




文章 (/blogs) > 大舒的博客 (/blog/qiukeke) > 文章详情 (/write)

Flux7 Docker 系列教程（七）： Docker API (/a/1190000002711455)

 **大舒** (/u/qiukeren) 3.2k 5月22日 发布

推荐

2 推荐

收藏

2 收藏，347 浏览

本系列教程翻译自 Flux7 Docker Tutorial Series，系列共有九篇，本文译自第七篇 Part 7: Ultimate Guide for Docker APIs (<http://blog.flux7.com/blogs/docker/docker-tutorial-series-part-7-ultimate-guide-for-docker-apis>)。该系列所有文章将参考其他学习资料翻译，也会加入自己的学习作为部分注解。如有错误，欢迎指正。

在以前的系列教程中，我们已经探讨了 Docker 中很多很重要的组件，本篇文章我们深入 Docker：探讨 Docker API。

值得注意的是，Docker 为了方便使用，提供了如下四种 API：

- 1. Docker Registry API
- 2. Docker Hub API
- 3. Docker OAuth API
- 4. Docker Remote API

本篇文章专门用来探讨 Docker Registry API、Docker Hub API 和 OAuth API。

Docker Registry API

Docker Registry API 为了简化镜像和仓库的存储而设计的 REST API。这些 API 并不涉及用户账户和用户认证。

取出镜像层：

```
GET /v1/images/(image_id)/layer
```

```

root@server2 ~ # curl -o test-image-layer -v --raw http://localhost:5000/v1/images/c5881f11ded97fd2252adf93268114329e985624c5d7bb86e439a36109d1124e/layer
* About to connect() to localhost port 5000 (#0)
* Trying 127.0.0.1... % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
           Dload  Upload  Total  Spent  Left  Speed
  0   0   0   0   0   0   0  --:--:-- --:--:-- --:--:--    0connected
> GET /v1/images/c5881f11ded97fd2252adf93268114329e985624c5d7bb86e439a36109d1124e/layer HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: localhost:5000
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: gunicorn/18.0
< Date: Tue, 24 Jun 2014 11:54:49 GMT
< Connection: keep-alive
< Content-Length: 18121987
< Accept-Ranges: bytes
< Expires: Wed, 24 Jun 2015 11:54:49 GMT
< Last-Modified: Thu, 01 Jan 1970 00:00:00 GMT
< Cache-Control: public, max-age=31536000
< Content-Type: application/octet-stream
< X-Docker-Registry-Version: 0.7.3
< X-Docker-Registry-Config: dev
<
{ [data not shown]
100 17.2M 100 17.2M  0   0 527M   0 --:--:-- --:--:-- --:--:-- 540M
* Connection #0 to host localhost left intact
* Closing connection #0
root@server2 ~ # ls -ltrh test-image-layer
-rw-r--r-- 1 root root 18M Jun 24 13:54 test-image-layer
root@server2 ~ #

```

插入镜像层：

PUT /v1/images/(image\_id)/layer

```

root@server2 ~ # curl -v --raw -X PUT http://localhost:5000/v1/images/c5881f11ded97fd2252adf93268114329e985624c5d7bb86e439a36109d1124e/layer --data-binary test-image-layer
* About to connect() to localhost port 5000 (#0)
* Trying 127.0.0.1... connected
> PUT /v1/images/c5881f11ded97fd2252adf93268114329e985624c5d7bb86e439a36109d1124e/layer HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: localhost:5000
> Accept: */*
> Content-Length: 16
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 16out of 16 bytes
< HTTP/1.1 409 CONFLICT
< Server: gunicorn/18.0
< Date: Tue, 24 Jun 2014 12:16:21 GMT
< Connection: keep-alive
< Expires: -1
< Content-Type: application/json
< Pragma: no-cache
< Cache-Control: no-cache
< Content-Length: 33
< X-Docker-Registry-Version: 0.7.3
< X-Docker-Registry-Config: dev
<
{"error": "Image already exists"}
root@server2 ~ #

