Docker入门教程（七）Docker API

【编者的话】DockerOne组织翻译了Flux7的Docker入门教程，本文是系列入门教程的第七篇，重点介绍了Docker Registry API和Docker Hub API。

纵观我们的Docker系列教程，我们已经讨论了很多重要的Docker组件与命令。在本文中，我们将继续深入学习Docker：剖析Docker APIs。

Docker提供了很多的API以便用户使用。这些API包含四个方面：

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

具体到这篇文章，我们将讨论Docker Registry API以及Docker Hub API。

Docker Registry API

Docker Registry API是Docker Registry的REST API，它简化了镜像和仓库的存储。该API不能访问用户帐户或者获得授权。你可以阅读Docker系列教程的第四章，以了解更多有关Registry的类型（译者注：Docker中有几种不同的Registry）。

Extract image layer:

取出镜像层：

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

```
root@server2: ~
root@server2 ~ # curl -o test-image-layer -v --raw http://localhost:5000/v1/images/c5881f11ded97
fd2252adf93268114329e985624c5d7bb86e439a36109d1124e/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      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 ~ #
```

Insert image layer:

插入镜像层：

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

Retrieve an image:

检索镜像:

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

Retrieve roots of an image:

检索根镜像：

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

Obtain all tags or specific tag of a repository:

获取库里所有的标签或者指定标签：

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

或者

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

```
root@server2: ~
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 ~ #
```

Delete a tag:

删除标签：

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

```
root@server2: ~
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 ~ #
```

Status check of registry:

registry状态检查：

`GET /v1/_ping`

```
root@server2: ~
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 Open
SSL/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
trueroot@server2 ~ #
```

Docker Hub API

Docker Hub API是Docker Hub的一个简单的REST API。再提醒一下，请参考Docker系列教程的第四篇文章了解Docker Hub。Docker Hub 通过管理校验（checksums）以及公共命名空间（public namespaces）来控制着用户帐户和授权。该API还支持有关用户仓库和library仓库的操作。

