

Postman 使用教程 - 手把手教你 API 接口测试

作者: [HiJiangChuan](#)

原文链接: <https://ld246.com/article/1640941436781>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



POSTMAN



Postman 使用教程 手把手教你 API 接口测试

卡拉云 kalacloud.com

本文首发: [Postman 使用教程 - API 接口自动化测试初探 - 卡拉云](#)

Postman 是一套 API 接口测试工具，它的强大在于灵活趁手的接口测试功能，极大的提高了 API 测试效率。本教程将由浅入深，带领大家一起学习如何使用 Postman 进行接口测试。

API 是什么？

API的英文即 **A**pplication **P**rogramming **I**nterface 首字母的缩写。不要被这么长的单词吓到，直译来的意思就是：程序之间的接口。我更倾向于把API理解为，程序之间的合约。有关 API 是什么及它意义这里就不展开讲了，了解更多可看卡拉云博客之前的文章《[API是什么: 一篇讲透API](#)》

Postman 是什么？

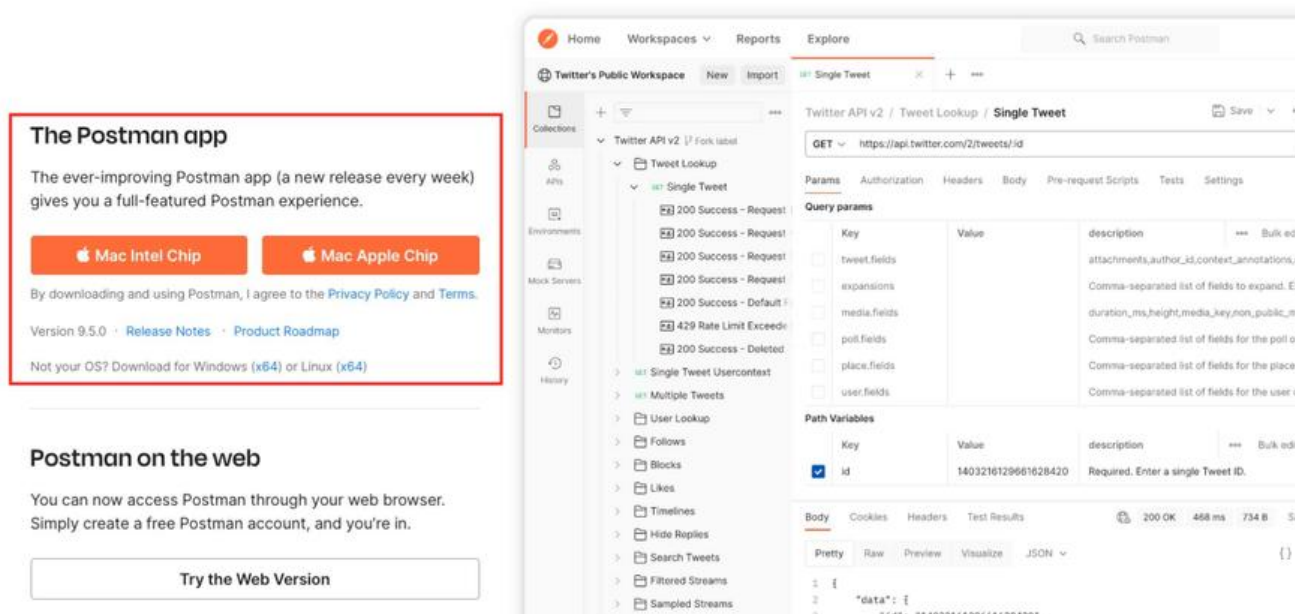
Postman 是一款 API 开发协作工具，它可以帮助你测试和开发 API，Postman 提供了测试 API 的友好界面和功能，使用简单便捷，安全可靠。Postman 是 API 接口测试工具的 Top 3，因为它简单可靠免费，目前有超过 800 万开发者用户使用。特别是 API 批量测试功能，在近几次的大版本更新中，用户体验得到了极大的提升。Postman 是每一位前后端开发者必掌握的开发工具。

一. 如何安装 Postman

Download Postman

卡拉云 kalacloud.com

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.



前往 [Postman 官网免费下载](#)，安装非常傻瓜，这里就不展开写了，如果安装过程有任何问题，欢迎论区提问。

Postman 支持 Windows、Mac 和 Linux，也可以直接跑在浏览器里，跨平台，使用相当便捷。

二. API 模拟工具 GoRest

本教程使用模拟 API 工具 [GoRest](#) 进行模拟 API 测试。GoRest 有非常多的使用场景，比如，后端还没有准备好时，我们可以先用 GoRest 模拟测试，又或者你自己暂时不想搞服务器，也不想搭后端也可以先用 GoRest 来测。

GoRest 可以理解为后端工程师帮你搭好了后端服务器，而且是完全测过的，几乎不可能有 Bug。GoRest 除了其中的数据是模拟的，所有 API 响应都是完全真实的，你可以通过 API 调用的返回数据判断的前端是否有问题。

Online REST API for Testing and Prototyping

fake data | real responses | 24/7 online

Resources		Trying it Out	
https://gorest.co.in/public/v1/users	1550	POST /public/v1/users	Create a new user
https://gorest.co.in/public/v1/posts	1243	GET /public/v1/users/123	Get user details
https://gorest.co.in/public/v1/comments	1202	PUT PATCH /public/v1/users/123	Update user details
https://gorest.co.in/public/v1/todos	1904	DELETE /public/v1/users/123	Delete user

GoRest 的 API 调用 URL 及规则。

通常 API 由 BaseURL + Endpoint 构成。比如上图中第一个 Resources:

<https://gorest.co.in/public/v1/users>

这个 API 是由 <https://gorest.co.in> + `/public/v1/users` 即 BaseURL + Endpoint 组成的。一般我们从 API 地址看出调取信息的大意，这段 URL 是调用 GoRest 上有关 users 的信息。

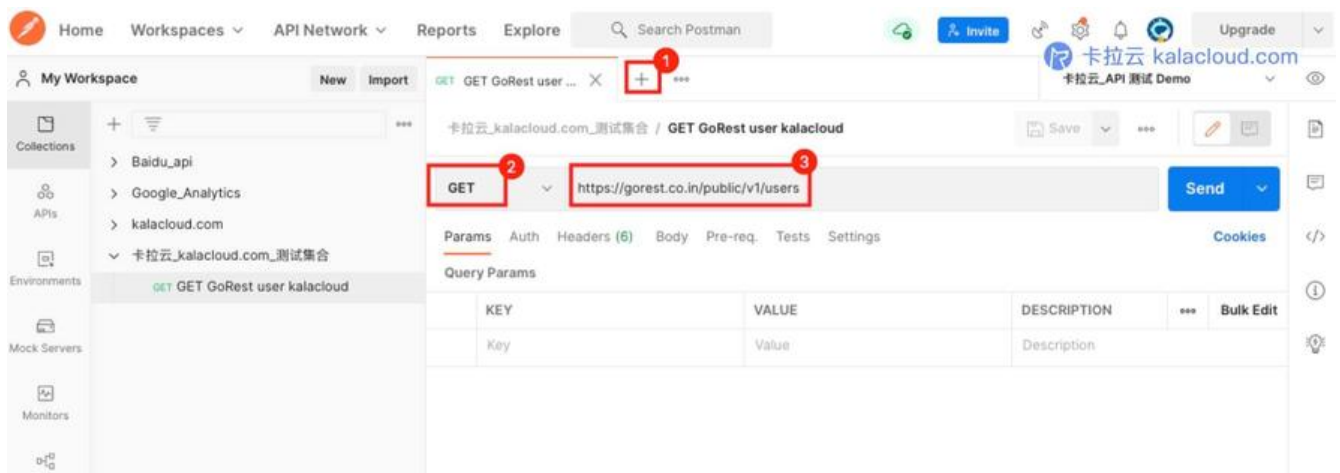
下一节，我们使用 GET 请求调用这个 API，看看返回结果是不是与 users 信息有关。

三. 用 Postman 发出第一个 GET 请求

1. GET 请求基本操作

GET 请求是 API 中使用最频繁的请求之一，GET 请求仅从数据库中请求读取数据，不会修改服务器的数据。

接下来，我们来创建一个基本的 GET 请求



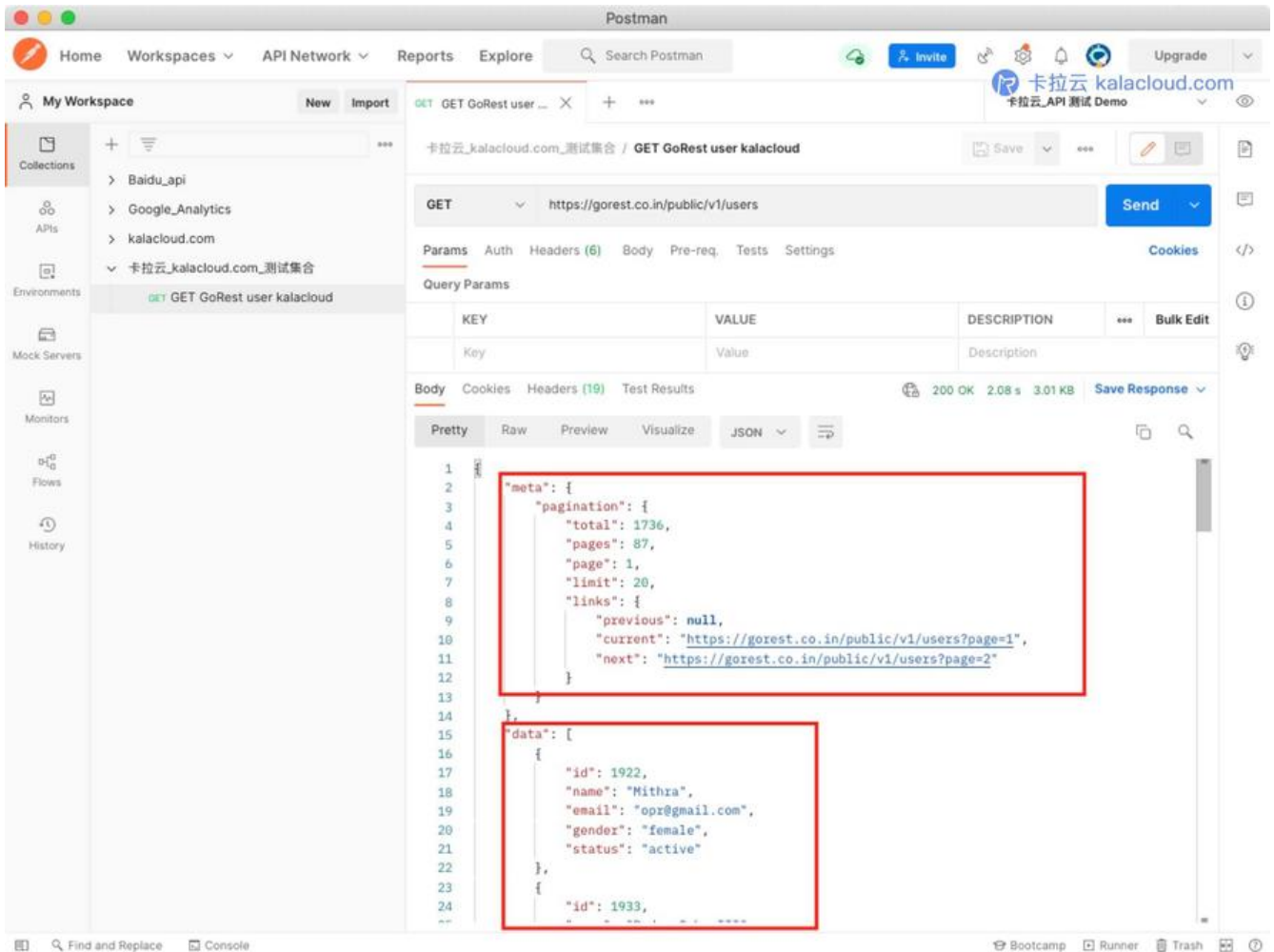
(1) 点击主界面「+」号，新建一个请求页

(2) 选择 GET 请求命令

(3) 输入 API 地址：

<https://gorest.co.in/public/v1/users>

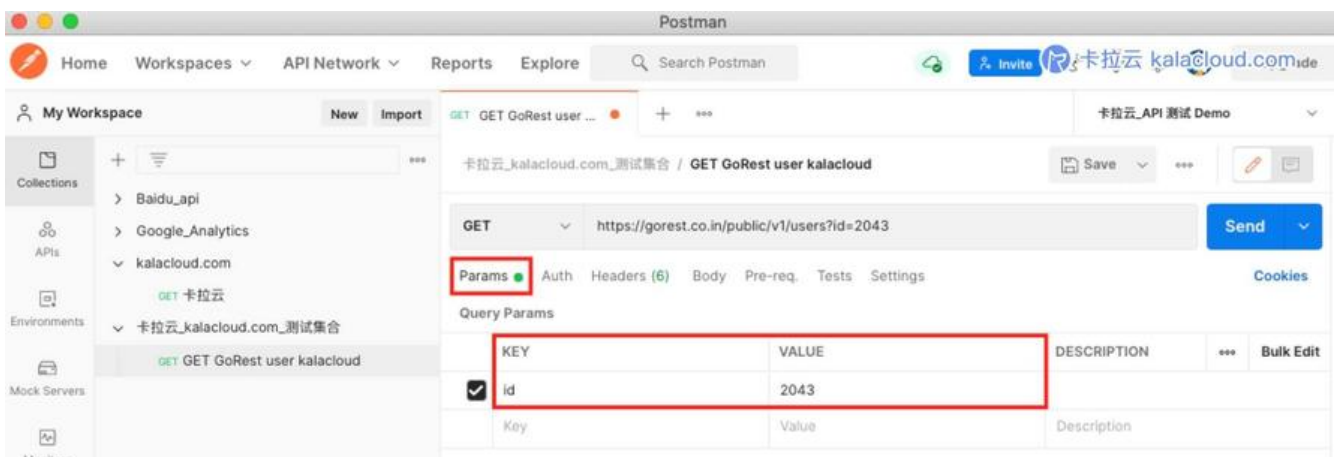
在 GoRest API 设计中 GET 请求无需鉴权（下一节 POST 请求会讲解 API 鉴权问题），所有我们直接点击「Send」即可远程调取服务器信息。



