Edge Cloud Computing

2019. 5.

안 종 석
james@jslab.kr
JS Lab

목차

목차

james@jslab.kr

I. 개요

❖ **A Vapor IO Kinetic Edge micro data center operating alongside a cellular tower.**

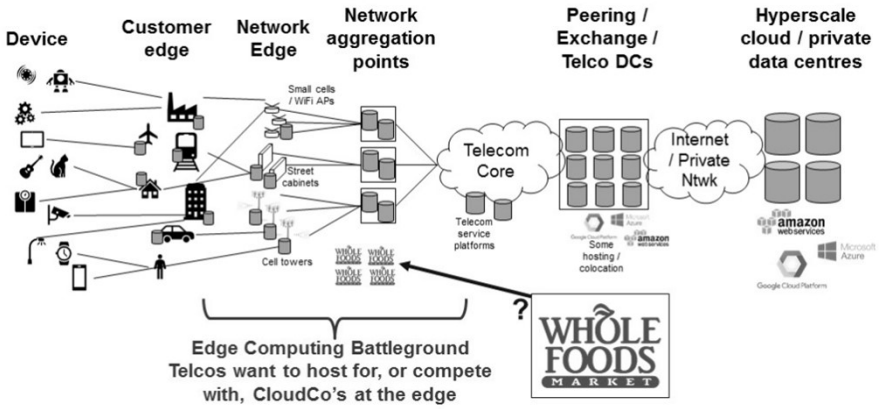
https://www.zdnet.com/article/where-the-edge-is-in-edge-computing-why-it-matters-and-how-we-use-it/

james@jslab.kr

JS Lab

I. 개요

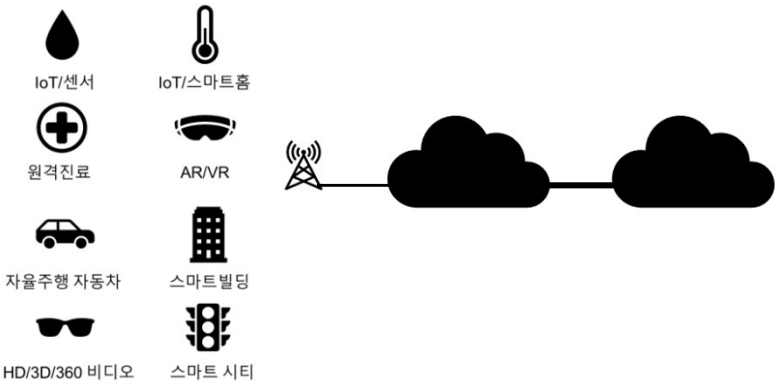
❖ 아마존은 왜 Whole Foods를 인수 했나?



https://disruptivewireless.blogspot.com/2017/06/does-amazons-purchase-of-whole-foods.html

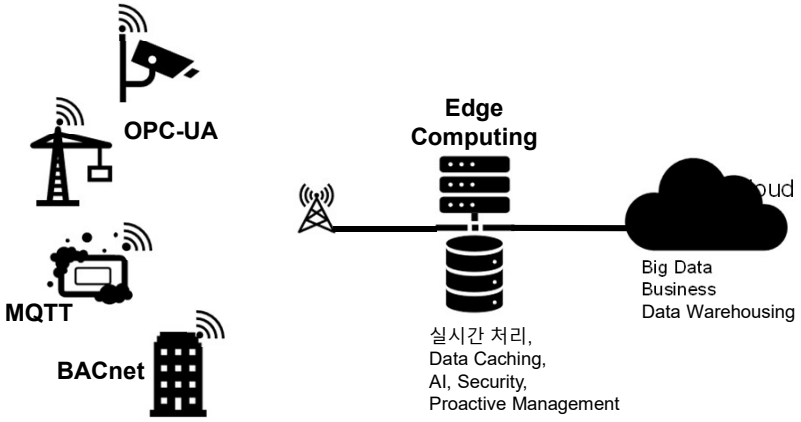
I. 개요

❖ **에지 클라우드 (Edge Cloud) :** 에지(Edge)에서 새로운 서비스와 애플리케이션을 지원하기 위한 서비스 사업자의 인프라 구성으로 새로운 시장의 기회가 열리고 5G를 포함한 발전단계를 위한 준비가 필요함



I. 개요

❖ **에지 컴퓨팅 (Edge Computing) :**데이터를 발생하는 사물 옆이나 내장하는 형태의 컴퓨팅



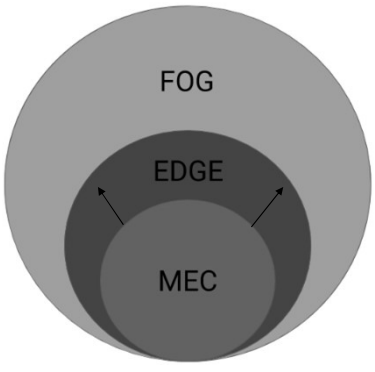
BACnet Building Automation and Control (BAC) networks **OPC** Unified Architecture (**OPC UA**) - Open Platform Communications

I. 개요

❖ **Edge computing is a more general concept than MEC and less general than fog computing.**

❖ **ETSI previously referred to MEC as Edge Computing. Over time, the field expanded the focus of edge computing beyond mobile implementations.**

Source: SDxCentral

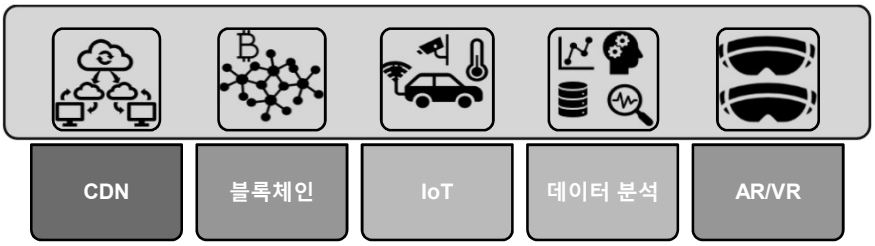


https://www.sdxcentral.com/edge/definitions/whats-the-difference-between-edge-computing-and-mec/

I. 개요

- **Edge Cloud Computing:** 에지(Edge)에서 클라우드 서비스 기반의 컴퓨팅을 제공
  - **지연 개선** (Reducing latency)
  - **대역폭 부족 완화** (Mitigating bandwidth limits)
- **제공 가능 서비스** (예) **:** CDN, 블록체인, IoT, 데이터 분석, AR/VR 등

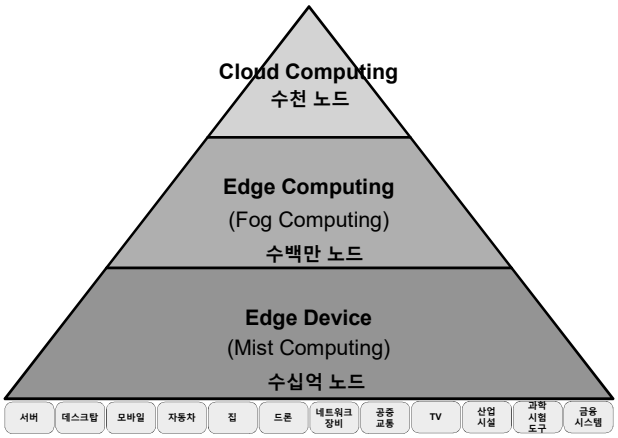
james@jslab.kr



I. 개요

- **클라우드 기술 적용 영역 확대**
- **네트워킹 오픈 프로젝트 증가**

james@jslab.kr



I. 개요

❖ **ETSI의 MEC (Mobile|Multi-access Edge Computing)**
❖ **OpenStack Foundation의 Edge Computing Group**

james@jslab.kr

▪ **ETSI**

Mobile Edge Computing → Multi-access Edge Computing (2017)

▪ **OpenStack Foundation**

Massive distributed Working Group (2016) → Fog Edge Massively Distributed Cloud(FEMDC) SIG (2017) → Edge Computing Group (2018)

https://www.openstack.org/edge-computing/cloud-edge-computing-beyond-the-data-center?lang=en_US

I. 개요

❖ **LF Edge: 리눅스 재단이 2019년 1월 시작한 프로젝트**
❖ **60 members, including Arm, AT&T, Dell, Ericsson, IBM, Intel, Huawei, Red Hat, Samsung**
❖ **Projects:**

- **Akraino Edge Stack**
- **EdgeX Foundry** (a common open framework for IoT edge computing)
- **Open Glossary of Edge Computing**
- **Home Edge Project**
- **EVE** (Edge Virtualization Engine, open and agnostic standard edge architecture)

james@jslab.kr

LF EDGE

AKRAINO EDGE STACK · HOME EDGE · OPEN GLOSSARY OF EDGE COMPUTING · EVE EDGE VIRTUALIZATION ENGINE · EDGE X FOUNDRY

https://www.lfedge.org/

I. 개요

❖ **Cloud Edge Computing: Beyond the Data Center**
❖ **Akraino, Airship, StalingX의 관계**

james@jslab.kr

OpenStack

(코드 개발)

airship

STARLINGX

Linux Foundation

(Use Case 정의,
Integration, 검증)

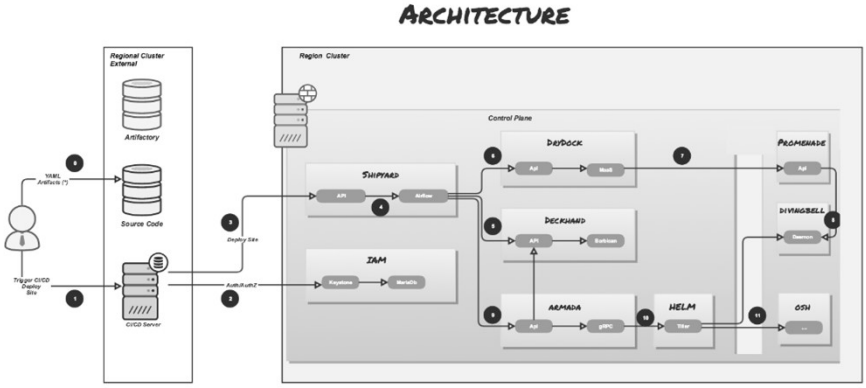
AKRAINO EDGE STACK

https://wiki.openstack.org/wiki/Edge_Computing_Group?fbclid=IwAR3GNTB5_2IOJO-SvaGsmhCC2jhLxG9X-ISt021v-mIfG-TxsR7jiPtrM80

I. 개요

❖ **Airship:** a collection of components that coordinate to form means of configuring and deploying and maintaining a Kubernetes environment using a declarative set of yaml documents. More specifically, the current focus of this project is the implementation of OpenStack on Kubernetes (OOK).

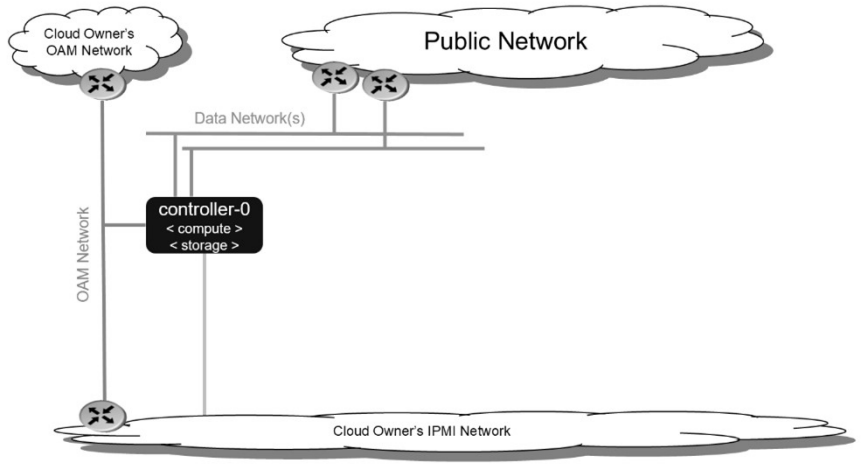
james@jslab.kr



https://www.airshipit.org/

I. 개요

❖ **StarlingX:** a fully featured and high performance Edge Cloud software stack that is based on the Wind River® Titanium Cloud R5 product

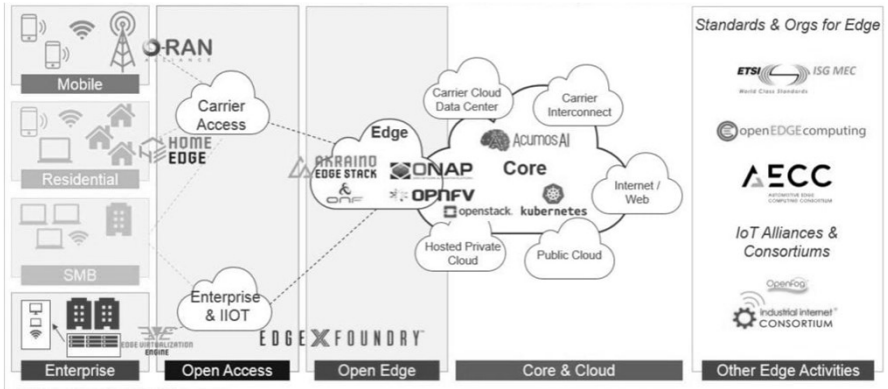


https://www.starlingx.io/

I. 개요

❖ **LF Open Source Edge**

❖ **Standards, Ref Arch and Ref Implementation**



THE LINUX FOUNDATION

I. 개요

❖ **Market:** Gartner Top 10 Trends Impacting Infrastructure & Operations for 2019

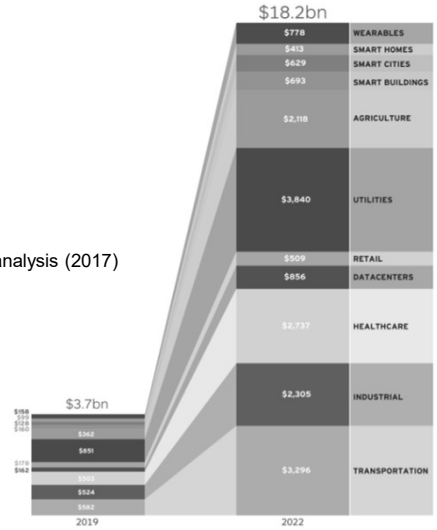
https://www.gartner.com/smarterwithgartner/top-10-trends-impacting-infrastructure-and-operations-for-2019/

james@jslab.kr

I. 개요

❖ **Market:** Growth of Fog opportunity by vertical market

451 Research OpenFog project analysis (2017)



james@jslab.kr

I. 개요

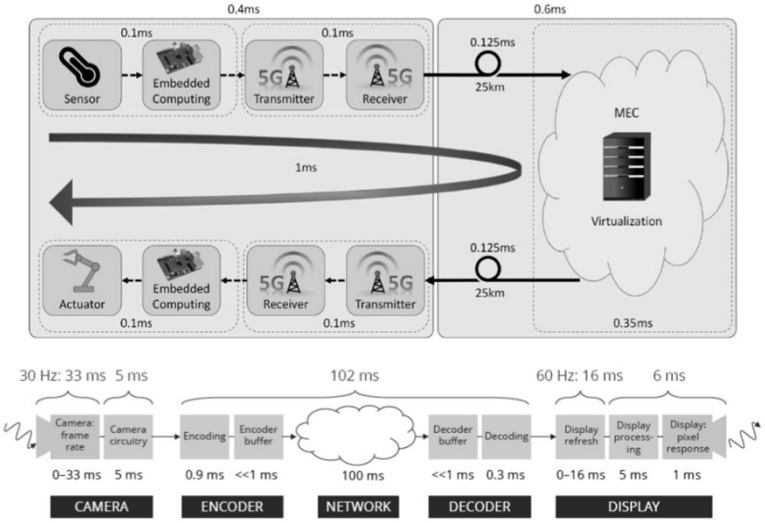
❖ **Edge Computing Landscape**

- **.ORG**
- **Traffic / Delivery**
- **Platform**
- **Infrastructure**
- **Hardware**
- **Connectivity**
- **Real Estate**
- **Supply Chain**

james@jslab.kr

I. 개요

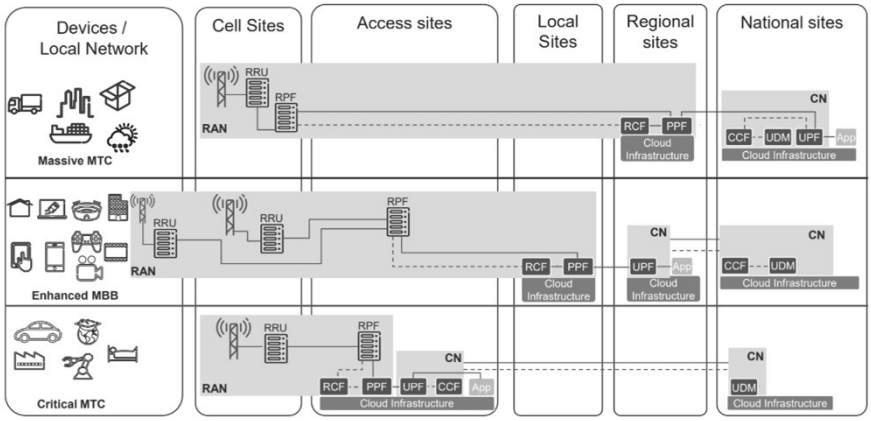
❖ **Latency 고려 Use Case 별 Telco 에지 인프라**



james@jslab.kr

I. 개요

❖ **네트워크 기능을 위한 클라우드 인프라의 위치는 서비스에 따라 다를 수 있음** (Ericsson 예 2018)



I. 개요

❖ **Preliminary 5G (NR) KPIs - 3GPP TR 38.913 (Draft 2016-09)**

항목	목표
최고 데이터 전송	20 Gbps 다운로드 / 10 Gbps 업로드
최고 스펙트럼 효율	30bps/Hz 다운로드 / 15bps/Hz 업로드
대역폭	1 GHz (DL+UL) , Pending ITU-R
제어 플레인 지연	10 ms
사용자 플레인 지연	URLLC: 0.5ms(DL)/0.5ms(UL), eMBB: 4ms(DL)/4ms(UL),
저사용 소형 패킷 지연	10 ms 이하
모빌리티 인터럽트	0 ms
시스템간 모빌리티	최소 LTE/LTE evolution 수준
신뢰성	eV2X와 URLLC를 위한 99.999%
커버리지 (범위)	LTE 수준
배터리	10년 이상, 15년 권장
셀 에지 스펙트럼 효율	IMT-Advanced의 3배
접속 밀도	도시환경 Km² 당 1,000,000대
지원 이동속도	500 Km/h

목표 값은 요구 진영에 따라 다를 수 있음

eMBB (enhanced mobile broadband) URLLC (ultra-reliable low latency)

I. 개요

❖ **권장 사용자 경험치**(UE Requirement: Data rate, Latency, Mobility)

Use case category	Connection Densit
Broadband access in dense areas	200-2500 /km^2
Indoor ultra-high broadband access	75,000 / km^2 (75/1000 m^2 office)
Broadband access in a crowd	150,000 / km^2 (30.000 / stadium)
50+ Mbps everywhere	400 / km^2 in suburban 100 / km^2 in rural
Ultra-low cost broadband access for low ARPU areas	16 / km^2
Mobile broadband in vehicles (cars, trains)	2000 / km^2 (500 active users per trai trains, or 1 active user per car x cars)
Airplanes connectivity	80 per plane 60 airplanes per 18,000 k
Massive low-cost/long-range/low-power MTC	Up to 200,000 / km^2
Broadband MTC	See the requirements for and 50+Mbps everywher
Ultra-low latency	Not critical
Resilience and traffic surge	10,000 / km^2
Ultra-high reliability & Ultra-low latency* (*) the reliability requirement for this category is described in Section 4.4.5	Not critical
Ultra-high availability & reliability* (*) the reliability requirement for this category is described in Section 4.4.5	Not critical
Broadcast like services	Not relevant

Use case category	User Experienced Data Rate	E2E Latency	Mobility
Broadband access in dense areas	DL: 300 Mbps UL: 50 Mbps	10 ms	On demand, 0-100 km/h
Indoor ultra-high broadband access	DL: 1 Gbps, UL: 500 Mbps	10 ms	Pedestrian
Broadband access in a crowd	DL: 25 Mbps UL: 50 Mbps	10 ms	Pedestrian
50+ Mbps everywhere	DL: 50 Mbps UL: 25 Mbps	10 ms	0-120 km/h
Ultra-low cost broadband access for low ARPU areas	DL: 10 Mbps UL: 10 Mbps	50 ms	on demand: 0-50 km/h
Mobile broadband in vehicles (cars, trains)	DL: 50 Mbps UL: 25 Mbps	10 ms	On demand, up to 500 km/h
Airplanes connectivity	DL: 15 Mbps per user UL: 7.5 Mbps per user	10 ms	Up to 1000 km/h
Massive low-cost/long-range/low-power MTC	Low (typically 1-100 kbps)	Seconds to hours	on demand: 0-500 km/h
Broadband MTC	See the requirements for the Broadband access in dense areas and 50+Mbps everywhere categories		
Ultra-low latency	DL: 50 Mbps UL: 25 Mbps	<1 ms	Pedestrian
Resilience and traffic surge	DL: 0.1-1 Mbps UL: 0.1-1 Mbps	Regular communication: not critical	0-120 km/h
Ultra-high reliability & Ultra-low latency	DL: From 50 kbps to 10 Mbps; UL: From a few bps to 10 Mbps	1 ms	on demand: 0-500 km/h
Ultra-high availability & reliability	DL: 10 Mbps UL: 10 Mbps	10 ms	On demand, 0-500 km/h
Broadcast like services	DL: Up to 200 Mbps UL: Modest (e.g. 500 kbps)	<100 ms	on demand: 0-500 km/h

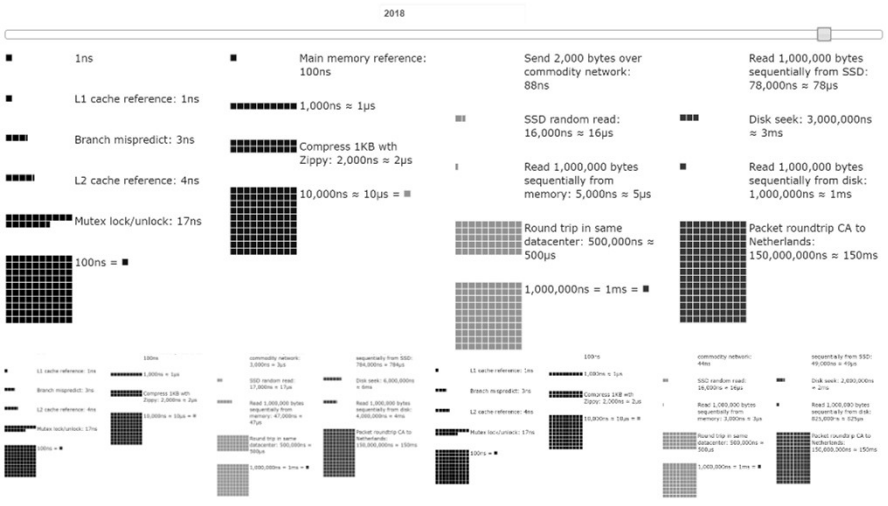
http://www.mgclab.com/

JS Lab

james@jslab.kr

I. 개요

❖ **Latency 고려 Core 설계** (Data Plane과 Control Plane의 위치 고려)



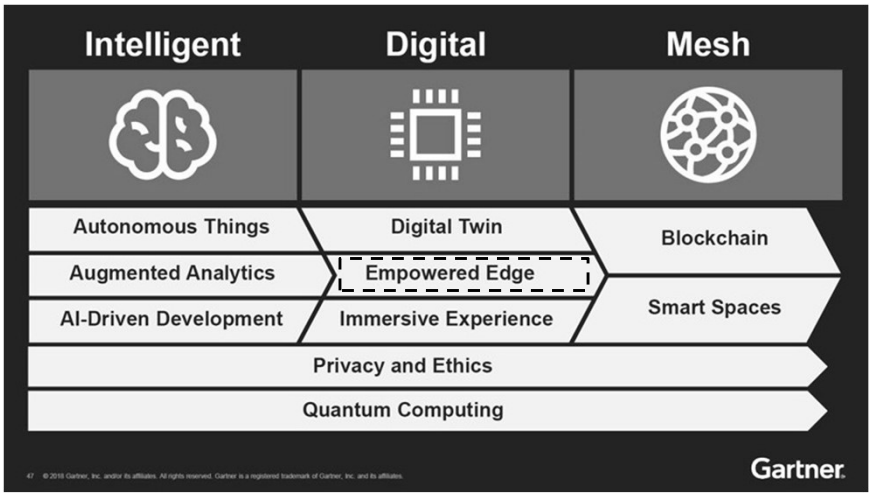
https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html

JS Lab

james@jslab.kr

I. 개요

❖ **Market:** Gartner Top 10 Strategic Technology Trends for 2019



https://www.pcmag.com/article/364429/gartners-top-10-strategic-technology-trends-for-2019

james@jslab.kr

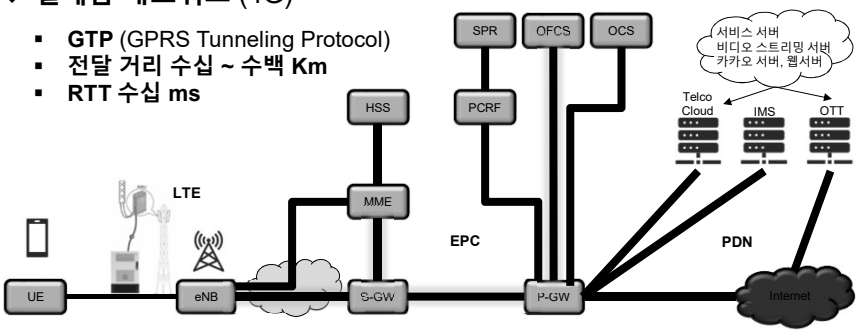
목차

james@jslab.kr

II. 현재 Telco 환경

❖ 텔레콤 네트워크 (4G)

- **GTP** (GPRS Tunneling Protocol)
- **전달 거리 수십 ~ 수백 Km**
- **RTT 수십 ms**



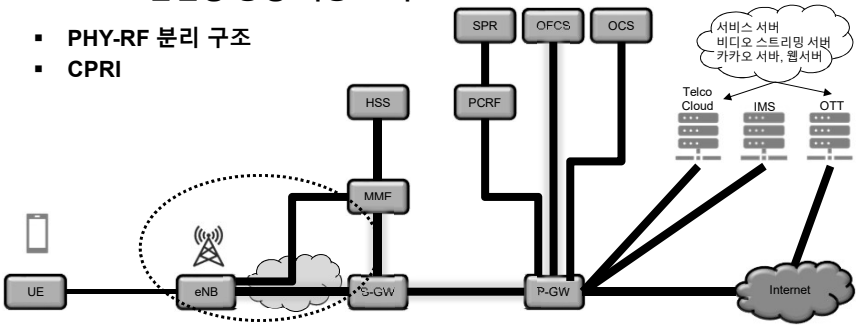
UE	S-GW	MME	PCRF	OFCS
▪ **User Equipment**: LTE칩, 안테나, USIM 사용 기기	▪ **Service-Gateway**: eNB간 핸드오버, 3GPP간 핸드오버를 위한 데이터 연결의 지역 모빌리티 앵커지점	▪ **Mobility Management Entity**: E-UTRAN 제어, 사용자 인증, UE 위치나 상태등 모빌리티 관리	▪ **Policy and Charging Rule Function**: UE 정책 결정과 PCC룰(QoS, charging Rule)을 P-GW에 제공	▪ **Offline Charging System**: P-GW에서 제공하는 UE별 오프라인 Charging 데이터를 관리
eNB	**P-GW**	**HSS**	**SPR**	**OCS**
▪ **Evolved Node B**: UE와 EPC간 무선 연결 제공 베이스 스테이션으로 제어/데이터 패킷의 암호화와 무결성 제공	▪ **PDN -Gateway**: UE의 IP주소/PDN 접속 제공과 QoS 강화 S-GW간 핸드오버 모빌리티 앵처지점으로 온라인 오프라인 과금	▪ **Home Subscriber Server**: 사용자 프로필 DB (IMSI, 인증키, QoS 등등 B/OSS에서 제공	▪ **Subscriber Profile Repository**: PCRF 데이터베이스로 B/OSS에서 제공하는사용자의 정책과 과금룰 유지	▪ **Online Charging System**: P-GW에서 제공하는 UE별 이벤트 기반의 온라인 Charging 데이터 데이터 사용량, 접속 시간 관리

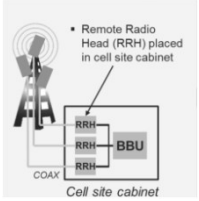
General Packet Radio Service (GPRS)

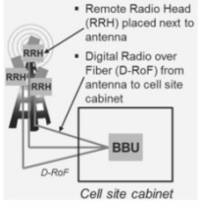
II. 현재 Telco 환경

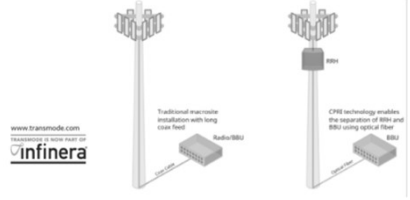
❖ Cell site 연결망 용량/비용 고려

- **PHY-RF 분리 구조**
- **CPRI**







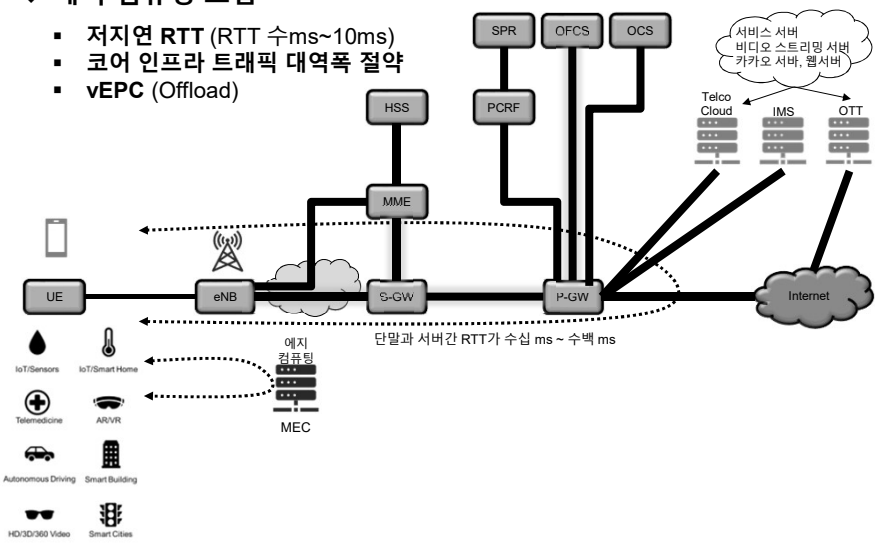


Remote Radio Head (RRH) Baseband Unit (BBU) Common Public Radio Interface (CPRI)

II. 현재 Telco 환경

❖ 에지 컴퓨팅 도입

- **저지연 RTT** (RTT 수ms~10ms)
- **코어 인프라 트래픽 대역폭 절약**
- **vEPC** (Offload)



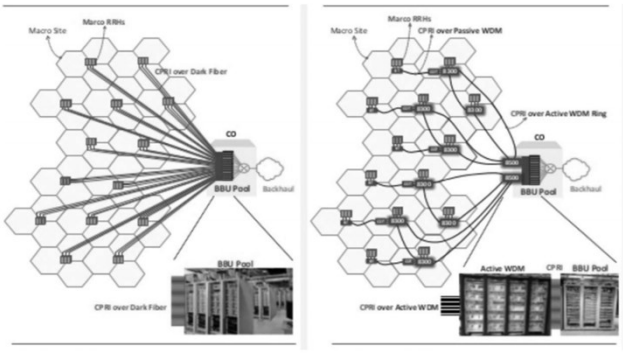
james@jslab.kr

MEC (Mobile|Mutli-access Edge Computing)

II. 현재 Telco 환경

❖ 통신사업자의 망 구조 고려

- **중앙집중 CO 구조 시 SPOF 존재로 모든 하단 RRH의 단절 가능**
- **라스트 마일 우회로 확보 필요** (예: Ring 구조)
- **링 구조와 듀얼홈잉 구조로 광 선로 장애와 국사 장애를 해결 가능**

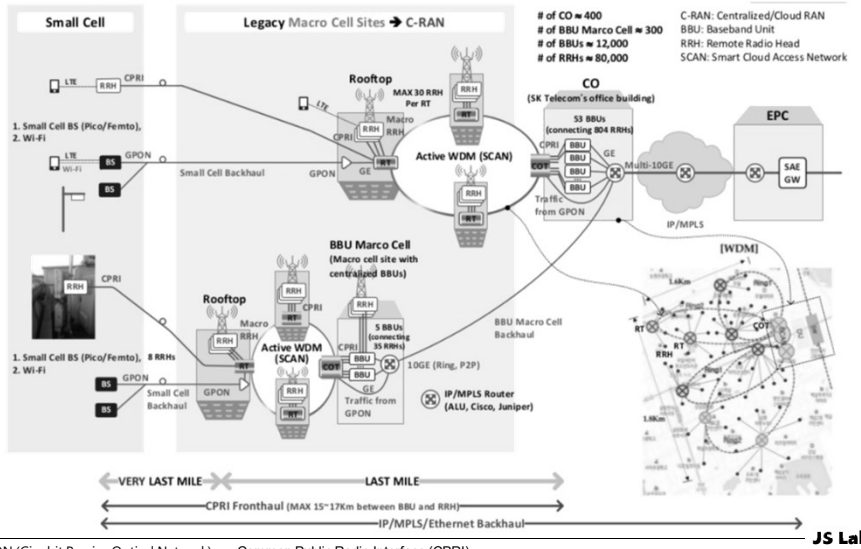


james@jslab.kr

Remote Radio Head (RRH) Baseband Unit (BBU) Common Public Radio Interface (CPRI)

