클라우드 네이티브 before vs after

(주) 맨텍 이진현
jhlee@mantech.co.kr

manTech
better solution

CONTENTS

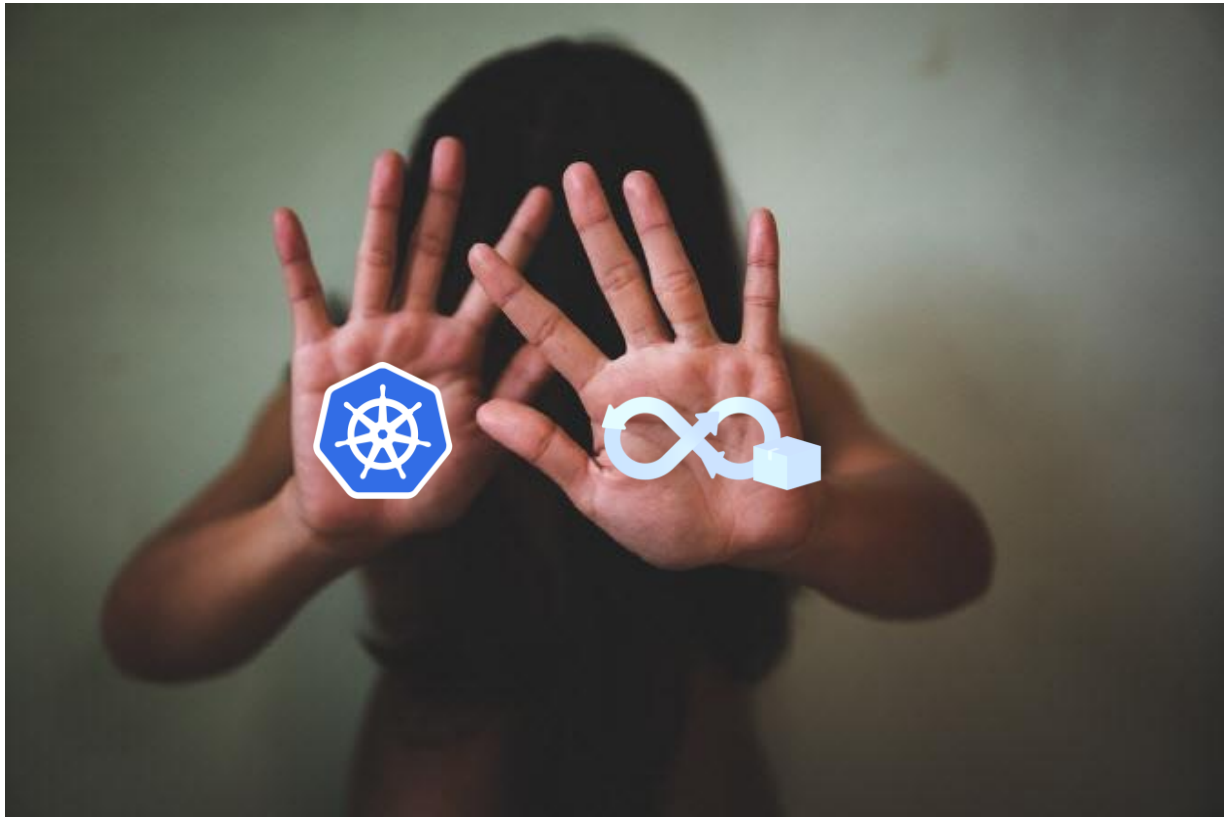
1

“ 클라우드 네이티브의 명암

선도기업들은 왜 클라우드 네이티브로 전환하는가?

