Reverse Proxy

Setting Up Reverse Proxy

This section explains you how to setup a reverse procxy for **Openemail dockers**

You don't need to change the Nginx site that comes with `nginx-openemail` container.
Openemail trusts the default gateway IP 172.22.1.1 as proxy.

1. Make sure you change HTTP_BIND and HTTPS_BIND in `openemail.conf` to a local address and set the ports accordingly, for example:

```
HTTP_BIND=127.0.0.1
HTTP_PORT=8080
HTTPS_BIND=127.0.0.1
HTTPS_PORT=8443
```

IMPORTANT: Do not use port 8081!

> **Warning**
>
> Make sure you run `generate_config.sh` before you enable any site configuration examples below. The script `generate_config.sh` copies snake-oil certificates to the correct location, so the services will not fail to start due to missing files.

> **Info**
>
> Using the site configs below will **forward ACME requests to Openemail** and let it handle certificates itself. The downside of using Openemail as ACME client behind a reverse proxy is, that you will need to reload your webserver after acme-openemail changed/renewed/created the certificate. You can either reload your webserver daily or write a script to watch the file for changes. On many servers logrotate will reload the webserver daily anyway.
>
> If you want to use a local certbot installation, you will need to change the SSL certificate parameters accordingly. **Make sure you run a post-hook script** when you decide to use external ACME clients. You will find an example at the bottom of this page.

2. Configure your local webserver as reverse proxy:

Apache 2.4 as a Reverse Proxy

** Install required modules:**

```
a2enmod rewrite proxy proxy_http headers ssl
```

We rewrite to `HTTPS`, `but keep requests to` autoconfig.*`` on a plain session.

Let's Encrypt will follow our rewrite, certificate requests will work fine.

Take care of highlighted lines.. You need to asjut

```
<VirtualHost *:80>
  ServerName CHANGE_TO_openemail_HOSTNAME
  ServerAlias autodiscover.*
  ServerAlias autoconfig.*
  RewriteEngine on

  RewriteCond %{HTTP_HOST} ^autoconfig\. [NC]
  RewriteRule ^ - [S=1]
  RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI}# [L,NE,R=permanent]
  RewriteRule ^ /autoconfig.php [PT]

  ProxyPass / http://127.0.0.1:8080/
  ProxyPassReverse / http://127.0.0.1:8080/
  ProxyPreserveHost On
  ProxyAddHeaders On
  RequestHeader set X-Forwarded-Proto "http"
</VirtualHost>
<VirtualHost *:443>
  ServerName CHANGE_TO_Openemail_HOSTNAME
  ServerAlias autodiscover.*

  # You should proxy to a plain HTTP session to offload SSL processing
  ProxyPass / http://127.0.0.1:8080/
  ProxyPassReverse / http://127.0.0.1:8080/
  ProxyPreserveHost On
  ProxyAddHeaders On
  RequestHeader set X-Forwarded-Proto "https"

  SSLCertificateFile openemail_PATH/data/assets/ssl/cert.pem
  SSLCertificateKeyFile openemail_PATH/data/assets/ssl/key.pem

  # If you plan to proxy to a HTTPS host:
  #SSLProxyEngine On

  # If you plan to proxy to an untrusted HTTPS host:
  #SSLProxyVerify none
  #SSLProxyCheckPeerCN off
  #SSLProxyCheckPeerName off
  #SSLProxyCheckPeerExpire off
</VirtualHost>
```

Nginx as a Reverse Proxy

In our Nginx reverse proxy template, we rewrite all requests to HTTPS, while keeping autoconfig.* domains on a plain session.

Let's Encrypt will follow our rewrite, certificate requests will work fine.

Take care of highlighted lines.

```
server {
  listen 80 default_server;
  listen [::]:80 default_server;
  server_name CHANGE_TO_openemail_HOSTNAME autodiscover.*;
  return 301 https://$host$request_uri;
}
server {
  listen 80;
  listen [::]:80;
  server_name autoconfig.*;
  rewrite ^/(.*)$ /autoconfig.php last;
  location / {
    proxy_pass http://127.0.0.1:8080/;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    client_max_body_size 0;
  }
}
server {
  listen 443;
  server_name CHANGE_TO_openemail_HOSTNAME autodiscover.* autoconfig.*;

  ssl on;
  ssl_certificate openemail_PATH/data/assets/ssl/cert.pem;
  ssl_certificate_key openemail_PATH/data/assets/ssl/key.pem;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ciphers HIGH:!aNULL:!MD5;

  location / {
      proxy_pass http://127.0.0.1:8080/;
      proxy_set_header Host $http_host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
      client_max_body_size 0;
  }
}
```

HAProxy as a Reverse Proxy

Important/Fixme: This example only forwards HTTPS traffic and does not use Openemails built-in ACME client.

```
frontend https-in
  bind :::443 v4v6 ssl crt openemail.pem
  default_backend openemail

backend openemail
  option forwardfor
  http-request set-header X-Forwarded-Proto https if { ssl_fc }
  http-request set-header X-Forwarded-Proto http if !{ ssl_fc }
  server openemail 127.0.0.1:8080 check
```

Using post-hook Scripts

Optional: Post-hook script for non-openemail ACME clients

Using a local certbot (or any other ACME client) requires to restart some containers, you can do this with a post-hook script. Make sure you change the pathes accordingly:

```
#!/bin/bash
cp /etc/letsencrypt/live/my.domain.tld/fullchain.pem /opt/openemail/data/assets/
ssl/cert.pem
cp /etc/letsencrypt/live/my.domain.tld/privkey.pem /opt/openemail/data/assets/
ssl/key.pem
# Either restart...
#postfix_c=$(docker ps -qaf name=postfix-openemail)
#dovecot_c=$(docker ps -qaf name=dovecot-openemail)
#nginx_c=$(docker ps -qaf name=nginx-openemail)
#docker restart ${postfix_c} ${dovecot_c} ${nginx_c}
# ...or reload:
docker exec $(docker ps -qaf name=postfix-openemail) postfix reload
docker exec $(docker ps -qaf name=nginx-openemail) nginx -s reload
docker exec $(docker ps -qaf name=dovecot-openemail) dovecot reload
```