II. 현재 Telco 환경

❖ **현재 통신사업자의 망 구성** (4G SKT 예) – by Netmanias

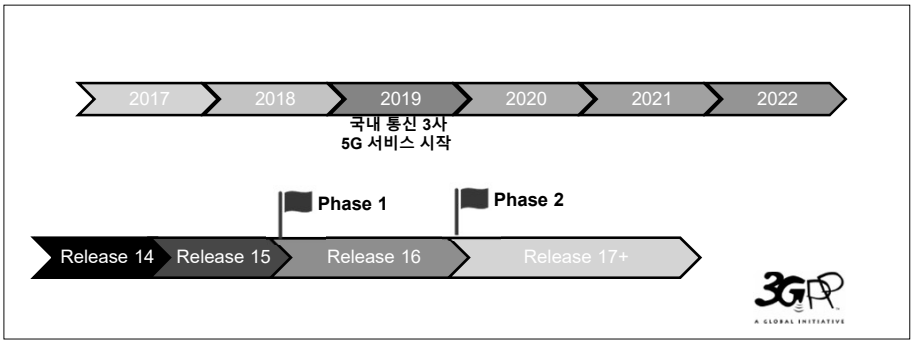


GPON (Gigabit Passive Optical Network) Common Public Radio Interface (CPRI)

II. 현재 Telco 환경

❖ **5G 표준**

- **Phase 1** (3GPP Rel. 15, 2018년 6월)
- **Phase 2** (3GPP Rel. 16, 2019년 12월)
- **3GPP Rel. 17은 5G 개선** (2020년 시작)
- **국내 통신 3사 5G 서비스 시작** (2019년)
- **표준 적용은 대개 18개월정도 예상**

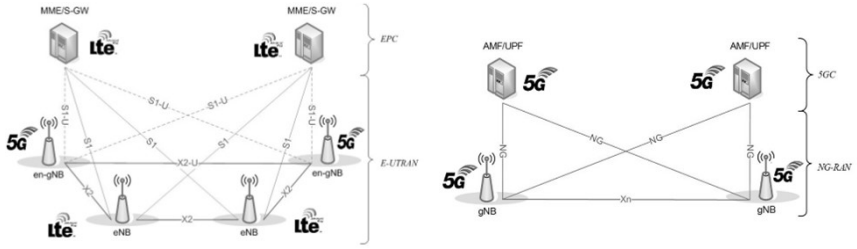


II. 현재 Telco 환경

❖ **5G 아키텍처 고려**

- **독립형(SA)**: Standalone Architecture
- **비독립형(NSA):** 기존 4G LTE/EPC 연동 Non-Standalone Architecture
- **5G 서비스를 위한 아키텍처 구성은 NSA 에서 SA로 점차 변화 예상**

james@jslab.kr



NSA Architecture **SA Architecture**

EPC (Evolved Packet Core E-UTRAN (Evolved Terrestrial Radio Access Network) LTE (Long-Term Evolution)

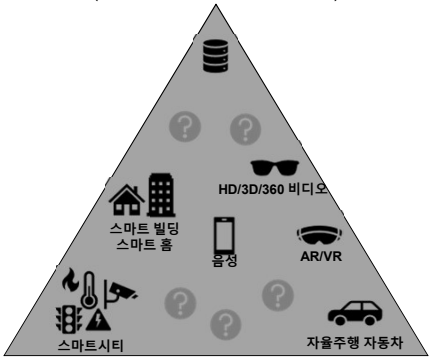
II. 현재 Telco 환경

❖ **5G 서비스 시나리오**

- **eMBB**(대역폭 개선), **mMTC**(기기 종류 및 수량 증가), **URLLC**(초저지연)

james@jslab.kr

eMBB: 대역폭 10배 이상 증가)
(enhanced mobile broadband)

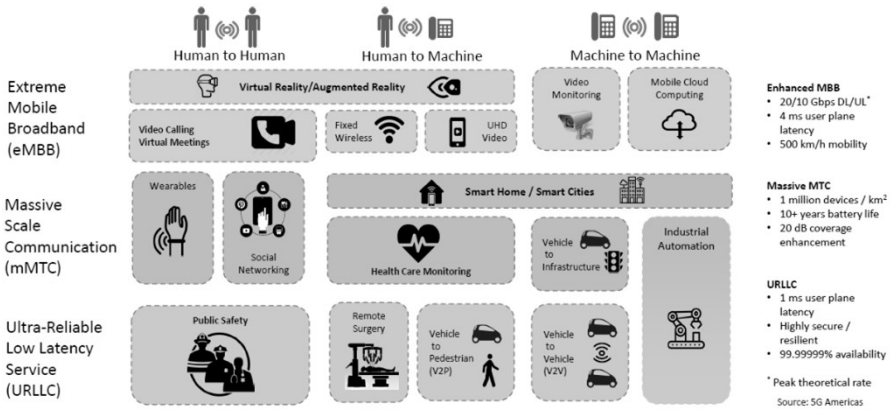


mMTC: 지원기기 종류/수량 대폭증가
(massive machine-type communication)

URLLC: 고신뢰 초저지연
(ultra-reliable low latency)

II. 현재 Telco 환경

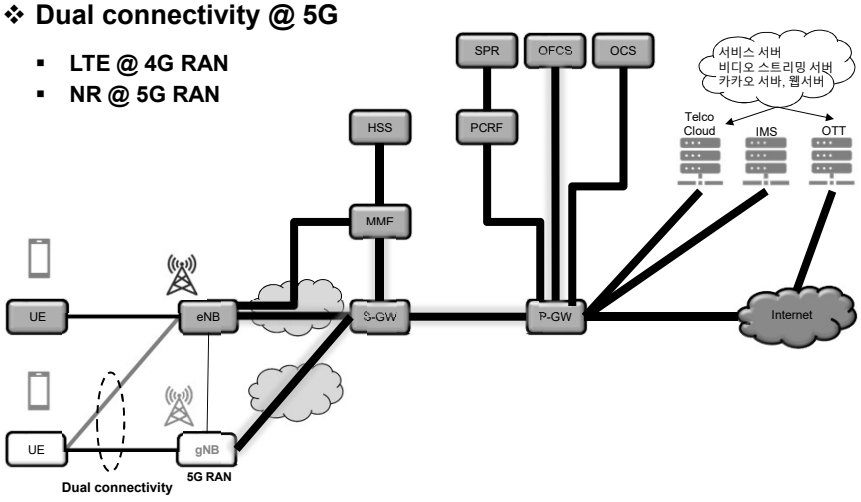
❖ **5G 서비스 시나리오** ('3G4G'의 Use Case 예)



II. 현재 Telco 환경

❖ **Dual connectivity @ 5G**

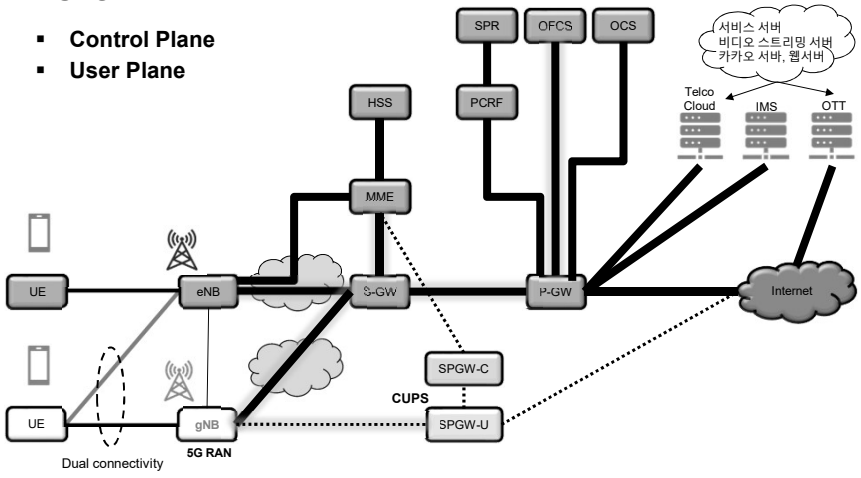
- **LTE @ 4G RAN**
- **NR @ 5G RAN**



II. 현재 Telco 환경

❖ **CUPS**

- **Control Plane**
- **User Plane**

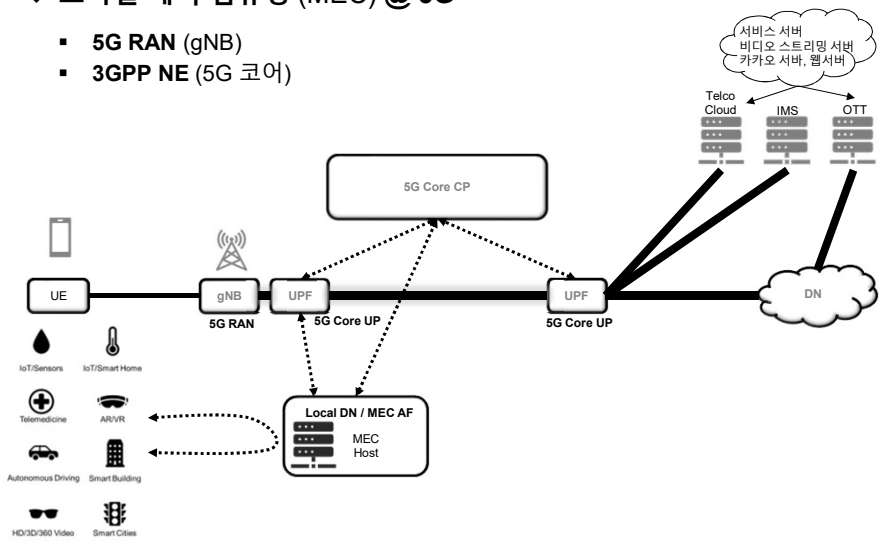


CUPS (Control and User Plane Separation)

II. 현재 Telco 환경

❖ **모바일 에지 컴퓨팅** (MEC) **@ 5G**

- **5G RAN** (gNB)
- **3GPP NE** (5G 코어)

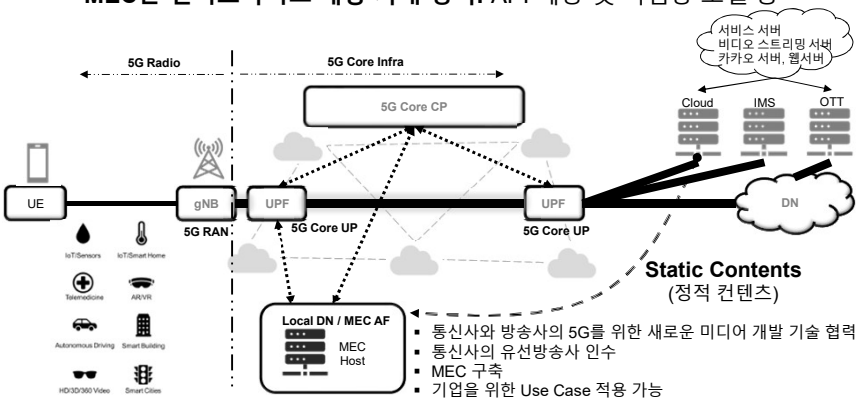


network element (**NE**)

II. 현재 Telco 환경

❖ **MEC** (Mobile | Multi-access Edge Computing)

- **에지의 데이터센터 기술 도입:** 국사/기지국의 데이터센터화
- **Phase 1을 위한 서비스:** 정적 컨텐츠를 위한 vCDN등
- **Phase 2를 위한 서비스:** Dynamic Contents 등의 서비스에 유리
- **MEC는 엔터프라이즈 개방 기대 영역:** API 제공 및 기업용 모델 등



- 통신사와 방송사의 5G를 위한 새로운 미디어 개발 기술 협력
- 통신사의 유선방송사 인수
- MEC 구축
- 기업을 위한 Use Case 적용 가능

RAN (Radio Access Network) MEC (Mobile Edge Computing / Multi-access Edge Computing) AF (Application Function)

II. 현재 Telco 환경

❖ **통신사의 5G 사업 모델**

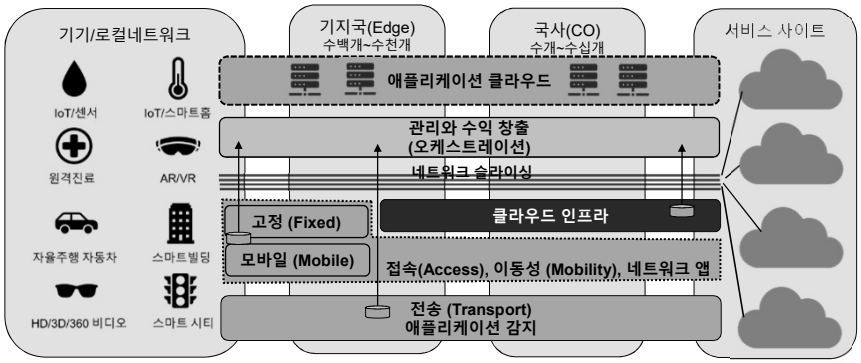
❖ **AT&T** (예)

- **제품과 서비스** (MEC는 K8s 기술 적용 영역)
- **API 공개** (Flow Designer, M2X, Video Optimizer)

II. 현재 Telco 환경

❖ **5G 코어 인프라(Core Infra)**

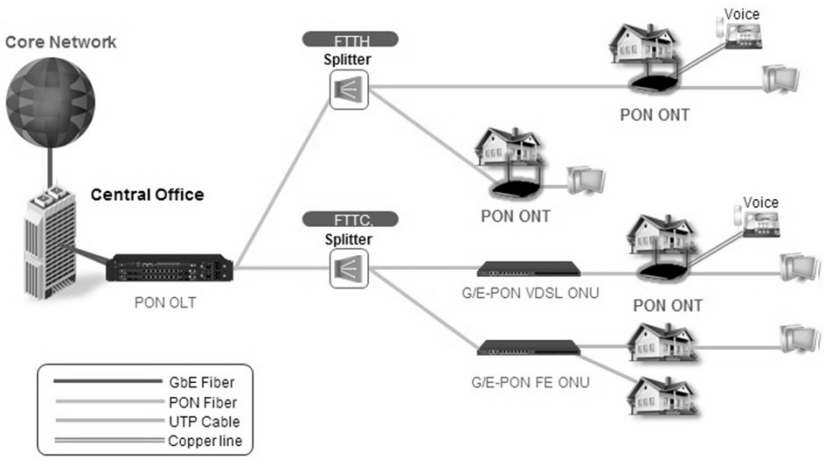
- **Edge(기지국)외 Central Office(국사)의 데이터센터 화**
- **클라우드 네이티브화** (애플리케이션 서비스, 관리, 인프라)
- **네트워크 슬라이싱** (Network Slicing)
- **클러스터링 확장성 고려** (갯수등)



II. 현재 Telco 환경

❖ **브로드밴드**

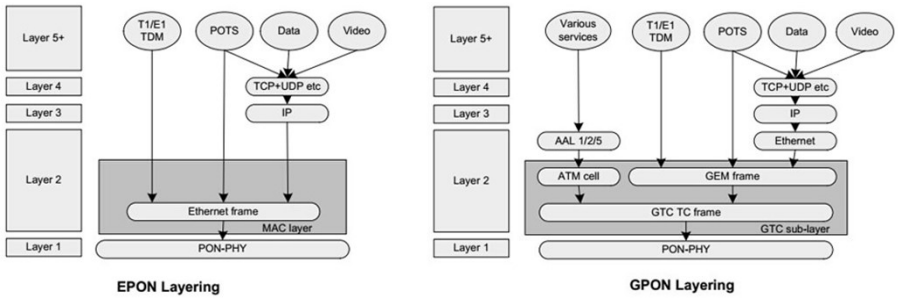
❖ **OLT, ONU, ONT**



Optical Line Terminal (OLT) Optical Network Unit (ONU) Optical Network Terminal (ONT)

II. 현재 Telco 환경

- **Passive Optical Networks** (PON)
- **EPON** (KT) **vs GPON** (SKT)



james@jslab.kr

EPON (Ethernet Passive Optical Network) GPON (Gigabit Passive Optical Network)

II. 현재 Telco 환경

- **5G Data Plane**
- **UPF** (User Plane Function)

5G: Data Plane - UPF [3GPP TS 23.501: System Architecture for the 5G System] [IETF] o-existence of 3GPP 5GS and Identifier Locator Separation Solution

- **5G**에서는 다양한 위치에서 응용 서비스를 제공할 수 있도록 **UPF(User Plane Function)을 분산배치** 가능 => MEC 지원이 용이하게 함
- **UPF** (User Plane Function): **UL CL를 이용하여 두 경로로 트래픽을 분리시킬** 수 있다 => UL CL는 Traffic Filter에 매칭되면(업링크 매 패킷의 Inner IP 패킷의 목적지 IP 주소가 Local DN에 할당된 주소이면) GTP 디캡하여 Local DN으로 IP 라우팅시키고, 그렇지 않으면 Next Hop UPF로 전달함. 다운링크는 Local DN에서 유입되는 IP 패킷을 GTP 인캡하고 cUPF에서 내려오는 GTP 패킷들과 Merge하여 gNB로 보냄.
- **UL CL** (Uplink Classifier): **Inner 패킷의 ScrIP 또는 DstIP 등을 보고** 트래픽을 Steering 해주며 Traffic filter은 SMF로부터 받음

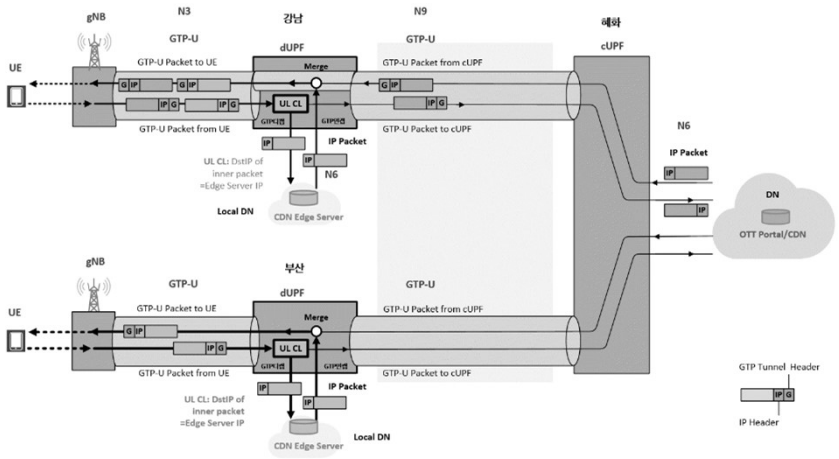
Netmanias

Intel

james@jslab.kr

II. 현재 Telco 환경

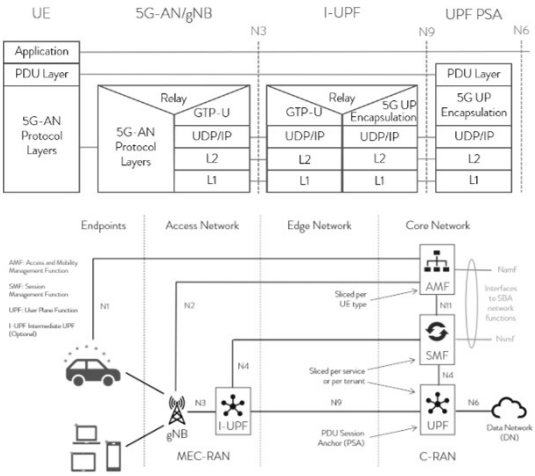
❖ **5G Data Plane**
❖ **UPF는 Edge를 위해 경로에 따라 GTP decapsulation 가능**



https://netmanias.com/ko/post/oneshot/14003/5g-mec/5g-data-plane-upf-user-plane-function

II. 현재 Telco 환경

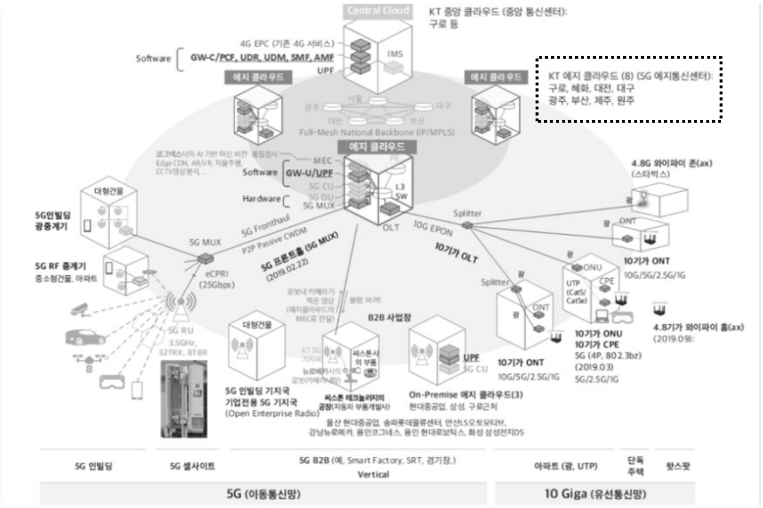
❖ **UPF 예:** Metaswitch Networks



https://www.metaswitch.com/knowledge-center/reference/what-is-the-5g-user-plane-function-upf

II. 현재 Telco 환경

❖ **통신사업자의 망 구성** (KT 5G/OLT 예) – by Netmanias



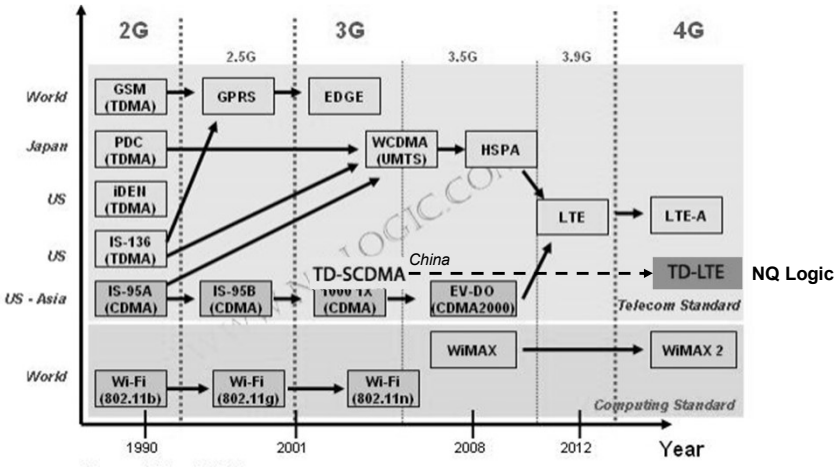
https://www.netmanias.com/ko/?m=view&id=oneshot&no=14155 Optical Line Termination (OLT)

JS Lab

james@jslab.kr

II. 현재 Telco 환경

❖ **모바일 기술의 발전**
❖ **1G는 FDMA 사용**



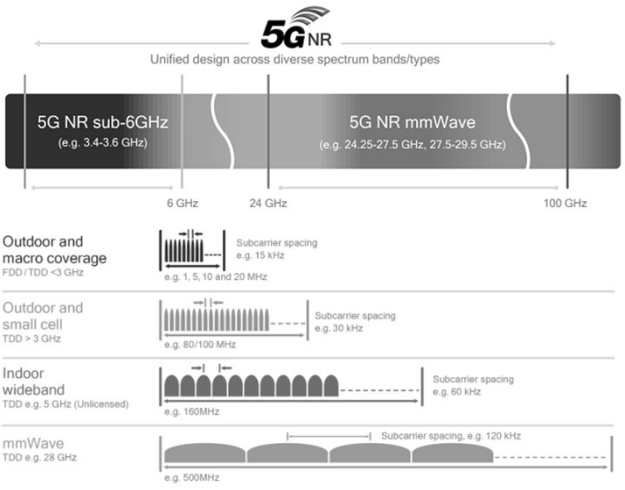
Frequency division multiple-access (**FDMA**); Time division multiple-access (**TDMA**); Code division multiple-access (**CDMA**)

http://arjoena27-te.blogspot.com/2015/09/the-evolution-of-mobile-communication.html

JS Lab

II. 현재 Telco 환경

❖ **5G NR:** Unlicensed spectrum, Millimeter wave, Massive MIMO, Dynamic TDD technology, Small Cell



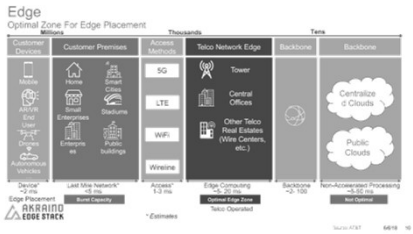
NR (New Radio)

james@jslab.kr

II. 현재 Telco 환경

❖ **MEC의 에지 접속 기술 환경 고려**

- **Base Stations, including mobile base stations, cell towers, central office base stations**
- **RAN for LTE/5G**
- **Radio network controller for WiFi**
- **Cable modem termination systems (CMTS) for cable**
- **PON OLT for fiber or the access points for other networks such as Zigbee, CBRS, LoRA, DSL, MuLTEfire, private LTE.**
- **Hot spots**
- **Small cells**
- **Data centers** (and micro-data centers)
- **Routers**
- **Switches**
- **WiFi access points**



james@jslab.kr

II. 현재 Telco 환경

❖ ETSI의 MEC PoC Projects

- **PoC#1:** "Video User Experience Optimization via MEC - A Service Aware RAN MEC PoC"
- **PoC#2:** "Edge Video Orchestration and Video Clip Replay via MEC"
- **PoC#3:** "Radio aware video optimization in a fully virtualized network"
- **PoC#4:** "FLIPS – Flexible IP-based Services"
- **PoC#5:** "Enterprise Services"
- **PoC#6:** "Healthcare – Dynamic Hospital User, IoT and Alert Status management"
- **PoC#7:** "Multi-Service MEC Platform for Advanced Service Delivery"
- **PoC#8:** "Video Analytics"
- **PoC#9:** "MEC platform to enable low-latency Industrial IoT"
- **PoC#10:** "Service Aware MEC Platform to enable Bandwidth Management of RAN"
- **PoC#11:** "Communication Traffic Management for V2X"
- **PoC#12:** "MEC enabled OTT business"
- **PoC#13:** "MEC infotainment for smart roads and city hot spots"

james@jslab.kr

II. 현재 Telco 환경

❖ 요약: 5G 기반의 예상 비즈니스와 필요 기술 고려

- **핵심 성능 파라미터 KPI** (Key Performance Indicator)
- **애플리케이션**
- **분야별 시장**
- **비즈니스 모델**

james@jslab.kr

KPI	애플리케이션	시장	비즈니스 모델
• 10x bandwidth per connection • Low-ms latency • Five 9's reliability • 100% coverage • >10x connections • 50Mbps per connection everywhere • 1000x bandwidth/area • 10 year battery life • Reduction in TCO	• Enhanced Mobile BB • Connected vehicles • AR/VR • S-UHD/3D Video • Haptics/Sensing • Massive IoT • Remote machine control • Mission critical services • Fixed-wireless access • …	• Consumer • Auto industry • Health • Industry 4.0 • Agriculture • Smart City/Public sector • Smart building • Utilities • Education • Transport • …	• B2C • B2B • B2B2C

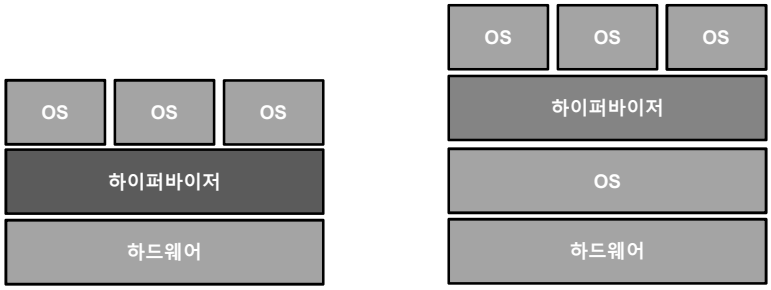
목차

james@jslab.kr

III. 가상화 / 클라우드

❖ 하이퍼바이저(Hypervisor)

① 컴퓨터 안에서 다수의 운영체제(OS)를 동시에 실행하기 위한 논리적인 플랫폼
② VMM(Virtual Machine Monitor/Manager)라고도 불림
③ 네이티브(Native) 하이퍼바이저(Type1)
④ 호스티드(Hosted) 하이퍼바이저(Type 2)



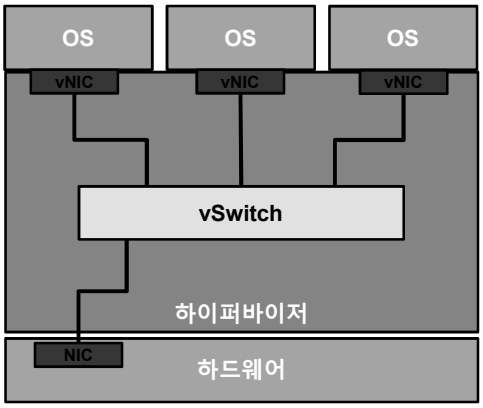
Type1 (Native/Bare metal)　　Type2 (hosted)

james@jslab.kr

JS Lab

III. 가상화 / 클라우드

- ❖ vNIC(Virtual Network Interface Card)
- ❖ vSwitch(Virtual Switch)



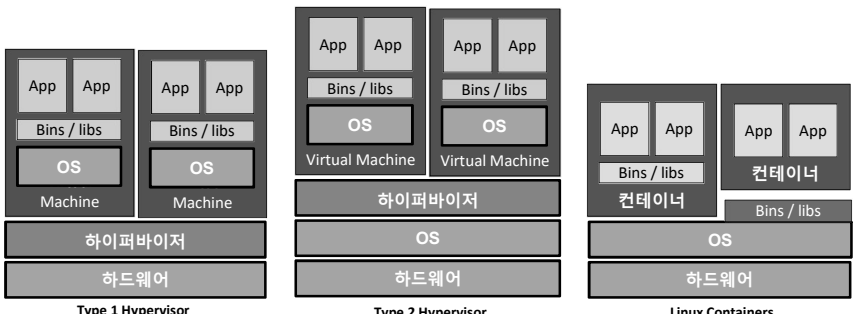
III. 가상화 / 클라우드

- ❖ 컨테이너

① 컨테이너의 어플리케이션을 독립적으로 유지

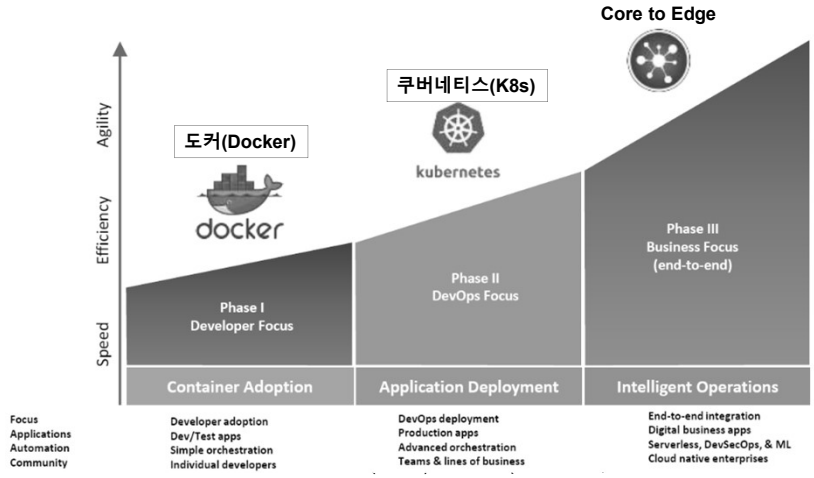
② 특징

- 가상머신보다 적은 자원 사용
- 스케일링을 가능하게 해주는 관리 도구 포함



III. 가상화 / 클라우드

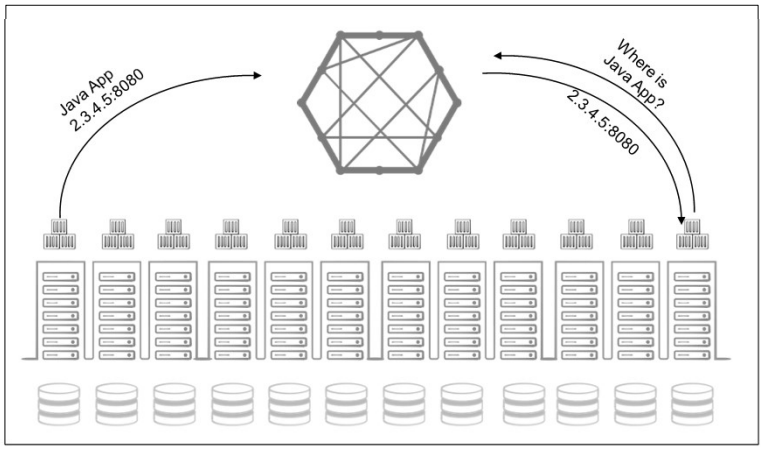
❖The Cloud Native Journey



III. 가상화 / 클라우드

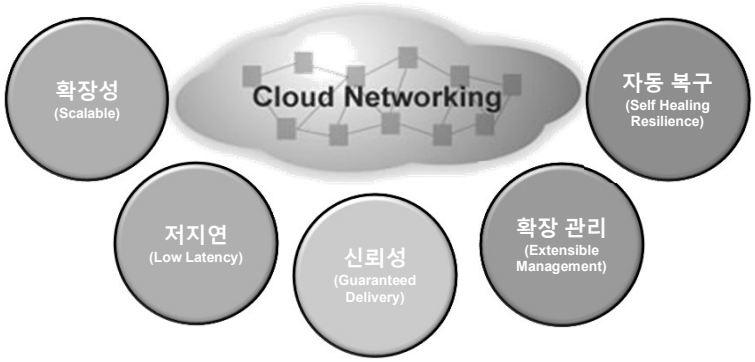
❖ 서비스 디스커버리 (Service Discovery)