```

检索镜像：

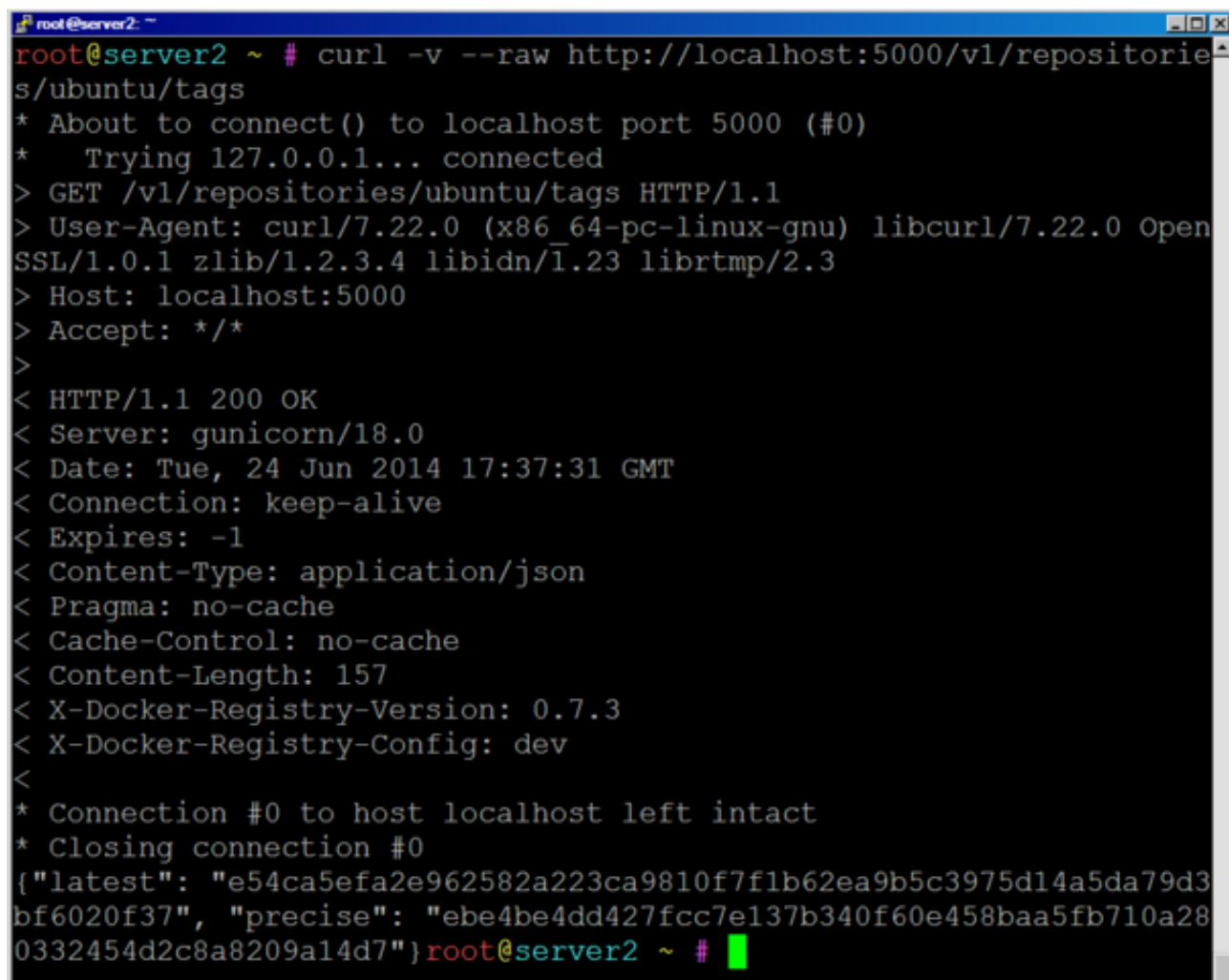
GET /v1/images/(image\_id)/json

检索镜像的根镜像：

```
GET /v1/images/(image_id)/ancestry
```

获取指定库的所有标签：

```
GET /v1/repositories/(namespace)/(repository)/tags
```



```
root@server2 ~ # curl -v --raw http://localhost:5000/v1/repositories/ubuntu/tags
* About to connect() to localhost port 5000 (#0)
*   Trying 127.0.0.1... connected
> GET /v1/repositories/ubuntu/tags HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: localhost:5000
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: gunicorn/18.0
< Date: Tue, 24 Jun 2014 17:37:31 GMT
< Connection: keep-alive
< Expires: -1
< Content-Type: application/json
< Pragma: no-cache
< Cache-Control: no-cache
< Content-Length: 157
< X-Docker-Registry-Version: 0.7.3
< X-Docker-Registry-Config: dev
<
* Connection #0 to host localhost left intact
* Closing connection #0
{"latest": "e54ca5efa2e962582a223ca9810f7f1b62ea9b5c3975d14a5da79d3bf6020f37", "precise": "ebe4be4dd427fcc7e137b340f60e458baa5fb710a280332454d2c8a8209a14d7"}root@server2 ~ #
```

获取指定库的指定标签：

```
GET /v1/repositories/(namespace)/(repository)/tags/(tag*)
```

删除标签：

```
DELETE /v1/repositories/(namespace)/(repository)/tags/(tag*)
```

```

root@server2 ~ # curl -v --raw -X DELETE http://localhost:5000/v1/repositories/ubuntu/latest
* About to connect() to localhost port 5000 (#0)
*   Trying 127.0.0.1... connected
> DELETE /v1/repositories/ubuntu/latest HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: localhost:5000
> Accept: */*
>
< HTTP/1.1 301 MOVED PERMANENTLY
< Server: gunicorn/18.0
< Date: Tue, 24 Jun 2014 13:58:11 GMT
< Connection: keep-alive
< Content-Type: text/html; charset=utf-8
< Content-Length: 311
< Location: http://localhost:5000/v1/repositories/ubuntu/latest/
