QINGCLOUD | KUBESPHERE

Prometheus 在 K8S 上的监控实践

霍秉杰

KubeSphere Observability Team

QINGCLOUD | KUBESPHERE

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

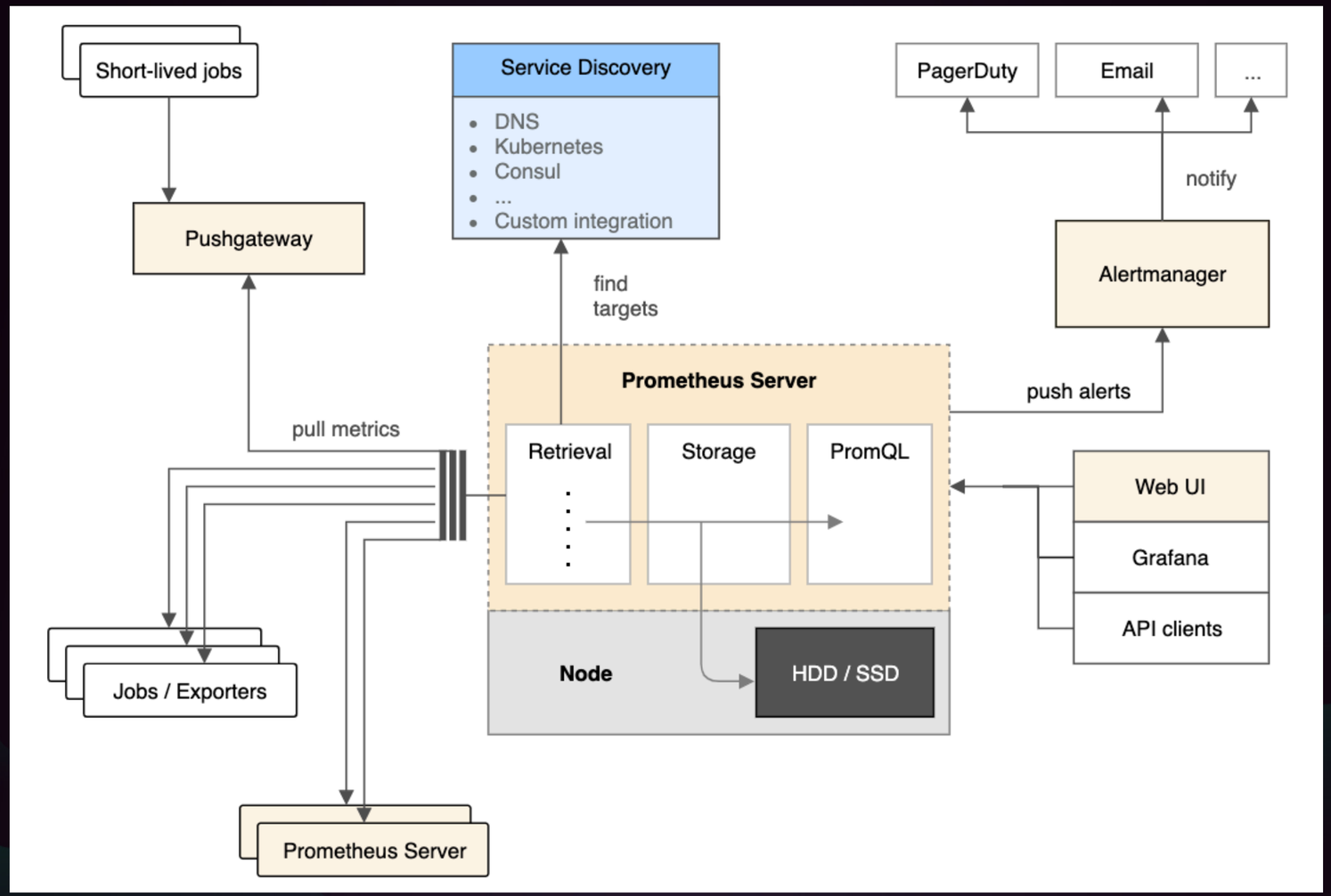
QINGCLOUD | KUBESPHERE

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：Cortex/InfluxDB/Thanos 等

QINGCLOUD | KUBESPHERE

Prometheus 架构



QINGCLOUD | KUBESPHERE

Prometheus 时序模型

```
# HELP node_cpu Seconds the cpus spent in each mode.
# TYPE node_cpu counter
node_cpu_seconds_total{cpu="0", mode="guest"} 0
node_cpu_seconds_total{cpu="0", mode="idle"} 2.03442237e+06
node_cpu_seconds_total{cpu="0", mode="iowait"} 3522.37
node_cpu_seconds_total{cpu="0", mode="irq"} 0.48
node_cpu_seconds_total{cpu="0", mode="nice"} 515.56
node_cpu_seconds_total{cpu="0", mode="softirq"} 953.06
node_cpu_seconds_total{cpu="0", mode="steal"} 0
node_cpu_seconds_total{cpu="0", mode="system"} 6605.46
node_cpu_seconds_total{cpu="0", mode="user"} 23343.01
node_cpu_seconds_total{cpu="1", mode="guest"} 0
node_cpu_seconds_total{cpu="1", mode="idle"} 2.03471439e+06
node_cpu_seconds_total{cpu="1", mode="iowait"} 3633.5
