基于 Prometheus SLO 告警实战

×

宋佳洋

OPPO
高级后端工程师

目 录

第一部分

为什么基于 SLO 告警

什么是 SLO

Service Level Objectives

Here are some example Service Level Objectives for Parca.

Category	SLI	SLO
Write		
Availability	The proportion of successful `ProfileStoreService.WriteRaw` requests (such as from Parca agent), as measured by Parca's gRPC metrics interceptor.	99.9% in 4w
Latency	The proportion of sufficiently fast requests to `ProfileStoreService.WriteRaw`, as measured by Parca's gRPC metrics interceptor.	95% of requests in < 100ms in 4w

参考地址 https://www.parca.dev/docs/observability

SLI

- 状态码 >= 500
- 请求延迟 > 200ms
- 进程运行非 0 状态码退出

时间窗口

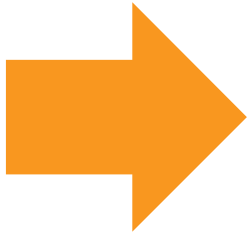
- 1w (7d)
- 4w(28d)
- 30d

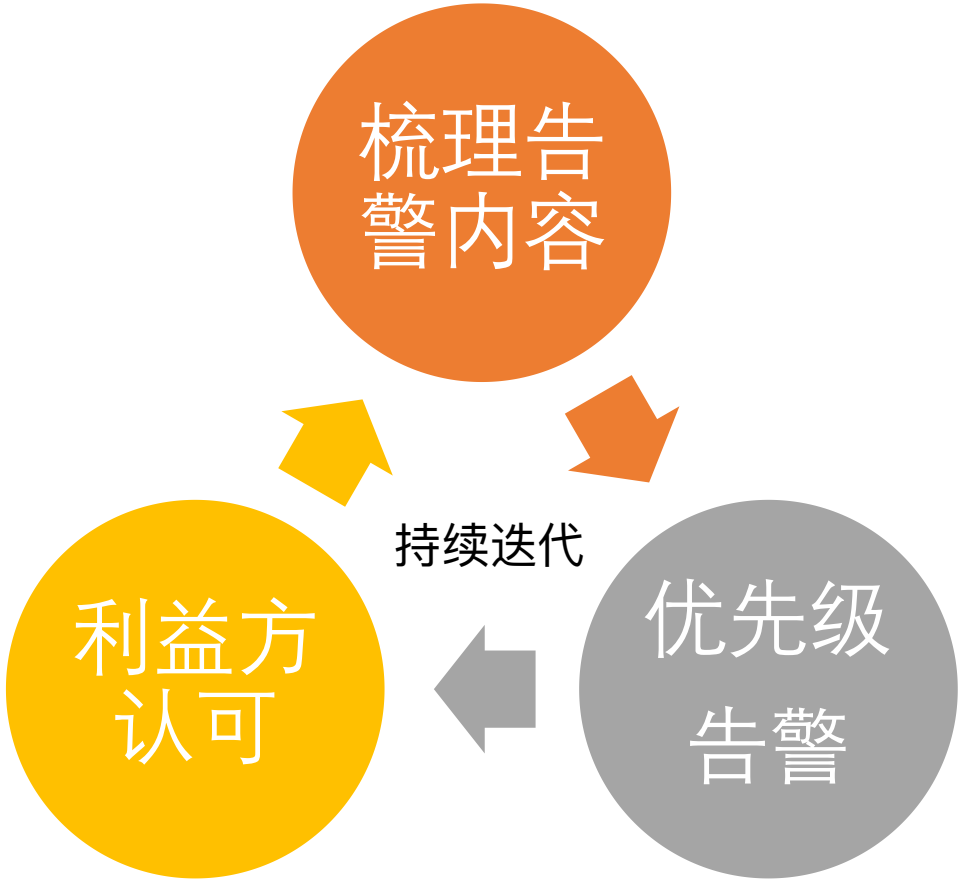
举个例子

- 时间周期：30天
- SLO：99.9%
- 错误预算：0.0999 (100−99.9)%
- 30 天总请求数：10000
- 允许的错误请求数：9.99 (10000 * 0.0999 / 100)

基于 SLO 告警重要性

~~100%~~





“

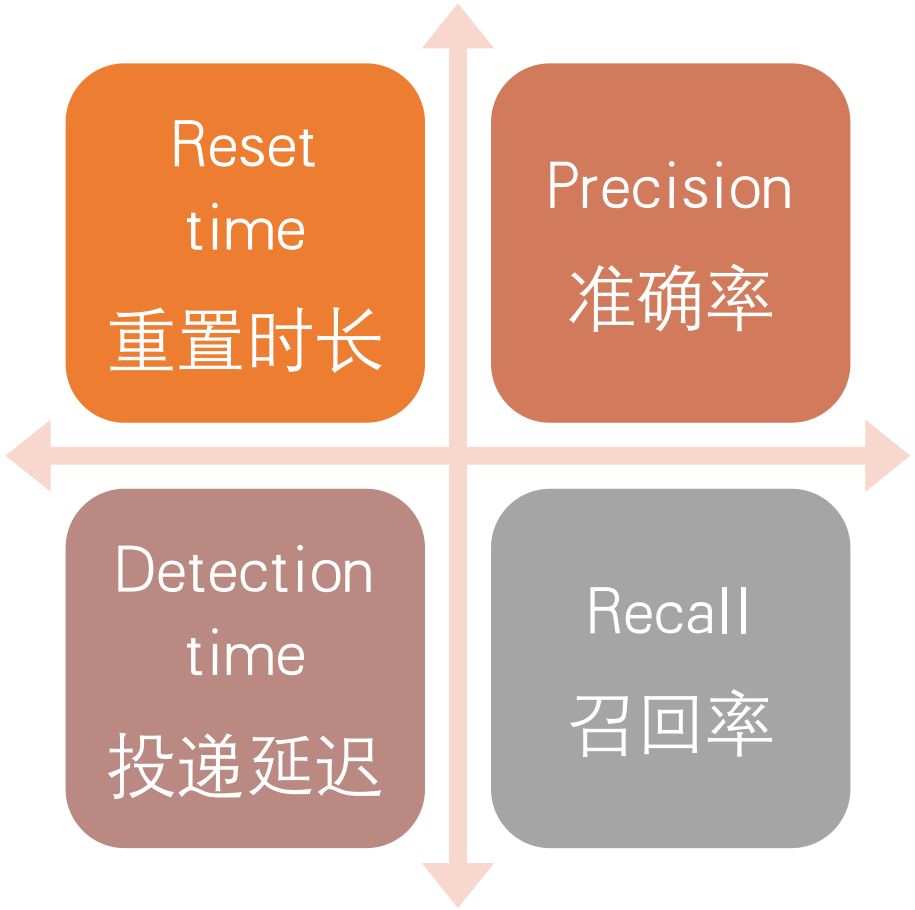
SRE Workbook-Chapter2

One could even claim that without SLOs, there is no need for SREs.

