链滴

利用 docker 部署 solo 并升级为 https

作者：CodingOX

原文链接：https://ld246.com/article/1558620386375

来源网站：链滴

前言

这篇文章的内容在大概2周前就已经完成了，但是因为个人懒惰，所以一直没有进行总结和更新记录大大的罪过啊，今天怎么也要开始重新梳理笔记了，接下来进入正题吧！

之前的博文：https://www.hicode.club/articles/2019/04/26/1556265891647.html 记录了安装博系统solo的过程，但是美中不足的是当时使用的是http，没能用上https，后面在学习和实践过程中将个人博客地址：www.hicode.club 改成了https，这里记录下其中的关键点。

本篇文章会包含前作的内容，所以前作你可以不看，将这篇文章作为solo的https的参考文章，仅供考！

最近好好学习了Docker，所以想把线上环境的所有组件替换为Docker部署，对于Solo这个个人博客统，至少需要部署3个Docker容器。

1. MySQL
2. Solo
3. Nginx

其顺序也是从上倒下依次部署最为妥当，Docker部署简单，对初学者而言难其中一个难点就是**网络信**，我这里就把各个过程做一个记录。

准备工作

因为部署了3个docker容器，那么他们要互相访问，就不能（更准确来讲"不应该"）通过传统的localhst或者127.0.0.1了。

官方也不再建议使用 --link了，那么使用什么了？

官方推荐的是建立一个docker的网络，代码如下所示，其中hicode是我定义的网络名，大家可以改自己喜欢的：

```
docker network create -d bridge hicode
```

后续每一个容器在建立的时候，都应该加入这个网络hicode，然后容器和容器之间就可以通过容器名行访问了。

MySQL

讲真，这个东西是最浪费时间。在Docker化之前，我服务器上用的是MySQL8，当时就折腾了不少间，虽然最终也部署起来了，但是后面考虑了下还是放弃了8这个版本，原因主要是如下几点：

- MySql8的资料相对较少，记录的问题解决方案也比较少，要爬坑很久才能找到解决方案。
- MySQL8的内存占用和CPU占用有点高，特别是内存占用持续高涨。我的云服务器是捋的腾讯羊毛当时本来是1核2G的配置，我为了延长使用时间，把服务器配置降级成了1核1G了，所以对于MySql8内存占用，我的小服务器有点扛不住。
- 在可以预期的未来，我们公司包括整个行业还是会在 5.X居多，所以8.0就留在后面有机会折腾了。

部署MySQL 5.X版本的指令如下所示

```
docker run --name mysql5.7 -p 3306:3306  \
--network hicode --privileged=true --restart=always \
-v /lcf/docker-data/mysql5.7/data:/var/lib/mysql \
-v /lcf/docker-data/mysql5.7/conf:/etc/mysql/conf.d \
-e TZ=Asia/Shanghai \
-e MYSQL_ROOT_PASSWORD=yourpwd -d mysql:5.7.26 \
--character-set-server=utf8mb4 --collation-server=utf8mb4_general_ci \
--lower_case_table_names=1 --default-time_zone='+8:00'
```

上述命令还是有几个细节的，如下所示：

- --network hicode是加入之前我们新建的桥接网络hicode
- --privileged=true 是保证MySQL具有操作外部映射出的文件的权限
- --restart=always是自动启动，类似Win的自动重启
- -v /lcf/docker-data/mysql5.7/data:/var/lib/mysql \ 将mysql的数据映射到磁盘上，避免新建容后数据丢失。
- -v /lcf/docker-data/mysql5.7/conf:/etc/mysql/conf.d 将mysql的配置放置在外部磁盘，方便修改
- -e TZ=Asia/Shanghai 指定mysql的时区为 东八区，也就是我们中国时间
- -e MYSQL_ROOT_PASSWORD=yourpwd 是需要你自己设置的密码

对于mysql的配置文件，因为我们映射到了外部，所以mysql没有自动创建。如果在不映射外部的情下，去看看mysql的内部配置文件，你会发现mysql官方帮我们定义了3个配置文件

- docker.cnf 专门为docker做的优化
- mysqldump.cnf dump的时候做的优化
- mysql.cnf 需要你自己定义

当然为了方便我们不需要这么麻烦，我们就合二为一创建一个mysql.cnf就够了，命令如下：

```
cd /lcf/docker-data/mysql5.7/conf
touch mysql.cnf
```

然后在mysql.cnf添加如下内容(其实这部你省略也是没有问题的)

```
[mysql]
# 以下是docker.cnf的内容
skip-host-cache
skip-name-resolve
# 以下是mysqldump.cnf的内容
quick
quote-names
max_allowed_packet  = 16M
# 以下内容如有需要请自己添加
```

