



K8S 监控实践

Prometheus, 多租户与多集群

霍秉杰

KubeSphere Observability Team



Prometheus



Prometheus 简介

- 受 Google Borgmon 启发由前 Google 工程师开发
- 2016 年继 K8S 之后第二个加入 CNCF 基金会
- 2018 年继 K8S 之后第二个从 CNCF 毕业
- 与 K8S 天然集成
- 云原生监控领域事实上的标准
- More than Cloud Native
- OpenMetrics

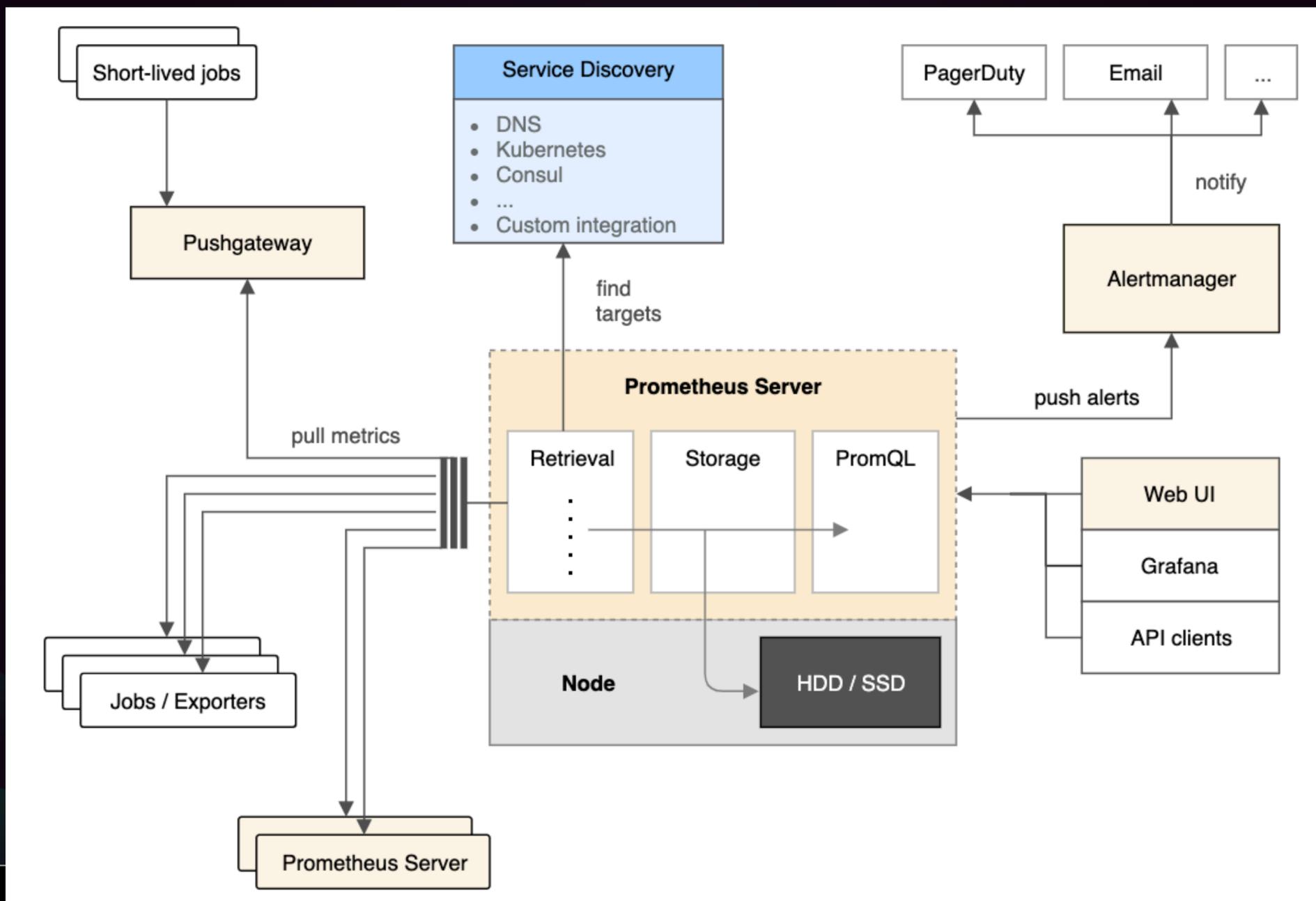


Prometheus 特性

- 多维数据模型
- 灵活强大的查询语言 PromQL
- Pull 模式收集时序数据
- 2.0 重写了底层时序存储引擎
- 聚焦核心: 只开发单节点的 Prometheus
- 依赖生态: exporters/remote_read/remote_write
- 持久化存储: TSDBs/Thanos 等



