Nginx-Rest Proxy SSL Cert Setup

Runbook for the certificates between nginx and the backend servers: each backend renews its own Let's Encrypt certificate and nginx verifies a shared pool name, so rotating certificates needs no nginx changes.

- Overview
- Full How to Guide

Overview

Overview

What this replaces

Previously, backend Spring Boot servers used a long-lived self-signed certificate. nginx pinned that exact cert file via `proxy_ssl_trusted_certificate` and used `proxy_ssl_name` to match its CN. Every cert rotation required touching nginx on every front-end machine, so rotation effectively never happened.

What this system does

Each backend machine issues and renews its own Let's Encrypt certificates using certbot (DNS-01 challenges via Route53), running entirely in Docker alongside the Spring app. Certificates hot-reload into the running JVM with no restart. nginx trusts the public CA root and verifies a shared "pool alias" name, so **nginx configuration never changes again** — adding backends, rotating certs, and promoting environments require zero nginx cert changes.

Full How to Guide

The steps taken to get this setup are outlined below, note that I tested this on a real beta.web.tasking.rpp.cybercrucible.com beta domain and environment, the examples included are for this environment but replace with the actual values for wherever the new setup is. I attached a full write up from claude on the process that goes into more details if desired, this should be in the sidebar on this page to download.

1. Create new AWS IAM user and save the credentials. Assign this user the already created policy called "acme-dns01-cybercrucible", this is a tightly scoped policy used for this purpose only. Making a new user per machine would be a good practice
2. Make sure the domains you wants are created in AWS Route53, no record gets made for the alias used in certs such as "upstream.beta.web.tasking.rpp.cybercrucible.com"
3. Springboot application.properties update. Get rid of all the "server.ssl.key-store..." values, the new docker setup has environment variables for SPRING_SSL_BUNDLE_PEM_ACME_KEYSTORE_CERTIFICATE and related fields. Deploy to the machine
4. SSH to the machine where the upstreams are, this would be azul for rest-web for example, and cd to the docker directory for the upstream. Make new files listed here, this is the sample layout for rest-web upstream in the docker dir:
    1. compose.yml
    2. certbot-deploy-hook.sh
    3. .env
    4. aws.env
    5. rest-web.war
    6. start.sh
    7. rest-mongo-keystore.jks
5. Edit the .env file, this is for cert variable names:
    1. CERT_NAME=rest-web-beta
       PRIMARY_DOMAIN=river.beaver.beta.web.tasking.rpp.cybercrucible.com
       ALIAS_DOMAIN=upstream.beta.web.tasking.rpp.cybercrucible.com
       EXTRA_DOMAINS=-d green.fox.beta.web.tasking.rpp.cybercrucible.com
       -Note that if this docker dir has multiple upstreams then list each one here with '-d' before the domain like shown here, if only 1 upstream then this can be blank and just list the 1 under PRIMARY_DOMAIN
       ACME_EMAIL=support@cybercrucible.com
6. Edit the aws.env file, also run sudo chmod 600 aws.env
   AWS_ACCESS_KEY_ID=AKIA...
   AWS_SECRET_ACCESS_KEY=...
   AWS_DEFAULT_REGION=us-west-2
   AWS_USE_DUALSTACK_ENDPOINT=true #if the host machine only has IPV6 then leave this line here, if it has ipv4 you can uncomment, this is if the host only has ipv6 not the docker env that is important

7. Edit the certbot-deploy-hook.sh file, also run chmod +x on the file
    1. #!/bin/sh
       set -eu
       LIVE="/etc/letsencrypt/live/${CERT_NAME}"
       DEST=/etc/letsencrypt/deployed
       mkdir -p "$DEST"
       cp -L "$LIVE/privkey.pem"   "$DEST/.privkey.tmp"   && chmod 600
       "$DEST/.privkey.tmp"
       cp -L "$LIVE/fullchain.pem" "$DEST/.fullchain.tmp" && chmod 644
       "$DEST/.fullchain.tmp"
       mv -f "$DEST/.privkey.tmp"   "$DEST/privkey.pem"
       mv -f "$DEST/.fullchain.tmp" "$DEST/fullchain.pem"
       echo "$(date -u +%FT%TZ) published ${CERT_NAME} to $DEST" >>
       /etc/letsencrypt/deploy.log
8. Edit compose.yml file, see the attachment section 4.5 for what to put
    1. You should only need to edit the x-restServiceTemplate: &restServiceTemplate and services sections and copy paste the other sections in without change
9. You can test the certbot-init by adding --staging in the certbot-init command section in compose.yml and run it and make sure logs look good. Then remove the --staging and force production reissue with this command:
    1. sudo docker compose run --rm certbot-init \
       "certbot certonly --dns-route53 --cert-name <CERT_NAME> \
       -d <PRIMARY_DOMAIN> -d <ALIAS_DOMAIN> <extra -d flags> \
       --key-type ecdsa --non-interactive --agree-tos -m <ACME_EMAIL> \
       --force-renewal && CERT_NAME=<CERT_NAME> /etc/letsencrypt/renewal-hooks/deploy/publish.sh"
    2. Command I used in beta setup:
        1. sudo docker compose run --rm certbot-init sh -c   "certbot certonly --dns-route53 --cert-name rest-web-beta \
           -d river.beaver.beta.web.tasking.rpp.cybercrucible.com \
           -d upstream.beta.web.tasking.rpp.cybercrucible.com \
           -d green.fox.beta.web.tasking.rpp.cybercrucible.com \
           --key-type ecdsa --non-interactive --agree-tos -m support@cybercrucible.com \
           -force-renewal && CERT_NAME=rest-web-beta /etc/letsencrypt/renewal-hooks/deploy/publish.sh"
10. Make sure the dockers are running
11. Verify with these commands
    1. # Cert content, from inside the container:
       sudo docker exec <containerName> sh -c \
       "openssl x509 -in /certs/deployed/fullchain.pem -noout -issuer -ext subjectAltName"
       # Want: issuer O=Let's Encrypt (no STAGING), all SANs listed.
       # App started:
       sudo docker logs <containerName> | grep -iE "started|error"
12. Updating nginx config. If the main nginx domain is ip split then update on both machines

1. Remove proxy_ssl_trusted_certificate /etc/nginx/ssl/rest-web.crt; and proxy_ssl_name "Unknown"; settings
2. Make sure to add these settings in:
    1. proxy_ssl_verify on;
       proxy_ssl_trusted_certificate /etc/ssl/certs/ca-certificates.crt;
       proxy_ssl_verify_depth 3;
       proxy_ssl_name <ALIAS_DOMAIN>;
       #upstream.beta.web.tasking.rpp.cybercrucible.com; for example
       proxy_ssl_server_name on;
3. Test and apply settings to nginx:
    1. sudo docker exec <nginx-container> nginx -t && sudo docker exec <nginx-container> nginx -s reload

13. Additional testing
    1. echo | openssl s_client -6 -connect [<backend IPv6>]:5050 \
       -servername <ALIAS_DOMAIN> 2>/dev/null \
       | openssl x509 -noout -issuer -dates -ext subjectAltName
        1. # Want: LE issuer, all SANs, ~90-day validity.
    2. Make sure public/health is working: curl https://<public domain>/public/health
    3. Test certbot renew: sudo docker compose exec certbot-renew certbot renew --dry-run