멀티클러스터에서 가시성 구축 사례 (feat. Thanos)

임성일

- Software Developer @ SK Telecom
- TACO development
  - LMA deployment
  - GitOps tools
  - Micro serivces
- OSS Contributor (~2019)
  - Openstack
  - Fluentbit
- eMail: si.im@sk.com, usnexp@gmail.com

Contents

Prometheus

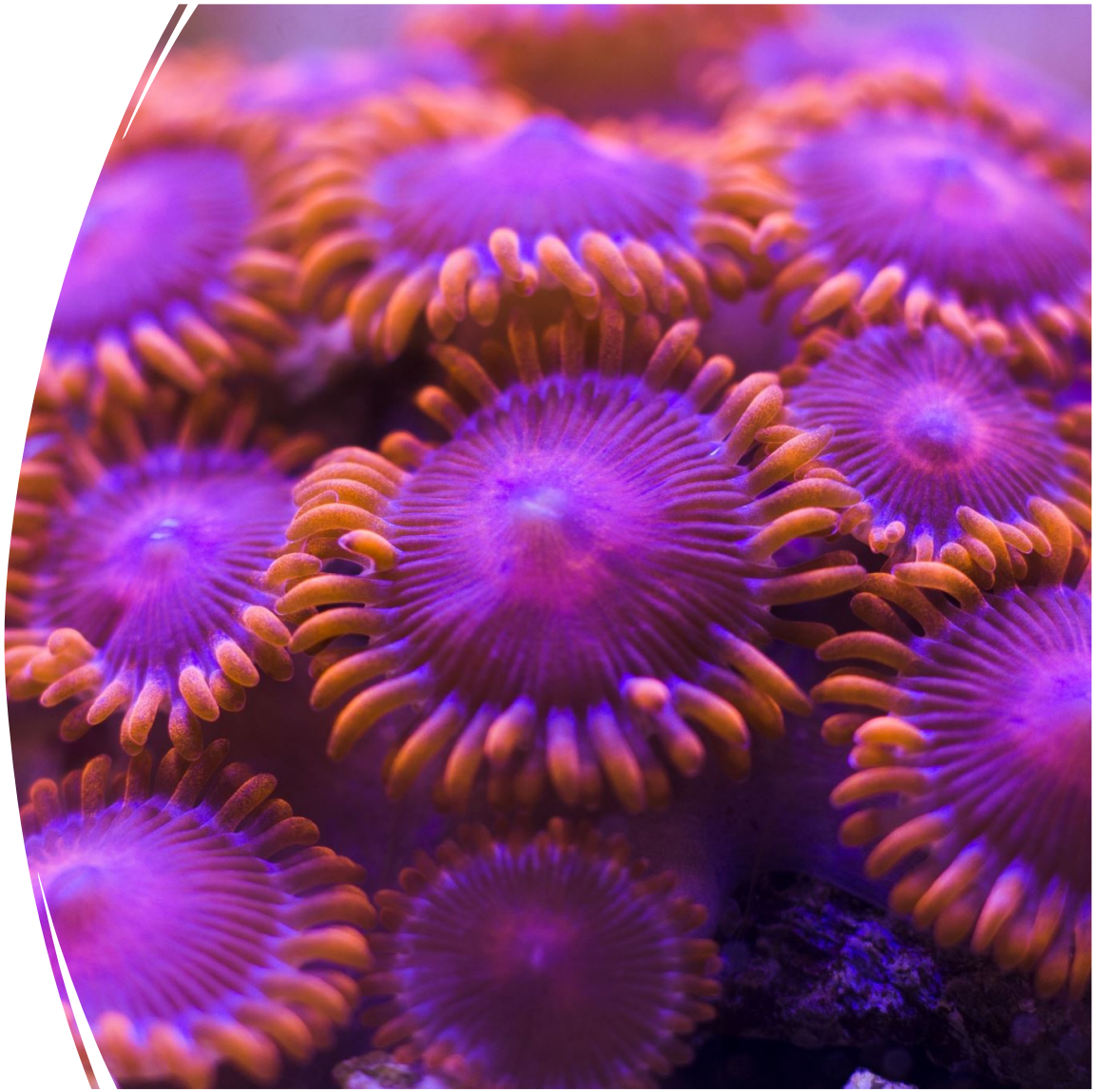
Federation

Thanos

component

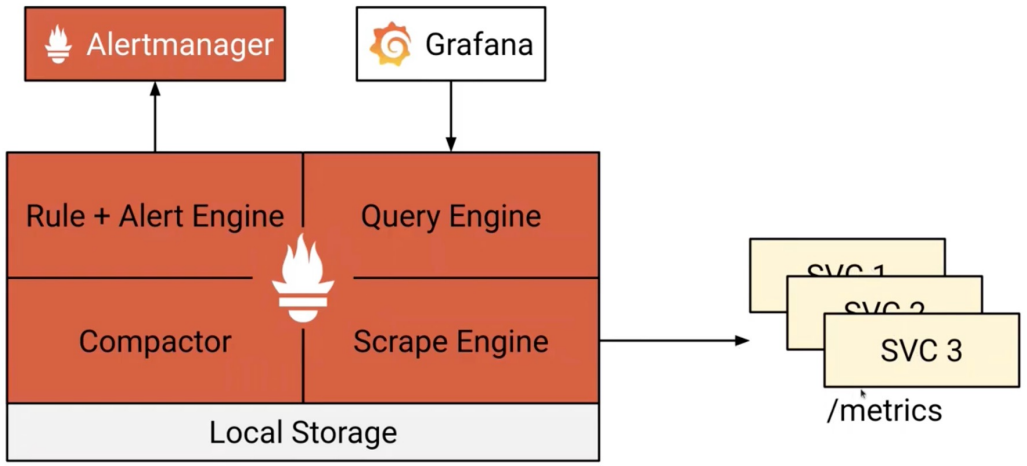
Thanos 적용사례 (TACO)

Prometheus



Prometheus?

- ***a multi-dimensional data model with time series data*** identified by metric name and key/value pairs
- PromQL, a ***flexible query language*** to leverage this dimensionality
- no reliance on distributed storage; single server nodes are autonomous
- time series collection happens via a ***pull model over HTTP***
- pushing time series is supported via an ***intermediary gateway***
- targets are discovered via service discovery or static configuration
- multiple modes of graphing and dashboarding support

