# Docker概述

## 1、为什么会出现Docker？

1.Docker解决了传统项目开发中环境部署和配置麻烦问题；

2.解决了开发和运维在项目交付过程中的矛盾；

## 2、什么是Docker？

Docker是基于 Go语言实现的开源容器项目，是一种解决了运行环境和配置问题的软件容器,方便做持续集成并有助于整体发布的容器虚拟化技术。

## 3、Docker能做什么？

1.可以通过定制应用镜像来实现持续集成、持续交付、部署；

2.提供了除内核外完整的运行时环境，确保了应用环境一致性；

## 4、Docker和虚拟机的区别

1.虚拟机是一个软件,通过软件(Vmware、Virtual Box等软件)可以虚拟出一台或者多台具有完整硬件功能的、运行在一个完全隔离的环境中的完整操作系统(电脑)，在实体计算机中能够完成的工作在虚拟机中都能够实现,且在虚拟机中的操作不会对真正的系统产生任何影响;虚拟机技术是虚拟化技术的一种；  
2.Docker容器是在操作系统层面实现虚拟化的,模拟的只是一个最核心的环境。Docker与宿主机共享硬件资源及操作系统,容器与容器之间共享内核因此更加轻便。

# 如何使用Docker

**1、Docker核心概念**

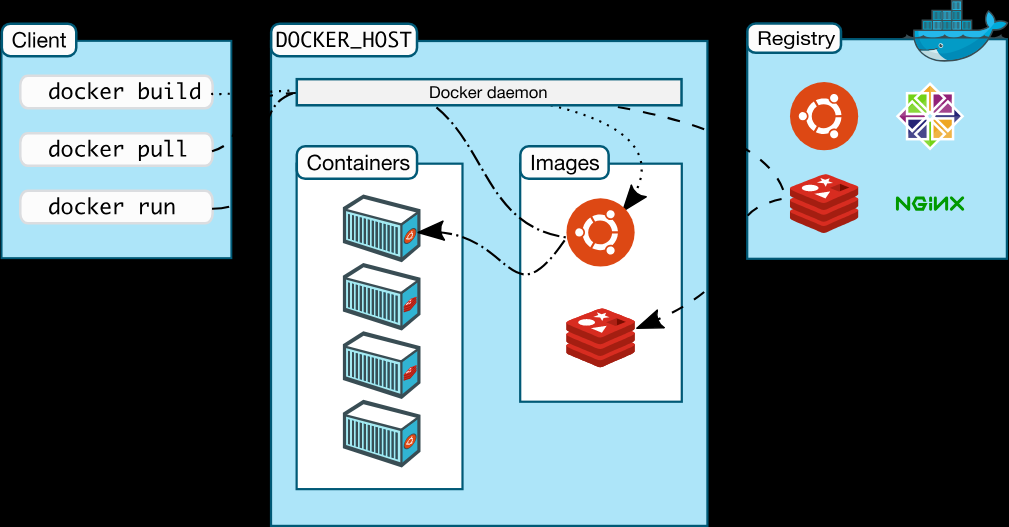
a.Docker主机(host):安装了docker程序的机器(Docker直接安装在操作系统之上)

b.Docker客户端（client）：连接docker主机进行操作;

c.Docker仓库(repository)：用来保存各种打包好的软件镜像;

d.Docker镜像(Images) :软件打包好的镜像,放在docker仓库;

e.Docker容器(container)：镜像启动后的实例称为一个容器,容器是独立运行的一个或一组应用;容器是镜像创建的实例;



**2、Docker的使用步骤**

1.安装Docker;

2.去Docker仓库找到需要安装的软件对应的镜像;

3.使用Docker运行该软件的镜像,这个镜像会产生一个Docker容器,容器就代表这个当前正在运行的应用;

4.对容器的启动停止就是对软件的启动停止;

**3、如何学习Docker**

**知识储备**

1.熟悉Linux命令及相关背景知识;

2.熟悉Maven/git方面的知识

**相关文档**

官网地址：[https://www.docker.com/](https://www.docker.com/" \o "https://www.docker.com/" \t "_blank)

官方文档地址：[https://docs.docker.com/](https://docs.docker.com/" \o "https://docs.docker.com/" \t "_blank) （官网文档入口位于官网底部）

**相关网站**

B站：[https://www.bilibili.com/](https://www.bilibili.com/" \o "https://www.bilibili.com/ " \t "_blank)

