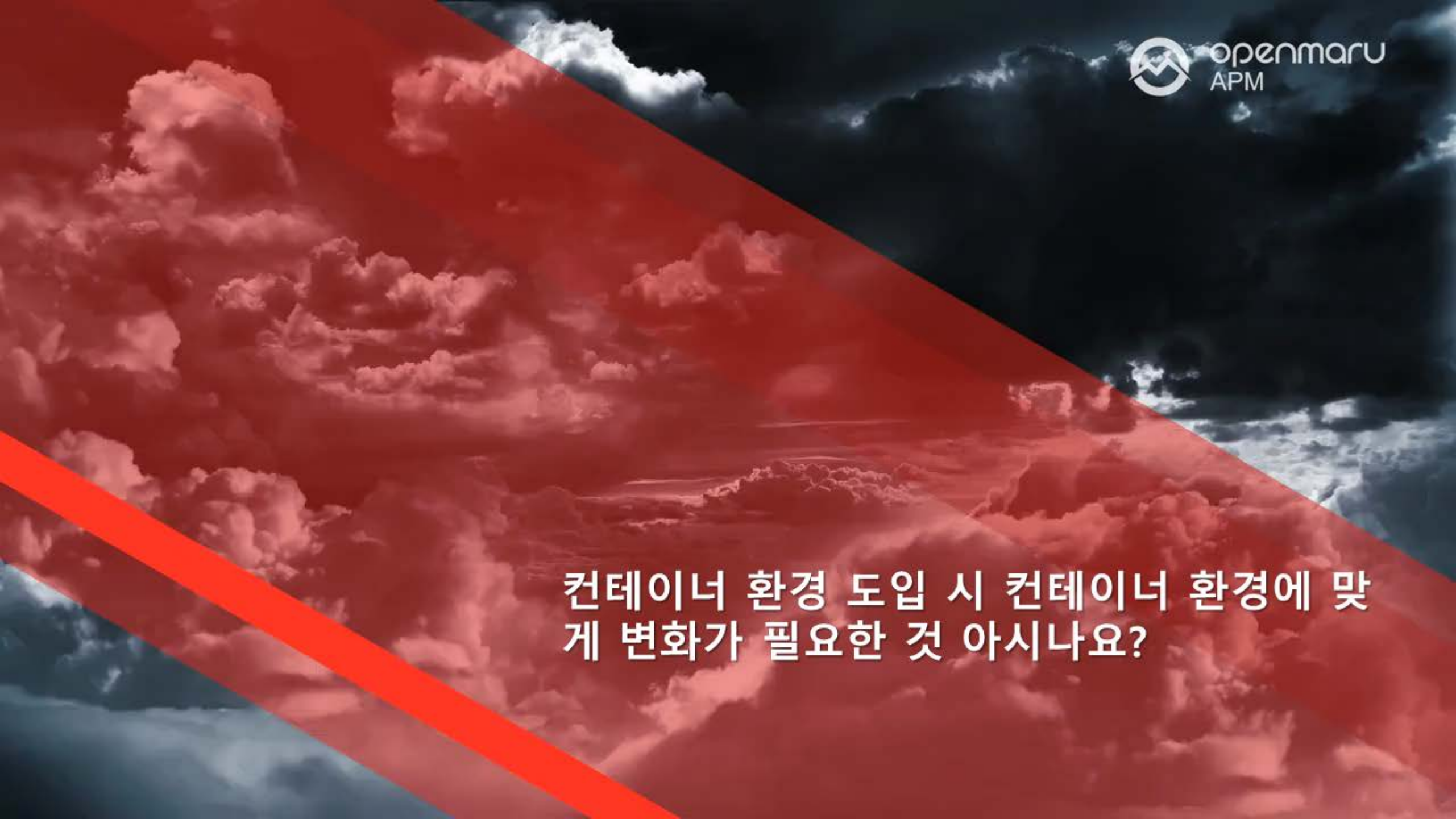
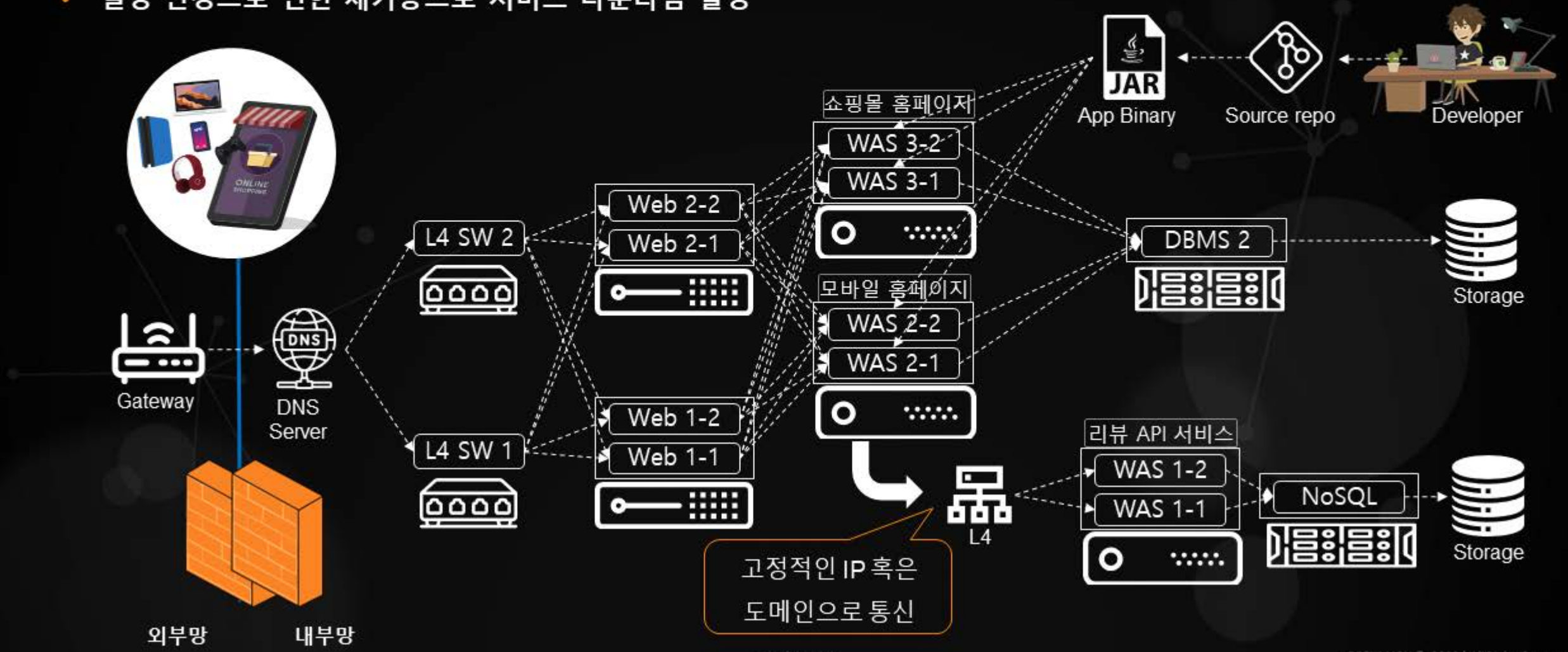

