KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

China 2024

Beyond the Basics: Towards making Thanos production-ready

Benjamin Huo, Manager of the Observability Team

Junhao Zhang, Sr. SE of the Observability Team

—— KubeSphere of QingCloud Technologies

Agenda

- Introducing Thanos
- Introducing Whizard
- The production adoption of Whizard in KubeSphere
- Roadmap


KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Introducing Thanos

Global Query View

Scale your Prometheus setup by enabling querying of your Prometheus metrics across multiple Prometheus servers and clusters.

Unlimited Retention

Extend the system with the object storage of your choice to store your metrics for unlimited time. Supports GCP, S3, Azure, Swift and Tencent COS.

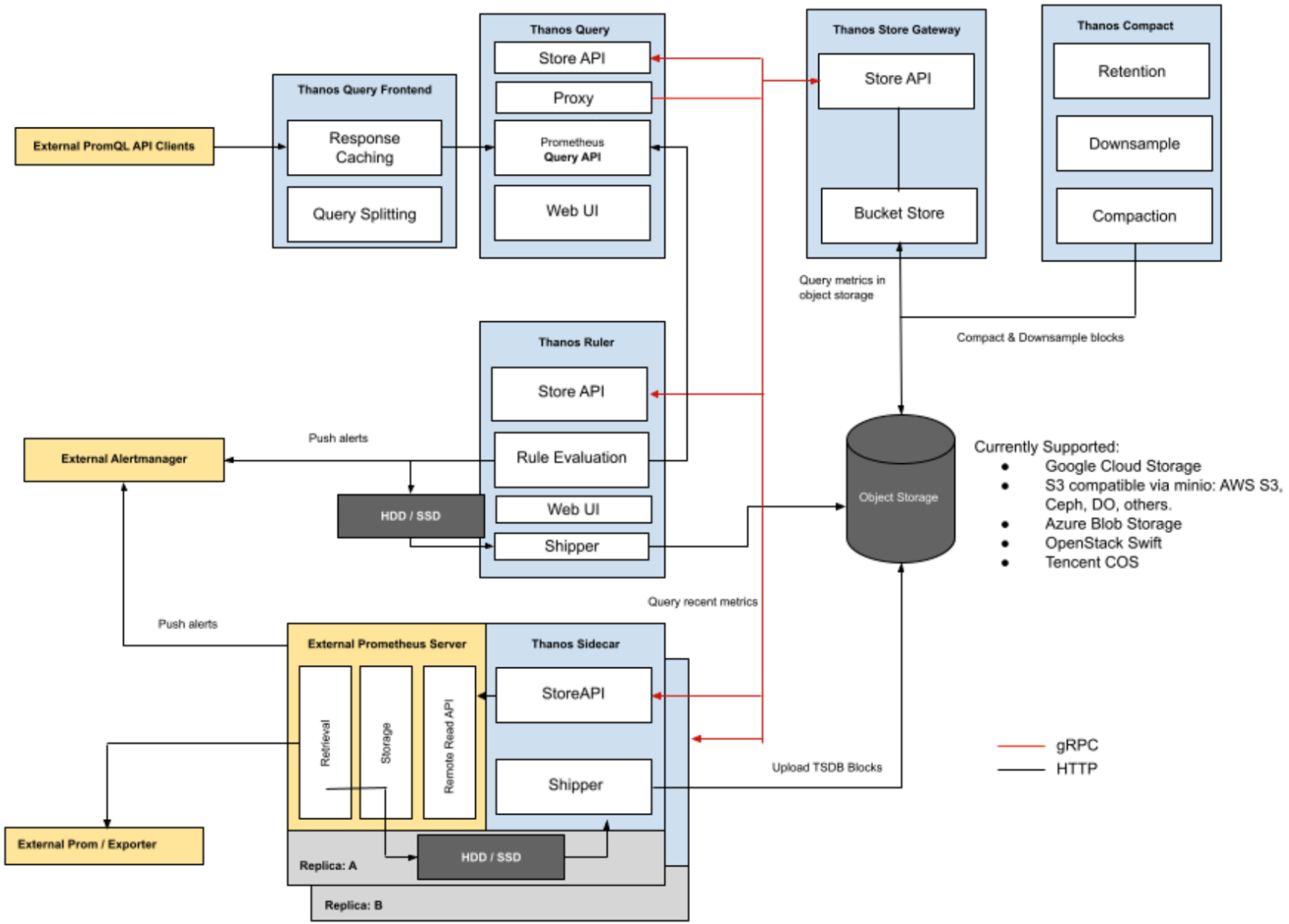
Prometheus Compatible

Use the same tools you love, such as Grafana and others, that support the Prometheus Query API.

