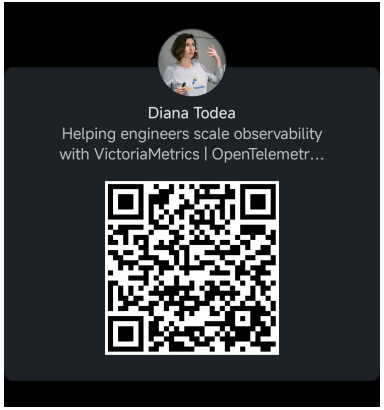
Green Observability: what needs to shuffle in open source?

SCaLE 23x | Cloud Native Days LA - March 7th 2026

Diana Todea - DevRel Engineer

MERGE FORWARD

- OpenTelemetry member and contributor
- Cloud Native Days Romania organizer
- Co-lead CNCF Merge-Forward Neurodiversity



https://github.com/didiViking/Conferences_Talks

The Cost of Observability is huge

- Observability budgets commonly sit around ~15–25% of an infra bill
- Cloud logging/storage costs are explicit and can be tens of cents per GiB
- Adding AI / LLMs increases telemetry significantly

Why Sustainability matters in Observability?

- More telemetry = more servers = more energy = more carbon footprint

Build Green Software from the Inside Out

https://greensoftware.foundation/manifesto

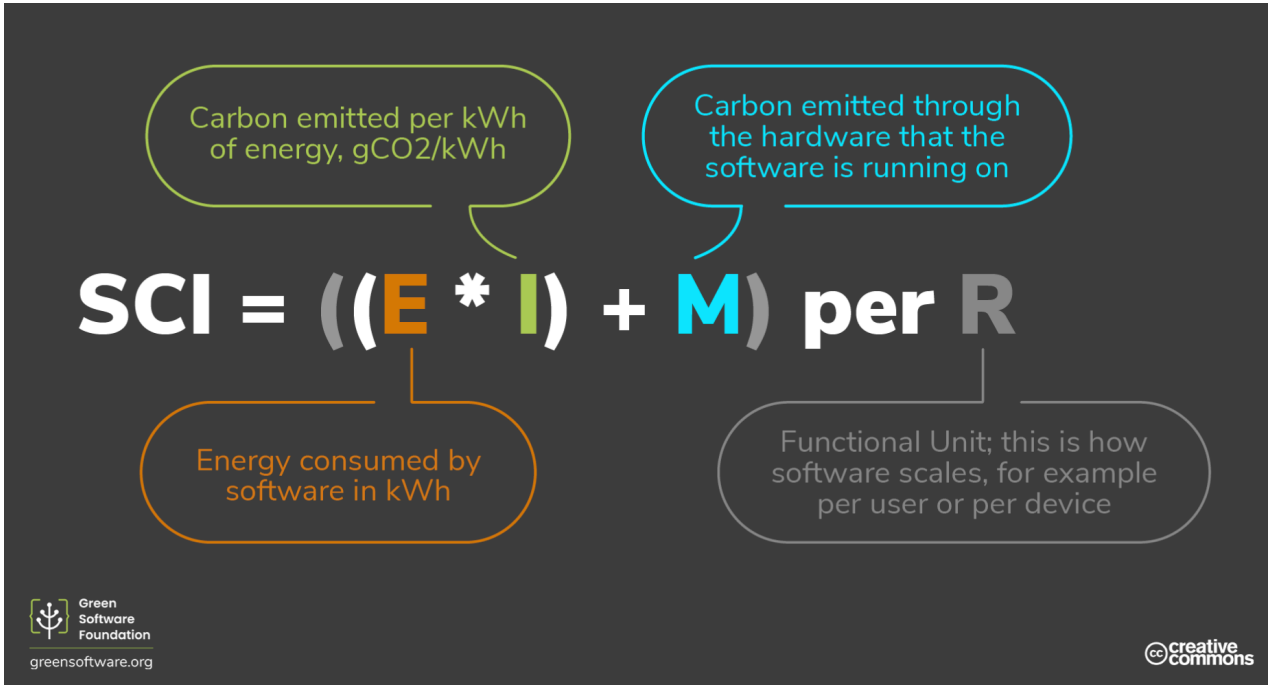
MINIMISE CARBON

The Foundation's mission is to reduce the total change in global carbon emissions associated with software. When evaluating choices we choose the option that advocates for abatement (reducing emissions) not neutralisation (offsetting emissions).

