
参考文档: 文档 --nginx&docker

1,容器介绍

- 目的: 能够知道容器的含义
- 含义: 相当与一个操作系统一样, 里面可以运行各种服务,比如: mysql,nginx, mongo….

2,服务器介绍

- 目的: 理解各种服务器类型
- 服务器含义:
  - 1,服务器实际就是一台电脑, 里面可以安装对应的服务mysql,redis
  - 2, 服务器一般选型两种: 搭建自己的服务器, 使用第三方服务器

3,docker安装和目录

- 目的: 可以安装docker和知道常见的两个配置目录

- 安装过程: 看文档即可
-
- 配置目录:
  - 配置文件: /etc/docker
  - 镜像容器目录: /var/lib/docker

4,docker下载镜像源指定(sl模块)

- 目的;可以配置docker镜像的下载源
- 操作流程:
  - 1, 指定配置下载源,运行之后可以在/etc/docker/daemon.json

```
curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://74f21445.m.daocl
```

  - 2, 重启即可

```
sudo systemctl restart docker.service
```

5,镜像资源操作(参考文档)

6,容器的操作(参考文档)

7,私有仓库的搭建

- 目的: 可以在本地搭建私有仓库
- 操作流程:
  - 1, 下载registry镜像

```
docker pull registry
```

  - 启动仓库容器

```
docker run -d -p 5000:5000 registry
```

  - 2, 配置daemon.js私有仓库地址

```
{"registry-mirrors": ["http://74f21445.m.daocloud.io"], =="insecure-registries":["172.16
```

  - 3, 重启docker

```
systemctl restart docker
```

  - 4, 备份镜像名字: ip地址/其他名字

```
docker tag nginx:v1.0 172.16.12.134:5000/nginx
```

- 5, 创建registry容器

```
docker run -d -p 5000:5000 registry
```

- 6, 测试推送, 下载容器

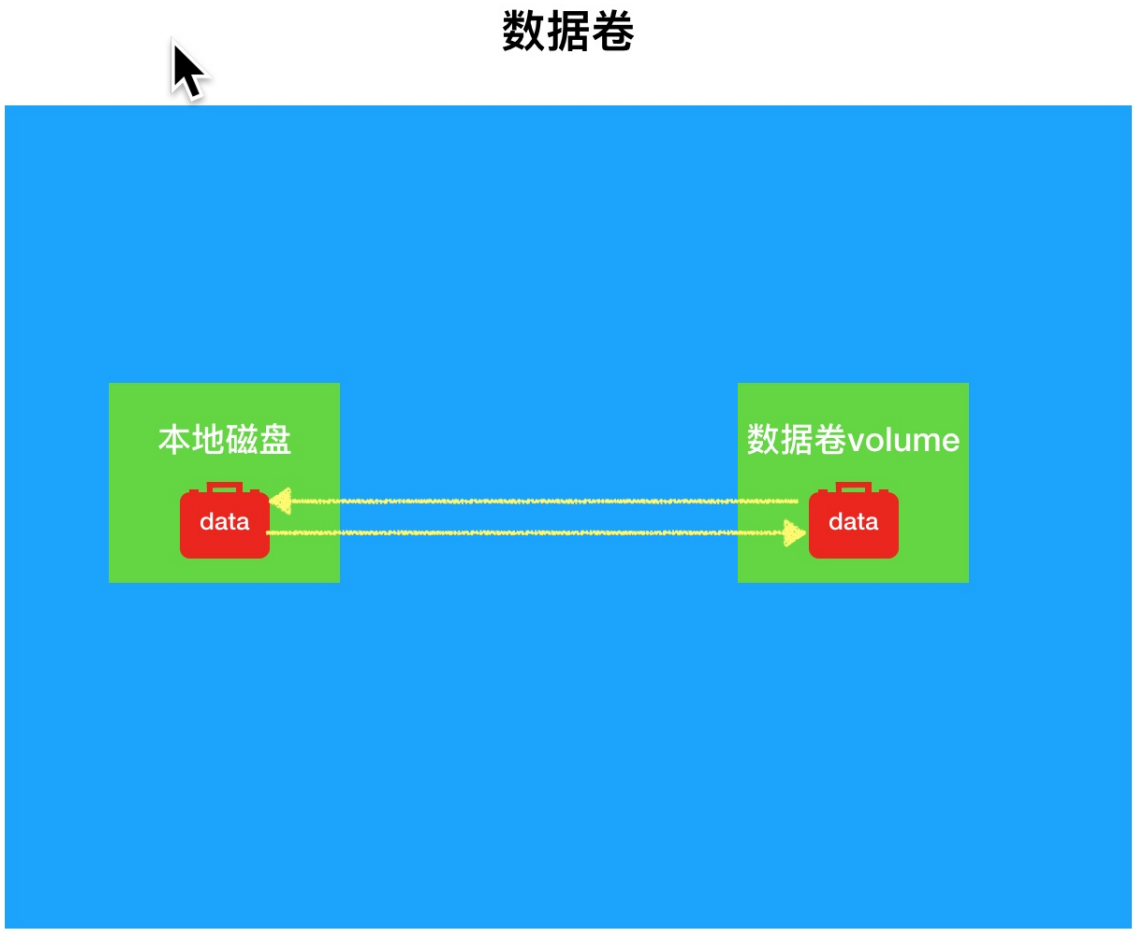
```
docker push 172.16.12.134:5000/nginx
docker pull 172.16.12.134:5000/nginx
```

