

# Docker入门



# 为什么需要docker

- 软件开发的麻烦之一，就是环境配置。
- 开发者必须保证两件事：操作系统的设置，各种库和组件的安装。只有它们都正确，软件才能运行。如果某些老旧的模块与当前环境不兼容，都可能会导致运行失败。

**Debug your app, not  
your environment**

Securely build and share any application, anywhere

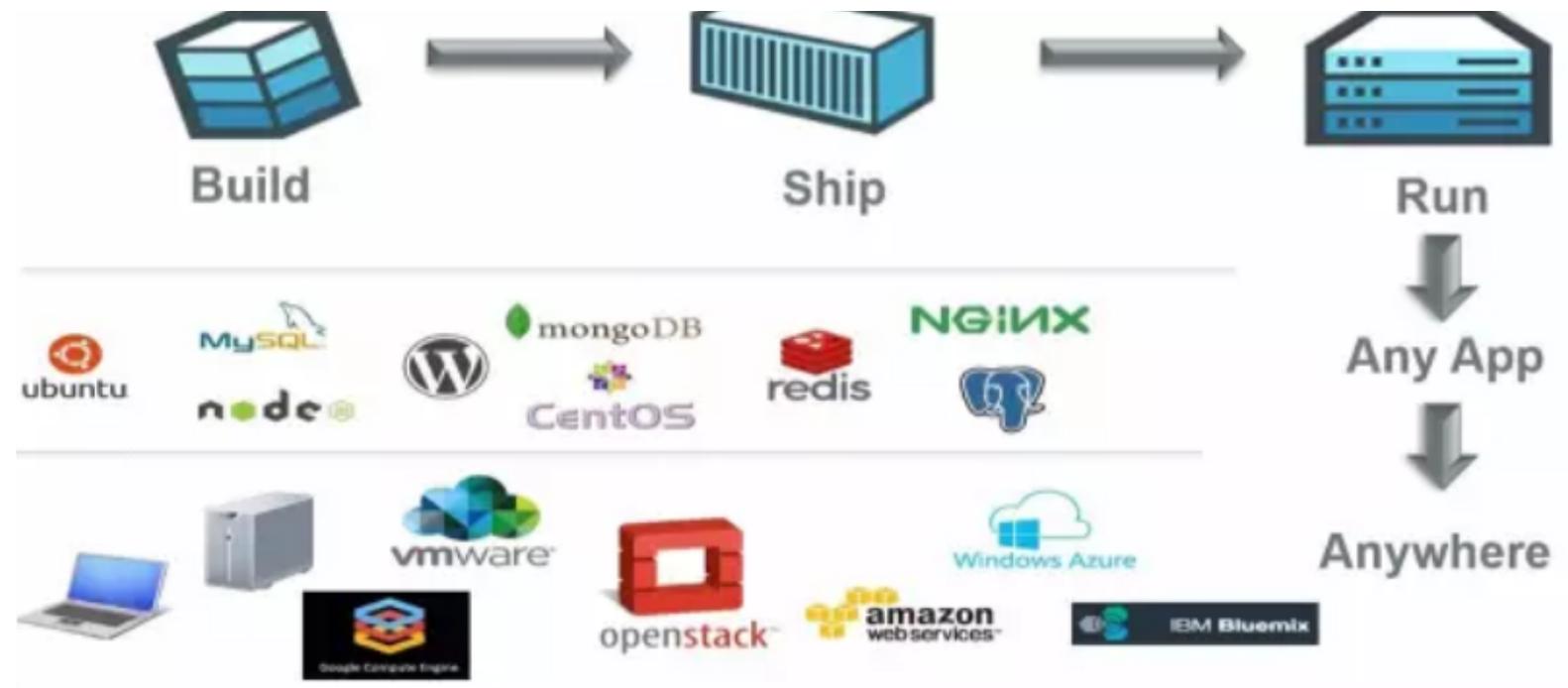
# 虚拟机 ( virtual machine )