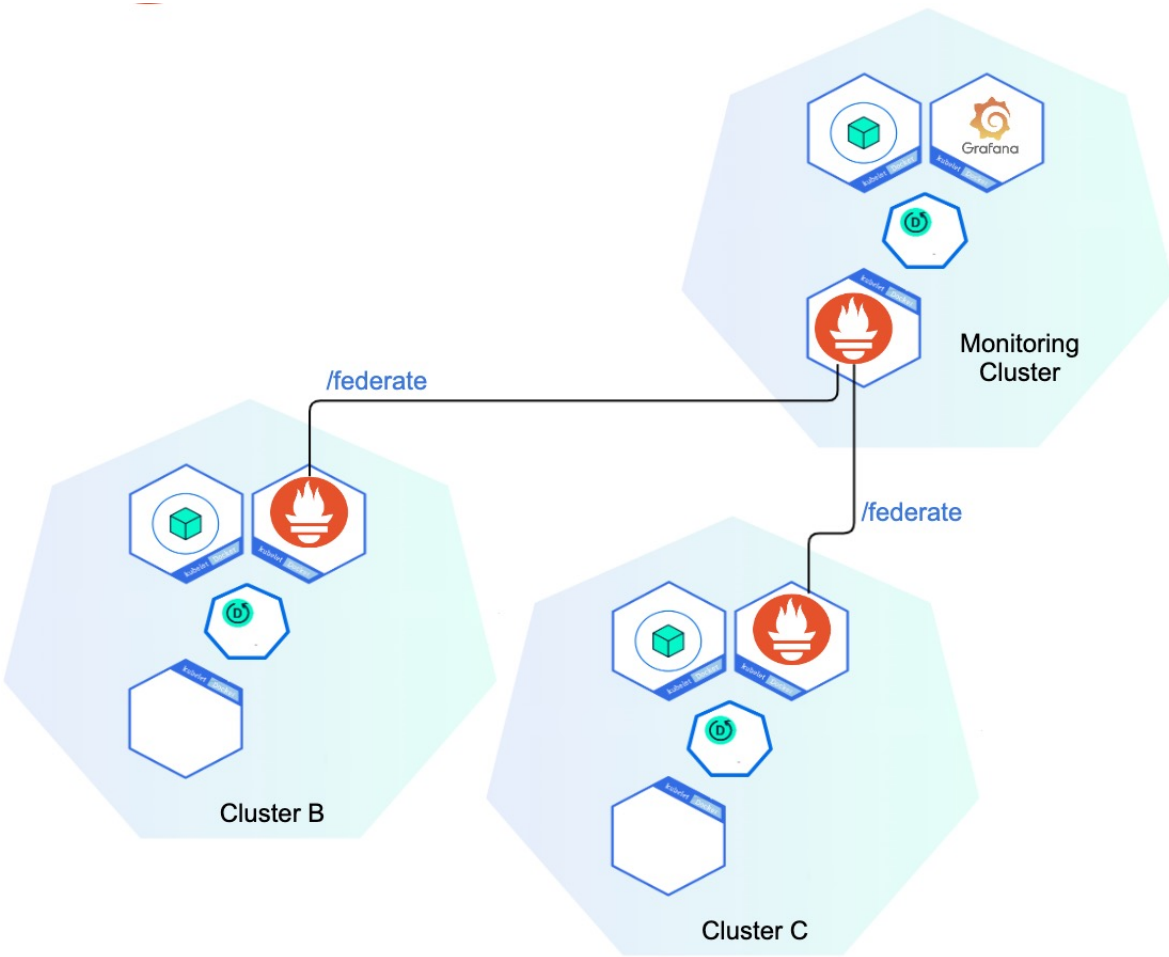


출처: prometheus.io

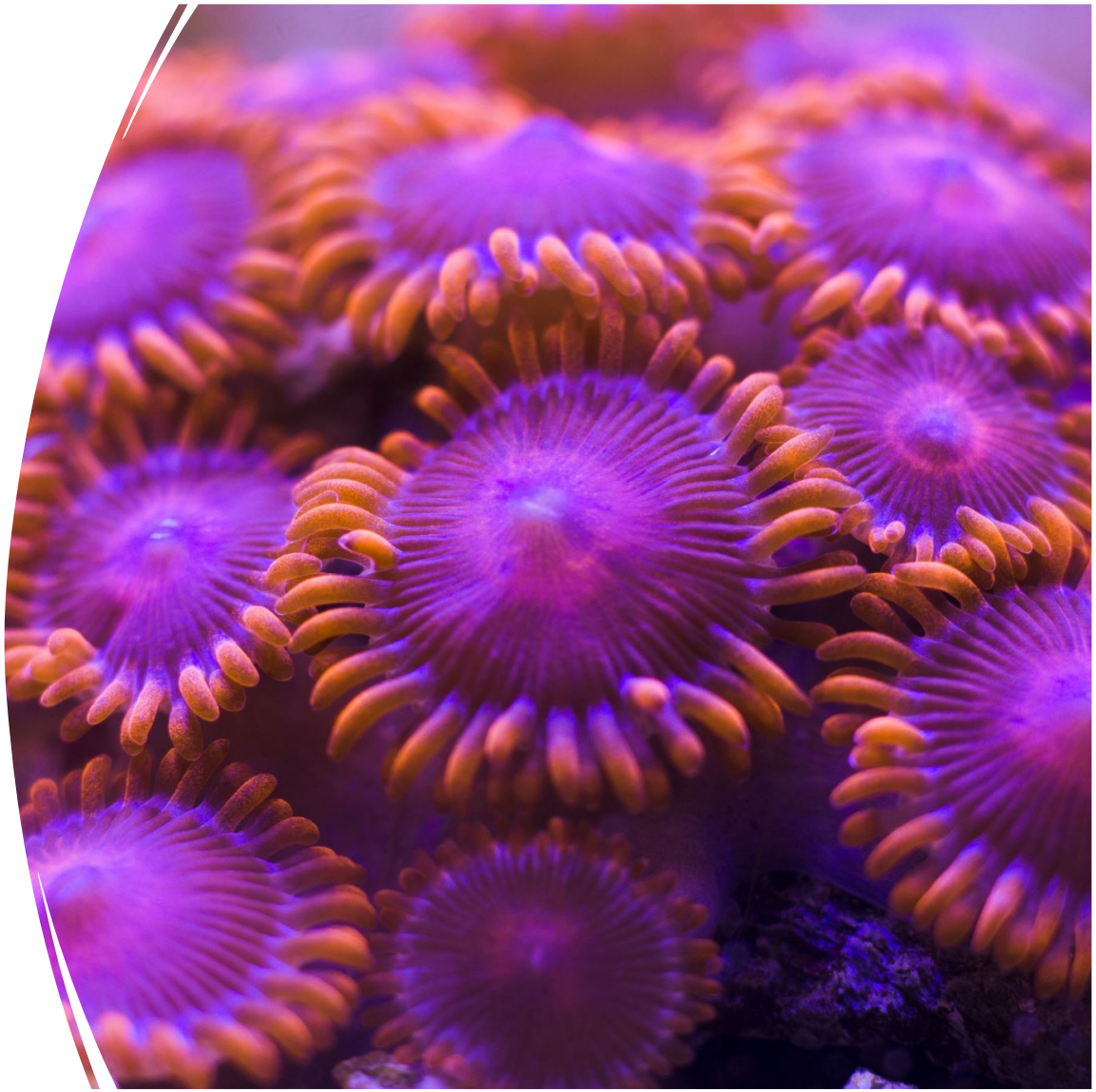
TACO-LMA의 멀티클러스터

(feat. Prometheus federation)

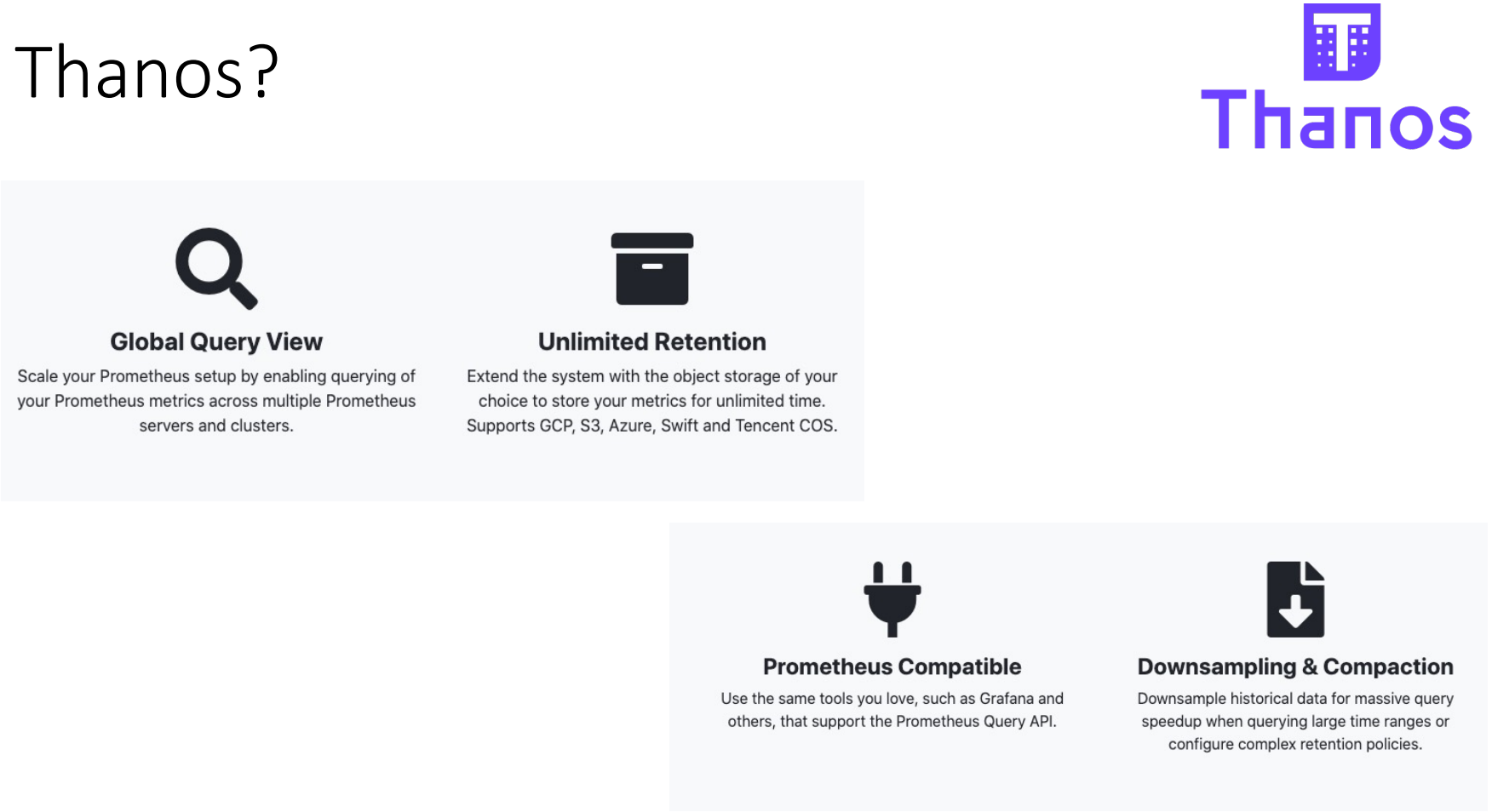
- 클러스터별 수집용 Prometheus 배포
- 모든 클러스터의 모니터링 정보를 모으는 전용 클러스터 구성 (논리적/물리적)
- Prometheus의 federation 기능 활용



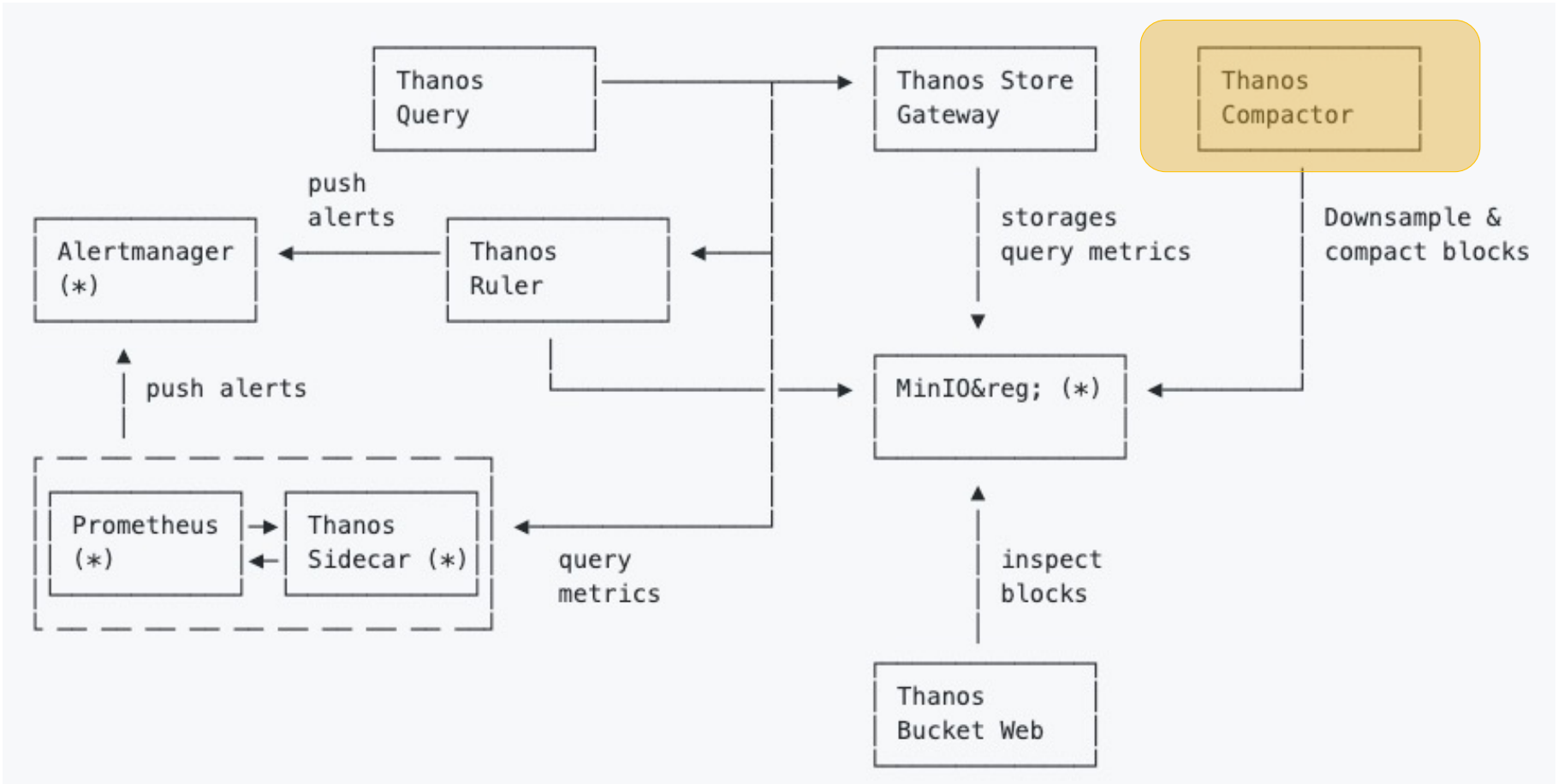
Thanos



Thanos?

