2020. 10.

KOREN 기술교육

네트워크를 위한 컨테이너 인프라

2020 AI NETWORK LAB BASIC

발표자: 안 종 석

2020. 10.

네트워크를 위한 컨테이너 인프라

2021년 5월까지 사용을 권장 합니다.

발표자: 안 종 석

JS Lab

목차

james@jslab.kr

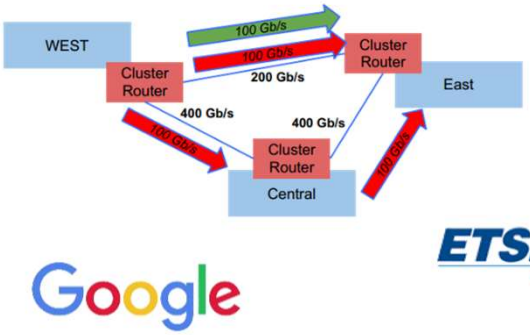
james@jslab.kr

- ❑ **Session 1: 컨테이너 인프라 개요**
  - **컨테이너 기술 발전**
  - **컨테이너 기술 특징**
- ❑ Session 2: 컨테이너 기반 인프라
  - 컨테이너 사용 인프라 구성
  - 컨테이너 종류별 기술 비교
- ❑ Session 3: 컨테이너 응용 네트워크 기술
  - 클라우드 네이티브 (K8s CNI, Service Mesh)
  - 통신 인프라 환경 컨테이너 기반 서비스

▪ 시장 기술의 변화

◻ 주요 기술 변화 History

- 2012년 4월: Google은 ONS에서 SDN 사례 발표
  - Google의 네트워크 설계 정책 변화: 50% → ≈ 100%
  - 자체 제작 네트워크 장비 사용
- 2012년 11월: ETSI는 통신사 주도로 ISG 'NFV' 설립
  - 통신사 주도 네트워크 가상화
- 2013년 3월: Docker 컨테이너 최초 버전 발표
- 2013년 4월: 리눅스 재단에서 제조사들 주도 SDN 프로젝트 OpenDaylight 발표
- 2014년 6월: Google에서 컨테이너 오케스트레이터 쿠버네티스(Kubernetes 또는 K8s) 발표



Google

docker

kubernetes

ETSI

OPEN DAYLIGHT

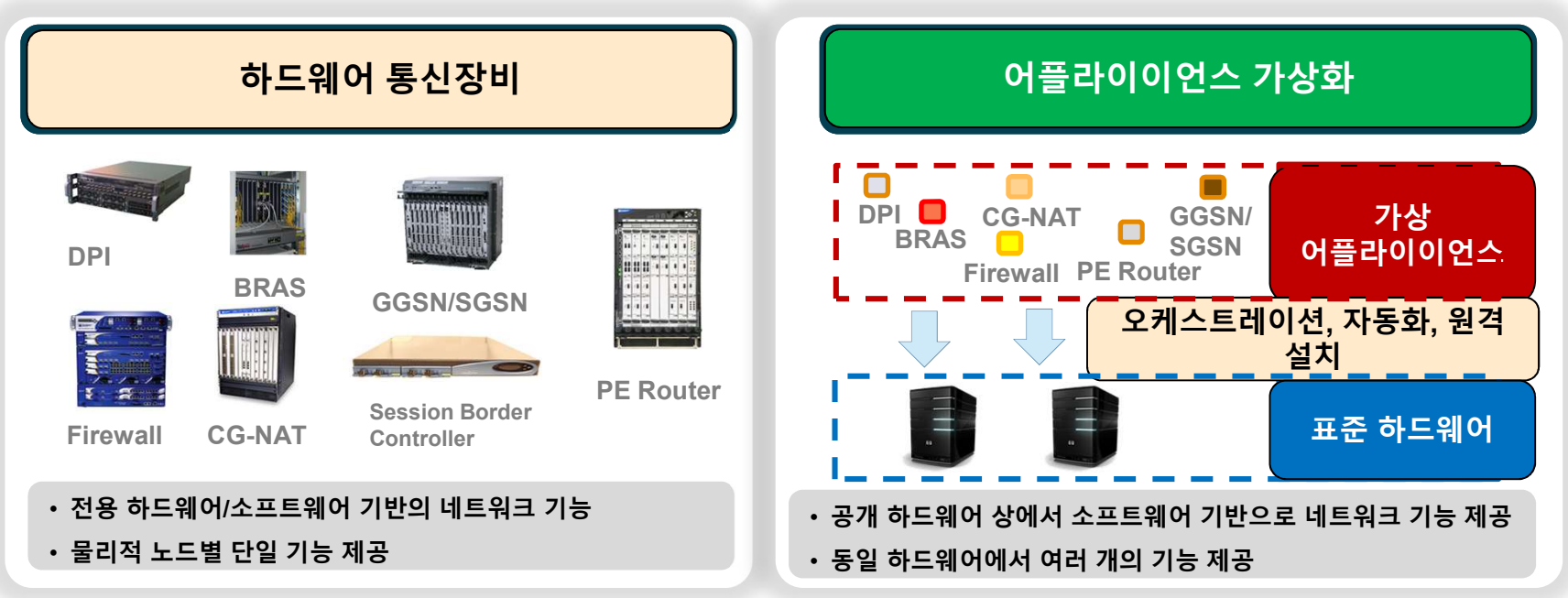
▪ 네트워크를 위한 컨테이너 인프라

- **NFV**(**N**etwork **F**unctions **V**irtualization)**는 VM**(**V**irtual **M**achine)**에서 클라우드 네이티브**(Cloud-Native)**로 발전 중**
- **클라우드 네이티브 워크로드**(Workload)**: 컴퓨팅**(Computing Unit)**은 컨테이너**(Container) **사용**
  - **쉬운 배포를 위한 패키징** (Packaging, easy for distributions)
  - **세분화된 모듈 테스트** (Fine-grained Modular Testing)
  - **이식성** (Portability)
  - **VM보다 적은 자원 사용** (Lighter than VM… less compute footprint…) → **VM 지원** (2020/10 현재 KubeVirt v0.34.0)
- **컨테이너 격리 보안강화** (Isolation as a solution: Two approaches…)
  - **유니커널 (UniKernel)**
  - **마이크로 가상머신** (Micro-VM)
- **오케스트레이션** (Managing Workloads: Orchestration)
  - **산업계 표준 '쿠버네티스'** (Kubernetes is unofficially de facto standard)
  - **멀티캐스트 코어 관리 프로토콜 LCM**(Lightweight Communications and Marshalling)**, 자가복구**(self-healing)**, 자동 성능 제어 오토스케일** (auto-scaling)**, 스케듈링** (scheduling)**, ….**

james@jslab.kr

▪ NFV (Network Functions Virtualization)

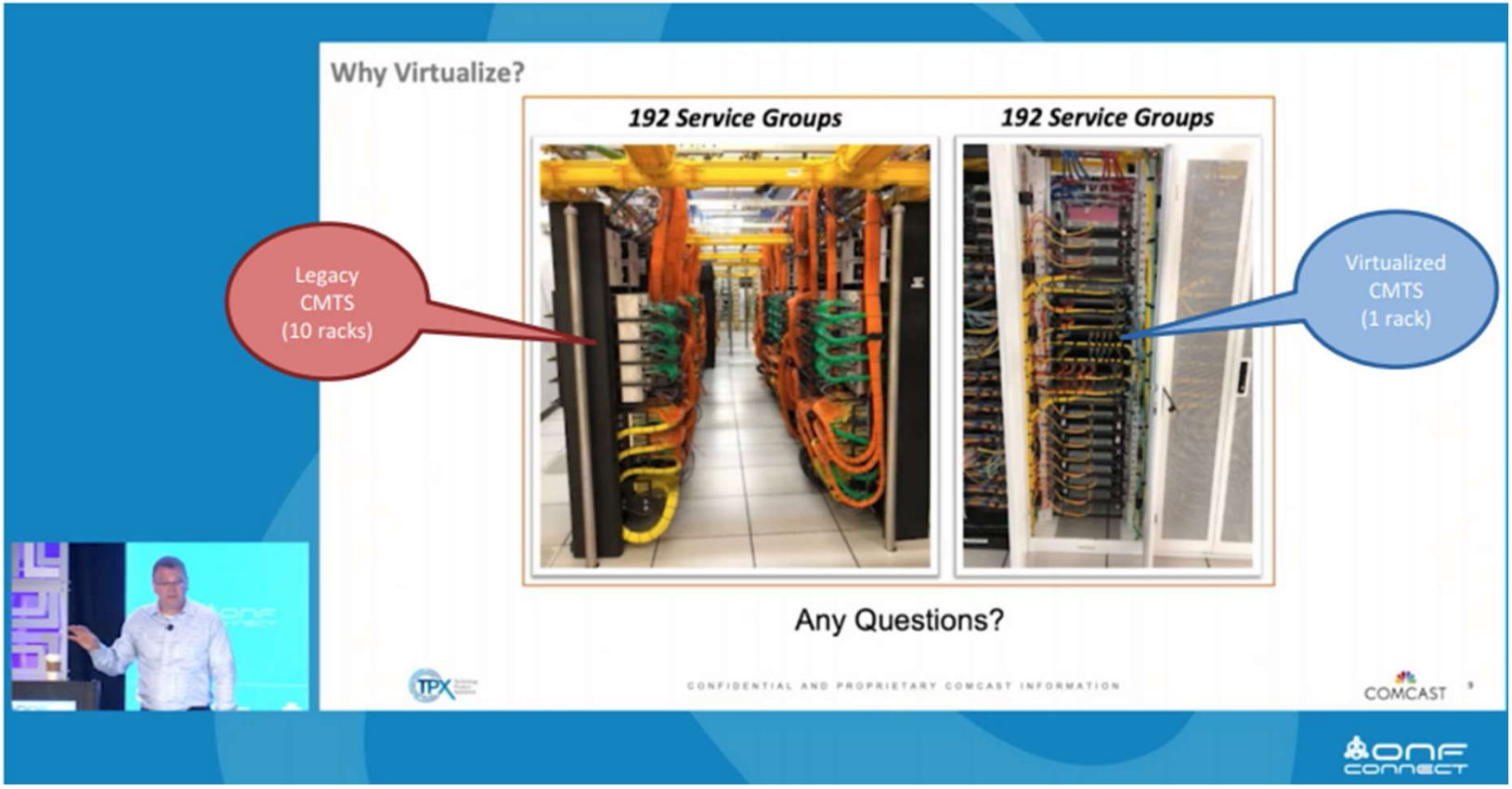
- **2012년 11월 ETSI는 ISG 'NFV' 설립**
- **NFV(Network Functions Virtualization): 하드웨어 의존성을 최소화 하는 네트워크 가상화로 독립적이며 유연하고 단순한 네트워크를 만드는 도구 제공**



European Telecommunications Standards Institute **(ETSI)** Industry specification group **(ISG)**

- **Why Virtualize?** (왜 가상화를 했나?)

- **비용절감, 자동화, 신속한 적용, …**
- **예: Cable Modem Termination System** (CMTS)

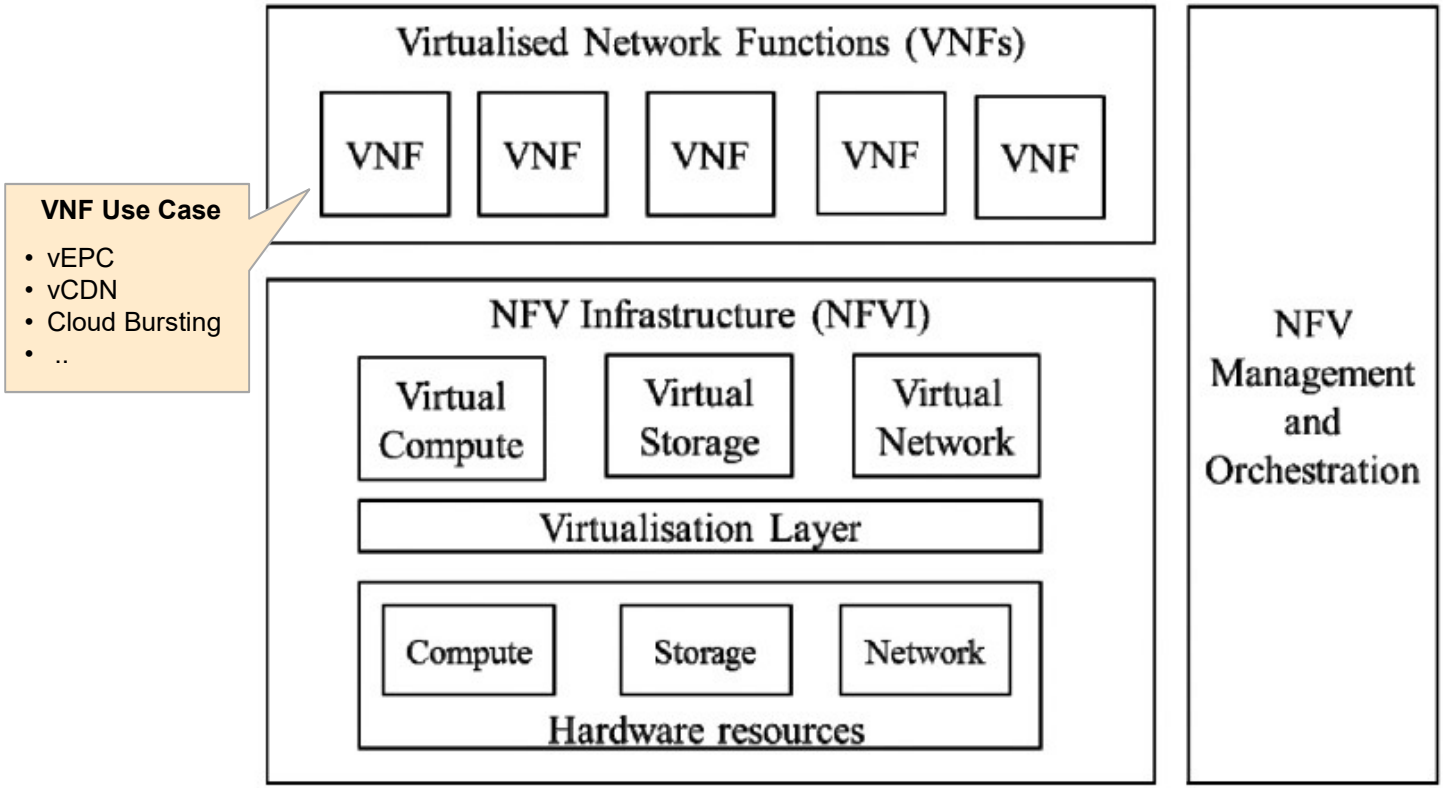


james@jslab.kr

▪ NFV 아키텍처

◻ NFV 아키텍처

- 2019-2020 NFV Release 4 (https://www.etsi.org/technologies/nfv) : 2년 주기
- NFV 관련 스펙 (https://www.etsi.org/committee/nfv): VNF, NFVI, NFV-MANO,...



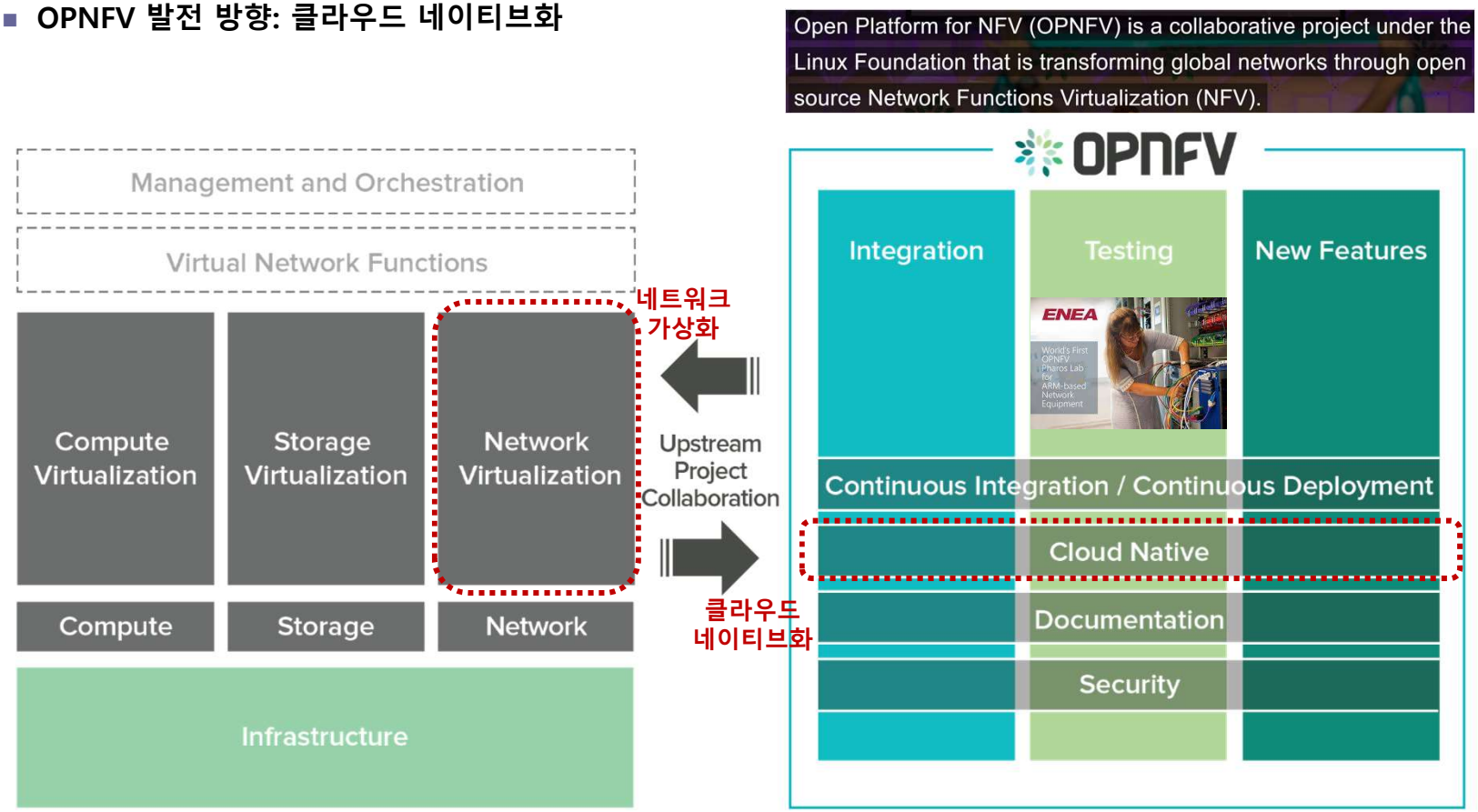
https://www.sdxcentral.com/networking/nfv/definitions/what-is-nfv-orchestration/

james@jslab.kr

▪ OPNFV

◻ **OPNFV** (**O**pen **P**latform for **NFV**): **2014년 9월 30일 결성(리눅스 재단), 오픈소스 NFV 협력 프로젝트**

