链滴

solo 博客搭建过程

作者：bingoct

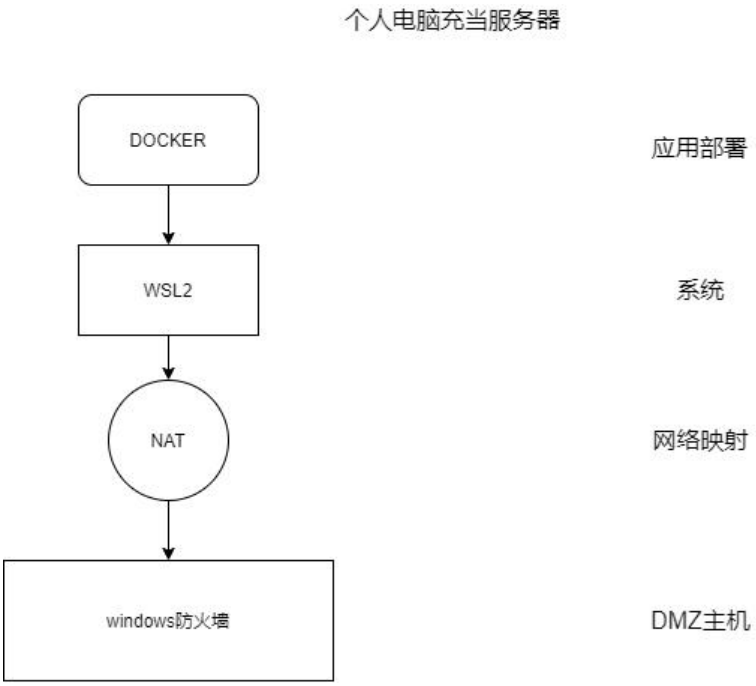
原文链接：https://ld246.com/article/1634194439894

来源网站：链滴

环境与框架

框架

PC充当服务器，应用部署在wsl2上，与windows系统隔离，windows系统充当了“堡垒机”的作用。
br />



系统与软件版本

windows：　win10 专业版 20H2

Docker for windows with wsl2 backend：　Client version 20.10.8，Server version 20.10.8

wsl2：　Ubuntu 20.04.3 LTS

mysql:　Ver 8.0.26-0ubuntu0.20.04.3 for Linux on x86_64 ((Ubuntu))

docker images：

name	image id
b3log/lute-http	da614c34e2d8
b3log/solo	4afbc7964f83
b3log/siyuan	ac88f341d886
nginx	87a94228f133

应用部署

Docker配置

安装docker软件"[3]""[4]"

docker三个默认虚拟网桥：bridge，host，none。为了方便容器通信，自定义虚拟网桥hinet。

```
$ docker network create --driver  bridge hinet
$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
843e947651e9   bridge    bridge    local
feab24d2504c   hinet     bridge    local
08af6d689554   host      host      local
74030816c0ea   none      null      local
```

如果你想为之后的容器设置静态 ip，参考"[2]"。

solo部署

先安装MYSQL，按照"[1]"创建用户，设置密码，授予权限，启动docker容器。MYSQL本地安装，方数据的保存与维护。

容器启动参数：

```
$sudo docker run --detach --name solo -p 8080:8080 --network hinet\
    --env RUNTIME_DB="MYSQL" \
    --env JDBC_USERNAME="你的数据库用户名" \
    --env JDBC_PASSWORD="你的数据库用户名对应的密码" \
    --env JDBC_DRIVER="com.mysql.cj.jdbc.Driver" \
    --env JDBC_URL="jdbc:mysql://你的宿主机ip:你的MYSQL端口/solo?useUnicode=yes&characerEncoding=UTF-8&useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true" \
    b3log/solo --listen_port=8080 --server_scheme=http --server_host=你的域名 --server_port\
    --static_server_scheme=https --static_server_host=cdn.jsdelivr.net\
    --static_server_port= --static_path=/gh/88250/solo/src/main/resources\
    --lute_http=http://lute:8249
```

踩坑点

将network设置为#host模式#数据库无法用localhost访问，问题记录SOLO DOCKER 容器启动失败。

因为#docker的C/S架构#，当容器访问宿主机时，实际上走WSL-hpyer-v网桥，所以宿主机接收到的i是WSL网桥的IP。将MYSQL用户的host改为了windows的"wsl网桥ip地址"，或者设置为"%"匹配所有p，记得注释掉mysql配置文件/etc/mysql/mysql.conf.d/mysqld.cnf中的bind-address。

```
# bind-address          = 127.0.0.1
```

lute&Gitalk

- lute-better markdown ，如法炮制，启动容器参数添加--network hinet
- Solo integrated Gitalk comment system

