主讲老师：Fox

有道笔记地址链接：<https://note.youdao.com/s/8eWHV1Jr>

2013年发布至今， [Docker](https://www.docker.com/) 一直广受瞩目，被认为可能会改变软件行业。

但是，许多人并不清楚 Docker 到底是什么，要解决什么问题，好处又在哪里？今天就来详细解释，帮助大家理解它，还带有简单易懂的实例，教你如何将它用于日常开发。



**1. Docker详解**

**1.1 Docker简介**

Docker是一个开源的容器化平台，可以帮助开发者将应用程序和其依赖的环境打包成一个可移植、可部署的容器。Docker的主要目标是通过容器化技术实现应用程序的快速部署、可移植性和可扩展性，从而简化应用程序的开发、测试和部署过程。

容器化是一种虚拟化技术，它通过在操作系统层面隔离应用程序和其依赖的运行环境，使得应用程序可以在一个独立的、封闭的环境中运行，而不受底层操作系统和硬件的影响。与传统的虚拟机相比，容器化具有以下优势：

轻量级: 容器与宿主机共享操作系统内核，因此容器本身非常轻量级，启动和停止速度快，资源占用少。

可移植性: 容器可以在任何支持相应容器运行时的系统上运行，无需关注底层操作系统的差异，提供了高度的可移植性。

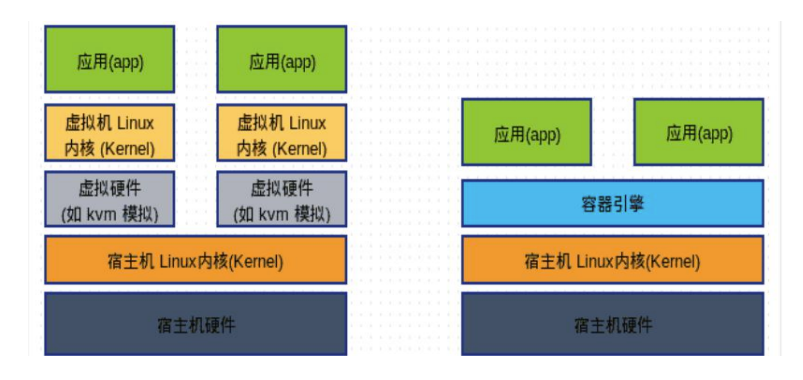
快速部署: 容器化应用程序可以通过简单的操作进行打包、分发和部署，减少了部署过程的复杂性和时间成本。

弹性扩展: 可以根据应用程序的需求快速创建、启动和停止容器实例，实现应用程序的弹性扩展和负载均衡。

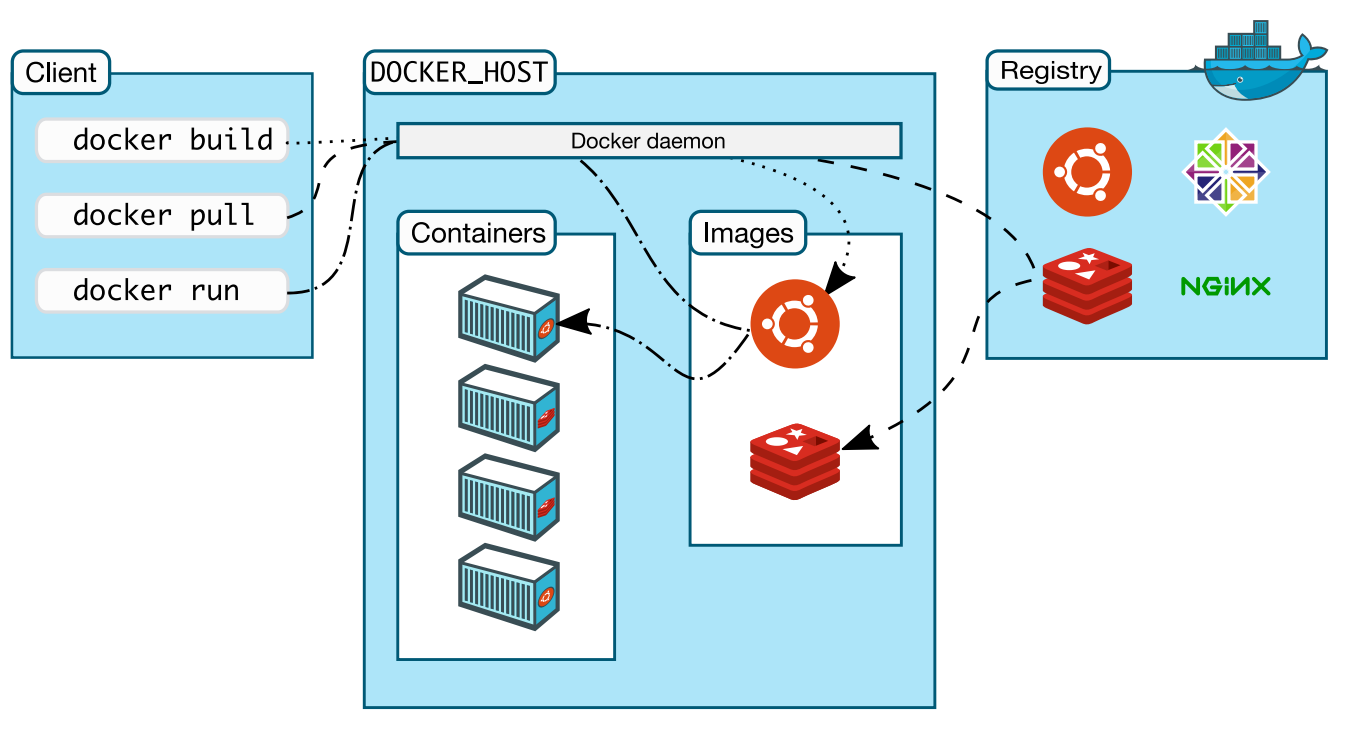
环境隔离: 每个容器都具有独立的运行环境，容器之间相互隔离，不会相互干扰，提供了更好的安全性和稳定性。

