第 1 章

Docker初探

Docker是一个开源项目，诞生于2013年年初。最初，它是dotCloud公司内部的一个业余项目，基于Google公司推出的Go语言实现。后来，Docker项目加入了Linux基金会，并遵从Apache 2.0协议，项目代码在GitHub上进行维护。

本章首先讲解Docker的概念、组成与特点，然后介绍Docker的安装和配置、Docker镜像与容器的概念、操作及相关命令，最后介绍通过Docker部署Nginx服务等实战案例。

本章内容：

- Docker概述
- Docker的组成与特点
- 安装和配置Docker
- Docker镜像与容器相关操作
- 通过Docker部署Nginx服务

1.1 Docker 概述

1.1.1 Docker 是什么

Docker是PaaS提供商dotCloud开源的一个基于LXC的高级容器引擎，可以轻松地为任何应用创建一个轻量级的、可移植的、自给自足的容器。开发者可以将他们的应用以及依赖项打包到一个可移植的镜像中，然后把镜像发布到任何支持Docker的机器上运行。容器之间完全使用沙箱机制，彼此之间没有任何接口调用。因为Kubernetes也需要容器运行时，所以在学习Kubernetes之前需要先学习Docker的一些基础知识。

容器技术起源于Linux，Google LLC对Linux内核做出了很多容器相关的技术贡献。近几年来，

对容器发展影响较大的技术包括内核命名空间（Kernel Namespace）、控制组（Control Group）和联合文件系统（Union File System）。当然，其中不可忽视的是Docker的贡献。

Docker的思想来自集装箱，集装箱解决了什么问题？在一艘大船上，可以把货物规整地摆放起来，并将各种各样的货物装载到集装箱中，集装箱和集装箱之间互相隔离。这样一来，就不再需要专门运送蔬菜的船和专门运送货物的船，只要这些货物在集装箱里封装好，就可以用一艘大船把它们运走。Docker的理念与此类似，它应用程序及其依赖项一同打包到标准化的容器中，实现应用程序的轻松部署和运行。这使得应用程序可以快速、可靠地在不同的环境中移植和部署，就像集装箱在不同船只之间运输货物一样。云计算就好比一艘大型货轮，而Docker则充当集装箱的角色。

Docker自推出以来受到广泛的关注。无论是从GitHub上的代码活跃度，还是从Red Hat在RHEL 6.5中集成对Docker的支持，都可以看出其受欢迎的程度，甚至Google的Compute Engine也支持Docker在其上运行。

一款开源软件能否在商业上取得成功，很大程度上依赖三个因素：成功的用例（User Case）、活跃的社区和一个好的故事。dotCloud之家的PaaS产品建立在Docker之上，长期维护且有大量的用户，社区也十分活跃，这也是Docker成功的原因。

1.1.2　Docker 的版本

最早的Docker版本从1.0逐渐累积到1.13。在2017年3月，Docker的版本发生了变化，成为2017.03版本，同时形成了CE和EE版本。从那时起，Docker开始按照每个季度定期发布版本。

本书将以Docker-CE版本为例进行讲解。

1.1.3　学习 Docker 的方式

无论是Mac系统还是Windows系统，为了更好地学习Docker，都需要在计算机上安装虚拟机来进行学习。然而，如果条件允许，购买阿里云、腾讯云、华为云等云服务会更加方便。

因为Docker是开源产品，所以学习时离不开官方网站。读者可以参考官方网站了解更多有关Docker的知识，网址为：https://www.docker.com/。

另外，读者还可以在存放Docker镜像的Docker Hub上搜索到运行Docker所需的镜像，Docker Hub的网址为：https://hub.docker.com/。

1.2　Docker 的组成与特点

1.2.1　Docker 的组成

一个完整的Docker一般是由以下几个部分组成的：

- 客户端
- 镜像
- 容器
- 仓库

（1）客户端：这是用户与Docker交互的主要方式，它提供了命令行工具和API接口，允许用户管理Docker容器和镜像等资源。Docker客户端可以运行在任何支持Docker的平台上。

（2）容器：Docker镜像是一个只读模板，它包含一个完整的应用程序运行所需的所有内容，包括代码、运行时、库、环境变量和配置文件等。Docker镜像可以通过Dockerfile或者从Docker仓库中获取。

（3）镜像：Docker容器是从Docker镜像创建的可运行实例，它提供了一个独立的运行环境，包括文件系统、系统工具、库和运行时等。Docker容器可以启动、停止、重启、删除、暂停等，容器之间相互隔离。

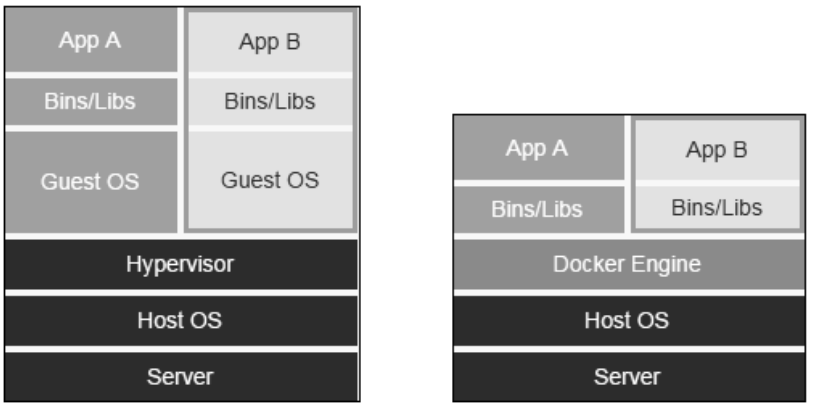
（4）仓库：Docker容器是从Docker镜像创建的可运行实例，它提供了一个独立的运行环境，包括文件系统、系统工具、库和运行时等。Docker容器可以启动、停止、重启、删除、暂停等，容器之间相互隔离。

综上所述，Docker通过将应用程序打包成镜像，再在镜像上创建容器来实现跨平台的应用程序部署。Docker还提供了丰富的CLI命令和API接口，使得用户可以方便地管理Docker容器和镜像等资源。

在Docker出现之前，人们很多时候使用虚拟机来部署应用，但Docker出现之后，使用Docker容器部署应用开始被广大用户接受。Docker部署应用与传统虚拟机部署应用的区别如下：

- Docker有着比虚拟机更少的抽象层。由于Docker不需要Hypervisor实现硬件资源虚拟化，运行在Docker容器上的程序直接使用的是实际物理机的硬件资源。因此，在CPU、内存利用率上Docker在效率上有优势。在IO设备虚拟化上，Docker的镜像管理有多种方案，比如利用AUFS文件系统或者Device Mapper实现Docker的文件管理。
- Docker利用的是宿主机的内核，而不需要Guest OS。因此，当新建一个容器时，Docker不需要和虚拟机一样重新加载一个操作系统内核。我们知道，引导、加载操作系统内核是一个比较费时费资源的过程，当新建一个虚拟机时，虚拟机软件需要加载Guest OS，这个新建过程是分钟级别的。而Docker由于直接利用宿主机的操作系统，省略了这个过程，因此新建一个Docker容器只需要几秒。另外，现代操作系统是复杂的系统，在一台物理机上新增加一个操作系统的资源开销是比较大的，因此，Docker对比虚拟机在资源消耗上占有比较大的优势。事实上，在一台物理机上，我们很容易建立成百上千的容器，而只能建立几个虚拟机。

