ServiceMesher

IT大咖说 知识共享平台

caicloud

才云

在 k8s 上部署高可用的 service mesh 监控

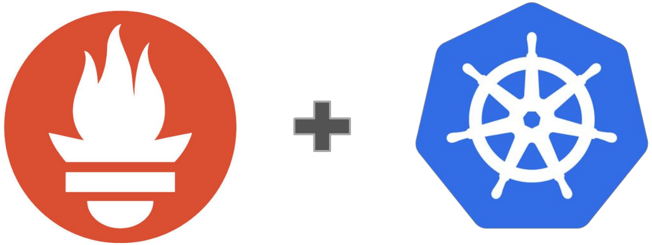
唐鹏程 才云科技

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.

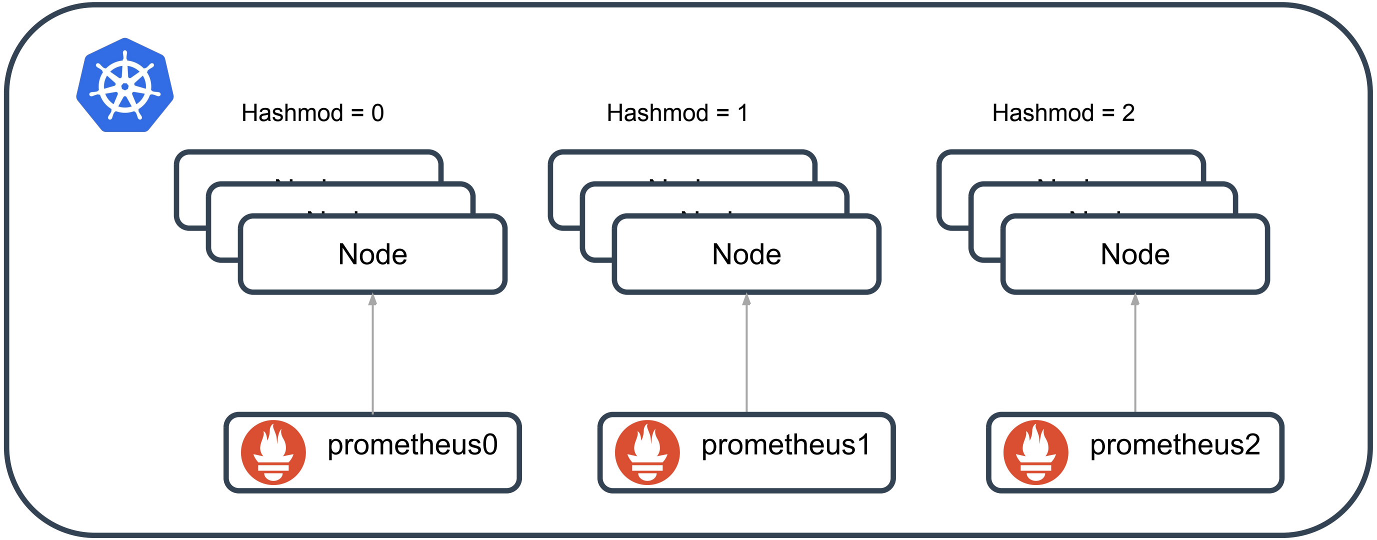


Prometheus 2.0 - deal with dynamic nature

- 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

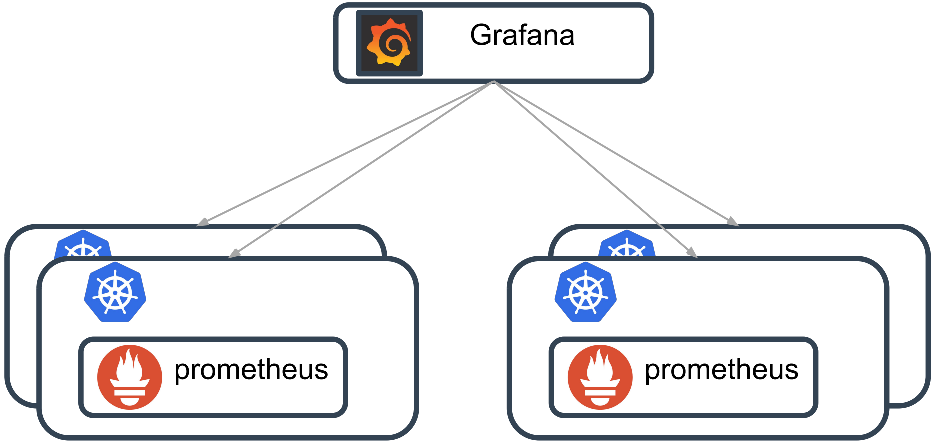
- In the old days…
  - one or more prometheus per cluster
  - hashmod sharding



Prometheus at scale

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

Almost works...



Prometheus at scale

- 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?

