
[SecureWP Infra: Automated WordPress System]

[Infrastructure Hardening with Ansible, Grafana, and Wordfence on Rocky]



CAB TA 4기 -

과정명 오픈소스 기반 클라우드 플랫폼(PaaS) 개발
엔지니어 양성과정

팀 원 이정민

일 자 2025. 04. 11

Table of Contents

1. Project Overview.....	3
2. Technology Stack & Justification	4
3. Environment Setup.....	6
4. Infrastructure Automation with Ansible	9
5. Security Hardening & Firewall Configuration	12
6. SSL Certificate and HTTPS Setup	16
7. Monitoring with Prometheus and Grafana	18
8. Troubleshooting & Real-world Challenges.....	22
9. System Architecture Diagram	23
10. Strengths of the System.....	24
11. Screenshots of Implementation.....	26
12. Troubleshooting Details	30
13. Conclusion & What I Learned.....	33

1. Project Overview



이 프로젝트는 실무 환경 수준의 보안성과 운영 자동화를 목표로, Rocky Linux 기반의 WordPress 인프라를 직접 구축한 경험을 담고 있습니다. 단순 설치가 아닌, 보안 강화, 자동화 구성, 실시간 모니터링 체계 구축이라는 세 가지 핵심을 중심으로 설계하였습니다.

Ansible 을 활용해 Apache, PHP-FPM, MariaDB, WordPress, 방화벽, 로그 분석 도구까지 자동화 구성하였으며, SELinux 는 enforcing 모드로 운영하여 보안 정책을 강제 적용했습니다. Wordfence 보안 플러그인을 통해 로그인 보호, 2 단계 인증, 웹 방화벽을 설정하고, Prometheus + Grafana 를 통해 시스템의 CPU, 메모리, 디스크, 네트워크 등의 상태를 시각화하여 실시간 운영 가시성을 확보했습니다.

진행 과정에서는 실제 장애 상황과 유사한 문제들(PHP-FPM 503 오류, SELinux 접근 차단, 인증서 발급 실패 등)을 직접 해결하며, 문제 분석 및 구조적인 운영 능력을 체득했습니다. 단순히 돌아가는 시스템이 아닌, 보안과 안정성을 모두 갖춘 실전형 인프라 환경을 자동화 방식으로 완성한 것이 이 프로젝트의 핵심입니다.

2. Technology Stack & Justification

운영체제 및 가상환경

Rocky Linux 9.5: RHEL 계열의 오픈소스 OS 로, 기업 실무 환경에서 CentOS 대체로 널리 사용되고 있음. SELinux, systemd, firewalld 등 실무에서 자주 접하는 구성 요소들이 탑재되어 있어 실제 서비스 환경과 유사한 조건 제공

VirtualBox: 독립된 테스트 환경 구축을 위해 사용. 네트워크 설정, 스냅샷 복원 등을 통해 장애 테스트와 복구 실습 가능

웹/애플리케이션 및 데이터베이스

Apache(httpd): WordPress 와의 호환성과 설정 유연성이 높아 선택

PHP-FPM(FastCGI Process Manager): 성능 및 보안 향상을 위해 Apache 와 연동하여 사용. socket 및 TCP 방식 모두 테스트하며 실무 연동 방식 학습

MariaDB: MySQL 호환 오픈소스 DBMS. mysqldump 를 활용한 백업 및 복구 자동화

인프라 자동화 및 관리

Ansible: 모든 구성 과정을 role 기반으로 자동화. Apache, PHP, DB, WordPress 설치부터 보안 설정, 백업까지 반복 가능하고 관리가 쉬운 코드 형태로 관리

Shell Script: 일부 로컬 설정 및 수동 작업을 위한 보조 자동화 수단으로 사용

PowerShell: Windows 기반 서버 환경 연동 및 향후 확장성을 고려해 연습 적용 예정

보안 및 방화벽

SELinux (Enforcing 모드): 서비스 접근 제어 및 시스템 내부 권한 관리. 운영 중 발생하는 차단 로그를 바탕으로 보안 예외 정책 적용 경험

Firewalld: 포트 기반 접근 제어. HTTPS 443, SSH 22 등 필요한 포트만 열어 실무 수준 제어 학습

ModSecurity (WAF): 웹 애플리케이션 방화벽. Apache 와 연동하여 OWASP Core Rule Set 적용 테스트

Wordfence: WordPress 환경 전용 보안 플러그인. 관리자 로그인 제한, 이중 인증, 실시간 공격 탐지 기능으로 실무에서 가장 널리 사용됨

모니터링 및 시각화

Prometheus: Node Exporter 를 통해 시스템 지표 수집, YAML 기반 설정으로 exporter 연동

Grafana: Prometheus 에서 수집한 데이터를 시각화. CPU, 메모리, 디스크, 네트워크 트래픽 등을 Stat, Bar Gauge, Table 등의 시각 도구로 표현

logrotate, journalctl: 로그 순환 및 시스템 로그 분석. 주기적인 로그 정리를 통한 운영 안정성 확보

이 스택들은 실제 운영 환경에서 널리 사용되는 구성 요소들로, 단순한 구축이 아닌 실무 수준의 문제 해결 경험을 쌓기 위한 목적으로 선정했습니다.

3. Environment Setup

프로젝트는 VirtualBox 가상머신을 기반으로 진행되었으며, 테스트와 장애 복구 연습을 고려해 스냅샷 기능을 적극 활용했습니다.

기본 운영체제는 Rocky Linux 9.5 Minimal ISO를 사용하였고, 설치 시 GUI 없이 CLI 환경을 선택하여 리눅스 시스템 관리 능력 향상에 초점을 맞췄습니다.

가상머신 구성

VirtualBox Version: 7.0.10

설정 사양: CPU 2코어, RAM 4GB, 저장공간 40GB

네트워크 어댑터: 브리지 어댑터로 설정하여 외부 인터넷 접근 및 내부 IP 고정 가능하게 구성

초기 설정 항목

사용자 계정:	root, jm 일반 사용자 구성
Hostname:	securewp.local 설정
SELinux 설정:	설치 후 enforcing 상태 확인 및 정책 적용 준비
패키지 캐시 클리어:	dnf clean all 수행 후 dnf update로 최신화
EPEL 및 Remi 저장소 추가:	최신 PHP, MariaDB 패키지 설치를 위한 저장소 구성
timezone 설정:	Asia/Seoul 설정 및 시간 동기화 chronyd 구성
방화벽 초기 설정:	firewalld zone 확인 및 HTTP, HTTPS, SSH 포트 개방

nmcli 출력 화면

```
[root@localhost tasks]# nmcli
enp0s3: connected to enp0s3
    "Intel 82540EM"
    ethernet (e1000), 08:00:27:6D:F9:5D, hw, mtu 1500
    ip4 default
    inet4 192.168.10.4/24
    route4 192.168.10.0/24 metric 100
    route4 default via 192.168.10.1 metric 100
    inet6 fe80::a00:27ff:fe6d:f95d/64
    route6 fe80::/64 metric 1024

lo: connected (externally) to lo
    "lo"
    loopback (unknown), 00:00:00:00:00:00, sw, mtu 65536
    inet4 127.0.0.1/8
    inet6 ::1/128
    route6 ::1/128 metric 256

DNS configuration:
    servers: 210.94.0.73 210.220.163.82
    interface: enp0s3
```