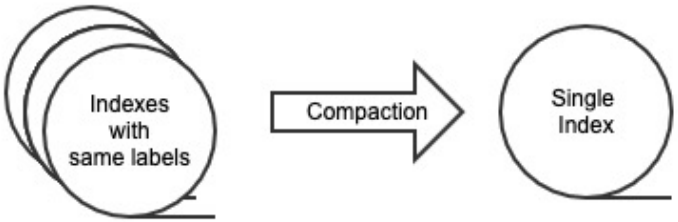


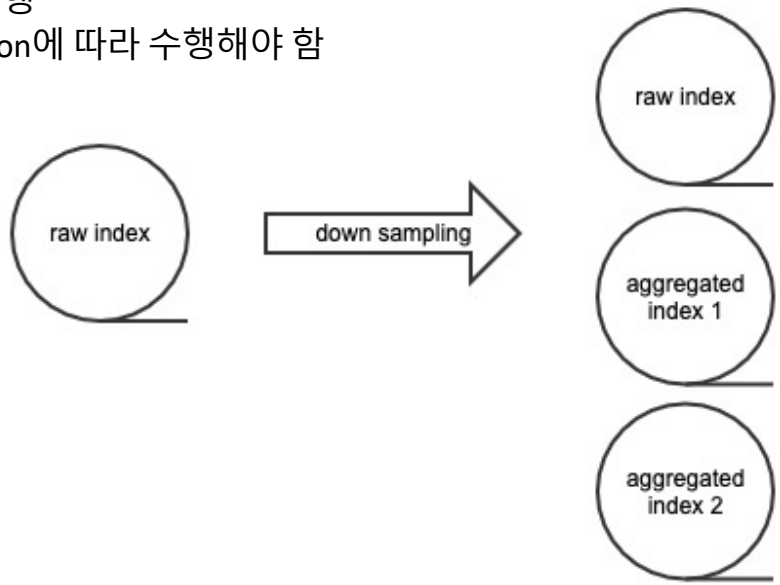
출처: thanos.io


Thanos
Query
Thanos Store
Gateway
Thanos
Compactor
push
alerts
Alertmanager
(*)
Thanos
Ruler
storages
query metrics
Downsample &
compact blocks
push alerts
MinIO® (*)
Prometheus
(*)
Thanos
Sidecar (*)
query
metrics
inspect
blocks
Thanos
Bucket Web

Thanos Compactor

- prometheus 2.0 저장소 엔진(storage engine)의 compaction 과정에 적용되는 Thanos의 명령어 compact
- Compaction
  - 객체저장소(Obj storage)에 저장된 블록 데이터를 대상으로 수행
  - 완전한 동시성을 제공하지 않으므로 개별 버켓에 대한 singleton에 따라 수행해야 함
- Downsampling
  - 시간범위에 따라 정확도를 유지하면서 통합된 데이터를 저장
  - 요청시 활용하여 질의성능 향상

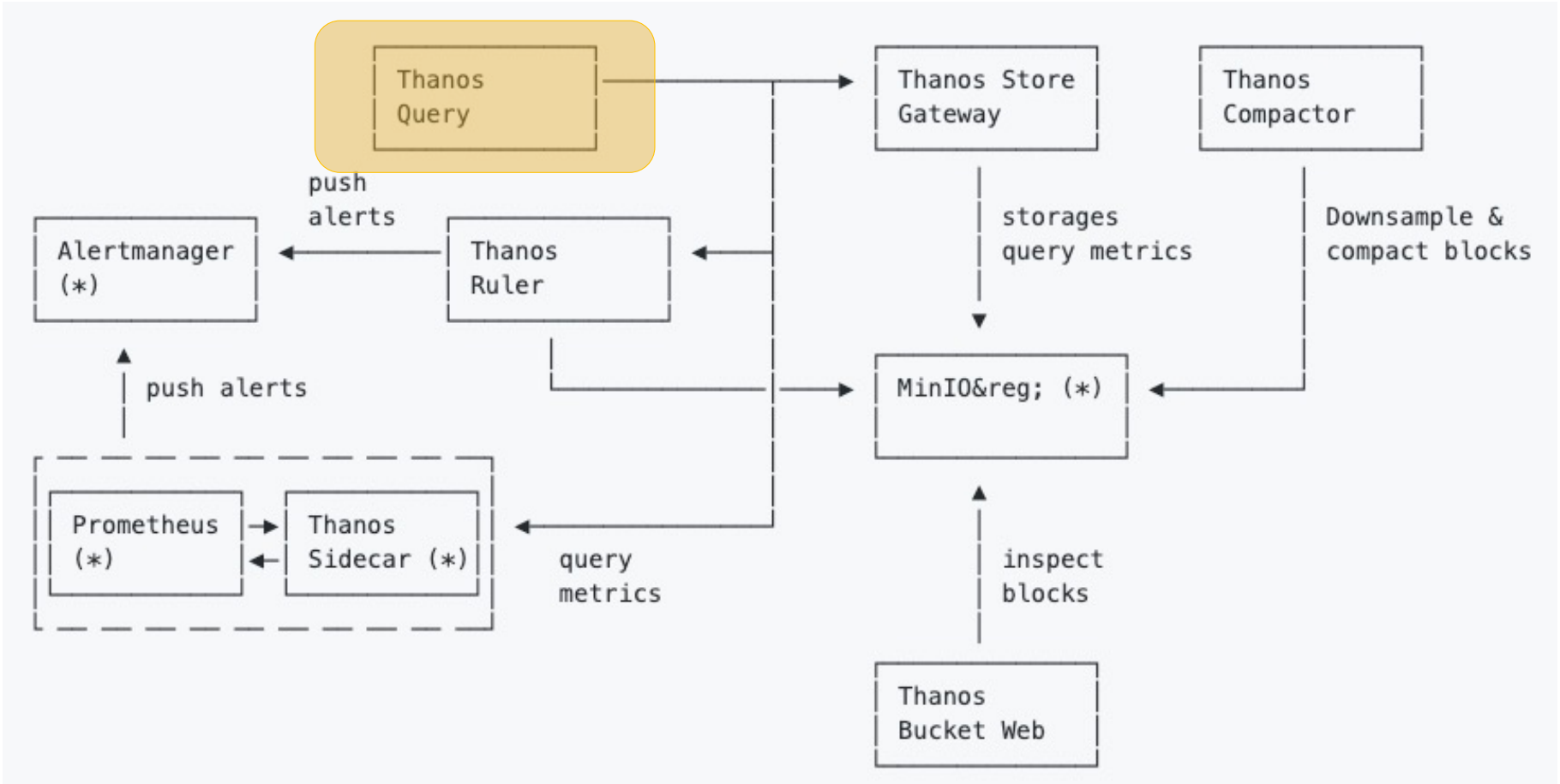




Thanos Compactor

- 주요활용방법
  - rollout: 수집된 데이터를 기준에 따라 장기보관용 데이터로 전환
  - rollover: aggregated 데이터 생성
  - deduplication: 동일한 데이터에 대한 복수의 수집기를 활용하고 복수제거를 사용하여 가용성을 향상(H.A.)
  - 비동기 동작: cron 또는 daemon형태로 사용
- 리소스 사용관련 주의사항
  - 원격 저장소의 대용량의 데이터를 다루는 작업으로 메모리와 네트워크의 사용량이 많음
  - 실행주기와 동시진행 작업 수 등을 조정이 필요

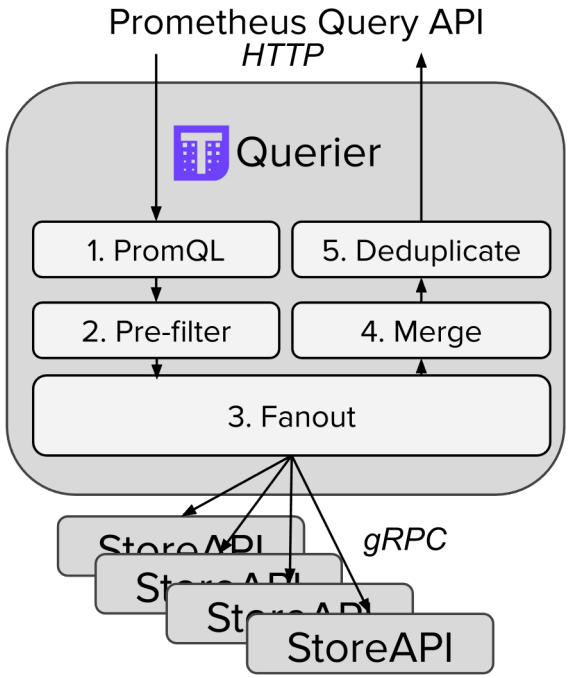
```
thanos compact --data-dir /tmp/thanos-compact \
--objstore.config-file=bucket.yml \
--retention.resolution-raw=10d \
--retention.resolution-5m=30d \
--retention.resolution-1h=365d \
--compaction.concurrency=2
```


Thanos
Query
Thanos Store
Gateway
Thanos
Compactor
push
alerts
Alertmanager
(*)
Thanos
Ruler
storages
query metrics
Downsample &
compact blocks
push alerts
MinIO® (*)
Prometheus
(*)
Thanos
Sidecar (*)
query
metrics
inspect
blocks
Thanos
Bucket Web

Thanos querier/query

- 다양한 backend의 데이터에 질의
  - Prometheus (see Sidecar)
  - Object Storage (see Store Gateway)
  - Global alerting/recording rules evaluations (see Ruler)
  - Metrics received from Prometheus remote write streams (see Receiver)
  - Another Querier (you can stack Queriers on top of each other)
  - Non-Prometheus systems! (e.g OpenTSDB)
- Run-time deduplication of HA groups

