收藏这篇两万字总结，❤□Docker❤□这一块保证你拿捏的死死的，我说的，耶稣都不行

原创

一条coding 于 2021-08-29 15:26:08 发布 13657 收藏 429

分类专栏： 大厂面试突击 文章标签： docker 面试 java linux 网络

本文链接：https://blog.csdn.net/skylibiao/article/details/119980951

大厂面试突击 专栏收录该内容

8 篇文章 759 订阅

订阅专栏

□欢迎订阅《大厂面试突击》专栏，面试10多家大厂总结出的高频面试知识，免费阶段大家赶快订阅

□更多精品专栏简介点这里

□更多java面试学习资料，请私信「资料」获取

> 岁月无情，余生有涯，将生活扛在肩上，风雨兼程。

前言

哈喽，大家好，我是一条。

《大厂面试突击》专栏目前已发布三篇万字总结，收获`500+`的订阅，感谢各位的支持。

面试10多家中大厂后的万字总结——❤□集合篇❤□

面试10多家中大厂后的万字总结——❤□JavaWeb篇❤□

面试10多家中大厂后的万字总结——❤□java基础篇❤□

但想成一个优秀的程序员，「算法」+「八股文」只能让你冲过第一关，随着年限的增长，面试官会看重你技术的广度和深度，更加注重你的经验和解决问题的能力。

所以一条开了一个新的专栏《技术专家修炼》，内容如下：

- 企业实战的讲解
- 中间件微服务的介绍
- 工作中遇到的坑和总结

总之这是帮助你一步步封神的秘籍！

今天给大家带来`docker`的万字总结，虽说我们是开发，但`docker`不能不会，技术的广度就在这里提现。

文章目录

安装docker

> 鉴于同学们用的设备都不一样，不能让大家在第一步就被劝退，所以三个平台的安装方式都准备了，请自行选择。
> 不推荐在 `windows` 安装

mac

命令行安装

> 需要先安装homebrew

homebrew国内镜像

```
/bin/zsh -c "$(curl -fsSL https://gitee.com/cunkai/HomebrewCN/raw/master/Homebrew.sh)"
```

执行后选择中科大的镜像，即数字 `1`

`clone` 时间过长，约 `5-10` 分钟。

安装docker

```
brew install --cask --appdir=/Applications docker
```

`Installing cask docker` 时请耐心等待，时间较长

```
[redacted] @ libiaodeMacBook-Pro in ~ [13:17:28] C:130
$ [redacted]nstall --cask --appdir=/Applications docker
==> Downloading https://desktop.docker.com/mac/stable/amd64/66501/Docker.dmg
Already downloaded: /Users/libiao/Library/Caches/Homebrew/downloads/f4d2b5ad1d50
86f62ac3a743ae36dbcc6522b49413e8bd117a0e9cffe321117f--Docker.dmg
==> Installing Cask docker
==> Moving App 'Docker.app' to '/Applications/Docker.app'
==> Linking Binary 'docker-compose.bash-completion' to '/usr/local/etc/bash_comp
==> Linking Binary 'docker.zsh-completion' to '/usr/local/share/zsh/site-functic
==> Linking Binary 'docker.fish-completion' to '/usr/local/share/fish/vendor_com
==> Linking Binary 'docker-compose.fish-completion' to '/usr/local/share/fish/ve
==> Linking Binary 'docker-compose.zsh-completion' to '/usr/local/share/zsh/site
==> Linking Binary 'docker.bash-completion' to '/usr/local/etc/bash_completion.d
🍺  docker was successfully installed!
```