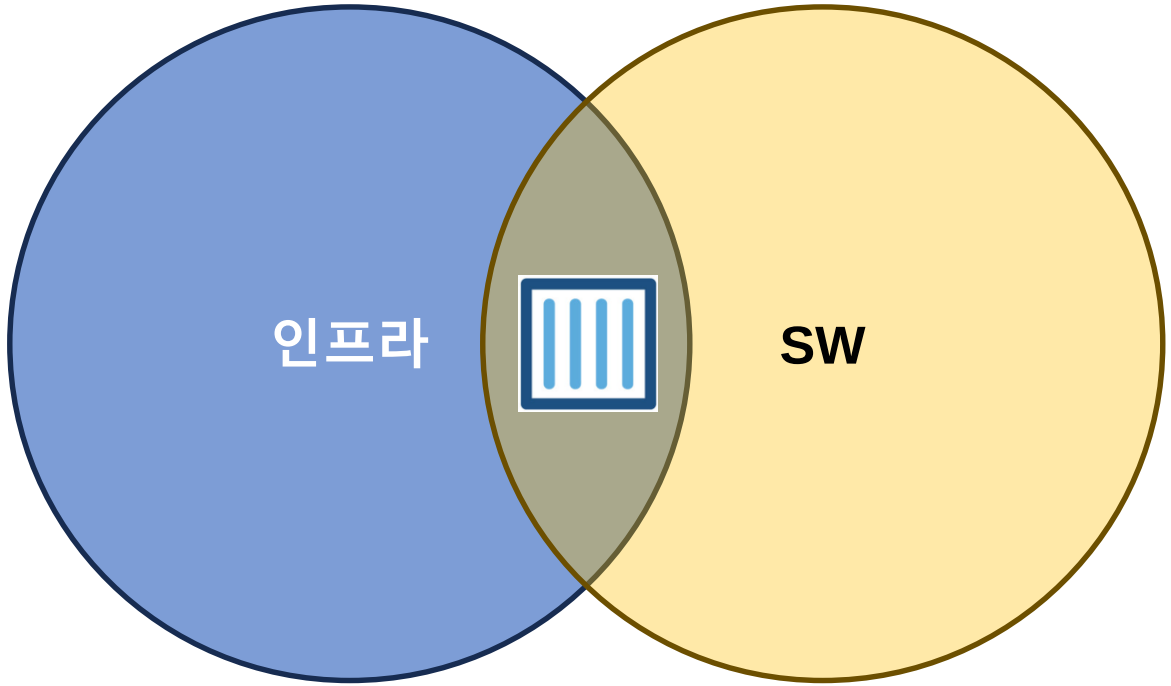
이렇게 좋은 것을 왜 주저하나?

- ✓ 이질감
- ✓ 기존 레거시의 익숙함
- ✓ 인식부족
- ✓ 인력부족
- ✓ 초보운전자의 사고 트라우마
- ✓ 방해공작



이질감

- ✓ VM은 확실히 인프라
- ✓ 컨테이너는 인프라인가?
  SW인가?
- ✓ OS가 없는데 관리를 어떻게?
- ✓ 데이터나 화일 저장이 안되네
- ✓ 운영 책임은 어떤 부서가?



기존 레거시의 익숙함

- ✓ 혁신의 가장 큰 걸림돌은 기존 환경의 익숙함
- ✓ 혁신적 솔루션의 목적은 편리함을 주는 것
- ✓ 문제가 있어도 익숙한 것을 더 선호하는 경향이 있음

인식부족

- ✓ 왜 해야 되는가?
- ✓ 우리에게 필요는 한가?
- ✓ 무엇이 좋아지는가?
- ✓ 그래서 어떻게 해야 되는데?

인력부족

- ✓ 수요자 중 아는 인력이 턱없이 부족
- ✓ 공급자 중 전문 인력이 턱없이 부족
- ✓ 도입 후 운영할 전문인력이 턱없이 부족

초보운전자의 사고 트라우마

- ✓ Inhouse 개발의 실패
- ✓ 검증되지 않은 업자를 통한 솔루션 신뢰도 영향
- ✓ 저가 사업 수주로 인한 질 낮은 서비스 품질
- ✓ 특정 솔루션 카테고리에 대한 부정적 인식 확산

