SLO-Based Observability For All Kubernetes Cluster Components

KubeCon NA 2022

Nadine Vehling
Grafana Labs

Matthias Loibl
Polar Signals

Objectives

NAME ↑	TIME WINDOW	OBJECTIVE	LATENCY	AVAILABILITY	ERROR BUDGET	ALERTS
apiserver-read-cluster-latency component=apiserver namespace=kube-system	2w	99.00%	5s	100.00%	100.00%	
apiserver-read-namespace-latency component=apiserver namespace=kube-system	2w	99.00%	5s	100.00%	100.00%	
apiserver-read-resource-latency component=apiserver namespace=kube-system	2w	99.00%	100ms	99.99%	99.31%	
apiserver-read-response-errors component=apiserver namespace=kube-system	2w	99.00%		100.00%	100.00%	
apiserver-write-response-errors component=apiserver namespace=kube-system	2w	99.00%		100.00%	100.00%	
coredns-response-errors component=coredns namespace=kube-system	2w	99.99%		100.00%	100.00%	
coredns-response-latency component=coredns namespace=kube-system	2w	99.00%	32ms	99.97%	97.30%	
kubelet-request-errors component=kubelet namespace=kube-system	2w	99.00%		100.00%	100.00%	
kubelet-runtime-errors component=kubelet namespace=kube-system	2w	99.00%		100.00%	100.00%	
prometheus-notification-errors component=prometheus namespace=monitoring	2w	99.00%		100.00%	100.00%	
prometheus-operator-http-errors component=prometheus-operator namespace=monitoring	2w	99.50%		100.00%	100.00%	
prometheus-operator-reconcile-errors component=prometheus-operator namespace=monitoring	2w	95.00%		No data	No data	
prometheus-query-errors component=prometheus namespace=monitoring handler=/api/v1/query_range	2w	99.00%		100.00%	100.00%	
prometheus-query-errors component=prometheus namespace=monitoring handler=/api/v1/query	2w	99.00%		100.00%	100.00%	
prometheus-rule-evaluation-failures component=prometheus namespace=monitoring	2w	99.99%		100.00%	100.00%	
prometheus-sd-kubernetes-errors component=prometheus namespace=monitoring	2w	99.00%		100.00%	100.00%	

All availabilities and error budgets are calculated across the entire time window of the objective.

Overview

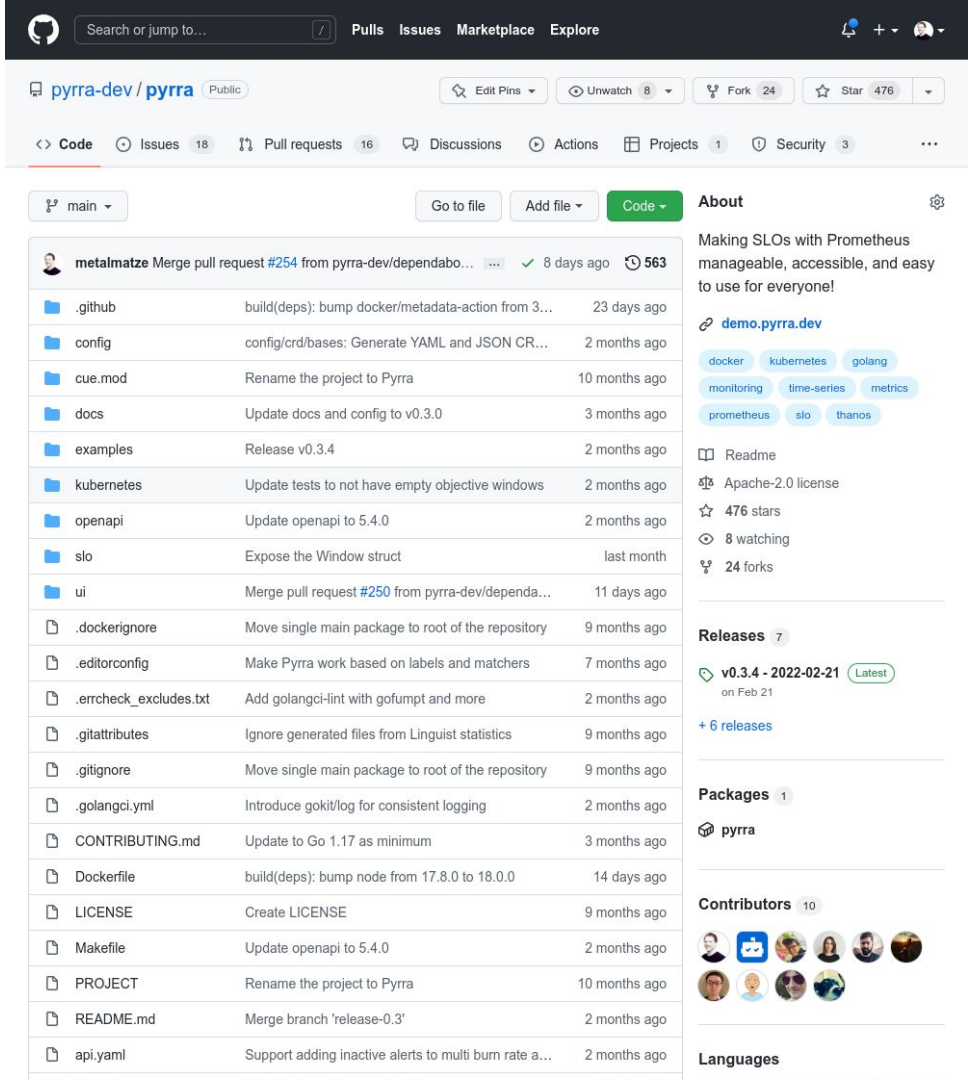

Fundamentals of SLOs
Pyrra and our approach
How to create an SLO?
Kubernetes Cluster
Demo
What's next?

What are SLOs?

Fundamentals

99% of requests succeed

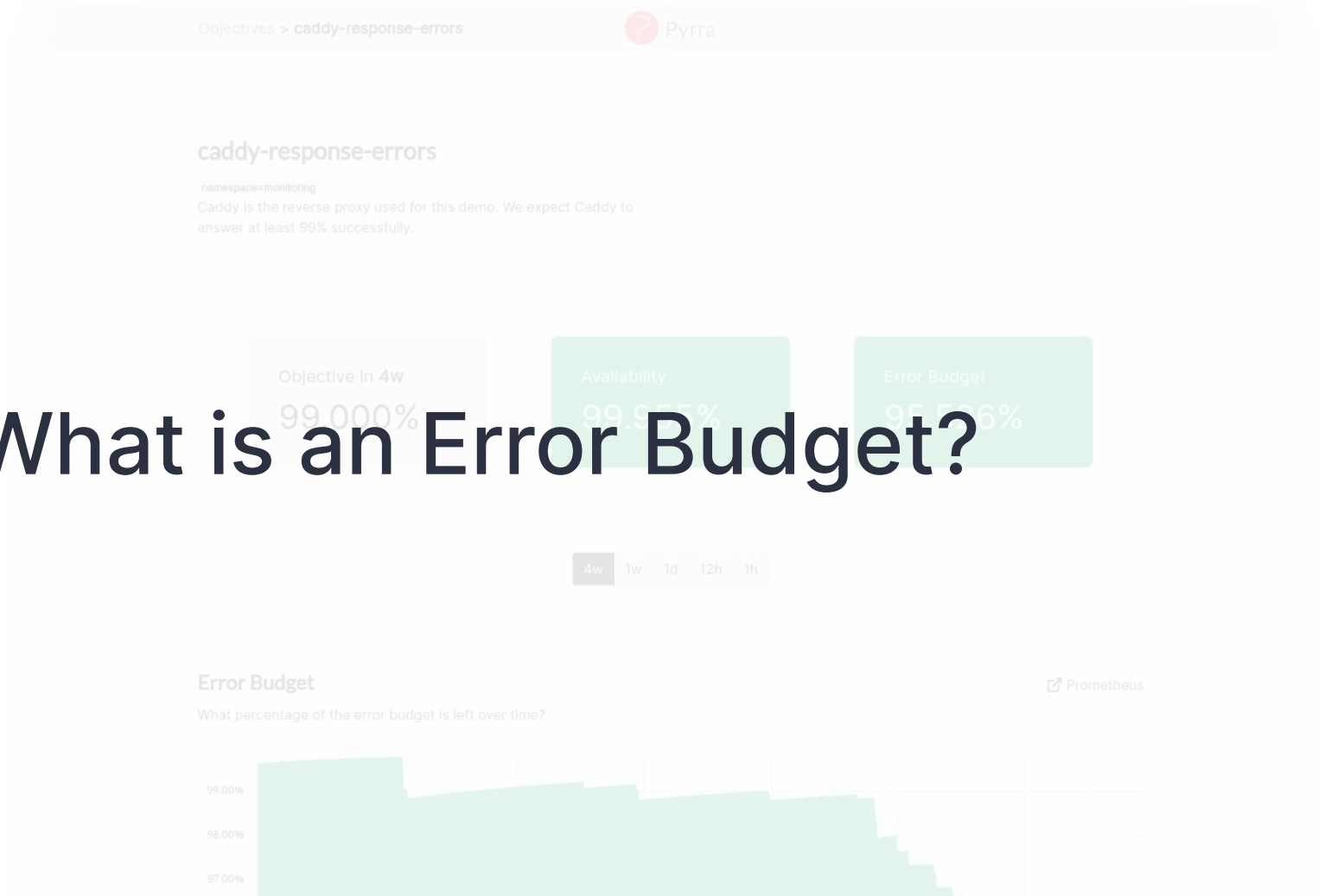
within 5s



What are SLIs?

What is Availability?

What is an Error Budget?



Error Budget

		Objective		Error Budget
100%	-	90%	=	10%
100%	-	95%	=	5%
100%	-	99%	=	1%
100%	-	99.9%	=	0.1%

Error Budget

(Availability	-	Objective)	/	Error Budget	=	Available Error Budget
100%		95%		5%		100%
95%		95%		5%		0%
98%		95%		5%		60%
93%		95%		5%		-40%

What is Pyrra?

Our approach and research

Pyrra

Making SLOs with Prometheus manageable, accessible, and easy to use for everyone!

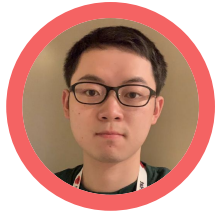
Maintainers & Contributors











How did we start?

