Traffic Ingress/NGINX

NEEDS UPDATE: Traffic now flows to HAProxy first and then to NGINX.

The Melton Backend would communicate with the front-ends with HTTP/HTTPS protocols. These protocols are reserved to use port 80 and 443 respectively. Since the server might run multiple services that need to be accessible via those ports, we would need an ingress to manage that port and pass on the traffic based on the URL.

The current choice I went with was NGINX, NGINX is a high performance load balancer, SSL termination and Ingress service. It receives a request at its defined ports (80/443) and looks up its own config and decides where to forward the traffic to.

In our scenario, The Melton API listens on port `8080` and has no built in SSL termination - i.e it uses unencrypted communication. Since this is generally a horrible idea, we will terminate SSL with NGINX and pass traffic from NGINX to the Melton API in an unencrypted manner - This isn't too big of a security risk since the unencrypted part happens only inside the confines of the server.

The Melton API also has some static files stored that need to be accessed, these are placed under the `/static/` endpoint and mirror the filestructure on the server. A request with this suffix should respond with the corresponding file on the server.

The config block of the NGINX service and the description are as below:

```
server {
        listen 443; □□□□□□□□□□# Listens on port 443 - HTTPS
        server_name meltonapp.com;□□□□□□□ □# The URL to check for
        ssl on;□□□□□□□□□□□□□# Defines if NGINX should use SSL
        ssl_certificate /------/fullchain.pem;□□□□□# Defines the location of the SSL certificate
        ssl_certificate_key /------/meltonapp.com/privkey.pem;□# Defines the location of the certificate key
        ssl_session_cache shared:SSL:10m;□□□□□□# Cache location for SSL

        location /static/ {□□□□□□□□□□# If the /static/ endpoint is called, the data from a specific location is served
            root /usr/local/share/staticfiles;□□□□□# Location of data to server
        }

        location / {                                    # In all other scenario's it passes on the traffic to the below location
                proxy_pass http://localhost:8000;□□□    # The port and location the Melton API is accessible from
```

```
            proxy_set_header Host $host;□□□□    # Passing on the headers
            proxy_redirect http:// https://; □□□    # Defines if a redirect is needed
        }
    }

    server {□□□□□□□□□□□□□□ □# A new server block to redirect HTTP traffic
        listen 80;□□□□□□□□□□□□□# Listens on port 80 - HTTP
        return 301 https://$host$request_uri;□□□□□□# Respond with a 301 (Redirect) to the HTTPS endpoint
    }
```

To harden the server a bit, ufw is used. UFW is a built in firewall that can be used to block external port access to the server. This can be used to force all traffic to pass through the ingress and use HTTPS.

TODO: Add info on /static/ file mapping and when a request URL is resolved by NGINX and when its resolved by the melton docker image

---

Revision #6
Created 2 August 2020 11:36:01 by pari
Updated 21 September 2020 21:03:00 by pari