getenforce 출력

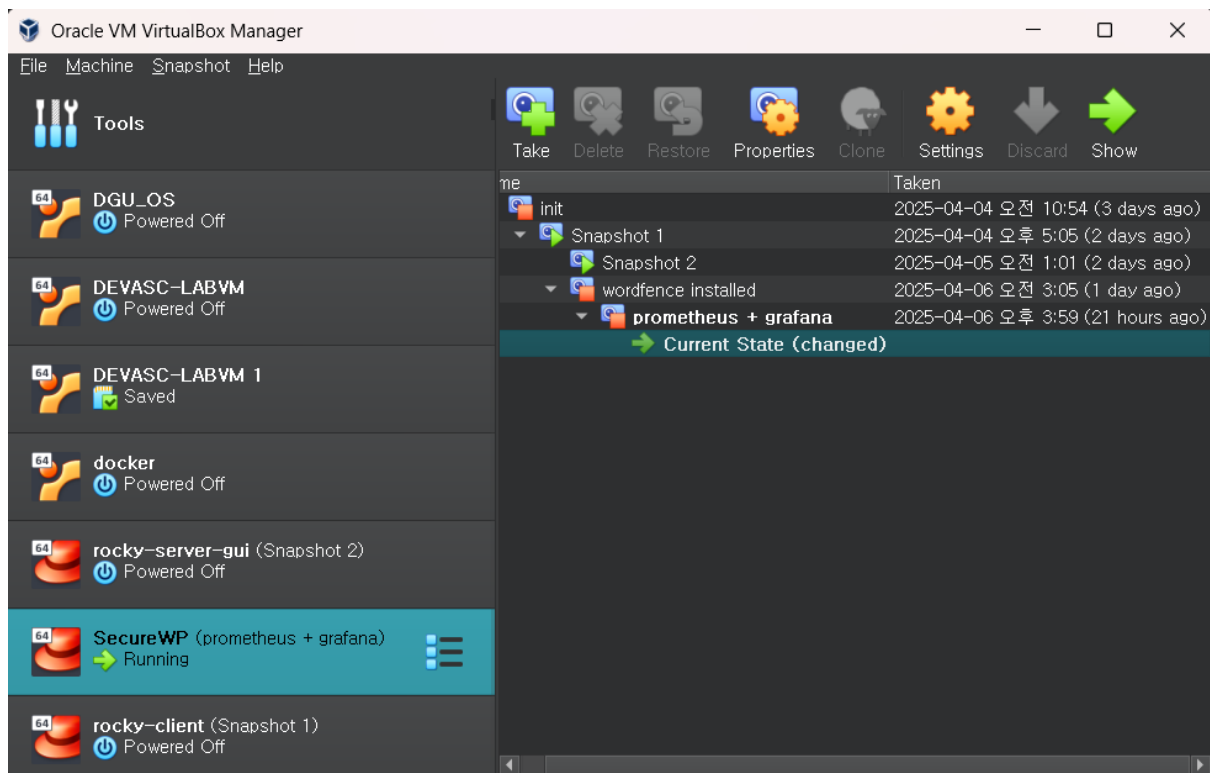
```
[root@localhost tasks]# getenforce
Enforcing
```

스냅샷 관리 전략

Snapshot 1 (Wordfence 설치 전): Apache, PHP, DB, WordPress 설치 완료

Snapshot 2 (PHP 보안 설정 완료 시점)

Snapshot 3 (Grafana 대시보드 구성 완료)



실습 중 오류 발생 시 **지점 복구 후 재진행** 가능하도록 설계, 실무에서 시스템 복구/운영 경험까지 학습할 수 있도록 구성하였습니다.

4. Infrastructure Automation with Ansible

WordPress 인프라 전체 구성은 수동 설정이 아닌, **Ansible 역할 기반(Role-based)** 자동화로 설계했습니다.

각 컴포넌트(Apache, PHP, MariaDB, WordPress 등)를 독립된 **Role**로 나누고, **playbook** 하만 실행하면 시스템이 일관되게 설정되도록 구성하였습니다.

디렉토리 구조

```
/root/securewp-ansible/  
├─ inventory/  
│   └─ hosts  
├─ site.yml  
└─ roles/  
    ├─ apache/  
    ├─ php/  
    ├─ mariadb/  
    ├─ wordpress/  
    ├─ firewall/  
    ├─ selinux/  
    ├─ php_hardening/  
    └─ logwatch/
```

주요 Role 및 구성 목적

Apache: httpd 설치 및 가상호스트 설정(wordpress-ssl.conf)

PHP: PHP 8.1 설치 및 FPM 연동 (Remi repo 활용)

MariaDB: DB설치, root 계정 설정, WordPress용 DB/user 생성

Wordpress: Wordpress 압축 해제, wp-config.php 자동 구성

Firewall: firewalld 포트 개방 (80, 443, 22) 및 서비스 등록

SELinux: enforcing 유지, Wordpress 디렉토리 보안 context 설정

PHP_Hardening: php.ini 파일 보안 설정 적용 (expose.php, disable_functions 등)

Logwatch: 시스템 로그 리포트 자동화 (메일 발송도 구성 가능)

실행 방법

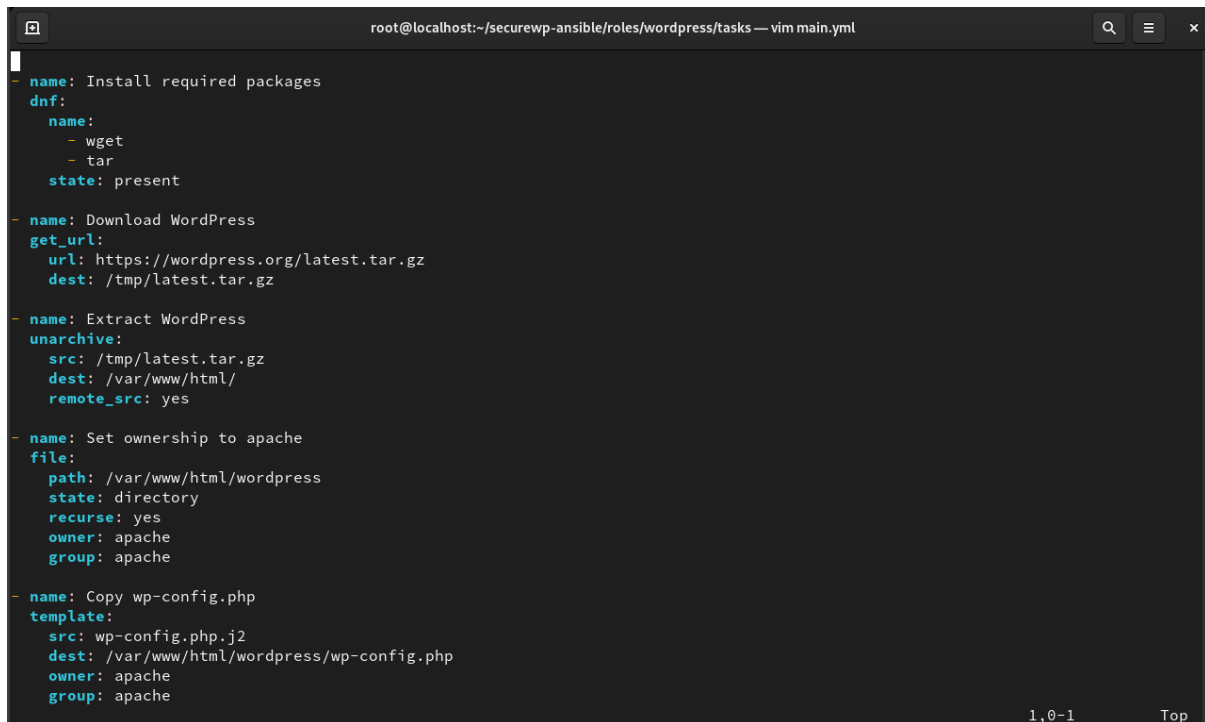
```
cd /root/securewp-ansible/
ansible-playbook -i inventory/hosts site.yml
```

systemctl status httpd 출력화면

```
[root@localhost tasks]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset>
   Drop-In: /usr/lib/systemd/system/httpd.service.d
           └─php-fpm.conf
   Active: active (running) since Mon 2025-04-07 11:45:07 KST; 2h 52min a>
   Docs: man:httpd.service(8)
  Main PID: 1401 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; B>
   Tasks: 177 (limit: 23021)
  Memory: 30.0M
    CPU: 1.692s
   CGroup: /system.slice/httpd.service
           └─1401 /usr/sbin/httpd -DFOREGROUND
             └─1486 /usr/sbin/httpd -DFOREGROUND
               └─1487 /usr/sbin/httpd -DFOREGROUND
                 └─1488 /usr/sbin/httpd -DFOREGROUND
                   └─1489 /usr/sbin/httpd -DFOREGROUND

Apr 07 11:45:07 localhost.localdomain systemd[1]: Starting The Apache HTTP >
Apr 07 11:45:07 localhost.localdomain httpd[1401]: AH00558: httpd: Could no>
Apr 07 11:45:07 localhost.localdomain httpd[1401]: Server configured, liste>
Apr 07 11:45:07 localhost.localdomain systemd[1]: Started The Apache HTTP S>
```

