From Monitoring to Observability: Left Shift your SLOs

Michael Friedrich
Senior Developer Evangelist

@dnsmichip
GitLab

A DevOps Tale
Security shifted left

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

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

- 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://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

https://sysdig.com/blog/golden-signals-kubernetes/
https://sre.google/books/

Image credit: Denise Yu
Story time:
Rocket Science with C++
Devs building software

- Slow REST & JSON-RPC API responses
- CPU overload with threads
- Let's use light-weight co-coroutines 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
Devs need to go SLO

- 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%

TOST 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.

Mitigation

- 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
  ○ login_time < 2m
  ○ login_time < 5m on low latency connections
● Quality gates = CI/CD failure = stop release
Story time:
It’s always DNS
DNS is hard

- 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
Slack down - analysis

- 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
Slack down - ideas

- 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)
Slack is not alone - DNSSEC with .at

- 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?
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
Facebook incident 2021-10-04

- 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.

https://twitter.com/dnsmichi/status/1445310247934087171
https://twitter.com/dnsmichi/status/1445468075357138953?s=21
https://twitter.com/dnsmichi/status/1445146654361989122
Facebook - SLOs?

- 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
Story time: Docker Hub Rate Limiting
Rate limits for image pulls

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

- 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.*

https://about.gitlab.com/blog/2020/10/30/mitigating-the-impact-of-docker-hub-pull-requests-limits/
Unknown state

- “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?

https://www.honeycomb.io/blog/so-you-want-to-build-an-observability-tool/
https://docs.gitlab.com/ee/user/project/clusters/kubernetes_pod_logs.html
Service Level Indicator and Objectives for Docker limits

- **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.
Prometheus architecture

Prometheus server:
- Retrieval
- TSDB
- HTTP server

Prometheus targets:
- Jobs / exporters
- Short-lived jobs

Pushgateway:
- Push metrics at exit
- Pull metrics

Service discovery:
- kubernetes
- file_sd

Prometheus alerting:
- pagerduty
- Email
- etc

PromQL

Alertmanager

Node
HDD/SSD

Prometheus web UI
Grafana
API clients

Data visualization and export

Picture from https://prometheus.io/docs/introduction/overview/#architecture
Prometheus Terminology: PromQL

# 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

https://prometheus.io/docs/prometheus/latest/querying/examples/
https://promlabs.com/promql-cheat-sheet/
https://demo.promlens.com/
Nice, where do I start as developer?

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

- **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

https://github.com/prometheus/client_python#three-step-demo
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

https://github.com/prometheus/client_python#three-step-demo
https://gitlab.com/everyonecancontribute/observability/prometheus_python_service
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://grafana.com/docs/loki/latest/installation/helm/
Traces

- 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/1MAVFeSsTNVWC9wpGOIg83wh8GFlR9hPbdVHuumtgOWA/edit
Profiling

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

[Links]
https://www.polarsignals.com/
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://everyonecancontribute.com/post/2020-11-11-cafe-8-keptn/
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://tutorials.keptn.sh/tutorials/keptn-quality-gates-prometheus-08/#0
https://www.youtube.com/watch?v=7ksL0V0tN_M
Keptn Service Level Indicator & Objective

---

```yaml
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: 1
    criteria: <=1
  - sli: request_throughput
    pass: >=80%
  - sli: go_routines
    pass: <=100
  - sli: response_time_p95
    total_score: 1938
    warning: 75%
```

https://everyonecancontribute.com/post/2020-11-11-cafe-8-keptn/
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

Image credit: Keptn.sh
Keptn Demo Playground - async

https://keptn.public.demo.keptn.sh/bridge/dashboard
Shift Left with Quality Gates & SLOs

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/
More SLOs ideas

- Automate everything
- Generate SLOs from DSLs

https://twitter.com/suprememoocow/status/1385571793600229388
GitLab.com SaaS insights

- 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
SLOs as code

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

https://openslo.com/
https://sloth.dev/
https://pyrra.dev/
SLO down on SLO unmatched

- 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
Go SLO

● 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
See the value in metrics, logs, traces

- 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
Wishlist - yes, still

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

DevSecOps becomes DevSLOSecOps.

Just kidding. 😬
Recap

- 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
Resources

- PromLabs Trainings: https://training.promlabs.com/
- Keptn SLOs: 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
- #EveryoneCanContribute cafe
- 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
Thank you!

https://dnsmichi.at/about/
https://www.linkedin.com/in/dnsmichi/
https://twitter.com/dnsmichi
https://www.polywork.com/dnsmichi