openmaru
APM
컨테이너 환경 도입 시 컨테이너 환경에 맞게 변화가 필요한 것 아시나요?

목차

Case 1. 애플리케이션 서비스 디스커버리

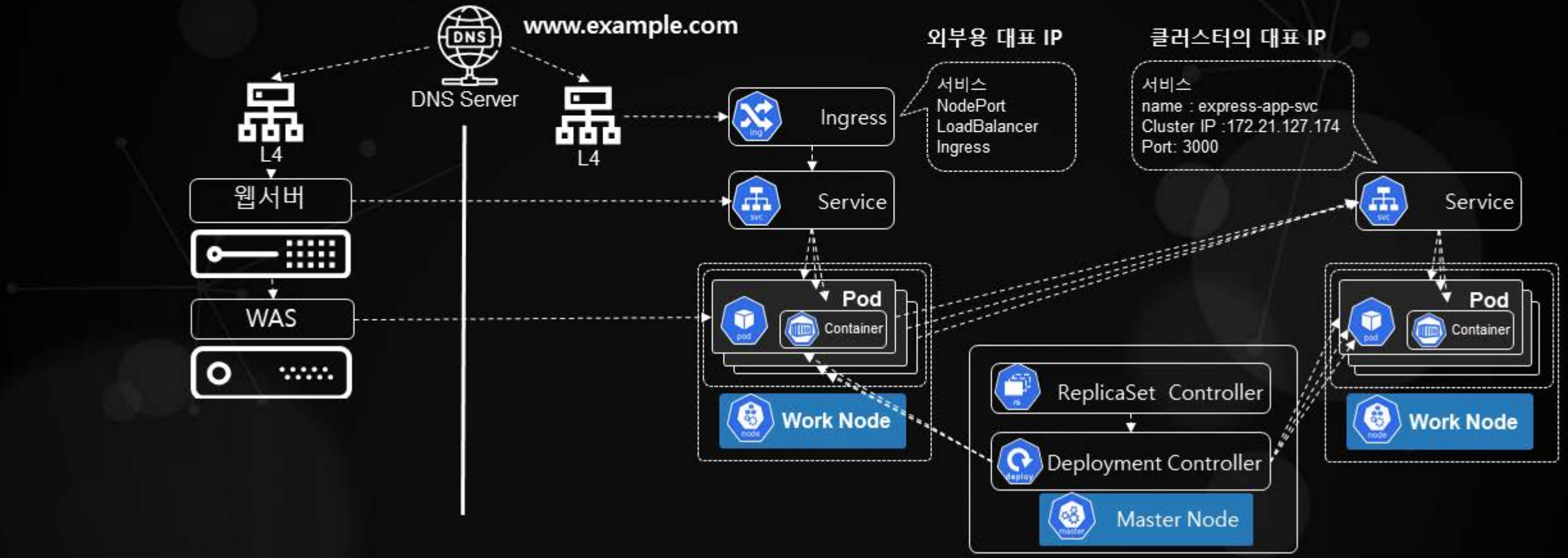
AS-IS(레거시) 환경의 고정IP를 이용한 애플리케이션간 통신

- 서비스 확장 시 모든 설정을 수작업
- 설정 변경으로 인한 재기동으로 서비스 다운타임 발생



컨테이너 간의 호출은 service를 이용

- 컨테이너 환경의 내부 Load Balancer(Domain 기반) 통신
- Service Object의 도메인 기반 으로 호출
- Pod는 유동적인 IP Pool을 갖으며 고정적인 IP를 가지지 않음



고정 IP 환경 (AS-IS) VS. 동적 IP 환경 (To-Be)

	고정 IP 환경	동적 IP 환경
구조	• 최초 구성부터 고정된 IP 를 할당 • **IP기반 호출방식**	• 지정 IP 가 없으며, 장애/ 배포/ 재시작 등 상태에 따라 IP 변경 • **Service 호출방식**
개발자	• 코드에 IP 정보 하드코딩	• **Service Domain 기준으로 서비스 디스커버리**
운영자	• 서버별로 수동 배포 • **서버 환경 변경(추가삭제)시 수작업 환경설정**	• CI/CD 자동화 • Auto scaling • Auto healing
기획자	• 고가의 보안소프트웨어와 비용 발생	• 비즈니스 유연한 확장성 • 클라우드 대응력 향상 • 소프트웨어 비용 절감

