Docker 教程

官方参考文档：Docker Guides

Docker 安装

设置仓库

1. 更新apt包索引并安装依赖项

```
$ sudo apt update

$ sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release
```

2. 添加Docker的官方GPG秘钥

```
$ curl -fsSL https://mirror.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu/gpg |
sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

   注：本教程使用的是清华源

3. 设置稳定版仓库

   对于amd64架构的计算机：

```
$ echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

安装Docker

1. 安装Docker Engine-Community

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

2. 测试 Docker 是否安装成功，输入以下指令，打印出以下信息则安装成功:

```
$ sudo docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b8dfde127a29: Pull complete
Digest:
sha256:7d91b69e04a9029b99f3585aaaccae2baa80bcf318f4a5d2165a9898cd2dc0a1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

```
To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent
it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

3. 设置以非root权限使用Docker

```
# 创建名为docker的用户组
$ sudo groupadd docker
# 添加用户到该用户组中
$ sudo usermod -aG docker $USER
# 更新用户组
$ newgrp docker
```

Docker 使用

基本命令

终端中输入`docker`即可查看到Docker的所有命令

```
$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config
                           files (default
                           "/home/antusheng/.docker")
  -c, --context string     Name of the context to use to
                           connect to the daemon
                           (overrides DOCKER_HOST env var
                           and default context set with
                           "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal") (default
"info")
      --tls                Use TLS; implied by --tlsverify
```

```
      --tlscacert string   Trust certs signed only by
                           this CA (default
                           "/home/antusheng/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file
                           (default
                           "/home/antusheng/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default
                           "/home/antusheng/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  app*        Docker App (Docker Inc., v0.9.1-beta3)
  builder     Manage builds
  buildx*     Build with BuildKit (Docker Inc., v0.6.1-docker)
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  scan*       Docker Scan (Docker Inc., v0.8.0)
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running
container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
```

```
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by
default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit
codes

Run 'docker COMMAND --help' for more information on a command.

To get more help with docker, check out our guides at
https://docs.docker.com/go/guides/
```

如果想要更深入的了解特定的docker命令用法， 可以使用`dcoker command --help`命令。

基本使用

1. 获取新镜像

   当我们在本地主机上使用一个不存在的镜像时，Docker会自动下载这个镜像：

   ```
   $ docker run hello-world

   Unable to find image 'hello-world:latest' locally
   latest: Pulling from library/hello-world
   b8dfde127a29: Pull complete
   Digest:
   sha256:7d91b69e04a9029b99f3585aaaccae2baa80bcf318f4a5d2165a9898cd2dc0a1
   Status: Downloaded newer image for hello-world:latest
   ```

   可以看到，镜像hello-world在本地主机上并不存在，而在run之后Docker自动从远程仓库中查找并下载。而如果想要预先下载这个镜像，可以使用如下命令：

   ```
   $ docker pull hello-world:latest

   latest: Pulling from library/hello-world
   b8dfde127a29: Pull complete
   Digest:
   sha256:7d91b69e04a9029b99f3585aaaccae2baa80bcf318f4a5d2165a9898cd2dc0a1
   Status: Downloaded newer image for hello-world:latest
   docker.io/library/hello-world:latest
   ```

   我们可以在Docker Hub搜索镜像，也可以使用`docker search`命令来搜索镜像。

2. 查看镜像列表

```
$ docker images
REPOSITORY    TAG                 IMAGE ID       CREATED         SIZE
ros_airsim    test                7c64051a5da4   14 hours ago    2.86GB
airsim_ros    test                0be40d15eb71   3 days ago      4.21GB
ros           melodic-perception  a49c556e58f4   4 weeks ago     2.32GB
hello-world   latest              d1165f221234   5 months ago    13.3kB
```

3. 启动容器

   以**命令行模式**启动并进入该容器：

```
$ docker run -it --net=host ros:melodic-perception bash
root@4c8dcebe1616:/#
```

   参数说明：

   - **i**: 交互式操作
   - **t**: 终端
   - **--net=host**: 指定容器的网络模式为host模式，即容器共享本地主机的网络命名空间
   - **ros:melodic-perception**: 使用的镜像 image:tag
   - **bash**: 放在镜像后的是容器启动后会运行的命令，这里使用bash使得我们能有一个交互式的Shell

