



OBSERVABILITY
SUMMIT 2023

可观测性峰会 第1届

基于 Prometheus 的 SLO 告警实战

宋佳洋 < KLLK



大纲

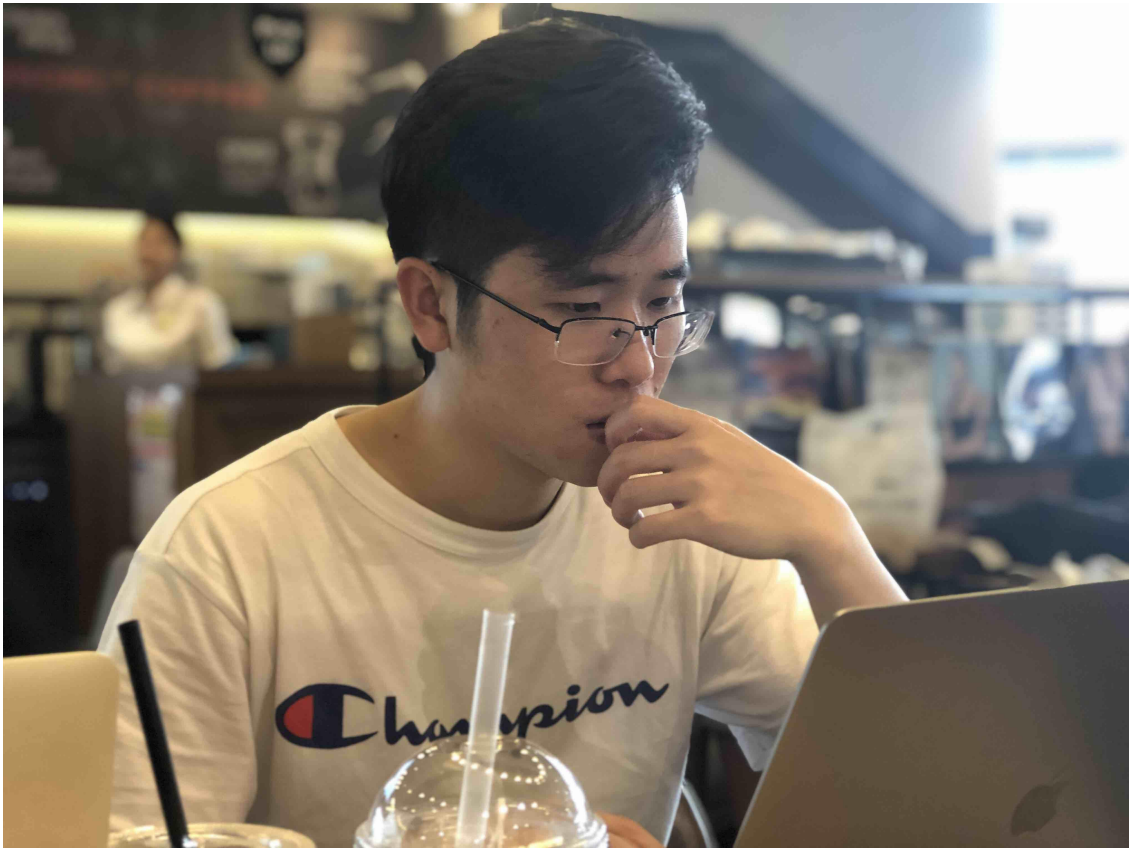
告警基础知识

哨基于! 项目构建

哨基于! 项目构建

哨多租户! 服务构建

关于我



先后就职于七牛云、京东云等公司，目前在! E π π E!从事云计算相关工作。



爱好开源，目前主要关注! ± 又和!云原生可观测领域、是开源项目!
π包《《豪《内卷 Vm取乾 → 改《的码贡献值。"



基《册《



微信公众号："π 邪《《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 $\omega\epsilon E$, 就没有! $\omega\pi f$!

SLO 相关概念

SLI

- 状态码!错误码
- 请求延迟!响应时间
- 进程运行非!异常状态码退出

时间窗口

- 小时
- 天
- 周

告警级别

- 严重
- 警告

错误预算

- 时间周期: 天
- 错误预算: 允许的失败次数
- 每天总请求数: 总请求数
- 允许的失败请求数: 允许的失败请求数

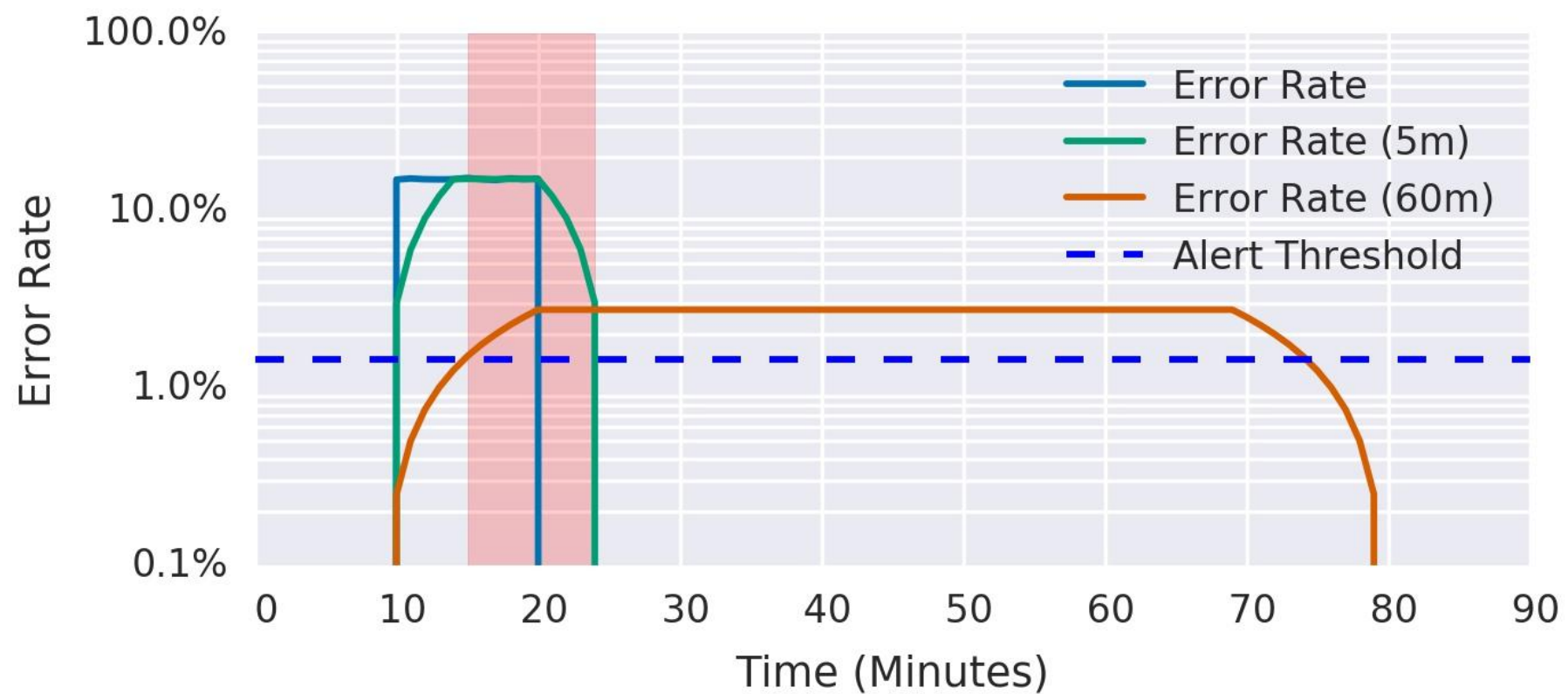
燃烧率

燃烧率	100%错误预算燃烧时间
	天
	天
	小时
	分钟

SLO 告警指导思想-MWMMR

- **יָבִיב, יָבִיב:** 准确率
- **×ב, פא:** 召回率（故障漏过未告警）
- **\בב, יָבִיב:** 投递延迟
- **×בב, יָבִיב:** 告警重置时长

