

自由微信安卓APP发布，立即下载！

该内容已被发布者删除 该内容被自由微信恢复。

文章于 3月17日 下午 8:00 被检测为删除。

3月17日 下午 8:00 ▾

查看原文

## 白嫖CDN，打造封不尽IP的代理池

Original Medicean 学蚁致用

### 写在前面

接上次 论如何防溯源连接WebShell 中提到的思路，我们来看看怎么借用云函数实现一个HTTP动态代理池。

提前说明一下：实现代理池的方式有很多，不局限这一种

先看效果

```
* ~ curl -x "http://localhost:3080" "http://cip.cc"
IP      : 42.194.176.222
地址    : 中国 中国

数据二 : 北京市海淀区 | 北龙中网(北京)科技有限公司

数据三 :

URL    : http://www.cip.cc/42.194.176.222
* ~ curl -x "http://localhost:3080" "http://cip.cc"
IP      : 42.194.198.175
地址    : 中国 中国

数据二 : 北京市海淀区 | 北龙中网(北京)科技有限公司

数据三 : 中国北京北京

URL    : http://www.cip.cc/42.194.198.175
* ~ curl -x "http://localhost:3080" "http://cip.cc"
IP      : 42.194.213.171
地址    : 中国 中国

数据二 : 北京市海淀区 | 北龙中网(北京)科技有限公司

数据三 :

URL    : http://www.cip.cc/42.194.213.171
* ~ curl -x "http://localhost:3080" "http://cip.cc"
IP      : 106.52.253.196
地址    : 中国 中国

数据二 : 北京市海淀区 | 北龙中网(北京)科技有限公司

数据三 :

URL    : http://www.cip.cc/106.52.253.196
* ~ curl -x "http://localhost:3080" "http://cip.cc"
IP      : 139.199.204.232
地址    : 中国 广东 广州
运营商 : tencent.com

数据二 : 广东省广州市 | 深圳市腾讯计算机系统有限公司IDC机房(电信)

数据三 :

URL    : http://www.cip.cc/139.199.204.232
* ~ curl -x "http://localhost:3080" "http://cip.cc"
IP      : 42.194.200.162
地址    : 中国 中国

数据二 : 北京市海淀区 | 北龙中网(北京)科技有限公司

数据三 :

URL    : http://www.cip.cc/42.194.200.162
```

学蚁致用

挂上代理之后，**每次 HTTP 请求，都会以不同的 IP 发送出去**，并且出口IP均为 IDC 机房 IP。随便挑选其中一个 IP 去 TI 上查询，可以看到 IP 很干净，专治各种买了威胁情报服务的目标。

The screenshot shows a detailed analysis of the IP address 42.194.200.162. Key details include:

- 地理位置:** 中国 广东省 广州市
- 用户标签:** IDC服务器
- 开放端口:** 0
- 与该IP通信样本:** 0
- IP反查:** 0
- 开放端口:** 0
- IP上相关URL:** 0
- 首次域名指向:**
- 本次域名指向:**
- RDNS:** CNNIC-TENCENT-NET-AP Shenzhen Tencent Computer Systems Company Limited, CN
- AS:** AS17417
- 情报判定:** 未知
- 相关事件:** 无
- 存在通讯的样本:** 无

## 原理

总结下来就是一句话：利用云厂商提供的云函数（函数计算）功能，将客户端的HTTP请求进行转发，由于云函数多出口的特性，让我们也变相拥有了代理池。

## 云函数

有关云函数的详细介绍可以直接去看各云厂商的产品文档。我们只提几个重点：

1. 云函数不需要服务器，也就是说你不需要去买 VPS。
2. 云函数只是云厂商用自己的服务器帮你运行你上传的代码片段，执行某个单一的逻辑，可以简单理解为只帮你执行一个函数。
3. 云函数无法长驻，调用的时候创建，执行完之后立即就销毁，所以无法直接保存状态。也正是这一点，让我们无法代理像 SSH 这种需要长连接的服务，只能代理 HTTP(s) 这种无状态的协议。

以腾讯云云函数服务为例，工作原理里面提到了复用原始连接的情况，为代理 TCP 连接提供了可能，但是目前鉴于时间原因没有深入尝试。

# 工作原理

最近更新时间: 2021-01-27 21:42:54

[前往 GitHub 编辑](#) [贡献](#) [我的收藏](#)