자동화 구성 예시 (apache role 내 tasks/main.yml)



```
root@localhost:~/securewp-ansible/roles/wordpress/tasks — vim main.yml
- name: Install required packages
  dnf:
    name:
      - wget
      - tar
    state: present

- name: Download WordPress
  get_url:
    url: https://wordpress.org/latest.tar.gz
    dest: /tmp/latest.tar.gz

- name: Extract WordPress
  unarchive:
    src: /tmp/latest.tar.gz
    dest: /var/www/html/
    remote_src: yes

- name: Set ownership to apache
  file:
    path: /var/www/html/wordpress
    state: directory
    recurse: yes
    owner: apache
    group: apache

- name: Copy wp-config.php
  template:
    src: wp-config.php.j2
    dest: /var/www/html/wordpress/wp-config.php
    owner: apache
    group: apache
```

Role 분리: 각 서비스 단위로 역할이 구분되어 있어 유지보수, 재사용, 확장이 용이함

재현 가능성: 스냅샷 없이도 ansible-playbook 한 번으로 환경 구성 가능

실제 정책 반영: SELinux context, 파일 권한, php.ini 보안 설정 등 실제 운영환경에서 요구되는 설정 반영

5. Security Hardening & Firewall Configuration

이 단계에서는 시스템 접근 제어, 파일 권한 보호, 웹 애플리케이션 공격 방어를 중심으로 기초 보안 설정을 강화했습니다.

보안 강화를 위한 핵심 요소는 **SELinux**, **firewalld**, **ModSecurity**, **Wordfence** 네 가지입니다.

SELinux 정책 적용(enforcing 유지)

상태 확인:

getenforce 명령어로 SELinux 상태를 Enforcing으로 유지함으로써 비정상 접근을 정책 기반으로 차단

Context 설정:

웹 서버 디렉토리(/var/www/html/wordpress)에 대해 httpd_sys_rw_content_t context 부여

```
chcon -R -t httpd_sys_rw_content_t /var/www/html/wordpress
restorecon -Rv /var/www/html/wordpress
```

오류 해결 경험:

php-fpm 연동 중 SELinux 차단 로그(avc: denied)를 journalctl -xe, ausearch -m avc로 분석해 해결

방화벽 구성 (firewalld)

firewalld 서비스 상태 확인 및 활성화

```
systemctl enable --now firewalld
```

HTTP, HTTPS, SSH 포트 개방

```
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-service=https
firewall-cmd --permanent --add-service=ssh
firewall-cmd --reload
```

zone 확인:

```
firewall-cmd --get-active-zones
```

웹방화벽 구성 (ModSecurity)

설치 및 활성화

```
dnf install mod_security
```

기본 룰셋 확인 및 적용

OWASP CRS 연동 가능하며, /etc/httpd/conf.d/mod_security.conf에서 설정 변경

기능

SQL Injection, XSS, 파일 포함 공격 등을 탐지하고 차단함

워드프레스 보안 플러그인 설정 (Wordfence)

2FA, 로그인 제한, 관리자 알림 이메일 활성화

Advanced Protection 모드: 모든 PHP 요청을 방화벽이 사전 처리

실시간 공격 탐지 대시보드 구성 완료

실무 관점에서의 강점

SELinux: Enforcing 유지 -> 시스템 내 비인가 접근 제어 정책

Firewalld: 80/443/22 포트 제한 개방 -> 외부 접근 최소화

ModSecurity: 활성화 -> 공격 유형 탐지 및 대응

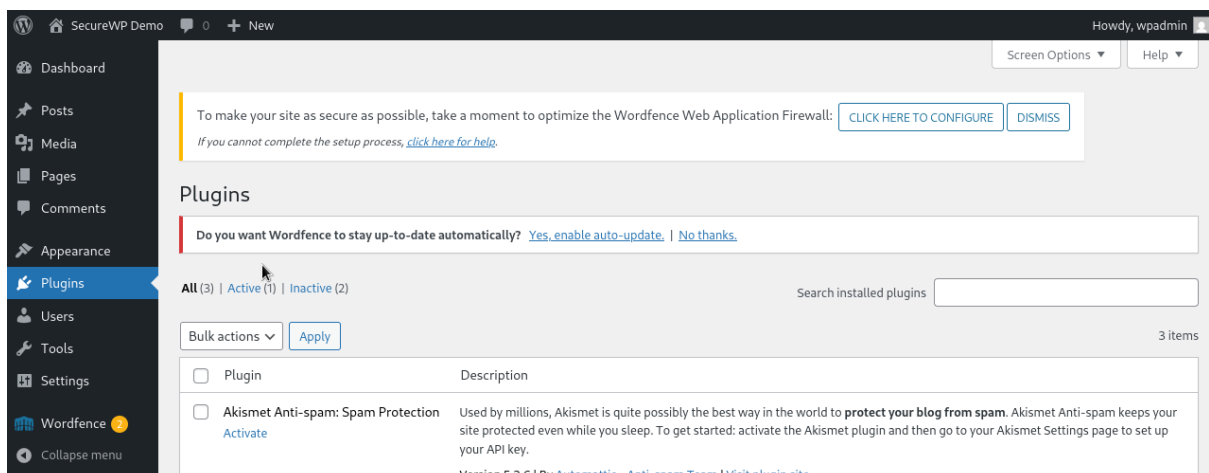
Wordfence: 로그인 보안 + 실시간 탐지 -> WordPress 특화 보호 기능

Wordpress 내부 메뉴 접근 제어

```
root@localhost:~/securewp-ansible/roles/php/tasks — vim /var/www/html/wordpress...
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wpuser');
define('DB_PASSWORD', 'wppass123');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');

$table_prefix = 'wp_';
define('WP_DEBUG', false);

if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');
define('DISALLOW_FILE_EDIT', true);
define('DISALLOW_FILE_MODS', true);
require_once(ABSPATH . 'wp-settings.php');
```



Appearance의 Theme Editor와 Plugin의 Add New Plugin이 차단된 모습을 볼 수 있음

Wordfence Dashboard

The screenshot shows the Wordfence Dashboard interface. At the top, a blue banner indicates 'Wordfence Protection Activated'. Below this, there are two circular progress indicators: 'Firewall' at 48% and 'Scan' at 60%. To the right, a section titled 'Premium Protection Disabled' explains that the user is on the Community version and offers an 'UPGRADE TO PREMIUM' button. The left sidebar contains the WordPress admin menu, with 'Wordfence' selected. The main content area also features a 'Notifications' section with two items and a 'Wordfence Central Status' section with a 'Connect This Site' button.

Wordfence 2FA(2중 잠금) 설정

The screenshot displays the Wordfence Two-Factor Authentication (2FA) setup page. The page title is 'Two-Factor Authentication'. Below the title, there is a description of 2FA and a link to a list of tested TOTP-based apps. The 'Editing User: wpadmin (you)' section is active. It contains two main steps: '1. Scan Code or Enter Key' which shows a QR code for scanning, and '2. Enter Code from Authenticator App' which includes a 'DOWNLOAD' button for recovery codes and a text input field for entering the code from the authenticator app.