网易云课堂：[https://study.163.com/](https://study.163.com/" \o "https://study.163.com/" \t "_blank)

腾讯课堂：[https://ke.qq.com/](https://ke.qq.com/" \o "https://ke.qq.com/ " \t "_blank)

慕课网：[https://www.imooc.com](https://ke.qq.com/" \o "https://www.imooc.com " \t "_blank)

**4、Docker是如何工作的**

Docker是一个Client-Server结构系统，Docker守护进程运行在主机上，然后通过Socket连接从客户端访问，守护进程从客户端接收命令并管理运行在主机上的容器



# 在CentOS上安装docker-ce？

## 1.在Linux上安装docker-ce

1.查看Linux系统的版本

必须是CentOS7及以上版本

cat /etc/os-release

2.查看Linux系统的内核版本

必须是3.10及以上

uname -r

3.升级软件包及内核;(如果Linux系统的内核版本低于3.10)

yum update

4.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

5.设置存储库 设置 Docker 存储库之后，可以从存储库安装和更新 Docker; a.安装yum-utils包（提供yum-config-manager 实用程序）并设置稳定存储库

sudo yum install -y yum-utils

b.设置存储库

sudo yum-config-manager \

--add-repo \

https://download.docker.com/linux/centos/docker-ce.repo

由于官方存储库较慢,建议使用国内的存储库:

sudo yum-config-manager \

--add-repo \

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

3.安装 docker-ce

sudo yum install docker-ce docker-ce-cli containerd.io

3.1安装后可以查看docker版本号

docker -v

4.启动docker

sudo systemctl start docker

5.将docker服务设为开机启动

sudo systemctl enable docker

备注：

Sudo ：切换到root，但不需要root的密码;

\：连接符

官方文档：<https://docs.docker.com/engine/install/centos/>

在其他平台上的安装，可以参考Docker的官网，Docker官网提供了各个平台安装Docker的相关文档;官网教程：[https://docs.docker.com/get-docker/](https://docs.docker.com/get-docker/" \o "https://docs.docker.com/get-docker/" \t "_blank)

**1.配置DockerHub加速器**

1.国内从 Docker Hub 拉取镜像速度比较慢，可以配置镜像加速器。Docker 官方和国内很多云服务商都提供了国内加速器服务:

Docker 中国官方镜像 https://registry.docker-cn.com

DaoCloud 镜像站 http://f1361db2.m.daocloud.io

Azure 中国镜像 https://dockerhub.azk8s.cn

科大镜像站 https://docker.mirrors.ustc.edu.cn

七牛云 https://reg-mirror.qiniu.com

网易云 https://hub-mirror.c.163.com

腾讯云 https://mirror.ccs.tencentyun.com

2.Linux实例配置（腾讯云为例）

1.创建或修改 /etc/docker/daemon.json 文件，并写入以下内容：

vim /etc/docker/daemon.json

{

"registry-mirrors": [

"https://mirror.ccs.tencentyun.com"

]

}

2.依次执行以下命令，重新启动 Docker 服务。

sudo systemctl daemon-reload

sudo systemctl restart docker

Ubuntu16.04 请执行 sudo systemctl restart dockerd 命令

3.检查加速器是否生效 执行 docker info 命令，返回结果中包含以下内容，则说明配置成功。

Registry Mirrors: <https://mirror.ccs.tencentyun.com>

支持配置多个镜像加速器服务

{

"registry-mirrors": [

"https://mirror.ccs.tencentyun.com",

"https://registry.docker-cn.com"

]

}

# 如何操作Docker镜像

## 基本命令

### 1.帮助命令

语法： docker version

说明： 查看docker版本信息

语法：docker info

说明：查看docker的详细信息

语法： docker --help

说明：帮助命令，当我们不了解一个命令的时候可以使用—help命令查看相关操作;

### 2.检索

基本语法：

语法：docker search [OPTIONS] TERM

说明：我们经常去docker hub上检索镜像的详细信息,如镜像的Tag;

默认去 docker hub 仓库搜索镜像; https://hub.docker.com/

常用选项

常用项[OPTIONS]:

-f，--filter filter：过滤

--no-trunc:显示完整的镜像信息

常用实例

实例:docker search redis

说明：搜索指定镜像信息

实例：docker search -f stars=200 mysql

说明:搜索镜像的STARS大于等于200的镜像;