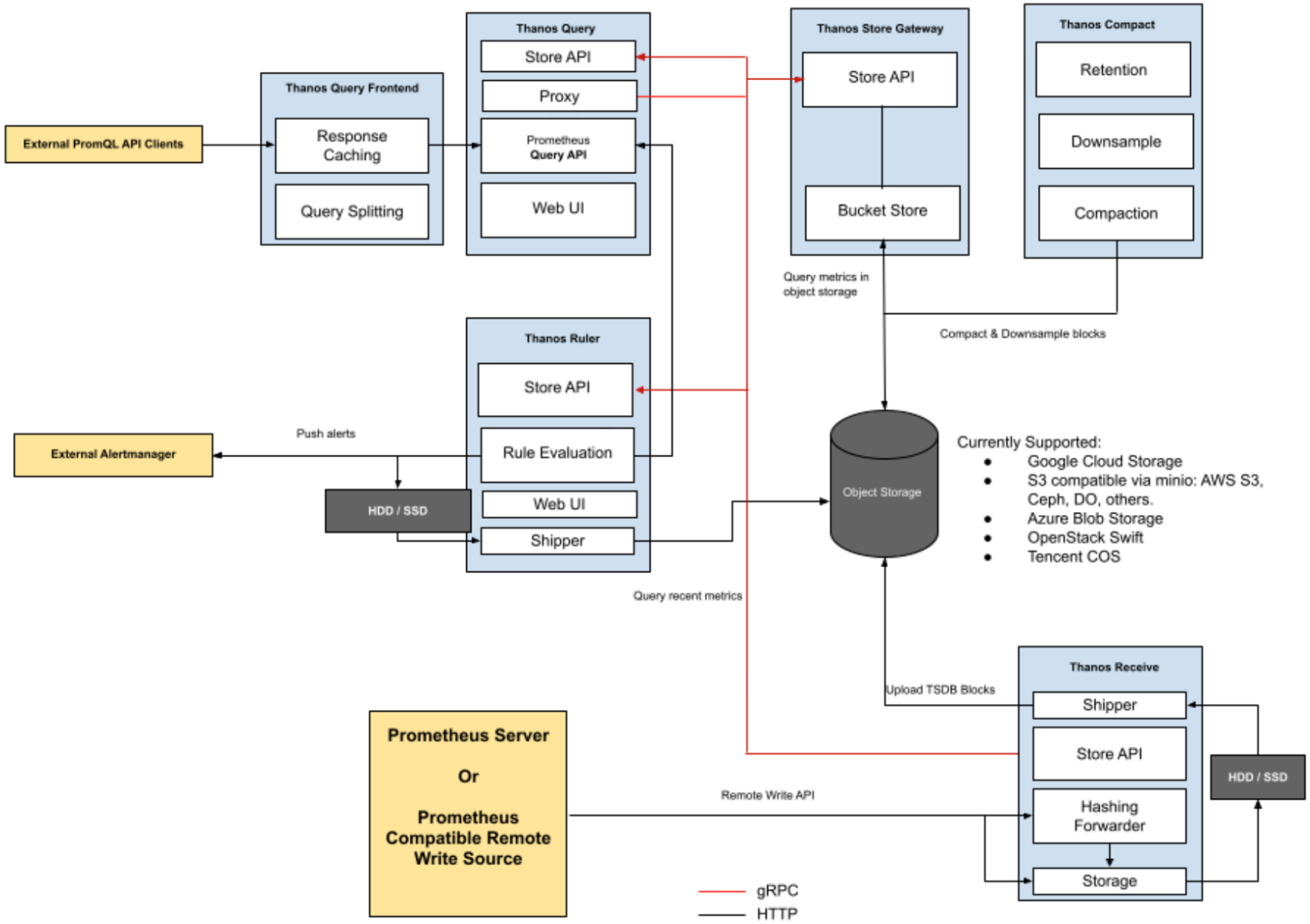
Downsampling & Compaction

Downsample historical data for massive query speedup when querying large time ranges or configure complex retention policies.

Deployment with Sidecar for Kubernetes:



Deployment with Receive in order to scale out or implement with other remote write compatible sources:



How do you customize & deploy Thanos?

Can I use kube-thanos ?

- Maintained by Thanos maintainers
- Customize Thanos setup with jsonnet
- Deploy with yaml manifests

Cons:

- Jsonnet is more developer friendly, not user friendly
- The sample deployment is not for production:
  - Only one instance of query, router, ingester, and store with one replica each.
  - No compactor, queryFrontend
  - The hashring contains only the default soft tenant
  - …
- Too many efforts needed to make it production-ready

How do you customize & deploy Thanos?

What about the Thanos Helm Chart ?

- Maintained by bitnami
- Customize Thanos setup with helm values
- Deploy with helm

Cons:

- Too many values to customize
- Thanos Stateful Components are scaled by CPU/Memory (HPA)
- No Tenant related setting
- Hashring is configured manually

What else do you need for production?

What's still missing?

- Create and maintain Thanos components and configs with ease like CRDs
- Tenant configs can be simpler without configuring the hashring manually
- Support deploying Thanos to mutiple K8s cluster
- Evaluating recording rules for each tenant and remote write the metrics back on tenant basis
- …

What else do you need for production?