传统虚拟化实现架构与Docker容器实现架构的对比图如图1-1所示。



传统虚拟化实现架构　　Docker容器实现架构

图 1-1　传统虚拟化实现架构与 Docker 容器实现架构对比图

1.2.2　Docker 的特点

随着传统架构向微服务架构的转型，很多公司都开始用Docker作为容器运行时，Docker除了拥有隔离应用依赖项、创建应用镜像并进行复制、创建容易分发的即启即用的应用、允许实例简单和快速地扩展、测试应用并随后销毁它们、自动化测试和持续集成及发布等优势外，还有以下特点。

1．快

Docker可以在几秒内启动容器，并且镜像的构建和分发也非常快，这极大地提高了开发、测试和运维部署代码的效率。

2．敏捷

Docker可以将应用程序及其依赖项打包成镜像，并将其部署在任何支持Docker的环境中，这使得开发人员可以更快地迭代和交付应用程序。

3．灵活

Docker容器可以轻松地在不同的平台和环境中移植和部署，而且容器之间可以相互隔离，这使得应用程序可以更加灵活地部署和扩展。

4．轻量

Docker镜像是基于分层的文件系统构建的，这意味着它是非常轻量级的，只包含必要的文件和依赖项，从而减少了部署时所需存储空间和网络带宽的使用。

5．便宜

Docker可以在单个物理服务器上运行多个容器，这大大提高了资源的利用率，从而降低了成本和复杂性。

综上所述，Docker的这些特点使得它成为一个受欢迎的容器化平台，被广泛应用于DevOps、云计算和微服务等领域。

1.3　安装和配置 Docker

通过前面的介绍，我们对Docker有了基本的认识，接下来安装一个Docker服务，用来演示Docker的具体用法。

Docker支持多种操作系统的安装运行，比如Ubuntu、CentOS、Red Hat、Debian、Fedora，甚至还支持Mac和Windows。这里我们使用一台Linux操作系统的服务器来安装，如果没有服务器，可以在自己的计算机上安装VMware Workstation，然后下载CentOS 7.6的ISO镜像，安装一个CentOS系统的虚拟机。

计算机器的配置要求如下：

- 主机IP：192.168.40.180。
- 操作系统：CentOS 7.6或者CentOS 7.9。
- 内存和CPU配置：4Gib/4vCPU。

由于CentOS系列操作系统不再提供长期支持，因此可以用Rocky Linux代替CentOS操作系统。本书中的所有实例，读者都可以基于Rocky Linux操作系统来实现。

下面介绍安装Docker的具体步骤。

1.3.1　配置主机名

执行以下命令可以配置主机名：

```
[root@localhost~]# hostnamectl set-hostname xianchaomaster1 && bash
```

该命令主要用来设置主机名，可以分为两部分：

- hostnamectl set-hostname xianchaomaster1：使用hostnamectl命令设置主机名为xianchaomaster1。
- && bash：符号&&表示如果前一个命令执行成功，则执行后面的命令，而bash是一个用于启动新的终端会话的命令，相当于打开一个新的命令终端窗口，所以这个命令将会打开一个新的命令终端窗口，并在该窗口中执行后续的命令。

Linux机器设置主机名的好处是，主机名设置之后，其他服务可以通过访问主机名找到此Docker机器。

1.3.2　关闭 Firewalld 防火墙

执行以下命令可以关闭Linux系统中的防火墙Firewalld：

```
[root@xianchaomaster1 ~]# systemctl stop firewalld && systemctl disable firewalld
```

- systemctl stop firewalld是把Firewalld服务停掉，不让它运行。
- systemctl disable firewalld是禁止Firewalld开机自启动，可以确保Firewalld一直处于禁用状态。

在CentOS 7中，有几种防火墙共存，即Firewalld、Iptables和Ebtables。默认情况下，使用Firewalld来管理Netfilter子系统，不过底层调用的命令仍然是Iptables等。相比Iptables，Firewalld的缺点在于需要为每个服务设置规则才能放行，因为默认是拒绝的。而Iptables默认允许每个服务，并且需要限制特定服务的访问。因此，关闭Firewalld防火墙可以帮助快速搭建Docker进行试验，无须后续再去放开端口。

1.3.3　关闭 SELinux

SELinux（Security Enhanced Linux，安全增强型Linux系统）是一个Linux内核模块，也是Linux的一个安全子系统。SELinux的主要作用是最大限度地减小系统中服务进程可访问的资源（最小权限原则）。

在本书的实例中，可以执行以下命令把SELinux设置成关闭状态：

```
[root@xianchaomaster1 ~]# setenforce 0
```

以上命令是临时关闭SELinux，重启服务器会自动开启。

如果要永久关闭，可以按照如下方法操作：

```
[root@xianchaomaster1 ~]# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/selinux/config
```

该命令的作用是修改/etc/selinux/config文件中的SELINUX=enforcing为SELINUX=disabled，使用了sed工具进行替换。

- sed：用于对文本内容进行替换。
- -i：表示对原始文件直接进行编辑，即直接修改文件内容。
- s/SELINUX=enforcing/SELINUX=disabled/g：s指的是替换操作，其中SELINUX=enforcing表示查找文件中 SELINUX=enforcing这个字符串，将其替换为SELINUX=disabled；g则表示进行全局替换。

因此，该命令的作用是将SELinux安全机制从强制模式改为停用模式。注意，修改该文件需要root权限。

修改SELinux配置文件之后，重启机器，SELinux才能永久生效。

要验证SELinux是否关闭，可以执行如下命令：

```
[root@xianchaomaster1 ~]# getenforce
```

该命令用来获取SELinux安全机制的当前状态。getenforce是一个Linux命令，用于获取SELinux的执行模式。该命令仅在SELinux已启用且运行时状态下使用。如果SELinux未启用，则输出Permissive，表示 SELinux未强制执行策略，但仍会生成警告和日志；如果SELinux正在运行且执行模式为非强制，则输出Enforcing。

如果该命令显示Disabled，那么表示SELinux关闭成功。

1.3.4　配置时间同步

为了避免主机因长期运行导致的时间偏差，进行时间同步（Synchronize）的工作是非常必要的。在Linux系统下，一般使用ntpdate命令来同步不同机器的时间，可以分别执行以下命令：

```
[root@xianchaomaster1 ~]# yum install -y ntpdate
```

yum install -y ntpdate是安装时间同步的命令。

```
[root@xianchaomaster1 ~]# ntpdate cn.pool.ntp.org
```

ntpdate cn.pool.ntp.org表示和网络源同步。

如果是内网机器，那么可以单独搭建专用的时间同步服务器，与内网服务器进行时间同步。

1.3.5　编写计划任务

对时间同步做计划任务，每一个小时同步一次，可以让时间更精准。执行以下命令：

```
[root@xianchaomaster1 ~]# crontab -e
```

