

프로젝트 결과보고서

- 프로젝트 명 : APM을 통한 WordPress 구현
- 보고자 : 전 상 언
- 진행 기간 : 2023.11.09 ~ 2023.11.11 (3일)



목차

- 1 프로젝트 개요
 - 프로젝트 배경 및 목표
 - 프로젝트 서버 구축 개념도
- 2 프로젝트 환경
- 3 프로젝트 진행
 - DB 서버 구축
 - Web 서버 구축
 - DNS 서버 구축
 - HTTPS 설정
 - Client
- 4 결과
- 5 소감

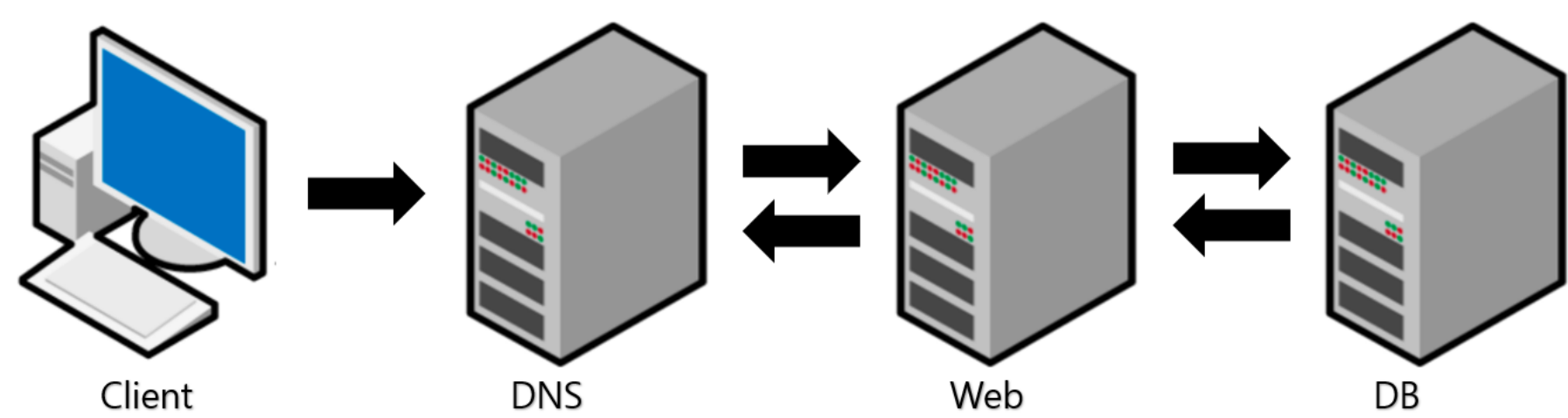
1 프로젝트 개요

프로젝트 배경 및 목표

예전에는 전통적인 온-프레미스(On-premise) 방식으로 네트워크 및 하드웨어 관리가 주로 인프라 엔지니어의 역할이었습니다. 그러나 최근에는 클라우드 환경에서 가상의 서버를 다루는 것이 일상화되어, 개발자도 배포와 테스트를 직접 수행할 수 있게 되었습니다. 이로 인해 인프라에 대한 이해와 습득이 더욱 필수적으로 떠올랐습니다.

이러한 변화에 발맞춰 기초적인 인프라 지식을 습득하고, 실제 인프라 구축 프로젝트를 진행함으로써 실무 능력을 향상 시키는 것이 이 프로젝트의 목표입니다.

프로젝트 서버 구축 개념도



클라이언트는 DNS를 통해 웹사이트를 IP주소 없이 Domain Name으로 접근할 수 있다.

웹사이트는 DB에 있는 내용을 끌어와서 호스팅합니다.

2 프로젝트 환경



구현 환경

Web 서버

- Domain : wp.test.example.com
- IP : 192.168.56.100

DB 서버

- Domain : db.test.example.com
- IP : 192.168.56.150

DNS 서버

- Domain : dns.test.example.com
- IP : 192.168.56.200

VirtualBox	6.1.46 r158378 (Qt5.6.2)
Rocky	Rocky-9.1-x86_64-dvd
MariaDB	10.5.22
Apache	2.4.53
Php	8.1.14
Wordpress	6.4.1

3 프로젝트 진행

일정	2023-11-09	2023-11-10	2023-11-11
DB 서버 구축	O		
Web 서버 구축	O		
DNS 서버 구축	O	O	
HTTPS 설정		O	
보고서 작성		O	O

DB 서버 구축

DB 서버로 사용할 가상머신에서 진행합니다.

