

- Git 
 - 图片总结
 - 新建仓库
 - 修改文件内容
 - 查看区域内容
 - 查看文件状态
 - 查看修改文件内容
 - 提交到缓存区
 - 提交到仓库区
 - 查看提交记录
 - 回退版本
 - 查看操作记录
 - 查看差异
 - 删除文件
 - 忽略文件
 - 配置密钥
 - 关联本地与远程仓库
 - 切换分支
 - 合并分支
 - 回退分支

图片总结

<div><div>初始化</div><div>初始化设置用户名和邮箱</div><div><pre>git config --global user.name "Your Name" git config --global user.email "email@example.com" git config --global credential.helper store</pre></div><div>创建仓库</div><div>创建一个新的本地仓库（省略project-name则在当前目录创建）</div><div><pre>git init <project-name></pre></div><div>下载一个远程仓库</div><div><pre>git clone <url></pre></div><div>Git的四个区域</div><div><ul style="list-style-type: none">• 工作区 (Working Directory)：就是你在电脑里能看到的目录• 暂存区 (Stage/Index)：一般存放在.git/目录下的index文件，所以我们将暂存区有时也叫作索引(index)• 本地仓库 (Repository)：工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库• 远程仓库 (Remote)：托管在远程服务器上的仓库</div><div>Git的三种状态</div><div><ul style="list-style-type: none">• 已修改 (Modified)：修改了文件，但没保存到暂存区• 已暂存 (Staged)：把修改后的文件放进暂存区• 已提交 (Committed)：把暂存区的文件提交到本地仓库</div><div>基本概念</div><div><ul style="list-style-type: none">• main：默认主分支• origin：默认远程仓库• HEAD：指向当前分支的指针• HEAD~：上一个版本• HEAD~n：上n个版本</div><div>特殊文件</div><div><ul style="list-style-type: none">• .gitignore：Git仓库的元数据和对象数据库• .gitignore：忽略文件，不需要提交到仓库的文件• .gitattributes：指定文件的属性，比如换行符• .gitkeep：使空目录被提交到仓库• .gitmodules：记录子模块的信息• .gitconfig：记录仓库的配置信息</div></div>	<div><div>添加和提交</div><div>添加一个文件到仓库</div><div><pre>git add <file></pre></div><div>添加所有文件到仓库</div><div><pre>git add .</pre></div><div>提交所有暂存区的文件到仓库</div><div><pre>git commit -m "message"</pre></div><div>提交所有已修改的文件到仓库</div><div><pre>git commit -am "message"</pre></div><div>分支</div><div><ul style="list-style-type: none">• 查看所有本地分支，当前分支前面会有一个*，在查看远程分支，在查看所有分支</div><div><pre>git branch</pre></div><div>创建一个新的分支</div><div><pre>git branch <branch-name></pre></div><div>切换到指定分支，并更新工作区</div><div><pre>git checkout <branch-name></pre></div><div>创建一个新分支，并切换到该分支</div><div><pre>git checkout -b <branch-name></pre></div><div>删除一个已经合并的分支</div><div><pre>git branch -d <branch-name></pre></div><div>删除一个分支，不管是否合并</div><div><pre>git branch -D <branch-name></pre></div><div>给当前的提交打上标签，通常用于版本发布</div><div><pre>git tag <tag-name></pre></div></div>	<div><div>合并分支</div><div>合并分支a到分支b，--no-commit参数表示禁用Fast forward模式，合并后的历史有分支，能看出曾经做过合并，而这样参数表示使用Fast forward模式，合并后的历史会变成一条直线</div><div><pre>git merge --no-ff -m "message" <branch-name></pre></div><div></div><div><pre>git merge --ff -m "message" <branch-name></pre></div><div></div><div>合并分支</div><div><pre>git merge --squash <branch-name></pre></div><div>rebase不会产生新的提交，而是把当前分支的每一个提交都“复制”到目标分支上，然后再把当前分支指向目标分支，而merge会产生一个新的提交，这个提交有两个分支的所有修改</div><div>Rebase</div><div><p>Rebase操作可以把本地来push的分支提交历史整理成直线，看起来更直观。但是，如果多人协作时，不要对已经推送到远程的分支执行Rebase操作。