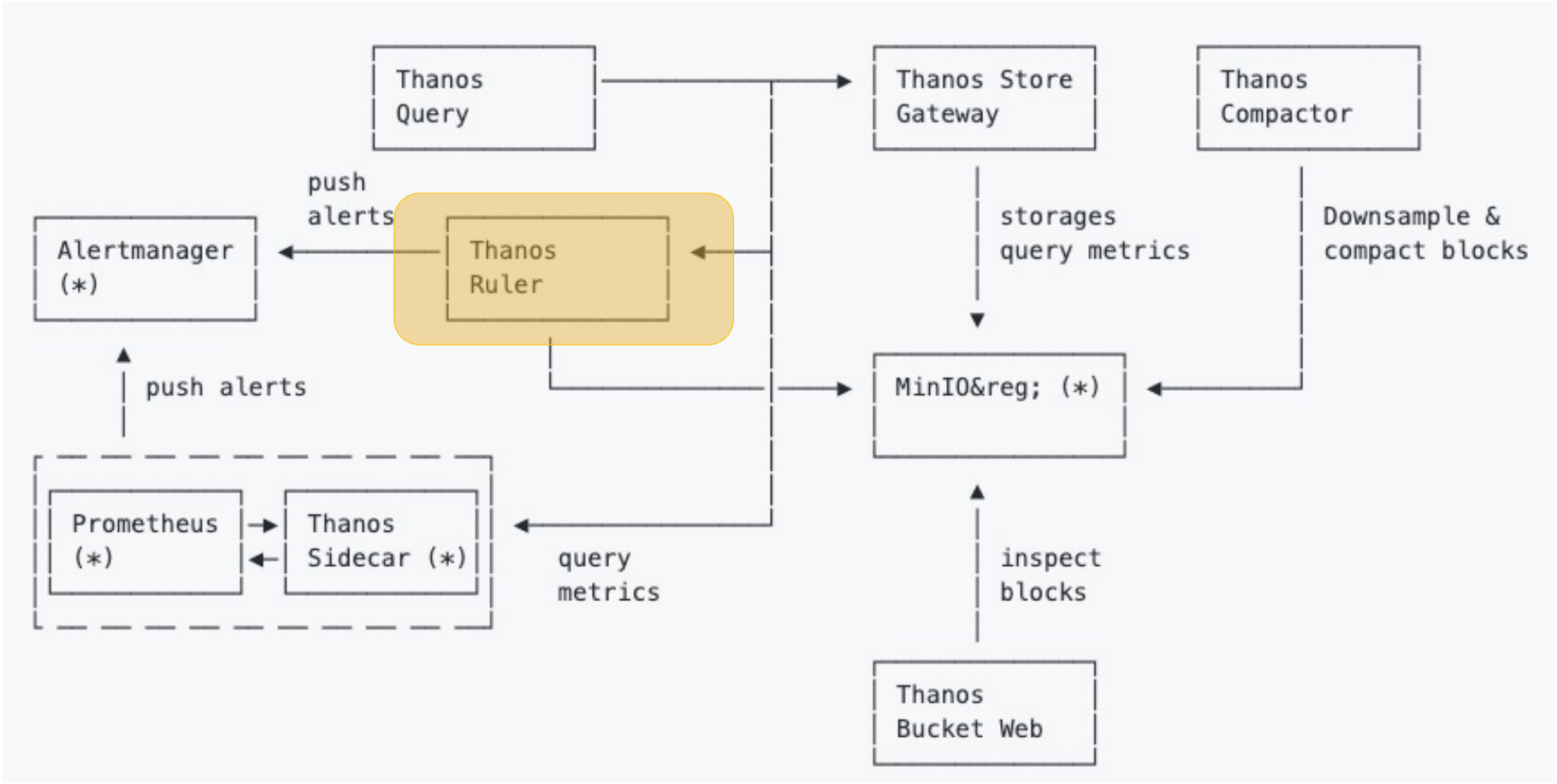
```
thanos query \
    --http-address       "0.0.0.0:9090" \
    --query.replica-label "replica" \
    --store              "<store-api>:<grpc-port>" \
    --store              "<store-api2>:<grpc-port>"
```



Thanos query frontend

- Thanos Queriers의 앞단에 읽기 성능 향상을 위한 서비스 제공
- Cortex Query Frontend 차용
- 주요기능
  - Splitting
  - Retry
  - Caching (in-memory, memcached)
  - Slow query logging
- 주요 활용
  - 수평적 읽기성능 확장
  - Safeguard

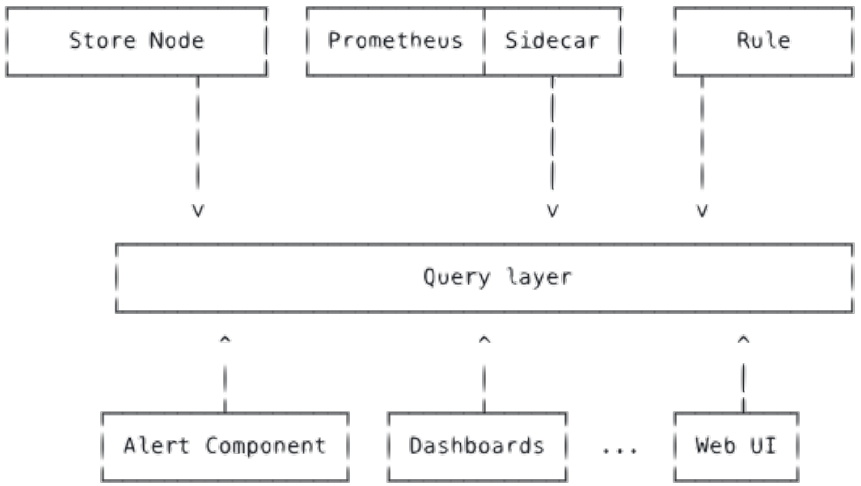
```
thanos query-frontend \
   --http-address     "0.0.0.0:9090" \
   --query-frontend.downstream-url="<thanos-querier>:<querier-http-port>"
```

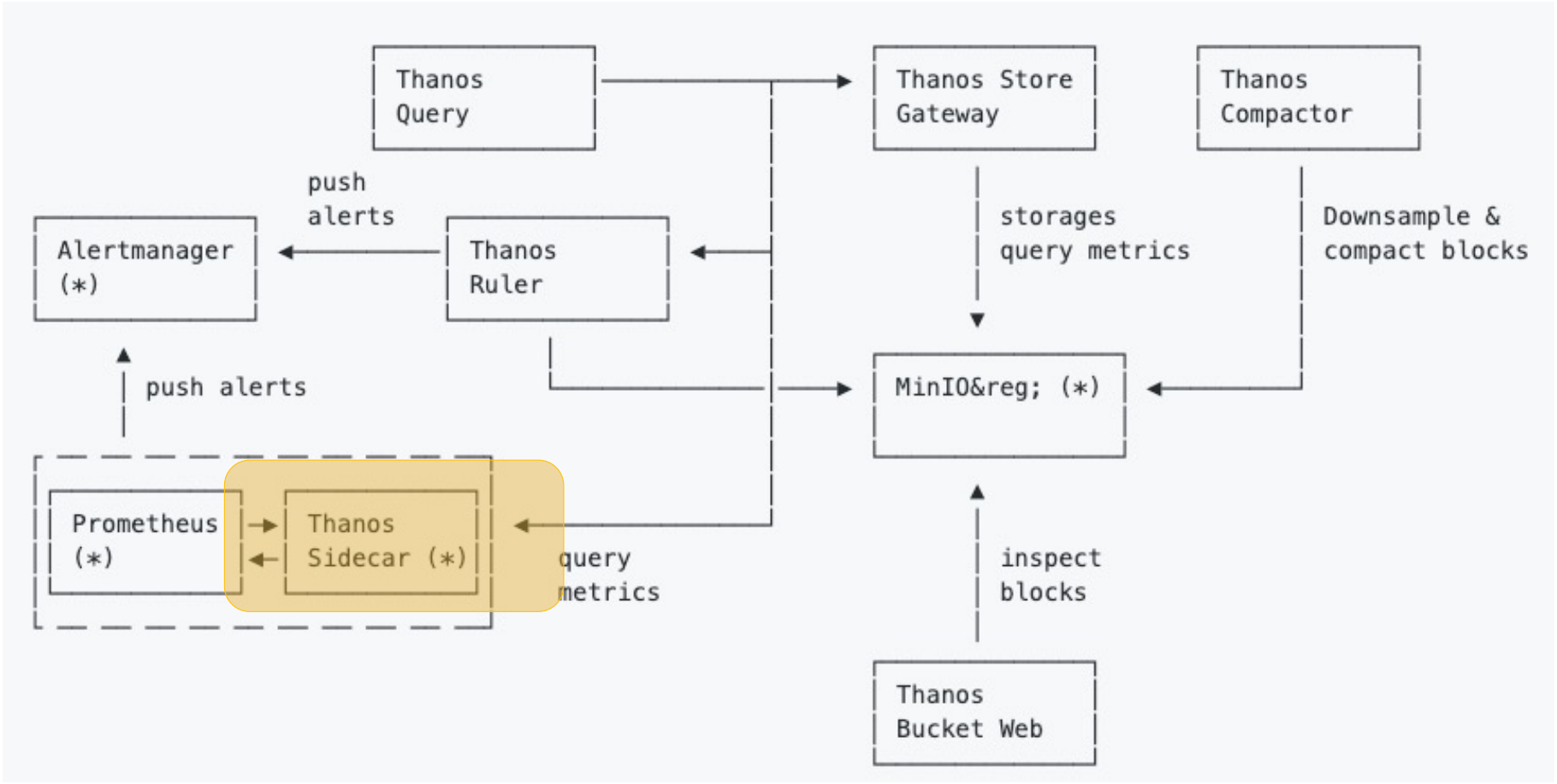

Thanos
Query
Thanos Store
Gateway
Thanos
Compactor
push
alerts
Alertmanager
(*)
Thanos
Ruler
storages
query metrics
Downsample &
compact blocks
push alerts
MinIO® (*)
Prometheus
(*)
Thanos
Sidecar (*)
query
metrics
inspect
blocks
Thanos
Bucket Web

Thanos Ruler

- 실시간 rule처리
  - Alerting rule: 알람발생
  - Recording rule: raw metric을 저장하는 방법지정
- 주의: 지속적으로 query를 사용하므로 성능에 영향이 발생