**docker和传统虚拟机区别**

虚拟机是一个主机模拟出多个主机，需要先拥有独立的系统。传统虚拟机，利用hypervisor，模拟出独立的硬件和系统，在此之上创建应用。docker 是在主机系统中建立多个应用及配套环境，把应用及配套环境独立打包成一个单位，是进程级的隔离。



**1.2 Docker架构**



**Docker daemon（ Docker守护进程）**

Docker daemon是一个运行在宿主机（ DOCKER-HOST）的后台进程。可通过 Docker客户端与之通信。

**Client（ Docker客户端）**

Docker客户端是 Docker的用户界面，它可以接受用户命令和配置标识，并与 Docker daemon通信。图中， docker build等都是 Docker的相关命令。

**Images（ Docker镜像）**

Docker镜像是一个只读模板，它包含创建 Docker容器的说明。**它和系统安装光盘有点像**，使用系统安装光盘可以安装系统，同理，使用Docker镜像可以运行 Docker镜像中的程序。

**Container（容器）**

容器是镜像的可运行实例。**镜像和容器的关系有点类似于面向对象中，类和对象的关系**。可通过 Docker API或者 CLI命令来启停、移动、删除容器。

**Registry**

Docker Registry是一个集中存储与分发镜像的服务。构建完 Docker镜像后，就可在当前宿主机上运行。但如果想要在其他机器上运行这个镜像，就需要手动复制。此时可借助 Docker Registry来避免镜像的手动复制。

一个 Docker Registry可包含多个 Docker仓库，每个仓库可包含多个镜像标签，每个标签对应一个 Docker镜像。这跟 Maven的仓库有点类似，如果把 Docker Registry比作 Maven仓库的话，那么 Docker仓库就可理解为某jar包的路径，而镜像标签则可理解为jar包的版本号。

Docker Registry可分为公有Docker Registry和私有Docker Registry。 最常⽤的Docker Registry莫过于官方的[Docker Hub](https://hub.docker.com/)， 这也是默认的Docker Registry。 Docker Hub上存放着大量优秀的镜像， 我们可使用Docker命令下载并使用。

**1.3 Docker 安装**

Docker 是一个开源的商业产品，有两个版本：社区版（Community Edition，缩写为 CE）和企业版（Enterprise Edition，缩写为 EE）。企业版包含了一些收费服务，个人开发者一般用不到。下面的介绍都针对社区版。

Docker CE 的安装请参考[官方文档](https://docs.docker.com/engine/install/centos/)，**我们这里以CentOS为例：**

1、Docker 要求 CentOS 系统的内核版本高于 3.10

通过 uname -r 命令查看你当前的内核版本

uname -r

2、使用 root 权限登录 Centos。确保 yum 包更新到最新。

yum -y update

3、卸载旧版本(如果安装过旧版本的话)

sudo yum remove -y docker\*

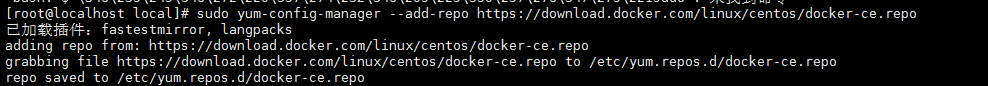
4、安装需要的软件包， yum-util 提供yum-config-manager功能，另外两个是devicemapper驱动依赖的

