git command line

base

```
vim some.html
# 提交
git commit -a -m "23"
# 推送
git push origin
```

create git branch

```
git checkout -b dev
or
git branch dev
git checkout dev
```

add file

```
git add readme.md
git commit -m "add readme file"
```

remove file

```
git rm filename
```

merge branch

```
git checkout master
git merge dev
```

view current branch

```
git branch
```

git init repo demo

```
mkdir demo
cd demo
git init
```

delete branch

```
git branch -d dev
```

create an gh-pages

```
git checkout --orphan gh-pages
```

push local repo to remote repo

```
git remote add origin https://github.com/username/demo.git
git push origin gh-pages
```

pull remote repo to local

```
git pull
```

打版本

```
git checkout master
git tag v1.0 [commitId]
git tag -d v1.0
git tag
git push origin v1.0 or --tags
git tag -d v0.9
git push origin :refs/tags/v0.9
```

git clone

```
# checkout release from remote repo
git clone <repourl>
git clone <版本库的网址> <本地目录名>

# examples
git clone http[s]://example.com/path/to/repo.git/
git clone ssh://example.com/path/to/repo.git/
git clone git://example.com/path/to/repo.git/
git clone /opt/git/project.git
git clone file:///opt/git/project.git
git clone ftp[s]://example.com/path/to/repo.git/
git clone rsync://example.com/path/to/repo.git/
```

git remote

```
# 为了便于管理，Git要求每个远程主机都必须指定一个主机名。
# git remote命令就用于管理主机名。
# 不带选项的时候，git remote命令列出所有远程主机。
git remote
# 使用-v选项，可以参看远程主机的网址。
git remote -v
# 可以查看该主机的详细信息
git remote show <主机名>
# 添加远程主机
git remote add <主机名> <网址>
# 删除远程主机
git remote rm <主机名>
# 改名
git remote rename <原主机名> <新主机名>
```

git fetch

```
# 新远程主机版本库里的更新取回到本地，默认取回所有分支的进程
git fetch <远程主机名>

# 取回指定分支的更新
git fetch <远程主机名> <分支名>

# 所取回的更新，在本地主机上要用"远程主机名/分支名"的形式读取
# 查看远程分支
git branch -r

# 查看所有分支
git branch -a

# 取回远程主机的更新以后，可以在它的基础上，使用git checkout命令创建一个新的分支
git checkout -b newBranch origin/master

# 也可以使用git merge命令或者git rebase命令，在本地分支上合并远程分支
git merge origin/master
or
git rebase origin/master
```

git pull

```
# 取回远程主机某个分支的更新，再与本地的指定分支合并
git pull <远程主机名> <远程分支名>:<本地分支名>

# 取回origin主机的next分支，与本地的master分支合并
git pull origin next:master

# 如果远程分支是与当前分支合并，则冒号后面的部分可以省略
git pull origin next
or
git fetch origin
git merge origin/next

# Git也允许手动建立追踪关系，git clone会自动建立追踪关系
# 下面命令指定master分支追踪origin/next分支
git branch --set-upstream master origin/next

# 如果当前分支与远程分支存在追踪关系，git pull可以省略远程远程分支
git pull origin

# 如果当前分支只有一个追踪分支，连远程主机名都可以省略
git pull

# 如果远程主机删除了某个分支，默认情况下，git pull 不会在拉取远程分支的时候，删除对应的本地分支。
# 这是为了防止，由于其他人操作了远程主机，导致git pull不知不觉删除了本地分支。
# 但是，你可以改变这个行为，加上参数 -p 就会在本地删除远程已经删除的分支。
git pull -p
or
git fetch --prune origin
git fetch -p

# 添加rebase模式
git pull --rebase <远程主机名> <远程分支名>:<本地分支名>
```

git push

```
# 用于将本地分支的更新推送到远程主机
# 注意，分支推送顺序的写法是<来源地>:<目的地>，
# 所以git pull是<远程分支>:<本地分支>，而git push是<本地分支>:<远程分支>
git push <远程主机名> <本地分支名>:<远程分支名>

# 如果省略本地分支名，则表示删除指定的远程分支，因为这等同于推送一个空的本地分支到远程分支。
# 下面命令表示删除origin主机的master分支
git push origin :master
or
git push origin --delete master

# 如果当前分支与远程分支之间存在追踪关系，则本地分支和远程分支都可以省略
git push origin

# 如果当前分支只有一个追踪分支，那么主机名都可以省略
git push

# 如果当前分支与多个主机存在追踪关系，则可以使用-u选项指定一个默认主机，
# 这样后面就可以不加任何参数使用git push。
git push -u origin master

# 不带任何参数的git push，默认只推送当前分支，这叫做simple方式。
# 如果要修改这个设置，可以采用git config命令
git config --global push.default matching
or
git config --global push.default simple

# 还有一种情况，就是不管是否存在对应的远程分支，
# 将本地的所有分支都推送到远程主机，这时需要使用--all选项
git push -all origin

# 如果远程主机的版本比本地版本更新，要先git pull再推送，也可以强制进行
git push --force origin
```

git log

```
git log
```

git reset

```
# 回退版本
git reset --hard commitId
```

git reflog

```
# 回到未来
git reflog
git reset --hard commitId
```

git status

```
# 查看文件状态
git status
```