IT 인프라 아키텍처 설계

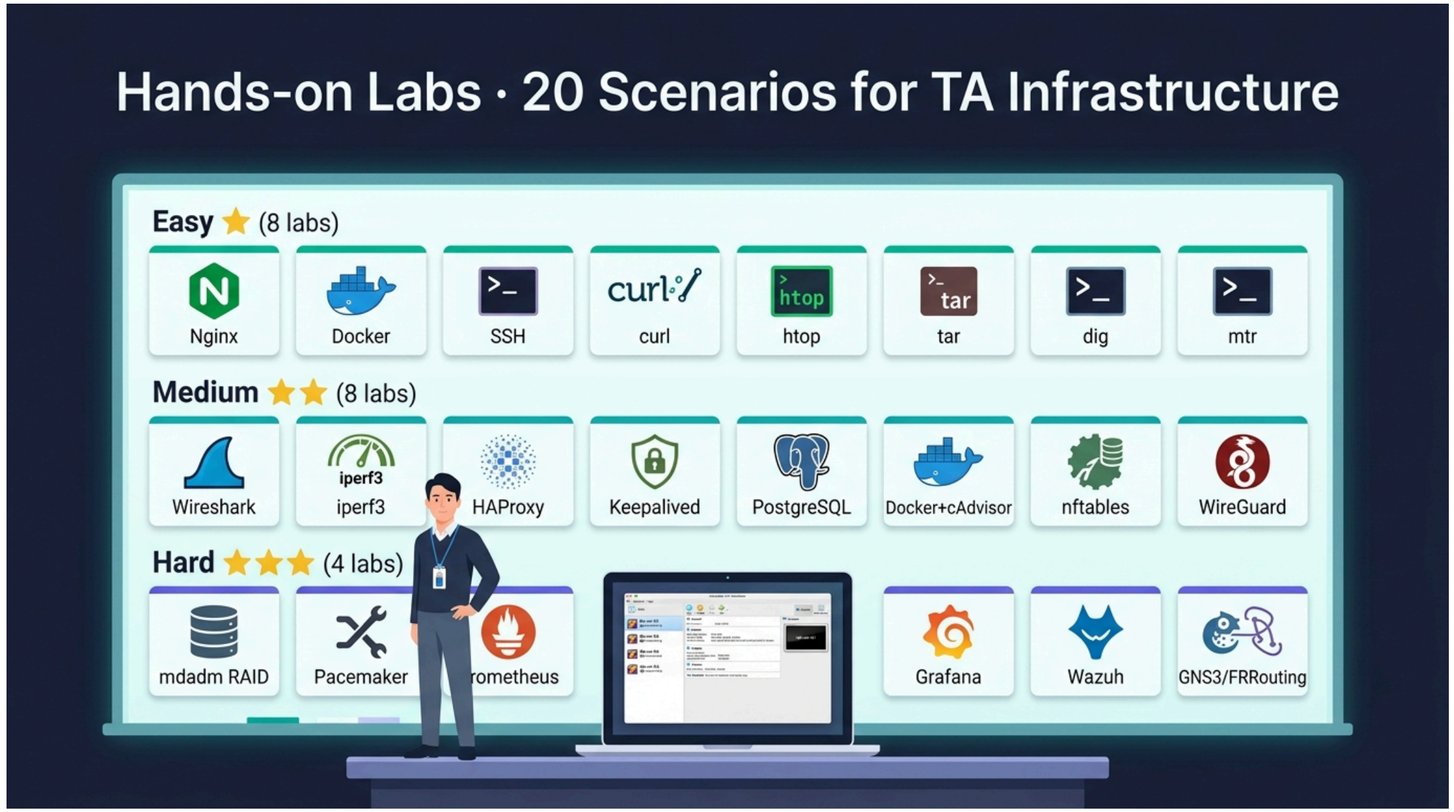
Session 20 — *실습 모음*
Hands-on Labs · 20 시나리오

VirtualBox / VMware + Linux + 모니터링 · 네트워크 · 스토리지 · 보안 · HA

난이도 ⭐ 쉬움 (8) · ⭐⭐ 중간 (8) · ⭐⭐⭐ 어려움 (4) — 강사가 수강생 난이도에 맞춰 선택

강사 박수현 · 🚀 젠아이랩스(GenAI Labs)

데이터센터 · 서버·OS·DB·GPU · 가상화·컨테이너 · 네트워크 L1~L7 · 스토리지 · 백업·DR · 보안 6대 도메인 · HA · 용량·성능·관찰성 · 설계 워크숍 · RFP · 5종 실전 케이스



VirtualBox / VMware + Linux + 모니터링 · 네트워크 · 스토리지 · 보안 · HA

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 19개 — *큐레이션 가이드* (20개 → 19개, 자원 모니터링 통합)

📌 **본 과정 실습 분류** (시간이 한정되어 모두 진행 불가) - **[필수] 7개** (⭐) — 강의 시간 내 또는 첫날 저녁 자율 — TA 입문 공통 도구 - **[권장] 5개** (⭐⭐) — Day 2·Day 3 자율 실습 — 운영·NW·HA·LB·보안 1축씩 - **[참고] 3개** (⭐⭐⭐없는 ⭐⭐) — 시간 여유 또는 관심 영역 - **[선택] 4개** (⭐⭐⭐) — 시니어 또는 추가 학습용

⭐ [필수] — 7개 · 입문 도구 1~2시간

1. **VirtualBox VM + SSH**
2. **Nginx Reverse Proxy**
3. **Docker run nginx** + 브라우저
4. ~~htop/free/df~~ → **#9에 통합** (자원 모니터링 8종)
5. **ssh-keygen** 공개키 인증
6. **curl + dig** HTTP/DNS
7. **tar/gzip/rsync** 백업 기초
8. **mtr/traceroute/ping + ss·tcpdump** 네트워크 진단

⭐⭐ [권장] — 5개 · 운영자 도구 3~4시간

9. **자원 모니터링 8종** (htop·top·free·df·du·vmstat·iostat·sar·dstat)
10. **Wireshark TCP** 3-way handshake
11. **HAProxy L7 LB** + Failover
12. **Keepalived VRRPv3** HA VIP
13. **nftables (+iptables 호환) + fail2ban** + nmap

⭐⭐ [참고] — 3개 · 관심 영역

11. **iperf3 + tc** 대역폭/지연/손실
12. **PostgreSQL pgbench** + 튜닝
13. **Docker + cAdvisor** + cgroup

⭐⭐⭐ [선택] — 4개 · 시니어 트랙

17. **mdadm RAID-5** + 디스크 장애·rebuild
18. **Pacemaker + Corosync** 2-node HA
19. **Prometheus + Grafana** + Node Exporter
20. **Wazuh SIEM·HIDS·EDR**

🛠️ 환경 사전 다운로드

- VirtualBox 7.x · VMware Workstation Pro 17.x (개인 무료)
- Ubuntu Server 24.04 LTS · Wireshark · GNS3 (선택)
- **호스트**: 16GB RAM 권장 · **VM**: 2 vCPU / 4GB / 40GB

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #2 — *Nginx Reverse Proxy — 환경·시나리오* ⭐

🎯 목적

- **Reverse Proxy** 동작 원리 이해
- Nginx 설정 파일 (`proxy_pass`·헤더 전달) 작성
- 백엔드 1대 + Nginx 1대 최소 구성으로 **L7 프록시** 이해

🛠️ 도구

- VirtualBox + Ubuntu Server 24.04 (실습 #1에서 만든 VM)
- **Nginx** (`apt install nginx`)
- **Python http.server** (백엔드 mock, 기본 내장)
- `curl` (검증)

💻 시나리오 (약 15분)

```
# 1) Nginx 설치 + 시작
sudo apt update && sudo apt install -y nginx
sudo systemctl status nginx     # active (running) 확인

# 2) 백엔드 띄우기 (다른 터미널)
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #2 — *Nginx Reverse Proxy — 분석·해설* ⭐

🔍 관전 포인트 (5종)

1. `proxy_pass http://127.0.0.1:3000` — Nginx가 80 → 백엔드 3000으로 *재전송*
2. `Host` **헤더** — Nginx가 받은 Host를 백엔드에 전달 (다중 도메인 시 핵심)
3. `X-Real-IP` / `X-Forwarded-For` — 백엔드가 *실제 클라이언트 IP* 알 수 있게
4. `nginx -t` — reload 전 syntax 검증 (운영 필수 습관)
5. **access.log** — 80번에 들어온 모든 요청 기록 (분석·디버깅)

✅ 결론 (수강생이 학습할 것)

- **L7 프록시 = HTTP 헤더를 보고 라우팅**하는 장치 (vs L4는 IP·포트)
- 백엔드 1대 → 여러 대로 늘릴 때 (`upstream` 블록) 자연스럽게 **로드밸런서**가 됨 (실습 #12 HAProxy로 확장)
- 운영 Nginx 설정의 **기본 패턴 4종** (proxy_pass·헤더 3종·access_log) 정착

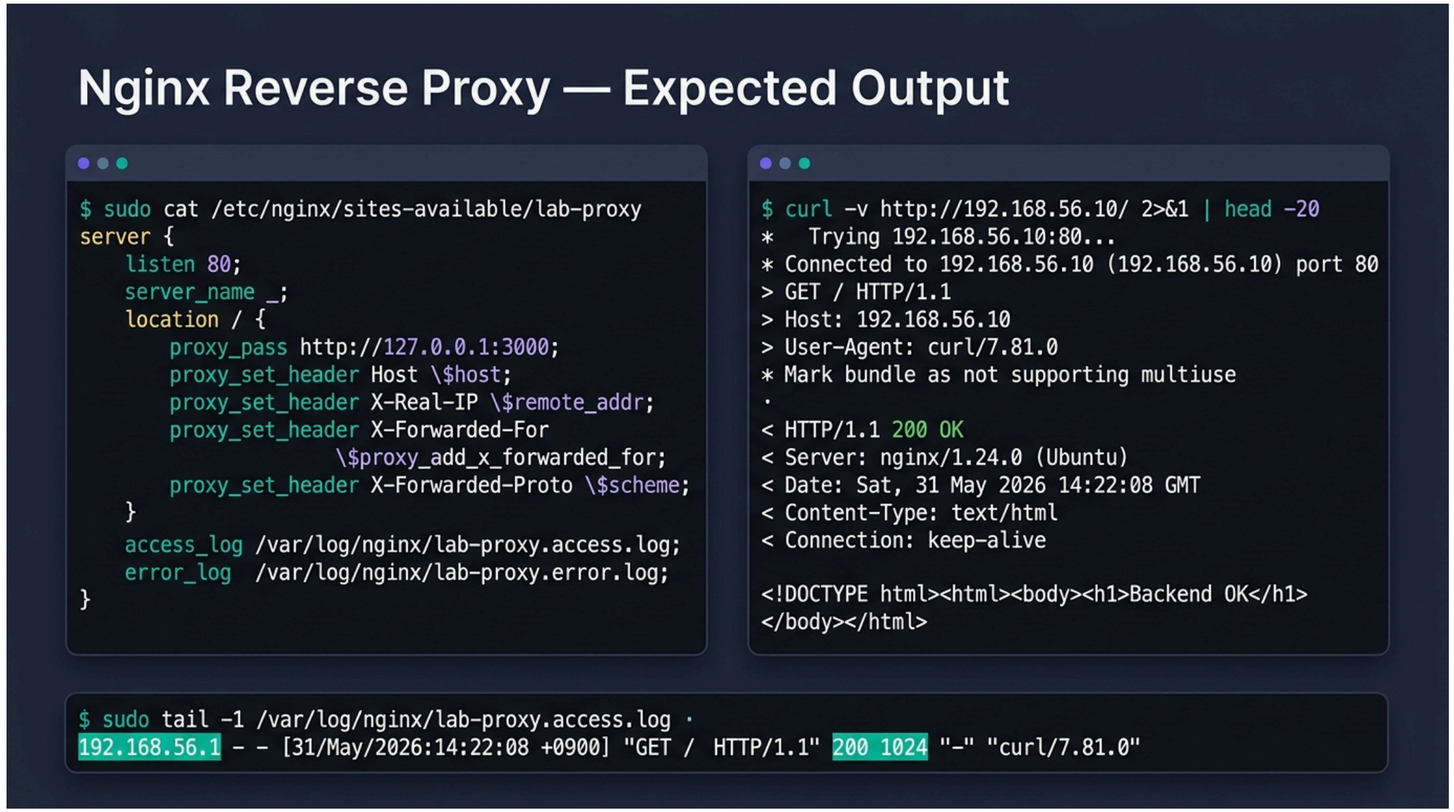
🎙️ 강사 해설 (약 5분)

- 왜 백엔드를 `127.0.0.1`에만 바인딩? → **외부 직접 접근 차단**, Nginx만 통하게 (보안 + 통합 로깅)
- `proxy_pass`의 슬래시 유무가 URI rewrite에 영향 (예: `proxy_pass http://backend/` vs `proxy_pass http://backend`)
- 운영에서 자주 만나는 **502 Bad Gateway** = 백엔드 다운/타임아웃 → access.log + error.log 같이 봐야
- 다음 실습 #12 HAProxy로 확장 시: `upstream backend { server 127.0.0.1:3000; server 127.0.0.1:3001; }` 한 줄로 LB

🐞 자주 만나는 오류

- `nginx: [emerg]` → `nginx -t`로 syntax 확인
- 502 → 백엔드 살아있나 (`curl http://127.0.0.1:3000`)
- 403 → SELinux 또는 디렉토리 권한
- ports.conf 충돌 → `sudo ss -tlnp | grep :80`

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #2 — Nginx Reverse Proxy ⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #1 — *VirtualBox VM 만들고 SSH 접속 — 환경·시나리오* ⭐

🎯 목적

- **VirtualBox로 격리된 Linux 환경** 만들기
- 가장 기초인 **SSH 원격 접속** 익히기
- NAT vs Bridge vs **Host-Only** 어댑터 이해

🛠️ 도구

- VirtualBox 7.x (virtualbox.org/wiki/Downloads)
- Ubuntu Server 24.04 LTS ISO (releases.ubuntu.com)
- OpenSSH (Ubuntu 기본 포함)

💻 시나리오 (약 15분)

```
# 1) VirtualBox UI에서 VM 생성
#    Name: lab-vm-01 / Type: Linux / Ubuntu 64-bit
#    Memory: 4096 MB / CPU: 2 / Disk: 40 GB VDI

# 2) Settings → Network
#    Adapter 1: NAT       (외부 인터넷)
#    Adapter 2: Host-Only (vboxnet0, 192.168.56.0/24)
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #1 — *VirtualBox VM 만들고 SSH 접속 — 분석·해설* ⭐

🔍 관전 포인트

1. **NAT vs Host-Only** — NAT는 외부 인터넷, Host-Only는 호스트↔VM 사설망
2. **Adapter 2 (192.168.56.10)** — 고정 사설 IP라 SSH 안정
3. **sshd 상태** — `systemctl status ssh` 가 `active (running)`
4. **첫 접속 시 fingerprint** — `~/.ssh/known_hosts` 에 추가됨
5. **VBoxManage list vms** — CLI로도 VM 관리 가능

✅ 결론

- **VM = 호스트 위에서 돌아가는 격리된 컴퓨터** — 망쳐도 호스트 멀쩡
- **SSH = 원격 운영의 기본** — 모든 후속 실습의 진입점
- **Host-Only 어댑터 = 실습 표준** — IP 고정 + 외부 차단