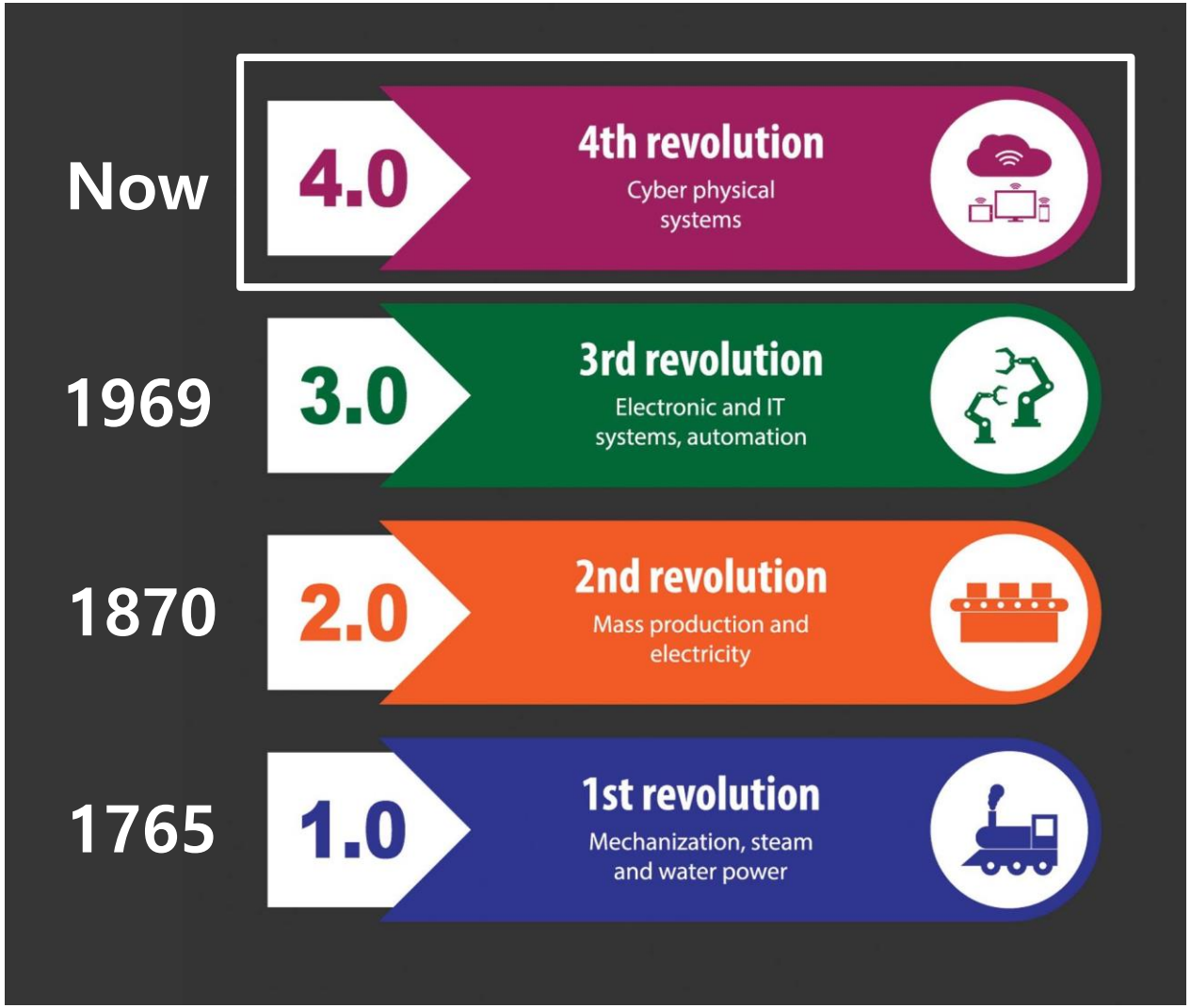
초보운전자의 사고 트라우마

방해공작

- ✓ 레거시 업자와 기존 유지보수 사업자들과의 이해 관계
- ✓ 모기업과 자회사 SI간의 이해관계
- ✓ 파괴적 혁신자로 인해 기존산업이 파괴되는 경험

그러함에도 불구하고 혁신의 불길은 못막는다

초연결 기반의 디지털 트랜스 포메이션의 모든 요소를 담을 수 있는 가장 현대화된 아케틱처는 **클라우드 네이티브**이다.