然后重启mysql就行了

```
docker restart mysql5.7
```

==========以下步骤Docker安装的MySQL应该是不需要的，默认都可以访问=========

如果你需要指定mysql所有ip都访问

```
docker exec -it mysql5.7 /bin/bash
mysql -h 127.0.0.1 -u root -p
# 此时要求你输入MySQL密码

# 此处特别注意了，需要你替换'yourpwd'为你的密码
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'yourpwd' WITH GRANT OPTION

FLUSH PRIVILEGES;
```

==========以上步骤不需要，默认都可以访问=========

solo

solo的安装注意2个细节即可，我们先看命令：

```
docker run -d  --restart=always --name solo \
    --network hicode\
    --env RUNTIME_DB="MYSQL" \
    --env JDBC_USERNAME="root" \
    --env JDBC_PASSWORD="yourpwd" \
    --env JDBC_DRIVER="com.mysql.jdbc.Driver" \
    --env JDBC_URL="jdbc:mysql://mysql5.7:3306/solo?useUnicode=yes&characterEncoding=
TF-8&useSSL=false&serverTimezone=UTC" \
    b3log/solo --listen_port=8080 --server_scheme=https --server_host=www.hicode.club
```

细节部分请注意了：

- --network=hicode加入hicode这个网络
- --env JDBC_DRIVER="com.mysql.jdbc.Driver" 因为我使用的是MySQL 5.7所以是这样的

- --env JDBC_PASSWORD="yourpwd"是安装myql时候时候指定的密码
- --server_host=www.hicode.club 需要你自己修改host
- --env JDBC_URL="jdbc:mysql://mysql5.7:3306/ 这个地方我写的不是localhost也不是127.0.0.1而是使用的mysql容器的名称mysql5.7
- --server_scheme=https 如果要通过https访问，那么此处必须是https

