

API7 Cloud 使用指南

文档更新时间：2023/03/23

可前往文档页面查看最新版本：<https://docs.api7.ai/cloud/overview/api7-cloud>

镜像使用

1. 镜像中默认安装了 Docker 和 [Cloud CLI](#) 工具（入门指南->添加网关实例）可直接部署云主机
2. 登陆后可以在由镜像创建的云主机安装 Apache APISIX（入门指南->添加网关实例）
3. 前往 API7 Cloud 注册登录，对云主机的 APISIX 进行托管
4. 需要开放的端口：
 - API7 网关
 - 80: HTTP 协议
 - 443: HTTPS 协议
 - 9091: 连接 Prometheus 和监控数据
 - API7 Dashboard
 - 9000: 访问 API7 企业版 Dashboard.
 - etcd
 - 2379: 存储数据和配置
 - 2380: 用于集群内的对等通信。

概述

什么是 API7 Cloud?

API7 Cloud 提供了一种简单而高效、安全的方式，可在多云环境中连接您的 API 和微服务。

现代企业 API 网关解决方案由数据平面（网关）和控制平面组成。网关处理流量并将请求代理到所需的后端，而控制平面则管理网关实例。它通常包含几个组件，如配置存储（存储网关配置），应用程序性能管理（帮助您测量 API 网关的运行状态）和用户门户以操作 API 网关。

传统上，管理员必须维护两个方面，特别是控制平面，由于它扮演了一些角色，如用户门户等。日常运维虽然繁琐但必不可少，否则您的 API 网关可能因不同原因随时停止工作。API7 Cloud 旨在从 API 网关控制平面管理中解放人们。用户仍需在其环境中部署它们（只要网络连接）。API7 Cloud 不会接触用户业务的任何流量，因此反过来，确保用户的隐私和安全。

现在，API7 Cloud 帮助您根据需求部署必要的服务，以便您可以：

1. 为网关配置[应用程序](#)和 [APIs](#)；
2. 了解网关实例的运行状态。

您的网关实例与 API7 Cloud 之间的连接是受保护的，因为双方都经过身份验证，即应用了双向 TLS（API7 Cloud 将生成证书）。

提示：

我们使用 Apache APISIX 作为 API7 Cloud 的数据平面网关。

Apache APISIX 如何连接到 API7 Cloud

在[概述](#)部分中，我们知道 API7 Cloud 的网关解决方案是 [Apache APISIX](#)，API7 Cloud 使用[官方 Apache APISIX 发行版](#)作为基础，并添加了一个额外的 Lua 模块来与 API7 Cloud 通信。

注意：

目前，API7 Cloud 仅支持 `APISIX/2.15.0` 及以上版本。

数据流

Apache APISIX 和 API7 Cloud 之间的数据流如下：

1. Apache APISIX 将定期向 API7 Cloud 发送 [Prometheus](#) 指标和统计数据（例如 API 调用次数）。
2. API7 Cloud 将向 Apache APISIX 发送配置更改（利用 Apache APISIX 配置中心）。

控制流

API7 Cloud 只与注册的 Apache APISIX 实例通信。因此，在数据流之前，Apache APISIX 需要向 API7 Cloud 注册自身。在初始注册之后，Apache APISIX 将定期发送心跳探测（每 10 秒一次）以保持活动状态。

如何检查 Apache APISIX 实例是否已注册

您可以在 API7 Cloud 概览页面（网关实例部分）上查看已注册的 Apache APISIX 实例。

ID	VERSION	STATUS	HOSTNAME	UPTIME	LAST REPORTED TIME
8e92cf73-c475-496d-8f7d-620f1a0af149	APISIX/2.15.0	Healthy	my-apisix	a few seconds	2023-01-13 14:40
06ce8c7d-6e3b-48a6-9a0b-6fece6bcbae4	APISIX/2.15.0	Offline	apisix-2023-01	a day	2023-01-12 15:56
7de9deb6-a593-4520-b78f-269dc9bbce63	APISIX/2.15.1	Healthy	qianyong-apisix-58fd774884-7fbcs	4 days	2023-01-13 14:40
8d6f8a5f-8a19-4eea-b187-1c12e71944fa	APISIX/2.15.1	Offline	qianyong-apisix-58fd774884-hpqdm	7 minutes	2023-01-09 16:31
a57b3aaa-5df6-465e-9f31-c8cb5f1fe426	APISIX/2.15.1	Offline	qianyong-apisix-58fd774884-njkwj	12 minutes	2023-01-09 16:24
862c92f8-3e38-494a-a4b4-a35a06aa3092	APISIX/2.15.0	Offline	3357e330b1e9	26 minutes	2023-01-06 17:37
4fc1f681-beec-4f6c-a89e-ad33b74e67d4	APISIX/2.15.1	Offline	qianyong-apisix-58fd774884-77lw9	3 days	2023-01-09 16:13

Results: 1-7 of 7

TIP

Learn [How to Deploy Apache APISIX](#) for more details.

