

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



What are SLOs?

Fundamentals



99% of requests succeed

within 5s



Search or jump to... / Pulls Issues Marketplace Explore

pyrra-dev / pyrra Public

Code Issues 18 Pull requests 16 Discussions Actions Projects 1 Security 3 ...

main Go to file Add file Code About

metalmatze Merge pull request #254 from pyrra-dev/dependabot ... 8 days ago 563

.github build(deps): bump docker/metadata-action from 3... 23 days ago

config config/crd/bases: Generate YAML and JSON CR... 2 months ago

cue.mod Rename the project to Pyrra 10 months ago

docs Update docs and config to v0.3.0 3 months ago

examples Release v0.3.4 2 months ago

kubernetes Update tests to not have empty objective windows 2 months ago

openapi Update openapi to 5.4.0 2 months ago

slo Expose the Window struct last month

ui Merge pull request #250 from pyrra-dev/dependabot ... 11 days ago

.dockerrcignore Move single main package to root of the repository 9 months ago

.editorconfig Make Pyrra work based on labels and matchers 7 months ago

.errcheck_excludes.txt Add golangci-lint with gofumpt and more 2 months ago

.gitattributes Ignore generated files from Linguist statistics 9 months ago

.gitignore Move single main package to root of the repository 9 months ago

.golangci.yml Introduce gokit/log for consistent logging 2 months ago

CONTRIBUTING.md Update to Go 1.17 as minimum 3 months ago

Dockerfile build(deps): bump node from 17.8.0 to 18.0.0 14 days ago

LICENSE Create LICENSE 9 months ago

Makefile Update openapi to 5.4.0 2 months ago

PROJECT Rename the project to Pyrra 10 months ago

README.md Merge branch 'release-0.3' 2 months ago

api.yaml Support adding inactive alerts to multi burn rate a... 2 months ago

demo.pyrra.dev

docker kubernetes golang monitoring time-series metrics prometheus slo thanos

Readme Apache-2.0 license 476 stars 8 watching 24 forks

Releases 7

v0.3.4 - 2022-02-21 (Latest) on Feb 21 + 6 releases

Packages 1

pyrra

Contributors 10

Languages

caddy-response-errors

namespace=monitoring

Caddy is the reverse proxy used for this demo. We expect Caddy to answer at least 99% successfully.

Objective in 4w

99.002%

Availability

99.955%

Error Budget

95.526%

What are SLIs?

4w 1w 1d 12h 1h

Error Budget

What percentage of the error budget is left over time?

 Prometheus



caddy-response-errors

namespace=monitoring

Caddy is the reverse proxy used for this demo. We expect Caddy to answer at least 99% successfully.

Objective in 4w

99.000%

Availability

97.955%

Error Budget

95.526%

What is Availability?

4w 1w 1d 12h 1h

Error Budget

What percentage of the error budget is left over time?

 Prometheus



caddy-response-errors

namespace=monitoring

Caddy is the reverse proxy used for this demo. We expect Caddy to answer at least 99% successfully.

Objective in 4w

99.000%

Availability

99.975%

Error Budget

95.516%

What is an Error Budget?

4w 1w 1d 12h 1h

Error Budget

What percentage of the error budget is left over time?

 Prometheus



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

What's an error budget?

My team deals with alert fatigue but it's hard and risky to get SLO-based alerting right.

How do I configure Multi Burn Rate Alerts?

How does my team get started?

Defining SLOs is super complicated.



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

Objectives > **caddy-response-latency**  Pyrra

caddy-response-latency

We want our demo to be fast and therefore we want 90% of our responses to be faster than 50ms as seen by Caddy.

Objective
90.000% in 4w
faster than 50ms

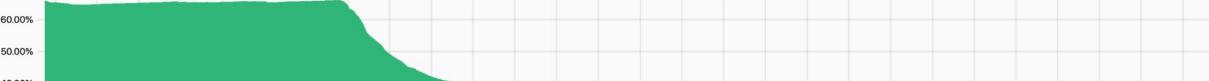
Availability
95.438%
Slow: 30,703
Total: 673,052

Error Budget
54.382%

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

Error Budget

What percentage of the error budget is left over time?



