

# Q/DX

## 青 岛 鼎 信 通 讯 股 份 有 限 公 司 技 术 文 档

Q/DX D121.089-2022

---



Docker 容器部署规范

V1.0

2022 - 12 - 21 发布

2022 - 12 - 21

青 岛 鼎 信 通 讯 股 份 有 限 公 司      发 布

## 目 次

1 范围 .....	2
2 规范性引用文件 .....	2
3 规范概述 .....	2
4 Docker 简介.....	2
4.1 Docker 简介.....	2
4.2 相关概念 .....	2
4.3 逻辑结构 .....	3
4.4 优缺点 .....	3
5 部署 .....	3
5.1 Windows10 系统下部署.....	3
5.1.1 安装文件下载 .....	3
5.1.2 安装 Hyper-V.....	4
5.1.3 开启 Hyper-V.....	4
5.1.4 安装 Docker Desktop for Windows.....	4
5.1.5 运行安装文件.....	4
5.2 Linux 系统下部署.....	6
5.2.1 centos 7 版本.....	6
5.2.2 安装 yum-config-manager.....	6
5.2.3 设置 yum 的下载资源仓库 .....	6
5.2.4 docker 安装.....	7
5.2.5 docker 启动.....	8
5.2.6 docker 安装成功, 版本查看.....	8
6 Docker 相关命令.....	8
6.1 配置加速器(网易云) .....	8
6.1.1 修改配置.....	8
6.1.2 重启 docker .....	9
6.2 基础操作 .....	9
6.2.1 查询 docker 信息.....	9
6.2.2 查看 docker 版本.....	9
6.2.3 查看命令帮助 .....	9
6.3 镜像操作.....	9
6.3.1 列出所有镜像.....	9
6.3.2 镜像查询 .....	9
6.3.3 Docker 仓库地址 .....	10
6.3.4 安装镜像.....	10
6.3.5 删除镜像.....	10
6.3.6 保存镜像.....	10

6.4 容器操作.....	10
6.4.1 查看容器.....	10
6.4.2 新建并运行容器.....	10
6.4.3 启停容器 .....	11
6.4.4 进入容器（正在运行） .....	11
6.4.5 执行命令 .....	11
6.4.6 退出容器 .....	12
6.4.7 删除容器.....	12
6.4.8 查看容器日志.....	12
6.4.9 查看容器内运行的进程.....	12
6.4.10 查看容器详细信息.....	12
6.4.11 容器文件拷贝.....	12
6.4.12 容器安装 vim.....	13
6.4.13 容器导出.....	13
6.4.14 容器导入.....	13
6.5 容器数据卷.....	13
6.5.1 添加数据卷.....	13
6.5.2 数据卷权限.....	13
6.5.3 查看容器信息.....	14
6.5.4 数据卷容器 .....	14

## 前 言

为保证工程技术本部研发的自动化软件系统的规范性对Docker容器的部署使用提出规范。

本规范定义了青岛鼎信通讯股份有限公司、青岛鼎信通讯消防安全有限公司、青岛鼎信通讯科技有限公司及相关公司的自动化软件系统Docker容器部署的建议和描述。

本标准由青岛鼎信通讯股份有限公司工程技术本部自动化部软件室起草。



# Docker 容器部署规范

## 1 范围

本规范定义了青岛鼎信通讯股份有限公司、青岛鼎信通讯消防安全有限公司、青岛鼎信通讯科技有限公司及相关公司的自动化系统的Docker容器部署规范及指导。

本规范适用于工程技术本部自动化部。

## 2 规范性引用文件

本标准根据工作实际情况进行整理，并进行规范要求，无其他参考整理。

## 3 规范概述

规范主要包含以下几方面：

- (1) Docker容器简介
- (2) Docker容器部署流程
- (3) 相关命令

## 4 Docker 简介

### 4.1 Docker 简介

Docker 是一个开源的应用容器引擎，基于Go语言 并遵从 Apache2.0 协议开源。Docker 可以让开发者打包他们的应用以及依赖包到一个轻量级、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。容器是完全使用沙箱机制，相互之间不会有任何接口，更重要的是容器性能开销极低。