- **홈페이지: https://www.opnfv.org/about**
- **OPNFV 발전 방향: 클라우드 네이티브화**



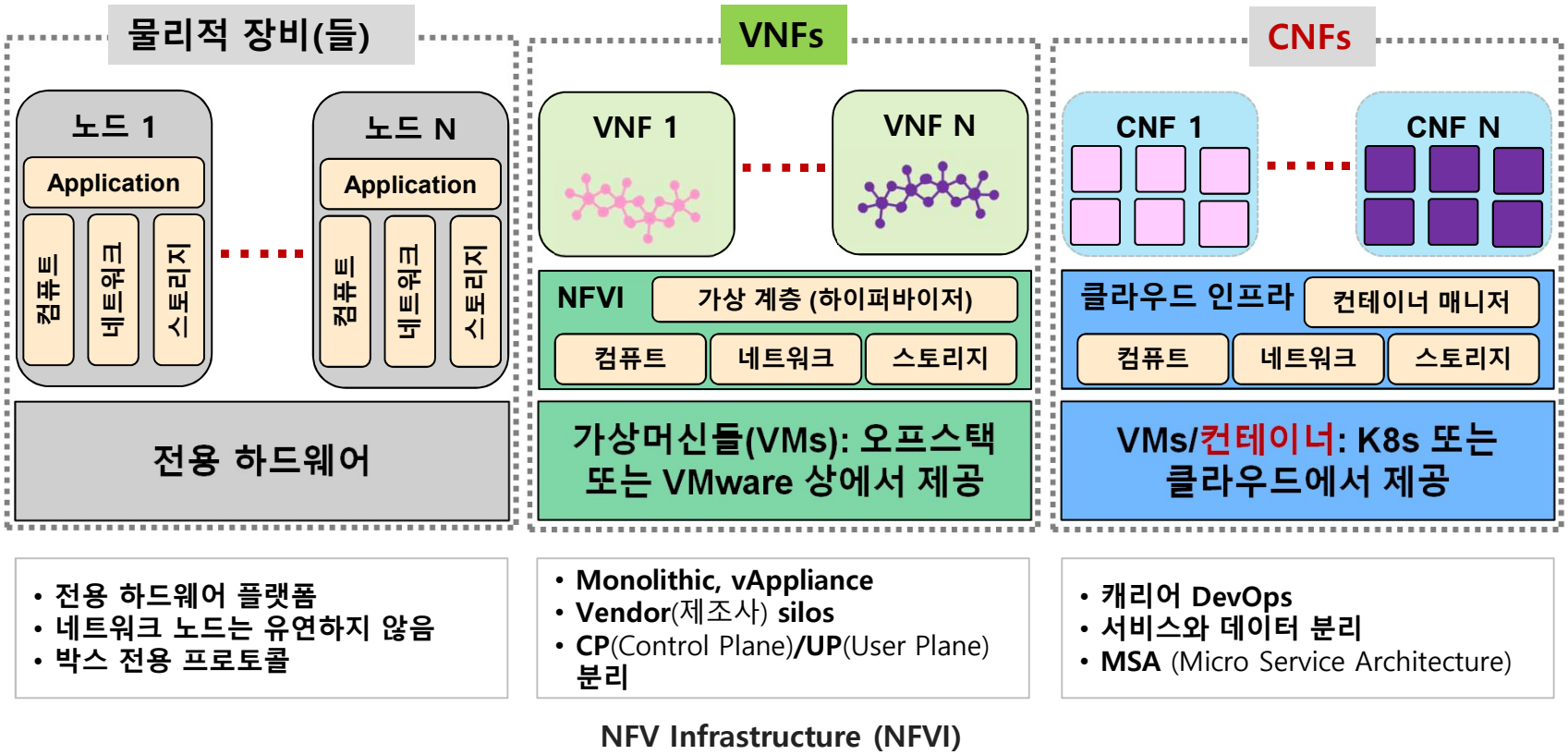
▪ 클라우드 네이티브(Cloud Native)와 네트워킹

- 클라우드 네이티브(Cloud Native)를 클라우드 네이티브 컴퓨팅 재단(CNCF)에서 정의
- **CNCF**(Cloud Native Computing Foundation)는 오픈소스와 제조사 중립 프로젝트의 생태계를 육성하고 유지하기 위한 리눅스 재단에서 관리
- 주요 글로벌 서비스 회사와 제조사 등이 CNCF에 참여 하는 등 수백 이상의 멤버가 활동
- 클라우드 네이티브(Cloud Native)는 자동화와 결합하여 엔지니어가 최소한의 노력으로 빈번하고 예측 가능한 변경 작업을 수행 할 수 있는 기술과 쉬운 관리 뛰어난 복원력을 제공 할 수 있음
- 클라우드 네이티브(Cloud Native)는 클라우드 고유 기술을 사용하며 동적 환경에서 구축하고 실행하여 서비스를 확장하기 위해 **컨테이너**, 서비스 메쉬, 마이크로서비스, 비가역적 인프라(Immutable Infrastructure) 및 선언적 API를 사용하는 접근 방식 사용
- CNCF에서 진행중인 네트워크 관련 프로젝트는 SDN 기술을 활용하는 **CNI**(Container Network Project), **CoreDNS**(서비스 디스커버리), **gRPC**(gRPC Remote Procedure Calls), **Linkerd**(서비스 메시), **Envoy**(서비스 메시) 등이 있음

james@jslab.kr

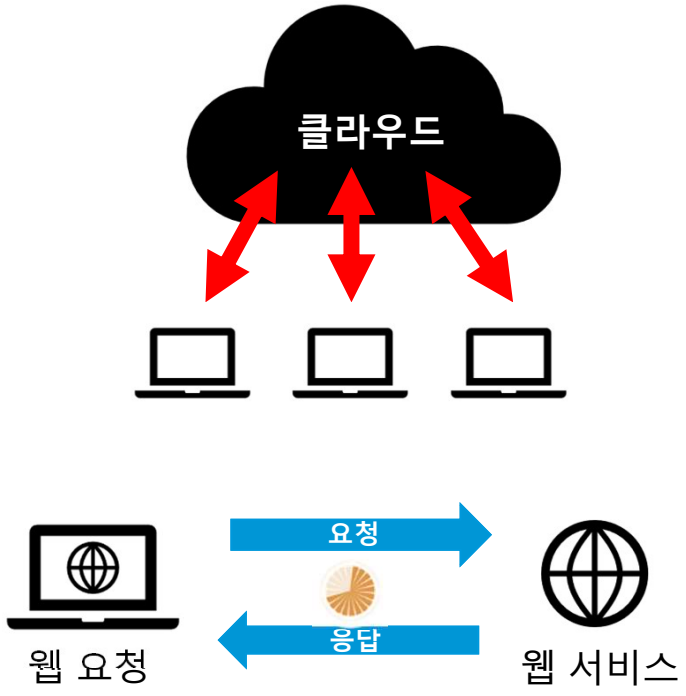
▪ 네트워크 (통신) 기반 기술 발전

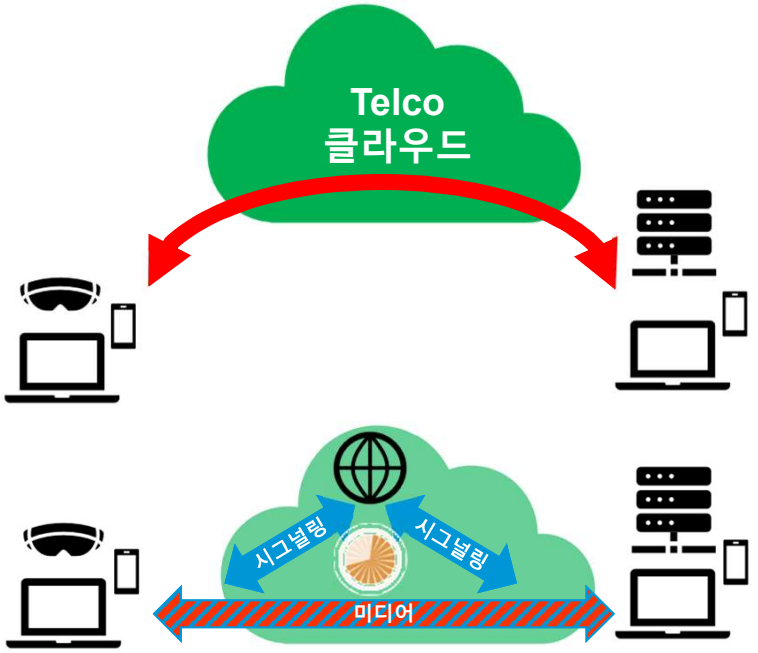
▫ 물리적 장비(들) → **VNFs** (Virtual Network Functions) → **CNFs** (Cloud Native Functions)



▪ Public vs. Telco 클라우드

- Public Cloud: 소프트웨어 정의 가상 인프라 기반 서비스 (웹서비스)
- Telco Cloud: 언더레이 인프라 기반 클라우드 서비스 (전송경로 제공)
- Telco Cloud는 하드웨어 인프라 환경 고려



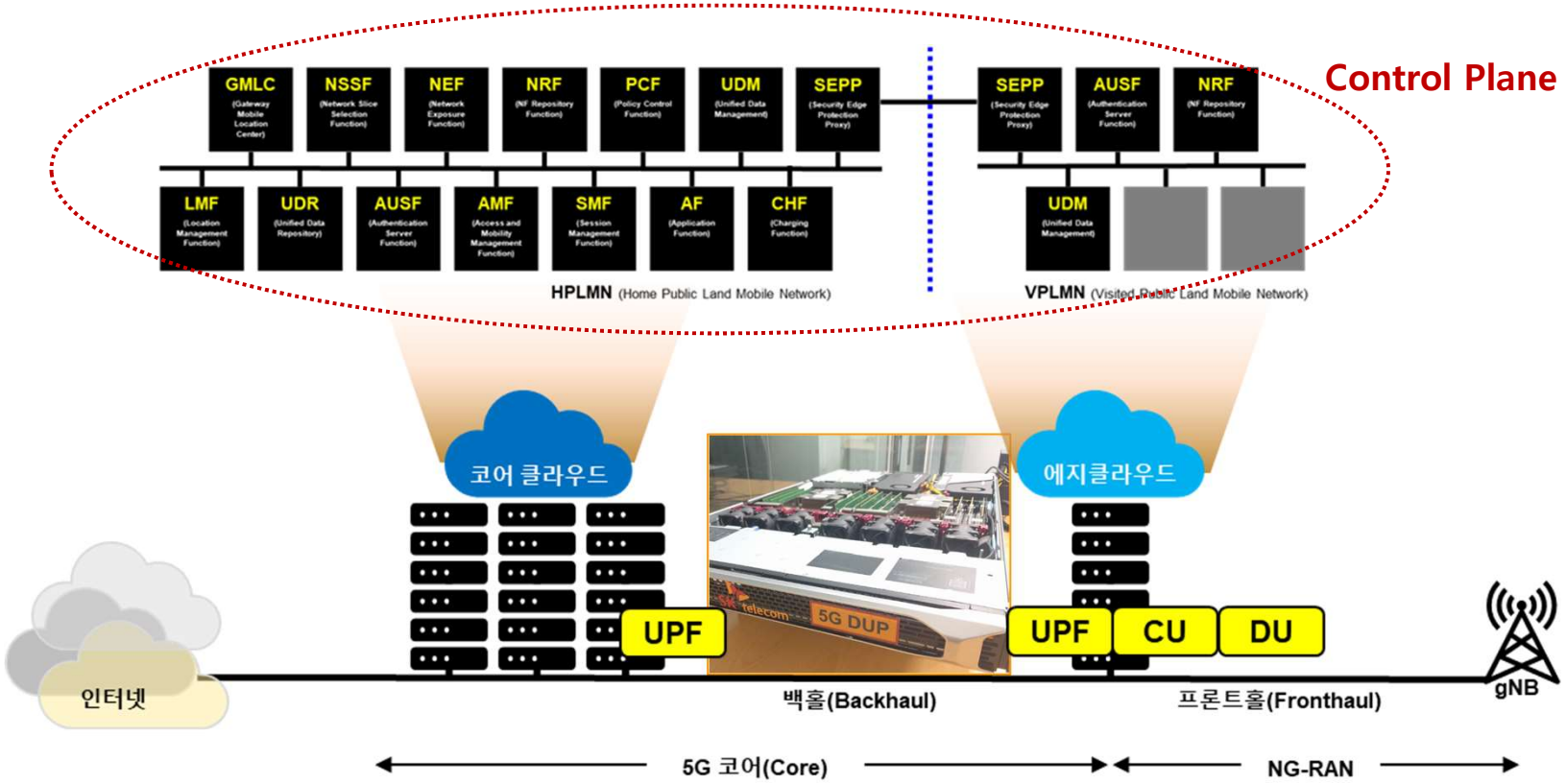


실시간 멀티미디어 애플리케이션
(고화질 스트리밍, 고화질 게임, AR, VR 등을 위한 경로)

james@jslab.kr

▪ 5G의 CNF 수용 (예)

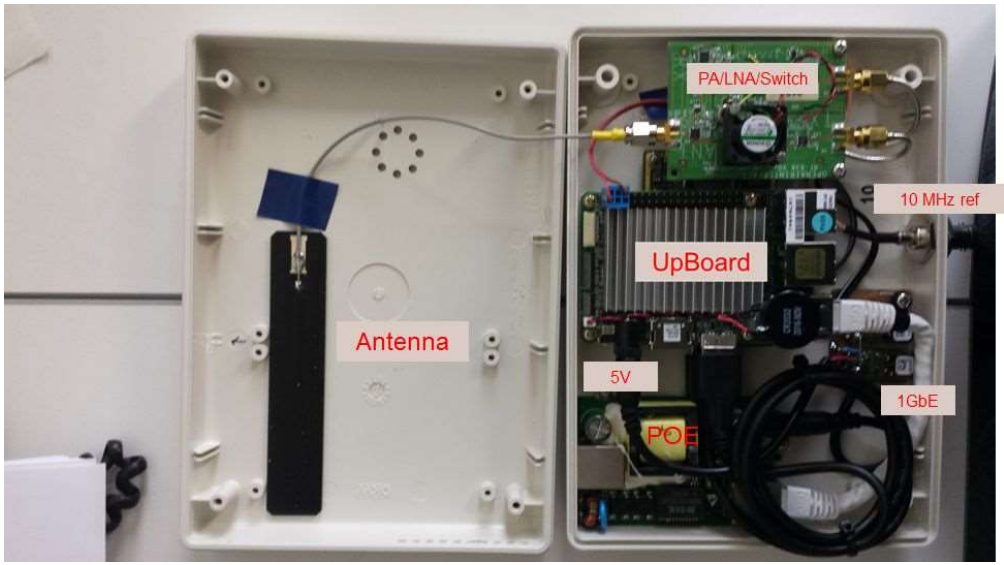
- **CP**(Control Plane)**/UP**(User Plane) **분리**
- **5G 콘트롤플레인**(Control Plane)**의 클라우드네이티브**(Cloud Native)**화**
- **5G 데이터플레인 UPF**(User Plane Function)



▪ 프로토타입 기지국 무선장비 제작

- Kubernetes / 오픈소스 사용 Low-End Prototyping Hardware
- 하드웨어 구매 리스트 (Shopping List):
  - USRP B200-mini ($500) - up to 50 MHz BW
  - custom 20 dBm PA/LNA/Switch ($300) - band 38, 42/43, n38/n77-78
  - Upboard/Upboard2 (low-end $90 PC)
  - GbE fronthaul POE+
  - Antenna
  - optional GPSDO

출처: Red Hat

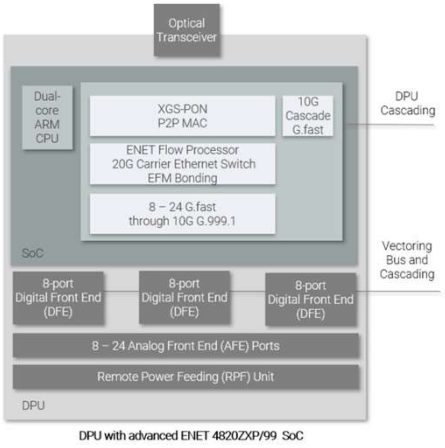


james@jslab.kr

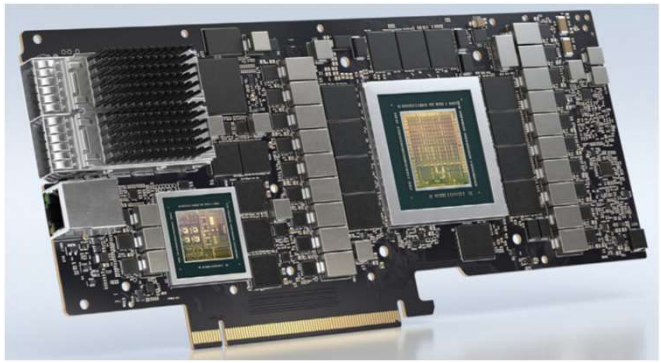
▪ 클라우드 네트워킹 가속 기능의 발전

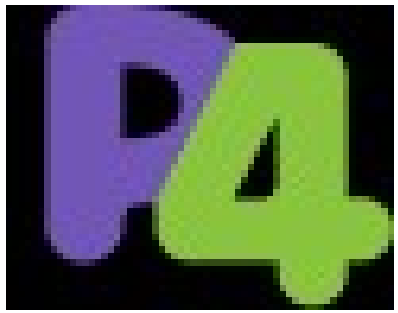
◻ 하드웨어 기반 네트워킹 가속: FPGA, DPU, P4

- OVS(open virtual switch)를 구현하기 위한 데이터 패킷 파싱·매칭·조작
- ZTR(Zero Touch RoCE)를 위한 RDMA 데이터 전송 가속
- CPU를 우회하고 네트워크로 연결된 데이터를 GPU로 직접 전송하는 GPU-다이렉트 가속
- RSS, LRO, 체크섬(checksum) 등을 포함한 TCP 가속
- VXLAN와 Geneve 오버레이 그리고 VTEP 오프로드를 위한 네트워크 가상화
- 5G를 위한 5T와 같은 텔코 클라우드(telco Cloud) RAN 용 정밀 타이밍 가속기
- IPSEC 및 TLS에 대한 암호화 가속이 인라인으로 수행되므로 다른 모든 가속이 계속 작동
- SR-IOV, VirtIO, 반가상화를 위한 가상화 지원
- 신뢰할 수 있는 루트, 안전한 부팅, 안전한 펌웨어 업그레이드, 인증된 컨테이너·애플리케이션 수명주기 관리



