DEVOXX MOROCCO

#DevoxxMa2024

LOW-COST, UNLIMITED METRICS STORAGE WITH THANOS:

Monitor All Your K8s Clusters Anywhere and More

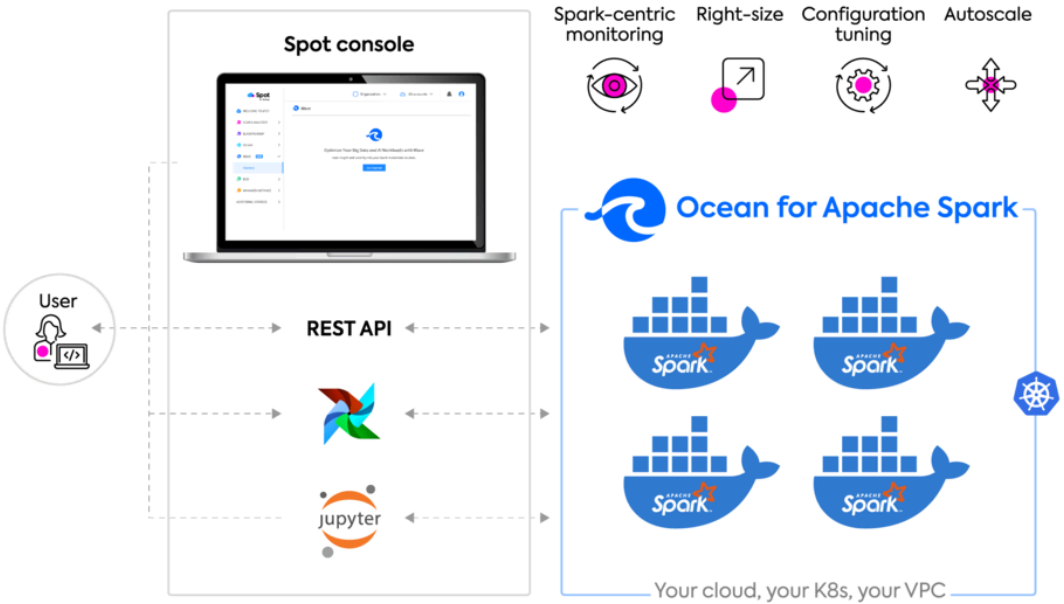
Zakaria EL BAZI

Zakaria EL BAZI
Infrastructure engineer at NetApp
(Ocean for Apache Spark team)

https://elbazi.me
https://awsmorocco.com

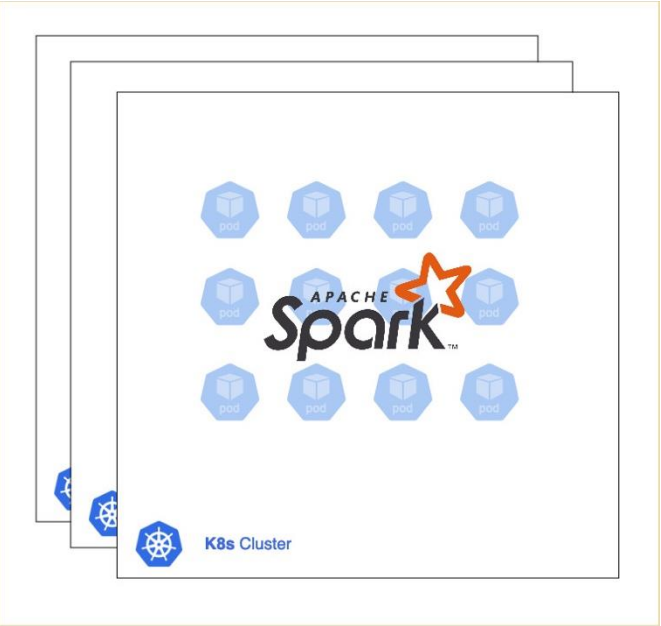
Ocean for Apache Spark

A data platform for running Apache Spark workloads (batch, streaming, notebooks) on Kubernetes in the cloud, offering an easy, "serverless-like", and cost-efficient solution.



Ocean for Apache Spark

- The platform is composed of multiple services (deployments) that manage the lifecycle of all Spark workloads.
- The platform runs in the customer's own cloud account (AWS, GCP, or Azure) on their own managed Kubernetes cluster (EKS, GKE, or AKS) in a dedicated namespace.
- There is no ingress to the customer's cluster (the platform operates in a pull-based manner).



But first let’s talk about k8s monitoring ！

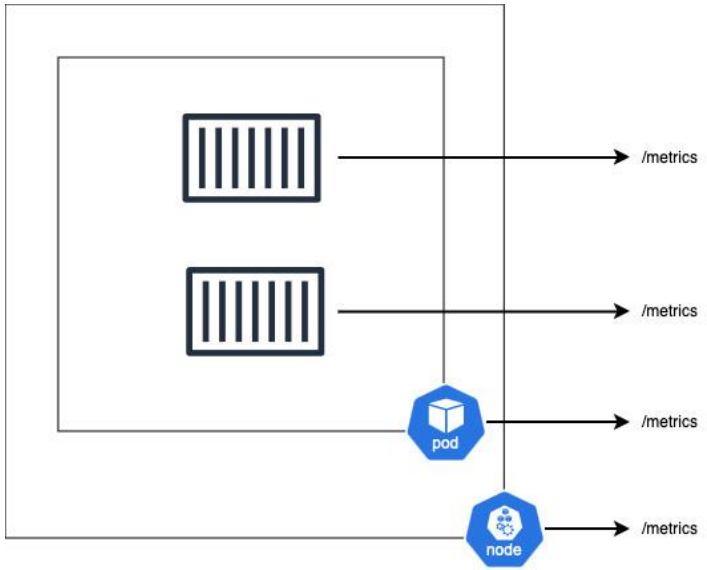
K8s Monitoring

Why monitoring k8s ?