点击「Send」，我们可以看到下方的 Body 显示了 GET API 返回值，第一段落是 users 信息概览，面是一组组 user 信息。

如果我们只想看调取其中一位用户的信息应该怎么办呢？我们可以在 API URL 中带上参数。

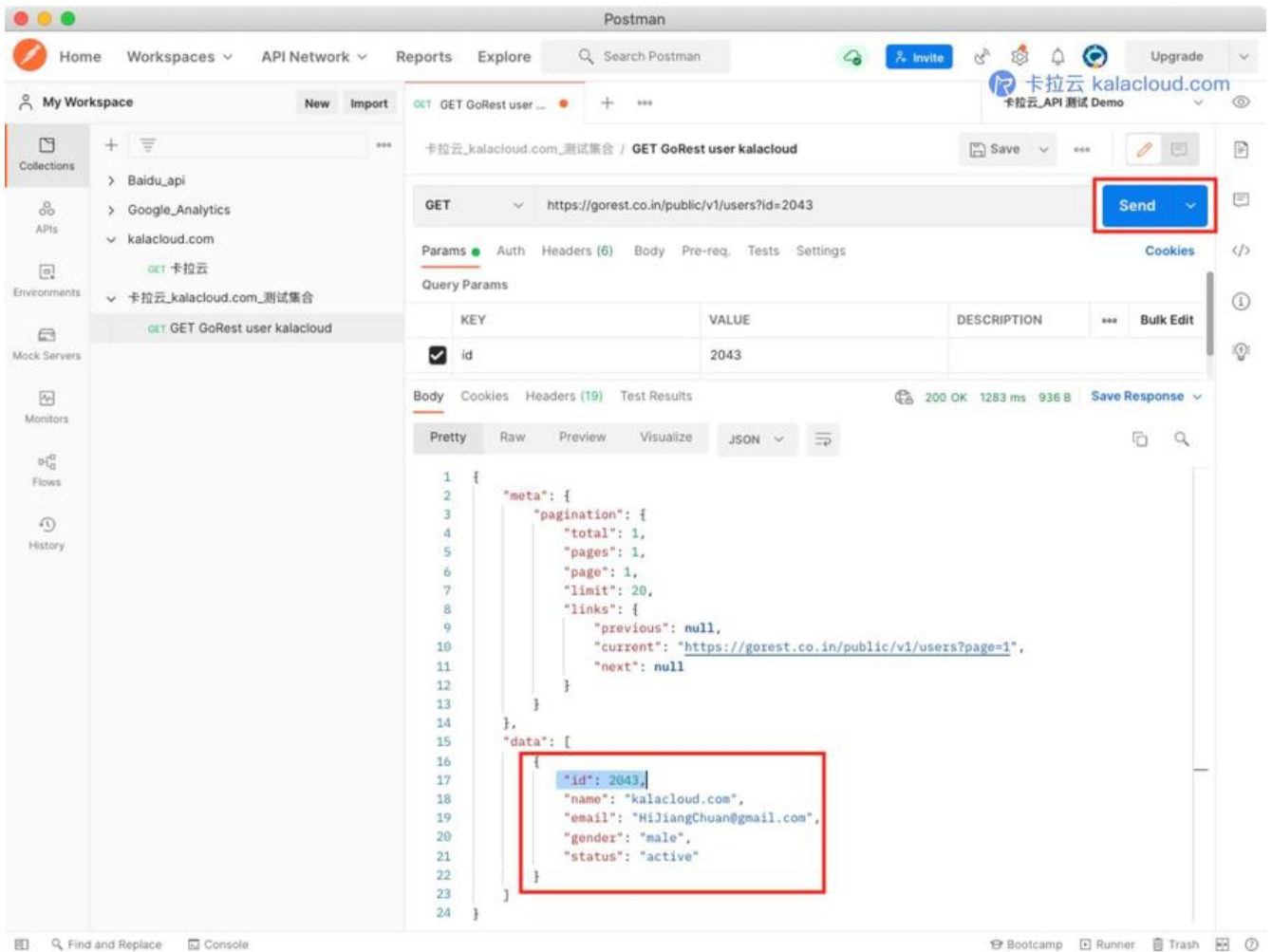
2. 带参数的 GET 请求



如果我们想查询 ID 为 2043 的用户信息，我们只需要在请求页面中的 Params(参数) 标签页的 KEY - VALUE 内填写对应的参数即可，之后 Postman 会自动在 API URL 中生成你填写的参数，使 URL 带参数 GET 请求。

<https://gorest.co.in/public/v1/users?id=2043>

设置完成后, 点击 [Send]



我们可以看到, 返回值中仅包含我们请求的 user id 为 2043 的用户信息。

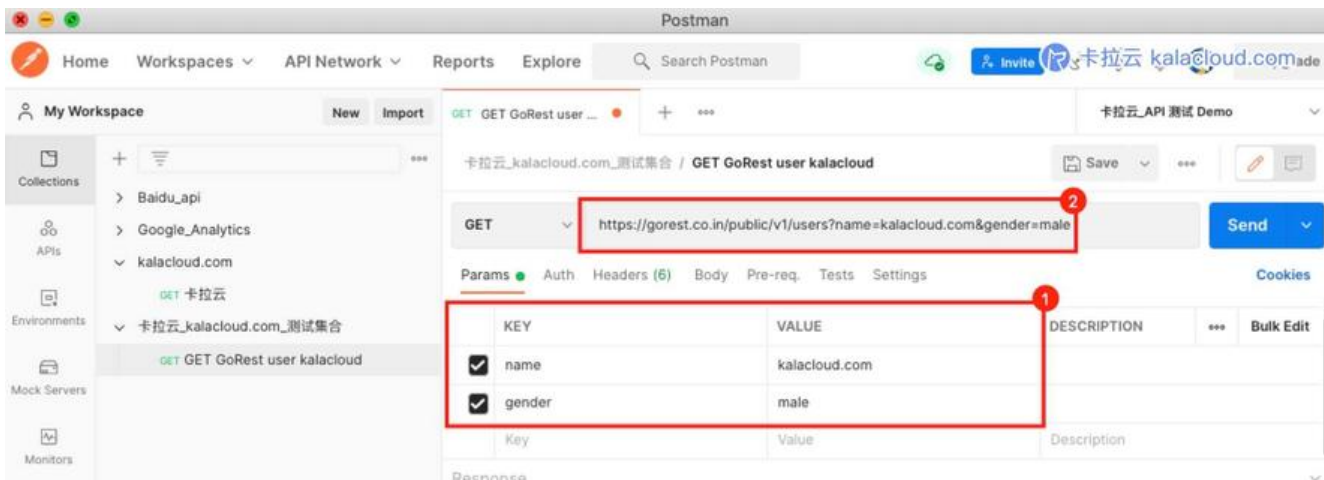
3. GET 请求中的多条件查询

有时, 我们需要使用 API 进行多条件查询操作, 比如想找 `name` 值为 `kalacloud.com`, 同时 `gender` 值为 `male` 的用户。(特别提示: 此格式是通用写法, 但最终要看 API 的开发者如何约定调用方式)

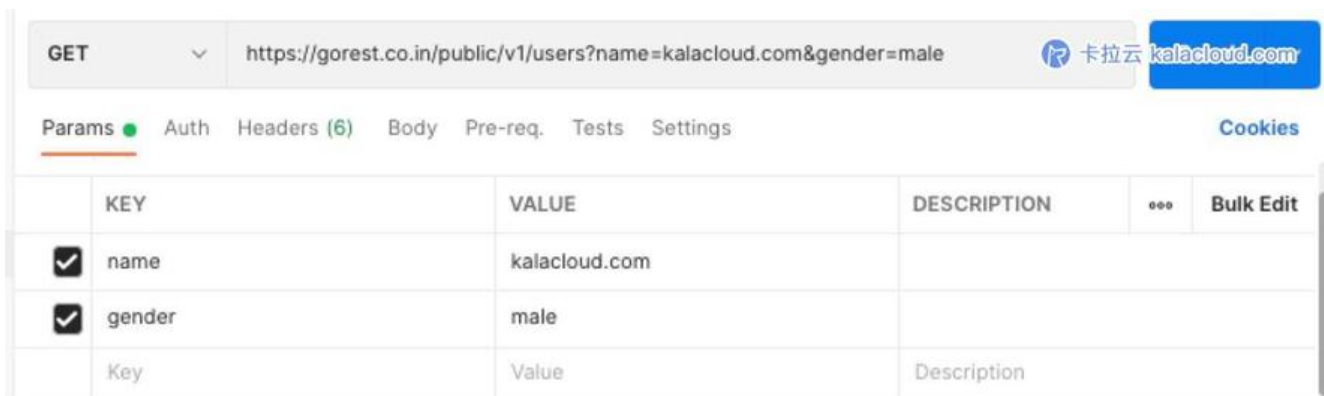
BaseURL + ResourceName + ? + key1 = value1 + & + key 2 = value 2

主 URL 之后使用 ? 连接参数, 参数与参数之间使用 & 连接符连接。

<https://gorest.co.in/public-api/users/?name=kalacloud.com&gender=male>



当然，我们可以直接在 Postman 的 Params 中直接填写 KEY - VALUE



让 Postman 帮我们生成，然后点击「Send」

GET GoRest user ... 卡拉云_API 测试 Demo

卡拉云_kalacloud.com_测试集合 / GET GoRest user kalacloud Save 卡拉云 kalacloud.com

GET https://gorest.co.in/public/v1/users?name=kalacloud.com&gender=male Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> name	kalacloud.com			
<input checked="" type="checkbox"/> gender	male			
Key	Value	Description		

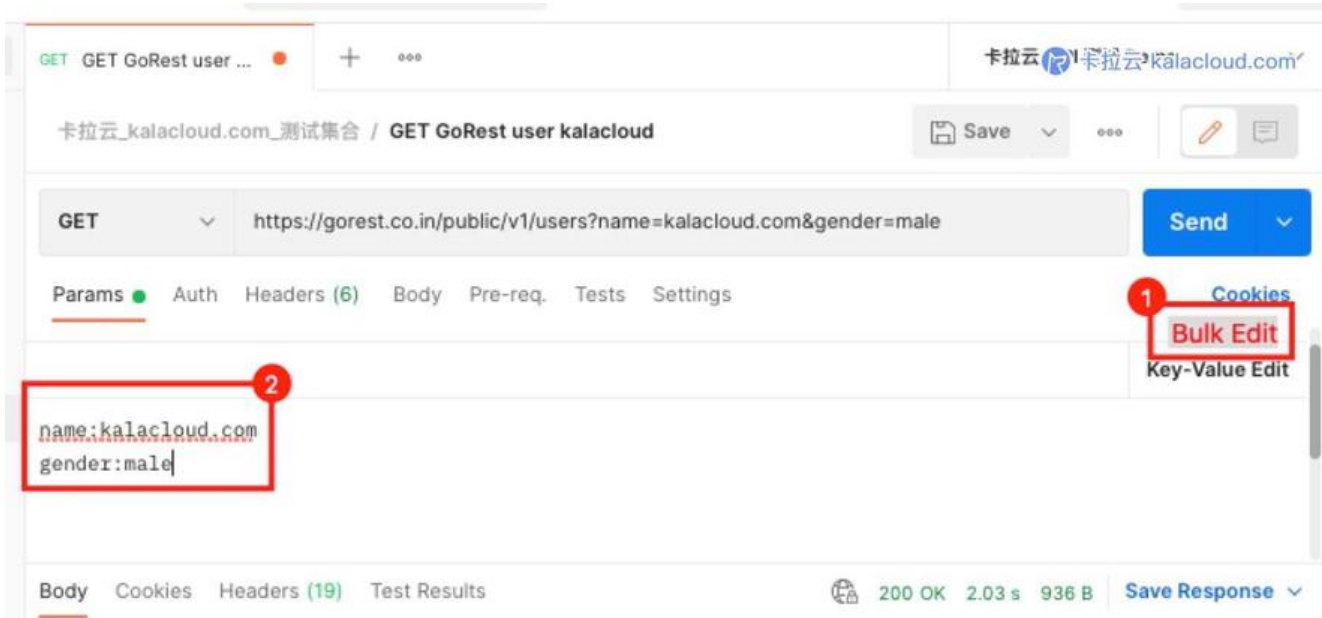
Body Cookies Headers (19) Test Results 200 OK 2.03 s 936 B Save Response

Pretty Raw Preview Visualize JSON

```
3     "pagination": {
4       "total": 1,
5       "pages": 1,
6       "page": 1,
7       "limit": 20,
8       "links": {
9         "previous": null,
10        "current": "https://gorest.co.in/public/v1/users?page=1",
11        "next": null
12      }
13    },
14  },
15  "data": [
16    {
17      "id": 2043,
18      "name": "kalacloud.com",
19      "email": "HiJiangChuan@gmail.com",
20      "gender": "male",
21      "status": "active"
22    }
23  ]
24 }
```

可以看到 API GET 调取了我们设定的两个 VALUE 值的 data 信息。

特别提示，你可以点击右上角的「Bulk Edit」进行参数的批量编辑



Params 批量编辑模式。

四. 在 Postman 中发送 POST 请求

这一节我们讲 POST 请求，POST 请求和 GET 请求最大的区别是 GET 请求仅使用只读形式读取数据而 POST 请求会修改服务器中的数据，比如创建新用户，创建用户信息，上传图片等操作都是用 POST 完成的。

1. POST 请求前，使用 Postman 对 API 鉴权

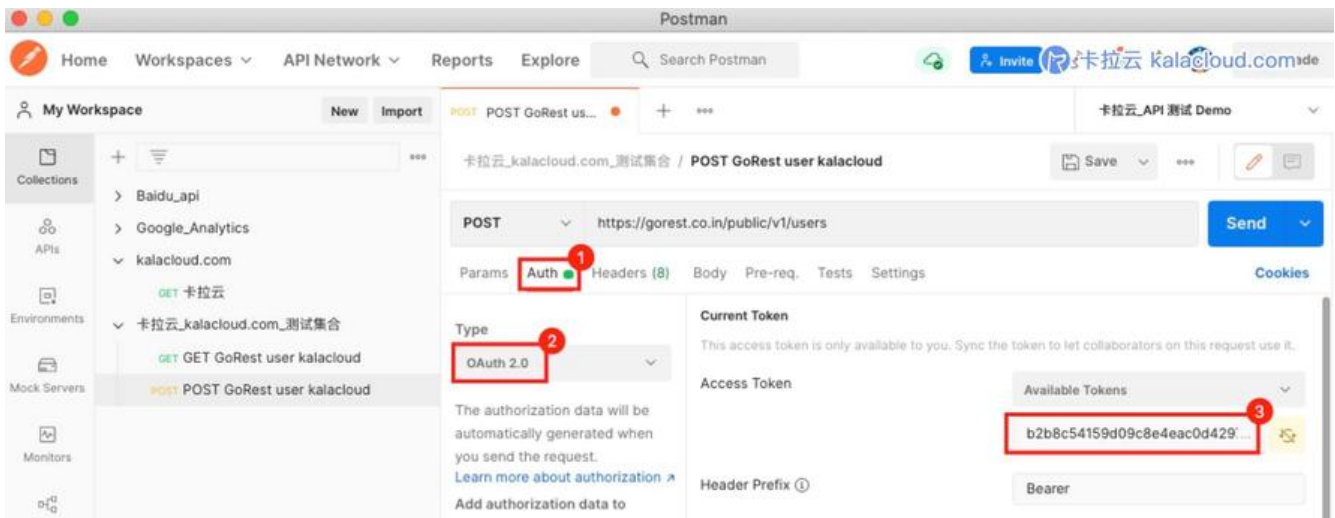
POST 不再仅仅是读取数据，会涉及到对数据的写入，这种敏感的操作，一定会涉及到账号鉴权操作。