Operationalise:

- We will consider how to minimise carbon emissions in every decision we make around how we conduct ourselves operationally and the standards and technology we create and use.


GHG protocol
ISO 14064
ISO 14067


Carbon emitted per kWh of energy, gCO2/kWh
Carbon emitted through the hardware that the software is running on
SCI = ((E * I) + M) per R
Energy consumed by software in kWh
Functional Unit; this is how software scales, for example per user or per device
Green Software Foundation
greensoftware.org
creative commons

The sustainability paradox

The observability sustainability paradox

1. Prometheus

- ☑ Default scrape configs collect everything
- ☑ High-cardinality labels

```
pod, request_id, user_id, session_id
```

- ☑ Same retention for all metrics

Prometheus - collect less

- ☑ Keep only metrics you actually use
- ☑ Drop high-cardinality labels early
- ☑ Reduce:
  - Time series count
  - Memory usage
  - Query CPU

```
scrape_configs:

- job_name: "kubernetes-pods"

  kubernetes_sd_configs:

  - role: pod

  metric_relabel_configs:

  - source_labels: [__name__]

    regex: "http_request_duration_seconds_bucket"

    action: keep

  - source_labels: [pod]

    action: labeldrop
```

🌱 Why this is green observability? 🌱

- Fewer active series → less RAM
- Smaller TSDB → less disk IO
- Faster queries → less compute time
- Same SLO visibility, lower footprint

2. Loki

- ☑ Logging everything at INFO
- ☑ Large unstructured messages
- ☑ Long default retention
- ☑ Logs are often the largest storage consumer

Drop low-value logs with Loki pipeline stages

- Drop debug logs before indexing
- Keep only actionable log levels
- Reduce ingestion and index size

```
pipeline_stages:

- match:

    selector: '{app="checkout"}'

    stages:

    - drop:

        expression: "level=\"debug\""
```

🌱 Why is this green observability? 🌱

- Less log volume → less disk
- Smaller index → faster queries
- Lower ingestion rate → less CPU
- Engineers still find errors fast

3. Fluent bit

- ☑ Logs shipped raw: includes noisy, low-value logs
- ☑ Filtering happens too late
- ☑ Wasteful network & storage usage

Drop noisy logs at the edge

- ☑ Filter, drop, normalize logs before shipping
- ☑ Drop health-check noise
- ☑ Control label and cardinality early
- ☑ Reduce:
  - ◆ Network traffic
  - ◆ Storage costs

```
[FILTER]
    Name    grep
    Match   kube.*
    Exclude level DEBUG
```

```
[FILTER]
    Name    grep
    Match   kube.*
    Exclude message /healthz/
```

🌱 Why this is green observability? 🌱

- Early filtering = biggest impact
- Smaller payloads = less energy per request
- Cleaner logs = better signal-to-noise

4. OpenTelemetry

- ☑ If you sample 100%:
- Every request creates spans
- CPU used for span processing
- Memory overhead
- Storage explosion
- Most traces never viewed

Solution 1.

- ☑ In the OTel Collector config use head based sampling

```
processors:
  probabilistic_sampler:
    sampling_percentage: 10
```

🌱 Why this is green observability? 🌱

- 90% reduction in trace volume
- Lower backend storage
- Lower processing cost
- Same traffic visibility trends
- BUT you still see latency distributions and errors.

Solution 2.