思源部署

启动容器[6]。

```
$sudo docker run -d --name "siyuan" -v 本地思源工作空间:/siyuan/workspace -p 6806:6806 --
etwork=hinet\
        -e LANG=zh_CN.UTF-8 -e LC_ALL=zh_CN.UTF-8  b3log/siyuan\
         --workspace=/siyuan/workspace --servePath="localhost:6806"
```

思源版本迭代快，某些情况下不成功可能并不是配置的问题，而是个bug。

当时部署1.3.9版本，某些移动端 Pad 设备上无法进入卡了一整天，结果更新1.4.0解决了，所以还是多关注社区。

Nginx部署

为了自定义添加模块，nginx最好编译安装。但是我又偷懒了，直接docker走起。启动容器，并挂载器。

```
sudo docker run -d --name nginx1.21.3 -p 443:443 -p 80:80  --network hinet\
   -v 本地配置文件:/etc/nginx/nginx.conf\
   -v 本地配置文件夹:/etc/nginx/conf.d\
   -v 本地日志文件夹:/var/log/nginx \
   -v 本地ssl证书文件夹:/opt/ssl\
   nginx
```

我个人习惯将不同的应用代理配置放在conf.d文件夹中作为单独的配置

SNI分流，443端口复用

进入容器确保docker-nginx开启了TLS SNI support enabled有--with-stream_ssl_module和 --with stream_ssl_preread_module模块：

```
$ docker ps
CONTAINER ID   IMAGE          COMMAND
NGINX的ID   nginx      "/docker-entrypoint...."
$ docker exec -it NGINX的ID /bin/bash
# nginx -V 2>&1 | grep -- 'stream_ssl_module'
# nginx -V 2>&1 | grep -- 'stream_ssl_module'
```

SNI分流有两种模式"[5]"，一种是TLS pass through即分流到监听端口的server再解析TLS，另一种是erminating TLS, forward TCP，直接在443端口server上直接解析TLS。这里选择前者来配置。

```
stream {
   # SNI 识别，将域名映射成一个配置名
   map $ssl_preread_server_name $backend_name {
      你的博客域名 blog;
      思源域名 siyuan;
   # 可以接着添加其他的配置
   # 域名都不匹配情况下的默认值
      default web;     #将流量转发到web服务上
   }
   # 这里的127.0.0.1并不是应用容器对应的ip，之后会用proxy_pass替换成容器ip
   upstream blog{
         server 127.0.0.1:8080;
   }
   upstream siyuan{
         server 127.0.0.1:6806;
```

```
    }

    # 监听 443端口 并开启 ssl_preread，
    server {
        listen 443 reuseport;
        listen [::]:443 reuseport;
        proxy_pass  $backend_name;
        ssl_preread on;
        proxy_protocol on;
    }
}
http {
    # 其他一些配置，比如gzip、日志等，也可以写在conf.d文件夹对应的配置文件中。
    include /etc/nginx/conf.d/*.conf
}
```

solo&siyuan反向代理

注意#X-real-ip#模块的写法，请按实际情况更改。

/etc/nginx/conf.d/blog.conf

```
upstream solo_end{
    server solo:8080;
}

server{
    listen 127.0.0.1:8080 ssl http2 proxy_protocol;
    server_name 你博客的域名;
    ssl_certificate 证书公钥路径;
    ssl_certificate_key 证书密钥路径;
    # 其他一些SSL、日志配置
    location / {
        proxy_pass http://solo_end$request_uri;
        proxy_set_header  Host $http_host;
        proxy_set_header X-Real-IP       $http_x_forwarded_for;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        client_max_body_size  10m;
}
```

/etc/nginx/conf.d/siyuan.conf 如法炮制

```
upstream sy_end{
    server siyuan:6806;
}

server{
    listen 127.0.0.1:6806 ssl http2 proxy_protocol;
    server_name 思源的域名;
    ssl_certificate 证书公钥路径;
    ssl_certificate_key 证书密钥路径;
    ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;
    # 其他一些SSL、日志配置
    location / {
```

```
        proxy_pass http://sy$request_uri;
        proxy_set_header X-Real-IP        $http_x_forwarded_for;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        client_max_body_size  10m;
}
```

强制https

方法多种[7]

- rewrite
- 497状态码
- meta刷新
- proxy_rediret

示例rewrite的简单写法：

```
server
{
    listen 80;
    server_name 你的域名;
    rewrite ^(.*)$ https://$host$1 permanent;
}
```

