Using nginx-proxy

The compose templates provided by the SSDT do not publish ports for the applications outside of the docker host. Each ITC must decide how they are going to be mapping the Tomcat port from each application (port 8080) to make it available to external web browsers. This article explains how to use `nginx-proxy` to create a reverse proxy which automatically updates as containers are started and stopped.

The goal of this article is to

- start with a basic reverse proxy
- add SSL encryption with a signed (or self-signed) certificate
- Optionally, use LetsEncrypt.org to automatically generate signed certificates

Tutorial

Below are steps to configure nginx-proxy on your docker host. The tutorial assumes the following:

Docker host	docker-01.example.com
sub-domain for virtual hosts	demo.example.com

Create a Reverse Proxy on port 80

1. Wildcard DNS: Create DNS records in the domain to point to the docker host.

```
*.demo.example.com   IN    CNAME docker-01.example.com
```

   This wildcard will be used for all virtual hosts on this machine. If you intend to also use other subdomains for this host, you can add hostnames in other domains. For example, if "sampletown" has their own domain, then you could also add these RR's to sampletown's domain:

```
usas                 IN CNAME docker-01.example.com
usps                 IN CNAME docker-01.example.com
```

   Because USAS and USPS run in separate containers they each must have their own virtual domain.
2. Create a directory on the docker host, e.g. `/data/proxy/` and create the following `docker-compose.yml`:

```
version: "3"
services:
  proxy:
     image: jwilder/nginx-proxy:alpine
     restart: always
     volumes:
       - /var/run/docker.sock:/tmp/docker.sock:ro
       - ./certs:/etc/nginx/certs:ro
       - ./vhost.d:/etc/nginx/vhost.d
       - ./html:/usr/share/nginx/html
       - ./conf.d:/etc/nginx/conf.d
     environment:
       - DEFAULT_HOST=demo.example.com
     ports:
       - "443:443"
       - "80:80"
```

   The volume definitions are not strictly necessary at this point. However, adding them here is harmless and will make the subsequent instructions easier.
3. Run the proxy:

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

   You should now be able to visit: http://sampletown.demo.example.com/. However, you'll receive a "*503 Service Temporarily Unavailable*" because nginx-proxy does not know how to discover our service containers.

4. In your district's compose project, add (or modify) `docker-compose.override.yml` file to define VIRTUAL_HOST and VIRTUAL_PORT environments for each service:

   ```
   usasapp:
     environment:
       - VIRTUAL_HOST=sampletown-usas.demo.example.com
       - VIRTUAL_PORT=8080
   uspsapp:
     environment:
       - VIRTUAL_HOST=sampletown-usps.demo.example.com
       - VIRTUAL_PORT=8080
   ```

   The nginx-proxy container will monitor docker events. Each time a container starts or stops, which has a VIRTUAL_HOST variable, it will create a new nginx configuration which reverse proxies port 80 for the virtual domain to 8080 of the container.
5. Rebuild the district's containers with:

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

   After the apps start, you should be able to reach the applications at: http://sampletown-usas.demo.example.com and http://sampletown-usps.demo.example.com.
6. Repeat the previous two steps for each school district.
7. If the host is not working, you can check the nginx-proxy log files with:

   ```
   cd /data/proxy
   docker-compose logs
   ```

HTTPS proxy on port 443

If you wish to use LetsEncrypt.org for automated certificate signing, then skip this section.

Now that we have a reverse proxy, we can secure the port using HTTPS. In this example, we are creating a wildcard certificate to match the wildcard DNS entry. In this example, the "*Common Name*" is "`*.demo.example.com`".

1. Create a certificate and CSR in the proxy's ./certs directory (this volume was mounted in the proxy's docker-compose.yml file above).

   ```
   data/proxy# mkdir -p certs
   data/proxy# cd certs
   data/proxy/certs# # Create a private key:
   data/proxy/certs# openssl genrsa -out demo.example.com.key 2048
   data/proxy/certs# # Create a CSR from the new key:
   data/proxy/certs# openssl req -new -sha256 -key demo.example.com.key -out demo.example.com.csr
   -----
   Country Name (2 letter code) [AU]:US
   State or Province Name (full name) [Some-State]:Ohio
   Locality Name (eg, city) []:Archbold
   Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your Organization name
   Organizational Unit Name (eg, section) []:Your OU
   Common Name (e.g. server FQDN or YOUR name) []:*.demo.example.com.
   Email Address []:hostmaster@example.com
   Please enter the following 'extra' attributes
   to be sent with your certificate request
   A challenge password []:
   An optional company name []:
   ```

2. Send the CSR to your favorite signing authority, or self sign it:

   ```
    data/proxy/certs# openssl x509 -req -sha256 -days 3650 -in  demo.example.com.csr -signkey  demo.example.
   com.key -out  demo.example.com.crt
   ```

3. Configure nginx to listen on port 443. Add port mapping to the proxy's docker-compose.yml file:

```
proxy:
   image: jwilder/nginx-proxy
   restart: always
   volumes:
     - /var/run/docker.sock:/tmp/docker.sock:ro
     - ./certs:/etc/nginx/certs:ro
     - ./vhost.d:/etc/nginx/vhost.d
     - ./html:/usr/share/nginx/html
   environment:
     - DEFAULT_HOST=demo.example.com
   ports:
     - "80:80"
     - "443:443"
```

4. Recreate the proxy container with:

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

This exposes port 443 for SSL. We are leaving port 80 exposed because the nginx-proxy will automatically redirect port 80 to 443. Now we can access our application at: https://sampletown.demo.example.com/. If the cert is self-signed, you'll get a browser warning. It will go away when/if you have the certificate signed.

