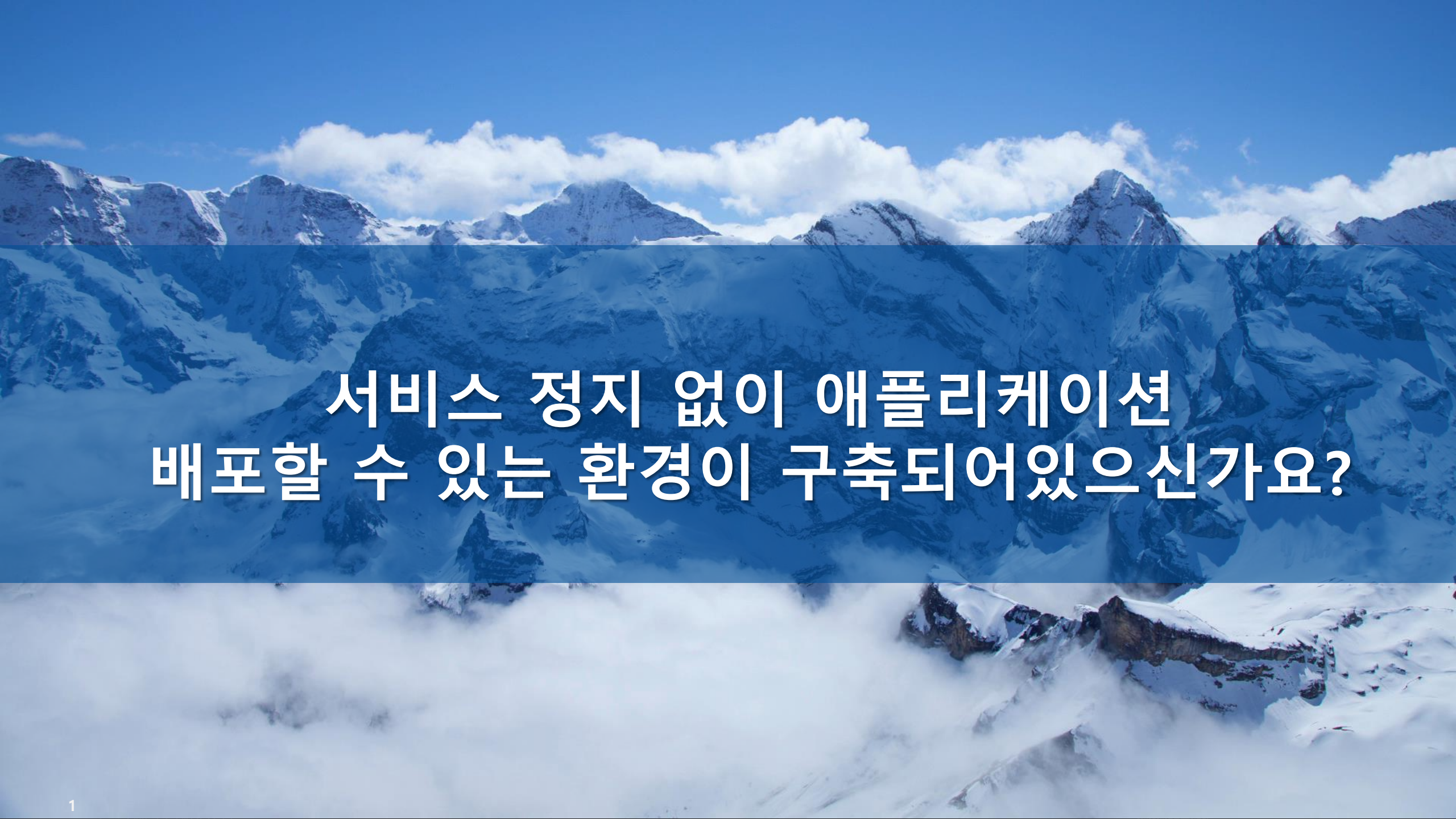
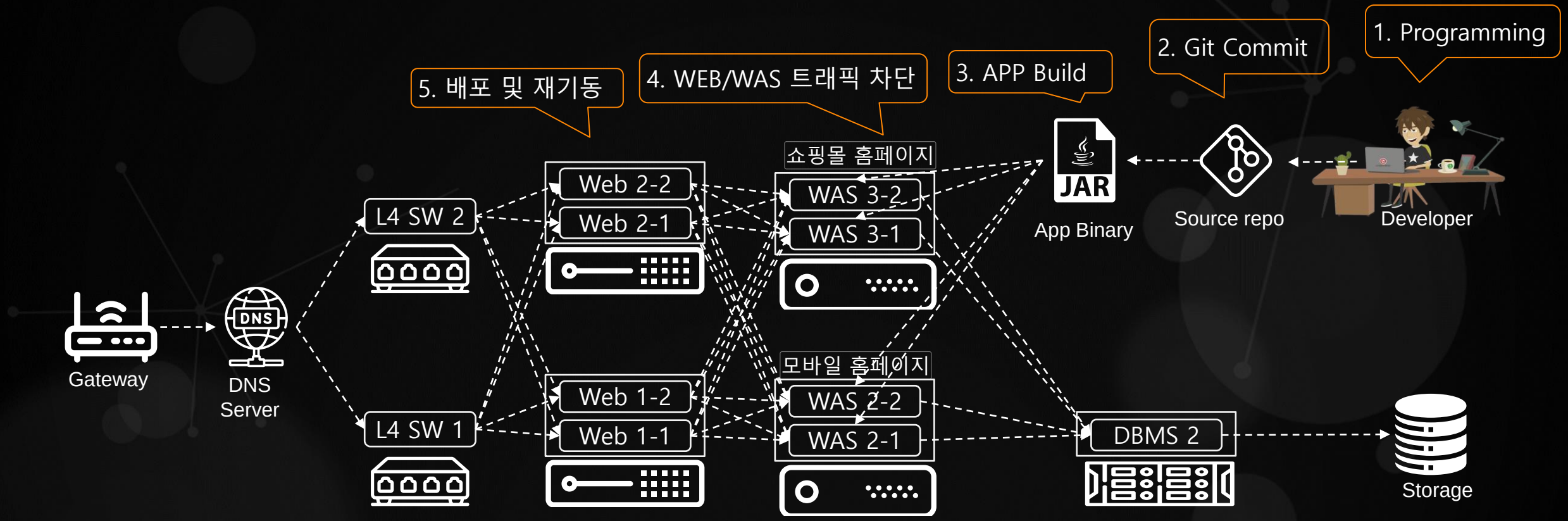
클라우드 네이티브 도입시 고민과 구현 방안


서비스 정지 없이 애플리케이션
배포할 수 있는 환경이 구축되어있으신가요?

