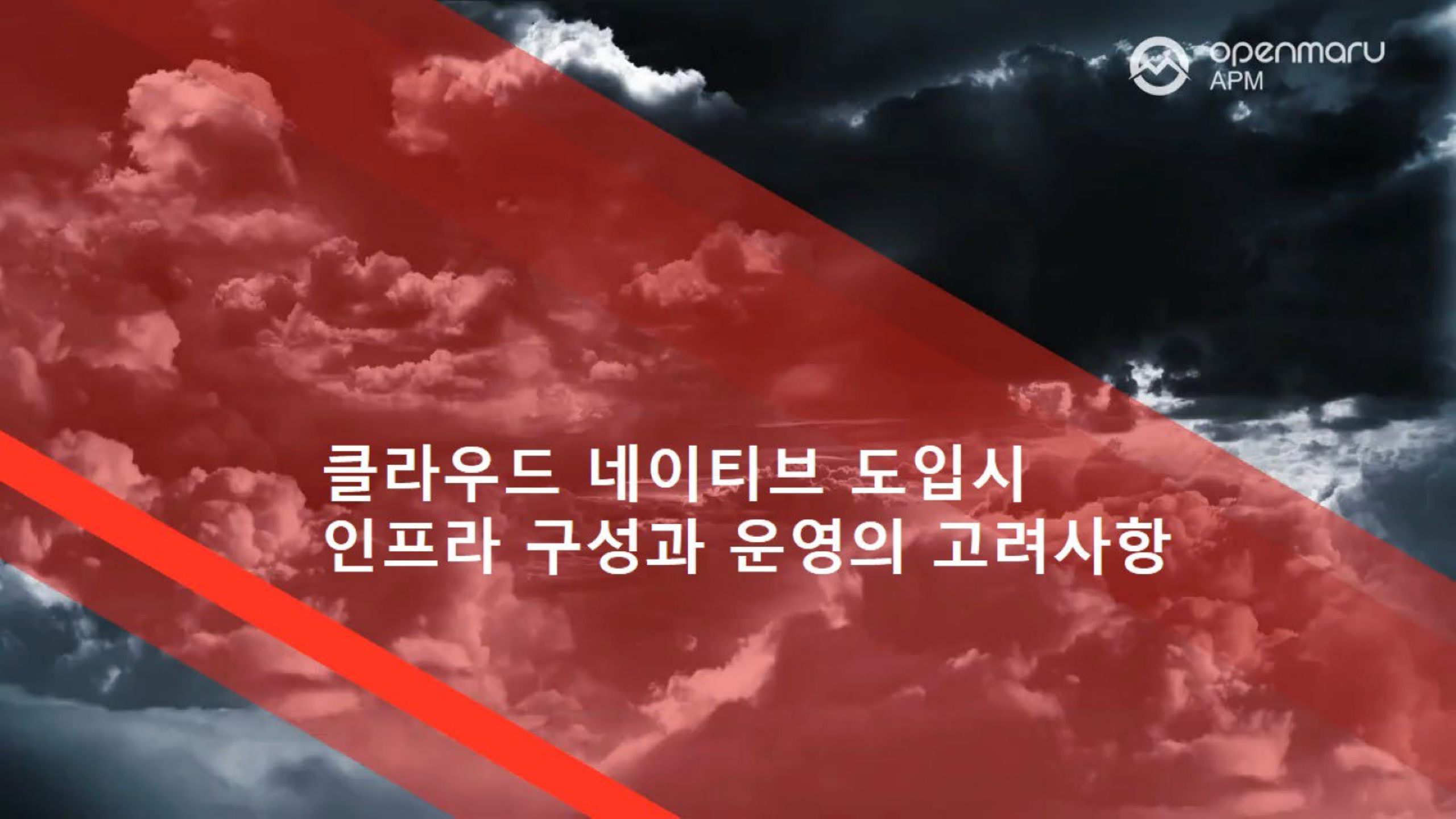
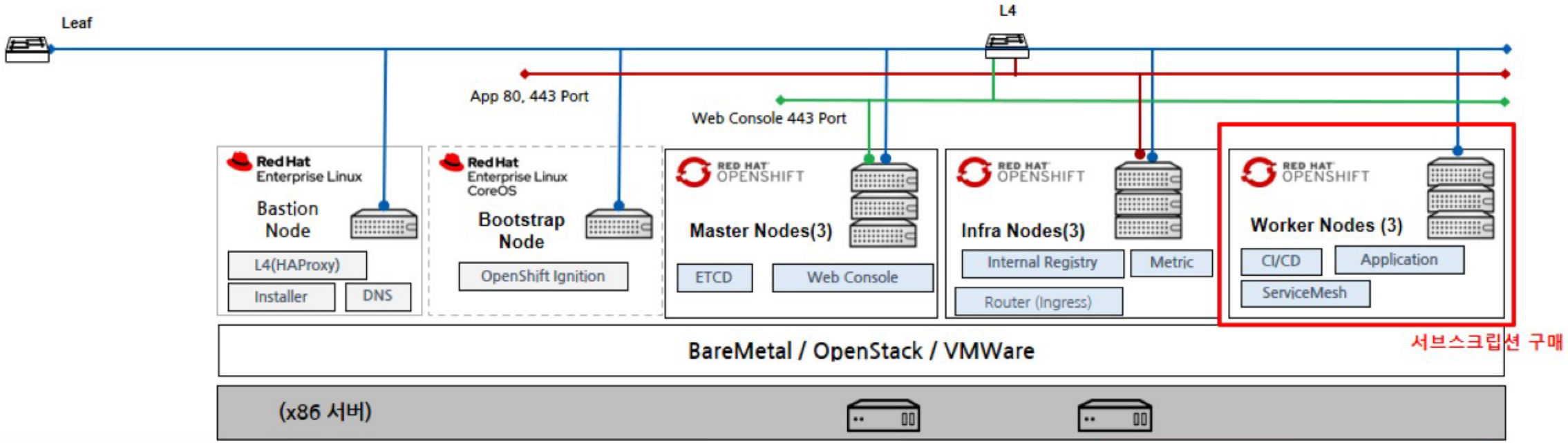

openmaru
APM
클라우드 네이티브 도입시
인프라 구성과 운영의 고려사항

OpenShift 인프라 구성시 가장 많이 질문하는 내용들

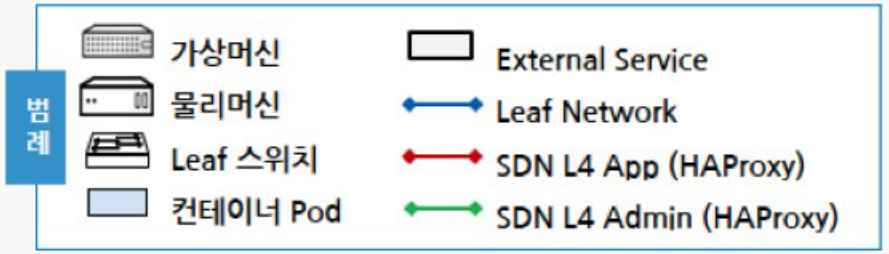
- 왜 서버가 이렇게 많이 필요한가요?
- 네트워크는 어떻게 연결해야 하나요?
- DMZ 구간이 필요한데, 어떻게 구성해야 하나요?
- 스토리지는 왜 필요한가요?
- 보안성 심의는 어떻게 처리해야 하나요?
- OpenShift 업그레이드는 어떻게 하나요?
- 백업은 어떻게 하나요?
- DR은 어떻게 구성해야 하나요?
- 빌드 배포는 꼭 GIT이어야 한다고 하는데, GIT 서버도 제공하나요?

OpenShift 구성방안

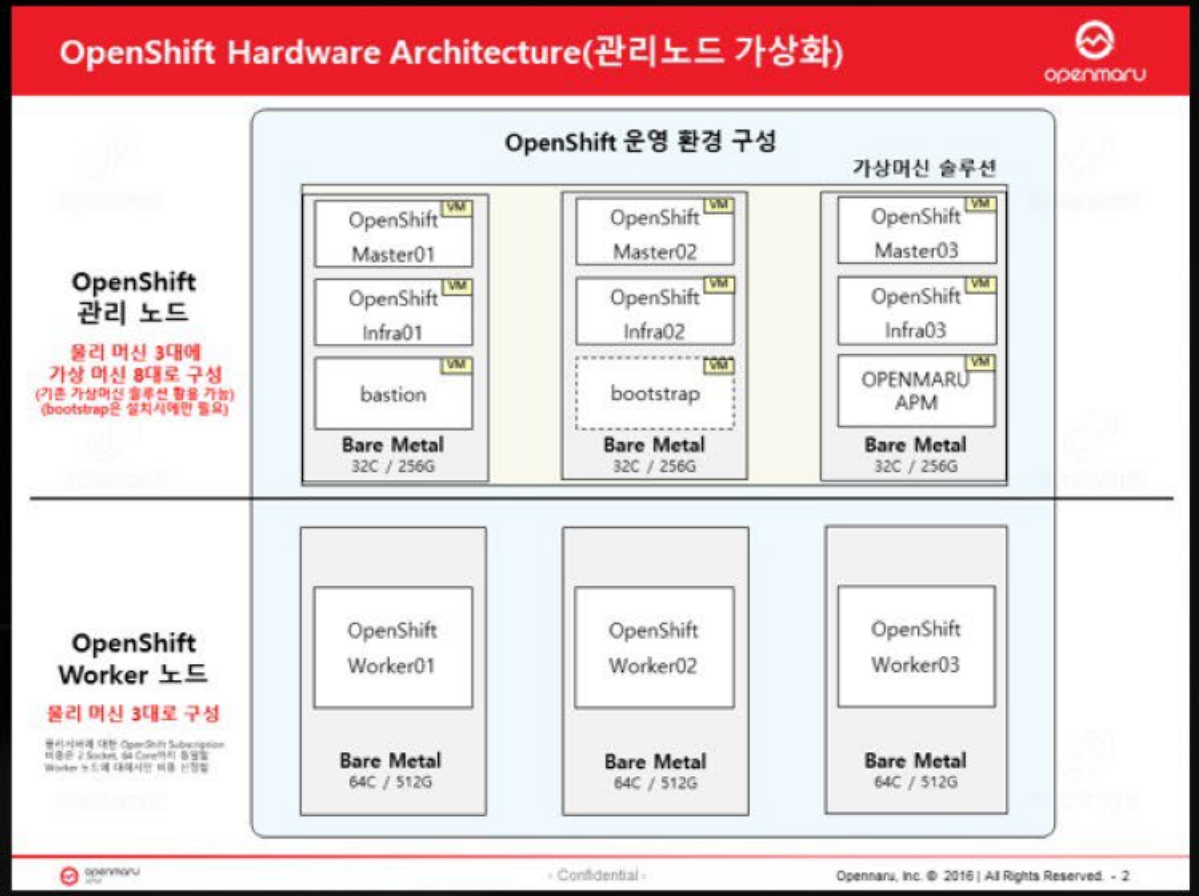
왜 이렇게 서버가 많이 필요한가요? > OpenShift 구성도

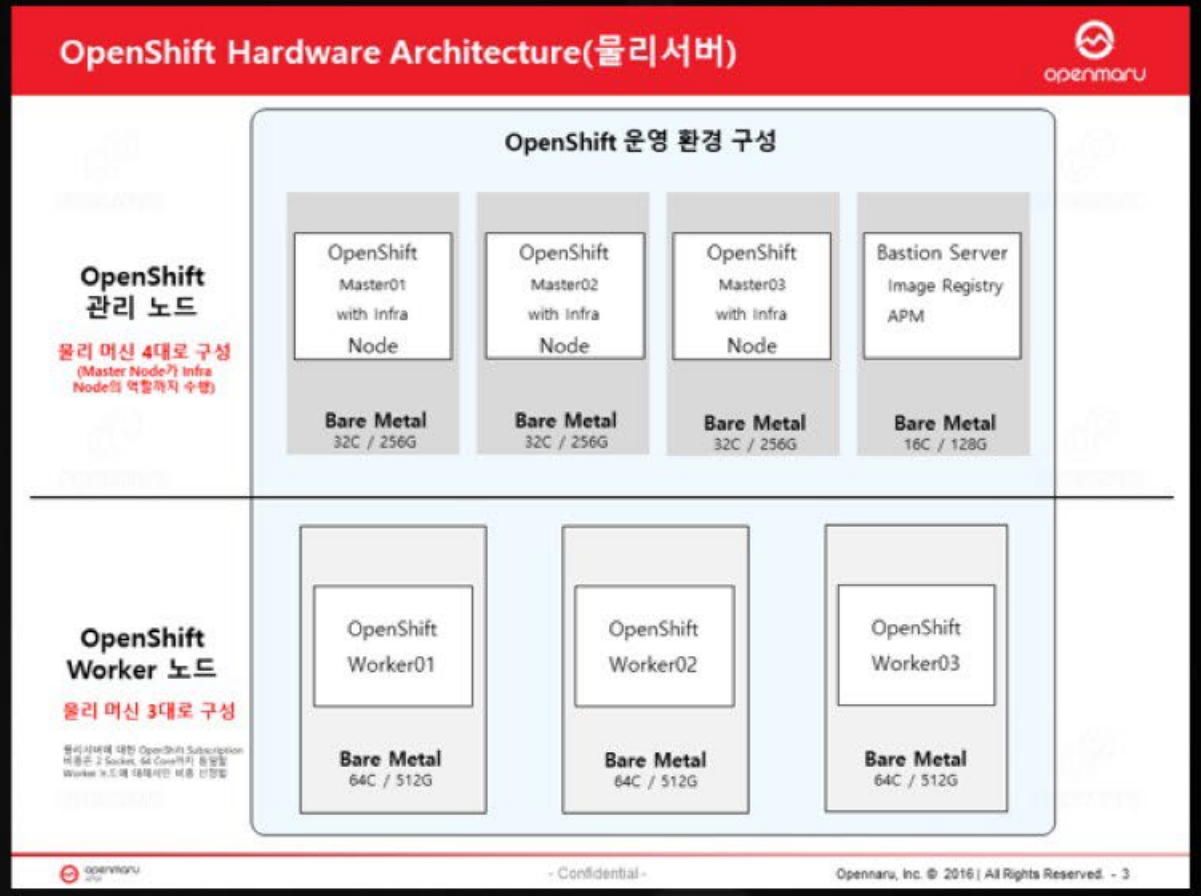


- Bastion Node : DNS 서비스 와 External Registry 를 함께 구성
- Bootstrap Node : 설치 시 필요한 임시 구성
- Master Nodes : ETCD 와 Web Console 를 함께 구성
- Infra Nodes : Internal Registry, Metrics, Service를 함께 구성, Router(Ingress) Pod 구성
- Worker(App) Nodes : Application 서비스용 컨테이너 구성
- Master Nodes 와 Router Nodes 는 외부 서비스 포트 오픈 필요
- 모든 Nodes는 물리(Bare Metal) 및 가상화(VM) 상에 모두 구성 가능



왜 이렇게 서버가 많이 필요한가요? > OpenShift 가상/물리 최소 구성도

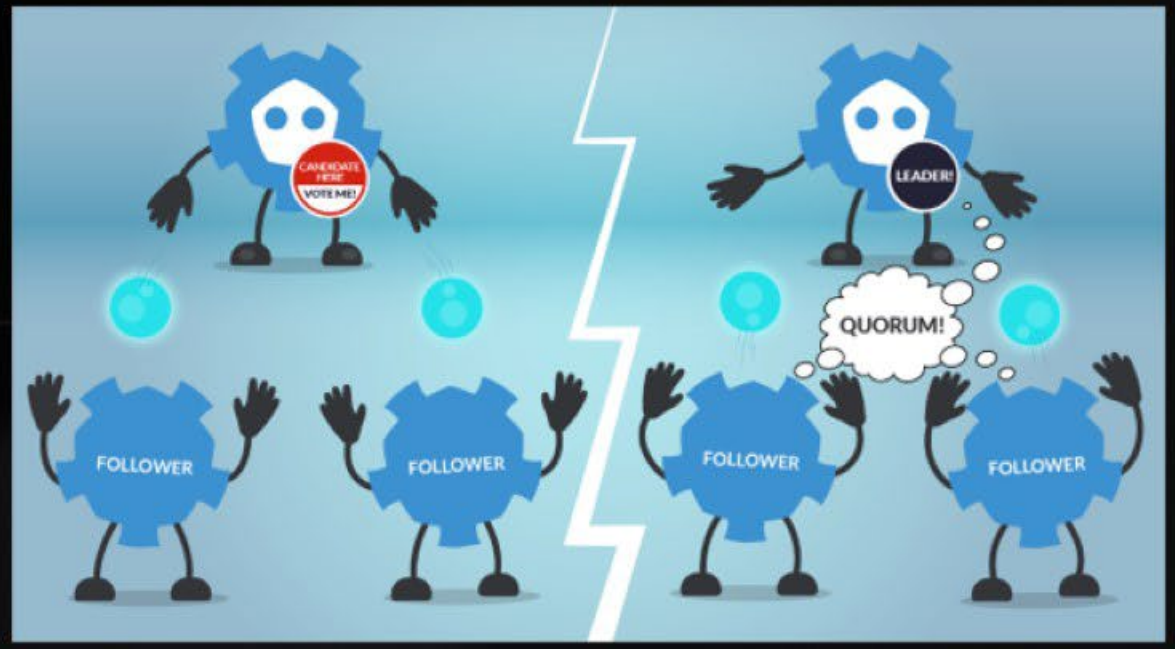




왜 서버가 이렇게 많이 필요한가요? > 왜 3대씩이나 필요한가요?

