云原生社区
Cloud Native Community

OBSERVABILITY
SUMMIT 2023
可观测性峰会 第1届

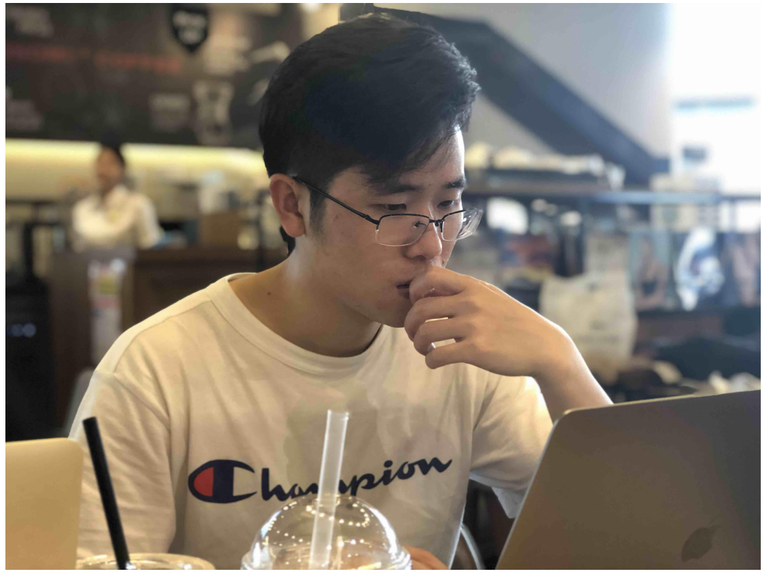
基于 Prometheus 的 SLO 告警实战

宋佳洋@OPPO

大纲

1. SLO 告警基础知识
2. 基于 Sloth 项目构建
3. 基于 Pyrra 项目构建
4. 多租户 SLO 服务构建

关于我



先后就职于七牛云、京东云等公司，目前在 OPPO 从事云计算相关工作。

爱好开源，目前主要关注 Go 和 云原生可观测领域、是开源项目 Prometheus、Cortex、Thanos 的代码贡献值。

songjiayang

微信公众号： Grafana 爱好者

为什么基于 SLO 告警

~~100%~~

- 梳理内容
- 优先级告警
- 利益方认可
- 持续迭代

没有 SLO，就没有 SRE

SLO 相关概念

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

告警级别

- page
- ticket

错误预算

- 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)

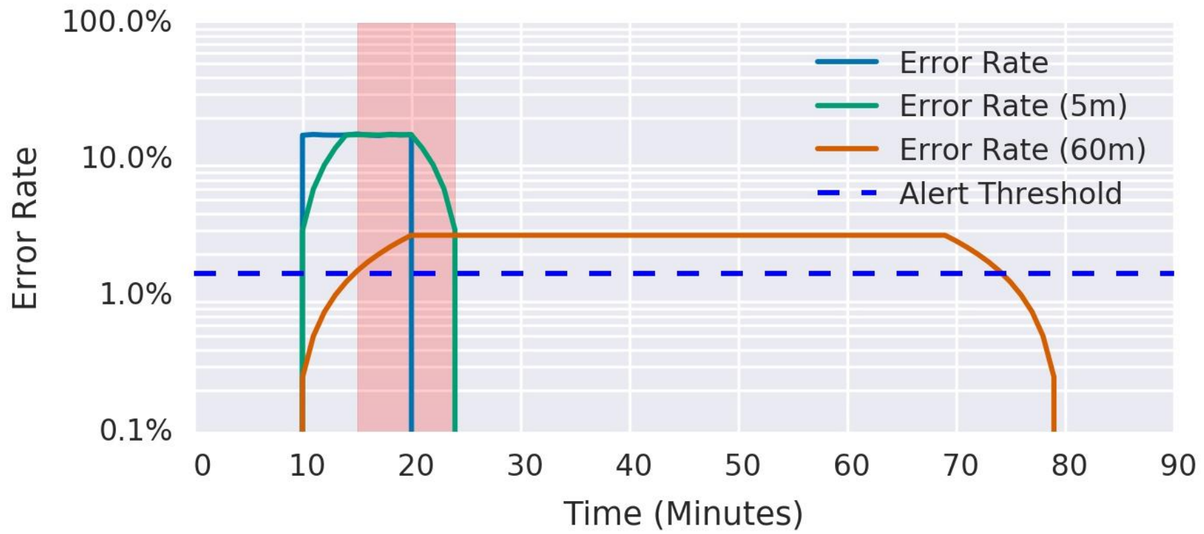
燃烧率

燃烧率	100%错误预算燃烧时间
1	30天
2	15天
60	12小时
1080	40分钟

SLO 告警指导思想-MWMR

- **Precision**：准确率
- **Recall**：召回率（故障漏过未告警）
- **Detection time**：投递延迟
- **Reset time**：告警重置时长

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%




```
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
```

参考 https://sre.google/workbook/alerting-on-slos

基于 Prometheus SLO 告警基础和挑战

开箱即用的 record and alert rule

```
# record
– record: code:prometheus_http_requests_total:sum
   expr: sum by (code) (prometheus_http_requests_total)

# alert
– alert: HighRequestLatency
  expr: job:request_latency_seconds:mean5m{job="myjob"} > 0.5
  labels:
    severity: page
```

加载和热更新

- 默认从本地文件加载。
- kill –hub pid
- curl –X POST/PUT http://localhost:9090/-/reload

Prometheus 告警基础

与时间窗口相关的多个 SLI rules

- slo_errors_per_request:ratio_rate5m
- slo_errors_per_request:ratio_rate30m

….

