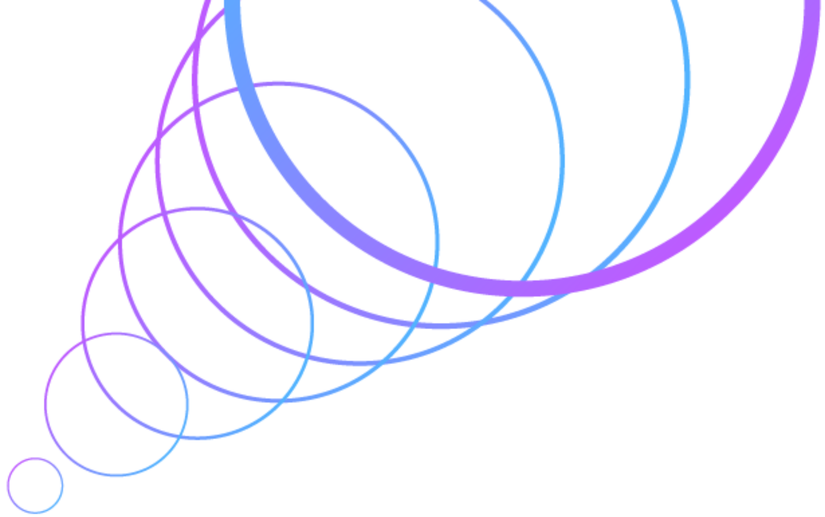
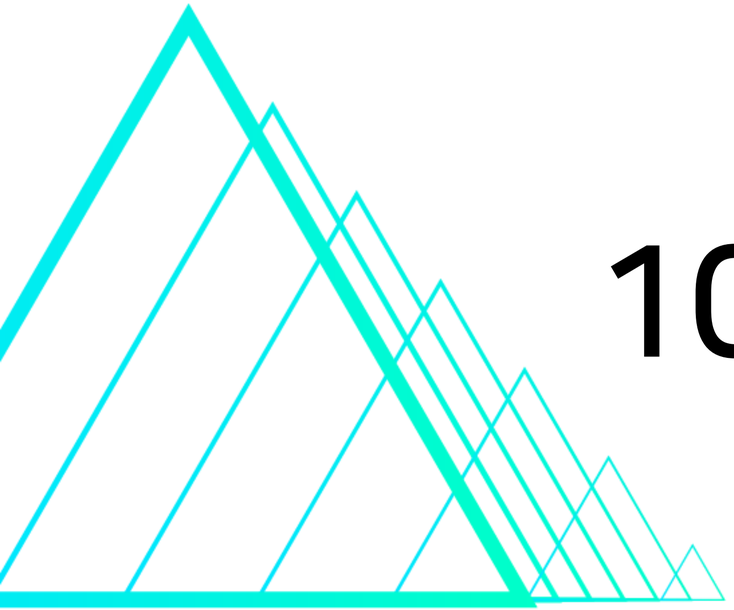




10만 컨테이너 양병론

안혁준 NAVER / Platform Labs / Computing Platform

CONTENTS

DEVIEW 2021

Intro

Who we are?

DEVIEW 2021

Computing Platform

Naver의 Computing 환경을 Container 기반으로 만들어 가는 팀

2019

- Multi-tenancy Kubernetes on Bare-metal server(Naver Container Cluster)

2020

- Container SRE
- K8s eBPF/XDP 기반 고성능&고가용성 NAT 시스템

Naver Container 환경의 성장

DEVIEW 2021

2019

12+Cluster

300+ Namespace

10k+ Pod

20k+ Container

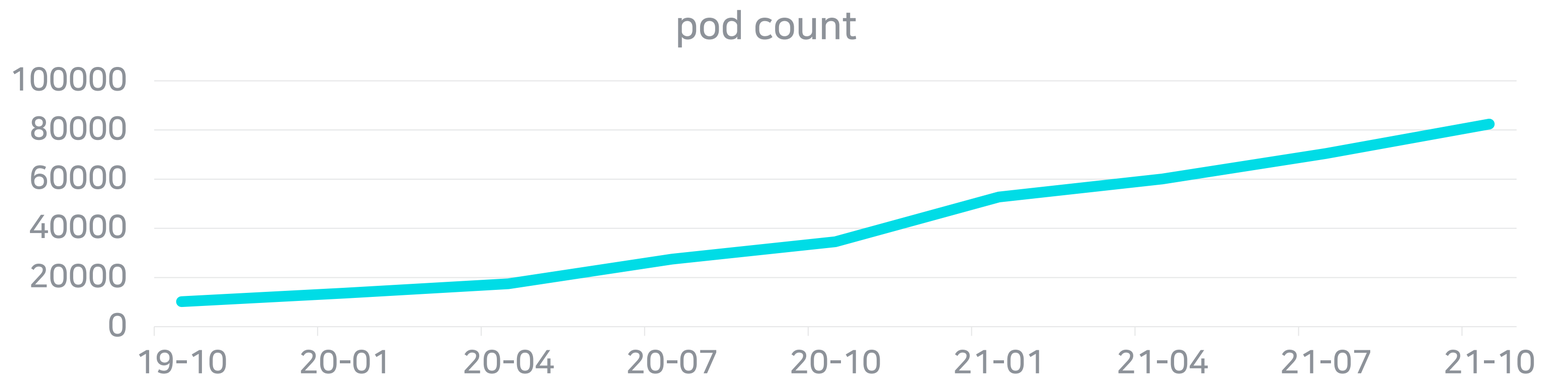
2021

25+ cluster

2000+ Namespace

80k+ Pod

200k+ Container

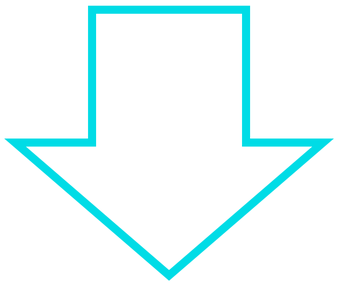


Who we are?