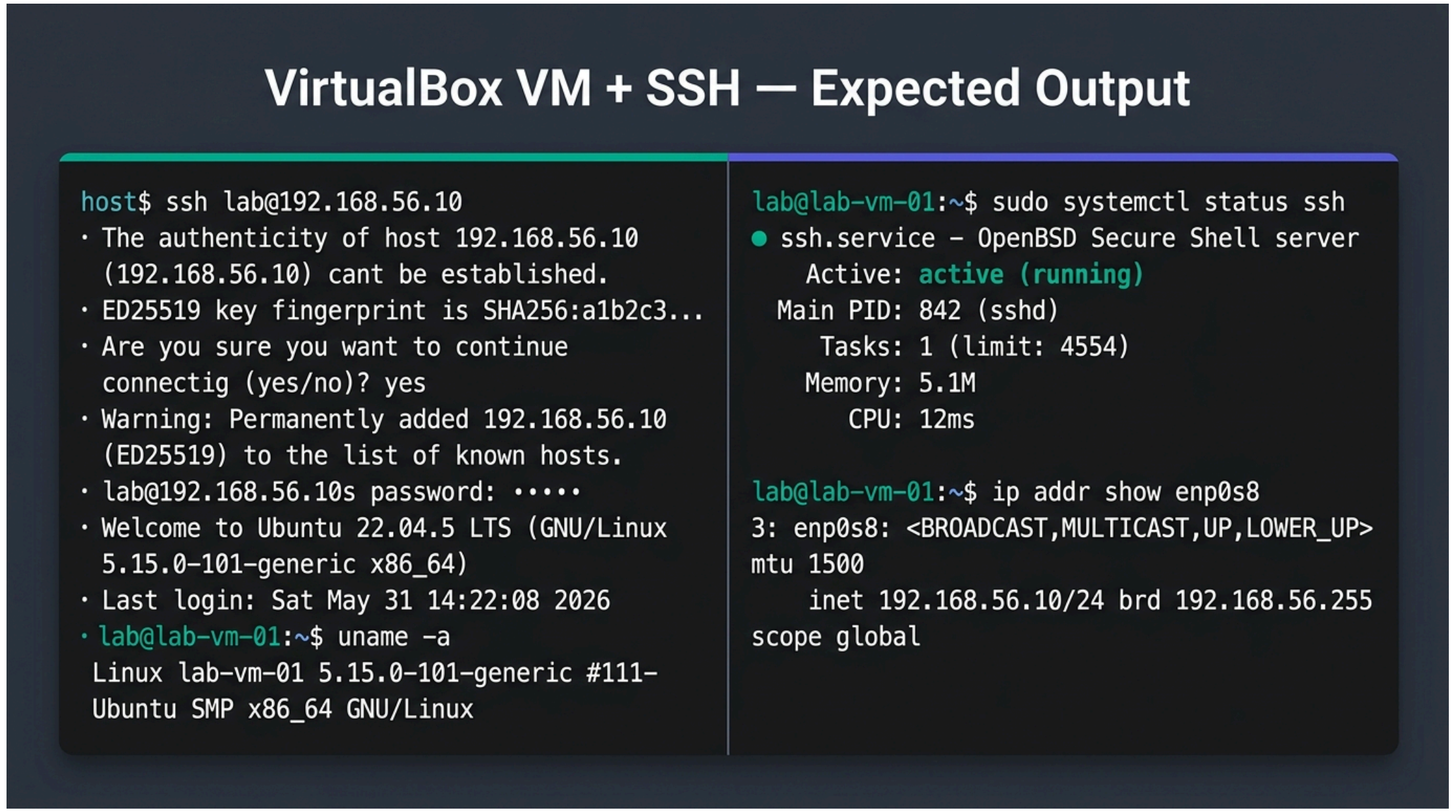
🎙️ 강사 해설 (약 5분)

- 네트워크 어댑터 4종: NAT / NAT Network / Bridged / Host-Only / Internal — 실습엔 NAT + Host-Only 조합 표준
- `~/.ssh/known_hosts` 변경 시 → `ssh-keygen -R 192.168.56.10`
- 다음 실습 #5 ssh-keygen으로 비밀번호 X 접속 확장
- VBoxManage = GUI 없이 CI/CD에서 VM 자동화 가능

🐞 자주 만나는 오류

- `Connection refused` → sshd 미실행 or 포트 차단 (`sudo ufw status`)
- `Host key verification failed` → `ssh-keygen -R <ip>`
- `192.168.56.10 도달 X` → Host-Only 어댑터 미추가
- 마우스가 VM에 갇힘 → Right-Ctrl 키로 해제

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #1 — VirtualBox VM 만들고 SSH 접속 ⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #3 — *Docker 한 줄로 웹 서버 띄우기 — 환경·시나리오* ⭐

🎯 목적

- **Docker 한 줄로 웹 서버** 띄우기 (모든 설치 0초)
- **컨테이너 = 격리된 프로세스** 개념 이해
- 포트 매핑 (`-p 80:80`), 백그라운드 (`-d`), 이름 (`--name`)

🛠️ 도구

- Docker Engine (get.docker.com 스크립트)
- 이미지: `nginx` (Docker Hub)
- `curl` (검증)

💻 시나리오 (약 10분)

```
# 1) Docker 설치 (한 줄)
curl -fsSL https://get.docker.com | sudo sh
sudo usermod -aG docker $USER
newgrp docker     # 또는 재로그인

# 2) nginx 컨테이너 한 줄로 띄우기
docker run -d --name web -p 80:80 nginx
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #3 — *Docker 한 줄로 웹 서버 띄우기 — 분석·해설* ⭐

🔍 관전 포인트

1. `-d` — detach (백그라운드) / 없으면 포그라운드
2. `-p 80:80` — 호스트 포트 : 컨테이너 포트
3. `--name web` — 이후 명령에서 ID 대신 이름 사용
4. **이미지 자동 pull** — 처음엔 Docker Hub에서 다운
5. `docker exec -it` — 살아있는 컨테이너 안으로 SSH-like 진입

✅ 결론

- **컨테이너 = 1줄로 띄우는 격리 프로세스**
- 이미지 = "스냅샷", 컨테이너 = "그 스냅샷의 실행 인스턴스"
- **운영 OSS의 90%는 Docker 한 줄로 시연 가능** — 학습 가속

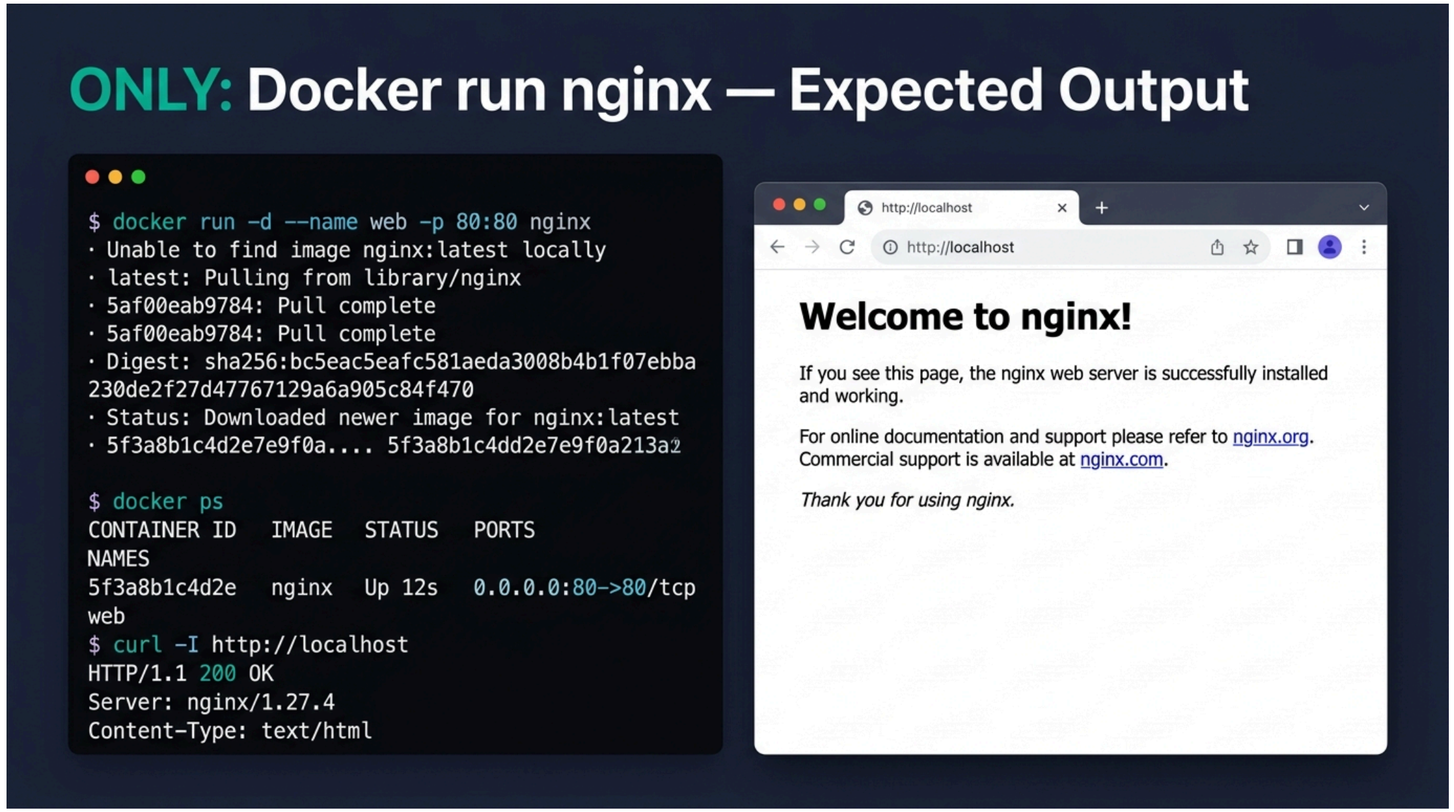
🎙️ 강사 해설 (약 5분)

- VM과 차이: VM = 별도 커널 (수 GB) · 컨테이너 = 호스트 커널 공유 (수 MB)
- 컨테이너 격리 = Linux namespace (PID·NET·MNT 등) + cgroup (CPU·Mem 제한)
- Docker Hub vs Private Registry (자체 이미지 보관)
- 다음 실습 #15에서 cgroup 한도 + cAdvisor로 격리 검증

🐞 자주 만나는 오류

- `permission denied` → docker 그룹 추가 + `newgrp docker`
- `port is already allocated` → `sudo ss -tlnp | grep :80` → 기존 nginx 중지
- `Cannot connect to the Docker daemon` → `sudo systemctl start docker`
- `docker: command not found` → 설치 스크립트 다시

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #3 — Docker 한 줄로 웹 서버 띄우기 ⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

type: content slide: 7 title: 실습 #5 — ssh-keygen 공개키 인증 ⭐ [필수] · 환경·시나리오 key: s20-016-lab-ssh-keygen-setup

실습 #5 — *ssh-keygen + ssh-copy-id 비밀번호 없는 SSH — 환경·시나리오* ⭐

🎯 목적

- **비밀번호 없이 SSH** — 보안 + 편의 동시 확보
- 공개키 vs 개인키 비대칭 암호화 이해
- `~/.ssh/config` 별칭으로 운영 효율

🛠️ 도구

- `ssh-keygen`, `ssh-copy-id`, `ssh` (openssh-client 기본)
- 실습 #1의 VM `lab@192.168.56.10` 활용

💻 시나리오 (약 10분)

```
# 1) 키 쌍 생성 (ED25519 — 작고 빠르고 안전)
ssh-keygen -t ed25519 -C "lab@$(hostname)"
# 기본 경로 ~/.ssh/id_ed25519 / 패스프레이즈는 선택
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #5 — *ssh-keygen + ssh-copy-id 비밀번호 없는 SSH — 분석·해설* ⭐

🔍 관전 포인트

1. **ED25519 vs RSA** — ED25519 추천 (작고 빠르고 강함)
2. `-N ""` — 패스프레이즈 없음 (실습용 / 운영은 패스프레이즈 권장)
3. `authorized_keys` 권한 = **600** (그 외엔 sshd 거부)
4. **fingerprint** — 첫 접속 시 `known_hosts` 등록
5. `~/.ssh/config` — 운영 SSH 별칭의 핵심

✅ 결론

- **PasswordAuthentication no** = 운영 SSH의 표준
- 공개키 = "서버에 두고 누구나 볼 수 있는 자물쇠"
- 개인키 = "절대 나가지 않는 열쇠" — 분실 시 전부 재발급

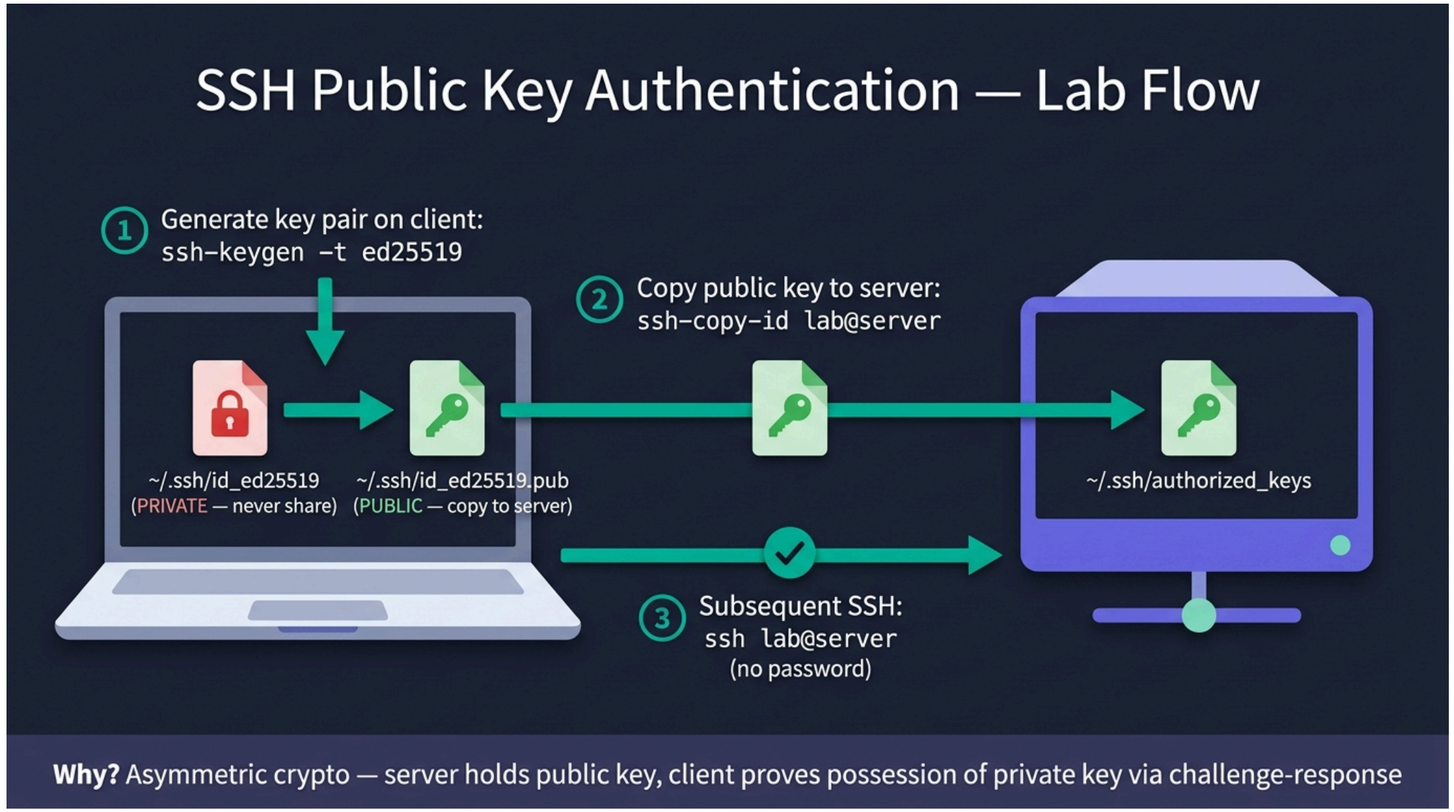
🎙️ 강사 해설 (약 5분)