- Etcd는 OpenShift(Kubernetes)의 모든 데이터를 저장하는 Key / Value 데이터베이스(가장 중요함)
- Etcd는 서버가 살아있다. 이 데이터를 기록하면 오염될까? 중요한 의사 결정에 Quorum(정족수) 방식을 사용함.
- 투표를 하여 과반수 이상 득표하면 Leader로 선출되어 의사결정을 하는데, 50:50이 나오면 판단할 수 없음
  → 1개, 3개, 5개 … 홀수개가 필요하지만, 1개는 이중화되지 않기에 최소 3개가 필요함

투표를 통해 Leader를 선출



서버가 하나 다운되어도 Leader 선출가능

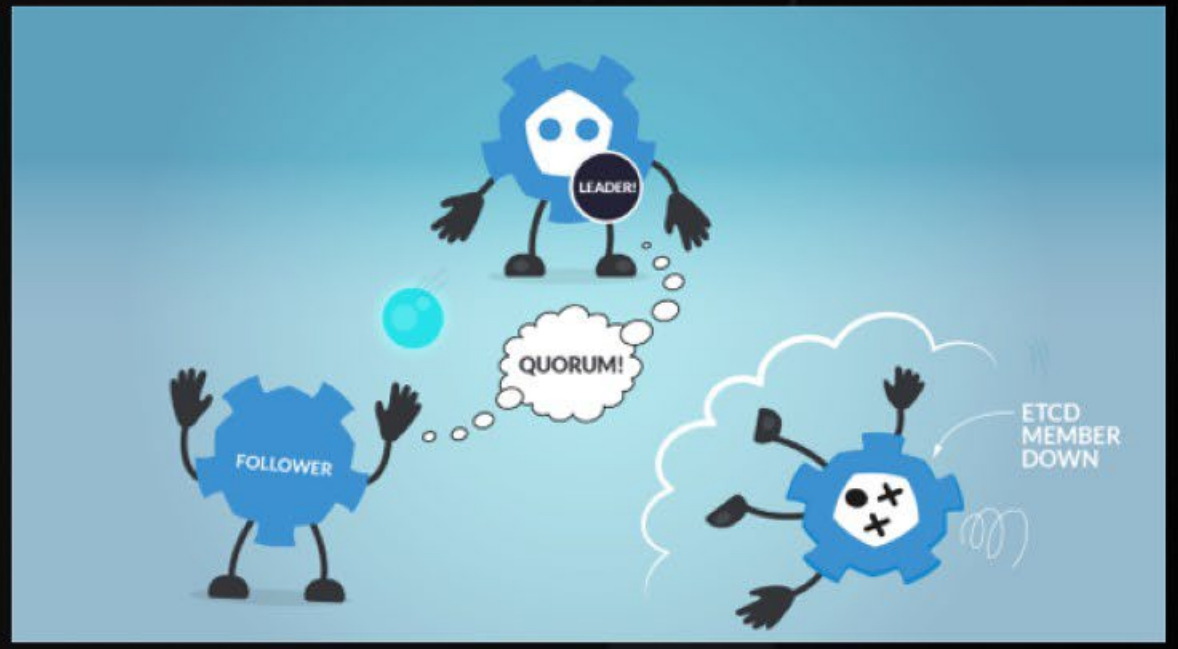
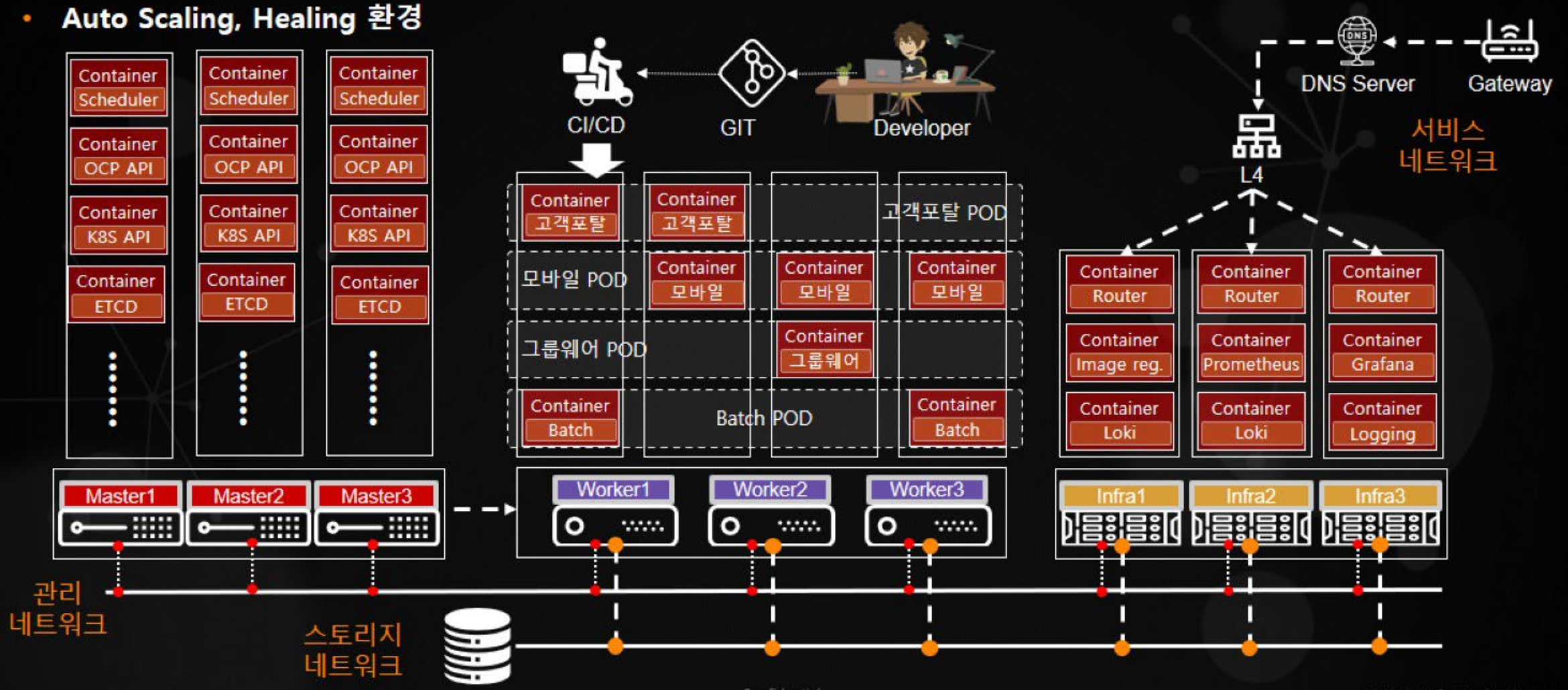


그림 출처 : https://sysdig.com/blog/monitor-etcd/

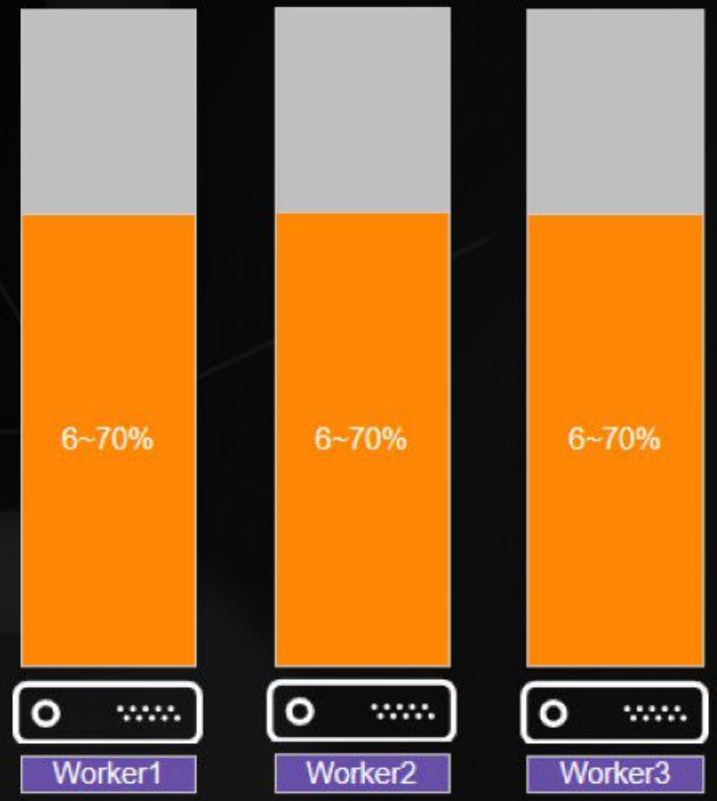
왜 서버가 이렇게 많이 필요한가요? > 클라우드 네이티브 기본 인프라 구성

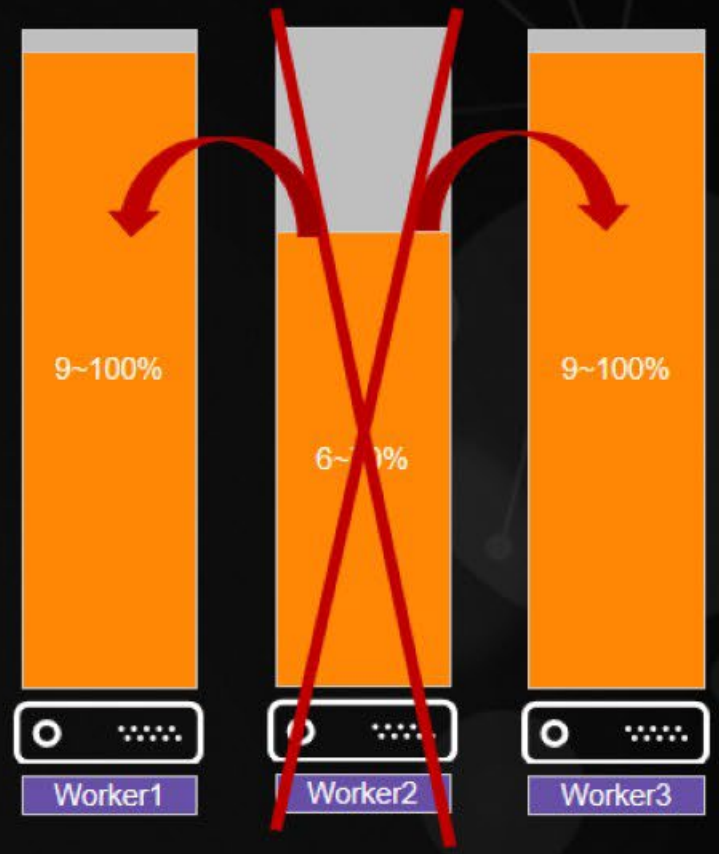
- 서버별로 업무가 나뉘는 것이 아니고, 스케쥴러에 의해 **적재 적소 서버에 애플리케이션 배치**
- **Auto Scaling, Healing 환경**



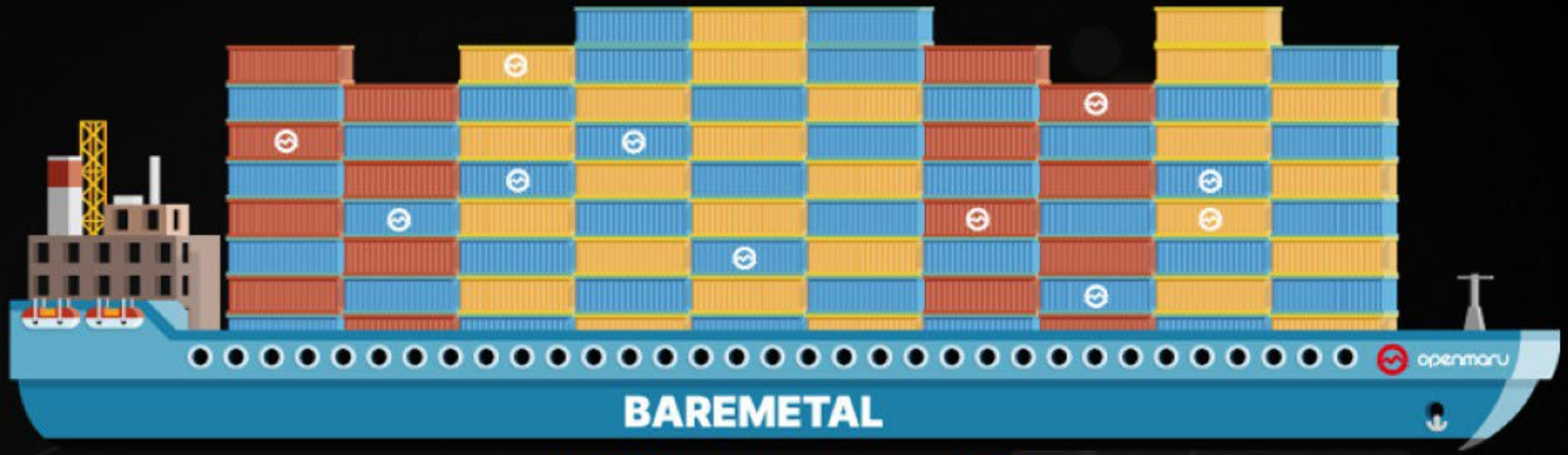
왜 서버가 이렇게 많이 필요한가요? > 왜 워커노드는 3대인가요?

- 3대 머신이 6~70%자원을 사용한다고 가정했을 때, 1대 머신 장애시 부하를 나머지 2대가 처리할 수 있음
- 2대 머신이라면, 1대 머신 장애시 나머지 1대 머신이 기존 부하를 처리할 수 없는 상황이 될 수 있음





왜 서버가 이렇게 많이 필요한가요? > 왜 워커노드는 물리서버여야 하나요?



