Docker

Docker 容器使用

获取镜像

```
$ docker pull ubuntu
```

启动容器

```
$ docker run -it ubuntu /bin/bash
```

参数说明：

- **-i**:交互式操作
- **-t**:终端
- **ubuntu**:ubuntu 镜像
- **/bin/bash**: 放在镜像名称后面的是命令，这里我们希望有一个交互式Shell，因此用的是/bin/bash.

要退出终端，直接输入**exit**.

启动已经停止运行的容器

```
$ docker ps -a # 列出所有停止的容器
# docker start [容器编号]
$ docker start b750bbbcfd88
```

后台运行

```
$ docker run -itd -name ubuntu-test ubuntu /bin/bash
```

停止一个容器

```
$ docker ps
# docker ps 列出所有的运行的容器。
$ docker stop [容器ID]
# 重启一个已经停止的容器
$ docker start [容器ID]
# 重启一个（正在运行）容器
$ docker restart <容器ID>
```

进入容器

在使用**-d**参数时，容器启动后会进入后台。此时想要进入容器，可以通过以下指令进入:

- **docker attach**
- **docker exec**: 推荐大家使用docker exec, 因为此退出终端，不会导致容器的停止。

```
$ docker attach 1e560fca3906
$ docker exec -it 243c32535da7 /bin/bash
```

导入和导出容器

如果要导出本地某个容器，可以使用**docker export**命令

```
$ docker export 1e560fca3906 > ubuntu.tar # 导出容器
$ cat docker /ubuntu.tar | docker import - test/ubutu:v1
$ docker import http://example.com/exampleimage.tgz example/imagerepo
```

删除容器

```
$ docker rm -f 1e560fca3906
$ docker container prune # 清理掉所有处于终止状态的容器。
```

运行一个Web应用

```
$ docker pull training/webapp #载入镜像
$ docker run -d -p training/webapp python app.py
```

参数说明:

- **-d**:让容器在后台运行
- **-p**:将容器内部使用的网络端口映射到我们使用的主机上。

我们也可以通过**-p**参数来设置不一样的端口:

```
$ docker run -d -p <主机port>:<容器port> training/webapp python app.py
```

网络端口查看

```
$ docker port [<容器ID> | <容器名称>]
```

查看web应用程序日志

docker logs [ID或者名称]

```
$ docker logs -f bf08b7f2cd89
```

-f让**docker logs**像使用**tail -f**一样来输出容器内部的标准输出。

查看web应用程序容器的进程

```
$ docker top <容器名称>
$ docker inspect <容器名称> # 他会返回一个JSON格式的文件记录
```

Docker 镜像使用

列出镜像列表

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu              14.04               90d5884b1ee0        5 days ago
 188 MB
php                 5.6                 f40e9e0f10c8        9 days ago
 444.8 MB
nginx               latest              6f8d099c3adc        12 days ago
182.7 MB
mysql               5.6                 f2e8d6c772c0        3 weeks ago
324.6 MB
httpd               latest              02ef73cf1bc0        3 weeks ago
194.4 MB
ubuntu              15.10               4e3b13c8a266        4 weeks ago
136.3 MB
hello-world         latest              690ed74de00f        6 months ago
 960 B
training/webapp     latest              6fae60ef3446        11 months ago
348.8 MB
```

各个选项说明:

- **REPOSITORY**:表示镜像的仓库源
- **TAG**:镜像的标签
- **IMAGE ID**:镜像ID
- **CREATED**:镜像创建时间
- **SIZE**:镜像大小

同一个仓库源可以有多个TAG，代表这个仓库源的不同版本，如Ubuntu仓库源中，有15.10，14.04

所以，我们如果要使用版本15.10的Ubuntu系统镜像来运行容器时，可以用

```
$ docker run -t -i ubuntu:15.10 /bin/bash
```

参数说明:

- **-i**:交互式操作

- **-t**:终端
- **ubuntu:15.10**:这是指用ubuntu 15.10版本镜像为基础来启动容器。
- **/bin/bash**:放在镜像名称后的是命令，这里我们希望有一个交互式Shell.

获取一个新镜像

```
$ docker pull ubuntu:13.10
# 查看镜像
$ docker search httpd
# 拖取镜像
$ docker pull httpd
# 使用镜像
$ docker run httpd
# 删除镜像
$ docker rmi hello-world
```

Docker 在Dotnet中的应用

1. 创建DotnetCore程序(API,WS都行)
2. 添加`Docker`的支持，注意几个修改地方

```
FROM microsoft/dotnet2.2-aspnetcore-runtime AS base
WORKDIR /app
EXPOSE 80
# 此处端口号需要和最后一行端口对应起来
ENV ASPNETCORE_URLS=http://+:5000
FROM microsoft/dotnet2.2-sdk AS Build
WORKDIR /src
COPY ["myappforwin/myappforwin.csproj","myappforwin"]
RUN dotnet restore "myappforwin/myappforwin.csproj"
COPY ..
WORKDIR "/src/myappforwin"
RUN dotnet publish "myappforwin.csproj" -c Release -o /app

FROM build AS final
WORKDIR /app
COPY --from=publish /app
# 此处修改为
ENTRYPOINT ["dotnet","myappforwin.dll","--server.urls","http://0.0.0.0:5000"]
```

3. 不要采用网上`docker`运行发布的程序，直接右键`dockerfile`生成`docker`镜像。
4. 生成成功后，在`docker`中输入`docker images`可以看到多了两个镜像，一个是你的netcore程序，本文为myappforwin，另外一个是空镜像。
5. 关键一部分，输入命令，运行docker并将docker的端口号对应到本地端口，比如docker run -it -p 1111:5000 myappforwin

   运行效果

   warning:Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]

   No Xml encryptor configurated. Key {abb28xxx.xxx.xxx} my be persisted to storage in unencrypted form.

   Hosting environment Production.

Content root path /app .

Now listening on:http://[...]:5000

Application started. Press Ctrl+C to shut down.

6. 在本地运行http://localhost:1111/xxx 能正常访问就行。