Git版本控制工具

最流行的版本控制工具

讲师: 胡双
博客: https://hd1611756908.github.io/

目录页

1 版本控制的概念

1、版本控制最主要的功能就是追踪文件的变更

2、它将什么时候、什么人更改了文件的什么内容等信息忠实地了已录下来

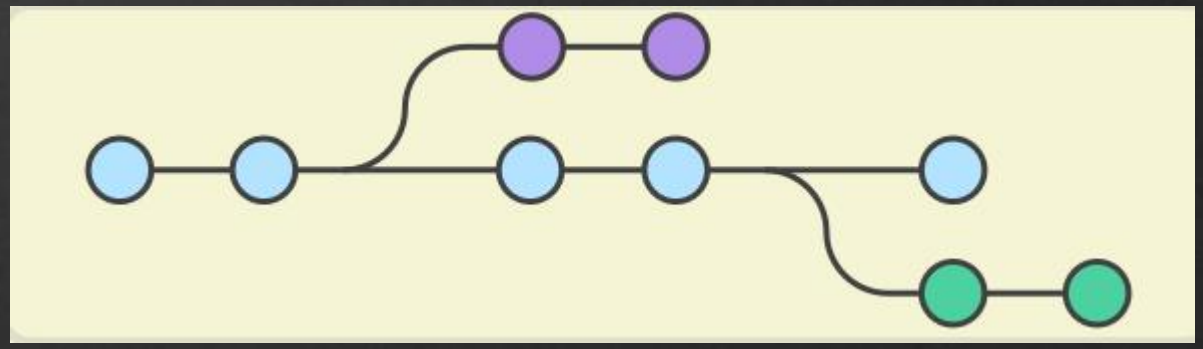
3、每一次文件的改变，文件的版本号都将增加

4、除了记录版本变更外，版本控制的另一个重要功能是并行开发

1 常用的版本控制工具

SVN

Git



2 常用的版本控制工具

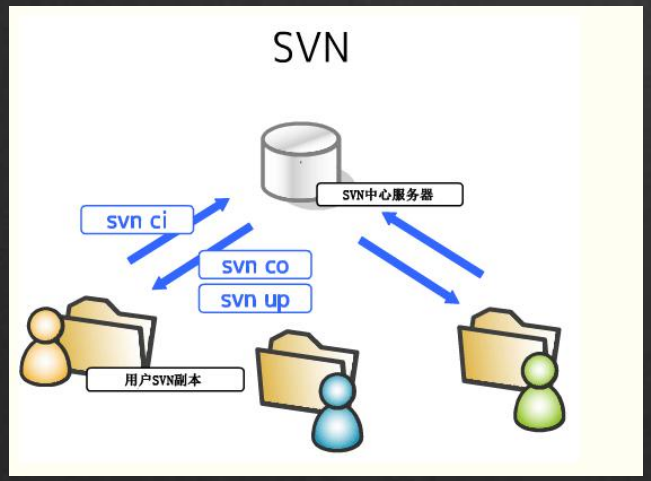
1、GIT是什么？

Git是目前世界上最先进的分布式版本控制系统

2、与SVN的区别

svn: 集中式、需联网

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



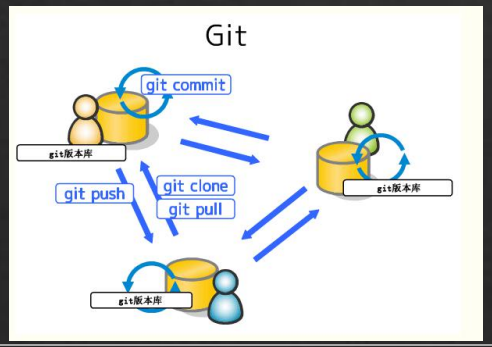
3 常用的版本控制工具

git: 分布式、离线

首先，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。既然每个人电脑上都有一个完整的版本库，那多个人如何协作呢？比方说你在自己电脑上改了文件A，你的同事也在他的电脑上改了文件A，这时，你们俩之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

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。

在实际使用分布式版本控制系统的时候，其实很少在两人之间的电脑上推送版本库的修改，因为可能你们俩不在一个局域网内，两台电脑互相访问不了，也可能今天你的同事病了，他的电脑压根没有开机。因此，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。



1 Git客户端的安装和配置

官网下载地址: https://git-scm.com/downloads

用户名和邮箱设置

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

1 Git的操作步骤

- 创建版本库 git init
- 创建一个文本文件并放在我们的版本库中或者是版本库的子目录中
注意:所有的版本控制软件只能追踪文本文件，不能追踪二进制文件
- 将创建的文件加入到暂存区(缓存) git add
- 将暂存区的文件提交到本地版本库 git commit -m “提交的注释”
- 继续修改文件，不提交，然后通过命令对比查看都修改了什么

