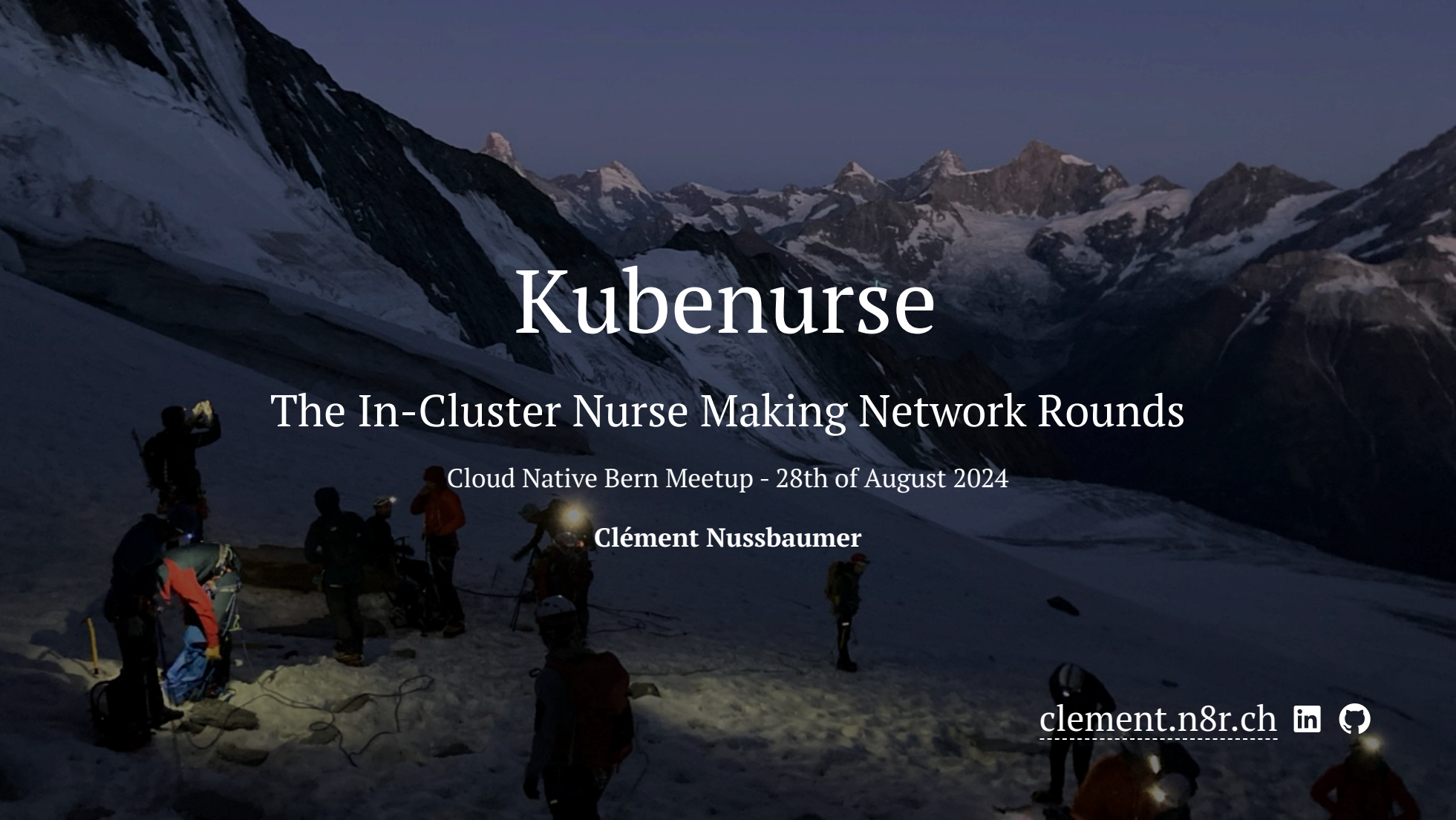


Kubenurse

The In-Cluster Nurse Making Network Rounds

Cloud Native Bern Meetup - 28th of August 2024

Clément Nussbaumer

