小沐标之 docker

作者：zjjaxx
原文链接：https://ld246.com/article/1604042250912
来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile.com
file/2020/10/u1589818293291301184 2fm26gp0-35c7c0cd.jpg?imageView2/2/interlace/1/for
at/jpg">
<h2 id="toc_h2_0">docker</h2>
<h2 id="toc_h2_1">docker 简介</h2>
<h3 id="toc_h3_2">docker由来</h3>
「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
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，也可以实现虚拟化。容器
完全使用沙箱机制，相互之间不会有任何接口。」
docker 解决了不同环境下配置不同的问题，docker可以把运行环境(系统、代码、配置、数据、
赖等)都打包到一个沙盒中(容器)，部署到任何流行的Linux机器上
<h3 id="toc_h3_3">docker vs 虚拟机</h3>
上图：
<img src="https://ld246.com/images/img-loading.svg" width="300" height="280" data-s
c="https://b3logfile.com/file/2019/12/Container2x-c6633470.png?imageView2/2/interlace/1/
ormat/jpg">
<img src="https://ld246.com/images/img-loading.svg" width="300" height="280" data-src=
https://b3logfile.com/file/2019/12/VM2x-97403df8.png?imageView2/2/interlace/1/format/jp
">
虚拟机属于虚拟化技术。而Docker这样的容器技术，也是虚拟化技术，属于轻量级的虚拟化。</
>
区别：虚拟机虽然可以隔离出很多“子电脑”，但占用空间更大，启动更慢，虚拟机软件可能还
花钱，但是docker不需要虚拟出整个操作系统，容器内的应用进程直接运行于宿主的内核。容器没有
己的内核，仅包含运行时的所需的runtime环境，所以占用资源少，启动快
<h2 id="toc_h2_4">docker 核心概念</h2>
<h3 id="toc_h3_5">上图</h3>
<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile.com
file/2020/10/u7933143034800816 97fm26gp0-150f58e3.jpg?imageView2/2/interlace/1/format
jpg">
<h3 id="toc_h3_6">仓库（Repository）</h
>
存放镜像地址的远程仓库（github类似）
<h3 id="toc_h3_7">镜像（Image）</h3>
镜像轻量级可执行的软件包，相当于是一个模板，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
据，其内容在构建之后也不会被改变。
<h3 id="toc_h3_8">容器（Container）</h3

按镜像生成的实例,每个容器相互隔离，可以把容器看成一个精简版的linux环境
<h2 id="toc_h2_9">docker 安装</h2>
首先换源 (yum或者apt)
根据官网安装，略~
<h3 id="toc_h3_10">阿里云上下载docker安装环境命令<span
</h3>
<pre class="custom"><code class="hljs">sudo yum-config-manager
nbsp;\
    --add-repo \
https://download.docker.com/linux/centos/docker-ce.repo
</code></pre>
替换成
<pre class="custom"><code class="hljs">sudo yum-config-manager
nbsp;\
    --add-repo \
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
</code></pre>
<h3 id="toc_h3_11">Docker安装时出现requires containerd.io >= 1
2.2-3错误</h3>
<pre class="custom"><code class="hljs">wget http://mirrors.aliyun.c

m/docker-ce/linux/centos/7/x86_64/stable/Packages/containerd.io-1.2.13-3.1.el7.x86_64.rpm
br>
yum -y install ./containerd.io-1.2.13-3.1.el7.x86_64.rpm
</c
de></pre>
<h2 id="toc_h2_12">docker 常用命令</h2>
<h3 id="toc_h3_13">docker 帮助命令</h3>

docker version
docker info
docker --help
<h3 id="toc_h3_14">docker 镜像命令</h3>
<h4 id="toc_h4_15">docker images</h4>
列出所有本地镜像模板
<h4 id="toc_h4_16">docker search</h4>
查找镜像
<pre class="custom"><code class="hljs">docker search nginx
nbsp;--filter=stars=800
</code></pre>
找出星星数大于800的镜像
<h4 id="toc_h4_17">docker pull</h4>
下载镜像
<pre class="custom"><code class="hljs">docker pull mysql:5.

