www.bonc.com.cn

BONC 东方国信

Docker容器技术入门讲解

北京东方国信科技股份有限公司

讲解人：郑行
讲解时间：2022-01-17

让数据改变你我生活

BONC 东方国信

目录/CONTENTS

BONC 东方国信

PART ONE

01 Docker概述

什么是Docker?

BONC 东方国信

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用以及依赖包到一个可移植的镜像中，然后发布到任何流行的 Linux或Windows操作系统的机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

通俗的可以理解为Docker就是 将应用+应用依赖的环境打包运行的技术。

为什么使用Docker?

BONC 东方国信

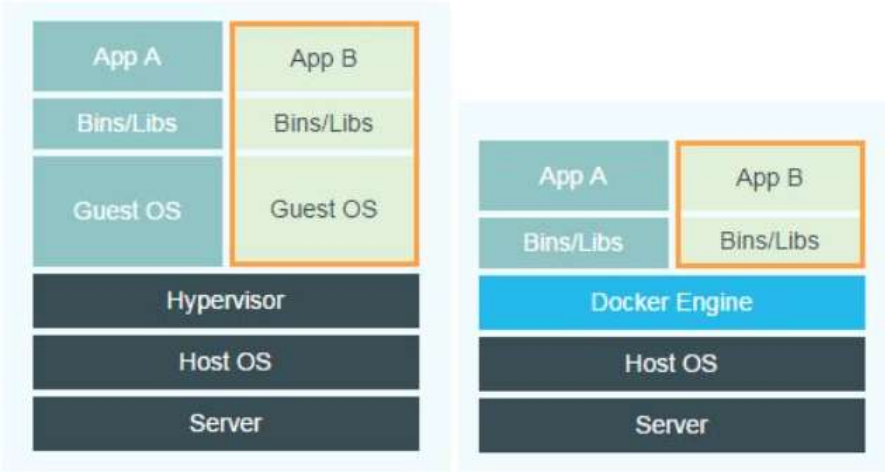
1、Docker 的镜像提供了除内核外完整的运行时环境，确保了应用运行环境一致性，从而不会再出现 “这段代码在我机器上没问题啊” 这类问题；—— 一致的运行环境
2、可以做到秒级、甚至毫秒级的启动时间。大大的节约了开发、测试、部署的时间。—— 更快速的启动时间
3、避免公用的服务器，资源会容易受到其他用户的影响。—— 隔离性
4、善于处理集中爆发的服务器使用压力；—— 弹性伸缩，快速扩展
5、可以很轻易的将在一个平台上运行的应用，迁移到另一个平台上，而不用担心运行环境的变化导致应用无法正常运行的情况。—— 迁移方便
6、使用 Docker 可以通过定制应用镜像来实现持续集成、持续交付、部署。—— 持续交付和部署

Docker VS VM

BONC 东方国信

- 传统虚拟机技术(VM)是虚拟出一套硬件后，在其上运行一个完整操作系统，在该系统上再运行所需应用进程；
- Docker容器技术是应用进程直接运行于宿主的内核，容器内没有自己的内核，而且也没有进行硬件虚拟。因此容器要比传统虚拟机更为轻便。

- 容器是一个应用层抽象，用于将代码和依赖资源打包在一起。 多个容器可以在同一台机器上运行，共享操作系统内核，但各自作为独立的进程在用户空间中运行 。与虚拟机相比， 容器占用的空间较少（容器镜像大小通常只有几十兆），瞬间就能完成启动 。
- 虚拟机 (VM) 是一个物理硬件层抽象，用于将一台服务器变成多台服务器。 管理程序允许多个 VM 在一台机器上运行。每个VM都包含一整套操作系统、一个或多个应用、必要的二进制文件和库资源，因此 占用大量空间 。而且 VM 启动也十分缓慢 。