- 앱들과 인프라가 자동으로 서로의 필요 부분을 찾아 연결
- 애플리케이션 서비스 트래킹을 지속하여 연결하여 사용



III. 가상화 / 클라우드

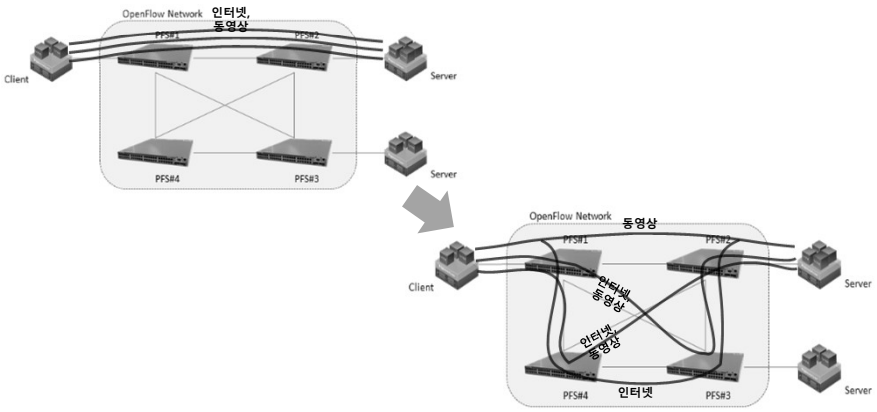
❖ 클라우드 네트워킹 이슈



III. 가상화 / 클라우드

❖ 가상 네트워크 구성

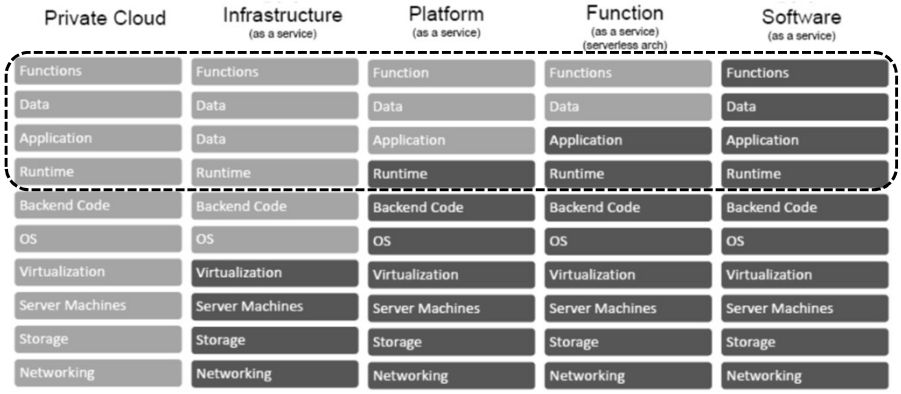
① 물리 네트워크
② 가상 네트워크



III. 가상화 / 클라우드

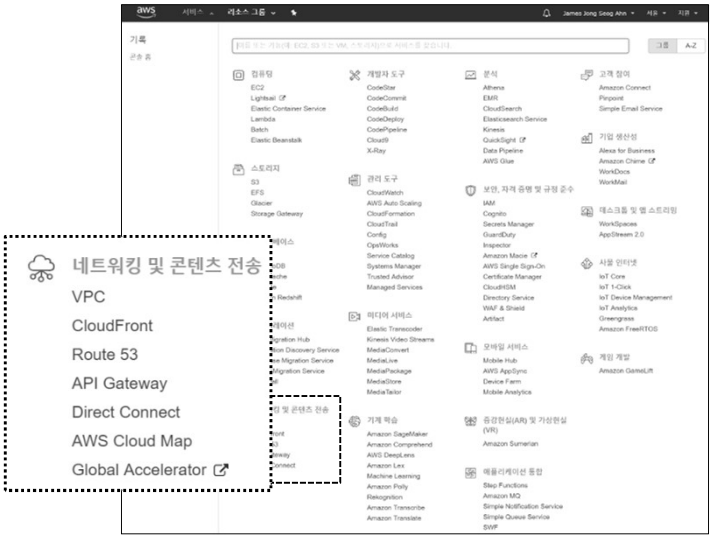
❖ 클라우드 서비스 비교

- **Private Cloud**
- **Public Cloud:** IaaS, PaaS, FaaS(Serverless), SaaS
- **Hybrid Cloud, Multi Cloud**



III. 가상화 / 클라우드

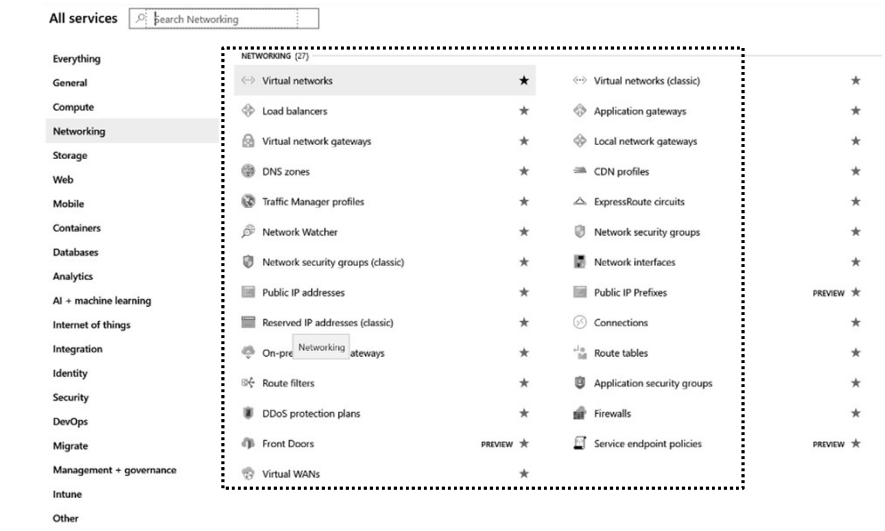
❖ 아마존 AWS



https://ap-northeast-2.console.aws.amazon.com/console/home?region=ap-northeast-2

III. 가상화 / 클라우드

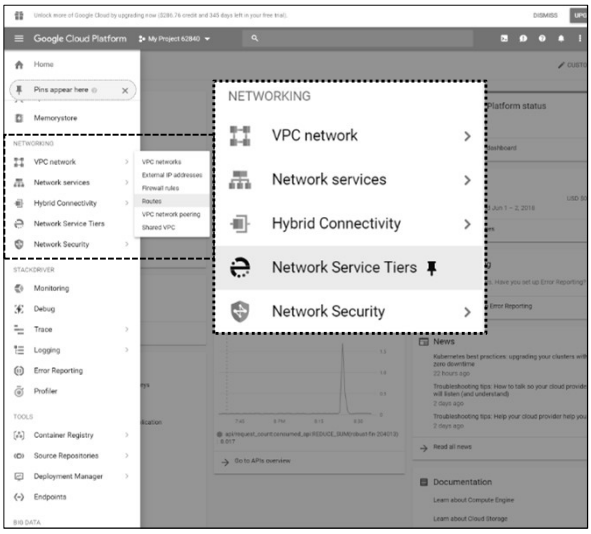
❖ 마이크로소프트 Azure



https://portal.azure.com

III. 가상화 / 클라우드

❖ Google Cloud Platform



https://console.cloud.google.com/home/dashboard?project=robust-fin-204013

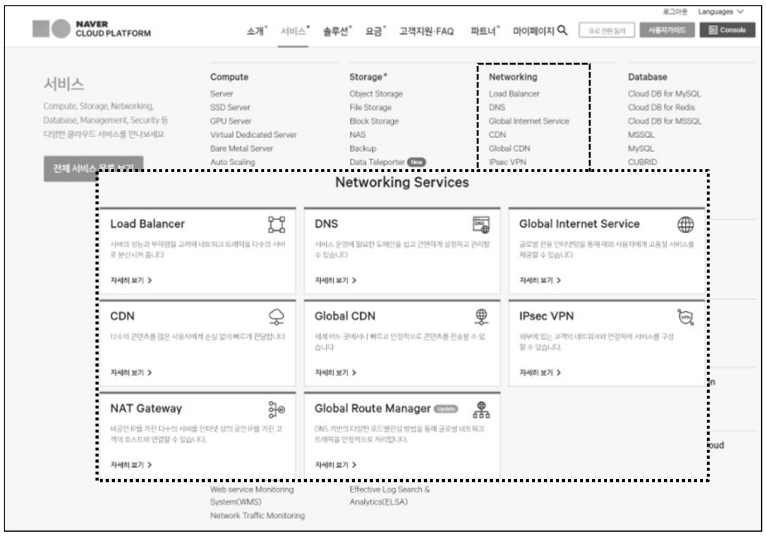
III. 가상화 / 클라우드

❖ **IBM Cloud**

https://console.bluemix.net/catalog/

III. 가상화 / 클라우드

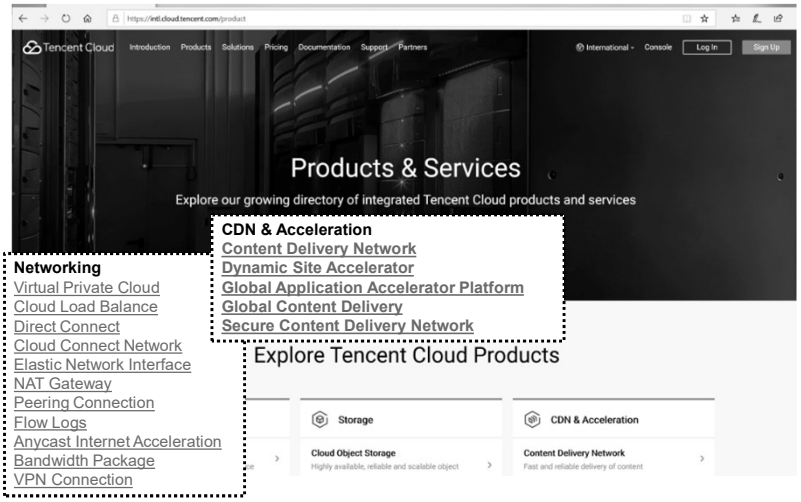
❖ **Naver Cloud Platform**



https://www.ncloud.com/

III. 가상화 / 클라우드

❖ **Tencent Cloud**

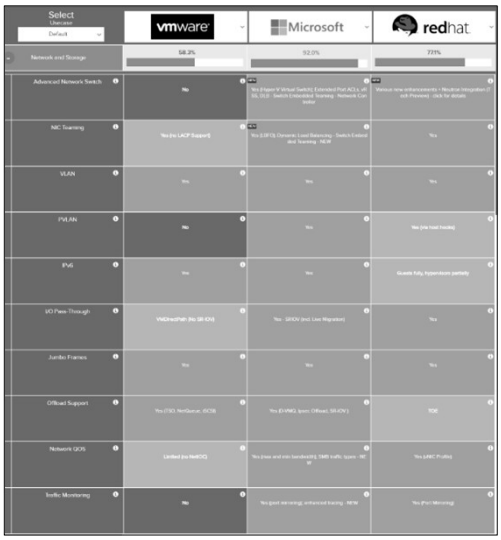


https://intl.cloud.tencent.com/

james@jslab.kr

III. 가상화 / 클라우드

❖ **프라이빗 클라우드 비교**



https://www.whatmatrix.com/comparison/Virtualization

james@jslab.kr

III. 가상화 / 클라우드

- ❖ **클라우드 서비스 기술에 적용하는 아키텍처 비교** (Agility, Deployment, Testability, Scalability, Performance, Simplicity)
- ❖ **User Plane – Control Plane 분리 : SDN Architecture**

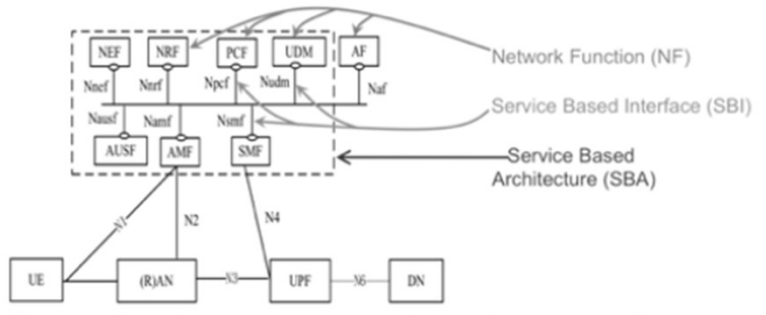
Architecture / Criterion	SOA	μ-service architecture	SBA	Monolithic
Agility	Low	High	Medium	Low
Deployment	Low	High	Medium	Low
Testability	Low	High	Medium	Medium
Scalability	Medium	High	Medium	Low
Performance	Low	Medium	Medium	High
Simplicity	Low	Medium	Medium	High

james@jslab.kr

Cloud Native White Paper, 5G-PPP Software Network Working Group, https://5g-ppp.eu/

III. 가상화 / 클라우드

- ❖ **SBA** (Service Based Architecture)
- ❖ **3GPP는 표준 Rel.15에서 Control Plane에서 적용하는 것으로 도입**



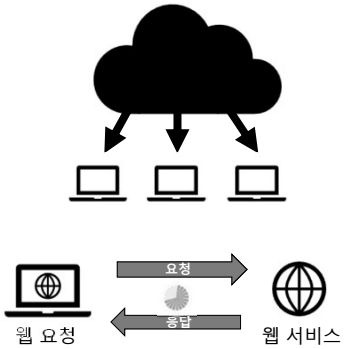
NEF	Network Exposure Function	AMF	Access &Mobility Management Function
NRF	Network Repository Function	SMF	Session Management Function
PCF	Policy Control Function	UE	User Equipment
UDM	Unified Data Management	(R)AN	(Radio) Access Network
AF	Application Function	UPF	User Plane Function
AUSF	Authentication Server Function	DN	Data Network

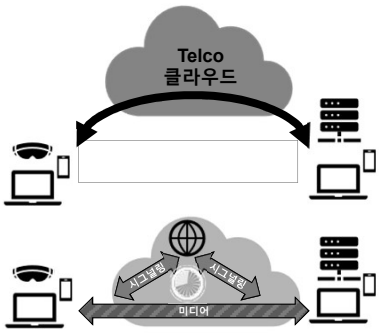
james@jslab.kr

III. 가상화 / 클라우드

❖ **5G 코어 인프라의 클라우드화**

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



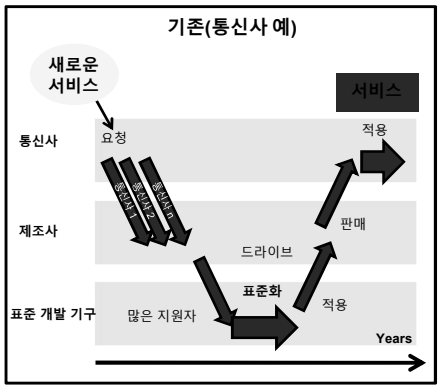


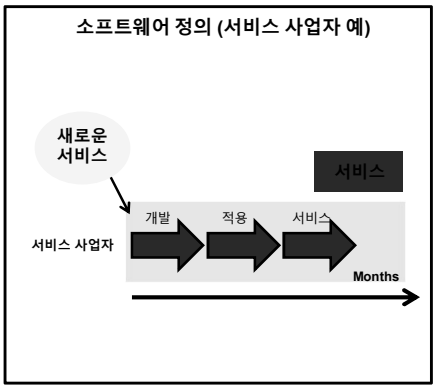
실시간 멀티미디어 애플리케이션
(고화질 스트리밍, 고화질 게임, AR, VR 등을 위한 경로)

james@jslab.kr

III. 가상화 / 클라우드

❖ **통신사와 서비스 사업자의 소프트웨어 정의**
❖ **개발능력 미보유 엔터프라이즈는 서비스 제공 가능 오픈소스나 제조사의 SDx 솔루션 필요**





james@jslab.kr

III. 가상화 / 클라우드

❖ 오케스트레이션 비교: Edge vs Central Cloud

	Edge (에지) 클라우드	중앙 클라우드
App의 위치	노드의 물리적 위치에서 중요한 서비스	비교적 위치와 독립적인 서비스
워크로드의 이동성	워크로드가 노드간 이동	클라우드 노드 장애 이외에는 비교적 고정
워크로드의 역동성	다양한 App들이 다양한 시간에 크게 다른 요구를 함	서비스를 적용하면 대부분의 시간에 안정적인 워크로드
아키텍쳐	다른 형태의 많은 수의 노드와 다양한 용량과 기술	대부분 동일하며 차이가 작음 (예: AWS, OpenStack, Azure 등)
지연	지연과 거리는 종단 사용자들을 위한 주요 역할	대부분 지연에 민감하지 않음
자원 가용성	에지노드는 작고, App을 위한 자원의 가용성을 보장하지 않음	가용성 확보는 중요하며 주요 기능 중 1개

james@jslab.kr

III. 가상화 / 클라우드

❖ 모니터링 비교: Edge vs Central Cloud

	Edge (에지) 클라우드	중앙 클라우드
분산 데이터 수집	네트워크 내의 분산된 수천 노드에서 수집이 필요	중앙 DB에 모든 것을 수집
아키텍처	다양한 제조사의 에지로 구성 되어 각자의 구조와 API 를 제공	클라우드 자체 수집과 모니터링을 제공 (예 AWS CloudWatch)
원인 근본 분석	분산 환경 서비스에 심각한 영향을 줄 수 있음	복잡 할 수 있으나 에지 네트워크 보다는 단순함
Closed Loop Control	지연과 거리는 장애복구와 마이그레이션 계획에 주요 역할	장애 복구는 비교적 단순하여 대부분 다른 인스턴스를 생성함

james@jslab.kr

III. 가상화 / 클라우드

❖ **데이터 관리 비교: Edge vs Central Cloud**

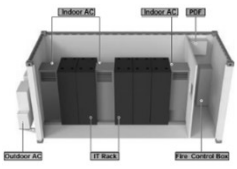
	Edge (에지) 클라우드	중앙 클라우드
트랜잭션 지원	모든 트랜잭션 DB는 분산화와 에지가 어려움	SQL DB등과 같이 모든 것을 한 곳에 위치
고 가용성	DB 복제는 대부분 실용적이지 않음	HA 솔루션을 기술적 문제로 인식하지 않음
지연	지연으로 인해 중앙 클라우드의 DB를 사용하지 못하여 App이 동작하는 지역에 DB가 필요함	App이 DB 가까이 위치하여 지연 이슈 없음
데이터 이동성과 가용성	모든 노드가 분산되어 있는 데이터를 접속하는 환경	이슈 없음

james@jslab.kr

III. 가상화 / 클라우드

❖ **Edge Cloud Computing의 Use Case 고려 사항** (OpenStack)

- **데이터 수집과 분석**
- **보안**
- **준수**
- **NFV** (Network Function Virtualization)
- **실시간 처리**
- **몰입** (Immersive)
- **네트워크 효율**
- **자동화 관리**
- **프라이버시**



james@jslab.kr

JS Lab

III. 가상화 / 클라우드

❖ **요약:** Edge Cloud Computing의 필요 기술

- **VM/컨테이너/베어메탈 관리** (구성, 스케듈, 적용, 대기, 재시작, 셧다운 등)
- **이미지 관리** (VM, 컨테이너)
- **네트워크 관리** (VM/컨테이너의 인프라 연결, 사용자를 위한 외부 연결)
- **스토리지 관리** (에지 애플리케이션을 위한 스토리지 서비스)
- **관리 도구** (분산 인프라를 위한 관리자 운영 인터페이스 등)
- **WAN 경유 시 스토리지 지연 고려** (지연 값 확인/고려 필요)
- **에지 보안 강화** (물리와 앱의 통합 사이트 모니터링, 필요시 제어 필요)
- **오케스트레이션 도구** (많은 사이트들의 통합, 제어 플레인의 피어링 "self organizing edge")
- **에지 플랫폼을 위한 오케스트레이션의 페더레이션**
- **동기화** (코어와 연결이 단절되는 것을 고려한 전달의 추상화)
- **네트워크 파티셔닝 이슈** (짧거나 긴 단절의 경향)
- **에지앱 라이프사이클 관리 도구** (지연 민감 앱 스케듈링의 'constraints 배치', provisioning/scheduling of applications, 내/외부 이벤트에 따른 Use Case와 성능 고려 재배치)
- **위치 감지 통합**
- **제한된 하드웨어 자원 고려 거시적 설계**

목차

IV. 컨테이너

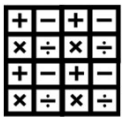
❖Trends – App Dev / Infrastructure

- Cloud-Native (public and private clouds)
- Containers
- Micro-service Architecture

james@jslab.kr

Cloud-Native
(public and private clouds)

컨테이너



MSA
(Micro-service Architecture)

Device-Native @ Telco Cloud

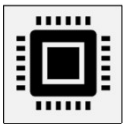
IV. 컨테이너

❖Application Platform 발전

- 베어메탈
- 가상머신
- 컨테이너
- 서버리스

james@jslab.kr

컨테이너



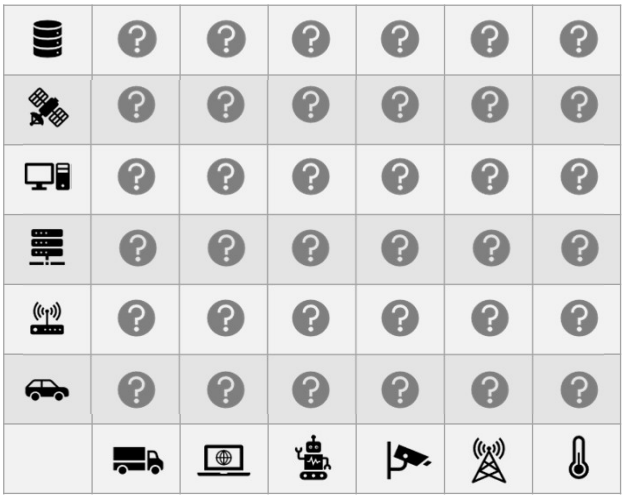
베어메탈

가상머신

서버리스

IV. 컨테이너

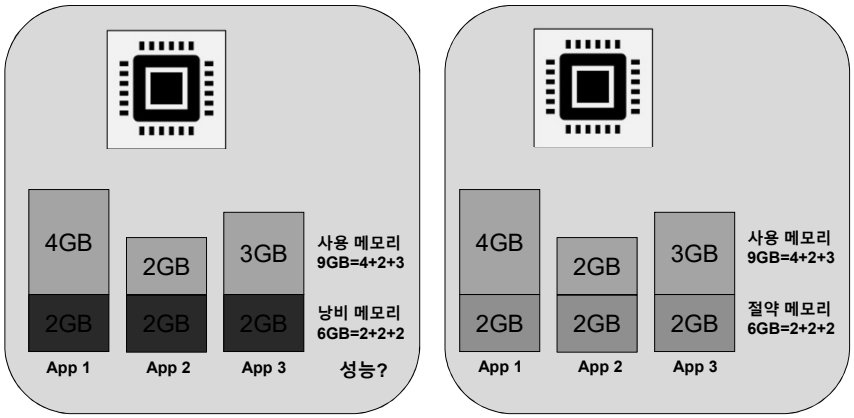
❖ Device Mesh 환경을 위한 Device Native 기술 필요



james@jslab.kr

IV. 컨테이너

❖가상머신과 컨테이너



james@jslab.kr

IV. 컨테이너

❖ **Fanless 하드웨어** (IoT Gateway)
❖ **오픈소스 기반**
❖ **SDN 기반 컨테이너 네트워킹 사용** (인증키 생성/서비스 배포/암호화 터널링 내장)

IoT Gateway 오픈 하드웨어 플랫폼 비교

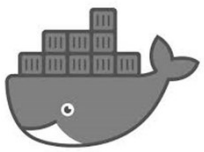
	Items	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6
IoT Gateway Hardware (Fanless)	CPU Type	Celeron J1900	ARM v7	ARM v7	ARM v6	Atom	Quark
	# of CPU Cores	4	4	4	1	2	1
	CPU Clock Speed	2 GHz	1.2 GHz	900 MHz	700MHz	500 MHz	400 MHz
	RAM	4 GB	1 GB	1 GB	512 MB	1 GB	256 MB
항목	Docker 설치	OK	OK	OK	OK	No	No
	Docker Show	OK	OK	OK	No	No	No
	Container 2 MB 구동	OK	OK	OK	No	No	No
	Container 784 MB 구동	OK	No	No	No	No	No
	Container Cluster Manager	OK	OK (동일H/W)	No	No	No	No
	Agent for UCP	Yes	No	No	No	No	No
	Alarm	Yes	Yes	Yes	Yes	Yes	N/A
	Sensor / Actuator	N/A	OK	OK	OK	OK	OK
	Local Logger	OK	OK	OK	OK	N/A	N/A
	OVS	OK	OK	OK	OK	No	No
	PoE 지원 (변환기 사용)	No	Yes(변환기)	Yes(변환기)	Yes(변환기)	No	Yes (전용 모듈)
	Wi-Fi 지원	Yes	Yes	Yes	Yes	Yes	Yes
	Bluetooth	Yes(동글)	Yes(내장)	Yes(내장)	Yes(동글)	Yes(내장)	N/A
	Flow Agent	Yes	N/A	N/A	N/A	N/A	N/A
	IDS	Yes	No	No	No	No	No
	보안 생태계 연동	Yes	Yes(저성능)	Yes(저성능)	N/A	N/A	N/A

james@jslab.kr

IV. 컨테이너

❖ **도커(Docker) 란?**

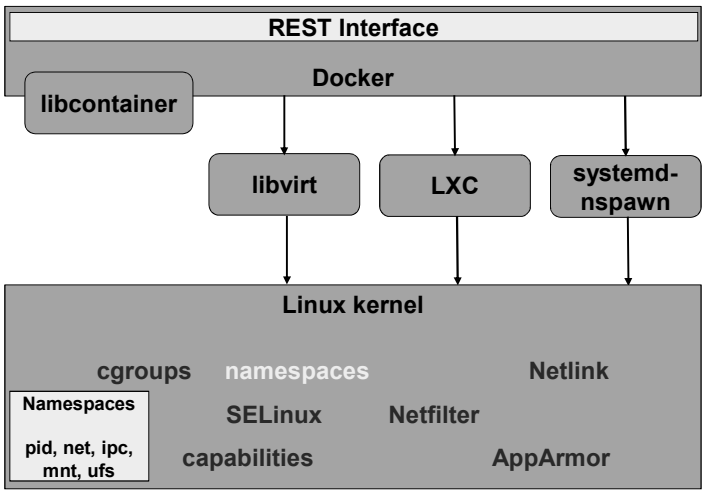
- **"Company"**
- **"Product"**
- **"Platform"**
- **"CLI Tool"**
- **"Computer Program"**



james@jslab.kr

IV. 컨테이너

❖How it works?



IV. 컨테이너

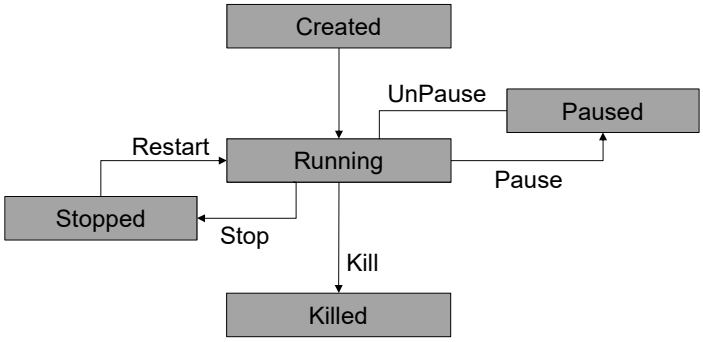
❖컨테이너는 리눅스 커널의 3가지 요소 기반하며, 리눅스 컨테이너(LXC)는 2008년에 chroot, cgroup, namespace를 조합하여 도입

- Chroot: jail로 알려져 있으며 1979년 유닉스부터 사용을 했으며 실행중인 프로세스가 사용하는 루트 디렉토리를 외부 디렉토리 트리의 children 에서 접속 할 수 없음
- Namespace: 2002년부터 리눅스 커널에서 사용을 했으며 애플리케이션이 운영 환경에서 네트워킹, 사용자 ID, 파일시스템, 프로세스 트리 등을 완전하게 독립하는 것을 허용
- Cgroup: 2008년부터 리눅스 커널에서 사용 했으며 CPU, 메모리, 디스크 I/O,네트워크 등의 자원을 분리하고 제한함

❖도커(Docker)는 컨테이너 생태계를 대중화 했으며 소프트웨어를 계층화 하는 Union File System(UFS)를 도입하고, 컨테이너 내의 애플리케이션 적용을 자동화

IV. 컨테이너

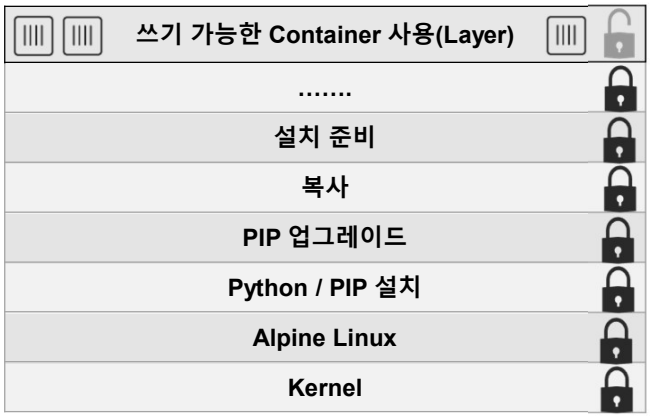
❖ **도커 컨테이너 라이프 사이클:** Build한 이미지(Image)는 컨테이너 실행 시 읽기 전용으로 사용하며 컨테이너에서 생성한 계층에서 쓰기와 읽기를 실행



IV. 컨테이너

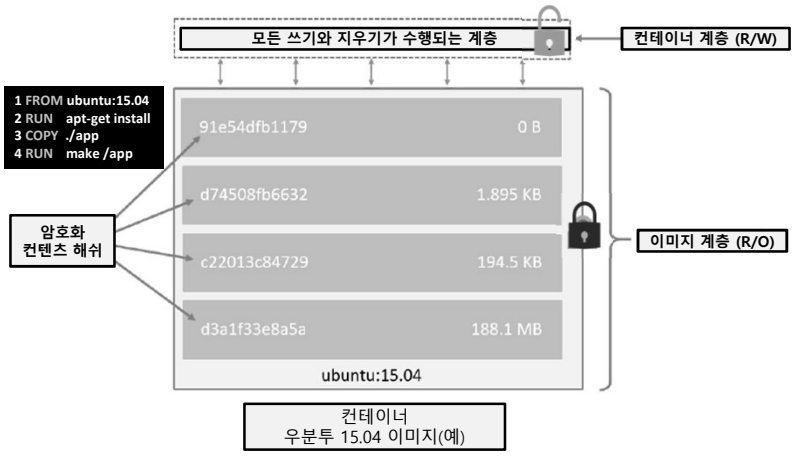
❖**이미지:** 도커 컨테이너의 기반이며, 앱을 표현함

❖**이미지 레이어**



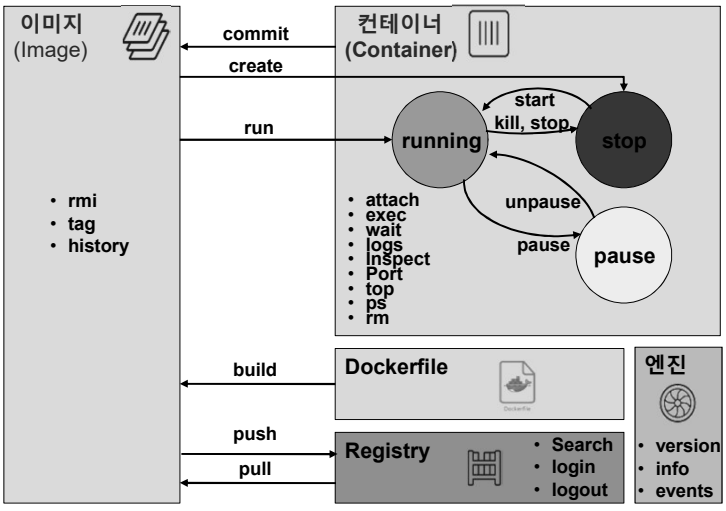
IV. 컨테이너

❖ **이미지 생성:** Build한 이미지(Image)는 컨테이너 실행 시 읽기 전용으로 사용하며 컨테이너에서 생성한 계층에서 쓰기와 읽기를 실행



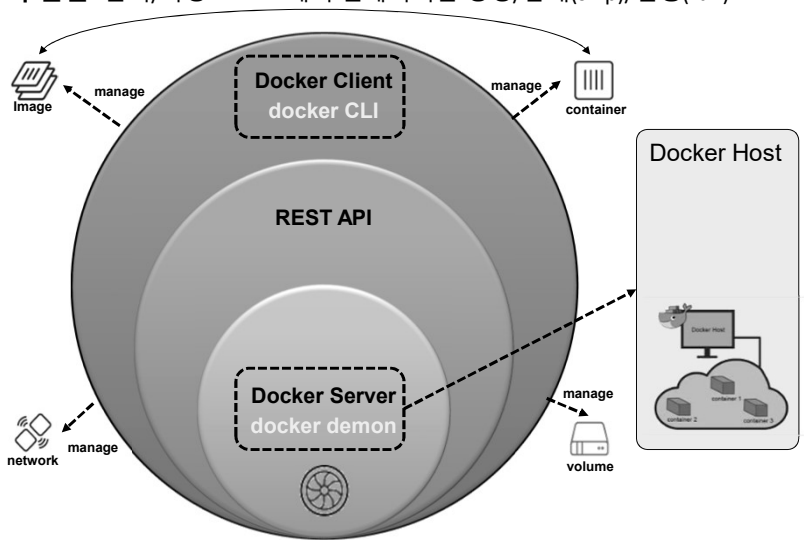
IV. 컨테이너

❖도커 명령어 다이어그램



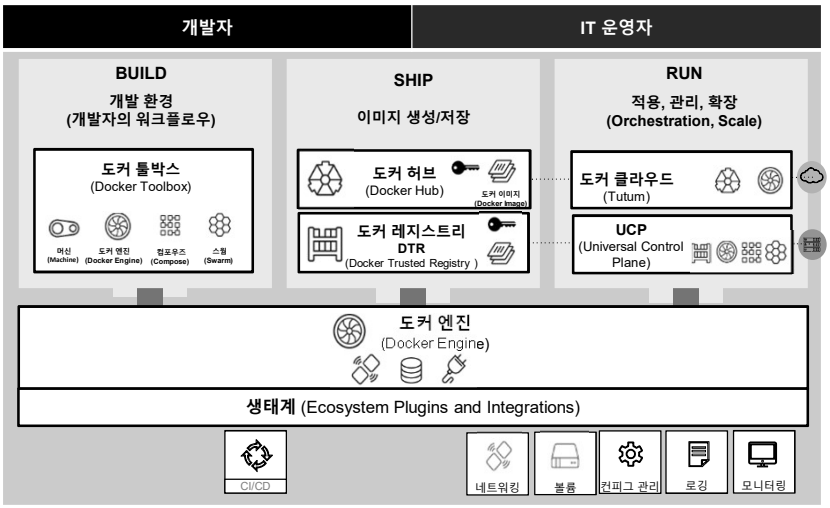
IV. 컨테이너

❖ **도커 엔진:** 물리/가상 호스트에서 컨테이너를 생성/탑재(ship)/실행(run)