- 虚拟机 ( virtual machine ) 就是带环境安装的一种解决方案。它可以在一种操作系统里面运行另一种操作系统。
  - 资源占用多
    - 虚拟机会独占一部分内存和硬盘空间。它运行的时候，其他程序就不能使用这些资源了。哪怕虚拟机里面的应用程序，真正使用的内存只有 1MB，虚拟机依然需要几百 MB 的内存才能运行。
  - 启动慢
    - 启动操作系统需要多久，启动虚拟机就需要多久。可能要等几分钟，应用程序才能真正运行。
  - 交互麻烦
    - 借助于 VM tools

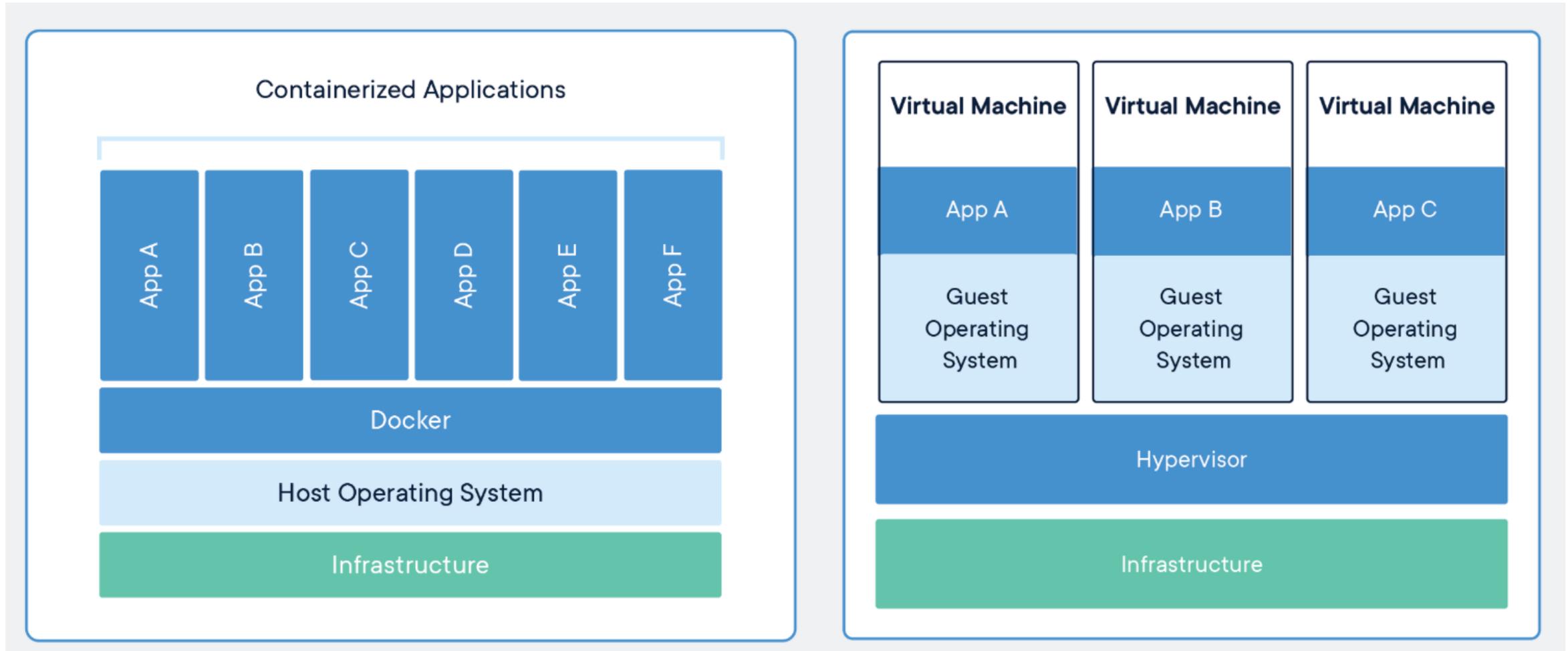


# Docker是什么?

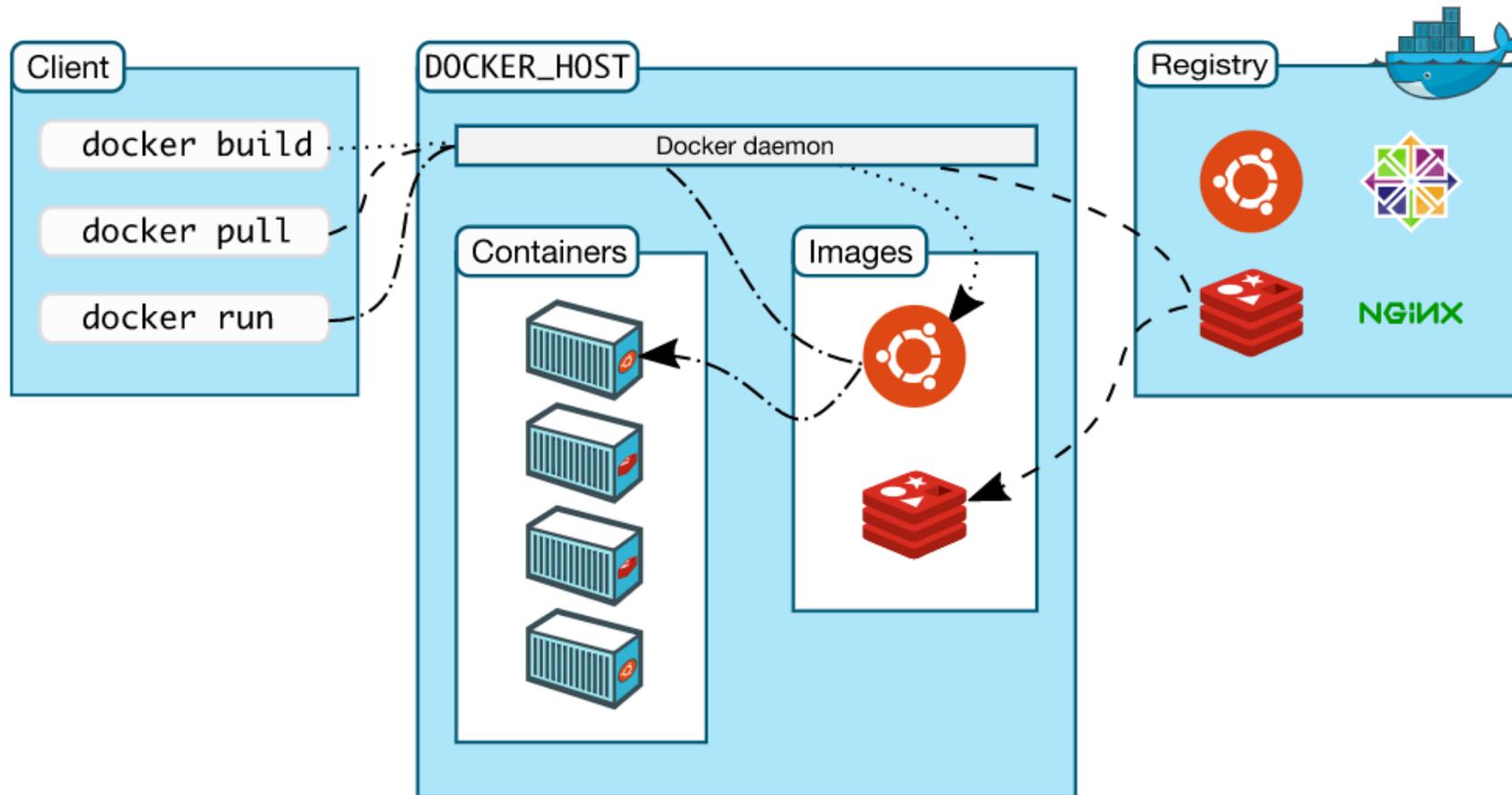
- Docker是基于Go语言实现的开源项目。Docker的主要目标是“Build, Ship and Run Any App, Anywhere”，也就是通过对应用组件的封装、分发、部署、运行等生命周期的管理，使用户的APP（可以是一个WEB应用或数据库应用等等）及其运行环境能够做到“一次封装，到处运行”。



