Docker入门与应用实战

讲师介绍

阿良

资深运维工程师，51CTO知名博主。曾就职在IDC，大数据，金融行业，现任职奇虎360公司。经重重磨炼，具有丰富的运维实战经验。

技术博客：http://blog.51cto.com/lizhenliang

Docker/K8s技术学员群：397834690

课程目录

学完这门课程会获得什么？

- 掌握Docker核心概念
- 熟悉Docker工作原理
- 独立使用Docker部署应用程序
- 接入CI/CD，实现环境标准化

入门须知

- 熟悉Linux操作系统
- 了解域名解析原理
- 了解网络协议

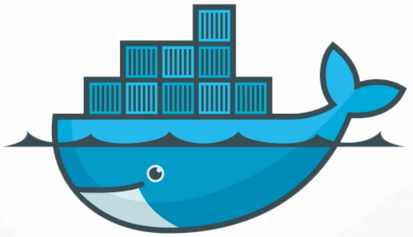
第 1 章 Docker概述

1. Docker是什么
2. Docker设计目标
3. Docker基本组成
4. 容器 vs 虚拟机
5. Docker应用场景

Docker是什么

- 使用最广泛的开源容器引擎
- 一种操作系统级的虚拟化技术
- 依赖于Linux内核特性：Namespace（资源隔离）和Cgroups（资源限制）
- 一个简单的应用程序打包工具