yum install -y yum-utils

5、设置yum源，并更新 yum 的包索引

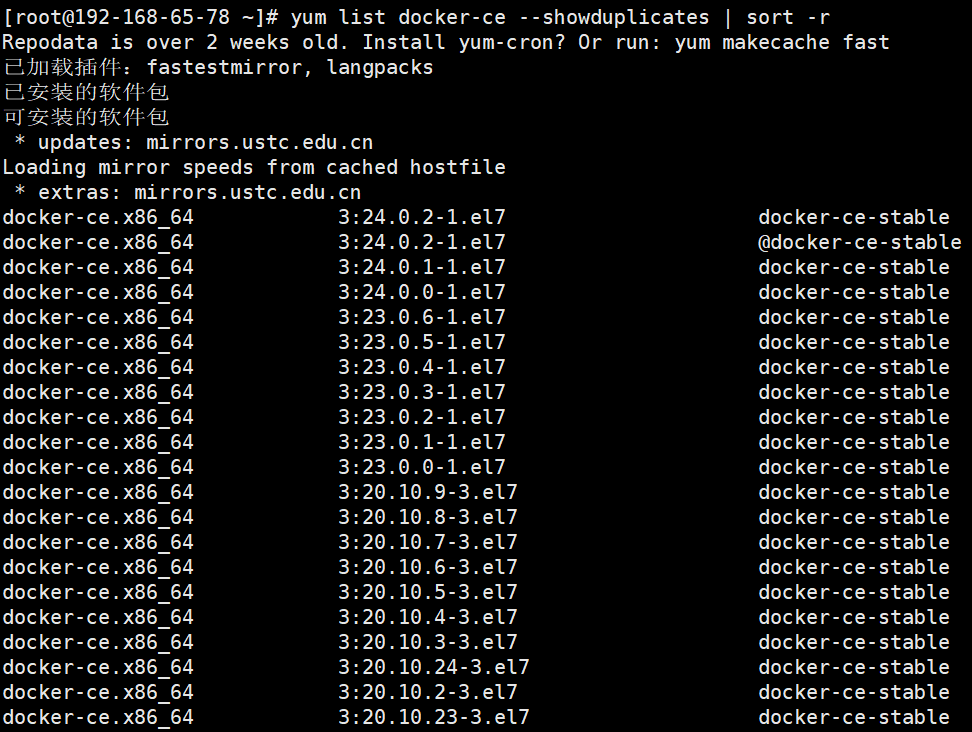
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

yum makecache fast



6、可以查看所有仓库中所有docker版本，并选择特定版本安装

yum list docker-ce --showduplicates | sort -r



7、安装docker

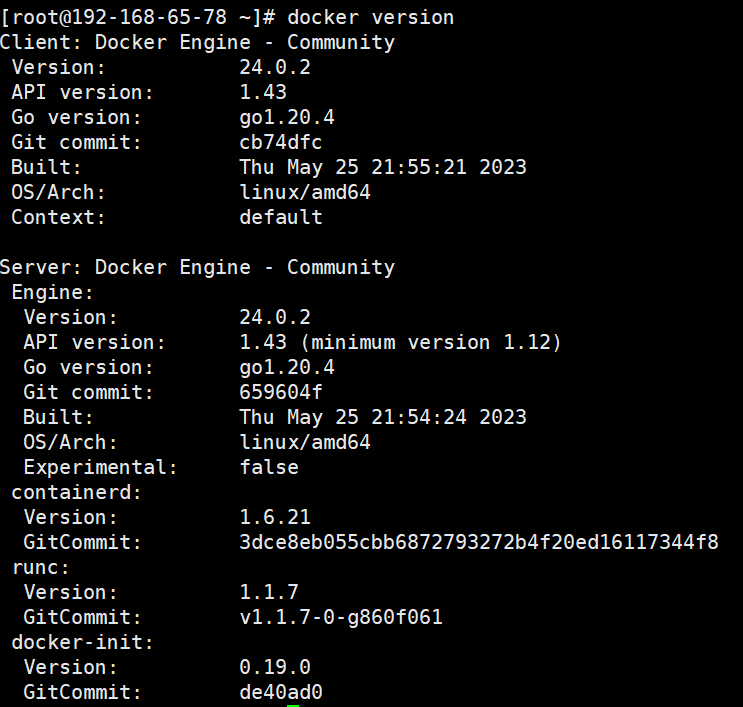
yum install -y docker-ce-3:24.0.2-1.el7.x86\_64 # 这是指定版本安装

8、启动并加入开机启动

systemctl start docker && systemctl enable docker

9、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)

docker version



**注意：一般需要配置docker镜像加速器**

我们可以借助阿里云的镜像加速器，登录阿里云(<https://cr.console.aliyun.com/#/accelerator>)