node_cpu_seconds_total{cpu="1", mode="irq"} 0.58
```

QINGCLOUD | KUBESPHERE

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

QINGCLOUD | KUBESPHERE

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
```

QINGCLOUD | KUBESPHERE

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

QINGCLOUD | KUBESPHERE

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)
```

QINGCLOUD | KUBESPHERE

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)
```

QINGCLOUD | KUBESPHERE

PromQL

Summary 类型：

```
# HELP prometheus_tsdb_wal_fsync_duration_seconds Duration of WAL fsync.
# TYPE prometheus_tsdb_wal_fsync_duration_seconds summary
prometheus_tsdb_wal_fsync_duration_seconds{quantile="0.5"} 0.012352463
prometheus_tsdb_wal_fsync_duration_seconds{quantile="0.9"} 0.014458005
prometheus_tsdb_wal_fsync_duration_seconds{quantile="0.99"} 0.017316173
prometheus_tsdb_wal_fsync_duration_seconds_sum 2.888716127000002
prometheus_tsdb_wal_fsync_duration_seconds_count 216
```

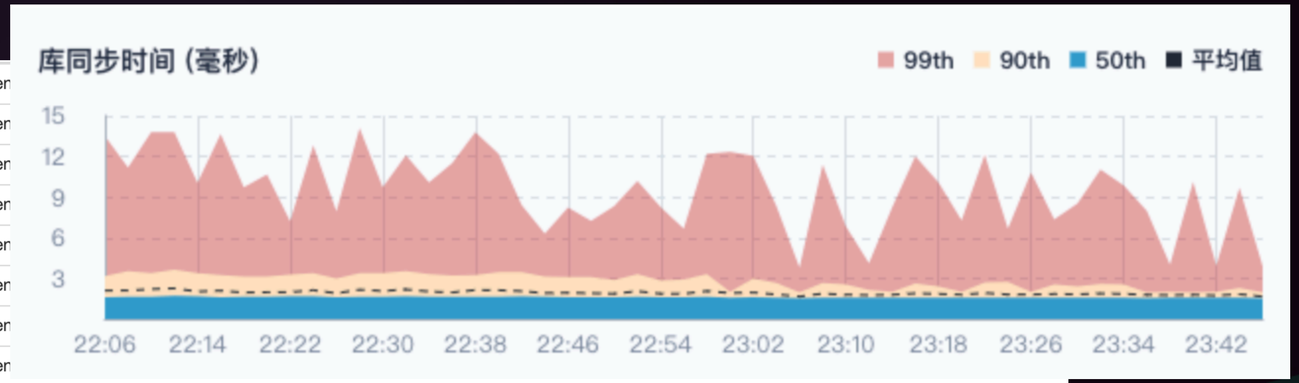
QINGCLOUD | KUBESPHERE

PromQL

Histogram 类型：

// prometheus chunk 首次压缩的时间