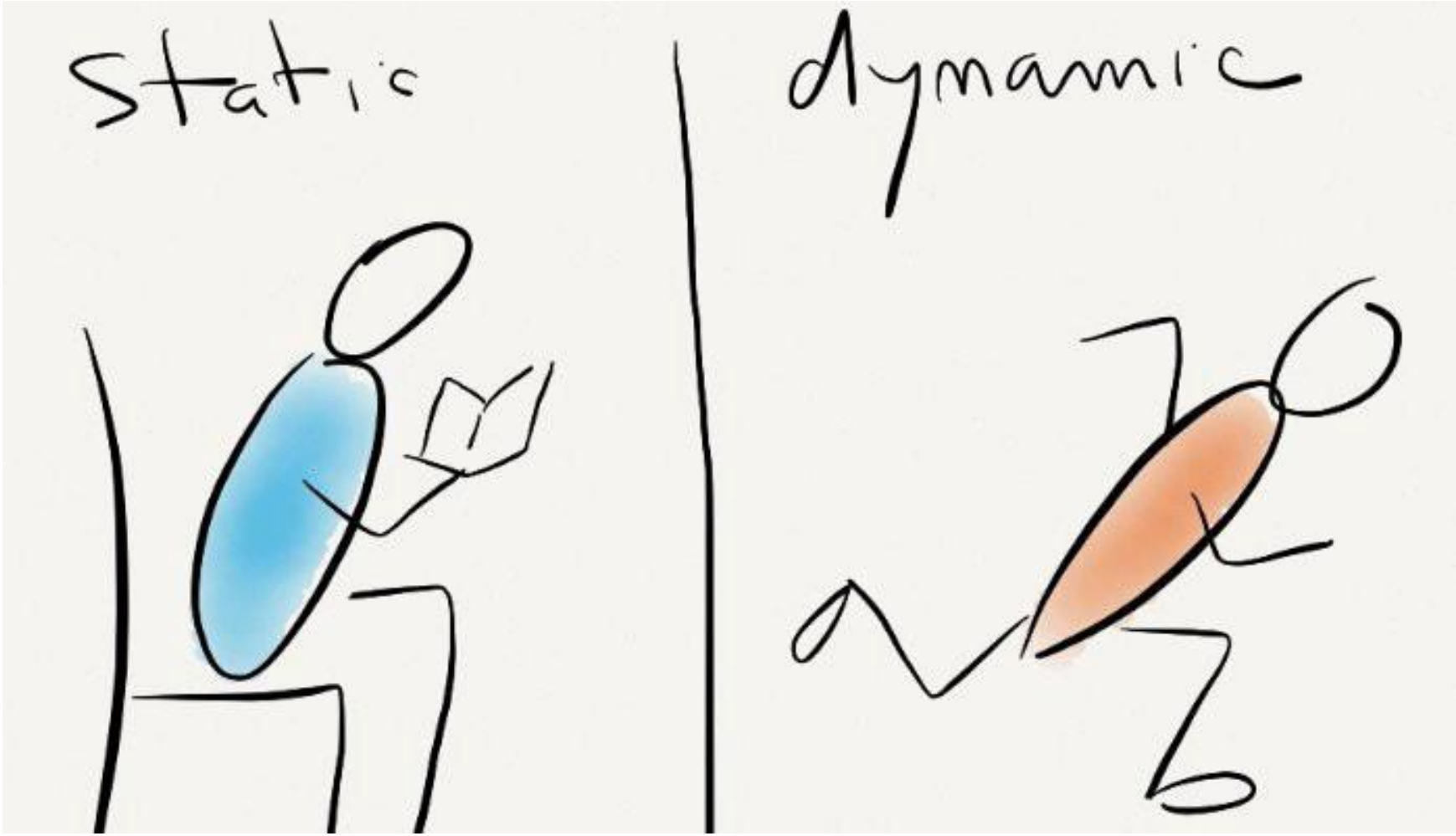


2

“ 레거시 vs 클라우드 네이티브

레거시

쿠버네티스



리소스 할당과 관리에 대한 비교

	레거시	클라우드 네이티브
IP주소	• 고정된 IP 수동 할당 • 서비스간 IP로 통신	• 가변적 IP 자동 할당 • 컨테이너 재시작 시 IP 바뀜 • 서비스 네임기반 통신
MAC주소	• 고정	• 재시작시 MAC 바뀜
Node Lock Key	• IP나 MAC에 바인딩 될 경우 네트워크 ID가 고정되어 있어 지속적으로 사용이 가능	• IP나 MAC에 바인딩 될 경우 컨테이너 재시작에 따라 Key값이 바뀌어야 함
CPU, Memory	• 독점적으로 할당량 사용	• 서버 자원을 공유하며, 컨테이너별 설정된 최대치내에서 사용
이동성	• 서버의 OS, Hypervisor가 동일해야 함 • IP대역이 다를 경우 APP 수정이 발생될 수도 있음	• OS 배포한, Hypervisor 등이 달라도 호환됨 • IP대역이 상이해도 네임기반 통신인 관계로 APP 수정 필요 없음

환경 변수의 영향도

레거시

OS와 앱의 환경변수에 종속

쿠버네티스

OS에 비종속적

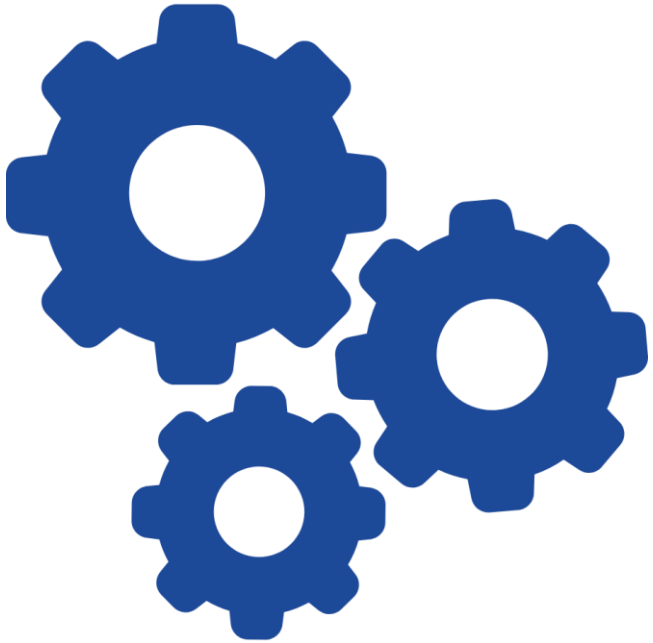
환경변수의 관리와 영향도

	레거시	클라우드 네이티브
OS 버전	• Host OS와 앱의 호환성 검증 필요	• Host OS는 K8S와의 호환성만 검증하면 됨 • 앱의 호환성 검증은 불필요
OS 패치 영향도	• 앱에 영향을 끼침	• 앱에 영향을 끼치지 않음
환경 변수 화일 관리	• OS 환경 변수 화일들 • 앱의 환경변수 화일들 • 각 OS와 앱마다 양식이 상이함	• Cofigmap과 Secrect으로 관리 • 앱의 종류에 관계없이 동일하게 관리
이동성	• 개발계, 테스트계, 운영계 별로 다양한 환경 변수를 수작업으로 변경해야 함 • 동일한 OS와 환경변수가 아닌 경우 앱의 정상 구동이 보장되지 않음	• 개발계, 테스트계, 운영계별로 환경변수를 다르게 관리할 수 있음 • Host OS버전이 달라고 앱의 정상 구동이 가능