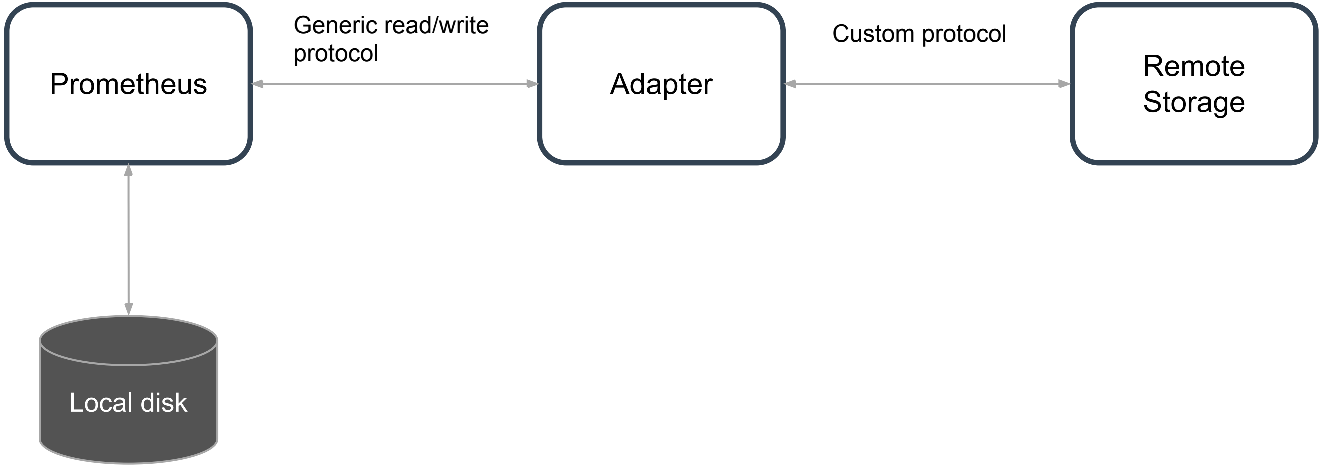
Prometheus at scale

- 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

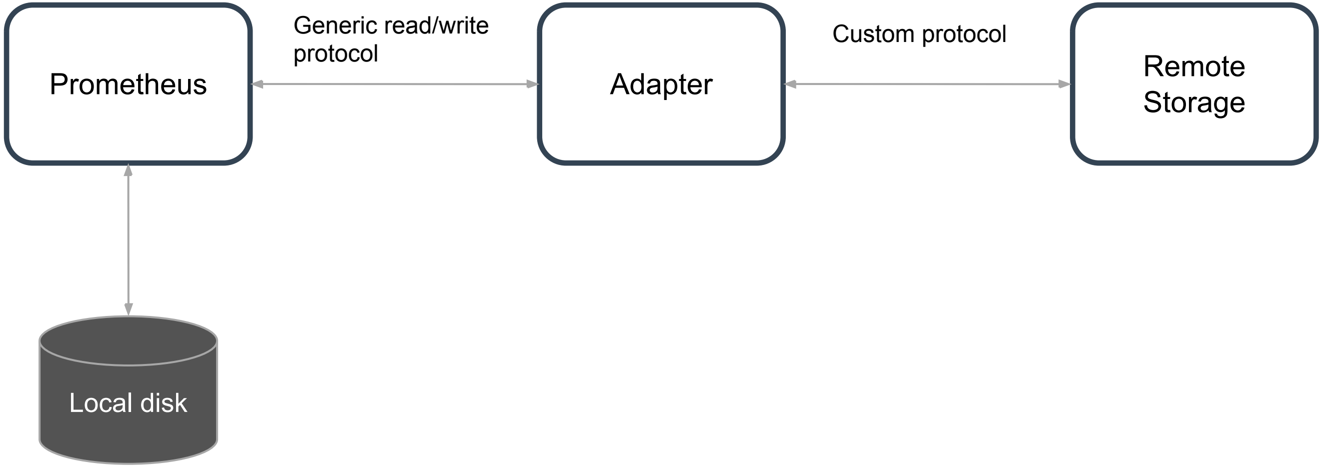
- Prometheus is by design **NOT** a persistent store.
- Have to live with all those DBs...



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

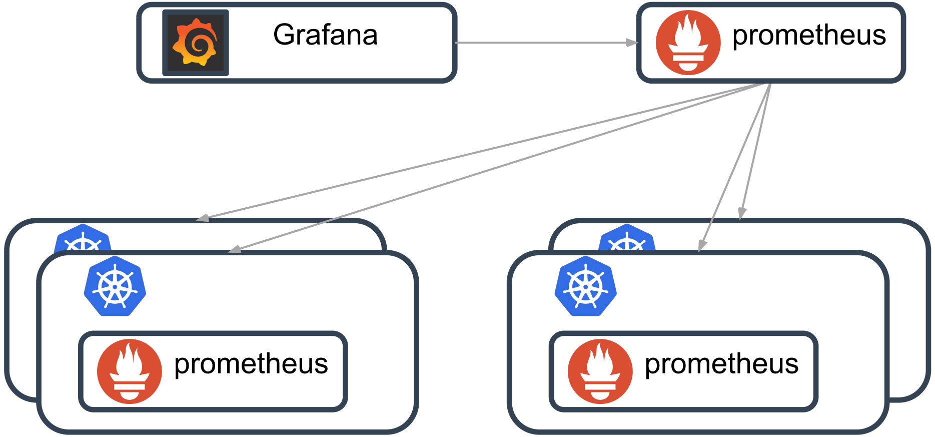
Global View

Old-schooled federation

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

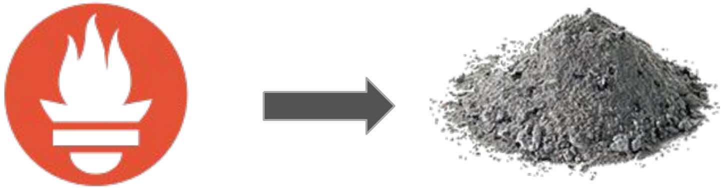
What’s wrong?

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



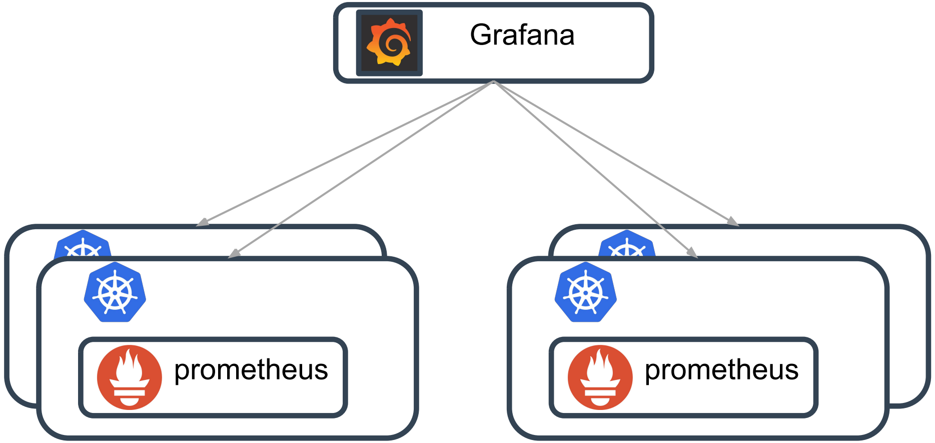
High Availability

- Too critical, should be the last one standing…
- But there are...
  - hardware failures
  - software failures
  - Maintenance and upgrades



