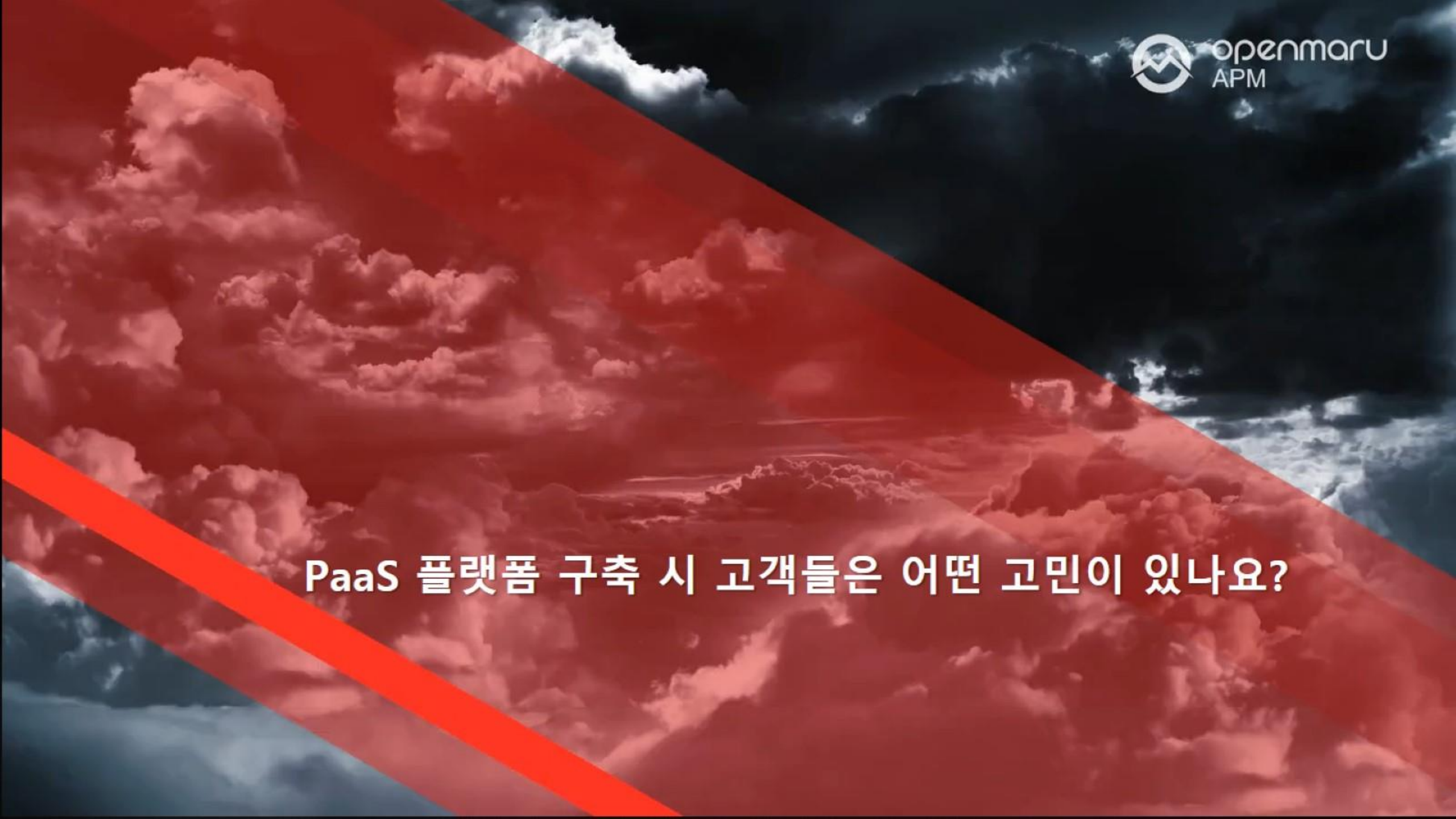

openmaru
APM
PaaS 플랫폼 구축 시 고객들은 어떤 고민이 있나요?


Container Orchestration
PaaS 구축 시 인프라는 어떻게 변경되나요?
openmaru
APM

PaaS를 운영할 때 필요한 인프라 리스트

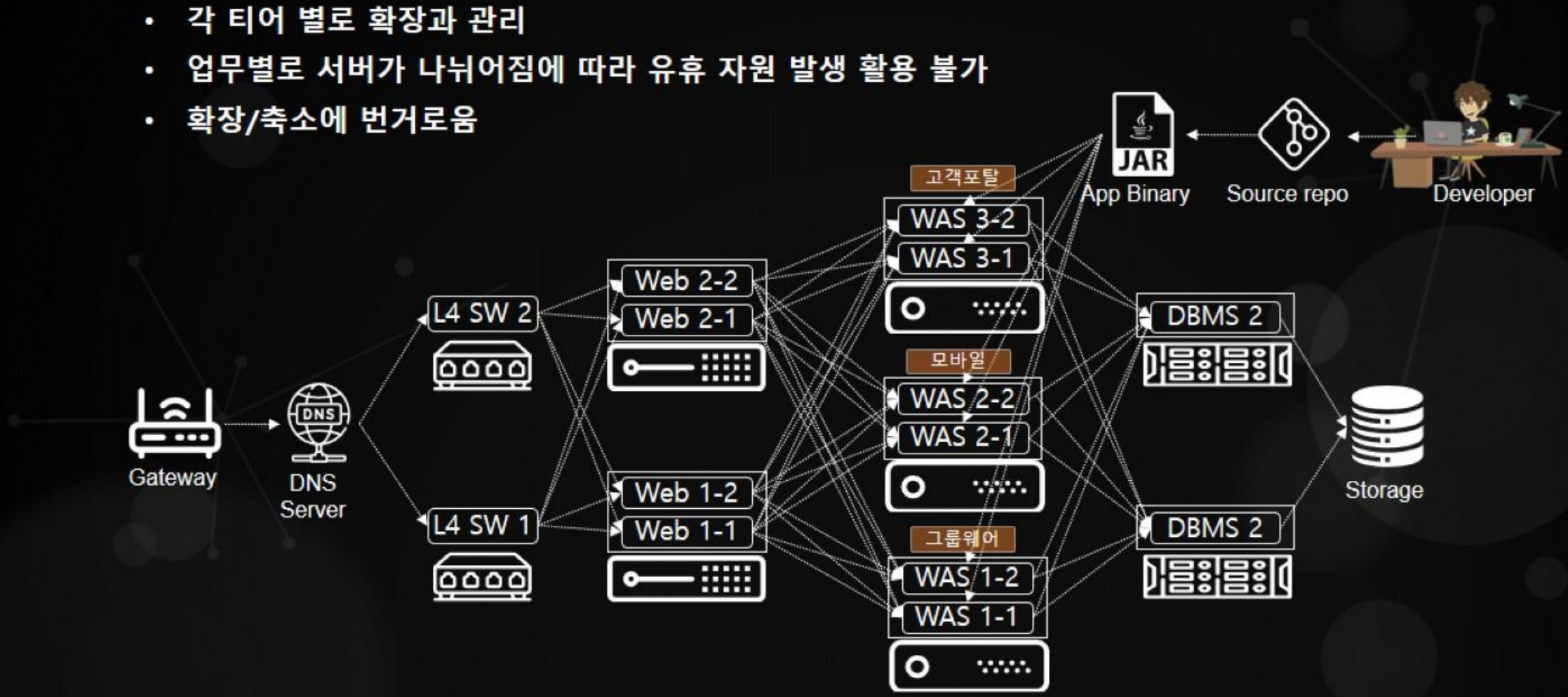
openmaru

- 고객분들의 오해 : 플랫폼이니까 PaaS만 설치만 하면 된다?

분류	이유	역할	비고
DNS	• 신규서비스 추가 시 배포와 함께 서비스 형태로 제공 되어야함 • 플랫폼 외부 네트워크에서 접속 시 도메인 기반의 애플리케이션 통신	• OpenShift Router Domain 등록	• New_App.apps.example.co.kr
L4	• 모든 Node가 Active 형태로 고가용성으로 구성됨에 따라 VIP, 헬스체크, 부하분산 필요	• Master Node, Infra Node VIP 및 로드밸런싱	
GIT	• 소스코드 저장방식의 고도화된 브랜치 전략을 가져갈 수 있음 • 소스코드만으로 컨테이너 이미지까지 빌드되는 OpenShift S2I 기능을 100% 활용	• OpenShift S2I(Source-To-Image)를 위한 소스 저장소	
Storage	• 애플리케이션들이 NAS처럼 활용 할 수 있는 공유 스토리지 • EFK, Monitoring, Image registry 등 OpenShift Infra 자원이 사용할 스토리지 공간	• 서비스가 이용할 공유 스토리지 및 OpenShift 운영 데이터 저장소	
Network	• 컨테이너 플랫폼을 구성하는 인프라 파드들과 애플리케이션 파드들도 기존 환경과는 달리 매우 많은 수가 기동되는 환경임에 따라 넓은 대역폭이 권고	• 10G 네트워크	