第二部分

SLO 告警指导思想

SLO 告警指导原则



两高两低，四者兼顾

- 准确率、召回率较高
- 投递延迟低、重置时长短

我们可能使用的一些策略

策略 1： 错误率 >= SLO 阀值

```
- alert: HighErrorRate
  expr: job:slo_errors_per_request:ratio_rate10m{job="myjob"} >= 0.001
```

问题： 准确率低

策略 2： 增加观察窗口

```
- alert: HighErrorRate
  expr: job:slo_errors_per_request:ratio_rate36h{job="myjob"} >= 0.001
```

问题： 重置时间长

策略 3： 告警持续性检测

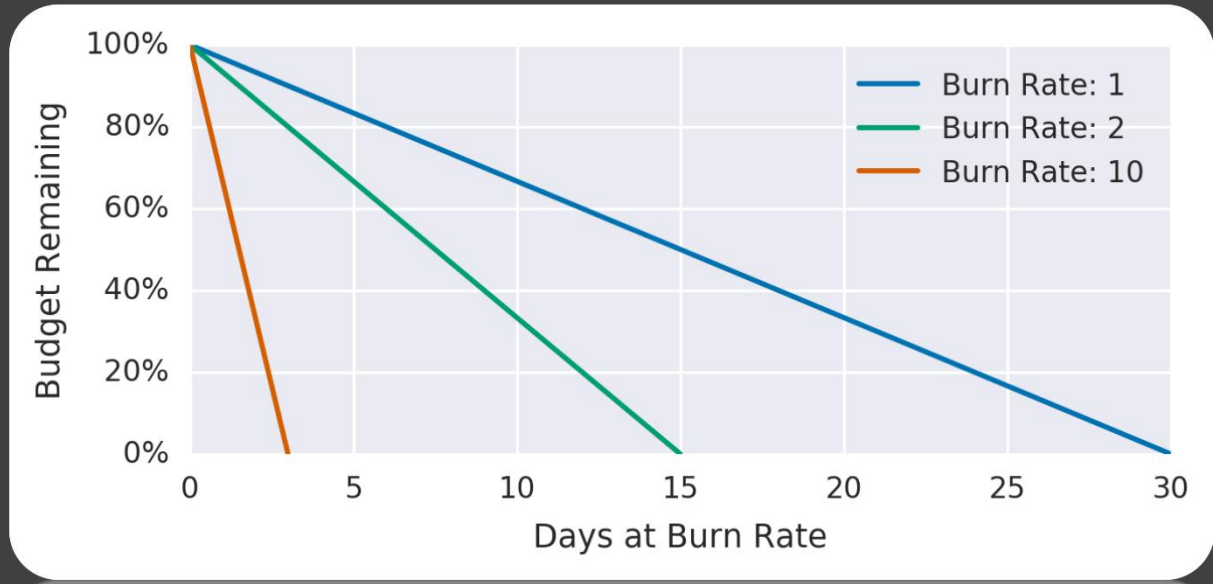
```
- alert: HighErrorRate
  expr: job:slo_errors_per_request:ratio_rate1m{job="myjob"} >= 0.001
  for: 1h
```

问题： 召回率低

策略 4： 基于单一燃烧率告警

```
- alert: HighErrorRate
  expr: job:slo_errors_per_request:ratio_rate1h{job="myjob"} >= 36 * 0.001
```

问题： 召回率较低



图片来源： Google SRE Workbook

策略 5： 基于多燃烧率告警

```
expr: (
        job:slo_errors_per_request:ratio_rate1h{job="myjob"} > (14.4 * 0.001)
      or
        job:slo_errors_per_request:ratio_rate6h{job="myjob"} > (6 * 0.001)
      )
severity: page

expr: job:slo_errors_per_request:ratio_rate3d{job="myjob"} > 0.001
severity: ticket
```

问题： 重置时间较长

SLO budget consumption	Time window	Burn rate	Notification
2%	1 hour	14.4	Page
5%	6 hours	6	Page
10%	3 days	1	Ticket

图片来源：Google SRE Workbook

策略 6： 基于多窗口多燃烧率告警

```
expr: (
        job:slo_errors_per_request:ratio_rate1h{job="myjob"} > (14.4 * 0.001)
      and
        job:slo_errors_per_request:ratio_rate5m{job="myjob"} > (14.4 * 0.001)
      )
    or
      (
        job:slo_errors_per_request:ratio_rate6h{job="myjob"} > (6 * 0.001)
      and
        job:slo_errors_per_request:ratio_rate30m{job="myjob"} > (6 * 0.001)
      )
severity: page
```

4 大指标平衡，做到最佳

Severity	Long window	Short window	Burn rate	Error budget consumed
Page	1 hour	5 minutes	14.4	2%
Page	6 hours	30 minutes	6	5%
Ticket	3 days	6 hours	1	10%

图片来源： Google SRE Workbook

第三部分

Prometheus 实践 MWMR 的挑战

手动编写 Prometheus rules 并不容易

需要编写多个时间窗口相关的 Record rule

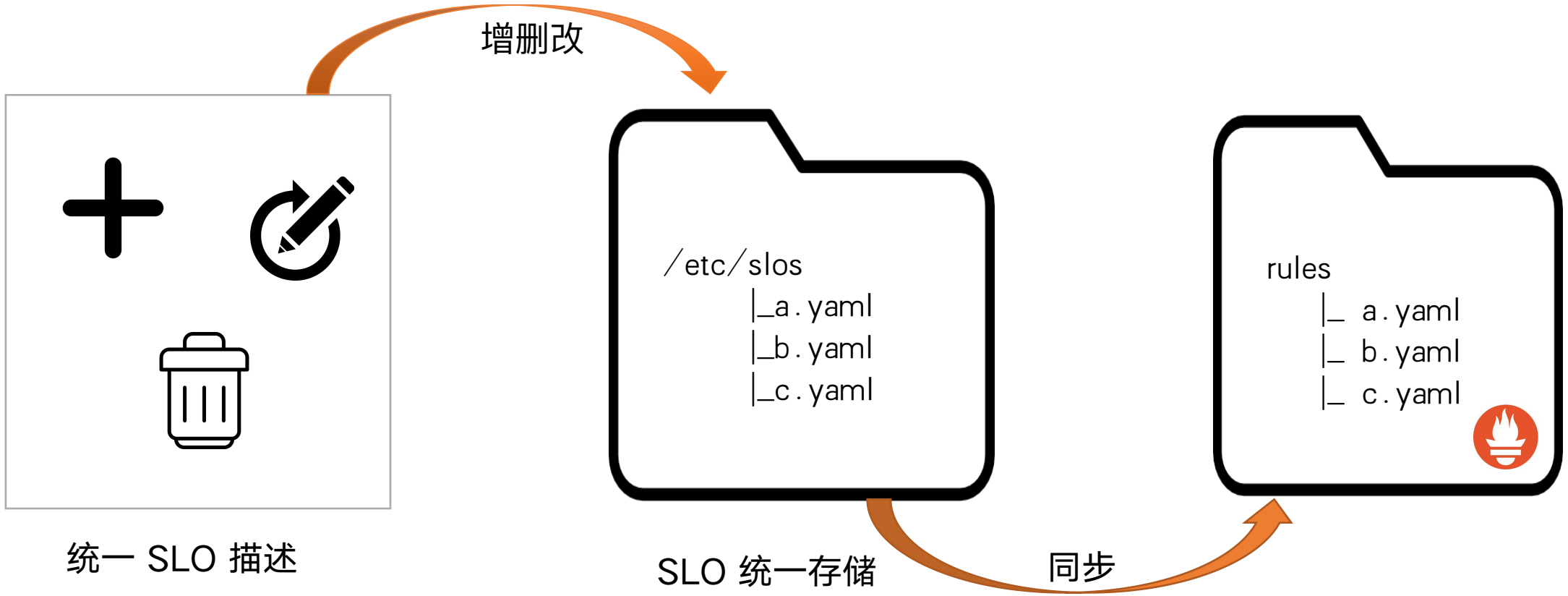
```
slo_errors_per_request:ratio_rate5m
slo_errors_per_request:ratio_rate30m
slo_errors_per_request:ratio_rate1h
……
```

