



## From Monitoring to Observability: Left Shift your SLOs



Michael Friedrich  
Senior Developer Evangelist



@dnsmichi



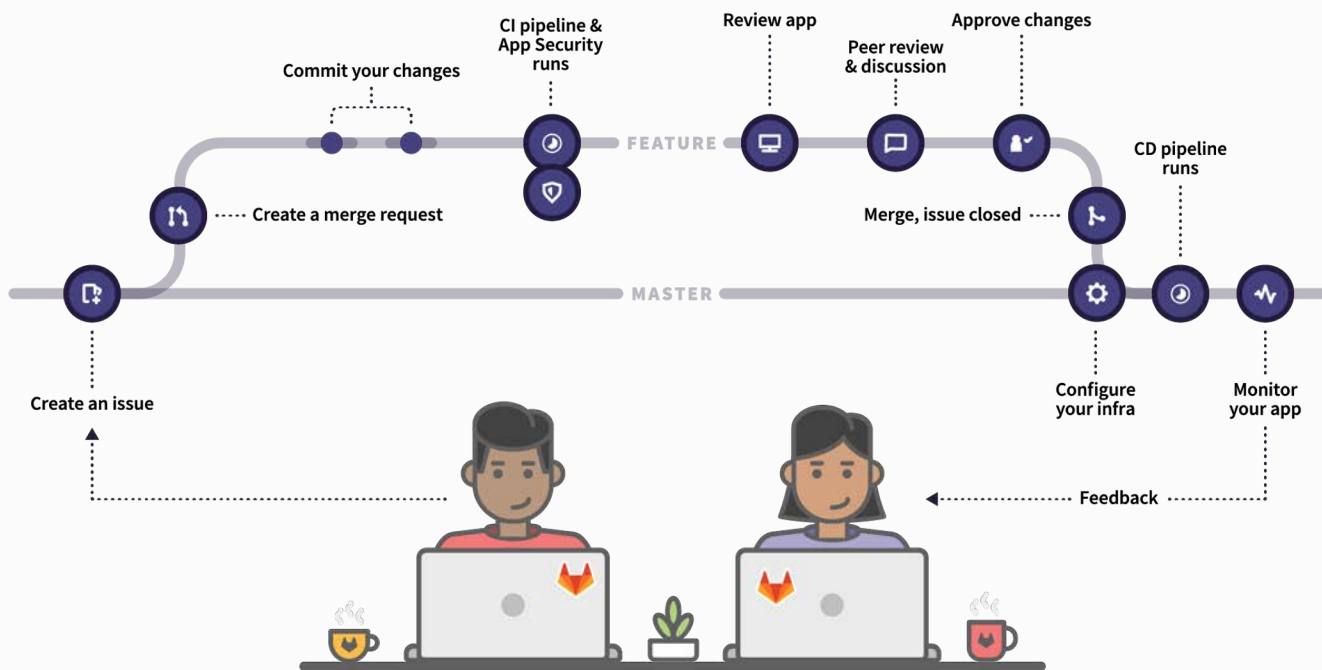
A DevOps Tale



# Security shifted left



- Security is not optional anymore
- Create + Verify, Secure + Protect





- State blackbox monitoring
- SLOly adding metrics
- Traditional SLA reporting
  - State changes over time
  - Metric data points and trends





- Service Level
  - Agreement - 99.5% availability
  - Objective - 99.9% availability
  - Indicator - errors, latency, ...
- Error budgets

<https://landing.google.com/sre/workbook/chapters/implementing-slos/>  
<https://engineering.bitnami.com/articles/implementing-slos-using-prometheus.html>  
<https://grafana.com/blog/2019/11/27/kubecon-recap-how-to-include-latency-in-slo-based-alerting/>  
<https://github.com/google/prometheus-slo-burn-example/blob/master/prometheus/slos.rules.yml>  
<https://github.com/prometheus/prometheus/issues/6209>



# Golden Signals



- Latency
- Traffic
- Errors
- Saturation

Code instrumentation needed







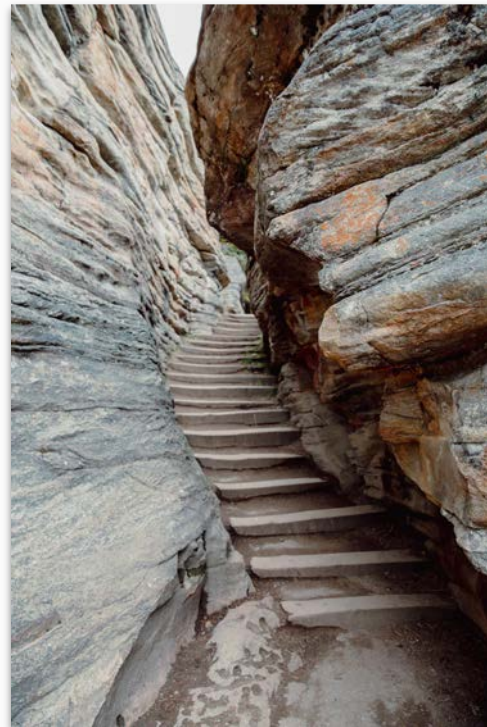
# GitLab

Story time:  
Rocket Science with C++





- Slow REST & JSON-RPC API responses
- CPU overload with threads
- Let's use light-weight co-routines in C++
  - Stackless co-routines
  - Function pointer on the heap
  - Stack unwinding for continuation

