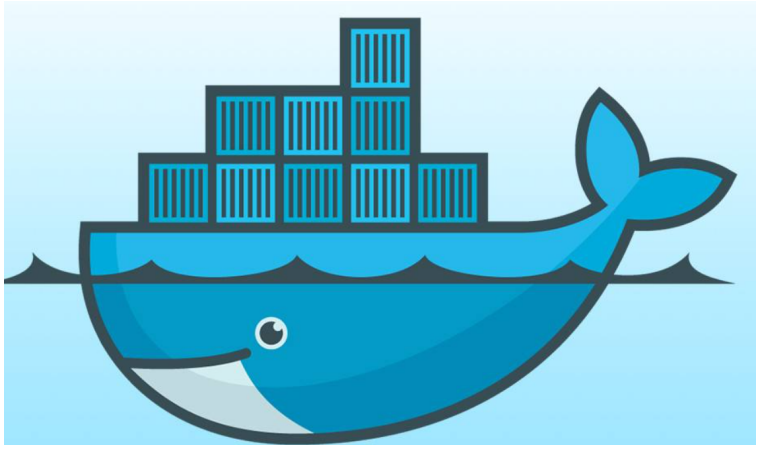
Docker讲义

讲师：元敬



一、Docker简介

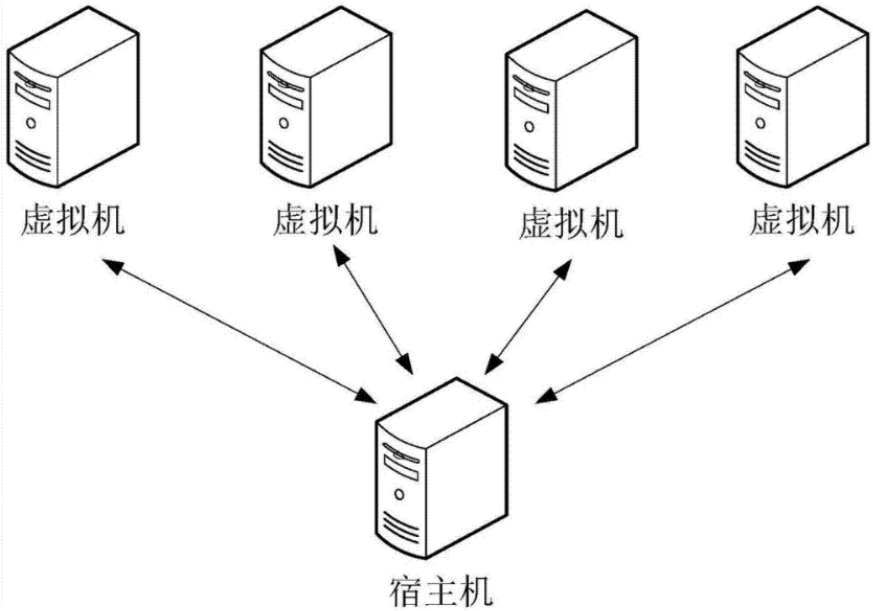
1.1 虚拟化技术

虚拟化技术是一种计算机资源管理技术，是将计算机的各种实体资源，如服务器、网络、内存及存储等，予以抽象、转换后呈现出来。虚拟化技术打破了计算机实体结构间的，不可切割的障碍。使用户可以比原本的组态更好的方式，来应用这些资源。

虚拟化技术主要作用：

- 高性能的物理硬件产能过剩和老的旧的硬件产能过低的重组重用，透明化底层物理硬件
- 软件跨环境迁移问题(代码的水土不服)

在一台主机上实现多个操作系统，关键技术就是硬件的虚拟化。



1.2 什么是Docker

首先，我们先来看几个问题：

1.合作开发的时候，在本机可以跑，别人的电脑跑不起来

这里我们拿Java Web应用程序举例，我们一个Java Web应用程序涉及很多东西，比如JDK、tomcat、spring等等。当这些其中某一项版本不一致的时候，可能就会导致应用程序跑不起来这种情况。Docker则将程序直接打包成镜像，直接运行在容器中即可。

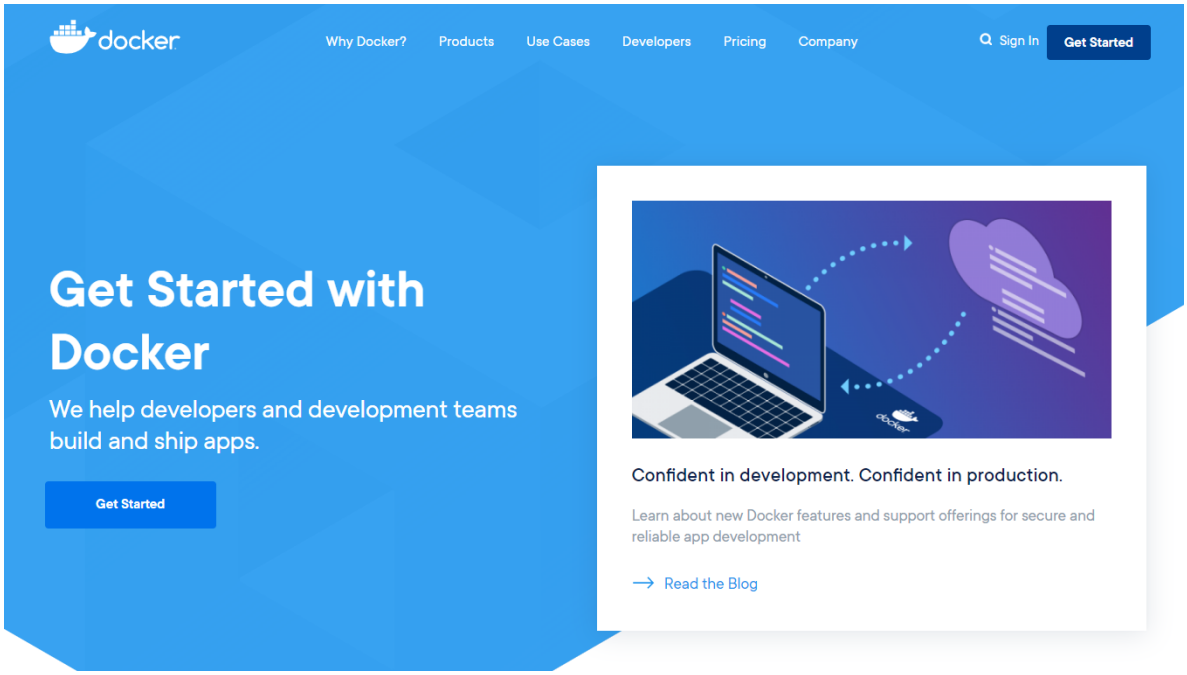
2.服务器自己的程序挂了，结果发现是别人程序出了问题把内存吃完了，自己程序因为内存不够就挂了

这种也是一种比较常见的情况，如果你的程序重要性不是特别高的话，公司基本上不可能让你的程序独享一台服务器的，这时候你的服务器就会跟公司其他人的程序共享一台服务器，所以不可避免地就会受到其他程序的干扰，导致自己的程序出现问题。Docker就很好解决了环境隔离的问题，别人程序不会影响到自己的程序。