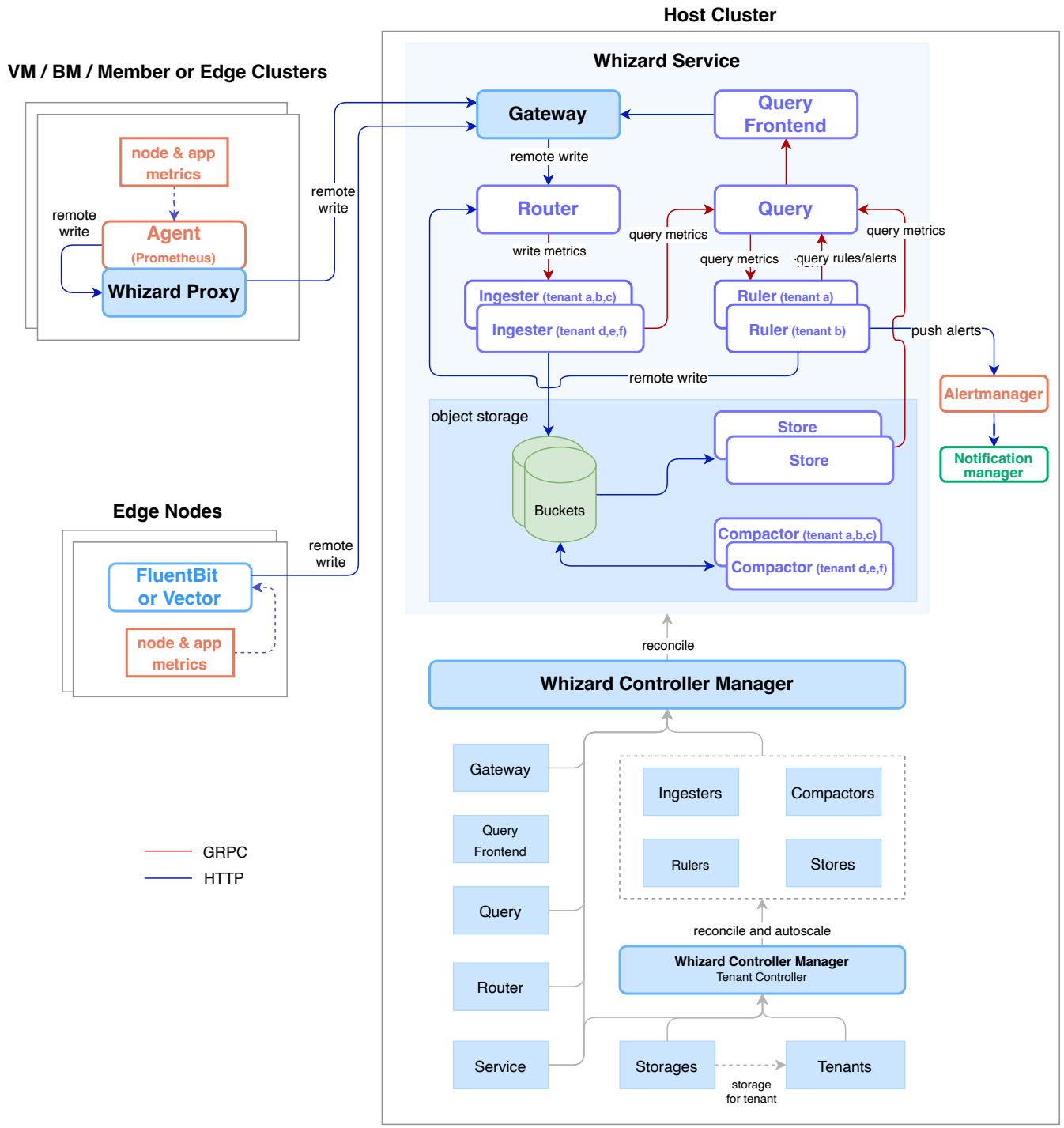
What if you've tens or even hundreds of clusters?

- Using a single ingester to receive all the data from all the clusters is not feasible:
  - The Ingester isn’t more powerful than a single instance of Prometheus
  - The Ingester has to be scalable to handle 10+ or 100+ clusters
- The Compactor, Store, Ruler has to be scalable to handle 10+ or 100+ clusters too
- Automatically configure Thanos whenever there is a cluster added or removed


KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Introducing Whizard

- All Thanos components are defined in CRD
- Introduced new Whizard CRDs:

  Service / Gateway / Storage / Tenant
- Introduce Tenant & Tenant-Based autoscaling
- Introduce RuleGroup-Based sharding for Ruler
- Introduce Time-Based Sharding for Store
- Gateway & Agent Proxy mechanism for tenant read/write control
- Query Optimization
- Security Enhancement:
  - Basic Auth to access Thanos WebUI
  - TLS config for all Thanos Components
- Service & Components 2-tiers config management



CRD definition of all components

Introduce CRDs to define all components:

- CRDs for Thanos:
  - **Router**
  - **Ingester**
  - **Ruler**
  - **Query / QueryFrontend**
  - **Store**
  - **Compactor**
- CRDs for Whizard:
  - **Service**
  - **Tenant**
  - **Gateway**
  - **Storage**

Service

Service:

- Default / Global settings for all components:
  - TenantHeader / DefaultTenantId / TenantLabelName
  - Gateway
  - Router
  - Ingester
  - Query / QueryFrontend
  - Ruler
  - Compactor
  - Store
  - Storage
  - RemoteWrites / RemoteQuery

