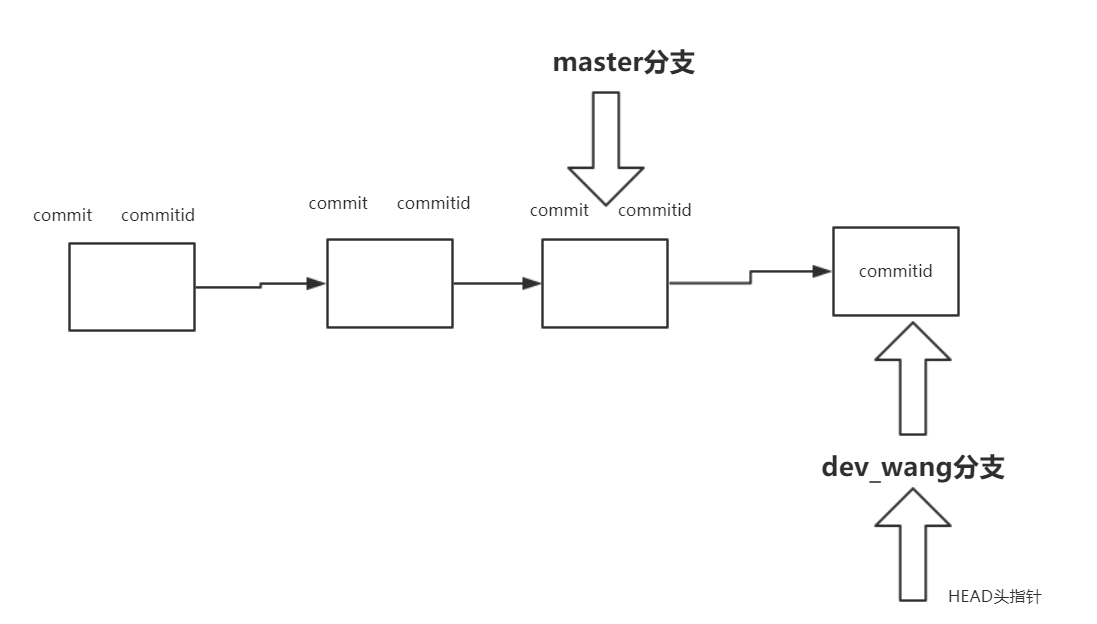
刚才一系列的操作都是在dev\_wang分支上进行了，现在再次切换到主分支

Git branch可以查看当前所有的分支情况，并且可以看到目前所处的分支(\*)

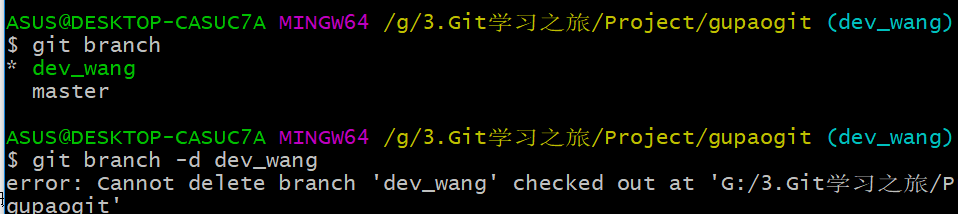


在不同的分支上进行的版本管理，只有在自己的分支上才会生效。

HEAD的一个补充：HEAD头指针指向的是当前分支最新的commitid



假设我们觉得一个分支没啥用了，想要删除该分支怎么办呢？



现在所处的分支是dev\_wang，想要删除它的话，你必须在其他分支上。

这时候你只能强制性的删除该分支

git branch -D dev\_wang

# 11 分支合并和冲突解决

主分支master我们不会去动，只会用于版本的发布。

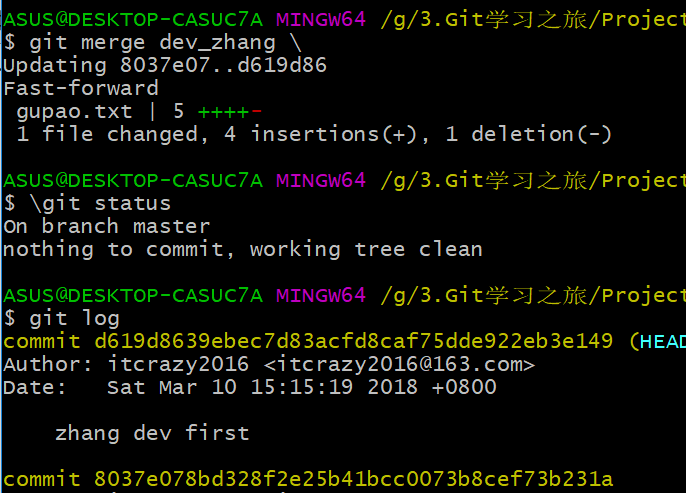
各自在自己的分支上开发完成之后，需要将开发的内容合并到主分支上去。这个时候成为merge。