3.公司要弄一个活动，可能会有大量的流量进来，公司需要再多部署几十台服务器

在没有Docker的情况下，要在几天内部署几十台服务器，这对运维来说是一件非常折磨人的事，而且每台服务器的环境还不一定一样，就会出现各种问题，最后部署地头皮发麻。用Docker的话，我只需要将程序打包到镜像，你要多少台服务，我就给你跑多少容器，极大地提高了部署效率。

> 官网地址：https://www.Docker.com



> Debug your app，not your environment!
>
> 调试你的应用程序，而不是你的开发环境；

官网的介绍是“Docker is the world’s leading software container platform.”官方给Docker的定位是一个应用容器平台。

Docker 是一个开源的应用容器引擎，诞生于 2013 年初，基于 Go 语言实现， dotCloud 公司出品（后改名为Docker Inc）；

Docker 可以让开发者打包他们的应用，以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上。Docker容器是完全使用沙箱机制，相互隔离，性能开销也极低。

从 17.03 版本之后，Docker分为 CE（Community Edition: 社区版） 和 EE（Enterprise Edition: 企业版）

Docker通俗的讲是**服务器中高性能的虚拟机**，可以将一台物理机虚拟N多台虚拟机的机器，互相之间隔离，互不影响。

想要搞懂Docker，其实看它的两句口号就行。

第一句，是**“Build, Ship and Run”**。

也就是，“搭建、发送、运行”，三板斧。

举个例子：

我来到一片空地，想建个房子，于是我搬石头、砍木头、画图纸，一顿操作，终于把这个房子盖好了。

结果，我住了一段时间，想搬到另一片空地去。这时候，按以往的办法，我只能再次搬石头、砍木头、画图纸、盖房子。

但是，跑来一个老巫婆，教会我一种魔法。

这种魔法，可以把我盖好的房子复制一份，做成“镜像”，放在我的背包里。

等我到了另一片空地，就用这个“镜像”，复制一套房子，摆在那边，拎包入住。

怎么样？是不是很神奇？

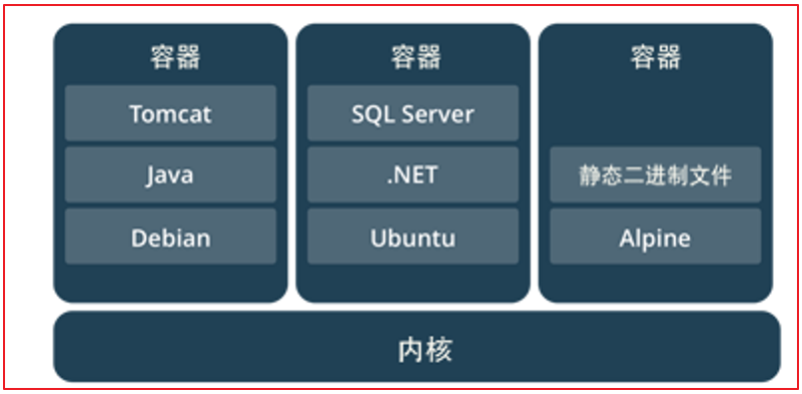
所以，Docker的第二句口号就是：**“Build once，Run anywhere（搭建一次，到处能用）”。**

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的Linux机器或Windows 机器上,也可以实现虚拟化，容器是完全使用沙箱机制，相互之间不会有任何接口。

特点：

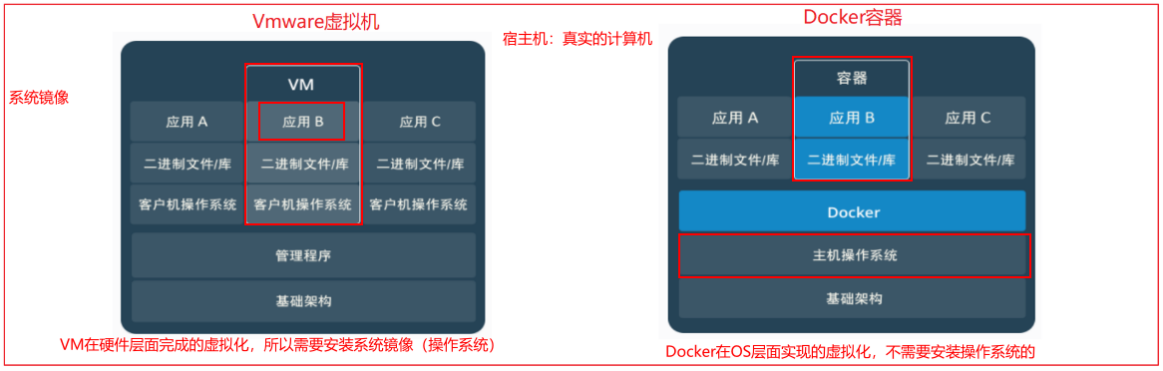
- 标准化交付：Docker将应用打包成标准化单元，用于交付、部署；
- 轻量级：容器及包含了软件运行所需的所有环境，而且非常轻量级
- 高可靠：容器化的应用程序，可以在任何Linux环境中始终如一的运行

- 隔离性：容器化的应用程序，具备隔离性，这样多团队可以共享同一Linux系统资源



1.3 容器与虚拟机比较

下面的图片比较了 Docker 和传统虚拟化方式的不同之处，可见Docker是在操作系统层面上实现虚拟化，直接复用本地主机的操作系统，而传统方式则是在硬件层面实现。



特性	容器	虚拟机
启动	秒级	分钟级
硬盘使用	一般为MB	一般为GB
性能	接近原生硬件	弱鸡
系统支持量	单机可跑几十个容器	单机几个虚拟OS
运行环境	主要在Linux	主要在window