clement.n8r.ch

Kubernetes @ PostFinance

30 Vanilla v1.29 Kubernetes

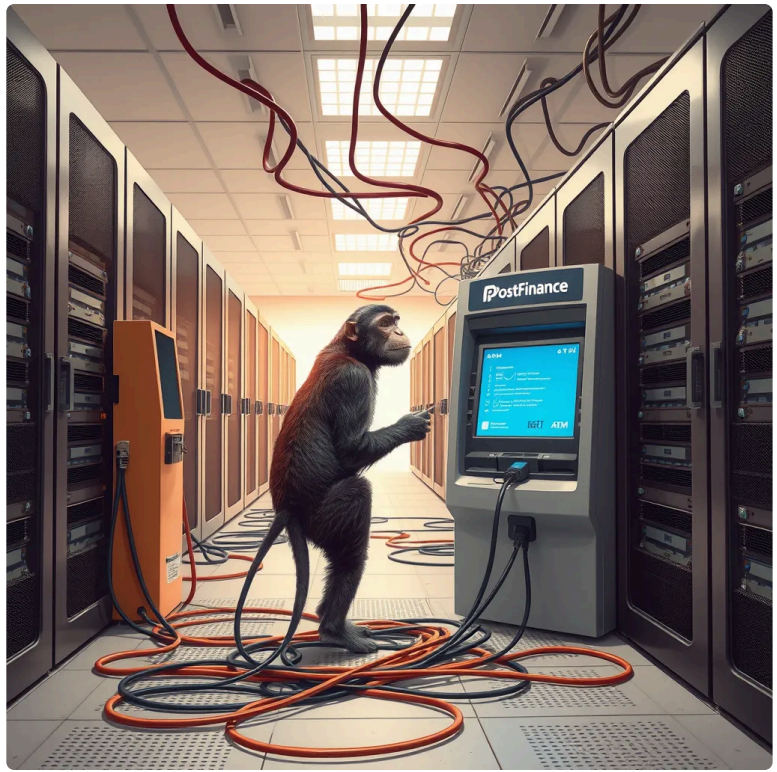
Oldest cluster `AGE 5y57d`

450 nodes (15 with GPU)

