



12.8k stars 2k forks 299 watching Branches Activity Tags

Public repository

master Branches Tags Go to file + Add file Code ...

Refresh

.github/workflows

images

scripts

README.md

_config.yml

README Edit Menu

科学上网

作者: 左耳朵 <http://coolshell.cn> 更新时间: 2023-05-02
这篇文章可以写的更好, 欢迎到 <https://github.com/haoel/haoel.github.io> 更新

- [1. 机场 vs 自建](#)
- [2. 购买VPS](#)
 - [2.1 常规VPS](#)
 - [2.2 CN2 线路](#)
- [3. 搭建相关代理服务](#)
 - [3.1 设置Docker服务](#)
 - [3.2 开启 TCP BBR 拥塞控制算法](#)
 - [3.3 申请域名和证书](#)
 - [3.4 用 Gost 设置 HTTPS 服务](#)
 - [3.5 设置 ShadowSocks 服务（不推荐）](#)
 - [3.6 设置L2TP/IPSec服务（不推荐）](#)
- [4. 客户端设置](#)
 - [4.1 电脑端设置](#)
 - [4.1.1 Chrome 代理设置](#)
 - [4.1.2 全局 Clash 代理设置](#)
 - [4.2 手机端设置](#)
 - [4.2.1 Android](#)
 - [4.2.2 iPhone](#)
 - [4.3 平板端设置](#)
 - [4.3.1 iPad](#)
- [5. 美国手机和支付](#)
 - [5.1 美国手机](#)
 - [5.2 美国支付](#)
- [6. 流量伪装和防探测](#)
- [7. 针对 IP 被封的解决方案](#)
- [8. 家用透明网关](#)
 - [8.1 OpenWRT 路由器](#)
 - [8.2 通过树莓派做旁路网关](#)
 - [8.3 安装 Clash](#)
 - [8.4 设置 iptables 转发](#)
- [9. 数据中心透明网关](#)
 - [9.1 AWS 网络构建](#)
 - [9.2 安装 Clash](#)
 - [9.3 配置私有子网中的 EC2](#)
 - [9.4 私有子网中的 Kubernetes](#)
- [10. 代理技巧](#)
 - [10.1 HTTP 隧道](#)
 - [10.2 SSH 隧道](#)
 - [10.3 Github / Git SSH 代理](#)
 - [10.4 Cloudflare Warp 原生 IP](#)
 - [10.4.1 WARP 模式](#)
 - [10.4.2 代理模式](#)
 - [10.4.3 Docker 代理](#)
- [11. 其它](#)
 - [11.1 其它方式](#)
 - [11.2 搭建脚本](#)

0. 序

首先，我们先明确一下，我科学上网的目的主要是为了学习、工作、交友、查资料、和丰富自己的眼界，不是其它的事。

对我来说，科学上网很重要，下面罗列一下需要科学上网，我才能真正学习工作和生活的网站：

Youtube 和 Vimeo 上的各种大会和教学视频，除了我自己要学，我的孩子也要学。

Wikipedia 维基百科是我目前唯一信得过的百科全书，我在上面可以比较系统地翻阅各种词条。

SlideShare 上有很多的技术文档和资料的PPT，是我的知识学习的地方。

Quora 问答网站，在上面有很多有趣的问答。

博客和论文，很多博客和论文站点都被墙了，比如：Blogspot 和 Medium。

Google 的各种服务，比如：Gmail, Map, Docs, Driver, 照片，图片搜索，Voices，论文搜索.....包括Google官方的各种技术文档.....

一些云服务，比如：Dropbox，IFTTT，Imgur，archive.org.....

Twitter 上 Follow 一些牛人和一些官方账号，比如：AWS、Docker.....

社交 Facebook，Telegram，Whatsapp，Slack.....，有一些我在国外的亲戚和朋友.....

Reddit 是一个聚合网站，一个新闻和文章的集散地，你可以认为是各种频道的今日头条.....

Pinterest 和 Instagram 上面有很多不错的图片和视频新闻，是我减压力的地方.....

新闻，如BBC。BBC是全球比较出众的媒体，有太多的有价值资源和内容了，比如纪录片、学英语.....

编程，有很多编程的场景需要翻墙，比如，Go语言编程时的 go get 中的很多库是放在 Google 的服务器上，然而Google是全部被墙，包括 Android 和其它一些文档和资源也是一样。包括 SourceForge 的某些项目也需要科学上网，Docker Registry也有部分被墙，还有偶尔抽风的 Github，以及不能访问的gist.....

.....等等

所以，科学上网后，英文是一件很重要的事。

为什么这么说？**这主要是针对计算机相关的知识，逻辑是这样的，如果你上了Google还是在用中文关键词，那么你好不容易出来了，结果又回去了，所以没什么意义。**换言之，科学上网的目的是为了进入广阔的世界范围与全世界的人交流，所以，英文是必备的，如果你英文有问题，VPN过去的用处也不大。

所以，我把这个前提条件放在第一的位置，就是说—— **真正的墙不是GFW，而是人的大脑！**意思是，屏蔽你获得信息能力的不是墙，而很大一部分则是我们自己的语言能力！

1. 机场 vs 自建

对于科学上网，我强烈建议通过自建的方式。因为使用一些“机场”会有如下的几个问题：

机场主需要公开售卖，使用的人也比较多，所以，都会被重点关注，非常容易被封；

机场需要购买一堆服务器，所以，很容易经营不善跑路；

使用机场很容易泄漏自己的隐私，因为你的所有流量都经过了机场的服务器；

自建梯子会是一条很好的路，不过，问题就是，自建梯子需要一定的技术能力，成本和门槛都比机场要高，但是，在安全性方面会比较好一些。自己动手，自力更生，让人有更多的安全感。

所以，如果你不是很有技术能力，或者不想花钱，那么，还是使用“机场”吧。

本文主要是针对自建的文档！

2. 购买VPS

自建梯需要购买一台服务器 VPS。**现在你买一台VPS也不贵了，一个月几十元钱。当然，如果需要好的线路得需要多花一些钱，但是也不贵。**

购买的 VPS 有几个要点 - “流量”和“位置”：

配置。服务器的配置不需要很高，用最小的型号就可以了，一般来说，1个CPU，1GB内存，50GB硬盘是足够了；

流量。服务器主是要要年有没有足够的流量和带宽，很多VPS提供商在流量和带宽上会收额外的费用，要小心。一般来说，1TB流量和30Mbps带宽是足够了，如果不看电影的话，500GB左右一个月的流量也够了；

位置。服务器的线路和地理位置还是比较重要的，一般来说，香港，台湾，日本，韩国，新加坡，美国，欧洲都是不错的选择。

线路。经过我长期的实践，GIA CN2 的线路是非常好的。**但是你要小心，很多VPS提供商会在宣传上说是 CN2 的线路，但是，实际上并不是。**

原生 IP。很多海外的应用都有地区限制，会屏蔽IP，所以，使用美国的原生 IP 也是很重要的。

上述的这些东西在后面都会讲到。**另外，VPS 建议安装成 Ubuntu，版本用 18.04，20.04，22.04 都行。**

2.1 常规VPS

对于常规 VPS，主要是一个云服务提供商，大的原厂商都提供免费使用和赠金，所以，可以白嫖一段时间。

[AWS LightSail](#) 是一个非常便宜好用的服务，最低配置一个月 \$3.5 美金，流量不限，目前的Zone不多，推荐使用日本，新加坡或美国俄勒冈（支持银联卡）。现对 2021/8/7 之后使用 LightSail 的用户提供3个月的免费试用。

[Microsoft Azure](#)提供免费一年的服务（B1S实例），而且每个月有 100G 的免费流量，并赠送200刀赠金。（需要国际信用卡）

[Google Cloud Platform](#)提供免费试用，赠送300刀赠金（需要国际信用卡）

[Oracle Cloud](#)两台VPS无限期使用，可选美日韩等地（需要国际信用卡）

[RackNerd](#)，价格在一年 10 刀-20 刀（如下所示）。支持支付宝、银联。购买的时候，有个 Location 选项，可以选择机房位置，后面有相应的 IP，你可以测试一下 ping 值，选择最低的那个。

