

# Build pipeline: Edft.DeepSea.Compass.Ssl

Use certbot to create a SSL certificate for compass server: <https://mindsers.blog/post/https-using-nginx-certbot-docker/>

Note: EDFT uses a chain of certificates that are required for any access to an SSL address.

The tutorial listed above and most others you will find by a google search, use the free service LetsEncrypt.

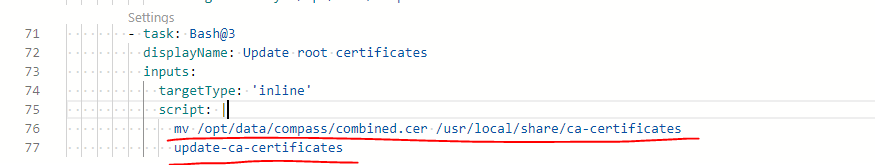
The main difference for an EDFT setup being that the whole certificate chain needs to be exported as a single base64 cert or CER and added to our certbot code to be able to generate the private key PEM files needed to perform client side handshakes when running Nginx.

To get a working certificate chain for use on the EDFT domain then follow these steps:

Go to the acme server at the following address in your web browser <https://acme.edftrading.com/acme/acme/directory> and export the Root, SubCa and Acme certificates (EDFT’s implementation of ACME cert service)

This article explains how to download and export a certificate chain as required by certbot <https://docs.microsoft.com/en-us/azure/application-gateway/mutual-authentication-certificate-management>

1. Update root certs



Move the EDFT root combined cert to the user cert location and install it with update-ca-certificates command.

This is required to authenticate within EDFT ACME server

Required for 2 things:

1. smoke test – needs to trust the compass site, which has the cert generated by the ACME server, so it needs to trust the CA.

2. certbot also needs to trust the ACME server AC – certbot uses these cert to make sure the ACME server

Plan to move this out to a separate nightly pipeline.

Right now it’s part of the release pipeline – if we don’t release for a while and the cert expires, we won’t get the new cert.

2. run the certbot and dummy site

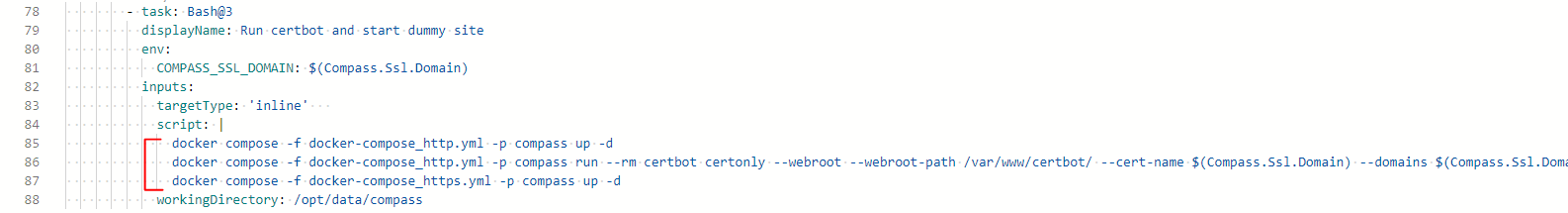
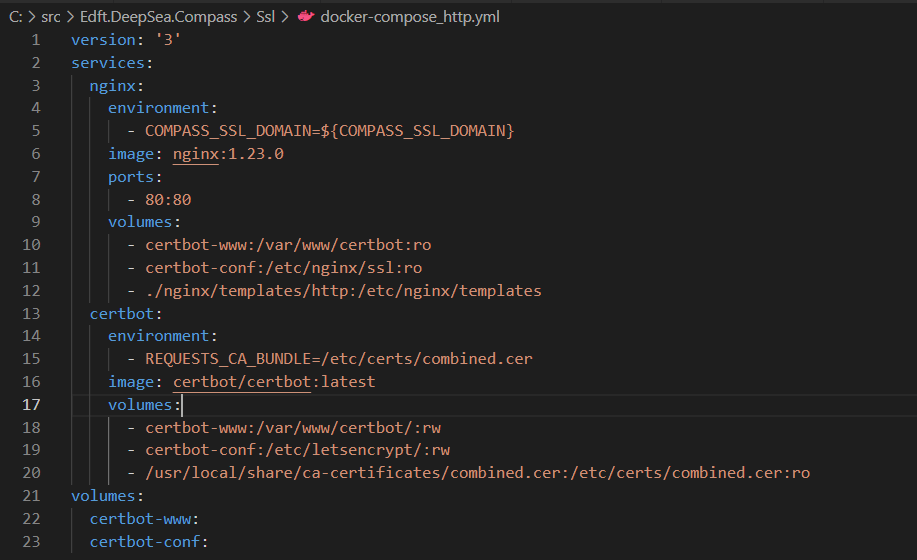