DEVIEW 2021

Computing Platform의 목표

Naver의 Computing 환경을 Container 기반으로 만든다.

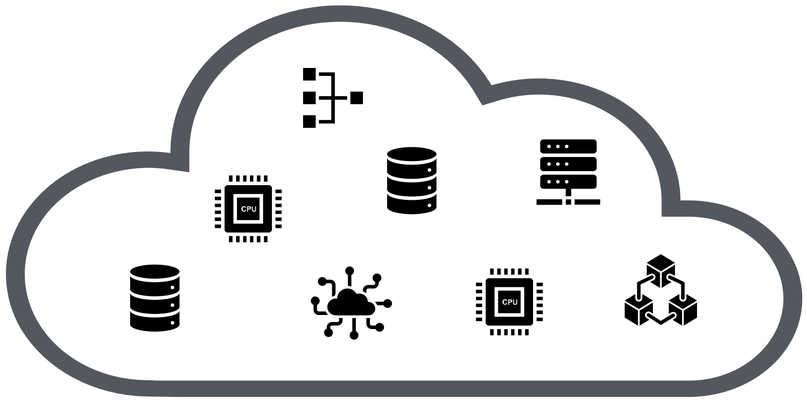


Naver의 Cloud Native 환경을 만든다.

Cloud Native

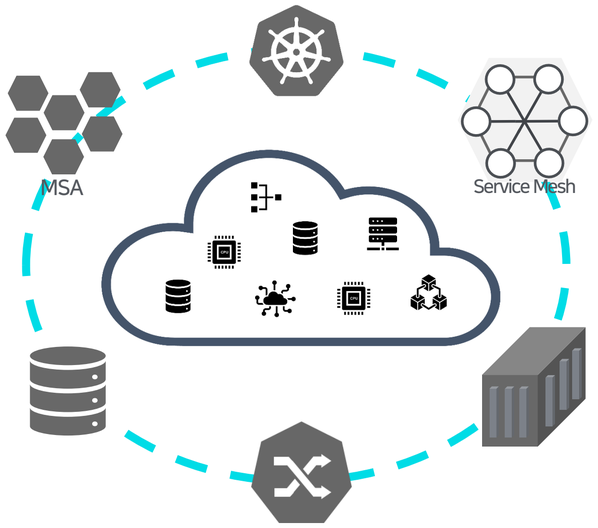
DEVIEW 2021

Cloud Computing



언제 어디서나 Cloud에 있는
Computing Resource가 제공되는 것

Cloud Native



Cloud Computing의 장점을
최대한 활용하는 어플리케이션을 개발하고 운영하는 것

Cloud Native

DEVIEW 2021

CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape https://github.com/cncf/landscape has a growing number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator

https://www.cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

http://cncf.io/kcsp

C. Join CNCF's End User Community

For companies that don't offer cloud native services externally

http://cncf.io/enduser

WHAT IS CLOUD NATIVE?

- **Operability:** Expose control of application/system lifecycle.
- **Observability:** Provide meaningful signals for observing state, health, and performance.
- **Elasticity:** Grow and shrink to fit in available resources and to meet fluctuating demand.
- **Resilience:** Fast automatic recovery from failures.
- **Agility:** Fast deployment, iteration, and reconfiguration.

www.cncf.io
info@cncf.io

v10

1. CONTAINERIZATION

- Normally done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing

3. ORCHESTRATION

- Pick an orchestration solution
- Kubernetes is the market leader and you should select a Certified Kubernetes Platform or Distribution
- https://www.cncf.io/ck

kubernetes CNCF Graduated

4. OBSERVABILITY & ANALYSIS

- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

Prometheus CNCF Incubating · fluentd CNCF Incubating · JAEGER CNCF Incubating · OPENTRACING CNCF Incubating

5. SERVICE MESH

- Connects services together and provides ingress from the Internet
- Service discovery, health checking, routing, load balancing
- Consider Envoy, Linkerd and CoreDNS

envoy CNCF Incubating · CoreDNS CNCF Incubating

6. NETWORKING

To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net.

CNI CNCF Incubating

7. DISTRIBUTED DATABASE

When you need more resiliency and scalability than you can get from a single database, Vitess is a good option for running MySQL at scale through sharding.