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Service
metadata:
  name: whizard
  namespace: kubesphere-monitoring-system
spec:
  tenantHeader: cluster
  tenantLabelName: cluster
  defaultTenantId: unknown
  compactorTemplateSpec:
    ...
    defaultTenantsPerCompactor: 10
    retention:
      retention1h: 360d
      retention5m: 180d
      retentionRaw: 60d
  gatewayTemplateSpec:
    image: docker.io/kubesphere/whizard-monitoring-gateway:v0.10.0
  ingesterTemplateSpec:
    ...
    defaultIngesterRetentionPeriod: 3h
    defaultTenantsPerIngester: 3
    disableTsdbCleanup: true
  queryFrontendTemplateSpec:
    image: docker.io/thanosio/thanos:v0.36.1
  queryTemplateSpec:
    flags:
    - --query.max-concurrent=200
    image: docker.io/thanosio/thanos:v0.36.1
    replicaLabelNames:
    - prometheus_replica
    - receive_replica
    - ruler_replica
  routerTemplateSpec:
    image: docker.io/thanosio/thanos:v0.36.1
  rulerTemplateSpec:
```

Tenant

Tenant:

- Defines a tenant for Thanos
- Tenant.status indicates relevant resources for this tenant including:
  - Ingester
  - Ruler
  - Compactor

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Tenant
metadata:
  labels:
    monitoring.whizard.io/service: kubesphere-monitoring-system.whizard
    monitoring.whizard.io/storage: default
  name: host-cluster
spec:
  tenant: host-cluster
status:
  ingester:
    name: whizard-local-auto-2
    namespace: kubesphere-monitoring-system
  ruler:
    name: host-cluster
    namespace: kubesphere-monitoring-system
```

Gateway

Gateway:

- Gateway for all reads/writes requests
- Supports TLS config
- Supports Basic Auth config
- Supports exposing Thanos WebUI with basic auth
- Supports exposing Thanos WebUI with OAuth2-Proxy

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Gateway
metadata:
  name: whizard
  namespace: kubesphere-monitoring-system
spec:
  nodePort: 30990
  # Enable Thanos WebUI
  debug: true
  enabledTenantsAdmission: true
  webConfig:
    # Config TLS
    httpServerTLSConfig:
      certSecret:
        ...
      keySecret:
        ...
    # Config Basic Auth
    basicAuthUsers:
    - username:
        ...
      password:
        ...
```

Storage

Storage:

- Defines the Object Storage settings for Thanos :
  - **Bucket**
  - **Endpoint**
  - **AccessKey**
  - **SecretKey**

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Storage
metadata:
  name: remote
  namespace: kubesphere-monitoring-system
spec:
  S3:
    bucket: "whizard-monitoring"
    endpoint: "s3.pek3b.qingstor.com:443"
    accessKey:
      name: storage-secret
      key: accessKey
    secretKey:
      name: storage-secret
      key: secretKey
```

Router

Router:

- Defines the settings for Thanos Receive Router :
  - ReplicationFactor
  - Replicas

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Router
metadata:
  name: whizard
  namespace: kubesphere-monitoring-system
spec:
  replicationFactor: 2
  replicas: 3
  image: docker.io/thanosio/thanos:v0.36.1
```

Ingester

Ingester:

- Defines the settings for Thanos Receive Ingester :
  - DefaultTenantsPerIngester
  - Tenants
  - LocalTsdbRetention
  - Replicas
  - DefaultIngesterRetentionPeriod

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Ingester
metadata:
  name: whizard
  namespace: kubesphere-monitoring-system
spec:
  defaultTenantsPerIngester: 3
  tenants:
  - cluster1
  - cluster2
  - cluster3
  localTsdbRetention: 7d
  replicas: 2
  # The Ingester will be deleted if there is no tenants
  # for defaultIngesterRetentionPeriod
  defaultIngesterRetentionPeriod: 3h
```

Ruler

Ruler:

- Defines the settings for global or tenant Ruler :
  - **Tenant**: Ruler can be global for all tenants or for one specific tenant
  - **Shards**: Number of shards to evaluate rules. Each shard is one ruler StatefulSet
  - **EvaluationInterval**
  - **RulerQueryProxy**: Ruler query the gateway to fetch data to evaluate
  - **RulerWriteProxy**: Ruler write back the evaluated recording rules back to ingester with the tenant label

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Ruler
metadata:
  name: host-cluster
  namespace: kubesphere-monitoring-system
spec:
  tenant: host-cluster # Shouldn't be set for a global ruler
  shards: 1 # Can be >1 if there're too many rules to evaluate
  evaluationInterval: 1m
  alertmanagersUrl: <alert-manage-url>
  ruleSelectors:
  - matchLabels:
      role: alert-rules
  rulerQueryProxy:
    image: docker.io/kubesphere/whizard-monitoring-gateway:v0.10.0
  rulerWriteProxy:
    image: docker.io/kubesphere/cortex-tenant:v1.12.5
```

Compactor

Compactor:

- Defines the settings for global or tenant Ruler :
  - **DefaultTenantsPerCompactor**: number of tenants whose blocks can be compacted by this compactor
  - **Tenants**: specify all the tenants whose blocks will be compacted by this compactor
  - **Retention**: how long to keep the blocks in S3
    - Retention1h
    - Retention5m
    - RetentionRaw
  - **DisableDownsampling**: whether to enable downsampling

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Compactor
metadata:
  name: whizard-compactor-1
  namespace: kubesphere-monitoring-system
spec:
  # One compactor can compact blocks of up to 10 tenants
  defaultTenantsPerCompactor: 10
  tenants:
  - cluster1
  - cluster2
  - cluster3
  retention:
    retention1h: 360d
    retention5m: 180d
    retentionRaw: 60d
  disableDownsampling: false
```

Store

Store:

- Defines the settings for Store:
  - **TimeRanges**: define the time range for each store shard

    Below example will create 2 store shards (StatefulSets):
    - One for data < now - 30d
    - Another for now - 30d < data < now - 36h