Alert rule 复杂，需要考虑不同时间窗口和告警级别

```
expr: (
          job:slo_errors_per_request:ratio_rate1h{job="myjob"} > (14.4*0.001)
        and
          job:slo_errors_per_request:ratio_rate5m{job="myjob"} > (14.4*0.001)
        )
        or
        (
          job:slo_errors_per_request:ratio_rate6h{job="myjob"} > (6*0.001)
        and
          job:slo_errors_per_request:ratio_rate30m{job="myjob"} > (6*0.001)
        )
 severity: page
```

单个 SLO 对应多条（10+）Prometheus rules

SLO 配置与 Prometheus rules 准确同步



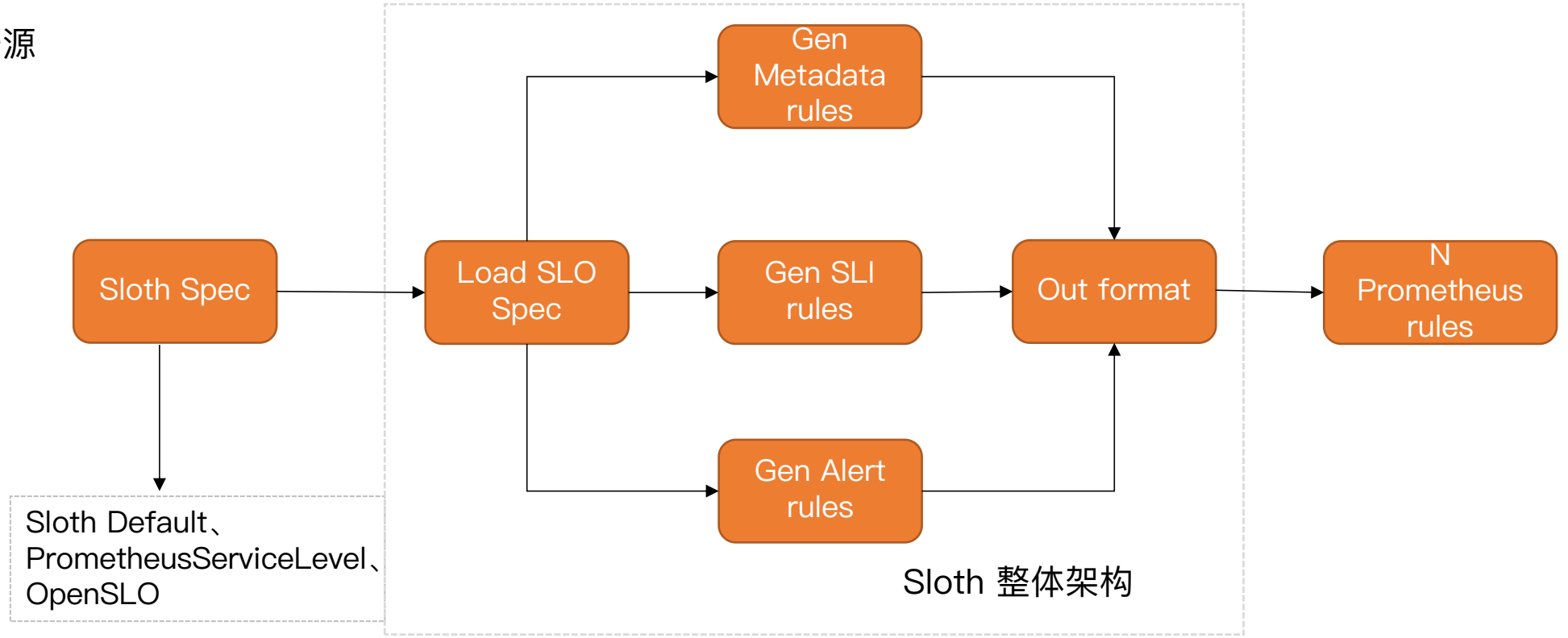
第四部分

基于 Sloth 实现 SLO 告警

Sloth 简介

Sloth 是一个简单易用的 Prometheus SLO 自动生成器、支持 命令行和 K8s Controller 两种使用方式，支持自定义告警窗口配置、提供开箱即用的 Grafana 看板。

- 1.5k star
- 2021/8 月开源



Sloth 整体架构

Sloth SLO 统一描述之 default spec

```
version: "prometheus/v1"
service: "myservice"
labels:
  owner: "myteam"
slos:
  - name: "requests-availability"
    objective: 99.9
    description: "Common SLO based on availability for HTTP request responses."
    sli:
      events:
        error_query: sum(rate(http_request_duration_seconds_count{job="myservice",code=~"(5..|429)"}[{{.window}}]))
        total_query: sum(rate(http_request_duration_seconds_count{job="myservice"}[{{.window}}]))
    alerting:
      name: MyServiceHighErrorRate
      ...
      page_alert:
        .....
      ticket_alert:
        ....
```

Sloth 另外两种 SLO spec

K8s CRD 格式

```yaml
apiVersion: sloth.slok.dev/v1
kind: PrometheusServiceLevel
metadata:
  name: sloth-slo-my-service
  namespace: monitoring
spec:
  service: "myservice"
  labels:
    owner: "myteam"
    repo: "myorg/myservice"
    tier: "2"
  slos:
    - name: "requests-availability"
      ... 和 default 单个 slo 配置相似 ...
```