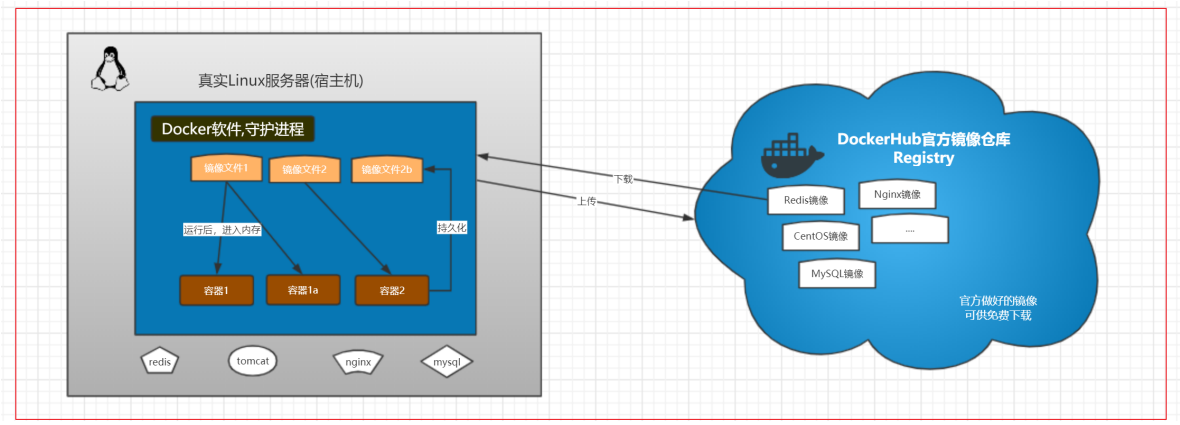
相同：容器和虚拟机都是虚拟化技术，具备资源隔离和分配优势

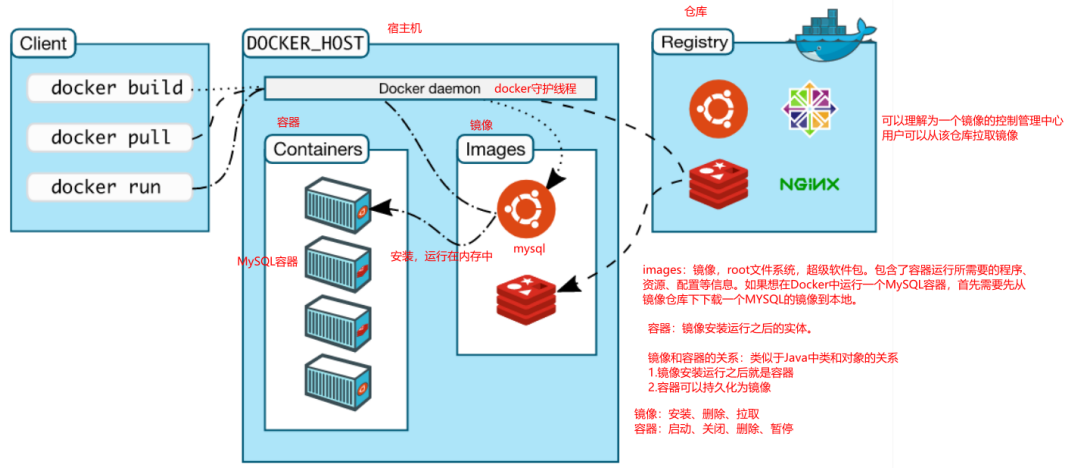
不同：

- Docker虚拟化的是操作系统，虚拟机虚拟化的是硬件
- 传统虚拟机可以运行不同的操作系统，Docker主要运行同一类操作系统(Linux)

比较上面两张图，我们发现虚拟机是携带操作系统，本身很小的应用程序却因为携带了操作系统而变得非常大，很笨重。Docker是不携带操作系统的，所以Docker的应用就非常的轻巧

1.4 Docker 基本概念





宿主机：安装Docker守护进程的Linux服务器，称之为宿主机；

镜像（Image）： Docker 镜像，就相当于是一个 root 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一些配置参数。

容器（Container）： 镜像运行之后的实体，镜像和容器的关系，就像是面向对象程序设计中的类和对象一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。

仓库（Repository）： 仓库可看成一个镜像控制中心，用来保存镜像。

举例：

刚才例子里面，那个放在包里的“镜像”，就是**Docker镜像。**而我的背包，就是**Docker仓库。**我在空地上，用魔法造好的房子，就是一个**Docker容器。**

说白了，这个Docker镜像，是一个特殊的文件系统。它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一些配置参数（例如环境变量）。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。

也就是说，每次变出房子，房子是一样的，但生活用品之类的，都是不管的。谁住谁负责添置。

每一个镜像可以变出一种房子。那么，我可以有多个镜像呀！

也就是说，我盖了一个欧式别墅，生成了镜像。另一个哥们可能盖了一个中国四合院，也生成了镜像。还有哥们，盖了一个非洲茅草屋，也生成了镜像

这么一来，我们可以交换镜像，你用我的，我用你的，是不是就很爽。

于是乎，就变成了一个大的公共仓库。

负责对Docker镜像进行管理的，是**Docker Registry服务**（类似仓库管理员）。

不是任何人建的任何镜像都是合法的。万一有人盖了一个有问题的房子呢？

所以，Docker Registry服务对镜像的管理是非常严格的。

最常使用的Registry公开服务，是官方的**Docker Hub**，这也是默认的 Registry，并拥有大量的高质量的官方镜像。

二、Docker安装与启动

本地电脑安装虚拟机，虚拟机搭配Linux操作系统，在Linux操作系统上安装Docker容器。

Docker需要从镜像仓库下载镜像，需要联网。

提供的虚拟机网卡地址：192.168.200.128

root

lagou

准备工作：确保Linux系统能够连接网络

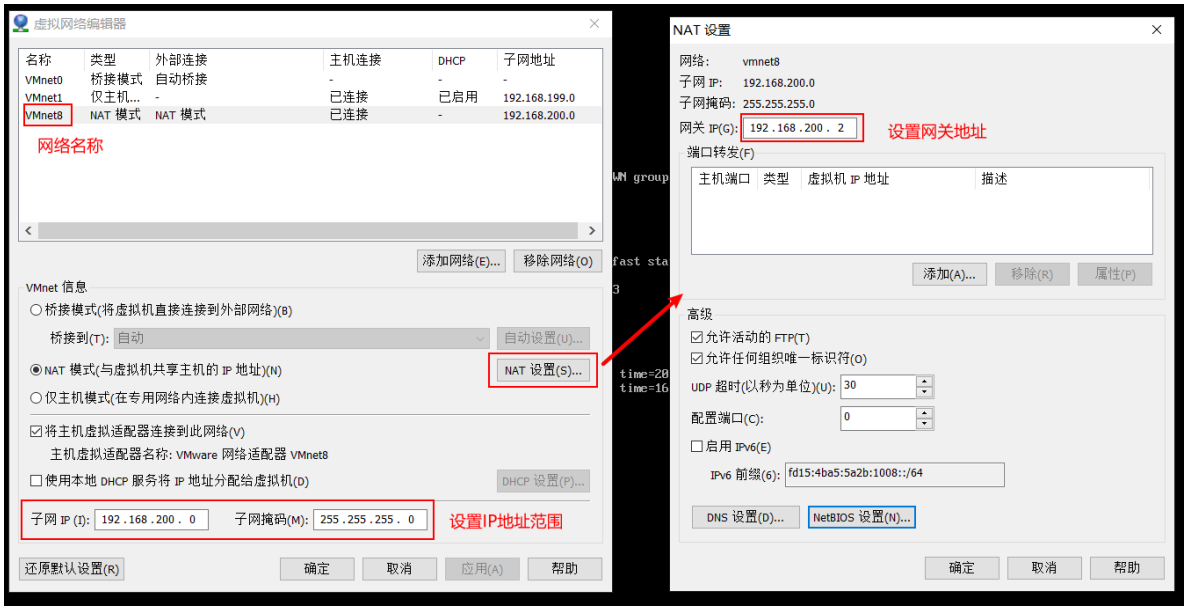
1、调整虚拟网络编辑器：

（1）桥接模式的虚拟机，就像一个在路由器"民政局"那里"上过户口"的成年人，有自己单独的居住地址，虽然和主机住在同一个大院里，但好歹是有户口的人，可以大摇大摆地直接和外面通信。

