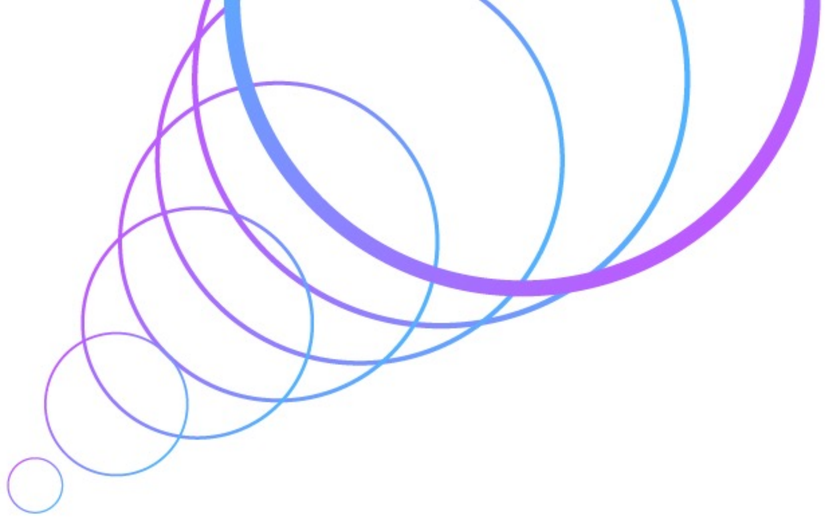
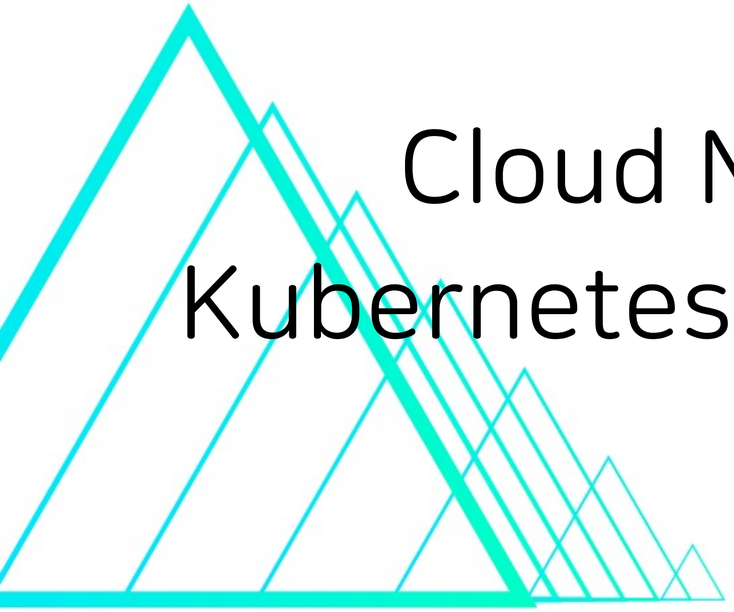
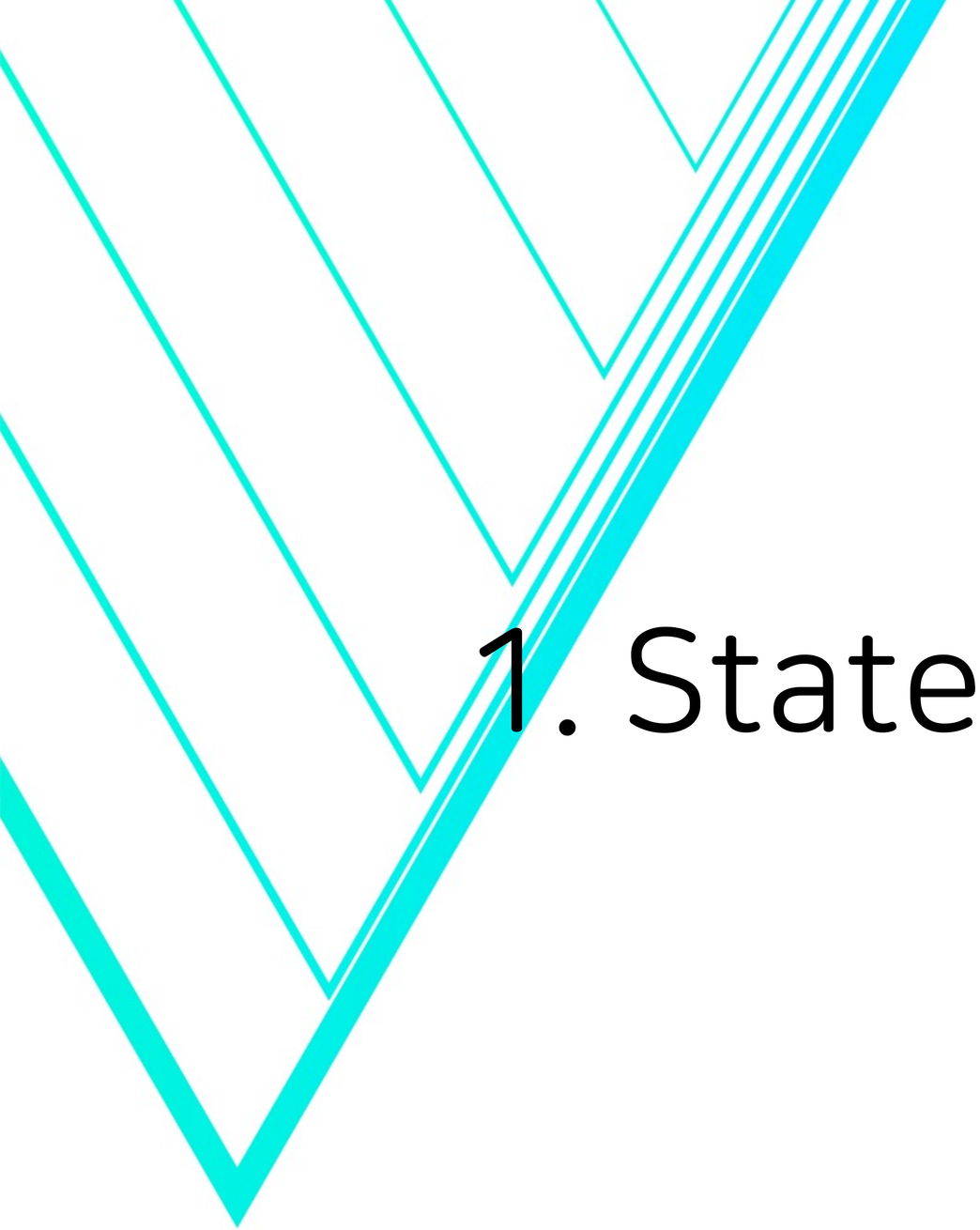




Cloud Native Storage for Kubernetes: Kubernetes Stateful Application데이터 저장 방법

유장선 NAVER

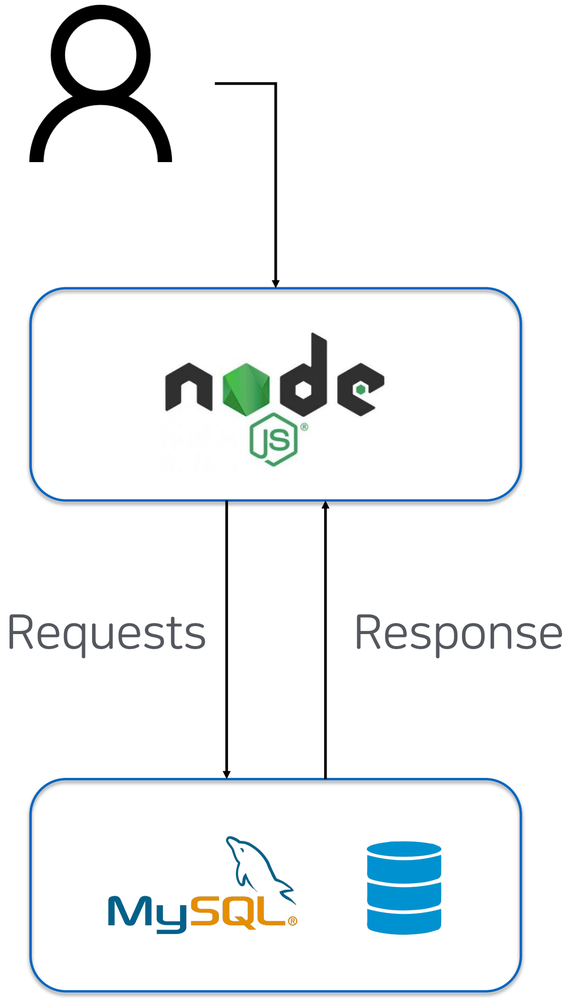
CONTENTS



DEVIEW 2021

1. Stateful Application

1.1 Stateful Application?



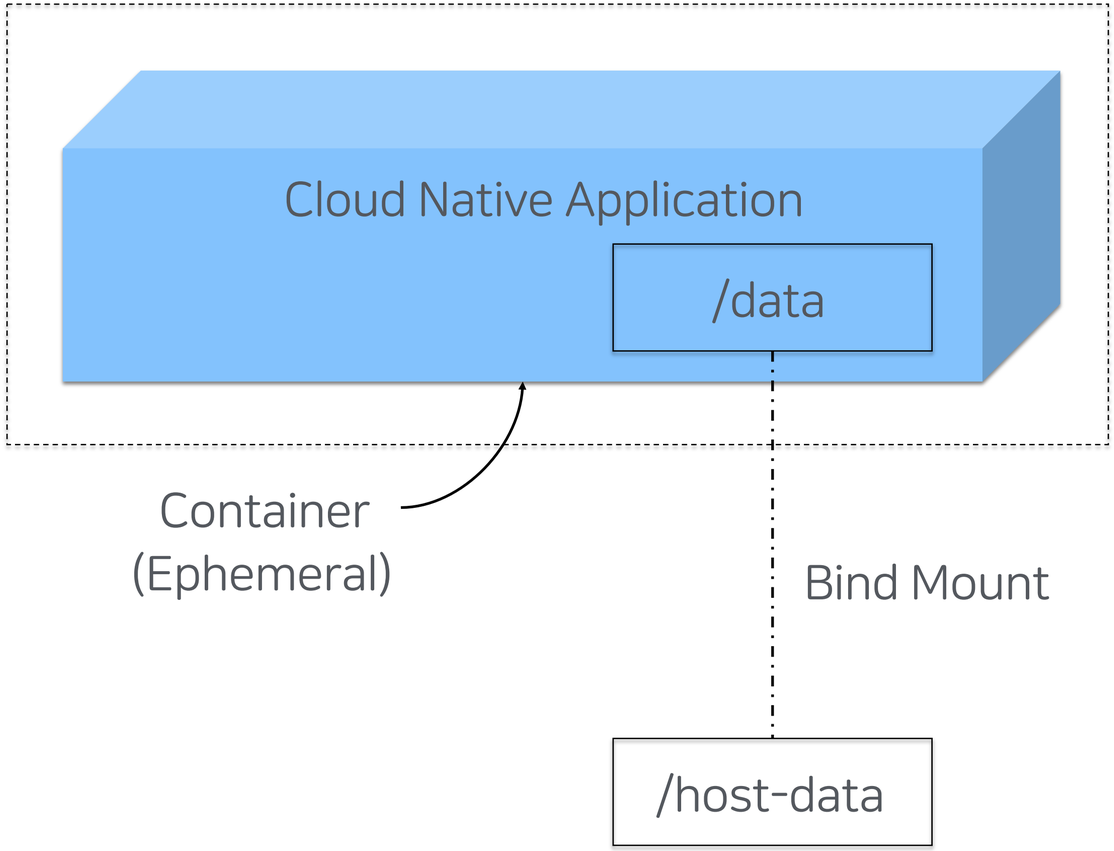
Stateless Application

- Does not require the server to retain information about the state.
- Architecture is simple
- Easy failover to new server
- Easy scale out

Stateful Application

- Requires a server to save information about the state.
- Architecture is complicated
- Hard failover to new server (data loss)
- Hard scale out

1.2 Cloud Native Application



- Designed for a cloud computing architecture
- Use a microservice architecture
- embrace rapid change
- Large scale and resilience

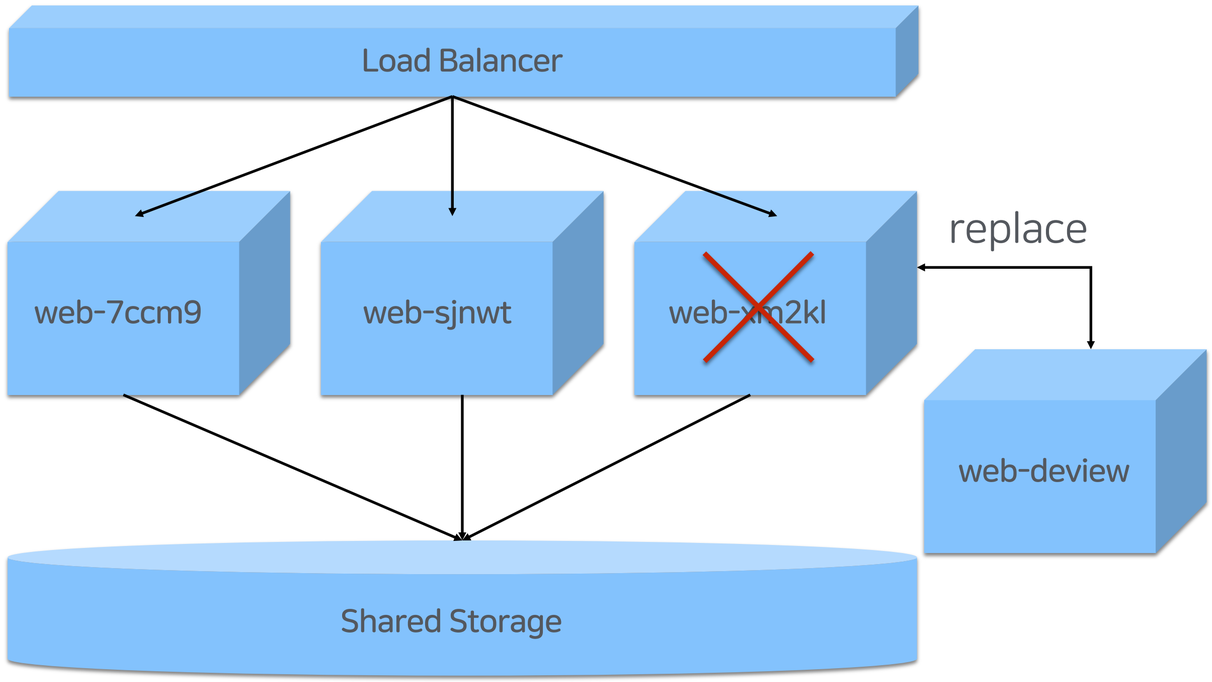
1.3 Cloud Native Application in K8S

DEVIEW 2021

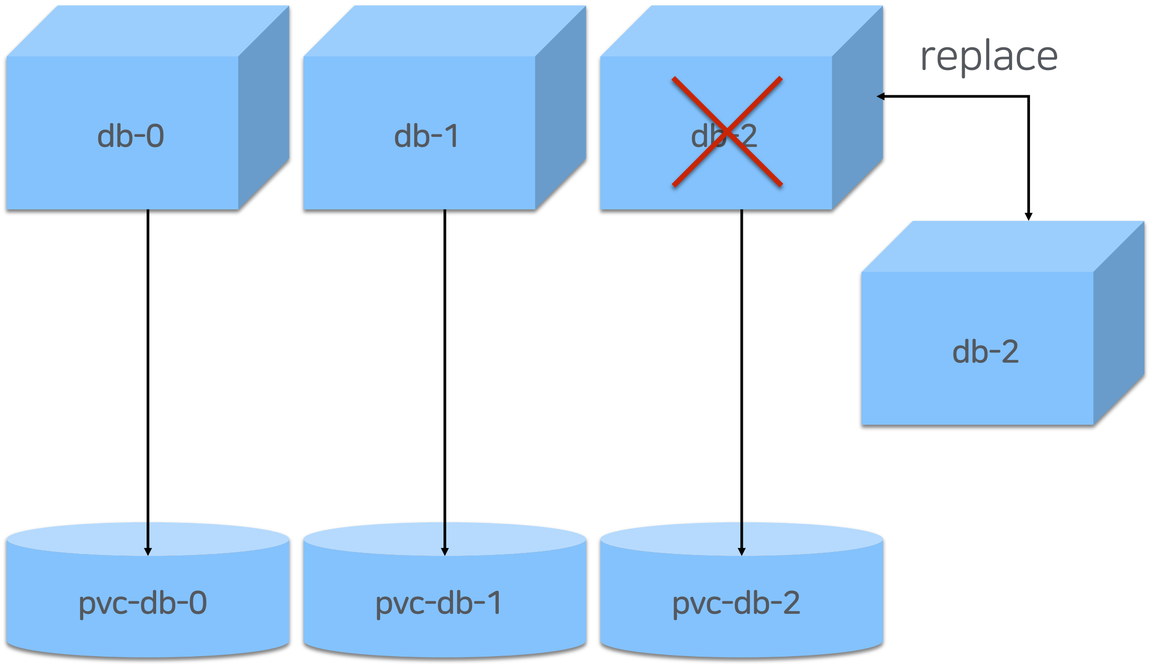
Kubernetes에서는 deployment, statefulset을 제공하고 있으며,