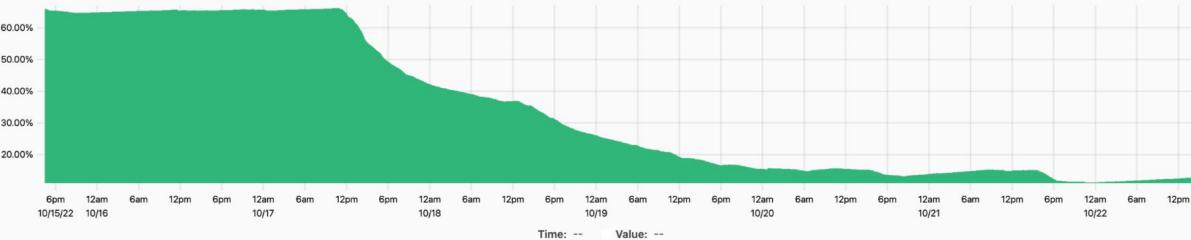
Prometheus 

4w 1d 12h 1h 10m

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?
p90 must be faster than 50ms.

 Prometheus



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

 Pyrra

Objectives

team=prometheus ×

NAME ◇		TIME WINDOW ◇	OBJECTIVE ◇	AVAILABILITY ◇	ERROR BUDGET ↑	ALERTS ◇
prometheus-api-errors	team=prometheus handler=/api/v1/query_range	2w	99.00%	100.00%	99.77%	
prometheus-rule-evaluation-failures	team=prometheus	2w	99.99%	100.00%	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/status/buildinfo	2w	99.00%	100.00% ⚠	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/write	2w	99.00%	100.00%	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/label/:name/values	2w	99.00%	100.00%	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/query	2w	99.00%	100.00%	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/metadata	2w	99.00%	100.00%	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/series	2w	99.00%	100.00% ⚠	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/format_query	2w	99.00%	100.00% ⚠	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/rules	2w	99.00%	100.00% ⚠	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/query_exemplars	2w	99.00%	100.00% ⚠	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/targets	2w	99.00%	100.00% ⚠	100.00%	
prometheus-api-errors	team=prometheus handler=/api/v1/labels	2w	99.00%	100.00% ⚠	100.00%	

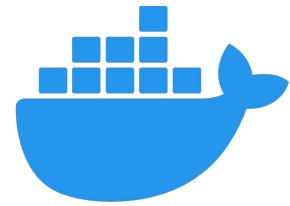
All availabilities and error budgets are calculated across the entire time window of the objective.



