受限网络访问Google云API的方法 (go/visitgapi)

Status: current
Authors:chandlerding@google.com
Last Updated: 2019-10-14

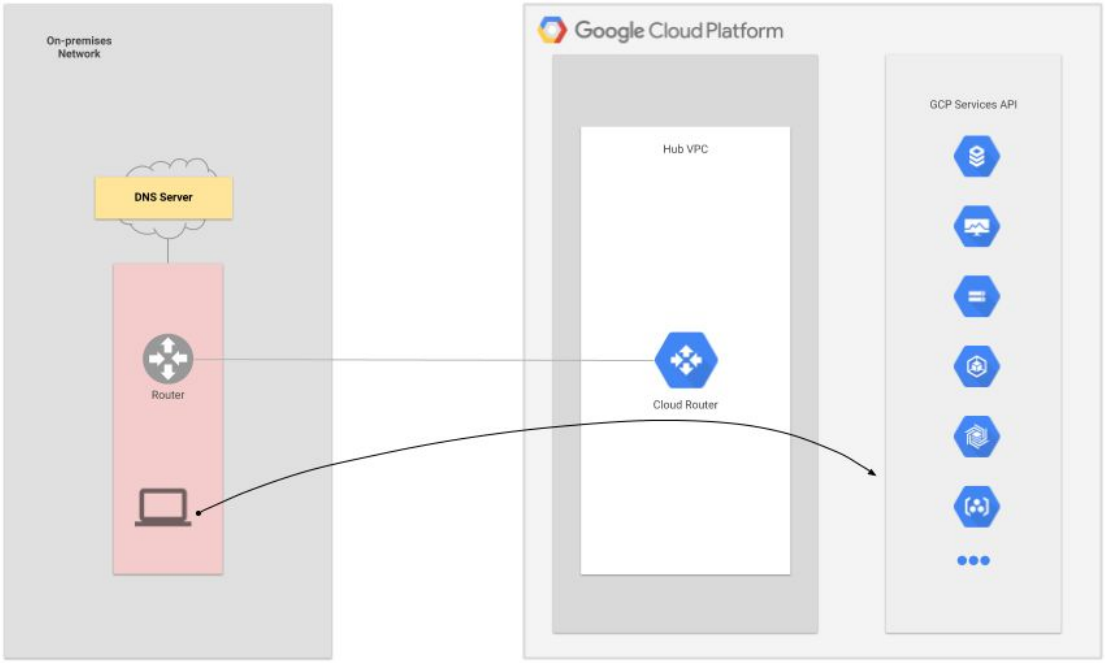
背景

本文尝试记录和总结Google Cloud Platform客户从受限网络访问GCP API服务端点 *.googleapis.com 的方法，供有需要的人士参考。

本文所指的受限网络场景，包括但不限于
- 与Internet隔离的企业内部网络
- 需要通过代理/网关实现受控访问的网络
- 由于当地法律法规无法通过Internet访问*.googleapis.com国家地区

场景概述

私有网络



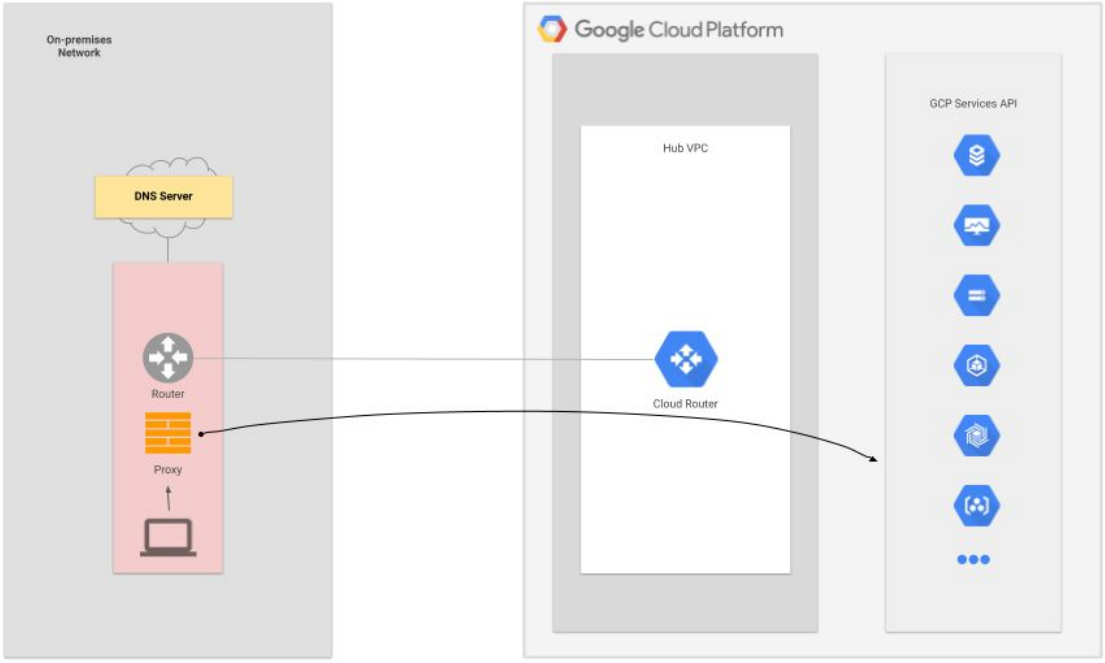
Google云平台支持客户通过专线、VPN等专有网络连接互联到客户的VPC并通过Private Google Access功能访问GCP API，过程简述如下。

