在国内开发[微信公众号](https://mp.weixin.qq.com)、[企业号](http://qydev.weixin.qq.com/)以及做前端开发的朋友想必对[ngrok](https://github.com/inconshreveable/ngrok)都不陌生吧，就目前来看，ngrok可是最佳的在内网调试微信服务的tunnel工 具。记得今年春节前，ngrok.com提供的服务还一切正常呢，但春节后似乎就一切不正常了。ngrok.com无法访问，ngrok虽然能连上 ngrok.com提供的服务，但微信端因为无法访问ngrok.com，导致消息一直无法发送到我们的服务地址上，比如xxxx.ngrok.com。 这一切都表明，ngork被墙了。没有了ngrok tunnel，一切开始变得困难且没有效率起来。内网到外部主机部署和调试是一件慢的让人想骂街的事情。

ngrok不能少。ngrok以及其服务端ngrokd都是开源的，之前我也知道通过源码可以自搭建ngrok服务。请求搜索引擎后，发现国内有个朋友已经搭建了一个www.tunnel.mobi的ngrok公共服务，与ngrok.com类似，我也实验了一下。

编写一个ngrok.cfg，内容如下：

server\_addr: "tunnel.mobi:44433"  
trust\_host\_root\_certs: true

用ngrok最新客户端1.7版本执行如下命令：

$ngrok -subdomain tonybaiexample -config=ngrok.cfg 80

可以顺利建立一个tunnel，用于本机向外部提供"tonybaiexample.tunnel.mobi"服务。

Tunnel Status                 online  
Version                       1.7/1.7  
Forwarding                    http://tonybaiexample.tunnel.mobi -> 127.0.0.1:80  
Forwarding                    https://tonybaiexample.tunnel.mobi -> 127.0.0.1:80  
Web Interface                 127.0.0.1:4040  
# Conn                        0  
Avg Conn Time                 0.00ms

而且国内的ngrok服务显然要远远快于ngrok.com提供的服务，消息瞬间即达。

但这是在公网上直接访问的结果。放在公司内部，我看到的却是另外一个结果：

Tunnel Status                 reconnecting  
Version                       1.7/  
Web Interface                 127.0.0.1:4040  
# Conn                        0  
Avg Conn Time                 0.00ms

我们无法从内网建立tunnel，意味着依旧不方便和低效，因为很多基础服务都在内网部署，内外网之间的交互十分不便。但内网连不上tunnel.mobi也是个事实，且无法知道原因，因为看不到server端的连接错误日志。

于是我决定自建一个ngrok服务。

**一、准备工作**

搭建ngrok服务需要在公网有一台vps，去年年末曾经在Amazon申请了一个体验主机[EC2](https://aws.amazon.com/cn)，有公网IP一个，这次就打算用这个主机作为ngrokd服务端。

需要一个自己的域名。已有域名的，可以建立一个子域名，用于关联ngrok服务，这样也不会干扰原先域名提供的服务。(不用域名的方式也许可以，但我没有试验过。）

搭建的参考资料主要来自下面三个：  
1) ngrok的官方SELFHOST指南：https://github.com/inconshreveable/ngrok/blob/master/docs/SELFHOSTING.md  
2) 国外一哥们的博客：http://www.svenbit.com/2014/09/run-ngrok-on-your-own-server/  
3) "海运的博客"中的一篇文章：http://www.haiyun.me/archives/1012.html

**二、实操步骤**

我的AWS EC2实例安装的是[Ubuntu](http://tonybai.com/tag/ubuntu) Server 14.04 x86\_64，并安装了[golang](http://tonybai.com/tag/golang) 1.4（go version go1.4 linux/amd64）。Golang是编译ngrokd和ngrok所必须的，建议直接从golang官方下载对应平台的二进制安装包（国内可以从 golangtc.com上下载，速度慢些罢了）。

**1、下载ngrok源码**

（GOPATH=~/goproj)  
$ mkdir ~/goproj/src/github.com/inconshreveable  
$ git clone https://github.com/inconshreveable/ngrok.git  
$ export GOPATH=~/goproj/src/github.com/inconshreveable/ngrok