实例：docker search -f stars=200 --no-trunc mysql

说明:搜索镜像的STARS大于等于200的镜像并且显示完整的镜像描述;

### 3.拉取

基本语法

语法：docker pull [OPTIONS] NAME[:TAG|@DIGEST]

说明：

NAME:镜像的名字

:tag是可选的,tag表示标签，多为软件的版本,默认为latest

如果指定tag必须在docker hub中存在

常用实例：

实例：docker pull mysql:8.0

说明：拉取指定镜像

### 4.列表

基本语法

语法：docker images [OPTIONS] [REPOSITORY[:TAG]]

说明：查看本地镜像列表

常用选项：

可选项[options]：

-a, --all ：查看所有本地镜像包括中间镜像;

-q, --quiet :只显示镜像的ID

--digests: 显示镜像的摘要信息

--no-trunc:显示完整的镜像信息

结果说明:

REPOSITORY: 镜像的仓库源

TAG: 镜像的标签

IMAGE ID:镜像的编号

CREATED:镜像的创建时间

SIZE：镜像的大小

常用实例：

实例：docker images

说明：查看所有的本地镜像

实例：docker images redis

说明：查看本地镜像中REPOSITORY为redis的镜像

### 5.删除

基本语法

语法：docker rmi [OPTIONS] IMAGE [IMAGE...]

说明：删除指定的本地镜像，多个用空格隔开

IAMGE:镜像的ID或镜像名称(REPOSITORY)

常用项

常用项[options]:

-f:强制删除

常用实例

实例 docker rmi snowdreamtech/frps(镜像名称)

docker rmi 24207ccc9cce(镜像ID)

说明：删除指定镜像

实例：docker rmi nginx:1.21 tomcat

说明：删除多个镜像(nginx:1.21镜像和tomcat镜像),不指定tag默认删除latest

实例：docker rmi -f $(docker images -aq)

说明：删除本地所有的镜像

## 构建镜像命令

### 1.build

基本语法：

语法：docker build [OPTIONS] PATH | URL | -docker build [OPTIONS] PATH | URL | -

说明：使用Dockerfile文件构建镜像

常用项：

-f: dockerfile的文件的位置，默认为'PATH/Dockerfile'

-t: 镜像名称和可选的tag 格式为: name[:tag]

常用实例

实例：docker build -f /root/mydocker/dockerfile -t ben/centos .

说明：docker build -t 名称:版本

.表示当前目录,会在当前目录上找dockerfile文件

### 2.commit

基本语法：

语法：docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]

说明: 将容器的文件更改或设置提交到新映像（用于定制自己的镜像）

CONTAINER ：容器的ID或容器的名称

常用项

-a ： 作者信息

-m: 描述信息

# 如何操作Docker容器？

## 基本命令

### 1运行

基本语法

语法:docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

说明：运行容器;(软件镜像–运行镜像—产生一个容器（正在运行的软件）)

常用选项

常用选项[OPTIONS]:

-d:后台运行容器即创建一个守护式容器在后台一直运行

Docker容器后台运行，就必须有一个前台进程，否则就会自动退出 解决方案就是将要运行的程序以前台进程的形式运行;

-i:表示以交互模式运行容器, 通常与-t同时使用;

-t:表示容器启动后会进入命令行,加上这个参数后，容器创建就能登录进去，即分配了一个伪终端.

-p:端口映射,将主机端口映射到容器内部的端口(指定容器端口)

语法：-p 主机端口：容器内部端口 (-p 6379:6379 )

-p ip:主机端口：容器内部端口

-p ip:容器端口

-p 容器端口

-P:随机端口

-v:给容器挂载存储卷，挂载到容器的某个目录

语法： -v 宿主机目录：容器目录 |

-v 容器目录

说明：将宿主机的目录挂载到容器中(将主机目录与容器目录之间进行共享);

如果不指定宿主机目录，会在宿主机的/var/lib/docker/volumes 目录下创建默认的数据卷;

--name:指定容器名字,后续可以通过名字进行容器管理

--restart=always:重启docker时，自动启动相关容器

--privileged=true 容器内部对挂载的目录拥有读写等特权

常用实例：

MYSQ：

docker run -d -p 3306:3306 \

--name ben-backup-mysql \

--restart=always \

-e TZ=Asia/Shanghai \

-e MYSQL\_ROOT\_PASSWORD=mysql@958958 \

-v /opt/data/mysql\_backup/mysqldata:/var/lib/mysql \

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

mysql:8.0 \

--lower-case-table-names=1

### 1.1RUN的工作流程

### 2.列表

基本语法

语法：docker ps [OPTIONS]