Project approach and research

Interviews and testing

Technical audience

Software Engineers
Site Reliability Engineers

Environment and experience level

Smaller and larger companies
Beginner to advanced

Interviews and testing

Interviews and testing

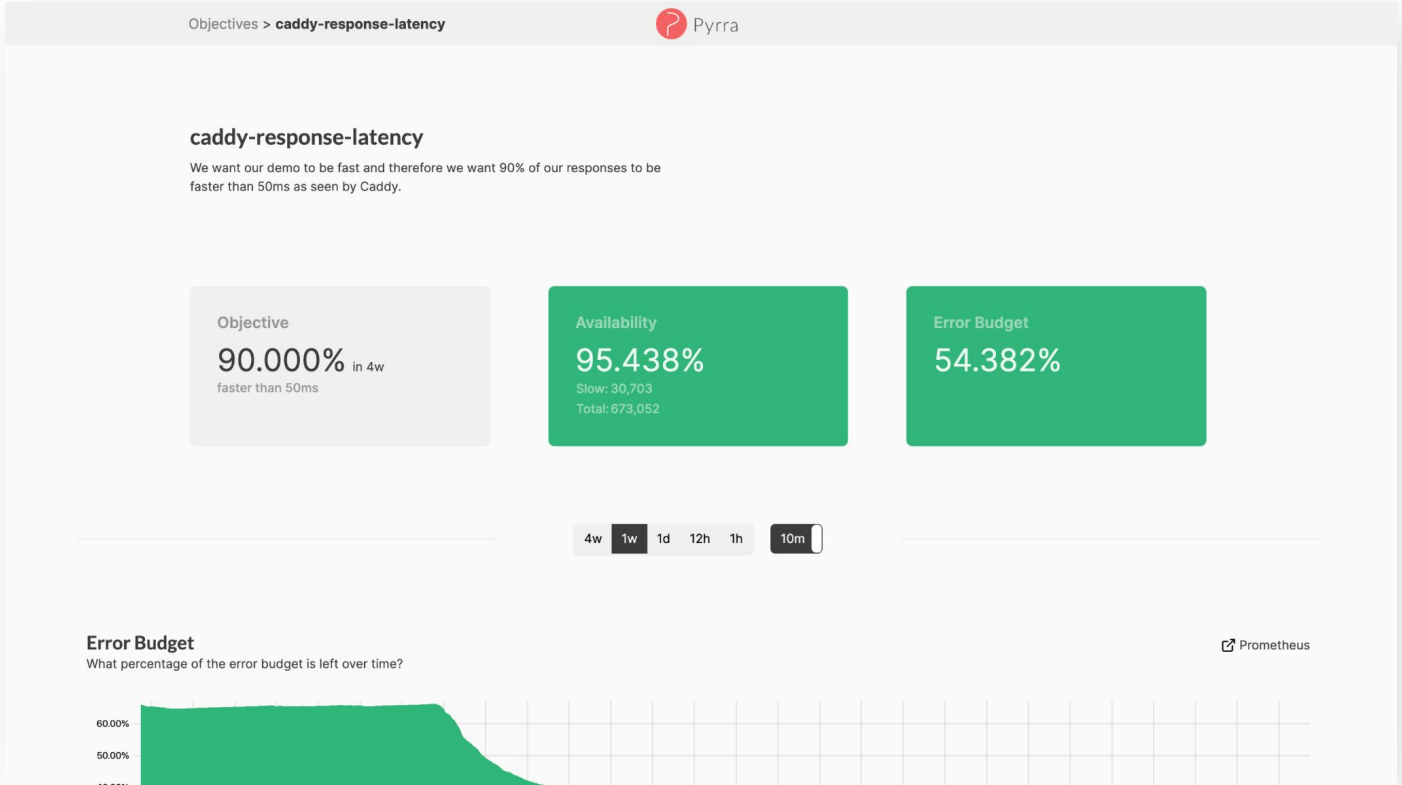
How might we reduce information overload?

How might we show relevant and actionable data?

How might we showcase multi burn rate alerts best?

How might we link to explore queries in Prometheus itself?

Pyrra detail

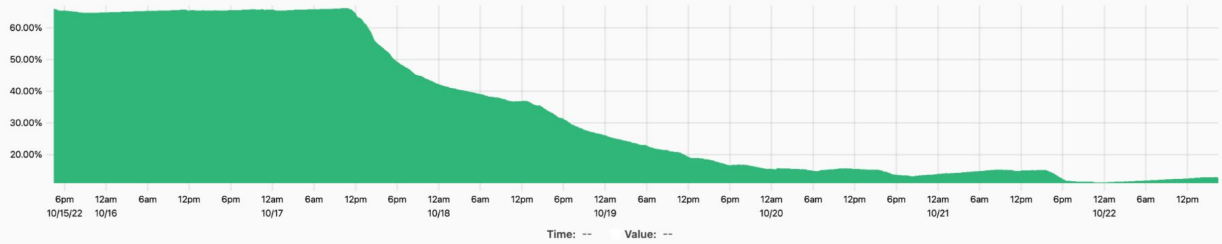


4w 1w 1d 12h 1h 10m

Error Budget

Prometheus

What percentage of the error budget is left over time?



Requests

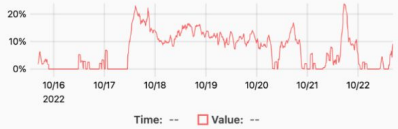
Prometheus

How many requests per second have there been?

Errors

Prometheus

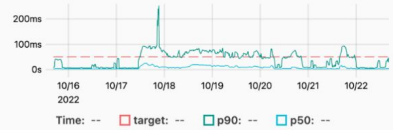
What percentage of requests were too slow?



Duration

Prometheus

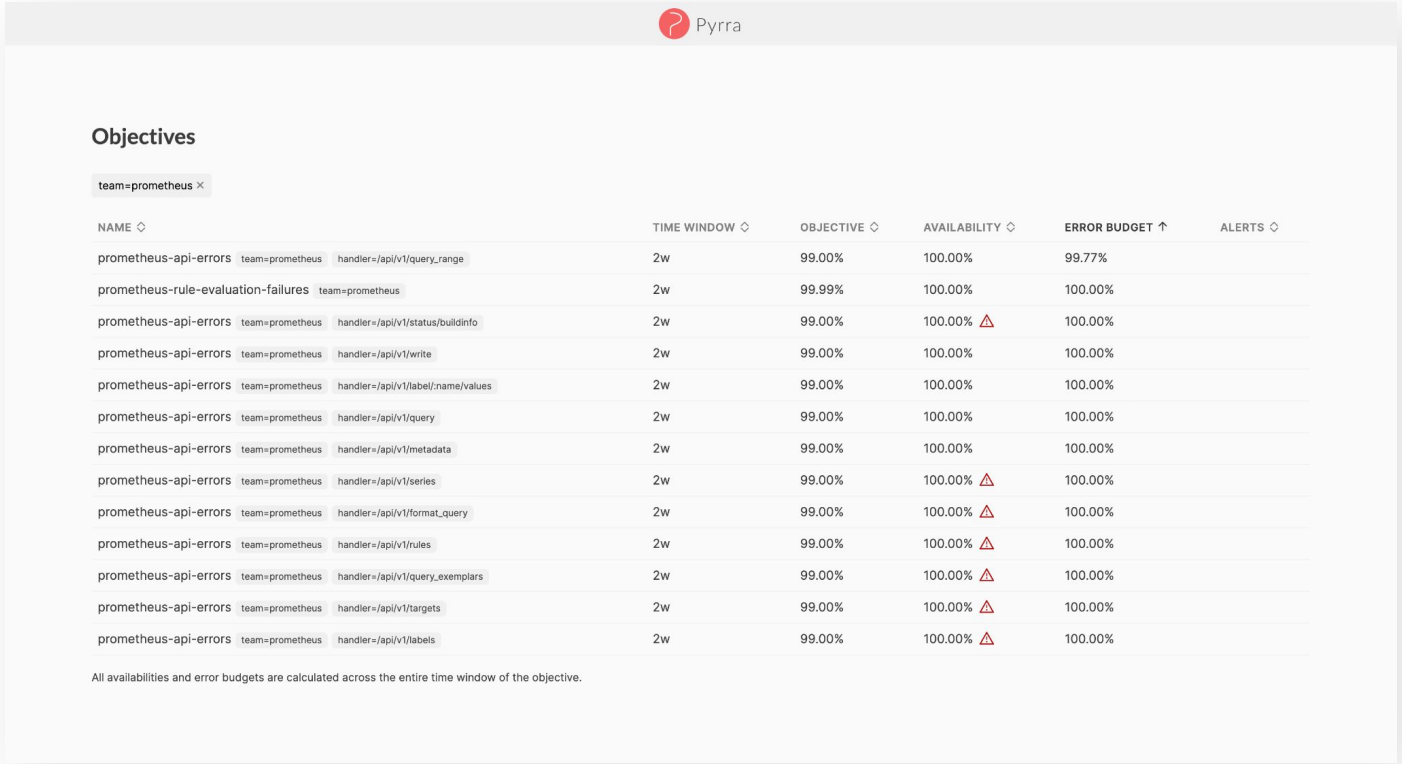
How long do the requests take?
p90 must be faster than 50ms.



Multi Burn Rate Alerts

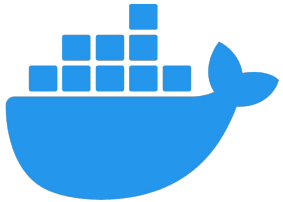
STATE	SEVERITY	EXHAUSTION	THRESHOLD		SHORT BURN		LONG BURN	FOR	PROMETHEUS
inactive	critical	2d	1.400	>	0.173 (5m)	and	0.092 (1h)		
inactive	critical	4d	0.700	>	0.122 (30m)	and	0.040 (6h)		
inactive	warning	2w	0.200	>	0.074 (2h)	and	0.074 (1d)		
inactive	warning	4w	0.100	>	0.040 (6h)	and	0.094 (4d)		