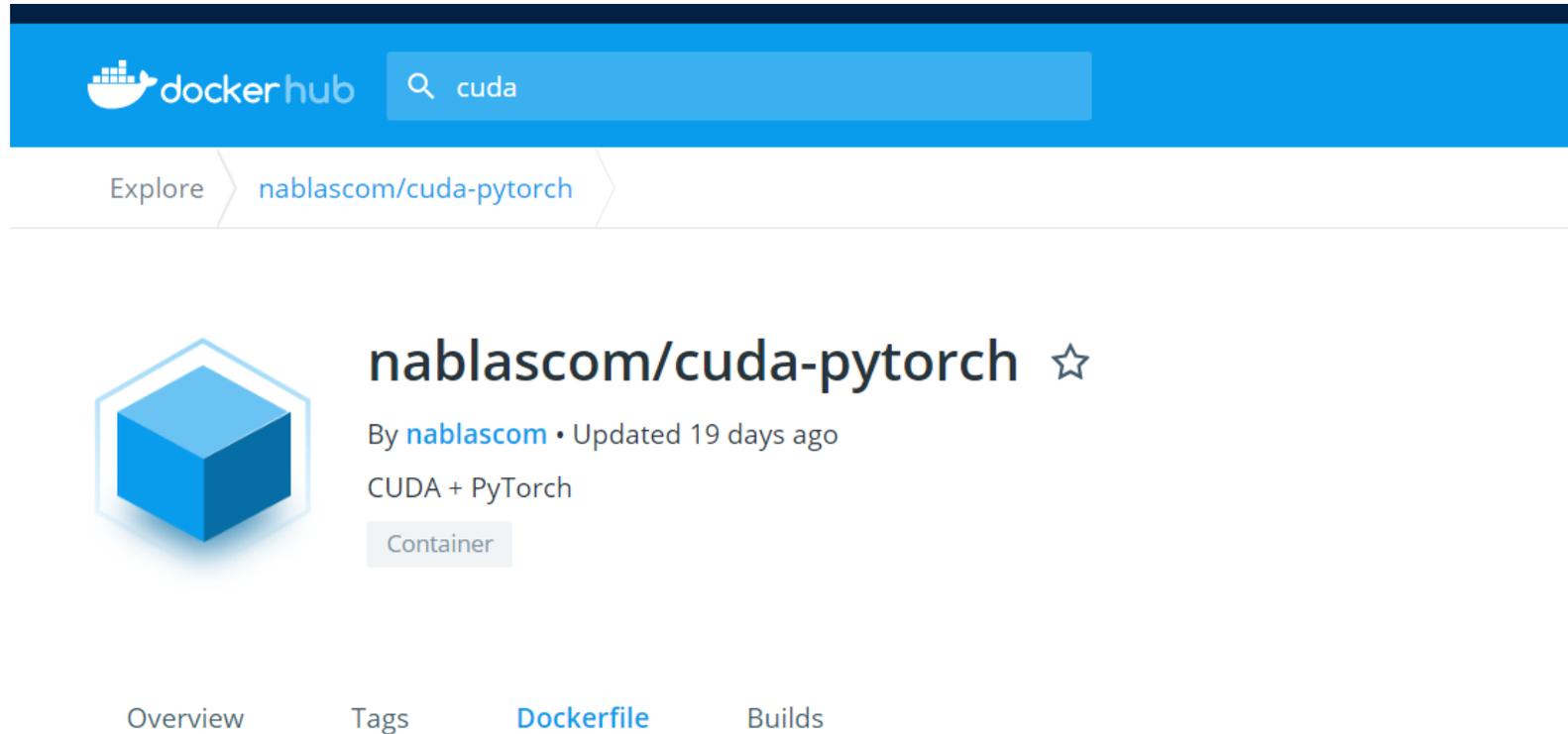
# Docker VS VirtualMachine



# Docker architecture



# Dockerhub ---- docker的github



The image shows a screenshot of the Docker Hub interface. At the top, there is a blue header with the Docker Hub logo and a search bar containing the text 'cuda'. Below the header, a breadcrumb trail shows 'Explore' followed by 'nablascom/cuda-pytorch'. The main content area displays the repository 'nablascom/cuda-pytorch' with a star icon. Below the repository name, it says 'By nablascom • Updated 19 days ago' and 'CUDA + PyTorch'. A 'Container' tag is visible. At the bottom, there are navigation tabs: 'Overview', 'Tags', 'Dockerfile', and 'Builds'.

