# 容器docker的简介

## ****1.什么是Docker****

Docker属于Linux容器的一种封装，提供简单易用的容器使用接口，它也是目前最流行的Linux容器解决方案。Docker 将软件代码和其依赖，全打包在一个文件中。运行单个文件，就会生成虚拟容器。在这个虚拟容器中，不管本地的操作系统是如何的不同，此容器都能照常运行。

简而言之，Docker的接口非常简单，可以帮助用户更好地创建和使用容器，让相同的代码在不同的环境上正常运行。

## ****2.虚拟机(Virtual Machine)****

1) 占用资源多：

虚拟机需要安装整个操作系统，自然会消耗大量内存和硬盘空间。如我们只需要运行1MB的软件，有时候也不得不安装几个G的环境才能运行。

2) 运行步骤冗余：

虚拟机安装的是完整的系统，每次运行程序都需要按部就班，打开系统、登入用户等等之类麻烦的步骤，很不方便。

3) 运行速度慢：

为了运行特定环境中的软件，虚拟机必须先运行系统，而系统占用的资源往往很多(网络，GUI，IO等等)，自然也会影响运行速度。

## **3.**Linux容器(Container)****

为了解决虚拟机存在的这些缺点，Linux发展出了另一种虚拟化的技术：Linux容器。Linux 容器不是模拟一个完整的操作系统，而是对进程进行隔离。或者说，就是在正常进程的外面套用了一个保护层。对于容器里面的进程来说，它接触到的各种资源都是虚拟的，从而实现与底层程序的隔离。由于容器是进程级别的，相比虚拟机有更多优势：