可以看到镜像加速地址如下图：



cd /etc/docker

查看有没有 daemon.json。这是docker默认的配置文件。

如果没有新建，如果有，则修改。

vim daemon.json

{

  "registry-mirrors": ["https://jbw52uwf.mirror.aliyuncs.com"]

}

保存退出。

重启docker服务

systemctl daemon-reload

systemctl restart docker

成功！

10、卸载docker

yum remove -y docker\*

rm -rf /etc/systemd/system/docker.service.d

rm -rf /var/lib/docker

rm -rf /var/run/docker

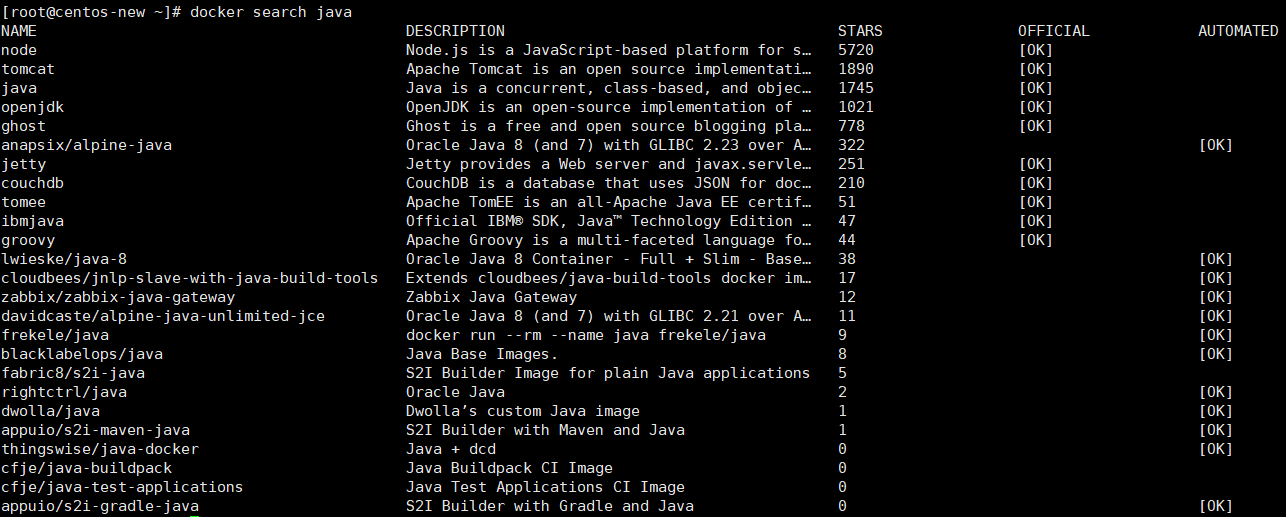
**1.4 Docker使用**

**镜像相关命令**

**1、搜索镜像**

可使用 docker search命令搜索存放在 Docker Hub中的镜像。执行该命令后， Docker就会在Docker Hub中搜索含有 java这个关键词的镜像仓库。

docker search java



以上列表包含五列，含义如下：

- NAME:镜像仓库名称。

- DESCRIPTION:镜像仓库描述。

- STARS：镜像仓库收藏数，表示该镜像仓库的受欢迎程度，类似于 GitHub的 stars0

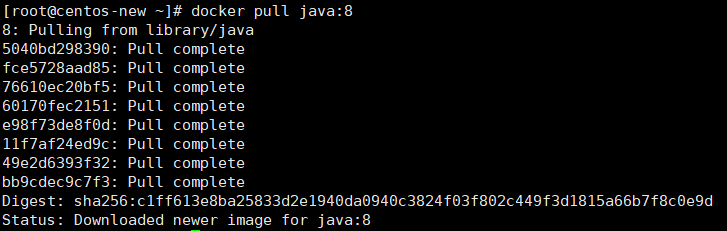
- OFFICAL:表示是否为官方仓库，该列标记为[0K]的镜像均由各软件的官方项目组创建和维护。

- AUTOMATED：表示是否是自动构建的镜像仓库。

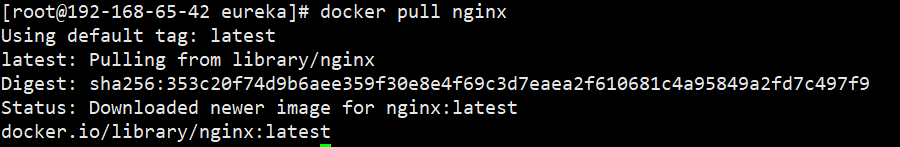
**2、下载镜像**

使用命令docker pull命令即可从 Docker Registry上下载镜像，执行该命令后，Docker会从 Docker Hub中的 java仓库下载最新版本的 Java镜像。如果要下载指定版本则在java后面加冒号指定版本，例如：docker pull java:8

docker pull java:8



docker pull nginx



**3、列出镜像**

使用 docker images命令即可列出已下载的镜像

docker images

截图.png

以上列表含义如下

- REPOSITORY：镜像所属仓库名称。

- TAG:镜像标签。默认是 latest,表示最新。

- IMAGE ID：镜像 ID，表示镜像唯一标识。

- CREATED：镜像创建时间。

- SIZE: 镜像大小。

**4、删除本地镜像**

使用 docker rmi命令即可删除指定镜像，强制删除加 -f

docker rmi java

删除所有镜像

docker rmi $(docker images -q)