Vitess CNCF Incubating

8. MESSAGING

When you need higher performance than JSON-REST, consider using gRPC.

gRPC CNCF Incubating

9. CONTAINER RUNTIME

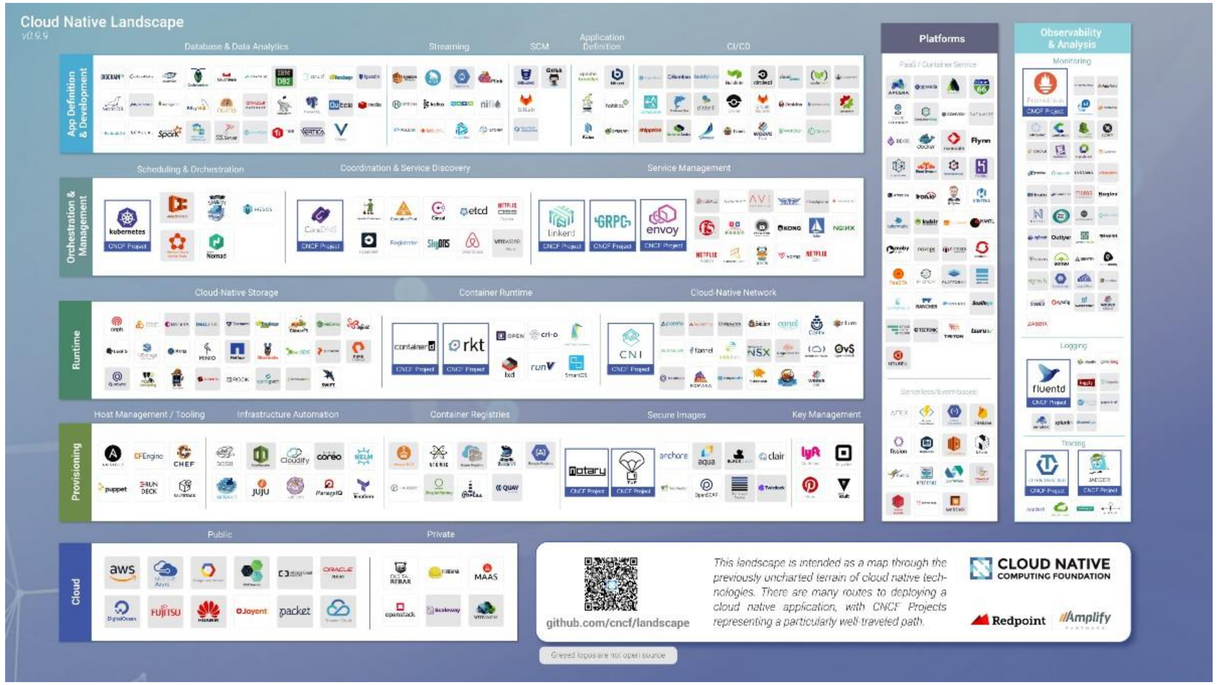
You can use alternative container runtimes. The most common, all of which are OCI-compliant, are containerd, rkt and CRI-O.

containerd CNCF Incubating · rkt CNCF Incubating

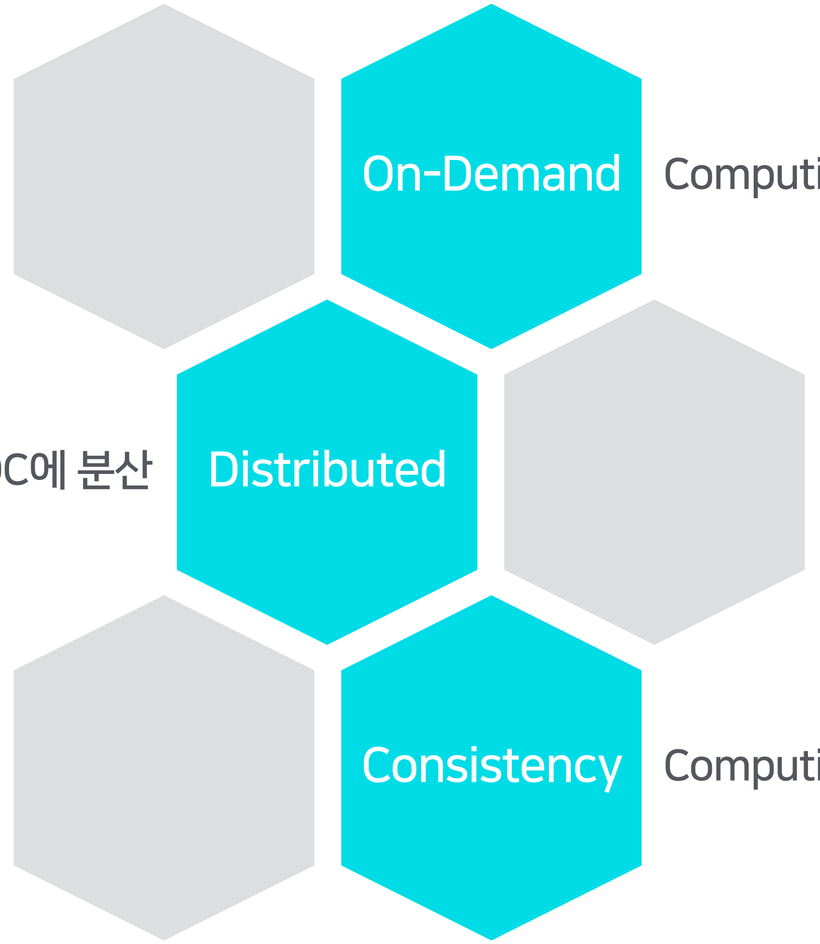
10. SOFTWARE DISTRIBUTION

If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.

Notary CNCF Incubating · TUF CNCF Incubating



Cloud Native Environment for Naver



Computing Resource는 사용자 요구에 따라 즉시 제공

Computing Resource는 여러 Region/IDC에 분산

Computing Resource는 동일한 Interface로 사용

DEVIEW 2021

10만 컨테이너 양병론

10만 양병론

율곡 이이

10만의 군사를 키워 후일을 대비 해야 합니다.

