# Docker的基本知识

本次Docker的安装环境是CentOS7.6，Docker必须在CentOS7版本及以上才能运行。

Docker官网：<https://www.docker.com/>

Docker公共镜像仓库：<https://hub.docker.com/>

## 自动安装Docker

# 使用官方安装脚本

curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun

## 手动安装Docker

# 1.卸载Docker旧版本

sudo yum remove docker \

docker-client \

docker-client-latest \

docker-common \

docker-latest \

docker-latest-logrotate \

docker-logrotate \

docker-engine

# 2.安装gcc

sudo yum -y install gcc-c++

# 3.安装需要的安装包

sudo yum install -y yum-utils \

device-mapper-persistent-data \

lvm2

# 4.设置镜像的仓库，这个是国外的(不推荐，很慢)

sudo yum-config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo>

# 5.设置镜像的仓库，这个是阿里云的(推荐)

sudo yum-config-manager \

--add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

# 6.更新yum软件包索引

sudo yum makecache fast

# 7.安装Docker引擎、容器、组件

sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# 8.启动Docker，如果这一步报错了，执行sudo service docker start手动启动服务

sudo systemctl start docker

# 9.查看Docker的运行状态

sudo systemctl status docker

# 10.简单运行一个镜像来检测Docker引擎的成功安装

sudo docker run hello-world

补充：在使用run命令运行一个镜像时，Docker会首先在本机寻找镜像，找到就运行。否则默认会去Docker Hub上找，找到后会下载到本机并运行，否则就报错-找不到镜像。在设置了新的镜像仓库后，比如阿里云镜像仓库，则将访问Docker Hub改为访问阿里云镜像仓库。

## 卸载Docker

# 1.关闭docker

systemctl stop docker

# 2.卸载Docker引擎、客户端、包

sudo yum remove docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin docker-ce-rootless-extras

# 3.删除镜像、容器、卷

sudo rm -rf /var/lib/docker

sudo rm -rf /var/lib/containerd

# Docker帮助启动类命令

# 1.启动docker

systemctl start docker

# 2.停止docker

systemctl stop docker

# 3.重启docker

systemctl restart docker

# 4.查看docker状态

systemctl status docker

# 5.开启启动

systemctl enable docker

# 6.查看概要信息，包括镜像和容器数

docker info

# 7.查看总体帮助文档

docker --help

# 8.查看具体命令帮助文档

docker 具体命令 --help

# Docker镜像

## 1.说明

docker镜像是一种轻量级、可执行的独立软件包，它包含运行某个软件所需的所有内容，我们把应用程序和配置依赖打包好形成一个可交付的运行环境(包括代码、运行需要的库、环境变量、配置文件等)，这个打包好的运行环境就是image镜像文件。

## 2.镜像命令

# 1.查看本机已有的镜像

docker images [可选项]

可选项：

-a # 列出所有镜像(默认)

-q # 只显示镜像的id

镜像名 # 只显示符合名称的镜像



REPOSITORY：表示镜像的仓库源

TAG：镜像的标签

IMAGE ID：镜像ID

CREATED：镜像创建时间

SIZE：镜像大小

同一仓库源可以有多个 TAG，代表这个仓库源的不同个版本，区分不同的镜像是使用REPOSITORY:TAG，即REPOSITORY+TAG。

# 2.搜索镜像

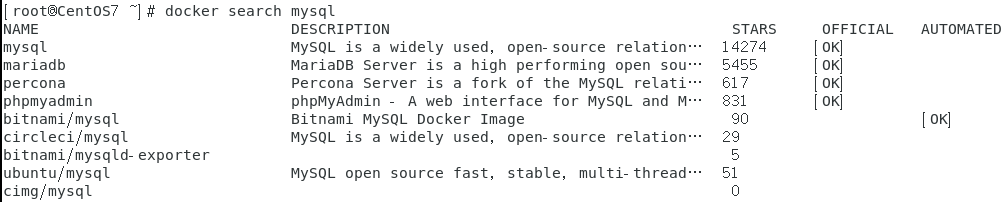
docker search 镜像仓库源名称

可选项：

--filter # 根据提供的条件过滤镜像仓库源。

如：docker search mysql --filter=STARS=3000 # 查看STARS大于等于3000的mysql镜像

docker search redis --limit 5 # 列出前5个镜像(默认是25个)