DPU with advanced ENET 4820ZXP/99 SoC

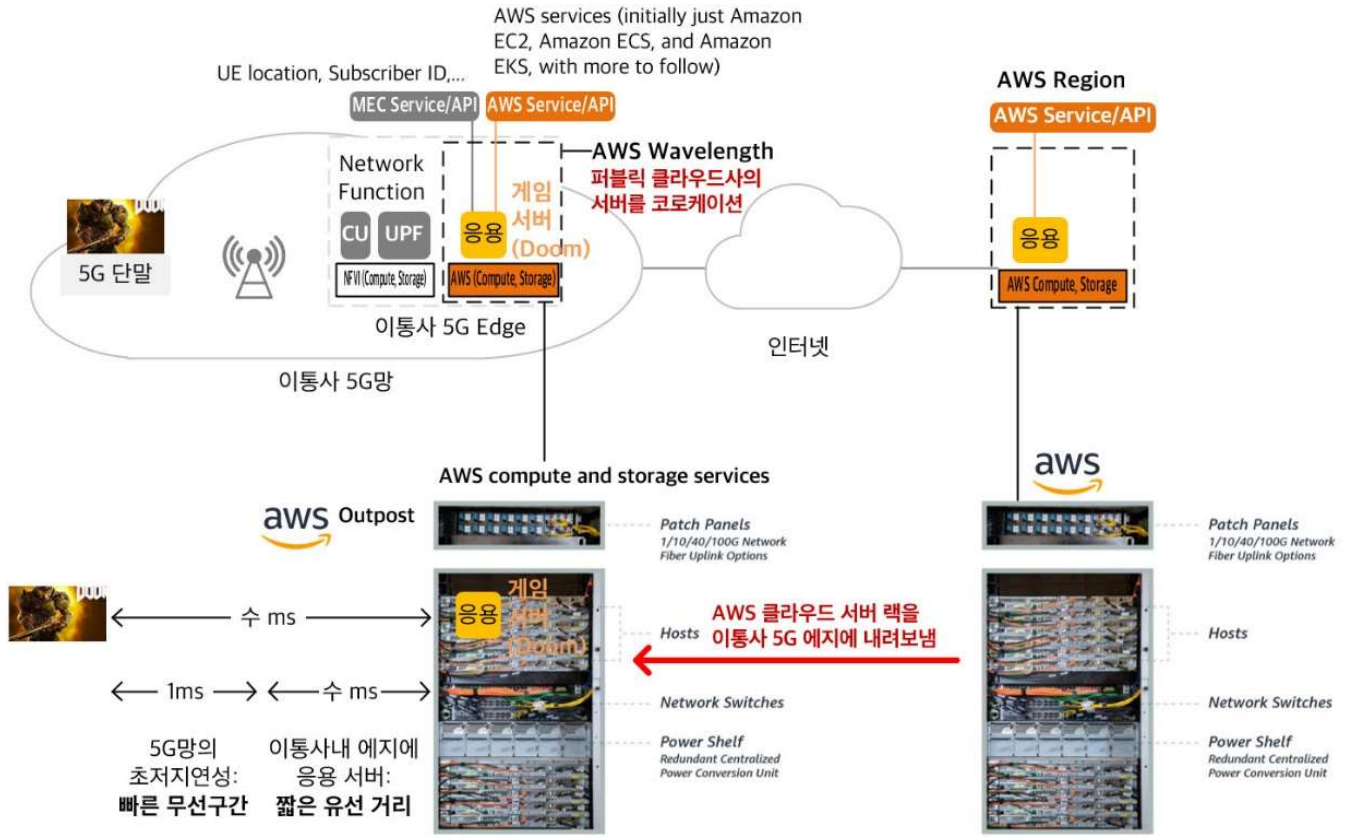

https://www.nvidia.com/ko-kr/


https://p4.org/code/

▪ 5G Edge 'Amazon AWS Wavelength'

◻ Amazon AWS Wavelength 구성

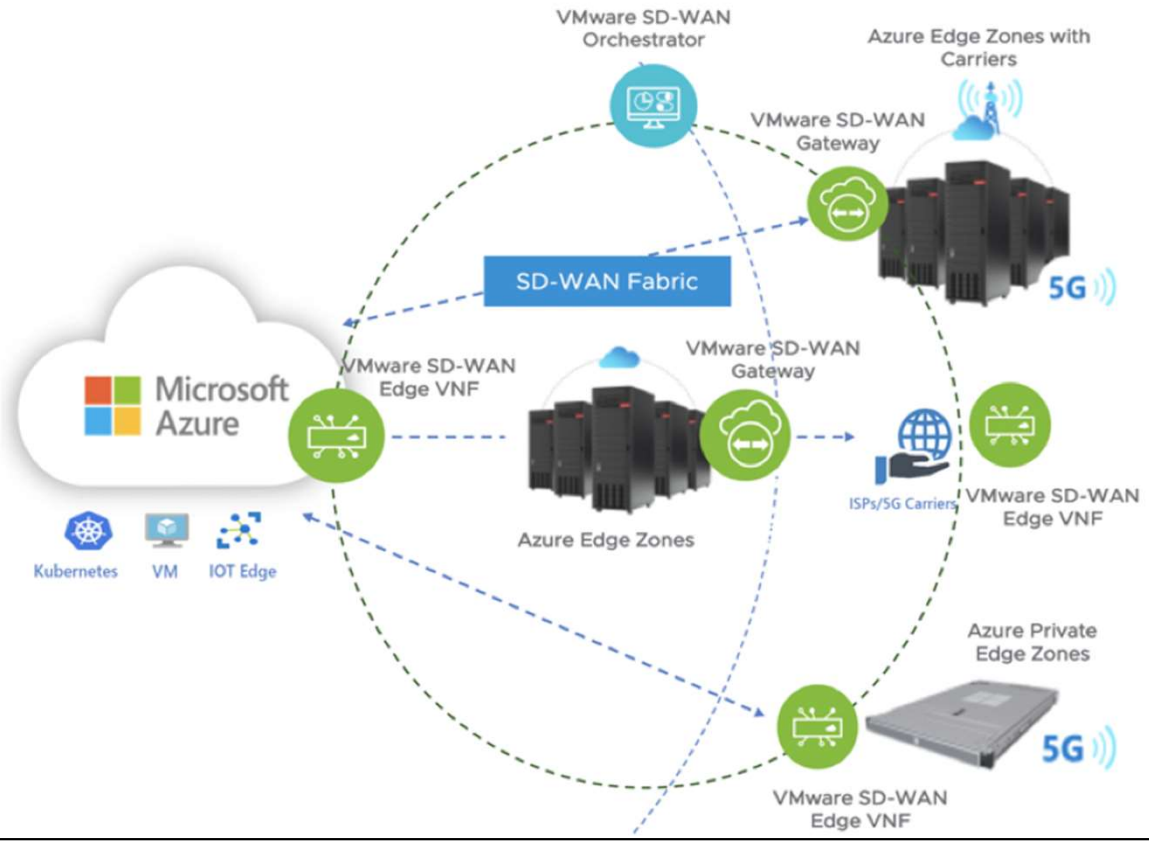
■ 출처: https://www.netmanias.com/ko/post/blog/14591/5g-edge-iot-mec/aws-wavelength



▪ 5G Edge 'Microsoft Azure Edge'

◻ Microsoft Azure Edge Zones: 5G 통신사와 연결하여 기업에 설치하는 사설망 구성

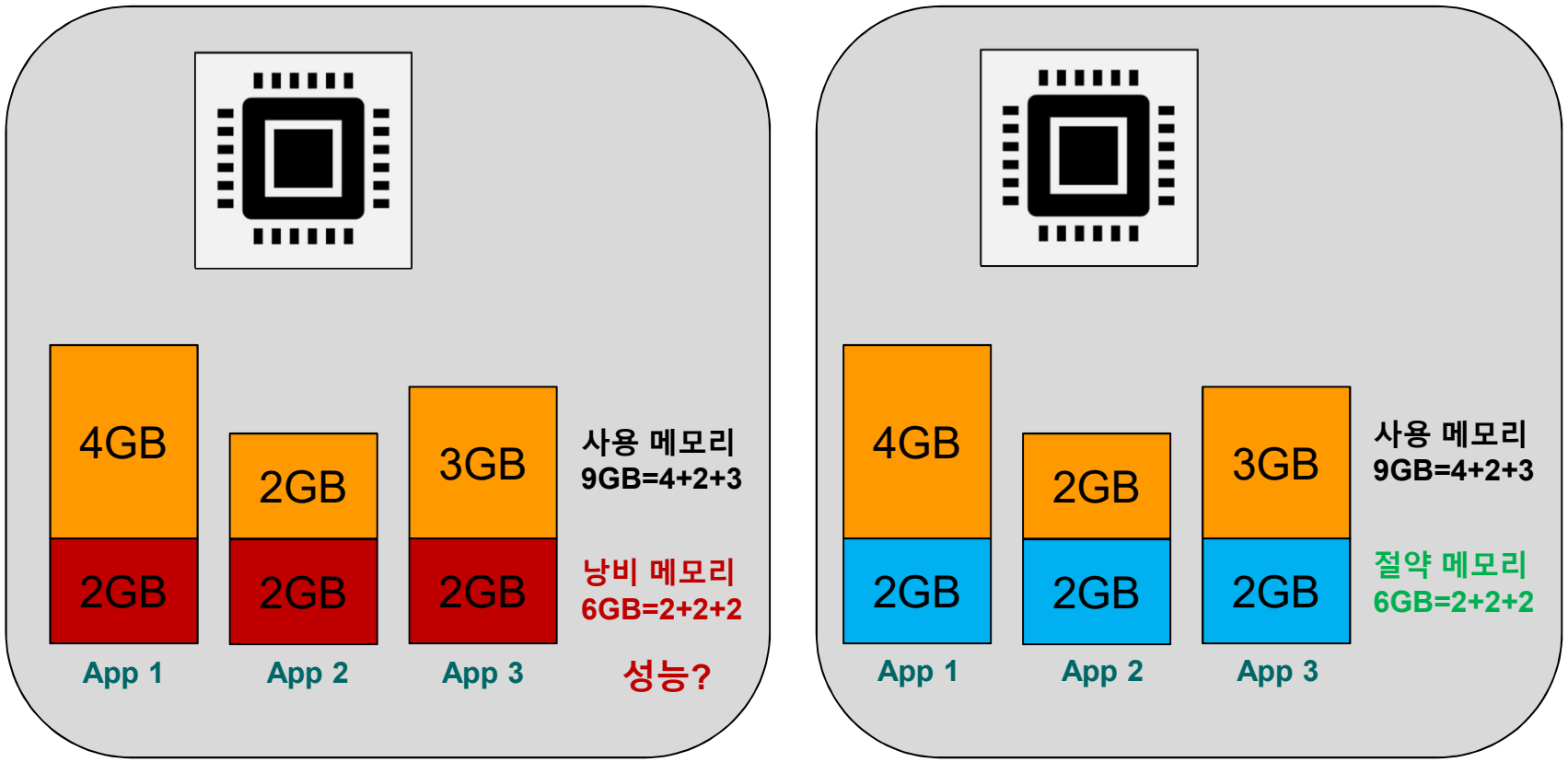
■ 출처: https://blogs.vmware.com/velocloud/2020/03/31/vmware-to-deliver-networking-solutions-with-azure-edge-zones/



james@jslab.kr

▪ 가상머신과 컨테이너 메모리 사용

◻ 클라우드를 위한 컨테이너



❑ Session 1: 컨테이너 인프라 개요

• 컨테이너 기술 특징

▪ 컨테이너와 쿠버네티스

◻ 클라우드를 위한 컨테이너

- 마이크로서비스를 격리
- 가상머신(VM)을 이용해서 마이크로서비스를 배포 할 때처럼 많은 오버헤드가 발생하지 않음
- 각각의 마이크로서비스에 가상머신(VM)을 할당하는 것은 비용이 많이 필요하여 컨테이너가 클라우드 배포에 이상적임

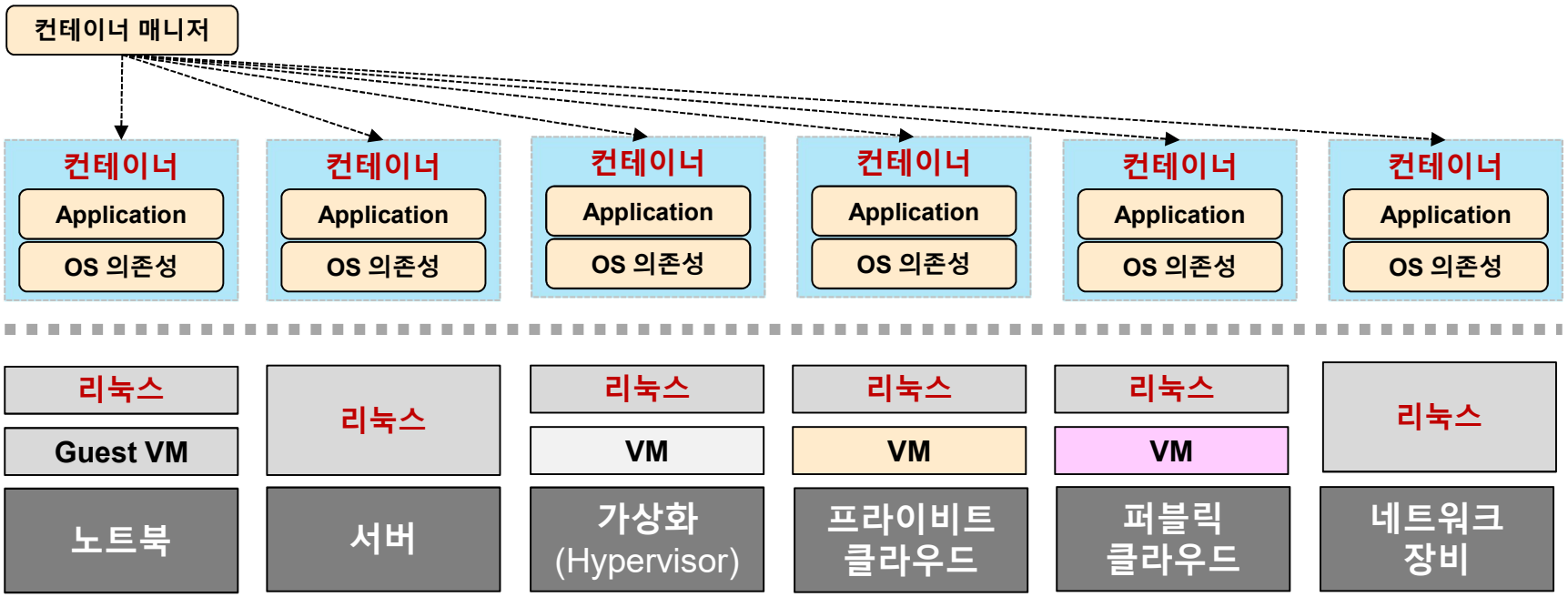
◻ 쿠버네티스(K8s)의 파드(Pod)

- 단일 컨테이너 내부에서 여러 애플리케이션 수행 시 다중 프로세스를 실행하는 슈퍼바이저 기능 필요하며, K8s의 작업 단위인 Pod는 이런 방법을 사용하지 않아도 Pod 내의 컨테이너들이 동일한 IP주소와 포트 공간을 가짐
- K8s의 작업 단위인 Pod 한 개 조는 여러 개의 컨테이너를 포함
- 동일한 시스템의 여러 Pod는 함께 스케줄링 됨
- Pod내 컨테이너들은 Local Host나 IPC(Inter-process Communication)을 사용하며 공유 로컬 저장소를 접근 가능
- K8s의 다음과 같은 특징으로 도커 등의 컨테이너 내부에 여러 Application을 실행 하지 않음
  - 투명성
  - 독립성
  - 편의성
  - 효율성

▪ 애플리케이션 이식성

◻ 컨테이너의 애플리케이션 이식성

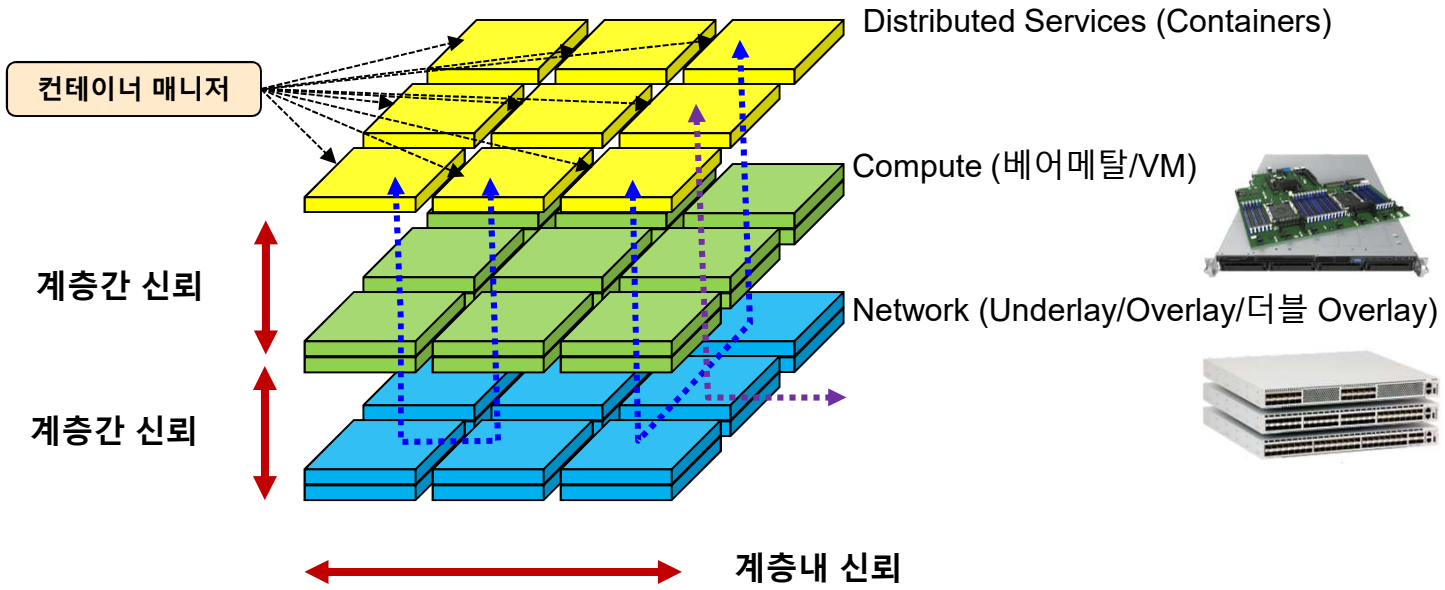
- 리눅스 커널을 사용하는 OS에 의존하는 계층의 추상화로 Application 이식성 제공
- 컨테이너는 가상머신(VM)을 이용해서 서비스를 배포 할 때처럼 많은 오버헤드가 발생하지 않음



▪ 계층별 네트워크

◻ 계층간 보안

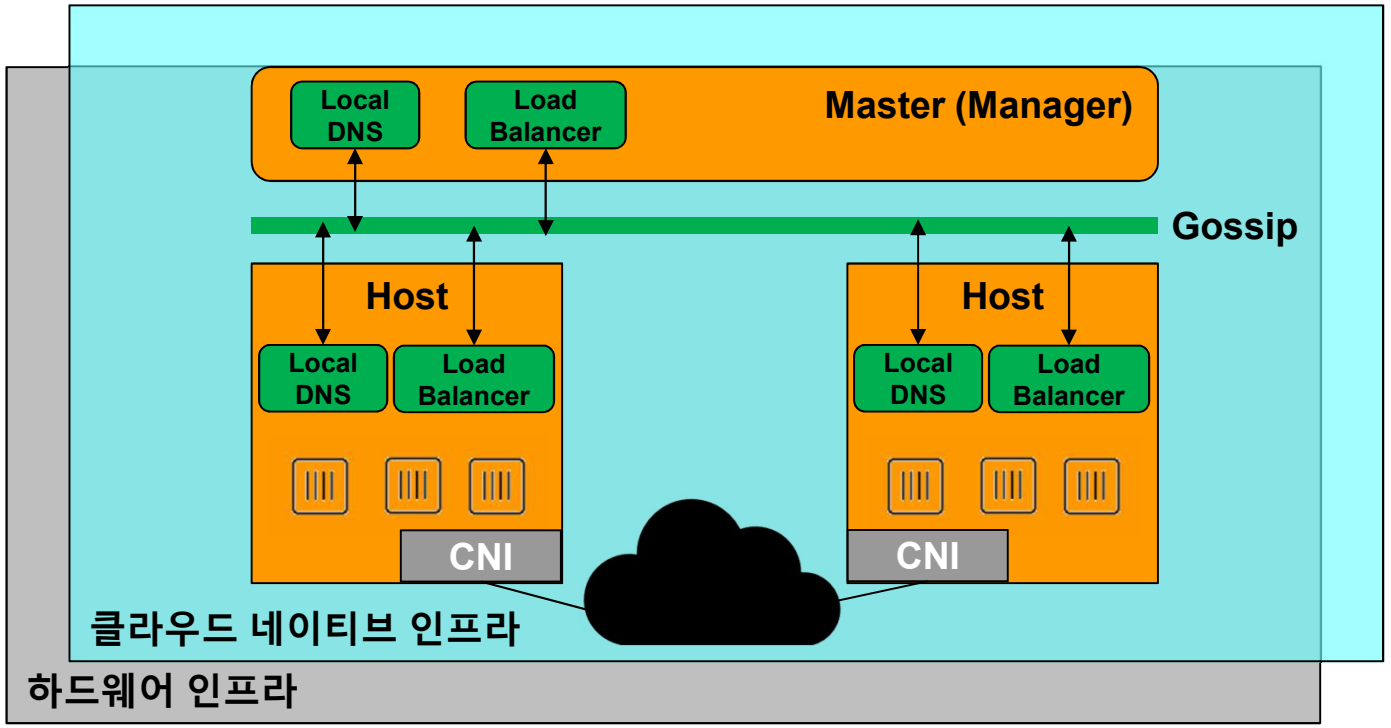
- 계층간 신뢰: 계층내 인증/터널링 연결하여 계층간 분리, 성능 이슈
- 계층내 신뢰: 논리적/물리적 분리로 계층 내의 폐쇄그룹 구성
- 경로의 신뢰성을 위한 서비스간 암호화 프로토콜 사용 (예): Transport Layer Security (TLS) Encryption
- 독립적인 오버레이 구성을 위한 프로토콜 사용 (예): VxLAN



