docker

启动 systemctl start docker

守护进程重启 sudo systemctl daemon-reload 重启docker服务 systemctl restart docker 重启docker服务 sudo service docker restart

关闭docker service docker stop 关闭docker systemctl stop docker

一. 安装docker

Docker要求运行在Centos 7上,要求系统为64位,系统内核版本3.10以上

- 1. uname -an 查看当前系统版本
- 2. yum -y install docker 下载安装docker
- 4. docker version 检查docker是否安装成功

```
[root@izwz9ccnzetdxgd4aeufzsz ~]# docker version
Client:
 Version:
                         1.13.1
 API version:
                         1.26
 Package version: docker-1.13.1-96.gitb2f74b2.el7.centos.x86_64
Go version: gol.10.3
                         go1.10.3
b2f74b2/1.13.1
 Git commit:
                          Wed May 1 14:55:20 2019
 OS/Arch:
                          linux/amd64
Gerver:
Version: 1.13.1

API version: 1.26 (minimum version 1.12)

Package version: docker-1.13.1-96.gitb2f74b2.el7.centos.x86_64

Go version: gol.10.3

Git commit: b2f74b2/1.13.1
                         Wed May 1 14:55:20 2019
linux/amd64
 Built:
 OS/Arch:
 Experimental:
                          false
```

二. 镜像操作

1. 创建容器要以镜像为基础,所以先讲下docker镜像的操作docker images 11/1s 查看下本机的是否已经有了镜像docker search java ,还能指定具体的版本进行下载

```
| TootélizZzehzeir87z18q99krk1z local]# docker search java | DESCRIPTION | STARS | OFFICIAL AUTOMA | Ocker.io/node | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/node | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/node | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/node | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/node | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/node | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/node | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] | [OK] | docker.io/openjdk | Mode.js is a JavaScript-based platform for... | 6456 | [OK] |
```

下载镜像

docker pull docker.io/nginx 进行下载

列表包含了仓库名,版本标签,镜像ID,创建时间以及所占用的空间

```
[root@iz2zehzeir87zi8q99krklz local]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

docker.io/java latest d23bdf5b1b1b 21 months ago 643 MB
```

删除没用的镜像docker rmi镜像id(image id)

三、镜像的创建与管理

前边我们已经下载好了Nginx的镜像,接下来我们就创建一个只有Nginx应用的容器docker run -i -t < IMAGE_ID> /bin/bash: -i: 标准输入给容器 -t: 分配一个虚拟终端 /bin/bash: 执行bash脚本,docker run -idt --name container_nginx -p 8080:80 docker.io/nginx 启动一个使用镜像docker.io/nginx,名字container_nginx的容器,-p 8080:80表示将容器的80端口映射到主机的8080端口,这样我们只要访问主机的8080端口就可以访问到容器的服务了。

注意: name前面是两个-, 端口前面有-p, docker.io/nginx是镜像名,8080是主机的端口,80是 Nginx应用的端口

主机上的一个端口只能映射一个容器端口,不可以多个容器端口对应一个主机端口(如果容器安装 的centos类的系统,那么容器端口随便设定,但如果容器内只是单纯的应用,那么容器端口要是应 用自身的端口)

exit 退出容器

docker ps 查看运行中的容器

docker ps -a 查看运行中和非运行中的所有容器

docker exec -it container nginx /bin/bash 进入容器

如果容器还未启动 执行docker start container nginx

/bin/bash: 这是表示载入容器后运行bash, docker中必须要保持一个进程的运行, 要不然整个容器就会退出。 这个就表示启动容器后启动bash。

删除容器

- 1. 容器删除之前先将容器停止
- 2. docker stop container_nginx 或者是容器的id
- 3. docker rm -f container_nginx 容器删除

docker start 与 docker run 的区别

- 1. docker start name 启动一个已经创建的容器
- 2. docker run 创建并启动一个容器

主机和容器文件相互复制

docker cp container_nginx:/usr/local/xin.txt /usr/local/software/ 容器向主机复制文件 docker cp /usr/local/xinzhifu.txt container nginx:/usr/local/ 主机向容器复制文件 请注意,以上这两个命令都是在主机中执行的,不能再容器中执行

镜像可以生成容器,容器可以做成镜像

docker commit container nginx image nginx:v1

容器名 自己起一个镜像的名字:版本号

```
root@izwz9ccnzetdxgd4aeufzsz ~]# docker
                                            IMAGĚ ID
                                                                   CREATED
                                                                                         SIZE
REPOSITORY
                      TAG
                                            5377fd8533c3
719cd2e3ed04
                                                                                         506 MB
109 MB
docker.io/tomcat
                      latest
                                                                   5 days ago
                                                                  9 daýs ago
pinx image_nginx:vl
docker.io/nginx
                      latest
[root@izwz9ccnzetdxgd4aeufzsz ~]# docker commit container nginx
sha256:5ab13cb87e2a80895f77c887f0a89becc099dd0bb63867953aa8375731899eb7
[root@izwz9ccnzetdxgd4aeufzsz ~]# docker images
REPOSITORY
                      TAG
                                                                   4 seconds ago
5 days ago
                                            5ab13cb87e2a
image_nginx
docker.io/tomcat
                      latest
                                            5377fd8533c3
docker.io/nginx
                      latest
                                            719cd2e3ed04
                                                                   9 days ago
```

