삼성 오픈소스 컨퍼런스

NHN은 어떻게 OpenStack으로 퍼블릭 클라우드를 운영할까?

구름 아래의 모든 일들

NHN 클라우드시스템실 조성수
2019.10.17

서비스의 기본 인프라, 클라우드

서비스 개발에 클라우드는 기본 인프라

언제든지 가상화된 인프라 자원을 클라우드로 부터 공급받는다

- 인스턴스
- 스토리지
- 공인 IP
- 로드 밸런서
- 사설 네트워크

클라우드 그 아래의 모습

클라우드는 무엇으로 만들어질까?

그곳에도 오픈소스가 사용될까?

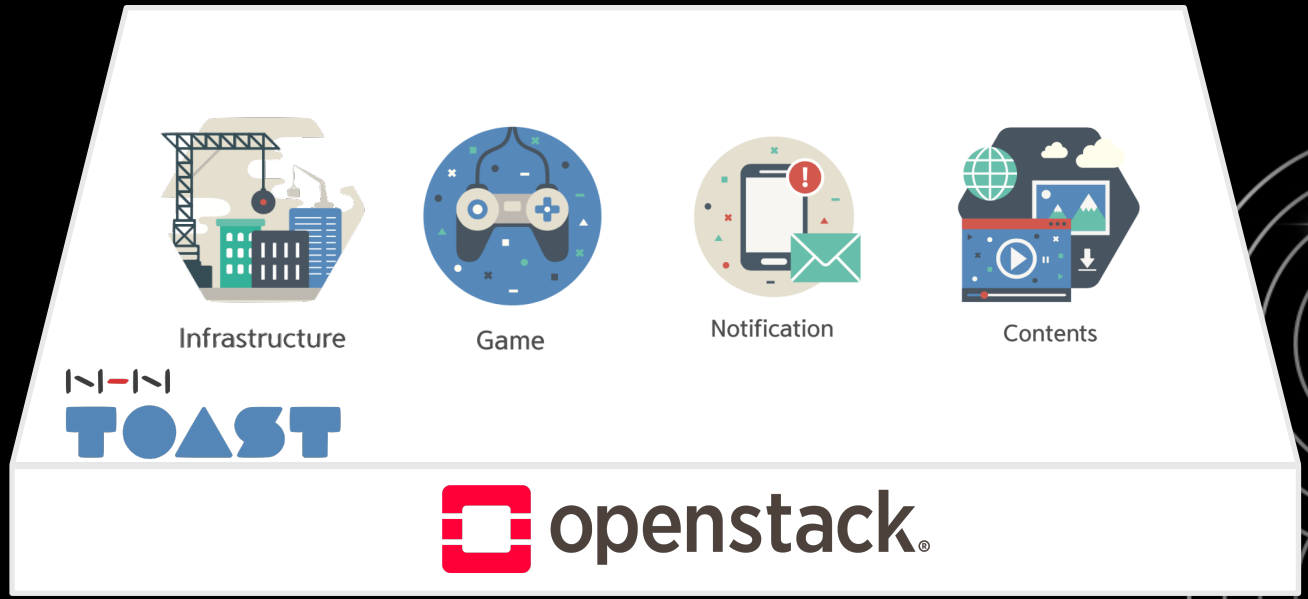
대부분의 클라우드 서비스의 내부 동작은 미지의 영역

오픈소스로 클라우드 서비스 만들기

오픈스택에 대한 이야기가 아닌, 오픈스택으로 어떻게

클라우드 서비스를 만들 수 있는지에 대한 이야기

NHN이 하는 클라우드 서비스 - NHN TOAST



오픈스택으로 같이

토스트 클라우드를 만들어 보아요

SOSCON 2019

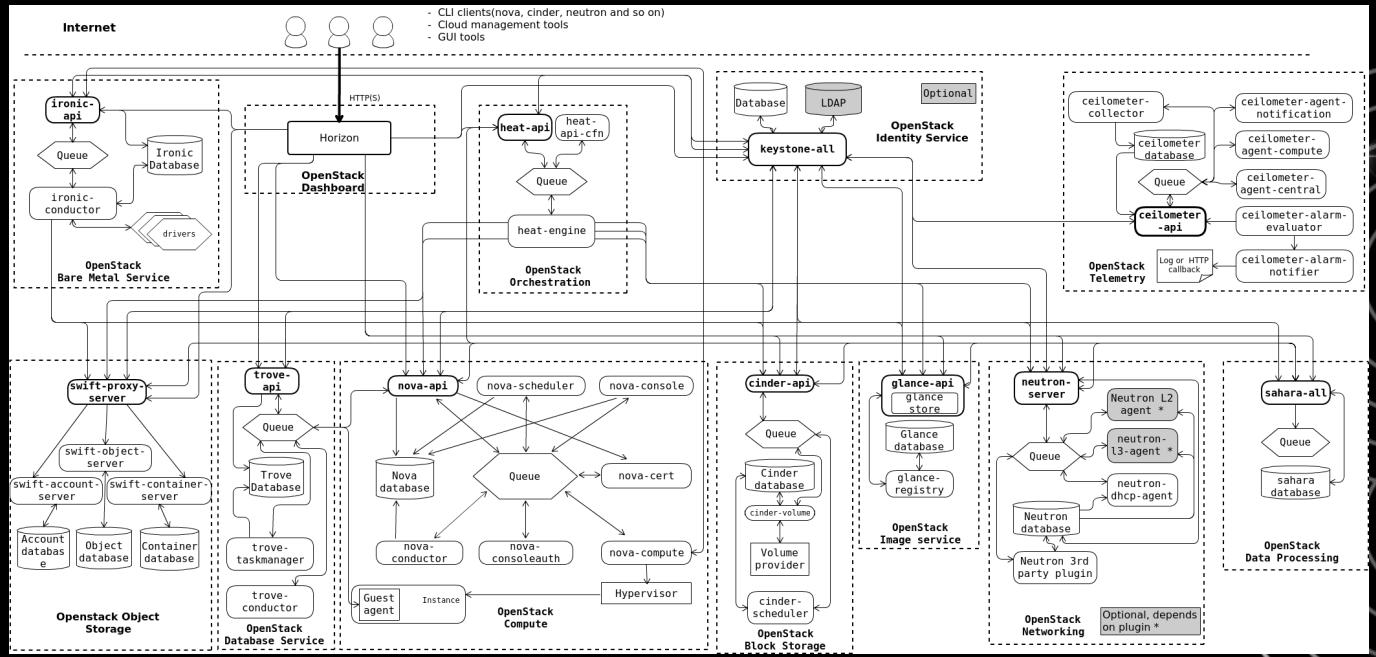
SAMSUNG OPEN SOURCE CONFERENCE 2019

클라우드 인프라 서비스 - IaaS

IaaS (Infrastructure as a Service) 를 이루는 5가지 필수 요소

클라우드 인프라 서비스 - IaaS

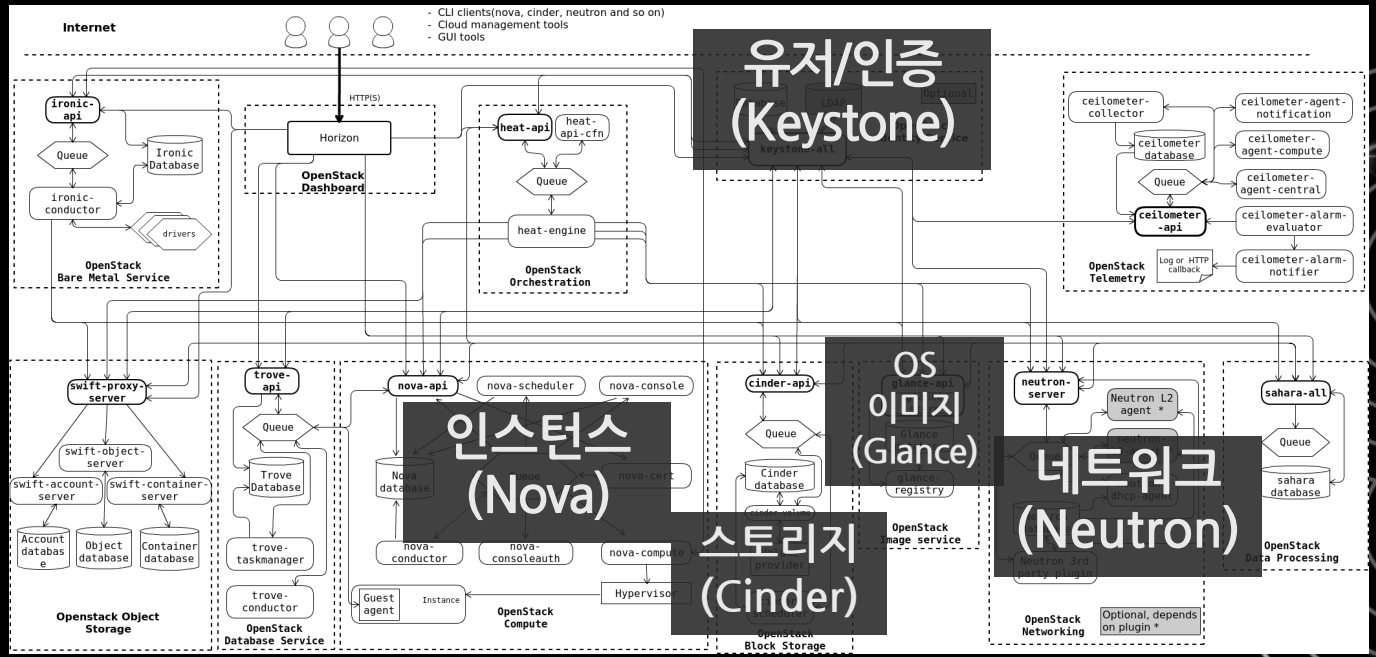
오픈스택은 여러 프로젝트들의 집합체 (역할 별로 구분된 컴포넌트)



출처: https://docs.openstack.org/install-guide/get-started-logical-architecture.html

클라우드 인프라 서비스 - IaaS

5가지 핵심 컴포넌트 (Nova, Cinder, Glance, Neutron, Keystone)



출처: https://docs.openstack.org/install-guide/get-started-logical-architecture.html

퍼블릭 클라우드를 위해 필요한 추가 요소

핵심 컴포넌트만으로도 인스턴스를 생성하고 사용할 수 있는 환경을 만들 수 있다

하지만, 이것을 사용자에게 서비스해야 한다면 추가적인 요소가 더 필요하다

예: 회원 체계, 웹 콘솔 , API, 운영자 도구, 운영 정책 등..

오픈스택기반 클라우드 서비스를 위한 고민

고민1. 오픈스택을 구동할 환경을 어떻게 만들 것인가?

컴포넌트들의 고 가용성(HA), 손쉬운 확장(Scalability)
가상 네트워크의 구성 그리고 인터넷으로 서비스 방법
글로벌 리전(Region) 배포

고민2. 오픈스택 서비스를 어떻게 사용자에게 제공할 것인가?

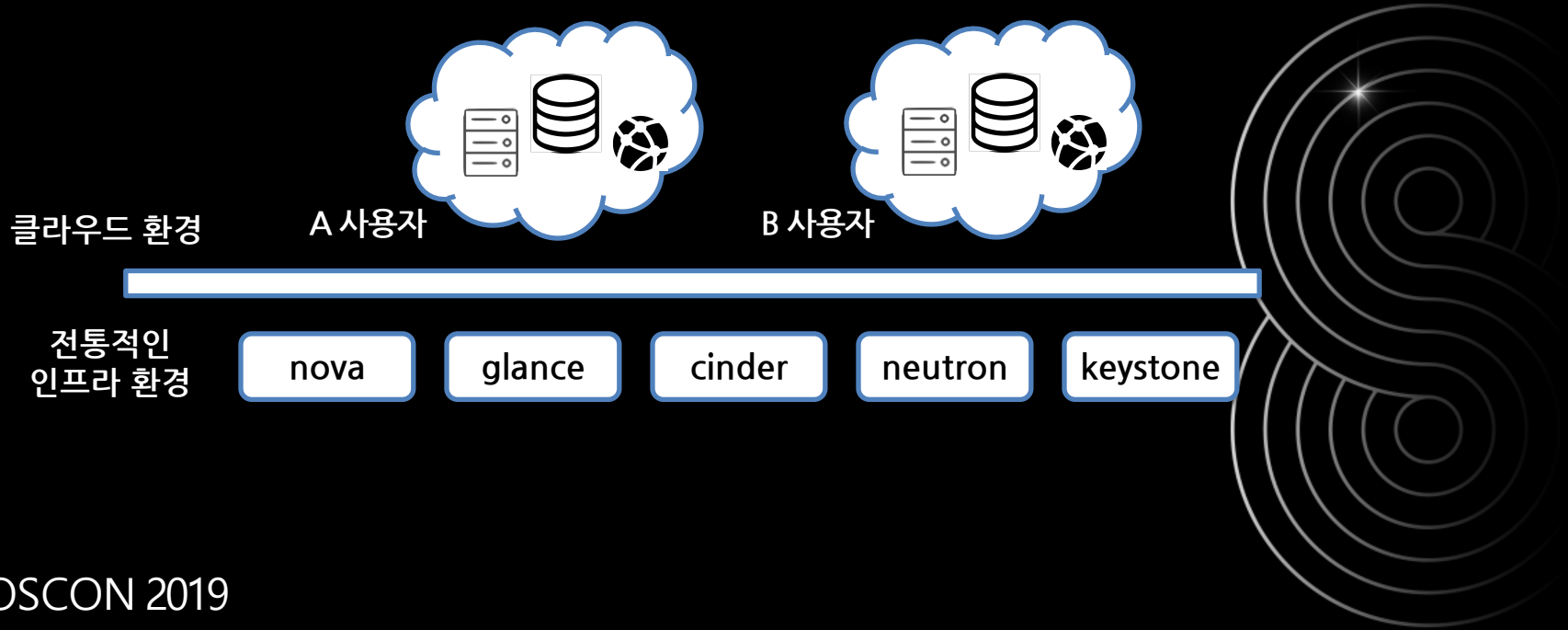
토스트 회원, 사용자 웹 콘솔, 오픈스택에 새로운 기능 개발

오픈스택을 구동할 환경을

어떻게 만들 것인가?

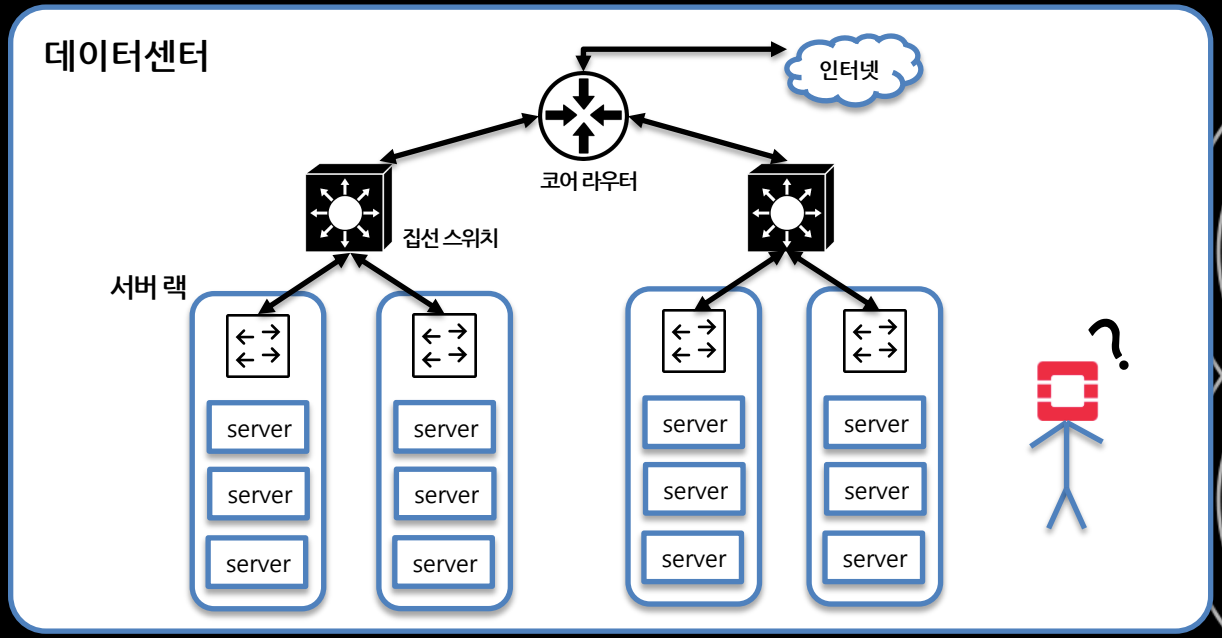
오픈스택 핵심 컴포넌트 배치

클라우드 환경을 만드는 핵심 컴포넌트도 어디에선가 구동해야한다.