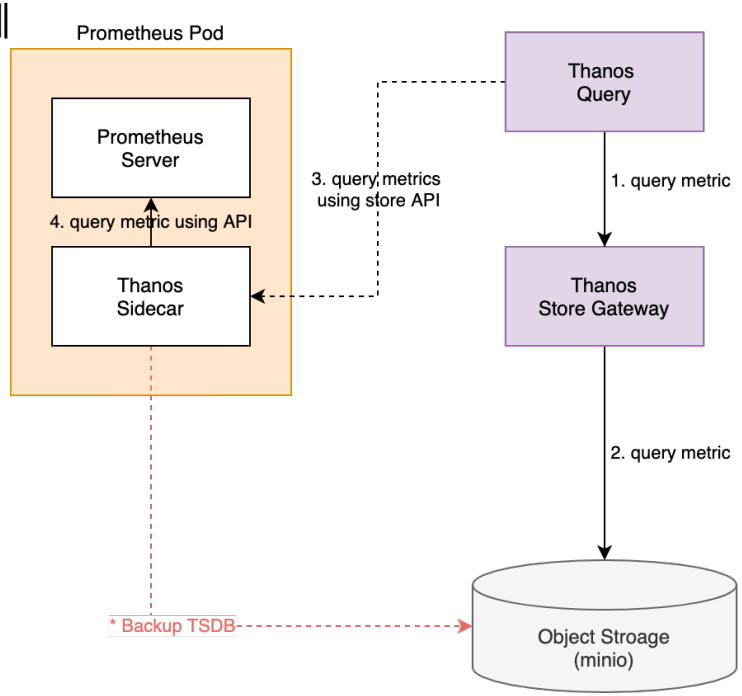
```
thanos rule \
    --data-dir             "/path/to/data" \
    --eval-interval        "30s" \
    --rule-file            "/path/to/rules/*.rules.yaml" \
    --alert.query-url      "http://0.0.0.0:9090"\
    --alertmanagers.url    "http://alert.thanos.io" \
    --query                "query.example.org" \
    --query                "query2.example.org" \
    --objstore.config-file "bucket.yml" \
    --label                'monitor_cluster="cluster1"'
    --label                'replica="A"'
```

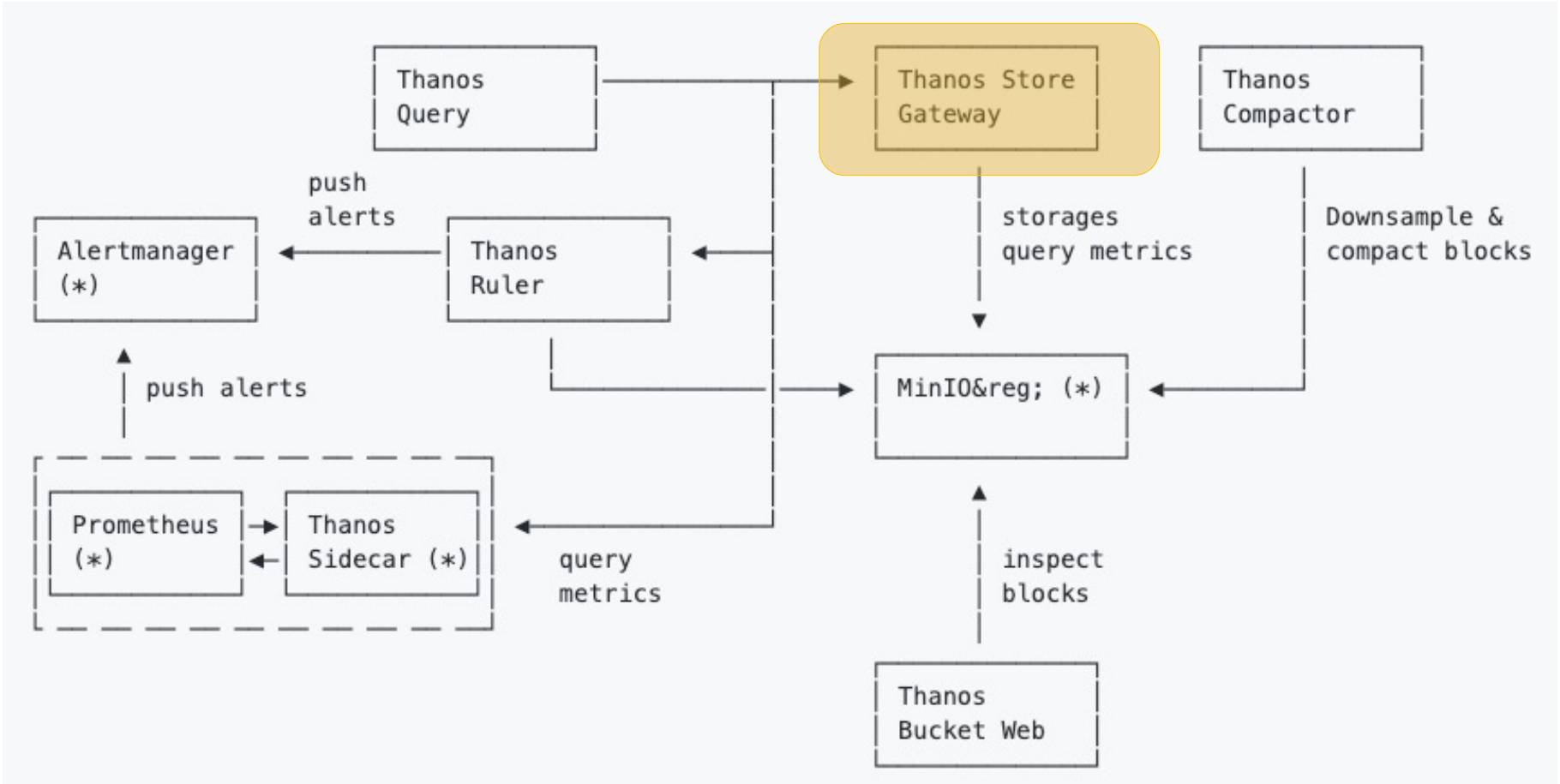



Thanos
Query
Thanos Store
Gateway
Thanos
Compactor
push
alerts
Alertmanager
(*)
Thanos
Ruler
storages
query metrics
Downsample &
compact blocks
push alerts
MinIO® (*)
Prometheus
(*)
Thanos
Sidecar (*)
query
metrics
inspect
blocks
Thanos
Bucket Web

Thanos Sidecar

- 주요기능
  - 백업: 일정 시간마다 tsdb데이터를 thanos object storage에 저장
  - 질의: Thanos Querier(Query)가 StoreAPI를 호출하여 Prometheus (local)에 질의
- 참고: 데이터 누락을 방지하기 위한 --min-time 설정
  - store-api를 통해 요청온 내용에 대한 처리에 대한 지침으로 -2h를 설정하면 2시간 전까지의 데이터는 처리
  - sidecar가 2시간에 한번씩 2시간 데이터를 object store에 기록하고 compactor는 30분의 대기 후 (consistencyDelay) 처리하므로 2:30 이상 설정해야 함

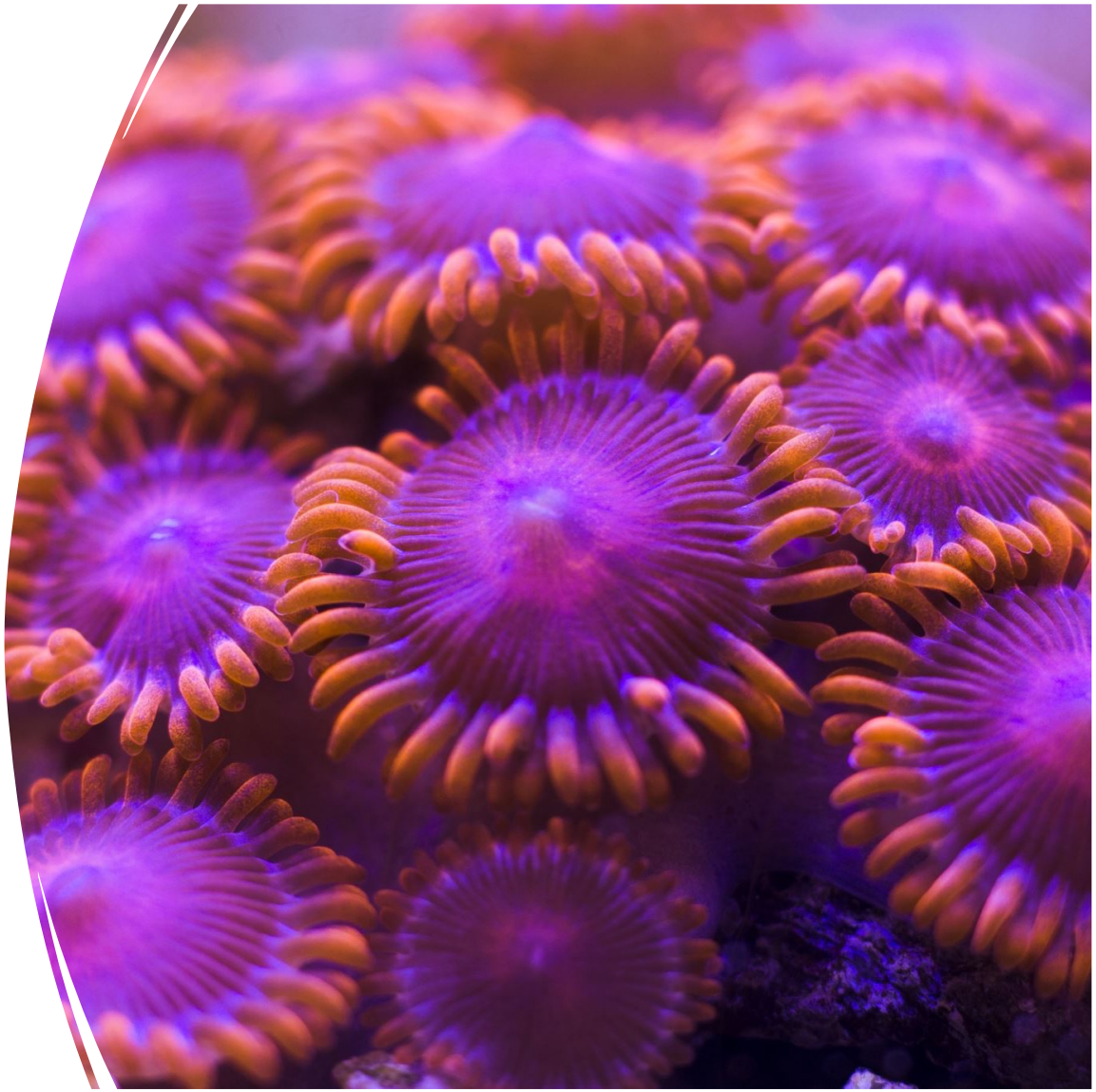



Thanos
Query
Thanos Store
Gateway
Thanos
Compactor
push
alerts
Alertmanager
(*)
Thanos
Ruler
storages
query metrics
Downsample &
compact blocks
push alerts
MinIO® (*)
Prometheus
(*)
Thanos
Sidecar (*)
query
metrics
inspect
blocks
Thanos
Bucket Web

Thanos Store Gateway

- Store API를 제공하며 이를 통해 오브젝트 스토리지 버킷에 저장된 Long-term 데이터를 접근하기 위한 서비스를 제공
- 기능
  - Shading / Partitioning: 시간, 레이블기반
  - Caching: index cache
    - In-memory, memcached

전환사례



TKS
(Taco Kubernetes Service)

TKS 는

- TACO를 기반으로 **hybrid cloud** 서비스 예정 (2022.1Q)
- 사용자가 TACO(kubernetes) 클러스터를 요청하면 실시간 생성 및 삭제 등 기능제공
  - decapod v2 도 함께 사용한 기술
  - 사용자는 할당된 범위에서 자유롭게 클러스터 생성/사용/삭제

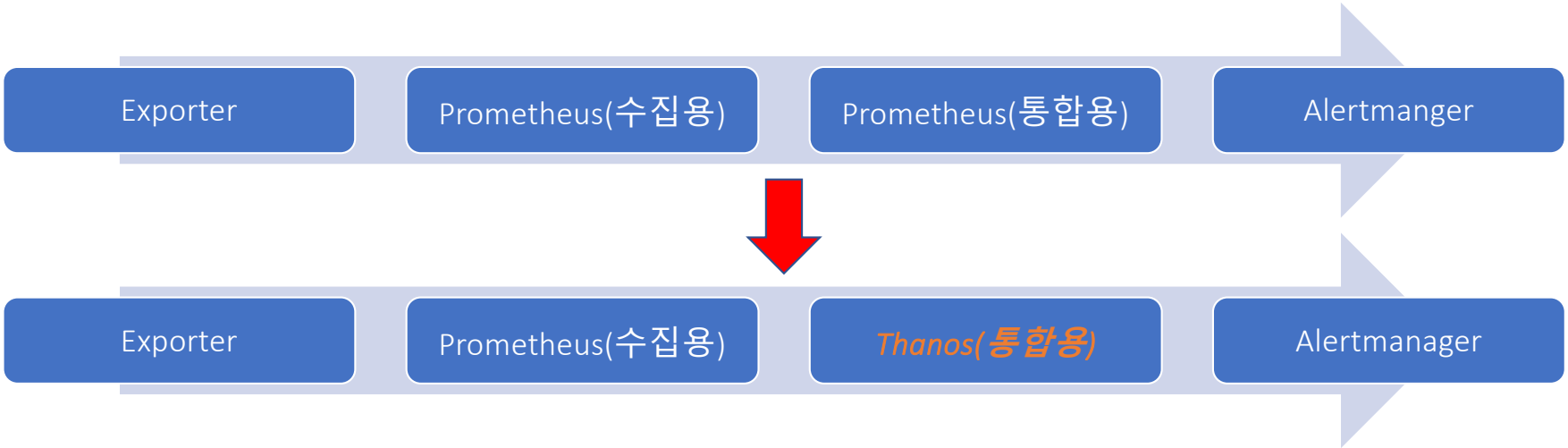
TKS 요건

- 클러스터별 역할(ex. master, admin, manger...)을 부여하지 않고 모두 동일한 구조로 구성 (최소변경요건)
- 사용자는 접근하는 클러스터에 관계없이 동일하게 전체 클러스터의 상태를 확인할 수 있어야 함
- 클러스터에서 발생한 장애가 다른 클러스터로 전파되면 안됨

Thanos 도입

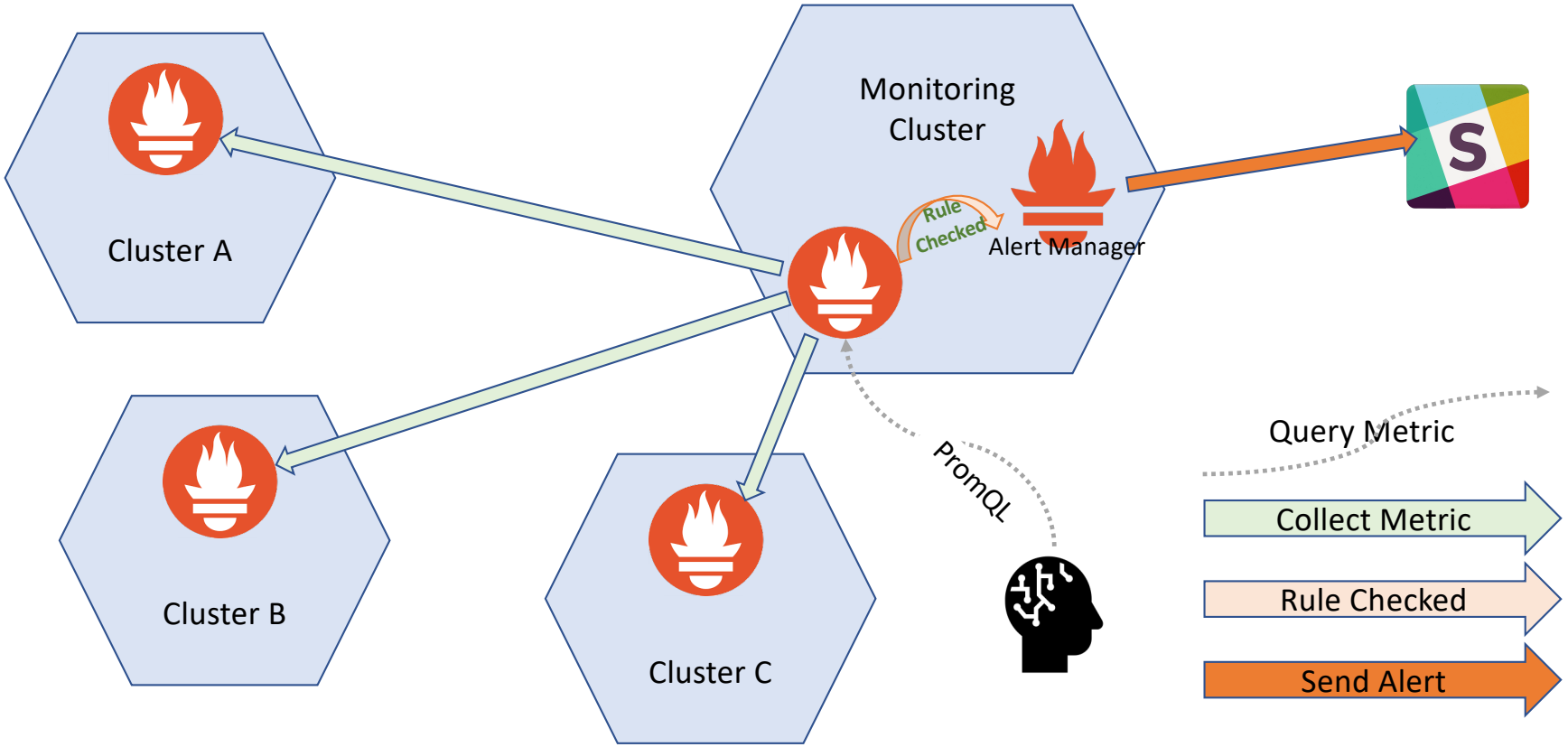
TKS**(Taco Kubernetes Service)** 서비스를 준비

- 멀티클러스터의 멀티 소스에 대한 통합적 가시성확보 필요
  - 클러스터에 다양한 메트릭을 추가하기위해 용도별 prometheus를 배포 (ex> 서비스 매시, 사용자 서비스,....)
  - 조회 및 활용 수준의 일관성 제공은 필수
- 장기데이터 보관을 필요

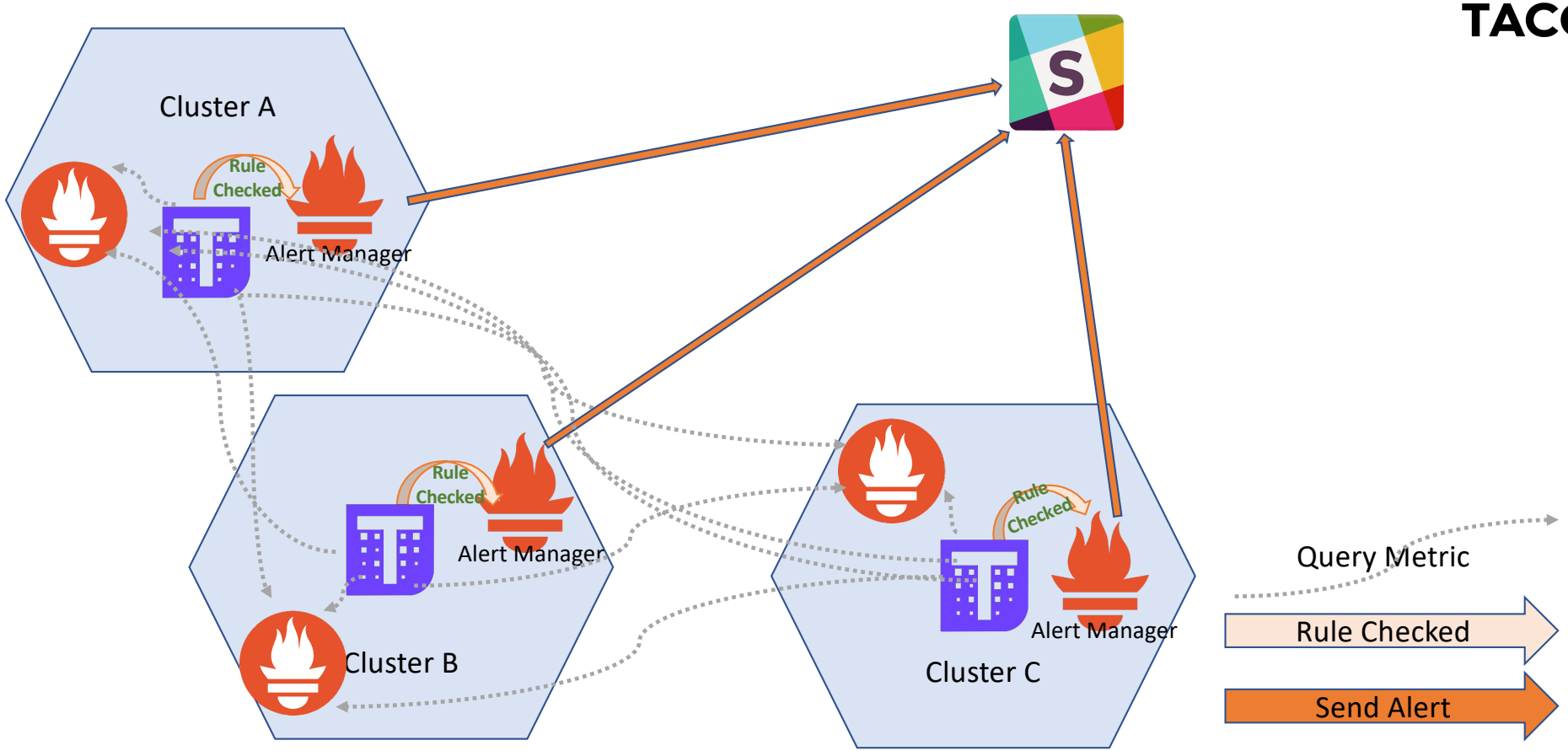


#1 알람 중복 이슈

TACO



#1 알람 중복 이슈


Cluster A
Rule Checked
Alert Manager
S
Rule Checked
Alert Manager
Cluster B
Rule Checked
Alert Manager
Cluster C
Query Metric
Rule Checked
Send Alert

