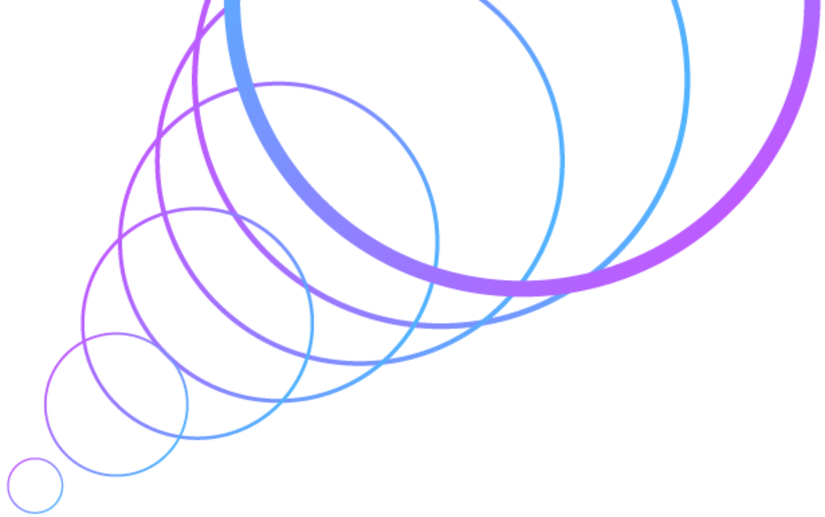
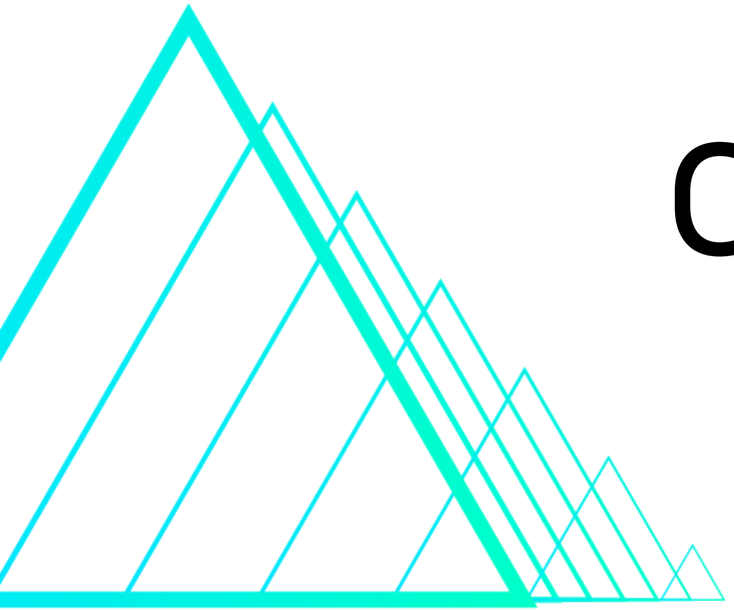




