

Docker入门教程（八）Docker Remote 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八篇，重点介绍了Docker Remote API。

在[Docker系列教程的上一篇文章](#)中，我们学习了Docker Hub 以及 Docker Registry API。在本文中，让我们来看看Docker Remote API。

Docker Remote API

Docker Remote API是一个取代远程命令行界面（rcli）的REST API。本文中，我们将使用命令行工具cURL来处理url相关操作。cURL可以发送请求、获取以及发送数据、检索信息。

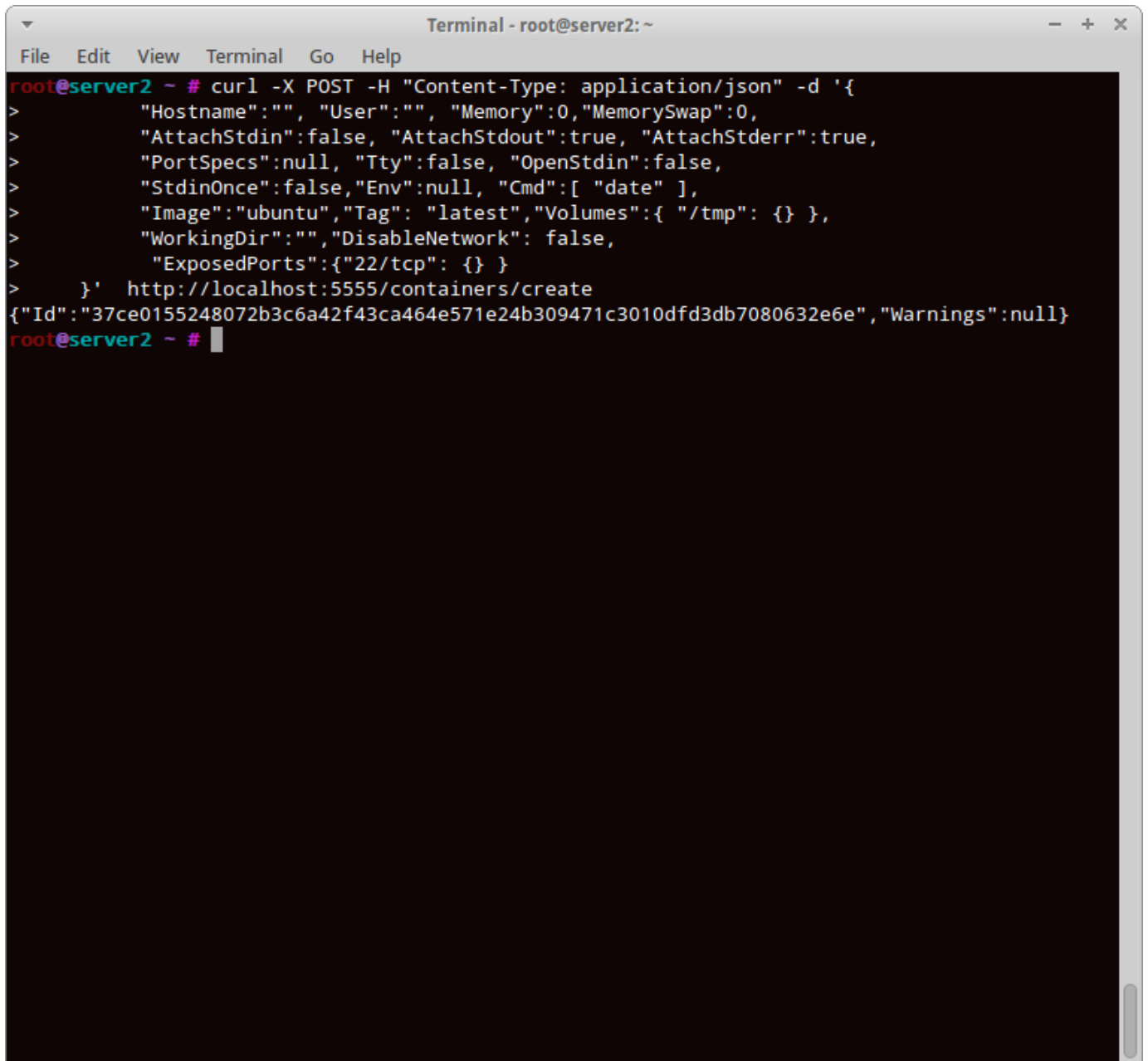
容器列表 获取所有容器的清单：

```
GET /containers/json
```