Severity	Long window	Short window	Burn rate	Error budget consumed

[illegible]

基于 Prometheus SLO 告警基础和挑战

开箱即用的 record and alert rule

嗟色又入
 四色又可殺又も發又くも豪の地を轟轟海の内に甚勝なりと内く
 !!!も落腹中くと標又くも雄壯又くも豪の地を轟轟海の内に甚勝なりと戦

唸ハ瀧
 凧ハ瀧カヲトシテ瀧ノ俤
 !!も瀧ハ瀧カヲトシテ瀧ノ俤ノ又ハ瀧ノ徑ハ瀧ノ俤ノ又ハ瀧ノ俤ノ又ハ瀧ノ俤ノ
 !!ハ瀧ノ俤
 !!!!ハ瀧ノ俤ノ又ハ瀧ノ俤ノ又ハ瀧ノ俤ノ又ハ瀧ノ俤ノ又ハ瀧ノ俤ノ又ハ瀧ノ俤ノ又ハ瀧ノ俤ノ

加载和热更新

- [illegible]

与时间窗口相关的多个 SLI rules

- [illegible]

Alert rule 复杂, 需要考虑不同时间窗口和告警级别

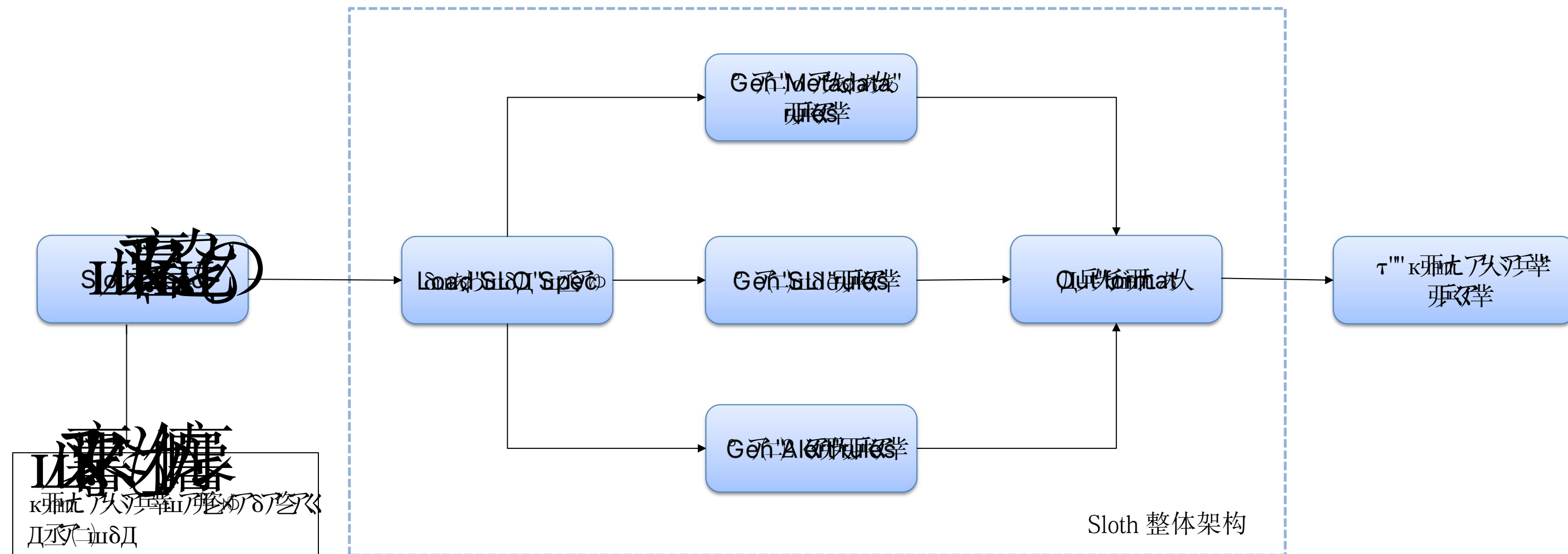
[illegible]

Prometheus 告警基础

Prometheus SLO 告警挑战

开源项目 sloth 简介

[Sloth](#) 是一个简单易用的! 告警! 生成器、支持! 命令行和! 两种使用方式, 支持自定义告警窗口配置、提供开箱即用的! 看板。



Sloth SLO 配置

1. 配置

```
version: "prometheus/v1"
service: "myservice"
labels:
  owner: "myteam"
  repo: "myorg/myervice"
  tier: "2"
slos:
  # We allow failing (5xx and 429) 1 request every 100
  - name: "requests-availability"
    objective: 99.9
    description: "Common SLO based on availability"
    sli:
      events:
        error_query: sum(rate(http_request_duration_seconds_5xx{service="myservice"}))
        total_query: sum(rate(http_request_duration_seconds{service="myservice"}))
    alerting:
      name: MyServiceHighErrorRate
      labels:
        category: "availability"
      annotations:
        # Overwrite default Sloth SLO alert summary
        summary: "High error rate on 'myservice' requests"
      page_alert:
        labels:
          severity: pageteam
          routing_key: myteam
      ticket_alert:
        labels:
          severity: "slack"
          slack_channel: "#alerts-myteam"
```

以!基值内来组织, 包含多个!基!基

公用标签

1个通用标签

一个告警级别告警标签

一个告警级别告警标签

X随基U π] !

```
apiVersion: sloth.slok.dev/v1
kind: PrometheusServiceLevel
metadata:
  name: sloth-slo-my-service
  namespace: monitoring
spec:
  service: "myservice"
  labels:
    owner: "myteam"
    repo: "myorg/myervice"
    tier: "2"
  slos:
    - name: "requests-availability"
      objective: 99.9
      description: "Common SLO based on availability"
      sli:
        events:
          errorQuery: sum(rate(http_request_duration_seconds_5xx{service="myservice"}))
          totalQuery: sum(rate(http_request_duration_seconds{service="myservice"}))
      alerting:
        name: MyServiceHighErrorRate
        labels:
          category: "availability"
        annotations:
          summary: "High error rate on 'myservice' requests"
        pageAlert:
          labels:
            severity: pageteam
            routing_key: myteam
        ticketAlert:
          labels:
            severity: "slack"
            slack_channel: "#alerts-myteam"
```

E 艺也! Ⅲ ε E

```
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_seconds{service="myservice"}))
        total:
          source: prometheus
          queryType: promql
          query: sum(rate(http_request_duration_seconds{service="myservice"}))
        target: 0.999
  timeWindows:
    - count: 30
      unit: Day
```