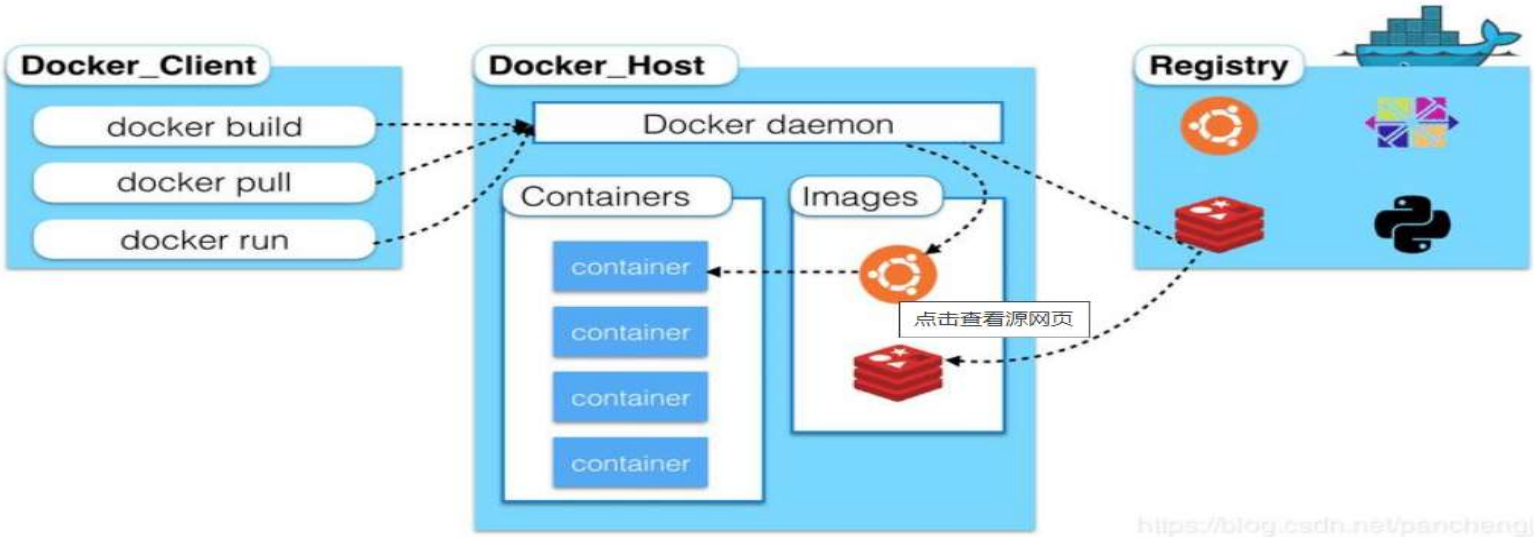


Docker VS VM

BONC 东方国信

对比项	VM	Docker
隔离性	较强	强
标准化	低，难以监视和控制虚拟机运行中的进程或文件变化	高，docker build定义容器结构与进程
计算资源开销	大	小
镜像大小	几百MB至几GB	可小至几MB
启动速度	数秒至数分钟	毫秒级
快速扩展能力	一般	强
跨平台迁移能力	一般	强
对微服务架构的支持	一般	强
对Devops的支持	一般	强

Docker中的基本概念



Docker 包括三个基本概念:

- 镜像（Image）：Docker 镜像相当于一个模板，可以通过这个镜像来创建一个容器。
- 容器（Container）：镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的类和实例一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。
- 仓库（Repository）：仓库可看成一个代码控制中心，用来保存镜像。仓库分为公有仓库和私有仓库。默认为Docker Hub(默认是国外的)，可以通过阿里云，华为云等配置镜像加速。

BONC 东方国信

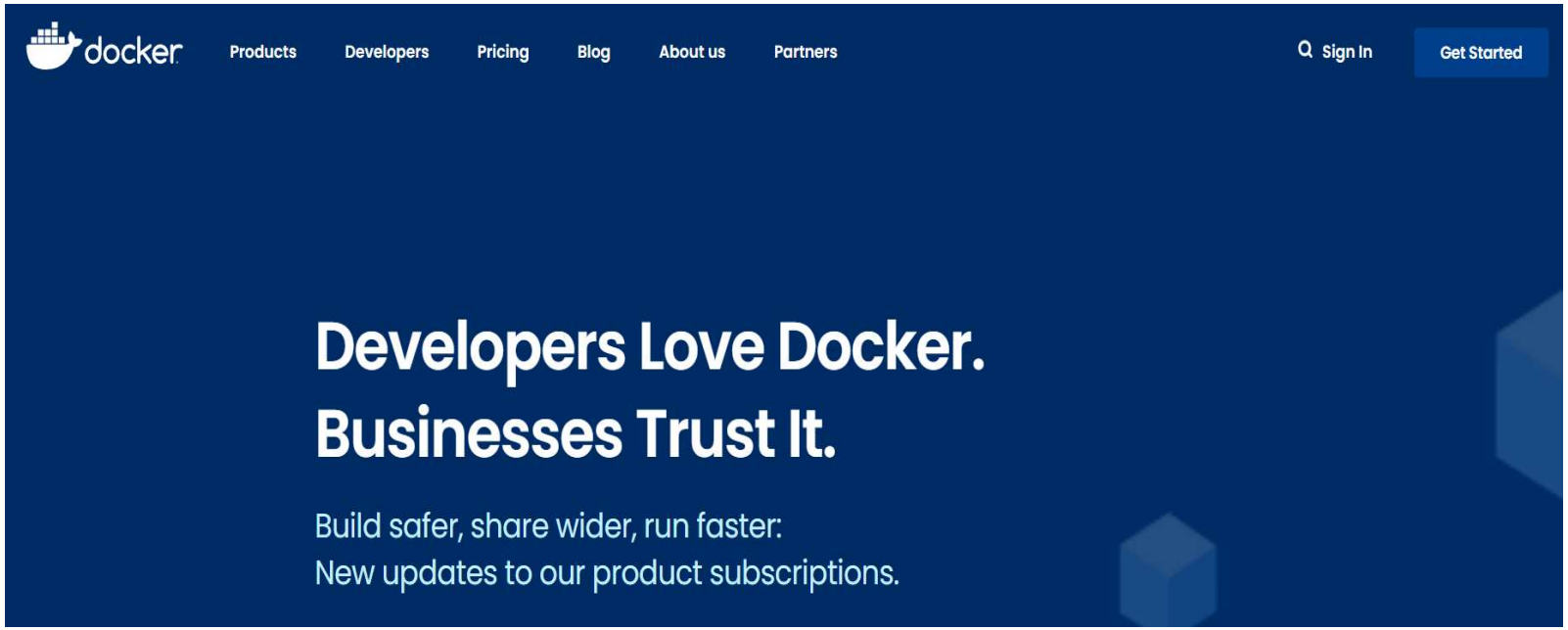
PART TWO

02 Docker安装

Docker相关的网站

BONC 东方国信

docker官网地址：https://www.docker.com/
docker文档地址：https://docs.docker.com/
docker仓库地址：https://hub.docker.com/



Docker支持的环境

BONC 东方国信

Platform	x86_64 / amd64	arm64 / aarch64	arm (32-bit)	s390x
CentOS	✔	✔		
Debian	✔	✔	✔	
Fedora	✔	✔		
Raspbian			✔	
RHEL				✔
SLES				✔
Ubuntu	✔	✔	✔	✔
Binaries	✔	✔	✔	

CentOS环境安装Docker(在线)

BONC 东方国信

1.卸载旧的版本
 sudo yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-engine
2.使用yum安装
 sudo yum install -y yum-utils
3.设置仓库
#国外的仓库地址
 sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
#阿里云的仓库地址
 sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
--(可选)更新yum索引
 sudo yum makecache fast
4.安装docker相关的包 docker-ce 社区版 ee 企业版
 sudo yum install docker-ce docker-ce-cli containerd.io

CentOS环境安装Docker(在线)

BONC 东方国信

--(可选)安装其他版本
yum list docker-ce --showduplicates | sort -r
通过其完全限定的包名称安装特定版本，即包名称 (docker-ce) 加上版本字符串（第 2 列），从第一个冒号 (:) 开始，一直到第一个连字符，用连字符 (-)分隔。例如，docker-ce-18.09.1。
sudo yum install docker-ce-<VERSION_STRING> docker-ce-cli-<VERSION_STRING> containerd.io
5.启动docker
sudo systemctl start docker
6.判断docker是否安装成功
docker version (docker info)
7.测试hello world
sudo docker run hello-world