## 函数运行时的容器模型

SCF 将在事件触发时代表您执行 SCF 函数，根据您的配置信息（如内存大小等）进行资源分配，并启动和管理容器（即函数的执行环境）。SCF 平台负责所有函数运行容器的创建、管理和删除清理操作，用户没有权限对其进行管理。

容器启动需要一些时间，这会使每次调用函数时增加一些延迟。但通常仅在首次调用函数、更新函数或长时间未调用时重新调用函数时才会察觉到延迟。因为平台为了尽量减少容器启动延迟，会尝试对后续调用重用容器，在调用函数后容器会根据平台的实际情况存留一段时间，预期用于下次调用，在此时间段内的调用会直接重用该容器。

容器重用机制的意义在于：

- 用户代码中位于 [执行方法](#) 外部的任何声明保持已初始化的状态，再次调用函数时可以直接重用。例如，如果您的函数代码中建立了数据库连接，容器重用时可以直接受用原始连接。您可以在代码中添加逻辑，在创建新连接之前检查是否已存在连接。
- 每个容器在 /tmp 目录中提供部分磁盘空间。容器存留时该目录内容会保留，提供可用于多次调用的暂时性缓存。再次调用函数时有可能可以直接使用该磁盘内容，您可以添加额外的代码来检查缓存中是否有您存储的数据。

### 注意:

请勿在函数代码中假定始终重用容器，因为是否重用和单次实际调用相关，无法保证是创建新容器还是重用现有容器。

## 临时磁盘空间

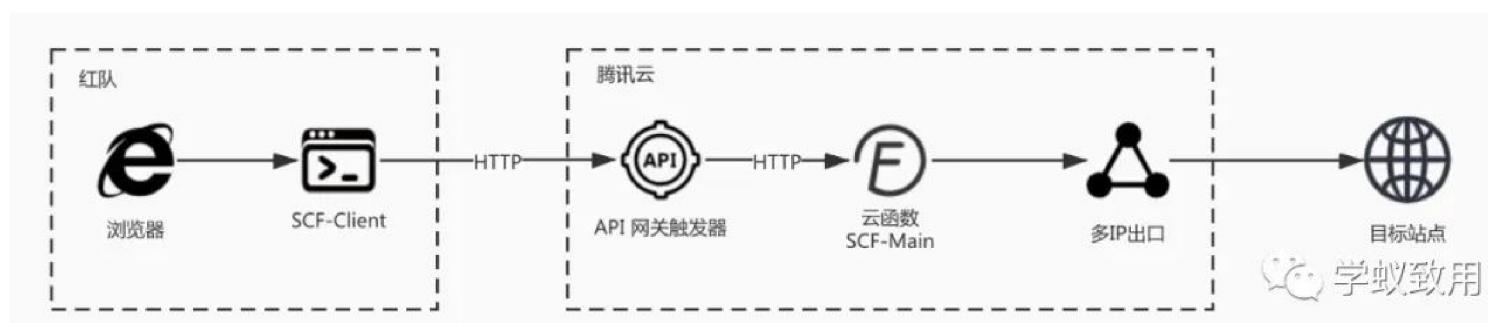
SCF 函数在执行过程中，都拥有一块512MB的临时磁盘空间 /tmp，用户可以在执行代码中对该空间进行一些读写操作，也可以创建子目录，但这部分数据可能不会在函数执行完成后保留。因此，如果您需要对执行过程中产生的数据进行持久化存储，请使用 COS 或 Redis/Memcached 等外部持久化存储。

 学蚁致用

# 设计

云函数不能直接调用，同时还需要创建一个「**触发器**」来触发云函数，为了方便我们选择使用「**API 网关触发器**」，好处是一个 HTTP 请求就可以触发，也不需要什么别的操作。