写入如下内容：

```
* * * * * /usr/sbin/ntpdate   cn.pool.ntp.org
```

这是一个Crontab的定时任务命令，表示每分钟执行一次ntpdate命令，将本地时间同步到cn.pool.ntp.org所指定的时间服务器上。

- *：Crontab任务的时间设置，依次为分、时、日、月、星期几。*表示每个时间单位都匹配，即每分钟执行一次ntpdate命令。
- /usr/sbin/ntpdate：表示ntpdate命令的完整路径。
- cn.pool.ntp.org：表示时间服务器的地址，通过该服务器同步本地时间。

1.3.6　重启 crond 服务使计划任务生效

执行以下命令：

```
[root@xianchaomaster1 ~]# systemctl restart crond
```

该命令的作用是重启crond服务。crond是一个系统服务，它的主要功能是在规定的时间内运行预定的系统任务，比如切割日志、备份数据库等，它在Linux系统中非常重要。

- systemctl：用于控制systemd系统和服务管理器。
- restart：用于重启服务。
- crond：表示要重启的服务名称，即crond守护进程服务。该服务通常由系统自动启动并在后台运行，用于管理系统的crontab和at任务。重新启动crond服务可以确保其当前配置的有效性，并重新加载所有已更改的crontab配置文件。

1.3.7　安装基础软件包

在CentOS操作系统的默认最小版本中，没有集成我们所需的很多软件包，因此需要手动安装它们。可以执行以下命令安装所需的一些基础软件包：

```
[root@xianchaomaster1 ~]# yum install -y  wget net-tools nfs-utils lrzsz gcc gcc-c++
make cmake libxml2-devel openssl-devel curl curl-devel unzip sudo ntp libaio-devel
ncurses-devel autoconf automake zlib-devel python-devel epel-release openssh-server socat
ipvsadm conntrack
```

该命令的作用是通过yum包管理器安装一系列实验和平时工作经常用到的基础软件包和依赖项，主要有以下几种：

（1）wget：用来从指定的URL下载文件的命令行工具。wget非常稳定，如果由于网络原因下载失败，那么wget会不断尝试，直到整个文件下载完毕。

（2）net-tools：包含一系列用于网络监控和分析的命令行工具，比如ifconfig、route、netstat等命令。

（3）nfs-utils：包含NFS网络文件系统的相关工具和文件。

（4）lrzsz：用于在Linux系统和其他Unix操作系统之间进行文件传输和通信的工具。

（5）gcc、gcc-c++、make、cmake：分别是GNU C/C++编译器、GNU Make工具和CMake构建工具，用于开发和编译程序。

（6）libxml2-devel、openssl-devel、curl-devel、libaio-devel、ncurses-devel、zlib-devel、python-devel：在编译和安装一些应用程序时需要的库和头文件。

（7）unzip：用于解压缩ZIP压缩文件的命令行工具。

（8）sudo：允许非root用户执行以 root 权限运行的命令。

（9）ntp：用于同步系统时间的服务。

（10）epel-release：安装该软件包后，可以从 Extra Packages for Enterprise Linux（EPEL）软件源中获取额外的软件包。

（11）openssh-server：SSH服务器，用于远程登录和执行命令。

（12）socat：多功能网络工具，可以建立各种类型的网络连接。

（13）ipvsadm：IP服务负载均衡器。一般配置了LVS后都要安装ipvsadm，用来查看LVS的状态以及进行问题排查，比如节点分配情况以及连接数等，ipvsadm是IPVS的管理器，需要yum安装。

（14）conntrack：网络连接跟踪器，用于找出网络问题或进行性能调优。

1.3.8　安装 Docker-CE

CentOS系统默认的yum源不能安装Docker，因此需要配置国内阿里云的yum源以安装Docker-CE。通过配置国内阿里云的yum源，安装Docker的速度会快很多。

执行以下命令可以配置阿里云：

```
    [root@xianchaomaster1 ~]# yum-config-manager -add-repo http://mirrors.aliyun.com/
Docker-ce/linux/centos/Docker-ce.repo
```

该命令将阿里云的Docker-CE仓库添加到yum软件包管理器中，以便在CentOS操作系统上安装Docker-CE软件。

- yum-config-manager是从一个.repo文件加载yum仓库配置的命令，可以添加、启用、禁用或删除yum仓库。
- --add-repo选项告诉yum，在/etc/yum.repos.d/目录下添加一个新的.repo文件，并从指定的URL中获取.repo文件。
- http://mirrors.aliyun.com/Docker-ce/linux/centos/Docker-ce.repo是第三方仓库URL，用来获取Docker-CE仓库的.repo文件。.repo文件提供了软件仓库的详细信息和文件列表。

运行完这个命令后，读者就可以使用yum命令来安装、更新、卸载Docker-CE软件了。

我们开始安装Docker-CE。下面介绍具体的安装步骤。

1. 安装Docker-CE

```
[root@xianchaomaster1 ~]# yum install docker-ce  -y
```

该命令通过yum软件包管理器在CentOS操作系统上自动安装Docker社区版，并在安装过程中自动回答yes，以便在不需要人为干预的情况下完成安装。

- yum是CentOS操作系统中的包管理器，可以从yum仓库下载和安装软件包。
- install选项告诉yum下载和安装指定的软件包。
- docker-ce是要安装的软件包的名称，表示Docker社区版。
- -y选项告诉yum在安装过程中自动回答yes。

2. 启动Docker服务

```
[root@xianchaomaster1 ~]# systemctl start docker && systemctl enable docker
```

该命令以root用户权限启动Docker服务，并将它设置为随系统开机自动启动。这样，每次系统启动时，Docker服务将自动启动，无须手动操作。

- systemctl是用于在systemd系统中管理系统服务的命令。
- start docker命令以root用户权限启动Docker服务。start选项表示启动服务。
- &&连接符将两个命令一起执行，即只有在第一个命令执行完毕且成功之后才会执行第二个命令。
- systemctl enable docker将Docker服务设置为系统开机启动。enable选项表示启用服务。

3. 查看Docker服务的状态

```
[root@xianchaomaster1 ~]# systemctl status docker
```

该命令查询Docker服务的状态信息，包括服务名称、是否加载、是否开机启动、是否正在运行以及服务运行时间等，其中Docker为服务的名称。

输出的内容如下：

```
   Docker.service - Docker Application Container Engine（提示Docker服务名称为Docker
Application Container Engine）
      Loaded: loaded (/usr/lib/systemd/system/Docker.service; enabled; vendor preset:
disabled)
      （Loaded: loaded提示Docker服务已经被加载。enabled提示Docker服务已被设置为开机启动）
   Active: active (running) since Thu 2021-07-01 21:29:18 CST; 30s ago
   （active (running)提示Docker服务正在运行
   since Thu 2021-07-01 21:29:18 CST; 30s ago提示Docker服务从此时间起运行约30秒）
       Docs: https://docs.Docker.com（提示查看Docker官方文档）
```

