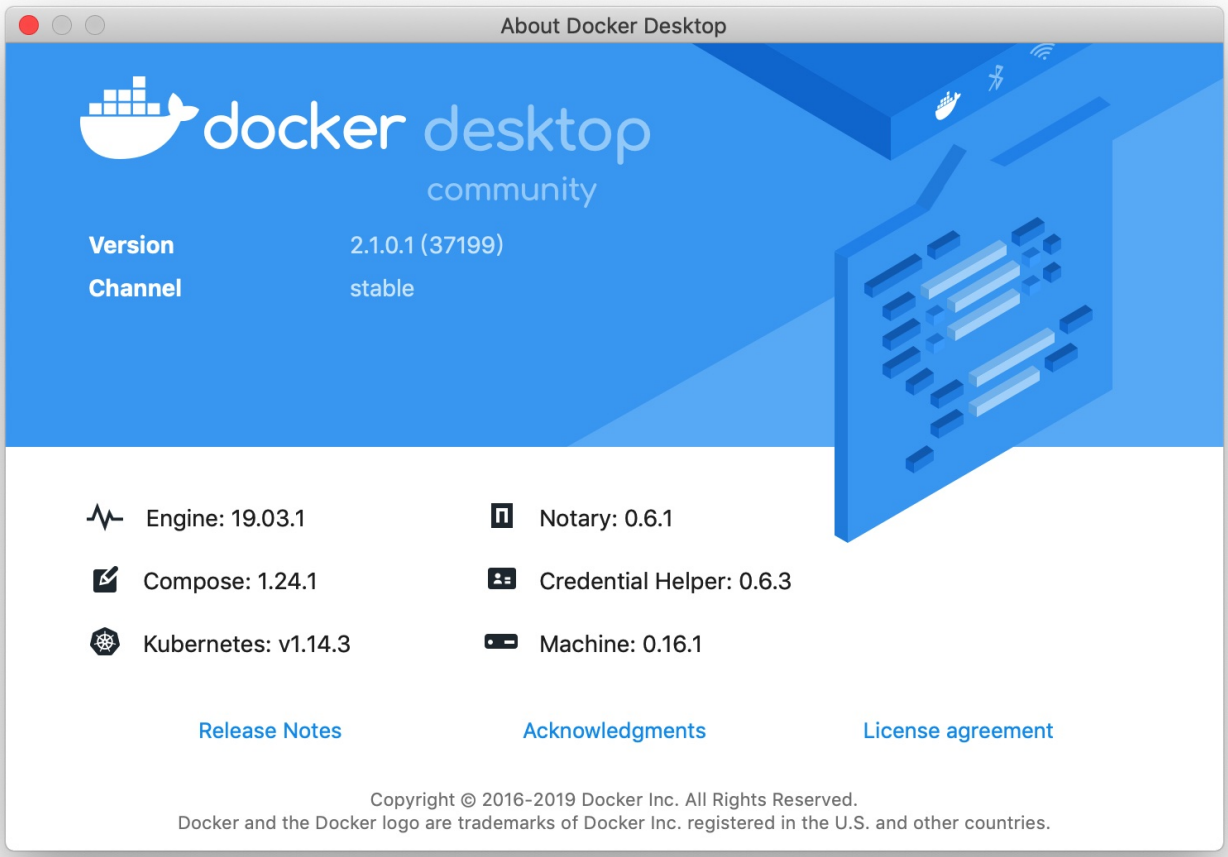
Docker for Mac 的网络问题及解决办法

2019-08-30 2019-11-24 2 Comments

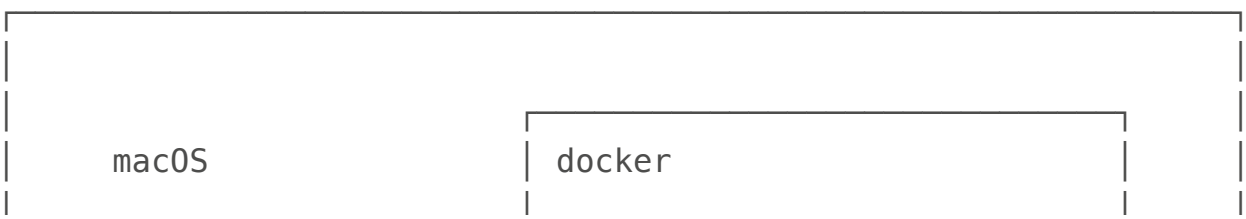


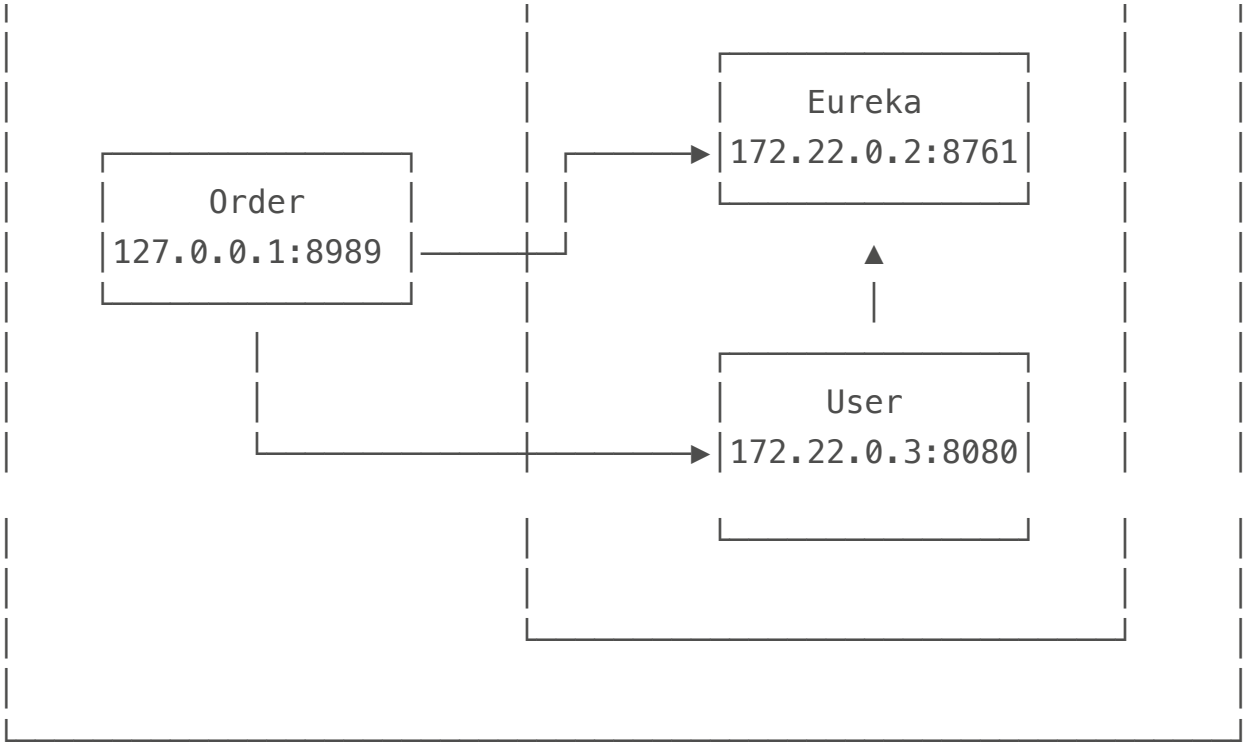
用 Docker for Mac 已经很久了，用它跑本地开发环境可以说是非常方便。

但是 Docker for Mac 自诞生以来就一直有一个问题，那就是在宿主机上看不到 `docker0`，无法访问容器所在的网络，也就是说不能 `ping` 通 Docker 给 Container 所分配的 IP 地址。关于这个问题，官方文档有描述：*Known limitations, use cases, and workarounds*

对于 `docker run` 启动的 Container 来说，通常会通过 `-p` 参数映射相应的服务端口，一般不会遇到要直接访问容器 IP 的情况。

但是当我们在 docker 中运行多个微服务并想进行本地调试的时候，在 Mac 上却没法实现。（`--net=host` **仅在 Linux 系统上有效**，我在这个上边被坑了两三个小时 [捂脸]）