8,数据卷

- 目的: 可以将宿主主机中的目录和容器进行关联,映射
- 操作流程:
  - 1, 现在本机桌面创建data目录
  - 2, 创建ubuntu容器,映射data目录

```
docker run -it --name=unbuntu1 -v ~/Desktop/data:/home ubuntu /bin/bash
```

- 图解:



9,数据卷volumes

- 目的: 可以使用系统提供的方法,对数据卷进行操作
  - 数据卷的操作命令

- 1, 查看

```
docker volume ls
```

- 2, 新建

```
docker volume create xxx
```

- 3,进入数据卷，一般放置在/var/lib/docker/volumes
- 3, 删除

```
docker volume rm xxx
```

- 操作流程:
  - 1, 创建数据卷django

```
docker volume create django
```

  - 2, 创建ubuntu容器和django关联

```
docker run -it -v django:/home ubuntu /bin/bash
```

  - 3, 删除容器,删除数据卷

```
docker rm 容器名
docker volume rm django
```

10,数据卷容器的使用

- 目的: 可以通过数据卷容器,将多个容器关联到一起
- 操作流程:
  - 1, 创建ubuntu1的数据卷容器

```
docker create  -v /data --name=ubuntu1 ubuntu
```

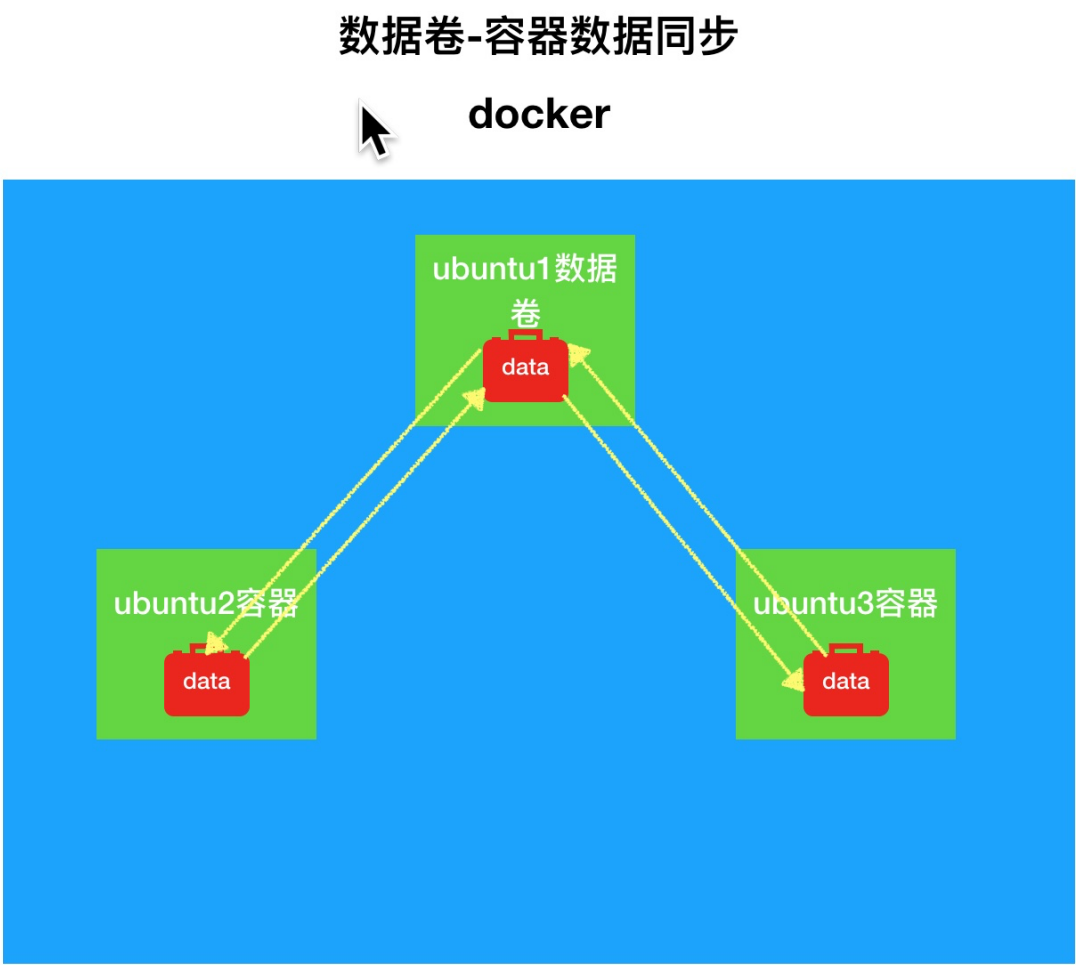
  - 2, 基于ubuntu1, 创建容器ubuntu2

```
docker run --volumes-from ubuntu1 -itd --name=ubuntu2 ubuntu
```

  - 3, 基于ubuntu1, 创建容器ubuntu3

```
docker run --volumes-from ubuntu1 -itd --name=ubuntu3 ubuntu
```

- 4,测试
  - 进入到任意的容器,在data中添加数据测试即可
- 图解:



11,端口映射使用

- 目的: 能够创建nginx容器的时候, 分配ip或者端口
- 有三种方式:
  - 1, 随机分配端口

```
docker run -itd -P --name=nginx1 nginx:v1.0
```

  - 2, 指定端口

```
docker run -itd -p 6001:80 --name=nginx2 nginx:v1.0