Cloud Monitoring Deep Dive

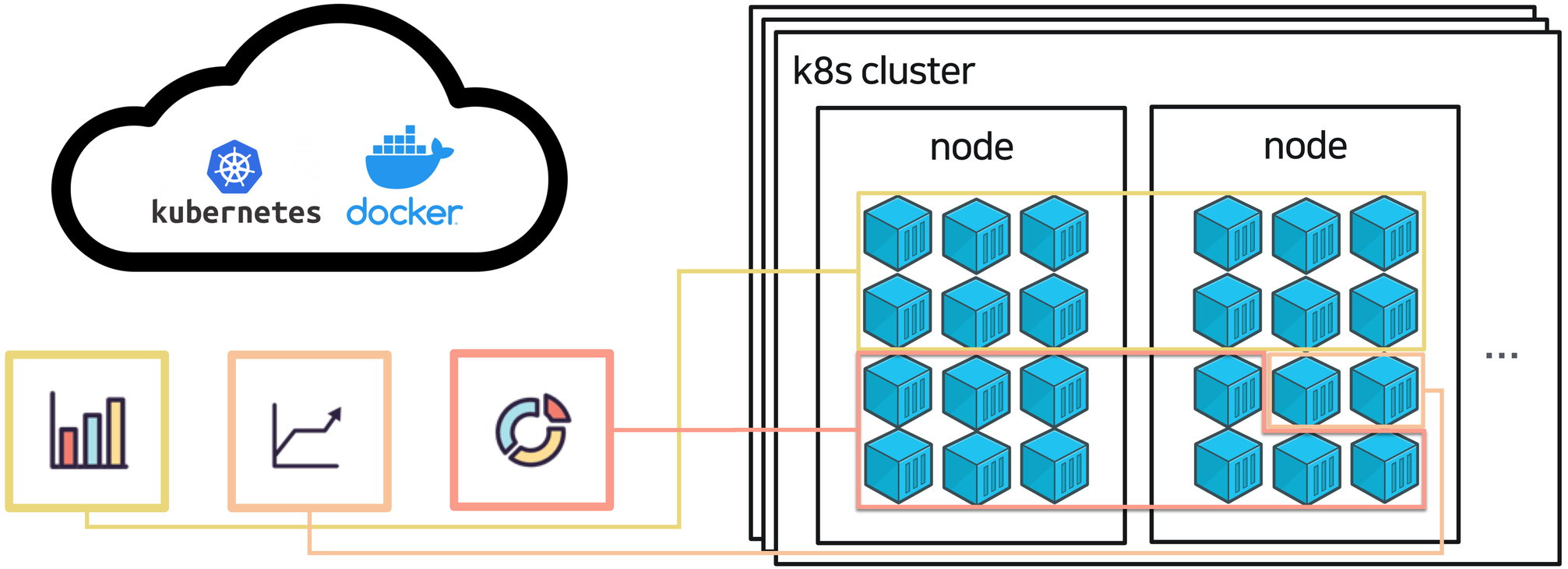
이승하 NAVER Computing platform

CONTENTS

1. Cloud Monitoring

1.1 Cloud Monitoring 이란?

Cloud 내 모든 구성 요소들의 상태를 지속적으로 수집하고,
수집 결과를 시간에 따른 데이터로 시각화하여 제공



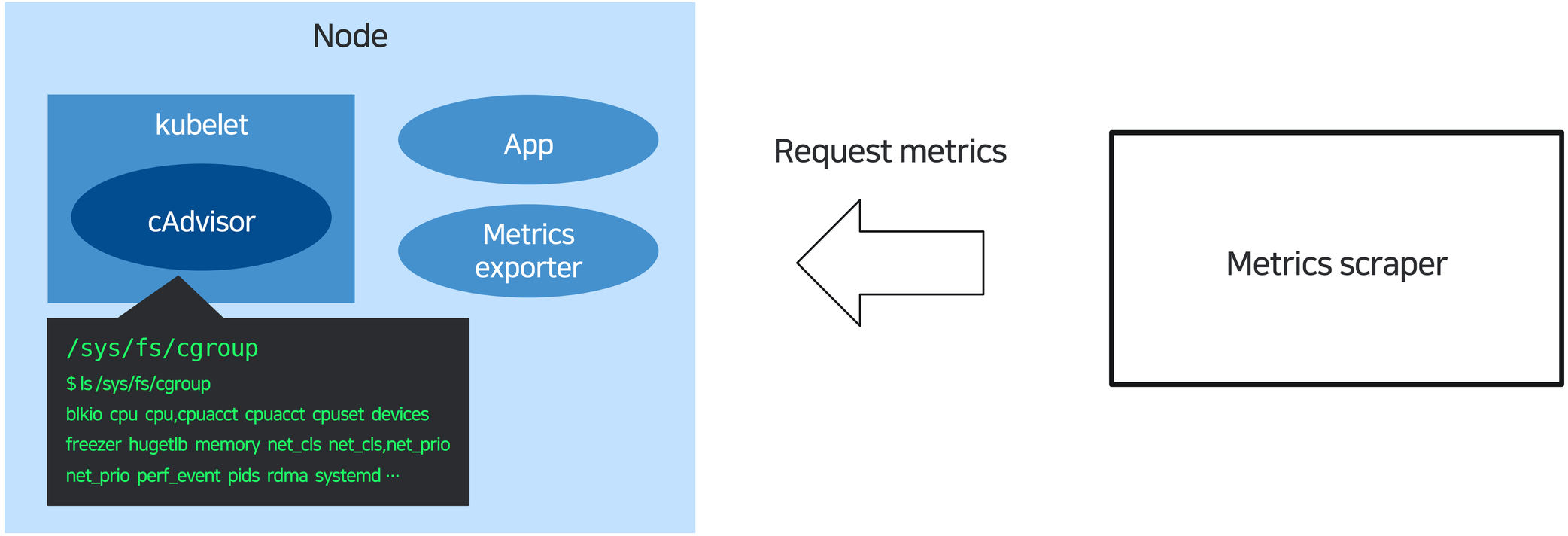
1.2 Monitoring을 위한 요소

1.2.1 수집 - 소비하는 리소스들에 대한 관찰

DEVIEW 2021

Container/Host resources, k8s meta-data등의 다양한 수집

- Container를 관리하기 위해 필요한 다양한 요소들의 상태 및 특징들을 수집

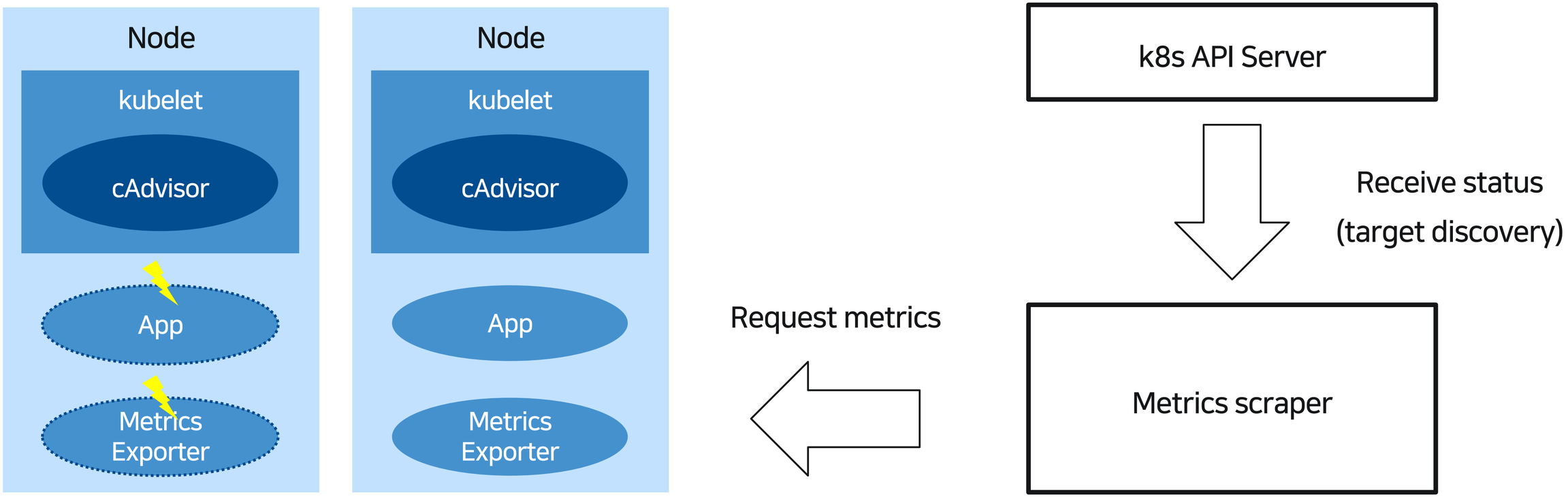


1.2.2 수집 대상 - Dynamic Target Discovery

DEVIEW 2021

Container는 유동적으로 다른 Node에서 구동 가능한 형태

- 수집 대상의 상태가 변해도 지속적인 수집이 요구됨

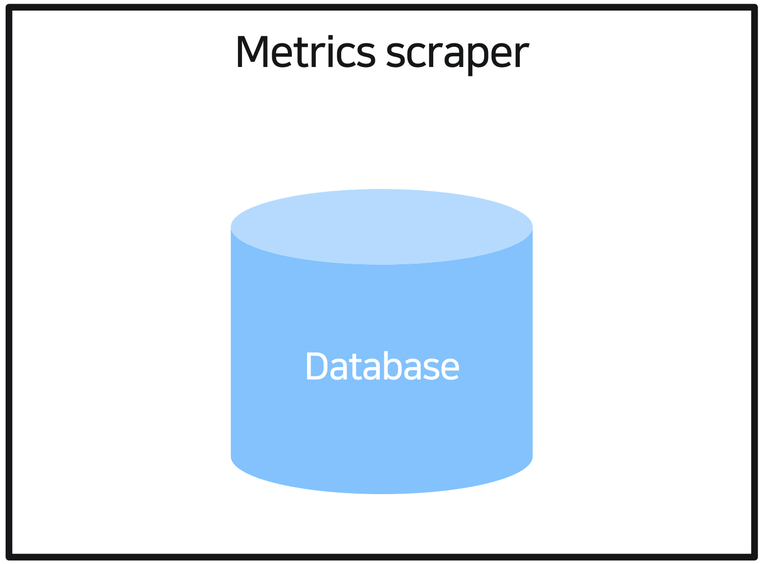


1.2.3 수집 결과 – 데이터베이스 관리

DEVIEW 2021

다양한 지표를 관리할 수 있는 저장공간

- 동적으로 다양한 지표가 발생하는 환경
- 대용량의 데이터를 저장하면서 이슈 대응 및 의사 결정을 위한 빠른 조회가 요구됨



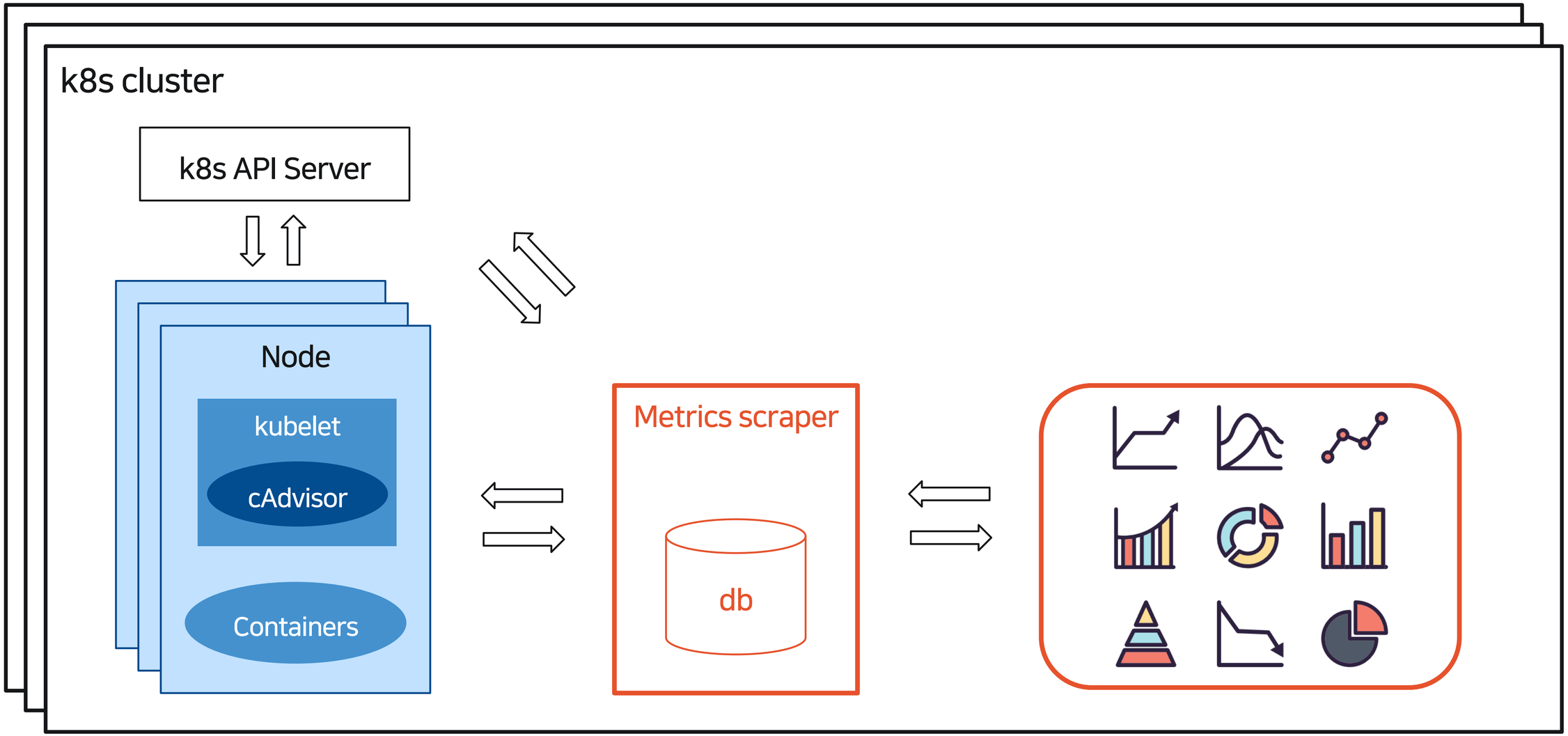
Request timeseries

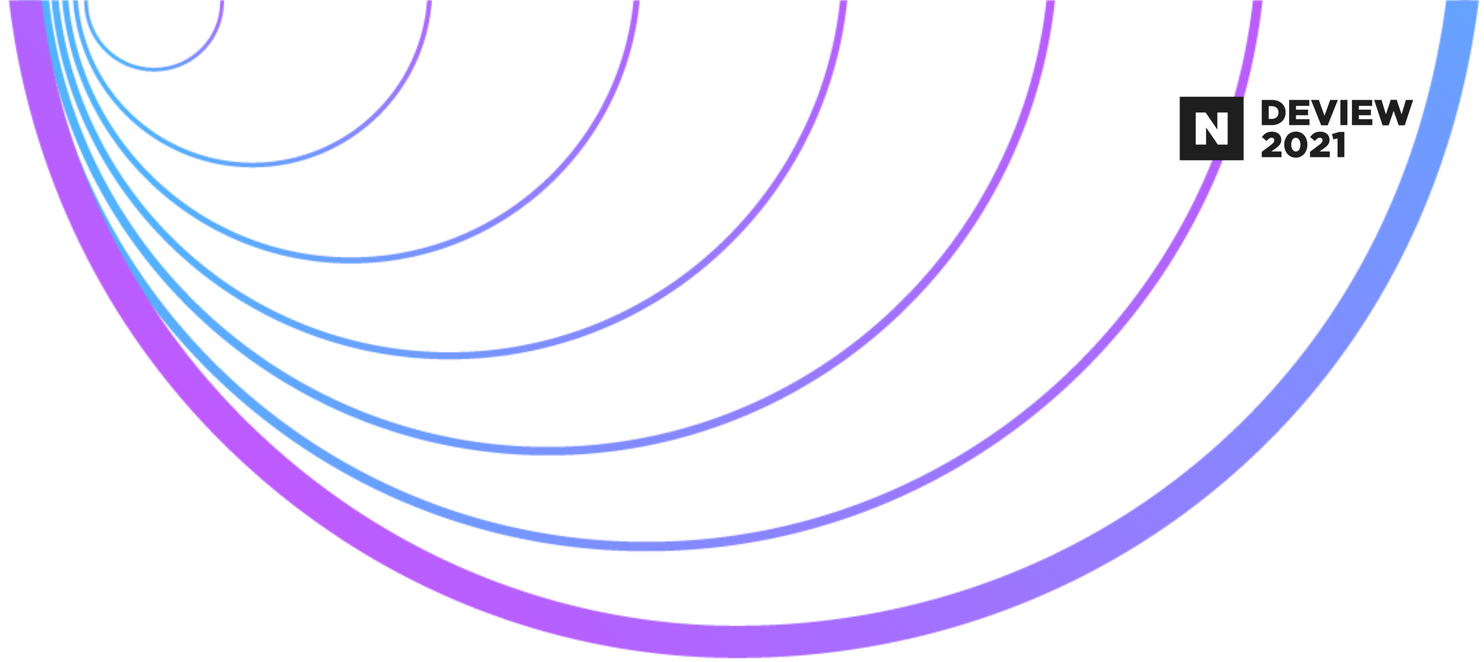
2021/09/13 10:11:12 ~ 2021/09/13 11:11:12

Returned timeseries

[{timestamp, value},{1631538673, 4.2},
{1631538674, 10.9},…{1631542272, 0}]

1.3 Cloud Monitoring Overview





2. Cluster Monitoring

2. Naver Container Cluster

DEVIEW 2021

Namespace 단위의 Multi-Tenancy Kubernetes Cluster

- Kubernetes 환경에서 Cluster를 관리하고, 사용자를 Namespace 단위로 구분하여, Namespace별로 격리된 리소스 사용성을 제공하고 있습니다.
- 관련 내용은 Multi-Tenancy Kubernetes on Bare Metal Servers를 참조해주시기 바랍니다.

https://deview.kr/2019/schedule/275

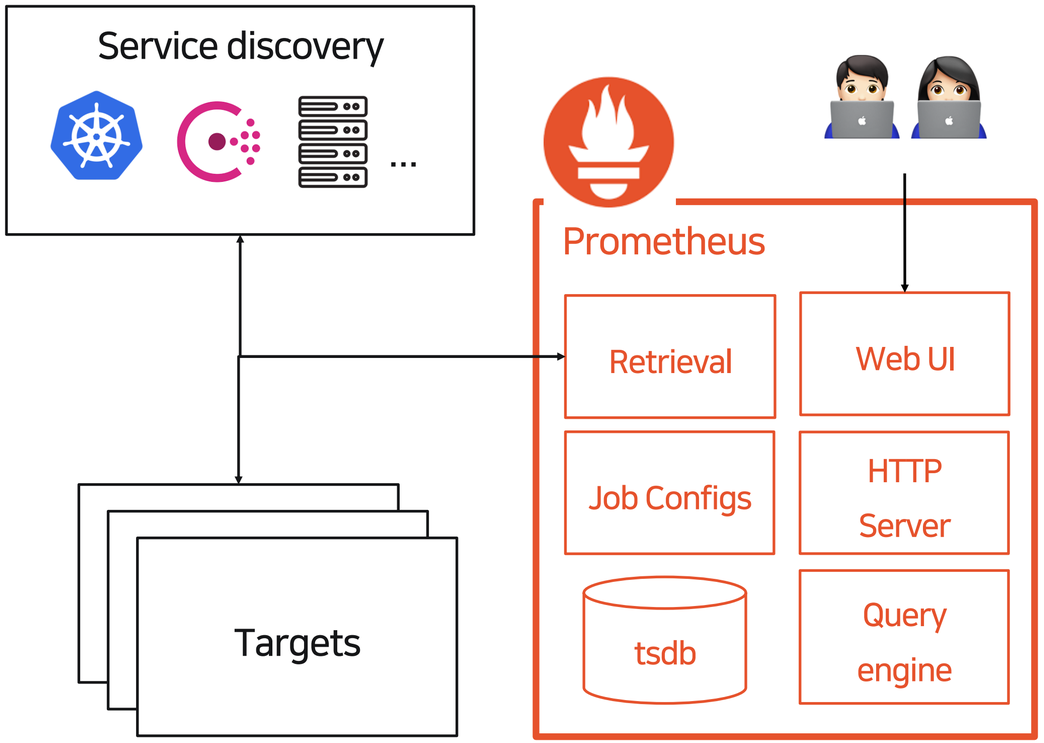
2.1 Cluster Monitoring을 위한 요소

다양한 종류의 지표와 Namespace 단위의 사용자 지표 제공이 요구됨

- k8s 환경에서 적절한 scraper, storage, visualizer 가 필요
  - k8s 환경에서의 Dynamic Target Discovery
  - 각 모니터링 서비스를 container 형태로 필요에 따라 배포
- k8s, node, pod, container 등 여러 출처의 지표에 대한 부하를 고려
  - 대상이 다양해서 수집에 대한 안정성이 우선
  - 수집 대상에 대한 다양한 상태와 특징을 정해줄 수 있어야함
- Namespace 사용자 단위로 분리된 지표 제공
- 1년 이상의 장기 추세(Long-term) 확인을 지표 저장