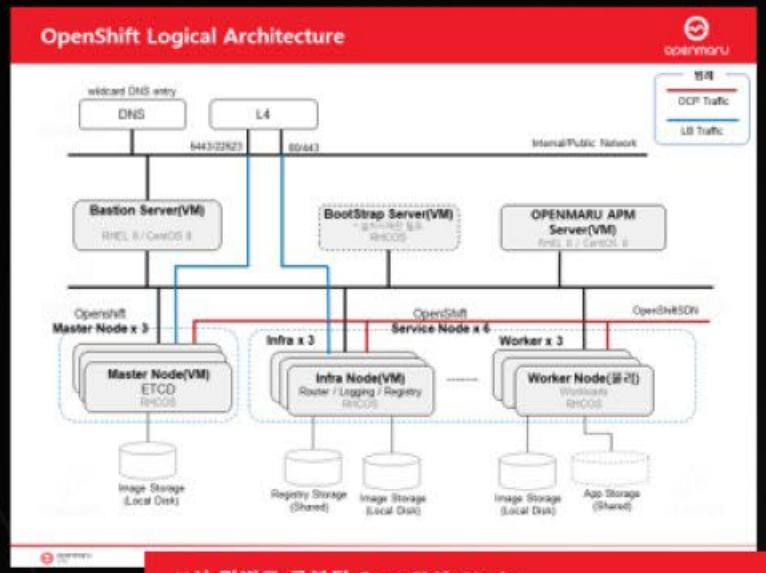
배위에 규격화된 컨테이너를 바로 올려요

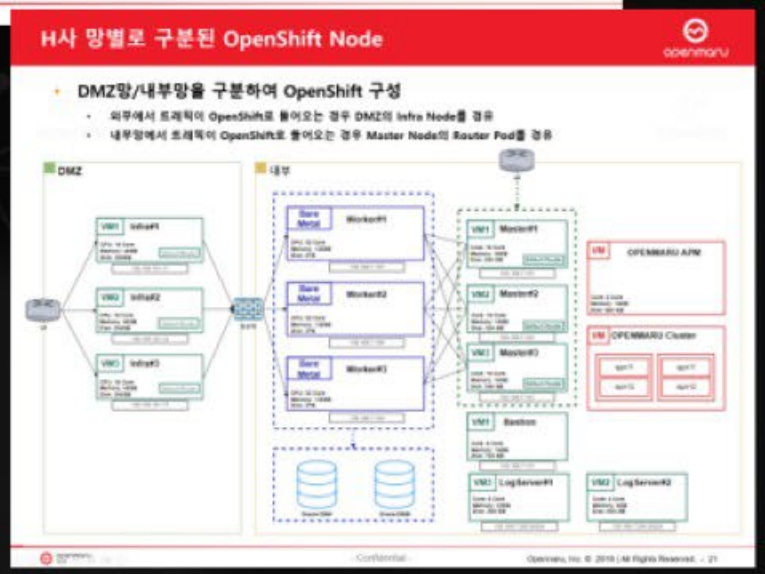
배위에 또 배를 싣고 그 위에 짐을 올려요

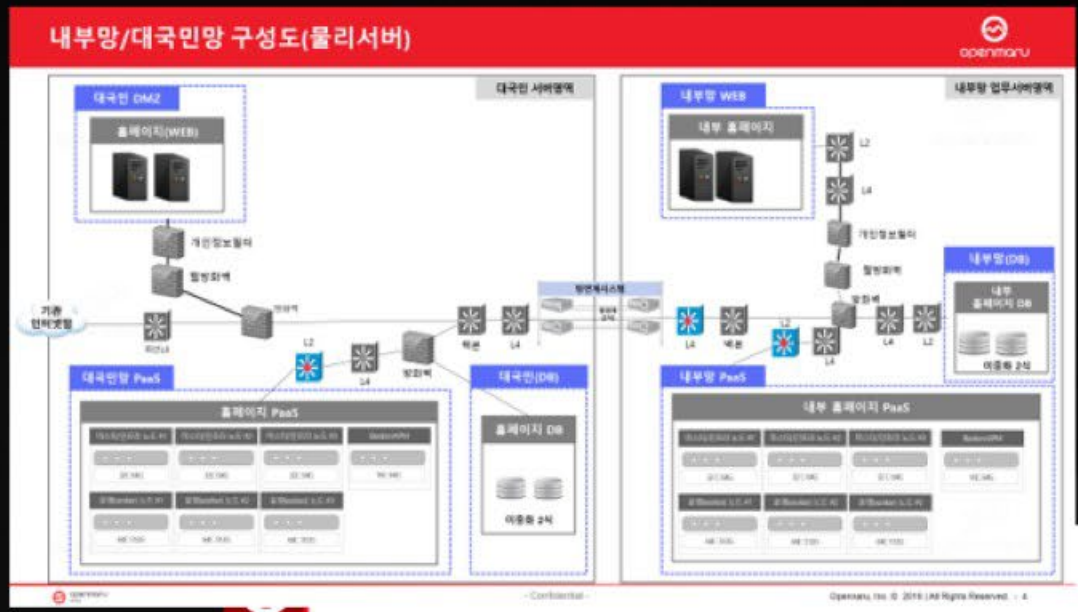
Platform As A Service

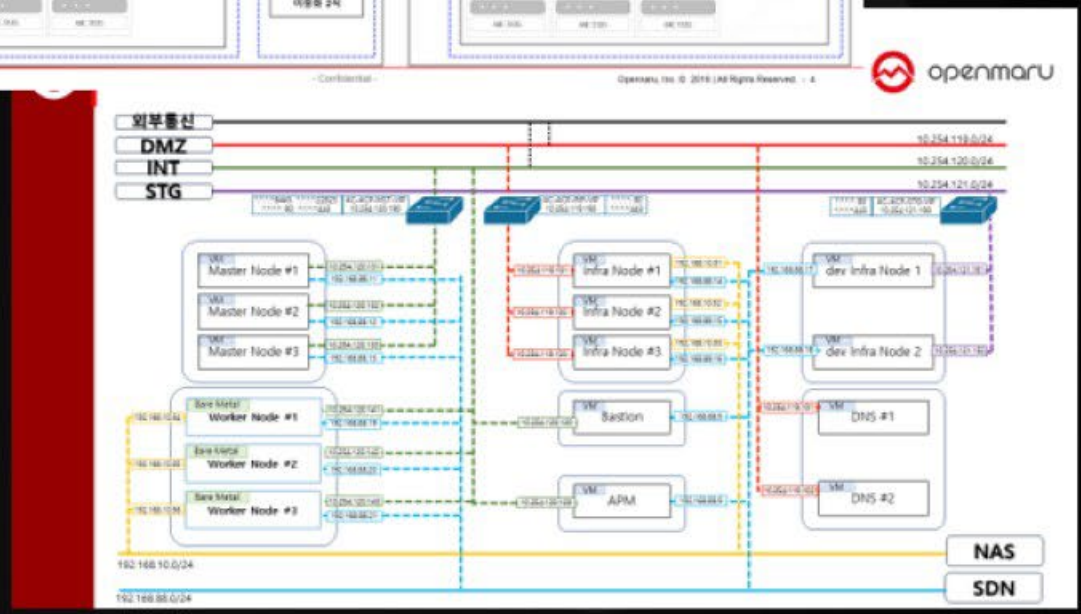
OpenShift Network 구성

네트워크는 어떻게 연결하는가? > 구성도



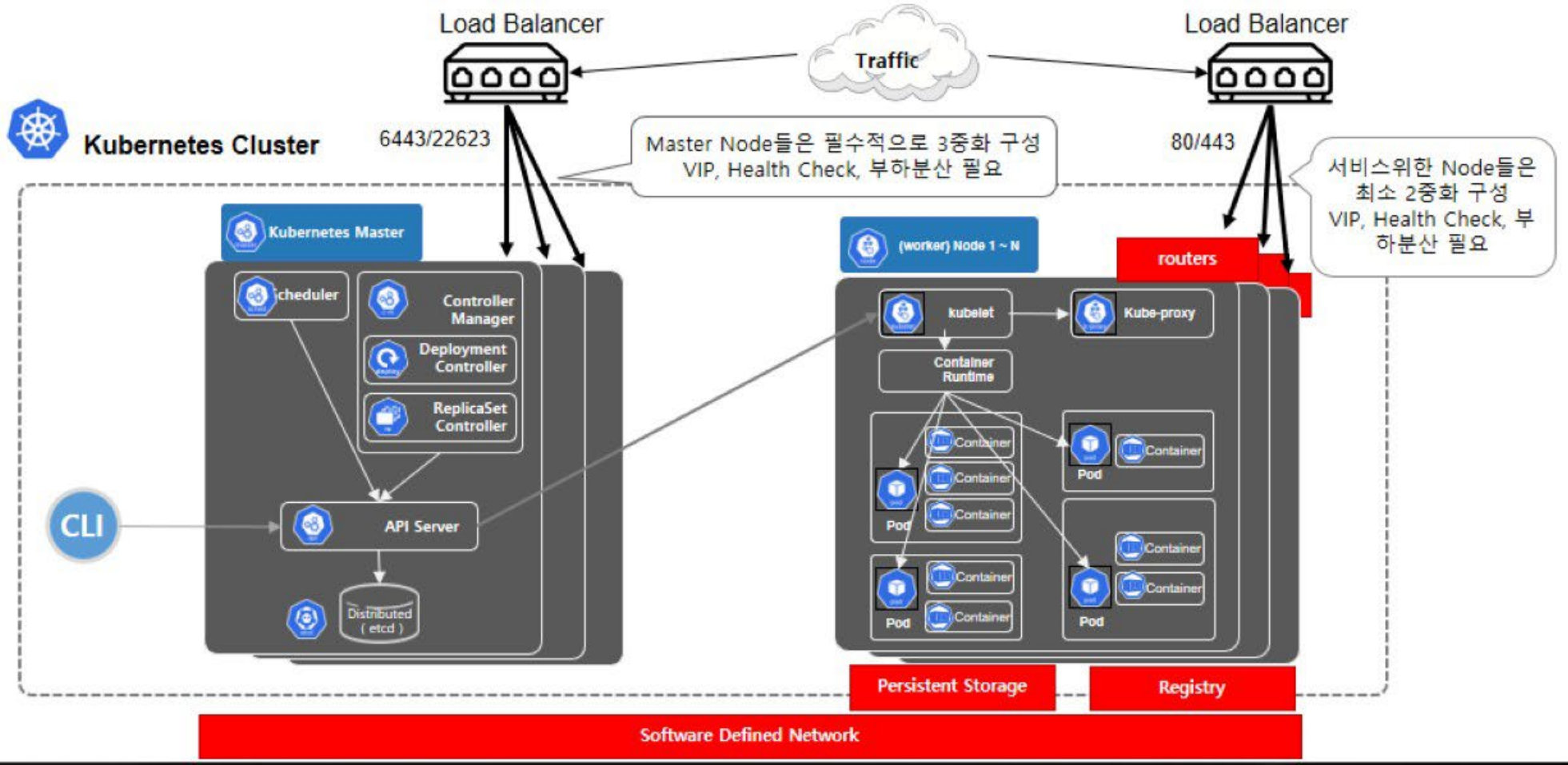






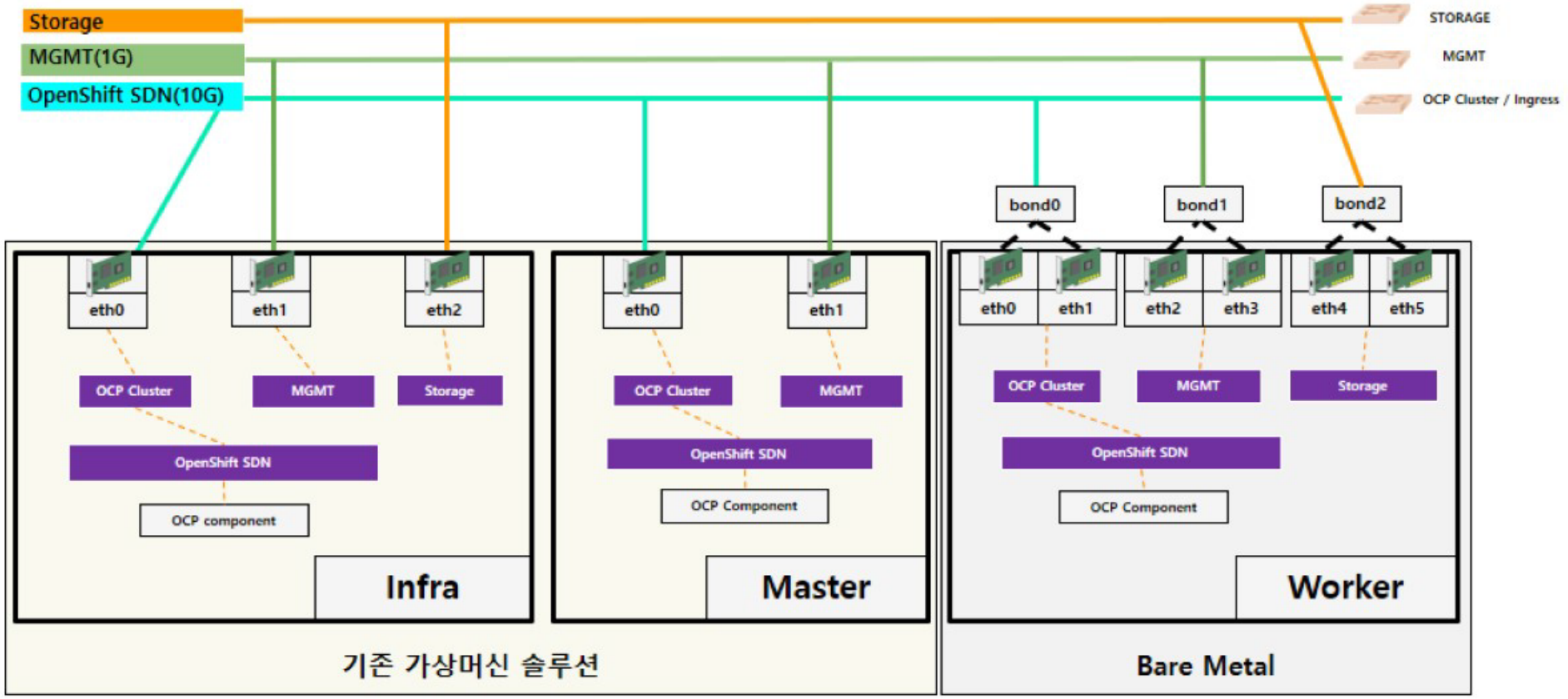
네트워크는 어떻게 연결하는가? > L4는 꼭 필요한가?

- OpenShift의 Node들은 고가용성을 위한 다중화 구성
- OpenShift의 모든 Node는 Active/Active 상태임
- 물리 L4에서, Master, Infra에 대한 Load Balancing / Health check 필요



네트워크는 어떻게 연결하는가? > NIC는 몇 개나 필요한가?

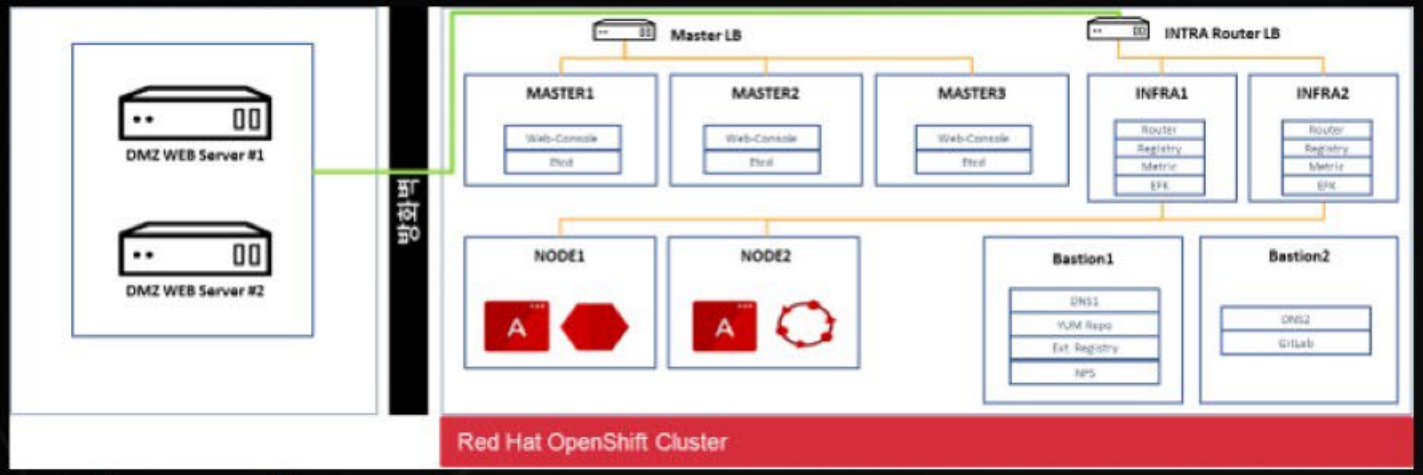
- OpenShift 기본 네트워크는 10G 네트워크가 필수
- 기타 연결되는 네트워크에 대한 NIC 필요
- 스토리지에 대한 연결이 필요하다면 스토리지 네트워크 필요
- Master/Infra가 물리서버라면 NIC가 각각 2개씩 필요



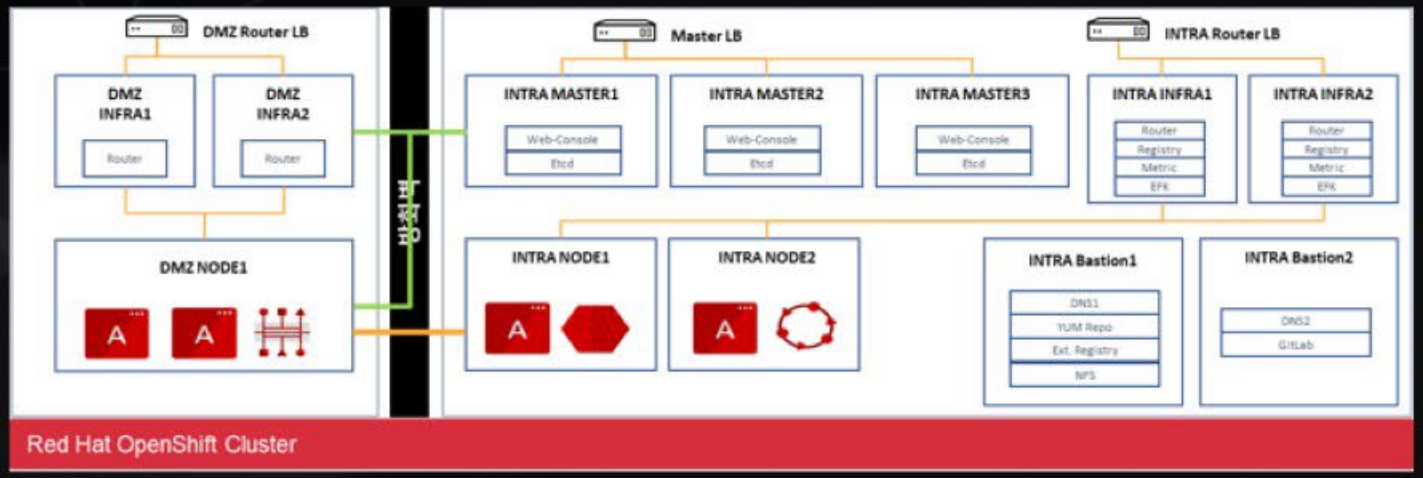
네트워크는 어떻게 연결하는가? > DMZ가 필요한데 어떻게 구성하나?

openmaru

1. WEB Proxy (HTTPD, HAProxy 등) 방식
DMZ의 Proxy Web Server 를 통해 OpenShift Router와 통신하는 방식

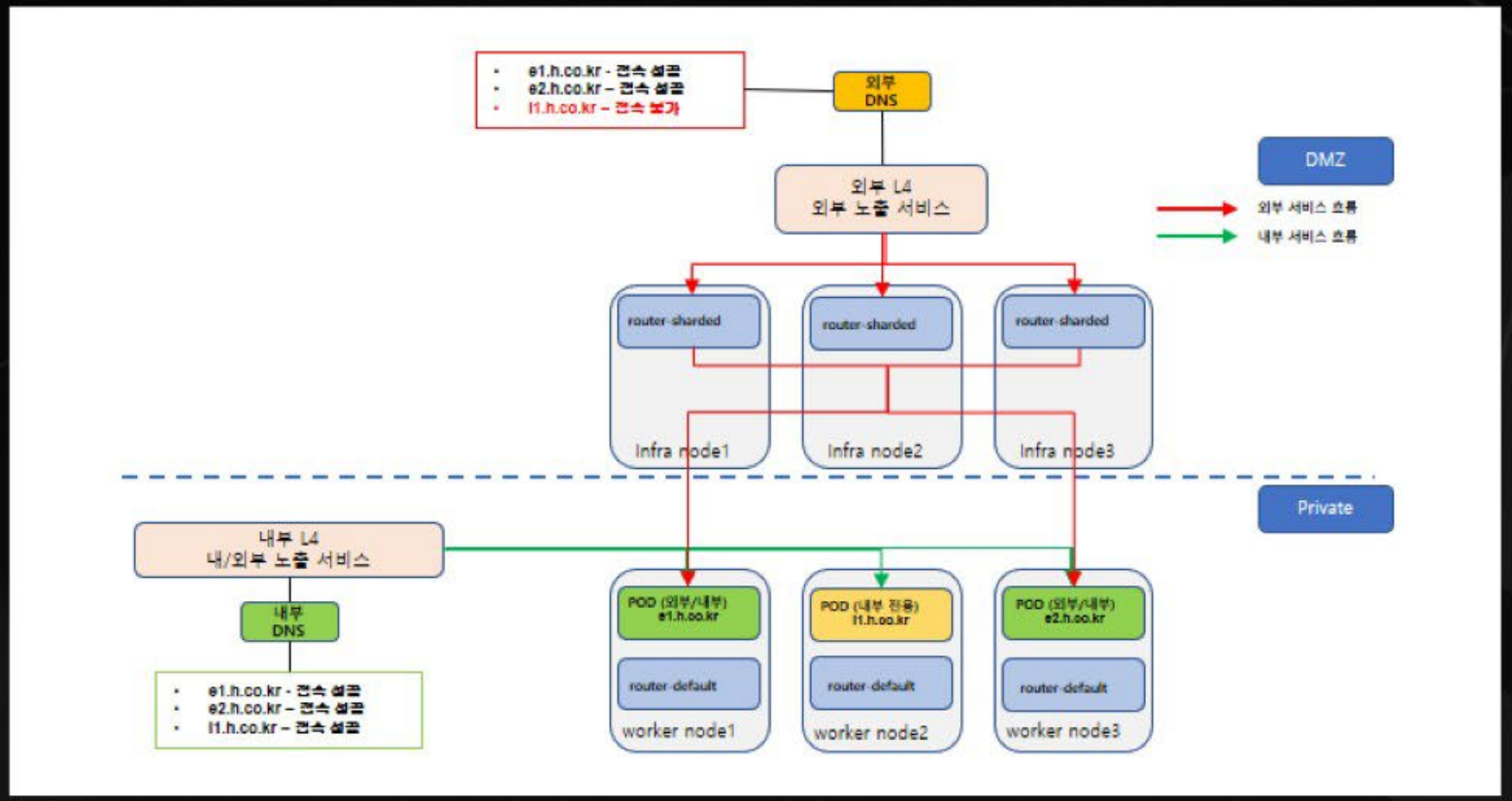


2. OpenShift Router 방식
OpenShift Router를 DMZ에 위치시켜서 OpenShift INTRA Router와 통신하는 방식



네트워크는 어떻게 연결하는가? > 내부/외부 DMZ 라우터 분리

- sharded이란 예를 들어서 지정된 경로만 제공하는 라우터 내부 또는 외부 수신 컨트롤러용
- router-default를 sharded용으로 생성하여 지정된 도메인만 제공 할 수 있는 기능
- 지정된 도메인은 외부서비스이며 내부 사용자들은 router-default를 사용하여 내/외부 서비스 모두 사용 가능