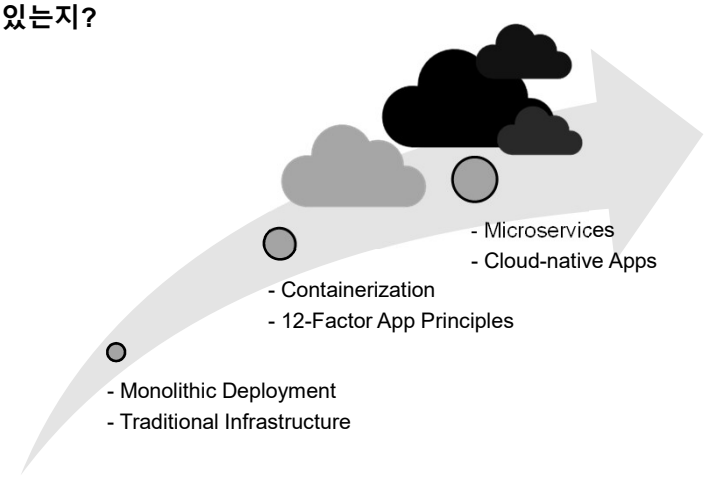
IV. 컨테이너

❖워크 플로우(Workflow)



IV. 컨테이너

❖현재 엔터프라이즈의 애플리케이션을 합리적인 노력으로 클라우드화 할 수 있는지?



james@jslab.kr

IV. 컨테이너

❖요약

james@jslab.kr

목차

james@jslab.kr

V. 오케스트레이션

❖ 클라우드 관리

	클라우드 관리 플랫폼 (Infrastructure First)	애플리케이션 플랫폼 (Application First Approach)	자동화/오케스트레이션 플랫폼 (Automation First)
장점	• Single Pane of Glass • 비용 분석과 제어	• 쿠버네티스(Kubernetes or K8s)와 컨테이너는 주요 클라우드에서 지원 • 애플리케이션과 마이크로서비스 중심	• 퍼블릭/프라이빗 클라우드등 다양하고 폭넓은 지원 • 모든 클라우드나 애플리케이션 지원 구조 • 커스터마이징 가능
단점	• 제한적인 애플리케이션 인식 • DevOps프로세스에 부적합 • 제한적인 클라우드 서비스	• 대부분 그린필드에 적합 • 다른 영역에 호환성 부족	• 애플리케이션과 클라우드 단위로 커스터마이징 필요
	RED HAT CLOUDFORMS, RIGHTSCALE CLOUD MANAGEMENT, IBM, vmware, MORPHEUS, SCALR	OPENSHIFT, RANCHER, CLOUDFOUNDRY, kubernetes	TERRAFORM, CLOUDIFY, ARIA, OASIS TOSCA

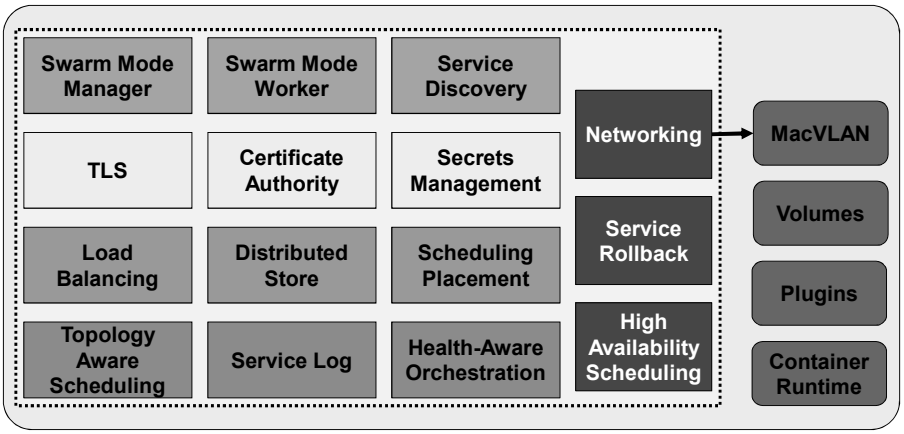
james@jslab.kr

JS Lab

V. 오케스트레이션

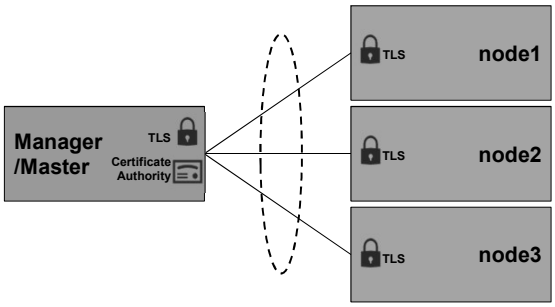
❖ **오케스트레이션 요소:** Docker Swarm Mode 예 (17.06)
❖ **Docker는 Swarm과 함께 Kubernetes를 적극 수용**

오케스트레이션 요소



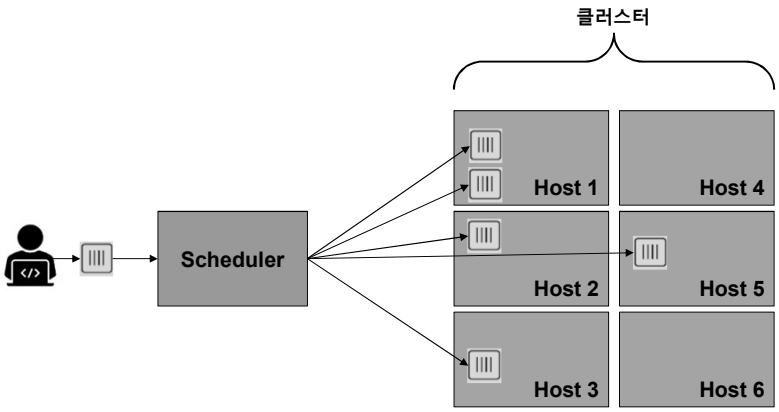
V. 오케스트레이션

❖**Overlay Network**



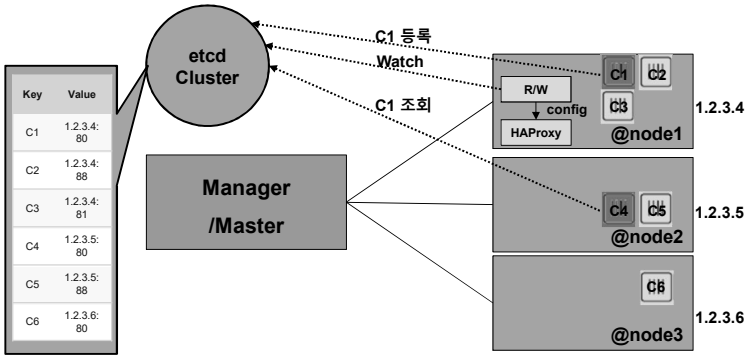
V. 오케스트레이션

❖Distributed system scheduler in action



V. 오케스트레이션

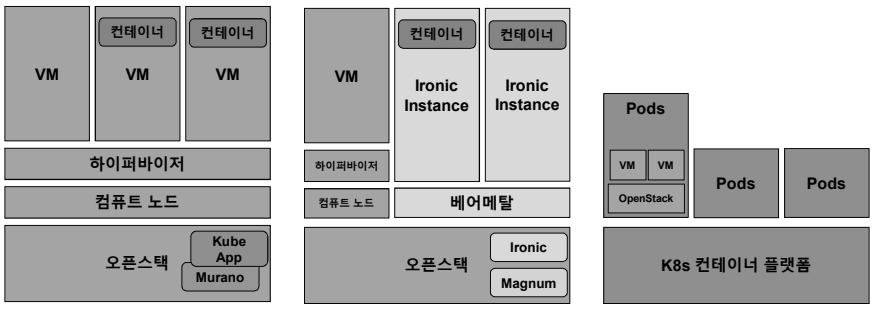
❖etcd service discovery setup is quite similar to the ZK setup
❖ The main difference is the usage of confd, which configures HAProxy, rather than having you write your own script.



V. 오케스트레이션

❖ **아키텍처 비교**

- **VM 기반 컨테이너 아키텍처:** Pet 기반 운영 가능
- **VM 기반 베어메탈 컨테이너:** 베어메탈 지원
- **오픈스택과 베어메탈 컨테이너 혼용:** 레거시 VM 필요시 오픈스택 구동

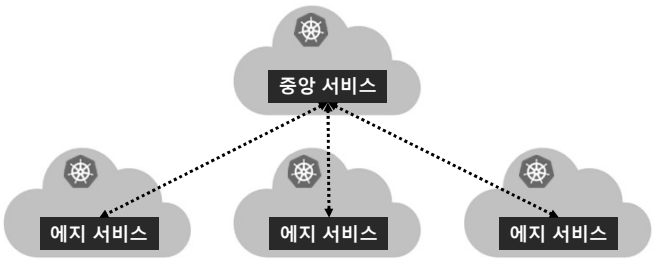


VM 기반 컨테이너 아키텍처 | VM 기반 베어메탈 컨테이너 | 오픈스택과 베어메탈 컨테이너 혼용

V. 오케스트레이션

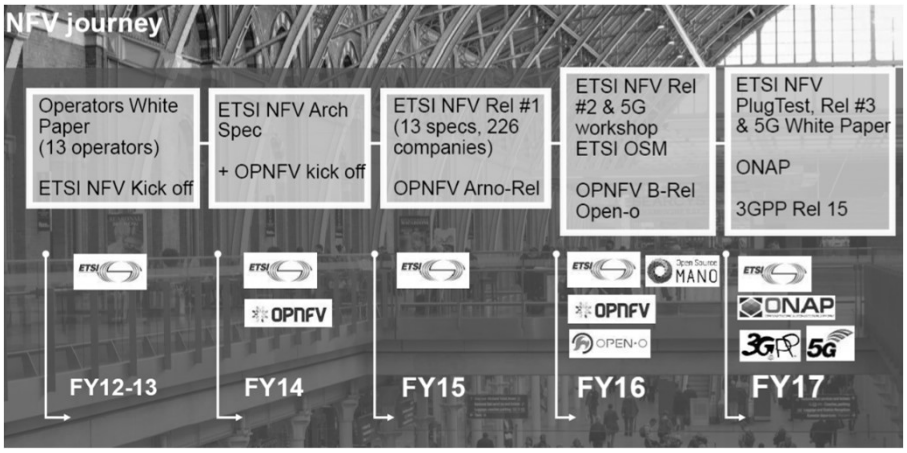
❖ **Kubernetes:**

- **K8s는 워크로드 Agnostic** (컨테이너, VM, Function)
- **K8s는 다양한 요구의 하드웨어 플랫폼 지원**
- **K8s는 역동적 서비스를 위한 앱의 이동과 처리 증가와 감소의 장점**
- **K8s는 상용 적용 확장을 위한 일관된 플랫폼의 검증**
- **K8s는 신개발이 필요하지 않은 수준의 많은 레퍼런스로 개발자가 익숙함**
- **기기 관리 필요** (PXE, DHCP, IPMI, TFTP, Discovery)



V. 오케스트레이션

❖ 소프트웨어 정의 기반 SDN/NFV 발전
❖ Container 기반 Cloud 기술 도입 가속화

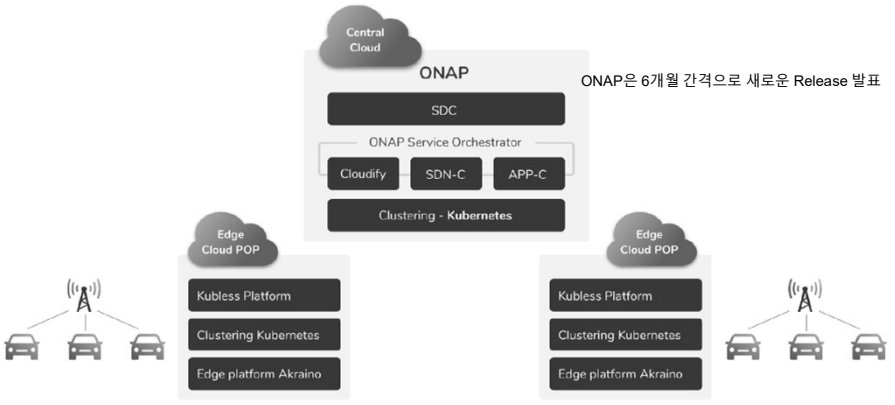


james@jslab.kr

V. 오케스트레이션

❖ 경로 종단간 Network Slicing을 위한 오케스트레이션과 모니터링
❖ ONAP 프로젝트의 SDC/SO (예)

- **Kubeless** (Kubernetes-native serverless framework)
- **Kata Containers** (초경량 VM을 위한 open source project)

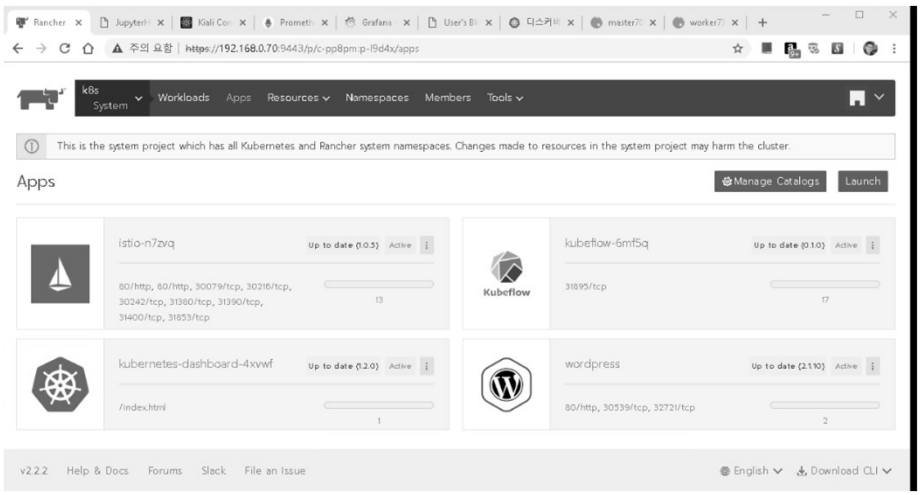


james@jslab.kr

ONAP (Open Network Automation Platform) Service Design and Creation (SDC) Service Orchestrator (SO)

V. 오케스트레이션

❖ **ONAP 프로젝트 테스트에 사용하는 Rancher (예)**



ONAP (Open Network Automation Platform) Service Design and Creation (SDC) Service Orchestrator (SO)

V. 오케스트레이션

❖ **에지 환경등 데이터센터 밖의 Kubernetes 지원 요구 수용**
❖ **Open source project 'K3s'** (예)

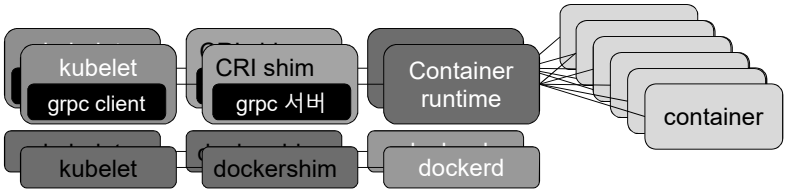
- **K8s 기본 기능을 유지하며 Production을 위해 300만 라인 정도 삭제한 바이너리 40 MB로 메모리 512 MB 사용 수준**
- **알파기능, 클라우드, 스토리지 등의 많은 기능 제거**
- **인텔과 ARM 계열 하드웨어 지원 (x86_64, ARM64, and ARMv7)**
- **6개월 내에 HA 지원 예정**
- **Docker는 선택으로 삭제 (containerd 추가)**
- **단일 프로세스에 총합하는 K8s master, Kubelet, containerd**
- **SQLite 를 etcd에 추가**

V. 오케스트레이션

❖ **CRI (Container Runtime Interface): Kubernetes(K8s)는 특정 런타임과 분리 추상화하며 K8s 1.6(2017년 3월) kublet에서 CRI를 정식 적용**

❖ **CRI 호환 런타임 프로젝트** (배포범위 확대: VM/하이퍼바이저/베어메탈)

- **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**



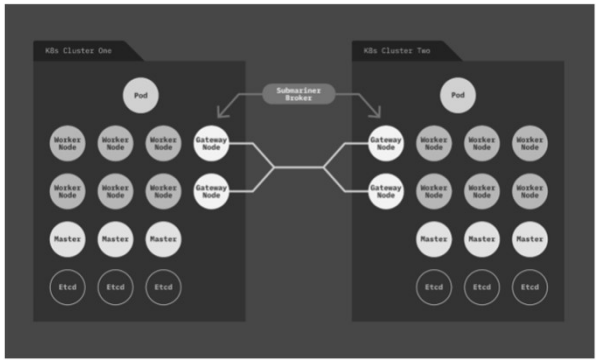
james@jslab.kr

V. 오케스트레이션

❖ **K8s Cluster간 네트워크 연결 요구**

❖ **Open source project 'Submariner'** (예)

- **카산드라(Cassandra)와 같은 HA 데이터베이스의 지역 분산**
- **분산화 트레이스 (Distributed Tracing)**
- **Service Mesh의 클러스터들 간에 확대**

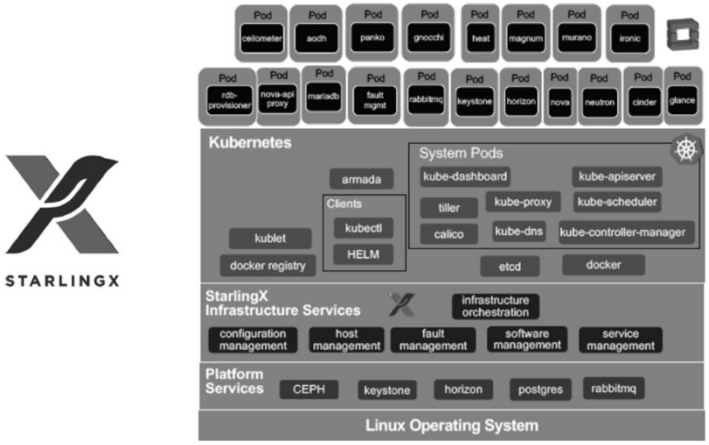


https://submariner.io/

james@jslab.kr

V. 오케스트레이션

- ❖ 에지 클라우드 플랫폼
- ❖ 필요시 VM 지원 가능한 컨테이너 아키텍처
- ❖ **StarlingX(예):** OpenStack Foundation

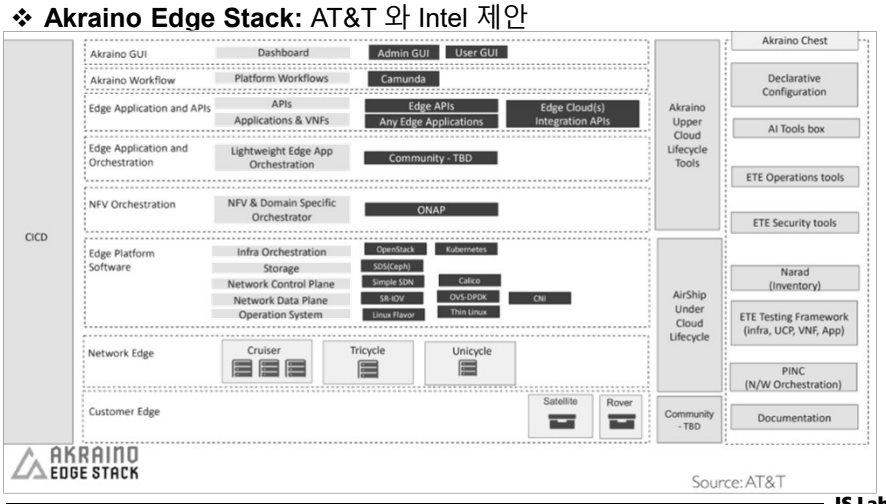


STARLINGX

https://www.starlingx.io/

V. 오케스트레이션

- ❖ 리눅스 재단의 에지 아키텍처 프로젝트 제안
- ❖ 통신사의 기지국 및 기업 환경 API 제공 응용 시장 (예)
- ❖ **Akraino Edge Stack:** AT&T 와 Intel 제안

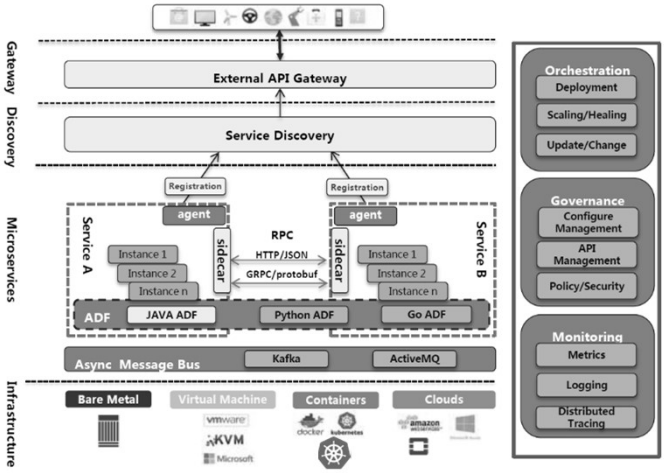


www.akraino.org

V. 오케스트레이션

❖ 국내 통신사 5G 서비스에 마이크로 서비스 아키텍처 도입 중

❖ **OMSA - ONAP Microservice Architecture** (예)



Note: this diagram is a functional view of OMSA, which is not mapped to specific projects

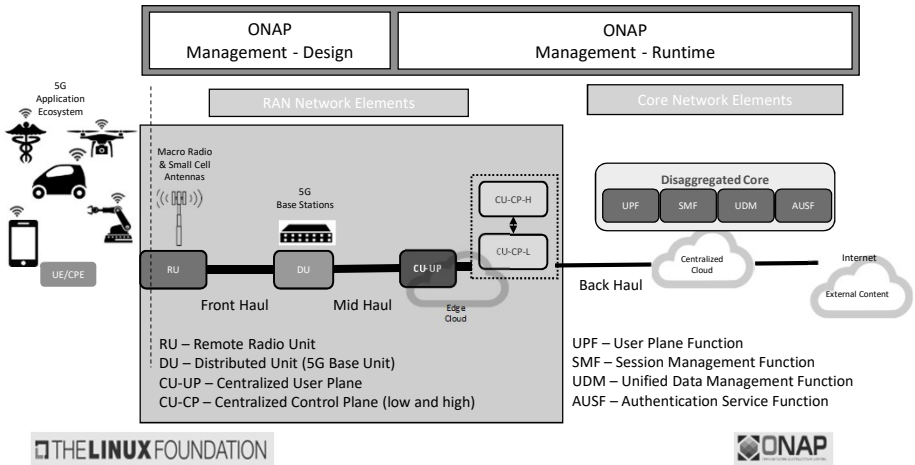
JS Lab

james@jslab.kr

V. 오케스트레이션

❖ **5G Enterprise Business 변화**

❖ **기업을 위한 5G LAN** (예: 5G 기지국, 5G Core, MEC 서버 제공)



THE LINUX FOUNDATION

ONAP

ONAP (Open Network Automation Platform)

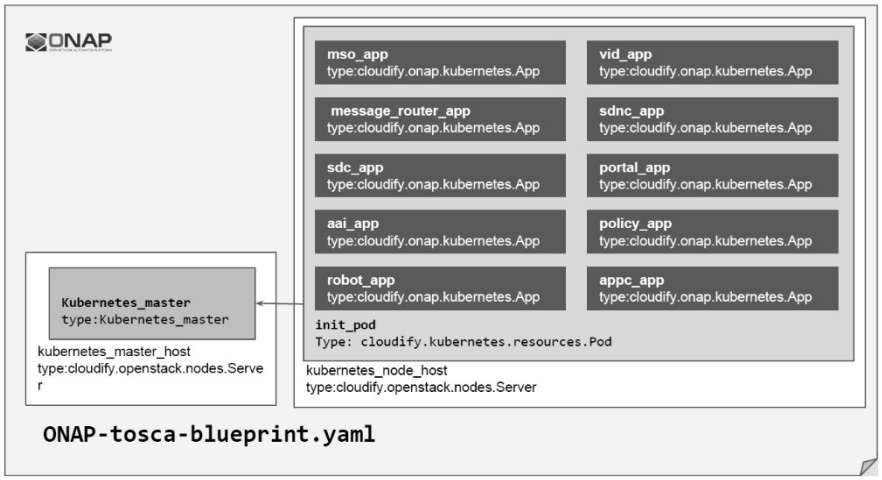
JS Lab

james@jslab.kr

V. 오케스트레이션

❖ **Telco용 멀티벤더 환경 관리 표준 TOSCA**

❖ **ONAP의 TOSCA Template** (예: Public Cloud, Private Cloud 적용가능)



https://gerrit.onap.org/r/gitweb?p=oom.git;a=blob;f=onap-blueprint.yaml

V. 오케스트레이션

❖ **관리 표준과 오픈소스**

표준

OASIS
Advancing open standards for the information society

OASIS TOSCA
Topology and Orchestration Specification for Cloud Applications (TOSCA)

tmforum

MEF
METRO ETHERNET FORUM

ETSI

IETF

오픈소스

OPEN-O

Open Source MANO

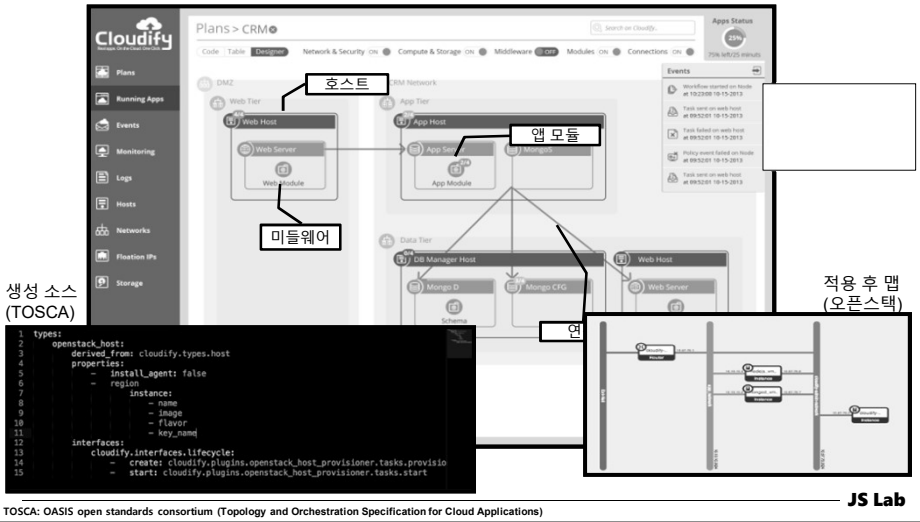
CLOUDIFY

ARIA

ONAP
OPEN NETWORK AUTOMATION PLATFORM

V. 오케스트레이션

❖ **멀티 클라우드 오케스트레이션 :** 표준 TOSCA 기반 GUI 서비스 (TOSCA 표준 적용 오픈소스 Cloudify 예)

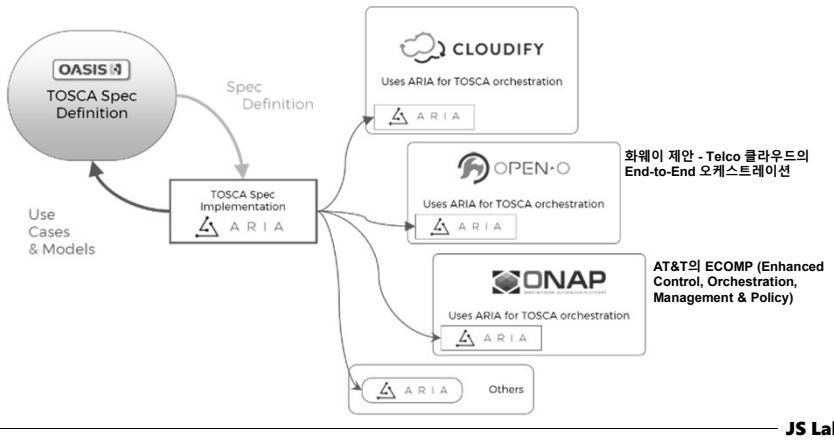


TOSCA: OASIS open standards consortium (Topology and Orchestration Specification for Cloud Applications)

V. 오케스트레이션

❖ **표준 TOSCA 스펙 적용 오픈소스 'ARIA'**

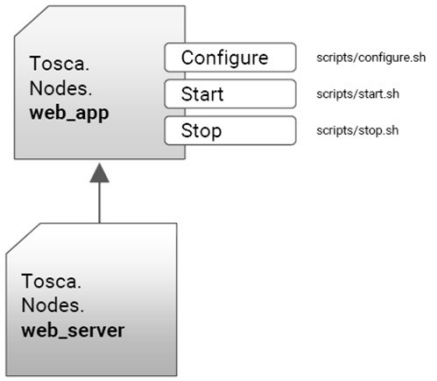
1. **오케스트레이션이 TOSCA 프로파일 지원을 위한 Python 라이브러리**
2. **TOSCA 애플리케이션 생성을 위한 SDK**
3. **CLI Tools: 오케스트레이션을 위한 TOSCA 템플릿**



V. 오케스트레이션

❖ **표준 TOSCA 적용 'ARIA' Hello World 예 (TOSCA/ARIA)**

1. 소스 공개
2. helloworld.yaml




```
1.  tosca_definitions_version: tosca_simple_yaml_1_0
2.  node_types:
3.    HelloWorld:
4.      derived_from: tosca:WebApplication
5.      requirements:
6.        - host:
7.            # Override to allow for 0 occurrences
8.            capability: tosca:Container
9.            occurrences: [ 0, UNBOUNDED ]
10. topology_template:
11.   inputs:
12.   node_templates:
13.     hello_world:
14.       type: HelloWorld
15.       capabilities:
16.         app_endpoint:
17.           properties:
18.             protocol: http
19.             port: 9090
20.       interfaces:
21.         Standard:
22.           configure: scripts/configure.sh
23.           start: scripts/start.sh
24.           stop: scripts/stop.sh
25.   outputs:
26.     port:
27.       type: integer
28.   value: { get_property: [ hello_world, app_endpoint, port ] }
```

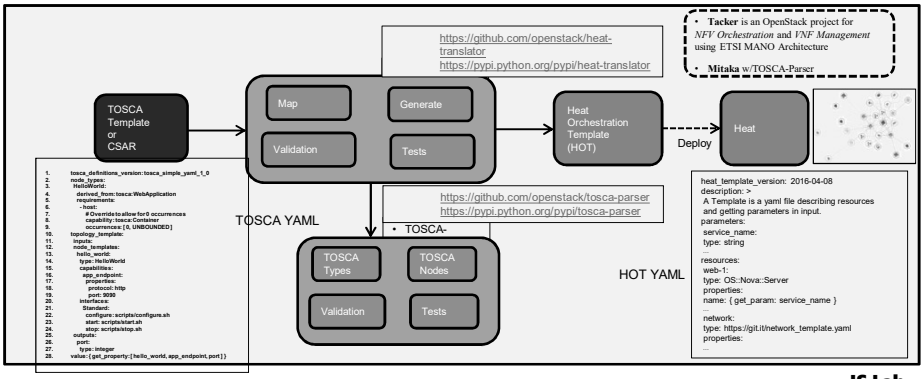
https://github.com/apache/incubator-ariatosca/blob/master/examples/hello-world/hello-world.yaml

james@jslab.kr

V. 오케스트레이션

❖ **TOSCA 처리 과정 (오픈스택 예)**

- TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0 Committee Specification Draft 03 (17 March 2016)
- 토스카 파서(TOSCA-Parser): Parser for TOSCA Simple Profile in YAML
- 히트번역기(Heat-Translator): An OpenStack project to map and translate non-Heat (e.g. TOSCA) templates to Heat Orchestration Template (HOT)

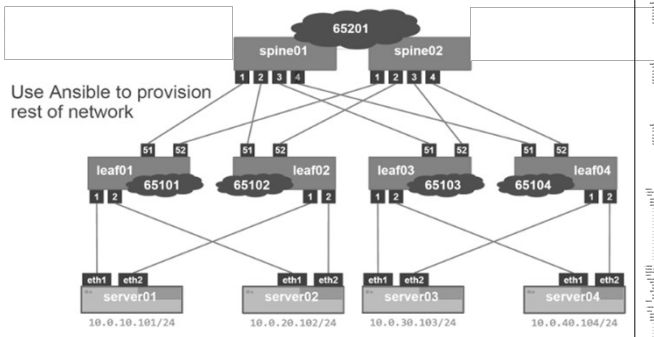


CSAR (Cloud Service Archive) TOSCA (Topology and Orchestration Specification for Cloud Applications)

james@jslab.kr

V. 오케스트레이션

- **Ansible Template** (Playbook)
- **ansible-playbook.yml**



V. 오케스트레이션

- **Operation for Ansible "ansible-playbook.yml"**

```
cumulus@oob-mgmt-server:~/NetworkAutomation$ ansible-playbook.yml
...
PLAY RECAP *********************************************************************************
leaf01                     : ok=10   changed=7    unreachable=0    failed=0
leaf02                     : ok=10   changed=6    unreachable=0    failed=0
leaf03                     : ok=10   changed=6    unreachable=0    failed=0
leaf04                     : ok=10   changed=6    unreachable=0    failed=0
server01                   : ok=3    changed=1    unreachable=0    failed=0
server02                   : ok=3    changed=1    unreachable=0    failed=0
server03                   : ok=3    changed=1    unreachable=0    failed=0
server04                   : ok=3    changed=1    unreachable=0    failed=0
spine01                    : ok=10   changed=6    unreachable=0    failed=0
spine02                    : ok=10   changed=6    unreachable=0    failed=0
Wednesday 11 October 2017  19:42:59 +0000 (0:00:05.074)       0:01:16.673 *****
===============================================================================
reset : Clear config ---------------------------------------------------------- 16.03s
Net 6 -- Deploy Configuration To All Leafs ------------------------------------ 13.61s
Net 6 -- Deploy Configuration to All Spines ----------------------------------- 9.70s
reset : Restore NTP ----------------------------------------------------------- 7.97s
Restart Networking ------------------------------------------------------------ 5.07s
Restart PTM Daemon to Apply new Topology.dot file ----------------------------- 3.87s
Gathering Facts --------------------------------------------------------------- 3.21s
Gathering Facts --------------------------------------------------------------- 3.01s
Restart the netq-agent -------------------------------------------------------- 2.93s
reset : Copy the Default Interface Configuration in Place --------------------- 2.83s
Download the topology.dot file from the OOB-MGMT-SERVER ----------------------- 2.11s
Gathering Facts --------------------------------------------------------------- 1.80s
reset : Apply Default Interface Configuration --------------------------------- 1.76s
Copy Interfaces Configuration File -------------------------------------------- 1.52s
Gathering Facts --------------------------------------------------------------- 1.20s
cumulus@oob-mgmt-server:~/NetworkAutomation$
```