[AWS EC2](#)香港、日本或韩国申请个免费试用一年的EC2 VPS （支持银联卡）

[Linode](#)买个一月USD5刀的VPS

[Vultr](#)上买一个日本的VPS，一个月5刀 - 6刀 （可以支付宝）

你要知道很多云厂商的 VPS 服务器和流量是分开计费的，所以，你要确认一下你的云账单以免出现不预期的费用。所以，最好购买包流量的 VPS。比如：AWS 的 LightSail, RackNerd 等。

在中国，因为太多的网络提供商，所以，不同的地方，不同的网络，不同的时间，到不同的国家完全不一样，而且还经常性地调整路由，所以，经常性地有时候快有时候慢，简直就是随机的。如果你对上网要求比较高的人，最好会备 2-3 个不同国家地区的 VPS。

香港网速应该比较好的，但是香港的成本也是比较高的（使用 Microsoft Azure 的 East Asia 可以把 VPS 买在香港区）。台湾的网速也是不错的（可以通过 Google Cloud 购买），日本的网速其次，新加坡再次之，然后是美国的东海岸。但是，因为线路的问题，如果没有为中国区优化的线路，丢包率是非常大的，日本区 ping 值虽然很低，但是经常性的丢包，好的线路的美国的 ping 值虽然大，但是也会飞快。

Google Cloud Platform - GCP 的香港和台湾节点也是很快的。但是你要能买 GCP 的主机，你还得先翻墙，所以，感觉有点死锁了。所以，你可能先用其它方式翻墙，然后再到 GCP 上购买。

2.2 CN2 线路

如果你需要更好更高速的网络服务（比如你要看 Youtube 的 1080P），那么，你需要下面的这些服务器资源了（价格也会高一些）

CN2 和 GIA 是两个关键词。CN2 GIA 全称 China telecom Next Carrier Network- Global Internet Access 电信国际精品网络，特征是路由线路上骨干节点均为 59.43 开头的 IP。如果想要寻找接入 CN2 线路的国外 VPS 提供商，建议使用 Next Carrier Network 或者 CN2 这个关键词搜索即可。

多说一句，CN2 本身又分为两种类型：

CN2 GT: CN2 里属于 Global Transit 的产品（又名 GIS-Global Internet Service），在 CN2 里等级低，省级/出国节点为 202.97 开头，国际骨干节点有 2~4 个 59.43 开头的 CN2 节点。在出国线路上拥堵程度一般，相对于 163 骨干网的稍强，相比 CN2 GIA，性价比也较高。

CN2 GIA: CN2 里属于 Global Internet Access 的产品，等级最高，省级/出国/国际骨干节点都以 59.43 开头，全程没有 202.97 开头的节点。在出国线路上表现最好，很少拥堵，理论上速度最快最稳定，当然，价格也相对 CN2 GT 偏高。

关于 CN2 线路的主机提供商，好些都不靠谱，只推荐下面两个，首推搬瓦工。

[搬瓦工](#) 这应该是美区最好的一个用来科学上网的 VPS 提供商了，实测飞快，无论在中国哪个地方都很快。购买时你需要注意 VPS 规格上的 CN2 和 GIA 的描述。（注：点击主页右上角的 register 以后，你可以看到页面上方有两个导航条，在下面的导航条上点 Services -> Order New Services 就可以看到所有的列表了。买完后，你可能需要重装一下操作系统，装成 64 位带 BBR 的）

[HostDare](#) 的 CN2 GIA 产品也是三网直连。专门有两个分类针对中国的 CN2 GIA，一个是 Premium China Optimized NVMeKVM，一组是 Premium China Optimized KVM。支持支付宝和银联。购买后要等一段时间（2-4 个小时）才会创建好，IP 有可能会给你是一个已经被封了的，如果是这样，你要在 3 天内申请换 IP，都是免费的。时间长了也可以换，就是时间比较长，而且还要花一定的费用（3 美金）。线路质量还不错，不过没有搬瓦工快。

更多的可以参考这篇文章《[CN2 GIA VPS 主机收集整理汇总-电信,联通,移动三网 CN2 GIA 线路 VPS 主机](#)》（注：随时间推移，这篇文章的内容可能会失效）

重点说一下，CN2 GIA + 香港机房，你会得到巨快无比的上网速度（无论你在中国的哪个位置，无论使用哪家运营商，CN2 GIA 都是最优的），然而，香港地区的 VPS 的确是有点贵了。在 Youtube 上看 4K 的视频毫无压力。虽然阿里云和腾讯的也有，但是被查到的风险基本上是 100%，不建议使用，被抓了别怪我没警告过你。

3. 搭建相关代理服务

注：如下的搭建和安装脚本可参看本库的 scripts 目录下的脚本，如：[Ubuntu 18.04 Installation Script](#)（感谢网友 [@gongzili456](#) 开发），另外，这个脚本可能年久失修，不一定能用，但是可以参考，如果有问题，可以提交 PR。

3.1 设置 Docker 服务

首先，你要安装一个 Docker CE 服务，这里你要去看一下 docker 官方的安装文档：

[CentOS 上的 Docker CE 安装](#)

[Ubuntu 上的 Docker CE 安装](#)

然后开始设置你的 VPN/SS 服务

3.2 开启 TCP BBR 拥塞控制算法

TCP BBR (Bottleneck Bandwidth and Round-trip propagation time) 是由Google设计, 于2016年发布的拥塞算法。以往大部分拥塞算法是基于丢包来作为降低传输速率的信号, 而BBR则基于模型主动探测。该算法使用网络最近出站数据分组当时的最大带宽和往返时间来创建网络的显式模型。数据包传输的每个累积或选择性确认用于生成记录在数据包传输过程和确认返回期间的时间内所传送数据量的采样率。该算法认为随着网络接口控制器逐渐进入千兆速度时, 分组丢失不应该被认为是识别拥塞的主要决定因素, 所以基于模型的拥塞控制算法能有更高的吞吐量和更低的延迟, 可以用BBR来替代其他流行的拥塞算法, 例如CUBIC。Google在YouTube上应用该算法, 将全球平均的YouTube网络吞吐量提高了4%, 在一些国家超过了14%。

BBR之后移植入Linux内核4.9版本, 并且对于QUIC可用。

如果开启, 请参看 [《开启TCP BBR拥塞控制算法》](#)

3.3 申请域名和证书

为了更为的隐蔽, 使用 HTTPS 服务, 你需要完成如下工作:

- 1) 一个域名 (可以上 [GoDaddy](#), 但一定要使用美国版)
- 2) 然后在 GoDaddy 上修改域名解析服务器, 把其指向 [Cloudflare](#) —— 当你注册完 Cloudflare 帐号后, Cloudflare 会告诉你怎么做。
- 3) 然后, 你在 Cloudflare 上创建一个子域名解析到你的 VPS 上 IP 上 (注: 不要开启 Cloudflare 的 Proxy 模式)
- 4) 最后, 使用 [Let's Encrypt](#) 来签一个证书。使用 Let's Encrypt 证书你需要在服务器上安装一个 [certbot](#), 点击 [certbot](#) 这个链接, 你可以选择你的服务器, 操作系统, 然后就跟着指令走吧。

接下来, 你需要申请一个证书 (我们使用standalone的方式, 然后, 你需要输入你的电子邮件和你解析到 VPS 的域名):

```
$ sudo certbot certonly --standalone
```

证书默认生成在 `/etc/letsencrypt/live/<YOUR.DOMAIN.COM/>` 目录下, 这个证书90天后就过期了, 所以, 需要使用一个 cron job 来定期更新 (稍后给出)

3.4 用 Gost 设置 HTTPS 服务

[gost](#) 是一个非常强的代理服务, 它可以设置成 HTTPS 代理, 然后把你的服务伪装成一个Web服务器, **我感觉这比其它的流量伪装更好, 也更隐蔽。这也是这里强烈推荐的一个方式。**

接下来就是启动 gost 服务了, 我们这里使用 Docker 的方式建立 gost 服务器。

```
#!/bin/bash

# 下面的四个参数需要改成你的
DOMAIN="YOU.DOMAIN.NAME"
USER="username"
PASS="password"
PORT=443

BIND_IP=0.0.0.0
CERT_DIR=/etc/letsencrypt
CERT=${CERT_DIR}/live/${DOMAIN}/fullchain.pem
KEY=${CERT_DIR}/live/${DOMAIN}/privkey.pem
sudo docker run -d --name gost \
    -v ${CERT_DIR}:${CERT_DIR}:ro \
    --net=host ginuerzh/gost \
    -l "http2://${USER}:${PASS}@${BIND_IP}:${PORT}?cert=${CERT}&key=${KEY}&probe_resist=code:404&knock=www.google.com"
```

