

# 大型架构及配置技术

**NSD ARCHITECTURE**

**DAY02**

# 内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	自定义镜像与仓库
	10:30 ~ 11:20	
	11:30 ~ 12:00	持久化存储
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	Docker网络架构
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



# 自定义镜像与仓库

---



# 自定义镜像

---

# docker commit

- 使用镜像启动容器，在该容器基础上修改
- 另存为另一个镜像

```
[root@jacob ~]# docker run -itd centos bash
[root@jacob ~]# docker ps
[root@jacob ~]# docker exec -it IDs bash
修改（增删改数据、安装软件、修改配置文件等）
```

```
[root@jacob ~]# docker commit IDs name:label
[root@jacob ~]# docker images
```



# Dockerfile

- Dockerfile语法格式
  - FROM:基础镜像
  - MAINTAINER:镜像创建者信息
  - EXPOSE:开放的端口
  - ENV:设置变量
  - ADD:复制文件到镜像
  - RUN:制作镜像时执行的命令，可以有多个
  - WORKDIR:定义容器默认工作目录
  - CMD:容器启动时执行的命令，仅可以有一条CMD



# Dockerfile ( 续1 )

- 使用Dockerfile工作流程
  - mkdir build; cd build
  - vim Dockerfile
  - docker build -t imagename Dockerfile



# Dockerfile ( 续2 )

- Dockerfile文件案例

```
[root@jacob build]# cat Dockerfile
FROM centos
MAINTAINER Jacob redhat@163.com
ENV NAME=Jacob
ENV environment=test
WORKDIR /var/www/html
ADD test.sh /root/test.sh
RUN mkdir /dockerfile
RUN echo "test" > /dockerfile/file.txt
RUN yum -y install httpd
RUN echo "test" > /var/www/html/index.html
EXPOSE 80
CMD [ "httpd", "-DFOREGROUND"]
```





