---Git学习笔记---

Git是什么?

Git 是一个免费的、开源的分布式版本控制系统，可以快速高效地处理从小型到大型的各种项目。

特点：

- 易于学习
- 体积小
- 性能极快

版本控制

版本控制是一种记录文件内容变化，以便将来查阅特定版本修订情况的系统。

版本控制其实最重要的是可以记录文件修改历史记录，从而让用户能够查看历史版本，方便版本切换

Git 常用命令

设置用户签名

命令：

```
git config --global user.name 用户名
git config --global user.email 邮箱
```

命令分别设置用户名和邮箱

只需要设置一次即可

签名的作用是区分不同操作者身份。用户的签名信息在每一个版本的提交信息中能够看 到，以此确认本次提交是谁做的。Git 首次安装必须设置一下用户签名，否则无法提交代码。

这里设置用户签名和将来登录 GitHub（或其他代码托管中心）的账号没有任何关系。

初始化本地库

命令：

```
git init
```

使用：

```
PS C:\Users\mao\Desktop\test> ls
PS C:\Users\mao\Desktop\test> git init
Initialized empty Git repository in C:/Users/mao/Desktop/test/.git/
PS C:\Users\mao\Desktop\test> ls
PS C:\Users\mao\Desktop\test> cd .git
PS C:\Users\mao\Desktop\test\.git> ls


    目录: C:\Users\mao\Desktop\test\.git


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/29     21:09                hooks
d-----         2022/6/29     21:09                info
d-----         2022/6/29     21:09                objects
d-----         2022/6/29     21:09                refs
-a----         2022/6/29     21:09            112 config
-a----         2022/6/29     21:09             73 description
-a----         2022/6/29     21:09             23 HEAD


PS C:\Users\mao\Desktop\test\.git> cd hooks
PS C:\Users\mao\Desktop\test\.git\hooks> ls


    目录: C:\Users\mao\Desktop\test\.git\hooks


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2022/6/29     21:09            478 applypatch-msg.sample
-a----         2022/6/29     21:09            896 commit-msg.sample
-a----         2022/6/29     21:09           4655 fsmonitor-watchman.sample
-a----         2022/6/29     21:09            189 post-update.sample
-a----         2022/6/29     21:09            424 pre-applypatch.sample
-a----         2022/6/29     21:09           1643 pre-commit.sample
-a----         2022/6/29     21:09            416 pre-merge-commit.sample
-a----         2022/6/29     21:09           1374 pre-push.sample
-a----         2022/6/29     21:09           4898 pre-rebase.sample
-a----         2022/6/29     21:09            544 pre-receive.sample
-a----         2022/6/29     21:09           1492 prepare-commit-msg.sample
-a----         2022/6/29     21:09           2783 push-to-checkout.sample
-a----         2022/6/29     21:09           3650 update.sample
```

```
PS C:\Users\mao\Desktop\test\.git\hooks>
```

查看本地库状态

命令：

```
git status
```

使用：

```
PS C:\Users\mao\Desktop\test> git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
PS C:\Users\mao\Desktop\test>
```

新增一个文件：hello.txt

内容为123456

```
PS C:\Users\mao\Desktop\test> ls


    目录: C:\Users\mao\Desktop\test


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2022/6/29     21:13              6 hello.txt


PS C:\Users\mao\Desktop\test>
```

再次查看：

```
PS C:\Users\mao\Desktop\test> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.txt

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\mao\Desktop\test>
```

添加至暂存区

将工作区的文件添加到暂存区

命令：

```
git add 文件名
```

使用：

```
PS C:\Users\mao\Desktop\test> git add hello.txt
PS C:\Users\mao\Desktop\test>
```

查看状态：

```
PS C:\Users\mao\Desktop\test> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hello.txt

PS C:\Users\mao\Desktop\test>
```

提交到本地库

将暂存区的文件提交到本地库

命令：

```
git commit -m "日志信息" 文件名
```

使用：

```
PS C:\Users\mao\Desktop\test> git commit -m "第一次提交" hello.txt
[master (root-commit) 50e12c6] 第一次提交
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt
PS C:\Users\mao\Desktop\test>
```

```
PS C:\Users\mao\Desktop\test> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\mao\Desktop\test>
```

```
PS C:\Users\mao\Desktop\test> git log
commit 50e12c68902ad1e302203dfbaf1d27b4a7382a6d (HEAD -> master)
Author: mao <1296193245@qq.com>
Date:   Wed Jun 29 21:18:00 2022 +0800

    第一次提交
PS C:\Users\mao\Desktop\test>
```