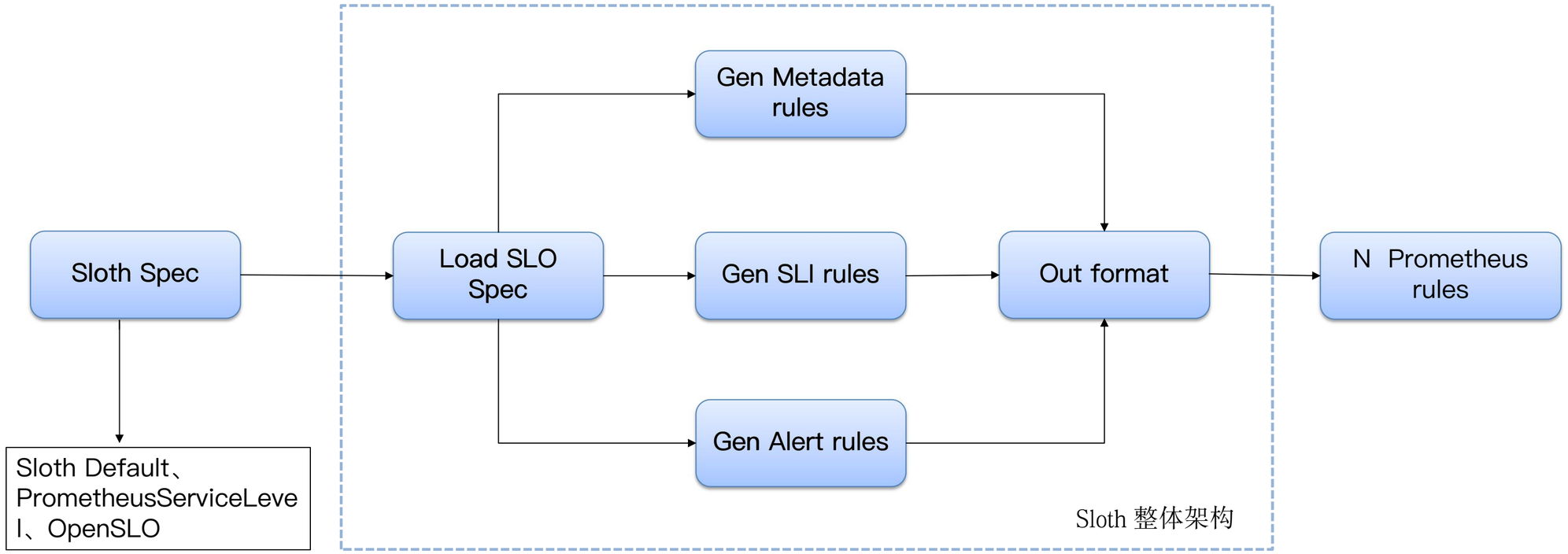
Alert rule 复杂, 需要考虑不同时间窗口和告警级别

```
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
```

Prometheus SLO 告警挑战

开源项目 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。



Sloth SLO 配置

Default sloth spec

```yaml
version: "prometheus/v1"
service: "myservice"
labels:
  owner: "myteam"
  repo: "myorg/myservice"
  tier: "2"
slos:
  # We allow failing (5xx and 429) 1 request every
  - name: "requests-availability"
    objective: 99.9
    description: "Common SLO based on availability
    sli:
      events:
        error_query: sum(rate(http_request_duration_
        total_query: sum(rate(http_request_duration_
    alerting:
      name: MyServiceHighErrorRate
      labels:
        category: "availability"
      annotations:
        # Overwrite default Sloth SLO alert summmary
        summary: "High error rate on 'myservice' rec
      page_alert:
        labels:
          severity: pageteam
          routing_key: myteam
      ticket_alert:
        labels:
          severity: "slack"
          slack_channel: "#alerts-myteam"
```

以 service 来组织，包含多个 slos

公用标签

Alert 通用标签

Page 级别告警标签

Ticket 级别告警标签

K8s CRD

```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"
      objective: 99.9
      description: "Common SLO based on availability
      sli:
        events:
          errorQuery: sum(rate(http_request_duration
          totalQuery: sum(rate(http_request_duration
      alerting:
        name: MyServiceHighErrorRate
        labels:
          category: "availability"
        annotations:
          summary: "High error rate on 'myservice' r
        pageAlert:
          labels:
            severity: pageteam
            routing_key: myteam
        ticketAlert:
          labels:
            severity: "slack"
            slack_channel: "#alerts-myteam"
```

Open SLO

```yaml
apiVersion: openslo/v1alpha
kind: SLO
metadata:
  name: sloth-slo-my-service
  displayName: Requests Availability
spec:
  service: my-service
  description: "Common SLO based on availability
  budgetingMethod: Occurrences
  objectives:
    - ratioMetrics:
        good:
          source: prometheus
          queryType: promql
          query: sum(rate(http_request_duration_s
        total:
          source: prometheus
          queryType: promql
          query: sum(rate(http_request_duration_s
      target: 0.999
  timeWindows:
    - count: 30
      unit: Day
```

Sloth AlertWindows 配置

```yaml
apiVersion: "sloth.slok.dev/v1"
kind: "AlertWindows"
spec:
  sloPeriod: 30d
  page:
    quick:
      errorBudgetPercent: 2
      shortWindow: 5m
      longWindow: 1h
    slow:
      errorBudgetPercent: 5
      shortWindow: 30m
      longWindow: 6h
  ticket:
    quick:
      errorBudgetPercent: 10
      shortWindow: 2h
      longWindow: 1d
    slow:
      errorBudgetPercent: 10
      shortWindow: 6h
      longWindow: 3d
```

SLO 时间周期

不同告警级别

不同错误预算/燃烧率

多窗口

```yaml
apiVersion: sloth.slok.dev/v1
kind: AlertWindows
spec:
  sloPeriod: 7d
  page:
    quick:
      errorBudgetPercent: 8
      shortWindow: 5m
      longWindow: 1h
    slow:
      errorBudgetPercent: 12.5
      shortWindow: 30m
      longWindow: 6h
  ticket:
    quick:
      errorBudgetPercent: 20
      shortWindow: 2h
      longWindow: 1d
    slow:
      errorBudgetPercent: 42
      shortWindow: 6h
      longWindow: 3d
```