Namespace self-service with inhouse operator

ArgoCD and GitOps for almost all deployments

Chaos Monkey on all clusters 🐒

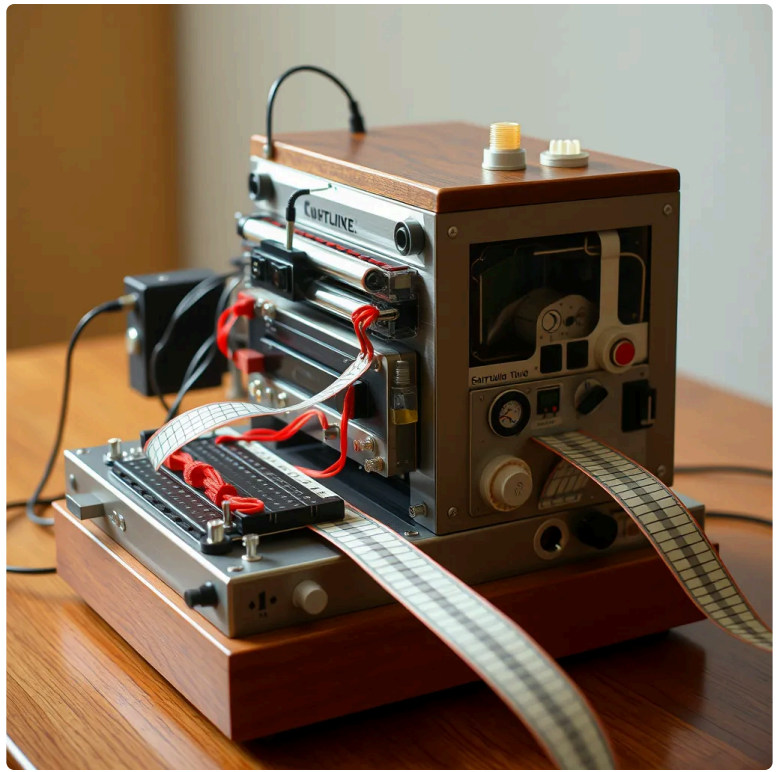


Created with FLUX.1 [dev] by Black Forest Labs

Computers are random access machines

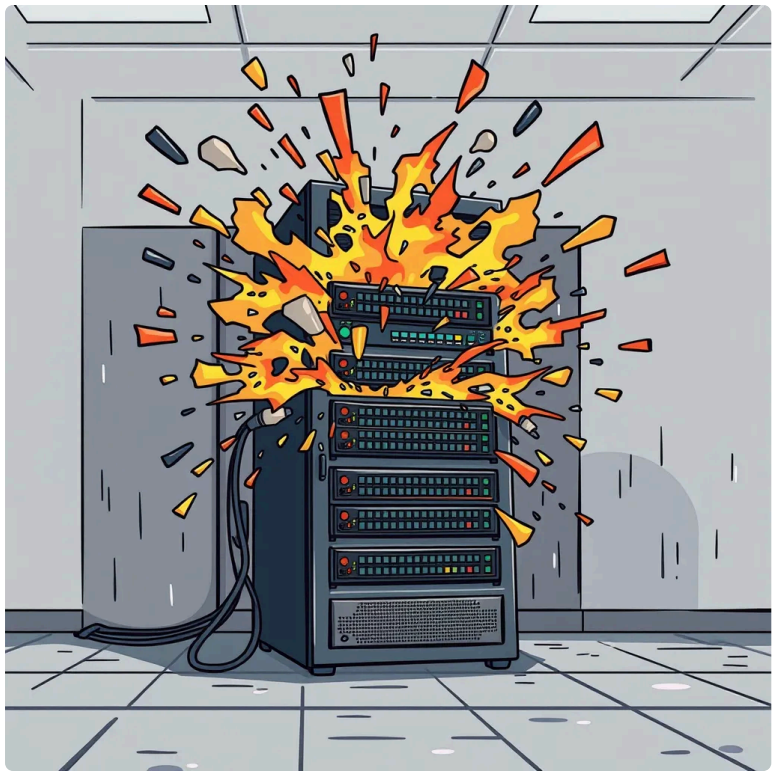
Definition: A model of computation whose memory consists of an unbounded sequence of registers, each of which may hold an integer. In this model, arithmetic operations are allowed to compute the address of a memory register.

Algorithms and Theory of Computation Handbook, CRC Press LLC, 1999, "random access machine", in Dictionary of Algorithms and Data Structures , Paul E. Black, ed. 17 December 2004. Available from: https://www.nist.gov/dads/HTML/randomaccess.html



Created with FLUX.1 [schnell] by Black Forest Labs

Computers are **random** ~~access machines~~



Created with FLUX.1 [schnell] by Black Forest Labs

> Definition: Computers always suprise us in random ways.
> Some examples: power failure, network links instability, disk failure, library incompatibilities, CPU overload, network bottlenecks, DDoS attacks, DNS problems (always), file system corruption, ...
>
> *Clément Nussbaumer, 2024 (and the list keeps getting longer)*