如果从上述结果中看到active (running)，那么表示Docker正常运行。

4. 查看Docker的版本信息

如果想要查看Docker的版本信息，那么可以执行以下命令：

```
[root@xianchaomaster1 ~]# Docker version
```

该命令将输出Docker Client和Docker Server的版本信息，包括版本号、构建信息、API版本等。

1.3.9　修改内核参数

在Linux内核中加载br_netfilter模块，执行以下命令：

```
[root@xianchaomaster1 ~]# modprobe br_netfilter
```

在安装Docker时，通常需要加载br_netfilter内核模块以确保Docker正常工作。这是因为Docker在运行时需要创建一个Linux网络桥接设备（Bridge Device），以便将容器内部的网络连接到宿主机上。而br_netfilter模块提供了必要的网络过滤功能，能够使Linux内核对网络数据包进行转发和过滤。

通过执行modprobe br_netfilter命令可以将该内核模块加载到系统中。该命令会在系统中查找并加载br_netfilter模块，使得Docker能够正常运行。在加载后，Docker会自动创建和配置网络设备，以便将容器的网络连接到宿主机上，从而实现容器间以及容器与外部网络的通信。

需要注意的是，有些Linux发行版中，br_netfilter模块可能未被默认加载。在这种情况下，必须手动加载该模块才能使 Docker 正常工作。因此，在安装 Docker 之前，需要确保该模块已经被正确加载到系统中。

然后，在/etc/sysctl.d/下创建docker.conf文件，把要放开的内核参数写进去。

```
[root@xianchaomaster1 ~]# cat > /etc/sysctl.d/docker.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

上述命令可以将一段文本内容写入/etc/sysctl.d/docker.conf文件中，用于配置Docker相关的系统内核参数，并且使用EOF来标记文本的结束。

- cat命令用于读取文件内容并打印到标准输出。
- >表示重定向符号，将输出的结果写到指定文件中。
- /etc/sysctl.d/docker.conf是要写入的文件路径。
- <<EOF是一个输入重定向符号，表示后面的文本作为输入内容。

说明如下：

（1）net.ipv4.ip_forward是Linux内核的一个配置项，用于控制是否开启IP转发功能。当该配置项被设置为1时，表示开启IP转发功能；当设置为0时，表示关闭该功能。

IP转发功能是指当Linux系统接收到一条数据包时，会根据目标IP地址的不同，将数据包转发到不同的网络接口上。如果IP转发功能被关闭，那么数据包只会在本机内部进行转发，无法被传递到其他网络中。

在某些场景下，需要开启IP转发功能。例如，当Linux系统作为路由器或NAT网关使用时，必须开启IP转发功能才能实现数据包的正常转发。此外，一些网络应用程序（如VPN、负载均衡等）也需要使用IP转发功能来进行数据包的路由和转发。

在安装Docker并使用它运行容器时，需要确保Linux系统开启了IP转发功能（net.ipv4.ip_forward=1），以便实现容器内部与外部网络的通信。

Docker的容器是隔离的虚拟环境，其网络栈是完全独立于宿主机的。为了让容器内部的应用程序可以与外部网络进行通信，需要进行网络地址转换（Network Address Translation，NAT）操作，将容器内部的IP地址映射到宿主机的IP地址上。这就需要开启IP转发功能，以便Linux系统将接收到的数据包转发到正确的容器中。

需要注意的是，开启IP转发功能可能会增加系统的安全风险。如果未经充分考虑就开启了该功能，可能会导致系统被攻击者利用并实施恶意行为。因此，在开启IP转发功能之前，需要进行充分的安全评估，并采取适当的措施来保护系统安全。

（2）net.bridge.bridge-nf-call-ip6tables和net.bridge.bridge-nf-call-iptables是两个Linux内核的配置项，用于开启Linux内核网络桥接功能的防火墙规则。

Linux系统提供了网络桥接功能，允许多个网络接口连接到同一个局域网上，从而实现网络设备之间的数据包转发。Docker通过使用Linux的网络命名空间和桥接技术，将多个容器连接到同一个虚拟局域网中，以便实现容器之间的通信。

为了保证容器的网络安全，Docker强制要求开启网络桥接的防火墙规则。具体来说，需要将上述两个配置项设置为1，以开启IPv4和IPv6的网络桥接防火墙规则。这样可以确保容器之间的通信只能通过Docker的桥接网络进行，而不能直接通过主机的网络接口进行。

需要注意的是，有时候开启网络桥接的防火墙规则会与主机上的其他网络服务冲突，导致网络连接失败。因此，在开启这些规则之前，需要进行充分的测试，并根据实际情况进行调整和优化。

接着执行以下命令，使上面修改的内核参数生效：

```
[root@xianchaomaster1 ~]# sysctl -p /etc/sysctl.d/docker.conf
```

这个命令将/etc/sysctl.d/docker.conf文件中的系统内核参数加载到当前运行的Linux系统中，从而应用这些参数配置。

- sysctl命令用于在运行时改变Linux系统内核的运行参数。
- -p参数表示从指定文件中加载sysctl参数。
- /etc/sysctl.d/docker.conf是要加载的sysctl配置文件路径。

最后，修改内核参数后需要重启Docker守护进程，才能使配置生效。

Docker 的守护进程会读取 net.ipv4.ip_forward 、 net.bridge.bridge-nf-call-ip6tables 和 net.bridge.bridge-nf-call-iptable这些参数。如果在运行Docker容器时修改了这些参数，那么修改将不会立即生效，因为Docker守护进程已经启动，并且已经使用旧的参数配置了网络桥接的防火墙规则。此时需要重启Docker守护进程，以便使新的参数配置生效。执行以下命令重启docker服务：

```
[root@xianchaomaster1 ~]# systemctl restart docker
```

这个命令可以重启docker服务，systemctl命令用于管理systemd系统和服务管理器。

- restart子命令用于重启指定的服务。
- docker是要重启的服务名。

1.3.10　配置 Docker 镜像加速器

国内从Docker Hub拉取镜像有时会遇到困难，此时可以配置镜像加速器。Docker官方和国内很多云服务商都提供了国内加速器服务，举例如下。

- 科大镜像站点：https://Docker.mirrors.ustc.edu.cn/。
- 网易163镜像站点：https://hub-mirror.c.163.com/。
- 阿里云镜像站点：https://<你的阿里云镜像仓库ID>.mirror.aliyuncs.com。

镜像加速器配置的具体步骤如下：

修改/etc/docker/daemon.json文件，输入如下内容：

