```
#配置使用 git 仓库的人员姓名
git config --global user.name "Your Name Comes Here"

#配置使用 git 仓库的人员 email
git config --global user.email you@yourdomain.example.com

#配置到缓存 默认 15 分钟
git config --global credential.helper cache

#修改缓存时间
git config --global credential.helper 'cache --timeout=3600'

git config --global color.ui true
git config --global alias.co checkout
git config --global alias.ci commit
git config --global alias.st status
git config --global alias.br branch
git config --global core.editor "mate -w"       # 设置 Editor 使用 textmate
git config -1 #列举所有配置
git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset
-%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"
cat .git/config         //配置
cat .gitconfig   //别名

#用户的 git 配置文件~/.gitconfig
git help <command>    # 显示 command 的 help
git show                    # 显示某次提交的内容
git show $id

git co    -- <file>     # 抛弃工作区修改
git co    .                  # 抛弃工作区修改

git add <file>          # 将工作文件修改提交到本地暂存区
git add .                  # 将所有修改过的工作文件提交暂存区

git rm <file>            # 从版本库中删除文件
git rm <file> --cached    # 从版本库中删除文件，但不删除文件

git reset <file>       # 从暂存区恢复到工作文件
git reset -- .          # 从暂存区恢复到工作文件
git reset --hard       # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改

git ci <file>
git ci .
```

```
git ci -a                  # 将 git add, git rm 和 git ci 等操作都合并在一起做
git ci -am "some comments"
git ci --amend             # 修改最后一次提交记录

git revert <$id>       # 恢复某次提交的状态，恢复动作本身也创建了一次提交对象
git revert HEAD        # 恢复最后一次提交的状态


git diff <file>        # 比较当前文件和暂存区文件差异
git diff
git diff <$id1> <$id2>     # 比较两次提交之间的差异
git diff <branch1>..<branch2> # 在两个分支之间比较
git diff --staged      # 比较暂存区和版本库差异
git diff --cached      # 比较暂存区和版本库差异
git diff --stat        # 仅仅比较统计信息

git log
git log <file>         # 查看该文件每次提交记录
git log -p <file>      # 查看每次详细修改内容的 diff
git log -p -2          # 查看最近两次详细修改内容的 diff
git log --stat         #查看提交统计信息

#初始化一个版本仓库
git init

#Clone 远程版本库
git clone git@xbc.me:wordpress.git

#添加远程版本库 origin，语法为 git add [shortname] [url]
git remote add origin git@xbc.me:wordpress.git

#查看远程仓库
git remote -v

#添加当前修改的文件到暂存区
git add .

#如果你自动追踪文件，包括你已经手动删除的，状态为 Deleted 的文件
git add -u

#提交你的修改
git commit  –m "你的注释"

#推送你的更新到远程服务器,语法为 git push [远程名] [本地分支]:[远程分支]
```

```
git push origin master

#查看文件状态
git status

#跟踪新文件
git add readme.txt

#从当前跟踪列表移除文件，并完全删除
git rm readme.txt

#仅在暂存区删除，保留文件在当前目录，不再跟踪
git rm －cached readme.txt

#重命名文件
git mv reademe.txt readme

#查看提交的历史记录
git log

#修改最后一次提交注释的，利用－amend 参数
git commit --amend

#忘记提交某些修改，下面的三条命令只会得到一个提交。
git commit －m &quot;add readme.txt&quot;
git add readme_forgotten
git commit －amend

#假设你已经使用 git add .，将修改过的文件 a、b 加到暂存区

#现在你只想提交 a 文件，不想提交 b 文件，应该这样
git reset HEAD b

#取消对文件的修改
git checkout －- readme.txt

git br -r                  # 查看远程分支
git br <new_branch> #  创建新的分支
git br -v                  # 查看各个分支最后提交信息
git br --merged         # 查看已经被合并到当前分支的分支
git br --no-merged    # 查看尚未被合并到当前分支的分支

git co <branch>         # 切换到某个分支
git co -b <new_branch> #  创建新的分支，并且切换过去
```

```
git co -b <new_branch> <branch>    # 基于 branch 创建新的 new_branch

git co $id                  # 把某次历史提交记录 checkout 出来，但无分支信息，切换到其他分支会自动删除
git co $id -b <new_branch>    # 把某次历史提交记录 checkout 出来，创建成一个分支

git br -d <branch>    # 删除某个分支
git br -D <branch>    # 强制删除某个分支 (未被合并的分支被删除的时候需要强制)


git merge <branch>                    # 将 branch 分支合并到当前分支
git merge origin/master --no-ff    # 不要 Fast-Foward 合并，这样可以生成 merge 提交

git rebase master <branch>           # 将 master rebase 到 branch，相当于：
git co <branch> && git rebase master && git co master && git merge <branch>

git diff > ../sync.patch              # 生成补丁
git apply ../sync.patch               # 打补丁
git apply --check ../sync.patch    #测试补丁能否成功

git stash                             # 暂存
git stash list                        # 列所有 stash
git stash apply                       # 恢复暂存的内容
git stash drop                        # 删除暂存区


git pull                              # 抓取远程仓库所有分支更新并合并到本地
git pull --no-ff                      # 抓取远程仓库所有分支更新并合并到本地，不要快进合并
git fetch origin                      # 抓取远程仓库更新
git merge origin/master               # 将远程主分支合并到本地当前分支
git co --track origin/branch         # 跟踪某个远程分支创建相应的本地分支
git co -b <local_branch> origin/<remote_branch>    # 基于远程分支创建本地分支，功能同上

git push                              # push 所有分支
git push origin master                # 将本地主分支推到远程主分支
git push -u origin master             # 将本地主分支推到远程(如无远程主分支则创建，用于初始化远程仓库)
git push origin <local_branch>     # 创建远程分支， origin 是远程仓库名
git push origin <local_branch>:<remote_branch>    # 创建远程分支
git push origin :<remote_branch>    #先删除本地分支(git br -d <branch>)，然后再 push 删除远程分支
```