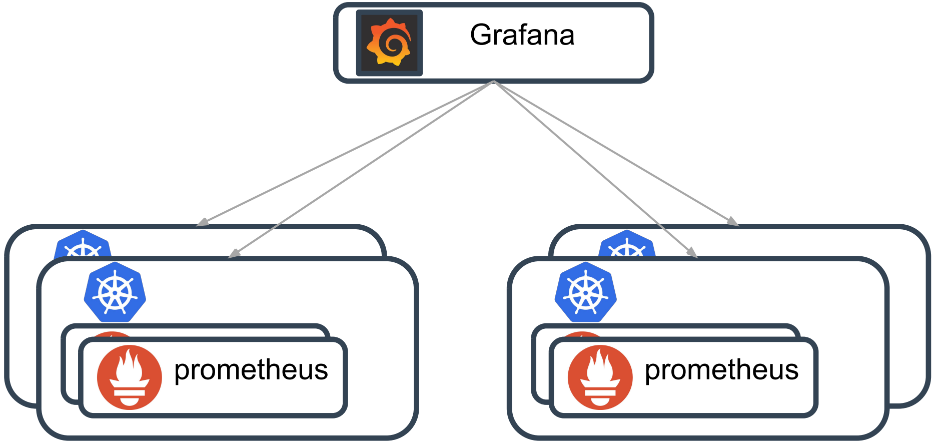
High Availability

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



High Availability

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

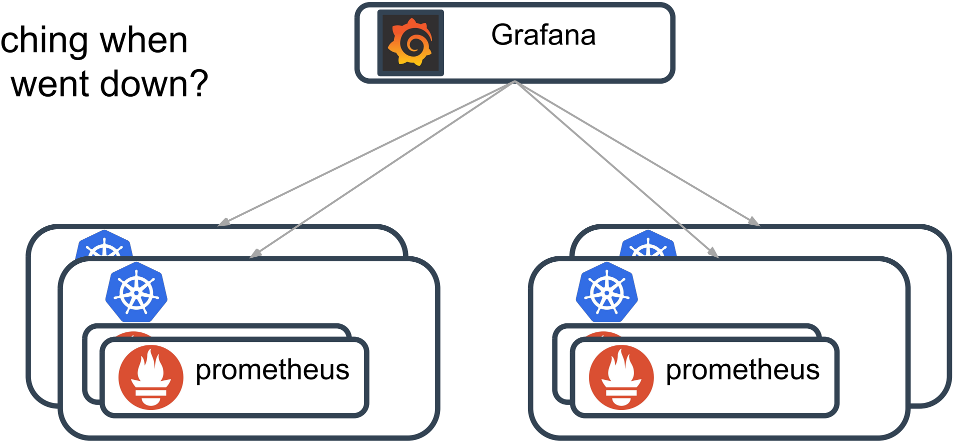


High Availability

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

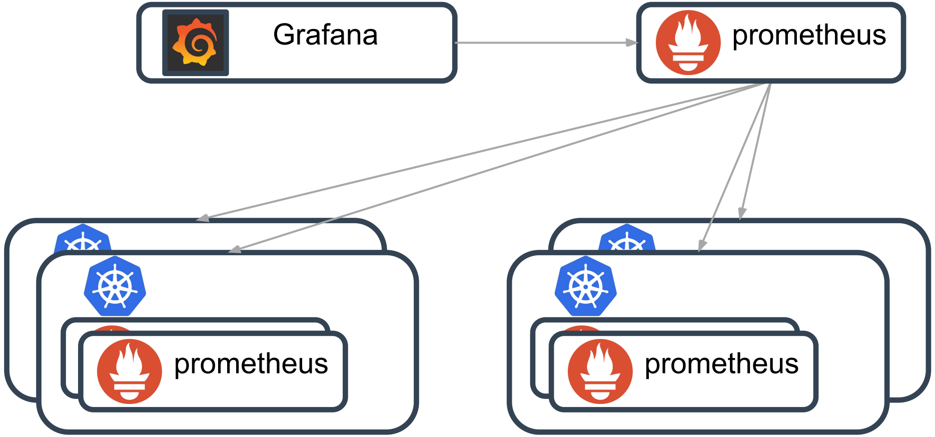
Which one to query?

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



Recap

- Global view
- High availability
- Scaling



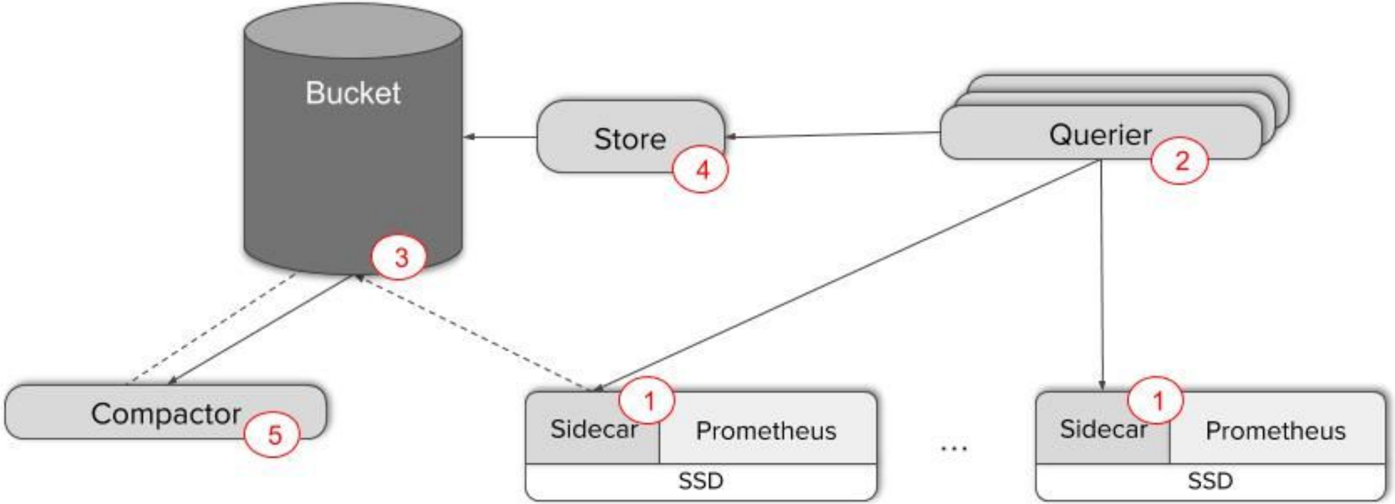
Thanos

- Global view
- High availability
- Scaling

Thanos

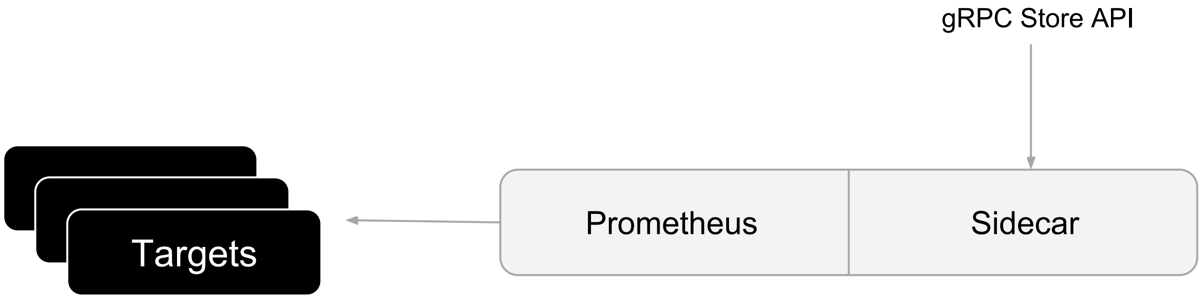
https://github.com/improbable-eng/thanos

Big picture


Bucket
Store
4
Querier
2
3
Compactor
5
Sidecar
1
Prometheus
SSD
...
Sidecar
1
Prometheus
SSD