```
{
 "registry-mirrors":["https://y8y6vosv.mirror.aliyuncs.com",
"https://registry.Docker-cn.com","https://Docker.mirrors.ustc.edu.cn",
"https://Dockerhub.azk8s.cn","http://hub-mirror.c.163.com"]
}
```

这是一个JSON格式的配置文件，其中包含一组Docker镜像仓库的镜像地址。其中registry-mirrors是键名，表示Docker镜像仓库的镜像地址列表；["https://y8y6vosv.mirror.aliyuncs.com", "https://registry.Docker-cn.com", "https://Docker.mirrors.ustc.edu.cn", "https://Dockerhub.azk8s.cn",

"http://hub-mirror.c.163.com"]是一个包含多个镜像地址的JSON数组。

每个镜像地址都是一个字符串，表示一个Docker镜像仓库的镜像地址。

这些镜像地址分别是https://y8y6vosv.mirror.aliyuncs.com、https://registry.Docker-cn.com、https://Docker.mirrors.ustc.edu.cn、https://Dockerhub.azk8s.cn、http://hub-mirror.c.163.com。

这些镜像地址都是用于加速Docker从相关的站点下载镜像的加速器地址，用户可以根据自己的需要进行选择和配置。

执行以下命令重启Docker让配置生效：

```
[root@xianchaomaster1 ~]# systemctl restart docker
```

通过上面的配置，从Docker Hub、阿里云、科大、163镜像站点拉取镜像会加快镜像的拉取速度。

1.4　Docker 镜像与容器

Docker安装成功之后，需要通过Docker部署容器，Docker 运行容器前需要本地存在对应的镜像，如果本地不存在该镜像，Docker 会从镜像仓库下载该镜像。本节将深入讲解Docker镜像和容器的常用命令。

1.4.1　Docker 镜像

Docker包含3个核心部分：镜像、容器和仓库。其中镜像是Docker运行容器的前提，可以将其理解为VM模板。镜像由多层组成，每层叠加之后形成一个独立的对象，内部包含精简的操作系统和应用运行所需的文件和依赖项。

容器是基于镜像创建的可运行实例，每个容器都是相互隔离的运行环境，拥有自己的文件系统、网络、进程空间等。容器是轻量级的，启动速度快，资源占用较少，因此在很多场景下比传统虚拟机更为适用。

仓库是存放镜像的场所，是Docker中用来管理镜像的中央存储服务。Docker官方提供了Docker Hub，用户可以在其中分享、获取和管理Docker镜像。此外，用户还可以在私有环境中搭建自己的Docker镜像仓库，以便管理和分享自己的镜像。

在Docker中，镜像是不可修改的，当需要对一个容器进行修改时，可以在基于该镜像的容器中进行修改并保存为新的镜像。因此，Docker的镜像和容器是可重复使用的，而且非常适合进行应用程序的部署、测试和交付。

一旦安装了Docker，就可以开始部署容器了。在运行容器之前，确保本地存在对应的镜像，如果本地不存在该镜像，Docker就会从镜像仓库下载该镜像。以下是Docker镜像和容器常用命令的详细说明。

1. 从 Docker Hub 上查找镜像

执行以下命令可以从Docker Hub上查找镜像：

```
[root@xianchaomaster1 ~]# docker search centos
```

在执行命令后，Docker客户端会向Docker Hub发送搜索请求，搜索所有符合条件的镜像，并将搜索结果打印到终端上。搜索结果包含各种镜像的信息，例如镜像名称、标签、描述、星级评价等。

- docker是Docker客户端的命令名。
- search是Docker命令的一个子命令，用于在Docker Hub上搜索镜像仓库。
- centos是要搜索的镜像的名称，表示搜索所有名称中包含centos的镜像。

显示结果如图1-2所示。

```
NAME                              DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
centos                            The official build of CentOS.                   7093      [OK]
```

图 1-2　搜索镜像的显示结果

图1-2的显示结果说明如下：

- NAME：镜像的名称。
- DESCRIPTION：镜像的描述。
- stars：类似于GitHub里面的star，表示点赞、喜欢的意思。
- OFFICIAL：是否由Docker官方发布。
- AUTOMATED：自动构建。

2. 从 Docker Hub 上拉取镜像

创建镜像有很多方法，用户可以从Docker Hub、阿里云镜像仓库等站点获取已有的镜像，也可以利用本地文件系统创建镜像。

如果不指定镜像的下载地址，那么默认从Docker Hub下载镜像。

1）从 Docker Hub 下载镜像

执行以下命令从Docker Hub站点下载镜像：

```
[root@xianchaomaster1 ~]# docker pull centos
```

这个命令使用Docker客户端从Docker Hub下载CentOS镜像到本地。

- docker是Docker客户端的命令名。

- pull是Docker命令的一个子命令，用于从Docker Hub下载指定的镜像。
- centos是要下载的镜像的名称，表示从Docker Hub上下载新版本的CentOS镜像。

在执行命令后，Docker客户端会向Docker Hub发送下载请求，下载CentOS镜像到本地，并在终端显示下载进度和状态。下载完成后，就可以使用Docker客户端来运行这个CentOS镜像了。

2）查看本地已经存在的镜像

执行以下命令可以查看本地已经存在的镜像：

```
[root@xianchaomaster1 ~]# docker images
```

这个命令可以帮助用户快速查看本地已经存在的Docker镜像。

- docker是Docker客户端的命令名。
- images是Docker命令的一个子命令，用于列出本地所有已经下载的Docker镜像。

在执行命令后，Docker客户端会查询本地的镜像仓库，并将已经下载的所有镜像打印到终端上，显示的信息包括：镜像ID、镜像名称、镜像标签、镜像大小等。

3）把镜像做成离线文件

如果Docker Hub或者其他镜像站点不能下载镜像，或者内网机器不能联网，则需要把镜像做成离线压缩包，然后上传到安装Docker的其他机器上。下面的命令用来制作离线镜像文件：

```
[root@xianchaomaster1 ~]# docker save -o centos.tar.gz centos
```

这个命令使用Docker客户端将本地的CentOS镜像保存到一个tar归档文件中，其中：

- docker是Docker客户端的命令名。
- save是Docker命令的一个子命令，用于将本地的一个或多个Docker镜像保存到一个文件中。
- -o centos.tar.gz是保存选项，表示将保存的镜像输出到一个名为centos.tar.gz的文件中。
- centos是要保存的镜像的名称，表示将保存本地新版本的CentOS镜像。

在执行命令后，Docker客户端会检查本地是否存在指定的镜像，如果存在的话，就将这个镜像保存到指定的文件中。

保存完成后，用户就可以通过将这个文件迁移到其他机器上，使用Docker客户端的load命令将这个镜像加载到目标机器的本地镜像仓库中。

4）手动解压离线镜像文件

将离线的镜像文件手动上传到已安装Docker的机器后，通过docker load解压出来即可，执行以下命令进行解压：

```
[root@xianchaomaster1 ~]# docker load -i centos.tar.gz
```

这个命令使用Docker客户端将一个包含CentOS镜像的文件加载到本地的Docker镜像仓库中。

- load是Docker命令的一个子命令，用于从一个文件中加载Docker镜像到本地的Docker镜像仓库中。
- -i centos.tar.gz是加载选项，表示从名为centos.tar.gz的文件中加载Docker镜像。

在执行命令后，Docker客户端会检查当前系统中是否已经存在指定的归档文件，如果存在的话，就将其中保存的镜像加载到本地的Docker镜像仓库中。

加载完成后，用户就可以通过执行docker images命令来查看本地的Docker镜像仓库中是否已经存在这个CentOS镜像。

5）删除镜像

对于本地镜像，如果不再需要，可以执行以下命令将其删除：