Sloth AlertWindows 配置

```
apiVersion: "sloth.slok.dev/v1"
kind: "AlertWindows"
```

```
spec:
```

```
sloPeriod: 30d
```

时间周期

```
page:
```

不同告警级别

```
quick:
```

```
errorBudgetPercent: 2
```

```
shortWindow: 5m
```

```
longWindow: 1h
```

不同错误预算燃烧率

```
slow:
```

```
errorBudgetPercent: 5
```

```
shortWindow: 30m
```

```
longWindow: 6h
```

多窗口

```
ticket:
```

```
quick:
```

```
errorBudgetPercent: 10
```

```
shortWindow: 2h
```

```
longWindow: 1d
```

```
slow:
```

```
errorBudgetPercent: 10
```

```
shortWindow: 6h
```

```
longWindow: 3d
```

```
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
```

1. 该平台主要用于告警配置，默认包含了两个配置，可自定义。

2. 该平台主要用于告警配置，默认包含了两个配置，可自定义。

Sloth CLI

生成SLI规则并生成Prometheus规则并生成Alert规则并生成Dashboard

```
INFO[0000] SLI plugins loaded
INFO[0000] Using custom slo period windows catalog
INFO[0000] SLO period windows loaded
INFO[0000] Generating from Prometheus spec
INFO[0000] Multiwindow-multiburn alerts generated
s.Service version=v0.11.0 window=30d
INFO[0000] SLI recording rules generated
rometheus.Service version=v0.11.0 window=30d
INFO[0000] Metadata recording rules generated
rometheus.Service version=v0.11.0 window=30d
INFO[0000] SLO alert rules generated
rometheus.Service version=v0.11.0 window=30d
INFO[0000] Prometheus rules written
window=30d

plugins=0 svc=storage.FileSLIPlugin version=v0.11.0 window=30d
svc=alert.WindowsRepo version=v0.11.0 window=30d
svc=alert.WindowsRepo version=v0.11.0 window=30d windows=2
version=v0.11.0 window=30d
out=rules slo=myservice-requests-availability svc=generate.prometheu
out=rules rules=8 slo=myservice-requests-availability svc=generate.p
out=rules rules=7 slo=myservice-requests-availability svc=generate.p
out=rules rules=2 slo=myservice-requests-availability svc=generate.p
format=yaml groups=3 out=rules svc=storage.IOWriter version=v0.11.0
```

生成SLI规则并生成Prometheus规则并生成Alert规则并生成Dashboard

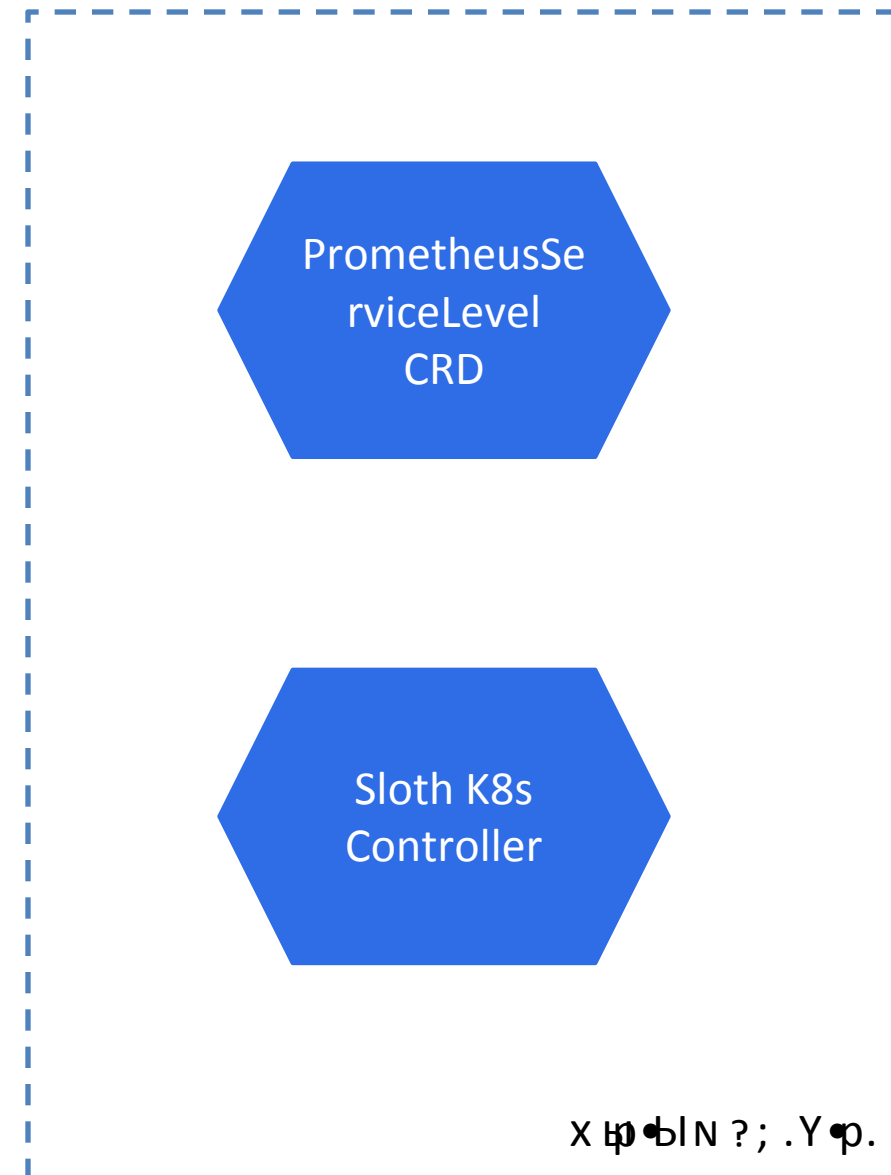
```
INFO[0000] SLI plugins loaded
INFO[0000] Using custom slo period windows catalog
INFO[0000] SLO period windows loaded
error: "generate" command failed: invalid default slo period: window period 672h0m0s missing
```

- 支持单个文件和目录批量生成。
- 会覆盖默认! 配置
- 对应! 不存在，会报错

Sloth 与 K8s



Prometheus Operator



Ⓔ 部署! ~~基~~ ~~富~~ ~~炮~~ ~~炮~~ ~~施~~

又ハ内モ裏」芝偶野ノ臺モ廻御共豪内ハ其色又寫書又ハ囁又囁又豪隙」ヲ
 又ハ關内モ色モ臺基欄」欄ハ囁又豪甚又堪ハ輝包又ハ豪」彼甚使カ」東」甚聞クル

や内も裏」芝罘 ス臺基龜御其臺内付其色又寫書又く囁又囁又豪閑」上欄
へ台又開闢内も色も臺基龜御囁又豪閑 くる

啾部署!基~~又~~富ЩεE

又「他も裏」芝偶の臺本に假芝臺の假を又得書又囃又囃又臺に上欄
 も齊くちに書欄基に臺の臺に臺へ撰くル

点击查看生成的!基和!基

やれも^の裏^にく又^は裏^に行^き書^け叔^が
やれも^の裏^にく又^は裏^に行^き書^け包^をくも豪^に彼^を袖^に基

生成 Prometheus rules 详解

sli rules

- 每个规则都基于
- 包含目标、错误预算、时间周期等

State	Name
Recording rule	slo:sli_error:ratio_rate5m
See graph	
Labels	
owner=myteam repo=myorg/myservice sloth_id=myservice-requests	
Expression	
$\frac{\sum(\text{rate}(\text{http_request_duration_seconds_count}\{\text{code}=\sim\})}{\sum(\text{rate}(\text{http_request_duration_seconds_count}\{\text{job}=\sim\})}$	
Recording rule	slo:sli_error:ratio_rate30m
Recording rule	slo:sli_error:ratio_rate1h
Recording rule	slo:sli_error:ratio_rate2h
Recording rule	slo:sli_error:ratio_rate6h
Recording rule	slo:sli_error:ratio_rate1d
Recording rule	slo:sli_error:ratio_rate3d
Recording rule	slo:sli_error:ratio_rate30d