```
histogram_quantile(0.9, sum(irate(etcd_disk_backend_commit_duration_seconds_bucket{job="etcd"}[5m])) by (instance, le))
```



etcd_disk_backend_commit_duration_seconds_bucket{endpoint="metrics",instance="192.168.0.8:2379",job="etcd",le="0.128",namespace="kube-system",service="etcd"}	26590073
etcd_disk_backend_commit_duration_seconds_bucket{endpoint="metrics",instance="192.168.0.8:2379",job="etcd",le="0.256",namespace="kube-system",service="etcd"}	26590125
etcd_disk_backend_commit_duration_seconds_bucket{endpoint="metrics",instance="192.168.0.8:2379",job="etcd",le="0.512",namespace="kube-system",service="etcd"}	26590204
etcd_disk_backend_commit_duration_seconds_bucket{endpoint="metrics",instance="192.168.0.8:2379",job="etcd",le="1.024",namespace="kube-system",service="etcd"}	26590276
etcd_disk_backend_commit_duration_seconds_bucket{endpoint="metrics",instance="192.168.0.8:2379",job="etcd",le="2.048",namespace="kube-system",service="etcd"}	26590287
etcd_disk_backend_commit_duration_seconds_bucket{endpoint="metrics",instance="192.168.0.8:2379",job="etcd",le="4.096",namespace="kube-system",service="etcd"}	26590287
etcd_disk_backend_commit_duration_seconds_bucket{endpoint="metrics",instance="192.168.0.8:2379",job="etcd",le="8.192",namespace="kube-system",service="etcd"}	26590287

QINGCLOUD | KUBESPHERE

Recording rules

```
- alert: KubePodCrashLooping
  annotations:
    message: Pod {{ $labels.namespace }}/{{ $labels.pod }} ({{ $labels.container
      }}) is restarting {{ printf "%.2f" $value }} times / 5 minutes.
    runbook_url: https://github.com/kubernetes-monitoring/kubernetes-mixin/tree/master/runbook.md#alert-name-kubepodcrashl
  expr: |
    rate(kube_pod_container_status_restarts_total{job="kube-state-metrics"}[15m]) * 60 * 5 > 0
  for: 1h
  labels:
    severity: critical
```

QINGCLOUD | KUBESPHERE

Prometheus 配置

```
global:
  # How fr
  [ scrape
  # How lo
  [ scrape
  # How fr
  [ evalua
  # The la
  # exterr
  external
    [ <lab
```

```
# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

```
mmunicating with
ager).
enabled). This will also reload any configured rule files.
```

QINGCLOUD | KUBESPHERE

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?