### 4.2 相关概念

容器是docker本身是镜像运行的实例，与虚拟机相同都是具有资源隔离和分配的能力，只不过容器虚拟的是操作系统，而虚拟机虚拟的是硬件系统，容器与Linux文件系统非常类似，容器一般用于将软件的代码和依赖资源打包成标准化的单元用于开发，交互和部署，用户可以方便地创建和使用容器，把自己的应用镜像放入docker成为容器对象。容器还可以进行版本管理、复制、分享、修改，就像管理普通的代码一样。

镜像就是指容器中的软件包，包含所有软件运行所需要的内容。镜像的实质是一个特殊的文件系统，Docker采用分层储存架构，镜像的构建时是由多层文件系统一层一层依次构建，任何一层的改变只在自己这一层而如果其他层也有相同的文件，那么其他层只会被打上标记不会真正的被操作，所以构建镜像时尽量使得每一层只有需要添加的东西，不需要的东西在构建结束前清理干净。

仓库就是集中存放镜像资源的地方，镜像构建成功后可以很方便的在当前宿主机上运行，但是要想其他服务器也可以方便的使用到这些镜像，就需要一个仓库对镜像进行统一的存储和管理，Docker Registry 服务就提供了构建仓库的服务，可以包含多个仓库，为每一个仓库分配标签，这就是为什么Docker由类似

于版本管理的功能。Docker Registry服务有公开服务和私有服务，一般公开服务允许用户免费上传下载公开的镜像例如常见的公开服务Docker Hub。

### 4.3 逻辑结构

Docker采用的是C/S架构，Docker客户端向服务端(docker-daemon)发送指令，docker-daemon负责构建，运行以及分发docker容器，docker的客户端与服务端可以运行在同一台主机上，也可以使用docker的客户端连接远程的服务端，客户端与服务端使用RSET API通信，也可以使用Unix套接字，或者是网络接口。另外还可以使用docker compose作为客户端，它可以控制一组docker容器的应用程序。

### 4.4 优缺点

优点：

- (1) 快速部署：短时间内可以部署成百上千个应用，更快速交付到线上。
- (2) 高效虚拟化：不需要额外的 hypervisor 支持，直接基于 linux 实现应用虚拟化，相比虚拟机大幅提高性能和效率。
- (3) 节省开支：提高服务器利用率，降低 IT 支出。
- (4) 简化配置：将运行环境打包保存至容器，使用时直接启动即可。
- (5) 快速迁移和扩展：可夸平台运行在物理机、虚拟机、公有云等环境，良好的兼容性可以方便将应用从 A 宿主机迁移到 B 宿主机，甚至是 A 平台迁移到 B 平台。

缺点：

- (1) 隔离性：各应用之间的隔离不如虚拟机彻底

## 5 部署

Docker 并非是一个通用的容器工具，它依赖于已存在并运行的 Linux 内核环境。Docker 实质上是在已经运行的 Linux 下制造了一个隔离的文件环境，因此它执行的效率几乎等同于所部署的 Linux 主机。因此，Docker 必须部署在 Linux 内核的系统上。如果其他系统想部署 Docker 就必须安装一个虚拟 Linux 环境。

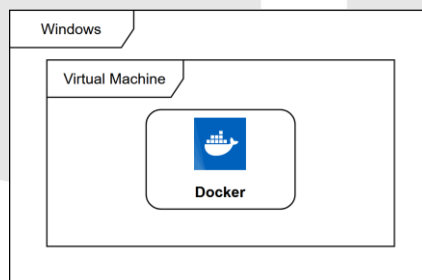


图 1 Docker 部署

### 5.1 Windows10 系统下部署

#### 5.1.1 安装文件下载

Docker Desktop 是 Docker 在 Windows 10 和 macOS 操作系统上的官方安装方式，这个方法依然属于先在虚拟机中安装 Linux 然后再安装 Docker 的方法。

Docker Desktop 官方下载地址：<https://hub.docker.com/editions/community/docker-ce-desktop-windows>