```
[root@xianchaomaster1 ~]# Docker rmi -f centos
```

这个命令使用Docker客户端删除本地的CentOS镜像。

- rmi是Docker命令的一个子命令，用于删除本地的一个或多个Docker镜像。
- -f是删除选项，表示强制删除指定的镜像，即使这个镜像正在被使用。
- centos是要删除的镜像的名称，表示删除本地新版本的CentOS镜像。

在执行命令后，Docker客户端会检查本地是否存在指定的镜像，如果存在的话，就会将这个镜像删除。

如果这个镜像正在被使用，那么需要使用-f 选项来强制删除这个镜像。

删除完成后，这个镜像就会从本地的Docker镜像仓库中移除，无法再被使用。

1.4.2　Docker 容器

Docker的主要目标是在任何地方创建、发布和运行应用程序，也就是通过对应用组件的封装、分发、部署、运行等生命周期的管理，将应用运行在Docker容器上，而Docker容器可以运行在任何操作系统上，这就实现了跨平台和跨服务器。只需要配置一次环境，换到别的机器上就可以一键部署，大大简化了操作。

下面介绍Docker容器的基本操作。

1. 以交互式启动并进入容器

以交互式启动容器意味着在容器内部启动一个shell终端，并将该终端连接到当前终端会话。

这样就可以直接在容器内部执行命令，查看容器内部的状态、文件等信息，并与容器内部进行交互。交互式启动容器通常使用-i和-t选项，即-it，以实现与容器的交互。

如果要以交互式启动并进入容器中，那么需要执行以下命令：

```
[root@xianchaomaster1 ~]# docker run --name=hello -it centos /bin/bash
[root@09c4933b5cd7 /]#
```

该命令表示在本地主机上以交互模式（-it）运行一个名为hello的容器，并使用CentOS镜像（如果本地不存在该镜像，那么Docker会从Docker Hub下载该镜像）在容器中启动一个Bash终端（/bin/bash）。使用docker run运行并创建容器的相关参数说明如下：

- --name=hello：指定容器的名字为hello。
- -i：交互式进入容器，一般与-t连用。
- -t：分配一个伪tty，一般与-i连用。
- centos：启动Docker需要的镜像。
- /bin/bash：说明你的shell类型为bash，bash shell是最常用的一种shell，是大多数Linux发行版默认的shell。此外，还有C shell等其他shell。

在Bash shell提示符[root@09c4933b5cd7 /]#后输入exit，可退出容器，如：

```
[root@09c4933b5cd7 /]#exit
```

退出之后，容器会停止，不再在前台运行。

2. 以守护进程方式启动容器

以守护进程方式启动容器意味着容器在后台运行，并且不会将shell终端连接到容器内部。容器以守护进程方式运行后，可以继续在主机上执行其他任务，而容器会在后台持续运行，直到被停止或删除。守护进程方式启动容器通常使用-d选项，如果要以守护进程方式启动容器，那么需要执行以下命令：

```
[root@xianchaomaster1 ~]# docker run --name=hello1 -td centos
```

- --name=hello1：指定容器的名称为hello1。
- -td：指定以后台守护进程方式运行容器，并分配一个虚拟终端。
- centos：指定使用CentOS镜像启动容器。

执行该命令后，命令行会立即返回容器的唯一标识符（Container ID），表示容器已经在后台启动。可以使用docker ps命令查看当前正在运行的容器，其中包括hello1容器。

3. 进入 Docker 容器

在使用Docker创建容器之后，就可以进入容器进行各种操作，进入Docker容器有多种方式，

这里介绍常用的进入Docker容器的方法——使用docker exec进入容器。docker exec是Docker命令中的一个子命令，用于在运行中的容器中执行命令，语法如下：

```
docker exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

- [OPTIONS]选项参数如下：
  - -d：指定容器在后台运行。
  - -i：保持 STDIN 打开，即使未连接到终端。
  - -t：分配一个伪终端（Pseudo-TTY）。
- CONTAINER参数指定要执行命令的容器名称或容器 ID。
- COMMAND参数指定要在容器中执行的命令，可以是任何可执行命令或 shell命令。在执行命令时，Docker 会在容器内部创建一个新进程，并在其中运行指定的命令。

（1）使用docker exec命令进入容器，执行以下命令查看docker exec帮助命令：

```
[root@xianchaomaster1 ~]# docker exec  --help
```

显示结果如图1-3所示。

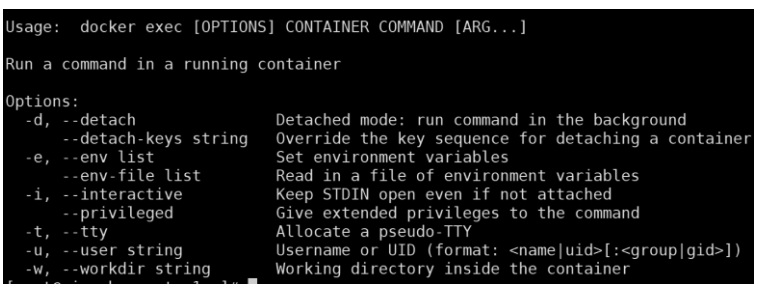


图 1-3　Docker exec 帮助命令显示的信息

上述命令显示结果说明如下：

- -d, --detach：以后台模式运行容器。
- --detach-keys string：重写分离容器的键序列。
- -e, --env list：设置环境变量。
- --env-file list：从文件中读取环境变量。
- -i, --interactive：即使未附加，也保持STDIN打开。
- --privileged：给命令扩展特权。
- -t, --tty：伪TTY。
- -u, --user string：用户名或UID（格式："<name|uid>[:<group|gid>]"）。
- -w, --workdir string：容器内的工作目录。

（2）进入容器。执行以下命令进入容器：

```
[root@xianchaomaster1 ~]# docker exec -it hello1 /bin/bash
```

该命令在已经运行的名为hello1的容器中启动一个Bash终端，并且使用交互模式进行连接。

- -i：交互模式。
- -t：分配伪终端。
- hello1：指定要进入的容器名称或容器ID。

4. 查看正在运行的容器

执行以下命令查看正在运行的容器：

```
[root@xianchaomaster1 ~]# docker ps
```

该命令将会列出当前正在运行的所有容器。

显示如下：

```
CONTAINER ID    IMAGE  CREATED  STATUS    NAMES
677f1b3a1f2d    centos     7 days ago   Up 7 days   hello1
```

上述显示结果说明如下：

- 677f1b3a1f2d：容器的ID。每个容器都有一个唯一的ID。
- CentOS：容器所使用的镜像名称，本例中使用的是CentOS镜像。
- hello1：容器的名称，由用户指定或自动生成。

docker ps命令只能查看正常运行的容器，参数hello1是容器的名字，如果想要查看所有的容器，包括正常运行和退出的容器，可以使用docker ps –a命令。

5. 停止容器

如果容器运行一段时间，任务完成后，想要停止容器，可使用如下命令：

```
[root@xianchaomaster1 ~]#docker stop hello1
```