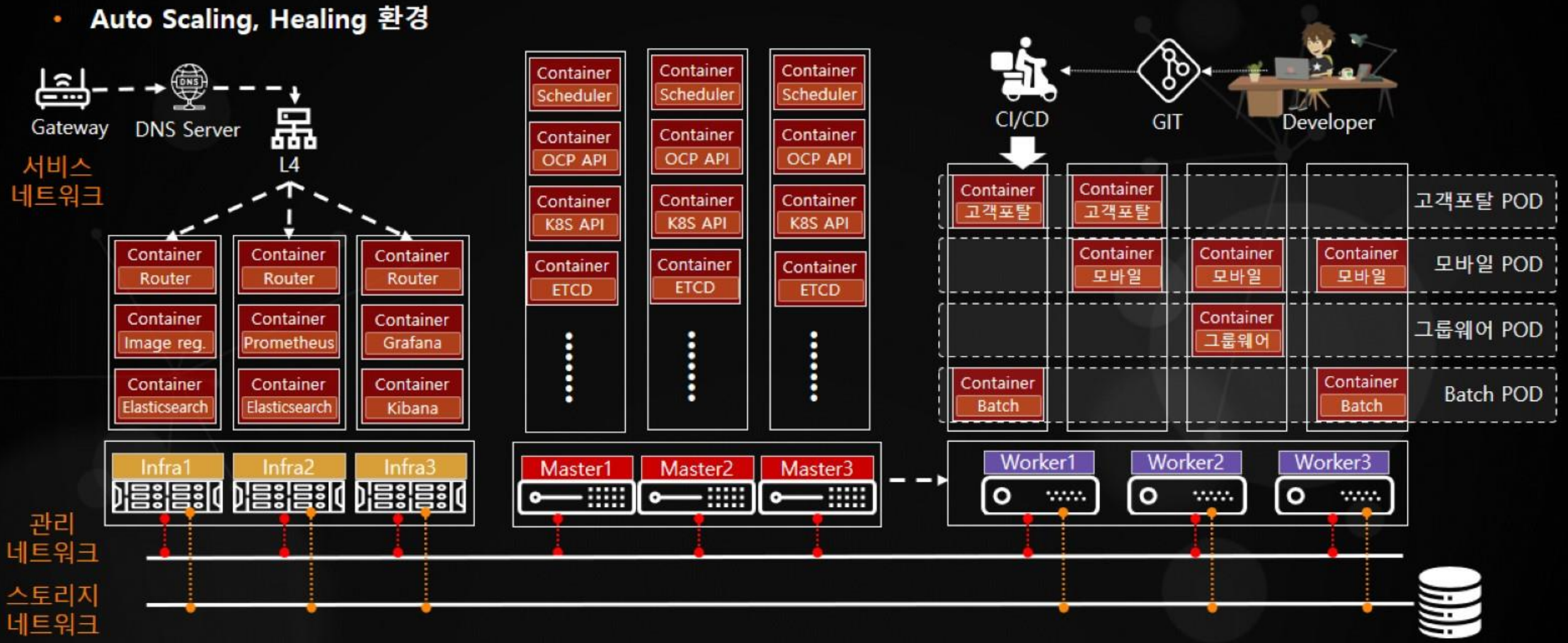
AS IS – 인프라 구성

- 웹서버, WAS 서버, 데이터베이스 의 역할 별로 티어를 나눈 3 티어 구조
  - 각 티어 별로 확장과 관리
  - 업무별로 서버가 나뉘어짐에 따라 유휴 자원 발생 활용 불가
  - 확장/축소에 번거로움



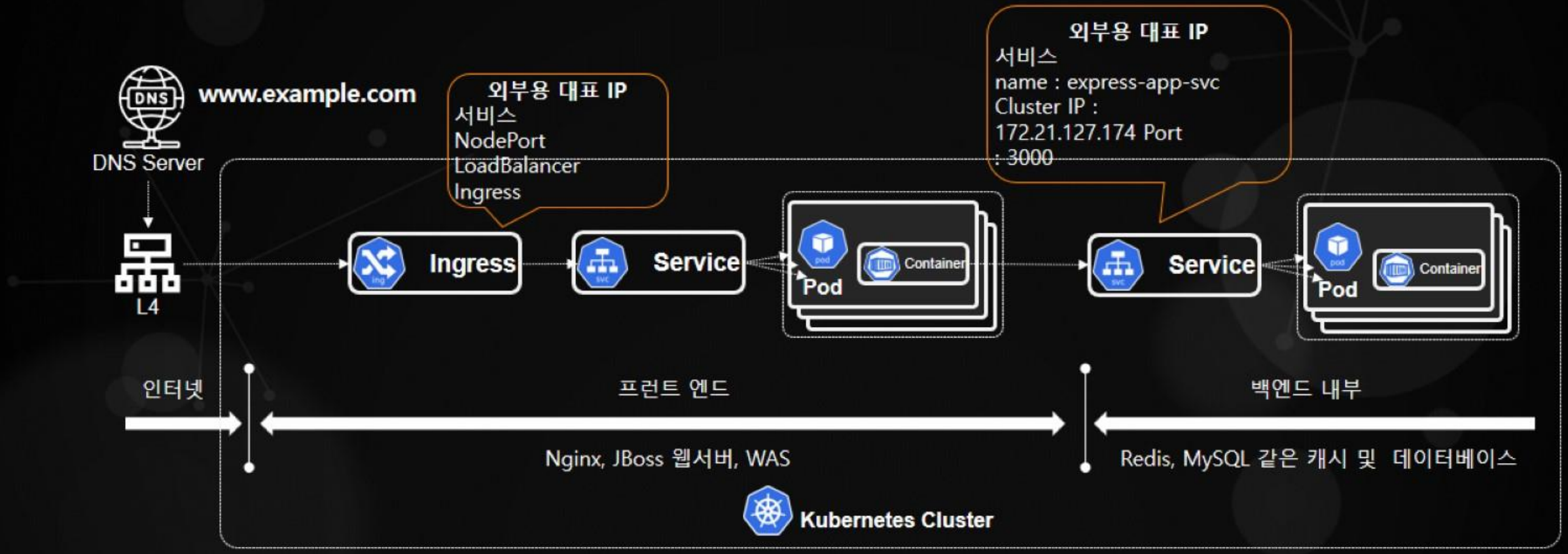
클라우드 네이티브 – 인프라 구성

- 서버별로 업무가 나뉘는 것이 아니고, 스케쥴러에 의해 **적재 적소 서버에 애플리케이션 배치**
- **Auto Scaling, Healing 환경**



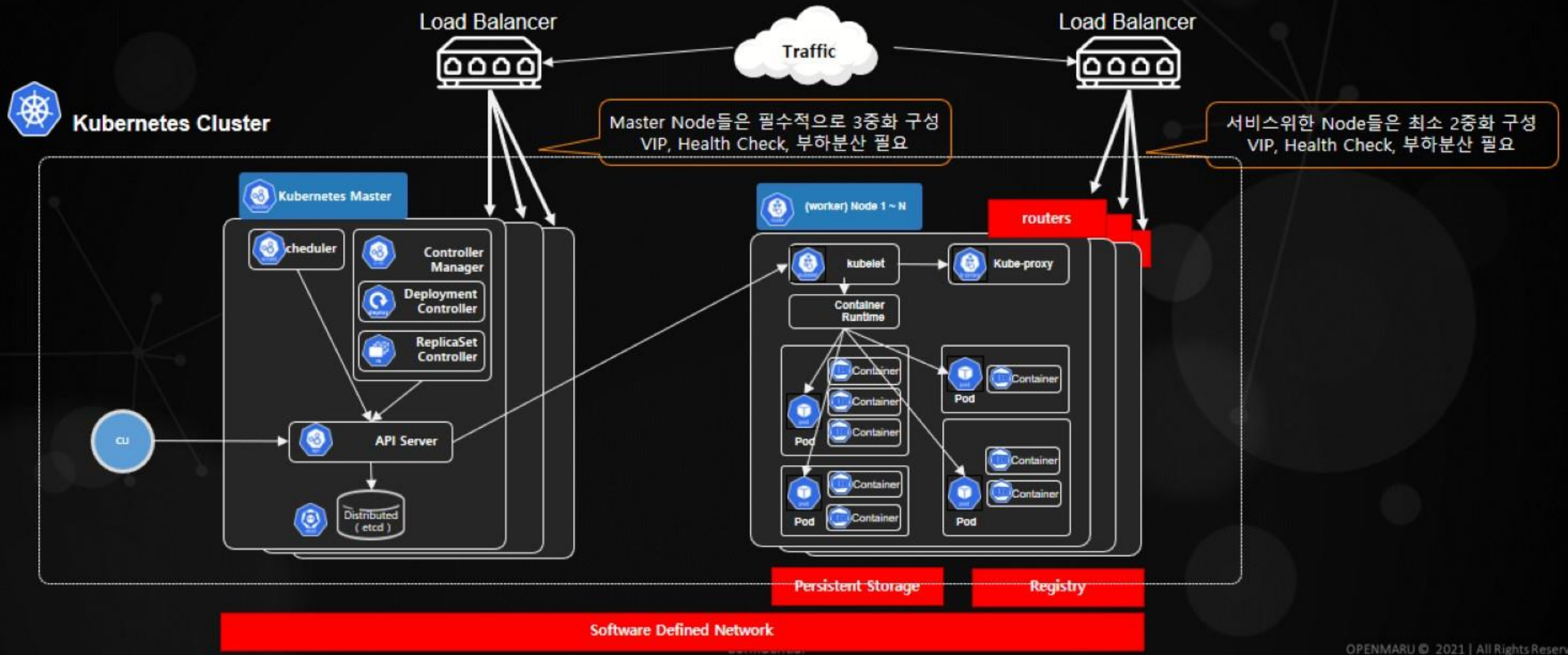
DNS는 왜 필요한가?

- 사용자들이 컨테이너 플랫폼 애플리케이션에 접속하기 위한 설정 Ingress / Route
- Domain 기반으로 OpenShift의 Pod로 들어올 수 있도록 하는 설정
- OpenShift의 애플리케이션은 Domain name 기반으로 통신



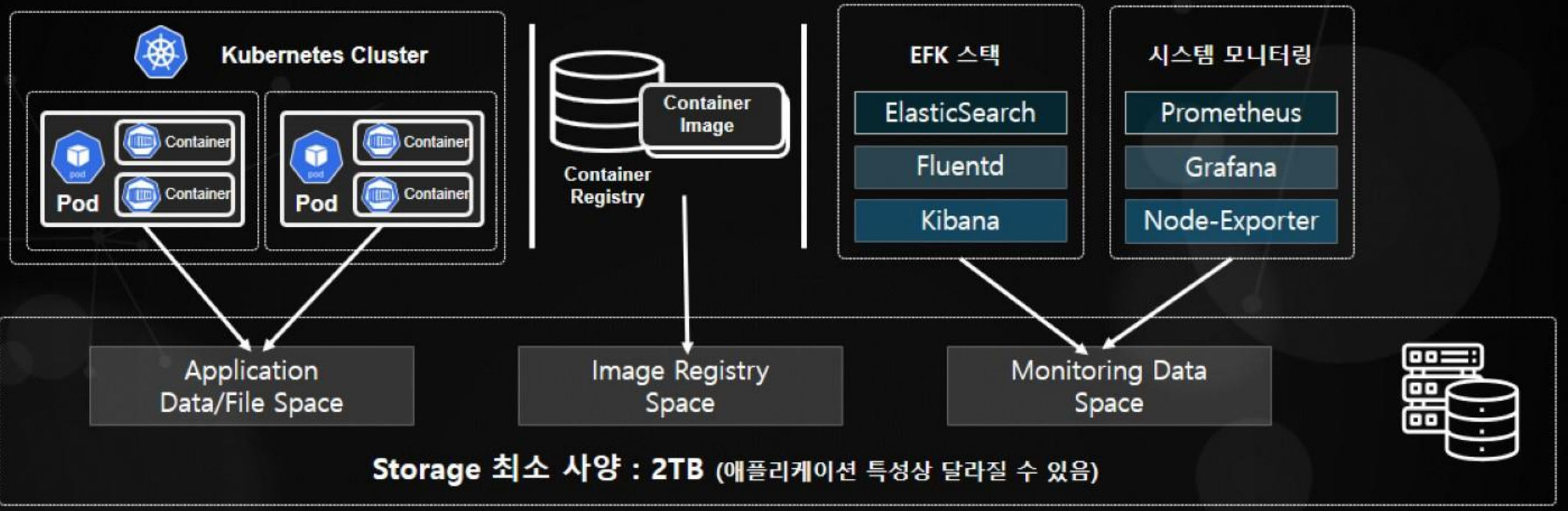
L4는 왜 필요한가?