注意：此方法仅适用于 Windows 10 操作系统专业版、企业版、教育版和部分家庭版！

### 5.1.2 安装 Hyper-V

Hyper-V 是微软开发的虚拟机，类似于 VMWare 或 VirtualBox，仅适用于 Windows 10。这是 Docker Desktop for Windows 所使用的虚拟机。

但是，这个虚拟机一旦启用，QEMU、VirtualBox 或 VMWare Workstation 15 及以下版本将无法使用！如果你必须在电脑上使用其他虚拟机（例如开发 Android 应用必须使用的模拟器），请不要使用 Hyper-V！

### 5.1.3 开启 Hyper-V

右键开始菜单，依次选择【应用和功能】--【程序和功能】--【启用或关闭 Windows 功能】--【选中 Hyper-V】，如下图：

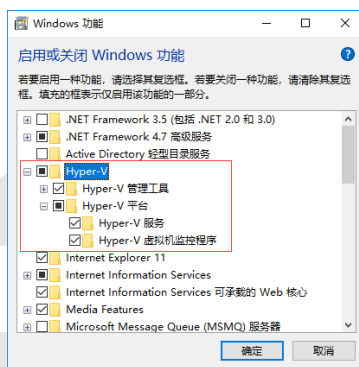


图 2 Hyper-V 设置

也可以通过命令来启用 Hyper-V，请右键开始菜单并以管理员身份运行 PowerShell，执行以下命令：

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
```

### 5.1.4 安装 Docker Desktop for Windows

点击 Get started with Docker Desktop，并下载 Windows 的版本，如果你还没有登录，会要求注册登录：

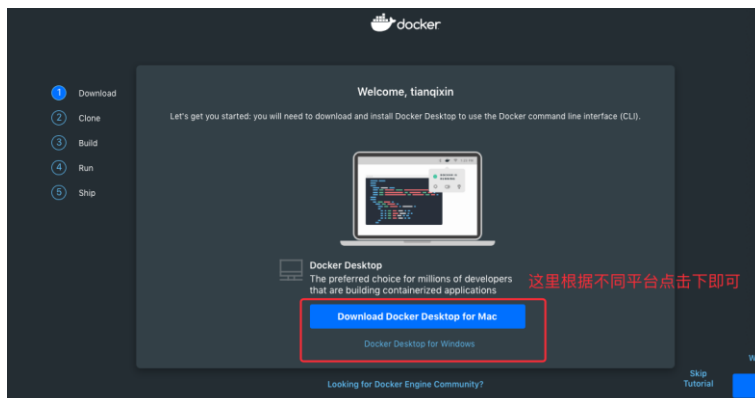


图 3 windows docker 下载界面

### 5.1.5 运行安装文件

双击下载的 Docker for Windows Installer 安装文件，一路 Next，点击 Finish 完成安装。

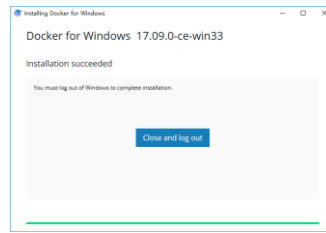



图 4 docker for windows installer 安装界面

安装完成后，Docker 会自动启动。通知栏上会出现个小鲸鱼的图标，这表示 Docker 正在运行。桌边也会出现三个图标，如下图所示。我们可以在命令行执行 `docker version` 来查看版本号，`docker run hello-world` 来载入测试镜像测试。如果没启动，你可以在 Windows 搜索 Docker 来启动：

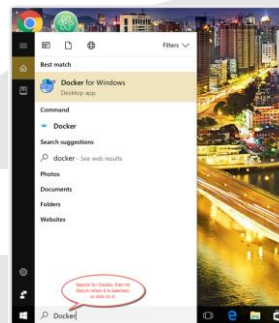


图 5 搜索 Docker

启动后，也可以在通知栏上看到小鲸鱼图标：

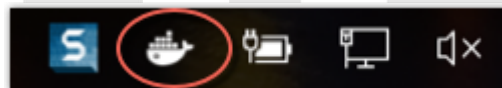


图 6 docker 图标

安装之后，可以打开 PowerShell 并运行以下命令检测是否运行成功：