Fig - 2

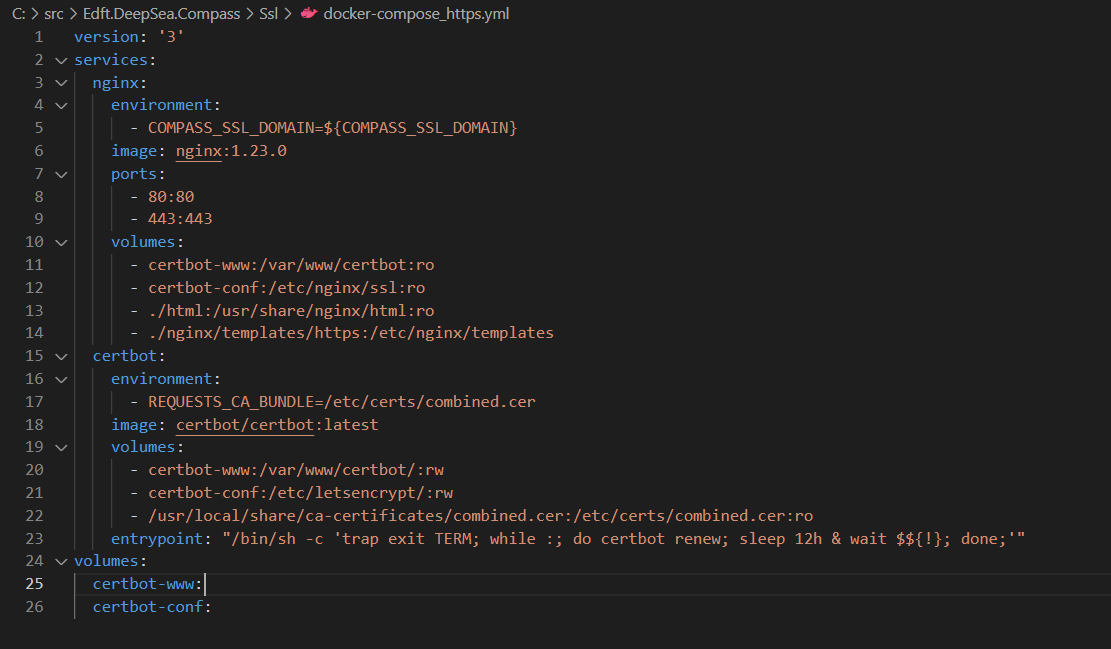
Line 85 uses docker compose to run the certbot and dummy site container.