历史版本

查看历史版本：

查看版本信息：

```
git reflog
```

查看版本详细信息：

```
git log
```

使用：

```
PS C:\Users\mao\Desktop\test> git reflog
5Oe12c6 (HEAD -> master) HEAD@{0}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test>
```

```
PS C:\Users\mao\Desktop\test> git log
commit 5Oe12c689O2ad1e3O22O3dfbaf1d27b4a7382a6d (HEAD -> master)
Author: mao <1296193245@qq.com>
Date:   Wed Jun 29 21:18:OO 2022 +0800

    第一次提交
PS C:\Users\mao\Desktop\test>
```

版本穿梭

命令:

```
git reset --hard 版本号
```

使用:

添加两个提交:

```
PS C:\Users\mao\Desktop\test> git reflog
5Oe12c6 (HEAD -> master) HEAD@{0}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
PS C:\Users\mao\Desktop\test> git commit -m "第二次提交" hello.txt
[master d5c5757] 第二次提交
 1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Users\mao\Desktop\test> git reflog
d5c5757 (HEAD -> master) HEAD@{0}: commit: 第二次提交
5Oe12c6 HEAD@{1}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
PS C:\Users\mao\Desktop\test> git commit -m "第三次提交" hello.txt
[master 4139c77] 第三次提交
 1 file changed, 1 insertion(+)
PS C:\Users\mao\Desktop\test> git reflog
4139c77 (HEAD -> master) HEAD@{0}: commit: 第三次提交
d5c5757 HEAD@{1}: commit: 第二次提交
5Oe12c6 HEAD@{2}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test> git status
```

```
On branch master
nothing to commit, working tree clean
PS C:\Users\mao\Desktop\test>
```

现在是第三个版本：

```
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
PS C:\Users\mao\Desktop\test>
```

```
PS C:\Users\mao\Desktop\test> git reflog
4139c77 (HEAD -> master) HEAD@{0}: commit: 第三次提交
d5c5757 HEAD@{1}: commit: 第二次提交
50e12c6 HEAD@{2}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test>
```

切换到第二个版本：

命令：

```
git reset --hard d5c5757
```

```
PS C:\Users\mao\Desktop\test> git reset --hard d5c5757
HEAD is now at d5c5757 第二次提交
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
PS C:\Users\mao\Desktop\test>
```

切换到第一个版本：

```
PS C:\Users\mao\Desktop\test> git reset --hard 50e12c6
HEAD is now at 50e12c6 第一次提交
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
PS C:\Users\mao\Desktop\test>
```

切换回第三个版本：

```
PS C:\Users\mao\Desktop\test> git reflog
50e12c6 (HEAD -> master) HEAD@{0}: reset: moving to master
50e12c6 (HEAD -> master) HEAD@{1}: reset: moving to 50e12c6
d5c5757 HEAD@{2}: reset: moving to d5c5757
4139c77 HEAD@{3}: commit: 第三次提交
d5c5757 HEAD@{4}: commit: 第二次提交
```

```
5Oe12c6 (HEAD -> master) HEAD@{5}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test> git reset --hard 4139c77
HEAD is now at 4139c77 第三次提交
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
PS C:\Users\mao\Desktop\test> git reflog
4139c77 (HEAD -> master) HEAD@{0}: reset: moving to 4139c77
5Oe12c6 HEAD@{1}: reset: moving to master
5Oe12c6 HEAD@{2}: reset: moving to 5Oe12c6
d5c5757 HEAD@{3}: reset: moving to d5c5757
4139c77 (HEAD -> master) HEAD@{4}: commit: 第三次提交
d5c5757 HEAD@{5}: commit: 第二次提交
5Oe12c6 HEAD@{6}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test>
```

Git 分支操作

在版本控制过程中，同时推进多个任务，为每个任务，我们就可以创建每个任务的单独 分支。使用分支意味着程序员可以把自己的工作从开发主线上分离开来，开发自己分支的时 候，不会影响主线分支的运行。对于初学者而言，分支可以简单理解为副本，一个分支就是 一个单独的副本。（分支底层其实也是指针的引用）