Case 2. 애플리케이션 및 서버의 설정 파일

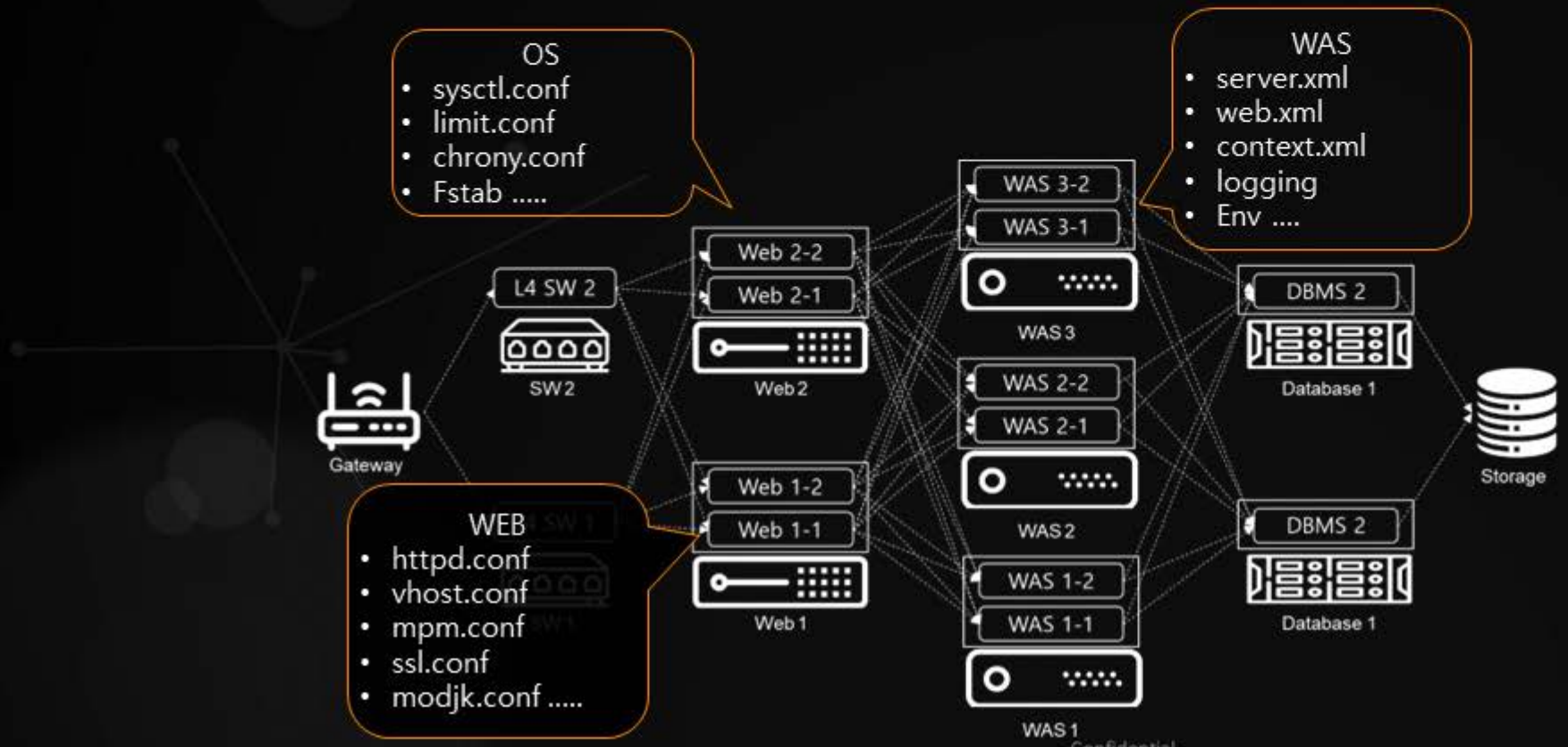
WAS, 애플리케이션 설정을 바꾸려면 무엇을 바꾸나요?

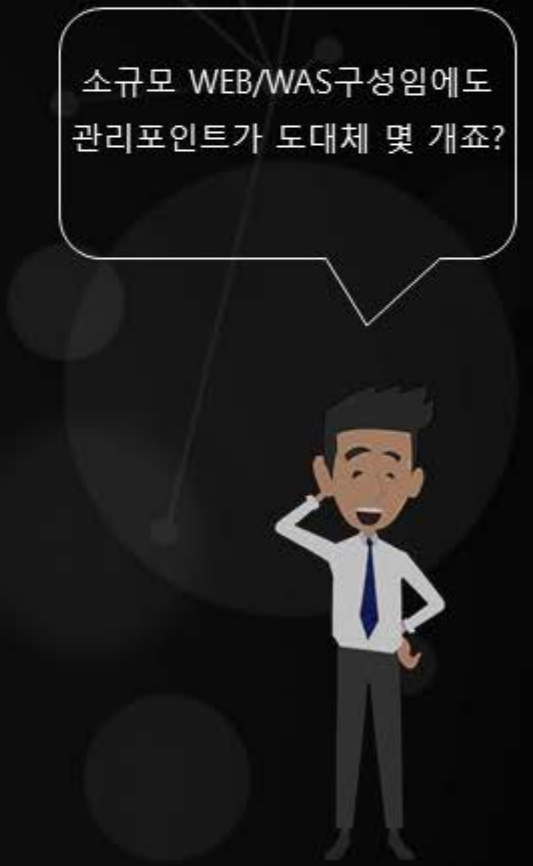
컨테이너를 직접 수정해도 되나요?

OS 설정은 어떻게 바꾸나요?