**2、生成自签名证书**

使用ngrok.com官方服务时，我们使用的是官方的SSL证书。自建ngrokd服务，我们需要生成自己的证书，并提供携带该证书的ngrok客户端。

证书生成过程需要一个****NGROK\_BASE\_DOMAIN****。 以ngrok官方随机生成的地址693c358d.ngrok.com为例，其NGROK\_BASE\_DOMAIN就是"ngrok.com"，如果你要 提供服务的地址为"example.tunnel.tonybai.com"，那NGROK\_BASE\_DOMAIN就应该 是"tunnel.tonybai.com"。

我们这里以NGROK\_BASE\_DOMAIN="tunnel.tonybai.com"为例，生成证书的命令如下：

$ cd ~/goproj/src/github.com/inconshreveable/ngrok  
$ openssl genrsa -out rootCA.key 2048  
$ openssl req -x509 -new -nodes -key rootCA.key -subj "/CN=tunnel.tonybai.com" -days 5000 -out rootCA.pem  
$ openssl genrsa -out device.key 2048  
$ openssl req -new -key device.key -subj "/CN=tunnel.tonybai.com" -out device.csr  
$ openssl x509 -req -in device.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out device.crt -days 5000

执行完以上命令，在ngrok目录下就会新生成6个文件：

-rw-rw-r– 1 ubuntu ubuntu 1001 Mar 14 02:22 device.crt  
-rw-rw-r– 1 ubuntu ubuntu  903 Mar 14 02:22 device.csr  
-rw-rw-r– 1 ubuntu ubuntu 1679 Mar 14 02:22 device.key  
-rw-rw-r– 1 ubuntu ubuntu 1679 Mar 14 02:21 rootCA.key  
-rw-rw-r– 1 ubuntu ubuntu 1119 Mar 14 02:21 rootCA.pem  
-rw-rw-r– 1 ubuntu ubuntu   17 Mar 14 02:22 rootCA.srl

ngrok通过bindata将ngrok源码目录下的assets目录（资源文件）打包到可执行文件(ngrokd和ngrok)中 去，assets/client/tls和assets/server/tls下分别存放着用于ngrok和ngrokd的默认证书文件，我们需要将它们替换成我们自己生成的：(因此这一步务必放在编译可执行文件之前)

cp rootCA.pem assets/client/tls/ngrokroot.crt  
cp device.crt assets/server/tls/snakeoil.crt  
cp device.key assets/server/tls/snakeoil.key

**3、编译ngrokd和ngrok**

在ngrok目录下执行如下命令，编译ngrokd：

$ make release-server

不过在我的AWS上，出现如下错误：

GOOS="" GOARCH="" go get github.com/jteeuwen/go-bindata/go-bindata  
bin/go-bindata -nomemcopy -pkg=assets -tags=release \  
        -debug=false \  
        -o=src/ngrok/client/assets/assets\_release.go \  
        assets/client/…  
make: bin/go-bindata: Command not found  
make: \*\*\* [client-assets] Error 127

go-bindata被安装到了$GOBIN下了，go编译器找不到了。修正方法是将$GOBIN/go-bindata拷贝到当前ngrok/bin下。

$ cp /home/ubuntu/.bin/go14/bin/go-bindata ./bin

再次执行make release-server。

[~/goproj/src/github.com/inconshreveable/ngrok$](mailto:ubuntu@ip-172-31-4-131:~/goproj/src/github.com/inconshreveable/ngrok$) make release-server  
bin/go-bindata -nomemcopy -pkg=assets -tags=release \  
        -debug=false \  
        -o=src/ngrok/client/assets/assets\_release.go \  
        assets/client/…  
bin/go-bindata -nomemcopy -pkg=assets -tags=release \  
        -debug=false \  
        -o=src/ngrok/server/assets/assets\_release.go \  
        assets/server/…  
go get -tags 'release' -d -v ngrok/…  
code.google.com/p/log4go (download)  
go: missing Mercurial command. See <http://golang.org/s/gogetcmd>  
**package code.google.com/p/log4go: exec: "hg": executable file not found in $PATH**  
github.com/gorilla/websocket (download)  
github.com/inconshreveable/go-update (download)  
github.com/kardianos/osext (download)  
github.com/kr/binarydist (download)  
github.com/inconshreveable/go-vhost (download)  
github.com/inconshreveable/mousetrap (download)  
github.com/nsf/termbox-go (download)  
github.com/mattn/go-runewidth (download)  
github.com/rcrowley/go-metrics (download)  
Fetching <https://gopkg.in/yaml.v1?go-get=1>  
Parsing meta tags from <https://gopkg.in/yaml.v1?go-get=1> (status code 200)  
get "gopkg.in/yaml.v1": found meta tag main.metaImport{Prefix:"gopkg.in/yaml.v1", VCS:"git", RepoRoot:["https://gopkg.in/yaml.v1"](https://gopkg.in/yaml.v1)} at <https://gopkg.in/yaml.v1?go-get=1>  
gopkg.in/yaml.v1 (download)  
make: \*\*\* [deps] Error 1