오픈스택 핵심 컴포넌트 배치 - 전통적인 인프라

전통적인 인프라 = 데이터센터 / 물리적인 서버 + 네트워크



오픈스택 컴포넌트의 주요 특성

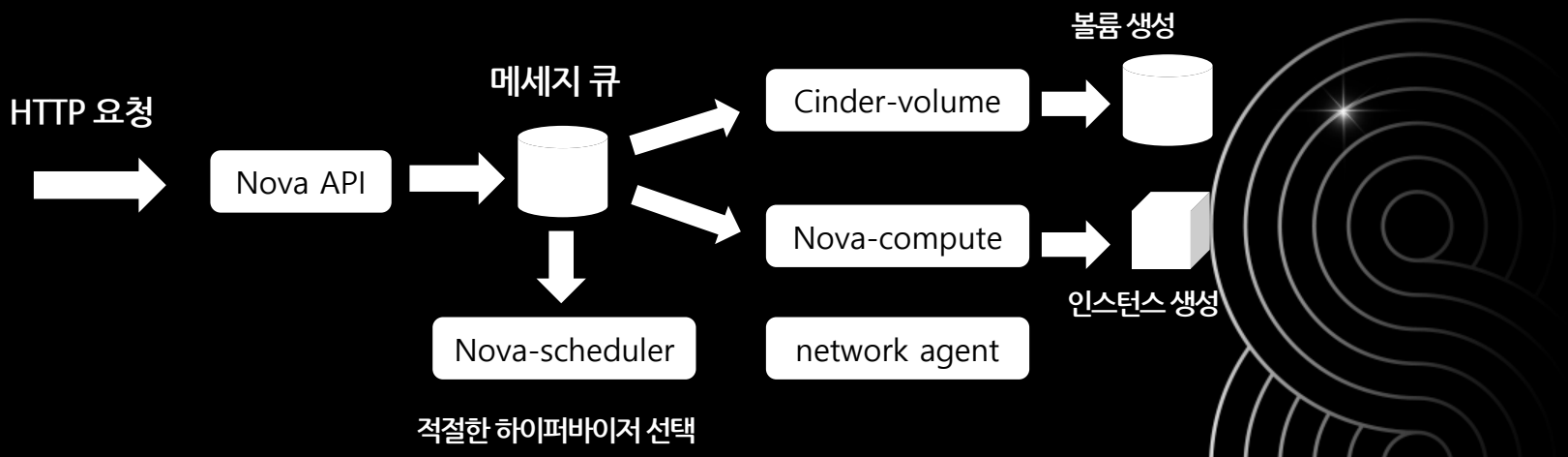
핵심 컴포넌트는 크게 2가지 특성으로 분류 가능

1. API 서비스 (nova-api, cinder-api 등..)

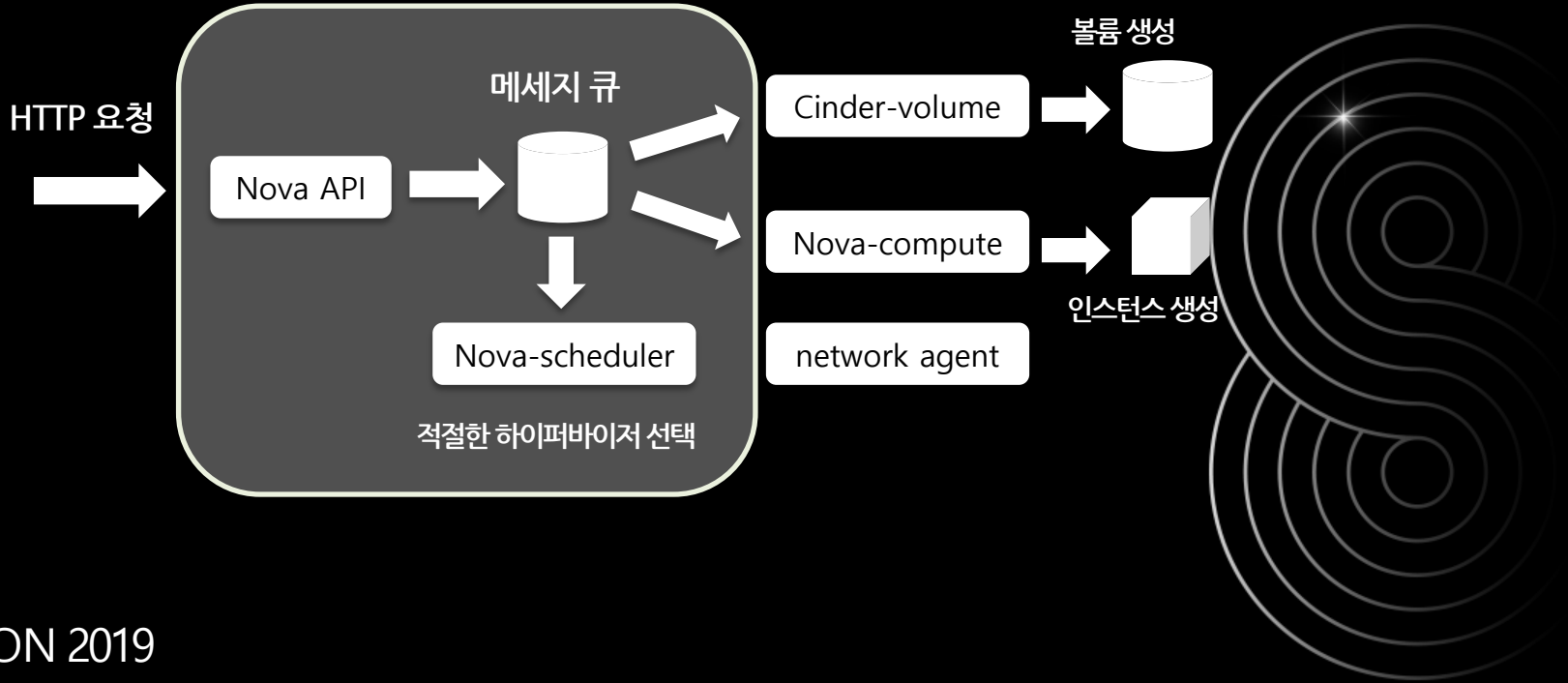
2. 에이전트 (nova-compute / openvswitch-agent 등..)

오픈스택 컴포넌트의 주요 특성

예: 인스턴스 생성 요청



특성에 따른 배치 - API 서비스

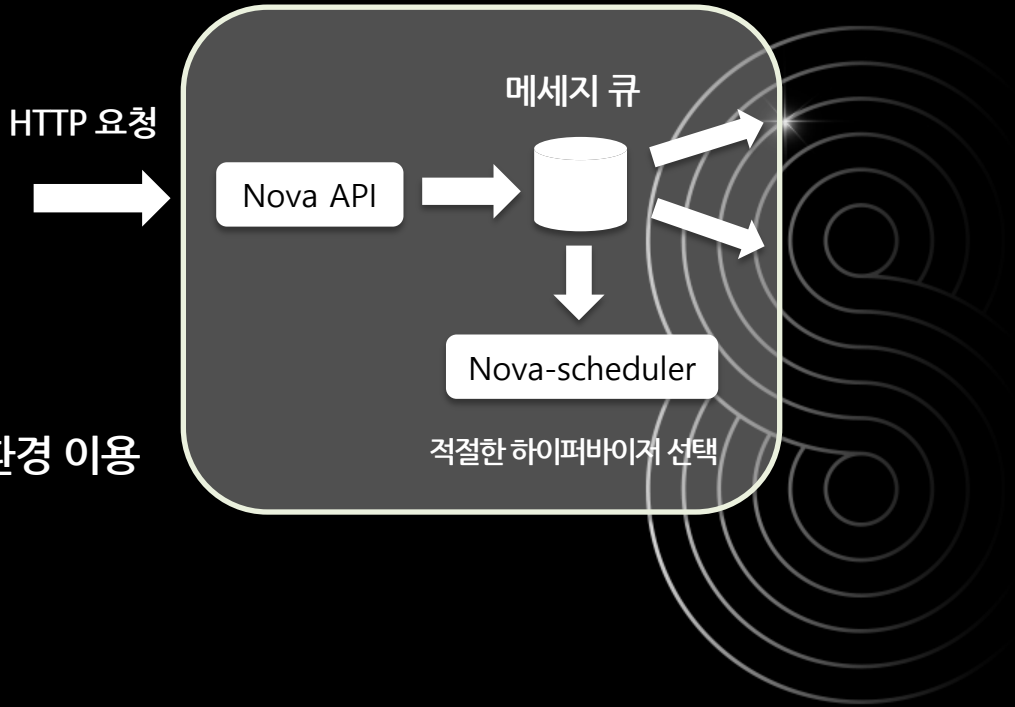


특성에 따른 배치 - API 서비스

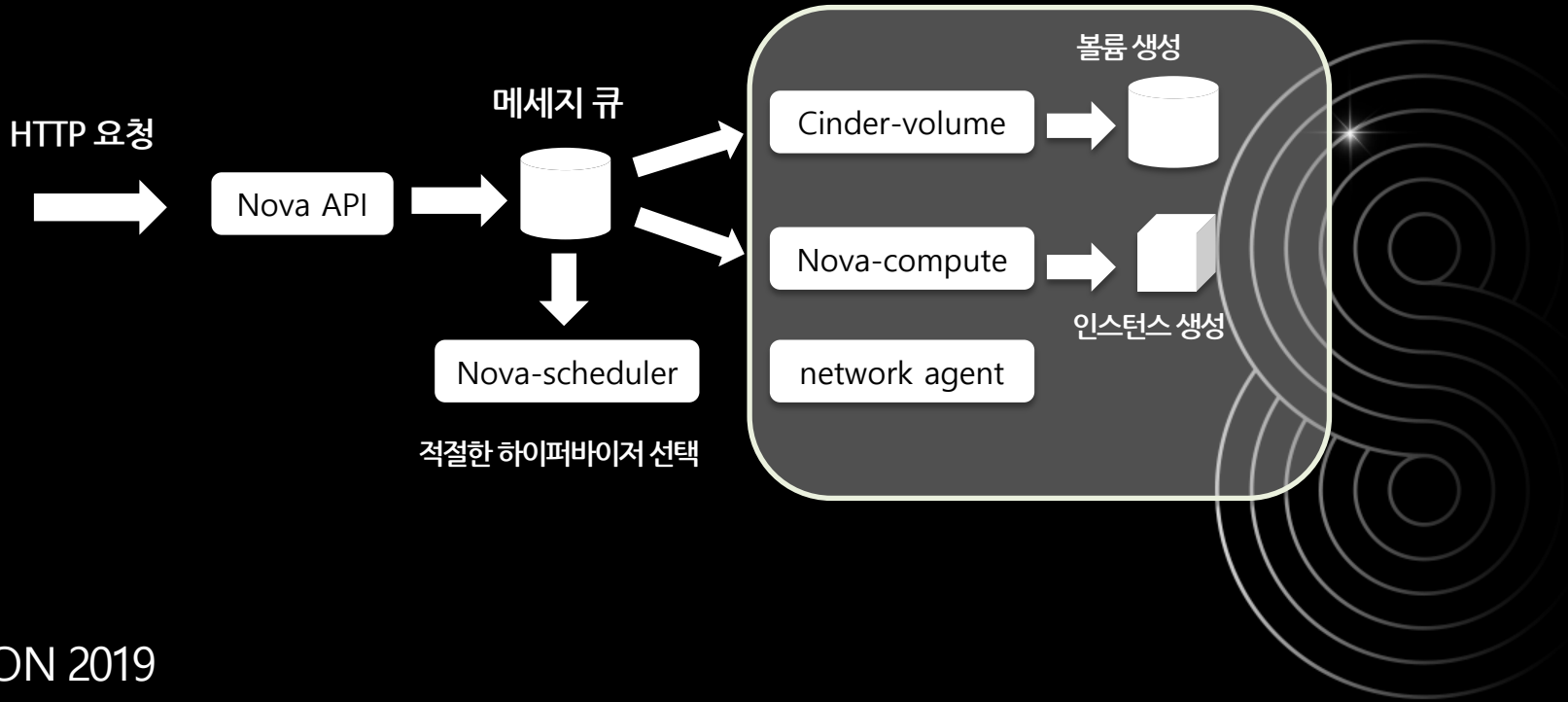
특징
- API 서비스
- 확장이 용이해야한다
- 여러개의 네트워크가 불필요하다

배치
- 가상 서버에 컴포넌트 배치
- 1개 서버에 여러개의 컴포넌트 배치

  (예: nova-api/cinder-api 를 같은 서버에 배치)
- 기존 사내 시스템에서 사용하는 가상 환경 이용



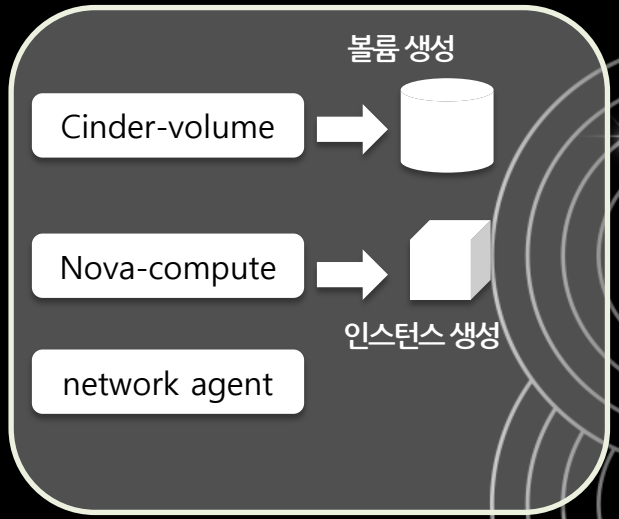
특성에 따른 배치 - 에이전트



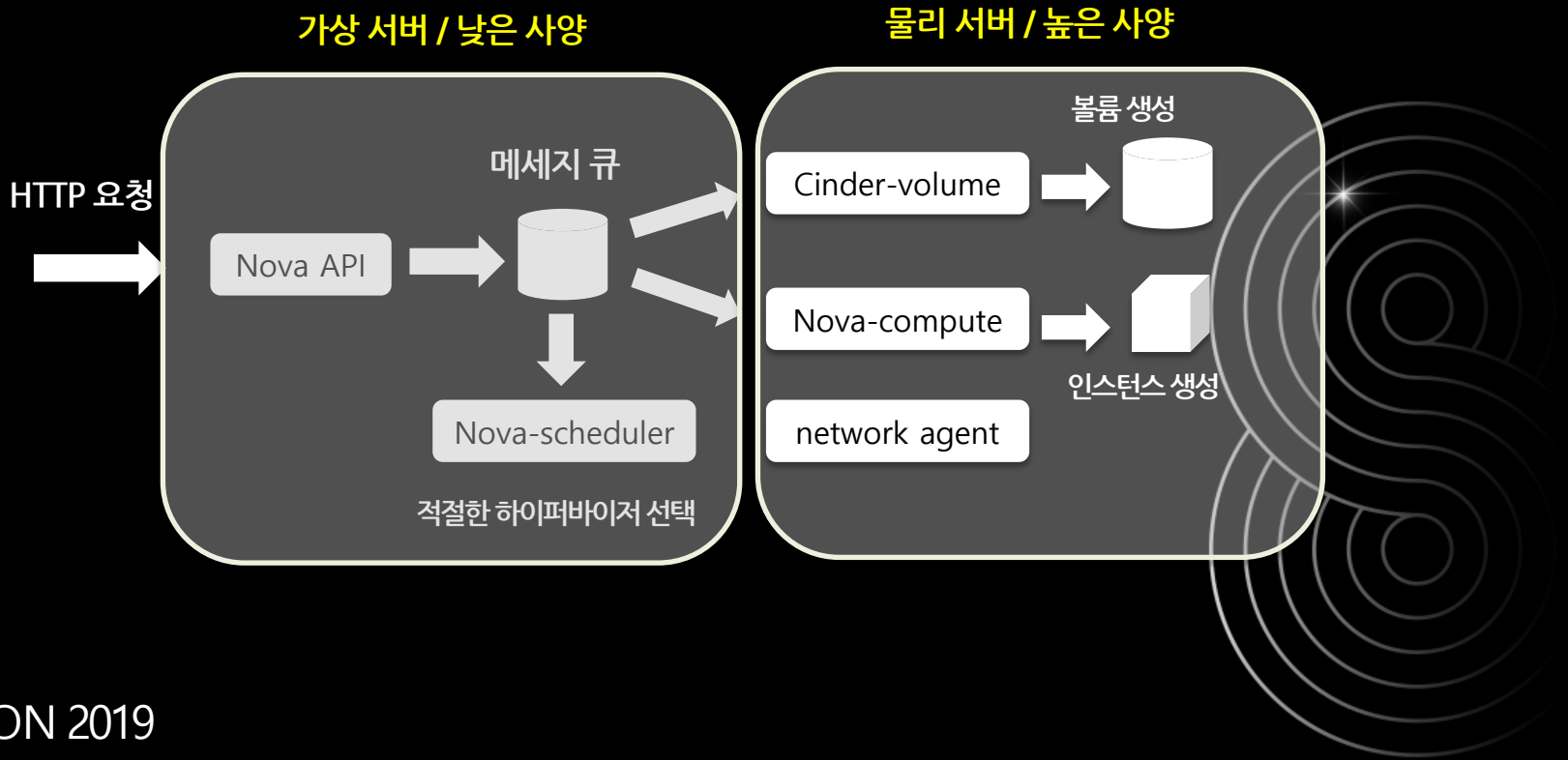
특성에 따른 배치 - 에이전트

특징
- 에이전트가 구동
- 성능이 중요하거나, 높은 사양이 필요
- 여러 개의 네트워크에 연결 되어야한다

배치
- 물리 서버에 컴포넌트 설치
- API 서비스 서버가 있는 랙과 다른 랙에 배치
- 실제 인스턴스가 구동되는 컴퓨트 노드는 신중히 배치



특성에 따른 배치 - 정리

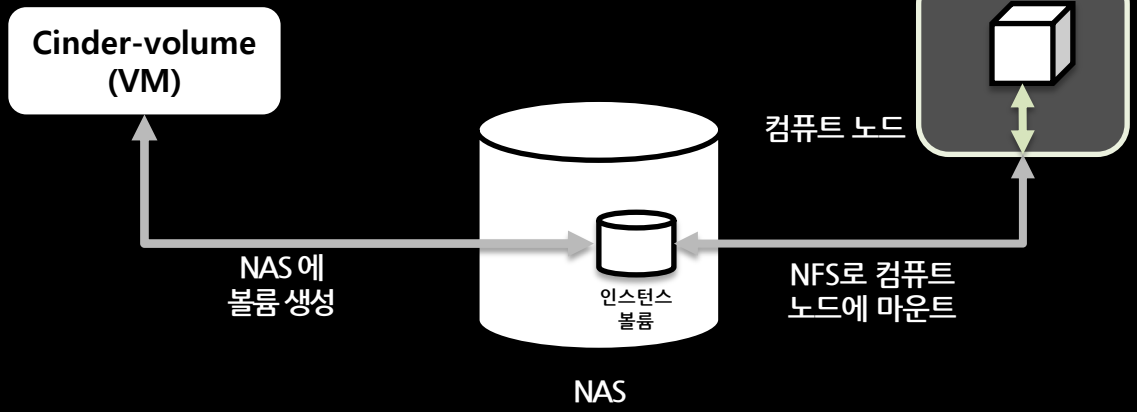

가상 서버 / 낮은 사양
물리 서버 / 높은 사양
HTTP 요청
메세지 큐
Nova API
Nova-scheduler
적절한 하이퍼바이저 선택
볼륨 생성
Cinder-volume
Nova-compute
인스턴스 생성
network agent

추가 요소 - 블록 스토리지

블록 스토리지는 인스턴스의 OS가 구동될 디스크 / 추가 디스크를 제공한다
(이를 볼륨이라 부른다)

블록 스토리지를 관리하는 컴포넌트인 cinder의 backend storage가 필요하다

토스트 클라우드에서는 NAS를 backend storage로 사용



가상 네트워크 (테넌트 네트워크) 제공 방법

사용자마다 가상 네트워크를 분리(isolated)해서 제공해야한다

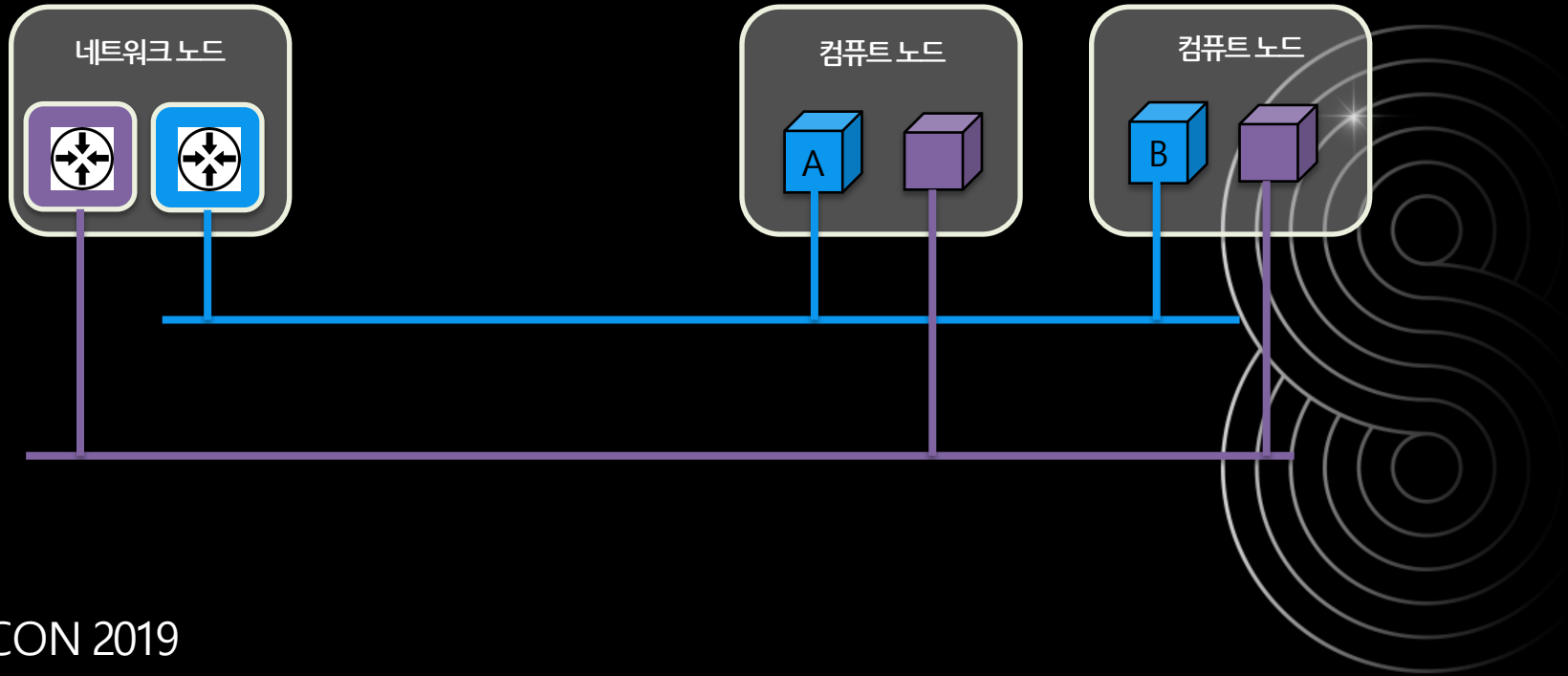
오픈스택에서 가상 네트워크를 구현하는 방법은 매우 다양하다

토스트 클라우드에서는 openvswitch를 활용한 방법을 사용하며,
DVR(Distributed Virtual Router)으로 가상 네트워크를 제공한다