• IP 부여하기

```
# nmcli con add con-name staticdb type ethernet ifname eth1 ip4 192.168.56.150/24 gw4 192.168.56.1
# nmcli con up staticdb
# nmcli con show

===
NAME                UUID                                  TYPE      DEVICE
eth0                 5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
staticdb             86a664b8-1cd7-4504-b477-7cf45395e6b4  ethernet  eth1    < 변경 확인
enp0s3               2c12d263-f3e2-440c-8b4e-6ef4e60716ae  ethernet  --
System eth1          9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  --
===
```

• MariaDB 설치 및 실행

```
# dnf -y install mariadb-server
- DB 서버를 설치합니다.

# systemctl enable mariadb.service --now
- DB 서비스를 부팅 시 자동으로 시작하도록 설정하고, 동시에 즉시 서비스를 시작합니다.
```

• 방화벽 실행 및 서비스 추가

```
# systemctl start firewalld.service
- firewalld 방화벽을 시작합니다.

# firewall-cmd --add-service=mysql --permanent
- 방화벽에 mysql 서비스를 추가하여 포트를 개방합니다. '--permanent' 옵션은 영구적으로 적용해주는 옵션입니다.
```

• MySql 설정

```
# mysql_secure_installation
- 데이터베이스 서버의 보안 강화를 위한 초기 설정을 수행하는 명령어 입니다.

===

Enter current password for root (enter for none):
- 현재 root 계정의 비밀번호를 입력, 초기값은 설정되지 않았기 때문에 Enter를 치면 됩니다.

Switch to unix_socket authentication [Y/n] y
- 유닉스 소켓이라는 인증 방식으로 전환할 것인지 물어보는 질문

Change the root password? [Y/n] y
New password:
- root 비밀번호를 변경할 건지 물어보는 질문 y를 선택한다면 비밀번호 설정을 할 수 있습니다.

Remove anonymous users? [Y/n] y
- 익명 사용자를 제거할 것인지 묻는 질문. y를 선택하면 로그인시 root로 로그인해야 합니다.
- n을 선택한다면 익명 사용자로 권한을 갖게 되므로 보안상 지워주는 것이 좋습니다.

Disallow root login remotely? [Y/n] n
- localhost의 ip가 아닌 곳에서 root로 로그인이 가능하게 할 것인지 질문하는 것
- Y를 선택하면 원격 로그인이 안됩니다.
- test 용도이며 장소를 옮긴다면 n을 권장, 로컬에서만 사용한다면 y를 선택하는 것이 좋습니다.

Remove test database and access to it? [Y/n] n
- test 데이터베이스를 제거할 것인지에 대한 질문으로 현재 실습에선 쓸 일이 없어서 제거해줍니다.

Reload privilege tables now? [Y/n] y
- 권한 변경 내용을 반영할 것인지 묻는 질문입니다.

===
```

• DB 생성하기

```
# mysql -u root -p
- MySQL 접속 'Enter password:'과 같은 대화가 나온다면 root 비밀번호를 입력하고 로그인을 합니다.

-- 주의사항 이번 SQL에서는 autocommit을 끄지 않고 진행합니다. --

MariaDB [(none)]> SHOW VARIABLES LIKE 'auto%';
- 해당 명령어를 입력하여 autocommit의 Value가 ON인지 확인하고 진행합니다.

===
+-----+-----+
| Variable_name | Value |
+-----+-----+
| auto_increment_increment | 1 |
| auto_increment_offset | 1 |
| autocommit | ON | < ON 상태 체크
| automatic_sp_privileges | ON |
+-----+-----+
===

-- DB 생성하기 --

> CREATE DATABASE wpdb1;
- 'wpdb1'라는 이름의 새로운 데이터 베이스를 생성합니다.
```

```
> CREATE USER wpdbuser1@'%' IDENTIFIED BY '1';
- 'wpdbuser1'라는 사용자를 생성하고 어떤 호스트에서 접근할 수 있도록 @'%'로 하여 권한을 부여합니다.
- IDENTIFIED BY로 비밀번호는 '1'로 부여하였습니다.
```

```
> GRANT ALL ON wpdb1.* TO wpdbuser1@'%;
- wpdbuser1 유저가 wpdb1 데이터베이스에 모든 권한을 가지도록 부여해줍니다.
```

```
> FLUSH PRIVILEGES;
- 사용자 및 권한 설정에 대한 변경사항을 적용하도록 서버에 명령하는 명령어로 변경사항을 적용시킵니다.
```

```
-- 유저 생성 확인 --
MariaDB [(none)]> USE mysql
- mysql 데이터베이스에 접속합니다.
```

```
MariaDB [mysql]> SELECT user FROM user;
- user테이블에서 생성된 user를 조회합니다.
```

```
===
+-----+
| User           |
+-----+
| wpdbuser1      | < 새로 생성한 유저
| mariadb.sys    |
| mysql          |
| root           |
+-----+
5 rows in set (0.011 sec)
===
```

```
MariaDB [mysql]> SHOW DATABASES;
- 데이터베이스도 잘 생성되었는지 확인합니다.
```

```
===
+-----+
| Database       |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| wpdb1          | < 새로 생성한 데이터베이스
+-----+
5 rows in set (0.000 sec)
===
```

```
MariaDB [mysql]> exit
Bye
```

Web 서버 구축

Web 서버로 사용할 가상머신에서 진행합니다. (DB서버의 가상머신과 다릅니다.)