CentOS环境安装Docker(离线，rpm,yum)

BONC 东方国信

使用rpm包安装docker,有yum或者rpm命令
1.前往https://download.docker.com/linux/centos/ 并选择您的 CentOS 版本。然后浏览 x86_64/stable/Packages/ 并下载.rpm要安装的 Docker 版本的文件。
2.安装 Docker Engine，将下面的路径更改为您下载 Docker 包的路径。
sudo yum install /path/to/package.rpm（docker-ce docker-ce-cli containerd.io docker-ce-rootless docker-scan-plugin）
-- 查找依赖 yum deplist 包名
3.启动 Docker。
sudo systemctl start docker
4.判断docker是否安装成功
docker version (docker info)

```
containerd.io-1.4.12-3.1.el7.x86_64.rpm
docker-ce-20.10.12-3.el7.x86_64.rpm
docker-ce-cli-20.10.12-3.el7.x86_64.rpm
docker-ce-rootless-extras-20.10.12-3.el7.x86_64.rpm
docker-scan-plugin-0.12.0-3.el7.x86_64.rpm
```

CentOS环境卸载Docker

BONC 东方国信

1.卸载 Docker 引擎、CLI 和 Containerd 软件包：
sudo yum remove docker-ce docker-ce-cli containerd.io
2.主机上的映像、容器、卷或自定义配置文件不会自动删除。要删除所有映像、容器和卷：
sudo rm -rf /var/lib/docker
sudo rm -rf /var/lib/containerd

CentOS环境安装Docker(离线)

BONC 东方国信

使用二进制文件安装docker
1.先决条件
- 64位安装
- 版本 3.10 或更高版本的 Linux 内核。建议使用适用于您的平台的最新版本的内核。
- iptables 版本 1.4 或更高版本
- git 1.7 或更高版本
- ps可执行，通常由提供procps或类似的包。
- XZ Utils 4.9 或更高版本
- 正确安装的 cgroupfs 层次结构；一个单一的、包罗万象的 cgroup 安装点是不够的。请参阅 Github 问题 #2683、#3485、#4568
2.下载静态二进制存档。转到 https://download.docker.com/linux/static/stable/ （或更改stable为nightly或test），选择您的硬件平台，然后下载.tgz与要安装的Docker Engine版本有关的文件。
3.使用该tar命令解压文件
 tar zxvf /path/to/<FILE>.tar.gz
4.将二进制文件移至可执行路径上的目录(/usr/bin/)下
$ sudo cp docker/* /usr/bin/

CentOS环境安装Docker(离线)

BONC 东方国信

5.启动Docker守护程序:
 sudo dockerd &
如果需要使用其他选项启动守护程序，请相应地修改以上命令，或者创建并编辑文件/etc/docker/daemon.json 以添加定制配置选项。
6.判断docker是否安装成功
docker version (docker info)

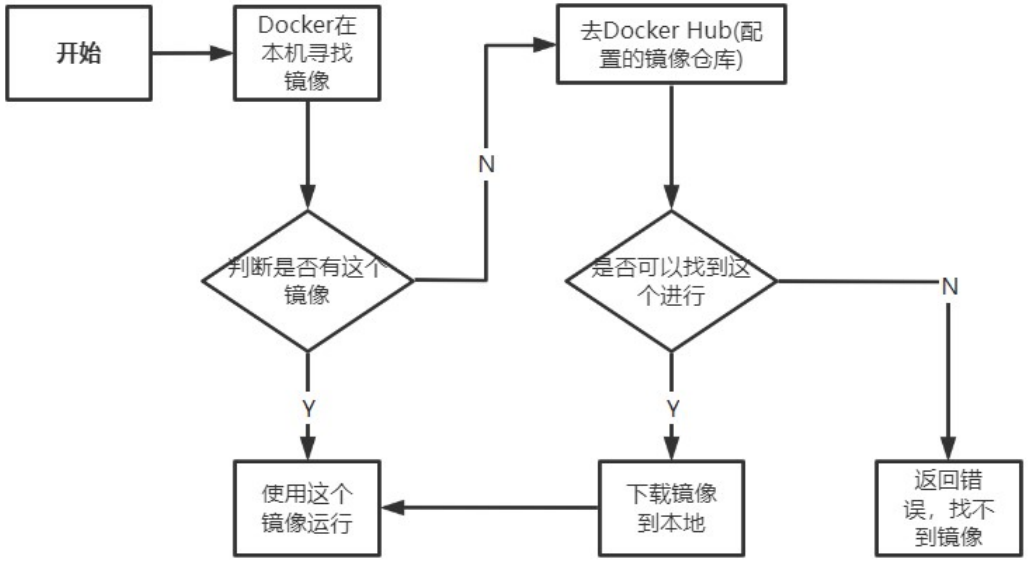
BONC 东方国信

PART THREE

03 Docker常用命令

Docker 工作流程原理

Docker是一个Client-Server结构的系统，Docker的守护进程运行在主机上，通过Socket从客户端访问!DockerServer接收到Docker-Client的指令，就会执行这个命令。



Docker帮助命令

BONC 东方国信

```
sudo docker version     #显示docker的版本信息
sudo docker info        #显示docker的系统信息，包括容器和镜像数量等
sudo docker --help      #帮助命令
```

帮助文档的地址

https://docs.docker.com/engine/reference/run/

Docker镜像命令

BONC 东方国信

sudo docker images
#查看本地镜像
REPOSITORY 镜像的仓库源(名称)
TAG 镜像的标签(版本)
IMAGE ID 镜像的ID
CARETED 镜像的创建时间
SIZE 镜像的大小

可选
-a,--all 列出所有镜像
-q,--queit 只显示id

```
[root@zhengahng5 docker]# sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
hello-world   latest    feb5d9fea6a5   3 months ago    13.3kB
centos        centos7   eeb6ee3f44bd   3 months ago    204MB
centos        latest    5d0da3dc9764   3 months ago    231MB
```