- OpenShift의 Node들은 고가용성을 위한 다중화 구성
- OS 이중화와는 다르게 모든 Node가 Active 상태
- 그에 따라 IP를 묶어주는 VIP와 Health check, 부하분산 필요



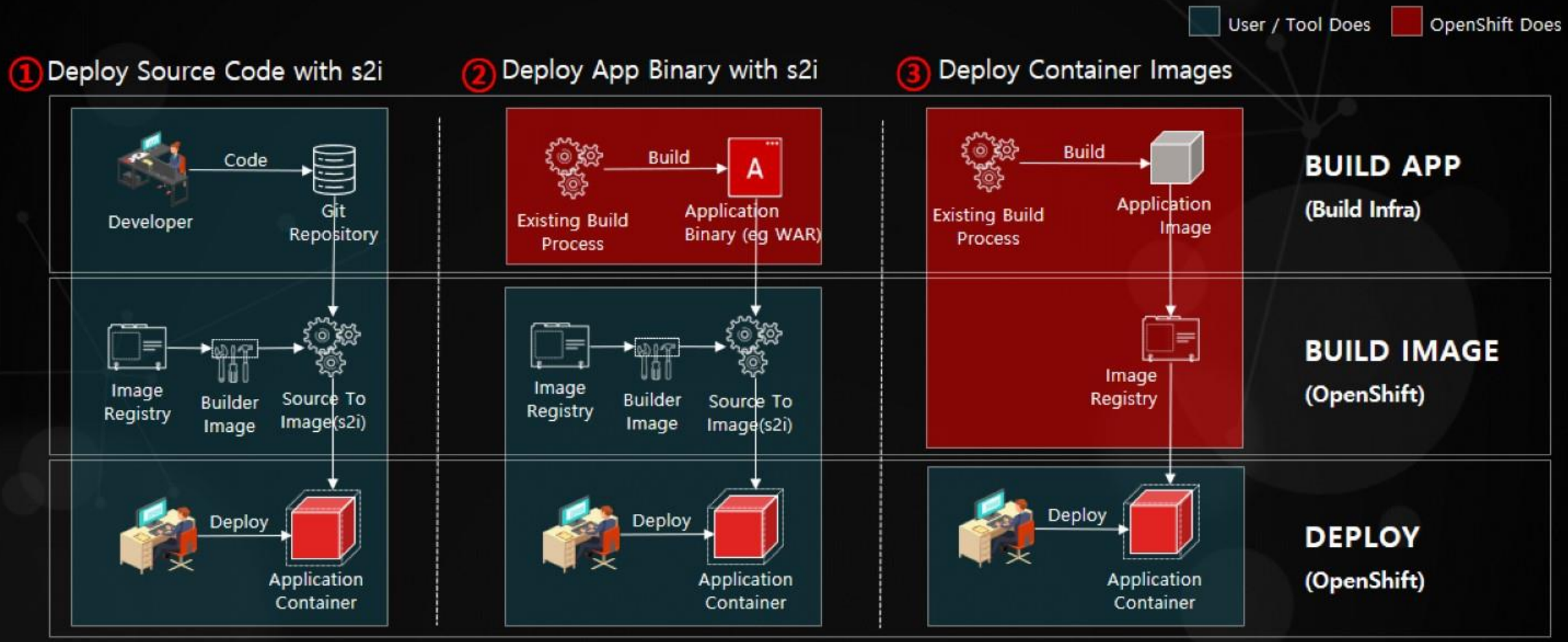
Storage는 왜 필요한가?

- Pod(애플리케이션)에서 NAS 용도로 활용하는 공간 (Upload 파일 등)
- 큰 용량을 가지는 라이브러리 들을 저장하는 공간
- 이미지를 저장시킬 레지스트리 공간
- 로그, 시스템 메트릭 수집 데이터를 저장시킬 공간



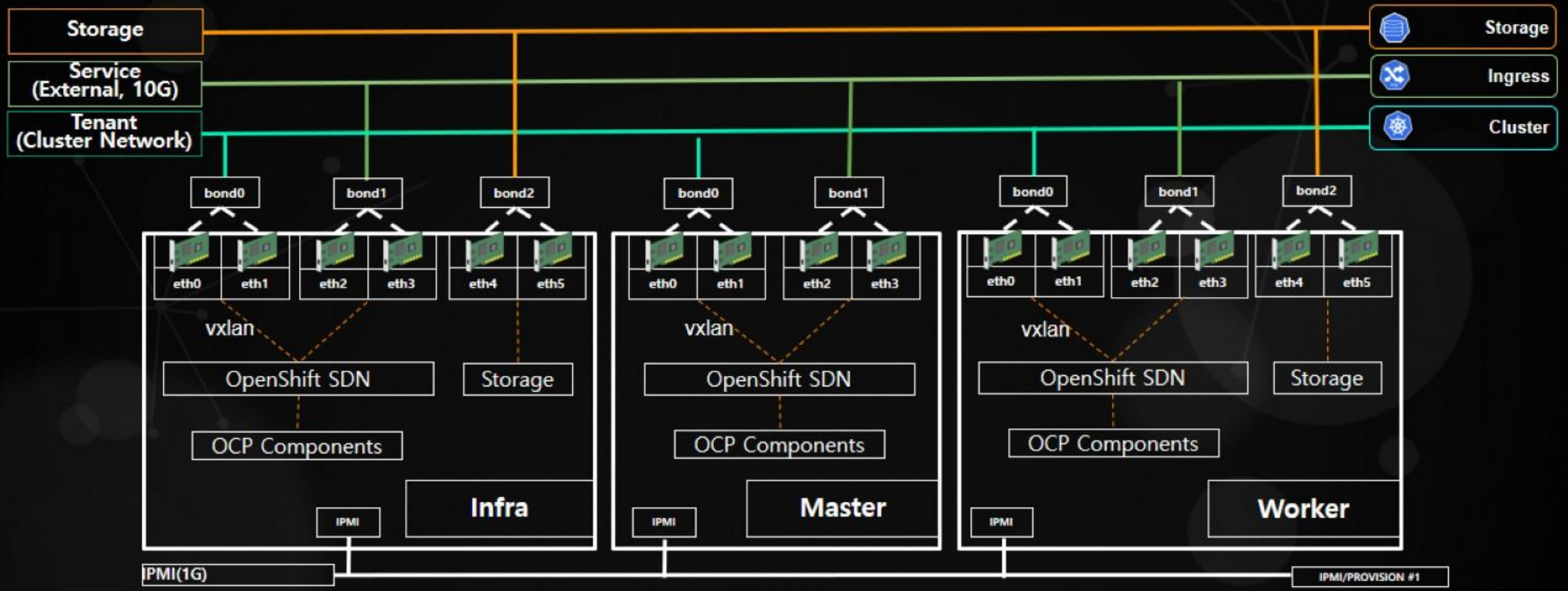
왜 OpenShift에 Git이 필요한가?

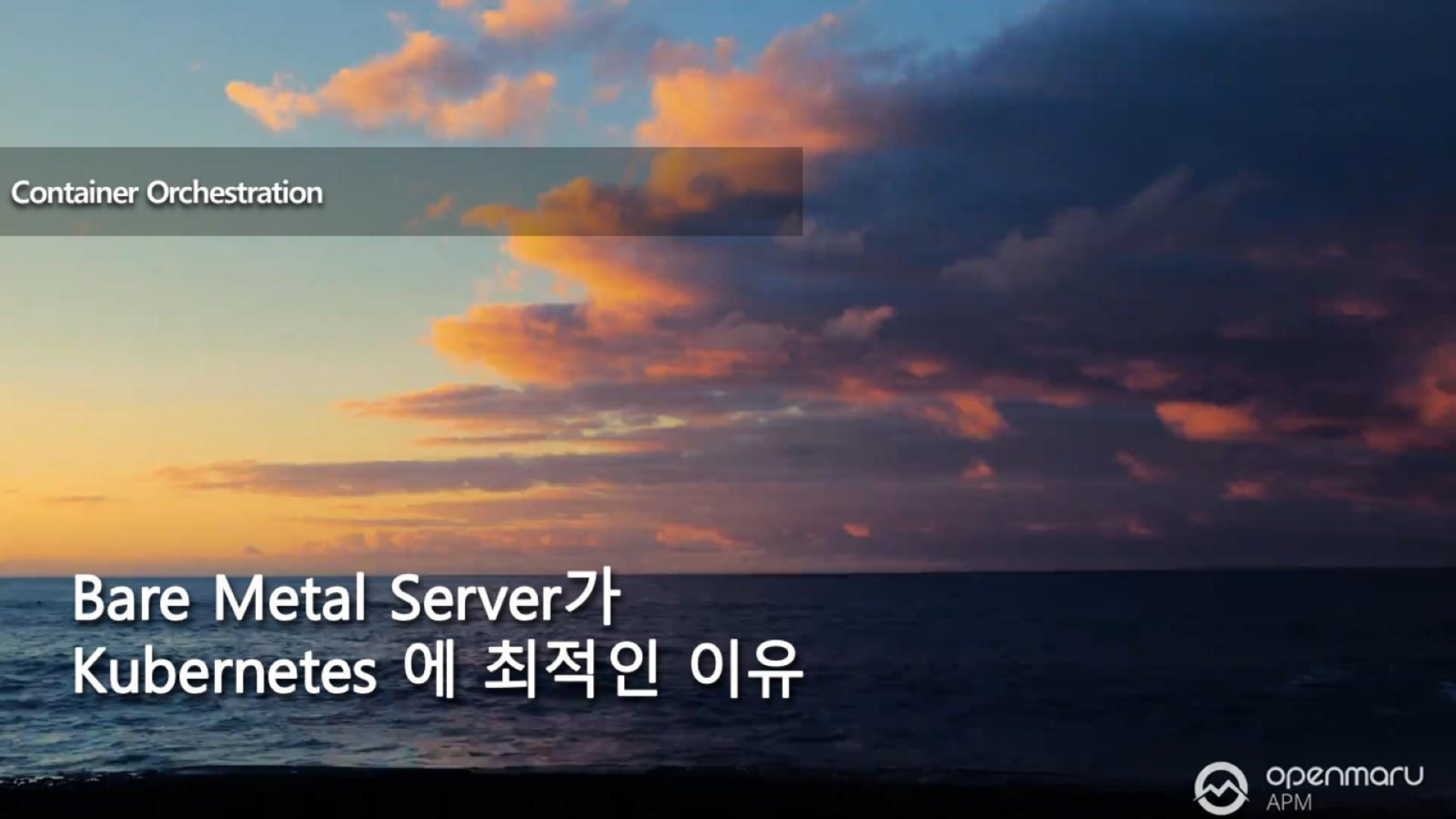
- 클라우드 네이티브 환경에 필수적인 강력한 분산기능, 확장성 : **Git**
- **Git에 소스만 저장시키면** 애플리케이션을 **OpenShift** 플랫폼에 배포시키는 **완전 자동 배포**



네트워크 구성에 대한 최적 안

- 필수적으로 필요한 네트워크(서비스),옵션으로 필요한 네트워크(스토리지,관리)
- 컨테이너 환경은 수많은 컨테이너가 엮인 형태
- 그에 따라 10G 네트워크 권장사항.




