
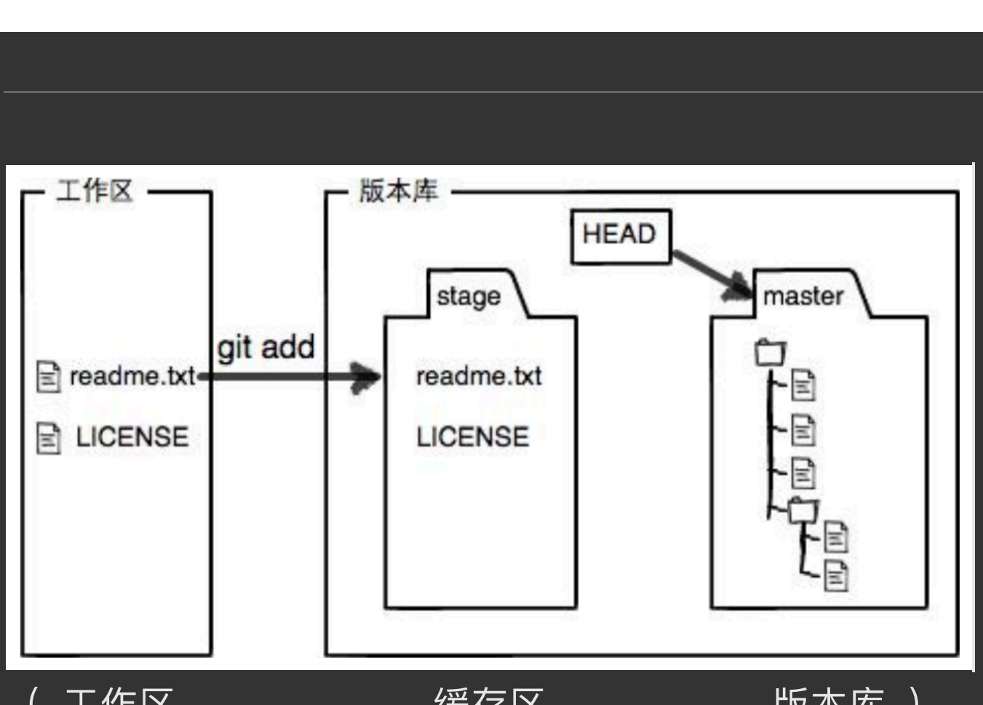

（ 工作区，　　　　缓存区，　　　　版本库 ）

☑ master (origin)　主分支master，默认远程主机origin
☑ dev　　　　　　分支dev

cat ： 查看文件
vim ： 编辑文件

git相比svn的优势：https://www.zhihu.com/question/19601997

1、本机版本管理：

令文件夹使用git进行管理，初始化：

```
git init
# 在文件夹下执行
# 使用（ls -ah）可以看到文件夹下的隐藏文件 .git
```

在git管理文件夹下添加文档到缓存区中（add后的文件才会进行commit操作）：

```
git add readme.txt
git add file2.txt file3.txt

# readme.txt需要提前建立
```

完成更改，commit并说明改动之处：

```
git commit -m "wrote a readme file”
```

查看（工作区）文件状态：

```
git status

cat readme.txt
# 查看文件
```

查看文件夹中被修改文件和版本库中的不同：

```
git diff

# 查看完按q退出
```

查看历史版本记录：

```
git log  # 详细显示（带时间）
git log --pretty=oneline  # 按行显示
git log --graph   # 可以看到分支合并图。
git log --graph --pretty=oneline --abbrev-commit  # 简洁时间线显示提交信息
```

回退至前版本：

```
git reset --hard HEAD^
# HEAD表示当前版本，HEAD^表示前一版本，HEAD^^表示前前个版本，HEAD~100表示前第100个版本

git reset HEAD readme.txt
# 可以把暂存区的修改撤销掉（unstage），重新放回工作区
```

查看历史命令：

```
git reflog
# 此命令可以显示每个版本的版本号，可用于回退之后恢复到新版本：
git reset --hard 1094a
# 1940a是新版本的前五位版本号
```

用版本库替换工作区 (add , commit 之前撤销全部更改)：

```
git checkout – readme.txt
# 同时也用于转向分支branch

# git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

# 一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；
# 一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。
```

```
场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。
场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD <file>
      就回到了场景1，第二步按场景1操作。
场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库。
```