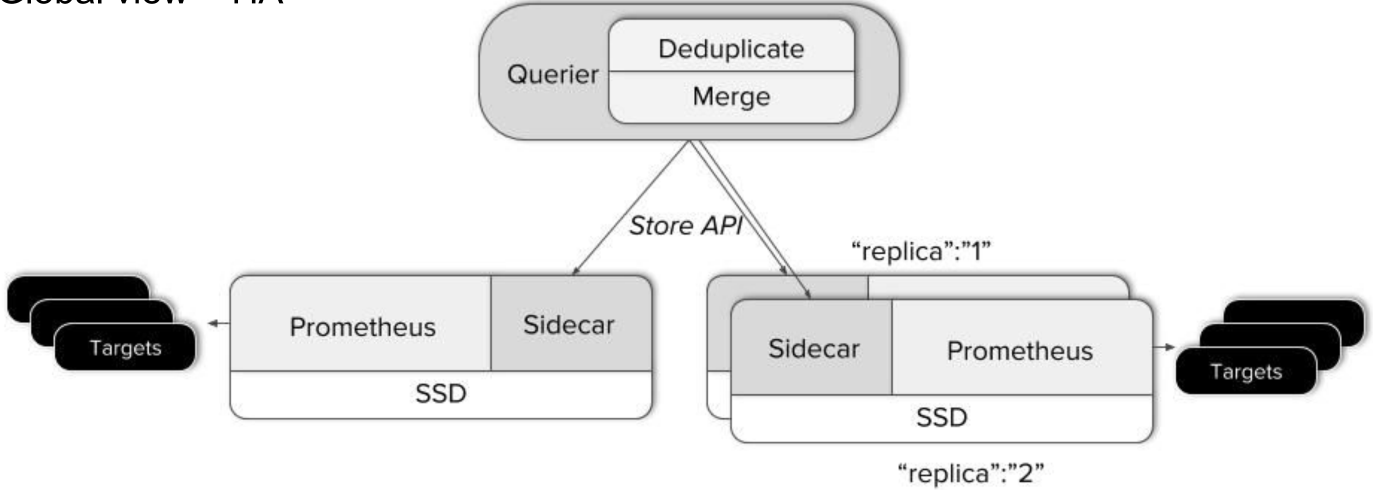
Global view - Sidecar

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



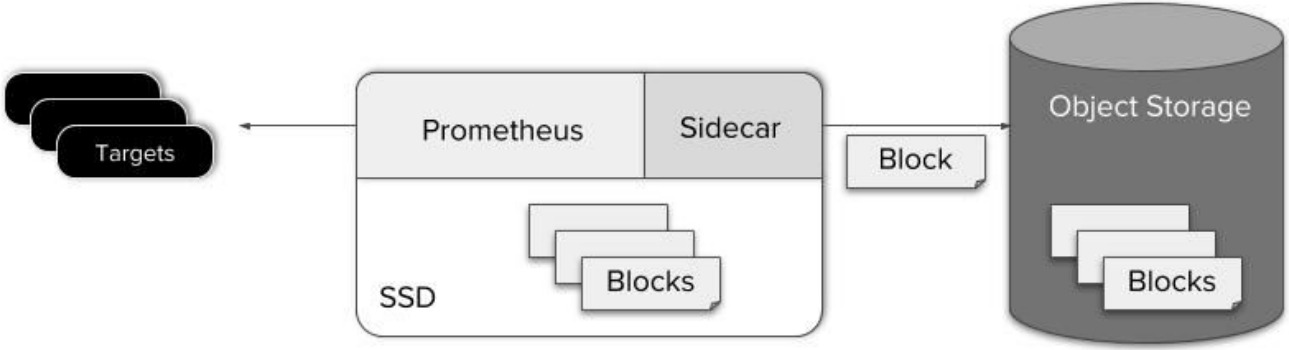
Global view - Querier

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



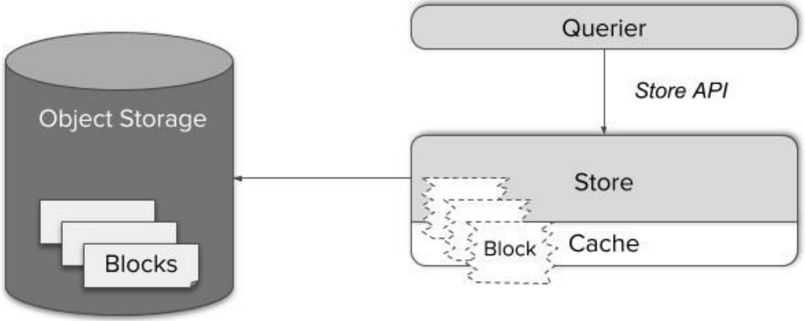
Unlimited Retention

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



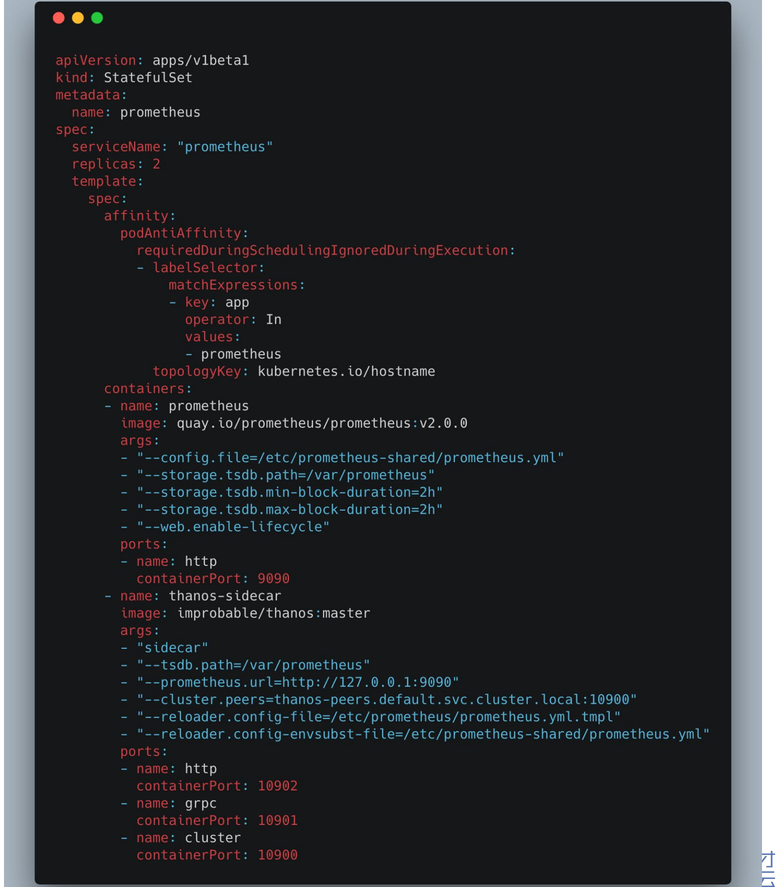
Retrieval from object storage

- 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

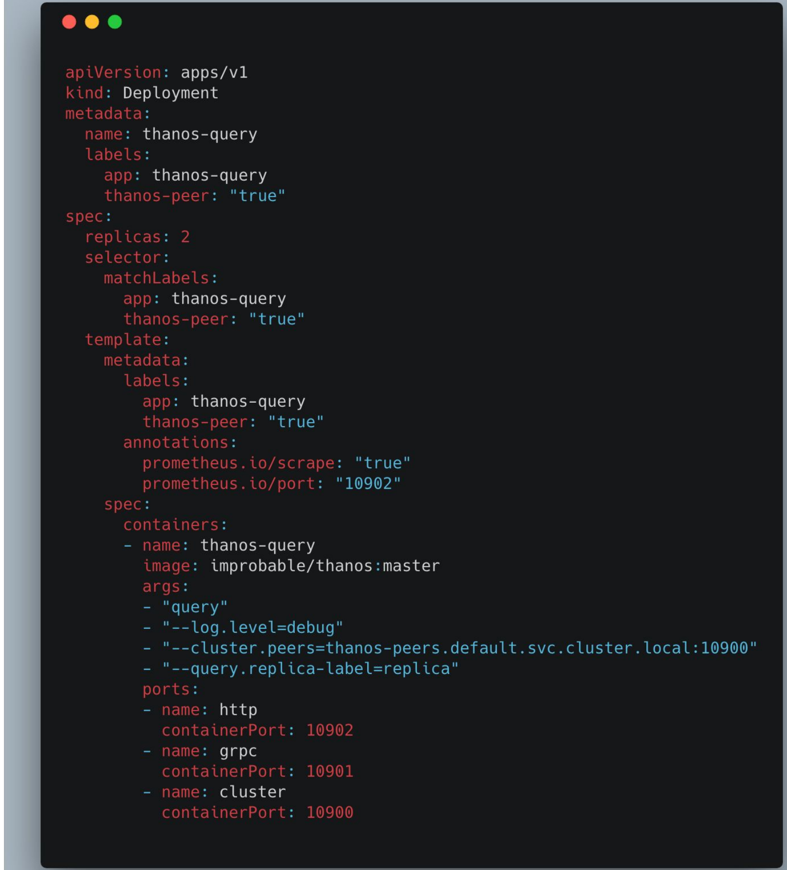
- https://github.com/improbable-eng/thanos/tree/master/kube/manifests
- Run as statefulset in a kubernetes cluster.
- Sidecar and prometheus run as two separate containers in the same pod.
- Sidecar exposes 10900 for gossip between thanos components



```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

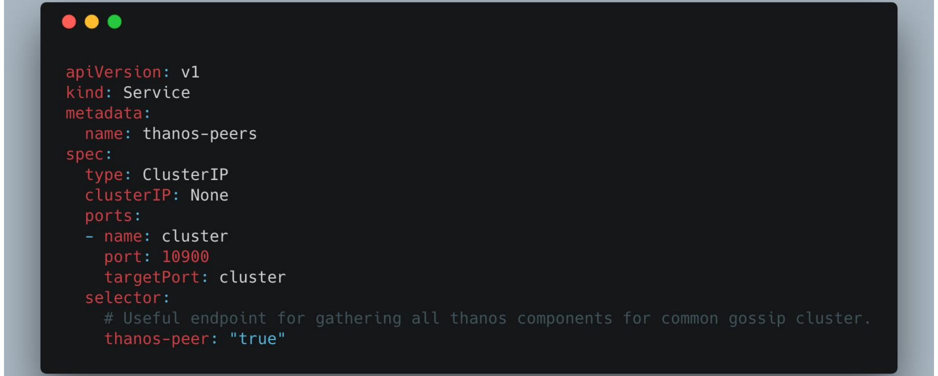