Container Orchestration
Bare Metal Server가
Kubernetes 에 최적인 이유
openmaru
APM

Bare Metal과 HCI의 구성의 특징과 비교 분석

openmaru

Bare Metal	VS	HCI
스위치(네트워크), 서버(컴퓨트), 스토리지로 구성된 전통적인 인프라 구성 방식	구성 개요	동일한 모델의 HCI 장비 여러 대를 묶어 하나의 인프라로 구성하는 방식
어떠한 Baremetal의 장애가 다른 Baremetal에 **영향을 끼치지 않음**	**장애 대처 난이도**	장애 발생시 **장애 구간 파악이 어려운 편**에 속하며 **하나의 HCI 장비에서 Software 방식으로 여러 역할을 수행하기** 때문에 Critical한 장애 발생시 파급력이 큼
개별 장비 추가로 확장성 확보	인프라 규모 확장성	확장에 제한적인 제품군이 존재
HCI에 비해 상대적으로 러닝커브가 적음	**운영 난이도**	**익숙하지 않은 운영자에게는 운영이 어려울 수 있음**
큰 어려움이 없음	신규 구성 난이도	**고려사항과 제약이 큰 편**
별도로 Resource를 차지하는 부분이 없음	Computing Power	**해당 CVM이 Compute Power를 점유함**
구매 후 장비 Warranty, Hypervisor 라이선스와 같이 **구축에 필수적인 라이선스 외 별도의 추가적인 라이선스는 필요 없음**	**라이선스 비용**	**HCI 장비의 연간 갱신 라이선스 및 환경 구성에 필요한 추가적인 라이선스**가 요구됨(ex. 용량 라이선스) Ex) Nutanix, VxRail, Simplvity, VxFlex 등
추후 다른 인프라 구성에 **추가하여 사용 가능**	**기존 인프라와 호환성**	**기존 사용 중이던 인프라의 Scale-Out에는 적합하지 못할 수 있음.**

Bare Metal 구성의 장점

구간별 필요한 장비의 수량 추가를 통해 **Scale-out**, 장비의 스펙 상승을 통한 **Scale-up 모두 가능한 확장성 제공**

HCI 자체를 유지하기 위한 리소스가 필요, Bare Metal은 불필요하게 Resource를 낭비하는 요소 없이 **온전한 성능을 제공할 수 있는 실용적인 구성.**

장애 구간의 파악에 큰 어려움이 없고 오랜 시간동안 쌓인 Knowledge Base를 통해 **빠른 대처와 운영 난이도를 낮추어 용이성 확보.**

네트워크, 스토리지, 서버 등 장비의 용도에 따른 구성 변경이 가능하며 **기존 사용자의 인프라와 높은 호환성을 지님.**

각 벤더 별 고도화된 서비스를 통한 전문적인 기술력을 바탕으로 서버, 네트워크, 스토리지 장비 개별적 부하구간을 파악하여 상응하는 맞춤형 인프라 도입이 가능.

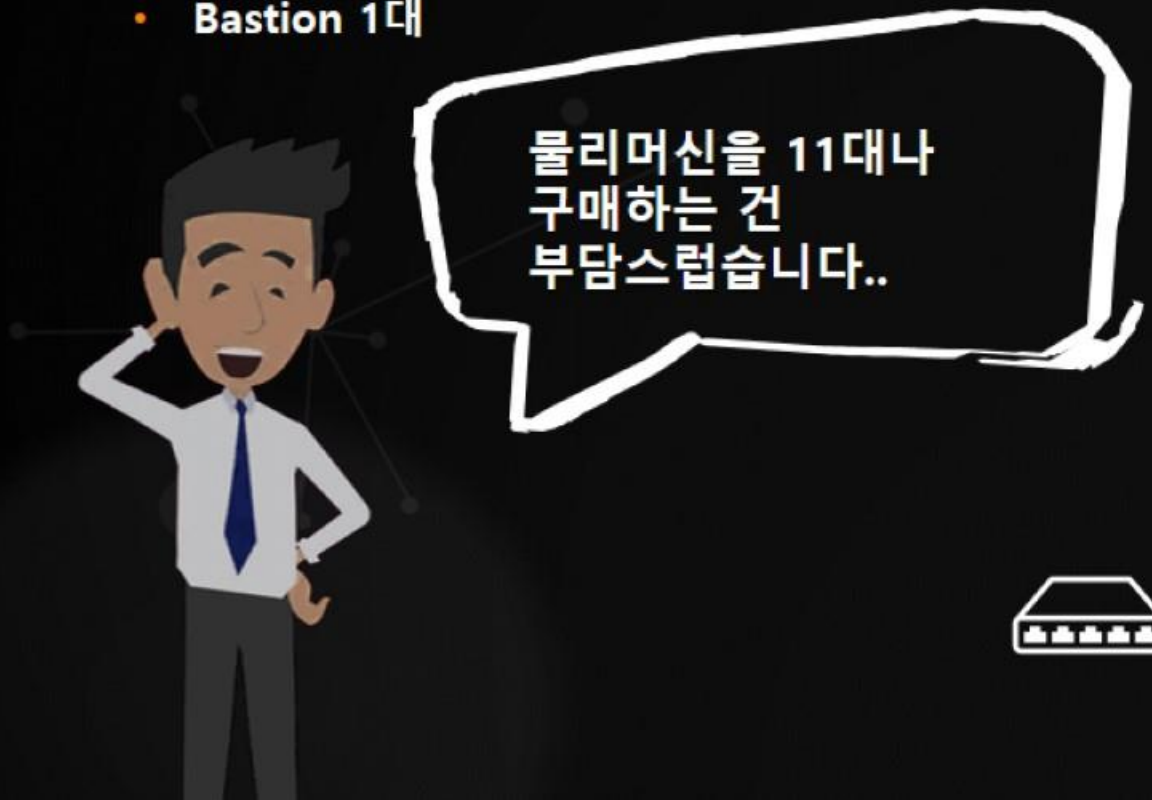
연간 갱신 라이선스와 **별도의 부가적인 라이선스에 필요한 지출을 최소화**하여 TCO 감소, 운영을 위한 교육비용 절감을 통한 경제성 창출.

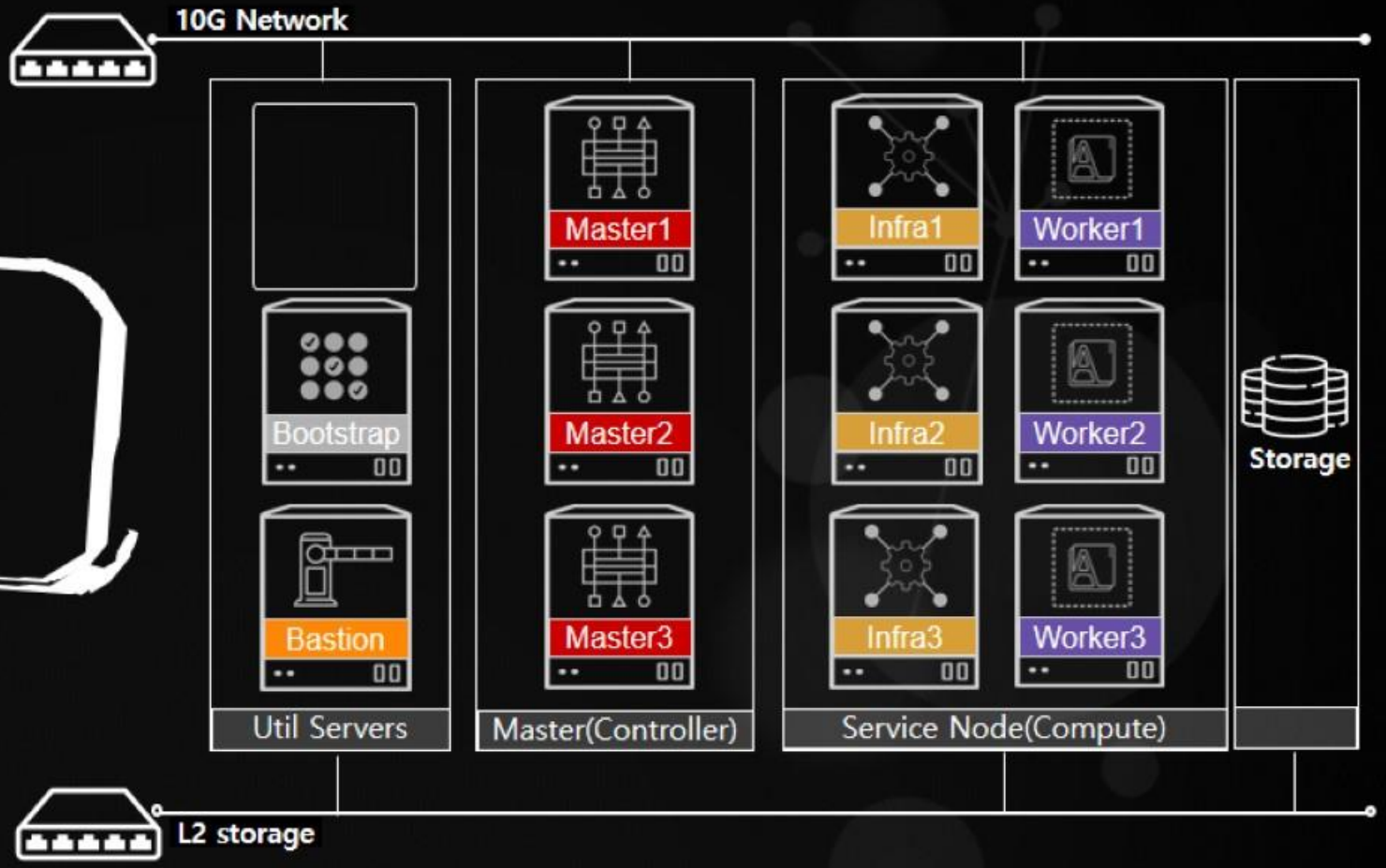
그럼 HCI나 가상화는 PaaS 플랫폼에 완전히 필요 없을까?