< X-Docker-Registry-Version: 0.7.3
< X-Docker-Registry-Config: dev
<
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>Redirecting...</title>
<h1>Redirecting...</h1>
* Connection #0 to host localhost left intact
* Closing connection #0
<p>You should be redirected automatically to target URL: <a href="http://localhost:5000/v1/repositories/ubuntu/latest/">http://localhost:5000/v1/repositories/ubuntu/latest/</a>. If not click the link.root@server2 ~ #

```

registry 状态检查：

GET /v1/\_ping

```

root@server2 ~ # curl -v --raw http://localhost:5000/v1/_ping
* About to connect() to localhost port 5000 (#0)
*   Trying 127.0.0.1... connected
> GET /v1/_ping HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: localhost:5000
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: gunicorn/18.0
< Date: Tue, 24 Jun 2014 17:38:59 GMT
< Connection: keep-alive
< X-Docker-Registry-Standalone: True
< Expires: -1
< Content-Type: application/json
< Pragma: no-cache
< Cache-Control: no-cache
< Content-Length: 4
< X-Docker-Registry-Version: 0.7.3
< X-Docker-Registry-Config: dev
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
root@server2 ~ #

```

## Docker Hub API

Docker Hub API 是为 Docker Hub 设计的 REST API。Docker Hub（也就是 Index）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。

## 特殊的仓库的操作

### 创建新的仓库

```
PUT /v1/repositories/(repo_name)/
```

### 删除现有仓库

```
DELETE /v1/repositories/(repo_name)/
```

### 更新仓库镜像

```
PUT /v1/repositories/(repo_name)/images
```

### 获取仓库镜像

```
GET /v1/repositories/(repo_name)/images
```

### 认证

```
PUT /v1/repositories/(repo_name)/auth
```

以上都是特殊仓库的操作，下面是对普通用户开放的 API。

特殊的 library 仓库和用户仓库的区别在于 library 仓库是官方仓库，可以直接使用 ubuntu 这种名字做 repo 的名字，而译者就只能使用 qiuker521/ubuntu 作为 repo 的名字。