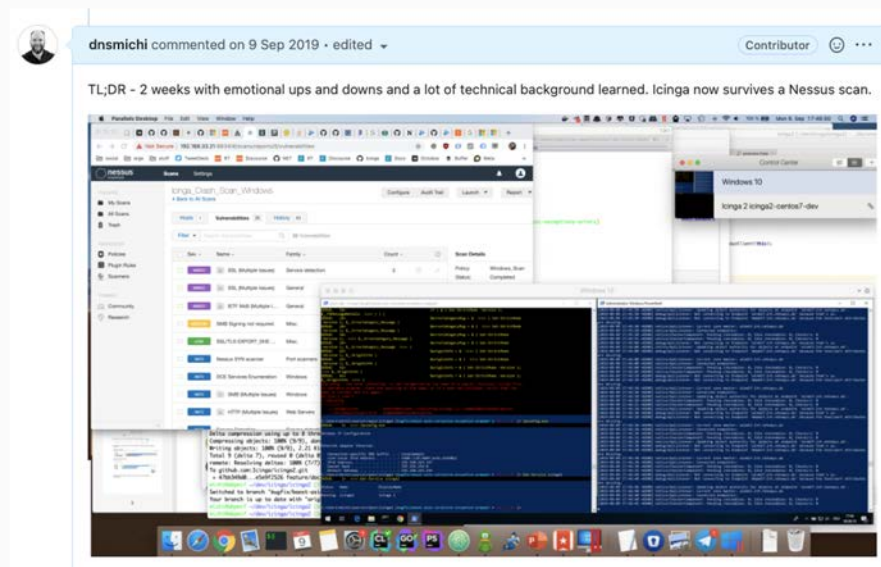




# Devs debugging software



- Crash happens in production
  - Only with 1000+ API clients
- Memory corruption
- Memory exhausted - leaks?
- Another crash only on Windows
  - Stack guards
  - Nessus security scanner



<https://github.com/Icinga/icinga2/issues/7532#issuecomment-574739579>  
<https://github.com/Icinga/icinga2/issues/7431#issuecomment-529543471>



- Stack & heap memory meets Ops requirements
- SLI
  - Heap/Stack memory usage level
- SLO
  - Not more than 10% increase
- Measure real-time production environments
- Add chaos engineering to API requests





Story time:  
Really SLO (w)





## How I cut GTA Online loading times by 70%

T0ST 2021-02-28

GTA Online. Infamous for its slow loading times. Having picked up the game again to finish some of the newer heists I was *shocked* (/s) to discover that it still loads just as slow as the day it was released 7 years ago.

It was time. Time to get to the bottom of this.



- Measure the login time
- Add application timing points
- 1st iteration
  - Metrics
- 2nd iteration
  - Tracing spans



- MR/PR deploys from CI/CD pipeline
- End-to-end test scopes
  - User login
  - User playing
- SLO
  - $\text{login\_time} < 2\text{m}$
  - $\text{login\_time} < 5\text{m}$  on low latency connections
- Quality gates = CI/CD failure = stop release





# GitLab

Story time:  
It's always DNS





- Slack.com returned SERVFAIL on 2021-09-30

```
→ dig slack.com

; <◇> DiG 9.10.6 <◇> slack.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 5213
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:;, udp: 512
;; OPT=15: 00 0a ("..")
;; QUESTION SECTION:
;slack.com.                                IN      A

;; Query time: 111 msec
;; SERVER: 192.168.2.1#53(192.168.2.1)
;; WHEN: Thu Sep 30 19:29:53 CEST 2021
;; MSG SIZE rcvd: 44
```

# Slack down - analysis



- Ask @8.8.8.8 (Google DNS) - it resolves
- Could be missing IPv6 glue record?
  - 💡 dig, dnstracer CLI

```
→ dig slack.com @8.8.8.8
; <<>> DiG 9.10.6 <<>> slack.com @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 54916
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags::; udp: 512
;; QUESTION SECTION:
;slack.com.                                IN      A
;
;; ANSWER SECTION:
slack.com.                                60      IN      A      3.123.248.34

;; Query time: 68 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Sep 30 19:31:48 CEST 2021
;; MSG SIZE rcvd: 54
```

```
\\ e.gtld-servers.net [com] (192.12.94.30)
\\ ns-1493.awsdns-58.org [slack.com] (205.251.197.213) (cached)
\\ ns-1493.awsdns-58.org [slack.com] (2600:9000:5305:d500:0000:0000:0001) Not queried
\\ ns-1901.awsdns-45.co.uk [slack.com] (205.251.199.109) (cached)
\\ ns-1901.awsdns-45.co.uk [slack.com] (2600:9000:5307:6d00:0000:0000:0001) Not queried
\\ ns-606.awsdns-11.net [slack.com] (205.251.194.94) (cached)
\\ ns-606.awsdns-11.net [slack.com] (2600:9000:5302:5e00:0000:0000:0001) Not queried
\\ ns-166.awsdns-20.com [slack.com] (205.251.192.166) (cached)

dev/everyonecancontribute/ci-cd-efficiency-workshop via 🐙 system
+ dig ns-166.awsdns-20.com any
;; Truncated, retrying in TCP mode.
;; Connection to fe80::1#53(fe80::1) for ns-166.awsdns-20.com failed: host unreachable.

; <<>> DiG 9.10.6 <<>> ns-166.awsdns-20.com any
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 8562
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags::; udp: 512
;; QUESTION SECTION:
;ns-166.awsdns-20.com.                    IN      ANY
;
;; ANSWER SECTION:
ns-166.awsdns-20.com.                    41603   IN      A      205.251.192.166
ns-166.awsdns-20.com.                    66875   IN      AAAA   2600:9000:5300:a600::1

;; Query time: 27 msec
;; SERVER: 192.168.2.1#53(192.168.2.1)
;; WHEN: Thu Sep 30 20:52:41 CEST 2021
;; MSG SIZE rcvd: 93
```



- DNSSEC validation fails
- DNSKEY/DS records in .com
  - Then removed, but cached
- Recursive resolvers (ISPs) do DNSSEC validation
  - KEY missing - SERVFAIL
- **.com zone time-to-live (wait) = 24h**
- Workaround
  - Google 8.8.8.8
  - Faster cache invalidation

```
→ dig slack.com @8.8.8.8

; <=> DiG 9.10.6 <=> slack.com @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->HEADER<-- opcode: QUERY, status: NOERROR, id: 54916
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;slack.com.                IN      A

;; ANSWER SECTION:
slack.com.                 60      IN      A      3.123.248.34

;; Query time: 68 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Sep 30 19:31:48 CEST 2021
;; MSG SIZE rcvd: 54
```



- Staging environment for DNS change
- Infrastructure as Code aka “Git Ops”
  - Git + CI/CD
  - Monitoring:
    - 5 global locations verify DNSSEC chain
  - SLI: Failure
  - SLO: 0 out of 5

Failed SLOs in CI/CD staging = never hits production

Failed SLOs in CI/CD prod = automated rollback

*(well, DNS cache cannot be tricked so easily)*





- Turn back time to 2011
  - .at ccTLD got DNSSEC
- Signing hardware = state machine of steps
- Friday afternoon script change
- No more signing
  - no DNS updates for domain delegation
- Monitoring?