AS-IS : 개별 서버 별로 환경 설정파일 관리

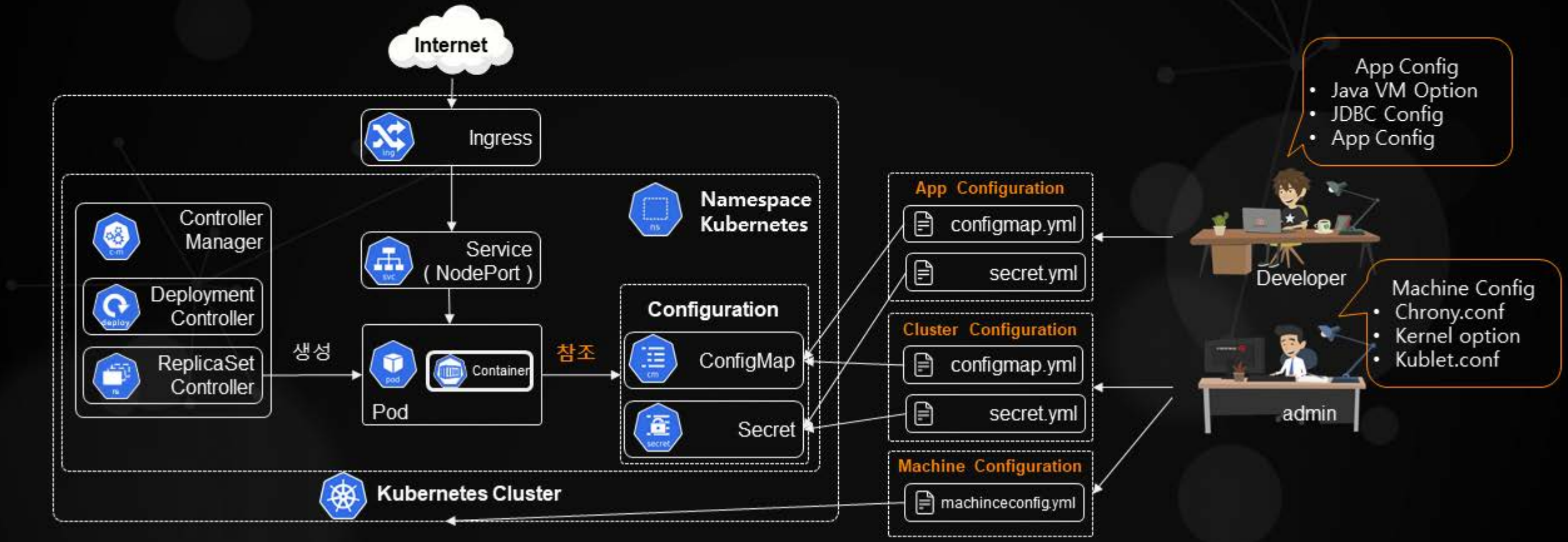
- OS/WEB/WAS 설정을 파일로 관리
  - Configuration Drift 발생
  - WEB/WAS의 구성이 많을수록 관리포인트 증가
- 설정이 변경된 후 WEB/WAS 재기동 다운타임 / 복잡한 무중단 구현





To-Be 환경 설정 통합 관리 (Configuration As A Service)

- 환경 의존 매개 변수를 외부화
- Configmap, Secret을 이용하여 WAS/APP 설정 통합 관리
- Machine Config를 이용하여 서버 설정 통합 관리



서버 별 환경설정 vs. 환경 설정 통합 관리(Configuration As A Service)

openmaru Red Hat

	개별 서버 별 설정 관리	클라우드 네이티브 환경
구조	• **개별 서버별로 설정파일 관리**	• **머신 컨피그 (Machineconfig) 와 Configmap 활용**
개발자	• 개발/스테이징/운영 환경 설정이 달라짐 • 개발과 장애 발생 시 환경 설정이 어려움 • **표준 개발환경 구성이 어려움** ▪ 구인과 인수/인계가 어려움 ▪ 개발 환경 교육 시간과 적응 기간이 오래 걸림	• 개발/스테이징/운영 환경이 동일함 • 애플리케이션 버전/개발환경 별로 필요한 환경 구성 • 표준개발환경으로 신규 인력의 적응 기간 단출
운영자	• Configuration drift • **서버가 많으면 많을 수록 작업량 증가** • 서버 별 설정 파일 관리	• **OS는 Machineconfig 기반의 환경 통합 관리** • **Application은 secret, Configmap 기반의 환경 통합 관리** • Git ops로 확장 가능
기획자	• 고가의 보안소프트웨어와 비용 발생	• 비즈니스 유연한 확장성 • 클라우드 대응력 향상 • 소프트웨어 비용 절감