#创建一个分支
git branch iss53

#切换工作目录到 iss53
git chekcout iss53

#将上面的命令合在一起，创建 iss53 分支并切换到 iss53
git chekcout –b iss53

#合并 iss53 分支，当前工作目录为 master
git merge iss53

#合并完成后，没有出现冲突，删除 iss53 分支
git branch –d iss53

#拉去远程仓库的数据，语法为 git fetch [remote-name]
git fetch

#fetch 会拉去最新的远程仓库数据，但不会自动到当前目录下，要自动合并
git pull

#查看远程仓库的信息

#建立本地的 dev 分支追踪远程仓库的 develop 分支
git checkout –b dev origin/develop

git remote -v # 查看远程服务器地址和仓库名称
git remote show origin # 查看远程服务器仓库状态
git remote add origin git@ github:robbin/robbin_site.git # 添加远程仓库地址
git remote set-url origin git@ github.com:robbin/robbin_site.git # 设置远程仓库地址(用于修改远程仓库地址)
git remote rm <repository> # 删除远程仓库

git clone --bare robbin_site robbin_site.git # 用带版本的项目创建纯版本仓库
scp -r my_project.git git@ git.csdn.net:~ # 将纯仓库上传到服务器上

mkdir robbin_site.git && cd robbin_site.git && git --bare init # 在服务器创建纯仓库
git remote add origin git@ github.com:robbin/robbin_site.git # 设置远程仓库地址
git push -u origin master # 客户端首次提交
git push -u origin develop # 首次将本地 develop 分支提交到远程 develop 分支，并且 track

git remote set-head origin master # 设置远程仓库的 HEAD 指向 master 分支

git clone --bare robbin_site robbin_site.git # 用带版本的项目创建纯版本仓库

```
scp -r my_project.git git@ git.csdn.net:~          # 将纯仓库上传到服务器上

mkdir robbin_site.git && cd robbin_site.git && git --bare init # 在服务器创建纯仓库
git remote add origin git@ github.com:robbin/robbin_site.git       # 设置远程仓库地址
git push -u origin master                                                    # 客户端首次提交
git push -u origin develop    # 首次将本地 develop 分支提交到远程 develop 分支，并且 track

git remote set-head origin master      # 设置远程仓库的 HEAD 指向 master 分支
```