上面这个脚本, 你需要配置: 域名 (DOMAIN), 用户名 (USER), 密码 (PASS) 和 端口号 (PORT) 这几个变量。

关于 gost 的参数, 你可以参看其文档: [Gost Wiki](#), 上面我设置一个参数 `probe_resist=code:404` 意思是, 如果服务器被探测, 或是用浏览器来访问, 返回404错误, 也可以返回一个网页 (如: `probe_resist=file:/path/to/file.txt` 或其它网站 `probe_resist=web:example.com/page.html`)

Note

开启了探测防御功能后, 当认证失败时服务器默认不会响应 407 Proxy Authentication Required, 但某些情况下客户端需要服务器告知代理是否需要认证 (例如 Chrome 中的 SwitchyOmega 插件)。通过 knock 参数设置服务器才会发送 407 响应。对于上面的例子, 我们的 knock 参数配置的是 `www.google.com`, 所以, 你需要先访问一下 `https://www.google.com` 让服务端返回一个 407 后, SwitchyOmega 才能正常工作。

如果认证信息 (也就是用户名和密码) 中包含特殊字符, 则可以 (应该是必须! 否则客户端一侧会有很多不兼容) 通过 auth 参数来设置, 下面是使用 auth 参数的例子 (注意, 需要 gost 在 2.9.2+ 以上版本):

```

DOMAIN="YOU.DOMAIN.NAME"
USER="username"
PASS="password"
PORT=443
AUTH=$(echo -n ${USER}:${PASS} | base64)

BIND_IP=0.0.0.0
CERT_DIR=/etc/letsencrypt
CERT=${CERT_DIR}/live/${DOMAIN}/fullchain.pem
KEY=${CERT_DIR}/live/${DOMAIN}/privkey.pem
sudo docker run -d --name gost \
    -v ${CERT_DIR}:${CERT_DIR}:ro \
    --net=host ginuerzh/gost \
    -L "http2://${BIND_IP}:${PORT}?auth=${AUTH}&cert=${CERT}&key=${KEY}&probe_resist=code:404&knock=www.google.com"

```

如无意外，你的服务就启动起来了。你可以使用如下命令在检查有没有启动成功：

```

sudo docker ps 来查看 gost 是否在运行。
netstat -nolp | grep 443 来查看 gost 是否在监听 443 端口。
sudo docker logs gost 来查看 gost 的日志。

```

你可以使用下面的命令验证你的 gost 服务是否正常。

```
curl -v "https://www.google.com" --proxy "https://DOMAIN" --proxy-user 'USER:PASS'
```

接下来就是证书的自动化更新。

可以使用命令 `crontab -e` 来编辑定时任务：

```

0 0 1 * * /usr/bin/certbot renew --force-renewal
5 0 1 * * /usr/bin/docker restart gost

```

这样，服务器就配置完成了。客户端请移动后面的客户端章节。

使用 Cloudflare 的注意事项

上述的方法并不支持 Cloudflare CDN，如果你想使用了 Cloudflare CDN，你需要使用 WebSocket 协议，如下所示，你需要使用 `mwss` 协议。

```
gost -L=mwss://username:password@:443?cert=/path/to/your/cert/file&key=/path/to/your/key/file
```

在 CloudFlare 上，请将 TLS/SSL 设置为 **完全**

如果你的客户端只能使用 `socks` 协议，你还要在客户端这边转一下：

```
gost -L socks://:YourLocalPort -F mwss://username:password@example.com:443
```

然后在其他软件中设置 socks5 代理即可

3.5 设置 ShadowSocks 服务（不推荐）

（注：ShadowSocks 被查的机率非常大，不推荐使用）

（如果有隧道转发，可以使用）

ShadowSocks 的 Docker 启动脚本（其中的 `SS_PORT` 和 `SS_PASSWD` 需要重新定义一下）

```

#!/bin/bash

SS_PORT=1984
SS_PASSWD=MyPasswd

sudo docker run -dt --name ss \
    -p ${SS_PORT}:${SS_PORT} mritd/shadowsocks \
    -s "-s 0.0.0.0 -p ${SS_PORT} -m aes-256-cfb -k ${SS_PASSWD} --fast-open"

```

3.6 设置 L2TP/IPSec 服务（不推荐）

(注: VPN方式被查的机率非常大, 不推荐使用)

L2TP/IPSec 的启动脚本, 其中的三个环境变量 USER , PASS 和 PSK 需要替换一下。

```
#!/bin/bash

USER=someone
PASS=password
PSK=psk_key

sudo docker run -d --privileged \
    -e PSK=${PSK} \
    -e USERNAME=${USER} -e PASSWORD=${PASS} \
    -p 500:500/udp \
    -p 4500:4500/udp \
    -p 1701:1701/tcp \
    -p 1194:1194/udp \
    siomiz/softethervpn
```

4. 客户端设置


4.1 电脑端设置

4.1.1 Chrome 代理设置

你可以使用 Chrome 插件 [SwitchyOmega](#)进行代理设置。在 SwitchyOmega 中, 你需要设置一个代理服务器。

打开 SwitchyOmega 的设置页面, 点击左边导航栏上的“New Profile”, 输入一个名字, 比如“代理”。

然后在 Protocol 中选择 HTTPS, 填上你的 Gost 的 VPS 服务器和端口号。

点后面的  按钮, 输入 Gost 服务器的用户名和密码。

具体的教程可以参看官方教程 - 《[最新 SwitchyOmega 使用教程快速入门篇](#)》

如果无法直接配置 HTTPS 代理, 具体原因可能是因为你设置了 probe_resist 以开启探测防御功能。这里, 你需要在服务器端设置 knock 参数 (参看 [用 Gost 设置 HTTPS 服务](#) 中的“注意”一节)

或是, 干脆使用 gost 客户端在本机启动一个 SOCKS 5的代理服务用来代替 (`gost -L socks5://:1080 -F 'https://USER:PASS@DOMAIN:443'`), 然后在 SwitchyOmega 配置代理为'127.0.0.1:1080'即可。

4.1.2 全局 Clash 代理设置

在电脑上, 使用 [Clash](#) 一个就可以了。Clash 支持很多翻墙协议: ShadowSocks(R), Vmess, Socks5, HTTP(s), Snell, Trojan。而且支持多个代理服务器的分组和负载均衡。

Clash 的客户端支持多种平台, 包括 Windows, MacOS, Linux 等, 你可以在 [Clash 的 Release 页面](#) 下载到最新的版本。这个是命令行版本, 但配置其实并不复杂。

另外, 有一些比较高级的功能, 如: 开启 Tun 网卡, 需要使用 [Clash Premium](#), 这个版本不收费, 但是是闭源的。

当然, 如果你要安装 GUI 版本, 你可以通过如下的项目安装:

[Clash for Windows](#). 注意: **这个软件是闭源软件。**

[Clash for MacOS](#)

[Clash for Android](#)

下面是 Clash 安装完后的配置目录结构:

```
|— clash                <- 建一个 clash 的目录
|   |— clash            <- 运行文件
|   |— config.yaml      <- 配置文件
|   |— Country.mmdb     <- IP地址库
|   |— ui               <- Clash 的 UI
|       |— index.html
|       |— ...
```

说明一下:

UI界面可以到 [haishah/yacd](#) 下载。放到clash的配置目录下 ui 目录下

一个是 Country.mmdb 这是IP地址的在哪个国家的数据库。你需要到这里下载 - [Country.mmdb](#) (当然, clash启动时, 会自动下载, 我这里给你一个手动下载的连接)

另一个是 config.yaml 文件, 这个文件详细解释可参看 - [官方Wiki](#)

