如何使用Xshell连接VMware上的Linux虚拟机

参考：<https://www.cnblogs.com/shireenlee4testing/p/9469650.html>

安装docker

参考：<https://www.cnblogs.com/qgc1995/archive/2018/08/29/9553572.html>

centos7使用yum安装软件提示 cannot find a valid baseurl for repo:base/7/x86\_64 的解决方法

由于是本地yum源安装软件，无法联网，因此使用yum安装软件时报了错，解决方法是：打开vi /etc/resolv.conf文件 新增内容如下：nameserver 8.8.8.8

1.把yum包更新到最新的（如果更新yum有问题，也可以不更新试试）

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

[root@localhost ~]# yum install -y yum-utils device-mapper-persistent-data lvm2

3.设置yum源

[root@localhost~]#yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

4.可以查看所有仓库中所有docker版本，并选择特定版本安装

[root@localhost ~]# yum list docker-ce --showduplicates | sort –r

5. 安装Docker，命令：yum install docker-ce-版本号，我选的是17.12.1.ce，如下

[root@localhost ~]# yum install docker-ce-17.12.1.ce

6. 启动Docker，命令：systemctl start docker，然后加入开机启动，如下

[root@localhost ~]# systemctl start docker

[root@localhost ~]# systemctl enable docker

Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

验证安装是否成功(有client和service两部分表示docker安装启动都成功了)

[root@localhost ~]# docker version

如果docker未启动则会报Cannot connect to the Docker daemon. Is the docker daemon running on this host?

Docker是服务器—客户端架构。命令行运行docker命令的时候，需要本机有docker服务。可以用下面的命令启动：

#service命令的用法

sudo service docker start

#systemctl命令的用法（centos7用这个）

sudo systemctl start docker

第一本DOCKER书阅读笔记

1.1Docker简介

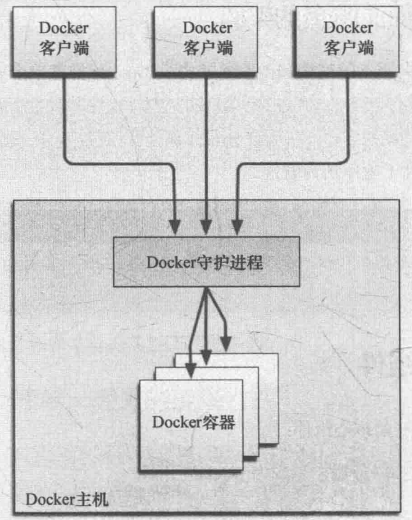
提供一个简单、轻量的建模方式；职责的逻辑分离；快速、高效的开发生命周期；鼓励使用面向服务的架构。

1.2 docker组件

核心组件：docker客户端和服务器，也称为docker引擎；docker镜像；registry；docker容器。

Docker客户端和服务器

Docker是一个客户端/服务器（C/S）架构的程序。Docker客户端只需向docker服务器或守护进程发出请求，服务器或守护进程将完成所有工作并返回结果。Docker守护进程有时也称为docker引擎。用户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进程。Docker架构如下：



Docker镜像

是构建docker世界的基石，用户基于镜像来运行自己的容器。而且镜像是基于联合（union）文件系统的一种层式的结构，由一系列指令一步一步构建出来的，例如：添加一个文件；执行一个命令；打开一个端口。

Registry

Docker用registry来保护用户构建的镜像。Registry分为公共和私有两种。

容器

用户把自己的应用程序或服务打包放进容器，容器是基于镜像启动起来的，容器中可以运行一个或多个进程。镜像是docker生命周期中的构建或打包阶段，而容器则是启动或执行阶段。总结起来，docker容器就是：一个镜像格式；一系列标准的操作；一个执行环境。

Docker借鉴了标准集装箱的概念。每个容器都包含一个软件镜像，也就是容器的货物，而且与真正的货物一样，容器里的软件镜像可以进行一些操作。例如，镜像可以被创建、启动、关闭、重启以及销毁。

1.5 docker的技术组件

包括以下几个部分：

一个原生的linux容器格式，docker中称为libcontainer

Linux内核的命名空间（namespace），用于隔离文件系统、进程和网络

文件系统隔离：每个容器都有自己的root文件系统

进程隔离：每个容器都运行在自己的进程环境中

网络隔离：容器间的虚拟网络接口和ip地址都是分开的。

资源隔离和分组：使用cgroups（即control group，linux的内核特性之一）将cpu和内存之类的资源独立分配给每个docker容器

写时复制：文件系统都是通过写时复制创建的，这就意味着文件系统时分层的、快速的，而且占用的磁盘空间更小。

日志：容器产生的STDOUT、STDERR和STDIN这些IO流都会被收集并记入日志，用来进行日志分析和故障排错

交互式shell：用户可以创建一个伪tty终端，将其连接到STDIN，为容器提供一个交互式的shell。

Docker入门

**查看docker程序是否正常工作：** sudo docker info

该命令会返回所有容器和镜像（镜像即是docker用来构建容器的“构建快”）的数量、docker使用的执行驱动和存储驱动（execution and storage driver），以及docker的基本配置。