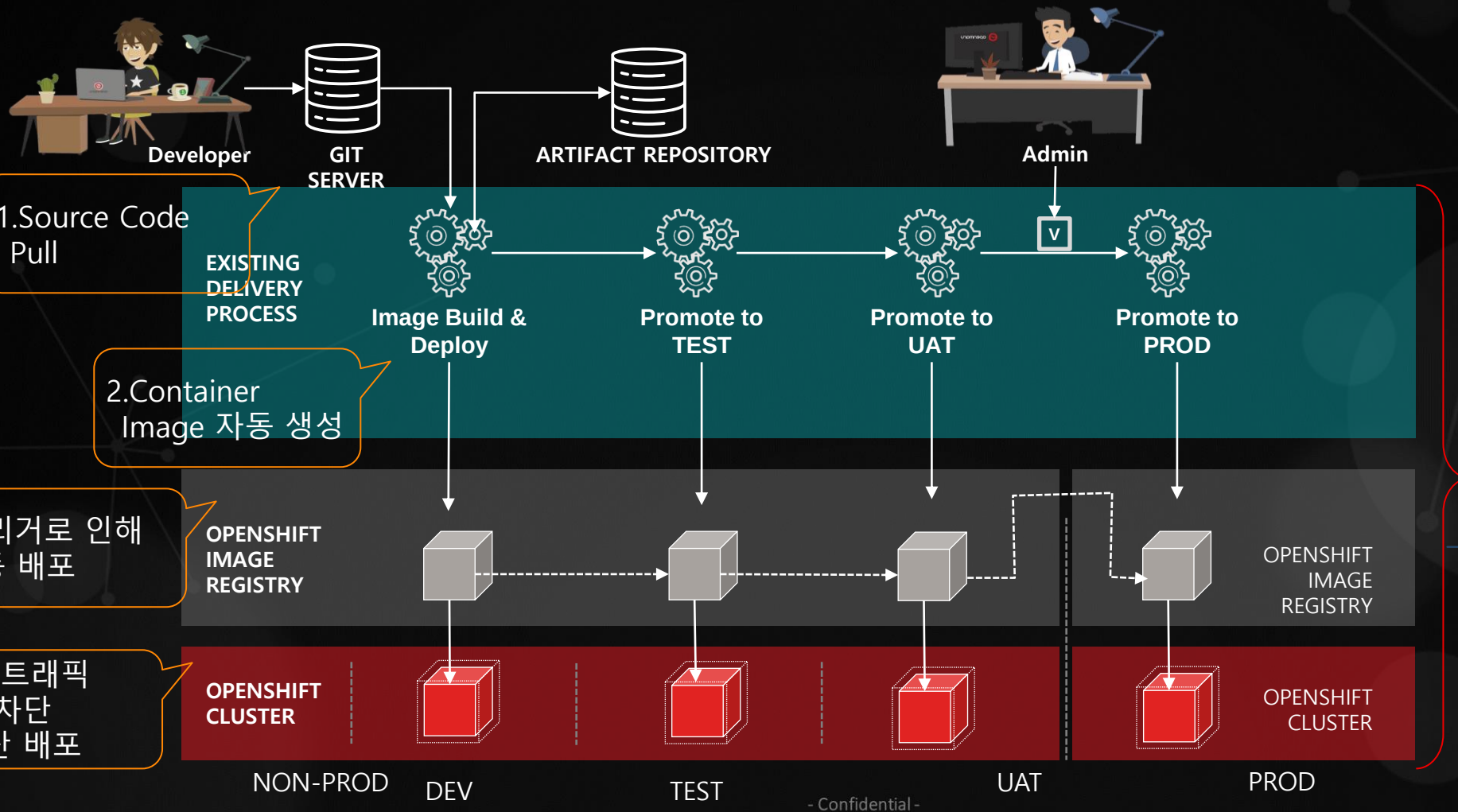
기존 환경의 애플리케이션 Build / Deploy

- 기존 환경에서의 수작업을 통한 반복적인 Build / Deploy
- 개발 → git commit / push → 애플리케이션 build → WAS 서버에 배포