가상 네트워크 트래픽을 처리하는 서버를 “네트워크 노드”라고 부른다

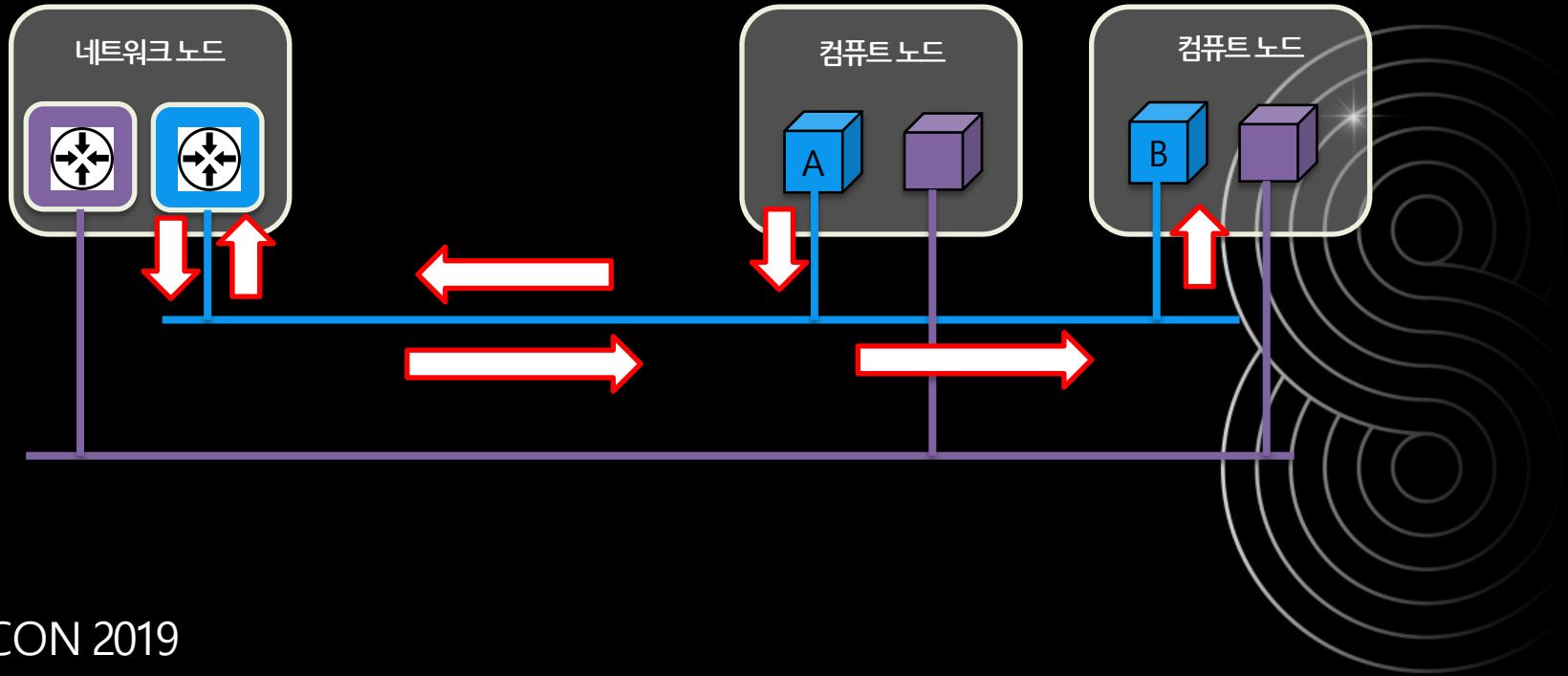
테넌트 네트워크의 트래픽 흐름 - 일반

네트워크 노드는 테넌트 네트워크를 제공해주고, 인스턴스들을 같은 네트워크로 묶는다



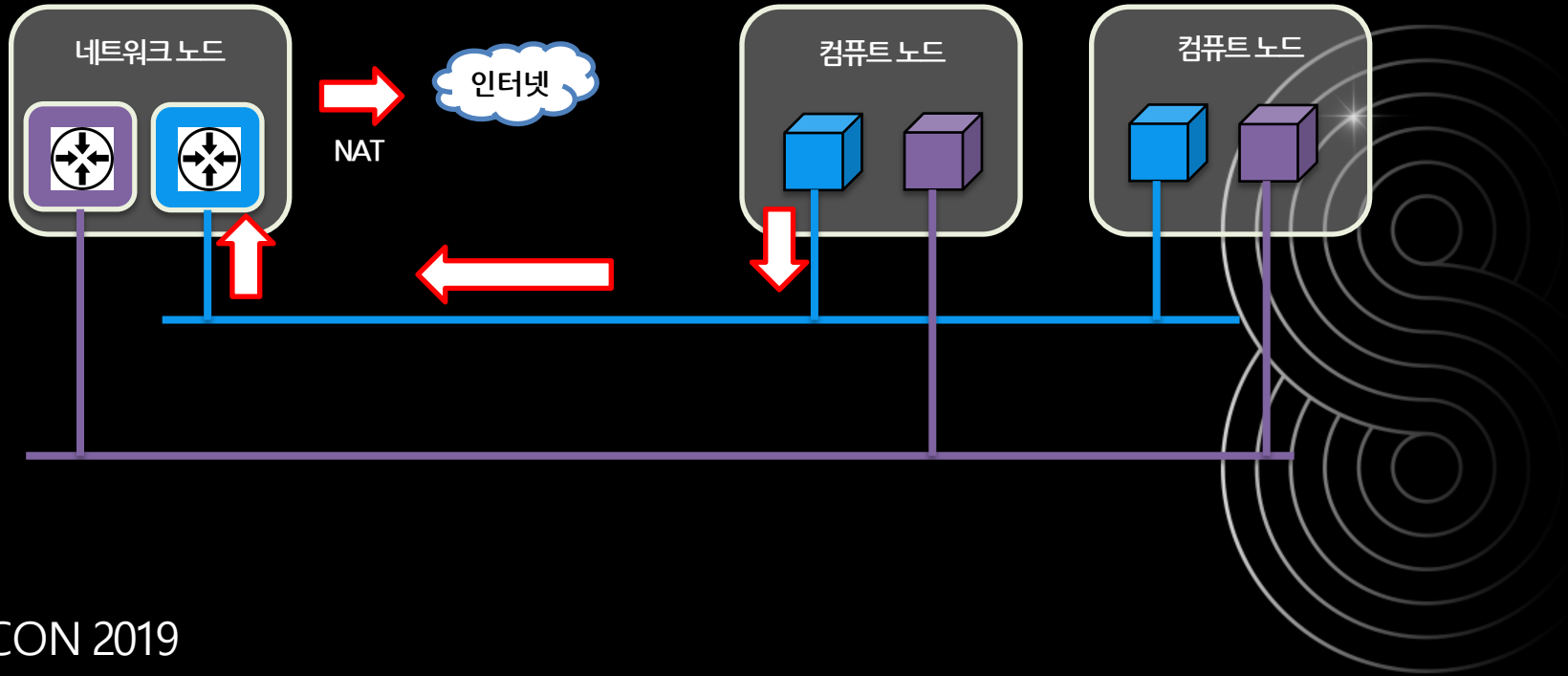
테넌트 네트워크의 트래픽 흐름(일반) - 인스턴스 간

서로 다른 컴퓨트 노드의 인스턴스 간 통신은 네트워크 노드를 거쳐야한다



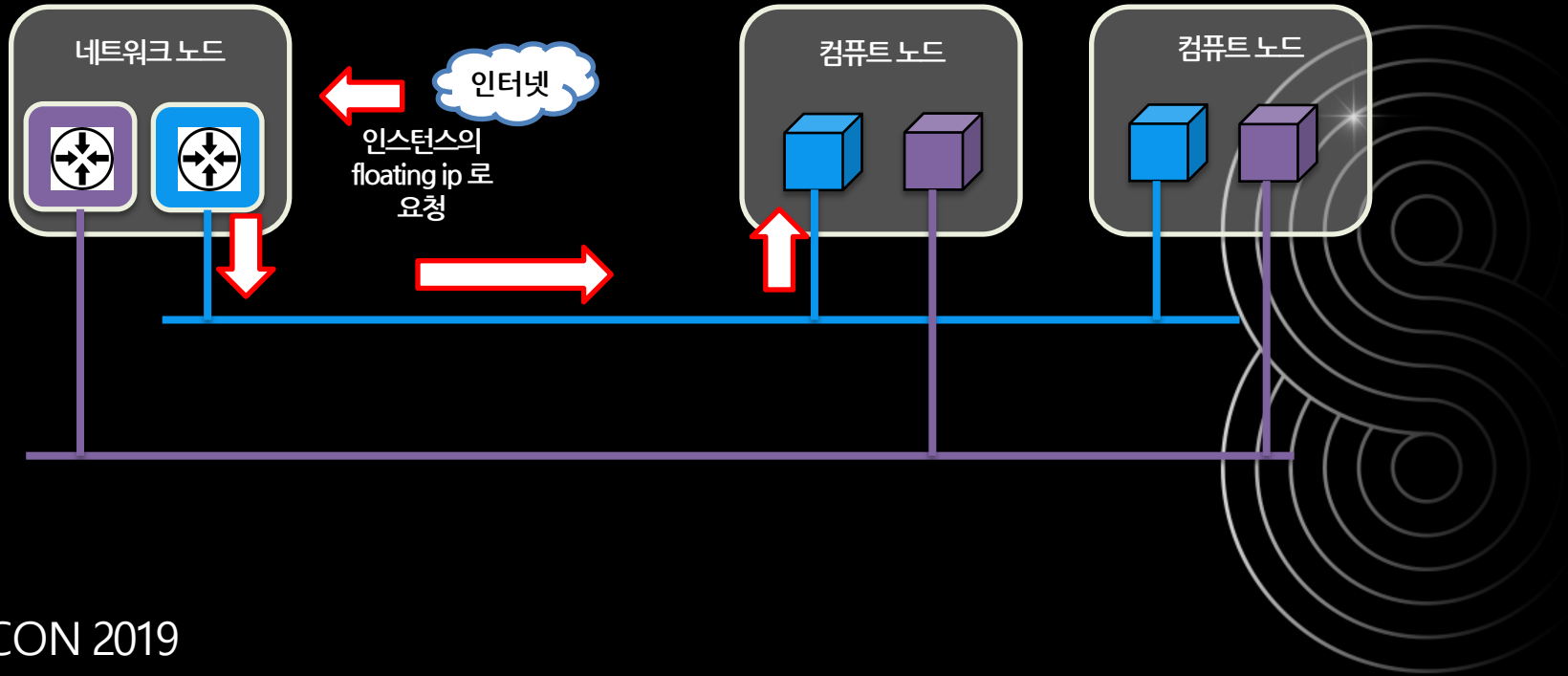
테넌트 네트워크의 트래픽 흐름(일반) - 인터넷

인스턴스가 인터넷과 통신하기 위해서도, 네트워크 노드를 거쳐야한다
(floating ip 가 없는 경우)



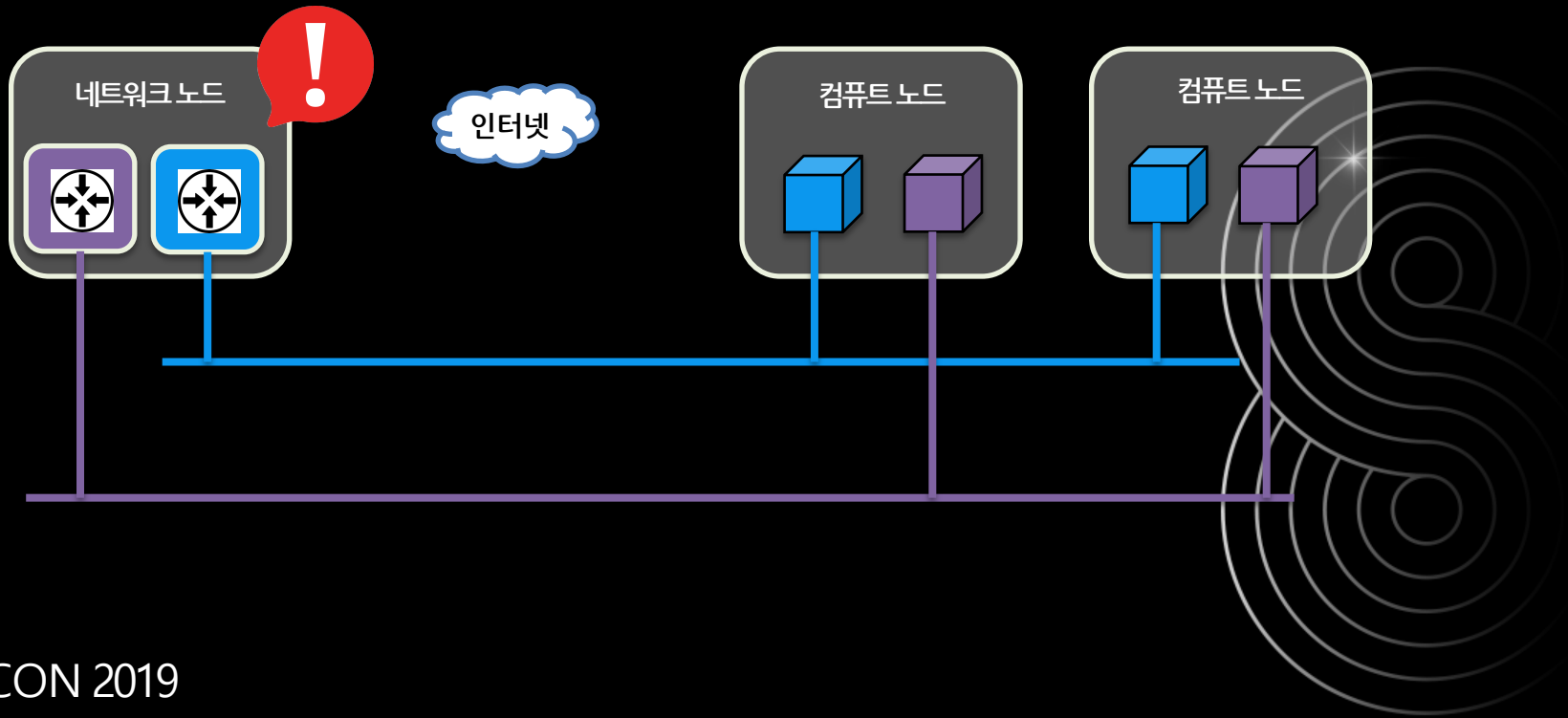
테넌트 네트워크의 트래픽 흐름(일반) - 인터넷

Floating ip 통신도 네트워크 노드를 거쳐야한다.