好处：

- 同时并行推进多个功能开发，提高开发效率
- 各个分支在开发过程中，如果某一个分支开发失败，不会对其他分支有任何影响。失败 的分支删除重新开始即可

Git分支命令

查看分支

命令：

```
git branch -v
```

使用：

```
PS C:\Users\mao\Desktop\test> git branch -v
* master 4139c77 第三次提交
PS C:\Users\mao\Desktop\test>
```

创建分支

命令：

```
git branch 分支名
```

使用：

```
PS C:\Users\mao\Desktop\test> git branch branch2
PS C:\Users\mao\Desktop\test> git branch -v
  branch2 4139c77 第三次提交
* master  4139c77 第三次提交
PS C:\Users\mao\Desktop\test> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\mao\Desktop\test> git reflog
4139c77 (HEAD -> master, branch2) HEAD@{0}: reset: moving to 4139c77
50e12c6 HEAD@{1}: reset: moving to master
50e12c6 HEAD@{2}: reset: moving to 50e12c6
d5c5757 HEAD@{3}: reset: moving to d5c5757
4139c77 (HEAD -> master, branch2) HEAD@{4}: commit: 第三次提交
d5c5757 HEAD@{5}: commit: 第二次提交
50e12c6 HEAD@{6}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test>
```

修改分支

```
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
123
PS C:\Users\mao\Desktop\test> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.txt
```

```
no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\mao\Desktop\test> git commit -m "第四次提交" hello.txt
[master ddc7a8a] 第四次提交
 1 file changed, 1 insertion(+)
PS C:\Users\mao\Desktop\test> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\mao\Desktop\test> git reflog
ddc7a8a (HEAD -> master) HEAD@{0}: commit: 第四次提交
4139c77 (branch2) HEAD@{1}: reset: moving to 4139c77
50e12c6 HEAD@{2}: reset: moving to master
50e12c6 HEAD@{3}: reset: moving to 50e12c6
d5c5757 HEAD@{4}: reset: moving to d5c5757
4139c77 (branch2) HEAD@{5}: commit: 第三次提交
d5c5757 HEAD@{6}: commit: 第二次提交
50e12c6 HEAD@{7}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test>
```

切换分支

命令：

```
git checkout 分支名
```

使用：

```
PS C:\Users\mao\Desktop\test> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\mao\Desktop\test>
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
123
PS C:\Users\mao\Desktop\test> git checkout branch2
Switched to branch 'branch2'
PS C:\Users\mao\Desktop\test> git status
On branch branch2
nothing to commit, working tree clean
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
PS C:\Users\mao\Desktop\test>
```

合并分支

命令：

```
git merge 分支名
```

更改hello.txt的内容

```
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
126
PS C:\Users\mao\Desktop\test>
```

切换到主分支

```
PS C:\Users\mao\Desktop\test> git status
On branch branch2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\mao\Desktop\test> git commit -m "branch2" hello.txt
[branch2 cd2bfa3] branch2
 1 file changed, 1 insertion(+)
PS C:\Users\mao\Desktop\test> git checkout master
Switched to branch 'master'
PS C:\Users\mao\Desktop\test>
```

合并分支：

```
PS C:\Users\mao\Desktop\test> git merge branch2
Auto-merging hello.txt
CONFLICT (content): Merge conflict in hello.txt
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\mao\Desktop\test>
```

产生冲突

冲突产生的表现：后面状态为 MERGING，或者出现CONFLICT

冲突产生的原因：

合并分支时，两个分支在同一个文件的同一个位置有两套完全不同的修改。Git 无法替 我们决定使用哪一个。必须人为决定新代码内容

查看状态（检测到有文件有两处修改）

```
PS C:\Users\mao\Desktop\test> git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\mao\Desktop\test>
```

解决冲突

编辑有冲突的文件，删除特殊符号，决定要使用的内容

特殊符号：<<<<<<< HEAD 当前分支的代码 ======= 合并过来的代码 >>>>>>> branch2

```
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
<<<<<<< HEAD
123
=======
126
>>>>>>> branch2
PS C:\Users\mao\Desktop\test>
```

处理后：

```
PS C:\Users\mao\Desktop\test> cat hello.txt
123456
12345
1234
123
126
PS C:\Users\mao\Desktop\test>
```

添加到暂存区：

```
PS C:\Users\mao\Desktop\test> git add hello.txt
PS C:\Users\mao\Desktop\test>
```