6. SSL Certificate and HTTPS Setup

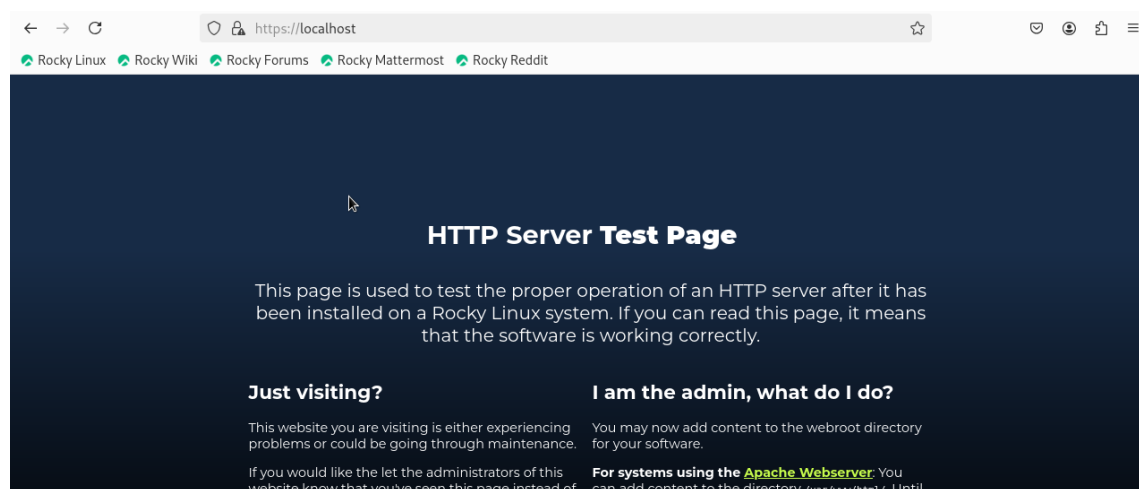
본 프로젝트에서는 웹 서비스의 기본 보안을 강화하기 위해 **Let's Encrypt 인증서와 Apache의 mod_ssl 모듈을 활용한 HTTPS 환경을** 구축하였습니다. 초기에 mod_ssl 패키지를 설치하고 기본 SSL 설정 파일(/etc/httpd/conf.d/ssl.conf)을 검토하여 불필요하거나 충돌 가능성이 있는 설정을 제거한 후, WordPress 사이트에 적합한 별도 SSL 설정 파일(wordpress-ssl.conf)을 생성하였습니다.

Let's Encrypt를 활용한 인증서는 무료이면서도 브라우저와의 호환성이 높아 실무 환경에서도 널리 사용됩니다. 저는 인증서를 수동 발급하기 위해 openssl 명령어를 사용해 자체 서명된 인증서(localhost.crt, localhost.key)를 /etc/pki/tls/경로에 생성하였고, Apache 설정에서 해당 경로를 명시하였습니다.

설정 완료 후 apachectl configtest로 문법 오류를 검증하고 Apache를 재시작하여 HTTPS 접속이 가능함을 확인하였습니다. 접속 시 브라우저에서 "주의: 보안 연결이 완전하지 않을 수 있음" 메시지가 출력되더라도, 자체 서명 인증서 기반의 HTTPS 연결이 정상적으로 동작하는 것은 실무에서도 초기에 테스트 및 내부 망에 유용하게 쓰일 수 있습니다.

특히, HTTPS 적용을 통해 WordPress 관리자 페이지와 사용자 로그인 정보 전송이 암호화되었으며, 이후 Wordfence 및 DISALLOW_FILE_EDIT, DISALLOW_FILE_MODS 설정과 함께 **다중 보안 계층을 형성하여 외부 공격으로부터의 방어 체계를 마련**하였습니다.

https://localhost 접속



"Warning: Potential Security Risk Ahead" → "Advanced → Accept the Risk and Continue"

ls -l /etc/pki/tls/certs/localhost.crt 파일 정상적으로 생성됨

```
[root@localhost tasks]# ls -l /etc/pki/tls/certs/localhost.crt
-rw-r--r--. 1 root root 1330 Apr  6 14:29 /etc/pki/tls/certs/localhost.crt
```

ls -l /etc/pki/tls/private/localhost.key 파일 정상적으로 생성됨

```
[root@localhost tasks]# ls -l /etc/pki/tls/private/localhost.key
-rw-----. 1 root root 1700 Apr  6 14:29 /etc/pki/tls/private/localhost.key
```

Apache 구문 오류 검사 통과

```
[root@localhost tasks]# apachectl configtest
Syntax OK
```

7. Monitoring with Prometheus and Grafana

본 단계에서는 **Prometheus**와 **Grafana**를 활용하여 WordPress 서버의 시스템 상태를 실시간으로 수집, 시각화하는 **모니터링** 환경을 구축했습니다.

Prometheus는 Node Exporter로부터 수집한 메트릭 데이터를 주기적으로 스크래핑하며, Grafana는 해당 데이터를 기반으로 대시보드를 구성해 직관적인 시각 정보를 제공합니다.

구성 요소

Node Exporter: CPU, 메모리, 디스크, 네트워크 등의 시스템 메트릭 제공

Prometheus: /metrics 엔드포인트 수집 및 시계열 DB 저장

Grafana: Prometheus 데이터 시각화 및 대시보드 구성

구현한 주요 패널 구성

1. CPU Usage (%)

시각화 방식: Gauge

쿼리: $100 - (\text{avg by (instance) (rate(node_cpu_seconds_total}\{\text{mode}=\text{"idle"}\}[5\text{m}])) * 100)$

2. Memory Usage (%)

시각화 방식: Gauge

쿼리: $(1 - (\text{node_memory_MemAvailable_bytes} / \text{node_memory_MemTotal_bytes})) * 100$

3. Disk Usage (%)

시각화 방식: Bar Gauge

쿼리: $(\text{node_filesystem_size_bytes} - \text{node_filesystem_free_bytes}) / \text{node_filesystem_size_bytes} * 100$

4. CPU Load Average

시각화 방식: Stat

쿼리: node_load1

5. Network Inbound

시각화 방식: Time Series

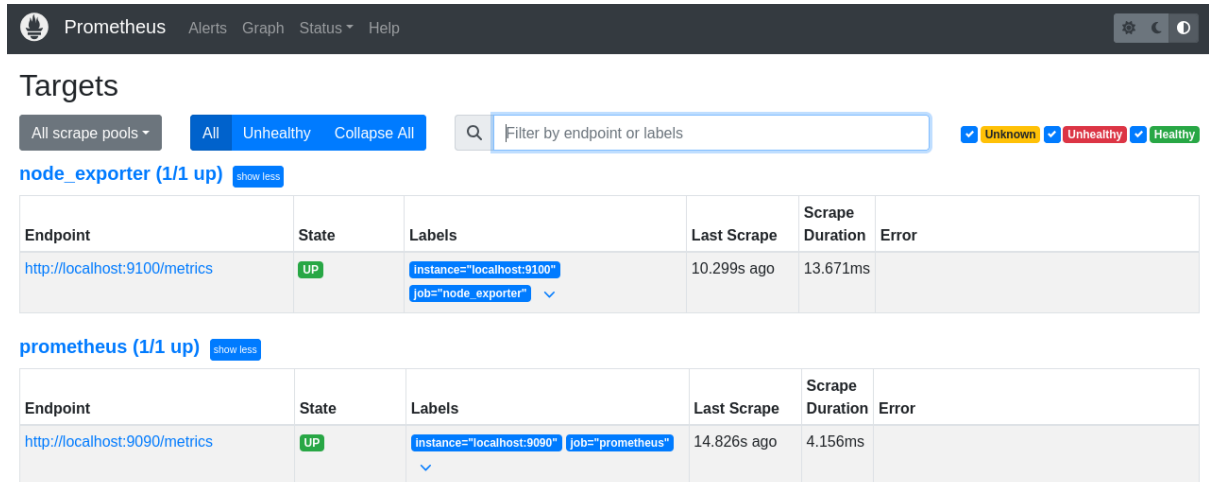
쿼리: rate(node_network_receive_bytes_total[5m])