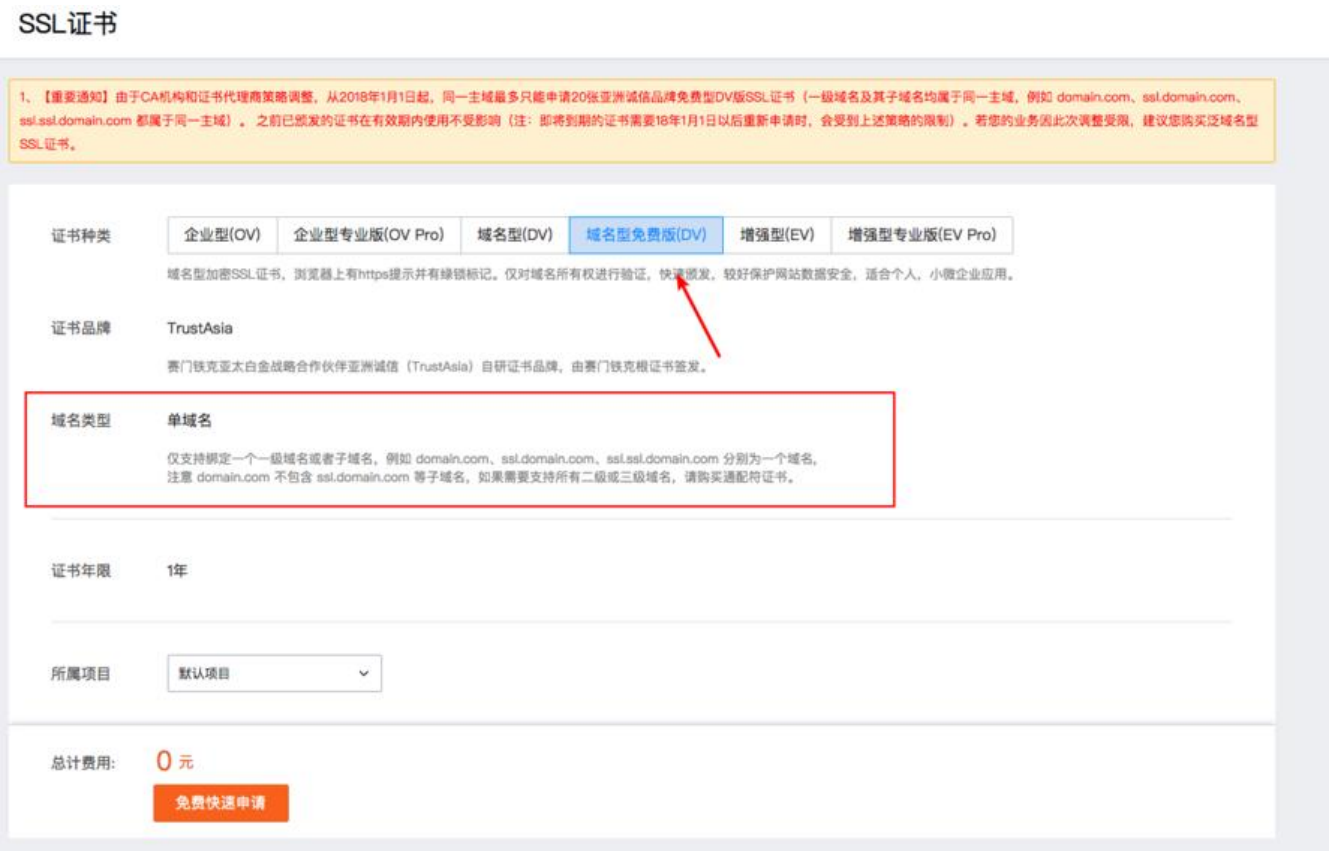
Nginx

证书申请

https相对比http，最核心的一个东西就是证书，证书有免费的也有收费的，我这里介绍下几个免费取的途径

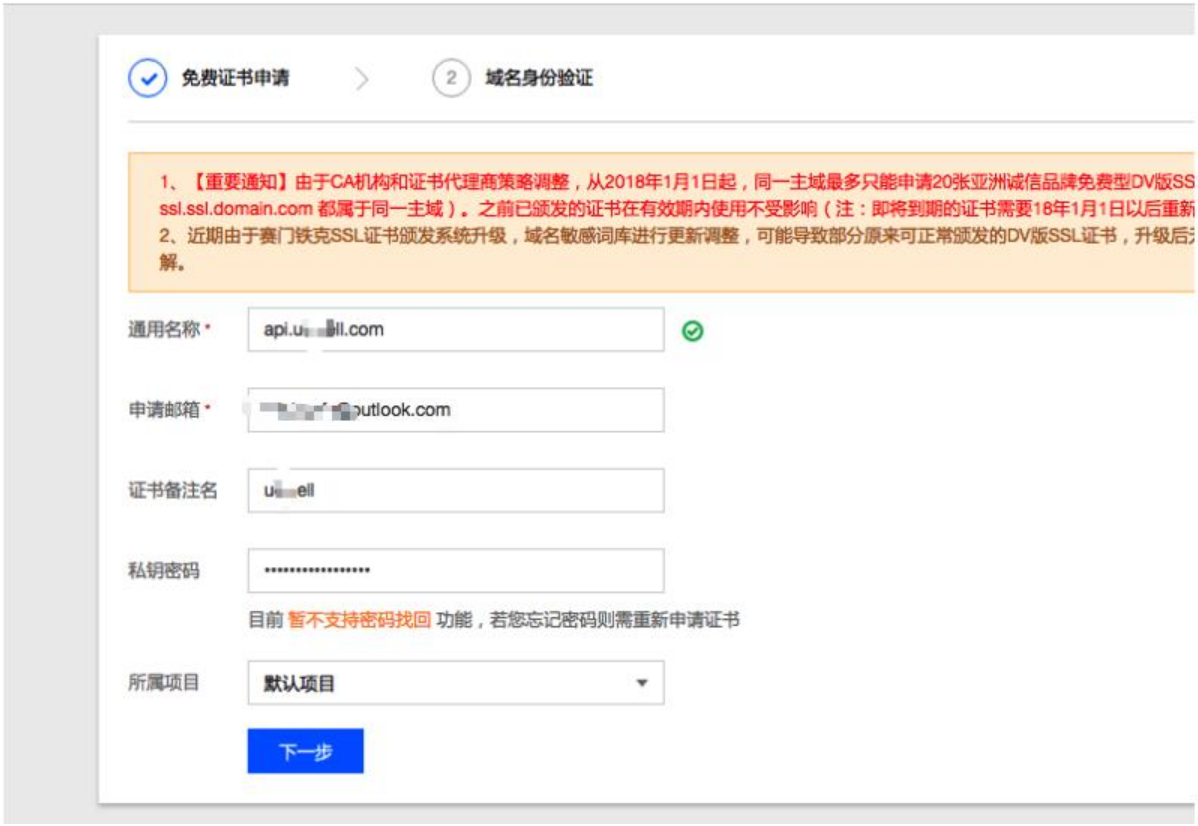
腾讯云

免费渠道：https://buy.cloud.tencent.com/ssl?fromSource=ssl



请注意选择域名免费版DV.