首先，让我们来看看特殊的library仓库（需要管理员权限）的命令：

1. **创建一个新的仓库**。使用以下命令可以创建新的library仓库：

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

其中，`repo_name`是新的仓库名称。

2. **删除已经存在的仓库**。命令如下：

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

其中，`repo_name`是要删除的仓库名称。

3. **更新仓库镜像**。命令如下：

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

4. **从仓库中获取镜像**。命令如下：

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

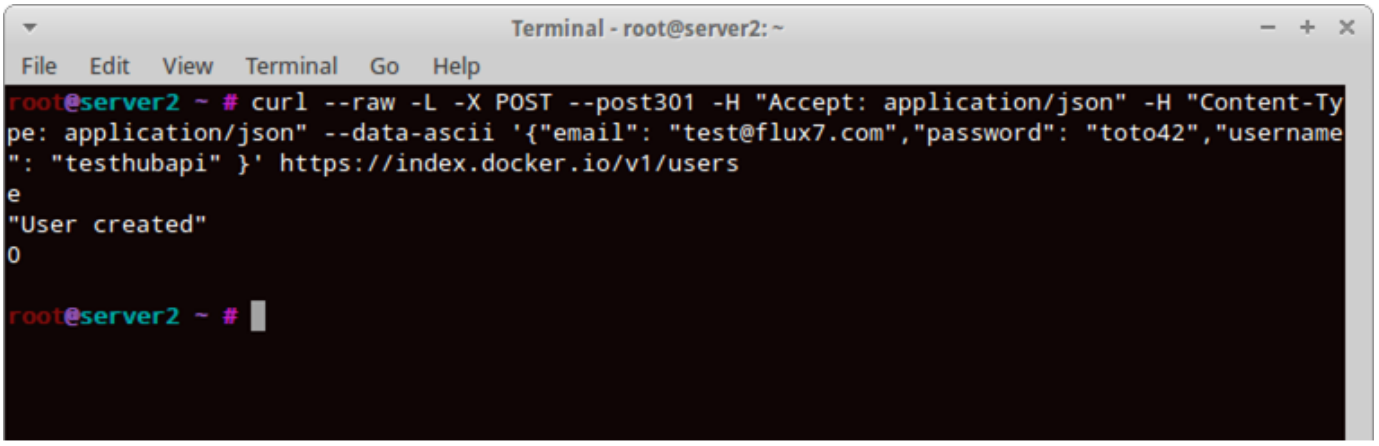
5. **授权**。使用Token获取仓库授权，如下：

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

接下来，让我们来看看用户仓库的命令。library仓库与用户仓库命令之间的主要区别是命名空间的使用。

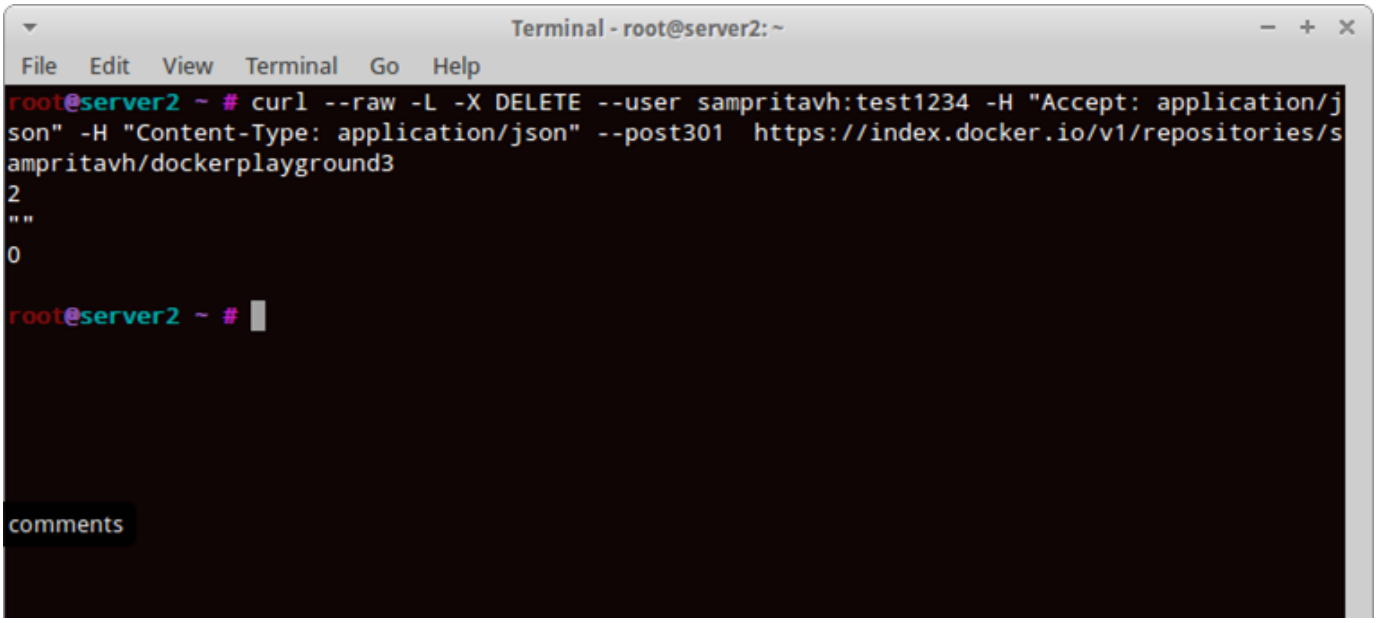
1.**创建用户仓库**。命令如下：

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



2.**删除用户仓库**，命令如下：

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



3.**更新用户仓库镜像**，命令如下：

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" : "079698a2ace7711326b27
d5d101682f22dd3f22e67e63ab097cd6adf119de976", "checksum": "b486531f9a779a0c17e3ed29dae8f12c4f
9e89cc6f0bc3c38722009fe6857087"}]' https://index.docker.io/v1/repositories/sampritavh/dockerp
layground/images
root@server2 ~ # curl --raw https://index.docker.io/v1/repositories/sampritavh/dockerplaygrou
nd/images
600
[{"checksum": "b486531f9a779a0c17e3ed29dae8f12c4f9e89cc6f0bc3c38722009fe6857087", "id": "0796
98a2ace7711326b27d5d101682f22dd3f22e67e63ab097cd6adf119de976"}, {"checksum": "", "id": "1d2e3
9f1d856a928e1ad808e8f6194b6b1deee3e5384ddf63095c247f9e70090"}, {"checksum": "", "id": "1f46da