• Apache와 Php 설치

```
# dnf -y install httpd
- 아파치 패키지를 설치하는 명령어입니다.

# dnf module enable php:8.1
- 이번 프로젝트에서는 php 버전을 8.1 버전으로 지정하여 진행합니다. 위 명령어는 버전을 지정하는 명령어 입니다.
# dnf -y install php
- php 설치 진행

-- php 관련 모듈 설치
dnf install php-gd php-soap php-intl php-mysqlnd php-pdo php-pear php-xml php-xmlrpc php-zip wget

# systemctl enable httpd --now

# systemctl start firewalld
# firewall-cmd --add-service=http --permanent
# firewall-cmd --reload
- 아파치 패키지 설치 후 방화벽 포트 개방
```

• Apache 파일 설정 변경

```
# vim /etc/httpd/conf/httpd.conf
```

```

===
168 <IfModule dir_module>
169     DirectoryIndex index.html index.php      <- [일부 추가]
170 </IfModule>

288     AddType application/x-compress .Z
289     AddType application/x-gzip .gz .tgz
290     AddType application/x-httpd-php .php      <- [라인 추가]
291     AddType application/x-https-phps .phps     <- [라인 추가]
===

```

```

# vim /etc/php-fpm.d/www.conf

===
55 listen.acl_users = apache,nginx
56 listen.acl_groups = apache                <- [라인 추가]
===

```

• PHP 적용 확인

```

# vi /var/www/html/phpinfo.php

===
<?php phpinfo();?>                                <- [라인 추가]
===

```

• 패키지 실행

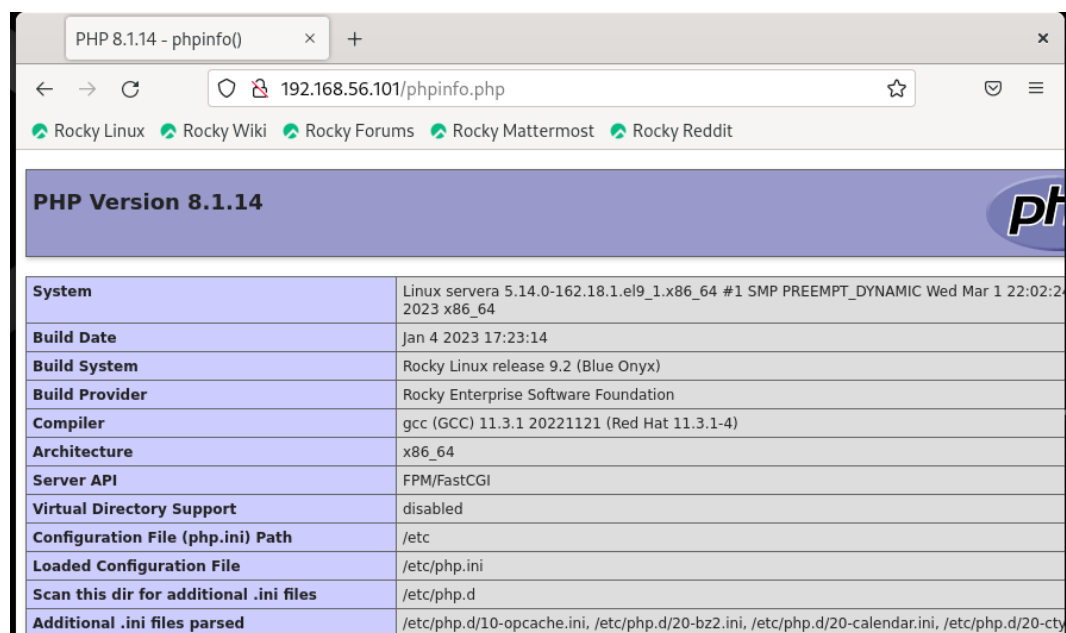
```

# systemctl restart httpd

# systemctl enable php-fpm --now

```

• php 적용 확인 (웹 브라우저에서 확인)



• IP 부여하기

```

# nmcli con add con-name staticweb type ethernet ifname eth1 ip4 192.168.56.100/24 gw4 192.168.56.1
# nmcli con up staticweb
# nmcli con show

===
NAME          UUID                                   TYPE      DEVICE
eth0          5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
staticweb     be23afd6-ddf3-4004-ae5e-42ce0ea6984c  ethernet  eth1    < 변경 확인
enp0s3        2c12d263-f3e2-440c-8b4e-6ef4e60716ae  ethernet  --
System eth1   9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  --
===

```

• Wordpress 설치

```
# wget https://wordpress.org/latest.tar.gz -O wordpress.tar.gz
# tar xf wordpress.tar.gz -C /var/www/html
- /var/www/html/ 안에 해당 tar를 해제합니다.
```

• 권한 부여

```
# chown -R apache:apache /var/www/html/wordpress/
# chmod -R 755 /var/www/html/wordpress/
```

• host 파일 생성

```
# cp /usr/share/doc/httpd-core/httpd-vhosts.conf /etc/httpd/conf.d/wordpress.conf
```

