

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

# Docker Api 加密调用与自动化

张岩峰老师微信，加我微信，邀请你加入 VIP 交流答疑群：

微信号：ZhangYanFeng0429

二维码：



## 1、环境准备

### 1.1、什么是 Api

#### (1) API 具体是什么？

API 这个词在维基百科里解释是这样的：应用程序接口（英语：application programming interface，缩写作 API），又称为应用编程接口，就是软件系统不同组成部分衔接的约定。看完这个解释估计你还是有点懵逼，不过没关系，下面我们会用通俗的语言来介绍什么是 API。

我们每个人都有手机，当手机没电了我们肯定会找固定的充电器和充电线来充电。苹果的用苹果，安卓的用安卓。但是你肯定不会用安卓的线去充苹果的手机，这道理很简单，因为你的苹果手机是 Lightning 接口，安卓的是 micro 接口。你要想充电或者对你手机传输数据，那么必须买合适的充电线和数据线，这是对于接口最简单易懂的认识。

类似的，程序的接口也是如此。每个程序都有固定对外的标准接口，这个接口由开发这个程序的开发者定义的，你要想连接它们，那么就应该遵循它们的接

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

口标准。

## (2) 什么是 REST

现在学习 API 经常看到一个词叫 REST, 英文全称是 Representational State Transfer。那么什么是 REST 呢？REST 一词是 Apache 基金会主席 Roy Fielding 博士提出来的，中文意思叫“表现层状态转化”。中文不太好理解，不过我们从下面几个方面去认识你就大概能明白什么是 rest。

### 1、什么是表现层？

这里的表现层指的是资源的表现层，所谓“资源”，就是网络上的一个具体信息。一个文本，一部电影，一个服务都可以算作一资源。那么这些资源用什么来确定和表现呢？那就得用到 URI，比如我们下载一个电影，肯定有对应的 URI 地址，我们看一部网络小说，也有对应的 URI 地址。而且这个地址是唯一的，独一无二的。资源用 URI 标识了，我们可以理解为这个资源已经在网络上“表现”了。所以说到这里，表现层的意思其实就是把“资源”具体呈现出来的形式。

### 2、什么是状态转化？

常识里，我们要把一物体发生状态改变，肯定需要一些操作和手段。网络上的资源也是如此，你下载一部电影，首先得下载，然后才能打开欣赏。下载获取都需要走 HTTP 协议，HTTP 协议里面，四个基本的操作方式：GET、POST、PUT、DELETE（获取，新建，更新，删除）。通过这基本的四种方法可以对网络上的资源进行一些状态转化操作。

所以，REST 就是表现层的状态转化，大家分开理解上面两点然后结合在一起就明白了。简单粗暴的可以理解为：方法 + URI 资源。

GET /movie/war/珍珠港

DELETE /movie/war/珍珠港

## 1.2、Docker Api 种类

docker 的 api 也遵循 rest 的风格，因此我们了解了上面两点后，我们开始学习 docker 本身 api 的相关知识。

首先，我们把 docker 当作一种资源，我们可以通过 api 来对 docker 进行操作，操作的方法也是 http 的那几种方法。

其次，我们要了解 docker 有哪些对外可使用的 api，这里 docker 官方主要有三大对外 api。

- Docker Registry API
- Docker Hub API
- Docker Remote API

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

## 1、Docker Registry API

这个是 docker 镜像仓库的 api，通过操作这套 API，你可以自由的自动化、程序化的管理你的镜像仓库。

## 2、Docker Hub API

Docker Hub API 是用户管理操作的 API，docker hub 是使用校验和公共 namespaces 的方式来存储账户信息、认证账户、进行账户授权。API 同时也允许操作相关的用户仓库和 library 仓库。

## 3、Docker Remote API

这套 API 用于控制主机 Docker 服务端的 API，等价于 docker 命令行客户端。有了它，你能远程操作 docker 容器，更重要的是你可以通过程序自动化运维 docker 进程。

### 1.3、API 使用前准备

前面我们说过，操作 rest api 用的就是 http 的那些方法。那么具体怎么使用这些方法呢？这里我们提供几种通用的方式来操作调用下 docker 的 API，然后体验下。在体验之前，我们需要开启 docker rest api，不然没开启，你是不能用的。具体开启的方法：

```
[root@localhost ~]# vi /usr/lib/systemd/system/docker.service
添加如下：
-H tcp://0.0.0.0:8088
注意：8088 端口自定义即可。

[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target docker.socket firewalld.service containerd.service
Wants=network-online.target
Requires=docker.socket containerd.service

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock -H tcp://0.0.0.0:8088
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229
```

重启 docker 服务

```
[root@localhost ~]# systemctl daemon-reload
[root@localhost ~]# systemctl restart docker
[root@localhost ~]# netstat -tlunp | grep 8088

[root@localhost ~]# netstat -tlunp | grep 8088
tcp6      0      0 :::8088          :::*              LISTEN      19951/dockerd
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

重启完成后，我们执行 `curl 127.0.0.1:8088/info | python -mjson.tool` 命令即可查看 docker 的状态状态（json 形式，python -mjson.tool 借用了这个工具，这样让 json 格式化，好阅读）

```
[root@localhost ~]# curl 127.0.0.1:8088/info | python -mjson.tool
```

```
[root@localhost ~]# curl 127.0.0.1:8088/info | python -mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %         0      0     0   354k      0 --:--:-- --:--:-- --:--:-- 398k
{
  "Architecture": "x86_64",
  "BridgeNfIp6tables": true,
  "BridgeNfIptables": true,
  "CPUSet": true,
  "CPUShares": true,
  "CgroupDriver": "systemd",
  "CgroupVersion": "1",
  "ContainerdCommit": {
    "Expected": "10c12954828e7c7c9b6e0ea9b0c02b01407d3ae1",
    "ID": "10c12954828e7c7c9b6e0ea9b0c02b01407d3ae1"
  },
  "Containers": 0,
  "ContainersPaused": 0,
  "ContainersRunning": 0,
  "ContainersStopped": 0,
```

启用了 docker API 后，我们还有个问题，那就是在哪查询 docker 现有的 API？既然 docker 提供了那 3 大 API 库：Docker Registry API、Docker Hub API、Docker Remote API。那么在哪里可以查看具体详细的 API，比如 Docker Registry API 下面到底有哪些 API 地址？有查询镜像的 API 吗？有删除的吗？其实这些都有，我们可以直接去官网 API 手册里查看即可，地址就是：

<https://docs.docker.com/engine/api/v1.38/>（想看什么版本的把最后的 v1.38 替换成目标版本号即可）

这里要注意的是，官方不再建议使用 API v1.12 之前的版本，建议使用 v1.24 或更高的版本。

查看本地 docker API 版本可以用 `docker version` 命令：