10만 양병론

DEVIEW 2021

율곡 이이

10만의 군사를 키워 후일을 대비 해야 합니다.

서애 류성룡

10만 군사를 키우기에는 너무 힘듭니다.

10만 양병론

임진왜란 발발!

서애 류성룡

이이의 말이 맞았구나

10만 Container 양병론

10만 개의 Co tainer를 만들어야 합니다?

10만 개의 Container를 안정적으로 서비스 할 수 있는 기반을 만들어야 한다.

Cloud Native 환경

DEVIEW 2021

N3R

Naver를 위한 Cloud Native 환경

- Kubernetes → k8s

  Greek for “helmsman” or “pilot” or “governor”

  (키잡이(조타수), 조종사, 총독/주지사/운영위원)

- Naver → n3r

  Navigator (조종사, 항해사) + -er (사람)

N3R

Multi-tenancy cluster
- 네이버 환경에서 최적화 된 컨테이너 환경
- Spec은 같지만 요구사항에 따라 권한/정책/설정이 변경

Network isolation
- Multi-Tenancy Kubernetes의 Virtual Private Cluster 구축(Policy Based)

Elastic Load Balancer
- L7 공통 기능 및 Load Balancer 제공

Cloud Application
- 공통의 컴퓨팅 리소스(ELB, Monitoring/Logging,…)를 활용 하여 쉽고 빠른 서비스 개발을 지원

Monitoring

Logging

Stateful
- ELK, Vitess, 등의 Open-Source Project 를 위한 Scheduling, Storage 정책/운영 지원

Multi-tenancy Cluster

DEVIEW 2021

Multi-Tenancy Kubernetes 기반

- 장점: 효율이 좋음, 사용자가 Kubernetes 운영하지 않아도 됨
- 단점: 다양한 요구 사항을 만족하기 어려움(GPU, Operator, CRD)

Spec은 같고 정책과 설정만 다른 다수의 Cluster

- Type별로 Cluster를 나누고 이 cluster를 Multi-Tenancy로 제공
  - n3r.public: 일반 웹 앱 사용자용.
  - n3r.dedicated: 특정 조직에 제공되는 Cluster(GPU 클러스터, 전자금융법 클러스터)
  - n3r.statuful: Stateful을 운영 하기 위한 Cluster(Elastic Search, Vitess, Kafka 전용 cluster)
- 사용자가 많으면 같은 정책이 필요한 사용자도 다수

Multi-Tenancy k8s의 장점 극대화/단점 최소화

- namespace는 많이 증가 했지만, Cluster는 상대적으로 적게 증가함

Network Isolation(1/3)

IP ACL

- 예전부터 사용 했던 방식
- 아직 다수의 Server 들이 IP ACL 를 사용함.

Multi-tenancy Kubernetes는 IP ACL이 어려움

- Floating IP, Grouping 불가
- Namespace 별 IP Group을 구분 할 방법이 없음.
- IP ACL은 고정된 IP 나 IP 대역이 필요하지만 k8s에서는 불가능

-> Namespace 별로 Virtual Private Cluster(VPC)를 구성

IP ACL을 위한 component를 추가

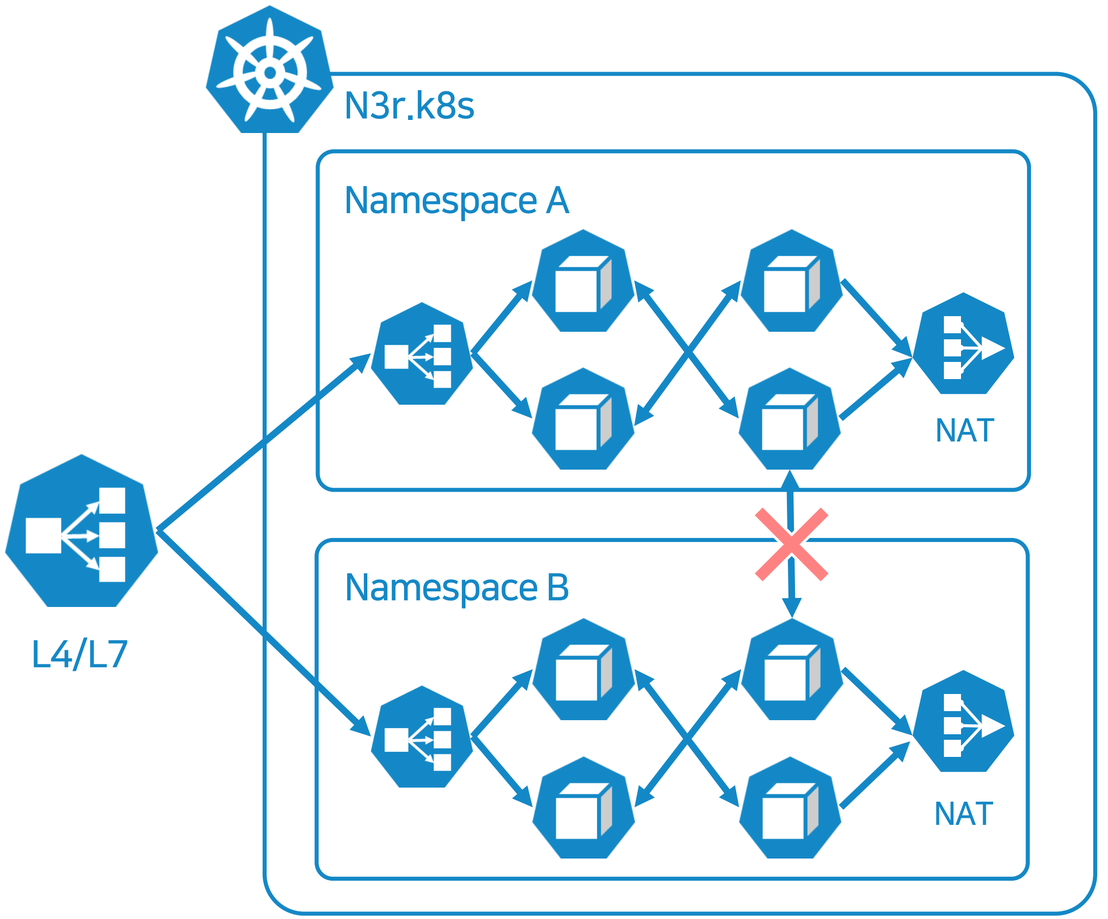
Network Isolation(2/3)