• host파일 편집

```
# vim /etc/httpd/conf.d/wordpress.conf

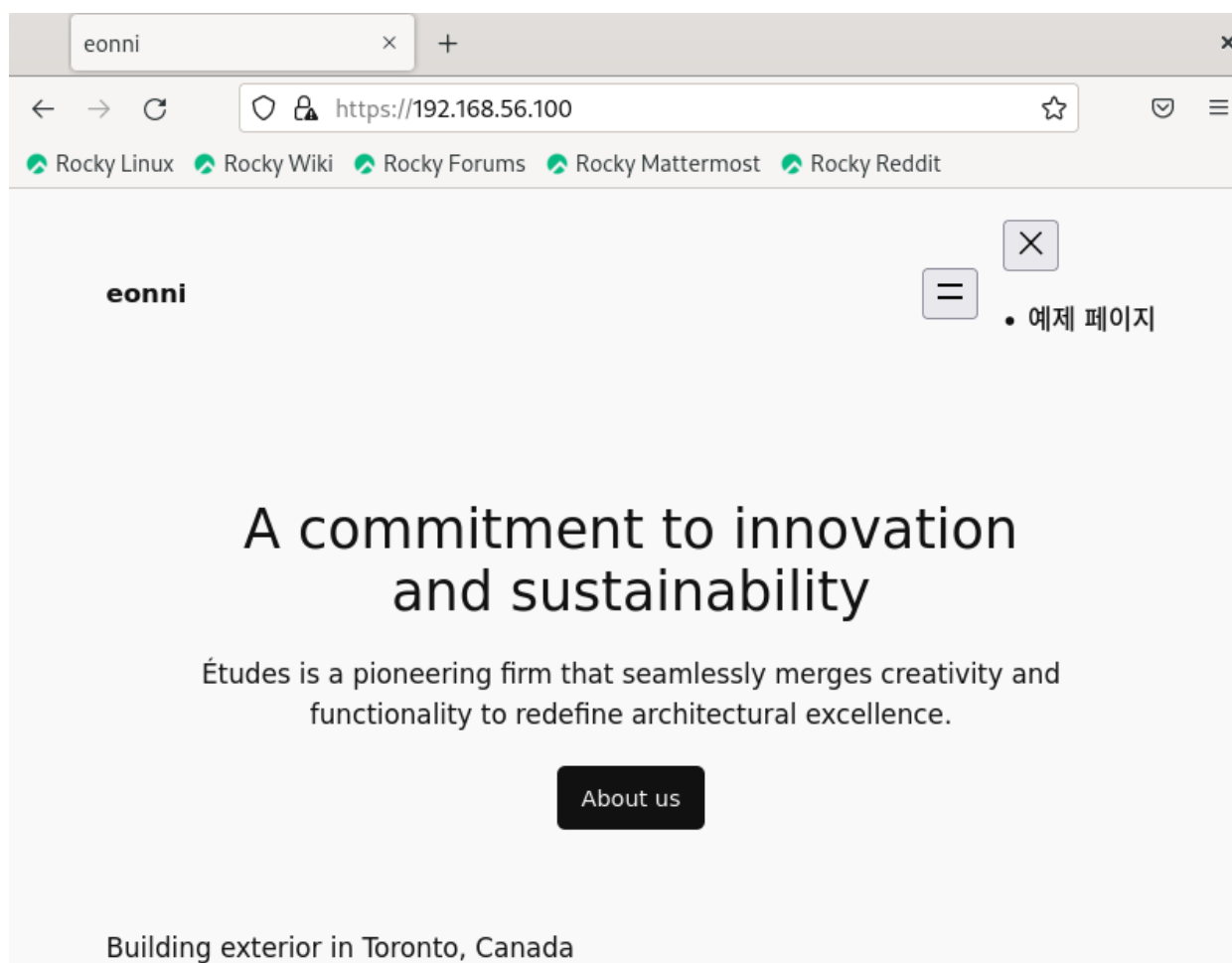
===
<VirtualHost *:80>
    ServerAdmin root@test.example.com          <- [변경]
    DocumentRoot "/var/www/html/wordpress"      <- [변경]
    ServerName wp.test.example.com              <- [변경]
    ServerAlias wp                             <- [변경]
    ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
    CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>

<Directory "/var/www/html/wordpress">          <- [Directory 추가]
Options Indexes FollowSymLinks
AllowOverride all
Require all granted
</Directory>
===
```

• 아파치 재실행

```
# systemctl restart httpd
```

• 접속 테스트 (웹 브라우저에서 테스트합니다.)



DNS 서버 구축

- IP 설정

```
# nmcli con add con-name static1 type ethernet ifname eth1 ip4 192.168.56.200/24 gw4 192.168.56.1
# nmcli con up static1
```

- 네트워크 설정

```
# vim /etc/resolv.conf

===
# Generated by NetworkManager
nameserver 192.168.56.200          <- [라인 추가]
===
```

- 패키지 설치

```
# dnf -y install bind
```

- 패키지 실행

```
# systemctl enable named.service --now
# systemctl start firewalld.service
```