- ☑ In the OTel Collector config use tail based sampling
- ☑ Keep:
  - ◆ Errors
  - ◆ High latency traces
  - ◆ Rare endpoints

```
processors:
  tail_sampling:
    policies:
      - name: errors
        type: status_code
        status_code:
          status_codes: [ERROR]
      - name: slow_requests
        type: latency
        latency:
          threshold_ms: 1000
```

🌱 Why this is green observability? 🌱

- You will store 100% of important traces
- ~5–20% of normal traffic

Solution 3.

- ☑ If traffic spikes:
  - ◆ Reduce sampling percentage automatically
  - ◆ Maintain constant resource envelope
- ☑ When system load increases → reduce telemetry load.
- ☑ Green observability that scales responsibly.

What if we built green-first observability?

- Built-in telemetry carbon dashboards
- Default short retention policies
- Cardinality warnings with “energy impact” hints
- Sampling presets based on energy budgets
- Storage footprint trend alerts
- Not “collect everything”, but “collect intentionally”

Kepler - Kubernetes-based Efficient Power Level Exporter

- A CNCF project that uses eBPF and system counters to estimate energy use of containers, pods, VMs, and processes in Kubernetes environments.
- Kepler is an observability metric exporter: it exposes energy consumption via Prometheus and can be integrated with observability stacks including OpenTelemetry.
- Kepler does not natively implement the Green Software Foundation's SCI specification, but its energy metrics can be used as the energy input in an SCI-based measurement pipeline so that teams can estimate carbon intensity per functional unit.

