



链滴

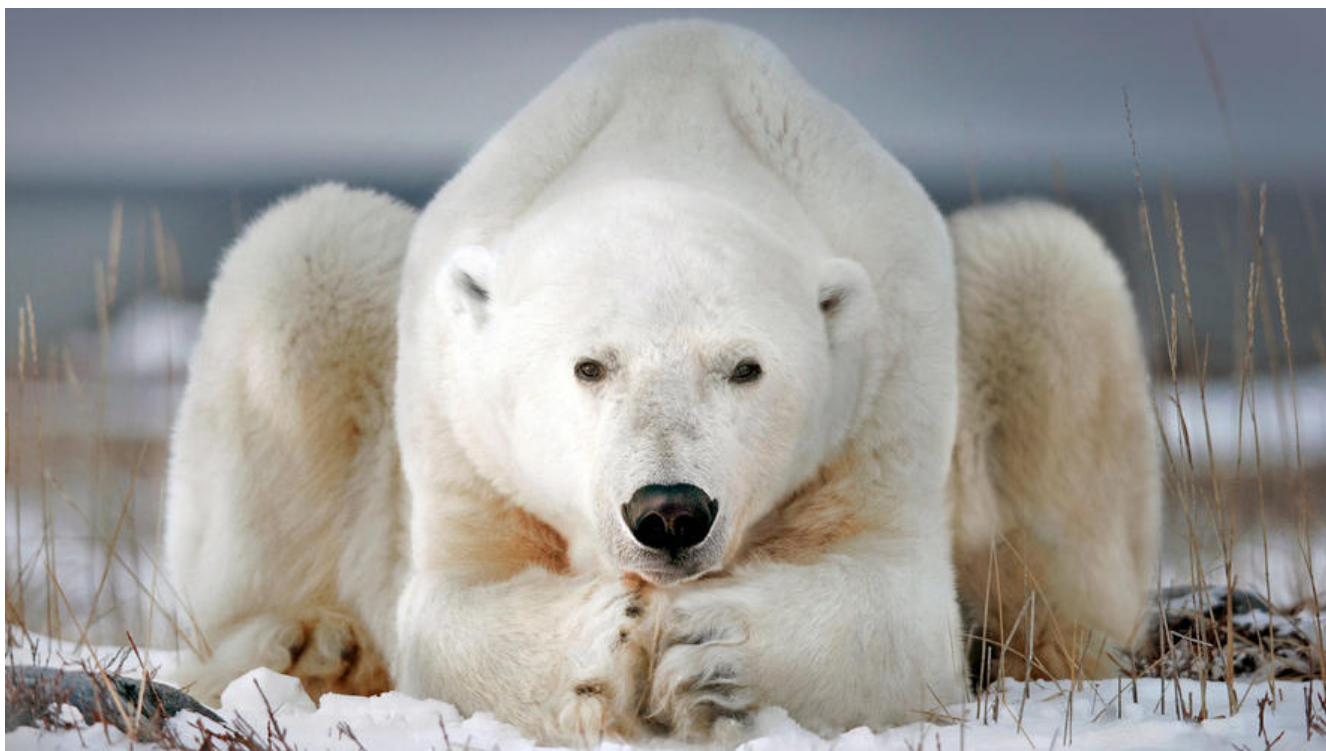
# docker 服务迁移，容器内文件修改

作者: [sirwsl](#)

原文链接: <https://ld246.com/article/1653494458273>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



## 一、问题

将docker容器进行迁移，迁移后需要部署到另一个服务器中，且需要修改配置文件中相应信息

## 二、故事

公司最近由于需要测试新的东西，但是之前没有搭建测试服务。所以导致现在新的东西测试不了，又能直接上到正式系统，因此需要进行服务迁移以及重新部署，但是服务还挺多，自己的权限又没那么，所以就准备直接copy Docker容器，然后启动。

## 三、步骤

### 1、打包所需要的docker镜像

1) 查看运行容器

命令: `docker ps`

```
Last login: wed May 25 17:12:04 2022 from 172.20.56.41
[root@localhost ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
60ded190b2de   test-yth-webscoket  "java -Djava.securit..." 7 hours ago   Up 7 hours
```