下面是个示例:

```
port: 7890
socks-port: 7891
redir-port: 7892
mixed-port: 7893
ipv6: false
allow-lan: true
mode: Rule
log-level: info
external-controller: '0.0.0.0:9090'
external-ui: ui
secret: ''
tun:
  enable: true
  stack: system
  dns-hijack:
    - tcp://8.8.8.8:53
    - udp://8.8.8.8:53
dns:
  enable: true
  ipv6: false
  listen: 0.0.0.0:53
  default-nameserver:
    - 114.114.114.114
  #enhanced-mode: redir-host
  enhanced-mode: fake-ip #如果要玩netflix, 需要使用fake-ip
  fake-ip-range: 198.18.0.1/16
  nameserver:
    - 114.114.114.114
    - 223.5.5.5
    - tls://8.8.8.8:853
  fallback:
    - tls://8.8.8.8:853

# 两个代理服务器
proxies:
  # http
  - name: "https01"
    type: http
    server: https.server.domain
    port: 443
    username: user
    password: "password"
    tls: true # https
    skip-cert-verify: true
  - name: "https01"
    type: http
    server: https.server.domain
    port: 443
    username: user
    password: "passowrd"
    tls: true # https
    skip-cert-verify: true

# 配置 Group
proxy-groups:
  # 自动切换
  - name: "auto"
    type: url-test
    proxies:
      - us01_https
      #- us02_https
      #- hk_https
    # tolerance: 150
    url: 'https://www.google.com/'
    interval: 300
  # 按需选择 - 可以在UI上选择
  - name: "netflix"
    type: select
    proxies:
      - us01_https
      - us02_https
      - hk_https

rules:
```



```
# LAN
- DOMAIN-SUFFIX,local,DIRECT
- IP-CIDR,127.0.0.0/8,DIRECT
- IP-CIDR,172.16.0.0/12,DIRECT
- IP-CIDR,192.168.0.0/16,DIRECT
- IP-CIDR,10.0.0.0/8,DIRECT

# Netflix
- DOMAIN-SUFFIX,fast.com,netflix
- DOMAIN-SUFFIX,api-global.netflix.com,netflix
- DOMAIN-SUFFIX,netflix.com,netflix
- DOMAIN-SUFFIX,netflix.net,netflix
- DOMAIN-SUFFIX,nflxext.com,netflix
- DOMAIN-SUFFIX,nflximg.com,netflix
- DOMAIN-SUFFIX,nflximg.net,netflix
- DOMAIN-SUFFIX,nflxso.net,netflix
- DOMAIN-SUFFIX,nflxvideo.net,netflix

# 最终规则（除了中国区的IP之外的，全部翻墙）
- GEOIP,CN,DIRECT
- MATCH,auto
```

更多的规则网上可以找到很多，也可以参看这里：[SS-Rule-Snippet/LAZY_RULES/clash.yaml](https://ss-rule-snippet.github.io/LAZY_RULES/clash.yaml)

Note

除了 Clash 外，你可以安装原生的客户端。如：

1) VPN 客户端

对于L2TP/IPSec，几乎所有的客户端操作系统（无论是Windows/Mac/Linux的电脑，还是iPhone/Android）都支持，你可以自行Google。

[全平台配置 IPsec/L2TP VPN 客户端](#)

[Apple 官网：在 Mac 上设置 VPN 连接](#)

[Windows 7操作系统配置L2TP VPN方法](#)

2) Shadowsocks 客户端

对于 Shadowsocks 客户端，可以到这里查看 [Shadowsocks Clients](#)

MacOS 上你可以下载 [ShadowsocksX-NG](#)

Windows上你可以下载 [Shadowsocks-Windows](#)，需要先安装 [.NET Framework](#)

注：关于 Shadowsocks 客户端的配制，加密协议必须是跟 Shadowsocks 服务端相一致。可以自行约定在 Shadowsocks，如：之前章节示例中使用的是 aes-128-gcm。

3) Gost 客户端

Gost 并不支持智能代理（也就是该翻的时候翻，不用翻的时候不翻），所以我们可以重用 ShadowSocks 的客户端。

对于电脑来说，你同样可以 [下载 gost 程序](#)，然后使用下面的命令行：

```
# 此处的ss服务端加密协议(eg: aes-128-gcm)必须是ss客户端支持的协议
gost -L ss://aes-128-gcm:passcode@:1984 -F 'https://USER:PASS@DOMAIN:443'
```

这样用 gost 在你的本机启动了一个 ShadowSocks 的服务，然后，把请求转到你在上面配置的 HTTPS服务器上，这样就完成转接。



ShadowSocks Client 主要完成：自动设置操作系统代理服务器的 pac（自动设置翻墙或是不翻墙的路由）

这样，你的ShadowSocks客户端只需要简单的配置一个本机的 SS 配置就好了。

4.2 手机端设置

4.2.1 Android

[Clash for Android](#)

Shadowsocks + ShadowsocksGostPlugin

安装Shadowsocks (Google Play Store或[github页面](#))

下载[ShadowsocksGostPlugin](#)

配置

```
Server: $DOMAIN
Remote Port: 443
Password: gost
Encrypt Method: RC4-MD5
Plugin: 选择ShadowsocksGostPlugin
Configure: -F https://$USER:$PASS@$SS_HOST:$SS_PORT
```

其中 \$DOMAIN, \$USER, \$PASS 分别为服务端的域名/子域名, 用户名和密码。

4.2.2 iPhone

对于 Apple 上的 iPhone/iPad, 就比较麻烦了。因为相关的客户端在国内的 App Store 上全都被下架了。所以, 你需要注册一个美国的苹果ID, 然后 iTunes/App Store 用这个美区的ID登录 (不是退出iCloud, 而是退出App Store)。

关于如何注册美区 Apple ID账号, 你需要有两个前提条件:

你需要有一个美国的手机号码。
还需要有一个美国的 PayPal 账号。

关于如何搞到美国的手机号以及美国的 PayPal 账号以及其必要性, 可以参看[5. 美国手机和支付](#)

然后, 你就可以用电脑登录 <https://appleid.apple.com/> 全新注册一个帐号, 注册的时候, 你需要选择美国的地区, 然后, 你需要用你的美国手机号码接收验证码, 最后, 你需要用你的美国 PayPal 账号来绑定你的信用卡。

iPhone 上的客户端我推荐使用下面这两个 (需要付费), 这两个客户端都支持很多协议, 如: ShadowSocks, HTTP(s), VMess, Socks5, 等。

[ShadowRocket](#), 又叫小火箭, 其中使用 HTTPS 的代理, 配置上就好了。

[Quantumult](#). 这个工具以前用的不多, 最近 1-2 年都在有这个工具, 感觉还是很好用的。对了, 大家一定要不要跟 Quantumult X 搞混, 我个人觉得 Quantumult X 非常难用。

Note

[Potatso](#) 作为 shadowsocks 的客户端也是可以的, 而且免费, 但是无法使用 Gost 的 HTTPS 代理, 所以, 我不推荐使用。

4.3 平板端设置 (服务端使用gost代理)

4.3.1 iPad

ShadowRocket: 打开app, 点击右上角 + 号添加服务器, 配置如下几项即可 (其余条目采用默认值即可)

```
Type: HTTPS
Address: $DOMAIN
Port: 443
User: $USER
Password: $PASS
```

其中, \$DOMAIN 为服务器的域名/子域名, \$USER 与 \$PASS 分别为服务端启动gost代理时设定的用户名和密码

5. 美国手机和支付

现在越来越多 APP 需要使用海外的手机和支付, 所以, 你需要有一个美国的手机号码和美国的支付方式。

5.1 美国手机

美国的手机号目前最好的方式是购买 [Ultra Mobile 的 PayGo](#) 月租 3 美金, 你可以上淘宝上购买。这个卡是真实的美国手机号码, 在中国漫游时, 短信和接听电脑会产生漫游费用, 所以, 你一定要[开启 WiFi Calling](#)。有时候, 你的网络可能无法让你开启 WiFi Calling, 这个可能是运营上 block 了相关路由, 你可能需要设置你的路由器, 这里[有篇文章](#)你可以了解一下。

这里不推荐使用 [Google Voice](#)。因为你注册 Google Voice 的时候也需要一个实体卡, 如果你有朋友在美国的话, 你可以借用朋友的手机帮你生成一个, 但是就算是生成了, 美国也有很多 APP 无法接收 Google Voice 的验证码, 因为风控的问题不支持虚拟运营商。