▪ Container Networking

◻ Container Networking 'CNI' 와 Gossip

- Container Network Interface (CNI): CoreOS가 제안, K8s, Kurma, rkt, Apache Mesos, Cloud Foundry, Cisco Contiv, Project Calico, Weave, Docker, …
- Gossip: P2P, 네트워크 크기와 무관하게 동작, 일정한 개수의 불특정 노드에 정보를 gossiped, 지정한 오버레이를 통해 다른 노드에 알림



▪ 클라우드 네이티브 기반 서비스간 네트워크

◻ **gRPC**(gRPC Remote Procedure Calls)

- **gRPC is faster than REST. gRPC uses HTTP2 by default**
- https://devopedia.org/grpc?ref=codebldr

	Goal	REST (HTTP/JSON)	gRPC
COMPATIBILITY	Single source of truth	✗	✔
	Multi-platform + languages built in	✗	✔
	Handle non-breaking changes.	✗	✔
PERFORMANCE	Network: connection handling	Manual, 1 per call ✗	Built-in, Multi per conn ✔
	Speed: Transmission of data	Human-readable Text ✗	Binary ✔
	CPU: Improved resource usage	✗	✔
MAINTENANCE	Tracing	Manual ✗	Easy to plug in ✔
	Logging	Manual ✗	Easy to plug in ✔
	Monitoring	Manual ✗	Easy to plug in ✔

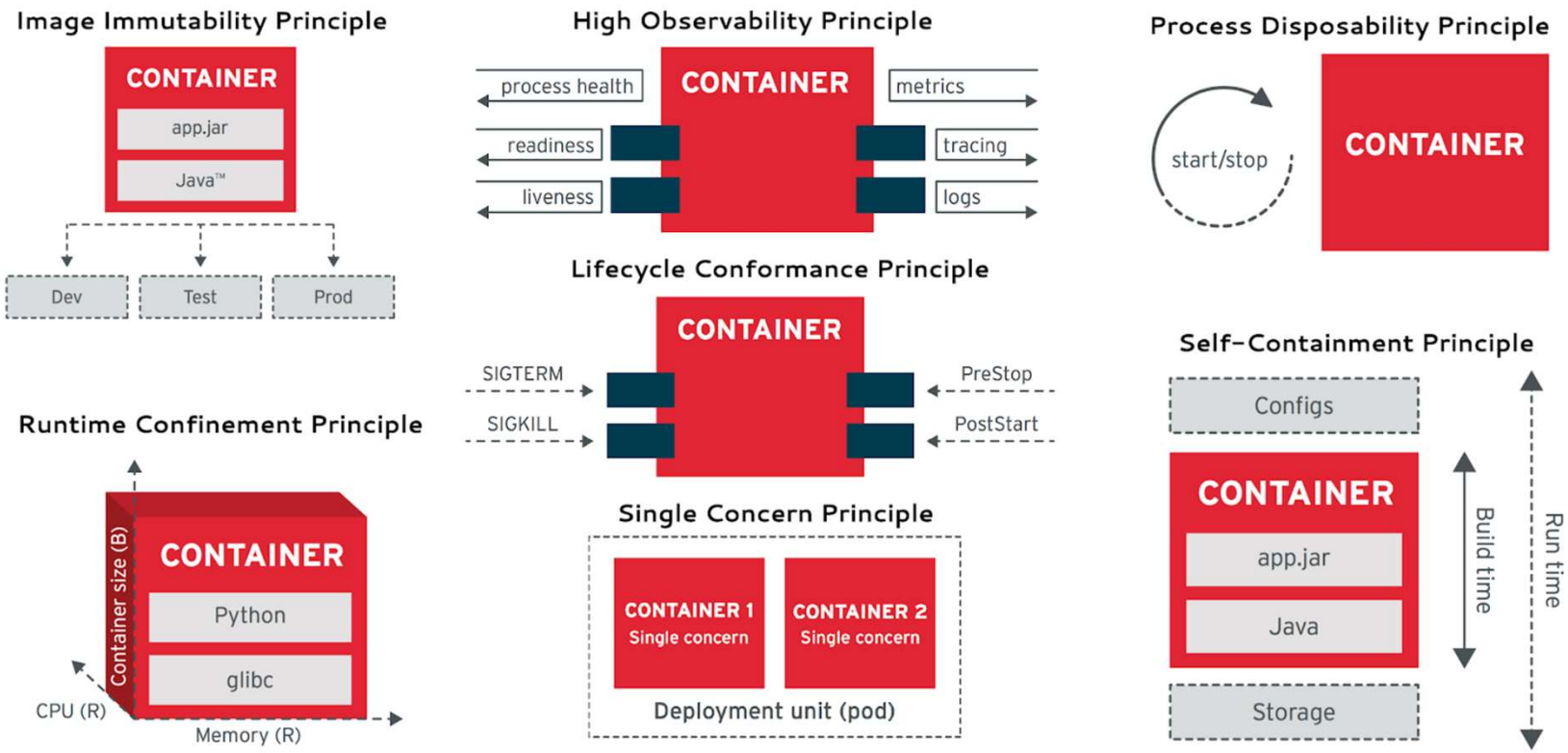
james@jslab.kr

james@jslab.kr

▪ Container-based Application

▫ Container-based Application Design

- https://kubernetes.io/blog/2018/03/principles-of-container-app--design/?fbclid=IwAR2oMrdP0d1Q6LXebtxNPnt-RS5DIlkCIwpaMSL5mmW7VMaQb6hRV8hkd38



james@jslab.kr

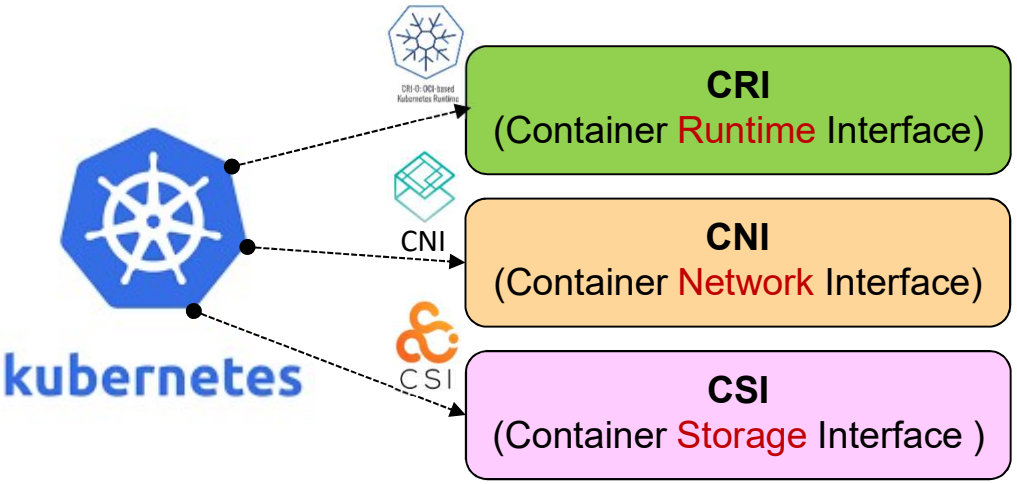
- Session 1: 컨테이너 인프라 개요
  - 컨테이너 기술 발전
  - 컨테이너 기술 특징
- **Session 2: 컨테이너 기반 인프라**
  - **컨테이너 사용 인프라 구성**
  - **컨테이너 종류별 기술 비교**
- Session 3: 컨테이너 응용 네트워크 기술
  - 클라우드 네이티브 (K8s CNI, Service Mesh)
  - 통신 인프라 환경 컨테이너 기반 서비스

▪ 쿠버네티스 오케스트레이션

◻ 컨테이너 오케스트레이션을 위한 오픈 플랫폼

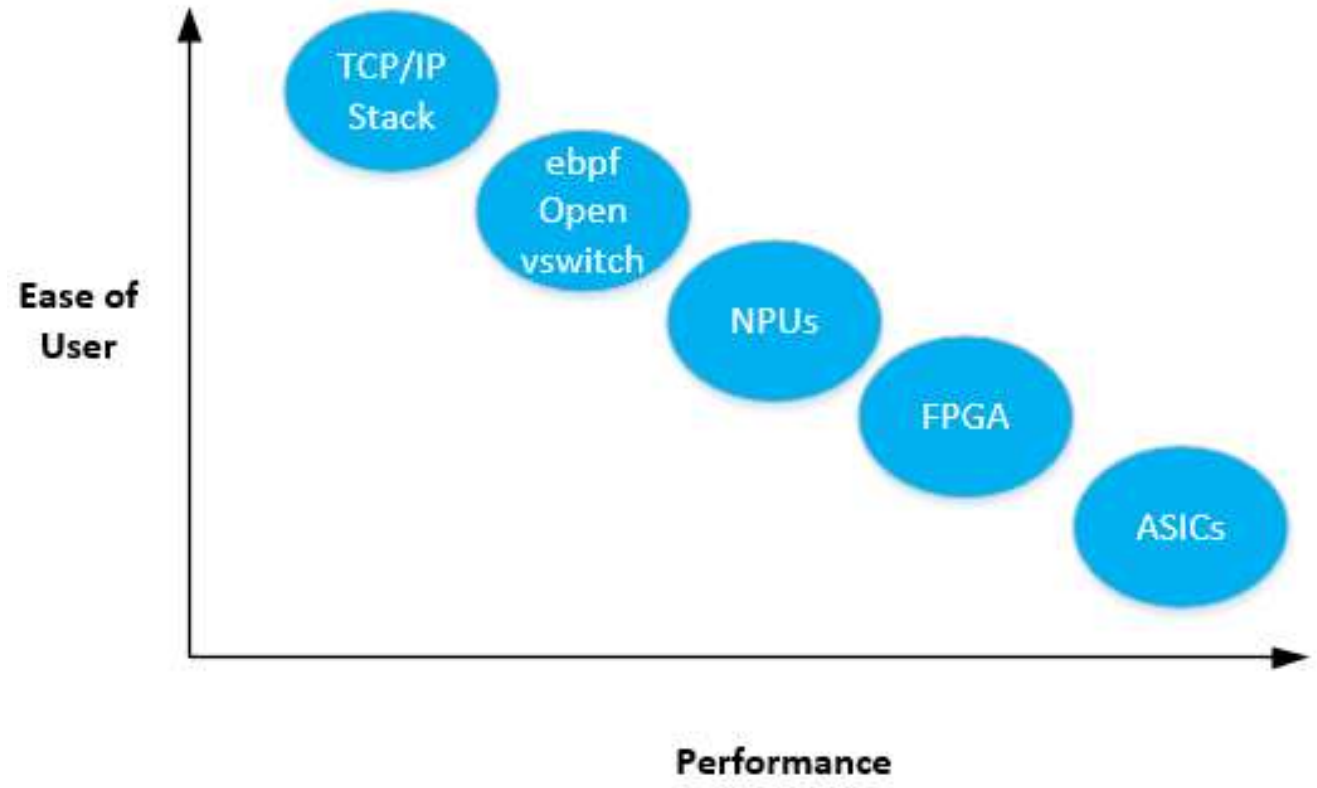
- **CRI** (Container Runtime Interface)
- **CNI** (Container Network Interface)
- **CSI** (Container Storage Interface)

```yaml
# B2C Accelerator Compose
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: hybrisacc
spec:
  replicas: 5
  selector:
    matchLabels:
      app: hybrisacc
  template:
    spec:
      containers:
      - name: hybrisacc
        image: 909204795480.dkr.ecr.us-east-1.amazonaws.com/hybris
        args: ["accstorefront"]
        readinessProbe:
          httpGet:
            path: /yacceleratorstorefront/?site=electronics
            port: 8088
            scheme: HTTPS
          initialDelaySeconds: 180
          timeoutSeconds: 1
          periodSeconds: 15
        env:
        - name: Example_Env
          value: "dynamic"
        volumeMounts:
        - mountPath: /etc/ssl/certs/hybris
          name: secrets
        - mountPath: /opt/hybris/data/media
          name: media
        ports:
        - containerPort: 8088
      volumes:
      - name: secrets
        hostPath:
          path: /efs/resources/secrets
      - name: media
        hostPath:
          path: /efs/state/media
      imagePullSecrets:
      - name: awsecr
```



Container Networking

Flexibility vs Performance in Different Platforms

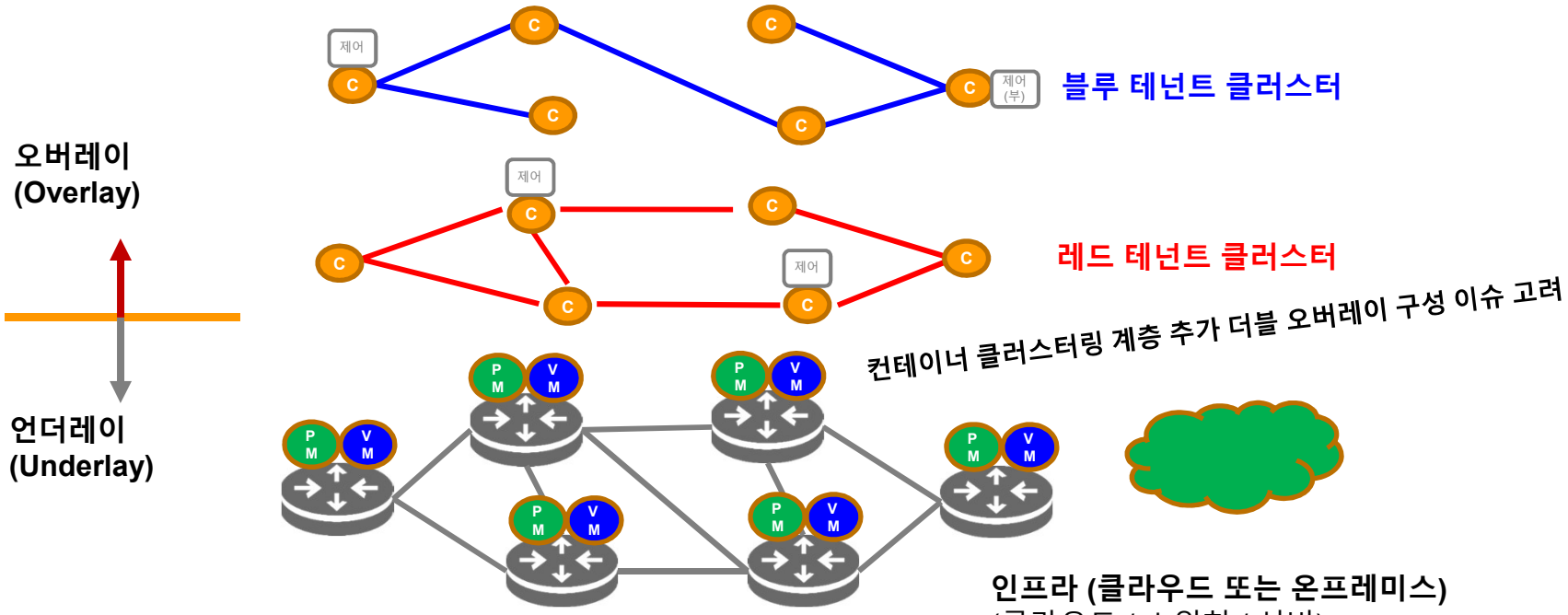


james@jslab.kr

▪ 계층별 네트워크

◻ 언더레이/오버레이 네트워크

- 오버레이(Overlay): 가상화 기반 네트워크 계층
- 언더레이(Underlay): 하드웨어 기반 네트워크 계층

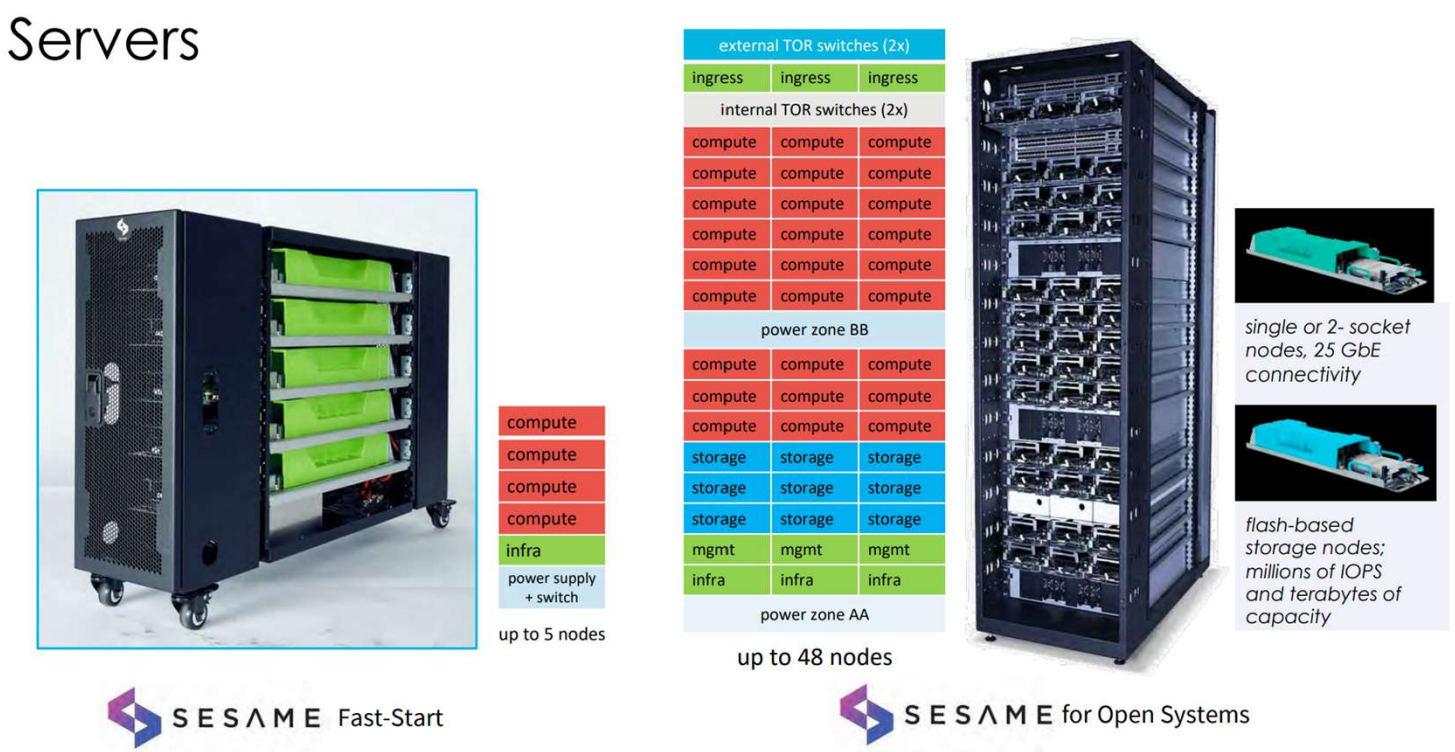


james@jslab.kr

▪ 쿠버네티스 지원 하드웨어

◻ 쿠버네티스를 위한 오픈 하드웨어 구성(예)