- Ensure application health and performance
- Optimize resource utilization (reduce costs)
- Troubleshoot issues quickly
- Capacity planning

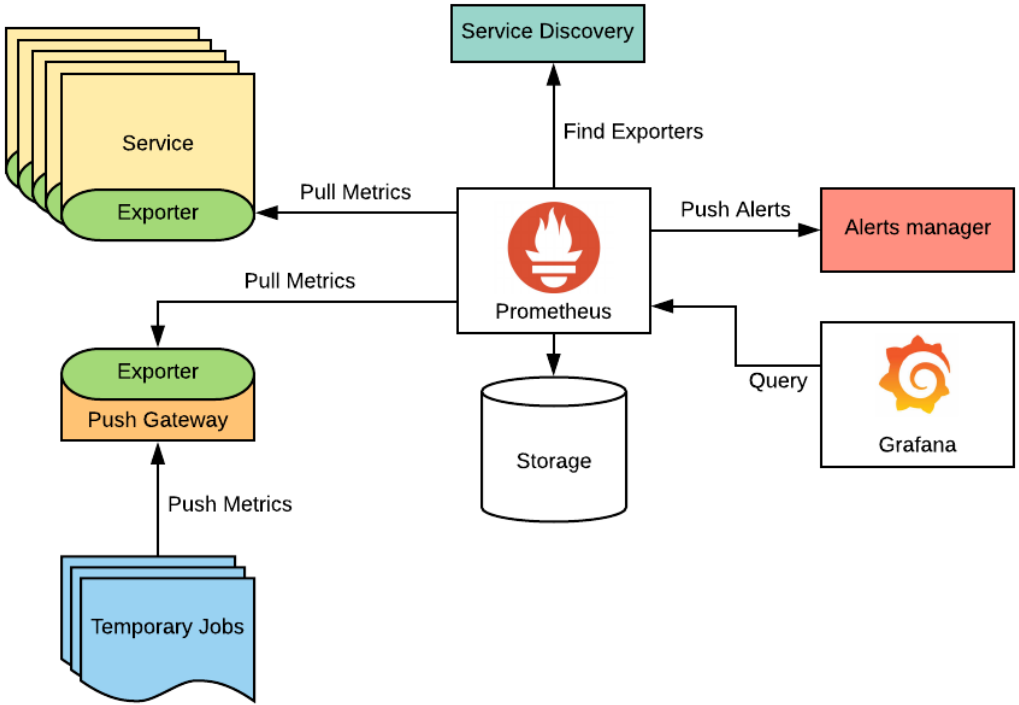
What metrics ?

- **Node-level:** CPU, Memory, Disk, Network
- **Pod-level:** Resource usage, Health status
- **Application-level**: Custom metrics, Latency, Throughput



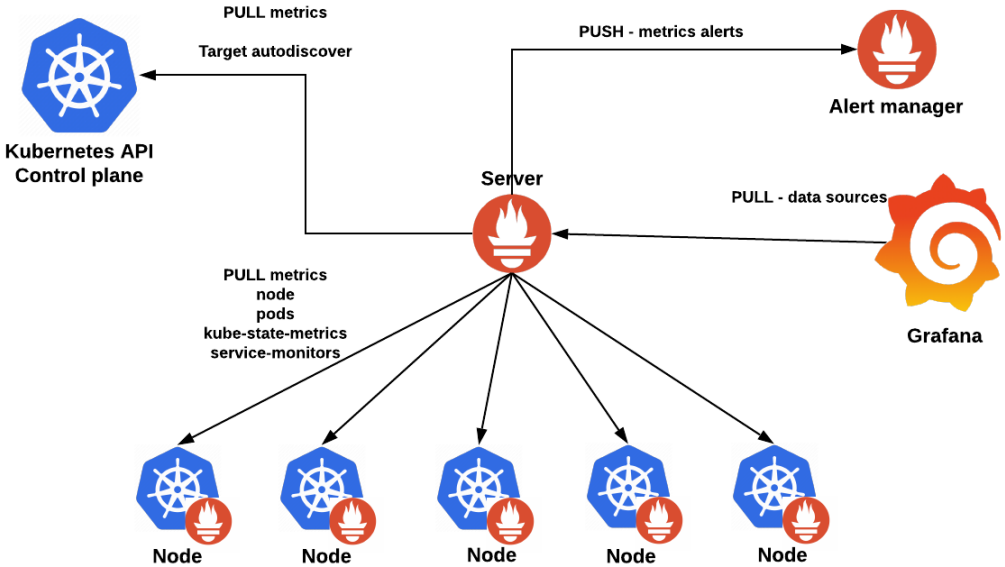
Prometheus

- Open-source monitoring system (Under the CNCF umbrella)
- Pull-based metrics collection
- Powerful query language (PromQL)
- Built-in alert manager