5.2 美国支付

关于美国的支付，你可以注册一个美国的 PayPal，这个对于 iPhone/iPad 转美区 App Store 来说是很方便的。

你可以使用你在国内的币种（支持 VISA/MasterCard）的信用卡来开国际 PayPal 帐号。开通国际 PayPal 帐号需要以下几个条件：

- 身在美国（系统会自动检测IP地址）
- 有效美国地址（最好是私人地址，必要时能提交地址证明）
- 有效美国手机号码（必需是能接收短信）

注意：如果你没有美国的 SSN 或是美国的银行帐号，你是可以支付的，但是不能收款的。所以，千万不要用来收款！

但是，现在很多美国的公司都不使用 PayPal 做他们的支付网关，而是使用 [Stripe](#)，这两家公司是竞争关系，所以，你是不能使用 PayPal 来支付的，而 Stripe 的风控比较严格，所以你需要有一个美国的信用卡。

拥有一个美国的信用卡是一件极其麻烦的事，因为你首先需要有一个美国的银行帐户，而美国的银行帐户也分对公和对私，对公的帐户是需要有美国的公司才能开的，对私的帐户是需要你有美国的 SSN 才能开的。所以，这个事情就变得很复杂了。

下面是一些可行的操作，你可以参考一下：

- 找你美国的朋友，让他们帮你开一个虚拟信用卡。
- 通过支付加密货币USDT到 [Depay](#) 上开通虚拟信用卡，操作复杂。
- 找淘宝店帮你支付可能会是一个比较好的选择，代价就是你需要把你的帐号和密码给他们，这个风险你需要自己承担。
- 有美国B1/B2旅游签证就可以在美国的银行开户。

如果大家有更好的方式，欢迎提交PR。

6. 流量伪装和防探测

无论你用VPN，SS，SSR，都有可以被识别，我实践下来的结果是，**只有使用 HTTP over TLS 的样子，才会跟正常的流量混在一起，很难被识别**，所以，目前来说，使用Gost 的 HTTPS 的方式 再加上防探测的机制，基本上来说，在网络四层上看到的都是TLS的包，很难被识别。另外，用V2Ray客户端 + Nginx + V2Ray服务端的方式也是可以，不过配置起来比较复杂。（注：关于流量识别和探测的细节，你可以看看[这篇文章](#)）

也就是说，在启动 Gost 的时候，需要设置 probe_resist 参数，关于这个参数，我们在上面的章节中已经提到过了，这里再说一下。

你可以先创建一个 如下的 /var/www/html/index.html 的文件：

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

然后，在 Gost 的启动上加上这个参数：probe_resist=file:/var/www/html/index.html，这样，当有人探测你的端口的时候，会返回这个文件的内容，这样，就会让探测者以为你的端口是一个 Web 服务器，而不是一个代理服务器。

完整的启动脚本如下：

```
#!/bin/bash

# 下面的四个参数需要改成你的
```

```
DOMAIN="YOU.DOMAIN.NAME"
USER="username"
PASS="password"
PORT=443

BIND_IP=0.0.0.0
CERT_DIR=/etc/letsencrypt
CERT=${CERT_DIR}/live/${DOMAIN}/fullchain.pem
KEY=${CERT_DIR}/live/${DOMAIN}/privkey.pem
sudo docker run -d --name gost \
    -v ${CERT_DIR}:${CERT_DIR}:ro \
    --net=host ginuerzh/gost \
    -L "http2://${USER}:${PASS}@${BIND_IP}:${PORT}?cert=${CERT}&key=${KEY}&probe_resist=file:/var/www/html/index.html&knock=www.google.com"
```

7. 针对 IP 被封的解决方案

花钱购买的 VPS 即便做了流量伪装依然有很大的几率 IP 被封锁，大多 VPS 服务商并不提供更换 IP 的服务，使用 CDN 可以让被封锁的 VPS 继续发挥翻墙功能。

Cloudflare 是一个 CDN 服务商，目前国内依然能正常的访问，可以作为跳板来实现翻墙。

注册 Cloudflare 帐号，并有一个空闲域名（需要使用二级域名），交给 Cloudflare 托管并将域名指向被封的 VPS IP。Cloudflare 只需免费方案足以，不必花钱。

使用 Cloudflare 进行套壳需要有如下的条件：

无论是 V2Ray 还是 gost，都需要使用 WebSocket 传输协议

Cloudflare 的端口号只能代理有限的几个，推荐使用：80，8080，443 或 8443

客户端这边需要使用对应的客户端，比如 gost 或 v2ray 的客户端进行 WebSocket 的连接

关于优选IP，可以手动更改本地hosts文件指向最佳IP。

目前支持 WebSocket 的免费 CDN 似乎只有 Cloudflare 一家，国内 CDN 服务商既不支持也不安全，不要考虑了。如果有更好的服务商欢迎补充。

网络延迟比直连增加不少，如果是频繁操作会很痛苦。网络带宽如果运气好可能比直连还优化了，用来看 Youtube 搞不好更流畅。

8. 家用透明网关

8.1 OpenWRT 路由器

所谓透明网关的意思是，一切都交给网关来做。最好的方式是你需要一个 OpenWRT 的路由器，推荐使用华硕的路由器，贵是贵一些，但是这几年用下来，非常不错。我用的是 **华硕（ASUS）RT-AC68U 1900M AC 双频智能无线路由器**。

路由器买来后，要刷一下固件。首先 Asuswrt 是华硕公司给他的路由器所开发的固件。Asuswrt-merlin是一个对Asuswrt固件二次开发进行各种改进和修正的项目。源代码在这里：<https://github.com/RMerl/asuswrt-merlin>

不必担心把路由器刷废了，华硕的路由器可以让你一键重置回来

1) **下载固件**。先到 <https://www.asuswrt-merlin.net/download> 下载相应的固件，并解压。（我下载的是 RT-AC68U_380.61_0.zip）

2) **升级固件**。登录到你的路由器后台 <http://192.168.1.1/>，在 系统管理 -> 固件升级 中上传固件文件（我上传的是：RT-AC68U_380.61_0.trx）

3) **打开 JFFS 分区**。系统管理 -> 系统设置 -> Persistent JFFS2 partition

Format JFFS partition at next boot - 否
Enable JFFS custom scripts and configs - 是

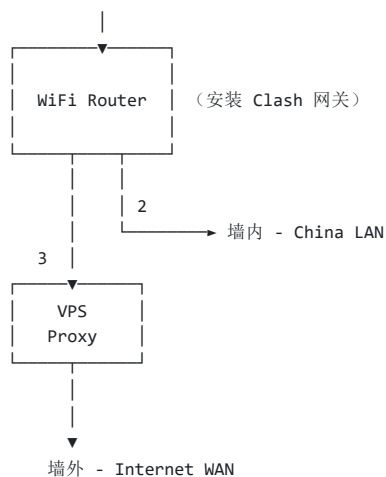
4) **打开 ssh 登录**。系统管理 -> 系统设置 -> SSH Daemon

Allow SSH password login - 是

接下来，在 WiFi 路由器上安装 Clash，就可以了。

大概的示意图如下所示。





8.2 通过树莓派做旁路网关

如果你的路由器不能刷 OpenWRT，也就是没法通过SSH登录上去装软件，你就用一个别的设备。比如用一个树莓派。我正好有一个很老旧的树莓派，刷了一个老旧的 Debian 7.5的操作系统。

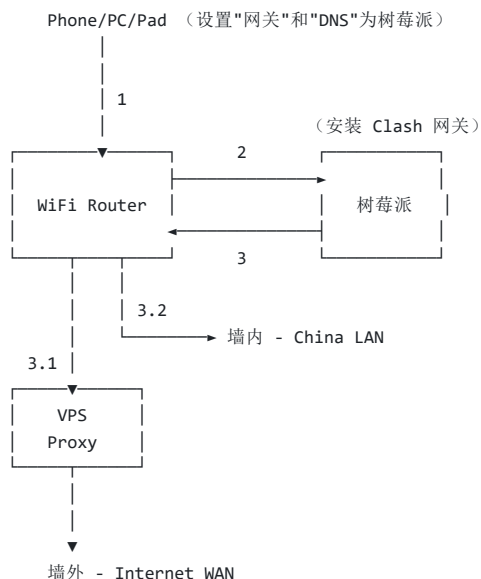
把它连上你的路由器上，然后，

你需要把你设备上的IP地址、网关和DNS服务器都要手动设置到这个树莓派上。

于是，所有的路由就会通过路由器转到树莓派上，再由树莓派决定是否要走代理。

大概的示意图如下所示。

- 1 --> 2 是设备把所有的请求都发给树莓派。
- 3 --> 3.1 或 3.2 是由树莓派来决定是否翻墙。



8.3 安装 Clash

Clash 的 Github项目是: [Dreamacro/clash](https://github.com/Dreamacro/clash)，在它的 Release 页面上，你可以找到相关的下载。（注：在本文更新的时候，如果你需要支持 Tun，你需要下载 Clash 的 [Premium 版本](#)