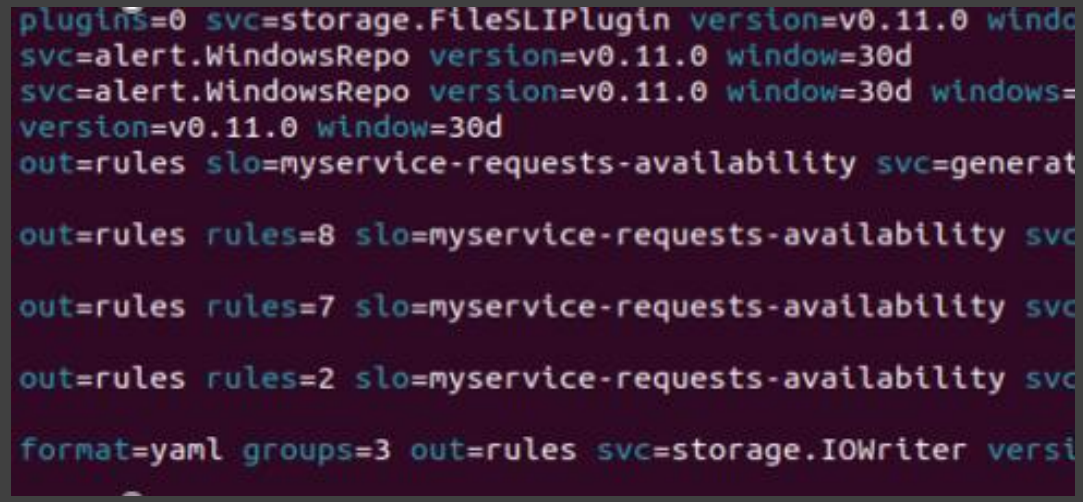
OpenSLO 格式

```yaml
apiVersion: openslo/v1alpha
kind: SLO
metadata:
  name: requests-availability
spec:
  service: my-service
  budgetingMethod: Occurrences
  objectives:
    - ratioMetrics:
        good:
          source: prometheus
          queryType: promql
          query:  xxxx
        total:
          source: prometheus
          queryType: promql
          query: xxxx
      target: 0.999
  timeWindows:
    - count: 30
```

Sloth 使用

命令行 cli

$ sloth generate –i slos –o rules ––slo–period–windows–path=./windows ––default–slo–period="30d"



K8s + Prometheus Operator

# 部署 sloth operator
$ kubectl apply —f https://raw.githubusercontent.com/slok/sloth/main/pkg/kubernetes/gen/crd/sloth.slok.dev_prometheusservicelevels.yaml

$ kubectl apply –f https://raw.githubusercontent.com/slok/sloth/main/deploy/kubernetes/raw/sloth.yaml

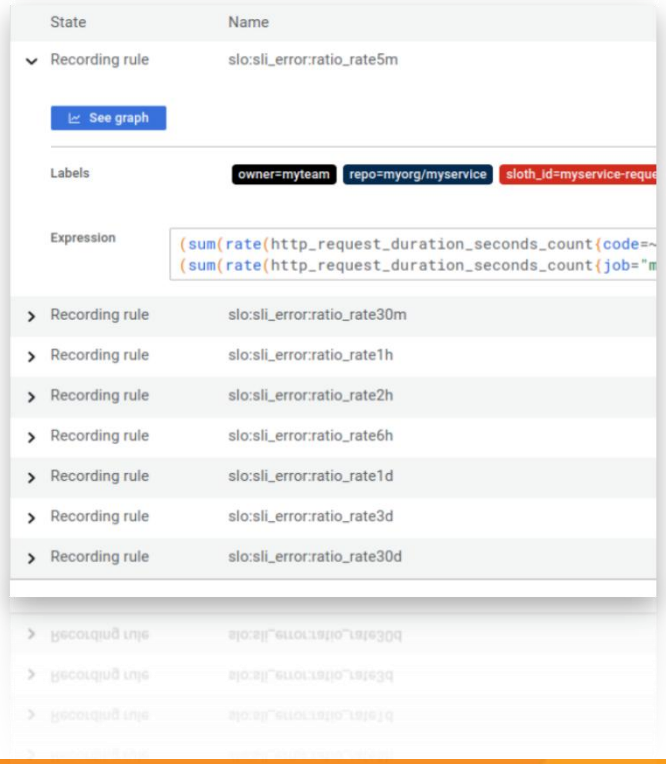
# 部署 sloth SLO
$ kubectl apply –f https://raw.githubusercontent.com/slok/sloth/main/examples/k8s–getting–started.yml

