


Let's Encrypt를 사용하여 Nginx에서 HTTPS를 설정하고 이를 Spring Web 애플리케이션과 연동하기 위해 단일 Docker 컨테이너에서 구현하는 절차를 상세히 안내하겠습니다.

전체 프로세스

1. 디렉토리 구조 준비
2. Dockerfile 작성
3. Nginx 설정 파일 작성
4. Spring Boot 애플리케이션 준비
5. Certbot을 사용하여 Let's Encrypt 인증서 발급
6. Docker 이미지 빌드 및 실행
7. HTTPS 테스트

1. 디렉토리 구조 준비


plaintext

 코드 복사

```
project/ |— Dockerfile |— nginx/ | |— nginx.conf | |— certbot.sh # 인증서 발급 스크립트 |— spring/ |   |— spring-web.jar
```

2. Dockerfile 작성


dockerfile

 코드 복사

```
# Base image FROM openjdk:21-jdk-slim # Install dependencies (Nginx, Certbot) RUN apt-get update && apt-get install -y \ nginx \ certbot \ python3-certbot-nginx && \ rm -rf /var/lib/apt/lists/* # Copy Nginx configuration COPY nginx/nginx.conf /etc/nginx/nginx.conf # Copy Spring application COPY spring/spring-web.jar /app/spring-web.jar # Expose ports (80 for HTTP, 443 for HTTPS) EXPOSE 80 443 # Copy Certbot script COPY nginx/certbot.sh /certbot.sh RUN chmod +x /certbot.sh # Start Nginx and Spring application CMD ["sh", "-c", "\ service nginx start && \ /certbot.sh && \ java -jar /app/spring-web.jar"]
```

3. Nginx 설정 파일 (nginx/nginx.conf)


nginx

 코드 복사

```
server { listen 80; server_name yourdomain.com www.yourdomain.com; location /.well-known/acme-challenge/ { root /var/www/certbot; } location / { return 301 https://$host$request_uri; } } server { listen 443 ssl; server_name yourdomain.com www.yourdomain.com; ssl_certificate /etc/letsencrypt/live/yourdomain.com/fullchain.pem; ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem; ssl_protocols TLSv1.2 TLSv1.3; ssl_prefer_server_ciphers on; location / { proxy_pass http://localhost:8080; proxy_set_header Host $host; proxy_set_header X-Real-IP $remote_addr; proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for; } }
```

4. Certbot 스크립트 (nginx/certbot.sh)

bash

 코드 복사

```
#!/bin/bash # Ensure the directory for certbot exists mkdir -p /var/www/certbot # Obtain Let's Encrypt certificate certbot certonly --webroot -w /var/www/certbot \ --email your-email@example.com --agree-tos --no-eff-email \ -d yourdomain.com -d www.yourdomain.com # Reload Nginx to apply the certificates service nginx reload
```

5. Spring Boot 애플리케이션 준비

application.properties 에 Spring Boot가 8080 포트에서 실행되도록 설정합니다.

properties


 코드 복사

```
server.port=8080
```

6. Docker 이미지 빌드 및 실행

Docker 이미지 빌드

```
bash
```

 코드 복사


```
docker build -t nginx-spring-https .
```

도메인 네임과 DNS 설정

- 도메인 이름(`yourdomain.com`)을 구입합니다.
- 도메인을 컨테이너가 실행되는 서버의 공인 IP로 연결합니다.

Docker 컨테이너 실행

```
bash
```

 코드 복사

```
docker run -p 80:80 -p 443:443 nginx-spring-https
```

7. HTTPS 테스트


1. 브라우저에서 `https://yourdomain.com` 으로 접속하여 인증서를 확인합니다.
2. Spring 애플리케이션의 엔드포인트가 제대로 동작하는지 확인합니다.

8. 인증서 자동 갱신 설정

Let's Encrypt 인증서는 기본적으로 90일 동안 유효합니다. 이를 자동으로 갱신하려면 `certbot renew` 를 cron job에 추가합니다. 컨테이너에서는 다음과 같이 처리할 수 있습니다:

갱신 스크립트 추가 (`nginx/renew.sh`)

```
bash
```

 코드 복사

```
#!/bin/bash certbot renew service nginx reload
```

Dockerfile에 Cron 설치 및 설정 추가

dockerfile

 코드 복사

```
# Install Cron RUN apt-get install -y cron # Add cron job for certificate renewal COPY
nginx/renew.sh /renew.sh RUN chmod +x /renew.sh && \ echo "0 0 * * 0 /renew.sh" >
/etc/cron.d/certbot-renew
```

이렇게 설정하면 Docker 컨테이너 내부에서 Nginx와 Spring 애플리케이션이 HTTPS로 연동되고, 인증서 갱신도 자동화됩니다