**容器相关命令**

**1、新建并启动容器**

使用以下docker run命令即可新建并启动一个容器，该命令是最常用的命令，它有很多选项，下面将列举一些常用的选项。

-d选项：表示后台运行

-P选项：随机端口映射

-p选项：指定端口映射，有以下四种格式。

-- ip:hostPort:containerPort

-- ip::containerPort

-- hostPort:containerPort

-- containerPort

--net选项：指定网络模式，该选项有以下可选参数：

--net=bridge:**默认选项**，表示连接到默认的网桥。

--net=host:容器使用宿主机的网络。

--net=container:NAME-or-ID：告诉 Docker让新建的容器使用已有容器的网络配置。

--net=none：不配置该容器的网络，用户可自定义网络配置。

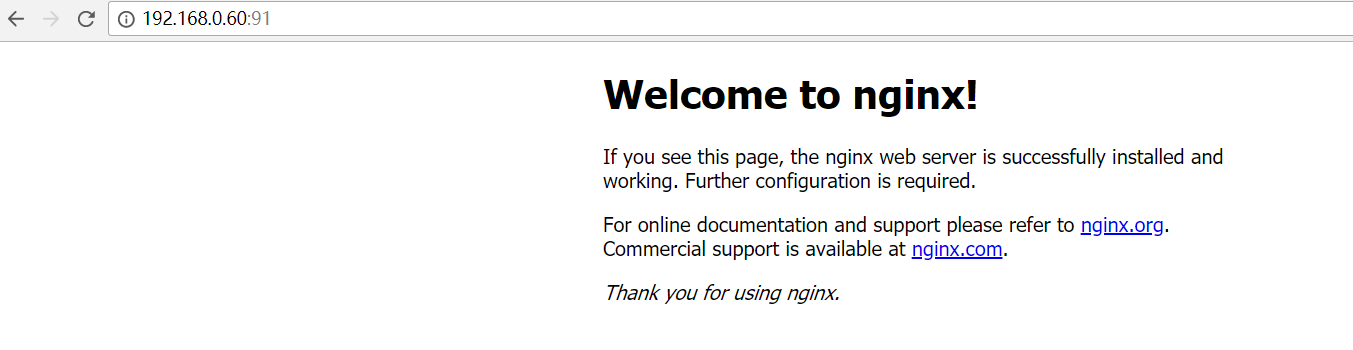
docker run -d -p 91:80 nginx

这样就能启动一个 Nginx容器。在本例中，为 docker run添加了两个参数，含义如下：

-d 后台运行

-p 宿主机端口:容器端口 #开放容器端口到宿主机端口

访问 http://Docker宿主机 IP:91/，将会看到nginx的主界面如下：



需要注意的是，使用 docker run命令创建容器时，会先检查本地是否存在指定镜像。如果本地不存在该名称的镜像， Docker就会自动从 Docker Hub下载镜像并启动一个 Docker容器。

