

2013年发布至今，[Docker](#) 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。



## Docker简介

Docker是一个开源的**容器引擎**，它有助于更快地交付应用。Docker可将应用程序和基础设施层隔离，并且能将基础设施当作程序一样进行管理。使用 **Docker可更快地打包、测试以及部署应用程序，并可以缩短从编写到部署运行代码的周期。**

Docker的优点如下：

### 1、简化程序

Docker 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 机器上，便可以实现虚拟化。Docker改变了虚拟化的方式，使开发者可以直接将自己的成果放入Docker中进行管理。方便快捷已经是 Docker的最大优势，过去需要用数天乃至数周的任务，在Docker容器的处理下，只需要数秒就能完成。

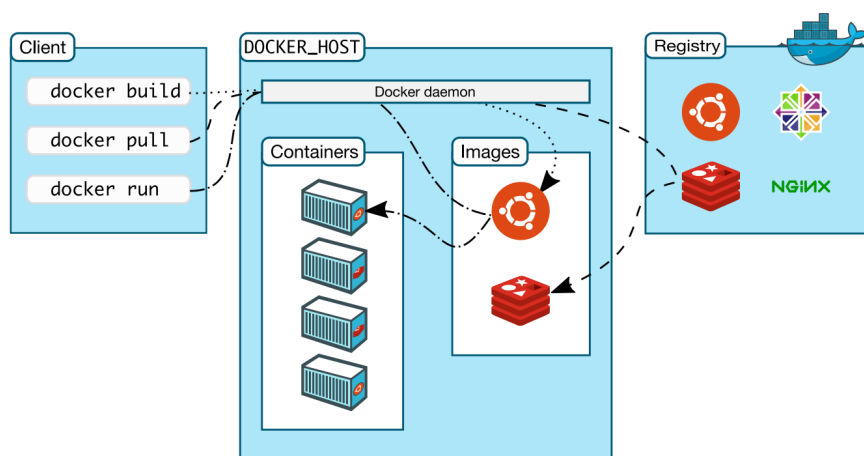
### 2、避免选择恐惧症

如果你有选择恐惧症，还是资深患者。Docker 帮你 打包你的纠结！比如 Docker 镜像；Docker 镜像中包含了运行环境和配置，所以 Docker 可以简化部署多种应用实例工作。比如 Web 应用、后台应用、数据库应用、大数据应用比如 Hadoop 集群、消息队列等等都可以打包成一个镜像部署。

### 3、节省开支

一方面，云计算时代到来，使开发者不必为了追求效果而配置高额的硬件，Docker 改变了高性能必然高价格的思维定势。Docker 与云的结合，让云空间得到更充分的利用。不仅解决了硬件管理的问题，也改变了虚拟化的方式。

## Docker的架构



- **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，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。

## Docker 的安装

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，**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

```
1  uname -r
```

2、使用 root 权限登录 Centos。确保 yum 包更新到最新。

```
1  yum -y update
```

3、卸载旧版本(如果安装过旧版本的话)

```
1  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

```
1  yum install -y yum-utils
```

5、设置yum源，并更新 yum 的包索引

```
1  yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
2  yum makecache fast
```

```
[root@localhost local]# sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
已加载插件: fastestmirror, langpacks
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
```

6、可以查看所有仓库中所有docker版本，并选择特定版本安装

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

```
[root@192-168-65-160 ~]# docker -version^C
[root@192-168-65-160 ~]# yum list docker-ce --showduplicates | sort -r
已加载插件: fastestmirror, langpacks
可安装的软件包
* updates: mirrors.163.com
Loading mirror speeds from cached hostfile
* extras: mirrors.163.com
docker-ce.x86_64      3:20.10.6-3.el7      docker-ce-stable
docker-ce.x86_64      3:20.10.5-3.el7      docker-ce-stable
docker-ce.x86_64      3:20.10.4-3.el7      docker-ce-stable
docker-ce.x86_64      3:20.10.3-3.el7      docker-ce-stable
docker-ce.x86_64      3:20.10.2-3.el7      docker-ce-stable
docker-ce.x86_64      3:20.10.1-3.el7      docker-ce-stable
docker-ce.x86_64      3:20.10.0-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.9-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.8-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.7-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.6-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.5-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.4-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.3-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.2-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.15-3.el7     docker-ce-stable
docker-ce.x86_64      3:19.03.14-3.el7     docker-ce-stable
docker-ce.x86_64      3:19.03.1-3.el7      docker-ce-stable
docker-ce.x86_64      3:19.03.13-3.el7     docker-ce-stable
docker-ce.x86_64      3:19.03.12-3.el7     docker-ce-stable
docker-ce.x86_64      3:19.03.11-3.el7     docker-ce-stable
docker-ce.x86_64      3:19.03.10-3.el7     docker-ce-stable
```

## 7、安装docker