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Store
metadata:
  name: whizard
spec:
  timeRanges:
    # minTime 为空或未配置时，对应的 store 工作负载将使用默认值 0000-01-01T00:00:00Z
    - minTime: ''
      maxTime: -30d
    - minTime: -30d
      # maxTime 为空或未配置时，对应的 store 工作负载将使用默认值 9999-12-31T23:59:59Z
      maxTime: -36h
```

Query

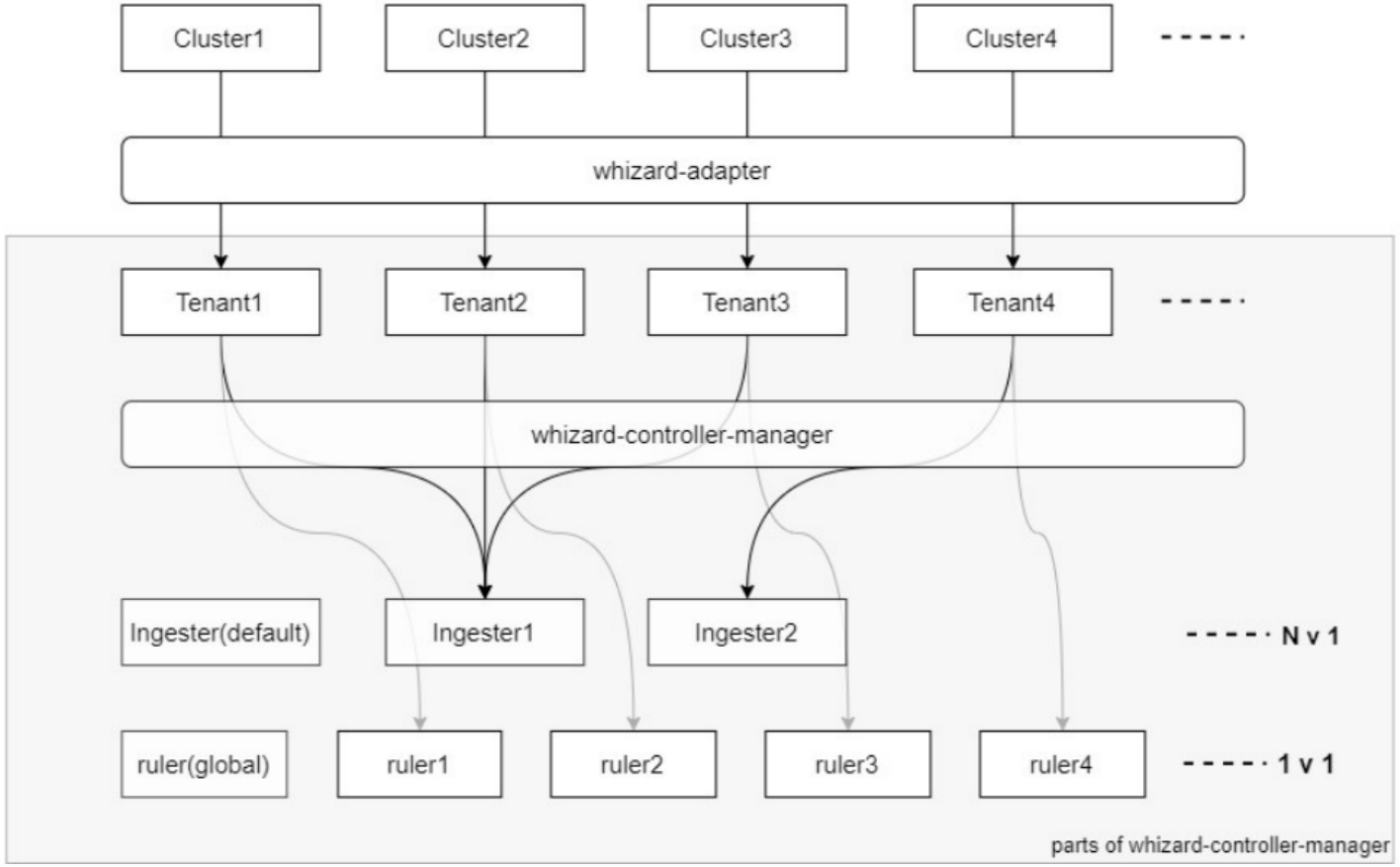
Query:

- Defines the settings for Query:
  - **PromqlEngine**: which promql engine to use, default to the fastest thanos engine
  - **Stores**: defines all the external query sources
- Query Ingester for data < 36h
- Query Store for data >= 36h