- ssh-copy-id 내부: scp + cat >> authorized_keys
- 개인키 유출 시: `ssh-keygen -y -f` 로 공개키 재추출 후 `authorized_keys` 교체
- 다중 키 관리: `~/.ssh/config` 의 Host별 IdentityFile
- **2FA 강화**: `sshd` PAM + Google Authenticator

🐞 자주 만나는 오류

- `Permission denied (publickey)` → authorized_keys 권한·소유자
- `Bad permissions` → `chmod 700 ~/.ssh && chmod 600 ~/.ssh/*`
- `Agent admitted failure to sign` → `ssh-add ~/.ssh/id_ed25519`
- `known_hosts` 충돌 → `ssh-keygen -R <ip>`

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #5 — ssh-keygen + ssh-copy-id 비밀번호 없는 SSH ⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #6 — *curl + dig — HTTP·DNS 1차 디버깅 — 환경·시나리오* ⭐

🎯 목적

- **HTTP 상태코드·헤더** 실측
- **DNS 레코드** (A / AAAA / MX / NS / TXT / PTR) 조회
- 운영 장애 **1차 디버깅** 도구 익히기

🛠️ 도구

- `curl` (8.x)
- `dig` (apt install **dnsutils**)
- 보너스: `nslookup`, `host`, `openssl s_client`

💻 시나리오 (약 15분)

```
# ═══ HTTP 디버깅 ===
# 1) 응답 헤더만
curl -I https://example.com
# HTTP/2 200 · content-type: text/html ...

# 2) 상세 (TLS · 인증서 · 리다이렉트)
curl -v https://example.com 2>&1 | head -30
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #6 — *curl + dig — HTTP·DNS 1차 디버깅 — 분석·해설* ⭐

🔍 관전 포인트

1. **HTTP 상태코드** — 2xx OK / 3xx redirect / 4xx client / 5xx server
2. `curl -v` **의** `*` = 디버그 정보, `>` = 요청, `<` = 응답
3. **dig ANSWER 섹션** — 실제 레코드 값
4. `time_namelookup` **vs** `time_connect` — DNS와 TCP 분리 측정
5. `+trace` — 전 세계 DNS 위계 추적

✅ 결론

- 장애 1차 진단 = `curl -I` + `dig +short`
- DNS는 운영 장애의 30% 원인 — TTL·캐시·전파 시간 이해 필수
- `curl -w` 로 응답 시간 단계별 측정 가능

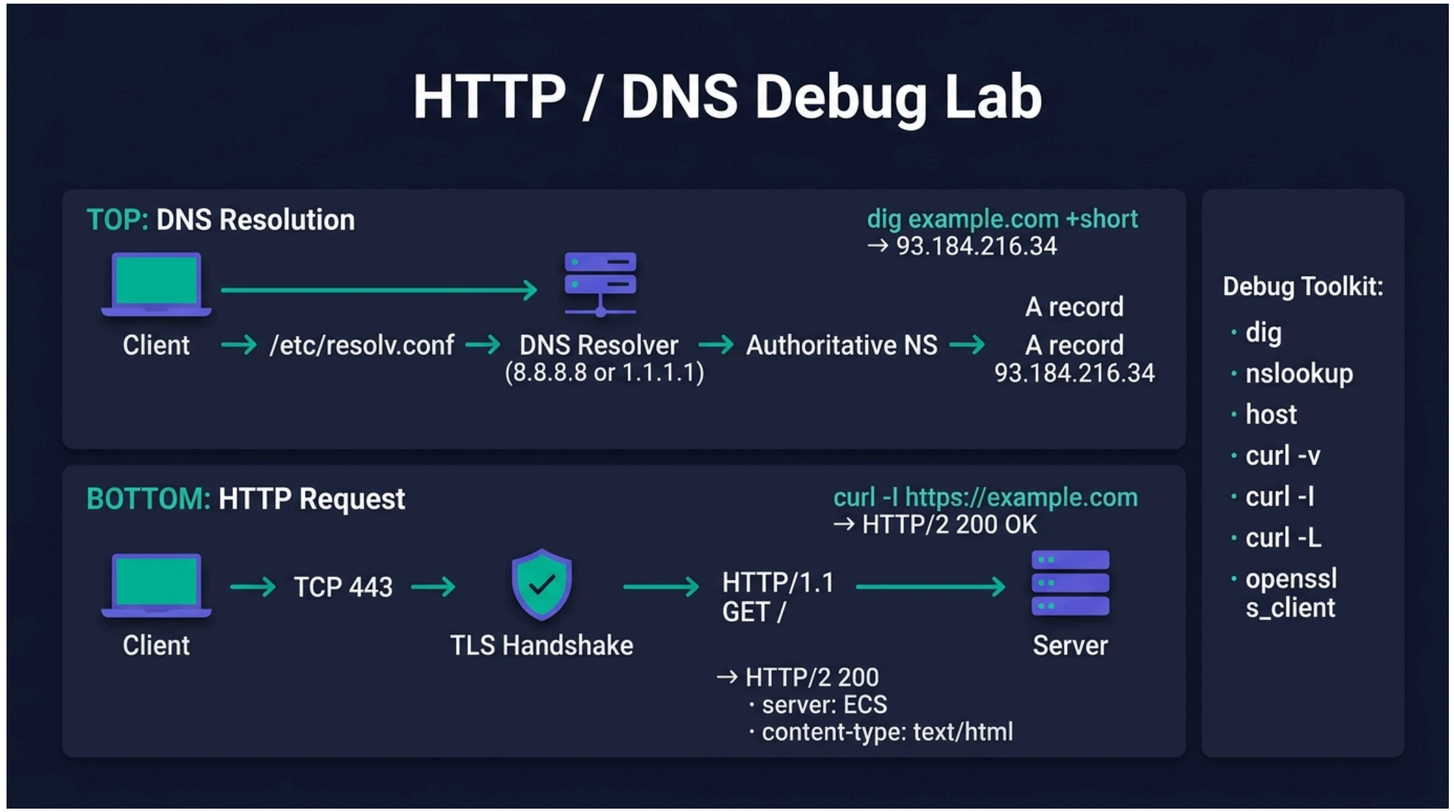
🎙️ 강사 해설 (약 5분)

- HTTP/2 멀티플렉싱 (curl `--http2`)
- TLS 1.3 핸드셰이크 0-RTT
- DNS over HTTPS (DoH) — `dig @1.1.1.1 +https`
- 다음 실습 #10 Wireshark로 TCP·TLS 패킷 단위 분석

🐞 자주 만나는 오류

- `dig: command not found` → `apt install dnsutils`
- `curl: (6) Could not resolve host` → DNS 실패 (`cat /etc/resolv.conf`)
- `curl: (35) SSL connect error` → TLS 인증서 / 시간 동기화 (NTP)
- `dig` 응답 SERVFAIL → DNSSEC 검증 실패

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #6 — curl + dig — HTTP·DNS 1차 디버깅 ⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #7 — *tar·gzip·rsync — 백업·증분·원격 복사 — 환경·시나리오* ⭐

🎯 목적

- **tar + gzip** 으로 전체 백업
- **rsync --link-dest** 로 증분 백업 (하드링크 트릭)
- **원격 백업** (rsync over SSH)

🛠️ 도구

- `tar`, `gzip` (기본 내장)
- `rsync` (`apt install rsync`)
- `cron` (정기 백업 스케줄)

💻 시나리오 (약 15분)

```
# ═══ tar + gzip 전체 백업 ===
# 1) 전체 백업 (압축)
sudo tar czf /tmp/etc-backup.tar.gz /etc/

# 2) 내용 확인 (압축 풀지 않고)
tar tzf /tmp/etc-backup.tar.gz | head -10
tar tzf /tmp/etc-backup.tar.gz | wc -l
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #7 — *tar·gzip·rsync — 백업·증분·원격 복사 — 분석·해설* ⭐

🔍 관전 포인트

1. `czf` / `xzf` / `tzf` — c(reate) / e(x)tract / list, z(gzip), f(ile)
2. `-avP` — `a` 모든 속성 보존 / `v` verbose / `P` 진행률+resume
3. `--link-dest` — 미변경 파일은 하드링크 → 디스크 절약
4. `/etc/` **vs** `/etc` — 슬래시 유무가 rsync 결과 다름
5. `du -sh` — 하드링크는 한 번만 카운트

✅ 결론

- **tar = 패키징 / rsync = 동기화**
- `--link-dest` 한 줄로 운영 수준 증분 백업
- 원격은 `-e ssh` 한 옵션만 추가하면 됨

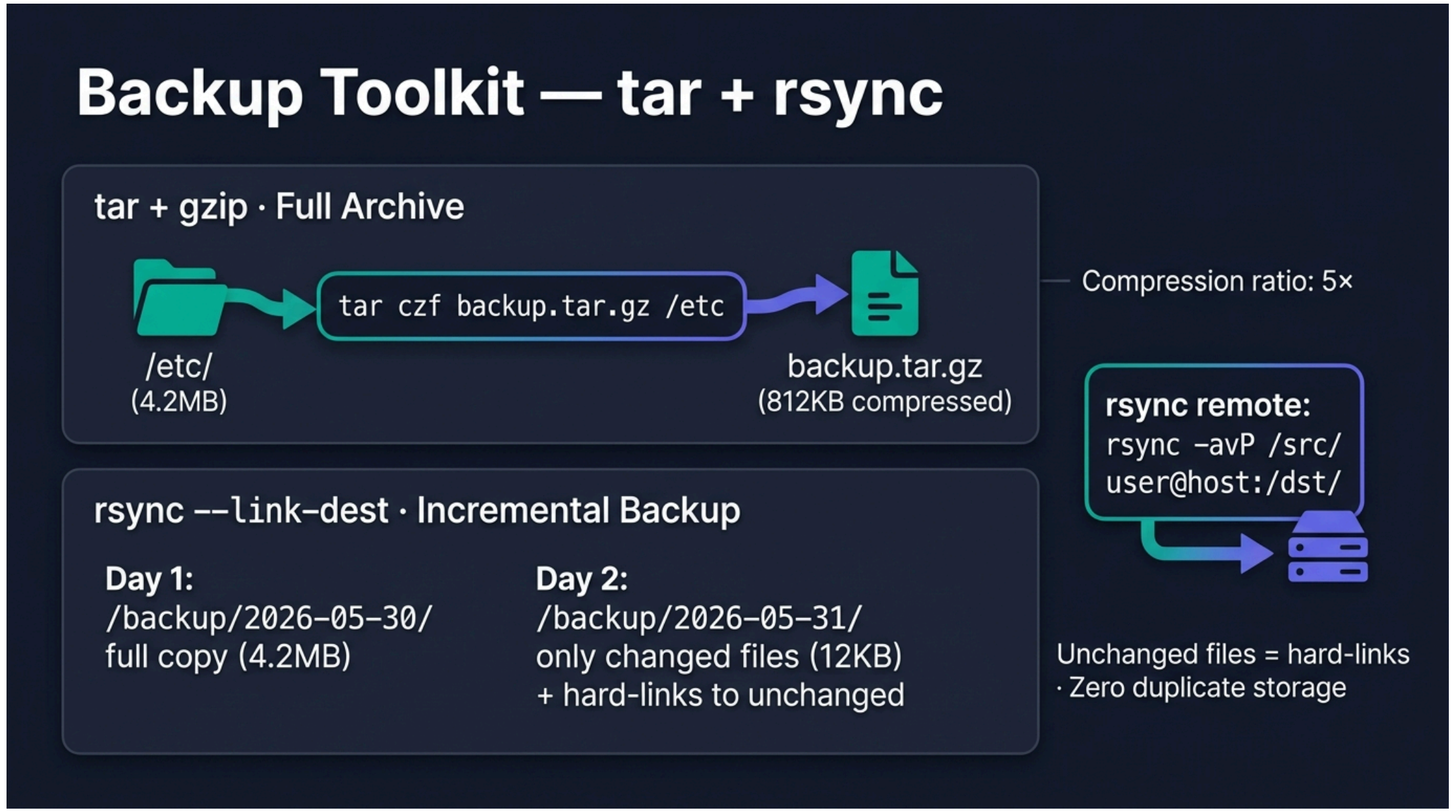
🎙️ 강사 해설 (약 5분)

- `--link-dest` 트릭 = Apple Time Machine 동일 방식
- `--delete` 옵션 = 원본에 없는 파일은 백업에서도 삭제 (양날의 검)
- `-z` (rsync) = 전송 중 압축, 저속 회선에서 유용
- 다음 실습 #17에서 mdadm RAID-5로 백업 대상 자체의 가용성 확장

🐞 자주 만나는 오류

- `rsync: failed to set times` → 권한 부족 (`sudo`)
- 슬래시 누락 → 디렉토리 중첩 (`/src` → `/dst/src/`)
- `--link-dest` 절대경로만 인식
- `tar: Removing leading /` → 안전 동작 (절대경로 → 상대)

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #7 — tar·gzip·rsync — 백업·증분·원격 복사 ⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #8 — *mtr·traceroute·ping — 네트워크 경로 추적 — 환경·시나리오* ⭐

🎯 목적

- **어느 hop에서 지연/손실** 발생하는가 식별
- ICMP TTL 메커니즘 익히기
- 운영 네트워크 장애 1차 진단

🛠️ 도구 (네트워크 진단 묶음)

- `ping` (기본 도달성)
- `traceroute` (apt install traceroute · 경로)
- `mtr` (apt install mtr · 지속 통계)
- `ss` (소켓·연결 상태 — netstat 후속)
- `tcpdump` (선택 — 패킷 캡처 1차 도구)
- 보너스: `ip route`, `ss -tlnp`

💻 시나리오 (약 15분)

```
# 1) ping 기본 (도달성)
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #8 — *mtr·traceroute·ping — 네트워크 경로 추적 — 분석·해설* ⭐

🔍 관전 포인트

1. **TTL 메커니즘** — 라우터가 1씩 감소시키고 0이면 ICMP Time Exceeded
2. **traceroute의** `* * *` — 라우터가 ICMP 차단 (응답 안 함, 통과는 함)
3. **mtr 누적 손실률** — 단발 손실 vs 지속 손실 구분
4. **첫 hop은 게이트웨이** — 안 보이면 인터페이스 문제
5. **MTU 1500** — 일반, MTU 1480 = PPPoE, MTU 9000 = Jumbo

✅ 결론

- 지연·손실의 **위치 격리** = mtr 1분이면 90%
- traceroute의 `*` = 차단이 아니라 ICMP 응답 미반환
- `ip route get` 으로 라우팅 결정 검증

🎙️ 강사 해설 (약 5분)

- 비대칭 라우팅: 정방향 traceroute가 역방향 경로 X
- 클라우드 (AWS·GCP)는 ICMP 거의 차단 → TCP traceroute
- MTU mismatch가 "ping 됨, 큰 데이터 안 됨" 증상
- 다음 실습 #11 iperf3+tc로 대역폭/지연 시뮬

🐞 자주 만나는 오류

- traceroute 모두 `*` → ICMP 차단 → `-T -p 443` (TCP)
- mtr 권한 → `sudo` 또는 setcap
- WiFi 첫 hop 지연 큼 → 무선 자체 지터
- 결과가 매번 다름 → ECMP 멀티 경로

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

Network Tracing — Expected Output