Clash 支持很多翻墙协议：ShadowSocks(R), Vmess, Socks5, HTTP(s), Snell, Trojan。

在你的 OpenWRT 或 树莓派 下用 `uname -m` 查看一下你的硬件架构是什么的，比如，我的是华硕和树莓派都是 `armv7l` 的，所以，需要下载 `clash-linux-armv7-....` 的版本（注：根据 clash 官方仓库 [Dreamacro/clash#189](https://github.com/Dreamacro/clash#189) 系列固件不适用 `armv7l` 架构的 AC68U，需选择 `armv5`）。下载完解压后，加个可执行权限 `chmod +x clash` 就可以运行了，不过，还差一个界面和两个配置文件，它们的目录关系如下：

```

├─ clash          <- 建一个 clash 的目录
├─ └─ clash       <- 运行文件
├─ └─ config.yaml <- 配置文件

```

```

|   | Country.mmdb    <- IP地址库
|   |   | ui          <- Clash 的 UI
|   |   |   | index.html
|   |   |   | ...
|

```

UI界面可以到 [haishah/yacd](https://github.com/haishah/yacd) 下载。放到clash的配置目录下 ui 目录下

一个是 Country.mmdb 这是IP地址的在哪个国家的数据库。你需要到这里下载 - [Country.mmdb](https://github.com/Lkened/country.mmdb) (当然, clash启动时, 会自动下载, 我这里给你一个手动下载的连接)

另一个是 config.yaml 文件, 这个文件详细解释可参看 - [官方Wiki](https://github.com/Dreamacro/clash/wiki)

下面是个示例:

```

port: 7890
socks-port: 7891
redir-port: 7892
mixed-port: 7893
ipv6: false
allow-lan: true
mode: Rule
log-level: info
external-controller: '0.0.0.0:9090'
external-ui: ui
secret: ''
tun:
  enable: true
  stack: system
  dns-hijack:
    - tcp://8.8.8.8:53
    - udp://8.8.8.8:53
dns:
  enable: true
  ipv6: false
  listen: 0.0.0.0:53
  default-nameserver:
    - 114.114.114.114
  #enhanced-mode: redir-host
  enhanced-mode: fake-ip #如果要玩netflix, 需要使用fake-ip
  fake-ip-range: 198.18.0.1/16
  nameserver:
    - 114.114.114.114
    - 223.5.5.5
    - tls://8.8.8.8:853
  fallback:
    - tls://8.8.8.8:853

# 两个代理服务器
proxies:
  # http
  - name: "https01"
    type: http
    server: https.server.domain
    port: 443
    username: user
    password: "password"
    tls: true # https
    skip-cert-verify: true
  - name: "https01"
    type: http
    server: https.server.domain
    port: 443
    username: user
    password: "passowrd"
    tls: true # https
    skip-cert-verify: true

# 配置 Group
proxy-groups:
  # 自动切换
  - name: "auto"
    type: url-test
    proxies:
      - us01_https
      #- us02_https
      #- hk_https
    # tolerance: 150

```

```

url: 'https://www.google.com/'
interval: 300
# 按需选择 - 可以在UI上选择
- name: "netflix"
  type: select
  proxies:
    - us01_https
    - us02_https
    - hk_https

rules:
# LAN
- DOMAIN-SUFFIX,local,DIRECT
- IP-CIDR,127.0.0.0/8,DIRECT
- IP-CIDR,172.16.0.0/12,DIRECT
- IP-CIDR,192.168.0.0/16,DIRECT
- IP-CIDR,10.0.0.0/8,DIRECT

# Netflix
- DOMAIN-SUFFIX,fast.com,netflix
- DOMAIN-SUFFIX,api-global.netflix.com,netflix
- DOMAIN-SUFFIX,netflix.com,netflix
- DOMAIN-SUFFIX,netflix.net,netflix
- DOMAIN-SUFFIX,nflxext.com,netflix
- DOMAIN-SUFFIX,nflximg.com,netflix
- DOMAIN-SUFFIX,nflximg.net,netflix
- DOMAIN-SUFFIX,nflxso.net,netflix
- DOMAIN-SUFFIX,nflxvideo.net,netflix

# 最终规则（除了中国区的IP之外的，全部翻墙）
- GEOIP,CN,DIRECT
- MATCH,auto

```

更多的规则网上可以找到很多，也可以参看这里：[SS-Rule-Snippet/LAZY_RULES/clash.yaml](https://github.com/SS-Rule-Snippet/LAZY_RULES/blob/master/clash.yaml)

这个时候你就可以启动 clash 了：

```
/path/to/clash/cash -d /path/to/clash &
```

然后，你就可以把你的上网设备上的 路由网关 和 DNS 服务器都手动地配置成这个网关就好了（OpenWRT应该不用配置了，树莓派的方式需要手动配置一下）

8.4 设置 iptables 转发

```

iptables -t nat -N CLASH
iptables -t nat -A CLASH -d 10.0.0.0/8 -j RETURN
iptables -t nat -A CLASH -d 127.0.0.0/8 -j RETURN
iptables -t nat -A CLASH -d 169.254.0.0/16 -j RETURN
iptables -t nat -A CLASH -d 172.16.0.0/12 -j RETURN
iptables -t nat -A CLASH -d 192.168.0.0/16 -j RETURN
iptables -t nat -A CLASH -d 224.0.0.0/4 -j RETURN
iptables -t nat -A CLASH -d 240.0.0.0/4 -j RETURN
iptables -t nat -A CLASH -p tcp -j REDIRECT --to-ports 7892

```

然后，你可以保存一下这些 iptables 的规则

```
iptables-save > /etc/iptables.up.rules
```

编辑 /etc/network/if-pre-up.d/iptables，在网卡启动的时候加载这些规则

```

#!/bin/sh
/sbin/iptables-restore < /etc/iptables.up.rules

```

然后，再 `chmod +x /etc/network/if-pre-up.d/iptables` 加上可执行权限就好了。

9. 数据中心透明网关

这里仅针对 AWS 进行说明，其它云平台应该大同小异，大家可以补充。

9.1 AWS 网络构建

构建一个 172.20.0.0/16 的 VPC，分成两个子网：

有公网IP的公有子网 - 172.20.1.0/24

无公网IP的私有子网 - 172.20.2.0/24

在公有子网里创建 [EC2 NAT Instance](#)

创建时，指定私网IP为 172.20.1.1

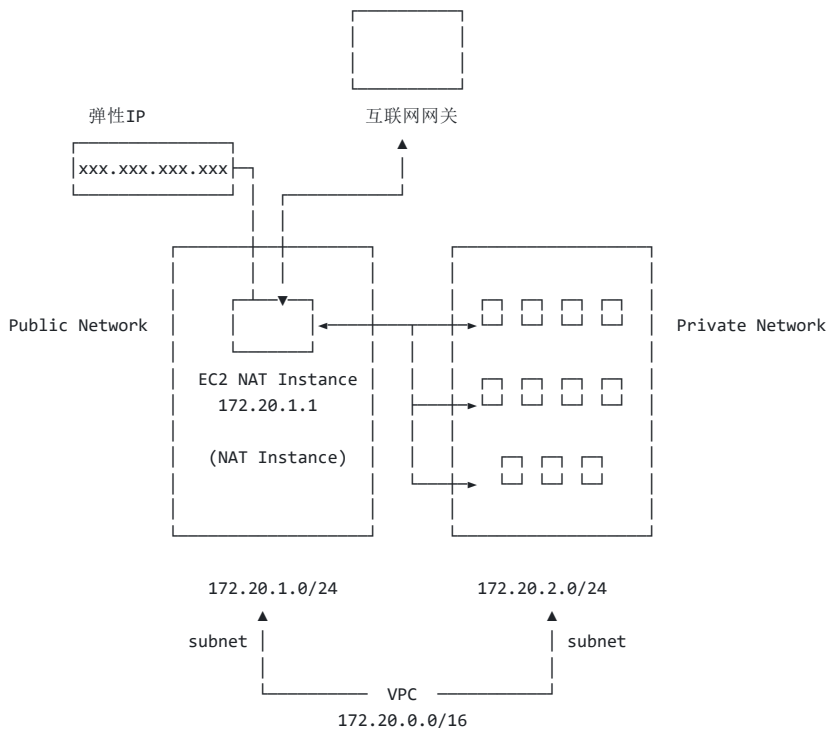
(Option) 为该实例分配弹性IP，可成为外网访问内网的跳板机

建立路由规则

创建“互联网网关”，并把“互联网网关”添加到公有子网 172.20.1.0/24 的路由表中

把 EC2 NAT Instance 172.20.1.1 添加到私有子网 172.20.2.0/24 的路由表中。

于是整个网络就如下所示。



注：你需要认真的按照 [EC2 NAT Instance](#) 的文档进行设置这个NAT实例。尤其需要设置下面几项：

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -A FORWARD -i eth0 -j ACCEPT
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