如果"使用CloudFlare CDN"，可以不用在nginx内配置强制https

windows设置

powershell用scoop安装sudo插件"[8]"，或者把sudo删掉直接管理员权限下执行下面的指令。

WSL2

WSL(windows subsystem for linux），可以从"[13]"了解关于WSL和WSL2的区别以及WSL的特性

- 安装wsl2 "[10]"，如果你想安装多个相同的发行版，参考"[9]"
- 导出发行版镜像文件，迁移安装目录，减小C盘空间大小 "[11]"
- 配置，调整内存和CPU占用上限 "[12]"

其实当我配置完后，我才发现当初选择wsl2是个糟糕的选择。因为

1. .wslconfig是全局的设置，无法对单个wsl2发行版配置。但是不设置.wslconfig，你就会见证wsl2如何吃掉你的内存。

Options for .wslconfig

Section label: [wsl2]

These settings affect the VM that powers any WSL 2 distribution.

2. "前文"说的docker network host问题

3. /mnt挂载，如果有关文件是挂载在windows上的，那么访问速率会下降
4. wsl2的ip不固定，需要额外 "设置"。

但是wsl2爽在vscode、windows terminal。毕竟用着windows的界面来操作linux多么舒服。

NAT

涉及powershell的netsh interface portproxy命令。

- 为WSL网桥添加固定子网段，并分配给wsl2固定ip。方便控制WSL2

```
> sudo netsh interface ip add address " vEthernet (WSL)" "192.168.50.1 255.255.255.0"
> wsl -d WSL发行版名字 -u root ip addr add 192.168.50.2/24 broadcast 192.168.50.255 dev et
0 label eth0:1
```

wsl2的ip为192.168.50.2，windows的ip为192.168.50.1

- 将wsl2的443端口和80端口按映射到windows的ipv6地址上

```
> sudo netsh interface netsh interface portproxy add v6tov4 listenport=443 connectaddress=
92.168.50.2 listenaddress=*
> sudo netsh interface netsh interface portproxy add v6tov4 listenport=80 connectaddress=1
2.168.50.2 listenaddress=*
> netsh interface portproxy show all
侦听 ipv6:                连接到 ipv4:

地址          端口        地址          端口
--------------- ----------  --------------- ----------
*             80          192.168.50.2    80
*             443         192.168.50.2    443
```

这里图方便直接绑定了所有的ipv6，如果想严格控制

补充：docker wsl2backend在未声明-p参数的宿主机ip时候绑定的ip是0.0.0.0（即使我们在wsl2部的容器，依旧可以在windows直接用localhost访问）。理论上不用做v6tov4的端口映射也能实现。是为了保险起见还是直接做NAT了。如果为了你想

windows防火墙设置

放行80和443端口，建议用直接用图形界面操作，如果是windows server的话，了解下netsh advfirewall firewall命令。

```
> sudo netsh advfirewall firewall add rule name= "web-server" dir=in action=allow protocol=
CP localport=80
> sudo netsh advfirewall firewall add rule name= "web-server" dir=in action=allow protocol=
CP localport=443
> netsh advfirewall firewall show rule "web-server"
规则名称:                             web-server
----------------------------------------------------------------------
已启用:                               是
方向:                                 出
配置文件:                             域,专用,公用
分组:
本地 IP:                              任何
```

远程 IP: 任何
协议: TCP
本地端口: 任何
远程端口: 80,443
边缘遍历: 否
操作: 允许
确定。

CDN

因为是ipv6的地址，以及目前家用路由器的ipv6防火墙功能不完善，我需要解决两个问题

- 隐藏真实ip，安全性更高
- ipv6转ipv4协议栈，方便他人访问

这自然就涉及到了CDN，当然CDN还有负载均衡、加速客户访问速度这些作用。CloudFlare免费提供DN服务，只需要把域名解析服务器设置为CloudFlare就行。

在CloudFlare中添加DNS记录，记得勾选代理，这样才能启动cloudflare的cdn服务。为了方便之后的ddns"，将二级域名blog和siyuan添加为CNAME记录。

顺便再嫖一下CloudFlare提供的免费通配域名TLS证书（不用为其他的二级域名反复声请）。CloudFlare证书分为三种：边缘证书、客户端证书、源服务器证书。加密模式为灵活、完全、完全（严格）。

● 边缘证书：客户端（浏览器）到CDN服务器加密通信用到的证书。在将域名迁移到CloudFlare解析时候自动签发，ECDSA SHA245加密，基本上多数的客户端（浏览器）支持，时效默认为域名时效可以免费续签。