## 普通用户的操作

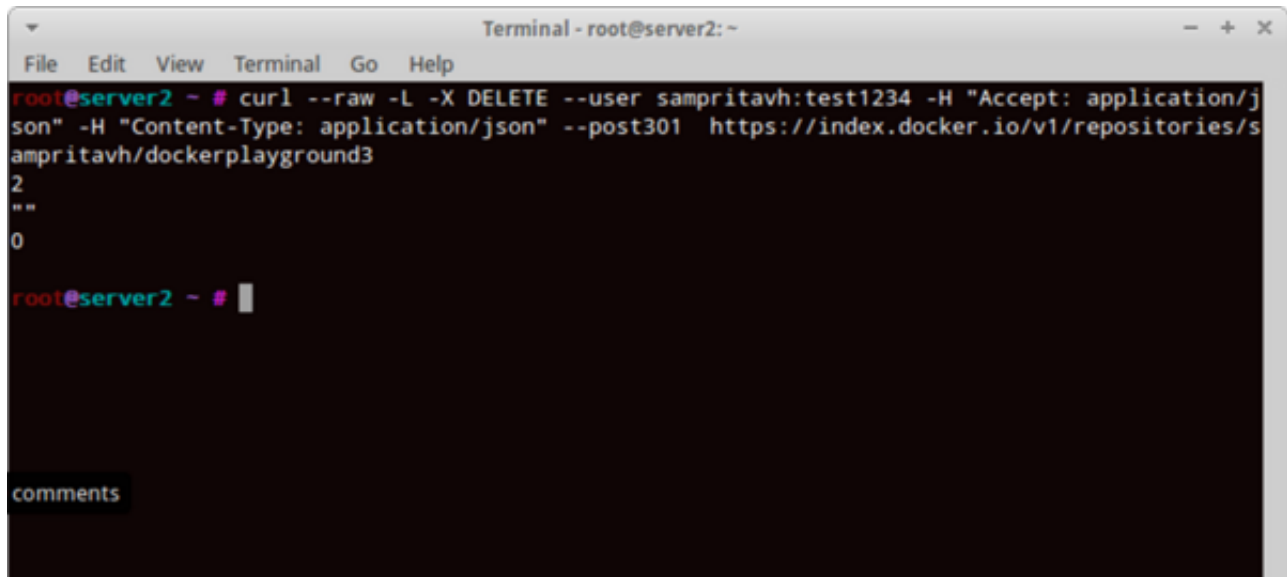
### 创建新的用户仓库

```
PUT /v1/repositories/(namespace)/(repo_name)/
```