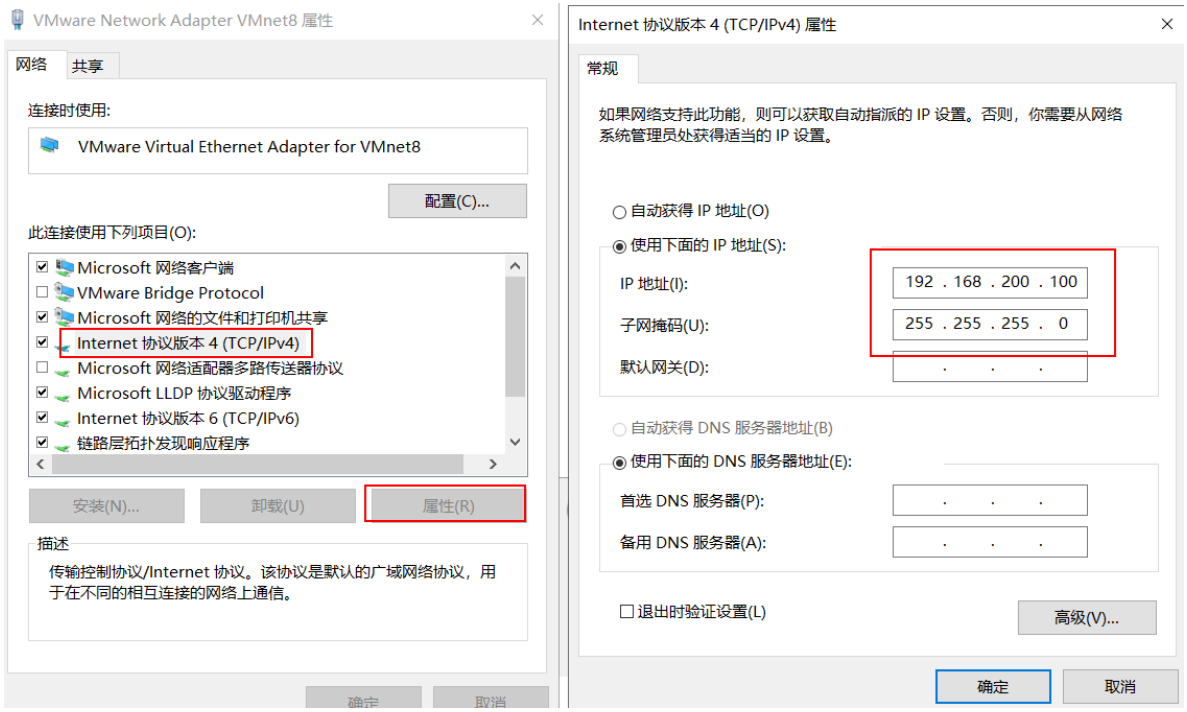
（2）NAT模式的虚拟机，纯粹就是一个没上过户口的黑户，路由器"民政局"根本不知道有这么个人，自然也不会主动和它通信。即使虚拟机偶尔要向外面发送点的信件，都得交给主机以主机的名义转发出去，主机还专门请了一位叫做NAT的老大爷来专门负责这些虚拟机的发信、收信事宜。

（3）仅主机模式的虚拟机，纯粹是一个彻彻底底的黑奴，不仅没有户口、路由器"民政局"不知道这么号人，还被主机关在小黑屋里，连信件也不准往外发。

其中这个仅主机模式能够保障我们在拔掉网线的情况下继续连接我们的虚拟机，不依靠公网连接，而是依靠物理机和虚拟机的关系连接。在断网的情况下，利用这个模式，我们可以继续连接虚拟机，实现我们的操作。

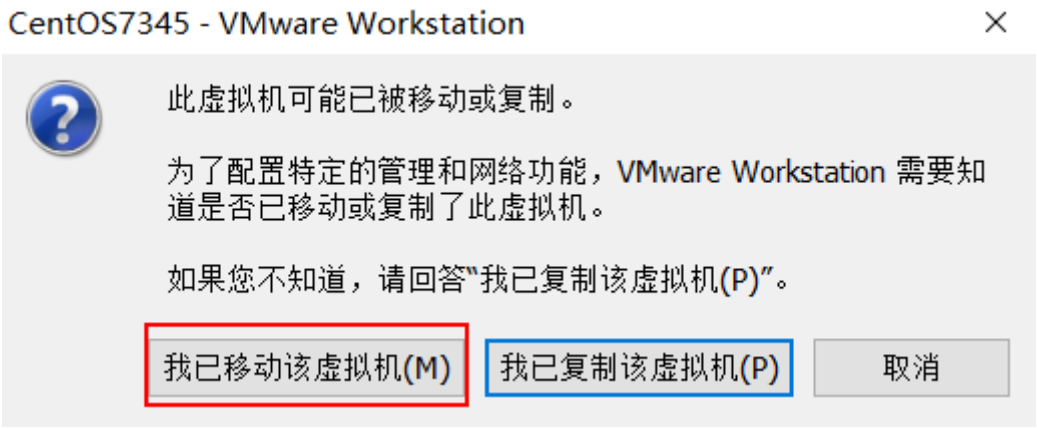


2、VMware Network Adapter VMnet8网卡设置



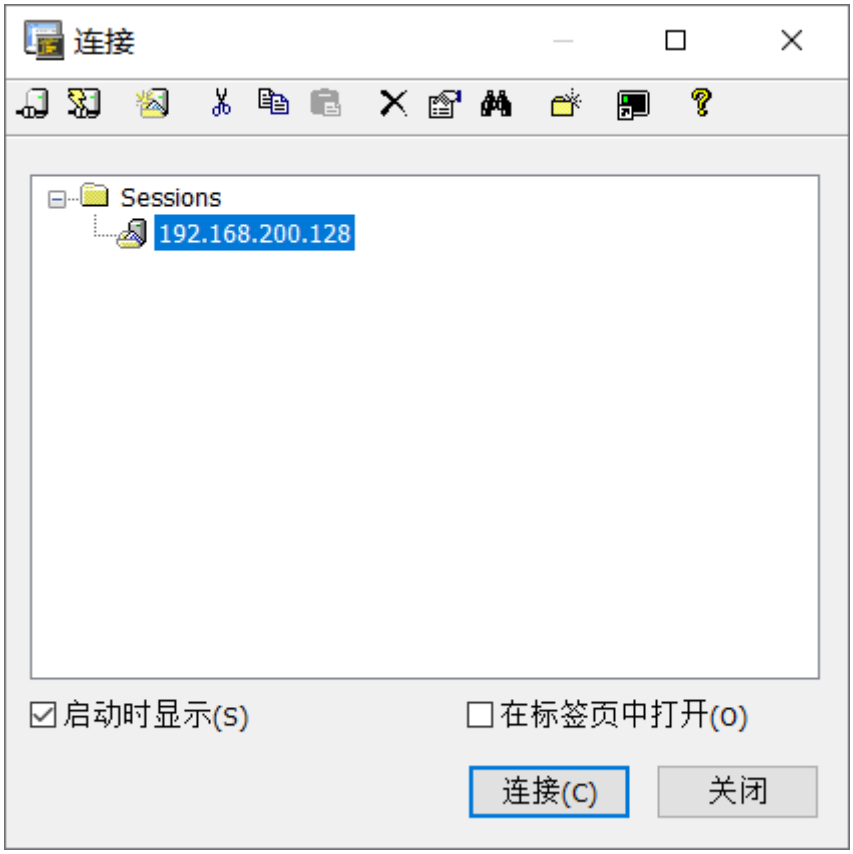
3、打开虚拟机文件：

提供的虚拟机文件已经安装了CentOS7，并且在CentOS7上已经安装了Docker环境，IP地址：192.168.200.128

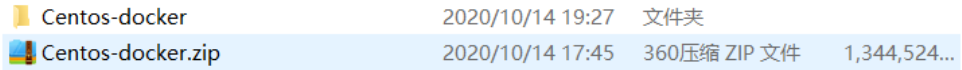


4、连接到虚拟机

SSH、SecureCRT



2.1 安装



Docker官方建议在Ubuntu中安装Docker软件。因为Docker基于Ubuntu发布，而且Docker出现问题时，Ubuntu系统版本的一般是先打补丁。很多版本在CentOS中，是不支持更新最新补丁包的。没有好的解决方案。