- Run as deployment in a kubernetes cluster.
- Stateless, scale as you like.
- Exposes 9090 for prometheus-like queries.
- Exposes 10900 for gossip between thanos components as well.



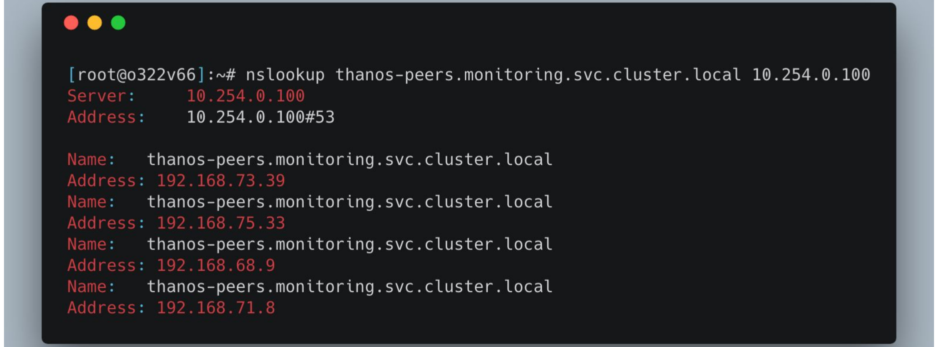
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: thanos-query
  labels:
    app: thanos-query
    thanos-peer: "true"
spec:
  replicas: 2
  selector:
    matchLabels:
      app: thanos-query
      thanos-peer: "true"
  template:
    metadata:
      labels:
        app: thanos-query
        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.

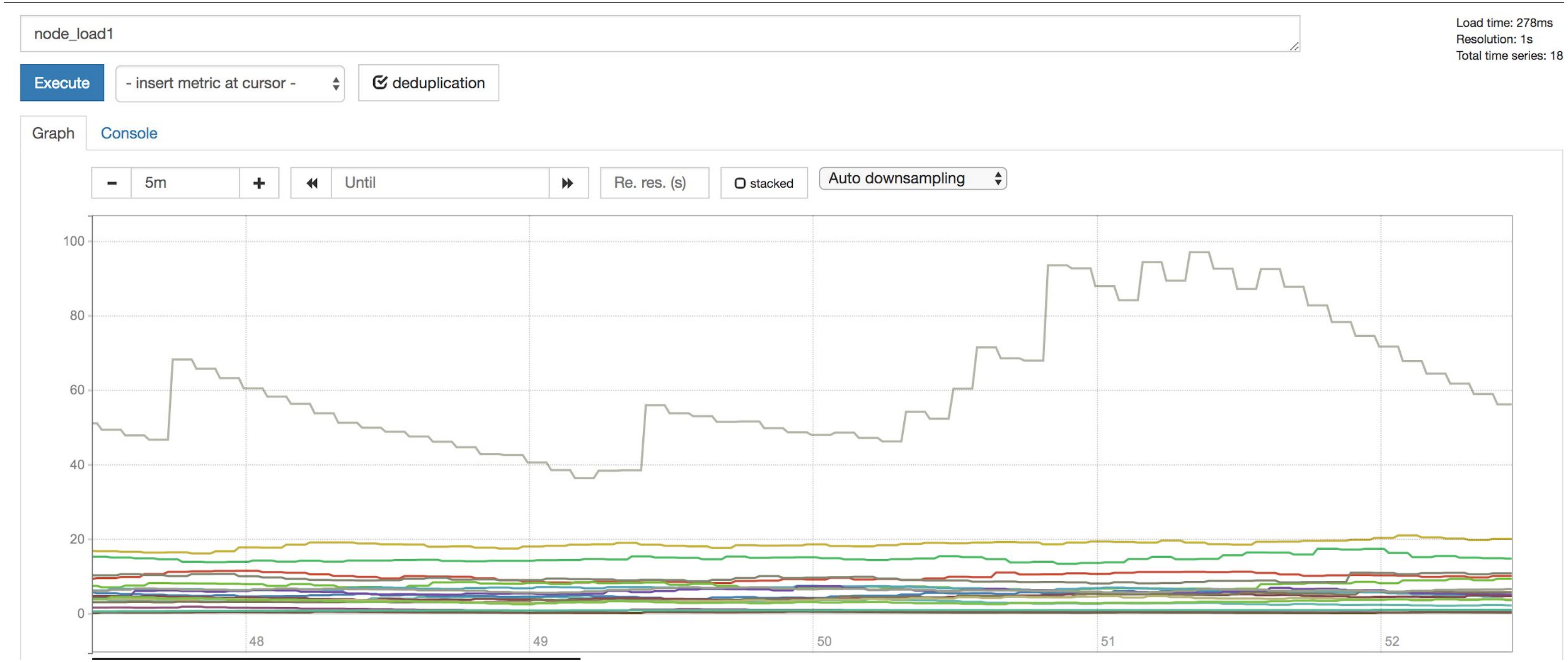


```yaml
apiVersion: v1
kind: Service
metadata:
  name: thanos-peers
spec:
  type: ClusterIP
  clusterIP: None
  ports:
  - name: cluster
    port: 10900
    targetPort: cluster
  selector:
    # Useful endpoint for gathering all thanos components for common gossip cluster.
    thanos-peer: "true"
```