Prometheus 架构





时序模型 - time series

time-series 是按照时间戳和值的序列顺序存放，称之为向量(vector)
每条time-series通过指标名称(metrics name)和一组标签集(labelset)命名





时序模型 - sample

time-series中的每一个点称为一个 sample，sample 由以下三部分组成：

1. 指标(metric)：metric name 和描述当前样本特征的 labelsets
2. 时间戳(timestamp)：一个精确到毫秒的时间戳
3. 样本值(value)：一个 float64 的浮点型数据表示当前样本的值

```
<----- metric -----><-timestamp -><-value->
http_request_total{status="200", method="GET"}@1434417560938 => 94355
http_request_total{status="200", method="GET"}@1434417561287 => 94334

http_request_total{status="404", method="GET"}@1434417560938 => 38473
http_request_total{status="404", method="GET"}@1434417561287 => 38544

http_request_total{status="200", method="POST"}@1434417560938 => 4748
http_request_total{status="200", method="POST"}@1434417561287 => 4785
```



时序模型 - metric

```
api_http_requests_total{method="POST", handler="/messages"}
```

等同

```
{__name__="api_http_requests_total", method="POST", handler="/messages"}
```

4 种类型的 metrics:

Counter: 只增不减

Gauge: 可增可减

Histogram和Summary: 主用用于统计和分析样本的分布情况



PromQL

Counter 类型:

//HTTP请求量的增长率

```
rate(http_requests_total[5m])
```

```
irate(http_requests_total[5m])
```

//访问量前10的HTTP地址

```
topk(10, http_requests_total)
```



PromQL

Gauge 类型:

// 直接查看系统的当前状态

```
node_memory_MemFree
```

//CPU温度在两个小时内的差异

```
delta(cpu_temp_celsius{host="zeus"}[2h])
```

//预测系统磁盘空间在4个小时之后的剩余情况

```
predict_linear(node_filesystem_free{job="node"}[1h], 4 * 3600)
```



Performance

“Local storage is enough for many orgs”

—Julius Volz , Co-Founder of prometheus

1 million+ samples/s

millions of time series

1~2 bytes per sample



Prometheus 配置

命令行配置参数：存储位置、数据保存时间等（需重启 Prometheus）

配置文件配置参数：global configs, remote read/write, scrape_configs, recording/alert rules.（需reload 配置文件）