GoRest 模拟 API 工具使用 HTTP Basic Authentication 鉴权方式



这种鉴权方式可以直接放在 Headers 中以 Key - Value 的形式进行验证，本教程演示稍微复杂点的 OAuth2 鉴权方式，好让大家学到更多。

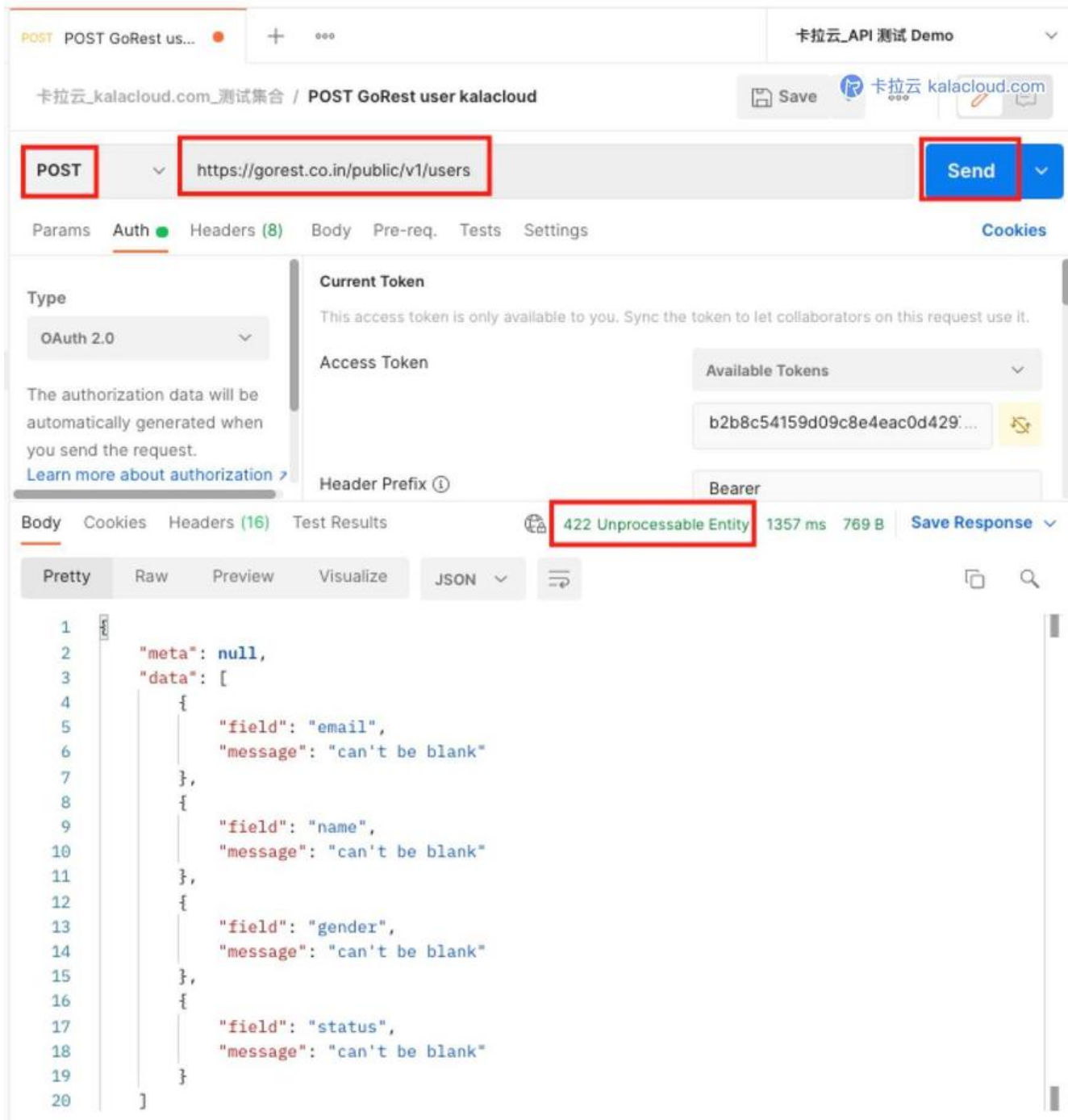
首先在 GoRest 注册账号，然后点击顶部的「Rest Console」进入控制页，我们可以在这个页面获得自己的 Auth Token，这段就是 API 用于鉴权的密钥。



在 Postman 请求页的 Auth 标签中，选择鉴权类型 (Type) 为 OAuth 2.0，然后在 Access Token 填写网页上的这段 Value 后，在请求类型中选择 POST，输入 API 请求地址

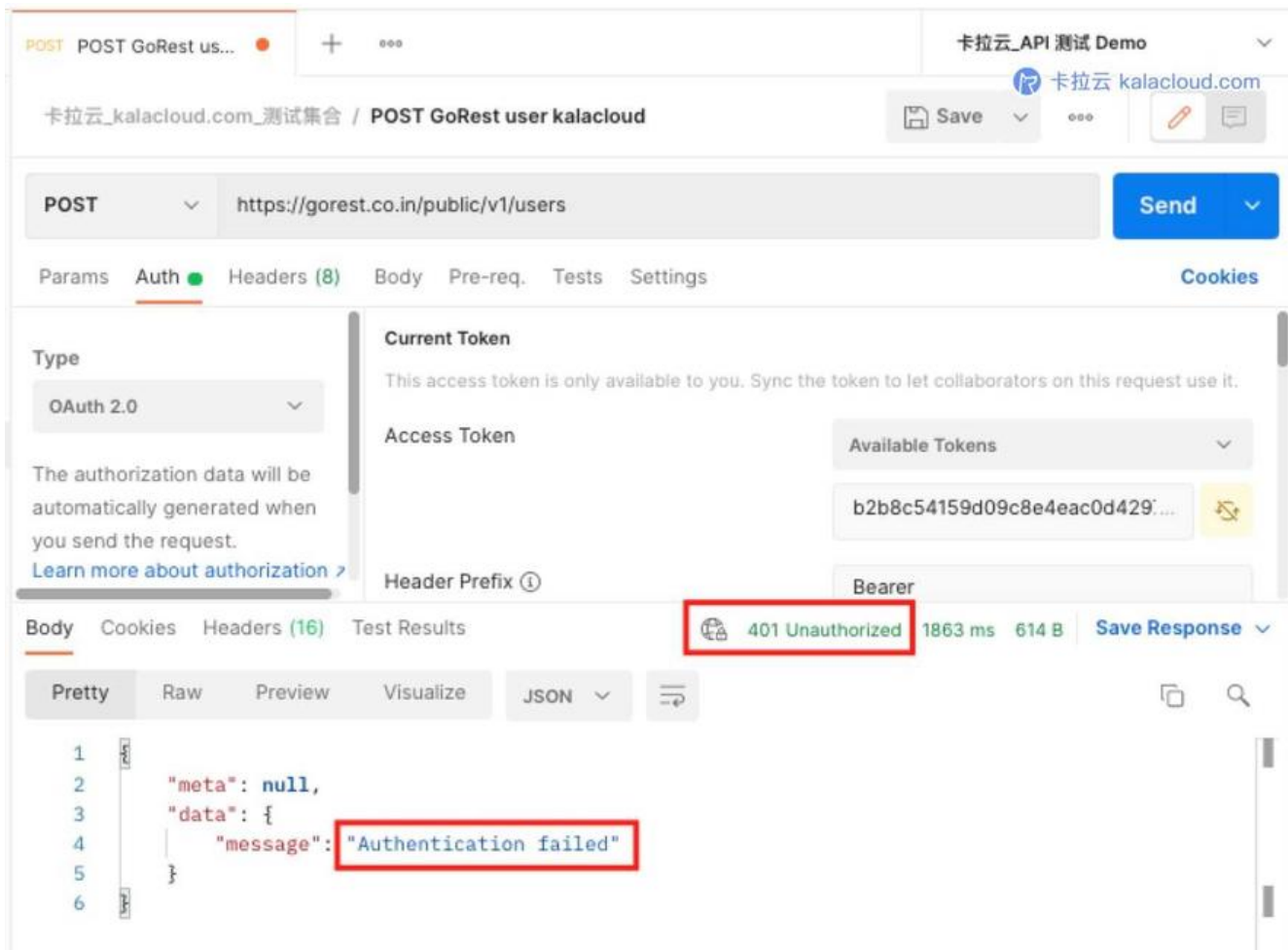
<https://gorest.co.in/public/v1/users>

然后点击 [Send]



如果 HTTP 响应代码返回 422，表示鉴权成功，只是没有填写数据。

特别提示： 在本案例中使用的GoRest API 是先检测 Token 是否正确，之后在检测 body 等信息。有先检测提交信息是否正确再鉴权的 API，这取决于 API 的开发怎么设定，那么这种情况 422 就代表鉴权成功。所以谨慎起见，所有 4XX 开头的响应代码，你都可以理解为出现了错误。



如果返回 401 说明鉴权验证失败 (Authentication failed) 你的 Token 可能复制错了, 请检查后再。

鉴权成功后, 下一节, 我们通过 API POST 请求将第一组数据写入服务器数据库。

(2) 用 Postman 发出第一个 POST 请求

上一节, 我们通过 API Token 鉴权成功, 下面我们在请求页设置 POST 请求信息。

1.请求命令选择 POST, 然后在地址栏填写 API 地址:

<https://gorest.co.in/public/v1/users>

2.在「Auth」标签页根据本文上一小节设置 API 鉴权 Access Token

3.在 Body 中选择 raw 然后选择 JSON 格式。

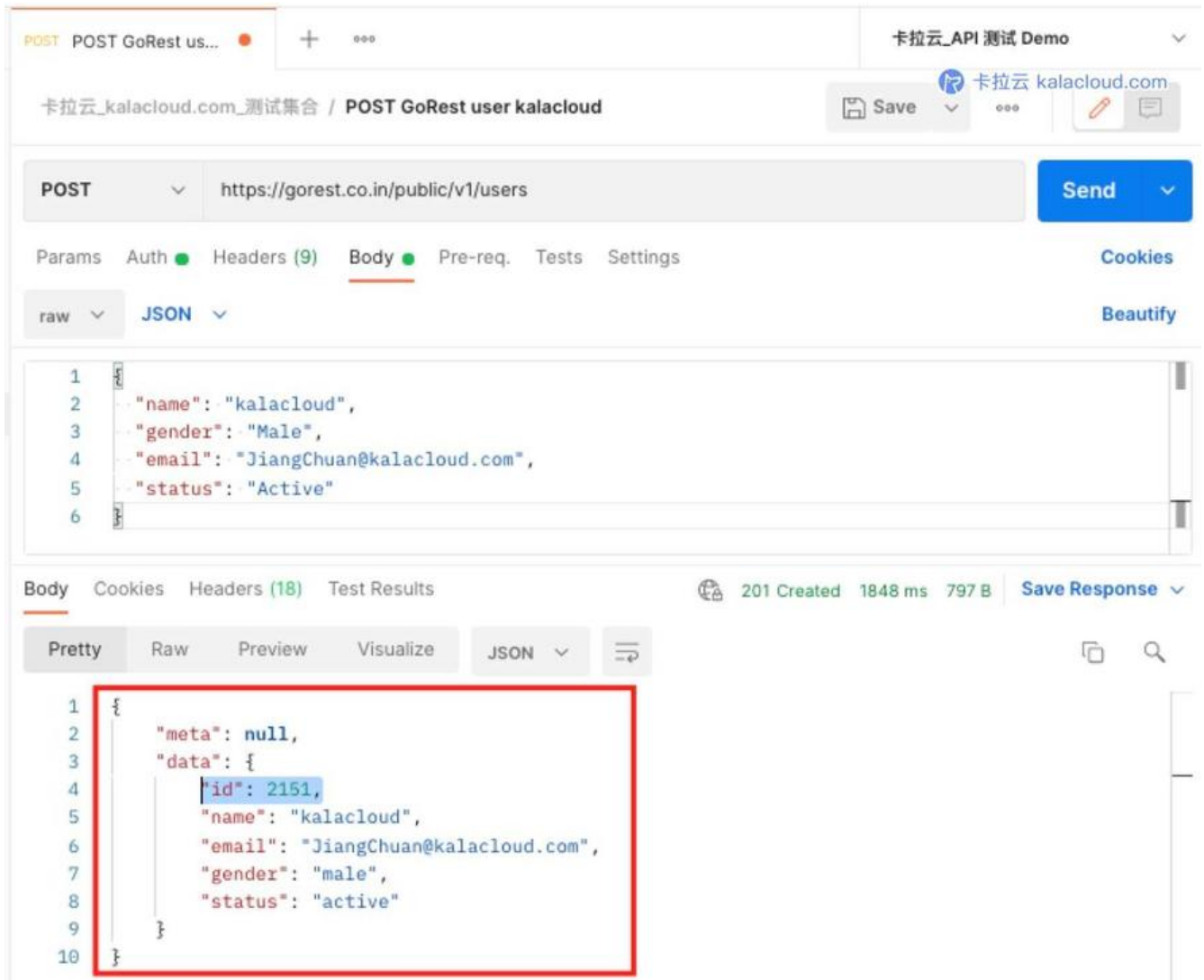
4.将下列 JSON 代码填入编辑框, 此为本次 POST 发送给服务器的内容。

5.我们将注册一个新用户, 用户信息包含在这个 JSON 文本里。

```
{  
  "name": "kalacloud",  
  "gender": "Male",  
  "email": "JiangChuan@kalacloud.com",  
}
```

```
"status": "Active"
}
```

最后，点击「Send」，发送 POST 请求。



我们可以看到 API 返回信息，已经在服务器中注册好新用户，新用户 ID 为 2151，并一起返回了我刚刚提交的用户注册信息。

如果你也看到了类似的返回结果，那么恭喜，你的第一个 POST 请求被服务器成功接受，你通过 API 服务器上注册了一个新用户。

扩展阅读：[最好的 6 个免费天气 API 接口对比测评](#)

五. 用 Postman 发送第一个 PUT 更新请求

PUT 请求一般用于更新服务器已有资源，如果服务器中没有对应的资源，那么 PUT 会创建相应的资源（特别提醒：虽然 PUT 有创建新资源的功能，但是否能创建成功，最终取决于你调用的 API 是否支持此功能）

打开你的 Postman 我们来创建一个 PUT 请求。

- 点击「+」号，新建一个请求页

- 请求类型选择 [PUT]