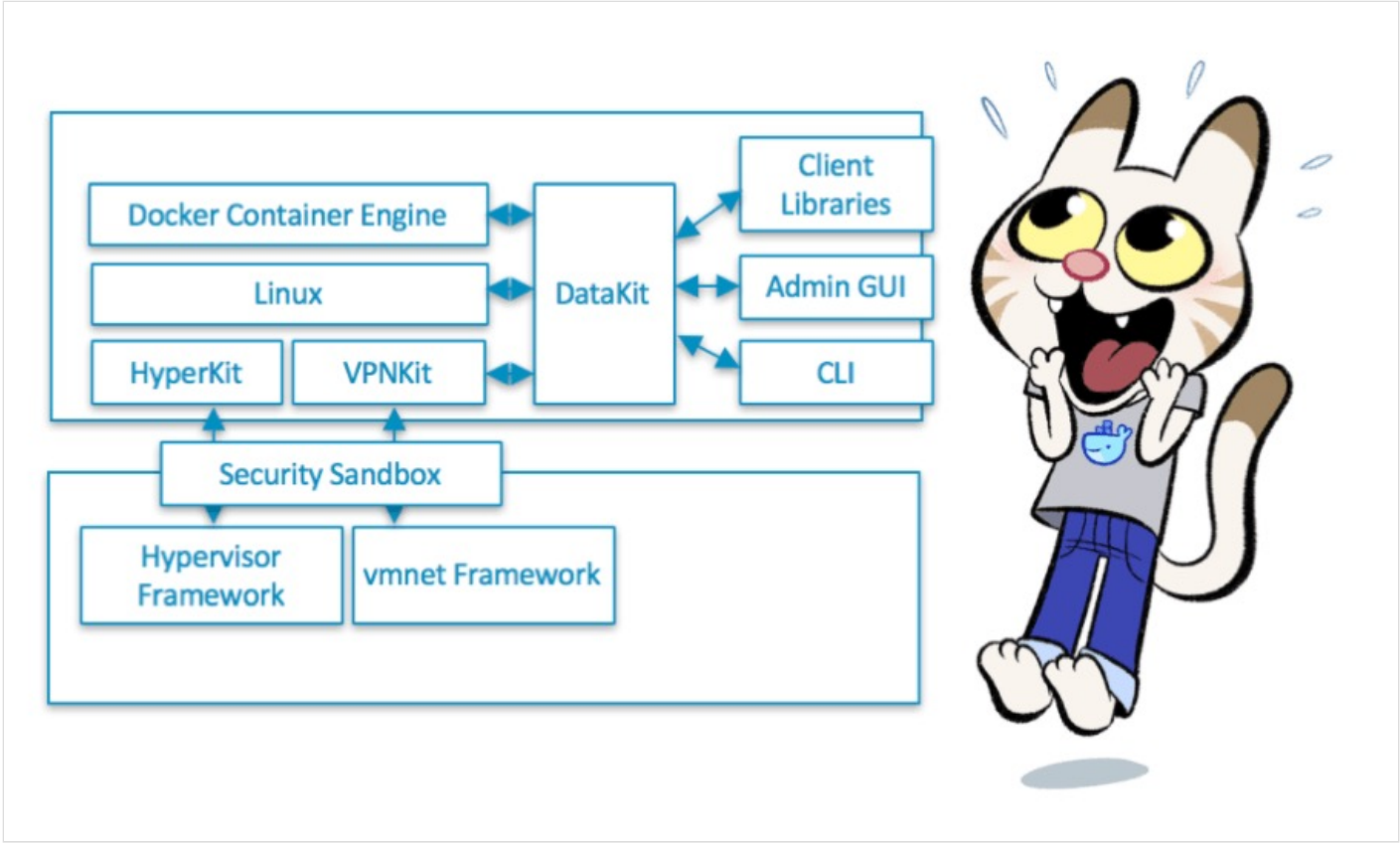


例如上图中，Eureka Server 和 User 服务均存在于容器中，本地调试 Order 服务。Order 需要调用 User，但是我们在本机上是访问不到 172.22.0.3:8080 的。