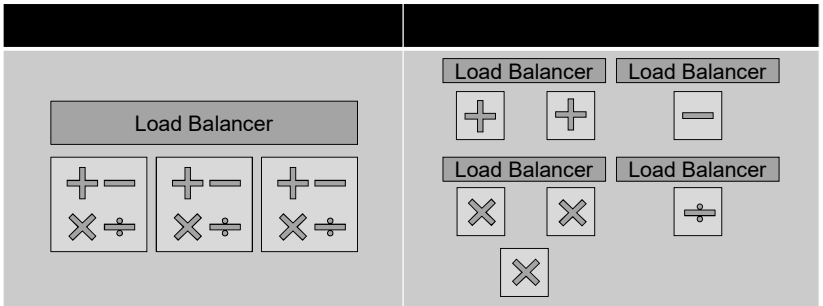
목차

james@jslab.kr

VI. 마이크로서비스 아키텍처

❖**What Are Microservices?**

- 마이크로서비스란 아키텍처이자 소프트웨어 작성을 위한 하나의 접근 방식으로, 애플리케이션을 상호 독립적인 최소 규모 구성 요소로 분할하여 마이크로서비스간 연결(연결 예: http, RESTFUL web service, 메세지큐)
- 모든 요소를 하나의 애플리케이션에 구축하는 전통적인 모놀리스식(Monolithic) 접근 방식 대신 마이크로서비스에서는 모든 요소가 분리되고 연동되어 동일한 태스크를 완수 이러한 각각의 구성 요소 또는 프로세스가 마이크로서비스



james@jslab.kr

Kocher, Parminder Singh. Microservices and Containers. Pearson Education

VI. 마이크로서비스 아키텍처

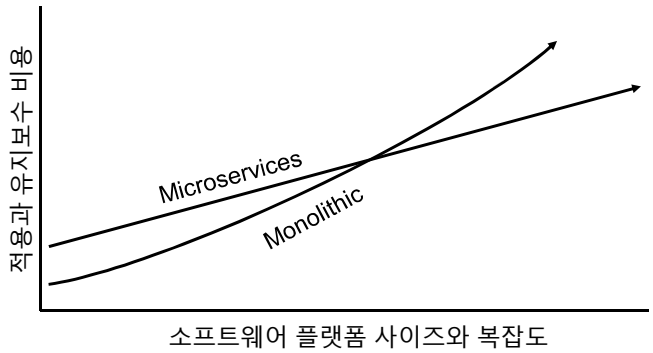
❖ **현재의 애플리케이션 아키텍처 문제** (예)

- **40명 이상의 개발자**
- **글로벌 사용자**
- **100,000 이상의 사용자**
- **100.000 라인 이상의 코드**

james@jslab.kr

VI. 마이크로서비스 아키텍처

❖ **비용**



james@jslab.kr

VI. 마이크로서비스 아키텍처

❖**MSA(Micro Service Architecture)를 선택한 이유 (SK Planet 예)**

Frontend/Backend 분리	회사의 Engineer Tech Tree와 동기화
코드 양이 커지고, 중복코드가 발생	코드의 양을 줄여서 누구나 쉽게 파악하도록
팀 전체의 Project	Working Group별 Product
시스템 간 연계 증대	API 기반의 Contract 관리를 강제화
새로운 기술에 대한 도입 욕구	Micro-Service 단위로 구현에 자율성 부여 (여러 기술을 사용하는 Polyglot)
재사용성 향상 및 지속적인 발전	Micro-Service 단위의 재사용, 자유로운 리팩토링
회사 인프라의 뒷받침	On-Premise Cloud, CI와 연계된 배포 자동화(Jarvis), 향후 Docker와 같은 Container 기술과 연계

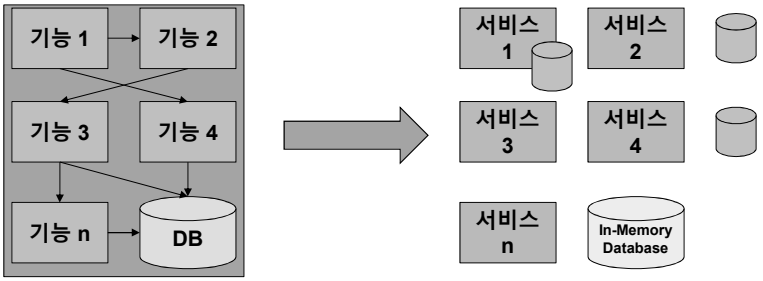
james@jslab.kr

VI. 마이크로서비스 아키텍처

❖**마이크로서비스 전환 시 추가 기능 고려**

- **비즈니스 기능을 위한 서비스 연결**
- **스탠드얼론 and/or 서비스의 부분 적용**
- **비동기 통신**
- **성능 개선을 위한 서비스 플랫폼 요소 교체** (프로그램 언어, 데이터베이스 등)
- **일정하지 않은 확장 요구 CI/CD**
- **각 엔지니어링 팀은 비즈니스 영역의 이해하고 책임을 소유**

❖**마이크로서비스 전환 시 조직 문화의 변화와 운영 프로세스 변화**



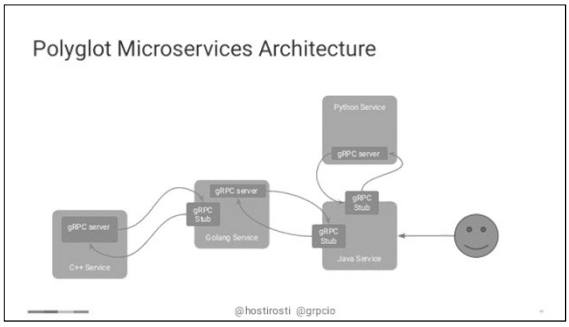
james@jslab.kr

CI/CD (Continuous integration and Continuous delivery)

VI. 마이크로서비스 아키텍처

❖gRPC 사용 마이크로서비스

- **gRPC is faster than REST. gRPC uses HTTP2 by default**
- **gRPC defines relationship between client and server and enforces strict rules of communication between them.** (In REST calls, the request and response are totally de-coupled)
- **gRPC supports useful additions like standard error responses and meta data.**



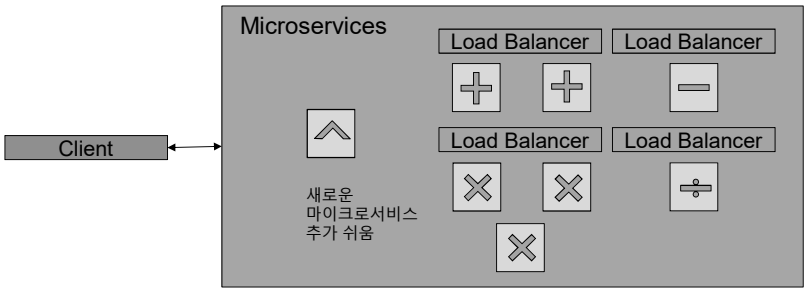
https://medium.com/@akshitjain_74512/inter-service-communication-with-grpc-d815a561e3a1

james@jslab.kr

VI. 마이크로서비스 아키텍처

❖마이크로서비스의 장점

- **단순성(Simplicity)**
- **확장성(Scalability)**
- **적용(Continuous delivery)**
- **낮은 의존성과 더 많은 자유**
- **장애 확인**
- **데이터 분리와 분산화**
- **다양한 선택**

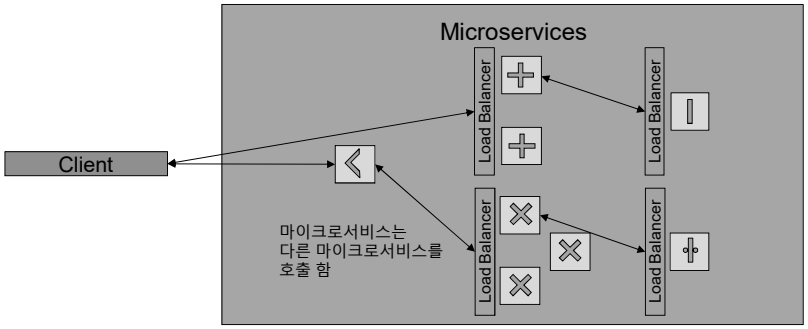


james@jslab.kr

VI. 마이크로서비스 아키텍처

❖마이크로서비스의 단점

- 트러블슈팅의 복잡성
- Latency 증가
- 운영의 복잡성
- 버전 제어



VI. 마이크로서비스 아키텍처

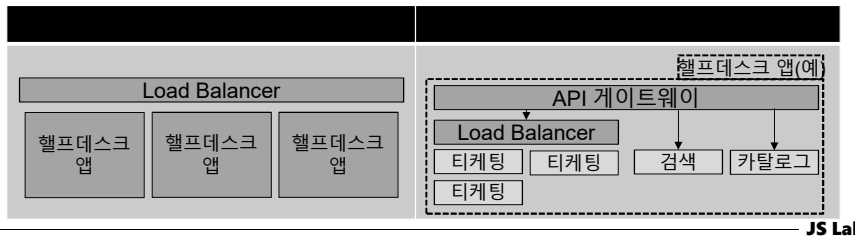
❖조직의 Learning Curve

- 단일 마이크로서비스 (Standalone Microservice) 많이 필요시
- 과도한 의존성으로 시간이 많이 필요하고 코드의 품질이 낮아질 때
- 한가지 요소로 애플리케이션 장애 시

❖마이크로서비스 전환 시 추가 기능 고려

- 비즈니스 기능을 위한 서비스 연결
- 스탠드얼론 and/or 서비스의 부분 적용
- 각 엔지니어링 팀은 비즈니스 영역의 이해하고 책임을 소유

❖마이크로서비스 전환 시 조직 문화의 변화와 운영 프로세스 변화



VI. 마이크로서비스 아키텍처

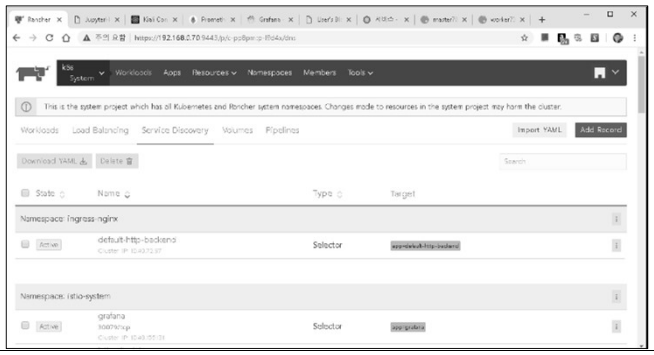
❖유지보수

- 기존 클라이언트 적용 지원
- 장애 고려 (적절한 error code 생성 등)
- 모니터링 (가용성이나 오케스트레이션 등을 지원하는 도구사용)
- 큐

VI. 마이크로서비스 아키텍처

❖Discovery Service

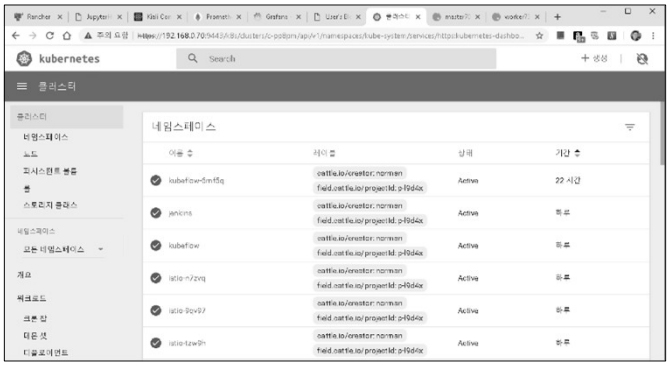
- Client는 기존에는 1개의 monolithic app을 하였으나 MSA에서는 동시에 여러 개의 서비스를 호출해야 함
- Client는 서비스들의 위치를 알아야 함 (Host/IP/Port 등)
- Client가 마이크로서비스 호출시 제공하는 entry point 실시간 위치를 하드코드(Hard code)보다는 유연한 방법으로 현재의 위치를 제공해야 함



VI. 마이크로서비스 아키텍처

❖ **API Gateway**

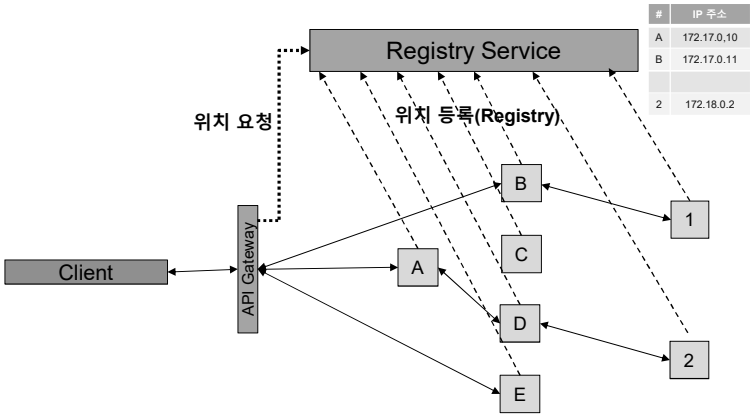
- **모든 호출에 대한 1개의 entry point가 필요**
- **내부의 복잡성을 숨김**
- **마이크로 서비스 변화/결합/구분/추가/제거 유연함**
- **Client와 Application사이의 왕복 단순화로 효율성 증대**



VI. 마이크로서비스 아키텍처

❖ **Service registry**

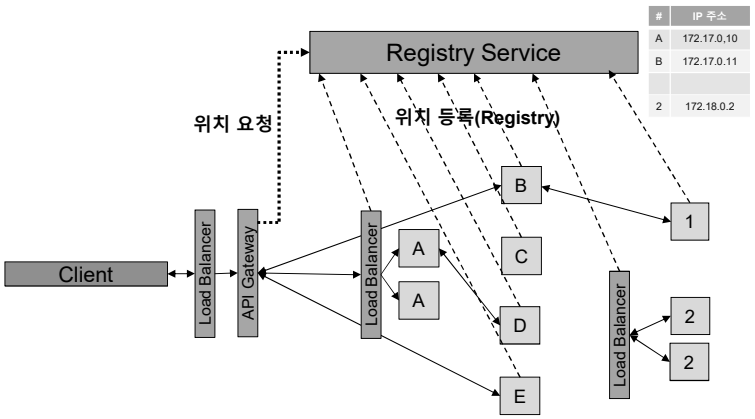
- **API Gateway는 모든 서비스에 대한 IP 주소를 알아야 하며 이의 DB 필요**
- **Registry 데이터의 안정성을 위해 오픈소스 사용** (Consul이나 SkyDNS)



VI. 마이크로서비스 아키텍처

❖로드밸런서 추가시 Service registry

- API Gateway는 모든 서비스에 대한 IP 주소를 알아야 하며 이의 DB 필요
- Registry 데이터의 안정성을 위해 오픈소스 사용(Consul이나 SkyDNS)



james@jslab.kr

VI. 마이크로서비스 아키텍처

❖마이크로서비스 기술의 어려움

- 서비스디스커버리 (Service Discovery)
- 운영의 부담 (100s+ of Services !!!)
- 분산된 시스템
- 서비스 의존성
  - 서비스 팬아웃(service fan-out)
  - 서비스 의존성의 부담 (dependency services running “hot”)
- 트래픽과 서비스별 부하 처리
- 서비스의 동작 상태 확인 / 장애 자동 복구
- Auto-Scale

james@jslab.kr

VI. 마이크로서비스 아키텍처

❖적용 기술 선택

- **Independency**
- **Source Control**
- **Environment**
- **Failsafe**
- **Reuse**
- **Tagging**

❖운영 기술 선택

- **모니터링: 인프라의 모든 요소를 모니터하며 라이브 대쉬보드와 이슈 팝업** (로그 관리 예: Splunk나 EFK or ELK 'Elasticsearch, Logstash, Kibana')
- **온디맨드 확장성: 수동 또는 자동화**
- **API 노출: Netflix의 경우 Zuul 오픈소스로 공개**
- **서킷 브레이커 Circuit Breaker): 모든 시비스에 대해 요청모드를 요구하여 장애를 확인하며, 복수의 장애시 특정 서비스를 차단**(Break the circuit)**함**

james@jslab.kr

VI. 마이크로서비스 아키텍처

❖성공 사례

- **아마존**
- **Netflix**
- **우아한 형제들 (배달의 민족) Spring Cloud zuul 적용**
- **11번가 Spring Cloud 적용**
- **SK C&C C-NAPS 방법론 개발한가지 요소로 애플리케이션 장애 시**

❖마이크로서비스 사례

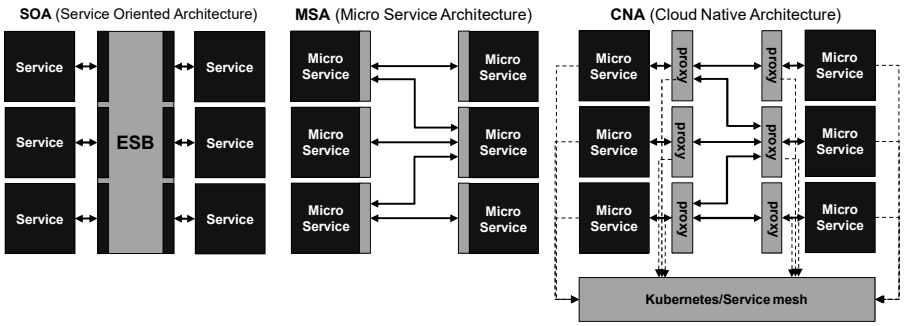
- **http://channy.creation.net/articles/microservices-by-james_lewes-martin_fowler (마틴 파울러의 마이크로 서비스 정의)**
- **https://www.youtube.com/watch?v=OczG5FQIcXw (넷플릭스 마이크로 서비스 가이드 – 혼돈의 제왕)**
- **http://woowabros.github.io/r&d/2017/06/13/apigateway.html (우아한 형제들 spring cloud zuul 적용기)**
- **https://www.popit.kr/why-microservice/ (마이크로 서비스의 장단점)**
- **https://blog.skcc.com/3306 (SK C&C C-NAPS)**

james@jslab.kr

VI. 마이크로서비스 아키텍처

❖컨테이너 기반 아키텍처

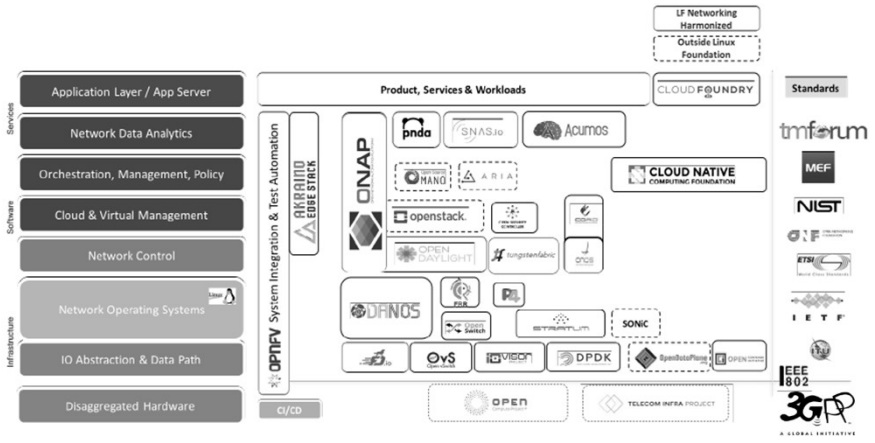
- **SOA** (Service Oriented Architecture): Smart pipes, dump endpoints
- **MSA** (Micro Service Architecture): Smart endpoints, dump pipes
- **CNA** (Cloud Native Architecture): Infrastructure focused smart platform, business logic focused smart services



목차

VII.SDN/컨테이너 네트워킹

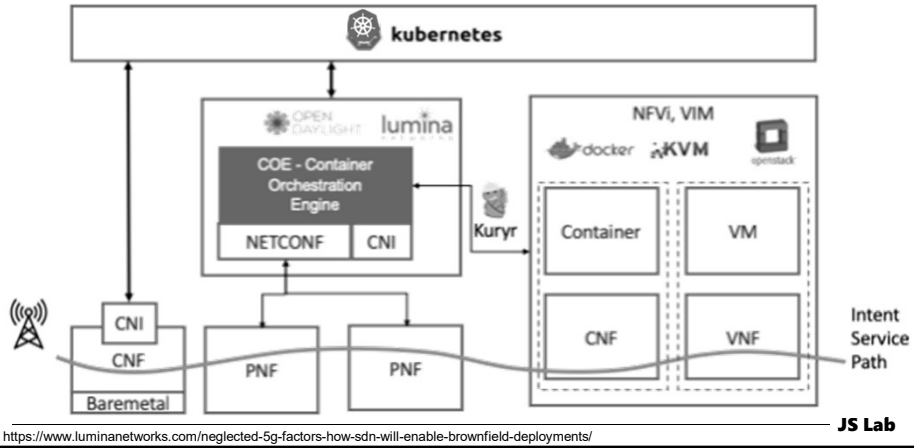
- ❖ **Open Source and Software Defined Networking Landscape**
- ❖ **리눅스 재단 내/외의 K8s 도입 네트워킹 프로젝트 증가 중**



https://www.linuxfoundation.org/projects/networking/

VII.SDN/컨테이너 네트워킹

- ❖ **기존 통신 장비 제조사와 협력이 필요한 브라운 필드(Brownfield) 적용**
  - **컨테이너, 가상머신(VM), 물리머신(BareMetal) 혼용 환경 추상화** (Intent)
  - **VNF**(Virtual Network Functions)**, CNF**(Container NF)**, PNF**(Physical NF)
  - **제조사 솔루션 (예): K8s 관리 기반의 SDN 제어기, 오픈스택, 도커 사용**

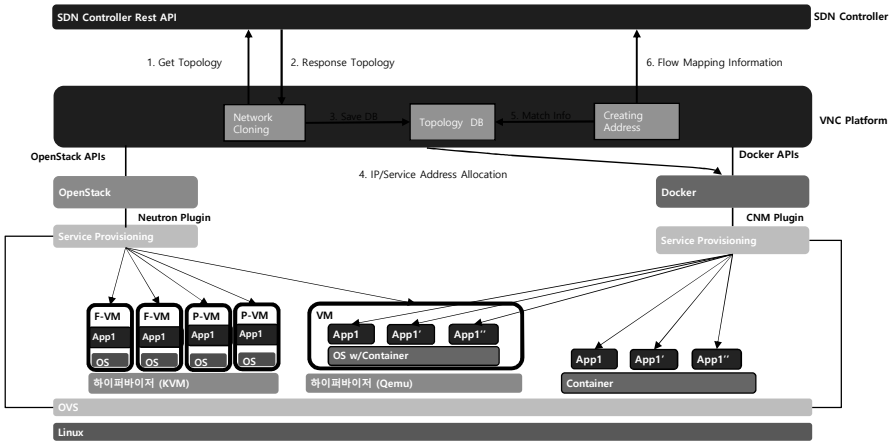


https://www.luminanetworks.com/neglected-5g-factors-how-sdn-will-enable-brownfield-deployments/

VII.SDN/컨테이너 네트워킹

❖ **가상 네트워크 클로닝 (VNC)**

- **VNC 속도 개선을 위한 구조** (주소 매핑 테이블 생성/관리)
- **자원 절약과 클로닝 확장성 제공**



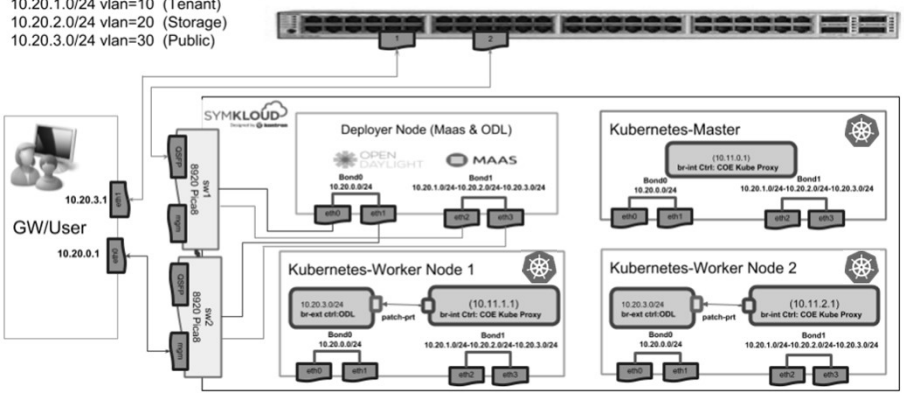
VII.SDN/컨테이너 네트워킹

❖ **Telco를 위한 K8s의 제어플레인 기능 배포**

- **Telco의 SDN 스택을 위한 K8s 채택** (예)
- **K8s는 제어기능의 배포 위치 변경 요구 수용 필요** (DC or UP)

10.20.0.0/24 untagged (Management)
10.20.1.0/24 vlan=10 (Tenant)
10.20.2.0/24 vlan=20 (Storage)
10.20.3.0/24 vlan=30 (Public)

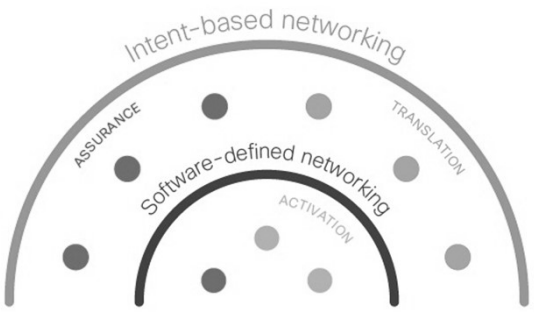
Telco의 SDN 스택을 위한 K8s 채택 제조사 제공 (예)



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

VII.SDN/컨테이너 네트워킹

- **IBN (Intent-Based Networking)**
  1. **SDN 확장 개념을 발전**
  2. **정책 적용 강화**
  3. **"Intent" enables the expression of both business purpose and network context through abstractions**



Source: Cisco

https://www.sdxcentral.com/sdn/definitions/intent-based-networking-vendors-debate-term/?utm_content=buffercb357a&utm_medium=referral&utm_source=facebook.com&utm_campaign=buffer&fbclid=IwAR1WpY54f52tsuDkM5Zu6AGAz_rWLhxbRHw8fcMQRrov0_YzKz-QfccUcF0

VII.SDN/컨테이너 네트워킹

- **Cisco의 IBN (Intent-Based Networking)**

Networking comparison

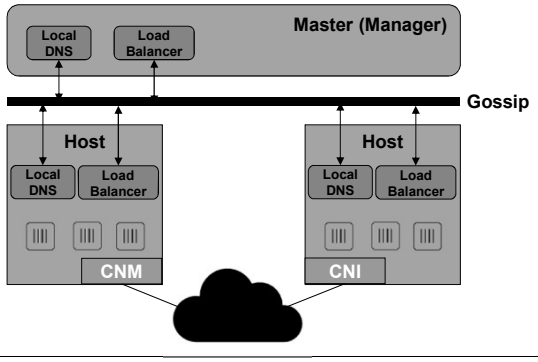
TRANSLATION	SOFTWARE-DEFINED	INTENT-BASED
Input intent		●
Translate to policy		●
Check integrity		●
ACTIVATION		
Orchestrate policies	●	●
Automate network configurations	●	●
ASSURANCE		
Visibility	CENTRALIZED (DOMAIN)	CENTRALIZED (WITH BROADER CONTEXT)
Insights (context + policy)		●
Continuous verification		●
Corrective actions		●

https://blogs.cisco.com/analytics-automation/why-is-intent-based-networking-good-news-for-software-defined-networking

VII.SDN/컨테이너 네트워킹

❖Container Networking 표준 - CNM / CNI / Gossip

- **Container Network Model (CNM):** Docker에서 제안, libnetwork에서 사용하며 Cisco Contiv, Kuryr, OVN, Project Calico, VMware, Vwave에서 사용
- **Container Network Interface (CNI):** CoreOS에서 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



VII.SDN/컨테이너 네트워킹

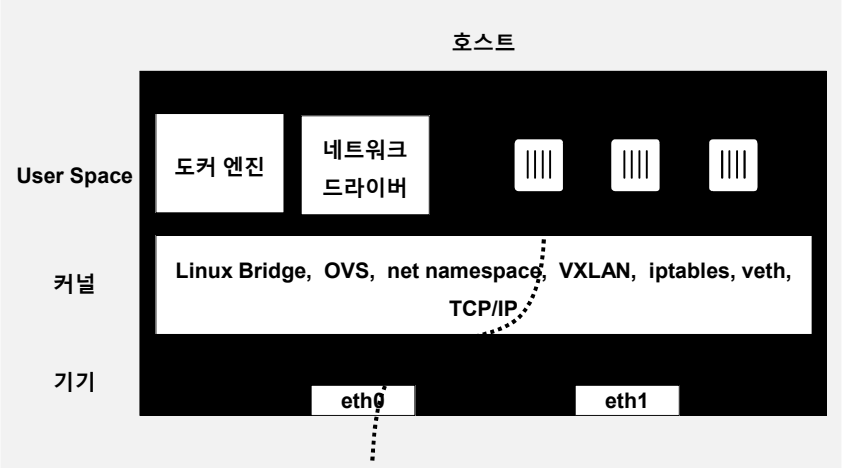
❖컨테이너 네트워킹(Container Networking) 종류

- **None: 호스트간 연결 없음**
- **브릿지(Bridge): L2 브릿지를 사용**
- **오버레이(Overlay): 터널링 사용 오버레이로 호스트 간 네트워크 연결**
- **언더레이(Underlay): 컨테이너를 물리적 인터페이스에 직접 연결**

Docker (예)	오버레이	브릿지 / 포트맵핑	언더레이
멀티 호스트 연결	Yes	No (native support)	No (native support)
서비스 발견 (Service Discovery)	클러스터 간의 글로벌 SD	호스트 네트워크 상의 로컬 SD	호스트 네트워크 상의 로컬 SD
로드밸런싱	-내부 글로벌 VIP 기반 -내부 글로벌 DNS 기반 -외부 라우팅 메쉬	내부 로컬 DNS 기반	내부 로컬 DNS 기반
IP Addressing	-컨테이너 당 내부 주소체계 -오버레이당 글로벌 범위	컨테이너 당 내부 주소체계 브릿지당 로컬 범위	물리 네트워크 상의 컨테이너당 외부 주소 체계
암호화	Yes, 선택	No	No
요구사항	엔진 1.12 이상 클러스터 스웜 (Swarm)모드	엔진 1.7 이상	호스트 인터페이스에 Promiscuous mode 필요

VII.SDN/컨테이너 네트워킹

❖도커 네트워킹은 리눅스 네트워킹

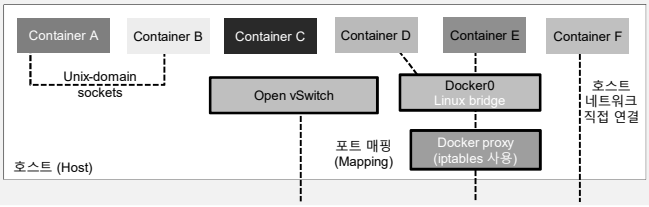


IPAM: IP Address Management

VII.SDN/컨테이너 네트워킹

❖하이레벨 (High-level) 기능

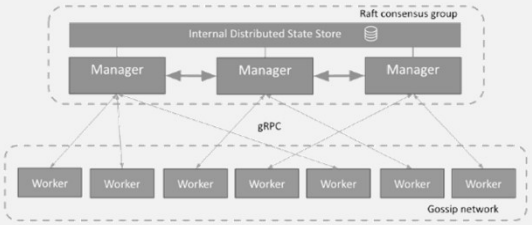
Namespace	/proc에서 프로세스 수준 관리의 컨테이너 네트워킹
Linux Bridge	커널에서 포워딩에 사용하는 L2/MAC을 인식하는 스위치
Open vSwitch	프로그램 가능하고 터널링을 지원하는 개선한 브릿지 (SDN 스위치)
NAT	네트워크 주소 변화 IP address + Ports (Types: SNAT, DNAT)
iptables	커널 내의 정책 엔진으로 패킷전송, 방화벽, NAT를 관리함
Unix domain sockets	단일 호스트 내 통신 기반의 File descriptor, FIFO 파이프로 동작
User-space vs Kernel-space	자원과 성능을 정상화 제어하는 애플리케이션도메인 • 컨테이너(Container) 애플리케이션(applications)은 user-space 에서 실행 • 네트워크 전송은 kernel space에서 실행



VII.SDN/컨테이너 네트워킹

- ❖ 토폴로지
- ❖ 스웜(Swarm) Manager 는 3개 or 5개 or 7개등 홀수 권장

스웜 매니저간들의 통신은 항상 쿼럼(Quorum) 가능



- /var/lib/docker/swarm/docker.state.json

{"LocalAddr":"","RemoteAddr":"192.168.99.118:2377","ListenAddr":"0.0.0.0:2377","AdvertiseAddr":""}

- /var/lib/docker/swarm/state.json

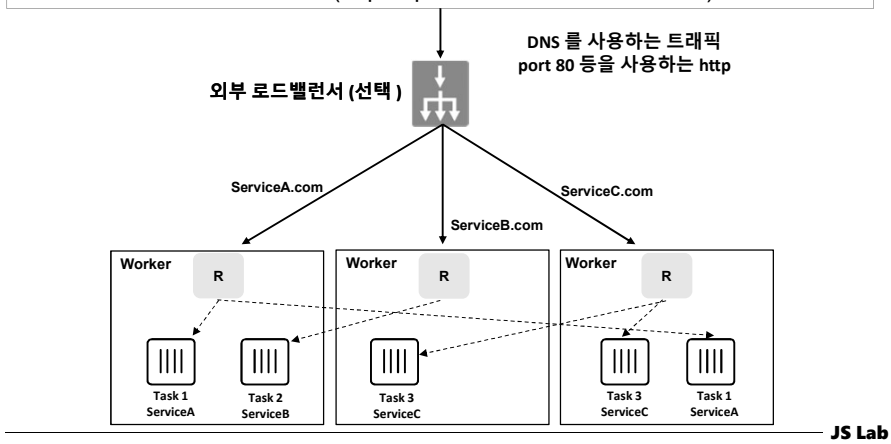
[{"node_id":"9c5eqant0s2w7arlfk47tkxm0","addr":"192.168.99.118:2377"},
{"node_id":"bexym9a2cxbd60ow40xibycw5","addr":"192.168.99.115:2377"},
{"node_id":"cviejn6myjln6s4ysw4wg59rn","addr":"192.168.99.119:2377"}]

```
Swarm: active
 NodeID: kj4l7bnzmr8yphsmln33zs7a8
 Is Manager: true
 ClusterID: tzzk07zwhg8axv0jayf2u58yd
 Managers: 3
 Nodes: 8
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 3
 Dispatcher:
  Heartbeat Period: 5 seconds
 CA Configuration:
  Expiry Duration: 3 months
  External CAs:
    cfssl: https://192.168.99.131:12381/api/v1/cfssl/sign
    cfssl: https://192.168.99.129:12381/api/v1/cfssl/sign
    cfssl: https://192.168.99.130:12381/api/v1/cfssl/sign
 Node Address: 192.168.99.129
 Manager Addresses:
  192.168.99.129:2377
  192.168.99.130:2377
  192.168.99.131:2377
```

'docker info'의 swarm 정보 표시(예)

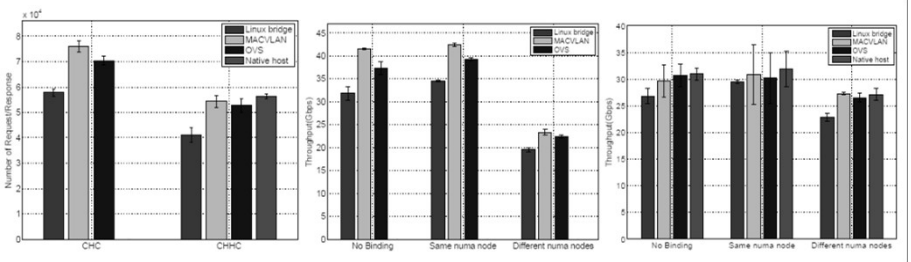
VII.SDN/컨테이너 네트워킹

- ❖ **라우팅 메쉬(Routing mesh):** 에지 라우팅을 위한 내장 라우팅 메쉬(routing Mesh)에서 모든 워커 노드(Worker Node)가 인그레스 라우팅 메쉬(Ingress Routing Mesh)에 참여하여 공개된 포트(Published Port)의 접속 요청을 수용하고 포트 변환은 워커노드에서 수행한다. 내부 로드밸런싱 매커니즘은 외부 요청에 동일하게 사용 (http/https 포트번호 임의 지정 가능)



VII.SDN/컨테이너 네트워킹

- ❖ **Performance of Container Networking Technologies**
- ❖ **Linux Bridge, MACVLAN, OVS, Native host**



Latency dominates request/response speed.