NAME: 镜像仓库源的名称

DESCRIPTION: 镜像的描述

STARS: 类似 Github 里面的 star，表示点赞、喜欢的意思。

OFFICIAL: 是否 docker 官方发布

AUTOMATED: 自动构建。

# 3.拉取镜像(默认tag为latest)

docker pull镜像仓库源名称

拉取指定tag的镜像的示例：

docker pull mysql:5.7 # 拉取tag为5.7的mysql

# 4.查看镜像、容器、数据卷所占的空间

docker system df

# 5.运行镜像(创建容器实例)

docker run [可选项] 镜像名称

可选项说明：

--name=”容器名称” 给容器起名以区分容器

-d 表示以后台方式运行

-it i表示交互式操作，t表示终端，它俩也可以拆开用

-p(小写) 指定端口

-p 主机端口:容器端口 # 主机端口映射为容器端口

-p 容器端口

容器端口

-P(大写) 表示随机指定端口

以下命令使用 centos 镜像启动一个容器，参数为以命令行模式进入该容器：

docker run -it centos /bin/bash # /bin/bash表示用交互式shell进入容器

# 退出容器，同时关闭容器

exit

# 退出容器，但是不关闭容器

ctrl + p + q

# 后台启动容器

docker run -d 镜像名称 # 这种后台启动方式会让某些容器启动后自动关闭，比如centos、

ubuntu等。因为容器内没有服务，所以容器后台启动的方式没有意义，自动关闭是为

了节省资源。

# 6.根据镜像名称删除镜像

docker rmi 镜像仓库源名称

# 根据镜像名称和tag删除镜像

docker rmi 镜像仓库源名称1:TAG镜像仓库源名称2:TAG ...

根据镜像id删除镜像的示例(镜像id可以写多个，用空格隔开)：

docker rmi -f 91b53e2624b4 2be84dd575ee # 镜像id是91b53e2624b4、2be84dd575ee

-f指的是强制删除，不再询问用户，这个-f在删除命令中是通用的。

删除本机所有镜像的示例：

docker rmi -f $(docker images -aq)

# 7.

## 3.虚悬镜像

说明：虚悬镜像 (Dangling Image) 指的是仓库名 (镜像名) 和标签 TAG 都是<none>的镜像。实际开发中，构建或删除镜像出错时有概率生成虚悬镜像，虚悬镜像会对系统造成一些潜在风险。因此，发现就应该删除。

# 查看本地所有虚悬镜像

docker image ls -f dangling=true

# 删除全部虚悬镜像

docker image prune

## 4.分层镜像

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，它支持对文件系统的修改作为一次提交来一层层的叠加，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继承，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的文件和目录。

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UnionFS。

# Docker容器

## 1.说明

docker容器是由docker镜像创建的运行实例，一个镜像每run一次就会创建一个新的容器。容器之间相互隔离。

## 2.容器命令

# 1.从仓库拉取一个镜像，这里拉取的是centos

docker pull centos

# 2.运行镜像(创建容器实例)

docker run [可选项] 镜像名称

可选项说明：

--name=”容器名称” 给容器起名以区分容器

-d 表示以后台方式运行

-it i表示交互式操作，t表示终端，它俩也可以拆开用

-p(小写) 指定端口

-p 主机端口:容器端口 # 主机端口映射为容器端口

-p 容器端口

容器端口

-P(大写) 表示随机指定端口

以下命令使用 centos 镜像启动一个容器，参数为以命令行模式进入该容器：

docker run -it centos /bin/bash # /bin/bash表示用交互式shell进入容器

# 退出容器，同时关闭容器

exit

# 退出容器，但是不关闭容器

ctrl + p + q

# 后台启动容器

docker run -d 镜像名称 # 这种后台启动方式会让某些容器启动后自动关闭，比如centos、

ubuntu等。因为容器内没有服务，所以容器后台启动的方式没有意义，自动关闭是为

了节省资源。

# 3.查看正在运行的容器

docker ps [可选项]

可选项：

-a # 查看正在运行的容器，包括曾经运行过的

-n=x # 查看最近x个运行的容器

-q # 只显示容器id

-l # 查看最近一次创建的容器

# 4.删除容器

docker rm 容器id/容器名 # 此命令不能删除正在运行的容器

docker rm -f 容器id/容器名 # 强制删除容器，正在运行的也可以

docker rm -f $(docker ps -aq) # 强制删除所有容器

docker ps -a -q|xargs docker rm # 删除所有容器

# 5.启动容器

docker start 容器id/容器名

# 6.重启容器

docker restart 容器id/容器名

# 7.关闭容器，stop会给容器内应用10s时间去停止服务

docker stop 容器id/容器名

# 8.强制关闭容器，跟stop不同，kill会立即停止服务

docker kill 容器id/容器名

# 9.交互式shell进入运行中的容器

docker exec -it 容器名称/容器id /bin/bash (推荐)