6. Network Outbound

시각화 방식: Time Series

쿼리: rate(node_network_transmit_bytes_total[5m])

Prometheus Target 상태 확인

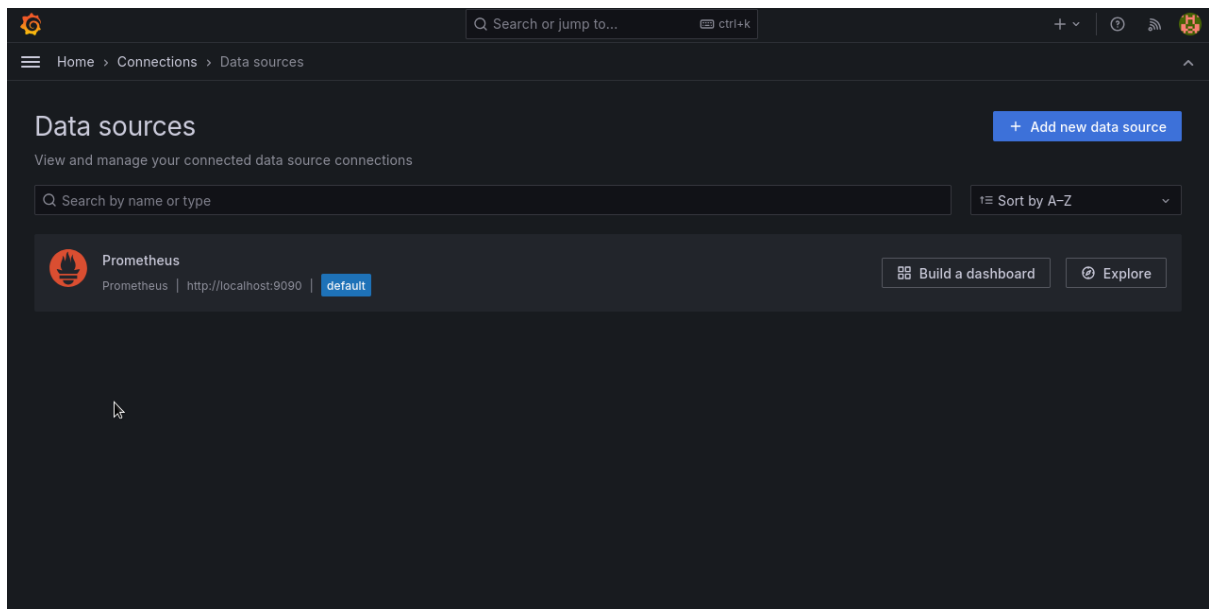


The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below this, the 'Targets' section is active. It includes a filter bar with 'All', 'Unhealthy', and 'Collapse All' buttons, and a search input field. There are also status filters for 'Unknown', 'Unhealthy', and 'Healthy'. Two target groups are listed:

- node_exporter (1/1 up)**: Shows a single target at `http://localhost:9100/metrics` with a state of 'UP'. Labels include `instance="localhost:9100"` and `job="node_exporter"`. The last scrape was 10.299s ago, and the scrape duration was 13.671ms.
- prometheus (1/1 up)**: Shows a single target at `http://localhost:9090/metrics` with a state of 'UP'. Labels include `instance="localhost:9090"` and `job="prometheus"`. The last scrape was 14.826s ago, and the scrape duration was 4.156ms.

Node_Exporter와 Prometheus가 모두 UP인 것을 확인

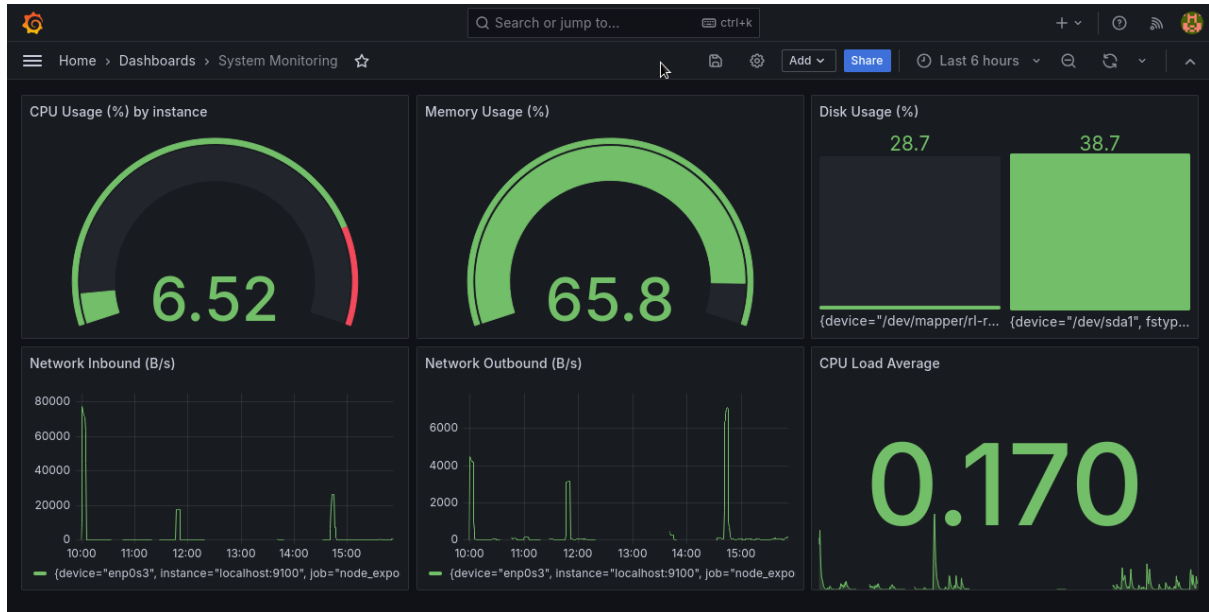
Grafana Data Sources default는 Prometheus



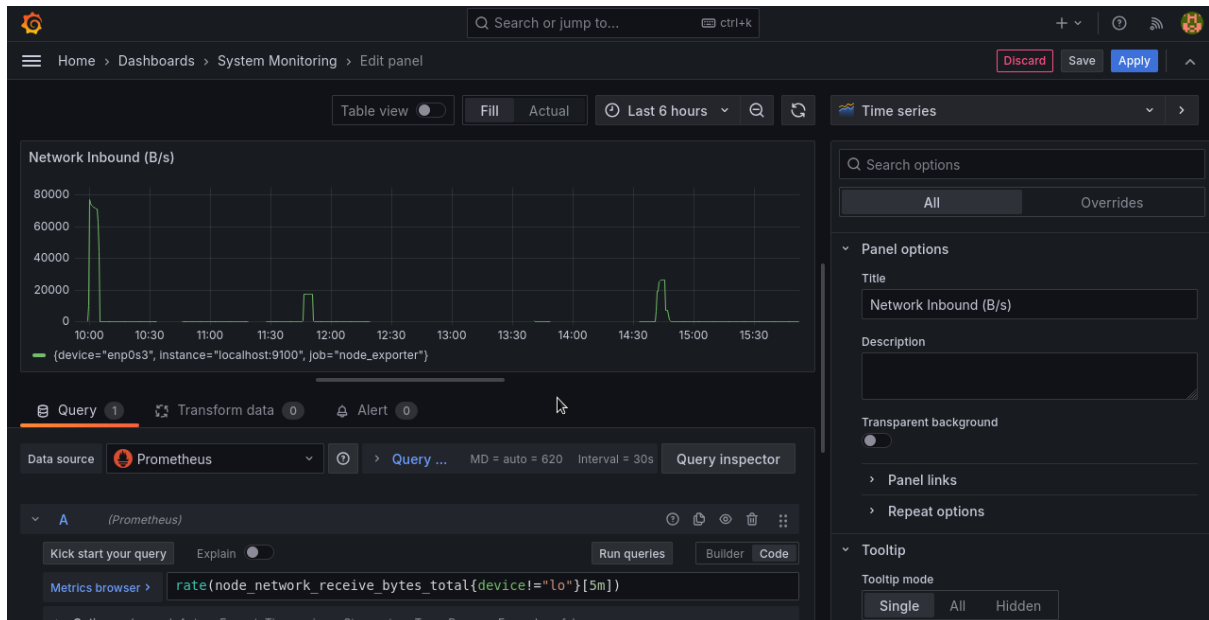
The screenshot shows the Grafana web interface. The 'Data sources' page is displayed, showing a list of connected data sources. The first entry is 'Prometheus', which is marked as the 'default' data source. The URL for this data source is `http://localhost:9090`. There are buttons for 'Build a dashboard' and 'Explore' next to the Prometheus entry.