# 查看生成的 slos 和 promtheus rules
$ kubectl –n monitoring get slos
kubectl –n monitoring get prometheusrules

Sloth Prometheus rules 详解

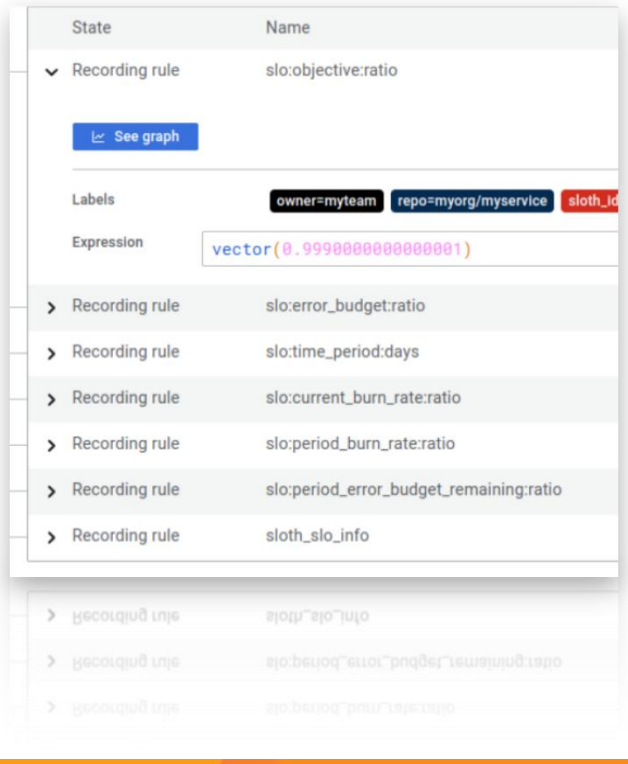
sli rules

- 8 个 record rules
- slo:sli_error:ratio_rate5m (30m、1h 、2h …



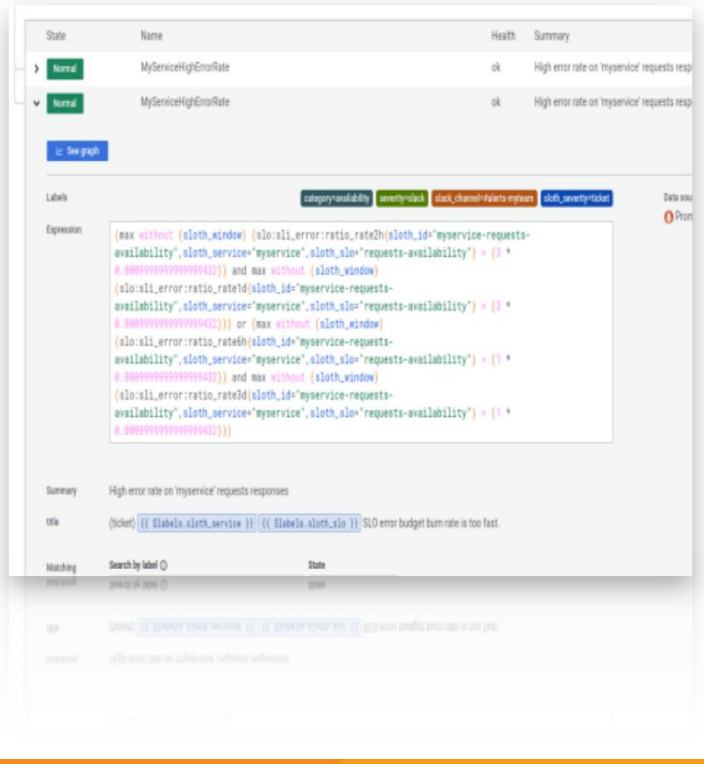
metadata rules

- 6 个 record rules
- 包含了 SLO 目标、错误预算、时间周期等



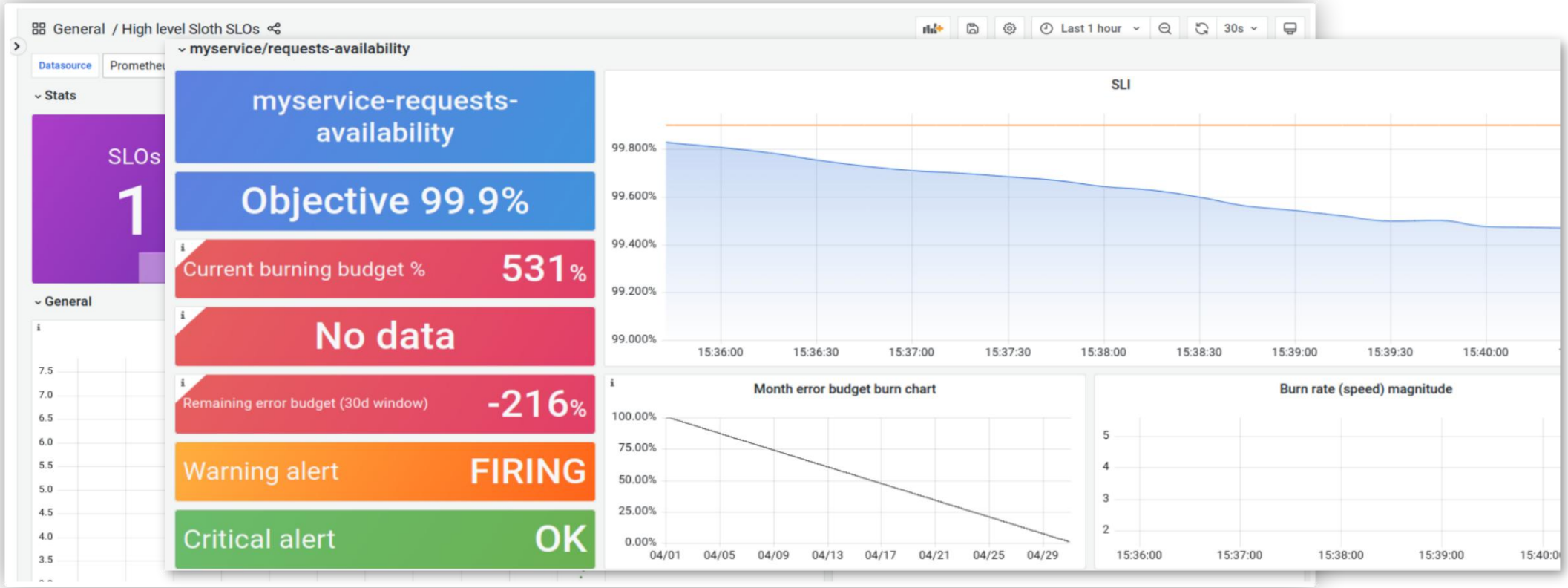
alert rules

- 2 个 alert rules
- 支持 MWMR



Sloth Dashboard 之 Grafana 模板

- 概览模板 id： 14643
- 详情模板 id： 14348



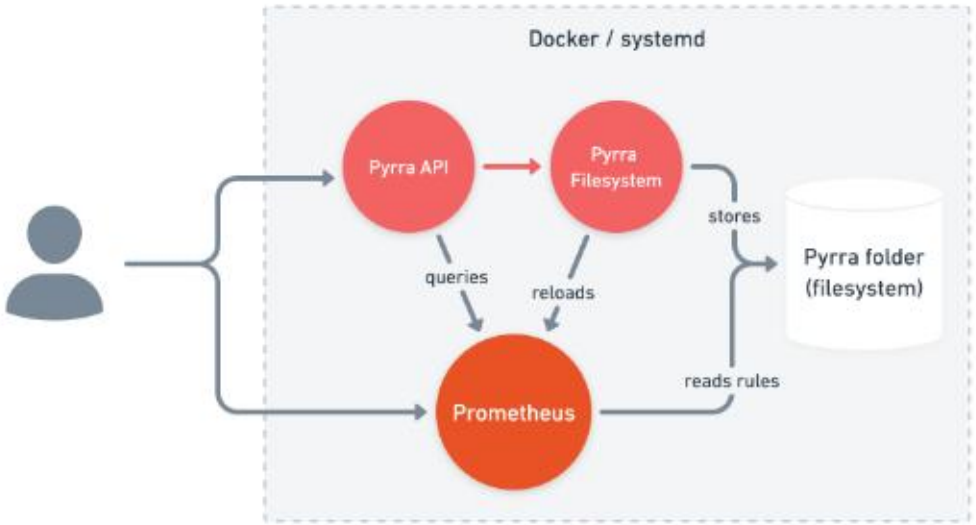
第五部分

基于 Pyrra 实现 SLO 告警

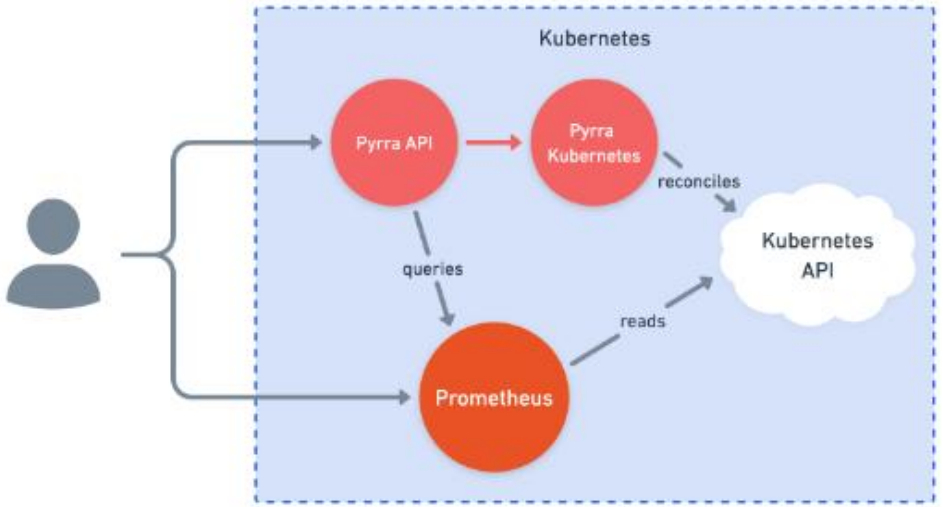
Pyrra 简介

Pyrra 是另外一个的 Prometheus SLO 生成器、支持 filesystem 和 kubernetes 两种模式，提供 UI 界面和 Grafana 模板作为可视化。