提交：

注意：此时使用 git commit 命令时不能带文件名

```
PS C:\Users\mao\Desktop\test> git commit -m "第四次提交"
[master fd481a8] 第四次提交
PS C:\Users\mao\Desktop\test>
```

状态：

```
PS C:\Users\mao\Desktop\test> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\mao\Desktop\test> git reflog
fd481a8 (HEAD -> master) HEAD@{0}: commit (merge): 第四次提交
ddc7a8a HEAD@{1}: checkout: moving from branch2 to master
cd2bfa3 (branch2) HEAD@{2}: commit: branch2
4139c77 HEAD@{3}: checkout: moving from master to branch2
ddc7a8a HEAD@{4}: commit: 第四次提交
4139c77 HEAD@{5}: reset: moving to 4139c77
50e12c6 HEAD@{6}: reset: moving to master
50e12c6 HEAD@{7}: reset: moving to 50e12c6
d5c5757 HEAD@{8}: reset: moving to d5c5757
4139c77 HEAD@{9}: commit: 第三次提交
d5c5757 HEAD@{10}: commit: 第二次提交
50e12c6 HEAD@{11}: commit (initial): 第一次提交
PS C:\Users\mao\Desktop\test>
```

Git 团队协作

查看远程仓库别名

命令：

```
git remote -v
```

使用：

```
PS C:\Users\mao\Desktop\test> git remote
PS C:\Users\mao\Desktop\test>
```

创建远程仓库别名

命令：

```
git remote add 别名 远程地址
```

使用：

https://github.com/maomao124/Redis_client

```
PS C:\Users\mao\Desktop\test> git remote add redis_cli
https://github.com/maomao124/Redis_client
PS C:\Users\mao\Desktop\test> git remote
redis_cli
PS C:\Users\mao\Desktop\test>
```

克隆远程仓库到本地

命令：

```
git clone 远程地址
```

```
PS C:\Users\mao\Desktop\redis-cli> ls
PS C:\Users\mao\Desktop\redis-cli> git clone
https://github.com/maomao124/Redis_client
Cloning into 'Redis_client'...
remote: Enumerating objects: 255, done.
remote: Counting objects: 100% (255/255), done.
remote: Compressing objects: 100% (129/129), done.
Receiving objects:  76% (194/255), 52.00 KiB | 31.00 KiB/sreused 0 eceiving
objects:  74% (189/255), 52.00 KiB | 31.00 KiB/s
Receiving objects: 100% (255/255), 72.13 KiB | 42.00 KiB/s, done.
Resolving deltas: 100% (49/49), done.
PS C:\Users\mao\Desktop\redis-cli> ls


    目录: C:\Users\mao\Desktop\redis-cli
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/30     22:44                Redis_client


PS C:\Users\mao\Desktop\redis-cli>
```

```
PS C:\Users\mao\Desktop\redis-cli> cd Redis_client
PS C:\Users\mao\Desktop\redis-cli\Redis_client> ls


    目录: C:\Users\mao\Desktop\redis-cli\Redis_client


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/30     22:44                .idea
d-----         2022/6/30     22:44                src
d-----         2022/6/30     22:44                target
-a----         2022/6/30     22:44          10812 pom.xml
-a----         2022/6/30     22:44           1842 README.md


PS C:\Users\mao\Desktop\redis-cli\Redis_client>
```

推送本地分支到远程仓库

命令:

```
git push 别名 分支
```

或者:

```
git push 地址 分支
```

使用:

```
PS C:\Users\mao\Desktop\redis-cli\Redis_client> git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
PS C:\Users\mao\Desktop\redis-cli\Redis_client> git reflog
0c885fd (HEAD -> master, origin/master, origin/HEAD) HEAD@{0}: clone: from
https://github.com/maomao124/Redis_client
PS C:\Users\mao\Desktop\redis-cli\Redis_client> git push
https://github.com/maomao124/Redis_client master
Everything up-to-date
PS C:\Users\mao\Desktop\redis-cli\Redis_client>
```