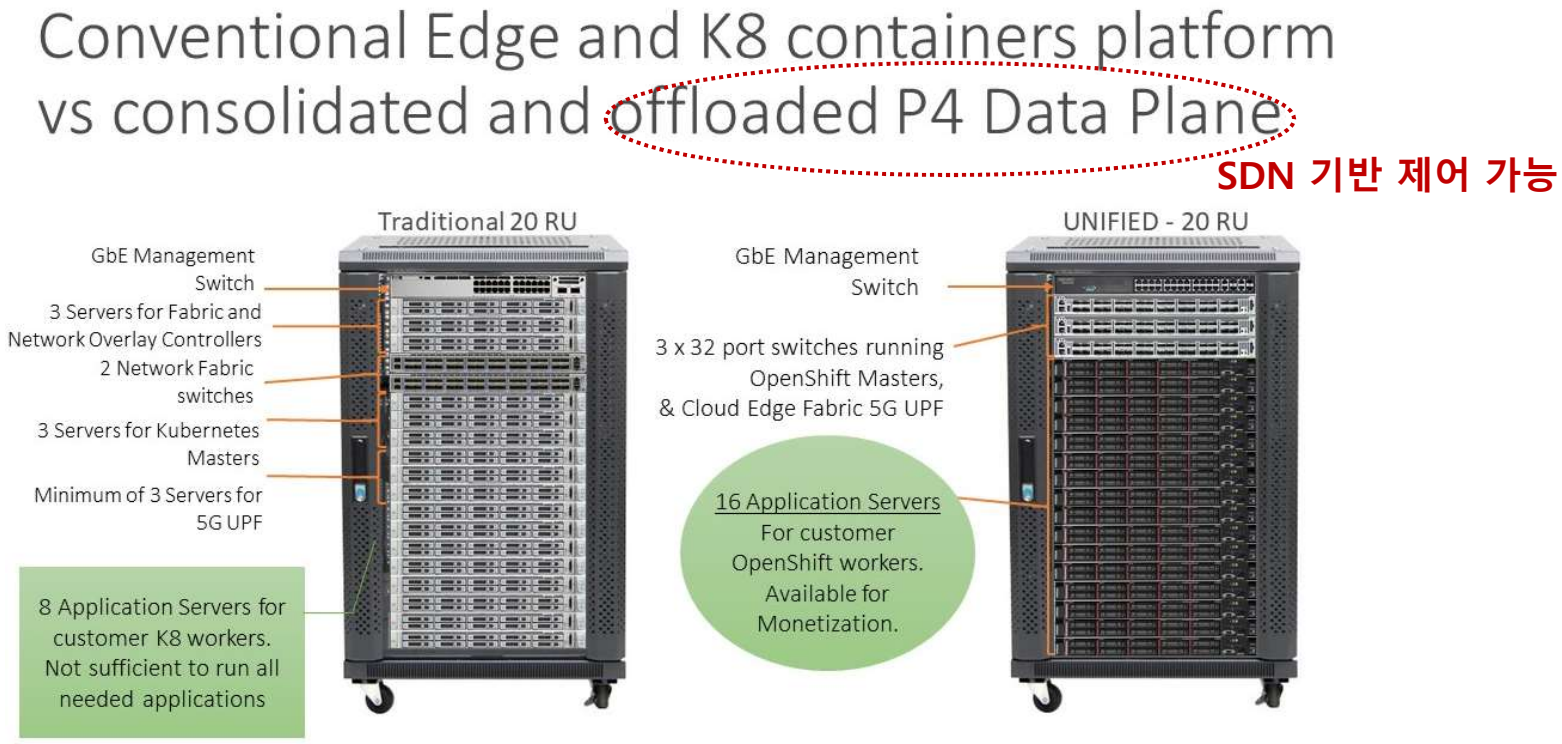
- Erik Riedel, PhD, Senior VP, Engineering | ITRenew



▪ 쿠버네티스 지원 하드웨어

◻ 쿠버네티스 데이터플레인 가속 하드웨어 구성(예)

- **Kaloom** (https://www.kaloom.com/)
- https://www.thefastmode.com/expert-opinion/17835-5g-drives-the-distributed-edge?fbclid=IwAR3DX6Vyi1BqH6SMyc8DnhGE_Zu0St4HxGjJ1qrvr8Qug72IWdEmJTo0x9Q



james@jslab.kr

▪ P4의 하드웨어 가속 지원

◻ P4: 데이터플레인 가속 지원 가능 언어

- 프로토콜 독립적 (Protocol Independent)
- P4는 소프트웨어 스위치에 지정하는 기능을 ASIC 수준 고속 처리
- 필드 업그레이드 가능(Programmable hardware)

Barefoot 'Tofino' Chip

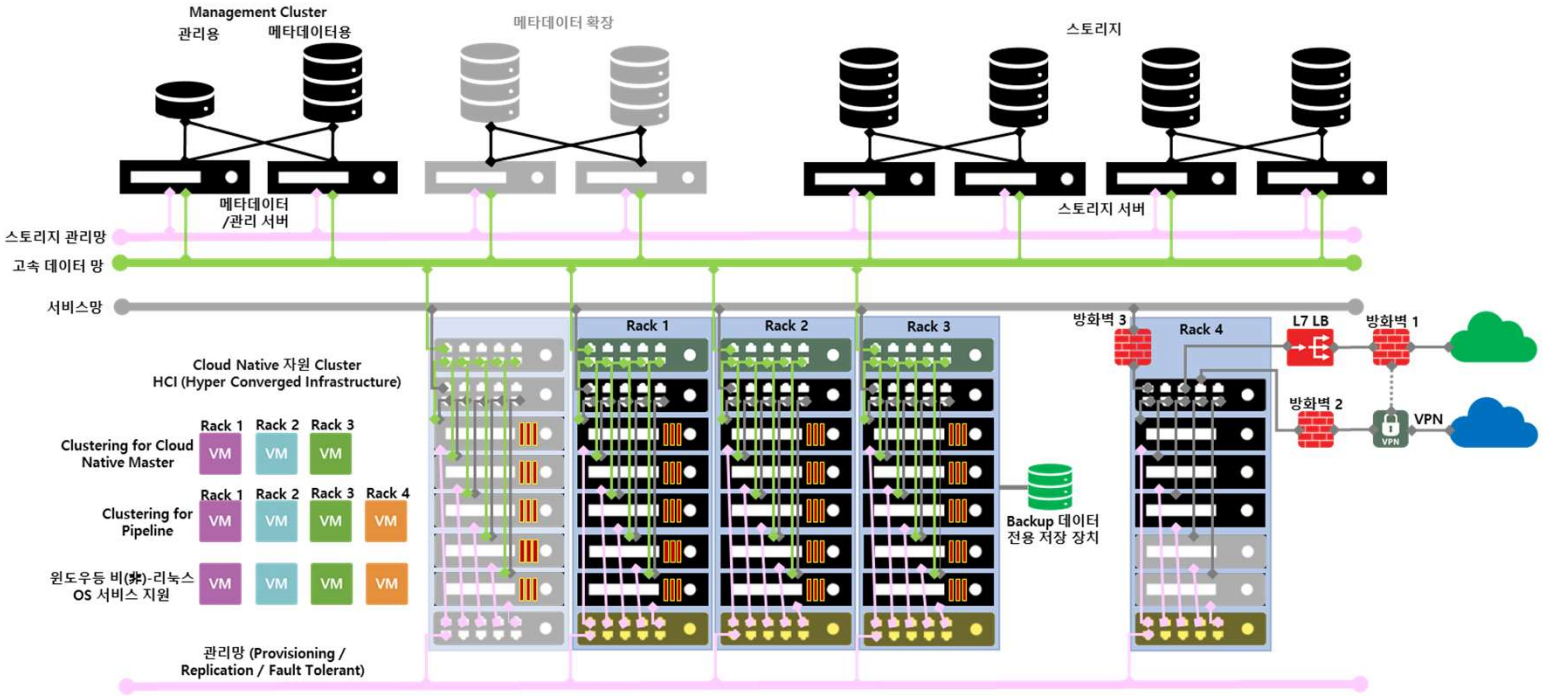
Pensando Systems 'DSC'
(Distributed Services Card)

james@jslab.kr

▪ 배포의 네트워크 경로 고려

◻ Cloud Native 기반 네트워킹 구성(예)

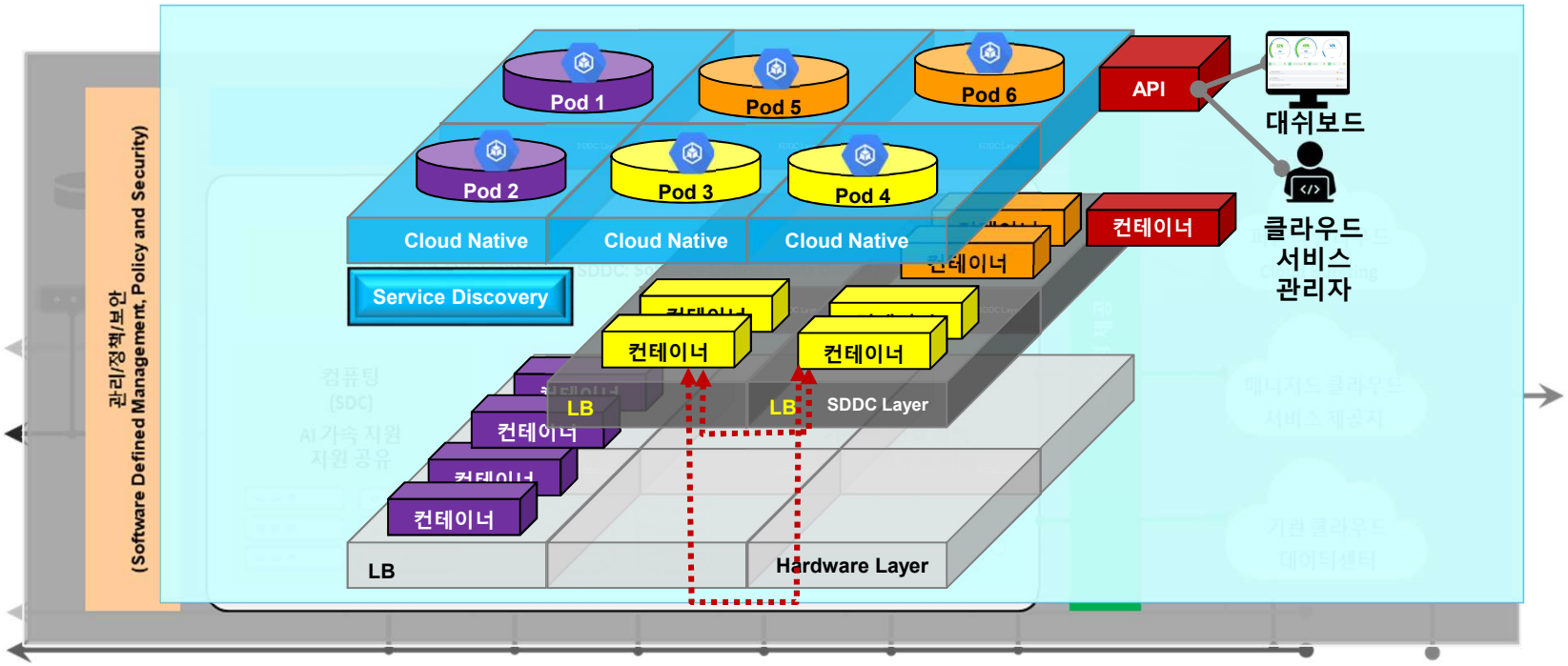
- **Kubernetes**는 자원 배포 단위로 컨테이너로 구성하는 **Pod** 사용
- 데이터 무한 증가에도 스토리지 네트워크 성능 저하 없는 구조
- 다이나믹한 컨테이너의 이동에도 DB 접속 성능 저하 없는 구조(예): **MSA**(Micro Service Architecture) / **HCI**(Hyper Converged Infrastructure)



▪ 컨테이너 배포의 네트워크 경로 고려

◻ Cloud Native 기반 네트워킹

- **Kubernetes는 자원 배포 단위로 Pod를 사용하여 추상화**
- **Pod는 Container로 구성**
- **오토스케일**(Auto-scale) **가능 서비스를 위한 LB** (Load Balancer) **내장**
- **쉬운 관리와 보안 강화를 위한 오버레이 구성을 위한 터널링 자원 사용** (VxLAN 패킷 헤더 추가 고려)

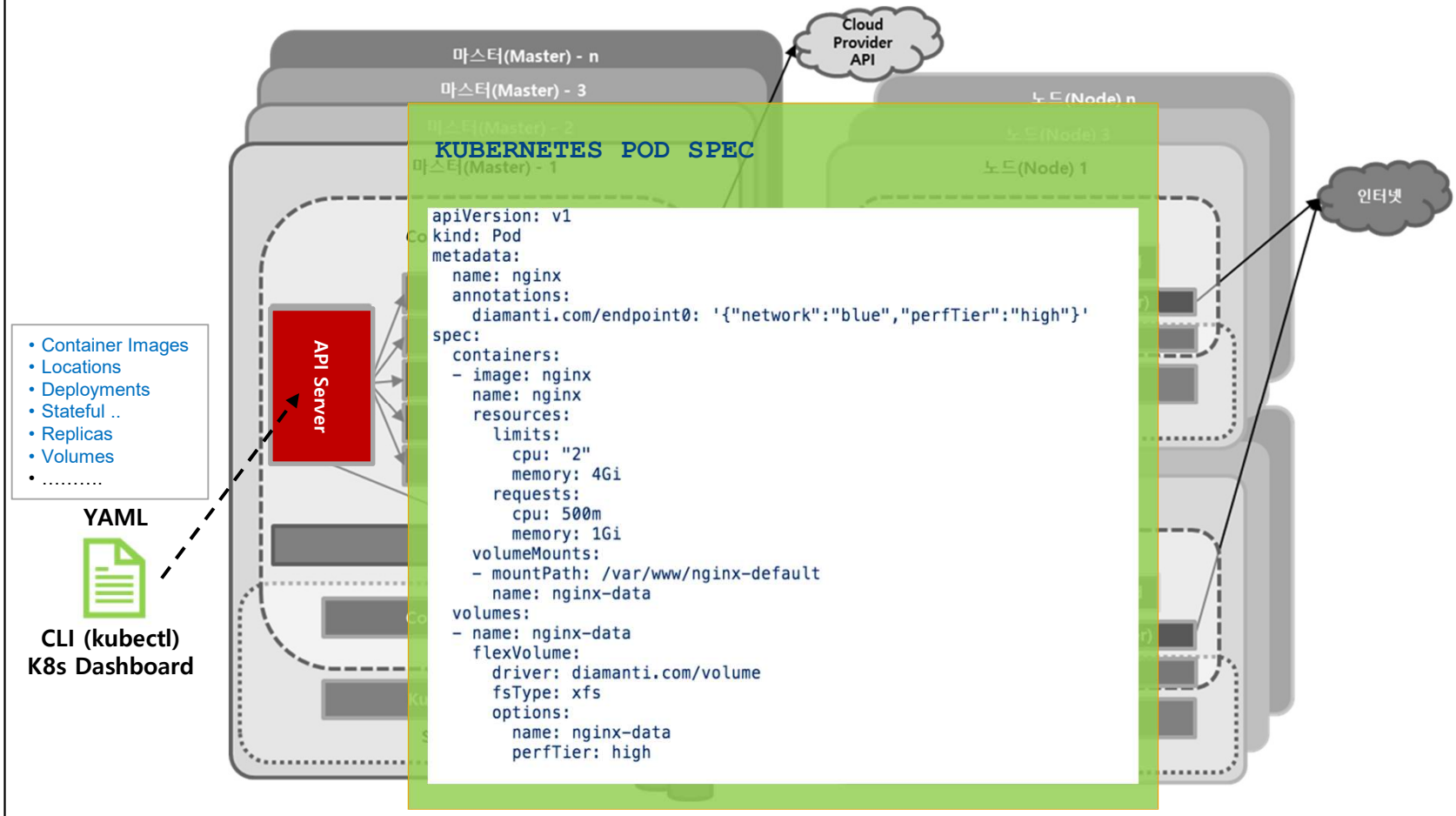


❑ Session 2: 컨테이너 기반 인프라

• 컨테이너 사용 인프라 구성

▪ 쿠버네티스 아키텍처

◘ 쿠버네티스 API: 선언적 API를 사용비, 비가역적 인프라(Immutable Infrastructure)

