

docker基础

docker引擎启动，停止，查看状态

```
查看:systemctl status docker
启动:systemctl start docker
停止:systemctl stop docker
重启:systemctl restart docker
```

docker帮助

```
docker --help
```

镜像

```
查看:docker images
拉取:docker pull 镜像名[:版本号]
```

配置镜像加速器

- 1、sudo vim /etc/docker/daemon.json，创建daemon.json文件，在文件中加入如下内容：

```
{
  "registry-mirrors": ["https://cs913o6k.mirror.aliyuncs.com"]
}
```

- 2、重新加载配置：sudo systemctl daemon-reload
- 3、重启docker：sudo systemctl restart docker

删除、查找镜像

```
查找镜像: docker search 镜像名
删除镜像: docker rmi 镜像名1/镜像id 镜像名2/镜像id
```

容器

创建容器

```
docker run [options] image command [ARG...]
options选项: -i、-t、-d、--name
-i: 交互式容器
-t: tty, 终端
-d: 守护式容器, 后台运行, 并打印容器id
如: docker run -i -t --name=ubuntu_i ubuntu /bin/bash
```

登陆容器

方式一：

`docker attach 容器名称/id` (`ps:exit`, 容器停止)

eg: `docker attach u3`

方式二：

`docker exec -it 容器名称/id /bin/bash` (`ps:exit`, 容器不会停止)

eg: `docker exec -it u3 /bin/bash`

查看容器

`docker ps`: 查看正在运行的容器

`docker ps -a`: 查看运行过的容器 (历史)

`docker ps -l`: 最后一次运行的容器

启动、停止容器

`docker start 容器名称/id`

`docker stop 容器名称/id`

`docker restart 容器名称/id`

删除容器

删除一个容器：

`docker rm 容器名称/id`

删除多个容器：

`docker rm 容器名称1/id1 容器名称2/id2 ...`

删除所有容器

`docker rm $(docker ps -a -q)`

PS: 无法删除正在运行的容器

查看容器/镜像的元数据

查看容器/镜像全部信息: `docker inspect 容器/镜像`

查看容器/镜像部分信息: `docker inspect -f '{{.NetworkSettings.IPAddress}}' 容器/镜像`
-f: 可通过--format代替

查看容器日志

`docker logs 容器名称/id`

文件拷贝

将宿主机中的文件拷贝到容器内

```
docker cp 需要拷贝的文件或目录 容器名称:容器目录
例如: docker cp 1.txt c2:/root
```

将容器内的文件拷贝的宿主机中

```
docker cp 容器名称:容器目录 需要拷贝的文件或目录
例如: docker cp c2:/root/2.txt /root
```

目录挂载

我们可以在创建容器的时候，将宿主机的目录与容器内的目录进行映射，这样我们就可以通过修改宿主机某个目录的文件从而去影响容器。

创建容器 添加-v参数 后边为 宿主机目录:容器目录

```
docker run -id --name=c4 -v /opt:/usr/local/myhtml centos /bin/bash
```

如果共享的是多级的目录，可能会出现权限不足提示

这是因为CentOS7中的安全模块selinux把权限禁掉了，我们需要添加参数 -- privileged=true 来解决挂载的目录没有权限的问题

```
docker run -id --privileged=true --name=c4 -v /opt:/usr/local/myhtml
centos /bin/bash
```

docker进阶

1、docker镜像制作

1.1 docker commit

1.1.1 制作步骤

docker commit: 提交一个正在运行的容器为一个新的镜像

本例：制作一个tomcat镜像，制作步骤：

1、拉取一个基础镜像（其始就是OS）

```
docker pull centos
```

2、创建一个交互式容器

```
docker run -it --name=mycentos centos:latest
```

3、软件上传：将宿主机Tomcat、jdk上传到容器中

```
docker cp apache-tomcat-7.0.47.tar.gz mycentos:/root/
```

```
docker cp jdk-8u161-linux-x64.tar.gz mycentos:/root/
```

4、在容器中安装jdk（yum install java-1.7.0-openjdk）

```
tar -zxvf jdk-8u161-linux-x64.tar.gz -C /usr/local/
```

编辑/etc/profile文件，添加如下内容：

```
JAVA_HOME=/usr/local/jdk1.8.0_161
export PATH=$JAVA_HOME/bin:$PATH
```

5、在容器中安装tomcat

```
tar -zxvf apache-tomcat-7.0.47.tar.gz -C /usr/local/
编辑tomcat/bin/setclasspath.sh文件，添加如下内容：
export JAVA_HOME=/usr/local/jdk1.8.0_161
export JRE_HOME=/usr/local/jdk1.8.0_161/jre
```