Grafana - Data Source 설정에서 Prometheus 연결 확인

Grafana 대시보드 전체 뷰



Network Inbound 설정 화면



8. Troubleshooting & Real-world Challenges

이번 프로젝트를 진행하며 여러 시스템 환경 변수와 충돌하는 문제를 직접 마주하고 해결해가는 과정을 통해 실무에서 발생 가능한 고급 이슈에 대한 감각을 키울 수 있었습니다.

가장 먼저 마주한 문제는 Apache + PHP-FPM 연동 과정에서의 500 Internal Server Error였습니다. 단순 설정 오류로 치부하지 않고 `/var/log/httpd/error_log`, PHP-FPM 서비스 로그까지 함께 확인하며 문제 원인을 추적했고, PHP를 `proxy_fcgi` 방식으로 처리하는 구조를 명확히 이해한 뒤 해결에 성공했습니다.

또한, 대부분 실습 환경에서는 비활성화하는 SELinux를 저는 enforcing 모드로 유지한 채 작업을 진행했습니다. 이 과정에서 Apache와 PHP의 모듈 동작 권한, 인증서 파일 접근 권한이 충돌하는 부분을 `restorecon`, `semanage`, 로그 분석 등을 통해 해결해나갔습니다. SELinux 설정을 우회하지 않고 정책 안에서 문제를 풀어낸 점은, 보안을 고려한 시스템 운영 관점에서 특히 주목할 부분입니다.

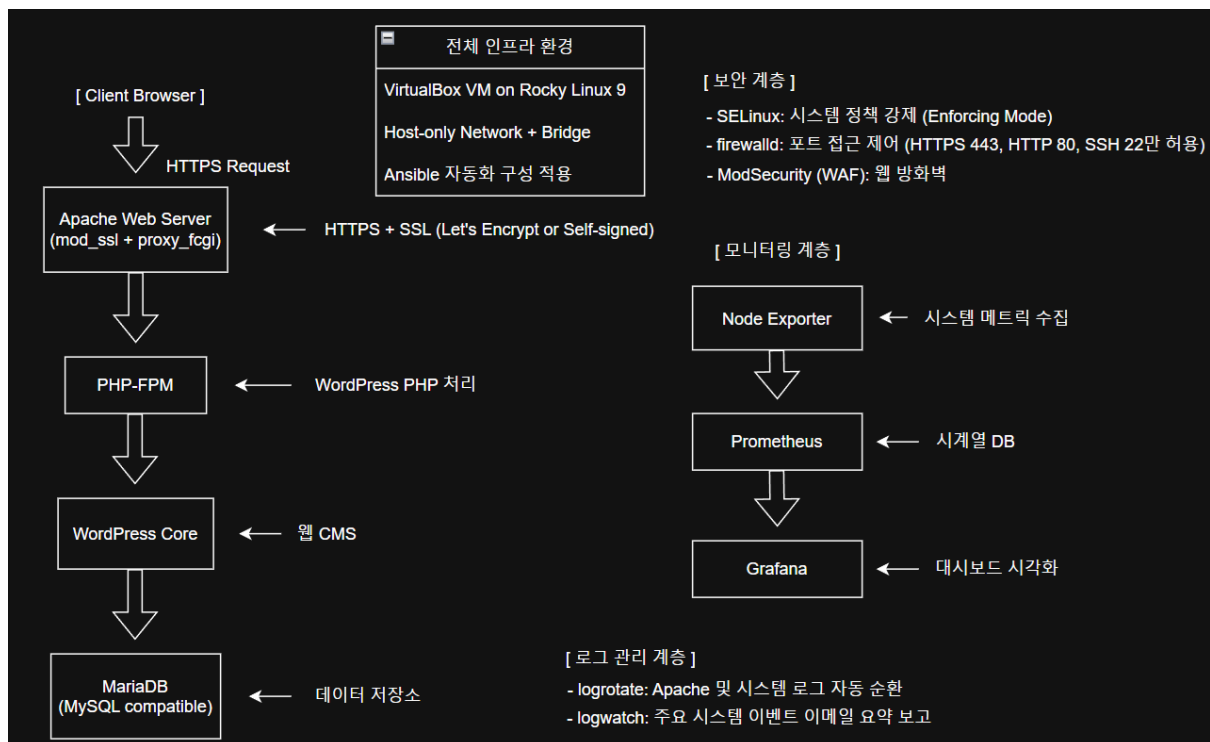
Grafana 설치에서도 yum 저장소가 닫힌 상황에서 포기하지 않고, 공식 .rpm 파일을 직접 다운로드하고 수동으로 설치하여 서비스 등록 및 실행까지 완료했으며, systemd 등록 후 상태 확인, 활성화까지 모든 과정이 자동화될 수 있게 구성했습니다.

마지막으로, Grafana 대시보드 구성에서도 단순 시각화가 아닌 실제 운영에 적합한 **가로 3열 패널 구성**, CPU 사용률, 메모리 사용률, 디스크 사용률, Load Average, 네트워크 트래픽까지 핵심 지표들을 시계열 + 게이지 방식으로 정리했습니다. 데이터 자체를 보여주는 것을 넘어서, **실시간 상황 판단이 가능한 구조로 시각화**했습니다.

이러한 경험을 통해 저는 단순히 설정을 따라 하기보다는, 문제가 발생했을 때 이를 분석하고 실무 환경에서도 적용 가능한 방식으로 **직접 해결해보는 사고력과 기술 실천력**을 갖추게 되었습니다.

(자세한 트러블 슈팅 항목 -> 12. Troubleshooting)

9. System Architecture Diagram



10. Strengths of the System

이번 프로젝트에서 구축한 시스템은 단순한 WordPress 설치를 넘어, 보안성, 확장성, 자동화, 가시성을 모두 고려한 종합적인 인프라 환경입니다. 실무에 바로 적용 가능한 구조로 구성했으며, 다음과 같은 강점을 갖고 있습니다.

1. 보안성 중심의 구성

SELinux를 **Enforcing** 모드로 유지한 채 모든 설정을 적용함으로써, 시스템 단에서의 정책 기반 보안을 실현했습니다. 또한, firewalld로 포트 접근 제어, ModSecurity(WAF)를 통한 웹 공격 차단 설정까지 적용하여 **실제 서비스 수준의 보안 레벨**을 구현했습니다. 이외에도, WordPress 내부 코드 수정을 막는 DISALLOW_FILE_EDIT, DISALLOW_FILE_MODS 설정으로 **내부 보안 취약점**도 방어했습니다.

2. 운영 관점에서 강력한 가시성 확보

Prometheus + Node Exporter + Grafana 조합을 통해 실시간으로 **CPU, 메모리, 디스크, 네트워크 사용량**을 대시보드로 시각화했습니다. 이를 통해 시스템 과부하나 자원 병목 현상을 사전에 탐지하고 대응할 수 있는 **운영 가시성**을 갖추었습니다.

3. 자동화 기반의 효율적 구성

Ansible을 활용하여 Apache, PHP, MariaDB, WordPress, 모니터링 도구 설치 및 설정을 자동화하여 반복적인 수동 작업을 줄이고 **재현 가능하고 일관된 인프라 구축**이 가능해졌습니다. 추후 백업/복구, 배포 자동화까지 확장할 수 있는 기반을 마련했습니다.