2.1.1 Prometheus

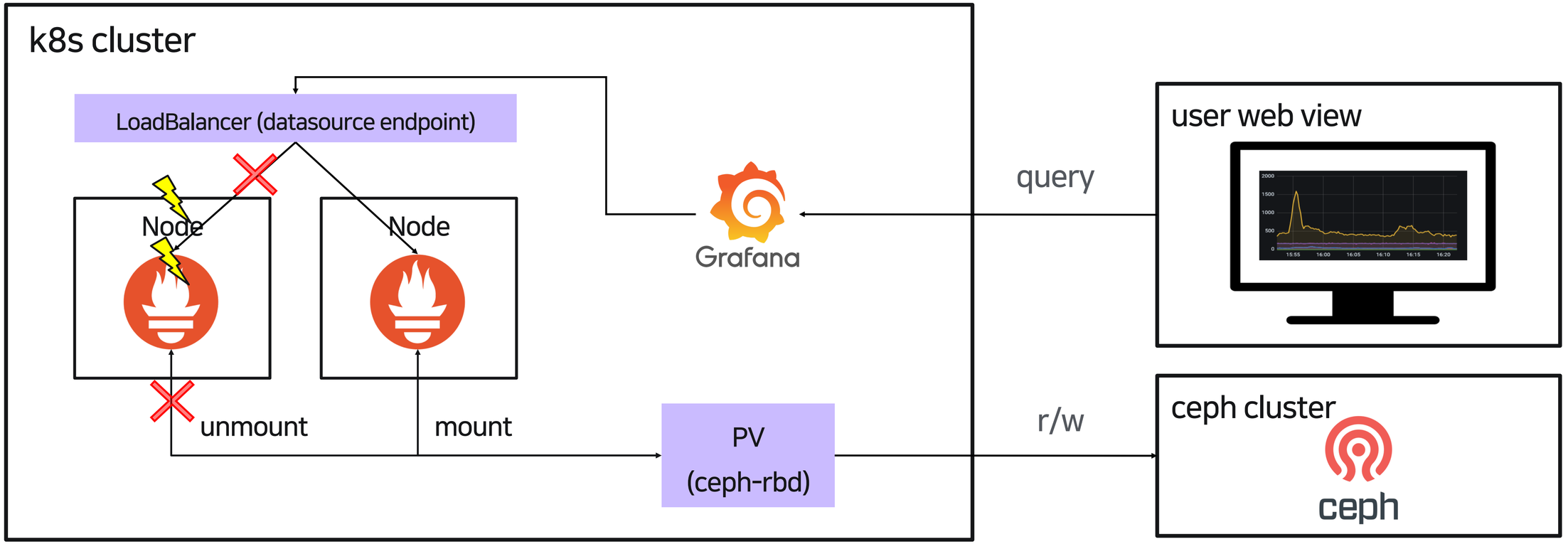
Prometheus로 Metrics를 일정 주기로 수집



- Dynamic Target Discovery
  - Discovered targets로 부터 일정한 주기마다 현재 값을 수집
  - Micro-Service가 늘어나는 k8s Cloud 환경에서 다양한 수집 대상에 대한 상태 확인
- PromQL
  - label key, value 기반으로 지표의 특징을 다양한 조합으로 쿼리가 가능
  - k8s 대부분의 지표가 Prometheus metrics 타입으로 제공 중
- tsdb
  - label을 index key로 가지는 inverted index, 연관된 series만 조회
  - 일정 주기로 timeseries block을 압축(compaction)

2.1.2 Storing & Graphing

부하를 분산하고 장애가 발생하면 다른 Node에서 구동가능하도록 구성



2.1.3 수집 부하

수집에 대한 과부하 발생

Container, Node 확장 => 관련 지표 일괄 증가

- Prometheus의 cpu, memory, storage 사용량 급증
- 부하로 인해 하나의 Prometheus가 비정상 동작하면 다수의 수집 누락이 발생

요청 응답 지연

- 다양한 지표들이 추가되면서 label cardinality가 낮아짐 index별로 조회해야 하는 block이 증가
- 특정 label key에 대한 응답의 크기가 비대해지면서 연산 및 응답 처리에 지연 발생

2.1.4 수집 부하 - Categorize

수집 대상을 카테고리별로 분산

Container, Node 확장

- category별로 튜닝포인트와 증설 계획을 분리해서 운영
  => 일괄 증가하는 부담을 축소
  => Prometheus의 리소스 부담 감소

요청 응답 지연

- Category(target)별로 label 구분
  => index 조합 결과를 줄여 상대적으로 빠른 조회가 가능
  => label 조합 수가 과도한 지표는 미리 연산하여 저장(rule recording)

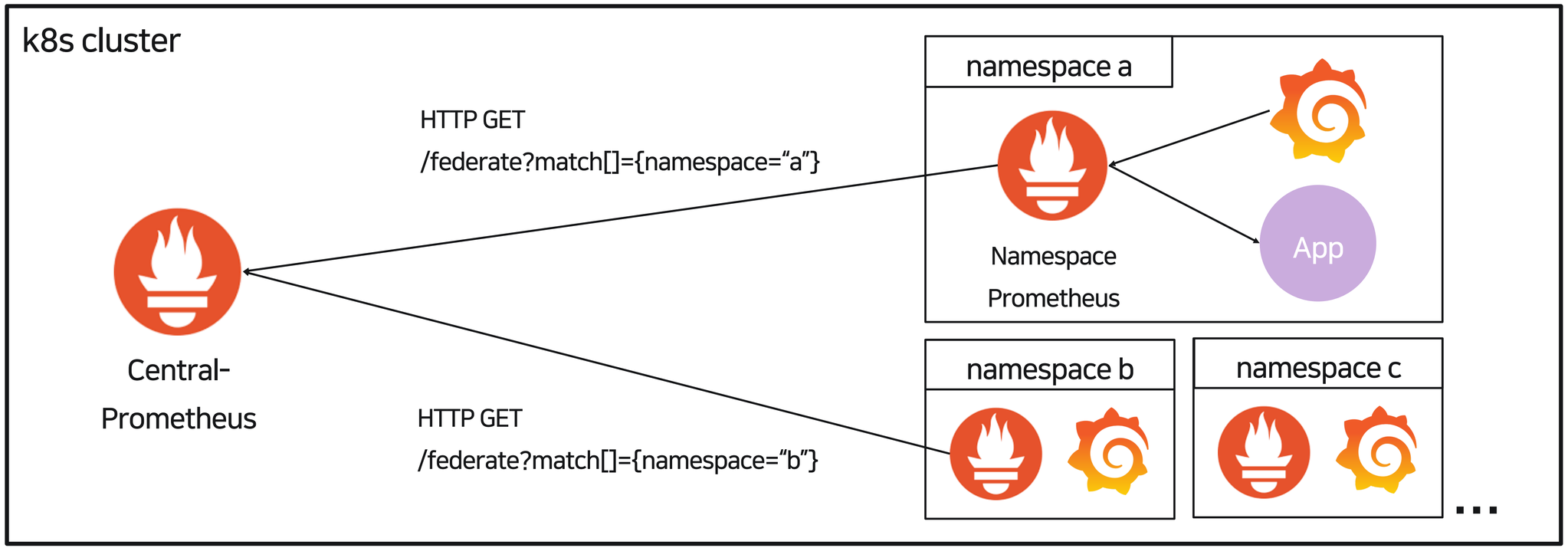
이슈 트래킹

- Target 문제로 인한 수집 실패시, Category 구분이 명확하므로 failure point가 빠르게 도출될 수 있음

2.1.5 Namespace-tenancy

DEVIEW 2021

Namespace 단위로 격리된 사용자 Monitoring 제공



2.1.6 Long-term 관리

1년 이상의 장기 데이터를 관리

- Long-term은 장기적인 추세를 확인하는 용도
  - cluster 증설 등에 참고
  - cluster 단위의 pod, container, resource 요구량 산정

e.g. 약 2,200,000 samples/sec을 수집하는 환경이라면,

retention 30d => 11TB x {replicas}

retention 1y => 132TB x {replicas} 😵

15s steps 조회 => 최소 5760 x {days}개의 sample 연산

* 사용상 과도한 비용을 부담하면서 수집할 필요가 없음
* 긴 time-range로 조회시, step에 따른 구간내 모든 sample들을 연산하는 부담 발생

3시간 주기로 수집했을때 용량을 다시 산정해보면,

retention 1y => 122GB x {replicas}

3h steps 조회 => 최소 8 x {days}개의 sample 연산

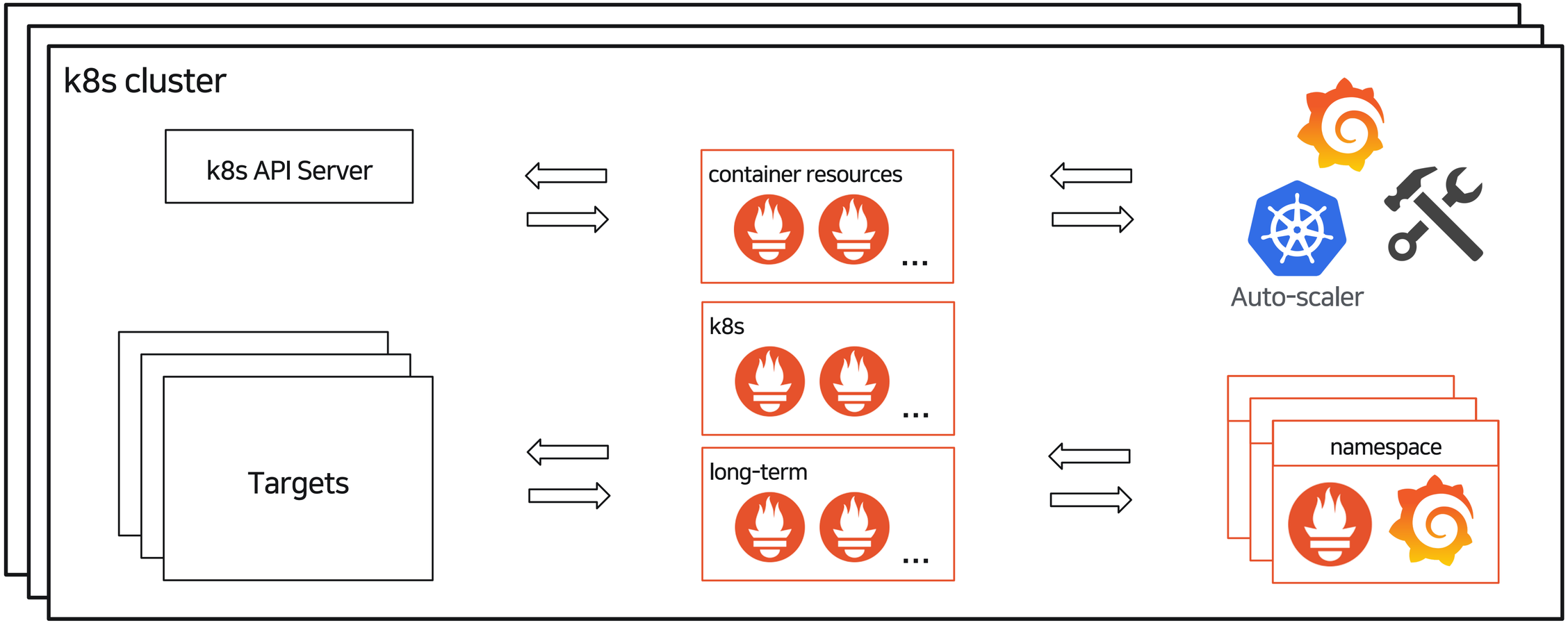
needed_disk_space = retention_time_seconds * ingested_samples_per_second * bytes_per_sample