顺便科普一下：

net.ipv4.ip_forward 是内核参数，主要是用来把Linux当成路由器来用的参数。一般来说，一个路由器至少要有两个网络接口，一个是WAN，的一个是LAN的，为了让LAN和WAN的流量相通，需要进行内核上路由。

iptables -A FORWARD -i eth0 -j ACCEPT 通行所有需要转发的包，只有机器成为一个路由器时，需要在两个网卡间进行网络包转发时，才需要配置这条规则。

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE 关键字 MASQUERADE 意思是“伪装”，NAT的工作原理是就像是一个宿舍收发室对学生宿舍一样，学生宿舍的地址外部不可见，邮递员只看得见整栋宿舍收发室的地址，邮递员把快递交给收发室，收发室再把快递转给学习宿舍（反之，如果学生要对外寄邮件，也是先到收发室，收发室传给邮局）。现在的问题是，所有的学生宿舍如何才能参与到任何快递的通信中，如果把学生宿舍地址发到外部，则没人能把信送回来。如果这个收发室是个自动化的机器人，他要干的事就是，把学生宿舍的地址换成收发室地址。这就是 MASQUERADE 的意思——**来自具有接收方 IP 地址的本地网络到达 Internet 某处的数据包必须进行修改，也就是让发送方的地址等于路由器的地址。**

9.2 安装 Clash

在 EC2 NAT Instance 上安装 clash 透明网关，安装配置参看 [8.3 安装 Clash](#)，基本一致。

注：在实际操作中，没有设置 iptables 转发规则

9.3 配置私有子网中的 EC2

只需要配置 `/etc/resolv.conf` 文件，把 EC2 NAT Instance 加入其中。如：

```
# /etc/resolv.conf
nameserver 172.20.1.1 #<--- 透明网关 EC2 NAT 实例
nameserver 172.20.0.2 #<--- AWS 的 DNS 服务
search [zone].compute.internal
```

Note

新版的 Ubuntu 已经把 DNS Resolver 托管给了 systemd 的 `systemd-resolved` 服务，所以要把 `/etc/resolv.conf` 文件改成软链接，指向 `systemd-resolved` 的配置文件，如：

```
sudo ln -sf /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

然后，在 `/etc/systemd/resolved.conf` 文件中，把 DNS 和 Domains 配置项加上，如：

```
DNS=172.20.0.2
```

9.4 私有子网中的 Kubernetes

K8s 里有两组 CoreDNS 部署和配置，一组是边缘的（或是叫本地的），一组是中心的。

边缘的 Pod 名叫 `nodelocaldns`，侦听在本机。如：169.254.25.10:53

中心的 Pod 名叫 `coredns`，侦听在 cluster IP 上，如：10.233.0.3:53

边缘的规则会把k8s的域名 `cluster.local`，`in-addr.arpa` `ip6.arpa` 转给中心的 CoreDNS 处理，其它的交给本地的 `/etc/resolv.conf` 处理。

Kubernetes 会把如下内容打到 Pod 里的 `/etc/resolv.conf`

```
nameserver 169.254.25.10
search default.svc.cluster.local svc.cluster.local cluster.local cn-northwest-1.compute.internal
options ndots:5
```

查看一下 `nodelocaldns` 的配置：

```
$ kubectl get cm nodelocaldns -n kube-system -o yaml
```

我们可以看到，除了 K8s 自己的域名外，其它的都交给了本机的 `/etc/resolv.conf`，如下所示：

```
.:53 {
  errors
  cache 30
  reload
  loop
  bind 169.254.25.10
  forward . /etc/resolv.conf # <--- 注意这条语句
  prometheus :9253
}
```

然而，本机的 `/etc/resolv.conf` 里有两个 DNS，一个是我们的透明网关，一个是AWS的。而 CoreDNS 的 `forward` 策略是随机挑选，所以，这样的会导致，时而交给AWS处理，时而交给我们自己的clash处理。最终导致IP解析紊乱。

通过以下命令进行修改：

```
$ kubectl edit cm nodelocaldns -n kube-system
```

修改如下：（AWS的归 172.20.0.2，其它的走我们自己的网关）

```
+   compute.internal:53 {
+     errors
+     cache 30
+     reload
+     loop
+     bind 169.254.25.10
+     forward . 172.20.0.2
```

```
+     prometheus :9253
+   }
+   .:53 {
+       errors
+       cache 30
+       reload
+       loop
+       bind 169.254.25.10
-   forward . /etc/resolv.conf
+   forward . /etc/resolv.conf {
+       policy sequential
+   }
+   prometheus: 9253
+ }
```

退出保存后，等大约30秒左右配置就会生效。

10. 代理技巧

看到这里，相信已经能够按照上面的教程搭建好自己的上网环境，但是灵活的应用网络，你还需要了解一些技巧，比如 SOCKS 协议, http 隧道和 ssh 网络隧道等。

[SOCKS 协议](#)

[HTTP 隧道](#)

10.1 HTTP 隧道

常见的软件 curl, git, wget 都能通过设置 HTTP_PROXY, HTTPS_PROXY, NO_PROXY 来配置一个网络代理，NO_PROXY 用来配置不需要代理的主机(多个用逗号隔开)，那么我们就可以编写一个 bash 函数来运行需要走代理的命令：

```
with_proxy(){
    HTTPS_PROXY=http://127.0.0.1:7890 HTTP_PROXY=http://127.0.0.1:7890 "$@"
}
```

把上面的 127.0.0.1:7890 改成你自己的网络代理，将上面脚本写入到 ~/.bashrc 中，source ~/.bashrc 后就能使用 with_proxy 这个函数了，比如我想要使用代理网络下载一个文件 with_proxy wget https://...., 想要使用代理网络从 github clone 一个项目 with_proxy git clone https://..., 当我们不用 with_proxy 这个函数的时候命令是不会走代理的，如果在 windows 上你也想要使用这样的功能，可以使用这个项目 [with-env](#)。

另外，你也可以使用如下的两个 alias:

```
SOCKS="socks5://127.0.0.1:1085"
alias proxy="export http_proxy=${SOCKS} https_proxy=${SOCKS} all_proxy=${SOCKS}"
alias unproxy='unset all_proxy http_proxy https_proxy'
```

这样，你就可以在需要代理的时候输入 proxy，不需要的时候输入 unproxy。

10.2 SSH 隧道

另外，我们可以使用 SSH Tunnel 来建立 SOCKS5 的代理（假设本地电脑无法访问，但是某台可以 SSH 的服务器能够访问外网，那么我们就可以使用如下的命令来建议翻墙代理：

```
ssh -D 1080 -qCN username@server:port
```

解释：

- D：本机SOCKS 服务端口
- q：quiet 模式，没有输出
- C：数据压缩，可以节约一些带宽
- N：不运行远程命令，只做端口转发

登录成功以后，本地 1080 端口会开启一个 socks5 协议的代理，只要配置好代理就能使用这个端口上网。

```
with_proxy(){
    HTTPS_PROXY=socks5://127.0.0.1:1080 HTTP_PROXY=socks5://127.0.0.1:1080 "$@"
}
```

如果是浏览器，配置好 SwitchyOmega 插件也能实现上外网。

10.3 Github / Git SSH 代理

现在访问 Github 速度很慢甚至不通，我们可以使用代理来加速，首先我们需要配置好代理，然后配置好 git 的代理，这样就能加速 git clone 和 git push 了。

当你使用 `git clone ssh://[user@]server/project.git` 或是 `git clone [user@]server:project.git` 的时候，你是在使用 SSH 协议。你需要配置 `~/.ssh/config` 来使用代理，比如我的本地有一个 Sock5 代理，端口是 1085，那么我可以这样配置：