Docker镜像命令

BONC 东方国信

sudo search 镜像名称
#搜索镜像，网络环境下使用

```
[root@zhengahng5 docker]# docker search centos
NAME                              DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
centos                            The official build of CentOS.                   6965      [OK]
ansible/centos7-ansible           Ansible on Centos7                              135                  [OK]
consol/centos-xfce-vnc            Centos container with "headless" VNC session…   133                  [OK]
jdeathe/centos-ssh                OpenSSH / Supervisor / EPEL/IUS/SCL Repos - …   121                  [OK]
```

--filter=STARS=135
#搜索出来的镜像就是STARS大于135

```
[root@zhengahng5 docker]# docker search centos --filter=STARS=135
NAME                      DESCRIPTION                     STARS     OFFICIAL   AUTOMATED
centos                    The official build of CentOS.   6965      [OK]
ansible/centos7-ansible   Ansible on Centos7              135                  [OK]
```

Docker镜像命令

BONC 东方国信

sudo pull 镜像名称:TAG
#下载镜像(拉取镜像)

```
[root@zhengahng5 docker]# docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
72a69066d2fe: Pull complete
93619dbc5b36: Pull complete
99da31dd6142: Pull complete
626033c43d70: Pull complete
37d5d7efb64e: Pull complete
ac563158d721: Pull complete
d2ba16033dad: Pull complete
688ba7d5c01a: Pull complete
00e060b6d11d: Pull complete
1c04857f594f: Pull complete
4d7cfa90e6ea: Pull complete
e0431212d27d: Pull complete
Digest: sha256:e9027fe4d91c0153429607251656806cc784e914937271037f7738bd5b8e7709
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
```

如果不写tag，默认为latest

分层下载，docker images的核心 联合文件系统

签名

真实地址

```
[root@zhengahng5 docker]# docker pull mysql:5.7
5.7: Pulling from library/mysql
72a69066d2fe: Already exists
93619dbc5b36: Already exists
99da31dd6142: Already exists
626033c43d70: Already exists
37d5d7efb64e: Already exists
ac563158d721: Already exists
d2ba16033dad: Already exists
0ceb82207cd7: Pull complete
37f2405cae96: Pull complete
e2482e017e53: Pull complete
70deed891d42: Pull complete
Digest: sha256:f2ad209efe9c67104167fc609cca6973c8422939491c9345270175a300419f94
Status: Downloaded newer image for mysql:5.7
docker.io/library/mysql:5.7
```

Docker镜像命令

BONC 东方国信

sudo docker rmi -f 镜像名称:TAG(镜像id)
#删除镜像
sudo docker rmi -f 镜像id 镜像id 镜像id 镜像id
#删除多个镜像
sudo docker rmi -f $(sudo docker images -aq)
#删除所有镜像

```
[root@zhengahng5 docker]# sudo docker rmi 3218b38490ce
Untagged: mysql:latest
Untagged: mysql@sha256:e9027fe4d91c015342960725165680 6cc784e914937271037f7738bd5b8e7709
Deleted: sha256:3218b38490cec8d31976a40b92e09d61377359eab878db49f025e5d464367f3b
Deleted: sha256:aa81ca46575069829fe1b3c654d9e8feb43b4373932159fe2cad1ac13524a2f5
Deleted: sha256:0558823b9fbe967ea6d7174999be3cc9250b3423036370dc1a6888168cbd224d
Deleted: sha256:a46013db1d31231a0e1bac7eeda5ad4786dea0b1773927b45f92ea352a6d7ff9
Deleted: sha256:af161a47bb22852e9e3caf39f1dcd590b64bb8fae54315f9c2e7dc35b025e4e3
Deleted: sha256:feff1495e6982a7e91edc59b96ea74fd80e03674d92c7ec8a502b417268822ff
```

Docker容器命令

BONC 东方国信

说明：有了镜像才能够创建容器

sudo docker run [可选参数] image

可选

--name="Name"　　容器名称,用来区分容器

-d　　后台方式运行

-it　　使用交互方式运行，进入容器查看内容

-p　　指定容器的端口

　　-p ip:主机端口:容器端口

　　-p 主机端口:容器端口(比较常用)

　　-p 容器端口

-P　　随机指定端口

-v　　把容器中的路径挂在到主机上

　　-v 主机目录:容器内目录

```
[root@zhengahng5 docker]# sudo docker run -it centos:centos7 /bin/bash
[root@da0be6490275 /]# ls -l
total 12
-rw-r--r--.   1 root root 12114 Nov 13  2020 anaconda-post.log
lrwxrwxrwx.   1 root root     7 Nov 13  2020 bin -> usr/bin
drwxr-xr-x.   5 root root   360 Jan 10 11:56 dev
drwxr-xr-x.   1 root root    66 Jan 10 11:56 etc
drwxr-xr-x.   2 root root     6 Apr 11  2018 home
lrwxrwxrwx.   1 root root     7 Nov 13  2020 lib -> usr/lib
lrwxrwxrwx.   1 root root     9 Nov 13  2020 lib64 -> usr/lib64
drwxr-xr-x.   2 root root     6 Apr 11  2018 media
drwxr-xr-x.   2 root root     6 Apr 11  2018 mnt
drwxr-xr-x.   2 root root     6 Apr 11  2018 opt
dr-xr-xr-x. 260 root root     0 Jan 10 11:56 proc
dr-xr-x---.   2 root root   114 Nov 13  2020 root
drwxr-xr-x.  11 root root   148 Nov 13  2020 run
lrwxrwxrwx.   1 root root     8 Nov 13  2020 sbin -> usr/sbin
drwxr-xr-x.   2 root root     6 Apr 11  2018 srv
dr-xr-xr-x.  13 root root     0 Jan  6 12:33 sys
drwxrwxrwt.   7 root root   132 Nov 13  2020 tmp
drwxr-xr-x.  13 root root   155 Nov 13  2020 usr
drwxr-xr-x.  18 root root   238 Nov 13  2020 var
```