Prometheus stores an average of only 1-2 bytes per sample

https://prometheus.io

2.2 Cluster Monitoring 구성

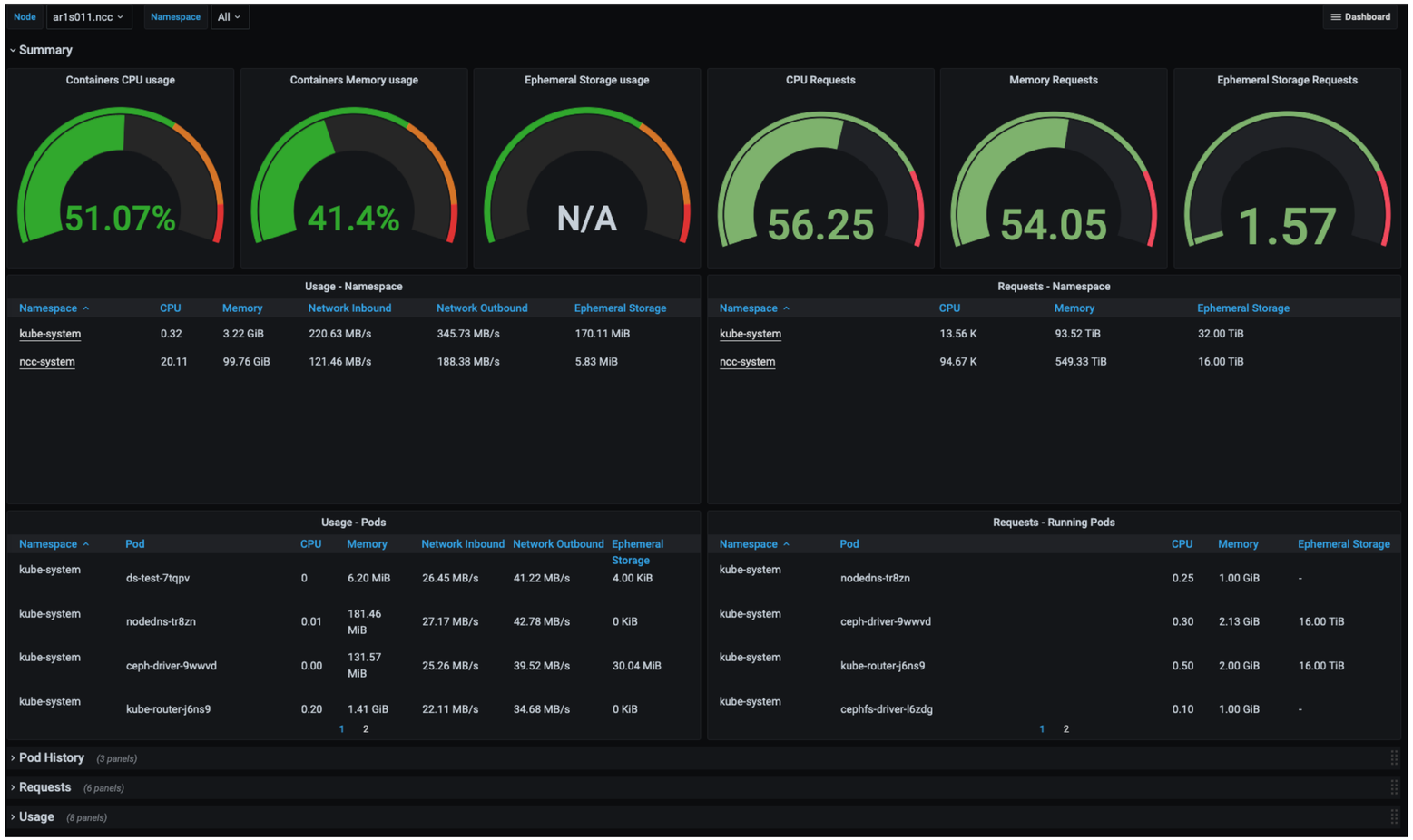
2019.10 기준 - 12+ Clusters, 47000+ Pods, 510+ Namespaces 의 지표 제공



2.2.1 Cluster Monitoring

DEVIEW 2021

Resource usage, node status 등을 체크하고 장애 대응 및 증설 등 유지보수에 활용



2.2.2 Namespace Monitoring

App의 resource usage와 k8s event를 제공하여 정상적인 구동을 판단할 수 있는 지표 제공

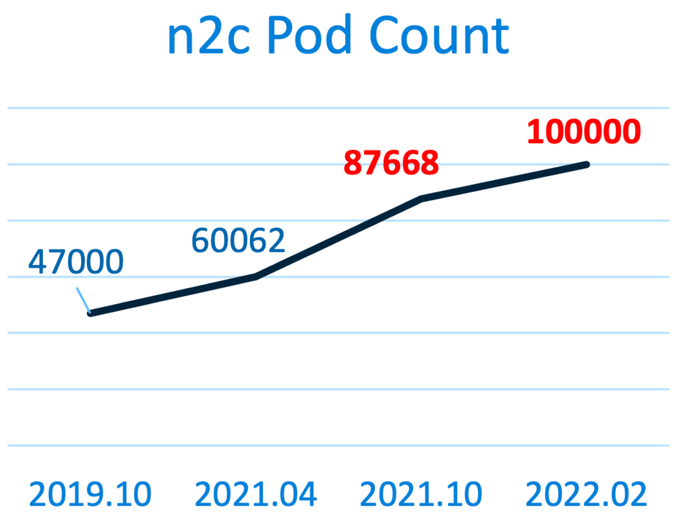
3. Large-scale Monitoring Crisis

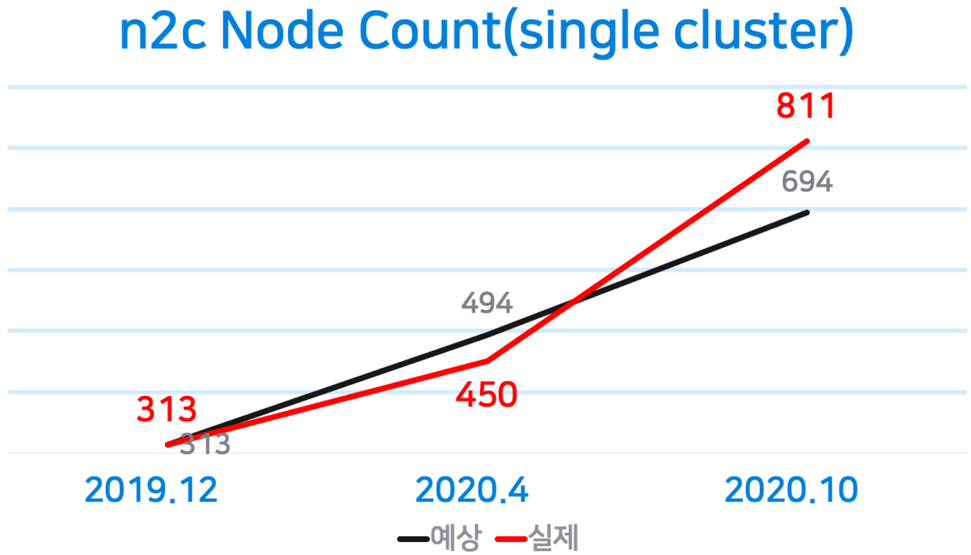
3.1 대용량 Prometheus의 한계

DEVIEW 2021

Cloud 사용자가 늘어나면서 Cluster는 점점 거대해져가는데...

- Node, Pod이 급격히 증가하면서 관련 수집되는 샘플량도 급증

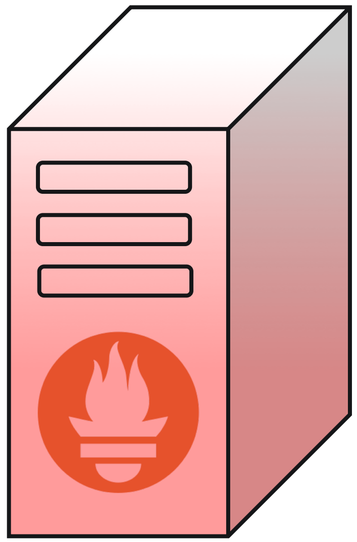




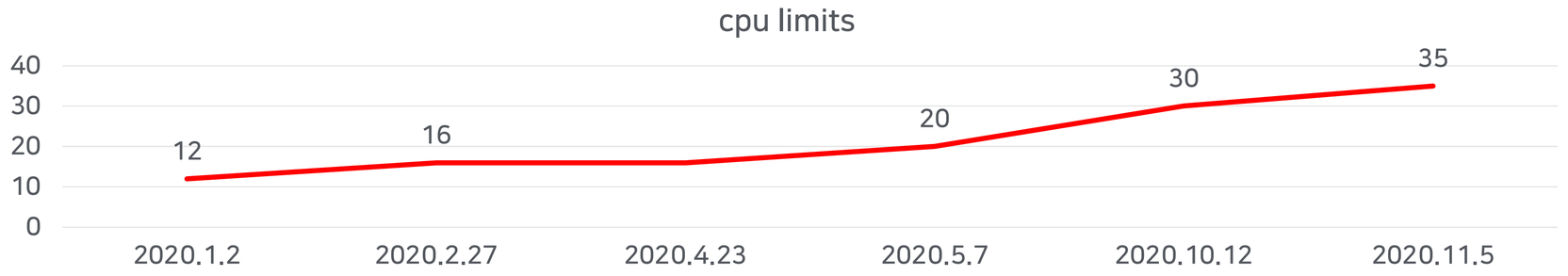
3.1.1 클러스터 수집량 증가

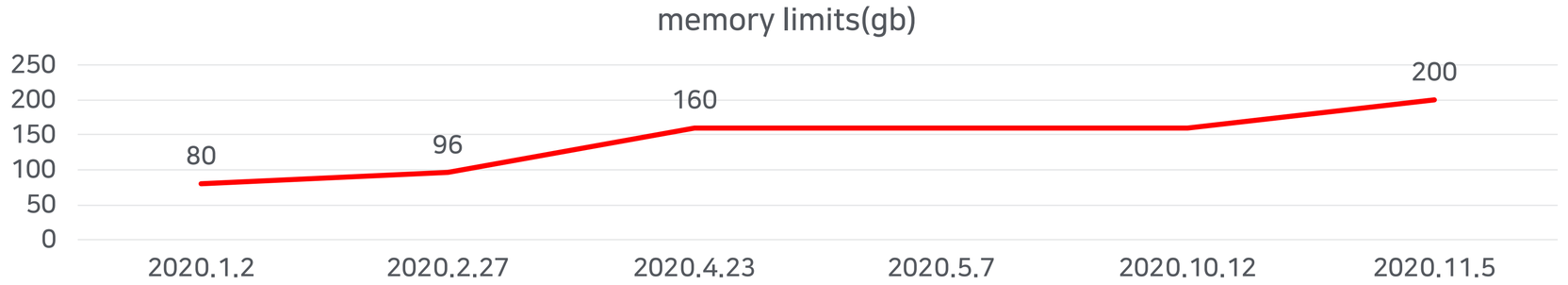
수집량 증가로 인한 Prometheus resource 요구량 증가

- Cloud 사용으로 배포가 간단해지면서 Pod, Container 수가 꾸준히 증가
- 사용량이 증가하고 있는 상황에서, PM에서도 감당하기 어려울 것으로 예상