VictoriaMetrics Sustainable Mission

Green Coding Practices

Preventing Code Bloat

Careful Use of External Libraries

Observability and Profiling

Language Choice

VictoriaMetrics Sustainable Features

1.7x
less memory

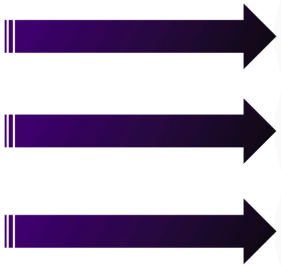
90%
energy cost reduction

2.5x
less disk space

10x
infrastructure cost Savings

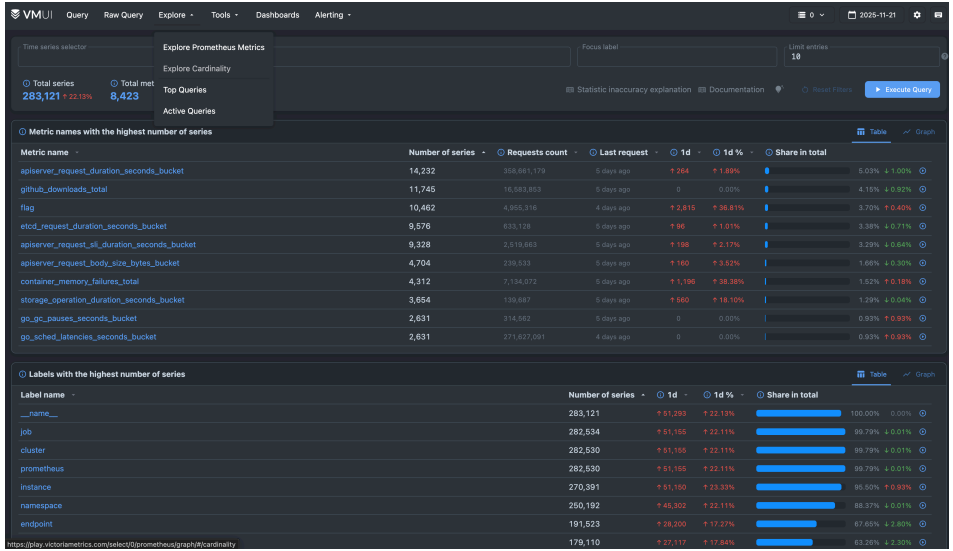
16x
faster query latency

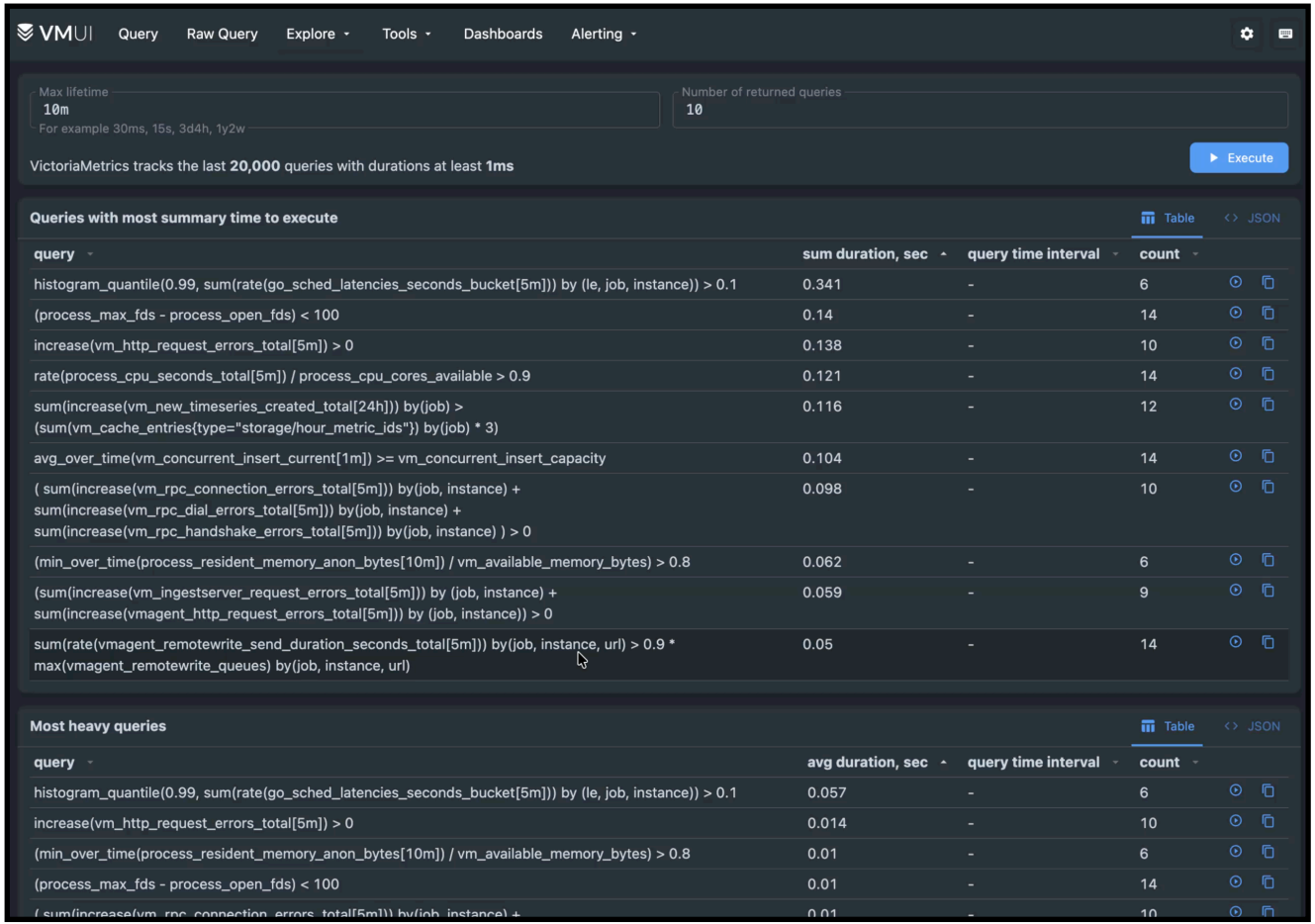
4x
network cost Savings



- Benchmarks
- Energy Savings
- Hardware reduction

Find your “most bloated” metrics in Cardinality Explorer



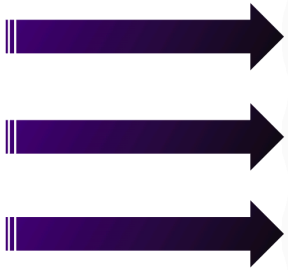

VMUI
Query
Raw Query
Explore
Tools
Dashboards
Alerting
Max lifetime
10m
For example 30ms, 15s, 3d4h, 1y2w
Number of returned queries
10
VictoriaMetrics tracks the last 20,000 queries with durations at least 1ms
Execute
Queries with most summary time to execute
Table
JSON
query
sum duration, sec
query time interval
count
histogram_quantile(0.99, sum(rate(go_sched_latencies_seconds_bucket[5m])) by (le, job, instance)) > 0.1
0.341
-
6
(process_max_fds - process_open_fds) < 100
0.14
-
14
increase(vm_http_request_errors_total[5m]) > 0
0.138
-
10
rate(process_cpu_seconds_total[5m]) / process_cpu_cores_available > 0.9
0.121
-
14