CHC single flow throughput with core-binding settings.

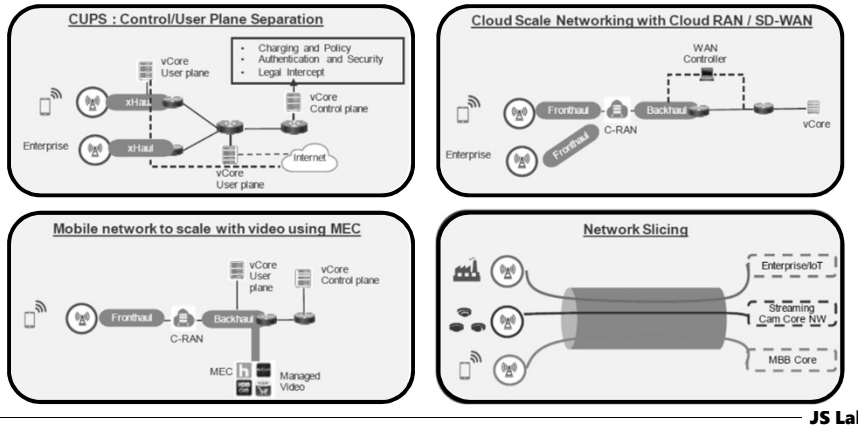
CHHC single flow throughput with core-binding settings.

HotConNet'17, August 25, 2017, Los Angeles, CA, USA

james@jslab.kr

VII.SDN/컨테이너 네트워킹

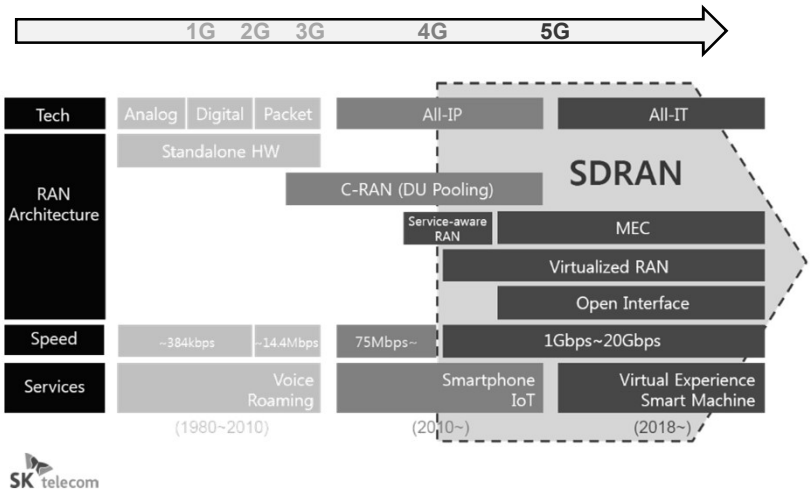
- ❖ **5G Enabling Technologies:**
  - **CUPS** (제어/사용자 플레인 분리)
  - **Cloud Scale** (클라우드 스케일)
  - **MEC** (모바일 에지 컴퓨팅)
  - **Network Slicing** (네트워크 슬라이싱)



james@jslab.kr

VII.SDN/컨테이너 네트워킹

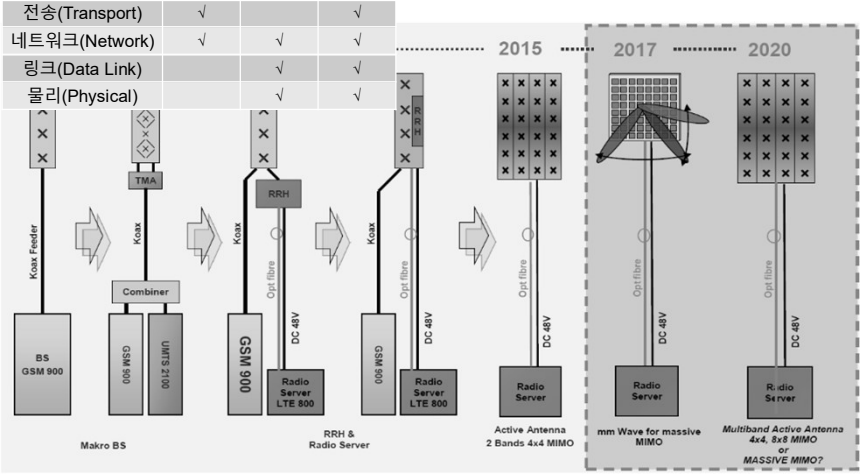
❖ 5G의 소프트웨어 정의 무선 환경



VII.SDN/컨테이너 네트워킹

❖ 무선 통신의 소프트웨어 정의화

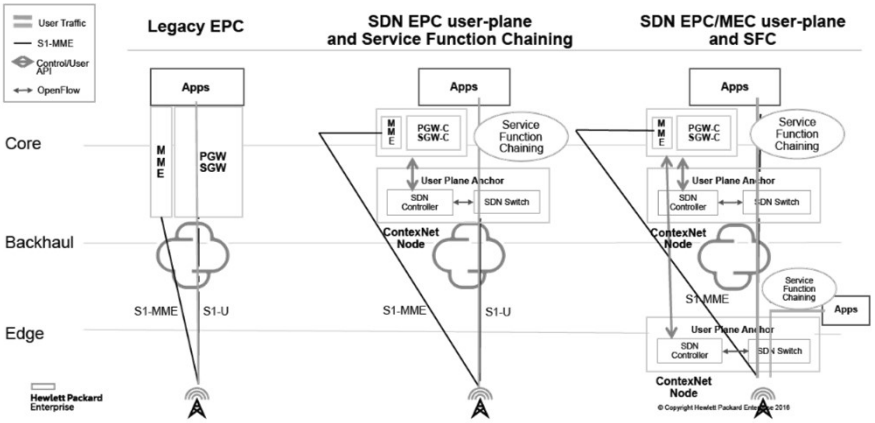
Layer \ Paradigm	SDN	SDR	가상화
전송(Transport)	√		√
네트워크(Network)	√	√	√
링크(Data Link)		√	√
물리(Physical)		√	√



SDR Forum: http://www.sdrforum.org → http://www.wirelessinnovation.org/

VII.SDN/컨테이너 네트워킹

❖ **Telecom 환경의 SDN 구성 : HPE의 무선(Radio) 영역 제시**



EPC(Evolved Packet Core)

james@jslab.kr

VII.SDN/컨테이너 네트워킹

❖ **지역간 경유 터널링을 위한 가속 기능 내장 PaaS** (or K8s)

- **광전송망 공유 유럽 국가간 지연 단축** (Public Cloud 제공 예)
- **통신 프로토콜 최적화 Site 가속 기능 제공**
- **엔터프라이즈 시장을 위한 제조사의 Hybrid Cloud 솔루션 응용**
- **게임과 메신저는 지역간 처리 위한 터널링 기반의 가속 기능과 보안 필요**

지역간 경유 터널링을 위한 가속 기능 내장 Public Cloud 제공 (Tencent Cloud 예)

	Dalas	San Jose	Toronto	Washington	Amsterdam	Frankfurt	London	Chennai	HongKong	Singapore	Sydney	Tokyo	Sao Paulo	Shanghai	Beijing	Guangzhou
Dalas		36	31	31	109	122	108	245	182	214	167	133	143	156	176	192
San Jose	36		56	59	137	144	133	209	142	166	146	97	178	120	140	154
Toronto	31	56		22	88	92	82	217	209	240	192	133	127	176	196	218
Washington	32	59	22		80	87	73	217	215	246	197	136	116	179	199	225
Amsterdam	109	137	90	80		6	8	147	204	163	254	258	182	234	251	216
Frankfurt	123	145	92	87	6		13	138	160	186	281	223	193	201	210	170
London	108	133	82	73	8	13		126	199	159	263	211	183	229	244	209
Chennai	245	209	219	216	144	146	136		66	33	127	105	311	96	111	76
HongKong	182	144	208	215	206	160	199	66		34	114	47	328	30	45	10
Singapore	214	166	240	246	160	186	162	33	34		96	76	359	64	79	44
Sydney	167	146	192	197	268	279	259	127	114	96		113	309	144	159	124
Tokyo	129	98	133	136	253	223	211	105	50	79	113		276	80	92	57
Sao Paulo	142	179	127	116	182	192	182	312	328	359	309	276		298	318	338
Shanghai	156	120	176	179	236	201	229	96	30	64	144	77	298		30	30
Beijing	176	140	196	199	251	210	244	111	45	79	159	92	318	30		40
Guangzhou	192	154	218	225	216	170	209	76	10	44	124	57	338	30	40	
Seoul	163	127	180	185	264	270	259	98	36	68	143	31	303	66	76	46

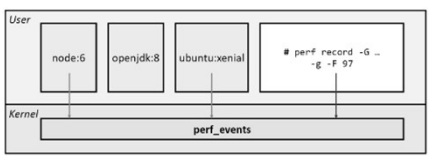
james@jslab.kr

목차

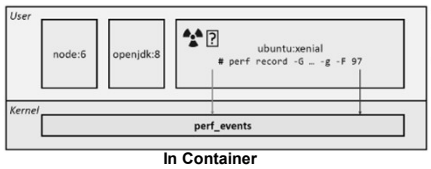
VIII.클라우드 인프라 관리 도구

❖**Tool Deployment**

- **호스트 내 설치** (On The Host)
- **컨테이너 내 설치** (In Container)
- **도구 전용 컨테이너** (“Sidecar” Container)

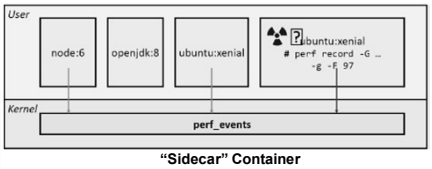


On The Host



In Container

SELA GROUP



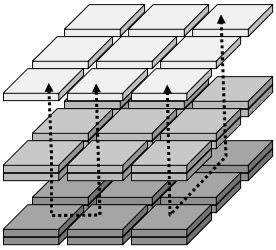
“Sidecar” Container

VIII.클라우드 인프라 관리 도구

❖컨테이너를 위한 인프라 보안/성능/운영 고려 관리도구 필요

- 컨테이너와 이의 오케스트레이션을 포함하는 분산 서비스 계층
- 베어메탈과 가상머신을 포함하는 컴퓨트 계층
- 언더레이/오버레이 네트워크 계층

james@jslab.kr

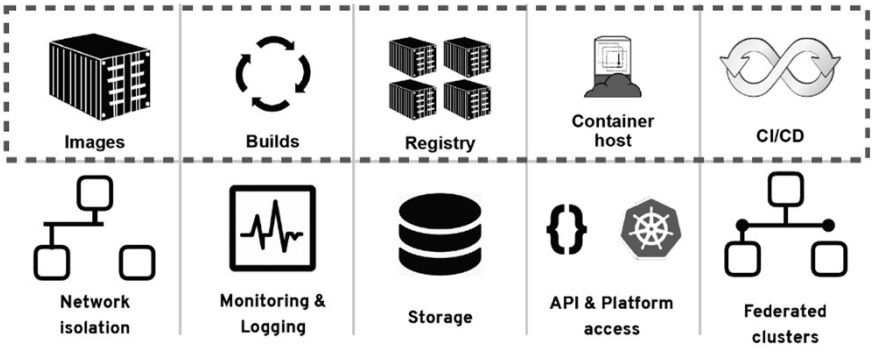


VIII.클라우드 인프라 관리 도구

❖보안 관리 요소

- 추가 고려: Images, Builds, Registry, Container Host, CI/CD

james@jslab.kr



VIII.클라우드 인프라 관리 도구

❖종단간 소프트웨어 제공 체인 관리

james@jslab.kr

빌드 (Build)

- 확인, 사인, 스캔
- 안전한 이미지 저장
- 안전한 민감 데이터

CI/CD

- 검증 가능한 보호 체인
- 정책 기반 자동화

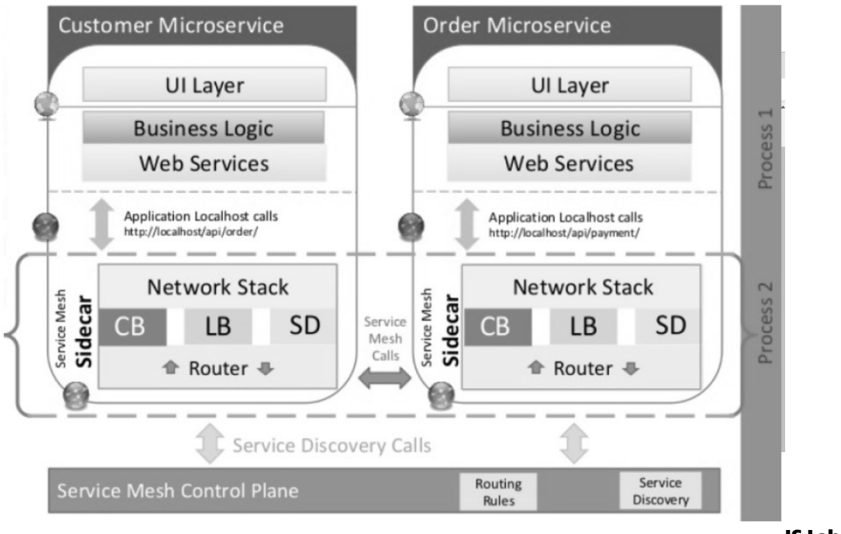
Run

- 운영보안

VIII.클라우드 인프라 관리 도구

❖ **Service Mesh 관리** (예: Sidecar Design Pattern 기반 관리)

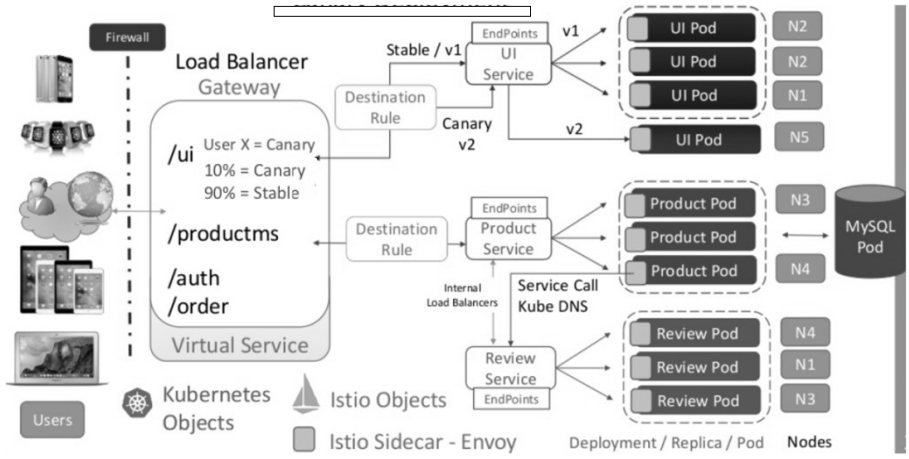
james@jslab.kr



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

VIII.클라우드 인프라 관리 도구

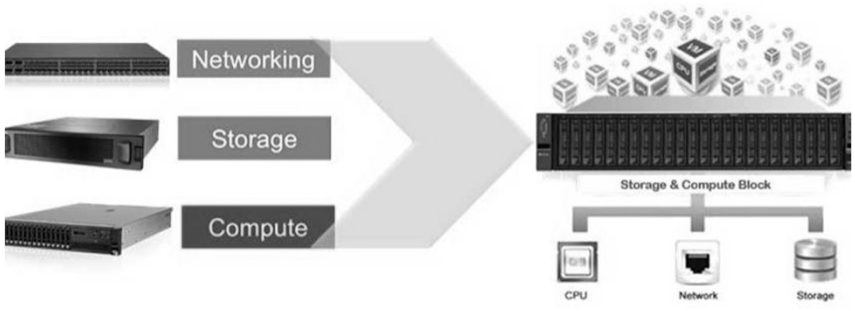
❖ **서비스 업그레이드 운영** (예: A/B Testing, Canary Deployment)



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

VIII.클라우드 인프라 관리 도구

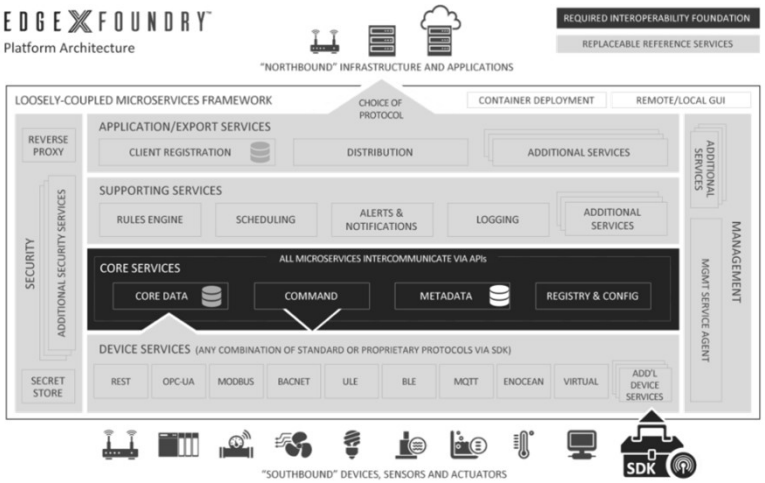
❖ **SDS** (Software-Defined Storage)
❖ **HCI** (Hyper Converged Infrastructure) **is a SDS architecture that combines the compute, storage, and networking functions.** (all-in-one product)



https://www.sdxcentral.com/data-center/hyperconverged/definitions/hci-mec/?c_action=related_articles

VIII.클라우드 인프라 관리 도구

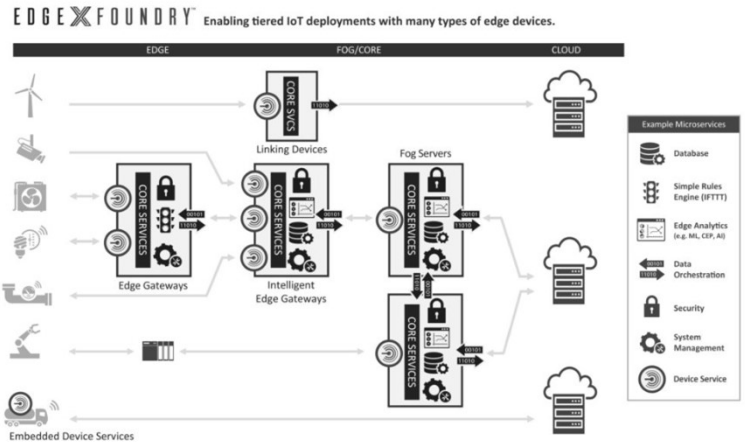
- ❖ 아키텍처 기반의 도구 고려
- ❖ 예: Loosely-Coupled Microservice Platform Architecture



https://www.edgexfoundry.org/about/

VIII.클라우드 인프라 관리 도구

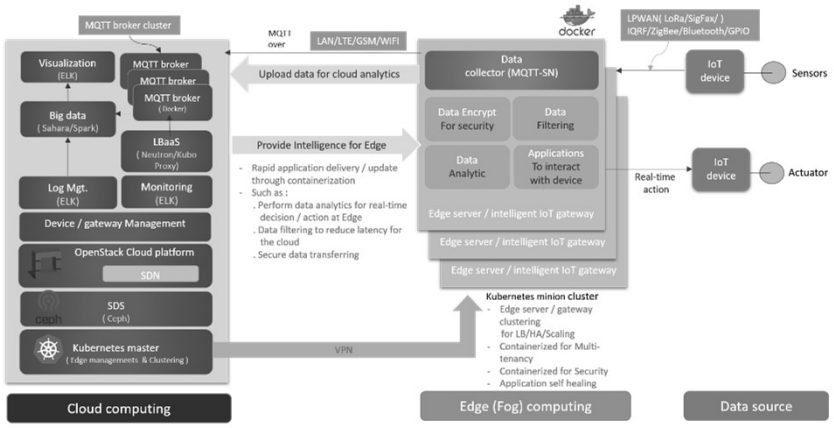
- ❖ 적용 방법 고려
- ❖ Optional Reference Services



https://www.edgexfoundry.org/about/

VIII.클라우드 인프라 관리 도구

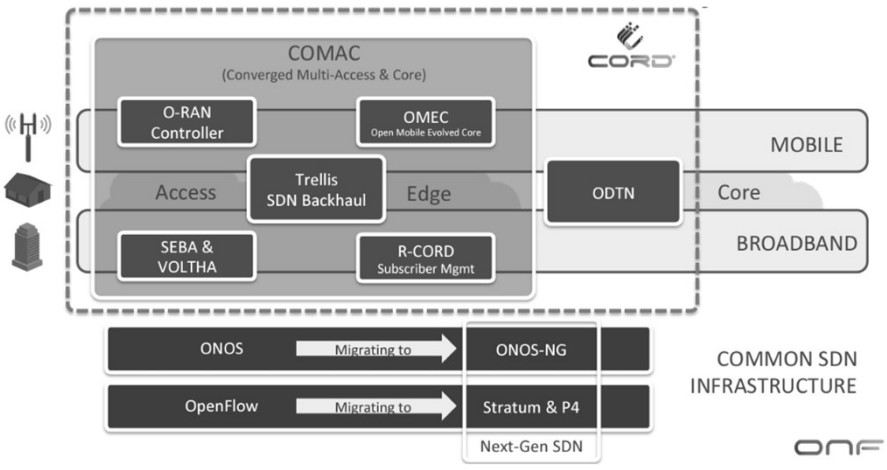
❖ **Edge Computing / Machine Learning 고려**
❖ **예: inwinstack**



https://www.inwinstack.com/en/products/

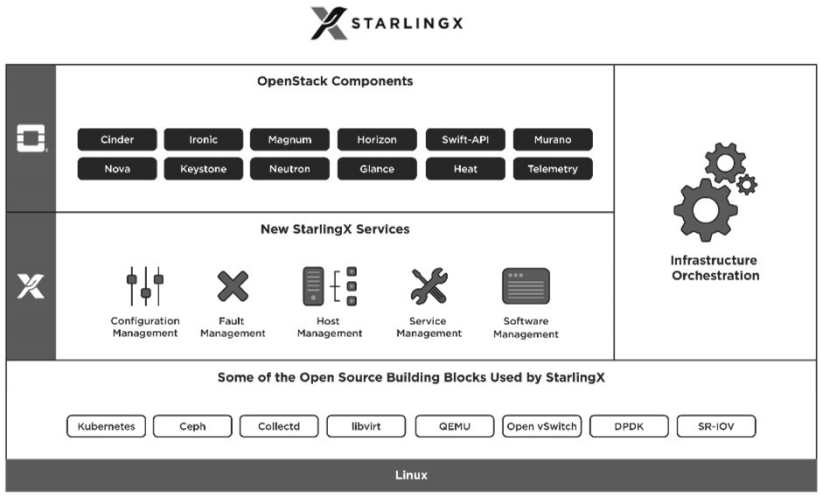
VIII.클라우드 인프라 관리 도구

❖ **COMAC 환경 고려**



VIII.클라우드 인프라 관리 도구

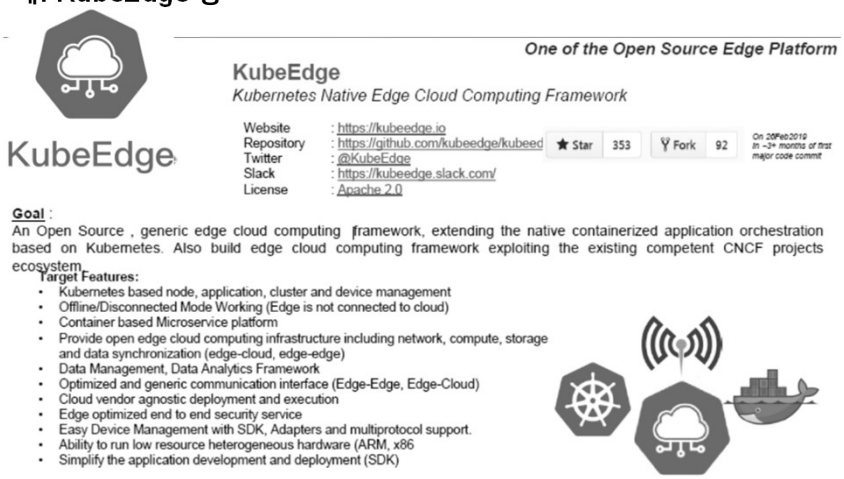
❖ 인프라 오케스트레이션 발전 방향 고려



VIII.클라우드 인프라 관리 도구

❖ 기술 변화 고려

❖ 예: KubeEdge 등

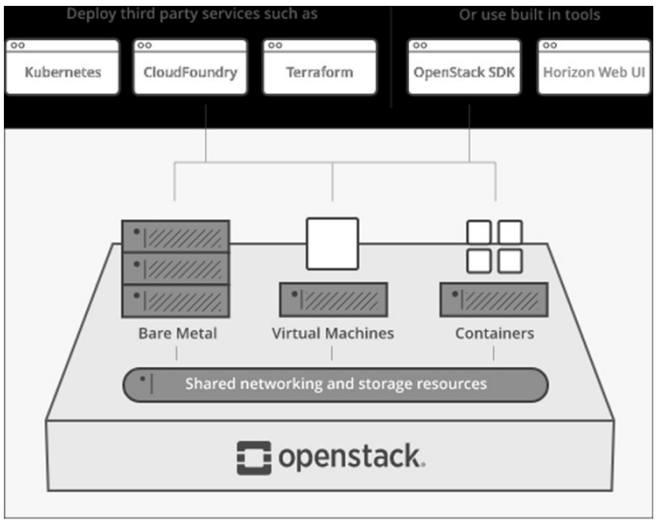


목차

james@jslab.kr

부록 1. OpenStack

❖ WHAT IS OPENSTACK?



https://www.openstack.org/software/

james@jslab.kr

JS Lab

부록 1. OpenStack

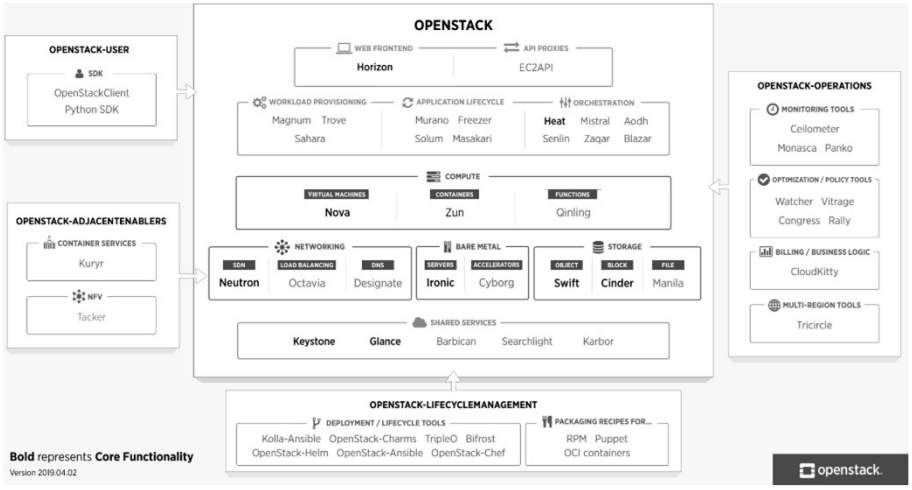
❖ **WHAT IS OPENSTACK?**

- 클라우드 환경에서 컴퓨팅 자원과 스토리지 인프라를 셋업하고 구동하기 위해 사용하는 오픈 소스 소프트웨어 프로젝트의 집합
- OpenStack은 공용 (Public) 클라우드와 사설 (Private) 클라우드 구축을 가능하게 하는 오픈 소스 소프트웨어
- OpenStack은 서버, 스토리지, 네트워크들과 같은 자원들을 모두 모아, 이들을 제어하고 운영하기 위한 클라우드 Operating System
- OpenStack은 오픈 소스를 기반으로 클라우드를 구축하고 운용하고자 하는 오픈 소스 개발자, 회사, 사용자들이 주축이 되어 발전하는 커뮤니티
- IaaS 형태의 클라우드 컴퓨팅 오픈 소스 프로젝트로 컴퓨팅, 스토리지, 네트워킹 자원을 관리하는 여러 개의 하위 프로젝트들로 이루어짐

https://www.openstack.org/software/

부록 1. OpenStack

❖ **THE OPENSTACK LANDSCAPE**



https://www.openstack.org/software/

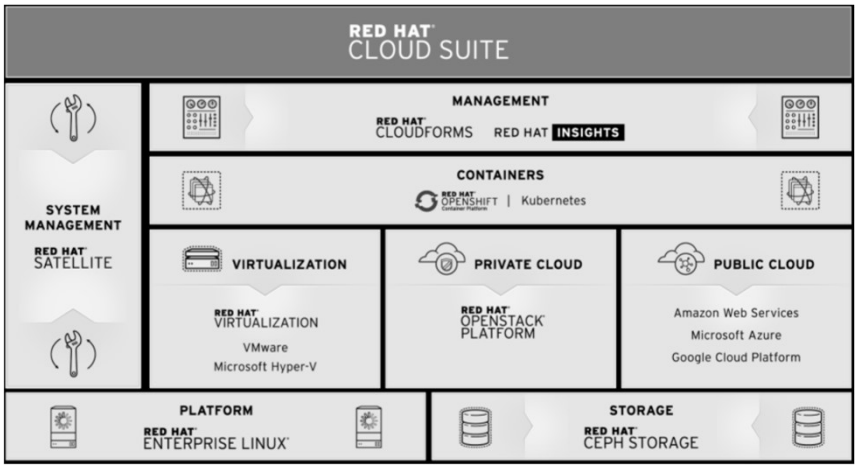
부록 1. OpenStack

❖ 6개월 주기로 릴리즈 발표

릴리즈 이름	릴리즈 일자	포함된 컴포넌트 코드 이름
Austin	21 October 2010	Nova, Swift
Bexar	3 February 2011	Nova, Glance, Swift
Cactus	15 April 2011	Nova, Glance, Swift
Diablo	22 September 2011	Nova, Glance, Swift
Essex	5 April 2012	Nova, Glance, Swift, Horizon, Keystone
Folsom	27 September 2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Grizzly	4 April 2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Havana	17 October 2013	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer
Icehouse	17 April 2014	+ Trove
Juno	16 October 2014	+ Sahara
Kilo	30 April 2015	+ Ironic
Liberty	15 October 2015	+ Zaqar, Manila, Designate, Barbican, Searchlight
Mitaka	7 April 2016	+ Magnum
Newton	6 October 2016	
Ocata	22 February 2017	
Pike	30 August 2018	+ >10 components
Queens	**28 February 2018**	
Rocky	2018년 8월 예정	

부록 1. OpenStack

❖ 응용 상용화 (예: Redhat)