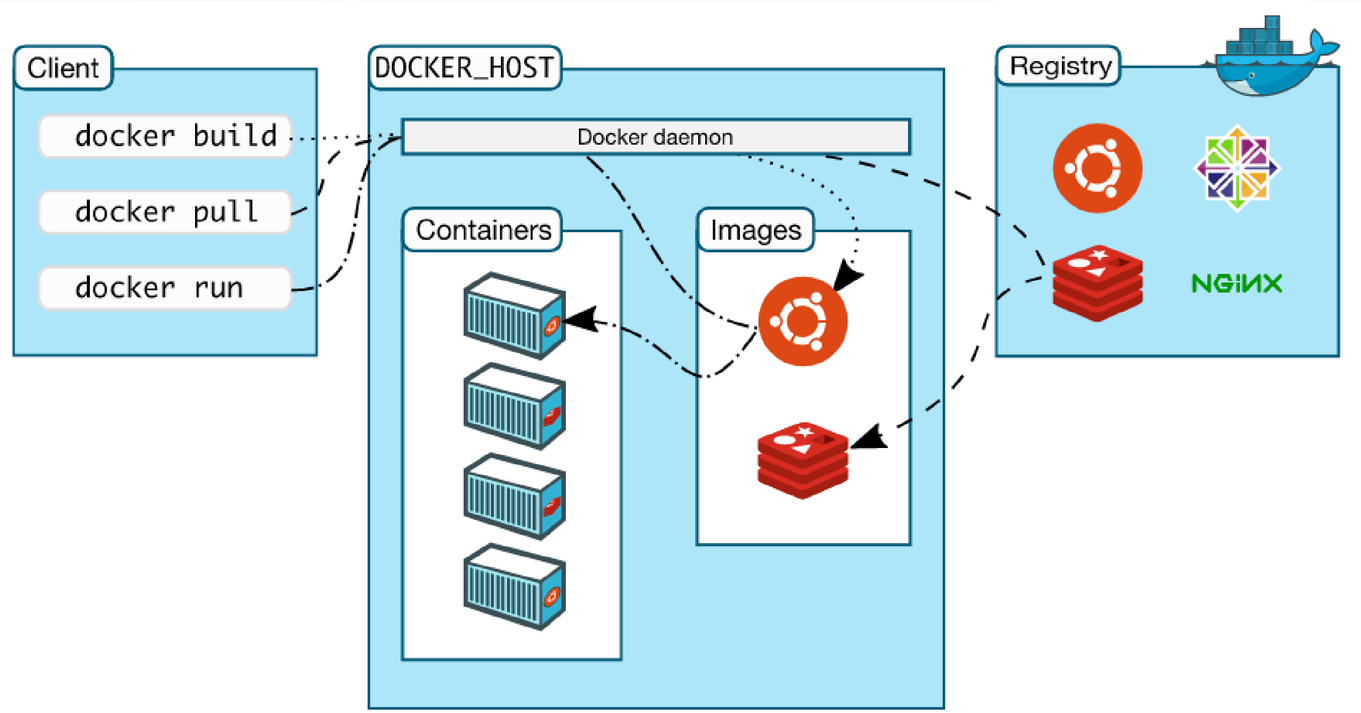
Docker设计目标



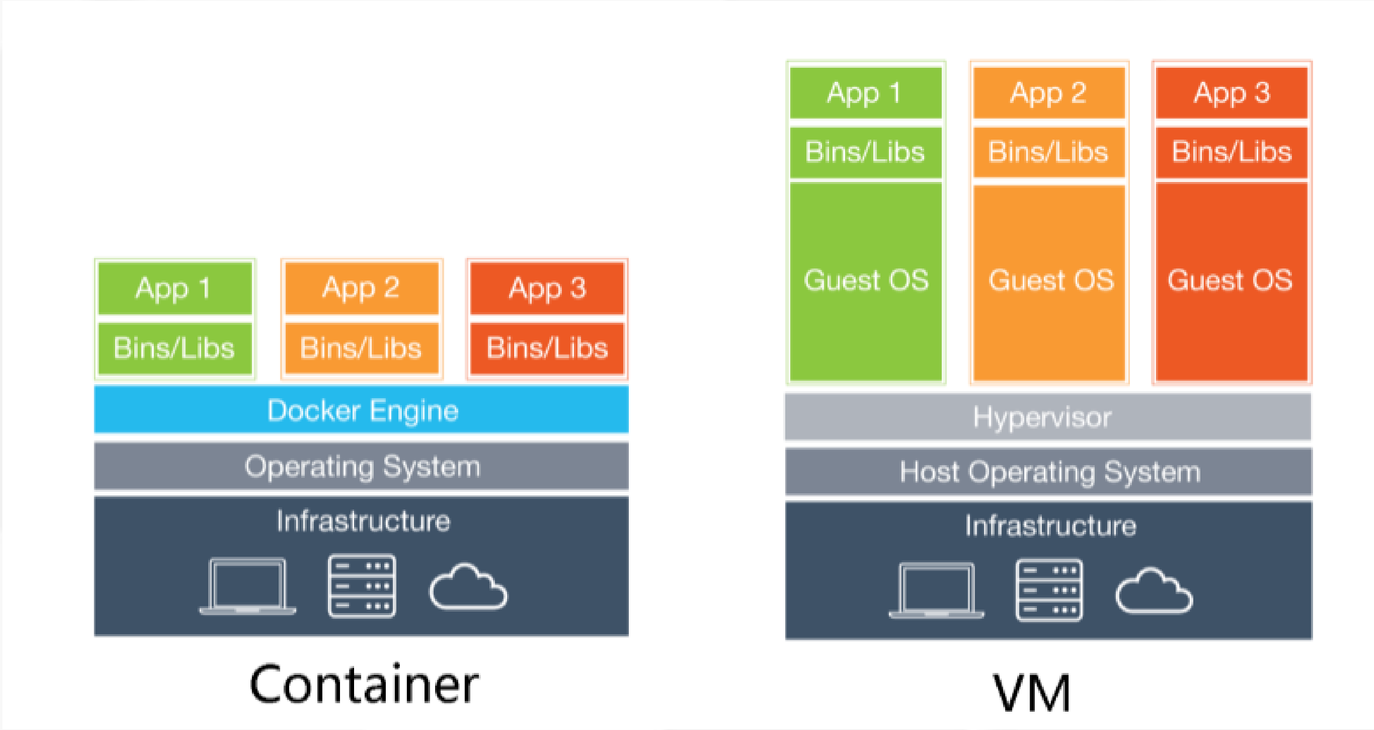
- 提供简单的应用程序打包工具
- 开发人员和运维人员职责逻辑分离
- 多环境保持一致性

Docker基本组成

- Docker Client：客户端
- Ddocker Daemon：守护进程
- Docker Images：镜像
- Docker Container：容器
- Docker Registry：镜像仓库



容器 VS 虚拟机


App 1
Bins/Libs
App 2
Bins/Libs
App 3
Bins/Libs
Docker Engine
Operating System
Infrastructure
Container
App 1
Bins/Libs
Guest OS
App 2
Bins/Libs
Guest OS
App 3
Bins/Libs
Guest OS
Hypervisor
Host Operating System
Infrastructure
VM

容器 VS 虚拟机

	Container	VM
启动速度	秒级	分钟级
运行性能	接近原生	5%左右损失
磁盘占用	MB	GB
数量	成百上千	一般几十台
隔离性	进程级别	系统级（更彻底）
操作系统	只支持Linux	几乎所有
封装程度	只打包项目代码和依赖关系，共享宿主机内核	完整的操作系统

