Bild: Albert Hulm

Frachtpapier-fabrik

Docker-Container automatisch mit Zertifikaten versorgen

Selbst komplexe Anwendungen gelangen im Handumdrehen ins Internet, wenn Docker mithelfen darf. Let's-Encrypt-Zertifikate fallen dabei fast vollautomatisch ab und ergänzen nicht nur Web-, sondern auch Mail-Server.

Von Peter Siering

Viele komplexe Anwendungen, etwa der GitHub-Klon GitLab, die aus vielen einzelnen Teilen bestehen, gibt es als Docker-Images. Um solche Dienste ungefährdet produktiv nutzen zu können, sind ordentliche Zertifikate wichtig. Die kann man in solche Anwendungen reinfummeln oder die Aufgabe an zwei weitere Docker-Images delegieren, die sich darauf spezialisiert haben.

Das Docker-Image jwilder/nginx-proxy sammelt Zugriffe auf Web-Anwendungen an den einschlägigen Ports 80 und 443 als Reverse-Proxy ein und leitet sie an den passenden Container weiter. Welcher Container das ist, macht es an Namen fest. Ein Zertifikat zum jeweiligen Namen beschafft das Docker-Image JrCs/docker-letsencrypt-nginx-proxy-companion und stellt es dem Reverse-Proxy bereit. An den Containern mit der Web-Anwendung ist dafür kein Handschlag nötig.

Umgebungsvariablen, die man beim Starten von Web-Anwendungen oder auch anderen Anwendungen aus Docker-Images heraus angibt, verbinden die Helfer miteinander. Der Reverse-Proxy und sein Helfer lauschen am Docker-Socket, erfahren vom Start eines Containers und lesen dessen Umgebungsvariablen aus. Denen entnehmen sie den Host- und Domainnamen sowie weitere zum Beantragen eines Let's-Encrypt-Zertifikats nötige Daten.

Der Ansatz hat den charmanten Vorteil, dass auf einem Host in Docker-Containern mehrere Web-Anwendungen auf diversen Ports laufen können, die der Proxy dann als virtuelle Hosts unter verschiedenen Namen auf den Standard-Ports erreichbar macht. Die Web-Anwendungen müssen oft nicht einmal auf SSL-Betrieb umgestellt werden. Und die ACME-Kommunikation erledigt der Reverse-Proxy transparent für sie.

Voraussetzung für all das ist, dass der Host mit den Containern im Internet erreichbar ist und die Namensauflösung der Container untereinander klappt. Das ist in einem gemeinsamen User Defined Network garantiert (das Default-Netzwerk eines Docker-Hosts bietet keine Namensauflösung). Damit die Daten des Reverse-Proxy und seines Begleiters Updates überleben, sollten sie auf separaten Volumes liegen.

Starthilfe

Folgende yaml-Datei für docker-compose richtet die beiden Hilfscontainer passend ein:

```
version: '2'
networks:
  ext:
volumes:
  certs:
  vhosts:
  html:
```

```
services:
  nginxproxy:
    image: jwilder/nginx-proxy
    container_name: nginxproxy
    hostname: nginxproxy
    restart:
      unless-stopped
    networks:
      - ext
    volumes:
      - certs:/etc/nginx/certs:ro
      - vhosts:/etc/nginx/vhost.d
      - html:/usr/share/nginx/html
      - /var/run/docker.sock:
↳/tmp/docker.sock:ro
    ports:
      - 80:80
      - 443:443
    labels:
      - com.github.jrcs.letsencrypt
↳_nginx_proxy_companion.nginx_proxy
  nginxproxy_comp:
    image: jrcs/letsencrypt-
↳nginx-proxy-companion
    container_name: nginxproxy_comp
    hostname: nginxproxy_comp
    restart:
      unless-stopped
    depends_on:
      - nginxproxy
    networks:
      - ext
    volumes:
      - certs:/etc/nginx/certs
      - /var/run/docker.sock:
↳/var/run/docker.sock:ro
    volumes_from:
      - nginxproxy
```

Die prox.yaml-Datei finden Sie über ct.de/ysgx zum Download ergänzt um einige Kommentare zu hier nicht vertieften Optionen. Legen Sie die Datei in ein Verzeichnis namens „le“ und starten Sie mit `docker-compose -f ./prox.yaml up -d` daraus die beiden Container. Das richtet ein User Defined Network und Volumes mit dem Prefix „le_“ ein. Die Container lauschen dann sowohl auf Port 80 als auch auf Port 443 Ihres Hosts.

Nunmehr können Sie eine Web-Anwendung über ein Container-Image starten, das automatisch Zertifikate erhält. Gesetzt den Fall, der Name www.example.com zeigt auf Ihren Docker-Host, startet der folgende Befehl einen Apache-Webserver mit SSL-Zertifikat:

```
docker run -d --name www 
↳ --network=le_ext -e "LETSENCRYPT_
↳HOST=www.example.com" -e "LETS
↳ENCRYPT_EMAIL=my@example.com" -e 
↳ "VIRTUAL_HOST=www.example.com" httpd
```

Mit `docker logs -f nginxproxy_comp` können Sie die Beschaffung des Zertifikats beobachten. Der Containername und der Hostnamensanteil in der Umgebungsvariable VIRTUAL_HOST sind dabei identisch, nämlich „www“ – das sorgt dafür, dass der nginx-Reverse-Proxy Anfragen an den richtigen Container weiterreichen kann. Welchen Port er anspricht, legt ExposedPort fest.