我们整个代理池的架构如下：



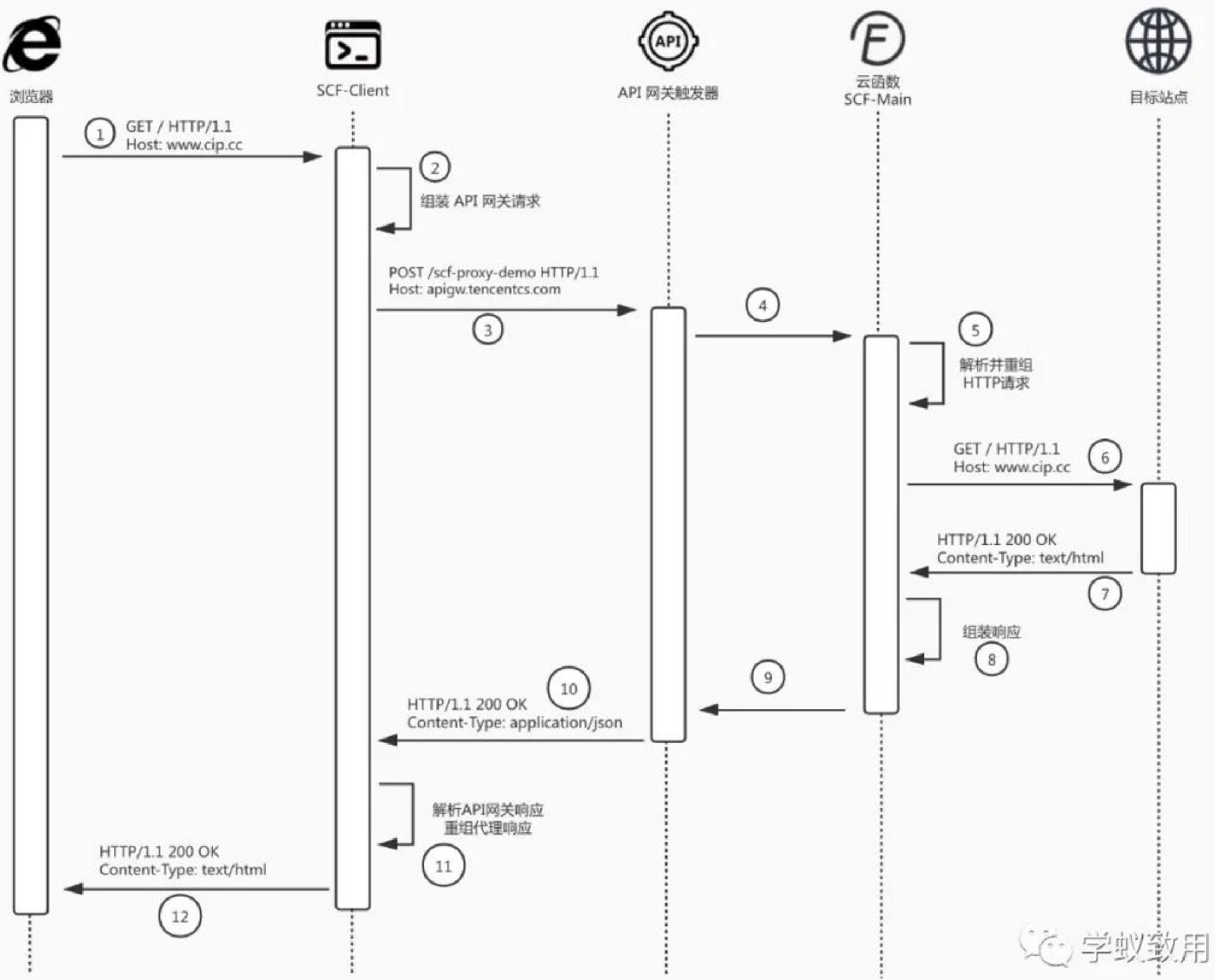
## SCF-Main

这个是我们云函数，我们需要部署到云上。负责解析结构化的数据，组装 HTTP 请求，发送之后把响应再组装好返回给 API 网关。

## SCF-Client

这是客户端程序，主要做 3 件事：

- 提供 HTTP/HTTPS 代理服务
- 把浏览器请求的数据包组装成我们自定义的格式，发送给 API 网关
- 把 API 网关返回的数据解析后，组装成 HTTP 代理的响应，返回给浏览器



学以致用

## 开发——云函数侧

云函数侧要做的事情很简单，只需要接到网关传过来的数据，解开之后自己拼出来一个 HTTP 请求，发送出去，然后把 Response 信息再拼好，返回给 API 网关即可。