docker attach容器名称/容器id

区别：exec是在容器中打开新的终端，并启动新的进程。用exit退出，不会导致容器的停止。

attach直接进入容器启动命令的终端，不会启动新的进程，用exit退出，会导致容器的停止。

# 10.打印容器的日志

docker logs 容器id/容器名

# 11.查看运行中的容器的进程信息

docker top容器id/容器名

# 12.查看容器的底层信息，包括容器的id、创建时间、运行状态、启动参数、目录挂载、网

路配置等等

docker inspect 容器id/容器名

# 13.拷贝容器内文件到主机中

docker cp 容器id:容器内路径 目的主机路径

# 14.导出容器的内容流作为一个tar归档文件(自制镜像)

docker export 容器id > 文件名.tar

# 15.从tar包中导入为镜像

cat 文件名.tar | docker import -镜像用户/镜像名:镜像版本号

# 16.提交容器副本使之成为一个新的镜像，新镜像在本地

docker commit -m=”提交的描述信息” -a=”作者” 容器ID 目标镜像名:标签名

# Docker其他命令

# 1.查看Docker版本信息

docker version

# 3.查看日志

docker logs

# Docker安装常用镜像

## 1.tomcat

(1)查找tomcat镜像

docker search tomcat

(2)拉取镜像(最新的版本现在是tomcat10.1.10)

docker pull tomcat

(3)查看是否有拉取到tomcat

docker images tomcat

(4)创建容器实例(运行镜像)并指定端口号，容器名称为tomcatContainer

docker run -it -p 8080:8080 --name tomcatContainer tomcat

(5)访问tomcat主页

最新的tomcat主页用<http://localhost:8080/>访问会报404。

解决方式是：

命令行进入容器，执行以下命令

rm -r webapps # 删除webapps文件夹

mv webapps.dist webapps # 将webapps.dist重命名为webapps

(6)重新访问<http://localhost:8080/>，此时主页显示正常。

补充：一般不会使用最新版，因为不稳定，所以可以选择8.0版本即可。

docker pull billygoo/tomcat8-jdk8。使用8.0就不会有一开始访问tomcat主页报404的问题。

## 2.mysql(不建议装)

(1)拉取5.7版本的mysql

docker pull mysql:5.7

(2)在运行镜像前检查linux是否已经安装过mysql，避免端口号冲突

ps -ef|grep mysql

(3)运行镜像

docker run -d -p 3306:3306 \

--privileged=true \

-v /zzyyuse/mysql/log:/var/log/mysql \

-v /zzyyuse/mysql/data:/var/lib/mysql \

-v /zzyyuse/mysql/conf:/etc/mysql/conf.d \

-e MYSQL\_ROOT\_PASSWORD=123456 \

--name containerMysql mysql:5.7

命令解释：

-v /zzyyuse/mysql/log:/var/log/mysql \

-v /zzyyuse/mysql/data:/var/lib/mysql \

-v /zzyyuse/mysql/conf:/etc/mysql/conf.d \

这三个命令类似，都是在主机创建容器卷，实际是将容器中的文件同步到主机上，这样即使容器删除了，主机上的文件还在，只要再使用以上的命令创建容器，那么原先容器中的文件就还在，库就还在。但是正在运行的库一旦被删，需要使用binlog恢复一部分数据。

(4)进入配置目录，新建配置文件，插入内容以修改编码集(mysql8.0不用改)

cd /zzyyuse/mysql/conf # 进入配置目录

vim my.cnf # 新建配置文件

插入内容如下：

[client]

default\_character\_set=utf8

[mysqld]

collation\_server = utf8\_general\_ci

character\_set\_server = utf8

然后按ESC键，再输入:wq! 回车即可，表示保存退出。

(5)查看my.cnf中的内容

cat my.cnf

(6)重启容器

docker restart 容器id

(7)使用exec命令进入mysql

docker exec -it 容器id /bin/bash

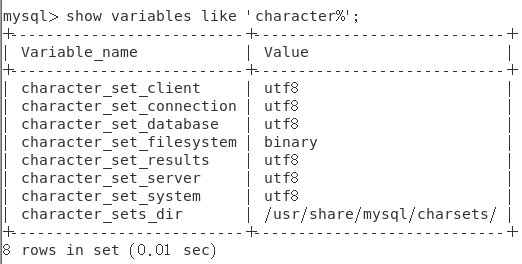
(8)使用mysql的登录命令登录mysql

mysql -uroot -p

123456

(9)输入命令，检查编码集

show variables like 'character%';



(10)使用sql命令即可，也可以用navicat工具连接并操作mysql。

为什么不建议用 docker 部署 mysql？

a.数据持久化：使用Docker部署MySQL需要将数据存储在Docker容器中，这意味着在删除容器时可能会导致数据丢失。为了解决这个问题，需要将数据存储在主机上或使用数据卷，这增加了一些复杂性。

b.性能问题：在Docker容器中运行MySQL会导致一些性能问题。例如，如果容器内存不足，MySQL可能会因为内存不足而崩溃。此外，Docker容器需要额外的虚拟化开销，这可能会导致MySQL在容器中运行的速度较慢。

c.安全问题：使用Docker部署MySQL可能会导致安全问题。例如，容器中的MySQL可能会受到容器逃逸攻击的威胁，因为容器中的其他进程可能会访问MySQL容器的文件系统。

## 3.redis