Pyrra overview



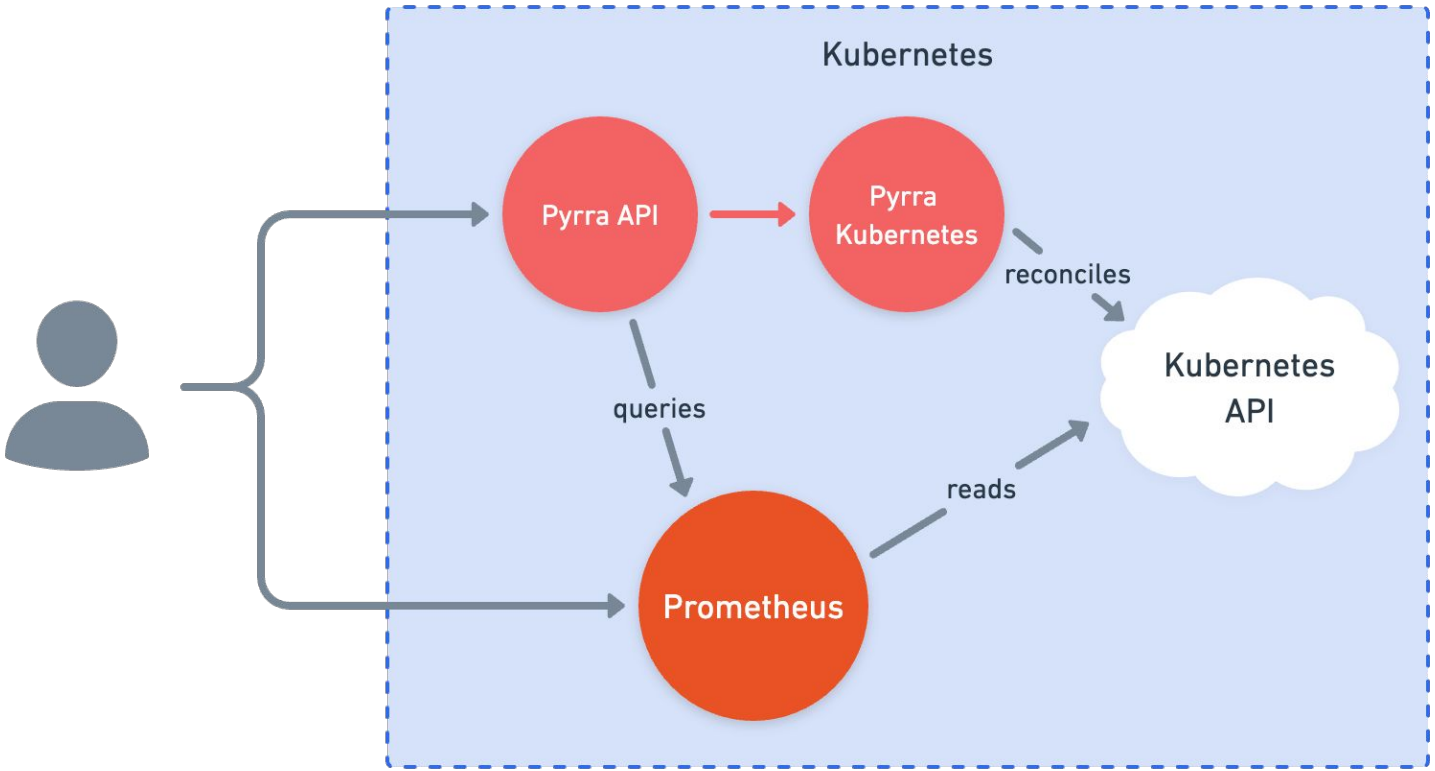
Deployments

kubernetes



docker

Deployments



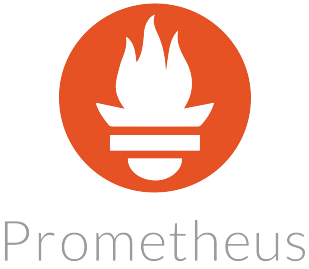
Deployments - API

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pyrra-api
  namespace: monitoring
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: pyrra
        args:
        - api
        - --api-url=http://pyrra-kubernetes.monitoring.svc.cluster.local:9444
        - --prometheus-url=http://prometheus-k8s.monitoring.svc.cluster.local:9090
        image: ghcr.io/pyrra-dev/pyrra:v0.5.0
        ports:
        - containerPort: 9099
```

Deployments - Kubernetes Backend

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pyrra-kubernetes
  namespace: monitoring
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: pyrra
        args:
        - kubernetes
        - --generic-rules
        image: ghcr.io/pyrra-dev/pyrra:main
        ports:
        - containerPort: 9099
```

Prometheus-like storages



How to create an SLO?

Looking at the configuration

Configuring SLOs

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: prometheus-operator-errors
  namespace: monitoring
spec:
  target: "99.99"
  window: 14d
  indicator:
    ratio:
      errors:
        metric: prometheus_operator_reconcile_errors_total
      total:
        metric: prometheus_operator_reconcile_operations_total
```

Configuring SLOs

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: prometheus-operator-errors
  namespace: monitoring
spec:
  target: "99.99"
  window: 14d
  indicator:
    ratio:
      errors:
        metric: prometheus_http_requests_total{code=~"5.."}
      total:
        metric: prometheus_http_requests_total
```

Configuring SLOs

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: prometheus-operator-errors
  namespace: monitoring
spec:
  target: "99.99"
  window: 14d
  indicator:
    ratio:
      errors:
        metric: prometheus_http_requests_total{handler="/api/query",code=~"5.."}
      total:
        metric: prometheus_http_requests_total{handler="/api/query"}
```

Configuring SLOs

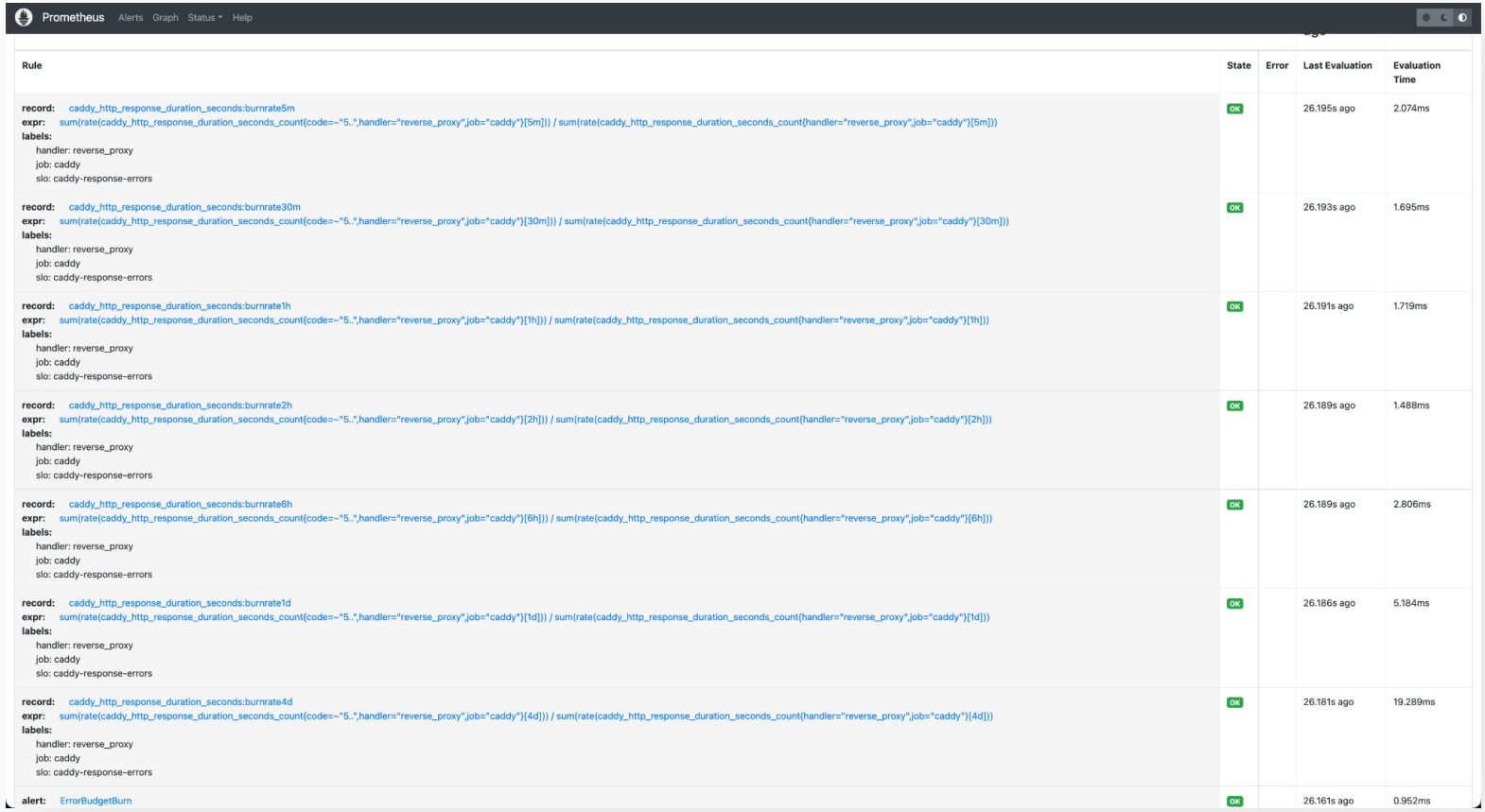
```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: prometheus-operator-errors
  namespace: monitoring
spec:
  target: "99.99"
  window: 14d
  indicator:
    ratio:
      errors:
        metric: prometheus_http_requests_total{handler=~"/api.*",code=~"5.."}
      total:
        metric: prometheus_http_requests_total{handler=~"/api.*"}
```

Configuring SLOs

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: prometheus-operator-errors
  namespace: monitoring
spec:
  target: "99.99"
  window: 14d
  indicator:
    ratio:
      errors:
        metric: prometheus_http_requests_total{handler=~"/api.*",code=~"5.."}
      total:
        metric: prometheus_http_requests_total{handler=~"/api.*"}
      grouping:
        - handler
```

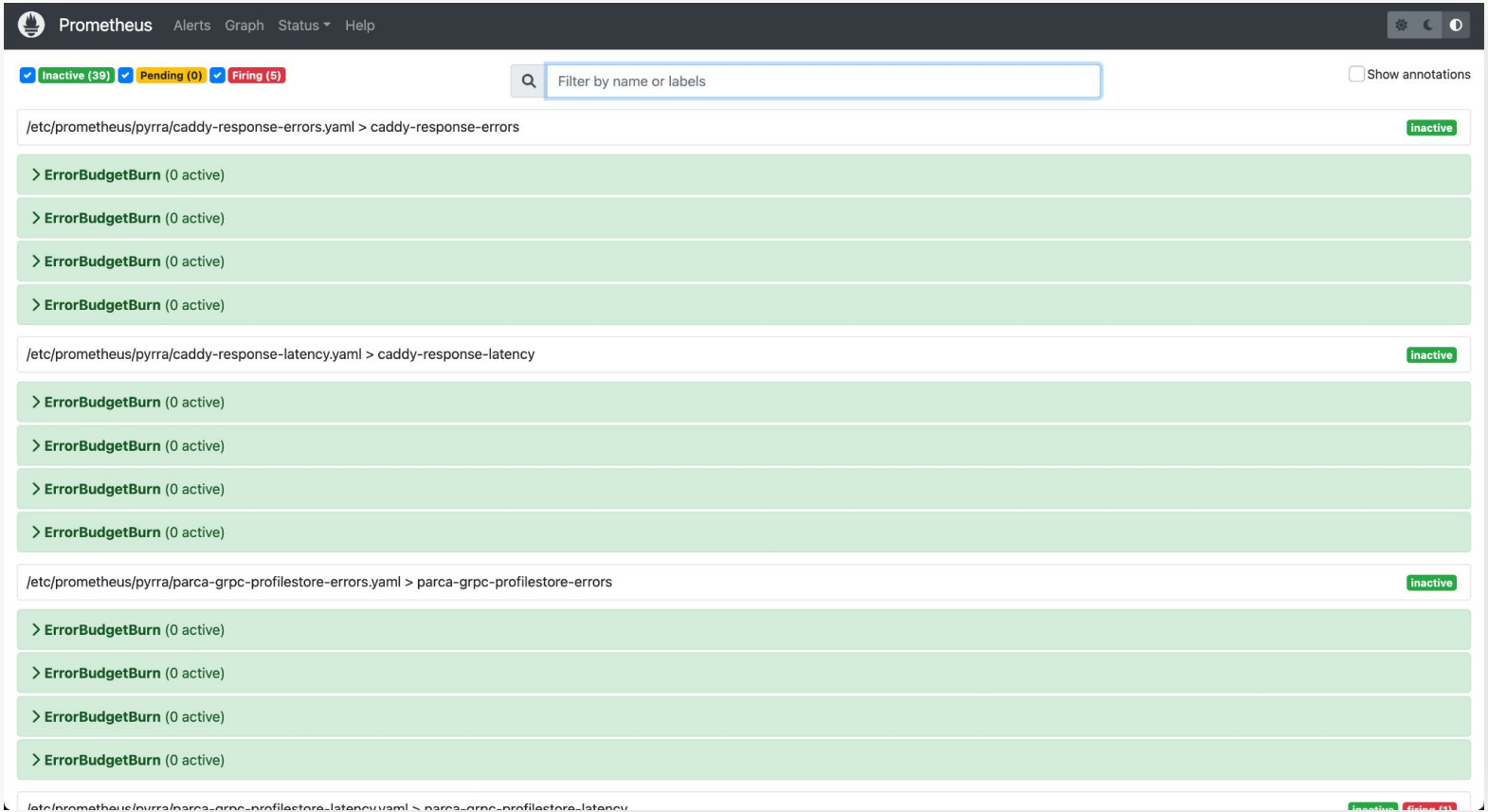
prometheus-api-errors	handler=/api/v1/alertmanagers	2w	99.00%	100.00% ⚠	100.00%
prometheus-api-errors	handler=/api/v1/query_range	2w	99.00%	100.00%	100.00%
prometheus-api-errors	handler=/api/v1/rules	2w	99.00%	100.00% ⚠	100.00%
prometheus-api-errors	handler=/api/v1/targets	2w	99.00%	100.00% ⚠	100.00%
prometheus-api-errors	handler=/api/v1/label/:name/values	2w	99.00%	100.00% ⚠	100.00%
prometheus-api-errors	handler=/api/v1/metadata	2w	99.00%	100.00% ⚠	100.00%
prometheus-api-errors	handler=/api/v1/query	2w	99.00%	100.00%	100.00%
prometheus-api-errors	handler=/api/v1/status/buildinfo	2w	99.00%	100.00% ⚠	100.00%
prometheus-api-errors	handler=/api/v1/status/config	2w	99.00%	100.00% ⚠	100.00%
prometheus-api-errors	handler=/api/v1/status/runtimeinfo	2w	99.00%	100.00% ⚠	100.00%