1. 通过专线、或者VPN建立从客户数据中心到Google云的专有连接
2. 配置内部网络，将目标子网199.36.153.8/30的流量指向专有连接
3. 设置内部DNS解析记录 *.googleapis.com A 199.36.153.8,199.36.153.9,199.36.153.10,199.36.153.11
4. 配置内网网络防火墙、VPC防火墙策略，放通访问上述子网的流量

详细配置过程可参考 [Configuring Private Google Access for on-premises hosts]

采用这种方式通过专有网络连接访问GCP API的优点是对客户端程序透明，无需做任何修改或者配置Proxy等操作。但是此方法仅适用于您可以控制DNS解析的网络，对于Internet等场景无法使用此方法。

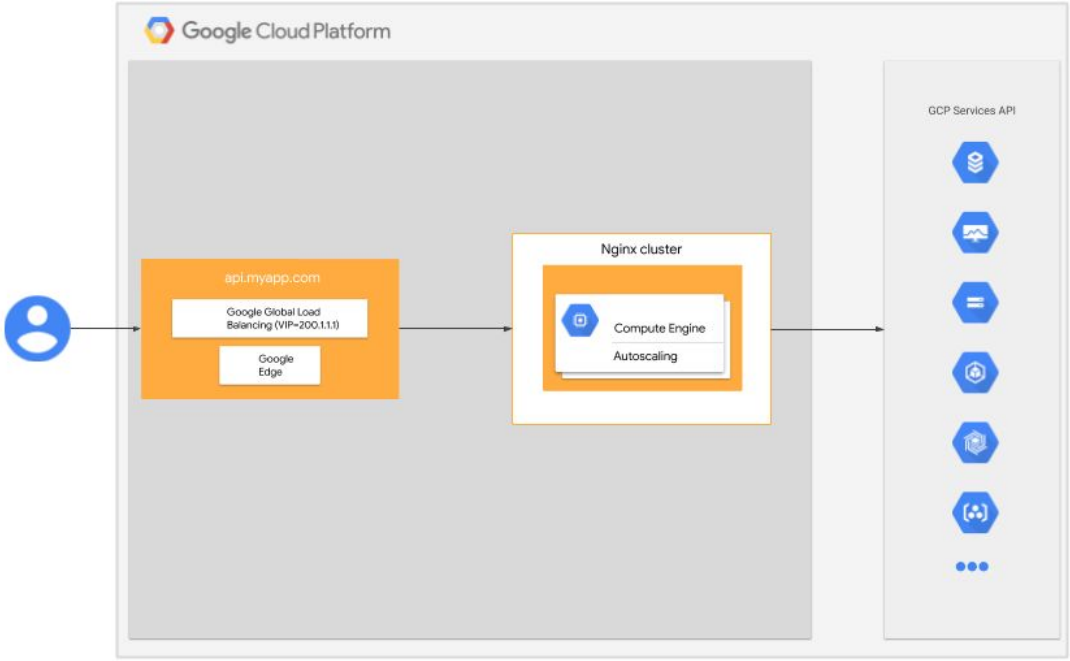
代理服务器



客户端也可以通过具备访*.googleapis.com条件的代理服务器访问GCP API。
具体请参考 [gcpsamples/proxy]

需要注意的是，不建议将代理服务器设置在境外。

Internet访问



在无法通过互联网访问*.googleapis.com的地区，可以通过搭建NGINX反向代理GCP API，最终用户使用您的自定义域名和IP访问。

搭建反向代理服务

Google Cloud API目前使用的协议有HTTP/S和GRPC。NGINX从1.13.10版本开始即支持GRPC协议的反向代理。 我们以GCE + Debian 9 + NGINX stable release为例说明如何搭建反向代理后端。
建议选择与您原本要访问的API服务所在Region创建反向代理VM；
以下假设开发者选用 storage.example.com 和 speech.example.com 来分别提供对storage api和speech api的访问。