```
$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
· 64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=42.3 ms
· 64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=41.8 ms
· 64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=44.1 ms
· 64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=42.7 ms
· 4 packets transmitted, 4 received, 0% packet loss, time 3004ms
· rtt min/avg/max/mdev = 41.821/42.711/44.105/0.836 ms

$ traceroute -n 8.8.8.8
 1  192.168.1.1     1.234 ms  1.421 ms    1.521 ms
 2  10.0.0.1        4.512 ms   4.621 ms   4.802 ms
 3  211.45.32.1     12.342 ms  12.421 ms  12.501 ms
 4  142.250.221.50  28.241 ms  28.342 ms  28.421 ms
 5  8.8.8.8         42.341 ms  42.421 ms  42.521 ms

$ mtr --report -c 100 -n 8.8.8.8
HOST: laptop            Loss%   Snt     Last     Avg    Best    Wrst  StDev
 1. 192.168.1.1          0.0%   100      1.2     1.4     1.0     3.2    0.3
 2. 10.0.0.1             0.0%   100      4.5     4.6     4.2     6.8    0.5
 3. 211.45.32.1         12.0%   100     12.3    13.1    11.8    28.4    2.1
 4. 142.250.221.50       0.0%   100     28.2    28.4    27.9    32.1    0.8
 5. 8.8.8.8              0.0%   100     42.3    42.7    41.8    45.2    0.6
```

실습 #8 — mtr·traceroute·ping — 네트워크 경로 추적 ⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #9 — *시스템 자원 모니터링 — 8종 도구 통합 — 환경·시나리오* ⭐⭐

🎯 목적

- **1분 안에 "이 서버 지금 바쁜가?"** 판단 (빠른 진단 4종)
- Linux **자원 종합 관찰** — CPU·Mem·Disk·NW (심층 분석 4종)
- `%iowait` · `%steal` 같은 고급 지표 판독
- `stress-ng` 로 인위적 부하 발생시켜 도구별 관점 비교

🛠️ 도구 (8종)

- **빠른 진단**: `htop`, `top`, `free`, `df` / `du`
- **심층 관찰**: `vmstat`, `iostat`, `sar`, `dstat`
- `sysstat` (vmstat·iostat·sar 포함), `stress-ng`, `tmux`
- 실습 #1의 VM에서 진행

💻 시나리오 (약 30분)

```
# 1) 설치
sudo apt update
sudo apt install -y htop sysstat stress-ng tmux dstat
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #9 — *시스템 자원 모니터링 — 8종 도구 통합 — 분석·해설* ⭐⭐

🔍 관전 포인트

1. **빠른 진단 → 심층 관찰 2단계** — `uptime` · `free` · `df` 로 1분 진단 → 의심 가는 축만 `vmstat` / `iostat` / `sar` 로 깊이 보기
2. `vmstat` `wa` **컬럼** — 디스크 병목 (CPU가 IO 기다림)
3. `vmstat` `st` **컬럼** — Steal Time (호스트가 vCPU 빼앗김 = 가상화)
4. `iostat %util` — 95%+ = 디스크 포화 (NVMe·RAID 검토)
5. `sar -d` 의 `await` — IO 응답 시간 (mSec)
6. `htop` **보라색** = `%iowait` (디스크 대기)
7. `MemAvailable` vs `free` — 캐시 회수 가능 메모리 (진짜 부족 판단 기준)

✅ 결론

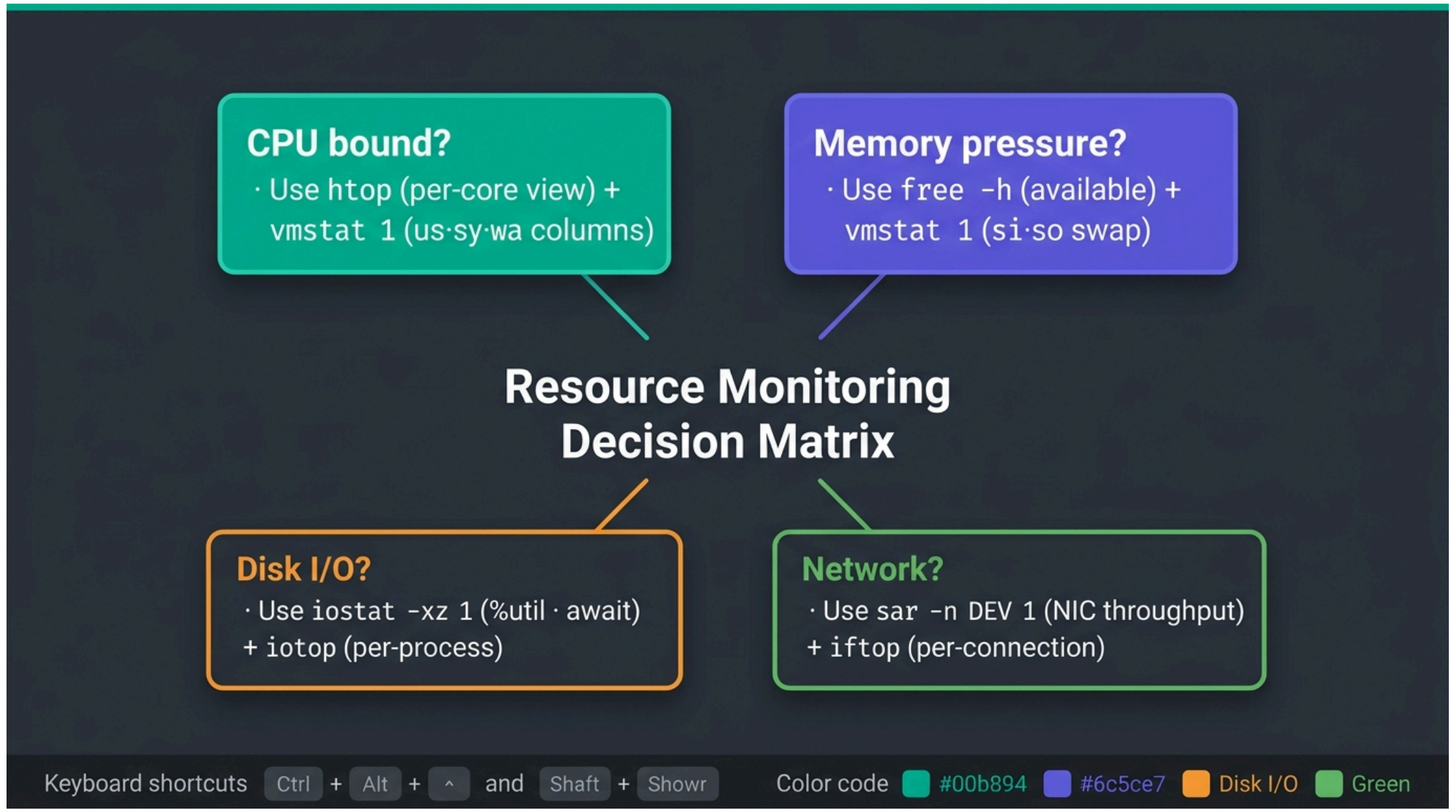
- **빠른 진단 4종 (htop·free·df·uptime)** = 운영 1분 진단의 표준
- **심층 관찰 4종 (vmstat·iostat·sar·dstat)** = "왜 바쁜가" 근본 원인 분석
- 같은 부하라도 도구마다 보는 관점이 다름 → 교차 검증이 핵심
- `sysstat` 백그라운드 수집 = 사후 분석의 핵심

🎙️ 강사 해설 (약 7분)

- load avg = "실행 중 + 실행 대기" 프로세스 평균. 8 코어 서버에서 load 8이면 100%, 16이면 200% 대기
- `buff/cache` = 회수 가능, 실제 부족 판단은 **available**
- `vmstat 1 5` — 5초만 보고 종료 (스크립트 친화)
- `iostat` 첫 출력 = 부팅 이후 누적 (무시), 두 번째부터 실측
- `sar` 데이터는 `/var/log/sa/` 에 저장 (10분 간격)
- `dstat` 는 deprecated 경고가 뜨지만 한 화면에 다 보여서 운영 현장에서 여전히 선호
- 컨테이너 환경: `cAdvisor` (실습 #15) 로 확장

🐞 자주 만나는 오류

- `htop: command not found` → `sudo apt install -y htop`



실습 #9 — htop·vmstat·iostat·sar — 자원 4종 동시 관찰 ⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #10 — *Wireshark — TCP 3-way handshake 패킷 분석 — 환경·시나리오* ⭐⭐

🎯 목적

- **TCP 3-way handshake** 패킷 단위 관찰
- Wireshark 필터·통계 사용법
- 운영 네트워크 분석 1차 도구

🛠️ 도구

- **Wireshark** (wireshark.org/download.html)
- `tcpdump` (CLI 보조, 기본 내장)
- `curl` (트래픽 발생)

💻 시나리오 (약 20분)

```
# 1) 도구 설치
sudo apt install -y wireshark tshark tcpdump
sudo usermod -aG wireshark $USER  # 비특권 캡처
# 재로그인 필요
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #10 — *Wireshark — TCP 3-way handshake 패킷 분석 — 분석·해설* ⭐⭐

🔍 관전 포인트

1. **3-way: SYN → SYN-ACK → ACK** 패킷 # 1, 2, 3 순서
2. **Seq·Ack 번호** — Seq 0 → Ack 1 (Initial Sequence Number)
3. **Window** — 받을 수 있는 버퍼 (혼잡 제어 시작점)
4. **Flow Graph** — 시간순 시각화 (Statistics 메뉴)
5. **재전송** — `tcp.analysis.retransmission` 필터

✅ 결론

- TCP는 신뢰성 = 3-way + Seq/Ack + 재전송
- Wireshark 1시간 = SE 1년 = TCP 동작 이해 가속
- tshark CLI = 운영 서버에서도 분석 가능

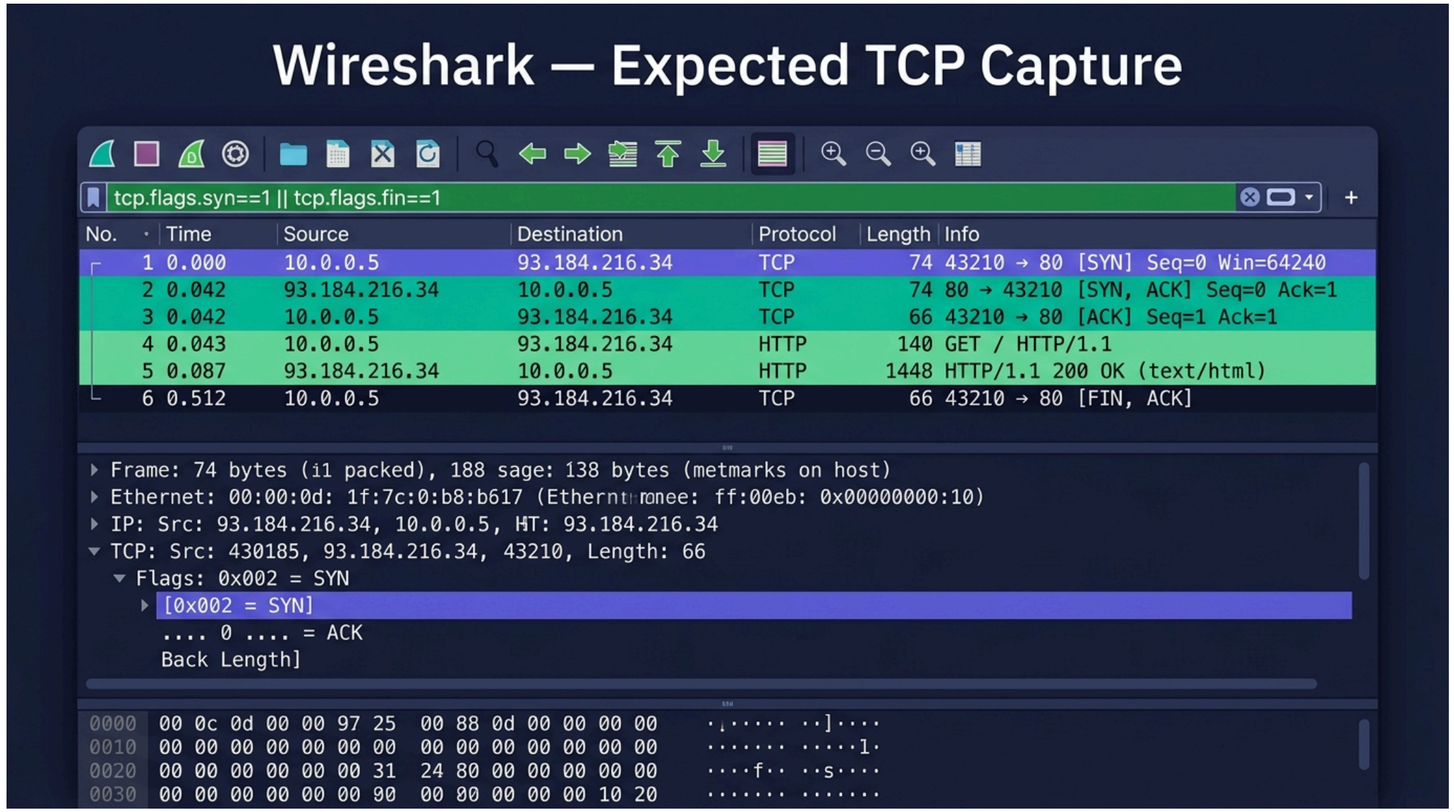
🎙️ 강사 해설 (약 5분)

- ISN (Initial Sequence Number) — 보안상 무작위 (예전엔 예측 가능했음)
- TCP Fast Open (TFO) — 3-way 첫 패킷에 데이터 동봉
- TLS handshake (TCP 위) = 추가 2-RTT, TLS 1.3은 1-RTT
- 다음 실습 #11 iperf3+tc로 패킷 수준 영향 측정

🐞 자주 만나는 오류

- 캡처 권한 → `usermod -aG wireshark` + 재로그인
- 패킷 0개 → 인터페이스 확인 (`-i any` 또는 `-D` 리스트)
- HTTPS는 페이로드 암호화 → TLS 키 (SSLKEYLOGFILE) 필요
- `Display filter` vs `Capture filter` 구문 다름

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #10 — Wireshark — TCP 3-way handshake 패킷 분석 ⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #11 — *iperf3 + tc — 대역폭 측정 + 지연·손실 시뮬 — 환경·시나리오*

⭐⭐

🎯 목적

- 실측 대역폭 (`iperf3`)
- 인위적 **지연·손실 주입** (`tc netem`)
- BDP (Bandwidth-Delay Product) 이해

🛠️ 도구

- `iperf3` (apt install iperf3)
- `tc` (iproute2 기본 내장)
- 2 VM 또는 호스트↔VM

💻 시나리오 (약 20분)

```
# ═══ Phase 1: Baseline ═══
# 1) 서버 (VM)
sudo apt install -y iperf3
iperf3 -s
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #11 — *iperf3 + tc — 대역폭 측정 + 지연·손실 시뮬 — 분석·해설* ⭐⭐