클라우드 네이티브 환경 자동 빌드/배포

- 신속한 개발과 효율적인 운영 환경 구축
- 코드 개발 이외에 개발자/관리자가 빌드 배포에 기여하는 수작업은 "빌드 명령" 밖에 없음

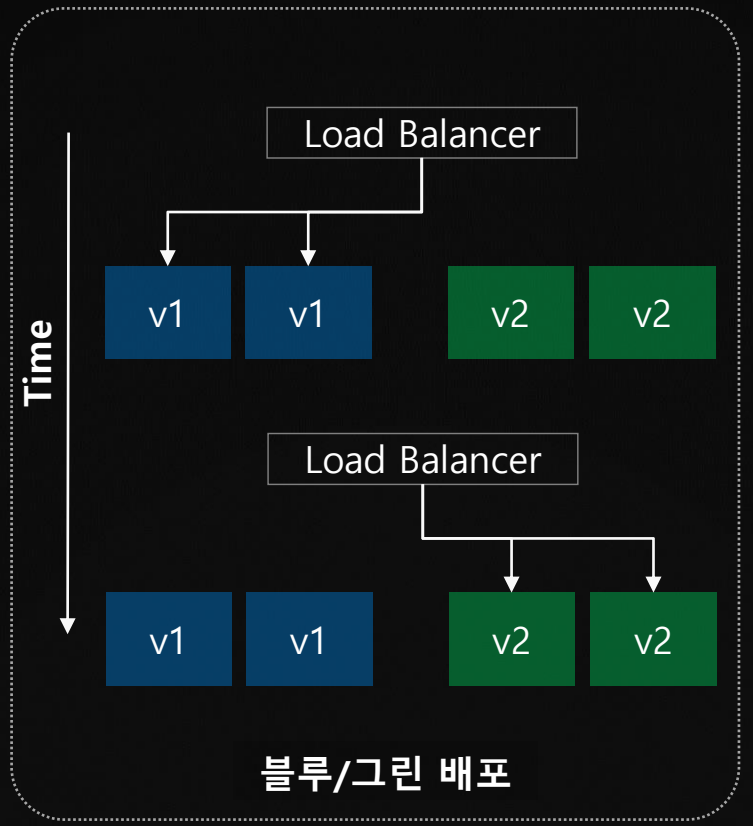


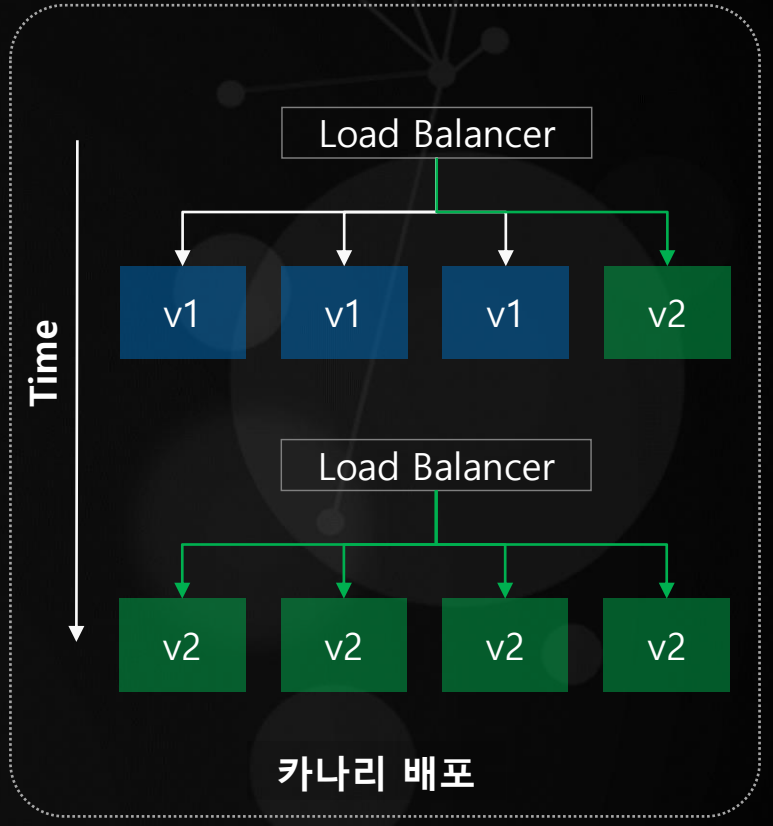
Red Hat OpenShift

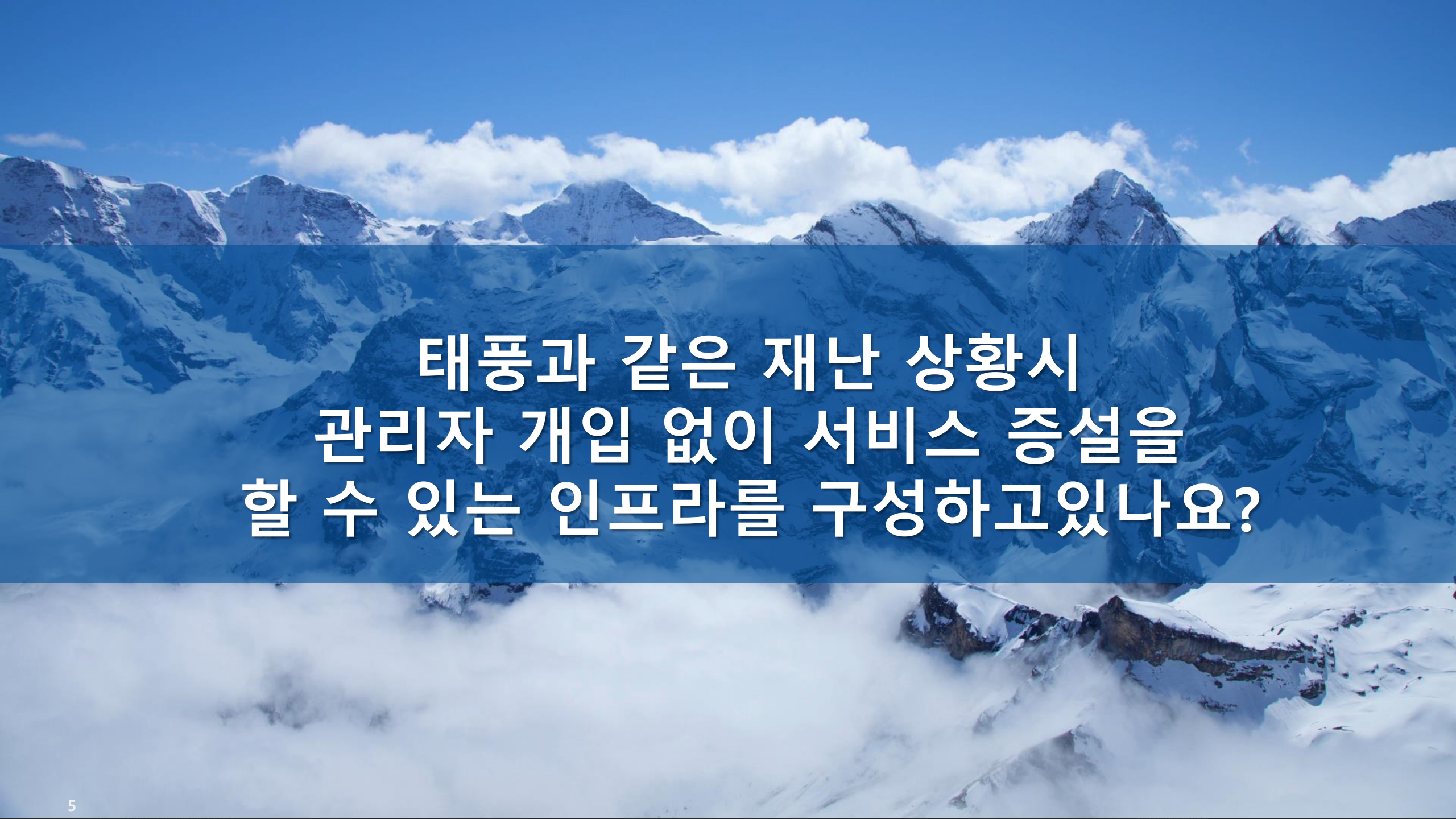
AUTONOMOUS PLATFORM

무중단 뿐만 아닌 고도화된 배포 방식

- 롤링 업데이트 - 점차 새로운 버전으로 변경
- 블루/그린 배포 – 블루 버전을 유지한채로 그린 버전을 테스트 (신속한 롤백)
- 카나리 배포 – 소규모로 검증 후 전체 반영 (신속한 롤백)



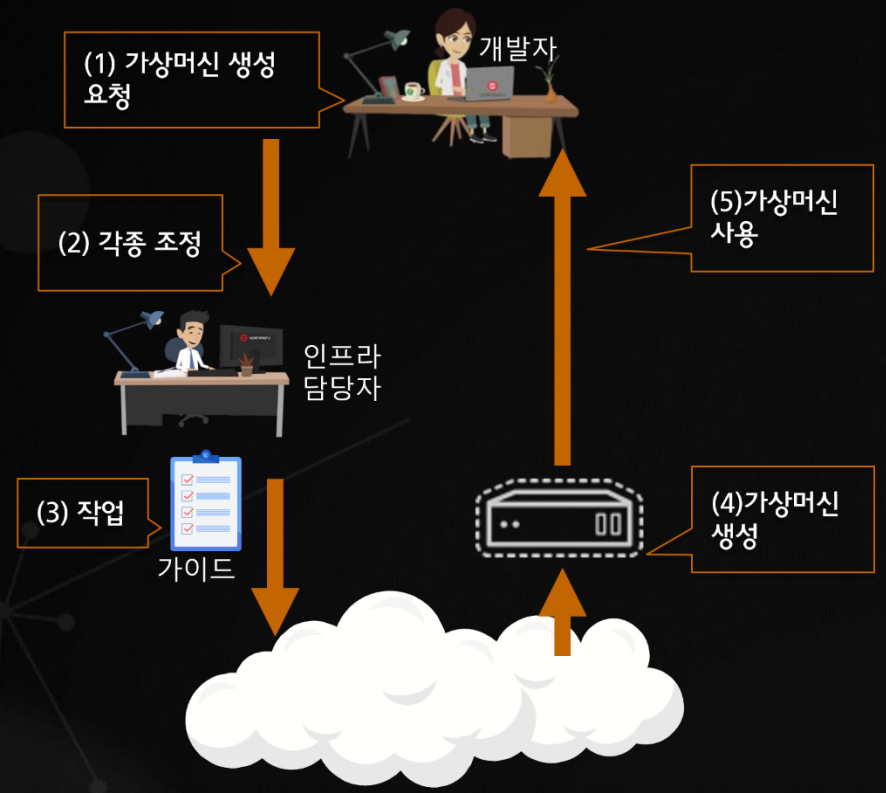



태풍과 같은 재난 상황시
관리자 개입 없이 서비스 증설을
할 수 있는 인프라를 구성하고있나요?