```
1 yum install -y docker-ce-3:19.03.9-3.el7.x86_64 # 这是指定版本安装
```

## 8、启动并加入开机启动

```
1 systemctl start docker && systemctl enable docker
```

## 9、验证安装是否成功(有client和服务两部分表示docker安装启动都成功了)

```
1 docker version
```

```
[root@192-168-65-160 docker]# docker version
Client: Docker Engine - Community
 Version:      19.03.9
 API version:   1.40
 Go version:    go1.13.10
 Git commit:    9d988398e7
 Built:         Fri May 15 00:25:27 2020
 OS/Arch:       linux/amd64
 Experimental:   false

Server: Docker Engine - Community
 Engine:
  Version:      19.03.9
  API version:   1.40 (minimum version 1.12)
  Go version:    go1.13.10
  Git commit:    9d988398e7
  Built:         Fri May 15 00:24:05 2020
  OS/Arch:       linux/amd64
  Experimental:   false
containerd:
 Version:      1.4.4
 GitCommit:    05f951a3781f4f2c1911b05e61c160e9c30eaa8e
runc:
 Version:      1.0.0-rc93
 GitCommit:    12644e614e25b05da6fd08a38ffa0cfe1903fdec
docker-init:
 Version:      0.18.0
 GitCommit:    fec3683
```

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：

阿里云

容器镜像服务

实例列表

镜像中心

镜像搜索

我的收藏

镜像工具

镜像加速器

使用加速器可以提升获取Docker官方镜像的速度

加速器

加速器地址

<https://m9r2r2uj.mirror.aliyuncs.com> 复制

操作文档

Ubuntu CentOS Mac Windows

1. 安装 / 升级Docker客户端

推荐安装 1.10.0 以上版本的Docker客户端，参考文档[docker-ce](#)

2. 配置镜像加速器

针对Docker客户端版本大于 1.10.0 的用户

您可以通过修改daemon配置文件 `/etc/docker/daemon.json` 来使用加速器

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<- 'EOF'
{
  "registry-mirrors": ["https://m9r2r2uj.mirror.aliyuncs.com"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

```
1 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。

```
1 vim daemon.json
2 {
3   "registry-mirrors": ["https://m9r2r2uj.mirror.aliyuncs.com"]
4 }
```

保存退出。

重启docker服务

```
1 systemctl daemon-reload
2 systemctl restart docker
```

成功！

## 10、卸载docker

```
1 yum remove -y docker*
2 rm -rf /etc/systemd/system/docker.service.d
3 rm -rf /var/lib/docker
4 rm -rf /var/run/docker
```

# Docker常用命令

## 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。

```
1 docker search java
```

```
[root@centos-new ~]# docker search java
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
node	Node.js is a JavaScript-based platform for s...	5720	[OK]	
tomcat	Apache Tomcat is an open source implementati...	1890	[OK]	
java	Java is a concurrent, class-based, and objec...	1745	[OK]	
openjdk	OpenJDK is an open-source implementation of ...	1021	[OK]	
ghost	Ghost is a free and open source blogging pla...	778	[OK]	
anapsix/alpine-java	Oracle Java 8 (and 7) with GLIBC 2.23 over A...	322		[OK]
jetty	Jetty provides a Web server and javax.servle...	251	[OK]	
couchdb	CouchDB is a database that uses JSON for doc...	210	[OK]	
tomee	Apache TomEE is an all-Apache Java EE certif...	51	[OK]	
ibmjava	Official IBM® SDK, Java™ Technology Edition ...	47	[OK]	
groovy	Apache Groovy is a multi-faceted language fo...	44	[OK]	
lwieske/java-8	Oracle Java 8 Container - Full + Slim - Base...	38		[OK]
cloudbees/jnlp-slave-with-java-build-tools	Extends cloudbees/java-build-tools docker im...	17		[OK]
zabbix/zabbix-java-gateway	Zabbix Java Gateway	12		[OK]
dauidcaste/alpine-java-unlimited-jce	Oracle Java 8 (and 7) with GLIBC 2.21 over A...	11		[OK]
frekele/java	docker run --rm --name java frekele/java	9		[OK]
blacklabelops/java	Java Base Images.	8		[OK]
fabric8/s2i-java	S2I Builder Image for plain Java applications	5		
rightctrl/java	Oracle Java	2		[OK]
dwolla/java	Dwolla's custom Java image	1		[OK]
appuio/s2i-maven-java	S2I Builder with Maven and Java	1		[OK]
thingswise/java-docker	Java + dcd	0		[OK]
cfje/java-buildpack	Java Buildpack CI Image	0		
cfje/java-test-applications	Java Test Applications CI Image	0		
appuio/s2i-gradle-java	S2I Builder with Gradle and Java	0		[OK]

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[OK]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