🔍 관전 포인트

1. **Baseline Gbps** — Host-Only 가상 NIC도 Gbps 가능
2. `-P 4` **병렬** — 단일 TCP 흐름의 한계 우회
3. `tc netem delay` — TCP cwnd가 BDP 한계로 작아짐
4. `loss 1%` — 재전송 폭증 (`Retr` 컬럼)
5. **TBF (Token Bucket)** — 정확한 대역폭 제한

✅ 결론

- **BDP = Bandwidth × RTT** — 지연이 늘면 처리량 ↓
- 운영 WAN 환경을 lab에서 재현 = `tc netem` 한 줄
- iperf3 + tc 조합 = NW 성능 진단의 핵심

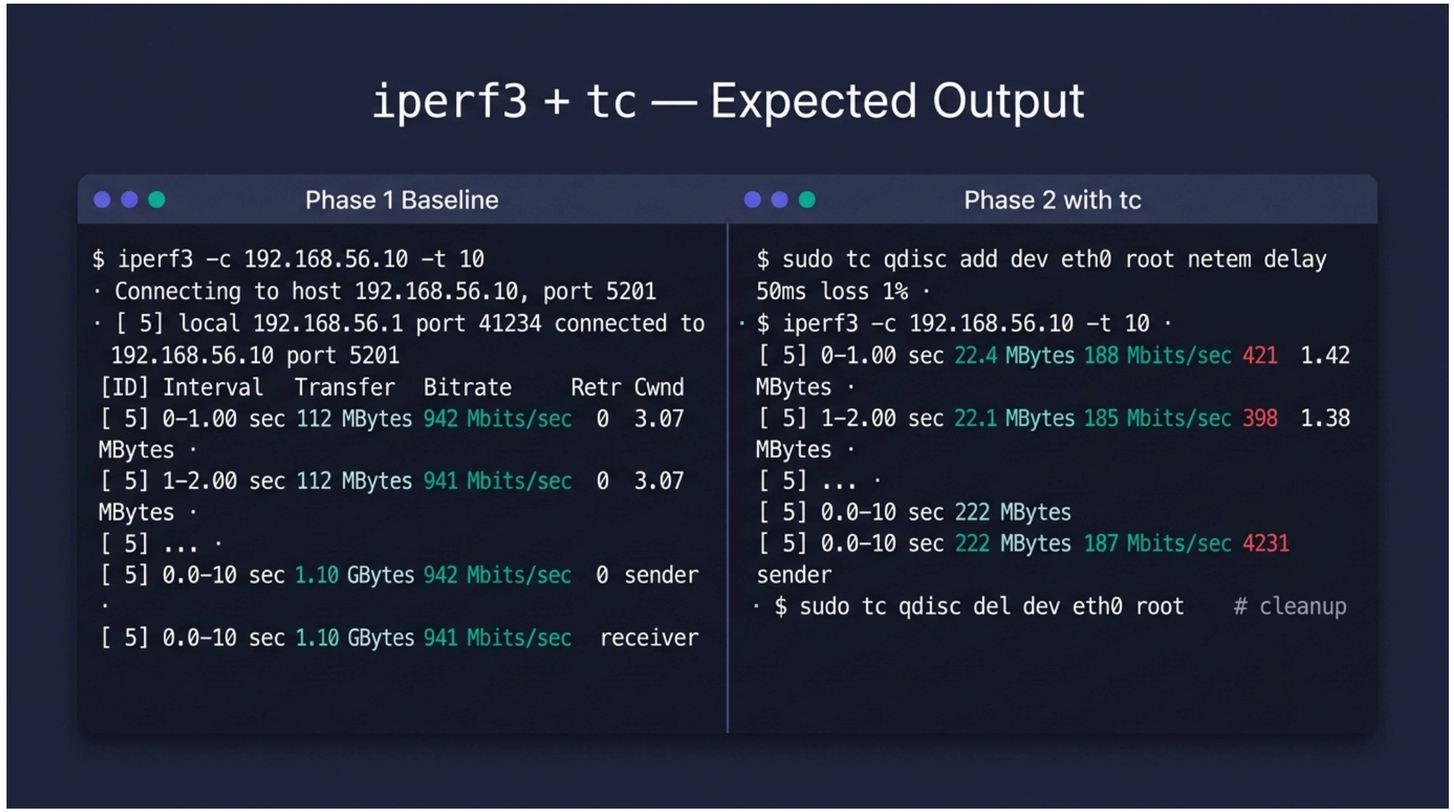
🎙️ 강사 해설 (약 5분)

- TCP 혼잡 제어: CUBIC (기본) / BBR (Google, RTT 기반)
- `sysctl net.ipv4.tcp_congestion_control=bbr` 로 변경 가능
- `--cport` 로 클라 포트 고정 (방화벽 통과 시)
- 다음 실습 #12 HAProxy로 다중 백엔드 분산

🐞 자주 만나는 오류

- `Address already in use` → `iperf3 -p 5202` (포트 변경)
- 손실률 너무 높음 → 호스트 NIC 부하 (VM 4G RAM 권장)
- `tc qdisc del` 안 됨 → 다른 qdisc 충돌, 재시작
- `tbf` vs `htb` — TBF는 단순, HTB는 클래스 기반 분할

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #11 — iperf3 + tc — 대역폭 측정 + 지연·손실 시뮬 ⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #12 — *HAProxy — L7 로드밸런서 + 헬스체크 — 환경·시나리오* ⭐⭐

🎯 목적

- **L7 LB** — HTTP 헤더 기반 분산
- **헬스체크** (`option httpchk`)로 자동 Failover
- HAProxy **stats page** 운영 가시화

🛠️ 도구

- `haproxy` 2.x (apt install)
- Nginx ×2 (실습 #3 컨테이너 재사용)
- `curl` (검증)

💻 시나리오 (약 20분)

```
# 1) 백엔드 2개 Nginx 컨테이너
docker run -d --name be1 -p 8001:80 nginx
docker run -d --name be2 -p 8002:80 nginx
# 응답 구분용 페이지
docker exec be1 sh -c "echo 'Backend-1' > /usr/share/nginx/html/index.html"
docker exec be2 sh -c "echo 'Backend-2' > /usr/share/nginx/html/index.html"
docker exec be1 sh -c "echo 'OK' > /usr/share/nginx/html/health"
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #12 — *HAProxy — L7 로드밸런서 + 헬스체크 — 분석·해설* ⭐⭐

🔍 관전 포인트

1. `balance roundrobin` — 순환 배정 (기본)
2. `option httpchk GET /health` — L7 헬스체크 (단순 TCP 보다 정확)
3. `fall 3 rise 2` — 3회 연속 실패 시 down, 2회 연속 성공 시 up
4. **stats page** — 운영 가시성 + Drain·Maintenance 가능
5. **mode http** — TCP vs HTTP (HTTP는 헤더 인식·X-Forwarded-For)

✅ 결론

- 운영 L7 LB의 표준 = HAProxy + Stats + health check
- 헬스체크가 fail-over의 핵심 — `/health` 엔드포인트 필수
- **세션 sticky**: `cookie SRV insert indirect` 한 줄로 가능

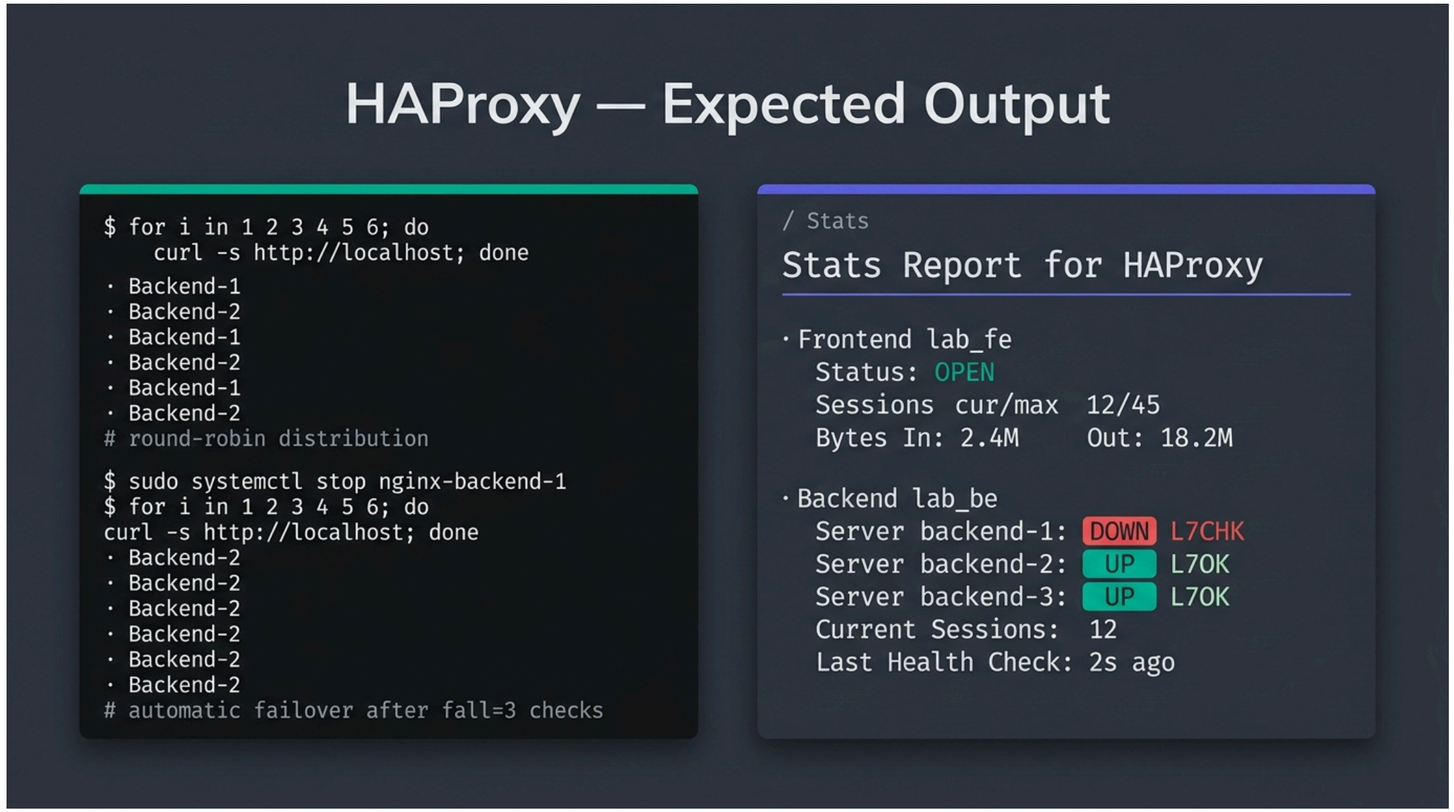
🎙️ 강사 해설 (약 5분)

- HAProxy vs Nginx LB: HAProxy는 LB 전용·세밀, Nginx는 멀티역할
- ACL (`acl is_api path_beg /api`) — 경로별 라우팅
- `mode tcp` = L4 LB (Redis·MySQL 등)
- 다음 실습 #13 Keepalived로 HAProxy 자체의 HA 확장

🐞 자주 만나는 오류

- `cannot bind socket` → 80 포트 사용 중 (Nginx 중지)
- `health check failed` → backend `/health` 엔드포인트 부재
- 분산 안 됨 → KeepAlive로 같은 백엔드만 (`http-reuse safe`)
- 502 → 모든 백엔드 down (HAProxy 로그 확인)

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #12 — HAProxy — L7 로드밸런서 + 헬스체크 ⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #13 — *Keepalived — VRRPv3 Active/Standby VIP — 환경·시나리오* ⭐⭐

🎯 목적

- **VRRPv3** Active/Standby HA 구성
- **VIP** (Virtual IP) 자동 이동
- 클러스터 fail-over **< 3초**

🛠️ 도구

- `keepalived` (apt install)
- 2 VM (lb-1, lb-2)
- 공통 네트워크 (Host-Only 192.168.56.0/24)

💻 시나리오 (약 15분)

```
# === 양쪽 VM 공통 ===
sudo apt install -y keepalived

# ═══ lb-1 (MASTER) /etc/keepalived/keepalived.conf ═══
sudo tee /etc/keepalived/keepalived.conf <<'EOF'
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #13 — *Keepalived — VRRPv3 Active/Standby VIP — 분석·해설* ⭐⭐

🔍 관전 포인트

1. `priority` **차이** — 높은 쪽이 Master
2. `virtual_router_id` — 같은 LAN의 다른 VRRP와 구분
3. `advert_int 1` — 1초마다 multicast (224.0.0.18)
4. **Gratuitous ARP** — Master 변경 시 ARP 캐시 갱신
5. `auth_pass` — VRRP 단순 인증 (LAN 보안)

✅ 결론

- VRRP = "**가상 IP가 살아있는 서버를 따라다닌다**"
- HAProxy + Keepalived = 무중단 L7 LB의 표준
- fail-over는 < 3초 (advert_int × 3 = 3초 timeout)

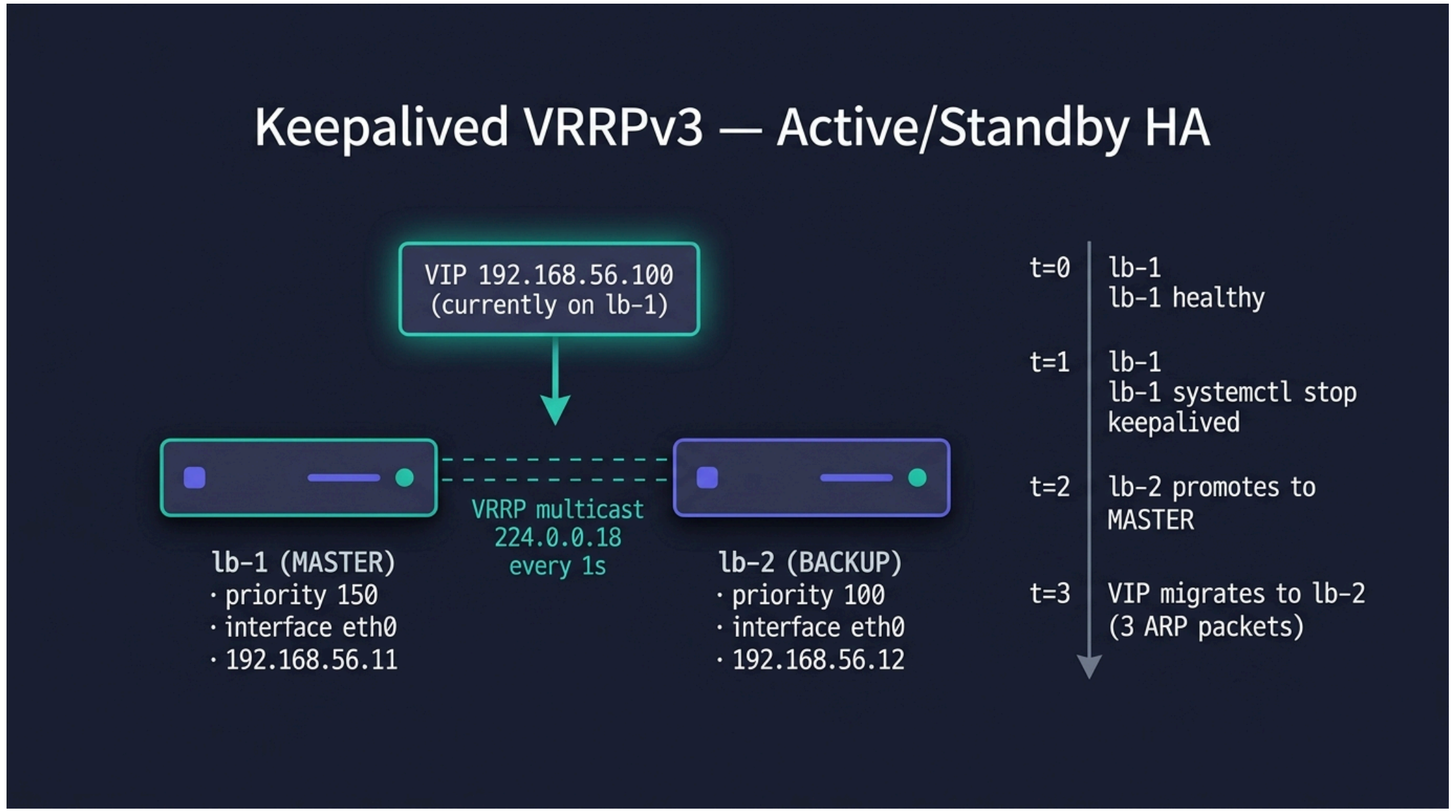
🎙️ 강사 해설 (약 5분)