metadata rules

- 每个规则都基于
- 包含了目标、错误预算、时间周期等

State	Name
Recording rule	slo:objective:ratio
See graph	
Labels	
owner=myteam repo=myorg/myservice sloth_id=	
Expression	
$\text{vector}(0.9990000000000001)$	
Recording rule	slo:error_budget:ratio
Recording rule	slo:time_period:days
Recording rule	slo:current_burn_rate:ratio
Recording rule	slo:period_burn_rate:ratio
Recording rule	slo:period_error_budget_remaining:ratio
Recording rule	sloth_slo_info

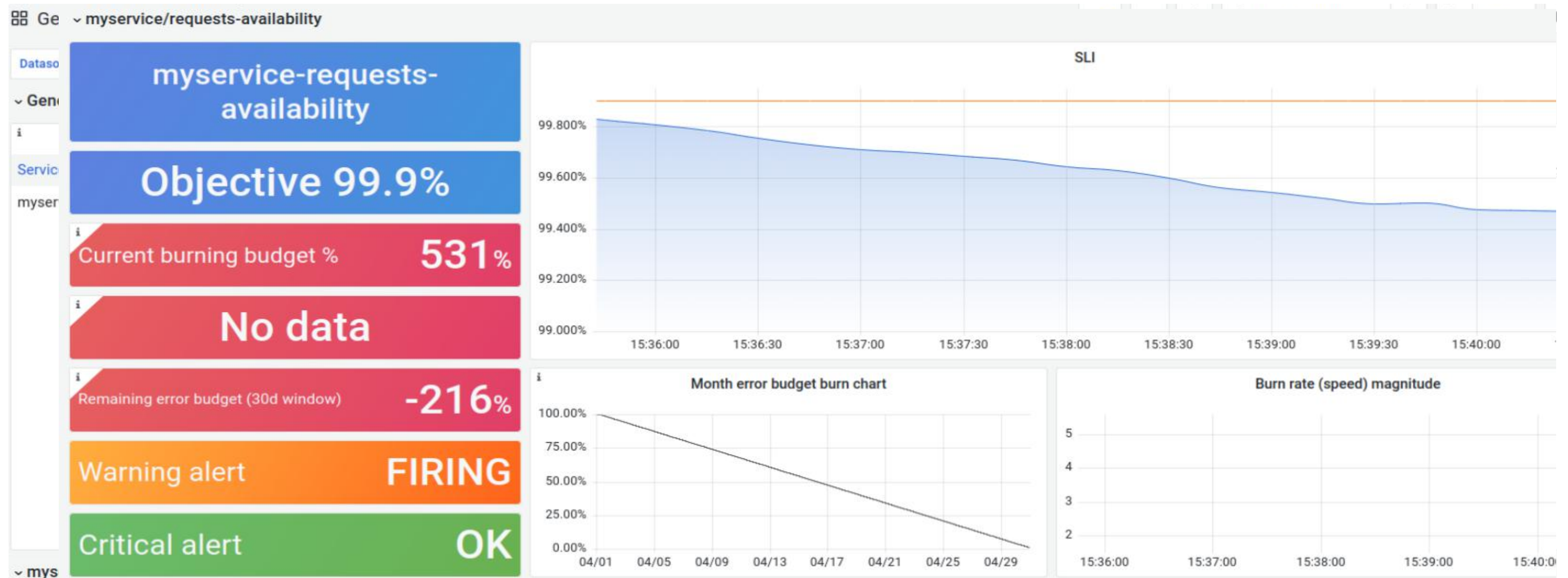
alert rules

- 每个规则都基于
- 支持

State	Name	Health	Summary
Normal	MyServiceHighErrorRate	ok	High error rate on 'myservice' requests resp
Normal	MyServiceHighErrorRate	ok	High error rate on 'myservice' requests resp
See graph			
Labels			
category=availability severity=slack slack_channel=#alerts-myservice sloth_severity=ticket			
Expression			
$\frac{\max(\text{without}(\text{sloth_window}) \{ \text{slo:sli_error:ratio_rate2h}\{\text{sloth_id}=\text{'myservice-requests-availability'}, \text{sloth_service}=\text{'myservice'}, \text{sloth_slo}=\text{'requests-availability'}\} \} * 0.8000000000000001) \text{ and } \max(\text{without}(\text{sloth_window}) \{ \text{slo:sli_error:ratio_rate1d}\{\text{sloth_id}=\text{'myservice-requests-availability'}, \text{sloth_service}=\text{'myservice'}, \text{sloth_slo}=\text{'requests-availability'}\} \} * 0.8000000000000001) \text{ or } (\max(\text{without}(\text{sloth_window}) \{ \text{slo:sli_error:ratio_rate6h}\{\text{sloth_id}=\text{'myservice-requests-availability'}, \text{sloth_service}=\text{'myservice'}, \text{sloth_slo}=\text{'requests-availability'}\} \} * 0.8000000000000001) \text{ and } \max(\text{without}(\text{sloth_window}) \{ \text{slo:sli_error:ratio_rate3d}\{\text{sloth_id}=\text{'myservice-requests-availability'}, \text{sloth_service}=\text{'myservice'}, \text{sloth_slo}=\text{'requests-availability'}\} \} * 0.8000000000000001))}$			
Summary			
High error rate on 'myservice' requests responses			
Title			
{ { \$labels.sloth_service } } { { \$labels.sloth_slo } } SLO error budget burn rate is too fast.			
Matching			
Search by label State			