```
### github.com
Host github.com
  Hostname github.com
  ProxyCommand nc -x localhost:1085 %h %p
# git-for-windows 下可以用 connect 代替 nc
# ProxyCommand connect -S localhost:1085 %h %p
```

10.4 Cloudflare Warp 原生 IP

很多我们需要访问的网站都需要使用“原生 IP”，比如：[Disney+](#)，[ChatGPT](#)，[New Bing](#) 等。

所谓“原生 IP”就是指该网站的 IP 地址和其机房的 IP 地址是一致的，但是，很多 IDC 提供商的 IP 都是从其它国家调配来的，这导致我们就算是翻墙了，也是使用了美国的 VPS，但是还是访问不了相关的服务。所以，我们需要使用 Cloudflare Warp 来访问这些网站。

下面有几种安装方式：

全局模式。这种模式下，所有流量都会通过 Cloudflare 的网络，相当于 VPN。

代理模式。通过在服务器本机启动一个 SOCKS5 代理，然后把需要的流量转发到这个代理上。

10.4.1 WARP 模式

WARP 模式是 Cloudflare 提供了一种全局代理模式，就是一个客户端的 VPN，它会将你的所有流量都通过 Cloudflare 的网络，这样就能访问到原生 IP 了。

你可以使用这个一键安装脚本来快速完成安装 <https://github.com/P3TERX/warp.sh>

下载脚本

```
wget git.io/warp.sh
chmod +x warp.sh
```

运行脚本 中文菜单

```
./warp.sh menu
```

先后执行如下安装：

- 1 - 安装 Cloudflare WARP 官方客户端
- 4 - 安装 WireGuard 相关组件
- 7 - 自动配置 WARP WireGuard 双栈全局网络

Note

如果没有 IPv6 网络，那么第 7 步可以换成第 5 步 自动配置 WARP WireGuard IPv4 网络，或是执行 `./warp.sh 4`。

你需要备份一下你的帐号和配置文件，在 `/etc/warp/` 目录下，主要是两个文件，一个是 `wgcf-account.toml`，一个是 `wgcf-profile.conf`

这个脚本会启动两个 Systemd 服务，一个是 `warp-svc`，另一个是 `wg-quick@wgcf`。第一个是 Cloudflare Warp 的服务，第二个是 WireGuard 的服务。

你可以使用 `./warp.sh status` 来查看服务的状态，如果正常的话，会显示如下的信息。如果你的服务没有正常启动，那么会自动 `disable wg-quick@wgcf`，如果这样的话，你可以使用 `./warp.sh d`（IPv4 + IPv6 双栈网络）或是 `./warp.sh 4`（IPv4 网络）来重头走一遍所有的安装过程。

```
WireGuard      : Running
IPv4 Network   : WARP
IPv6 Network   : WARP
```

使用 `curl ipinfo.io` 命令来检查你的 IP 地址，如果显示的是 Cloudflare 的 IP 地址，那么恭喜你，你已经成功了。如：

```
{
  "ip": "104.28.227.191",
  "city": "Los Angeles",
  "region": "California",
  "country": "US",
  "loc": "34.0522,-118.2437",
  "org": "AS13335 Cloudflare, Inc.",
  "postal": "90009",
  "timezone": "America/Los_Angeles",
  "readme": "https://ipinfo.io/missingauth"
}
```

10.4.2 代理模式

如果 WARP 模式安装不能成功，那么你可以使用如下的代理模式。代理模式也就是通过一个代理来访问 Cloudflare WARP 的服务，这样就可以访问到原生 IP 了。

Note

下面的步骤主要使用 Cloudflare 的官方客户端，也许会随时间流逝导致过时，你可以参考 [Cloudflare WARP 的官方文档](#)

1) 安装软件包

如果是 Ubuntu，那么可以使用以下命令在添加安装源：

```
sudo apt install curl
curl https://pkg.cloudflareclient.com/pubkey.gpg | gpg --yes --dearmor --output /usr/share/keyrings/cloudflare-warp-archive-keyring.g
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/cloudflare-warp-archive-keyring.gpg] https://pkg.cloudflareclient.com/ $(lsb_rele
```

添加完源后，安装 Cloudflare WARP 客户端：

```
sudo apt update
sudo apt install cloudflare-warp
```

Note

安装过程中，可能会出现如下错误：

```
The following packages have unmet dependencies:
cloudflare-warp : Depends: nftables but it is not going to be installed
                  Depends: gnupg2 but it is not going to be installed
                  Depends: desktop-file-utils but it is not going to be installed
                  Depends: libnss3-tools but it is not going to be installed
linux-headers-aws : Depends: linux-headers-5.3.0-1028-aws but it is not going to be installed
E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a solution).
```

你先执行 `sudo apt --fix-broken install`，然后再执行 `sudo apt install cloudflare-warp` 即可。

2) 配置 Cloudflare WARP

如果是第一次安装，你需要先注册一个帐号。其注册信息会在这里 `/var/lib/cloudflare-warp/reg.json`

```
warp-cli register
```

然后设置代理模式，这点非常重要，因为默认是 WARP 模式，这个会把你的整个 VPS 带到 Cloudflare 的 VPN 网络中，那么就会出现无法连接的情况。

```
warp-cli set-mode proxy
```

然后，设置永久连接模式。

```
warp-cli enable-always-on
```

配置完后，你可以使用 `warp-cli settings` 来查看配置。你也可以通过查看配置文件来看是否配置成功，配置文件在 `/var/lib/cloudflare-warp/settings.json`。

3) 连接 Cloudflare WARP

使用如下命令来连接 Cloudflare WARP:

```
warp-cli connect
```

你可以使用 `warp-cli status` 来查看连接状态。如:

```
> warp-cli status
Status update: Connected
Success
```

连接成功后, 你可以会在本地有一个 Socks5 代理, `127.0.0.1:40000`, 你可以使用如下命令来查看:

4) 验证 Cloudflare WARP

你可以使用如下命令来验证是否成功:

```
curl -x "socks5://127.0.0.1:40000" ipinfo.io
```

如果输出出现如下的信息, 那么恭喜你, 你已经成功了

```
"ip": "104.28.247.70",
"org": "AS13335 Cloudflare, Inc."
```

5) 配置 Gost 路由转发

你可以使用如下命令来启动 gost 转发规则。下面的命令行意思是, 把本地的 8080 端口转发到 Cloudflare WARP 的 Socks5 代理上。

```
gost -L "http://:8080" -F "socks5://127.0.0.1:40000"
```

当然, 上面的配置是不够好的, 我们最好使用有证书的 HTTPS 代理。这里的内容参见于 前的面 [3.3 用 Gost 设置 HTTPS 服务](#)。

为了使用两种不同的代理, 你可以启动两个 gost 服务:

一个是通过 443 端口直接代理

另一个是通过 8443 端口转发到 Cloudflare WARP 的 Socks5 代理上

```
# 下面的四个参数需要改成你的
DOMAIN="YOU.DOMAIN.NAME"
USER="username"
PASS="password"
PORT=443

BIND_IP=0.0.0.0
CERT_DIR=/etc/letsencrypt
CERT=${CERT_DIR}/live/${DOMAIN}/fullchain.pem
KEY=${CERT_DIR}/live/${DOMAIN}/privkey.pem
sudo docker run -d --name gost \
  -v ${CERT_DIR}:${CERT_DIR}:ro \
  --net=host ginuerzh/gost \
  -L "http2://${USER}:${PASS}@${BIND_IP}:${PORT}?cert=${CERT}&key=${KEY}&probe_resist=code:404&knock=www.google.com"

sudo docker run -d --name gost-warp \
  -v ${CERT_DIR}:${CERT_DIR}:ro \
  --net=host ginuerzh/gost \
  -L "http2://${USER}:${PASS}@${BIND_IP}:8443?cert=${CERT}&key=${KEY}&probe_resist=code:404&knock=www.google.com" \
  -F "socks5://localhost:40000"
```

Note

你也可以使用 gost 的 `bypass` 参数来让相应的域名的流量转发到 Cloudflare WARP 的 Socks5 代理上。如: -

```
-F "socks5://localhost:40000bypass=*.google.com,google.com&probe_resist=code:404&knock=www.google.com"
```