Prometheus in k8s

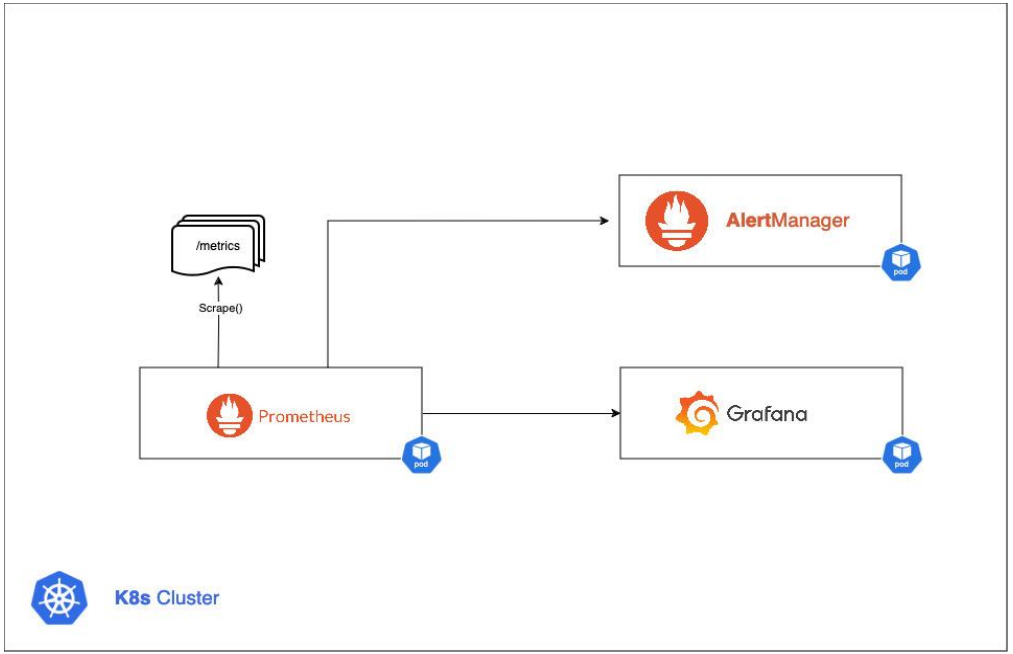
- **Automatic Service discovery**
- **Kubernetes-native deployment**
  - The Prometheus operator
  - Integration with Kubernetes Components (Direct scrapin of kubelet metrics, etc).
- **Rich ecosystem of exporters**
  - Dbs
  - Cloud
  - Hardware
  - etc



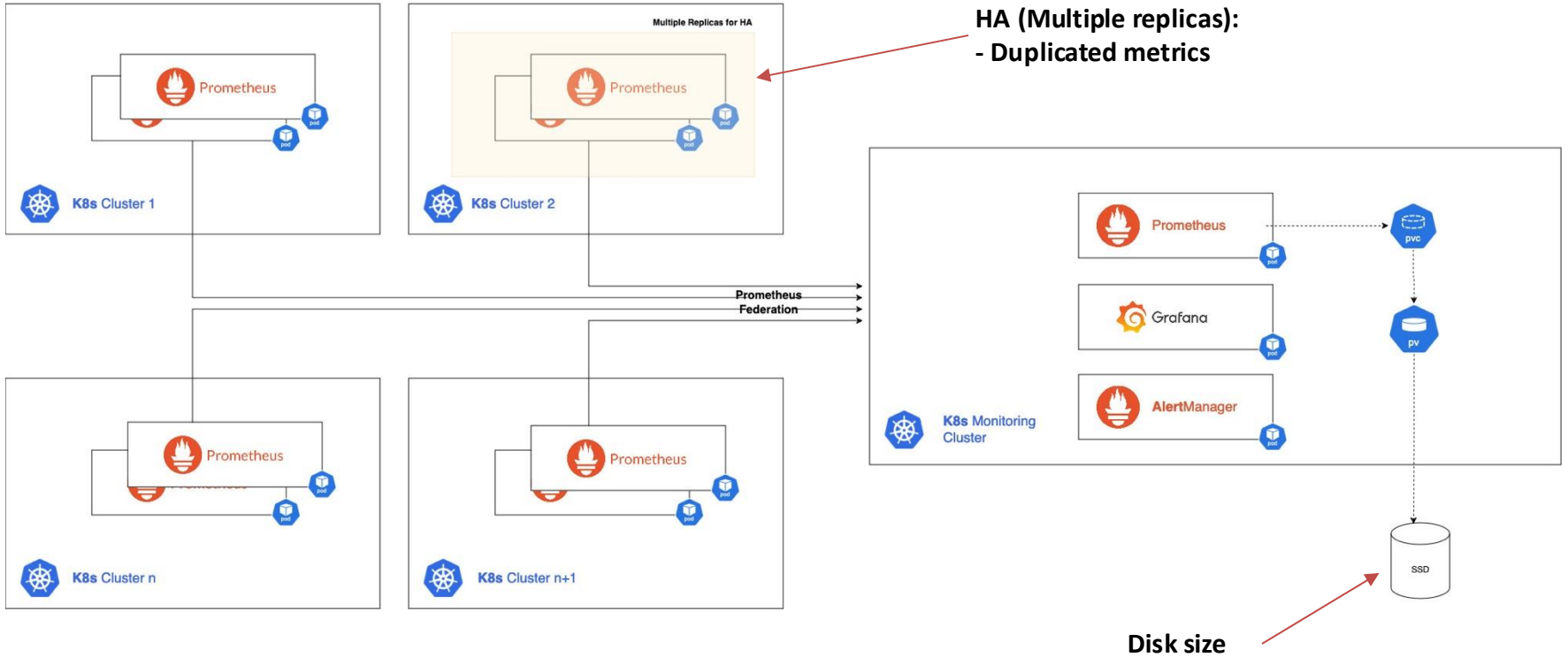
https://prometheus.io/docs/instrumenting/exporters/

But ...

- Prometheus is designed for single cluster monitoring and lacks native multi-cluster support.
- Scale and complexity.
- Data volume and retention.
- (If there in an ingress to the cluster) the high volume of data may cause performance issue with complexes queries.



But ...


Multiple Replicas for HA
Prometheus
K8s Cluster 1
Prometheus
K8s Cluster 2
HA (Multiple replicas):
- Duplicated metrics
Prometheus
Federation
Prometheus
Grafana
AlertManager
K8s Monitoring Cluster
pvc
pv
SSD
Prometheus
K8s Cluster n
Prometheus
K8s Cluster n+1
Disk size