**2、列出容器**

用 docker ps命令即可列出**运行中**的容器

docker ps

截图.png

如需列出所有容器（包括已停止的容器），可使用-a参数。该列表包含了7列，含义如下

- CONTAINER\_ID：表示容器 ID。

- IMAGE:表示镜像名称。

- COMMAND：表示启动容器时运行的命令。

- CREATED：表示容器的创建时间。

- STATUS：表示容器运行的状态。UP表示运行中， Exited表示已停止。

- PORTS:表示容器对外的端口号。

- NAMES:表示容器名称。该名称默认由 Docker自动生成，也可使用 docker run命令的--name选项自行指定。

**3、停止容器**

使用 docker stop命令，即可停止容器

docker stop f0b1c8ab3633

其中f0b1c8ab3633是容器 ID,当然也可使用 docker stop容器名称来停止指定容器

**4、强制停止容器**

可使用 docker kill命令发送 SIGKILL信号来强制停止容器

docker kill f0b1c8ab3633

**5、启动已停止的容器**

使用docker run命令，即可**新建**并启动一个容器。对于已停止的容器，可使用 docker start命令来**启动**

docker start f0b1c8ab3633

**6、查看容器所有信息**

docker inspect f0b1c8ab3633

**7、查看容器日志**

docker container logs f0b1c8ab3633

**8、查看容器里的进程**

docker top f0b1c8ab3633

**9、容器与宿主机相互复制文件**

从容器里面拷文件到宿主机：

docker cp 容器id:要拷贝的文件在容器里面的路径 宿主机的相应路径

如：docker cp 7aa5dc458f9d:/etc/nginx/nginx.conf /mydata/nginx

从宿主机拷文件到容器里面：

docker cp 要拷贝的宿主机文件路径 容器id:要拷贝到容器里面对应的路径

**10、进入容器**

使用docker exec命令用于进入一个正在运行的docker容器。如果docker run命令运行容器的时候，没有使用-it参数，就要用这个命令进入容器。一旦进入了容器，就可以在容器的 Shell 执行命令了

docker exec -it f0b1c8ab3633 /bin/bash (有的容器需要把 /bin/bash 换成 sh)

**11、容器内安装vim、ping、ifconfig等指令**

apt-get update

apt-get install vim #安装vim

apt-get install iputils-ping #安装ping

apt-get install net-tools #安装ifconfig

**12、删除容器**

使用 docker rm命令即可删除指定容器

docker rm f0b1c8ab3633

该命令只能删除**已停止**的容器，如需删除正在运行的容器，可使用-f参数

强制删除所有容器

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

**2. 使用Dockerfile构建Docker镜像**

Dockerfile是一个文本文件，其中包含了若干条指令，指令描述了构建镜像的细节

先来编写一个最简单的Dockerfile，以前文下载的Nginx镜像为例，来编写一个Dockerfile修改该Nginx镜像的首页

1、新建一个空文件夹docker-demo，在里面再新建文件夹app，在app目录下新建一个名为Dockerfile的文件，在里面增加如下内容：

FROM nginx

RUN echo '<h1>This is Tuling Nginx!!!</h1>' > /usr/share/nginx/html/index.html

该Dockerfile非常简单，其中的 FROM、 RUN都是 Dockerfile的指令。 FROM指令用于指定基础镜像， RUN指令用于执行命令。

2、在Dockerfile所在路径执行以下命令构建镜像：

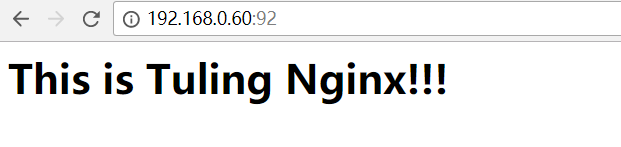
docker build -t nginx:tuling .

其中，-t指定镜像名字，命令最后的点（.）表示Dockerfile文件所在路径

3、执行以下命令，即可使用该镜像启动一个 Docker容器

docker run -d -p 92:80 nginx:tuling

4、访问 http://Docker宿主机IP:92/，可看到下图所示界面



**2.1 Dockerfile常用指令**

|  |  |
| --- | --- |
| 命令 | 用途 |
| FROM | 基础镜像文件 |
| RUN | 构建镜像阶段执行命令 |
| ADD <src> <dest> | 添加文件，从src目录复制文件到容器的dest，其中 src可以是 Dockerfile所在目录的相对路径，也可以是一个 URL,还可以是一个压缩包 |
| COPY | 拷贝文件，和ADD命令类似，但不支持URL和压缩包 |
| CMD | 容器启动后执行命令 |
| EXPOSE | 声明容器在运行时对外提供的服务端口 |
| WORKDIR | 指定容器工作路径 |
| ENV | 指定环境变量 |
| ENTRYPINT | 容器入口， ENTRYPOINT和 CMD指令的目的一样，都是指定 Docker容器启动时执行的命令，可多次设置，但只有最后一个有效。 |
| USER | 该指令用于设置启动镜像时的用户或者 UID,写在该指令后的 RUN、 CMD以及 ENTRYPOINT指令都将使用该用户执行命令。 |
| VOLUME | 指定挂载点，该指令使容器中的一个目录具有持久化存储的功能，该目录可被容器本身使用，也可共享给其他容器。当容器中的应用有持久化数据的需求时可以在 Dockerfile中使用该指令。格式为： VOLUME["/data"]。 |

注意：RUN命令在 image 文件的构建阶段执行，执行结果都会打包进入 image 文件；CMD命令则是在容器启动后执行。另外，一个 Dockerfile 可以包含多个RUN命令，但是只能有一个CMD命令。