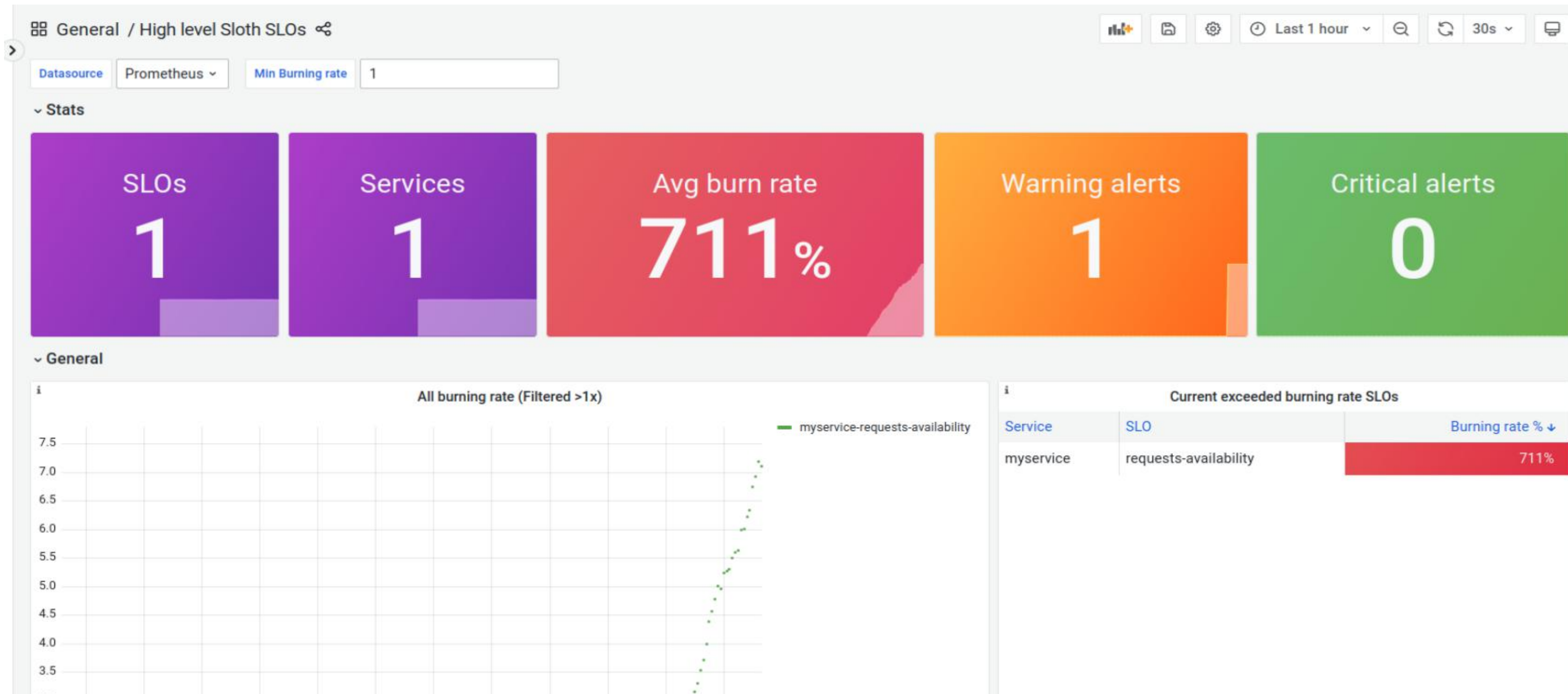
Sloth Dashboard-Details

~~故障!~~ "信保保"



