在 k8s 上部署高可用的 service mesh 监控

唐鹏程 才云科技

pctang@caicloud.io

TOC

- Old-school monitoring
- Solving issues in a new way
- Monitoring your service mesh

Prometheus + Kubernetes

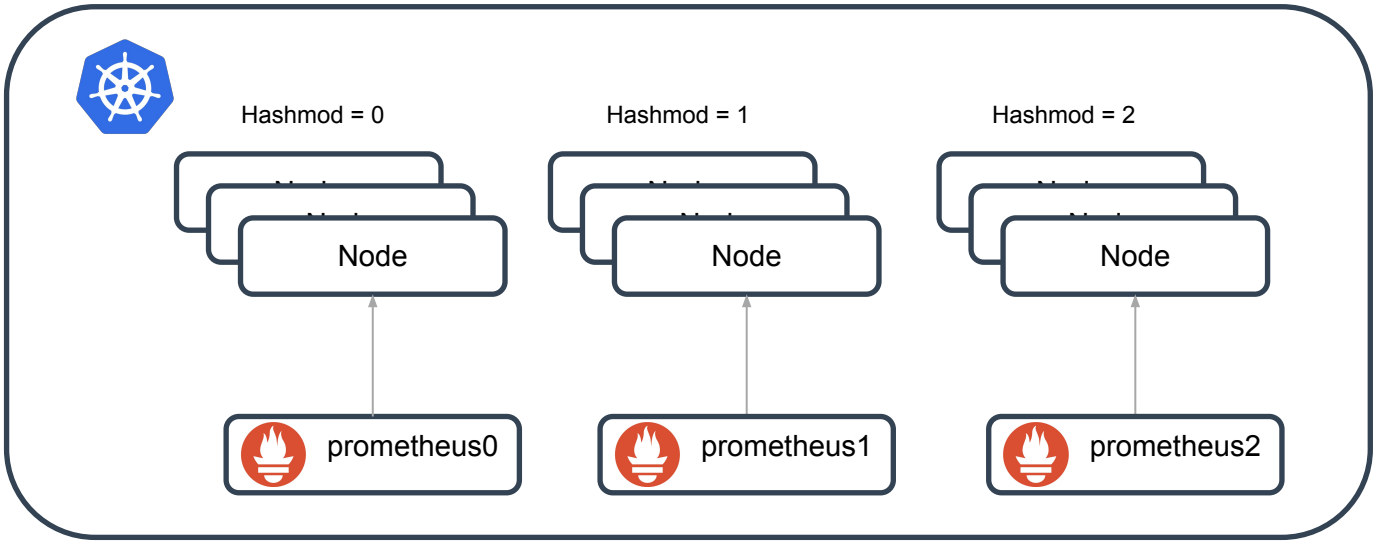
- A time series based monitoring system.
- Borgmon for mere mortals.
- Seamless integration with kubernetes at infrastructure and app level.
- Key - value data model with powerful PromQL.
- Emerging open source community.

Prometheus 2.0 - deal with dynamic nature

- Storage engine redesign & reimplementation
  - Save more CPU, RAM & IOPS
  - Scale to far high number of time series
  - Better performance in face of pod churn
- Improved staleness semantic

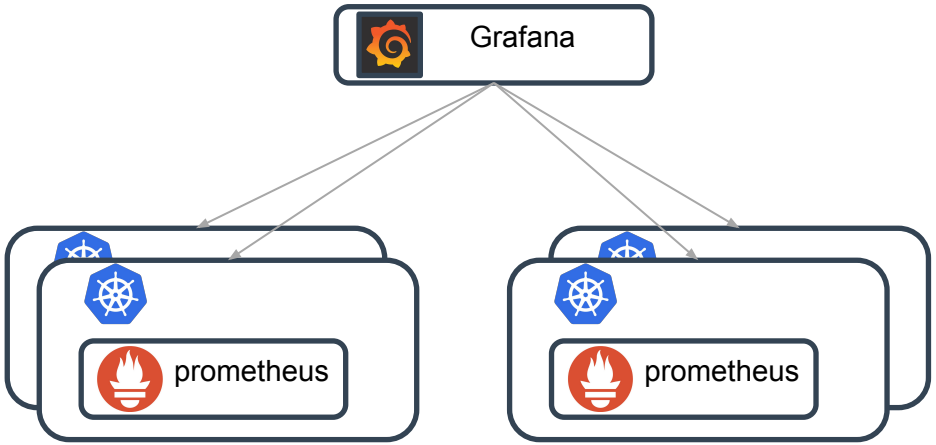
Prometheus at scale

- In the old days…
  - one or more prometheus per cluster
  - hashmod sharding



- In the old days…
  - one or more prometheus per cluster
  - hashmod sharding

Almost works...



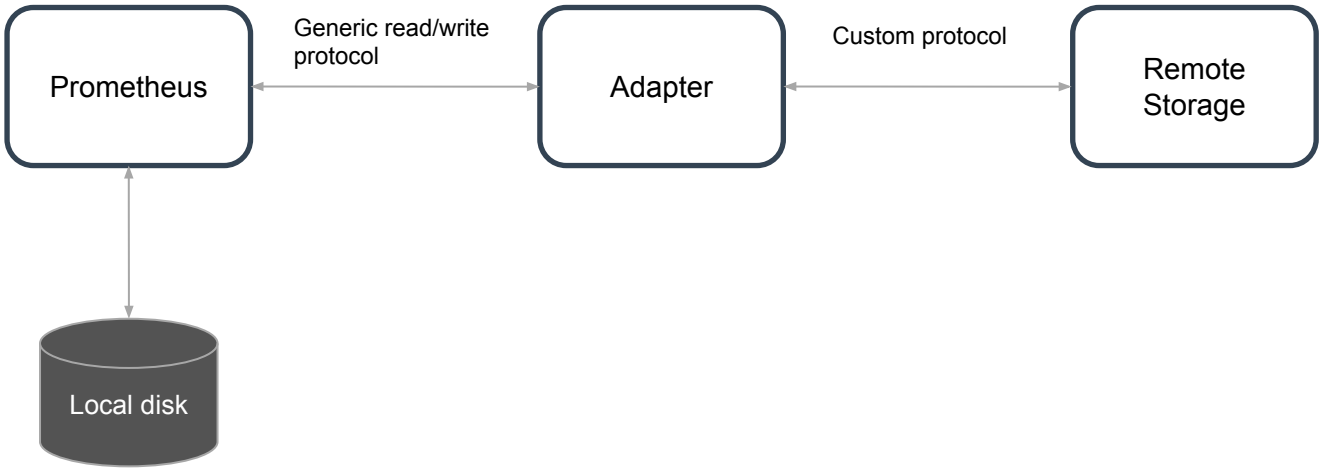
- Did we achieve our SLO goal this quarter?
- Is our network bandwidth saturated this year?
- What is the resource usage for our server farm across the years?

