# Docker常用操作

## 1. 下载镜像

- 搜索镜像：docker search
- 下载镜像：docker pull
- 查看已下载镜像：docker images
- 删除镜像：docker rmi

dockerhub网站，常用的docker镜像搜索发布网站。https://hub.docker.com/

## 2. 启动容器

- 运行：docker run

  -d 启动容器后在后台运行

  -p 88:80 端口映射，将docker容器80端口映射到主机88端口

  -v /usr/html/:/usr/nginx/html 存储映射

  -v ngconf:/usr/nginx/conf 卷映射

  -e 设置容器运行的环境变量，具体开放的环境变量，以docker镜像说明文档为准

  —name 自定义容器名称
- 查看：docker ps
- 停止：docker stop
- 启动：docker start
- 重启：docker restart
- 状态：docker stats
- 日志：docker logs
- 删除：docker rm

  docker rm -f $(docker ps -aq) 可以快速删除所有容器，测试阶段特别好用。
- 进入：docker exec

  docker exec -it mynginx /bin/bash 以交互模式进入容器
- 容器信息：docker inspect

## 3. 修改页面

```
docker run --name mynginx -d -p 88:80 -v
/usr/bin/myhtml:/usr/share/nginx/html nginx
echo 222 > /usr/bin/myhtml/index.html
```

# 4. 保存分享

- 提交文件修改

  docker commit -a "hah" -m "update" 01d mynginx:v1.0.0

  将修改后的容器提交，形成一个新的镜像
- 保存镜像

  docker save -o mynginx.tar mynginx

  将docker镜像保存为一个.tar的文件
- 加载镜像文件

  docker load -i mynginx.tar

  加载tar文件格式的镜像

# 5. 存储管理

- 目录挂载

  外部自动创建目录

  -v /app/nghtml:/usr/share/nginx/html
- 卷挂载

  自定创建卷，同步容器内部的文件到卷

  -v ngconf:/ect/nginx
- 卷管理

  列出所有卷：docker volume ls

  查看具体卷的细节：docker volume inspect ngconf

# 6. 网络管理

- 查看网络

docker network ls

- 创建网络

docker network create netng

- 创建容器指定网络

docker run -d --name app01 --network netng -p 99:80 nginx

docker run -d --name app02 --network netng -p 99:80 nginx

docker exec -it app01

curl http://app02:99

# 7. 最佳实践

## 7.1 redis主从环境搭建

- 创建主节点

```
docker run -d --name redis01 -p 6379:6379 \
-v /data/app/redis01:/bitnami/redis/data \
-e REDIS_PASSWORD=123456 \
-e REDIS_REPLICATION_MODE=master \
--network net-redis bitnami/redis
```

- 创建从节点

```
docker run -d --name redis02 -p 6380:6379 \
-v /data/app/redis02:/bitnami/redis/data \
-e REDIS_PASSWORD=123456 \
-e REDIS_REPLICATION_MODE=slave \
-e REDIS_MASTER_HOST=redis01 \
-e REDIS_MASTER_PORT_NUMBER=6379 \
-e REDIS_MASTER_PASSWORD=123456 \
--network net-redis bitnami/redis
```

## 7.2 wordpress环境搭建

- 创建mysql

```
docker run -d --name mysql -p 3306:3306 \
-v /data/app/mysql/conf:/etc/mysql/conf.d \
-v mysqldata:/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=123456 \
-e MYSQL_DATABASE=wordpressdb \
--network blog mysql:8.4.0
```

- 创建wordpress

```
docker run -d --name wordpress -p 8080:80 \
-v wordpress:/var/www/html \
-e WORDPRESS_DB_HOST=mysql \
-e WORDPRESS_DB_USER=root \
-e WORDPRESS_DB_PASSWORD=123456 \
-e WORDPRESS_DB_NAME=wordpressdb \
--network blog wordpress
```

# 8. docker compose

## 8.1 创建yml文件

```
services:
  mysql:
    image: mysql:8.4.0
    restart: always
    ports:
      - 3306:3306
    volumes:
      - /data/app/mysql/conf:/etc/mysql/conf.d
      - mysqldata:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=123456
      - MYSQL_DATABASE=wordpressdb
    networks:
      - blog

  wordpress:
    image: wordpress
    restart: always
    ports:
      - 8080:80
    volumes:
      - wordpress:/var/www/html
    environment:
      WORDPRESS_DB_HOST: mysql
```

```
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: 123456
      WORDPRESS_DB_NAME: wordpressdb
    networks:
      - blog
    depends_on:
      - mysql

volumes:
  wordpress:
  mysqldata:
networks:
  blog:
```

## 8.2 docker compose

```
docker compose -f compose.yml up -d
docker ps
```

> 对于ubuntu必须使用sudo来运行docker，可以将当前用户假如到docker分组实现免sudo运行。命令：sudo usermod -aG docker $USER。重新登录可生效。或者运行命令 newgrp docker

## 8.3 geoserever

```
version: '2'

services:
    geoserver:
        build: .
        ports:
            - "8888:8080"
        volumes_from:
            # reference to the service which has the volume with the
preloaded geoserver_data_dir
            - data_dir_conf
    data_dir_conf:
        image: geonode/geoserver_data:2.18.2
        container_name: geoserver_data_dir # named data container
        command: /bin/true
        volumes:
            - /geoserver_data/data

volumes:
    # reference to the named data container that holds the preloaded
```

```
geoserver data directory
    geoserver_data_dir:
```