sum(increase(vm_new_timeseries_created_total[24h])) by(job) > (sum(vm_cache_entries{type="storage/hour_metric_ids"}) by(job) * 3)
0.116
-
12
avg_over_time(vm_concurrent_insert_current[1m]) >= vm_concurrent_insert_capacity
0.104
-
14
(sum(increase(vm_rpc_connection_errors_total[5m])) by(job, instance) + sum(increase(vm_rpc_dial_errors_total[5m])) by(job, instance) + sum(increase(vm_rpc_handshake_errors_total[5m])) by(job, instance)) > 0
0.098
-
10
(min_over_time(process_resident_memory_anon_bytes[10m]) / vm_available_memory_bytes) > 0.8
0.062
-
6
(sum(increase(vm_ingestserver_request_errors_total[5m])) by (job, instance) + sum(increase(vmagent_http_request_errors_total[5m])) by (job, instance)) > 0
0.059
-
9
sum(rate(vmagent_remotewrite_send_duration_seconds_total[5m])) by(job, instance, url) > 0.9 * max(vmagent_remotewrite_queues) by(job, instance, url)
0.05
-
14
Most heavy queries
Table
JSON
query
avg duration, sec
query time interval
count
histogram_quantile(0.99, sum(rate(go_sched_latencies_seconds_bucket[5m])) by (le, job, instance)) > 0.1
0.057
-
6
increase(vm_http_request_errors_total[5m]) > 0
0.014
-
10
(min_over_time(process_resident_memory_anon_bytes[10m]) / vm_available_memory_bytes) > 0.8
0.01
-
6
(process_max_fds - process_open_fds) < 100
0.01
-
14