该命令用于停止运行名为hello1的容器。

- docker stop：停止运行中的容器。
- hello1：要停止的容器名称或容器ID。

6. 启动已经停止的容器

如果容器已经停止了，那么可以通过docker start命令再次启动容器。

```
[root@xianchaomaster1 ~]#docker start hello1
```

该命令启动已经停止的名为hello1的容器。

- docker start：启动已经停止的容器。

7. 删除容器

如果要删容器，那么可以执行以下命令：

```
[root@xianchaomaster1 ~]# docker rm -f hello1
```

该命令强制删除名为hello1的容器，即使该容器正在运行中。

- docker rm：删除一个或多个容器。
- -f：强制删除容器，即使容器正在运行中也会被删除。
- hello1：指定要删除的容器名称或容器ID。

8. 查看 Docker 帮助命令

Docker的命令很多，可以通过如下方法查看Docker支持哪些命令：

```
[root@xianchaomaster1 ~]# docker  --help
```

显示如下：

```
Usage:  Docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
     --config string     Location of client config files (default
                         "/root/.Docker")
  -c, --context string   Name of the context to use to connect to the
                         daemon (overrides DOCKER_HOST env var and
                         default context set with "Docker context use")
  -D, --debug            Enable debug mode
  -H, --host list        Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                         ("debug"|"info"|"warn"|"error"|"fatal")
                         (default "info")
     --tls               Use TLS; implied by --tlsverify
     --tlscacert string  Trust certs signed only by this CA (default
                         "/root/.Docker/ca.pem")
     --tlscert string    Path to TLS certificate file (default
                         "/root/.Docker/cert.pem")
     --tlskey string     Path to TLS key file (default
                         "/root/.Docker/key.pem")
```

```
      --tlsverify         Use TLS and verify the remote
    -v, --version         Print version information and quit

  Management Commands:
    app*           Docker App (Docker Inc., v0.9.1-beta3)
    builder        Manage builds
    buildx*        Build with BuildKit (Docker Inc., v0.6.3-Docker)
    config         Manage Docker configs
    container      Manage containers
    context        Manage contexts
    image          Manage images
    manifest       Manage Docker image manifests and manifest lists
    network        Manage networks
    node           Manage Swarm nodes
    plugin         Manage plugins
    scan*          Docker Scan (Docker Inc., v0.9.0)
    secret         Manage Docker secrets
    service        Manage services
    stack          Manage Docker stacks
    swarm          Manage Swarm
    system         Manage Docker
    trust          Manage trust on Docker images
    volume         Manage volumes

  Commands:
    attach         Attach local standard input, output, and error streams to a running
container
    build          Build an image from a Dockerfile
    commit         Create a new image from a container's changes
    cp             Copy files/folders between a container and the local filesystem
    create         Create a new container
    diff           Inspect changes to files or directories on a container's filesystem
    events         Get real time events from the server
    exec           Run a command in a running container
    export         Export a container's filesystem as a tar archive
    history        Show the history of an image
    images         List images
    import         Import the contents from a tarball to create a filesystem image
    info           Display system-wide information
    inspect        Return low-level information on Docker objects
    kill           Kill one or more running containers
    load           Load an image from a tar archive or STDIN
    login          Log in to a Docker registry
    logout         Log out from a Docker registry
    logs           Fetch the logs of a container
    pause          Pause all processes within one or more containers
    port           List port mappings or a specific mapping for the container
    ps             List containers
    pull           Pull an image or a repository from a registry
    push           Push an image or a repository to a registry
```

```
  rename          Rename a container
  restart         Restart one or more containers
  rm              Remove one or more containers
  rmi             Remove one or more images
  run             Run a command in a new container
  save            Save one or more images to a tar archive (streamed to STDOUT by default)
  search          Search the Docker Hub for images
  start           Start one or more stopped containers
  stats           Display a live stream of container(s) resource usage statistics
  stop            Stop one or more running containers
  tag             Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top             Display the running processes of a container
  unpause         Unpause all processes within one or more containers
  update          Update configuration of one or more containers
  version         Show the Docker version information
  wait            Block until one or more containers stop, then print their exit codes
```

上述这些参数可以用来配置Docker的客户端和管理Docker中的容器、镜像、网络等对象。以下是这些参数的解释说明。

- --config string：指定客户端配置文件的位置，默认为“/root/.docker”。
- -c, --context string：指定连接Docker守护进程时使用的上下文名称，覆盖DOCKER_HOST环境变量和使用docker context use设置的默认上下文。
- -D, --debug：启用调试模式。
- -H, --host list：连接Docker守护进程的套接字地址。
- -l, --log-level string: 设置日志级别（可选值为debug、info、warn、error和fatal，默认为info）。
- --tls：使用TLS安全连接（--tlsverify 选项会隐式启用此选项）。
- --tlscacert string：指定只信任该CA签名的证书文件路径，默认为“/root/.docker/ca.pem”。
- --tlscert string：指定TLS证书文件路径，默认为“/root/.docker/cert.pem”。
- --tlskey string：指定TLS密钥文件路径，默认为“/root/.docker/key.pem”。
- --tlsverify：使用TLS安全连接，并验证远程服务器的身份。
- -v, --version：显示Docker的版本信息。

此外，该命令还包含一系列用于管理 Docker容器、镜像、网络等对象的子命令，例如：

- docker container：管理容器。
- docker image：管理镜像。
- docker network：管理网络。
- docker volume：管理数据卷。

每个子命令还有自己的选项和参数，可以使用“docker <子命令> --help”查看帮助信息。

1.5　案例：通过 Docker 部署 Nginx 服务

前面我们学习了Docker的基本知识和常见命令，本节通过具体的案例来加深对Docker的理解。假设公司有需求，Nginx服务不想部署在物理机上，而是需要通过Docker部署Nginx服务，根据我们现在掌握的知识，可以基于CentOS镜像启动Docker容器，进入Docker容器，然后安装Nginx服务，下面介绍具体的实现方法。

1.5.1　基于 CentOS 镜像运行一个 Docker 容器

首先执行以下命令运行一个Docker容器：

```
[root@xianchaomaster1 ~]# docker run --name nginx -p  80:80 -itd centos
```

以上命令会在后台以交互模式基于CentOS镜像启动一个容器，容器名字是nginx，并将容器的80端口映射到主机的80端口。

- docker run：启动一个新的容器。
- --name nginx：给容器命名为nginx。
- -p 80:80：将容器的80端口映射到主机的80端口。
- -itd：以交互模式启动容器，并在后台运行。
- centos：使用CentOS镜像启动容器。

1.5.2　查看 Docker 容器是否正常运行

执行以下命令查看Docker容器是否正常运行：

```
[root@xianchaomaster1 ~]# docker ps | grep nginx
```

该命令表示在正在运行的容器中查找名称包含nginx的容器，并显示其信息。

- docker ps：列出正在运行的容器。
- |：管道符号，将 docker ps命令的输出结果传递给grep命令。
- grep nginx：在docker ps命令的输出结果中查找包nginx的行，并将其显示出来。