dmg安装

> 点击链接下载安装即可，并带有可视化界面。但个人觉得并不好用。

https://download.docker.com/mac/edge/Docker.dmg

启动docker服务

点击图标或者

```
open /Applications/Docker.app
```

windows

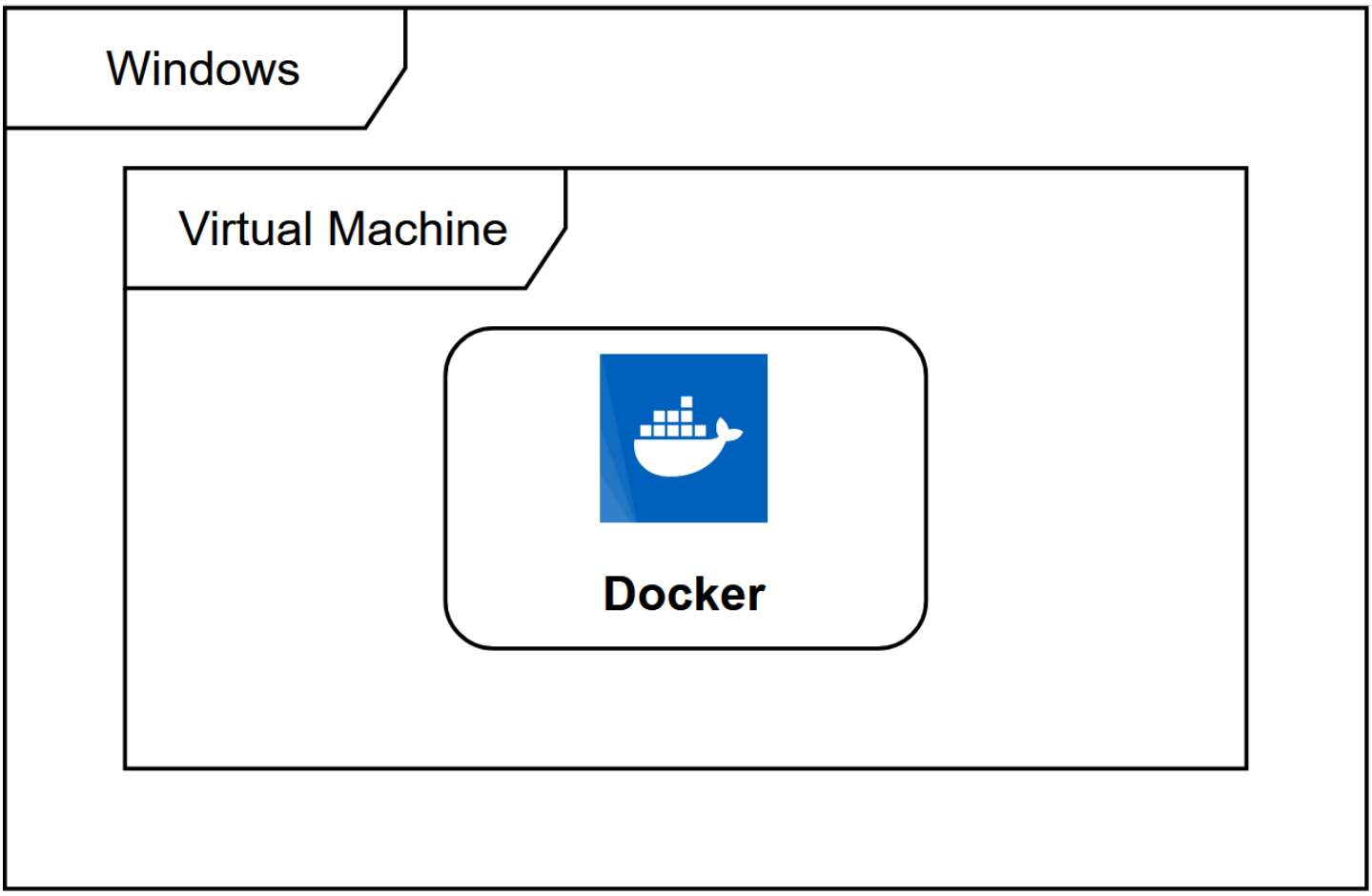
> 不推荐在windows安装，如果实在没有也可以装。

教程参考：https://www.runoob.com/docker/windows-docker-install.html

Docker 并非是一个通用的容器工具，它依赖于已存在并运行的 Linux 内核环境。

Docker 实质上是在已经运行的 Linux 下制造了一个隔离的文件环境，因此它执行的效率几乎等同于所部署的 Linux 主机。

因此，Docker 必须部署在 Linux 内核的系统上。如果其他系统想部署 Docker 就必须安装一个虚拟 Linux 环境。



在 Windows 上部署 Docker 的方法都是先安装一个虚拟机，并在安装 Linux 系统的的虚拟机中运行 Docker。

Docker Desktop 是 Docker 在 Windows 10 和 macOS 操作系统上的官方安装方式，这个方法依然属于先在虚拟机中安装 Linux 然后再安装 Docker 的方法。

Docker Desktop 官方下载地址： https://hub.docker.com/editions/community/docker-ce-desktop-windows

注意：此方法仅适用于 Windows 10 操作系统专业版、企业版、教育版和部分家庭版！

安装 Hyper-V

Hyper-V 是微软开发的虚拟机，类似于 VMWare 或 VirtualBox，仅适用于 Windows 10。这是 Docker Desktop for Windows 所使用的虚拟机。

但是，这个虚拟机一旦启用，QEMU、VirtualBox 或 VMWare Workstation 15 及以下版本将无法使用！如果你必须在电脑上使用其他虚拟机（例如开发 Android 应用必须使用的模拟器），请不要使用 Hyper-V！