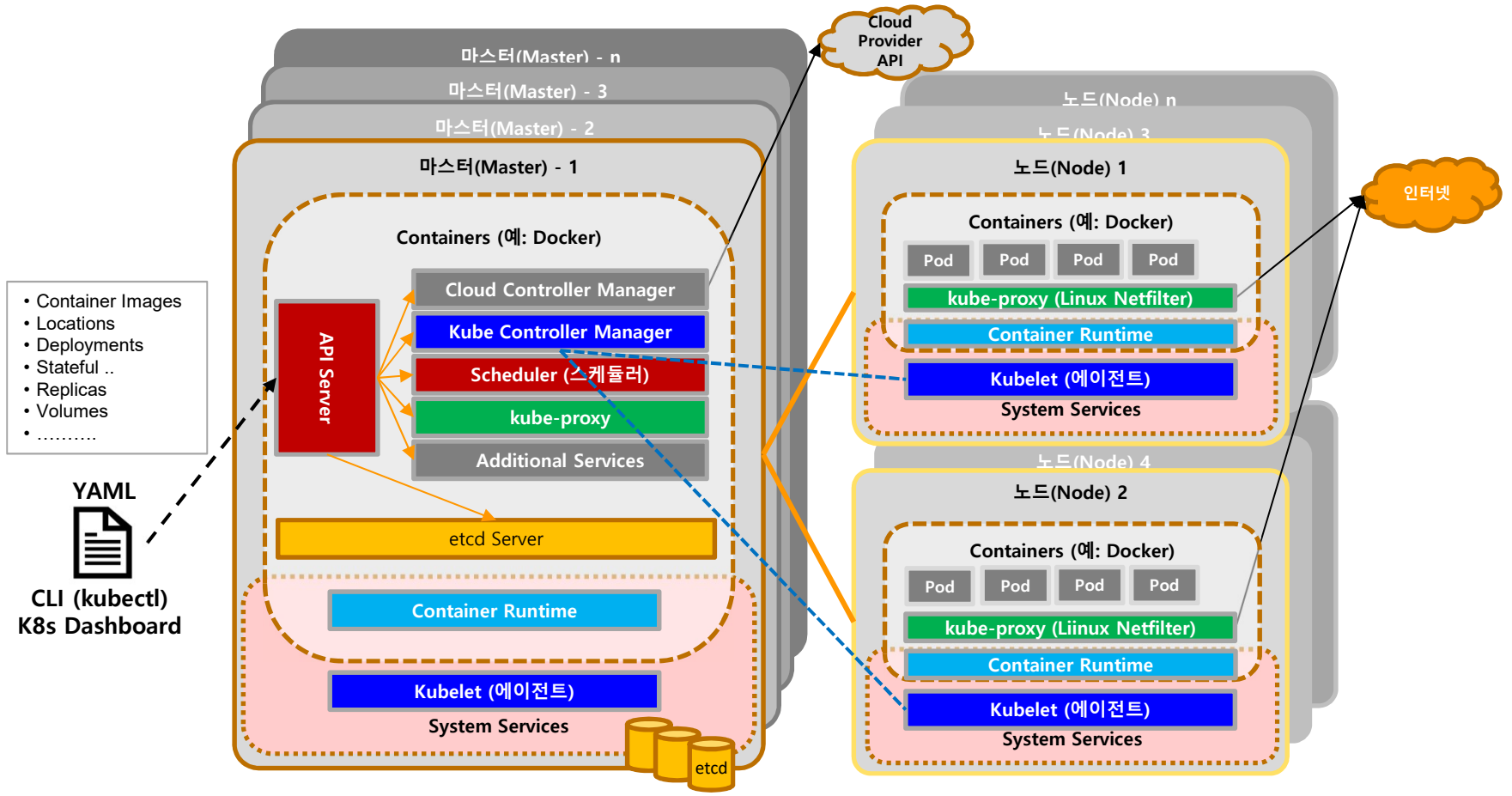


https://k8syaml.com/?fbclid=IwAR2KEzz0EsrwHlg_kIlOB4YYQRpd3Vzrm72xhbFulz_pVDvw0tb2ZIe5-LA

james@jslab.kr

▪ 쿠버네티스 아키텍처

◻ 쿠버네티스 아키텍처: 구성



❑ Session 2: 컨테이너 기반 인프라

• 컨테이너 사용 인프라 구성

▪ 쿠버네티스 아키텍처

◻ 쿠버네티스 아키텍처: 쿠버네티스(K8s) 실행을 위해 구동중인 Docker Container

```
jslab@ubuntu:~$ sudo docker run -d --privileged --restart=unless-stopped --net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run rancher/rancher-agent:v2.5.1 --server https://192.168.50.106 --token
s9t9bm5f6hzc9mnmd2srtgjskwvx45rk7mwflggnjbdtbfq5ldzfb2 --ca-checksum db4984d8f41907eb1f5b1b2fba99cb0cc32eedf224aa7c551cb69c097f717349 --etcd --controlplane --worker
jslab@ubuntu:~$
jslab@ubuntu:~$ sudo docker ps
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS          PORTS     NAMES
ff15cc5a49d1   e2987812f35f                                   "fleetagent"             About an hour ago   Up About an hour             k8s_fleet-agent_fleet-agent-64d854c5b-qwr6l_fleet-system_d0ac8d2f-406d-4b73-bd0c-75598bab5df7_0
9df1753d5e29   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_fleet-agent-64d854c5b-qwr6l_fleet-system_d0ac8d2f-406d-4b73-bd0c-75598bab5df7_0
a42456ac8767   141c7125f5a2                                   "run.sh"                 About an hour ago   Up About an hour             k8s_cluster-register_cattle-cluster-agent-5b5795676-w98lw_cattle-system_8d2a4eca-9c49-44f2-9990-bfe5a72a158f_1
1403f3c9c826   rancher/cluster-proportional-autoscaler        "/cluster-proportion…"   About an hour ago   Up About an hour             k8s_autoscaler_coredns-autoscaler-79599b9dc6-ttsbj_kube-system_27006763-8a27-41d5-91f1-b6e5bb6d2e21_0
99269bcd3c46   rancher/metrics-server                         "/metrics-server --k…"   About an hour ago   Up About an hour             k8s_metrics-server_metrics-server-8449844bf-cdhfb_kube-system_ee4f14c6-301f-4938-a751-f41c2db0edea_0
cbac9e001d44   rancher/calico-kube-controllers                "/usr/bin/kube-contr…"   About an hour ago   Up About an hour             k8s_calico-kube-controllers_calico-kube-controllers-649b7b795b-pws9t_kube-system_45c48b5e-e5ab-4e89-b698-644e8161d693_0
0844bced66fc   rancher/coredns-coredns                        "/coredns -conf /etc…"   About an hour ago   Up About an hour             k8s_coredns_coredns-6f85d5fb88-zf5dv_kube-system_6435e780-566f-4fbd-80ad-6a421d62a97b_0
8eb6024e55bc   rancher/nginx-ingress-controller-defaultbackend   "/server"                About an hour ago   Up About an hour             k8s_default-http-backend_default-http-backend-65dd5949d9-t7fkg_ingress-nginx_611f08b2-f083-45f2-9a72-d46d24d46340_0
9bdceca98288   rancher/coreos-flannel                         "/opt/bin/flanneld -…"   About an hour ago   Up About an hour             k8s_kube-flannel_canal-5fkbj_kube-system_c6a797cb-7666-4310-ba34-b4b8ac765d80_0
32eabf69b0c5   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_metrics-server-8449844bf-cdhfb_kube-system_ee4f14c6-301f-4938-a751-f41c2db0edea_7
7221b5b29e8a   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_coredns-autoscaler-79599b9dc6-ttsbj_kube-system_27006763-8a27-41d5-91f1-b6e5bb6d2e21_6
4b1f4bf8580a   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_cattle-cluster-agent-5b5795676-w98lw_cattle-system_8d2a4eca-9c49-44f2-9990-bfe5a72a158f_4
2b6f1264fcde   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_calico-kube-controllers-649b7b795b-pws9t_kube-system_45c48b5e-e5ab-4e89-b698-644e8161d693_6
168e115eea6e   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_coredns-6f85d5fb88-zf5dv_kube-system_6435e780-566f-4fbd-80ad-6a421d62a97b_6
a94a33c47bf2   rancher/kube-api-auth                          "/bin/sh -c 'kube-ap…"   About an hour ago   Up About an hour             k8s_kube-api-auth_kube-api-auth-7ftqw_cattle-system_0b06d3bb-7bee-46be-a509-2f76c7e1e570_0
cfcdf27df0b7   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_default-http-backend-65dd5949d9-t7fkg_ingress-nginx_611f08b2-f083-45f2-9a72-d46d24d46340_4
96d16db38392   rancher/calico-node                            "start_runit"            About an hour ago   Up About an hour             k8s_calico-node_canal-5fkbj_kube-system_c6a797cb-7666-4310-ba34-b4b8ac765d80_0
017acaa16227   rancher/nginx-ingress-controller               "/usr/bin/dumb-init …"   About an hour ago   Up About an hour             k8s_nginx-ingress-controller_nginx-ingress-controller-841pv_ingress-nginx_5a4abef3-728e-4200-ab1b-4da07f96290a_0
7440e50492e2   rancher/rancher-agent:v2.5.1                   "run.sh --server htt…"   About an hour ago   Up About an hour             upbeat_perlman
2d1d39eda669   141c7125f5a2                                   "run.sh"                 About an hour ago   Up About an hour             k8s_agent_cattle-node-agent-g7dp5_cattle-system_7081e4b5-e842-4971-90c5-0cb661d9130d_0
a15de0c34848   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_kube-api-auth-7ftqw_cattle-system_0b06d3bb-7bee-46be-a509-2f76c7e1e570_0
b5bf7b44e16b   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_cattle-node-agent-g7dp5_cattle-system_7081e4b5-e842-4971-90c5-0cb661d9130d_0
1246a31b72f4   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_nginx-ingress-controller-841pv_ingress-nginx_5a4abef3-728e-4200-ab1b-4da07f96290a_0
c6cce211e893   rancher/pause:3.2                              "/pause"                 About an hour ago   Up About an hour             k8s_POD_canal-5fkbj_kube-system_c6a797cb-7666-4310-ba34-b4b8ac765d80_0
0ac9607f674f   rancher/hyperkube:v1.19.3-rancher1             "/opt/rke-tools/entr…"   About an hour ago   Up About an hour             kube-proxy
3ca878ea36e9   rancher/hyperkube:v1.19.3-rancher1             "/opt/rke-tools/entr…"   About an hour ago   Up About an hour             kubelet
3fc884e9d069   rancher/hyperkube:v1.19.3-rancher1             "/opt/rke-tools/entr…"   About an hour ago   Up About an hour             kube-scheduler
c69d320650e3   rancher/hyperkube:v1.19.3-rancher1             "/opt/rke-tools/entr…"   About an hour ago   Up About an hour             kube-controller-manager
a7f598a35020   rancher/hyperkube:v1.19.3-rancher1             "/opt/rke-tools/entr…"   About an hour ago   Up About an hour             kube-apiserver
d4c82441f902   rancher/coreos-etcd:v3.4.13-rancher1           "/usr/local/bin/etcd…"   About an hour ago   Up About an hour             etcd
```

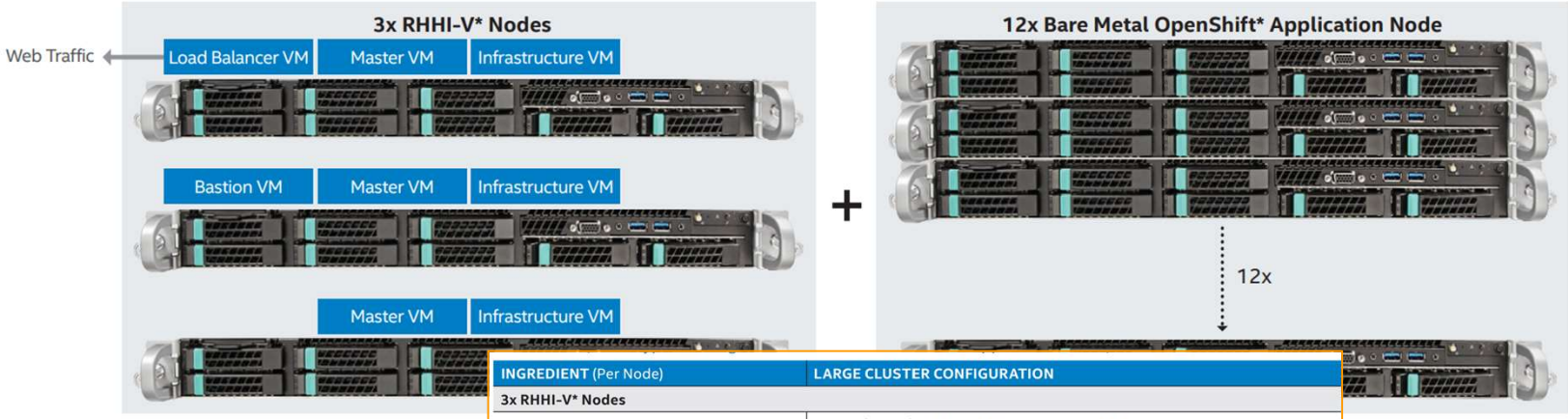
❑ Session 2: 컨테이너 기반 인프라

• 컨테이너 사용 인프라 구성

▪ 오픈 하드웨어 구성

◻ 인텔의 Red Hat 'OpenShift' 권장 하드웨어 인프라 구성 (예)

- OpenShift는 Kubernetes 오케스트레이션 사용



https://www.intel.com/content/dam/www/public/us/en/documents/reference-architectures/deploying-red-hat-openshift-container-platform-3-11-with-container-native-storage.pdf

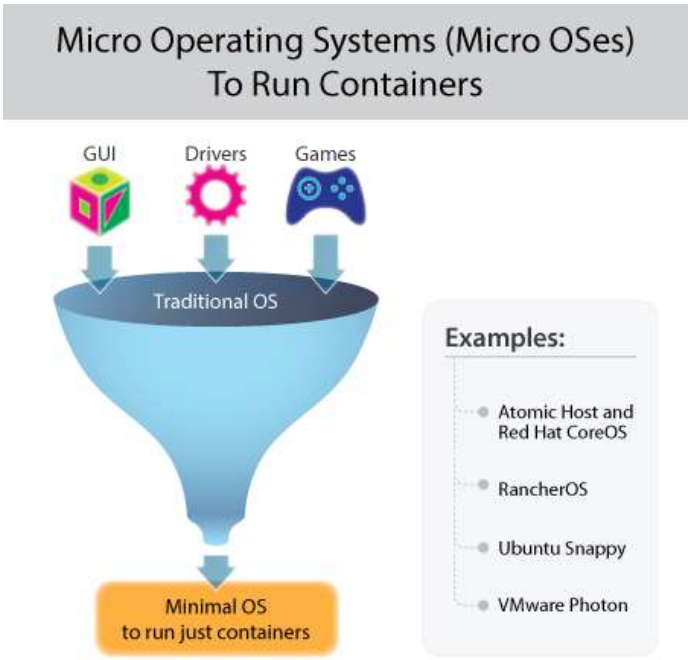
INGREDIENT (Per Node)	LARGE CLUSTER CONFIGURATION
3x RHHI-V* Nodes	
Processor	2x Intel® Xeon® Gold 5218 Processor at 2.3 GHz
Memory	384 GB (12x32 GB)
Boot Drive	2x Intel® SSD D3-S4510 Series 480 GB 2.5-inch RAID1
RHHI-V Storage	1x Intel® SSD DC P4510 Series 4 TB
OpenShift* Container Storage (Registry)	1x Intel® SSD DC P4510 Series 4 TB
Storage HBA Controller[a]	Intel® RAID Module RMSP3HD080E
Remote Management Module[a]	Intel® Remote Management Module 4 Lite 2 AXXRMM4LITE2
Network	1x Intel® Ethernet Network Adapter XXV710-DA2, Dual-port 25 Gbps SFP28
12x OpenShift* Application Nodes	
Processor	2x Intel® Xeon® Gold 6230 processor (2.1 GHz, 22 cores, 44 threads)
Memory	384 GB or higher (12x32 GB)
Intel® Optane™ DC Persistent Memory	1024 GB (4x256 GB)
Boot Drive	2x Intel® SSD D3-S4510 Series 480 GB 2.5-inch RAID1
OpenShift* Container Storage (Application)	1x Intel® SSD DC P4510 Series 4 TB (NVMe*)
Storage HBA Controller[a]	Intel® RAID Module RMSP3HD080E
Remote Management Module[a]	Intel® Remote Management Module 4 Lite 2 AXXRMM4LITE2
Network	1x Intel® Ethernet Network Adapter XXV710-DA2, Dual-port 25 Gbps SFP28

[a] *Recommended but not required.*

▪ 컨테이너 전용 OS

◻ Micro OSes for containers

- Atomic Host and Red Hat CoreOS
- RancherOS
- Ubuntu Snappy
- VMware Photon



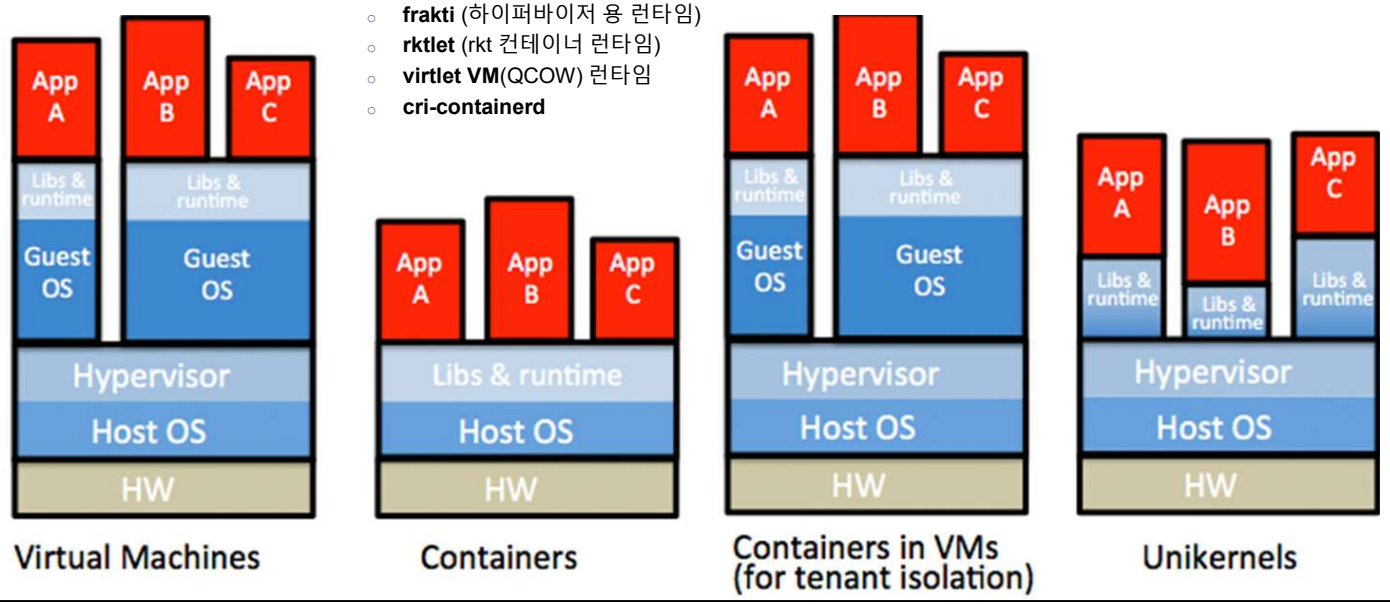
james@jslab.kr

▪ 쿠버네티스 배포 가능 자원

◻ 쿠버네티스 배포 가능 자원

- 커널 분리 컨테이너 (Docker Container, LXC, …)
- 커널 포함 UniKernel
- Micro-VM
- VM

- **Docker** (CRI shim 라이브러리 사용)
- **dockershim**
- **Rkt** (CoreOS에서 파생)
- **cri-o** (도커와 같이 OCI 호환하는 OCI confirmed 런타임, K8s 프로젝트)
- **frakti** (하이퍼바이저 용 런타임)
- **rktlet** (rkt 컨테이너 런타임)
- **virtlet VM**(QCOW) 런타임
- **cri-containerd**