- 800+ star
- 2021/4 月开源



Pyrra with filesystem



Pyrra with kubernetes

Pyrra SLO 统一描述

```
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: pyrra-api-errors
  namespace: monitoring
  labels:
    prometheus: k8s
    role: alert-rules
    pyrra.dev/team: operations
spec:
  target: “99.9"
  window: 2w
  description: Pyrra's API requests and response errors over time grouped by route.
  indicator:
    ratio:
      errors:
        metric: http_requests_total{job="pyrra",code=~"5.."}
      total:
        metric: http_requests_total{job="pyrra"}
      grouping:
        - route
```

- 统一配置格式，无论是 K8s 还是 filesystem
- 一个 spec 定义一个 SLO
- SLO 通过标签进行过滤，缺少 service 这层概念，按照配置文件名进行组织
- 不支持配置自定义告警窗口

Pyrra 使用

Filesystem 模式

$ pyrra filesystem –config–files=“./slos/*.yaml” –prometheus–folder=./rules/

```
/ # pyrra filesystem --help
Usage: pyrra filesystem

Runs Pyrra's filesystem operator and backend for the API.

Flags:
  -h, --help                                    Show context-se

      --config-files="/etc/pyrra/*.yaml"        The folder wher
      --prometheus-url=http://localhost:9090    The URL to the
      --prometheus-folder="/etc/prometheus/pyrra/"    The folder wher
      --generic-rules                           Enabled generic
/ #
```

- 与 Prometheus server 处于同一实例，确保生成 rules 能被 Prometheus 加载。
- 通过配置的 prometheus-url ，进行 hot reload。

K8s 模式

部署 pyrra operator

$ kubectl apply –f ./config/crd/bases/pyrra.dev_servicelevelobjectives.yaml
$ kubectl apply –f ./config/rbac/role.yaml
$ kubectl apply –f ./config/api.yaml
$ kubectl apply –f ./config/kubernetes.yaml

部署 pyrra slos

$ kubectl apply –f ./examples/kubernetes/slos/

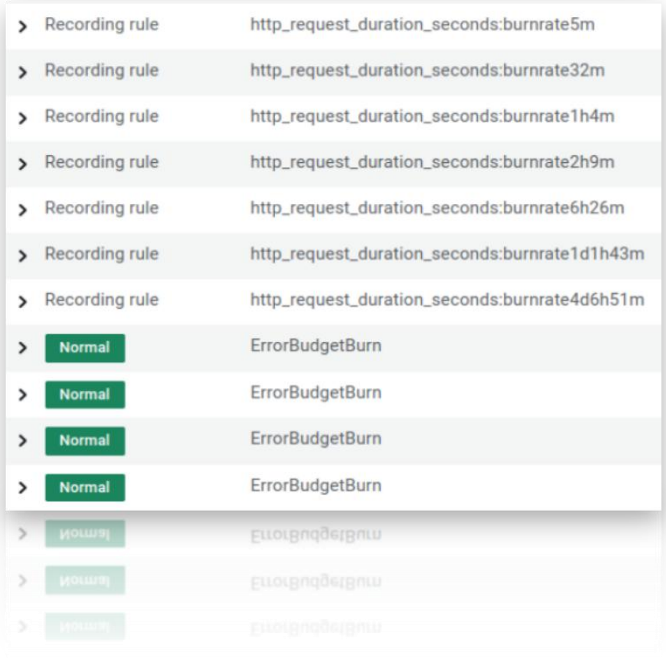
查看生成的 slos 和 promtheus rules

$ kubectl –n monitoring get servicelevelobjectives
$ kubectl –n monitoring get prometheusrules

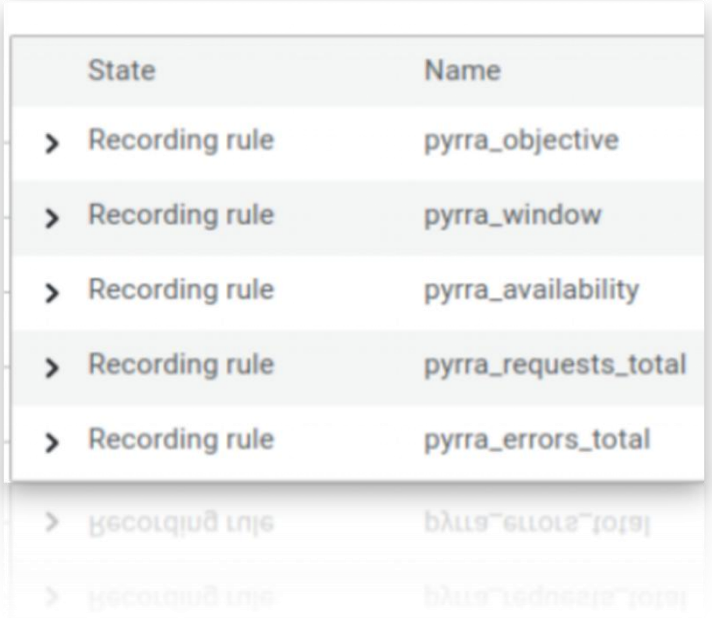
生成 Prometheus rules 详解

单个 SLO 包括三个 Prometheus 告警分组，分别为 xxx-availability、xxx-availability-generic、 xxx-availability-increase.