### 删除现有仓库

```
DELETE /v1/repositories/(namespace)/(repo_name)/
```



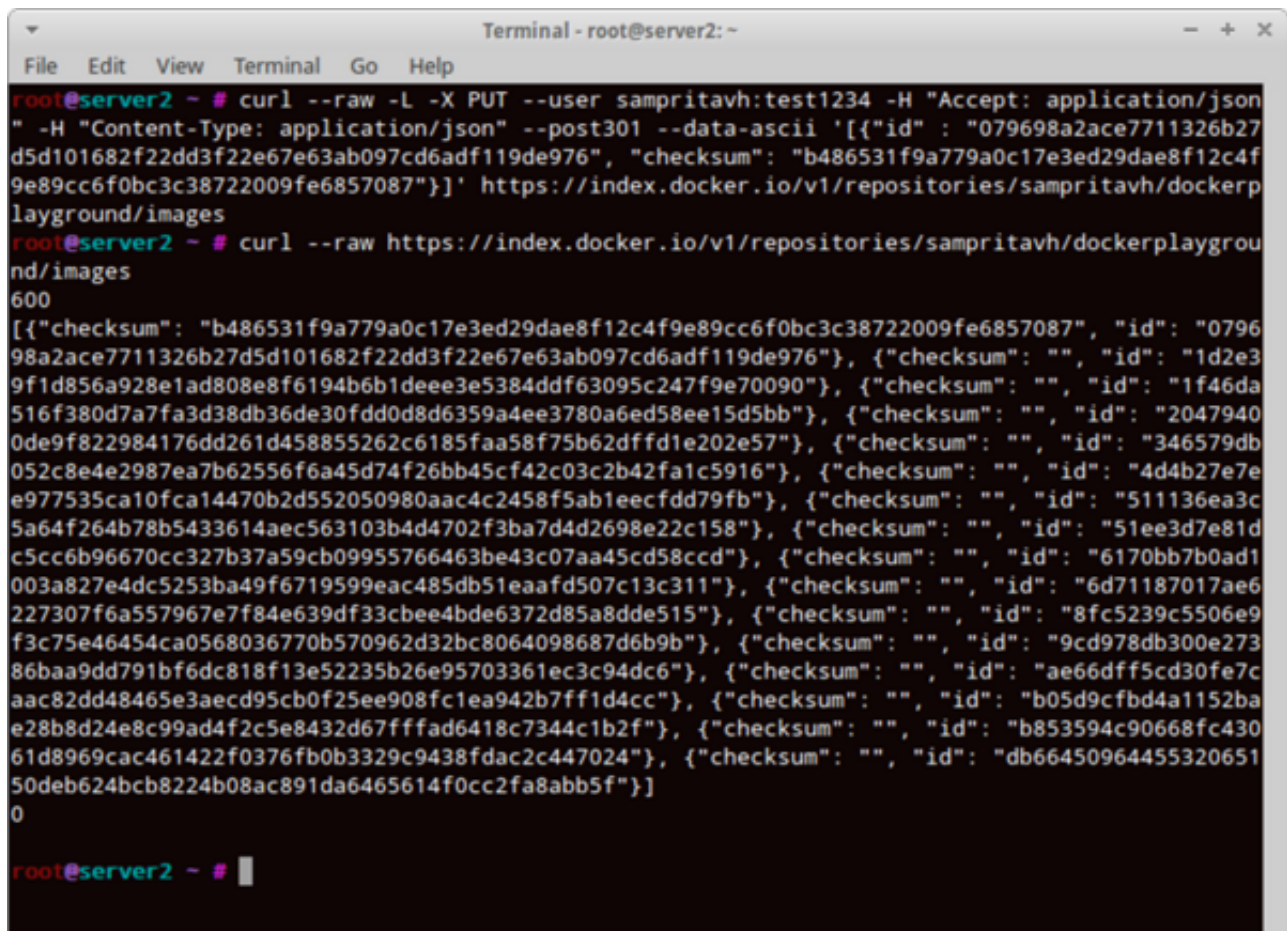


```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl --raw -L -X DELETE --user sampritavh:test1234 -H "Accept: application/json" -H "Content-Type: application/json" --post301 https://index.docker.io/v1/repositories/sampritavh/dockerplayground3
2
""
0
root@server2 ~ #
```