james@jslab.kr

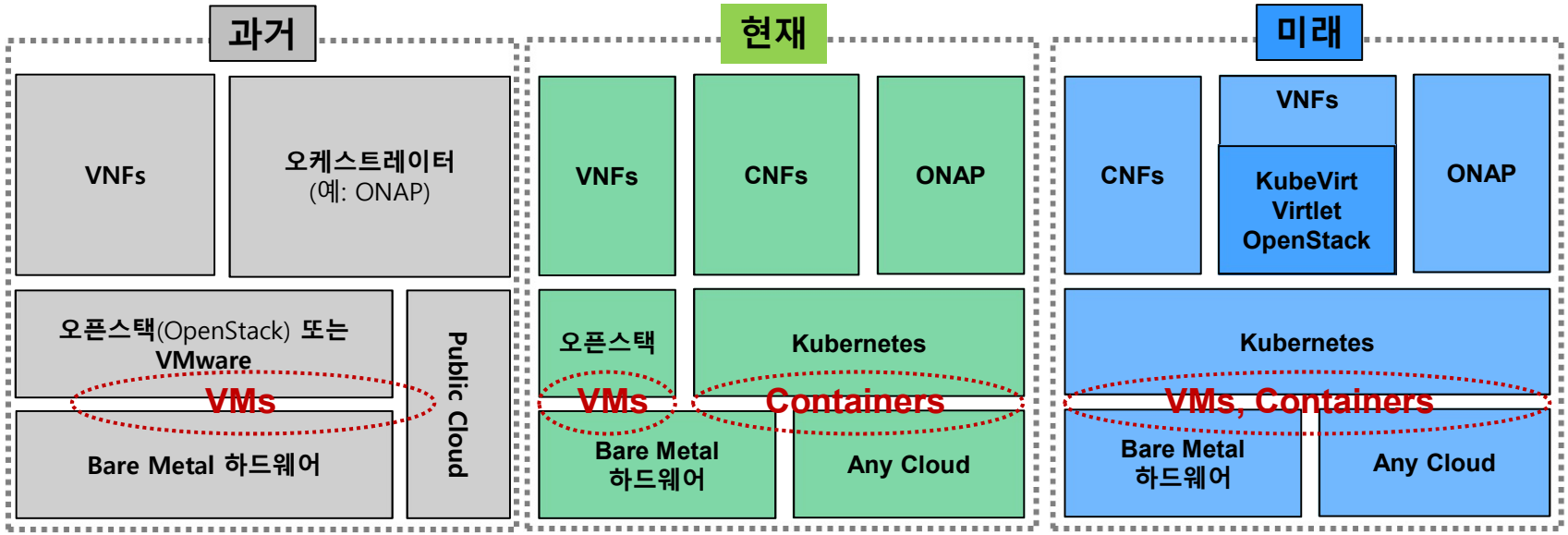
- ❑ Session 1: 컨테이너 인프라 개요
  - 컨테이너 기술 발전
  - 컨테이너 기술 특징
- ❑ Session 2: 컨테이너 기반 인프라
  - 컨테이너 사용 인프라 구성
  - 컨테이너 종류별 기술 비교
- ❑ **Session 3: 컨테이너 응용 네트워크 기술**
  - **클라우드 네이티브 (K8s CNI, Service Mesh)**
  - **통신 인프라 환경 컨테이너 기반 서비스**

james@jslab.kr

▪ Running VNFs/CNFs on Kubernetes

◻ Current challenges

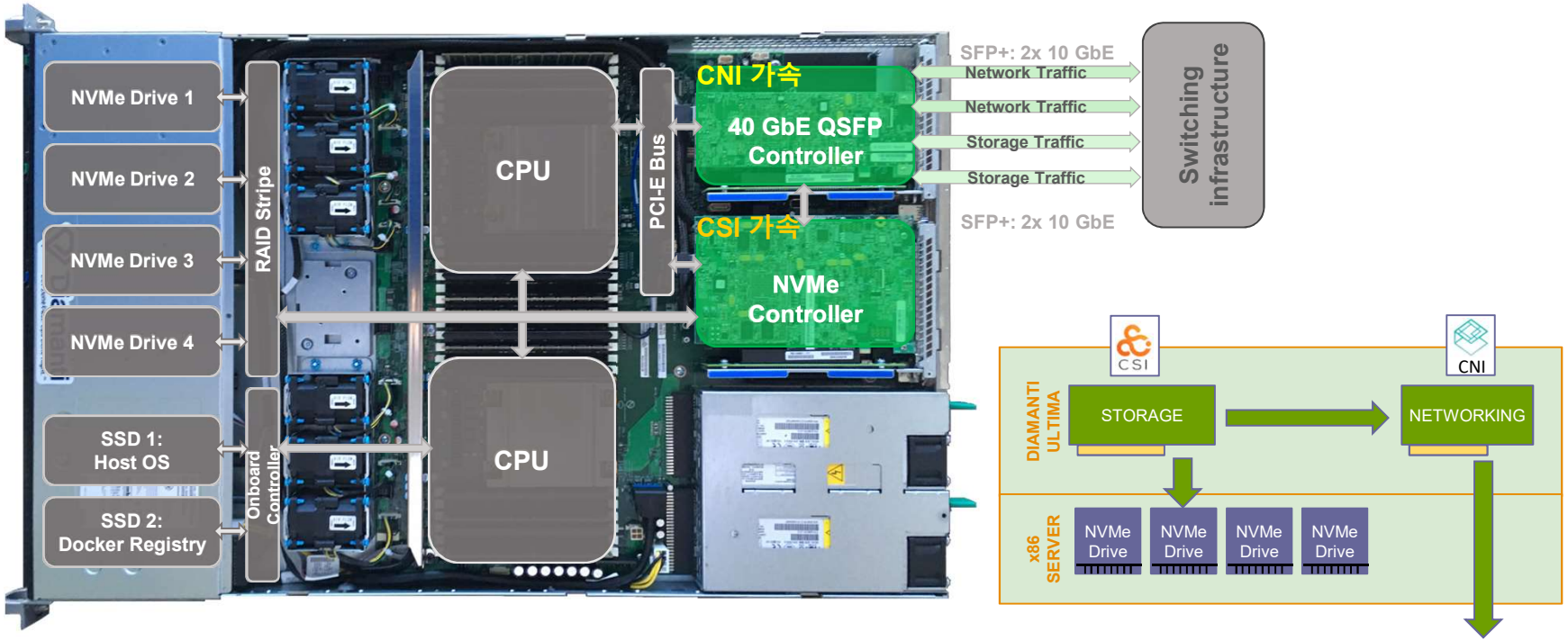
- Resource affinity and awareness for workloads
- Building a fast data-path



▪ 쿠버네티스 CNI 가속 지원 하드웨어

◻ 쿠버네티스 가속 (예): Diamanti D10의 CNI/CSI 가속

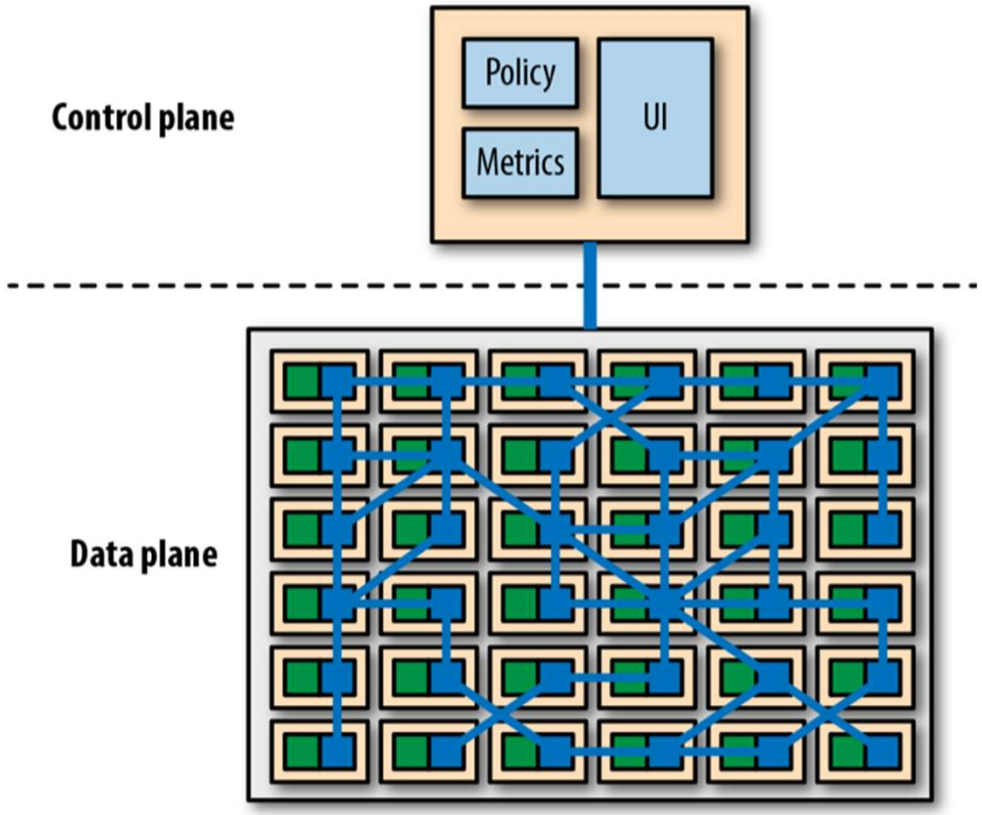
- Diamanti Platform Networking: Layer-2 interfaces made available to all containers
- Diamanti Platform Storage: Persistent, high-speed, sub-ms NVMe flash is distributed throughout the cluster



▪ 서비스 메시 (Service Mesh)

◻ 서비스 메시 구성

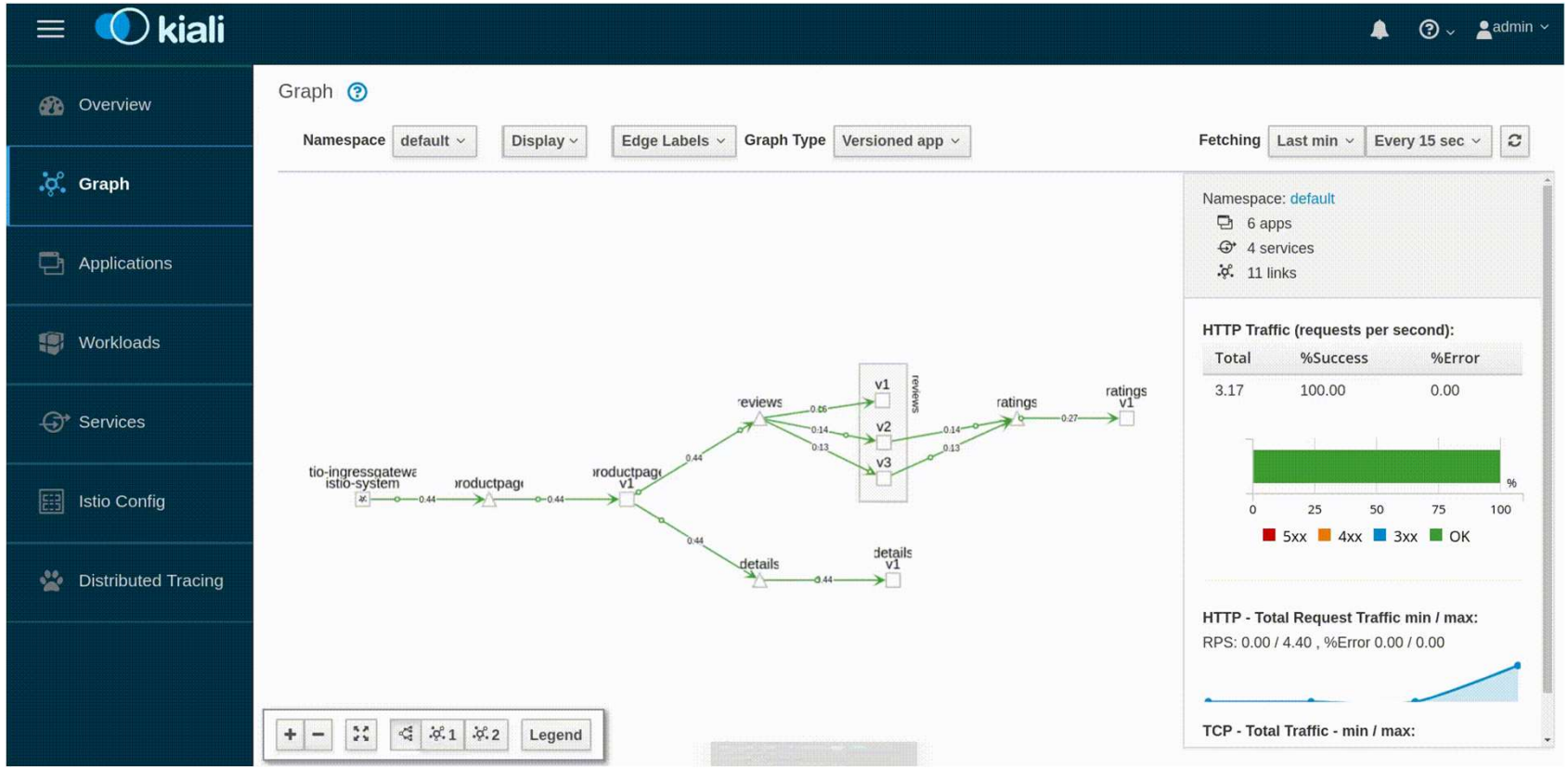
- A service mesh is a dedicated infrastructure layer for handling service-to-service communication
- https://awskrug.github.io/eks-workshop/servicemesh_with_istio/



▪ 서비스 메시 (Service Mesh)

◻ 서비스 메시 구성

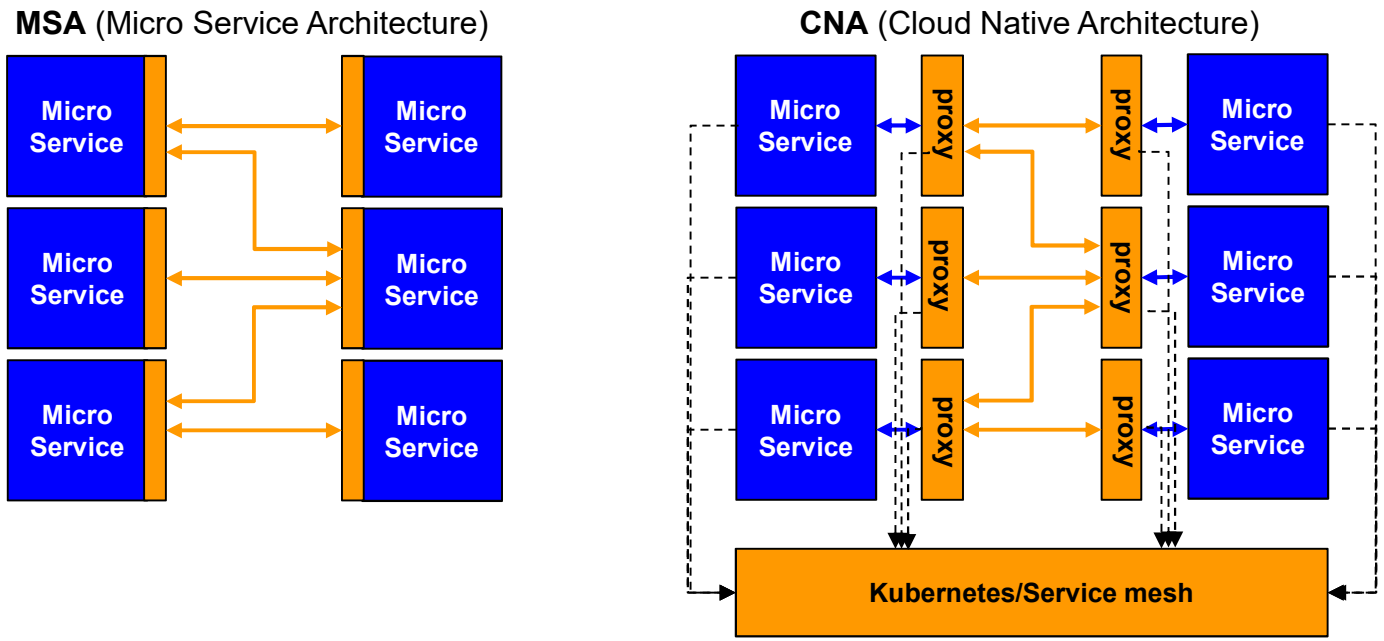
- Service Registry & Discovery
- Service Mesh (예: 리눅스재단 CNCF의 Istio)



▪ 컨테이너 기반 인프라의 SDN 기반 아키텍처 구성

▫ User Plane – Control Plane 분리 : SDN Architecture

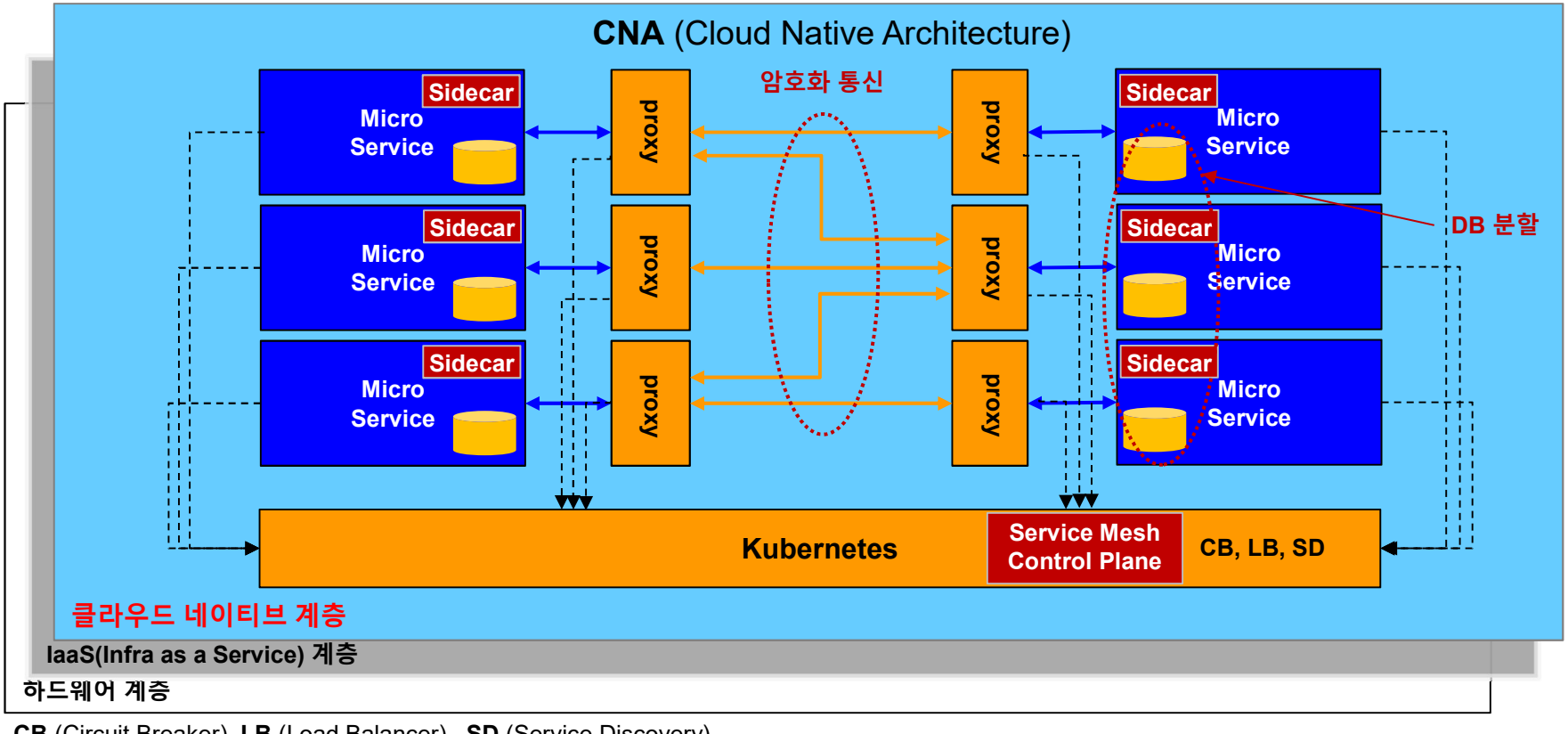
- 오케스트레이션은 쿠버네티스 사용
- 컨테이너 기반 아키텍처
- **MSA**(Micro Service Architecture)**: Smart endpoints, dump pipes**
- **CNA**(Cloud Native Architecture)**: Infrastructure focused smart platform, business logic focused smart services**