레거시

반복적이고 복잡한 수작업

쿠버네티스



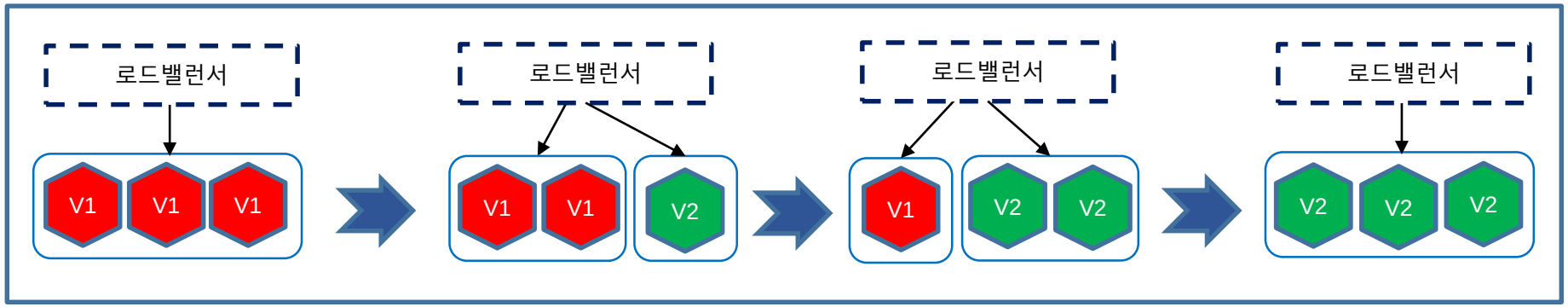
자동화된 빌드/배포/롤백

애플리케이션 배포 방식의 차이

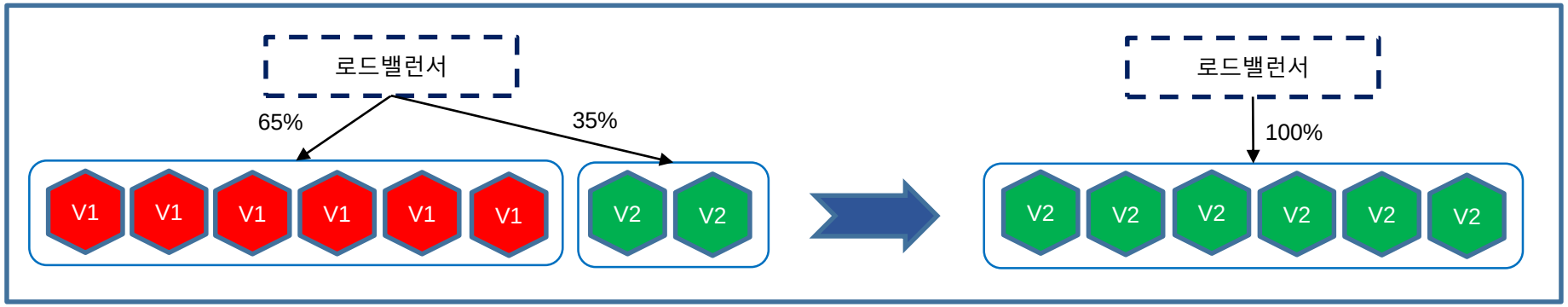
	레거시	클라우드 네이티브
빌드	• CI 솔루션을 통한 자동화 • 빌드된 앱의 버전 관리를 대체로 하지 않음	• CI 솔루션을 통한 자동화 • 앱의 버전 관리를 자동화
배포	• 배포 전문가에 의한 수작업이 주류 • 배포 시 기존 앱 다운 → 신규 앱 구동의 다운타임 발생	• CI/CD Pipeline을 통한 자동 배포 • 롤링 업데이트, A/B 배포, 카나리 배포 등의 무중단 및 고급 배포 전략을 세울 수 있음
롤백	• 수작업에 의한 롤백 (기존 서비스 중지 → 백업본 restore 등) • 장시간의 다운타임 발생	• 기존 앱 이미지로 2초 이내 롤백 • 무중단 롤백으로 다운타임 없음
이동성	• 서로 상이한 클라우드와 호스트에 배포 시 앱의 오류 발생 가능성이 큼	• 서로 상이한 클라우드와 호스트에 배포 가능 • 서로 다른 K8S 클러스터간 자동화된 배포 가능