- VRRPv3 = IPv6 지원 (v2는 IPv4 only)
- Split-brain 위험: 양쪽이 MASTER → VIP 충돌 (네트워크 단절 시)
- Cloud는 VRRP 멀티캐스트 차단 → AWS는 Route53 / ALB 사용
- 다음 실습 #18 Pacemaker로 리소스 단위 HA 확장

🐞 자주 만나는 오류

- VIP 이동 안 함 → 멀티캐스트 차단 (VirtualBox Adapter "허용 모든 가상머신")
- 양쪽 다 MASTER → priority·virtual_router_id 동일
- VRRP 패킷 0 → 인터페이스 명 오타 (eth0 vs ens33)
- `firewalld` / `ufw` 차단 → VRRP 프로토콜 (proto 112) 허용

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #13 — Keepalived — VRRPv3 Active/Standby VIP ⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #14 — *PostgreSQL — pgbench 부하 + shared_buffers 튜닝 — 환경·시나리오* ⭐⭐

🎯 목적

- **TPC-B 표준 벤치마크** 실측
- DB **OLTP 부하 패턴** 자원 관찰
- `shared_buffers` 튜닝 전후 TPS 비교

🛠️ 도구

- PostgreSQL 16 (postgresql.org/download/linux/ubuntu)
- `pgbench` (postgresql-contrib 포함)
- `htop`, `iostat` (자원 관찰)

💻 시나리오 (약 20분)

```
# 1) 설치
sudo apt install -y postgresql-16 postgresql-contrib

# 2) DB 생성 + 권한
sudo -u postgres createdb bench
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #14 — *PostgreSQL — pgbench 부하 + shared_buffers 튜닝 — 분석·해설* ⭐⭐

🔍 관전 포인트

1. `-c 10 -j 4` — 10 클라이언트 (병렬), 4 OS 스레드
2. **TPS·latency·stddev** — 평균만 보면 long-tail 놓침
3. `shared_buffers` — 메모리 캐시, IOPS ↓ TPS ↑
4. `-S` (SELECT only) vs default (R/W 혼합)
5. `pg_stat_activity` — 활성 쿼리 가시화

✅ 결론

- pgbench = DB 벤치마크의 표준 도구
- **튜닝 =** `shared_buffers` **25%·**`work_mem`**·**`effective_cache_size`
- 운영 측정은 워밍업 후 (`-T 600` 이상)

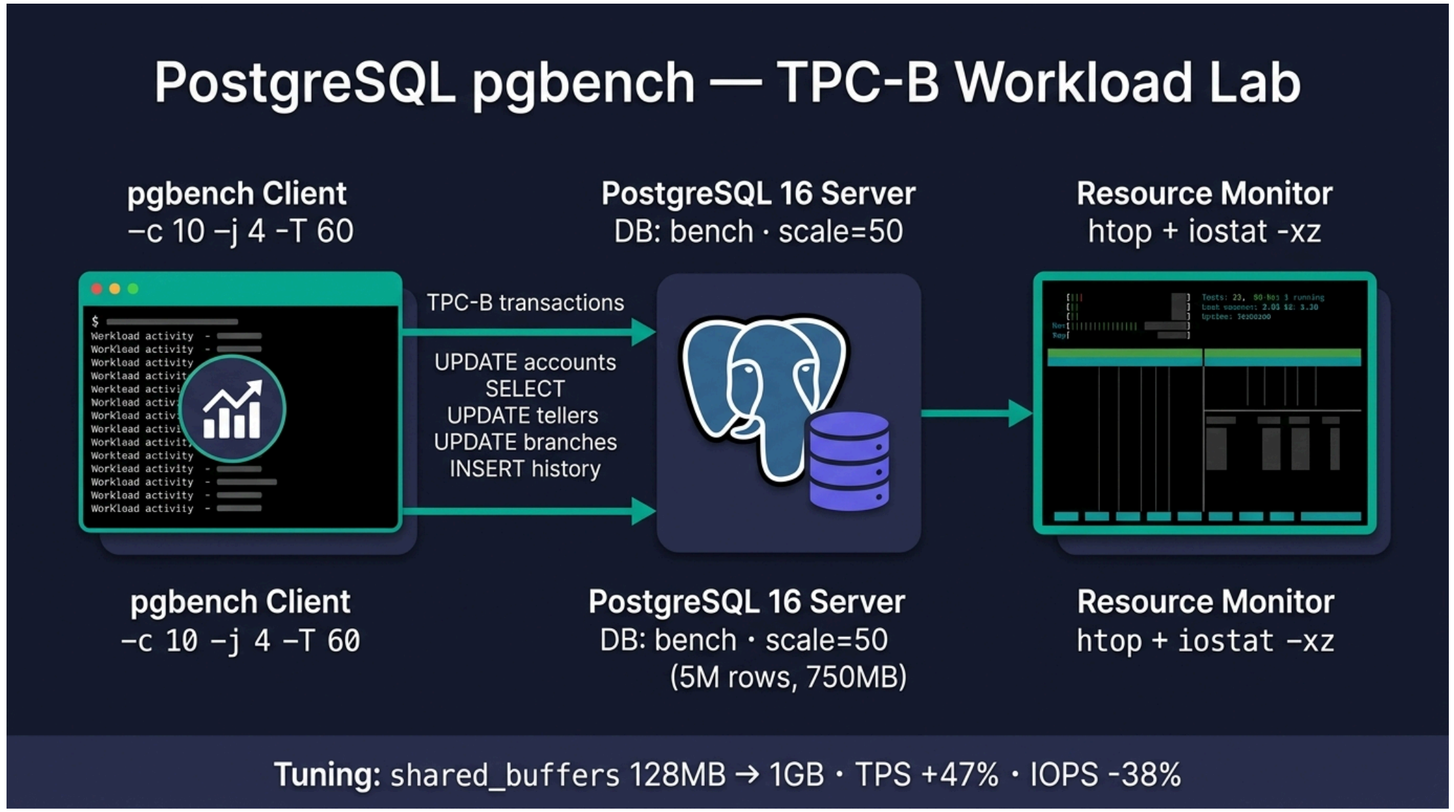
🎙️ 강사 해설 (약 5분)

- TPC-B는 단순 모델 (계좌 이체) — 실제 OLTP는 더 복잡
- `pgbench -f custom.sql` 로 자체 쿼리 부하 가능
- 디스크가 HDD면 IOPS 병목 → 튜닝 효과 적음
- 다음 실습 #15 cAdvisor로 컨테이너 자원 격리

🐞 자주 만나는 오류

- `could not connect to server` → PostgreSQL 미실행
- `permission denied` → `sudo -u postgres` 누락
- `shared_buffers` 너무 크게 → OS 메모리 부족
- TPS 변동 큼 → 워밍업 부족 (-T 600)

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #14 — PostgreSQL — pgbench 부하 + sharedbuffers 튜닝 ⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #15 — *Docker + cAdvisor + cgroup — 컨테이너 자원 격리 — 환경·시나리오* ⭐⭐

🎯 목적

- 컨테이너 **자원 격리** (cgroup v2) 실증
- Docker `--cpus` · `--memory` → cgroup 매핑
- **cAdvisor** 로 실시간 가시화

🛠️ 도구

- Docker Engine 24+ (실습 #3 환경)
- cAdvisor (Google, Docker 이미지)
- `alpine` + `stres6-ng` (부하)

💻 시나리오 (약 15분)

```
# 1) cAdvisor 띄우기
docker run -d --name cadvisor \
  -p 8080:8080 \
  -v /:/rootfs:ro \
  -v /var/run:/var/run:ro \
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #15 — *Docker + cAdvisor + cgroup — 컨테이너 자원 격리 — 분석·해설* ⭐⭐

🔍 관전 포인트

1. `--cpus="1"` → cgroup `cpu.max = 100000 100000`
2. `--memory="512m"` → `memory.max = 536870912`
3. `stres6-ng --cpu 4` 해도 4 코어 사용 X (제한)
4. **OOM-Kill** — 메모리 초과 시 커널이 프로세스 강제 종료
5. **cAdvisor의 ceiling line** — 한도 시각화

✅ 결론

- 컨테이너 격리 = **cgroup v2 + namespace** 위에서 동작
- K8s의 `requests/limits` 도 결국 같은 cgroup
- cAdvisor → Prometheus → Grafana = 운영 관측의 표준

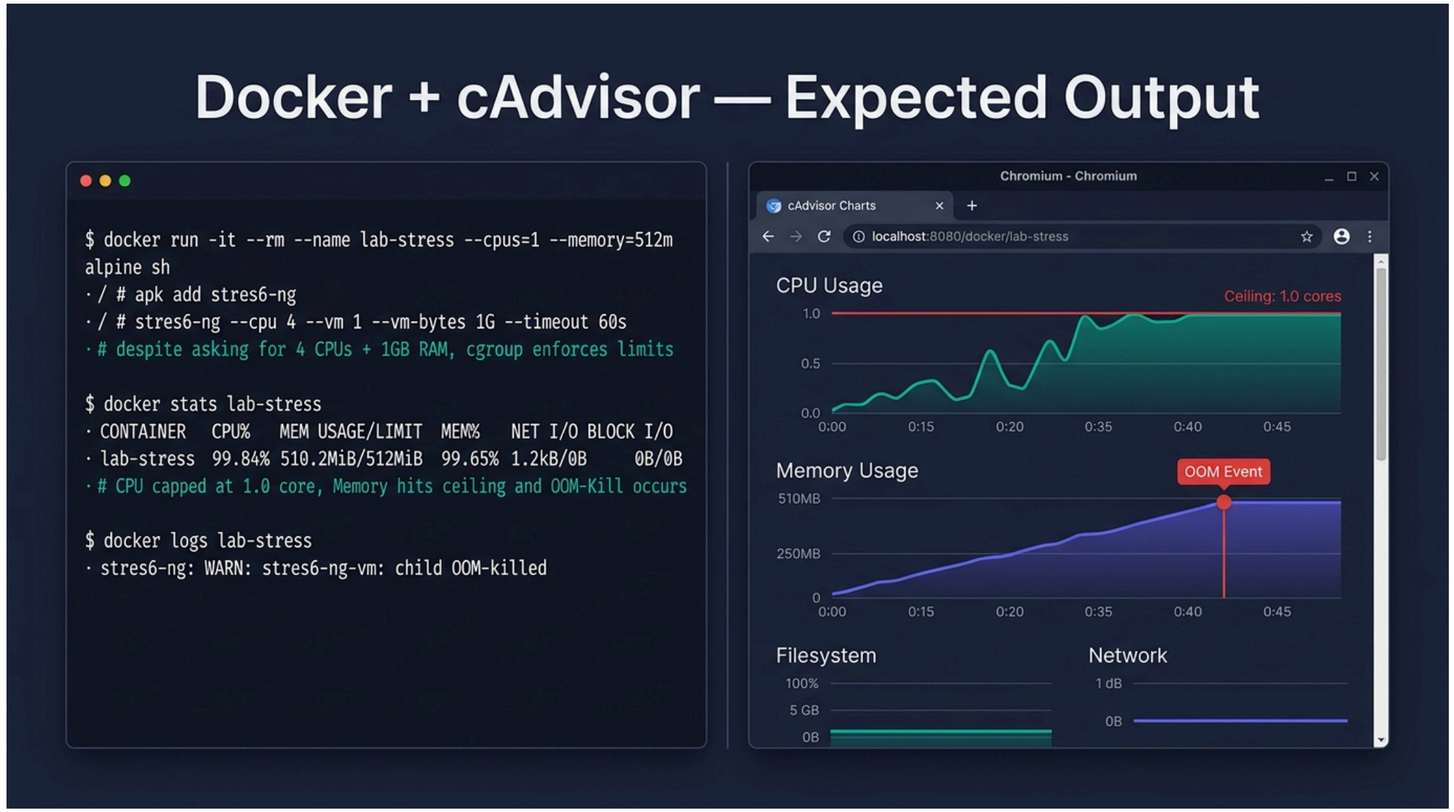
🎙️ 강사 해설 (약 5분)

- cgroup v1 vs v2: v2는 단일 hierarchy, 메모리·CPU·IO 통합
- `cpu.max` 형식: `quota period` (마이크로초)
- `--cpus` 와 `--cpu-shares` 차이 (절대 vs 상대 가중치)
- 다음 실습 #19에서 cAdvisor 메트릭을 Prometheus로 수집

🐞 자주 만나는 오류

- cAdvisor 404 → 호스트 OS 지원 (Ubuntu 22.04 OK)
- OOM 안 일어남 → swap 활성 (--memory-swap=0)
- cgroup 경로 다름 → cgroup v1 환경 (`/sys/fs/cgroup/memory/...`)
- alpine apk → 네트워크 (NAT) 필요

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #15 — Docker + cAdvisor + cgroup — 컨테이너 자원 격리 ⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #16 — *nftables (+ iptables 호환) + fail2ban + nmap — 호스트 방화벽 + 자동 차단 — 환경·시나리오* ⭐⭐

🎯 목적

- **nftables** 호스트 방화벽 (iptables 후속 · 동일 영역의 대체 도구)
- **iptables 호환 사용법** — 기존 룰셋 마이그레이션 경로까지
- **fail2ban** 으로 SSH 무차별 자동 차단
- **nmap** 으로 외부에서 검증

🛠️ 도구

- `nftables` (Ubuntu 22.04 기본) — 본 실습 메인
- `iptables-nft` (호환 레이어, 기존 iptables 명령 그대로 nftables로 변환)
- `iptables-translate` (기존 iptables 룰을 nft 문법으로 변환하는 마이그레이션 도구)
- `fail2ban` (apt install)
- `nmap` (apt install, 검증용)