Sloth Dashboard-High Level

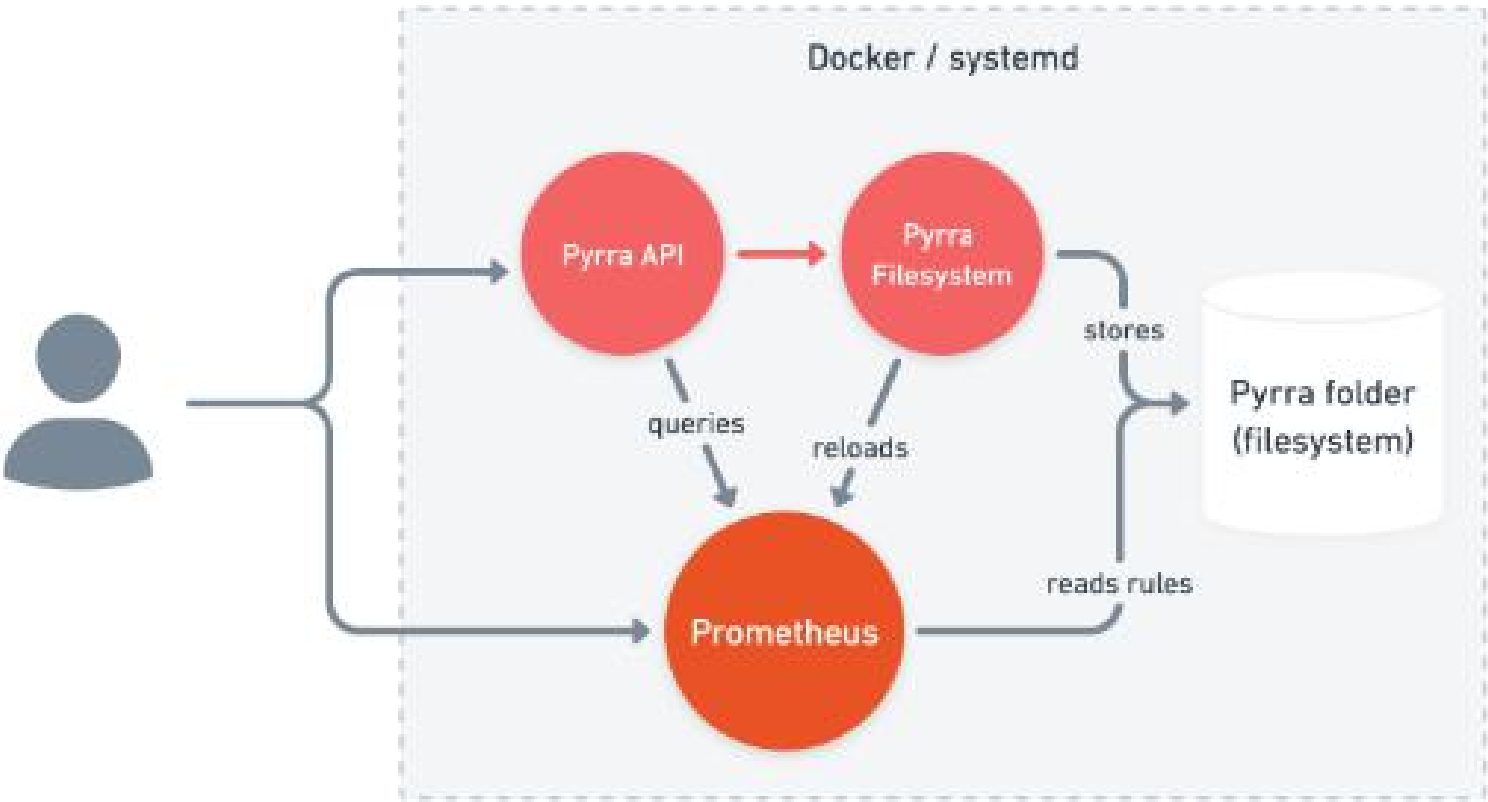
信保网



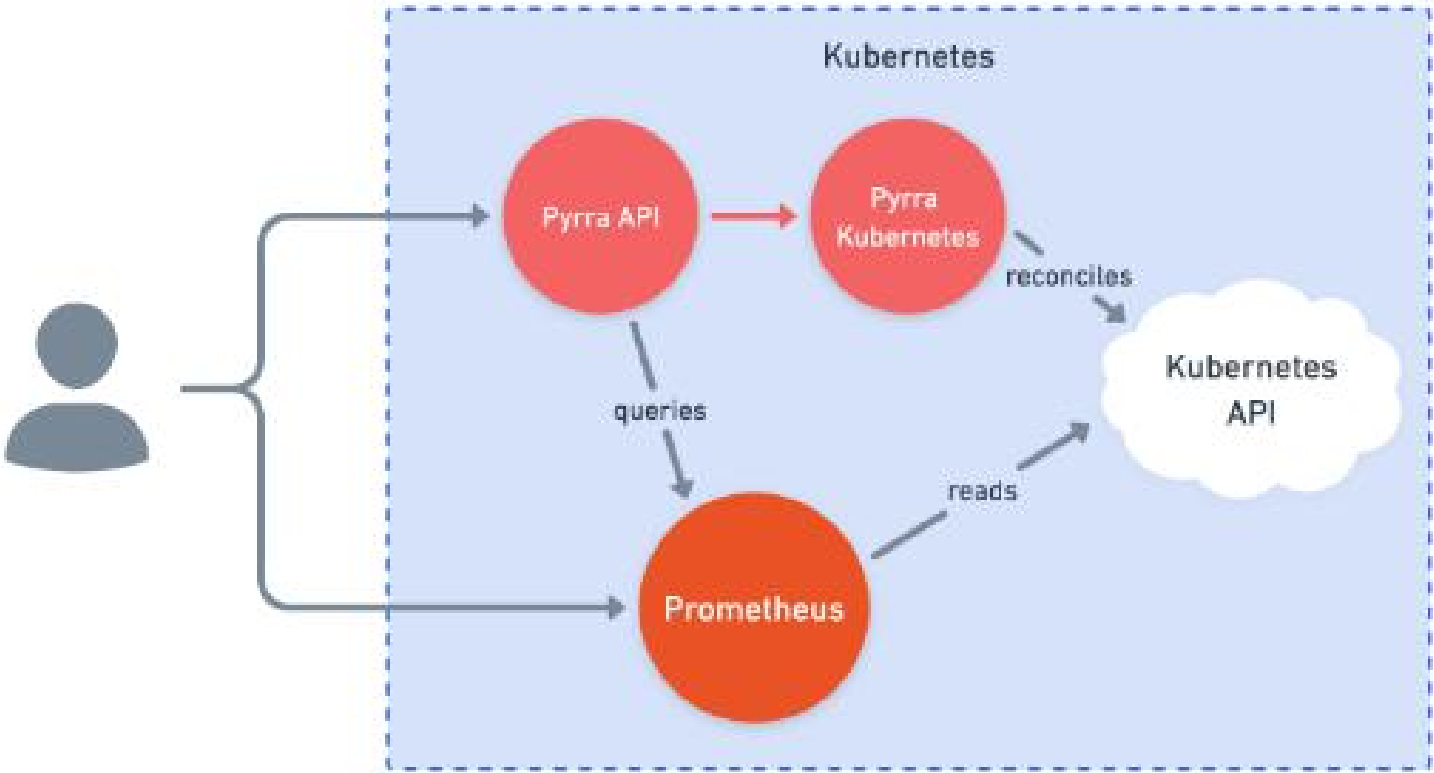
开源项目 Pyrra 介绍

简介

是另外一个的包，它基于 kubeE生成器、支持 kubeE和 kubeE两种模式，提供 UI界面和 CLI包作为可视化。



Pyrra 架构图



Pyrra 架构图

Pyrra SLO 配置

```
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
```

基和!」!基的基标签

目标值

时间窗口

分组

- 统一配置格式，无论是!X还是!カ!基
- 一个!基の定义一个!基
- !通过标签进行过滤!，缺少!基这层概念，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!部署! 基又基 命令