```yaml
apiVersion: monitoring.whizard.io/v1alpha1
kind: Query
metadata:
  name: whizard
  namespace: kubesphere-monitoring-system
spec:
  promqlEngine: thanos
  flags:
  - --query.max-concurrent=200
  image: docker.io/thanosio/thanos:v0.36.1
  replicaLabelNames:
  - prometheus_replica
  - receive_replica
  - ruler_replica
  stores:
  - addresses:
    - <extra-external-data-source-1>
    - <extra-external-data-source-2>
```

Tenant-Based components auto-scaling

- Each Ingester can hold data for 3 tenants by default
- Each Compactor can compact blocks for 10 tenants by default
- There is one dedicated ruler for each tenant to evaluate recording rules for that tenant

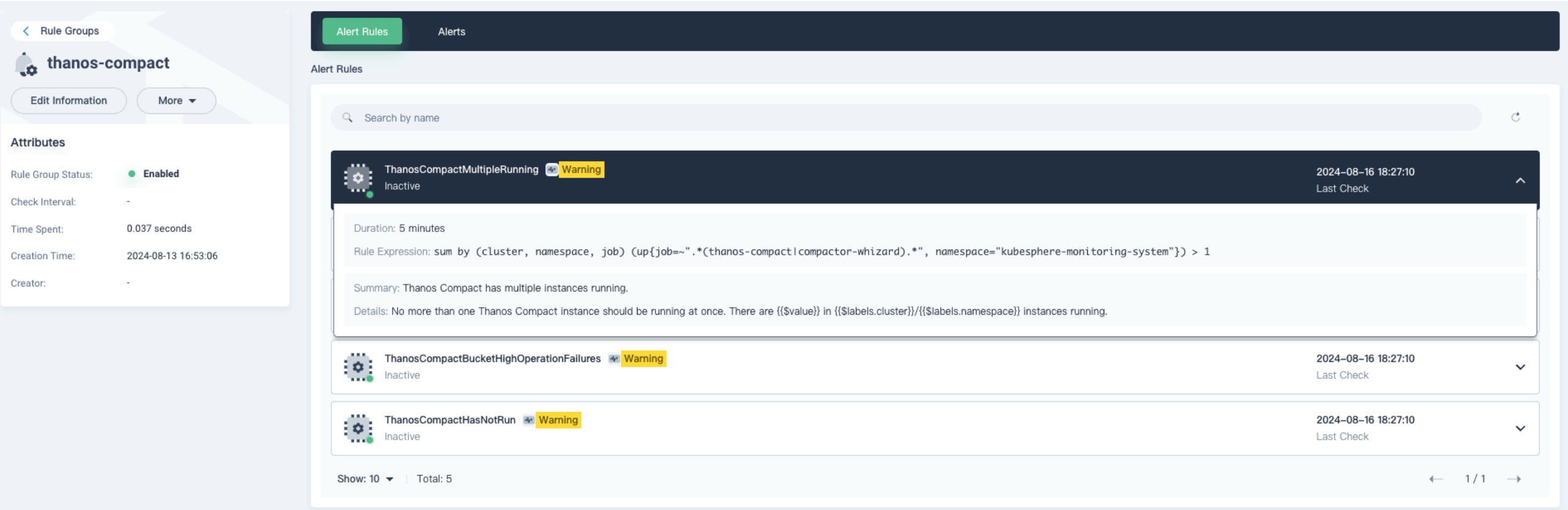


Use RuleGroup instead of PrometheuesRule

- **PrometheusRule** contains rules from multiple rule groups
- **RuleGroup** only contains rules from one rule group:
  - Easier to manage and visualization
  - Easier to edit concurrently
  - Avoid creating too many CRs:
    - Up to 40 rules in one RuleGroup
  - Keep the rule evaluation concurrency:
    - Each RuleGroup is evaluated by one single goroutine
- **RuleGroup** for namespace rules
- **ClusterRuleGroup** for cluster rules
- **GlobalRuleGroup** for global, multi-cluster/tenant rules

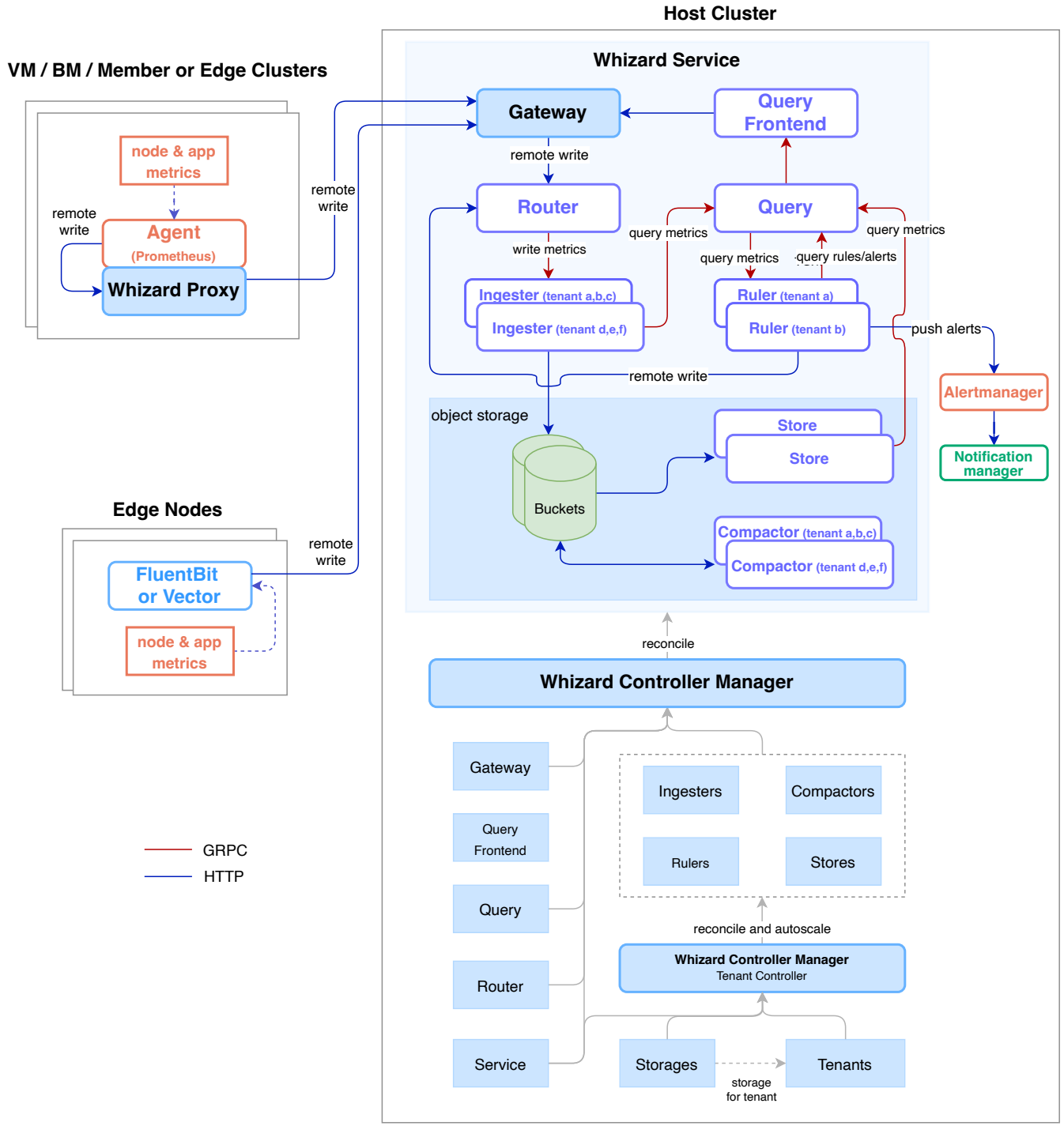
```yaml
apiVersion: alerting.kubesphere.io/v2beta1
kind: GlobalRuleGroup
metadata:
  name: thanos-compact
spec:
  rules:
  - alert: ThanosCompactMultipleRunning
    annotations:
      description: No more than one Thanos Compact instance sh
        There are {{$value}} in {{$labels.cluster}}/{{$labels.
        running.
      runbook_url: https://alert-runbooks.kubesphere.io/latest
      summary: Thanos Compact has multiple instances running.
    expr: sum by (cluster, namespace, job) (up{job=~".*(thanos
      namespace="kubesphere-monitoring-system"}) > 1
    for: 5m
    id: b098b14c1f9c7179038289b21fa32a1b
    severity: warning
  - alert: ThanosCompactHalted
    annotations:
      description: Thanos Compact {{$labels.job}} in {{$labels
        has failed to run and now is halted.
      runbook_url: https://alert-runbooks.kubesphere.io/latest
      summary: Thanos Compact has failed to run and is now hal
    expr: thanos_compact_halted{job=~".*(thanos-compact|compac
      == 1
    for: 5m
    id: bdd26ccadf2423e163cf83e419b52f5d
    severity: warning
```

Use RuleGroup instead of PrometheuesRule



- All Thanos components are defined in CRD
- Introduced new Whizard CRDs:

  Service / Gateway / Storage / Tenant
- Introduce Tenant & Tenant-Based autoscaling
- Introduce RuleGroup-Based sharding for Ruler
- Introduce Time-Based Sharding for Store
- Gateway & Agent Proxy mechanism for tenant read/write control
- Query Optimization
- Security Enhancement:
- Service & Components 2-tiers config management



What’re we still expecting from Thanos

- Better multi-tenant support and improved scalability of Ruler:
  - Issue: Implement multi-tenant Ruler: multitsdb and multiagent
  - Proposal: Enable Receiver to extract Tenant from a label present in incoming timeseries
  - PRs: receive/handler: implement tenant label splitting

    More PRs to come and more tests are needed
- Support adding external labels to tenants:
  - Issues: Receive: Allow specifying tenant-specific external labels
  - Proposal: Allow statically specifying tenant-specific external labels in Receivers
  - What’s still missing: dynamic external label , Specify different external labels for every tenant
- Extract router and ingestor as separate components:
  - Issue: Receiver: Logically split router and ingestor mode

    https://github.com/thanos-io/thanos/pull/5623#issuecomment-1221855169


KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
The production adoption of Whizard in KubeSphere

Whizard 可观测中心

- 资源监控
  - 多集群监控
  - 资源统计排行
- 全局告警
- 资源查询

收起

概览

收起集群列表 | 选择集群

14 集群	52 节点	116 项目	330 容器组	146 部署
76 有状态副本集	23 守护进程集	17 任务	16 定时任务	142 持久卷声明
157 服务	43 应用路由	49 集群成员	11 集群角色	

CPU 用量	内存用量	容器组用量	磁盘用量
37%	91%	34%	56%
68.2 核	2012 Gi	660	1357.47 GB

全局告警

103 告警

91 已触发

12 验证中

紧急	重要	警告	提醒
26	43	13	28

2023-08-22 09:46:28	紧急	节点 node2, node3 可用内存 > 1GB	验证中	host
2023-08-22 09:43:51	提醒	节点 node2, node3 可用内存 > 1GB	验证中	cluser1

集群列表

- acmecorp 主集群
- fakeco 成员集群
- alphawidgets 成员集群
- betaIndustries 成员集群
- omegasolutions 成员集群
- pseudoLabs 成员集群
- nebulasystems 成员集群
- galactictech 成员集群
- zenithcorp 成员集群
- quantumcomputing 成员集群

Whizard 可观测中心

资源监控
多集群监控
资源统计排行
全局告警
内置告警
自定义告警
内置规则组
自定义规则组
资源查询
容器组

收起

资源统计排行

资源统计排行提供集群资源用量排行情况。

资源统计排行

选择集群

集群 | 节点 | 项目 | 容器组

按 CPU 用量 | 降序排序

集群	CPU 用量	CPU 使用率	内存用量	内存使用率	磁盘用量	磁盘使用率	容器组数量	容器组使用率
opela 成员集群	4.23 Core	90%	18.45 Gi	96%	85.37 Gi	95%	282	76%
aerified 成员集群	2.49 Core	80%	17.22 Gi	89%	79.14 Gi	78%	238	66%
opela 成员集群	2.11 Core	77%	14.12 Gi	82%	76.23 Gi	68%	218	64%
tres-zap 成员集群	2.06 Core	76%	11.29 Gi	75%	60.11 Gi	51%	198	58%
zoolab 成员集群	1.92 Core	67%	10.03 Gi	68%	53.23 Gi	42%	181	53%
zamit 成员集群	1.88 Core	55%	8.56 Gi	59%	42.19 Gi	38%	171	42%
sonsing 成员集群	1.54 Core	50%	8.12 Gi	48%	25.21 Gi	22%	170	38%
tresom 成员集群	1.08 Core	44%	5.39 Gi	37%	18.45 Gi	17%	157	31%

Whizard 可观测中心

资源监控

全局告警

内置告警

自定义告警

内置规则组

自定义规则组

资源查询

收起

内置告警

当资源指标满足规则组中配置的条件时，系统将生成告警。

搜索

消息	状态	告警级别	规则名称	规则组	监控目标	集群	触发时间
Pod container waiting longer than 1 hour pod/test-d84fb4764-lfhzf in namespace xxxx2 on container container-6nso0e has been in waiting state for longer than 1 hour.	验证中	警告	KubeContainerWaiting	kubernetes-apps	-	qs-host	2024-02-05 18:48:16