```
docker run hello-world
```

在成功运行之后应该会出现以下信息：

```
Windows PowerShell
PS C:\Users\Runoob> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdc26d7: Pull complete
Digest: sha256:e7c70bb24b462baa86c102610182e3efcb12a04854e8c582838d92970a09f323
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

PS C:\Users\Runoob>
```



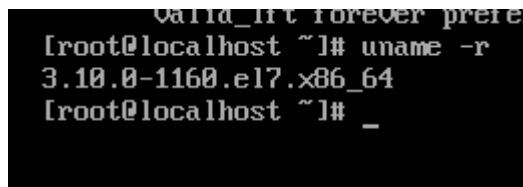
图 7 docker 运行成功界面

## 5.2 Linux 系统下部署

以 Centos 7 为例进行安装。

### 5.2.1 centos 7 版本

Docker 要求 CentOS 系统的内核版本高于 3.10，查看本页面的前提条件来验证你的 CentOS 版本是否支持 Docker。通过 `uname -r` 命令查看当前的内核版本。



```
[root@localhost ~]# uname -r
3.10.0-1160.el7.x86_64
[root@localhost ~]# _
```

图 8 centos 查看

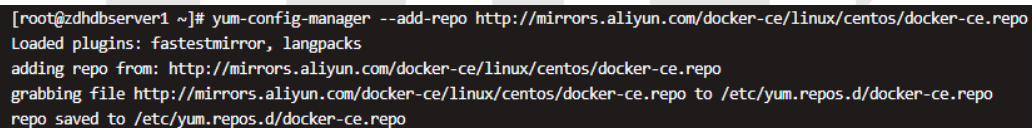
### 5.2.2 安装 yum-config-manager

执行指令：

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

### 5.2.3 设置 yum 的下载资源仓库

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



```
[root@zdhdbserver1 ~]# yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror, langpacks
adding repo from: http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
grabbing file http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
```

图9 设置yum

如果报错先安装：

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

```

[root@localhost ~]# yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
-bash: yum-config-manager: 未找到命令
[root@localhost ~]# yum -y install yum-utils
已加载插件: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.bfsu.edu.cn
 * extras: mirrors.bfsu.edu.cn
 * updates: mirrors.bfsu.edu.cn
正在解决依赖关系
--> 正在检查事务
--> 软件包 yum-utils.noarch.0.1.1.31-54.el7_8 将被 安装
--> 正在处理依赖关系 python-kitchen, 它被软件包 yum-utils-1.1.31-54.el7_8.noarch 需要
--> 正在处理依赖关系 libxml2-python, 它被软件包 yum-utils-1.1.31-54.el7_8.noarch 需要
--> 正在检查事务
--> 软件包 libxml2-python.x86_64.0.2.9.1-6.el7.5 将被 安装
--> 正在处理依赖关系 libxml2 = 2.9.1-6.el7.5, 它被软件包 libxml2-python-2.9.1-6.el7.5.x86_64 需要
--> 软件包 python-kitchen.noarch.0.1.1.1-5.el7 将被 安装
--> 正在处理依赖关系 python-chardet, 它被软件包 python-kitchen-1.1.1-5.el7.noarch 需要
--> 正在检查事务
--> 软件包 libxml2.x86_64.0.2.9.1-6.el7_2.3 将被 升级
--> 软件包 libxml2.x86_64.0.2.9.1-6.el7.5 将被 更新
--> 软件包 python-chardet.noarch.0.2.2.1-3.el7 将被 安装
--> 解决依赖关系完成

[root@localhost ~]# yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
已加载插件: fastestmirror
adding repo from: http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
grabbing file http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo to /etc/yum/repos.d/docker-ce.repo
repo saved to /etc/yum/repos.d/docker-ce.repo

```

图10 yum-config-manager

#### 5.2.4 docker 安装

查看是否已安装 docker:

yum list installed | grep docker