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

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

Flags:
  -h, --help                Show context-sensitive help.
  --config-files="/etc/pyrra/*.yaml"
                             The folder where the configuration files are
                             located.
  --prometheus-url=http://localhost:9090
                             The URL to the Prometheus instance.
  --prometheus-folder="/etc/prometheus/pyrra/"
                             The folder where the Prometheus metrics are
                             stored.
  --generic-rules            Enabled generic rules.
```

- 与! 包<< 豪 伪基 处于同一实例，注册加载目录配置，生成! 包<< 能够确保被! 包<< 豪 伪基 加载。
- 通过配置的! 包<< 豪 伪基 进行! 又 包<< 豪 伪基 重新加载配置。

Filesystem 模式

部署!

! 部署! 基又基 命令

! 部署! 基又基 命令

! 部署! 基又基 命令

! 部署! 基又基 命令

! 部署! 基又基 命令

! 部署! 基又基 命令

!!

! 部署! 基又基 命令

! 部署! 基又基 命令

! 部署! 基又基 命令

K8s 模式

Pyrra 生成 Prometheus rules 详解

单个!ⅢεE!包括三个!π包くも豪し内甚告警分组，分别为!儲確網俅九どナ譽 韓龍倭姿あぬし儀倭ア二アゆ "韓龍倭姿あぬし儀倭の所ア幸ア秋

韓龍倭姿あぬし儀倭 "包含!嚷个!ⅢεE!包の又!包し甚和!啗个!π→πⅡ!」も儀倭し甚

韓龍倭姿あぬし儀倭ア二アゆ: "俤个!包の又!包し甚 "主要用于看板的!スギし束し!汇总统计 ("og^ ")。

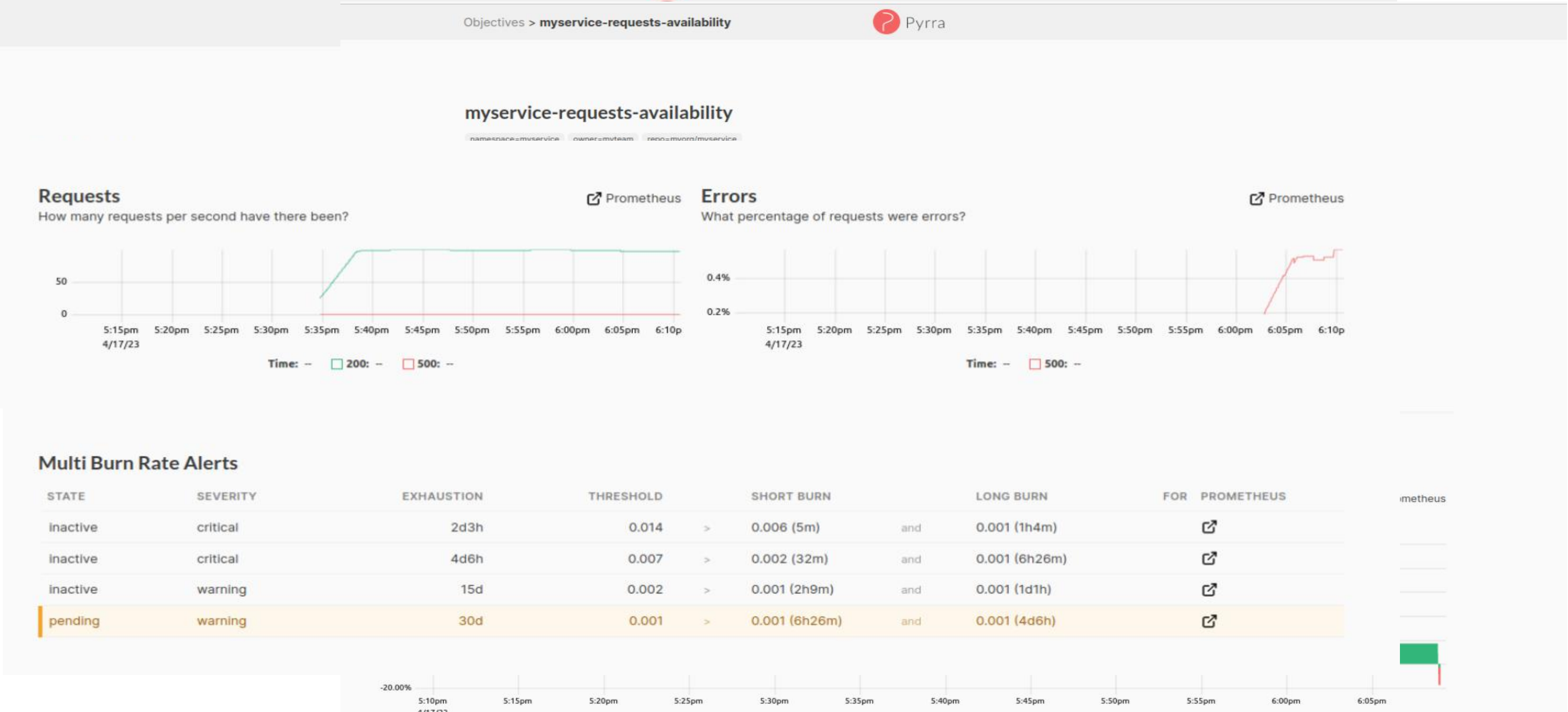
韓龍倭姿あぬし儀倭の所ア幸ア: "信个!包の又!包し和!啗个!」も儀倭し "主要用于统计整个窗口周期总请求数和!久へ」譽告警。

> 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

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

> Recording rule	http_request_duration_seconds:increase30d
> Normal	SLOMetricAbsent

Pyrra Dashboard-API



Multi Burn Rate Alerts

STATE	SEVERITY	EXHAUSTION	THRESHOLD		SHORT BURN		LONG BURN	FOR	PROMETHEUS
inactive	critical	2d3h	0.014	>	0.006 (5m)	and	0.001 (1h4m)		
inactive	critical	4d6h	0.007	>	0.002 (32m)	and	0.001 (6h26m)		
inactive	warning	15d	0.002	>	0.001 (2h9m)	and	0.001 (1d1h)		
pending	warning	30d	0.001	>	0.001 (6h26m)	and	0.001 (4d6h)		

-20.00%

5:10pm

5:15pm

5:20pm

5:25pm

5:30pm

5:35pm

5:40pm

5:45pm

5:50pm

5:55pm

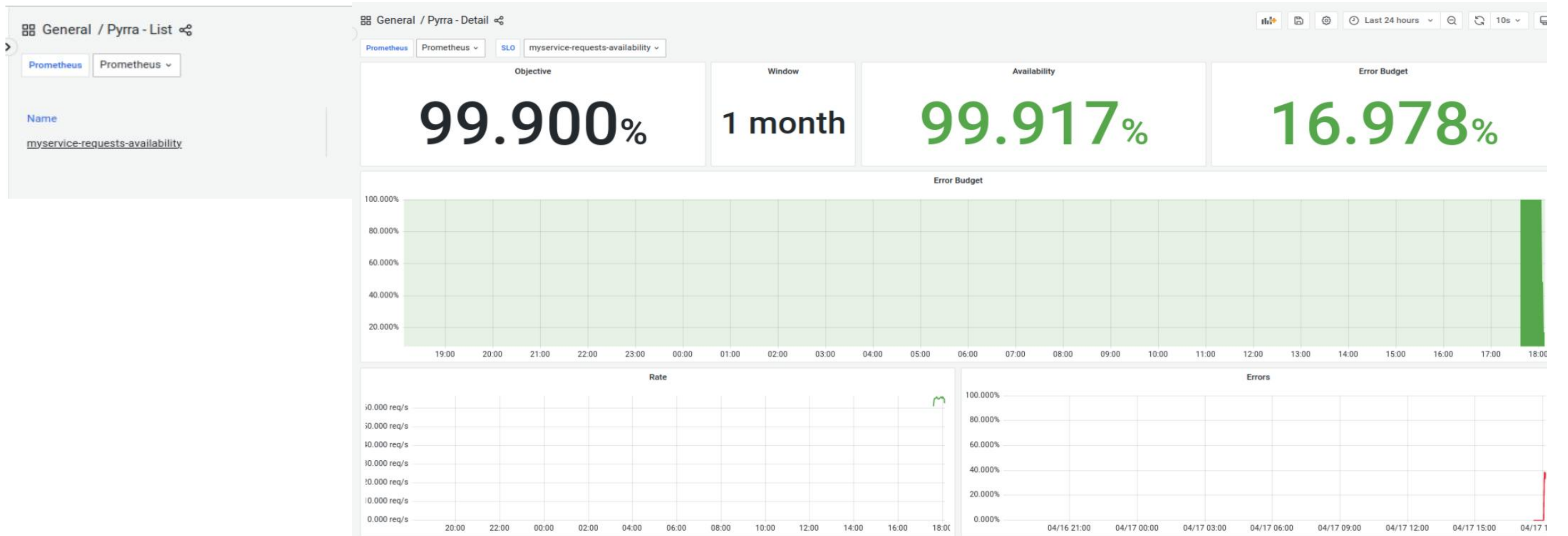
6:00pm

6:05pm

4/17/23

Pyrra Dashboard-Grafana

包拯

[illegible]

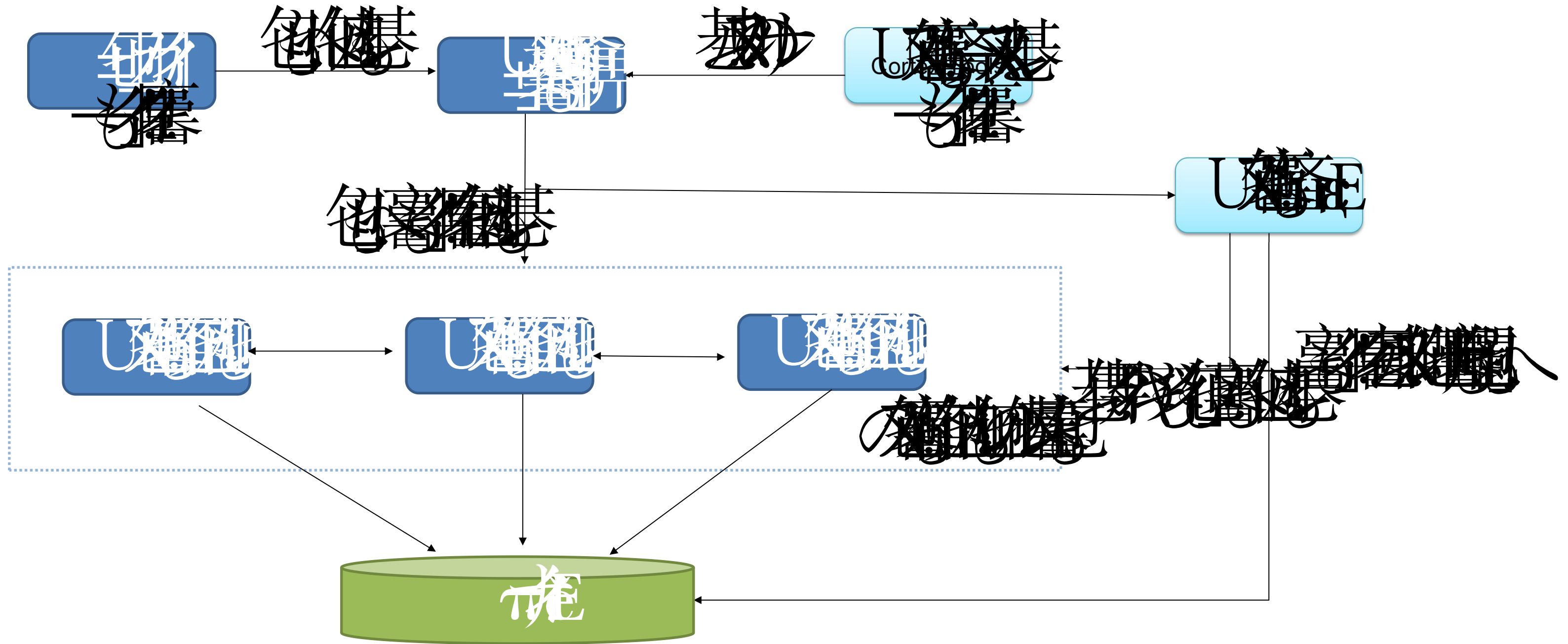
Sloth Vs Pyrra

	Sloth	Pyrra
	支持	支持
	不支持	支持
	支持	不支持
	支持	不支持
可读性	高	一般
		、k8s 2 kē

简单总结：

- 两个都是优秀的! 包々も喜ぶ! 基E! ギョル包喜也 幸い"开源时间较早，协议支持广泛，永登"属于后起之秀，有自己的!へ]基と又]包。
- 因为!基又喜"生成的!包々基"可读性更强，如果有二开需求并直接使用!±包カレ!作为看板，建议采用!基又喜。"

基于 Cortex 的多租户 SLO 服务构建



Cortex-tools 扩展

统一封装到!基!子命令