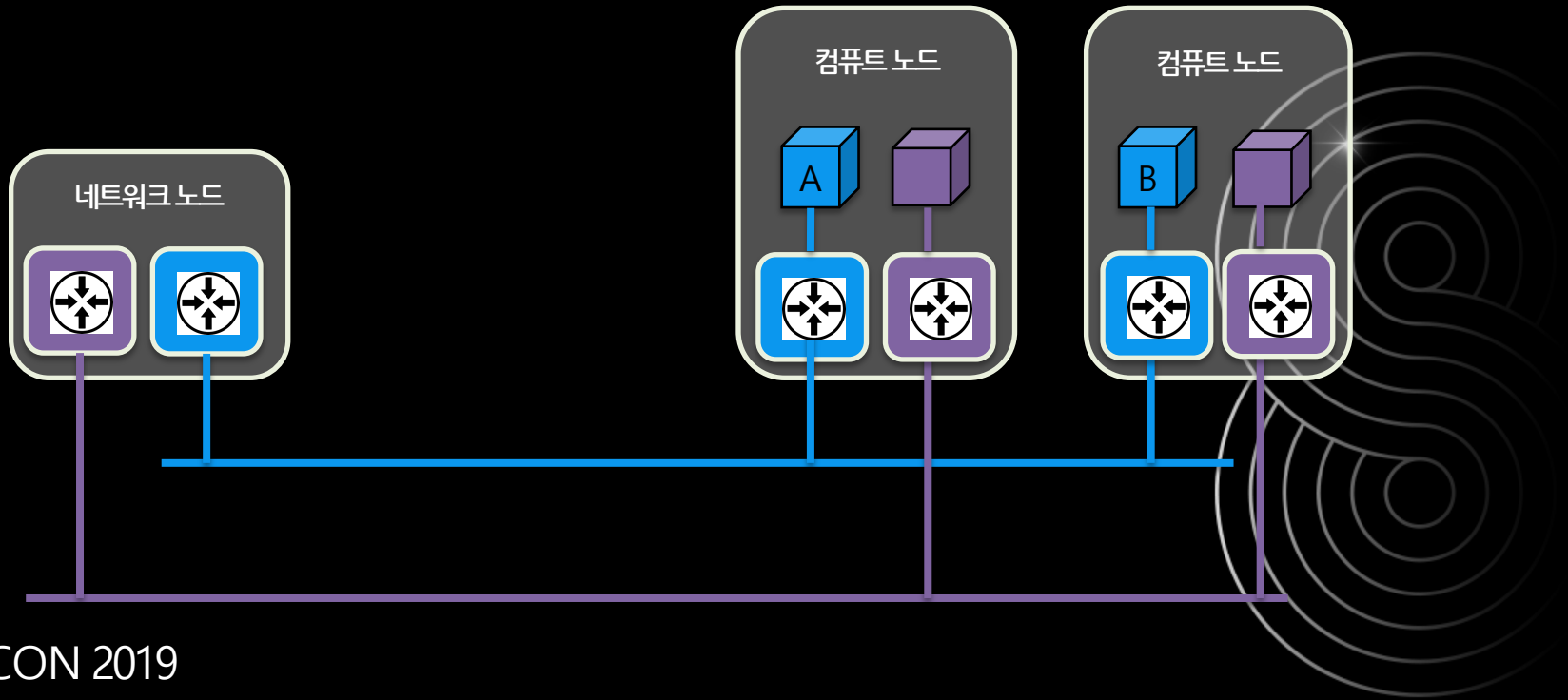
테넌트 네트워크의 트래픽 흐름 - 일반

네트워크 노드의 부하가 심해지고, 단일 장애 지점이 된다



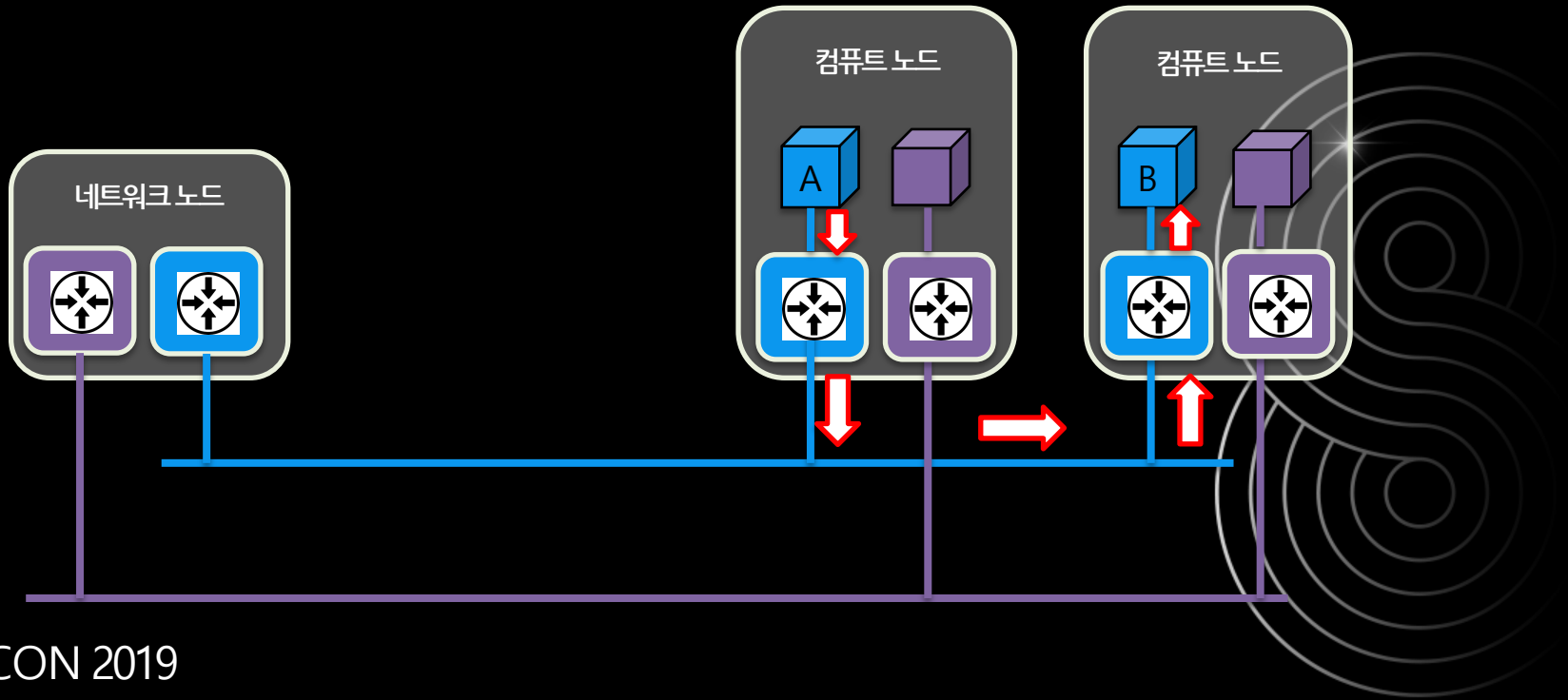
테넌트 네트워크의 트래픽 흐름 - DVR

테넌트 네트워크의 가상 라우터를 컴퓨트 노드마다 분산 배치



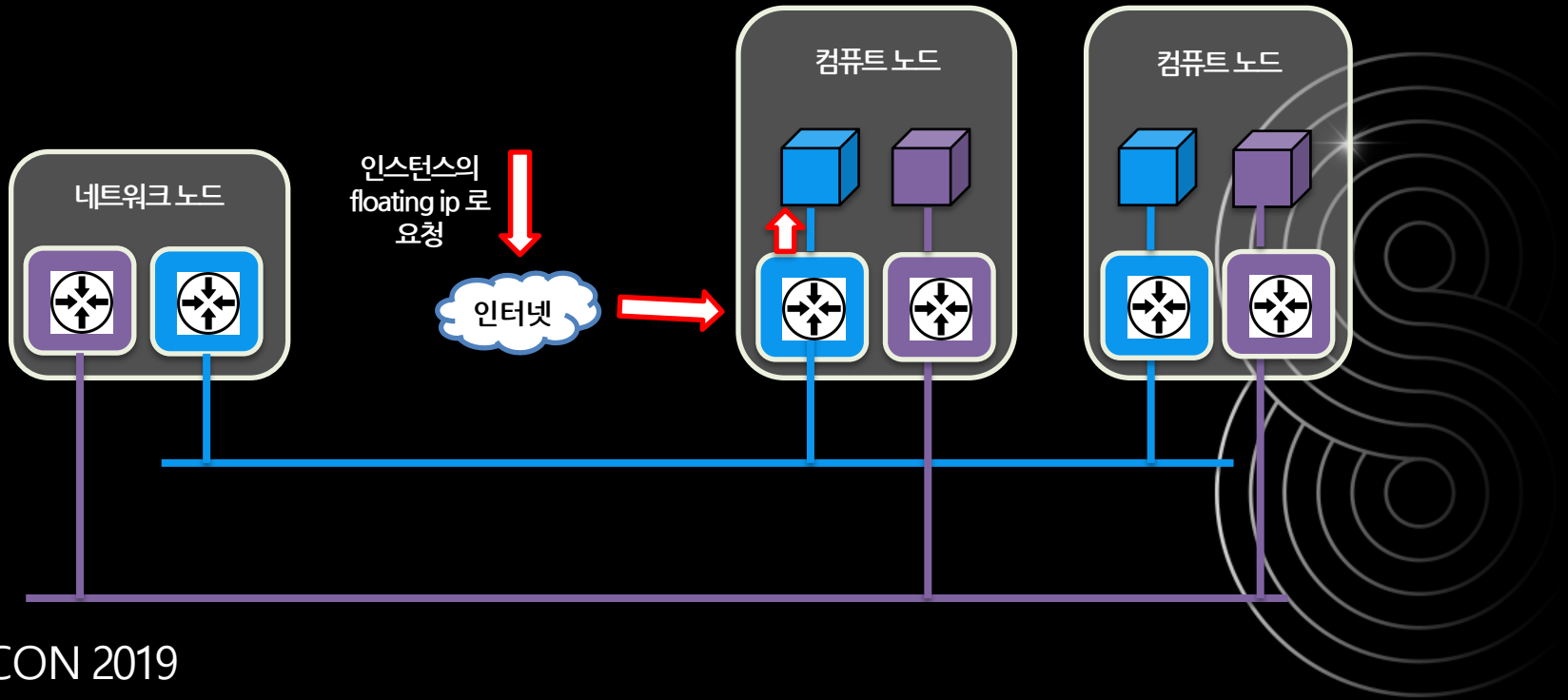
테넌트 네트워크의 트래픽 흐름 - DVR

인스턴스 간 통신을 네트워크 노드 없이 컴퓨트 노드끼리 통신할 수 있다



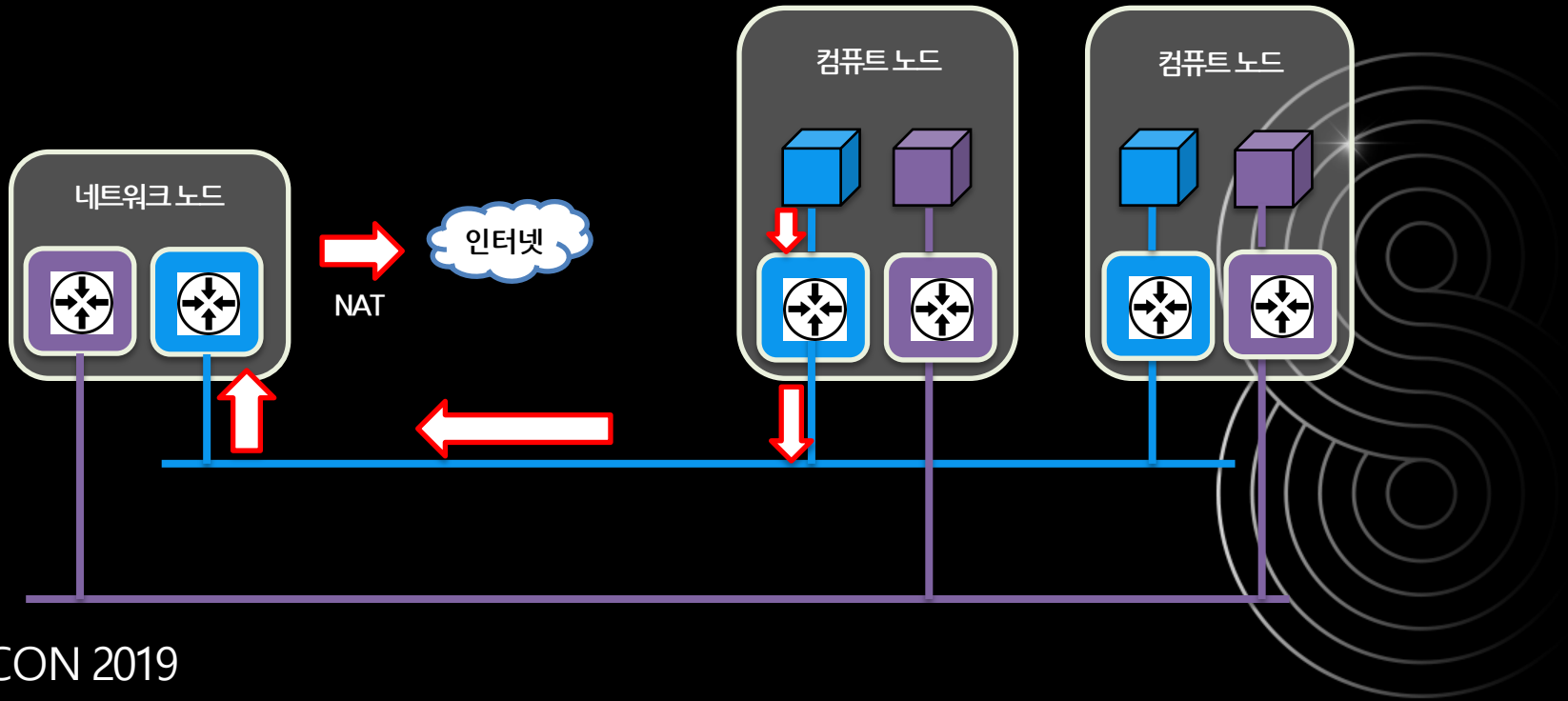
테넌트 네트워크의 트래픽 흐름 - DVR

Floating IP 트래픽도 바로 컴퓨트 노드로 들어올 수 있다

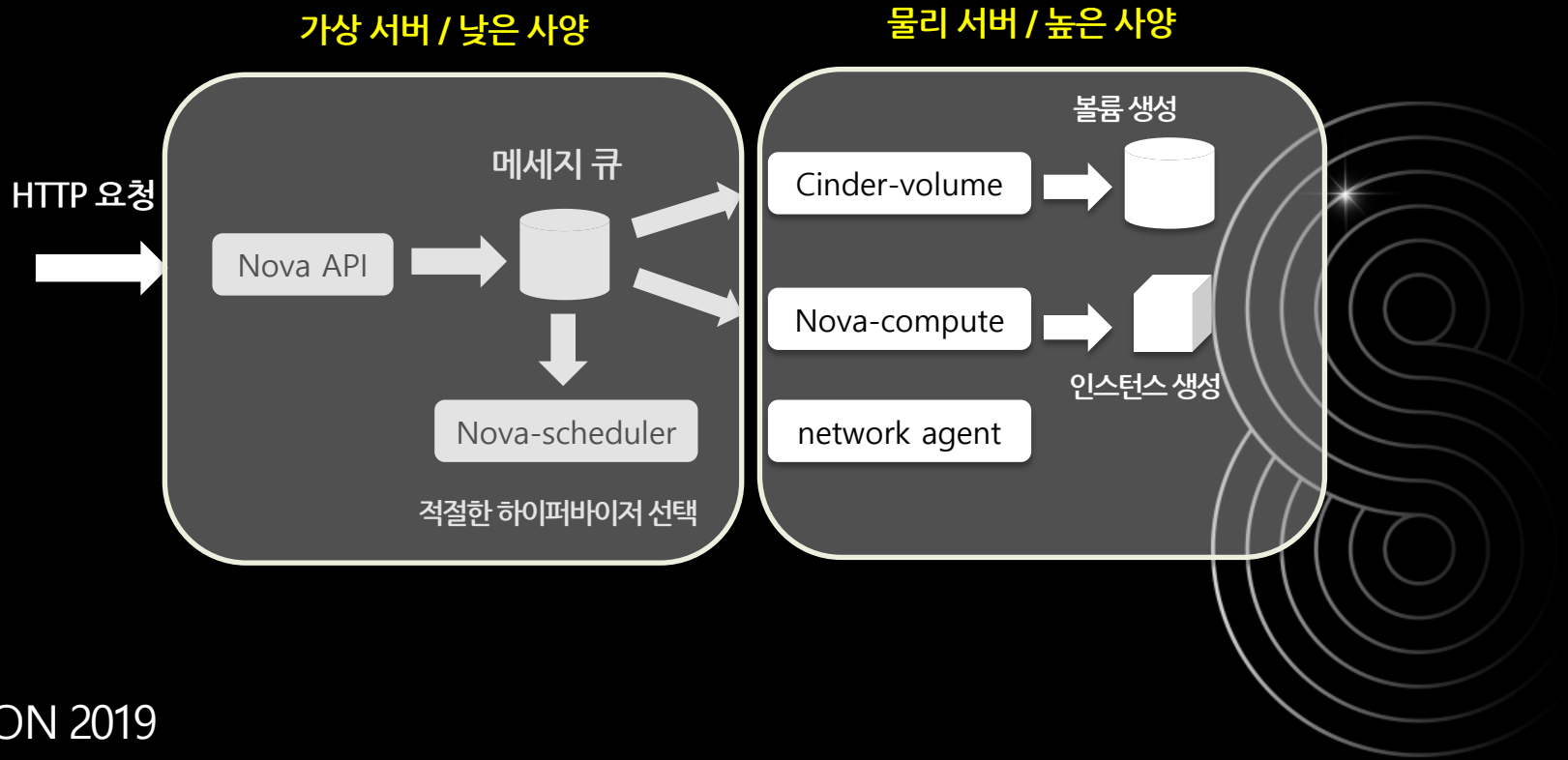


테넌트 네트워크의 트래픽 흐름 - DVR

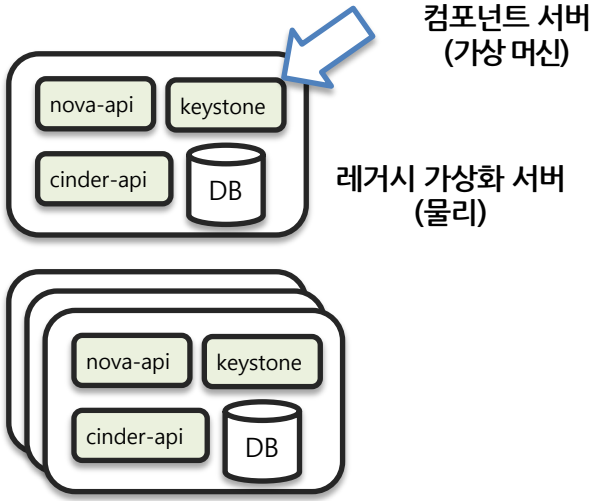
단, 인스턴스가 인터넷과 통신하고 싶다면 네트워크 노드를 통해야한다



다시 떠올리기 - 오픈스택 컴포넌트 배치



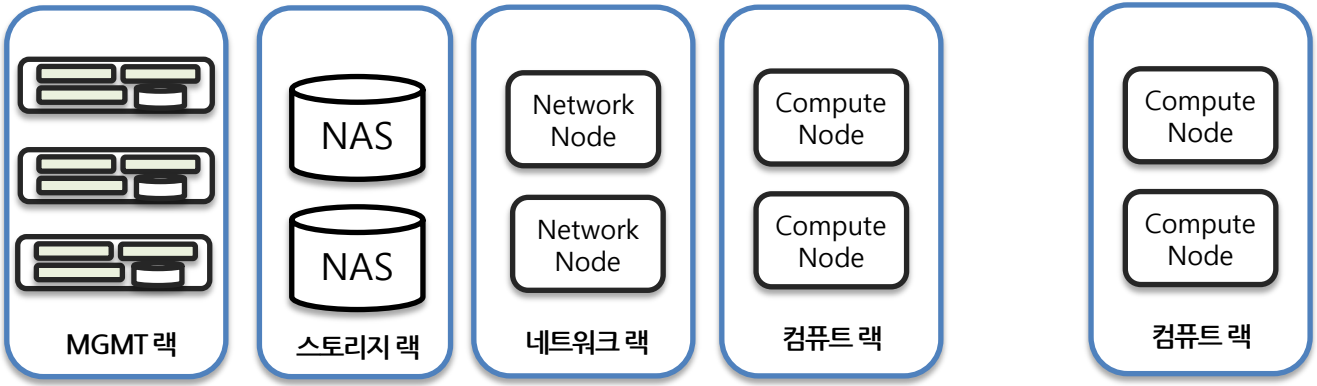
오픈스택 핵심 컴포넌트 물리 배치



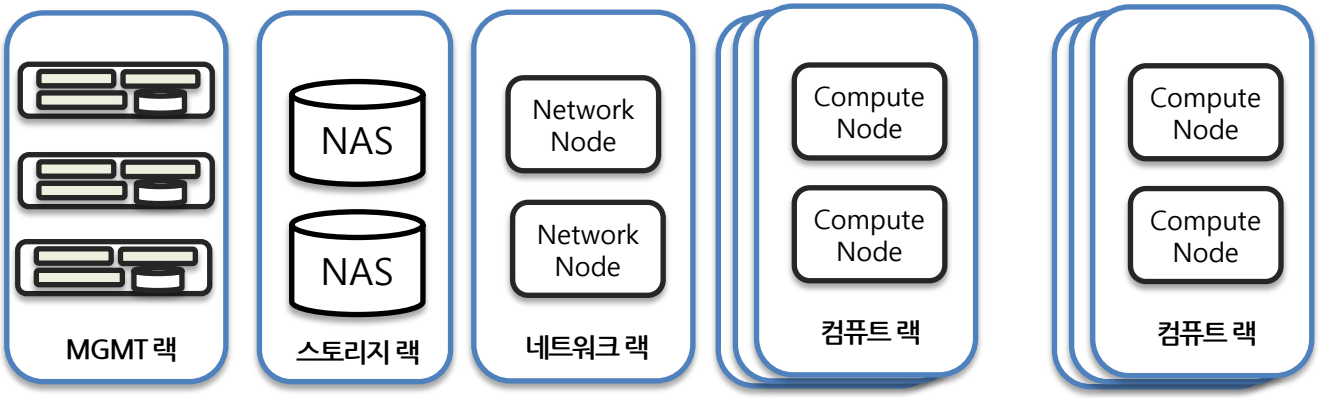
컴포넌트 서버
(가상 머신)