说明：查看本地所有容器，默认查询所有运行中的容器

常用选项

常用项：

-a 查看所有容器(包括历史运行过的容器)

-l 显示最近创建的容器

-n 显示最近创建的N个容器

-q 静态模式，只显示容器编号

--no-trunc 不截断输出

常用实例：

实例：docker ps

说明：默认查看运行中的容器)

实例：docker ps -a

说明：加上-a 可以查看所有容器（包括历史运行过的容器）

实例：docker ps -n 2

docker ps -n=2

说明：查看最近创建的2个容器

### 3.启动

语法： docker start [OPTIONS] CONTAINER [CONTAINER...]

说明：启动一个或多个容器

CONTAINER ：容器的ID或容器的名称

### 4停止

语法：docker stop [OPTIONS] CONTAINER [CONTAINER...]

说明：停止一个或多个在运行中的容器

CONTAINER ：容器的ID或容器的名称

### 4.1重启

语法：docker restart [OPTIONS] CONTAINER [CONTAINER...]

说明：重启一个或多个容器

CONTAINER ：容器的ID或容器的名称

### 5删除

命令：docker rm [OPTIONS] CONTAINER [CONTAINER...]

说明：删除指定的未启动的一个或多个容器

多个容器之间用空格隔开

常用实例

实例：docker rm -f $(docker -aq)

docker ps -aq |xargs docker rm

说明：删除所有的容器

### 6.退出容器：

exit:容器停止退出

ctrl+p+q：容器不停止退出

### 7端口映射

语法：-p 主机端口：容器内部端口 (-p 6379:6379 )

docker run -d -p 6379:6379 -name myredis docker.io/redis

说明：-p:主机端口(映射到)容器内部的端口

### 8.容器日志

基本命令：

命令：docker logs [option] container

说明：查看容器日志

container：容器名称|容器ID

常用项[options]:

-f:动态查询日期

-t:加入时间戳

--tail ：显示最后多少条

实例：

实例：docker logs -f container-name/container-id

说明： -f动态查询日志

## 其他命令

### 1.top

语法：docker top CONTAINER [ps OPTIONS]

说明： 显示容器的运行的进程

CONTAINER :容器id或容器名称

### 2.inspect

基本语法：

语法：docker inspect [OPTIONS] NAME|ID [NAME|ID...]

说明：查看Docker对象的相关细节信息

常用实例

实例:docker inspect 容器ID或容器名称

说明：查看指定容器的细节信息;

### 3.exec

基本语法

语法： docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

说明：进入容器, 在容器中打开新的终端，并且可以启动新的进程(进入正在运行中的容器并以命令行交互)

CONTAINER ：容器名称或容器ID

实例

常用实例： docker exec -it container-id baseShell

docker exec -it容器id或容器名称 ls -l /tmp 进入容器做指定操作，不需要进入容器

docker exec -it 容器id或容器名称 /bin/bash 进入容器做指定操作

### 3.1.attach

基本语法：

语法： docker attach [OPTIONS] CONTAINER

说明：进入容器,直接进入容器启动命令的终端，不会启动新的进程

CONTAINER:容器的ID或容器名称

### 4.cp

语法：

语法： docker cp [OPTIONS] CONTAINER:SRC\_PATH DEST\_PATH|-

docker cp [OPTIONS] SRC\_PATH|- CONTAINER:DEST\_PATH

说明：在容器和本地文件系统之间复制文件/文件夹(容器与主机之间数据拷贝)

CONTAINER：容器ID或容器名称；

SRC\_PATH：源目录或文件

DEST\_PATH：目标目录或文件

使用“-”作为源从stdin读取tar归档文件并将其解压缩到容器中的目录目标；

使用“-”作为目标，以流的式传输容器源到stdout。

实例

实例:docker cp [options] 容器名称或容器ID:容器目录或文件 宿主机目录或文件

说明：将容器中的文件或目录复制到宿主机目录或文件

实例：docker cp [options] 宿主机目录或文件 容器名称或容器ID:容器目录或文件

说明：将宿主机目录或文件复制到容器目录或文件

# Docker如何管理数据

## 1、Docker容器管理数据的两种方式

1.数据卷(Data Volumes) ：容器内数据直接映射到本地主机环境；

2.数据卷容器（Data Volume Containers ）:使用特定容器维护数据卷;