Inbound Traffic Aggregation
- L4/L7

Outbound Traffic Aggregation
- NAT Gateway

Namespace 간 Traffic 격리
- Policy Based Network

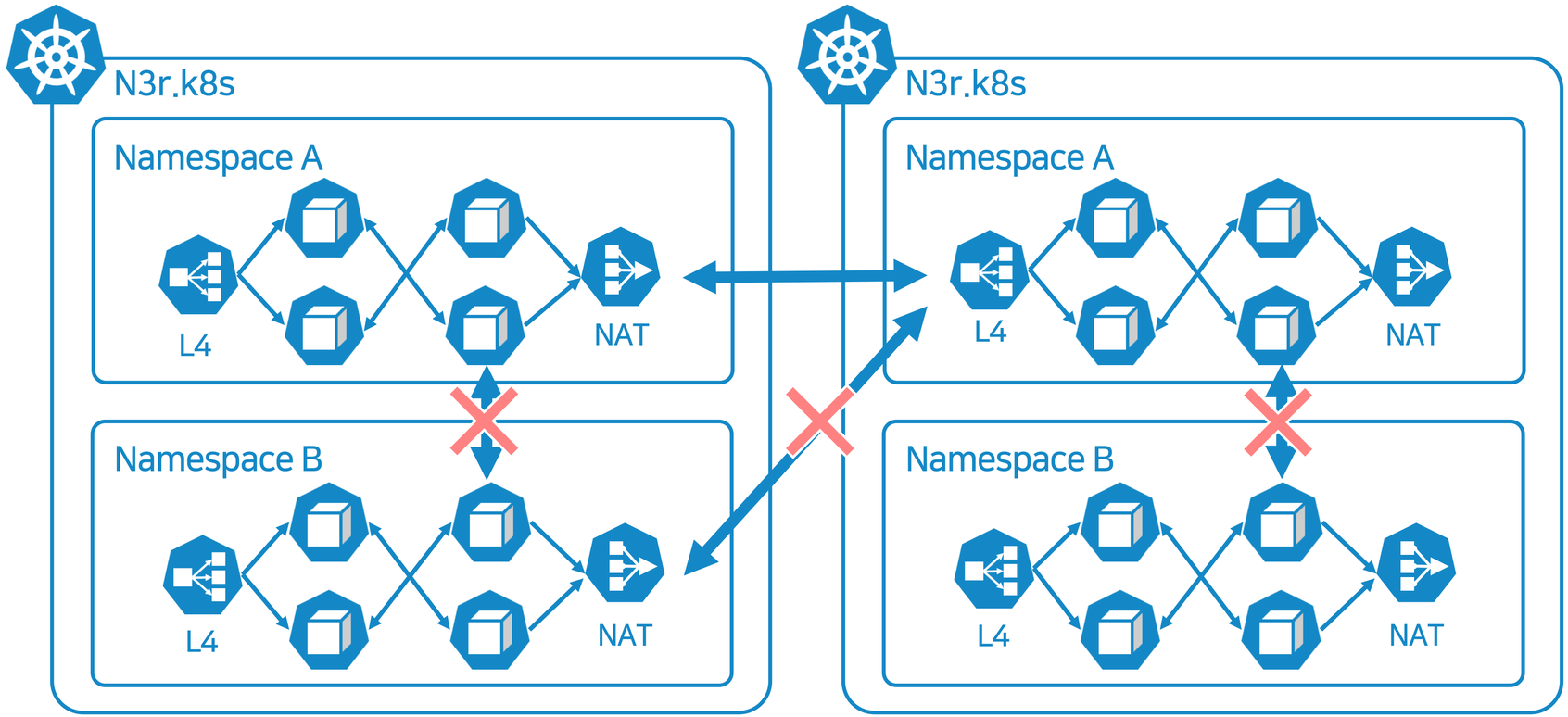
L4, NAT Gateway의 IP는 고정
- 이를 이용하여 IP ACL



Network Isolation(3/3)

Inter-Cluster간 ACL Policy 제어

- NAT – LB 간 ACL Policy 자동/중앙 집중 제어



Elastic Load Balancer

N3R L7에서 필요한 기능

- Failover/High Availability/Scale out
- Cert
- Authentication
- Path/Header based routing
- Rate limit/Circuit break
- Programable Interface