AlertWindows 主要用于MWMR 告警配置，sloth 默认包含了 google-30d 和 google-28d 两个配置，可自定义。

https://github.com/slok/sloth/tree/main/internal/alert/window

Sloth CLI

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

```
INFO[0000] SLI plugins loaded                            plugins=0 svc=storage.FileSLIPlugin version=v0.11.0 window=30d
INFO[0000] Using custom slo period windows catalog       svc=alert.WindowsRepo version=v0.11.0 window=30d
INFO[0000] SLO period windows loaded                     svc=alert.WindowsRepo version=v0.11.0 window=30d windows=2
INFO[0000] Generating from Prometheus spec               version=v0.11.0 window=30d
INFO[0000] Multiwindow-multiburn alerts generated        out=rules slo=myservice-requests-availability svc=generate.prometheu
s.Service version=v0.11.0 window=30d
INFO[0000] SLI recording rules generated                 out=rules rules=8 slo=myservice-requests-availability svc=generate.p
rometheus.Service version=v0.11.0 window=30d
INFO[0000] Metadata recording rules generated            out=rules rules=7 slo=myservice-requests-availability svc=generate.p
rometheus.Service version=v0.11.0 window=30d
INFO[0000] SLO alert rules generated                     out=rules rules=2 slo=myservice-requests-availability svc=generate.p
rometheus.Service version=v0.11.0 window=30d
INFO[0000] Prometheus rules written                      format=yaml groups=3 out=rules svc=storage.IOWriter version=v0.11.0
window=30d
```

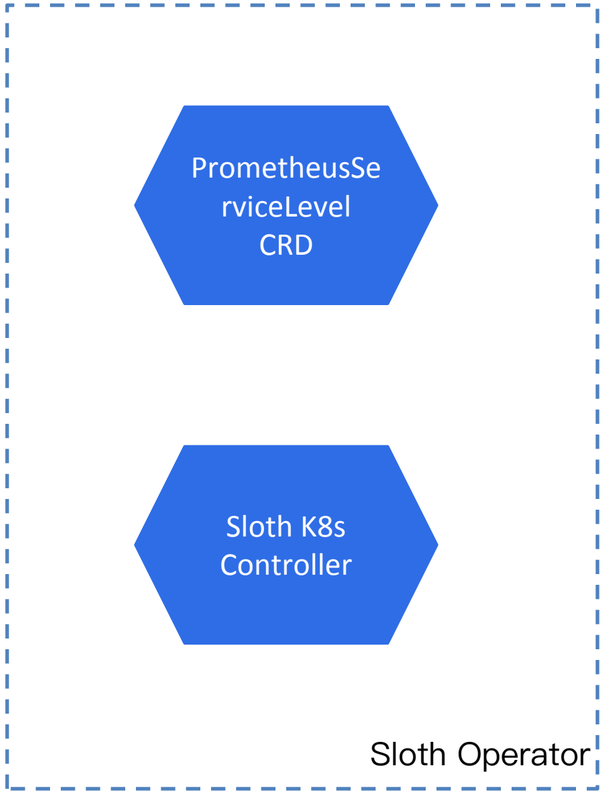
sloth generate –i slos –o rules ––slo–period–windows–path=./windows ––default–slo–period=“28d"

```
INFO[0000] SLI plugins loaded                            plugins=0 svc=storage.FileSLIPlugin
INFO[0000] Using custom slo period windows catalog       svc=alert.WindowsRepo version=v0.11
INFO[0000] SLO period windows loaded                     svc=alert.WindowsRepo version=v0.11
error: "generate" command failed: invalid default slo period: window period 672h0m0s missing
```

- 支持单个文件和目录批量生成。
- slo–period–windows 会覆盖默认 alert windows 配置
- default–slo–period 对应 alert windows 不存在，会报错

Sloth 与 K8s

Prometheus Operator

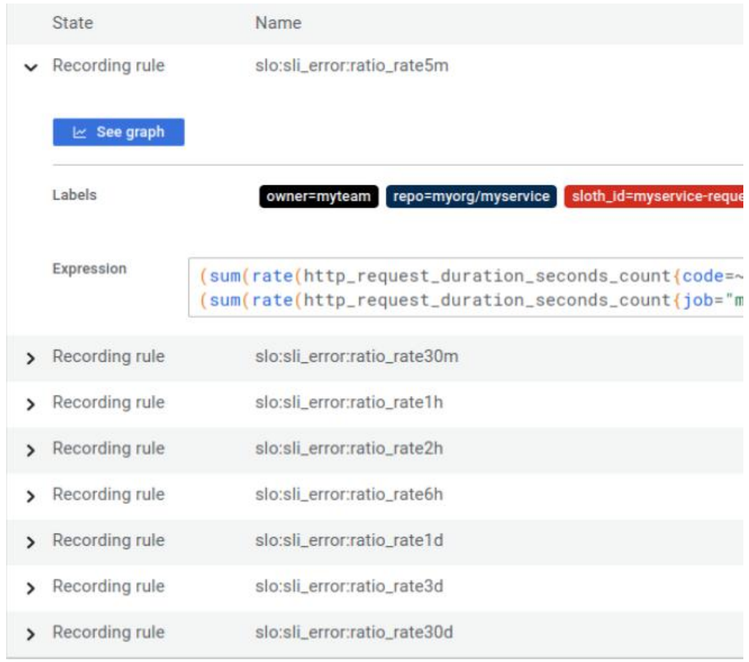



```
# 部署 sloth operator

kubectl apply —f
https://raw.githubusercontent.com/slok/sloth/main/
pkg/kubernetes/gen/crd/sloth.slok.dev_prometheusservicelevels.ya
ml

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
```

生成 Prometheus rules 详解

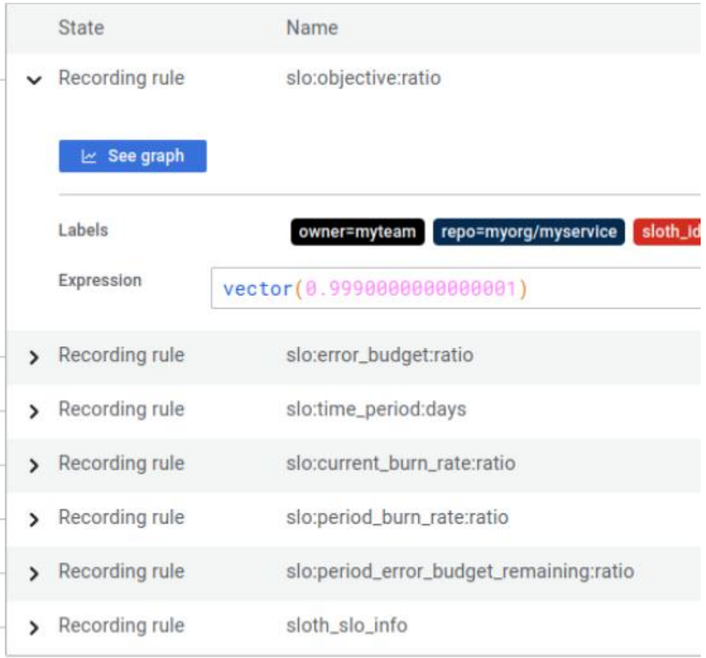
sli rules

- 8 个 record rules
- slo:sli_error:ratio_rate5m (30m、1h 、2h 、6h ...)