● 源服务器证书：CDN拉取源服务器内容加密通信用到的证书。需要手动申请，默认是RSA2048加，一些客户端（浏览器）可能会提示不安全，有效期最高15年。

下载申请好的源服务器证书，"部署到wsl2的nginx上"。

选择完全（严格）模式，在用户访问网站（实际为CloudFlare的CDN服务器）的时候，用到的证书是缘证书；CloudFlare访问WSL2服务器用到的是源服务器证书。

如果你想避免用户绕过CDN直接访问服务器，还可以在防火墙里设置80/443入站的ip只能为CloudFlae的CDN服务器地址。

最后再白嫖一波CloudFlare的强制https功能，省去"配nignx"的烦恼。

DDNS

分配的ipv6会随着变动，需要向CloudFlare定时更新IP记录。创建CloudFlare的API令牌，参考Cloudflare提供的api文档，根据"[17]"用powershell7的restful api仿照别人的脚本写一个吧：

```
[cmdletbinding()]
$Email = "注册CLOUDFLARE的用户时用的邮箱"
```

```
$Token = "创建的令牌"
$Domain = "你的域名"
$type = "DNS解析记录类型"
$Record = "解析记录的域名"

# Build the request headers once. These headers will be used throughout the script.
$headers = @{
    "X-Auth-Email"  = $($Email)
    "Authorization" = "Bearer $($Token)"
    "Content-Type"  = "application/json"
}


$date = Get-Date
Write-Output "============================================="
Write-Output "$($date)"


#Region Token Test
## This block verifies that your API key is valid.
## If not, the script will terminate.

$uri = "https://api.cloudflare.com/client/v4/user/tokens/verify"

$auth_result = Invoke-RestMethod -Method GET -Uri $uri -Headers $headers -SkipHttpError
heck
if (-not($auth_result.result)) {
    Write-Output "API token validation failed. Error: $($auth_result.errors.message). Terminatin
 script."
    # Exit script
    return
}
Write-Output "API token validation [$($Token)] success. $($auth_result.messages.message)."
#EndRegion

#Region Get Zone ID
## Retrieves the domain's zone identifier based on the zone name. If the identifier is not fou
d, the script will terminate.
$uri = "https://api.cloudflare.com/client/v4/zones?name=$($Domain)"
$DnsZone = Invoke-RestMethod -Method GET -Uri $uri -Headers $headers -SkipHttpErrorCh
ck
if (-not($DnsZone.result)) {
    Write-Output "Search for the DNS domain [$($Domain)] return zero results. Terminating scr
pt."
    # Exit script
    return
}
## Store the DNS zone ID
$zone_id = $DnsZone.result.id
Write-Output "Domain zone [$($Domain)]: ID=$($zone_id)"
#End Region

#Region Get DNS Record
## Retrieve the existing DNS record details from Cloudflare.
```

```
$body = @{
    "type"  = $($type)
    $Record = $($Record)
}
$uri = "https://api.cloudflare.com/client/v4/zones/$($zone_id)/dns_records"
$DnsRecord = Invoke-RestMethod -Method GET -Uri $uri -Headers $headers -Body $body -S
ipHttpErrorCheck
if (-not($DnsRecord.result)) {
    Write-Output "Search for the DNS record [$($Record)] return zero results. Terminating scrip
."
    # Exit script
    return
}
## Store the existing IP address in the DNS record
$old_ip = $DnsRecord.result.content
## Store the DNS record type value
$record_type = $DnsRecord.result.type
## Store the DNS record id value
$record_id = $DnsRecord.result.id
## Store the DNS record ttl value
$record_ttl = $DnsRecord.result.ttl
## Store the DNS record proxied value
$record_proxied = $DnsRecord.result.proxied
Write-Output "DNS record [$($Record)]: Type=$($record_type), IP=$($old_ip)"
#EndRegion

#Region Get Current Public IP Address
$new_ip = (ipconfig | select-string "IPv6" | out-string -Stream)[1].Split(" : ")[1]
Write-Output "Public IP Address: OLD=$($old_ip), NEW=$($new_ip)"
#EndRegion

#Region update Dynamic DNS Record
## Compare current IP address with the DNS record
## If the current IP address does not match the DNS record IP address, update the DNS recor
.
if ($new_ip -ne $old_ip) {
    Write-Output "The current IP address does not match the DNS record IP address. Attempt
o update."
    ## Update the DNS record with the new IP address
    $uri = "https://api.cloudflare.com/client/v4/zones/$($zone_id)/dns_records/$($record_id)"
    $body = @{
        type    = $record_type
        name    = $Record
        content = $new_ip
        ttl     = $record_ttl
        proxied = $record_proxied
    } | ConvertTo-Json

    $Update = Invoke-RestMethod -Method PUT -Uri $uri -Headers $headers -SkipHttpErrorC
eck -Body $body
    if (($Update.errors)) {
        Write-Output "DNS record update failed. Error: $($Update[0].errors.message)"
        ## Exit script
        return
```

```
    }

    Write-Output "DNS record update successful."
    return ($Update.result)
}
else {
    Write-Output "The current IP address and DNS record IP address are the same. There's no
eed to update."
}
#EndRegion
```