例如: A、B两台机器都想安装redis, A机器上创建容器并在容器中做好redis的一切配置, 让后将这个容器 docker commit 成镜像image redis, B机器也想要安装redis, 直接用镜像image redis创建容器就行了, docker就是做这样一劳永逸的事情。

而且传统方式得在每台机器上安装配置redis非常麻烦

四. 镜像的导入与导出

镜像压缩打包(主机上进行操作),有两种方式 docker save 与 docker load 和

docker export 与 docker import

docker save nginx | gzip > nginx xin image.tar.gz 将现有的镜像压缩打包

docker load -i nginx_xin_image.tar.gz 压缩的镜像解压

docker images 进行查看

docker save 是直接将镜像进行打包 docker save 〈镜像名〉或〈镜像id〉 docker export container_nginx> nginx_image.tar

cat nginx_image.tar | sudo docker import - nginx_image:import

docker export 是直接将容器进行打包 docker save 〈容器名〉或〈容器id〉

需要注意两种方法配套的,切不可混用。虽然导入导出时没问题,但是在创建容器时候会报错

如果使用import导入save产生的文件,虽然导入不提示错误,但是启动容器时会提示失败,

会出现类似"docker: Error response from daemon: Container command not found or does not exist"的错误。

类似,使用load载入export产生的文件,也会出现问题。

容器的十大误区

https://blog.csdn.net/salove_y/article/details/80095566

https://mp.weixin.qq.com/s/ZVifX50pb210oh gYU1Fgw (原网址)

1.1、docker是linux容器的一种封装,提供简单易用的容器使用接口。它是最流行的Linux容器解决方案。

docker的接口相当简单,用户可以方便的创建、销毁容器。

docker将应用程序与程序的依赖,打包在一个文件里面。运行这个文件就会生成一个虚拟容器。

程序运行在虚拟容器里,如同在真实物理机上运行一样,有了docker,就不用担心环境问题了。

1.2 应用场景

web应用的自动化打包和发布 自动化测试和持续集成、发布 在服务型环境中部署和调整数据库或其他应用

1.4 Docker的三大概念及优势

镜像 image

容器 container

仓库 repository

docker容器的优势

#1、更高效的利用系统资源

由于容器不需要进行硬件虚拟以及运行完整操作系统等额外开销, Docker 对系统 资源的利用率更高。

无论是应用执行速度、内存损耗或者文件存储速度,都要比传统虚拟机技术更高效。因此,相比虚拟机技术,一个相同配置的主机,往往可以运行更多数量的应用。

2、更快速的启动时间

传统的虚拟机技术启动应用服务往往需要数分钟,而 Docker 容器应用,由于直接 运行于宿主内核,无需启动完整的操作系统,因此可以做到秒级、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。

#3、一致的运行环境

开发过程中一个常见的问题是环境一致性问题。由于开发环境、测试环境、生产环 境不一致,导致有些 bug 并未在开发过程中被发现。

而 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。

#4、持续交付和部署

对开发和运维(DevOps)人员来说,最希望的就是一次创建或配置,可以在任意 地方正常运行。

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。开发人员 可以通过 Dockerfile 来进行镜像构建,并结合持续集成(Continuous Integration) 系 统进行集成测试,而运维人员则可以直接在生产环境中快速部署该镜像,甚至结合 持续部署(Continuous Delivery/Deployment) 系统进行自动部署。

而且使用 Dockerfile 使镜像构建透明化,不仅仅开发团队可以理解应用运行环 境,也方便运维团队理解应用运行所需条件,帮助更好的生产环境中部署该镜像。

#5、更轻松的迁移

由于 Docker 确保了执行环境的一致性,使得应用的迁移更加容易。Docker 可以在 很多平台上运行,无论是物理机、虚拟机、公有云、私有云,甚至是笔记本,其运 行结果是一致的。

因此用户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。

2、0Docker安装

系统环境: docker最低支持centos7且在64位平台上,内核版本在3.10以上版本:社区版,企业版(包含了一些收费服务)

官方版安装教程(英文)

https://docs.docker.com/install/linux/docker-ce/centos/#upgrade-docker-after-using-the-convenience-script

博主版安装教程:

- # 安装dockeryum install docker
- # 启动docker systemctl start/status docker
- # 查看docker启动状态docker version

配置加速器

简介: DaoCloud 加速器 是广受欢迎的 Docker 工具,解决了国内用户访问 Docker Hub 缓慢的问题。DaoCloud 加速器结合国内的 CDN 服务与协议层优化,成倍的提升了下载速度。

DaoCloud官网:

https://www.daocloud.io/mirror#accelerator-doc

一条命令加速(记得重启docker)curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s

http://95822026.m.daocloud.io

或者

cat /etc/docker/daemon.json #修改这个文件为如下内容

"registry-mirrors": [

"http://95822026.m.daocloud.io"

```
],
"insecure-registries": []
}
```