Generated recording rules



Rule	State	Error	Last Evaluation	Evaluation Time
record: caddy_http_response_duration_seconds:burnrate5m **expr:** sum(rate(caddy_http_response_duration_seconds_count{code=~"5..",handler="reverse_proxy",job="caddy"}[5m])) / sum(rate(caddy_http_response_duration_seconds_count{handler="reverse_proxy",job="caddy"}[5m])) **labels:** handler: reverse_proxy job: caddy slo: caddy-response-errors	OK		26.195s ago	2.074ms
record: caddy_http_response_duration_seconds:burnrate30m **expr:** sum(rate(caddy_http_response_duration_seconds_count{code=~"5..",handler="reverse_proxy",job="caddy"}[30m])) / sum(rate(caddy_http_response_duration_seconds_count{handler="reverse_proxy",job="caddy"}[30m])) **labels:** handler: reverse_proxy job: caddy slo: caddy-response-errors	OK		26.193s ago	1.695ms
record: caddy_http_response_duration_seconds:burnrate1h **expr:** sum(rate(caddy_http_response_duration_seconds_count{code=~"5..",handler="reverse_proxy",job="caddy"}[1h])) / sum(rate(caddy_http_response_duration_seconds_count{handler="reverse_proxy",job="caddy"}[1h])) **labels:** handler: reverse_proxy job: caddy slo: caddy-response-errors	OK		26.191s ago	1.719ms
record: caddy_http_response_duration_seconds:burnrate2h **expr:** sum(rate(caddy_http_response_duration_seconds_count{code=~"5..",handler="reverse_proxy",job="caddy"}[2h])) / sum(rate(caddy_http_response_duration_seconds_count{handler="reverse_proxy",job="caddy"}[2h])) **labels:** handler: reverse_proxy job: caddy slo: caddy-response-errors	OK		26.189s ago	1.488ms
record: caddy_http_response_duration_seconds:burnrate6h **expr:** sum(rate(caddy_http_response_duration_seconds_count{code=~"5..",handler="reverse_proxy",job="caddy"}[6h])) / sum(rate(caddy_http_response_duration_seconds_count{handler="reverse_proxy",job="caddy"}[6h])) **labels:** handler: reverse_proxy job: caddy slo: caddy-response-errors	OK		26.189s ago	2.806ms
record: caddy_http_response_duration_seconds:burnrate1d **expr:** sum(rate(caddy_http_response_duration_seconds_count{code=~"5..",handler="reverse_proxy",job="caddy"}[1d])) / sum(rate(caddy_http_response_duration_seconds_count{handler="reverse_proxy",job="caddy"}[1d])) **labels:** handler: reverse_proxy job: caddy slo: caddy-response-errors	OK		26.186s ago	5.184ms
record: caddy_http_response_duration_seconds:burnrate4d **expr:** sum(rate(caddy_http_response_duration_seconds_count{code=~"5..",handler="reverse_proxy",job="caddy"}[4d])) / sum(rate(caddy_http_response_duration_seconds_count{handler="reverse_proxy",job="caddy"}[4d])) **labels:** handler: reverse_proxy job: caddy slo: caddy-response-errors	OK		26.181s ago	19.289ms
alert: ErrorBudgetBurn	OK		26.161s ago	0.952ms