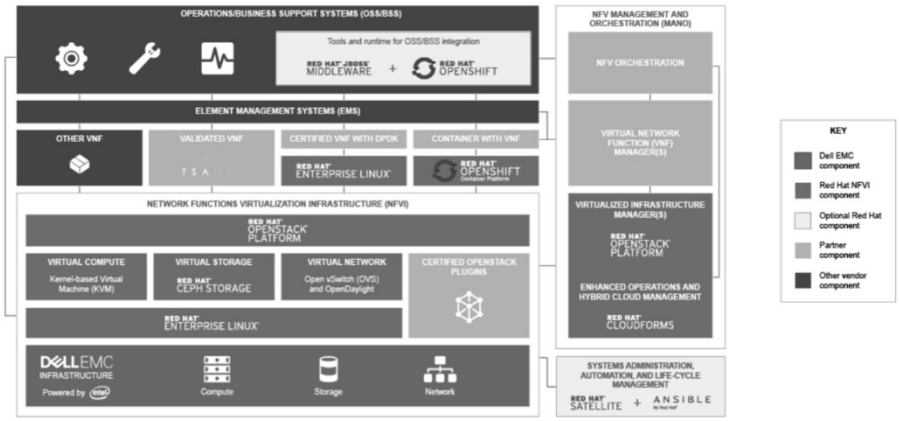


james@jslab.kr

JS Lab

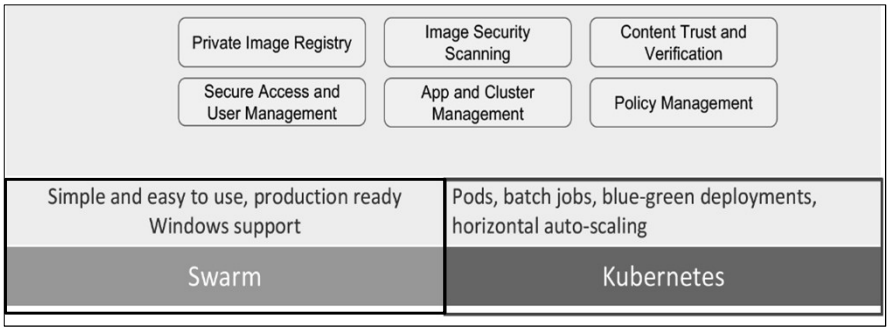
부록 1. OpenStack

❖ 상용화 Use Case (예: Dell EMC)



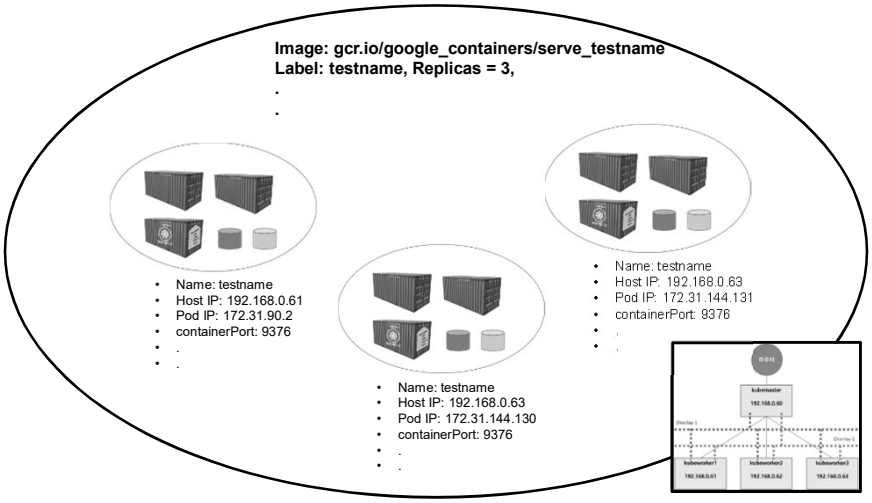
부록 2. Kubernetes (K8s)

❖ Swarm and Kubernetes



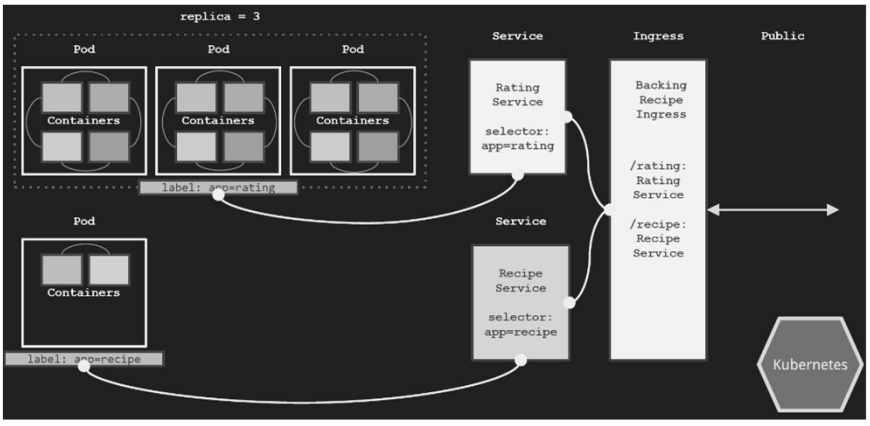
부록 2. Kubernetes (K8s)

❖ **K8s Pod 구성(예): 3**개 노드에 3개 Pod 복제(Replica) 구성 (예)



부록 2. Kubernetes (K8s)

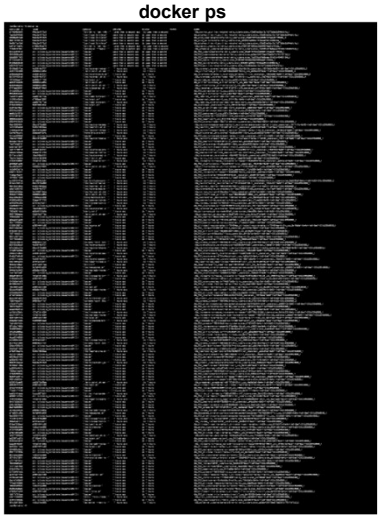
❖ **Kubernetes-based microservice**



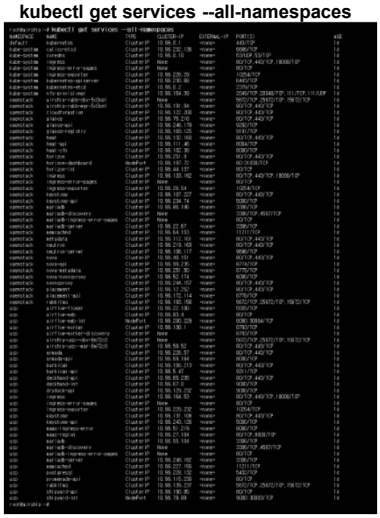
부록 2. Kubernetes (K8s)

❖ **Docker and Kubernetes for Airship**

docker ps

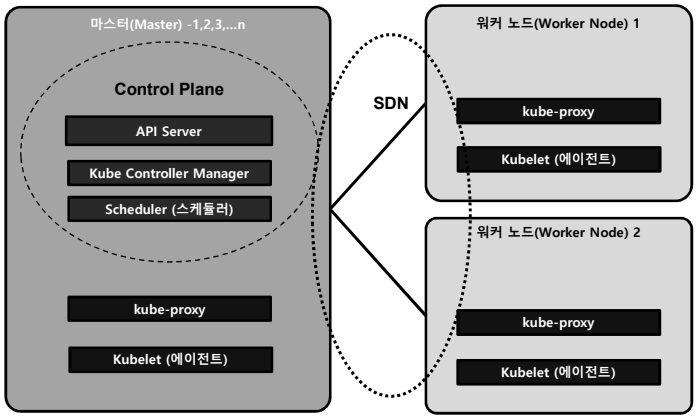


kubectl get services --all-namespaces



부록 2. Kubernetes (K8s)

❖ **K8s 기본 구성:** HA 가능 마스터 노드(Master Node)와 실제 앱이 구동하는 컨테이너들의 Pod를 호스팅하는 1개 이상의 워커 노드(Worker Node)로 클러스터(Cluster) 구성 (**실행 단위는 모두 컨테이너임)

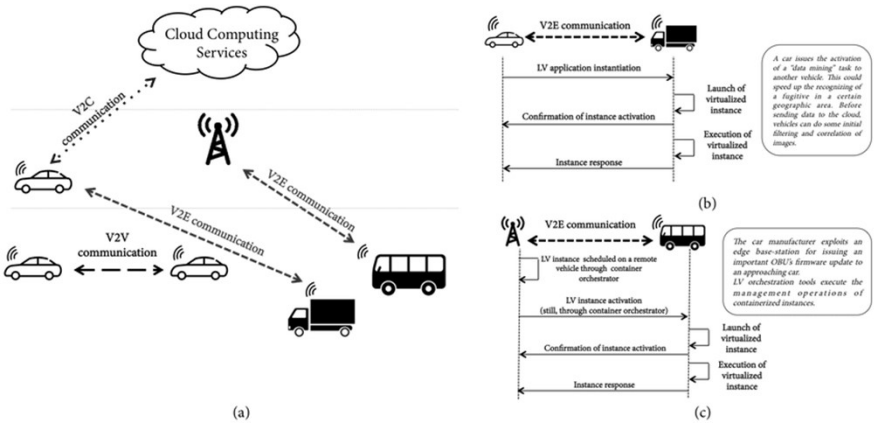


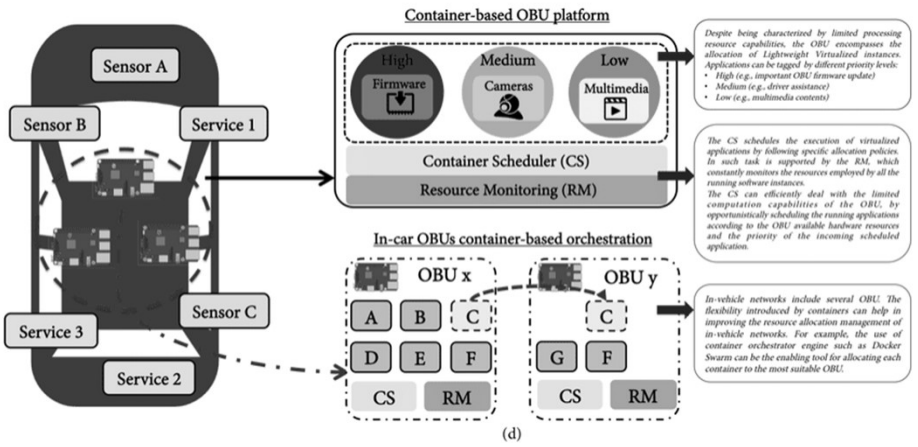
목차

james@jslab.kr

james@jslab.kr

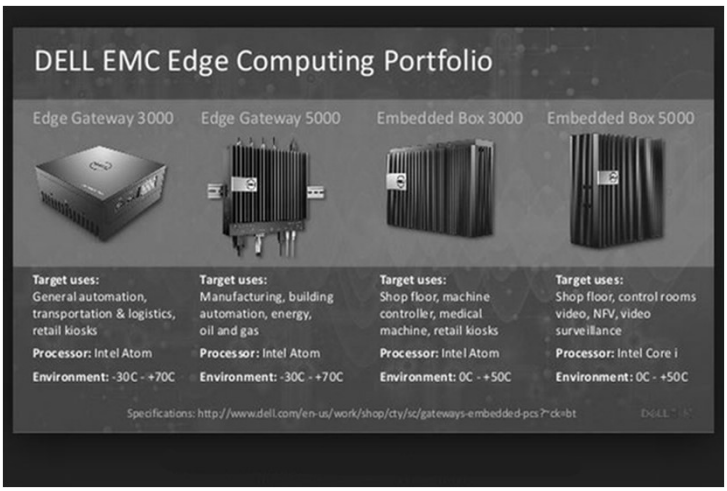
JS Lab





I. Edge Computing 개요

❖ **Optional Reference Services**

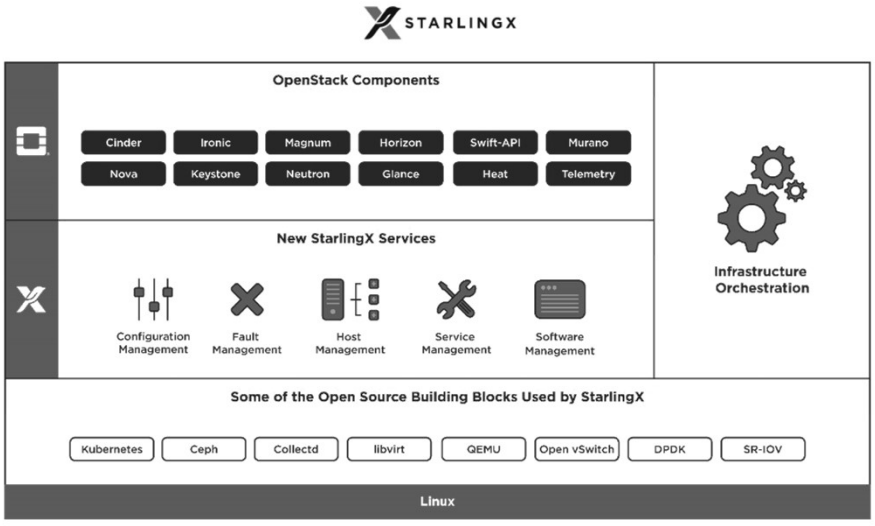


https://www.edgexfoundry.org/about/

VIII.클라우드 인프라 관리 도구

❖ **Eclipse IoT has 30+ open source project**
❖ **Eclipse Kura, Eclipse ioFog and Eclipse fogO5 focus on edge computing**
❖ **Implementations of key standards: MQTT, OPC-UA, DDS, CoAP, LWM2M**
❖ **30+ members, including Bosch, Red Hat, Eurotech, ADLink**

james@jslab.kr



james@jslab.kr

One of the Open Source Edge Platform

KubeEdge

Kubernetes Native Edge Cloud Computing Framework

KubeEdge

Website : https://kubeedge.io
Repository : https://github.com/kubeedge/kubeed ★ Star 353 Fork 92
Twitter : @KubeEdge
Slack : https://kubeedge.slack.com/
License : Apache 2.0

On 26Feb2019 in ~3+ months of first major code commit

Goal :
An Open Source , generic edge cloud computing framework, extending the native containerized application orchestration based on Kubernetes. Also build edge cloud computing framework exploiting the existing competent CNCF projects ecosystem.

Target Features:

- Kubernetes based node, application, cluster and device management
- Offline/Disconnected Mode Working (Edge is not connected to cloud)
- Container based Microservice platform
- Provide open edge cloud computing infrastructure including network, compute, storage and data synchronization (edge-cloud, edge-edge)
- Data Management, Data Analytics Framework
- Optimized and generic communication interface (Edge-Edge, Edge-Cloud)
- Cloud vendor agnostic deployment and execution
- Edge optimized end to end security service
- Easy Device Management with SDK, Adapters and multiprotocol support.
- Ability to run low resource heterogeneous hardware (ARM, x86
- Simplify the application development and deployment (SDK)

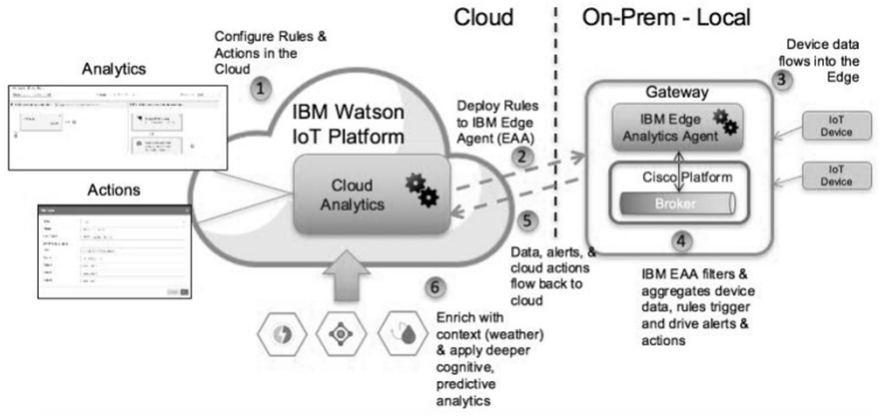
I. Edge Computing

- ❖ Eclipse IoT has 30+ open source project
- ❖ Eclipse Kura, Eclipse ioFog and Eclipse fogO5 focus on edge computing
- ❖ Implementations of key standards: MQTT, OPC-UA, DDS, CoAP, LWM2M
- ❖ 30+ members, including Bosch, Red Hat, Eurotech, ADLink

james@jslab.kr

I. Edge Computing 개요

- ❖ Edge Analytics in Watson IoT Platform



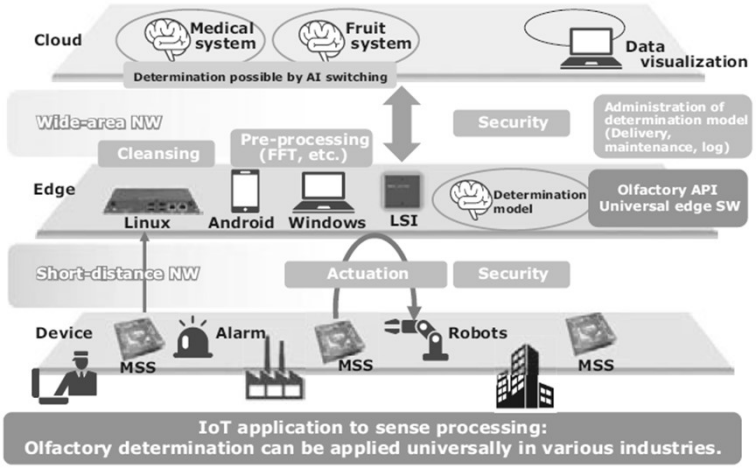
https://developer.ibm.com/recipes/tutorials/getting-started-with-edge-analytics-in-watson-iot-platform/

james@jslab.kr

JS Lab

I. Edge Computing 개요

❖ **Edge computing in olfactory IoT**



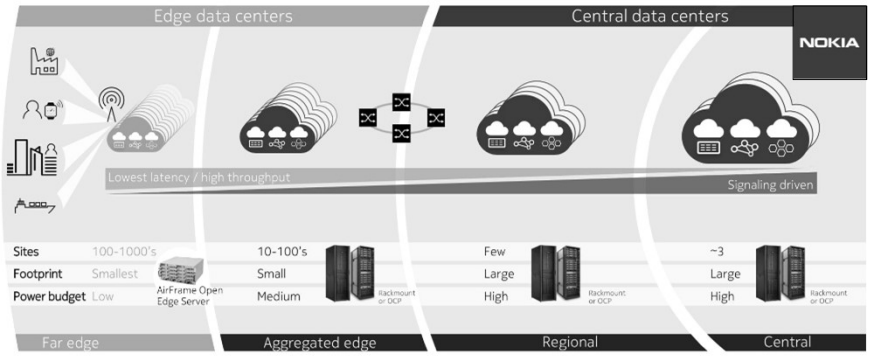
JS Lab

https://www.nec.com/en/global/techrep/journal/g17/n01/170106.html

I. Edge Computing 개요

❖ **Edge Cloud Computing:** 에지(Edge)에서 클라우드 서비스 기반의 컴퓨팅을 제공

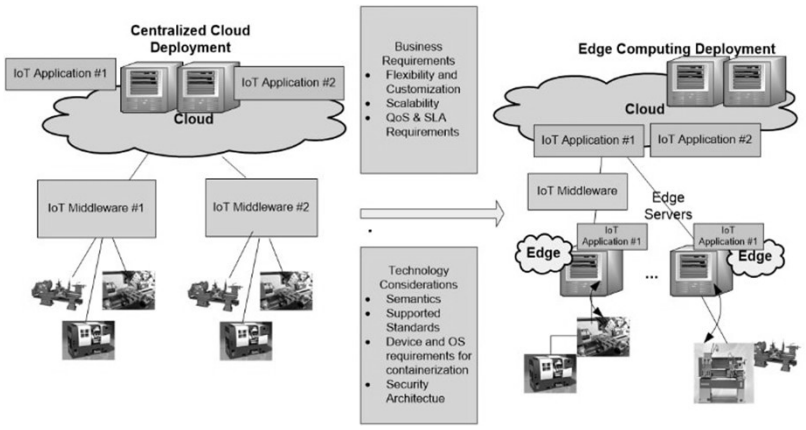
- **지연 개선** (Reducing latency)
- **대역폭 부족 완화** (Mitigating bandwidth limits)



JS Lab

https://networks.nokia.com/solutions/edge-cloud

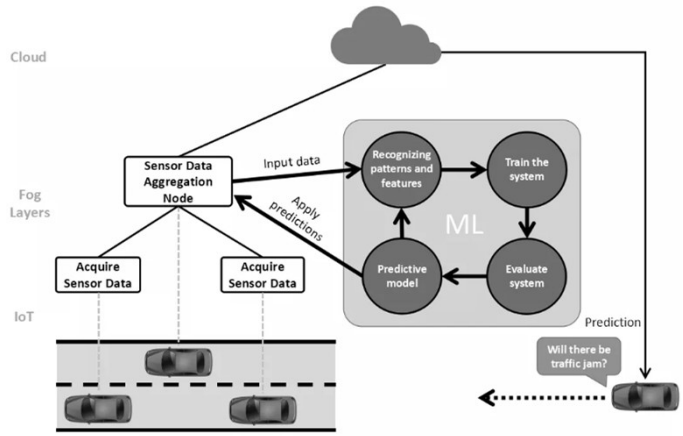
❖ **IoT at the Edge (semiwiki)**



https://www.semiwiki.com/forum/content/5935-iot-tutorial-chapter-6-iot-edge.html

james@jslab.kr

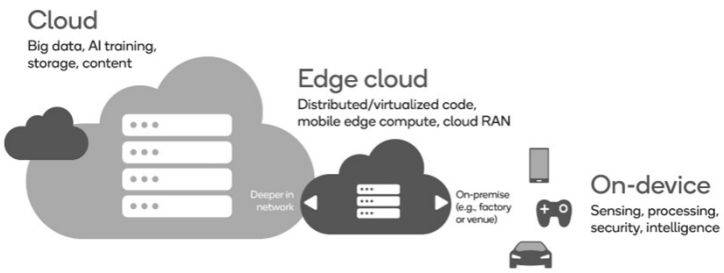
❖ **Complications Between Edge Computing and Machine Learning**
❖ **(Software engineering daily)**



https://softwareengineeringdaily.com/2018/09/14/edge-computing-and-the-future-of-the-cloud/

james@jslab.kr

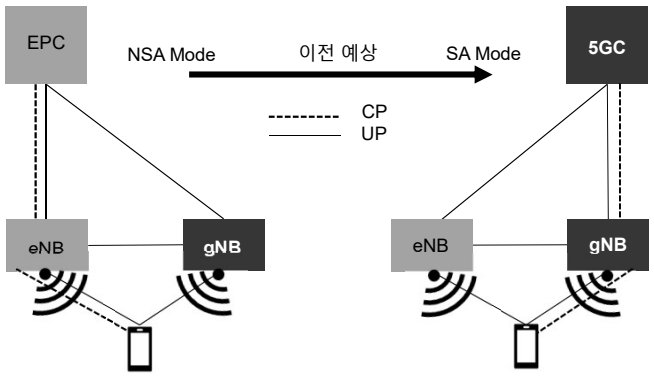
❖ **Wireless Edge (Qualcomm Technologies)**



https://developer.qualcomm.com/blog/taking-your-development-wireless-edge

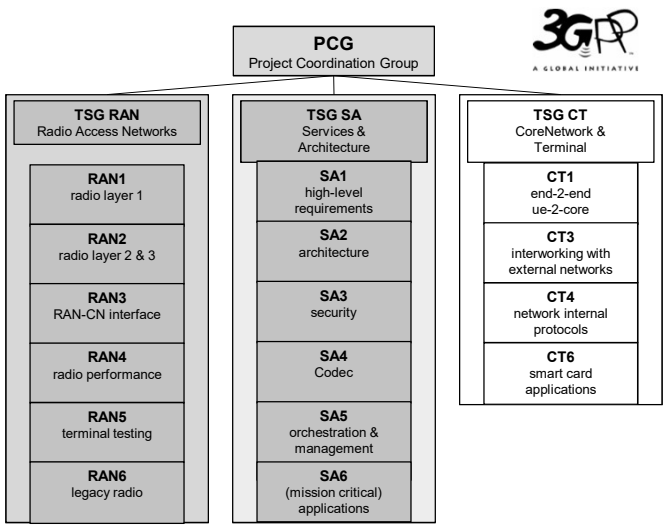
Cloud Edge Computing: Beyond the Data Center

❖ **OpenStack Foundation의 Edge Computing Group**



I. 5G 개요

❖ 3GPP



I. 5G 개요

❖ Mobile Network Evolution (in Logos)

I. 5G 개요

❖ **AI for Cell Selection**

❖ **36.304, cell reselection procedure:**

- **35parameters for system information**
- **10parameters for speed dependent selection**
- **13parameters for interworking**
- **The list is getting larger:** New technologies -beam sweeping, New services

james@jslab.kr

Without AI	With AI
Periodically measured	Triggered measurement
Threshold based (lots of parameters/configurations)	No threshold
Static configurations	Real-time, adapted to changes of the context (e.g., speed)

I. Edge Computing 개요

❖ **Edge Computing**

- **Top 10 Strategic Technology Trends for 2019: #5 Empowered Edge**
  (Gartner)
- **"Get Ready For Edge Computing's Rise In 2019"**
  (Forrester)
- **"The Edge Will Eat The Cloud"**
  (Gartner)

james@jslab.kr


james@jslab.kr
EDGEXFOUNDRY
AKRAINO EDGE STACK
STARLINGX
FOREST GIANT
HOME EDGE
KubeEdge
OPNFV
ONAP
kura
JS Lab

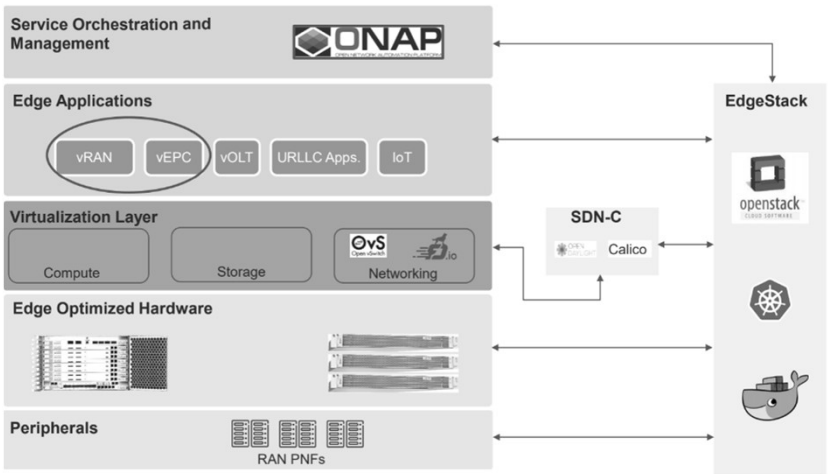

james@jslab.kr
LF EDGE
AKRAINO EDGE STACK
OPEN GLOSSARY OF EDGE COMPUTING
EDGEXFOUNDRY
HOME EDGE
EDGE VIRTUALIZATION ENGINE
Premier
arm
AT&T
Baidu
DELLEMC
DIANOMIC
ERICSSON
hp
Hewlett Packard Enterprise
HUAWEI
IBM
intel
JUNIPER
MobiledgeX
NETSIA
NOKIA
NTT
OSIsoft
Qualcomm
Radisys
redhat
SAMSUNG
SEAGATE
Tencent
WIND
wipro
ZEDEDA
General
ubuntu
CertusNet
IOTech
Linaro
packet
redislabs
REPLY
Vapor
vitro
Associate
AECC
ETRI
JS Lab

AKRAINO EDGE STACK	• AT&T와 인텔이 시작(리눅스 재단 프로젝트, 2018년 2월) • 에지를 위한 고가용 클라우드 서비스 최적화 지원 • https://www.edgexfoundry.org/	
EDGE X FOUNDRY	• IoT 에지를 위한 오픈플랫폼 (리눅스 재단 프로젝트, 2017년 4월) • https://www.edgexfoundry.org/	
STARLINGX	• 인텔과 윈드리버가 공동 오픈소스 개발 (오픈스택 재단) • 분산에지를 위한 클라우드 기능 • https://wiki.openstack.org/wiki/StarlingX	
OPNFV	• OPNFV의 에지클라우드를 위한 레퍼런스 플랫폼 설계 프로젝트 • https://wiki.opnfv.org/display/PROJ/Edge+cloud	Reference platform design for edge cloud in OPNFV – main focus.
ONAP	• ONAP의 에지 관리를 위한 데이터 수집, 프로세싱, 정책관리, 제어 루프 모델, 보안 • https://wiki.onap.org/display/DW/Edge+Automation+through+ONAP	Looking Edge & ONAP components in terms of data collection, processing, policy management, resource management, control loop models, security.
	• There are several projects under Eclipse Foundation supports IoT and Edge platforms.	There are several projects under Eclipse Foundation supports IoT and Edge platforms.

james@jslab.kr

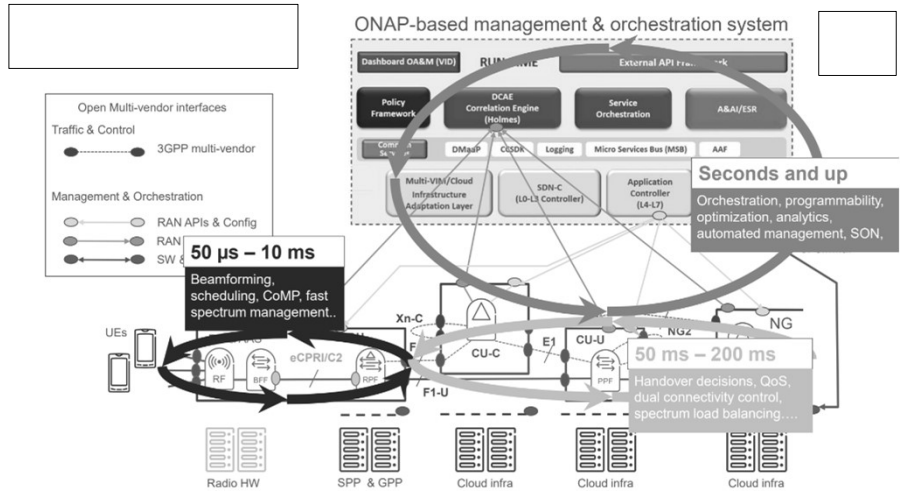
Proposed Edge Stack Requirements

❖ **Project Akraino of LFN**



Proposed Edge Stack Requirements

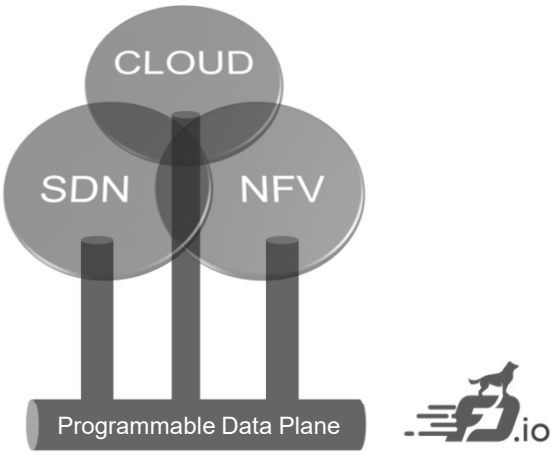
❖ 3-Loop Control (Ericsson 2018)



james@jslab.kr

Evolution of Programmable Networking

❖ FD.io



james@jslab.kr

Evolution of Programmable Networking

❖ Issues/Limitations with Existing Data Plane Solutions

- Known issues with Performance, Scalability & Stability
- Overly Complex Architectures
- Hard to evolve
- Slow rate of innovation
- Steep learning curve
- Hard to deploy/upgrade/operate
- slow cycles, too many kernel dependencies
- Lack of :
- automated end-to-end system testing frameworks
- leads to unpredictable system behavior
- support for diverse/custom hardware
- portability across compute platforms
- optimal use of compute microarchitectures
- network level instrumentation
- Few debugability features
- Few if any Statistics/Counters exposed

james@jslab.kr

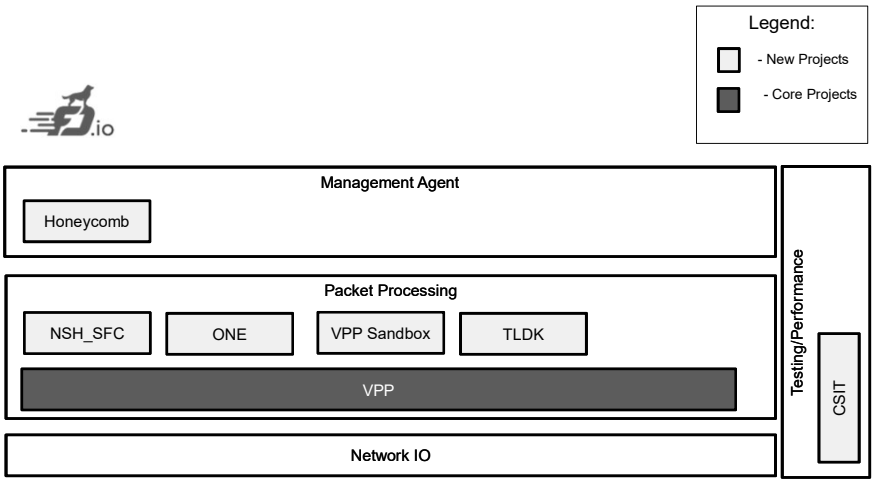
Evolution of Programmable Networking

❖ Fd.io Members

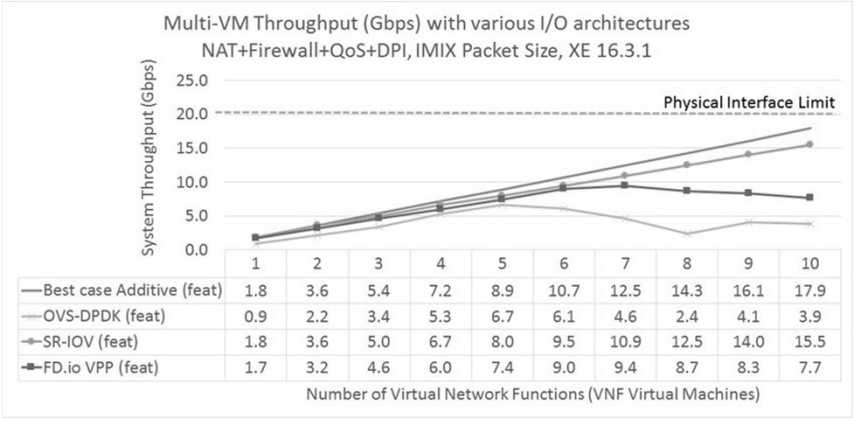
james@jslab.kr

Evolution of Programmable Networking

- ❖ Fd.io Projects
- ❖ VPP: Vector Packet Processing



james@jslab.kr



	1	2	3	4	5	6	7	8	9	10
Best case Additive (feat)	1.8	3.6	5.4	7.2	8.9	10.7	12.5	14.3	16.1	17.9
OVS-DPDK (feat)	0.9	2.2	3.4	5.3	6.7	6.1	4.6	2.4	4.1	3.9
SR-IOV (feat)	1.8	3.6	5.0	6.7	8.0	9.5	10.9	12.5	14.0	15.5
FD.io VPP (feat)	1.7	3.2	4.6	6.0	7.4	9.0	9.4	8.7	8.3	7.7