Docker应用场景

- 应用程序打包和发布
- 应用程序隔离
- 持续集成
- 部署微服务
- 快速搭建测试环境
- 提供PaaS产品（平台即服务）

第 2 章 Linux安装Docker

1. Docker版本
2. 支持平台
3. CentOS7.x安装Docker

Docker版本

- 社区版（Community Edition，CE)
- 企业版（Enterprise Edition，EE)

支持平台

- Linux（CentOS,Debian,Fedora,Oracle Linux,RHEL,SUSE和Ubuntu）
- Mac
- Windows

CentOS7.x安装Docker

```
# 安装依赖包
yum install -y yum-utils device-mapper-persistent-data lvm2
# 添加Docker软件包源
yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
# 安装Docker CE
yum install -y docker-ce
# 启动Docker服务并设置开机启动
systemctl start docker
systemctl enable docker
```

官方文档：https://docs.docker.com

第 3 章 镜像管理

1. 镜像是什么
2. 镜像与容器联系
3. 管理镜像常用命令

镜像是什么

镜像是什么?

- 一个分层存储的文件
- 一个软件的环境
- 一个镜像可以创建N个容器
- 一种标准化的交付
- 一个不包含Linux内核而又精简的Linux操作系统

镜像不是一个单一的文件，而是有多层构成。我们可以通过docker history <ID/NAME> 查看镜像中各层内容及大小，每层对应着Dockerfile中的一条指令。Docker镜像默认存储在/var/lib/docker/\<storage-driver\>中。

镜像从哪里来?

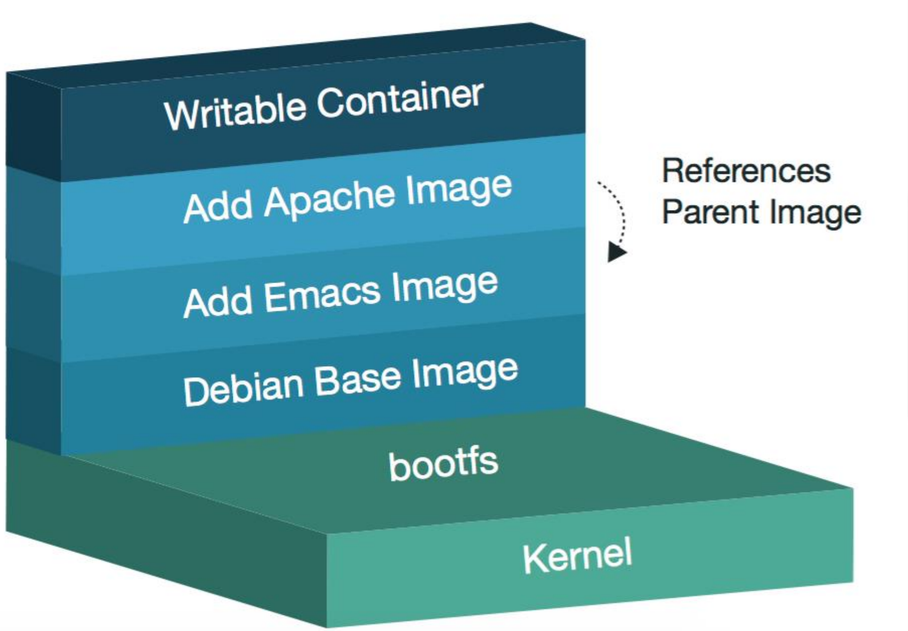
Docker Hub是由Docker公司负责维护的公共注册中心，包含大量的容器镜像，Docker工具默认从这个公共镜像库下载镜像。

地址：https://hub.docker.com/explore

配置镜像加速器：https://www.daocloud.io/mirror

curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://f1361db2.m.daocloud.io

镜像与容器联系



如图，容器其实是在镜像的最上面加了一层读写层，在运行容器里文件改动时，会先从镜像里要写的文件复制到容器自己的文件系统中（读写层）。

如果容器删除了，最上面的读写层也就删除了，改动也就丢失了。所以无论多少个容器共享一个镜像，所做的写操作都是从镜像的文件系统中复制过来操作的，并不会修改镜像的源文件，这种方式提高磁盘利用率。

若想持久化这些改动，可以通过docker commit 将容器保存成一个新镜像。

管理镜像常用命令

指令	描述
ls	列出镜像
build	构建镜像来自Dockerfile
history	查看镜像历史
inspect	显示一个或多个镜像详细信息
pull	从镜像仓库拉取镜像
push	推送一个镜像到镜像仓库
rm	移除一个或多个镜像
prune	移除未使用的镜像。没有被标记或被任何容器引用的。
tag	创建一个引用源镜像标记目标镜像
export	导出容器文件系统到tar归档文件
import	导入容器文件系统tar归档文件创建镜像
save	保存一个或多个镜像到一个tar归档文件
load	加载镜像来自tar归档或标准输入