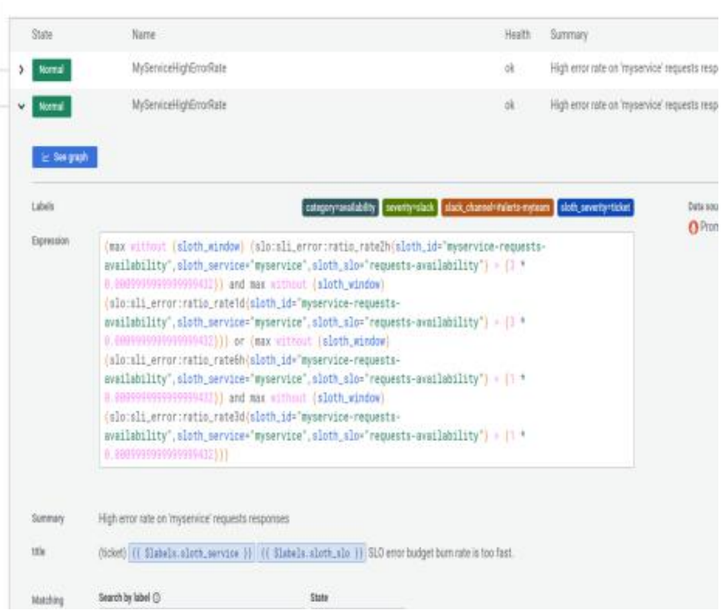
metadata rules

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



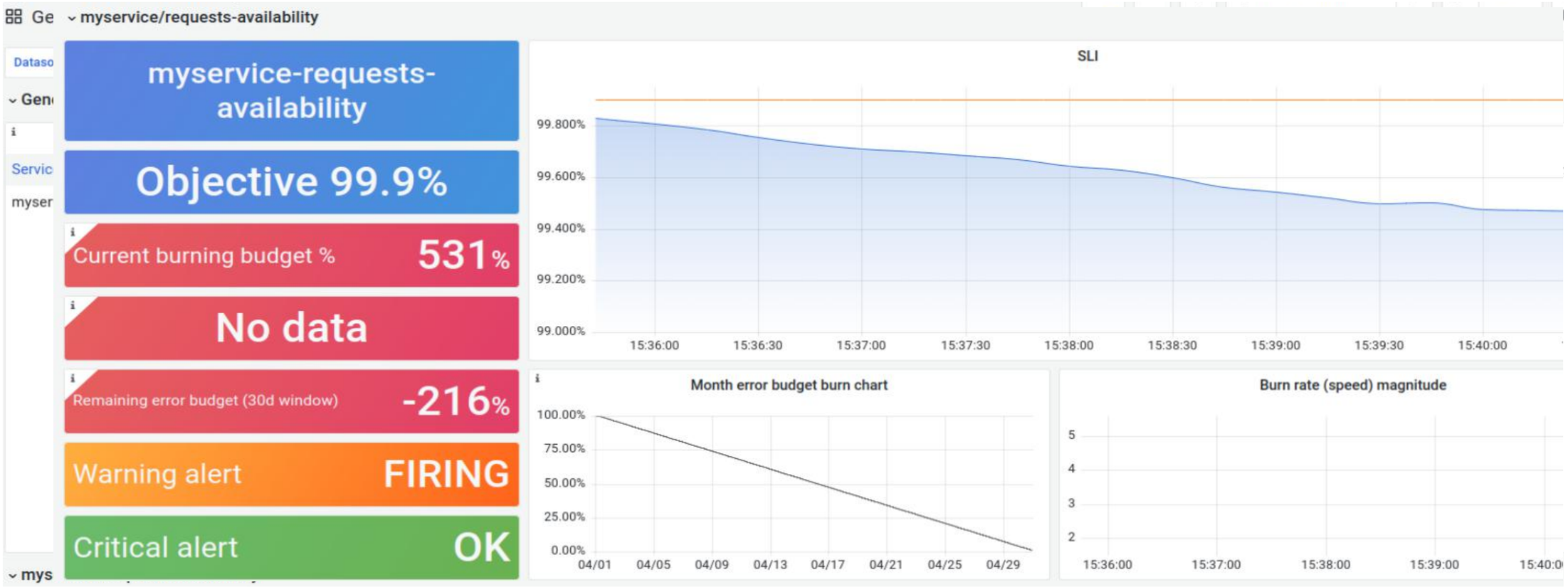
alert rules

- 2 个 alert rules
- 支持 MWMR



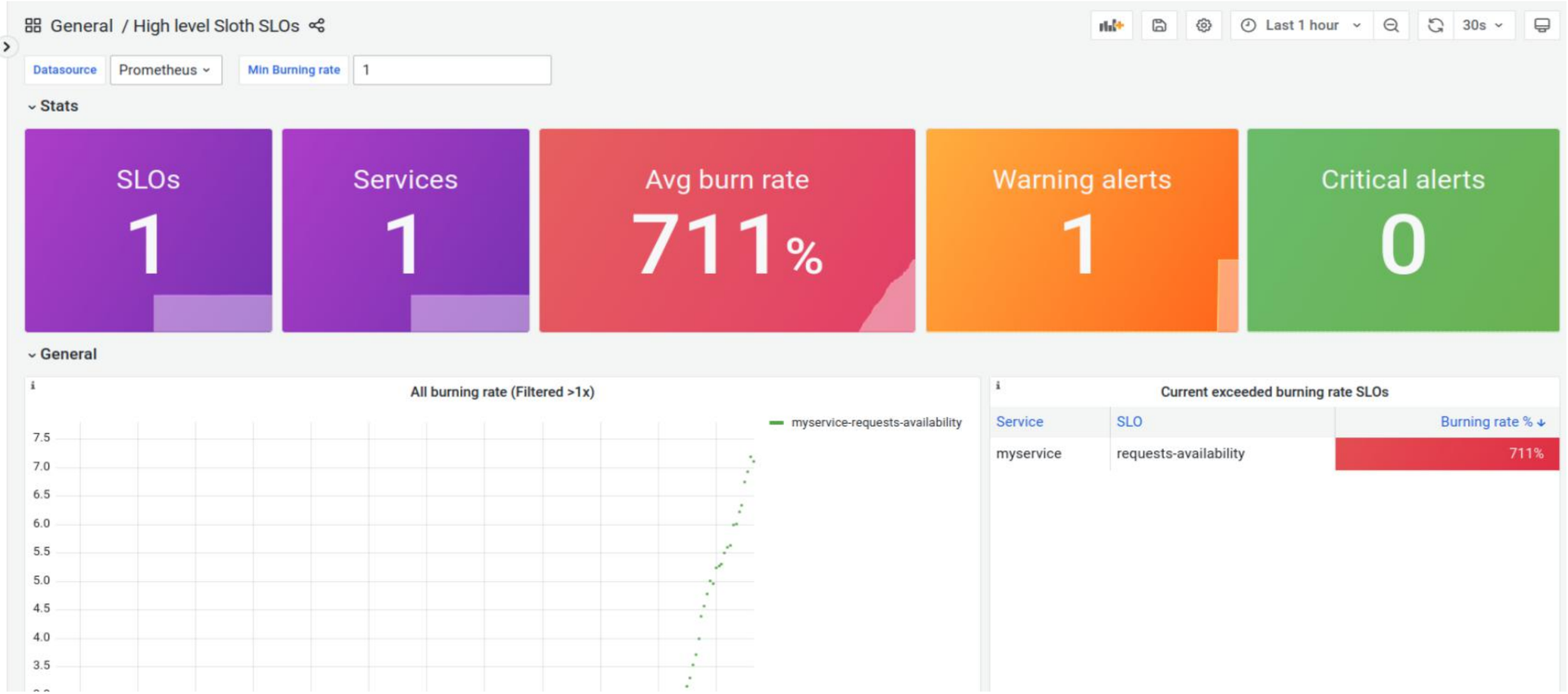
Sloth Dashboard-Details

Template ID： 14348



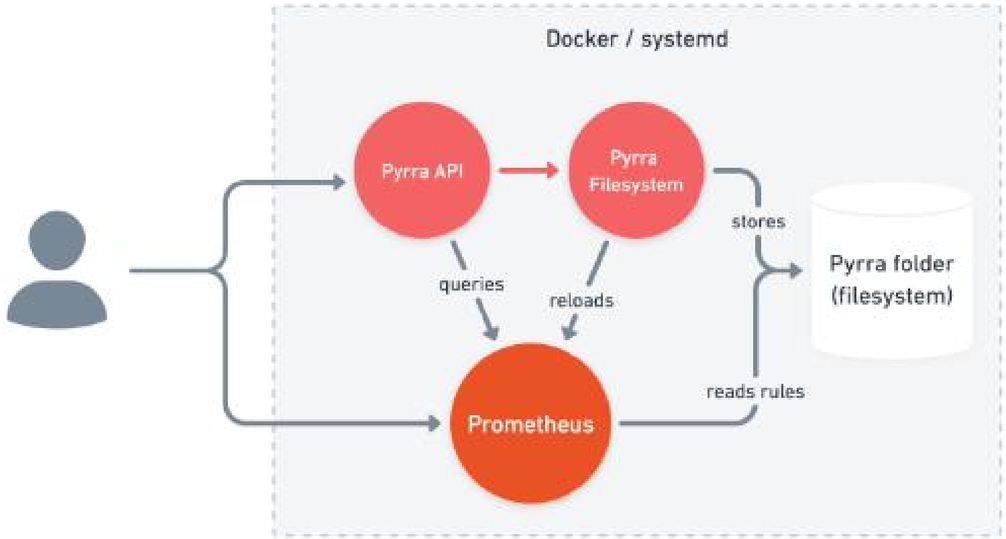
Sloth Dashboard-High Level

Template ID： 14348

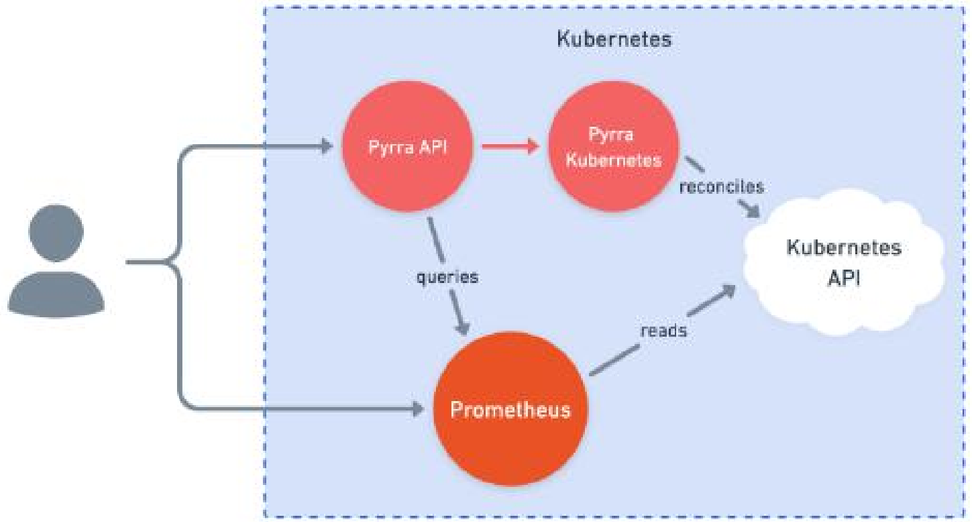


开源项目 Pyrra 介绍

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。



Pyrra with filesystem



Pyrra with kubernetes

Pyrra SLO 配置

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: pyrra-api-errors
  namespace: monitoring
  labels:
    prometheus: k8s
    role: alert-rules
    pyrra.dev/team: operations # Any labels prefixed with 'pyrra.dev/' will be propaga
spec:
  target: "99"
  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
```

sli 和 alert ruels 标签

SLO 目标值

时间窗口

Sli 分组

- 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
- 不支持自定义告警窗口配置

Filesystem 和 K8s 模式使用

主要使用 pyrra filesystem 命令