# 案例1：制作自定义镜像

## 制作自定义镜像

- 基于centos镜像使用commit创建新的镜像文件
- 基于centos镜像使用Dockerfile文件创建新的镜像文件

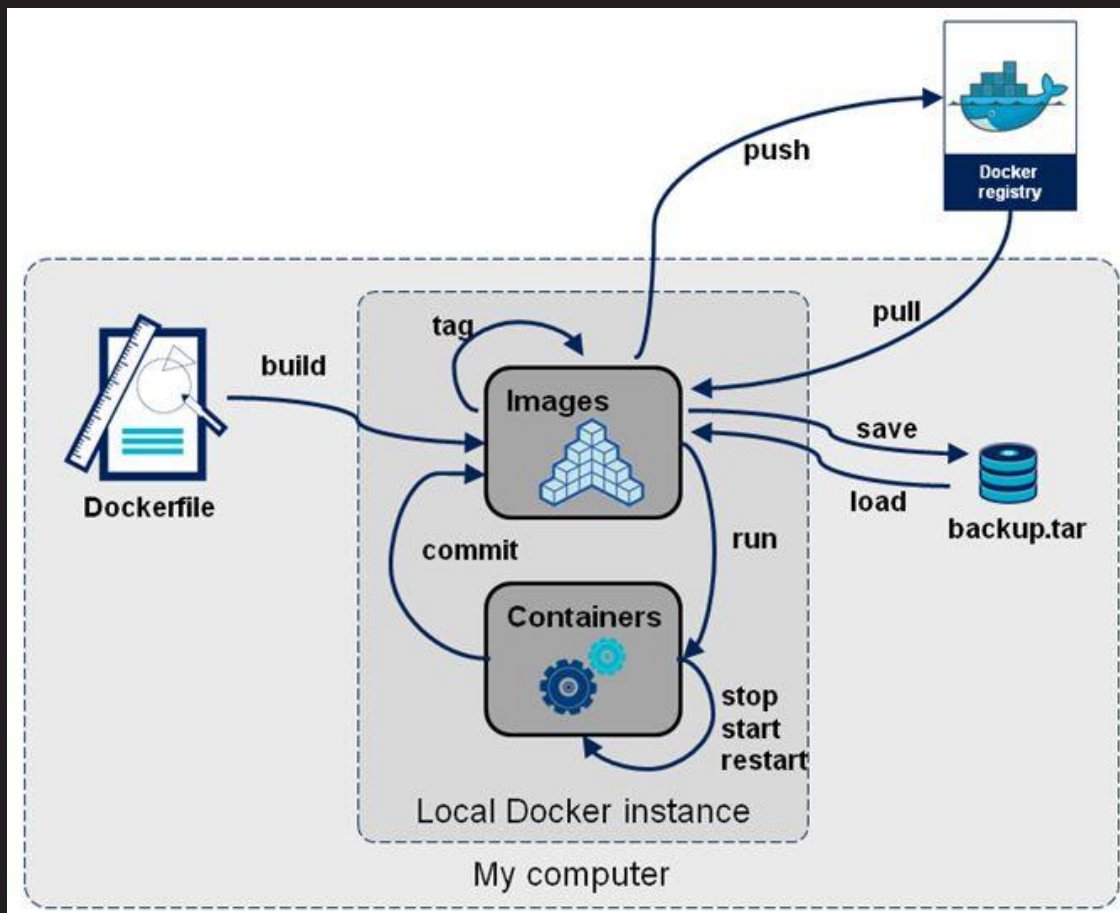


# 自定义镜像仓库

---

# registry基本概念

- 共享镜像的一台服务器（镜像化的一台服务器）



# 自定义私有仓库

- 流程：
  - docker pull registry
  - vim /usr/lib/systemd/system/docker.service  
    ExecStart=/usr/bin/dockerd --insecure-registry=ip:5000
  - systemctl daemon-reload
  - systemctl restart docker
  - docker run -id -p 5000:5000 registry
  - docker tag 镜像 IP:5000/镜像:label
  - docker push IP:5000/镜像:label
- 进入registry容器查看/etc/docker/registry/config.yml



## 案例2：创建私有镜像仓库

构建私有镜像仓库：

- Docker主机：192.168.4.100
- 镜像仓库服务器：192.168.4.5



# 持久化存储



# 存储卷



# 卷的概念

- docker容器不保持任何数据
- 重要数据请使用外部卷存储（数据持久化）
- 容器可以挂载真实机目录或共享存储为卷





# 主机卷的映射

- 将真实机目录挂载到容器中提供持久化存储

```
[root@jacob ~]# docker run -v /data:/data -it centos bash
```



# 共享存储



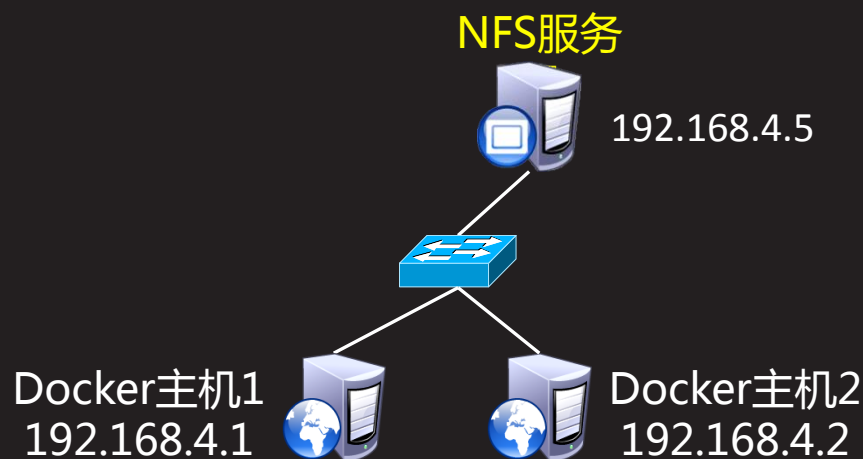
# 共享存储基本概念

- 一台共享存储服务器可以提供给所有Docker主机使用
- 共享存储服务器（NAS、SAN、DAS等）
- 如：
  - 使用NFS创建共享存储服务器
  - 客户端挂载NFS共享，并最终映射到容器中



# 使用共享存储的案例

- 服务器
  - `yum -y install nfs-utils`
  - `vim /etc/exports`
  - `systemctl start nfs`
- Docker主机
  - mount挂载共享
  - 运行容器时，使用-v选项映射磁盘到容器中