Deployments



kubernetes

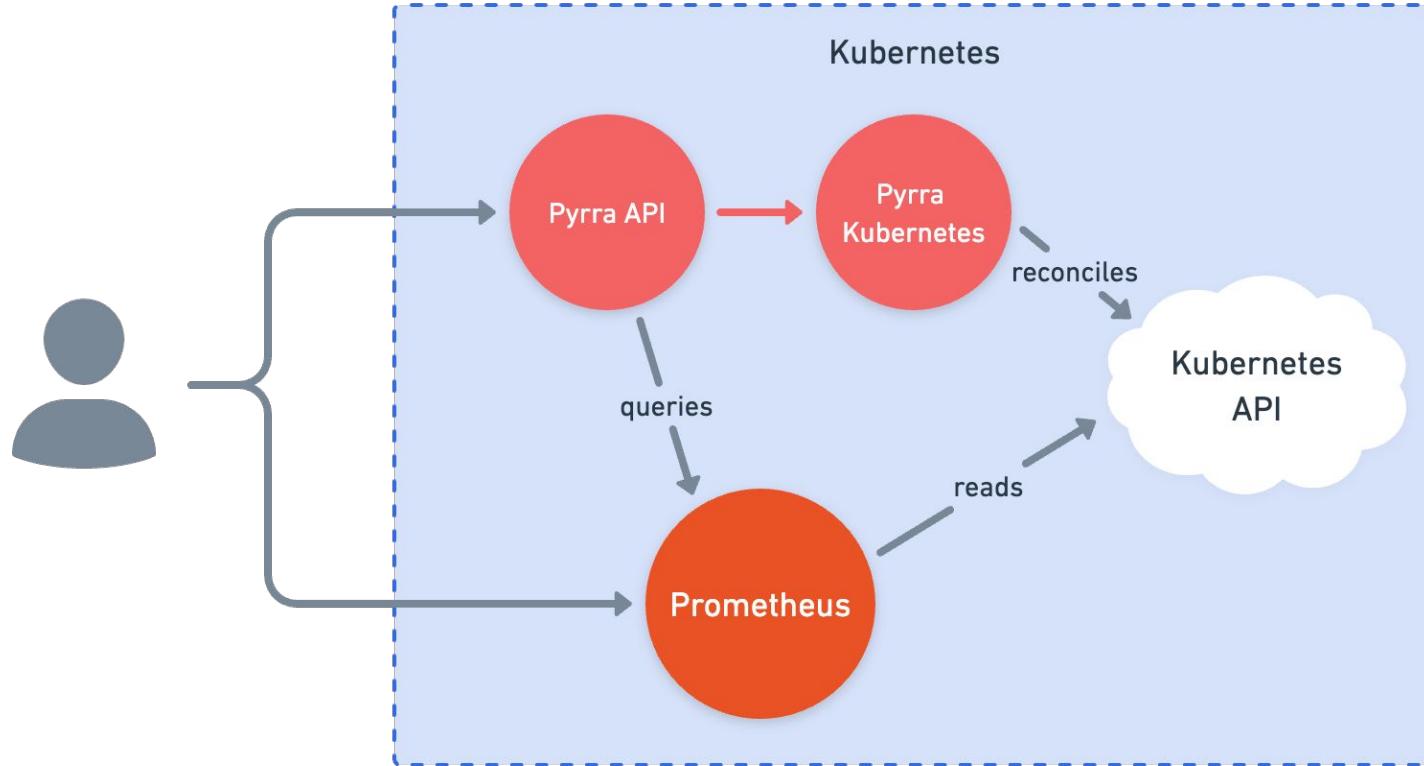


docker®

[● ◀] **systemd**



Deployments



Deployments - API

```
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

```
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



Prometheus



cortex



How to create an SLO?

Looking at the configuration



Configuring SLOs

```
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

```
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

```
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

```
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

```
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%