```
/ # 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，pyrra 加载目录配置，生成 rules 能够确保被 Prometheus server 加载。
- 通过配置的 prometheus-url ,进行 hot reload让 Promettheus 重新加载配置。

Filesystem 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
```

K8s 模式

Pyrra 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。

Recording rule	http_request_duration_seconds:burnrate5m
Recording rule	http_request_duration_seconds:burnrate32m
Recording rule	http_request_duration_seconds:burnrate1h4m
Recording rule	http_request_duration_seconds:burnrate2h9m
Recording rule	http_request_duration_seconds:burnrate6h26m
Recording rule	http_request_duration_seconds:burnrate1d1h43m
Recording rule	http_request_duration_seconds:burnrate4d6h51m
Normal	ErrorBudgetBurn
Normal	ErrorBudgetBurn
Normal	ErrorBudgetBurn
Normal	ErrorBudgetBurn

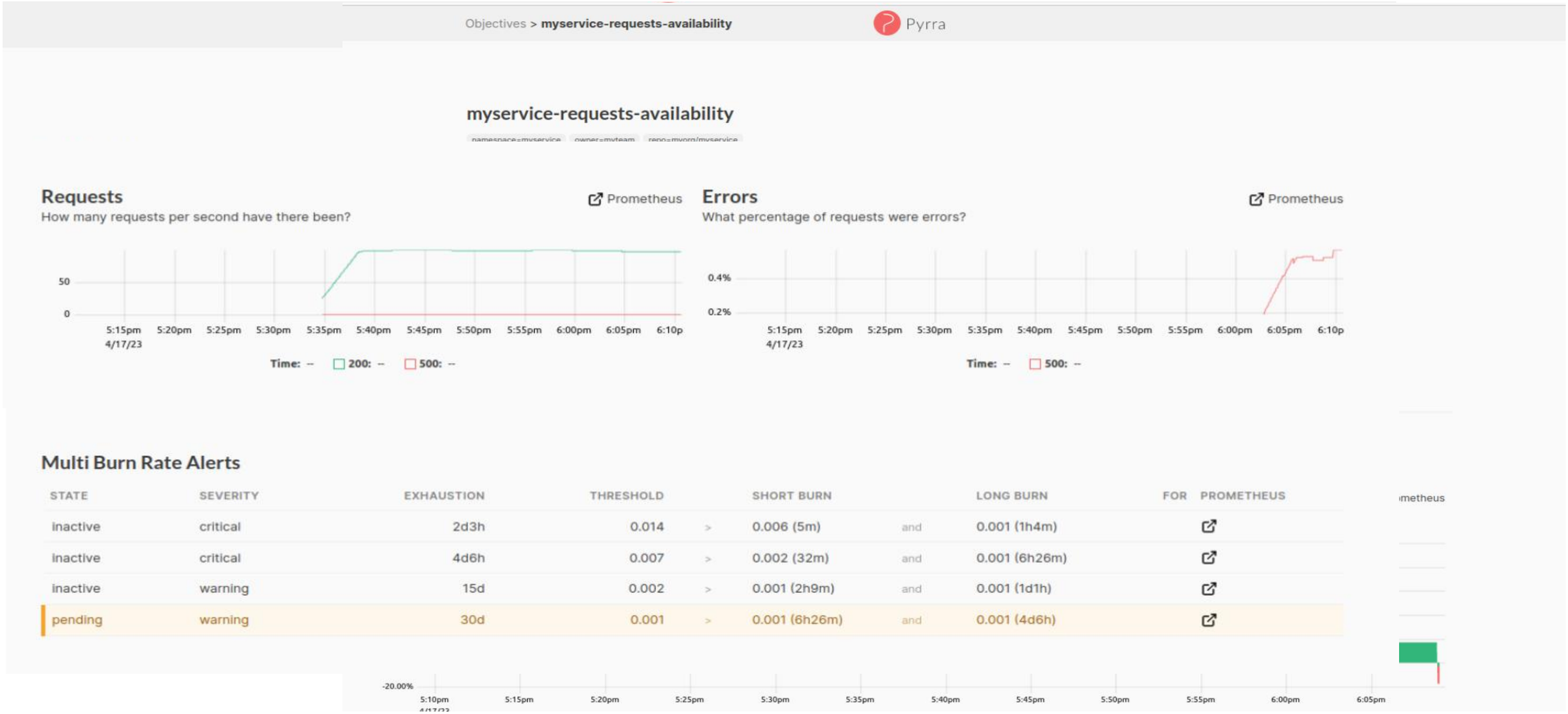
