**Docker学习**

# 简介

Docker 是一个开源的应用容器引擎，基于 Go 语言 并遵从Apache2.0协议开源。

Docker 可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。

容器是完全使用沙箱机制，相互之间不会有任何接口（类似 iPhone 的 app）,更重要的是容器性能开销极低。

原理：

1. docker其实就是一个沙盒，基于linux，可以安装各种环境
2. 在本地开发环境配置好docker后，上传到云服务器上
3. 在需要部署的计算机中安装docker，然后从云服务器上拉取对应的项目到本地中
4. 在拉取好的项目中运行我们的python程序

# 文档

官方文档：<https://docs.docker.com/get-started/#docker-concepts>

下载地址：<https://docs.docker.com/install/#supported-platforms>

下载ce版本，免费的，如果要下载ee版本，是收费的

Docker Toolbox下载：<https://docs.docker.com/toolbox/overview/>

Docker Toolbox安装指南：<https://docs.docker.com/toolbox/toolbox_install_windows/#step-1-check-your-version>

# 安装

## windows10安装docker

安装下载的 Docker for Windows Installer.exe ，如果出现：

|  |
| --- |
| Docker for Windows requires Windows 10 Pro or Enterprise version 14393, or Windows server 2016 RTM to run |

出现上面的提示，表示你的操作系统不能安装Docker for Windows Installer.exe

那么就安装 Docker Toolbox：

在 文档 章节中查看下载链接和安装指南，下载并且正确安装 Docker Toolbox

注意事项：

1. 安装前，需要去bios启用虚拟设置，这样虚拟机才能运行
2. 安装后，运行 Docker Quickstart Terminal，

如果出现 windows 正在查找 bash.exe

那么就修改桌面 Docker Quickstart Terminal 的 属性，目标："D:\Program Files\Git\bin\bash.exe" --login -i "d:\Program Files\Docker Toolbox\start.sh" 修改对应的bash.exe的位置

## ubuntu安装docker

### 确认ubuntu版本

只能下面这3个版本才能成功安装：

* Artful 17.10 (Docker CE 17.11 Edge and higher only)
* Xenial 16.04 (LTS)
* Trusty 14.04 (LTS)

### 卸载旧版本

$ sudo apt-get remove docker docker-engine docker.io

如果当前ubuntu中没有 docker， 也OK

### 更新apt索引

$ sudo apt-get update

### 安装软件包以允许apt通过HTTPS使用存储库

$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

### 添加Docker的官方GPG密钥

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

添加后，确认：

$ sudo apt-key fingerprint 0EBFCD88

### 设置稳定的存储库

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

### 安装最新版本的Docker CE

$ sudo apt-get install docker-ce

# 教程

## 测试版本

运行docker --version并确保您拥有受支持的Docker版本：

docker --version

Docker version 18.03.0-ce, build 0520e24302

运行docker info或（docker version without --）查看关于docker安装的更多详细信息：

docker info

Containers: 2

Running: 0

Paused: 0

Stopped: 2

Images: 1

Server Version: 18.04.0-ce

Storage Driver: aufs

...

## 测试安装

1. 通过运行简单的Docker镜像[hello-world来](https://hub.docker.com/_/hello-world/)测试您的安装是否工作正常 ：

docker run hello-world

Unable to find image 'hello-world:latest' locally

latest: Pulling from library/hello-world

ca4f61b1923c: Pull complete

Digest: sha256:ca0eeb6fb05351dfc8759c20733c91def84cb8007aa89a5bf606bc8b315b9fc7

Status: Downloaded newer image for hello-world:latest

Hello from Docker!

This message shows that your installation appears to be working correctly.

...