Thanos

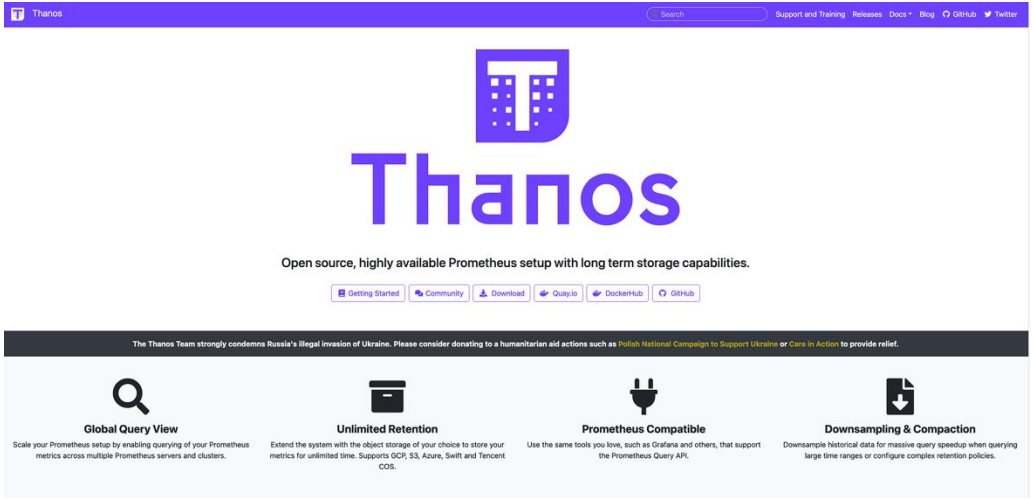
What is Thanos ?

Thanos

- Open-source project extending Prometheus capabilities (**CNCF Incubating project**) with unlimited metrics storage in multi-cluster environments.

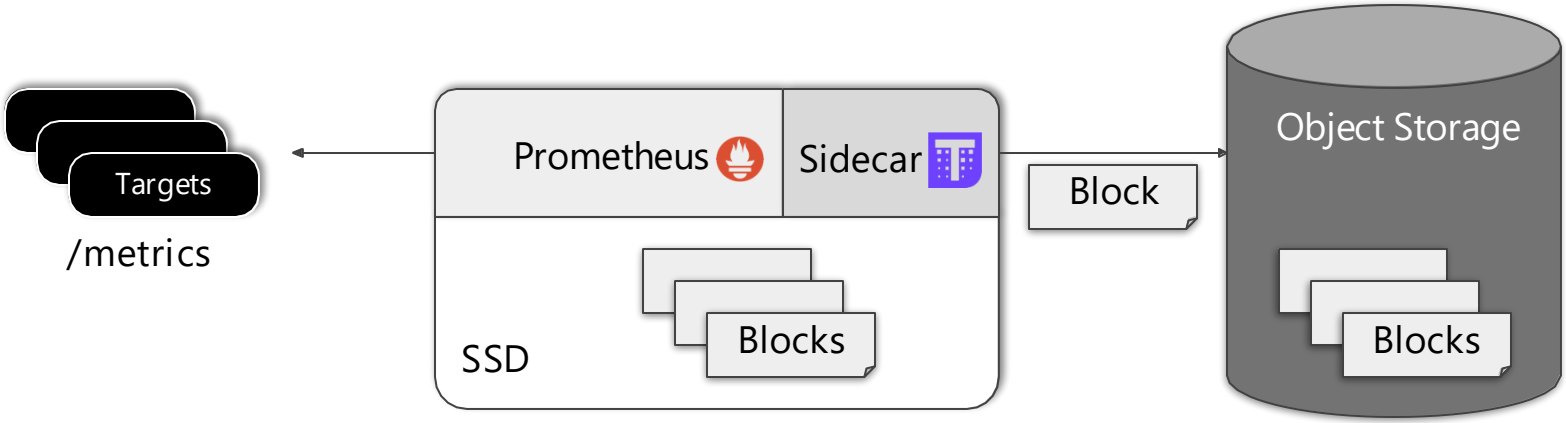
- **High availability** and **fault tolerance** for Metrics storage.

- **Downsampling** for efficient long-term storage.



What is Thanos ?

- Scalable from simple to complex use cases
- Components can be used independently or together.
- Adapt to various architectures and requirements:



(simple setup) Use Sidecar for basic long-term storage with object Storage

Architecture

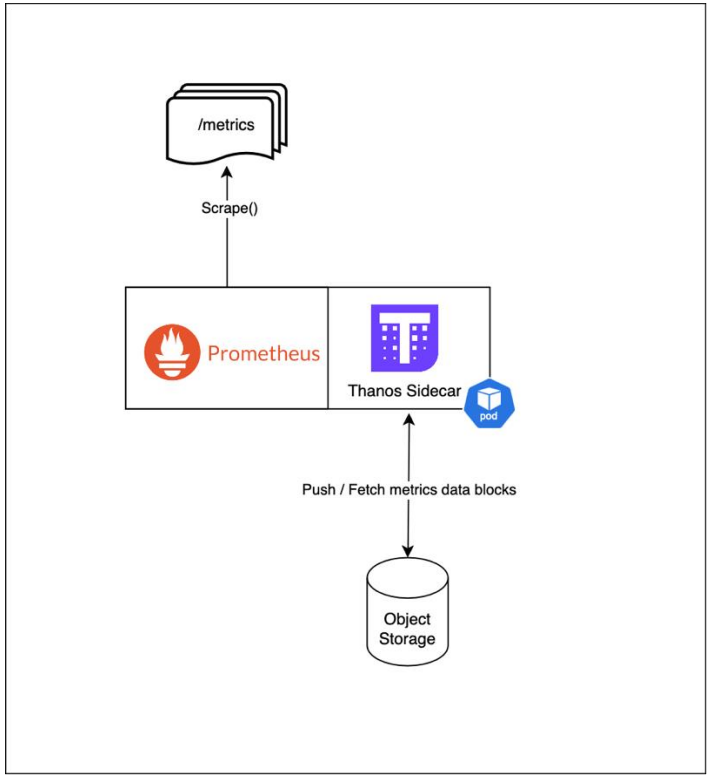
Thanos Sidecar

Role:

- Uploads metrics to object storage

Key features:

- Runs alongside Prometheus instances
- Uploads TSDB blocks to object storage (e.g., S3, GCS)
- Enables long-term storage without affecting Prometheus performance