After receiving the signed certificate from the signing authority, replace the `.crt` file created above with the signed certificate. In the example above, the self-signed certificate is named `demo.example.com.crt`. That file should be replaced with the file from the signing authority. Note: The names of the certificate files are important. The certificate file name must be must match the domain name it applies to.   Again, from the above example, the wildcard domain name is *.demo.example.com so the certificate and keys must be named `demo.example.com.crt` and `demo.example.com.key`.

By convention, nginx-proxy will use the domain name to find the most specific certificate first and then drop prefixes until it finds a match.   In this case, it will look for sampletown.demo.example.com.crt and then `demo.example.com.crt` . This allows you to have different signed certificates for different domain names on the same proxy.

Automatic Signed Certificates with LetsEncrypt.org

LetsEncrypt is a service which issues free automated certificates. Before using the service, you should review the https://letsencrypt.org/about/ and current rate limits.

LetsEncrypt.org is a service which automates the process of creating, signing, installing and renewing Domain Validation (DV) Certificates. These are certificates provide the lowest level of host verification but do ensure encrypted traffic to the users browser.

The steps below show how to configure an extra container to automatically create and install certificates using jrcs/letsencrypt-nginx-proxy-companion. Thi s is a non-intrusive way to add letsencrypt to an existing proxy configuration.

If the certificate renewal appears to stop working, make sure you have the most current version of the image(s) and restart the application. To pull a newer image of ngnix-proxy for example:

```
docker pull jwilder/nginx-proxy:alpine
```

Requirements:

You ***must expose port 80 and 443 of your docker host*** to the outside via your firewall. That is, the docker host must have a public IP address and be accessible on both port 80 and 443 to the outside. DNS entries must exist in the global DNS for the virtual host(s) which point to the docker host's IP address. When your host makes a certificate request, LetsEncrypts service will callback to your host for verification. If the remote service can not reach your host, then they can not verify your control of the domain name and the signing request will fail.

Steps to enable LetsEncrypt

1. First, if you haven't already exposed port 80 and 443 on the nginx-proxy. The proxy's docker-compose.yml should look like this:

```
version: "3"
services:
  proxy:
     image: jwilder/nginx-proxy:alpine
     restart: always
     volumes:
       - /var/run/docker.sock:/tmp/docker.sock:ro
       - ./certs:/etc/nginx/certs:ro
       - ./vhost.d:/etc/nginx/vhost.d
       - ./html:/usr/share/nginx/html
       - ./conf.d:/etc/nginx/conf.d
     environment:
       - DEFAULT_HOST=demo2.ssdt.io
     ports:
       - "443:443"
       - "80:80"
     labels:
       - "com.github.jrcs.letsencrypt_nginx_proxy_companion.nginx_proxy=true"
  le:
    image: jrcs/letsencrypt-nginx-proxy-companion
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./certs:/etc/nginx/certs:rw
      - ./vhost.d:/etc/nginx/vhost.d:rw
      - ./html:/usr/share/nginx/html:rw
    environment:
      - NGINX_PROXY_CONTAINER=proxy_proxy_1
```

Notice that the companion shares the volumes with the nginx-proxy container. The companion container will monitor docker events and automatically create certificates and place them in the correct directories for the primary nginx server.

2. Recreate the proxy with

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

3. For each district you want to have a certificate created for, edit the `docker-compose.override.yml` file to create the LETSENCRYPT_HOST and LETSENCRYPT_EMAIL variables, like:

```
usasapp:
  environments:
    - VIRTUAL_HOST=usas.sampletown.org
    - VIRTUAL_PORT=8080
    - LETSENCRYPT_HOST=usas.sampletown.org
    - LETSENCRYPT_EMAIL=hostmaster@example.com
```

The companion image will create, sign and install a certificate for any service with the LETSENCRYPT_HOST variable. In most cases, the VIRTUAL_HOST and LETSENCRYPT_HOST will be set to the same value.

4. Restart the district's services with:

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

Because nginx-proxy and the companion container use separate environment variables, you can use a traditionally signed certificate for some hosts (see previous section) and letsencrypt certificates for others. For example, you might use a wildcard certificate for "*.demo.example.com" hosts and a letsencrypt certificate for a district's "vanity domain" (usas.sampletown.org and usps.sampletown.org).

We recommend making adjustments to the timeout configuration.

Related articles

- Backup USxS Databases
- Working with SSDT Supplied Scripts
- Using nginx-proxy
- Install Docker and Docker tools on Ubuntu

- Adjusting nginx-proxy Timeout Configuration