我们定义两个结构，分别是 HTTP 请求和响应：

```

1 // DefineEvent 请求结构
2 type DefineEvent struct {
3     URL      string `json:"url"`    // 目标的 URL, eg: http://cip.cc/
4     Content  string `json:"content"` // 最原始的 HTTP 报文, base64
5 }
6 // RespEvent 响应结构
7 type RespEvent struct {
8     Status  bool   `json:"status"` // 请求是否正常
9     Error   string `json:"error"` // 错误信息

```

```
10 Data string `json:"data"` // HTTP 响应原始报文, base64
11 }
```

利用 `http.ReadRequest` 直接从原始报文创建 request 对象

```
1 req, err := http.ReadRequest(bufio.NewReader(strings.NewReader(string(rawreq))))
```

接下来就是 dump 出 response 报文，组装成 API 网关返回的格式返回

```
98     defer resp.Body.Close()
99     dump, err := httputil.DumpResponse(resp, true)
100    if err != nil {-
101        }
102        ret := base64.StdEncoding.EncodeToString(dump)
103        retVal.StatusCode = 200
104        respevent.Status = true
105        respevent.Data = ret
106        respevent.Error = ""
107        respeventbyte, _ := json.Marshal(respevent)
108        retVal.Body = string(respeventbyte)
109        return retVal, nil
110    }
111
112
113
114
115
116
```

学蚁致用

至此云函数端就已经完成了

## 开发——SCF-Client 侧

首先按照一个正常的 http 代理服务去开发，处理浏览器和本地代理端口的数据包，这个没啥说的，大篇文章可以找到，100行就能搞定。

常规的 HTTP 代理是把浏览器发过来的包直接就转发出去了，我们需要把这一块的逻辑重写一下。

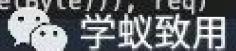
把浏览器侧发过来的 HTTP 数据包，组装成 SCF-Main 请求格式，发送给 API 网关

```
90     dumpreq, err := httputil.DumpRequest(req, true)
91     if err != nil {-
92         }
93         event := &DefineEvent{
94             URL:      req.URL.String(),
95             Content: base64.StdEncoding.EncodeToString(dumpreq),
96         }
97
98 }
```

学蚁致用

拿到API网关响应包解析并处理后，转换成本地 HTTP 请求正确的响应包，返回给浏览器

```
127 var respevent RespEvent
128 err = json.Unmarshal(body, &respevent)
129 > if err != nil {-
132 }
133 > if respevent.StatusCode > 0 && respevent.StatusCode != 200 {-
136 }
137 > if !respevent.Status {-
140 }
141 retByte, err := base64.StdEncoding.DecodeString(respevent.Data)
142 > if err != nil {-
145 }
146 proxyresp, err := http.ReadResponse(bufio.NewReader(strings.NewReader(string(retByte))), req)
147 > if err != nil {-
150 }
```



## 部署&使用步骤

### 云函数部分

1. 打开 「**函数服务**」 (<https://console.cloud.tencent.com/scf/list?rid=1&ns=default>) 点击「新建」

The screenshot shows the Tencent Cloud Functions management console. On the left, there's a sidebar with 'Cloud Functions' selected. The main area has tabs for 'Cloud Functions' (selected), 'Overview', and 'Layers'. A large blue 'New' button is prominently displayed. Above it, there are dropdown menus for 'Region' (set to 'Guangzhou') and 'Namespace' (set to 'default'). Below the 'New' button, there are fields for 'Function Name' and other settings, along with tabs for 'Monitoring' and 'Runtime Environment'. A watermark '学蚁致用' is visible in the bottom right corner.