서비스 증설 작업 비교

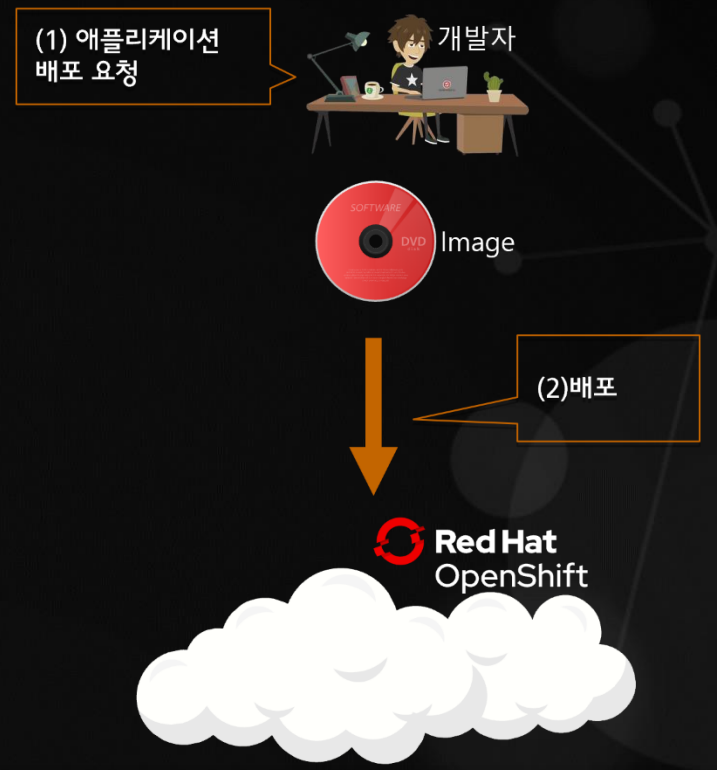
Red Hat | openmaru 오픈마루(주)