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl http://localhost:5555/containers/json?all=1
[{"Command":"apt-get install -y php5","Created":1403632835,"Id":"a57ae6c5946f94ce0862f06e2dc014a51e80011d0ac78e10df2c038930bd439a","Image":"localhost:5000/ubuntu:precise","Names":["/angry_fermi"],"Ports":[],"Status":"Exited (100) 47 hours ago"}, {"Command":"apt-get install -y php","Created":1403632828,"Id":"0c70652cb40c89d49170cac27ea6a23219d17a7bfce9a1da12aafca99e5e2b2e","Image":"localhost:5000/ubuntu:precise","Names":["/berse_rk_hopper"],"Ports":[],"Status":"Exited (100) 47 hours ago"}, {"Command":"apt-get install -y php5","Created":1403617250,"Id":"5266ab00510bd8a9fd30dd1c8fe5175d05fd8b7bb37dc8f6f64ebe78cd61354a","Image":"localhost:5000/ubuntu:latest","Names":["/agitated_colden"],"Ports":[],"Status":"Exited (0) 2 days ago"}, {"Command":"/bin/sh -c 'exec docker-registry'", "Created":1403523035, "Id":"27ab5fdd7ff0c0bf33fc0d11062ea42b1ed0d6b1e51b9bba0dc44f00fc0f7677", "Image":"registry:0.7.3", "Names":["/stupefied_almeida"], "Ports":[{"IP":"0.0.0.0", "PrivatePort":5000, "PublicPort":5000, "Type":"tcp"}], "Status":"Up 2 minutes"}, {"Command":"/bin/bash", "Created":1403521900, "Id":"aef35ae99a71fb208820039913c81052ca0117a17d913ea1b390f8074c02c935", "Image":"localhost:5000/ubuntu:latest", "Names":["/angry_turing"], "Ports":[], "Status":"Exited (127) 3 days ago"}, {"Command":"/bin/sh -c 'exec docker-registry'", "Created":1403503952, "Id":"9cb61e928f4e27118227f55e31e103f8d3d9cac1676389e76f137c4ad3389dca", "Image":"registry:0.7.3", "Names":["/sleepy_sammiet"], "Ports":[{"IP":"0.0.0.0", "PrivatePort":5000, "PublicPort":5000, "Type":"tcp"}], "Status":"Exited (0) 3 days ago"}, {"Command":"/bin/sh -c 'exec docker-registry'", "Created":1403503832, "Id":"ab3590e5d08f7918262836be12e2a812a2f0df2663f5d71458d3feb0170e750c", "Image":"registry:0.7.3", "Names":["/lonely_hawking"], "Ports":[{"IP":"0.0.0.0", "PrivatePort":5000, "PublicPort":5000, "Type":"tcp"}], "Status":"Exited (0) 3 days ago"}, {"Command":"/bin/bash", "Created":1402312032, "Id":"a716706a96dcfb28d08415efa3e8bc3aaa16725514f2a77dce8cf3a94e34aae8", "Image":"samprita:mb6", "Names":["/happy_feynmann"], "Ports":[], "Status":"Exited (0) 2 weeks ago"}]
root@server2 ~ #
```

创建新容器。命令如下：

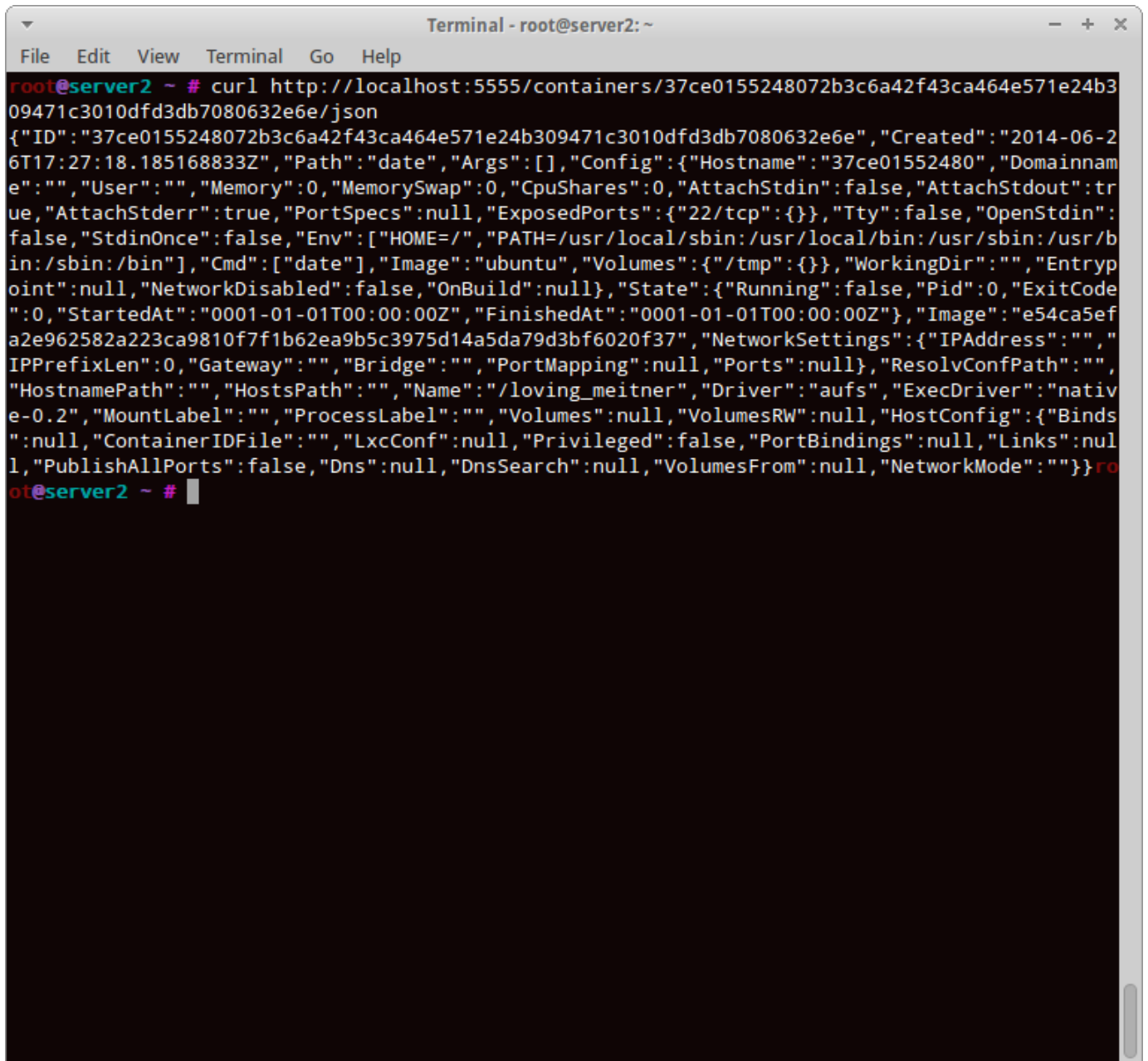
```
POST /containers/create
```

A terminal window titled "Terminal - root@server2: ~" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows a curl command being executed to create a Docker container. The command is: curl -X POST -H "Content-Type: application/json" -d '{> "Hostname": "", "User": "", "Memory": 0, "MemorySwap": 0,> "AttachStdin": false, "AttachStdout": true, "AttachStderr": true,> "PortSpecs": null, "Tty": false, "OpenStdin": false,> "StdinOnce": false, "Env": null, "Cmd": ["date"],> "Image": "ubuntu", "Tag": "latest", "Volumes": { "/tmp": {} },> "WorkingDir": "", "DisableNetwork": false,> "ExposedPorts": { "22/tcp": {} }> }' http://localhost:5555/containers/create. The output is a JSON object: {"Id": "37ce0155248072b3c6a42f43ca464e571e24b309471c3010dfd3db7080632e6e", "Warnings": null}. The prompt returns to root@server2 ~ #.

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl -X POST -H "Content-Type: application/json" -d '{
>   "Hostname": "", "User": "", "Memory": 0, "MemorySwap": 0,
>   "AttachStdin": false, "AttachStdout": true, "AttachStderr": true,
>   "PortSpecs": null, "Tty": false, "OpenStdin": false,
>   "StdinOnce": false, "Env": null, "Cmd": [ "date" ],
>   "Image": "ubuntu", "Tag": "latest", "Volumes": { "/tmp": {} },
>   "WorkingDir": "", "DisableNetwork": false,
>   "ExposedPorts": { "22/tcp": {} }
> }' http://localhost:5555/containers/create
{"Id": "37ce0155248072b3c6a42f43ca464e571e24b309471c3010dfd3db7080632e6e", "Warnings": null}
root@server2 ~ #
```