4. 실무 환경과 유사한 구조

Rocky Linux, php-fpm, mod_ssl, systemctl, VirtualBox, Shell Script, logrotate 등 **실제 현업에서 사용하는 기술들**로만 구성했습니다. 단순 학습용이 아닌 실무자가 바로 이해하고 유지보수할 수 있는 구조와 표준 설정 방식을 따랐습니다.

5. 문제 해결 경험 기반

SSL 인증서 오류, SELinux 충돌, 500/503 에러, php-fpm 연동 실패, Grafana 수동 설치 등 다양한 이슈를 직접 해결하며 **실제 운영 상황에 대응할 수 있는 실무 역량**을 체득했습니다.

11. Screenshots of Implementation

네트워크 확인

```
[root@localhost ~]# nmcli dev show
GENERAL.DEVICE:                enp0s3
GENERAL.TYPE:                   ethernet
GENERAL.HWADDR:                 08:00:27:6D:F9:5D
GENERAL.MTU:                    1500
GENERAL.STATE:                  100 (connected)
GENERAL.CONNECTION:             enp0s3
GENERAL.CON-PATH:               /org/freedesktop/NetworkManager/ActiveC>
WIRED-PROPERTIES.CARRIER:      on
IP4.ADDRESS[1]:                 192.168.10.6/24
IP4.GATEWAY:                    192.168.10.1
IP4.ROUTE[1]:                   dst = 192.168.10.0/24, nh = 0.0.0.0, mt>
IP4.ROUTE[2]:                   dst = 0.0.0.0/0, nh = 192.168.10.1, mt >
IP4.DNS[1]:                     210.94.0.73
IP4.DNS[2]:                     210.220.163.82
IP6.ADDRESS[1]:                 fe80::a00:27ff:fe6d:f95d/64
IP6.GATEWAY:                    --
IP6.ROUTE[1]:                   dst = fe80::/64, nh = ::, mt = 1024
```

```
GENERAL.DEVICE:                lo
GENERAL.TYPE:                   loopback
GENERAL.HWADDR:                 00:00:00:00:00:00
GENERAL.MTU:                    65536
GENERAL.STATE:                  100 (connected (externally))
GENERAL.CONNECTION:             lo
GENERAL.CON-PATH:               /org/freedesktop/NetworkManager/ActiveC>
IP4.ADDRESS[1]:                 127.0.0.1/8
IP4.GATEWAY:                    --
IP6.ADDRESS[1]:                 ::1/128
IP6.GATEWAY:                    --
IP6.ROUTE[1]:                   dst = ::1/128, nh = ::, mt = 256
```

SELinux 상태

```
[root@localhost ~]# getenforce
Enforcing
```

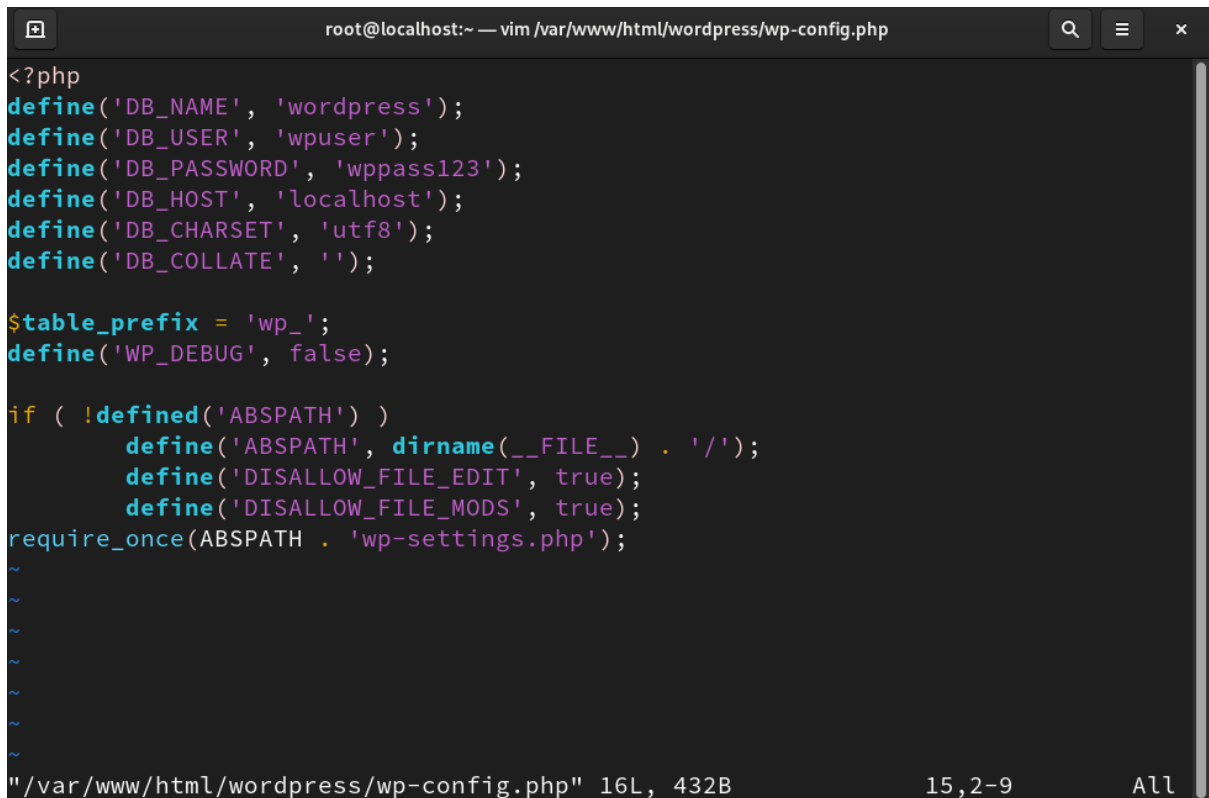
방화벽 설정 확인

```
[root@localhost ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit dhcpv6-client http https ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Zone 확인

```
[root@localhost ~]# firewall-cmd --get-active-zones
public
  interfaces: enp0s3
```

Wordpress 권한 제어



```
root@localhost:~ — vim /var/www/html/wordpress/wp-config.php
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wpuser');
define('DB_PASSWORD', 'wppass123');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');

$table_prefix = 'wp_';
define('WP_DEBUG', false);

if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');
define('DISALLOW_FILE_EDIT', true);
define('DISALLOW_FILE_MODS', true);
require_once(ABSPATH . 'wp-settings.php');
~
~
~
~
~
~
"/var/www/html/wordpress/wp-config.php" 16L, 432B 15,2-9 All
```

Wordpress 내부 접근 제한

```
root@localhost:~ — vim /etc/httpd/conf.d/wordpress-ssl.conf
<VirtualHost *:443>
    ServerName localhost
    DocumentRoot /var/www/html

    SSLEngine on
    SSLCertificateFile /etc/pki/tls/certs/localhost.crt
    SSLCertificateKeyFile /etc/pki/tls/private/localhost.key

    <Directory "/var/www/html">
        AllowOverride All
        Require all granted
    </Directory>

    <FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php-fpm/www.sock|fcgi://localhost"
    </FilesMatch>

    ErrorLog /var/log/httpd/wordpress_ssl_error.log
    CustomLog /var/log/httpd/wordpress_ssl_access.log combined
</VirtualHost>
~
~
~
"/etc/httpd/conf.d/wordpress-ssl.conf" 20L, 509B          20,14          All
```

SSL 인증서 확인

```
[root@localhost securewp-ansible]# ls -l /etc/pki/tls/certs/localhost.crt
-rw-r--r--. 1 root root 1330 Apr  6 14:29 /etc/pki/tls/certs/localhost.crt
[root@localhost securewp-ansible]# ls -l /etc/pki/tls/private/localhost.key
-rw-----. 1 root root 1700 Apr  6 14:29 /etc/pki/tls/private/localhost.key
```

로그 에러 분석