Kubenurse

Created with FLUX.1 [dev] by Black Forest Labs

The In-Cluster Nurse Making Network Rounds

Kubenurse

Oct 4th 2018
first OSS commit

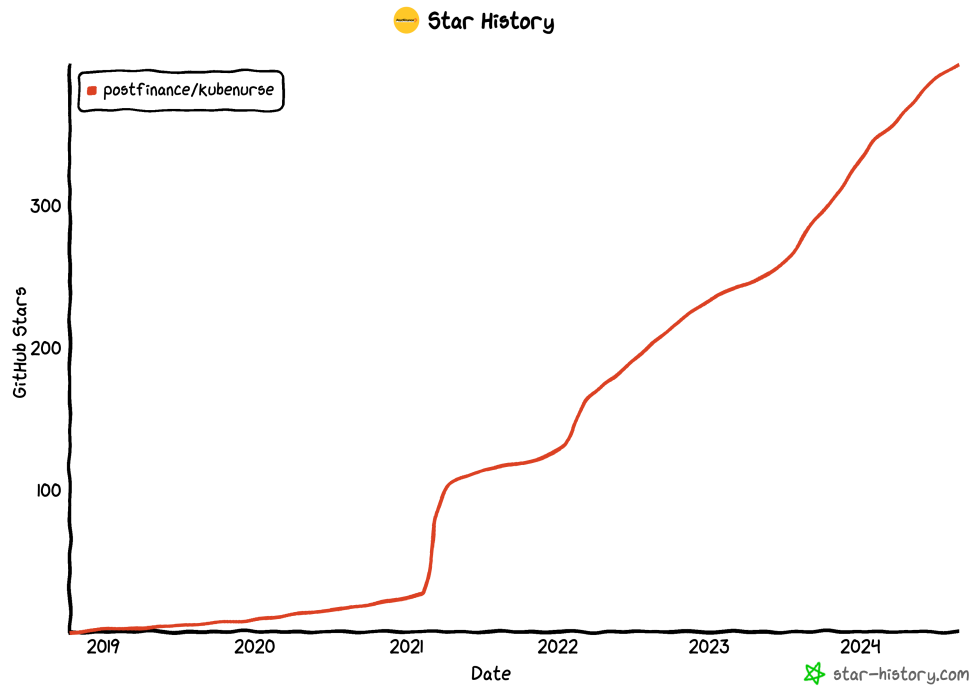
Feb 6th 2020
first external PR

Sep 16th 2021
helm chart PR

Nov 22nd 2022
histogram buckets

Mar 15th 2024
node discovery
at scale

Jul 23rd 2024
parallel checks



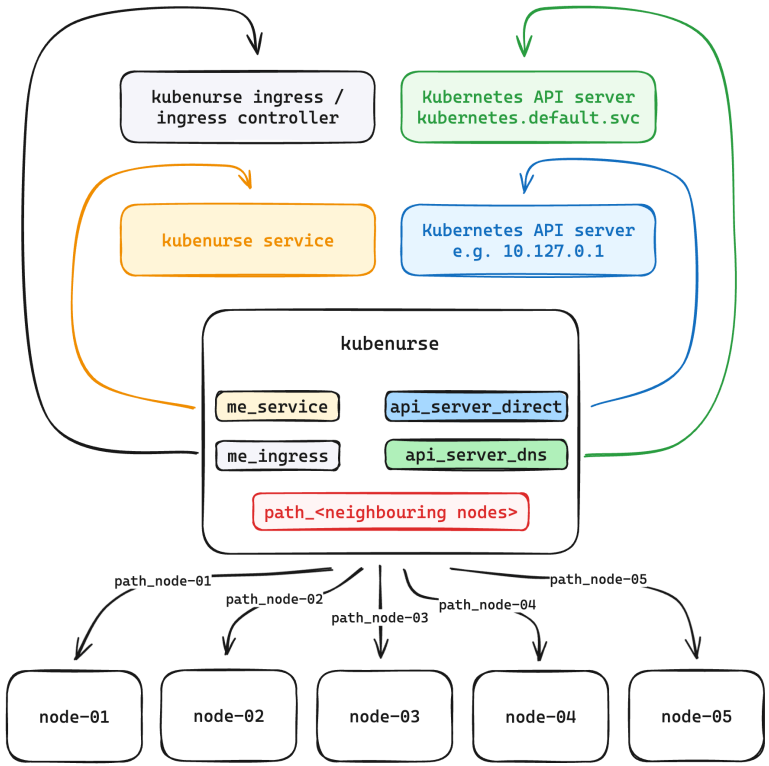
Kubenurse

```
❯ kubectl get daemonset kubenurse
NAME        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kubenurse   6         6         6       6            6           <none>          35h

❯ kubectl top pod
NAME              CPU(cores)   MEMORY(bytes)
kubenurse-74gd4   34m          38Mi
kubenurse-gfb7x   31m          38Mi
kubenurse-h86h4   38m          38Mi
kubenurse-ptj8n   35m          42Mi
kubenurse-rv66b   31m          32Mi
kubenurse-snpmz   39m          38Mi

❯ kubectl get pods -o yaml | yq '.items[0].spec.containers[].env'
- name: KUBENURSE_HISTOGRAM_BUCKETS
  value: .0005,.001,.0025,.005,.01,.025,.05,0.1,0.25,0.5,1
- name: KUBENURSE_CHECK_INTERVAL
  value: 0.5s
- name: KUBENURSE_REUSE_CONNECTIONS
  value: "false"
- ...
```