```
[root@localhost ~]# docker version
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
[root@localhost ~]# docker version
Client: Docker Engine - Community
 Version:      20.10.17
 API version:  1.41
 Go version:   go1.17.11
 Git commit:   100c701
 Built:        Mon Jun  6 23:05:12 2022
 OS/Arch:     linux/amd64
 Context:      default
 Experimental: true

Server: Docker Engine - Community
 Engine:
  Version:      20.10.17
  API version:  1.41 (minimum version 1.12)
  Go version:   go1.17.11
  Git commit:   a89b842
  Built:        Mon Jun  6 23:03:33 2022
  OS/Arch:     linux/amd64
  Experimental: false
 containerd:
```

## 2、Docker Api 操作镜像

### 2.1、查看所有 Docker 镜像的详细信息

docker 命令查看镜像：

[root@localhost ~]# docker images				
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	605c77e624dd	6 months ago	141MB
centos	latest	5d0da3dc9764	9 months ago	231MB

客户端通过 api 接口请求获取镜像信息：

```
[root@localhost ~]# curl -X GET http://192.168.43.11:8088/images/json | python -mjson.tool
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
[root@localhost ~]# curl -X GET http://127.0.0.1:8088/images/json | python -mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100    912    100    912      0      0   633k      0 --:--:-- --:--:-- --:--:--   890k
[
  {
    "Created": 1640806109,
    "Id": "sha256:605c77e624ddb75e6110f997c58876baa13f8754486b461117934b24a9dc3a85",
    "Labels": {
      "maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"
    },
    "ParentId": "",
    "RepoDigests": [
      "nginx@sha256:0d17b565c37bcbd895e9d92315a05c1c3c9a29f762b011a10c54a66cd53c9b31"
    ],
    "RepoTags": [
      "nginx:latest"
    ],
    "SharedSize": -1,
    "Size": 141479488,
    "VirtualSize": 141479488
  },
  {
    "Created": 1631730005,
    "Id": "sha256:5d0da3dc976460b72c77d94c8a1ad043720b0416bfc16c52c45d4847e53fadb6",
    "Labels": {
  }
```

## 2.2、搜索 Docker 镜像

### ● 语法说明

```
语法：
GET /images/search

参数：
term: 要搜索的镜像名称，例如 nginx:latest
```

客户端通过 api 接口请求搜索 docker 镜像：

示例：搜索 nginx 镜像

```
[root@localhost ~]# curl -v -X GET http://192.168.1.11:8088/images/search?term=nginx | python -mjson.tool
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
[root@localhost ~]# curl -v -X GET http://192.168.1.11:8088/images/search?term=nginx | python -mjson.tool
* About to connect() to 192.168.1.11 port 8088 (#0)
*   Trying 192.168.1.11...
  % Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left     Speed
  0     0    0     0    0     0      0      0      0     0      0      0     0
> GET /images/search?term=nginx HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.168.1.11:8088
> Accept: */*
>
< HTTP/1.1 200 OK
< Api-Version: 1.41
< Content-Type: application/json
< Docker-Experimental: false
< Ostype: linux
< Server: Docker/20.10.17 (linux)
< Date: Mon, 04 Jul 2022 00:41:25 GMT
< Transfer-Encoding: chunked
<
{ [data not shown]
100 3433  0 3433  0     0 3536    0 --:--:-- --:--:-- --:--:-- 3535
* Connection #0 to host 192.168.1.11 left intact
[
  {
    "description": "Official build of Nginx.",
    "is_automated": false,
    "is_official": true,
    "name": "nginx",
    "star_count": 17036
  },
  {
    "description": "Bitnami nginx Docker Image"
```

## 2.3、Pull Docker 镜像

语法:

POST /images/create

参数:

fromImage: 要下载的镜像名称, 例如 nginx

tag: 要下载的镜像版本, 例如 latest 最新版

客户端通过 api 接口请求下载 docker 镜像:

### 示例：下载 nginx:latest 镜像

```
[root@localhost ~]# curl -X POST -d 'fromImage=nginx' -d 'tag=latest' http://192.168.128.11:8088/images/create
```

## 2.4、删除 Docker 镜像

语法:

DELETE /images/(name)

客户端通过 api 接口请求删除 docker 镜像:

### 示例：删除 nginx:latest 镜像

```
[root@localhost ~]# docker images
REPOSITORY      TAG          IMAGE ID      CREATED      SIZE
nginx           latest      55f4b40fe486  10 days ago  142MB

[root@localhost ~]# curl -v -X DELETE http://192.168.1.11:8088/images/nginx:latest
```

版权声明，本文档全部内容版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
[root@localhost ~]# docker images
REPOSITORY    TAG        IMAGE ID      CREATED      SIZE
```

### 3、Docker Api 操作容器

#### 3.1、创建 Docker 容器

语法：POST /containers/create

Hostname：容器内的主机名  
Domainname：容器内的域名  
User：容器内运行的用户  
Image：指定启动容器的镜像  
Env：配置容器内的环境变量

HostConfig 选项：

将容器的 80 端口映射到宿主机的 8080 端口

```
    "PortBindings": {
      "80/tcp": [{"HostPort": "8080"}]
    },
```

限制容器内存最大为 512MB：

```
    "Memory": 512000000,
```

限制 CPU 最大为 0.1 核：

```
    "CpusetCpus": "0,1",
```

限制最多可打开 1024 个文件，如果超过 hard 就需要 root 权限：

```
    "Ulimits": [
      {
        "Name": "nofile", "Soft": 1024, "Hard": 2048
      }
    ]
```

容器重启策略，always 表示当容器停止后，会一直启动此容器：

```
    "RestartPolicy": {
      "Name": "always"
    },
```