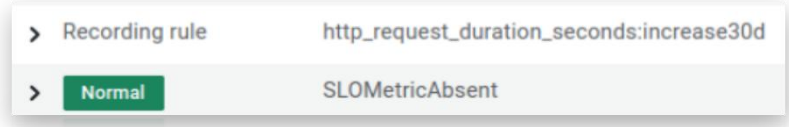
- **xxx-availability**： 包含 7 个 SLI record rules 和 4个 MWMR alert rules



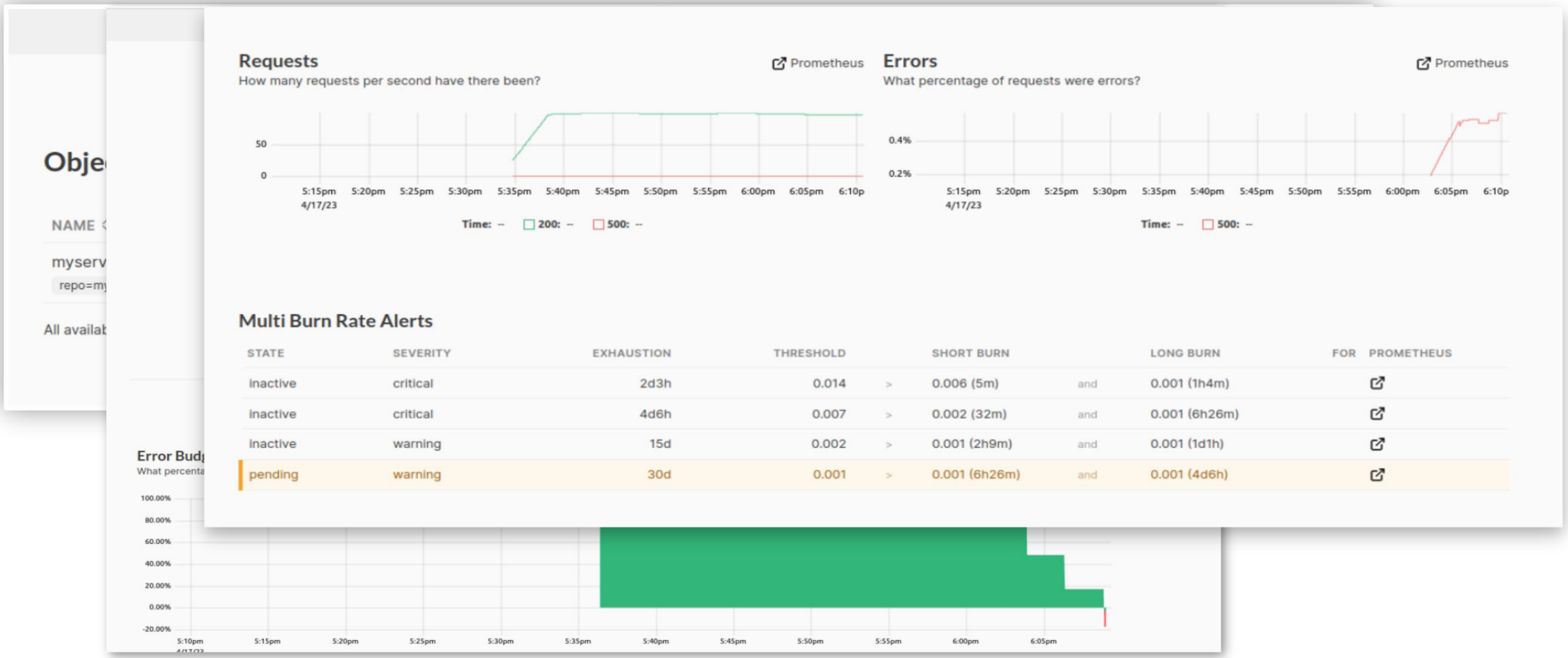
- **xxx-availability-generic**： 5 个 record rules，主要用于看板的 high level 汇总统计（ RED ）



- **xxx-availability-increase**： 1 个 record rule 和 1 个 alert rule， 主要用于统计整个窗口周期总请求数和 nodata 告警

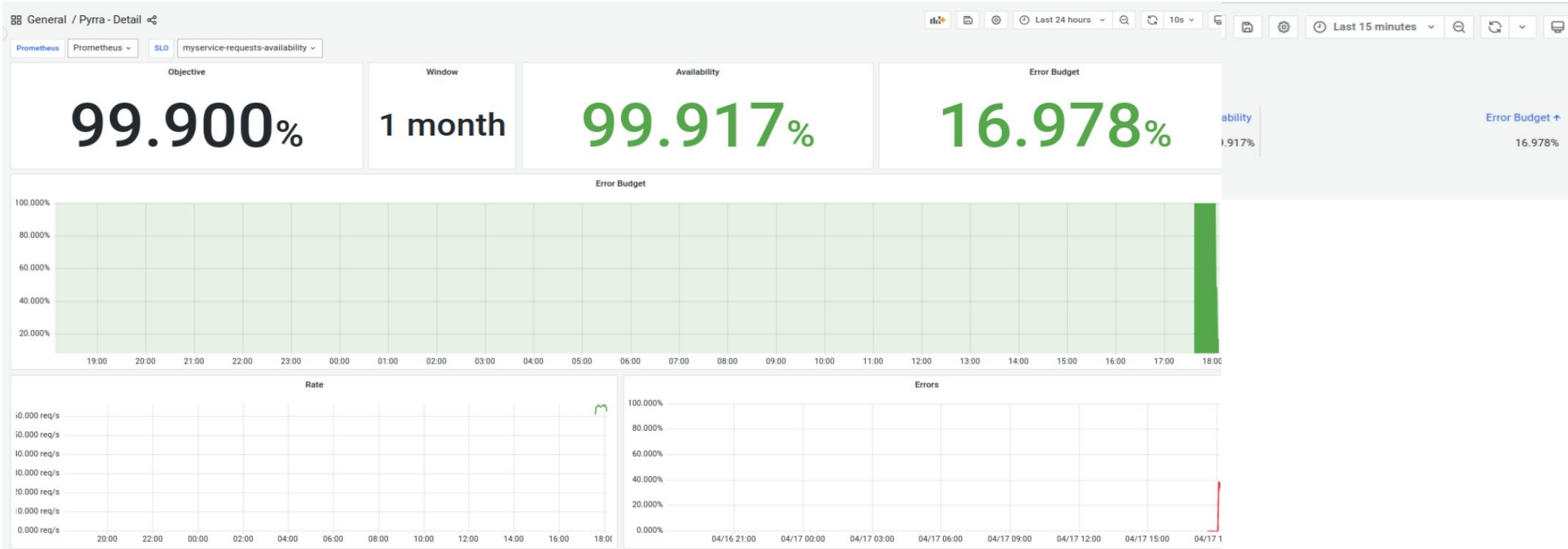


Pyrra 自带 Web UI



Pyrra 也提供 Grafana 模板

模板 JSON https://github.com/pyrra-dev/pyrra/tree/main/examples/grafana



Sloth Vs Pyrra

	Sloth	Pyrra
Github Star	1.5k	800+
K8s	支持	支持
Filesystem	不支持	支持
OpenSLO	支持	不支持
SLI 可读性	高	一般
Dashboard	Grafana	Grafana、Pyrra API