Stateless application은 deployment를, Stateful application은 statefulset을 사용 할 수 있음

Deployment



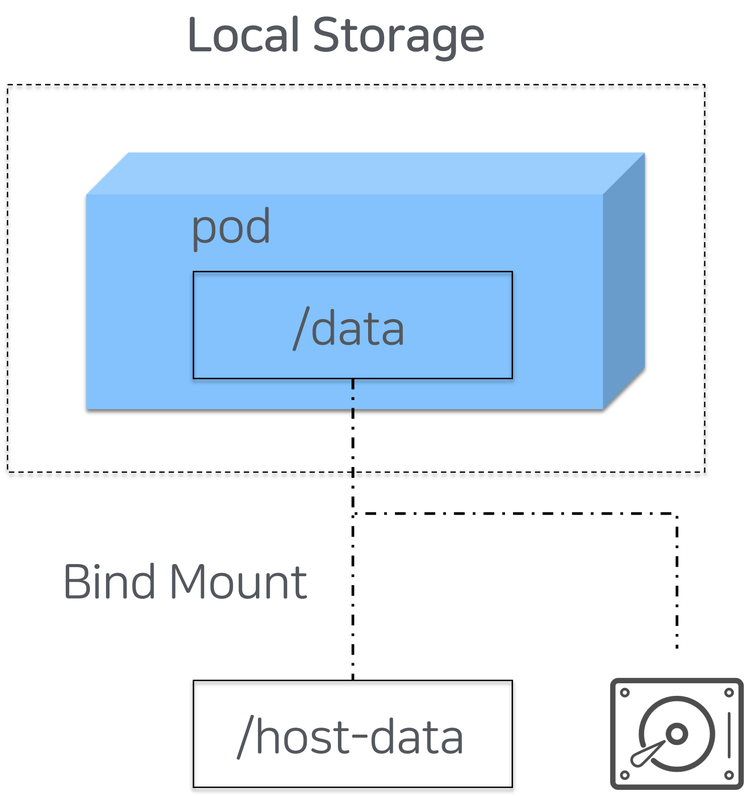
Statefulset



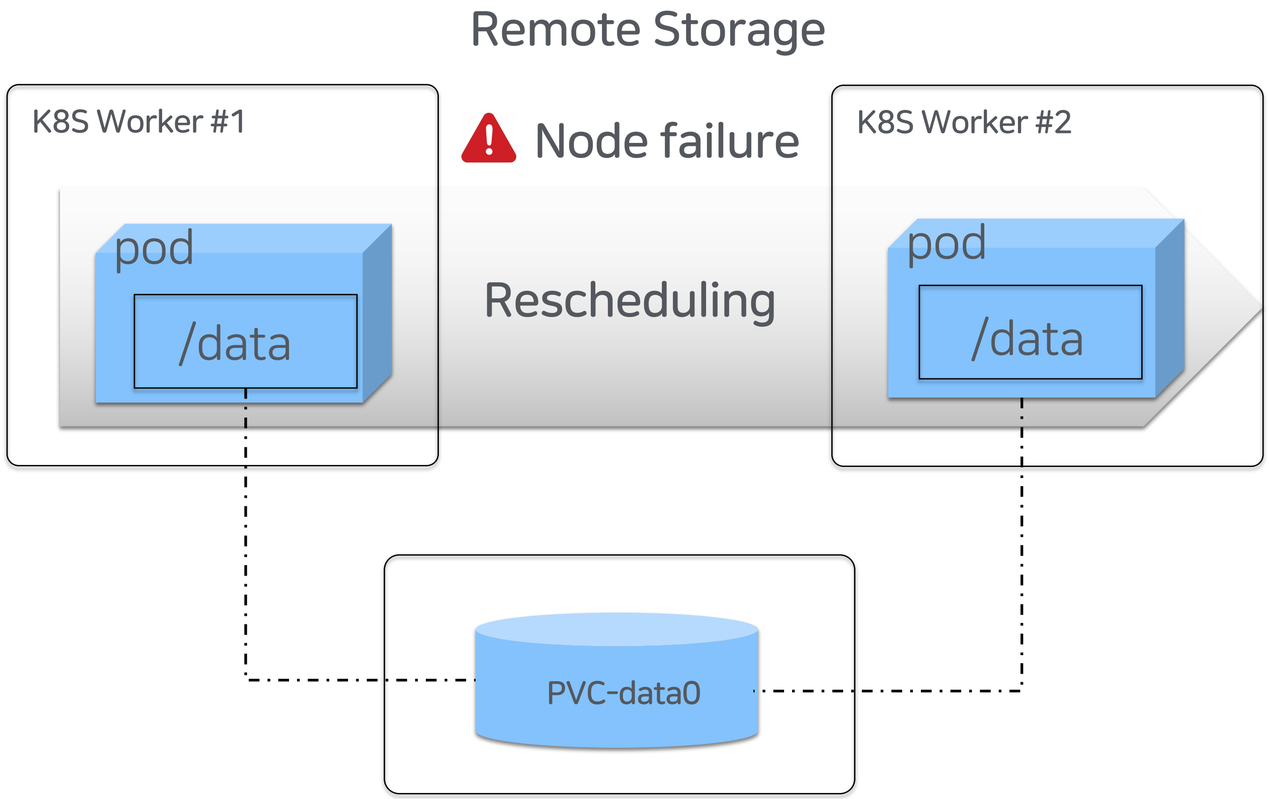
1.4 Local vs Remote Storage

Local Storage는 빠른 IO처리가 가능하지만, Worker 노드 장애 시 데이터가 삭제되는 문제가 발생함

Remote Storage는 데이터를 외부 노드에 저장하므로, Worker 노드 장애 시에도 Pod재할당 후 서비스 가능



노드 장애 시 데이터 삭제

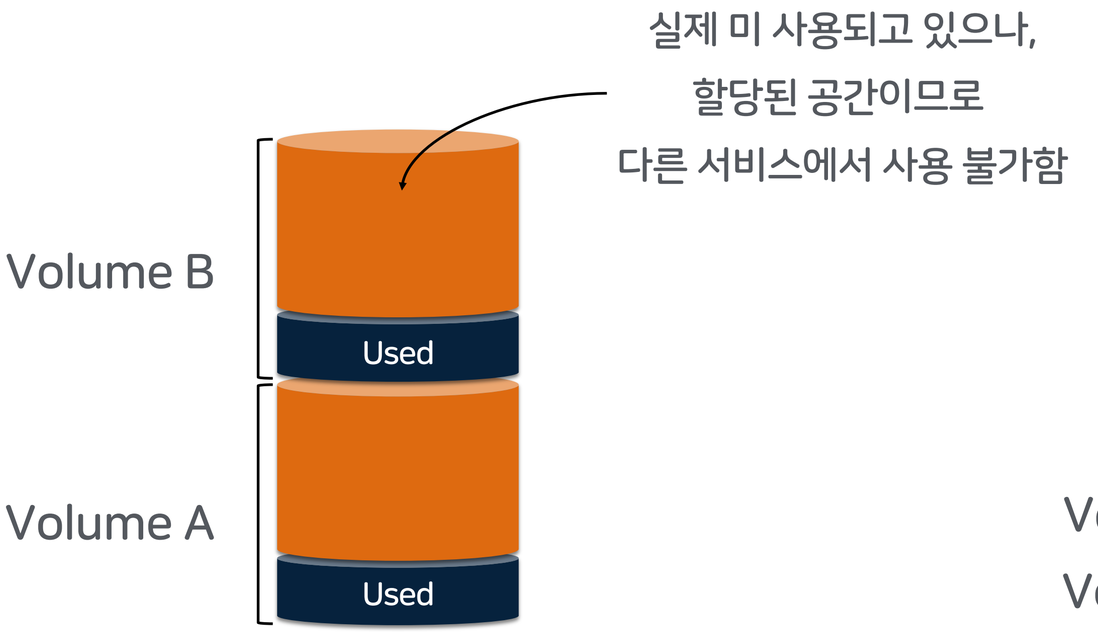


Remote Storage사용시 Pod재할당 후 서비스 재개됨

1.5 Thick vs Thin Provisioning

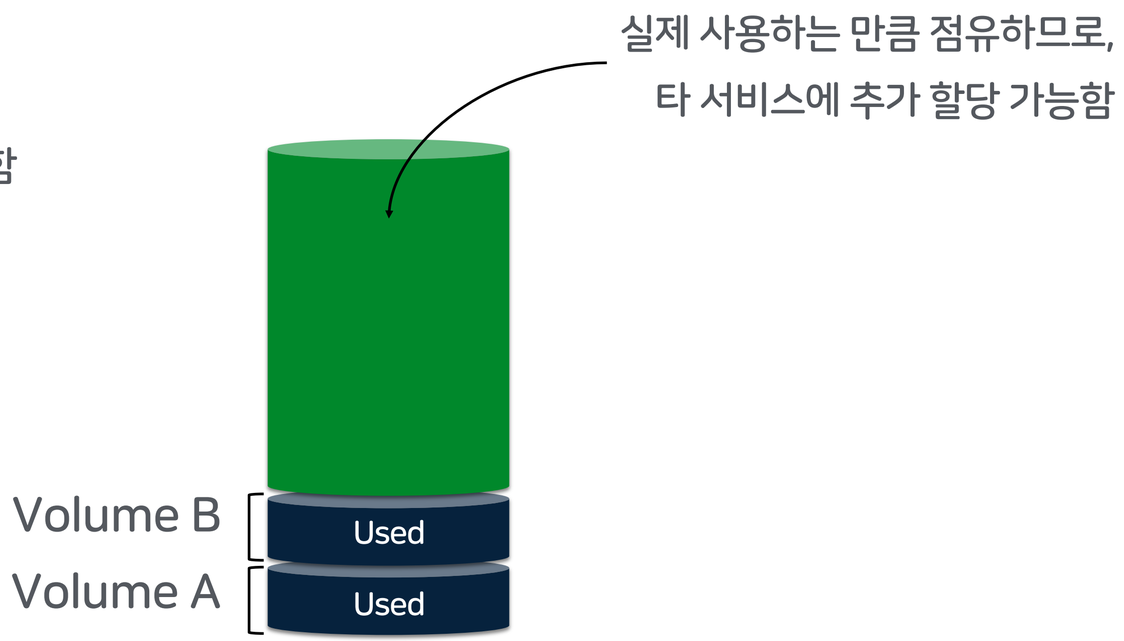
Local Disk는 용량 부족 시 증설이 불가능하므로, 최대 용량만큼까지 할당이 가능함

Remote Storage는 용량 부족 시 클러스터 증설이 가능하므로 Over Provisioning이 가능함



Thick

(Local Disk)



Thin

(Remote Storage)

Stateful Application서비스를 위해

Remote Storage / Thin Provisioning방식을 이용한,

Cloud Native Storage를 사용하고 있습니다.