Docker for Mac 基本原理

要解决这个问题，得先搞清楚 Docker for Mac 的原理。

我们都知道 Docker 是利用 Linux 的 Namespace 和 Cgroups 来实现资源的隔离和限制，容器共享宿主机内核，所以 Mac 本身没法运行 Docker 容器。不过不支持不要紧，我们可以跑虚拟机，最早还没有 Docker for Mac 的时候，就是通过 `docker-machine` 在 Virtual Box 或者 VMWare 直接起一个 Linux 的虚拟机，然后在主机上用 Docker Client 操作虚拟机里的 Docker Server。



Docker for Mac Architecture

Docker for Mac 也是在本地跑了一个虚拟机来运行 Docker，不过 Hypervisor 采用的是 xhyve，而 xhyve 又基于 Mac 自带的虚拟化方案 Hypervisor.framework，虚拟机里运行的发行版是 Docker 自己打包的 LinuxKit，之前用的发行版好像是 Alpine Linux。

总而言之就是 Docker for Mac 跑的这个虚拟机非常轻量级，性能也会更好。

解决办法

网络上对于这个问题的解法多种多样，这里只说三个常见并靠谱的：

1. 放弃 Docker for Mac 换 Docker Toolbox。

   实际上就是用 Virtual Box 换掉 xhyve，然后设置容器实例和虚拟机处于同一网段。

   这有点像“刮骨疗伤”，**不推荐**，毕竟 Docker for Mac 就是为了替代 Docker Toolbox 的。

2. 在 Docker for Mac 的虚拟机里跑一个 OpenVPN Server，然后从本地连过去。流程如下：

   ```
   1  Mac <-> Tunnelblick <-> socat/service <-> OpenVPN Server <-> Containers
   ```

   因为 Docker for Mac 在重启的时候不会保留虚拟机里的改动，所以这个 OpenVPN Server 必须要跑在容器里，并且网络模式需要设置为 `host`，这样才可以访问到所有的 Docker 网络。

