記事
Akio Hashimoto · 2020年10月25日 5m read

Dockerを利用した開発環境下でのIRISへのSSL接続方法

Docker環境下でWebアプリのコンテナとIRISのコンテナをRESTfulAPIで連携した時の
SSL化方法。
ちなみにこの方法で管理ポータル等もSSL通信が可能になります。

参考サイト：
* https://one-it-thing.com/63/

構成

- Webプリケーション -> Vue.js (http://192.168.10.100:3014)
- REST ful API -> IRISのDockerコンテナ (http://192.168.10.100:9092)

※ Vue.jsはローカル上でホットリロードを利用した開発環境。
※ IRISはローカル上に構築したDockerコンテナ下で稼働中。

今回、IRISをDockerコンテナで稼働させる方法は割愛します。

WebアプリケーションをHTTPSで利用する

Vue.jsで開発中のアプリケーションをhttpsでホットリロード利用出来るように起動する。

```
npm run serve -- --https
```

これで開発中のデバッグ画面をhttpsで呼び出して利用する事ができるようになるが、このままでは、httpsからhttpにアクセスする事が出来ない為、IRISのRESTを呼び出すことが出来ない。

Nginxでプロキシサーバーを立てる

Nginxでプロキシサーバーを立てて、httpsでアクセスされたら全てIRISのサーバーにリダイレクトするようにする。

https://192.168.10.100/ -> http://192.168.10.100:9092/

Dockerfile

```
FROM nginx

ENV _NGINX_CONF /etc/nginx

RUN mkdir -p $_NGINX_CONF/ssl
RUN apt-get update
RUN apt-get install openssl
```

※ NginxのコンテナにOpenSSLをインストールしておく

docker-compose.yml

```
version: '3.7'
services:
    proxy:
        build:
            context: .
        image: proxyiris:latest
        ports:
            - "443:443"
        restart: always
        environment:
            - TZ=Asia/Tokyo
```

Nginxのコンテナを作成し起動する。

```
docker-compose up -d
```

自己証明書を作成する

コンテナにログイン

```
docker exec -it proxy_proxy_1 /bin/bash
```

作業ディレクトリを準備

- 作業ディレクトリに移動

```
cd /etc/nginx/ssl
```

- openssl.cnfの変更

```
cp /etc/ssl/openssl.cnf /etc/ssl/openssl.cnf.org
```

※ openssl.cnfのバックアップ

openssl.cnf

```
???
[ CA_default ]

#dir        = ./demoCA      # Where everything is kept
dir     = .     # Where everything is kept
???
```

※編集方法は適宜。私の場合はローカルにコピーしてエディタで修正しました。

ローカルへのコピー方法

```
docker cp proxy_proxy_1:/etc/ssl/openssl.cnf {???????}
```

- CA署名時用のシリアルナンバーファイルを準備する

```
touch index.txt
echo 00 > serial
```

CA（証明書認証局）を作成

CA秘密鍵の作成。

```
openssl genrsa -aes256 -out cakey.pem 2048
```

※パスワードを入力します。

CA証明書リクエストの作成。

```
openssl req -new -key cakey.pem -config /etc/ssl/openssl.cnf -out cacert.csr
```

※Common name (e.g, server FQDN or YOUR name) []:192.168.10.100

x509 v3拡張領域の追加内容をファイルで指定する。

```
touch cav3.txt
echo "basicConstraints = critical, CA:true" > cav3.txt
echo "keyUsage = critical, cRLSign, keyCertSign" >> cav3.txt
echo "subjectKeyIdentifier=hash" >> cav3.txt
```

CA証明書の作成。

```
openssl ca -in cacert.csr -selfsign -keyfile cakey.pem -notext -config /etc/ssl/opens
sl.cnf -outdir . -days 365 -extfile cav3.txt -out cacert.pem
```

※ iOS13以降、有効期限は13ヶ月となっているので、今回は1年にしています。

サーバー証明書を作成

サーバー秘密鍵の作成。

```
openssl genrsa -aes256 -out server.key 2048
```

※パスワードを入力します。

サーバー証明書リクエストの作成。

```
openssl req -new -key server.key -config /etc/ssl/openssl.cnf -out server.csr
```

※「Organization Name」と「Common Name」は認証局の時と同じにしておく。

x509 v3拡張領域の追加内容をファイルで指定する。

```
touch v3server.txt
echo "[SAN]" > v3server.txt
echo "basicConstraints = CA:false" >> v3server.txt
echo "keyUsage = critical, digitalSignature, keyEncipherment" >> v3server.txt
echo "extendedKeyUsage = serverAuth" >> v3server.txt
echo "authorityKeyIdentifier=keyid,issuer" >> v3server.txt
echo >> v3server.txt
echo "subjectAltName=@alt_names" >> v3server.txt
echo "basicConstraints=CA:FALSE" >> v3server.txt
echo "[alt_names]" >> v3server.txt
echo "DNS.1=localhost" >> v3server.txt
echo "IP.1=192.168.10.100" >> v3server.txt
echo "IP.2=127.0.0.1" >> v3server.txt
```

サーバー証明書の作成。

```
openssl ca -in server.csr -config /etc/ssl/openssl.cnf -keyfile cakey.pem -outdir . -
extfile v3server.txt -extensions SAN -out server.crt -days 365
```

パスワード無しサーバー秘密鍵の作成。

```
openssl rsa -in server.key -out nopass_server.key
```

Nginxに設定ファイルを追加して再起動する

設定ファイル

設定ファイルの作成

```
touch /etc/nginx/conf.d/irisproxy.cnf
```

irisproxy.cnf

```
server {
  listen 443 ssl http2 default_server;
  ssl_certificate         /etc/nginx/ssl/server.crt;
  ssl_certificate_key     /etc/nginx/ssl/nopass_server.key;
  location / {
    proxy_pass http://192.168.10.100:9092;
    proxy_redirect          http:// https://;
    proxy_set_header        Host            $host;
```

```
    proxy_set_header        X-Real-IP       $remote_addr;
    proxy_set_header        X-Forwarded-Proto $scheme;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
  }
}
```

コンテナの再起動

```
docker restart proxy_proxy_1
```

以上で、ローカル上のIRISへHTTPS接続出来るようになります。
Webアプリから「http://...:9092/xxx/」とアクセスしてRESTを呼び出している箇所を
「https://.../xxx/」と置き換えるだけです。

#Docker #REST API #SSL #開発環境 #InterSystems IRIS

ソースURL:
https://jp.community.intersystems.com/post/docker%E3%82%92%E5%88%A9%E7%94%A8%E3%81%97%E3%81%9F%E9%96%8B%E7%99%BA%E7%92%B0%E5%A2%83%E4%B8%8B%E3%81%A7%E3%81%AEiris%E3%81%B8%E3%81%AEssl%E6%8E%A5%E7%B6%9A%E6%96%B9%E6%B3%95