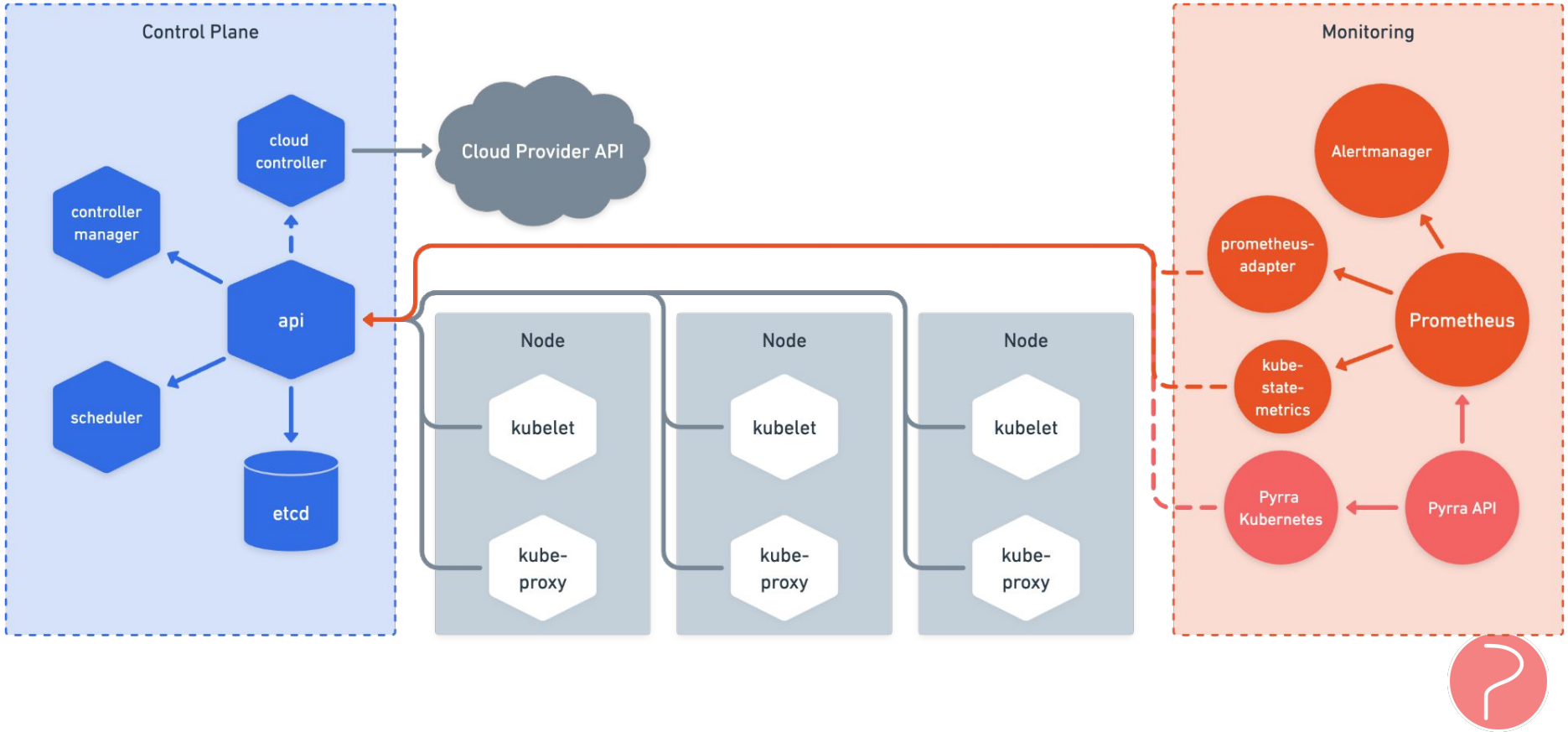
Generated alerting rules



Kubernetes Cluster

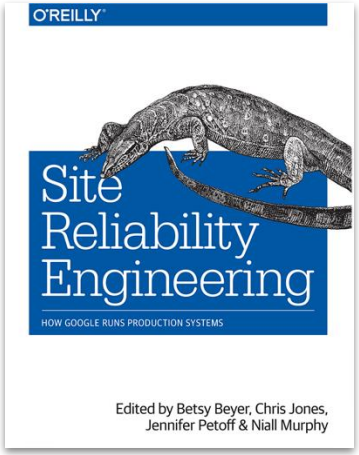
Architecture Overview

Kubernetes Architecture

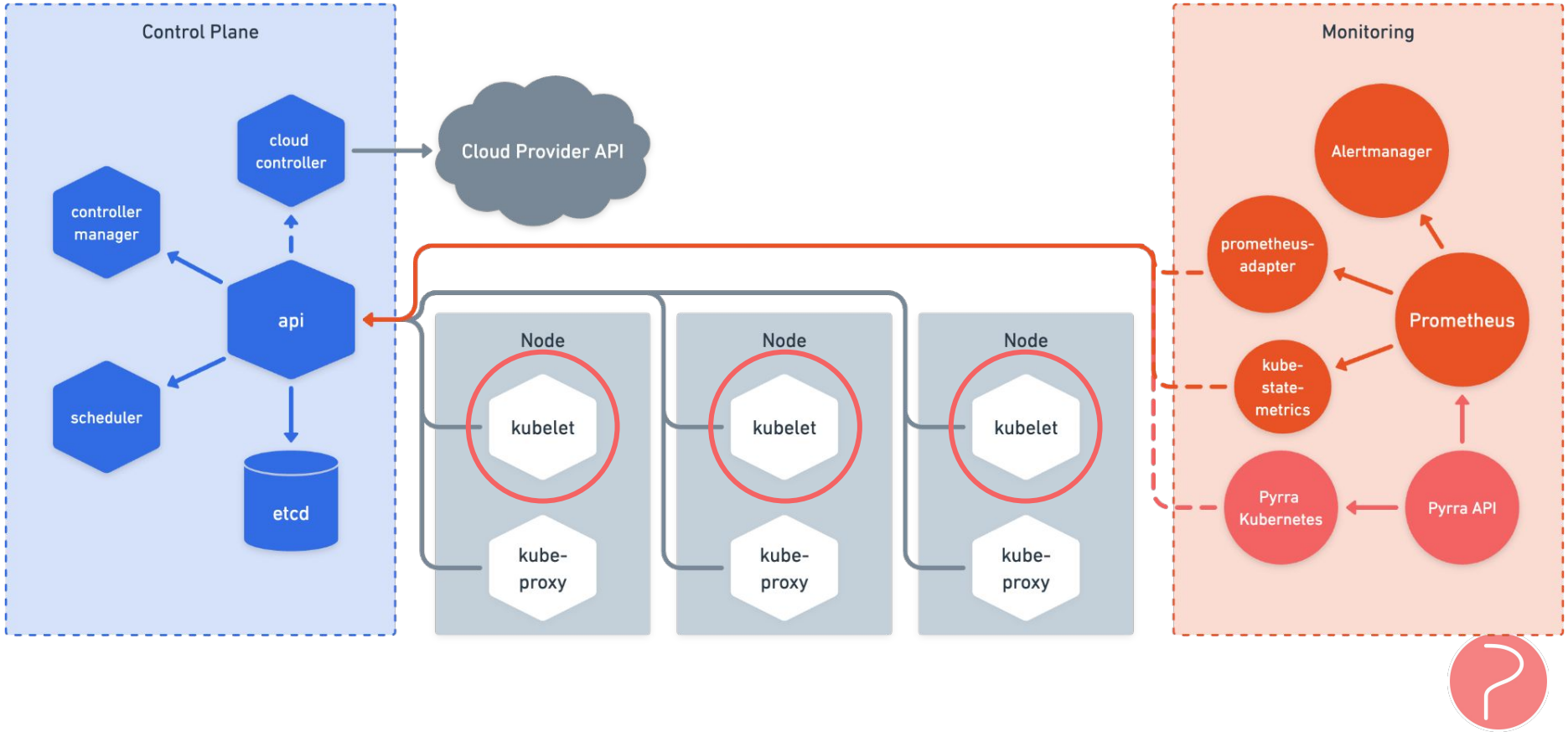


Don't create too many SLOs

[...] An understanding of what your users want from the system will inform the judicious selection of a few SLOs



Kubernetes Architecture



kubelet - runtime

The kubelet is the primary "node agent" that runs on each node.

[...] It ensures that the containers described in those PodSpecs are running and healthy.

Errors
`kubelet_runtime_operations_errors_total`

Total
`kubelet_runtime_operations_total`

kubelet - runtime

The kubelet is the primary "node agent" that runs on each node.

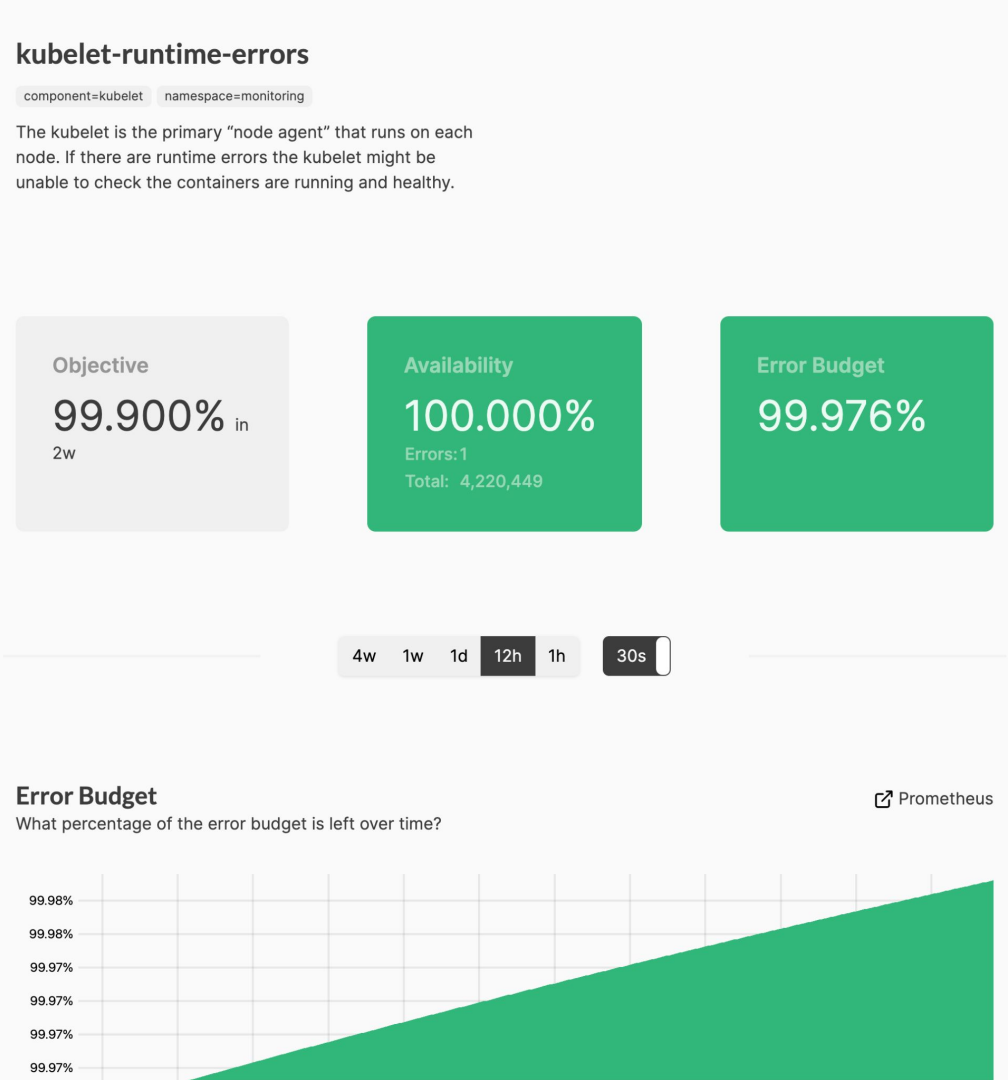
[...] it ensures that the containers described in those PodSpecs are running and healthy.

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: kubelet-runtime-errors
  namespace: monitoring
  labels:
    pyrra.dev/component: kubelet
spec:
  target: "99.9"
  window: 2w
  indicator:
    ratio:
      errors:
        metric:
          kubelet_runtime_operations_errors_total
      total:
        metric:
          kubelet_runtime_operations_total
```

kubelet - runtime

The kubelet is the primary "node agent" that runs on each node.

[...] it ensures that the containers described in those PodSpecs are running and healthy.



kubelet - requests

The kubelet is the primary "node agent" that runs on each node.

[...] It ensures that the containers described in those PodSpecs are running and healthy.

Errors

```
rest_client_requests_total{
     job="kubelet",
     code=~"5.."
}
```

Total

```
rest_client_requests_total{
     job="kubelet
}
```

kubelet - requests

The kubelet is the primary "node agent" that runs on each node.

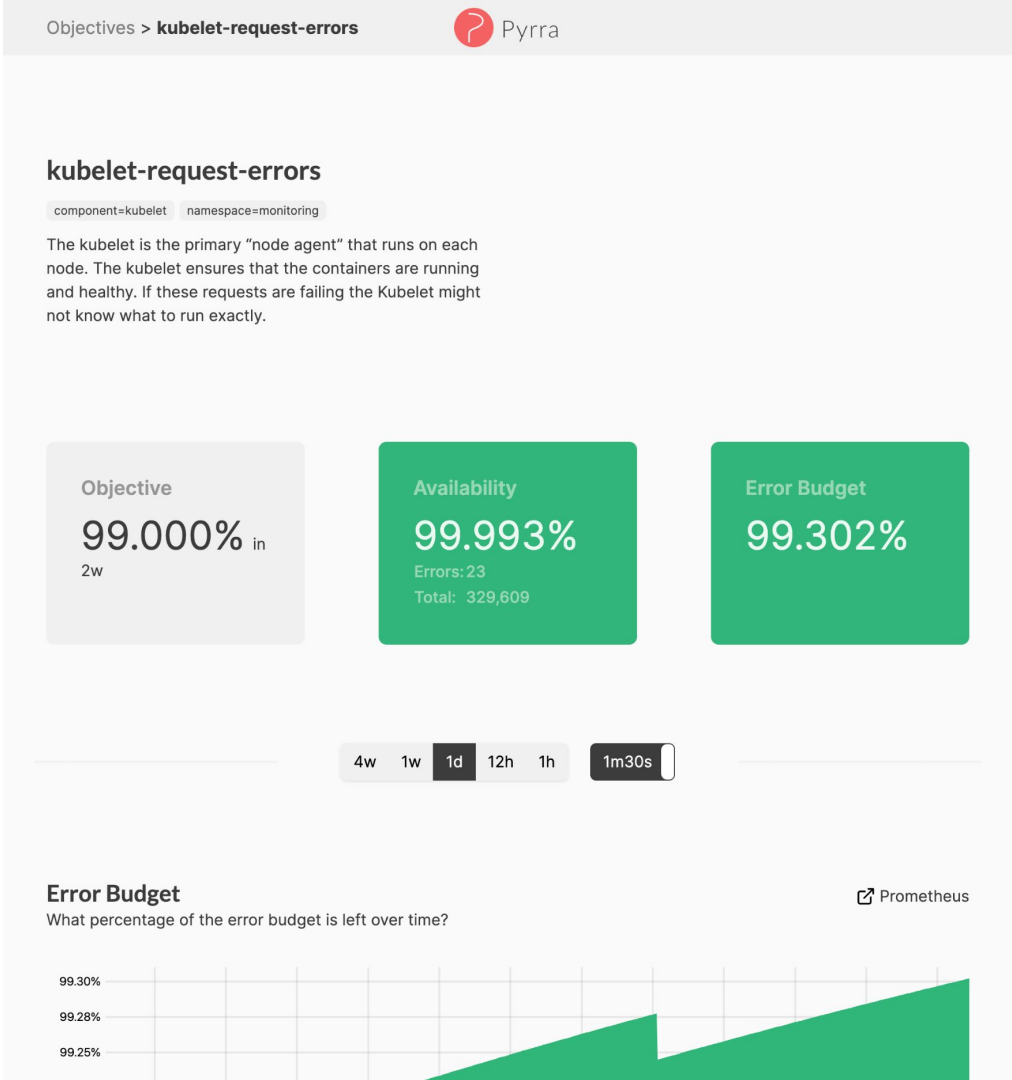
[...] It ensures that the containers described in those PodSpecs are running and healthy.

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: kubelet-request-errors
  namespace: monitoring
  labels:
    pyrra.dev/component: kubelet
spec:
  target: "99"
  window: 2w
  indicator:
    ratio:
      errors:
        metric:
          rest_client_requests_total{
            job="kubelet",
            code=~"5.."
          }
      total:
        metric:
          rest_client_requests_total{
            job="kubelet"
          }
```