又出错！提示找不到hg，原来是aws上没有安装hg。install hg后（sudo apt-get install mercurial），再编译。

$ make release-server  
bin/go-bindata -nomemcopy -pkg=assets -tags=release \  
        -debug=false \  
        -o=src/ngrok/client/assets/assets\_release.go \  
        assets/client/…  
bin/go-bindata -nomemcopy -pkg=assets -tags=release \  
        -debug=false \  
        -o=src/ngrok/server/assets/assets\_release.go \  
        assets/server/…  
go get -tags 'release' -d -v ngrok/…  
code.google.com/p/log4go (download)  
go install -tags 'release' ngrok/main/ngrokd

同样编译ngrok:

$ make release-client  
bin/go-bindata -nomemcopy -pkg=assets -tags=release \  
        -debug=false \  
        -o=src/ngrok/client/assets/assets\_release.go \  
        assets/client/…  
bin/go-bindata -nomemcopy -pkg=assets -tags=release \  
        -debug=false \  
        -o=src/ngrok/server/assets/assets\_release.go \  
        assets/server/…  
go get -tags 'release' -d -v ngrok/…  
go install -tags 'release' ngrok/main/ngrok

AWS上ngrokd和ngrok被安装到了$GOBIN下。

**三、调试**

1、**启动ngrokd**

$ ngrokd -domain="tunnel.tonybai.com" -httpAddr=":8080" -httpsAddr=":8081"  
[03/14/15 04:47:24] [INFO] [registry] [tun] No affinity cache specified  
[03/14/15 04:47:24] [INFO] [metrics] Reporting every 30 seconds  
[03/14/15 04:47:24] [INFO] Listening for public http connections on [::]:8080  
[03/14/15 04:47:24] [INFO] Listening for public https connections on [::]:8081  
[03/14/15 04:47:24] [INFO] Listening for control and proxy connections on [::]:4443  
… …

**2、公网连接ngrokd**

将生成的ngrok下载到自己的电脑上。

创建一个配置文件ngrok.cfg，内容如下：

server\_addr: "tunnel.tonybai.com:4443"  
trust\_host\_root\_certs: false

执行ngrok：  
$ ngrok -subdomain example -config=ngrok.cfg 80

Tunnel Status                 reconnecting  
Version                       1.7/  
Web Interface                 127.0.0.1:4040  
# Conn                        0  
Avg Conn Time                 0.00ms

连接失败。此刻我的电脑是在公网上。查看ngrokd的日志，没有发现连接到达Server端。试着在本地ping tunnel.tonybai.com这个地址，发现地址不通。难道是DNS设置的问题。之前我只是设置了"\*.tunnel.tonybai.com"的A地址，并未设置"tunnel.tonybai.com"。于是到DNS管理页面，添加了"tunnel.tonybai.com"的A记录。

待DNS记录刷新OK后，再次启动ngrok：

Tunnel Status online  
Version 1.7/1.7  
Forwarding http://epower.tunnel.tonybai.com:8080 -> 127.0.0.1:80  
Forwarding https://epower.tunnel.tonybai.com:8080 -> 127.0.0.1:80  
Web Interface 127.0.0.1:4040  
# Conn 0  
Avg Conn Time 0.00ms

这回连接成功了！

**3、内网连接ngrokd**

将ngrok拷贝到内网的一台PC上，这台PC设置了公司的代理。

按照同样的步骤启动ngrok：

$ ngrok -subdomain example -config=ngrok.cfg 80

Tunnel Status                 reconnecting  
Version                       1.7/  
Web Interface                 127.0.0.1:4040  
# Conn                        0  
Avg Conn Time                 0.00ms