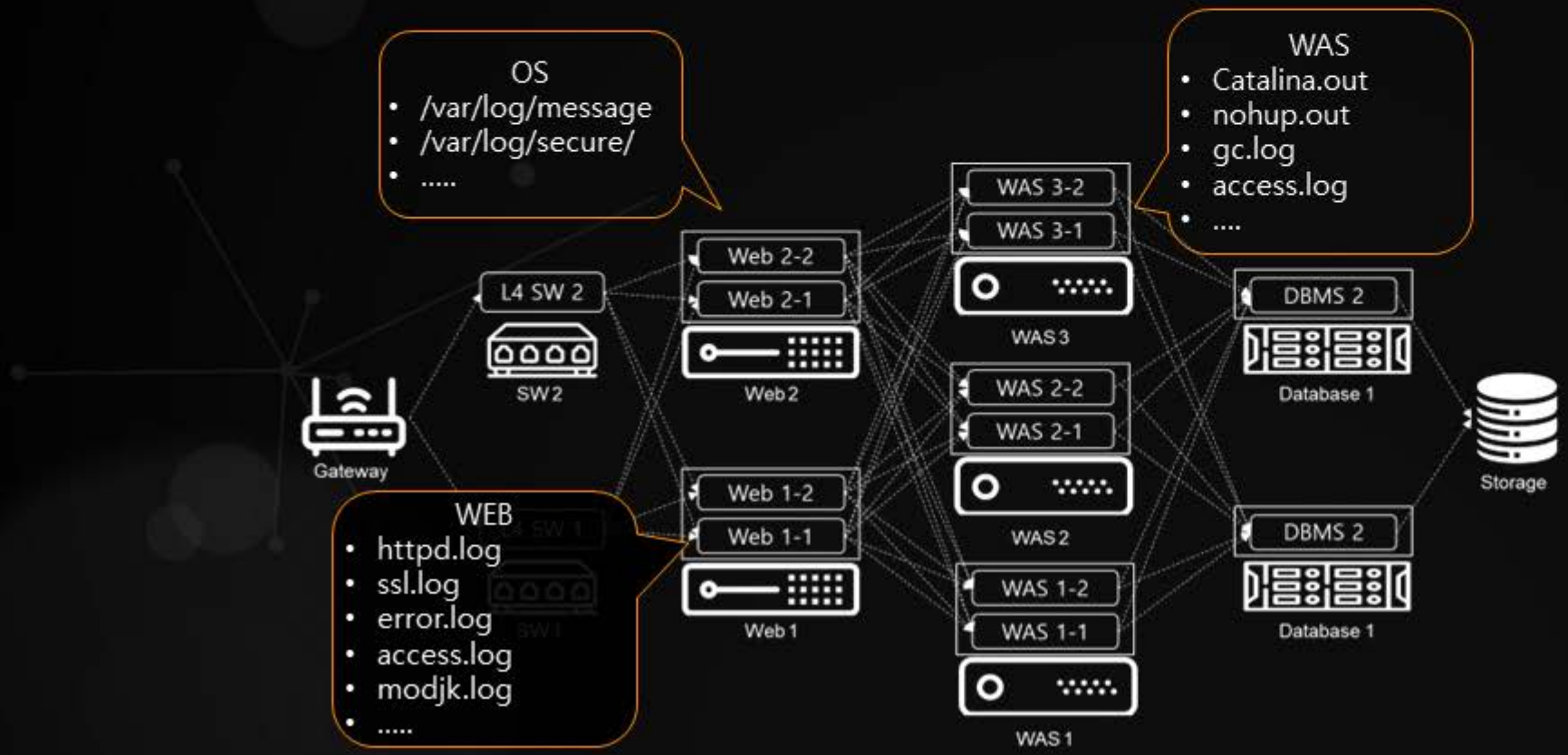
Case 3. 애플리케이션 로그

로그를 확인하고 싶은데,
어디서 확인하는거죠?

로그를 컨테이너의
어디에 쌓는건가요?

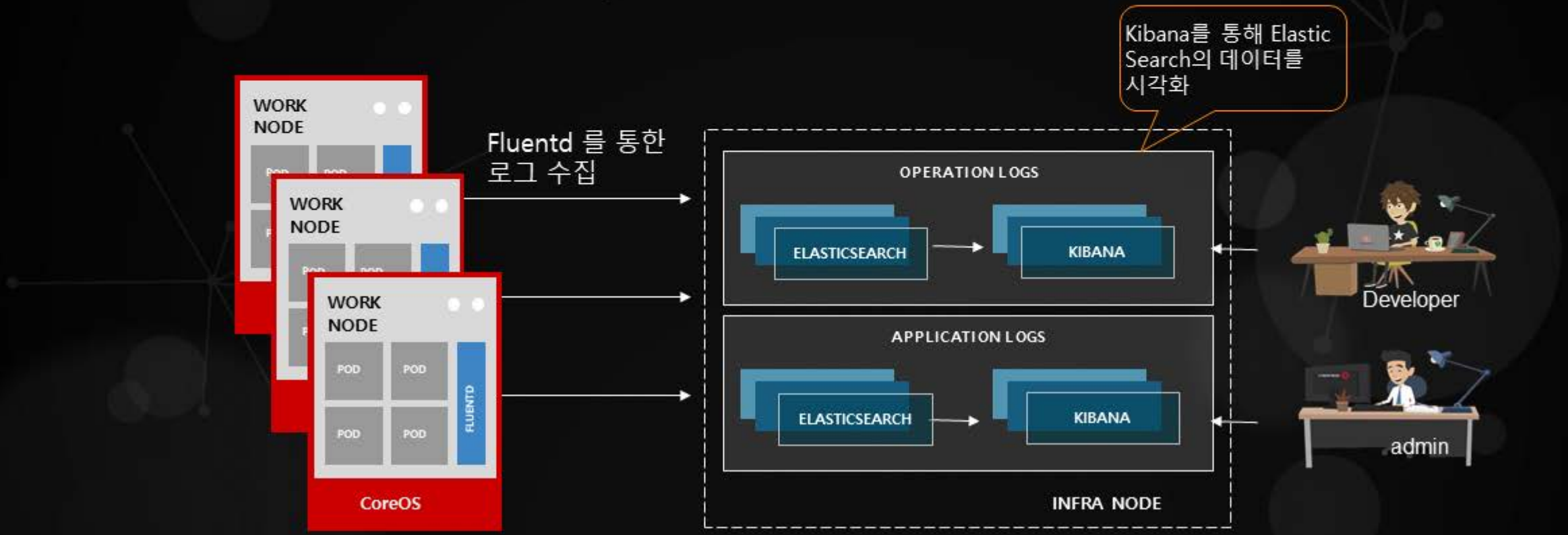
AS-IS – 개별 서버 별로 로그 파일 관리

- 발생하는 로그들이 각각의 서버 **Local Disk**에 저장됨
- 에러 발생시 에러가 발생한 서버를 찾아야 하고 발생한 에러로그를 분석
- 서버 대수가 늘어날 수록 모니터링 해야하는 **Log**의 수도 늘어남