监控容器。使用容器id获取该容器底层信息：

```
GET /containers/(id)/json
```

A terminal window titled "Terminal - root@server2: ~" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows a curl command being executed: `curl http://localhost:5555/containers/37ce0155248072b3c6a42f43ca464e571e24b309471c3010dfd3db7080632e6e/json`. The output is a large JSON object representing container configuration and state. The prompt `root@server2 ~ #` is visible at the end of the output line.

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl http://localhost:5555/containers/37ce0155248072b3c6a42f43ca464e571e24b309471c3010dfd3db7080632e6e/json
{"ID":"37ce0155248072b3c6a42f43ca464e571e24b309471c3010dfd3db7080632e6e","Created":"2014-06-26T17:27:18.185168833Z","Path":"date","Args":[],"Config":{"Hostname":"37ce01552480","Domainname":"","User":"","Memory":0,"MemorySwap":0,"CpuShares":0,"AttachStdin":false,"AttachStdout":true,"AttachStderr":true,"PortSpecs":null,"ExposedPorts":{"22/tcp":{}},"Tty":false,"OpenStdin":false,"StdinOnce":false,"Env":["HOME=/","PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/bin:/sbin:/bin"],"Cmd":["date"],"Image":"ubuntu","Volumes":{"tmp":{}},"WorkingDir":"","EntryPoint":null,"NetworkDisabled":false,"OnBuild":null},"State":{"Running":false,"Pid":0,"ExitCode":0,"StartedAt":"0001-01-01T00:00:00Z","FinishedAt":"0001-01-01T00:00:00Z"},"Image":"e54ca5efa2e962582a223ca9810f7f1b62ea9b5c3975d14a5da79d3bf6020f37","NetworkSettings":{"IPAddress":"","IPPrefixLen":0,"Gateway":"","Bridge":"","PortMapping":null,"Ports":null},"ResolvConfPath":"","HostnamePath":"","HostsPath":"","Name":"/loving_meitner","Driver":"aufs","ExecDriver":"native-0.2","MountLabel":"","ProcessLabel":"","Volumes":null,"VolumesRW":null,"HostConfig":{"Binds":null,"ContainerIDFile":"","LxcConf":null,"Privileged":false,"PortBindings":null,"Links":null,"PublishAllPorts":false,"Dns":null,"DnsSearch":null,"VolumesFrom":null,"NetworkMode":""}}
root@server2 ~ #
```

进程列表。 获取容器内进程的清单：

```
GET /containers/(id)/top
```

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl http://localhost:5555/containers/27ab5fdd7ff0/top
{"Processes":[[{"root","6824","6800","0","19:06","?","00:00:00","/usr/bin/python /usr/local/bin/gunicorn --access-logfile - --debug --max-requests 100 --graceful-timeout 3600 -t 3600 -k gevent -b 0.0.0.0:5000 -w 4 docker_registry.wsgi:application"}, {"root","6870","6824","0","19:06","?","00:00:00","/usr/bin/python /usr/local/bin/gunicorn --access-logfile - --debug --max-requests 100 --graceful-timeout 3600 -t 3600 -k gevent -b 0.0.0.0:5000 -w 4 docker_registry.wsgi:application"}, {"root","6871","6824","0","19:06","?","00:00:00","/usr/bin/python /usr/local/bin/gunicorn --access-logfile - --debug --max-requests 100 --graceful-timeout 3600 -t 3600 -k gevent -b 0.0.0.0:5000 -w 4 docker_registry.wsgi:application"}, {"root","6876","6824","0","19:06","?","00:00:00","/usr/bin/python /usr/local/bin/gunicorn --access-logfile - --debug --max-requests 100 --graceful-timeout 3600 -t 3600 -k gevent -b 0.0.0.0:5000 -w 4 docker_registry.wsgi:application"}, {"root","6877","6824","0","19:06","?","00:00:00","/usr/bin/python /usr/local/bin/gunicorn --access-logfile - --debug --max-requests 100 --graceful-timeout 3600 -t 3600 -k gevent -b 0.0.0.0:5000 -w 4 docker_registry.wsgi:application"}], "Titles":["UID","PID","PPID","C","STIME","TTY","TIME","CMD"]}
```