注意第77行$new_ip = (ipconfig | select-string "IPv6" | out-string -Stream)[1].Split(" : ")[1]，这默认选择第一个ipv6地址，如果你想替换成其他的ip地址，记得更改。

开机自启

将上面的代码分别整理成ps脚本或者vbs脚本，添加到计算机管理的任务计划程序中。可能会面临执权限的问题。网上的教程很多，自行搜索。

Q&A

Q：为什么 network 不直接用host模式

A：docker for wsl2 backend的host模式和直接在wsl2上安装docker的#host模式#不一样，是无法 localhost或是127.0.0.1访问宿主机。使用wsl -l可以发现docker安装了对应的wsl发行版和数据：docker-desktop和docker-desttop-data。"因此"我在之后的容器与宿主机(wsl2)之间的通信方式都为直访问"宿主机的ip"。

Docker Desktop for windows方式，其实质是利用#docker的C/S架构#，将windows模式下的docker对应docker.sock，docker客户端二进制和docker的数据目录挂载到WSL2里面的linux机器，在此linux机器下执行docker命令(**docker命令为docker客户端**)，实质为客户端通过 挂载的/var/run/docer.sock文件与windows里面的dockerd服务端进程通信。

win10利用WSL2安装docker的2种方式 - JustDoIT的文章 - 知乎

```
> wsl -l
适用于 Linux 的 Windows 子系统分发版:
Ubuntu-20.04 (默认)
docker-desktop-data
docker-desktop
```

Q：为什么nginx的X-Real-IP的写法和"[1]"中的不一样

A：因为我后面做了wsl2-windows的NAT和cdn代理，$remote_addr永远是127.0.01。如果你不放那种方法是对的，可以日志里输出查看一下。关于#X-real-ip#和proxy_protocol的详见"[14]"和"[15]"理论上做了CDN的配置应该如"[16]"配置，我用日志看到$http_x_forwarded_for也是客户端实际ip就偷懒了。

Q：为什么做了HTTPS，却没有像[1]和[6]中启动容器的时使用https参数?

A：因为是在nginx的容器里先解析了tls，也就是说在反向代理到实际的应用端口的时候已经没有tls
。

参考资料

[1] b3log/solo https://github.com/88250/solo

[2] docker-network https://docs.docker.com/network/bridge/

[3] WSL 2 上的 Docker 远程容器入门 https://docs.microsoft.com/zh-cn/windows/wsl/tutorials/sl-containers

[4] Docker wsl2 backend https://docs.docker.com/desktop/windows/wsl/

[5] nginx SNI分流 https://gist.github.com/kekru/c09dbab5e78bf76402966b13fa72b9d2

[6] b3log/siyuan https://ld246.com/article/1619868273581

[7] Nginx强制跳转https https://www.jianshu.com/p/116fc2d08165

[8] powershell-sudo插件 https://dev.to/amehdaly/sudo-for-windows-4dcb

[9] LxRunOffline https://github.com/DDoSolitary/LxRunOffline

[10] 安装wsl https://docs.microsoft.com/zh-cn/windows/wsl/install

[11] wsl2自定义发行版 https://docs.microsoft.com/en-us/windows/wsl/use-custom-distro

[12] 设置.wslconfig https://docs.microsoft.com/en-us/windows/wsl/wsl-config#configure-global-options-with-wslconfig

[13] 比较wsl1和wsl2 https://docs.microsoft.com/zh-cn/windows/wsl/compare-versions

[14] nginx-using-proxy-protocol文档 https://docs.nginx.com/nginx/admin-guide/load-balancer/using-proxy-protocol/

[15] proxyprotocol文档 http://www.haproxy.org/download/1.8/doc/proxy-protocol.txt

[16] nginx-cloudflare客户端原始ip https://support.cloudflare.com/hc/en-us/articles/20017078-How-do-I-restore-original-visitor-IP-with-Nginx-

[17] cloudflare使用powershell脚本动态DDNS https://adamtheautomator.com/cloudflare-dynamic-dns/