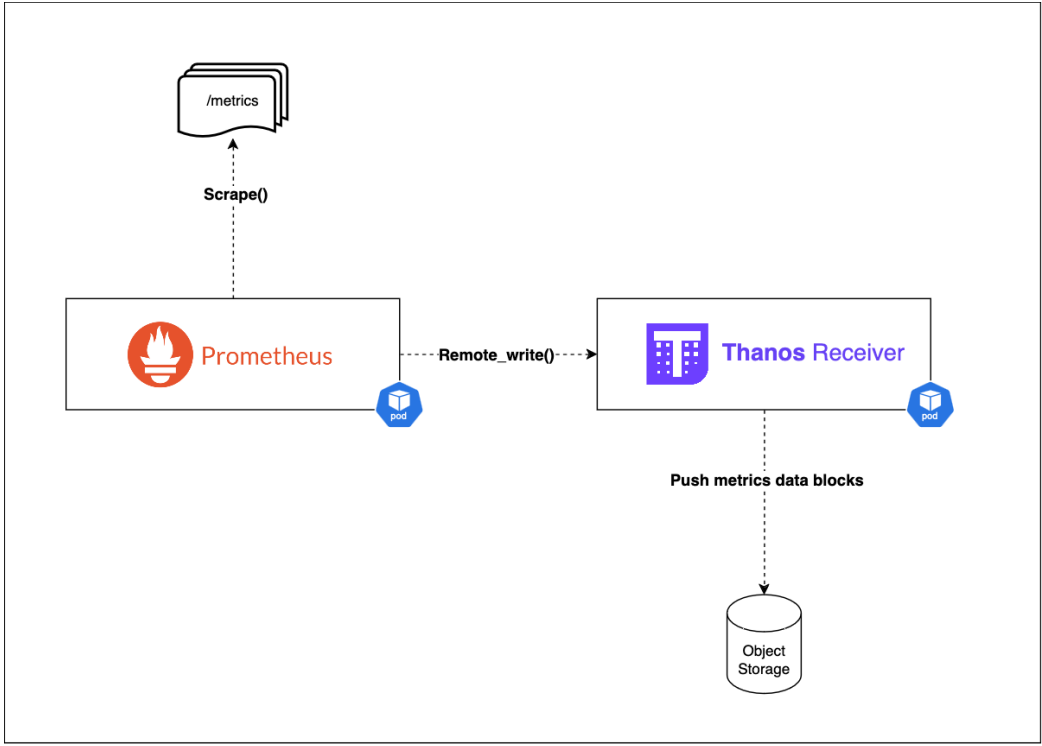
Thanos Receiver

Role:

- Ingests metrics from remote sources (Prometheus remote_write)

Key features:

- Accepts remote write from Prometheus
- Writes data to object storage
- Exposes metrics to Thanos Queriers for real-time viewing.



Thanos Receiver


```
thanos receive \
  --objstore.config=$(OBJSTORE_CONFIG) \
  --tsdb.path=/var/thanos/receive \
  --label=cluster_name="aws-morocco-outpost-cluster" \
  --grpc-address=0.0.0.0:10901 \
  --http-address=0.0.0.0:10902 \
  --remote-write.address=0.0.0.0:19291 \
  --receive.local-endpoint=127.0.0.1:10901 \
  --receive.grpc-compression=none \
  --tsdb.retention=1d \
  --tsdb.min-block-duration=5m \
  --tsdb.max-block-duration=5m \
  --receive.hashrings-file=/var/lib/thanos-receive/hashrings.json
```

→

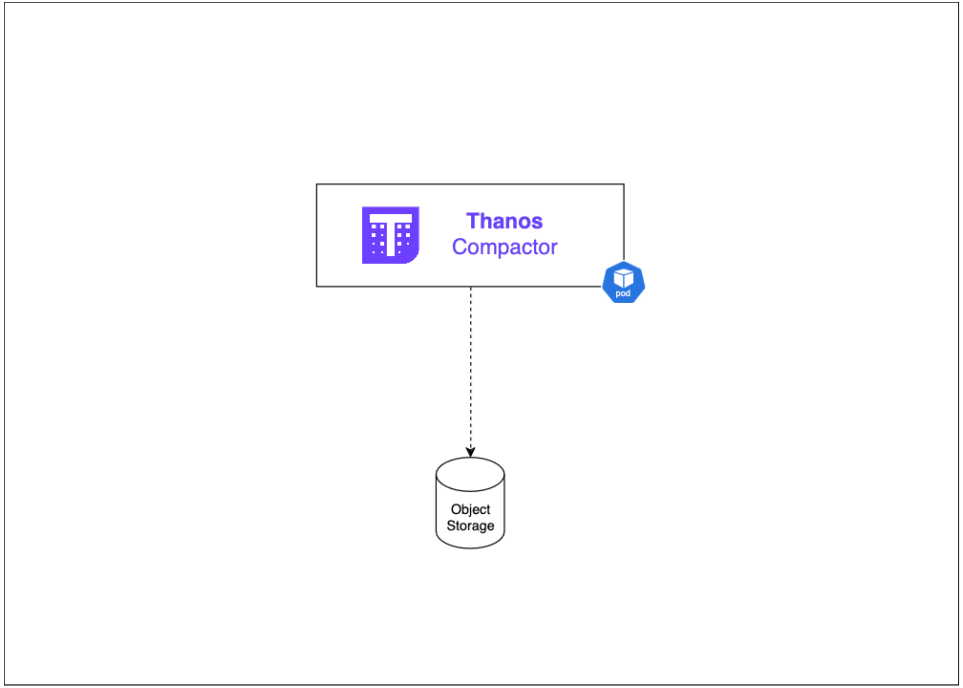
Thanos Compactor

Role:

- Optimizes object storage data.

Key features:

- Compacts data for efficient storage
- Creates summarized versions of historical data at lower resolutions (Typically produces 5-minute and 1-hour resolution datasets from raw data)
- Applies retention policies.