```
1 docker pull java:8
```

```
[root@centos-new ~]# docker pull java:8
8: Pulling from library/java
5040bd298390: Pull complete
fce5728aad85: Pull complete
76610ec20bf5: Pull complete
60170fec2151: Pull complete
e98f73de8f0d: Pull complete
11f7af24ed9c: Pull complete
49e2d6393f32: Pull complete
bb9cdec9c7f3: Pull complete
Digest: sha256:c1ff613e8ba25833d2e1940da0940c3824f03f802c449f3d1815a66b7f8c0e9d
Status: Downloaded newer image for java:8
```

```
1 docker pull nginx
```

```
[root@192-168-65-42 eureka]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:353c20f74d9b6aee359f30e8e4f69c3d7eaea2f610681c4a95849a2fd7c497f9
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

## 3、列出镜像

使用 docker images命令即可列出已下载的镜像

```
1 docker images
```

```
[root@centos-new ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
java	8	d23bdf5b1b1b	17 months ago	643MB

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

```
1 docker rmi java
```

删除所有镜像

```
1 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络。

--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。

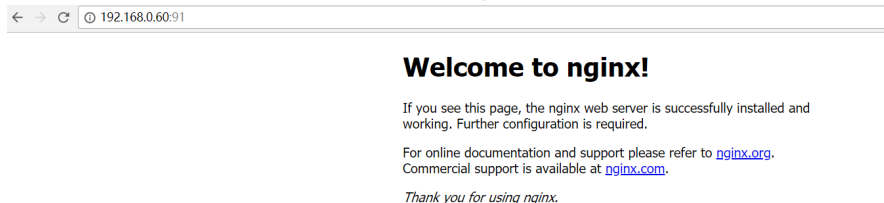
```
1 docker run -d -p 91:80 nginx
```

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

```
1 docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f0b1c8ab3633	nginx	"nginx -g 'daemon of..."	About a minute ago	Up About a minute	0.0.0.0:91->80/tcp	xenodochial_neumann

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

```
1 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

```
1 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**

```
1 docker start f0b1c8ab3633
```

### 6、查看容器所有信息

```
1 docker inspect f0b1c8ab3633
```

### 7、查看容器日志

```
1 docker container logs f0b1c8ab3633
```

### 8、查看容器里的进程

```
1 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：

```
1 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
2 如：docker cp 7aa5dc458f9d:/etc/nginx/nginx.conf /mydata/nginx
```

- 从宿主机拷文件到容器里面：

```
1 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

```
1 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

```
1 apt-get update
2 apt-get install vim #安装vim
3 apt-get install iputils-ping #安装ping
4 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

```
1 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

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

## 将微服务运行在docker上



## 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：

```
1 FROM nginx
2 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：

```
1 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

```
1 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

← → ↻ 192.168.0.60:92

# This is Tuling Nginx!!!

## 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
ENTRYPOINT	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。

## 使用Dockerfile构建微服务镜像

以项目05-ms-eureka-server为例，将该微服务的可运行jar包构建成为docker镜像



1、将jar包上传linux服务器/usr/local/docker-app/docker-demo/app/eureka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

```
1 # 基于哪个镜像
2 From java:8
3 # 复制文件到容器
4 ADD microservice-eureka-server-0.0.1-SNAPSHOT.jar /app.jar
5 # 声明需要暴露的端口
6 EXPOSE 8761
7 # 配置容器启动后执行的命令
8 ENTRYPOINT java ${JAVA_OPTS} -jar /app.jar
```

3、使用docker build命令构建镜像