## 案例3：NFS共享存储

### 1. 服务器创建NFS共享存储

1. 共享目录为/content，权限为所有主机rw

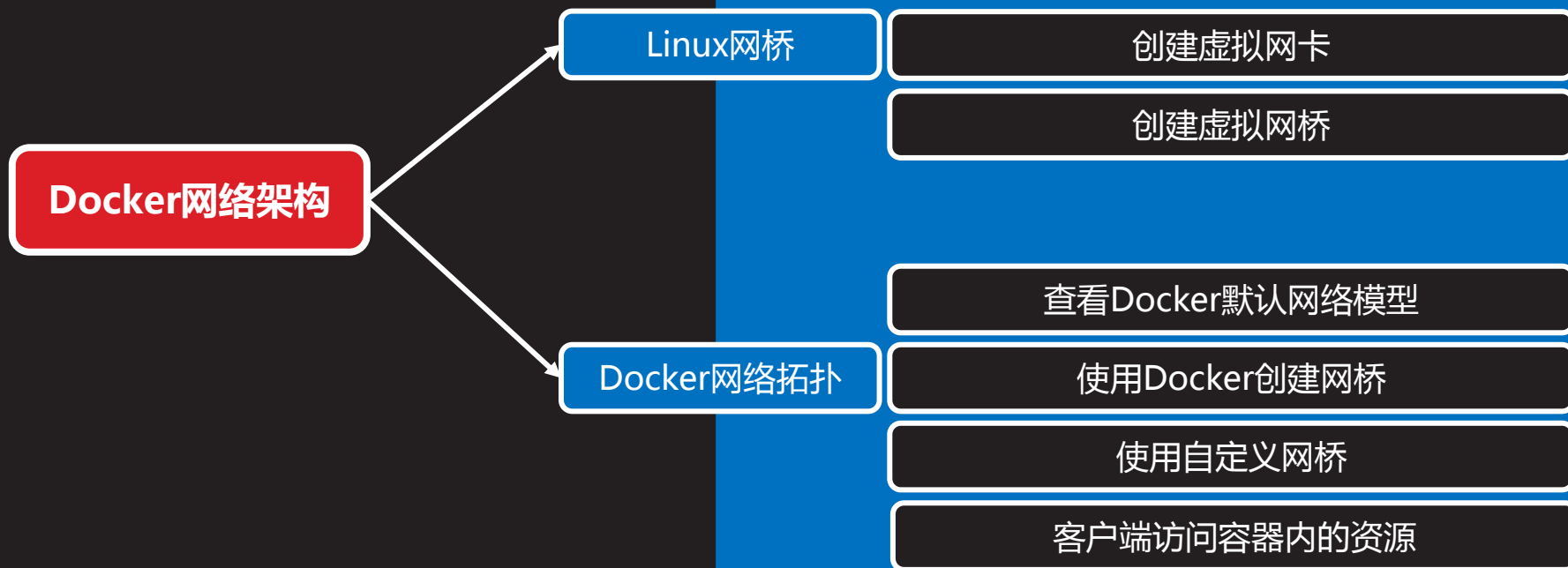
### 2. 客户端挂载共享

1. 将共享目录映射到容器中



# Docker网络架构

---



# Linux网桥

---

# 创建虚拟网卡

- 真实网卡配置文件
  - cat /etc/sysconfig/network-scripts/ifcfg-eth0
- 虚拟网卡配置文件
  - cat /etc/sysconfig/network-scripts/ifcfg-eth0:0

```
[root@jacob ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0:0
TYPE=Ethernet
BOOTPROTO=static
... ..
NAME=eth0:0
DEVICE=eth0:0
ONBOOT=yes
IPADDR=192.168.4.15
```





# 创建虚拟网桥

```
[root@jacob ~]# cat /etc/sysconfig/network-scripts/ifcfg-br0  
TYPE=Bridge  
BOOTPROTO=static
```

```
... ..
```

```
NAME=br0  
DEVICE=br0  
ONBOOT=yes  
IPADDR=192.168.4.15
```

```
[root@jacob ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0  
TYPE=Ethernet  
BOOTPROTO=static
```

```
... ..
```

```
NAME=eth0  
DEVICE=eth0  
BRIDGE=br0  
ONBOOT=yes  
IPADDR=192.168.4.15
```

```
[root@jacob ~]# ~]# brctl show
```