</code></pre>
下载指定标签的镜像 默认:latest
<h4 id="toc_h4_18">docker rmi</h4>
删除镜像
<pre class="custom"><code class="hljs">docker rmi $(docker
nbsp;images -qa)
</code></pre>
删除所有镜像
<h3 id="toc_h3_19">容器命令</h3>
<h4 id="toc_h4_20">docker run</h4>
启动容器，如果没有该镜像，会从远程仓库中拉取镜像，新建容器并启动
常用的options说明；

--name 容器名
-d 后台运行容器，即启动守护进程 全拼--detach
-i 已交互
式运行容器，通常与-t一起使用
-t 为容器重新分配一个伪输入终端
-p 指定端口映

ip:hostPort:containerPort
ip::containerPort
hostPort:containerPort
<l
>containerPort
-P(大写) 随机端口映射
-e 作用是指定容器内的环境变量
-rm 执行docker
run命令带--rm命令选项，等价于在容器退出后，执行docker rm。
<pre class="custom"><code class="hljs">docker run -it
-name myCentos centos
</code></pre>
启动centos 并打开交互终端
<pre class="custom"><code class="hljs">docker run centos&a
sp;-d
</code></pre>
输入<code>docker ps -a</code>查看会发现容器已经退出了，
很重要的一点：docker容器后台运行，就必须有一个前台进程，这是docker机制，
最佳的解决方案是将你要运行的程序以前台进程的形式运行
<pre class="custom"><code class="hljs">docker run -d
entos /bin/bash -c "while true;do
cho hello zzyy;sleep 2;done"
</code></pre>
<code>/bin/bash</code>：默认
<pre class="custom"><code class="hljs">docker run -d
p 8888:8080 tomcat
</code></pre>
这是可以后台运行不退出的 ,因为tomcat执行不退出
<h4 id="toc_h4_21">docker ps</h4>

列出当前所有正在运行的容器
<h4 id="toc_h4_22">exit</h4>
容器停止退出
<h4 id="toc_h4_23">ctrl+P+Q</h4>
容器不停止退出
<h4 id="toc_h4_24">docker stop</h4>
关闭容器
<h4 id="toc_h4_25">docker kill</h4>
强制关闭容器
<h4 id="toc_h4_26">docker rm</h4>
删除已停止的容器
<pre class="custom"><code class="hljs">docker rm -f $
docker ps -aq)
</code></pre>
<pre class="custom"><code class="hljs">docker ps -aq
 xargs docker rm
</code></pre>

xargs 可变参数 |管道符 把上一个命令参数传递给下个命令
删除所有容器
<h4 id="toc_h4_27">docker start</h4>
启动容器
<h4 id="toc_h4_28">docker restart</h4>
重启容器
<h4 id="toc_h4_29">docker logs</h4>
查看容器日志
options:

-t 是加入时间戳
-f 跟随最新的日志打印 一直输入不停止
tail 数字 显示最后
少条
<h4 id="toc_h4_30">docker top</h4>
查看容器内运行的进程
<h4 id="toc_h4_31">docker inspect</h4>
查看容器内部细节
<h4 id="toc_h4_32">docker exec</h4>
在容器中打开新的终端，并且可以启动新的进程，
<pre class="custom"><code class="hljs">docker <span class="hljs-bu
lt_in">exec -it centosContainer ls /tmp
</code></pr
>
执行完返回，不停留再交互终端
<pre class="custom"><code class="hljs">docker <span class="hljs-bu
lt_in">exec -it centosContainer /bin/bash
</code></pre>
停留在交互终端
后面的/bin/bash的作用是表示载入容器后运行bash
<h4 id="toc_h4_33">docker attach</h4>
直接进入容器启动命令的终端，不会启动新的进程，停留在交互终端
<h4 id="toc_h4_34">docker cp</h4>
拷贝容器内的文件到宿主机中
<pre class="custom"><code class="hljs">docker cp centosCon
ainer:/test/a.text /test
/span>
</code></pre>
<h4 id="toc_h4_35">docker commit</h4>
提交容器副本 使之成为一个新的镜像
<pre class="custom"><code class="hljs">docker commit -a=<
pan class="hljs-string">"zjjaxx" -m="fix init"
/span> dabceae191c9 zjjaxx/tomcat:1.1
</code></pre>

<code>-a</code>表示作者 <code>-m</code>表示描述 <code>zjjaxx/</code>表示包名
/p>
<h4 id="toc_h4_36">docker build</h4>
生成一个新的镜像
<pre class="custom"><code class="hljs">docker build -f
/mydocker/dockerfile -t zjj/centos .
</code></pre>
<code>-f</code>指名dockerfile文件路径 <code>-t</code>表示包名+镜像名 <code>.</c
de>表示在当前目录下
<h3 id="toc_h3_37">docker 命令options说明</span
</h3>
<code>-a</code>:表示列出本地所有
<code>-q</code>:只显示id
<code>-f</code>:强制执行
<h2 id="toc_h2_38">docker 镜像相关概念<
h2>
<h3 id="toc_h3_39">UnionFS(联合文件系统)</span
</h3>
Union文件系统（UnionFS）是一种分层、轻量级并且高并发的文件系统，它支持<sp
n>「对文件系统的修改作为一次提交来一层层的叠加」,同时
以将不同目录挂载到同一个虚拟文件系统下。Union文件系统时Docker镜像的基础，镜像可以通过分
来进行继承，基于基础镜像，可以制作各种具体的应用镜像
<h3 id="toc_h3_40">docker 镜像的加载原理
/h3>
liunx 系统分为两个部分：