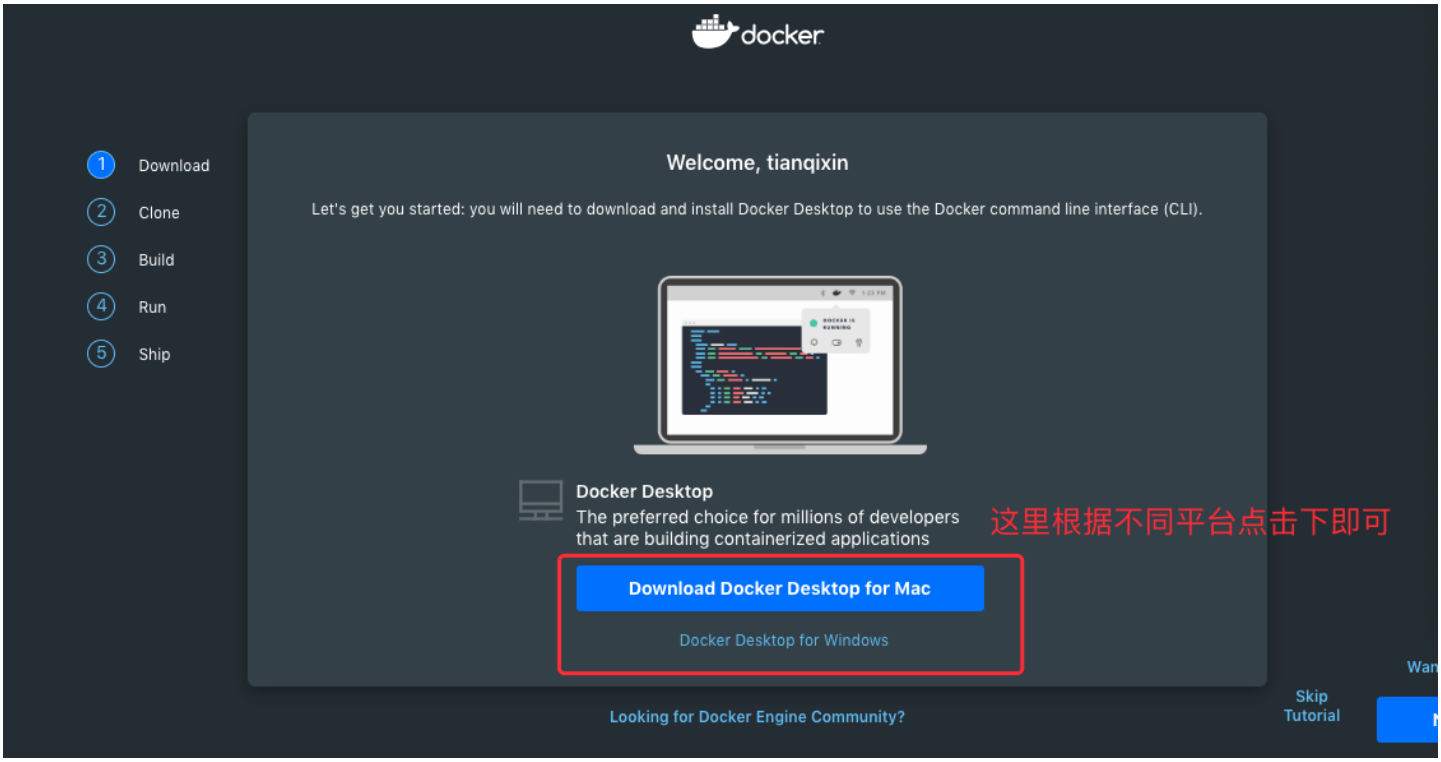
开启 Hyper-V

右键开始菜单并以管理员身份运行 PowerShell，执行以下命令：

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
```

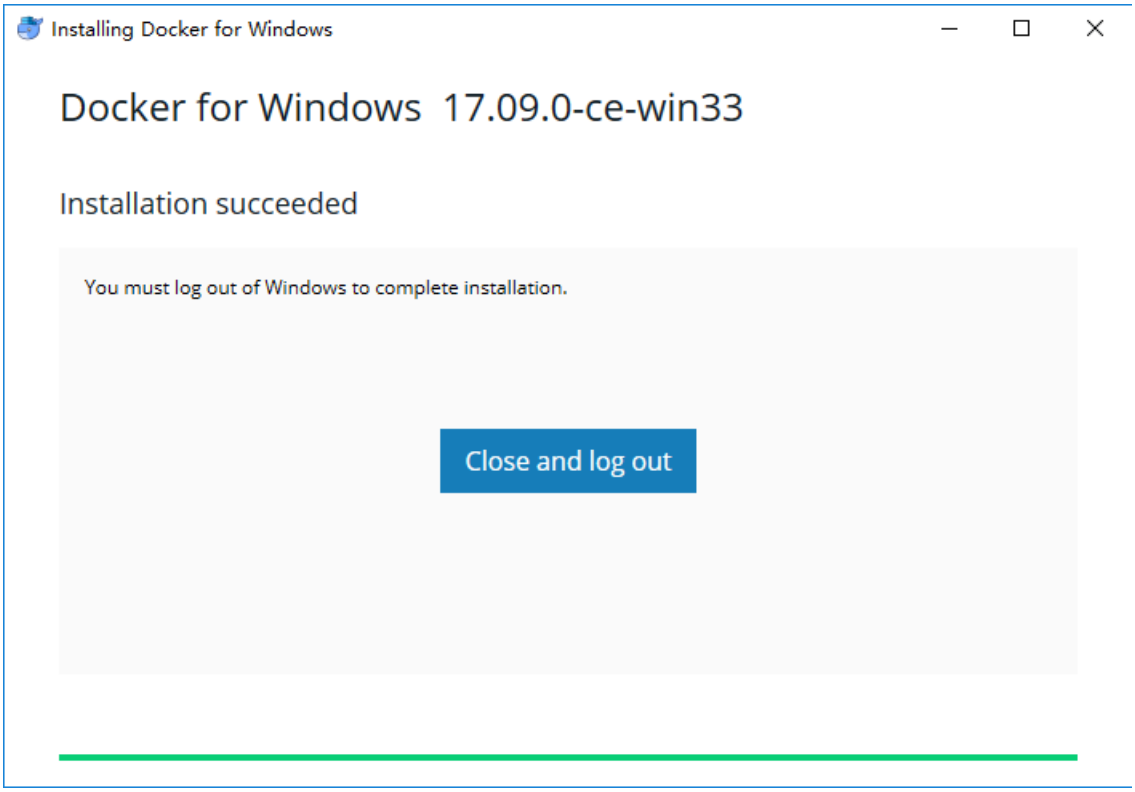
安装 Docker Desktop for Windows

点击 Get started with Docker Desktop，并下载 Windows 的版本，如果你还没有登录，会要求注册登录：



运行安装文件

双击下载的 Docker for Windows Installer 安装文件，一路 Next，点击 Finish 完成安装。



安装完成后，Docker 会自动启动。通知栏上会出现个小鲸鱼的图标 ，这表示 Docker 正在运行。

我们可以在命令行执行 `docker version` 来查看版本号。

阿里云（linux）

> 基于阿里云服务器的安装方式，推荐！

查看配置

```
# 内核版本查看
uname -r
```

系统版本：CentOS7

内核版本：3.10.0-514.26.2.el7.x86_64

安装

> 安装有两种方式:
> 1.官方脚本安装（本文讲解）
> 2.手动安装

安装：此为国内镜像。安装完提示如果想在非root用户使用，需将用命名加入组，并重启。

```
curl -fsSL https://get.docker.com/ | sh
```

□□如果报错缺少deltarpm，执行下面命令

```
yum provides '*/applydeltarpm'    #查看依赖包的位置
yum -y  install deltarpm             #安装命令
```

启动docker服务

```
service docker start
```

❤□本文以下全部讲解均基于 `Linux` 系统❤□

运行实例

> 本着一切语言都是从 `hello-world` 开始的原则，我们先运行官方的实例体验一下。

官方提供了hello-world实例。运行前需要在官网注册docker id并创建仓库。

官网地址：https://hub.docker.com

注册时注意id起的复杂一点，很容易重复。

启动docker服务

```
systemctl start docker
```

拉取镜像

```
docker pull hello-world
```

运行镜像

```
docker run hello-world
```

```
[root@lib ~]# docker run --name hello hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

查看容器

```
# -a 查看所有
docker ps -a
```

镜像加速

> 如果刚才拉取镜像时感觉速度过慢可以配置加速，速度正常可跳过此步。

鉴于国内网络问题，后续拉取 Docker 镜像十分缓慢，我们可以需要配置加速器来解决。

网易的镜像地址：**http://hub-mirror.c.163.com**。

在任务栏点击 `Docker for mac` 应用图标

`Perferences... -> Daemon -> Registry mirrors`

在列表中填写加速器地址即可。

修改完成之后，点击 `Apply & Restart` 按钮，Docker 就会重启并应用配置的镜像地址了。

通俗理解什么是docker?