如此复杂的配置如何管理？

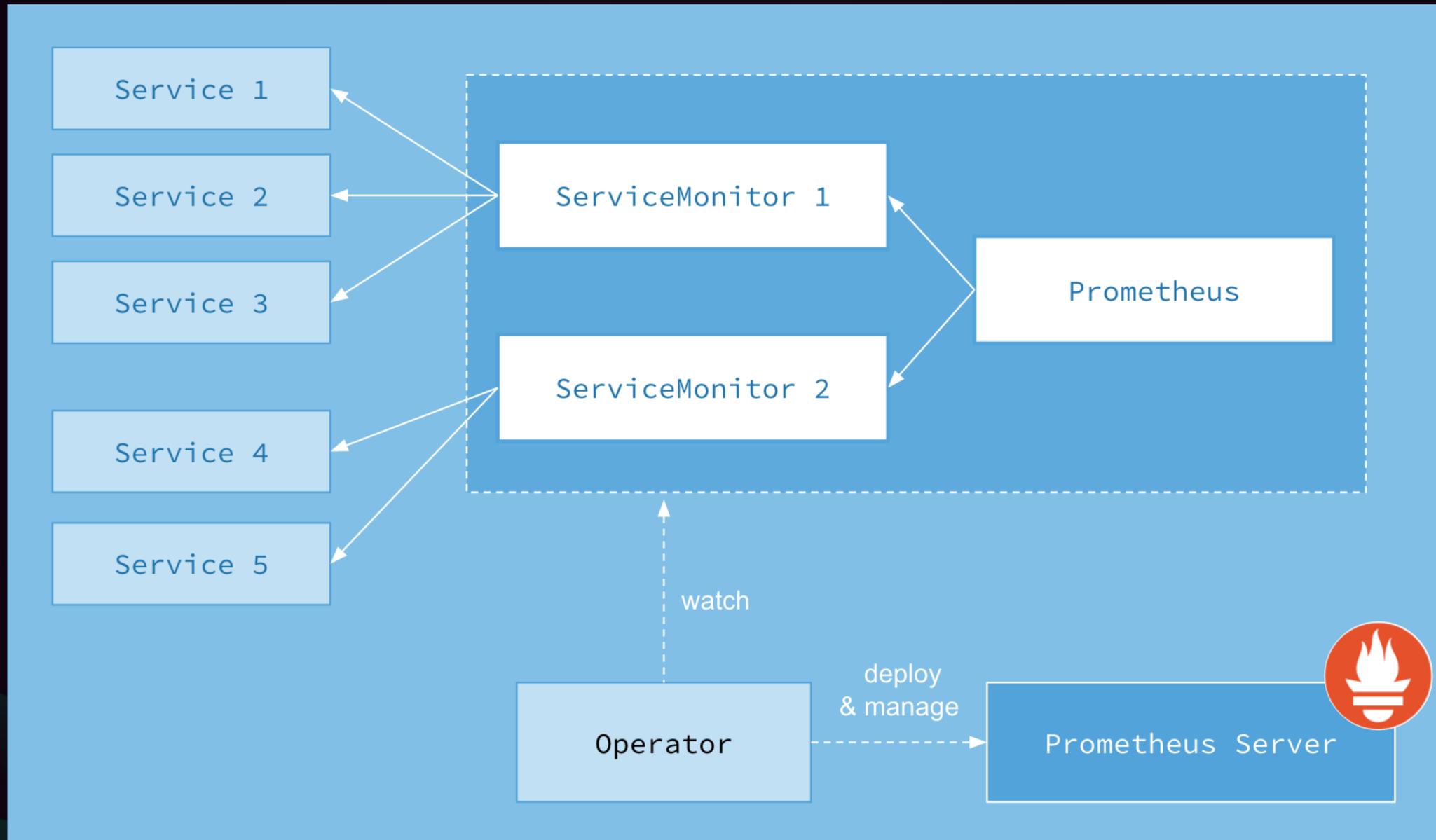


Prometheus Operator

- Operator 模式: CustomResourceDefinition (CRD) + Custom Controller
- 自定义 Custom Resource Definitions (CRD) 管理 Prometheus:
 - ServiceMonitor: 负责管理监控目标配置;
 - Prometheus: 负责创建和管理 Prometheus Server 实例;
 - PrometheusRules: 记录 Recording Rules
- 如何工作?
 - 在 K8s 集群上部署 Operator;
 - Operator 根据 CRD 配置自动创建出相关工作负载