openmaru

OpenShift의 필요 물리 서버대수 11대

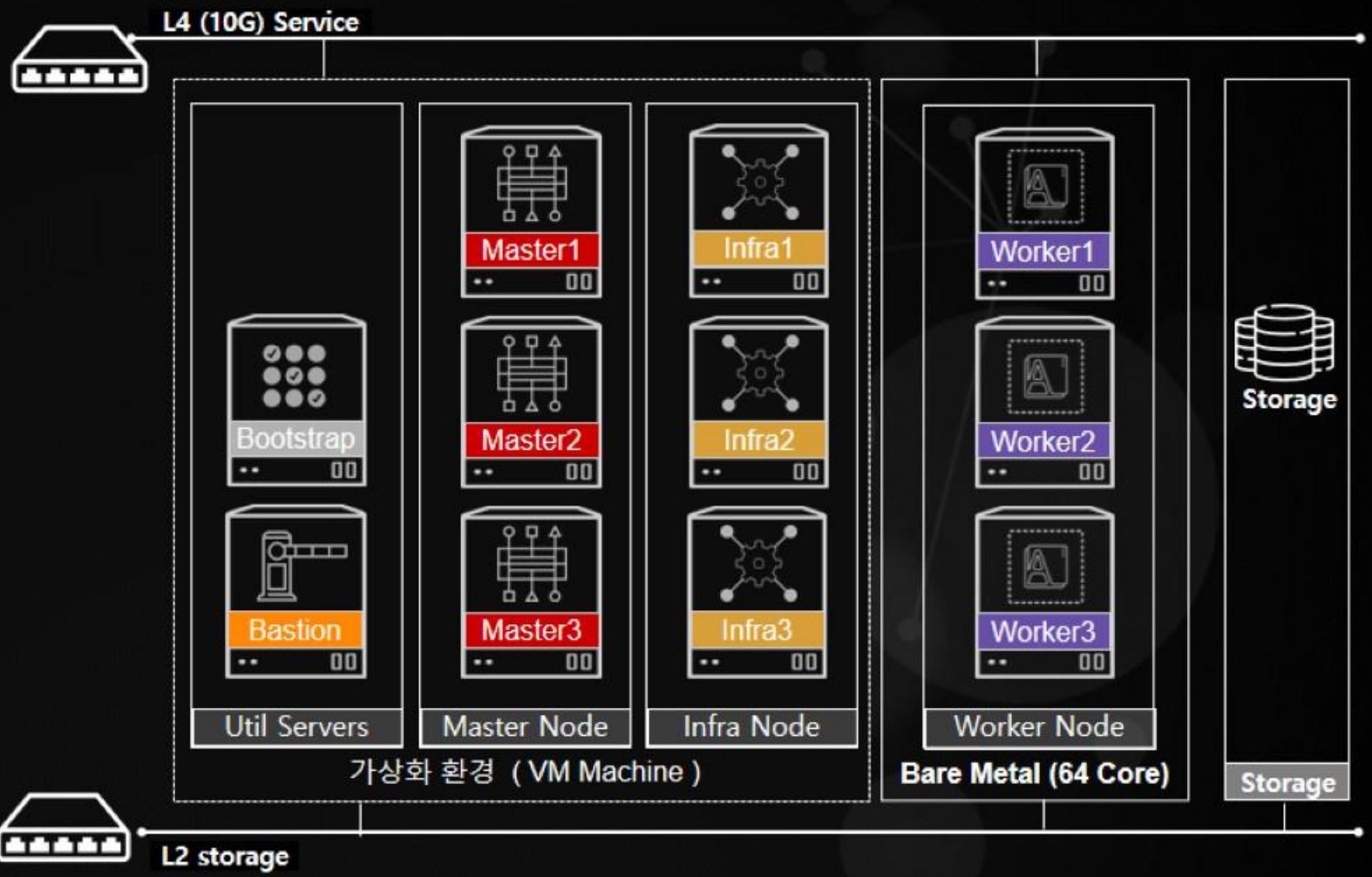
- Master Node 3대
- Infra Node 3대
- Worker Node 3대
- Bootstrap 1대
- Bastion 1대





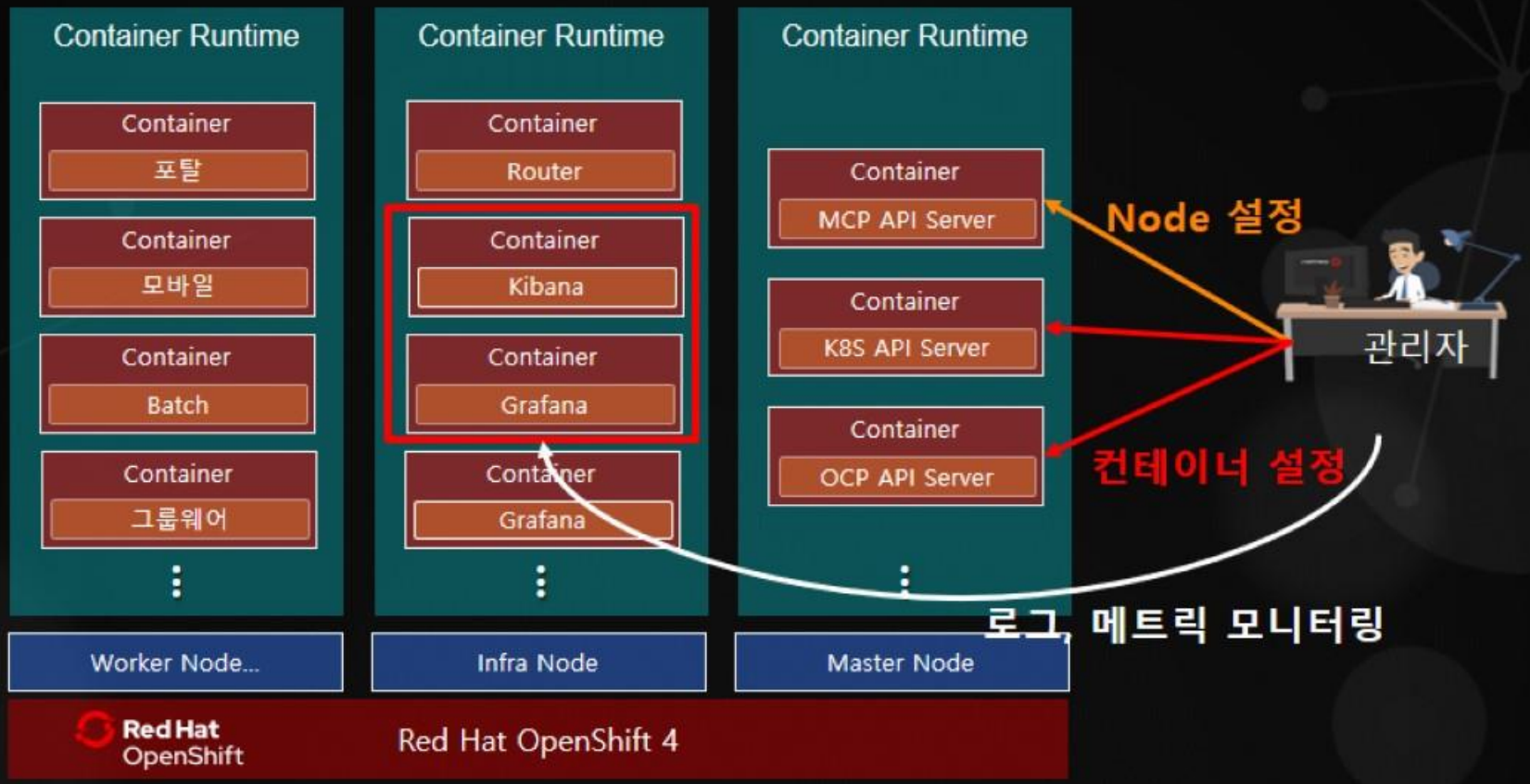
Bare Metal과 VM을 활용한 최적 방안 하드웨어 구성

- OpenShift의 필요 물리 서버대수 6대
  - HyperVisor(가상화 솔루션) 3대
  - Worker Node 3대
- 가상화 환경
  - 컨테이너 운영관리를 위한 서버들
    부하가 상대적으로 적음
  - Master / Infra Node
  - Bootstrap
  - Bastion
- 물리 머신 환경
  - 실제 서비스를 수행할 서버
    부하가 상대적으로 큼
  - Worker Node



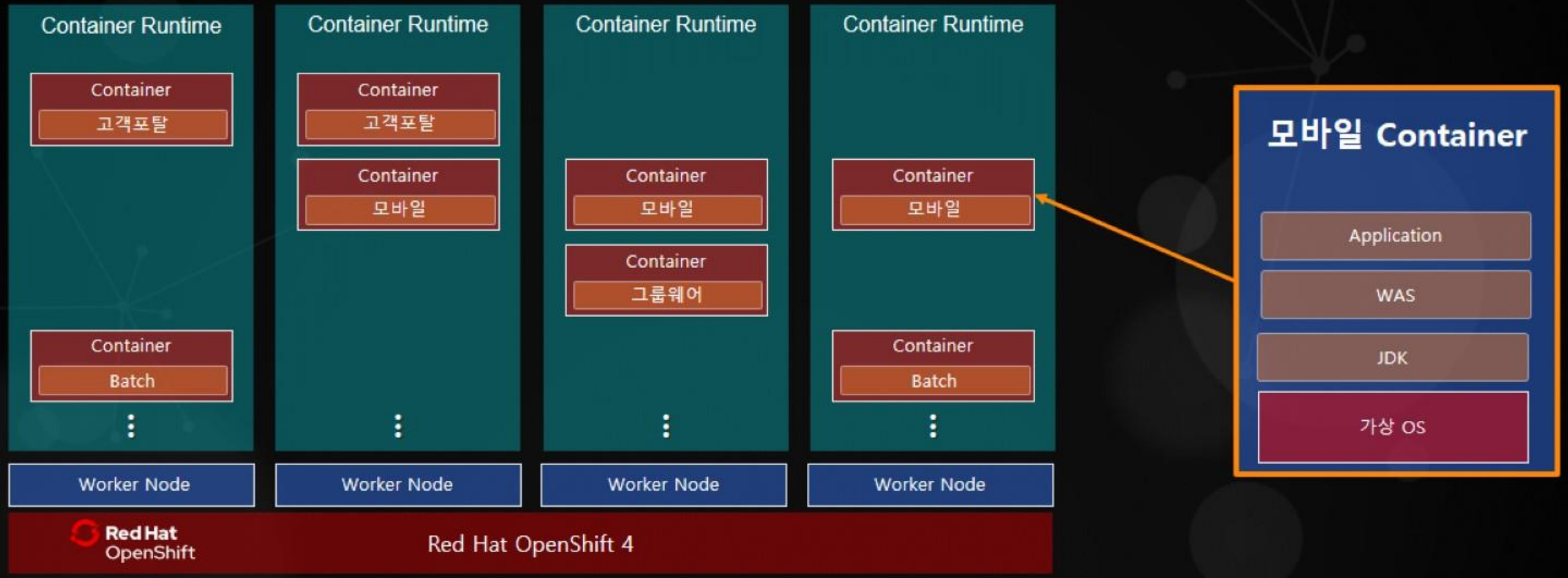
OpenShift 소프트웨어 컴포넌트 요약 구성

- OpenShift의 플랫폼을 유지/관리/구성하는 Master, Infra Node
- Master Node에 컨테이너, 머신 설정 명령
- Infra Node를 통해 라우팅, 모니터링



OpenShift 소프트웨어 컴포넌트 요약 구성

- 실제 서비스가 동작하는 Worker(서비스) 노드
- Scheduler에 의해 적재적소의 Worker Node에 서비스(애플리케이션) 배치
- Container는 가상OS, JDK, WAS, Application이 포함된 Image를 기반으로 기동됨.