第 4 章 容器管理

1. 创建容器常用选项
2. 容器资源限制
3. 管理容器常用命令

创建容器常用选项

选项	描述
-i, –interactive	交互式
-t, –tty	分配一个伪终端
-d, –detach	运行容器到后台
-e, –env	设置环境变量
-p, –publish list	发布容器端口到主机
-P, –publish-all	发布容器所有EXPOSE的端口到宿主机随机端口
–name string	指定容器名称
-h, –hostname	设置容器主机名
–ip string	指定容器IP，只能用于自定义网络
–network	连接容器到一个网络
–mount mount	将文件系统附加到容器
-v, –volume list	绑定挂载一个卷
–restart string	容器退出时重启策略，默认no，可选值：[always|on-failure]

容器资源限制

选项	描述
-m，–memory	容器可以使用的最大内存量
–memory-swap	允许交换到磁盘的内存量
–memory-swappiness=<0-100>	容器使用SWAP分区交换的百分比（0-100，默认为-1）
–oom-kill-disable	禁用OOM Killer
–cpus	可以使用的CPU数量
–cpuset-cpus	限制容器使用特定的CPU核心，如(0-3, 0,1)
–cpu-shares	CPU共享（相对权重）

容器资源限制

示例：

内存限额：

允许容器最多使用500M内存和100M的Swap，并禁用 OOM Killer：

docker run -d --name nginx03 --memory="500m" --memory-swap="600m" --oom-kill-disable nginx

CPU限额：

允许容器最多使用一个半的CPU：

docker run -d --name nginx04 --cpus="1.5" nginx

允许容器最多使用50%的CPU：

docker run -d --name nginx05 --cpus=".5" nginx

管理容器常用命令

选项	描述
ls	列出容器
inspect	查看一个或多个容器详细信息
exec	在运行容器中执行命令
commit	创建一个新镜像来自一个容器
cp	拷贝文件/文件夹到一个容器
logs	获取一个容器日志
port	列出或指定容器端口映射
top	显示一个容器运行的进程
stats	显示容器资源使用统计
stop/start	停止/启动一个或多个容器
rm	删除一个或多个容器

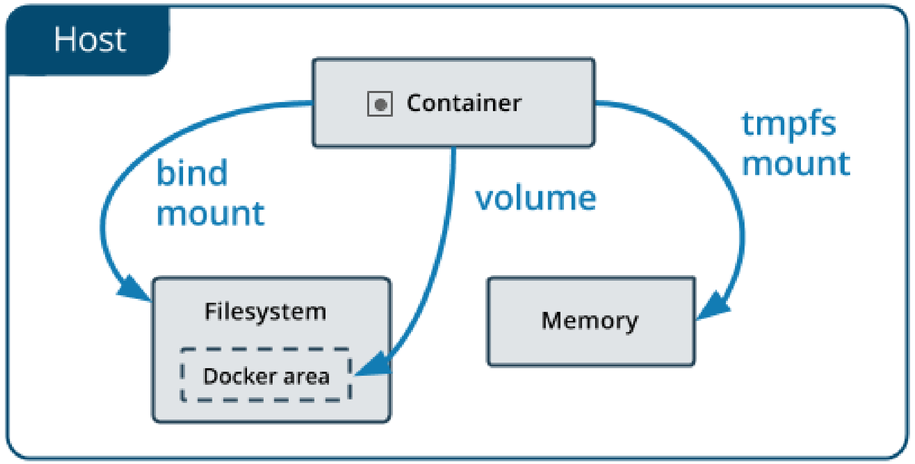
第 5 章 管理应用程序数据

1. 将数据从宿主机挂载到容器中的三种方式
2. Volume
3. Bind Mounts

将数据从宿主机挂载到容器中的三种方式

Docker提供三种方式将数据从宿主机挂载到容器中：

- volumes：Docker管理宿主机文件系统的一部分（/var/lib/docker/volumes）。保存数据的最佳方式。
- bind mounts：将宿主机上的任意位置的文件或者目录挂载到容器中。
- tmpfs：挂载存储在主机系统的内存中，而不会写入主机的文件系统。如果不希望将数据持久存储在任何位置，可以使用tmpfs，同时避免写入容器可写层提高性能。