```
[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
```

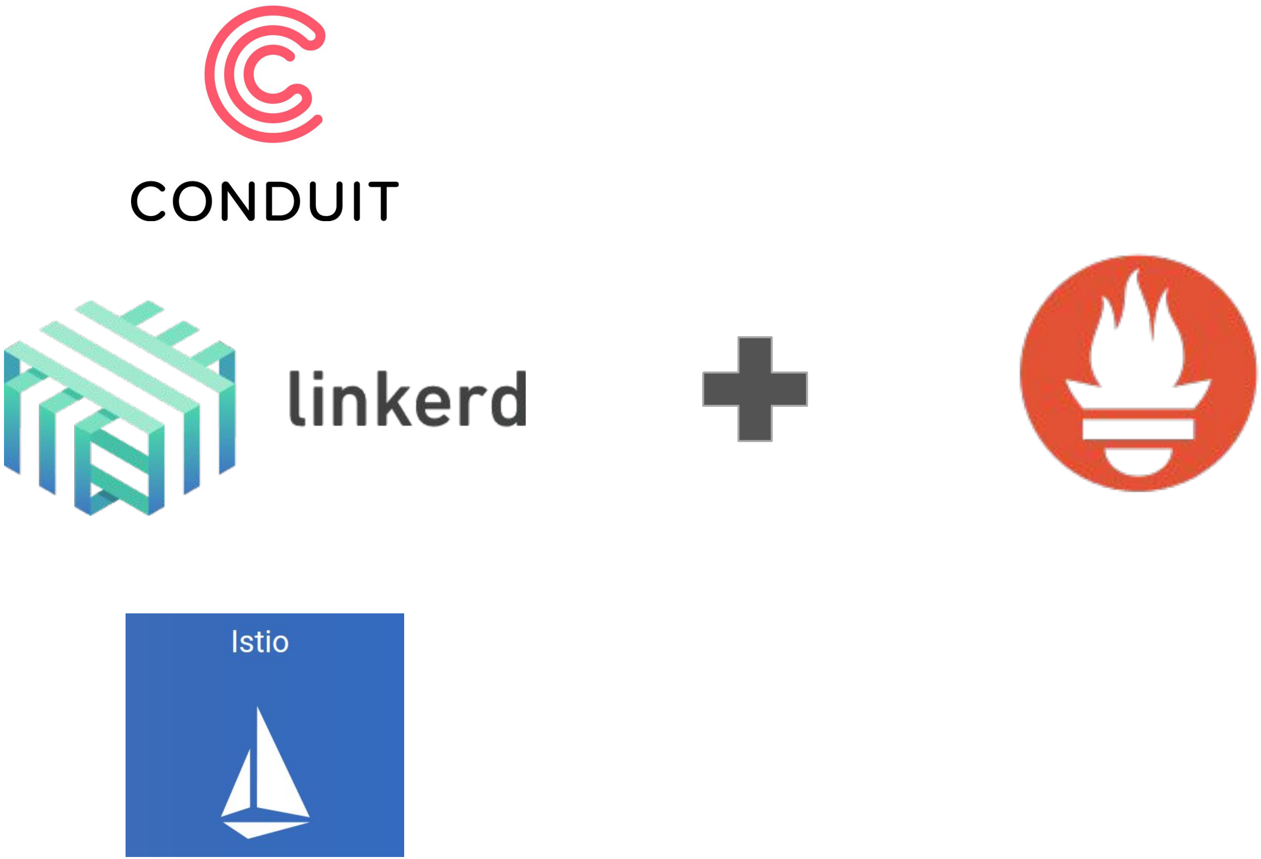
Voila!



node_load1

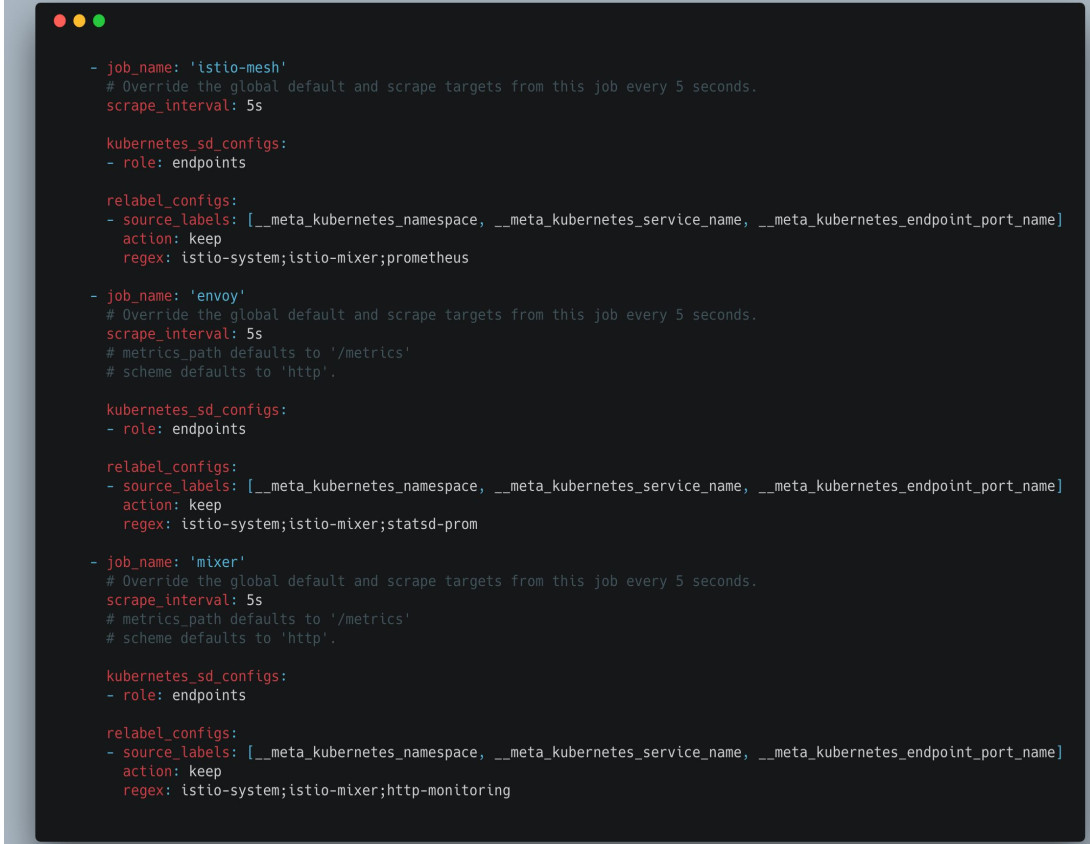
Load time: 278ms
Resolution: 1s
Total time series: 18

Monitoring your service mesh



Monitoring your service mesh - istio

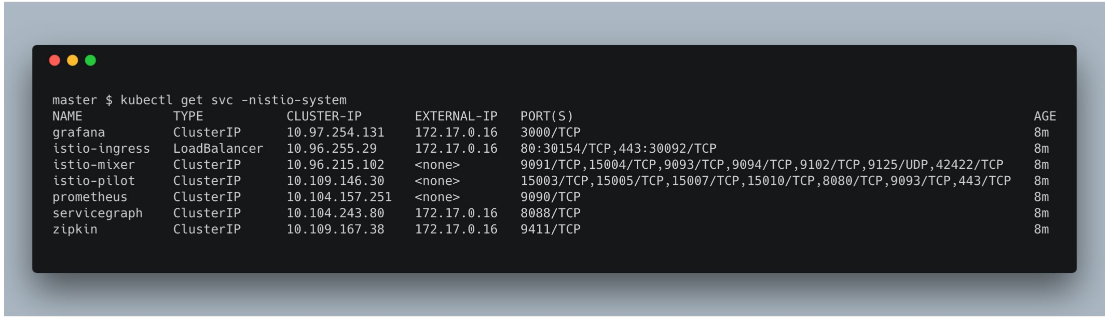
- https://www.katacoda.com/courses/istio/deploy-istio-on-kubernetes
- Mixer, pilot and envoy exposes prometheus metrics by default.
- Configure prometheus to collect data from istio components.
- Deploy example bookinfo app using istio.