```
metric: prometheus_http_requests_total{handler=~"/api.*",code=~"5.."}  
metric: prometheus_http_requests_total{handler=~"/api.*"}
```

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

Prometheus Alerts Graph Status ▾ Help

Inactive (39) Pending (0) Firing (5)

Show annotations

Filter by name or labels

/etc/prometheus/pyrra/caddy-response-errors.yaml > caddy-response-errors inactive

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

/etc/prometheus/pyrra/caddy-response-latency.yaml > caddy-response-latency inactive

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

/etc/prometheus/pyrra/parca-grpc-profilestore-errors.yaml > parca-grpc-profilestore-errors inactive

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

> ErrorBudgetBurn (0 active)

/etc/prometheus/pyrra/parca-grpc-profilestore-latency.yaml > parca-grpc-profilestore-latency inactive

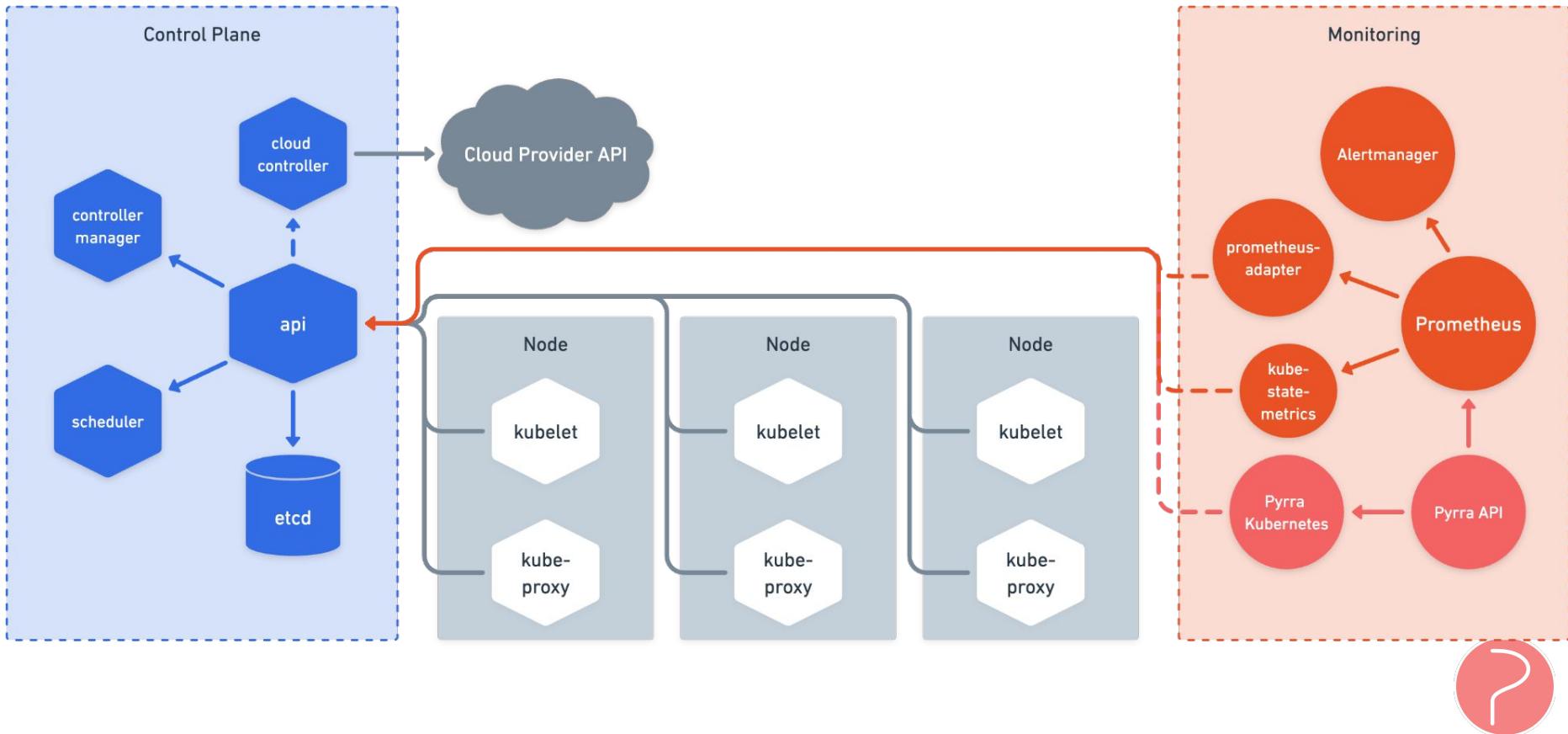
?

Kubernetes Cluster

Architecture Overview

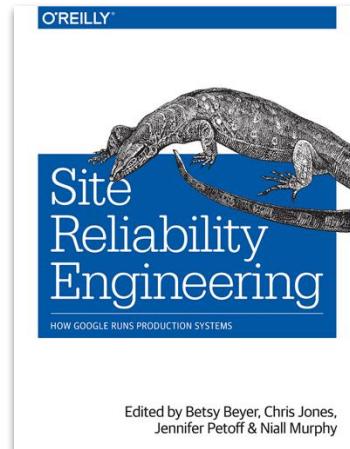


Kubernetes Architecture

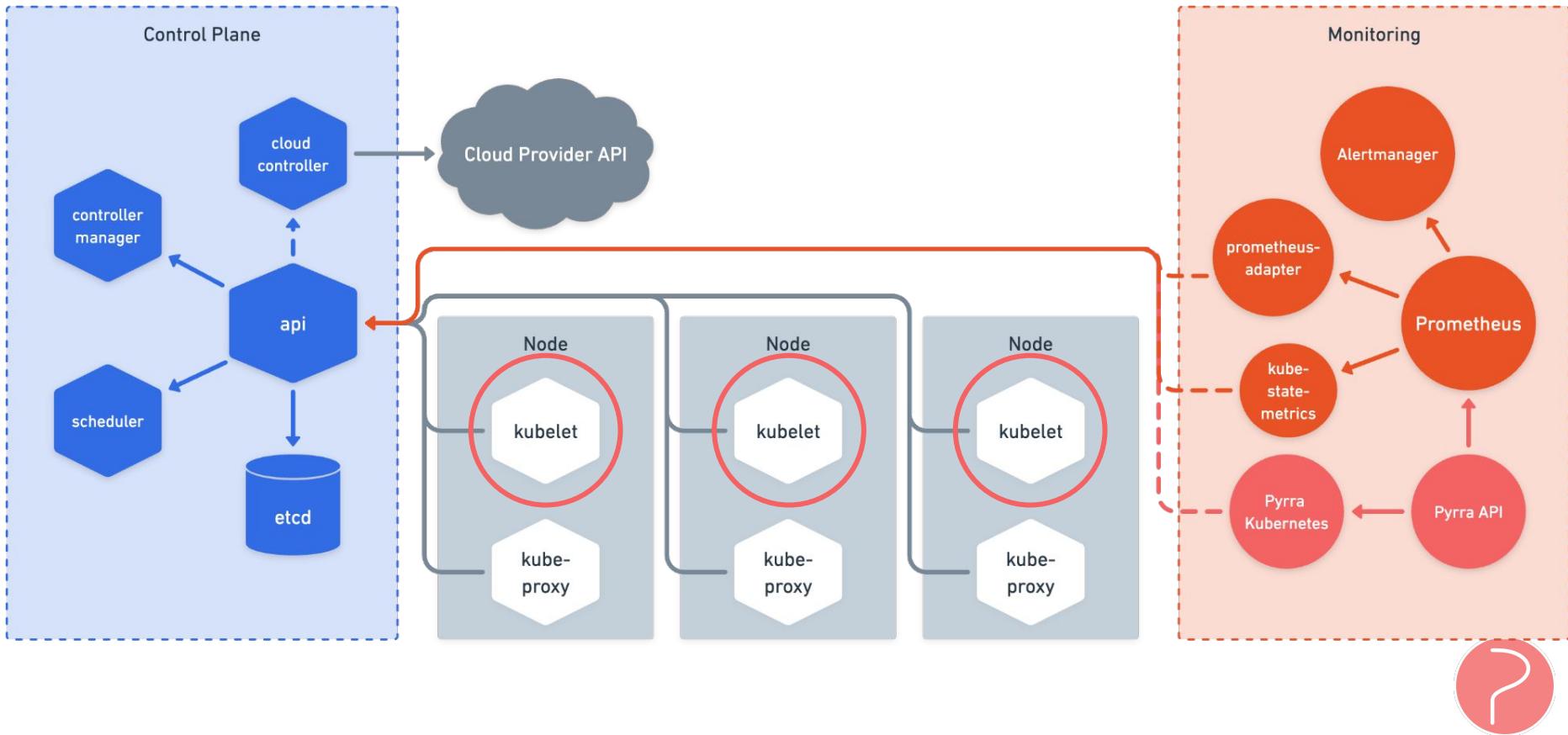


Don't create too many SLOs

[...] 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.