示例：

```
curl -X POST -H "Content-Type: application/json" -d '{
  "Image": "nginx:latest",
  "Env": ["nginx=latest"],
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
    "ExposedPorts": {
      "80/tcp": {}
    },
    "HostConfig": {
      "PortBindings": {
        "80/tcp": [{"HostPort": "8080"}]
      },
      "Memory": 512000000,
      "CpusetCpus": "0,1",
      "RestartPolicy": {
        "Name": "always"
      },
      "Ulimits": [
        {"Name": "nofile", "Soft": 1024, "Hard": 2048}
      ]
    }
  }
}' http://192.168.43.11:8088/containers/create

返回值:
{"Id":"4fc5ab994cdbe496acd6e5263fd88c05267d1095dc37c9d32afff8ab60d8602e","Warnings":[]}
```

这里的 ID 就是这个容器的 ID 值，等下可以通过这个 ID 启动这个容器。

### 3.2、启动 Docker 容器

语法:

POST /containers/{id}/start

根据上面的容器 ID 启动这个容器。

```
[root@localhost ~]# curl -X POST -H "Content-Type: application/json"
http://192.168.43.11:8088/containers/4fc5ab994cdbe496acd6e5263fd88c05
267d1095dc37c9d32afff8ab60d8602e/start

[root@localhost ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4fc5ab994cd	nginx:latest	"/docker-entrypoint..."	18 minutes ago	Up 10 minutes	0.0.0.0:8080->80/tcp, :::8080->80/tcp	amazing_dhawan

### 3.3、停止 Docker 容器

语法:

POST /containers/{id}/stop

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

根据容器 ID 停止这个容器

```
[root@localhost ~]# curl -X POST -H "Content-Type: application/json"
http://192.168.43.11:8088/containers/4fc5ab994cdbe496acd6e5263fd88c05
267d1095dc37c9d32afff8ab60d8602e/stop

[root@localhost ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4fc5ab994cd	nginx:latest	"/docker-entrypoint..."	31 minutes ago	Exited (0) 1 second ago		amazing_dhawan

### 3.4、删除已停止的 Docker 容器

语法：

```
DELETE /containers/{id}
```

删除已停止的 Docker 容器

```
[root@localhost ~]# curl -X DELETE
http://192.168.43.11:8088/containers/4fc5ab994cdbe496acd6e5263fd88c05
267d1095dc37c9d32afff8ab60d8602e
```

### 3.5、查看所有 Docker 容器

语法：

```
GET /containers/json
```

查看节点所有容器：

```
[root@localhost ~]# curl -X GET
http://192.168.43.11:8088/containers/json | python -mjson.tool
```

## 4、Docker 客户端与服务端 TLS 认证

通过前面学会了如何连接到 Docker Remote API。从安全的角度上看，这存在一点儿安全问题。如果不做安全方面的限制，那么任何人都可以连接到你的服务器上通过 API 接口创建高权限容器，从而拿到你服务器权限。

Docker 的 0.9 版本开始 Docker Remote API 开始提供了认证机制。这种认证机制采用了 TLS/SSL 证书来确保用户与 API 之间连接的安全。

### 4.1、创建证书授权中心

下面创建所需 CA 证书和密钥的方法，在大多数平台下这个过程都是非常标准的。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

(1) 确保系统安装了 openssl

```
[root@localhost ~]# which openssl
/usr/bin/openssl
```

(2) 创建证书存放目录

```
[root@localhost ~]# mkdir /etc/docker/certs
```

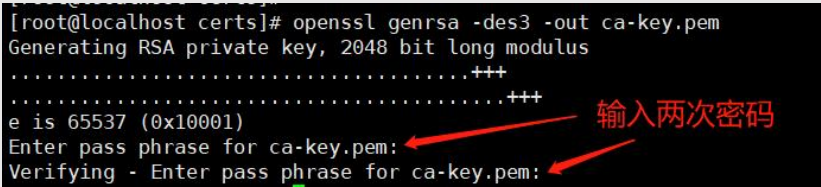
(3) 生成私钥，在创建私钥的过程中，为 CA 秘钥设置一个密码（要自己记住这个密码，在新 CA 中，我们需要用这个秘钥来创建并对证书签名）

```
[root@localhost certs]# echo 01 | sudo tee ca.srl
01

# 生成私钥
[root@localhost certs]# openssl genrsa -des3 -out ca-key.pem

[root@localhost certs]# openssl genrsa -des3 -out ca-key.pem
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for ca-key.pem:
Verifying - Enter pass phrase for ca-key.pem:

[root@localhost certs]# ls
ca-key.pem  ca.srl
```



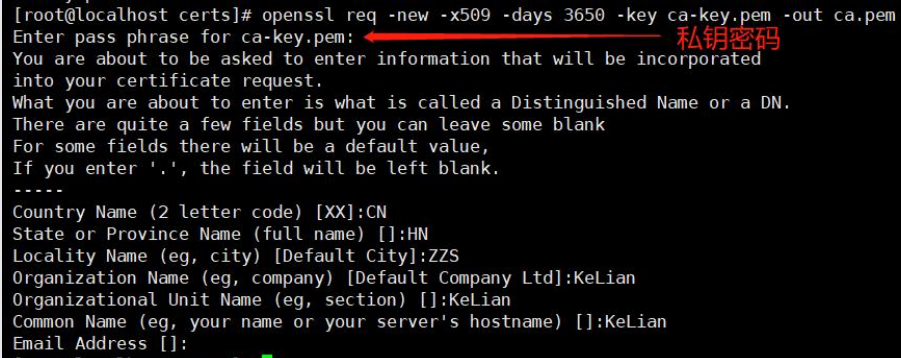
(4) 创建一个 CA 证书

国家=“CN”  
省=“HN”  
市/县=“ZZS”  
组织=“KeLian”  
组织单位=“KeLian”

# 创建 CA 证书

```
[root@localhost certs]# openssl req -new -x509 -days 3650 -key ca-key.pem -out ca.pem
```

```
[root@localhost certs]# openssl req -new -x509 -days 3650 -key ca-key.pem -out ca.pem
Enter pass phrase for ca-key.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:HN
Locality Name (eg, city) [Default City]:ZZS
Organization Name (eg, company) [Default Company Ltd]:KeLian
Organizational Unit Name (eg, section) []:KeLian
Common Name (eg, your name or your server's hostname) []:KeLian
Email Address []:
```



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
# 查看生成的证书文件
[root@localhost certs]# ls
ca-key.pem  ca.pem  ca.srl
现在有了 CA，就可以用它为 Docker 服务器创建证书和密钥。
```

## 4.2、创建服务器的证书签名请求和密钥

用 CA 来为 Docker 服务器进行证书签名请求和密钥的签名和验证。

(1) 为 Docker 服务器创建一个密钥，同理也需要输入这个密钥的密码（注意：请设置一个密码，我们将会在使用之前清除这个密码（见下面第 4 步）。用户只需要在后面的几步中使用密码）。