Before



- ✓ 사람에 의한 조정과 작업
- ✓ Human Error 와 품질의 차이

After

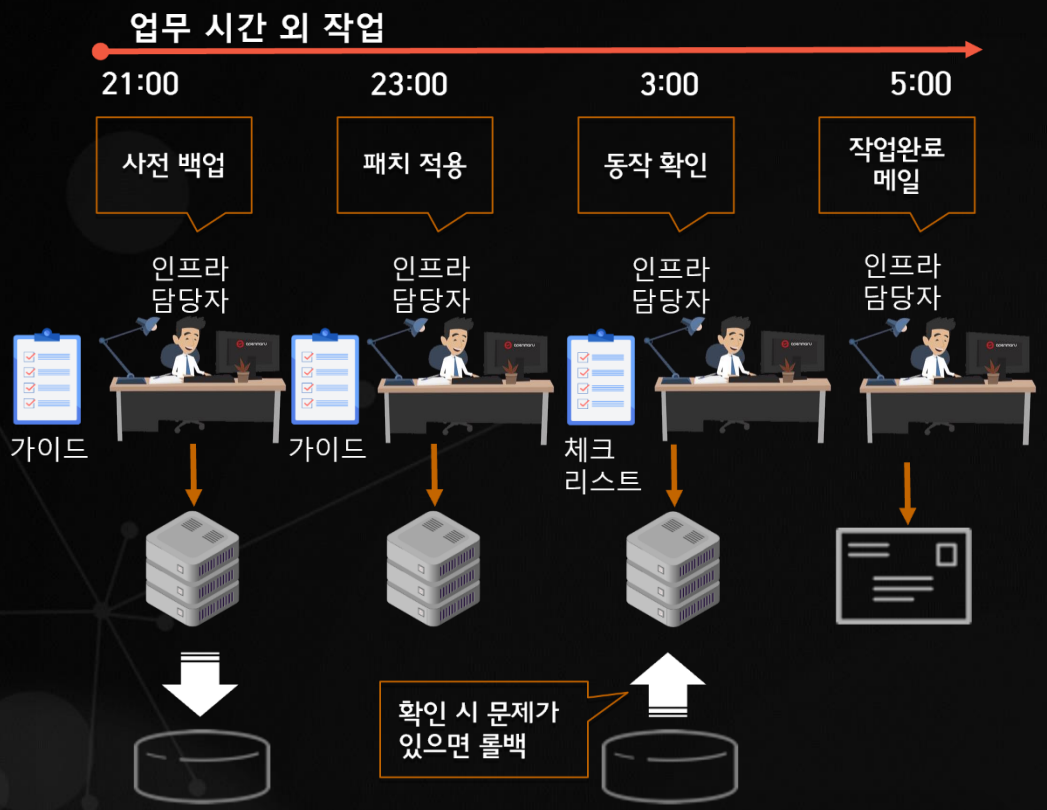


- ✓ 테스트까지도 자동화
- ✓ 일정한 품질 유지

운영환경 패치 작업 비교

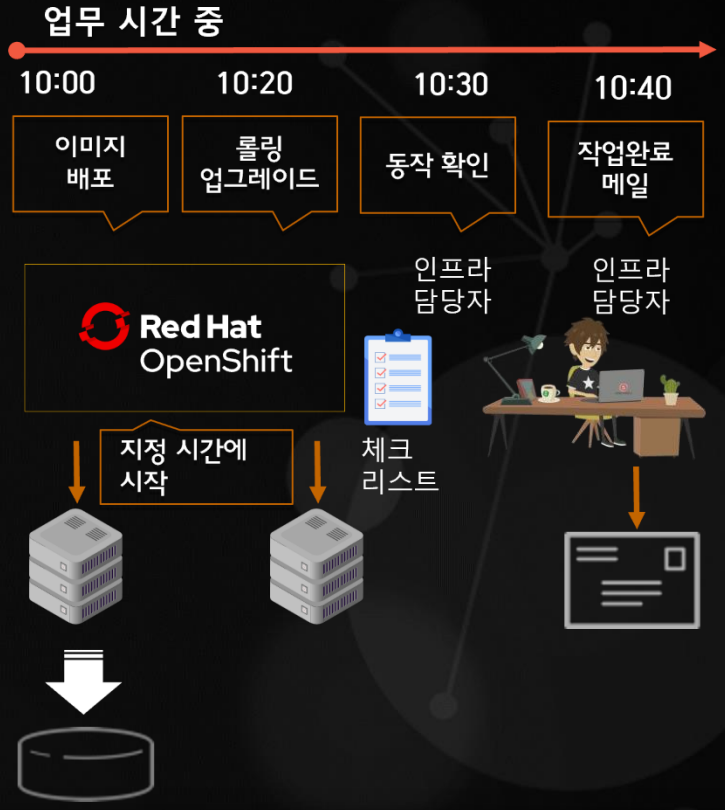
Red Hat | openmaru 오픈마루㈜

Before

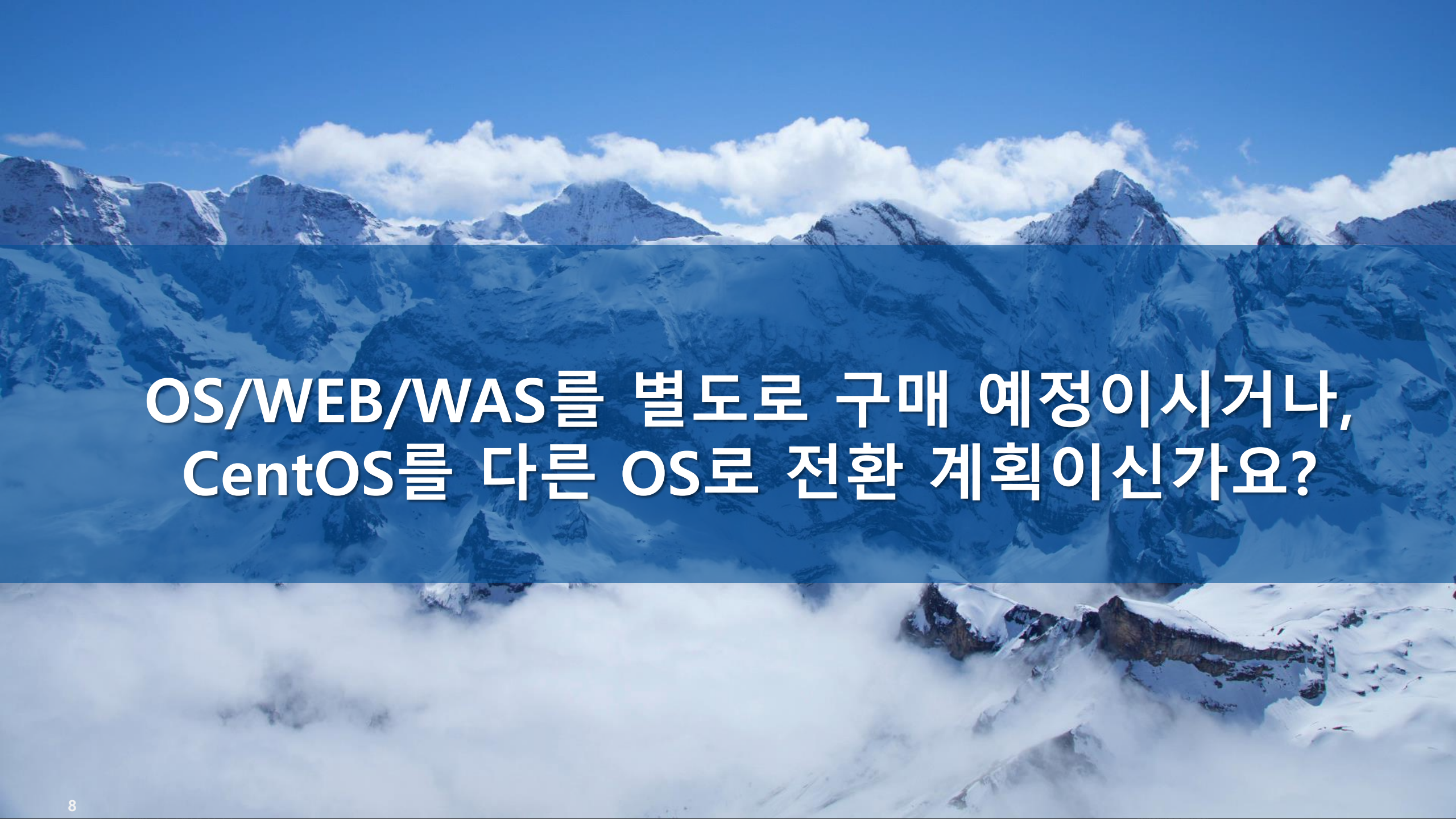


- ✓ 작업하는 동안 상주 지원하며 시간 소요
- ✓ 오류 확인 및 롤백에 대한 위험요소
- ✓ 품질의 차이

After

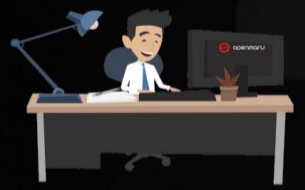


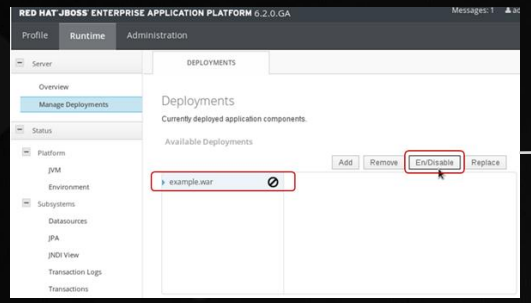
- ✓ 비상주 지원과 유비보수 시간 단축
- ✓ Human Error 차단
- ✓ 일정한 품질 유지


OS/WEB/WAS를 별도로 구매 예정이시거나,
CentOS를 다른 OS로 전환 계획이신가요?

인프라 운영 - WAS 관리 방법의 변화

- WAS 컨테이너 이미지는 관리 콘솔을 생략하고 배포
- 더 이상 개별 WAS 인스턴스에 대한 관리가 무의미 함

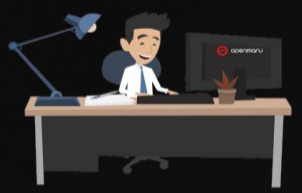




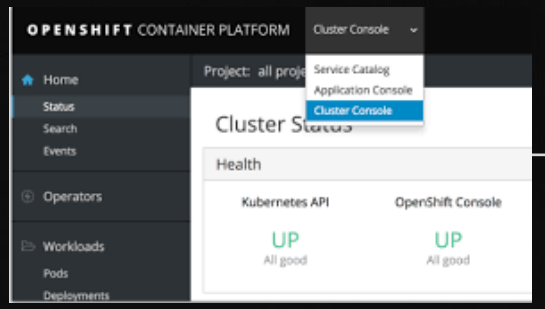
JBoss Admin Console

- WAS 수동 시작/정지
- WAS 수동 장애 복구

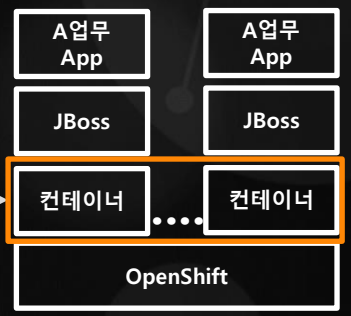
JBoss 관리 콘솔 - JBoss 관리



OpenShift 관리 콘솔 - 컨테이너(JBoss포함)



OpenShift Admin Console



- 컨테이너 자동확장/축소
- 컨테이너 자동장애 복구
- 컨테이너 서비스체크
- 컨테이너 모니터링

아직도 WAS BMT 나 POC를 하시나요?