2. 按照图中步骤填写, **不要手欠直接点完成**

基础配置

创建方式

模板创建  
使用示例模版快速创建一个函数或应用

自定义创建  
使用helloworld示例自定义创建函数

1

基础配置

函数名称

scf-proxy-main 2 名字随你喜好

只能包含字母、数字、下划线、连字符、以字母开头、以数字或字母结尾，2~60个字符

地域

广州 3 区域随你喜好

运行环境

Go1 4 选 Go1 环境

函数代码

选择本地上传 zip 包，执行方法写 main

提交方法

在线编辑  本地上传zip包  本地上传文件夹  通过cos上传zip包

执行方法

main ①

函数代码

选择打包好的 main.zip

main.zip

重新上传

请上传zip格式的代码包，最大支持50M（如果zip大于10M，仅显示入口文件）

高级配置

① 函数调用日志会默认发送到日志服务 SCF 专用日志主题。腾讯云日志服务CLS为独立计费产品。SCF专用日志主题各占用CLS免费额度。查看详情



## 高级配置

命名空间 \*  ▼

描述

最大支持1000个英文字母、数字、空格、逗号、句号、中文

### 环境配置

内存  ▼ ①

执行超时时间  秒 ① ← 此处默认是3秒，容易超时，建议60秒

时间范围：1-900秒

### 环境变量

key	value	①
请输入环境变量	请输入环境变量的值	<span style="border: 1px solid black; padding: 2px;">X</span>

### 权限配置

运行角色  启用 ①

### 日志配置

日志投递  默认投递 ①  
 自定义投递 ①

### 层配置

层  ①

### 网络配置

公网访问  启用 ① ← 这里仅需要选择「公网访问」，绝对不能勾「固定公网IP」

固定出口IP  启用 ①

私有网络  启用 ①

### 文件系统

文件系统 请点击链接 [至CFS控制台进行开通](#)，完成后请刷新页面。

### 执行配置

异步执行  启用 ①

异步执行启用后，函数执行超时时间最大可支持 24 小时，如有需要，请到执行超时时间调整。  
请在日志配置中指定日志投递主题，否则会导致日志丢失。

其它参数保持默认，然后打开下一个面板



### 触发器配置

**网络配置**

公网访问  启用 ①

固定出口IP  启用 ①

私有网络  启用 ①

**文件系统**

请点击链接 [至CFS控制台进行开通](#), 完成后请刷新页面。

**执行配置**

异步执行  启用 ①  
异步执行启用后, 函数执行超时时间最大可支持 24 小时, 如有需要, 请到执行超时时间调整, 请在日志配置中指定日志投递主题。否则会导致日志丢失。

状态追踪  启用 ①

**并发配置**

并发配置

=

并发数 X

配置内存

**标签**

标签  启用

**触发器配置**

创建触发器  自定义创建 ①

触发版本 版本: \$LATEST

触发方式 API网关触发

API服务类型 ①  新建API服务  使用已有API服务

API服务 SCF\_PROXY\_DEMO\_API\_SERVICE 填API服务名字

请求方法 ① ANY

发布环境 ① 发布

鉴权方法 ① 免鉴权

集成响应 ①  启用 勾上集成响应

暂不创建

确认无误后点「完成」

**字节致用**

**完成** **取消**

3. 点击完成后, 自动上传 zip 包并部署, 同时会创建 API 网关, 直接点「立即访问」

部署日志

时间	操作	状态
2021-03-12 13:03:48	部署包	操作成功
	API网关类部署包创建中...此时离开页面可自动管理部署包创建进度 API网关类部署包创建成功。点击 <a href="#">访问</a> 。API服务: SCF_PROXY_DEMO_API_SERVICE 翻译	
2021-03-12 13:03:49	部署包	操作成功
	函数包构建中... 函数包构建成功	

学以致用

#### 4. 获取云函数的 API 访问路径

[scf-proxy-main 正常](#)