   启动并**后台运行**容器：

```
$ docker run -itdp 32:32 ros:melodic-perception
```

   参数说明：

   - **d**: 后台运行，若想进入容器可使用`docker exec`命令
   - **p**: 端口映射
   - **32:32**: 将本地主机的32端口映射到容器的32端口

   进入容器：

   可使用如下命令进入后台运行的容器：

```
$ docker exec -it 8434e01d3583 bash
root@8434e01d3583:/#
```

4. 查看容器

   查看正在运行的容器：

```
$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED
    STATUS         PORTS                                  NAMES
8434e01d3583   ros:melodic-perception   "/ros_entrypoint.sh …"   3 seconds
ago   Up 2 seconds   0.0.0.0:32->32/tcp, :::32->32/tcp   wonderful_mendel
```

   查看所有的容器：

```
$ docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED
     STATUS                       PORTS                                NAMES
8434e01d3583   ros:melodic-perception   "/ros_entrypoint.sh …"   28 seconds
ago   Up 27 seconds                 0.0.0.0:32->32/tcp, :::32->32/tcp
wonderful_mendel
83cad1bf76f3   ros:melodic-perception   "/ros_entrypoint.sh …"   2 minutes
ago    Exited (0) 2 minutes ago
upbeat_khorana
dbc2afe16620   ros:melodic-perception   "/ros_entrypoint.sh …"   2 minutes
ago    Exited (0) 2 minutes ago
romantic_swartz
3f29abad5c32   ros:melodic-perception   "/ros_entrypoint.sh …"   12 minutes
ago   Exited (0) 12 minutes ago
loving_sanderson
26ee2a8b5e01   ros:melodic-perception   "/ros_entrypoint.sh …"   13 minutes
ago   Exited (0) 13 minutes ago
sad_lichterman
4c8dcebe1616   ros:melodic-perception   "/ros_entrypoint.sh …"   18 minutes
ago   Exited (0) 13 minutes ago
upbeat_moser
e8a422952ee4   ros_airsim:test          "/ros_entrypoint.sh …"   13 hours ago
    Exited (130) 24 minutes ago
amazing_cerf
```

5. 删除容器

   使用`docker stop <the-container-id>`停止容器

```
$ docker stop 8434e01d3583
8434e01d3583
```

   使用`docker rm <the-container-id>`删除容器

```
$ docker rm 8434e01d3583
8434e01d3583
```

   注：可以使用`docker rm -f <the-container-id>`命令同时停止并删除容器

6. 删除镜像

   使用`docker rmi`命令删除镜像：

```
$ docker rmi hello-world:latest
Untagged: hello-world:latest
Untagged: hello-
world@sha256:7d91b69e04a9029b99f3585aaaccae2baa80bcf318f4a5d2165a9898cd2dc0a1
Deleted:
sha256:d1165f2212346b2bab48cb01c1e39ee8ad1be46b87873d9ca7a4e434980a7726
Deleted:
sha256:f22b99068db93900abe17f7f5e09ec775c2826ecfe9db961fea68293744144bd
```