516f380d7a7fa3d38db36de30fdd0d8d6359a4ee3780a6ed58ee15d5bb"}, {"checksum": "", "id": "2047940
0de9f822984176dd261d458855262c6185faa58f75b62dffd1e202e57"}, {"checksum": "", "id": "346579db
052c8e4e2987ea7b62556f6a45d74f26bb45cf42c03c2b42fa1c5916"}, {"checksum": "", "id": "4d4b27e7e
e977535ca10fca14470b2d552050980aac4c2458f5ab1eecfdd79fb"}, {"checksum": "", "id": "511136ea3c
5a64f264b78b5433614aec563103b4d4702f3ba7d4d2698e22c158"}, {"checksum": "", "id": "51ee3d7e81d
c5cc6b96670cc327b37a59cb09955766463be43c07aa45cd58ccd"}, {"checksum": "", "id": "6170bb7b0ad1
003a827e4dc5253ba49f6719599eac485db51eaafd507c13c311"}, {"checksum": "", "id": "6d71187017ae6
227307f6a557967e7f84e639df33cbee4bde6372d85a8dde515"}, {"checksum": "", "id": "8fc5239c5506e9
f3c75e46454ca0568036770b570962d32bc8064098687d6b9b"}, {"checksum": "", "id": "9cd978db300e273
86baa9dd791bf6dc818f13e52235b26e95703361ec3c94dc6"}, {"checksum": "", "id": "ae66dff5cd30fe7c
aac82dd48465e3aecd95cb0f25ee908fc1ea942b7ff1d4cc"}, {"checksum": "", "id": "b05d9cfbd4a1152ba
e28b8d24e8c99ad4f2c5e8432d67fffad6418c7344c1b2f"}, {"checksum": "", "id": "b853594c90668fc430
61d8969cac461422f0376fb0b3329c9438fdac2c447024"}, {"checksum": "", "id": "db66450964455320651
50deb624bcb8224b08ac891da6465614f0cc2fa8abb5f"}]
0

root@server2 ~ #
```

4.**从仓库中下载镜像。**如下：

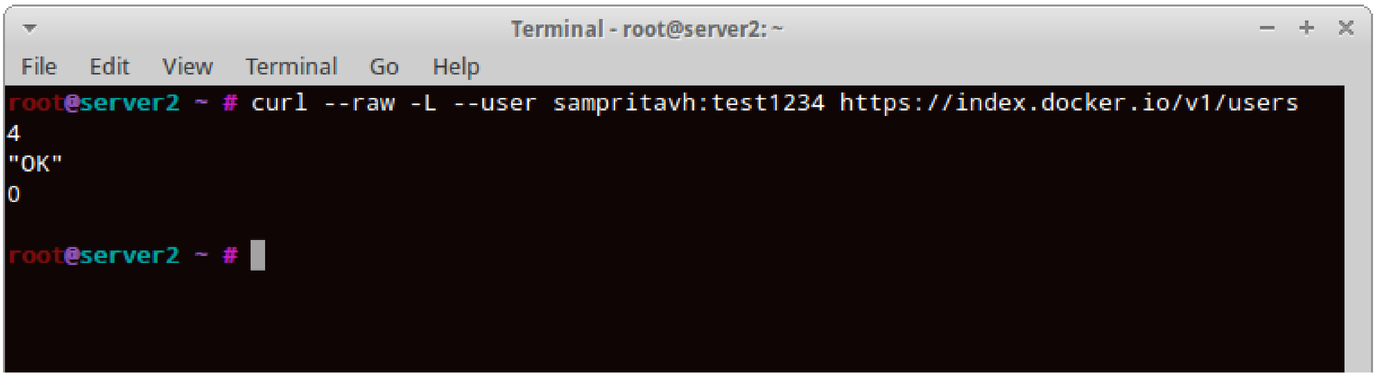
GET /v1/repositories/(namespace)/(repo_name)/images