▪ CNA 의 서비스 메쉬(Service Mesh) 관리

◻ User Plane – Control Plane 분리 : SDN Architecture

- Sidecar Design Pattern: 라우터 내장 CB, LB, SD 내장
- CNCF의 'Istio'는 정책 강화 Telemetry 제공

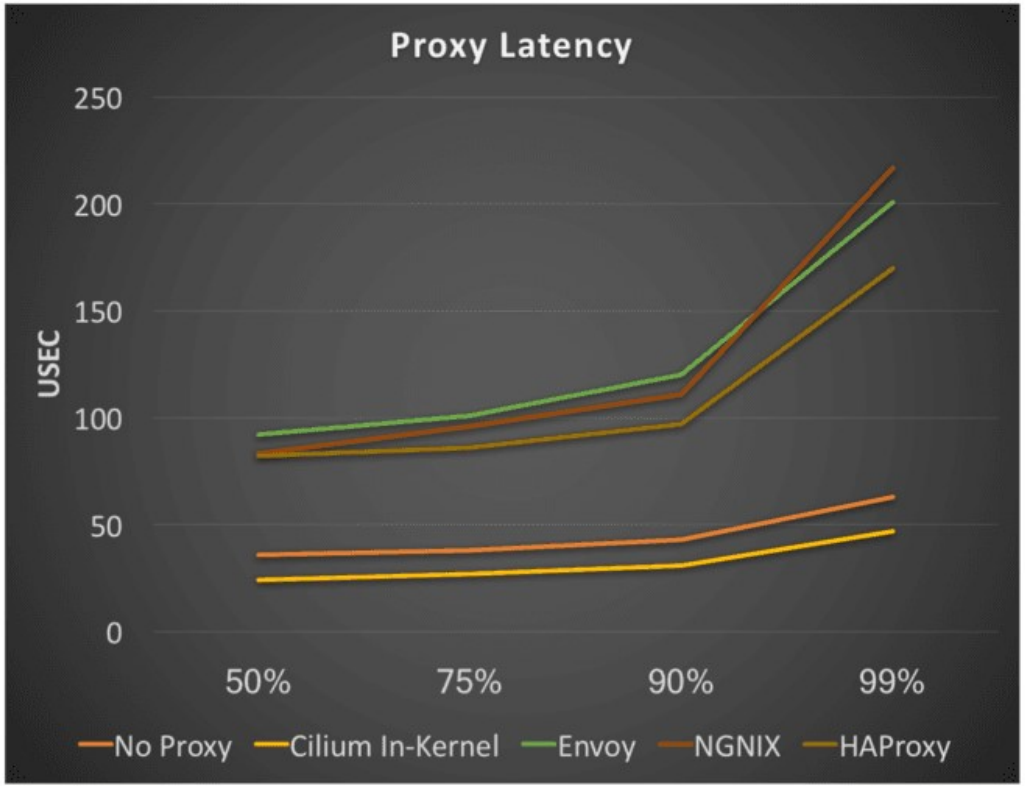


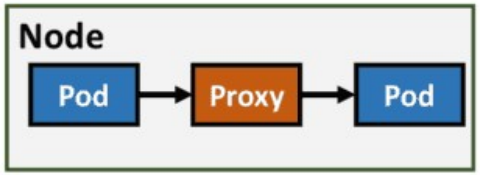
CB (Circuit Breaker), **LB** (Load Balancer), **SD** (Service Discovery)

▪ CNA 의 서비스 메쉬(Service Mesh) 가속

▫ CNCF 서비스메시 Istio는 현재 Envoy 사용

- 지연(Latency): eBPF(Cilium)<No Proxy<HAProxy<NGINX<Envoy





Notes:

- This excludes parsing logic and policy rule execution
- Istio may route requests through mixer which will add latency

james@jslab.kr

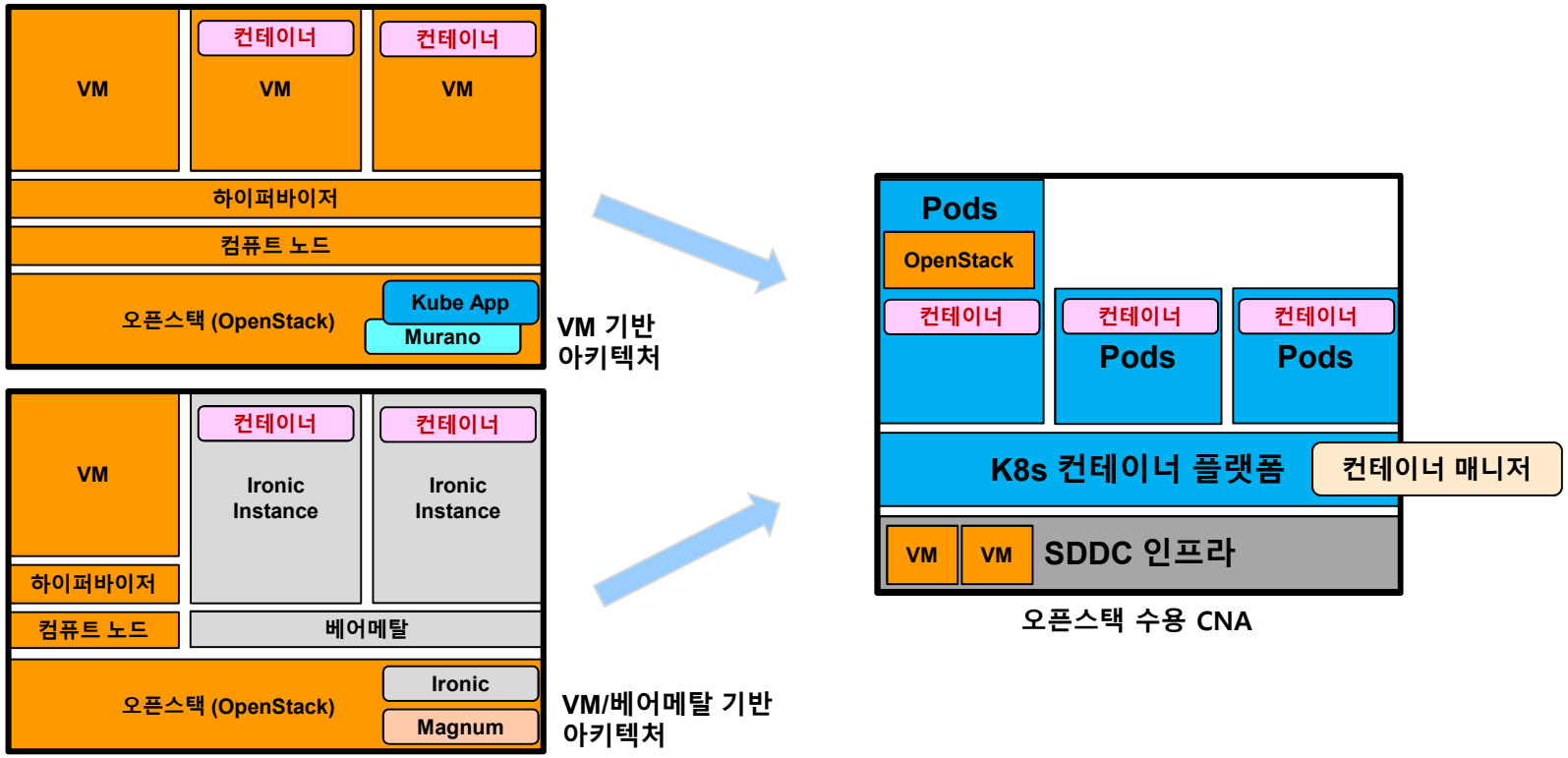
▪ 멀티클라우드 환경

◻ 컨테이너 기반 인프라 지원 멀티클라우드 환경

- 오케스트레이션은 쿠버네티스 사용

▪ 오픈스택의 CNF 수용 (예)

- 오픈스택은 쿠버네티스 연계 확장 지원 (Murano)
- 오픈스택은 제어서비스의 컨테이너들로 쿠버네티스 상에서 오케스트레이션 제어 가능 (Kolla)

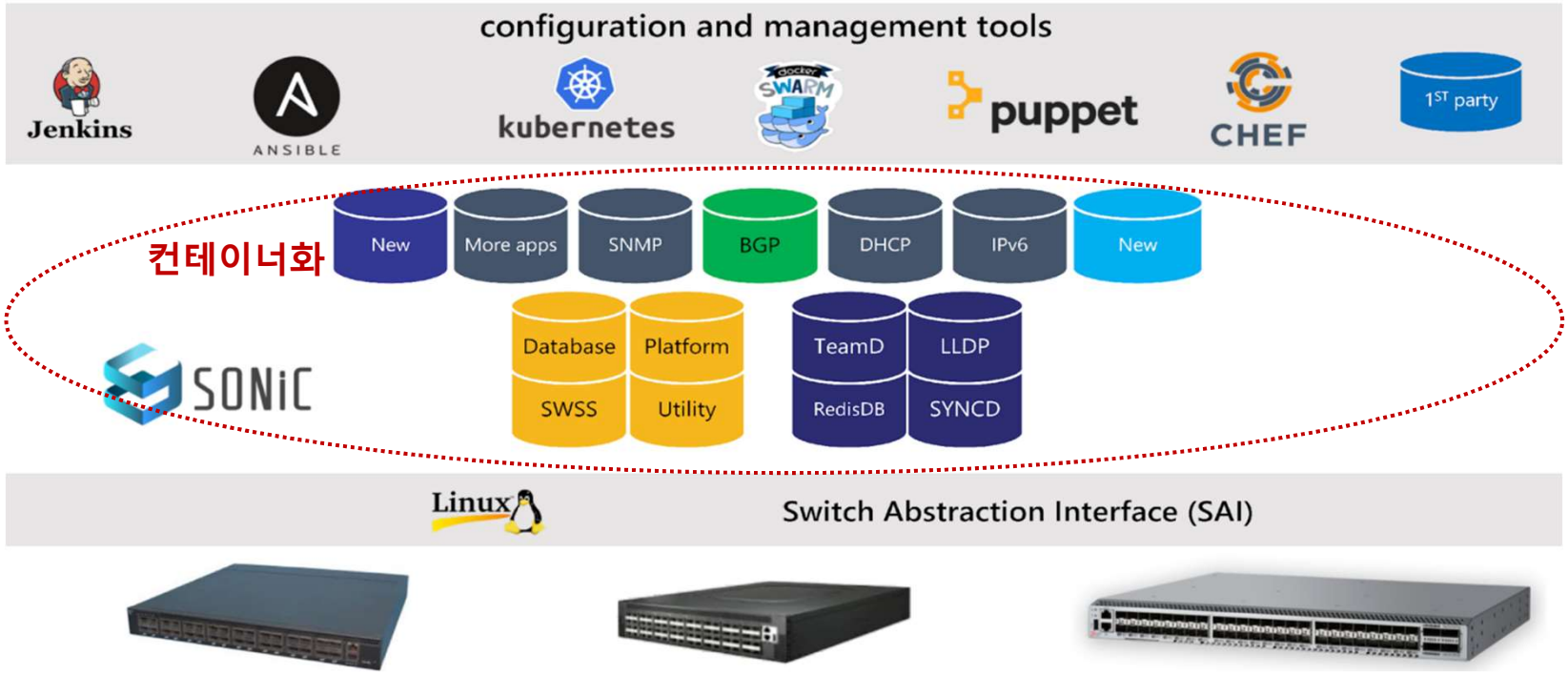


james@jslab.kr

▪ SONiC

SONiC (Software for Open Networking in the Cloud)

- SONiC은 마이크로소프트에서 주도로 네트워크 기능을 컨테이너화하는 아키텍처 클라우드 네이티브 수용 체계
- https://github.com/Azure/SONiC/wiki/Architecture
- Kubernetes는 자원 배포 단위로 Pod를 사용하여 추상화
- Pod는 Container로 구성, 제조사는 추상화 지원

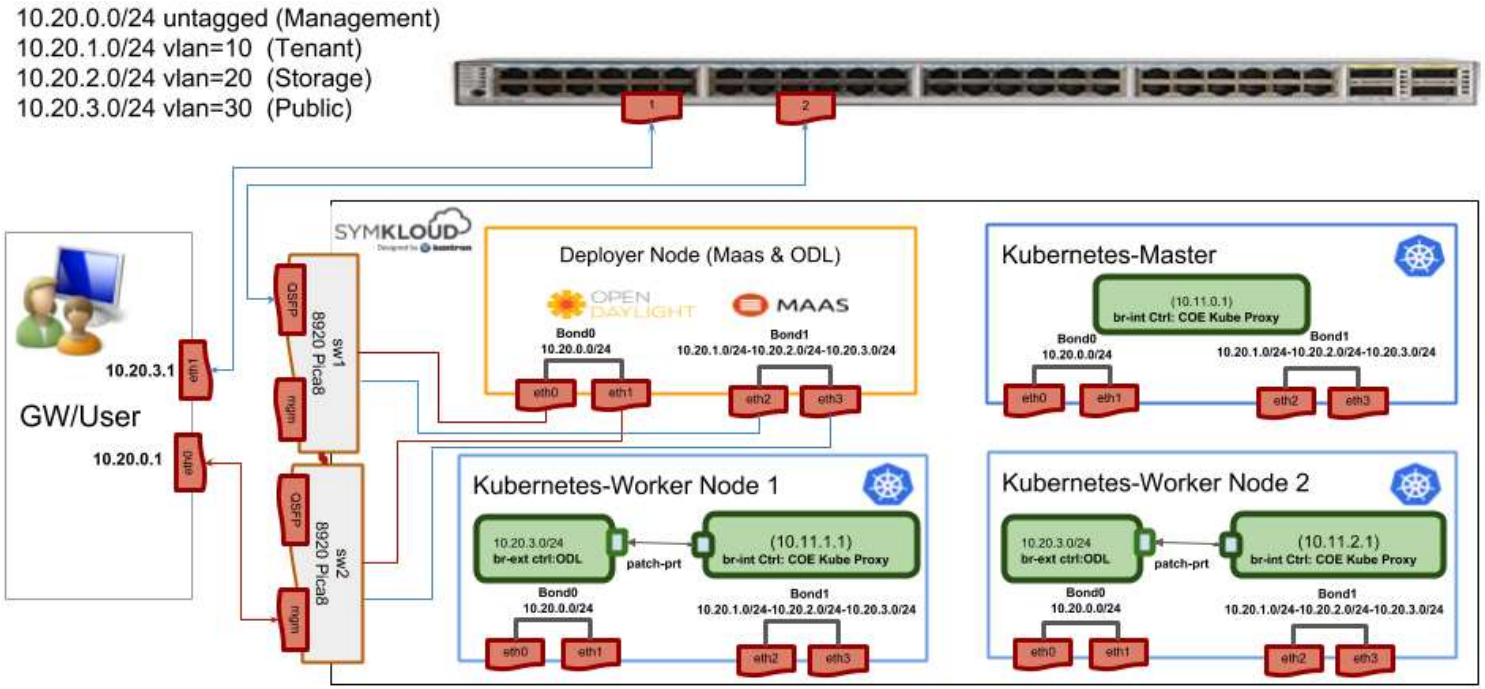


james@jslab.kr

▪ 텔코(Telco)를 위한 SDN 스택에 쿠버네티스 적용

◻ Telco를 위한 K8s의 제어플레인 기능 배포

- Telco의 SDN 스택을 위한 K8s 채택 (예)
- K8s는 제어기능의 배포 위치 변경 요구 수용 필요 (Datacenter 또는 User-plane)

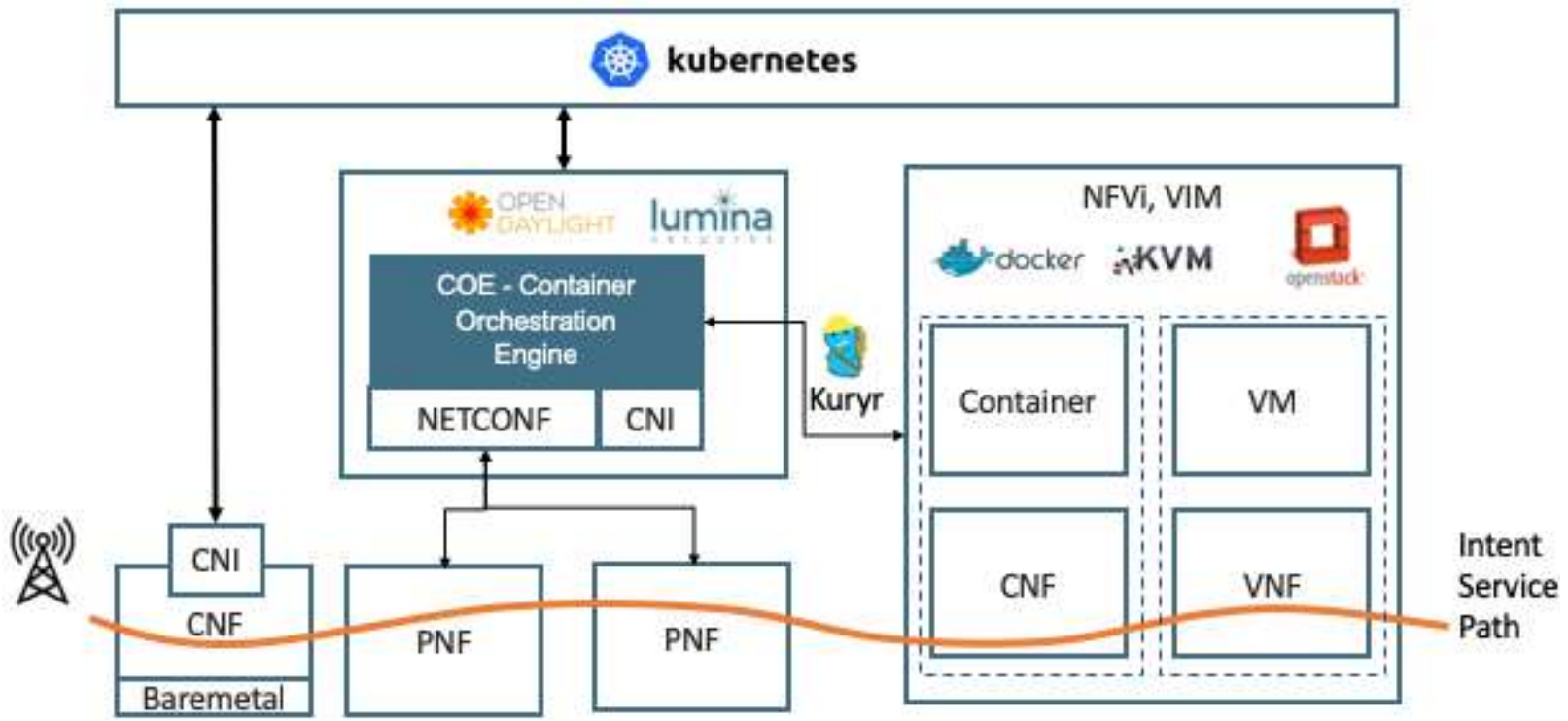


https://blog.symkloud.com/adapting-kubernetes-to-the-sdn-stack-for-telco

▪ 통신망의 종단간 (End-to-End) 구성

◻ 기존 통신 장비 제조사와 협력이 필요한 브라운 필드(Brownfield) 적용

- 컨테이너, 가상머신(VM), 물리머신(BareMetal) 혼용 환경 추상화 (Intent)
- 제조사 솔루션 (예): SDN 제어기, 오픈스택, 도커 사용



james@jslab.kr

감사합니다