```

[root@localhost ~]# yum list installed | grep docker
[root@localhost ~]#

```

图 11 查看是否已安装 docker

安装 docker:

yum -y install docker

```

已安装:
docker.x86_64 2:1.13.1-208.git7d71120.el7_9

作为依赖被安装:
PyYAML.x86_64 0:3.10-11.el7
container-selinux.noarch 2:2.119.2-1.911c772.el7_8
docker-common.x86_64 2:1.13.1-208.git7d71120.el7_9
libnl.x86_64 0:1.1.4-3.el7
oci-register-machine.x86_64 1:0-6.git2b44233.el7
python-IPy.noarch 0:0.75-6.el7
python-dmidecode.x86_64 0:3.12.2-4.el7
python-pytoml.noarch 0:0.1.14-1.git7dea353.el7
setools-libs.x86_64 0:3.3.8-4.el7
subscription-manager-rhsm-certificates.x86_64 0:1.24.48-1.el7.centos
atomic-registries.x86_64 1:1.22.1-33.gitb507039.el7_8
container-storage-setup.noarch 0:0.11.0-2.git5eaf76c.el7_8
fuse-overlayfs.x86_64 0:0.7.2-6.el7_8
libseccomp.x86_64 0:2.3.1-4.el7
oci-systemd-hook.x86_64 1:0.2.0-1.git05e6923.el7_6
python-backports.x86_64 0:1.0-8.el7
python-ethnettool.x86_64 0:0.8-8.el7
python-setuptools.noarch 0:0.9.8-7.el7
slirp4netns.x86_64 0:0.4.3-4.el7_8
usermode.x86_64 0:1.111-6.el7

作为依赖被升级:
policycoreutils.x86_64 0:2.5-34.el7

```

图 12 安装 docker

如果安装失败, 使用以下指令删除 docker:

sudo yum remove docker \

```
docker-client \  
docker-client-latest \  
docker-common \  
docker-latest \  
docker-latest-logrotate \  
docker-logrotate \  
docker-engine \  
docker-selinux
```

### 5.2.5 docker 启动

#### (1) 暂时启动

```
systemctl start docker
```

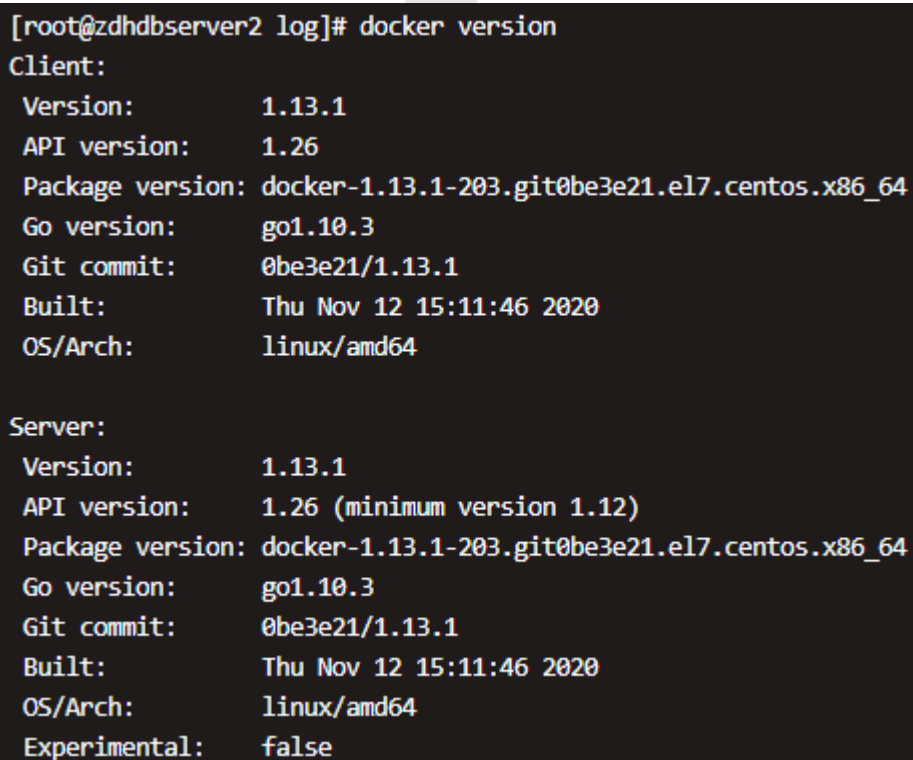
#### (2) 永久启动