- Did we achieve our SLO goal this quarter?
- Is our network bandwidth saturated this year?
- What is the resource usage for our server farm across the years?

Need For More Retention!

Prometheus at scale

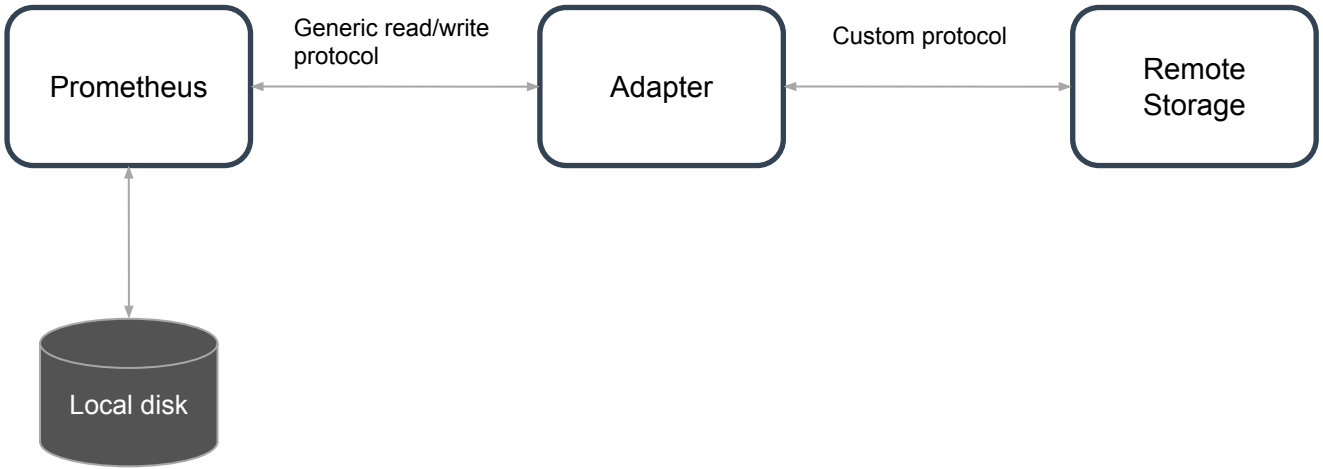
- Prometheus is by design **NOT** a persistent store.
- Have to live with all those DBs...



Prometheus at scale

- Prometheus is by design **NOT** a persistent store.
- Have to live with all those DBs...

Performance and reliability aside,
more things to maintain.
Bad news for the ops… :(



What is missing?

Global View

caicloud 才云

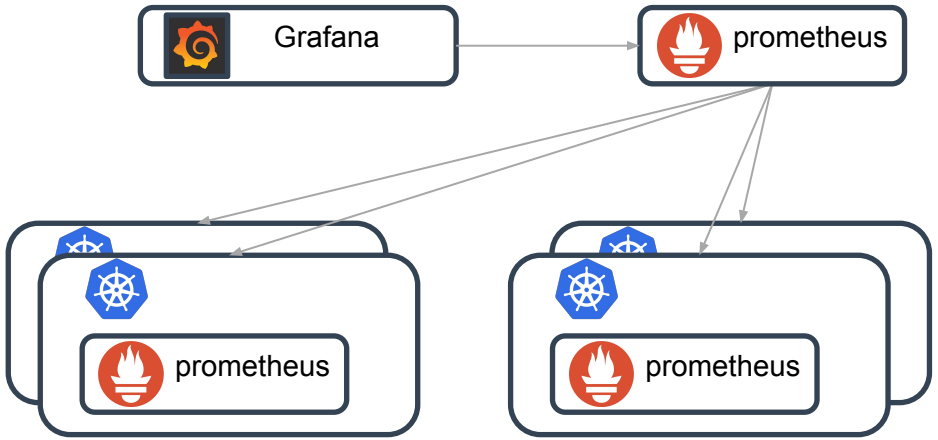
Global View

Old-schooled federation

- “Slave” prometheis collecting metrics for one cluster.
- Top level prometheus scraping from slaves.
- Top level prometheus as a query entry point.

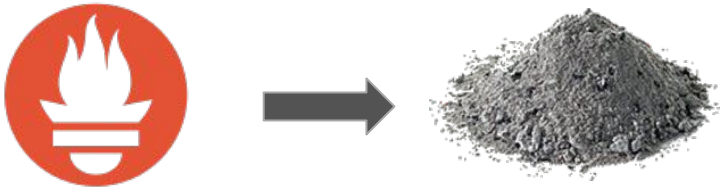
What’s wrong?