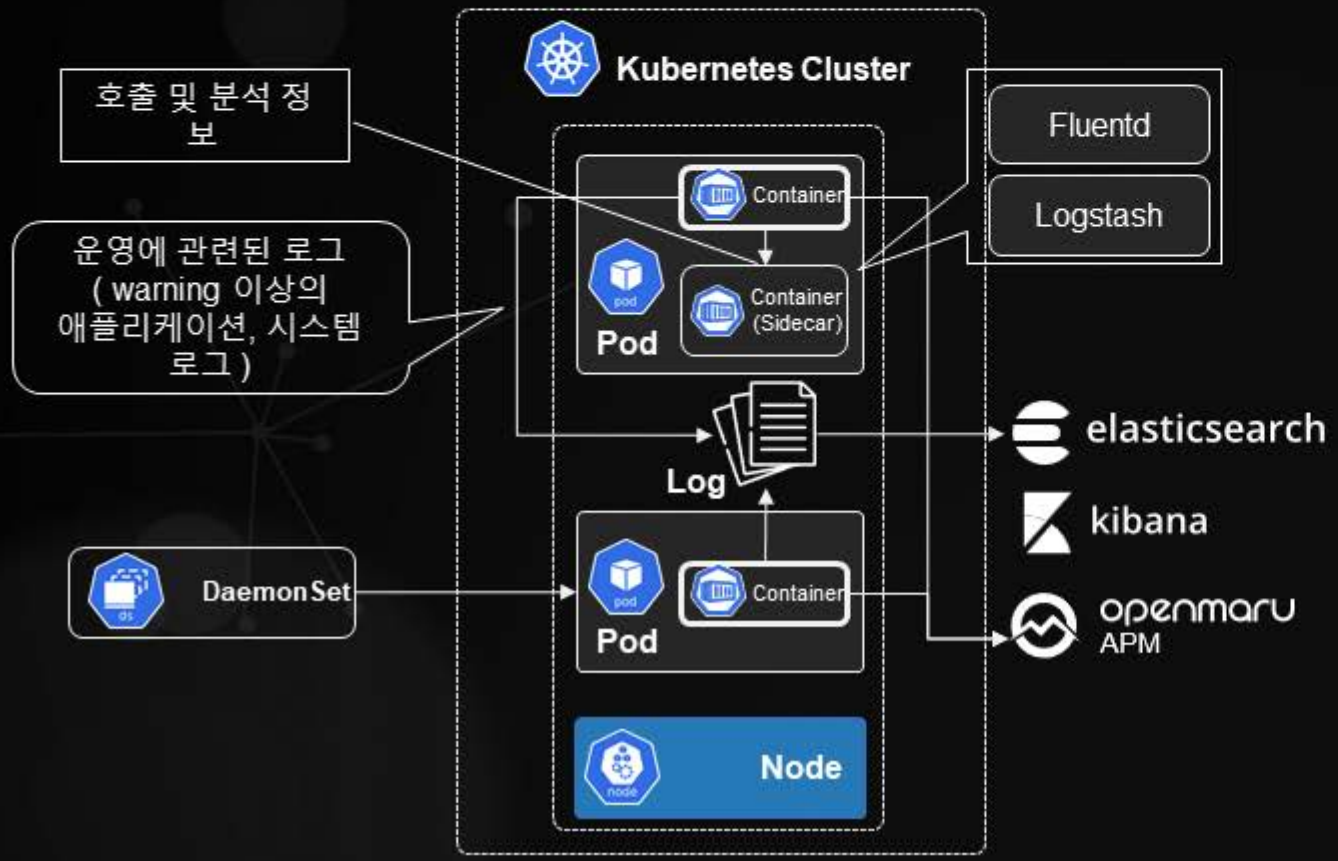
To-Be : 통합 로그 관리

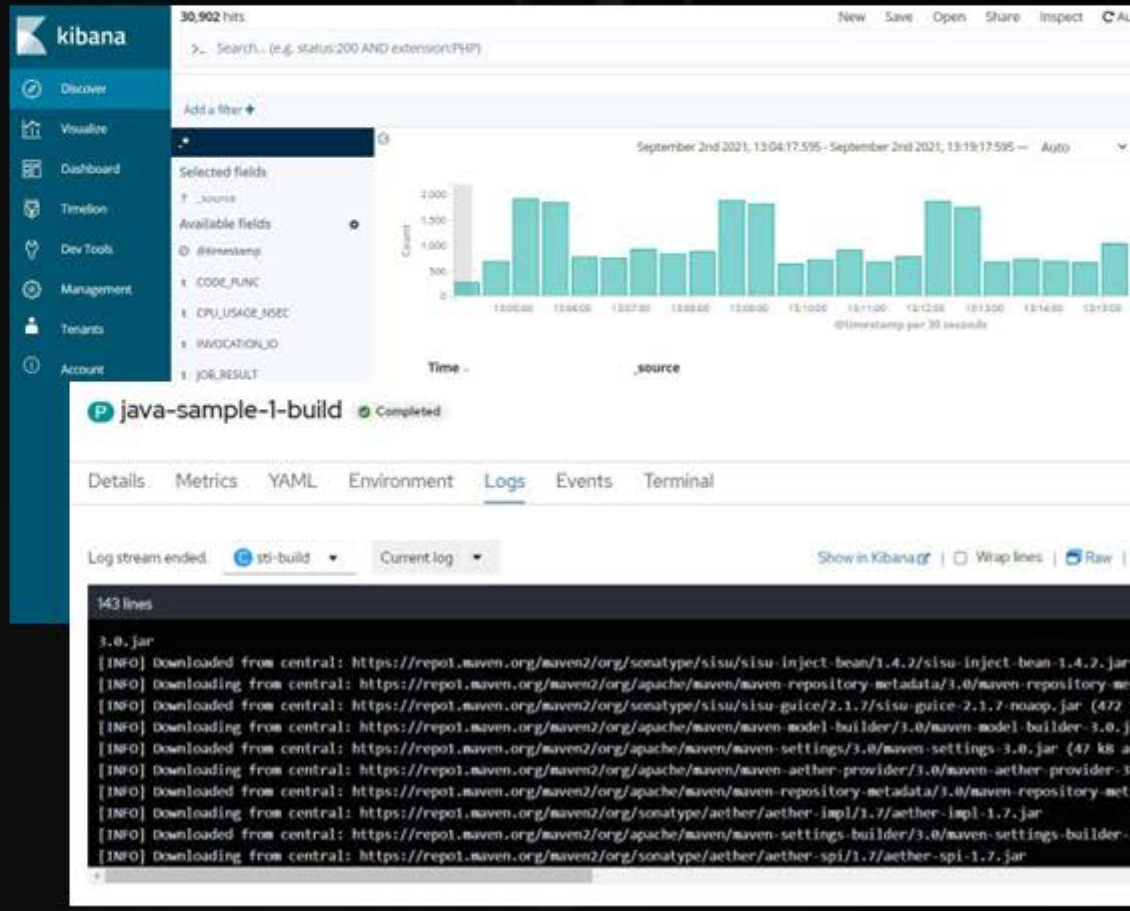
- PaaS 플랫폼의 경우 약 200개 이상의 플랫폼 인프라 컨테이너 기동
- 애플리케이션 컨테이너도 기존 환경에 비해 몇배 많은 수가 기동
- 수많은 컨테이너의 로그를 통합 모니터링하는 스택이 필수
- EF(L)K (Elastic Search + Fluentd(Logstash) + Kibana) 스택