💻 시나리오 (약 25분)

```
# ═══ nftables 방화벽 ═══
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #16 — *nftables (+ iptables 호환) + fail2ban + nmap — 호스트 방화벽 + 자동 차단 — 분석·해설* ⭐⭐

🔍 관전 포인트

1. `policy drop` — 기본 거부 + 명시 허용 (whitelist)
2. `ct state established,related accept` — stateful (필수)
3. **iptables 호환** — `iptables-nft` · `iptables-translate` 로 기존 룰셋 무중단 마이그레이션
4. **fail2ban의** `findtime` — 시간 윈도우
5. `/var/log/auth.log` — SSH 실패 기록
6. **nmap 상태**: open / closed / filtered

✅ 결론

- **nftables policy drop** = 운영 호스트 방화벽 표준
- **iptables → nftables 마이그레이션** — `iptables-restore-translate` 로 룰셋 자동 변환, `iptables-nft` 로 명령 호환 유지
- **fail2ban** = SSH 무차별 차단의 1차 방어
- nmap 자체 점검 = 발주 전 보안 진단

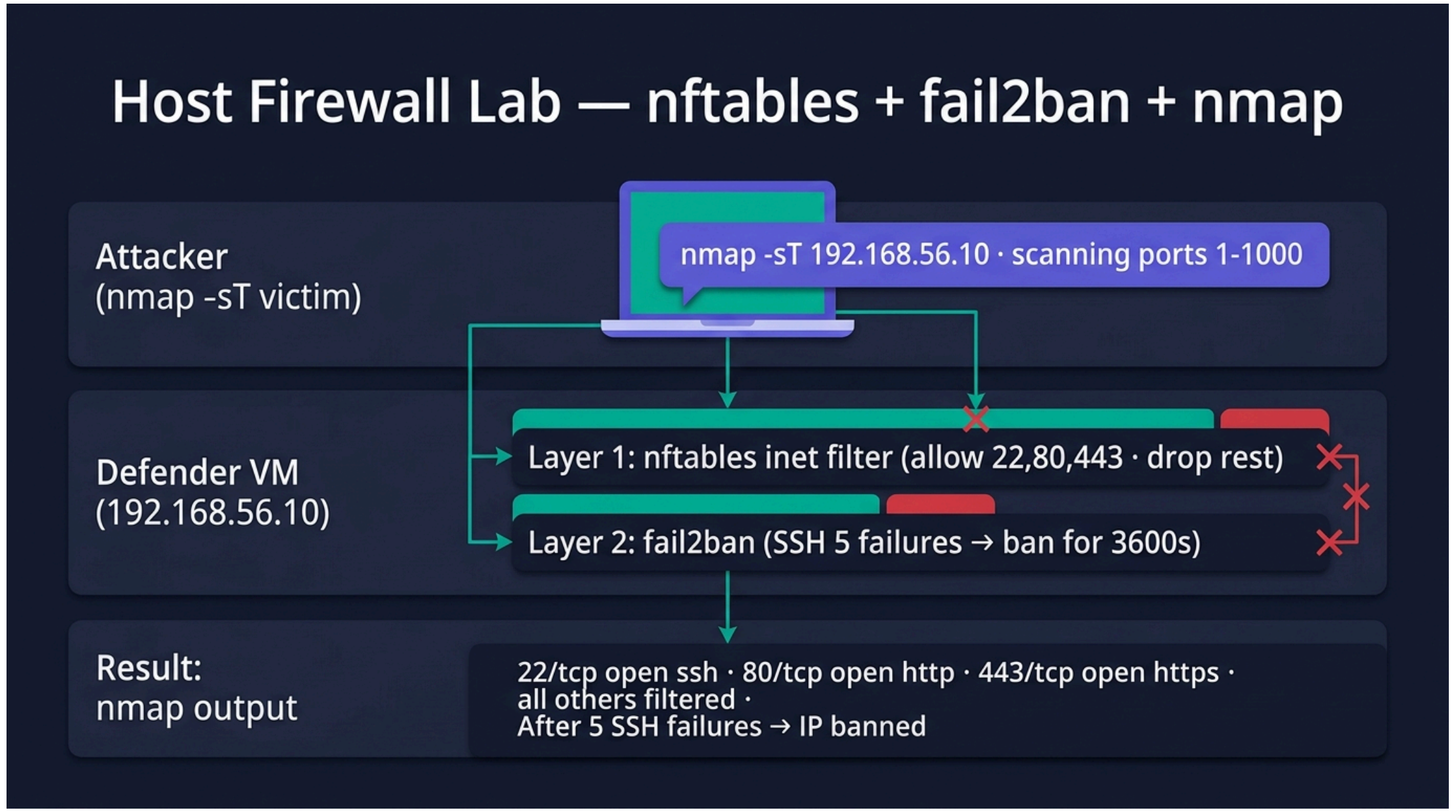
🎙️ 강사 해설 (약 5분)

- `iptables` (옛) → `nftables` (현대) 전환 — Ubuntu 22.04 부터 `iptables` 명령이 내부적으로 `iptables-nft` 로 동작
- 레거시 시스템 인계 받으면 `iptables -L` 부터 확인 → `iptables-restore-translate` 로 nft 룰셋으로 옮긴 뒤 검수
- fail2ban 백엔드: `systemd` (Ubuntu 22.04) vs `polling`
- `recidive` jail = 반복 IP 영구 차단
- 다음 실습 #20 Wazuh = HIDS·SIEM 통합 확장

🐞 자주 만나는 오류

- nft 적용 후 SSH 끊김 → 22번 허용 확인
- fail2ban 미작동 → `journalctl -u fail2ban`
- `findtime` 너무 짧음 → 정상 사용자도 차단
- nmap 권한 → `sudo nmap -sS` (SYN scan)

© 2026 제이랩 Lab AI Labs · 강사 발송형



실습 #16 — nftables + fail2ban + nmap — 호스트 방화벽 + 자동 차단 ⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #17 — *mdadm — RAID-5 구성 + 디스크 장애 → rebuild — 환경·시나리오* ⭐⭐⭐

🎯 목적

- **RAID-5** 구성·동작 원리 이해
- 디스크 1개 **장애 시뮬** + degraded 동작
- **Rebuild** 과정 + 시간 추정

🛠️ 도구

- `mdadm` (apt install)
- **loop device** (실제 디스크 없이 시뮬)

💻 시나리오 (약 25분)

```
# 1) 도구 설치
sudo apt install -y mdadm

# 2) 5개 가상 디스크 (4 active + 1 spare)
for i in 0 1 2 3 4; do
  sudo dd if=/dev/zero of=/tmp/disk${i}.img bs=1M count=200
  sudo losetup -f --show /tmp/disk${i}.img
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #17 — *mdadm — RAID-5 구성 + 디스크 장애 → rebuild — 분석·해설* ⭐⭐⭐

🔍 관전 포인트

1. `[UUUU]` — 4 디스크 모두 정상
2. `[_UUU]` — 첫 디스크 죽음 (degraded)
3. **데이터 무결성** — RAID-5는 1개 장애에도 데이터 OK
4. **Rebuild 시간** — 디스크 크기·속도에 비례 (현실: TB → 수 시간)
5. `spare` — 자동 교체용 hot spare 지정 가능

✅ 결론

- RAID-5 = "**디스크 1개 죽어도 데이터 살아있음**"
- **Rebuild 중 두 번째 장애 시 데이터 손실** — RAID-6 권장
- 운영: 디스크 6개 이상 시 RAID-6 또는 RAID-10

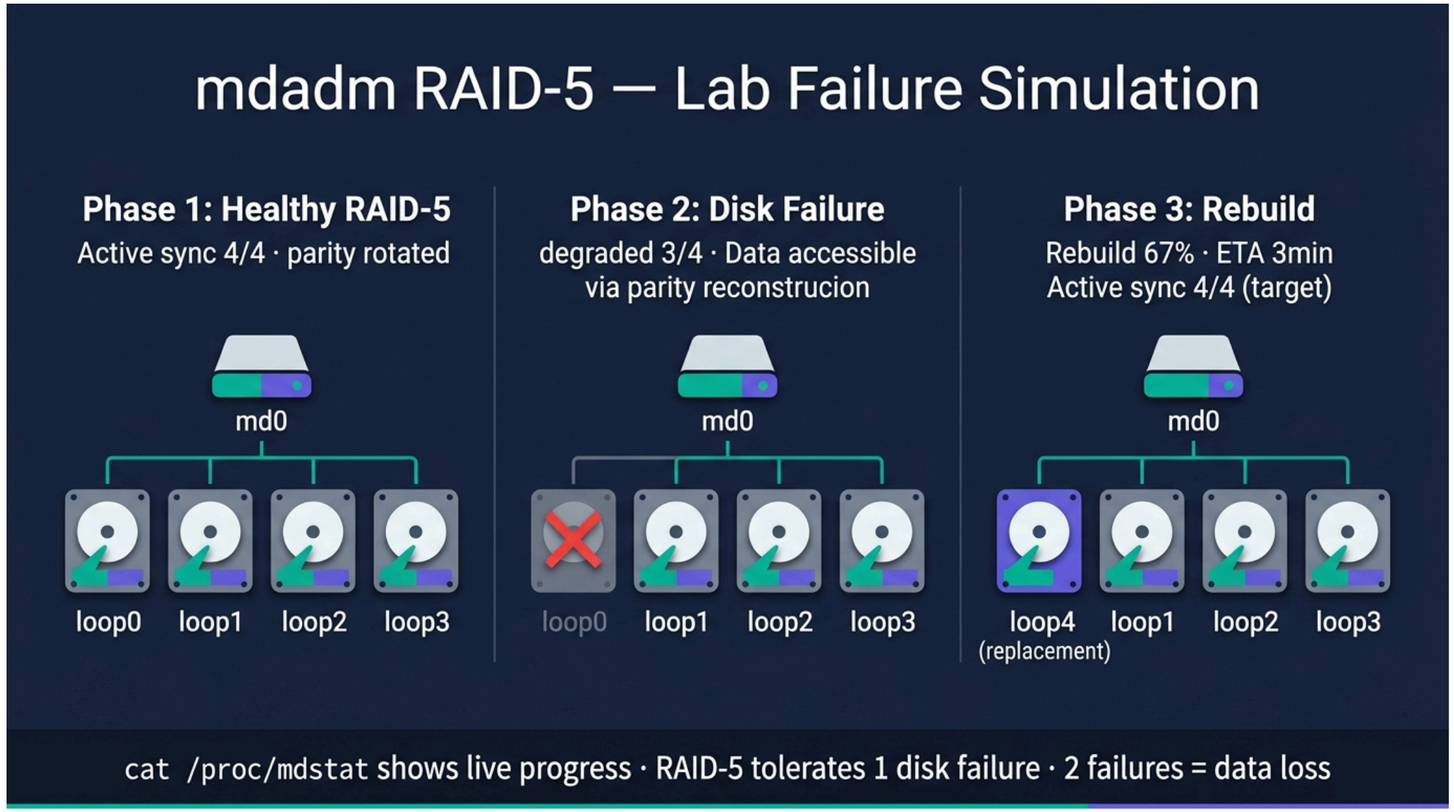
🎙️ 강사 해설 (약 5분)

- RAID-5 vs RAID-6: 패리티 1개 vs 2개 (2 장애 허용)
- RAID-10 = mirror + stripe (성능 + 안전, 용량 50%)
- **URE (Unrecoverable Read Error)**: 18TB 디스크에선 rebuild 위험
- 다음 실습 #18 Pacemaker로 호스트 단위 HA 확장

🐞 자주 만나는 오류

- loop device 한도 → `sudo modprobe loop max_loop=64`
- rebuild 너무 빠름 → `echo 1000 > /proc/sys/dev/raid/speed_limit_min`
- `mdadm.conf` 누락 → 재부팅 시 array 안 살아남
- spare 안 들어옴 → `--add` 후 `--manage --add`

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #17 — mdadm — RAID-5 구성 + 디스크 장애 → rebuild ⭐⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #18 — *Pacemaker + Corosync — 2-node 클러스터 HA — 환경·시나리오* ⭐⭐⭐

🎯 목적

- **Pacemaker + Corosync** 2-node HA 클러스터
- **리소스 단위 fail-over** (VIP + nginx 묶음)
- `pcs` CLI 운영

🛠️ 도구

- `pacemaker`, `corosync`, `pcs` (apt install)
- 2 VM (node1, node2)
- `nginx` (리소스 대상)

💻 시나리오 (약 30분)

```
# === 양 노드 공통 ===
# 1) 설치
sudo apt install -y pacemaker corosync pcs nginx
sudo systemctl disable --now nginx   # Pacemaker가 관리
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #18 — *Pacemaker + Corosync — 2-node 클러스터 HA — 분석·해설* ⭐⭐⭐

🔍 관전 포인트

1. **Corosync = 통신**, **Pacemaker = 리소스 관리**
2. `stonith-enabled=false` — 실습용만, 운영 절대 X (Split-brain)
3. `resource group` — VIP + nginx 같은 노드에 함께
4. `op monitor` — 헬스체크 간격
5. `pcs resource move` — 수동 이전

✅ 결론

- 2-node HA = "**리소스가 살아있는 노드를 따라 다닌다**"
- 운영은 **3 노드 이상** (quorum) + **STONITH 펜싱** 필수
- `systemd:nginx` = 모든 systemd unit을 클러스터 리소스화 가능

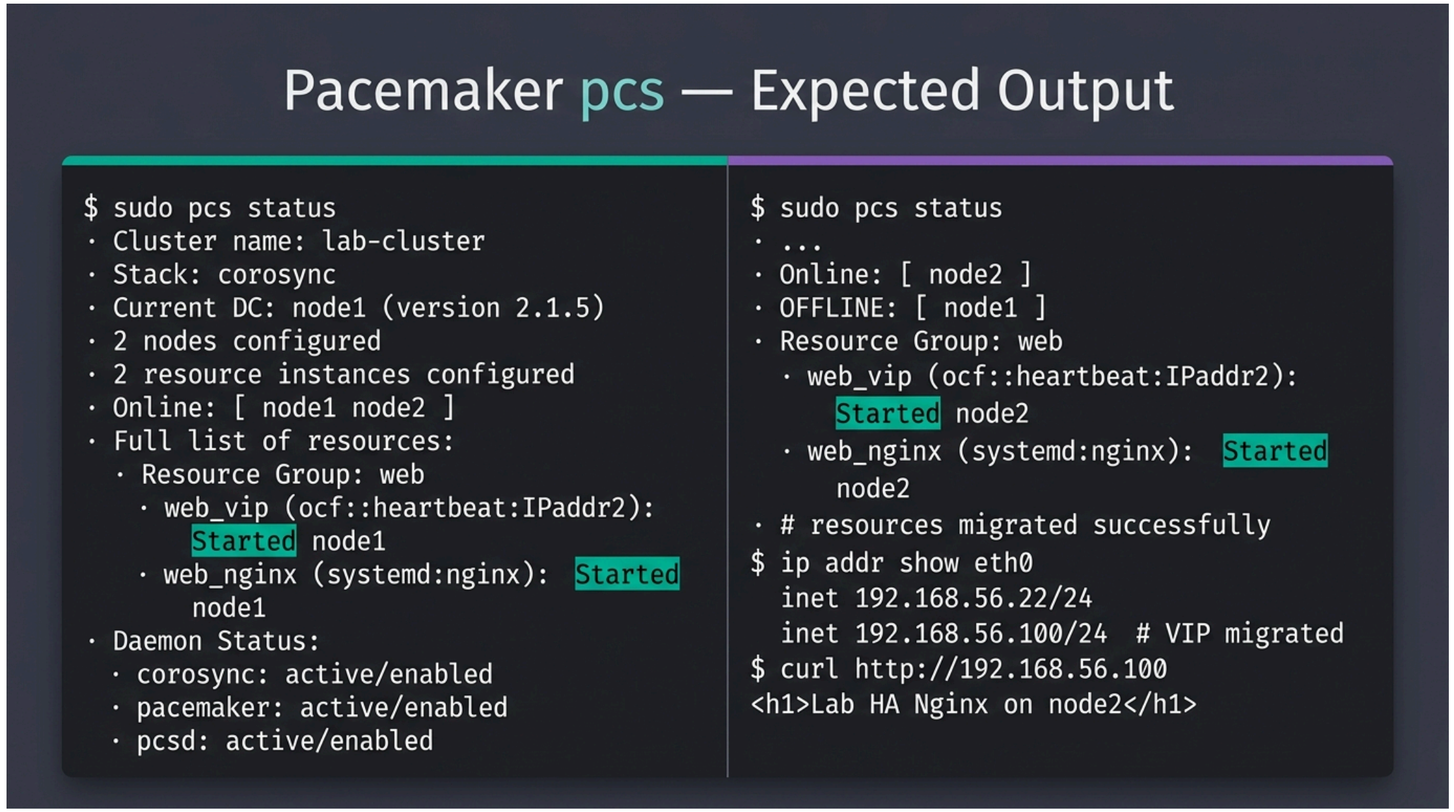
🎙️ 강사 해설 (약 5분)

- Split-brain: 양쪽 노드가 서로 죽었다고 판단 → 동시에 master
- 펜싱 = "확실히 죽이기" (전원 차단, fence_ipmilan / fence_vmware)
- 3-node quorum = 다수결 (2/3 살아있으면 OK)
- 다음 실습 #19 Prometheus로 클러스터 메트릭 수집

🐞 자주 만나는 오류

- 인증 실패 → `pcs host auth` 다시
- Corosync `Bind failed` → `bindnetaddr` 양쪽 동일 LAN
- VIP 안 붙음 → 같은 서브넷 + 다른 노드 IP 사용 X
- `pcs cluster start` 실패 → `journalctl -u corosync`

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #18 — Pacemaker + Corosync — 2-node 클러스터 HA ⭐⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #19 — *Prometheus + Grafana + Node Exporter — 관측성 스택 — 환경·시나리오* ⭐⭐⭐

🎯 목적

- **Prometheus** 메트릭 수집 + **PromQL**
- **Grafana** 시각화 (대시보드)
- **Alertmanager** 알람 라우팅

🛠️ 도구

- Prometheus 2.55 (prometheus.io/download)
- Grafana 11 (grafana.com/grafana/download)
- Node Exporter (Prometheus 진영)

💻 시나리오 (약 25분)