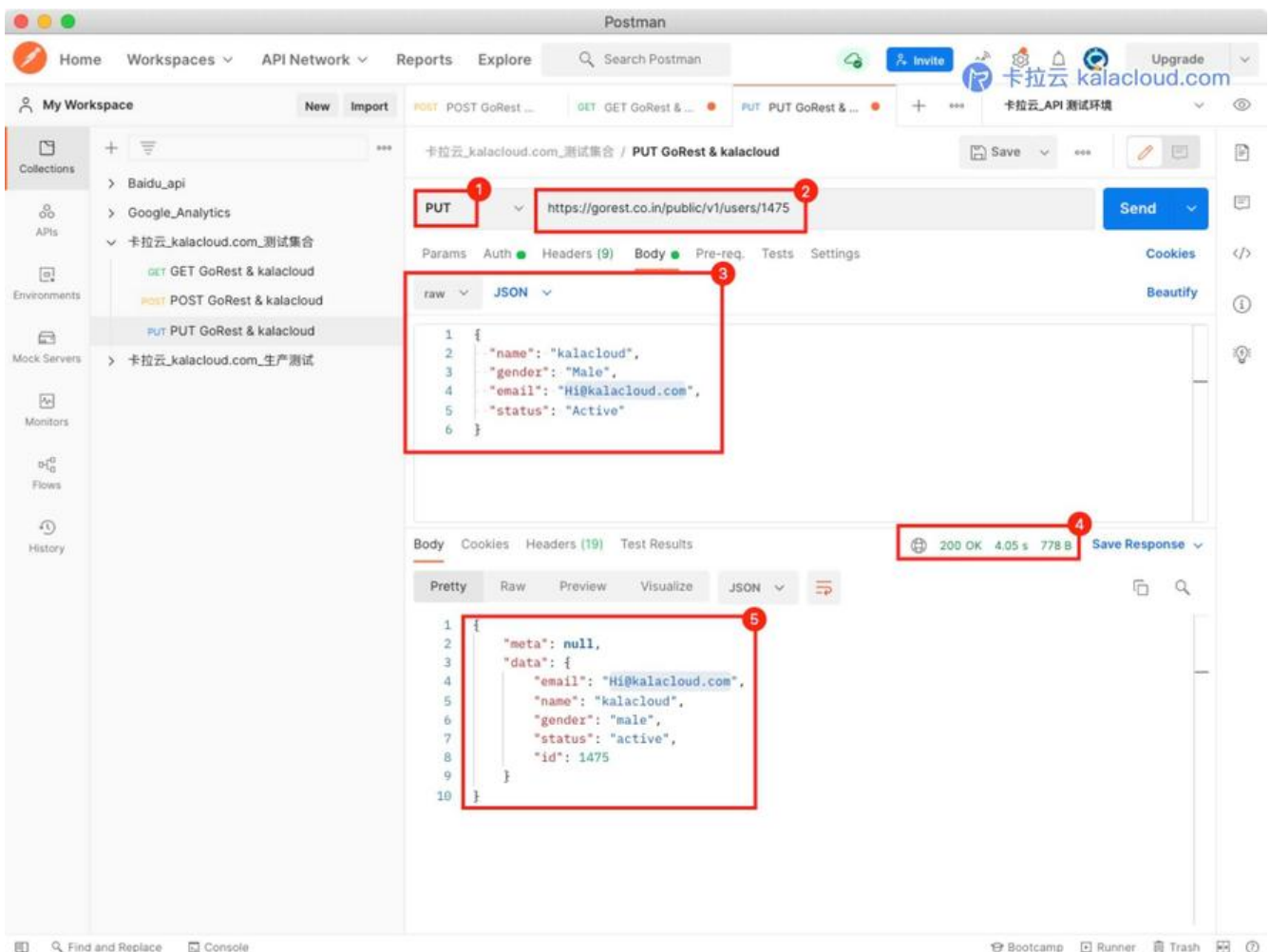
● 根据第四节我们使用 POST 请求创建的资源 ID 为 1475，又根据 GoRest API 的文档得知，修资源的 API 地址为 <https://gorest.co.in/public/v1/users/{{ID}}>，所以我们要使用 PUT 修改 ID 为 1475 资源的请求地址应该写：

<https://gorest.co.in/public/v1/users/1475>

- 选择在 Body 标签中填写 JSON 格式的资源修改信息。我们将 1475 中的邮箱由 JiangChuan@kacaloud.com 修改为 Hi@kacaloud.com，所以我们在 Body 中填写以下代码。

```
{  
  "name": "kalacloud",  
  "gender": "Male",  
  "email": "Hi@kalacloud.com",  
  "status": "Active"  
}
```

- 选择 Auth 标签，进行 API 鉴权，鉴权方法详见本文第四节《使用 Postman 对 API 鉴权》
- 点击 [Send] 发送 PUT 请求



- 如上图所示，可以看到红4位置 响应代码返回 200，这说明 PUT 请求已经执行成功。

- 返回的 Body 信息中，email 字段已经更新为 Hi@kacaloud.com

六. 用 Postman 发送第一个 PATCH 更新请求

PATCH 请求一般用于服务器资源的部分更新，它相对于 PUT 提交的数据更少，不用提整个数据，只要提交需要修改的字段即可。有关 PUT 和 PATCH 的更多区别，可查看本文第七节。

打开你的 Postman 我们来创建一个 PATCH 请求。

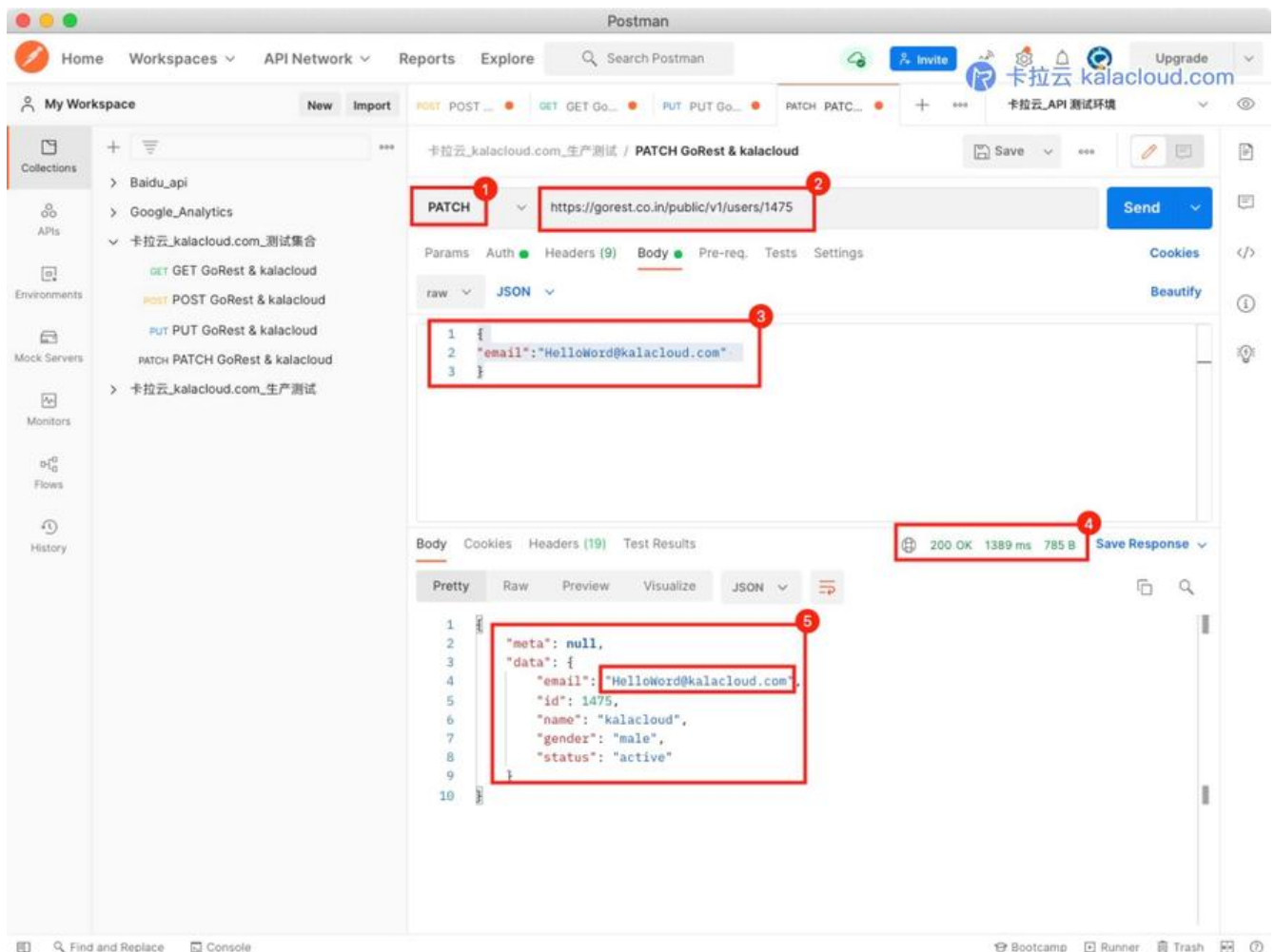
- 点击「+」号，新建一个请求页
- 请求类型选择「PATCH」
- 根据第四节我们使用 POST 请求创建的资源 ID 为 1475，又根据 GoRest API 的文档得知，修资源的 API 地址为 <https://gorest.co.in/public/v1/users/{{ID}}>，所以我们要使用 PATCH 修改 ID 1475 资源的请求地址应该写，到这里都和 PUT 请求修改资源的方法一样。

<https://gorest.co.in/public/v1/users/1475>

选择在 Body 标签中填写 JSON 格式的资源修改信息。上一节我们已经将 ID 为 1475 资源的邮箱改为 i@kalacloud.com，接着我们用 PATCH 请求把这个邮箱改为 [HelloWord@kalacloud.com](mailto>HelloWord@kalacloud.com)

```
{  
  "email": "HelloWord@kalacloud.com"  
}
```

- 选择 Auth 标签，进行 API 鉴权，鉴权方法详见本文第四节《使用 Postman 对 API 鉴权》
- 点击「Send」发送 PATCH 请求



- 如上图所示，可以看到红4位置 响应代码返回 200，这说明 PATCH 请求已经执行成功。
- 返回的 Body 信息中，email 字段已经更新为 `HelloWord@kalacloud.com`

七. PUT 和 PATCH 的区别

在 HTTP 协议中，PUT 和 PATCH 都是用于更新服务器资源的命令，但他们有着不同的格式和用途。

PUT 请求：一般用于更新服务器已有资源，如果服务器中没有对应的资源，那么 PUT 会创建相应的源（特别提醒：虽然 PUT 有创建新资源的功能，但是否能创建最终取决于你调用的 API 是否支持此能）

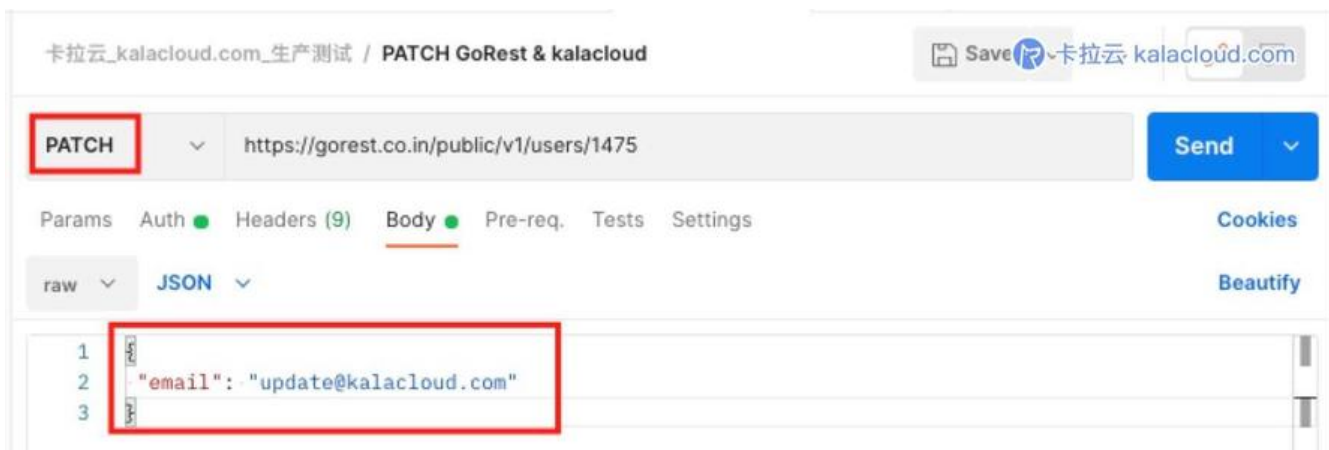
PATCH 请求：用于局部更新服务器现有资源，它不用像 PUT 更新资源中的一点点也要提交所有字段信息，PATCH 更新哪个字段就提交哪个字段的更新信息即可。

举例说明PUT 和 PATCH 的区别：

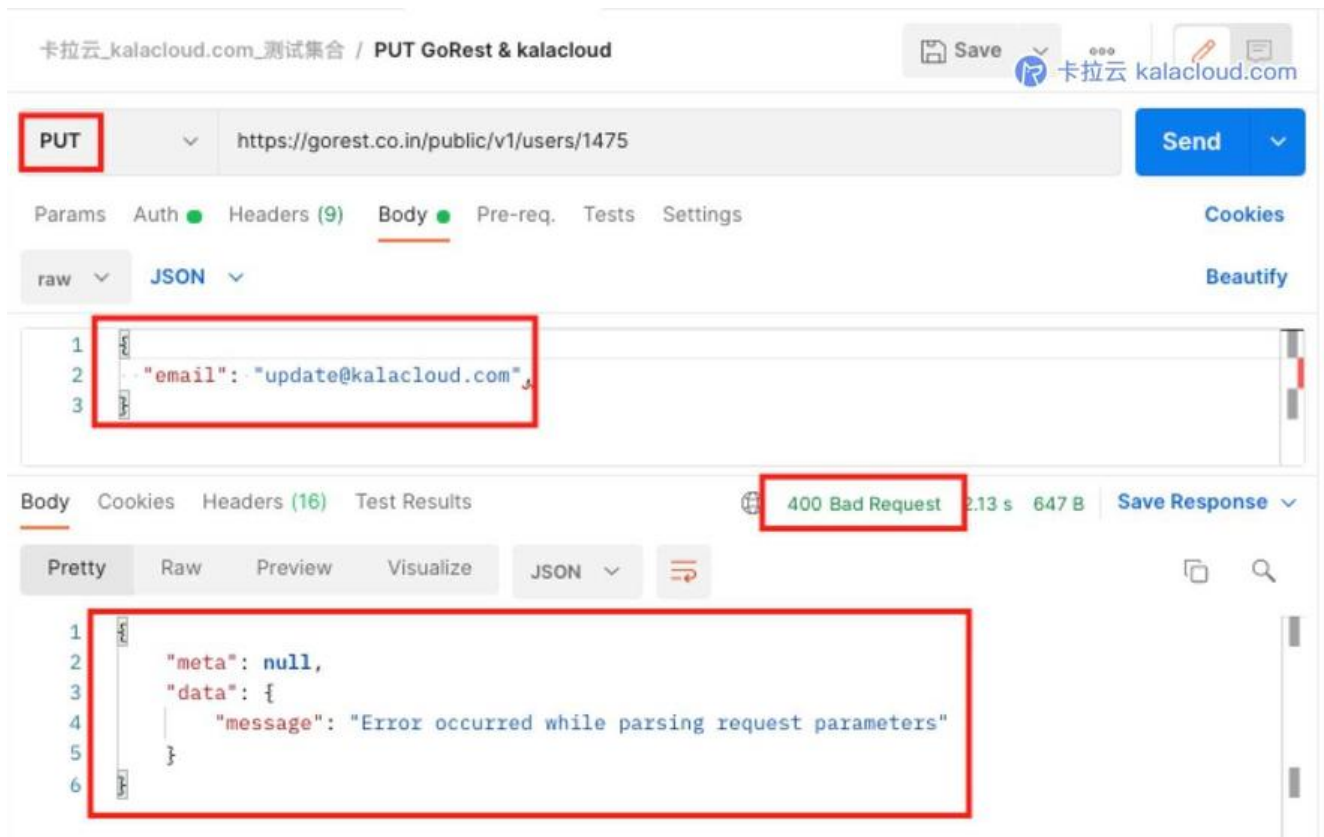


同样是更新资源中的 Email 信息，PUT 需要带上资源中的所有信息，然后在更新（上图）

而 PATCH 则仅需要提交更新部分，即仅提交邮箱信息即可（下图）



那么，如果 PUT 像 PATCH 一样仅提交资源的局部信息会发生什么呢？会 400 报错。



PUT 不论修改多少，必须把修改资源的全部字段写全，否则会 400 报错。

扩展阅读：[最好用的 5 款 React 富文本编辑器](#)