## 2、什么是数据卷？

数据卷( Data Volumes) 个可供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，类似于 Linux 中的 mout 行为。

## 3、为什么要使用数据卷？

Docker容器产生的数据，当容器删除后，数据就没有了，为了将数据保存下来，更好的管理数据，Docker提供了卷来管理数据。

## 4、数据卷有什么特性？

1.数据卷可以在容器之间共享和重用，容器间传递数据将变得高效与方便；

2.对数据卷内数据的修改会立马生效，无论是容器内操作还是本地操作；

3.对数据卷的更新不会影响镜像，解耦开应用和数据；

4.卷会一直存在 ，直到没有容器使用，可以安全地卸载它;

## 5、如何创建卷？

### 1.运行容器时通过-v来创建数据卷

语法

语法：docker run -v 宿主机绝对路径目录:容器内目录 镜像名

docker run -v 宿主机绝对路径目录:容器内目录:or 镜像名

说明：:or表示容器目录只读

实例

实例：docker run -d -p 3309:3306 \

--name ben-dev-mysql \

--restart=always \

-e TZ=Asia/Shanghai \

-e MYSQL\_ROOT\_PASSWORD=mysql@958958 \

-v /opt/data/mysql\_dev/mysqldata:/var/lib/mysql \

-v /etc/mysql/mysq\_dev/conf.d:/etc/mysql/conf.d \

-v /etc/mysql/mysq\_dev/my.cnf:/etc/mysql/my.cnf \

mysql:8.0 \

--lower-case-table-names=1

### 2.使用volume命令

1.创建卷create

语法：docker volume create [options] [volume]

说明：创建卷

常用项:

-d 指定驱动名称,默认‘local’

实例：docker volume create -d test

2.数据卷列表ls

语法：docker volume ls [OPTIONS]

说明：列出已有的数据卷

3.卷详细信息inspect

语法：docker volume inspect [options] volume [volume...]

说明：显示一个或多个卷的详细信息

4.删除为使用的本地卷prune

语法：docker volume prune [options]

说明：删除所有未使用的本地卷

5.删除卷rm

语法：docker volume rm [options] volume [volume...]

说明：删除一个或多个卷。无法删除容器正在使用的卷。

常用项：

-f, --force 强制删除一个或多个卷

### 3.使用dockerfile文件

在dockerfile中使用VOLUME指令来给镜像添加一个或多个数据卷

VOLUME [“/mydata0”,”/mydata1”,”/mydata2”]

出于可移植性和分享的考虑，用-v这种方式不能直接在dockerfile中实现。 由于宿主机的目录是依赖特定的宿主机的，并不能保证所有的宿主机都存在这样的特定目录。所有volume指令只支持容器内的；

## 6、如何绑定数据卷？

除了使用Volume子命令来管理数据卷外，还可以在创建容器的时候将主机本地的任意路径挂载到容器内作为数据卷，这种形式创建的数据卷被称为绑定数据卷。 语法：

在使用docker run [options] image [command] [arg...]命令的时候，可以使用-mount选项来使用数据卷；

--mount 选项支持三种类型的数据卷：

Volume:普通数据卷，映射到主机 /var/lib/docker/volumes路径下；

Bind：绑定数据卷，映射到主机指定路径下;

Tmpfs:零时数据卷,只存在内存中。

实例：

实例：docker run -d -p 9093:9093 \

--name itour-online-web \

–mount type=bind,source=/webapp,destination=/opt/webapp \

ben/itour-online-manager-web

等同于

docker run -d -p 9093:9093 \

--name itour-online-web \

–v webapp:opt/webapp \

ben/itour-online-manager-web

其中本地目录必须为绝对路径，容器内的路径可以为相对路径，如果路径不存在，docker会默认创建。

## 7、什么是数据卷容器？

命名的容器挂载数据卷，其他容器通过挂载这个容器实现数据共享，挂载数据卷的容器，称为数据卷容器。

## 8、为什么使用数据卷容器？

如果需要多个容器之间共享一些持续性更新的数据，最简单的方式是使用数据卷容器。数据卷容器也是一个容器，但是它的目录是专门提供数据卷给其他容器挂载。

## 9、如何实现数据卷容器之间数据共享？

1.创建数据卷容器

2.在其他容器使用 --volumes-from List (数据卷容器)来挂载数据卷容器中的数据卷。

## 10、总结

1.可以多次使用 --volumes-from List (数据卷容器)来从多给容器挂载多个数据卷，也可以从已经挂载了容器卷的容器来挂载数据卷；

2.使用--volumes-from List (数据卷容器)所挂载的容器自身不需要保持运行状态;