dockerhub

Explore > nablascom/cuda-pytorch

 **nablascom/cuda-pytorch** ☆

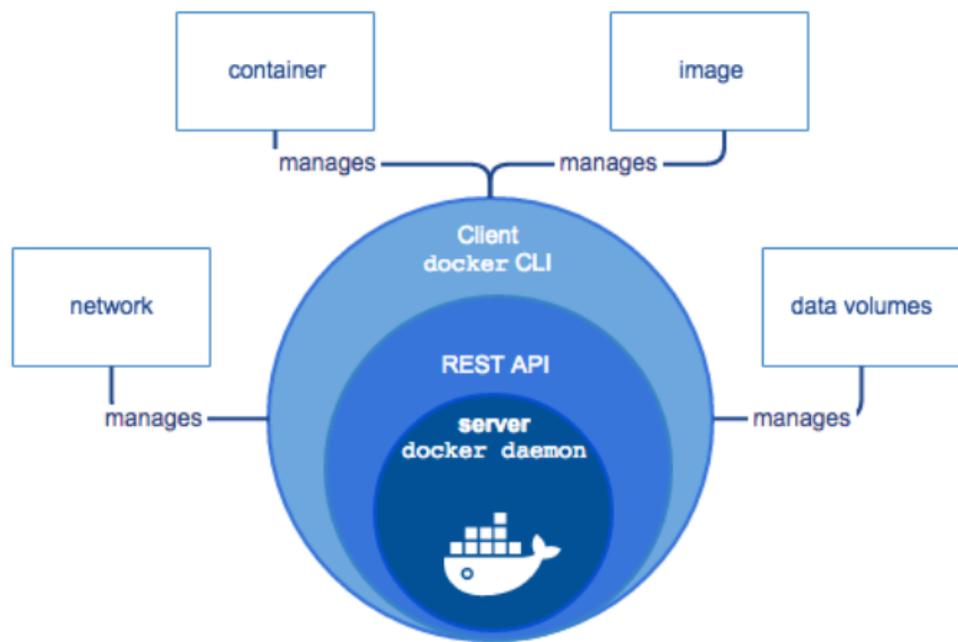
By nablascom • Updated 19 days ago

CUDA + PyTorch

Container

Overview Tags **Dockerfile** Builds

# Docker CLI



- ``docker images``
  - + ``docker images -a``
  - + ``docker images -aq``
- ``docker search 镜像名`` 在dockerhub 中寻找镜像
- ``docker pull 镜像名`` 在镜像源Pull镜像到本地
- ``docker rmi`` 删除镜像
  - \* ``docker rmi -f 镜像ID`` 删除指定镜像
  - \* ``docker rmi -f $(docker images -qa)`` 删除所有镜像
- ``docker run`` 新建并启动容器
  - \* ``docker run -i -t imageid`` 交互式登入容器, 并进入伪终端(-i 和 -t 可以一起写成 -it )
  - \* ``docker run -it --name 新命名 imageid`` 给登入容器取个名字
  - \* ``docker run -d 镜像名`` 后台启动容器
- ``docker ps`` 当前运行的容器
- docker 退出
  - \* ``exit`` 关闭容器并退出
  - \* ``ctrl+P+Q`` 容器不停止退出
- docker 启动容器
  - \* ``docker start 容器ID``
- docker 重启容器
  - \* ``docker restart``
- docker 停止容器
  - \* ``docker stop 容器ID`` 一般停止
  - \* ``docker kill 容器ID`` 强制停止
- docker 删除容器
  - \* ``docker rm 容器ID`` (rmi 是删除镜像,rm是删除容器)
  - \* 删除多容器 ``docker rm -f $(docker ps -aq)``
  - \* 等价于 ``docker ps -aq | xargs docker rm``
- docker 容器信息
  - \* ``docker top 容器ID`` 容器正在运行的进程
  - \* ``docker inspect 容器ID`` 容器内部细节json形式返回
  - \* 与正在运行的容器进行交互
    - ``docker exec -it 容器ID`` 执行的指令` 比如说可以直接在外面命令容器执行指令而不需要登进去,`ls -l`、
    - ``docker attach 容器ID`` 登进正在运行的容器
    - ``docker cp 容器ID:/tmp.yum.log /home`` 将容器中的文件拷到宿主机

# Dockerfile

```
1 FROM ubuntu
2 MAINTAINER https://github.com/chineseocr/chineseocr
3 LABEL version="1.0"