Pod container waiting longer than 1 hour Deployment xxxx2/test has not matched the expected number of replicas for longer than 15 minutes.	已触发	警告	KubeDeploymentReplicasMismatch	kubernetes-apps	-	qs-host	2024-02-05 07:21:16
Info-level alert inhibition. Deployment xxxx2/test has not matched the expected number of replicas for longer than 15 minutes.	已触发	提醒	InfoInhibitor	general-rules	-	qs-host	2024-02-04 11:59:33
Processes experience elevated CPU throttling. 40% throttling of CPU in namespace kubesphere-logging-system for container kube-auditing-webhook in pod kube-auditing-webhook-deploy-5c4b98cbcd-bvn22.	已触发	紧急	CPUThrottlingHigh	kubernetes-resources	-	qs-host	2024-02-04 11:40:43

Whizard 可观测中心
资源监控
全局告警
资源查询
容器组
收起

容器组

容器组（Pod）是 Kubernetes 应用程序的基本执行单元，是您创建或部署的 Kubernetes 对象模型中最小和最简单的单元。

全部集群 搜索

名称	状态	节点	容器组 IP 地址	项目	集群	更新时间
prometheus-k8s-0 运行中	运行中	node2（172.18.0.55）	10.233.96.198	kubesphere-monitoring-system	qs-host	2024-02-06 11:30:00
openpitrix-import-job-tc28x 运行中	运行中	node1（172.18.0.54）	10.233.90.102	kubesphere-controls-system	qs-host	2024-02-06 11:00:00
notification-manager-operator-65f9... 运行中	运行中	node1（172.18.0.54）	10.233.90.26	kubesphere-monitoring-system	qs-host	2024-02-06 10:30:00
kubectl-test7385136450-6df8869cb... 运行中	运行中	kse-member（172.18.0.57...	10.233.82.234	kubesphere-monitoring-system	qs-member	2024-02-04 11:37:50
notification-manager-deployment-7... 运行中	运行中	kse-member（172.18.0.57...	10.233.92.175	kubesphere-monitoring-system	qs-member	2024-02-04 11:24:26
query-whizard-5ff8fc566d-2jkmq 运行中	运行中	node1（172.18.0.54）	10.233.90.105	kubesphere-monitoring-system	qs-host	2024-01-31 14:42:36
whizard-agent-proxy-5d5f7d64dc-b... 运行中	运行中	kse-member（172.18.0.57...	10.233.82.45	kubesphere-monitoring-system	qs-member	2024-01-29 17:03:47
query-whizard-5ff8fc566d-f26j6 运行中	运行中	node1（172.18.0.54）	10.233.90.237	kubesphere-monitoring-system	qs-host	2024-01-29 17:03:33
compactor-whizard-remote-m44q5-0 运行中	运行中	node1（172.18.0.54）	10.233.92.131	kubesphere-monitoring-system	qs-host	2024-01-29 10:54:07
ruler-sdqwfewfew-0-0 运行中	运行中	node3（172.18.0.56）	10.233.96.61	kubesphere-monitoring-system	qs-host	2024-01-29 10:54:06

Roadmap

KubeSphere Whizard Observability Center => **WhizardTelemetry Observability Platform**:

- More observability signals will be added: Logging, Tracing, Events, Auditing, Notification
- More observability features will be supported: Events/Logs alerting, cost management etc.
- OpenTelemetry Support
- Observability powered by eBPF
- The combination of OpenTelemetry and eBPF
- LLM applications observability
- AI Infra observability

KubeCon | CloudNativeCon China 2024

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

AI_dev Open Source GenAI & ML Summit

Whizard has now been open-sourced at KubeCon HK 2024!
https://github.com/WhizardTelemetry/whizard

Join the community by wechat: