# Docker

## 概念

### 镜像

用于创建容器的只读模板。

### 容器

简易版Linux系统环境与应用程序打包而成的盒子。

基于镜像创建，启动时在镜像的最上层创建一个可写层。

容器是无状态的，不会保存至原有镜像，需要基于该容器重新构建一个镜像。

### 仓库

默认使用DockerHub仓库，概念类似于Github，但服务器在国外，上传下载速度慢。

可使用第三方云平台作为仓库，如[阿里云](https://cs.console.aliyun.com/)。



## 安装Docker

### 2.1 Centos7下安装Docker

**1.centos7安装docker ce**

# 安装依赖包

sudo yum install -y yum-utils device-mapper-persistent-data lvm2

# 添加软件源

sudo yum-config-manager --add-repo https://mirrors.ustc.edu.cn/docker-ce/linux/centos/docker-ce.repo

# 更新缓存/安装docker ce

sudo yum makecache fast

sudo yum install docker-ce

**2.启动docker ce**

sudo systemctl enable docker

sudo systemctl start docker

**3.将docker加入用户组**

sudo groupadd docker

sudo usermod -aG docker $USER

**4.测试docker**

docker run hello-world

参考：[Centos下安装docker](https://yeasy.gitbooks.io/docker_practice/install/centos.html)

## 镜像

### 3.1 镜像加速

利用阿里云容器镜像服务中的加速器来下载镜像。

可以通过docker info来查看是否启用了该加速器。



### 3.2 搜索镜像

在官方提供的镜像中所有你所需要的镜像

docker search --filter=is-official=true nginx

### 3.3 获取镜像

下载镜像至本地

docker pull 镜像名称:版本标签

### 3.4 查看镜像信息

docker inspect 镜像id 或者 镜像名称:标签

### 3.5 删除镜像

建议先删除依赖该镜像的所有容器，再来删除镜像

docker rm 容器id 或者 容器名称

docker rmi 镜像id 或者 镜像名称

或者，强制删除镜像，依赖它的容器会一并删除

docker rmi 镜像id 或者 镜像名称 -f

### 3.6 清理镜像

强制清理镜像

docker image prune -f

## 容器

### 4.1 新建并启动容器

docker run -it 镜像id 或者 镜像名称:便签

-i 标准输入

-t 伪终端

-d 守护状态运行

--name 命名容器

-p 端口映射(宿主机端口:容器端口)

-P 端口映射(随机端口)

-v 文件挂载 (-v $PWD/redis/conf/redis.conf:/etc/redis/redis.conf)

本地文件/路径:容器文件/路径

不论修改宿主机还是容器内挂载的文件，会同时变化，避免要修改容器中文件内容

而需要安装vim或者复制文件至容器的操作。

--link 容器间通信(--link django\_tsdrm:django\_server)

关联的容器名称/id:关联容器服务器地址的别名(自动加载至hosts文件)

### 4.2 进入容器

进入的容器必须已经启动

docker exec -it 容器id 或者 容器名称 /bin/bash

### 4.3 启动容器

docker start 容器id 或者 容器名称

### 4.4 暂停容器

docker pause 容器id 或者 容器名称(unpause取消暂停)

### 4.5 关闭容器

docker stop 容器id 或者 容器名称

### 4.6 查看容器详情

docker inspect 容器id 或者 容器名称

### 4.7 删除容器

docker rm 容器id 或者 容器名称(只删除已经关闭或者退出的容器，-f表示强制)

### 4.8 复制文件至容器

docker cp 本地命令执行所在相对路径 容器id/名称：/root/(容器绝对路径)

## 示例

### 5.1 MySQL容器

# 宿主机相关文件

mysql >> conf >> my.cnf # 配置文件

data # 挂载的数据库文件

logs # 挂载的日志文件

# 创建容器(mysql目录下执行):宿主机3306端口可能被宿主机占用

docker run -p 3307:3306 --name mysql\_tsdrm -v $PWD/mysql/conf:/etc/mysql/conf.d -v $PWD/mysql/logs:/logs -v $PWD/mysql/data:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=password -d mysql:5.6

# 进入容器

docker exec -it mysql\_tsdrm /bin/bash # 容器开启的状态下

# 复制sql文件至容器

docker cp /home/username/桌面/tesudrm.sql mysql\_tsdrm:/

# 创建数据库/导入数据

mysql -uroot -p

create dababase example charset=utf8;

mysql -uroot -p example < tesudrm.sql

### 5.2 Redis容器

# 宿主机相关文件

redis >> conf >> redis.conf

data

# docker中redis默认无配置文件启动

# 配置文件redis.conf(根据需求自己配置,参数说明：https://www.cnblogs.com/qq78292959/archive/2013/09/21/3331032.html)

requirepass tesunet

daemonize no

port 6379

# 启动/拉取redis:4.0，同时命名，同时挂载配置文件/数据文件至宿主机当前文件下(执行命令的位置)

docker run -p 6378:6379 --name redis\_tsdrm -v $PWD/redis/conf/redis.conf:/etc/redis/redis.conf -v $PWD/redis/data:/data -d redis:4.0 redis-server /etc/redis/redis.conf

# 添加账户:requirepass tesunet

# 认证: redis-cli: >> AUTH "tesunet"

### 5.3 Django项目容器

# 宿主机相关文件

django\_tsdrm >> requirements.txt

>> TSDRM(项目)

# settings.py mysql的host改成db,mysql\_tsdrm表示mysql容器的名称(容器间通信)。

docker run -p 8000:8000 -v $PWD/django\_data/TSDRM:/TSDRM --link mysql\_tsdrm:db\_server --link redis\_tsdrm:redis\_server --name django\_tsdrm -itd username/tp\_pro:1.1 (命令位置写错很麻烦)

# 复制requiremens.txt至容器

docker cp requirements.txt django\_tsdrm:/

# 进入容器

docker exec -it django\_tsdrm /bin/bash

# 安装依赖

pip3 install -r requirements.txt

# 修改settings.py

>>> redis/mysql的host改为:db\_server/redis\_server

# 运行项目(django内置的wsgiref模块)

python manage.py runserver 0.0.0.0:8000

# 访问项目url

http://ip:8000

# 运行celery/celery beat/celery flower

python manage.py celery -A TSDRM worker -l info

python manage.py celery -A TSDRM flower -l info

python manage.py celery -A TSDRM beat -l info

# \*\*\*\*\*\*\* 基于uwsgi启动django \*\*\*\*\*\*\*\*\*

# 1.安装uwsgi

pip3 install uwsgi

# 2.配置uwsgi.ini

[uwsgi]

socket=0.0.0.0:8000 # 配置nginx代理时使用

#http=0.0.0.0:8000

chdir=/TSDRM

#module=TSDRM.wsgi:application

wsgi-file=/TSDRM/TSDRM/wsgi.py

master=True

process=4

pidfile=/TSDRM/TSDRM-master.pid

vacuum=True

max-requests=5000

daemonize=/TSDRM/log/wsgi.log

static-map=/static=/TSDRM/static

# 3.collectstatic命令收集静态文件至项目目录下

python manage.py collectstatic

# 4.注释掉STATIC\_ROOT，添加

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, "static")

]

# 5.urls.py添加

from django.conf import settings

from django.conf.urls.static import static

urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

# 附：

# 1.命令的方式启动

uwsgi --static-map /static=/TSDRM/static --http 0.0.0.0:80 --chdir /TSDRM --wsgi-file /TSDRM/TSDRM/wsgi.py

# 2.配置文件的方式启动

uwsgi --ini uwsgi.ini

### 5.4 Nginx容器

# 1.下载镜像/安装容器

docker run -v $PWD/nginx/log:/var/log/nginx/ -v $PWD/nginx/nginx.conf:/etc/nginx/nginx.conf --link django\_tsdrm:django\_server --name nginx\_tsdrm -d -p 1337:80 nginx

docker run --name test\_nginx -d -p 1338:80 nginx

# 2.进入容器

docker exec -it nginx /bin/bash

# 缺省的Nginx安装在/var/lib/nginx下，配置在/etc/nginx下面，包括nginx.conf主配置文件，conf.d和site-enabled是附加的配置文件。后面，我们将会该目录映射到宿主机目录，以便于修改和管理。

# 3.配置nginx

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

server {

listen 80;

server\_name localhost;

charset utf-8;

location / {

include uwsgi\_params;

uwsgi\_pass django\_server:8000;

#uwsgi\_param UWSGI\_SCRIPT TSDRM.wsgi;

#uwsgi\_param UWSGI\_CHDIR /TSDRM;

}

#location /static/ {

# alias /TSDRM/static/;

#}

}

}

## 宿主机文件目录设置

tsdrm\_docker

> django\_data

> requirements.txt

> TSDRM

> mysql

> conf

> data

> log

> redis

> conf

> redis.conf

> data

> nginx

> log

> nginx.conf

## 强调

**执行命令时注意命令中宿主机挂载文件的路径，在tsdrm\_docker下执行。**

## 八．[其他具体操作](https://www.cnblogs.com/pymkl/articles/10477833.html)