需求:一个新的开发人员dev\_zhang

1. git checkout -b dev\_zhang;
2. 进行文件的修改并且add，commit
3. 此时在小张的分支上多了一个commitid，这时候需要把这个小张修改的内容进行版本的发布，就需要把小张的修改内容合并到master分支上。

切换到master分支，合并dev\_zhang的开发内容

合并的操作：快速合并 在master分支上，**git merge dev\_zhang;**



1. git branch -d dev\_zhang; //上一个小节我们这样去做的时候，它有一个不能删除的提示，这个提示是告诉你需要进行分支的合并。

当你的项目开发人员很多，2个，小张，小李，合作开发一个项目

需求：小张，小李合作开发一个项目，这个项目两个人负责不同的模块

小张：商品管理的模块

小李：订单管理的模块

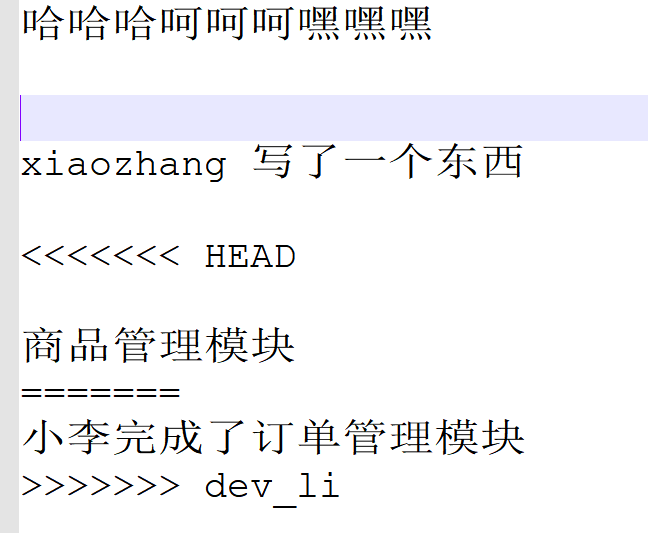
master分支作为一个版本发布的分支，不应该进行直接在上面开发

1. git checkout -b dev\_zhang
2. git checkout -b dev\_li
3. 分别在小张和小李的分支上进行开发之后

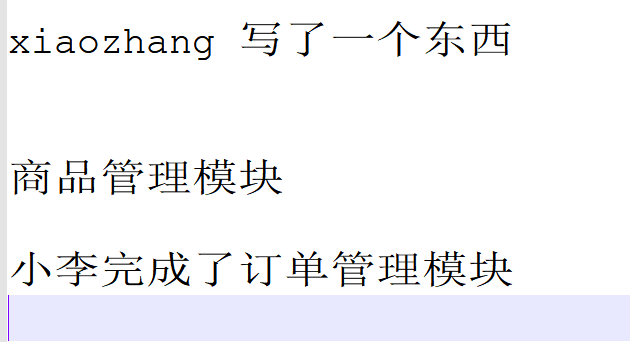
发现master分支上并没有小张小李的开发内容

1. 小张和小李的开发内容发布到master分支

分支的合并



合并的冲突的问题



需要手动解决冲突，并且再去进行add,commit的操作。

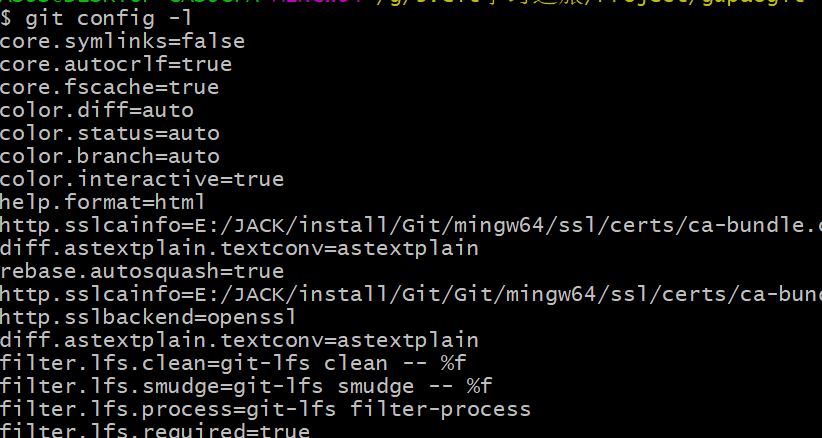
git checkout -b test

# 12 Git config和配置别名

在你刚装完git的时候，老师让你去配置了一个email name，只有这样你才能使用git

git config配置git的命令

git config -l 查看所有的配置信息



这里所有的信息实际上都是整合出来的，整合哪边的呢？

仓库级别，全局级别，系统级别

优先级是什么

所谓的仓库级别的配置：当前仓库级别下的.git>config文件

全局级别的：当前用户之下表示的是全局级别的

系统级别：在我们的git安装目录下etc

Git config --global --add user.name itcrazy2016

Git config --global --unset user.name

这个命令其实比较实用