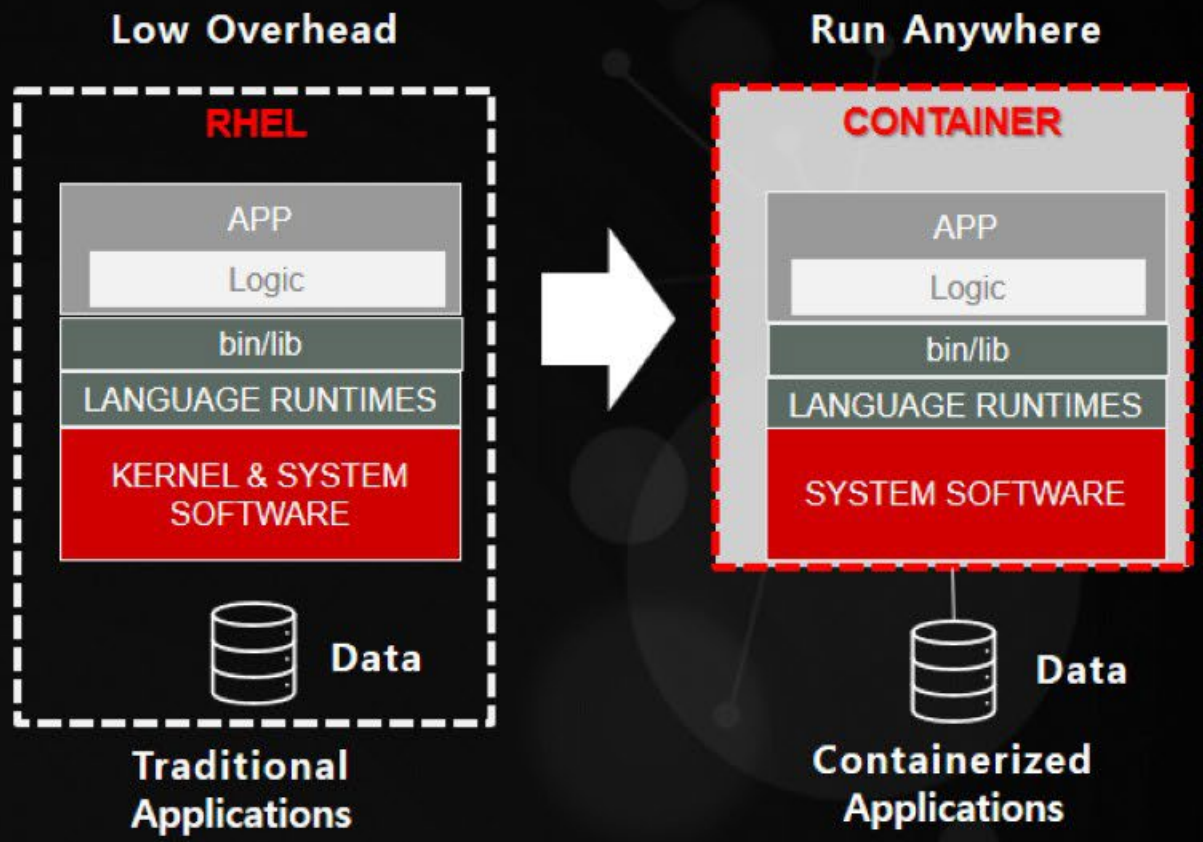


Platform As A Service

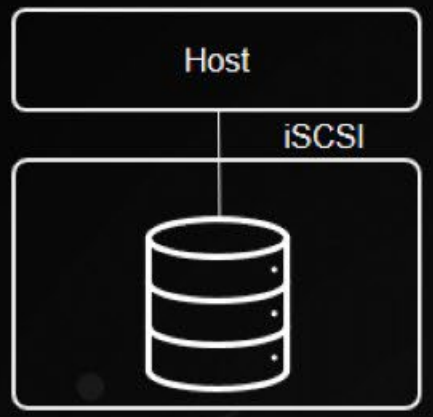
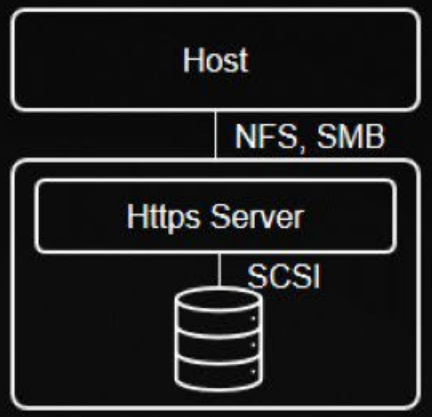
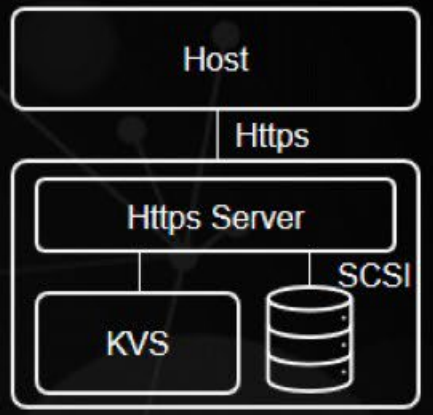
OpenShift 스토리지 연결

OpenShift 스토리지 연결 > Persistent 란?

- Ephemeral Storage (휘발성)
  - 포드 삭제/재배포와 동시에 데이터가 사라짐
  - emptyDir Volume, log directory, writable layers
- Persistent Storage (지속성, 영구적)
  - 포드가 삭제/재배포 되어도 데이터가 사라지지 않음
  - Persistent Volume(PV)

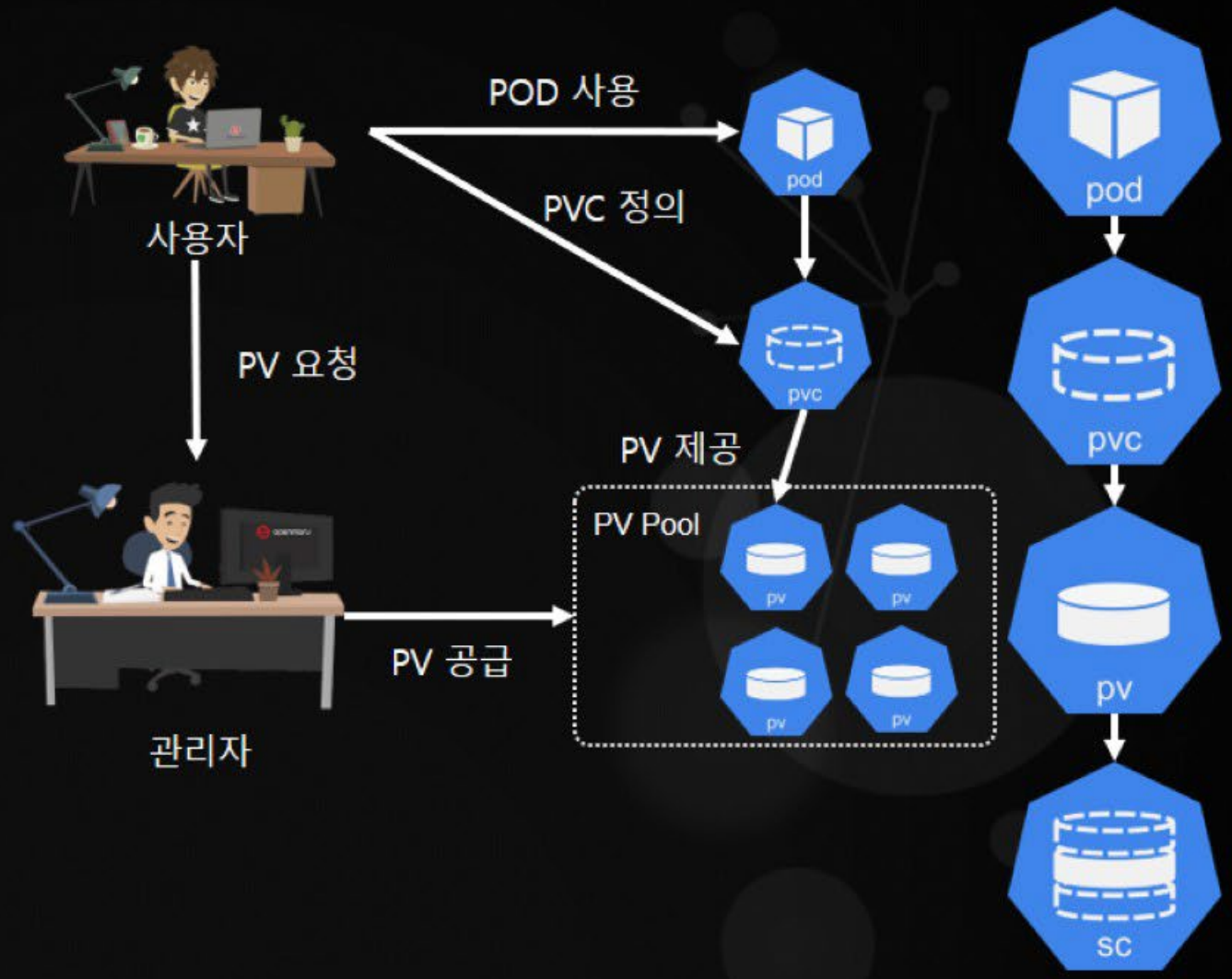


OpenShift 스토리지 연결 > 스토리지 타입

분류	블록 스토리지	파일 스토리지	오브젝트 스토리지
CaaS			
특징	• 내장 드라이브와 동일한 원시 장치 • 파일 시스템은 자유롭게 선택	• 네트워크 드라이브 • 파일 시스템은 스토리지임 해, 변경 불가	• 객체 단위로 액세스 • 파일 시스템 독립적 • 대량 데이터 저장
데이터 전송 프로토콜	• iSCSI, FC, NVMe	• NFS,SMB	• HTTPS(HTTP)
마운트 경로	/dev/sda	₩₩192.168.0.1₩share₩hoge	https://aaa.org/storage/hoge
성능*	• High	• Middle	• Low
주요 용도	• DB, OS(Boot Disk)	• 파일 공유	• 사진, 동영상 저장

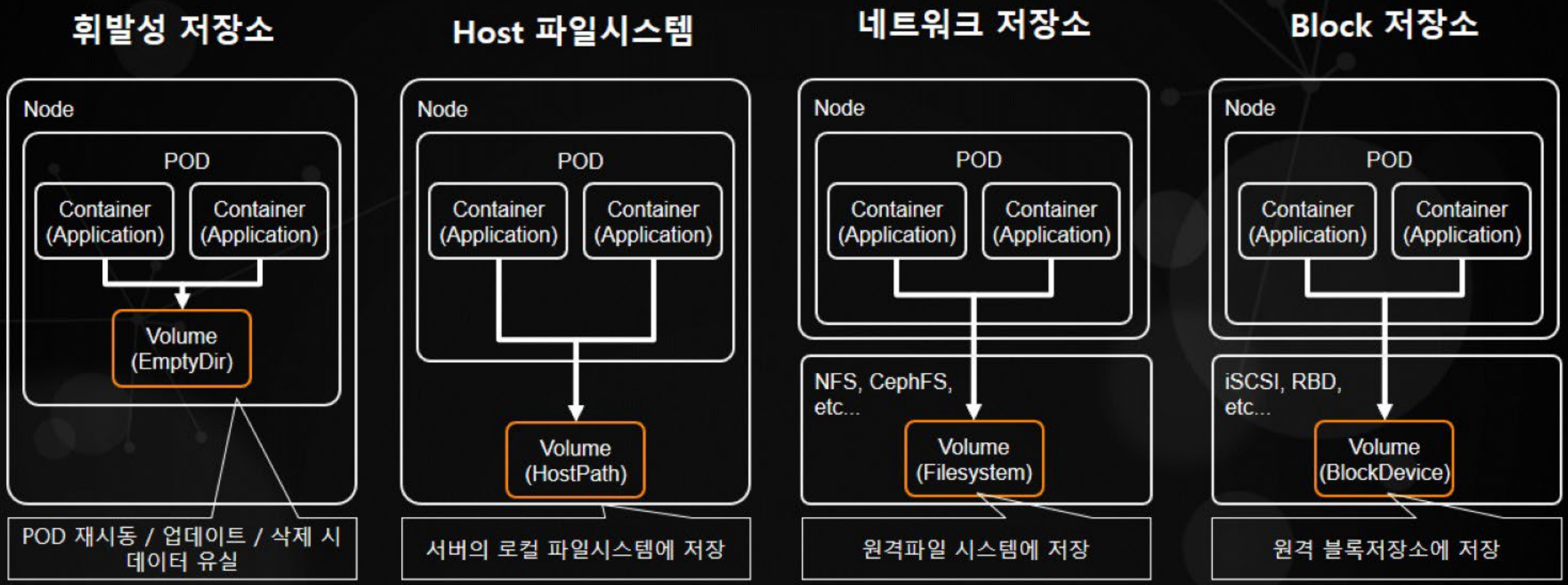
OpenShift 스토리지 연결 > 스토리지 모델

- 벤더 독립 모델
  - PersistentVolumeClaim
  - PersistentVolume
  - StorageClass
- 관리자와 사용자의 역할을 고려한 모델
- Volume 을 컨테이너에 연결
- 주로 블록 스토리지/파일 스토리지 대상



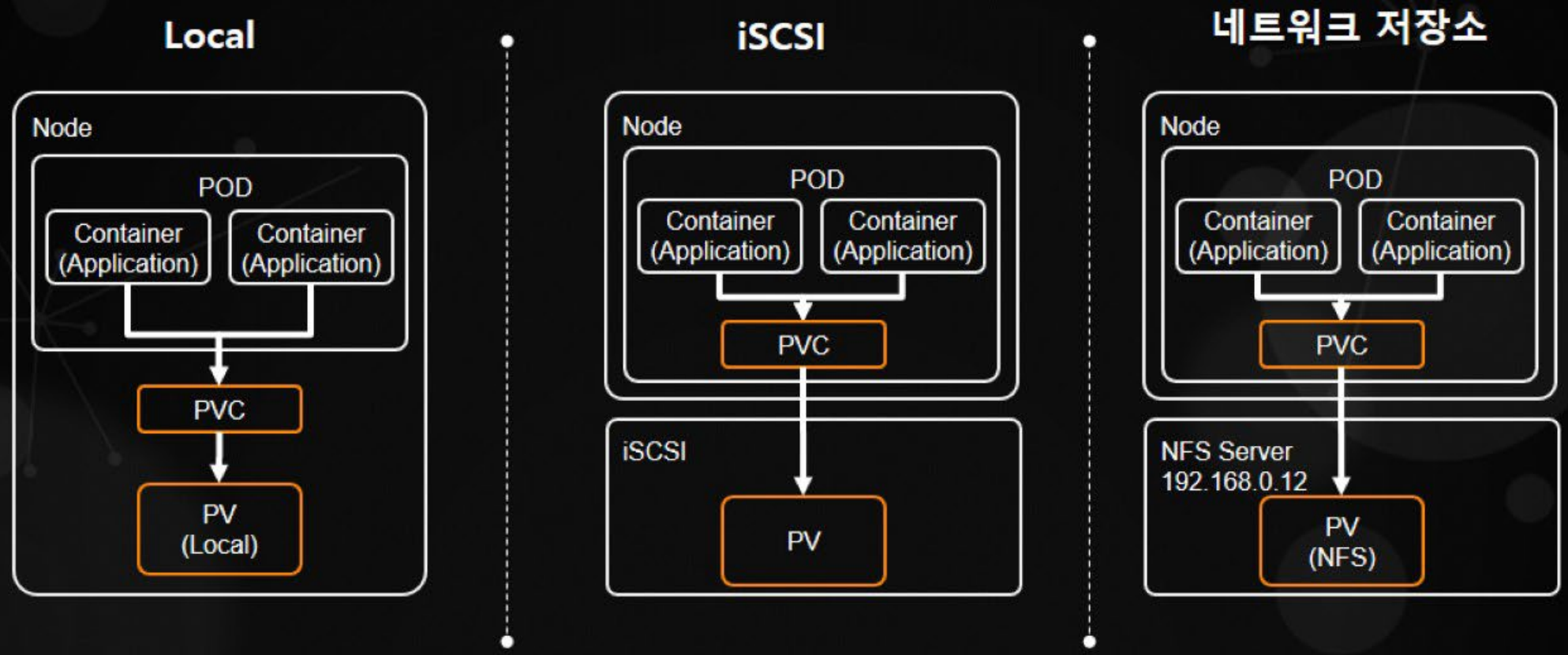
OpenShift 스토리지 연결 > Pod에서 Volume 을 지정하는 방법