클라우드 네이티브 상에서의 배포 전략

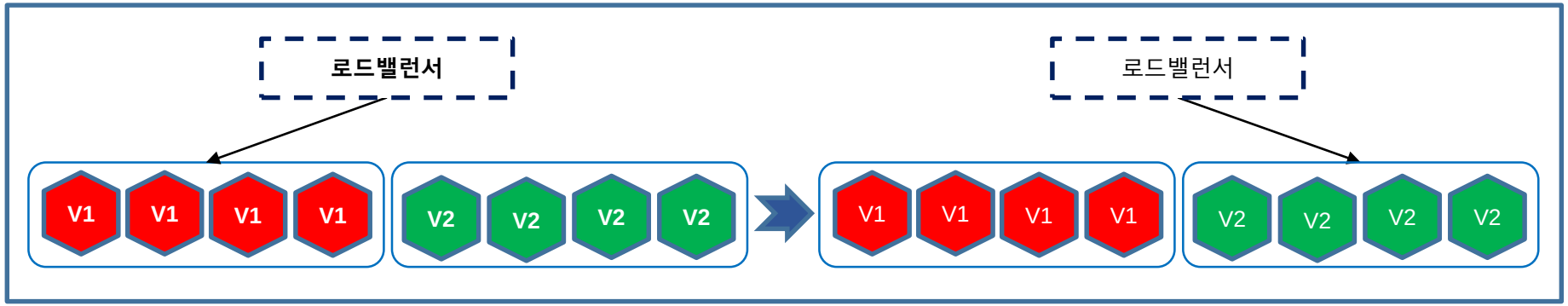
롤링업데이트



카나리 배포



AB 배포



서비스 장애 처리

레거시

무조건 복구

쿠버네티스

폐기처분 → 재생성

장애 처리의 방식

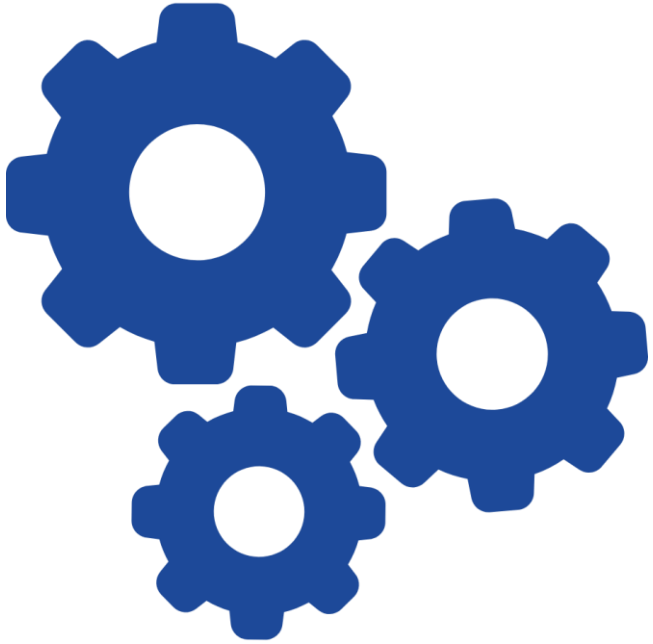
	레거시	클라우드 네이티브
장애 감시	• 별도의 솔루션 필요	• 별도 솔루션 필요 없음 • 노드, 네트워크, POD 등의 장애 자동 감시
장애 처리	• 별도 솔루션을 통한 앱의 재기동 혹은 대기서버로의 페일오버 • 수작업에 의한 복구	• POD를 자동으로 재기동 • POD의 Min값의 수 많큼 기동을 항상 보장 • Master 장애시에도 POD는 정상 구동됨
서비스 복구 시간	• 서버 재구동의 경우 수분~수시간이 소요될 수 있음 • Host Sandby로의 페일오버일 경우 수분~수십분 소요	• 컨테이너 자체는 수십초 내 구동 • 앱의 종류에 따라 수십초~수십분 소요
휴먼 에러 가능성	• 수작업 복구 시 엔지니어 스킬 레벨에 의하여 좌우됨	• 대부분 자동화된 복구로 휴먼 에러의 영향도가 거의 없음 • 고급 운영인력의 비용을 대폭 낮출 수 있음
재해복구	• 운영과 DR간 애플리케이션과 환경의 최신 변경 분 관리가 어려움 • 양 싸이트간 OS버전, 하이퍼바이저 다를 경우 복구 가능성 매우 낮음	• 최신 변경분에 대한 동기화가 매우 용이하며 자동화된 방식으로 변경관리 가능 • 양 싸이트간 OS버전, 하이퍼바이저가 달라고 서비스 복구 가능