八. 用 Postman 发送第一个 DELETE 删除请求

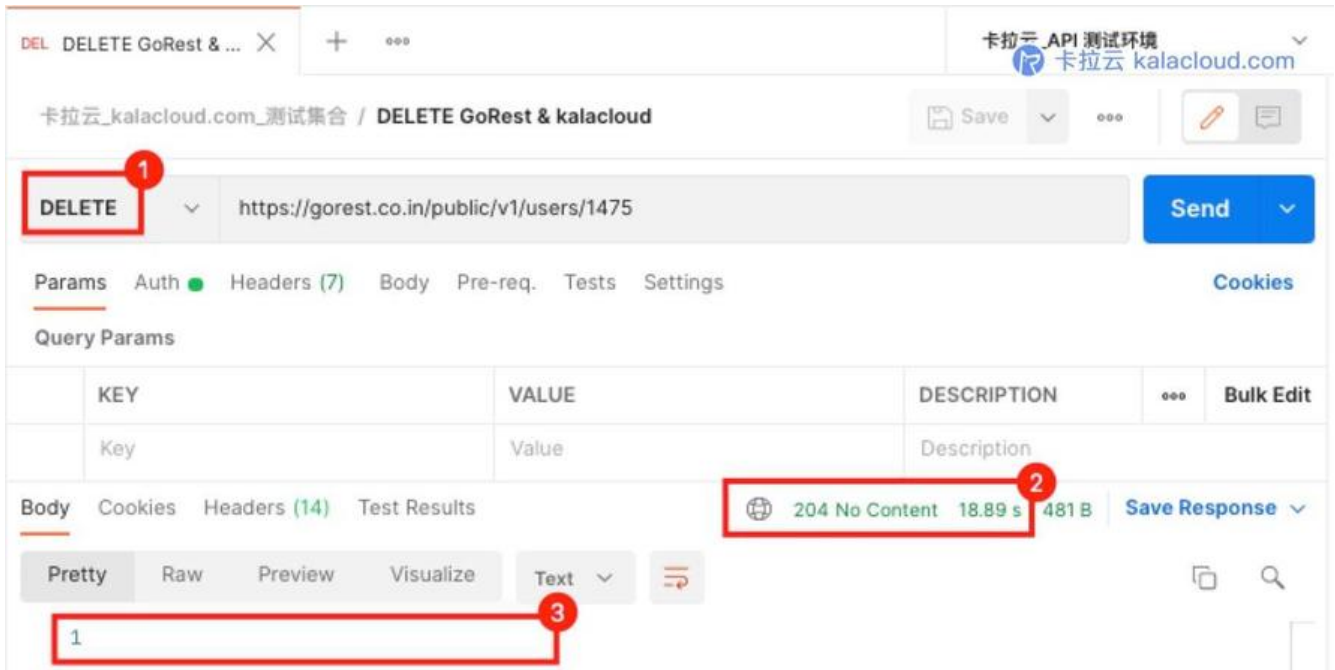
我们在上文讲了获取(GET)，创建(POST)，更新(PUT / PATCH) 请求，接着我们来说说删除(DELETE) 请求。顾名思义，DELETE 请求执行可删除整个资源。我们来直接实践一次你就明白了。

打开你的 Postman，跟随本教程一起创建一个 DELETE 请求。

- 点击「+」号，新建一个请求页
- 请求类型选择「DELETE」
- 我们来把上文刚刚创建的 ID 为 **1475** 的资源彻底删掉。根据 GoRest API 的文档得知，删除资源的 API 请求地址为 <https://gorest.co.in/public/v1/users/{{ID}}>，所以我们要使用 **PATCH** 删除 ID 1475 资源的请求地址应该写：

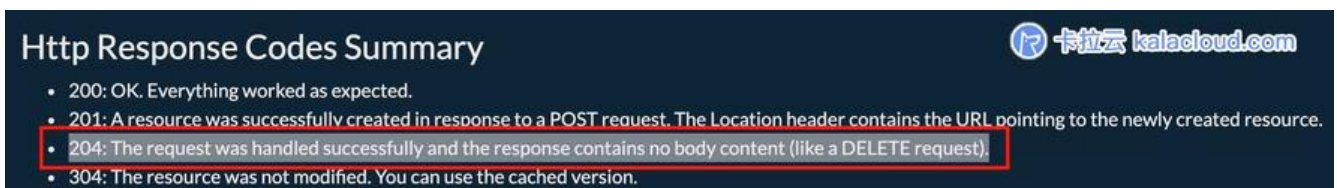
<https://gorest.co.in/public/v1/users/1475>

- 选择 Auth 标签，进行 API 鉴权，鉴权方法详见本文第四节《使用 Postman 对 API 鉴权》
- 点击「Send」发送 DELETE 请求，删除对应的资源。



如上图所示，提交 DELETE 请求后，响应代码为 204，返回的 body 为空，删除成功。

特别提示：在 GoRest 的文档说明中，特别说了 DELETE 删除返回值的状态。



API 返回状态具体是怎么样的，还要看 API 的开发者是如何约定的，并非只有返回 200 才是成功的。

扩展阅读：[PAW 使用教程 - 手把手教你 API 接口测试](#)

九. Postman 中的全局变量、环境变量、集合变量的设置

1. Postman 设置变量的意义

Postman 里有多种变量，我们可以把某些重要的值抽象出来变成变量，方便我们做场景 / 条件切换。比如，我们可以把 baseURL 抽出来，在环境变量里设置「生产环境变量」和「测试环境变量」，之，我们只需要切换标签即可快速将数据从一个环境切换到另一个环境中，非常方便。

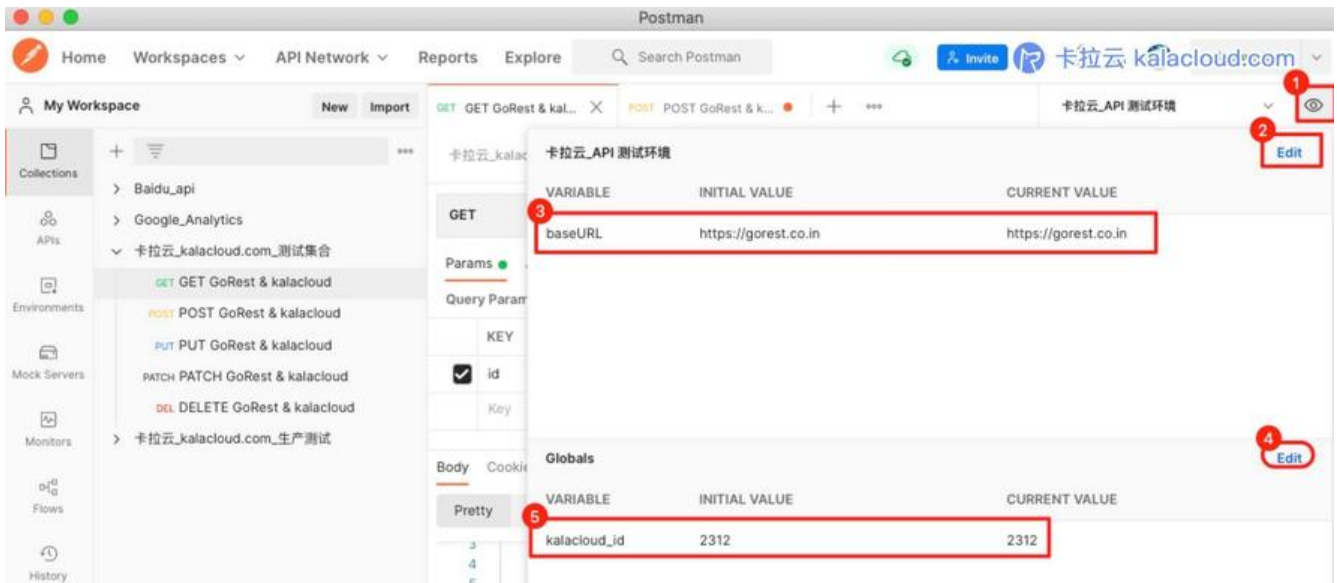
2. Postman 常用的三种变量形式

- 全局变量：全局变量一旦声明，即可应用到 Postman 中所有测试的 API 中。任何请求都可以直接用全局变量，它的作用域是全局的。
- 环境变量：Postman 的环境变量可以理解为一组选项，当这组环境变量选项被选中时，才会生效。特别适合「生产环境」和「测试环境」之间的切换等应用场景。
- 集合变量：集合变量是针对集合 (Collections) 生效的，一个集合下可能有 N 条 API 请求，集合量可以一次修改集合下的所有变量数值。

以上三种变量的作用域从大到小为 全局 > 集合 > 环境，当三个变量形式同时作用于一个 API 测试条

, Postman 会优先使用最小作用域变量。

3. 如何在 Postman 设置全局变量与环境变量



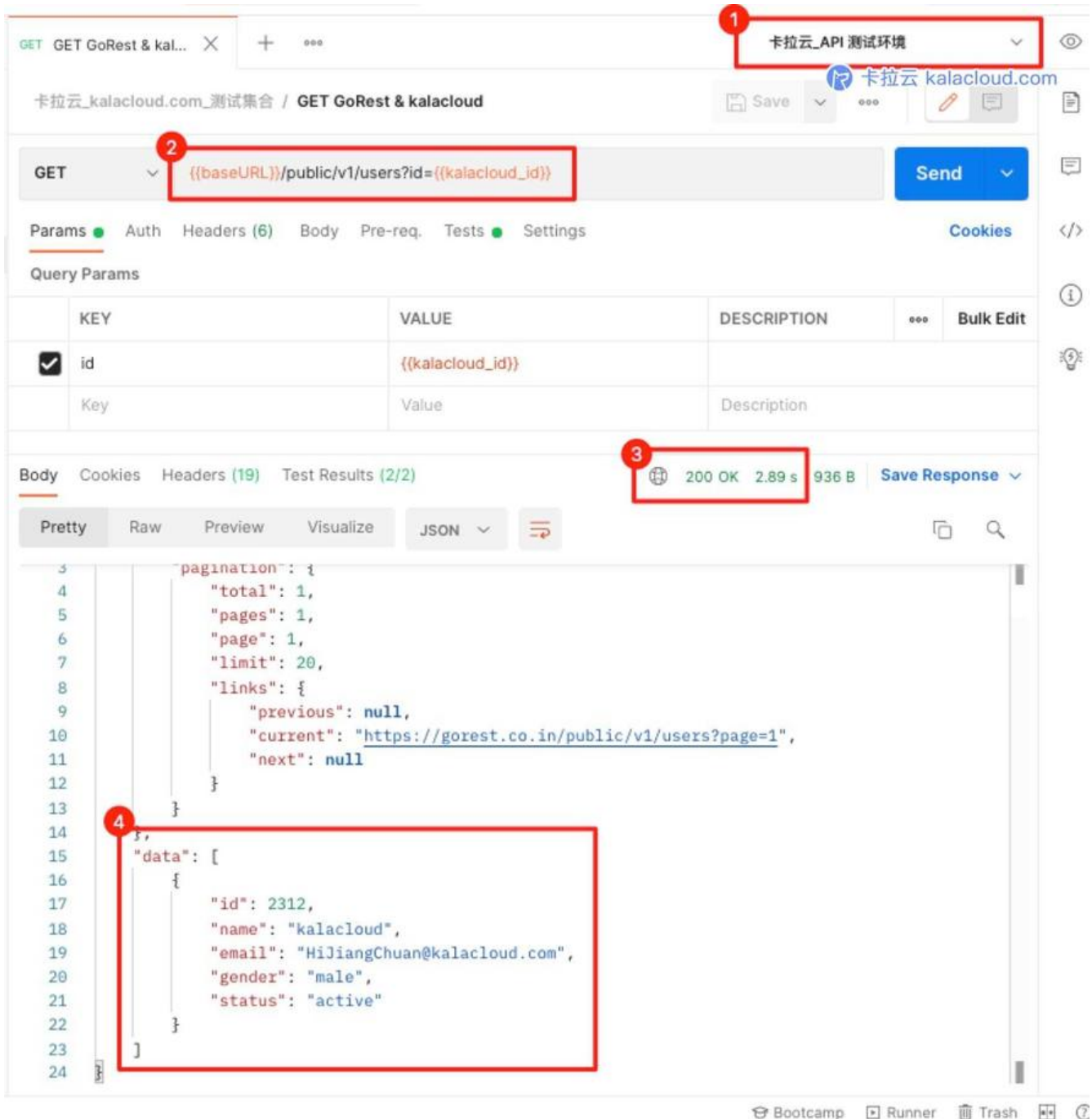
- 新建一个请求页，点击右上角的「小眼睛」进入变量设置页。
- 页面上方为「环境变量」，我们点击编辑设置环境变量名为「卡拉云_API 测试环境」
- VARIABLE 设置为 `baseUrl`，INITIAL VALUE 设置为 `https://gorest.co.in`，保存之后我们就可使用 `{{baseUrl}}` 变量来替代 API URL 了。
- 页面下方为「全局变量」，VARIABLE 设置为 `kalacloud_id`，INITIAL VALUE 设置为 `2312` (2312 为 GoRest 中的一个已存在的用户信息 ID)，保存后我们就可以使用 `{{kalacloud_id}}` 变量来替代 ID 了。