Volume

管理卷：

docker volume create nginx-vol

docker volume ls

docker volume inspect nginx-vol

用卷创建一个容器：

docker run -d --name=nginx-test --mount src=nginx-vol,dst=/usr/share/nginx/html nginx

docker run -d --name=nginx-test -v nginx-vol:/usr/share/nginx/html nginx

清理：

docker stop nginx-test

docker rm nginx-test

docker volume rm nginx-vol

注意：

1. 如果没有指定卷，自动创建。
2. 建议使用--mount，更通用。

https://docs.docker.com/engine/admin/volumes/volumes/#start-a-container-with-a-volume

Bind Mounts

用卷创建一个容器：

docker run -d -it --name=nginx-test --mount type=bind,src=/app/wwwroot,dst=/usr/share/nginx/html nginx

docker run -d -it --name=nginx-test -v /app/wwwroot:/usr/share/nginx/html nginx

验证绑定：

docker inspect nginx-test

清理：

docker stop nginx-test

docker rm nginx-test

»¹¿ÉÒÔÖ±½ÓÕâÑùdocker run \ -p 27917:27017 \-v /my/mongo/db:/data/db \-d mongo:3.6

注意：

1. 如果源文件/目录没有存在，不会自动创建，会抛出一个错误。
2. 如果挂载目标在容器中非空目录，则该目录现有内容将被隐藏。

https://docs.docker.com/engine/admin/volumes/bind-mounts/#start-a-container-with-a-bind-mount

小结

Volume特点：

- 多个运行容器之间共享数据。
- 当容器停止或被移除时，该卷依然存在。
- 多个容器可以同时挂载相同的卷。
- 当明确删除卷时，卷才会被删除。
- 将容器的数据存储在远程主机或其他存储上
- 将数据从一台Docker主机迁移到另一台时，先停止容器，然后备份卷的目录（/var/lib/docker/volumes/）

Bind Mounts特点：

- 从主机共享配置文件到容器。默认情况下，挂载主机/etc/resolv.conf到每个容器，提供DNS解析。
- 在Docker主机上的开发环境和容器之间共享源代码。例如，可以将Maven target目录挂载到容器中，每次在Docker主机上构建Maven项目时，容器都可以访问构建的项目包。
- 当Docker主机的文件或目录结构保证与容器所需的绑定挂载一致时

第 6 章 容器网络

1. 网络模式
2. 容器网络访问原理

网络模式

- **bridge**

  –net=bridge

  默认网络，Docker启动后创建一个docker0网桥，默认创建的容器也是添加到这个网桥中。
- **host**

  –net=host

  容器不会获得一个独立的network namespace，而是与宿主机共用一个。这就意味着容器不会有自己的网卡信息，而是使用宿主机的。容器除了网络，其他都是隔离的。
- **none**

  –net=none

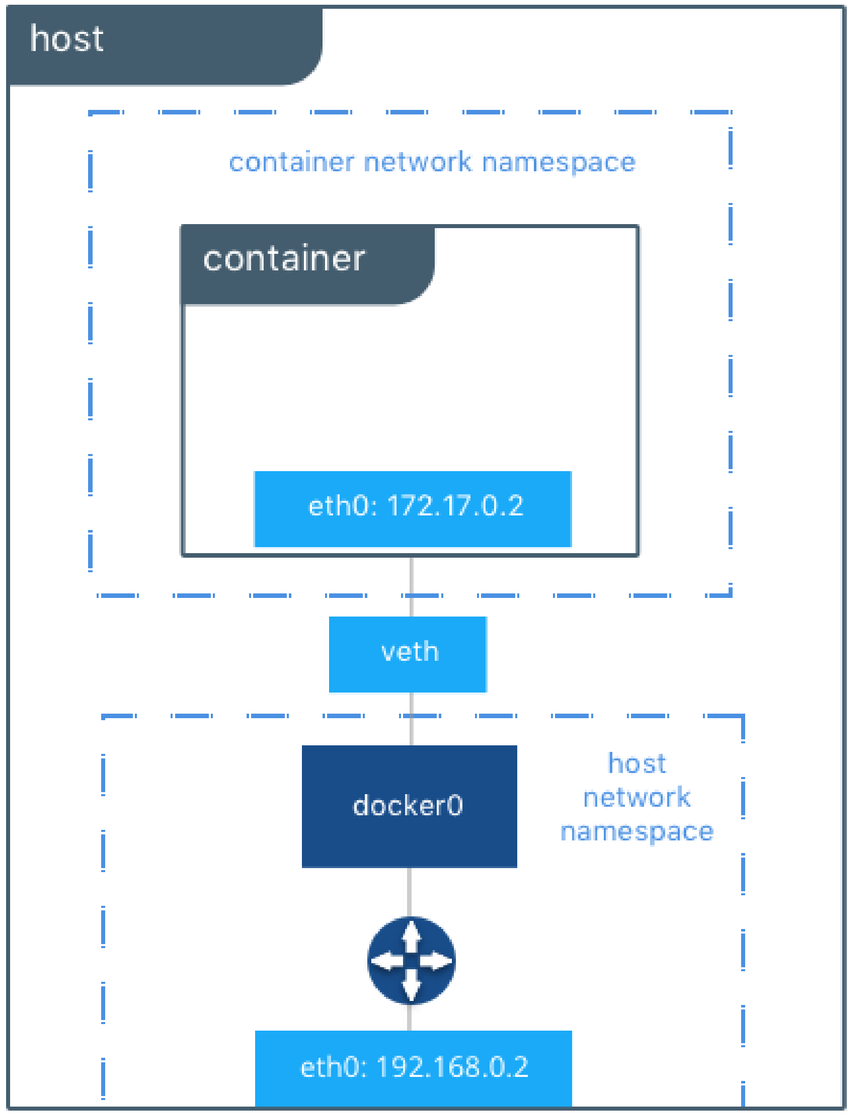
  获取独立的network namespace，但不为容器进行任何网络配置，需要我们手动配置。
- **container**

  –net=container:Name/ID

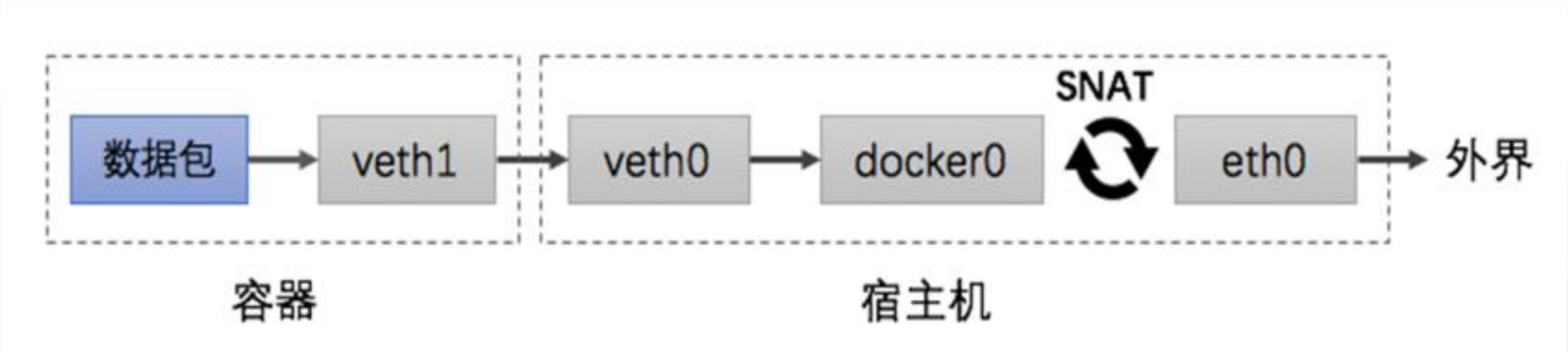
  与指定的容器使用同一个network namespace，具有同样的网络配置信息，两个容器除了网络，其他都还是隔离的。
- **自定义网络**

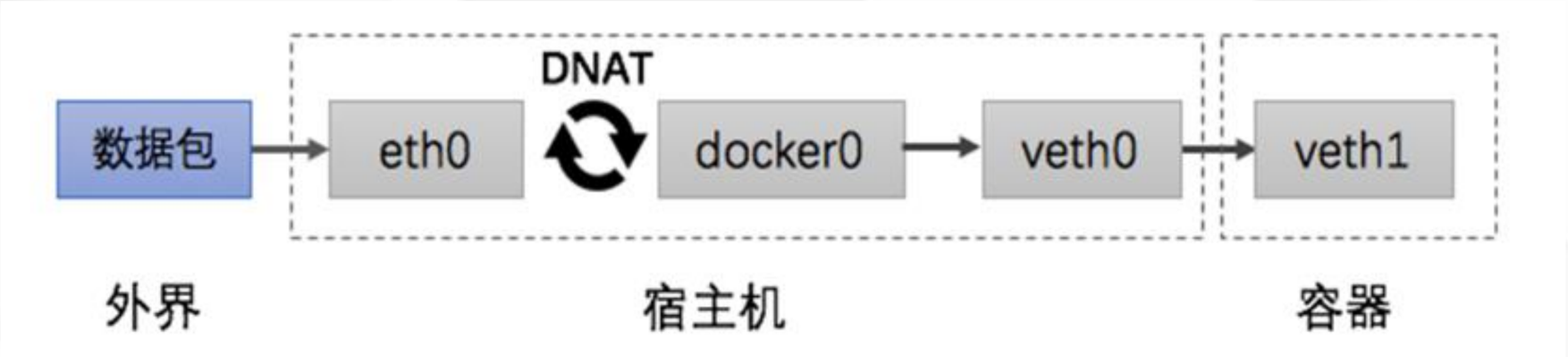
  与默认的bridge原理一样，但自定义网络具备内部DNS发现，可以通过容器名或者主机名容器之间网络通信。

容器网络访问原理


host
container network namespace
container
eth0: 172.17.0.2
veth
docker0
host
network
namespace
eth0: 192.168.0.2

容器网络访问原理

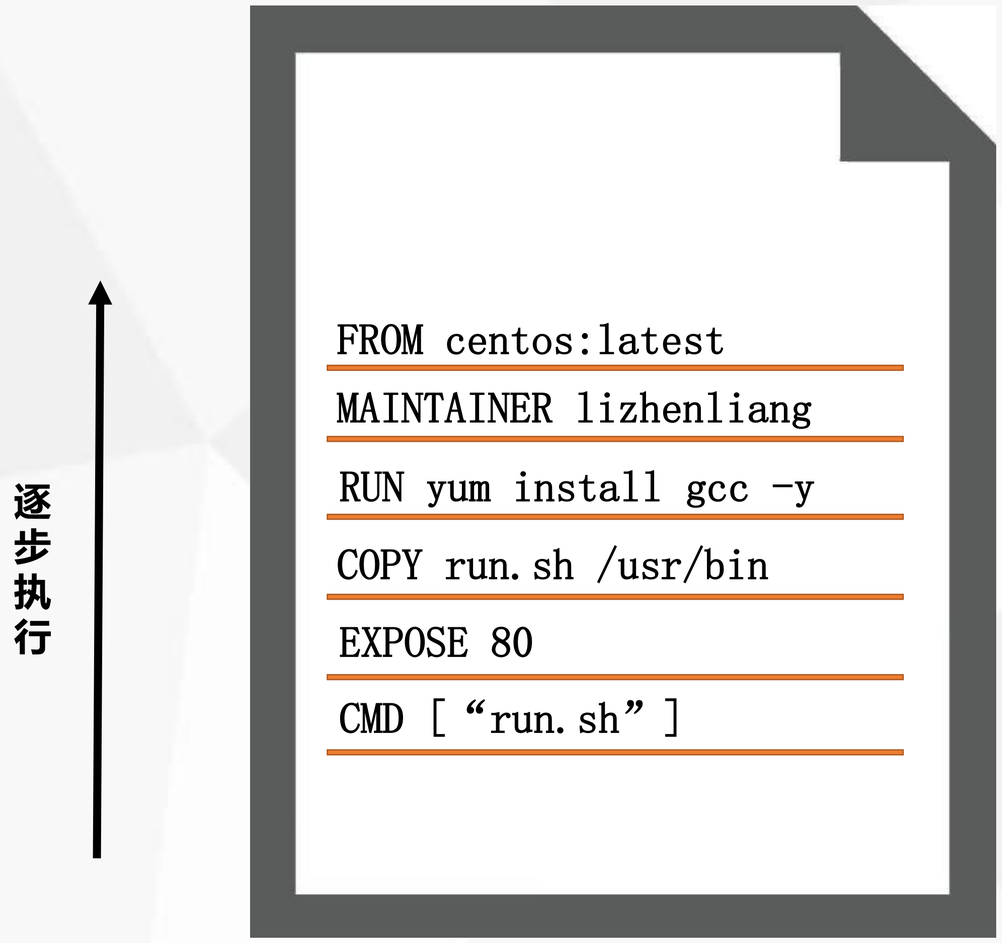

SNAT
数据包
veth1
veth0
docker0
eth0
外界
容器
宿主机


DNAT
数据包
eth0
docker0
veth0
veth1
外界
宿主机
容器

第 7 章 Dockerfile

1. Dockerfile格式
2. Dockerfile指令
3. Build镜像
4. 构建Nginx，PHP，Tomcat基础镜像
5. 快速搭建LNMP网站平台

Dockerfile格式



Dockerfile指令

指令	描述
FROM	构建新镜像是基于哪个镜像
MAINTAINER	镜像维护者姓名或邮箱地址
RUN	构建镜像时运行的Shell命令
COPY	拷贝文件或目录到镜像中
ENV	设置环境变量
USER	为RUN、CMD和ENTRYPOINT执行命令指定运行用户
EXPOSE	声明容器运行的服务端口
HEALTHCHECK	容器中服务健康检查
WORKDIR	为RUN、CMD、ENTRYPOINT、COPY和ADD设置工作目录
ENTRYPOINT	运行容器时执行，如果有多个ENTRYPOINT指令，最后一个生效
CMD	运行容器时执行，如果有多个CMD指令，最后一个生效

Build镜像

Usage: docker build [OPTIONS] PATH | URL | - [flags]

Options:

-t, --tag list   # 镜像名称

-f, --file string # 指定Dockerfile文件位置

docker build .

docker build -t shykes/myapp .

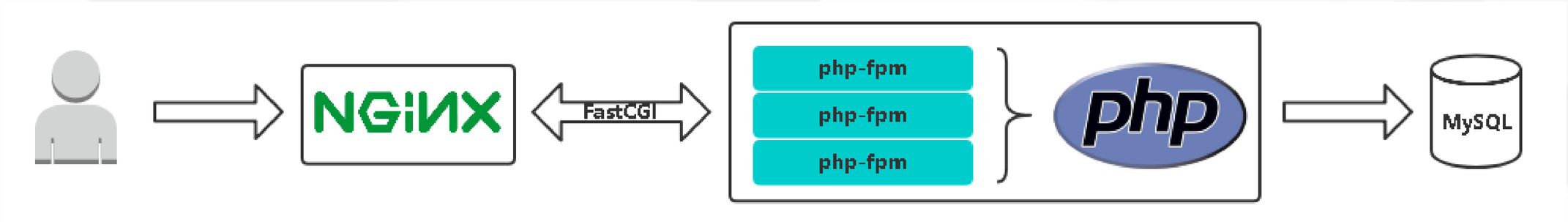
docker build -t shykes/myapp -f /path/Dockerfile /path

docker build -t shykes/myapp http://www.example.com/Dockerfile

构建业务基础镜像

- 构建Nginx基础镜像
- 构建PHP基础镜像
- 构建Tomcat基础镜像

快速部署LNMP网站平台



快速部署LNMP网站平台

1、自定义网络

docker network create lnmp

2、创建Mysql容器

```
docker run -d \
--name lnmp_mysql \
--net lnmp \
--mount src=mysql-vol,dst=/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=123456 -e MYSQL_DATABASE=wordpress  mysql:5.7  --character-set-server=utf8
```

3、创建PHP容器

```
docker run -d --name lnmp_php --net lnmp --mount src=wwwroot,dst=/wwwroot php:v1
```

4、创建Nginx容器

```
docker run -d --name lnmp_nginx --net lnmp -p 88:80 \
--mount type=bind,src=$(pwd)/nginx.conf,dst=/usr/local/nginx/conf/nginx.conf --mount src=wwwroot,dst=/wwwroot nginx:v1
```

5、以wordpress博客为例

https://cn.wordpress.org/wordpress-4.9.4-zh_CN.tar.gz

第 8 章 企业级镜像仓库Harbor

1. Harbor概述
2. Harbor部署
3. 基本使用

Harbor概述

Habor是由VMWare公司开源的容器镜像仓库。事实上，Habor是在Docker Registry上进行了相应的企业级扩展，从而获得了更加广泛的应用，这些新的企业级特性包括：管理用户界面，基于角色的访问控制 ，AD/LDAP集成以及审计日志等，足以满足基本企业需求。

官方地址：https://vmware.github.io/harbor/cn/

组件	功能
harbor-adminserver	配置管理中心
harbor-db	Mysql数据库
harbor-jobservice	负责镜像复制
harbor-log	记录操作日志
harbor-ui	Web管理页面和API
nginx	前端代理，负责前端页面和镜像上传/下载转发
redis	会话
registry	镜像存储

Harbor部署

Harbor安装有3种方式：

- 在线安装：从Docker Hub下载Harbor相关镜像，因此安装软件包非常小
- 离线安装：安装包包含部署的相关镜像，因此安装包比较大
- OVA安装程序：当用户具有vCenter环境时，使用此安装程序，在部署OVA后启动Harbor

Harbor部署

```
# tar zxvf harbor-offline-installer-v1.5.1.tgz
# cd harbor
# vi harbor.cfg
hostname = 10.206.240.188
ui_url_protocol = http
harbor_admin_password = 123456
# ./prepare
# ./install.sh
```

基本使用

1、配置http镜像仓库可信任

vi /etc/docker/daemon.json

{"insecure-registries":["reg.ctnrs.com"]}

systemctl restart docker

2、打标签

docker tag centos:6 reg.ctnrs.com/library/centos:6

3、上传

docker push reg.ctnrs.com/library/centos:6

4、下载

docker pull reg.ctnrs.com/library/centos:6

谢谢