注意，指定了CMD命令以后，docker container run命令就不能附加命令了（比如前面的/bin/bash），否则它会覆盖CMD命令。

**2.2 使用Dockerfile构建微服务镜像**

以项目tulingmall-member为例，将该微服务的可运行jar包构建成docker镜像

1、将jar包上传linux服务器/root/tulingmall/tulingmall-member目录，在jar包所在目录创建名为Dockerfile的文件

2、在Dockerfile中添加以下内容

# 基于哪个镜像

From java:8

# 复制文件到容器

ADD tulingmall-member-0.0.5.jar /tulingmall-member-0.0.5.jar

# 声明需要暴露的端口

EXPOSE 8877

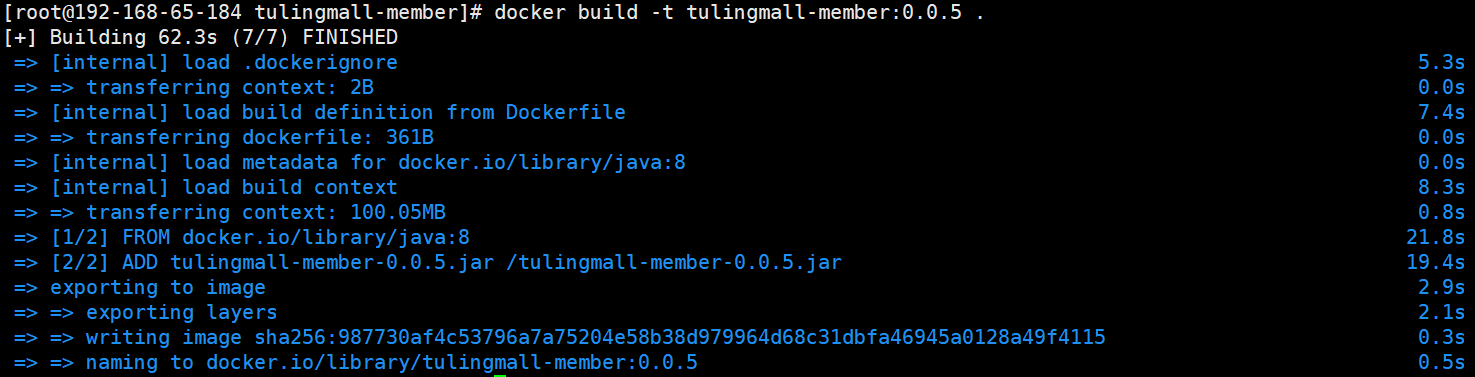
# 配置容器启动后执行的命令

ENTRYPOINT java ${JAVA\_OPTS} -jar /tulingmall-member-0.0.5.jar

3、使用docker build命令构建镜像

docker build -t tulingmall-member:0.0.5 .

# 格式： docker build -t 镜像名称:标签 Dockerfile的相对位置



4、启动镜像，加-d可在后台启动

docker run -d -p 8877:8877 tulingmall-member:0.0.5

加上JVM参数：

# --cap-add=SYS\_PTRACE 这个参数是让docker能支持在容器里能执行jdk自带类似jinfo，jmap这些命令，如果不需要在容器里执行这些命令可以不加

docker run -d -p 8877:8877 \

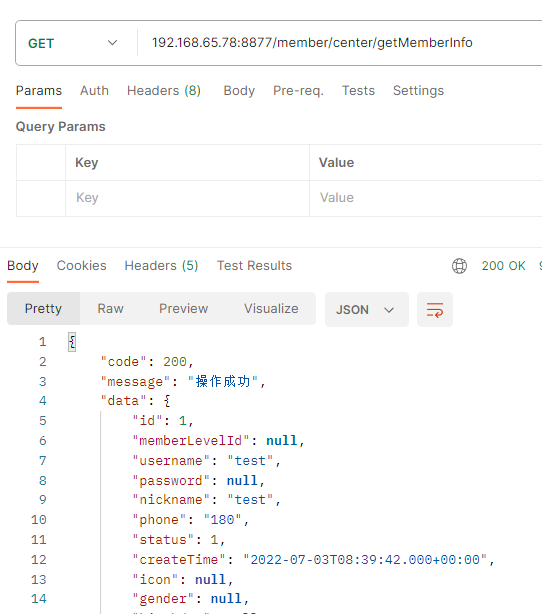
-e SPRING\_CLOUD\_NACOS\_CONFIG\_SERVER\_ADDR=192.168.65.174:8848 \