创建VM并配置NGINX

 # 在Cloud Shell中运行gcloud命令行工具，创建一个名为nginx的VM，
gcloud compute instances create nginx --image-project=debian-cloud --image-family=debian-9 --zone=asia-east1-b --machine-type=n1-highcpu-2 --tags=https-server

 # SSH登陆刚刚创建的VM
gcloud compute ssh nginx

 # 安装nginx , ssl-cert 生成测试用自签名证书
sudo apt-get -y install nginx ssl-cert -t stretch-backports

 # 配置NGINX

 # 删除默认站点配置文件
sudo unlink /etc/nginx/sites-enabled/default

 # 优化连接保持时长, 75s -> 650s
sudo sed -i 's/keepalive_timeout.*/keepalive_timeout 650s;/' /etc/nginx/nginx.conf

 # 生成googleapis站点配置文件
cat <<EOF | sudo tee /etc/nginx/sites-enabled/googleapis
keepalive_requests 10000;
client_max_body_size 0; # allow client to upload file in any size
proxy_max_temp_file_size 0; # disable buffer upstream responses to disk, only memory buffer.

# HTTP/S 反向代理配置样例
server {
    server_name **storage.example.com**; #替换为实际域名
    listen 443 ssl http2;
    include snippets/snakeoil.conf; #自签名证书，生产环境建议配置正式SSL证书

    # 可选：配置gcs bucket白名单，只允许访问bucket-a,bucket-b,bucket-c
    # location ~* ^/(upload/)?storage/v1/(b$|projects/|b/(bucket-a|bucket-b|bucket-c)) {
    location ~* ^/(upload/)?storage/ {
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_pass https://www.googleapis.com$request_uri;
    }

```
        location / {
            add_header Cache-Control "public, max-age=60" always;
            return 410;
        }

}

 # GRPC 反向代理配置样例
server {
        server_name speech.example.com;  #替换为实际域名
        listen 443 ssl http2;
        include snippets/snakeoil.conf;  #自签名测试用证书，生产环境建议配置正式SSL证书

        location / {
            grpc_pass grpcs://speech.googleapis.com:443;
        }
}

server {
        # 对于主机名不匹配的请求默认返回410拒绝
        server_name  _;
        listen 443 ssl http2 default_server;
        include snippets/snakeoil.conf;  #自签名测试用证书，生产环境建议配置正式SSL证书

        location / {
            add_header Cache-Control "public, max-age=60" always;
            return 410;
        }
}
EOF

# 重启NGINX服务
sudo nginx -t && sudo systemctl restart nginx

# 至此，VM及NGINX配置完毕，退出VM SSH登陆
exit
```

创建HTTP Load Balancing

从性能、可用性和安全性角度，建议用户使用上述VM作为后端，创建HTTPS Load Balancing服务面向最终用户提供服务，可以充分利用Google遍布全球的边缘节点实现最优路由接入和动态内容加速来提升用户体验；
如需了解更多请参考 创建HTTPS Load Balancing

回到Cloud Shell中，创建全球HTTPS负载均衡

```
# 对外提供服务使用的域名
DOMAIN1=storage.example.com
DOMAIN2=speech.example.com

#为选定域名配置CAA记录，选取Letsencrypt作为CA签发GCP托管SSL证书
# storage.example.com. CAA 0 issue "letsencrypt"
# speech.example.com. CAA 0 issue "letsencrypt"

# 预留对外服务的v4和v6地址,并测试分配的IP是否可以从中国大陆访问
gcloud compute addresses create lb-apiproxy-ipv4-1 \
    --ip-version=IPV4 \
    --global

gcloud compute addresses create lb-apiproxy-ipv6-1 \
    --ip-version=IPV6 \
    --global

# 获取分配的IPv4和IPv6地址，据此设置两个域名的A记录和AAAA记录
gcloud compute addresses list lb-apiproxy-ipv4-1
gcloud compute addresses list lb-apiproxy-ipv6-1

# 创建免费的GCP托管SSL证书
gcloud beta compute ssl-certificates create apiproxy-cert-1 --global \
  --domains ${DOMAIN1}
gcloud beta compute ssl-certificates create apiproxy-cert-2 --global \
  --domains ${DOMAIN2}

# 或者 上传自有的SSL证书和密钥
gcloud compute ssl-certificates create apiproxy-cert \
    --certificate [CRT_FILE_PATH] \
    --private-key [KEY_FILE_PATH]

# 创建unmanaged instance group并添加前述名为nginx的实例到实例组
```