Prometheus Operator



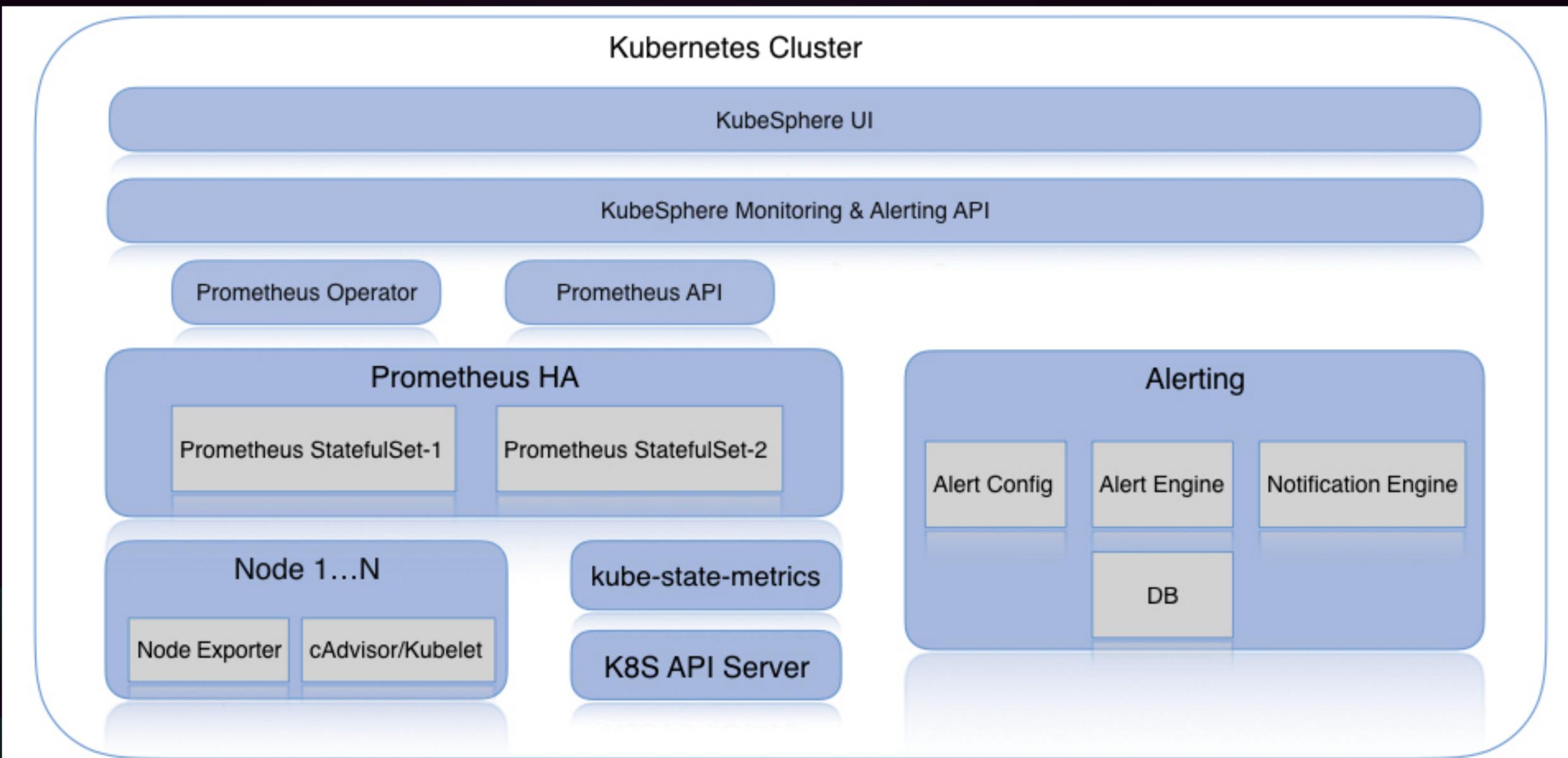


Prometheus Operator

- 方便：更改 Prometheus 配置后，无须手动重启 Prometheus 或手动 reload 配置
- K8S 原生的管理方式：将复杂的配置按种类与用途抽象为 3 个不同的 crd
kubectl edit prometheus k8s
- K8S 原生 API 的支持



Prometheus + K8S





Kubesphere 对 Prometheus 的优化

KubeSphere 1.0.x

KubeSphere 2.0.0

每 1000 pods

内存消耗

1 ~ 1.2 GB

238MB



回报社区 - Our Contributions

- 为 Prometheus operator 增加最大并发数等 query option , 并成为 Prometheus Operator 0.27.0 版本的 feature 之一
<https://github.com/coreos/prometheus-operator/pull/2194>
- 更新 Prometheus Operator 支持的 Prometheus 版本
<https://github.com/coreos/prometheus-operator/pull/2664>
- 为 Prometheus operator 增加 inode rule
<https://github.com/kubernetes-monitoring/kubernetes-mixin/pull/126>
- 为 Prometheus crd 添加 toleration
<https://github.com/coreos/kube-prometheus/pull/102>
- 为 Prometheus Operator 更新 Thanos v0.5.0 支持
<https://github.com/coreos/kube-prometheus/pull/140>