**运行我们的第一个容器**：可以使用docker run命令创建容器

sudo docker run –i –t Ubuntu /bin/bash

参数-i标志保证容器中STD是开启的，尽管我们并没有附着到容器中、持久的标准输入是交互式shell的“半边天”

参数-t标志则是另外“半边天”，它告诉docker为要创建的容器分配一个伪tty终端。这样，新创建的容器才能提供一个交互式shell。若要在命令行下创建一个能与之进行交互的容器，而不是一个运行后台服务的容器，则这两个参数是最基本的了。

(执行过程中遇到的报错：error pulling image configuration: Get https://production.cloudflare.docker.com/registry78a7489334c85e16bf4a9e08440c3286781783015a6c43d0/data?verify=1549179284-fa%2BB1UwaM6BuZdot yet valid ，原因是系统时间不同步，执行时间同步操作即可。ntpdate time.windows.com ，如果提示ntpdate 未找到命令，则需要执行yum install ntpdate，安装命令即可)

示例中使用的是ubuntu镜像。——首先docker会检查本地是否存在Ubuntu镜像，如果本地没有该镜像，那么docker就会连接官方维护的docker hub registry，查看docker hub中是否有该镜像。Docker一旦找到该镜像，就会下载该镜像并将其保存到本地宿主机中。随后，docker在文件系统内部用这个镜像创建了一个新容器。该容器拥有自己的网络、ip地址，以及一个用来和宿主机进行通信的桥接网络接口。最后，我们告诉docker在新容器中要运行什么命令，在本例中我们在容器中运行/bin/bash命令启动了一个bash shell。当容器创建完毕之后，docker就会执行容器中的/bin/bash命令，这时就可以看到容器内的shell了。

容器可以通过exit命令退出

docker ps

默认情况下，当执行docker ps命令时，只能看到正在运行的容器。如果指定-a标志的话，那么会列出所有容器，包括正在运行和已经停止的。也可以指定-l标志，列出最后一个运行的容器，无论其正在运行还是已经停止；也可以通过—format标志，进一步控制显示哪些信息以及如何显示这些信息。

该命令的输出结果包括：id、用于创建该容器的镜像、容器最后执行的命令、创建时间已经容器的退出状态、每个容器的名称。

容器命名

Docker会为创建的每一个容器自动生成一个随机的名称。也可以通过--name标志为容器指定一个名称

sudo docker run -–name bob\_the\_container –i –t Ubuntu /bin/bash

上述命令将会创建一个名为bob\_the\_container的容器。一个合法的容器名称只能包含以下字符：小写字母a~z、大写字母A~Z、数字0~9、下划线、圆点、横线（即[a-zA-Z0-9\_.-]）

在很多docker命令中，都可以用容器名称来替代容器id。容器名称有助于分辨容器。

容器的命名必须是唯一的，如果试图创建两个名称相同的容器，则命令将会失败。如果要使用的容器名称已经存在，可以先用docker rm删除已有的同名容器后，再创建新的容器。

**重新启动已经停止的容器**

通过容器名称：

docker start bob\_the\_container

也可以用容器id来指定容器：docker start aafldje309f

也可以使用docker restart命令重新启动一个容器。这时运行不带-a标志的docker ps命令就可以看到容器已经开始运行了。

也可以通过docker create命令创建一个容器但是并不运行它。这让我们可以在自己的容器工作流中对其进行细粒度的控制。

**附着到容器上**

在docker重新启动后，可以用docker attach命令重新附着到该容器的会话上：

docker attach bob\_the\_container

也可以使用容器id。通过这个命令又重新回到了容器的bash提示符。如果退出容器的shell，容器会再次停止运行。

**创建守护式容器**

除了交互式运行的容器，也可以创建长期运行的容器。守护式容器（daemonized container）没有交互式会话，非常适合运行应用程序和服务。

sudo docker run --name daemon\_dave -d ubuntu /bin/sh -c "while true; do echo hello world; sleep 1; done"

通过-d参数，docker会将容器放到后台运行。上面还在容器要运行的命令里使用了一个while循环，会一直打印hello world直到容器或其进程停止运行。

该命令会返回一个容器id，如果执行docker ps命令，可以看到一个正在运行的容器。

**容器内部都在干些什么**

可以通过docker logs命令来获取容器的日志：docker logs daemon\_dave

可以看到while循环正向日志里打印hello world；可以通过-f参数来监控docker的日志，这与tail –f命令很相似

（tail -f等同于--follow=descriptor，根据文件描述符进行追踪，当文件改名或被删除，追踪停止）



提示：可以通过Ctr+C退出日志跟踪。