不巧，怎么又失败了！从Server端来看，还是没有收到客户端的连接，显然是连接没有打通公司内网。从我自己的squid代理服务器来看，似乎只有443端口的请求被公司代理服务器允许通过，4443则无法出去。

1426301143.558 9294 10.10.126.101 TCP\_MISS/000 366772 CONNECT api.equinox.io:443 – DEFAULT\_PARENT/proxy.xxx.com -   通过了  
1426301144.441 27 10.10.126.101 TCP\_MISS/000 1185 CONNECT tunnel.tonybai.com:4443 – DEFAULT\_PARENT/proxy.xxx.com -  似乎没有通过

只能修改server监听端口了。将-tunnelAddr由4443改为443(注意AWS上需要修改防火墙的端口规则，这个是实时生效的，无需重启实例)：

$ sudo ngrokd -domain="tunnel.tonybai.com" -httpAddr=":8080" -httpsAddr=":8081" -tunnelAddr=":443"  
[03/14/15 04:47:24] [INFO] [registry] [tun] No affinity cache specified  
[03/14/15 04:47:24] [INFO] [metrics] Reporting every 30 seconds  
[03/14/15 04:47:24] [INFO] Listening for public http connections on [::]:8080  
[03/14/15 04:47:24] [INFO] Listening for public https connections on [::]:8081  
[03/14/15 04:47:24] [INFO] Listening for control and proxy connections on [::]:443  
… …

将ngrok.cfg中的地址改为443：

server\_addr: "tunnel.tonybai.com:443"

再次执行ngrok客户端：

Tunnel Status                 online  
Version                       1.7/1.7  
Forwarding                    http://epower.tunnel.tonybai.com:8080 -> 127.0.0.1:80  
Forwarding                    https://epower.tunnel.tonybai.com:8080 -> 127.0.0.1:80  
Web Interface                 127.0.0.1:4040  
# Conn                        0  
Avg Conn Time                 0.00ms

这回成功连上了。

**4、80端口**

是否大功告成了呢？我们看看ngrok的结果，总感觉哪里不对呢？噢，转发的地址怎么是8080端口呢？为何不是80？微信公众号/企业号可只是支持80端口啊！

我们还需要修改一下Server端的参数，将-httpAddr从8080改为80。

$ sudo ngrokd -domain="tunnel.tonybai.com" -httpAddr=":80" -httpsAddr=":8081" -tunnelAddr=":443"

这回再用ngrok连接一下：  
Tunnel Status                 online  
Version                       1.7/1.7  
Forwarding                    http://epower.tunnel.tonybai.com -> 127.0.0.1:80  
Forwarding                    https://epower.tunnel.tonybai.com -> 127.0.0.1:80  
Web Interface                 127.0.0.1:4040  
# Conn                        0  
Avg Conn Time                 0.00ms

这回与我们的需求匹配上了。

**5、测试**

在内网的PC上建立一个简单的http server 程序：hello

//hello.go  
package main

import "net/http"

func main() {  
    http.HandleFunc("/", hello)  
    http.ListenAndServe(":80", nil)  
}

func hello(w http.ResponseWriter, r \*http.Request) {  
    w.Write([]byte("hello!"))  
}

$ go build -o hello hello.go  
$ sudo ./hello

通过公网浏览器访问一下“http://epower.tunnel.tonybai.com”这个地址，如果你看到浏览器返回"hello!"字样，那么你的ngrokd服务就搭建成功了！

**四、注意事项**

客户端ngrok.cfg中server\_addr后的值必须严格与-domain以及证书中的NGROK\_BASE\_DOMAIN相同，否则Server端就会出现如下错误日志：

[03/13/15 09:55:46] [INFO] [tun:15dd7522] New connection from 54.149.100.42:38252  
[03/13/15 09:55:46] [DEBG] [tun:15dd7522] Waiting to read message  
[03/13/15 09:55:46] [WARN] [tun:15dd7522] Failed to read message: remote error: **bad certificate**  
[03/13/15 09:55:46] [DEBG] [tun:15dd7522] Closing

文章地址：https://tonybai.com/2015/03/14/selfhost-ngrok-service/