To-Be : 통합 애플리케이션 로그 관리 방안

- 기존 레거시 환경에서는 개별 서버에 로그 생성 후 별도 작업으로 통합 관리
- EFK (ElasticSearch, Fluentd, Kibana) 를 활용한 자동 로그 통합 관리
  - 애플리케이션의 로그 정책을 STDOUT으로 설정





개별 서버 로그 관리 vs. 통합로그 관리 (Observability)

openmaru Red Hat

	개별 서버 로그관리	통합로그관리
구조	• 개별 서버에 로그가 쌓임	• EFK 스택을 통해 로그 자동 수집 및 통합 , 검색
개발자	• **서버별로 장애가 난 로그를 확인해야함** • 로그 파일 검색	• **Kibana를 통해서 로그 웹콘솔로 확인(서버접속불가)**
운영자	• 로그 로테이션 설정 • **로그로 인한 서버 접속 보안 관리**	• 시스템 운영상에 대한 메트릭 커스텀으로 생성 후 모니터링 • 애플리케이션, 서버로그도 EFK에 저장
기획자	• 로그통합 프로젝트 필요 • 프로젝트 비용 발생	• 비즈니스 유연한 확장성 • 클라우드 대응력 향상 • 소프트웨어 비용 절감

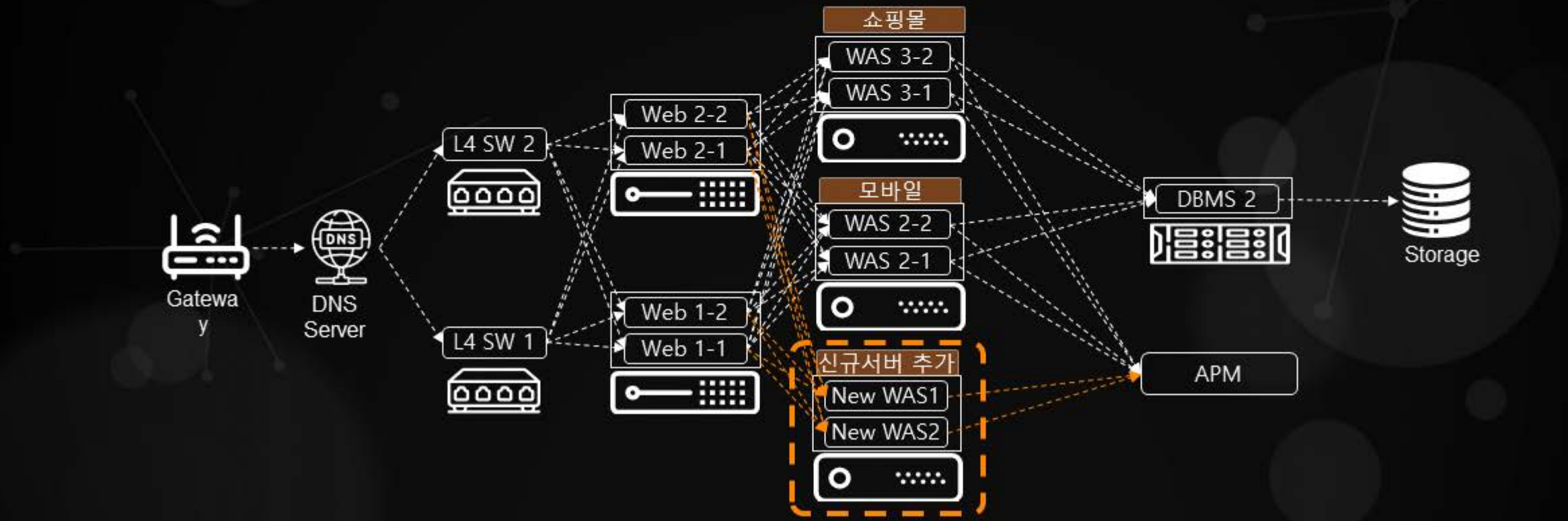
Case 4. 애플리케이션 모니터링

컨테이너 환경이라고
애플리케이션 모니터링이
다른가요?

다르다면 무엇이
다른거죠?