提示：

了解[如何部署 Apache APISIX](#) 以获取更多详细信息。

API7 Cloud Lua 模块

数据流和控制流逻辑不是开源 Apache APISIX 的标准部分，我们将它们实现为单独的 Lua 模块：[Cloud Lua 模块](#)。由于 [Lua 模块钩子](#) 功能，我们可以轻松地挂载此模块，而不需要对 Apache APISIX 核心进行任何修改。

mTLS 支持

无论是数据流还是控制流，Apache APISIX 实例都可以通过 [mTLS](#) 安全地与 API7 Cloud 通信。证书和私钥可以从 API7 Cloud 下载，并应该为 Apache APISIX 实例配置。

Cloud CLI

我们使用 [Cloud CLI](#) 来操作 Cloud Lua 模块和证书的配置，因此用户不必手动执行上述步骤。因此，上述步骤对用户来说是不可感知的。

入门指南

您将开始您的 API7 Cloud 之旅，特别是通过学习管理自己的 API 的步骤来开始。

您的旅程将是：

1. 创建一个账户并登录到 API7 Cloud；
2. 添加一个网关实例并将其连接到 API7 Cloud；
3. 创建 HTTPBIN 应用程序；
4. 为 HTTPBIN 应用程序创建 API；
5. 使用 [curl](#) 测试 API 是否正常工作；
6. 将[限制计数插件](#)附加到 API 上；
7. 总结。

您准备好了吗？我们开始吧！

设置账户和登录

信息：

API7 Cloud 正在接受早期访问的 beta 申请者。感谢您填写表格。我们会尽快与您联系。
一旦您拥有 API7 Cloud 账户，您就可以访问[控制台](#)，我们的系统将引导您进入登录页面。



Welcome

Log in to api7-cloud-prod to continue to cloud-prod.

Email address

demo@api7.ai

Password

.....