我们来一起测一下刚刚设置好的「全局变量」和「环境变量」是否生效。

- 新建一个 GET 请求页，地址栏填入：

```
{{baseUrl}}/public/v1/users?id={{kalacloud_id}}
```

- 点击「Send」



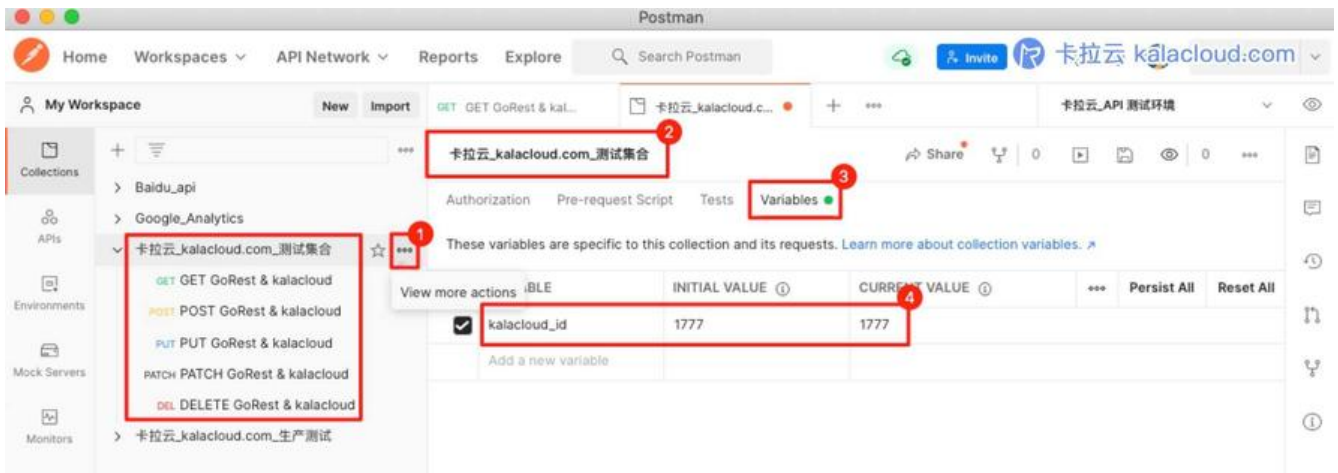
返回响应代码为 200 说明请求成功，返回的 Body 信息是 ID 为 2312 的用户信息，说明全局和环境量已生效。

4. 如何在 Postman 设置集合变量

集合变量是指应用在整个集合所有请求中的变量，集合变量优先与其他变量应用与请求，也就是说如有集合变量，那么其他变量与集合变量相冲突的化，优先执行集合变量。

集合变量很适合临时修改整个集合中的变量，来针对集合进行测试。

打开你的 Postman，我们一起操作一遍。



- 选中一个集合，点击集合标题右侧「...」选择编辑。
- 进入集合设置页，选择 Variables 设置集合变量
- 此时，整个集合下所有请求页，都应用了此集合变量。

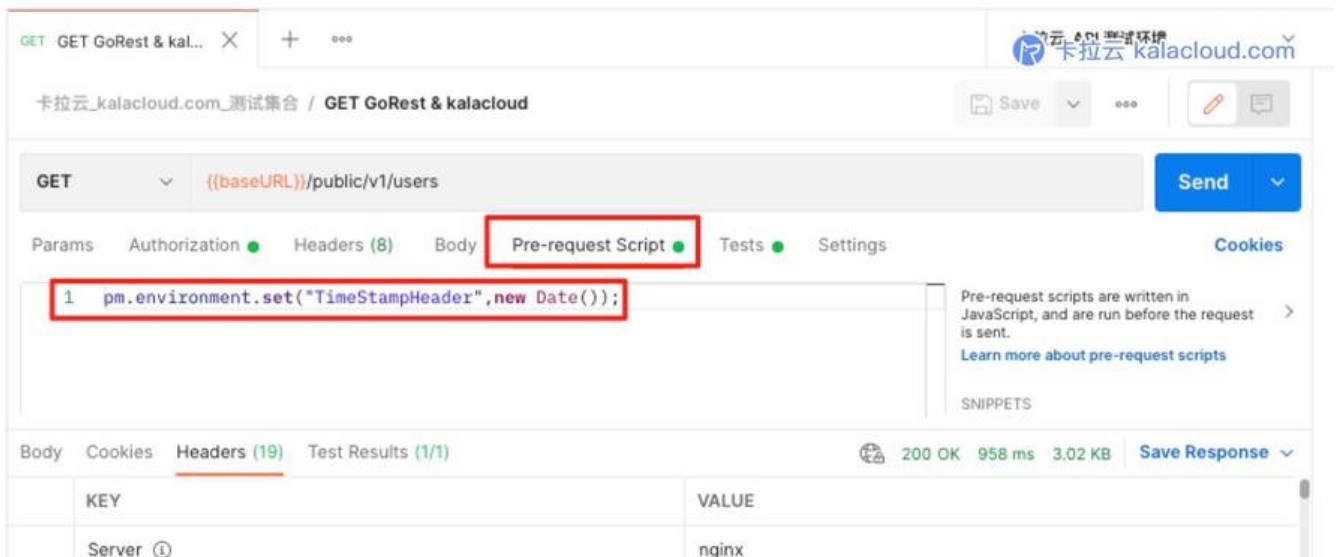
十. 如何使用 Postman Pre-request scripts 预请求脚本

Pre-request scripts 预请求脚本是在 API 请求之前执行的脚本，我们可以临时更改请求的某些变量一般预请求脚本有这么两种常见的应用场景。(1)设置动态请求头信息。(2)设置动态请求参数信息。如，当我们要请求一个与时间有关的资源时，我们可以在预请求脚本中添加 `timestamp` 字段，这是个动态值，我们可以通过前置请求脚本来实现。

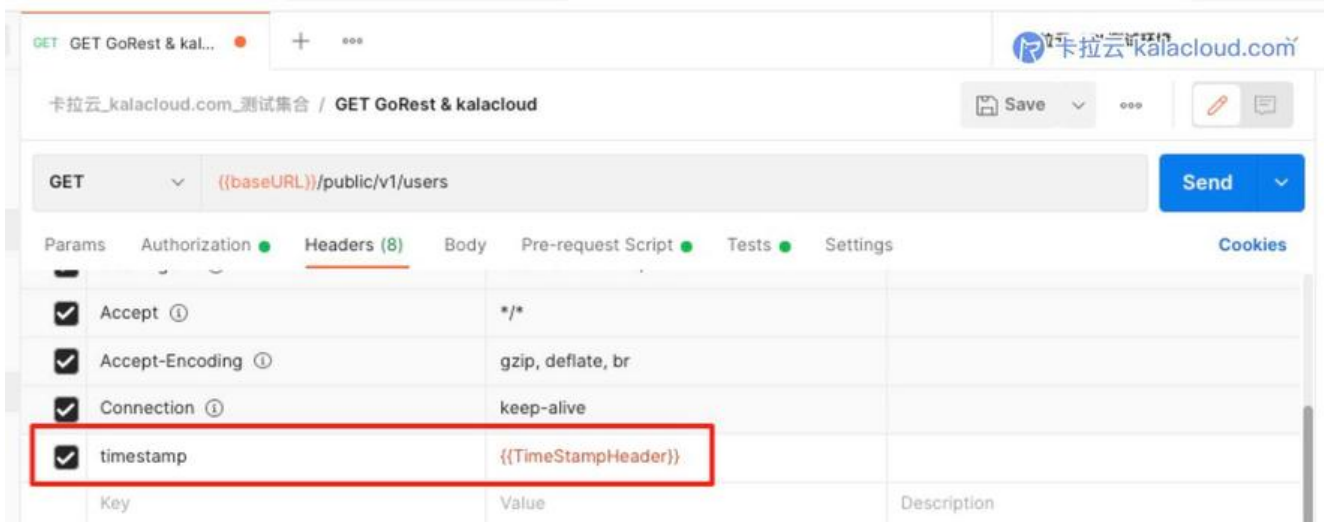
举例说明：比如我们要在 header 中包含一个时间戳，我们可以这样操作

- 在 Pre-request scripts 中添加获取时间戳的代码

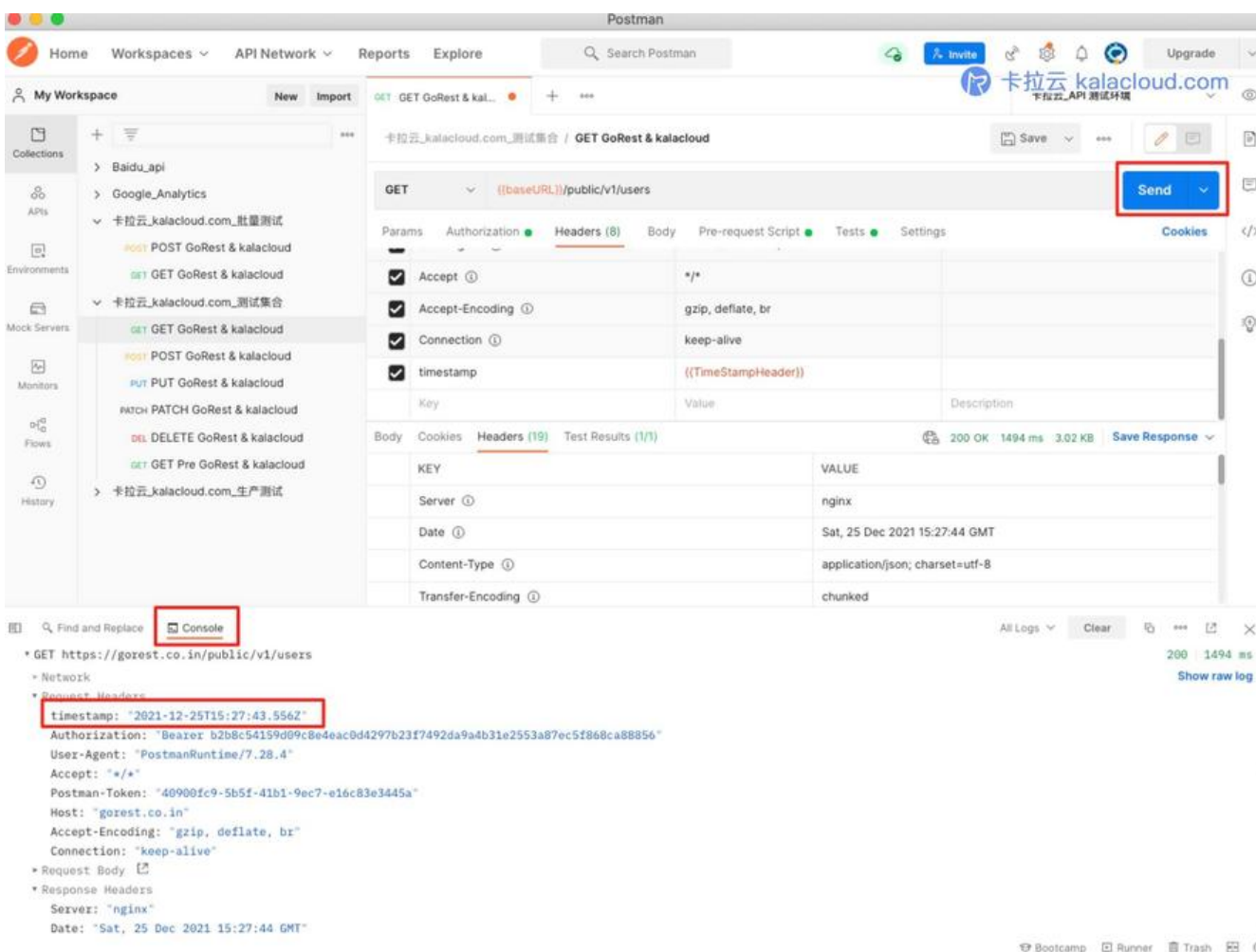
```
pm.environment.set("TimeStampHeader",new Date());
```



- 在 header 中添加预脚本中的变量 `TimeStampHeader` 当请求时，Postman 会先执行预脚本获取时间戳，然后再将时间戳赋予到 header 中 `timestamp` 值中。



● 接着我们来执行这条 GET 请求，打开控制台，在控制台中，可以看到 Request Headers 中包我们刚刚设置的时间戳「timestamp」特别提示：有关控制台的讲解，在本教程第十二节。



附：常用的 Pre-request scripts：

获取变量

//通用语法

postman.getGlobalVariable("key"); //获取全局变量

```
postman.getEnvironmentVariable("key"); //获取环境变量
```

```
//postman native app 特有语法  
pm.globals.get("key"); //获取全局变量  
pm.environment.get("key"); //获取环境变量
```

设置变量

```
//通用语法  
postman.setGlobalVariable("key","value"); //设置全局变量  
postman.setEnvironmentVariable("key","value"); //设置环境变量
```

```
//postman native app 特有语法  
pm.globals.get("key","value"); //设置全局变量  
pm.environment.get("key","value"); //设置环境变量
```

清除变量

```
//通用语法  
postman.clearGlobalVariable("key"); //清除全局变量  
postman.clearEnvironmentVariable("key"); //清除环境变量
```

```
//postman native app 特有语法  
pm.globals.unset("key"); //清除全局变量  
pm.environment.unset("key"); //清除环境变量
```

将数组、嵌套对象存储到全局&环境变量中

```
//将数组储存到环境变量中  
var array = [1, 2, 3, 4];  
postman.setEnvironmentVariable("array", JSON.stringify(array));
```

```
//将嵌套对象储存到环境变量中  
var obj = { a: [1, 2, 3, 4], b: { c: 'val' } };  
postman.setEnvironmentVariable("obj", JSON.stringify(obj));
```

```
//从环境变量中获取数组对象  
var array = JSON.parse(postman.getEnvironmentVariable("array"));
```

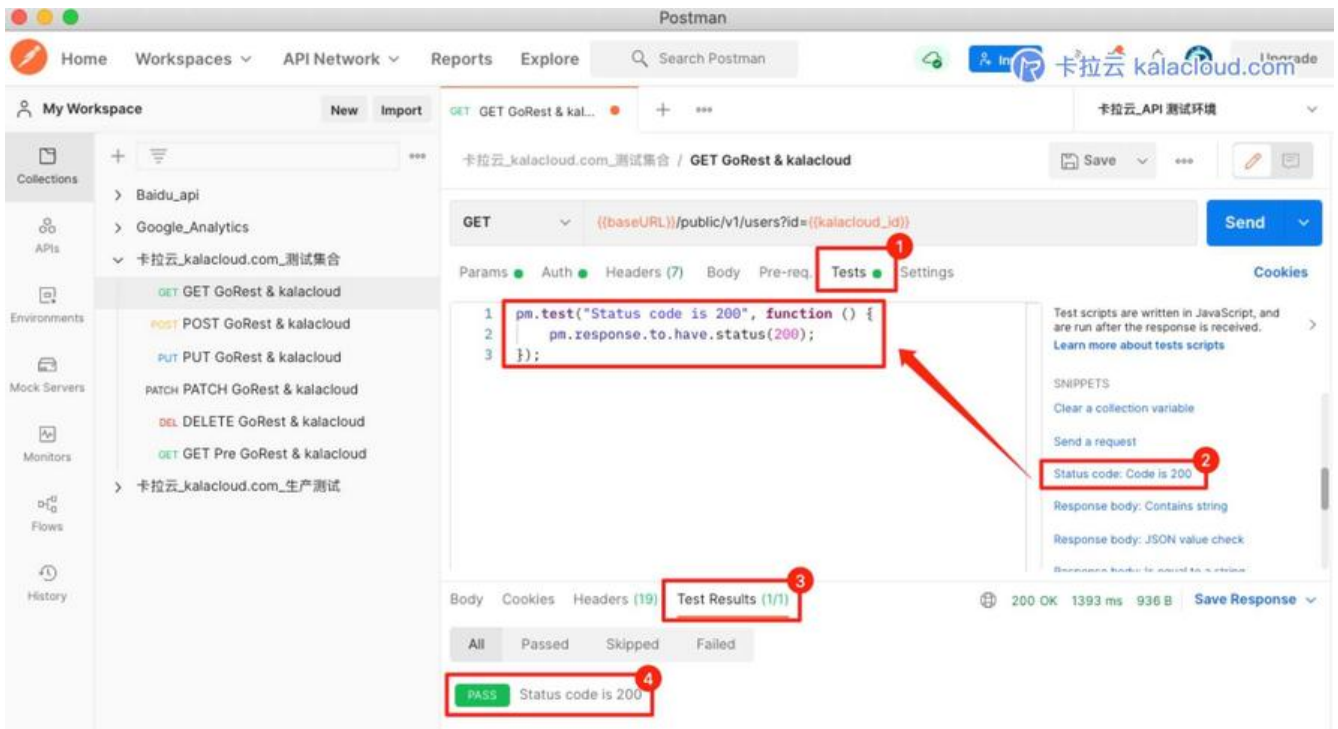
```
//从环境变量中获取嵌套对象/json对象  
var obj = JSON.parse(postman.getEnvironmentVariable("obj"));
```