bootfs 主要包含bootloader和kernel,bootloader主要时引导加载kernel,linux刚启动时会加载b
otfs文件系统，「docker镜像的最底层是bootfs,镜像没有内核，所有镜
公用宿主机内核」
rootfs 在bootfs之上，包含liunx系统中的/
ev,/proc,/bin等标准目录，不同的Linux系统各不相同，所有每个容器为其提供不同的rootfs，对于
个精简的os，rootfs可以很小，所有docker比虚拟机小得多
<h3 id="toc_h3_41">镜像分层</h3>
镜像分层为了复用共享
<h2 id="toc_h2_42">数据卷</h2>
功能：实现容器数据持久化，容器之间共享数据
<h3 id="toc_h3_43">命令添加</h3>
<pre class="custom"><code class="hljs">docker run -it
v /宿主机绝对路径目录:/容器内目录 镜像名
</code></pre>
<pre class="custom"><code class="hljs"> docker run -i
 -d -v /tomcatVolume:/containerVolume --name myTomcat
nbsp;-p 8888:8080 tomcat
</code></pre>
<pre class="custom"><code class="hljs"> docker run -i
 -d -v /tomcatVolume:/containerVolume:ro --name myTomc
t -p 8888:8080 tomcat
</code></pre>
<code>:ro</code>options 表示read-only,在容器中只读，只允许主机中修改
<h2 id="toc_h2_44">数据卷容器</h2>
「命名的容器挂载数据卷，其它容器通过挂载这个实现数据共享，
载数据卷的容器，称之为数据卷容器」
实现不同容器数据的共享
<pre class="custom"><code class="hljs">docker run -it
-name container2 --volumes-from container1 centos
</code
</pre>
<h2 id="toc_h2_45">Dockerfile</h2>
Dockerfile是用来构建Docker镜像的构建文件,是由一些列命令和参数构成的脚本
<h3 id="toc_h3_46">Dockerfile基本规则</h

>

每条保留字指令都必须为大写字母且后面要跟随至少一个参数
指令按照从上到下，顺执行
#表示注释
每条指令都会创建一个新的镜像层，并对镜像进行提交
<h3 id="toc_h3_47">Dockerfile的大致流程<h3>

docker从基础镜像运行一个容器
执行一条指令并对容器做出修改
执行类似docker commit 的操作提交一个新的镜像层
docker在基于刚提交的镜像运行一个新的容器</li
执行Dockerfile中的下一条指令直到所有指令都执行完成
<h3 id="toc_h3_48">Dockerfile保留关键字<h3>
<h4 id="toc_h4_49">FROM</h4>
基础镜像，表示当前新镜像是基于哪个镜像的
<h4 id="toc_h4_50">MAINTAINER</h4>
镜像的维护人员的姓名+邮箱地址
<h4 id="toc_h4_51">RUN</h4>
容器构建时需要运行的命令
<h4 id="toc_h4_52">EXPOSE</h4>
当前容器对外暴露出的端口
<h4 id="toc_h4_53">WORKDIR</h4>
指定在创建容器后，终端默认登入进来的工作目录
<h4 id="toc_h4_54">ENV</h4>
用来在构建镜像过程中设置的环境变量
<h4 id="toc_h4_55">ADD</h4>
将宿主机目录下的文件拷贝进镜像且ADD命令会自动处理URL和解压tar压缩包
<h4 id="toc_h4_56">COPY</h4>
只拷贝不解压
<h4 id="toc_h4_57">VOLUME</h4>
设置数据卷，用于数据保存和持久化工作
<h4 id="toc_h4_58">CMD</h4>
指定一个容器启动时要运行的命令，可以有多个CMD指令，但只有最后一个生效，会被docker rn 之后的参数替换
<h4 id="toc_h4_59">ENTRYPOINT</h4>
和CMD不同的是 docker run 之后的参数会在ENTRYPOINT后面追加
<h4 id="toc_h4_60">ONBUILD</h4>
当构建一个被继承的Dockerfile时运行命令,父镜像在被继承后父镜像的ONBUILD被触发
<h2 id="toc_h2_61">Docker Compose</h>
Compose是一个用于定义和运行多容器Docker应用程序的工具。
<h3 id="toc_h3_62">安装</h3>
官网链接
<h3 id="toc_h3_63">compose 配置参考</h>
默认名称为docker-compose.yml