PM 40 core / 256 gb





3.1.2 사용자 수집량 증가

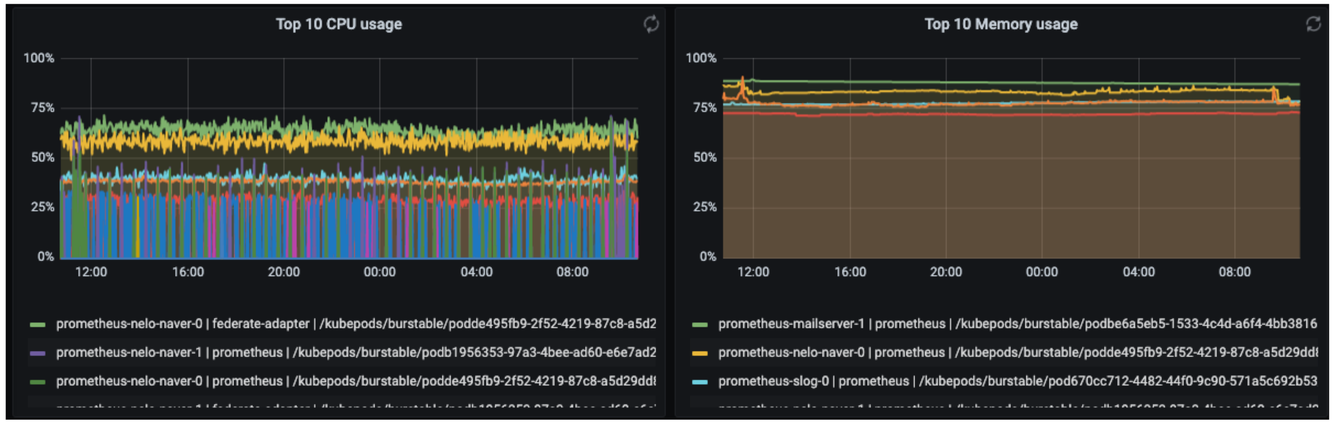
사용자 또한 수집량 증가로 resource를 과점유

- Over-provisioning 상한 관리를 위해 제공되는 기본 리소스 단위를 초과
- 수집량이 많아진 사용자의 Prometheus는 OOM이 빈번
  - Prometheus에서 OOM은 복구가 어려운 장애들을 유발
    (HEAD 유실, segment fault로 인한 무한 재시작 등 문제가 발생)

Container

cpu : 8 core

mem: 16 GiB



3.1.3 불완전한 High Availability

HA를 보장하기 어려운 상황들이 발생

#1. 복구 지연 및 실패

- 장애가 생긴 Prometheus는 자동 복구될 수 있도록 구성했으나, 복구 시간이 길어지거나 segment 문제로 인해 복구 실패가 발생
- 늘어난 sample로 write-ahead log file의 크기가 비대해졌고, Prometheus 재시작시 wal loading에만 최대 5시간 정도 소요된 경우도 발견 (Prometheus v2.19.0 이전 버전에서 확인된 경우이며, 근래 버전에서는 개선된 것으로 보임)

#2. API Serving

- 일부 Prometheus의 장애로 유입이 차단되면, 다른 Prometheus로 유입이 집중되는데, 유입이 몰린 Prometheus도 가용치를 넘어서면서 장애발생
- 유입이 몰린 prometheus가 down되고 이제 막 복구된 prometheus에 다시 요청이 집중되면서 2, 3 차 장애 발생 지속

#3. 넓은 수집 범위

- 대부분의 리소스를 수집에 사용하며 수집 대상이 많아질수록 리소스를 과점유하여 비정상동작을 야기할 수 있는 확률이 증가
- 수집 범위가 분산되지 않아서, Prometheus 장애 발생시 Cluster 내 모든 container 지표가 복구 지연 시간만큼 누락

3.2 운영하면서 받은 문의

DEVIEW 2021

제공중인 모니터링 시스템으로 해결이 어려운 문의들이 점차 증가

Q. Prometheus resource를 늘릴 수 있을까요?

- Cluster내 pod은 8 core 16GiB로 제한하여 관리하고 있음
- 전체의 리소스 비율을 관리하고 있기 때문에 일괄 상향이나 수집량이 많은 일부 사용자만 별도로 관리하기 어려운 부분이 존재

Q. 클러스터별 지표를 한곳에 모을 수 없을까요?

- 클러스터별로 독립된 Monitoring set을 제공중
- 쿼리한 결과를 한군데에서 aggregation해줄 컴포넌트가 요구됨

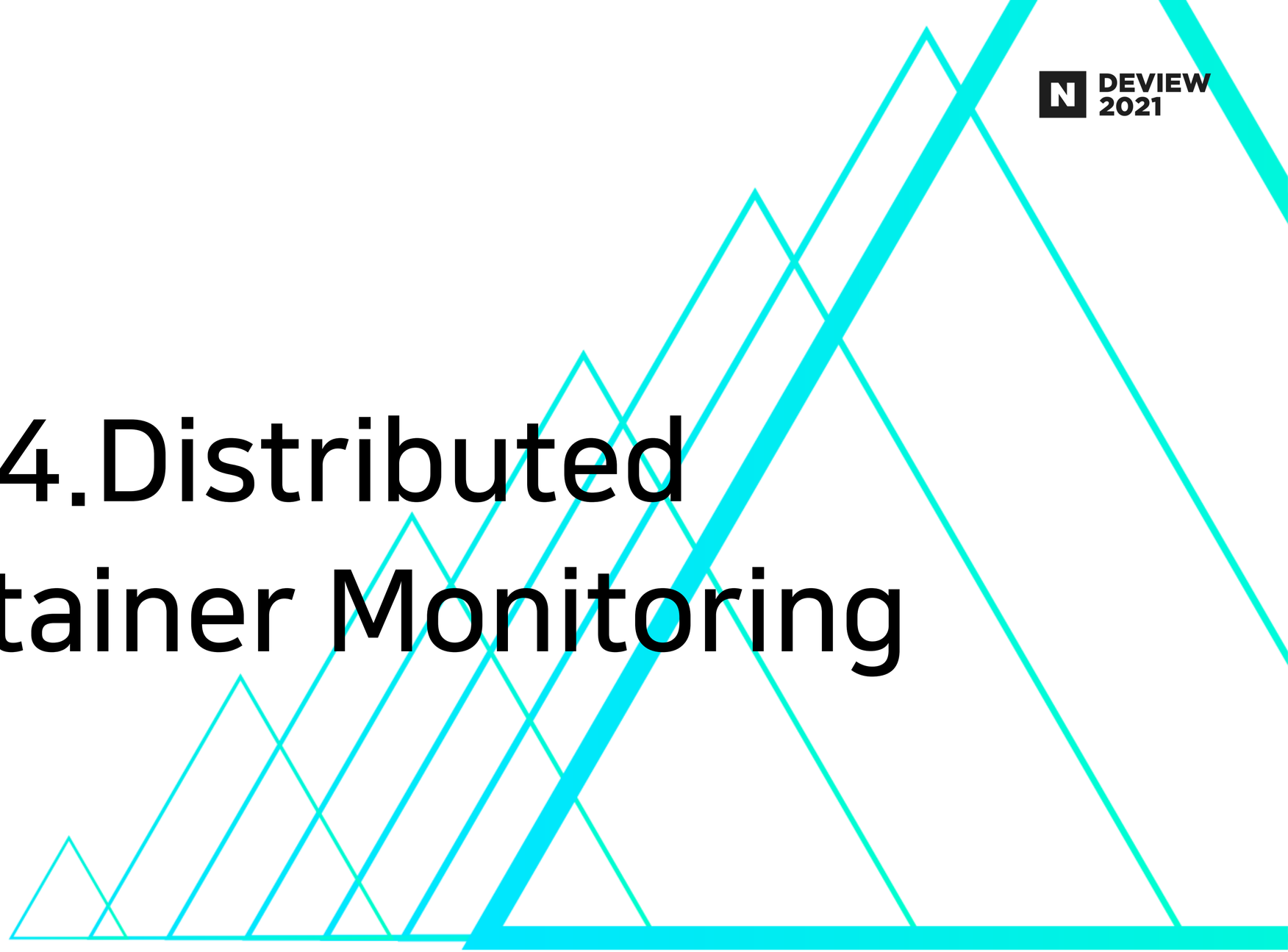
Q. 쿼리가 느려요

- Container가 점차 늘어나고 쿼리에 요구되는 결과값이 늘어나면서 쿼리지연이 발생

3.3 시스템에 대한 개선

가장 큰 이유는 수직적 확장의 한계

- 클러스터의 크기에 따라, 장비 내에서 Prometheus의 리소스 점유율이 증가한다고 해서 장비의 물리적인 CPU/Memory를 무한정 확장할 수 없음
- '작은 클러스터는 충분히 수용 가능했는데' 라는 경험으로 '큰 클러스터 안을 작은 단위로 나눌 수는 없을까' 라는 생각을 하게 되었고..
- 클러스터를 나누어, 확장 가능한 여러 개의 모니터링 그룹을 만든다는 목적으로 Distributed Container Monitoring 프로젝트를 시작 🚀



4.Distributed Container Monitoring

4.1 분산 처리의 필요성

대용량 분산 처리를 위해 주요 문제와 요구를 해결하는 새 시스템을 제공

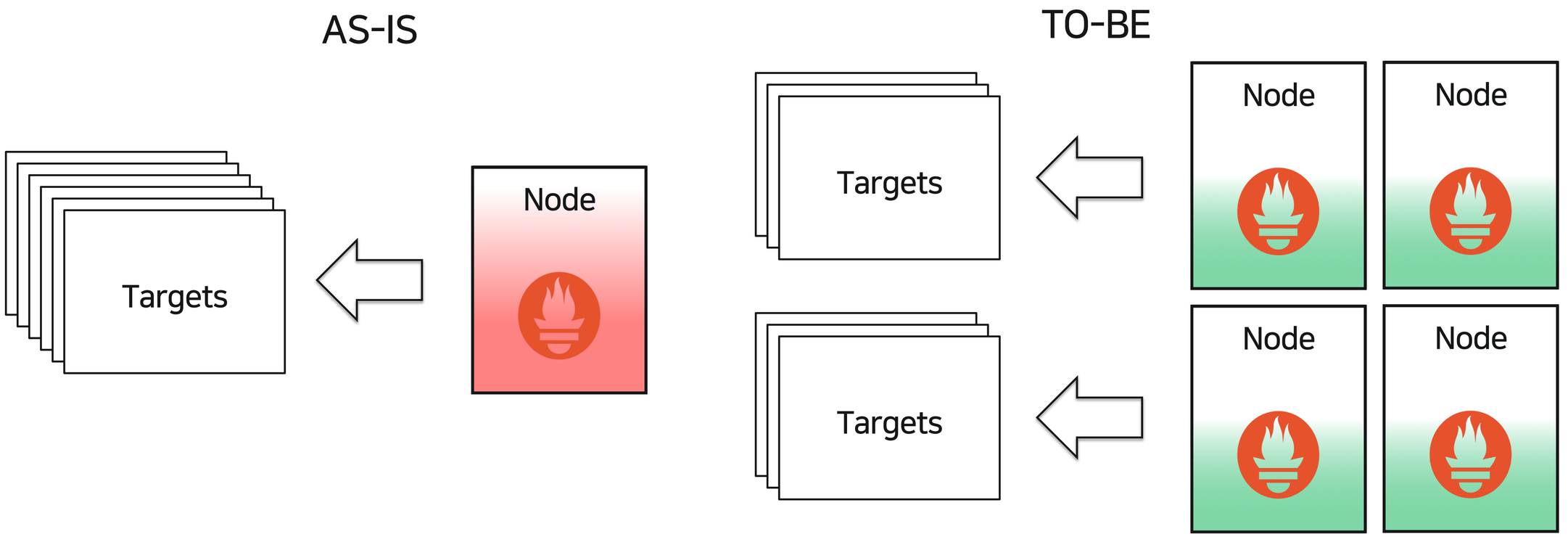
운영 관점	사용자 관점
대용량 지표 수집을 위한 리소스 사용 문제	대용량 지표 수집을 위한 리소스 사용 문제
복구 지연 및 실패	복구 지연 및 실패
쿼리 지연	쿼리 지연
사용자 Federation API Serving	Cluster 별 지표를 모아서 보기
과도한 수집 범위	

수집 단위를 나누어 개별 부담을 분산할 수 있을지?	분산 수집
분산된 수집의 결과를 어떻게 모아서 쿼리할지?	분산 쿼리
Query layer를 분리해서 확장하고 Cluster별 지표를 함께 연산할 수 있을지?	
분산 쿼리 제공을 위한 리소스 증가 범위가 과도하지 않은지?	