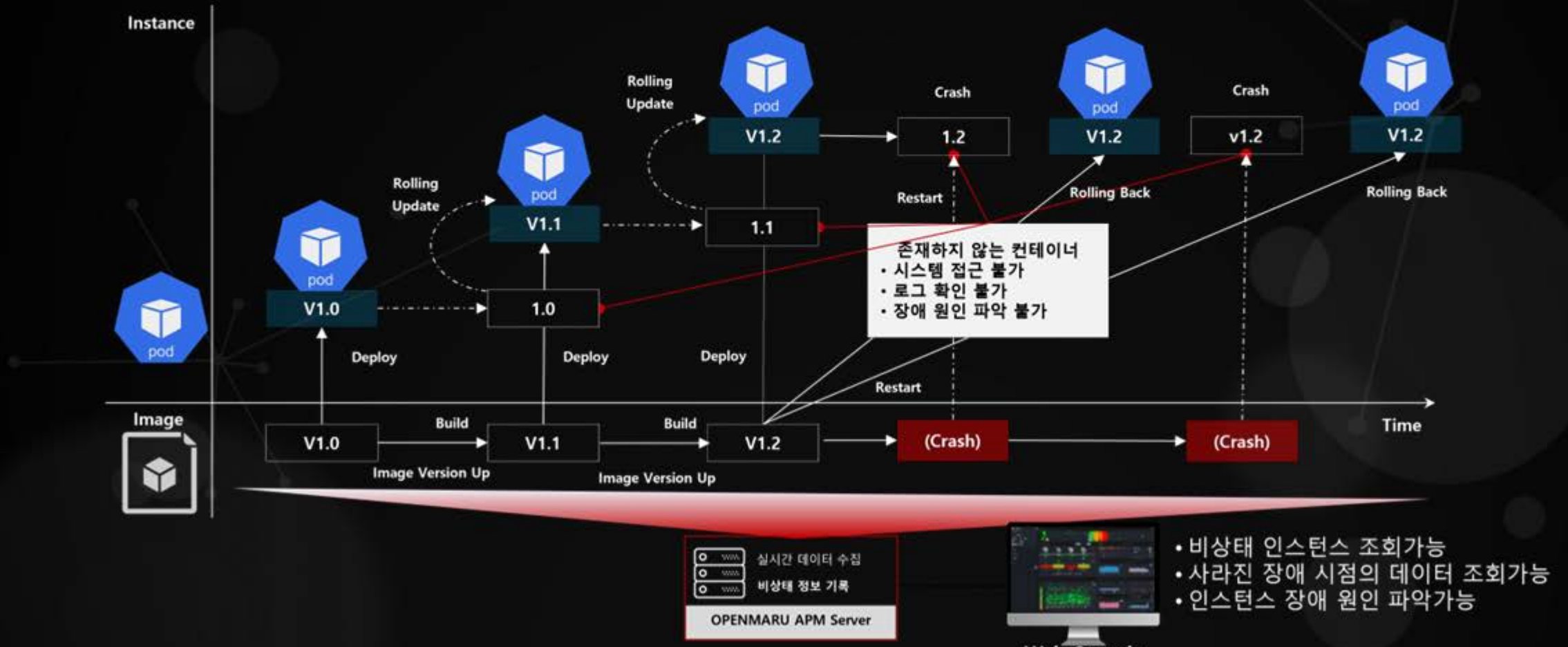
AS-IS : 레거시에서 모니터링

- 기존 환경은 WAS별 Instance가 고정적인 상태정보 (IP, Hostname 등)를 가지고 있음
- WAS 혹은 Instance들이 유연하게 확장/축소 되지 않는 환경
- 확장/축소가 유연하지 않음에 따라 APM도 그에 맞는 로직이 존재하지 않음
- WAS/Application에 장애가 발생하여도 기존 데이터가 사라지지 않음



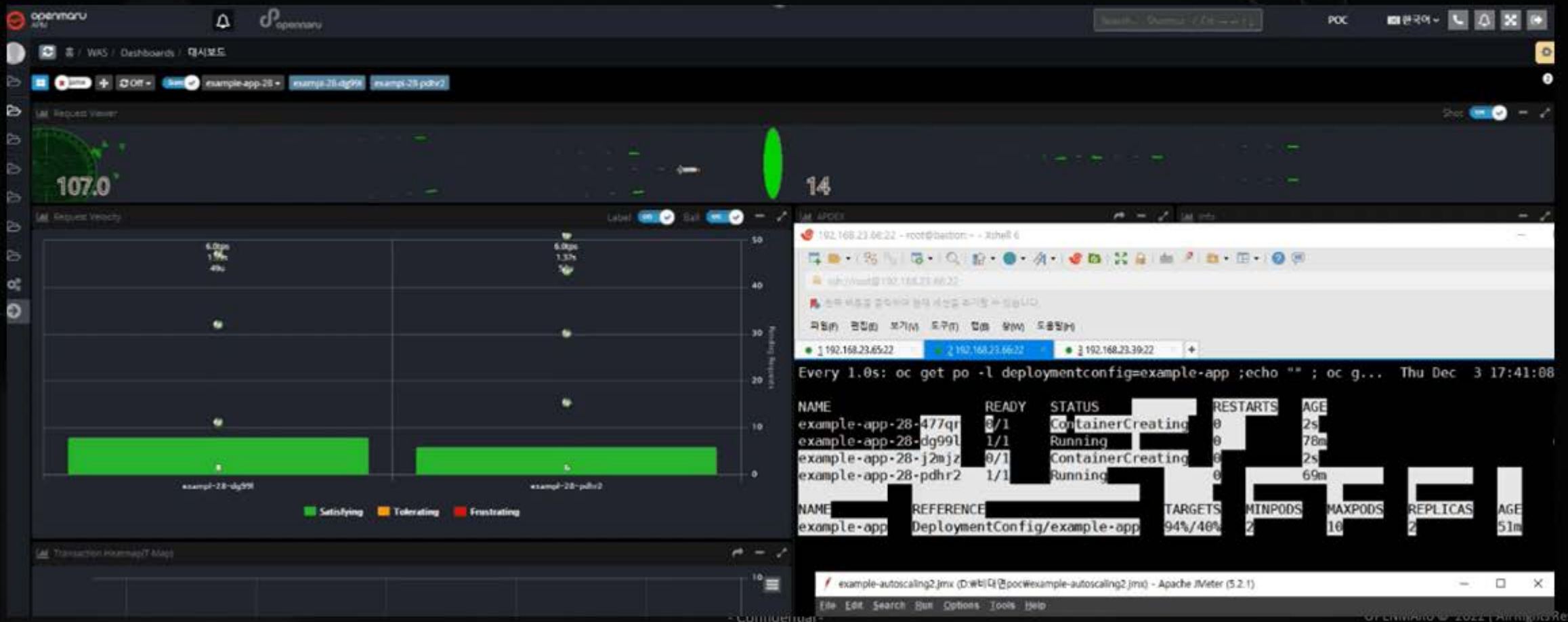
To-Be : 무상태 (Immutable) 인프라스트럭처로 상태를 저장하지 않음

- PaaS 환경에서 컨테이너가 중지된 후 장애원인 파악을 위한 방법을 제공
- APM 에서 사라진 컨테이너에 대한 정보를 보관하여 장애원인을 파악할 수 있음



To-Be(컨테이너) 환경에서 모니터링 구조

- Auto Scaling으로 Container(Pod)가 유연하게 Scale Out/In이 발생하는 환경
- APM 서버에는 그러한 환경을 뒷받침할 라이선스 정책과 로직이 필요함




openmaru

제품 / 서비스에 관한 문의

- 콜 센터 : 02-469-5426 (휴대폰 : 010-2243-3394)
- 전자 메일 : sales@opennaru.com