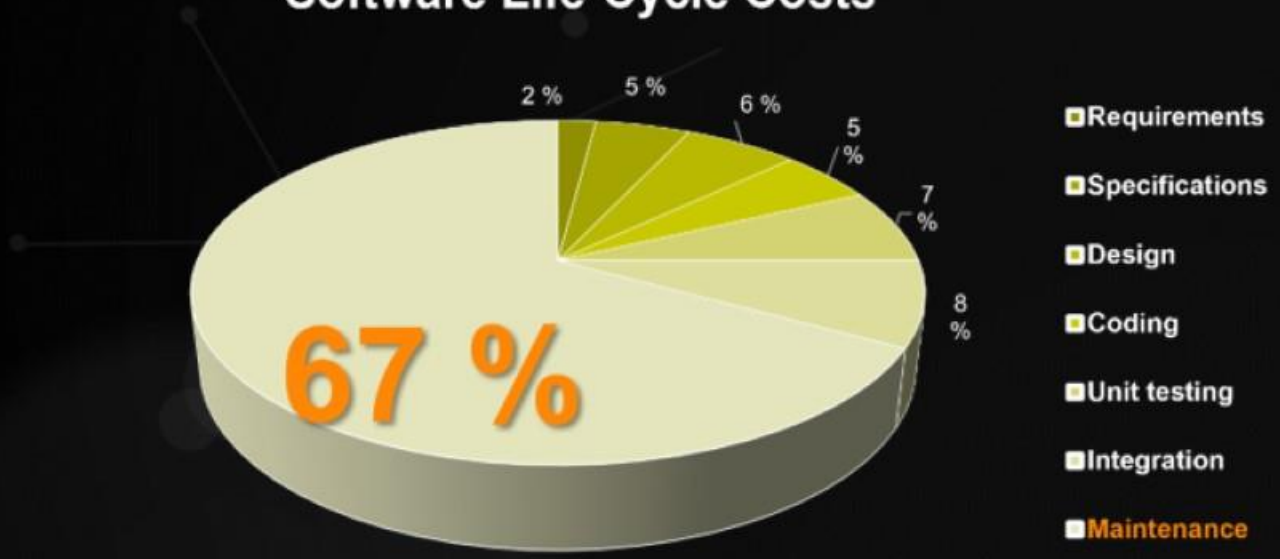



Application Performance Management
SRE가 무엇인지 아시나요?

Software Engineer - Maintenance**

- 소프트웨어 엔지니어링과 육아의 공통점
  - 출산 이후에 더 많은 노력이 필요
  - 소프트웨어 엔지니어링도 제품 출시 이후가 중요
    - 시스템 총비용 중 약 40%~90%가 제품을 출시한 이후 발생 (Maintenance)

Software Life-Cycle Costs



Source : Digital Aggregates

10억 명 이상의 클라우드 서비스

openmaru

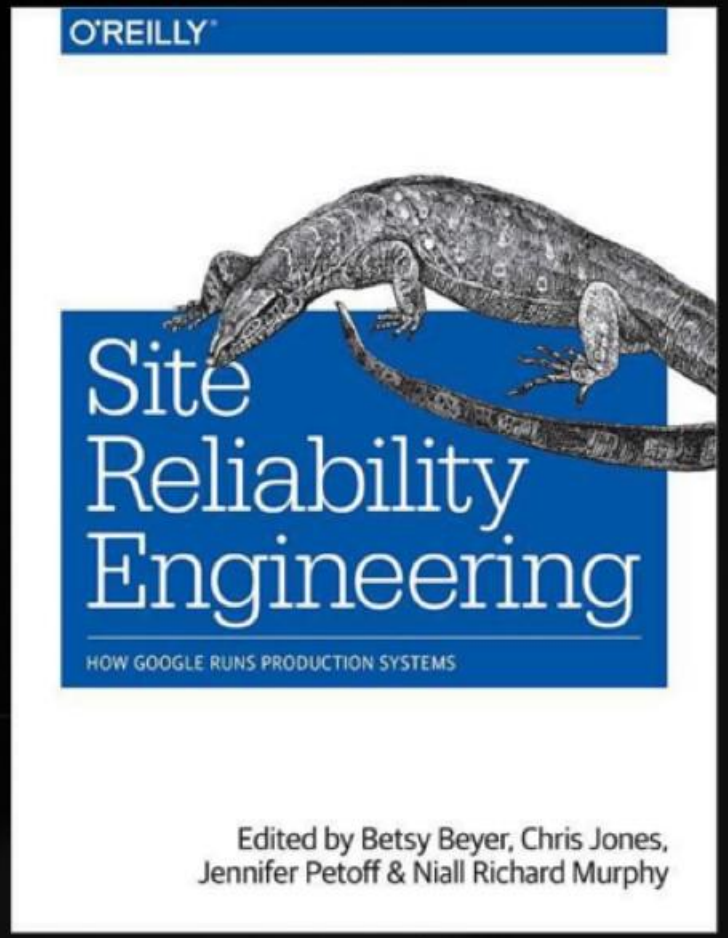
Google Cloud

- 2조 / 매년 - 구글 검색
- 650억/ 매년 - Google PlayStore 앱 다운로드
- 10억 명/ 매월 - 모바일 Chrome 브라우저.
- 2억 명/ 매월 - Google 포토

SRE

- SRE의 시작은 2003년 Google 에서 시스템 운영환경을 개선하기 위한 프로젝트
  - 구글의 대규모 시스템을 안정적이고 효율적으로 운영 및 유지보수 하기 위한 새로운 환경 제공
  - 새로운 서비스와 기능을 지속적으로 유연하게 도입될 수 있도록 변화를 수용하면서 우수한 품질을 end-user 에게 제공하는 것이 프로젝트의 목표
- SRE라는 단어를 처음 사용한 구글의 24/7 운영 부서 VP인 벤 트레이노 슬로스 (Ben Treynor Sloss)는 신뢰성을 모든 제품이 가장 기본적으로 갖추어야 할 기능으로 정의
- SRE ? SysAdmin
  - 구글은 SRE팀에 OPS (티켓, 전화 응대, 수작업 등) 업무에 최대 50%의 시간만 투입
  - 구글의 SRE팀은 반드시 남은 50%의 시간을 오롯이 개발을 위해 활용해야 한다는 것
  - SRE인원수는 서비스 규모에 비례하지 않음 (Google 은 불가능)

시작은 이 책: 사이트 신뢰성 엔지니어링



"지메일 같은 대용량 서비스를 개발하는 일은 결코 쉬운 일이 아니다. 이런 서비스들을 높은 신뢰성으로 운영하는 것은 그보다 더 어려운 일이다. 특히, 이 서비스들이 거의 매일 수정되고 업그레이드되는 상황이라면 더더욱 그렇다. SRE는 구글이 이 어려운 일을 어떻게 해냈는지를 보여주는 일종의 '레시피'다."

- 우르스 회즐(Urs Holzle) (구글의 테크니컬 인프라스트럭처 SVP)

"구글 직원들은 거대한 스케일과 높은 신뢰도를 모두 이룩한 구글의 서비스들을 개발해 오면서 자신들이 시행했던 프로세스나 겪은 실수들을 이 책을 통해 모두 공유하고 있다."

- 릭 패로우(Rik Farrow), USENIX

" 구글의 서비스와 보다 작은 규모의 서비스라 하더라도, 그간 구글이 축적해 온 노하우 없이는 제대로 운영할 수 없는 세상에 살고 있다. 스케일과 신뢰도, 그리고 운영에 관련된 도전에 직면한 사람이라면 SRE 를 참조해야 한다."

- 데이비드 N. 블랭크-애들맨(David N. Blank-Edelman), USENIX 이사회 이사

온라인에서 무료로 책 읽기: https:// landing.google.com/sre/book.html

Google SRE

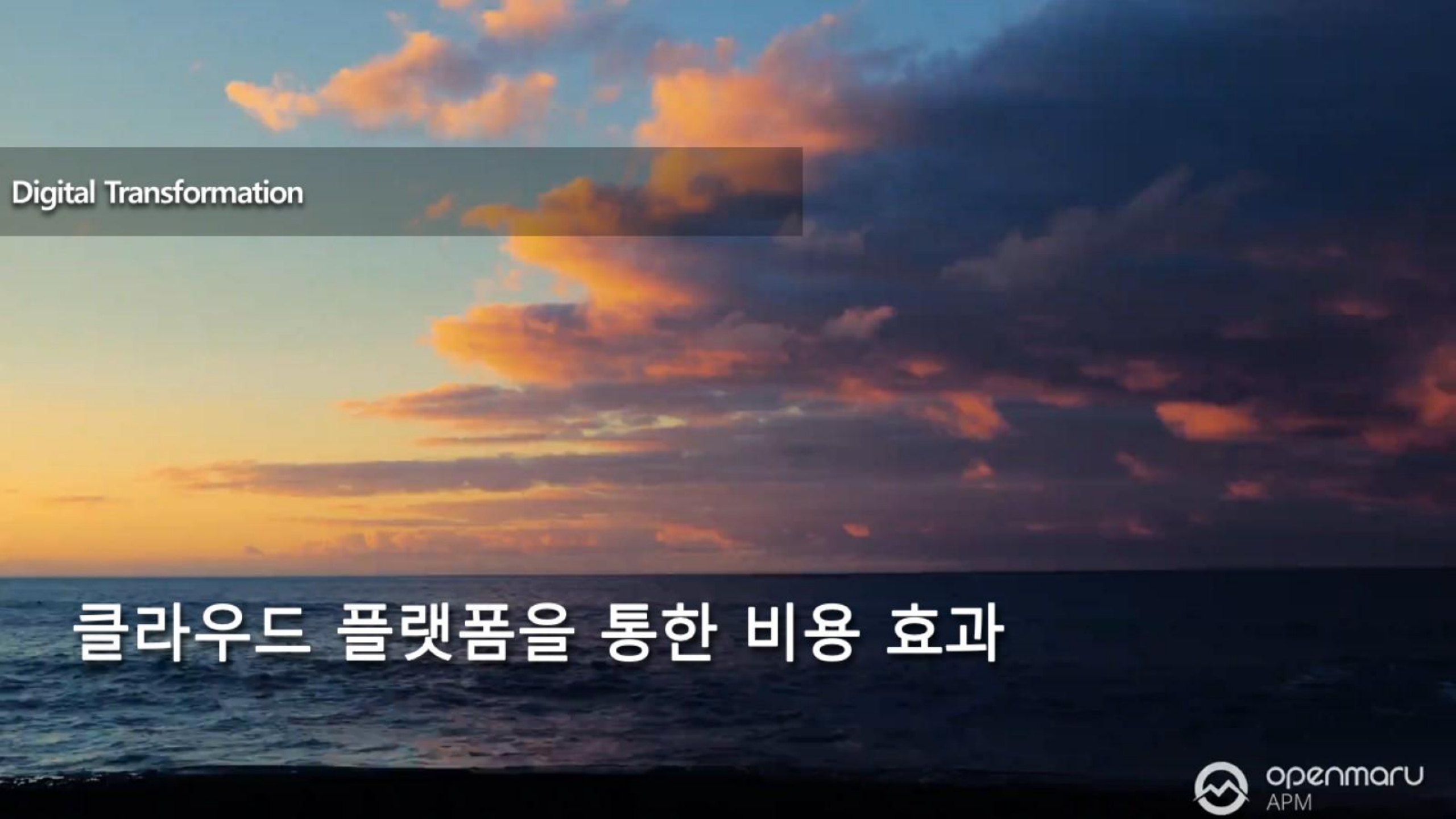
- https://landing.google.com/sre/
  - SRE는 운영상의 문제를 소프트웨어적으로 해결하기 위한 엔지니어링입니다. 우리의 사명은 Google서비스의 가용성, 응답 시간, 성능, 용량을 항상 감시하면서 지키고 발전시키는 것입니다.
- https://landing.google.com/sre/sre-book/chapters/introduction/
  - SRE팀은 서비스 가용성, 대기 시간, 성능, 효율성, 변경 관리, 모니터링, 긴급 대응, 용량 계획에 책임이 있습니다.