- SPOF
- Have to configure for each and every prometheus instance.
- Top level prometheus only scrape part of the data

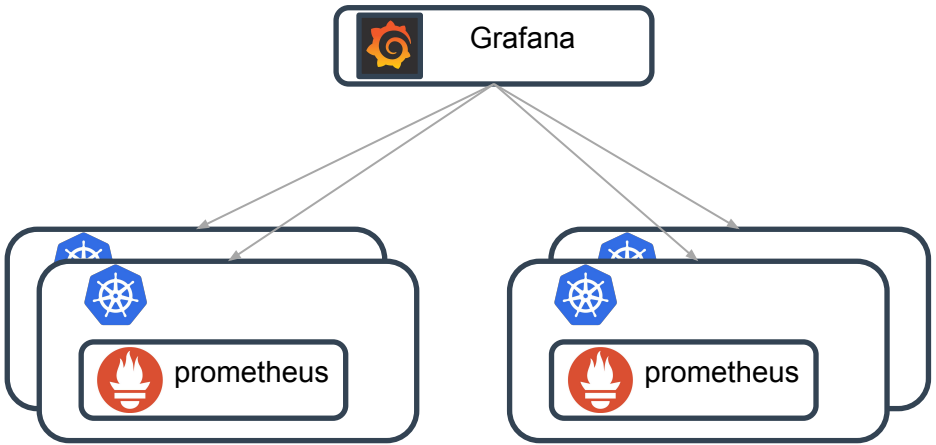


- Too critical, should be the last one standing…

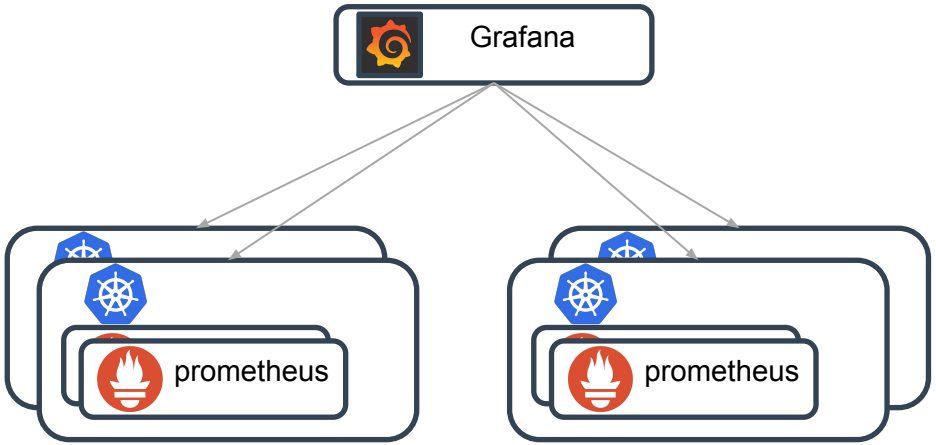
- But there are...
  - hardware failures
  - software failures
  - Maintenance and upgrades



- In the old days...
  - Adding more independent replicas



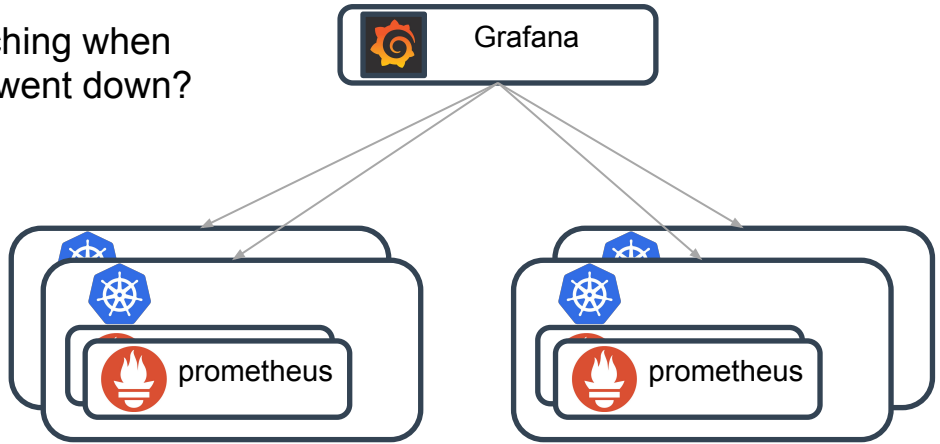
- In the old days...
  - Adding more independent replicas



- In the old days...
  - Adding more independent replicas

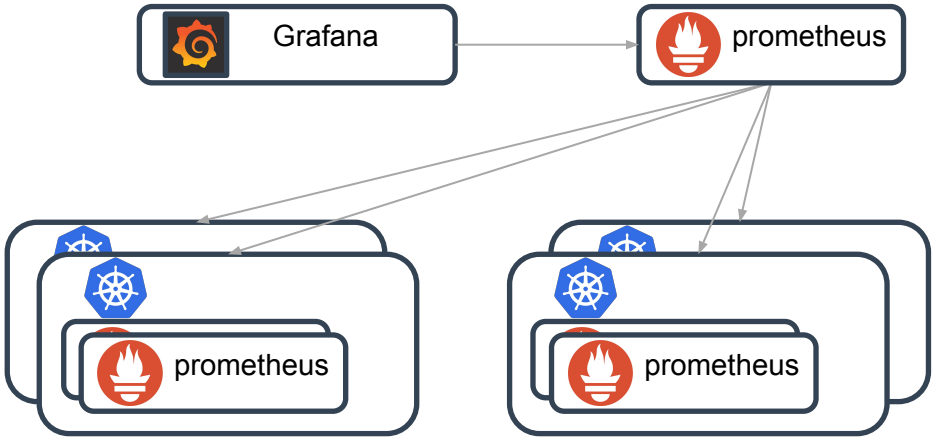
Which one to query?

How to do the switching when one of the replicas went down?