#1 알람 중복 이슈

- 동일한 형상유지를 위해 모든 클러스터에 알람 기능이 들어가고 기존의 룰을 그대로 사용하면서 발생
  - 모든 클러스터에 알람기능(Thanos rule)이 설치되고 클러스터의 개수 만큼의 alert이 발생
  - Thanos alert은 알럿의 수에 비례하여 질의가 발생하고 이는 성능상 과부하 문제가 있음
- 해결방법
  - 알람룰을 개별 클러스터로 분배하고 Prometheus rule engine에서 처리하도록 함
- Prometheus와 Thanos rule은 완전 호환되므로 형식을 바꿀 필요는 없으나 cluster별 query를 하기 위해 들어갔던 내용을 삭제할 필요가 있었음
- 클러스터간 연관관계를 통한 알람을 정의하기 위해서는 thanos의 ruler를 사용해야 하며 이경우는 대상클러스터에 하나의 ruler만 설치/운영해야 함

#2 데이터 쓰기경합 체크

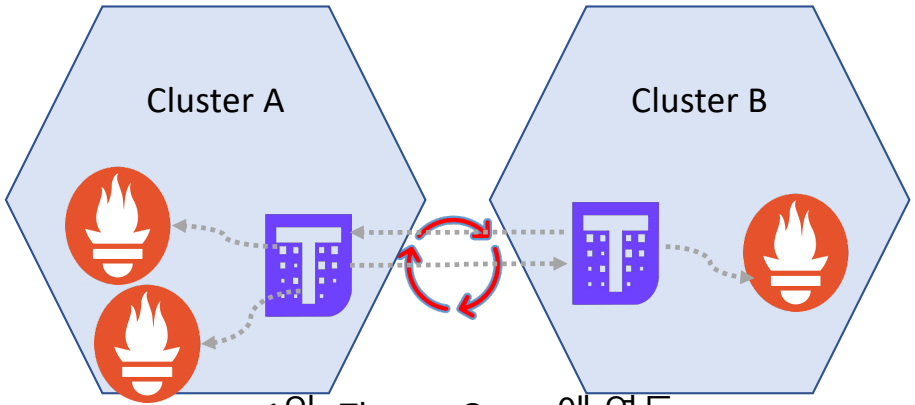
- Thanos는 Object storage를 공유하여 데이터를 저장하고 이를 조회하도록 구성되어 있음
- 다수의 thanos에 동일한 작업을 수행하도록 하면서 문제가 발생할 수 있는 부분을 체크필요

Thanos Component	Read	Write	비고
Bucket Web	O	X	
Compactor	O	O	전체 데이터를 대상으로 rollout, rollover등 재가공이 이뤄지므로 전체 클러스터에 하나만 운영해야 함
Minio	X	X	
Query	O	X	
Query-frontend	X	X	
Ruler	O	X	
Store Gateway	O	X	
Thanos sidecar(in Prometheus)	O	O	sidecar는 해당 클러스터에서 발생하는 데이터를 백업하므로 클러스터별 sidecar사용에 문제는 없음

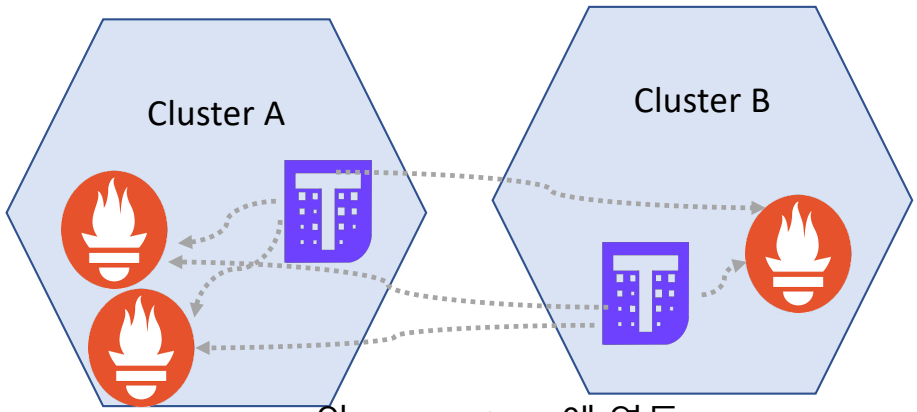
#3 순환 Graph 발생 주의

TACO

- Thanos는 query/querier를 통해 다양한 backend를 연동
- Thanos의 query는 또 다른 query를 backend로 연결가능
- 결정된 멀티클러스터 구조에는 모든 노드에 prometheus, thanos가 있고 서로 연동되어야 함
- 두가지 연결방법이 존재
  - ***1안의 경우 순환 graph가 발생하게 되므로 2안의 방식으로 연동해야 함***



1안: Thanos Query에 연동



2안: Prometheus에 연동

Github에서 구체적 설정확인 가능

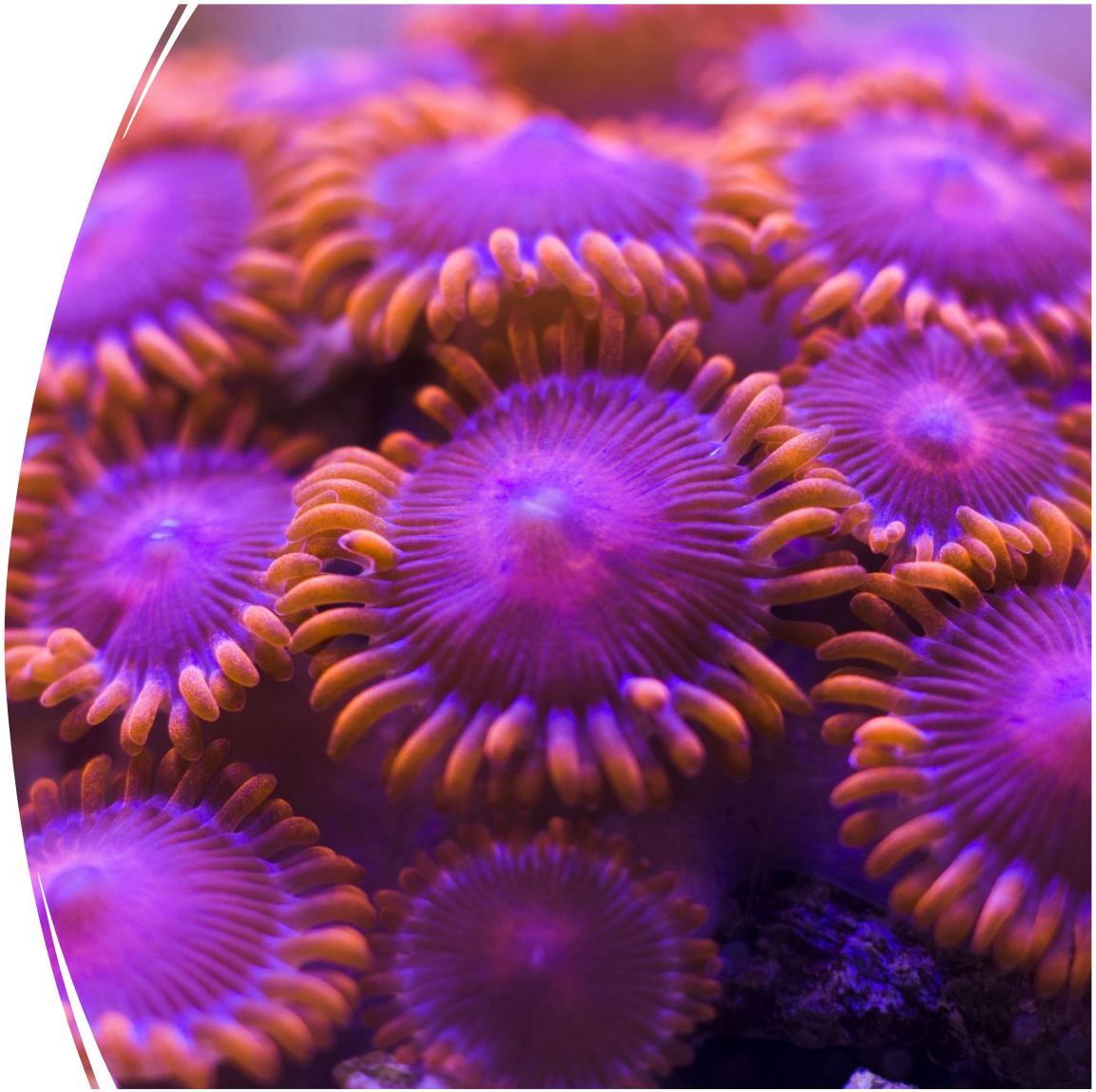
- https://url.kr/bsfl76
- https://url.kr/dwlr8h

```
    queryFrontend:
      enabled: false
      nodeSelector: {}
      service:
        type: TO_BE_FIXED
        http:
          port: 9090
          nodePort: TO_BE_FIXED
      config: |-
        type: IN-MEMORY
        config:
          max_size: TO_BE_FIXED
          max_size_items: TO_BE_FIXED
          validity: TO_BE_FIXED
      extraFlags: []
        # query-range.split-interval: TO_BE_
        # query-range.max-retries-per-reques
        # query-frontend.log-queries-longer-
```