james@jslab.kr

JS Lab

Dataplane Performance Testing Options

❖ **Daily tests on master and stable branch in OPNFV Lab**

james@jslab.kr

Solution Stack ↑

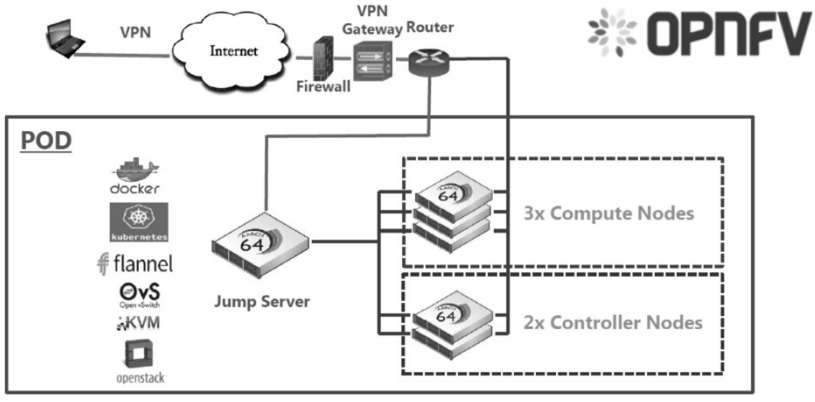
Workload (DUT)	Traffic Generator (HW or SW)	Automation code (Test framework)
Sample VNFs • SampleVNF (vACL, vFW, vCGNAT, …) • Open source VNF catalogue	Hardware - commercial • **Ixia** • Spirent • Xena	Compliance • Dovetail
Test VMs • vloop-vnf (dpdk-testpmd, Linux bridge, L2fwd module) • **Spirent stress-VM** • Virtual Traffic classifier	Virtual - commercial • Ixia • **Spirent**	VIM and MANO • NFVbench
Virtual switching • OVS • **OVS-dpdk** • **VPP**	Software - Open Source • Pktgen • **Moongen** • **TREX** • **PROX**	VIM, no MANO • Yardstick • Qtip • Bottleneck
Physical / virtual interfaces • NIC (**10GE**, 40GE, …) • **Vhost-user** • Pass-through, SR-IOV		No VIM or MANO • **VSPERF** • Storperf
HW offload • TSO • encrypt/decrypt • SmartNIC		***Used in test examples presented**

https://build.opnfv.org/ci/view/vswitchperf/

Dataplane Performance Testing Options

❖ **ARM based OPNFV Container PoC**

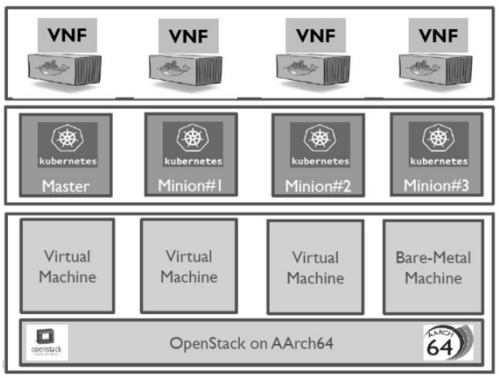
james@jslab.kr



PoC

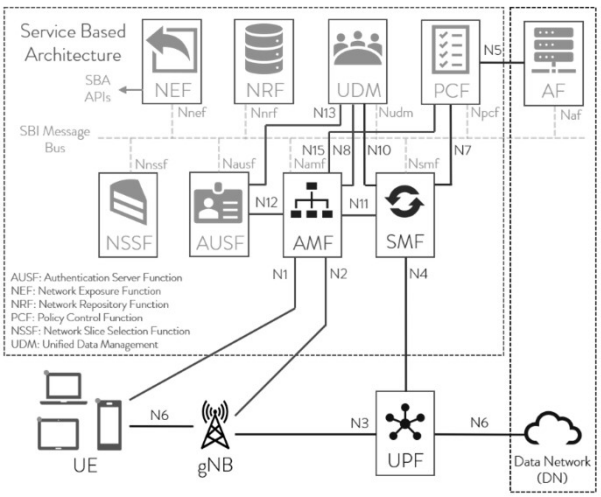
❖ **Containerized VNFs on ARM NFVi**

- **ARM-based OpenStack as VIM**
- **OpenStack Magnum as container service**
- **Kubernetes as COE**
- **Flannel plugin for Kubernetes as CNI**
- **Containerized VNFs within a VM and bare metal, both**



LF Open Source Edge

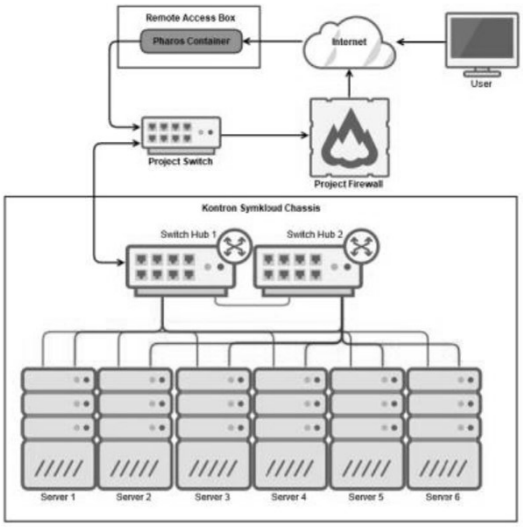
❖ **Standards, Ref Arch and Ref Implementation**



LinuxFoundationX: LFS164x

Pharos Lab

Pharos Lab



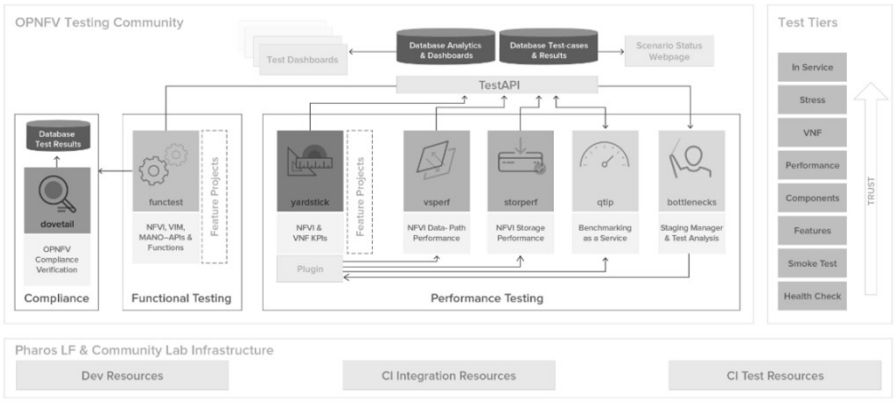
Containers in Telco Cloud with Open Infra

❖ **Project: AIRSHIP**

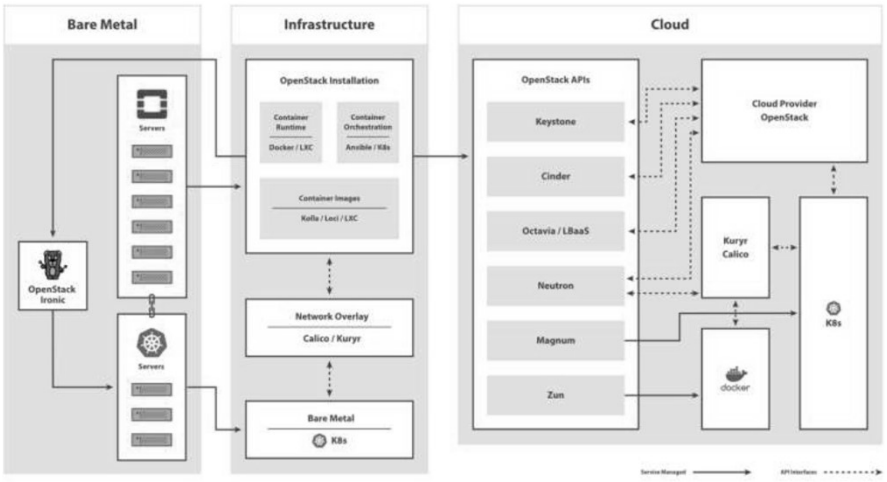
- **Airship, a New Open Infrastructure Project for OpenStack, is Delivering a Unified, Declarative and Cloud-Native Way for Operators to Manage Containerized Software Delivery of Cloud Infrastructure Services**
- **Built on the foundation laid by the OpenStack-Helm project launched in 2017.**
- **The initial focus of this project is the implementation of a declarative platform to introduce OpenStack on Kubernetes (OOK), and the lifecycle management of the resulting cloud, with the scale, speed, resiliency, flexibility and operational predictability demanded of Network Clouds.**

airship

What Are OPNFV Testing Projects? - 2018

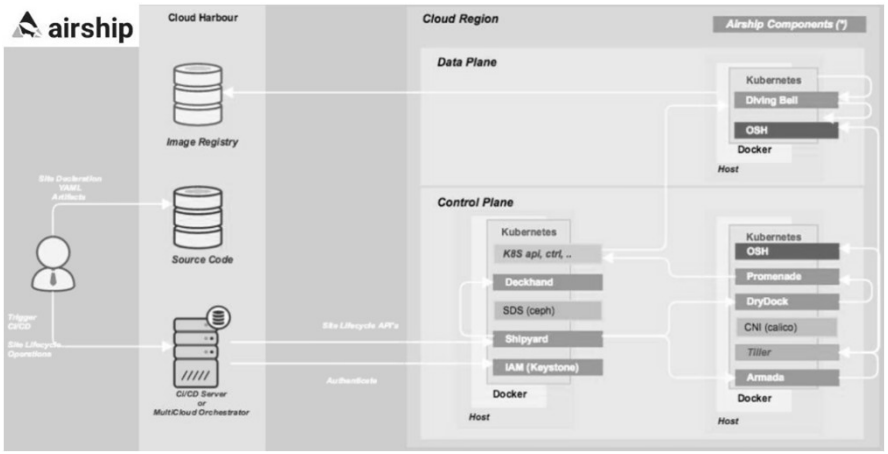


OpenStack Support for Containers



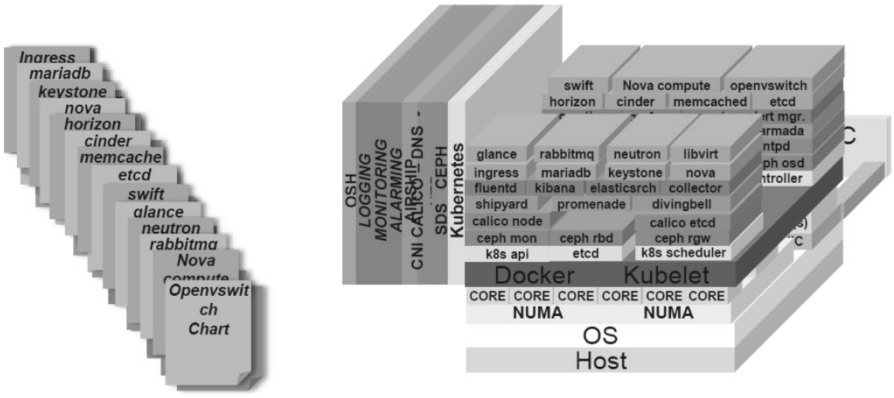
Airship – Architecture Overview

- ❖ Project: AIRSHIP
- ❖ Airship – Architecture Overview



Airship – Architecture Overview

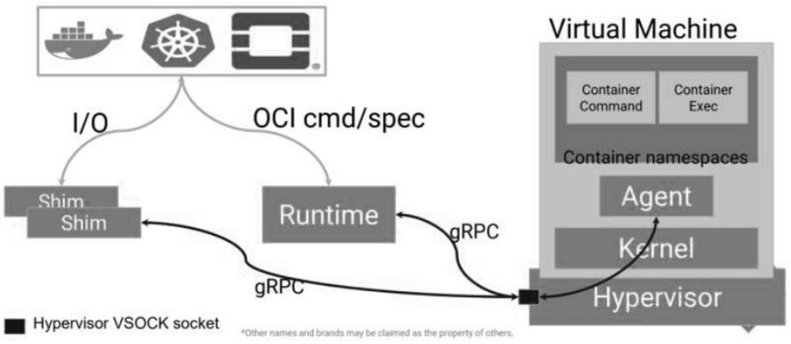
- ❖ Project: AIRSHIP
- ❖ Using AIRSHIP – What it means 'Declarative'



Airship – Architecture Overview

❖ **Container Ecosystem and Seamless integration**

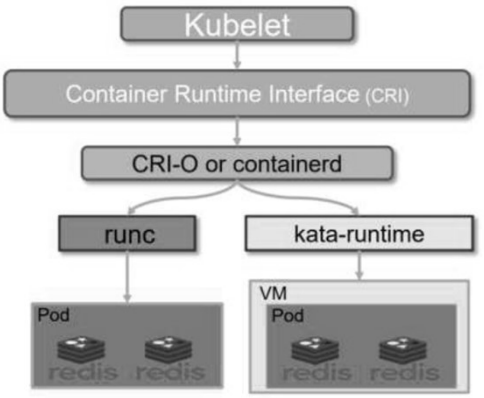
james@jslab.kr



Airship – Architecture Overview

❖ **Container Ecosystem and Seamless integration**

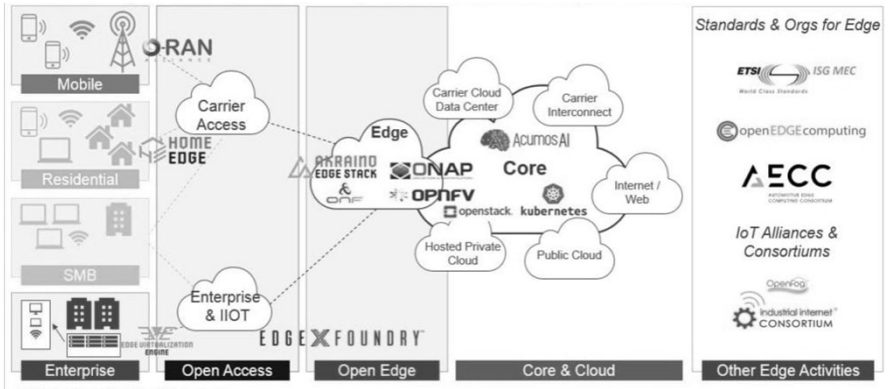
james@jslab.kr



- OpenStack and Kubernetes – Complement each other with their benefits to orchestrate and secure Cloud infrastructure.
- Open Infrastructure initiative from OpenStack Foundation focuses on strengthening the Orchestration and Security for Telco Clouds.
- Airship and Kata Containers are evolving to solve Telco Cloud challenges.

LF Open Source Edge

- Standards, Ref Arch and Ref Implementation



THE LINUX FOUNDATION

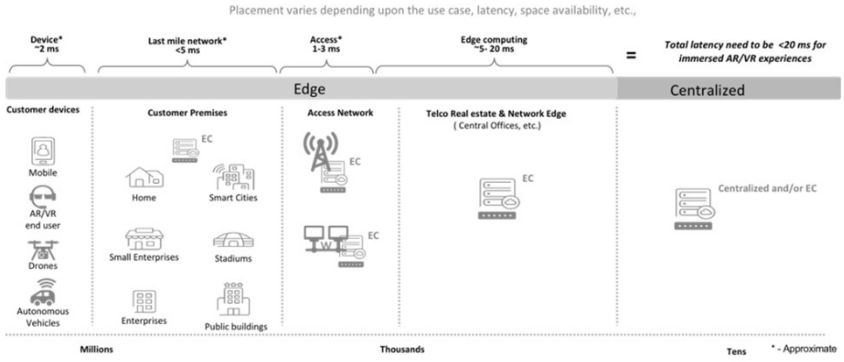
james@jslab.kr

JS Lab

LinuxFoundationX

james@jslab.kr

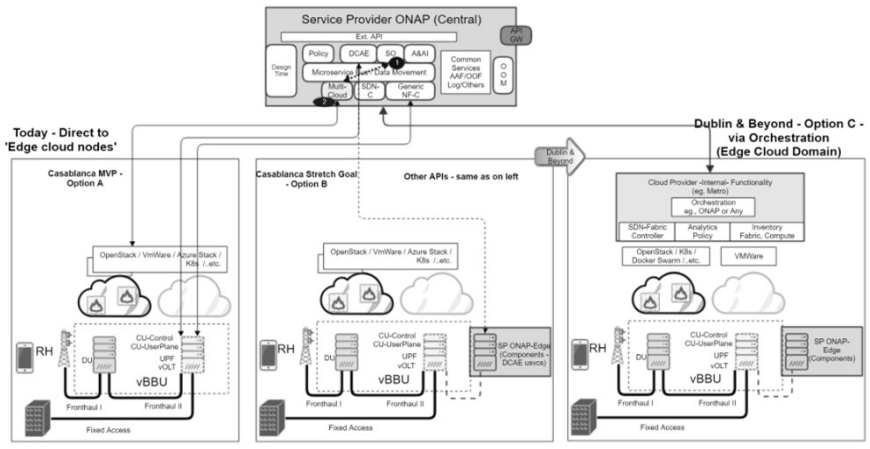
❖ Edge Application Profiles



https://wiki.onap.org/pages/viewpage.action?pageId=28381325

james@jslab.kr

❖ Hierarchical (ONAP Central, Edge) Architecture



https://wiki.onap.org/pages/viewpage.action?pageId=28381325

I. Edge Computing

❖ Open Source Communities

james@jslab.kr

I. Edge Computing

❖ Open Source Communities

james@jslab.kr

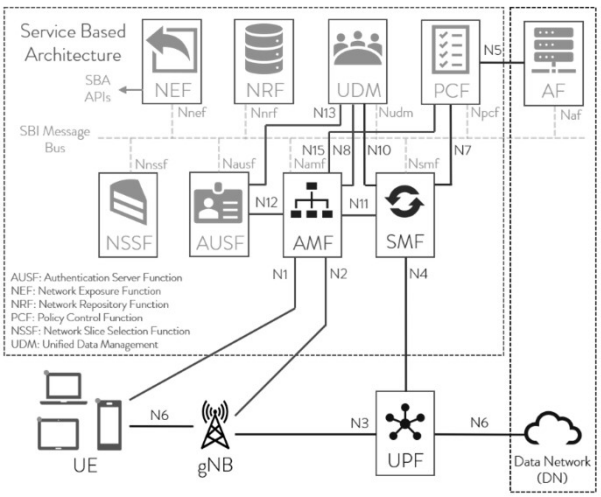
JS Lab

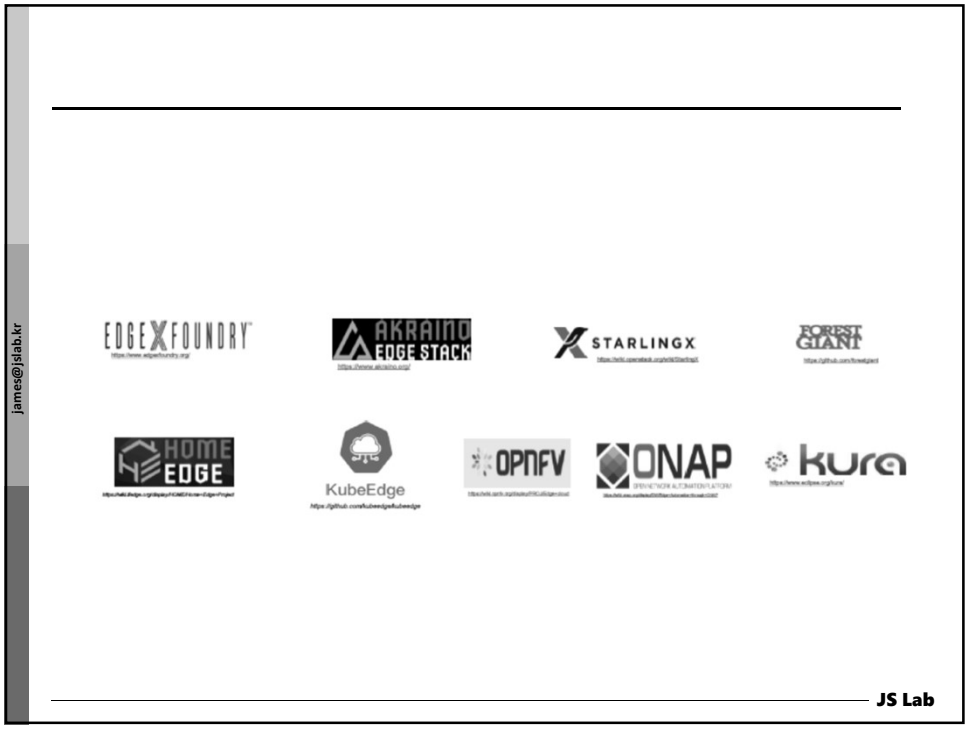
1. Memo

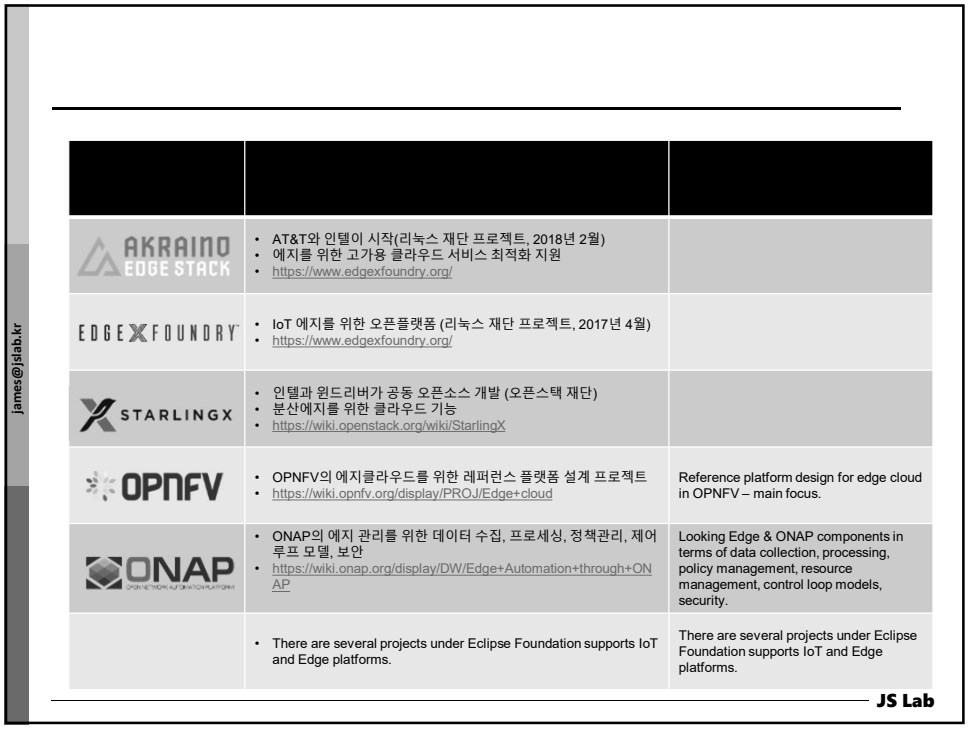
- ❖ 컨테이너(Container)의 Privilege Mode 사용 (SONiC, IoT)
- ❖ 스위치 제조사가 K8s 개발 업체와 협력
- ❖ 5G 통신장비 제조사의 Cloud 전문가와 협업
- ❖ 5G Use Case
  - vCDN @ MEC
  - 국사/기지국
  - SDN for uRLLC
- ❖ Telco Cloud 구성
  - K8s API 아래 이야기

LF Open Source Edge

- ❖ Standards, Ref Arch and Ref Implementation




EDGE X FOUNDRY
AKRAINO EDGE STACK
STARLINGX
FOREST GIANT
HOME EDGE
KubeEdge
OPNFV
ONAP
kura
james@jslab.kr
JS Lab


AKRAINO EDGE STACK
• AT&T와 인텔이 시작(리눅스 재단 프로젝트, 2018년 2월)
• 에지를 위한 고가용 클라우드 서비스 최적화 지원
• https://www.edgexfoundry.org/
EDGE X FOUNDRY
• IoT 에지를 위한 오픈플랫폼 (리눅스 재단 프로젝트, 2017년 4월)
• https://www.edgexfoundry.org/
STARLINGX
• 인텔과 윈드리버가 공동 오픈소스 개발 (오픈스택 재단)
• 분산에지를 위한 클라우드 기능
• https://wiki.openstack.org/wiki/StarlingX
OPNFV
• OPNFV의 에지클라우드를 위한 레퍼런스 플랫폼 설계 프로젝트
• https://wiki.opnfv.org/display/PROJ/Edge+cloud
Reference platform design for edge cloud in OPNFV – main focus.
ONAP
• ONAP의 에지 관리를 위한 데이터 수집, 프로세싱, 정책관리, 제어 루프 모델, 보안
• https://wiki.onap.org/display/DW/Edge+Automation+through+ONAP
Looking Edge & ONAP components in terms of data collection, processing, policy management, resource management, control loop models, security.
• There are several projects under Eclipse Foundation supports IoT and Edge platforms.
There are several projects under Eclipse Foundation supports IoT and Edge platforms.
james@jslab.kr
JS Lab

Backup

❖ SONiC

(4) OCP 기반 네트워킹 소프트웨어 분석

- OCP의 SONiC은 마이크로소프트에서 주도를 하며, 네트워크 기능을 컨테이너화하는 아키텍처를 사용하여 클라우드 네이티브 추세를 수용하는 체계
- OCP 네트워킹 하드웨어에 탑재 가능한 오픈 NOS (네트워크 운영 소프트웨어)는 OpenSwitch 또는 OPS (http://www.openswitch.net), Open Netowk Linux 또는 ONL (http://openlinux.org) 등이 있음
- 제조사 솔루션은 Barefoot Networks, Big Switch Networks, Cumulus Networks, Apstra 등에서 제공하고 있으며, 국내는 Barefoot Networks 솔루션을 선호하는 웹레퓸 환경의 개발자 중심 생태계와 Cumulus Networks 솔루션을 선호하는 엔터프라이즈 시장 생태계가 형성되어 있음 |

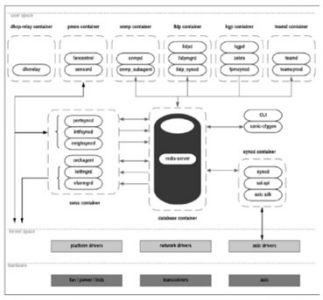
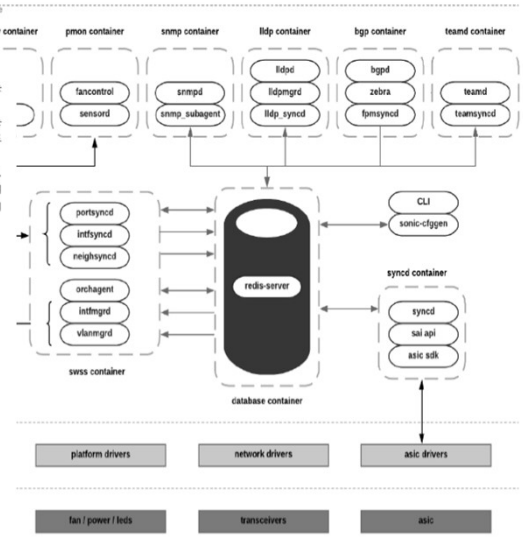
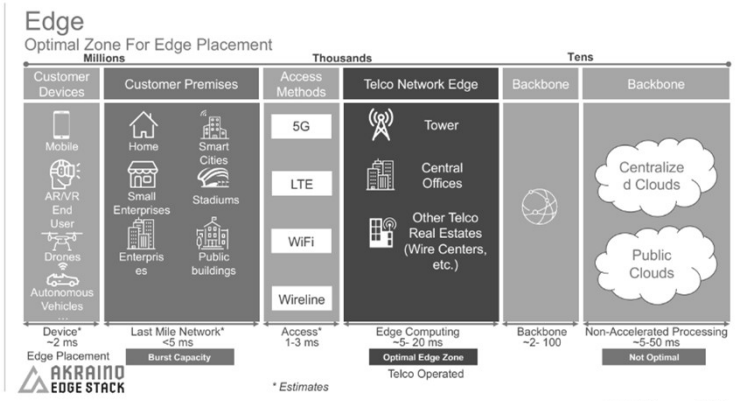


그림 6. SONiC(Software for Open Networking in the Cloud) 아키텍처



Backup

❖ **Network Edge in service providers is where the service provider logically connects to their customers. It is controlled by the service provider. Akraino also includes the CPE device, which is generally located on customer premises.**

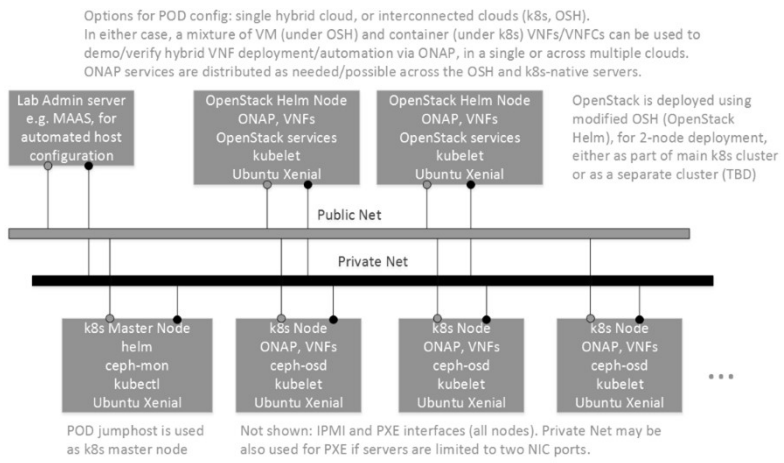


Backup

And, the Cloud Native Computing Foundation (CNCF), which houses the Kubernetes project, late last year partnered with the Eclipse Foundation to launch the Kubernetes IoT Edge Working Group. Mike Milinkovich, executive director of the Eclipse Foundation, boldly said the working group is looking to see how far it can push the centralized Kubernetes platform out into the distributed edge and IoT ecosystem.

"When you deal with Kubernetes in the core of a network, dealing with quarterly updates is one thing," said Brian Gracely, director of product strategy at Red Hat. "But when you have 2,000 branch offices, that is a different model. Having to touch thousands of those environments is a big challenge."

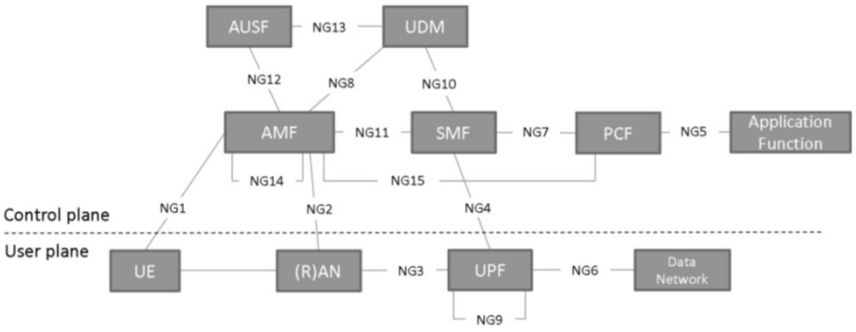
https://www.sdxcentral.com/articles/news/kubernetes-push-to-the-edge-shows-innovation-challenges/2019/03/



https://wiki.opnfv.org/display/AUTO/Auto+Use+Cases

Backup

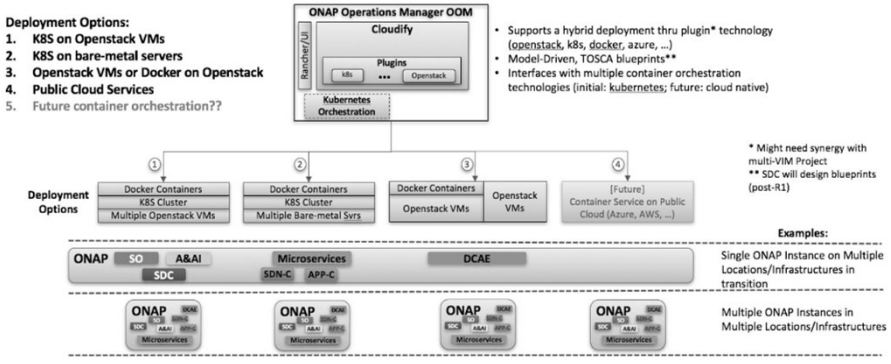
- ❖ 3GPP proposed architecture and reference points for 5G networks
- ❖ User Plane – Control Plane Split : SDN Architecture



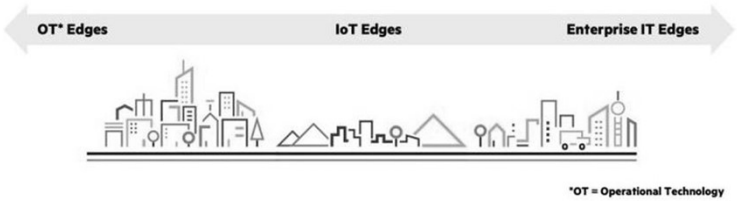
- User Equipment (UE);
- (Radio) Access Network (RAN);
- User Plane Function (UPF);
- Access and Mobility Function (AMF);
- Session Management Function (SMF);
- Policy Control Function (PCF);
- Authentication Server Function (AUSF);
- User Data Management (UDM);

Backup

- ❖ ONAP 요약



james@jslab.kr



(Image: Hewlett Packard Enterprise)