- 정방향 만들기

```
-- 디렉토리 이동
# cd /var/named/

# cp named.empty test.example.com.zone
- 템플릿 복제

# vim test.example.com.zone
- 복제한 템플릿 활용하여 zone파일 생성하기
===

$TTL 3H
@      IN SOA      test.example.com. root.test.example.com. (          <- [변경]
                                0          ; serial
                                1D         ; refresh
                                1H         ; retry
                                1W         ; expire
                                3H )       ; minimum

      NS      dns.test.example.com.          <- [변경]
      A      192.168.56.200                  <- [변경]

wp IN  A      192.168.56.100                  <- [라인 추가]
db IN  A      192.168.56.150                  <- [라인 추가]
dns IN  A      192.168.56.200                  <- [라인 추가]
===

# chown :named test.example.com.zone
- 권한 부여
```

- 역방향 만들기

```
# cp test.example.com.zone 192.168.56.zone
- 정방향 파일 복제

# vim 192.168.56.zone
- zone파일 수정하기
===

$TTL 3H
@      IN SOA      test.example.com. root.test.example.com. (
                                0          ; serial
```

```

                                1D      ; refresh
                                1H      ; retry
                                1W      ; expire
                                3H )    ; minimum

NS      dns.test.example.com.
A      192.168.56.200

100 IN  PTR      wp.test.example.com.      <- [변경]
150 IN  PTR      db.test.example.com.      <- [변경]
200 IN  PTR      dns.test.example.com.      <- [변경]
===

# chown :named 192.168.56.zone
- 권한 부여
```

• 반영하기

```
# systemctl restart named
```

• DNS 방화벽에 추가하기

```
# systemctl start firewalld.service
# firewall-cmd --add-service=dns --permanent
# firewall-cmd --reload
```

• 연결 확인해보기

```
# nslookup wp.test.example.com 192.168.56.200

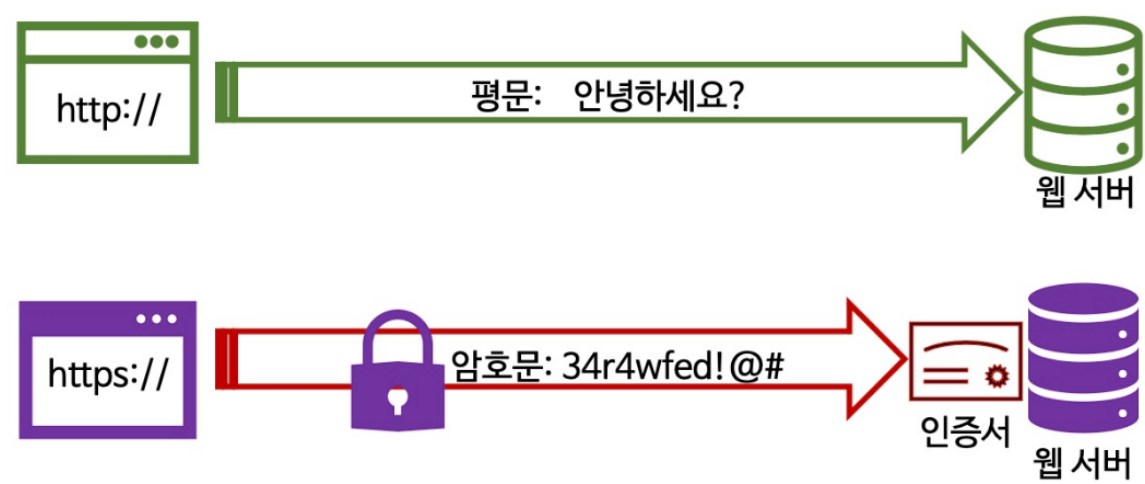
===
Server:      192.168.56.200
Address:     192.168.56.200#53

Non-authoritative answer:
Name: wp.test.exampel.com
Address: 192.64.151.240
===
```

HTTPS 설정

- Web서버 가상머신에서 진행합니다.

HTTPS란?



이미지 출처 : <https://yozm.wishket.com/magazine/detail/1852/>

클라이언트와 서버 간의 모든 커뮤니케이션을 암호화하여 보안을 강화해줍니다.

• 패키지 설치 및 방화벽 추가


```
# dnf -y install mod_ssl
# firewall-cmd --add-service=https --permanent
# firewall-cmd --reload
```


• 패키지 버전 체크

```
# httpd -v
Server version: Apache/2.4.53 (Rocky Linux)
Server built:   Apr 28 2023 00:00:00

# openssl version
OpenSSL 3.0.1 14 Dec 2021 (Library: OpenSSL 3.0.1 14 Dec 2021)
```

확인한 버전을 토대로 아래 사이트에서 Apache 버전에 맞는 내용을 확인합니다.