```
systemctl enable docker
```

### 5.2.6 docker 安装成功，版本查看

执行指令：

```
docker version
```



```
[root@zdhdbserver2 log]# docker version  
Client:  
Version:      1.13.1  
API version:  1.26  
Package version: docker-1.13.1-203.git0be3e21.el7.centos.x86_64  
Go version:   go1.10.3  
Git commit:   0be3e21/1.13.1  
Built:        Thu Nov 12 15:11:46 2020  
OS/Arch:      linux/amd64  
  
Server:  
Version:      1.13.1  
API version:  1.26 (minimum version 1.12)  
Package version: docker-1.13.1-203.git0be3e21.el7.centos.x86_64  
Go version:   go1.10.3  
Git commit:   0be3e21/1.13.1  
Built:        Thu Nov 12 15:11:46 2020  
OS/Arch:      linux/amd64  
Experimental: false
```

图13 docker版本查看

## 6 Docker 相关命令

### 6.1 配置加速器（网易云）

#### 6.1.1 修改配置

修改文件：/etc/docker/daemon.json

```
{"registry-mirrors":["http://hub-mirror.c.163.com"]}
```

配置好后需要重启。

注意：这里使用 CentOS7，CentOS6 的配置方法不同。没有文件则新建。

### 6.1.2 重启 docker

```
$ systemctl daemon-reload  
$ systemctl restart docker
```

## 6.2 基础操作

### 6.2.1 查询 docker 信息

```
docker info
```

### 6.2.2 查看 docker 版本

```
$ docker --version
```

### 6.2.3 查看命令帮助

```
$ docker --help  
$ man docker-images #查看镜像命令说明
```

## 6.3 镜像操作

### 6.3.1 列出所有镜像

```
$ docker images  
$ docker images -a #列出所有镜像  
$ docker images -aq #列出所有镜像 ID
```

### 6.3.2 镜像查询

根据镜像名查询仓库中所有镜像

```
# docker search [OPTIONS] 镜像名  
# docker search tomcat  
# docker search -s 30 tomcat #查找 30 颗星以上的镜像
```

### 6.3.3 Docker 仓库地址

```
https://hub.docker.com/
```

### 6.3.4 安装镜像

```
$ docker pull nginx:latest
```

### 6.3.5 删除镜像

```
$ docker rmi -f 3fa112fd3642 #镜像 ID
```

### 6.3.6 保存镜像

将自定义的一个容器保存成一个镜像

```
docker commit -m="delete docs" -a="gongxr" e218edb10161 gxr/tomcat:1.0
```

各个参数说明：

- (1) -m: 提交的描述信息
- (2) -a: 指定镜像作者
- (3) e218edb10161: 容器 ID
- (4) gxr/tomcat:1.0: 指定要创建的镜像名和版本号

## 6.4 容器操作

### 6.4.1 查看容器

```
$ docker ps -a # 查看所有容器  
$ docker ps # 查看正在运行的容器
```

### 6.4.2 新建并运行容器

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]  
$ docker run -d --name nginx-test -p 18080:80 nginx
```

参数说明:

- (1) --name nginx-test: 容器名称。
- (2) -p 18080:80: 端口进行映射, 将本地 18080 端口映射到容器内部的 80 端口。
- (3) -d: 设置容器后台运行。
- (4) -it: 交互式登录
- (5) nginx: 镜像名

```
docker run -it centos # 新建并运行容器, 交互式登录, 打开伪终端  
docker run -it --name centos_1 centos # 重命名为 centos_1  
docker run -d centos /bin/sh -c "while true;do echo hello zzyy;sleep 2;done"
```