레거시 가상화 서버
(물리)

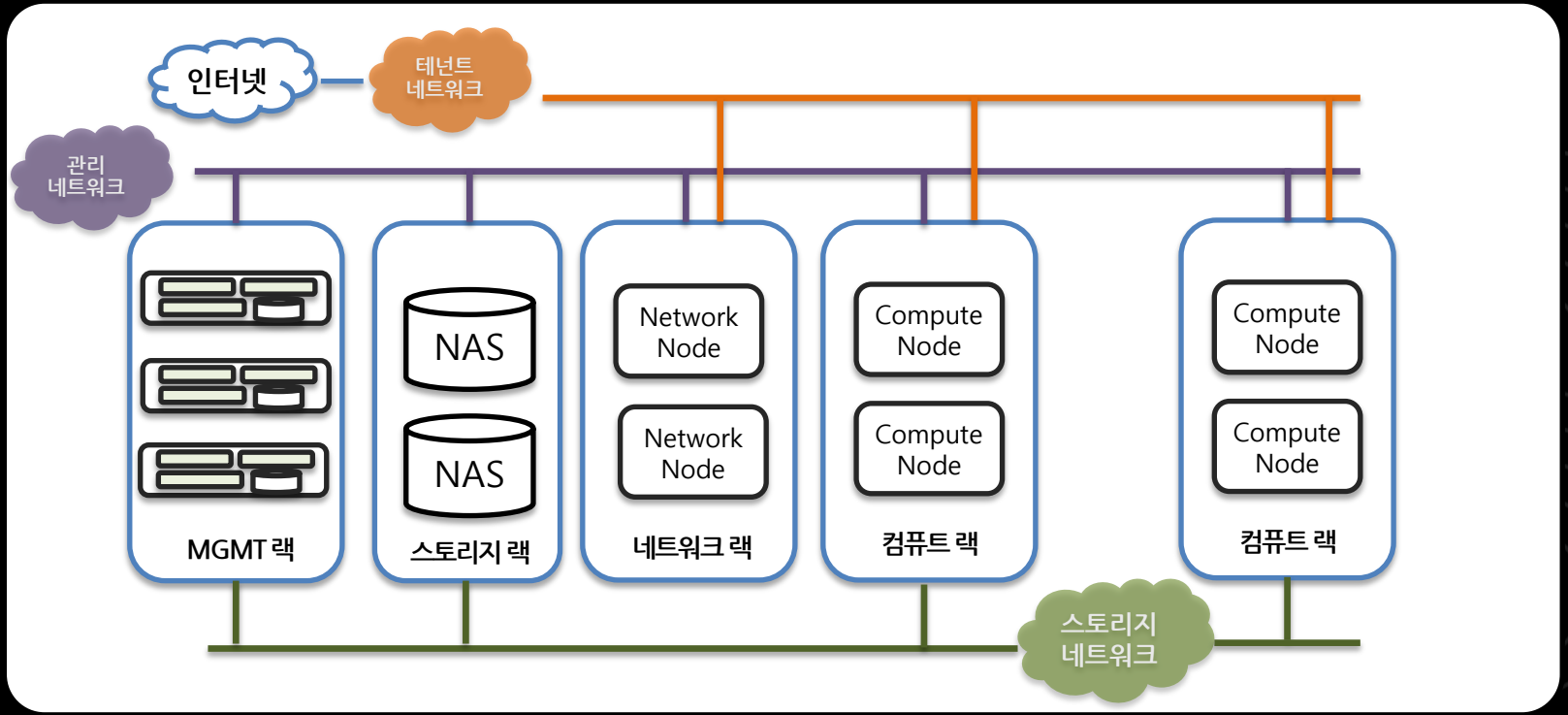
오픈스택 핵심 컴포넌트 물리 배치



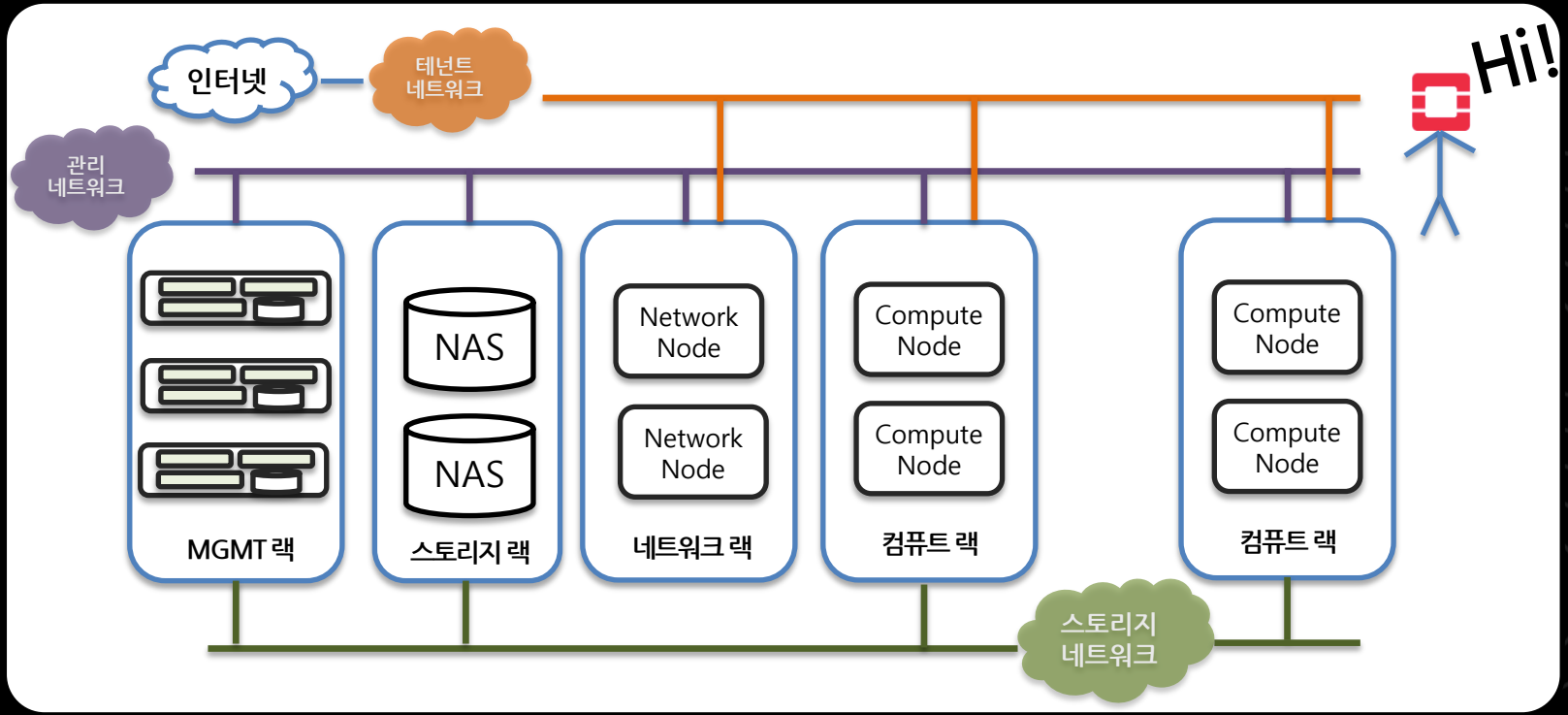
오픈스택 핵심 컴포넌트 물리 배치



오픈스택 핵심 컴포넌트 물리 배치 - 네트워크

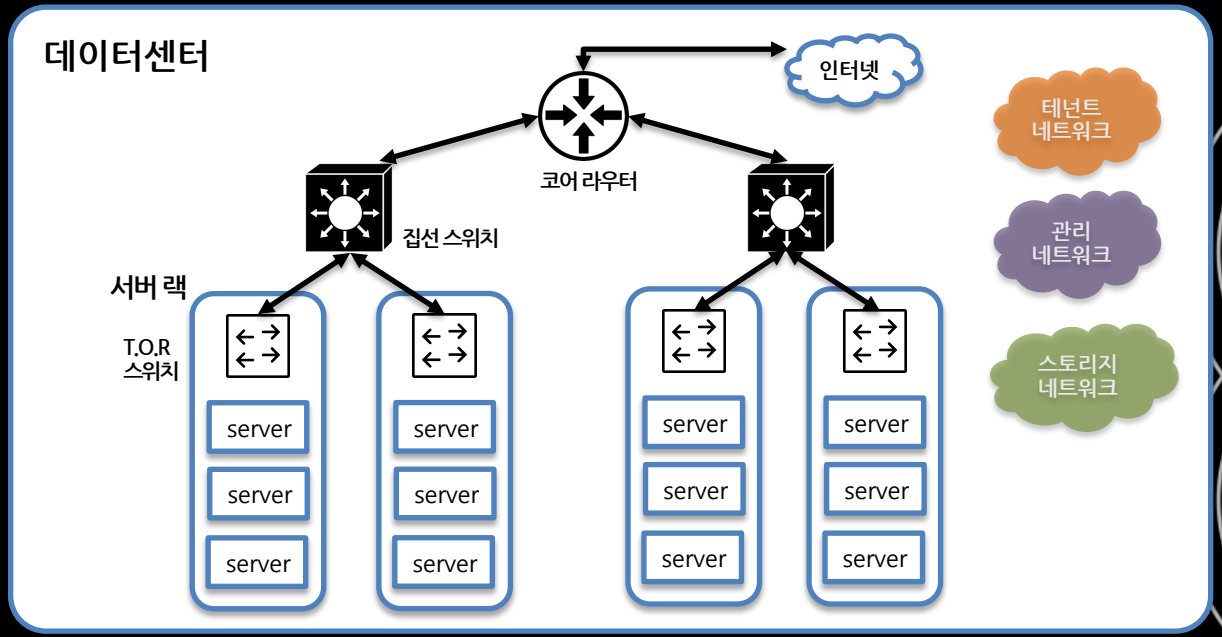


오픈스택 배치 완료!


인터넷
테넌트
네트워크
관리
네트워크
Hi!
MGMT 랙
NAS
NAS
스토리지 랙
Network
Node
Network
Node
네트워크 랙
Compute
Node
Compute
Node
컴퓨트 랙
Compute
Node
Compute
Node
컴퓨트 랙
스토리지
네트워크

오픈스택 배치 완료! - 데이터센터에 반영

물리 네트워크 구성 / 서버 구매 / 케이블링 등등 …



오픈스택기반 클라우드 서비스를 위한 고민

고민1. 오픈스택을 구동할 환경을 어떻게 만들 것인가?

컴포넌트들의 고 가용성(HA), 손쉬운 확장(Scalability)
가상 네트워크의 구성 그리고 인터넷으로 서비스 방법
글로벌 리전(Region) 배포

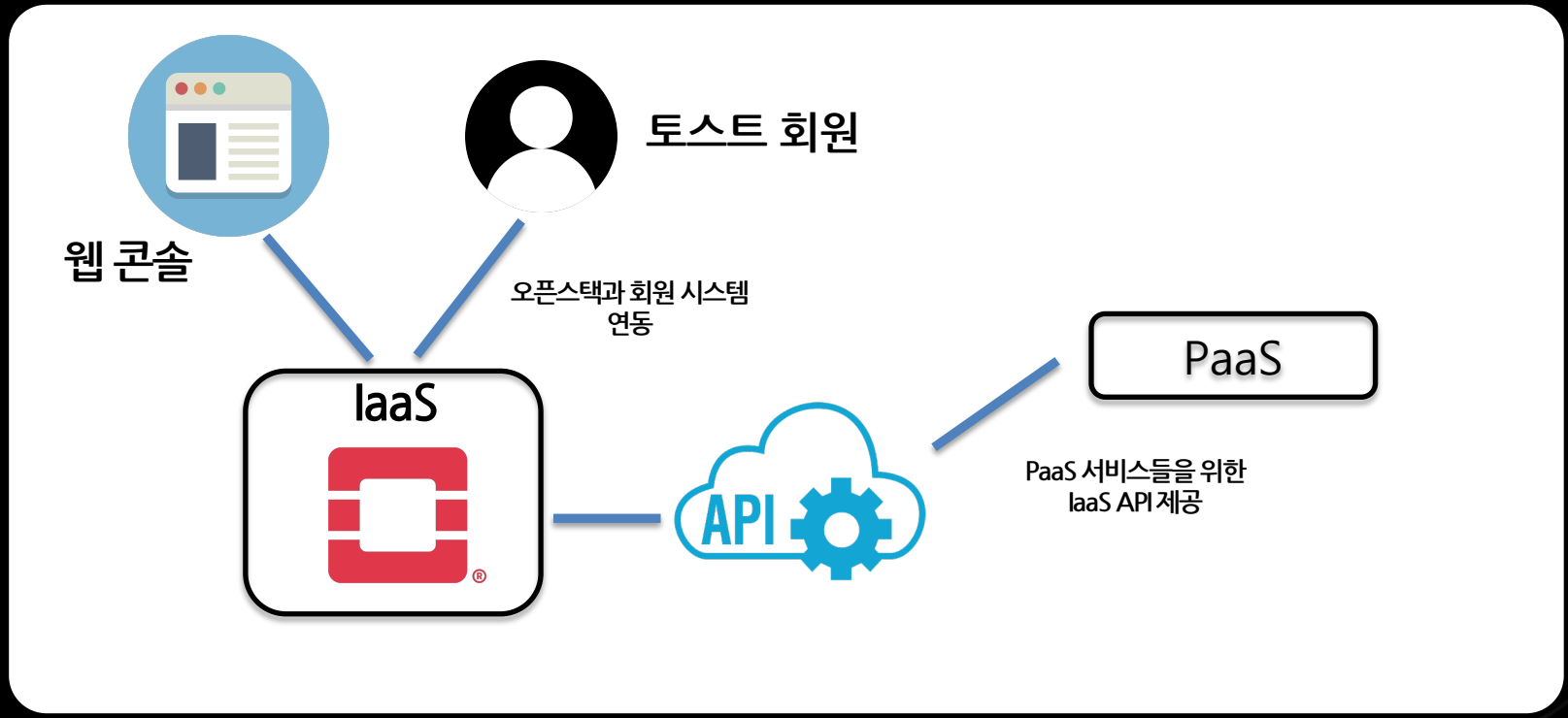
고민2. 오픈스택 서비스를 어떻게 사용자에게 제공할 것인가?

토스트 회원, 사용자 웹 콘솔, 오픈스택에 새로운 기능 개발

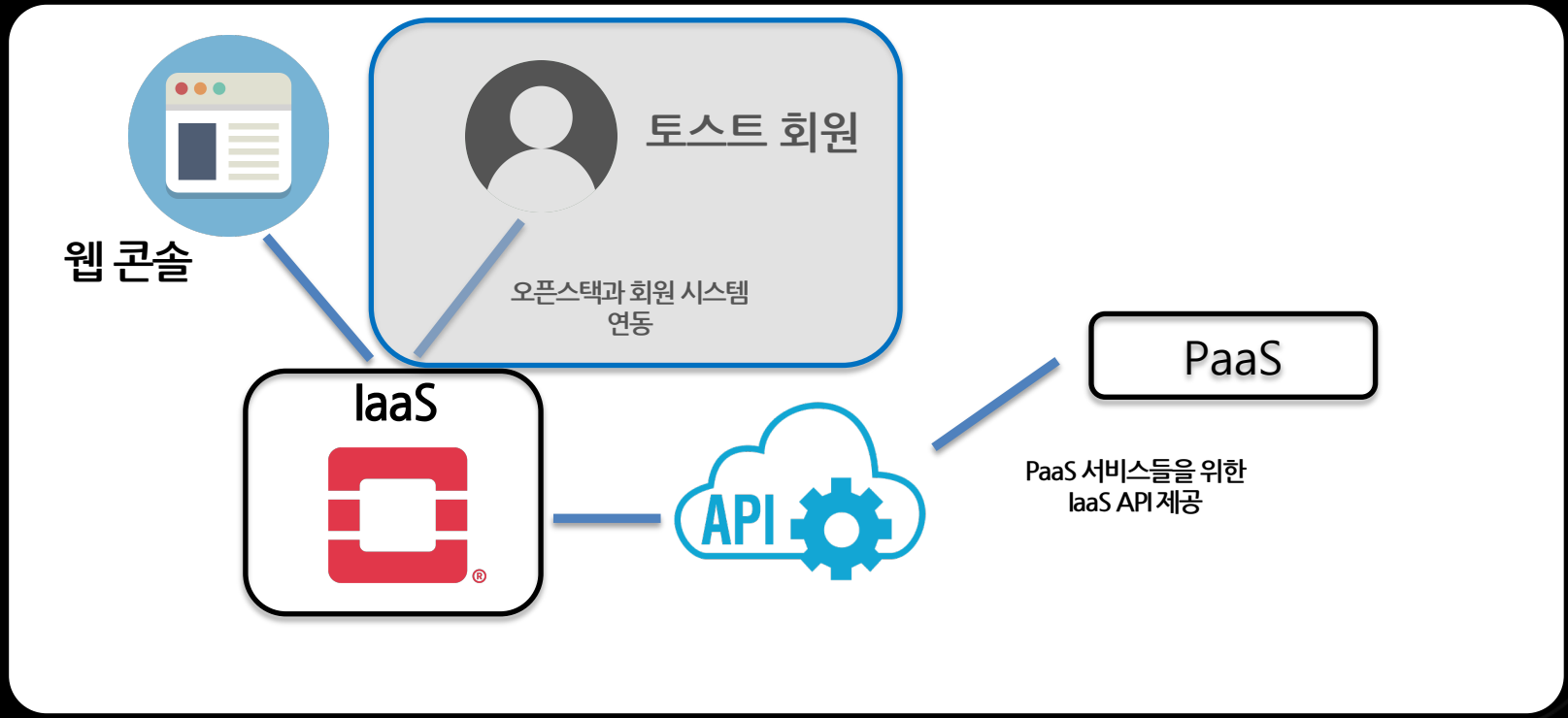
오픈스택 서비스를 어떻게

사용자에게 제공할 것인가?