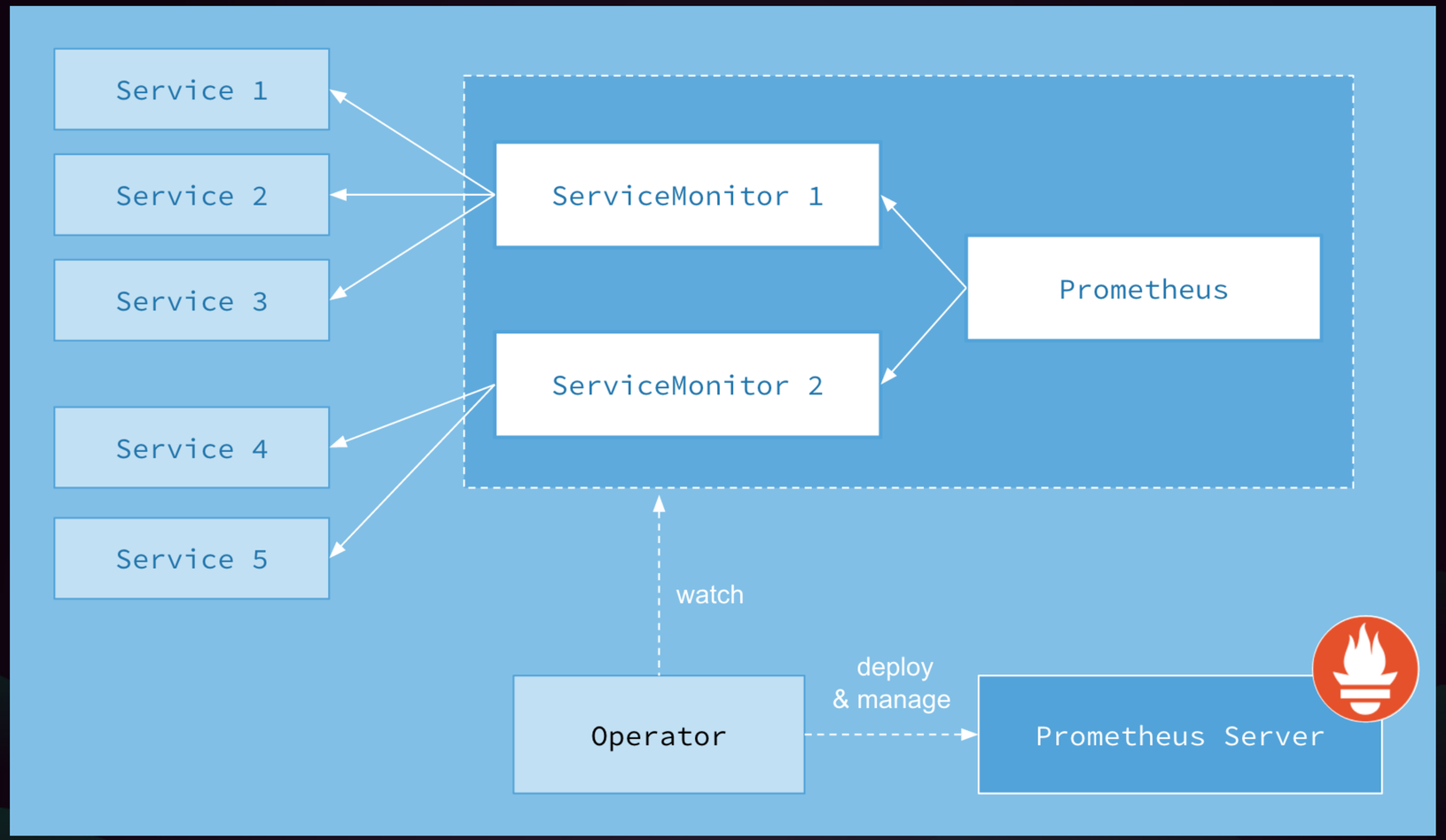
QINGCLOUD | KUBESPHERE

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 Prometheus Pods

QINGCLOUD | KUBESPHERE

Prometheus 架构



QINGCLOUD | KUBESPHERE

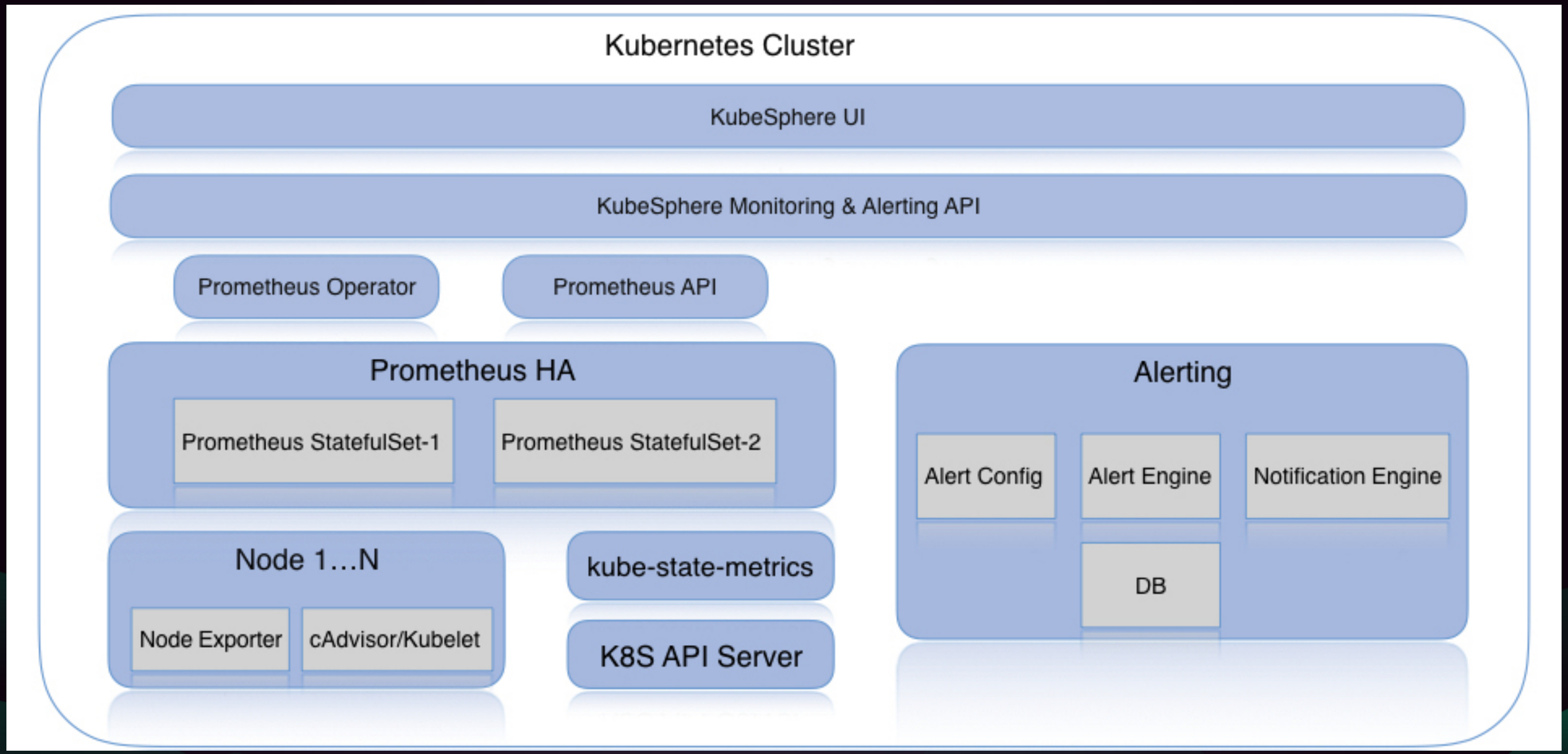
Prometheus Operator 优势

```
root@i-nsoydb08:~# kubectl get crd | grep .*monitoring.coreos.com
alertmanagers.monitoring.coreos.com                 2019-03-20T05:50:09Z
prometheuses.monitoring.coreos.com                  2019-03-20T05:50:09Z
prometheusrules.monitoring.coreos.com               2019-03-20T05:50:09Z
servicemonitors.monitoring.coreos.com               2019-03-20T05:50:09Z
root@i-nsoydb08:~#
root@i-nsoydb08:~# kubectl get servicemonitors.monitoring.coreos.com -n kubesphere-monitoring-system
NAME                      AGE
coredns                   86d
etcd                      86d
kube-apiserver            86d
kube-controller-manager   86d
kube-scheduler            86d
kube-state-metrics        86d
kubelet                   86d
node-exporter             86d
prometheus                86d
prometheus-system         86d
root@i-nsoydb08:~#
```

置

QINGCLOUD | KUBESPHERE

Prometheus + K8S



QINGCLOUD | KUBESPHERE

Our Contributions

- 为 Prometheus operator 增加 inode rule
  https://github.com/kubernetes-monitoring/kubernetes-mixin/pull/126

- 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

- 为 Prometheus crd 添加 toleration
  https://github.com/coreos/kube-prometheus/pull/102

QINGCLOUD | KUBESPHERE

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

QINGCLOUD | KUBESPHERE

总结

- 一些优化建议：
  - 将经常使用、计算耗时的 PromQL 写成 Recording Rules；
  - Drop 无用指标；
  - 多个 Prometheus 实例，分工抓取不同目标；
  - 使用 Prometheus Operator 方式来配置 Prometheus；
  - query、storage 等调参。
- 后续工作：
  - 数据持久化: remote write, 对象存储。
  - 自定义监控

QINGCLOUD | KUBESPHERE

Demo

QINGCLOUD | KUBESPHERE

We’re open sourcing

- https://kubesphere.io/
- https://github.com/kubesphere/kubesphere