DEVIEW 2021

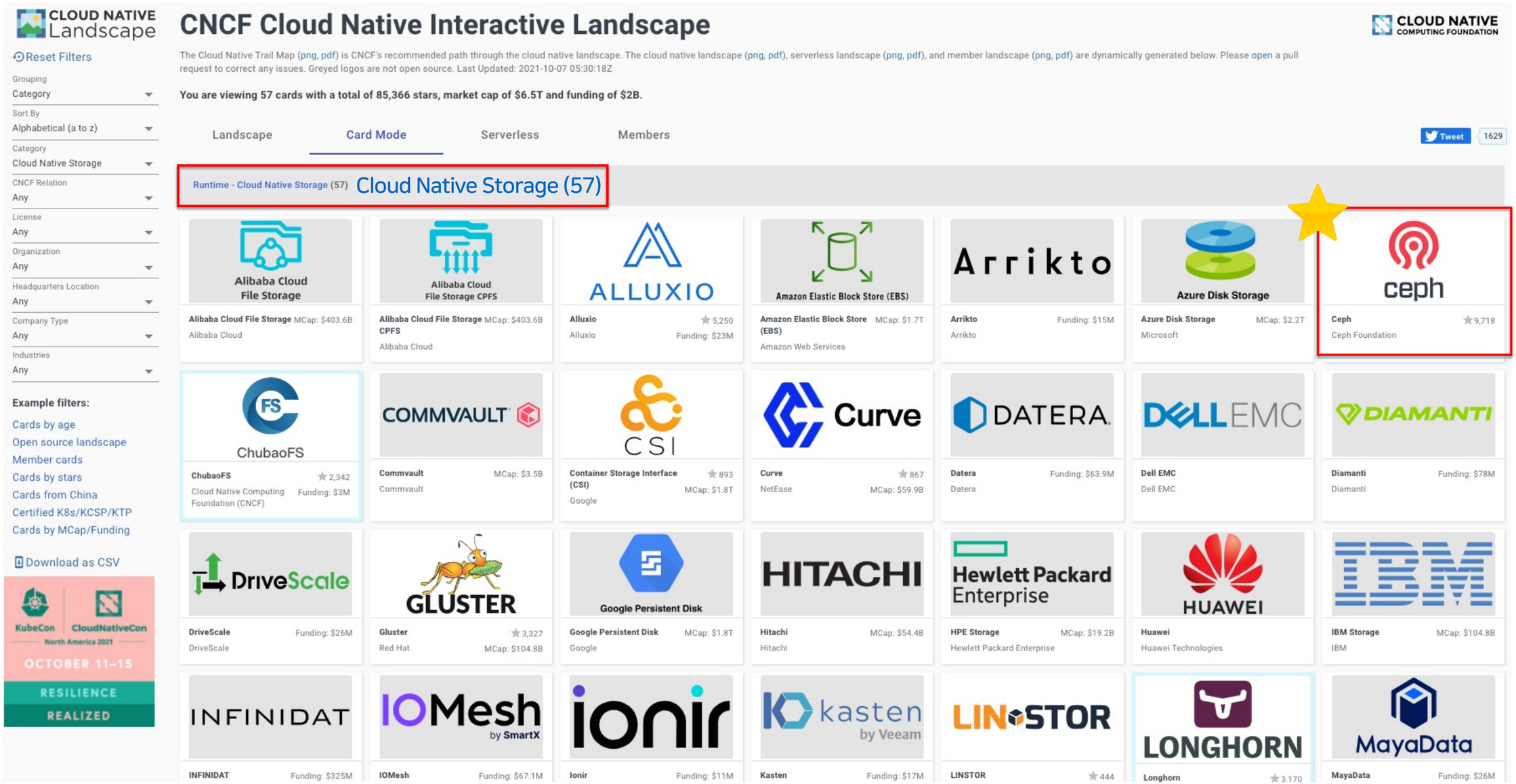
2. Cloud Native Storage

2.1 Cloud Native

DEVIEW 2021

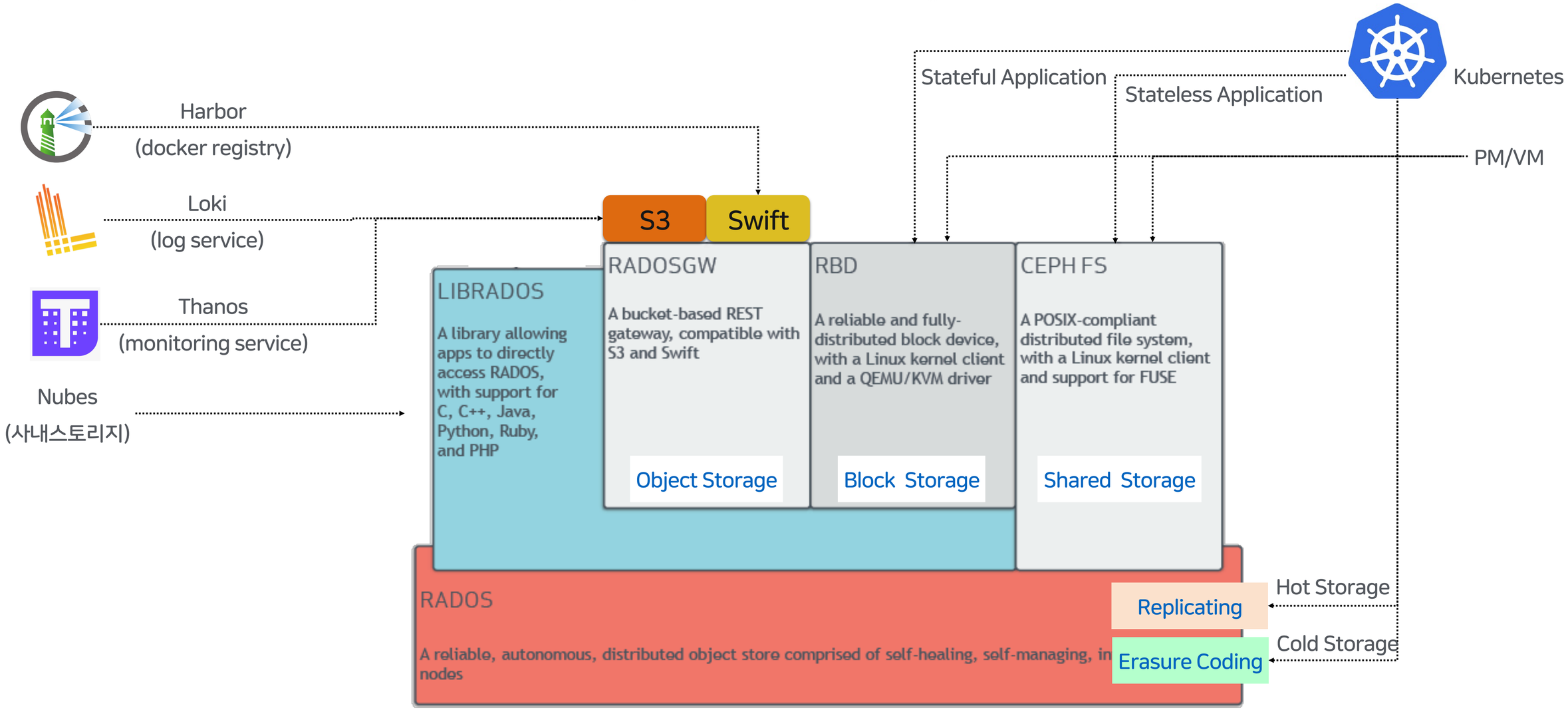
- Horizontally scalable
- No single point of failure
- Resilient and survivable
- Minimal operator overhead
- Decoupled from the underlying platform

2.2 Cloud Native Storage



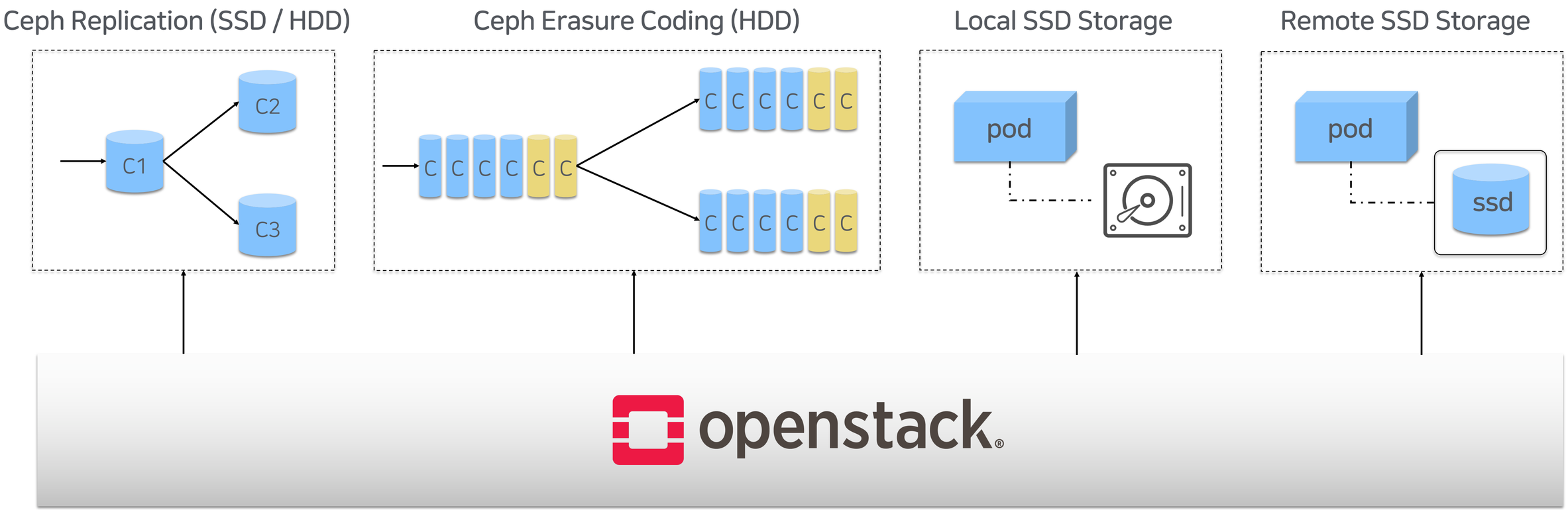
2.3 Ceph Storage

Ceph는 오픈소스 스토리지이며, 하나의 스토리지를 통해 Object, Block, Shared Storage를 모두 제공합니다.



2.4 Multiple Storage Type

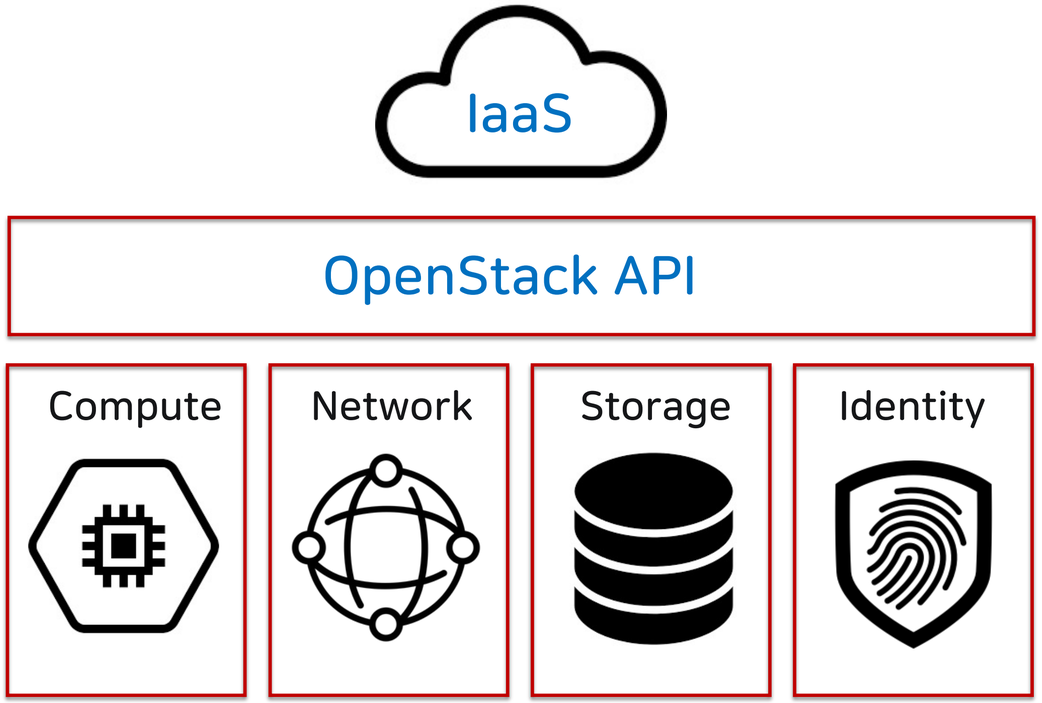
다양한 워크로드를 지원하기 위하여 NVMe/SSD/HDD디바이스로 스토리지를 구성하고,
Replicating / Erasure Coding 구성 및 Local / Remote SSD등을 제공하고 있습니다.



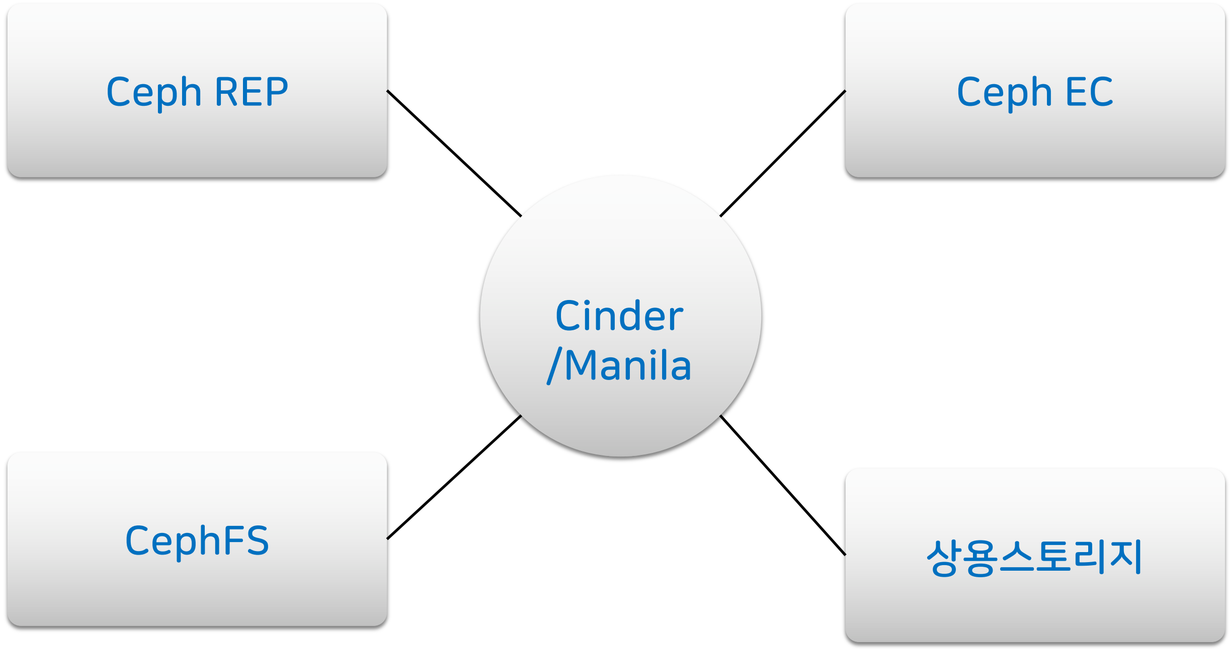
2.5 OpenStack

오픈스택은 클라우드 컴퓨팅 오픈소스 프로젝트이며, 다양한 하위 프로젝트로 구성되어 있습니다.

Keystone (인증) / Cinder (블록스토리지) / Manila (공유스토리지)를 도입하여 멀티 클러스터환경을 지원중입니다.



IaaS OpenSource Computing Platform



OpenStack을 통하여 하위 스토리지들을 볼륨타입으로 관리

2.6 Kubernetes Volume Plugin

DEVIEW 2021

Kubernetes는 다양한 스토리지 Volume Plugin을 내장하고 있습니다.
저희는 사내 시스템 및 인증 연동, 멀티 테넌시, 운영 자동화 기능 등을 사용하기 위해 직접 개발하여 사용 중입니다.

볼륨 플러그인	내부 프로비저너	설정 예시
AWSElasticBlockStore	✓	AWS EBS
AzureFile	✓	Azure 파일
AzureDisk	✓	Azure 디스크
CephFS	-	-
Cinder	✓	OpenStack Cinder
FC	-	-
FlexVolume	-	-
Flocker	✓	-
GCEPersistentDisk	✓	GCE PD
Glusterfs	✓	Glusterfs
iSCSI	-	-
Quobyte	✓	Quobyte
NFS	-	NFS
RBD	✓	Ceph RBD
VsphereVolume	✓	vSphere
PortworxVolume	✓	Portworx 볼륨
ScaleIO	✓	ScaleIO
StorageOS	✓	StorageOS
Local	-	Local

2.7 Kubernetes Custom Volume Plugin

RBD, CephFS, iSCSI(Remote SSD) 지원을 위한 Provisioner (dynamic volume provisioning)와 Driver를 개발함
Storage Watcher를 통해 주기적으로 운영 작업을 수행함

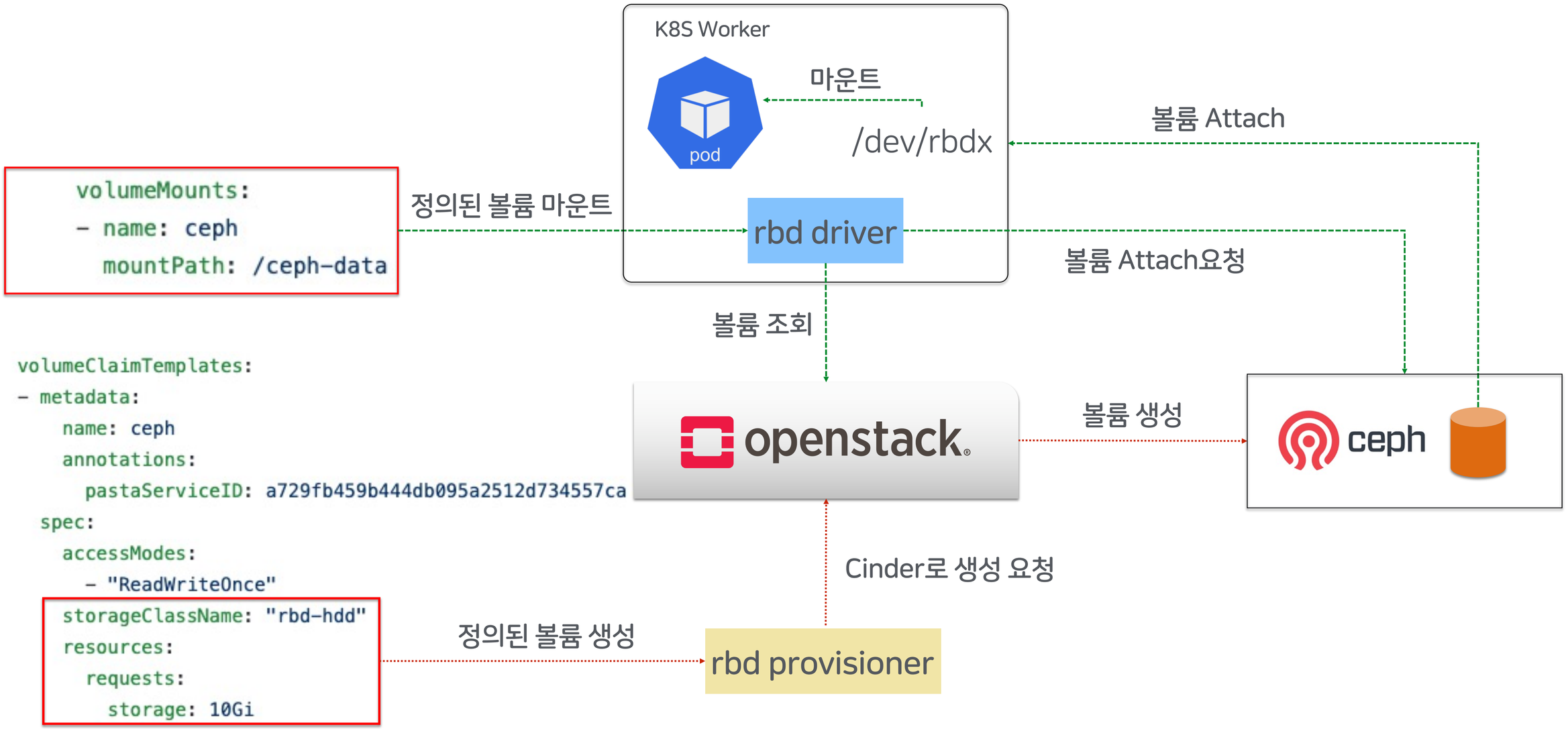
- Provisioner (RBD, CephFS, iSCSI)
- Driver (RBD, CephFS, iSCSI)
- Storage Watcher (rate limiting, orphan, reclaim, usage)