Git status git add git commit

Git log --pretty=oneline

这些命令太长了，能不能简化？

Git st 表示查看用户状态

Git cm 提交commit

Git log one 查看一行信息？

配置别名？

git config --global alias.st status 表示用st代表status。

# 13 打标签和忽略文件

打标签：

Commitid不容易记住，能不能用一种比较独特的方式去记住每个版本

118.222.113.253 这样访问网站方便？

[www.baidu.com](http://www.baidu.com) 域名访问网站方便？

Commitid as2731aa <------> 标签

想要给最新版本的id打上一个标签 -----git tag v1 将最新版本的commitid对应上v1

查看一下当前仓库的标签列表 git tag

给之前已经错过的commitid去打上一个标签，怎么操作？

git tag v1pre d619d86

这个标签你看起来还是不够过瘾，你也可以给这个标签加上一个说明

Git tag v2.0 -m “这里打上了一个标签”

就得能够删除标签 git tag -d v1

**忽略文件**

创建了一个Person.class的类文件，发现git标注为Untracked file 未被追踪的文件

Git add Person.class 肯定是可以的

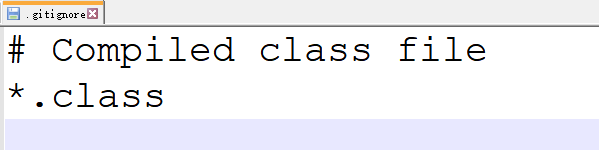
我希望这个文件不能被git管理起来

能不能让git将这个文件（这类文件）给忽略呢？

.class git你不要帮我去管理.class的类文件

仓库的根目录下创建一个.gitignore 文件

这个规则你要让git能够看懂



告诉git让他不要帮你去管理这个文件了

# 14 本地仓库和远程仓库

假设主分支master已经存在，这时候多了一个开发人员，你怎样给他一个分支？

每个人要至少在自己的电脑上进行开发？

而这个master分支仅仅用于版本的发布的。

用一台公共的电脑作为一个远程仓库，供每个开发者使用，每个开发者可以去创建各自独立的分支进行开发，开发完成之后，把各自的代码推到远程仓库上。

局域网：一台远程仓库 私有的公共仓库

项目：很小，想要让所有开发者都参与进来，那么就应该有这样一台公共的让所有人都能访问的远程仓库。同时这样做的好处是：可以更加方便的管理开源代码。 大家都一起使用的仓库。

