Table of contents

1 Caddy로 무료 HTTPS 운영하기

1.1 Caddy

Let's Encrypt Wildcard SSL 적용을 위해 이것 저것 검색해보다 Traefik과 함께 얻어 걸린 녀석이다. 웹서버 자체에 Let's Encrypt SSL 인증서 발급과 갱신에 관련된 것이 포함되어 있어서 정말 편리해 보인다. Caddy 다운로드 페이지에서 사용하고 있는 DNS Provider(godayy 같은)용 플러그인이나 git 플러그인등을 선택해서 binary를 다운로드할 수 있는데, 문제는 다음과 같은 라이센스이다.

> Our build service is free for personal use. A subscription is required to use this download page for internal company use, commercialized distribution, or other business purpose.

좀 이상했다. 분명히 Caddy의 Git repository를 보면 Apache License 2.0인데, 제공되는 바이너리를 상업용으로 사용하려면 Subscription을 구매해야 하는 것이었다. 수익을 위해 조금 특이한 형태의 제약을 둔 거 같다.

조금 더 찾아보니 Caddy docker image을 찾게 되었다. Binary를 소스로 부터 직접 build해서 라이센스 문제를 해결한 것이었다. 게다가 플러그인들도 골라서 설치할 수 있었다.

1.2 Docker와 Docker-compose를 먼저 설치해야 한다.

```
sudo curl -sS https://get.docker.com/ | sh
sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname
↪  -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

sudo usermod -aG docker pointbre해서 현재 사용자를 docker 그룹에 추가하고 나면 docker를 사용할 수 있지만, 그것은 위의 명령을 실행한 터미널에서만 가능하고 다른 터미널에서는 로그아웃 후 다시 로그인하기 전까지는 사용이 되지 않는다.

1.3 Plugin를 지정해서 빌드

Wildcard 인증서(*.abc.com에 대해 인증서를 발급받으면 추후 sub1.abc.com, sub2.abc.com 등의 서브 도메인에 대해서도 추가 인증서 발급없이 사용가능하다는 이유)를 사용하려면 DNS Challenge(인증서 TXT 레코드를 DNS 등록정보에 입력케 해서 인증하는 방식) 방식을 사용해야 한다. 현재 보유한 도메인은 GoDaddy를 통해서 등록했는데, 다행히도 지원이 된다.

1.4 빌드 실패

godaddy 플러그인을 추가해서 빌드하기 위해 docker build --no-cache -t mycaddy:1.0.0 --build-arg version=1.0.0 --build-arg plugins=godaddy github.com/abiosoft/caddy-docker.git 하니 다음처럼 실패했다.

```
go: finding golang.org/x/net v0.0.0-20180724234803-3673e40ba225
go: error loading module requirements
error at 'building caddy'
The command '/bin/sh -c VERSION=${version} PLUGINS=${plugins} /bin/sh /usr/bin/builder.sh' returned a non-zero code:
↪  1
```

문의해보니, DNS Provider를 빌드하는 방식이 Go Module에 의해 아직은 지원되지 않기 때문이라고 했다. 이런 이유로, 플러그인을 추가해서 빌드하려면, 현 시점에서는 바로 직전 버전인 0.11.5를 사용해야 한다.

docker build --no-cache -t mycaddy:1.0.0 --build-arg version=0.11.5 --build-arg plugins=godaddy github.com/abiosoft/caddy-docker.git 하니 아무 문제없이 잘 되었다.

1.5 Timezone 설정

그런데 위의 방법대로 해서 사용을 하게 되면, Caddy 실행중 사용되는 timezone이 문제가 된다. Alpine 리눅스를 이용해서 빌드되는 이미지인데 timezone 설정이 안되어 있기 때문이다. 이를 해결하려면 Docker 이미지 빌드할 때 apk add tzdata를 실행하고, 실행시 TZ=Pacific/Auckland 환경변수를 줘야 한다.

```
cd ~
mkdir caddy
cd caddy
git clone https://github.com/abiosoft/caddy-docker.git
cd caddy-docker
vi Dockerfile
```

다음처럼 RUN apk add tzdata를 중간에 끼워 넣어준다.