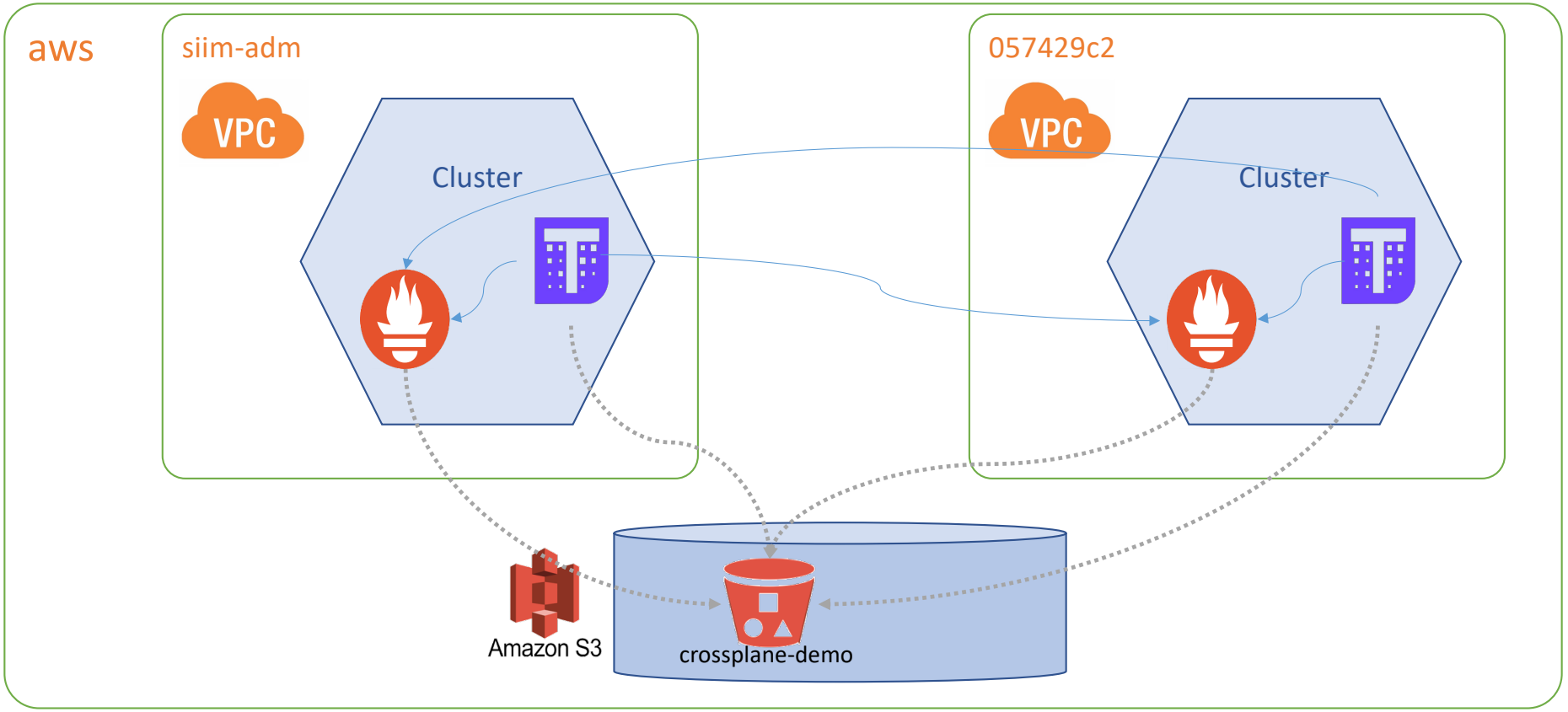
```
    compactor:
      enabled: TO_BE_FIXED
      logLevel: info
      retentionResolutionRaw: TO_BE_FIXED
      retentionResolution5m: TO_BE_FIXED
      retentionResolution1h: TO_BE_FIXED
      consistencyDelay: TO_BE_FIXED
      resources:
        limits:
          memory: TO_BE_FIXED
          cpu: TO_BE_FIXED
      nodeSelector: {}
      service:
        type: ClusterIP
        http:
          port: 9090
      persistence:
        enabled: true
        accessModes:
          - ReadWriteOnce
        size: TO_BE_FIXED
      extraFlags: []
        # Compaction 수행 관련 커맨트
        # 아래와 같은 block stream들이 있을때
        # external_labels: {cluster="eu1", replica="1", receive="true", environment="production"}
        # external_labels: {cluster="eu1", replica="2", receive="true", environment="production"}
        # external_labels: {cluster="us1", replica="1", receive="true", environment="production"}
        # external_labels: {cluster="us1", replica="1", receive="true", environment="staging"}
```

```
apiVersion: helm.fluxcd.io/v1
kind: HelmRelease
metadata:
  labels:
    name: thanos
  name: thanos
spec:
  helmVersion: v3
  chart:
    type: helmrepo
    repository: https://charts.bitnami.com/bitnami
    name: thanos
    version: 3.4.0
  releaseName: thanos
  targetNamespace: lma
  values:
    global:
      storageClass: local-path
    clusterDomain: TO_BE_FIXED
    existingObjstoreSecret: TO_BE_FIXED
    query:
      nodeSelector: {}
      dnsDiscovery:
        enabled: true
        sidecarsService: TO_BE_FIXED
        sidecarsNamespace: lma
    queryFrontend:
      enabled: false
```

Demo



Demo


aws
siim-adm
VPC
Cluster
057429c2
VPC
Cluster
Amazon S3
crossplane-demo

Demo - 참고

https://devocean.sk.com/blog/techBoardDetail.do?ID=163458

https://devocean.sk.com/blog/techBoardDetail.do?ID=163447

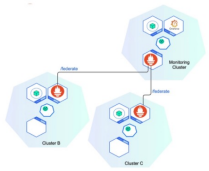
https://youtu.be/-sfWvPqiIrU

Thanos를 이용한 멀티클러스터 모니터링

지난 블로그(https://devocean.sk.com/blog/techBoardDetail.do?ID=163447)에서는 Prometheus를 사용하여 kubernetes를 모니터링 하는 방법과 TACO의 예를 통해 실제 배포하는 예를 설명하였다. 이어서 본 블로그에서는 Prometheus에서 부족했던 부분을 채우고 다양한 통합기능을 제공하는 Thanos에 대해 설명하고자 하고 설치하는 방법을 알아본다. Prometheus의 H.A. 문제 Prometheus는 cloud native 환경에...

#Prometheus

sungil | 21.11.10 298 5 2

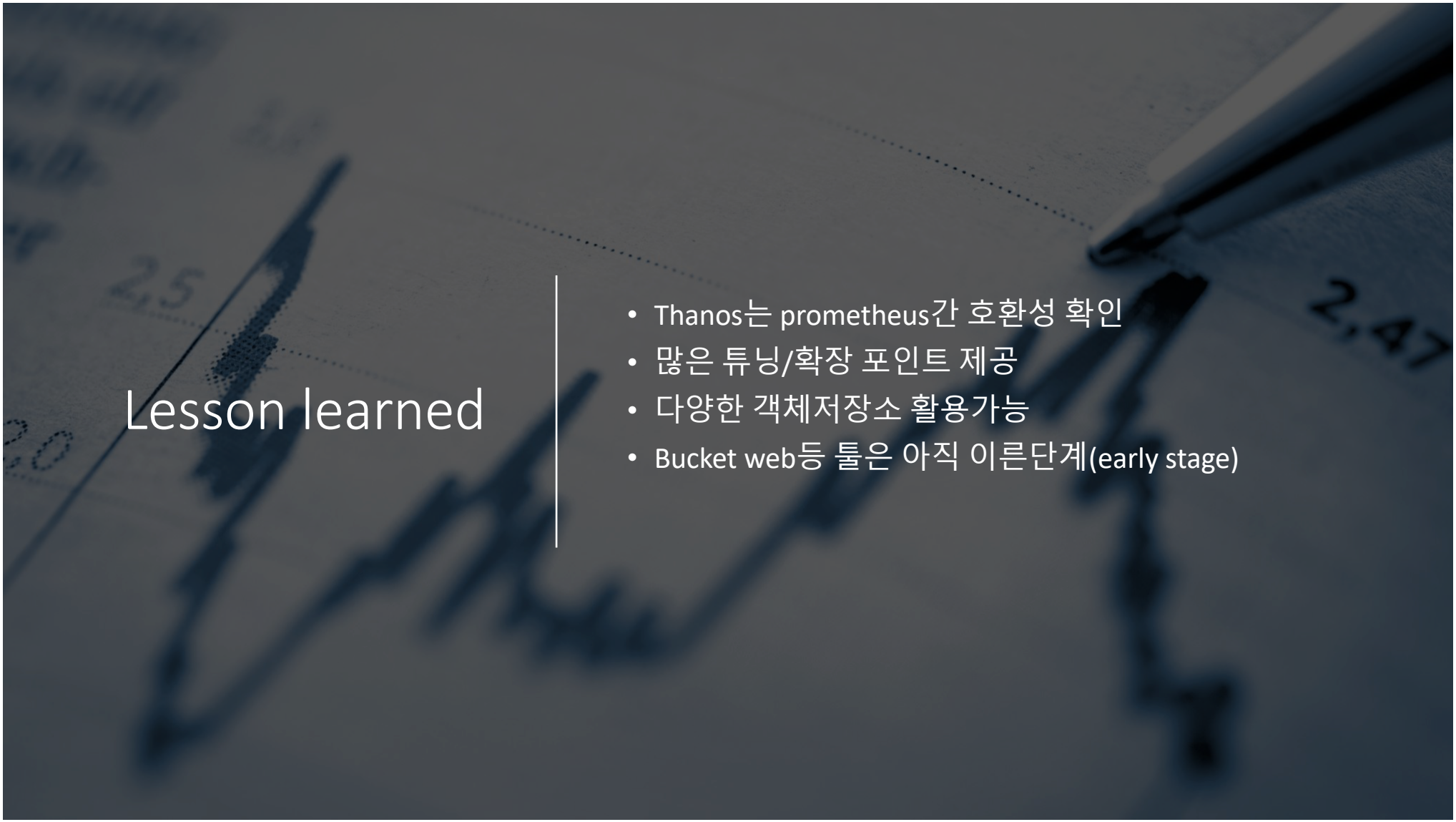


Prometheus를 사용한 kubernetes 모니터링

기존의 모놀리스한 방식의 소프트웨어와 달리, 다수의 Micro service들을 통해 서비스가 만들어지는 Container 및 kubernetes 환경에서 개별 동작에 대한 가시성의 확보는 점점 중요해지고 있다. 시스템, 소프트웨어, 서비스 등 IT 전반에서 정상적 운영과 활용을 위해 운영 자원들에 대한 모니터링은 필수 요소이다. TACO는 Cloud Native한 환경을 지원하기 위한 Container 플랫폼을 제공하고 있으며, 플랫폼 및 내부의 구성요소를 모니터링...

#Programming Language #Prometheus

sungil | 21.11.02 1599 5 0



Lesson learned

- Thanos는 prometheus간 호환성 확인
- 많은 튜닝/확장 포인트 제공
- 다양한 객체저장소 활용가능
- Bucket web등 툴은 아직 이른단계(early stage)


Thanks