Recap

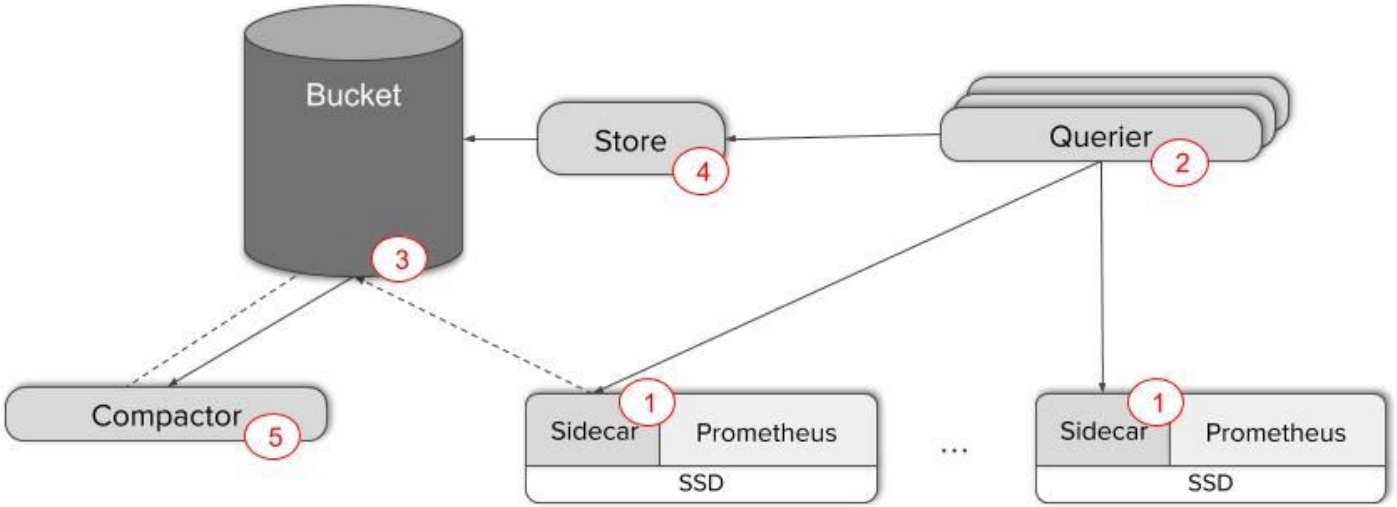
- Global view
- High availability
- Scaling



- Global view
- High availability
- Scaling

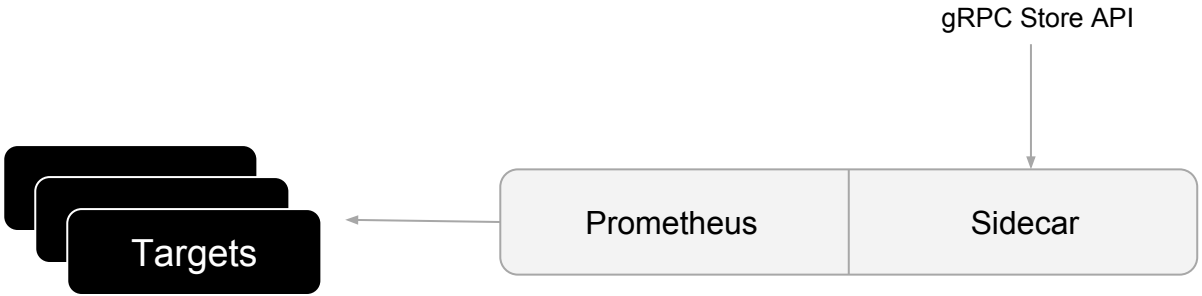
https://github.com/improbable-eng/thanos

Big picture

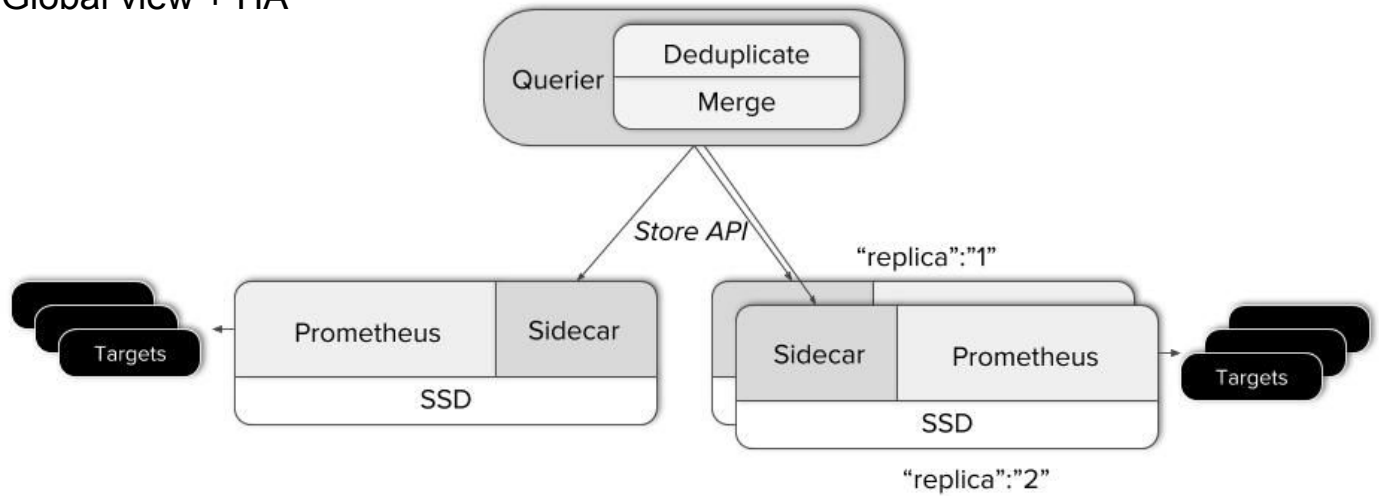


Global view - Sidecar

- Deployed along with each prometheus
- Serves prometheus data through gRPC-based thanos store API