1. 列出hello-world下载到您的机器的镜像：

docker image ls

1. 列出hello-world显示消息后退出的容器（由镜像衍生）。如果它仍在运行，则不需要该--all选项：

docker container ls --all

CONTAINER ID IMAGE COMMAND CREATED STATUS

54f4984ed6a8 hello-world "/hello" 20 seconds ago Exited (0) 19 seconds ago

## 回顾命令

## List Docker CLI commands

docker

docker container --help

## Display Docker version and info

docker --version

docker version

docker info

## Execute Docker image

docker run hello-world

## List Docker images

docker image ls

## List Docker containers (running, all, all in quiet mode)

docker container ls

docker container ls --all

docker container ls -aq

## 创建容器

### Dockerfile

创建一个空目录。将目录（cd）更改为新目录，创建一个名为的文件Dockerfile，将以下内容复制并粘贴到该文件中并保存。注意解释新Dockerfile中每条语句的注释。

# Use an official Python runtime as a parent image

FROM python:3.6

# Set the working directory to /app

WORKDIR /app

# Copy the current directory contents into the container at /app

ADD . /app

# Install any needed packages specified in requirements.txt

RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container

EXPOSE 80

# Define environment variable

ENV NAME World

# Run app.py when the container launches

CMD ["python", "app.py"]

### requirements.txt

Flask

Redis

现在运行build命令。这会创建一个Docker镜像，我们将使用-t它来标记，因此它有一个友好的名称。

docker build -t friendlyhello .

镜像在你的机器的本地Docker镜像注册表中：

$ docker image ls

REPOSITORY TAG IMAGE ID

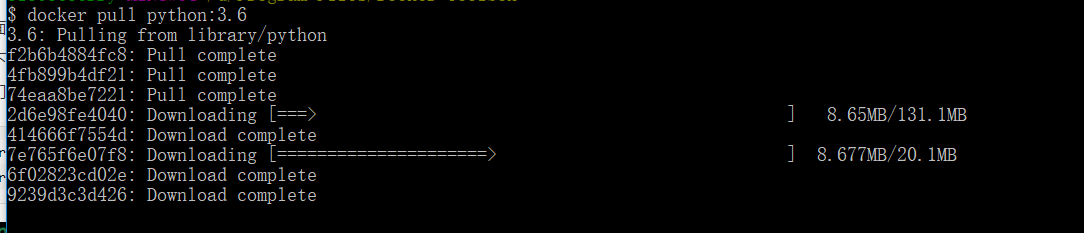
friendlyhello latest 326387cea398

## 从仓库获取资源

命令：

|  |
| --- |
| docker pull python:3.6 |

结果如下：



## 镜像

|  |
| --- |
| # 查看所有镜像  docker images  # 删除镜像  docker rmi 镜像id或镜像名 |

## 容器

docker ps -a  
主要有些参数要说一下   
1. 不加参数，表示查看当前正在运行的容器  
2. -a，查看所有容器包括停止状态的容器  
3. -l，查看最新创建的容器   
4. -n=x，查看最后创建的x个容器   
列一下docker ps的结果   
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES   
这些分别代表什么   
CONTAINER ID：容器ID，唯一标识容器   
IMAGE：创建容器时所用的镜像   
COMMAND：在容器最后运行的命令   
CREATED：容器创建的时间   
STATUS：容器的状态（你会看到UPXXX，表示运行状态）  
PORTS：对外开放的端口号   
NAMES：容器名（也具有唯一性，docker是不允许创建容器名相同的容器的）

|  |
| --- |
| # 查看容器  docker ps -a  # 终止容器  docker stop 容器名或容器ID  # 删除  docker rm 容器名 |

# 项目

创建自己的镜像

## 创建目录

|  |
| --- |
| mkdir py  cd py |

## 编写 Dockerfile

|  |
| --- |
| # Use an official Python runtime as a parent image  FROM python:3.6  # Set the working directory to /app  WORKDIR /app  # Copy the current directory contents into the container at /app  ADD . /app  # Install any needed packages specified in requirements.txt  RUN pip install -i http://pypi.douban.com/simple/ --trusted-host=pypi.douban.com -r requirements.txt  # Make port 80 available to the world outside this container  EXPOSE 80  # Define environment variable  ENV NAME World  # Run app.py when the container launches  # CMD ["scrayd"] |

## 编写requirements.txt

|  |
| --- |
| scrapy==1.5.0  scrapyd==1.2.0  enum-compat==0.0.2  w3lib==1.19.0  redis==2.10.6  scrapy-redis==0.6.8 |