4.2 분산 수집

하나의 수집기가 가지는 수집 부담을 분산하고 확장 가능한 형태로 구성

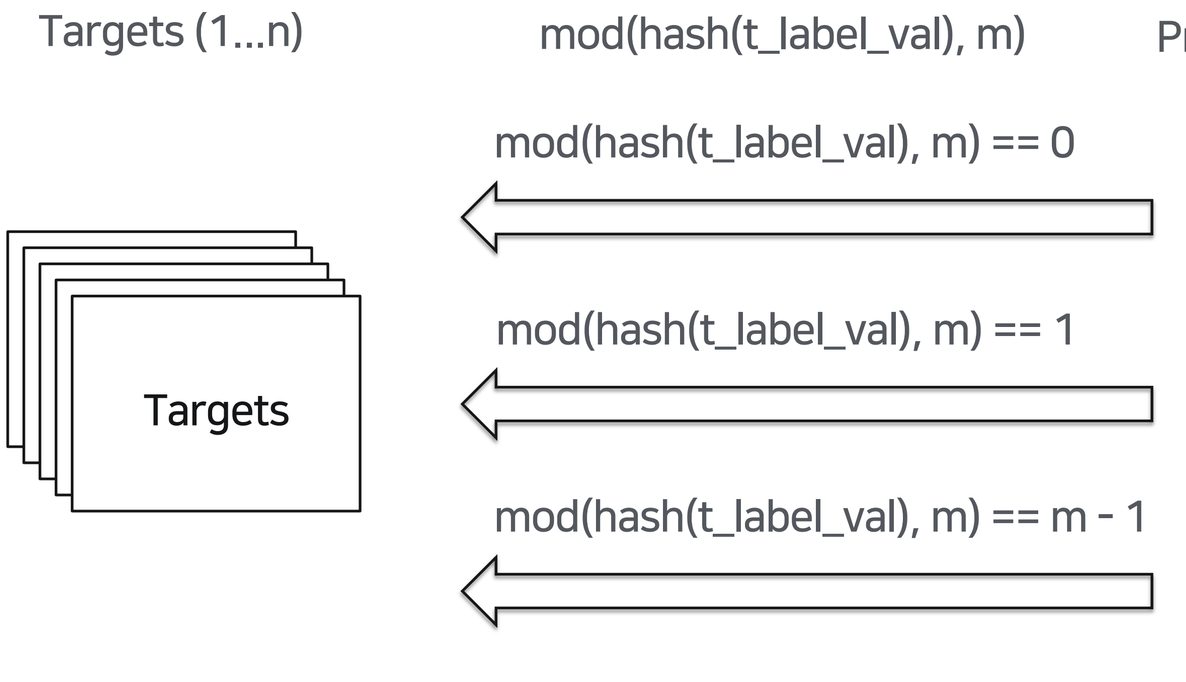
- AS-IS : Node의 리소스 대부분을 차지
- TO-BE : 수집기 하나당 부담해야하는 대상을 나누어 여러 Node로 분산/확장

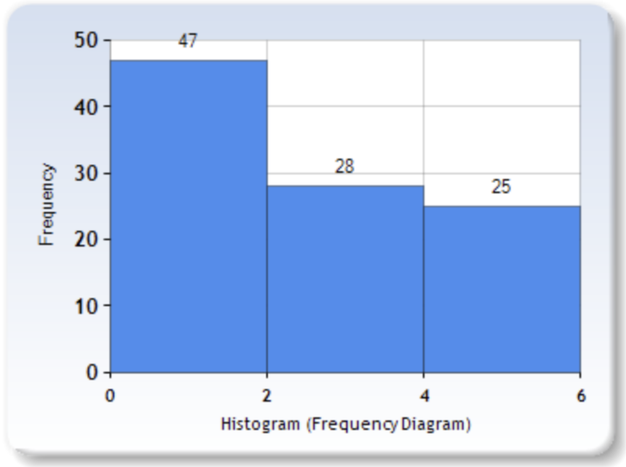


4.2.1 분산 수집 - 동적 대상 분산

Prometheus의 scrape target을 동적으로 분산

- Hashmod : target의 label value를 연산한 값과 설정 값이 일치하는 경우에만 수집
- Hashing한 값으로 판단하므로 항상 고르게 분포 되진 않지만 분산 효과는 유의미함





100 targets에 대한 분산 시뮬레이션

4.3 분산 쿼리 - Solution 검토

Issues	Thanos	Victoria-metrics
수집 단위를 나누어 개별 부담을 분산할 수 있을지?	hashmod를 활용하여 분산 확장 가능	hashmod를 활용하여 분산 확장 가능
분산된 수집을 어떻게 모아서 쿼리할지?	Thanos-query를 통해 단기 데이터(raw)는 prometheus로부터 장기 데이터(downsample)는 object-storage로부터 쿼리 결과를 모아서 제공 label 기반으로 deduplication	vm-select를 통해 지정된 storage로부터 중복을 제거한 쿼리 결과 제공 가능 timestamp 기반으로 deduplication
Query layer를 분리해서 확장하고 Cluster별 지표를 모아서 연산할 수 있을지?	Thanos-query query layer 확장, Multi-datasource 연동 가능	vm-select query layer 확장, Multi-datasource 연동 가능
분산 쿼리 제공을 위한 리소스 증가 범위가 과도하지 않은지?	설치가 요구되는 컴포넌트 수가 많음 별도의 대용량 Object-storage 필요	규모에 따라 single/cluster version을 선택 single version은 Promethues를 대체

4.3.1 Thanos 검토

DEVIEW 2021

Problems

- Thanos 적용 전 대비 쿼리 응답 속도에 대한 차이가 거의 없음
- 대부분의 Prometheus HTTP API를 지원하지만 Federate API를 미지원
  - thanos grpc -> http federate로 변환하는 adapter를 별도로 개발
- Thanos의 컴포넌트 수가 많아서, Namespace별로 제공되면,
  전체 pod 수가 비효율적으로 증가할 가능성

4.3.1 Thanos 검토

Problems

- Thanos 메모리 과점유 현상
  - 긴 time-range의 쿼리 경우,
    object-storage에서 받아야하는 timeseries block이 많아지는데,
    이때 object-storage 앞단의 thanos-store-gateway에서 OOM 발생
  - time, block_id 단위로 요청/응답을 분할할 수 있지만,
    단순 몇 개의 block 사이즈가 크면, 이를 로딩하면서 메모리를 소진
- object-storage 성능에 따라 지연 발생
  - query 종류에 따라 순간 다운로드 받는 용량이 비대해질 수 있음
  - 대용량 모니터링을 위한 대용량 object-storage가 요구됨

4.3.2 Victoria-metrics 검토

DEVIEW 2021

Problems

- 초기 loading시 운영에 필요한 resource의 두배 가량 사용
  - 기존 리소스 사용률이 높다면, 서버 재시작시 oom이 발생
  - Victoria-metrics에서도 50%의 여유분을 가지고 운영하도록 안내
- Downsampling 미지원
  - 지원 예정 https://github.com/VictoriaMetrics/VictoriaMetrics/issues/36
  - 기존처럼 수집 주기를 넓혀서 long-term을 제공

4.3.3 Thanos vs Victoria-metrics

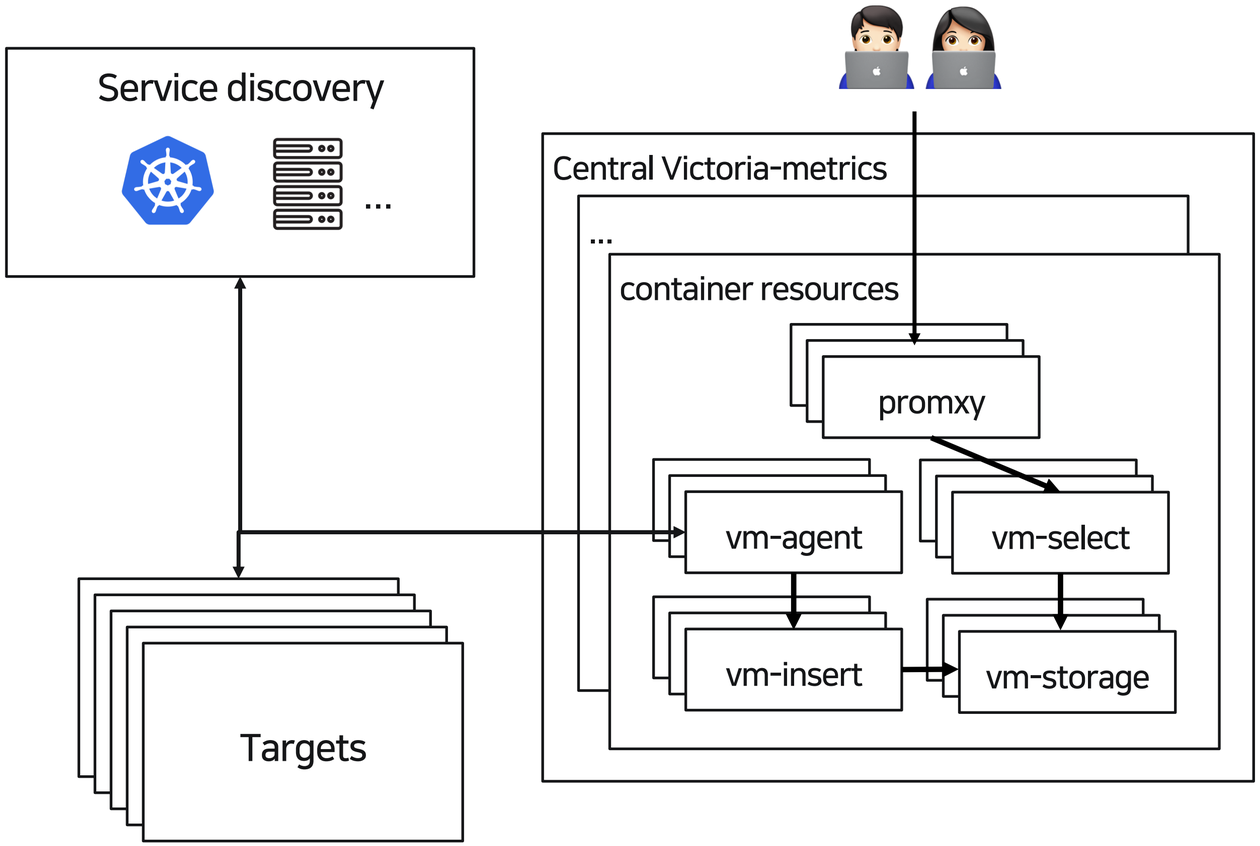
운영상 Thanos 대비 적은 리소스로 안정적이고 빠른 응답 결과 도출

- Cluster 테스트시 thanos(prometheus)대비 vm이 cpu/memory를 약 40%/23% 정도 적게 사용
- 다양한 케이스에 대한 쿼리 테스트를 진행, 아래 약 26k samples를 쿼리한 케이스는 각 특징을 잘 보여줌
  - thanos(prometheus)는 초기(캐시 전) 응답 시간이 매우 지연되지만 캐시 이후 응답은 빠름
  - vm도 캐시전 지연이 있지만 thanos 케이스처럼 지연이 과도하지 않음