```

  - 3, 指定ip+端口

```
docker run -itd -p 172.16.12.134:6002:80 --name=nginx3 nginx:v1.0
```

12,网络模式说明

- 目的: 知道docker常见的四种网络模式

```
docker run -itd --name=nginx7 --network=bridge_test nginx:v1.0
```

1, 桥接模式，bridge

- 测试: 创建nginx1,2,3容器

```
docker run -itd --network=bridge --name=nginx1 nginx:v1.0
```

- 特点: **各个容器网段都是一样的,** 如果不指定桥接模式,默认就是

2, host模式

- 测试: nginx4容器

```
docker run -itd –network=host --name=nginx1 nginx:v1.0
```

- 特点: **和宿主主机,共用网卡,使用宿主机的ip和端口**

3,none模式

- 测试: nginx5容器

```
docker run -itd –network=none --name=nginx1 nginx:v1.0
```

- 特点:**分配独立网卡,不和其他容器相连**

4,container模式

- 测试: nginx6容器

```
docker run -itd --network=container:nginx2 --name=nginx6 nginx:v1.0
```

- 特点:**可以复制(拷贝)其他的容器的网络模式**

13,自定义network的使用

- 目的: 可以基于上面的四种网络模式,创建自己的模式
- 常见的命令:
  - 1, 查看网络模式

```
docker network ls
```

  - 2, 删除网络模式:

```
docker network remove xxx
```

  - 3, 创建网络模式bridge_test

```
docker network create --driver bridge bridge_test
```

- 4,创建容器nginx7,指定网络模式bridge_test

```
docker run -itd --name=nginx7 --network=bridge_test nginx:v1.0
```

- 5, 断掉bridge_test网络模式

```
docker network disconnect bridge_test nginx7
```

- 6, 连接bridge_test网络模式

```
docker network connect bridge_test nginx7
```