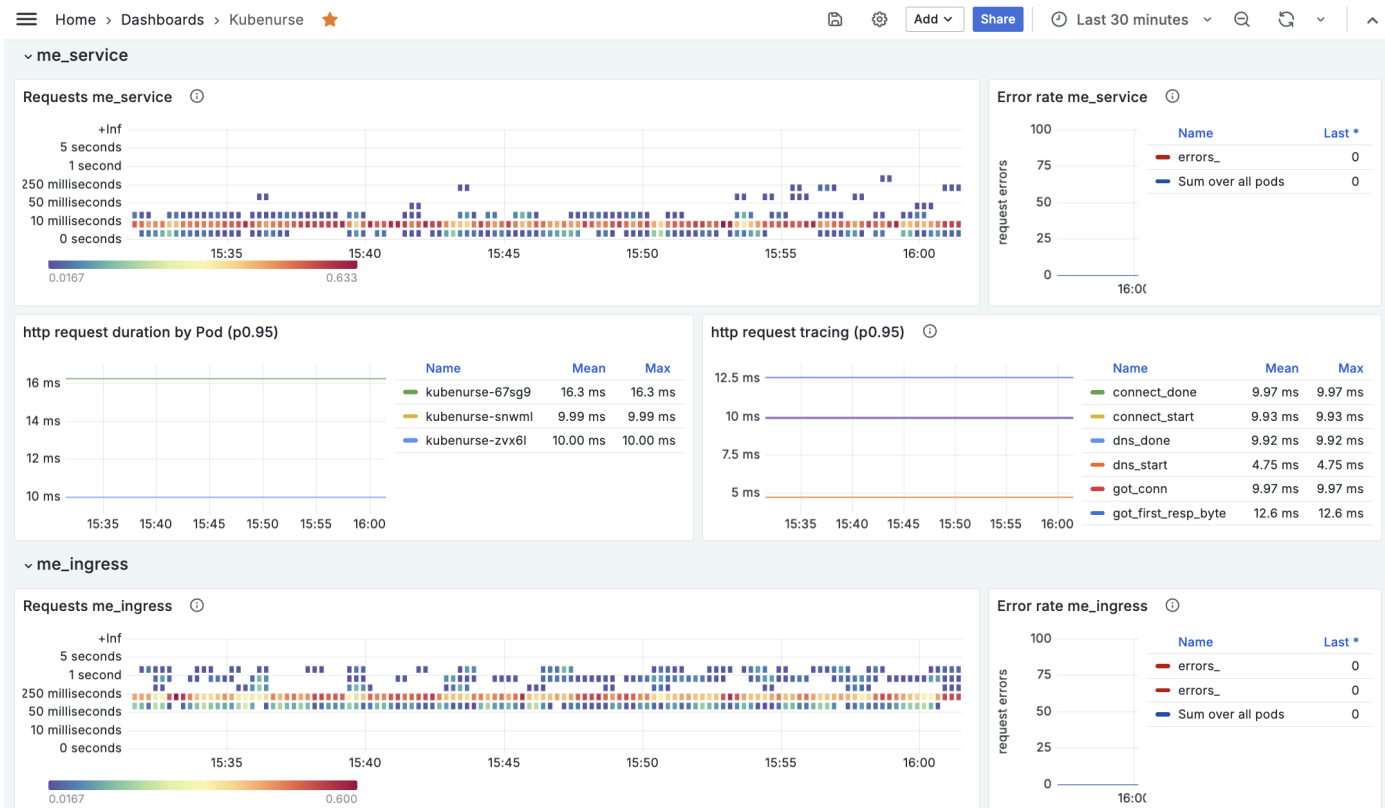
Kubenurse request types



Kubenurse metrics

metric name	labels	description
`kubenurse httpclient requests total`	`type`, `code`	counter for the total number of http requests, partitioned by HTTP code and request type
`kubenurse errors total`	`type`, `event`	error counter, partitioned by httptrace event and request type
`kubenurse httpclient request duration seconds`	`type`	latency histogram for request duration, partitioned by request type