</p></div><div><pre>git checkout <dev> git rebase <main></pre></div><div></div></div>	<div><div>撤销</div><div>移动一个文件到新位置</div><div><pre>git mv <file> <new-file></pre></div><div>从工作区和暂存区中删除一个文件，然后暂存删除操作</div><div><pre>git rm <file></pre></div><div>只从暂存区中删除一个文件，工作区中的文件没有变化</div><div><pre>git rm --cached <file></pre></div><div>恢复一个文件到之前的版本</div><div><pre>git checkout <file> <commit-id></pre></div><div>创建一个新的提交，用来撤销指定的提交，后者所有变化都将被前者抵消，并且应用到当前分支</div><div><pre>git revert <commit-id></pre></div><div>重置当前分支的HEAD为之前的某个提交，并且删除所有之后的提交，--hard参数表示重置工作区和暂存区，--soft参数表示重置暂存区，--mixed参数表示重置工作区</div><div><pre>git reset --mixed <commit-id></pre></div><div>撤销暂存区的文件，重新放回工作区 (git add的反向操作)</div><div><pre>git restore --staged <file></pre></div><div>查看</div><div><p>列出还未提交的新的或修改的文件</p></div><div><pre>git status</pre></div><div>查看提交历史，--oneline可省略</div><div><pre>git log --oneline</pre></div><div>查看未暂存的文件更新了哪些部分</div><div><pre>git diff</pre></div><div>查看两个提交之间的差异</div><div><pre>git diff <commit-id> <commit-id></pre></div></div>	<div><div>Stash</div><div>Stash操作可以把当前工作现场“储藏”起来，等以后恢复现场后继续工作。--no参数表示把所有未跟踪的文件也一并存储，--keep参数表示把所有未跟踪的文件和忽略的文件也一并存储，save参数表示存储的信息，可以不同。</div><div><pre>git stash save "message"</pre></div><div>查看所有stash</div><div><pre>git stash list</pre></div><div>恢复最近一次stash</div><div><pre>git stash pop</pre></div><div>恢复指定的stash，stash@{2}表示第三个stash，stash@{0}表示最近的stash</div><div><pre>git stash pop stash@{2}</pre></div><div>重新接受最近一次stash</div><div><pre>git stash apply</pre></div><div>push和apply的区别是，push会把stash内容删除，而apply不会，可以用git stash drop来删除stash。</div><div><pre>git stash drop stash@{2}</pre></div><div>删除所有stash</div><div><pre>git stash clear</pre></div><div>远程仓库</div><div>添加远程仓库</div><div><pre>git remote add <remote-name> <remote-url></pre></div><div>查看远程仓库</div><div><pre>git remote -v</pre></div><div>删除远程仓库</div><div><pre>git remote rm <remote-name></pre></div></div>	<div><div>重命名远程仓库</div><div><pre>git remote rename <old-name> <new-name></pre></div><div>从远程仓库拉取代码</div><div><pre>git pull <remote-name> <branch-name></pre></div><div>fetch默认以远程仓库 (origin) 当前分支的代码，然后合并到本地分支</div><div><pre>git pull</pre></div><div>将本地没动的代码用Rebase到远程仓库最新代码上 (为了有一个干净的，线性的提交历史)</div><div><pre>git pull --rebase</pre></div><div>推送代码到远程仓库 (然后再发起pull request)</div><div><pre>git push <remote-name> <branch-name></pre></div><div>获取所有远程分支</div><div><pre>git fetch <remote-name></pre></div><div>查看远程分支</div><div><pre>git branch -r</pre></div><div>fetch某一个特定的远程分支</div><div><pre>git fetch <remote-name> <branch-name></pre></div><div>Git Flow</div><div><p>GitFlow是一种流程模型，用于在Git上管理软件开发项目。</p><ul style="list-style-type: none">• 主分支 (master)：代表了项目的稳定版本，每个提交到主分支的代码都应该经过测试和审核的。• 开发分支 (develop)：用于日常开发，所有功能分支、发布分支和修补分支都应该从develop分支派生。• 功能分支 (feature)：用于开发单独的功能或特性，每个功能分支都应该从develop分支派生，并在开发完成后合并回develop分支。• 发布分支 (release)：用于准备项目发布，发布分支应该从develop分支派生，并在准备好发布版本后合并回master和develop分支。• 热修复分支 (hotfix)：用于修复主分支上的问题，热修复分支应该从master分支派生，并在修复完成后合并回master和develop分支。</div></div>
---	--	--	---	--	--