   使用起来略麻烦，虽说已经有写好了的 OpenVPN 的配置与脚本，但是要装软件、启动容器、修改配置等，所以还是**不推荐**。

3. 使用 Docker for Mac 里的一个实验性功能：SOCKS 代理。

   强烈推荐，傻瓜操作，在 docker 没有启动的时候也能“自适应”—— 不用频繁的来回改代理的配置。

下面就详细讲一下第 3 种。

首先需要安装一下 `jq`

```
$ brew install jq
```

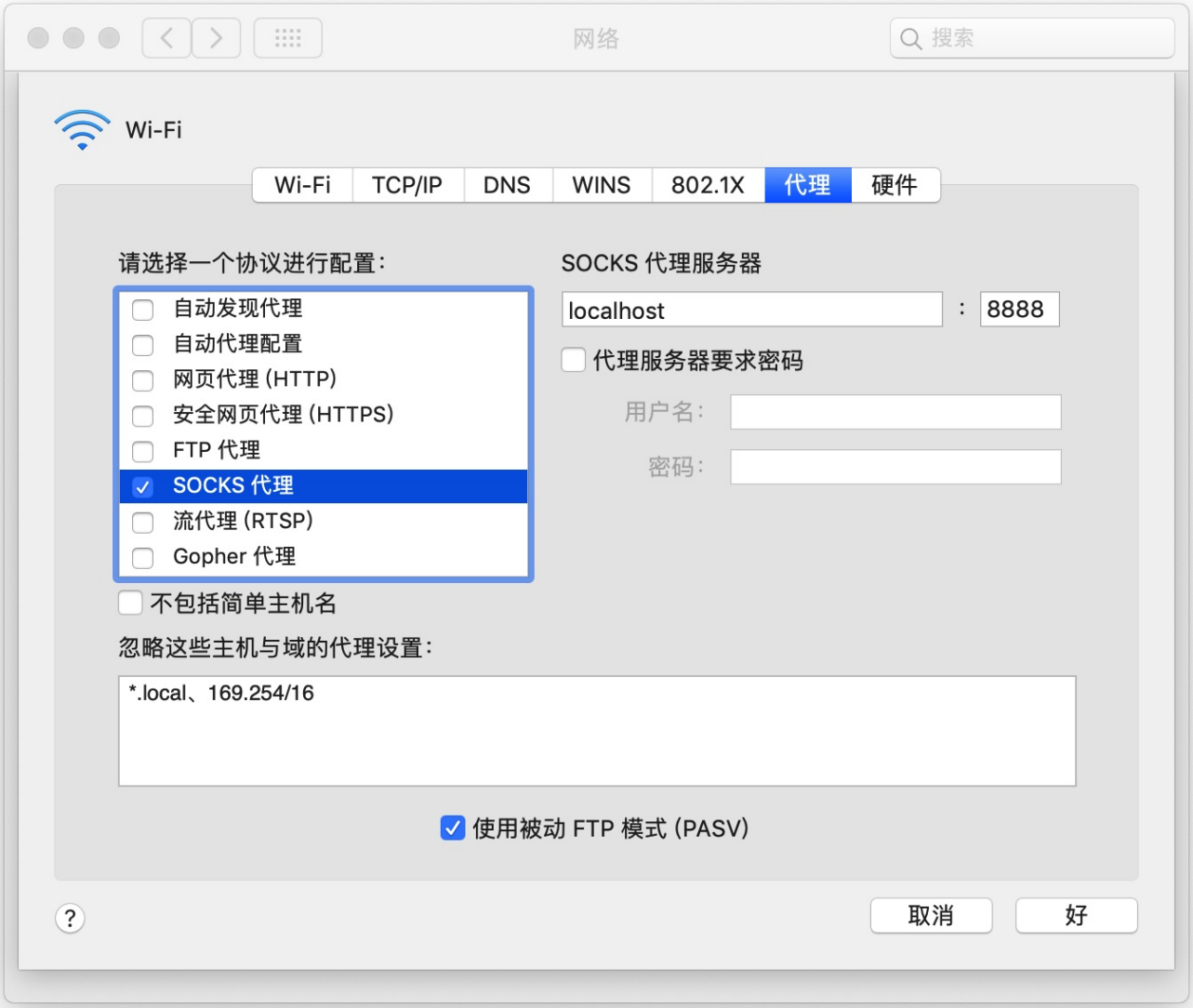
然后执行

```
$ cd ~/Library/Group\ Containers/group.com.docker/
$ mv settings.json settings.json.backup
$ cat settings.json.backup | jq '.["socksProxyPort"]=8888' > settings.json
```

重启 Docker for Mac

前往 系统偏好设置 -> 网络 -> 高级 -> 代理

设置 SOCKS 代理：localhost:8888 并保存、应用



跑个 nginx 看下效果

```
$ # 启动 nginx
$ docker run -d --name nginx nginx

$ # 找到容器 ip
$ docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
172.17.0.2

$ # 访问
$ http_proxy=socks5://localhost:8888 curl http://172.17.0.2
<!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>
```

好了，至此就完事大吉了，可以关机睡觉了。

先关 Docker ……

等等，我怎么上不了网了?

上边配置 SOCKS 的方法需要每次在 Docker for Mac 开启 / 关闭 的时候手动去网络里配置一下。这就和我们

“人至懒则无敌”的信仰有点背道而驰了。那么有没有解决办法呢?