但是，由于我们学习的环境都使用CentOS。因此，这里我们将Docker安装到CentOS上。注意，一定安装在CentOS 7.x及以上版本，**CentOS6.x的版本中有Bug**!

> 请直接挂载课程配套资料的Centos7.x镜像

（1）查看电脑上已经已经安装Docker

```
yum list installed | grep docker
```

（2）安装docker

```
yum -y install docker
```

（3）安装后查看docker版本

```
docker -v
```

```
[root@localhost ~]# docker -v
Docker version 19.03.5, build 633a0ea
```

2.2 Docker守护进程相关命令

概要：

- 启动docker服务
- 停止docker服务
- 重启docker服务
- 查看docker服务状态
- 开机启动docker服务
- 查看docker概要信息

详解：

systemctl命令是系统服务管理器指令

启动docker：

```
systemctl start docker
```

停止docker：

```
systemctl stop docker
```

重启docker：

```
systemctl restart docker
```

查看docker状态：

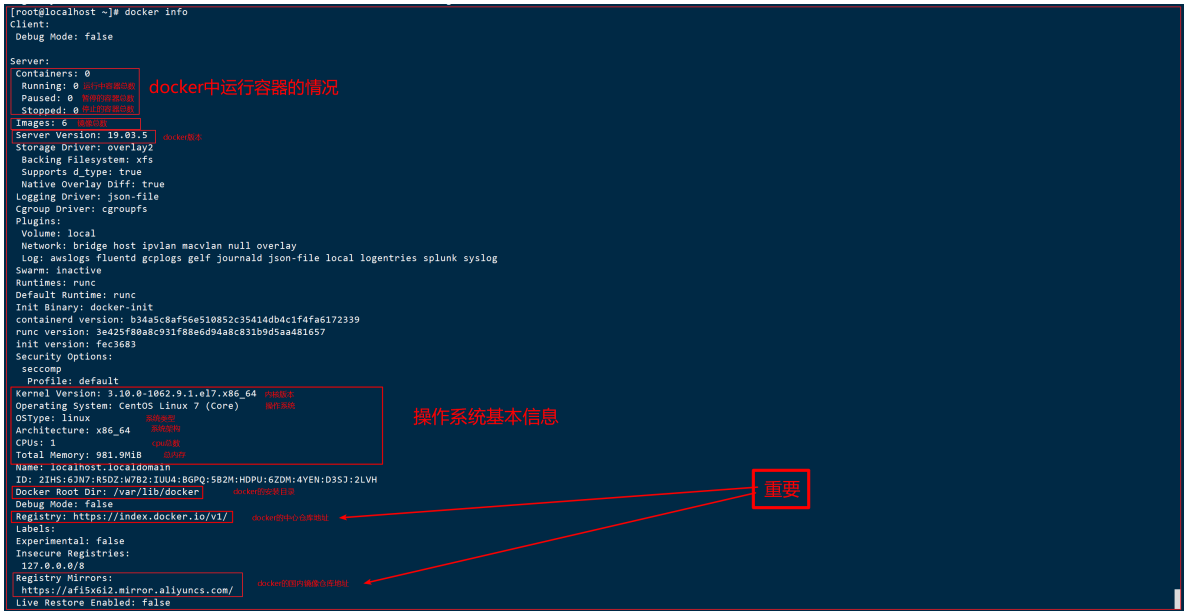
```
systemctl status docker
```

开机启动：

```
systemctl enable docker
```

查看docker概要信息

```
docker info
```



查看docker帮助文档

```
docker --help
```

2.3 镜像加速的2个方案

默认情况，将从docker hub（https://hub.docker.com/）下载docker镜像太慢，一般都会配置镜像加速器；

方案一：中科大

中国科学技术大学(ustc)是老牌的linux镜像服务提供者了，还在遥远的ubuntu 5.04版本的时候就在用。ustc的docker镜像加速器速度很快。ustc docker mirror的优势之一就是不需要注册，是真正的公共服务。

https://lug.ustc.edu.cn/wiki/mirrors/help/docker

编辑该文件：

```
vim /etc/docker/daemon.json
```

在该文件中输入如下内容：

```
{
"registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]
}
```

方案二：阿里云

如果中科大镜像加载速度很慢，建议配置阿里云镜像加速，这个镜像仓库如果不好使，可以自己从阿里云上申请！速度杠杠的~

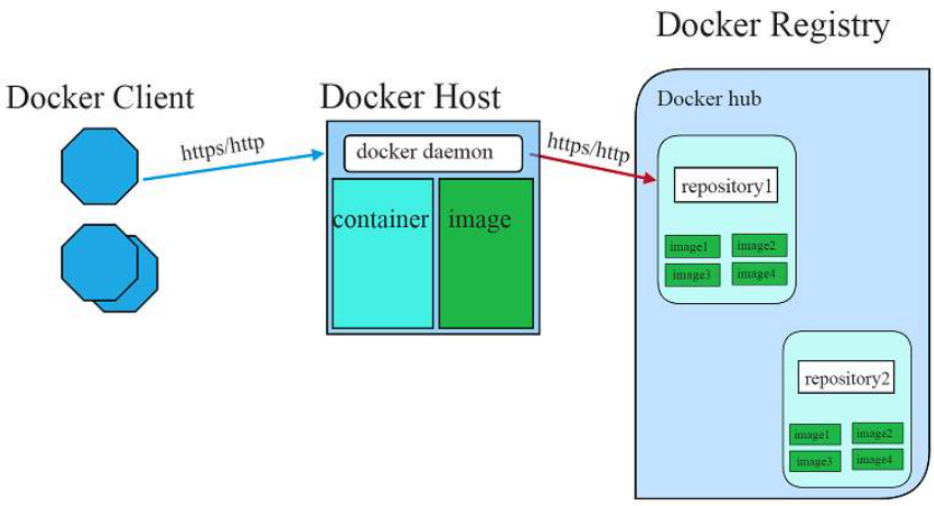
```
{
  "registry-mirrors": ["https://3ad96kxd.mirror.aliyuncs.com"]
}
```

必须要注册，每个人分配一个免费的docker镜像加速地址，速度极快

配置完成记得刷新配置

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

三、Docker常用命令



3.1 镜像相关命令

概要:

- 查看镜像　docker images
- 搜索镜像　docker search imageName
- 拉取镜像　docker pull imageName：version
- 删除镜像　docker rmi imageId　　rmi-->remove Image

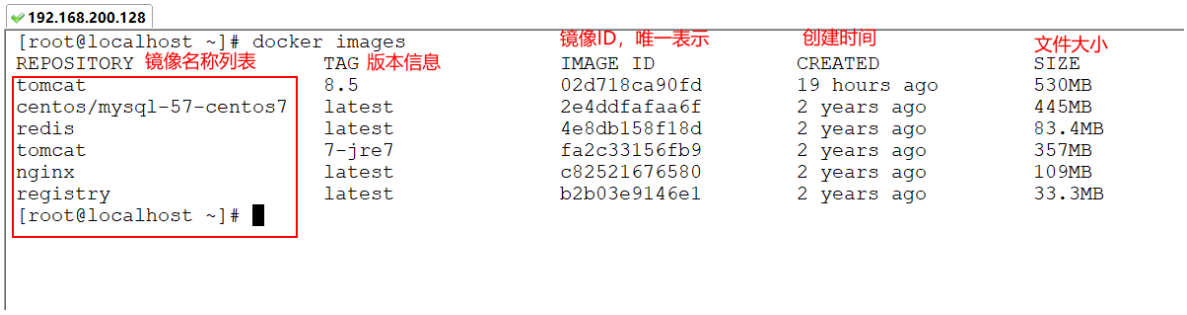
详解:

3.1.1 查看镜像

查看本地所有镜像

```
docker images
```

```
[root@localhost ~]# docker images
REPOSITORY 镜像名称                TAG 镜像标签        IMAGE ID 镜像ID      CREATED 镜像创建日期   SIZE 镜像大小
centos/mysql-57-centos7   latest              2e4ddfafaa6f        15 months ago       445MB
centos                    7                   5182e96772bf        16 months ago       200MB
redis                     latest              4e8db158f18d        16 months ago       83.4MB
tomcat                    7-jre7              fa2c33156fb9        16 months ago       357MB
nginx                     latest              c82521676580        16 months ago       109MB
registry                  latest              b2b03e9146e1        17 months ago       33.3MB
```



这些镜像都是存储在Docker宿主机的/var/lib/docker目录下

3.1.2 搜索镜像

如果你需要从网络中查找需要的镜像，可以通过以下命令搜索；**==注意，必须确保当前系统能联网==**

```
docker search 镜像名称
```

3.1.3 拉取镜像

拉取镜像:从Docker仓库下载镜像到本地，镜像名称格式为 名称:版本号，如果版本号不指定则是最新的版本。如果不知道镜像版本，可以去docker hub 搜索对应镜像查看。

```
docker pull 镜像名称
```

例如，我要下载centos7镜像

```
docker pull centos:7
```

3.1.4 删除镜像

按镜像ID删除镜像

```
docker rmi 镜像ID
```

删除所有镜像

删除之前要确认此镜像已经没有被容器在使用，如果存在正在运行的docker容器，删除会报错“Error: container_delete: Impossible to remove a running container, please stop it first”

```
docker images -q   #查看所有镜像的ID
docker rmi `docker images -q`  #批量删除镜像
```

3.2 容器相关命令

概要：

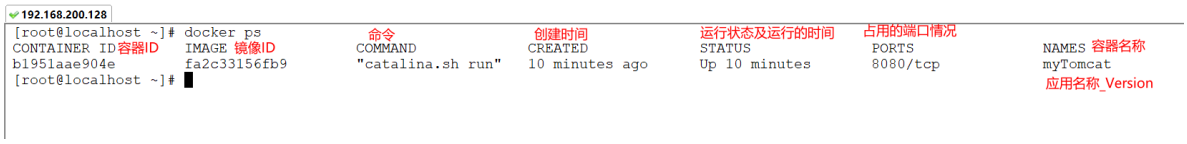
- 查看容器
- 创建容器：交互式、守护式
- 进入容器
- 启动容器
- 停止容器
- 删除容器

详解：

3.2.1 查看容器

查看正在运行的容器

```
docker ps
```



查看所有容器（查看正在运行的和已经停止运行的）

```
docker ps -a
docker ps -all
```

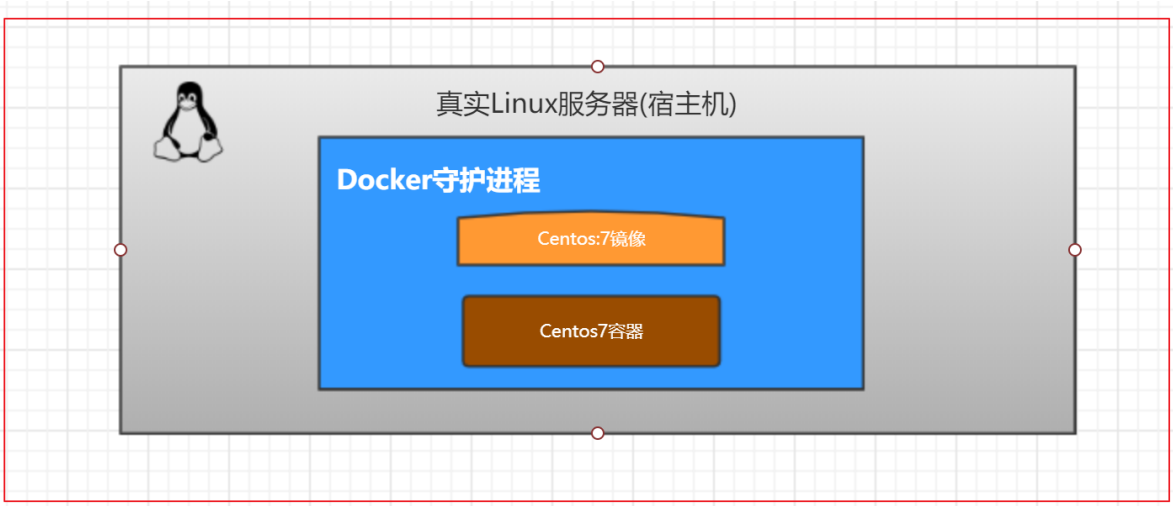
查看最后一次运行的容器

```
docker ps -l
```

查看停止的容器

```
docker ps -f status=exited
```

3.2.2 创建与启动容器



创建容器命令：

```
docker run 参数 镜像名称:镜像标签 /bin/bash
```

创建容器常用的参数说明：

```
## 命令参数详解
-i：表示运行容器,如果不加该参数那么只是通过镜像创建容器，而不启动。

-t：表示容器启动后会进入其命令行。加入这两个参数后，容器创建就能登录进去。即分配一个伪终端(如果只加it两个参数，创建后就会自动进去容器)。

-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）。

--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。

-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，例如：可以将Docker中Tomcat容器的8080端口映射到宿主机上的某一个端口8080，那么以后访问tomcat只需要：http://宿主机的IP：8080/

进入容器之后，初始化执行的命令：/bin/bash；可写可不写
```

（1）交互式容器

使用交互式运行容器，容器运行后直接进入到容器内部，退出容器内部后，容器直接关闭

只有第一次才会这样，以后再启动容器就是一个守护式的。

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

这时我们通过ps命令查看，发现可以看到启动的容器，状态为启动状态

```
[root@localhost ~]# docker ps
CONTAINER ID 容器ID  IMAGE          COMMAND 命令      CREATED 容器创建时间  STATUS 容器状态   PORTS 端口   NAMES
fb360444d8a8         centos:7       "/bin/bash"       54 seconds ago       Up 53 seconds                 c1 容器名称
[root@localhost ~]# 镜像名称：镜像标签                                      up上线
```

退出当前容器

```
exit
```

(2) 守护式容器:

```
docker run -di --name=容器名称 镜像名称(或镜像ID):标签 /bin/bash
```

(3) 登录容器/进入容器的目录:

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

注意:这里的登陆容器之后执行的脚本/bin/bash必须写

3.2.3 停止与启动容器

停止容器:

```
docker stop 容器名称(或者容器ID)
```

启动容器:

```
docker start 容器名称(或者容器ID)
```

3.2.4 文件拷贝

如果我们需要将宿主机的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

#新建一个空文件
touch latou.html
#拷贝到tomcat容器的webapps目录下
docker cp lagou.html 59b35c0bbe6d:/usr/local/tomcat/webapps
#切换到tomcat容器中查看
docker exec -it tomcat容器ID /bin/bash
```

也可以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

#将copy到tomcat容器的文件再copy出来
docker cp 59b35c0bbe6d:/usr/local/tomcat/webapps/lagou.html ./
```

3.2.5 目录挂载

我们可以在创建容器的时候,将宿主机的目录与容器内的目录进行映射,这样我们就可以通过修改宿主机某个目录的文件从而去影响容器。

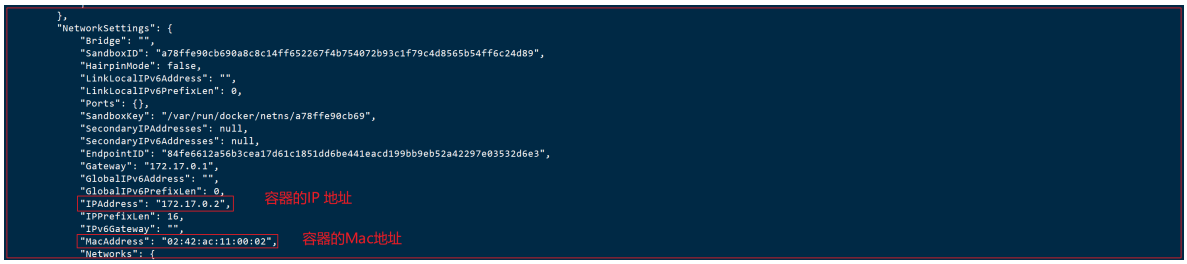
创建容器 添加-v参数 后边为 宿主机目录:容器目录 例如：

```
docker run -di -v /usr/local/myhtml:/usr/local/myhtml --name=mycentos3 centos:7
```

3.2.6 查看容器IP地址

我们可以通过以下命令查看容器运行的各种数据

```
docker inspect 容器名称（容器ID）
```




```
"Networks": {
    "bridge": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": null,
        "NetworkID": "d8a57f47d192da9af9a6ea03b5821f0da81cba3e0cd93358408e9cf4d34bae91",
        "EndpointID": "d89647d24f1a8d9f98138f45a232eeb25ae3a616fd29a730fad342f93b3adce5",
        "Gateway": "172.17.0.1", 网关
        "IPAddress": "172.17.0.2", IP地址
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:11:00:02", MAC地址
        "DriverOpts": null
    }
}
```

也可以直接执行下面的命令直接输出IP地址

```
docker inspect --format='{{.NetworkSettings.IPAddress}}' 容器名称（容器ID）
```

3.2.7 删除容器

删除指定的容器，正在运行的容器无法删除

```
#删除容器
docker rm 容器名称（容器ID）
#删除镜像
docker rmi 镜像ID(镜像名称)
```

> 问题：是否可以删除正在运行的容器?

四、Docker数据卷（Volumes）

4.1 数据卷概述

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，一个容器也可以被挂载多个数据卷。

简单来说数据卷本质其实是共享文件夹，是宿主机与容器间数据共享的桥梁。

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

4.2 数据卷配置方式

(1).1个容器挂载1个数据卷

创建启动容器时，使用 -v 参数 设置数据卷

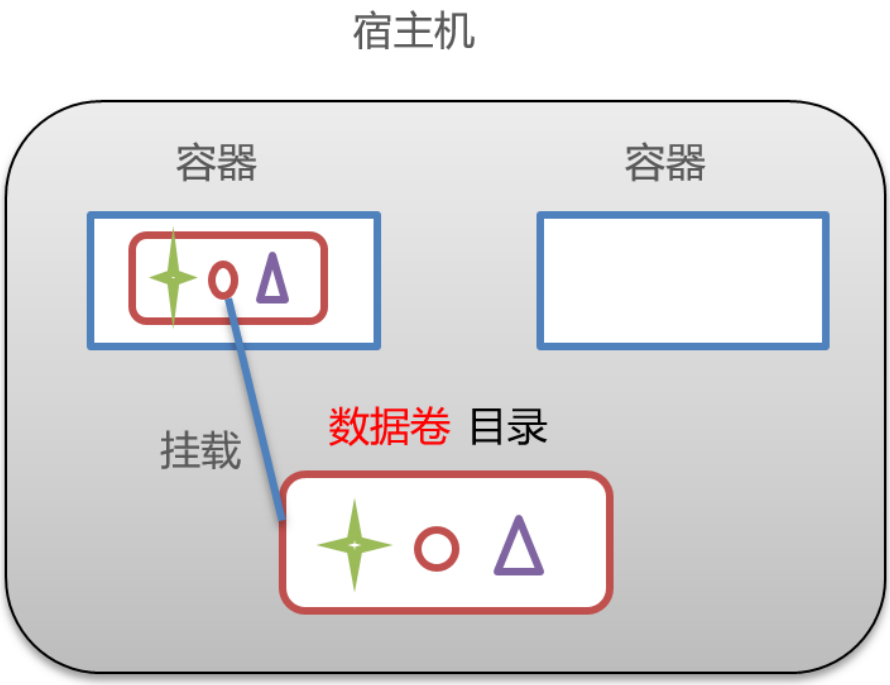
```
docker run ... -v 宿主机目录(文件):容器内目录(文件) ...
```

注意事项：

1. 目录必须是绝对路径
2. **如果宿主机目录不存在，会自动创建**
3. 可以挂载多个数据卷

案例：

```
#拉取centos镜像
docker pull centos:7
#安装启动容器并挂载
docker run -di --name=c1 -v /root/host_data1:/root/c1_data centos:7 /bin/bash
```