comments

## 更新镜像

PUT /v1/repositories/(namespace)/(repo\_name)/images



```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl --raw -L -X PUT --user sampritavh:test1234 -H "Accept: application/json" -H "Content-Type: application/json" --post301 --data-ascii '[{"id": "079698a2ace7711326b27d5d101682f22dd3f22e67e63ab097cd6adf119de976", "checksum": "b486531f9a779a0c17e3ed29dae8f12c4f9e89cc6f0bc3c38722009fe6857087"}]' https://index.docker.io/v1/repositories/sampritavh/dockerplayground/images
root@server2 ~ # curl --raw https://index.docker.io/v1/repositories/sampritavh/dockerplayground/images
600
[{"checksum": "b486531f9a779a0c17e3ed29dae8f12c4f9e89cc6f0bc3c38722009fe6857087", "id": "079698a2ace7711326b27d5d101682f22dd3f22e67e63ab097cd6adf119de976"}, {"checksum": "", "id": "1d2e39f1d856a928e1ad808e8f6194b6b1deee3e5384ddf63095c247f9e70090"}, {"checksum": "", "id": "1f46da516f380d7a7fa3d38db36de30fdd0d8d6359a4ee3780a6ed58ee15d5bb"}, {"checksum": "", "id": "20479400de9f822984176dd261d458855262c6185faa58f75b62dff1e202e57"}, {"checksum": "", "id": "346579db052c8e4e2987ea7b62556f6a45d74f26bb45cf42c03c2b42fa1c5916"}, {"checksum": "", "id": "4d4b27e7e977535ca10fca14470b2d552050980aac4c2458f5ab1eecd9fb"}, {"checksum": "", "id": "511136ea3c5a64f264b78b5433614aec563103b4d4702f3ba7d4d2698e22c158"}, {"checksum": "", "id": "51ee3d7e81dc5cc6b96670cc327b37a59cb09955766463be43c07aa45cd58ccd"}, {"checksum": "", "id": "6170bb7b0ad1003a827e4dc5253ba49f6719599eac485db51eafd507c13c311"}, {"checksum": "", "id": "6d71187017ae6227307f6a557967e7f84e639df33cbee4bde6372d85a8dde515"}, {"checksum": "", "id": "8fc5239c5506e9f3c75e46454ca0568036770b570962d32bc8064098687d6b9b"}, {"checksum": "", "id": "9cd978db300e27386baa9dd791bf6dc818f13e52235b26e95703361ec3c94dc6"}, {"checksum": "", "id": "ae66dff5cd30fe7caac82dd48465e3aec95cb0f25ee908fc1ea942b7ff1d4cc"}, {"checksum": "", "id": "b05d9cfbd4a1152bae28b8d24e8c99ad4f2c5e8432d67ffad6418c7344c1b2f"}, {"checksum": "", "id": "b853594c90668fc43061d8969cac461422f0376fb0b3329c9438fdac2c447024"}, {"checksum": "", "id": "db6645096445532065150deb624bcb8224b08ac891da6465614f0cc2fa8abb5f"}]
0
root@server2 ~ #
```

## 获取镜像

GET /v1/repositories/(namespace)/(repo\_name)/images

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl --raw https://index.docker.io/v1/repositories/sampritavh/dockerplayground/images
5c0
[{"checksum": "", "id": "079698a2ace7711326b27d5d101682f22dd3f22e67e63ab097cd6adf119de976"}, {"checksum": "", "id": "1d2e39f1d856a928e1ad808e8f6194b6b1deee3e5384ddf63095c247f9e70090"}, {"checksum": "", "id": "1f46da516f380d7a7fa3d38db36de30fdd0d8d6359a4ee3780a6ed58ee15d5bb"}, {"checksum": "", "id": "20479400de9f822984176dd261d458855262c6185faa58f75b62dff1e202e57"}, {"checksum": "", "id": "346579db052c8e4e2987ea7b62556f6a45d74f26bb45cf42c03c2b42fa1c5916"}, {"checksum": "", "id": "4d4b27e7ee977535ca10fca14470b2d552050980aac4c2458f5ab1eecfdd79fb"}, {"checksum": "", "id": "511136ea3c5a64f264b78b5433614aec563103b4d4702f3ba7d4d2698e22c158"}, {"checksum": "", "id": "51ee3d7e81dc5cc6b96670cc327b37a59cb09955766463be43c07aa45cd58ccd"}, {"checksum": "", "id": "6170bb7b0ad1003a827e4dc5253ba49f6719599eac485db51eaafd507c13c311"}, {"checksum": "", "id": "6d71187017ae6227307f6a557967e7f84e639df33cbee4bde6372d85a8dde515"}, {"checksum": "", "id": "8fc5239c5506e9f3c75e46454ca0568036770b570962d32bc8064098687d6b9b"}, {"checksum": "", "id": "9cd978db300e27386baa9dd791bf6dc818f13e52235b26e95703361ec3c94dc6"}, {"checksum": "", "id": "ae66dff5cd30fe7caac82dd48465e3aec95cb0f25ee908fc1ea942b7ff1d4cc"}, {"checksum": "", "id": "b05d9cfbd4a1152bae28b8d24e8c99ad4f2c5e8432d67ffad6418c7344c1b2f"}, {"checksum": "", "id": "b853594c90668fc43061d8969cac461422f0376fb0b3329c9438fdac2c447024"}, {"checksum": "", "id": "db6645096445532065150deb624bcb8224b08ac891da6465614f0cc2fa8abb5f"}]
0