多租户 Multi-Tenant



Kubesphere 多租户监控

- 原生 Prometheus HTTP API

GET /api/v1/query?query=http_requests_total{method="GET"}

- 多租户管理：认证、鉴权、代理

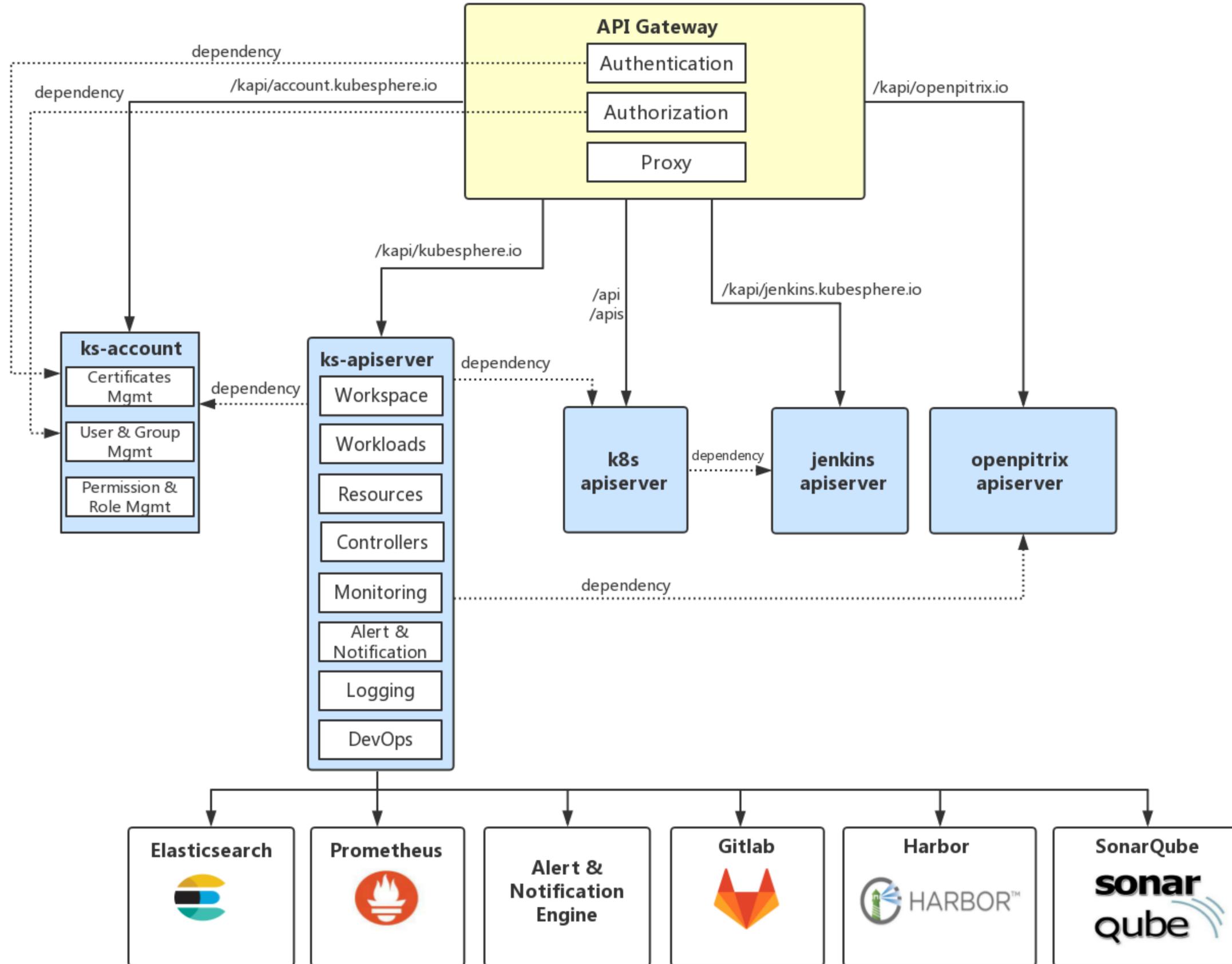
<https://docs.kubesphere.io/advanced-v2.0/zh-CN/multi-tenant/intro/>