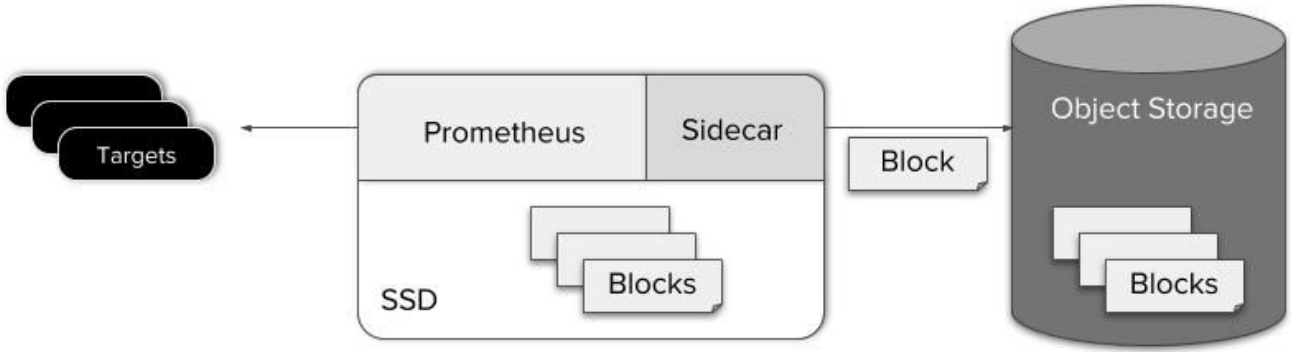


- Stateless, horizontally scalable.
- Fan out queries to all sidecars and stores. Merge and deduplicate query results.
- Global view + HA



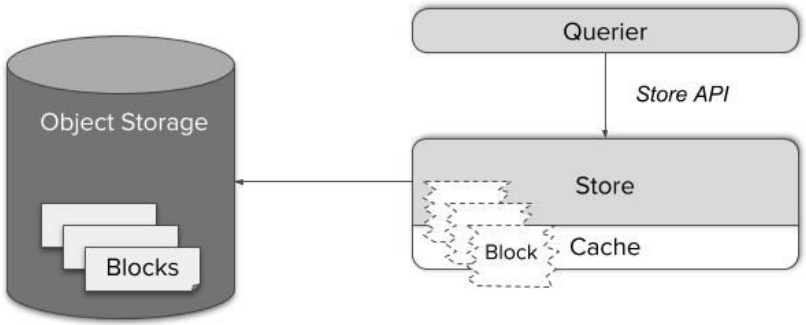
Unlimited Retention

- Prometheus packs data points for two hours into a block file.
- Sidecar uploads newly created block file to object store.



Retrieval from object storage

- Backing up is easy, how about retrieval?
- Thanos store as a data retrieval proxy.
- Implements store api as well.
- Pulling from object storage is expensive, caching is necessary.



Deploy on kubernetes - Prom + Sidecar

- https://github.com/improbable-eng/thanos/tree/master/kube/manifests
- Run as statefulset in a kubernetes cluster.
- Sidecar and prometheus run as two separate containers in the same pod.
- Sidecar exposes 10900 for gossip between thanos components

```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: prometheus
spec:
  serviceName: "prometheus"
  replicas: 2
  template:
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - prometheus
            topologyKey: kubernetes.io/hostname
      containers:
      - name: prometheus
        image: quay.io/prometheus/prometheus:v2.0.0
        args:
        - "--config.file=/etc/prometheus-shared/prometheus.yml"
        - "--storage.tsdb.path=/var/prometheus"
        - "--storage.tsdb.min-block-duration=2h"
        - "--storage.tsdb.max-block-duration=2h"
        - "--web.enable-lifecycle"
        ports:
        - name: http
          containerPort: 9090
      - name: thanos-sidecar
        image: improbable/thanos:master
        args:
        - "sidecar"
        - "--tsdb.path=/var/prometheus"
        - "--prometheus.url=http://127.0.0.1:9090"
        - "--cluster.peers=thanos-peers.default.svc.cluster.local:10900"
        - "--reloader.config-file=/etc/prometheus/prometheus.yml.tmpl"
        - "--reloader.config-envsubst-file=/etc/prometheus-shared/prometheus.yml"
        ports:
        - name: http
          containerPort: 10902
        - name: grpc
          containerPort: 10901
        - name: cluster
          containerPort: 10900
```