#### 6.4.3 启停容器

```
docker start a0ea4b82de57 #启动, 容器 ID  
docker restart a0ea4b82de57 #重启  
docker stop cd47805da0f6 #停止  
docker kill cd47805da0f6 #强制停止
```

#### 6.4.4 进入容器 (正在运行)

```
docker attach 容器 ID #进入容器并打开交互终端  
docker exec -it 容器 ID /bin/bash #进入容器并打开交互终端
```

Attach: 直接进入容器启动命令的终端, 不会启动新的进程

Exec: 在容器中打开新的终端, 并可以启动新的进程

#### 6.4.5 执行命令

直接执行容器内命令并返回结果, 但不进入终端, 默认使用bash

```
docker exec 容器 ID 执行命令  
docker exec cd47805da0f6 ls -l
```

#### 6.4.6 退出容器

\$ exit 关闭并退出

Ctrl+Q+P 退出但不关闭

#### 6.4.7 删除容器

\$ docker rm 8f3c5e5f4327 #容器 ID

\$ docker rm -f 8f3c5e5f4327 #强制删除

\$ docker rm -f \$(docker ps -qa) #删除所有容器

\$ docker ps -qa | xargs docker rm #删除所有容器

#### 6.4.8 查看容器日志

docker logs -t -f --tail 10 0721dba03b7e

- (1) -t 是加入时间戳
- (2) -f 跟随最新的日志打印
- (3) --tail 数字 显示最后多少条

#### 6.4.9 查看容器内运行的进程

docker top 0721dba03b7e

#### 6.4.10 查看容器详细信息

docker inspect 0721dba03b7e

#### 6.4.11 容器文件拷贝

把文件从容器拷贝到宿主机中。

docker cp 容器 ID:容器路径 本地路径

docker cp 8a06155fa70b:/usr/local/tomcat/test.txt /home/root

把文件从宿主机拷贝到容器中

```
docker cp 本地路径 容器 ID:容器路径
```

```
docker cp /home/root/test.txt 8a06155fa70b:/usr/local/tomcat
```

#### 6.4.12 容器安装 vim

```
$ apt-get update
```

```
$ apt-get install -y vim
```

#### 6.4.13 容器导出

```
$ docker export cb3056931dda > nginx2.tar
```

#### 6.4.14 容器导入

```
$ cat nginx2.tar | docker import - nginx2:v1.0
```

### 6.5 容器数据卷

数据卷用于容器的持久化、继承和数据共享。数据卷是一个可供一个或多个容器使用的特殊目录，它将主机操作系统目录直接映射进容器。

它可以提供很多有用的特性：

- (1) 数据卷 可以在容器之间共享和重用；
- (2) 对数据卷的修改会立马生效；
- (3) 对数据卷的更新，不会影响镜像；
- (4) 数据卷 默认会一直存在，即使容器被删除。

#### 6.5.1 添加数据卷

```
docker run -it -v /宿主机目录:/容器内目录 镜像名 /bin/bash
```

```
docker run -it -v /root/hostVol:/usr/local/containerVol centos /bin/bash #默认读写
```

#### 6.5.2 数据卷权限

设置只读权限，默认可读写

```
docker run -it -v /root/hostVol:/usr/local/containerVol:ro centos /bin/bash
```



### 6.5.3 查看容器信息

查看容器的数据卷挂载信息如目录、权限等

```
docker inspect 容器 ID
```

### 6.5.4 数据卷容器

命名的容器挂载数据卷，其它容器通过挂载这个(父容器)实现数据共享，挂载数据卷的容器，称之为数据卷容器。容器之间配置信息的传递，数据卷的生命周期一直持续到没有容器使用它为止。

--volumes-from: 容器间数据卷传递共享。

```
docker run -it --name 容器名 --volumes-from 共享容器名 镜像名
```

```
docker run -it --name dc2 --volumes-from dc1 centos
```

## 版本记录

版本编号 / 修改状态	拟制人/修改人	审核人	批准人	备注
V1.0	王增涛	陈旭	张路	

