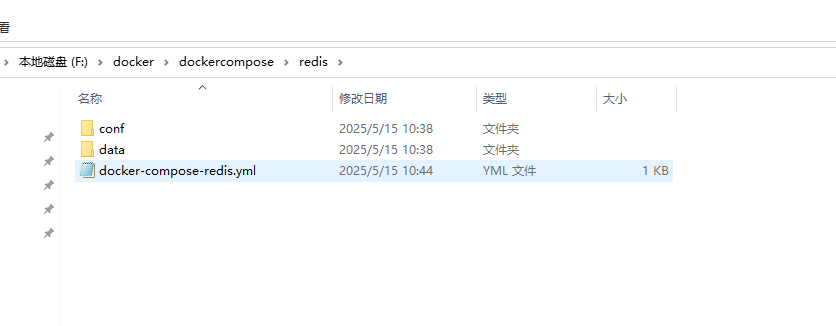
# 写在前面

本文档的的步骤执行之前需要你的Linux已经安装了docker和docker-compose

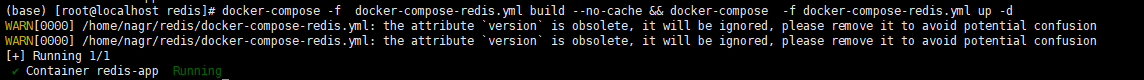
# Redis





只需要将上述文件复制到Linux系统中，修改yml对应的路径，因为我在配置文件中写的是绝对路径。在Linux系统中执行

docker-compose -f docker-compose-redis.yml build --no-cache && docker-compose -f docker-compose-redis.yml up -d

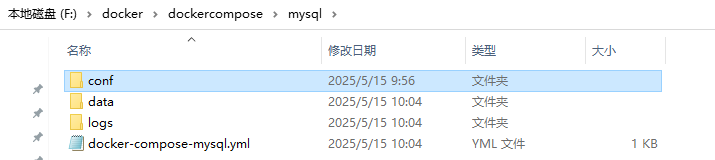


补充说明：redis 的配置文件下载地址

https://download.redis.io/redis-stable/

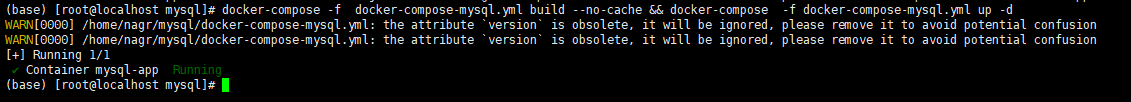
# Mysql





只需要将上述文件复制到Linux系统中，修改yml对应的路径，因为我在配置文件中写的是绝对路径。在Linux系统中执行

docker-compose -f docker-compose-mysql.yml build --no-cache && docker-compose -f docker-compose-mysql.yml up –d



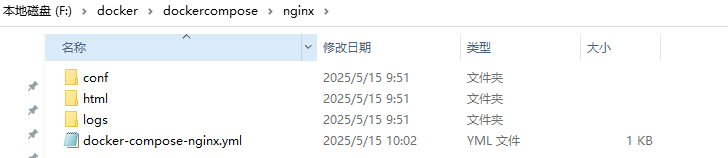
如果后期你需要重新构建，建议清空数据目录（包括隐藏文件）

sudo rm -rf /home/nagr/mysql8/data/\*

# 确保 data 是空的，权限还是 mysql 用户（如果用 root 启动 docker 可以忽略）

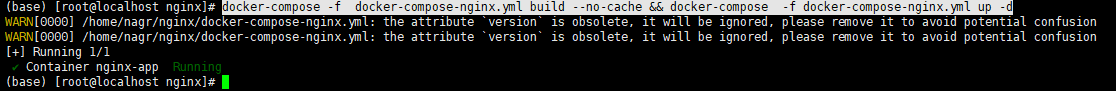
# Nginx





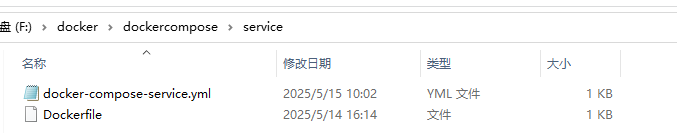
只需要将上述文件复制到Linux系统中，修改yml对应的路径，因为我在配置文件中写的是绝对路径。在Linux系统中执行

docker-compose -f docker-compose-nginx.yml build --no-cache && docker-compose -f docker-compose-nginx.yml up –d