<https://ssl-config.mozilla.org/>



SSL Configuration Generator

Server Software

☒ Apache
 ☐ AWS ALB
 ☐ AWS ELB
 ☐ Caddy
 ☐ Dovecot
 ☐ Exim
 ☐ Go
 ☐ HAProxy
 ☐ Jetty
 ☐ lighttpd

☐ MySQL
 ☐ nginx
 ☐ Oracle HTTP
 ☐ Postfix
 ☐ PostgreSQL
 ☐ ProFTPD
 ☐ Redis
 ☐ Squid
 ☐ Tomcat
 ☐ Traefik

Mozilla Configuration

☐ Modern
Services with clients that support TLS 1.3 and don't need backward compatibility
 ☒ Intermediate
General-purpose servers with a variety of clients, recommended for almost all systems
 ☐ Old
Compatible with a number of very old clients, and should be used only as a last resort

Environment

Server Version2.4.53

OpenSSL Version3.0.1

Miscellaneous

☒ HTTP Strict Transport Security
This also redirects to HTTPS, if possible
 ☒ OCSP Stapling

apache 2.4.53, intermediate config, OpenSSL 3.0.1

Supports Firefox 27, Android 4.4.2, Chrome 31, Edge, IE 11 on Windows 7, Java 8u31, OpenSSL 1.0.1, Opera 20, and Safari 9

```
# generated 2023-11-10, Mozilla Guideline v5.7, Apache 2.4.53, OpenSSL 3.0.1, intermediate configuration
# https://ssl-config.mozilla.org/#server=apache&version=2.4.53&config=intermediate&openssl=3.0.1&guideline=5.7

# this configuration requires mod_ssl, mod_socache_shmcb, mod_rewrite, and mod_headers
<VirtualHost *:80>
    RewriteEngine On
    RewriteCond %{REQUEST_URI} !^/\..well-known/acme/-challenge/
    RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>

<VirtualHost *:443>
    SSLEngine on

    # curl https://ssl-config.mozilla.org/ffdhe2048.txt >> /path/to/signed_cert_and_intermediate_certs_and_dhparams
    SSLCertificateFile      /path/to/signed_cert_and_intermediate_certs_and_dhparams
    SSLCertificateKeyFile   /path/to/private_key

    # enable HTTP/2, if available
    Protocols h2 http/1.1

    # HTTP Strict Transport Security (mod_headers is required) (63072000 seconds)
    Header always set Strict-Transport-Security "max-age=63072000"
</VirtualHost>

# intermediate configuration
SSLProtocol               all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite             ECDHE - ECDSA - AES128 - GCM - SHA256 : ECDHE - RSA - AES128 - GCM - SHA256 : ECDHE - ECDSA - AES256 - GCM - SHA384 : ECDHE - RSA - AES256 - GCM - SHA384 : ECDHE - ECDSA - CHACHA20 - POLY1305 : ECDHE - RSA - CHACHA20 - POLY1305 : DHE - RSA - AES128 - GCM - SHA256 : DHE - RSA - AES256 - GCM - SHA384 : DHE - RSA - CHACHA20 - POLY1305
SSLHonorCipherOrder       off
SSLSessionTickets         off

SSLUseStapling On
SSLStaplingCache "shmcb:logs/ssl_stapling(32768)"
```

Copy

• 인증서 생성

```
# openssl genrsa -out private.key 2048
- 개인키 생성
# openssl req -new -key private.key -out cert.csr
- CSR 생성
# openssl x509 -req -signkey private.key -in cert.csr -out cert.crt
- CRT 생성
```

파일이동

```
# mv private.key /etc/pki/tls/private/
# mv cert.crt /etc/pki/tls/certs/
```

```
# vim /etc/httpd/conf.d/ssl.conf
- ssl 설정하기

===
43 DocumentRoot "/var/www/html/wordpress"          <- # 제거 후 wordpress 경로 추가
44 ServerName wp.test.example.com:443                <- # 제거 후 hostname 설정
...
59 SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1            <- Mozilla 사이트에서 받은 SSLProtocol로 변경
...
73 SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384
:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-POLY1305
LY1305                                                <- Mozilla 사이트에서 받은 SSLCipherSuite로 변경
...
85 SSLCertificateFile /etc/pki/tls/certs/cert.crt      <- 변경
...
93 SSLCertificateKeyFile /etc/pki/tls/private/private.key <- 변경
===

# systemctl restart httpd
```

Client

제작한 Wordpress 게시판에 접속 테스트를 진행할 Client 가상머신에서 진행합니다.

```
# vim /etc/resolv.conf

===
# Generated by NetworkManager
nameserver 192.168.56.200          <- [추가]
===
```

• 연결 확인

```
# nslookup wp.test.example.com

===
Server:      192.168.56.200
Address:     192.168.56.200#53

Name: wp.test.example.com
Address: 192.168.56.100
===
```

```
# ping 192.168.56.200

===
PING 192.168.56.200 (192.168.56.200) 56(84) bytes of data.
64 bytes from 192.168.56.200: icmp_seq=1 ttl=64 time=0.510 ms
64 bytes from 192.168.56.200: icmp_seq=2 ttl=64 time=0.553 ms
64 bytes from 192.168.56.200: icmp_seq=3 ttl=64 time=0.625 ms
===
```

⚠ Client에서 연결을 진행하였지만, vm에서는 웹사이트가 정상적으로 접근 되지 않았습니다.

- 로컬 환경에서 접근 진행
- 로컬 환경 → 설정 → 네트워크 및 인터넷 → 이더넷 → DNS 서버 할당 → 수동으로 전환 → IPv4 On → 기본 설정 DNS (192.168.56.200) 입력 후 저장

DNS 설정 편집

수동

IPv4

켜

기본 설정 DNS

192.168.56.200

HTTPS를 통한 DNS

끔

대체 DNS

HTTPS를 통한 DNS

끔

IPv6

끔

저장

취소

인증 설정

편집

데이터 통신 연결

일부 앱은 이 네트워크에 연결되어 있을 때 데이터 사용량을 줄이기 위해 다르게 작동할 수 있습니다.