```
1 $ ./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!的!地址以及租户信息

配置!的!地址以及租户信息

导入!和查询!御!又!基

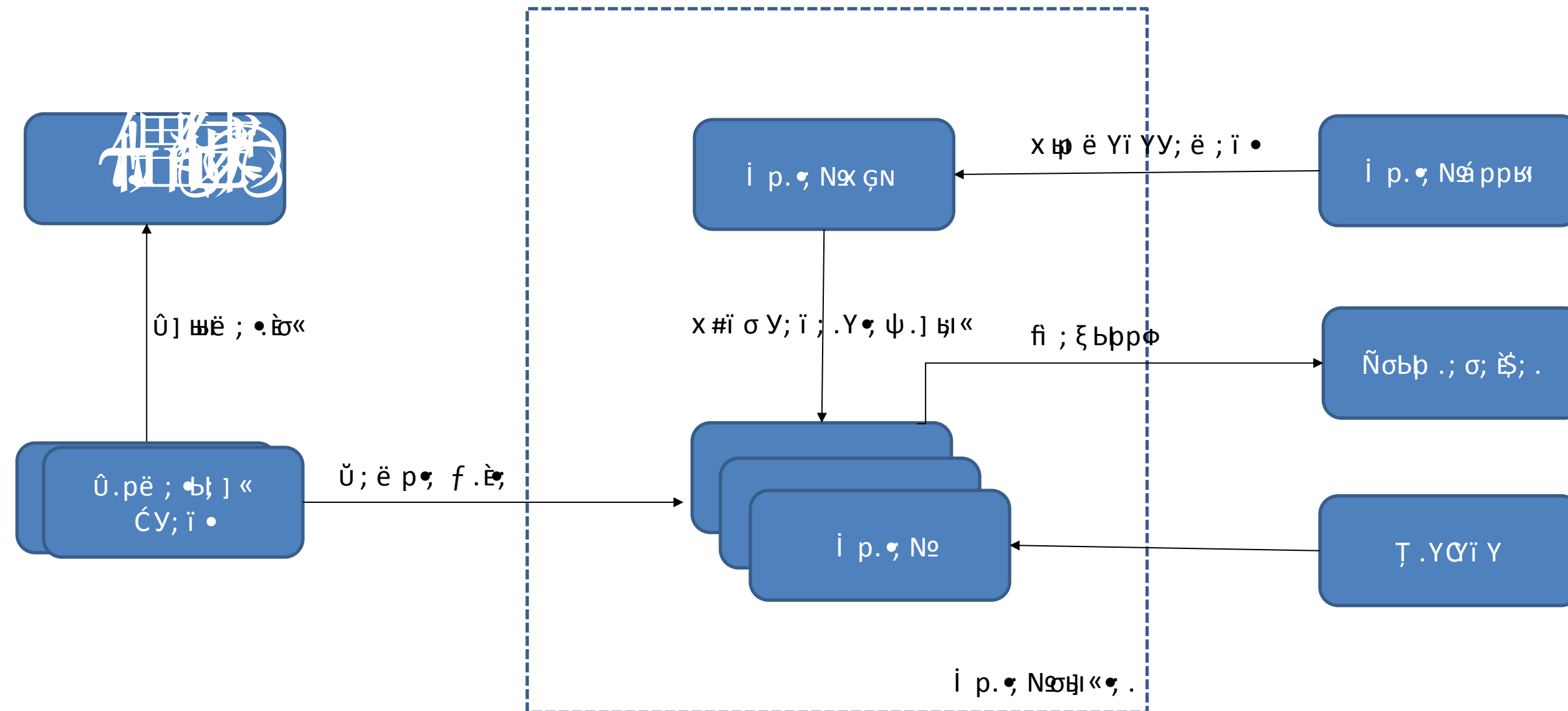
```
11:44 $ cortextool slos load-windows ./config/slos/windows/*
INFO[0000] 7d.yaml windows loaded
INFO[0000] google-30d.yaml windows loaded
✓ /home/service/workspace/tower/play-with-cortex-slo [main|+2...5]
1 $ cortextool slos list-windows

INFO[0000] Windows:
7d
google-30d
✓ /home/service/workspace/tower/play-with-cortex-slo [main|+2...5]
$ cortextool slos load ./config/slos/*.yaml --windows google-30d
INFO[0000] myservice.yaml slos loaded with google-30d alert windows
✓ /home/service/workspace/tower/play-with-cortex-slo [main|+2...5]
1 $ cortextool slos list

INFO[0000] Slos:
myservice
✓ /home/service/workspace/tower/play-with-cortex-slo [main|+2...5]
1.5 $ 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! ± 豪侏 仓库! フ 豪 侏 填 又 囃 包 カ レ カ レ 囃 也 何 豪 回 又 豪 侏 港 又

资料

- [📄 云原生社区 2023 年 1 月 1 日 更新](#)
- [📄 云原生社区 2023 年 1 月 1 日 更新](#)
- [📄 云原生社区 2023 年 1 月 1 日 更新](#)
- [📄 云原生社区 2023 年 1 月 1 日 更新](#)
- [📄 云原生社区 2023 年 1 月 1 日 更新](#)



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