Red Hat | openmaru 오픈마루㈜

- WAS 의 가용성/확장성/성능/신뢰성 등의 기능은 플랫폼으로 이전
- 더 가볍고 더 빠르고, 자동화에 친화적인 WAS 로 전환

WAS 제품 BMT

ORACLE WEBLOGIC

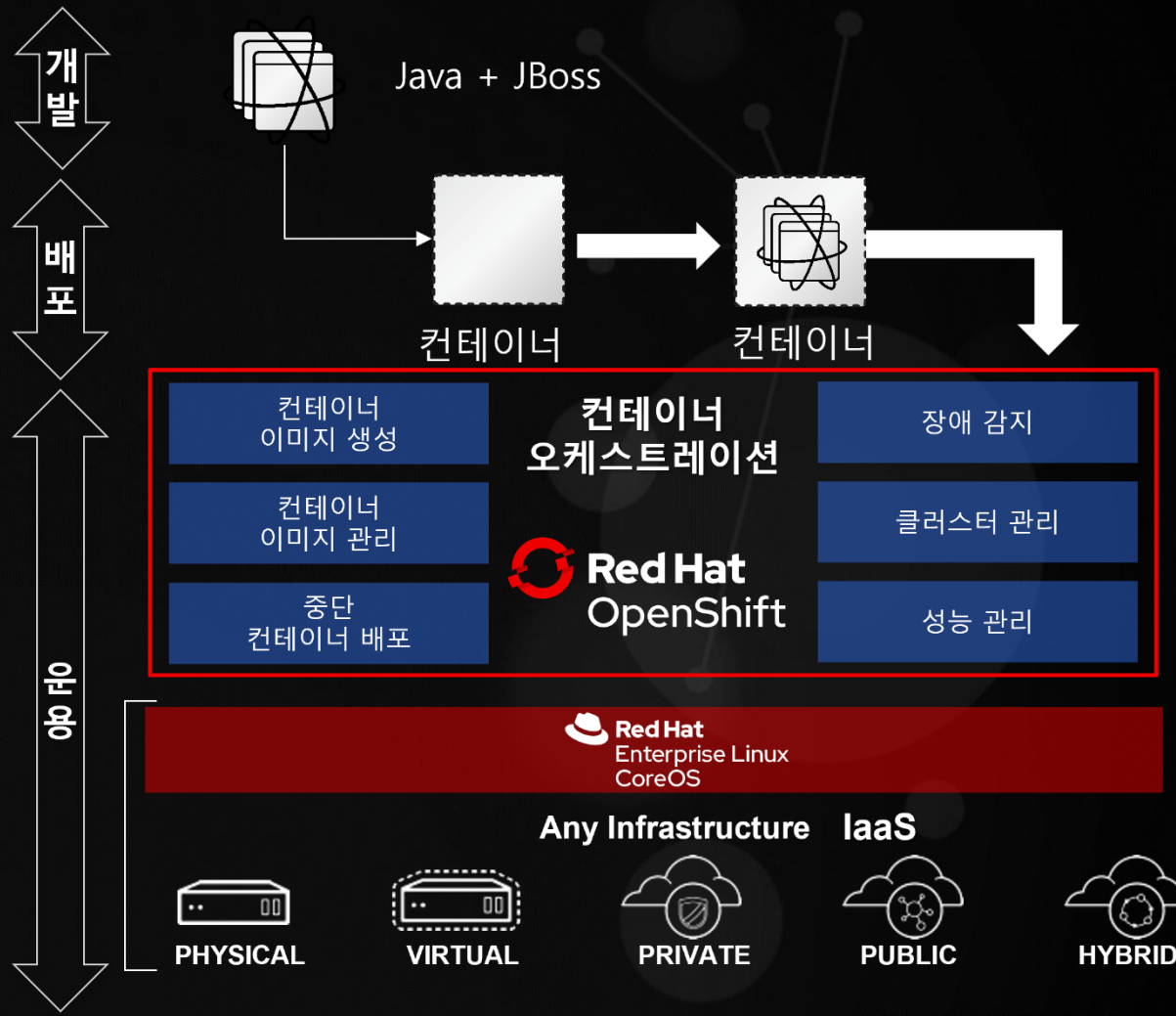
JBoss by Red Hat

WebSphere

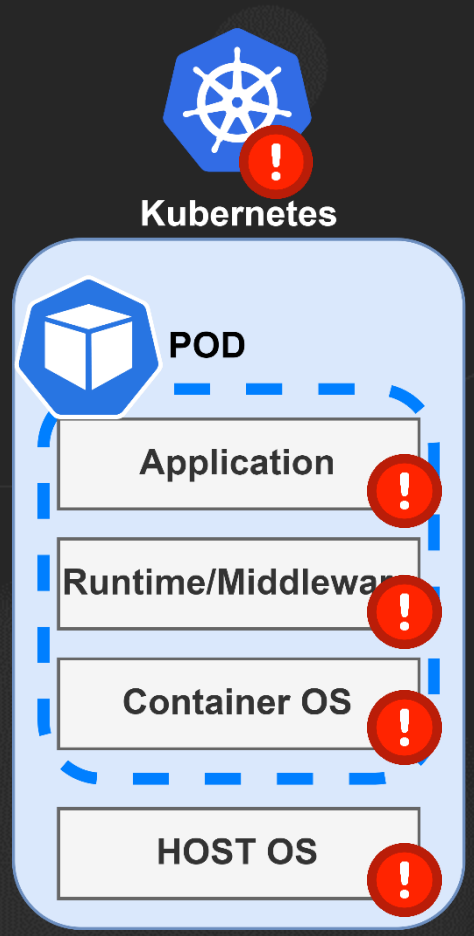
Tmax Web Application Server JEUS

RASP 에 의한 WAS평가

- 신뢰성 (Reliability):
  - 과부하 테스트
- 가용성 (Availability):
  - 일부 장애 발생시 전체 시스템 영향 최소화
- 확장성 (Scalability):
  - 자원 추가에 따른 선형적인 성능 개선
- 성능 (Performance):
  - TPS, 응답시간 등 평가
- 보안 기능 (Security)
  - (RBAC 등)
- WAS 도구 평가 (Manageability)
  - Admin Console