拉取最新分支

将远程仓库对于分支最新内容拉下来后与 当前本地分支直接合并

命令：

```
git pull 远程库地址别名 远程分支名
```

使用：

```
PS C:\Users\mao\Desktop\redis-cli\Redis_client> git pull
https://github.com/maomao124/Redis_client master
From https://github.com/maomao124/Redis_client
 * branch            master     -> FETCH_HEAD
Already up to date.
PS C:\Users\mao\Desktop\redis-cli\Redis_client> git reflog
0c885fd (HEAD -> master, origin/master, origin/HEAD) HEAD@{0}: clone: from
https://github.com/maomao124/Redis_client
PS C:\Users\mao\Desktop\redis-cli\Redis_client>
```

跨团队协作

- fork：复制别人的远程库
- Pull request：先fork别人的项目，然后本地修改完成提交到自己的个人 fork 仓库，最后提交 PR 等待别人合入你的代码
- merge：合并仓库

IDEA 集成 Git

配置 Git 忽略文件

创建忽略规则文件 xxxx.ignore（前缀名随便起，建议是 git.ignore）

这个文件的存放位置原则上在哪里都可以，为了便于让~/.gitconfig 文件引用，建议也放在用户目录下

git.ignore 文件模版内容：

```
# Compiled class file
*.class
# Log file
*.log
# BlueJ files
*.ctxt
# Mobile Tools for Java (J2ME)
.mtj.tmp/
# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar
# virtual machine crash logs, see
http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*
.classpath
.project
.settings
target
.idea
*.iml
```

在.gitconfig 文件中引用忽略配置文件

```
[user]
name =
email =
[core]
excludesfile = C:/Users/asus/git.ignore
```

注意：excludesfile要使用“正斜线（/）”，不要使用“反斜线（\）”

注意事项

- push 是将本地库代码推送到远程库，如果本地库代码跟远程库代码版本不一致， push 的操作是会被拒绝的。也就是说，要想 push 成功，一定要保证本地库的版本要比远程 库的版本高！因此一个成熟的程序员在动手改本地代码之前，一定会先检查下远程库跟本地 代码的区别！如果本地的代码版本已经落后，切记要先 pull 拉取一下远程库的代码，将本地 代码更新到最新以后，然后再修改，提交，推送
- pull 是拉取远端仓库代码到本地，如果远程库代码和本地库代码不一致，会自动 合并，如果自动合并失败，还会涉及到手动解决冲突的问题

GitLab

GitLab 是由 GitLabInc.开发，使用 MIT 许可证的基于网络的 Git 仓库管理工具，且具有 wiki 和 issue 跟踪功能。使用 Git 作为代码管理工具，并在此基础上搭建起来的 web 服务。 GitLab 由乌克兰程序员 DmitriyZaporozhets 和 ValerySizov 开发，它使用 Ruby 语言写 成。后来，一些部分用 Go 语言重写。截止 2018 年 5 月，该公司约有 290 名团队成员，以 及 2000 多名开源贡献者。GitLab 被 IBM，Sony，JülichResearchCenter，NASA，Alibaba， Invincea，O'ReillyMedia，Leibniz-Rechenzentrum(LRZ)，CERN，SpaceX 等组织使用。

Docker安装

在设置其他所有内容之前，请配置一个新的环境变量 `$GITLAB_HOME`，指向配置、日志和数据文件所在的目录。 确保该目录存在并且已授予适当的权限。

对于 Linux 用户，将路径设置为 `/srv/gitlab`

```
export GITLAB_HOME=/srv/gitlab
```

GitLab 容器使用主机装载的卷来存储持久数据

本地位置	容器位置	使用
`$GITLAB_HOME/data`	`/var/opt/gitlab`	用于存储应用程序数据。
`$GITLAB_HOME/logs`	`/var/log/gitlab`	用于存储日志。
`$GITLAB_HOME/config`	`/etc/gitlab`	用于存储极狐GitLab 配置文件。

创建一个 `docker-compose.yml` 文件：

```
version: '3.6'
services:
  web:
    image: 'registry.gitlab.cn/omnibus/gitlab-jh:latest'
    restart: always
    hostname: 'gitlab.example.com'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'https://gitlab.example.com'
        # Add any other gitlab.rb configuration here, each on its own line
    ports:
      - '80:80'
      - '443:443'
      - '22:22'
    volumes:
      - '$GITLAB_HOME/config:/etc/gitlab'
      - '$GITLAB_HOME/logs:/var/log/gitlab'
      - '$GITLAB_HOME/data:/var/opt/gitlab'
    shm_size: '256m'
```

启动:

```
docker-compose up -d
```

---

end

---

by mao

2022 07 01