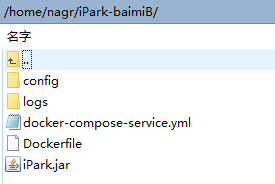


# Service



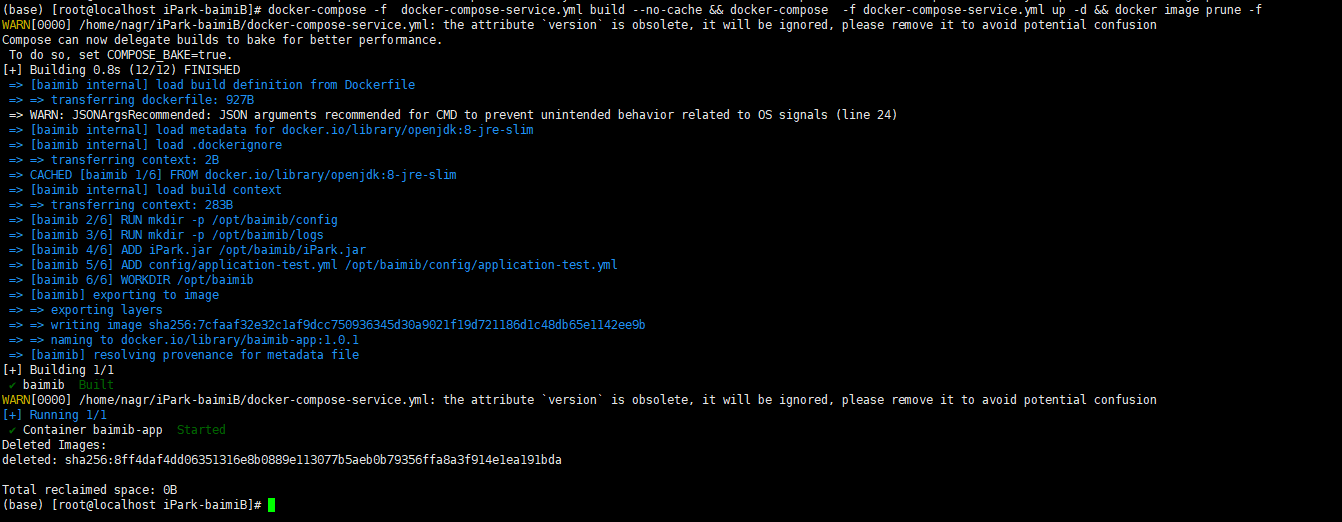


只需要将上述文件复制到Linux系统中，修改yml对应的路径，因为我在配置文件中写的是绝对路径。并添加对应项目的jar包，需要修改相关的配置。



在Linux系统中的结构应该是这样的。最后在Linux系统中执行

docker-compose -f docker-compose-service.yml build --no-cache && docker-compose -f docker-compose-service.yml up -d && docker image prune -f



清空多余的缓存

Podman/docker system prune –a

docker volume rm $(docker volume ls -qf dangling=true)

**命令的作用是：**

Docker 会检查它管理的所有数据卷，然后只显示那些当前没有任何容器关联的、处于孤立状态的数据卷。

**这个命令通常用于以下目的：**

* **清理未使用的磁盘空间：** 悬挂数据卷占用了磁盘空间，但不再被任何容器使用。你可以使用 docker volume prune 命令来删除所有悬挂数据卷，释放这些空间。在执行 prune 之前，通常会先使用 docker volume ls -f dangling=true 来查看将要被删除的数据卷，以避免意外删除重要数据。
* **诊断问题：** 有时，查看悬挂数据卷可以帮助你理解容器和数据卷的生命周期，以及是否存在未正确清理的资源。

总而言之，docker volume ls -f dangling=true 是一个用于查找和识别 Docker 中不再被任何容器使用的孤立数据卷的命令。