函数管理	触发管理																				
触发管理	<a href="#">创建触发器</a> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <b>API网关触发</b> 所属版本: \$LATEST           <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>API服务名</td> <td>SCF_PROXY_DEMO_API_SERVICE</td> </tr> <tr> <td>serviceId</td> <td>service-23e5sr7o</td> </tr> <tr> <td>apiId</td> <td>api-eio8gy18</td> </tr> <tr> <td>请求方法</td> <td>ANY</td> </tr> <tr> <td>发布环境</td> <td>发布</td> </tr> <tr> <td>鉴权方式</td> <td>免鉴权</td> </tr> <tr> <td>启用集成响应</td> <td>已启用</td> </tr> <tr> <td>支持CORS</td> <td>否</td> </tr> <tr> <td>后端超时</td> <td>15s</td> </tr> <tr> <td>访问路径</td> <td><a href="https://service-23e5sr7o-1257105570.gz.apigw.tencentcs.com/release/scf-proxy-main">https://service-23e5sr7o-1257105570.gz.apigw.tencentcs.com/release/scf-proxy-main</a> </td> </tr> </table> <div style="text-align: right; margin-top: 10px;"> <span style="color: red;">复制「访问路径」，这个地址就是 API 网关地址了</span> </div> </div>	API服务名	SCF_PROXY_DEMO_API_SERVICE	serviceId	service-23e5sr7o	apiId	api-eio8gy18	请求方法	ANY	发布环境	发布	鉴权方式	免鉴权	启用集成响应	已启用	支持CORS	否	后端超时	15s	访问路径	<a href="https://service-23e5sr7o-1257105570.gz.apigw.tencentcs.com/release/scf-proxy-main">https://service-23e5sr7o-1257105570.gz.apigw.tencentcs.com/release/scf-proxy-main</a>
API服务名	SCF_PROXY_DEMO_API_SERVICE																				
serviceId	service-23e5sr7o																				
apiId	api-eio8gy18																				
请求方法	ANY																				
发布环境	发布																				
鉴权方式	免鉴权																				
启用集成响应	已启用																				
支持CORS	否																				
后端超时	15s																				
访问路径	<a href="https://service-23e5sr7o-1257105570.gz.apigw.tencentcs.com/release/scf-proxy-main">https://service-23e5sr7o-1257105570.gz.apigw.tencentcs.com/release/scf-proxy-main</a>																				

学以致用

#### Client 侧

scf-client 的参数如下：

```
1 → scf-client ./scf-client -h
2 Usage of ./scf-client:
```

```
3 -api string // 必选项
4     API Gateway url, eg: https://service-xxxxx-1257105570.gz.apigw.tencentcs.com
5 -l string
6     Listen Addr (default "0.0.0.0") // 代理监听的地址
7 -p int
8     HTTP Listen Port (default 3080) // HTTP 代理监听的端口
9 -password string
10    Auth Password, eg: --password 123456 // 如果你要开启认证的话, 就请设置 user 和 pa
11 -sp int
12    HTTPS Listen Port (default 3443)
13 -timeout int
14    Request Timeout (second) (default 60)
15 -user string
16    Auth User, eg: --user admin
```

运行 scf-client

```
1 ./scf-client --api https://service-23e5sr7o-1257105570.gz.apigw.tencentcs.com/rele
```

看到开启 HTTP 代理成功之后，我们使用 curl 进行测试

```
1 # http 代理 访问 http 站点
2 curl -x "http://localhost:3080" "http://cip.cc"
3 # http 代理 访问 https 站点
4 curl -x "http://localhost:3080" "https://httpbin.org/post?a=b" -X POST -d "user=h
5 # https 代理 访问 http 站点
6 curl -x "https://localhost:3443" "http://cip.cc" --proxy-insecure
7 # https 代理 访问 https 站点
8 curl -x "https://localhost:3443" "https://httpbin.org/post?a=b" -X POST -d "user=
```

```
medicean@Lumia-2:~  
→ scf-client ./scf-client --api https://service-23e5sr7o-1257105570.gz.apigw.tencentcs.com/release/scf-proxy-main  
2021/03/12 13:10:05 SCF Proxy Client Started  
2021/03/12 13:10:05 Listening HTTP 0.0.0.0:3080  
2021/03/12 13:10:05 Listening HTTPS 0.0.0.0:3443  
2021/03/12 13:10:26 INFO: Proxy 1 1: GET http://cip.cc/  
[]  
  
→ ~ curl -x "http://localhost:3080" "http://cip.cc"  
IP      : 42.194.200.118  
地址    : 中国 中国  
  
数据二  : 北京市海淀区 | 北龙中网(北京)科技有限公司  
  
数据三  : 中国北京北京  
  
URL     : http://www.cip.cc/42.194.200.118  
→ ~ [
```



## Debug

开发过程中，难免少不了 Debug，你只需要在云函数侧用 Print 把需要的数据打印出来就行了，然后在云函数的日志里，就可以看到你 Print 的数据。

我们把 response 的数据打印了出来，可以看到响应包的 base64 数据

The screenshot shows the Cloud Function Log interface with the following details:

- Log Stream:** scf-proxy-main
- Time Range:** 2021-03-12 13:10:00 - 2021-03-12 13:11:00
- Request ID:** 343a4e9144bc91fc0899cb0d4a78e6a06
- Timestamp:** 2021-03-12 13:10:28
- Response Content:** (base64 encoded response data)
- Content:** (Large base64 encoded string starting with START RequestId:343a4e9144bc91fc0899cb0d4a78e6a06...)



base64 解开之后就是最终响应的内容了

```
1 HTTP/1.1 200 OK
2 Transfer-Encoding: chunked
3 Connection: keep-alive
4 Content-Type: text/html; charset=UTF-8
5 Date: Fri, 12 Mar 2021 05:10:26 GMT
6 Server: openresty
7 Vary: Accept-Encoding
8 X-Cip-C: M
9
10 bd
11 IP  : 42.194.200.118
12 地址  : 中国 中国
13
14 数据二  : 北京市海淀区 | 北龙中网(北京)科技有限公司
15
16 数据三  : 中国北京北京
17
18 URL : http://www.cip.cc/42.194.200.118
19
20 0
21
22 |
```

学以致用

## 有关 HTTPS 的实现

我们知道 HTTPS 实际上第一个包是 CONNECT，打开 TCP 通道之后，先互传证书，之后再传 HTTP 报文。在云函数这个场景下，可以简单理解为一次 TCP 会话，TCP 握手之后，Client 只能向 Server 发一个 TCP 包，Server 也只能回一个包，之后就要断开连接。

这个问题也困扰了我很久，最终放弃了透传证书的方式，直接改用中间人（类似 Burpsuite 的 HTTPS 代理形式）

我们采用中间人的方式，在 SCF-Client 里内置了一个根证书，和浏览器交换证书的过程都在本地实现，只把最核心的 HTTP 报文发给 API 网关。

## 测试

HTTP 代理和 HTTPS 代理均可以对 HTTPS 站点代理

### HTTP 代理

```
medicean@Lumia-2:~
```

```
+ scf-client ./scf-client
2021/03/11 17:43:35 SCF Proxy Client Started
2021/03/11 17:43:35 Listening HTTP 0.0.0.0:3080
2021/03/11 17:43:35 Listening HTTPS 0.0.0.0:3443
2021/03/11 17:43:45 INFO: Proxy 1 1: POST http://httpbin.org/post?a=b
2021/03/11 17:44:35 INFO: OnConnect: httpbin.org:443
2021/03/11 17:44:35 INFO: Proxy 2 1: POST https://httpbin.org:443/post?a=b
[]
```