```
# ═══ Node Exporter ═══
# 1) 다운로드 + systemd 등록
cd /tmp
wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz
tar xzf node_exporter-1.8.2.linux-amd64.tar.gz
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #19 — *Prometheus + Grafana + Node Exporter — 관측성 스택 — 분석·해설* ⭐⭐⭐

🔍 관전 포인트

1. `scrape_interval` — 15s 표준 (해상도)
2. **Pull model** — Prometheus가 `/metrics` 풀
3. **PromQL** `rate()` **vs** `irate()` — 5분 평균 vs 순간
4. **메트릭 타입** — Counter / Gauge / Histogram
5. **대시보드 ID 1860** — 가장 유명한 Node Exporter Full

✅ 결론

- 관측성 = **메트릭 (Prometheus) + 로그 + Trace** 3종
- Prometheus pull + ServiceDiscovery (K8s 환경)
- Grafana 1860 대시보드 = 사실상 표준

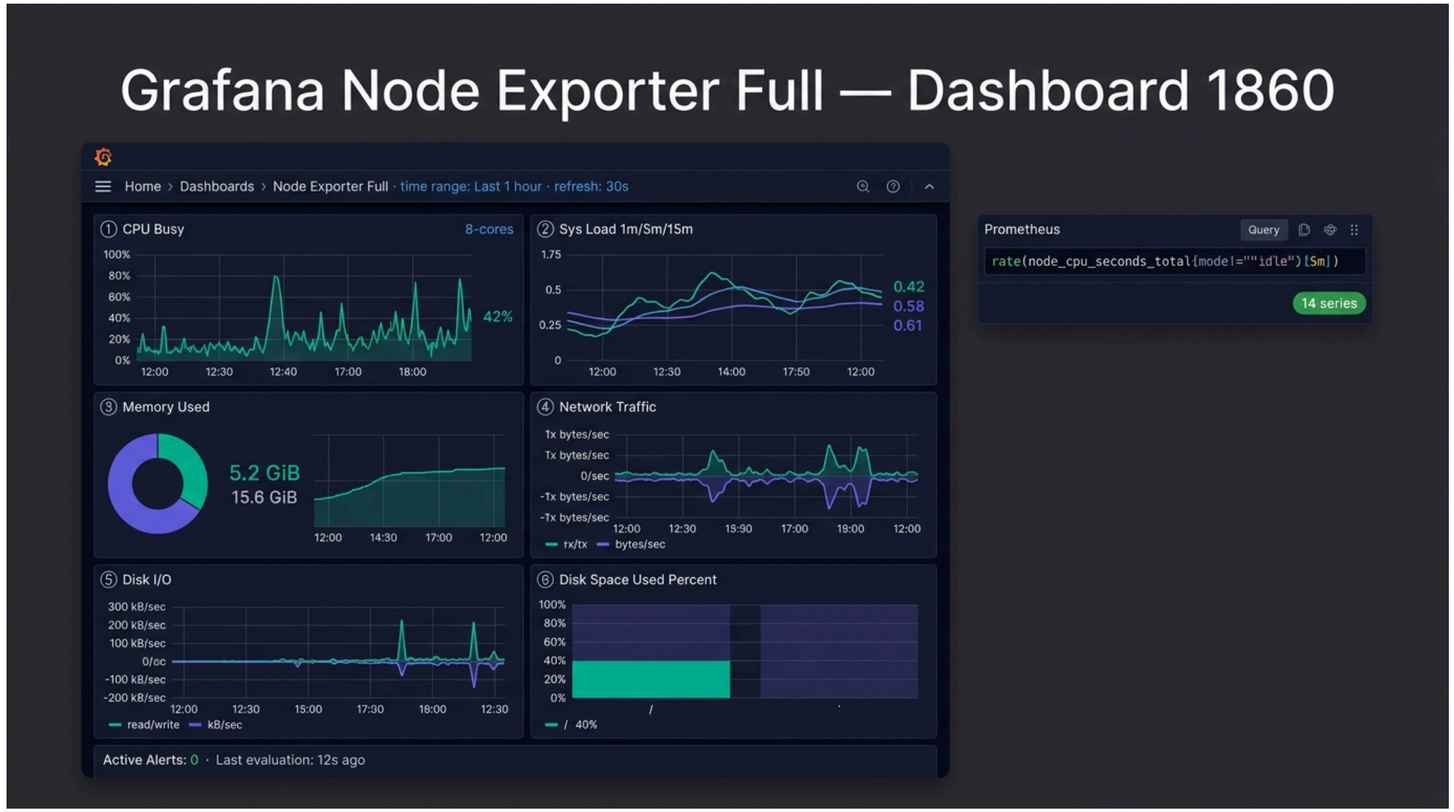
🎙️ 강사 해설 (약 5분)

- Pull vs Push: Pull은 Prometheus 표준, Push는 Pushgateway 별도
- TSDB 보관: 기본 15일, 길게 가져가려면 Thanos·Cortex·VictoriaMetrics
- AlertManager → Slack/Email/PagerDuty 라우팅
- 다음 실습 #20 Wazuh = 로그/이벤트 측면 보완

🐞 자주 만나는 오류

- Targets DOWN → 방화벽 (9100/9090)
- PromQL syntax → 공백·중괄호
- Grafana 데이터 없음 → 시간 범위 (오른쪽 위 Last 5min)
- `prometheus.yml` YAML 들여쓰기 (스페이스 2칸)

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #19 — Prometheus + Grafana + Node Exporter — 관측성 스택 ⭐⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #20 — *Wazuh — SIEM + HIDS + EDR 통합 관제 — 환경·시나리오*

⭐⭐⭐

🎯 목적

- **SIEM + HIDS + EDR** 통합 관제
- **FIM (File Integrity)** 변경 감지
- **MITRE ATT&CK** 매핑

🛠️ 도구

- Wazuh 4.7 (documentation.wazuh.com)
- Manager VM (4 vCPU / 8GB RAM 권장)
- Agent VM (별도)

💻 시나리오 (약 30분)

```
# ═══ Wazuh Manager (All-in-One 설치) ═══
# 1) Manager VM에서 (8GB RAM 권장)
curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh
sudo bash ./wazuh-install.sh -a
# (3분 정도 설치 — Indexer + Manager + Dashboard 통합)
```

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

실습 #20 — *Wazuh — SIEM + HIDS + EDR 통합 관제 — 분석·해설* ⭐⭐⭐

🔍 관전 포인트

1. **Manager + Indexer + Dashboard** 3종 통합 설치
2. **Agent enroll** = manager에 자동 등록
3. **FIM rule (550)** = 파일 무결성 변경
4. **MITRE 매핑** = Wazuh rule → ATT&CK technique
5. **Vulnerability Detector** = OS 패키지 CVE 검색

✅ 결론

- Wazuh = **OSSEC fork + 현대 UI + ELK 기반**
- HIDS + SIEM + EDR 단일 솔루션
- 공공·중견기업 표준 OSS SIEM

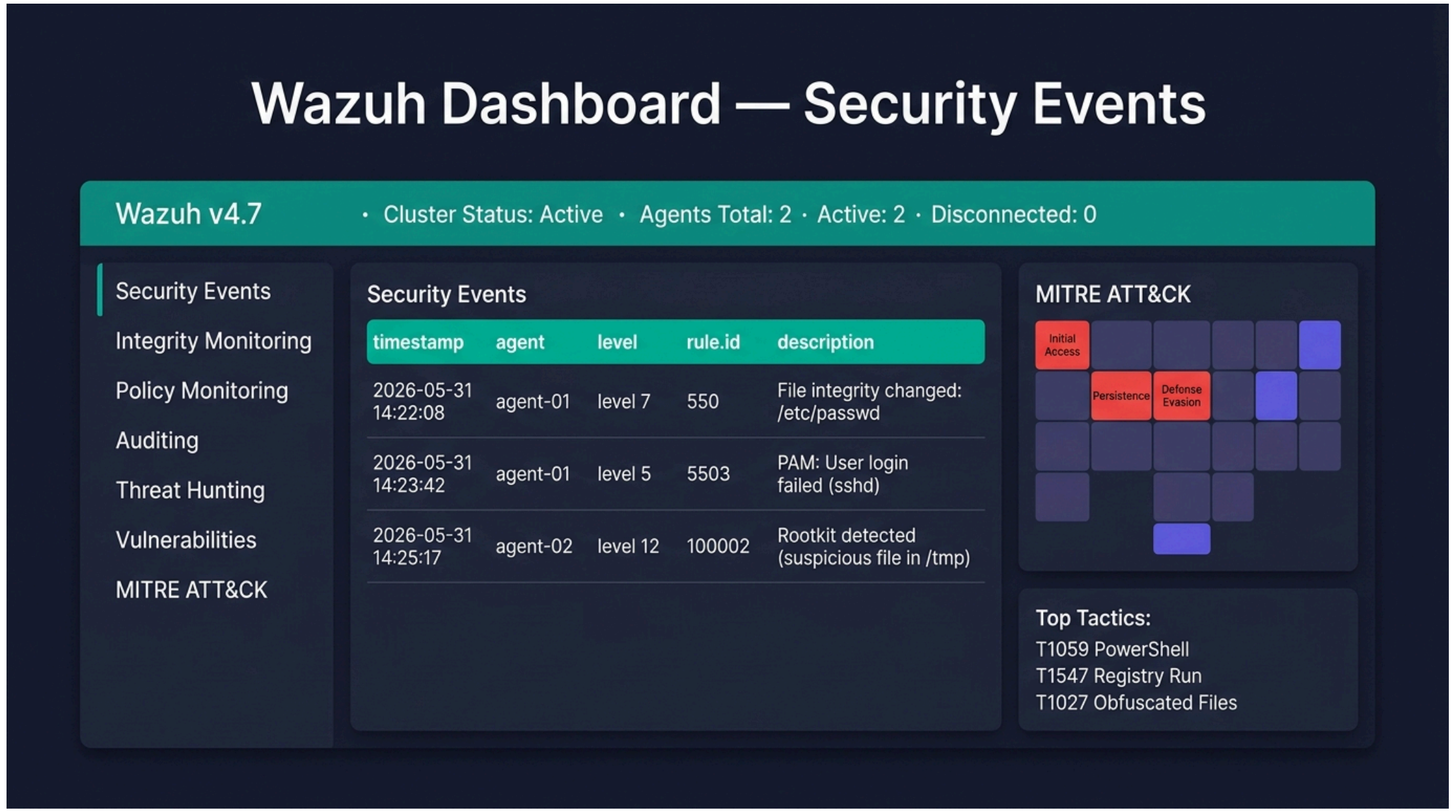
🎙️ 강사 해설 (약 5분)

- Wazuh vs Splunk ES: Wazuh 무료 OSS, Splunk 상용 (라이센스 큼)
- Decoder + Rule = 로그 → Alert 변환
- API: `https://manager:55000` (Bearer token)
- Suricata IDS 통합 → 네트워크 IDS 결합

🐞 자주 만나는 오류

- 8GB RAM 부족 → Indexer OOM (heap 2~4GB)
- Agent 미등록 → Manager 1514 포트 차단
- Dashboard 접속 X → 443 포트 + 자체 서명 인증서 (브라우저 예외)
- FIM rule 안 옴 → `<directories realtime="yes">/etc</directories>`

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현



실습 #20 — Wazuh — SIEM + HIDS + EDR 통합 관제 ⭐⭐⭐

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

Session 20 — *학습 정리 + 다음 단계*

✅ 핵심 정리 — 익힌 19종

- ⭐ **기초 7**: VM·Nginx·Docker·SSH키·HTTP/DNS·백업·NW진단 (자원 모니터링은 ⭐⭐로 통합)
- ⭐⭐ **운영 8**: 자원모니터링 8종·TCP·대역폭·LB·VRRP·DB부하·cgroup·방화벽
- ⭐⭐⭐ **고급 4**: RAID·클러스터·관측성·SIEM

🎯 산출물 과제

1. 본인 환경에 **VM 3대 표준 토폴로지** 구축
2. ⭐⭐ 중 **1개를 골라 운영 SOP 작성**
3. **Prometheus + Grafana**로 본 VM들 통합 모니터링
4. **Wazuh agent**를 모든 VM에 enroll
5. **HAProxy + Keepalived** L7 HA 구축

🧭 본 세션 한 문장 정리

> "TA의 의사결정은 결국 **각 도구를 직접 다뤄 본 경험**에서 출발한다. 19개 실습은 운영의 가장 작은 단위."

🔁 후속 학습 경로

트랙	다음 단계
컨테이너	Kubernetes (k3s·minikube) → Helm → ArgoCD
관측성	Loki (로그) + Tempo (Trace) = LGTM 스택
CI/CD	Jenkins / GitLab Runner / Argo Workflows
IaC	Terraform + Ansible 통합
클라우드	AWS LZ / Azure Landing Zone
보안	OpenSCAP + STIG + 모의 해킹 (Kali)
DB 고급	PostgreSQL HA (Patroni) + Read Replica
NW 고급	BGP + VXLAN + Calico

💡 실전 한 줄

- 운영 = "**실제로 다룰 줄 아는 도구 × 자동화**" — 둘 다 필요
- Day 1 (이론) + Day 2 (설계) + Day 3 (실습) = TA 1주 표준
- 실습 환경은 **재사용 가능한 VM 스냅샷**으로 보관 (다음 강의 가속)

📚 추천 자료

- **Linux 공식**: kernel.org/doc/html/latest
- **K8s**: kubernetes.io/docs
- **Prometheus**: prometheus.io/docs
- **Wazuh**: documentation.wazuh.com

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현