6、将正在运行的容器提交为一个新的镜像

```
docker commit mycentos mytomcat
```

1.1.2 端口映射

```
docker run -itd --name=t1 -p 8888[宿主机端口]:8080[容器端口] mytomcat /bin/bash
docker exec t1 /usr/local/apache-tomcat-7.0.47/bin/startup.sh
```

通过宿主机访问: <http://ip:port>

1.1.3 目录挂载

1.1.4 容器/镜像打包

镜像打包：

1、镜像打包: `docker save -o 打包后文件名 要打包的镜像`

```
docker save -o /root/tomcat7.tar mytomcat
```

2、将打包的镜像上传到其他服务器

```
scp tomcat7.tar 其他服务器ip:/root
```

3、导入镜像

```
docker load -i /root/tomcat7.tar
```

容器打包：

1、容器打包: `docker export -o 打包后文件名 要打包的容器`

```
docker export -o /root/t1.tar t1
```

2、导入容器

```
docker import t1.tar mytomcat:latest
```

1.2 docker builder

1.2.1 DSL语法



1.2.2 dockerfile示例

- 1、创建一个目录
- 2、在目录下创建Dockerfile文件以及其他文件

```
#pull down centos image
FROM docker.io/centos
MAINTAINER ruanwen onlien033_login@126.com

#install nginx
RUN yum -y install make pcre pcre-devel openssl openssl-devel gcc vim wget net-
tools
RUN useradd www -M -s /sbin/nologin
RUN cd /usr/local/src && wget http://nginx.org/download/nginx-1.17.9.tar.gz &&
tar -zxvf nginx-1.17.9.tar.gz
RUN cd /usr/local/src/nginx-1.17.9 && rm -rf objs && ./configure --
prefix=/usr/local/nginx --user=www --group=www --with-http_stub_status_module --
with-http_ssl_module
#ADD ./Makefile /usr/local/src/nginx-1.17.9/objs/Makefile
RUN cd /usr/local/src/nginx-1.17.9 && make && make install

ENTRYPOINT /usr/local/nginx/sbin/nginx && tail -f
/usr/local/nginx/logs/access.log
```

- 3、在Dockerfile所在目录下构建镜像

```
sudo docker build -t nginx_i --rm=true .
```

nginx_i 为镜像名称

-t 表示选择指定生成镜像的用户名，仓库名和tag

--rm=true 表示指定在生成镜像过程中删除中间产生的临时容器。

注意：上面构建命令中最后的.符号不要漏了，表示使用当前目录下的Dockerfile构建镜像

4、测试

```
sudo docker run -it -d --name nginx_c -p 8080:80 nginx_i /bin/bash
sudo docker exec nginx_c /bin/bash
```

通过浏览器访问：<http://8080:8899>

2、docker仓库

详情见 [docker02.pdf](#)

3、docker网络管理

详情见 [docker02.pdf](#)

4、搭建docker swarm集群

架构介绍见 [docker02.pdf](#)

搭建步骤

1、环境准备：

1.1、准备三台已安装docker engine的centos/Ubuntu系统主机（docker版本必须在1.12以上的版本，老版本不支持swarm）

1.2、docker容器主机的ip地址固定，集群中所有工作节点必须能访问该管理节点

1.3、集群管理节点必须使用相应的协议并且保证端口可用

集群管理通信：TCP，端口2377

节点通信：TCP和UDP，端口7946

覆盖型网络(docker网络)：UDP，端口4789 overlay驱动

说明：三台容器主机的ip地址分别为：

192.168.200.162（管理节点）

192.168.200.163（工作节点）

192.168.200.158（工作节点）

主机名称分别为：manager1、work1以及work2

vim /etc/hostname（修改完成后需要重启）

2、创建docker swarm

2.1、在manager1机器上创建docker swarm集群

```
docker swarm init --advertise-addr 192.168.200.162
```

（--advertise-addr将该IP地址的机器设置为集群管理节点；如果是单节点，无需该参数）

2.2、查看管理节点集群信息：

```
docker node ls
```

3、向docker swarm中添加工作节点：在两个工作节点中分别执行如下命令，ip地址是manager节点的

3.1、添加两个work节点

```
docker swarm join --token xxx 192.168.200.138:2377 (worker1)
docker swarm join --token xxx 192.168.200.138:2377 (worker2)
```

(`--token xxx`:向指定集群中加入工作节点的认证信息, `xxx`认证信息是在创建docker swarm时产生的)

添加节点时如果产生网络连接错误, 关掉所有机器的防火墙后重试: `systemctl stop firewalld`

3.2、继续查看管理节点集群信息与之前的区别

`docker node ls`

4、在docker swarm中部署服务

在Docker Swarm集群中部署服务时, 既可以使用Docker Hub上自带的镜像来启动服务, 也可以使用自己通过Dockerfile构建的镜像来启动服务。如果使用自己通过Dockerfile构建的镜像来启动服务那么必须先将镜像推送到Docker Hub中心仓库。为了方便读者的学习, 这里以使用Docker Hub上自带的alpine镜像为例来部署集群服务

