

1

```
[jinxianhua:home rimi$ du -sh d*
    0B    dim
    0B    dom
```

2

```
[jinxianhua:home rimi$ ls s*
sqwe:

ss:
asd
```

3

```
jinxianhua:Desktop rimi$ ls
P1901lesson    kobe.log          nginx-1.12.2.tar.gz
kobe            nginx-1.12.2      rose.log
jinxianhua:Desktop rimi$ rm -i *.log
remove kobe.log? y
remove rose.log? y
jinxianhua:Desktop rimi$ ls
P1901lesson    kobe              nginx-1.12.2      nginx-1.12.2.tar.gz
jinxianhua:Desktop rimi$
```

4

cp

-u --update 使用这项参数之后，只会在源文件的修改时间(Modification Time)较目的文件更新时，或是名称相互对应的目的文件并不存在，才复制文件

-n --no-clobber 不要覆盖已存在的文件

5

find ~ -name '*.py' type f -exec ls -l {} \; > res.txt

```
jinxianhua:~ rimi$ cd Desktop
jinxianhua:Desktop rimi$ > nbastar
jinxianhua:Desktop rimi$ ls -l *.py > nbastar
ls: *.py: No such file or directory
jinxianhua:Desktop rimi$ > kobe.py
jinxianhua:Desktop rimi$ > rose.py
jinxianhua:Desktop rimi$ > ai.py
jinxianhua:Desktop rimi$ ls -l *.py > nbastar
jinxianhua:Desktop rimi$
```

6

```
[jinxianhua:~ rimi$ ls / | xargs -n 3
Applications Library Network
System Users Volumes
bin cores dev
etc home installer.failurerequests
net private sbin
tmp usr var
```

7

```
jinxianhua:/ rimi$ top | head -3
Processes: 287 total, 2 running, 285 sleeping, 1136 threads
2019/03/20 19:55:54
Load Avg: 2.56, 2.54, 2.42
```

8

该1磁盘已经挂载 重新挂载

9

ftp 20 21

http 80

mysql 3306

ssh服务端端口为22，ssh免密方式原理：



例：

```
ssh-keygen
```

```
ssh-copy-id p1901@10.2.1.78
```

```
ssh p1901@10.2.1.78
```

10:

权限755: 7 111 用户读写执行, 5 101 用户组读执行, 5 101 其他读执行 改所有w权限 `chmod a-w 文件名`

Git 作业

1

`git add` 和 `git stage` 没有区别, 都是将修改添加到暂存区

2

`git rm --cached` 当我们需要删除暂存区或分支上的文件, 但本地又需要使用, 只是不希望这个文件被版本控制, 可以使用 `git rm --cached` `git rm --cached` 会从index里面删除该文件, 下次commit的时候会修改git仓库, 但是本地的文件还是保留

`Git rm -f`: 要删除之前修改过且已经放到暂存区的文件, 则必须使用强制删除, 即 `git rm -f` 这是一种安全特性, 用于防止误删还没有添加的数据, 这样的数据不能被git 恢复

git不仅仅是个版本控制系统，它也是个内容管理系统(CMS),工作管理系统等

1.GIT是分布式的，SVN不是：

这是GIT和其它非分布式的版本控制系统，例如SVN，CVS等，最核心的区别。如果你能理解这个概念，那么你就已经上手一半了。需要做一点声明，GIT并不是目前第一个或唯一的分布式版本控制系统。还有一些系统，例如[Bitkeeper](#), [Mercurial](#)等，也是运行在分布式模式上的。但GIT在这方面做的更好，而且有更多强大的功能特征。

GIT跟SVN一样有自己的集中式版本库或服务器。但，GIT更倾向于被使用于分布式模式，也就是每个开发人员从中心版本库/服务器上check out代码后会在自己的机器上克隆一个自己的版本库。可以这样说，如果你被困在一个不能连接网络的地方时，就像在飞机上，地下室，电梯里等，你仍然能够提交文件，查看历史版本记录，创建项目分支，等。对一些人来说，这好像没多大用处，但当你突然遇到没有网络的环境时，这个将解决你的大麻烦。

同样，这种分布式的操作模式对于开源软件社区的开发来说也是个巨大的恩赐，你不必再像以前那样做出补丁包，通过email方式发送出去，你只需要创建一个分支，向项目团队发送一个推请求。这能让你的代码保持最新，而且不会在传输过程中丢失。[GitHub.com](#)就是一个这样的优秀案例。

有些谣言传出来说是subversion将来的版本也会基于分布式模式。但至少目前还看不出来。

2.GIT把内容按元数据方式存储，而SVN是按文件：

所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn,.cvs等的文件夹里。如果你把.git目录的体积大小跟.svn比较，你会发现它们差距很大。因为,.git目录是处于你的机器上的一个克隆版的版本库，它拥有中