- Identity / Multi-tenancy (keystone)
- Multiple Block Storage Backend (Cinder)
- Multiple Shared Storage Backend (Manila)

- Block Storage (Ceph RBD, Local, iSCSI)
- Shared Storage (CephFS)
- Object Storage (swift, s3)

2.8 Flow

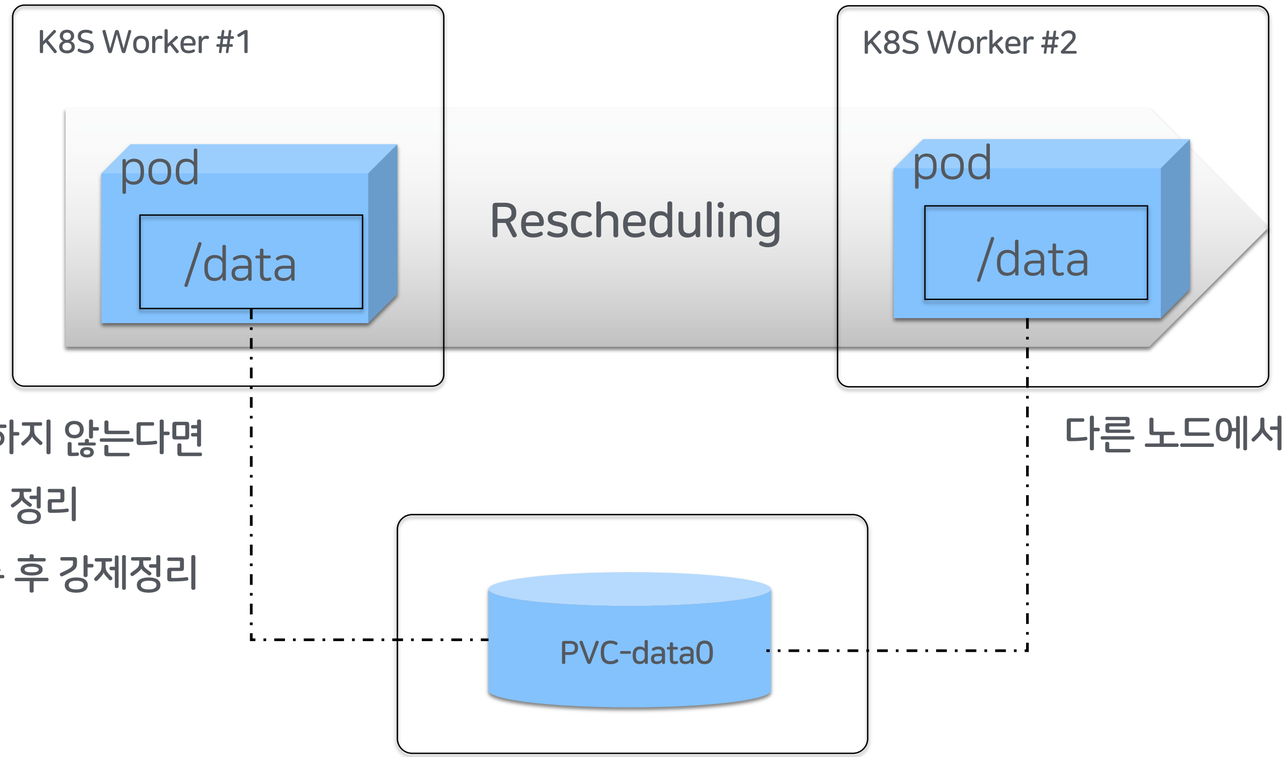
Statefulset yaml정의를 통해 볼륨을 신규 생성하고, 이를 POD에 자동으로 마운트해주는 작업을 진행함



3.TroubleShooting

3.1 Multi-attach / Orphan volume

두개 노드에서 하나의 볼륨을 동시에 사용 시 파일시스템 corruption이 발생 할 수 있습니다.
멀티어태치 및 오펀드 볼륨 관리를 통하여 Rescheduling이 원활하게 동작되도록 지원합니다.

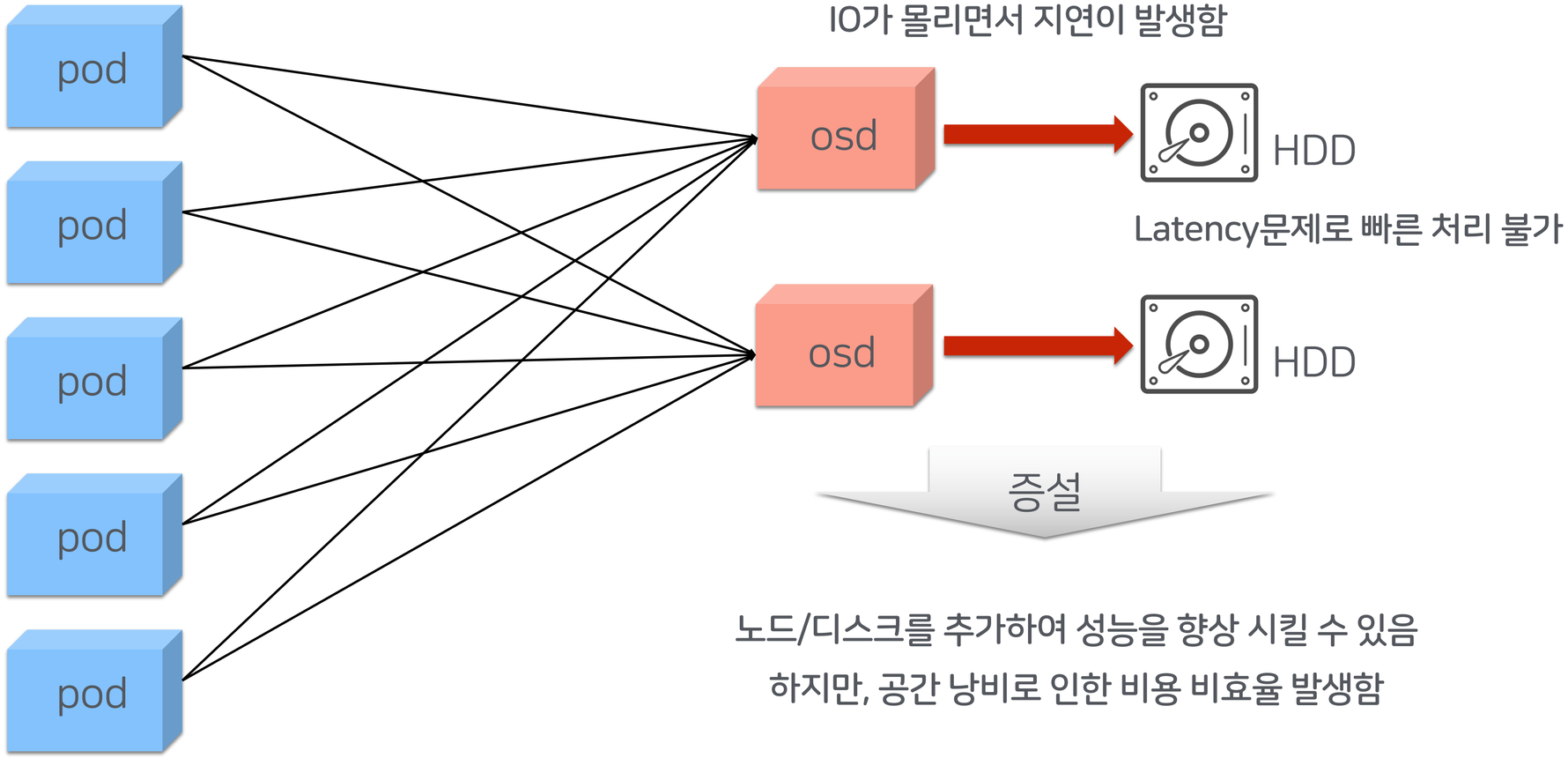


볼륨이 Attach상태지만, Pod이 존재하지 않는다면
Orphan volume판단 후 자동 정리
Umount불가 시 ceph blacklist 등록 후 강제정리

다른 노드에서 사용 중인지 확인이 필요함

3.2 (HDD) 클러스터 성능 이슈

수 십개 Pod로 구성된 서비스에서 새벽에 백업이 동작하여 IO 가 몰리는 경우,
HDD Latency이슈로 인해 서비스 IO 영향이 발생하게됨



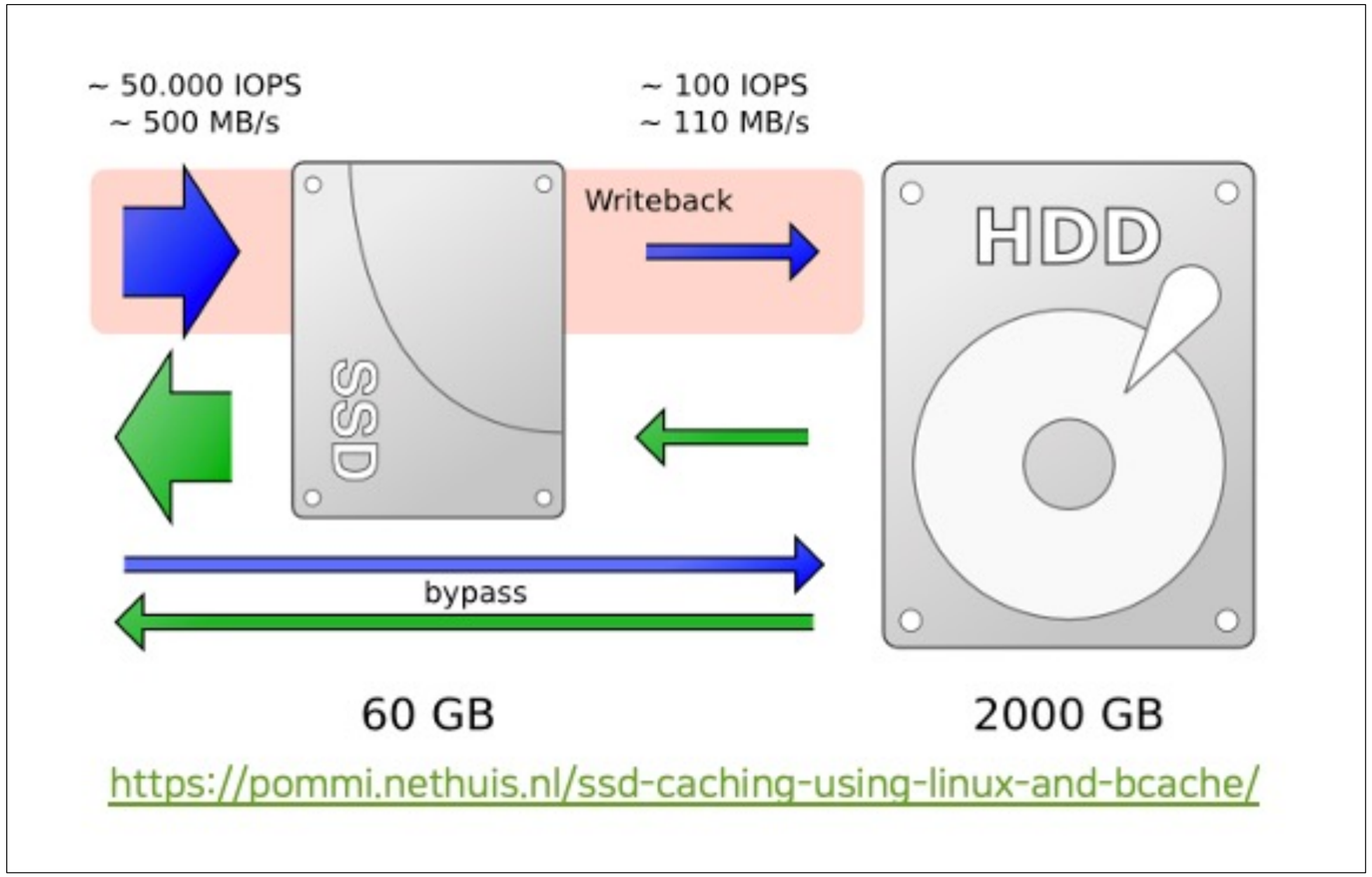
노드/디스크를 추가하여 성능을 향상 시킬 수 있음
하지만, 공간 낭비로 인한 비용 비효율 발생함

3.2 (HDD) 클러스터 성능 이슈

Block Cache는 느린 HDD 디스크의 성능 개선을 위해 앞 단에 빠른 디스크를 두고,
IO를 먼저 처리한 후 HDD로 천천히 write하는 기술입니다.

Random write는 빠른 디스크 처리

Sequential write는 Random Cache를
제거할 수 있기 때문에 bypass 시킴



3.2 (HDD) 클러스터 성능 이슈

Bcache는 kernel 3.10 에 포함되어 있어서, 생성 툴(bcache-tools) 빌드 후 사용 가능함

구성은 쉬운 편이지만, 디스크가 많다면 복잡도는 증가됨

- Kernel Module Enable

$ modprobe bcache

- Install bcache-tools

$ git clone https://evilpiepirate.org/git/bcache-tools.git

$ cd bcache-tools

$ make

$ make install

- Create bcache

$ make-bcache -C /dev/nvme0n1p18 -B /dev/sdh2 –writeback

$ lsblk -o NAME,MAJ:MIN,RM,SIZE,TYPE,FSTYPE,MOUNTPOINT,UUID,PARTUUID | grep bcache

__nvme0n1p18 259:18 0 99G part bcache 4190a4e9-5825-4e2e-8a91-026447b352da 07c25f99-6d71-494b-a044-618d10efa959

_ __bcache0 252:0 0 5.5T disk

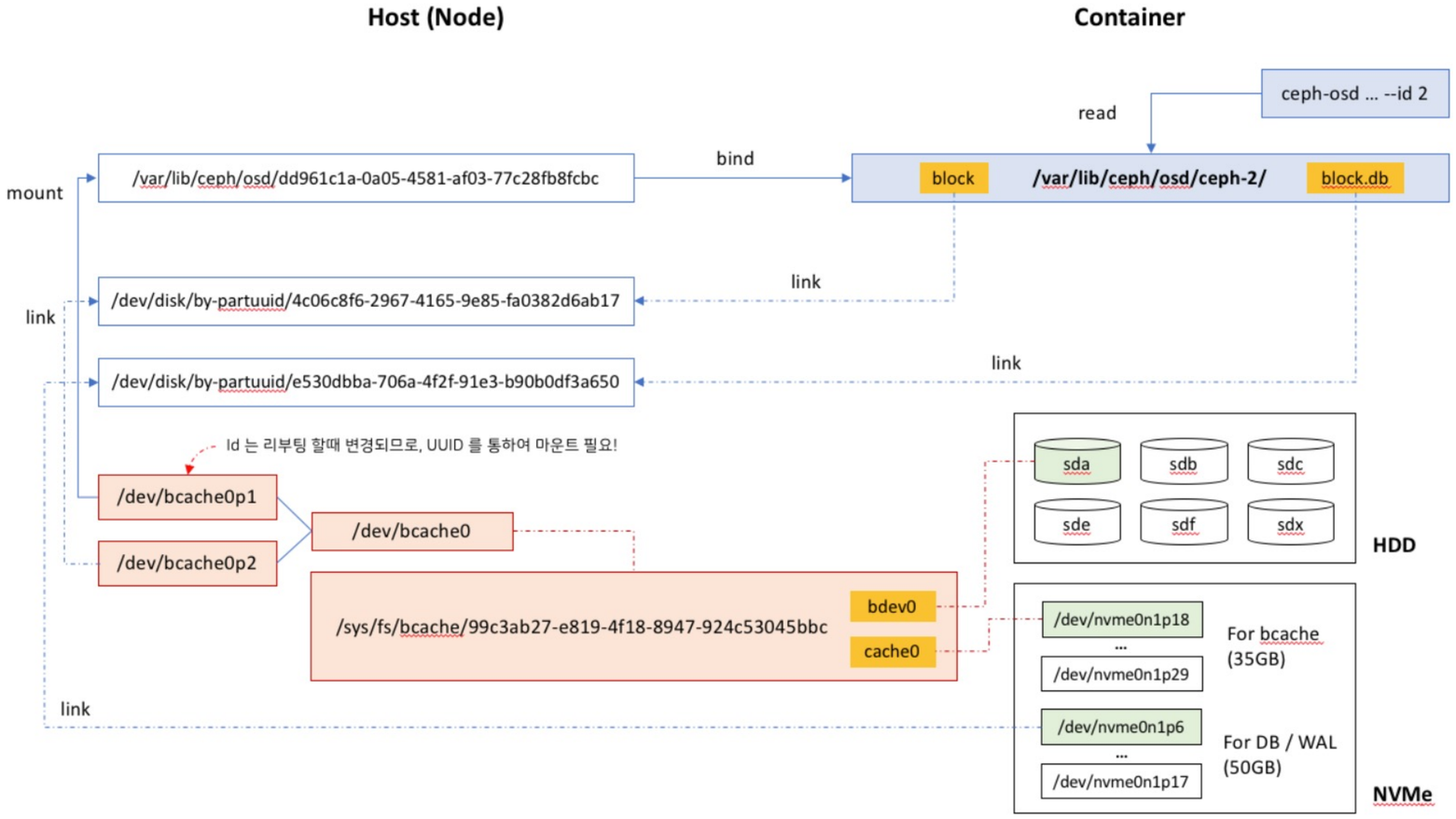
__sdh2 8:114 0 5.5T part bcache fe715aac-f186-43e9-98e7-11bb0eebefa3 e65062e6-05a4-4d10-b718-15928a008f31