很显然是存在，GitHub 国外的远程仓库 世界各地的开发者都可以在上面进行自己代码的管理

码云 国内的远程仓库

问题是，你把代码放到他们上面，除非你交一些会员费，代码才不能让别人看见，不然你提交到你账号上的代码都公开的。

搭建自己的私有仓库的话，让别人不可见，怎么做呢？Gitlab

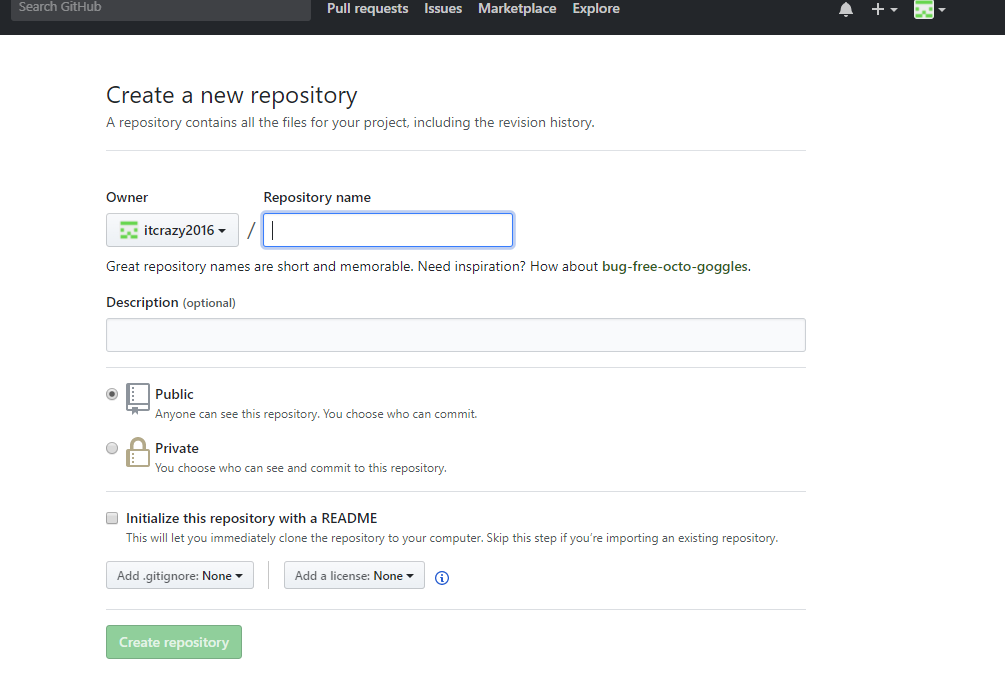
# 15 GitHub和码云

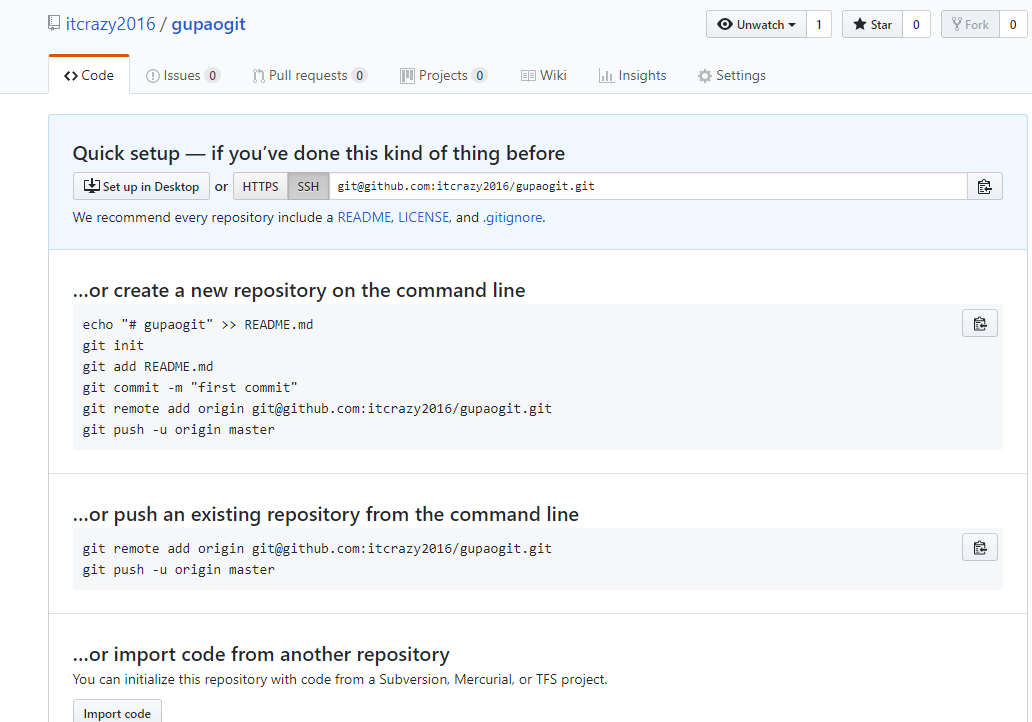
Github 远程仓库

Gupaogit 本地仓库

把本地仓库gupaogit 上传到github上去进行。

1. 在远程仓库github创建一个对应的项目比如gupaogit repository仓库





# 16 操作远程和本地仓库

push，pull和clone。

代码的内容推送 ，将本地的内容推送到具体的远程仓库上

本地的gupaogit -----> github/码云 gupaogit

1. 要让两者有关联

在本地仓库中配置一下它和远程仓库的关系，它俩是一家人。

当前本地仓库是否有远程仓库，如果有，那么它的远程仓库是什么？ git remote

本地仓库中:git remote add [origin](mailto:origin远程仓库的地址？git@github.com:itcrazy2016/gupaogit.git)**[远程仓库的地址？git@github.com:itcrazy2016/gupaogit.git](mailto:origin远程仓库的地址？git@github.com:itcrazy2016/gupaogit.git)**