4.1、部署服务

`docker service create --replicas 1 --name helloworld alpine ping docker.com`

`docker service create`指令: 用于在Swarm集群中创建一个基于alpine镜像的服务

`--replicas`参数: 指定了该服务只有一个副本实例

`--name`参数: 指定创建成功后的服务名称为helloworld

`ping docker.com`指令: 表示服务启动后执行的命令

5.查看docker swarm集群中的服务

查看服务列表: `docker service ls`

查看部署具体服务的详细信息: `docker service inspect` 服务名称

查看服务在集群节点上的分配以及运行情况: `docker service ps` 服务名称

6、修改副本数量

在manager1上, 更改服务副本的数量 (创建的副本会随机分配到不同的节点)

`docker service scale helloworld=5`

7、删除服务 (在管理节点)

`docker service rm` 服务名称

8、访问服务

8.1、查看集群环境下的网络列表: `docker network ls`

8.2、在manager1上创建一overlay为驱动的网络 (默认使用的网络连接ingress)

`docker network create -d=overlay my-multi-host-network`

8.3、在集群管理节点manager1上部署一个nginx服务

`docker service create \`

`--network my-multi-host-network \`

`--name my-web \`

`-p 8080:80 \`

`--replicas 2 \`

`nginx`

8.3、在管理节点查看服务的运行情况:

`docker service ps my-web`

8.4、访问测试

访问测试时, ip要使用服务部署运行的机器ip, 否则访问不成功的

5、docker compose编排工具

5.1、docker compose安装与卸载

1、环境要求：Docker Compose是依赖于Docker引擎的，所以在安装Docker Compose之前 要确保机器上已经安装了Docker。<https://github.com/docker/compose/releases> （查看docker compose版本）

2、下载docker-compose工具

```
curl -L https://github.com/docker/compose/releases/download/1.24.0/docker-  
compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
```

3、设置docker compose可执行文件权限

```
chmod +x /usr/local/bin/docker-compose (r:read w:write x:ex )
```

4、查看docker compose版本

```
docker-compose --version
```

PS:卸载docker compose

```
sudo rm /usr/local/bin/docker-compose
```

5.2、docker compose使用

例子

步骤：分为三步（在创建的一个空目录下执行）

1、编写Dockerfile文件（为每个服务构建需要的镜像，方便迁移-不是必须的）

2、编写docker-compose.yml文件（编写部署服务相关指令）

3、运行docker-compose up（启动yml文件中服务）

案例：

1、准备：两个镜像（本次演示就不通过Dockerfile构建了）

```
docker pull mysql:5.7
```

```
docker pull wordpress
```

2、需要新建一个空白目录，例如rwtest。新建一个docker-compose.yml,编辑该文件：

```
version: '3'
```

```
services:
```

```
  db:
```

```
    image: mysql:5.7
```

```
    volumes:
```

```
      - db_data:/var/lib/mysql
```

```
    restart: always
```

```
    environment:
```

```
      MYSQL_ROOT_PASSWORD: wordpress
```

```
      MYSQL_DATABASE: wordpress
```

```
      MYSQL_USER: wordpress
```

```
      MYSQL_PASSWORD: wordpress
```

```
  wordpress:
```

```
    depends_on:
```

```
      - db
```

```
    image: wordpress:latest
```

```
    ports:
```

```
      - "8001:80"
```

```
    restart: always
```

```
    environment:
```

```
      WORDPRESS_DB_HOST: db:3306
```

```
      WORDPRESS_DB_USER: wordpress
```

```
      WORDPRESS_DB_PASSWORD: wordpress
```

```
volumes:
```

```
  db_data:
```


该文件中内容：新建db和wordpress容器。等同于：

```
docker run --name db -e MYSQL_ROOT_PASSWORD=123456 -d mysql
```

```
docker run --name some-wordpress --link db:mysql -p 8001:80 -d wordpress
```

3、启动docker compose

```
docker-compose up
```

这一步如果反复报错，可以将当前用户加入到docke组： `sudo gpasswd -a ${USER} docker`
不行就：

```
su some_user
```

`docker-compose up -d #` 这两步要连起来，先登录用户，然后才能正确运行，光进入命令行是不够的
或者重启docker服务： `sudo systemctl restart docker`

4、浏览器访问： `http://ip:8001`

5、停止/重启服务： `docker-compose stop/restart`

指令详情见： [docker02.pdf](#)

6、docker的web可视化管理工具