```
...
# install process wrapper
COPY --from=builder /go/bin/parent /bin/parent

# Add timezone
RUN apk add tzdata

ENTRYPOINT ["/bin/parent", "caddy"]
...
```

```
docker build --no-cache -t mycaddy:1.0.0 --build-arg version=0.11.5 --build-arg plugins=godaddy .
```

나중에는 Dockerfile 변경할 필요없이 쓰기 위해 이 이슈를 보고해뒀다.

1.6 Wildcard 인증서

이 글을 보면 Caddyfile안에 다음처럼 설정하면 된다는 걸 알 수 있다.

```
*.example.com
tls {
   dns provider
}
```

하지만 추후 변경의 여지가 있으므로 다음처럼 공통 설정을 잡고, 서브 도메인마다 import하는 식으로 하는 게 좋을 거 같다.

```
(wildcard_cert) {
   tls {
      dns provider
      wildcard
   }
}
sub1.example.com {
   import wildcard_cert
   ...
}
.
.
.
sub2.example.com {
   import wildcard_cert
   ...
}
```

1.7 GoDaddy API key & secret

Caddy의 GoDaddy 플러그인 사용을 위해서는 GODADDY_API_KEY과 GODADDY_API_SECRET을 설정해줘야 한다.

먼저 GoDaddy가서 API key & secret을 생성해야 한다. 로그인 후에는 Create New API Key 버튼을 클릭한 후 Name에는 아무거나 입력하고, Production을 선택한 후 Next를 누르면 Key와 Secret이 생성된다. 생성된 Key & Secret을 잘 보관해두자.

1.8 Caddyfile

```
cd ~
mkdir caddy
cd caddy
mkdir .caddy
mkdir log
```

이제 Caddy용 설정파일을 생성하자. vi Caddyfile한 후 다음을 붙여넣는다.

```
(common_config) {
  gzip
  log /root/log/access.log
  errors /root/log/error.log
```

```
  tls {
    dns godaddy
    wildcard
  }
}

comment.madforfamily.com {
  proxy / comment.madforfamily.com:8080 {
    transparent
  }
  import common_config
}
```

나중에 서브 도메인 추가를 쉽게 하기 위해 공통 configuration에 tls 설정을 넣고, 사이트별 설정에 그 설정을 import하는 식으로 했다.

1.9 docker-compose.yml

손쉽게 Caddy를 시작/종료하기 위해 이제는 docker-compose.yml을 생성하자. vi docker-compose.yml한 후 다음을 붙여넣는다. 한 가지 여기서 주의할 점은, 반듯 ACME_AGREE: "true"여야 한다는 점이다. ACME_AGREE: true 처럼 double quote이 없으면 에러가 발생한다.

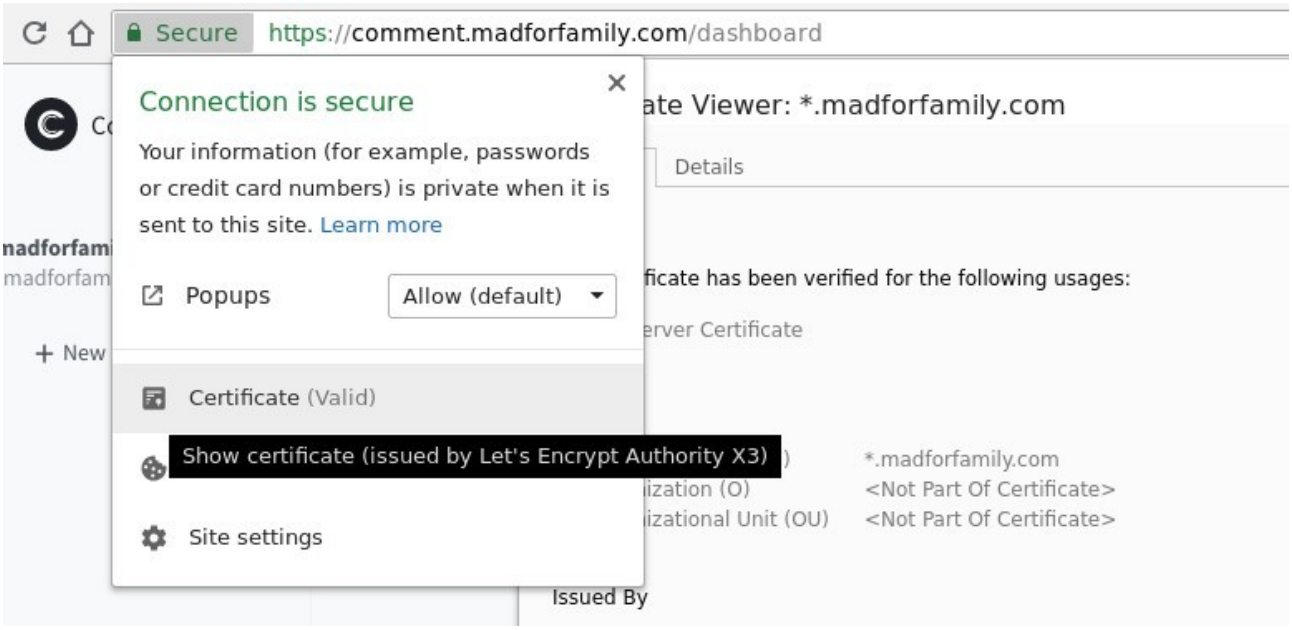
```
version: '3'