十一.如何创建 Postman Tests 测试脚本 - Postman 断言

在 Postman 中 Pre-request 和 Tests 是两兄弟，一个是在调用前执行（Pre-request），一个是在用后执行（Tests），我们可以在 Tests 中使用 JavaScript 校验代码协助我们验证结果，可以说 Tests 是 Postman 的断言功能

1.Postman Tests 断言的实际应用

- Postman 状态类断言



1.我们首先创建一个 GET 请求，然后点击 Postman 中 Tests 标签，进入断言设置。

2.我们可以在右侧已经预设好的断言代码，我们先点击「Status code: Code is 200」，可以看到预的代码直接写入编辑框。这段代码的意思是，如果执行调用，服务器返回响应代码为 200 时，判断为 PASS 即调用成功。

3.点击「Send」执行 GET 请求，返回的断言可以在 Test Results 中看到结果。

4.绿色的 PASS，说明服务器返回的响应代码为 200，调用成功。

● Postman 结果比较类断言

我们再添加一条带有变量的 JavaScript 断言设置，比较预期结果和实际返回结果之间是否一致。



我们刚刚 GET 请求了 ID 2312 的用户信息，其中 name 的值为 kalacloud

那么我们接下来写一个 JS 判断预期与返回结果是否一致。即预期为 name 的值为「kalacloud」，断言自动判断返回结果的name值是否也是「kalacloud」

1.在 Tests 选项卡右侧选择「Response body:JSON value check」，我们来检测 ID 为 2312 的返回值中，name 的值是否为 kalacloud

2.我们将「Your Test Name」替换为「检查 ID 为 2312 的 name 返回值为 kalacloud」让这条测试名字直接反应出我们想测试的内容。

3.使用jsonData.data[0].name代替jsonData.value，即检测第一个返回值中的 name 的 value

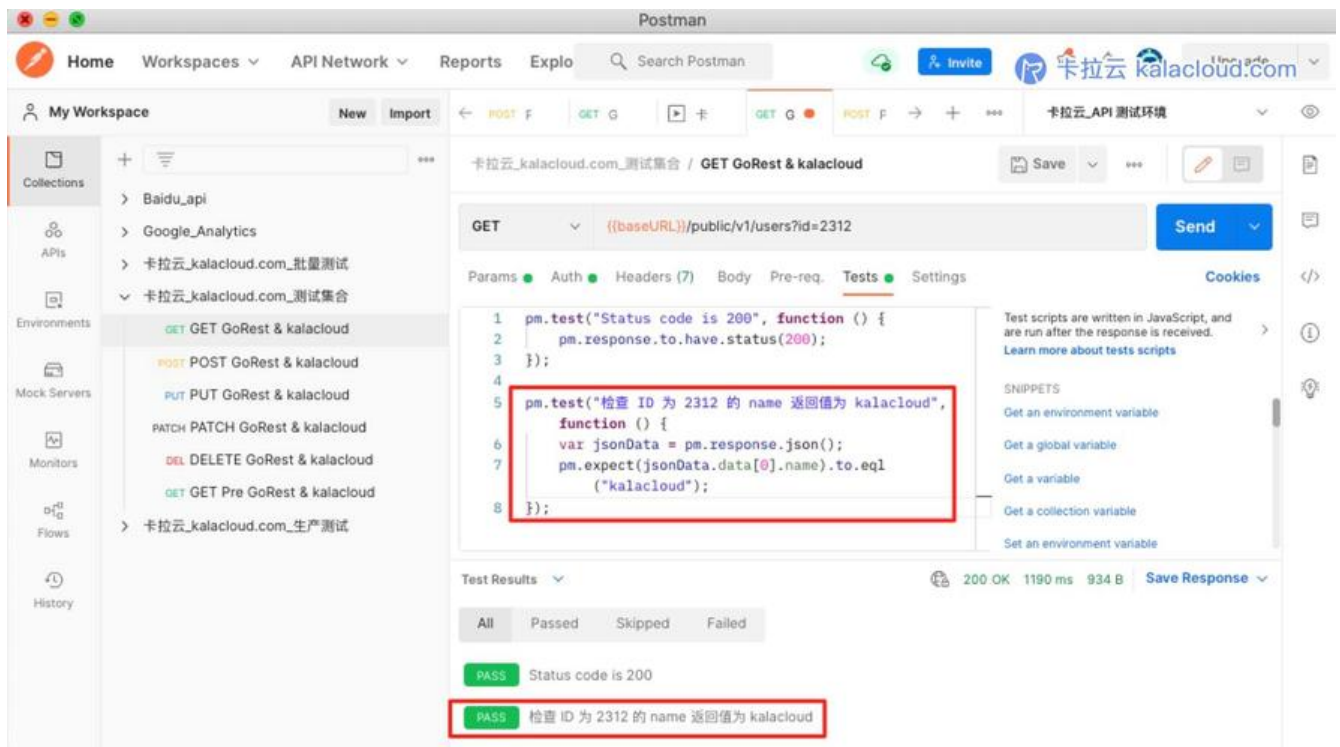
4.检测返回值：在 to.eql() 中输入待检测值 "kalacloud"，即需要检测的 text。

5.代码如下，你可以复制并根据你的情况简单修改，然后在 Postman 中，跟随教程一起测试。

特别注意： 这里的 ID = 2312 是我这里的情况，你需要根据你的情况进行相应修改。

```
pm.test("Status code is 200", function () {  
  pm.response.to.have.status(200);  
});
```

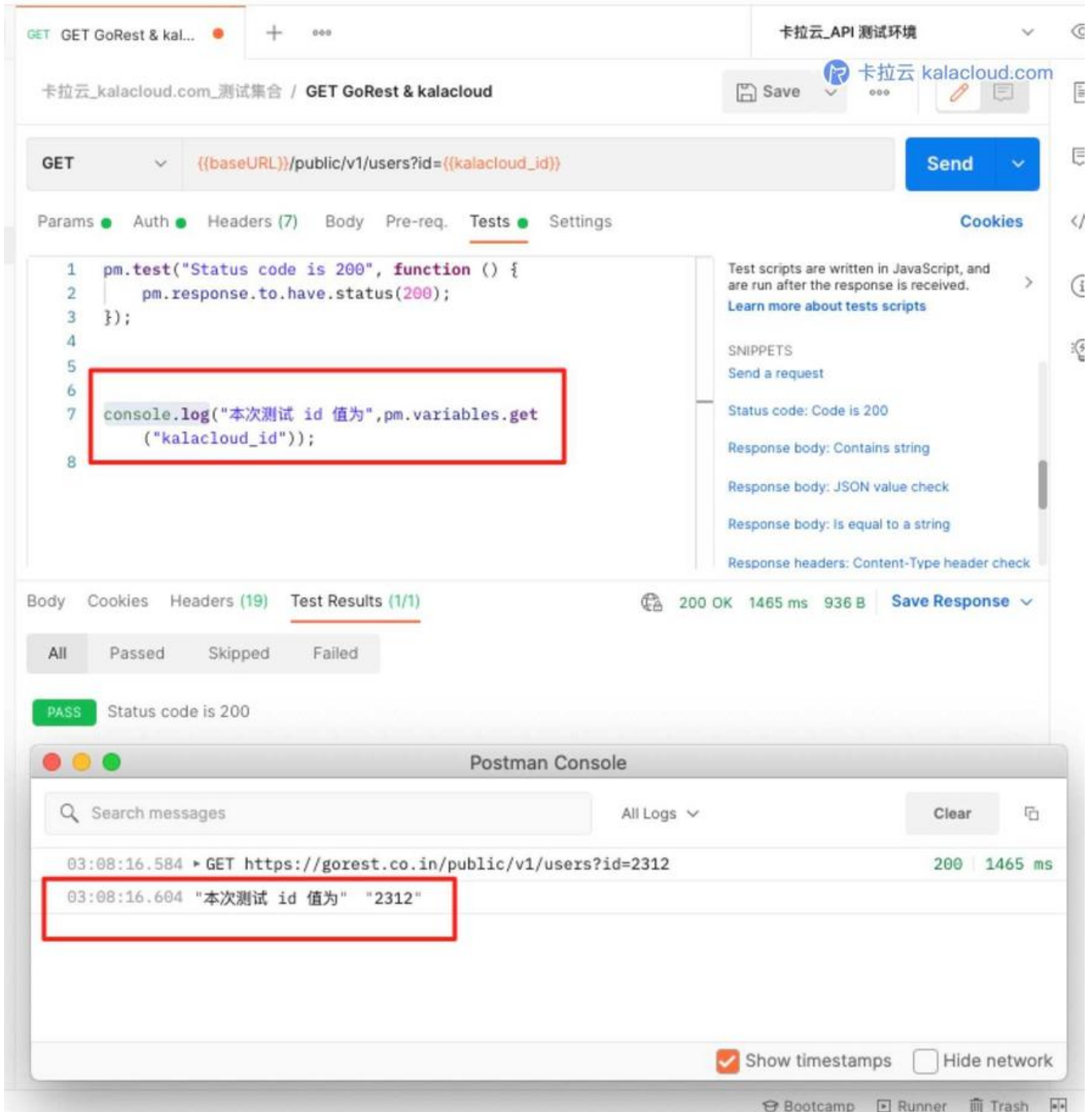
```
pm.test("检查 ID 为 2312 的 name 返回值为 kalacloud", function () {  
  var jsonData = pm.response.json();  
  pm.expect(jsonData.data[0].name).to.eql("kalacloud");  
});
```



十二. 如何在 Postman 中使用控制台

控制台可以非常直观的显示当前调用的一系列信息，我们可以在「菜单 → view → Show Postman Console」或者点击 Postman 左下角的「Console」图标，打开控制台。

我们可以在 Tests 测试脚本中加入 `console.log` 来显示我们需要在控制台显示的调用信息。



如上图，我们在 Test 脚本中加入以下代码

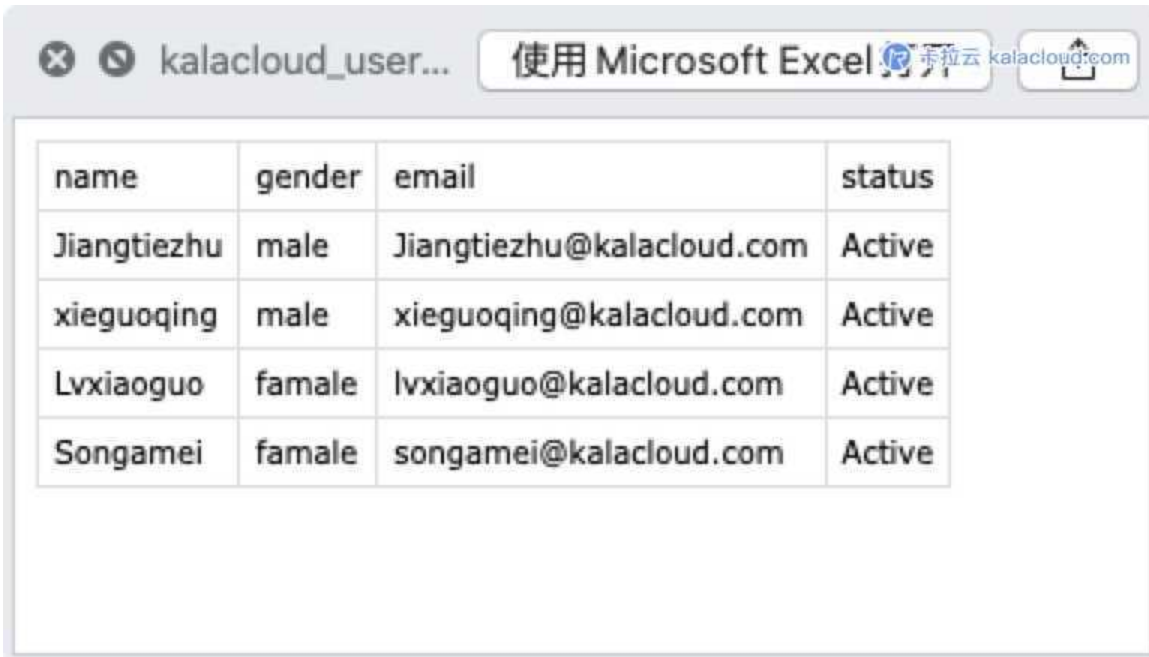
```
console.log("本次测试 id 值为", pm.variables.get("kalacloud_id"));
```

可以显示隐藏在变量下面的具体变量值，方便我们测试时，进行相应的判断。

十三. 如何使用 Runner 批量执行测试，批量更换变量测试

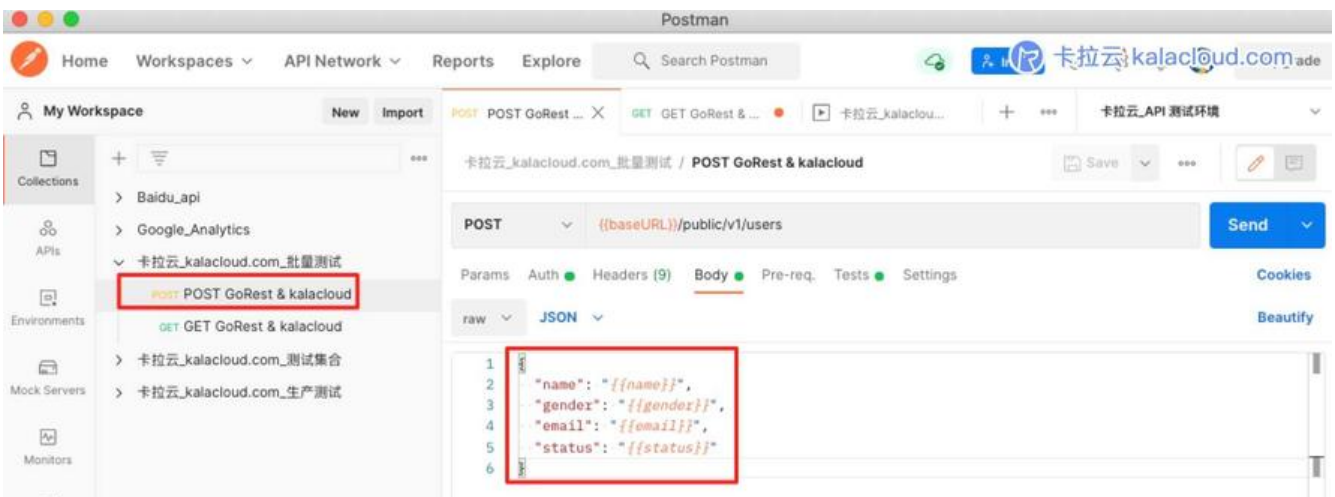
当我们有一组 API 且这一组之间相互关联的关系，使用手动测试效率非常低。这时，我们就要用到 Postman 的批量执行 (Runner) 功能，Runner 不仅可以批量执行 API 调用，还可以批量更换变量。掌握此方法，大幅度提升 API 测试效率。