## .at is outdated - who you gonna call?



- Zone serial was renumbered
  - Unix timestamp (DNSSEC requirement)
- Serial + offset < now?
  - Alarm at 3am per email
  - Escalation at 4am per SMS
  - *(from all nameservers, no grouped alerts)*
  - Debugging at 5am is not fun
- The change was persisted in a Git repository
  - And then rolled to prod





- Staging signing hardware
- Changes are rolled with IaC / GitOps
- Service Level Indicator
  - Zone serial age
- Service Level Objective
  - `now() - serial_age < 1h`





- Border gate protocol (BGP) for routing announcements
- DNS nameservers as anycast
  - Routers announce FB AS32934
- Routing announcements dropped
- Recursive resolvers (Google, Cloudflare, ...)
  - All \*.facebook.com queries failed
  - No routes, no DNS query responses
  - No caching, resolvers on CPU 🔥
  - Collateral damage in DNS

<https://twitter.com/FSchweitzer/status/1445157800427073542>  
<https://twitter.com/GossiTheDog/status/1445065065527394321>  
<https://twitter.com/dnsmichi/status/1445314162377338888>

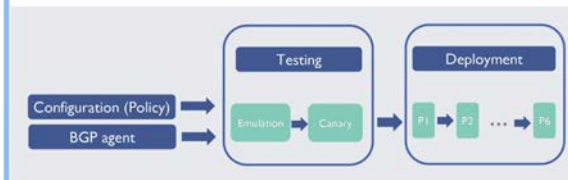


# Facebook uses Continuous Delivery



- BGP agent development
- Policies
- Testing & Deployment
- Datacenter deployment

To minimize impact on production traffic while achieving high release velocity for the BGP agent, we built our own testing and incremental deployment framework, consisting of unit testing, emulation, and canary testing. We use a multi-phase deployment pipeline to push changes to agents.



*Testing and deployment pipeline.*

<https://twitter.com/dnsmichi/status/1445310247934087171>  
<https://twitter.com/dnsmichi/status/1445468075357138953?s=21>  
<https://twitter.com/dnsmichi/status/1445146654361989122>



- Policy audit
- Develop stronger rules set to disallow BGP policy changes
- SLI
  - Audit fail
- SLO
  - count(failed policy pushes)

## Production SLOs

- Nameservers detecting unreachable DCs

This was the source of yesterday's outage. During one of these routine maintenance jobs, a command was issued with the intention to assess the availability of global backbone capacity, which unintentionally took down all the connections in our backbone network, effectively disconnecting Facebook data centers globally. Our systems are designed to audit commands like these to prevent mistakes like this, but a bug in that audit tool didn't properly stop the command.

<https://twitter.com/dnsmichi/status/1445310247934087171>  
<https://twitter.com/dnsmichi/status/1445468075357138953?s=21>  
<https://twitter.com/monkchips/status/1445330303271292930>



# GitLab

## Story time: Docker Hub Rate Limiting







- Turn back time to September 2020
- Docker announced rate limits
- Possibly affected
  - CI/CD pipelines
  - Cloud native deployments
  - Organisations behind a NAT
  - Cloud providers





- Limits are applied
- Where are the values?
- Simulate a pull
- Header response
- Script to parse
- Prometheus Exporter



*Pull simulation was later changed into a HEAD request not affecting the remaining count.*



- “Docker pull” environments
- Kubernetes clusters
- CI/CD pipelines
- “429 - too many requests” in app logs?

*Application presents from prices to 33% of customers ... because new release deployment failed because “docker pull” reached the hard limit?*





- SLI
  - Pull counts: Remaining, Limit
- SLO
  - Remaining < 10 (arbitrary number)
- Gates
  - SLO failed - don't start a deployment which again pulls images