(2).查看容器已挂载的数据卷

我们可以通过以下命令，查看容器中挂载的数据卷

```
docker inspect 容器名称（容器ID）
```

(3).1个容器挂载多个数据卷

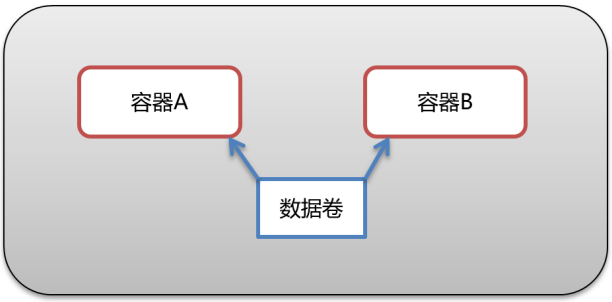
我们可以通过以下命令，挂载多个数据卷

```
docker run -di --name=c1 -v /root/host_data1:/root/c1_data1 -v
/root/host_data2:/root/c1_data2 centos:7 /bin/bash
```

(4).多个容器挂载1个数据卷

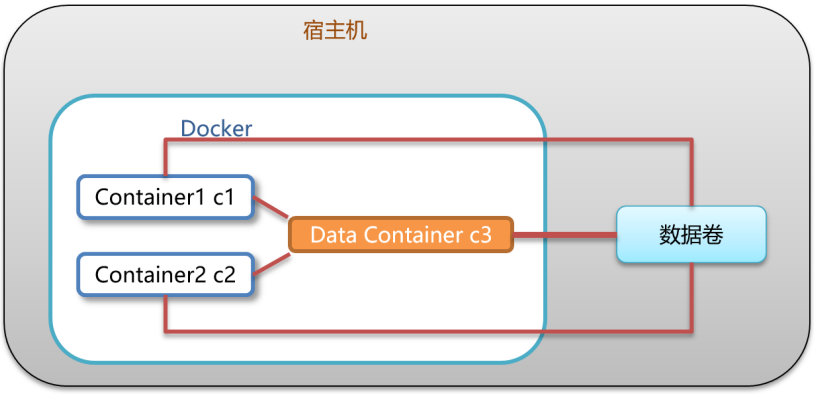
多个容器挂载1个数据卷，实现数据共享

```
docker run -di --name=c2  -v /root/host_data_common:/root/c2_data centos:7
docker run -di --name=c3  -v /root/host_data_common:/root/c3_data centos:7
```



多个容器挂载1个容器(这个容器挂载1个数据卷)

```
##创建启动c3数据卷容器，使用 -v 参数 设置数据卷
docker run -di --name=c3 -v /root/host_data_common:/root/c3_data centos:7
/bin/bash
##创建启动 c1 c2 容器，使用 --volumes-from 参数 设置数据卷
docker run -di --name=c1 --volumes-from c3 centos:7 /bin/bash
docker run -di --name=c2 --volumes-from c3 centos:7 /bin/bash
```



五、在Docker中部署软件

5.1 MySQL部署

实现步骤：

1. 搜索MySQL镜像
2. 拉取MySQL镜像
3. 创建容器、设置端口映射、设置数据卷
4. 进入容器操作mysql
5. 使用Navicat连接MySQL

实现过程：

1. 搜索mysql镜像

```
docker search mysql
```

2. 拉取mysql镜像

```
docker pull  mysql:5.7
```

3. 创建容器，设置端口映射、目录映射

```
docker run -di --name=mysql -p 3307:3306 -v /root/mysql/logs:/logs -v
/root/mysql/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=123456 mysql:5.7
```

- 参数说明：
  - **-p 3307:3306**：将容器的 3306 端口映射到宿主机的 3307 端口。
  - **-v /root/mysql/logs:/logs**：将主机目录(/root/mysql)下的 logs 目录挂载到容器中的 /logs。日志目录
  - **-v /root/mysql/data:/var/lib/mysql** ：将宿主机目录(/root/mysql)下的data目录挂载到容器的 /var/lib/mysql 。数据目录
  - **-e MYSQL_ROOT_PASSWORD=123456**：初始化 root 用户的密码。

4. 进入容器，操作mysql

```
docker exec -it c_mysql /bin/bash
```

5. 使用Navicat连接容器中的mysql

   因为我们做了端口映射，所以连接的是192.168.200.128:3307

5.2 Tomcat部署

实现步骤：

1. 搜索Tomcat镜像
2. 拉取Tomcat镜像
3. 创建容器、设置端口映射、设置数据卷
4. 向Tomcat中部署服务
5. 使用外部机器访问Tomcat，测试部署服务

实现过程：

1. 搜索tomcat镜像

```
docker search tomcat
```

2. 拉取tomcat镜像

```
docker pull tomcat:8-jdk8
```

3. 创建容器，设置端口映射、目录映射

```
docker run -di --name=c_tomcat -p 8080:8080 -v
/root/tomcat/webapps:/usr/local/tomcat/webapps tomcat:8-jdk8
```

- 参数说明：
  - **-p 8080:8080**：将容器的8080端口映射到主机的8080端口

    -v /root/tomcat/webapps:/usr/local/tomcat/webapps：将主机目录(/root/tomcat/webapps)挂载到容器的webapps

4. 向Tomcat中部署服务，使用FinalShell文件上传
5. 使用外部机器访问Tomcat，测试部署服务

5.3 Nginx部署

实现步骤：

1. 搜索Nginx镜像

```
docker search Nginx
```

2. 拉取Nginx镜像

```
docker run -di --name=mynginx -p 80:80 nginx
```

测试访问：http://192.168.200.128:80

5.4 Redis部署

实现步骤：

1. 搜索Redis镜像
2. 拉取Redis镜像
3. 创建容器、设置端口映射
4. 使用外部机器连接Redis，测试

实现过程：

1. 搜索redis镜像

```
docker search redis
```

2. 拉取redis镜像

```
docker pull redis
```

3. 创建容器，设置端口映射

```
docker run -id --name=c_redis -p 6379:6379 redis
```

4. 使用外部机器连接redis，测试

六、迁移与备份

应用场景：

开发环境Docker，在Docker中安装很多的容器，进行对应的配置，将Docker中的运行的容器持久化为镜像，将对应的镜像安装到生产环境中。

1.将开发环境下的Docker中对应的容器持久化为镜像

2.将镜像保存为一个压缩包，发送到生产环境服务器中

3.生产环境中需要将压缩包-->镜像-->容器

6.1、容器保存为镜像

我们可以通过以下命令将容器保存为镜像

```
docker commit {正在运行容器名称/容器ID} {镜像名称}:{镜像标签}
{ImageName} : {Tag}
# 例如
docker commit dbc261edcdff redis:version_lagou_1.0.0
```

镜像标签如果不写默认为latest

6.2、镜像备份

我们可以通过以下命令将镜像保存为tar 文件

```
docker save -o {压缩包存放路径} {镜像名称/镜像ID}
# 举例
docker save -o redis.tar redis:version_lagou_1.0.0  #压缩包在生产环境下会还原为一个image，还原之后的name和tag
# -o :输出到的文件
```

6.3、镜像恢复与迁移

首先我们先删除掉c_tomcat_bak镜像 然后执行此命令进行恢复

```
docker load -i {备份的镜像文件}
# 举例
docker load -i redis.tar
# -i :指定导入的文件
```

执行后再次查看镜像，可以看到镜像已经恢复，可以再次运行测试

```
docker run -di --name=mytomcat -p 8081:8080 -v
/root/tomcat/webapps/:/usr/local/tomcat/webapps redis:version_lagou_1.0.0
```