   注：命令`docker rmi`后面可以接镜像名，也可以接镜像ID

7. 创建镜像

   我们可以使用两种方式定制我们需要的镜像：

- 从已有的镜像创建的容器中更新镜像，并提交这个镜像
- 使用Dockerfile创建一个新的镜像

更新镜像

以定制一个装有Eigen库的Ubuntu20.04为例：

```
# 使用基础镜像Ubuntu20.04创建一个容器
$ docker run -it ubuntu:20.04 bash
# 安装Eigen库
root@4c387ecfd9b7:/# apt install libeigen3-dev
root@4c387ecfd9b7:/# exit
# 查看按需求修改的容器ID
$ docker ps -a
CONTAINER ID   IMAGE          COMMAND   CREATED         STATUS
      PORTS     NAMES
4c387ecfd9b7   ubuntu:20.04   "bash"    7 minutes ago   Exited (0) 8 seconds
ago             upbeat_black
# 通过docker commit命令提交容器
# -m表示提交的描述信息， -a指定镜像作者（注意换成自己的用户名）
$ docker commit -m="Eigen installed" -a="antu3heng" 4c387ecfd9b7
antu3heng/ubuntu:20.04withEigen
# 通过docker images命令查看镜像
$ docker images
REPOSITORY             TAG                 IMAGE ID       CREATED
SIZE
antu3heng/ubuntu       20.04withEigen      ce80b05fc3e8   2 minutes ago
204MB
antu3heng/ros_airsim   test                7c64051a5da4   16 hours ago
 2.86GB
airsim_ros             test                0be40d15eb71   4 days ago
 4.21GB
ros                    melodic-perception  a49c556e58f4   4 weeks ago
2.32GB
ubuntu                 20.04               1318b700e415   4 weeks ago
72.8MB
```

使用Dockerfile构建镜像

利用docker build命令使用Dockerfile文件创建一个新的镜像，本教程以竞赛示例程序为例，构建一个包含示例程序及其依赖项的Ubuntu环境。

1. 在解压得到的`modified python`目录下创建名为`Dockerfile`的文件，并包含以下内容：

```
FROM ubuntu:18.04

ENV WORKSPACE=/root/modified_python

RUN apt-get update && apt-get install -y \
    python3-dev python3-pip libgl1-mesa-glx && \
    rm -rf /var/lib/apt/lists/* && \
    pip3 install -U pip && \
    pip3 install numpy scipy tornado==4.5.3 && \
    pip3 install opencv-contrib-python==4.5.3.56 \
    airsim==1.5.0 && \
    mkdir -p $WORKSPACE

COPY ./modified_python/ $WORKSPACE/
```

```
WORKDIR $WORKSPACE

# CMD python3 main.py
CMD ["python3", "main.py"]
```

参数说明：

- **FROM**: 用于定制镜像的基础镜像，这里的基础镜像是ubuntu18.04，后续的操作都是基于ubuntu。
- **ENV**: 设置环境变量，后续的指令可以使用该环境变量。
- **RUN**: 用于执行命令行命令，等同于在终端操作的shell命令。
- **COPY**: 复制指令，用于从Dockerfile文件的上下文目录中复制文件或目录到容器的指定路径里。
- **WORKDIR**: 用于指定工作目录，后续操作都会在这个目录下执行。
- **CMD**: 为启动的容器指定默认要运行的程序。

注：RUN 和 CMD 都有两种写法：

```
RUN/CMD <shell命令>
RUN/CMD ["可执行文件", "参数1", "参数2"] # 等价于 RUN 可执行文件 参数1 参数2
```

2. 在`Dockerfile`文件所在目录下，运行终端，输入`docker build`指令构建镜像：

```
$ docker build -t modified_python .
```

参数说明：

- **t**: 指定要创建的目标镜像名
- modified_python: 目标镜像名
- **.**: Dockerfile文件所在目录，可以指定Dockerfile的绝对路径

3. 查看镜像列表

```
$ docker images
REPOSITORY             TAG                 IMAGE ID       CREATED
      SIZE
modified_python        latest              7bc50e2b5859   13 minutes
ago      968MB
antu3heng/ubuntu       20.04withEigen      ce80b05fc3e8   About an hour
ago   204MB
antu3heng/ros_airsim   test                7c64051a5da4   18 hours ago
       2.86GB
airsim_ros             test                0be40d15eb71   4 days ago
       4.21GB
ros                    melodic-perception  a49c556e58f4   4 weeks ago
      2.32GB
ubuntu                 20.04               1318b700e415   4 weeks ago
      72.8MB
ubuntu                 18.04               39a8cfeef173   4 weeks ago
      63.1MB
```

8. 分享镜像

注册登录Docker Hub，并分享镜像到Docker Hub上。

在终端中输入如下指令将构建的镜像推送到Docker Hub上：

```
# 使用docker login登录Docker Hub
$ docker login -u $username
# 使用docker tag命令给即将要推送的镜像改名
$ docker tag modified_python $username/modified_python
# 使用docker push命令将镜像推送到Docker Hub
$ docker push $username/modified_python
```

注：$username换成自己的Docker Hub用户名