서비스 확장

레거시

반복적이고 복잡한 수작업

쿠버네티스

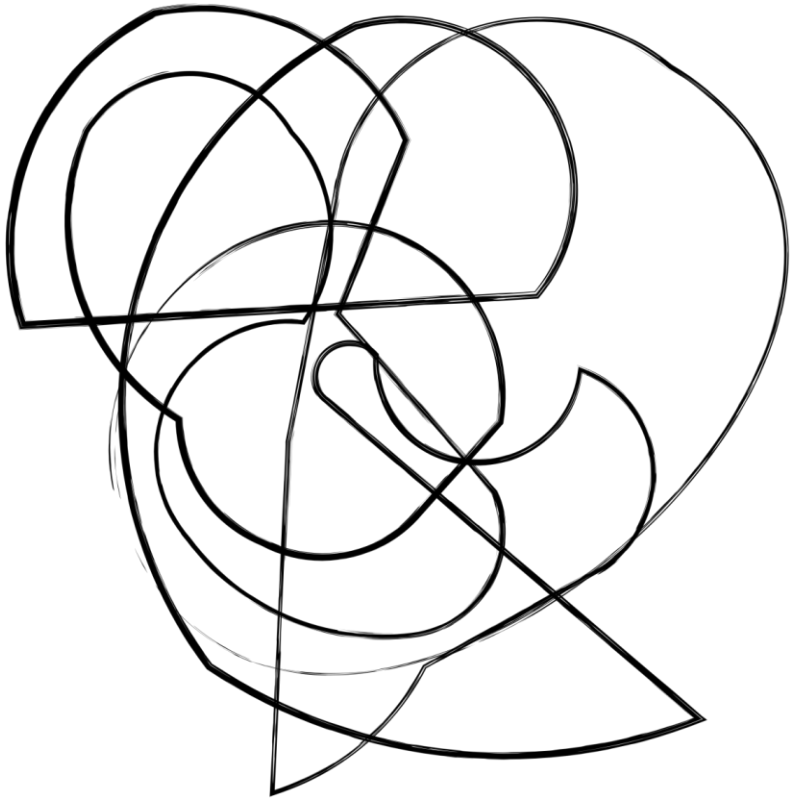


성능 수치에 따른 자동 확장

서비스 확장

	레거시	클라우드 네이티브
인프라 확장	• 물리서버의 경우 수작업 증설 • VM의 경우 자동 프로비저닝 제공 • 로드밸런싱 그룹에 Join	• 자동화된 POD 증설 • 증설된 POD에 대한 로드밸런싱 그룹 Join
애플리케이션 확장	• 서버가 자동 확장되더라도 애플리케이션은 수작업 설정 필요 • 로드밸런싱 그룹 Join도 수작업 설정 필요	• 애플리케이션 자동 확장 • 로드밸런싱 그룹 자동 Join • 자동 scale in 또한 가능
멀티 클러스터간 확장	• OS 및 Hypervisor가 다를 경우 호환성 체크가 먼저 선행되어야 함	• 자동화된 확장 가능