Public class SRE implements DevOps {

- DevOps가 철학이라면, SRE는 그 철학을 구현하는 방법
- DevOps 와 SRE 는 경쟁이 아닌 상호 보완적인 관계

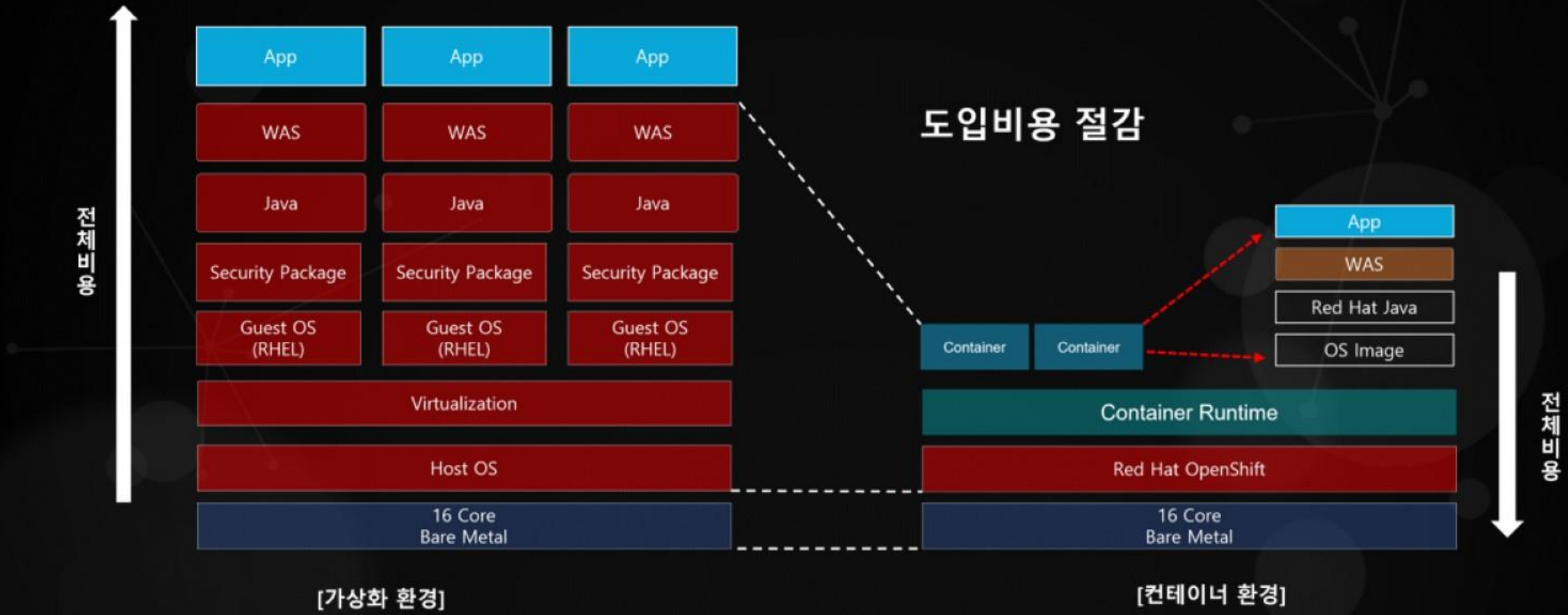
DevOps	SRE
• 조직의 사일로화 제거	• 개발팀과 함께 고민을 공유하고 제품과 기술 스택 전체를 최적화
• 오류는 발생할 수 있는 것으로 가정	• SLO 를 정의 하고 장애에 대한 비난이 아닌 포스트모템 기록
• 단계적으로 변경	• 실패했을 때 코스트를 줄여 빠른 변화 유도
• 도구와 자동화 활용	• 자동화를 통한 장기적인 가치를 제공하는 노력에 집중 • 수작업을 최소화
• 모든 것을 측정	• 운영의 문제를 소프트웨어로 해결하려고 노력 • 사이트 신뢰성 지표를 측정

}


Digital Transformation
클라우드 플랫폼을 통한 비용 효과
openmaru
APM

기존 시스템 구축 비용 대비 클라우드 플랫폼

- 가상화 대비 Guest OS, WAS 유지보수, 라이선스, 관리비용 제거
- 서버 접근제어를 비롯한 보안 솔루션 제거



그럼 Kubernetes만 있으면 되는걸까?

- Kubernetes 훌륭한 기초 기술이지만, 스토리지, 네트워킹, 보안, 애플리케이션 프레임 워크 등을 통합하고 이를 분기별로 갱신하는 것은 큰 부담입니다

- - Ashesh Badani, Red Hat

Why running your own Kubernetes deployment could be a terrible idea

Kubernetes is hard, but becomes doubly so when you take on the burden of supporting this fast-moving project.

By Matt Asay | June 21, 2018, 11:23 AM PST

https://www.techrepublic.com/article/why-running-your-own-kubernetes-deployment-could-be-a-terrible-idea/

KUBERNETES 를 제대로 운영하기 어려운 이유는?

