

Designing an Internal SLO Platform

Domains first, a small stack, and honest lessons from production

Julio Leal · Software Engineer · Nubank

Salt Lake City · November 9, 2026

Who we are

139M
customers

4
countries
Brazil · Mexico · Colombia · US

#1
digital bank
in Latin America

- Thousands of microservices, on Kubernetes, in AWS
- A Clojure company: our config files are EDN

Source: Nu Holdings, Q2 2026 results (quarter ended June 30, 2026)

The metrics platform in numbers

58M
samples per second
about 5 trillion a day

2B
active time series

30T
data points stored
8 days · ~300 TB on disk

600K+
alert rules
most run every minute

30K
queries per second
90% from alert rules

800K+
scrape targets

VictoriaMetrics · 35+ clusters · 8 accounts · 7 regions · measured in October 2026

"Is our credit card healthy right now?"

- The credit card is how Nubank started. Millions of our customers use it.
- A business leader asks a simple question.
- We had thousands of services and no single answer.
- Each team measured "healthy" in its own way.
- Feature work usually won over reliability work.

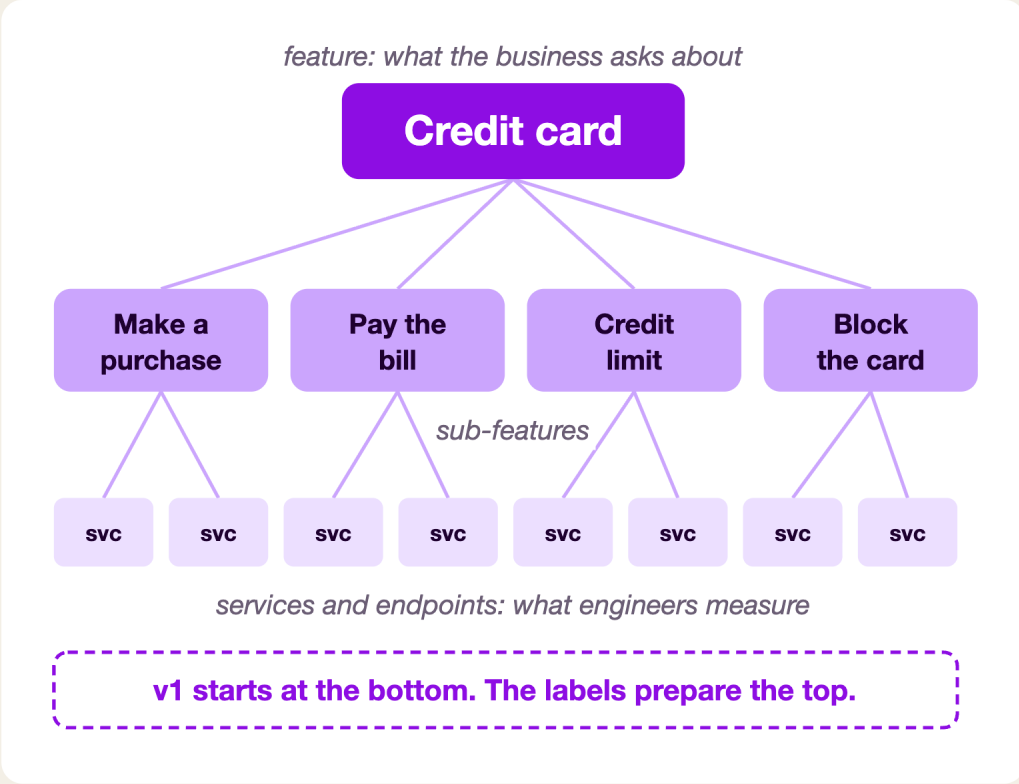
Nobody argues against SLOs. Few teams really want to use them to make decisions.

Where we started

- Raw metrics live for only 8 days. Our SLOs use a 30-day window.
- Thousands of services, 8 production accounts
- One team, one semester for v1

The dream: a pyramid of SLOs, from services up to features.

January 2026: first RFC · August 2026: production launch



Build, buy, or compose?