__bcache0 252:0 0 5.5T disk

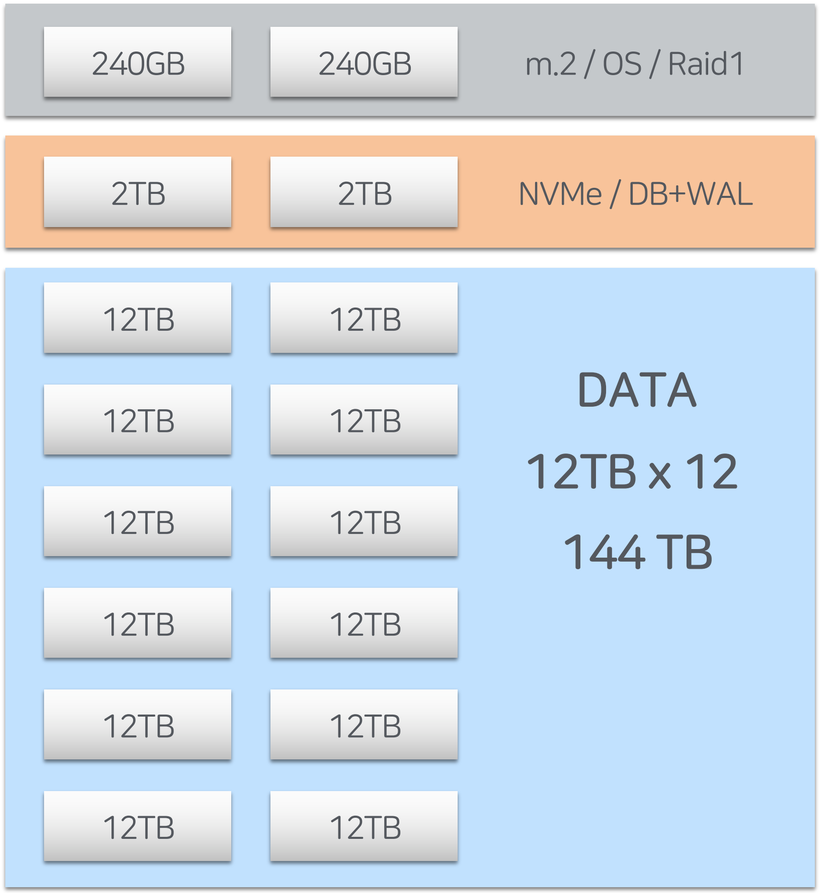
3.2 (HDD) 클러스터 성능 이슈

DEVIEW 2021

1개 디스크를 bcache로 구성 시 모습

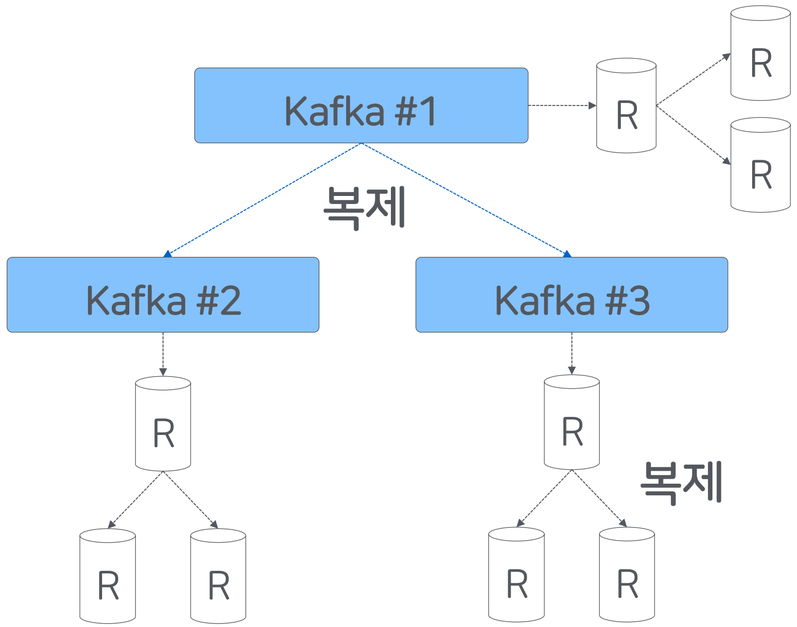


12TB x 12 + m.2 x 2 + NVMe x 2



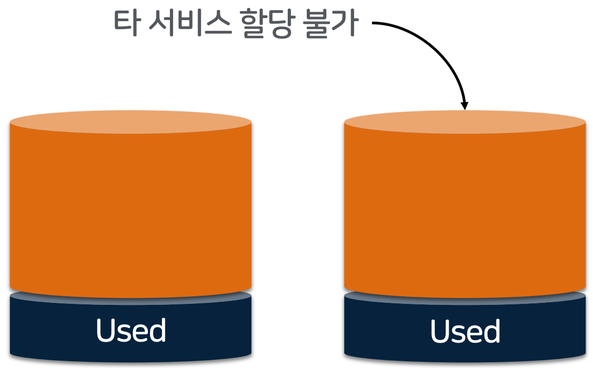
3.3 분산 서비스 on 분산 스토리지 이슈

분산 서비스는 자체 복제방식으로 데이터를 저장하고 있고,
분산 서비스에서 분산 스토리지 사용 시 많은 복제가 발생하게 됩니다.
로컬 디스크는 제한적인 수량과 미 사용시 비용 비효율이 발생합니다.



3 copy * 3 copy -> 9 copy 이슈

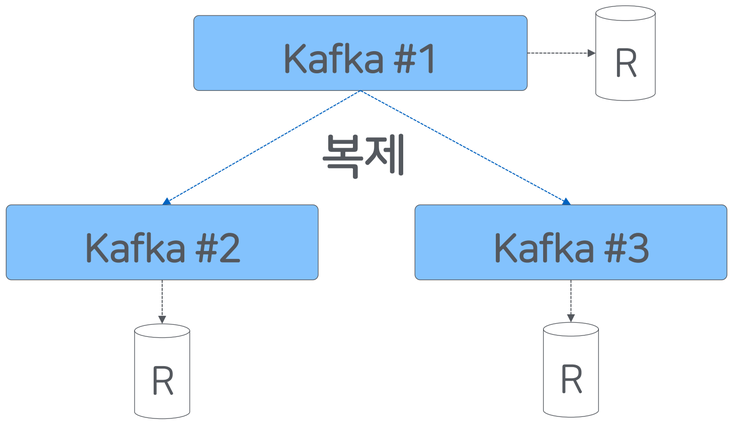
제한적인 로컬 디스크 수



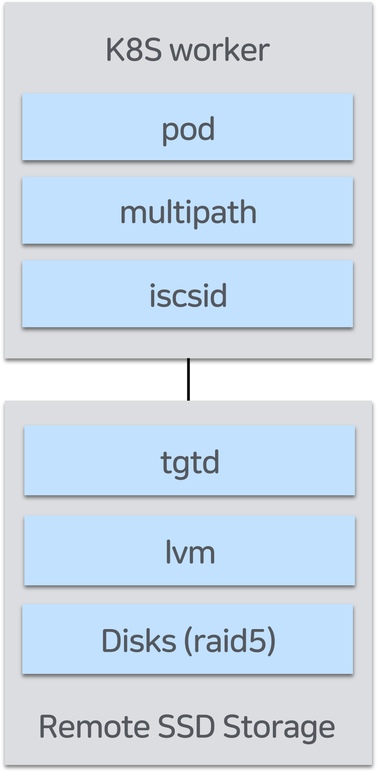
미사용 시 비용 비효율

3.3 분산 서비스 on 분산 스토리지 이슈

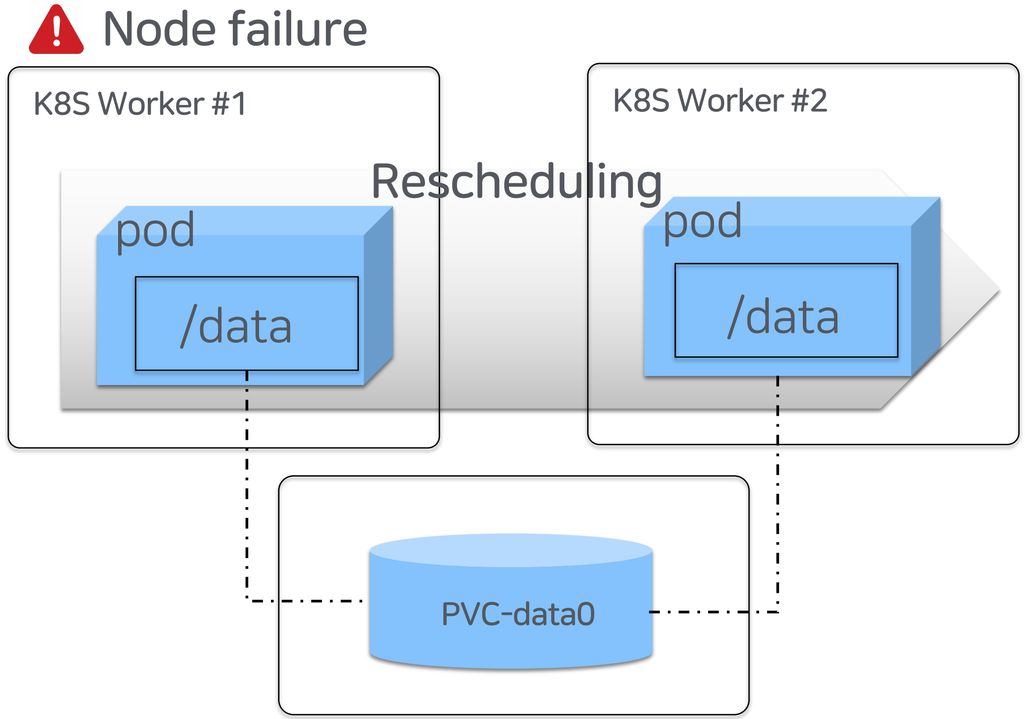
원본만 저장하는 (1 copy) Remote SSD Storage를 제공하여 불필요한 공간 낭비 제거,
Worker Node장애 시 데이터 보존되며, 바로 서비스 재개 가능



1 copy : 추가 복제 없음



필요한 만큼 할당 (낭비 최소화)

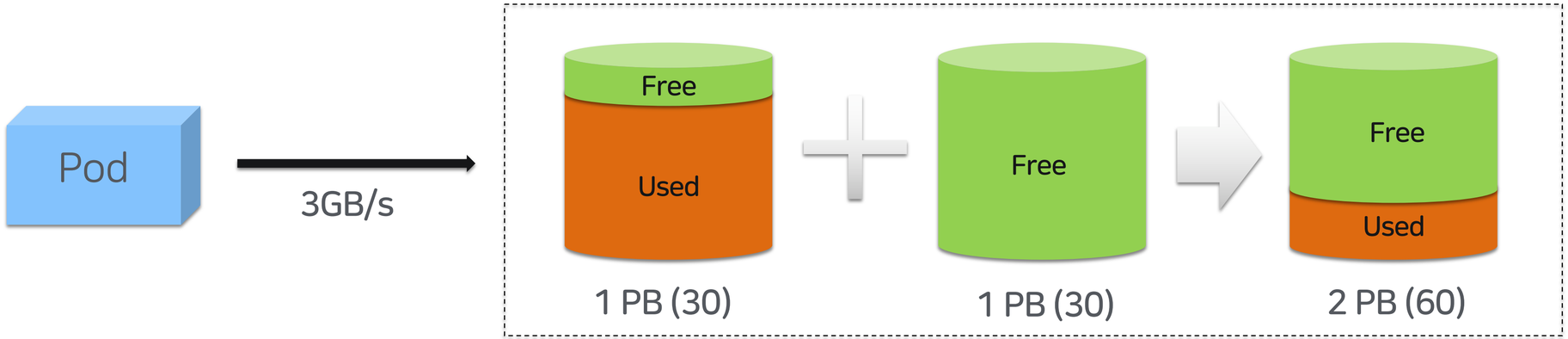


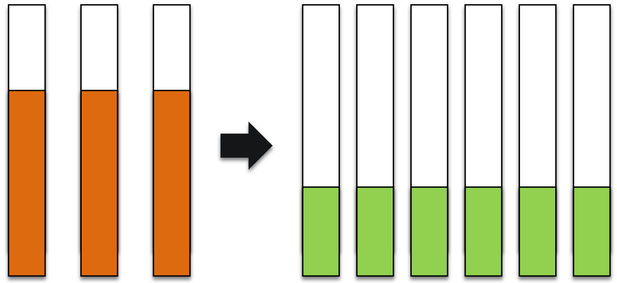
노드 장애 시 데이터는 보존되며,
Rescheduling시 서비스 재개

3.4 스토리지 증설의 현실적인 문제점

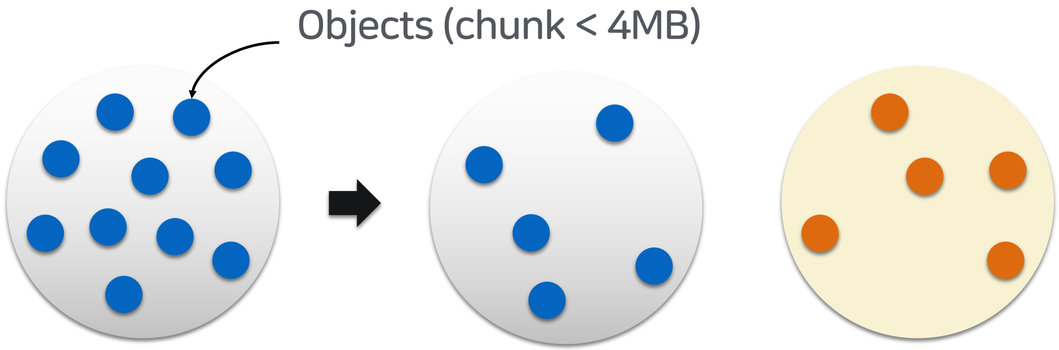
서비스 중에도 증설이 가능합니다.

증설은 노드 증설과 PG (Placement Group) 증가 작업으로 구분됩니다.





