1：git工作流程：

1. 在工作区中修改文件。
2. 将你想要下次提交的更改选择性地暂存，这样只会将更改的部分添加到暂存区。
3. 提交更新，找到暂存区的文件，将快照永久性存储到 Git 目录

2：git config 带变量 三种 1:--system （需要管理员权限） 2：--global (全部可用)

3：--local (默认状态，针对当前使用的仓库) 等级高的配置会覆盖低等级配置，1——3为从低到高

3:git 配置：首先就是配置用户名和邮箱：

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

双引号内为自己配置的东西,再次强调，如果使用了 --global 选项，那么该命令只需要运行一次，因为之后无论你在该系统上做任何事情， Git 都会使用那些信息。 当你想针对特定项目使用不同的用户名称与邮件地址时，可以在那个项目目录下运行没有 --global 选项的命令来配置。

4:配置文本编辑器：git config –global core.editor 进行配置，但是具体参数需要自行查找，一般没用

5：git 中\*为通配符 git status 列出状态变化文件。文件修改或者未添加都可运行：git add 来添加，后可接文件路径。符号说明：星号（\*）匹配零个或多个任意字符；[abc] 匹配任何一个列在方括号中的字符 （这个例子要么匹配一个 a，要么匹配一个 b，要么匹配一个 c）； 问号（?）只匹配一个任意字符；如果在方括号中使用短划线分隔两个字符， 表示所有在这两个字符范围内的都可以匹配（比如 [0-9] 表示匹配所有 0 到 9 的数字）。 使用两个星号（）表示匹配任意中间目录，比如 a//z 可以匹配 a/z 、 a/b/z 或 a/b/c/z 等。

写入一个以.gitignore结尾的文件一下匹配符号可以忽略文件中一些文件，将之不上传。

# 忽略所有的 .a 文件

\*.a

# 但跟踪所有的 lib.a，即便你在前面忽略了 .a 文件

!lib.a

# 只忽略当前目录下的 TODO 文件，而不忽略 subdir/TODO

/TODO

# 忽略任何目录下名为 build 的文件夹

build/

# 忽略 doc/notes.txt，但不忽略 doc/server/arch.txt

doc/\*.txt

# 忽略 doc/ 目录及其所有子目录下的 .pdf 文件

doc/\*\*/\*.pdf

6：git diff 查看尚未暂存的文件更新了哪部分 git diff –staged或者git diff –cached 查看已经添加暂存但是未提交与上次提交的文件的差别

7:git commit –m “\*\*\*\*\*” 提交文件，同时带有注释

8：git commit –a 可以将所有已经追踪过的文件直接暂存后提交，就可以跳过git add ，但只会是包含已经追踪过的，新添加文件未git add 则不能上传

9:git rm 与git add 相对，将某一文件移除追踪状态，也就从暂存状态移除，同时git status 不在追踪，同时会在文件夹中删除相关文件。如果要删除之前修改过或已经放到暂存区的文件，则必须使用强制删除选项 -f（译注：即 force 的首字母）。 这是一种安全特性，用于防止误删尚未添加到快照的数据，这样的数据不能被 Git 恢复。若想在本地保留这一文件，则是需要git rm --cached 后接文件名称或者匹配规则

10: git mv file\_from file\_to 重命名文件夹名称。并且将原文件移除暂存区，新文件则添加。

11:git log 回顾提交历史 很多参数，自行查找[Git - 查看提交历史 (git-scm.com)](https://git-scm.com/book/zh/v2/Git-%E5%9F%BA%E7%A1%80-%E6%9F%A5%E7%9C%8B%E6%8F%90%E4%BA%A4%E5%8E%86%E5%8F%B2)

12：git commit –amend 这一次提交会取代上一次提交，也是提交上去了

13：git reset HEAD <file> 将已经在暂存区中的文件取消暂存，也就是从add中移除

14: git checkout -- <filename> 将此文件的修改删除至上次提交上传的原样。

15：git remote 查看远程仓库 git remote –v 显示git保存的简称和其对应的URL

16:：git remote add <shortname> <url> 添加一个新的远程 Git 仓库，同时指定一个方便使用的简写，这之后可以用pb代替你的URL

17：git fetch <remote> remote 为上个中简称，或者默认的origin 拉取你当前没有但仓库中存在的东西，相当于本地补全。

18：git push <remote><branch> 将现在的branch分支推送到remote服务器中

19：git remote show <remote> 查看特定服务器仓库的信息

20：git remote rename 重命名一个远程仓库的名字 git remote rename pb paul

21：git remote remove 删除本地与一个远程库的连接 git remote remove paul

22：git tag 列标签 同时可以在后面加上匹配规则，双引号内部即可,同时加-l

23：git tag –a <name> -m “information” name为标签名 information 为标签附带的信息，必需,此为附注标签 轻量标签则只要git tag <name> 这样不会附带额外信息，如作者姓名，邮箱等

24：git tag –a <name> <检验和> 给以前的某条加上标签 检验和为对应某条的检验和

25：git push <remote><tagname> 将标签上传，一般标签不上传，或者加—tags 上传全部未上传的标签

26: git tag -d <tagname> 删除某个标签 git push origin --delete <tagname> 删除远程库上的tag

27：git branch <name> 创建分支 加参数 –d 删除分支 –D 强制删除

28：git checkout <name> 切换到指定名字分支 带参数 –b 可以实现创建的同时跳转

29：git merge <name> 将分支合并进现在所在分支，所以在切换前需要先切分支。git merge 中有冲突文件，冲突解决需要用git add 暂存标记为已解决冲突

30：git branch 显示全部分支 加参数 –v 查看每个分支的最后一次提交 –-merged 显示已经合并至本分支的分支 --no-merged 显示所有包含未合并工作的分支

31：git checkout –track origin/other originname = git checkout –b <branch> <remote>/<branch> 将本地分支与远程分支跟踪，也就是相绑定，决定传至哪个分支下。

32：git push <remote> --delete <branch> 将branch上传同时删除branch分支

33：git rebase 变基 将现在分支对比另一个分支，之间的差异，将差异插入到另一分支的上面，就如同补丁效果。参数 --onto 可以将在一个分支上分出的另一分支2 变基到分支3上。将分支与分支2的差异变基到分支3. 。之后再合并就可以实现代码整合。也可git rebase <branch1><branch2> 将branch2变基到branch1