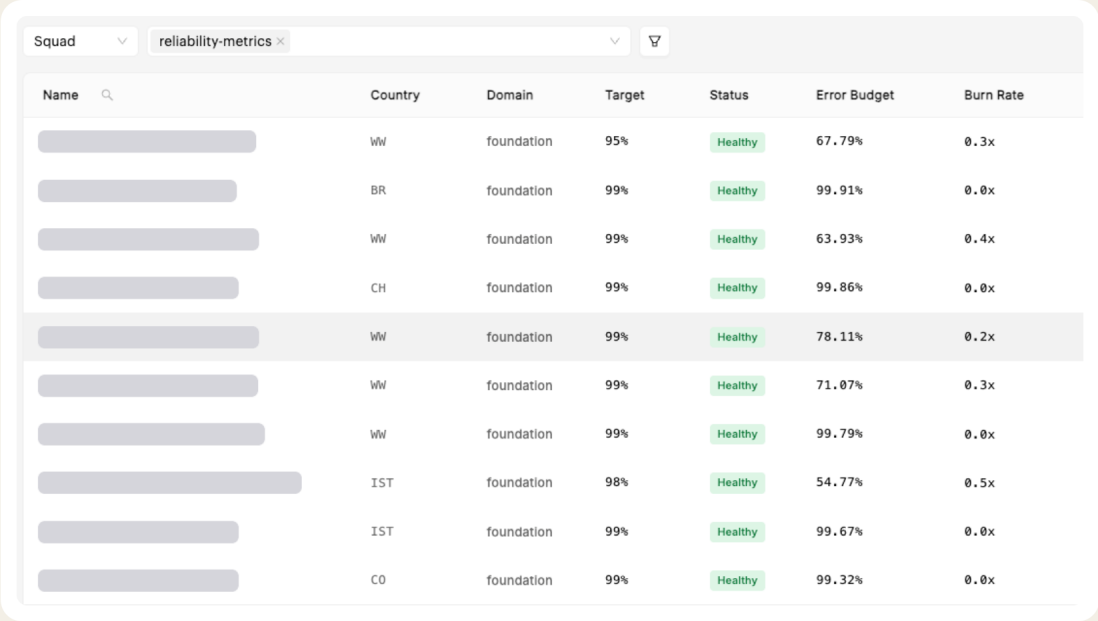
Option	What we found
Buy a hosted SLO product	A big monthly bill. Our data leaves our cloud.
Build from zero	We rebuild what open source already does.
Compose: our pick	Reuse VictoriaMetrics, Grafana, and our config pipeline. Add an open-source engine.

The engine: Sloth or slo-generator (Google)?

- Sloth generates the rules once. Prometheus does the math. slo-generator is a service that computes budgets at runtime.
- We chose slo-generator: many backends, many exporters, our metric names.
- The price: one more service to run, in Python.

Decision 1: Organize by domain, not by service

- The data comes from services. We organize it by domain.
- Every SLO has an owner: a squad and a domain, from the service catalog.
- Find SLOs by domain, subdomain, or squad.
- An SLO is a business promise. Services are the parts.




```
sli_measurement{slo_name="card-purchase-availability", squad="card-team",
                domain="credit-card", subdomain="purchases"}   0.9994   # example
```