-e JAVA\_OPTS='-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m' \

--cap-add=SYS\_PTRACE \

tulingmall-member:0.0.5

5、访问会员服务接口



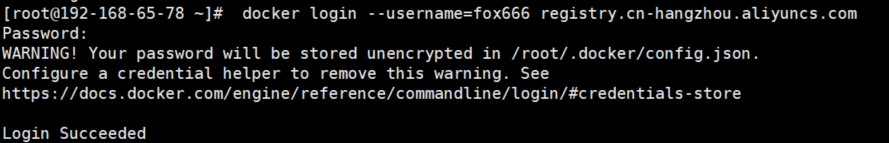
**3. 将微服务镜像发布到阿里云远程镜像仓库**

我们制作好了微服务镜像，一般需要发布到镜像仓库供别人使用，我们可以选择自建镜像仓库，也可以直接使用官方镜像仓库，这里我们选择

阿里云docker镜像仓库：<https://cr.console.aliyun.com/cn-hangzhou/instance/repositories>

首先，我们需要注册一个阿里云账号，创建容器镜像服务

然后，在linux服务器上用docker login命令登录镜像仓库

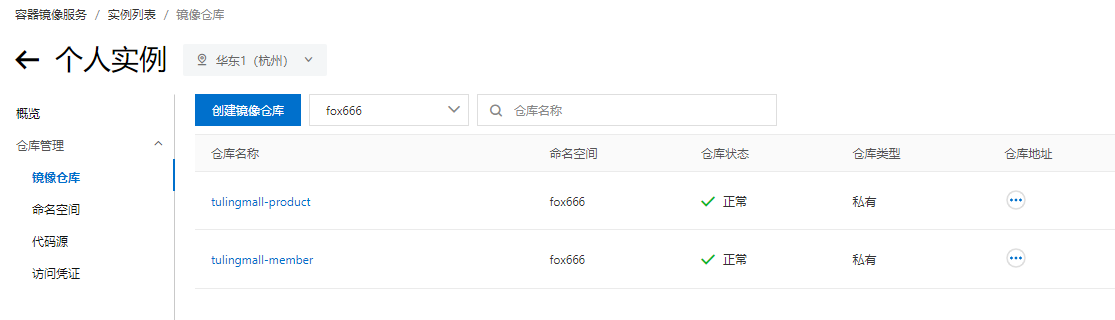


要把镜像推送到镜像仓库

docker tag tulingmall-member:0.0.5 registry.cn-hangzhou.aliyuncs.com/fox666/tulingmall-member:0.0.5

最后将镜像推送到远程仓库

docker push registry.cn-hangzhou.aliyuncs.com/fox666/tulingmall-member:0.0.5



**4. 将微服务镜像发布到私有镜像仓库**

**4.1 搭建私有docker镜像仓库**

配置 Docker 私有仓库：

创建一个用于存储仓库数据的目录，例如 /data/docker-registry。

创建一个名为

docker-compose.yml 的文件，并在其中定义 Docker 私有仓库的配置。示例配置如下：

version: '3'

services:

registry:

container\_name: docker-registry

image: registry:2

ports:

- 5000:5000

volumes:

- /data/docker-registry:/var/lib/registry

这将创建一个名为 docker-registry 的容器，并将其映射到主机的 5000 端口。仓库数据将存储在主机上的 /data/docker-registry 目录中。

启动私有仓库：

在包含 docker-compose.yml 文件的目录中，运行以下命令启动私有仓库容器：

docker compose up -d

私有仓库将在后台运行，并监听主机的 5000 端口。

设置私有仓库的用户名和密码

在 CentOS 7.9 中，可以使用 httpd-tools 软件包中的 htpasswd 工具来生成加密密码。

yum install httpd-tools

# 生成密码文件

htpasswd -Bc auth.htpasswd <用户名>

配置 Docker Daemon：

vim /etc/docker/daemon.json

# 将 <私有仓库地址> 替换为实际的私有仓库地址

{

"registry-mirrors": ["https://jbw52uwf.mirror.aliyuncs.com"],"insecure-registries": ["192.168.65.78:5000"]

}

重启 Docker Daemon:

systemctl daemon-reload && systemctl restart docker

**4.2 上传镜像到私有仓库**

要上传镜像到私有仓库，您可以按照以下步骤进行操作：

构建您的镜像：

在本地开发环境中使用 Dockerfile 构建您的镜像。确保您的镜像正确地命名为私有仓库的地址，例如 192.168.65.78:5000/tulingmall-product:latest。

运行以下命令来构建并标记您的镜像：

docker build -t 192.168.65.78:5000/tulingmall-member:latest .

登录到私有仓库：

在上传镜像之前，您需要登录到私有仓库以进行身份验证。

运行以下命令来登录到私有仓库：

docker login 192.168.65.78:5000

输入您的用户名和密码，以登录到私有仓库。

推送镜像到私有仓库：

完成登录后，您可以使用以下命令将镜像推送到私有仓库：

docker push 192.168.65.78:5000/tulingmall-member:latest

Docker 将会上传您的镜像到私有仓库中。

可以通过以下命令来验证镜像是否已经成功推送到私有仓库：

curl -X GET http://192.168.65.78:5000/v2/\_catalog