The compose yml file defines 2 services:

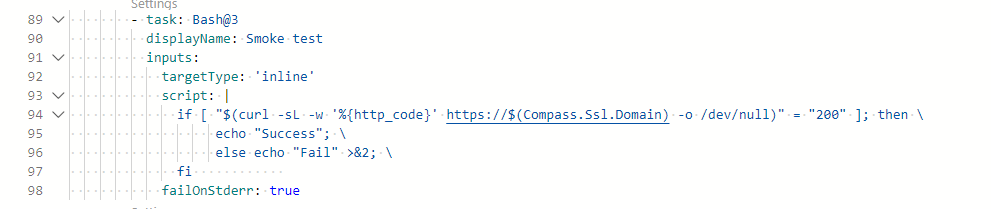
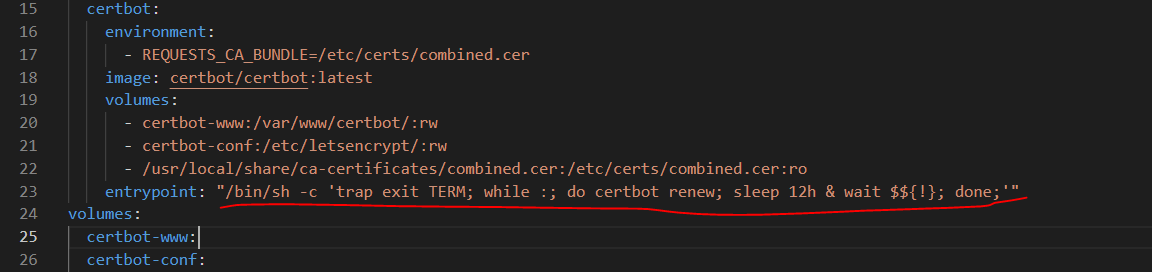
* nginx (the dummy web service) so that it can respond to letsencrypt challenge call back.
* Certbot - to obtain SSL certificate from letsencrypt server

Line 86 runs the certbot command to generate and obtain the SSL certificate. Store it in the shared valume – so that the nginx web server can use it.

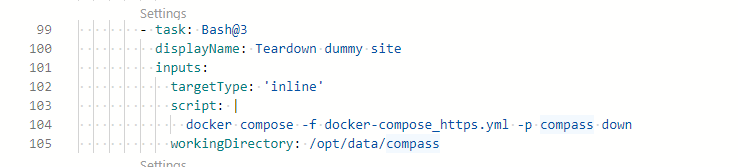
Line 87 in fig-2, starts another set of nginx and certbot containers specified in docker-compose\_https.yml: 

Given there’s the entrypoint specified, docker compose up will automatically run this command therefore no compose run step needed.

Purpose of these 2 servers from \_http:yml:

* Run smoke test check https working
* Run shell command to renew with certbot every 12 hours

