1. 集中式和分布式版本控制
2. git分布式，svn集中式
3. 查看版本

```
git --version
```

4. 配置作者信息

   全局设置

```
git congig --global user.name "your name"
git congig --global user.email "your email"
```

   当前设置

```
git congig user.name "your name"
git congig user.email "your email"
```

5. git初始化，建立新仓库

```
git init
```

6. 克隆远端项目到本地

```
git clone 项目网址
```

7. 添加文件

```
git add 文件名
```

8. 查看状态

```
git status
```

9. 提交到仓库

```
git commit -m "test1"  //提交并备注test1
```

10. 修改后再查看状态会有显示修改的文件
11. 添加所有文件

```
git addd .
```

12. 忽略添加文件方法，新建.gitignore文件，编辑.gitignore文件写上忽略文件名，示范格式

```
*.txt
!a.txt  //忽略所有.txt文件，除了a.txt
/vendor/  忽略vendor文件夹
```

13. git不跟踪空文件夹
14. 移除并删除本地文件

```
git rm 文件名
```

15. 只移除不删除本地文件

```
git rm --cached 文件名
```

16. 查看仓库里有哪些文件

```
ls或git ls-files
```

17. 仓库里的文件改名

```
git mv 当前文件名 修改的文件名
```

18. 查看提交日志

```
git log
```

19. 修改日志提交描述或者提交到上次版本并添加描述

```
git commit --amend
```

20. 添加到暂存区未提交到仓库，从暂存区撤出

```
git rm --cached 文件名
```

21. 提交到仓库，更改文件后添加到暂存区，从暂存区撤出

```
git reset HEAD 文件名
```

22. 恢复到文件上次状态

```
git checkout -- 文件名
```

23. 退回上一个版本

```
git reset --hard HEAD^
```

24. 退回上两个版本

```
git reset --hard HEAD^^
```

25. 退回上20个版本

```
git reset --hard HEAD~20
```

26. 使用git log查看版本提交ID，退回到那个版本，输入退回ID

```
git reset --hard 52602b0b04b57b33310fa256707518b126d2648c  //后面是查看到的ID
```

27. 回退到某一个文件之后，又需要返回到最近更新的某个版本，使用git log查看不到提交信息，可以使用git reflog命令

```
git reflog
git reset --har d2716ff6
```

28. 设置命令快捷键方法，编辑~/.gitconfig文件，添加内容

```
[alias]
    快捷键 = 命令
```

29. 查看分支

```
git branch
```

30. 创建分支

```
git branch 分支名
```

31. 切换分支

```
git checkout 分支名
```

32. 创建并切换到分支

```
git checkout -b 分支名
```

33. 合并到当前分支

```
git merge 分支名
```

34. 删除分支

```
git branch -d 分支名
```

35. 冲突问题：从主分支分出几个分支，文件几个个分支都改了，合并时，第一次合并不会有问题，后面分支再合并时就会冲突，并会修改冲突文件展示哪些有冲突，命令窗口会显示哪些文件有冲突，解决办法：找到冲突文件，修改好再重新添加提交

36. 显示已合并的分支

```
git branch --merged
```

37. 显示未合并的分支

```
git branch --no-merged
```

38. 创建分支都是从当前所在分支提交点创建的，也就是复制一份当前分支的文件
39. 未合并分支使用-d参数会提示未合并，删不掉，要强制删除

```
git branch -D 分支名
```

40. 添加临时存储区

```
git stash
```

41. 查看临时存储区

```
git stash list
```

42. 从临时存储恢复

```
git stash apply
```

43. 删除临时存储

```
git stash drop 临时存储区名
```

44. 恢复并删除临时存储

```
git stash pop
```

45. 文件已经提交过，目前修改好在暂存区未提交，要切换分支会提示未提交，可以添加临时存储区，之后就可以切换分支了，后面再回来时，可以从临时存储区直接恢复
46. 添加标签，如v1.0

```
git tag v1.0
```

47. 查看标签

```
git tag
```

48. 生成zip发布压缩包

```
git archieve master --prefix='文件夹名' --forma=zip > 文件名.zip   //将master分支的文件放在文件夹并打包成zip文件
```

49. 从一个分支生成另一个分支，生成的分支改变并提交，而原分支未改变提交，在原分支合并生成的分支，只是将指针移到修改后的分支上，不会产生合并记录，如果两个分支都改变提交了，在合并时，会显示合并记录或者产生冲突，要想不产生合并记录，生成的分支要rebase（replace base），再合并

```
git rebase master
```

50. 生成密钥

```
ssh-keygen -t rsa
```

生成密钥文件再用户目录的.ssh文件夹里面，将公钥复制添加到GitHub，就可以通过ssh，无需输入GitHub账号密码进入连接传输了

51. 从远端克隆的项目自动与远程仓库关联，推送到远程仓库

```
git push
```

52. 有时git与远端连接不上，可以删除~/.ssh/know_hosts文件里的内容
53. 未克隆远程仓库，与远程仓库关联

```
git remote add origin 远程仓库地址   //origin是远程仓库别名
```

54. 查看已关联的远程仓库

```
git remote -v
```

55. 推送数据到远程仓库的master分支

```
git remote -u origin master
```

56. 推送数据包括所有分支到远端

```
git push -all origin
```

57. 显示所有分支包括远程分支

```
git branch -a
```

58. 在本地创建远端没有的分支ask，推送到远端，提示fetal，运行

```
git push --set-upstream origin ask
```

59. 直接克隆远端到本地发现只克隆了主分支，要获得其他分支到本地

```
git pull orgin ask:ask    //克隆远端ask分支到本地的ask分支里，前面的ask分支是远端ask分支，后面是本地分支名，可以取不同名字
```

60. git push是将本地当前所在分支推送到远端
61. 远端master更新了，本地其他分支也更新了，要合并本地分支到远端主分支并推送到远端，在主分支执行git pull更新到最新版本，再切换到要合并的分支，rebase，再合并再推送
62. 删除远程分支

```
git push origin --delete ask
```