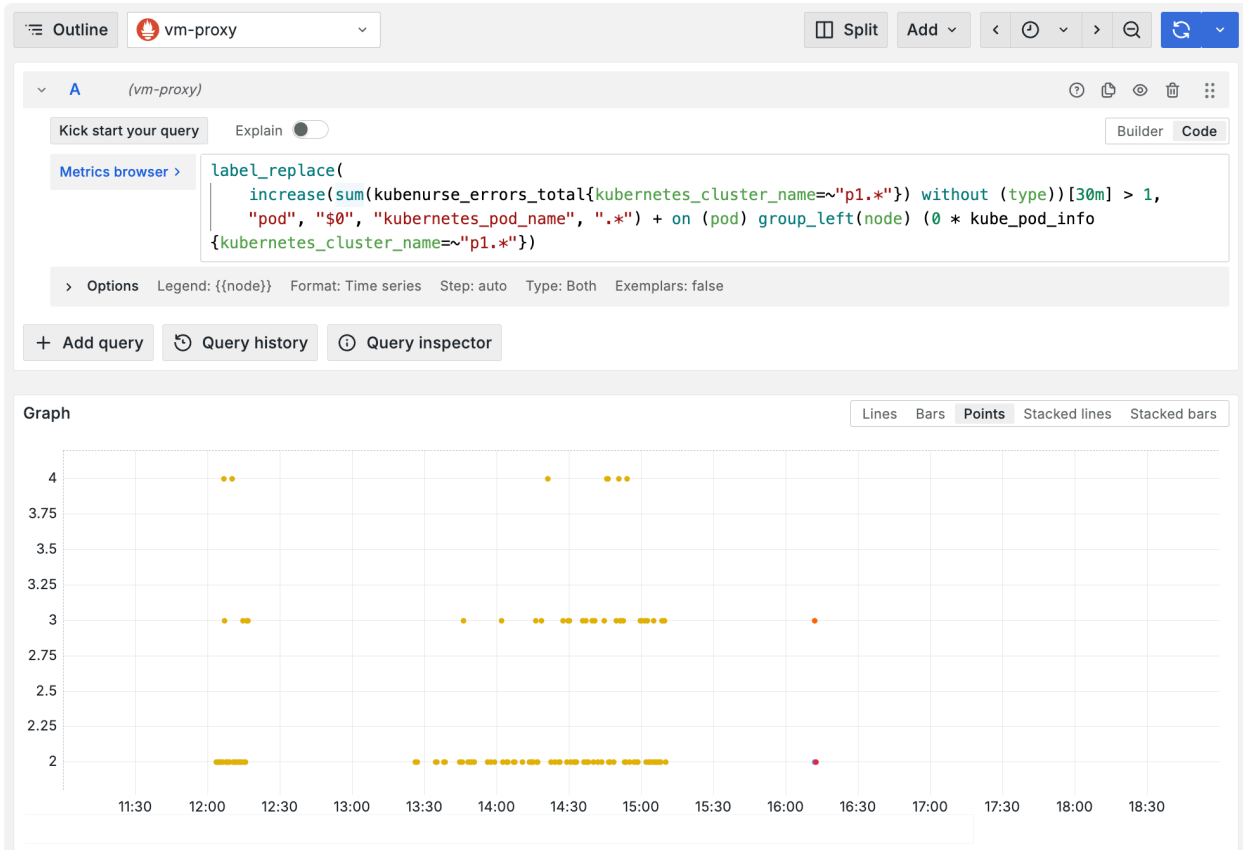
Kubenurse dashboard



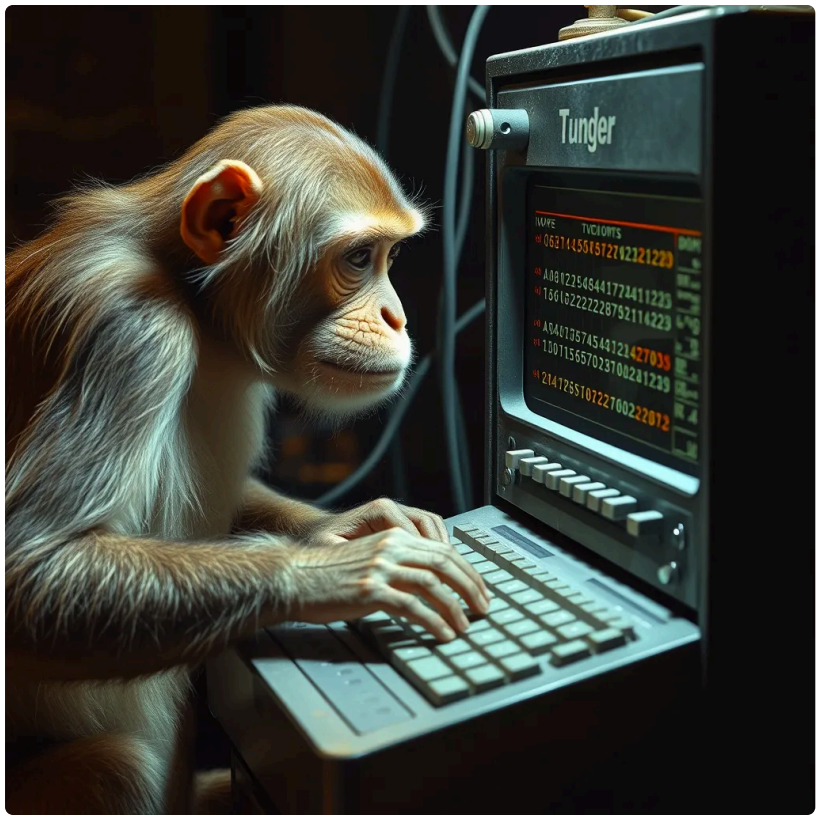
Real-world example

→ how do you notice **random** packet drops ?

→ how do you notice a slight latency increase ?



Demo



Created with FLUX.1 [dev] by Black Forest Labs

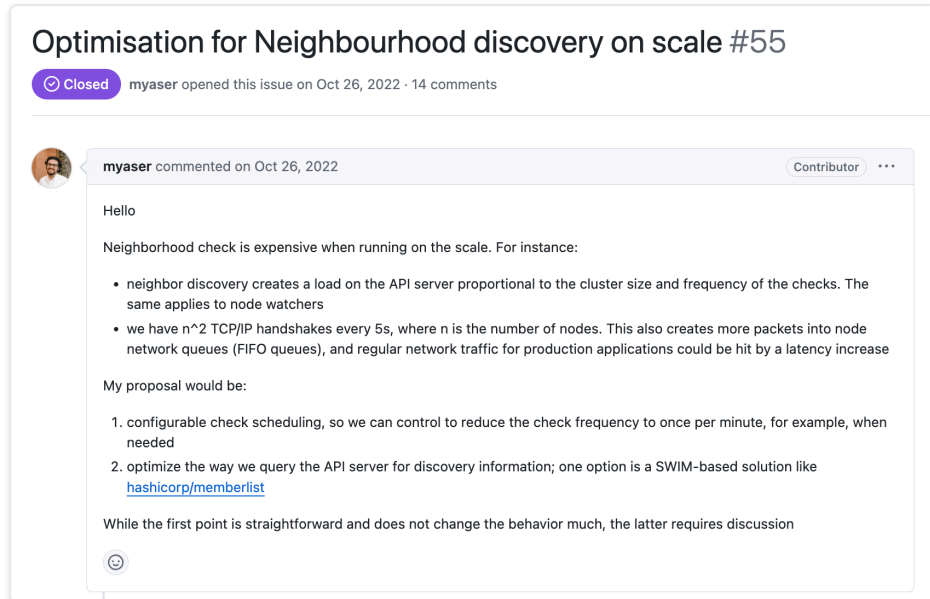
Installing Kubenurse

Node discovery at scale

GitHub issue #55

mostly 2 problems:

1. no caching
2. $O(n^2)$ neighbouring checks



Optimisation for Neighbourhood discovery on scale #55

Closed **myaser** opened this issue on Oct 26, 2022 · 14 comments

myaser commented on Oct 26, 2022 Contributor

Hello

Neighborhood check is expensive when running on the scale. For instance:

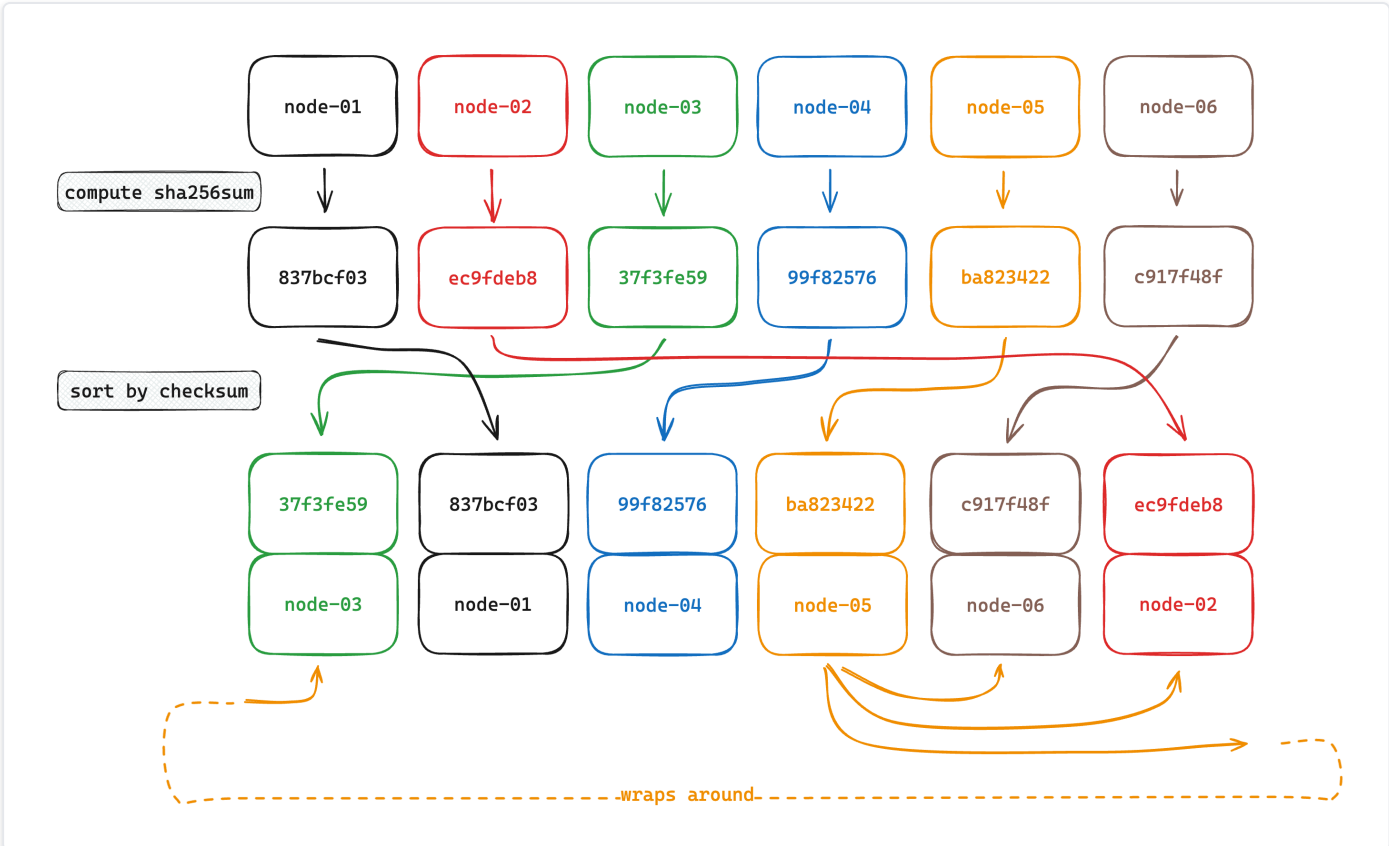
- neighbor discovery creates a load on the API server proportional to the cluster size and frequency of the checks. The same applies to node watchers
- we have n^2 TCP/IP handshakes every 5s, where n is the number of nodes. This also creates more packets into node network queues (FIFO queues), and regular network traffic for production applications could be hit by a latency increase

My proposal would be:

1. configurable check scheduling, so we can control to reduce the check frequency to once per minute, for example, when needed
2. optimize the way we query the API server for discovery information; one option is a SWIM-based solution like hashicorp/memberlist

While the first point is straightforward and does not change the behavior much, the latter requires discussion

Node discovery at scale



measure randomness

clement.n8r.ch

Created with FLUX.1 [dev] by Black Forest Labs