**Developers need to know why CI/CD and reviews are blocked before the limit is reached.**

Mitigation: Use local registry & dependency proxy

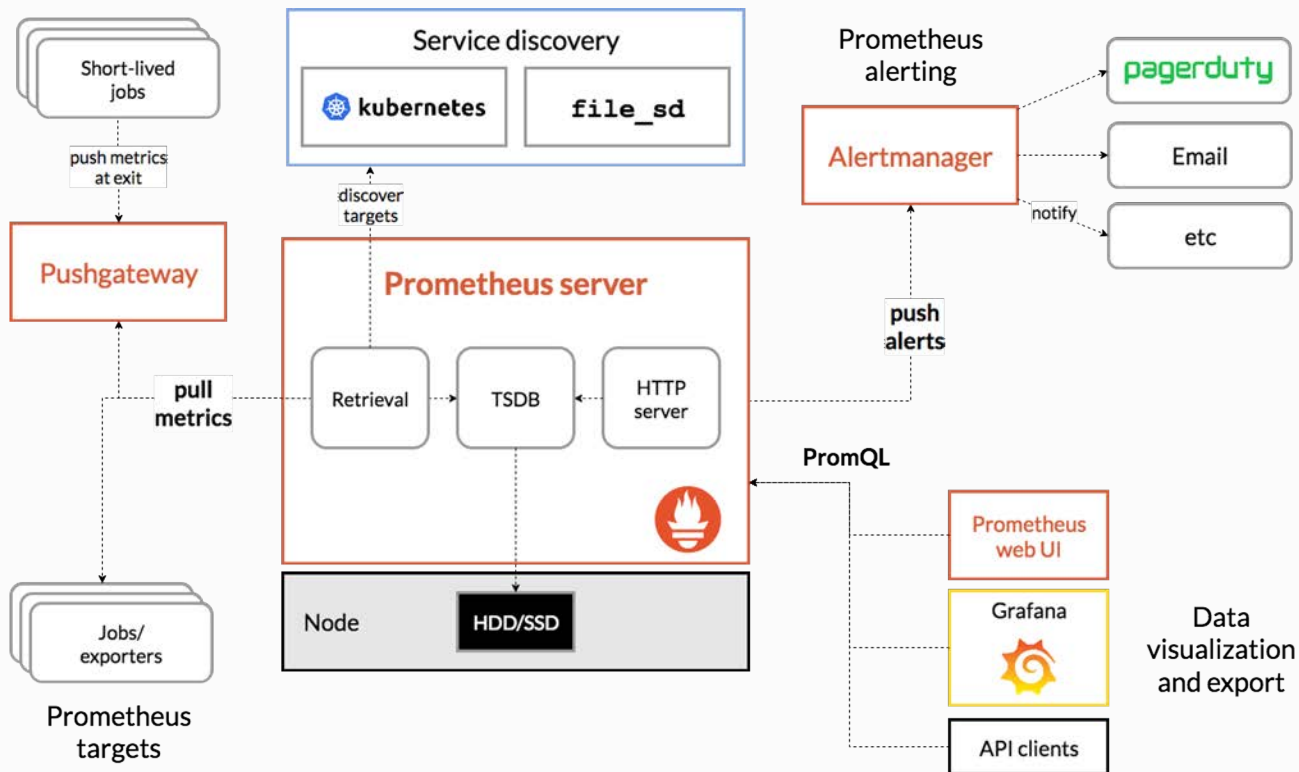




Go SLO



# Prometheus architecture



Picture from <https://prometheus.io/docs/introduction/overview/#architecture>





```
# Latest sample
metric_name

# Range
metric_name[5m]

# Labels
metric_name{label1="a",label2="b"}

# Functions
rate(metric_name[5m])
sum(metric_name)
delta(metric_name[5m])

# Comparisons
metric_name > 10*1024
```

# Nice, where do I start as developer?



- SLO
  - Monitoring
    - Metrics
      - key/tag
        - Values
          - ?





- Infrastructure
  - Memory, CPU, IO
  - Monitored on the node / pod / cluster
- Services
  - Prometheus Exporters
- Instrumentation of your apps

<https://prometheus.io/docs/instrumenting/exporters/>  
<https://training.promlabs.com/>





- Learn with playful examples
- Dockerfile
- CI/CD build image
  - Container registry

▼ app.py

```
...  ...  @@ -4,6 +4,7 @@ import time
4      4
5      5      # Create a metric to track time spent and requests made.
6      6      REQUEST_TIME = Summary('request_processing_seconds', 'Time spent processing request')
7      7  + WS_MICHI = Summary('ws_michi', 'Time spent processing request')
8      8
9      9      # Decorate function with metric.
10     10     @REQUEST_TIME.time()
...  ...  @@ -11,9 +12,15 @@ def process_request(t):
11     12         """A dummy function that takes some time."""
12     13         time.sleep(t)
13     14
15     15  + @WS_MICHI.time()
16     16  + def process_request_michi(t):
17     17  +     """A dummy function that takes some time."""
18     18  +     time.sleep(t)
19     19  +
14     20  if __name__ == '__main__':
15     21     # Start up the server to expose the metrics.
16     22     start_http_server(8000)
17     23     # Generate some requests.
18     24     while True:
19     25         process_request(random.random())
26     26  +     process_request_michi(random.random())
```

[https://github.com/prometheus/client\\_python#three-step-demo](https://github.com/prometheus/client_python#three-step-demo)

[https://gitlab.com/everyonecancontribute/observability/prometheus\\_python\\_service](https://gitlab.com/everyonecancontribute/observability/prometheus_python_service)



- Prometheus Operator
- ServiceMonitor CRD
- Inspect metrics
  - Prometheus
  - Grafana

```
$ git clone https://github.com/prometheus-operator/kube-prometheus
$ cd kube-prometheus

# 1. Create monitoring namespace and custom resource definitions

$ kubectl create -f manifests/setup

# 2. Wait until the ServiceMonitor CRD is available

$ until kubectl get servicemonitors --all-namespaces ; do date; sleep 1; echo ""; done

# 3. Apply remaining manifests

$ kubectl create -f manifests/
```

```
$ kubectl create -f ./manifests/ecc-python-service.yml
$ kubectl create -f ./manifests/ecc-python-service-monitor.yml
```

# Wait, there is more than metrics?



- Metrics
- Logs/Events
- Traces
- Profiling

Shifting from monolith to microservices.

Breath.

First, continue adding more metrics to your app.

<https://samnewman.io/books/monolith-to-microservices/>





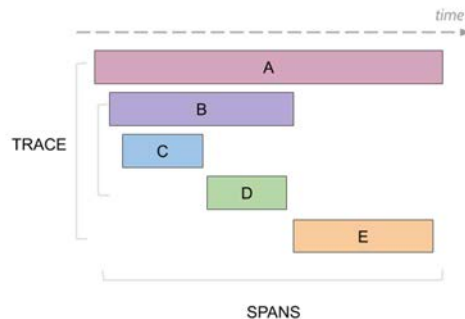
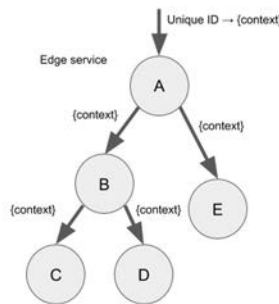
- Central log management
  - Evaluate all options for performance, maintenance, usability
- Elasticsearch
  - Beats as sidecars (DaemonSet) with auto-discovery
  - Elasticsearch as storage
  - Kibana as query frontend
- Loki
  - Promtail/fluentd as sidecar
  - S3 object storage
  - Grafana as frontend

<https://www.elastic.co/blog/monitoring-kubernetes-and-docker-containers-with-beats-logs-metrics-and-metadata>  
<https://grafana.com/docs/loki/latest/installation/helm/>  
<https://nws.netways.de/tutorials/logging-with-loki-and-grafana-in-kubernetes/>





- Different to logs
  - Spans with start/end time, context
  - Need app code additions
- Tools
  - Jaeger Tracing
  - Grafana Tempo
  - OpenTelemetry