1) 占有资源少：

容器只占用需要的资源，不占用那些用不到的资源。相比于虚拟机安装完整的操作系统，容器需要消耗的空间自然就少了很多。

2) 资源利用率高：

虚拟机都是独享资源，电脑需要为每个虚拟环境单独分配资源，不仅仅占用空间大，而且资源的利用率很低。而容器之间可以共享资源，最大化资源的利用率。

3) 运行速度快：

容器里面的应用就是底层系统的一个进程，所以启动容器相当于直接运行本机的一个进程，而不是一个完整并臃肿的操作系统，自然就快很多。

## ****4.Docker的用途****

Docker目前主要有以下三个用途：

1) 提供一次性的环境：本地测试别人的软件、持续集成的时候提供单元测试和构建的环境。

2) 提供弹性的云服务：因为Docker容器可以随时启动或关闭，所以非常适合动态规划和缩容。

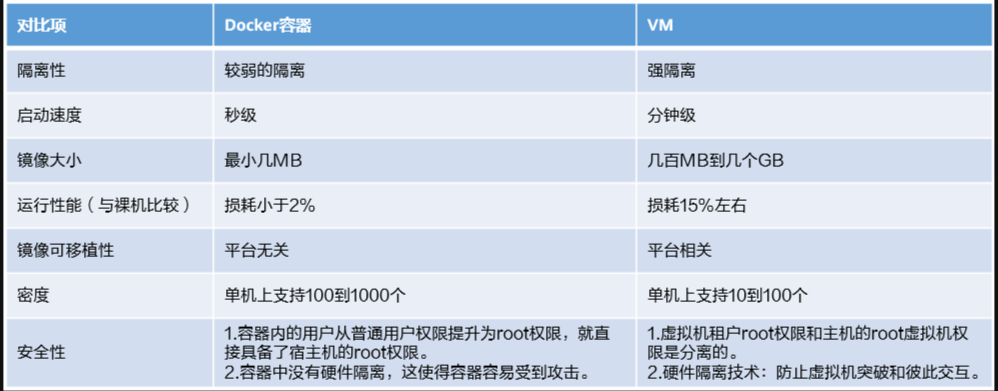
3) 组建微服务构架：通过多个容器，服务的部署能更加灵活，帮助实现微服务构架。

4) 不需要虚拟硬件和操作系统，轻量级，占用体积小，启动快。

## ****5.Docker容器与VM的区别****

VM：使用Hypervisor提供虚拟机的运行平台，管理每个VM中操作系统的运行。每个VM都要有自己的操作系统、应用程序和必要的依赖文件等。

Docker容器：使用Docker引擎进行调度和隔离，提高了资源利用率，在相同硬件能力下可以运行更多的容器实例；每个容器拥有自己的隔离化用户空间。



# 二、****Docker容器的工作流程****

## 1.Docker的三大组成要素

### 1) 镜像

Docker镜像是一个特殊的文件系统，除了提供容器运行时所需的程序、库、资源、配置等文件外，还包含了一些为运行时准备的配置参数。 镜像不包含任何动态数据，其内容在构建之后也不会被改变。镜像可以用来创建Docker容器，用户可以使用设备上已有的镜像来安装多个相同的Docker容器。

### 2) 容器

镜像创建的运行实例，Docker利用容器来运行应用。每个容器都是相互隔离的、保证安全的平台。我们可以把容器看做是一个轻量级的Linux运行环境。

### 3) 镜像仓库

集中存放镜像文件的地方。用户创建完镜像后，可以将其上传到公共仓库或者私有仓库，需要在另一台主机上使用该镜像时，只需要从仓库上下载即可。

## 2.Docker的运行流程

### 1) Docker客户端

用于和Docker守护进程(Docker Daemon)建立通信的客户端。Docker客户端只需要向Docker服务器或者守护进程发出请求(Docker构建、Docker拉取和Docker启动等指令)，服务器或者守护进程将完成所有工作并返回结果。

### 2) Docker主机

一个物理或者虚拟的机器用于执行 Docker守护进程和容器。

### 3) Docker守护进程

接收并处理Docker客户端发送的请求，监测Docker API的请求和管理Docker对象，比如镜像、容器、网络和数据卷。

## 3.Docker的组件

### 1) Docker服务器与客户端

Docker是一个客户端-服务器(C/S)架构程序。Docker客户端只需要向Docker服务器或者守护进程发出请求，服务器或者守护进程将完成所有工作并返回结果。Docker提供了一个命令行工具Docker以及一整套RESTful API。你可以在同一台宿主机上运行Docker守护进程和客户端，也可以从本地的Docker客户端连接到运行在另一台宿主机上的远程Docker守护进程。

### 2) Docker镜像与容器

镜像是构建Docker的基石。用户基于镜像来运行自己的容器。镜像也是Docker生命周期中的“构建”部分。镜像是基于联合文件系统的一种层式结构，由一系列指令一步一步构建出来。

### 3) Registry(注册中心)

Docker用Registry来保存用户构建的镜像。Registry分为公共和私有两种。Docker公司运营公共的Registry叫做Docker Hub。用户可以在Docker Hub注册账号，分享并保存自己的镜像(说明：在Docker Hub下载镜像巨慢，可以自己构建私有的Registry)。

# 三、Docker安装与启动

## 1.系统：Ubuntu

安装步骤如下：

### 1) 卸载(前提是安装过docker)

sudo apt-get remove docker docker-engine docker.io containerd runc

### 2) 安装必要工具

第一步：sudo apt-get update

第二步：sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

### 3) 安装GPG证书

**Docker官方：**

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

**阿里云：**

curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -

**Docker官方：**

sudo add-apt-repository “deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable”

**阿里云：**

sudo add-apt-repository “deb [arch=amd64] http://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb\_release -cs) stable”

**注意：**一般以执行阿里云为主

### 4) 更新并安装docker-ce

sudo apt-get -y install docker-ce

### 5) 开启docker服务

systemctl status docker

### 6) 查看docker版本

docker version

**注意：**安装完成后，如果出现权限禁止，需要切换到root用户执行

## 2.系统：****centos****

### 1) yum包更新到最新

sudo yum update

### 2) 安装需要的软件包

yum-util 提供yum-config-manager功能，另外两个是devicemapper驱动依赖的

sudo yum install -y yum-utils device-mapper-persistent-data lvm2

### 3) 设置yum源为阿里云

sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

### 4) 安装docker

sudo yum install docker-ce

### 5) 安装后查看docker版本

docker -v

### 6) 启动或重启docker服务

systemctl start docker

systemctl stop docker

systemctl restart docker

### 7) 开机启动

systemctl enable docker

### 8) 查看概要信息

docker info

## 3.系统：****windows****

### 1) 安装WSL2

WSL，即Windows Subsystem on Linux，中文叫“适用于 Linux 的 Windows 子系统”。可以在windows里，直接启动一个linux系统。因为docker依赖linux内核，只能在linux下使用，windows就需要安装linux虚拟机来运行，而微软已经在win10内置了一个轻量级虚拟机，WSL2 便是运行在虚拟机上的一个完整的 linux 内核，所以需要利用WSL2安装docker。

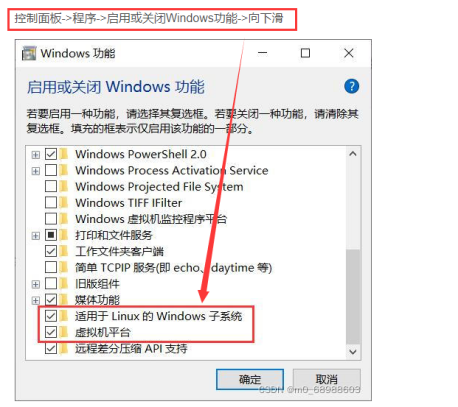
**第一步：**微软要求Win10版本2004及更高版本(内部版本19041及更高版本)。



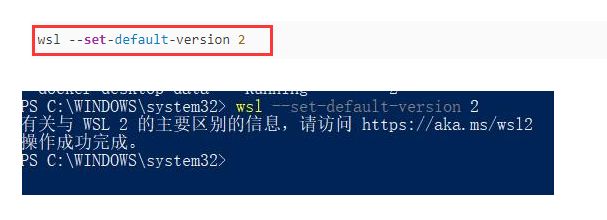
**第二步：**下载wsl2需要的linux内核,点击下载最新的更新包，下好后双击运行安装即可。(<https://docs.microsoft.com/zh-cn/windows/wsl/wsl2-kernel>)



**第三步：**打开系统虚拟机平台和WSL

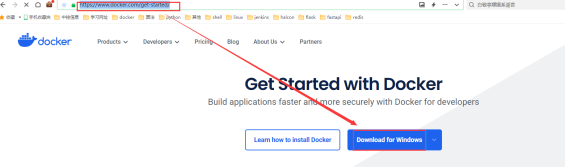


**第四步：**将WSL2设置为默认版本，以管理员身份运行命令行。

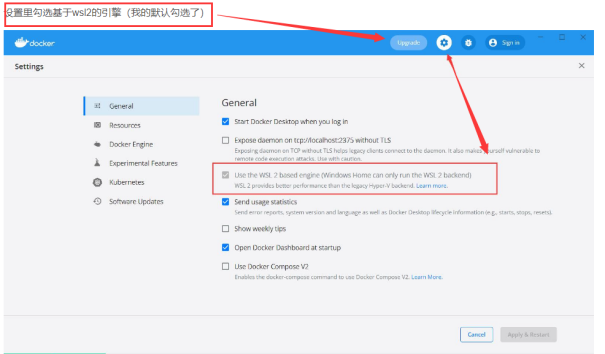


### 2) 进入官网，下载docker

下载windows版本的docker(https://www.docker.com/get-started/)；

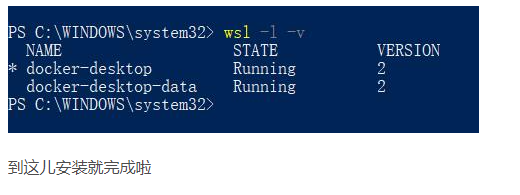


### 3) 安装docker(安装过程省略)



### 4) 检查安装情况

输入命令：wsl -l -v



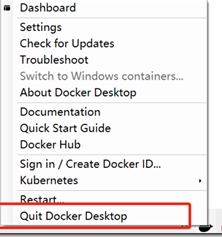
### 5) 更改镜像源

在国内从官方的DockerHub拉取镜像经常会比较慢，所以我们需要利用国内的镜像源或者加速器。



### 6) 更改数据存储位置

第一步：停止docker服务。



第二步：备份数据到另一个文件夹。

wsl --export docker-desktop-data "E:\DockerData\docker-desktop-data.tar"

第三步：删除原有数据。

wsl --unregister docker-desktop-data

第四步：更改数据存储盘并恢复数据。

wsl --import docker-desktop-data "E:\DockerData" "E:\DockerData\docker-desktop-data.tar" --version 2



# 四、常用命令(以下命令针对centos系统)

## 1．镜像常用命令

### 1) 创建镜像

**方式一：**

使用Dockerfile构建镜像：

docker build -t <image\_name> .

其中，<image\_name>是自定义的镜像名称，.表示当前目录。

例子：

docker build -t my\_image .

**方式二：**

使用docker run命令创建临时镜像：

docker run --name <container\_name> <image\_name> /bin/bash

其中，<container\_name>是自定义的容器名称，<image\_name>是要运行的镜像名称。

例子：

docker run --name my\_container my\_image /bin/bash

**方式三：**

docker commit -m "描述信息" -a "作者信息" 容器id 镜像名称:版本

#常用选项：

-m 说明信息；

-a 作者信息；

例子：

docker commit -m "new-images" -a "test" dda50e36fd55 centos:test

**方式四：**

使用docker import命令从tar文件导入镜像：

docker import <tar\_file\_path> <new\_image\_name>

其中，<tar\_file\_path>是包含镜像内容的tar文件路径，<new\_image\_name>是自定义的新镜像名称。

例子：

docker import my\_tar\_file.tar new\_image

### 2) 查看镜像

docker images

docker images –q

命令返回结果：

REPOSITORY：镜像名称

TAG：镜像标签

IMAGE ID：镜像ID

CREATED：镜像的创建日期(不是获取该镜像的日期)

SIZE：镜像大小

### 3) 搜索镜像

docker search 镜像名称

docker search [OPTIONS] TERM

命令返回结果：

NAME：仓库名称

DESCRIPTION：镜像描述

STARS：用户评价，反应一个镜像的受欢迎程度

OFFICIAL：是否官方

AUTOMATED：自动构建，表示该镜像由Docker Hub自动构建流程创建的

# 命令参数(OPTIONS)：

-f, --filter filter 根据提供的格式筛选结果

--format string 利用Go语言的format格式化输出结果

--limit int 展示最大的结果数，默认25个

--no-trunc 内容全部显示

实例：

docker search -f is-offical=true ubuntu

docker search ubuntu

### 4) 拉取镜像

本地拉取：

docker pull 镜像名称

例子：docker pull centos:7

远程拉取：

docker login --username=wdd930817 registry.cn-hangzhou.aliyuncs.com

docker pull registry.cn-hangzhou.aliyuncs.com/wdd\_project/wdd\_docker:[镜像版本号]

### 5) 推送镜像

本地推送：

docker push [镜像名]:[标签]

远程推送：

docker login --username=wdd930817 registry.cn-hangzhou.aliyuncs.com

docker tag ImageId registry.cn-hangzhou.aliyuncs.com/wdd\_project/wdd\_docker:镜像版本号

docker push registry.cn-hangzhou.aliyuncs.com/wdd\_project/wdd\_docker:[镜像版本号]

### 6) 删除镜像

按镜像ID删除镜像：

docker rmi 镜像ID

docker rmi `docker images -q` ### 删除所有本地镜像

删除所有镜像：

docker rmi `docker images -q`

容器保存为镜像：

docker commit 容器ID 镜像名称:版本

例子：docker commit mynginx mynginx\_i

镜像恢复与迁移：

docker load -i mynginx.tar

### 7) 镜像存放位置

如果C盘空间不足，可以按照以下步骤将Docker镜像目录变更到其他盘上：

(1)打开Docker Desktop。

(2)点击界面右上方的齿轮图标（Settings）。

(3)点击界面左侧菜单：Resources -> ADVANCED。

(4)滚动右侧内容到最底部，出现一项：Disk image location，即为Docker Desktop镜像存储位置。

(5)点击"Browser"，选择新的镜像目录。

(6)改变镜像目录后，会提示重启Docker Desktop，确定即可。

## 2．容器docker常用命令

### 1) 创建容器

**方式一：**

**交互式方式创建容器：**

docker run -it --name=容器名称 镜像名称:标签 /bin/bash

**守护式方式创建容器：**

docker run -di --name=容器名称 镜像名称:标签

**登录守护式容器方式：**

docker exec -it 容器名称 (或者容器ID) /bin/bash

参数说明：

**-i：**表示运行容器

**-t：**表示容器启动后会进入其命令行。加入-i -t这两个参数后，容器创建就能登录进去。即分配一个伪终端。

**–name：**为创建的容器命名。

**-v：**表示目录映射关系（前者是宿主机目录，后者是映射到宿主机上的目录），可以使用多个－v做多个目录或文件映射。注意：最好做目录映射，在宿主机上做修改，然后共享到容器上。

**-d：**在run后面加上-d参数,则会创建一个守护式容器在后台运行（这样创建容器后不会自动登录容器，如果只加-i -t两个参数，创建后就会自动进去容器）。

**-p：**表示端口映射，前者是宿主机端口，后者是容器内的映射端口。可以使用多个-p做多个端口映射。

**方式二：**

docker create [OPTIONS] IMAGE [COMMAND] [ARG...]

命令参数(OPTIONS)：

-t, --tty：分配一个伪TTY，也就是分配虚拟终端；

-i, --interactive：即使没有连接，也要保持STDIN打开；

--name：为容器起名，如果没有指定将会随机产生一个名称；

命令参数(COMMAND\ARG)：

COMMAND表示容器启动后，需要在容器中执行的命令，如ps、ls 等命令；

ARG表示执行 COMMAND 时需要提供的一些参数，如ps 命令的aux、ls命令的-a等。

命令演示：

docker create --name test-container centos:centos7 ps -A

docker create -ti --name test-container2 centos /bin/bash

### 2) 查看容器

docker ps

查看所有容器：

docker ps –a

# 格式化查看

docker ps -a --format "table {{.Image}}\t{{.ID}}\t{{.Ports}}\t{{.Names}}"

查看最后一次运行的容器：

docker ps –l

查看停止的容器：

docker ps -f status=exited

### 3) 进入容器

docker exec -it 容器id/bin/bash

docker exec -it 427 /bin/bash

### 4) 查看容器ip

docker inspect 容器id

只查看容器ip地址

docker inspect --format='{{.NetworkSettings.IPAddress}}' 容器名称/容器ID

### 5) 启动和停止容器

docker start 容器名称（或者容器ID）

docker stop 容器名称（或者容器ID）

## 3．文件拷贝

将文件拷贝到容器内可以使用cp命令：

docker cp 需要拷贝的文件或目录 容器名称:容器目录

将文件从容器内拷贝出来：

docker cp 容器名称:容器目录 需要拷贝的文件或目录

## 4．目录挂载

数据卷：

数据卷是宿主机中的一个目录或文件，当容器目录和数据卷目录绑定后，对方的修改会立即同步，一个数据卷可以被多个容器同时挂载，一个容器也可以挂载多个数据卷；

数据卷作用：

容器数据持久化；外部机器和容器间接通信；容器之间数据交换；数据集操作命令；

数据卷操作的基本语法如下：

docker volume [COMMAND]

COMMAND的命令：

create：创建一个volume

inspect：显示一个或多个volume的信息

ls：列出所有的volume

prune：删除未使用的volume

rm：删除一个或多个指定的volume

**创建数据卷：**

docker volume create html

**查看所有数据：**

docker volume ls

**查看数据卷详细信息卷：**

docker volume inspect html

**删除数据卷：**

docker volume rm 数据卷名称

## 5．Dockerfile文件

Dockerfile是由一系列命令和参数构成的脚本，这些命令应用于基础镜像并最终创建一个新的镜像。

1) 对于开发人员：可以为开发团队提供一个完全一致的开发环境；

2) 对于测试人员：可以直接拿开发时所构建的镜像或者通过Dockerfile文件构建一个新的镜像开始工作了；

3) 对于运维人员：在部署时，可以实现应用的无缝移植。



常用命令：



# 五、docker案例

## 1.一般的脚本文件，如何进行docker管理

具体步骤：

1) 切换至所需管理的脚本文件目录；

2) 手动创建Dockerfile文件，并在里面编写命令；

3) Dockerfile文件编写完成后，执行生成镜像文件的命令；

4) 镜像生成后，推送至docker仓库进行保存。

## 2.dockerfile构建镜像实例

### 1) dockerfile构建httpd实例

**第一步：**

mkdir /opt/apache

cd /opt/apache

#创建并进入文件夹,每个服务一个文件夹

**第二步：**

vim Dockerfile

#创建Dockerfile文件，文件内容如下

FROM centos:7

#基于的基础镜像指定为centos:7，注意本机要有此基础镜像

MAINTAINER this is apache image <test>

#说明镜像维护人信息为，指定用户为test

RUN yum -y update

RUN yum -y install httpd

#镜像RUN指令指定启动容器后的运行命令，yum安装update和httpd服务

EXPOSE 80

#开启80端口

ADD index.html /var/www/html/index.html

#复制宿主机index.html文件到容器的 /var/www/html/index.html

ENTRYPOINT ["/usr/sbin/apachectl"]

CMD ["-D","FOREGROUND"]

#使用前台启动apache注意使用绝对路径，如果后台启动，启动完shell就结束了容器就结束了，保存退出。

**第三步：**

echo "this is test web">index.thml

#准备网站的网页，注意必须在和Dockerfile文件相同的目录

**第四步：**

docker build -t httpd:centos .

#将dockerfile 文件生成镜像注意最后有个点。

**第五步：**

docker run -d -p 40330:80 httpd:centos

#新镜像运行容器，映射宿主机的40330端口为容器的80端口

访问192.168.30.11:40330 进行测试是否可以访问容器的httpd主业

### 2) dockerfile构建sshd实例

**第一步：略；**

**第二步：**

FROM centos:7

#指定sshd基于centos:7的镜像

MAINTAINER this is ssh image <ssh>

#说明镜像信息，镜像用户为ssh

RUN yum -y update

#此内容若保存可注释掉

#运行RUN指令执行update更新

RUN yum -y install openssh\* net-tools lsof telnet passwd

#RUN指令执行安装openssh相关工具，net工具，lsof、telnet、passwd程序

RUN echo "abc123" |passwd --stdin root

#免交互设置root密码为abc123

RUN sed -i 's/UsePAM yes/UsePAM no/g' /etc/ssh/sshd\_config

#关闭sshd服务的pam认证

RUN sed -i '12d' /etc/pam.d/sshd

#删除/etc/pam.d/sshd的12行，删除12行为取消pam的限制

RUN ssh-keygen -t rsa -A

#生成秘钥认证文件

RUN mkdir -p /root/.ssh && chown root.root /root && chmod 700 /root/.ssh

#递归创建/root/.ssh文件夹并修改属主属租为root，添加权限为700只允许属组属组用户操作

EXPOSE 22

#规定端口为22

ENTRYPOINT [ "/usr/sbin/sshd" ]

CMD ["-D", "FOREGROUND"]

#/usr/sbin/sshd -D用于前台启动sshd服务

**第三步：**

docker build -t sshd:centos .

#创建sshd；centos镜像

**第四步：**

docker run -d -p 40022:22 sshd:centos

#运行sshd:centos容器并映射容器中的22端口为40022端口

**第五步：**

docker ps

#查看容器信息

ssh localhost -p 40022

ifconfig

#宿主机执行验证sshd容器

### 3) dockerfile构建systemd实例

**第一步：**

mkdir /opt/systemctl

cd /opt/systemctl

#创建systemd目录

**第二步：**

vim Dockerfile

#编辑Dockerfile文件内容如下

FROM sshd:centos

#以sshd:centos为基础镜像，注意要做上面的sshd实例，不然无此进项不能制作systemd镜像

MAINTAINER this is systemctl image <systemd>

ENV container docker

#除了systemd-tmpfiles-setup.service,删除其它所有文件

RUN (cd /lib/systemd/system/sysinit.target.wants/; for i in \*; do [ $i == systemd-tmpfiles-setup.service ] || rm -f $i;done); \

rm -f /lib/systemd/system/multi-user.target.wants/\*; \

rm -f /etc/systemd/system/\*.wants/\*; \

rm -f /lib/systemd/system/local-fs.target.wants/\*; \

rm -f /lib/systemd/system/sockets.target. wants/\*udev\*; \

rm -f /lib/systemd/system/sockets.target.wants/\*initctl\*; \

rm -f /lib/systemd/system/basic.target.wants/\*;\

rm -f /lib/systemd/system/anaconda.target.wants/\*;

VOLUME [ "/sys/fs/cgroup" ]

CMD ["/usr/sbin/init"]

**第三步：**

docker build -t systemd:centos .

#生成systemd镜像

**第四步：**

docker run --privileged -d -P -v /sys/fs/cgroup:/sys/fs/cgroup:ro systemd:centos /sbin/init

#启动容器，并挂载宿主机目录挂载到容器中，和进行初始化

#--privileged:使container内的root拥有真正的root权限。否则，container内的root只是外部的一个普通用户权限。

**第五步：**

docker ps -a

docker exec -it 56b06afcef9a /bin/bash

#进入容器

systemctl start sshd

systemctl status sshd

### 4) dockerfile构建nginx实例

**第一步：**

mkdir /opt/nginx

cd /opt/nginx/

cp /opt/nginx-1.12.0.tar.gz /opt/nginx

#创建nginx目录，将nginx安装包放到创建的nginx目录下，必须与Dockerfile文件在同一目录下

**第二步：**

vim Dockerfile

#编辑nginx的dockerfile文件内容如下

FROM centos:7

#基于基础镜像，centos

MAINTAINER this is nginx image <nginx>

#用户信息，镜像维护用户为nginx

RUN yum -y update

#此内容若保存可注释掉

RUN yum -y install pcre-devel zlib-devel gcc gcc-C++ make

RUN useradd -M -s /sbin/ nologin nginx

#安装编译安装工具

ADD nginx-1.12.0.tar.gz /opt/

#上传nginx软件压缩包，docker自动解压

WORKDIR /opt/nginx-1.12.0

RUN ./ configure \

--prefix=/usr/local/nginx \

--user=nginx \

--group=nginx \

--with-http\_stub\_status\_module && make && make install

ENV PATH /usr/local/nginx/sbin:$PATH

#指定工作目录

EXPOSE 80

EXPOSE 443

#指定http和https端口

RUN echo "daemon off;" >> /usr/local/nginx/conf/nginx.conf

#关闭nginx 在后台运行

ADD run.sh /run.sh

RUN chmod 755 /run.sh

CMD ["/run.sh"]

#添加宿主机中run.sh到容器中

**第三步：**

vim run.sh

#创建run.sh脚本，注意必须和dockerfile在同一路径下，内容如下

#!/bin/bash

/usr/local/nginx/sbin/nginx

**第四步：**

docker build -t nginx:centos .

**第五步：**

docker run -itd -P nginx:centos /bin/bash

**第六步：**

docker ps -a

#查看nginx容器,访问80对应的随机端口验证，443的端口验证不了，ngingx中每家ssl模块

http://192.168.30.11:32769

# 六、docker中常见问题

**问题一：docker build -t myimage:v1 .中的“.”可不可以不加？**

在 docker build -t myimage:v1 . 命令中，. 是指当前目录，表示 Dockerfile 文件所在的目录。在构建镜像时，Docker 需要知道 Dockerfile 文件的位置，以便根据其中的指令构建镜像。

因此，. 是不能省略的。如果您省略了 .，Docker 将无法找到 Dockerfile 文件，并会报错。所以确保在构建命令中指定正确的 Dockerfile 文件所在的目录。

**问题二：在一台服务器上拉取docker镜像，还需不需要编写dockerfile文件？**

如果在另一台服务器上拉取已有的 Docker 镜像，就不需要再次编写 Dockerfile 文件。Dockerfile 是用于构建 Docker 镜像的文本文件，其中包含了各种指令和配置，用于构建镜像。

**问题二：docker exec -it 容器名称 (或者容器ID) /bin/bash中的/bin/bash可不可以不加，加了有什么好处？**

在docker exec -it 容器名称 (或者容器ID) /bin/bash 命令中，/bin/bash 是用于在容器内部启动一个 bash shell，它是一个命令行解释器，可以让您在容器内部执行命令和操作。如果您省略了 /bin/bash，那么 docker exec 命令将在容器内部执行默认的 shell，这通常是 sh 或其他 shell。然而，这可能不是您想要的操作环境，因为您可能更熟悉或更习惯使用 bash shell。此外，加上 /bin/bash 还具有以下优点：

1) 交互性：通过启动 bash shell，您可以与容器进行交互，执行任意命令，查看日志文件，安装软件包等。这对于调试、测试和操作容器内部是非常有用的。

2) 命令行补全：Bash shell 通常支持命令行补全功能，这使得在容器中执行命令更加方便和高效。

3) 环境设置：Bash shell 可以设置和更改环境变量，这对于在容器内部进行开发或测试时调整环境设置非常有用。

因此，加上 /bin/bash 可以提供更好的交互性和更方便的操作体验，特别是在需要进行调试、测试或开发的情况下。如果您只是想执行一个简单的命令，并且不需要交互，那么可以省略 /bin/bash。

# 七、参考网址：

第一个：

[https://blog.csdn.net/Alex\_yuan666/article/details/125126947?ops\_request\_misc=&request\_id=&biz\_id=102&utm\_term=docker&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduweb~default-1-125126947.142^v96^pc\_search\_result\_base5&spm=1018.2226.3001.4449](https://blog.csdn.net/Alex_yuan666/article/details/125126947?ops_request_misc=&request_id=&biz_id=102&utm_term=docker&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-1-125126947.142%5ev96%5epc_search_result_base5&spm=1018.2226.3001.4449)

第二个：

[https://blog.csdn.net/lxlsygxs2017/article/details/129973983?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_baidulandingword~default-0-129973983-blog-128113610.235^v38^pc\_relevant\_anti\_vip&spm=1001.2101.3001.4242.1&utm\_relevant\_index=3](https://blog.csdn.net/lxlsygxs2017/article/details/129973983?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-129973983-blog-128113610.235%5ev38%5epc_relevant_anti_vip&spm=1001.2101.3001.4242.1&utm_relevant_index=3)

第三个：

[https://blog.csdn.net/weixin\_67287151/article/details/130264947?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_baidulandingword~default-0-130264947-blog-121115171.235^v38^pc\_relevant\_anti\_vip&spm=1001.2101.3001.4242.1&utm\_relevant\_index=3](https://blog.csdn.net/weixin_67287151/article/details/130264947?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-130264947-blog-121115171.235%5ev38%5epc_relevant_anti_vip&spm=1001.2101.3001.4242.1&utm_relevant_index=3)