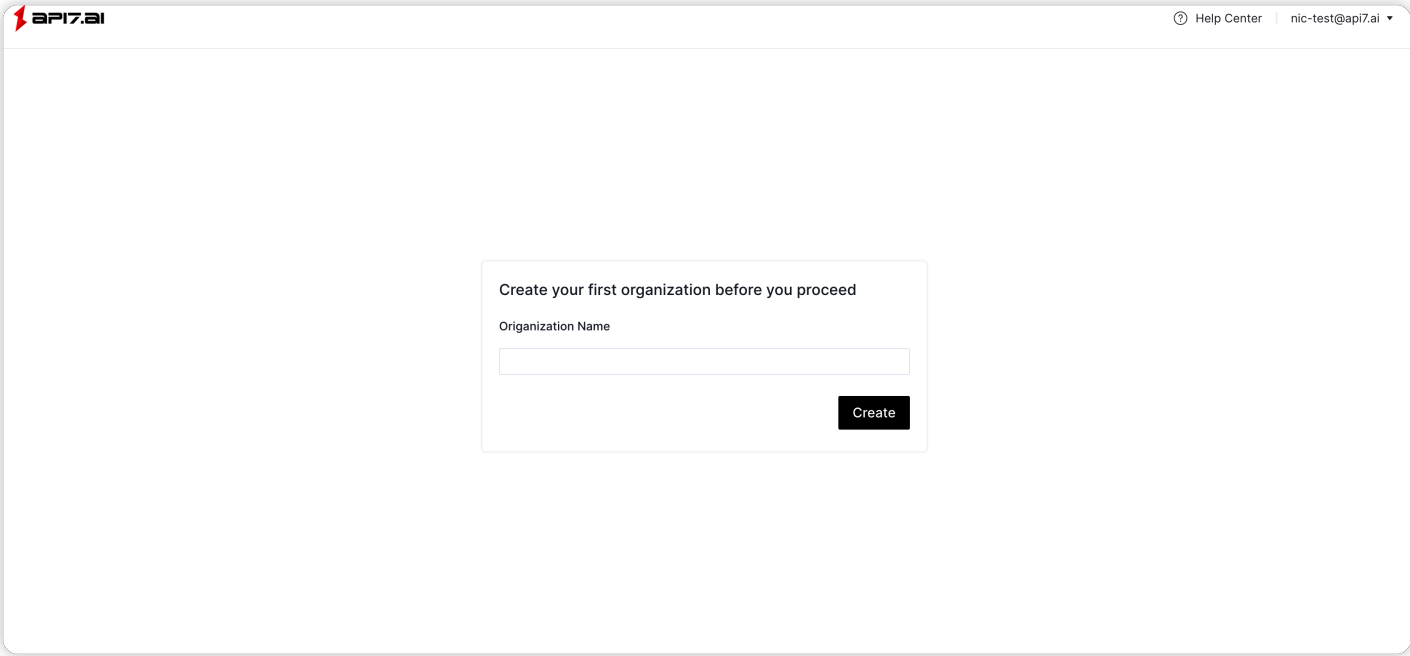


[Forgot password?](#)

Continue

成功登录后，API7 Cloud 将重定向您到控制台页面。当您在概述页面上时，您可能会看到“请稍等，您的服务正在部署中”的提示。如果出现这种情况，请耐心等待，API7 Cloud 很快就会完成部署。

请注意，如果这是您第一次登录控制台，API7 Cloud 将要求您输入组织名称。



添加网关实例

在注册后，您需要做的第一件最重要的事情是添加一个网关实例（[Apache APISIX](#)）并让它连接到 API7 Cloud。

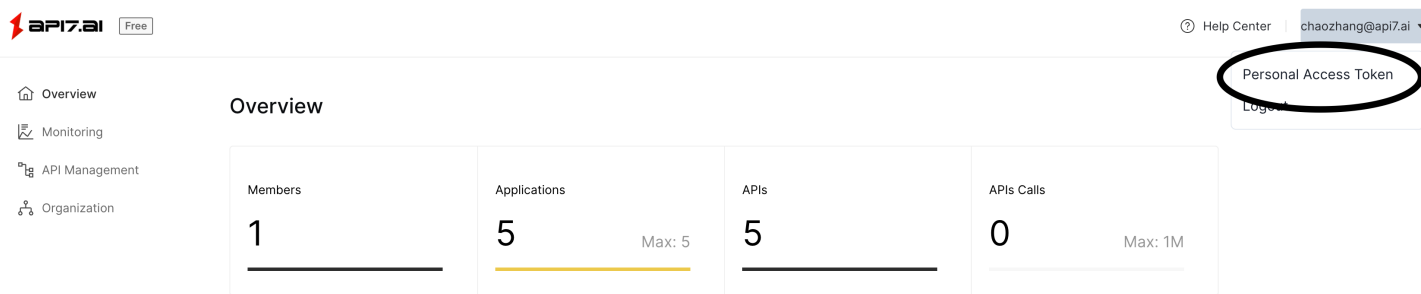
作为演示，我们将使用 [Cloud CLI](#) 在 Docker 上部署一个 APISIX 实例，因此在继续之前，请确保：

- 1. [Docker](#) 已安装在您的机器上。
- 2. 您的机器已连接到互联网。

提示：
如果您想了解更多详细信息，请参阅[如何部署 Apache APISIX 指南](#)。

首先，请参考 [Cloud CLI 安装](#) 以安装 Cloud CLI。

安装 Cloud CLI 后，请登录到 API7 Cloud 并创建 Personal Access Token。您可以通过单击帐户下拉框中的“Personal Access Token”按钮进入 API7 Cloud 上的个人访问令牌页面。