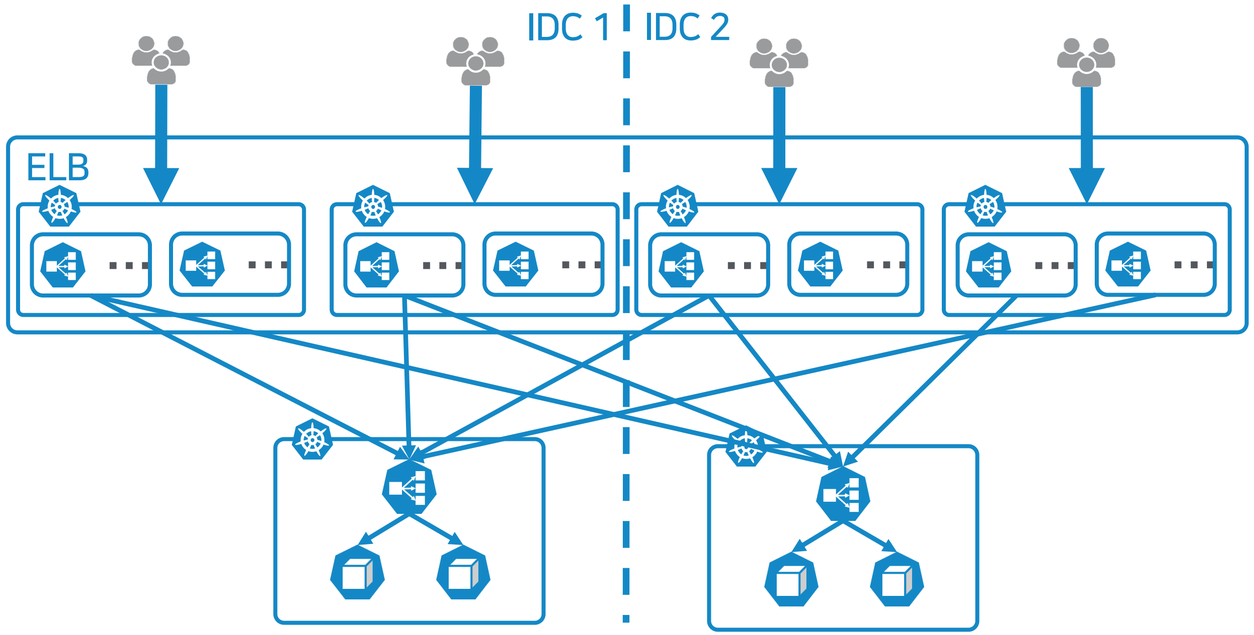
-> 기능 전체를 모두 제공 하는 시스템이 없음

Elastic Load Balancer

DEVIEW 2021

Inbound Traffic을 Region/Cluster로 분배

- 외부에서 Naver로 들어오는 관문
  - 어느 Region 으로 들어오더라도 Application이 배포되어 있는 Region/Cluster로 Routing
- 2개 Region * 2개 Cluster = 4중화
  - Cluster간 Failover
- Scale out에 유리한 구조



"envoy로 동적 L7 로드벨런서 만들기 – 김동경" 발표 참조

Cloud Application

DEVIEW 2021

Kubernetes에서 서비스를 띄우기 위해서 학습 해야 할 것

- Docker, Kubernetes, container monitoring/logging, helm chart, build/deploy strategy, CI/CD …
- 평균적으로 2주~4주 가량 소요