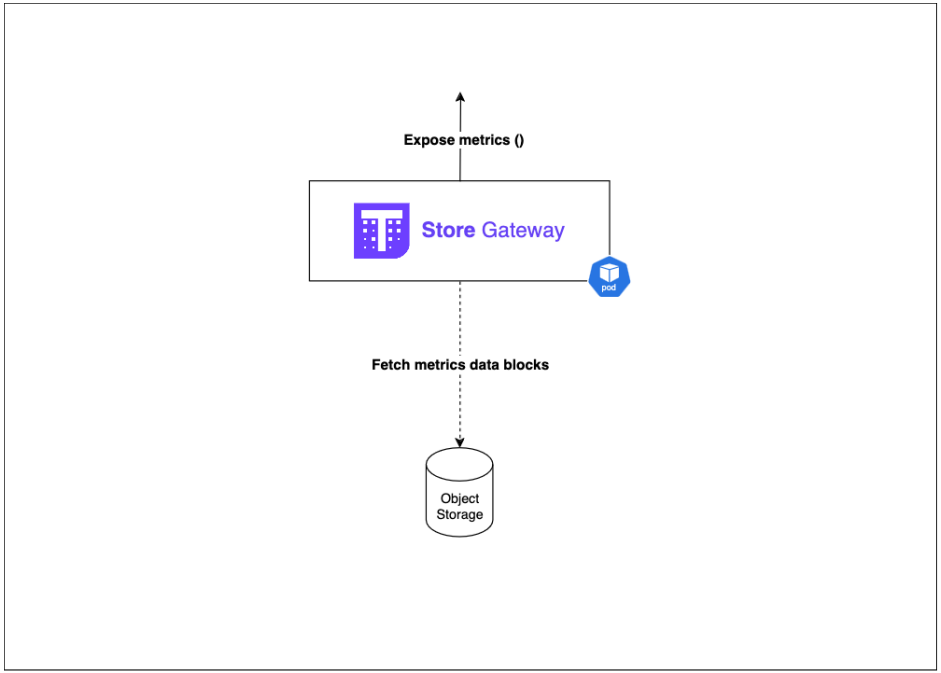
Thanos Store Gateway

Role:

- Provides access to object storage data.

Key features:

- Caches object storage data for faster access
- Optimizes data retrieval for queries
- Acts as a proxy between Querier and object storage.



Thanos Querier

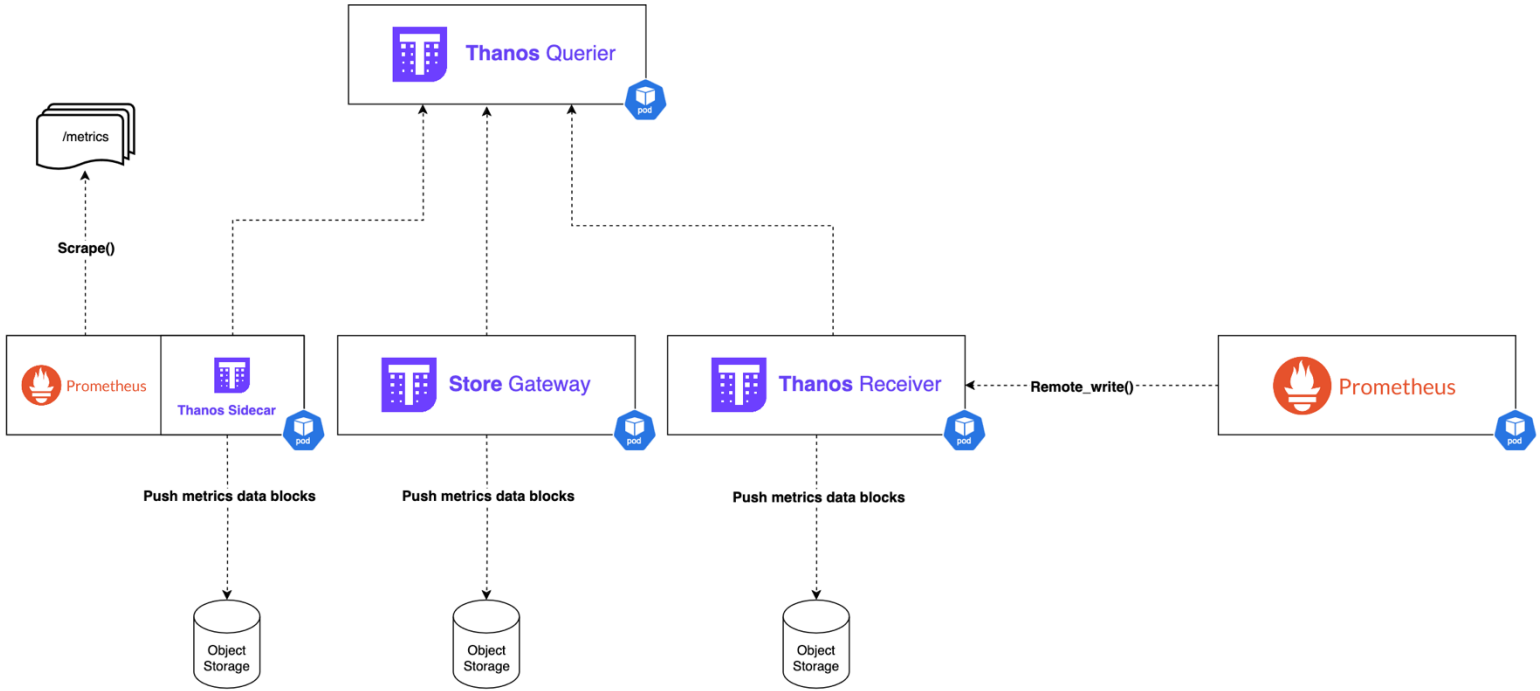
Role:

- Global query interface.