---

```
+ ~ curl cip.cc/42.194.202.189
IP      : 42.194.202.189
地址    : 中国 中国

数据二  : 北京市海淀区 | 北龙中网(北京)科技有限公司
```

```
+ ~ curl cip.cc/42.194.209.96
IP      : 42.194.209.96
地址    : 中国 中国

数据二  : 北京市海淀区 | 北龙中网(北京)科技有限公司
```

```
+ ~ curl cip.cc/42.194.209.96
IP      : 42.194.209.96
地址    : 中国 中国

数据三  :
```

```
URL      : http://www.cip.cc/42.194.202.189
+ ~ curl cip.cc/42.194.209.96
IP      : 42.194.209.96
地址    : 中国 中国

数据二  : 北京市海淀区 | 北龙中网(北京)科技有限公司
```

```
+ ~ curl -x "http://localhost:3080" "http://httpbin.org/post?a=b" -X POST -d
"user=hahaha"
{
  "args": {
    "a": "b"
  },
  "data": "",
  "files": {},
  "form": {
    "user": "hahaha"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip",
    "Content-Length": "11",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.64.1",
    "X-Amzn-Trace-Id": "Root=1-6049e652-69ac6d955390984b62c089ea"
  },
  "json": null,
  "origin": "42.194.202.189",
  "url": "http://httpbin.org/post?a=b"
}
```

```
+ ~ curl -x "http://localhost:3080" "https://httpbin.org/post?a=b" -X POST -d
"user=hahaha" -k
{
  "args": {
    "a": "b"
  },
  "data": "",
  "files": {},
  "form": {
    "user": "hahaha"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip",
    "Content-Length": "11",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.64.1",
    "X-Amzn-Trace-Id": "Root=1-6049e685-2575bdb34f8d2aec5b3ccb09"
  },
  "json": null,
  "origin": "42.194.209.96",
  "url": "https://httpbin.org/post?a=b"
}
```



```

medicean@Lumia-2:~ 
+ scf-client ./scf-client
2021/03/11 17:38:45 SCF Proxy Client Started
2021/03/11 17:38:45 Listening HTTP 0.0.0.0:3080
2021/03/11 17:38:45 Listening HTTPS 0.0.0.0:3443
2021/03/11 17:39:05 INFO: Proxy 1 1: POST http://httpbin.org/post?a=b
2021/03/11 17:39:43 INFO: OnConnect: httpbin.org:443
2021/03/11 17:39:43 INFO: Proxy 2 1: POST https://httpbin.org:443/post?a=b
[]

+ ~ curl -x "https://localhost:3443" "http://httpbin.org/post?a=b" -X POST -d
  "user=hahaha" --proxy-insecure
{
  "args": {
    "a": "b"
  },
  "data": "",
  "files": {},
  "form": {
    "user": "hahaha"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip",
    "Content-Length": "11",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.64.1",
    "X-Amzn-Trace-Id": "Root=1-6049e539-2c1c8666736aa4051585c837"
  },
  "json": null,
  "origin": "106.52.210.128",
  "url": "http://httpbin.org/post?a=b"
}

+ ~ curl -x "https://localhost:3443" "https://httpbin.org/post?a=b" -X POST -d
  "user=hahaha" -k --proxy-insecure
{
  "args": {
    "a": "b"
  },
  "data": "",
  "files": {},
  "form": {
    "user": "hahaha"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip",
    "Content-Length": "11",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.64.1",
    "X-Amzn-Trace-Id": "Root=1-6049e561-66388a046e112d1a0aab4962"
  },
  "json": null,
  "origin": "123.207.14.250",
  "url": "https://httpbin.org/post?a=b"
}

+ ~ curl cip.cc/106.52.210.128
IP      : 106.52.210.128
地址    : 中国 中国
数据二  : 北京市海淀区 | 北龙中网(北京)科技有限公司
数据三  :
URL     : http://www.cip.cc/106.52.210.128
+ ~ curl cip.cc/123.207.14.250
IP      : 123.207.14.250
地址    : 中国 广东 广州
运营商  : tencent.com
数据二  : 广东省广州市 | 腾讯云
数据三  :
URL     : http://www.cip.cc/123.207.14.250
+ ~ []

```

学蚁致用

最后

## 优点

- 自动切换出口IP，对使用者完全透明，不需要软件去实现代理切换的功能  
例如你挂 sqlmap 去注入，可以保证每一个数据包都走不同的IP出去，而命令行参数上只需要填写一个 proxy 地址
- IP 多如牛毛，封不完的IP，打不死的小强😊
- 该方法的 IP 都是白 IP，不在微步在线等威胁情报平台的黑名单里
- 便宜，便宜，便宜，每个账号每个月有免费额度，拉上你的同事朋友把多个账号凑一起，然后拿 nginx 做个负载均衡，能把白嫖嫖到底

⑦ 说明：

- 每个月的免费额度，会在每月开始时重置，不会进行累积。
- 计费时，结算顺序为：免费额度 > 资源包 > 按量计费（代金券结算）> 按量计费。即优先使用免费额度，超出免费额度的部分使用处于有效期内的资源包结算。若没有有效资源包或资源包已用完，则进行按量计费结算。
- 预置并发闲置量无法使用资源使用量的免费额度扣抵。

计费项	每月免费额度
资源使用量	400000GBs (40万GBs)
调用次数	1000000 次 (100万次)

下表标明在配置为不同内存时，单个函数每月可免费运行的时长：

内存 (MB)	免费时长 (秒)
64	6,400,000
128	3,200,000
256	1,600,000
512	800,000
1024	400,000
1536	266,667
3072	133,333

QQ 学以致用

## 缺点

- 仅支持 HTTP 和 HTTPS，这种无状态的协议。不支持其它协议，如果后期改改，勉强能适配一下 redis 协议
- 速度上来说，可能有时候会慢一些

## 工具

本文只谈思路，暂时不考虑提供现成工具，有兴趣的同学可以自己参照设计思路去实现，反正没几行代码。



不如关注一波再走？

```
total: 0.2 s
s3 versions: 0.03 s
s3 get: 0.07 s
```