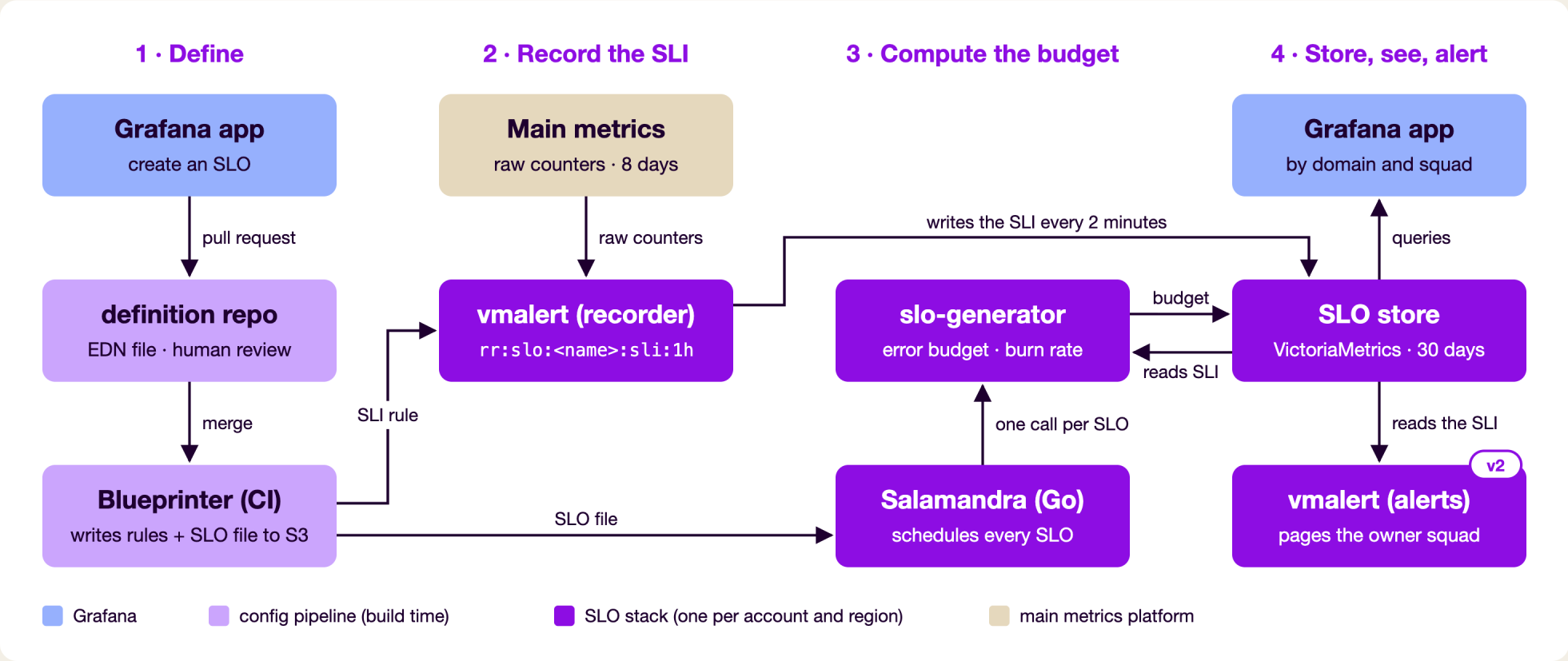
Decision 2: SLOs as code, with an escape hatch

- An SLO is a file in Git. A human reviews it.
- Standard SLI: pick 1 of 7 templates. Set the goal.
- Code editor: write your own PromQL.
- Write the file yourself, or let our Grafana app write it. Both open a pull request.

```
;; services/my-service.edn
{:name   :my-service
 :squad  :my-squad
 :tags   {:business-domain    "credit-card"
          :business-subdomain "purchases"}
 :slos?  true
 :slos   {:http-availability-slo
          {:goal "99.9%"}}}
```

Templates: availability · error rate · latency p95 and p99 · throughput · CPU · memory

How it works



One SLI rule per SLO. All five budget windows (1h, 6h, 3d, 15d, 30d) read the same rule.

Decision 3: A small, isolated SLO stack

- Own VictoriaMetrics stack in each of the 8 production accounts
- Keeps SLO data 30 days. Raw metrics keep 8.
- Today, 70+ SLOs from 12 squads use about 10,000 series. The main platform has 2 billion.
- A heavy query on the main platform cannot hide your SLO.
- We also run a copy for our own platform (dogfood).

```
rr:slo:<name>:sli:1h                          # good / valid over 1h, every 2 min
avg_over_time(rr:slo:<name>:sli:1h[30d])      # every window reads the same rule
```

What v1 left out, and what came next

Left out of v1	What happened
Burn-rate alerts	The first thing users asked for. Shipped in v2.
Multi-region SLOs	Needed as the US goes active/active in two regions. Shipped in v2.
History beyond 30 days	Still missing. One team keeps a spreadsheet.
Domain-level SLOs	Not yet. They need good and valid counts.
User-made templates	Not needed so far.

"Our SLOs work as a dashboard and not as a signal." · one of our teams, before v2

v2: Why not slo-generator's alerts?

	slo-generator alert: true	Our generated vmalert rules
Pages a person?	No. It sets a flag in a report.	Yes, through Alertmanager
Long and short window together?	No. One window at a time.	Yes, in every tier
A real 5-minute window?	No. It reads the 1-hour SLI.	Yes, from its own rule
Fresh?	Only when slo-generator runs	On every rule evaluation
Maintained?	No release since February 2024	Already runs our 600K+ alert rules

VictoriaMetrics is lock-in too. But it is the same tool, the same routing, and the same on-call as every other alert.

v2: Burn-rate alerts

- Blueprinter generates them with the SLI rule. Every new SLO gets alerts for free.
- A second vmalert reads the SLO store. One vmalert has only one datasource.
- The squad label on the SLI series sends each alert to the SLO owner.
- critical pages someone. warning goes to Slack. A second type: no data.