Tipps & Tricks

Mehr muss man für Spezialfälle wissen: Mancher Web-Anwendung muss man sagen, dass sie hinter einem Reverse-Proxy arbeitet, GitLab etwa lädt andernfalls nicht SSL/TLS-gesicherten Content nach und lässt die gesamte Site als unsicher dastehen. Ein Eintrag in der gitlab.rb-Konfigurationsdatei genügt, um das abzustellen (Proxied SSL, siehe ct.de/ysgx).

Web-Anwendungen erlauben häufig größere Uploads. Hier muss man dem Reverse-Proxy auf die Sprünge helfen. Dazu bietet sich die `client_max_body_size`-Option in der Konfiguration des nginx-Proxys an. Die kann entweder in der globalen Konfigurationsdatei oder in der vhost-spezifischen stehen – im Fall des obigen Beispiels wäre das /etc/nginx/vhost.d/www.example.com (ohne angehängtes .conf).

Von nginxproxy_comp beschaffte Let's-Encrypt-Zertifikate können Sie aus anderen Containern abgreifen oder direkt für Dienste in einem Container heranziehen. Das gelingt, indem Sie das Volume des nginxproxy_comp-Containers mit den Zertifikaten im anderen Container für Lesezugriffe zugänglich machen, etwa mit `-v nginxproxy_certs:/etc/nginx/certs:ro` beim `docker run`-Aufruf.

Auf diese Weise profitiert auch ein Container, der für E-Mail-Empfang- und Versand zuständig ist, recht elegant von Let's-Encrypt-Zertifikaten. Der lässt sich wie jede Web-Anwendung mit den passenden Umgebungsvariablen starten:

```
docker run -d --name mail 
↳ --network=le_ext -e "LETSENCRYPT_
↳HOST=mail.example.com" -e "LETS
↳ENCRYPT_EMAIL=my@example.com" -e 
↳ "VIRTUAL_HOST=mail.example.com" 
↳ -v le_nginxproxy_certs:
↳/etc/nginx/certs:ro -p 25:25 -p 587
↳:587 -e maildomain=mail.example.com 
↳ -e smtp_user=joe:pass12 
↳ catatnight/postfix
```

Dass darin gar kein Webserver läuft, stört nicht, weil ja der Reverse-Proxy und sein Kompagnon gemeinschaftlich die Aushandlung mit Let's Encrypt für das Zertifikat erledigen. Über das als Read-Only-Volume hereingereichte Verzeichnis mit den Zertifikaten lässt sich dann der E-Mail-Server in dem Container konfigurieren. Da hier der Reverse-Proxy bei Zugriffen nicht als Vermittler auftreten kann (er beherrscht die E-Mail-Protokolle nicht), muss der Container die für E-Mail zuständigen Ports nach außen freigeben (mit der Option `-e`).

Bauen Sie das Beispiel auf keinen Fall 1:1 nach, es unterschlägt allerhand, was für produktiven Betrieb tunlichst zu beachten ist: Die Konfigurationsdaten des Containers sollten auf einem separaten Volume liegen, damit die Image-Updates überleben. Ein in Eigenregie betriebener Mail-Server wird schnell Ziel von Trittbrettfahrern, die ihn für den Spam-Versand einspannen. Um das Risiko zu minimieren, sollte er nur Nachrichten für bekannte Adressen akzeptieren und E-Mail nur nach vorheriger Authentifizierung zum Versand annehmen.

Für das Einbinden der Let's-Encrypt-Zertifikate in den E-Mail-Server oder einen anderen per SSL/TLS zu schützenden Dienst, wie LDAP, stehen für das obige Beispiel im Verzeichnis /etc/nginx/certs/mail.example.com mehrere Dateien zur Auswahl: Für E-Mail-Server empfehlen sich stets key.pem sowie fullchain.pem. Bezogen auf den obigen E-Mail-Server genügt es also, in der Datei /etc/postfix/main.cf `smtpd_tls_cert_file` und `smtpd_tls_key_file` auf die genannten Zertifikatsdateien fullchain.pem und key.pem zu verbiegen. Nach einem Neustart von Postfix, in dem Container mit `supervisorctl restart postfix`, benutzt der Mail-Server dann die Zertifikate von Let's Encrypt.

Nicht jeden Wunsch erfüllt das Reverse-Proxy-Container-Gespann. Die offenen und geschlossenen Tickets auf GitHub sprechen dazu Bände: So haben diverse Nutzer Patches vorgeschlagen, um die Anwendung eines Containers als Unterverzeichnis eines Vhosts aktiv werden zu lassen (etwa example.com/mail) – bis heute hat es aber keiner in das nginxproxy-Image geschafft. Mancher Nutzer setzt deshalb auf Alternativen wie etwa Traefik – die bieten in der Regel aber keine Integration von Let's Encrypt. *(ps@ct.de)* c't

Docker-Compose-Datei, Doku-Links:
ct.de/ysgx