이 네트워크의 데이터 사용량 제어를 위해 데이터 제한 설정

IP 할당:

자동(DHCP)

편집

DNS 서버 할당:

수동

편집

IPv4 DNS 서버:

192.168.56.200(암호화되지 않음)

링크 속도(수신/송신):

1000/1000(Mbps)

링크-로컬 IPv6 주소:

fe80::97ec:714a:3015:a591%2

IPv4 주소:

192.168.0.98

IPv4 DNS 서버:

192.168.56.200(암호화되지 않음)

제조사업체:

Intel

설명:

Intel(R) Ethernet Connection (17) I219-V

드라이버 버전:

12.19.2.45

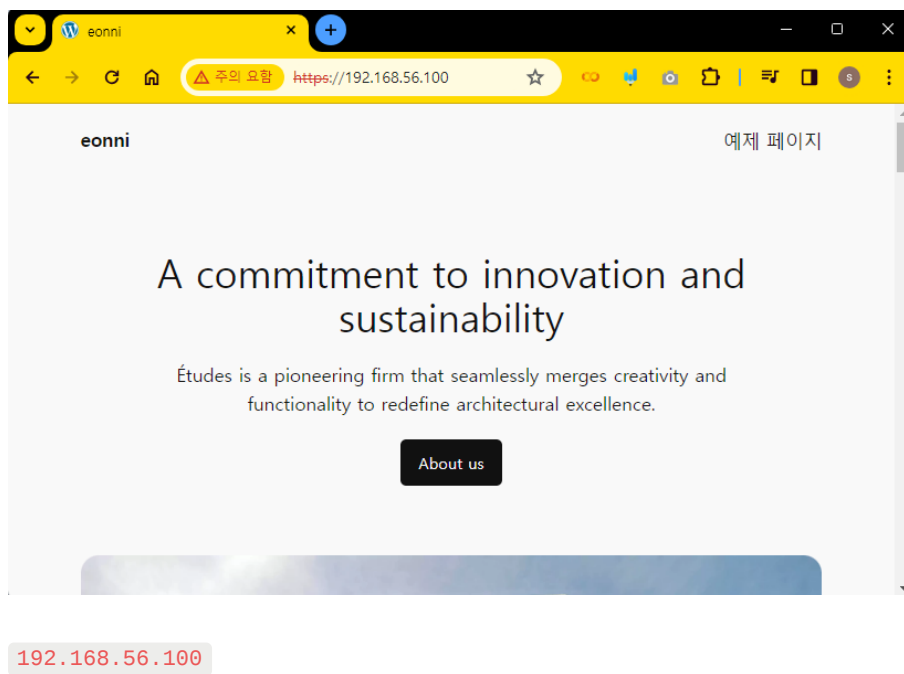
물리적 주소(MAC):