노드 증설 (Reallocation)



PG Split (2048 -> 4096)

3.4 스토리지 증설의 현실적인 문제점

증설은 서비스 영향도 증가, 오랜 증설에 따른 운영 리소스 증가, 잦은 재배치로인한 디스크 마모도 증가 문제가 발생함

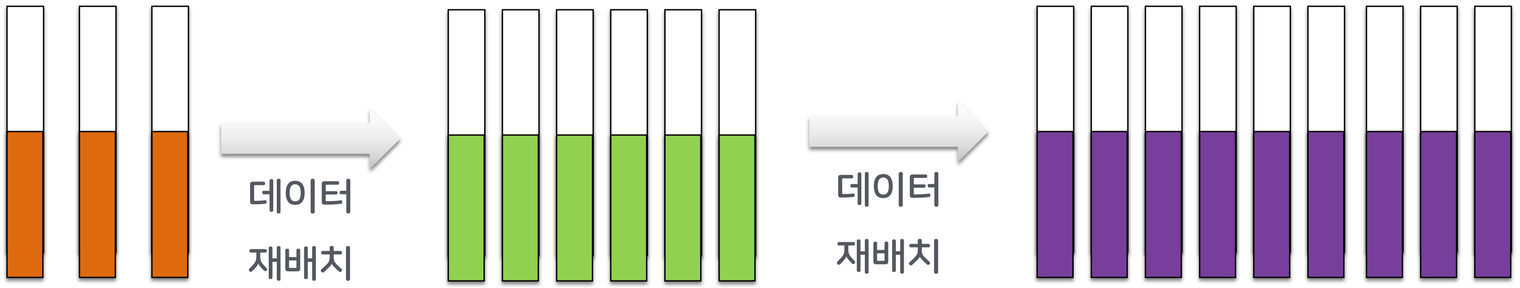
긴 증설 시간
(수개월의 운영 작업)

- 최저 속도로 진행 (osd_max_backfills 1)
- 사용량 높은 낮 시간 중지 (밤에만 진행)

→

- 수 개월 이상 증설 작업 필요
- 쌓이는 데이터 > 확보되는 공간

디스크 마모도 증가

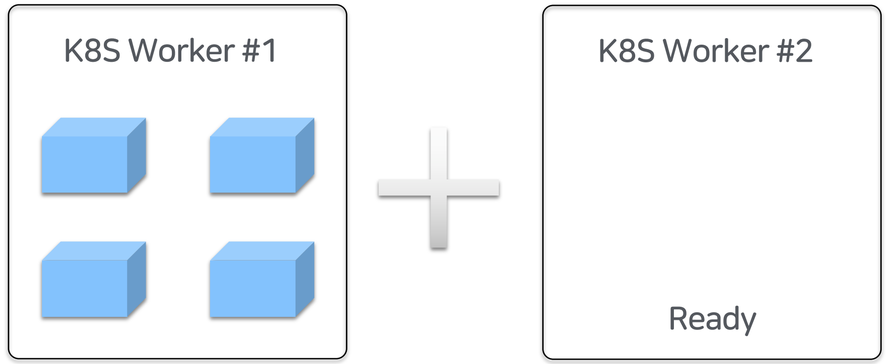


잦은 데이터 재배치 작업으로 read/write 증가에 따른 디스크 마모도 증가

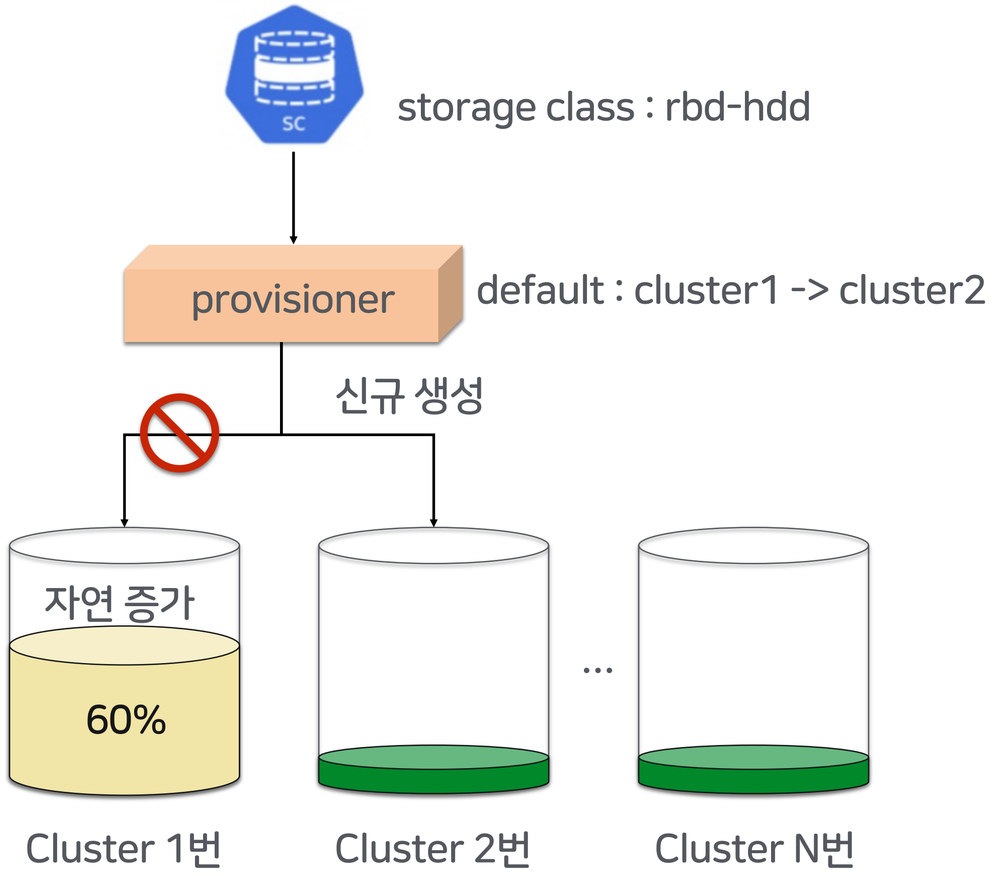
3.4 스토리지 증설의 현실적인 문제점

Compute Node처럼 투입 시 바로 사용 가능한 구조를 Storage 에도 적용함

Provisioner를 통해 볼륨생성 클러스터를 지정하는 방식으로 재배치 없이 증설이 가능한 구조로 변경됨



노드 부족 시 신규 노드 투입(신규 노드는 바로 서비스 가능 상태)



DEVIEW 2021

4. Operation Tools and Tips

4.1 클러스터 현황

클러스터

40

노드

800

디스크

10,000

구축 용량

37 PiB

볼륨

60,000

IO

25GB/s

4.2 Performance Monitoring

모든 이벤트는 사내 메신저를 통해 전달받고 있으며, 중요 알람 확인을 위해 레벨 단위로 처리

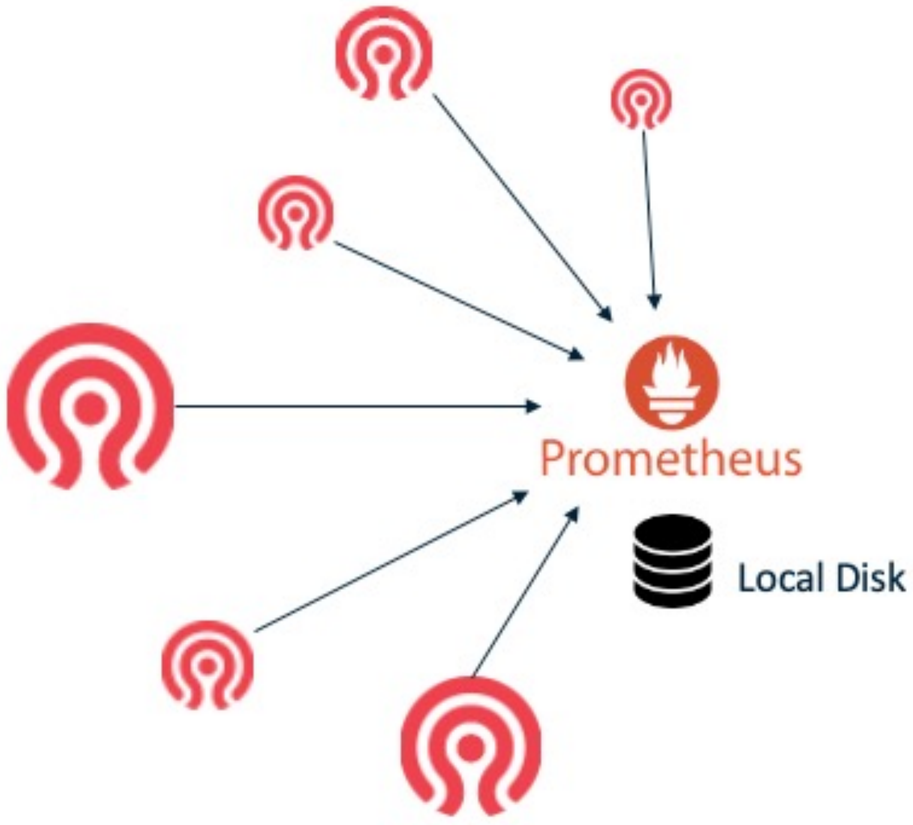
Prometheus – Grafana를 활용하여 모니터링 중이며, 다수 Exporter를 개발하여 사용