<https://www.jaegertracing.io/docs/1.25/architecture/>

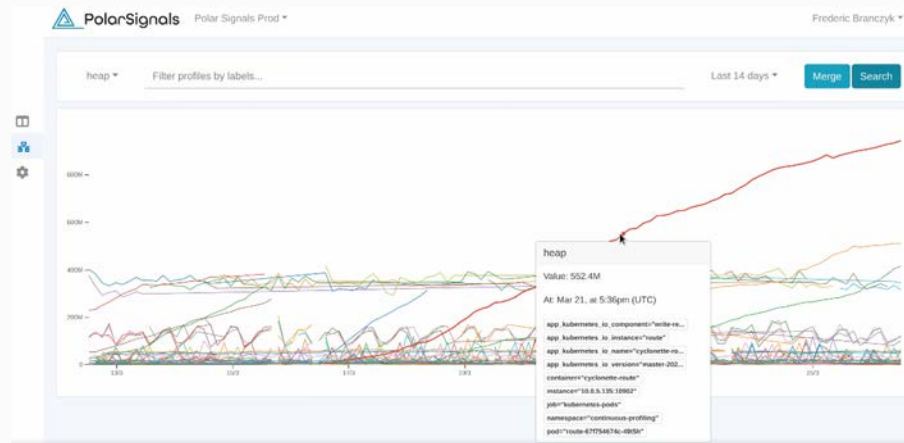
<https://github.com/grafana/tempo/tree/main/example/helm>

<https://github.com/open-telemetry/opentelemetry-operator>

<https://docs.google.com/presentation/d/1MAVFeSsTNVWC9wPGOlq83wh8GFtR9hPbdVHuamtgOWA/edit>



- Application Performance Insights
- Continuous Profiling
- — 3 4 pillars of Observability





# GitLab

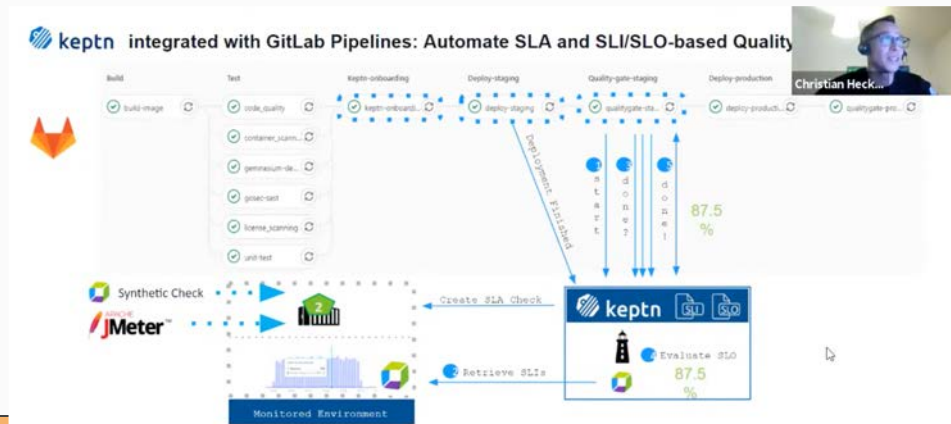
There was once a SLO that  
put to prod ...



# SLOs with Prometheus



- Metrics and Alert calculation
- PromQL queries in OpenSLO format
- Keptn uses SLOs for quality gates

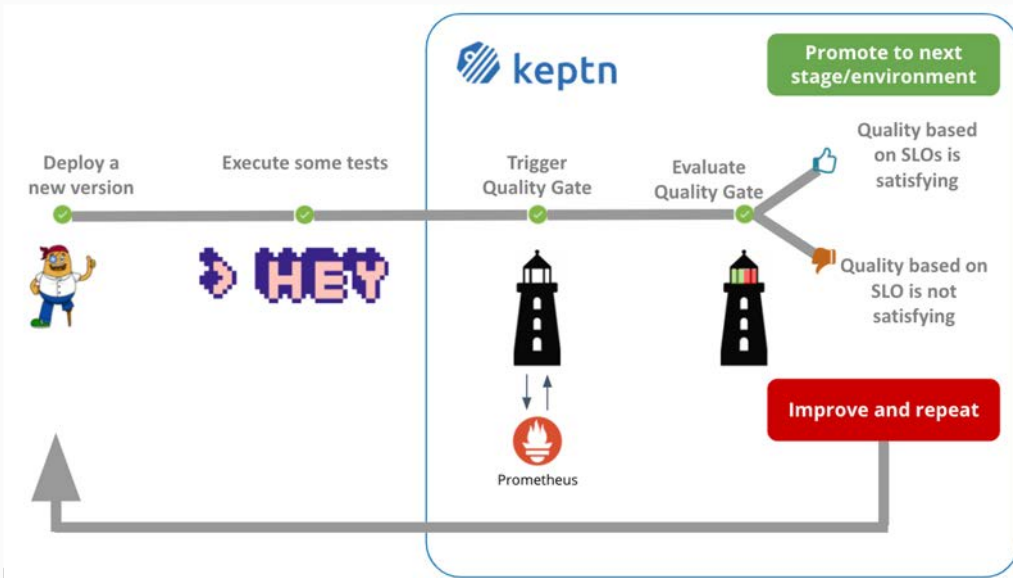


<https://everyonecontribute.com/post/2020-11-11-cafe-8-keptn/>

<https://gitlab.com/checkelmann/keptn-demo>

zoom

# Oh Keptn, my Keptn



<https://everyonecancontribute.com/post/2020-11-11-cafe-8-keptn/>  
[https://keptn.sh/docs/0.9.x/quality\\_gates/slo/](https://keptn.sh/docs/0.9.x/quality_gates/slo/)  
<https://tutorials.keptn.sh/tutorials/keptn-quality-gates-prometheus-08/#0>  
[https://www.youtube.com/watch?v=7ksL0V0tN\\_M](https://www.youtube.com/watch?v=7ksL0V0tN_M)

The screenshot shows the Keptn Prometheus-QG interface. The left sidebar lists 'Environment', 'Services', 'Sequences', and 'Integration'. The main content area displays a list of evaluation results for the 'helloservice' in the 'quality-gate' sequence. The results are as follows:

Sequence	Task	Status	Score	Started	Last task
evaluation failed	helloservice in quality-gate	failed	33	Started today at 11:55	get-sli
	evaluation succeeded	succeeded	100	Started today at 11:49	get-sli
	Configure monitoring succeeded	succeeded		Started today at 11:47	Configure monitoring
create succeeded	helloservice	succeeded		Started today at 11:47	create

The right sidebar shows the 'evaluation failed' details for the 'quality-gate' sequence, including the context 'eb1b18ad-2a22-4b9c-aa67-1e6b2e59e605' and the 'helloservice'.