有，并且不止一种。

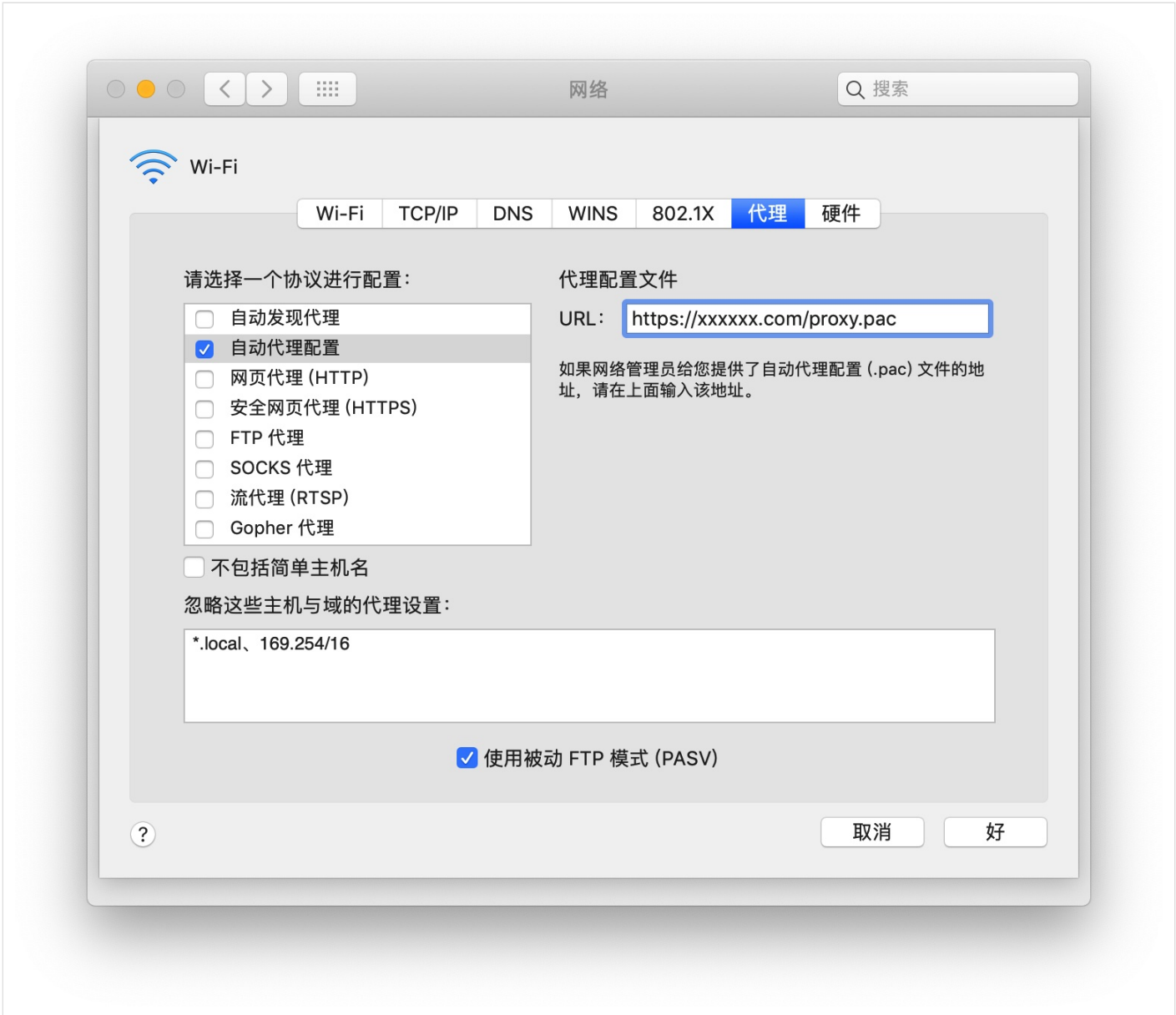
这里介绍个大家之前常用的 PAC 文件的方法吧。

下面是我 `proxy.pac` 文件的一部分，大家可以参考（都是老司机了，不用说明了吧～）

```
function FindProxyForURL(url, host) {
  if (isInNet(host, '172.17.0.0', '255.255.0.0')) {
    return 'SOCKS5 127.0.0.1:8888'
  }
  if (isInNet(host, '172.22.0.0', '255.255.0.0')) {
    return 'SOCKS5 127.0.0.1:8888'
  }
  return 'DIRECT'
}
```

因为实测现在新的 macOS 系统不支持本地的 PAC 文件，所以你还得找个空间把这个文件扔到“云上”。

有了 PAC 文件的 http 地址，我们还是配置需要一次网络，不过这次就不是“SOCKS 代理”而是“自动代理配置”



至此已经趋近完美了。至于这个实验性的功能嘛，目前看来还是比较稳定的，不过一直没被加到 Docker for Mac 的界面设置里，希望官方能早日添加以解放我们吧。