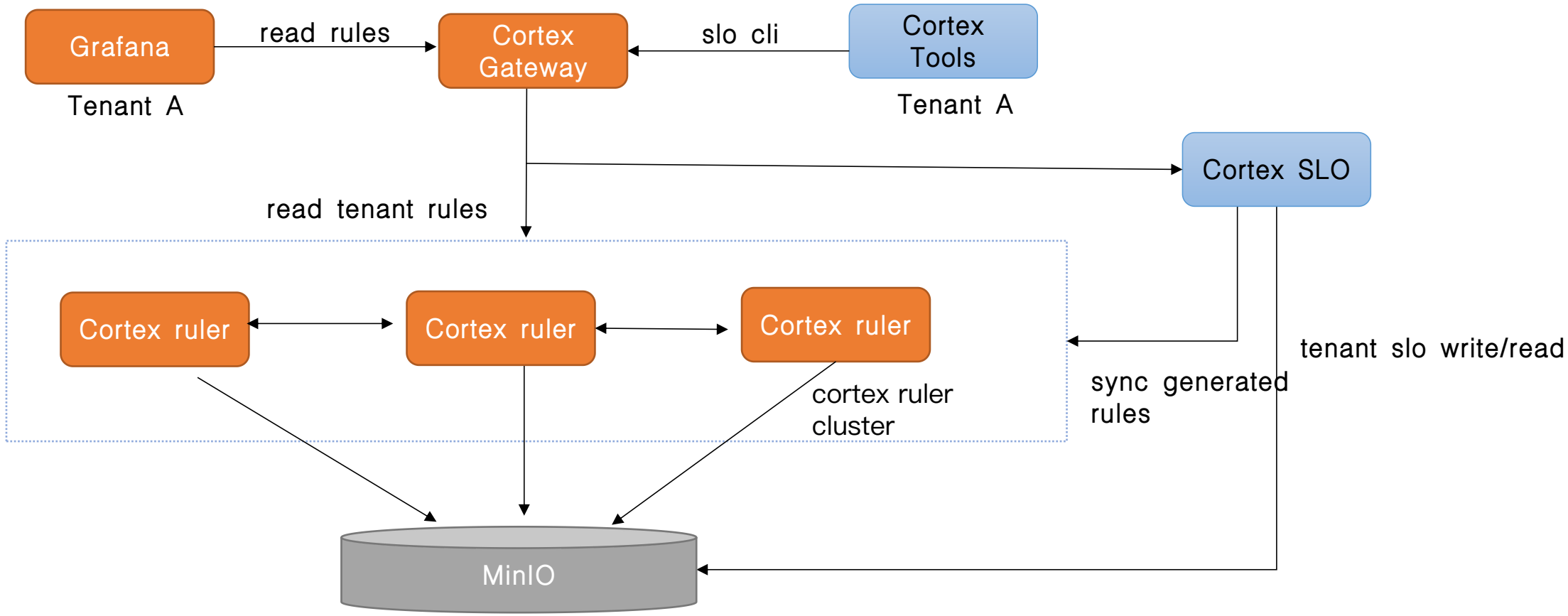
简单总结：

- 两个都是优秀的 Prometheus SLO generator，sloth 开源时间较早，协议支持广泛，pyrra 属于后起之秀，有自己的 dashboard。
- 因为 sloth 生成的 rules 可读性更强，如果有二开需求并直接使用 Grafana 作为看板，建议采用 sloth。

第六部分

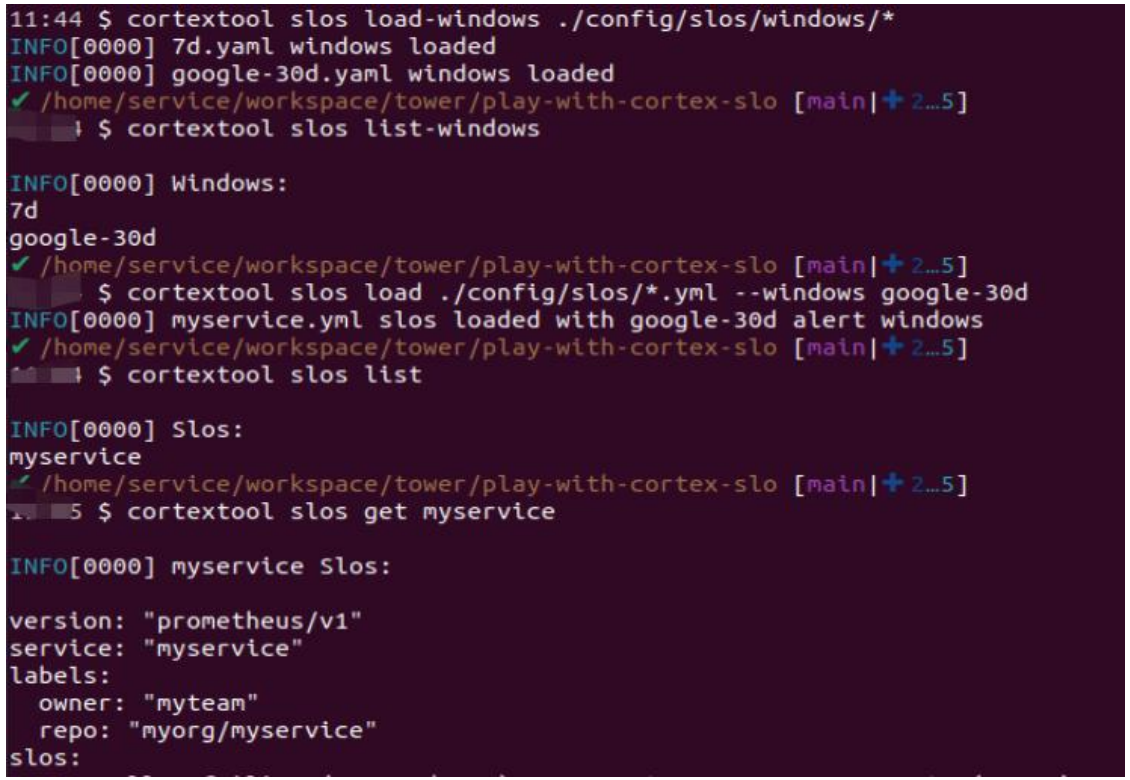
Cortex SLO 多租户方案实现

基于 Cortex 多租户 SLO 服务构建



Cortex 命令行工具扩展

```
# 配置 cortex-slo 地址以及租户信息

export CORTEX_ADDRESS=http://localhost:6666
export CORTEX_TENANT_ID=demo


#导入 和查询 windows

cortextool slos load-windows ./config/slos/windows/*
cortextool slos list-windows

# 导入和查询 slos

cortextool slos load ./config/slos/*.yml --windows google-30d
cortextool slos list
cortextool slos get myservice
```



一个例子： https://github.com/grafanafans/play-with-cortex-slo

简单总结

- 基于 SLO 告警非常重要，可以说没有 SLO 就没有SRE需要必要。
- 实践 SLO 告警一般使用 MWMR 方法。
- 手动维护 SLO 费时费力，可以使用 Sloth 和 Pyrra 来搭建。
- 如果有二开需求，个人更推荐 Sloth。
- Pyrra 的 Web UI 可以考虑作为 Stats page。

谢谢