Deploy on kubernetes - Querier

- Run as deployment in a kubernetes cluster.
- Stateless, scale as you like.
- Exposes 9090 for prometheus-like queries.
- Exposes 10900 for gossip between thanos components as well.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: thanos-query
  labels:
    app: thanos-query
    thanos-peer: "true"
spec:
  replicas: 2
  selector:
    matchLabels:
      app: thanos-query
      thanos-peer: "true"
  template:
    metadata:
      labels:
        app: thanos-query
        thanos-peer: "true"
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "10902"
    spec:
      containers:
      - name: thanos-query
        image: improbable/thanos:master
        args:
        - "query"
        - "--log.level=debug"
        - "--cluster.peers=thanos-peers.default.svc.cluster.local:10900"
        - "--query.replica-label=replica"
        ports:
        - name: http
          containerPort: 10902
        - name: grpc
          containerPort: 10901
        - name: cluster
          containerPort: 10900
```

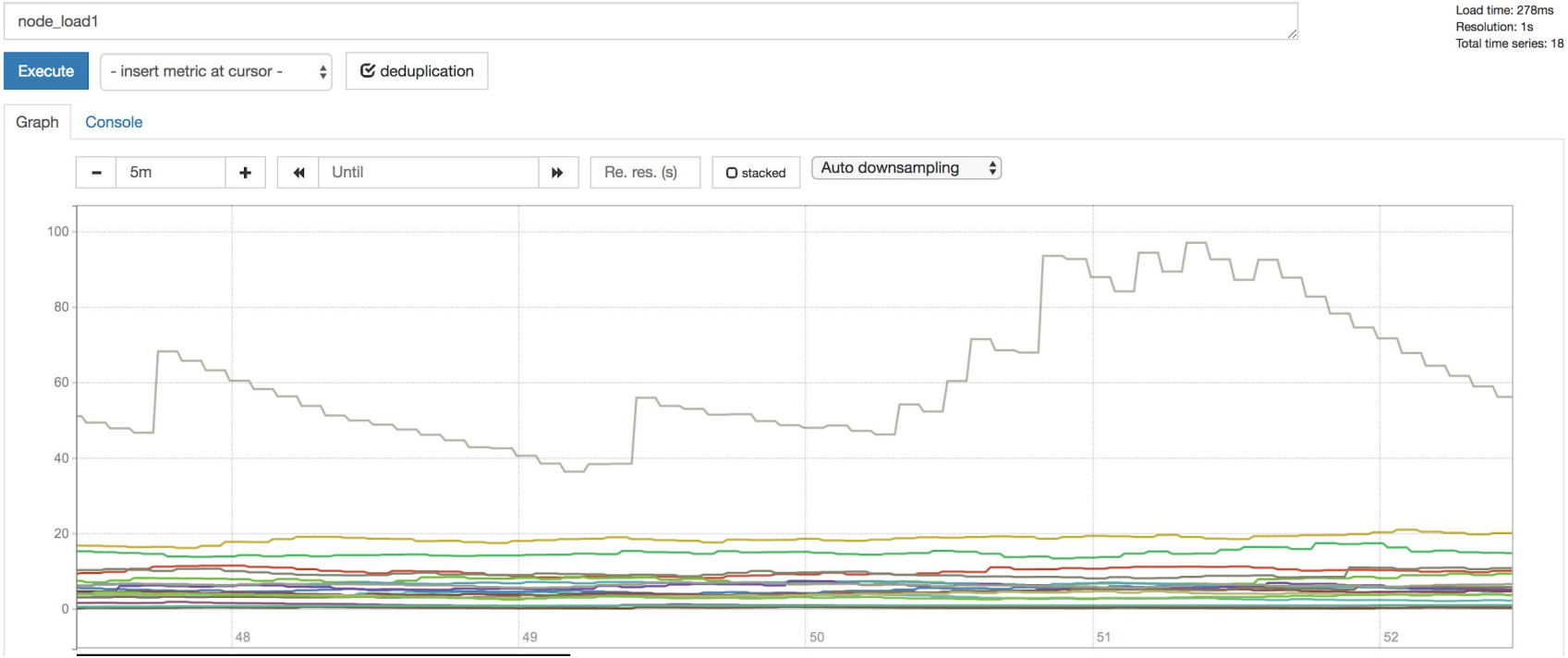
Deploy on kubernetes - Gossip Cluster

- Kubernetes headless service which resolves to all the thanos query, sidecar and store pod IPs in the cluster.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: thanos-peers
spec:
  type: ClusterIP
  clusterIP: None
  ports:
  - name: cluster
    port: 10900
    targetPort: cluster
  selector:
    # Useful endpoint for gathering all thanos components for common gossip cluster.
    thanos-peer: "true"
```

```
[root@o322v66]:~# nslookup thanos-peers.monitoring.svc.cluster.local 10.254.0.100
Server:     10.254.0.100
Address:    10.254.0.100#53

Name:   thanos-peers.monitoring.svc.cluster.local
Address: 192.168.73.39
Name:   thanos-peers.monitoring.svc.cluster.local
Address: 192.168.75.33
Name:   thanos-peers.monitoring.svc.cluster.local
Address: 192.168.68.9
Name:   thanos-peers.monitoring.svc.cluster.local
Address: 192.168.71.8
```

Voila!


node_load1
Load time: 278ms
Resolution: 1s
Total time series: 18
Execute
- insert metric at cursor -
deduplication
Graph
Console
5m
Until
Re. res. (s)
stacked
Auto downsampling
100
80
60
40
20
0
48
49
50
51
52

Monitoring your service mesh

CONDUIT

linkerd

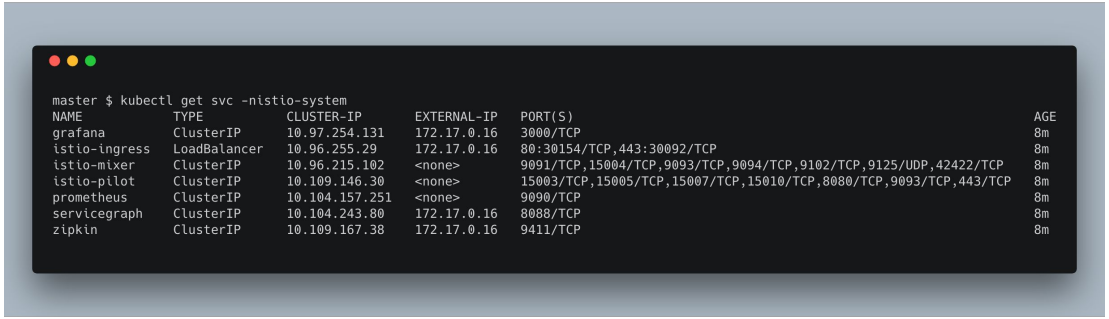
Monitoring your service mesh - istio

- https://www.katacoda.com/courses/istio/deploy-istio-on-kubernetes
- Mixer, pilot and envoy exposes prometheus metrics by default.
- Configure prometheus to collect data from istio components.
- Deploy example bookinfo app using istio.

```
    - job_name: 'istio-mesh'
      # Override the global default and scrape targets from this job every 5 seconds.
      scrape_interval: 5s

      kubernetes_sd_configs:
      - role: endpoints

      relabel_configs:
      - source_labels: [__meta_kubernetes_namespace, __meta_kubernetes_service_name, __meta_kubernetes_endpoint_port_name]
        action: keep
        regex: istio-system;istio-mixer;prometheus

    - job_name: 'envoy'
      # Override the global default and scrape targets from this job every 5 seconds.
      scrape_interval: 5s
      # metrics_path defaults to '/metrics'
      # scheme defaults to 'http'.

      kubernetes_sd_configs:
      - role: endpoints

      relabel_configs:
      - source_labels: [__meta_kubernetes_namespace, __meta_kubernetes_service_name, __meta_kubernetes_endpoint_port_name]
        action: keep
        regex: istio-system;istio-mixer;statsd-prom

    - job_name: 'mixer'
      # Override the global default and scrape targets from this job every 5 seconds.
      scrape_interval: 5s
      # metrics_path defaults to '/metrics'
      # scheme defaults to 'http'.

      kubernetes_sd_configs:
      - role: endpoints

      relabel_configs:
      - source_labels: [__meta_kubernetes_namespace, __meta_kubernetes_service_name, __meta_kubernetes_endpoint_port_name]
        action: keep
        regex: istio-system;istio-mixer;http-monitoring
```