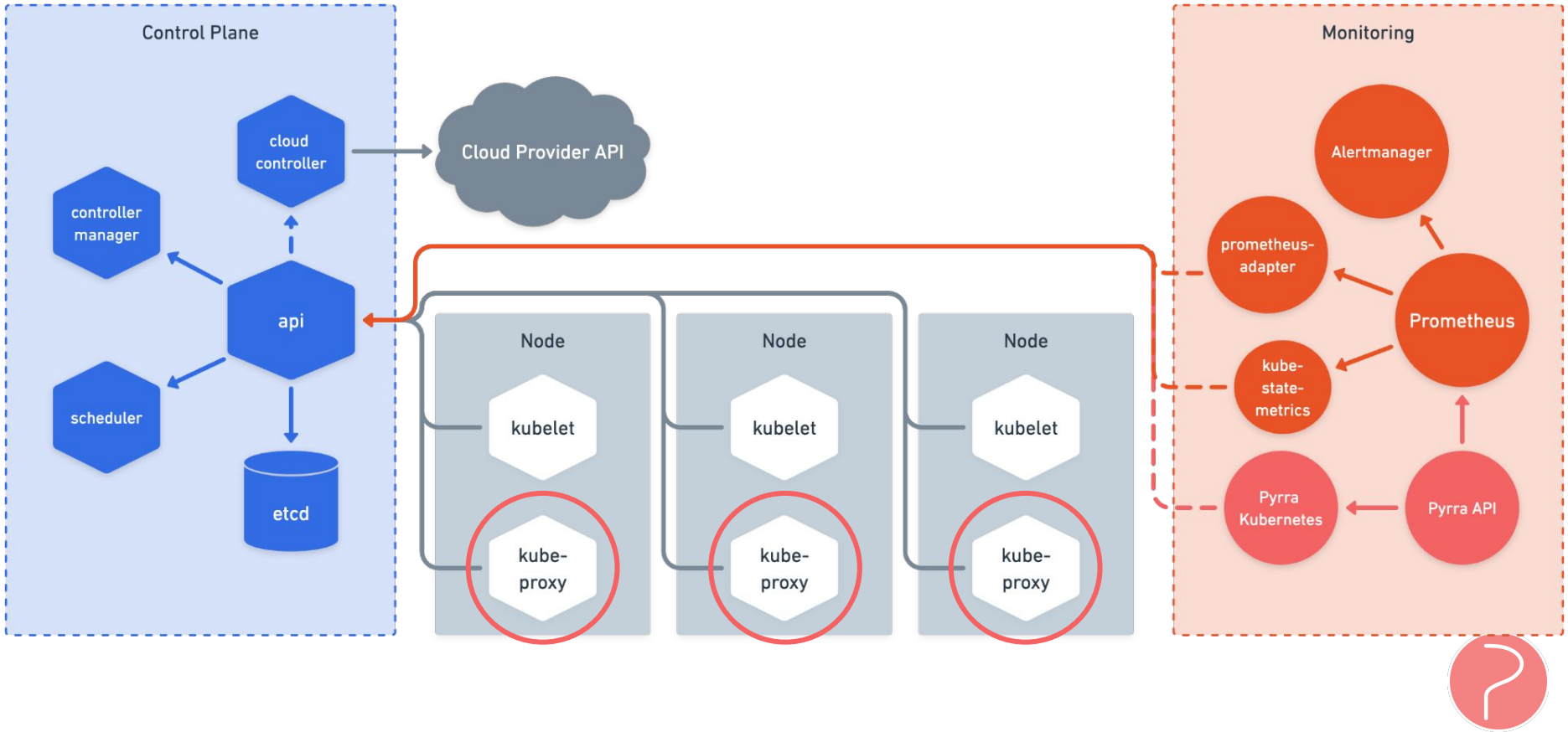
kubelet - requests

The kubelet is the primary "node agent" that runs on each node.

[...] It ensures that the containers described in those PodSpecs are running and healthy.



Kubernetes Architecture



kube-proxy

The Kubernetes network proxy runs on each node.

This reflects services as defined in the Kubernetes API on each node and can do simple TCP, UDP stream forwarding or round robin TCP,UDP forwarding across a set of backends.

Success

```
kubeproxy_sync_proxy_rules_duration_seconds_bucket{
     le="0.512"
}
```

Total

```
kubeproxy_sync_proxy_rules_duration_seconds_count
```

kube-proxy

The Kubernetes network proxy runs on each node.

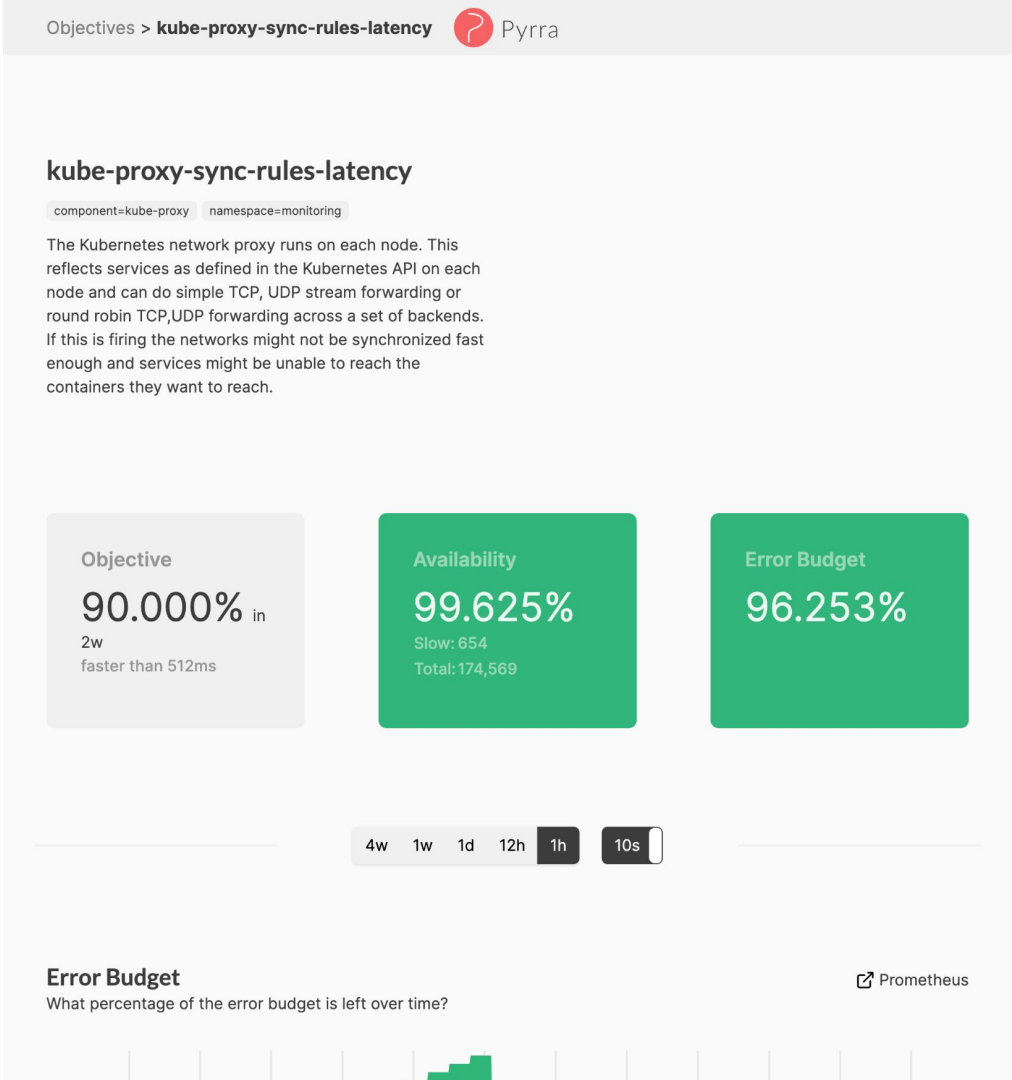
This reflects services as defined in the Kubernetes API on each node and can do simple TCP, UDP stream forwarding or round robin TCP,UDP forwarding across a set of backends.