사내 메신저 연동을 위해 Grafana webhook서버를 개발하여 운영

4.2 Performance Monitoring

서비스 초기 중앙의 프로메테우스 서버로 모든 클러스터의 성능 데이터를 수집

메트릭이 많아지면서 중앙 수집 구조의 문제점들이 발생하기 시작함

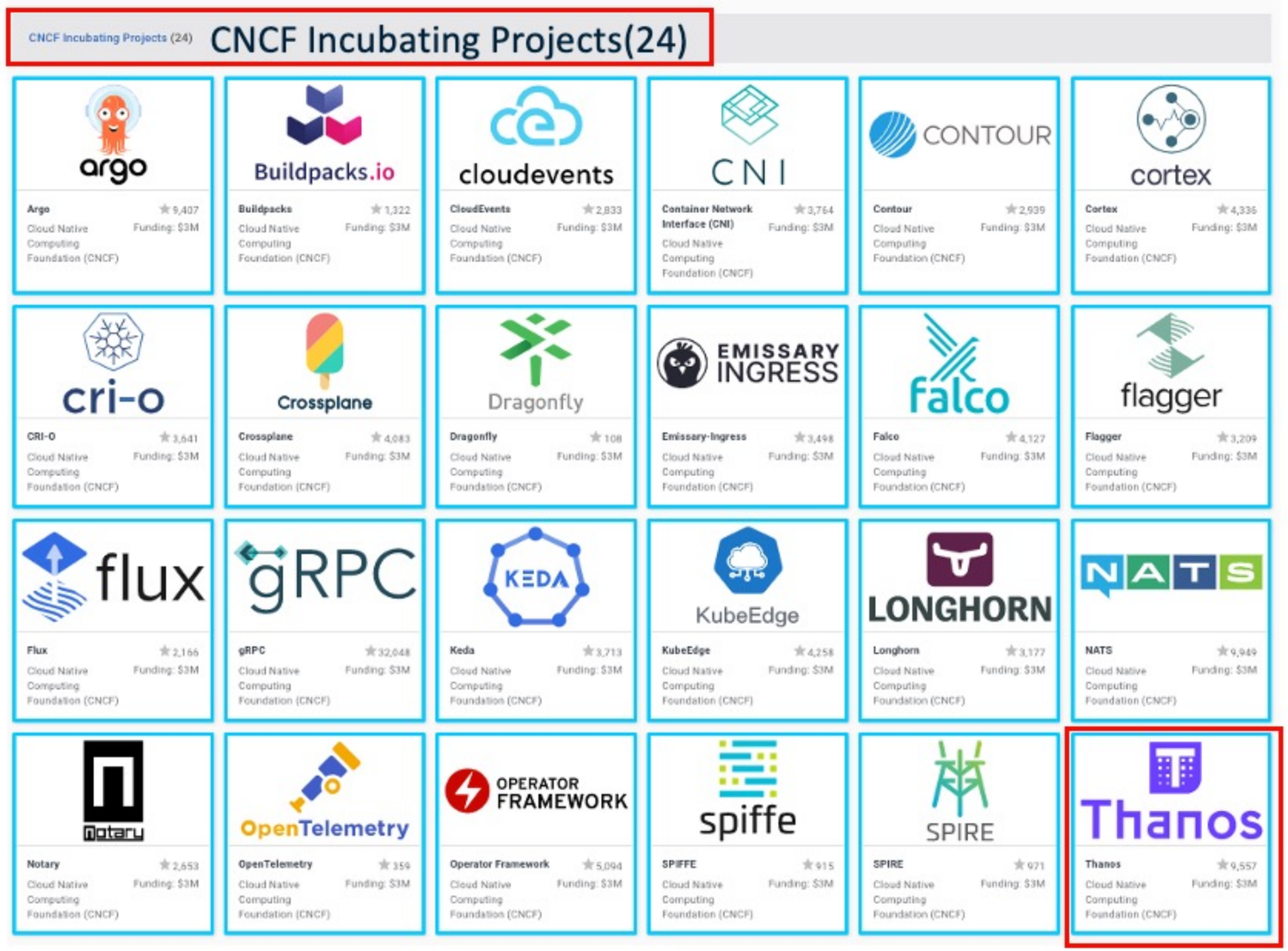


확장성 문제
- 로컬 디스크 부족 시 대응이 어려움

가용성 문제
- HA 지원되지 않음 -> 2개 기동

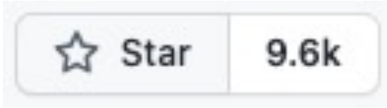
4.2 Performance Monitoring

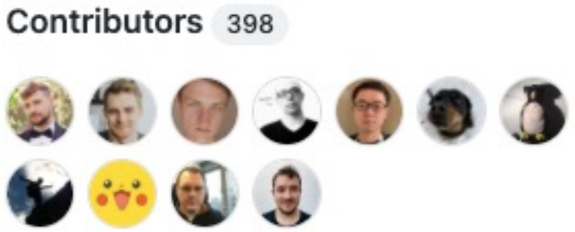
프로메테우스의 문제점을 해결하기 위하여 Thanos를 도입함



github.com/thanos-io/thanos

https://thanos.io/

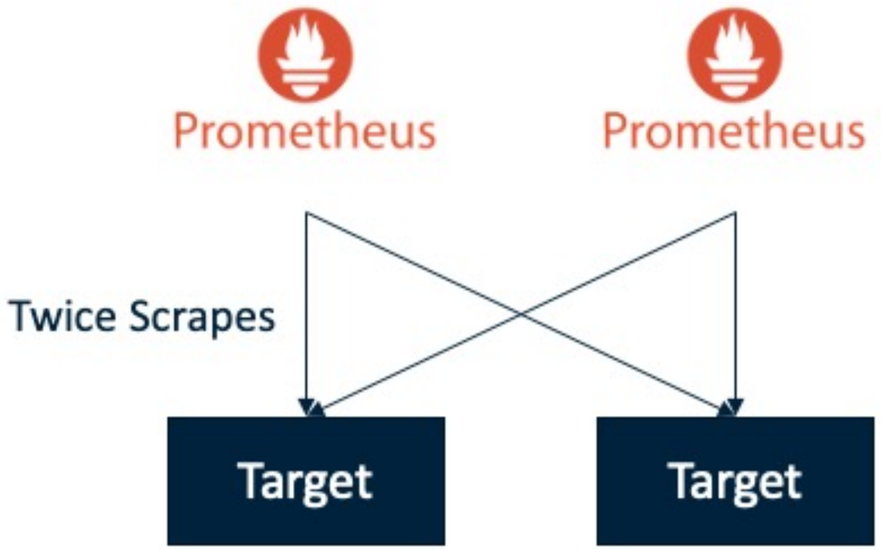




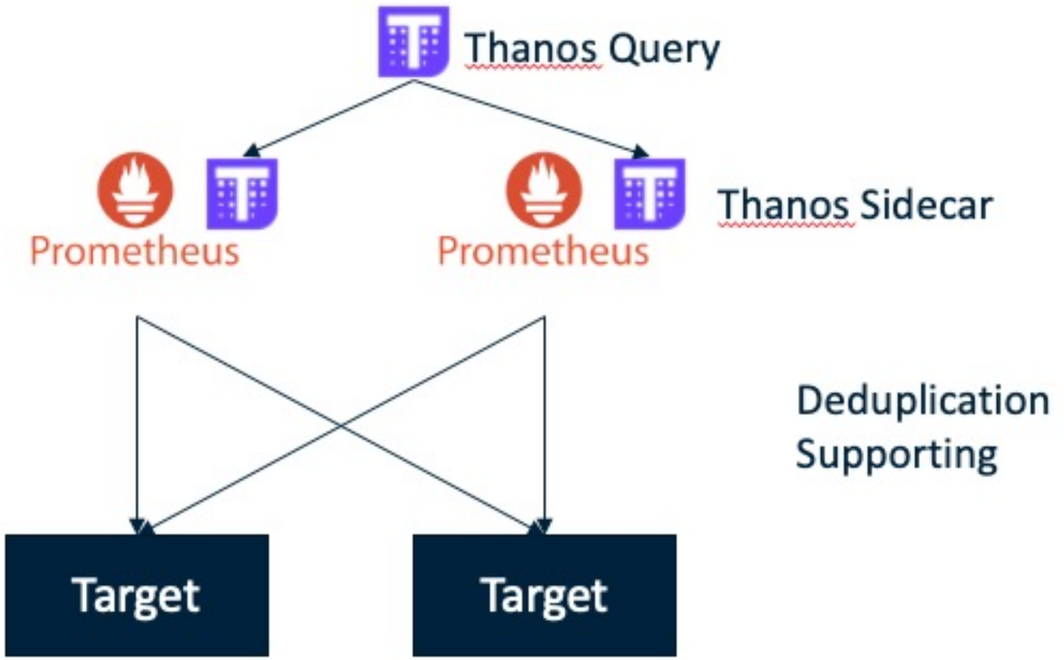
4.2 Performance Monitoring

Thanos는 Deduplication을 지원하여 Prometheus의 중복 저장 이슈를 해결

Prometheus HA



Thanos HA

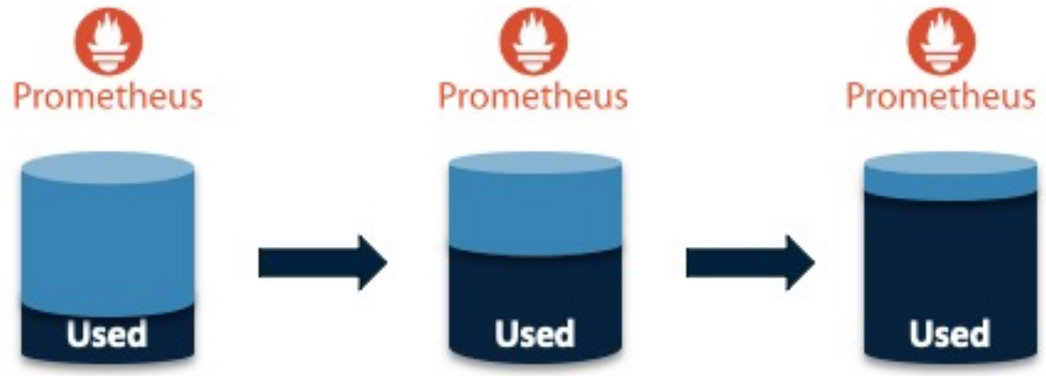


4.2 Performance Monitoring

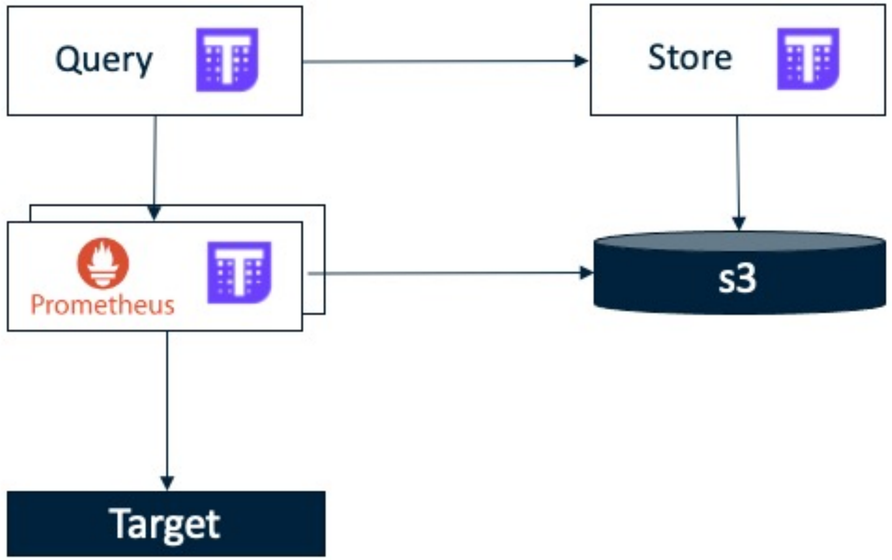
Thanos는 수집된 메트릭을 S3에 저장하여 로컬 디스크 용량 이슈를 해결함

(S3, Google Cloud, Azure Storage, Swift지원)

Prometheus Local Storage



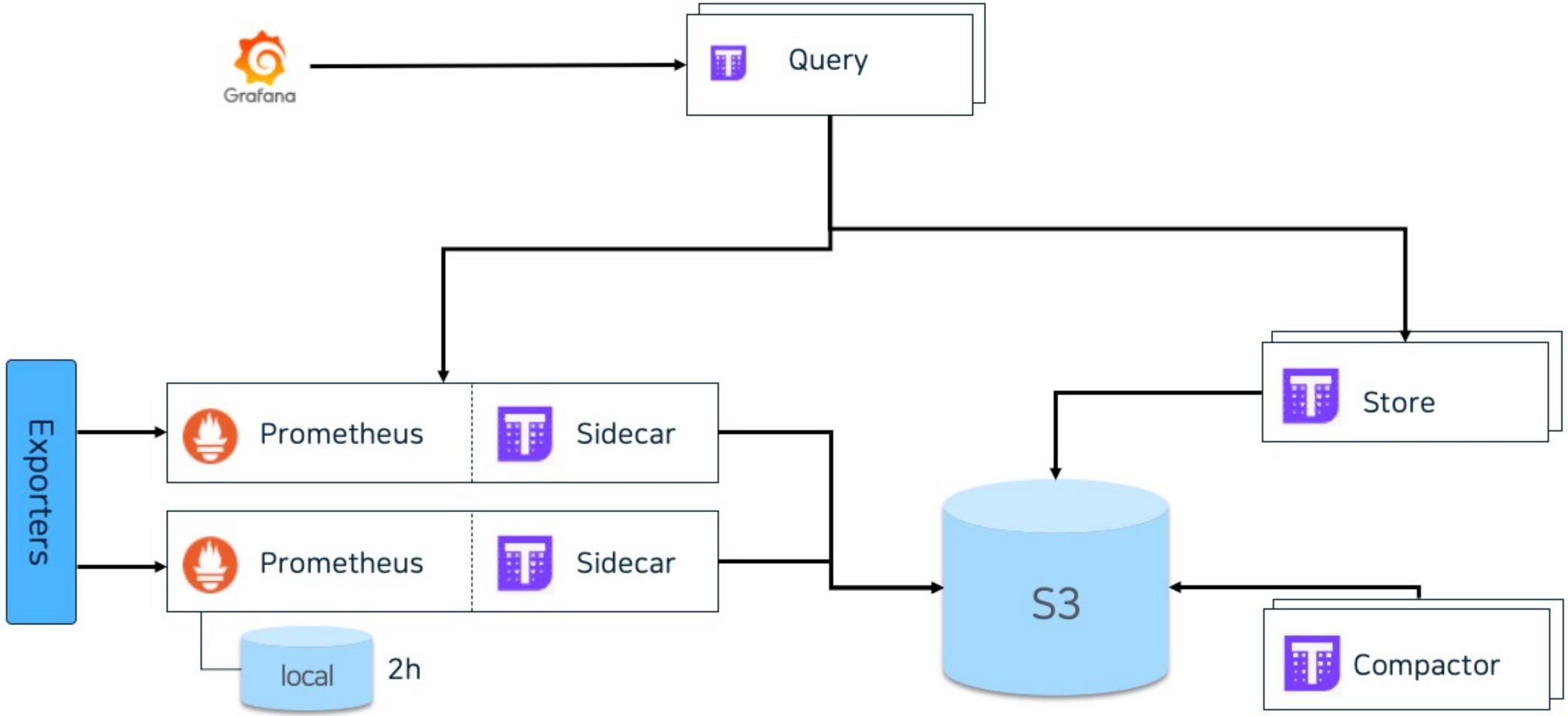
Thanos Remote Storage



4.2 Performance Monitoring

중앙 수집방식에서 클러스터 별 수집방식으로 변경함

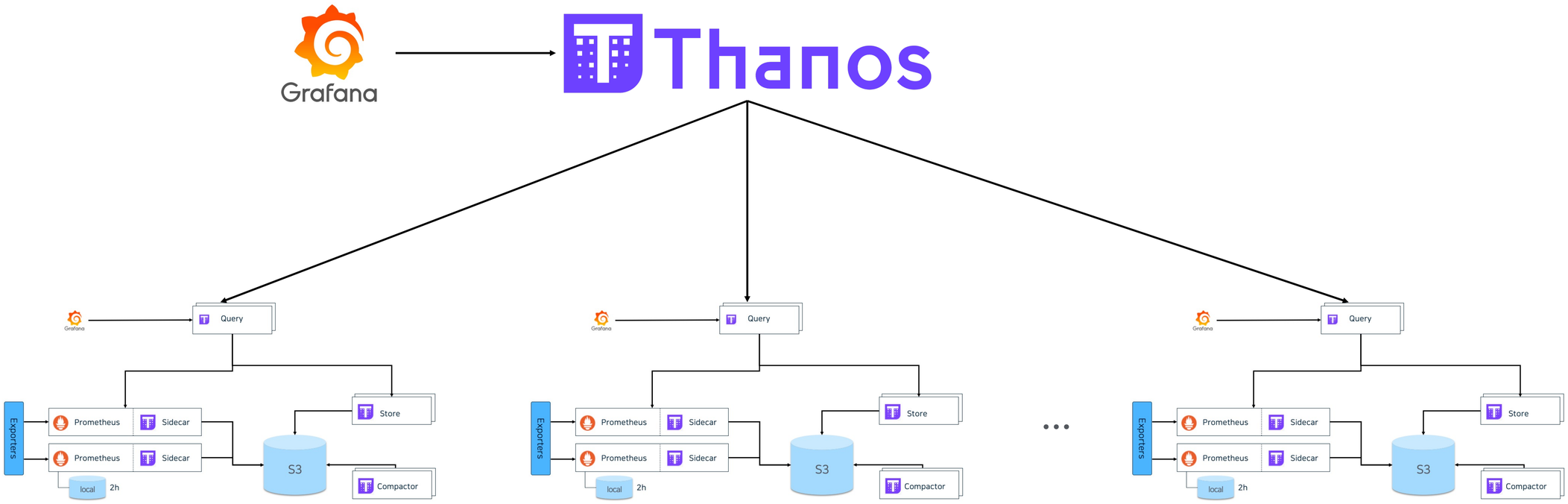
모든 클러스터에는 타노스가 적용됨



4.2 Performance Monitoring

모든 클러스터 정보를 한눈에 확인하기 위하여 글로벌 타노스를 적용함

상세 모니터링은 개별 로컬 클러스터를 통해 확인함

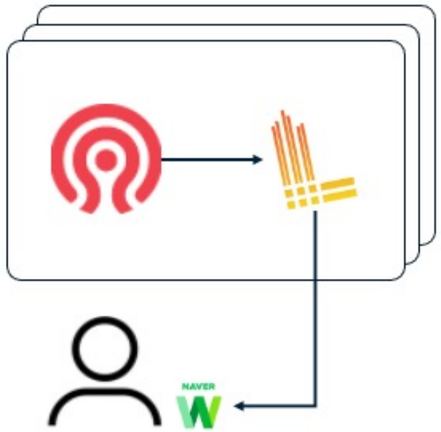


4.3 Log Monitoring

글로벌 구축이 시작되면서 IDC간 로그 전송이 발생함

로그는 발생된 IDC에서 처리 후 Alert Event만 전송되는 구조로 변경함

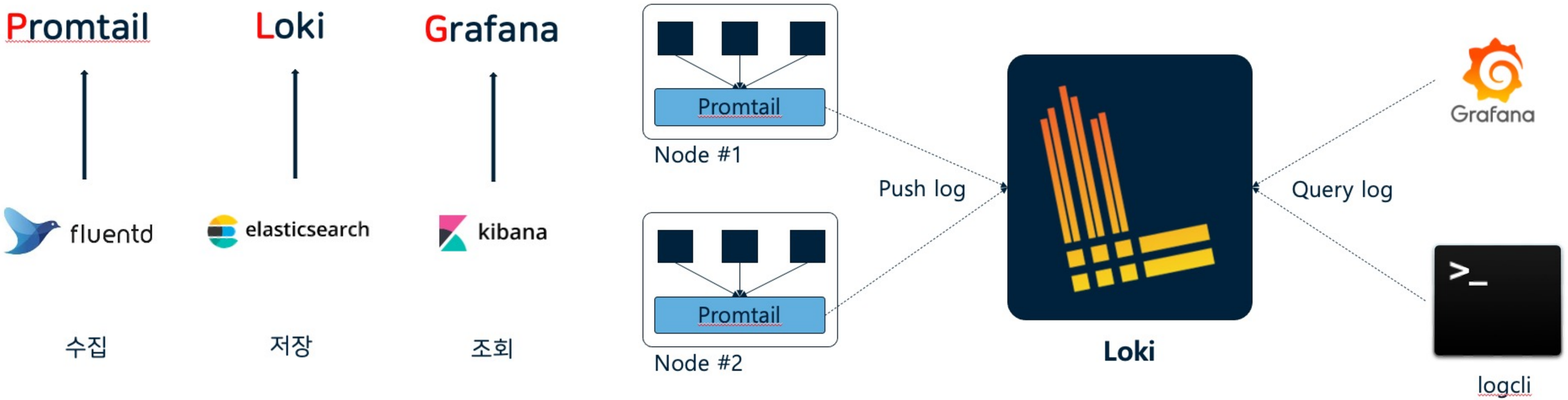
중앙수집구조



클러스터별 수집구조

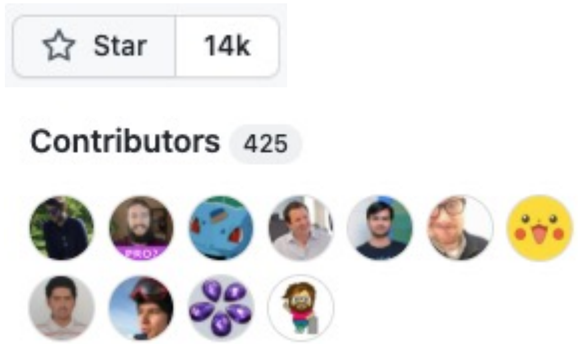
4.3 Log Monitoring

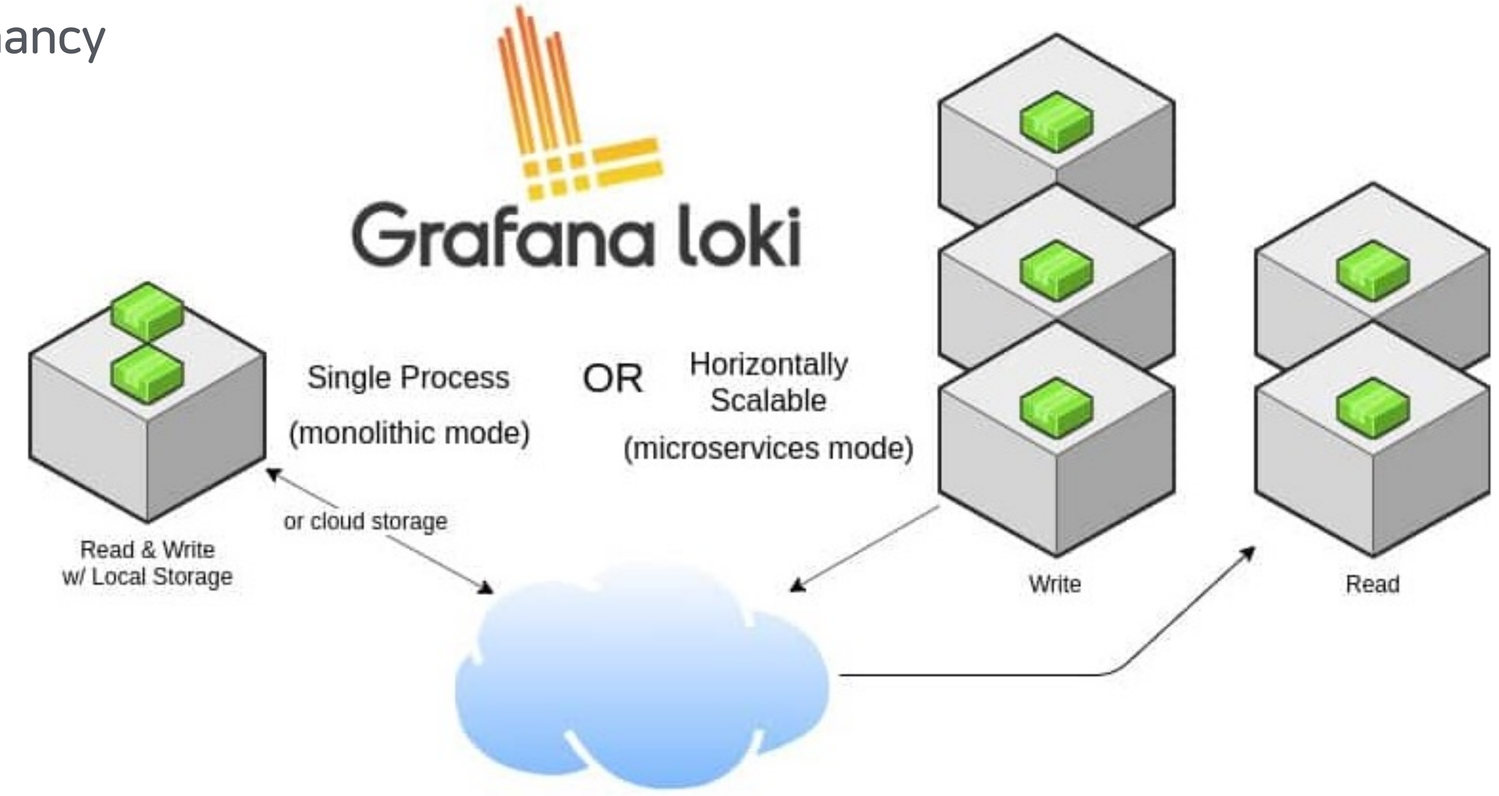
PLG Stack을 도입하고, 클러스터별 로그는 수집하는 구조로 변경



4.3 Log Monitoring

- Loki project started at Grafana Labs in 2018.
- Scalable, Highly-available, Multi-tenancy
- Designed cost effective
- Easy to operate
- https://github.com/grafana/loki



