The recommended way to run Faraday using SSL is through NGINX.

Nginx

You can find a detailed guide on how to install it in the official NGINX documentation.

After installing and configuring NGINX the setup should be as follows:

- CouchDB on port `5984` using HTTP (CouchDB config files)
- Faraday Server on port `5985` using HTTP (`~/.faraday/config/server.ini`)
- GTK using HTTPS (`~/.faraday/config/user.xml`) and run:`$ python2 faraday.py --cert path_to_cert(PEM format)`
- Web UI using `https://example_domain:port/_ui`
- NGINX on port `80` redirecting to HTTPS

Below you can find a sample config file for NGINX. Please keep in mind that you need to change `example_domain`, `example_cert` and `example_key` to your domain, cert and key.

```
# Faraday conf
# don't send the nginx version number in error pages and Server header
server_tokens off;

add_header X-Frame-Options SAMEORIGIN;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
#add_header Content-Security-Policy "default-src 'self'; script-src 'self'
'unsafe-inline' 'unsafe-eval' https://ssl.google-analytics.com
https://assets.zendesk.com https://connect.facebook.net; img-src 'self'
https://ssl.google-analytics.com https://s-static.ak.facebook.com
https://assets.zendesk.com; style-src 'self' 'unsafe-inline'
https://fonts.googleapis.com https://assets.zendesk.com; font-src 'self'
https://themes.googleusercontent.com; frame-src https://assets.zendesk.com
https://www.facebook.com https://s-static.ak.facebook.com https://tautt.zendesk.com;
object-src 'none'";

server {
        listen *:443;
        server_name example_domain.com;

        ssl on;
        ssl_certificate /etc/ssl/example_cert.pem;
        ssl_certificate_key /etc/ssl/example_key.key;
        # enable session resumption to improve https performance
        # http://vincent.bernat.im/en/blog/2011-ssl-session-reuse-rfc5077.html
        ssl_session_cache shared:SSL:50m;
        ssl_session_timeout 5m;

        # enables server-side protection from BEAST attacks
        # http://blog.ivanristic.com/2013/09/is-beast-still-a-threat.html
        ssl_prefer_server_ciphers on;
        # disable SSLv3(enabled by default since nginx 0.8.19) since it's less
secure then TLS http://en.wikipedia.org/wiki/Secure_Sockets_Layer#SSL_3.0
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        # ciphers chosen for forward secrecy and compatibility
        #
http://blog.ivanristic.com/2013/08/configuring-apache-nginx-and-openssl-for-forward
-secrecy.html
        #ssl_ciphers 'AES128+EECDH:AES128+EDH';
        ssl_ciphers
"ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:
DHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA
-AES256-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES128-SHA256:DHE-RS
A-AES256-SHA:DHE-RSA-AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES256-
GCM-SHA384:AES128-GCM-SHA256:AES256-SHA256:AES128-SHA256:AES256-SHA:AES128-SHA:DES-
CBC3-SHA:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!MD5:!PSK:!RC4";

        # config to enable HSTS(HTTP Strict Transport Security)
https://developer.mozilla.org/en-US/docs/Security/HTTP_Strict_Transport_Security
        # to avoid ssl stripping
https://en.wikipedia.org/wiki/SSL_stripping#SSL_stripping
        add_header Strict-Transport-Security "max-age=31536000;
includeSubdomains;";

        location / { #~ ^/(.*)/_changes {
                proxy_pass http://localhost:5985;
                proxy_redirect off;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Ssl on;

                location /_utils {
                        deny all;
                        proxy_pass http://localhost:5985;
                        proxy_redirect off;
                        proxy_set_header Host $host;
                        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                        proxy_set_header X-Forwarded-Ssl on;
                }
        }
}

server {
       listen *:80;
       server_name example_domain.com;
       return 301 https://$host$request_uri;
}
```

For information on *how to generate self signed certificates* you can read Apache's FAQ on how to do this.

Even though we recommend the configurations by nginx explained above, we also support SSL through Apache.

Apache

Place the Apache configuration file on the respective location.

```
    # Enable session resumption to improve https performance
    SSLSessionCache shmcb:/var/cache/mod_ssl/scache(512000)
    SSLSessionCacheTimeout  300

   <VirtualHost *:80>
      Redirect permanent / https://127.0.0.1/
   </VirtualHost>

   <VirtualHost *:443>
    # Apache logs configuration.

    ServerName localhost
    ServerAdmin webmaster@localhost
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    SSLEngine On
# Dont use SSL
SSLProtocol all -SSLv2 -SSLv3
# Server-side protection from BEAST attacks
SSLHonorCipherOrder on
# Use only secure ciphers
SSLCipherSuite
"ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:
DHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA
-AES256-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES128-SHA256:DHE-RS
A-AES256-SHA:DHE-RSA-AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES256-
GCM-SHA384:AES128-GCM-SHA256:AES256-SHA256:AES128-SHA256:AES256-SHA:AES128-SHA:DES-
CBC3-SHA:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!MD5:!PSK:!RC4"

# Set the path to SSL certificate
    SSLCertificateFile /home/user/ca.crt
    SSLCertificateKeyFile /home/user/ca.key

# Reverse proxy configuration
    ProxyPreserveHost On
# ProxyPass /server-status !
    ProxyPass "/" http://127.0.0.1:5985/
    ProxyPassReverse "/" http://127.0.0.1:5985

# Security headers
Header set X-Frame-Options SAMEORIGIN
Header set X-Content-Type-Options nosniff
Header set X-XSS-Protection "1; mode=block"
Header set Strict-Transport-Security "max-age=31536000; includeSubdomains;"

<Location /_utils>
    Order deny,Allow
    Allow from localhost
    Deny from all
</Location>
```

```
```

Troubleshooting

To ensure that the issue is not with your certificates, test from the command line using

```
$ curl -k -v https://127.0.0.1:5984/
```

You can test your certificates separately using:

```
$ openssl s_server -key <keyfile> -cert <certfile> -www
```

Make sure that when you create the certificate the commonName field contains the name of your domain.

If for any chance you get an error stating "SSL certificate validation failure" when running GTK, re-generate the certificate and run again.

Certificate signed with internal CA.

If you are using a certificate signed by a internal CA you need follow the next steps for connect the Faraday client to the server using this certificate.

For example, with the following CA chain:

Root CA (root.crt) -> intermediate CA (intermediate.crt) -> server cert (server.crt)

Take all the three certs in PEM format and append all in one file:

touch bundle.crt

cat root.crt >> bundle.crt

cat intermediate.crt >> bundle.crt

cat server.crt >> bundle.crt

Execute faraday: python2 ./faraday.py --cert bundle.crt