services:
  server:
    image: mycaddy:1.0.0
    ports:
      - 80:80
      - 443:443
    environment:
      ACME_AGREE: "true"
      TZ: Pacific/Auckland
      GODADDY_API_KEY: GoDaddy_API_Key_here
      GODADDY_API_SECRET: GoDaddy_API_Secret_here
    volumes:
      - /home/pointbre/caddy/Caddyfile:/etc/Caddyfile
      - /home/pointbre/caddy/.caddy:/root/.caddy
      - /home/pointbre/caddy/log/:/root/log/
```

1.10 Test

터미널 두개를 띄워서 확인해보자. 터미널 1에서는 nc -l 8080해서 comment.madforfamily.com:8080으로의 접속이 들어오는지 체크해보자. 터미널 2에서는 cd ~/caddy; docker-compose up해서 Caddy를 실행해두자.

이제 웹브라우저로 http://comment.madforfamily.com이나 https://comment.madforfamily.com에 접속했을 때 nc를 실행중인 터미널에 HTTP 요청이 들어오면 설정이 제대로 된 것이다. 정말 편리하게도, Caddy는 HTTP로 접속시 자동으로 HTTPS로 리다이렉션시켜주는 걸 알 수 있다.

브라우저로 접속 후 인증서를 확인해보면 다음처럼 *.madforfamily.com에 대해 발급이 된 걸 알 수 있다.



1.11 Service로 caddy를 부팅시 자동 실행

systemd에 등록하면 된다. 먼저 sudo vim /etc/systemd/system/caddy.service 한 후 다음을 붙여 넣는다.

```
[Unit]
Description=Caddy service
After=docker.service
Wants=network-online.target docker.socket
Requires=docker.socket

[Service]
Restart=always
User=pointbre
Group=docker
# Shutdown container (if running) when unit is stopped
ExecStartPre=/usr/local/bin/docker-compose -f /home/pointbre/caddy/docker-compose.yml down
# Start container when unit is started
ExecStart=/usr/local/bin/docker-compose -f /home/pointbre/caddy/docker-compose.yml up
# Stop container when unit is stopped
ExecStop=/usr/local/bin/docker-compose -f /home/pointbre/caddy/docker-compose.yml down

[Install]
WantedBy=multi-user.target
```

이제 systemd에 등록하고 시작해보자.

```
sudo systemctl daemon-reload
sudo systemctl enable caddy
sudo systemctl start caddy
```

docker ps해서 caddy가 실행중인지 확인해보면 끝이다.

1.12 생성된 인증서

~/caddy/.caddy/acme/acme-v02.api.letsencrypt.org/sites/를 보면 wildcard_.madforfamily.com라는 디렉토리가 생성되어 있고, 그 안에 wildcard_.madforfamily.com.crt와 wildcard_.madforfamily.com.key이라는 2개의 파일을 볼 수 있다.