- 다양한 volume을 직접 지정하는 방식
- 컨테이너에서 데이터를 저장하는 공간으로 volume을 별도로 지정 가능하도록 정의



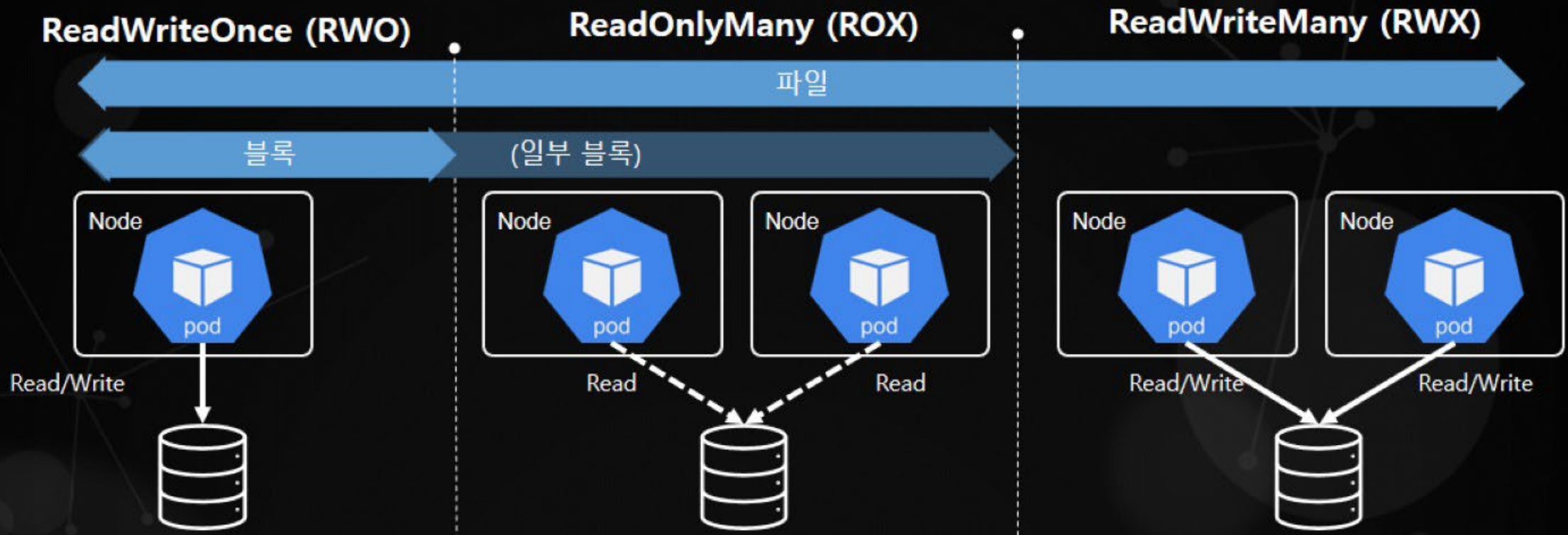
OpenShift 스토리지 연결 > Pod에서 Persistent Volume 을 지정하는 방법

- pod에서 추상화된 가상의 volume 객체 (persistent volume claim)를 생성하고, 실제 volume의 유형, 사이즈 등 세부적인 스펙은 persistent volume을 연결하는 방식
- POD는 PVC 만 연결하므로, 실제 volume이 변경되면 pvc에서 persistent volume만 다른 것으로 교체
- PVC 를 통해 언제든지 PV 변경이 가능



OpenShift 스토리지 연결 > 스토리지 AccessModes

- 액세스 제어 방법을 식별하는 모드
  - Kubernetes/OpenShift 에서 액세스 제어



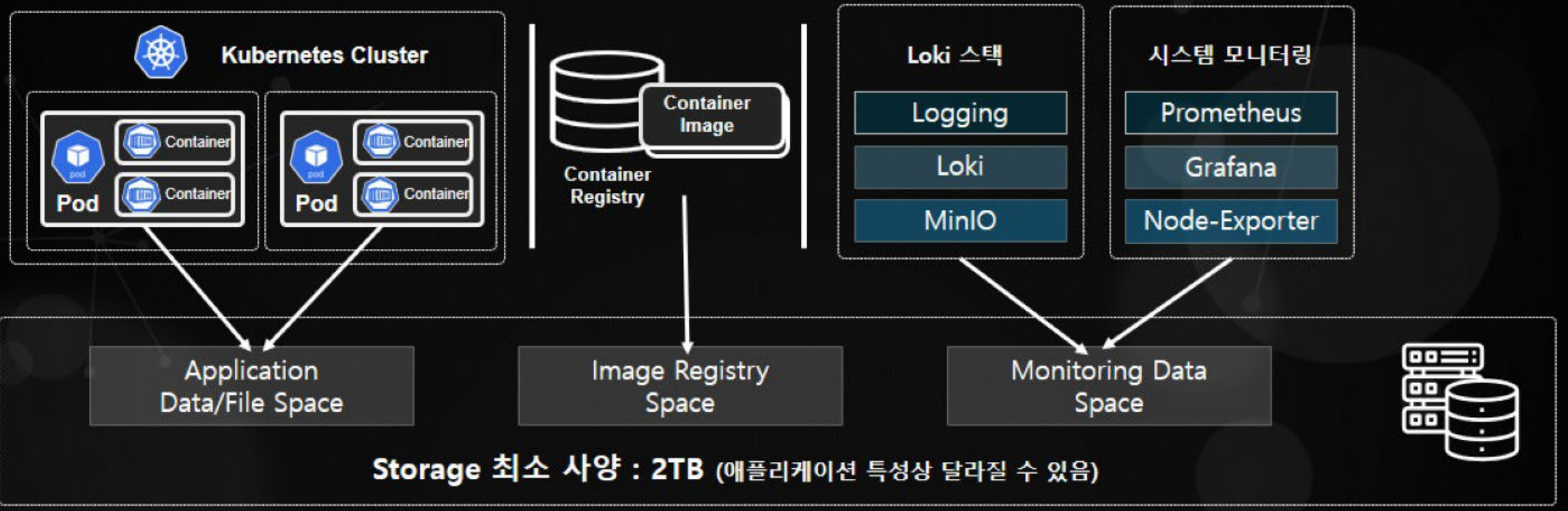
- 하나의 노드에서 Read / Write로 마운트 가능
- 거의 모든 스토리지에서 사용 가능
- 가장 많이 사용되는 기본 모드

- 여러 노드에서 Read Only로 마운트 가능
- 용도가 제한적임

- 여러 노드에서 Read/ Write로 마운트 가능
- 기본적으로 파일 스토리지에서 사용 가능

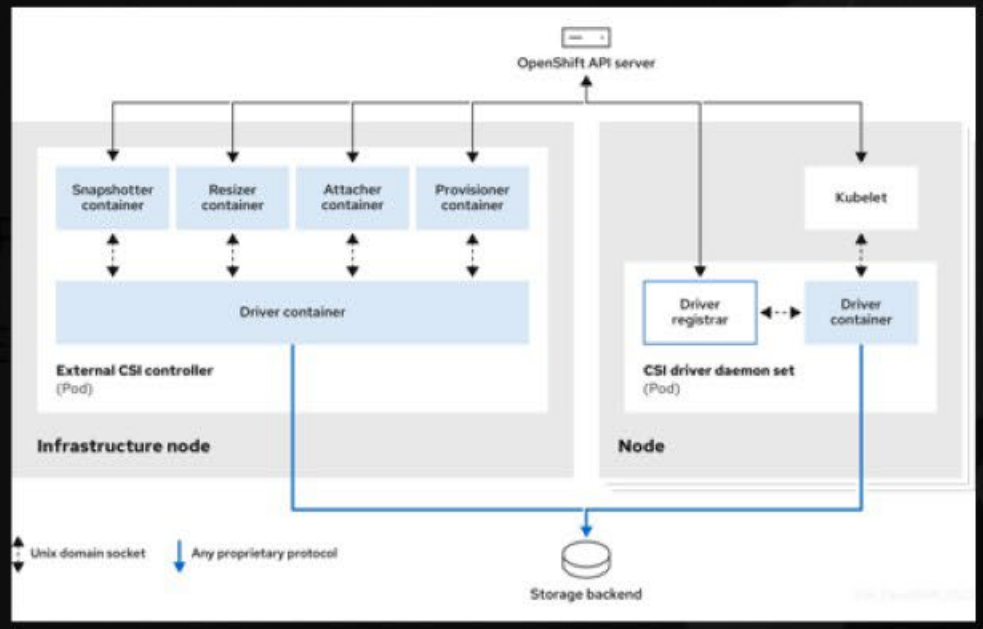
OpenShift 스토리지 연결 > Storage는 왜 필요한가?

- 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



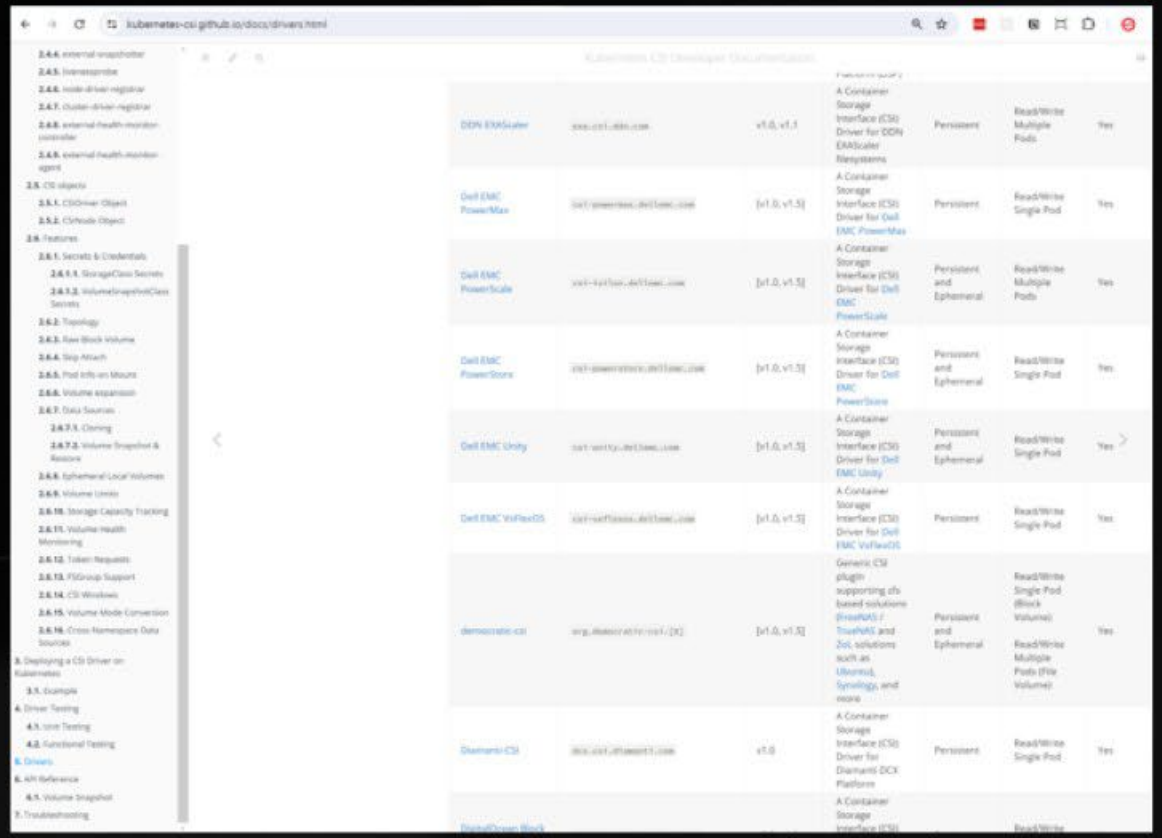
OpenShift 스토리지 연결 > CSI(Container Storage Interface)

- 컨테이너 오케스트레이션을 위한 표준 사양
- Kubernetes는 StorageClass의 Provisioner 인터페이스로 채택
- 2017년 12월 첫 공개 공개
- 주요 컨테이너 오케스트레이션 채택
  - (Kubernetes, CloudFoundry, Mesos, Docker, etc)
- Kubernetes의 소스 트리(in-tree)로 관리되고 있던 스토리지 장치와 관련된 구현이 3rd 벤더에서 독자적으로 개발/제공할 수 있다



OpenShift 스토리지 연결 > CSI 드라이버를 지원하는 스토리지 목록

- 최근 스토리지 업체들이 CSI 드라이버를 제공하고 있음



https://kubernetes-csi.github.io/docs/drivers.html

Platform As A Service

OpenShift 보안심사는?

OpenShift 보안은? > 보안성 심의 기준?

- **OpenShift와 같은 Container Platform S/W에 대한 보안성 심의 기준이 아직 마련되어 있지 않음**
- **따라서 각 기관 및 회사 별 자체 보안 기준에 따라 보안성 심의를 진행하고 있음**
- **대부분 OpenShift가 설치된 노드의 OS 레벨에서 보안성 심사를 진행함**
- **접근제어, 계정관리, 암호화 등 필요한 보안 요소는 OpenShift 자체 제공 혹은 연동하여 처리**

[국가정보자원관리원 : 공공]

✓ 국가정보자원관리원과 같은 공공기관은 보안제품을 도입하면 "국가정보원의 보안적합성 검사"를 받아야 함

✓ OpenShift는 보안제품으로 분류되지 않아 국가정보원의 보안적합성 검사를 받을 필요가 없었고 국가정보자원관리원의 자체 정보보안지침에 따라 보안성 심의를 진행하였음

[KCB : 금융]

✓ 금융기관은 기본적으로 금융감독원의 보안성 심의 기준에 따라 보안 검사 진행

✓ OpenShift 구축 시 금융감독원 및 KCB 자체 정보 보안 심의 기준에 따라 보안성 검토를 진행함
(주로 노드의 OS에 대한 보안 심의 진행)

[롯데카드 : 금융]

✓ 금융기관은 기본적으로 금융감독원의 보안성 심의 기준에 따라 보안 검사 진행

✓ OpenShift 도입 시 금융감독원 및 롯데카드 자체 정보 보안 심의 기준에 따라 보안성 검토를 진행함
(주로 노드의 OS에 대한 보안 심의 진행)

OpenShift 보안은? > 필수 보안 프로그램 설치 방안

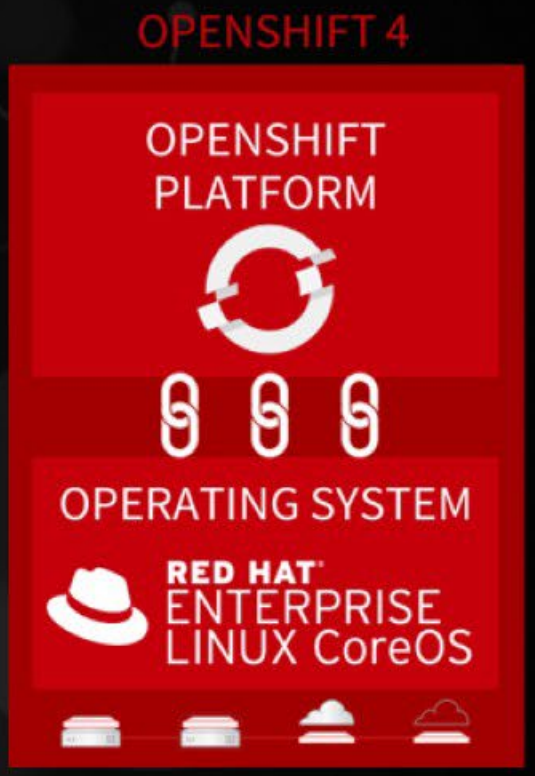
OpenShift 보안은? > RHEL과 CoreOS비교

openmaru

	RED HAT ENTERPRISE LINUX General Purpose OS	RED HAT ENTERPRISE LINUX CoreOS Immutable container host
BENEFITS	• 10 년 이상의 기업 수명주기 • 업계 표준 보안 • 모든 인프라에서 높은 성능 제공 • 파트너 솔루션의 광범위한 생태계와 사용자 정의 및 호환 가능	• 자체 관리, 원격 업데이트 • OpenShift와 불변하고 긴밀하게 통합 • 호스트 격리는 컨테이너를 통해 수행 • 일반적으로 인프라에 최적화 된 성능
WHEN TO USE	사용자 지정 및 추가 솔루션과의 통합이 필요한 경우	클라우드 기본, 핸즈프리 작업이 최우선 과제 인 경우

Core OS

- **Based on RHEL8/9**
- Kernel 4.18.x
- OpenShift를 위한 모든 패키지 포함
- **Immutable OS**
- OpenShift와 CoreOS에 대한 **Over-The-Air 업데이트**
- 거의 모든 환경 관련 설정을 미리 정의
- 원격 업데이트



OpenShift 보안은? > CoreOS FAQ

- OpenShift CoreOS에는 보안 에이전트, 모니터링, 로그 에이전트등 우리 회사에서 꼭 설치해야할 보안 5종 프로그램을 설치하지 못하나요?
  - 예, 설치할 수 없습니다. Immutable 이미지 기반의 OS라 RPM을 설치하지 못하며, 컨테이너만 실행할 수 있습니다.
- 보안 5종 에이전트가 없어도 안전한가요?
  - 오히려 더 안전합니다. CoreOS의 중요 파일시스템들은 수정하더라도 재부팅하면 초기화되며, 일반적인 SSH 로그인 등은 모두 제한되어 있습니다.
- 회사 정책상 보안 에이전트를 설치해야 하는데 어떻게 해야 하나요?
  - VMWare의 ESXi도 같은 전용 OS로 Appliance로 예외처리 하셨을 것 입니다. 동일합니다.
- OpenShift Host에 필요한 소프트웨어는 어떻게 설치하나요?
  - 컨테이너로 실행할 수 있으면 됩니다. 벤더사에 S/W가 컨테이너를 지원하는지 확인해 주세요.
- CoreOS를 도입한 사례가 있나요?
  - 공공기관, 금융기관, 민간기업 등 다수의 사례가 있습니다.