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: kube-proxy-sync-rules-latency
  namespace: monitoring
  labels:
    pyrra.dev/component: kube-proxy
spec:
  target: "90"
  window: 2w
  indicator:
    latency:
      success:
        metric:
          kubeproxy…seconds_bucket{le="0.512"}
      total:
        metric:
          kubeproxy…seconds_count
```

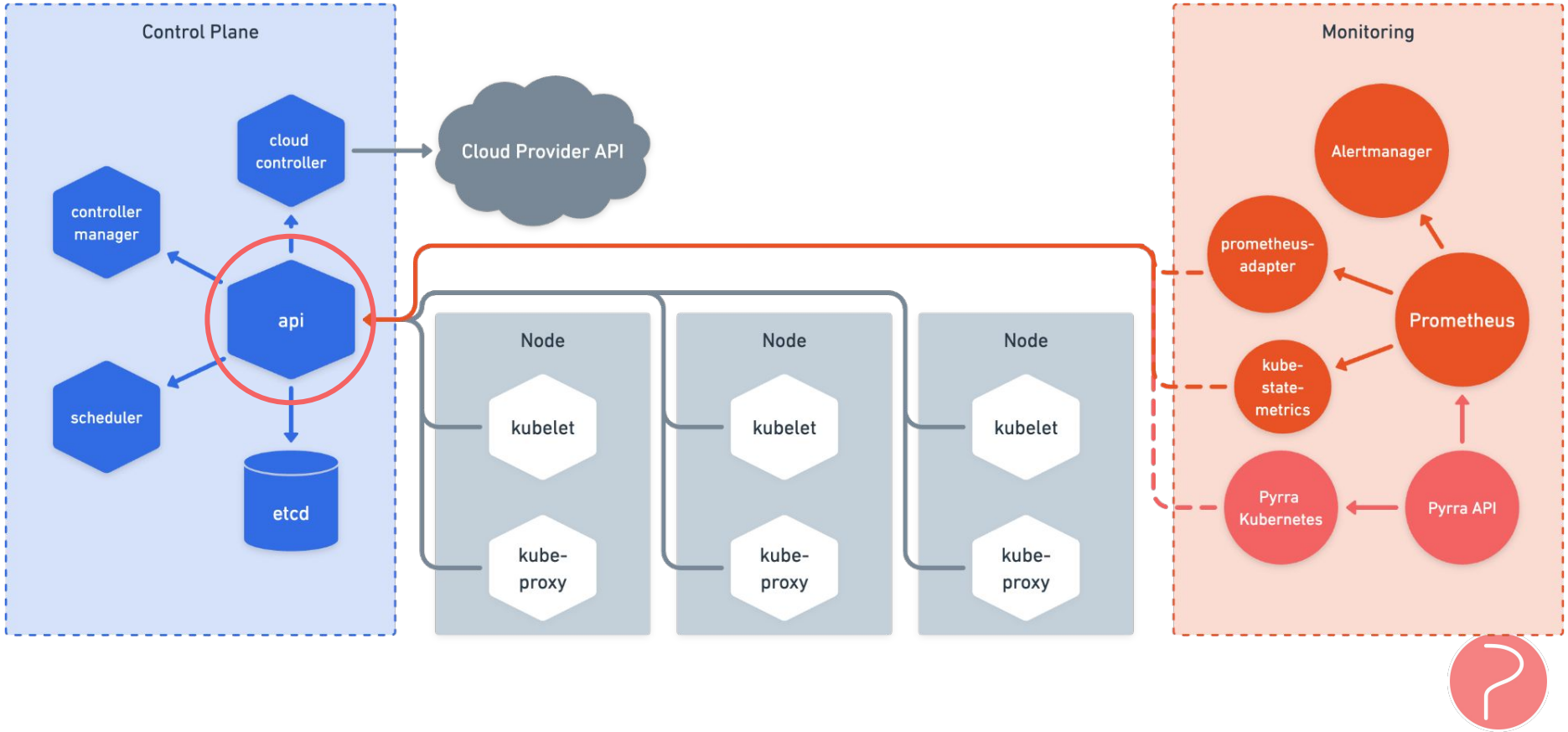
kube-proxy

The Kubernetes network proxy runs on each node.

This reflects services as defined in the Kubernetes API on each node and can do simple TCP, UDP stream forwarding or round robin TCP,UDP forwarding across a set of backends.



Kubernetes Architecture

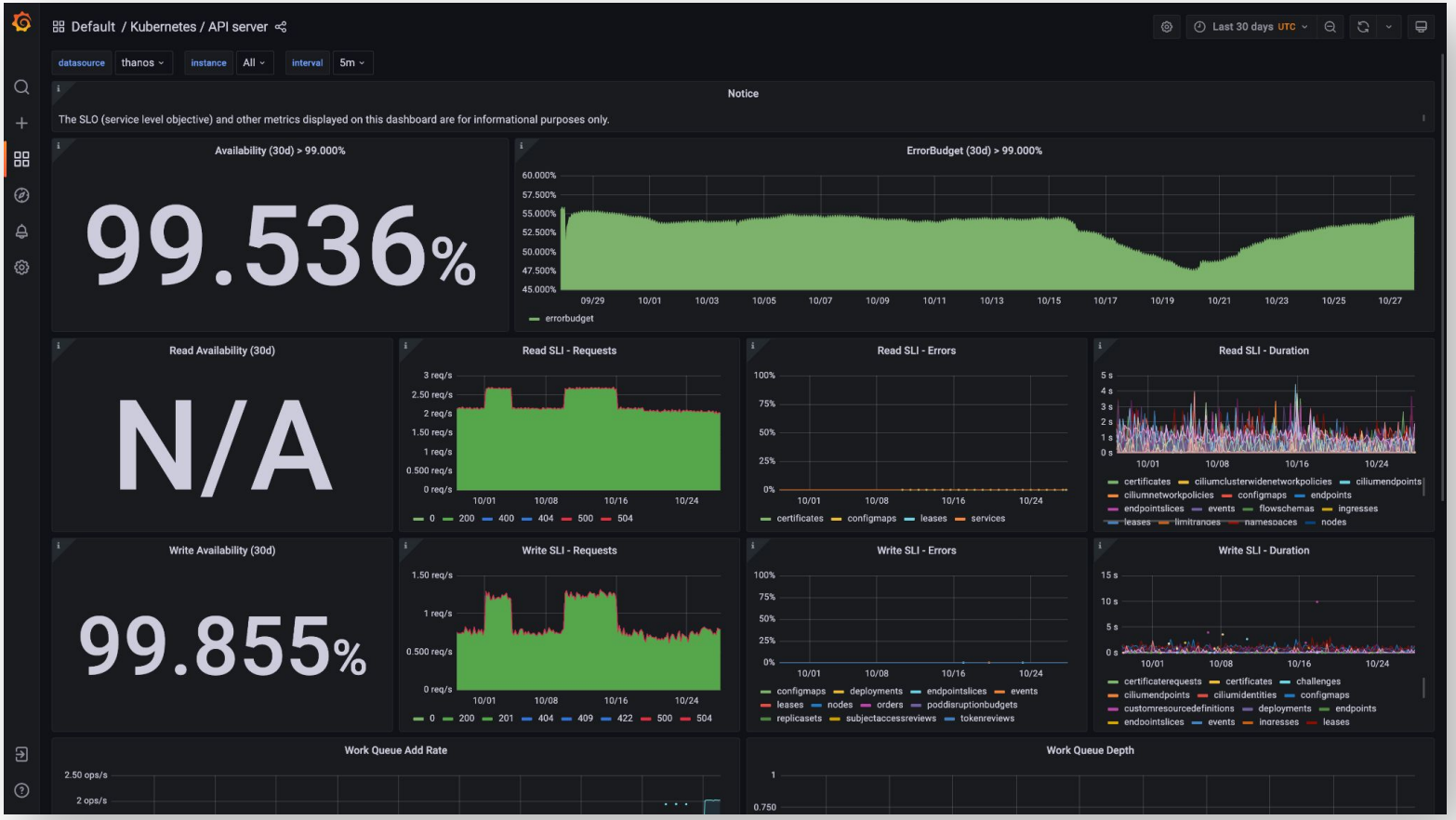


SIG Scalability

Publishes SLOs they test scale with

Status	SLI	SLO
Official	Latency of mutating API calls for single objects for every (resource, verb) pair, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, for every (resource, verb) pair, excluding virtual and aggregated resources and Custom Resource Definitions, 99th percentile per cluster-day[1] <= 1s
Official	Latency of non-streaming read-only API calls for every (resource, scope) pair, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, for every (resource, scope) pair, excluding virtual and aggregated resources and Custom Resource Definitions, 99th percentile per cluster-day[1] (a) <= 1s if `scope=resource` (b) <= 30s otherwise (if `scope=namespace` or `scope=cluster`)
Official	Startup latency of schedulable stateless pods, excluding time to pull images and run init containers, measured from pod creation timestamp to when all its containers are reported as started and observed via watch, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, 99th percentile per cluster-day[1] <= 5s

kubernetes-mixin



kubernetes-mixin

Errors (by)
• Read
• Write

Latency (by)
• Cluster
• Namespace
• Resource

kube-apiserver - read

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

Errors

```
apiserver_request_total{
      job="apiserver",
      verb=~"LIST|GET",
      code=~"5.."
}
```

Total

```
apiserver_request_total{
      job="apiserver",
      verb=~"LIST|GET"
}
```

kube-apiserver - read

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

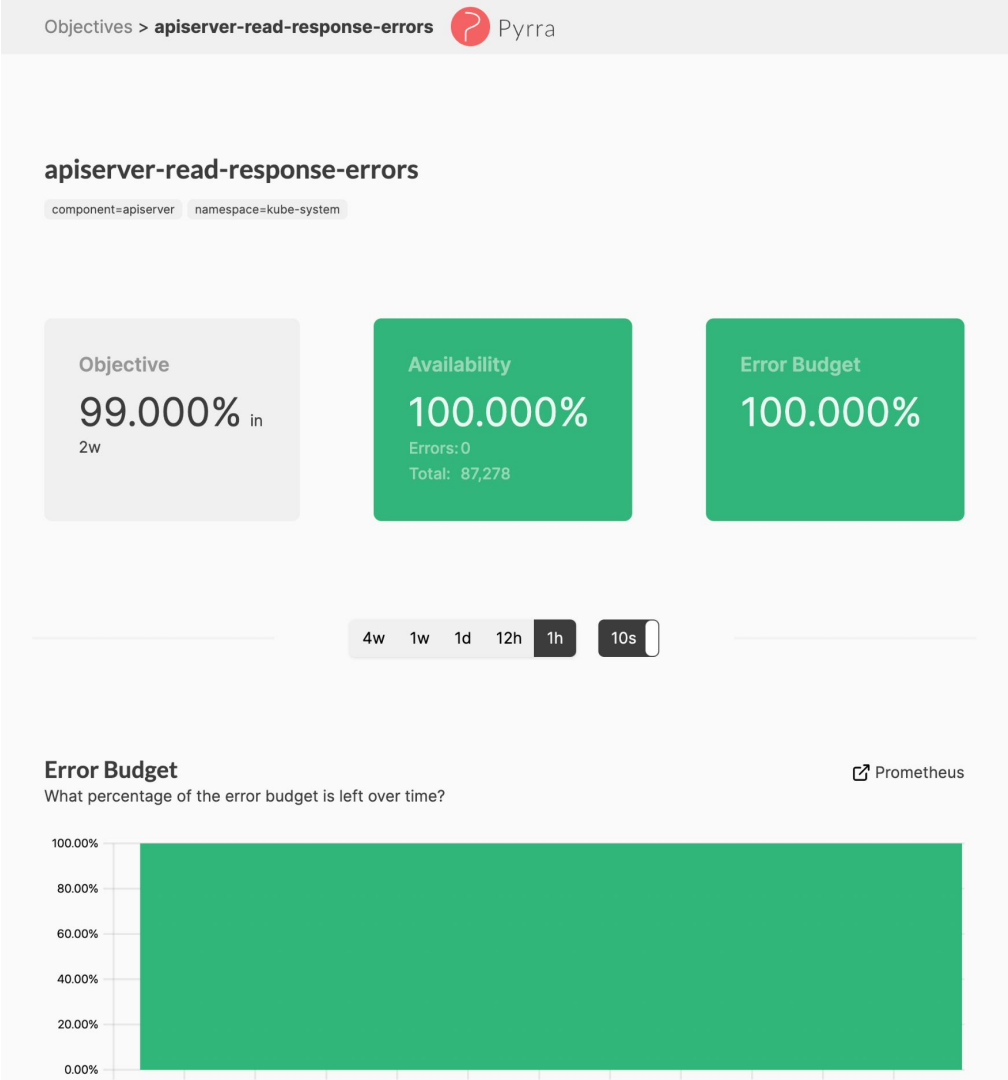
The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

```
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  labels:
    pyrra.dev/component: apiserver
  name: apiserver-write-response-errors
  namespace: kube-system
spec:
  target: "99"
  window: 2w
  indicator:
    ratio:
      errors:
        metric:
          apiserver_request_total{
            verb=~"LIST|GET",
            code=~"5.."
          }
      total:
        metric:
          apiserver_request_total{
            verb=~"LIST|GET",
          }
```

kube-apiserver - read

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.



kube-apiserver - write

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

Errors

```
apiserver_request_total{
       job="apiserver",
       verb=~"POST|PUT|PATCH|DELETE",
       code=~"5.."
}
```

Total