打开你的 Postman，跟随本教程一起操作一遍吧。

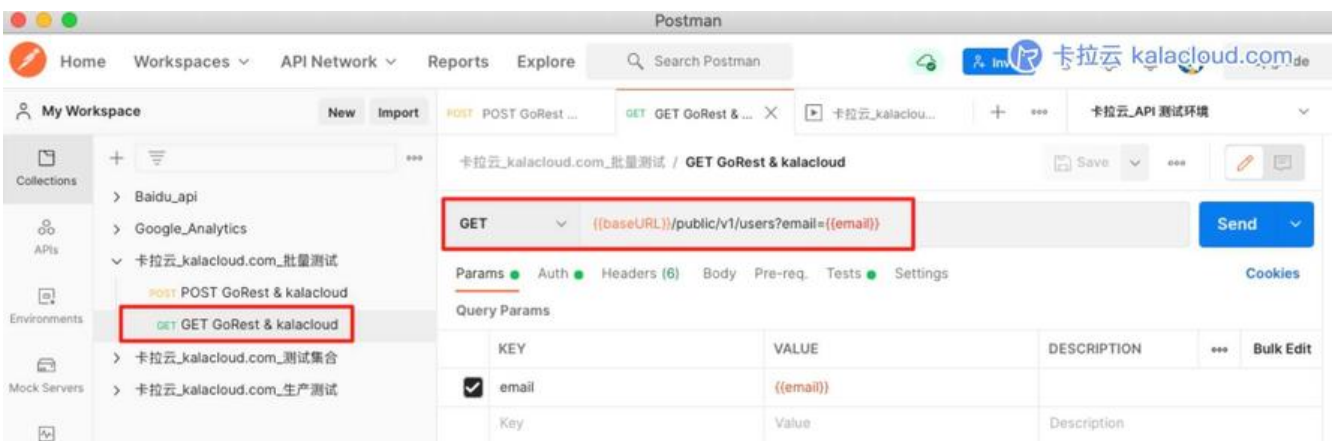


name	gender	email	status
Jiangtiezhu	male	Jiangtiezhu@kalacloud.com	Active
xieguoqing	male	xieguoqing@kalacloud.com	Active
Lvxiaoguo	famale	lvxiaoguo@kalacloud.com	Active
Songamei	famale	songamei@kalacloud.com	Active

● 本次批量 API 测试，我们先导入一个 CSV 文件，文件中包含四组等待新建的用户信息，将 CSV 文导入 Runner 中待用。



● 新建 POST 调用页，在 Body 里写上创建用户所需信息，所有值使用变量替代，这些变量将从 CSV 中读取。

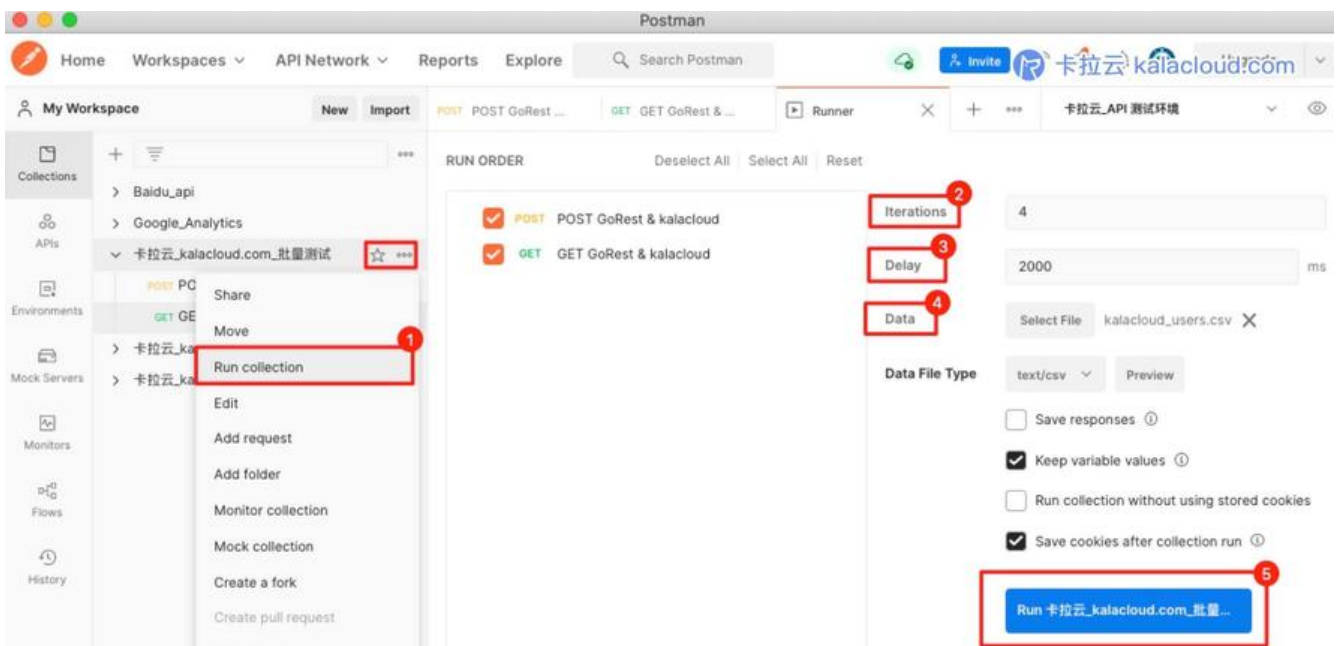


- 新建 GET 调用页，使用 **email** 作为查询 KEY 进行查询，如果上一步 POST 执行成功，那么 GET 就能成功查询到新建用户对应的 email，查询到表示 POST 创建成功。

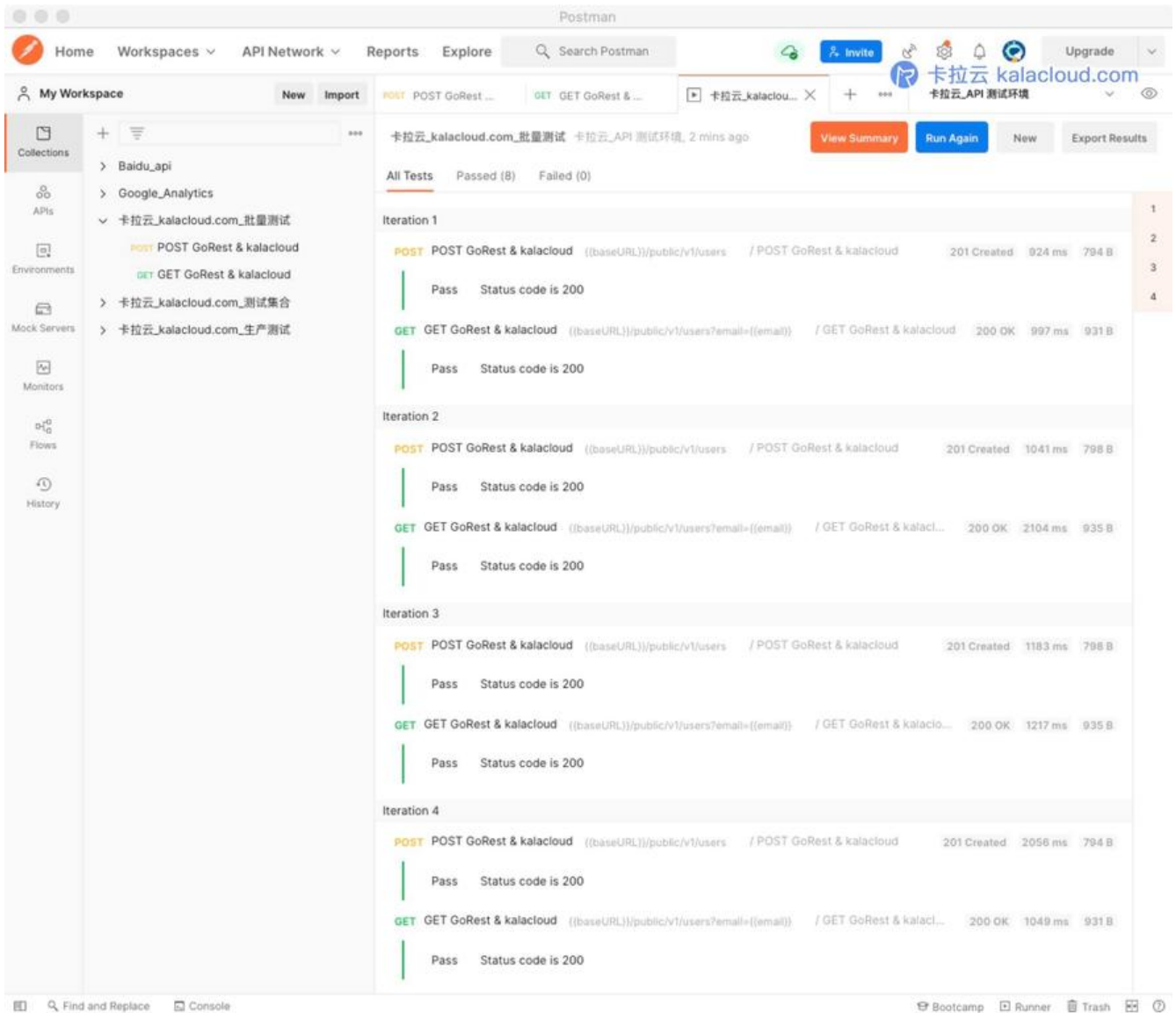


- 在「卡拉云 kalacloud.com批量测试」这个集合中设置 Tests 中设置全局断言，每当一个调用执行完毕时，进行 Tests 一次判断。

```
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
```



- 打开「卡拉云 kalacloud.com批量测试」合集的「Run Collection」的设置页
- Iterations: 这是测试组，我们 CSV 文件中有 4 组测试条目
- Delay: 延迟，一般填 2000 毫秒，太密集的请求，容易被服务器拒绝
- Data: 这里选择我们刚刚的 CSV 文件：kalacloud_users.csv 导入测试数据
- 点击 RUN 蓝色按钮开始执行批量测试

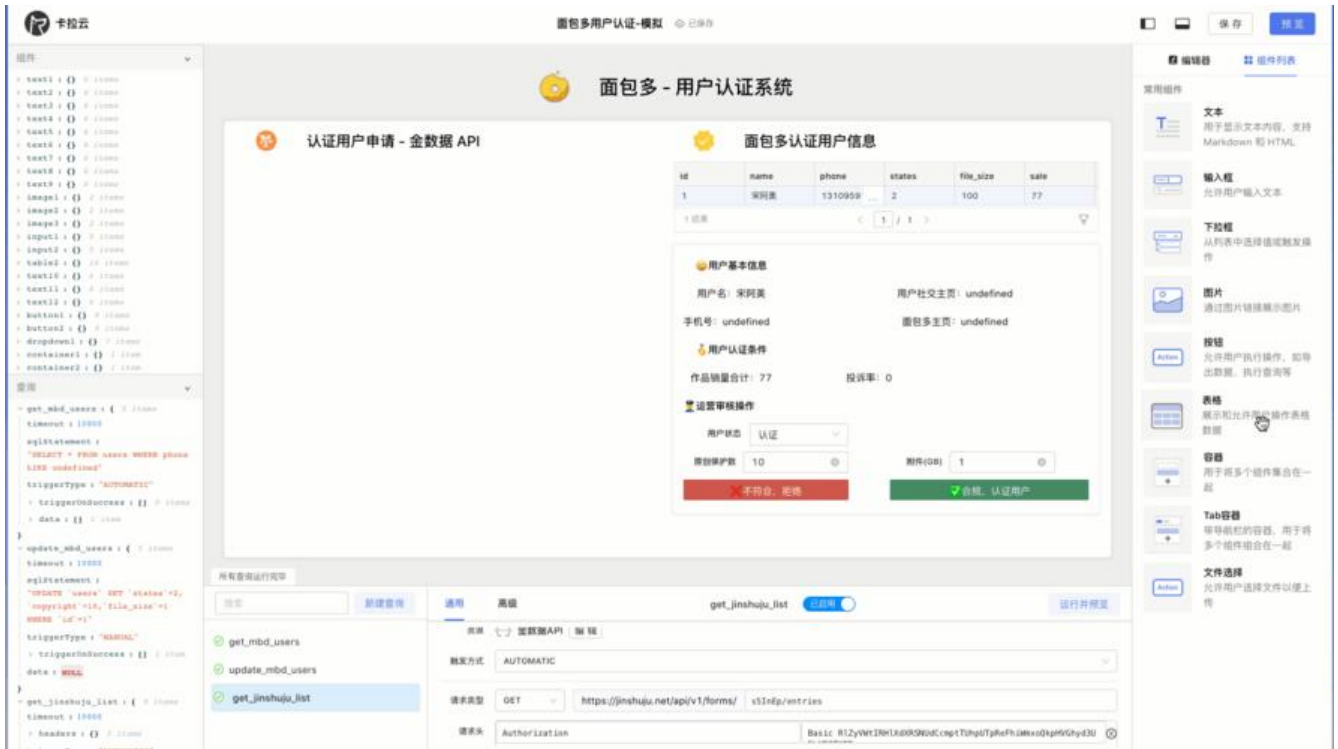


从上图可以看到，Postman 按顺序提交了 POST 请求和 GET 请求，并连续测试了从 CSV 文件导入的 4 组数据。一键批量测试，相当高效。

十四. Postman 接口测试总结

本文从 GET / POST 功能开始讲起，然后详细讲解了 API 接口自动化测试所需要的几个功能特点，全/环境/集合变量设置，测试断言，测试集合等，以及最后的自动化测试工具 Runner。这些功能共同成了 Postman API 接口自动化测试功能。

接着推荐一下卡拉云，卡拉云是一套低代码开发工具，支持多种数据库及 API 接入。你不仅可以在卡拉云中测试 API，还能直接把返回的结果映射到组件上，不用写一行代码，只需简单拖拽即可搭建属于自己的后台工具。



上图为使用「卡拉云」调用金数据 API，然后将返回结果直接映射到表格组件中。你仅需一步，即验了 API 可用性，又直接把你正在搭的工具给作出来了。立即[试用卡拉云](#)，一分钟快速搭建属于你自己 API 工具。

扩展阅读：

- [MySQL 时间戳用什么类型 - MySQL 时间函数详解](#)
- [最好用的七大顶级 API 接口测试工具](#)
- [最好用的 5 款 React 富文本编辑器](#)
- [如何在 MySQL / MariaDB 中跳过多张表导出或指定多张表导出备份](#)
- [如何将 MySQL / MariaDB 的查询结果保存到文件](#)
- [如何在 MySQL 中导入和导出 CSV / Excel 文件](#)