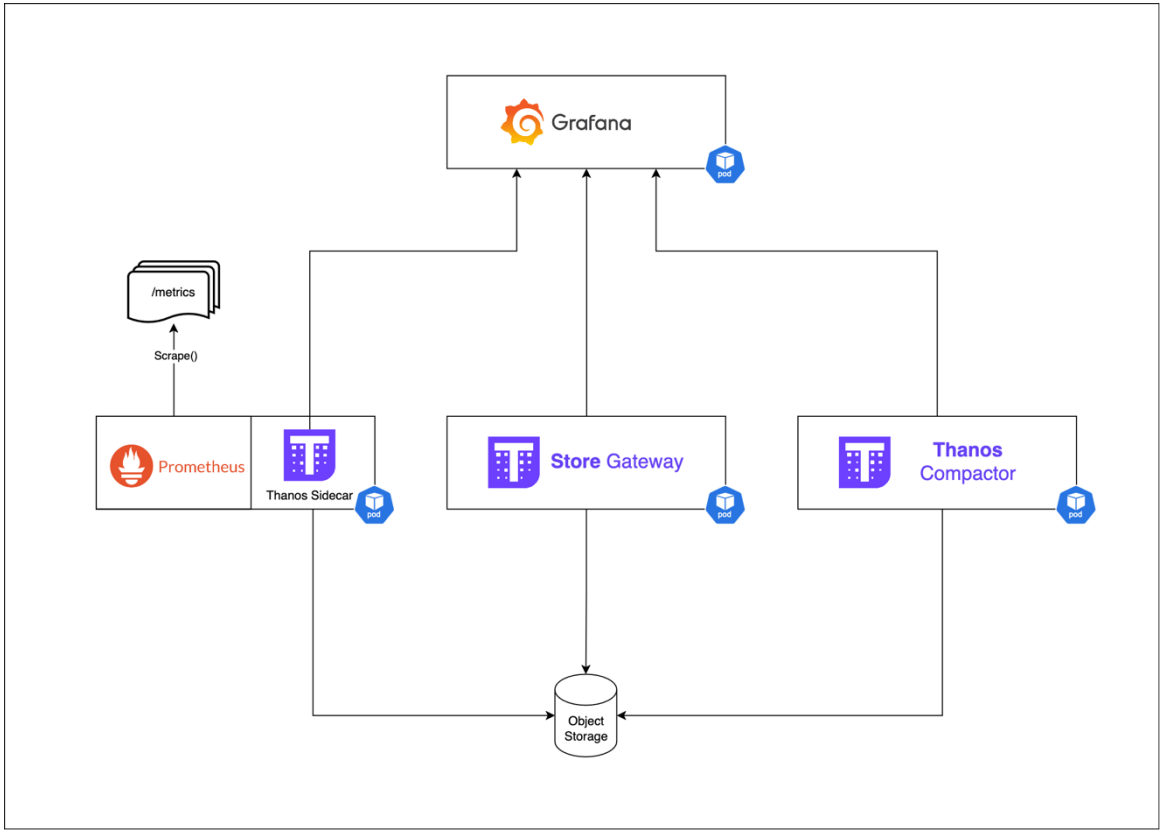
Key features:

- Provides PromQL interface for querying
- Deduplicates metrics from different sources.
- Aggregate data from all the sources (Sidecars, Store Gatways, Prometheus,etc).

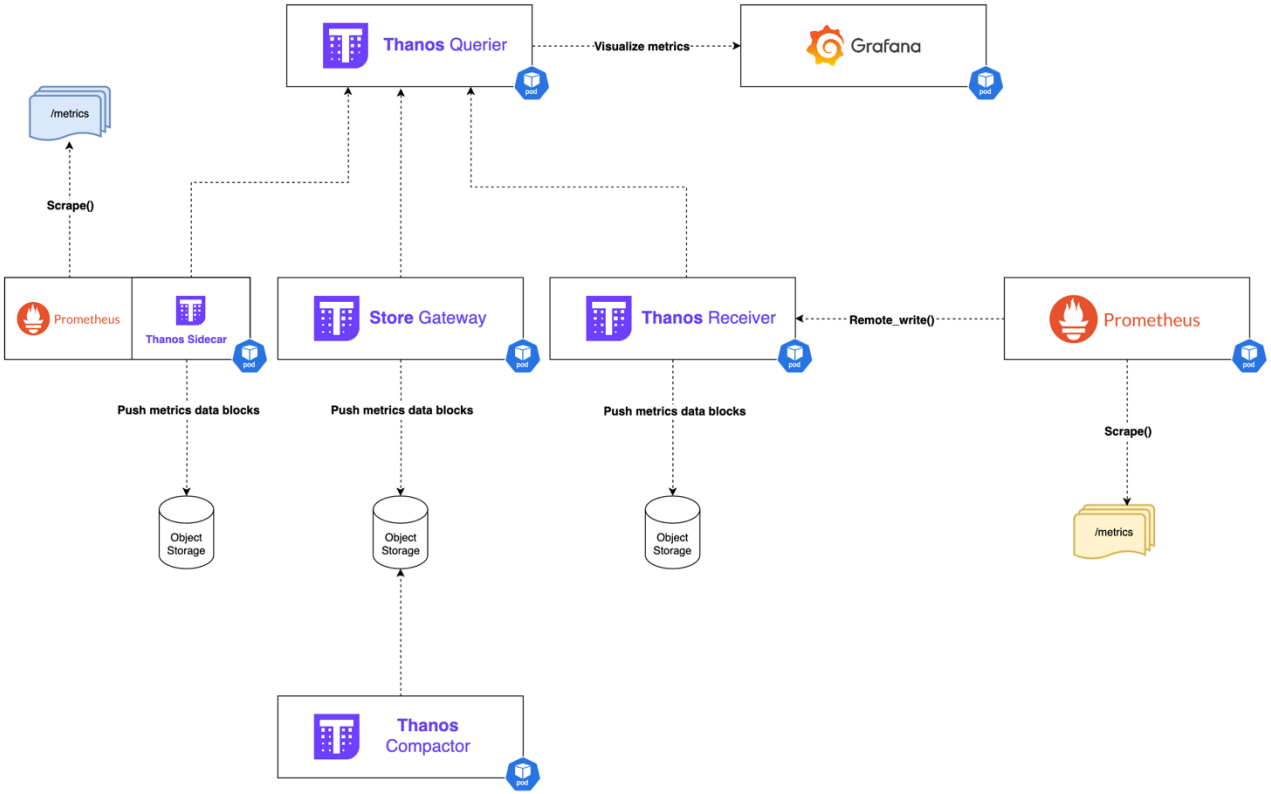
Thanos Querier



Example Deployment (simple)