心版本库上所有的东西，例如标签，分支，版本记录等。

3.GIT分支和SVN的分支不同：

分支在SVN中一点不特别，就是版本库中的另外的一个目录。如果你想知道是否合并了一个分支，你需要手工运行像这样的命令 [svn propget svn:mergeinfo](#)，来确认代码是否被合并。感谢Ben同学指出这个特征。所以，经常会发生有些分支被遗漏的情况。

4

```
git log --since='2018-10-1' --before='2018-10-20' --no-merges --graph
```

5

git init 是在本地初始化一个空的仓库

git clone 初始化远程仓库

6 每次提交都忽略.idea文件夹里面的东西怎么办

7

git reset --soft 要撤销的id

8 如何检出标签？

Git可以给历史中的某一个提交打上标签，以示重要。它很像一个不会改变的分支——只是一个特定提交的引用。比较有代表性的是人们会使用这个功能来标记发布结点(v1.0.0等)。使用命令git tag，可以查看所有标签。如果要定义标签，只需要在git tag后添加要定义的标签名。例如：git tag v1.0.0。

默认情况下，git push命令并不会传送标签到远程仓库服务器上。在创建完标签后，你必须显示地推送标签到共享服务器上。

```
$ git tag v1.0.0
```

```
$ git push origin v1.0.0
```

```
Total 0 (delta 0), reused 0 (delta 0)
```

```
To github.com:benben/testBranch.git
```

```
* [new tag]      v1.0.0 -> v1.0.0
```

在标签创建完成后，你在上线之前又做过多次修改。这时你可以检出你之前那个准备发布的版本(打标签的版本)进行部署。

Git中不能真的检出一个标签，因为他们并不能像分支一样来回移动。如果想要工作目录与仓库中特定地标签版本完全一致，可以使用`git checkout -b [分支名] [标签名]`在特定地标签上创建一个新分支。例如：

```
$ git checkout -b version1 v1.0.0  
Switched to a new branch 'version1'
```

9

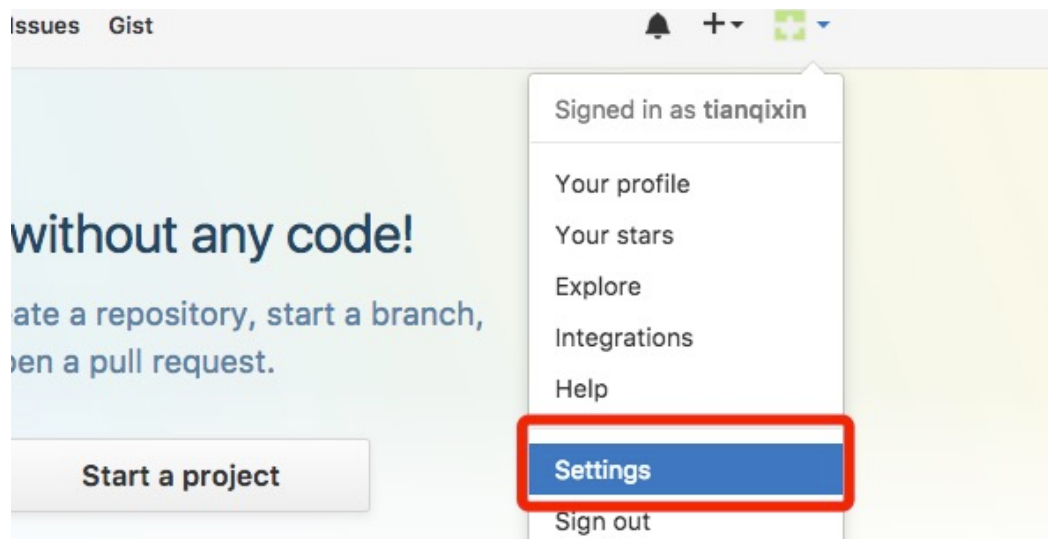
`git fetch`：相当于是从远程获取最新版本到本地，不会自动merge

`git pull`：相当于是从远程获取最新版本并merge到本地

10

添加远程仓库

```
git remote add 随便取一个名字  
ssh-keygen -t rsa -C "youremail@example.com"
```



SSH keys

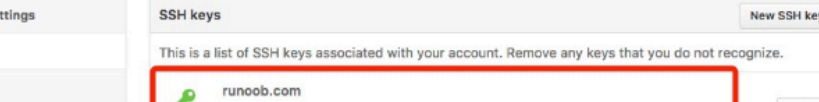
New SSH key

Title

Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key





Personal settings

- Profile
- Account
- Emails
- Notifications
- Billing
- SSH and GPG keys**
- Security
- Blocked users

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

	runoob.com Fingerprint:  SSH Added on 31 Dec 2015 — Last used within the last day	Delete
---	---	------------------------

② Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

GPG keys

[New GPG key](#)

There are no GPG keys with access to your account.

```
$ ssh -T git@github.com
Hi tianqixin! You've successfully authenticated, but GitHub does not provide shell access.
```

仓库地址:



```
git commit -m '...'
```

git remote add 刚才取的名字 仓库地址

git push 刚才取的名字 分支