从版本库中删除文件：

```
git rm test.txt
```

2、github远程仓库管理：

（需在git管理的文件夹路径中进行操作）
（本机已与github账号建立ssh秘钥认证链接）

在github建立项目后，与本机项目进行连接：

```
git remote add origin git@github.com:doctorf18/learngit.git

# origin：默认远程主机名称，也可以起其他名字
# doctorf18：github用户名
# learngit：github项目名

连接失败时：
git remote rm origin
移除origin远程主机
之后重新连接
```

把本地master分支的最新修改推送至GitHub：

```
git push origin master

(master) 代表推送的是主线master
如果要推送分支dev：    git push origin dev

git push -u origin master
第一次推送时可以加 -u 命令，指将origin远程主机当做默认主机
以后用 git push 就可以把当前分支推送到这个主机上的对应分支
```

把远程库拉取到本地进行同步：

```
git pull origin master
git pull origin dev

# 如果git pull提示no tracking information
# 则说明本地分支和远程分支的链接关系没有创建
# 用命令git branch --set-upstream-to <branch-name> origin/<branch-name>
```

从远程库中克隆项目到本地：

```
git clone git@github.com:doctorf18/gitskills.git

# 此时默认只会克隆主分支master到本地

git checkout -b dev origin/dev

# 创建远程origin的dev分支到本地
```

查看远程库信息：

```
git remote
git remote -v
```

3、git分支管理：

创建分支，并转向分支：

```
git branch dev  # 创建分支
git checkout dev  # 转向分支

用一条语句表示为：
git checkout -b dev
```

查看当前分支：

```
git branch
列出所有分支，当前分支前面会标一个*号
```

合并指定分支到当前分支：

```
git merge dev
# 合并dev到master
# fastforward方式合并

git merge --no-ff -m "merge with no-ff” dev
# 使用普通方式合并，合并后可以保留分支的历史记录
```

删除分支：

```
git branch -d dev
# 创建+切换分支： git checkout -b dev

-D : 使用大写D强制删除
```

在某个分支中保存现场，以便不用add和commit就切换到其他分支：
（正常情况下，在某一个分支中进行了改动，不add和commit，是无法切换到其他分支的）

```
保存现场：( 注意：使用stash前要将分支中的新建文件add到暂存区 )
        ( 确认stash前分支中的所有文件都不是untrack状态，如果是，就add一下 )
        ( 否则恢复现场后新建文件会丢失，会跑到master里 )
git stash

切换回该分支后，使用：
git stash list    # 查看stash保存现场的版本

恢复现场：
git stash apply   # 恢复现场，并保存现场版本
git stash pop    # 恢复现场，并删除该现场版本
git stash apply stash@{0}   # 恢复某个现场版本
```

本地未push的分叉提交历史整理成直线：

```
git rebase

# rebase的目的是使得我们在查看历史提交的变化时更容易
```

4、标签管理：

给当前分支的已提交版本打标签：

```
git tag v1.0

# 给指定版本打标签：
git tag v0.9 10bac7

# 标签总是和某个commit挂钩
# 如果这个commit既出现在master分支，又出现在dev分支
# 那么在这两个分支上都可以看到这个标签
```

查看所有标签（按字母排序）：

```
git tag
```

查看标签信息：

```
git show v1.0
```

删除标签：

```
git tag -d v1.0
# 删除没有推送到远程的本地标签

git tag -d v1.0
git push origin :refs/tags/v1.0
# 删除已经上传到远端的tag标签：先删本地再删远程
```

创建有文字说明的标签：

```
git tag -a v0.1 -m "version 0.1 released” 1094adb

# -a 指定版本号    -m 指定文字说明
```

推送标签到远程：

```
git push origin v1.0
```

配置别名：

```
git config --global alias.st status

# –global ： 针对用户起作用，不加即针对此文件夹起作用
# 配置完成之后即可用st代替status
```

忽视某些文件的提交：

（在文件夹下创建.gitignore文件，此文件需要add）示例：

```
# Windows:
Thumbs.db
ehthumbs.db
Desktop.ini

# Python:
*.py[cod]
*.so
*.egg
*.egg-info
dist
build

# My configurations:
db.ini
deploy_key_rsa
```

忽略的文件进行强制添加：

```
git add -f App.class
```

```
忽略文件的原则是：
1. 忽略操作系统自动生成的文件，比如缩略图等；
2. 忽略编译生成的中间文件、可执行文件等，也就是如果一个文件是通过另一个文件自动生成的
    那自动生成的文件就没必要放进版本库，比如Java编译产生的.class文件；
3. 忽略你自己的带有敏感信息的配置文件，比如存放口令的配置文件。
```

https://www.liaoxuefeng.com/wiki/896043488029600/900004590234208