root@server2 ~ #
```

## 用户认证

GET /v1/users

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl --raw -L --user sampritavh:test1234 https://index.docker.io/v1/users
4
"OK"
0

root@server2 ~ #
```

## 创建新用户

POST /v1/users

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl --raw -L -X POST --post301 -H "Accept: application/json" -H "Content-Type: application/json" --data-ascii '{"email": "test@flux7.com", "password": "toto42", "username": "testhubapi"}' https://index.docker.io/v1/users
e
"User created"
0

root@server2 ~ #
```

## 更新用户信息

PUT /v1/users/(username)/

## 总结

Docker API 已经讲了三个，而且都是 Docker Hub 架构相关的 API。下面两篇文章将专门讲 Docker Remote API，操作宿主机 Docker 服务端的 API。

[docker \(/t/docker/blogs\)](#)   [运维 \(/t/%E8%BF%90%E7%BB%B4/blogs\)](#)   [api \(/t/api/blogs\)](#)

[链接 \(/a/1190000002711455\)](#)   [更多 ▾](#)   [分享](#)

2 推荐

收藏

本文由 [大舒 \(/u/qiukeren\)](#) 创作，采用 **知识共享署名-相同方式 3.0 中国大陆许可协议** (<http://creativecommons.org/licenses/by-sa/3.0/cn>) 进行许可。  
转载、引用前需联系作者，并署名作者且注明文章出处。

## 你可能感兴趣的文章

- [Flux7 Docker 系列教程（八）： Docker Remote API \(/a/1190000002711475\)](#)   3 收藏，399 浏览
- [Docker 系列教程，Part 8: Docker Remote API \(/a/1190000000595056\)](#)   10 收藏，1.8k 浏览
- [Flux7 Docker 系列教程（九）：用于镜像操作的 10 个 Docker Remote API \(/a/1190000002711508\)](#)   3 收藏，446 浏览

## 讨论区

添加评论

提交评论

### 语法提示

评论支持部分 Markdown 语法：  
`**bold**`   `_italic_`   `[link](http://example.com)`   `>` 引用   ``code``  
- 列表。  
同时，被你 @ 的用户也会收到通知





本文隶属于专栏

## 大舒的博客 (/blog/qiukeke)

我都不写PHP了你们还挤兑我。。

关注专栏

---

## 系列文章

Flux7 Docker 系列教程（九）：用于镜像操作的 10 个 Docker Remote API (/a/1190000002711508)  
3 收藏， 446 浏览

Flux7 Docker 系列教程（八）： Docker Remote API (/a/1190000002711475) 3 收藏， 399 浏览

Docker 安全：通过 Docker 提升权限 (/a/1190000002789942) 9 收藏， 550 浏览

---

Copyright © 2011-2015 SegmentFault. 当前呈现版本 15.06.11  
浙ICP备15005796号-2 (<http://www.miibeian.gov.cn/>)