```
apiserver_request_total{
       job="apiserver",
       verb=~"POST|PUT|PATCH|DELETE"
}
```

kube-apiserver - write

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

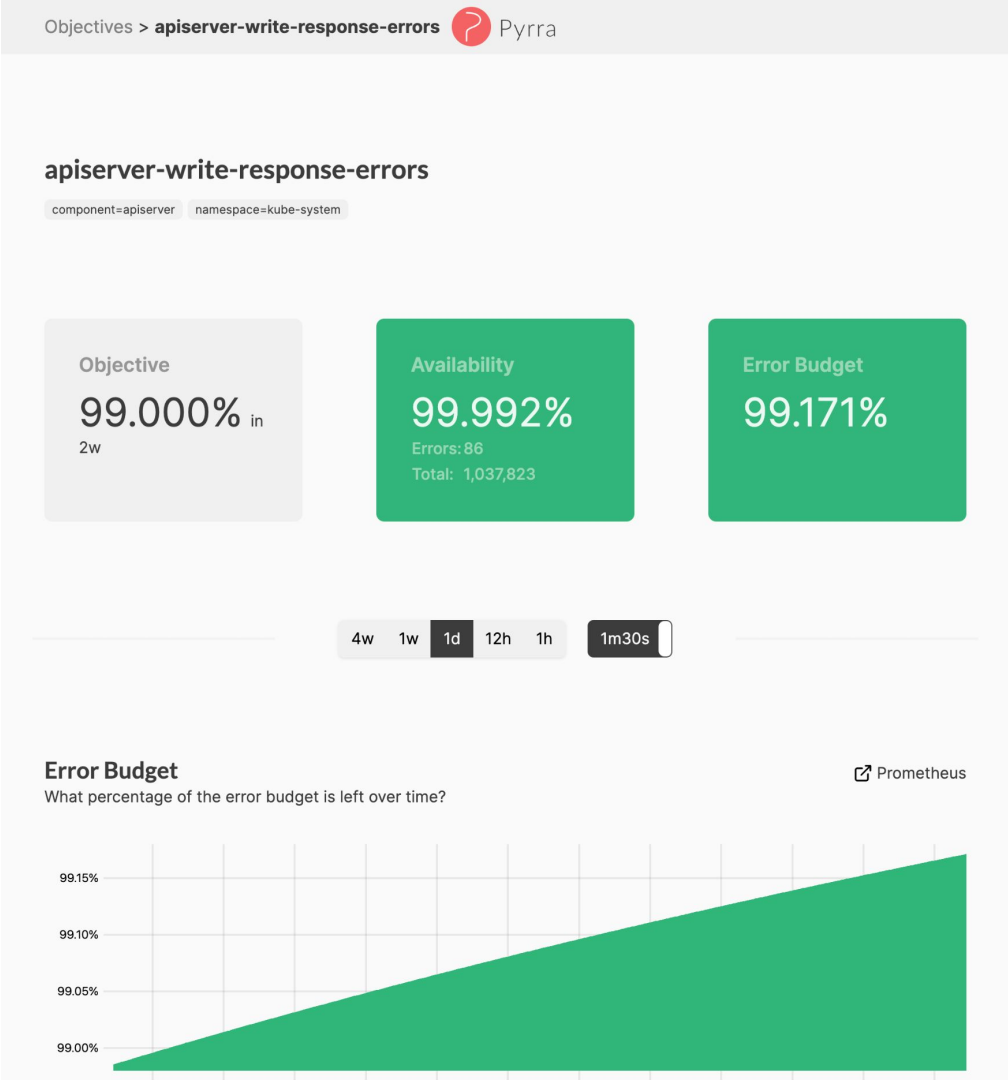
The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  labels:
    pyrra.dev/component: apiserver
  name: apiserver-write-response-errors
  namespace: kube-system
spec:
  target: "99"
  window: 2w
  indicator:
    ratio:
      errors:
        metric:
          apiserver_request_total{
            verb=~"POST|PUT|PATCH|DELETE",
            code=~"5.."
          }
      total:
        metric:
          apiserver_request_total{
            verb=~"POST|PUT|PATCH|DELETE",
          }
```

kube-apiserver - write

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.



kube-apiserver - read latency

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

Success

```
apiserver_request_duration_seconds_bucket{
        verb=~"LIST|GET",
        le="0.1"
}
```

Total

```
apiserver_request_duration_seconds_count{
        verb=~"LIST|GET",
}
```

OUTDATED

kube-apiserver - read latency

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

Success

```
apiserver_request_slo_duration_seconds_bucket{
       verb=~"LIST|GET",
       le="0.1"
}
```

Total

```
apiserver_request_slo_duration_seconds_count{
       verb=~"LIST|GET",
}
```

kube-apiserver - read latency

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

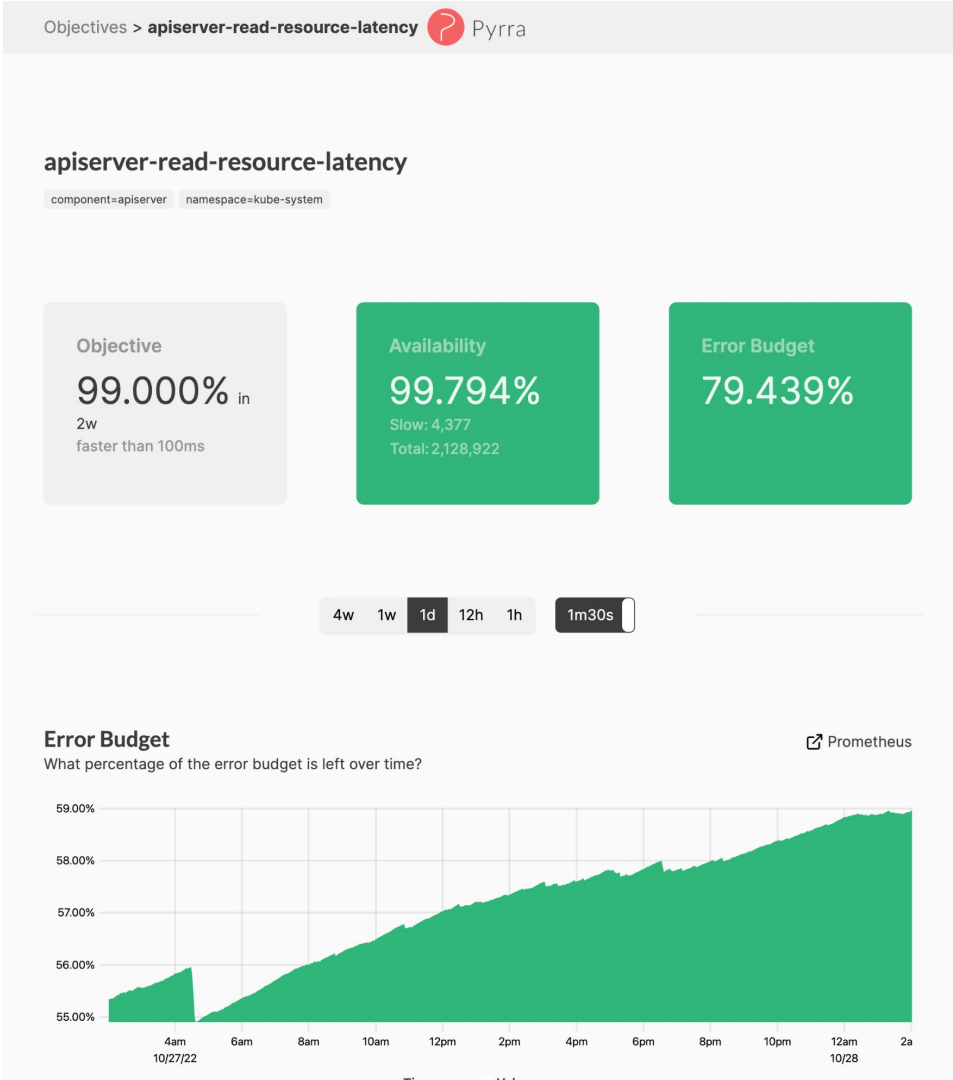
The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

```yaml
apiVersion: pyrra.dev/v1alpha1
kind: ServiceLevelObjective
metadata:
  name: apiserver-read-resource-latency
  namespace: kube-system
spec:
  target: "99"
  window: 2w
  indicator:
    latency:
      success:
        metric:
          apiserver_request_slo_duration…_bucket{
            scope=~"resource|",
            verb=~"LIST|GET",
            le="0.1"
          }
      total:
        metric:
          apiserver_request_slo_duration…_count{
            scope=~"resource|",
            verb=~"LIST|GET"
          }
```

kube-apiserver - read latency

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others.

The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.



Demo

Let's look at a real cluster

Typhoon Kubernetes Cluster

Typhoon Kubernetes cluster to manager all components

Running Fedora CoreOS on DigitalOcean

Deployed kube-prometheus

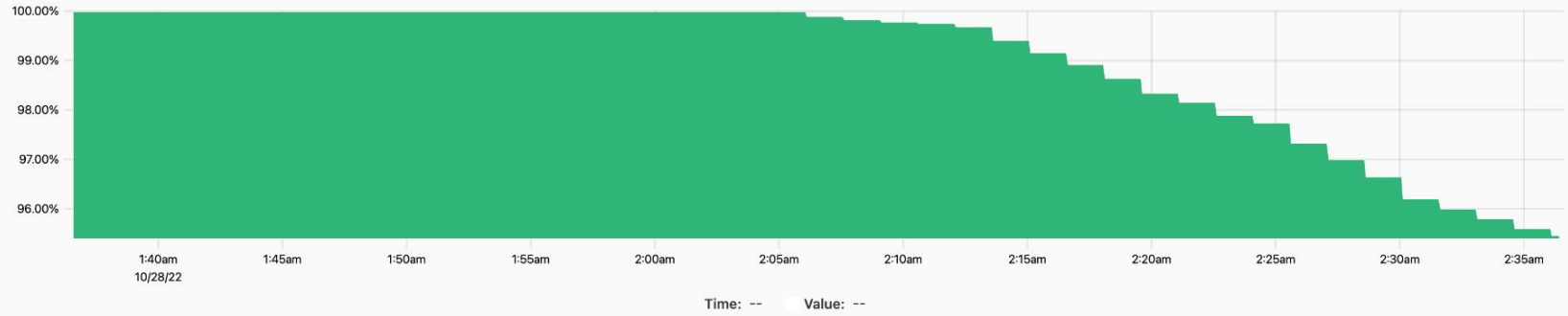
kube-prometheus

Both Pyrra and all SLOs are going to be part of kube-prometheus

Error Budget

What percentage of the error budget is left over time?

Prometheus



Requests

How many requests per second have there been?

Prometheus

Errors

What percentage of requests were too slow?

Prometheus

Duration

How long do the requests take?
p99 must be faster than 32ms.

Prometheus

What’s next?

v0.5 and future roadmap

v0.5 and future roadmap

v0.5 is about to release

→ Histograms are shown for Latency SLOs

→ Auto reloading detail page

→ Optional Grafana Dashboard

→ Absent alert for missing SLIs

→ OpenShift example

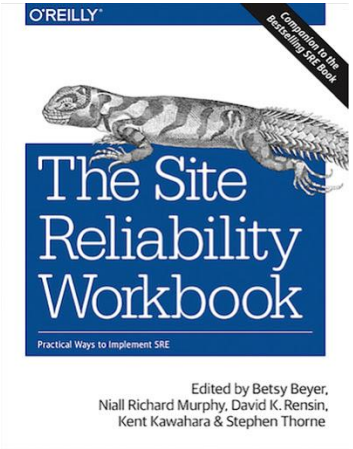
Future

→ Editor to interactively create SLOs

→ Graphs for the burn rate

Try Pyrra live on

demo.pyrra.dev



THANK YOU!

Say hi 👋

@PyrraSLO

@nadinevehling

@MetalMatze