接着点击 “Generate a new token” 按钮。

< Back

Personal Access Token^①

+ Generate a new token

NOTES	STATUS	EXPIRY DATE
No Data		

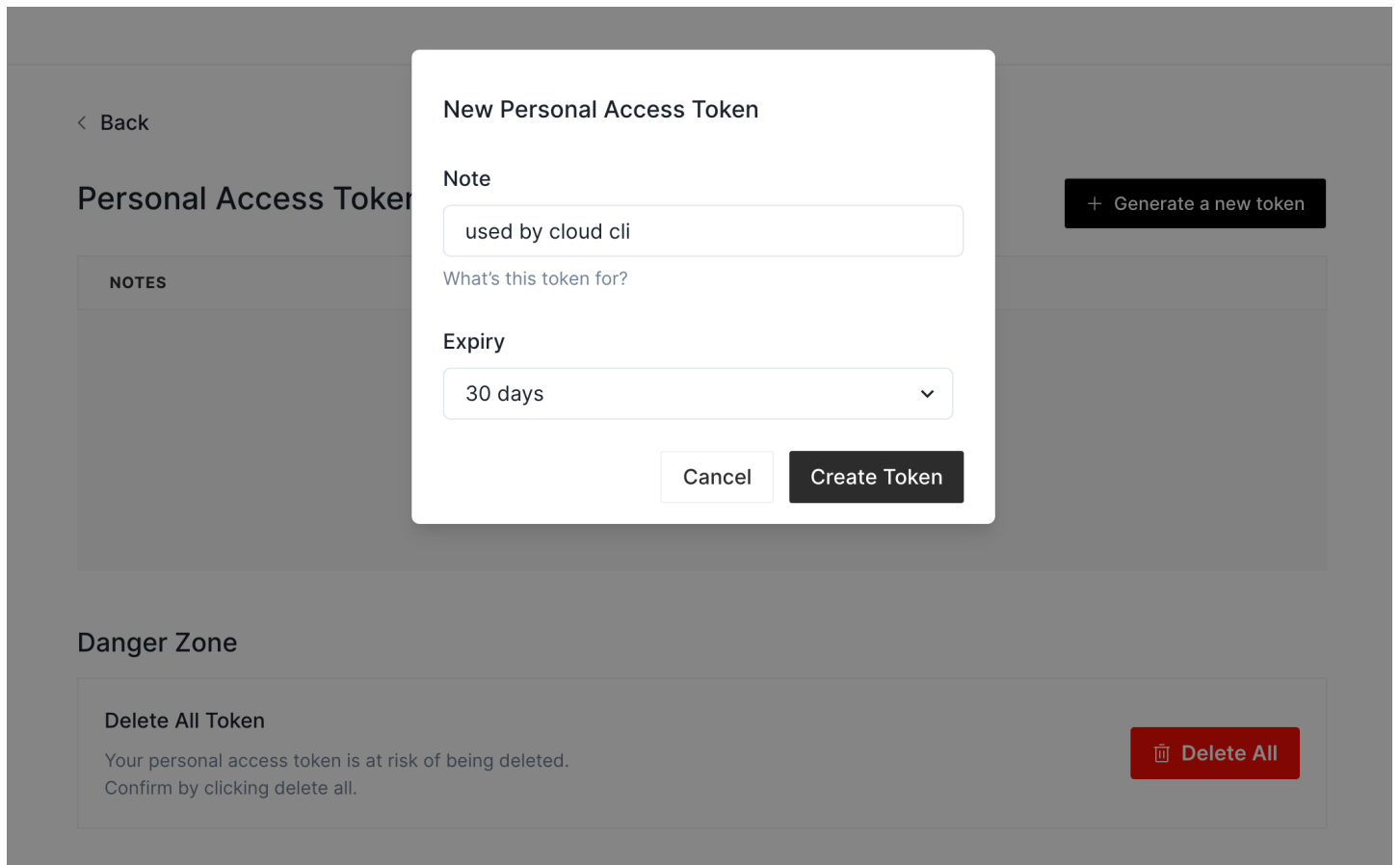
Danger Zone

Delete All Token

Your personal access token is at risk of being deleted.
Confirm by clicking delete all.

 Delete All

在您单击它之后，API7 Cloud 将显示一个弹出窗口，您需要填写令牌说明（这个令牌是用来做什么的？）和过期时间（这个令牌多长时间有效？）。



< Back

Personal Access Token

+ Generate a new token

NOTES
No Data

Danger Zone

Delete All Token

Your personal access token is at risk of being deleted.
Confirm by clicking delete all.

 Delete All

New Personal Access Token

Note

What's this token for?

Expiry

30 days

Cancel Create Token

点击“Create Token”按钮。API7 Cloud 将重定向您到个人访问令牌页面，您的新令牌将在此处显示。

< Back

Personal Access Token^①

+ Generate a new token

✓ You have generated a new personal access token for API7 Cloud
Make sure to copy your personal access token now. You won't be able to see it again!