OpenShift 보안은? > 컨테이너 환경에서 필요한 보안

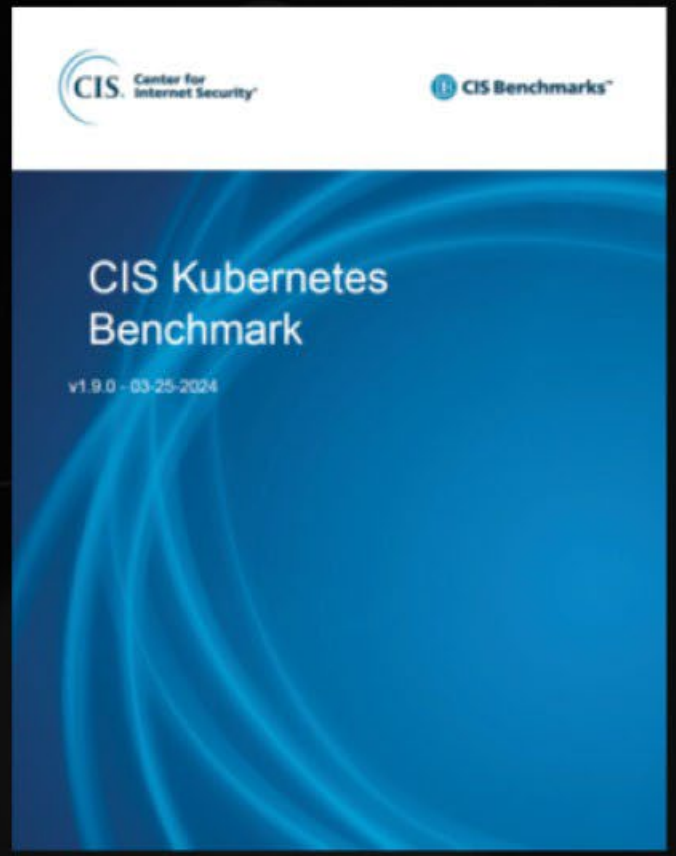
왜 AS-IS 환경의 보안들이 필요하지 않을까?

실질적으로 컨테이너환경에서 필요한 보안들

- 컨테이너 실행 권한에 대한 보안 : SCC
  - Container breakout
- 취약한 Container Image 사용
  - 악성코드, 채굴 프로그램이 포함된 이미지
- Role Base Access Controle : RBAC
  - 사용자별 필요 권한 부여
- 컨테이너 플랫폼의 Audit 로깅
  - EFK
- 신뢰할 수 있는 컨테이너 런타임 보안
  - Capabilities, SELinux, Seccomp & Namespace
- 컨테이너간의 네트워크 격리
  - Network Policy

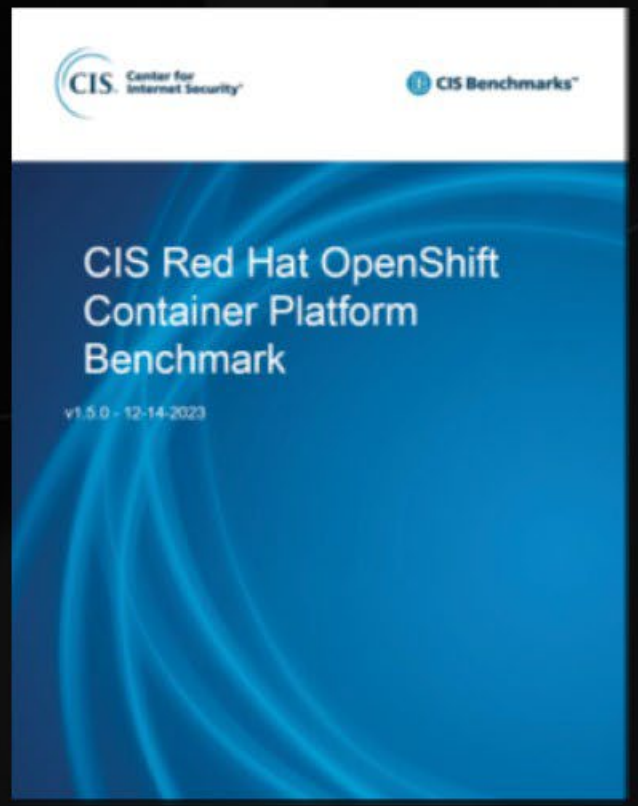
OpenShift 보안은? > 최근 보안업체에서 컨테이너 보안가이드 출시

- 최근 보안업체에서도 CIS(Center for Internet Security)의 Kubernetes Benchmark등의 자료를 기반으로 Kubernetes 보안 가이드를 출시



OpenShift 보안은? > OpenShift에 맞는 컨테이너 보안진단 체크리스트 준비

- Kubernetes에 비해 OpenShift는 모든 기반 컴포넌트를 컨테이너로 실행하여 CIS(Center for Internet Security)에서도 별도의 CIS Red Hat OpenShift Container Platform Benchmark 보안 레포트를 발행
- 오픈마루에서 국내 보안업체 수준의 OpenShift 보안 가이드를 작성하여 고객사에 적용




Platform As A Service
OpenShift 업그레이드
openmaru
APM

OpenShift 업그레이드 > 클라우드 네이티브 도입시 OS 업그레이드 절차 비교

openmaru

OpenShift 환경의 자동화된 OS 업그레이드

- 기존 환경과 달리 **컨테이너 기반의 애플리케이션이기 때문에 의존성 최소화**
- 업그레이드는 **관리자 개입 없는 OpenShift 자동 수행**
- **Red Hat에서 Upgrade Path 제공**
- **호스트 OS에 대한 업그레이드도 함께 진행됨**

Legacy

5. 서비스 확인

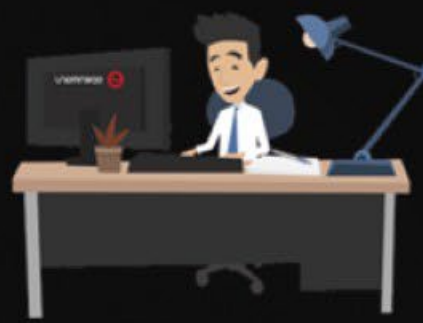
4. 업그레이드 수행
   - RPM/Kernel Upgrade
   - OS 튜닝

3. Repository 설정
   - Minor Upgrade Repo
   - Major Upgrade Repo

2. 서비스 중지

1. 의존성 조사
   - Runtime 버전
   - 3rd party 버전
   - OS 튜닝 등

OpenShift



3. 서비스 확인

Upgrade 자동화

2. 업그레이드 수행

1. Registry 설정

OpenShift 업그레이드 > EOS 대응을 위한 OS 업그레이드 소요시간

openmaru

CentOS To Red Hat Enterprise Linux

- 최신 마이너버전이 아닐 경우 최신 메이저 버전으로 업그레이드 불가
- VM 및 Baremetal의 수량에 따라 작업시간 증가

CentOS 7.x Version
↓
CentOS/RHEL 7 Latest Version
↓
CentOS Stream /RHEL 8.x Version
↓
CentOS Stream /RHEL 8 Latest Version
↓
CentOS Stream/RHEL 9 Latest Version

X VM or Baremetal Count

서버 1대에 약 2시간 소요 → **100대 VM: 400시간**

CentOS To OpenShift Container Platform

- OpenShift의 경우 OpenShift 버전 업그레이드시 Host OS(RHCOS) 자동 업그레이드 진행
- 클러스터 단위로 업그레이 진행

OpenShift 4.12.x Version
↓
OpenShift 4.12 Latest Version
↓
OpenShift 4.13 Version

X OpenShift Cluster Count

Worker Node 6대 → **1개 클러스터: 7시간**

OpenShift 업그레이드 > 업그레이드 절차

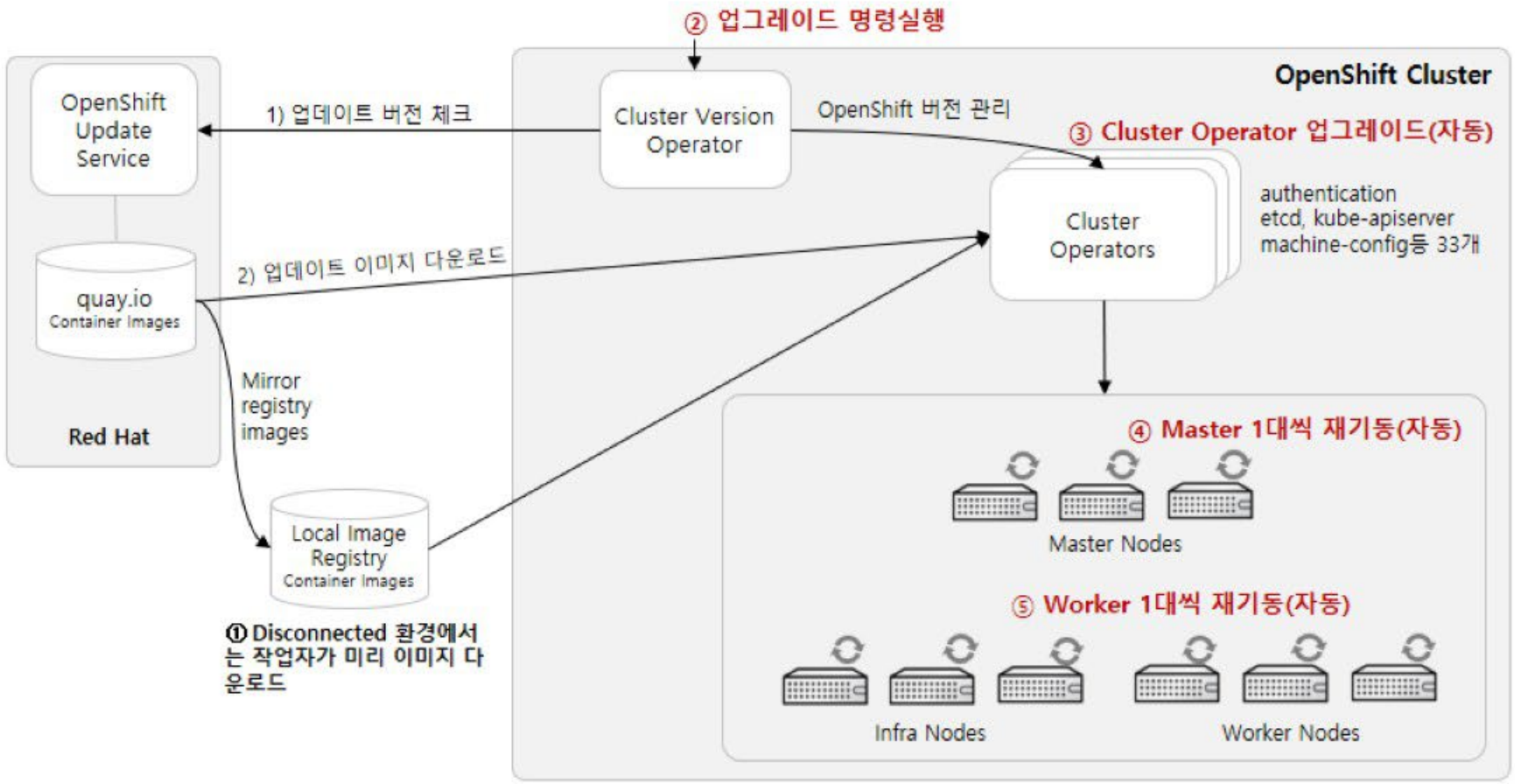
① [Disconnected일 경우] OpenShift Upgrade Mirror 이미지 Registry 준비

② OpenShift Upgrade 버전 선택 & **명령 실행**(이후 프로세스는 **모두 자동으로 진행**됨)

③ **Cluster Operator 업그레이드**(33개)

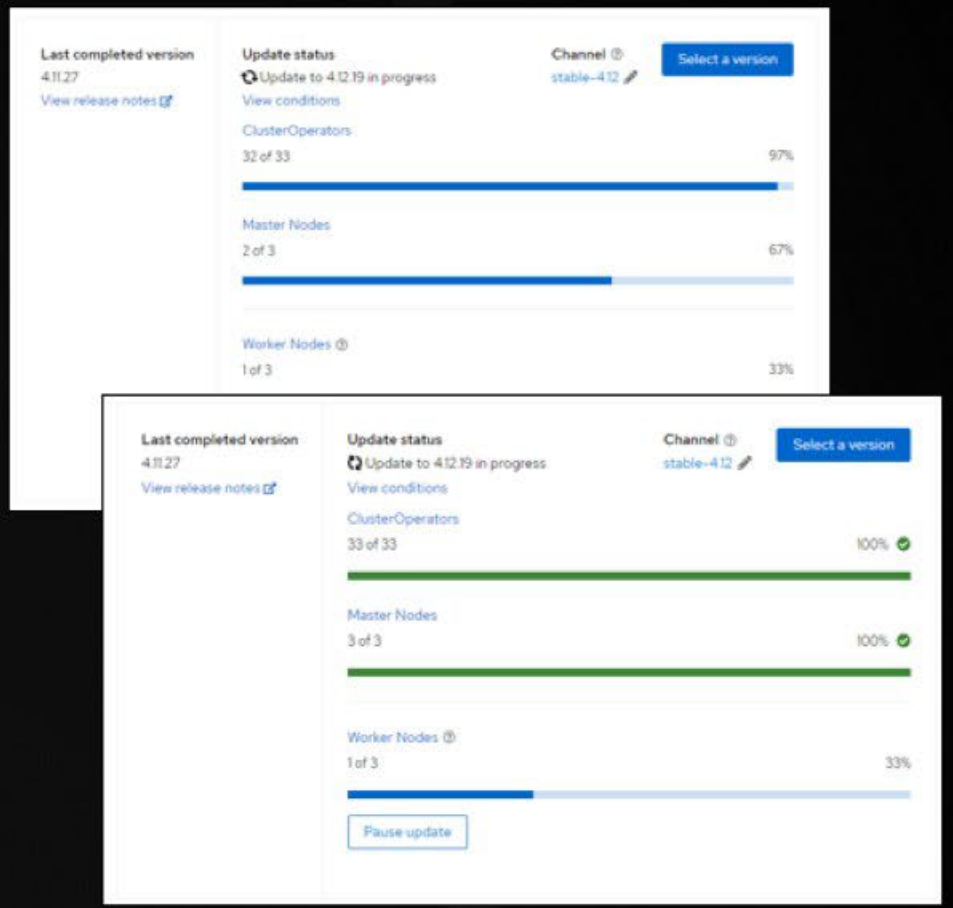
④ Master 노드 3대 **순차적 CoreOS 업그레이드 & 리부팅**

⑤ Worker 노드 n대 **순차적 CoreOS 업그레이드 & 리부팅**



OpenShift 업그레이드 > 업그레이드 주의 사항

- 특이사항
  - 3중화된 서버를 1대씩 재기동되기 때문에 서비스는 중단되지 않음
  - 머신의 부팅시간이 5분 40초 이상(Pod Evict 시간) 걸릴 때 Worker노드의 Pod가 다른 머신으로 재분배되기 때문에 Worker 노드의 자원(CPU, Memory) 1대 머신의 POD가 재분배될 만큼 여유있어야 함
  - 업그레이드가 실패하더라도 Cluster Operator가 계속 재시도하고 있을 뿐, 업무 서비스(POD)는 중지되지 않음(레드햇 기술지원 서비스를 받아 처리할 수 있음)
  - OpenShift Cluster를 Downgrade 하는 것은 지원하지 않음
- 주의 사항
  - 2개 이상의 Pod가 기동 중이어야 한 대의 Worker 머신이 리부팅 되더라도 서비스가 중단되지 않음
  - 이중화(HA)를 설정하지 않은 DBMS를 Pod로 사용중인 경우, DBMS Pod를 중지하고 다른 머신에 띄워야 해서 중단이 필요함
  - 동시에 하나의 Pod만 Write할 수 있는 ReadWriteOnce 형식의 Persistence Volume를 사용하는 경우 중단이 필요함



OpenShift 업그레이드 과정 예시
(4.11.27 → 4.12.19)


Application Performance Management
OpenShift Backup
openmaru
APM

OpenShift Backup > 백업 대상은?

openmaru

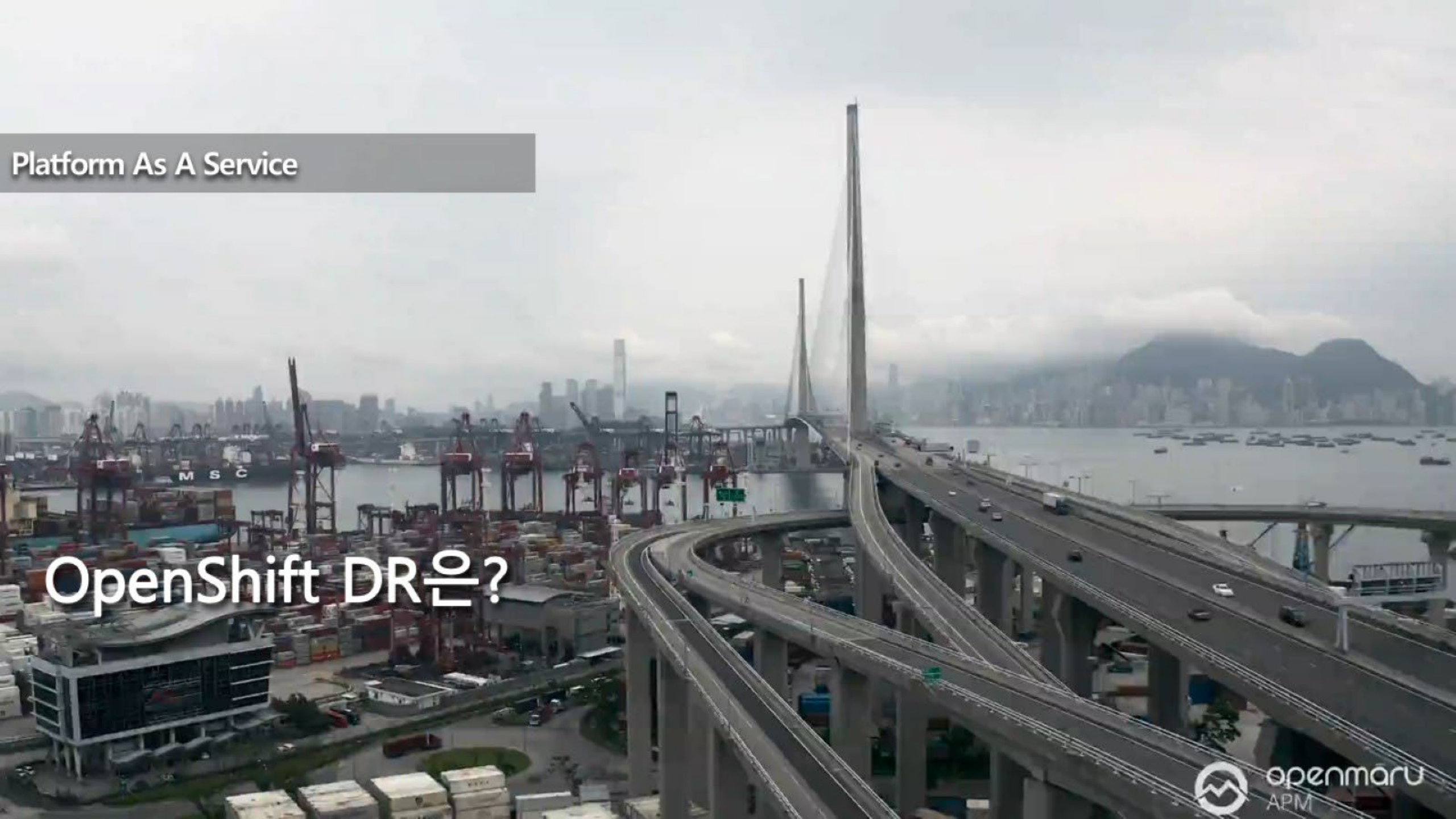
백업 종류	백업 대상	상세 내용	비고
클러스터 백업	• 클러스터 전체 구성정보(etcd) • Registry storage • 애플리케이션 데이터	• etcd 설정파일/데이터 백업 • Container Image • 애플리케이션 데이터	• OS 레벨의 스케줄링(ex. Cron)으로 자동화 • PV Registry Storage는 스토리지 레벨 백업 • 애플리케이션 데이터 또한 스토리지 레벨 백업

프로젝트 백업	• 프로젝트 구성 정보
	• 프로젝트 상태 정보
	• 프로젝트 내 모든 리소스 정보
		• 프로젝트 내 모든 Object (애플리케이션 포함)
		  • Namespace / Project
		  • Role Bindings
		  • Service Accounts
		  • Secrets
		  • Persistent Volume Claims
			• OS 레벨의 스케줄링(ex. Cron)으로 자동화
			• 또는, Application Migration Toolkit으로 백업
			oc get -o yaml --export all > project.yaml
			oc get -o json --export all > project.json

	• 애플리케이션 데이터	• 애플리케이션 생성 데이터 (Persistent Volume/Persistent Volume Claims)	• 애플리케이션에 따라 다름 (Stateful만 해당) • PV를 사용할 경우 스토리지 레벨 백업
	• 애플리케이션 구성정보	• Build Config • Service • Route • Secrets • Configmap • Deployment Config / Deployment • Image Stream	• OS 레벨의 스케줄링(ex. Cron)으로 자동화 oc get all --selector app=frontend -o yaml –export ₩ > frontend.yaml

OpenShift Backup > 백업 방법은?