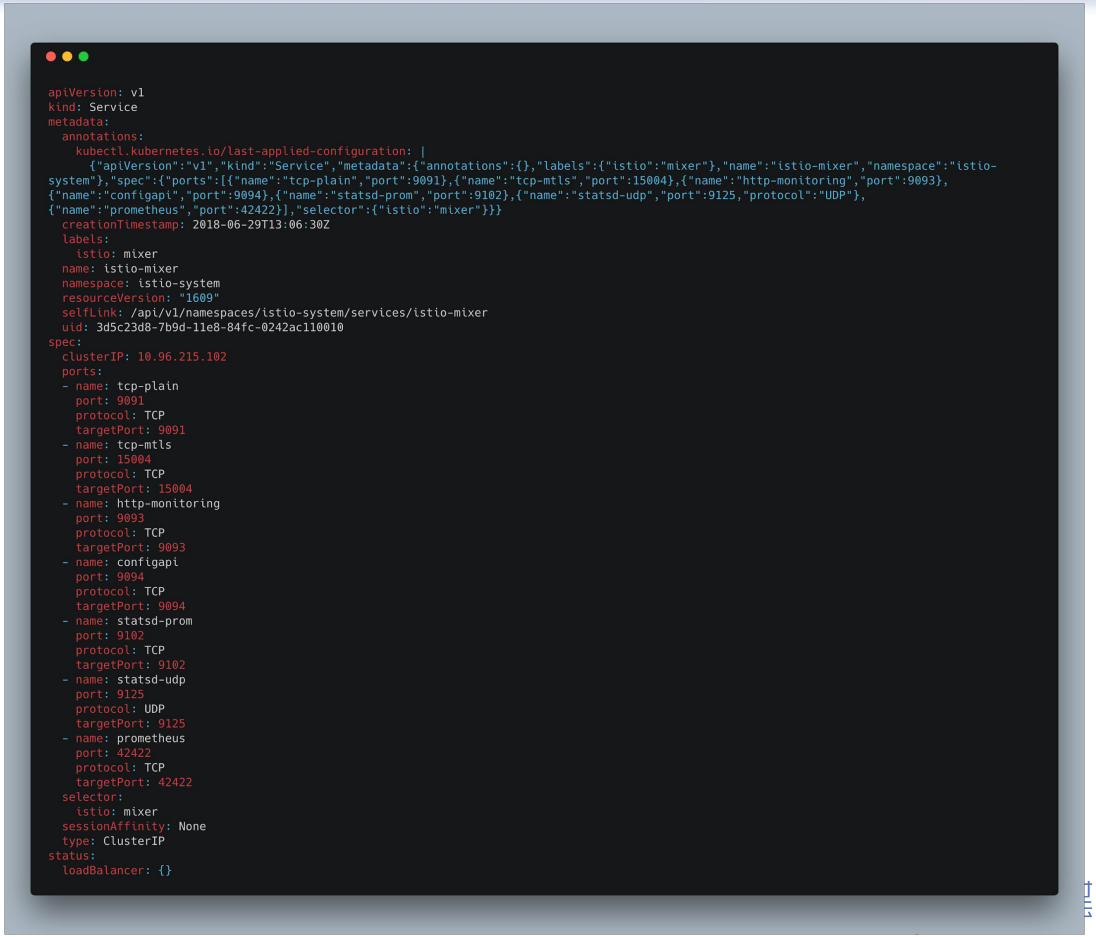
Monitoring your service mesh - istio

- https://www.katacoda.com/courses/istio/deploy-istio-on-kubernetes
- Mixer, pilot and envoy exposes prometheus metrics by default.
- Configure prometheus to collect data from istio components.
- Deploy example bookinfo app using istio.