약 26k samples를 5회 요청했을 때					
Thanos	15s	331ms	55ms	81ms	76ms
vm	1s	268ms	211ms	206ms	223ms

4.3.4 Victoria-metrics

기존 모니터링 구조를 대체하고 확장 가능한 구조로 제공

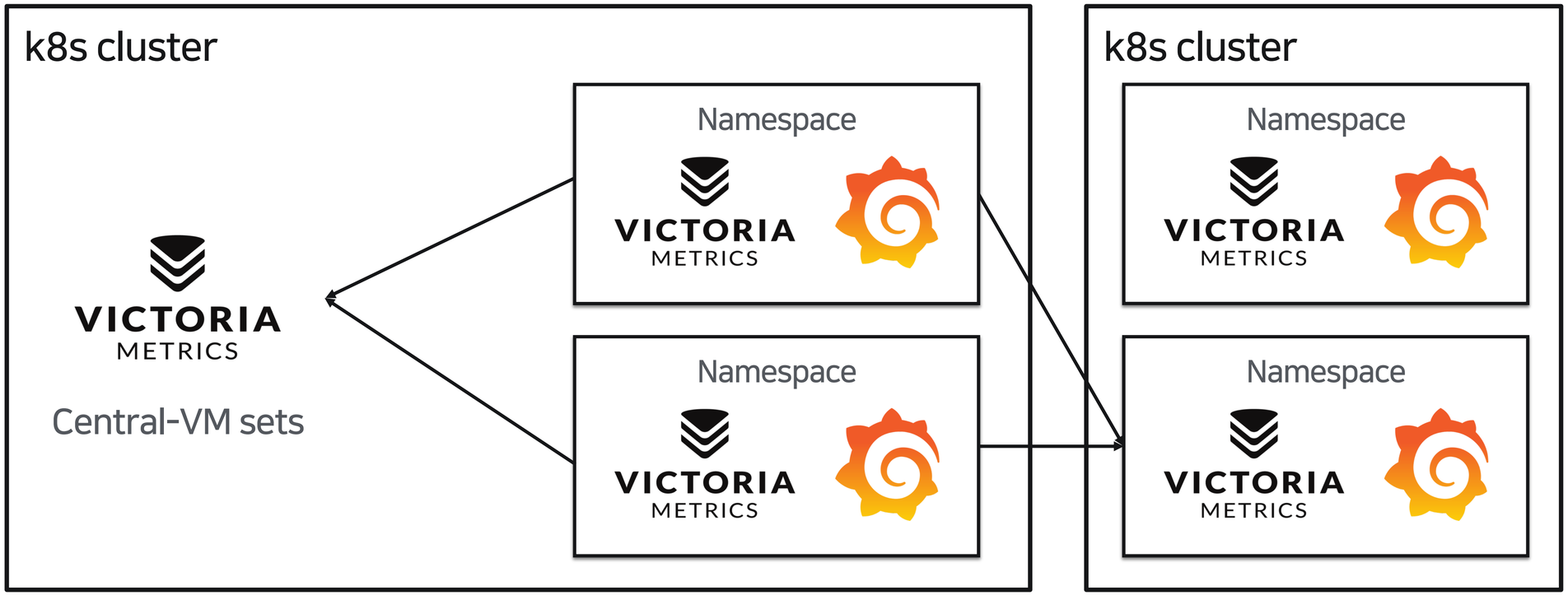


- Dynamic Target Discovery
- PromQL, MetricsQL
  - 확장된 형태의 쿼리 제공
- tsdb
- 개별 layer가 분리되어 각 컴포넌트를 확장할 수 있는 구조
- 상대적으로 빠른 응답 속도
  - 운영에서는 확장 가능한 형태로 적용했는데,
    사용자에도 좀 더 빠른 single-version을 제공할 수 있음

4.4 사용자 관점의 분산 처리

사용자 모니터링도 분산 처리가 가능할까?

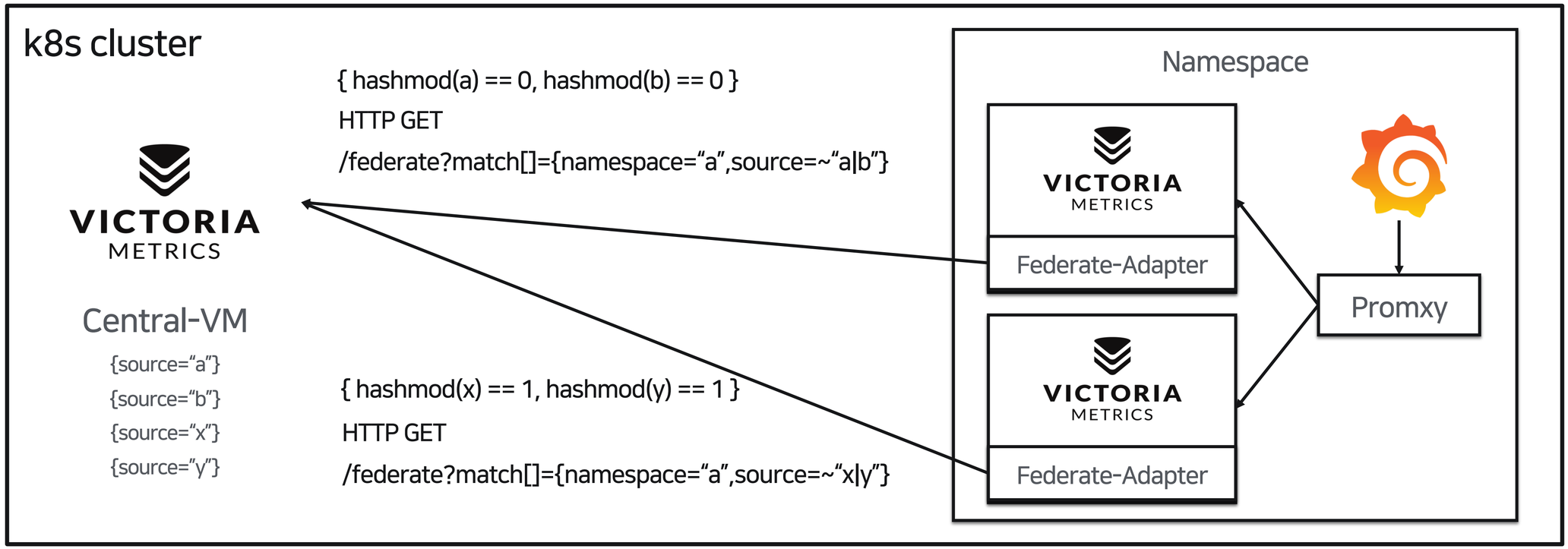
- 사용자 수집 target은 Central-VM 하나라서, target을 분산하는 방법으로 분산이 안되는 문제
- 여러 클러스터에 쿼리할 수 있는 설정을 어떻게 제공할 수 있을까?



4.4.1 사용자 분산 수집

사용자 수집도 수평 분산 확장 가능한 구조로 제공

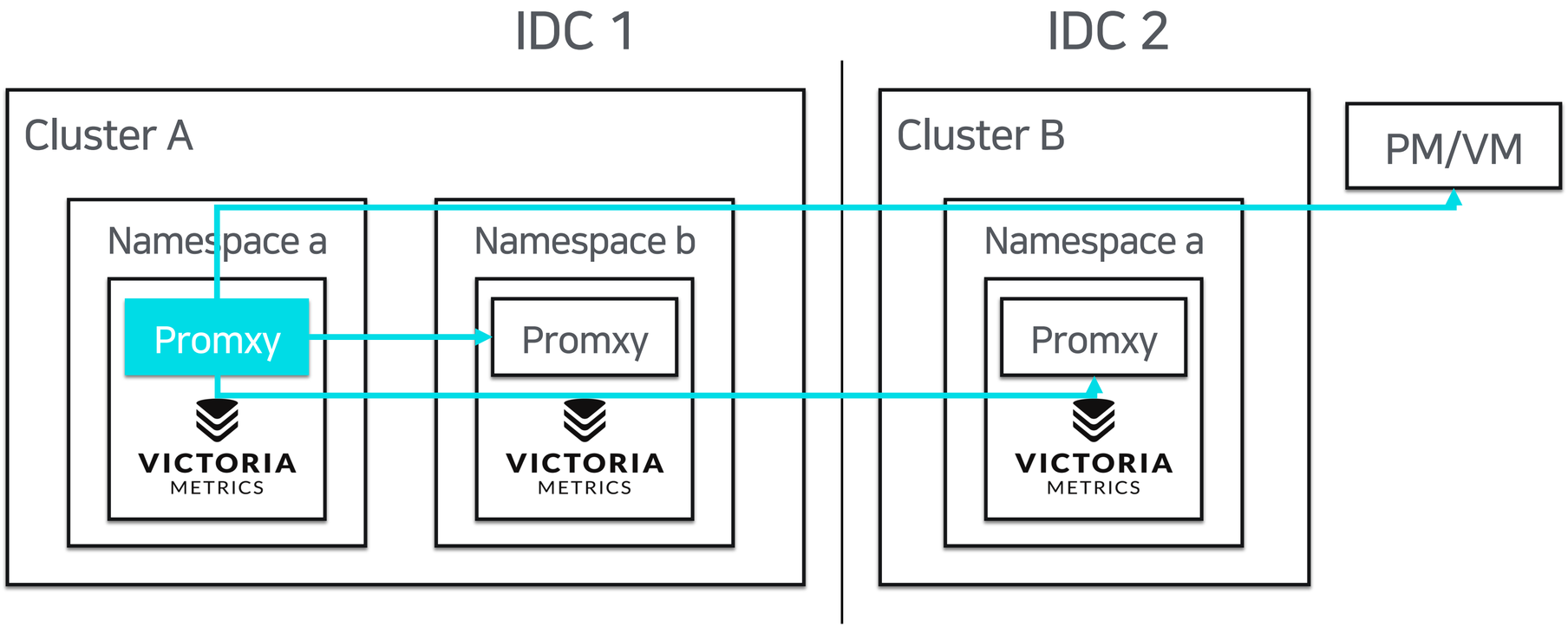
- 특정 label을 기준으로 federate label-matcher를 구성하는 adapter 구현
- 주기적으로 label을 확인하고 hashmod 연산하여 동적으로 분산



4.4.2 사용자 분산 쿼리

Query datasource endpoint를 제공하고 상호 연동할 수 있도록 제공

- Multi IDC/Cluster/PM/VM에 대한 Query Aggregation 제공

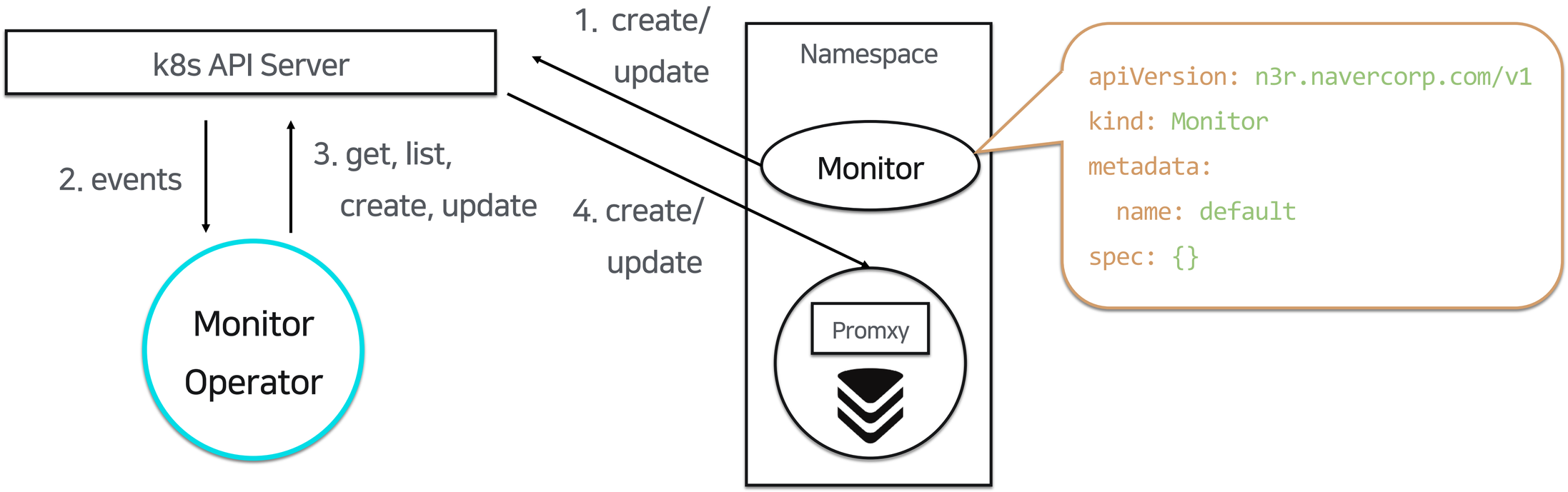


4.4.3 사용자 분산 제공 자동화 - Operator

DEVIEW 2021

사용자향 Monitoring Set을 동적으로 Provisioning

- Monitor custom resources를 생성 및 편집
- Operator를 통해 create/update event를 받아서 Monitoring Set을 제공