```
- 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

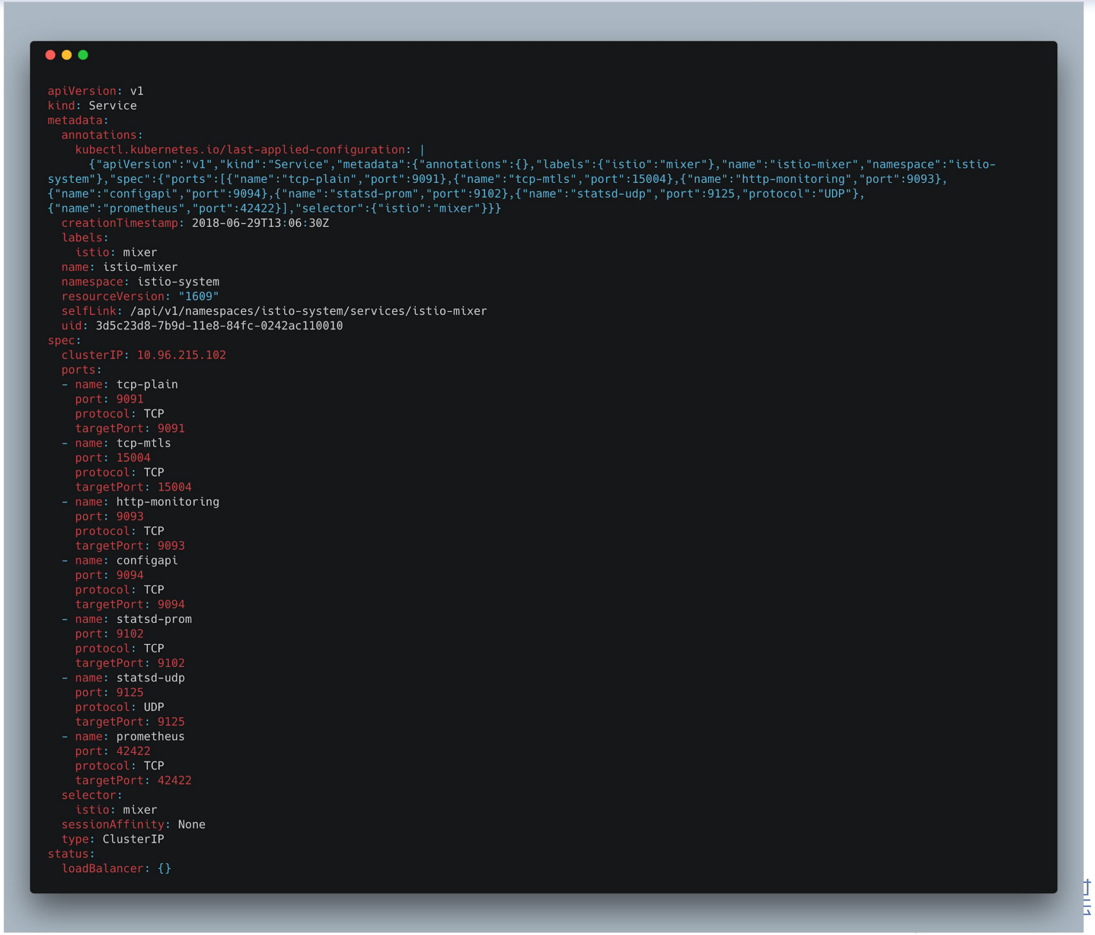
- https://www.katacoda.com/courses/istio/deploy-istio-on-kubernetes
- Mixer, pilot and envoy exposes prometheus metrics by default.
- Configure prometheus to collect data from istio components.
- Deploy example bookinfo app using istio.



```
master $ kubectl get svc -nistio-system
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                                                             AGE
grafana         ClusterIP      10.97.254.131    172.17.0.16   3000/TCP                                                            8m
istio-ingress   LoadBalancer   10.96.255.29     172.17.0.16   80:30154/TCP,443:30092/TCP                                          8m
istio-mixer     ClusterIP      10.96.215.102    <none>        9091/TCP,15004/TCP,9093/TCP,9094/TCP,9102/TCP,9125/UDP,42422/TCP    8m
istio-pilot     ClusterIP      10.109.146.30    <none>        15003/TCP,15005/TCP,15007/TCP,15010/TCP,8080/TCP,9093/TCP,443/TCP   8m
prometheus      ClusterIP      10.104.157.251   <none>        9090/TCP                                                            8m
servicegraph    ClusterIP      10.104.243.80    172.17.0.16   8088/TCP                                                            8m
zipkin          ClusterIP      10.109.167.38    172.17.0.16   9411/TCP                                                            8m
```

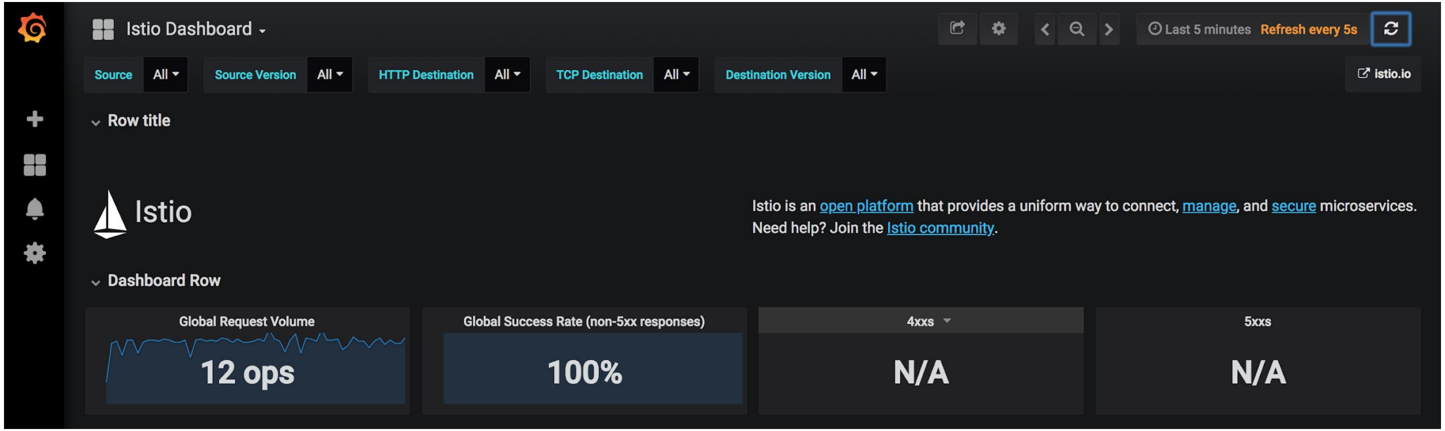
Monitoring your service mesh - istio

- 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.



```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"labels":{"istio":"mixer"},"name":"istio-mixer","namespace":"istio-
system"},"spec":{"ports":[{"name":"tcp-plain","port":9091},{"name":"tcp-mtls","port":15004},{"name":"http-monitoring","port":9093},
{"name":"configapi","port":9094},{"name":"statsd-prom","port":9102},{"name":"statsd-udp","port":9125,"protocol":"UDP"},
{"name":"prometheus","port":42422}],"selector":{"istio":"mixer"}}}
  creationTimestamp: 2018-06-29T13:06:30Z
  labels:
    istio: mixer
  name: istio-mixer
  namespace: istio-system
  resourceVersion: "1609"
  selfLink: /api/v1/namespaces/istio-system/services/istio-mixer
  uid: 3d5c23d8-7b9d-11e8-84fc-0242ac110010
spec:
  clusterIP: 10.96.215.102
  ports:
  - name: tcp-plain
    port: 9091
    protocol: TCP
    targetPort: 9091
  - name: tcp-mtls
    port: 15004
    protocol: TCP
    targetPort: 15004
  - name: http-monitoring
    port: 9093
    protocol: TCP
    targetPort: 9093
  - name: configapi
    port: 9094
    protocol: TCP
    targetPort: 9094
  - name: statsd-prom
    port: 9102
    protocol: TCP
    targetPort: 9102
  - name: statsd-udp
    port: 9125
    protocol: UDP
    targetPort: 9125
  - name: prometheus
    port: 42422
    protocol: TCP
    targetPort: 42422
  selector:
    istio: mixer
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
```