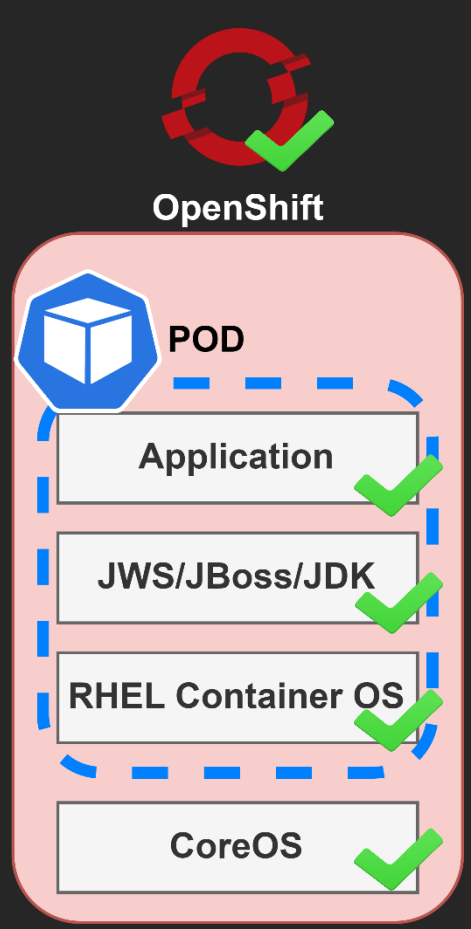


PaaS는 OS/WAS/Runtime를 제공해야 함



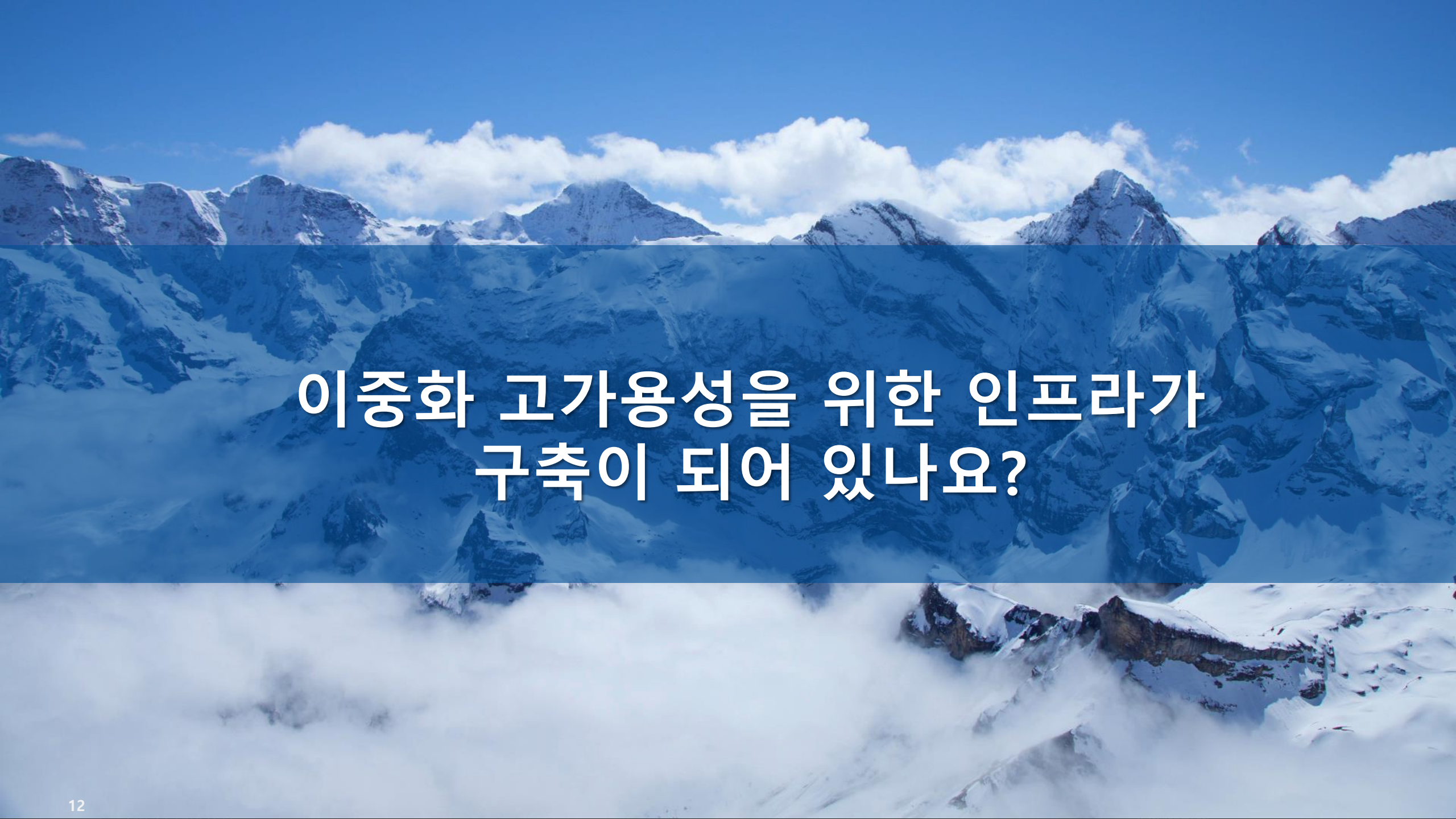
Kubernetes + DIY Stack

- 신뢰할 수 없는 컨테이너 이미지
  - 컨테이너 보안 업데이트 X
- QA팀을 통한 안정화 테스트 X
- HOST OS 유지보수
  - 업그레이드
  - 패치
  - 트러블슈팅
  - 구매비용 발생
- Runtime/Middleware 유지보수
  - 업그레이드
  - 패치
  - 트러블슈팅
  - 구매비용 발생



OpenShift Container Platform

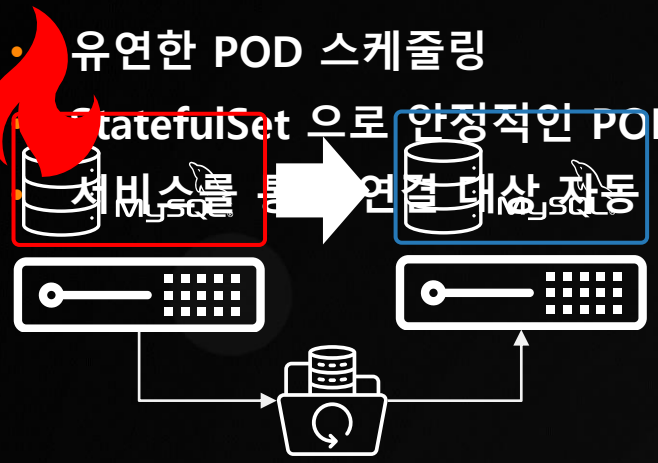
- 신뢰할 수 있는 컨테이너 이미지
  - Quay, red hat registry
- QA팀을 통한 안정화 테스트
- Red Hat Core OS
  - 업그레이드 지원
  - 버그 패치 지원
  - 트러블슈팅 지원
  - 컨테이너 특화 OS
  - 구매비용 발생 X
- OpenJDK / JBoss Web Server, EAP
  - 업그레이드 지원
  - 패치 지원
  - 트러블슈팅 지원
  - 구매비용 발생 X
    (EAP의 경우 구매비용 발생)

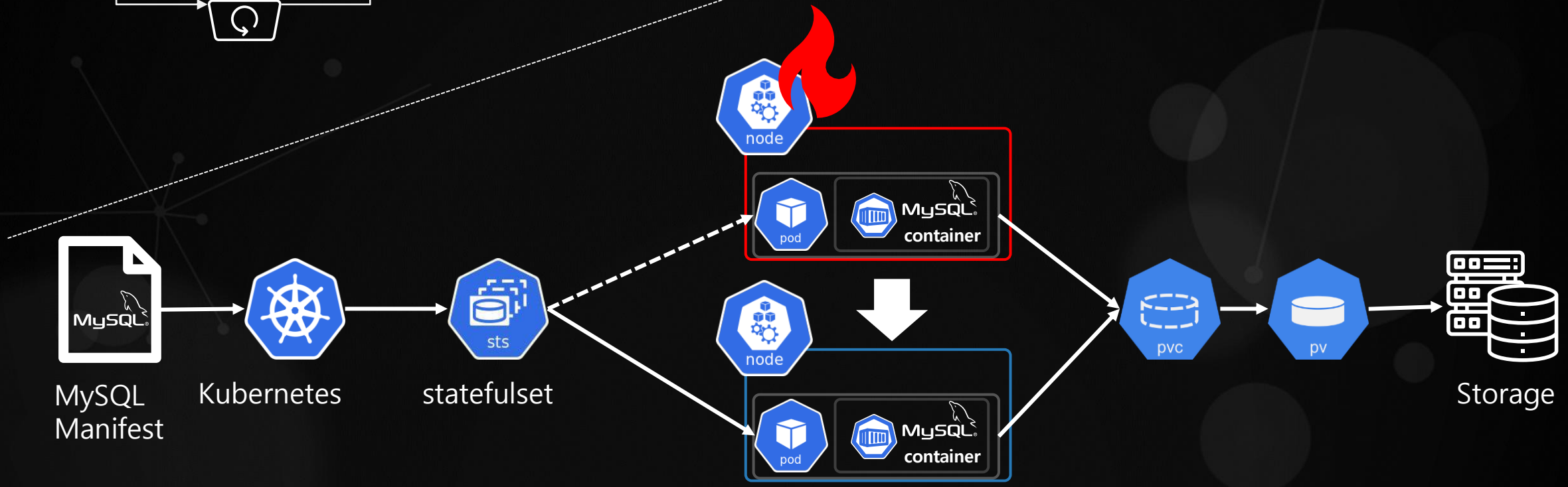

이중화 고가용성을 위한 인프라가
구축이 되어 있나요?