使用 git diff <文件名> 命令查看不同

查看完有什么不同之后可以放心的提交了代码了

提交一个更新之后的文件和提交一个新文件步骤是一样的 两步

第一步: git add 在提交之前可以通过git status 查看一下文件的状态

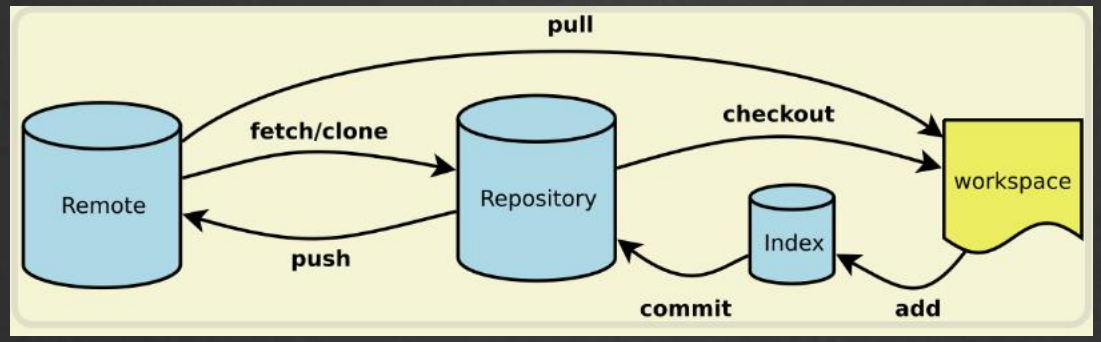
第二步: git commit -m ‘提交注释’ 提交之后再通过git status 查看文件状态，或者查看暂存区

1 Git的版本回退

- 查看提交记录　详细信息查询: git log　　精简信息查询: git log --pretty=oneline
- 回退到上一个版本　git reset --hard HEAD^　　回退到上上一个版本　git reset --hard HEAD^^
- 回退到上100个版本　git reset --hard HEAD~100
- 现在我回退后悔了我想还回到最新版本：只要你的窗口没有关，在上面还能找到刚刚的commit id就可以很轻松的回到最新版本　git reset --hard a1faca5
- 继续修改文件，不提交，然后通过命令对比查看都修改了什么

- 但是当我回退了上一个版本，但是电脑关了，第二天我想起来，不想回退了，我在想回来，但是窗口已经关了，commit id找不到了，怎么办　git reflog　：查看历史操作

1 Git的工作区和暂存区



- Workspace：工作区
- Index / Stage：暂存区
- Repository：仓库区（或本地仓库）
- Remote：远程仓库

1.工作区就是我们创建的自定义目录
2.暂存区是我们初始化自定义目录后里面生成的一个.git文件夹中的一部分
3.工作区里面的.git文件夹被称为版本库
暂存区 stage
git给我们默认创建的master分支
4.git的工作原理
(1) 创建一个新文件
(2) 将新文件git add 到暂存区
(3) git commit 将暂存区的文件提交到分支
5.git管理的是修改而非文件
总结:如果修改不被添加到暂存区就算commit也是不会被提交的

1 Git的撤销和修改

1、取消已经暂存的文件(取消暂存区域中的文件以及如何取消工作目录中已修改的文件)

将加入到暂存区中的文件修改从暂存区撤回来(对文件本身的内容修改没有影响)
git reset HEAD <file> (此条命令通过git status 可以查到)

2、取消对文件内容的修改

git checkout -- abc.txt

这条命令是危险的，执行完这条命令后所有此文件的修改内容都没有了。就像从来没有出现过一样；并且此次操作是不可逆的，因为修改过的内容还没有被加入到版本库中进行管理，所以不能通过版本控制操作找回原有数据。

命令 git checkout -- abc.txt 的意思就是把abc.txt文件在工作区的修改全部撤销，这里有两种情况：

一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态
一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态

总之，就是让这个文件回到最近一次git commit或git add时的状态

1 Git的删除操作

删除一个已经提交到版本库的文件
第一步: 首先删除本地库文件 rm abc.txt
第二步: 这时候使用 git status git就会检测到你本地的文件被删除了
第三步: 选择真删除还是假删除(恢复)
第四步: 真删除直接使用 git rm abc.txt 然后 git commit -m “” 完成删除
第五步: 本地库删错了(没关系，虽然本地库没有了，但是我们版本库里面还有，可以进行恢复)