Docker容器命令

BONC 东方国信

sudo docker ps
#列出当前正在运行容器

```
[root@zhengahng5 ~]# docker ps -a
CONTAINER ID   IMAGE            COMMAND       CREATED         STATUS                     PORTS     NAMES
da0be6490275   centos:centos7   "/bin/bash"   24 hours ago    Up 24 hours                          intelligent_cohen
f7a902e2a105   centos           "/bin/bash"   24 hours ago    Up 24 hours                          ecstatic_jones
378b8335ce17   hello-world      "/hello"      4 days ago      Exited (0) 4 days ago                nice_jennings
[root@zhengahng5 ~]# docker ps -a -n=1
CONTAINER ID   IMAGE            COMMAND       CREATED         STATUS         PORTS     NAMES
da0be6490275   centos:centos7   "/bin/bash"   24 hours ago    Up 24 hours              intelligent_cohen
```

可选
-a　列出所有的容器(正在运行容器+运行过的容器)
-n=? 显示最近创建的容器
-q　只显示容器id

退出容器
exit　直接停止并退出容器，使用exec时不会停止容器
Ctrl + P + Q 快捷键退出容器，并且不会停止容器。

Docker容器命令

BONC 东方国信

sudo docker rm 容器Id
#删除容器（不会删除运行中的容器，强制删除 -f)
sudo docker rm -f $(sudo docker ps -aq)
#删除所有容器
sudo docker –a –q | xargs sudo docker rm -f
#删除所有容器
sudo docker start 容器Id
#启动容器
sudo docker restart 容器Id
#重启容器
sudo docker stop 容器Id
#停止容器
sudo docker kill 容器Id
#强制停止容器

```
[root@zhengahng5 ~]# docker ps -a
CONTAINER ID   IMAGE           COMMAND       CREATED        STATUS                      PORTS     NAMES
da0be6490275   centos:centos7  "/bin/bash"   24 hours ago   Exited (137) 39 seconds ago           intelligent_cohen
f7a902e2a105   centos          "/bin/bash"   24 hours ago   Up 24 hours                           ecstatic_jones
378b8335ce17   hello-world     "/hello"      4 days ago     Exited (0) 4 days ago                 nice_jennings
[root@zhengahng5 ~]# docker rm 378b8335ce17
378b8335ce17
[root@zhengahng5 ~]# docker ps -a
CONTAINER ID   IMAGE           COMMAND       CREATED        STATUS                      PORTS     NAMES
da0be6490275   centos:centos7  "/bin/bash"   24 hours ago   Exited (137) 58 seconds ago           intelligent_cohen
f7a902e2a105   centos          "/bin/bash"   24 hours ago   Up 24 hours                           ecstatic_jones
```

```
[root@zhengahng5 ~]# docker ps
CONTAINER ID   IMAGE           COMMAND       CREATED        STATUS        PORTS     NAMES
da0be6490275   centos:centos7  "/bin/bash"   24 hours ago   Up 24 hours             intelligent_cohen
f7a902e2a105   centos          "/bin/bash"   24 hours ago   Up 24 hours             ecstatic_jones
[root@zhengahng5 ~]# docker kill da0be6490275
da0be6490275
[root@zhengahng5 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED        STATUS        PORTS     NAMES
f7a902e2a105   centos    "/bin/bash"   24 hours ago   Up 24 hours             ecstatic_jones
```

Docker容器命令

BONC 东方国信

通过docker run –d 镜像名

docker run –d centos

注意：docker容器使用后台运行，就必须要有一个前台进程，docker发现没有应用，就会自动停止。

```
[root@zhengahng5 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@zhengahng5 ~]# docker run -d centos
c610e5be48cab47454b7bdb435d7fcff6a2e61781158409df6a9c8dc7a0f2a56
[root@zhengahng5 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@zhengahng5 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                      PORTS
c610e5be48ca   centos    "/bin/bash"   12 seconds ago   Exited (0) 10 seconds ago
```

Docker容器命令

BONC 东方国信

sudo docker logs [参数] 容器id

参数:

--details 显示提供给日志的额外详细信息

-f, --follow 持续输出文件

--since string 显示自时间戳（例如 2013-01-02T13:23:37Z）或相对时间（例如 42m 42 分钟）以来的日志

-n, --tail string 从日志末尾开始显示的行数（默认为“all”）

-t, --timestamps 显示时间戳

--until string 在时间戳（例如 2013-01-02T13:23:37Z）或相对时间（例如 42m 42 分钟）之前显示日志

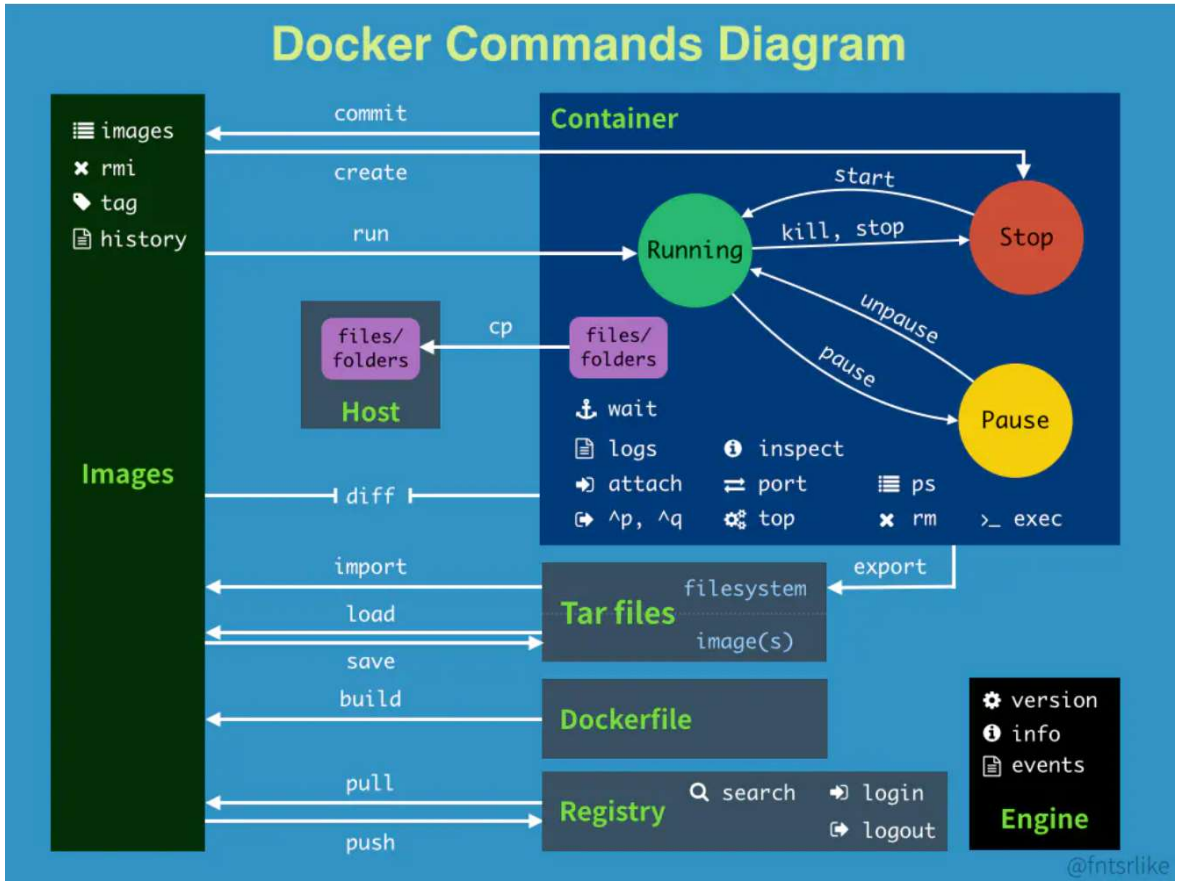
```
[sunxuhui@XCLOUD17 tiangong]$ sudo docker logs -f -t --tail 10 9835db8c3447
2022-01-12T11:05:14.919699774Z 2022/01/12 11:05:14 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:05:44.931421290Z 2022/01/12 11:05:44 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:06:14.965489166Z 2022/01/12 11:06:14 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:06:44.987750434Z 2022/01/12 11:06:44 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:07:14.994555197Z 2022/01/12 11:07:14 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:07:45.006056070Z 2022/01/12 11:07:45 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:08:15.009425100Z 2022/01/12 11:08:15 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:08:45.012923830Z 2022/01/12 11:08:45 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:09:15.016183614Z 2022/01/12 11:09:15 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:09:45.020167845Z 2022/01/12 11:09:45 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:10:15.023880605Z 2022/01/12 11:10:15 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:10:45.027210978Z 2022/01/12 11:10:45 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:11:15.030651310Z 2022/01/12 11:11:15 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
2022-01-12T11:11:45.033727427Z 2022/01/12 11:11:45 Metric client health check failed: the server could not find the requested resource (get services heapster). Retrying in 30 seconds.
```

Docker容器命令

BONC 东方国信

sudo docker top 容器Id
#查看容器中的进程信息
sudo docker inspect 容器Id
#查看镜像的元数据
sudo docker exec –it 容器Id /bin/bash（bashshell）
#进入容器，开启一个新的终端（常用）
docker attach 容器Id
#进入容器，进入正在运行的命令行，exit退出后，如果没有前台进程，将会把容器停止，不会启动新的进程（少用）
docker cp 容器id:容器内路径 主机路径
#从容器内将文件拷贝到主机上
docker cp主机路径 容器id:容器内路径
#从主机上将文件拷贝到容器内，可以理解为scp命令

Docker容器命令


Docker Commands Diagram
images
rmi
tag
history
commit
create
run
Container
start
kill, stop
Running
Stop
unpause
pause
Pause
files/ folders
cp
files/ folders
Host
wait
logs
inspect
attach
port
ps
^p, ^q
top
rm
>_ exec
Images
diff
import
load
save
export
filesystem
Tar files
image(s)
build
Dockerfile
pull
push
Registry
search
login
logout
version
info
events
Engine
@fntsrlike

BONC 东方国信

PART FOUR

04 制作镜像

联合文件系统

BONC 东方国信

1 Docker镜像是什么

镜像是一种轻量级、可执行的独立软件包，用来打包软件运行环境和基于运行环境的开发软件，它包含运行某个软件所需的所有内容，包括代码、运行时、库、环境变量和配置文件。

2 UnionFS(联合文件系统)

Union文件系统(UnionFS) 是一种分层、轻量级并且高性能的文件系统，他支持对文件系统的修改作为一次提交来层层的叠加，同时可以将不同目录挂载到同一个虚拟文件系统下（unite several directories into a single virtual filesystem）。Union文件系统是Docker镜像的基础。镜像可以通过分层来进行集成，基于基础镜像（没有父镜像），可以制作各种具体的应用镜像。