4 EXPOSE 8080
5 RUN apt-get update
6 RUN apt-get install libsm6 libxrender1 libxext-dev gcc -y
7 ##下载Anaconda3 python 环境安装包 放置在chineseocr目录 url地址https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86_64.sh
8 WORKDIR /chineseocr
9 ADD . /chineseocr
10 RUN cd /chineseocr && sh -c '/bin/echo -e "\nyes\n\nyes" | sh Anaconda3-2019.03-Linux-x86_64.sh'
11 RUN echo -e "\ny" | /root/anaconda3/bin/conda install python=3.6
12 RUN /root/anaconda3/bin/pip install easydict opencv-contrib-python==3.4.2.16 Cython h5py pandas requests bs4 matplotlib lxml -U pillow
13 RUN /root/anaconda3/bin/pip install web.py==0.40.dev0
14 RUN echo -e "\ny" | /root/anaconda3/bin/conda install pytorch-cpu torchvision-cpu -c pytorch
15 RUN rm Anaconda3-2019.03-Linux-x86_64.sh
16 #RUN cd /chineseocr/text/detector/utils && sh make-for-cpu.sh
17 #RUN conda clean -p
18 #RUN conda clean -t
```

- Dockerfile

- 基础镜像信息
- 维护者信息
- 镜像操作指令
- 容器启动时的执行命令

- Docker执行Dockerfile的大致流程

- docker从基础镜像运行一个容器
- 执行一条指令并对容器做出修改
- 执行类似docker commit的操作提交一个新的镜像层
- docker再基于刚提交的镜像运行一个新容器
- 执行Dockerfile中的下一条指令直到所有指令都执行完成

# Docker的优势

- 可移植性
  - 环境配置方便，不必考虑依赖问题
  - 传输快，部署快
- 环境独立
  - 每个虚拟容器一个环境
  - 区别于 python virtualenv ，而且像CUDA这种非python库的环境无法用虚拟环境独立。
- 弹性云计算
  - 根据并发需求，弹性改变容器的数量。
- 对于深度学习
  - NVIDIA发布了NVIDIA-docker将显卡虚拟化，极大地支持了深度学习项目的docker部署

# Docker使用实践

- <https://hub.docker.com/r/zergmk2/chineseocr>



zergmk2/chineseocr ☆

By [zergmk2](#) • Updated 10 months ago

docker image for chineseocr project.