오픈스택 그 넘어, 토스트 클라우드 서비스



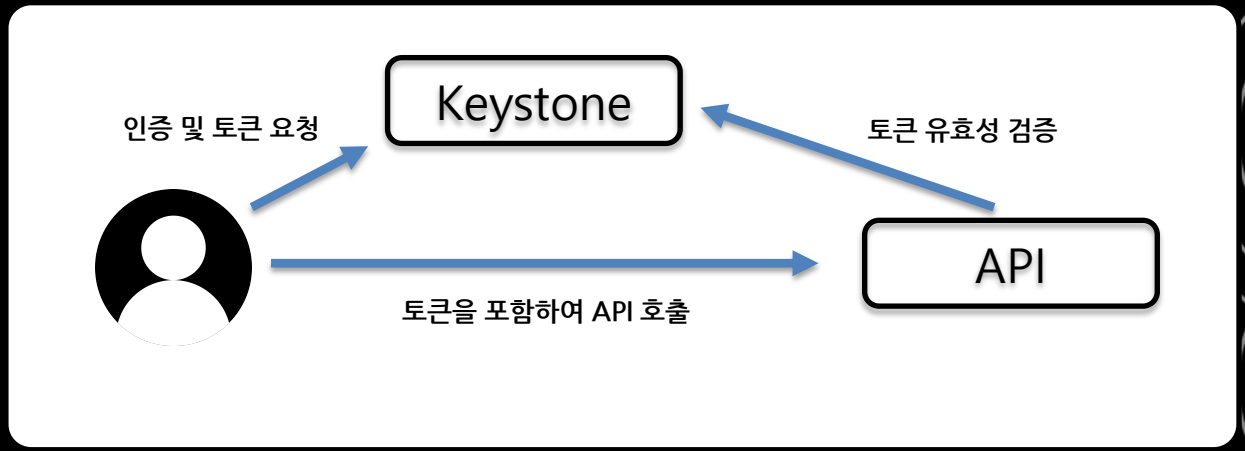
오픈스택 그 넘어, 토스트 클라우드 서비스



Keystone - 유저 관리 / 인증 컴포넌트

Keystone은 오픈스택에서 유저 관리와 인증을 담당하는 컴포넌트

컴포넌트 API를 호출하려면 반드시 Keystone에 인증 후 토큰을 발급 받아야 한다



Keystone의 구성 요소

Keystone에는 유저를 관리하기 위해 3가지 개념이 존재

- 프로젝트 (테넌트 - tenant)

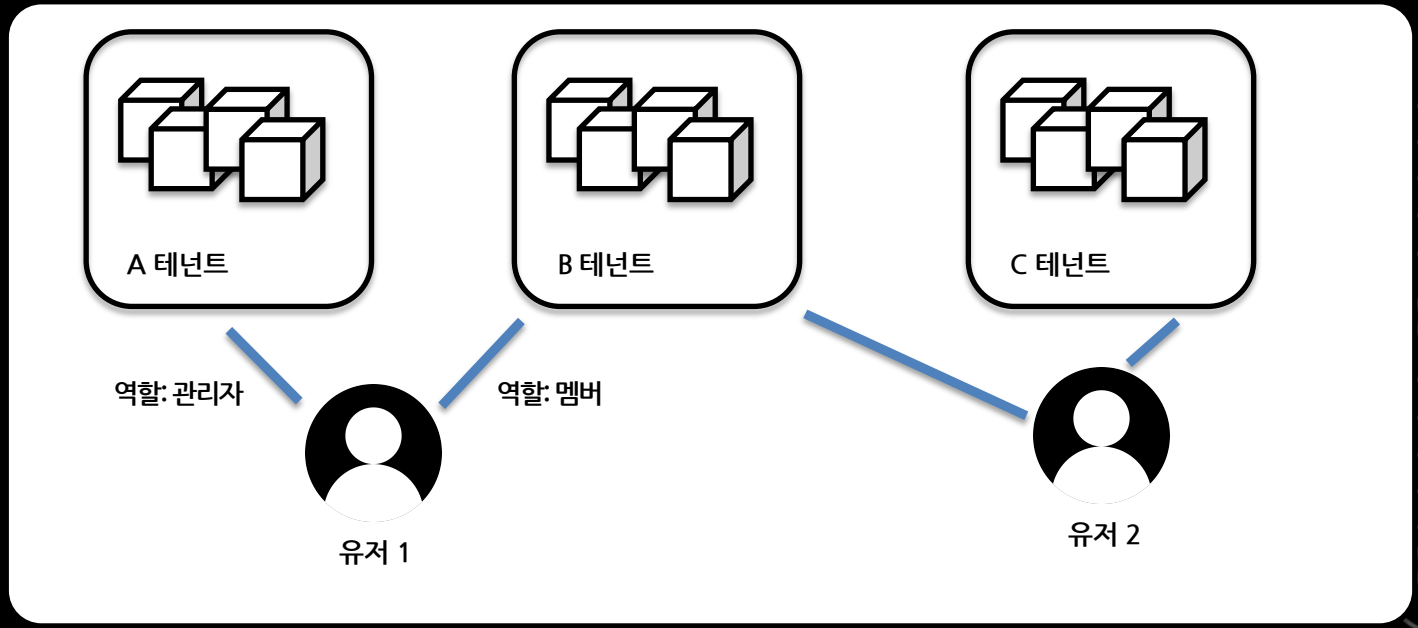
  사용자 그룹 / 자원을 소유하는 주체

  예: 인스턴스는 1개의 프로젝트에 속한다

- 사용자 (User)

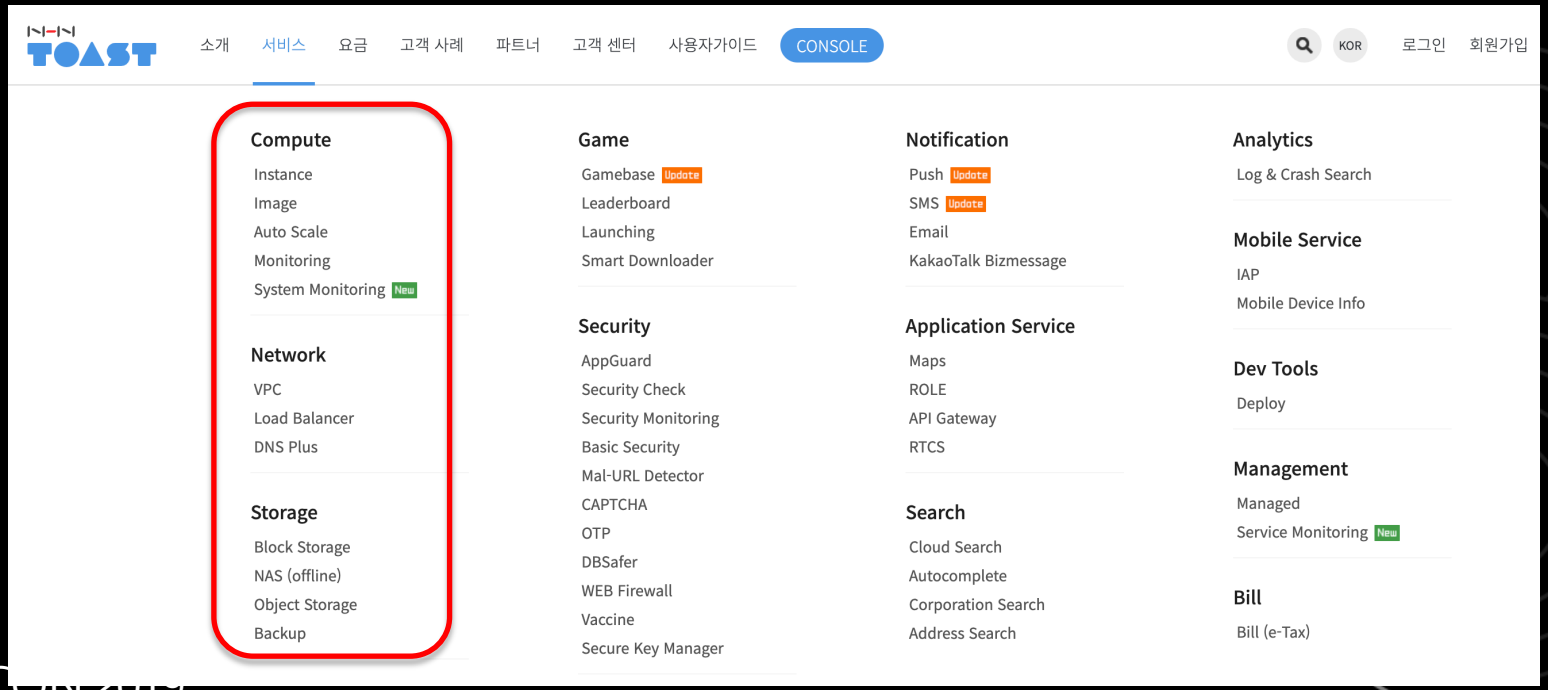
- 역할 (Role)

Keystone의 구성 요소



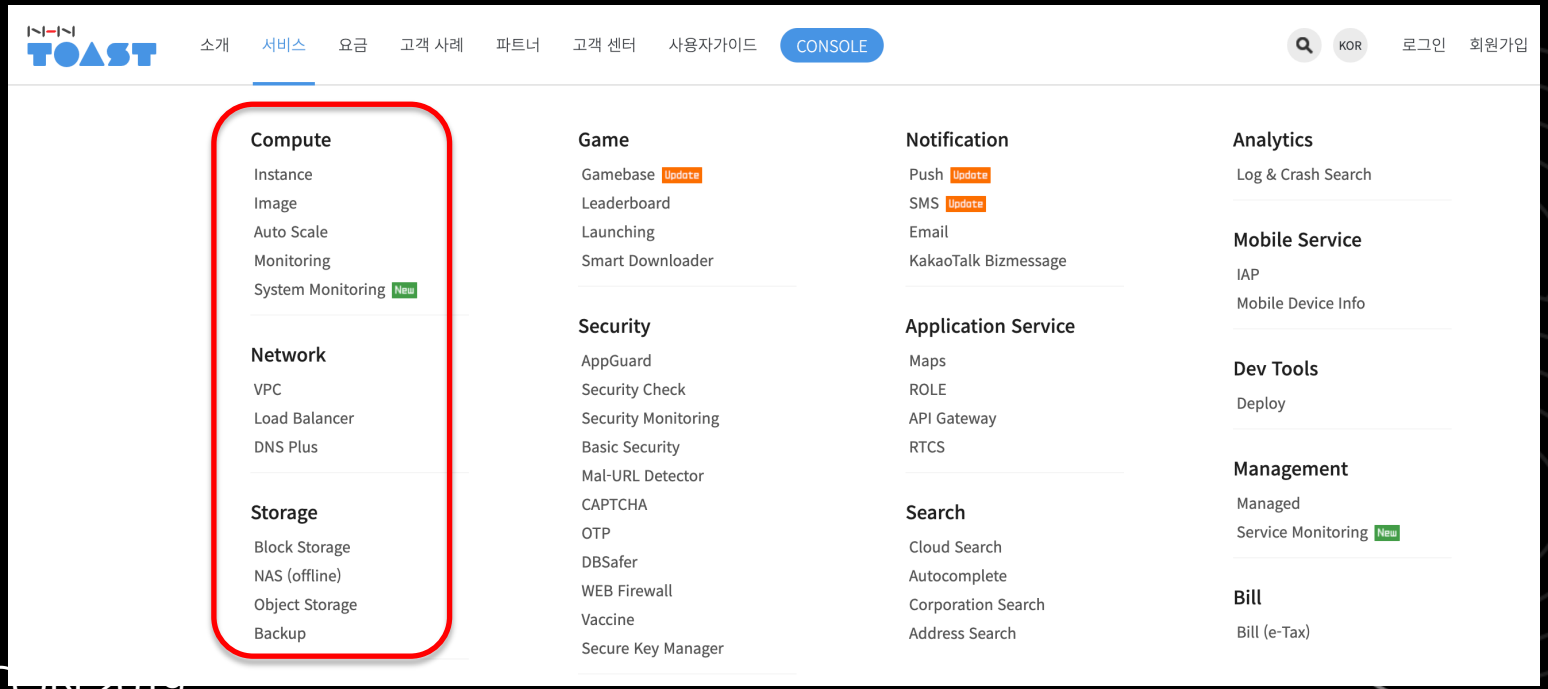
토스트 클라우드 서비스

IaaS는 토스트 클라우드 전체 서비스 중 하나이다



토스트 클라우드 서비스

Keystone으로 이 모든 서비스들의 회원 정보를 관리할 수 없다 (개인정보 등..)