```
[root@localhost certs]# openssl genrsa -des3 -out server-key.pem
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for server-key.pem:
Verifying - Enter pass phrase for server-key.pem:

[root@localhost certs]# ls
ca-key.pem  ca.pem  ca.srl  server-key.pem
```

(2) 创建服务器的证书签名请求（CSR）

国家=“CN”  
省=“HN”  
市/县=“ZZS”  
组织=“KeLian”  
组织单位=“KeLian”

Common Name 比较重要，这个选项的值要么为 Docker 服务器（即从 DNS 中解析后得到的结果，比如 docker.example.com）的 FQDN（fully qualified domain name，完全限定域名）形式，要么为\*（这将允许在任何服务器上使用该服务器证书）。

```
[root@localhost certs]# openssl req -new -key server-key.pem -out
server.csr
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
[root@localhost certs]# openssl req -new -key server-key.pem -out server.csr
Enter pass phrase for server-key.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:HN
Locality Name (eg, city) [Default City]:ZZS
Organization Name (eg, company) [Default Company Ltd]:KeLian
Organizational Unit Name (eg, section) []:KeLian
Common Name (eg, your name or your server's hostname) []:* 
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

```
[root@localhost certs]# ls
ca-key.pem  ca.pem  ca.srl  server.csr  server-key.pem
```

### (3) 生成服务器证书

```
[root@localhost certs]# openssl x509 -req -days 3650 -in server.csr
-CA ca.pem -CAkey ca-key.pem -out server-cert.pem
[root@localhost certs]# openssl x509 -req -days 3650 -in server.csr -CA ca.pem -CAkey ca-key.pem -out
server-cert.pem
Signature ok
subject=C=CN/ST=HN/L=ZZS/O=KeLian/OU=KeLian/CN=*
Getting CA Private Key
Enter pass phrase for ca-key.pem: 
```

上面命令输入完成之后会生成一个名为 server-cert.pem 的文件，这个文件就是我们的服务器证书。

```
[root@localhost certs]# ls
ca-key.pem  ca.pem  ca.srl  server-cert.pem  server.csr
server-key.pem
```

### (4) 清除服务器密钥的密码

不想在 Docker 守护进程启动的时候再输入一次密码，因此需要清除它。

```
[root@localhost certs]# openssl rsa -in server-key.pem -out
server-key.pem
[root@localhost certs]# openssl rsa -in server-key.pem -out server-key.pem
Enter pass phrase for server-key.pem: 
writing RSA key
```

### (5) 移动服务器证书文件并授权（授权对证书进行保护）

```
[root@localhost certs]# mkdir /etc/docker/server
[root@localhost certs]# mv server* /etc/docker/server/
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
# 对目录授权
[root@localhost server]# chmod 0600 /etc/docker/server/
[root@localhost server]# ls -ld /etc/docker/server/ -l
drw-----. 2 root root 69 7月  4 14:26 /etc/docker/server/

# 对文件授权
[root@localhost server]# chmod 0600 /etc/docker/server/*
[root@localhost server]# ls -l /etc/docker/server/
总用量 12
-rw-----. 1 root root 1151 7月  4 14:20 server-cert.pem
-rw-----. 1 root root  980 7月  4 14:12 server.csr
-rw-----. 1 root root 1675 7月  4 14:24 server-key.pem
```

### 4.3、配置 Docker 守护进程

#### (1) 修改 docker.service 文件

```
[root@localhost ~]# vi /usr/lib/systemd/system/docker.service
在 ExecStart 后添加如下：
--tlsverify --tlscacert=/etc/docker/certs/ca.pem
--tlscert=/etc/docker/server/server-cert.pem
--tlskey=/etc/docker/server/server-key.pem

[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target docker.socket firewalld.service containerd.service
Wants=network-online.target
Requires=docker.socket containerd.service

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock -H tcp://0.0.0.0:8088 --tlsverify --tlscacert=/etc/docker/certs/ca.pem --tlscert=/etc/docker/server/server-cert.pem --tlskey=/etc/docker/server/server-key.pem
ExecReload=/bin/kill -s HUP $MAINPID
```

--tlscacert 指定 CA 证书位置  
--tlscert 指定服务器证书位置  
--tlskey 指定服务器密钥位置

#### (2) 重启 Docker 服务

```
[root@localhost ~]# systemctl daemon-reload
[root@localhost ~]# systemctl restart docker
[root@localhost ~]# systemctl status docker

[root@localhost ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since 2022-07-04 14:35:20 CST; 8s ago
     Docs: https://docs.docker.com
    Main PID: 11506 (dockerd)
      Tasks: 8
     Memory: 34.2M
    CGroup: /system.slice/docker.service
            └─11506 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock -H tcp://0.0.0.0:8088 --tlsverify --tlscacert=/etc/docker/certs/ca.pem --tlscert=/etc/docker/se...

7月 04 14:35:20 localhost.localdomain dockerd[11506]: time="2022-07-04T14:35:20.446382722+08:00" level=info msg="ClientConn switching balancer to 'pick_first'" module=grpc
7月 04 14:35:20 localhost.localdomain dockerd[11506]: time="2022-07-04T14:35:20.463884973+08:00" level=info msg="[graphdriver] using prior storage driver: overlay2"
7月 04 14:35:20 localhost.localdomain dockerd[11506]: time="2022-07-04T14:35:20.469269563+08:00" level=info msg="Loading containers: start."
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

#### 4.4、创建客户端证书和密钥

服务器已经启用了 TLS，下面我们需要为客户端创建和签名证书和密钥，以保证我们的 Docker 客户端的安全性。

##### (1) 客户端通过 API 访问服务器端测试

```
[root@localhost ~]# curl -X GET http://192.168.1.11:8088/images/json
[root@localhost ~]# curl -X GET http://192.168.1.11:8088/images/json
Client sent an HTTP request to an HTTPS server.