openmaru

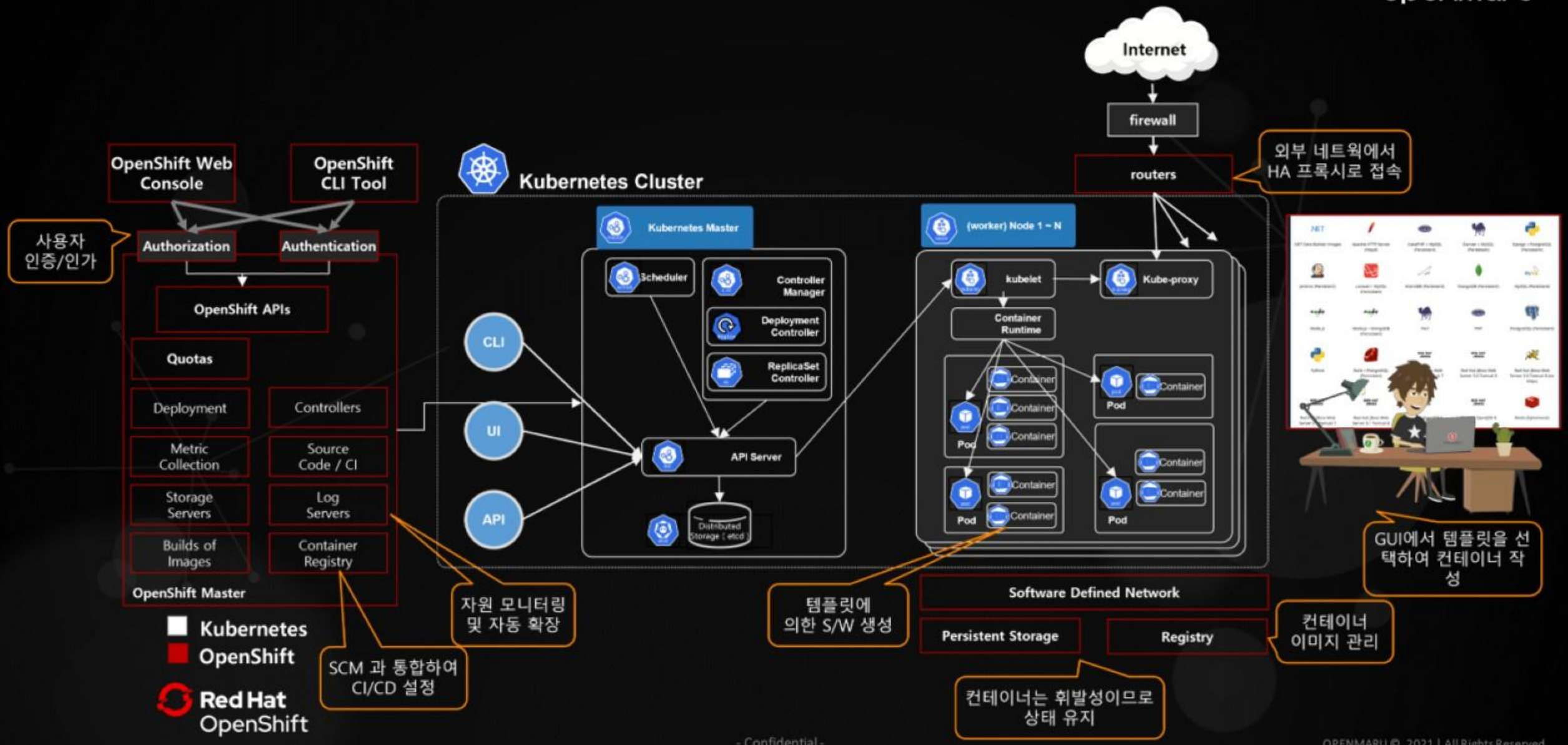
Kubernetes 사용자 중 **75 %** 는

구축과 운영의 복잡성으로 도입하기 어렵다고 함

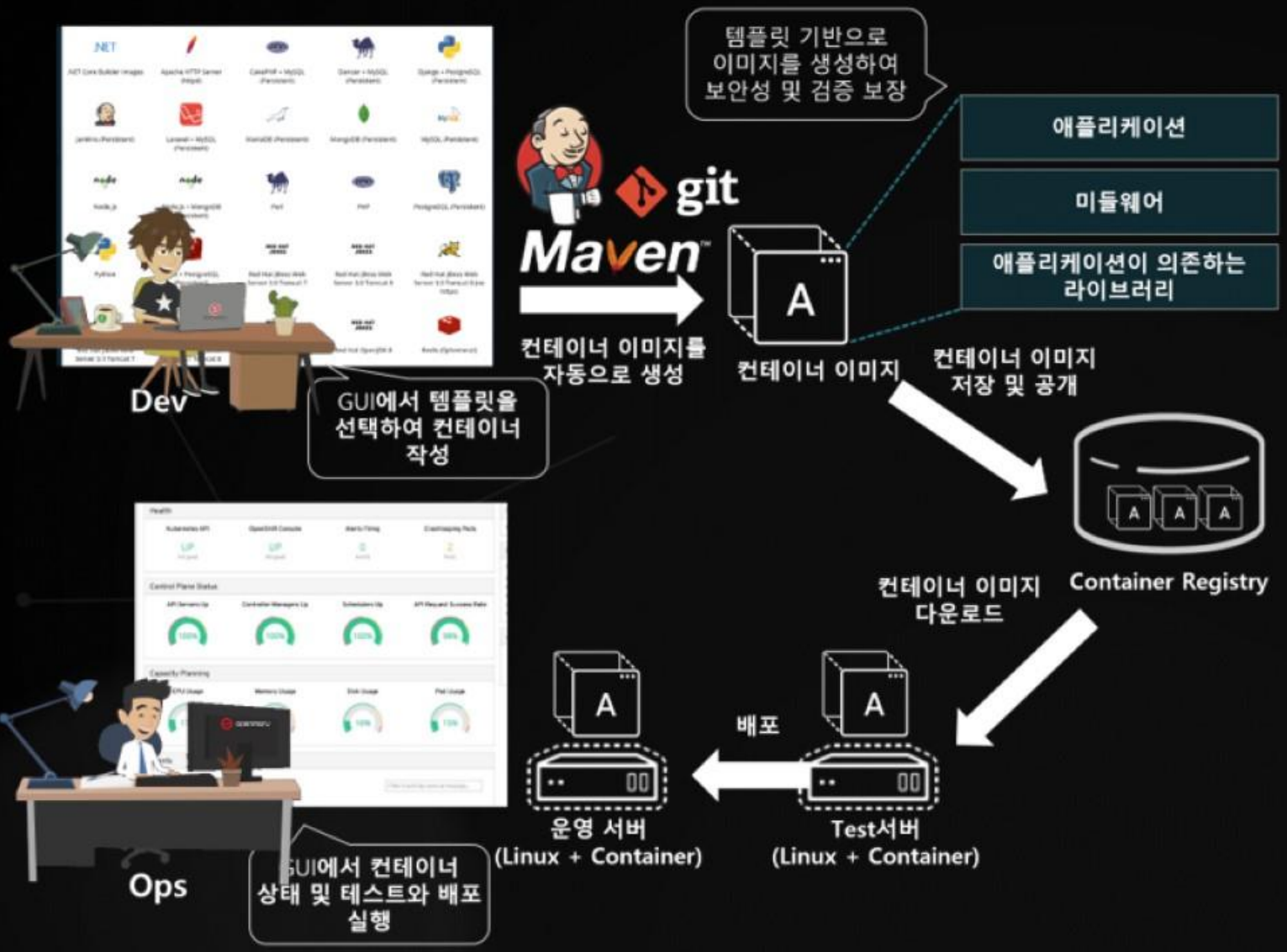
INSTALL	DEPLOY	HARDEN	OPERATE
• Templating	• Identity & Security Access	• Platform Monitoring & Alerts	• OS Upgrade 및 Patch
• Validation	• App Monitoring & Alerts	• Metering & Chargeback	• Platform Upgrade 및 Patch
• OS Setup	• Storage & Persistence	• Platform Security Hardening	• Image Upgrade 및 Patch
	• Egress, Ingress & Integration	• Image Hardening	• App Upgrade 및 Patch
	• Host Container Images	• Security Certifications	• Security Patches
	• Build / Deploy Methodology	• Network Policy	• Continuous Security Scanning
		• Disaster Recovery	• Multi-environment Rollout
		• Resource Segmentation	• Enterprise Container Registry
			• Cluster & App Elasticity
			• Monitor, Alert, Remediate
			• Log Aggregation

Source : The New Stack, The State of the Kubernetes Ecosystem, August 2017

Kubernetes 에 추가된 OpenShift 기능 들



OpenShift 와 Kubernetes 기능 측면 비교



카테고리	Kubernetes	OpenShift
기업용 리눅스	구매	포함
기업용 미들웨어	구매	포함
이미지 레지스트리	별도	포함
스토리지 인터페이스	별도	포함
SDN	별도	포함
통합로그 관리	별도	포함
멀티 테넌시	별도	포함
미터링	별도	포함
CI/CD	별도	포함
인증된 DB,M/W등	없음	포함
무정지 업그레이드	불가	지원
24X7 기술지원	자체	지원
보안	자체	지원

OpenShift 와 Kubernetes 비용 측면 비교

	Tasks		kubernetes Months	Engineer	Total Cost($)	Red Hat OpenShift Hours	Engineer	Total Cost($)
1	Environment Provisioning	Implementation	2	3	$ 99,804	40	2	$ 7,692
2	Security		1	1	$ 16,634	8	2	$ 1,538
3	HA & Healing		1	1	$ 16,634	8	1	$ 769
4	Load Balancing		1	1	$ 16,634	4	1	$ 385
5	Monitoring		1	1	$ 16,634	4	1	$ 385
6	Upgrading		2	1	$ 33,268	8	1	$ 769
7	Environment Support		2	1	$ 33,268	8	1	$ 769
8	Training				$ 150,000			$ 100,000
9	OpenShift Subscription	Subscription			N/A			$ 200,000
			1년 합계		$ 382,875	1년 합계		$ 312,307
1	Security Updateds	Support & Maintenance	2	1	$ 33,268	106	1	$ 10,192
2	Load Balancing		2	1	$ 33,268	106	1	$ 10,192
3	Monitoring		6	1	$ 99,804	212	1	$ 20,384
4	Software Upgrading		6	1	$ 99,804	212	1	$ 20,384
5	Troubleshooting Support		6	1	$ 99,804	212	1	$ 20,384
6	Lost productivity cost due to outages				$ 100,000			$ 100,000
7	Openshift Subscription	Subscription			N/A			$ 200,000
			2년 합계		$ 465,947	2년 합계		$ 381,535

[엔지니어 인건비]

Kubernetes Cost Table		OpenShift Cost Table	
Labor Cost & Assumption ($) 1 Year		Labor Cost & Assumption ($) 1 Year	
Engineer	$ 200,000	Engineer	$ 200,000
Hourly Rate (2200 hours)	$ 96.15	Hourly Rate (2200 hours)	$ 96.15
Monthly Working Hours	173	Monthly Working Hours	173
Monthly Rate	$ 16,634	Monthly Rate	$ 16,634
Annual Growth	30%	Annual Growth	30%

* RED HAT 본사 Engineer팀에서 직접 산출한 자료 기준

[OpenShift 적용 시 Kubernetes 적용 대비 절감 비율]

1년 기준	-18%
2년 기준	-18%

OpenShift와 Kubernetes의 컴포넌트별 비교

openmaru

분류	이유	비고
이미지 레지스트리	• 별도 솔루션 필요- 담당 엔지니어가 **직접** 설치/구성/업데이트 진행	• OpenShift 포함
로그관리	• 별도 솔루션 필요- 담당 엔지니어가 **직접** 설치/구성/업데이트 진행	• OpenShift 포함
SDN (Software-Defined Network)	• 별도 솔루션 필요- 담당 엔지니어가 **직접** 설치/구성/업데이트 진행	• OpenShift 포함
운영체제	• Linux 라이선스 **별도 구매**	• 번들로 포함하여 무료 (CoreOS)
컨테이너 런타임	• Containerd/CRI-O **직접** 선택필요	• 표준 컨테이너인 CRI-O
보안	• 별도 솔루션 필요- 담당 엔지니어가 **직접** 보안 책임	• OS 레벨에서 보안/취약점 대응 • 안전한 컨테이너 사용을 위한 이미지 스캐닝, 암호화, 실행 사용자 제한 등 각종 기능 제공
운영관리	• 별도 솔루션 필요-담당 엔지니어가 **직접** 운영관리	• 앤서블을 사용한 설치 및 설정 • 레드햇이 제공하는 클라우드 시스템 관리 제품과의 연계
CI / CD	• 별도 솔루션 필요-담당 엔지니어가 **직접** 설치/구성	• OpenShift에 포함된 Jenkins로 원활한 CI/CD 통합 가능

- **OpenShift** 는 **레드햇과 공식파트너 오픈나루가 책임!**
- **Kubernetes**는 제품 **직접** 선택/ **직접** 설치/ **직접** 구성/ **직접** 업데이트 / **직접 책임**

Kubernetes (K8S) 도입 당시 상황, 도입 후 문제점, 효과 (E-Commerce Case)

openmaru

- 개요
  - 개발계만 소수 인원으로 K8S 사용 중이었으나 운영상의 문제로 운영계까지 K8S 도입은 어렵다고 판단
  - 운영계는 안정적인 OpenShift 도입
  - 추후 기존 개발계까지 OpenShift 로 전환완료
- K8S 도입 후 문제점
  - 개발환경을 K8S로 Logging, Monitoring, Dashboard 기능을 직접 구성, 설치, 관리 어려움
  - Prometheus, Grafana, EFK 직접 업데이트 어려움 및 버전관리 어려움
  - K8S의 컨테이너 이미지에 대한 보안 취약 문제점 발생
- Why Red Hat OpenShift?
  - 위 문제점 Red Hat OpenShift로 해결하기 위해 도입
- OpenShift 도입 효과
  - 인력 및 추가 유지보수 낮춤으로서 총 운영 비용 절감
  - OpenShift 컨테이너 기반의 CI/CD 시스템 운영관리체계 표준화 실현
  - 검증된 보안으로 안정적인 운영 가능
- OpenShift 설치 부터 운영까지 기간
  - 설치에서 안정적인 설정까지 총 3개월 소요

OpenShift의 특징

openmaru

- Kubernetes 환경을
- 편하게 **구축** 하는
- 편하게 **유지 보수** 하는
- 편하게 **개발** 하는
- 편하게 **확장** 하는

Red Hat OpenShift


openmaru