VictoriaLogs Sustainable Features

1000x
boosts haystack search speed

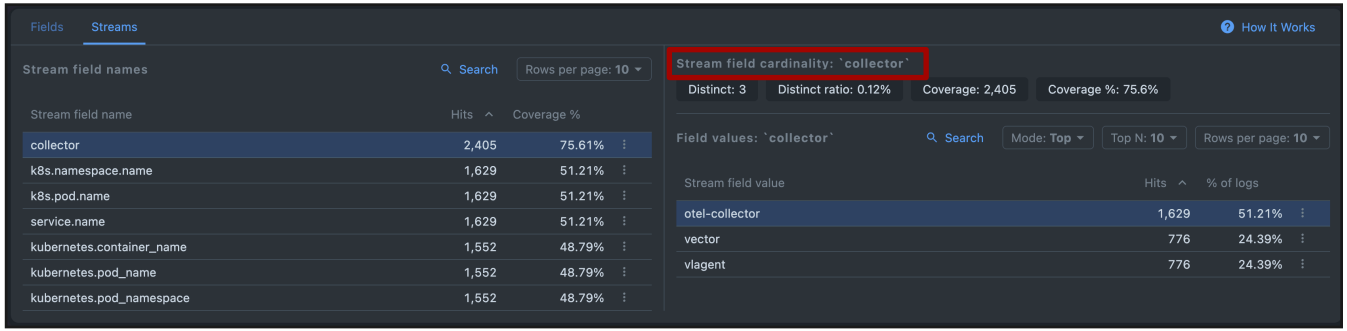
30x
less RAM

15x
less disk space



- Benchmarks
- Improved Query Performance
- Resource Efficiency

_msg	[2026-02-26T11:31:53.207Z] "GET /api/products/LS4PSXUNUM HTTP/1.1" 200 - via_upstream - "-" 0 535 5 5 "-" "python-requests/2.32.5" "dff63aa8-72f4-9f3b-b730-6d0dca765bd8" "frontend-proxy:8080" "10.71.129.233:8080" frontend 10.71.11.41:57750 10.71.11.41:8080 10.71.11.43:49258 - -
_stream	{collector="otel-collector",k8s.namespace.name="play-otel",k8s.pod.name="frontend-proxy-6f796d6c45-7wjhv",service.name="frontend-proxy"}
_stream_id	0000000000000000042968cdf4338a8977ef3126bacacfe13
_time	2026-02-26T11:31:53.207076Z
collector	otel-collector
destination.address	10.71.129.233
event.name	proxy.access
host.name	otel-collector-b9dd8f965-7lklb
k8s.deployment.name	frontend-proxy
k8s.namespace.name	play-otel
k8s.node.name	gke-sandbox-e2-standard-8-20250715071-5b0a2ce9-stoq
k8s.pod.ip	10.71.11.41
k8s.pod.name	frontend-proxy-6f796d6c45-7wjhv
k8s.pod.start_time	2026-02-21T12:47:55Z
k8s.pod.uid	f69bd6b9-27c0-4895-92bc-bd056b76ea53
log_name	otel_envoy_access_log
os.type	linux
scope.name	unknown
scope.version	unknown
server.address	10.71.11.41:8080
service.instance.id	f69bd6b9-27c0-4895-92bc-bd056b76ea53
service.name	frontend-proxy
severity	Unspecified
source.address	10.71.11.43
span_id	6ca94cdc5cdbc237
trace_id	411d12a6df0244d0c7af4477f8bb06bf
upstream.cluster	frontend