Image credit:  
[Keptn.sh](https://keptn.sh)

# Keptn Service Level Indicator & Objective



```
---
spec_version: '0.1.0'
comparison:
  compare_with: "single_result"
  include_result_with_score: "pass"
  aggregate_function: avg
objectives:
  - sli: http_response_time_seconds_main_page_sum
    pass:
      - criteria:
          - "<=1"
      warning:
          - criteria:
              - "<=0.5"
    - sli: request_throughput
      pass:
          - criteria:
              - ">=-80%"
    - sli: go_routines
      pass:
          - criteria:
              - "<=100"
    - sli: response_time_p95
total_score:
  pass: "90%"
  warning: "75%"
```

<https://everyonecancontribute.com/post/2020-11-11-cafe-8-keptn/>  
[https://keptn.sh/docs/0.9.x/quality\\_gates/slo/](https://keptn.sh/docs/0.9.x/quality_gates/slo/)  
<https://tutorials.keptn.sh/tutorials/keptn-quality-gates-prometheus-08/#0>  
[https://www.youtube.com/watch?v=7ksL0V0tN\\_M](https://www.youtube.com/watch?v=7ksL0V0tN_M)

keptn / Choose project v

helloservice

Context: 94a86985-4146-4344-83b2-9ac854912385

Stage: quality-gate

Evaluation finished

2021-04-21 11:49

Heatmap Chart



Show SLO

Ignore for comparison

Evaluation of test on quality-gate

90 <= 100 Result: pass

Evaluation timeframe: 2021-04-21 11:47 - 2021-04-21 11:49 (2 minutes 0 seconds)



SLO breakdown

Name	Value	pass Criteria	warning Criteria	Result	Score
response_time_p95	0.004				0
http_response_time_seconds_main_page_sum	0	<=1	<=0.5	passed	33.33
request_throughput	7427	>=-80%		passed	33.33
go_routines	44	<=100		passed	33.33

Image credit:  
[Keptn.sh](https://keptn.sh)

# Keptn Demo Playground - async



 **keptn** / Choose project ▾ 


4 Projects

**litmus**  
1 Stages, 1 Services  
Shipyard version: 0.2.0  
<https://github.com/keptn-public-demo/litmus.git>

chaos


Recent sequences:

helioservice in  
chaos 100

 **delivery\_succeeded**


today at 11:00

helioservice in  
chaos 50

 **delivery\_failed**


today at 09:00

helioservice in  
chaos 100

 **delivery\_succeeded**


yesterday at 11:00

helioservice in  
chaos 0

 **delivery\_failed**

yesterday at 09:00

helioservice in  
chaos 100

 **delivery\_succeeded**


last Monday at 11:00

**podtatohead**  
2 Stages, 1 Services  
Shipyard version: 0.2.0  
<https://github.com/keptn-public-demo/podtatohead.git>

hardening production


Recent sequences:

helioservice in  
production --

 **delivery\_succeeded**


today at 15:00

helioservice in  
production --

 **delivery\_succeeded**


today at 12:00

helioservice in  
production --

 **delivery\_succeeded**


yesterday at 12:00

helioservice in  
production --

 **delivery\_succeeded**

last Monday at 15:00

helioservice in  
production --

 **delivery\_succeeded**

last Monday at 12:00

**sockshop**  
3 Stages, 2 Services  
Shipyard version: 0.2.0  
<https://github.com/keptn-public-demo/sockshop.git>

dev staging production

Recent sequences:

cars in  
staging 0

 **delivery\_failed**

today at 13:00

cars in  
staging 0

 **delivery\_failed**

today at 10:00

cars in  
staging 0

 **delivery\_failed**

yesterday at 17:00

cars in  
staging 0

 **delivery\_failed**

yesterday at 10:16

cars in  
staging 0

 **delivery\_failed**

yesterday at 10:00

**unleash**  
1 Stages, 2 Services  
Shipyard version: 0.2.0  
<https://github.com/keptn-public-demo/unleash.git>

production

Recent sequences:

unleash in  
production --

 **delivery\_succeeded**

2021-03-09 17:18

<https://keptn.public.demo.keptn.sh/bridge/dashboard>

Image credit:  
[Keptn.sh](https://keptn.sh)





Continuous Delivery with

Keptn as quality gate

Prometheus for SLOs

Apps will always run “as is”, simulating production is hard.

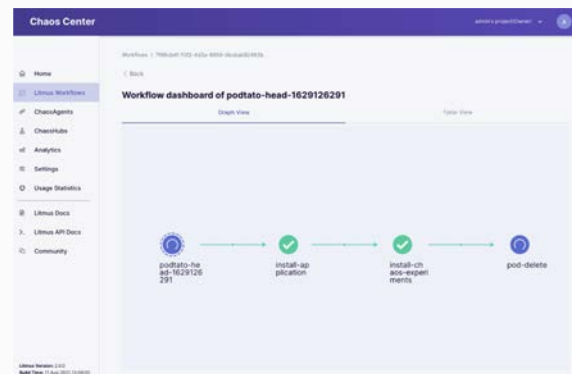
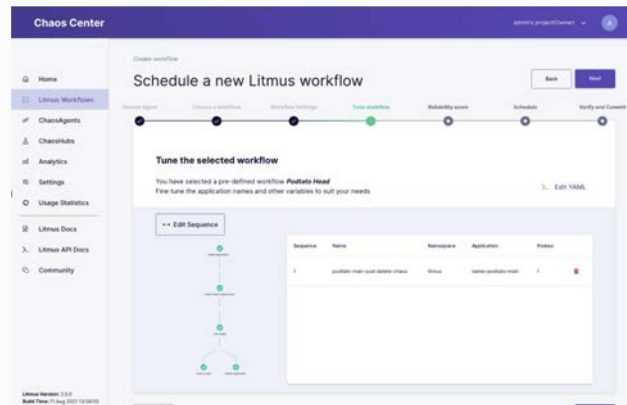
Add Chaos to your deployments 💡