$ git checkout -- abc.txt ：其实是用版本库里的版本替换工作区的版本

1 Git的远程仓库

我们使用github/码云作为测试的git的远程仓库（一般企业会自己搭建git服务器）
1、注册github账号(略)
2、密匙生成和配置
3、在github上创建一个仓库
4、将本地仓库和github上的远程仓库关
git remote add origin https://github.com/hd1611756908/testgithub.git

5、把本地库的所有内容推送到远程仓库
git push -u origin master

把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，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

只有第一次时才会添加 -u 命令以后直接推送 git push origin master

上面的每一次上github推送数据的时候都会跳出添加用户名和密码，这样比较麻烦，git提供了解决方案，通过添加秘钥的方式进行数据推送

2 Git的远程仓库

生成秘钥的方式:

第一步: $ ssh-keygen -t rsa -C "自己的邮箱地址"
第二步: 一般情况本地密匙会生成到
C:\Users\Administrator\.ssh 目录下，里面有公匙和私匙，私匙不能泄露，公匙可以对外使用
第三步: 将公匙id_rsa.pub里面的内容设置到github里面即可

注意: 可能设置好密匙以后，在向远程仓库push数据时还会让输入用户名和密码问题，不是设置的密匙没有生效，而是因为在进行本地库和远程库连接时使用的远程仓库的地址是https的地址，而ssh的密匙只针对于ssh提交有效，所以需要修改远程库地址为ssh方式

修改https为ssh的方式：

第一步: 找到本地仓库里面的.git文件夹(是一个隐藏文件)
第二步: 修改里面的config配置文件里的url属性值为ssh地址即可

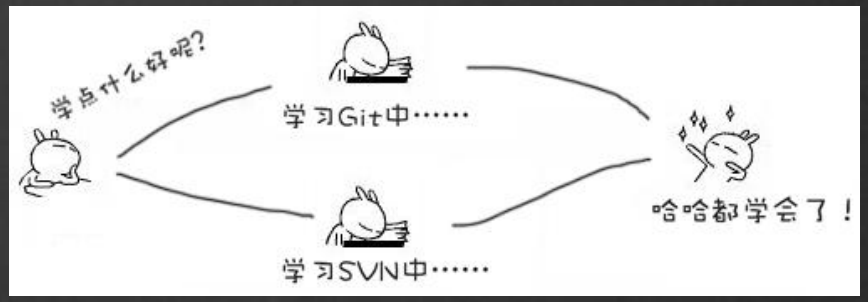
这时在向远程仓库提交就不会再让输入用户名和密码了。

6、克隆远程仓库到本地

git clone 仓库地址

1 Git的分支管理

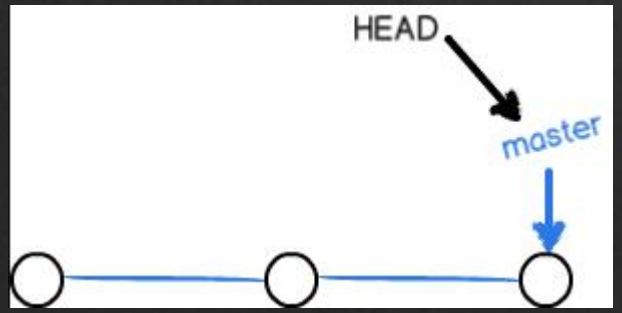
分支就是科幻电影里面的平行宇宙，当你正在电脑前努力学习Git的时候，另一个你正在另一个平行宇宙里努力学习SVN。如果两个平行宇宙互不干扰，那对现在的你也没啥影响。不过，在某个时间点，两个平行宇宙合并了，结果，你既学会了Git又学会了SVN！



1、分支的概念

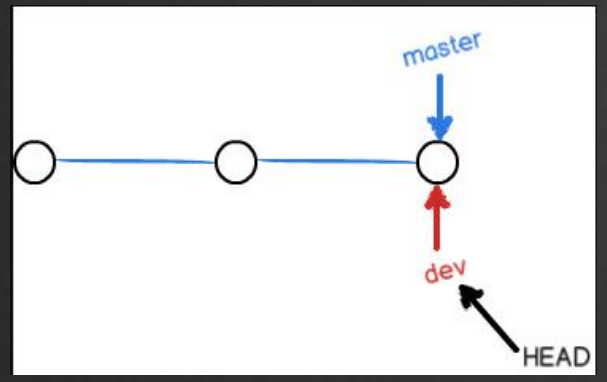
在版本回退里，你已经知道，每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支。截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

一开始的时候，master分支是一条线，Git用master指向最新的提交，再用HEAD指向master，就能确定当前分支，以及当前分支的提交点：



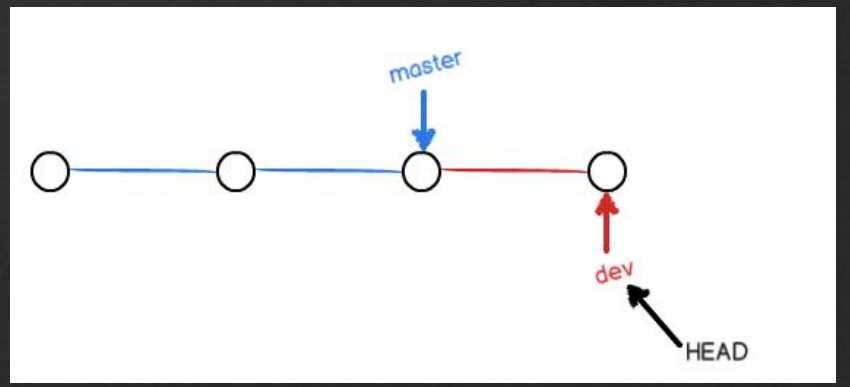
每次提交，master分支都会向前移动一步，这样，
随着你不断提交，master分支的线也越来越长

2 Git的分支管理

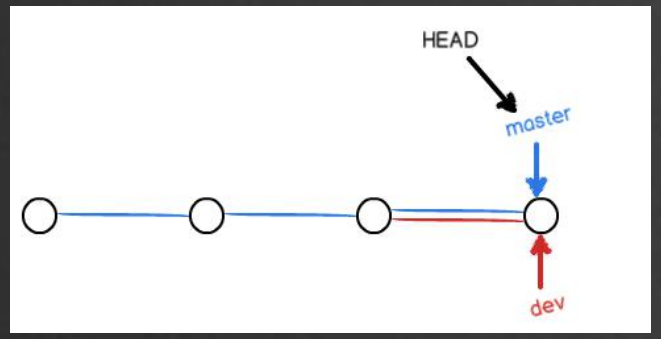


当我们创建新的分支，例如dev时，Git新建了一个指针叫dev，
指向master相同的提交，再把HEAD指向dev，就表示当前分支在dev上

Git创建一个分支很快，因为除了增加一个dev指针，改改HEAD的指向，
工作区的文件都没有任何变化,不过，从现在开始，对工作区的修改和
提交就是针对dev分支了，比如新提交一次后，dev指针往前移动一步，
而master指针不变.

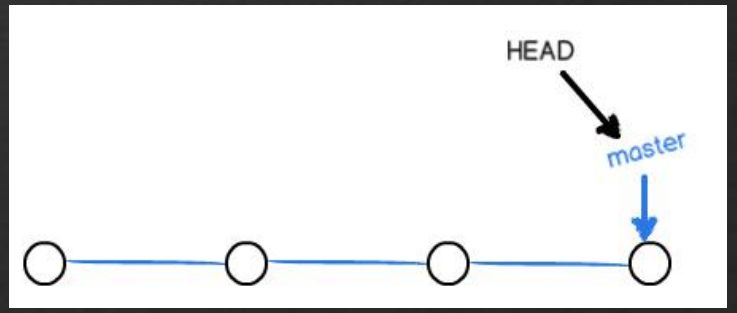


3 Git的分支管理



假如我们在dev上的工作完成了，就可以把dev合并到master上。
Git怎么合并呢？最简单的方法，就是直接把master指向dev的当前提交，就完成了合并

合并完分支后，甚至可以删除dev分支。
删除dev分支就是把dev指针给删掉，
删掉后，我们就剩下了一条master分支



4 Git的分支管理

2、分支创建

创建dev分支，然后切换到dev分支　　$ git checkout -b dev

git checkout命令加上-b参数表示创建并切换，相当于以下两条命令：

```
$ git branch dev
$ git checkout dev
```

5 Git的分支管理

3、查看当前分支

用git branch命令查看当前分支

```
$ git branch
* dev
  master
```

git branch命令会列出所有分支，当前分支前面会标一个*号

然后，我们就可以在dev分支上正常进行文件的修改和提交了

比如给我们的文件gittest.txt文件中添加一行数据 ANCDEFG 然后提交。

如果当前分支的工作做完了，想切换到master分支，这时需要进行分支切换。

6 Git的分支管理

4、切换分支　　$ git checkout master

切换回master分支后，gittest.txt文件，刚才添加的内容不见了！因为那个提交是在dev分支上，
而master分支此刻的提交点并没有变.
当我们dev分支工作完成之后，要将dev分支合并到master分支上来

5、分支合并　　$ git merge dev
git merge命令用于合并指定分支到当前分支。合并后，再查看gittest.txt的内容，就可以看到，
和dev分支的最新提交是完全一样的
合并完成后，就可以放心地删除dev分支了：　$ git branch -d dev
删除后，查看branch，就只剩下master分支了

```
$ git branch
* master
```

7 Git的分支管理

6、冲突解决　人生不如意之事十之八九，合并分支往往也不是一帆风顺的。

准备新的feature1分支，继续我们的新分支开发　$ git checkout -b feature1

gittest.txt最后一行，改为：Creating a new branch is quick AND simple

在feature1分支上提交：

$ git add readme.txt

$ git commit -m "AND simple"

切换到master分支：

$ git checkout master

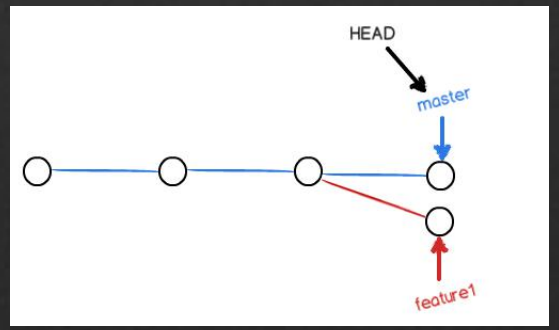
在master分支上把readme.txt文件的最后一行改为：Creating a new branch is quick & simple.

提交：

$ git add readme.txt

$ git commit -m "& simple"

现在，master分支和feature1分支各自都分别有新的提交，变成了这样：



8 Git的分支管理

这种情况下，Git无法执行“快速合并”，只能试图把各自的修改合并起来，但这种合并就可能会有冲突：

```
$ git merge feature1
Auto-merging readme.txt
CONFLICT (content): Merge conflict in readme.txt
Automatic merge failed; fix conflicts and then commit the result.
```

可以使用git status查看冲突的位置

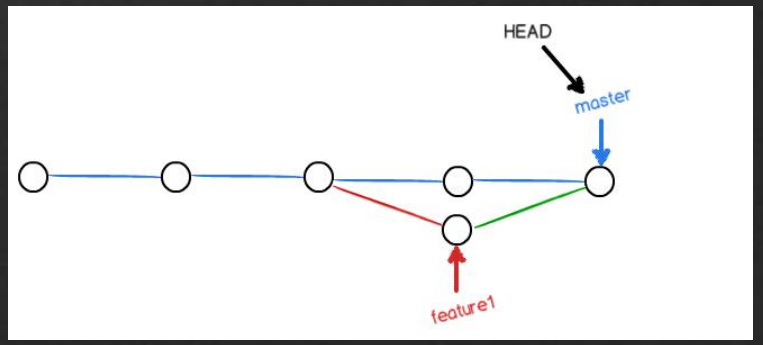
然后打开我们的gittest.txt文件解决冲突

```
Git is a distributed version control system.
Git is free software distributed under the GPL.
Git has a mutable index called stage.
Git tracks changes of files.
<<<<<<< HEAD
Creating a new branch is quick & simple.
=======
Creating a new branch is quick AND simple.
>>>>>>> feature1
```

Git用<<<<<<<，=======，>>>>>>>标记出不同分支的内容，我们修改之后保存然后在提交。

现在，master分支和feature1分支变成了下图所示

最后，删除feature1分支： $ git branch -d feature1



9 Git的分支管理

7、推送分支到远程

（1）查看远程分支

$ git remote 或者，用git remote -v显示更详细的信息

（2）推送分支

推送分支，就是把该分支上的所有本地提交推送到远程库。推送时，要指定本地分支，这样，Git就会把该分支推送到远程库对应的远程分支上

$ git push origin master

如果要推送其他分支，比如dev，就改成 $ git push origin dev

强制向远程仓库分支推送数据 $ git push -f origin dev

（3）拉取更新 $ git pull

注意: 在向远程仓库push(提交)代码之前一定要先pull(拉取最新的代码)，避免过多的冲突

8、团队协作

步骤一: 注册github账号

步骤二: 创建组织,注册团队

步骤三: 要求其他github开发者到组织团队中(也可以后邀请)

步骤四: 创建仓库(完成)

1 GIT 的 .gitignore文件

1、这是一个忽略文件
2、就是将系统中不需要提交的文件，忽略掉，不让他提交到git服务器上
3、不需要我们自己去写，有模板使用，只需要我们简单修改即可
地址: https://github.com/github/gitignore/ 跳转

谢谢观看

作者 胡双

邮箱 hd1611756908@163.com

QQ