클라우드 네이티브를 이용한 MySQL 이중화

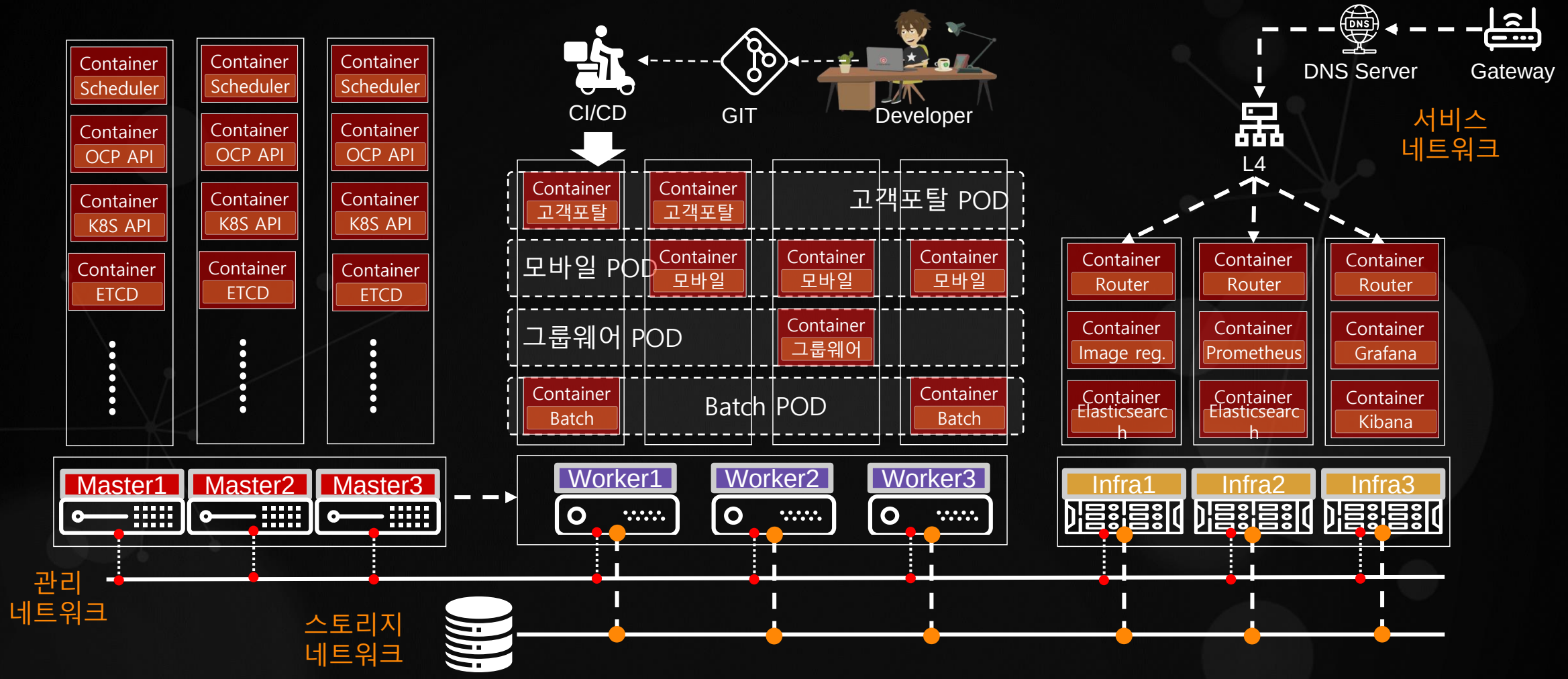
- 유연한 POD 스케줄링
- StatefulSet 으로 안정적인 POD 업데이트
- 서비스를 통한 연결 대상 자동 전환 및 부하 분산

- 수작업에 의한 장시간 조치 및 오류
- HA 솔루션을 통한 이중화
- DB 장애 시 전환 작업

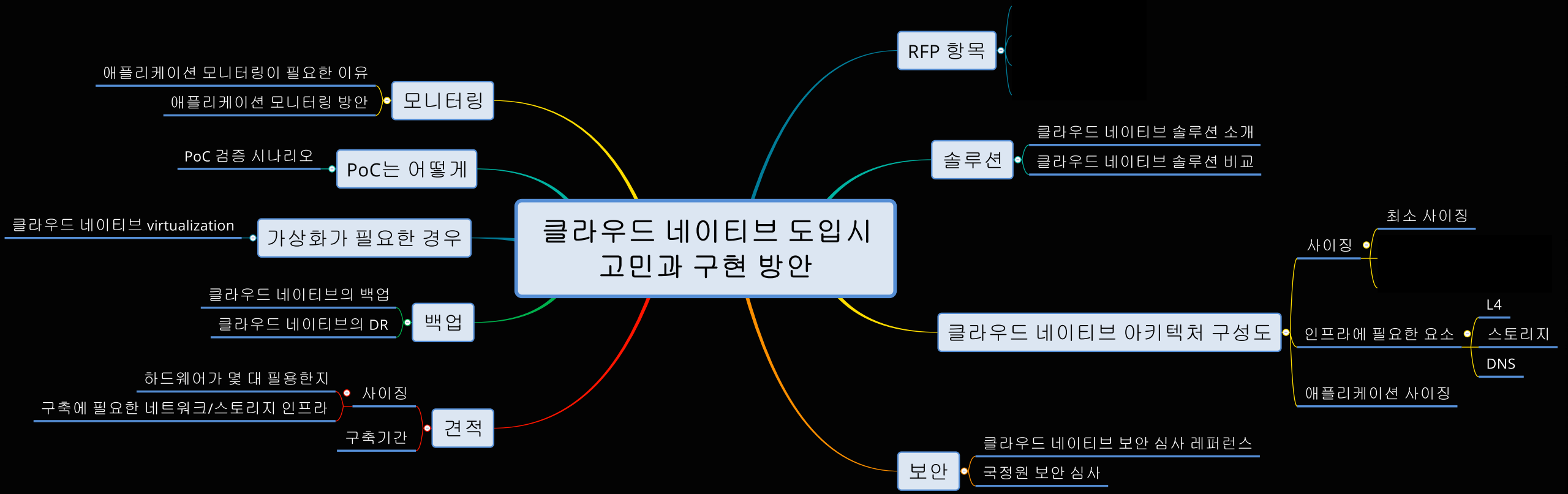




클라우드 네이티브 고가용성 구성



클라우드 네이티브 도입시 고민과 구현 방안




openmaru