[root@localhost ~]# curl -X GET https://192.168.1.11:8088/images/json
curl: (60) Peer's Certificate issuer is not recognized.
More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
of Certificate Authority (CA) public keys (CA certs). If the default
bundle file isn't adequate, you can specify an alternate file
using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
the bundle, the certificate verification probably failed due to a
problem with the certificate (it might be expired, or the name might
not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
the -k (or --insecure) option.
```

可以看出，必须携带客户端证书访问，如果没有携带客户端证书，就无法请求。

##### (2) 创建客户端的密钥，同理，也需要输入一个临时的密码

```
[root@localhost ~]# cd /etc/docker/certs/
[root@localhost certs]# openssl genrsa -des3 -out client-key.pem

[root@localhost ~]# cd /etc/docker/certs/
[root@localhost certs]# openssl genrsa -des3 -out client-key.pem
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for client-key.pem:
Verifying - Enter pass phrase for client-key.pem:

[root@localhost certs]# ls
ca-key.pem  ca.pem  ca.srl  client-key.pem
```

##### (3) 创建客户端 CSR 文件


国家=“CN”

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

省=“HN”  
市/县=“ZZS”  
组织=“KeLian”  
组织单位=“KeLian”

```
[root@localhost certs]# openssl req -new -key client-key.pem -out client.csr

[root@localhost certs]# openssl req -new -key client-key.pem -out client.csr
Enter pass phrase for client-key.pem:  密码验证
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:HN
Locality Name (eg, city) [Default City]:ZZS
Organization Name (eg, company) [Default Company Ltd]:KeLian
Organizational Unit Name (eg, section) []:KeLian
Common Name (eg, your name or your server's hostname) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

[root@localhost certs]# ls
ca-key.pem  ca.pem  ca.srl  client.csr  client-key.pem
```

(4) 添加一些扩展的客户端 SSL 认证属性，来开启密钥的客户端身份认证

```
[root@localhost certs]# echo extendedKeyUsage = clientAuth > extfile.cnf

[root@localhost certs]# cat extfile.cnf
extendedKeyUsage = clientAuth
```

(5) 在 CA 中对客户端 CSR 进行签名

此处我们使用 CA 密钥的密码创建另一个证书：client-cert.pem。

需要输入创建 ca-key.pem 文件时的密码

```
[root@localhost certs]# openssl x509 -req -days 3650 -in client.csr -CA ca.pem -CAkey ca-key.pem -out client-cert.pem -extfile extfile.cnf
Signature ok
subject=/C=CN/ST=HN/L=ZZS/O=KeLian/OU=KeLian
Getting CA Private Key
Enter pass phrase for ca-key.pem: <==输入创建 ca-key.pem 文件时的密码
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
# 得到 client-cert.pem 证书文件
[root@localhost certs]# ls -l
总用量 28
-rw-r--r--. 1 root root 1751 7月 4 11:04 ca-key.pem
-rw-r--r--. 1 root root 1285 7月 4 11:12 ca.pem
-rw-r--r--. 1 root root 3 7月 4 15:42 ca.srl
-rw-r--r--. 1 root root 1176 7月 4 15:42 client-cert.pem
-rw-r--r--. 1 root root 964 7月 4 15:21 client.csr
-rw-r--r--. 1 root root 1743 7月 4 15:13 client-key.pem
-rw-r--r--. 1 root root 30 7月 4 15:40 extfile.cnf
```

(6) 同理，我们需要清除 client-cert.pem 文件中的密码，以便在 Docker 客户端中使用该文件，如下所示：回车之后需要输入上面创建 client-key.pem 密钥文件指定的密码。

```
[root@localhost certs]# openssl rsa -in client-key.pem -out
client-key.pem
[root@localhost certs]# openssl rsa -in client-key.pem -out client-key.pem
Enter pass phrase for client-key.pem:
writing RSA key
```

## 4.5、客户端通过认证访问

### (1) 推送客户端证书

将 ca.pem、client-cert.pem 和 client-key.pem 这 3 个文件复制到想运行 Docker 客户端的宿主机上。

```
# 客户端创建路径
[root@localhost ~]# mkdir /root/docker-client

# 服务器端推送证书到客户端
[root@localhost certs]# scp ca.pem client-cert.pem client-key.pem
root@192.168.1.21:/root/docker-client
```

### (2) 测试访问

```
[root@localhost ~]# cd /root/docker-client/
[root@localhost docker-client]# curl
https://192.168.1.11:8088/images/json --cert ./client-cert.pem
--key ./client-key.pem -k | python -mjson.tool
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
[root@localhost docker-client]# curl https://192.168.1.11:8088/images/json --cert ./client-cert.pem --key ./client-key.pem -k | python -mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100    391    100    391    0     0    5498      0  --:--:-- --:--:-- --:--:--   5585
{
  "Containers": -1,
  "Created": 165957604,
  "Id": "sha256:55f4b40fe486a5b734b46bb7bf28f52fa31426bf23be068c9e7b19e58d9b8deb",
  "Labels": {
    "maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"
  },
  "ParentId": "",
  "RepoDigests": [
    "nginx@sha256:10f14ffa93f8dedf1057897b745e5ac72ac5655c299dade0aa434c71557697ea"
  ],
  "RepoTags": [
    "nginx:latest"
  ],
  "SharedSize": -1,
  "Size": 141531591,
  "VirtualSize": 141531591
}
```

## 5、Python 调用 Docker Api

### 5.1、安装第三方库

对于原生 Docker Remote API 而言，虽然功能很强大，但是使用起来并不是十分的方便。

为了简化 Docker Remote API 的使用，Python 针对 Docker API 进行了封装，提供了一套完整的 Python 第三方库：docker，专门用于操作调用 Docker 服务。

接下来，我们将会详细详解如何使用 Python 的第三方库来调用 Docker。

安装 docker 库：

```
C:\Users\ 岩 峰 >pip install docker -i
https://pypi.tuna.tsinghua.edu.cn/simple
```

Python 官方手册：<https://docker-py.readthedocs.io/en/stable/index.html>

Docker 官方手册：<https://docs.docker.com/engine/api/sdk/examples/>

### 5.2、镜像的基本操作

#### 5.2.1、查看 Docker 镜像

Python 调用 docker api 查看 docker 有哪些镜像

示例：

源码如下：

```
import docker
# 导入证书
tls_config = docker.tls.TLSConfig(
    client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
# 获取 docker 镜像信息赋值给变量 images
images = client.images.list()
# 循环打印 docker 镜像的信息
print("镜像\t\t\t\t\t版本\t\t\t\t\tID")
for img in images:
    # 对镜像信息做过滤，过滤出想要的信息
    print(img.attrs["RepoTags"][0] + ':' + img.attrs["Id"][7:19])
```

执行结果如下：