3.如果删除了挂载的容器，数据卷并不会自动删除，如果要删除一个数据卷，必须在删除最后一个还挂载着他的容器时使用docker rm -v 命令来指定同时删除关联的容器。

4.使用数据卷容器可以让用户在容器之间自由地升级和移动数据卷。

5.容器之间的配置信息的传递，数据卷的生命周期一直持续到没有容器使用它为止。

# 如何使用Dockerfile构建镜像

## 1、什么是Dockerfile？

Dockerfile 是一个文本格式的配置文件， 使用 Dockerfile 可以快速构建镜像。Dockerfile是镜像的描述文件。

## 2、Dockerfile文件的基本结构？

Dockerfile 由一行行命令语句组成， 并且支持以＃开头的注释行。 一般而言， Dockerfile 主体内容分为四部分：基础镜像信息、 维护者信息、 镜像操作指 令和容器启动时执行指令。

## 3、Dockerfile中文件的规则

1.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。

2.指令按照从上到下，顺序 执行。

3.#表示注释。

4.每条指令都会创建一个新的镜像层，并对镜像进行提交。

## 4、Docker 执行dockerfile的大致流程：

1.docker从基础镜像运行一个容器

2.执行一条指令并对容器做出修改

3.执行类似docker commit的操作提交一个新的镜像层

4.Docker再基于刚提交的镜像运行一个新的容器

5.执行dokerfile中的下一条指令直到所有指令都执行完成。

## 5、Dockerfile文件中的指令

| **指令** | **说明** |
| --- | --- |
| FROM | 基础镜像，当前镜像是基于那个镜像 |
| MAINTAINER | 镜像维护者姓名和邮箱地址 |
| RUN | 容器构建时需要运行的命令 |
| EXPOSE | 当前容器对外暴露的端口 |
| WORKDIR | 指定在创建容器后，终端默认登录进来的工作目录，一个落脚点。 |
| ENV | 用来在构建镜像过程中设置环境变量，这个环境变量可以在后续的指令中使用。 |
| ADD | 将宿主机目录下的文件拷备到镜像且ADD命令会自动处理URL和解压tar压缩包。 |
| COPY | 类似ADD,拷备文件和目录到镜像中 |
| VOLUME | 容器数据卷，用于数据保存和持久化操作。 |
| CMD | 指定一个容器启动时要运行的命令 。 Dockerfile中可以有多个CMD指令，但是只有最后一个生效，CMD会被docker run之后的参数替换 |
| ENTRYPOINT | 指定一个容器启动是要运行的命令。 |
| ONBUILD | 当构建一个被继承的dockerfile运行命令时，父镜像在被子镜像继承后父镜像的onbuild被触发。 |

## 6、构建镜像

### 镜像构建三步骤：

1.编写dokerfile文件

2.Docker build

3.Docker run

# Docker方式安装mysql

[Docker](http://localhost:9093/itour/work/detail?id=15)

阅读  最后发布于 2022-05-06 16:10

## 1.使用Docker方式安装MYSQL

docker run -d -p 3306:3306 \

--name ben-backup-mysql \

--restart=always \

-e TZ=Asia/Shanghai \

-e MYSQL\_ROOT\_PASSWORD=mysql@12345 \

-v /opt/data/mysql\_backup/mysqldata:/var/lib/mysql \

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

-v /etc/mysql/mysql\_backup/my.cnf:/etc/mysql/my.cnf \

mysql:8.0 \

--lower-case-table-names=1

## 2.使用Docker-compose方式管理

version: '2'

services:

mysql-backup:

image: mysql:8.0

restart: always

container\_name: ben-backup-mysql

environment:

MYSQL\_ROOT\_PASSWORD: mysql@12345

TZ: Asia/Shanghai

volumes:

- /opt/data/mysql\_backup/mysqldata:/var/lib/mysql

- /etc/mysql/mysql\_backup/conf.d:/etc/mysql/conf.d

- /etc/mysql/mysql\_backup/my.cnf:/etc/mysql/my.cnf

ports:

- 3306:3306

command:

--lower-case-table-names=1

--default-authentication-plugin=mysql\_native\_password

networks:

- ben\_net

networks:

ben\_net:

## 3.注意：

1.如果挂载存储卷时是具体的配置文件，配置文件必须存在，否则报错；

# Docker方式安装Redis

## 1.使用Docker方式安装Redis

docker run -d -p 6379:6379 \

--restart=always \

--name ben-backup-redis \

-v /opt/data/redis\_backup:/data \

-v /etc/redis/redis-backup/conf/redis.conf:/usr/local/etc/redis/redis.conf \

redis:6.2 \

--requirepass 'redis@12345'\

--appendonly yes

## 2.使用Docker-compose方式管理

version: '2'

services:

redis-backup:

image: redis:6.2

restart: always

container\_name: ben-backup-redis

volumes:

- /opt/data/redis\_backup:/data

- /etc/redis/redis\_backup/conf/redis.conf:/usr/local/etc/redis/redis.conf

ports:

- 6379:6379

command:

--requirepass 'redis@12345'

--appendonly yes

networks:

- ben\_net

networks:

ben\_net:

## 3.注意

如果挂载存储卷时是具体的配置文件，配置文件必须存在，否则报错；

# Eclipse或Idea基础Docker实现一键部署项目到服务器

## 基本步骤

1.使用Docker插件推送项目到服务器；  
2.运行容器运行镜像  
3.浏览器访问

## 一、基础环境搭建

服务器器端：（基础环境搭建）  
1.安装Docker  
2.基于Docker安装MYSQL  
3.基于Docker安装Redis  
本地：  
1.安装开发工具Eclipse或Idea  
2.安装Docker插件  
3.安装Linux远程工具（xshell）  
4.准备项目（springboot项目）

## 二、安装Docker插件

### Eclipse

Help-->Eclipse Marketplace-->搜索 Eclipse Docker Tooling

eclipse配置docker  
window-->show view-->other-->选择docker explorer

### Idea

打开idea,从File-->Settings-->Plugins-->Install jetBrains plugin 进入插件安装界面;

b.在搜索框中输入docker，可以看到Docker integration,点击右边的install按钮进行安装，安装后重启idea;

Idea配置docker  
a.配置docker，连接到远程的docker容器  
从File-->Setttings-->Builde,Excution,Deployment->Docker打开配置界面;

## 三、配置Docker插件连接到Docker

1.Docker引擎默认不允许外部访问需要开启远程访问;

#修改Docker服务文件

vim /usr/lib/systemd/system/docker.service

#修改ExecStart这行,

#在后面追加-H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock

ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock

#重新加载配置文件

systemctl daemon-reload

#重启服务

systemctl restart docker.service

#查看端口是否开启

netstat -nlpt #如果找不到netstat命令，可以进行安装 yum install net-tools

#检查curl是否生效，注意，必须关闭防火墙或腾讯云安全组开放2375端口，这个端口也可以自定义

curl http://服务器地址：2375/info

注意ip地址必须和服务器地址统一;

相关文档：https://blog.csdn.net/boling\_cavalry/article/details/100049996

## 四、配置Docker插件

### 两种方式:

1.直接在pom.xml中配置;

2.在pom中指定Dockfile文件的位置;

### 在pom.xml中直接配置

<!-- docker 插件 可以帮助我们在Maven工程中通过简单的配置，

自动生成镜像并推送到仓库中-->

<plugin>

<groupId>com.spotify</groupId>

<artifactId>docker-maven-plugin</artifactId>

<version>1.0.0</version>

<configuration>

<!-- 镜像名称 -->

<imageName>${docker.image.prefix}/${project.artifactId}</imageName>

<!-- 指定标签 -->

<imageTags>

<imageTag>latest</imageTag>

</imageTags>

<!-- 基础镜像java 1.8 -->

<baseImage>java</baseImage>

<!-- 制作者提供本人信息 -->

<maintainer>wangtao wwangtaoc11@163.com</maintainer>

<!-- 切换到/root目录 -->

<workdir>/ROOT</workdir>

<cmd>["java","-version"]</cmd>

<entryPoint>["java","-jar","${project.build.finalName}.jar"]</entryPoint>

<!-- 指定远程docker api 地址 -->

<dockerHost>http://106.54.162.159:2375</dockerHost>

<!-- 指定Dockerfile文件的位置

<dockerDirectory>src/main/docker</dockerDirectory> -->

<resources>

<resource>

<targetPath>/ROOT</targetPath>

<!-- 指定需要复制的根目录:${project.build.directory}

表示target目录 -->

<directory>${project.build.directory}</directory>

<!-- 指定需要复制的文件

${project.build.finalName}.jar指的是打包后的文件-->

<include>${project.build.finalName}.jar</include>

</resource>

</resources>

</configuration>

<!-- 以上还需要通过mvn clean package docker:build

手动构建镜像，在<plugin>添加以下插件即可以

在项目打包Package的时候即可自动构建镜像,

只需要run就可以访问了;