- APIs in Kubesphere

GET /monitoring.kubesphere.io/cluster

GET /monitoring.kubesphere.io/namespaces/{namespace}

GET /monitoring.kubesphere.io/namespaces/{namespace}/pods





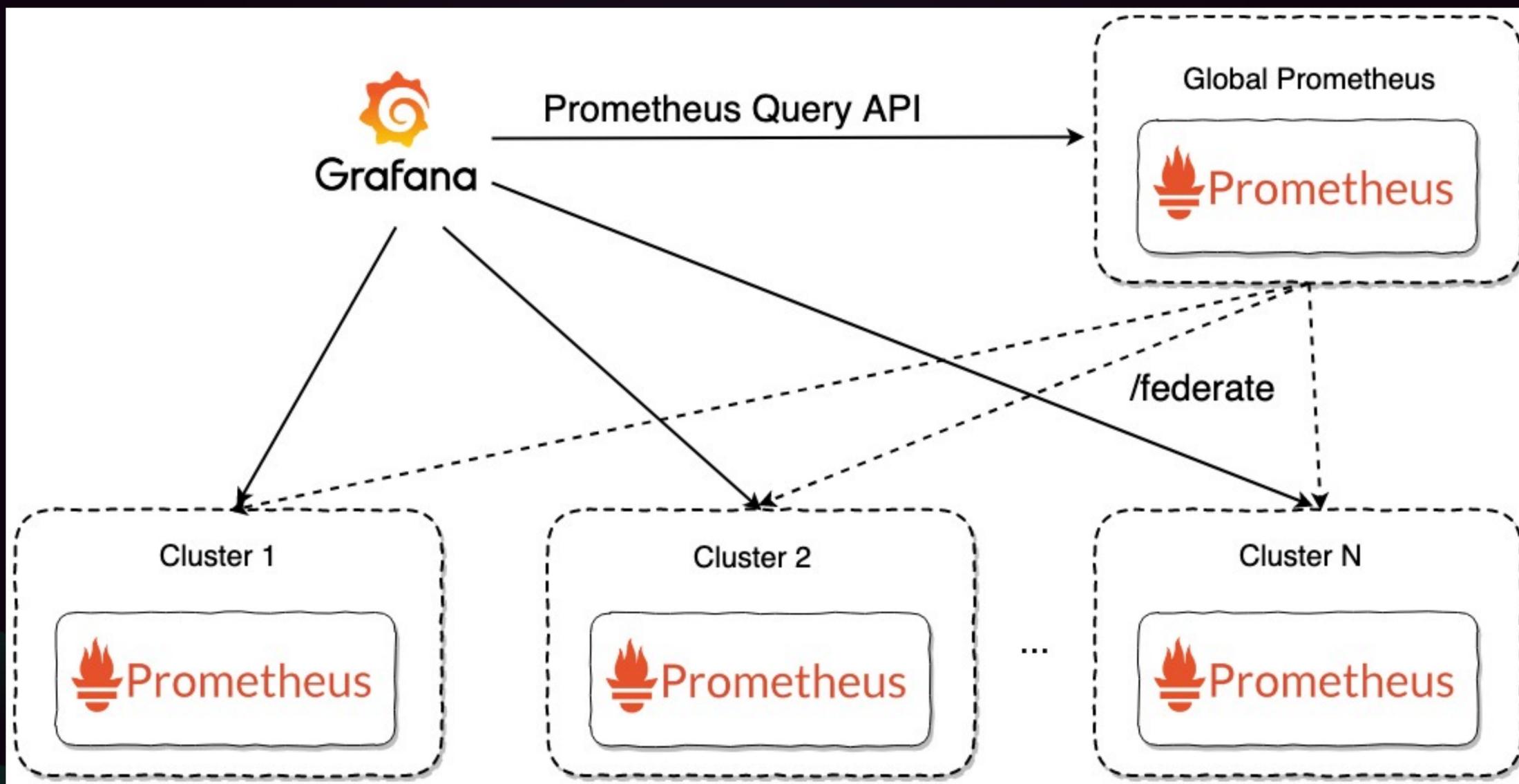
Demo



多集群 Multi-Cluster