```
镜像      版本      ID
nginx:latest:55f4b40fe486
httpd:latest:ac826143758d
centos:7:eeb6ee3f44bd
```

代码说明：

tls\_config: 包含了客户端的密钥信息

```
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
```

说明：配置 Docker 服务端的基本信息，包含了 base\_url（Docker 服务端的地址）以及 version。version 设置为"auto" 可以自动检查 Docker API 的版本。

具体过滤方法：

```
[root@localhost docker-client]# curl https://192.168.1.11:8088/images/json --cert ./client-cert.pem --key ./client-key.pem -k | python -mjson.tool
% Total % Received % Xferd Average Speed Time Time Time
Dload Upload Total Spent Left Speed
100 1444 100 1444 0 0 18065 0 --:--:-- --:--:-- --:--:-- 18278
{
  {
    "Containers": -1,
    "Created": 1658957604,
    "Id": "sha256:55f4b40fe486a5b734b46bb7bf28f52fa31426bf23be068c8e7b19e58d9b8deb",
    "Labels": {
      "maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"
    },
    "ParentId": "",
    "RepoDigests": [
      "nginx@sha256:10f14ffa93f8dedf1057897b745e5ac72ac5655c299dade0aa434c71557697ea"
    ],
    "RepoTags": [
      "nginx:latest"
    ],
    "SharedSize": -1,
    "Size": 141531591,
    "VirtualSize": 141531591
  },
  {
    "Containers": -1,
```

## 5.2.2、下载 Docker 镜像

Python 调用 docker api 下载镜像

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

示例：

源码如下：

```
import docker
# 导入证书
tls_config = docker.tls.TLSConfig(
    client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
# 下载镜像
image_name = input("请输入要下载镜像的名称：")
image_tag = input("请输入要下载镜像的版本：")
image = client.images.pull(repository=image_name, tag=image_tag)
print(image, '已下载')
```

运行结果如下：

```
请输入要下载镜像的名称：nginx
请输入要下载镜像的版本：latest
<Image: 'nginx:latest'> 已下载
```

### 5.2.3、删除 Docker 镜像

Python 调用 docker api 删除镜像

示例：

源码如下：

```
import docker
# 导入证书
tls_config = docker.tls.TLSConfig(
    client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
#打印镜像信息
images = client.images.list()
print("镜像\t\t 版本\t\tID")
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
for img in images:
    print(img.attrs["RepoTags"][0] + ':' + img.attrs["Id"][7:19])
#删除镜像
images_id = input("请输入要删除镜像的 ID 值: ")
image = client.images.remove(images_id)
print(image, '已删除')
```

执行结果如下：

```
镜像      版本      ID
nginx:latest:605c77e624dd
请输入要删除镜像的 ID 值: 605c77e624dd
None 已删除
```

## 5.3、容器的基本操作

### 5.3.1、创建 Docker 容器

Python 调用 docker api 创建容器

Python 官方手册：

<https://docker-py.readthedocs.io/en/stable/containers.html>

示例：

源码如下：

```
import docker
# 导入证书
tls_config = docker.tls.TLSConfig(
    client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
# 获取 docker 镜像信息赋值给变量 images
images = client.images.list()
# 循环打印 docker 镜像的信息
print("镜像\t\t版本\t\tID")
for img in images:
    # 对镜像信息做过滤，过滤出想要的信息
    print(img.attrs["RepoTags"][0] + ':' + img.attrs["Id"][7:19])
# 创建容器
container = client.containers.run(
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
name="nginx_web", #container 的名称
hostname="nginx_web", #容器内的主机名
image="nginx:latest", #要运行的镜像
mem_limit="512m", #内存限制
cpu_count=1, #CPU 限制
memswap_limit="-1", #关闭 SWAP
ports={'80/tcp':8080}, #容器的 80 映射到宿主机的 8080
detach=True #将容器放到后台运行，等同于-d
)
#打印创建容器的 ID 值
print(str(container))
```

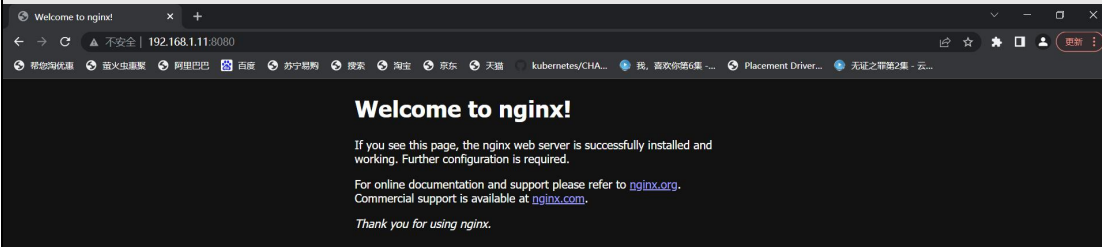
运行结果如下：

```
镜像      版本      ID
nginx:latest:605c77e624dd
<Container: f8cd89d255>
```

```
[root@docker01 ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f8cd89d255f8	nginx:latest	"/docker-entrypoint..."	5 seconds ago	Up 4 seconds	0.0.0.0:8080->80/tcp, :::8080->80/tcp	nginx_web

访问测试：



端口随机映射：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
create_pod.py x
16 # 创建容器
17 container = client.containers.run(
18     name="nginx_web", #container的名称
19     hostname="nginx_web", #容器内的主机名
20     image="nginx:latest", #要运行的镜像
21     mem_limit="512m", #内存限制
22     cpu_count=1, #CPU限制
23     memswap_limit="-1", #关闭SWAP
24     ports={'80/tcp':''}, #容器的80端口随机映射到宿主机
25     detach=True #将容器放到后台运行，等同于-d
26 )
27 #打印创建容器的ID值
28 print(str(container))
```

### 5.3.2、查看 Docker 容器

Python 调用 docker api 查看容器

示例：

源码如下：

```
import docker
# 导入证书
tls_config = docker.tls.TLSConfig(
    client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
# 获取 docker 镜像信息赋值给变量 images
images = client.images.list()
# 循环打印 docker 镜像的信息
print("所有容器的信息为：")
# 获取已运行容器的信息
container = client.containers.list(all) #默认只打印运行的容器，all
表示所有容器
for i in container:
    print(i.name + ":" + i.id[7:19] + ":" + i.status)
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

运行结果如下：

所有容器的信息为：

nginx\_web:0d291630ed8b:running

### 5.3.3、停止 Docker 容器

Python 调用 docker api 删除容器

示例：

源码如下：

```
import docker
# 导入证书
tls_config = docker.tls.TLSConfig(
    client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
# 获取 docker 镜像信息赋值给变量 images
images = client.images.list()
# 循环打印 docker 镜像的信息
print("所有容器的信息为：")
# 获取所有容器的信息
container = client.containers.list(all)
for i in container:
    print(i.name + ":" + i.id[7:19] + ":" + i.status)
#停止容器
Stop_Container = input("请输入要停止容器的名字或 ID 值：")
for i in client.containers.list(Stop_Container):
    i.stop()
    print(Stop_Container + "已停止 ☺")
```

运行结果如下：

所有容器的信息为：

nginx\_web:0d291630ed8b:running

请输入要停止容器的名字或 ID 值：nginx\_web

nginx\_web 已停止 ☺

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

### 5.3.4、启动 Docker 容器

Python 调用 docker api 启动容器

示例：

源码如下：

```
import docker
# 导入证书
tls_config = docker.tls.TLSConfig(
    client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
# 获取 docker 镜像信息赋值给变量 images
images = client.images.list()
# 循环打印 docker 镜像的信息
print("所有容器的信息为：")
# 获取所有容器的信息
container = client.containers.list(all)
for i in container:
    print(i.name + ":" + i.id[7:19] + ":" + i.status)
# 运行容器
Start_Container = input("请输入要启用容器的名字或 ID 值：")
for i in client.containers.list(Start_Container):
    i.start()
    print(Start_Container + "已启动☺")
```

运行结果如下：

```
所有容器的信息为：
nginx_web:0d291630ed8b:running
请输入要启用容器的名字或 ID 值：nginx_web
nginx_web 已启动☺
```

### 5.3.5、删除 Docker 容器

Python 调用 docker api 删除容器

注意：已经运行的容器必须先停止再删除

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

示例：

源码如下：

```
import docker
# 导入证书
tls_config = docker.tls.TLSConfig(
    client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")
# 获取 docker 镜像信息赋值给变量 images
images = client.images.list()
# 循环打印 docker 镜像的信息
print("所有容器的信息为：")
# 获取所有容器的信息
container = client.containers.list(all)
for i in container:
    print(i.name + ":" + i.id[7:19] + ":" + i.status)
# 删除容器
Del_Container = input("请输入要删除容器的名字或 ID 值：")
for i in client.containers.list(Del_Container):
    i.stop()
    i.remove()
    print(Del_Container + "已删除 ☺")
```

执行结果如下：

```
所有容器的信息为：
nginx_web:0d291630ed8b:running
请输入要删除容器的名字或 ID 值：nginx_web
nginx_web 已删除 ☺
```

## 5.4、Docker 管理系统开发

v1 版本：

源码如下：

```
import docker
import time

# 导入证书
tls_config = docker.tls.TLSConfig(
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
client_cert=(r'F:\docker-client\client-cert.pem',
r'F:\docker-client\client-key.pem'),
    verify=False
)
# 配置服务器基本信息
client = docker.DockerClient(base_url='https://192.168.1.11:8088',
tls=tls_config, version="auto")

# 查询 Docker 镜像
def List_Images():
    # 获取 docker 镜像信息赋值给变量 images
    images = client.images.list()
    # 循环打印 docker 镜像的信息
    print("镜像\t版本\tID")
    for img in images:
        # 对镜像信息做过滤，过滤出想要的信息
        print(img.attrs["RepoTags"][0] + ' ' + img.attrs["Id"][7:19])
        time.sleep(1)

# 下载 Docker 镜像
def Download_Images():
    # 下载镜像
    image_name = str(input("请输入要下载镜像的名称: "))
    if image_name == "":
        print("请您重新选择要操作的选项并输入正确的镜像名称")
        return True
    image_tag = str(input("请输入要下载镜像的版本[不输入默认下载latest 版本]: "))
    if image_tag == "":
        image_tag = "latest"
    try:
        image = client.images.pull(repository=image_name,
tag=image_tag)
    except:
        print("下载超时，请重试")
    print(image, '已下载')
    time.sleep(1)

# 删除 Docker 镜像
def Delete_Images():
    images_id = input("请输入要删除镜像的名称或 ID 值: ")
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
try:
    image = client.images.remove(images_id)
except:
    print(images_id, "不存在，请核实！")
    return True
try:
    list = client.images.get(images_id)
except:
    print(images_id, '已删除，请确认！')
time.sleep(1)

# 创建容器
def Create_Container():
    name = str(input("请输入要创建容器的名称："))
    hostname = str(input("请输入容器内的主机名："))
    image = str(input("请输入要使用的镜像[例如格式为:nginx:latest]:"))
    mem_limit = str(input("请输入对容器内存的限制[例如格式为:512m]:"))
    cpu_count = int(input("请输入对容器 CPU 的限制[例如格式为: 1]:"))
    while True:
        ports_key = str(input("请输入要容器要映射的端口[例如格式为:80/tcp]:"))
        ports_value = str(input("请输入映射到宿主机的端口[例如格式为: 80, 不写随机映射]:"))
        ports[ports_key] = ports_value
        con = str(input("请问您是否还要继续映射端口[YES || NO]:"))
        if con == "NO" or con == "no":
            break
    # 创建容器
    try:
        container = client.containers.run(
            name=name, # container 的名称
            hostname=hostname, # 容器内的主机名
            image=image, # 要运行的镜像
            mem_limit=mem_limit, # 内存限制
            cpu_count=cpu_count, # CPU 限制
            memswap_limit="-1", # 关闭 SWAP
            ports=ports, # 容器的 80 映射到宿主机的 8080
            detach=True # 将容器放到后台运行，等同于-d
        )
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
# 打印创建容器的 ID 值
print("容器创建成功，容器 ID 为：" + str(container))
except:
    print("创建失败，请检查是否配置正确！")
time.sleep(1)

# 查询 Docker 容器
def List_Container():
    # 循环打印 docker 镜像的信息
    print("所有容器的信息为：")
    # 获取已运行容器的信息
    container = client.containers.list(all) # 默认只打印运行的容器，all 表示所有容器
    for i in container:
        print(i.name + ":" + i.id[0:10] + ":" + i.status)
    time.sleep(1)

# 停止容器
def Stop_Container():
    stop_container = str(input("请输入要停止容器的名字或 ID 值："))
    try:
        i = client.containers.get(stop_container)
        i.stop()
        print(stop_container + "已停止")
    except:
        print("停止容器失败了~，请检查您输入的信息是否正确~")
    time.sleep(1)

# 启动容器
def Start_Container():
    start_container = str(input("请输入要启动容器的名字或 ID 值："))
    try:
        i = client.containers.get(start_container)
        i.start()
        print(start_container + "已启动")
    except:
        print("启动容器失败了~，请检查您输入的信息是否正确~")
    time.sleep(1)

# 重启容器
def Restart_Container():
    restart_container = str(input("请输入要重启容器的名字或 ID 值："))
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
)  
    try:  
        i = client.containers.get(restart_container)  
        i.restart()  
        print(restart_container + "已启动")  
    except:  
        print("重启容器失败了~，请检查您输入的信息是否正确~")  
        time.sleep(1)  
  
# 删除容器  
def Delete_Container():  
    delete_container = input("请输入要删除容器的名字或 ID 值：")  
    try:  
        for i in client.containers.list(delete_container):  
            i.stop()  
            i.remove()  
            print(delete_container + "已删除")  
    except:  
        print("删除容器失败了~，请检查您输入的信息是否正确~")  
        time.sleep(1)  
  
# 变量  
ports = {}  
  
# Welcome Docker Manager  
def Welcome_Docker_Manager():  
    print("【欢迎使用 Docker 管理系统】")  
    print("1、查询 Docker 镜像")  
    print("2、下载 Docker 镜像")  
    print("3、删除 Docker 镜像")  
    print("4、创建 Docker 容器")  
    print("5、查询 Docker 容器")  
    print("6、停止 Docker 容器")  
    print("7、启动 Docker 容器")  
    print("8、重启 Docker 容器")  
    print("9、删除 Docker 容器")  
    print("10、重新打印帮助信息")  
    print("0、退出本程序")  
  
if __name__ == '__main__':  
    Welcome_Docker_Manager()  
    while True:
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
num = str(input("请输入您要执行的动作[0-10]☆："))
if num == "0":
    break
elif num == "1":
    List_Images()
elif num == "2":
    Download_Images()
elif num == "3":
    Delete_Images()
elif num == "4":
    Create_Container()
elif num == "5":
    List_Container()
elif num == "6":
    Stop_Container()
elif num == "7":
    Start_Container()
elif num == "8":
    Restart_Container()
elif num == "9":
    Delete_Container()
elif num == "10":
    Welcome_Docker_Manager()
else:
    print("输入错误")
```

运行结果如下：

镜像操作演示：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
F:\docker-client>python Docker_Manager.py
【欢迎使用Docker管理系统】
1、查询Docker镜像
2、下载Docker镜像
3、删除Docker镜像
4、创建Docker容器
5、查询Docker容器
6、停止Docker容器
7、启动Docker容器
8、重启Docker容器
9、删除Docker容器
10、重新打印帮助信息
0、退出本程序
请输入您要执行的动作[0-10]☆: 1
镜像    版本    ID
nginx:latest:605c77e624dd
请输入您要执行的动作[0-10]☆: 2
请输入要下载镜像的名称: tomcat
请输入要下载镜像的版本[不输入默认下载latest版本]:
<Image: 'tomcat:latest'> 已下载
请输入您要执行的动作[0-10]☆: 1
镜像    版本    ID
nginx:latest:605c77e624dd
tomcat:latest:fb5657adc892
请输入您要执行的动作[0-10]☆: 3
请输入要删除镜像的名称或ID值: tomcat
tomcat 已删除, 请确认!
请输入您要执行的动作[0-10]☆: 1
镜像    版本    ID
nginx:latest:605c77e624dd
请输入您要执行的动作[0-10]☆: 0
```

创建容器操作演示:

```
F:\docker-client>python Docker_Manager.py
【欢迎使用Docker管理系统】
1、查询Docker镜像
2、下载Docker镜像
3、删除Docker镜像
4、创建Docker容器
5、查询Docker容器
6、停止Docker容器
7、启动Docker容器
8、重启Docker容器
9、删除Docker容器
10、重新打印帮助信息
0、退出本程序
请输入您要执行的动作[0-10]☆: 4
请输入要创建容器的名称: nginx
请输入容器内的主机名: nginx.web.com
请输入要使用的镜像[例如格式为: nginx:latest]: nginx:latest
请输入对容器内存的限制[例如格式为: 512m]: 1024m
请输入对容器CPU的限制[例如格式为: 1]: 2
请输入要容器要映射的端口[例如格式为: 80/tcp]: 80/tcp
请输入映射到宿主机的端口[例如格式为: 80, 不写随机映射]: 80
请问您是否还要继续映射端口[YES|NO]: yes
请输入要容器要映射的端口[例如格式为: 80/tcp]: 8080/tcp
请输入映射到宿主机的端口[例如格式为: 80, 不写随机映射]: 8080
请问您是否还要继续映射端口[YES|NO]: no
容器创建成功, 容器ID为: <Container: f8dee5333b>
请输入您要执行的动作[0-10]☆: 5
所有容器的信息为:
nginx:f8dee5333b:running
```

停止容器演示:

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
请输入您要执行的动作[0-10]☆: 5
所有容器的信息为:
nginx:f8dee5333b:running
请输入您要执行的动作[0-10]☆: 6
请输入要停止容器的名字或ID值: nginx
nginx已停止
请输入您要执行的动作[0-10]☆: 5
所有容器的信息为:
nginx:f8dee5333b:exited
```

启动容器演示:

```
请输入您要执行的动作[0-10]☆: 5
所有容器的信息为:
nginx:f8dee5333b:exited
请输入您要执行的动作[0-10]☆: 7
请输入要启动容器的名字或ID值: nginx
nginx已启动
请输入您要执行的动作[0-10]☆: 5
所有容器的信息为:
nginx:f8dee5333b:running
```

删除容器演示:

```
请输入您要执行的动作[0-10]☆: 5
所有容器的信息为:
nginx:f8dee5333b:running
请输入您要执行的动作[0-10]☆: 9
请输入要删除容器的名字或ID值: nginx
nginx已删除
请输入您要执行的动作[0-10]☆: 5
所有容器的信息为:
请输入您要执行的动作[0-10]☆:
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**