容器日志。 获取容器的标准输出和错误日志：

```
GET /containers/(id)/logs
```

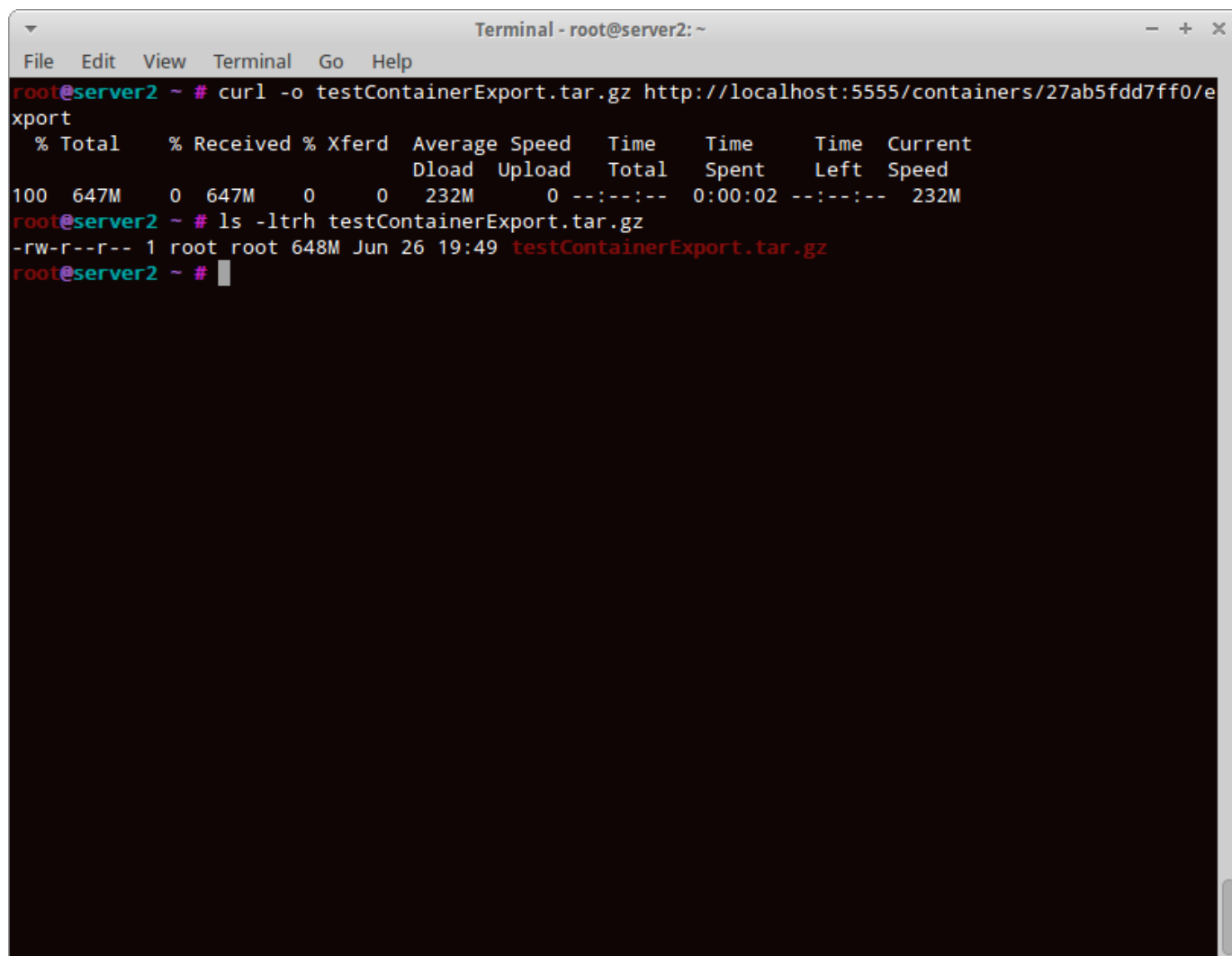
```

Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl http://localhost:5555/containers/27ab5fdd7ff0/logs?stderr=1&stdout=1&timestamps=1&follow=1
[1] 6979
[2] 6980
[3] 6981
root@server2 ~ # 2014-06-23 11:30:35,588 DEBUG: Will return docker-registry.drivers.file.Storage
24c5d7bb86e439a36109d1124e/checksum HTTP/1.1" 200 4 "-" "docker/0.11.1 go/go1.2.1 git-commit/fb99f99 kernel/3.8.0-33-generic os/linux arch/amd64"
2014-06-23 11:35:26,868 INFO: 172.17.42.1 - - [23/Jun/2014:11:35:26] "PUT /v1/images/c5881f11d97fd2252adf93268114329e985624c5d7bb86e439a36109d1124e/checksum HTTP/1.1" 200 4 "-" "docker/0.11.1 go/go1.2.1 git-commit/fb99f99 kernel/3.8.0-33-generic os/linux arch/amd64"
2014-06-23 11:35:26,884 DEBUG: args = {'tag': 'test'}
2014-06-23 11:35:26,885 DEBUG: [put_tag] namespace=library; repository=test; tag=test
172.17.42.1 - - [23/Jun/2014:11:35:26] "PUT /v1/repositories/test/tags/test HTTP/1.1" 200 4 "-" "docker/0.11.1 go/go1.2.1 git-commit/fb99f99 kernel/3.8.0-33-generic os/linux arch/amd64"
2014-06-23 11:35:26,886 INFO: 172.17.42.1 - - [23/Jun/2014:11:35:26] "PUT /v1/repositories/test/tags/test HTTP/1.1" 200 4 "-" "docker/0.11.1 go/go1.2.1 git-commit/fb99f99 kernel/3.8.0-33-generic os/linux arch/amd64"