# Docker网络拓扑



# 查看Docker默认网络模型

- 查看默认Docker创建的网络模型

```
[root@jacob ~]# docker network list
```

NETWORK ID	NAME	DRIVER	SCOPE	
c0ae28d57b18	bridge	bridge	local	桥接模型
b69d4c0c735f	host	host	local	主机模型
4dc88be13b81	none	null	local	无网络

```
[root@jacob ~]# ip a s docker0
```

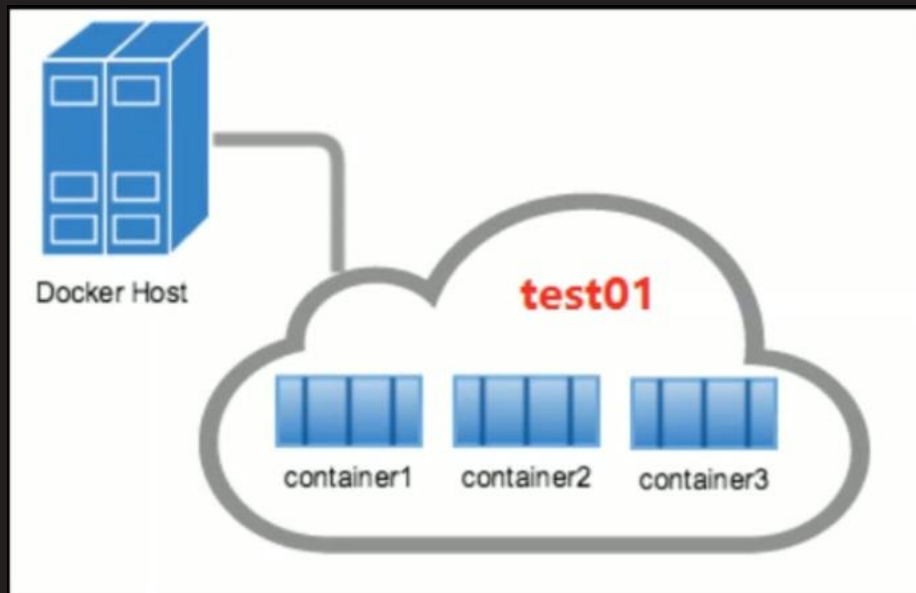
```
[root@jacob ~]# brctl show docker0 //启动容器会绑定该网桥
```



# 使用Docker创建网桥

- 新建Docker网络模型

```
[root@jacob ~]# docker network create --driver bridge test01
[root@jacob ~]# docker network list
[root@jacob ~]# ip a s
[root@jacob ~]# docker network inspect test01
```



# 使用Docker创建网桥（续1）

- 查看默认Docker创建的网络模型
  - 自定义网段

```
[root@jacob ~]# docker network create --subnet=172.30.0.0/16 test01
```



# 使用自定义网桥

- 启动容器，使用刚刚创建的自定义网桥

```
[root@jacob ~]# docker run --network=bridge|host|none ... ..  
[root@jacob ~]# docker run --network=test01 -id nginx
```



# 客户端访问容器内的资源

- 默认容器通过SNAT可以访问外网
- 但外部网络的主机不可以访问容器内的资源
- 端口映射
  - 使用端口映射可以实现外部网络访问容器内的资源

```
[root@jacob ~]# docker run -p 8080 80 -id nginx
```

//如：真实机IP为192.168.4.5 ,  
使用-p映射真实机的8080端口到容器中的80端口

```
[root@client ~]# firefox http://192.168.4.5
```



## 案例4：创建自定义网桥

### 1. 创建自定义网桥

1. 创建网桥设备docker1
2. 设定网段为172.30.0.0/16

### 2. 启动nginx容器

1. nginx容器桥接docker1设备
2. 映射真实机8080端口与容器的80端口





# 总结和答疑

---

总结和答疑

提交镜像

问题现象

故障分析及排除

# 提交镜像



# 问题现象

- 推送镜像到registry，提示错误：

```
[root@jacob ~]# docker push centos
The push refers to a repository [docker.io/library/centos]
Put https://index.docker.io/v1/repositories/library/centos/: dial tcp:
lookup index.docker.io on 172.40.1.10:53: read udp
172.40.50.118:43696->172.40.1.10:53: i/o timeout
```



# 故障分析及排除

- 原因分析
  - 问题1：提示The push refers to a repository [docker.io/library/centos]
- 解决办法
  - 问题1：先要修改镜像tag，在可以继续push镜像到registry