[Redacted Token]



NOTES	STATUS	EXPIRY DATE	
used by cloud cli	Normal	Expires on Sun May 1, 2022	Delete
fgfgfgf	Normal	Expires on Fri Apr 8, 2022	Delete

Results: 1-2 of 2

< 1 >

Danger Zone

Delete All Token

Your personal access token is at risk of being deleted.
Confirm by clicking delete all.

🗑 Delete All

⚠ 危险提示：API7 Cloud 提醒您，您只有一次查看令牌的机会，请确保您正确保存它！

请从页面中复制 Token，并通过运行以下命令为 Cloud CLI 配置它：

```
1 cloud-cli configure
```

它将提示您输入 Token。

现在让我们尝试在 Docker 上部署 APISIX 实例。

```
1 cloud-cli deploy docker \
2   --apisix-image apache/apisix:2.15.0-centos \
3   --name my-apisix \
4   --docker-run-arg --detach \
5   --docker-run-arg --rm
```

如果部署成功，您将看到下面的输出。由于随机生成机制，容器 ID 和 APISIX ID 可能会有所不同。

```
1 Congratulations! Your APISIX instance was deployed successfully
2 Container ID: 0904c39c2551
3 APISIX ID: 19cb8b57-6436-40bd-8dc2-d3571a356b86
```

现在，让我们检查“网关实例”表格，您将看到刚刚部署的 APISIX 实例。

Gateway Instances

+ Add Instances

ID	VERSION	STATUS ⓘ	HOSTNAME	UPTIME ⚡	LAST REPORTED TIME
8e92cf73-c475-496d-8f7d-620f1a0af149	APISIX/2.15.0	Healthy	my-apisix	a few seconds	2023-01-13 14:40

如果您想停止 APISIX 实例，只需运行以下命令：

```
1 cloud-cli stop docker --name my-apisix
```

做的很好！您刚刚在 Docker 上部署了一个 APISIX 实例；它将从 API7 Cloud 接收配置，并向 API7 Cloud 报告状态。现在，让我们尝试在 API7 Cloud 上创建一个应用程序，以驱动您的 APISIX 实例。

创建 HTTPBIN 服务

现在是学习如何创建服务的时候了。通常情况下，您可以将物理微服务映射到 API7 Cloud 上的服务。在本节中，我们将创建一个 HTTPBIN 服务，以代理 HTTPBIN 网站。请按照以下步骤操作：

1. 打开 [API7 Cloud 控制台](#)。
2. 在左侧导航栏中选择 **API Management**，然后从二级菜单中选择 **Services**。
3. 点击 “**Create Service**” 按钮。
4. 填写表单，例如：
 - 我们将 “Path Prefix” 设置为 `/v1`，这意味着所有请求的 URI 路径都应该以 `/v1` 开头。
 - 我们将 HTTP 主机设置为 `cloud.httpbin.org`，这意味着所有请求到这个服务都应该带有 Host 头，值为 `cloud.httpbin.org`。
 - 我们将 “Upstream URL” 设置为 “`https://httpbin.org`”，这意味着所有请求到这个服务都将被代理到 “`httpbin.org`”。

做的很好！现在您已经创建了 HTTPBIN 服务，请继续下一节以在其中创建一些 API。

为 HTTPBIN 服务创建路由

现在我们已经成功创建了 HTTPBIN 服务，并指定了上游 URL。但是我们还没有任何路由。

在本节中，我们将创建一个路由，从 <https://httpbin.org> 返回一个 JSON 字符串。要创建这样的路由，请执行以下步骤：

1. 打开 [API7 Cloud 控制台](#)。
2. 从左侧导航栏中选择 **API Management**，然后从二级菜单中选择 **Services**。
3. 点击 HTTPBIN 服务的名称。
4. 在服务详细信息页面上，单击 **“Create Route”**。
5. 填写表单以创建目标路由。
 - 将“Path”字段设置为 `/json`。
 - 选择“Strip Path Prefix”选项，这意味着 `/v1/json` 将在代理到后端之前重写为“/json”；
 - 仅允许 `GET` 方法。

恭喜！您已经创建了 JSON 路由。现在，让我们发送一些请求以验证它是否正常工作！

测试 HTTPBIN 服务和路由

现在您已经创建了 HTTPBIN 服务和一个 JSON 路由。让我们发送一个请求以验证它是否正常工作。

```
1 curl http://127.0.0.1:9080/v1/json -H 'Host: cloud.httpbin.org' -v
```