88-AE-DD-26-7A-2B

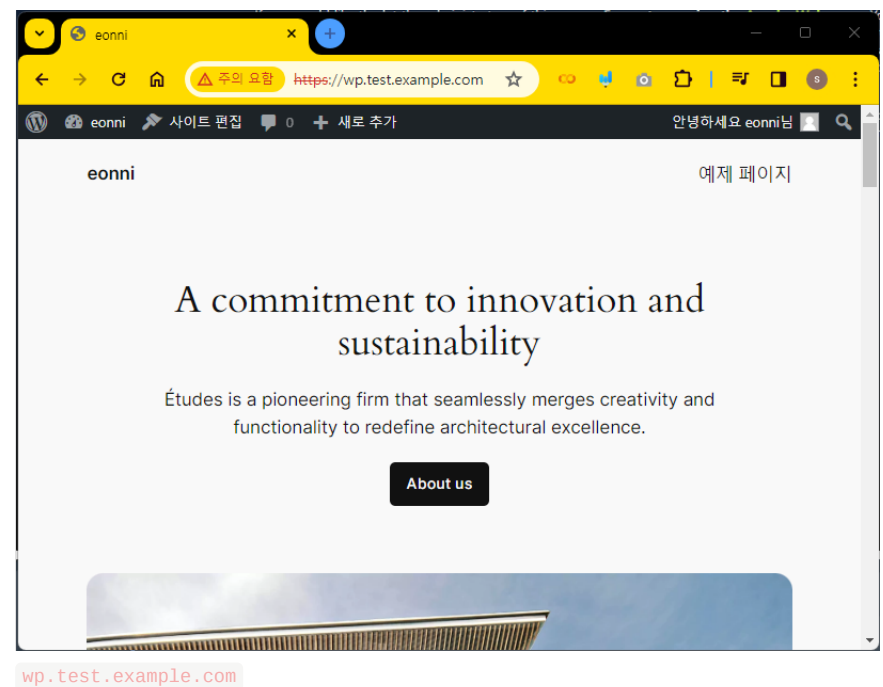
복사

4 결과

IP 주소 접근 결과

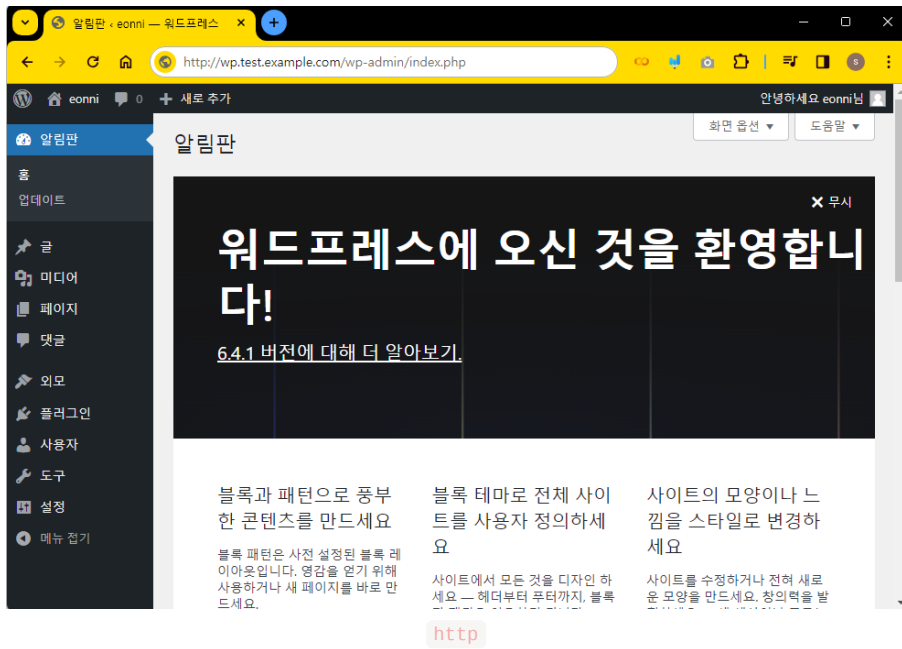


도메인 접근 결과

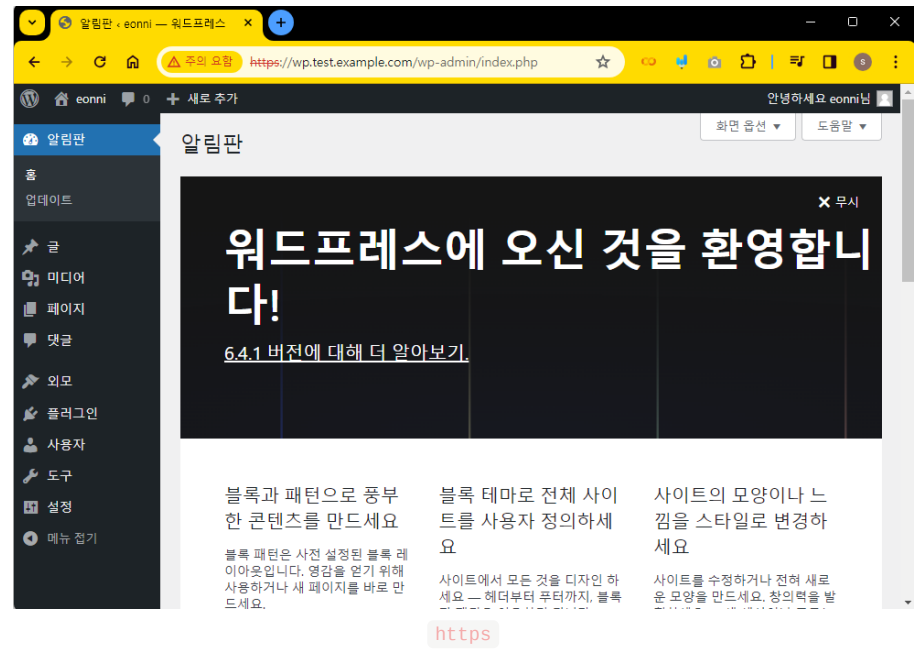


Https 반영 확인

반영 전



반영 후



5 소감

이론을 배우고 실습을 진행할 당시에는 정말로 재밌는 경험이었습니다. 그러나 배운 내용들을 하나로 모아 프로젝트로 진행해보니 어디서부터 시작해야 할지 막막함을 느꼈습니다.

프로젝트를 진행하면서 단편화된 학습 내용들을 조립하는 과정이 쉽지 않았지만, 이런 어려움을 극복하면서 앞으로 학습하는 내용들도 연계시켜서 생각하고 실습하는 습관을 길러야겠다는 각오를 다졌습니다.

프로젝트를 통해 트러블슈팅의 중요성이 많이 느껴졌습니다. 리눅스 환경에서의 경험이 많은 도움이 되었고, 이를 통해 손으로 명령어를 많이 쳐보고 실험해보는 것의 중요성을 깨달았습니다. 평상시에도 더 많은 실습과 경험을 쌓아야겠다는 생각이 들었습니다.