```
Terminal - root@server2: ~
File  Edit  View  Terminal  Go  Help
root@server2 ~ # curl --raw https://index.docker.io/v1/repositories/sampritavh/dockerplaygrou
nd/images
5c0
[{"checksum": "", "id": "079698a2ace7711326b27d5d101682f22dd3f22e67e63ab097cd6adf119de976"},
{"checksum": "", "id": "1d2e39f1d856a928e1ad808e8f6194b6b1deee3e5384ddf63095c247f9e70090"}, {
"checksum": "", "id": "1f46da516f380d7a7fa3d38db36de30fdd0d8d6359a4ee3780a6ed58ee15d5bb"}, {"
checksum": "", "id": "20479400de9f822984176dd261d458855262c6185faa58f75b62dffd1e202e57"}, {"c
hecksum": "", "id": "346579db052c8e4e2987ea7b62556f6a45d74f26bb45cf42c03c2b42fa1c5916"}, {"ch
ecksum": "", "id": "4d4b27e7ee977535ca10fca14470b2d552050980aac4c2458f5ab1eecfdd79fb"}, {"che
cksum": "", "id": "511136ea3c5a64f264b78b5433614aec563103b4d4702f3ba7d4d2698e22c158"}, {"chec
ksum": "", "id": "51ee3d7e81dc5cc6b96670cc327b37a59cb09955766463be43c07aa45cd58ccd"}, {"check
sum": "", "id": "6170bb7b0ad1003a827e4dc5253ba49f6719599eac485db51eaafd507c13c311"}, {"checks
um": "", "id": "6d71187017ae6227307f6a557967e7f84e639df33cbee4bde6372d85a8dde515"}, {"checksu
m": "", "id": "8fc5239c5506e9f3c75e46454ca0568036770b570962d32bc8064098687d6b9b"}, {"checksum
": "", "id": "9cd978db300e27386baa9dd791bf6dc818f13e52235b26e95703361ec3c94dc6"}, {"checksum"
: "", "id": "ae66dff5cd30fe7caac82dd48465e3aecd95cb0f25ee908fc1ea942b7ff1d4cc"}, {"checksum":
 "", "id": "b05d9cfbd4a1152bae28b8d24e8c99ad4f2c5e8432d67fffad6418c7344c1b2f"}, {"checksum":
"", "id": "b853594c90668fc43061d8969cac461422f0376fb0b3329c9438fdac2c447024"}, {"checksum": "
", "id": "db66450964455320651 50deb624bcb8224b08ac891da6465614f0cc2fa8abb5f"}]
0

root@server2 ~ #
```

5.**验证用户登录**，如下：

`GET /v1/users`



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

root@server2 ~ #
```

6.**添加新用户**，命令如下：

`POST /v1/users`

7.**更新用户信息**，如下：

`PUT /v1/users/(username)/`

现在，我们已经走过了Docker API之旅的第一站，第二站是有关Docker OAuth以及Remote API的内容，我们将在Docker系列教程的下一篇见。

原文链接：Ultimate Guide for Docker APIs（翻译：田浩浩 审校：李颖杰）

＝ ＝

译者介绍

田浩浩，悉尼大学USYD硕士研究生，目前在珠海从事Android应用开发工作。业余时间专注Docker的学习与研究，希望通过DockerOne把最新最优秀的译文贡献给大家，与读者一起畅游Docker的海洋。

Docker入门教程（一）介绍
Docker入门教程（二）命令
Docker入门教程（三）DockerFile
Docker入门教程（四）Docker Registry
Docker入门教程（五）Docker安全
Docker入门教程（六）另外的15个Docker命令
Docker入门教程（七）Docker API
Docker入门教程（八）Docker Remote API
Docker入门教程（九）10个镜像相关的API