xxx-availability-generic： 5 个 record rules， 主要用于看板的 high level 汇总统计（ RED ）。

State	Name
Recording rule	pyrra_objective
Recording rule	pyrra_window
Recording rule	pyrra_availability
Recording rule	pyrra_requests_total
Recording rule	pyrra_errors_total

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。

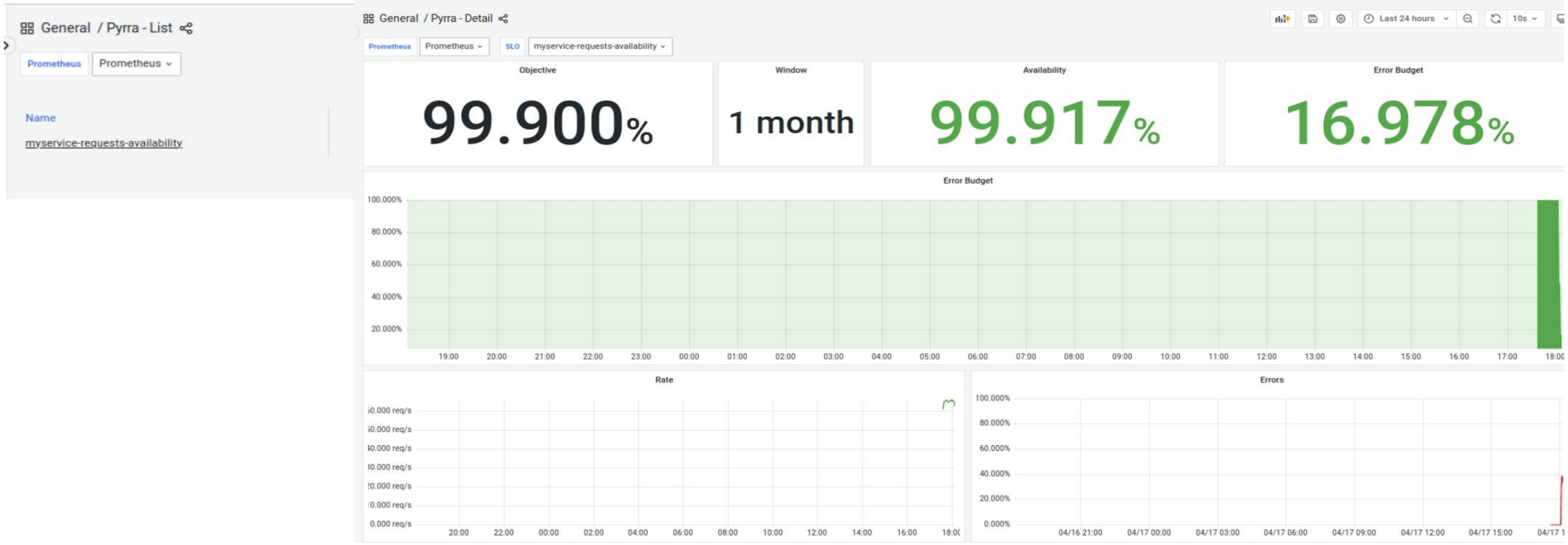
Recording rule	http_request_duration_seconds:increase30d
Normal	SLOMetricAbsent

Pyrra Dashboard-API



Pyrra Dashboard-Grafana

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



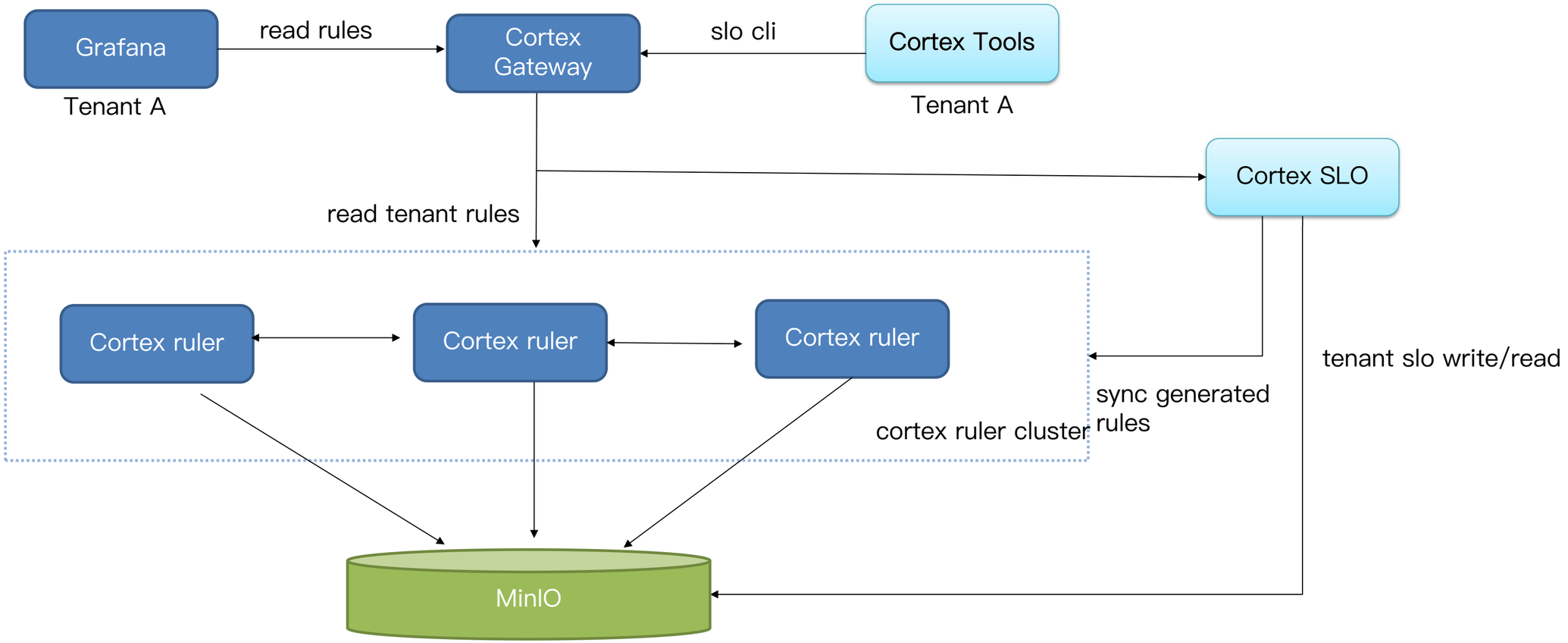
Sloth Vs Pyrra

	Sloth	Pyrra
Github Star	1.5k	800+
K8s	支持	支持
Filesystem	不支持	支持
CLI	支持	不支持
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。

基于 Cortex 的多租户 SLO 服务构建



Cortex-tools 扩展

统一封装到 slos 子命令

```
$ ./cortextool slos --help
usage: cortextool slos [<flags>] <command> [<args> ...]

View & edit slos stored in cortex.