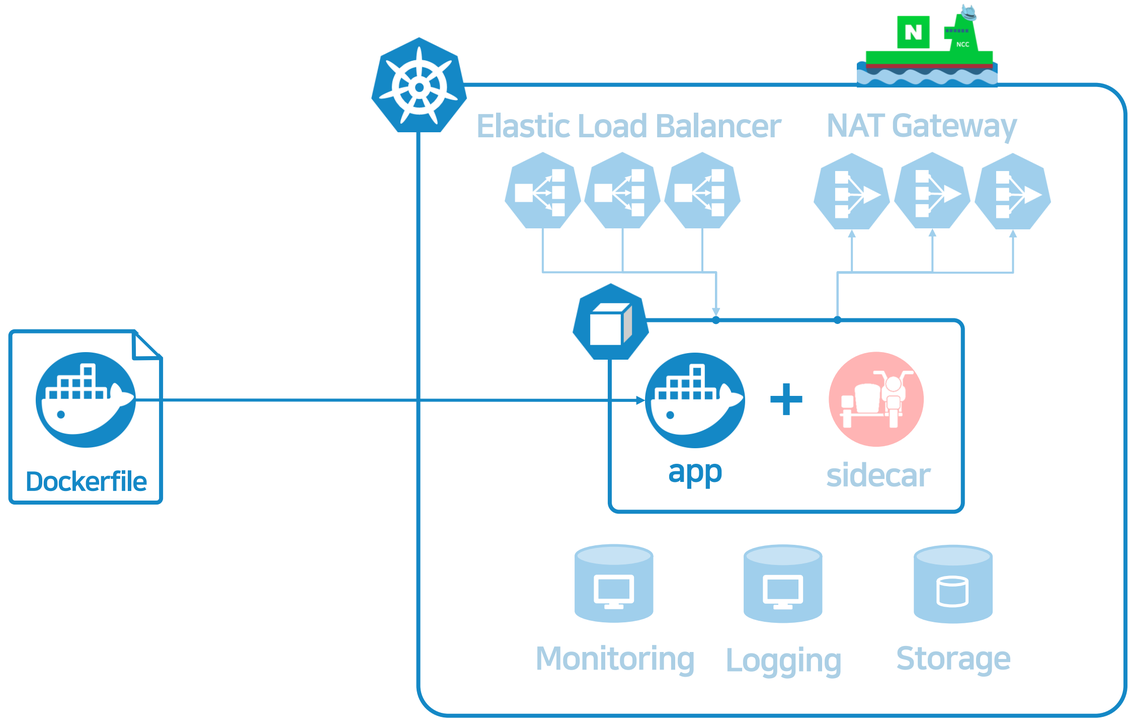
사용자의 70%는 거의 같은 설정으로 사용

- Deployments, Configmap, Volume, Health check, Secrets, HPA, …

Cloud Application

가볍고 빠른 Container App Release

- Dockerfile만으로 App을 구성
  - 필요한 N3R component를 권장 설정으로 자동 설정
  - Heroku/Cloud Run과 유사
- 좋은점
  - 빠른 Application 개발
  - 다양한 실수에 의한 장애 예방



Monitoring

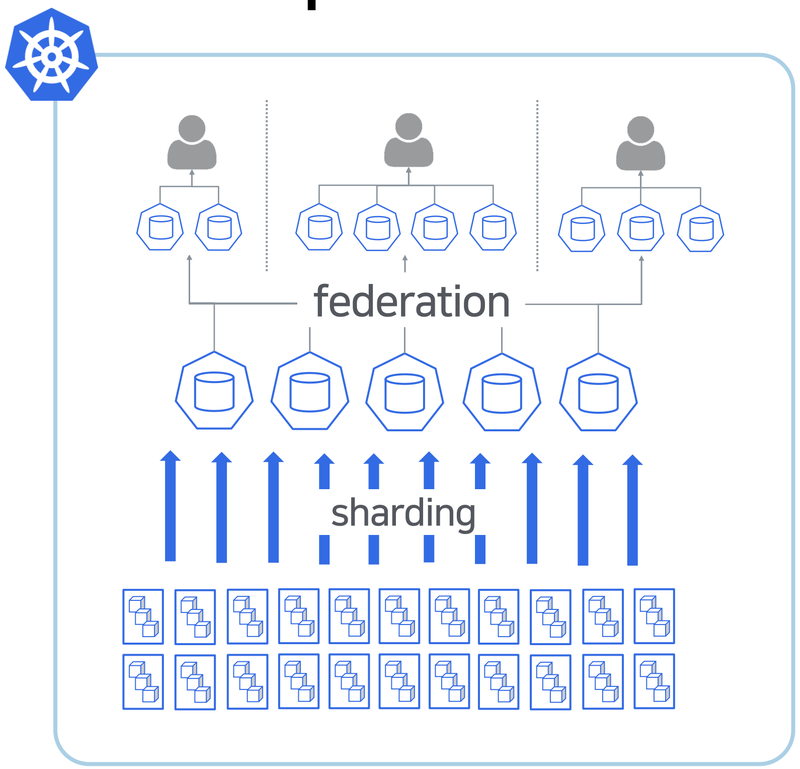
기본적인 Monitoring는 누구나 필요

Monitoring의 Issue

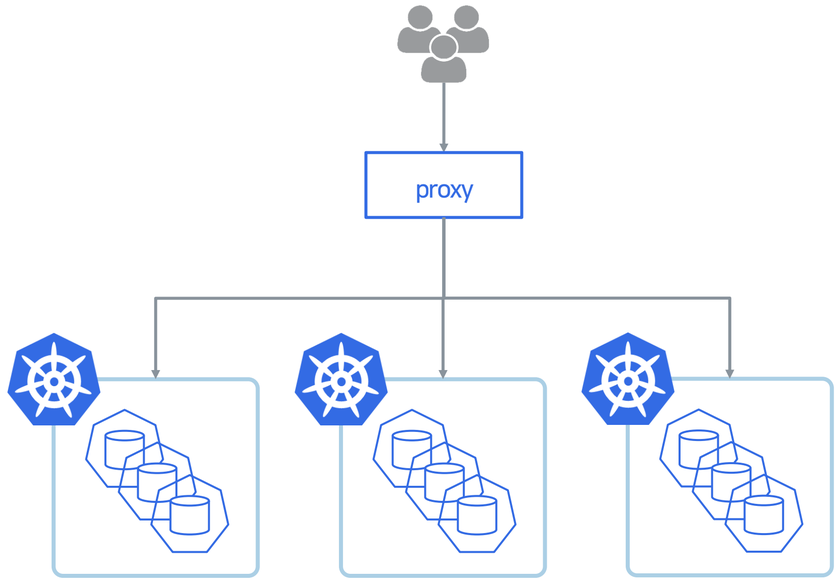
- 사용자 별로 요구사항이 달라질 경우가 많음
  - 서비스 개발자
  - 플랫폼 개발자
  - N3R 운영자
- 필요에 따라서 Alert의 기준 등 Rule이나 설정의 변경이 많음
- Scale out이 쉬워야 함

Monitoring

Divide & Conquer



Horizontally Divide : intra-cluster monitoring



Vertically Divide : inter-cluster monitoring

"Cloud Monitoring Deep Dive – 이승하" 발표 참조

Logging

Log 데이터의 특성

- 쌓아 두고 문제가 생길 때 만 본다.
- 데이터가 일정 기간 반드시 보존 되어야 한다.
- 쌓인 데이터의 중요도를 판별 하기 어렵다.