```
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

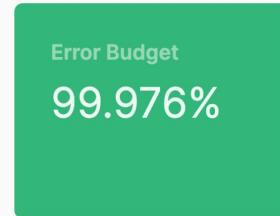
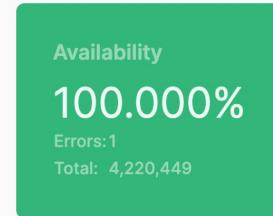
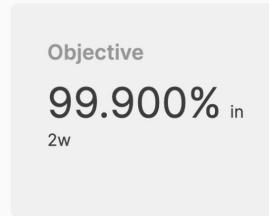
The kubelet is the primary "node agent" that runs on each node.

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

kubelet-runtime-errors

component=kubelet namespace=monitoring

The kubelet is the primary "node agent" that runs on each node. If there are runtime errors the kubelet might be unable to check the containers are running and healthy.



4w 1w 1d 12h 1h 30s

Error Budget

What percentage of the error budget is left over time?

 Prometheus



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.

```
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

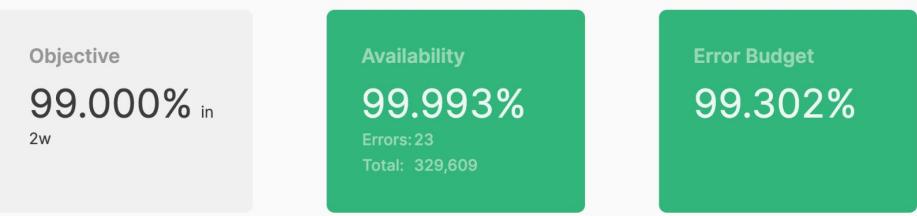
The kubelet is the primary "node agent" that runs on each node.

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

kubelet-request-errors

component=kubelet namespace=monitoring

The kubelet is the primary "node agent" that runs on each node. The kubelet ensures that the containers are running and healthy. If these requests are failing the Kubelet might not know what to run exactly.



4w 1w 1d 12h 1h 1m30s

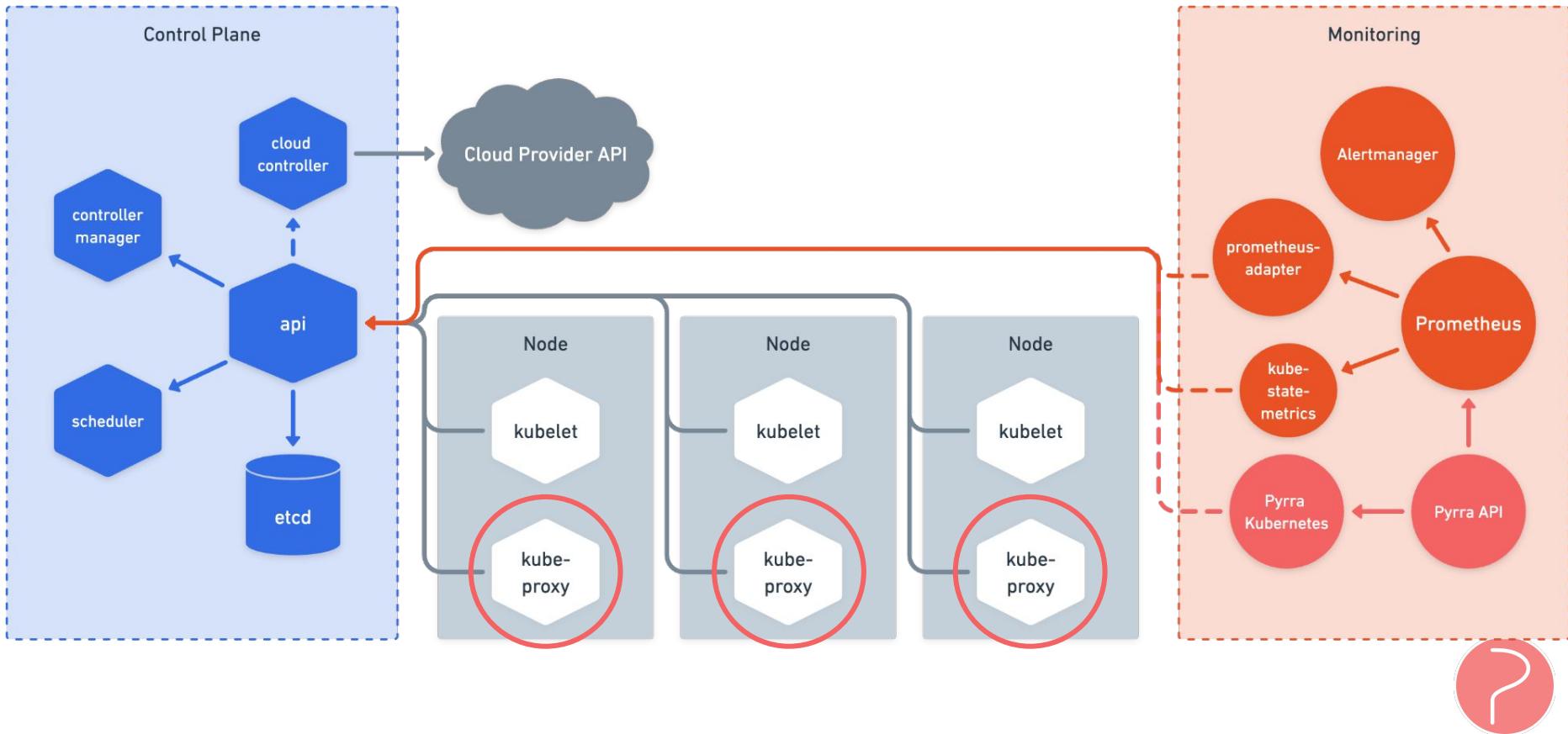
Error Budget

What percentage of the error budget is left over time?

Prometheus



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.

```
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

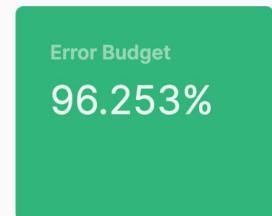
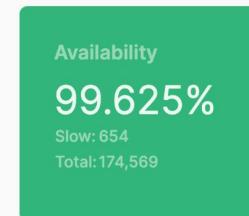
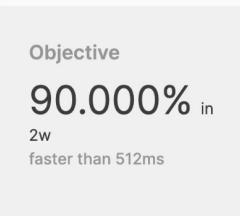
The Kubernetes network proxy runs on each node.