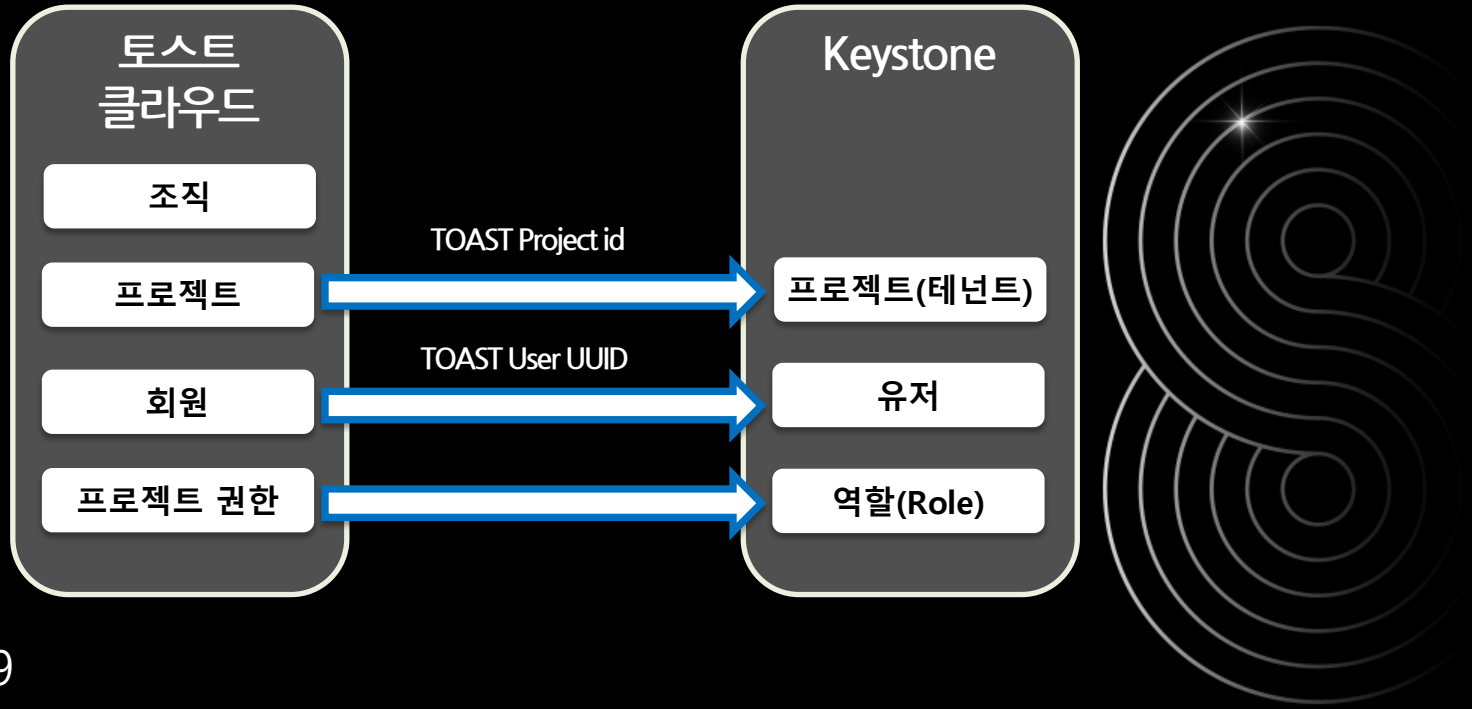
토스트 클라우드 서비스 구조

토스트 클라우드는 크게 조직이 있고, 조직 아래 여러 개의 프로젝트가 만들어진다

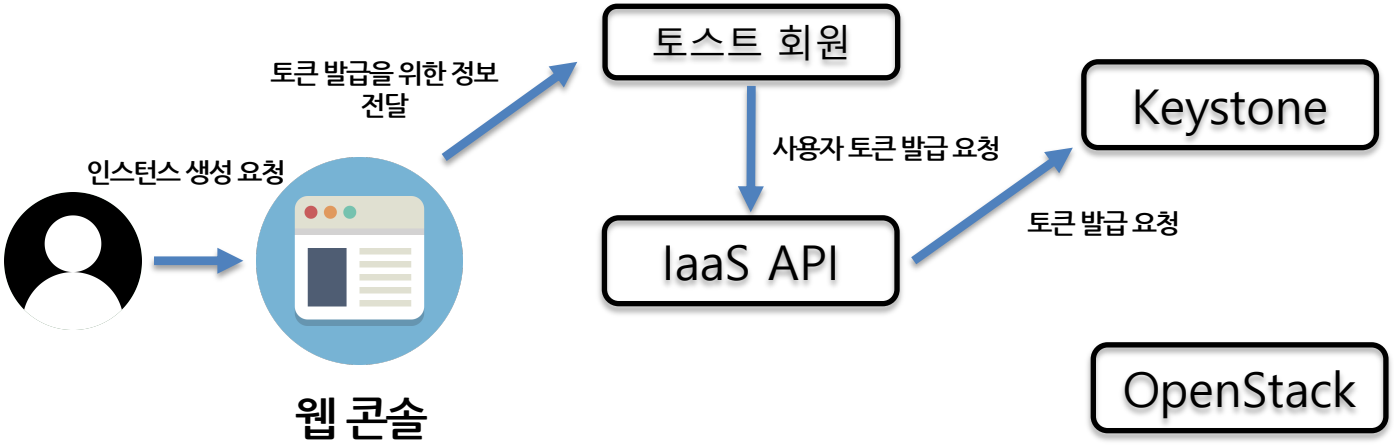
각 프로젝트에서 서비스를 활성화하여 사용할 수 있다

토스트 클라우드와 Keystone 연동

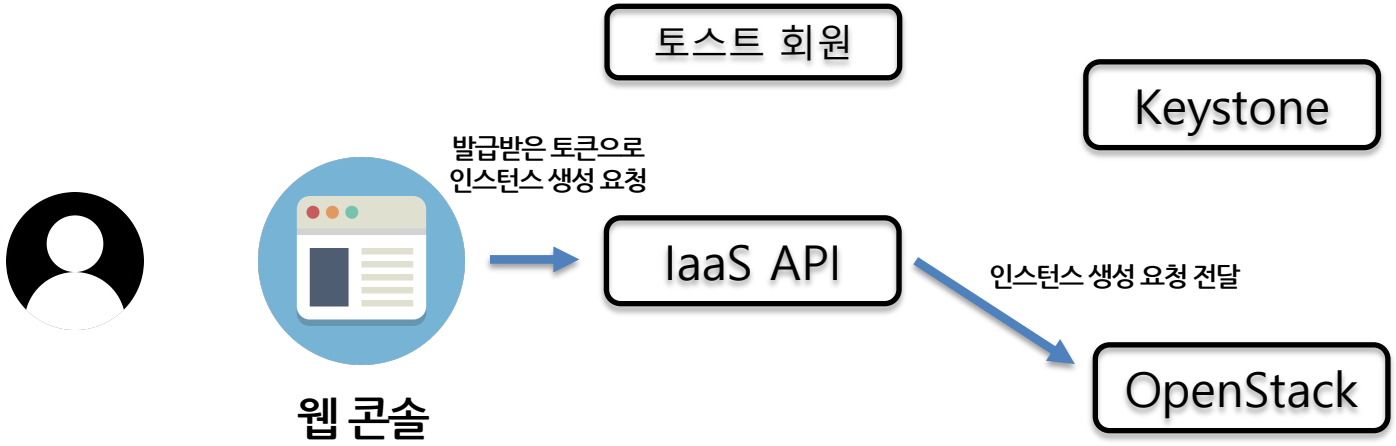
토스트 클라우드 회원 개념을 Keystone에 반영해야한다



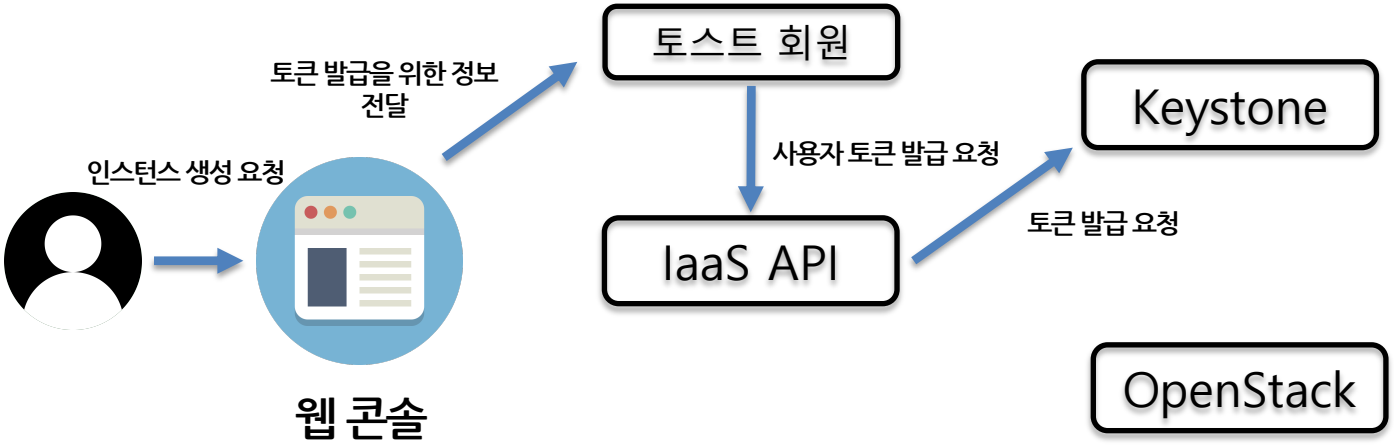
토스트 클라우드에서 키스톤 인증 방법



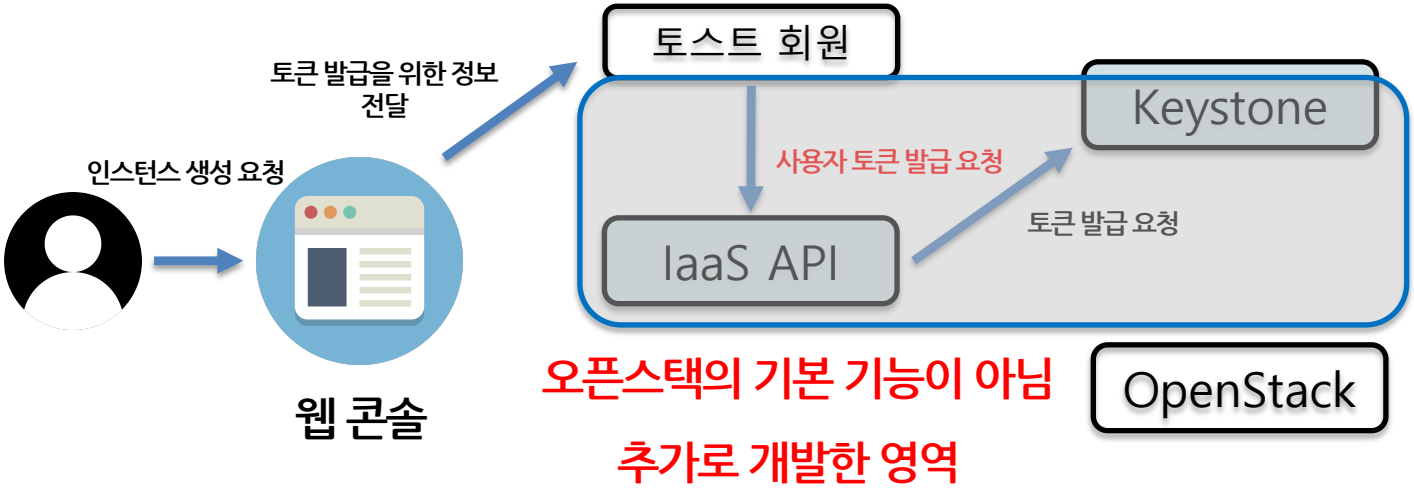
토스트 클라우드에서 키스톤 인증 방법



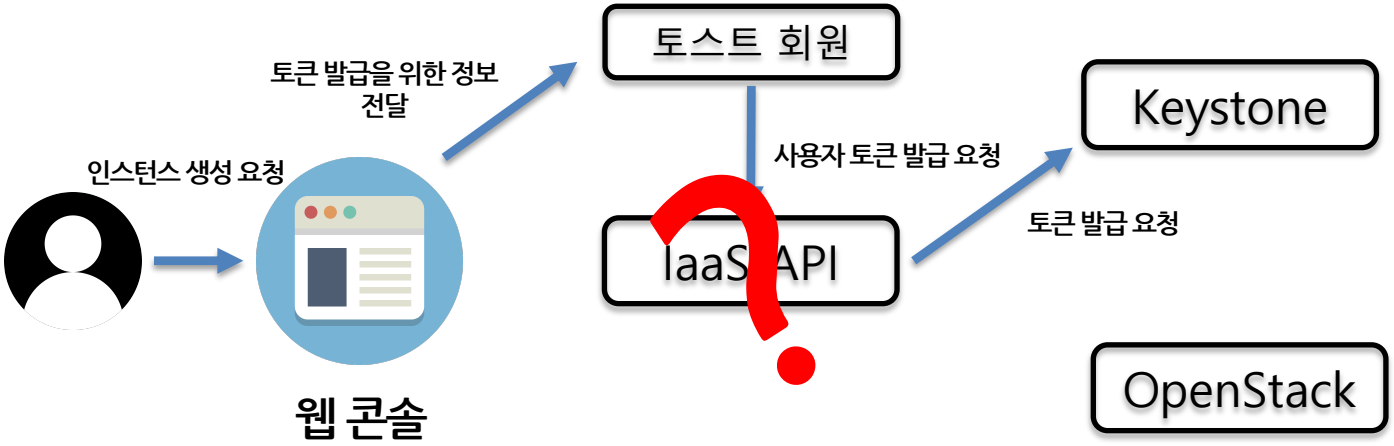
토스트 클라우드에서 키스톤 인증 방법



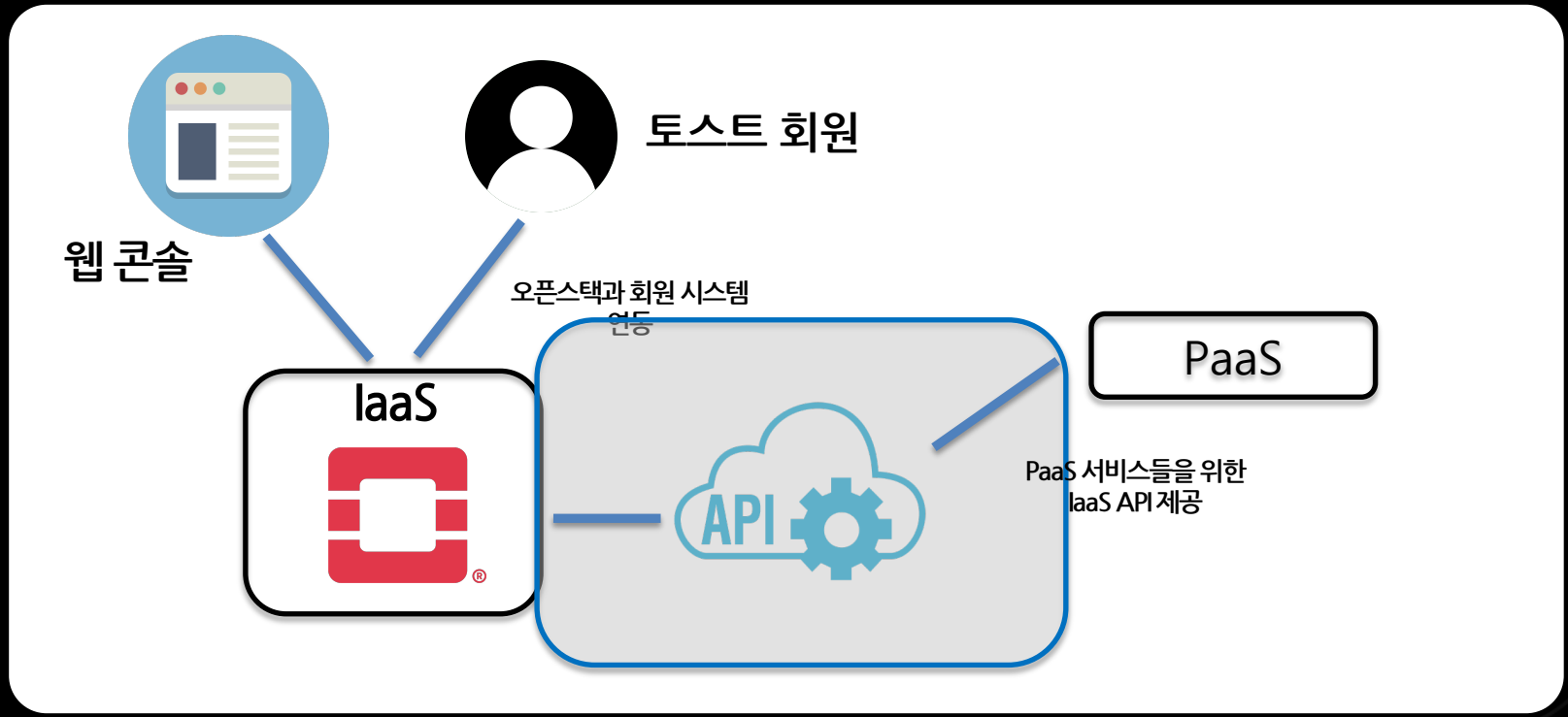
토스트 클라우드에서 키스톤 인증 방법



토스트 클라우드에서 키스톤 인증 방법



클라우드 서비스에 특화된 오픈스택 API



클라우드 서비스에 특화된 오픈스택 API

토스트 클라우드 IaaS의 요구사항

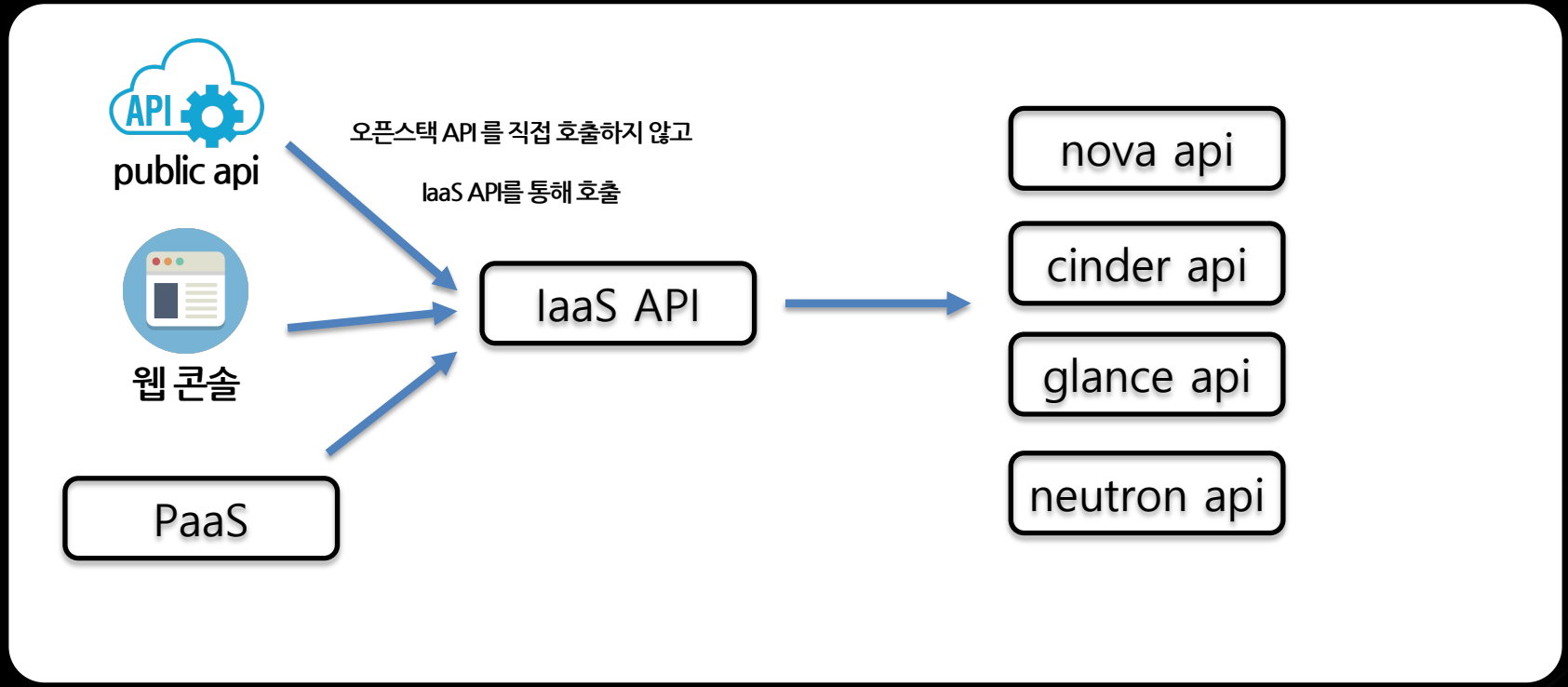
예시 1. 고객 별로 노출해야 하는 네트워크와 서브넷 필터링
- neutron에 등록된 네트워크와 서브넷을 환경에 맞게 필터링해서 보여준다

예시 2. 고객 별로 노출해야 하는 OS이미지 필터링
- 예: A 조직에만 제공하는 이미지를 필터링하여 표시

IaaS API 사용이 필요한, TOAST PaaS 서비스에 API 제공

오픈스택 API을 한번 감싼, 매핑된 API를 제공

클라우드 서비스에 특화된 오픈스택 API



오픈스택 코드 개발

토스트 클라우드는 kilo 버전을 기반으로 기능 개선 / 버그 패치를 통해 안정된 오픈스택을 운영중

오픈스택은 6개월마다 새로운 릴리즈 발표

지속적으로 새로운 버전으로 업그레이드 / 버그 패치 / 기능 추가 필요

오픈스택 코드 개발 - 업그레이드

최신 버전

업그레이드 지원 버전

사용중인 버전

Series	Status	Initial Release Date
Ussuri	Future	2020-05-13 *estimated* (schedule)
Train	Development	2019-10-16 *estimated* (schedule)
Stein	Maintained	2019-04-10
Rocky	Maintained	2018-08-30
Queens	Maintained	2018-02-28
Pike	Extended Maintenance	2017-08-30
Ocata	Extended Maintenance	2017-02-22
Newton	End Of Life	2016-10-06
Mitaka	End Of Life	2016-04-07
Liberty	End Of Life	2015-10-15
Kilo	End Of Life	2015-04-30
Juno	End Of Life	2014-10-16

오픈스택 코드 개발 - 업그레이드

Series	Status	Initial Release Date
Ussuri	Future	2020-05-13 *estimated* (schedule)
Train	Development	2019-10-16 *estimated* (schedule)
Stein	Maintained	2019-04-10
Rocky	Maintained	2018-08-30
Queens	Maintained	2018-02-28
Pike	Extended Maintenance	2017-08-30
Ocata	Extended Maintenance	2017-02-22
Newton	End Of Life	2016-10-06
Mitaka	End Of Life	2016-04-07
Liberty	End Of Life	2015-10-15
Kilo	End Of Life	2015-04-30
Juno	End Of Life	2014-10-16

어떻게든
여기까진
올라가야

오픈스택 코드 개발 - 업그레이드

손쉬운
업그레이드
가능

어떻게든
여기까진
올라가야

Series	Status	Initial Release Date
Ussuri	Future	2020-05-13 *estimated* (schedule)
Train	Development	2019-10-16 *estimated* (schedule)
Stein	Maintained	2019-04-10
Rocky	Maintained	2018-08-30
Queens	Maintained	2018-02-28
Pike	Extended Maintenance	2017-08-30
Ocata	Extended Maintenance	2017-02-22
Newton	End Of Life	2016-10-06
Mitaka	End Of Life	2016-04-07
Liberty	End Of Life	2015-10-15
Kilo	End Of Life	2015-04-30
Juno	End Of Life	2014-10-16

오픈스택 코드 개발 - 업그레이드

한 단계씩

업그레이드

현실적으로
불가능

Series	Status	Initial Release Date
Ussuri	Future	2020-05-13 *estimated* (schedule)
Train	Development	2019-10-16 *estimated* (schedule)
Stein	Maintained	2019-04-10
Rocky	Maintained	2018-08-30
Queens	Maintained	2018-02-28
Pike	Extended Maintenance	2017-08-30
Ocata	Extended Maintenance	2017-02-22
Newton	End Of Life	2016-10-06
Mitaka	End Of Life	2016-04-07
Liberty	End Of Life	2015-10-15
Kilo	End Of Life	2015-04-30
Juno	End Of Life	2014-10-16

오픈스택 코드 개발 - 백포팅

상위 버전에서 해결된 버그 패치 / 새로운 기능을 가져온다

상위 버전의 코드를 하위 버전 코드로 백포팅(Back Porting)

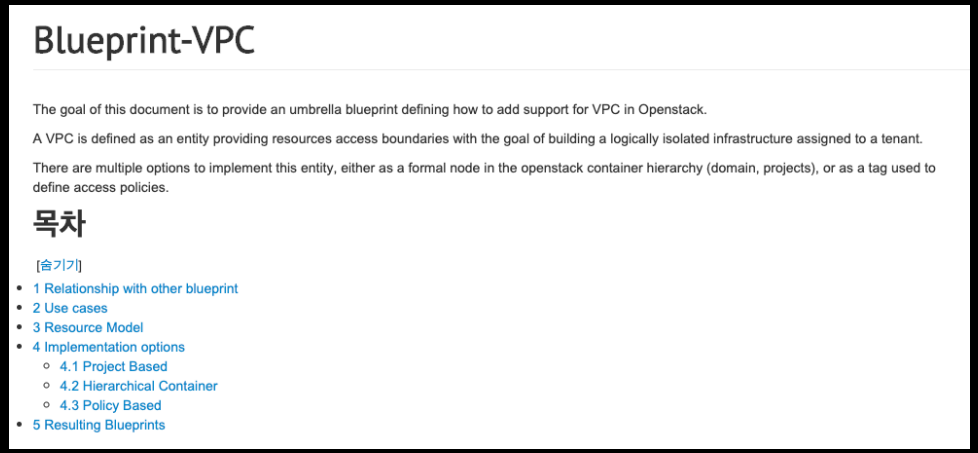
업그레이드가 아닌 추가 개발 및 자체 유지 보수

오픈스택 코드 개발 - 신규 기능 개발

아직 오픈스택에 없는 신규 기능 개발

예 1. VPC (Virtual Private Cloud) 지원

아직 오픈스택에 없는 기능 / 아이디어 제안 상태 (Blueprint)



Blueprint-VPC

The goal of this document is to provide an umbrella blueprint defining how to add support for VPC in Openstack.