2014-06-23 11:35:26,888 DEBUG: args = {'images': True}
172.17.42.1 - - [23/Jun/2014:11:35:27] "PUT /v1/repositories/test/images HTTP/1.1" 204 - "-" "docker/0.11.1 go/go1.2.1 git-commit/fb99f99 kernel/3.8.0-33-generic os/linux arch/amd64"
2014-06-23 11:35:27,091 INFO: 172.17.42.1 - - [23/Jun/2014:11:35:27] "PUT /v1/repositories/test/images HTTP/1.1" 204 - "-" "docker/0.11.1 go/go1.2.1 git-commit/fb99f99 kernel/3.8.0-33-generic os/linux arch/amd64"
172.17.42.1 - - [23/Jun/2014:11:37:28] "GET /v1/_ping HTTP/1.1" 200 4 "-" "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"
2014-06-23 11:37:28,465 INFO: 172.17.42.1 - - [23/Jun/2014:11:37:28] "GET /v1/_ping HTTP/1.1" 200 4 "-" "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"
172.17.42.1 - - [23/Jun/2014:11:38:16] "GET /v1/_ping HTTP/1.1" 200 4 "-" "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"
2014-06-23 11:38:16,242 INFO: 172.17.42.1 - - [23/Jun/2014:11:38:16] "GET /v1/_ping HTTP/1.1" 200 4 "-" "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"
172.17.42.1 - - [23/Jun/2014:11:41:54] "GET /v1/_ping HTTP/1.1" 200 4 "-" "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"