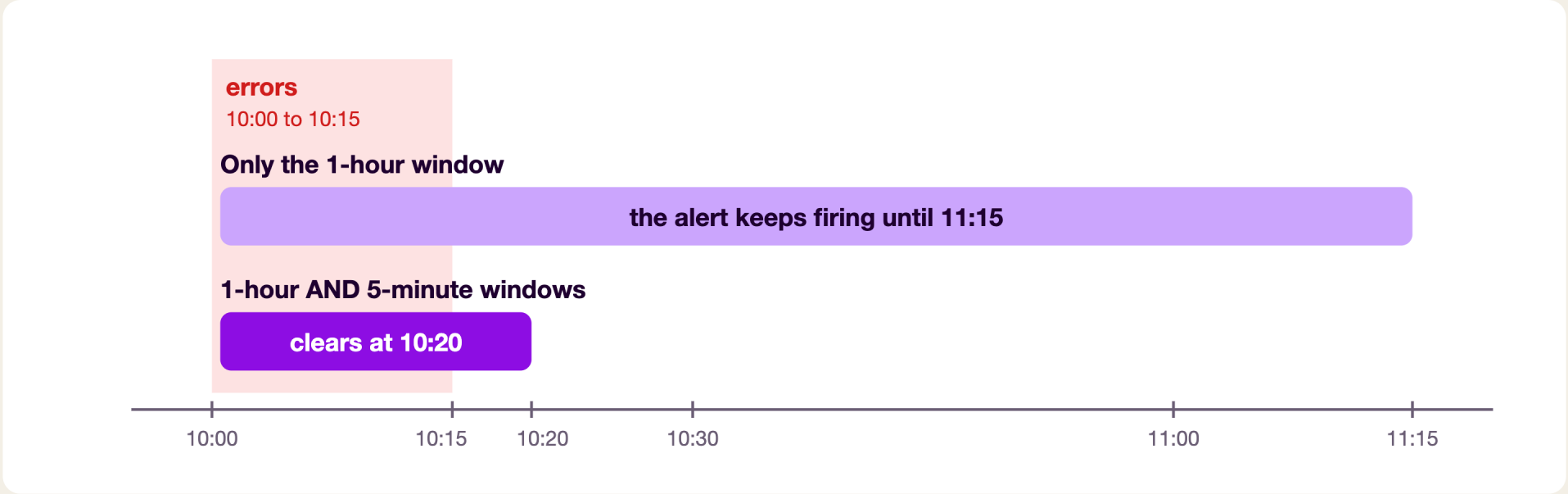
```
# The 1-hour tier of a 99.9% SLO. Fire only when BOTH windows burn.
(1 - rr:slo:<name>:sli:1h) > 14.4 * (1 - 0.999)
and
(1 - rr:slo:<name>:sli:5m) > 14.4 * (1 - 0.999)
```

v2: What does 14.4 mean?

1. SLO 99.9% over 30 days → error budget: 0.1% of requests can fail.
2. Burn rate 1 → the budget lasts exactly 30 days.
3. 30 days = 720 hours. Page when 1 hour burns 2% of the budget: 0.02 × 720 = 14.4
4. At 14.4, a 99.9% SLO has 1.44% errors. The budget is gone in about 2 days.

Severity	Long · short window	Burn rate	Budget spent	Budget gone in
critical: page	1h · 5m	14.4	2% in 1 hour	about 2 days
critical: page	6h · 30m	6	5% in 6 hours	5 days
warning: Slack	3d · 6h (also 15d, 30d)	1	10% in 3 days	30 days

v2: Why two windows?



- Long window: no page for a short blip.
- Short window: the alert clears fast after the fix.
- Its own rule: rr:slo:<name>:sli:5m. An average of the 1-hour SLI is too slow.

v2: One SLO across two regions

- The US is going active/active in two AWS regions.
- Customers feel one service: one realm SLO per service.
- Never average percentages. Sum the events, then divide.