A VPC is defined as an entity providing resources access boundaries with the goal of building a logically isolated infrastructure assigned to a tenant.

There are multiple options to implement this entity, either as a formal node in the openstack container hierarchy (domain, projects), or as a tag used to define access policies.

목차

[숨기기]

참고 : https://wiki.openstack.org/wiki/Blueprint-VPC

오픈스택 코드 개발 - 신규 기능 개발

오픈스택 외부 시스템을 이용한 기능 개발

예 2. 컴퓨트 노드 장애 시, 인스턴스 자동 evacuate

- Rokcy 버전부터 Masakari 라는 프로젝트에서 기능 지원 (인스턴스 대상)

- 외부 시스템을 활용하여 컴퓨트 노드 장애를 감지하고, 수 분 내로 인스턴스를 다른 컴퓨트 노드로 옮기는 기능을 추가 개발

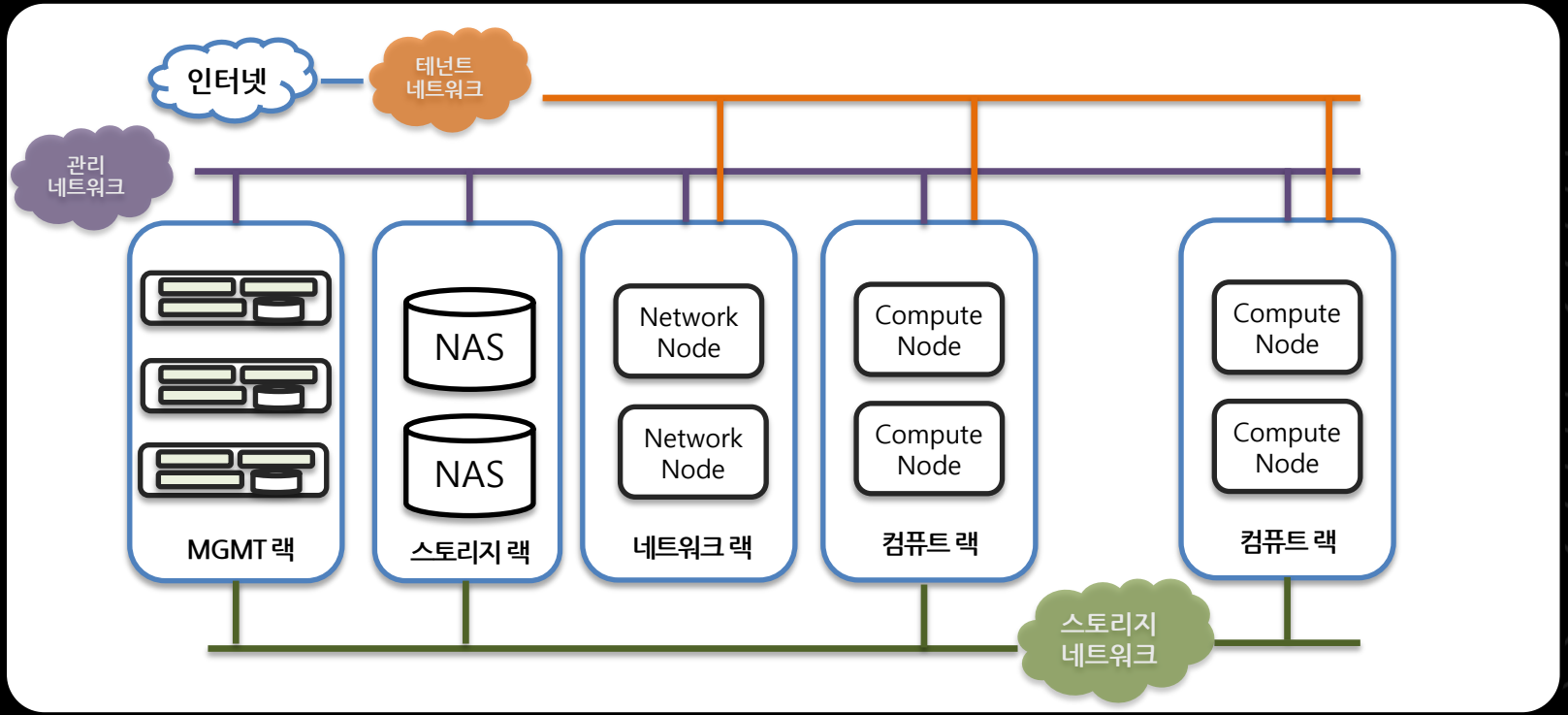
클라우드 인프라는

어떻게 개발할까?

SOSCON 2019

SAMSUNG OPEN SOURCE CONFERENCE 2019

개발환경을 어떻게 만들 것인가?



개발환경을 어떻게 만들 것인가?

오픈스택 공식 개발환경 : devstack

1개 혹은 2개 이상의 VM으로 오픈스택 환경을 손 쉽게 구축 가능

〈구축 예시〉

```
$ git clone https://opendev.org/openstack/devstack
$ cd devstack
$ ./stack.sh
```

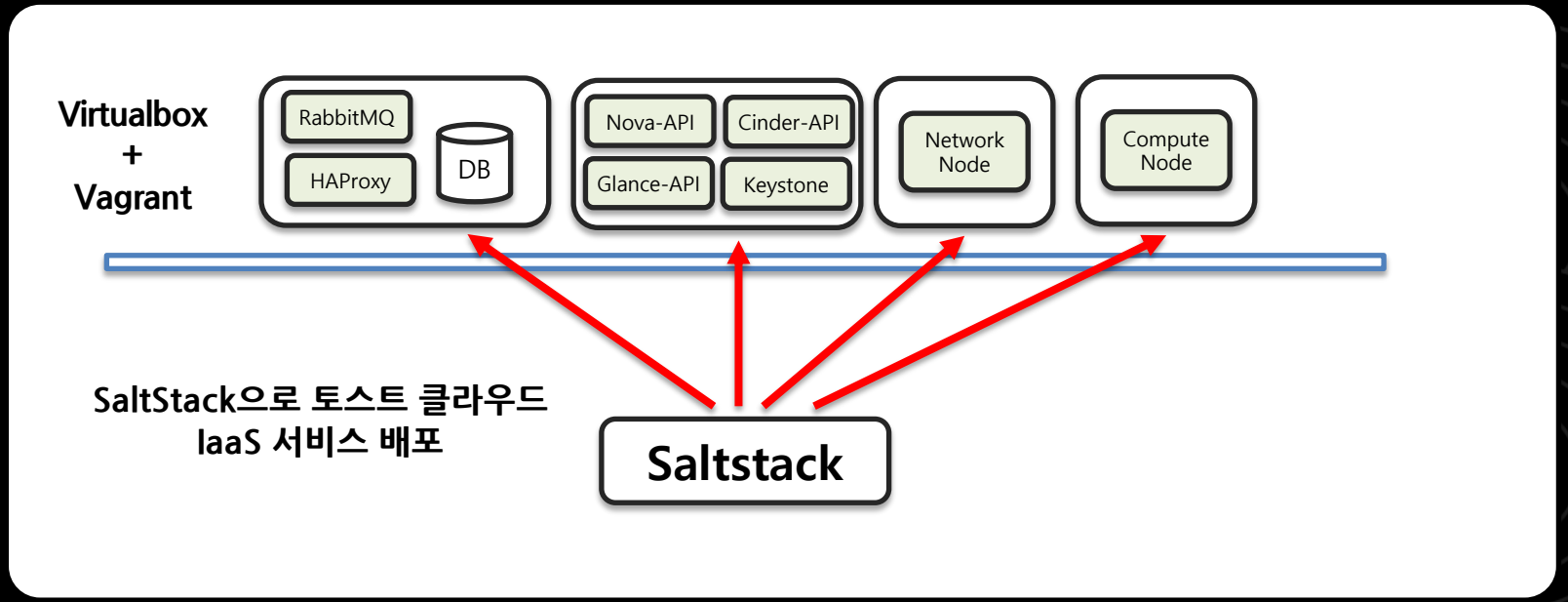
개발환경을 어떻게 만들 것인가?

Devstack의 한계

- 토스트 클라우드 환경에 맞는 설정 불가
- 토스트 클라우드 IaaS 배포 시스템 활용 불가
- 자체적으로 개발한 코드를 유연하게 배포할 수 없음

손 쉽게 만드는 토스트 클라우드 개발 환경

Virtualbox + Vagrant + Saltstack 으로 토스트 클라우드 개발 환경 구축



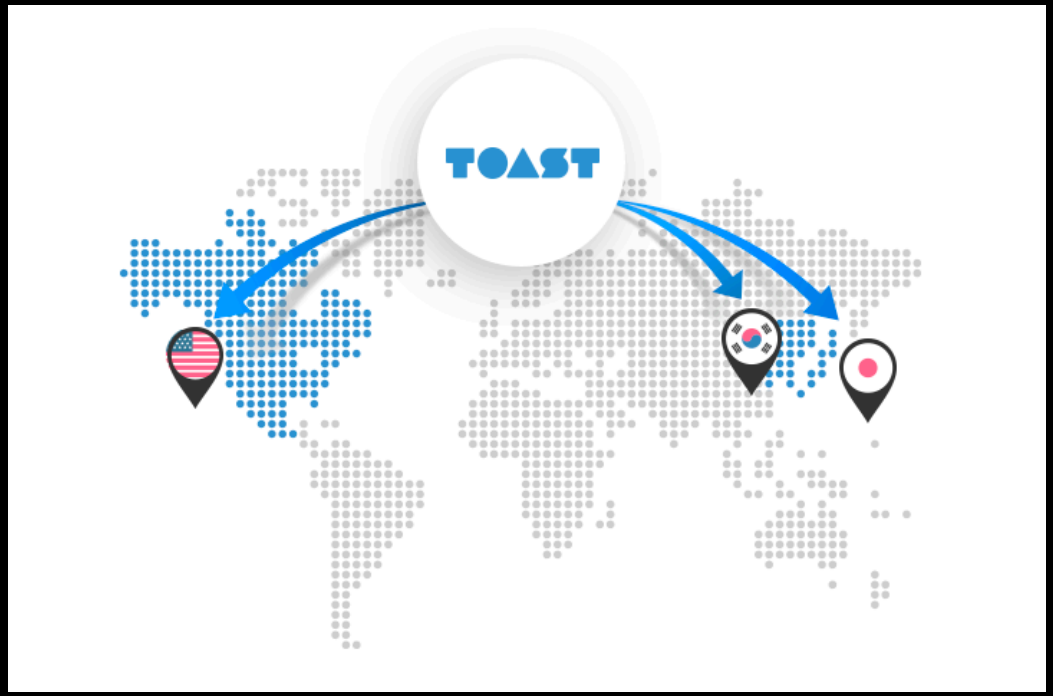
글로벌 리전으로 확장하기

SOSCON 2019

SAMSUNG OPEN SOURCE CONFERENCE 2019

오픈스택으로 글로벌 리전 만들기

토스트 클라우드는 한국/일본/미국 리전 그리고 공공/금융 환경을 가지고 있다

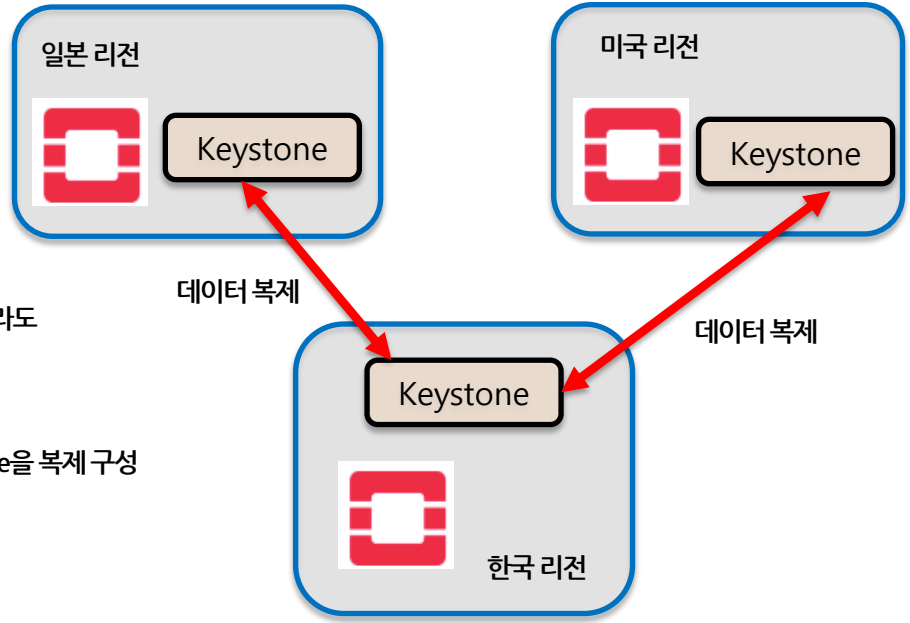


오픈스택으로 글로벌 리전 만들기

각 리전마다 분리된 오픈스택 환경을 구축

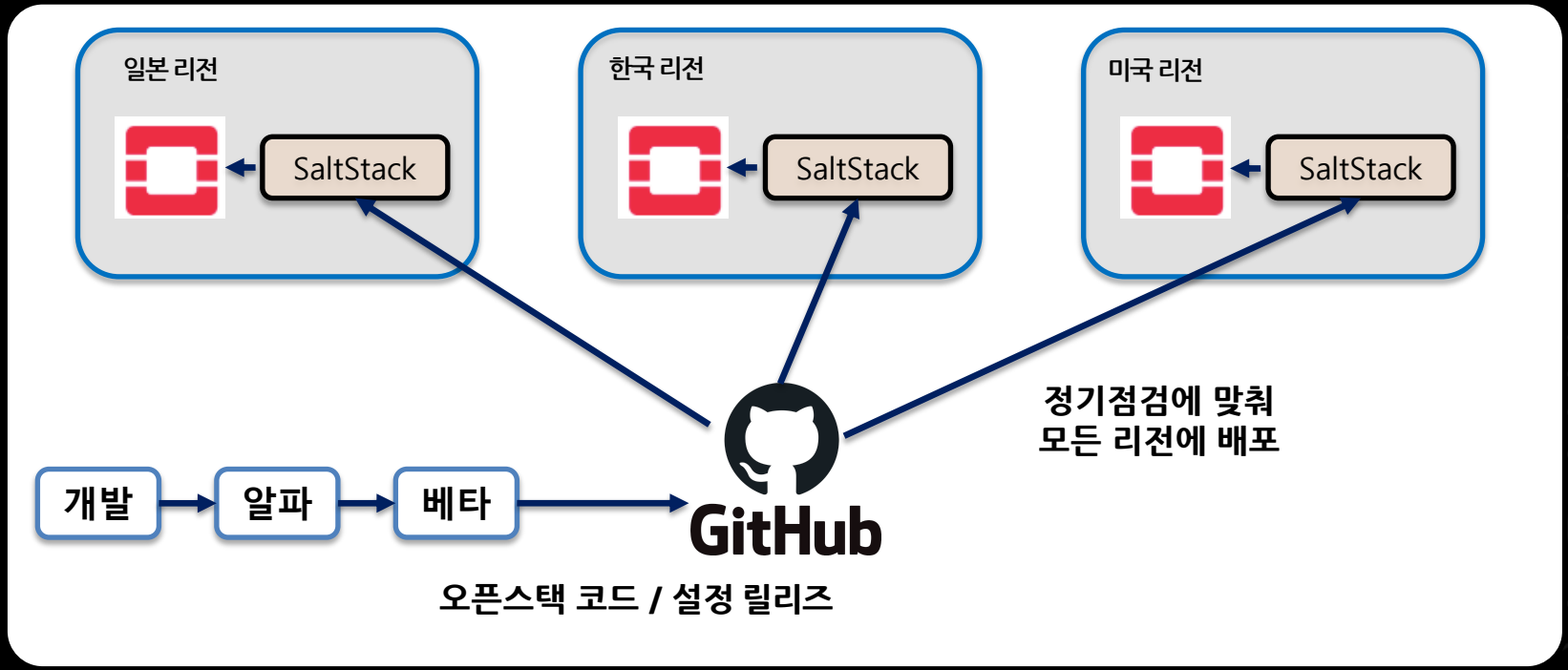
1. 한국 리전에서 가입 후, 일본 리전에 인스턴스를 만들더라도
   키스톤 회원 정보가 유지되어야 한다.

2. 한국 리전의 Keystone을 master로 각 리전에 keystone을 복제 구성



오픈스택으로 글로벌 리전 만들기

리전마다 오픈스택 배포하기



더 나은 클라우드를 위한

새로운 아키텍처와 컴포넌트 개발

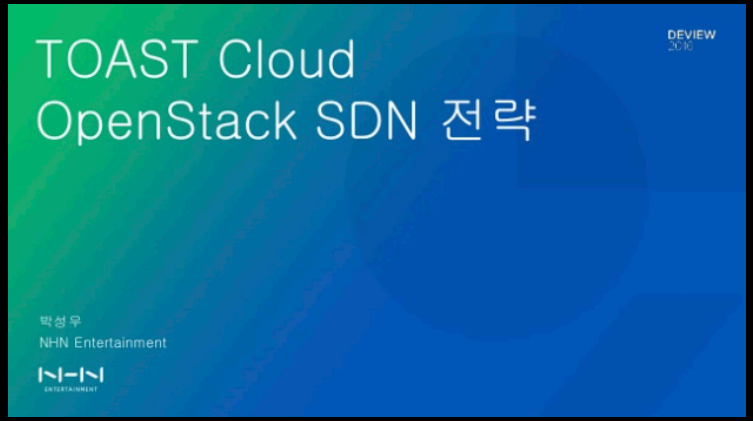
SOSCON 2019

SAMSUNG OPEN SOURCE CONFERENCE 2019

더 나은 클라우드를 위한 새로운 아키텍처

새로운 버전 / 네트워크 성능 향상 / 손 쉬운 확장을 위한 새로운 아키텍처

DPDK 기반의 neutron 전용 TOAST vSwitch 개발



Deview 2016: TOAST Cloud OpenStack SDN 전략

Open Infrastructure Days Korea 2019

마무리

SOSCON 2019

SAMSUNG OPEN SOURCE CONFERENCE 2019

오픈스택기반 클라우드 서비스를 위한 지속적인 고민

고민1. 오픈스택을 구동할 환경을 어떻게 만들 것인가?

컴포넌트들의 고 가용성(HA), 손쉬운 확장(Scalability)
가상 네트워크의 구성 그리고 인터넷으로 서비스 방법
글로벌 리전(Region) 배포

고민2. 오픈스택 서비스를 어떻게 사용자에게 제공할 것인가?

토스트 회원, 사용자 웹 콘솔, 오픈스택에 새로운 기능 개발

오픈스택으로 퍼블릭 클라우드 서비스 하기

모두가 오픈스택을 잘 알아야 한다

오픈스택 코드를 이해하고 개발할 수 있는 개발팀

오픈스택을 잘 이해하고 문제 상황을 잘 대처할 수 있는 운영팀

클라우드 서비스와 오픈스택의 관계를 잘 이해하는 고객대응팀

오픈스택으로도 퍼블릭 클라우드 서비스를 할 수 있다

오픈스택 코드를 잘 이해하고, 개발할 수 있어야 한다

THANK YOU

SOSCON 2019

SAMSUNG OPEN SOURCE CONFERENCE 2019