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.

kube-proxy-sync-rules-latency

component=kube-proxy namespace=monitoring

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. If this is firing the networks might not be synchronized fast enough and services might be unable to reach the containers they want to reach.



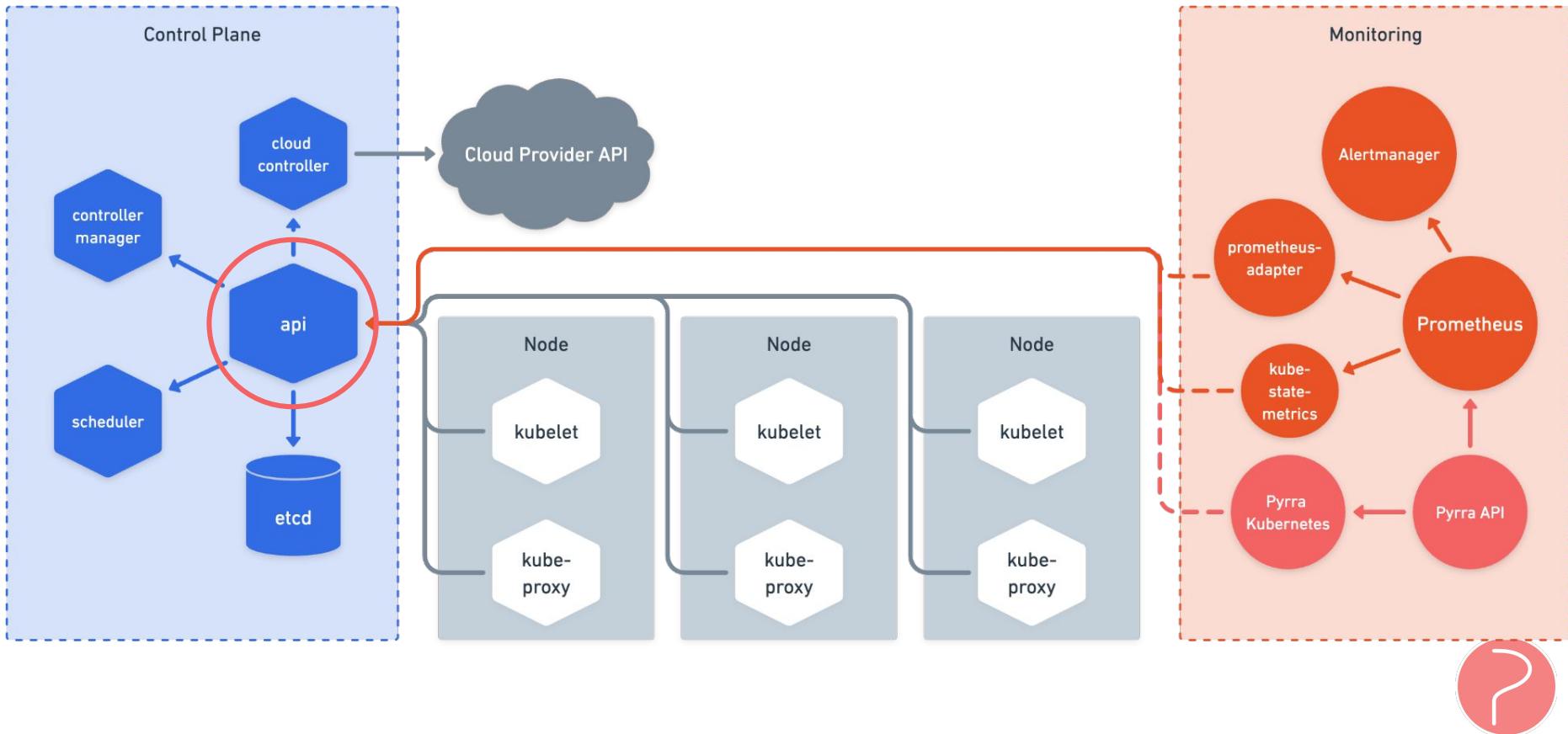
4w 1w 1d 12h 1h 10s

Error Budget

What percentage of the error budget is left over time?