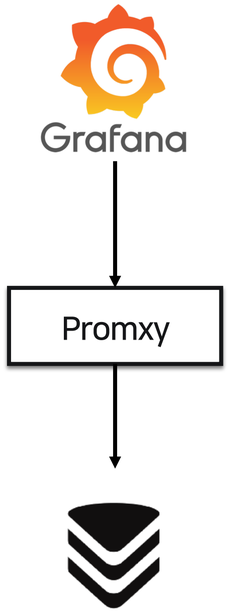


4.4.4 사용자 모니터링 구성

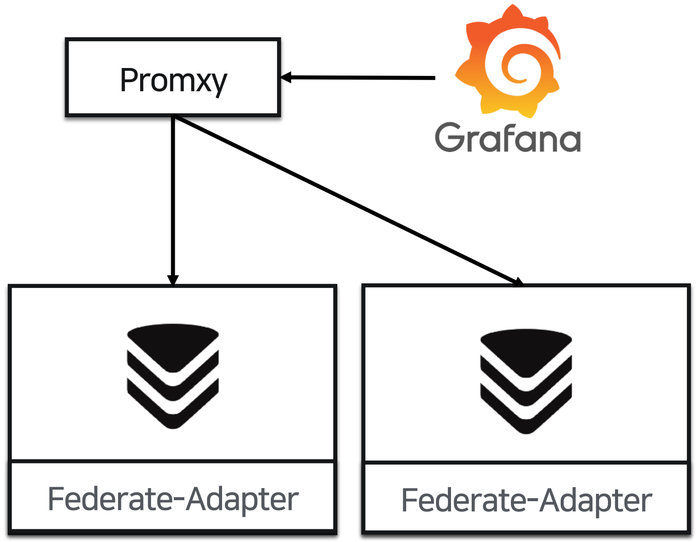
사용자향 Monitoring set을 확장 가능한 형태로 제공

- 기본 제공되는 형태로부터 분산 및 long-term 확장 제공

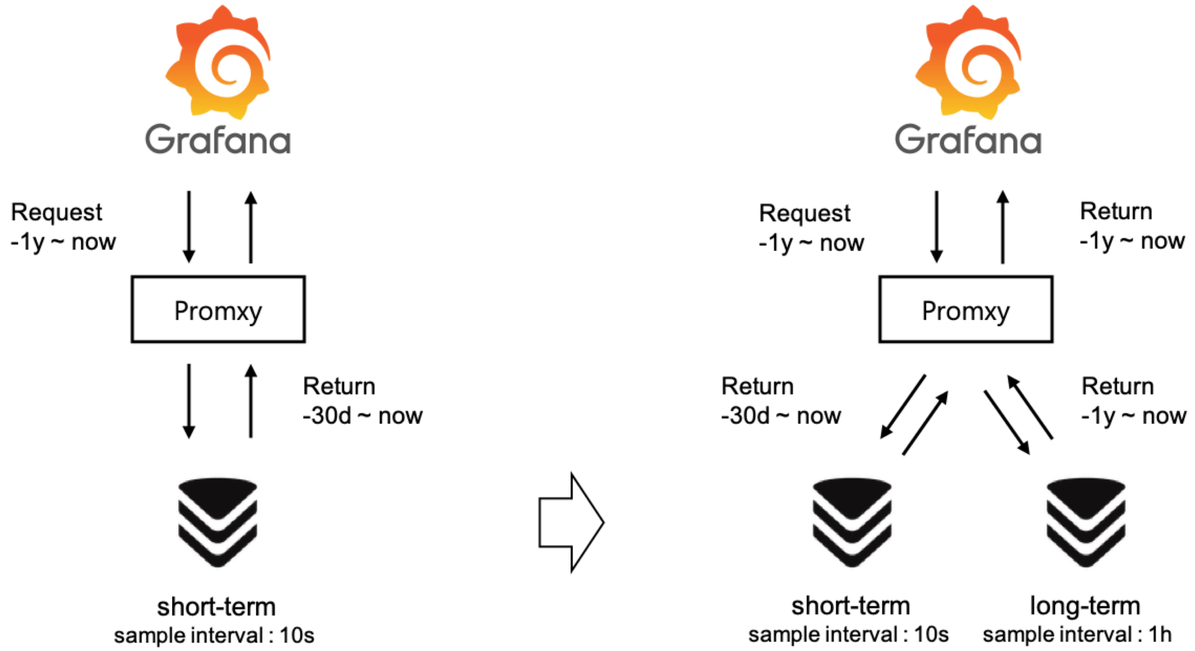
Basic



Extended



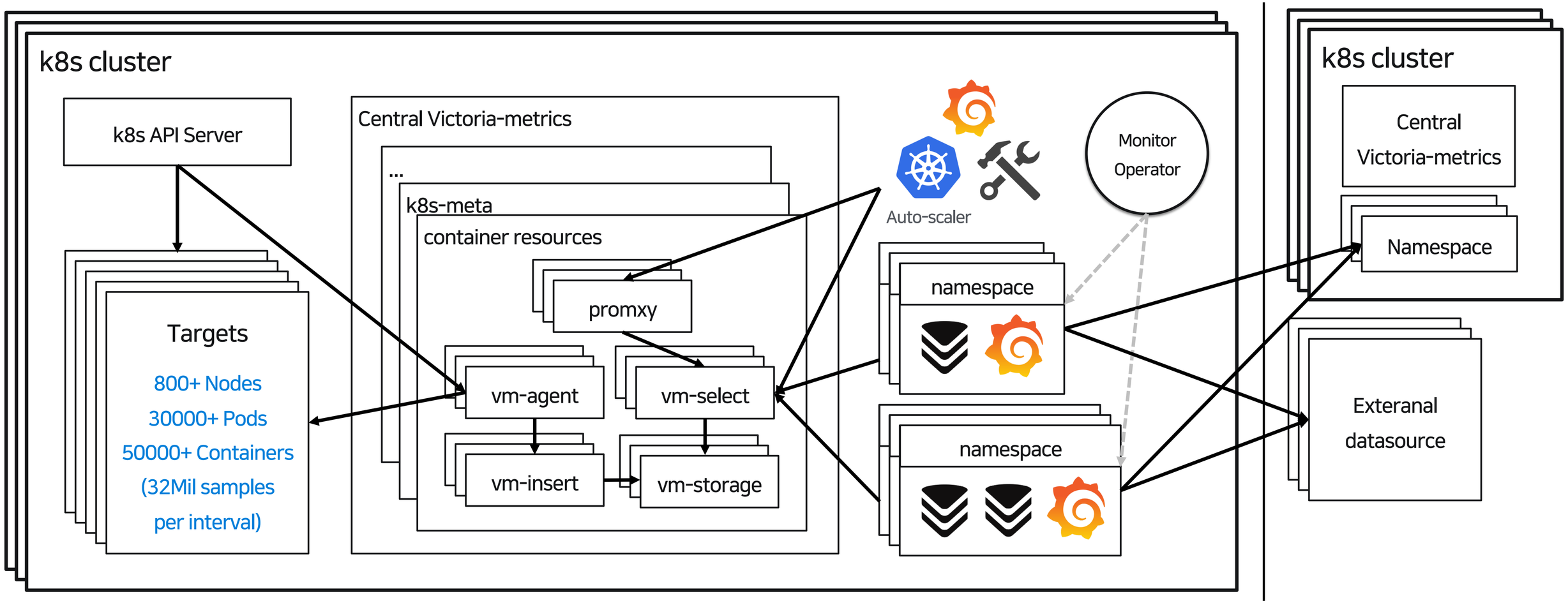
Long-term

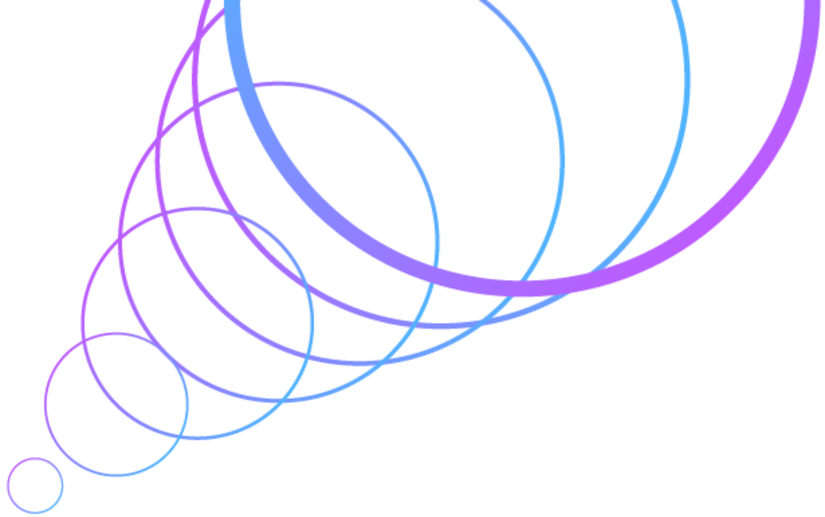


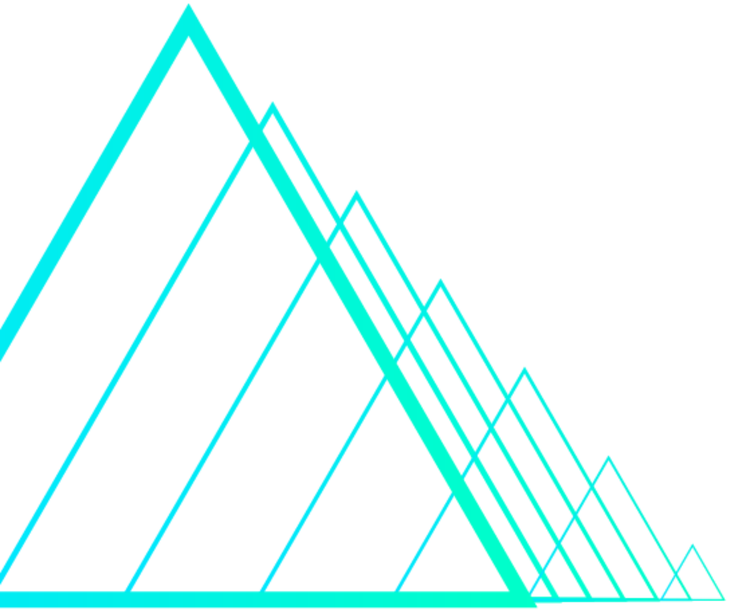
4.5 Distributed Container Monitoring

DEVIEW 2021

2021.10 기준 - 25+ Clusters, 100000+ Pods, 4400+ Namespaces에 대한 지표 제공







Q & A

Thank You