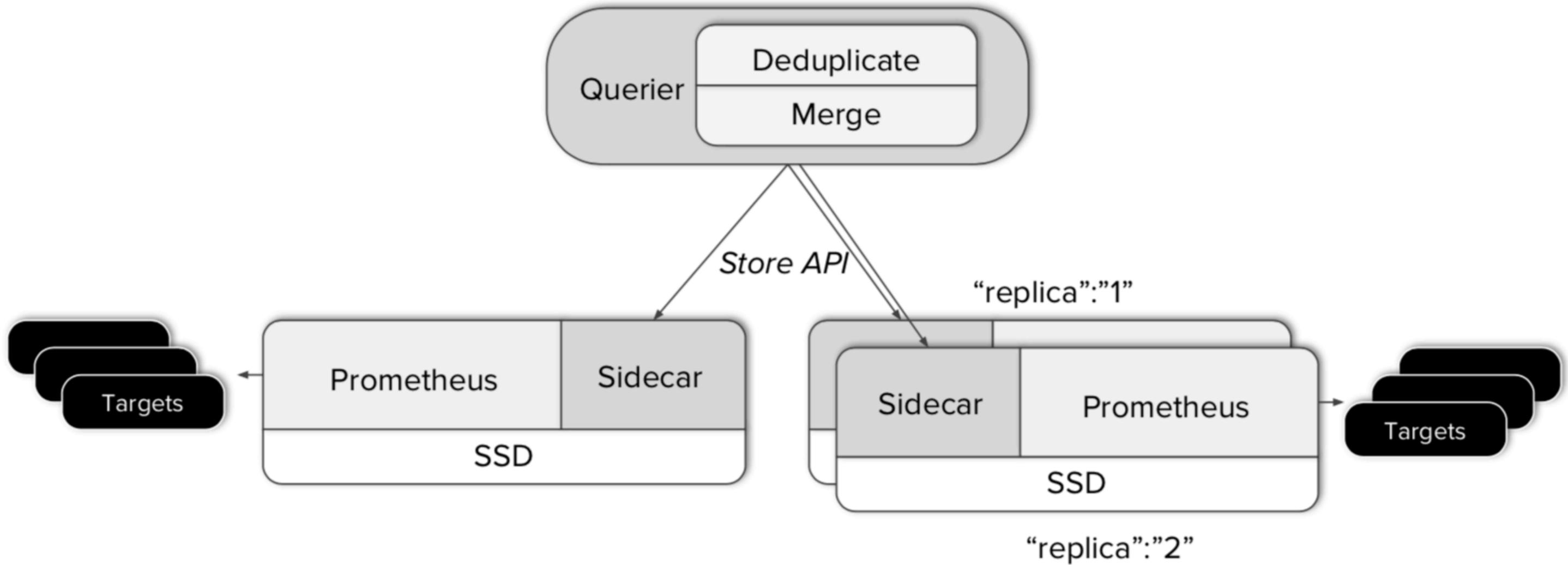


Prometheus Federation

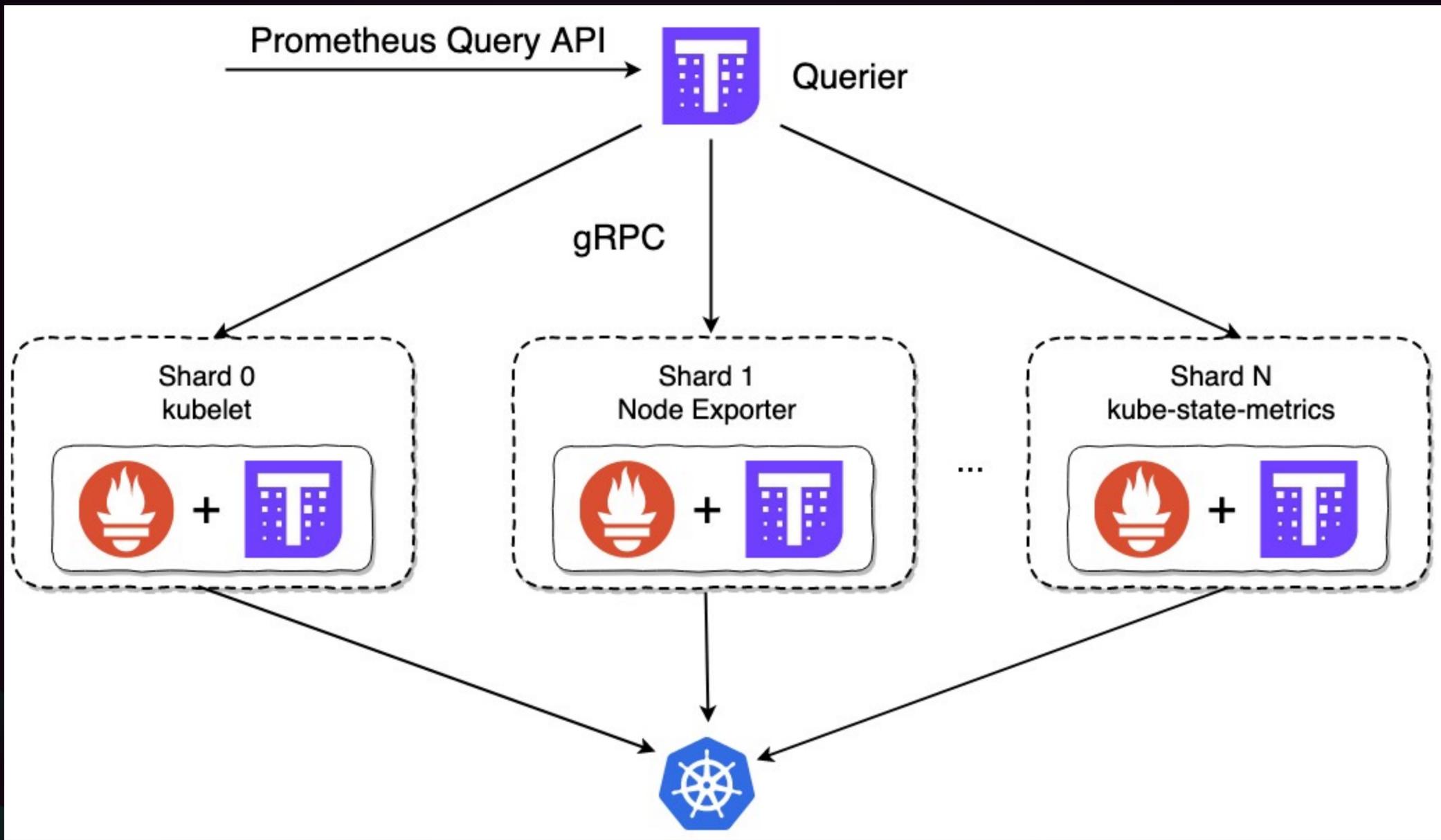




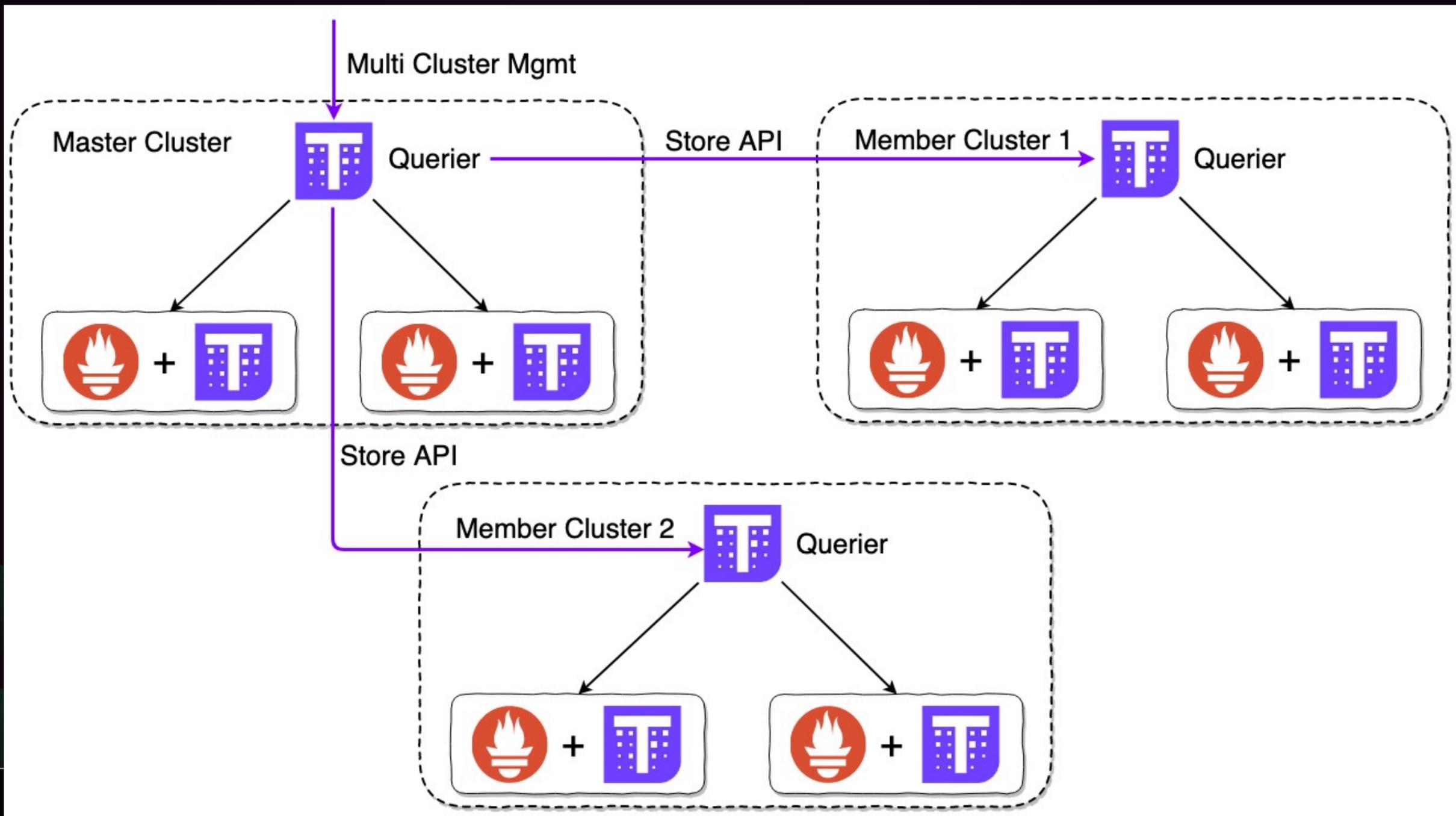
Thanos



 +  Global Query



 +  Federation Query





+



Demo



We're open sourcing

- <https://kubesphere.io/>
- <https://github.com/kubesphere/kubesphere>
- <https://github.com/kubesphere/custom-metrics-operator>
- <https://kubesphere.io/docs/advanced-v2.0/zh-CN/api-reference/monitoring-metrics/> kubesphere 监控指标文档

@benjaminhuo