Docker的思想来自于集装箱，集装箱解决了什么问题？

在一艘大船上，可以把货物规整的摆放起来。并且各种各样的货物被集装箱标准化了，集装箱和集装箱之间不会互相影响。那么我就不需要专门运送水果的船和专门运送化学品的船了。只要这些货物在集装箱里封装的好好的，那我就可以用一艘大船把他们都运走。

docker就是类似的理念。现在都流行云计算了，**云计算就好比大货轮。docker就是集装箱。**

不同的应用程序可能会有不同的应用环境，比如.net开发的网站和php开发的网站依赖的软件就不一样，如果把他们依赖的软件都安装在一个服务器上就要调试很久，而且很麻烦，还会造成一些冲突。比如IIS和Apache访问端口冲突。这个时候你就要隔离.net开发的网站和php开发的网站。常规来讲，我们可以在服务器上创建不同的虚拟机在不同的虚拟机上放置不同的应用，但是虚拟机开销比较高。docker可以实现虚拟机隔离应用环境的功能，并且开销比虚拟机小，小就意味着省钱了。

你开发软件的时候用的是Ubuntu，但是运维管理的都是centos，运维在把你的软件从开发环境转移到生产环境的时候就会遇到一些Ubuntu转centos的问题，比如：有个特殊版本的数据库，只有Ubuntu支持，centos不支持，在转移的过程当中运维就得想办法解决这样的问题。这时候要是有docker你就可以把开发环境直接封装转移给运维，运维直接部署你给他的docker就可以了。而且部署速度快。

在服务器负载方面，如果你单独开一个虚拟机，那么虚拟机会占用空闲内存的，docker部署的话，这些内存就会利用起来。

总之docker就是集装箱原理。

docker与虚拟机的对比

物理机：别墅

虚拟机：楼房

docker：酒店式公寓

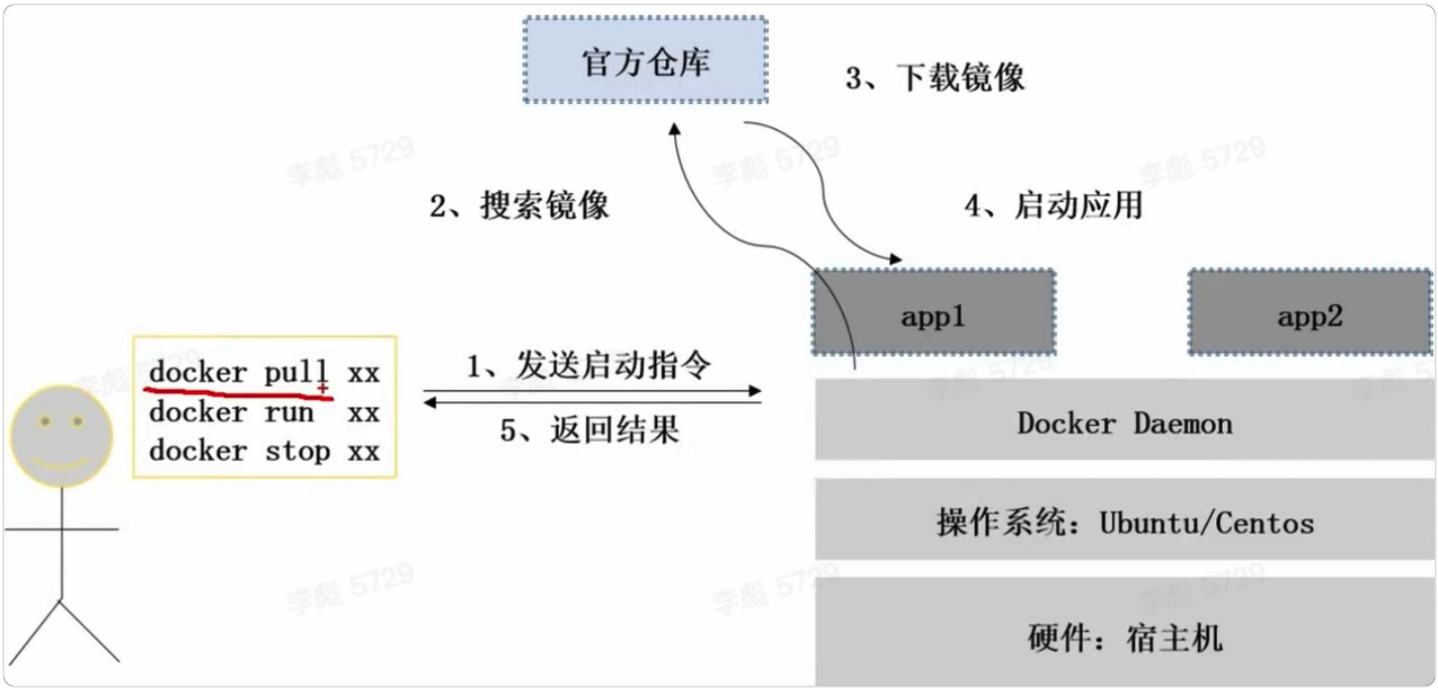
docker三大概念

库：一个总的仓库，包含所有的镜像，使用时可以从库拉取镜像到本地。

镜像：从库中拉取下来的应用，比如mysql。

容器：镜像运行之后就是容器，容器和镜像可以互相转换。

docker工作流程



docker命令

docker指令基本用法：

```
docker 命令关键字 -参数
```

基本操作

```
# 查看docker信息
docker info

# docker版本
docker version

# 查找镜像
docker search nginx

#拉取镜像
docker pull nginx
```

入门案例

快速搭建wordpress博客

查找镜像

```
docker search name
## wordpress
## mariadb
```

```
[root@lib ~]# docker search wordpress
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMAT
ED
wordpress                           The WordPress rich content management system…   4320      [OK]
appcontainers/wordpress             Centos/Debian Based Customizable Wordpress C…   34                   [OK]
aveltens/wordpress-backup           Easily backup and restore your WordPress blo…   23                   [OK]
conetix/wordpress-with-wp-cli       WordPress with wp-cli integration               17                   [OK]
arm32v7/wordpress                   The WordPress rich content management system…   15
```

拉取镜像

```
docker pull wordpress
# mariadb就是mysql
docker pull mariadb
```

运行镜像

```
docker run --name db -p 3306:3306 --env MYSQL_ROOT_PASSWORD=root -d mariadb
docker run --name mywordpress --link db:mysql -p 8080:80 -d wordpress
```

运行成功，访问wordpress

```
http://libiao:8080
```

根据提示配置数据库信息，一个个人博客网站就搭建好了

查看端口映射

```
docker ps

docker port CONTAINER_ID
```

举例：xxjob的8080端口映射到宿主机的8089端口

```
[root@data-pi-dev-db bin]# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED       STATUS       PORTS
                                  NAMES
9ef320797106   xuxueli/xxl-job-admin:2.3.0    "sh -c 'java -jar $J…"   12 days ago   Up 12 days   0.0.0.0:8089->8080/tcp
                                  my-xxl-job-2.3.0
aa36920a6e9b   nacos/nacos-server             "bin/docker-startup.…"   4 weeks ago   Up 4 weeks   0.0.0.0:8848->8848/tcp, 0.0.0.
                                  affectionate_lumiere
39929eac25f1   consul                         "docker-entrypoint.s…"   6 weeks ago   Up 6 weeks   8300-8302/tcp, 8600/tcp, 8301-
00/tcp, 0.0.0.0:8600->8600/udp    badger
711c90f2d3f8   kong:2.1.4                     "/docker-entrypoint.…"   7 weeks ago   Up 7 weeks   0.0.0.0:8000-8001->8000-8001/t
3-8444/tcp                        kong
8fcd92d6a65f   pantsel/konga                  "/app/start.sh"          7 weeks ago   Up 7 weeks   0.0.0.0:1337->1337/tcp
                                  konga
f9b60366854c   postgres:9.6                   "docker-entrypoint.s…"   7 weeks ago   Up 7 weeks   0.0.0.0:5432->5432/tcp
                                  postgres
5f3fdb8394af   redis                          "docker-entrypoint.s…"   7 weeks ago   Up 7 weeks   0.0.0.0:6379->6379/tcp
                                  myredis
d254a0cd1d31   yandex/clickhouse-server       "/entrypoint.sh"         7 weeks ago   Up 7 weeks   0.0.0.0:8123->8123/tcp, 0.0.0.
9090->9000/tcp                    clickhouse-server
cdf63ef766b4   mysql                          "docker-entrypoint.s…"   7 weeks ago   Up 7 weeks   33060/tcp, 0.0.0.0:13306->3306
                                  mysql
[root@data-pi-dev-db bin]# docker port 9ef320797106
8080/tcp -> 0.0.0.0:8089
```

linux设置docker开机自启

```
systemctl enable docker
```

查看镜像

```
docker images

#因为docker是分层，所以显示的文件大小要大于实际占用磁盘的大小
```

运行镜像

```
docker run --name db -p 3306:3306 --env MYSQL_ROOT_PASSWORD=root -d mariadb

# --name 别名

# --env 环境变量

#-d 后台执行

docker run --name mywordpress --link db:mysql -p 8080:80 -d wordpress

# --link ip映射

# -p 端口映射

docker logs -f 7a38a1ad55c6

# 像tail -f一样查看容器内日志

docker top name

#查看容器内的进程
```

删除镜像

```
docker rmi hello-world:latest

docker rmi id

# 4位即可
```

复合命令

```
docker rm -f $(docker ps -a -q)

#删除全部容器
```

查看运行的容器

```
docker ps
```

docker compose

一个方便维护多个容器的yaml文件，docker认为一个容器对应一个进程，但一个应用会有多个进程，例如上面的mysql和wordpress。

个人觉得docker compose类似于shell脚本，但他实际都python实现，访问的是docker的一些api。

Docker compose一般随docker一起安装，所以要求版本对应

```
docker version

docker-compose --version
```

在yaml文件制定镜像的名字，版本，端口映射后用 up -d 启动

```
docker-compose.yaml up -d
```

查看日志

```
docker-compose logs
```

容器管理

进入容器

```
docker exec -it name /bin/sh
```

查看容器详细信息

容器的详细信息会以json的形式返回。

```
# docker inspect name
[root@lib mysh]# docker inspect mywordpress

[
    {
       "Id": "6253e66959047c6f8de891abe1c661f7766fdef7407f00e07d1788310e0ea6a9",

        "Created": "2021-08-04T20:11:43.649001354Z",

        "Path": "docker-entrypoint.sh",

        "Args": [

            "apache2-foreground"

        ],

        "State": {

            "Status": "running",

            "Running": true,

            "Paused": false,

            "Restarting": false,

            "OOMKilled": false,

            "Dead": false,

            "Pid": 28041,

            "ExitCode": 0,

            "Error": "",

            "StartedAt": "2021-08-04T20:11:43.947511209Z",

            "FinishedAt": "0001-01-01T00:00:00Z"

        }

        "Name": "/mywordpress",

......
```

容器启停

```
docker start name

docker stop name

docker restart name
```

删除容器

```
# 删除时容器需要处于停止状态

docker rm name
```

查看日志

```
# docker ps -a
docker logs container_Id
```

占用资源

```
docker stats name
```

```
[root@lib ~]# docker stats
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
5a93ef4c22d5   nginx         0.00%     1.422MiB / 1.796GiB   0.08%     124kB / 0B        0B / 0B           2
6253e6695904   mywordpress   0.00%     205.7MiB / 1.796GiB   11.18%    103MB / 67.8MB    596MB / 9.21MB    11
64c1c842c038   db            0.52%     257.7MiB / 1.796GiB   14.01%    4.05GB / 5.67GB   668MB / 7.86GB    213
```

镜像特性

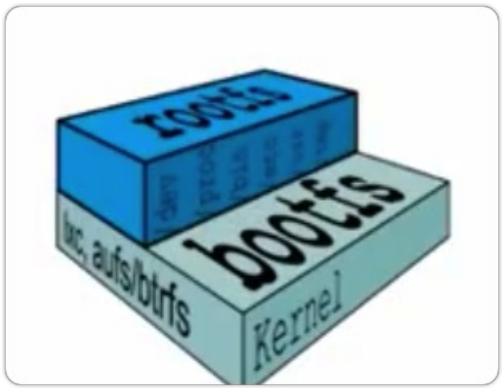
> 镜像是一种轻量级、可执行的独立软件包，用来打包软件运行环境和基于运行环境开发的软件，他包含运行某个软件所需的所有内容，包括**代码、运行时库、环境变量和配置文件**。将所有的应用和环境直接打包为docker镜像，就可以直接运行。
> 镜像名和版本号共同组成唯一标识，默认是最新版——lastest

分层原理

> Docker的镜像通过联合文件系统将将各层文件系统叠加在一起。

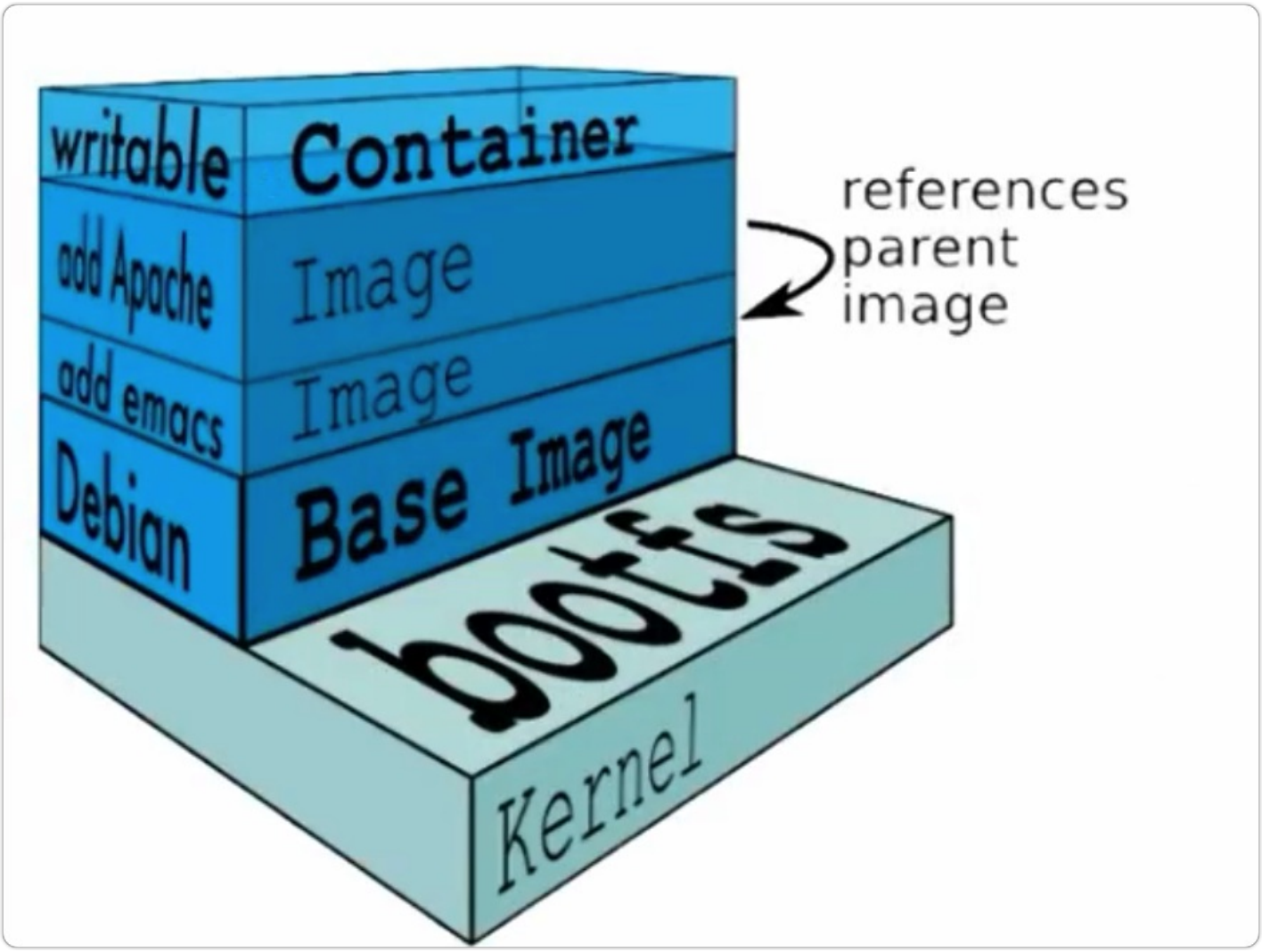
引导方式

- bootfs：传统操作系统，用于系统引导的文件系统，包括BootLoader和kernel，容器启动完成后会被卸载以节省内存资源。



- rootfs：位于bootfs之上，表现为docker容器的根文件系统
  - 传统模式中，系统启动时，内核首先挂载为”只读“模式，完成全部自检后挂载为”读写“模式。
  - docker中，rootfs由内核挂载为”只读“模式，而后通过UFS技术挂载一个”可写“层。

□□注意：已有的分层只能读不能写，上层镜像优先级大于底层镜像



当我们使用pull命令时，我们可以看到docker的镜像是一层一层的在下载。这样做最大的好处就是资源共享了。

比如多个镜像都从Base镜像构建而来，那么宿主机只需要在磁盘上保留一份base镜像，同时内存中也只需要加载一份base镜像，这样就可以为所有的容器服务了，而且镜像的每一层都可以被共享。查看镜像分层的方式可以通过docker image inspect命令。

所有的Docker镜像都起始于一个基础镜像，当进行修改或者增加新的内容时，就会在当前的镜像层之上，创建新的镜像层。在添加额外的镜像层的同时，镜像始终保持当前所有镜像的组合，
Docker通过存储引擎的方式来实现镜像层堆栈，并保证多镜像层对外展示为统一的文件系统。

UFS（联合文件系统）

UFS时一种分层、轻量级并且高性能的文件系统。

它支持对文件系统的修改作为一次提交来一层层的叠加，同时可以将不同目录挂载到同一个虚拟文件系统下。

UnionFS是Docker镜像的基础。镜像可以通过分层来进行继承，基于基础镜像，可以制作各种具体的应用镜像。一次同时加载多个文件系统，但从外面看起来，只能看到一个文件系统，联合加载会把各层文件，系统叠加起来，这样最终的文件系统会包含所有底层的文件和目录。

加载原理

Linux刚启动时会加载bootfs文件系统，在Docker镜像的最底层时bootfs。

当boot加载完成后整个内核就在内存中了，此时内存的使用权已由bootfs转交给内核，此时系统也会卸载bootfs。rootfs在bootfs之上，rootfs包含的就是典型Linux系统中的`/dev、/proc、/bin、/etc`等目录和文件。rootfs就是各种不同的操作系统发行版。

Docker File

> 仓库没有的镜像怎么办？
> 可以自己创建镜像吗？

容器->镜像

```
docker commit CID -t xx.xx.xx
```

□□□□□□□□ 工作在前台的守护进程至少一个

网易蜂巢：开源镜像仓库

```
[root@localhost ~]# docker commit mysql mysql:5.1
sha256:768b332d1bb1ee6aec44bd2c1f468af469a04d5176f92e8346f2f1a1b8d6b959
[root@localhost ~]# docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
mysql                         5.1       768b332d1bb1   4 seconds ago    748 MB
<none>                        <none>    f15c7c5a4b36   2 minutes ago    748 MB
wordpress                     latest    c96daeeff307   4 days ago       421 MB
mariadb                       latest    97df12fa9319   8 days ago       368 MB
mysql                         5.7       98455b9624a9   2 weeks ago      372 MB
nginx                         latest    2bcb04bdb83f   2 weeks ago      109 MB
centos                        6.8       82f3b5f3c58f   3 weeks ago      195 MB
hub.c.163.com/public/centos   6.7-tools b2ab0ed558bb   2 years ago      602 MB
```

编写docker file

> Dockerfile 是一个用来构建镜像的文本文件，文本内容包含了一条条构建镜像所需的指令和说明。

指令讲解

FROM

指定基础镜像，必须为第一个命令，有且只有一个

```
# FROM <image>
# FROM <image>:<tag>
# FROM <image>@<digest>
FROM mysql:5.6
```

MAINTAINER

创建者信息

```
# MAINTAINER <name>
MAINTAINER yitiao
```

RUN

用于在镜像容器中执行命令，其有以下两种命令执行方式：

```
#shell执行
#    RUN <command>
#exec执行
#    RUN ["executable", "param1", "param2"]
RUN apk update
RUN ["/etc/execfile", "arg1", "arg1"]
```

ADD

将本地文件添加到容器中，tar类型文件会自动解压(网络压缩资源不会被解压)，可以访问网络资源，类似wget

```
# ADD <src>... <dest>
ADD hom?.txt /mydir/      # ? 替代一个单字符,例如：“home.txt”
```

COPY

功能类似ADD，但是是不会自动解压文件，也不能访问网络资源

CMD

构建容器后调用，也就是在容器启动时才进行调用。

```
# CMD command param1 param2 (执行shell内部命令)
CMD echo "This is a test." | wc -
#CMD不同于RUN，CMD用于指定在容器启动时所要执行的命令，而RUN用于指定镜像构建时所要执行的命令
```

ENTRYPOINT

配置容器，使其可执行化。配合CMD可省去"application"，只使用参数。

```
# ENTRYPOINT ["executable", "param1", "param2"] (可执行文件, 优先)
# ENTRYPOINT command param1 param2 (shell内部命令)FROM ubuntu
ENTRYPOINT ["top", "-b"]
CMD ["-c"]
```

LABEL

用于为镜像添加元数据

```
# LABEL <key>=<value> <key>=<value> <key>=<value> ...
LABEL version="1.0" description="一条coding" by="一条"
```

ENV

设置环境变量

```
# ENV <key> <value>
# <key>之后的所有内容均会被视为其<value>的组成部分，因此，一次只能设置一个变量
# ENV <key>=<value> ...
# 可以设置多个变量，每个变量为一个"<key>=<value>"的键值对
ENV myName John Doe
ENV myDog Rex The Dog
ENV myCat=fluffy
```

EXPOSE

指定于外界交互的端口

```
格式：
    EXPOSE <port> [<port>...]
示例：
    EXPOSE 80 443
    EXPOSE 8080
    EXPOSE 11211/tcp 11211/udp
注：
   EXPOSE并不会让容器的端口访问到主机。要使其可访问，需要在docker run运行容器时通过-p来发布这些端口，或通过-P参数来发布EXPOSE导出的所有端口
```

VOLUME

用于指定持久化目录

```
格式：
    VOLUME ["/path/to/dir"]
示例：
    VOLUME ["/data"]
    VOLUME ["/var/www", "/var/log/apache2", "/etc/apache2"
注：
   一个卷可以存在于一个或多个容器的指定目录，该目录可以绕过联合文件系统，并具有以下功能：
1 卷可以容器间共享和重用
2 容器并不一定要和其它容器共享卷
3 修改卷后会立即生效
4 对卷的修改不会对镜像产生影响
5 卷会一直存在，直到没有任何容器在使用它
```

WORKDIR

工作目录，类似于cd命令

```
# WORKDIR /path/to/workdir
WORKDIR /a  #(这时工作目录为/a)
WORKDIR b  #(这时工作目录为/a/b)
WORKDIR c  #(这时工作目录为/a/b/c)
#通过WORKDIR设置工作目录后，Dockerfile中其后的命令RUN、CMD、ENTRYPOINT、ADD、COPY等命令都会在#该目录下执行。在使用docker run运行容器时，可以通过-w参数覆盖构建时所设置的工作目录。
```

USER

指定运行容器时的用户名或 UID，后续的 RUN 也会使用指定用户。使用USER指定用户时，可以使用用户名、UID或GID，或是两者的组合。当服务不需要管理员权限时，可以通过该命令指定运行用户。并且可以在之前创建所需要的用户。

使用USER指定用户后，Dockerfile中其后的命令RUN、CMD、ENTRYPOINT都将使用该用户。镜像构建完成后，通过`docker run`运行容器时，可以通过-u参数来覆盖所指定的用户。

```
# USER user
# USER user:group
# USER uid
# USER uid:gid
USER www
```

ARG

用于指定传递给构建运行时的变量

```
# ARG <name>[=<default value>]
ARG site
ARG build_user=www
```

ONBUILD器

用于设置镜像触发

```
# ONBUILD [INSTRUCTION]
ONBUILD ADD . /app/src
ONBUILD RUN /usr/local/bin/python-build --dir /app/src
#当所构建的镜像被用做其它镜像的基础镜像，该镜像中的触发器将会被钥触发
```

一图理解，yyds

图片源于网络

docker file demo

```
# 一条coding
# Version 1.0

# Base images 基础镜像
FROM centos

#MAINTAINER 维护者信息
MAINTAINER tianfeiyu

#ENV 设置环境变量
ENV PATH /usr/local/nginx/sbin:$PATH

#ADD  文件放在当前目录下，拷过去会自动解压
ADD nginx-1.8.0.tar.gz /usr/local/
ADD epel-release-latest-7.noarch.rpm /usr/local/

#RUN 执行以下命令
RUN rpm -ivh /usr/local/epel-release-latest-7.noarch.rpm
RUN yum install -y wget lftp gcc gcc-c++ make openssl-devel pcre-devel pcre && yum clean all
RUN useradd -s /sbin/nologin -M www

#WORKDIR 相当于cd
WORKDIR /usr/local/nginx-1.8.0

RUN ./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_ssl_module --with-pcre && make && m
ake install

RUN echo "daemon off;" >> /etc/nginx.conf

#EXPOSE 映射端口
EXPOSE 80

#CMD 运行以下命令
CMD ["nginx"]
```

网络通信

> docker是如何与内部和外部进行数据交换的？
>
> - 容器内部
> - 内部访问外部
> - 外部访问内部

网络知识补充

eth0

eth0 物理网卡是指服务器上实际的网络接口设备。设备用于接收以太网数据接口，数据包在各个节点中转发和路由。

veth

veth 顾名思义，veth-pair 是一对的虚拟设备接口，它都是成对出现的。

一端连着协议栈，一端彼此相连着。一个设备从协议栈读取数据后，会将数据发送到另一个设备上去。

正因为有这个特性，它常常充当着一个桥梁，连接着各种虚拟网络设备，典型的例子像“两个 namespace 之间的连接”，“Bridge、OVS 之间的连接”，“Docker 容器之间的连接” 等等，以此构建出非常复杂的虚拟网络结构，比如 OpenStack Neutron。

bridge

Bridge 设备是一种纯软件实现的虚拟交换机，可以实现交换机的二层转发。与现实世界中的交换机功能相似。

与其他虚拟网络设备一样，可以配置 IP、MAC。Bridge 的主要功能是在多个接入 Bridge 的网络接口间转发数据包。

网络模型

我们在使用docker run创建Docker容器时，可以用`--net`选项指定容器的网络模式，Docker有以下4种网络模式：

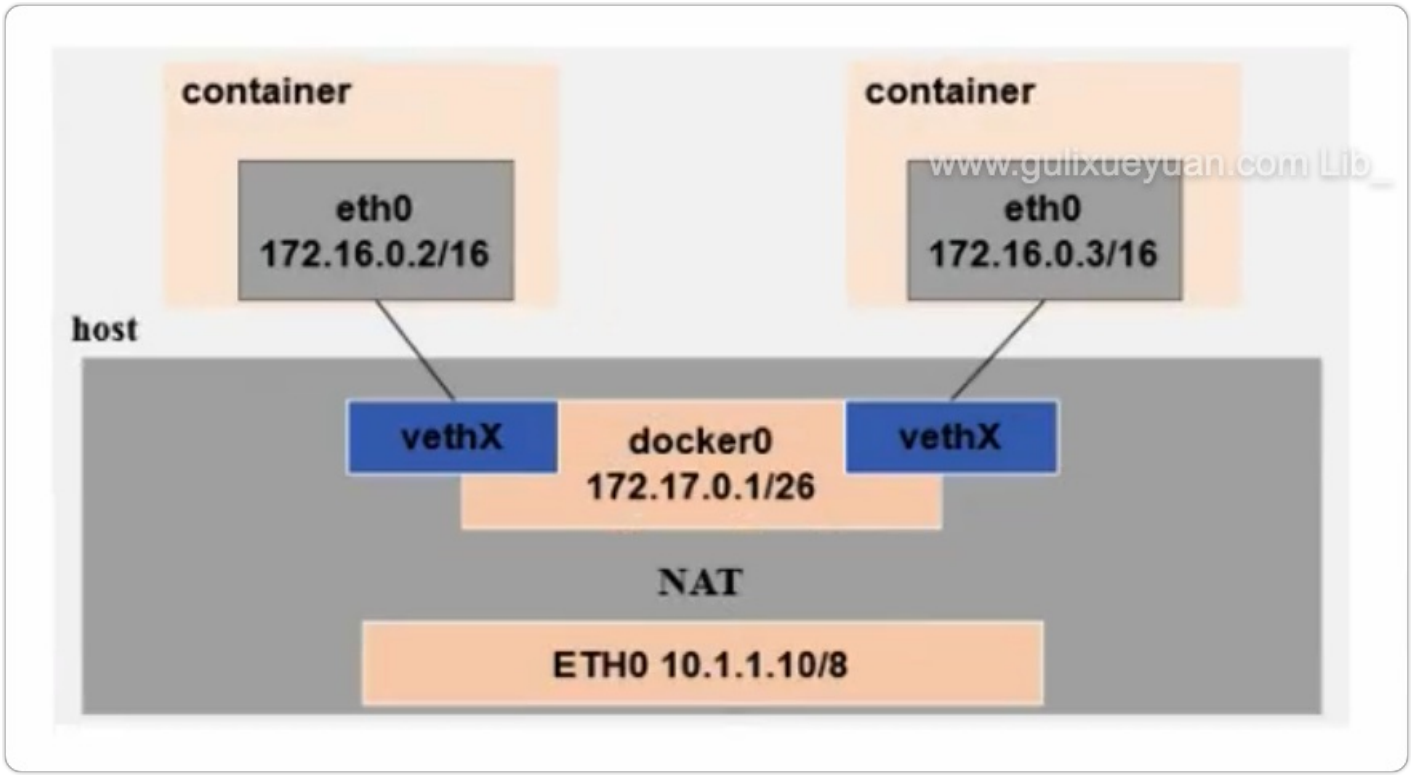
host模式，使用–net=host指定。

container模式，使用–net=container:NAME_or_ID指定。

none模式，使用–net=none指定。

bridge模式，使用–net=bridge指定，默认设置。

除这四种基本的之外，还支持各种自定义模型。



容器内部访问

通常情况下，docker使用网桥+NAT的方式进行通信。Bridge 模式会为容器创建独立的网络 namespace ，拥有独立的网卡等网格栈。

NAT：可以理解为网卡

Dcoker0:就是网桥，交换机，`ifconfig`可见

```
[root@lib ~]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:03:76:2a:fe  txqueuelen 0  (Ethernet)
        RX packets 17217729  bytes 5567914596 (5.1 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 26789289  bytes 4207163328 (3.9 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

同一宿主机上，Bridge 模式创建的容器会通过 DHCP 链接到 docker0 上，通过 docker0 实现网络的互通。「容器之间都是连接掉docker0这个网桥上的，它作为虚拟交换机使容器可以相互通信」。

内外部通信

宿主机的 IP 地址与容器 veth pair 的 IP 地址不在同一个网段，宿主机外的网络无法主动发现容器的存在，不能直接进行容器通信。所以 Docker 提供了端口映射的方式，就是将宿主机上的端口流量映射转发到容器内的端口上。

---

ok，至此docker的全部知识总结完成，作为java开发，掌握这些足以让你如鱼得水。

❤□可以三连收藏啦！！❤□

最后

> □今天是坚持刷题更文的第**45**/100天
> □各位的点赞、关注、收藏、评论、订阅就是一条创作的最大动力
> □更多干货欢迎订阅专栏《技术专家修炼》

为了回馈各位粉丝，礼尚往来，给大家准备了一条多年积累下来的优质资源，包括 学习视频、面试资料、珍藏电子书等

需要的小伙伴请私信「资料」，记得先关注哦！