显示正在运行的Docker容器的信息如下：

```
ecfa046e9681   centos                                                "/bin/bash"
5 seconds ago   Up 4 seconds   0.0.0.0:80->80/tcp, :::49153->80/tcp   nginx
```

说明如下：

- ecfa046e9681：容器的ID。

- centos：容器使用的镜像名称，即使用CentOS镜像启动的容器。
- /bin/bash：容器启动时执行的命令，即在容器中运行 bash。
- 5 seconds ago：容器启动的时间，即容器创建的时间距离现在的时间。
- Up 4 seconds：容器运行的时间，即容器启动后运行的时间。
- 0.0.0.0:80->80/tcp, :::80->80/tcp：容器的端口映射信息，将容器的80端口映射到主机的80端口。
- nginx：容器的名称。

以上显示结果表明Docker正在运行。

1.5.3　在 Docker 中安装 Nginx 容器

首先，执行以下命令进入正在运行的名为nginx的容器，并在容器中启动一个新的Bash终端：

```
[root@xianchaomaster1 ~]#docker exec -it nginx /bin/bash
```

- docker exec：在运行的容器中执行命令。
- -it：以交互模式运行容器中的命令，并分配一个伪终端。
- nginx：要执行命令的容器名称。
- /bin/bash：在容器中要运行的命令，即启动一个新的Bash终端。

接着，在容器中执行以下命令查看容器的IP：

```
[root@ecfa046e9681]# ip addr
```

- ip addr：显示网络接口信息的命令，包括每个接口的IP地址、MAC地址和网络状态等。此命令执行后，会显示容器中所有的网络接口信息。

显示如下：

```
    1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
           valid_lft forever preferred_lft forever
    2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
        link/ipip 0.0.0.0 brd 0.0.0.0
    21: eth0@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default
        link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0
           valid_lft forever preferred_lft forever
```

通过上述结果可以看到容器的IP是172.17.0.3。

1.5.4　在 Docker 容器中通过 yum 安装 Nginx

在容器中执行以下命令安装Nginx服务：

```
[root@ecfa046e9681]#yum install nginx -y
```

该命令使用yum包管理器安装Nginx软件包，对于所有的安装确认提示自动应答yes。

- yum：CentOS/RHEL系统中的包管理器，在容器中用于安装、更新和卸载软件包。
- install：yum的子命令，用于安装指定的软件包。
- nginx：要安装的软件包，即Nginx。
- -y：对于所有的安装确认提示自动应答yes，不需要手动确认。这个选项可以避免因为等待用户确认而导致安装过程被阻塞。

1. 安装文本编辑器vim

```
[root@ecfa046e9681]#yum install vim-enhanced -y
```

该命令的作用是使用yum包管理器安装vim-enhanced软件包，对于所有的安装确认提示自动应答yes。

- vim-enhanced：要安装的软件包，即 vim的增强版，包含更多的功能和插件。

2. 创建Nginx的静态页面

```
[root@ecfa046e9681]#mkdir /var/www/html -p
```

该命令创建一个名为/var/www/html的目录，并使用-p选项创建整个目录路径。

- mkdir：在容器中创建一个目录。
- /var/www/html：要创建的目录路径，即在根目录下创建一个名为var的目录，再在var目录下创建一个名为www的目录，最后在www目录下创建一个名为html的目录。
- -p：如果要创建的目录的上级目录不存在，则自动创建上级目录。如果不使用-p选项，则必须手动创建上级目录，否则会报错。

```
[root@ecfa046e9681]#cd /var/www/html/
```

将当前路径切换到/var/www/html/目录下。

```
[root@ecfa046e9681]#vim index.html
```

在当前目录下创建或编辑一个名为index.html的文件。

如果文件不存在，则会创建一个新文件；如果文件已经存在，则会打开源文件并允许进行编辑操作。

文件内容如下：

```
<html>
       <head>
              <title>nginx in Docker</title>
       </head>
       <body>
             <h1>hello,My Name is xianchao</h1> #首页内容
       </body>
</html>
```

其中，<h1>hello,My Name is xianchao</h1>是自己定义的Nginx首页内容。

3. 修改Nginx配置文件中的root路径

```
[root@ecfa046e9681]#vim /etc/nginx/nginx.conf
```

我们将root更改为/var/www/html/。

这里使用vim编辑器打开Nginx的配置文件/etc/nginx/nginx.conf，并把其中的root指令的值修改为/var/www/html/。将其修改为/var/www/html/后，当用户访问Nginx服务器时，Nginx将会从/var/www/html/目录下提供静态文件。

4. 启动Nginx

```
[root@ecfa046e9681]#/usr/sbin/nginx
```

该命令会启动Nginx服务器。其中，/usr/sbin/nginx表示启动Nginx服务器的命令路径，即在/usr/sbin/目录下找到名为nginx的可执行文件并执行。执行这个命令后，Nginx服务器将会在后台以守护进程的方式运行，监听指定的端口并提供静态文件服务。

5. 访问Docker中的Nginx服务，复制一个终端窗口

执行如下命令：

```
[root@xianchaomaster1 ~]# docker ps | grep nginx
ecfa046e9681   centos                                          "/bin/bash"
12 minutes ago   Up 12 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp   nginx
```

通过上述命令能查看到Docker部署的Nginx容器在物理机映射的端口是80。

接着执行以下命令，通过访问物理机IP和容器在物理机映射的端口访问容器：

```
[root@xianchaomaster1 ~]# curl http://192.168.40.180:80
```

该命令使用curl工具向指定IP地址和端口发送HTTP GET请求，并显示服务器响应的结果。

- curl：一个命令行工具，用于在终端发送HTTP请求并显示服务器响应。

- http://192.168.40.180:80：要发送请求的URL，其中包括服务器的IP地址和端口号。这里的IP地址为192.168.40.180，端口号为80。

GET请求是指HTTP中的一种请求方法，用于获取指定资源的信息。当curl工具发送HTTP GET请求时，服务器将会返回所请求资源的内容。

请求到的内容如下：

```
<html>
   <head>
        <title>nginx in Docker</title>
   </head>
   <body>
       <h1>hello,My Name is xianchao</h1>
   </body>
</html>
```

说明Docker部署成功，Docker中的Nginx容器运行正常。

当Docker容器运行在物理机上时，可以通过访问物理机的IP地址和映射的端口来访问容器中部署的应用程序。具体来说，访问流程如下：

（1）访问物理节点的IP地址和映射的端口，这些端口可以是容器暴露的端口或者是使用Docker提供的端口映射功能映射到的容器的端口。

（2）Docker守护进程收到请求后，将其路由到对应的容器，使用容器的IP地址和端口。

（3）容器收到请求后，将其转发到容器中运行的应用程序，使用应用程序的IP地址和端口。

通过这样的流程，可以方便地访问容器中部署的应用程序，实现快速开发和部署。

1.6　本章小结

本章主要从Docker零基础开始，介绍Docker容器的基本概念、特点、组成、安装和镜像加速器的配置，并对Docker的容器、镜像等命令做了详细介绍，让读者可以通过Docker运行容器，在Docker中快速部署服务。