2) 将运行的容器保存为image

命令: `docker commit 容器名 将要保存的镜像名`

`docker commit xxl-job test-xxl-job`

```
[root@localhost ~]# docker commit test-yth-webscoket test
sha256:528cb5f43b4cfceec06732b89b6e55d003a3f226bcb582e2e0f73b7f120ad6607
[root@localhost ~]#
```

### 3) 查看保存的image

命令: **docker images**

```
sha256:528cb5f43b4c1ce06752b89b6e55d003a3f2266cb582e2e0f73b7f120ad6607
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
test                 latest             528cb5f43b4c      45 seconds ago    690MB
```

### 4) 将images 打包

命令: **docker save -o 保存的地址 镜像名称多个**

-o : 保存地址

PS: 打包需要时间所以莫急, 慢慢等, 并不是卡住了

`docker save -o /data/test.tar test`

`docker save test > images.tar`

```
[root@localhost ~]# docker save -o /data/test.tar test
[root@localhost ~]# cd /data
[root@localhost data]# ls
minio  mysql  mysql57  nacos  redis  test.tar  xxl-job
[root@localhost data]#
```

或者是

```
[root@localhost data]# docker save test > images.tar
[root@localhost data]# ls
images.tar  minio  mysql  mysql57  nacos  redis
[root@localhost data]#
```

### 5) 将tar包 传输或下载到目标服务器

命令: **scp 传输文件 目标账号@ip地址: 目标服务器地址**

`scp -r test.tar root@172.0.0.29:/data`

```
images.tar  minio  mysql  mysql57  nacos  redis  test.tar  xxl-job
[root@localhost data]# scp -r test.tar root@172.0.0.29:/data
root@172.0.0.29's password:
test.tar
100% 682MB 170.5MB/s csc00:04
```

如果是备份到此结束

## 2、构建运行镜像

### 1) 登录目标服务器、查看文件

命令: **ssh 用户名@ip**

`ssh root@172.0.0.1`

```
[root@localhost ~]# cd /data
[root@localhost data]# ls
nginx  test.tar
[root@localhost data]#
```

## 2) 解压tar包

**命令: docker load -i tar包名称**

`docker load -i test.tar`

```
[root@localhost data]# ls
nginx test.tar
[root@localhost data]# docker load -i test.tar
e0c46f1bcad4: Loading layer [=====>] 55.77MB/55.77MB
21fbcf1ddf25: Loading layer [=====>] 1.536kB/1.536kB
bbca9ecee8a8: Loading layer [=====>] 2.048kB/2.048kB
5fa23ecd1301: Loading layer [=====>] 2.56kB/2.56kB
Loaded image: test:latest
[root@localhost data]#
```

## 3) 运行容器

**命令: docker run ... image名称**

-d, --detach=false, 指定容器运行于前台还是后台, 默认为false

-t, --tty=false, 分配tty设备, 该可以支持终端登录, 默认为false

-w, --workdir="", 指定容器的工作目录

-e, --env=[], 指定环境变量, 容器中可以使用该环境变量

-p, --publish=[], 指定容器暴露的端口

-v, --volume=[], 给容器挂载存储卷, 挂载到容器的某个目录

--name="", 指定容器名字, 后续可以通过名字进行容器管理, links特性需要使用名字

--net="bridge", 容器网络设置: bridge host none

--privileged=false, 指定容器是否为特权容器, 特权容器拥有所有的capabilities

--restart="no", 指定容器停止后的重启策略: no on-failure always

--rm=false, 指定容器停止后自动删除容器(不支持以docker run -d启动的容器)

--add-host=xxx.ip 指定host

`docker run -d --network=host --name test1 test`

```
[root@localhost data]# docker run -d --network=host --name test1 test
3a17e92c6ed679a33943b0d826eae8814a196251bae40b73afc8c499de5a57
[root@localhost data]# dokcer ps
-bash: dokcer: 未找到命令
[root@localhost data]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
3a17e92c6ed679a33943b0d826eae8814a196251bae40b73afc8c499de5a57    test               "java -Djava.securit..."   8 seconds ago      Up 7 seconds              test1 @sirwsl
```