我们设置了 Host 头为 `cloud.httpbin.org`，请求路径为 `/json`。根据指南，您在 `127.0.0.1:9080` 上有一个 APISIX 实例。您将看到下面的输出。

```
1 * Trying 127.0.0.1:9080...
2 * Connected to 127.0.0.1 (127.0.0.1) port 9080 (#0)
3 > GET /v1/json HTTP/1.1
4 > Host: cloud.httpbin.org
5 > User-Agent: curl/7.76.1
6 > Accept: */*
7 >
8 * Mark bundle as not supporting multiuse
```

```
9 < HTTP/1.1 200 OK
10 < Content-Type: application/json
11 < Content-Length: 429
12 < Connection: keep-alive
13 < X-RateLimit-Limit: 5
14 < X-RateLimit-Remaining: 4
15 < Date: Fri, 22 Apr 2022 02:20:41 GMT
16 < Access-Control-Allow-Origin: *
17 < Access-Control-Allow-Credentials: true
18 < Server: APISIX/2.15.0
19 <
20 {
21   "slideshow": {
22     "author": "Yours Truly",
23     "date": "date of publication",
24     "slides": [
25       {
26         "title": "Wake up to WonderWidgets!",
27         "type": "all"
28       },
29       {
30         "items": [
31           "Why <em>WonderWidgets</em> are great",
32           "Who <em>buys</em> WonderWidgets"
33         ],
34         "title": "Overview",
35         "type": "all"
36       }
37     ],
38     "title": "Sample Slide Show"
39   }
40 }
```

正如您所看到的，我们从 APISIX 得到了一个 JSON 字符串，这是我们想要得到的响应（从 <https://httpbin.org>）。

注意：

HTTPBIN 服务会随机生成 JSON 字符串数据，因此您得到的数据可能与从此请求获取的数据不同。

恭喜！现在您已经测试了 HTTPBIN 服务及其 JSON 路由，一切都正常工作。

添加 Limit Count 插件

到目前为止，您已经知道如何配置[服务和路由](#)，使您的 APISIX 实例按照您的期望运行。本节将使用 [Limit Count 插件](#) 扩展路由。Limit Count 可以保护您的路由免受大量请求的困扰。

要为 HTTPBIN 服务添加 Limit Count 插件，请执行以下操作：

1. 打开 [API7 Cloud 控制台](#)。
2. 从左侧导航栏中选择 **API Management**，然后从二级菜单中选择 **Services**。
3. 点击 HTTPBIN 服务名称，进入服务详细信息页面。
4. 点击 “**Add Plugin**”，选择 Limit Count 插件。
5. 填写表单：
 - 将计数器 **Count** 设置为 `5`。
 - 将周期 **Period** 设置为 `60`。
 - 将拒绝状态码 **Rejected Status Code** 设置为 `429`。
 - 将错误消息 **Error Message** 设置为 `{"error_msg": "Too many requests"}`。

在本例中，我们配置了：

1. 网关实例在一分钟内仅接受五个请求（针对 JSON 路由）；
2. 如果请求数量超过限制，网关实例将使用 `429` 状态码拒绝请求，响应正文将是 “Too many requests”。

现在让我们尝试验证 Limit Count 插件。

同样，我们将使用 [curl](#) 进行验证。这次我们将连续发送请求。

```
1 for ((i=0; i<6; i++)); do
2 curl http://127.0.0.1:9080/v1/json -H 'Host: cloud.httpbin.org' -s -o/dev/null -
3 done
```

输出将为：

```
1 status code: 200
2 status code: 200
3 status code: 200
4 status code: 200
5 status code: 200
6 status code: 429
```

正如您所看到的，我们在一分钟内发送了 `6` 个请求。前 `5` 个请求响应了 `200` 状态码，而最后一个请求响应了预期的 `429`，但响应正文如何呢？让我们通过以下命令单独发送一个请求。

```
1 curl http://127.0.0.1:9080/v1/json -H 'Host: cloud.httpbin.org' -s
```

您将在屏幕上看到打印了 `{"error_msg":"Too many requests"}`。

注意：

Apache APISIX将错误消息封装在JSON字符串中。

信息：

当您运行上面的命令时，限制配额可能已经重置。如果您看不到此输出，请尝试几次。

恭喜，您已经掌握了在路由中使用 Limit Count 插件的方法。

总结

欢迎来到这里！您已经完成了 API7 Cloud 的入门教程。如果您想进一步了解 API7 Cloud，可以阅读手册中的其他部分。有关更多详细信息，[请参阅 API7 Cloud 手册](#)。