Flags:
  --help          Show context-sensitive help (also try --help
  --authToken=""  Authentication token for bearer token or JWT

Subcommands:
  slos list [<flags>]
    List the slos currently in the telnant.

  slos load [<flags>] <files>...
    Load a set of slos to a designated cortex endpoint.

  slos get [<flags>] <service>
    Retrieve the sevice slos from the cortex.

  slos delete [<flags>] <service>
    Remove the sevice slos from the cortex.

  slos list-windows [<flags>]
    List slo windows currently in the telnant.

  slos load-windows [<flags>] <files>...
    Load a set of windows to a designated cortex endpoint.

  slos get-windows [<flags>] [<windows>]
    Get windows to from remote store.

  slos delete-windows [<flags>] [<windows>]
    Get windows to from remote store.
```

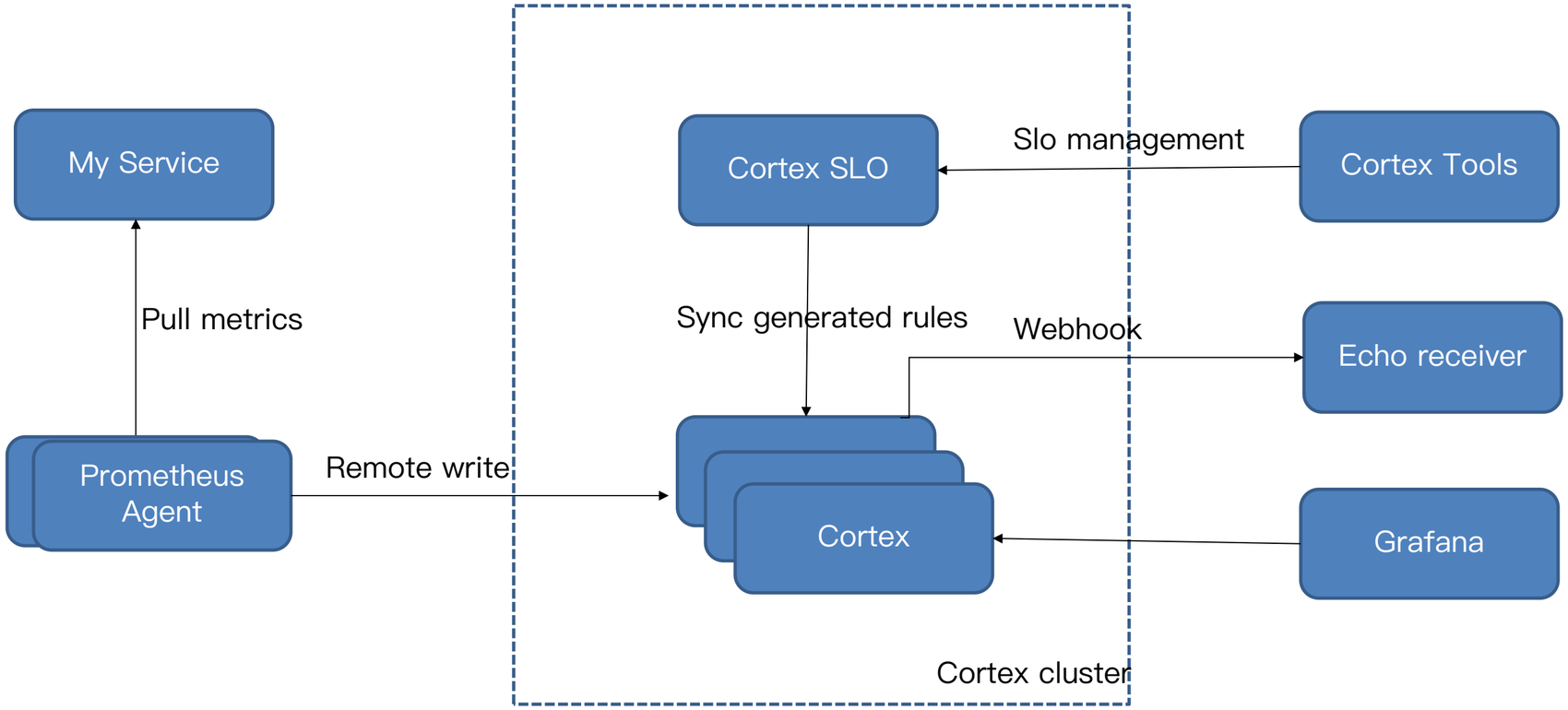
配置 cortex-slo 地址以及租户信息

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

#导入 和查询 windows

```
11:44 $ cortextool slos load-windows ./config/slos/windows/*
INFO[0000] 7d.yaml windows loaded
INFO[0000] google-30d.yaml windows loaded
✔ /home/service/workspace/tower/play-with-cortex-slo [main|✚ 2…5]
$ cortextool slos list-windows

INFO[0000] Windows:
7d
google-30d
✔ /home/service/workspace/tower/play-with-cortex-slo [main|✚ 2…5]
$ cortextool slos load ./config/slos/*.yml --windows google-30d
INFO[0000] myservice.yml slos loaded with google-30d alert windows
✔ /home/service/workspace/tower/play-with-cortex-slo [main|✚ 2…5]
$ cortextool slos list

INFO[0000] Slos:
myservice
✔ /home/service/workspace/tower/play-with-cortex-slo [main|✚ 2…5]
$ cortextool slos get myservice

INFO[0000] myservice Slos:

version: "prometheus/v1"
service: "myservice"
labels:
  owner: "myteam"
  repo: "myorg/myservice"
slos:
```

Demo with docker-compose



参考 GitHub 仓库 https://github.com/grafanafans/play-with-cortex-slo

资料

- Google SRE workbook https://sre.google/workbook/alerting-on-slos
- Sloth GitHub https://github.com/slok/sloth
- Pyrra GitHub https://github.com/pyrra-dev/pyrra
- Play with sloth https://github.com/grafanafans/play-with-sloth
- Play with pyrra https://github.com/grafanafans/play-with-pyrra

云原生社区
Cloud Native Community

OBSERVABILITY
SUMMIT 2023
可观测性峰会 第 1 届

Thank you

关注我们获取更多云原生资讯