**Git常用命令**

请确保已经安装里git客户端

**一般配置**

git --version //查看git的版本信息

git config --global user.name //获取当前登录的用户

git config --global user.email //获取当前登录用户的邮箱

**登录git**

/\* 如果刚没有获取到用户配置，则只能拉取代码，不能修改 要是使用git，你要告诉git是谁在使用\*/

git config --global user.name 'userName' //设置git账户，userName为你的git账号，

git config --global user.email 'email'

**创建一个文件夹**

mkdir nodejs //创建文件夹nodejs

cd nodejs //切换到nodejs目录下

**初始化git仓库**

git init //在nodejs文件夹下初始化一个仓库，此时文件里会到一个.git的隐藏文件夹

**创建忽略文件**

touch .gitignore //不需要服务器端提交的内容可以写到忽略文件里

/\*

.git

.idea

\*/

**查看目录**

ls -al

**创建文件并写入内容**

* 如果文件不存在则会创建文件
* echo "hello git"

> index.html //将'hello git' 写入到index.html中

单个>箭头表示写入， >>表示追加

**查看文件内容**

cat index.html

**增加到暂存区中**

git add index.html

git add -A //全部添加到缓存区

**增加到版本库中**

git commit -m '备注信息'

**查看版本**

git log --oneline

**比较差异**

* 比较的是暂存区和工作区的差异

git diff

* 比较的是暂存区和历史区的差异

git diff --cached

* 比较的是历史区和工作区的差异（修改）

git diff master

**撤回内容**

(如果修改了工作区的文件后发现改错了，可以用暂存区或者版本库里的文件替换掉工作区的文件)

* 用暂存区中的内容或者版本库中的内容覆盖掉工作区

git checkout index.html

**取消增加到暂存区的内容（添加时）**

git reset HEAD index.html

//显示目录的状体 有没有添加或者修改文件

git status

**删除本地文件**

rm fileName

**删除暂存区**

* 保证当前工作区中没有index.html

git rm index.html --cached

使用--cached 表示只删除缓存区中的内容

**回滚版本**

* 回滚最近的一个版本 git log

git reset --hard HEAD/commit\_id

**回滚到未来**

git reflog

**分支管理**

**创建分支**

git branch dev

**切换分支**

git checkout dev

**创建分支并切换分支**

git checkout -b dev

**删除分支**

git branch -d dev

**在分支上提交新的版本**

git commit -a -m 'dev1'

**合并分支**

git merge dev

**分支的合并后显示log**

git log --oneline --graph --decorate

**在分支开发的过程中遇到其他问题需要切换其他分支**

* 保留写好的内容在切换到主干
* 保留内容

git stash

**在次切换分之后需要应用一下保留的内容**

git stash apply

**丢掉保存的内容**

git stash drop

**使用并丢掉**

git stash pop

**最佳分支**

-有的时候开发需要合并指定的内容，而不是合并所有的提交，所以我们需要挑选最好的，自己生产版本

**合并分支把树杈掰到主干上**

git rebase

**添加远程的仓库**

**push -u**

-u参数 upstream

git push origin master -u //获取最新代码

**连接远程仓库**

git remote add origin 仓库的地址

**查看远程仓库**

git remote -v

**删除远程仓库**

git remote rm origin

**git常用命令**

**安装及配置：**

Ubuntu下安装：sudo apt-get install git  
配置用户名：git config --global user.name "你的名字"  
配置e-mail：git config --global user.email "你的邮箱@xx.com"

**与添加有关的：**

将当前目录变为仓库：git init  
将文件添加到暂存区：git add 文件名 [可选：另一个文件名]  
将暂存区提交到仓库：git commit –m "描述"

**与查询有关的：**

查询仓库状态：git status  
比较文件差异（请在git add之前使用）：git diff 文件名  
查看仓库历史记录(详细)：git log  
查看仓库历史记录(单行)：git log --pretty=online 或 git log --online  
查看所有版本的commit ID：git reflog

**与撤销有关的：**

撤销工作区的修改：git checkout -- 文件名  
撤销暂存区的修改：git reset HEAD 文件名  
回退到历史版本：git reset --hard 该版本ID  
回退到上个版本：git reset --hard HEAD^  
上上版本是HEAD^^，也可用HEAD~2表示，以此类推

**与标签有关的：**

为当前版本打标签：git tag 标签名  
为历史版本打标签：git tag 标签名 该版本ID  
指定标签说明：git tag –a 标签名 –m "标签说明" [可选：版本ID]  
查看所有标签：git tag  
查看某一标签：git show 标签名  
删除某一标签：git tag –d 标签名

**与GitHub有关的：**

**先有本地库，后有远程库，将本地库push到远程库**

关联本地仓库和GitHub库：git remote add origin 网站上的仓库地址  
第一次将本地仓库推送到GitHub上：git push –u origin master

**先有远程库，后有本地库，从远程库clone到本地库**

从远程库克隆到本地：git clone 网站上的仓库地址

网站地址可以选择HTTPS协议（https://github.com...）、SSH协议（git@github.com...）。  
如果选择SSH协议，必须将Ubuntu的公钥添加到GitHub上。见下一步

**SSH Key**

生成SSH Key：ssh-keygen –t rsa –C "你的邮箱@xx.com"  
生成Key时弹出选项，回车选择默认即可。  
Key保存位置：/root/.ssh  
登陆GitHub，创建new SSH key，其内容为/root/.ssh/id\_rsa.pub中文本

**已经有了本地库和远程库，二者实现同步**

本地库的改动提交到远程库：git push origin master  
更新本地库至远程库的最新改动：git pull

**Git常用命令**

请确保已经安装里git客户端

**一般配置**

git --version //查看git的版本信息

git config --global user.name //获取当前登录的用户

git config --global user.email //获取当前登录用户的邮箱

**登录git**

/\* 如果刚没有获取到用户配置，则只能拉取代码，不能修改 要是使用git，你要告诉git是谁在使用\*/

git config --global user.name 'userName' //设置git账户，userName为你的git账号，

git config --global user.email 'email'

**创建一个文件夹**

mkdir nodejs //创建文件夹nodejs

cd nodejs //切换到nodejs目录下

**初始化git仓库**

git init //在nodejs文件夹下初始化一个仓库，此时文件里会到一个.git的隐藏文件夹

**创建忽略文件**

touch .gitignore //不需要服务器端提交的内容可以写到忽略文件里

/\*

.git

.idea

\*/

**查看目录**

ls -al

**创建文件并写入内容**

* 如果文件不存在则会创建文件
* echo "hello git"

> index.html //将'hello git' 写入到index.html中

单个>箭头表示写入， >>表示追加

**查看文件内容**

cat index.html

**增加到暂存区中**

git add index.html

git add -A //全部添加到缓存区

**增加到版本库中**

git commit -m '备注信息'

**查看版本**

git log --oneline

**比较差异**

* 比较的是暂存区和工作区的差异

git diff

* 比较的是暂存区和历史区的差异

git diff --cached

* 比较的是历史区和工作区的差异（修改）

git diff master

**撤回内容**

(如果修改了工作区的文件后发现改错了，可以用暂存区或者版本库里的文件替换掉工作区的文件)

* 用暂存区中的内容或者版本库中的内容覆盖掉工作区

git checkout index.html

**取消增加到暂存区的内容（添加时）**

git reset HEAD index.html

//显示目录的状体 有没有添加或者修改文件

git status

**删除本地文件**

rm fileName

**删除暂存区**

* 保证当前工作区中没有index.html

git rm index.html --cached

使用--cached 表示只删除缓存区中的内容

**回滚版本**

* 回滚最近的一个版本 git log

git reset --hard HEAD/commit\_id

**回滚到未来**

git reflog

**分支管理**

**创建分支**

git branch dev

**切换分支**

git checkout dev

**创建分支并切换分支**

git checkout -b dev

**删除分支**

git branch -d dev

**在分支上提交新的版本**

git commit -a -m 'dev1'

**合并分支**

git merge dev

**分支的合并后显示log**

git log --oneline --graph --decorate

**在分支开发的过程中遇到其他问题需要切换其他分支**

* 保留写好的内容在切换到主干
* 保留内容

git stash

**在次切换分之后需要应用一下保留的内容**

git stash apply

**丢掉保存的内容**

git stash drop

**使用并丢掉**

git stash pop

**最佳分支**

-有的时候开发需要合并指定的内容，而不是合并所有的提交，所以我们需要挑选最好的，自己生产版本

**合并分支把树杈掰到主干上**

git rebase

**添加远程的仓库**

**push -u**

-u参数 upstream

git push origin master -u //获取最新代码

**连接远程仓库**

git remote add origin 仓库的地址

**查看远程仓库**

git remote -v

**删除远程仓库**

git remote rm origin