新建仓库

mkdir 文件名 --->新建本地工作区

cd 文件名 --->更换bit命令操作位置，即打开该文件夹

git init 文件名 --->初始化该仓库

修改文件内容

echo "内容" > 文件名 --->将内容写进文件中，有空格要加引号

echo "内容" >> 文件名 --->将内容追加进文件中，有空格要加引号

查看区域内容

ls --->查看工作区所有文件

ls -a --->查看所有工作区文件，包括隐含文件

git ls -files --->查看缓存区文件

ls -ltr 列出当前目录下所有文件，并以修改时间从新到旧显示

查看文件状态

git status ---> 直接查看工作区文件状态

查看修改文件内容

cat 文件名 ---> 直接查看

vi 文件名 ---> 依据vim语法修改(即下述提交时语法)

提交到缓存区

git add 文件名---> 提交该文件

git add *.文件后缀名---> 提交所有该后缀名文件

git add .---> 提交所有文件

提交到仓库区

git commit -m "提交信息"---> 提交缓存区所有文件到仓库区

git commit ---> 不写-m会进入交互式页面，方向键移动光标，"i"键进入编辑模式，"esc"退出，再输入":wq"退出界面

查看提交记录

git log ---> 直接查看记录

git log --oneline ---> 查看简洁的提交记录

回退版本

git reset --sort 返回版本号 ---> 保留工作区文件，保留暂存区文件

git reset --hard 返回版本号 --->不保留工作区文件，不保留暂存区文件

git reset --mixed 返回版本号 --->保留工作区文件，不保留暂存区文件

查看操作记录

git reflog --->包括查看删除回退操作

查看差异

git diff --->默认比较工作区域暂存区的文件内容差异

git diff HEAD --->比较工作区与版本库的文件内容差异

git diff --cached --->比较暂存区与版本库的文件内容差异

git diff ID1 ID2 --->比较两次ID对应提交之间的文件内容差异

git diff HEAD~ HEAD --->比较最新一次提交和上一次提交的文件内容差异

HEAD表示最新提交，HEAD~表示上一次提交，HEAD ~i表示前i次提交

git diff HEAD~ HEAD 文件名 --->只查看该文件的内容差异

git diff bran1 bran2 --->查看两个分支之间的差异

删除文件

rm 文件名 --->删除本地工作区的该文件，记得随后要更新暂存区，版本库

git rm 文件名 --->删除本地工作区和暂存区的该文件，记得更新版本库

git rm --cached 文件名 --->删除版本库的该文件，记得更新版本库

忽略文件

.gitignore文件

用echo命令将对应文件放入.gitignore文件中可实现忽略 也可以用vim命令直接修改.gitignore文件中的内容，添加要忽略的文件

配置密钥

```
ssh-keygen -t rsa -C "邮箱"
```

注意多次配置密钥需要重新命名，并添加config文件

关联本地与远程仓库

```
git remote add 远程仓库别名 GitHub仓库SSH地址
```

```
git remote -v ---> 查看当前文件夹对应的远程仓库的别名与地址
```

```
git push -u origin master:master ---> 把本地仓库的master分支添加到origin远程仓库的master分支中
```

```
git pull origin master:master ---> 把origin中的master文件推送到本地master中
```

切换分支

```
git switch 分支名称
```

合并分支

```
git merge 分支名称
```

```
git rebase
```

在a分支中，rebase b分支，将b分支在公共祖先处插入，合并为一个流程

回退分支

```
git checkout -b 分支名 删除前的编号
```