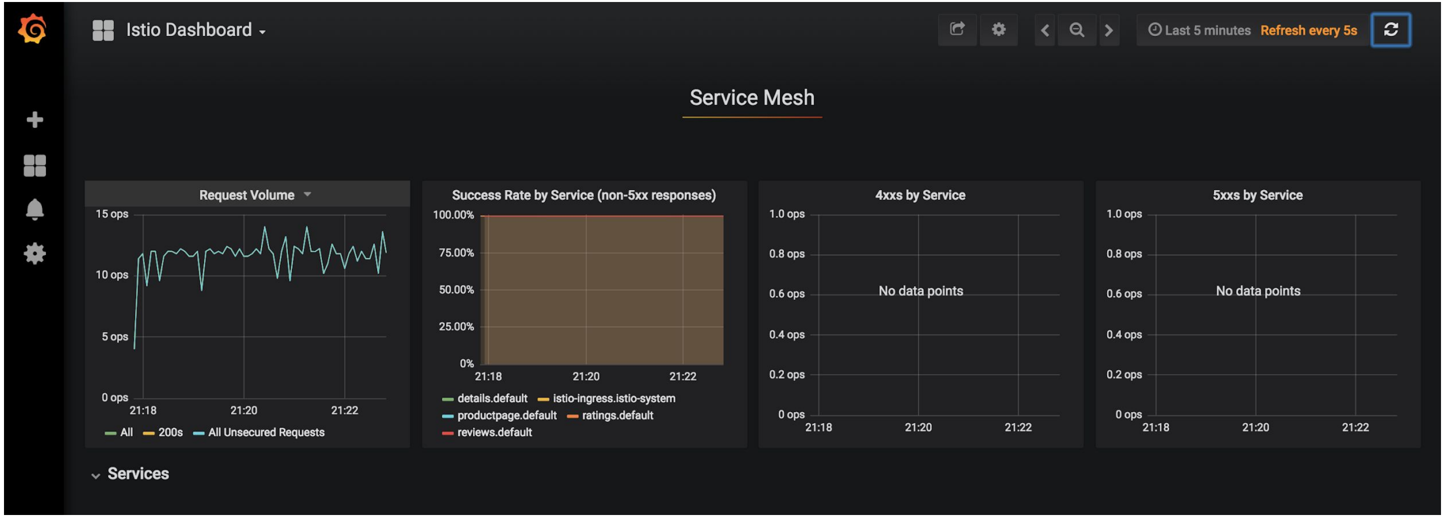
Build monitoring dashboard - Grafana



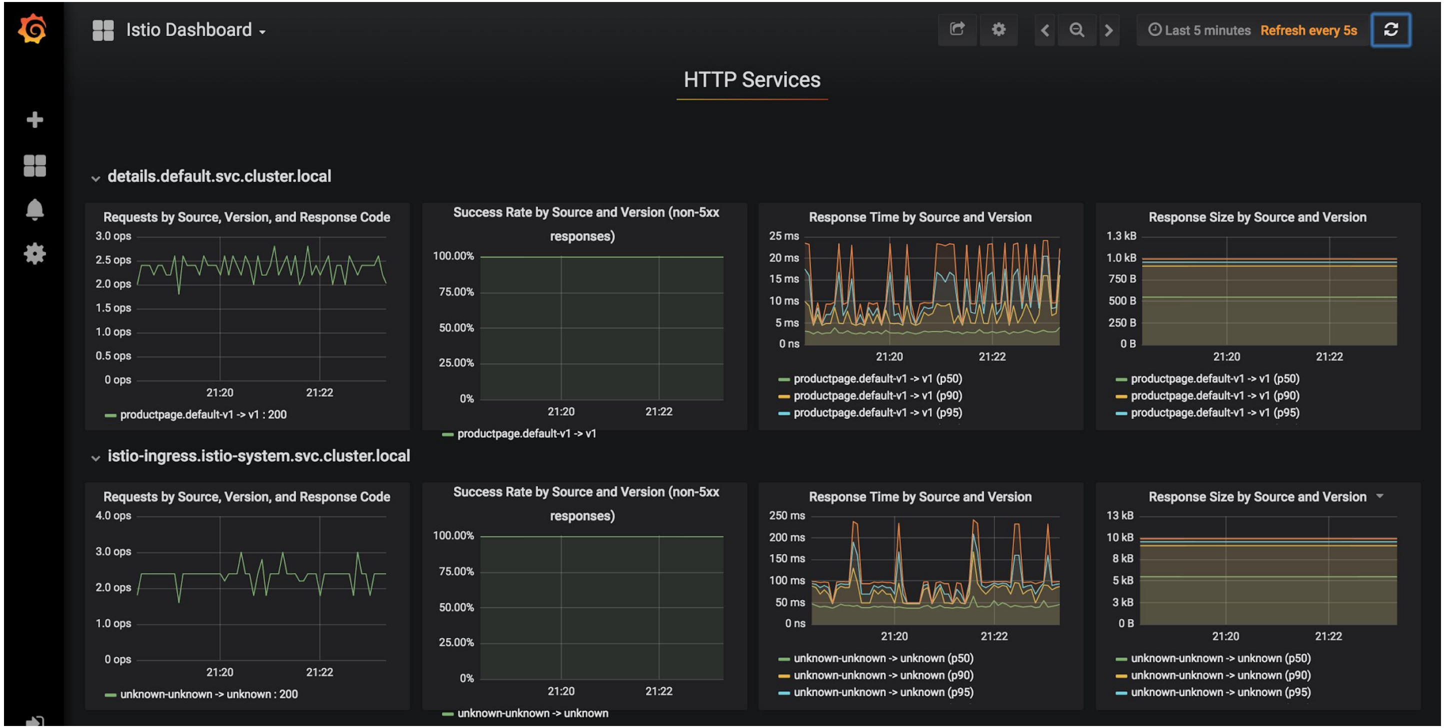
Source: All
Source Version: All
HTTP Destination: All
TCP Destination: All
Destination Version: All

Istio is an open platform that provides a uniform way to connect, manage, and secure microservices. Need help? Join the Istio community.

Build monitoring dashboard - Grafana



Build monitoring dashboard - Grafana


Istio Dashboard
Last 5 minutes Refresh every 5s
HTTP Services
details.default.svc.cluster.local
Requests by Source, Version, and Response Code
3.0 ops
2.5 ops
2.0 ops
1.5 ops
1.0 ops
0.5 ops
0 ops
21:20
21:22
productpage.default-v1 -> v1 : 200
Success Rate by Source and Version (non-5xx responses)
100.00%
75.00%
50.00%
25.00%
0%
21:20
21:22
productpage.default-v1 -> v1
Response Time by Source and Version
25 ms
20 ms
15 ms
10 ms
5 ms
0 ns
21:20
21:22
productpage.default-v1 -> v1 (p50)
productpage.default-v1 -> v1 (p90)
productpage.default-v1 -> v1 (p95)
Response Size by Source and Version
1.3 kB
1.0 kB
750 B
500 B
250 B
0 B
21:20
21:22
productpage.default-v1 -> v1 (p50)
productpage.default-v1 -> v1 (p90)
productpage.default-v1 -> v1 (p95)
istio-ingress.istio-system.svc.cluster.local
Requests by Source, Version, and Response Code
4.0 ops
3.0 ops
2.0 ops
1.0 ops
0 ops
21:20
21:22
unknown-unknown -> unknown : 200
Success Rate by Source and Version (non-5xx responses)
100.00%
75.00%
50.00%
25.00%
0%
21:20
21:22
unknown-unknown -> unknown
Response Time by Source and Version
250 ms
200 ms
150 ms
100 ms
50 ms
0 ns
21:20
21:22
unknown-unknown -> unknown (p50)
unknown-unknown -> unknown (p90)
unknown-unknown -> unknown (p95)
Response Size by Source and Version
13 kB
10 kB
8 kB
5 kB
3 kB
0 B
21:20
21:22
unknown-unknown -> unknown (p50)
unknown-unknown -> unknown (p90)
unknown-unknown -> unknown (p95)

caicloud
才云

Thank you!