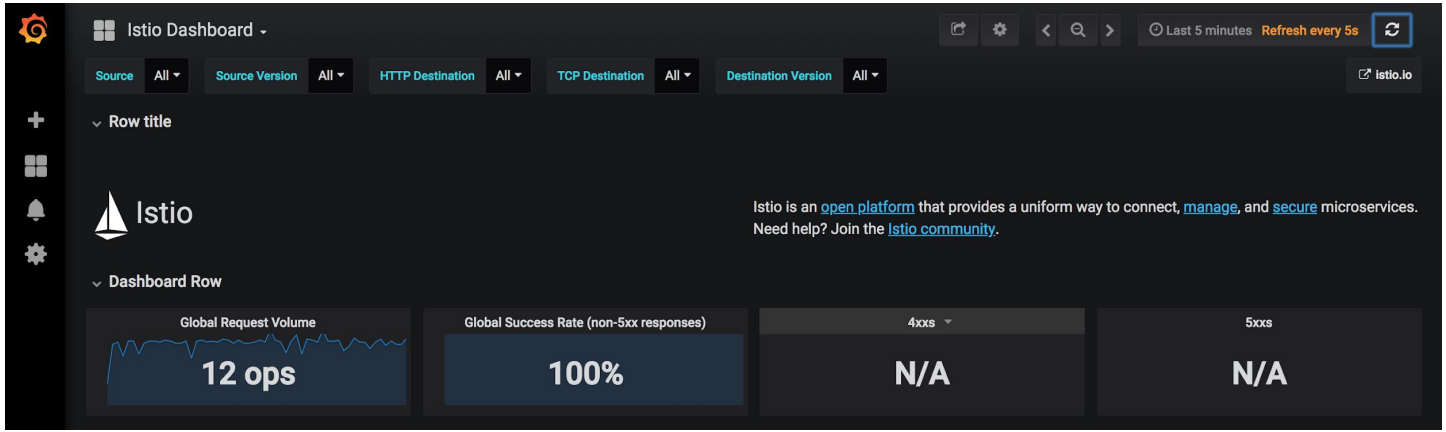


Monitoring your service mesh - istio

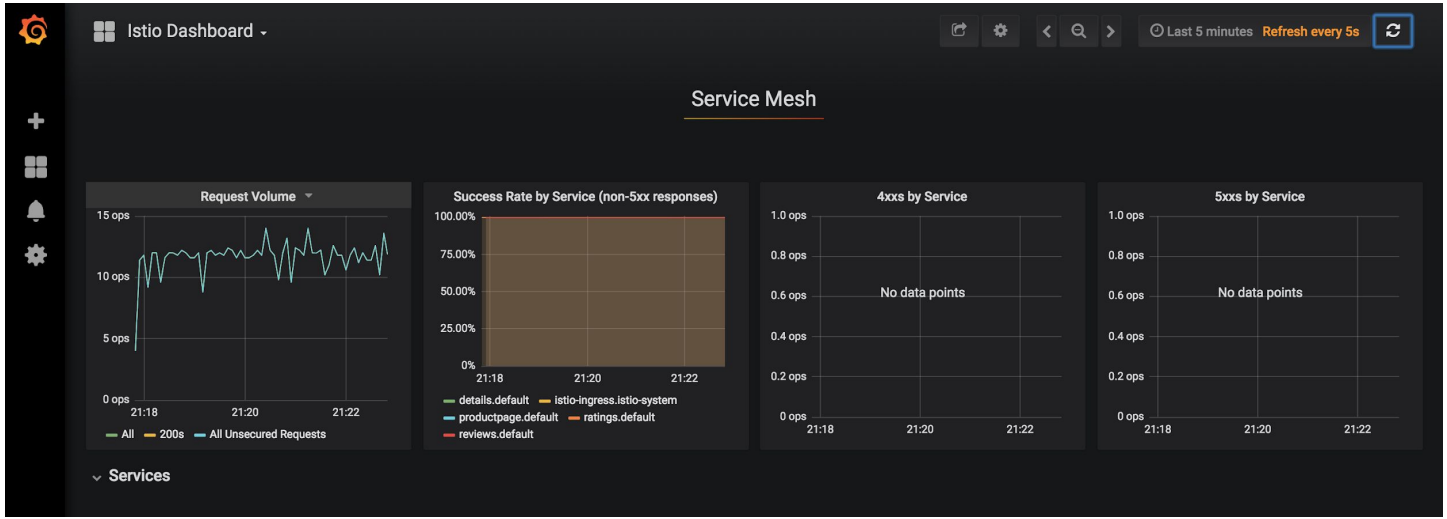
- https://www.katacoda.com/courses/istio/deploy-istio-on-kubernetes
- Mixer, pilot and envoy exposes prometheus metrics by default.
- Configure prometheus to collect data from istio components.
- Deploy example bookinfo app using istio.



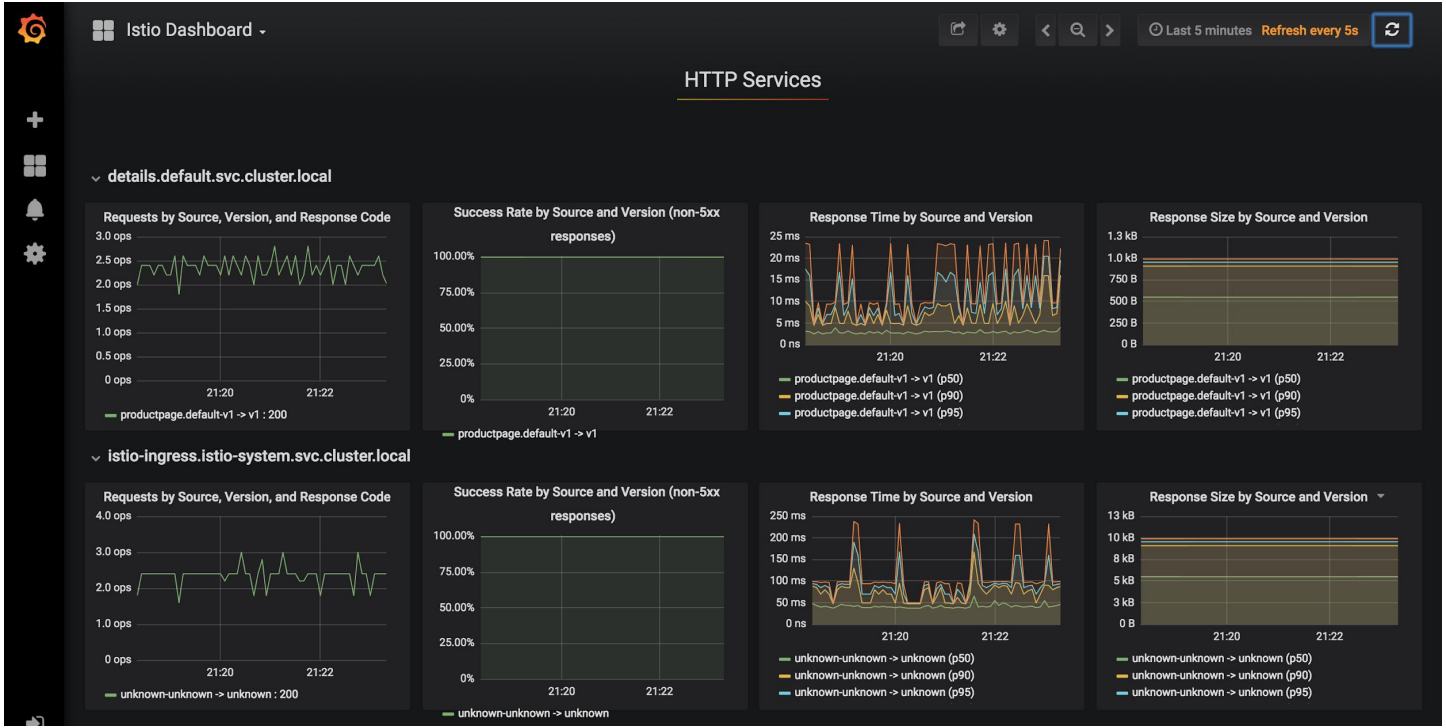
Build monitoring dashboard - Grafana



Build monitoring dashboard - Grafana



Build monitoring dashboard - Grafana



caicloud

才云

Thank you!