# Shift Left with Chaos



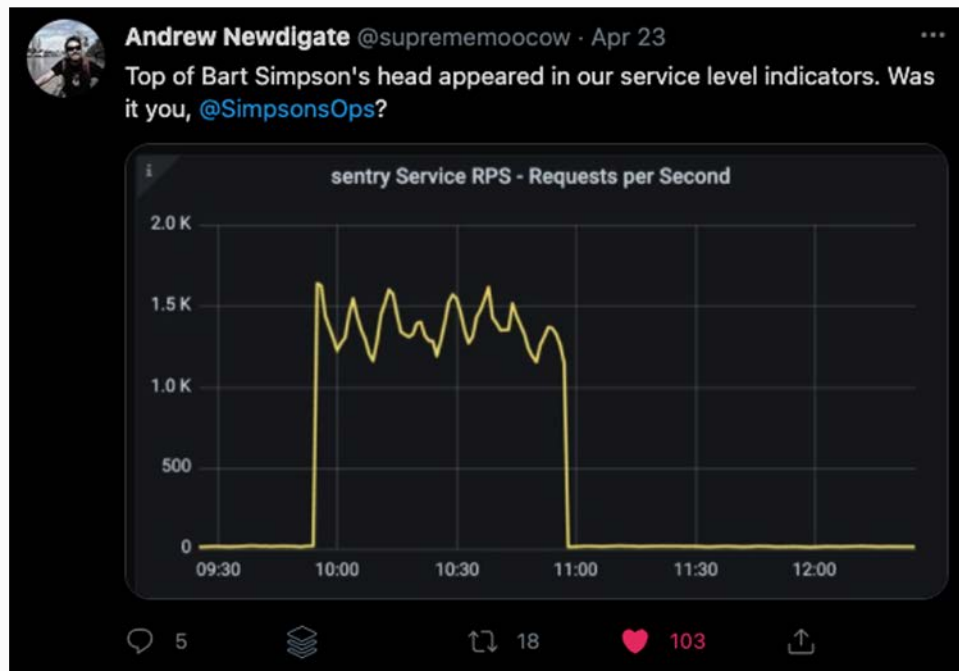
- Litmus Chaos
- Fail your infrastructure
- See how the app behaves
- See if SLOs still match
- Define actions & improvements



<https://twitter.com/dnsmichi/status/1427287749065953291>  
<https://docs.litmuschaos.io/docs/getting-started/run-your-first-workflow/>



- Automate everything
- Generate SLOs from DSLs





- Define Metrics Catalog
- Generate alerts and dashboards
  - Key metrics
  - Thresholds
  - Description + Observability tool URLs



How We are Dealing with Metrics at Scale on GitLab.com - Andrew Newdigate, GitLab

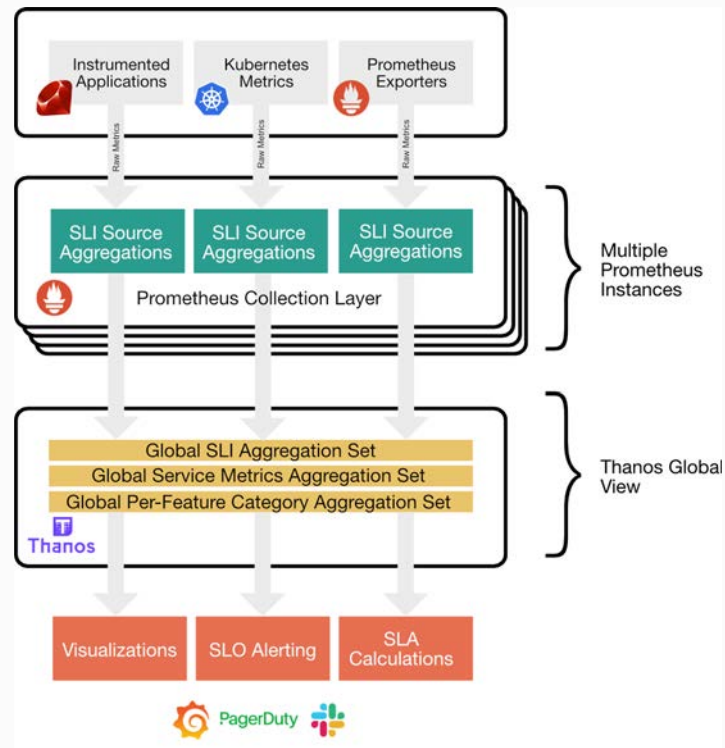
<https://www.youtube.com/watch?v=6sfr2IGJQXk>

<https://about.gitlab.com/handbook/engineering/monitoring/>

<https://dashboards.gitlab.net/d/web-main/web-overview?from=now-1h&orgId=1&to=now>

<https://gitlab.com/gitlab-com/runbooks/-/tree/master/metrics-catalog>

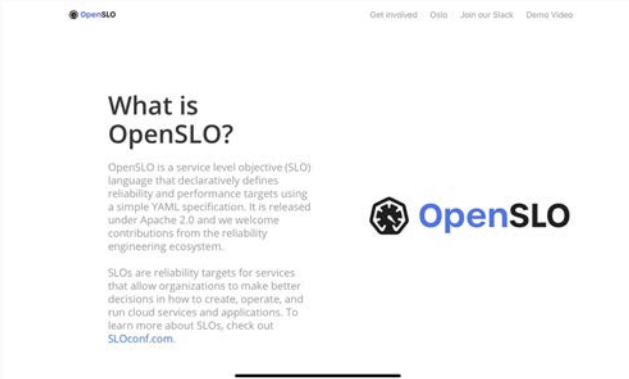
<https://gitlab.com/gitlab-com/gl-infra/gitlab-monitoring/-/tree/main/gitlab-monitoring>



# SLOs as code



- SLO spec - OpenSLO
- SLO generators - Sloth
- SLO management - Pyrra




OpenSLO

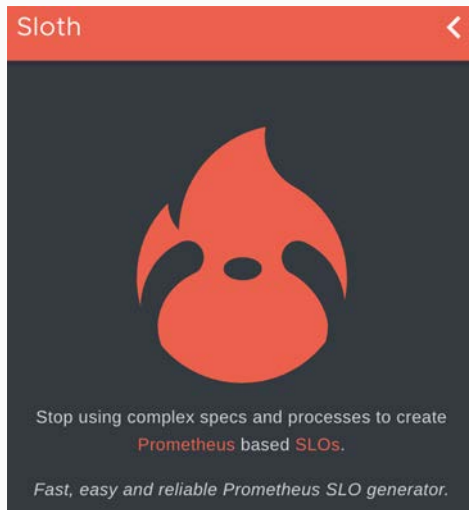
Get involved · Oslo · Join our Slack · Demo Video

## What is OpenSLO?


OpenSLO is a service level objective (SLO) language that declaratively defines reliability and performance targets using a simple YAML specification. It is released under Apache 2.0 and we welcome contributions from the reliability engineering ecosystem.