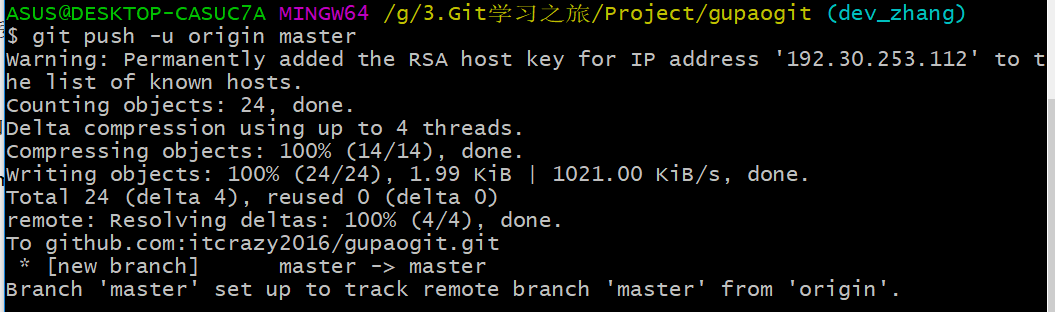
origin 本地仓库和远程仓库的地址进行一个关联

git push -u origin master 是推送代码，那么就不是指定远程仓库。

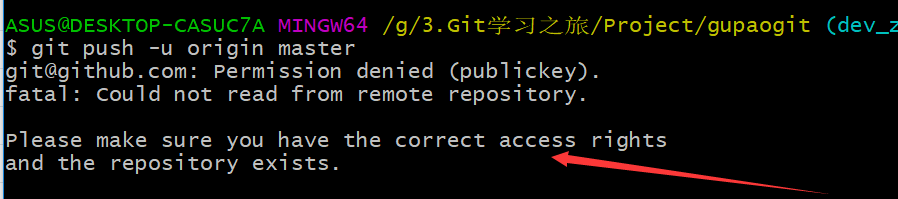
这样关联之后，接下来就是把代码推送到远程仓库上

1. 本地仓库和远程仓库进行关联之后，就可以进行的推送了

git push -u origin master 将本地仓库的内容推送到远程仓库



无论是推送还是拉代码，需要有这样一个权限，也就是说你目前使用的这台电脑是否安全，如果不安全，人家不会让你去推送



【没有权限的情况下】

需要添加权限

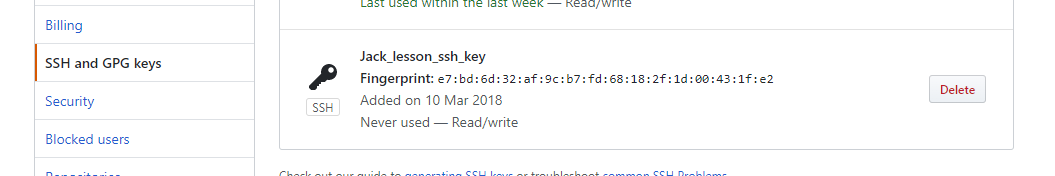
1. 需要在本地中生成一个ssh key

ssh-keygen -t rsa -C ["itcrazy2016@163.com"](mailto:\"itcrazy2016@163.com\") 在自己的计算中中增加一个安全ssh key

盖上了一个章，就表示你这个电脑认证后的ssh\_key

B.需要把这个key告诉github/码云

把公钥放到ssh key



保证数据传输的一个安全性



如果说是在其他分支进行的修改，需要进行一个分支的合并

要推送其他分支的，指定一下要推送的分支即可git push -u origin dev\_zhang;

新来了一个哥们，小王，需要进行一个开发

需要把远程仓库的代码拉倒本地进行开发

1. clone 克隆操作

前提是：sshkey也要添加完成

Git clone [git@github.com:itcrazy2016/gupaogit.git](mailto:git@github.com:itcrazy2016/gupaogit.git)

当远程仓库的版本内容如果和本地仓库的内容不一致，需要先git pull，把远程仓库的内容拉下来到最新版本才行，然后进行手动解决冲突

Gitlab 视频 咕泡学院官网

<http://bbs.gupaoedu.com/forum.php?mod=viewthread&tid=258&extra=page%3D1&_dsign=495dcb55>

如果大家后期反馈有需要老师帮忙录制一个视频操作的话，后面再给大家补上这个视频