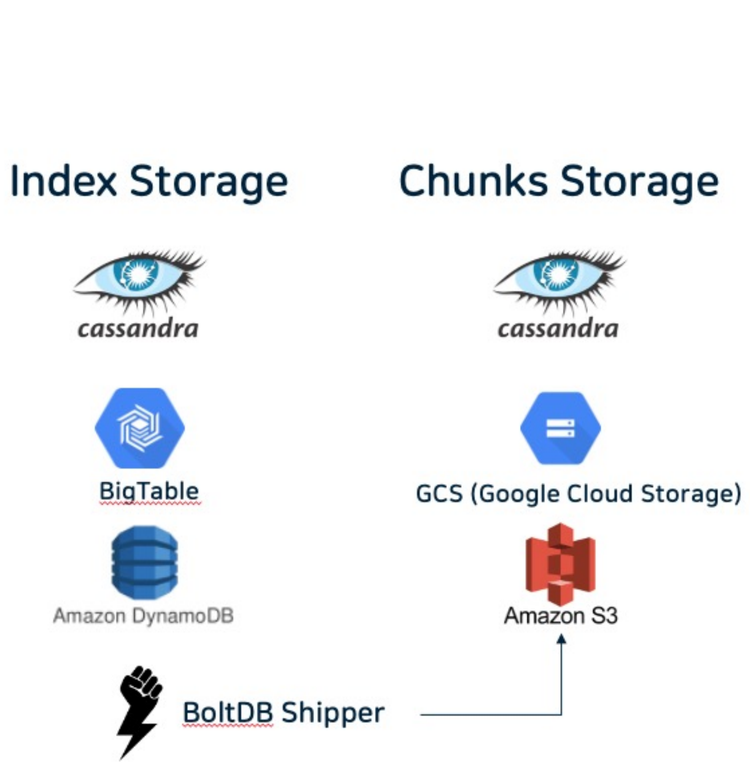


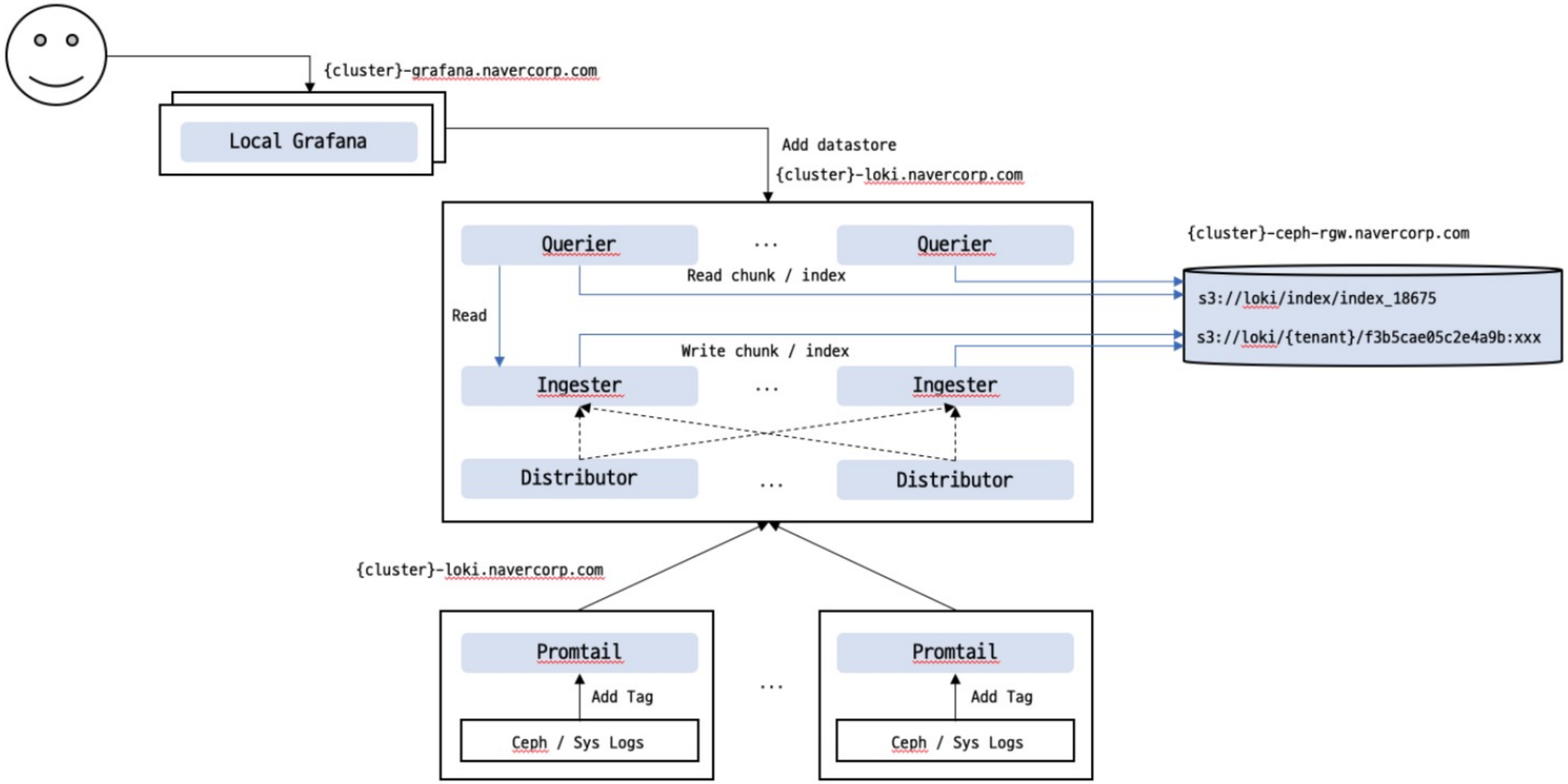
https://grafana.com/blog/2018/12/12/loki-prometheus-inspired-open-source-logging-for-cloud-natives/

4.3 Log Monitoring

BoltDB와 BoltDB Shipper지원으로 S3만 있으면 Loki구축이 가능해짐

Ceph Object Storage S3 API를 통해 서비스 구축

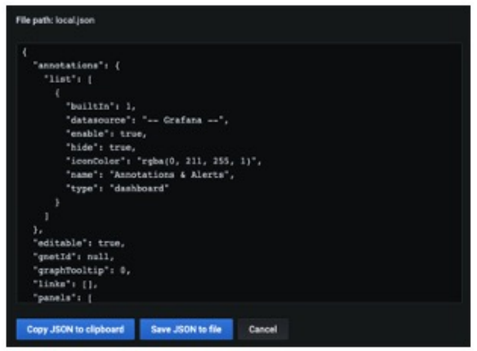




4.4 Grafana Template

클러스터별 Grafana도입으로 인해 템플릿을 통한 배포 수행

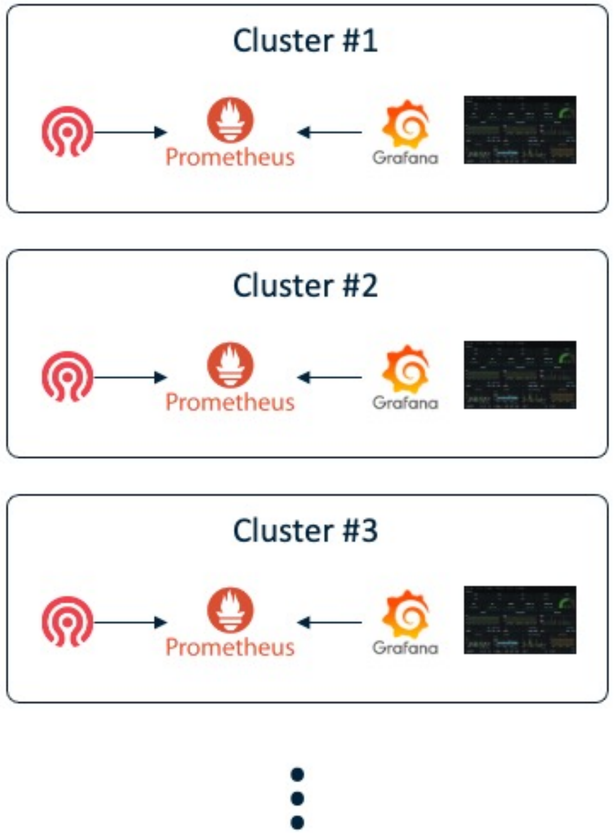
Dashboard 생성



Json export

/etc/grafana/dashboards/dash.json Grafana Node

템플릿 적용



템플릿 수정 시 반영이 어려움

4.4 Grafana Template

Dashboard API를 통해 수 십개 Grafana관리를 쉽고, 빠르게 처리함

› HTTP API › Dashboard HTTP API

Dashboard API

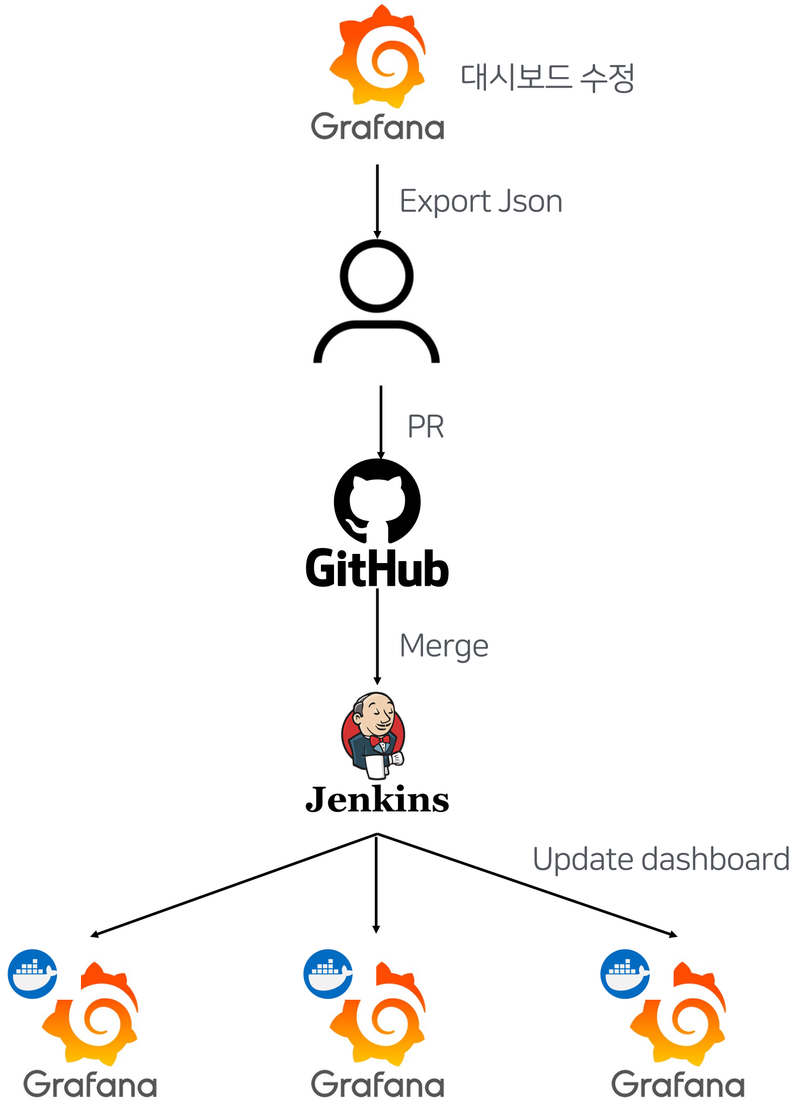
Create / Update dashboard

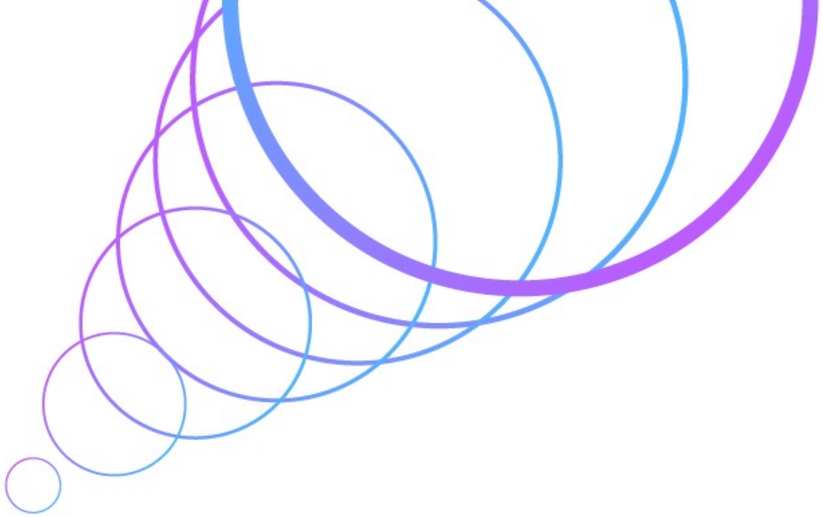
`POST /api/dashboards/db`

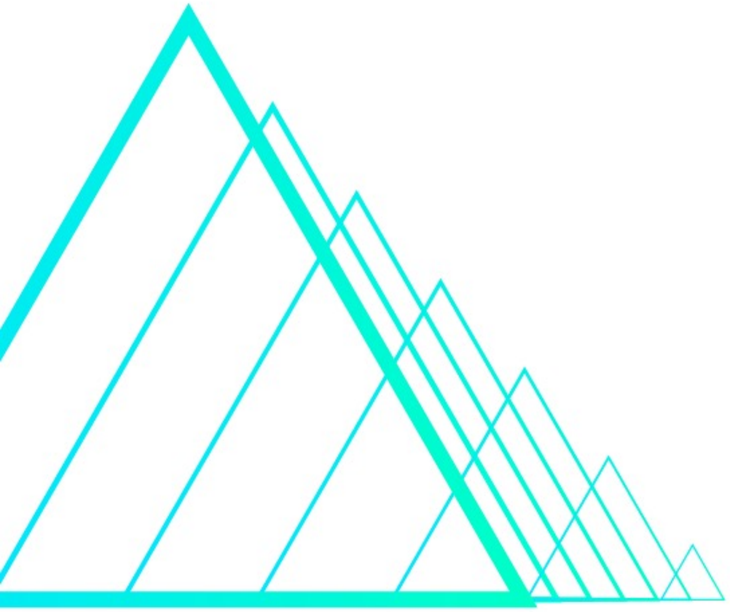
```
POST /api/dashboards/db HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "dashboard": {
    "id": null,
    "uid": null,
    "title": "Production Overview",
```

https://grafana.com/docs/grafana/latest/http_api/dashboard/#create--update-dashboard







Thank You

유장선 / jangseon.ryu@navercorp.com