```

导出容器。导出容器内容：

```
GET /containers/(id)/export
```



```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl -o testContainerExport.tar.gz http://localhost:5555/containers/27ab5fdd7ff0/export
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed
100 647M    0 647M    0     0  232M      0 --:--:-- 0:00:02 --:--:-- 232M
root@server2 ~ # ls -ltrh testContainerExport.tar.gz
-rw-r--r-- 1 root root 648M Jun 26 19:49 testContainerExport.tar.gz
root@server2 ~ #
```

启动容器。如下：

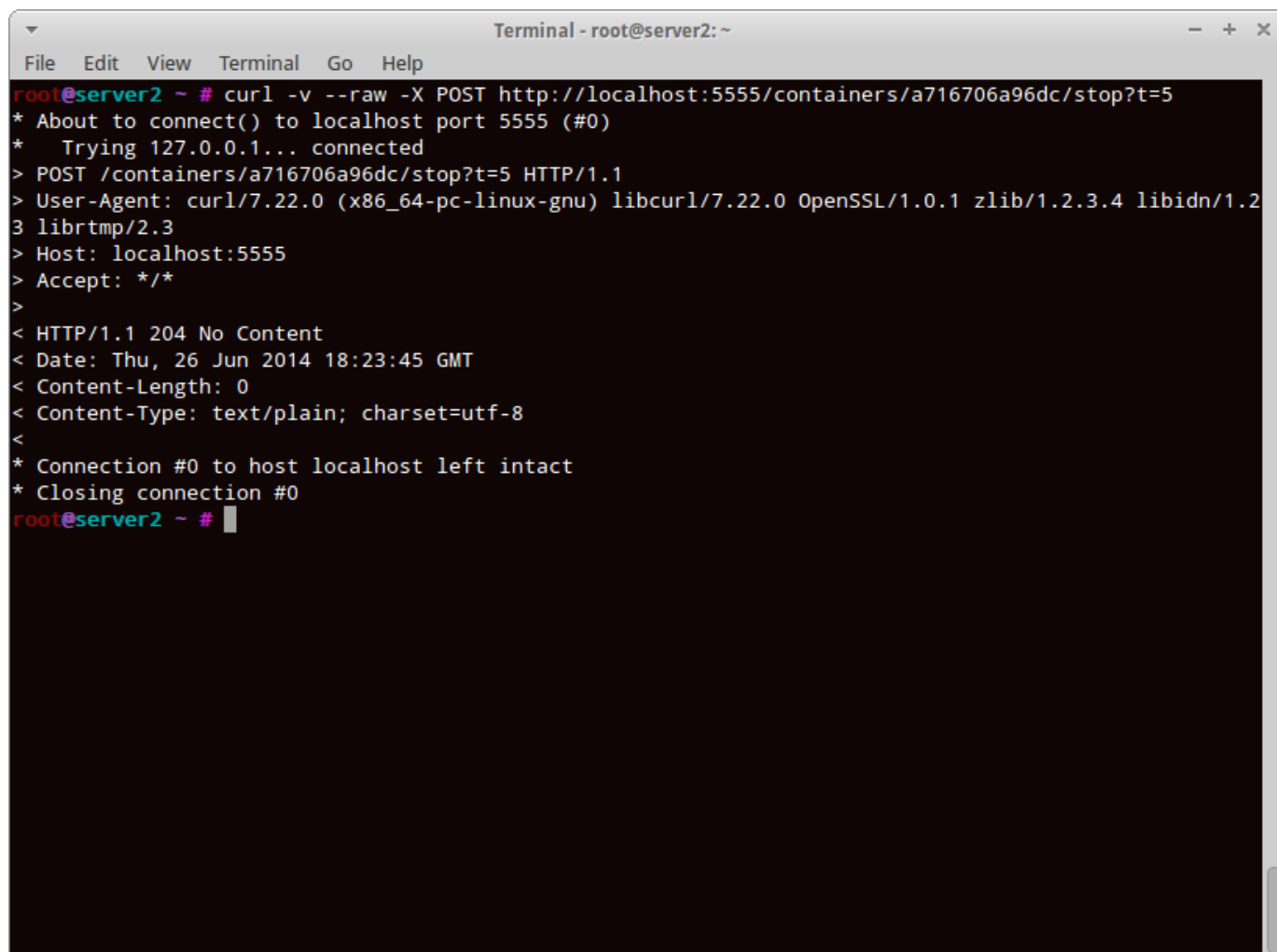
```
POST /containers/(id)/start
```



```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl -v -X POST -H "Content-Type: application/json" -d '{"Binds":["/tmp:/tmp"],
    "PortBindings":{"22/tcp": [{ "HostPort": "11022" }] }},
    "PublishAllPorts":false,
    "Privileged":false }' http://localhost:5555/containers/37ce0155248072b3c6a/start
* About to connect() to localhost port 5555 (#0)
*   Trying 127.0.0.1... connected
> POST /containers/37ce0155248072b3c6a/start 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/1.2
3 librtmp/2.3
> Host: localhost:5555
> Accept: */*
> Content-Type: application/json
> Content-Length: 152
>
* upload completely sent off: 152out of 152 bytes
< HTTP/1.1 204 No Content
< Date: Thu, 26 Jun 2014 18:20:00 GMT
< Content-Length: 0
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host localhost left intact
* Closing connection #0
root@server2 ~ #
```