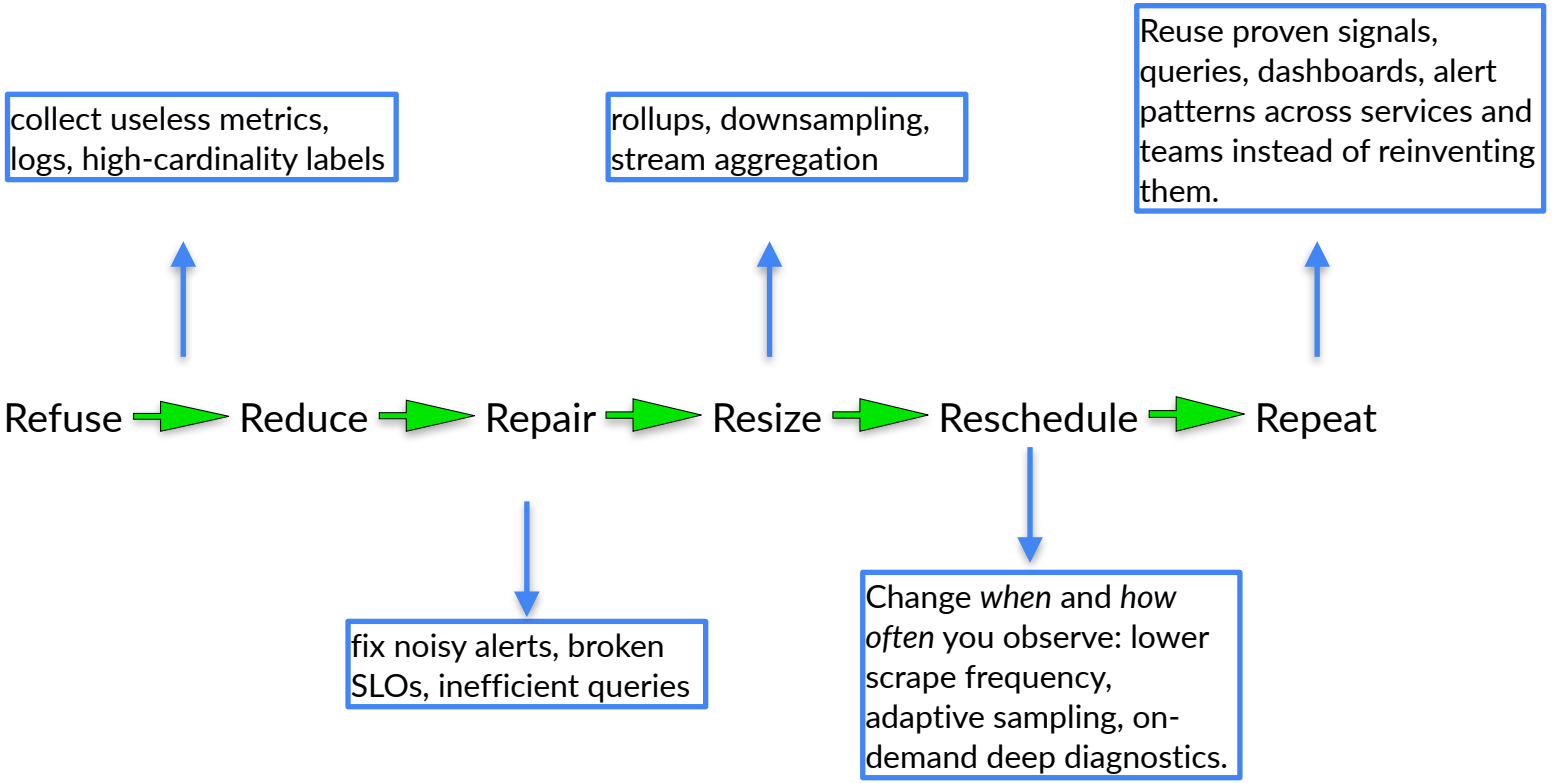


VictoriaTraces Sustainable Features

10x
compression

little CPU, RAM, and disk usage

Resource Efficiency


collect useless metrics, logs, high-cardinality labels
rollups, downsampling, stream aggregation
Reuse proven signals, queries, dashboards, alert patterns across services and teams instead of reinventing them.
Refuse
Reduce
Repair
Resize
Reschedule
Repeat
fix noisy alerts, broken SLOs, inefficient queries
Change *when* and *how often* you observe: lower scrape frequency, adaptive sampling, on-demand deep diagnostics.

Take-aways

- Observability companies should start applying Green Software Foundation's actions since day 1
- Green observability depends on intentional design choices, not specific tools.
- Ultimately, we are all responsible to maintain it.

MERGE FORWARD

Learn more

community.cncf.io/merge-forward

Creating diverse, supportive communities and ally networks for shared learning, mentorship, friendship, and collaborative idea exchange.

#merge-forward on the CNCF Slack!

CLOUD NATIVE COMPUTING FOUNDATION

Resources

https://greensoftware.foundation/articles/what-is-green-software
https://github.com/cncf/tag-env-sustainability
https://www.thegreenwebfoundation.org/news/creating-a-standard-for-measuring-software-carbon-intensity-for-the-web/
CNCF Slack #tag-operational-resilience

https://play.victoriametrics.com/
https://docs.victoriametrics.com/
Sustainability at VictoriaMetrics

Thank you!

Bsky: @didiviking.bsky.social
X: @dianavtodea
Github: @didiViking/Conferences_Talks
LinkedIn: @diana-todea-b2a79968

VictoriaMetrics Community