Container

Overview

Tags

Dockerfile

Builds

A pre-built docker image for trying chineseocr Environment info: Pytorch 1.0 Python 3.6 chineseocr can be cloned from <https://github.com/chineseocr/chineseocr>

```
PS C:\> docker pull zergmk2/chineseocr:pytorch1.0-cpu-py3.6
pytorch1.0-cpu-py3.6: Pulling from zergmk2/chineseocr
5d9a20cbabf3: Pulling fs layer
0a2b43a72660: Pulling fs layer
18bdd1e546d2: Downloading [=====> ] 474B/514B
8198342c3e05: Pulling fs layer
f56970a44fd4: Pulling fs layer
92b9667eb723: Pulling fs layer
2c3d25f5a487: Pulling fs layer
8597202e617d: Pulling fs layer
b4a8cb3bab4e: Pulling fs layer
d790cc0f4953: Pulling fs layer
e816830f5f8e: Pulling fs layer
da4b2b62f818: Pulling fs layer
bff03blcebb1: Pulling fs layer
81b0dacaf9ab: Pulling fs layer
e502853649e5: Pulling fs layer
6b840ble3116: Waiting
6e0dbeeffca6: Waiting
fed497c698ac: Waiting
```

```
PS C:\> docker run -d -p 8080:8080 zergmk2/chineseocr:pytorch1.0-cpu-py3.6
397798129e0a480c3a3cca41f39d87ff4bb124c8cb6dbccb209bbfba542ea7f9
PS C:\>
```

```
PS C:\> docker run -d -p 8080:8080 zergmk2/chineseocr:pytorch1.0-cpu-py3.6 /bin/bash
281d70caae42fc05cc7f994286752b1d515d4b4765d7f0e168caaf5eb4b1313
PS C:\> docker run -it -p 8080:8080 zergmk2/chineseocr:pytorch1.0-cpu-py3.6 /bin/bash
root@144ee00e43f9:/app# ls
Dockerfile      LICENSE      __pycache__  apphelper    config.py    darknet      model.py     post-demo.py  requirements_cpu.txt  setup.md     templates    test.ipynb   tools
Dockerfile_cpu  README.md   app.py       application  crnn         docker.sh   models       requirements.txt  setup-cpu.md  static      test         text         train
root@144ee00e43f9:/app#
```

```
PS C:\> docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED            STATUS              PORTS                               NAMES
abelb0867e31       zergmk2/chineseocr:pytorch1.0-cpu-py3.6  "ipython app.py 8080"   6 minutes ago     Up 6 minutes       0.0.0.0:8080->8080/tcp, 8888/tcp    happy_bouman
PS C:\> docker logs abelb0867e31
2020-03-24 05:41:29.732320: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
PS C:\> docker inspect abelb0867e31
[
  {
    "Id": "abelb0867e3189f4bf871b0ccb28d04eeb19aa7b2597274c9e524dc3d7901348",
    "Created": "2020-03-24T05:41:17.255371749Z",
    "Path": "ipython",
```

```
PS C:\> docker top abelb0867e31
PID                USER              TIME              COMMAND
2886               root              0:12             {ipython} /usr/local/bin/python /usr/local/bin/ipython app.py 8080
PS C:\>
```

# <http://localhost:8080/ocr>

选择文件 Demo.png

复旦大学 版权所有 © 2010 - 2013. 服务电话: 65643207 65643247 Email: urp@fudan.edu.cn  
Powered by Coremail

耗时:5.5707秒,识别结果为:

序号	值
0	复旦大学版权所有@2010-2013.服务电话:6564320765643247Email:urp@fudan.edu.cn
1	PoweredbyCoremail