当执行mvn package时执行mvn clean package docker:build

-->

<executions>

<execution>

<id>build-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

</plugin>

## 五.自动构建镜像

1.执行 mvn package [-P test]  
当执行mvn package时执行mvn clean package docker:build帮助构建镜像；  
2.运行镜像，使用浏览器查看是否部署成功；

## 一、为什么使用Docker-Compose

当容器实例太多时，使用docker-Compose管理容器;类似于spring中appplicationContext.xml对bean对象统一集中管理；

不使用compose效果如何，有哪些问题？

a.先后顺序要求固定，想mysql+redis才能启动微服务访问成功;

b.需要单独运行容器(多次run)

c.容器键的启停或宕机，有可能导致ip地址对应的容器实例变化，映射出错，要么生产环境IP写死(不推荐)，要么通过服务调用

## 二、什么是Docker-Compose

Docker-Compose是Docker官方的开源项目，负责实现对Docker容器集群的快速编排； 可以管理多个Docker容器组成一个应用，我们只需要定义一个Yaml格式的配置文件docker-compose.yml，写好读个容器之间的调用关系，然后，只要一个命令，就能同时启动或停止这些容器。

## 三、Docker-Compose能干什么

1.Docker建议每一个容器只运行一个服务，因为docker容器本身占用资源极少，所以最好是将每个服务器单独的分割开来但是这要面临一个问题，如果我们需要同时部署多个服务，每个服务单独编写dockerfile然后构建镜像，构建容器，工作量大;所以docker的官方提供了docker-compose多服务部署工具；

2.Compose允许用户通过一个单独的docker-compose.yml模板文件（yml格式）来定义一组相关联的应用容器为一个项目；可以很容易地用一个配置文件定义一个多容器的应用，然后使用一条指令安装这个应用的所有依赖，完成构建。Docker-comose解决了容器与容器之间如何管理的问题。

## 四、安装Docker-Compose

官网地址：<https://docs.docker.com/compose/compose-file/compose-file-v3/> 官网下载：<https://docs.docker.com/compose/install/>

安装Compose步骤：  
要求：  
Docker-compose和docker 版本需要对应

1.下载 Docker Compose 的当前稳定版本,将下载好的文件放入/usr/local/bin

https://github.com/docker/compose/releases

2.对二进制文件应用可执行权限：

$ sudo chmod +x /usr/local/bin/docker-compose

3.查看版本信息

docker-compose --version

# Docker Compose使用

## 1.核心概念

一个文件：docker-compose.yml  
两个要素：服务和工程  
服务：一个个应用容器实例，如订单服务、mysq容器；  
工程：由一组关联的应用容器组成的一个完整的业务单元,在docker-compose.yml文件中定义；

## 2.如何使用

Docker-compose使用步骤：  
1.编写Dockerfile定义各个微服务应用并构建出对应的镜像文件;  
2.使用docker-compose.yml定义一个完整业务单元，安排好整体应用中的各个容器服务。最后，执行docker-compose up命令来启动并运行整个应用程序，完成一键部署上线；

## 3.常用命令

docker-compose -h #查看帮助

docker-compose up #启动所有docker-compose服务

docker-compose up -d #启动所有docker-compose服务并在后台运行

docker-compose down #停止并删除容器、网络、卷、镜像；

docker-compose exec yml里面的服务id #进入容器实例内部

docker-compose exec docker-compose.yml文件中写的服务id /bin/bash

docker-compose ps #展示当前docker-compose编排过的运行的所有容器

docker-compose top #展示当前docker-compose编排过的容器进程

docker-compose logs yml里面的服务id #查看容器输出日志

docker-compose config #检查配置

docker-compose config -q #检查配置，有问题才有输出

docker-compose restart #重启服务

docker-compose start #启动服务

docker-compose stop #停止服务