```
1 docker build -t microservice-eureka-server:0.0.1 .
```

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

在这里，使用-t选项指定了镜像的标签。执行该命令后，终端将会输出如下的内容

```
[root@centos-new soft]# docker build -t microservice-eureka-server:0.0.1 .
Sending build context to Docker daemon 39.93MB
Step 1/5 : From java:8
----> d23bdf5b1b1b
Step 2/5 : VOLUME /tmp
----> Running in 47df11adc0e5
Removing intermediate container 47df11adc0e5
----> 0122276d643d
Step 3/5 : ADD microservice-eureka-server-0.0.1-SNAPSHOT.jar /app.jar
----> 14d35bbae674
Step 4/5 : EXPOSE 8761
----> Running in 3adb5133d715
Removing intermediate container 3adb5133d715
----> 0fda876dda27
Step 5/5 : ENTRYPOINT ["java","-jar","/app.jar"]
----> Running in 3d3ad98c0e2d
Removing intermediate container 3d3ad98c0e2d
----> 5ab4d23c9b69
Successfully built 5ab4d23c9b69
Successfully tagged microservice-eureka-server:0.0.1
```

4、启动镜像，加-d可在后台启动

```
1 docker run -d -p 8761:8761 microservice-eureka-server:0.0.1
```

使用 -v 可以挂载一个主机上的目录到容器的目录


```
1 docker run -d -p 8761:8761 -v /log:/container-log microservice-eureka-server:0.0.1
```

加上JVM参数:

```
1 # --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
2 docker run -e JAVA_OPTS='-Xms1028M -Xmx1028M -Xmn512M -Xss512K -XX:MetaspaceSize=256M -XX:MaxMetaspaceSize=256M' --cap-add=SYS_PTRACE -d -p 8761:8761 microservice-eureka-server:0.0.1
```

5、访问http://Docker宿主机IP:8761/, 可正常显示Eureka Server首页

← → 192.168.0.60:8761

HOME LAST 1000 SINCE STARTUP

### System Status

Environment	test	Current time	2018-07-01T05:46:09 +0000
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

### DS Replicas

localhost

### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

### General Info

Name	Value
total-avail-memory	63mb
environment	test

## 将微服务镜像发布到远程镜像仓库

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docker官方镜像仓库，这里我们选择docker官方镜像仓库：<https://hub.docker.com/>

首先，我们需要在docke官方镜像仓库里注册一个账号

然后，在linux服务器上用docker login命令登录镜像仓库

```
[root@k8s-master ~]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one
Username: zhuge666
Password:
Login Succeeded
```

要把镜像推送到镜像仓库，需要将镜像前面加个分组名(一般就是docker hub的账户名)，执行如下命令修改镜像名字

```
1 docker tag microservice-eureka-server:0.0.1 zhuge666/microservice-eureka-server:0.0.1
```

最后将镜像推送到远程仓库


```
1 docker push zhuge666/microservice-eureka-server:0.0.1
```

```
[root@k8s-master ~]# docker push zhuge666/microservice-eureka-server:0.0.1
The push refers to repository [docker.io/zhuge666/microservice-eureka-server]
399d113acc87: Pushed
35c20f26d188: Pushed
c3fe59dd9556: Pushed
6ed1a81ba5b6: Mounted from pinpointdocker/pinpoint-hbase
a3483ce177ce: Mounted from pinpointdocker/pinpoint-hbase
ce6c8756685b: Pushed
30339f20ced0: Pushed
0eb22bf707d: Pushed
a2ae92ffcd29: Pushed
0.0.1: digest: sha256:091f8b6a231596234d784cdba7ce6f78eebf5ced9c44e2aba6dcfd5b111532bc size: 2212
```

我们登录到docker镜像查看下刚刚推送的镜像，这样镜像就能给别人用了

← → X registry.hub.docker.com/repositories

New customers SAVE 20% off Docker Subscriptions with code: DOCKERCON21 See terms

Search for great content (e.g., mysql)ExploreRepositoriesOrganizationsGet Helpzhuge666

zhuge666Search by repository nameCreate Repository

zhuge666 / microservice-eureka-server  
Updated 16 minutes ago

Not Scanned0 stars4 downloadsPublic

Tip: Not finding your repository? Try switching namespace via the top left dropdown.