 Prometheus

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 $\leq 1\text{s}$
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) $\leq 1\text{s}$ if <code>scope=resource</code> (b) $\leq 30\text{s}$ otherwise (if <code>scope=namespace</code> or <code>scope=cluster</code>)
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 $\leq 5\text{s}$

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.

apiserver-read-response-errors

component=apiserver namespace=kube-system

Objective

99.000%
in
2w

Availability

100.000%
Errors: 0
Total: 87,278

Error Budget

100.000%

4w 1w 1d 12h 1h 10s

Error Budget

What percentage of the error budget is left over time?

 Prometheus



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.

```
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",
        }
```



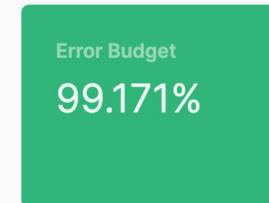
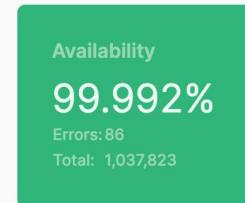
kube-apiserver - write

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.

apiserver-write-response-errors

component=apiserver namespace=kube-system



4w 1w 1d 12h 1h 1m30s

Error Budget

What percentage of the error budget is left over time?

Prometheus



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.

```
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.

apiserver-read-resource-latency

component=apiserver namespace=kube-system

Objective

99.000% in

2w
faster than 100ms

Availability

99.794%

Slow: 4,377
Total: 2,128,922

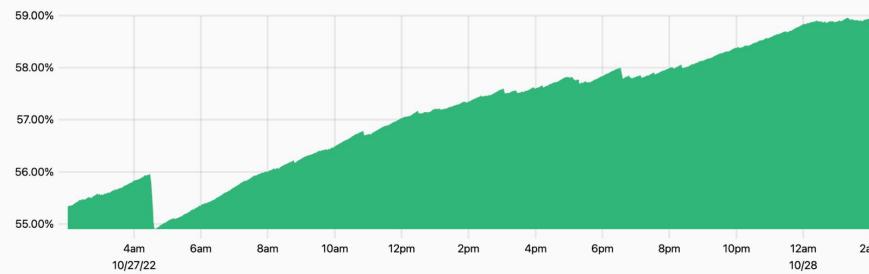
Error Budget

79.439%

4w 1w 1d 12h 1h 1m30s

Error Budget

What percentage of the error budget is left over time?



Demo

Let's look at a real cluster

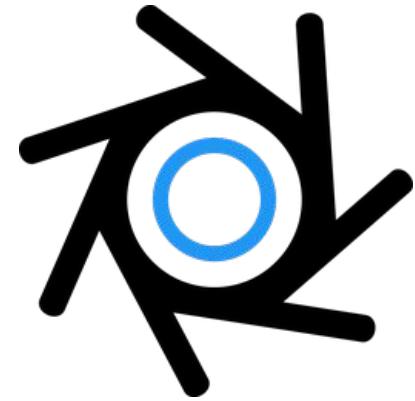


Typhoon Kubernetes Cluster

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

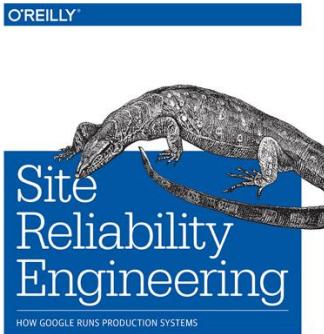
- Histograms are shown for Latency SLOs
- Auto reloading detail page
- Optional Grafana Dashboard
- Absent alert for missing SLIs
- OpenShift example

Future

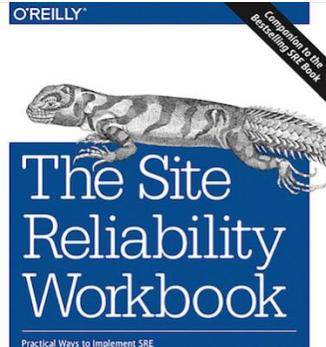
- Editor to interactively create SLOs
- Graphs for the burn rate



Try Pyrra live on
demo.pyrra.dev



Edited by Betsy Beyer, Chris Jones,
Jennifer Petoff & Niall Murphy



Edited by Betsy Beyer,
Niall Richard Murphy, David K. Rensin,
Kent Kawahara & Stephen Thorne



THANK YOU!

Say hi 

@PyrraSLO

@nadinevehling

@MetalMatze