- Etcd 백업 방법
  - OpenShift의 Cronjob으로 설정하여 주기적 백업
- OpenShift의 프로젝트 백업
  - 백업 스크립트를 이용한 백업
  - OADP(OpenShift API for Data Protection; Velero기반)를 이용한 백업(S3 저장소 필요)
  - GitOps를 이용하여 구축하면 Git에 모두 저장되어 있음
- 스토리지 백업
  - 기존과 마찬가지로 백업 Agent를 설치하여 백업(별도 OS에 설치)
- VM 백업
  - 마스터, Infra가 VM이라면 VM 스냅샷 기능을 이용한 백업
- 전문 백업툴을 이용한 백업
  - Veeam의 Kasten과 같은 OpenShift(Kubernetes) 전문 백업 도구가 있음
  - 스토리지, 프로젝트, Etcd 항목 별로 백업기능 제공함


Platform As A Service
OpenShift DR은?
openmaru
APM

OpenShift DR > DR 종류별 특징

openmaru

Cold DR

일부 인프라는 있지만 서비스를 복원하는데 필요한 전체 리소스(하드웨어, 소프트웨어)는 없습니다.

- 데이터만 DR Zone에 보관하고, 서비스를 위한 시스템은 확보하지 않거나 최소한으로 확보한다.
- 재해 발생시 데이터를 근간으로 시스템을 복구를 개시하는 방식
- RPO가 수주이며 RTO도 수주 수개월이다.

Warm DR

서비스를 기동시킬 일부 하드웨어, 소프트웨어가 갖춰져 있지만 고객 데이터가 포함되어있지 않습니다.

- 중요도에 따라 중요성이 높은 시스템만 부분적으로 DR Zone에 구성하는 방식
- 실시간 미러링을 진행하지 않고 (RPO가 수시간~1일) 재해 발생시 RTO가 수일 ~ 수주

Hot DR

운영이 가능한 상태의 시스템을 Standby 상태로 대기합니다.

- 동기적, 비동기적 실시간 미러링을 통해 최신의 상태를 유지하며 재해시 RPO=0(목표복구시점) 을 지향하는 방법.
- 재해 발생시 RTO(목표소요시간)은 수시간(약 4시간 이내).
- 데이터베이스의 경우 Stanby로 있다가 재해 발생시 Active로 전환하는 방식이 일반적.

Mirror DR

운영 환경과 동일 상태이며 Active 상태로 운영 시스템과 동시에 서비스 합니다.

- 재해 발생시 복구까지의 소요시간(RTO)가 즉시(이론적으로는 0)이다.
- 초기투자 및 유지보수에 높은 비용이 소요됨.
- 데이터베이스 같이 데이터의 업데이트 빈도가 높은 시스템의 경우 Sync를 위한 높은 부하가 요구됨.
- 웹 애플리케이션 같은 데이터 업데이트 빈도가 낮은 시스템에 적합

OpenShift DR > DR 환경 구축을 위한 고려사항

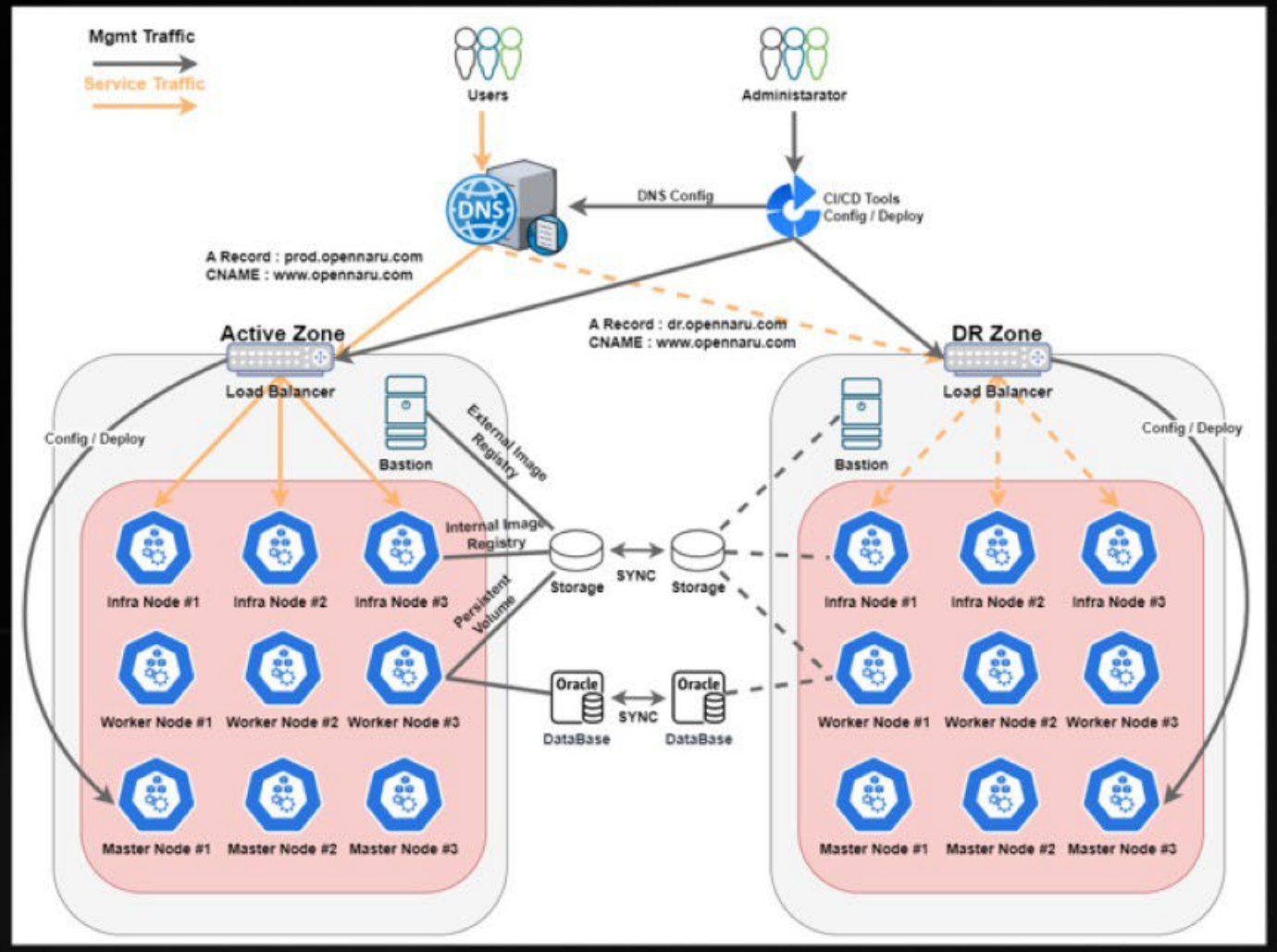
openmaru

- 네트워크
  - **도메인 기반의 로드밸런싱을 어떻게 구현할 것인지?**
    - Active / DR이 같은 도메인 이름을 가지게 됨
    - Active 환경에서 장애시 DR로 라우팅해야 함

- 애플리케이션
  - **서비스를 어떻게 백업 사이트로 백업하고 복구 할 것인지?**
    - Active 사이트에서 운영중인 애플리케이션의 버전, 이미지를 백업사이트로 똑같이 복제해야 함
    - 클러스터 설정을 동일하게 유지해야 함

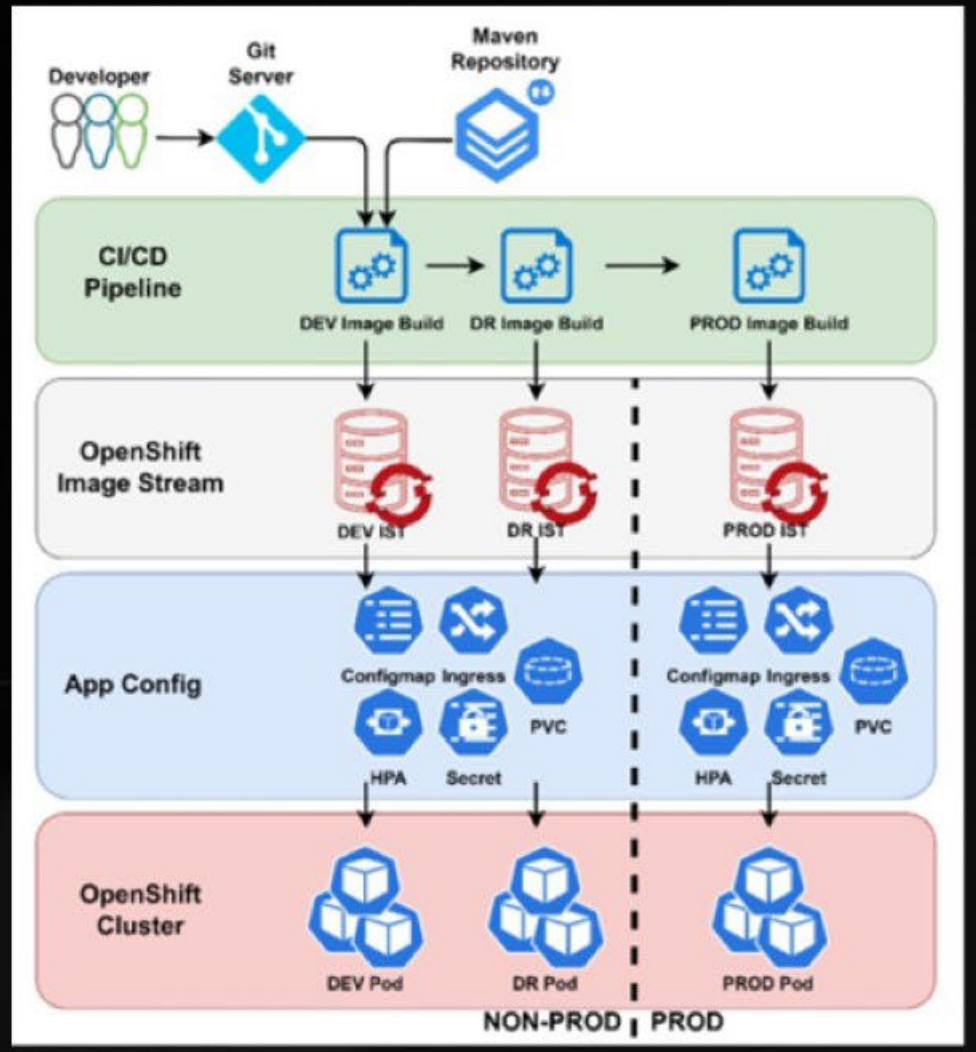
- 저장 인프라
  - **데이터베이스, 스토리지 데이터를 어떻게 백업 및 복구할 것인지?**
    - 기존과 동일한 데이터베이스 및 스토리지 백업 방법을 적용함

OpenShift DR > Active Standby DR 구성방안



- DR Zone은 서비스를 하지 않는 Standby 상태
- CI/CD 및 자동화 툴을 이용하여 설정, 배포 자동화
  - 애플리케이션 배포
  - 클러스터 설정
  - DNS 설정 등
- 주사이트 장애 발생시 DNS 레코드 변경하여 DR로 라우팅
  - DNS A 레코드, CNAME
- 스토리지와 데이터베이스는 기존 백업/복구 솔루션 활용

OpenShift DR > CI/CD를 활용한 Active / Standby DR 방법



CI/CD 도구를 활용한 Active / Standby DR 방식

- 운영환경에 배포하기 전/후 DR 사이트에 배포를 진행하는 CI/CD 파이프라인을 구성

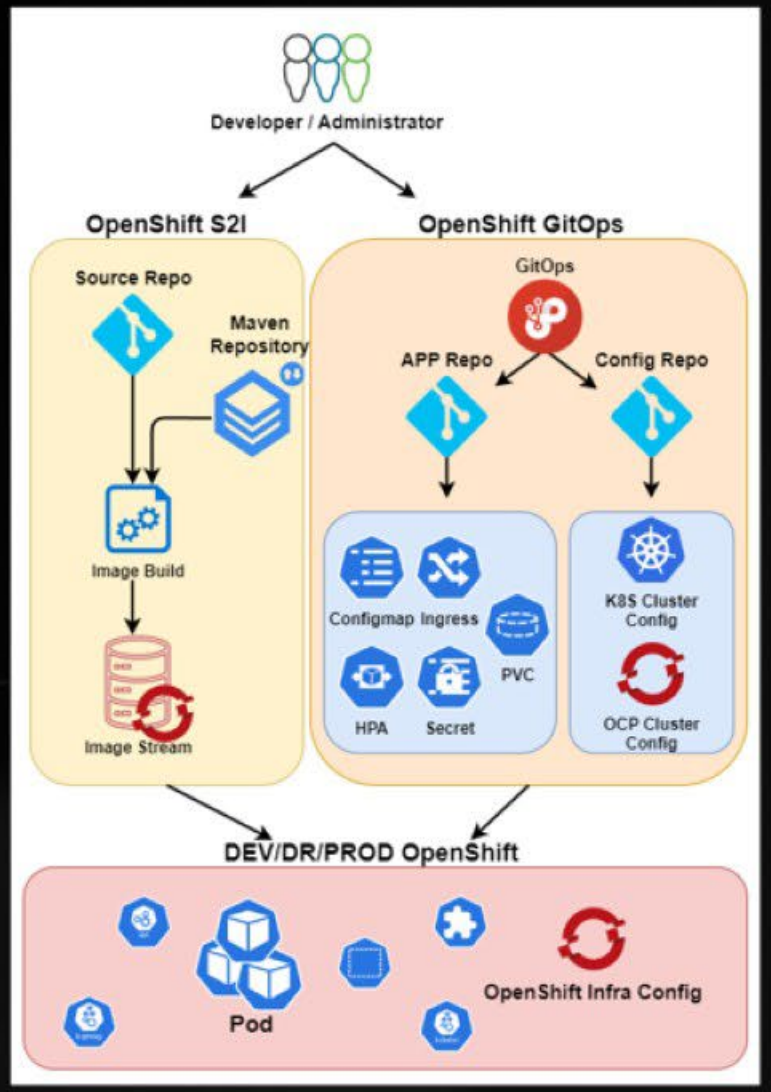
장점

- 별도의 DR 관리 솔루션이 필요없음
- CI/CD 파이프라인이 있다면 비교적 수월하게 구축가능

단점

- OpenShift Cluster간의 설정(ConfigMap, Secret등)들을 별도로 관리하여야 한다.
- 애플리케이션의 설정도 Zone에 따라 별도로 관리하여야 한다.

OpenShift DR > GitOps를 활용한 Active / Standby DR 방법



구성방안

- OpenShift에 적용할 애플리케이션, 클러스터 구성 등의 정보를 모두 Git 저장소에 구성하고, GitOps를 이용하여 Active Zone과 DR Zone을 동기화함

장점

- 애플리케이션의 모든 설정을 YAML로 Git에 관리하여 동기화 가능

단점

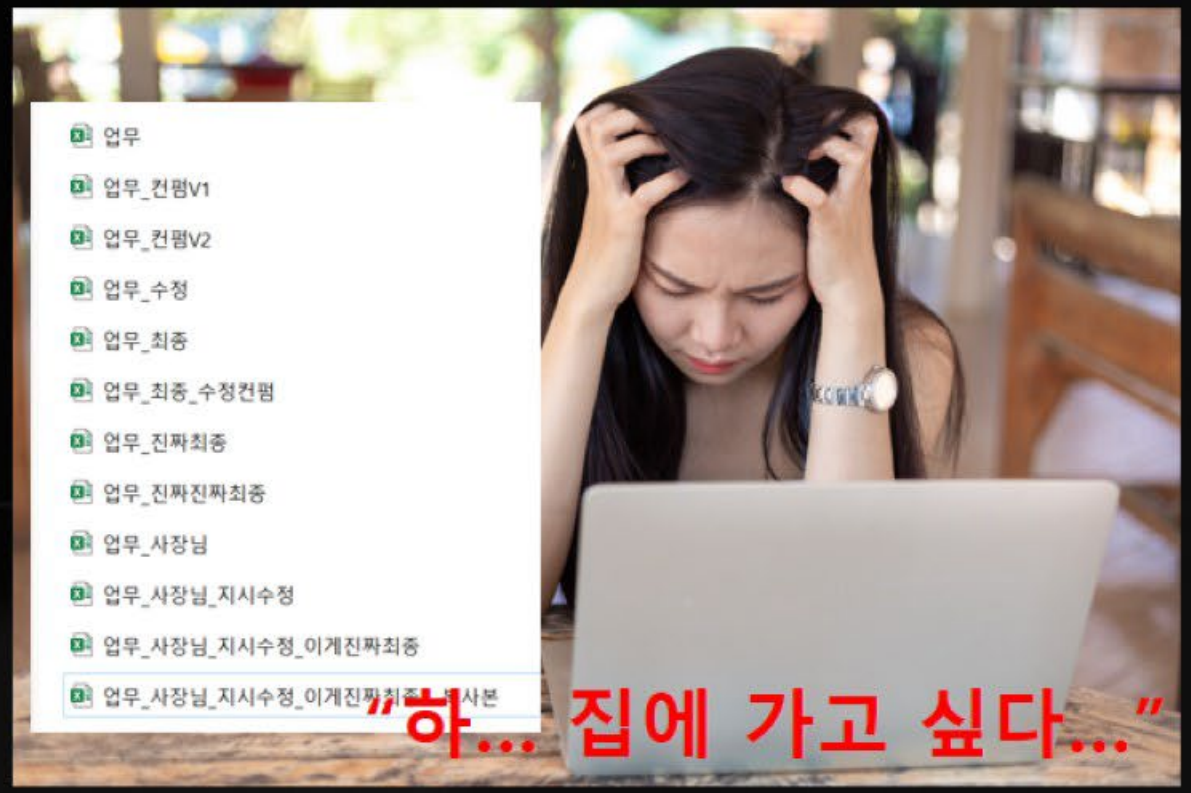
- 운영 사이트에 GitOps를 도입해야만 함

빌드/배포 GIT서버는?

형상관리란?

- 형상관리는 **변경사항을 체계적으로 추적, 통제**하는 것을 말합니다.

[형상관리가 없을 때]



어떤 버전이 진짜 최종인지 알 수 없음

[형상관리가 있을 때]

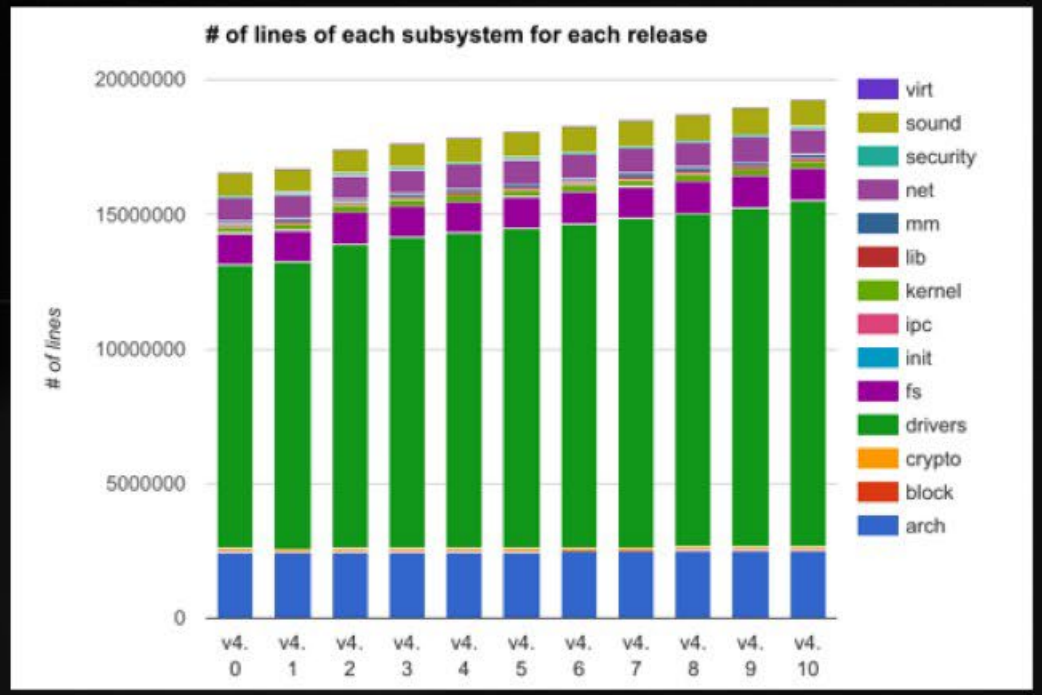
모든 변경사항이 기록되고, 원하는 버전으로 롤백가능

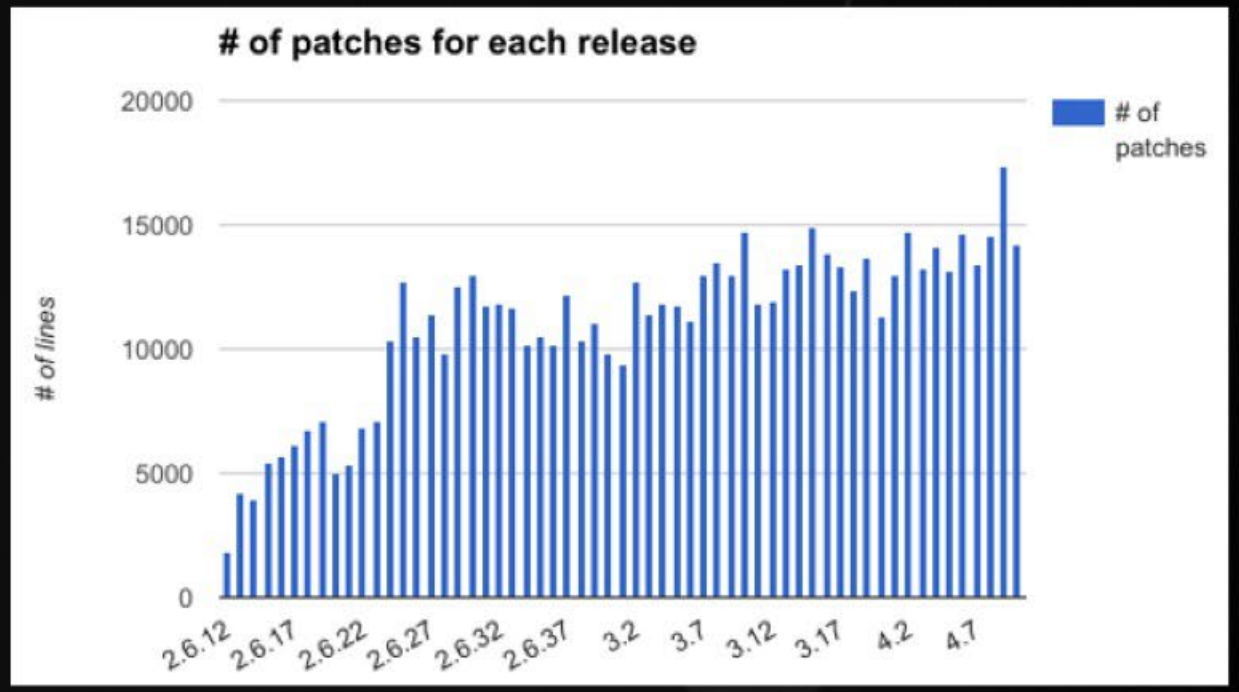
형상관리 솔루션의 종류

항목	GIT	Subversion	CVS
관리방식	분산형 버전관리	중앙집중형	중앙집중형
충돌가능성	Branch와 Merge를 통해 충돌 가능성 낮음	동시 업로드시 충돌 가능성 많음	동시 업로드시 충돌 가능성 많음
속도	빠름 작업은 로컬에서 업로드만 네트워크 이용	느림 모든 작업이 네트워크를 사용	느림 모든 작업이 네트워크를 사용
네트워크	옵션	필수	필수
최초개발	2005년	2000년대 초	1990년대 초

리눅스 커널의 소스

- 리눅스는 1991년 리누스 토발스가 개발함
- 현재 퍼블릭 클라우드 서버의 90%, 스마트폰 82%, 임베디드 기기 66%, 슈퍼컴퓨터 99%가 리눅스 사용 중
- 커널 소스는 4.10 버전의 경우 2천만라인의 코드로 약 14,000명의 프로그래머가 참여하고 있음
- 패치 버전마다 약 15,000라인이 변경됨

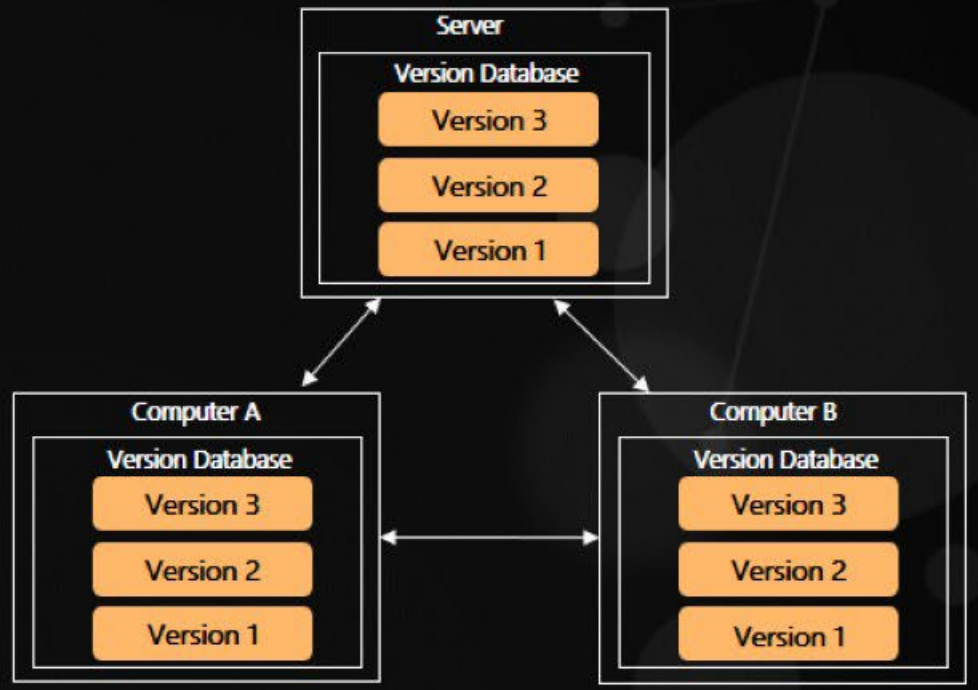




GIT은 누가 왜 만들었는가?

- 리눅스 커널은 가장 규모가 큰 오픈소스 프로젝트 중의 하나
- 1991~2002년 까지는 patch와 단순 압축 파일로 소스 관리
- 분산 버전관리시스템 BitKeeper를 사용하다가 문제가 발생함
- 리눅스 커널을 관리하기 위한 버전관리 시스템으로 2005년 리누스 토발즈가 2주만에 개발
- 버전관리 시스템의 사실상의 표준

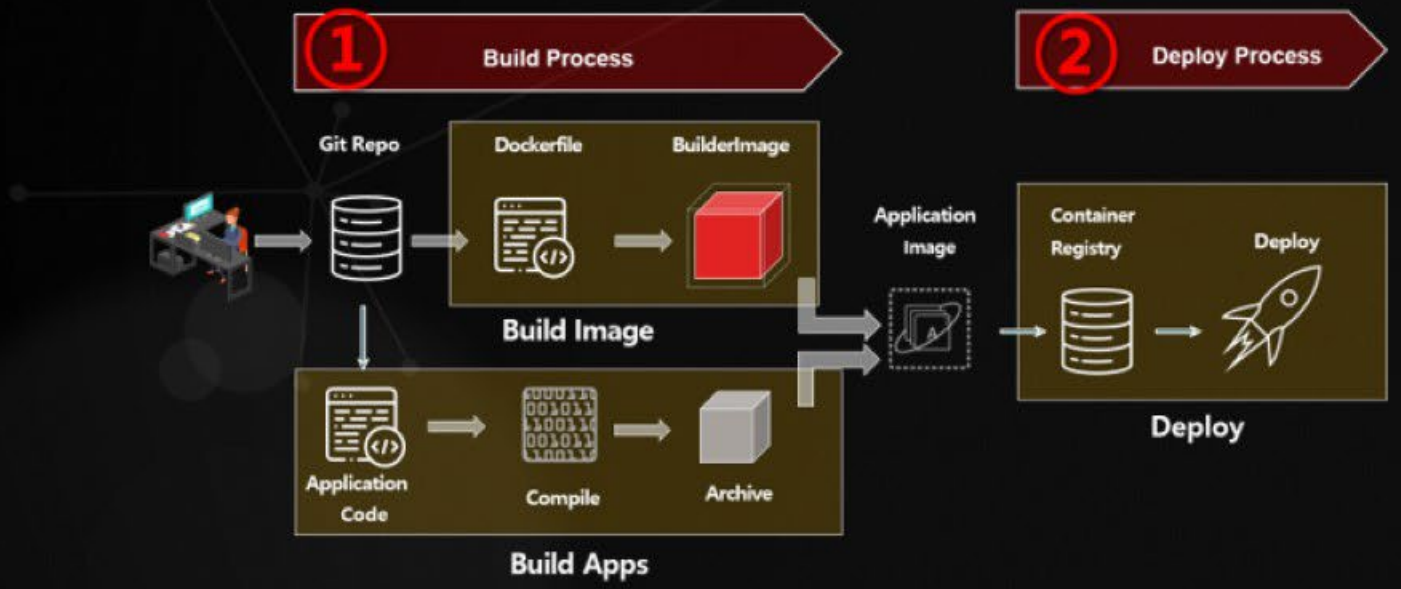
- 주요 특징과 목표
  - 빠른 속도
  - 단순한 구조
  - 비선형적 개발(전세계 개발자 동시개발)
  - 완벽한 분산
  - 대형 프로젝트에서도 유용할 것

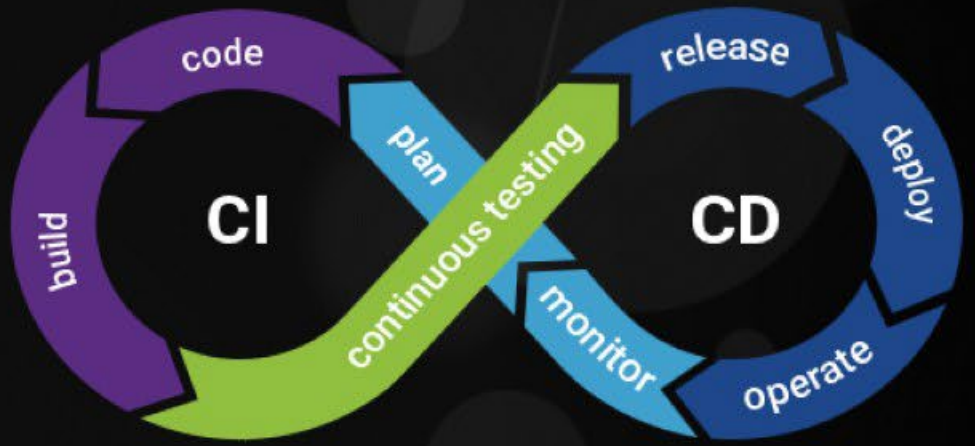


분산형 버전관리 시스템 GIT

클라우드 네이티브의 DevOps(CI/CD) 기본 환경

- 클라우드 네이티브 환경에서 DevOps를 구축하기 위해서는 CI/CD 환경이 필수
- CI/CD(Continuous Integration / Continuous Delivery)는 애플리케이션 개발단계를 자동화하여 더욱 짧은 주기로 고객에게 개선된 서비스를 제공하는 방법
- 지속적인 통합, 지속적인 배포가 이루어지기 위해서는 소스 형상관리의 소스로 부터 시작
- CI/CD 도구들은 대부분의 오픈소스 프로젝트가 사용하는 Git 저장소와 연결을 자동화
- 클라우드 네이티브 환경의 CI/CD 절차
  - GIT 소스 → 빌드 → 컨테이너 이미지 생성 → 클라우드 네이티브 환경에 배포





Git vs Github vs Gitlab

openmaru

항목	GIT	Github	GitLab
주요특징	버전관리 소프트웨어	Git 서비스(SaaS) 플랫폼	Github과 유사한 설치형 서비스 플랫폼
목적	분산 버전관리 시스템으로 프로젝트 소스를 효과적으로 관리하기 위한 도구	Git 저장소를 호스팅하고 온라인에서 소스 코드를 관리하기 위한 서비스 플랫폼	설치형으로 사용할 수 있는 Git 저장소 호스팅, 관리 플랫폼(Github 과 유사)
특징	• 로컬에서 작업하고 변경사항을 추적하기 위해서 사용됨. 네트워크 연결이 필요치 않음	• 웹기반 인터페이스를 통해 저장소 관리 • 이슈 추적, 협업 기능 제공 • Pull Request 및 코드 리뷰 등 협업지원 • Github Action CI/CD 자동화 제공 • 오픈소스는 무료 / 저장소 별 유료	• 웹기반 인터페이스를 통해 Git 저장소 관리 • 이슈트래킹, 코드리뷰, CI/CD, 협업도구, 프로젝트 관리 기능을 제공 • GitLab Runner로 CI/CD 자동화 제공 • 오픈소스, 설치형, 서비스형 모두 제공
기술지원	• 없음(오픈소스)	• 유료 서비스	• 설치형에 대한 Enterprise 기술지원 가능


openmaru

제품 / 서비스에 관한 문의

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