레거시



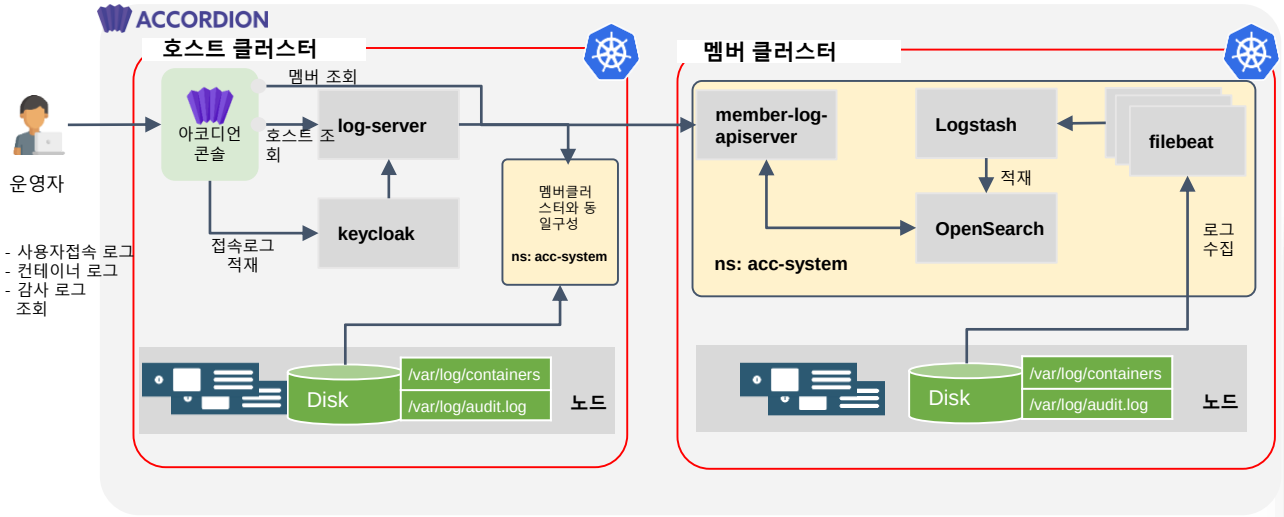
OS, 플랫폼, 앱별 복잡한 관리

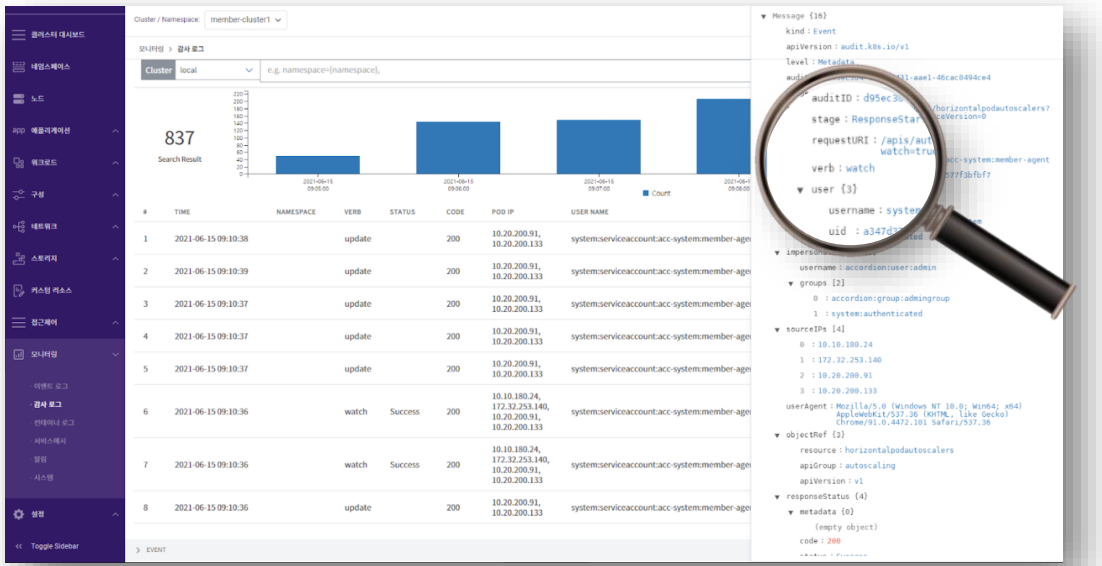
쿠버네티스

중앙 집중의 간단한 관리

로그관리

✓ 컨테이너는 물리 및 VM 서버 대비 1,000~10,000 이상의 가상 자원이 구동됨에 따라 개개별 로그 관리가 불가능





로그관리

	레거시	클라우드 네이티브
로그 저장소	• OS, SW, 인프라별 별도 저장 • 각 서버 별 로컬 디스크에 저장	• EFK와 같은 별도 통합 로그 저장소에 통합 관리 • 컨테이너 내부에는 저장 불가능
관리	• OS, SW, 인프라별 로그 수집, 검색, 분석 별도 개별로 진행 • 통합 관리를 위해서는 별도 개발과 프로젝트 진행이 필요하며, 각 요소별 호환성 맞추기가 쉽지 않음	• 앱 종류에 관계없이 대부분 컨테이너에 대한 로그 저장 및 관리로 해결 가능 • 앱, K8S 클러스터, 인프라 로그를 통합 관리 및 모니터링
분석	• 개개별 요소마다 전문 엔지니어를 통해 진행	• 일관된 방식으로 가시화된 분석 가능 • 분석과 운영에 대한 비용 절감

클라우드 네이티브 전환에 따른 환경의 변화 요약

	역할	AS IS	TO BE
개발	개발환경 구축	HW 구성	VM기반 리소스 할당 받기
		시스템 SW 구성	컨테이너 사용 권한 할당 받기
		형상관리, 빌드툴 구축 및 환경 구성	형상관리, 빌드툴 할당 받아 환경 구성
	CI/CD Pipeline	SCM→앱빌드→배포	SCM→앱빌드→**컨테이너 빌드**→배포
	신규 버전 배포 시 서비스 영향도	서비스 중단 발생	롤링 업그레이드를 통한 서비스 중단 최소화
		신규 버전에 대한 테스트 환경 별도 구성	별도 인프라 구축없이 카나리 배포, AB 배포, 미러링, 장애 유발 등의 다양한 테스팅 시나리오 구성
	앱 롤백	백업본 restore	원클릭 롤백
	다양한 OS 배포판 이식	추가 개발과 테스트 진행	필요 없음
운영	장애처리	HA 미구축 시 수작업 처리	자동 재기동
	서비스 확장	인스턴스 재구성 및 로드밸런싱 작업	Auto Scale과 로드밸런싱
	Host OS 업데이트와 패치	기존 앱의 영향도 고려하여 작업	기존 앱 영향도와 무관하게 작업
	필요 지식	HW, OS, VM, 시스템 운영 등	HW, OS, VM, **컨테이너, K8S**, 시스템 운영 등
	인프라 관리 역할	HW 접근 권한 통제, 가상자원 사용 승인과 할당	HW 접근 권한 통제, 가상자원 사용 승인과 할당, PaaS 플랫폼의 권한 관리/승인/할당

Thank You