然后根据自己的情况进行资料填写。



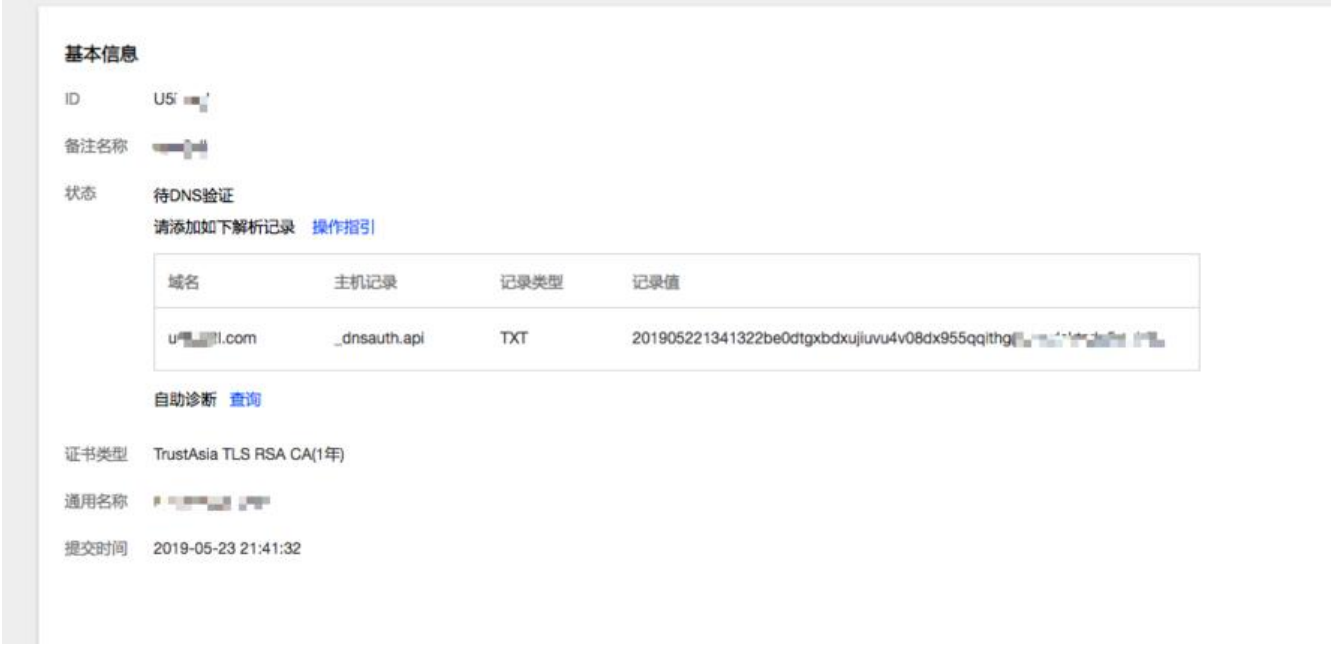
接下来就是验证域名归属，只有验证成功后才会给你颁发证书。

![image-20190523213951118](/Users/alistar/Library/Application Support/typora-user-images/image-20190523213951118.png)

如上所示，认证办法通常有2类

- 修改DNS记录：推荐使用，在DNS提供商(腾讯云、阿里云等)的解析中添加一条 txt的值。
- 文件验证：省略

确定申请后，会出现如下图



原文链接：利用 docker 部署 solo 并升级为 https

那么在添加txt记录的时候，主机记录和记录值分别做上述添加即可。

具体的流程可以参考：

- https://cloud.tencent.com/document/product/400/6814
- https://cloud.tencent.com/document/product/400/4142

阿里云

免费渠道：https://common-buy.aliyun.com/?spm=5176.7968328.1266638..5e971232kiO2Lf&ommodityCode=cas#/buy