特性：一次同时加载多个文件系统，但从外面看起来，只能看到一个文件系统，联合加载会把各层文件系统叠加起来，这样最终的文件系统会包含所有底层文件和目录。

联合文件系统

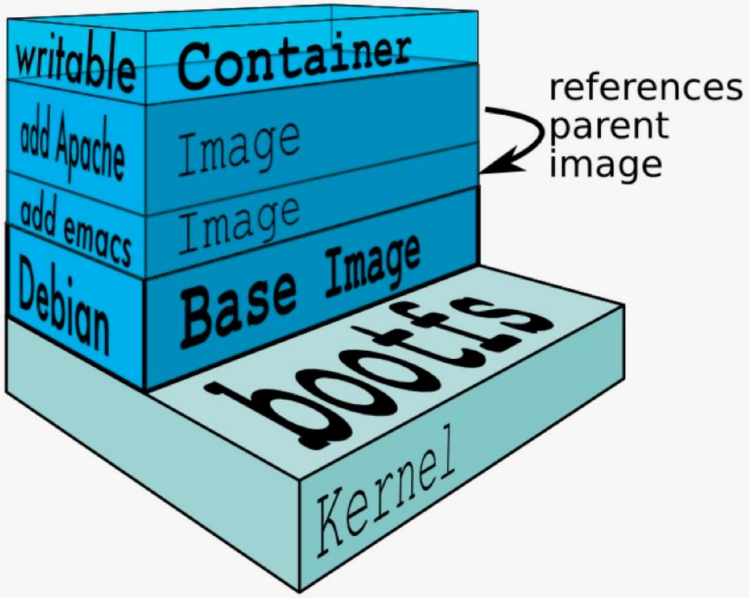
3 Docker 镜像加载原理

docker 的镜像实际上由一层一层的文件系统组成，这种层级的文件系统就是联合文件系统(UnionFS)。

bootfs(boot file system—>boot文件系统) 主要包含bootloader(boot加载器)和kernel(内核)，其中boot加载器主要是用来引导加载内核。Linux刚启动时会加载bootfs(boot文件系统)，在Docker镜像的最底层是bootfs(boot文件系统)。这一层与典型的Linux/Unix系统是一样的，包含boot加载器和内核。当boot加载完成之后整个内核就存在内存中了，此时内存的使用权已由bootfs(boot文件系统)转交给内核，此时系统就会卸载bootfs(boot文件系统)。

roorfs （root file system—>root文件系统），在bootfs(boot文件系统)之上。包含的就是典型Linux系统中的 /dev ，/proc，/bin ，/etc 等标准的目录和文件。rootfs(root文件系统)就是各种不同的操作系统发行版，比如Ubuntu，Centos等等。

对于不同的Linux发行版，bootfs(boot文件系统)基本是一致的，rootfs(root文件系统)会有差别，因此不同的发行版(如Ubuntu，Centos等)可以公用bootfs(boot文件系统)。



commit命令

sudo docker commit 提交容器成为一个镜像
sudo docker commit –a=“作者” –m=“提示信息” 容器id 镜像名称:[TAG]

```
[root@zhengahng5 ~]# sudo docker commit -a="zhenghang" -m="add test.java" 7918a6323a2a centos_test
sha256:68758c6784e2dd98ed9c1e15f8ec6b8b320eae7f22259621e05bf248e77ce99e
```

save load命令

sudo docker save –o 文件路径+文件名称 镜像名称(镜像Id)

sudo docker load –i文件路径+文件名称

```
[root@zhengahng5 ~]# ll
总用量 8
-rw-------. 1 root root 1760 1月   7 03:50 anaconda-ks.cfg
-rw-r--r--. 1 root root 1788 1月   6 19:53 initial-setup-ks.cfg
-rw-r--r--. 1 root root    0 1月  12 19:29 test1.txt
-rw-r--r--. 1 root root    0 1月  12 19:27 test.java
[root@zhengahng5 ~]# docker save -o centos.tar centos:centos7
[root@zhengahng5 ~]# ll
总用量 206744
-rw-------. 1 root root      1760 1月   7 03:50 anaconda-ks.cfg
-rw-------. 1 root root 211696640 1月  16 14:21 centos.tar
-rw-r--r--. 1 root root      1788 1月   6 19:53 initial-setup-ks.cfg
-rw-r--r--. 1 root root         0 1月  12 19:29 test1.txt
-rw-r--r--. 1 root root         0 1月  12 19:27 test.java
[root@zhengahng5 ~]#
```

```
[root@zhengahng5 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
tomcat       latest    fb5657adc892   3 weeks ago    680MB
centos       centos7   eeb6ee3f44bd   4 months ago   204MB
[root@zhengahng5 ~]# docker rmi centos:centos7
Untagged: centos:centos7
Untagged: centos@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987
Deleted: sha256:eeb6ee3f44bd0b5103bb561b4c16bcb82328cfe5809ab675bb17ab3a16c517c9
Deleted: sha256:174f5685490326fc0a1c0f5570b8663732189b327007e47ff13d2ca59673db02
[root@zhengahng5 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
tomcat       latest    fb5657adc892   3 weeks ago   680MB
[root@zhengahng5 ~]# docker load -i centos.tar
174f56854903: Loading layer [==================================================>]  211.7MB/211.7MB
Loaded image: centos:centos7
[root@zhengahng5 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
tomcat       latest    fb5657adc892   3 weeks ago    680MB
centos       centos7   eeb6ee3f44bd   4 months ago   204MB
[root@zhengahng5 ~]#
```

Dockerfile

Dockerfile 是用来构建docker镜像的文件，命令参数脚本。
构建步骤：
1.编写一个Dockerfile文件
2.docker build 构建成为一个镜像
sudo docker build –f Dockerfile文件名称 –t 镜像名称:[TAG] .
如果Dockerfile没有改名，可以不加-f,不要忘记最后的点，这是指定工作路径

Dockerfile编写的注意事项
1.每个保留关键字（指令）都是大写字母。
2.执行顺序从上到下依次执行
3.#表示注释
4.每一个命令都会创建提交成一个新的镜像层。
尽量将相同的步骤（环境变量、配置等操作写在前边），从而能够充分的利用docker缓存。

FROM 指令的格式为：

FROM <image> 或者 FROM <image>:<tag>

FROM指令的功能是为后面的指令提供基础镜像，因此Dockerfile必须以FROM指令作为第一条非注释指令。从公共镜像库中拉取镜像很容易,基础镜像可以选择任何有效的镜像。在一个Dockerfile中FROM指令可以出现多次，这样会构建多个镜像。tag的默认值是latest，如果参数image或者tag指定的镜像不存在，则返回错误。

RUN指令格式1：

RUN <command> (shell格式)

RUN指令格式2：

RUN ["executable", "param1", "param2"] (exec格式，推荐使用)

RUN指令会在前一条命令创建出的镜像的基础上创建一个容器，并在容器中运行命令，在命令结束运行后提交容器为新镜像，新镜像被Dockerfile中的下一条指令使用。通常用来下载依赖包。

RUN指令的两种格式表示命令在容器中的两种运行方式。当使用shell格式时，命令通过/bin/sh -c运行。

当使用exec格式时，命令是直接运行的，容器不调用shell程序，即容器中没有shell程序。

COPY 指令格式:

COPY <src> <dest>

COPY指令复制所指向的文件或目录，将它添加到新镜像中，复制的文件或目录在镜像中的路径是<dest>。<src>所指定的源可以有多个。<src>可以使用通配符指向所有匹配通配符的文件或目录，例如，**"COPY home* /mydir/"** 表示添加所有以"hom"开头的文件到目录/mydir/中。

当<src>指定多个源时，<dest>必须是目录。如果<dest>不存在，则路径中不存在的目录会被创建。

ADD 的优点：在执行 <源文件> 为 tar 压缩文件的话，压缩格式为 gzip, bzip2 以及 xz 的情况下，会自动复制并解压到 <目标路径>。

ADD 的缺点：在不解压的前提下，无法复制 tar 压缩文件。会令镜像构建缓存失效，从而可能会令镜像构建变得比较缓慢。具体是否使用，可以根据是否需要自动解压来决定。

Dockerfile

WORKDIR指令格式:

WORKDIR /path/to/workdir

WORKDIR指令设置工作目录，它之后的RUN、CMD、ENTRYPOINT、COPY以及ADD指令都会在这个工作目录下运行。如果这个工作目录不存在，则会自动创建一个。

WORKDIR指令可在Dockerfile中多次使用。如果提供了相对路径，则它将相对于上一个WORKDIR指令的路径。

例如：

WORKDIR /a
WORKDIR b
WORKDIR c

当前使用的路径为 **/a/b/c**

EXPOSE指令格式:

EXPOSE <port> [<port>/<protocol>...]

EXPOSE指令通知Docker该容器在运行时侦听指定的网络端口。可以指定端口是侦听TCP还是UDP，如果未指定协议，则默认值为TCP。

这个指令仅仅是声明容器打算使用什么端口而已，并不会自动在宿主机进行端口映射，可以在运行的时候通过docker -p指定

CMD指令格式1：

CMD <command> (shell格式)

CMD指令格式2：

CMD ["executable", "param1", "param2"] (exec格式，推荐使用)

CMD指令格式3：

CMD ["param1", "param2"] (为ENTRYPOINT指令提供参数)

CMD指令提供容器运行时的默认值，这些默认值可以是一条指令，也可以是一些参数。一个Dockerfile中可以有多条CMD指令，但只有最后一条CMD指令有效。

CMD ["param1", "param2"] 格式是在CMD指令和ENTRYPOINT指令配合时使用的，CMD指令中的参数会添加到ENTRYPOING指令中.使用shell和exec格式时，命令在容器中的运行方式与RUN指令相同。

不同之处在于，RUN指令在构建镜像时执行命令，并生成新的镜像；CMD指令在构建镜像时并不执行任何命令，而是在容器启动时默认将CMD指令作为第一条执行的命令。如果用户在命令行界面运行docker run命令时指定了命令参数，则会覆盖CMD指令中的命令。

ENTRYPOINT指令格式1：

ENTRYPOINT <command> (shell格式)

ENTRYPOINT指令格式2：

ENTRYPOINT ["executable", "param1", "param2"] (exec格式，推荐格式)

ENTRYPOINT指令和CMD指令类似，都可以让容器在每次启动时执行相同的命令，但它们之间又有不同。一个Dockerfile中可以有多条ENTRYPOINT指令，但只有最后一条ENTRYPOINT指令有效。

docker run命令提供的运行命令参数可以覆盖CMD，但不能覆盖ENTRYPOINT，只会在后边添加参数，比如:

CMD [“/bin/bash”,”ls -a”],docker run 后边添加-l 会报错

ENTRYPOINT [“/bin/bash”, “ls”, “-a”]。docker run 后边添加-l,会变成/bin/bash ls –a –l。

搭建镜像仓库registry

BONC 东方国信

docker 搭建私人镜像仓库

需要准备一台机器作为私人仓库服务器假设地址为192.168.72.200；需要服务端（要使用镜像的机器），假设地址为182.168.72.201。

服务器端需要的操作：

1. 从可以联网的docker上pull registry

```
$ docker pull registry:2
```

2. 保存镜像为归档文件

```
$ docker save -o registry.tar registry:2
```

3. 上传镜像到内网docker

```
$ docker load -i  registry.tar
```

4. 启动容器

```
$ docker run -d -v /root/docker/registry:/var/lib/registry -p 5000:5000 --name myregistry
--restart=always registry:2
```

搭建镜像仓库registry

客户端需要的操作：

1. 配置私有仓库地址

```
vim /etc/docker/daemon.json # 严格的json文件格式
加入以下内容：
{
"insecure-registries": ["192.168.72.200:5000"] # 私有仓库服务器的地址和端口（-p映射的端口）
}
```

2. 重启docker服务

```
$ systemctl daemon-reload # 重载unit配置文件

$ systemctl restart docker # 重新启动Docker
```

搭建镜像仓库registry

使用私有仓库

1. push 操作

```
# 修改镜像标签为服务器地址:端口/镜像名:tag; docker会按照标签的地址和端口推送镜像
$ docker tag busybox:latest 192.169.72.200:5000/busybox:v0.1

# 上传镜像
$ docker push 192.169.72.200:5000/busybox:v0.1
```

2. pull 操作

```
# 镜像为服务器地址:端口/镜像名
$ docker pull 192.169.72.200:5000/busybox:v0.1
```

3. 查询镜像

```
# 服务器的地址和端口
$ curl http://192.169.72.200:5000/v2/_catalog

# 查询镜像版本 curl http://your-server-ip:5000/v2/your-image-name/tags/list
$ curl http://192.169.72.200:5000/v2/busybox/tags/list
```

BONC 东方国信

Thank You

感谢观看

中国 北京 朝阳区创达三路1号东方国信大厦
BONC，NO.1 Chuangda 3rd Road，Chaoyang District，Beijing，China，100102
T: +86-10-8486666 F:+86-10-64398978
E:investor@bonc.com.cn
www.bonc.com.cn

让数据改变你与世界