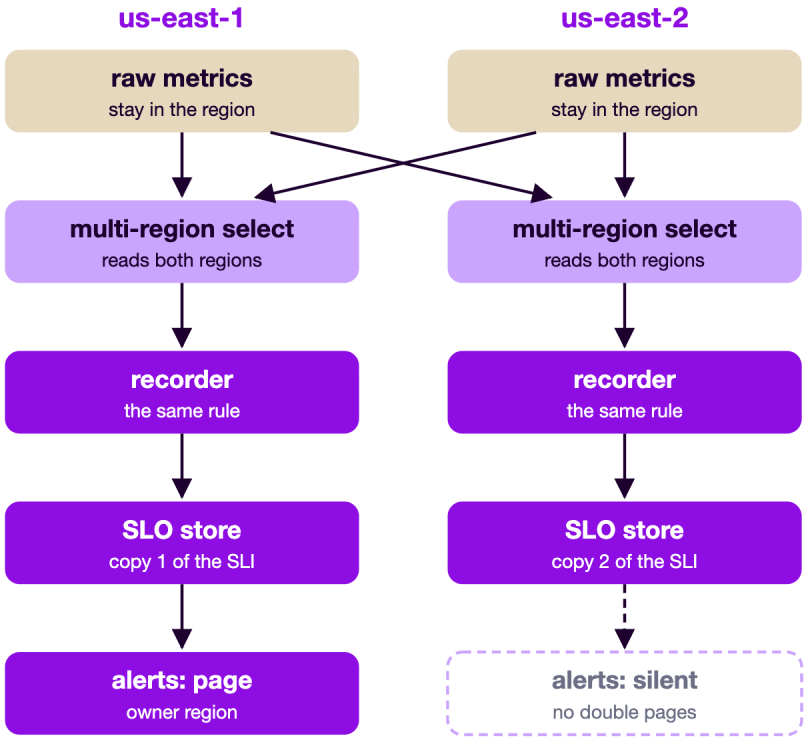
	good	valid	SLI
us-east-1	900	1,000	90%
us-east-2	50	100	50%
average: wrong			70%
total: right	950	1,100	86.4%

v2: How multi-region works

- Raw metrics stay in their region.
- Each recorder reads both regions.
- Each region keeps a copy of the small SLI.
- Copies are cheap. Extra queries are the real cost.
- Only the owner region sends alerts.

Fail open: after failover, the healthy region is what customers feel.

The price: missing metrics can look healthy.



Raw metrics are never copied. Only the small SLI exists twice.

Lesson 1: Design for the decision

June, before the launch: we showed the beta to one internal team.

- Most liked: the prediction. If nothing changes, will we miss the SLO this month?
- Also liked: a warning state, between healthy and violated.
- Least liked: the hourly view. One bad hour barely matters in a week.
- The data was there: five windows, up to 30 days. The beta had no way to pick one.
- They asked for: alerts. Burn-rate math is hard to do by hand.

People don't need more data. They need to know what to do next.

Lesson 2: Silence is not health

- No data looks like “no errors”. A burn-rate alert stays quiet.
- So we added a liveness alert: “this SLI has no data for 1 hour”.
- It found 3 SLOs that never computed.
- v2: the no-data alert reads the SLO store.

An incomplete number can give you a false sense of safety.

Lesson 3: Adoption is earned, not forced

- September: a team with its own stability index asked: is ours the official SLO platform?
- Yes. Like our logs, metrics, and traces platforms.
- But no mandate. A tech mandate is the last option.
- Instead: the August engineering monthly, leadership meetings, and sponsors in different areas.
- October: one team sent us 11 SLOs to review at once, with a design doc.
- Today: 70+ SLOs from 12 squads. Still early.

Nobody uses a platform because you tell them to. They use it because it helps them.

What we would do differently

1

Keep good and valid counts, not only the percentage.

Today: one percentage per hour. Fine for one service. But a percentage forgets its traffic. The pyramid needs:

```
sum(good) / sum(valid)
```

2

Ship a signal with v1.

An alert. A dashboard does not make decisions.

A v1 checklist for your SLO platform

Do in v1

- SLOs as code, reviewed in a PR
- Templates, plus an escape hatch
- An owner on every SLO
- Good and valid counts, stable labels
- Isolated storage, longer retention
- A CI dry-run for every generated rule
- A liveness check for every SLI
- One signal that reaches a human

Can wait

- More than one metrics store
- Custom windows
- Long history
- User-made templates



An SLO platform is not done when SLOs exist. It is done when someone acts on them.

Thank you! Questions?

Julio Leal · Nubank

github.com/juliocleal13 · linkedin.com/in/juliocleal13