FreeSSL

免费渠道：https://freessl.cn/

视频教程：https://blog.freessl.cn/how-to-use-freessl-issue-free-certificates/

关于证书的100问：https://blog.freessl.cn/freessl-org-frequently-asked-questions/

同时这个网站有大量通俗易懂的文章，推荐新手看看。

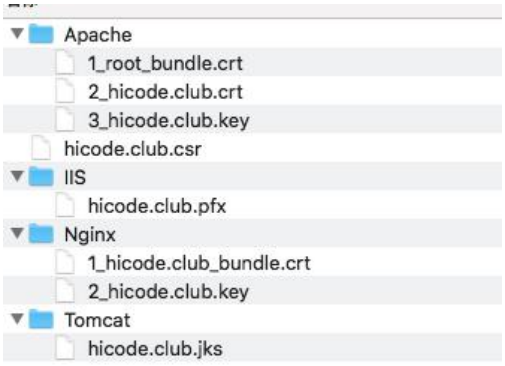
其他

https://certbot.eff.org/?utm_source=hacpai.com

https://github.com/Neilpang/acme.sh/wiki/%E8%AF%B4%E6%98%8E

证书下载

当验证了域名归属后，对应的机构就会给你颁发证书了，不同的HTTP服务器，需要不同的证书类型
下面是我在腾讯云申请的证书，默认提供了4类。



因为我用的是Nginx，所以记录如下：

- xxx.key 就是你的私钥文件，不要泄漏给别人哦。
- xxx.crt 是你的证书文件

配置Nginx

在证书下载下来后，接下来就是配置证书和反向代理了。

首先创建nginx的配置文件nginx.conf

```
cd /lcf/docker-data/nginx/conf
touch nginx.conf
```

接下来请把2个证书相关的文件移动到nginx.conf的同级目录，然后进行如下配置。

```
worker_processes 1;
events {
    worker_connections 1024;
}
http {
    include mime.types;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    default_type application/octet-stream;
    log_format main  '$remote_addr - $remote_user [$time_local] "$request" '
                     '$status $body_bytes_sent "$http_referer" '
                     '"$http_user_agent" "$http_x_forwarded_for" "$request_body"';
    access_log /var/log/nginx/logs/access.log  main;
    sendfile on;
    #开启gzip压缩
    gzip on;
    gzip_disable "MSIE [1-6]\.";
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_buffers 16 8k;
    gzip_min_length 256k;
    gzip_types text/plain application/javascript text/css application/json application/x-javascrip
 text/xml application/xml text/javascript;
```

```
    #后端的服务器
    upstream backend {
        server solo:8080 max_fails=3 fail_timeout=30s;
    }

    #个人博客
    server {
        # 最新写法，ssl on 的写法已经不推荐了！
        listen 443 ssl;
        server_name hicode.club www.hicode.club;
          # 证书相关,https新增
        ssl_certificate hicode.club_bundle.crt;
        ssl_certificate_key hicode.club.key;
        ssl_session_timeout 5m;

        access_log /var/log/nginx/logs/hicode/access.log  main;
        location / {
            proxy_pass http://backend$request_uri;
            proxy_set_header Host $host:$server_port;
            proxy_set_header X-Real-IP  $remote_addr;
            proxy_set_header http_x_forwarded_for $remote_addr;
            client_max_body_size 10m;
        }
    }
    server {
        # http跳转到https，这样就只存在http的博客了
        listen 80;
        server_name hicode.club www.hicode.club;
        rewrite ^(.*)$    https://$host$1    permanent;
    }
}
```

上述内容中有2个server

- 第一个是　https的配置，关键是其中证书的配置。
- 第二是　http跳转到https的配置。

安装Nginx

然后执行如下命令启动nginx的容器

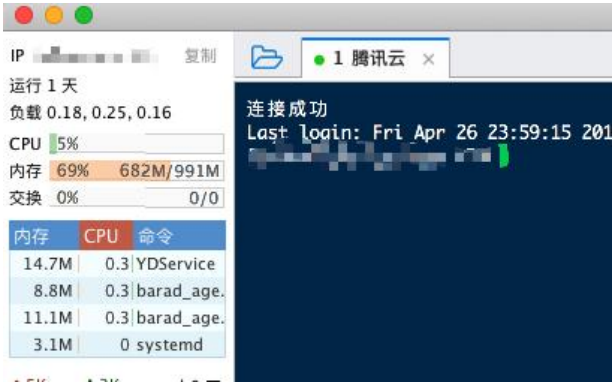
```
docker run -p 80:80 -p 443:443 -m 200m --restart always \
 --name nginx --network hicode \
 -v /lcf/docker-data/nginx/www:/www \
 -v /lcf/docker-data/nginx/conf/:/etc/nginx/ \
 -v /lcf/docker-data/nginx/logs:/var/log/nginx/logs \
 -v /lcf/docker-data/nginx/wwwlogs:/wwwlogs \
 nginx
```

- 相比于单纯的　http，多一个443的端口映射。

其他

基本上，按照上面操作，就可以搞定了，docker用好了，很方便，因为一行指令搞定的时候，不需要折腾环境了。

有朋友担心资源占用的问题，这里我截图，给大家看看情况：



好了，差不多写到这里，有问题，欢迎留言。

最后归纳下同http部署方式的不同：

- 申请证书
- solo指定为https
- nginx多映射一个443端口

文章发布自：hicode.club，转载请注明出处，谢谢！

原文链接：利用 docker 部署 solo 并升级为 https