3. Tear down

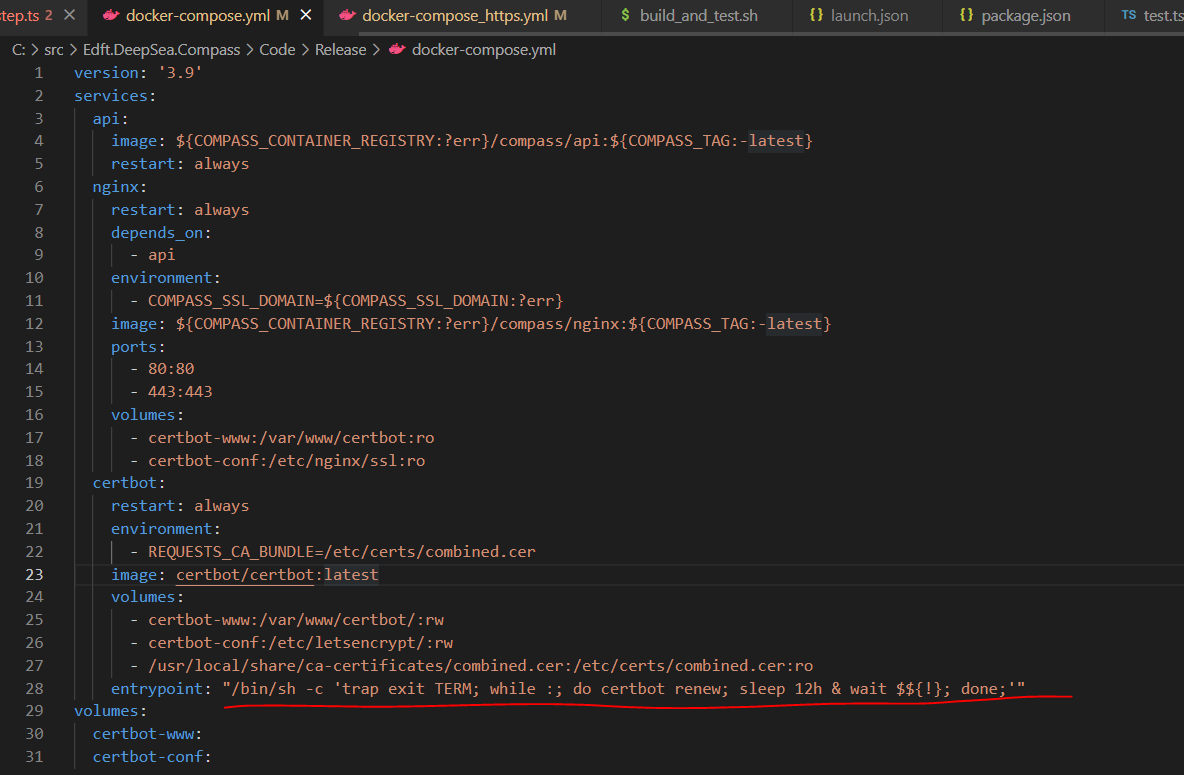


Why only stop containers from \_https.yml? Why not the \_http.yml containers?

If the certbot container from \_https.yml is torn down, wouldn’t it stop the 12-hourly certbot renew command?

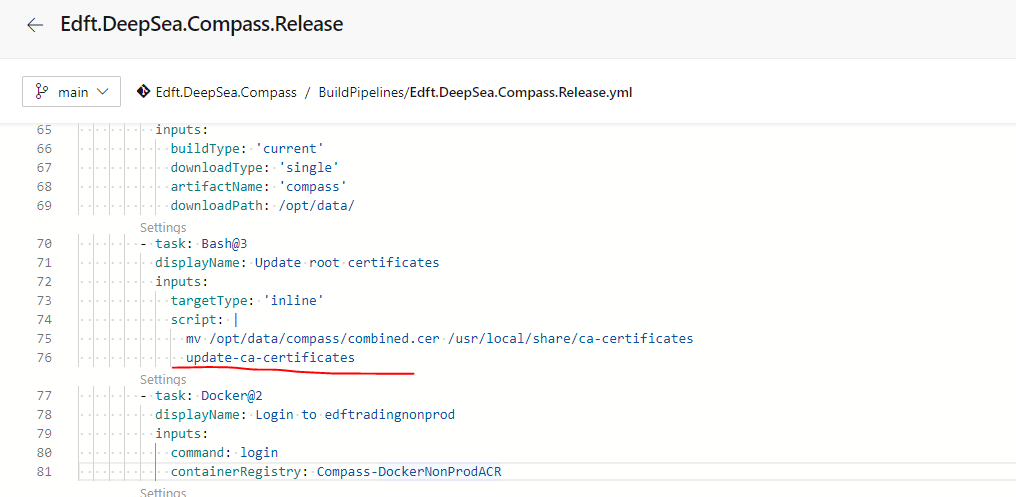
Yes it would. This command here is unnecessary. Stephen to remove this command from https.yml.

In Release pipeline, the yml file starts the nginx server as well as the certbot server. The certbot will be running at the background and run this 12 hourly

What we need is to have a certbot running at the background. This will be done as part of the release pipeline. Below is the compose yml used by the release pipeline. 

The relaese pipeline starts up both the nginx web server as well as the certbot server. The certbot server will kep running and renew the certificate every 12 hours.

4. in Release pipeline



Install EDFT’s root CA certificate

Is it because the SSL cert created by certbot was issued by EDFT’s letsencrypt server, therefore we need to trust the CA cert for it to work? Yes exactly.