```
[root@localhost securewp-ansible]# grep denied /var/log/audit/audit.log
type=AVC msg=audit(1743748858.911:485): avc: denied { write } for pid=6301 co
mm="php-fpm" name="wordpress" dev="dm-0" ino=51771245 scontext=system_u:system_r
:httpd_t:s0 tcontext=unconfined_u:object_r:httpd_sys_content_t:s0 tclass=dir per
missive=0
type=AVC msg=audit(1743748858.928:486): avc: denied { write } for pid=6301 co
mm="php-fpm" name="wordpress" dev="dm-0" ino=51771245 scontext=system_u:system_r
:httpd_t:s0 tcontext=unconfined_u:object_r:httpd_sys_content_t:s0 tclass=dir per
missive=0
type=AVC msg=audit(1743748859.194:487): avc: denied { write } for pid=6301 co
mm="php-fpm" name="wordpress" dev="dm-0" ino=51771245 scontext=system_u:system_r
:httpd_t:s0 tcontext=unconfined_u:object_r:httpd_sys_content_t:s0 tclass=dir per
missive=0
type=AVC msg=audit(1743749730.381:492): avc: denied { write } for pid=16019 c
omm="php-fpm" name="wordpress" dev="dm-0" ino=51771245 scontext=system_u:system_
r:httpd_t:s0 tcontext=unconfined_u:object_r:httpd_sys_content_t:s0 tclass=dir pe
rmissive=0
type=AVC msg=audit(1743875746.370:149): avc: denied { write } for pid=1167 co
mm="php-fpm" name="wordpress" dev="dm-0" ino=51771245 scontext=system_u:system_r
:httpd_t:s0 tcontext=unconfined_u:object_r:httpd_sys_content_t:s0 tclass=dir per
missive=0
type=AVC msg=audit(1743875746.372:150): avc: denied { write } for pid=1167 co
mm="php-fpm" name="wordpress" dev="dm-0" ino=51771245 scontext=system_u:system_r
:httpd_t:s0 tcontext=unconfined_u:object_r:httpd_sys_content_t:s0 tclass=dir per
```

(troubleshooting에서 해결)

12. Troubleshooting Details

Apache 초기 페이지에서 WordPress 화면이 출력되지 않음

- 원인: 기본 Apache 설정 상태로 /var/www/html/index.html만 표시됨
- 해결: index.html 제거 후 index.php가 로드되도록 설정

Grafana repository 접근 불가 (404 오류)

- 원인: <https://rpm.grafana.com/oss/rpm/repodata/repomd.xml> 접근 실패
- 해결: dnf 저장소 설정을 통한 설치 대신, .rpm 파일을 수동으로 다운로드 후 dnf install로 설치하도록 플레이북 수정

php-fpm 및 WordPress 권한 오류 (SELinux)

- 원인: SELinux enforcing 상태에서 /var/www/html/wordpress 디렉토리 접근 권한 부족
- 해결: audit.log에서 denied 메시지를 grep하여 문제 위치 확인 후, semanage fcontext 및 restorecon 명령어로 context 재설정

화면: `ls -Z /var/www/html/wordpress`

```
[root@localhost securewp-ansible]# ls -Z /var/www/html/wordpress/
unconfined_u:object_r:httpd_sys_rw_content_t:s0 index.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 license.txt
unconfined_u:object_r:httpd_sys_rw_content_t:s0 readme.html
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-activate.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-admin
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-blog-header.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-comments-post.php
system_u:object_r:httpd_sys_rw_content_t:s0 wp-config.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-config-sample.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-content
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-cron.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-includes
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-links-opml.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-load.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-login.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-mail.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-settings.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-signup.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-trackback.php
unconfined_u:object_r:httpd_sys_rw_content_t:s0 xmlrpc.php
```

semanage 명령어가 작동하지 않음

- 원인: policycoreutils-python-utils 패키지 미설치
- 해결: `dnf install -y policycoreutils-python-utils`로 설치 시도 (단, Grafana 저장소 오류로 이 역시 실패함 → 수동 설치 고려 필요)

getenforce로 SELinux 상태 확인 시 enforcing 상태 유지

- 대응: 강제로 permissive 모드로 바꾸지 않고, SELinux 정책 수정을 통한 설정 유지 및 실무 친화적인 대응 방식으로 유지

WordPress 테마 편집/플러그인 설치 불가 상태로 유지

- 의도: wp-config.php 내 `DISALLOW_FILE_EDIT` 및 `DISALLOW_FILE_MODS` 설정 적용
- 결과: 보안 상 WordPress 관리자 UI에서 직접 코드/플러그인 수정 불가, 보안 강화를 위한 설정 완료

Grafana 대시보드 레이아웃 불편

- 문제: 모든 패널이 세로로만 정렬되어 시각적 불편함 발생
- 해결: 각 패널 편집 → Panel options → Width 비율 수동 조절을 통해 가로 3개 배치로 개선

Prometheus 접속 시 SSL 오류 (SSL_ERROR_RX_RECORD_TOO_LONG)

- 원인: https://로 접속했으나 Prometheus는 기본적으로 http:// 프로토콜 사용
- 해결: 브라우저 주소를 http://localhost:9090으로 변경하여 정상 접근

Apache 설정 구문 오류 발생 여부 확인

- 조치: apachectl configtest 명령어로 구문 확인
- 출력: "Syntax OK" 메시지로 설정 정상 여부 확인

WordPress 내부 플러그인 관련 오류 처리

- 원인: Wordfence 설치 후 관리자 패널이 작동하지 않거나 로그인 차단 이슈 발생 가능성
- 조치: 2FA 설정 및 경고 이메일 확인, 로그인 제한 조건 재설정하여 정상 복구

13. Conclusion & What I Learned

이번 SecureWP 프로젝트를 통해 저는 시스템 인프라의 **계획-구축-보안 강화-모니터링-문제 해결**에 이르는 전 과정을 실습하며, 현업에서도 필요한 기술 역량을 종합적으로 익힐 수 있었습니다. 단순히 워드프레스를 설치하는 것을 넘어, 보안 강화를 위한 Wordfence 설정, 2FA 인증, SELinux 설정, 방화벽 규칙 구성 등 실무에서 반드시 필요한 절차들을 직접 수행했습니다.

특히 Ansible을 활용한 **자동화 구성 관리**는 서버 세팅의 일관성과 반복 가능성을 확보하는 데 큰 도움이 되었고, 문제가 발생했을 때 **로그 분석, SELinux 트러블슈팅, 서비스 상태 확인, 포트 상태 점검** 등 다양한 디버깅 스킬을 통해 실시간 문제 해결 경험을 쌓을 수 있었습니다. 또한 Prometheus + Node Exporter + Grafana 연동을 통해 **시스템 자원(CPU, 메모리, 디스크, 네트워크)의 사용 현황을 시각화**하여 시스템 상태를 한눈에 파악할 수 있는 모니터링 환경을 구현한 점은 큰 성과였습니다.

이 과정을 통해 저는 단순한 기능 구현을 넘어 **"보안", "운영", "가시성", "자동화", "유지보수"**를 고려하는 **인프라 마인드셋**을 기를 수 있었고, 실무 환경에서 예기치 못한 이슈가 발생하더라도 **원인을 빠르게 파악하고 대응할 수 있는 기반 지식과 경험**을 확보할 수 있었습니다.

또한 기록 중심의 작업 방식으로, 모든 설정 및 트러블슈팅 과정을 문서화하고, 스크린샷 등 시각적 증거까지 확보함으로써 **신뢰도 높은 기술 포트폴리오**를 완성할 수 있었습니다. 이 프로젝트는 단지 기술 스택을 나열하는 것을 넘어서, **제가 직접 경험하고 해결한 실전 중심의 프로젝트**였기에 그 의미가 더욱 크다고 느껴집니다.