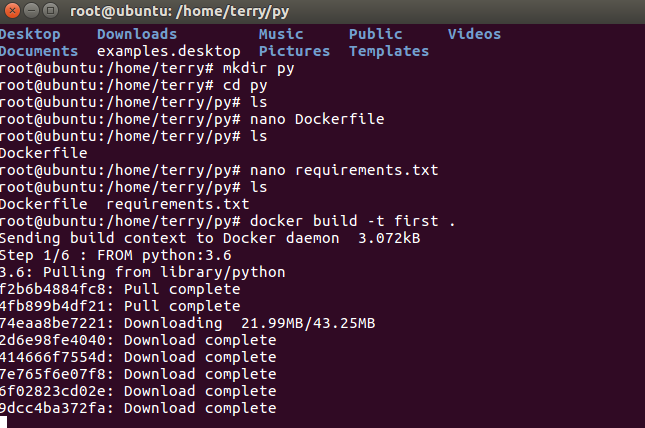
## 创建 镜像

|  |
| --- |
| docker build -t dtest . |

PS： dtest 是自定义的镜像名

后面有个 . 号，不要忘记

这一步有时候网络连接有问题，显示timeout，换个时间重新执行就好



下载完成后，通过以下命令查看

|  |
| --- |
| docker image ls  docker iamges |

## 启动容器

|  |
| --- |
| # 启动指定镜像名，不指定tag，则默认是lastest  docker run -it -p 6800:6800 镜像名 sh |

## 拷贝文件

从主机复制到容器

|  |
| --- |
| # 查看容器ID  docker ps -a  docker cp host\_path containerID:container\_path  # 拷贝scrapyd的配置文件 , f0a485e9d01a容器对应的ID  docker cp f:/default\_scrapyd.conf f0a485e9d01a:/usr/local/lib/python3.6/site-packages/scrapyd/default\_scrapyd.conf |

从容器复制到主机

|  |
| --- |
| docker cp containerID:container\_path host\_path  docker cp f0a485e9d01a:/usr/local/lib/python3.6/site-packages/scrapyd/default\_scrapyd.conf d:/default\_scrapyd.conf |

## 生成新的镜像

|  |
| --- |
| docker commit 容器id或容器名 新的镜像名:tag |

## 启动容器

输入以下命令：

|  |
| --- |
| # 启动指定镜像名，不指定tag，则默认是lastest  docker run -it -p 6800:6800 镜像名 sh  # 启动指定镜像ID  docker run -it 镜像id |

可以进入安装好的镜像中，就像普通的linux一样

## 启动scrapyd

|  |
| --- |
| # 在docker容器中  scrapyd |

## 修改scrapy项目配置

|  |
| --- |
| # 修改scrapy项目的settings  REDIS\_URL = **'redis://192.168.99.1:6379'**  #修改scrapy项目的scrapy.conf  **[deploy:127] url** = **http://192.168.99.100:6800/addversion.json** |

## 修改redis的配置

|  |
| --- |
| bind 0.0.0.0 |

## 上传项目到scrapyd

|  |
| --- |
| *python scrapyd-deploy 127 -p sr* |

# 上传项目到阿里云

## 阿里云注册账号

<https://dev.aliyun.com/search.html>

## 创建镜像仓库的命名空间

例如： mumutest

## 创建镜像仓库

例如：mumu

## 在docker登录阿里云

|  |
| --- |
| docker login --username=mumuloveshine1 registry.cn-hangzhou.aliyuncs.com |

## 修改镜像的名称

|  |
| --- |
| # 查看所有镜像，找到需要上传的镜像id  docker images  # 186a34ea8e3d：镜像ID  # mumutest： 命名空间  # mumu： 镜像仓库  docker tag 186a34ea8e3d registry.cn-hangzhou.aliyuncs.com/mumutest/mumu |

## 上传镜像到阿里云

|  |
| --- |
| docker push registry.cn-hangzhou.aliyuncs.com/mumutest/mumu |

## 下载服务器的镜像

|  |
| --- |
| docker pull registry.cn-hangzhou.aliyuncs.com/mumutest/mumu |