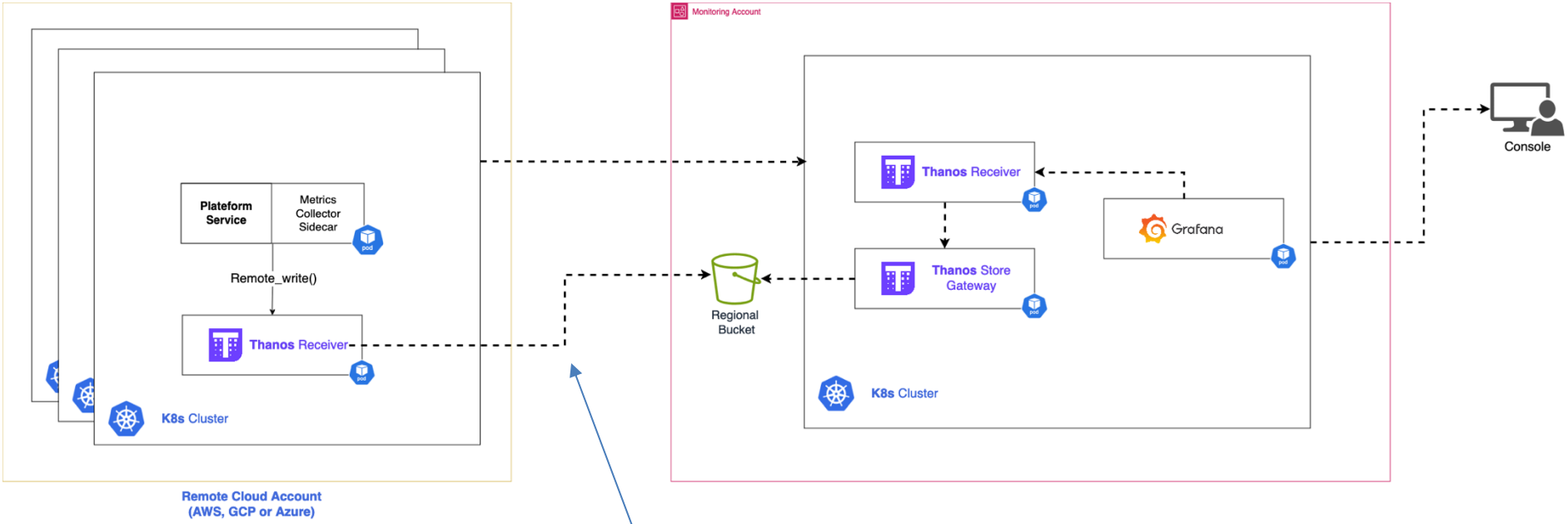


Example Deployment (Complete)

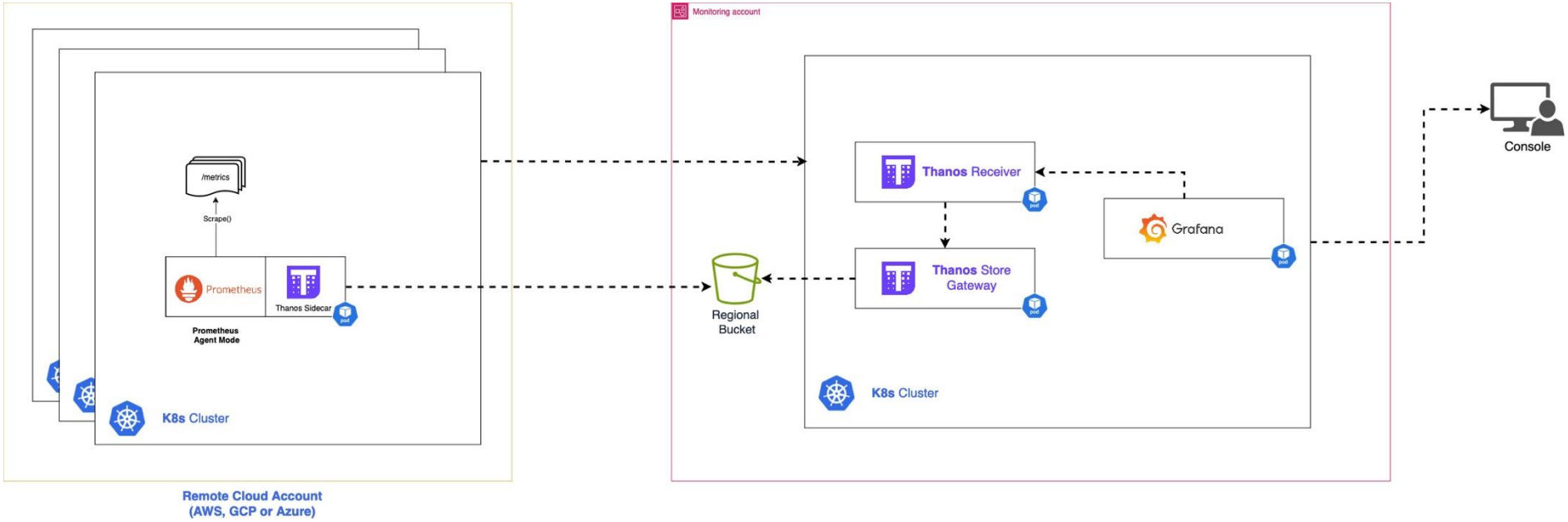


Multiple clusters monitoring

Multiple clusters monitoring (option1)



Multiple clusters monitoring (option2)



Thank you

https://elbazi.me

https://awsmorocco.com