SLOs are reliability targets for services that allow organizations to make better decisions in how to create, operate, and run cloud services and applications. To learn more about SLOs, check out [SLOconf.com](https://sloconf.com).



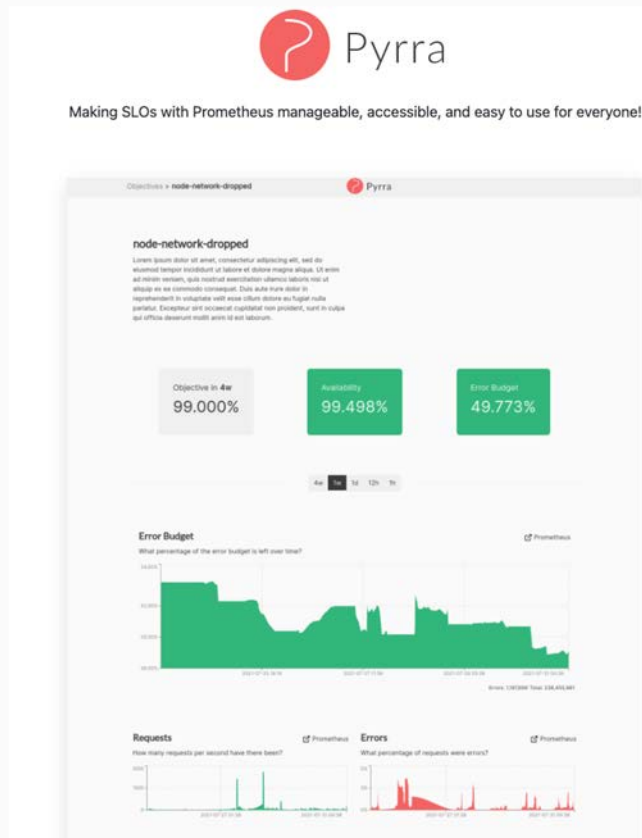
Sloth



Stop using complex specs and processes to create  
**Prometheus based SLOs.**

*Fast, easy and reliable Prometheus SLO generator.*

<https://openslo.com/>  
<https://sloth.dev/>  
<https://pyrra.dev/>



Pyrra

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

Objectives > node-network-dropped

### node-network-dropped


Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Objective in 4w	Availability	Error Budget
99.000%	99.498%	49.773%

4w 1w 1d 1h 1m

### Error Budget

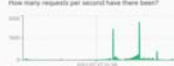
What percentage of the error budget is left over time?



Errors: 1/17/2020 Total: 230,203,001

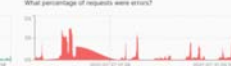
### Requests

How many requests per second have there been?



### Errors

What percentage of requests were errors?





# GitLab

Shift Left Possibilities





- Developers get SLO feedback
- No co-routines because memory corruption
- Re-work login algorithms
- Test infrastructure changes just like code
- Add code quality checks from lessons learned







- Correlate SLOs with
  - Instrumentation
  - Observability
- Make SLOs part of your DevSecOps workflows
- Use dynamic resource environment benefits
  - Auto-scaling VMs never has been easier
  - Add chaos into your container clusters





- Application insights for non-Devs
  - Metrics, logs, traces
  - Help everyone not familiar with your code
    - SREs, Ops & DevOps engineers
  - Is the problem code or something else?
- Use boring solutions
  - Start with metrics - /metrics HTTP endpoint
  - Add structured logging
  - Ops with a dashboard
  - Prometheus, Keptn, Litmus



# Don't do everything yourself



- Not every written test makes sense
- 100% test coverage is impossible
- Bring chaos & data into the DevOps lifecycle
  - Security scanning
  - Fuzz testing
  - Chaos Engineering
- Practice Observability methods
  - SLOs for quality gates in CI/CD
  - Embrace the unknown unknowns
- Learn, document & educate





- CI/CD Tracing - pipelines & runners with executors
- ML to curate correlating metrics and traces

DevSecOps becomes DevSLOSecOps.

Just kidding. 😏





- Observability in your CI/CD pipeline
- Quality gates with Keptn + Prometheus
- Infrastructure as Code, GitOps and Monitoring
  - Prometheus
  - Terraform
  - Deployment & Upgrades
- Chaos Engineering with Litmus
- Continuous Profiling



- Prometheus: Up & Running: <https://www.oreilly.com/library/view/prometheus-up/9781492034131/>
- PromLabs Trainings: <https://training.promlabs.com/>
- Keptn SLOs: [https://keptn.sh/docs/0.9.x/quality\\_gates/slo/](https://keptn.sh/docs/0.9.x/quality_gates/slo/)
- Litmus Chaos: <https://docs.litmuschaos.io/>
- 100 Days of Kubernetes: <https://100daysofkubernetes.io/observability/prometheus-exporter.html>
- Achieving Observability in async workflows - Netflix Tech Blog <https://netflixtechblog.com/achieving-observability-in-async-workflows-cd89b923c784>
- #EveryoneCanContribute cafe
  - <https://everyonecancontribute.com/post/2021-05-19-cafe-30-kubernetes-monitoring-prometheus/>
  - <https://everyonecancontribute.com/post/2020-11-11-cafe-8-keptn/>
- [Monitoring Kubernetes with Prometheus and Grafana workshop](#) 💡
- KubeCon EU 2021: Putting Chaos into Continuous Delivery to Increase Application Resiliency  
[https://www.youtube.com/watch?v=\\_DgCc4-BLW8](https://www.youtube.com/watch?v=_DgCc4-BLW8)



Thank you!



<https://dnsmichi.at/about/>

<https://www.linkedin.com/in/dnsmichi/>

<https://twitter.com/dnsmichi>

<https://www.polywork.com/dnsmichi>