建议在不知道端口的情况下直接--network=host 出现冲突再解决

如果是运行容器到此结束

## 3、修改docker 中jar包配置文件

### 1) 进入容器

**命令: docker exec -it 容器名 /bin/bash**

`docker exec -it test1 /bin/bash`

```
bash: docker: 未找到命令
[root@localhost data]# docker exec -it test1 /bin/bash
root@localhost:/#
```

## 2) 查看jar包文件

**命令: jar tf jar包名称**

jar tf xxx.jar

```
root@localhost:/# ls
bin dev home lib64 mnt proc run srv tmp var
boot etc lib media opt root sbin sys usr web_socket.jar
root@localhost:/# jar tf web_socket.jar
```

结果:

```
BOOT-INF/classes/cn/yth/df/socket/config/SocketIOConfig.class
BOOT-INF/classes/cn/yth/df/socket/config/SocketIOProperties.class
BOOT-INF/classes/cn/yth/df/socket/config/SocketIOConfig$2.class
BOOT-INF/classes/cn/yth/df/socket/spring/R.class
BOOT-INF/classes/cn/yth/df/socket/spring/BaseController.class
BOOT-INF/classes/cn/yth/df/socket/spring/Api.class
BOOT-INF/classes/db/data.db
BOOT-INF/classes/application.yml
META-INF/maven/
META-INF/maven/cn.yth.df/
META-INF/maven/cn.yth.df/web_socket/
META-INF/maven/cn.yth.df/web_socket/pom.xml
```

## 3) 找到对应的配置文件, yml或者properties文件,提取文件

**命令: jar xf jar包名称 路径**

jar xf web\_socket.jar BOOT-INF/classes/application.yml

```
root@localhost:/# jar xf web_socket.jar BOOT-INF/classes/application.yml
root@localhost:/# ls
BOOT-INF boot etc lib media opt root sbin sys usr web_socket.jar
bin dev home lib64 mnt proc run srv tmp var yth-form.jar
root@localhost:/# cd /BOOT-INF
root@localhost:/BOOT-INF# cd /classes
bash: cd: /classes: No such file or directory
root@localhost:/BOOT-INF# cd classes
root@localhost:/BOOT-INF/classes# ls
application.yml
root@localhost:/BOOT-INF/classes#
```

## 4)退出容器将文件cp出来编辑

**命令: docker cp 容器名: 路径 目标文件**

docker cp test1:/BOOT-INF/classes/application.yml application.yml

```
exit
[root@localhost data]# docker cp test1:/BOOT-INF/classes/application.yml application.yml
[root@localhost data]# ls
application.yml nginx test.tar
[root@localhost data]#
```

因为docker容器内没得vim, 不嫌麻烦可以安装一个, 建议copy出来修改。之后使用vim or vi 编辑

件，修改好后保存

---

5) 将修改好的文件copy回原来的容器

**命令： docker cp 源文件 容器名: 地址**

`docker cp application.yml test1:/BOOT-INF/classes/application.yml`

```
[root@localhost data]# docker cp application.yml test1:/BOOT-INF/classes/application1.yml
[root@localhost data]# docker cp application.yml test1:/BOOT-INF/classes/application.yml
[root@localhost data]# docker exec -it test1 /bin/bash
root@localhost:/# cd /BOOT-INF/classes/
root@localhost:/BOOT-INF/classes# ls
application.yml  application1.yml
root@localhost:/BOOT-INF/classes#
```

CSDN @sirwsl

PS: 此处的application1.yml 只是因为做演示，copy时候切记别修改文件名称  
也可使用 cat命令 查看是否已经修改 cat xxx

---

6) 将文件copy回jar中

**命令： jar uf jar包名称 文件地址**

`jar uf web_socket.jar BOOT-INF/classes/application.yml`

```
dev      jartmp8675595264873860055.tmp  mnt      run      tmp      yth-form.jar
root@localhost:/# jar uf web_socket.jar BOOT-INF/classes/application.yml
root@localhost:/#
```

CSDN @sirwsl

7) 退出容器， docker restart 结束!

**OVER**