停止容器。命令如下：

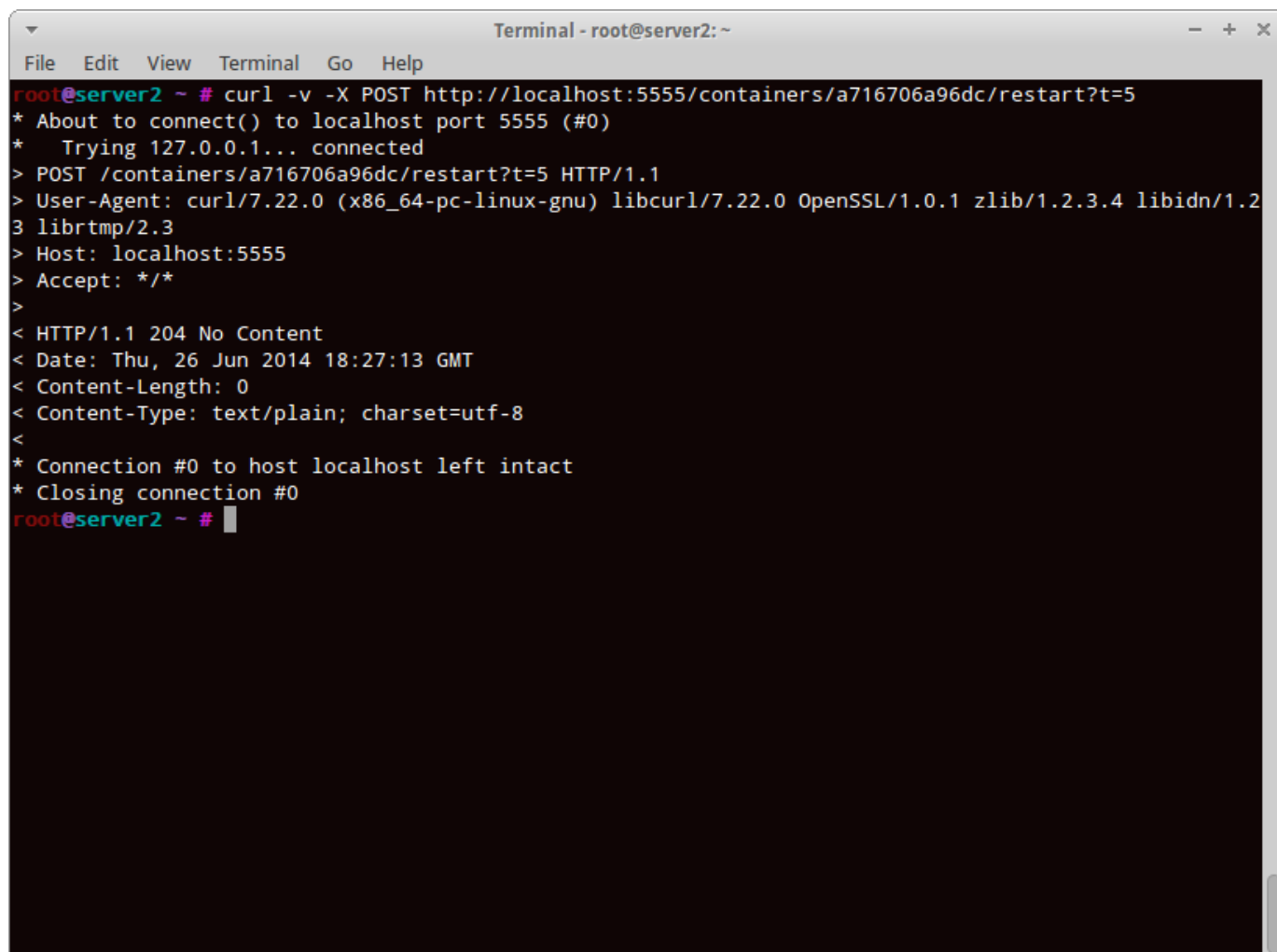
```
POST /containers/(id)/stop
```


A terminal window titled "Terminal - root@server2: ~" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows a curl command being executed to stop a container. The output shows the connection details and the successful completion of the request.

```
root@server2 ~ # curl -v --raw -X POST http://localhost:5555/containers/a716706a96dc/stop?t=5
* About to connect() to localhost port 5555 (#0)
*   Trying 127.0.0.1... connected
> POST /containers/a716706a96dc/stop?t=5 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/1.2
3 librtmp/2.3
> Host: localhost:5555
> Accept: */*
>
< HTTP/1.1 204 No Content
< Date: Thu, 26 Jun 2014 18:23:45 GMT
< Content-Length: 0
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host localhost left intact
* Closing connection #0
root@server2 ~ #
```

重启容器，如下：

```
POST /containers/(id)/restart
```

A terminal window titled "Terminal - root@server2: ~" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows a curl command being executed: `curl -v -X POST http://localhost:5555/containers/a716706a96dc/restart?t=5`. The output is a verbose HTTP response showing the connection to localhost:5555, the POST request, and the response headers: `HTTP/1.1 204 No Content`, `Date: Thu, 26 Jun 2014 18:27:13 GMT`, `Content-Length: 0`, and `Content-Type: text/plain; charset=utf-8`. The terminal ends with the prompt `root@server2 ~ #`.

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl -v -X POST http://localhost:5555/containers/a716706a96dc/restart?t=5
* About to connect() to localhost port 5555 (#0)
*   Trying 127.0.0.1... connected
> POST /containers/a716706a96dc/restart?t=5 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/1.2
3 librtmp/2.3
> Host: localhost:5555
> Accept: */*
>
< HTTP/1.1 204 No Content
< Date: Thu, 26 Jun 2014 18:27:13 GMT
< Content-Length: 0
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host localhost left intact
* Closing connection #0
root@server2 ~ #
```

终止容器：

```
POST /containers/(id)/kill
```

```
Terminal - root@server2: ~
File Edit View Terminal Go Help
root@server2 ~ # curl -v -X POST http://localhost:5555/containers/a716706a96dc/kill
* About to connect() to localhost port 5555 (#0)
* Trying 127.0.0.1... connected
> POST /containers/a716706a96dc/kill 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/1.2
3 librtmp/2.3
> Host: localhost:5555
> Accept: */*
>
< HTTP/1.1 204 No Content
< Date: Thu, 26 Jun 2014 18:30:41 GMT
< Content-Length: 0
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host localhost left intact
* Closing connection #0
root@server2 ~ # docker ps -a | grep a716706a96dc
a716706a96dc      samprita:mb6      /bin/bash      2 weeks ago      Exite
d (-1) 31 seconds ago      happy_feynmann
root@server2 ~ #
```

现在，我们已经带你走过了Docker API的第二站，[Docker系列教程的下一篇文章](#)会介绍有关镜像的Docker Remote API命令。我们所有的Docker系列教程你都可以在[这里](#)找到。

原文链接：[Docker Remote API](#)（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](#)