기존 Log System(Elastic Search)의 단점

- 모든 log가 Index 되어 저장

Logging

Log 시스템을 세가지 Type으로 분리하여 처리

- Loki
  - Log를 그냥 보관
  - 문제가 생기기 전 확인 하지 않는 log(e.g. Access log)
- Elastic Search
  - Log를 index 하여 보관
  - 자주 확인 하고 검색해야 하는 log(e.g. Audit log)
- Sentry
  - Log를 정제해서 Event/Alert 발생
  - 특정한 규칙에 따라 잘 정제된 log 및 Alert용

Stateful

Stateful

- K8s 에 설치 되는 Open Source들

Stateful이 종류가 많아져야 서비스 개발자가 손쉽게 사용할 수 있다.

Stateful

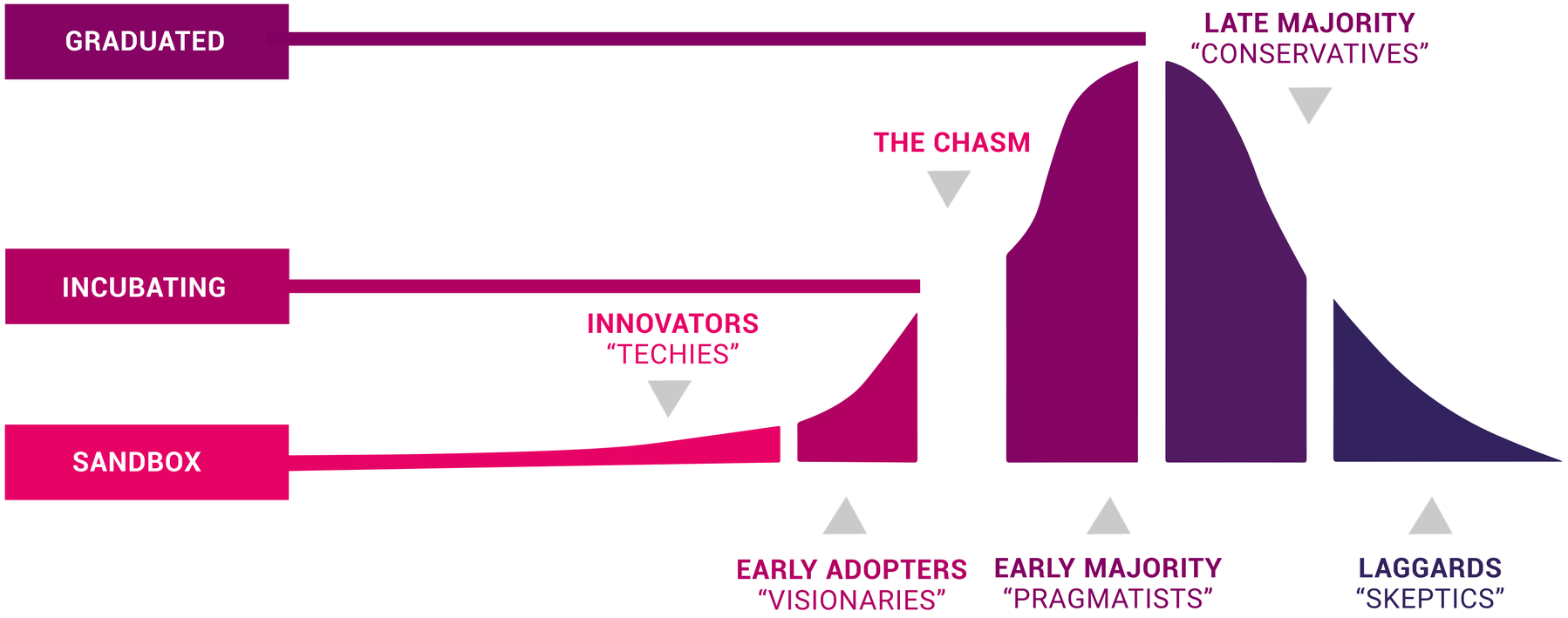
Open source를 k8s에서 Production Grade로 운영

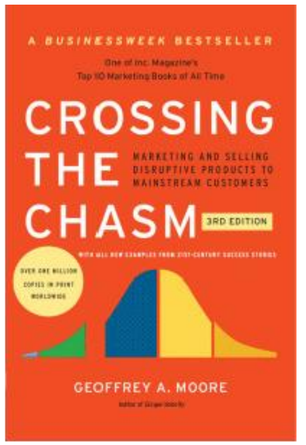
- 특화된 기능들이 필요

- Schedule
  - 요구사항에 따른 최적의 스케줄링 정책
    - Resource Aware, Locality, Usage Based, …
- Network
  - 고정 Inbound/Outbound Endpoint
    - Dynamic L4/L7, NAT Gateway
- Storage
  - 비용과 성능에 적합한 다양한 스토리지 타입 지원
    - Remote SSD, Ceph RDB, Nubes

발표를 마무리 하며…

Crossing the Chasm

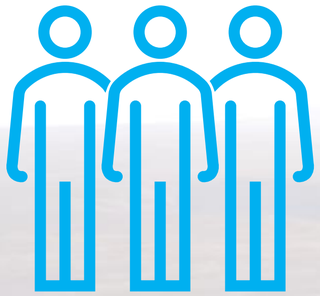




https://www.cncf.io/projects/

Key Point : “Trust”

Reference



얼마나 많은 동료가 사용했는가?

Technology

이 기술 자체는 믿을 만 한가?

Support

이 기술은 잘 지원 받을 수 있을 것인가?

Thank You