 # 生产环境中建议开发者使用managed-instance-group和auto scaling提供自动伸缩和VM故障自愈能力。

```
gcloud compute instance-groups unmanaged create ig-apiproxy-taiwan-b \
   --zone asia-east1-b

gcloud compute instance-groups unmanaged add-instances  ig-apiproxy-taiwan-b  \
   --zone=asia-east1-b \
   --instances=nginx

# 设置实例组提供服务的命名端口
gcloud compute instance-groups unmanaged set-named-ports  ig-apiproxy-taiwan-b \
   --named-ports https:443 \
   --zone=asia-east1-b

# 创建健康检查规则
gcloud compute health-checks create tcp tcp-basic-check \
   --port 443

# 创建backend-service并设置LB使用HTTP2协议访问后端VM
gcloud compute backend-services create apiproxy-backend-service \
   --protocol HTTP2 \
   --timeout 1200s \
   --health-checks tcp-basic-check \
   --global


gcloud compute backend-services add-backend apiproxy-backend-service \
   --balancing-mode=UTILIZATION \
   --max-utilization=0.8 \
   --capacity-scaler=1 \
   --instance-group=ig-apiproxy-taiwan-b \
   --instance-group-zone=asia-east1-b \
   --global



# 创建URLMap，Target Proxy和Forwarding Rules等组件
gcloud compute url-maps create apiproxy-map \
   --default-service apiproxy-backend-service

gcloud compute target-https-proxies create apiproxy-target \
   --url-map apiproxy-map --ssl-certificates apiproxy-cert-1,apiproxy-cert-2
```

```
gcloud compute forwarding-rules create https-apiproxy-v4 \
   --address=lb-apiproxy-ipv4-1 \
   --global \
   --target-https-proxy=apiproxy-target \
   --ports=443

gcloud compute forwarding-rules create https-apiproxy-v6 \
   --address=lb-apiproxy-ipv6-1 \
   --global \
   --target-https-proxy=apiproxy-target \
   --ports=443
```

至此，Global HTTPS Load Blancing创建完毕，等待SSL证书和相关配置应用到Google全球边缘节点后即可通过您指定的域名访问后端VM和相关的API；

客户端代码样例

通过反向代理访问GCP API时，客户端SDK代码需要修改访问域名将请求送往自己的反向代理服务。
不同服务、不同语言的SDK修改域名的方式大同小异，这里给出几个常见服务的常见语言SDK为例供参考。未列举的服务、语言请查阅对应服务/语言的Client Library文档。

Google Cloud Storage API

Java Client Library

```
package com.example.storage;

import com.google.cloud.storage.Bucket;
import com.google.cloud.storage.BucketInfo;
import com.google.cloud.storage.Storage;
import com.google.cloud.storage.StorageOptions;

public class QuickstartSample {
  public static void main(String... args) throws Exception {
    // 设置自定义hostname
```

```
    StorageOptions options=
StorageOptions.newBuilder().setProjectId("your-project-id")
.setHost("https://storage.example.com")
.build();

    Storage storage = options.getService();

    System.out.printf("The hostname is %s .%n", options.getHost());
    // The name for the new bucket
    String bucketName = "my-new-bucket";

    // Creates the new bucket
    Bucket bucket = storage.create(BucketInfo.of(bucketName));

    System.out.printf("Bucket %s created.%n", bucket.getName());
  }
}
// [END storage_quickstart]
```

Python Client Library

```
from google.api_core.client_options import ClientOptions
from google.cloud import storage

options = ClientOptions(api_endpoint="https://storage.example.com")
storage_client = storage.Client(client_options=options)

blobs = storage_client.list_blobs("bucket-a")
for blob in blobs:
        print(blob.name)
```

Google Cloud Speech API

Google Cloud Speech Client Libary 使用grpc协议访问API。

Python Client Library Example

```
from google.api_core.client_options import ClientOptions
from google.cloud import speech_v1
from google.cloud.speech_v1 import enums
```

options = ClientOptions(api_endpoint="speech.example.com")
client = speech_v1.SpeechClient(client_options=options)

```
encoding = enums.RecognitionConfig.AudioEncoding.FLAC
sample_rate_hertz = 16000
language_code = 'en-US'
config = {'encoding': encoding, 'sample_rate_hertz': sample_rate_hertz, 'language_code':
language_code}
uri = 'gs://cloud-samples-tests/speech/brooklyn.flac'
audio = {'uri': uri}
response = client.recognize(config, audio)
print (response)
```