

Prometheus 3.0 and the new Governance: Setting up the project for the next decade



Josh Abreu

Principal Software Engineer, Grafana Labs

Josh “gotjosh” Abreu

Principal Software Engineer

Prometheus & Alertmanager maintainer

The “Alerting” guy but works on next generation of our metrics product



gotjosh

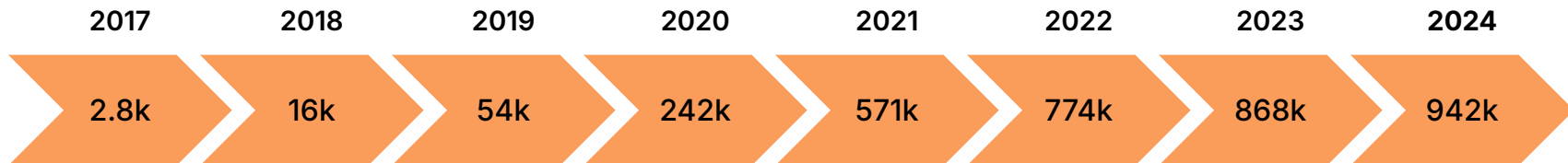


Show of hands



History

- Started at SoundCloud in 2012
- Joined the CNCF in 2016; 2nd ever
- Released 2.0 in 2017
- Graduated as a CNCF project in 2018; 2nd ever



What is Prometheus?

Metrics-based monitoring & alerting stack.

- Instrumentation ecosystem for application and systems
- Metrics collection and storage
- Querying, Alerting & Dashboarding
- For all levels of the stack!

Design from the ground up for dynamic cloud environments.

Architecture

Targets

Web Application
via Client Lib

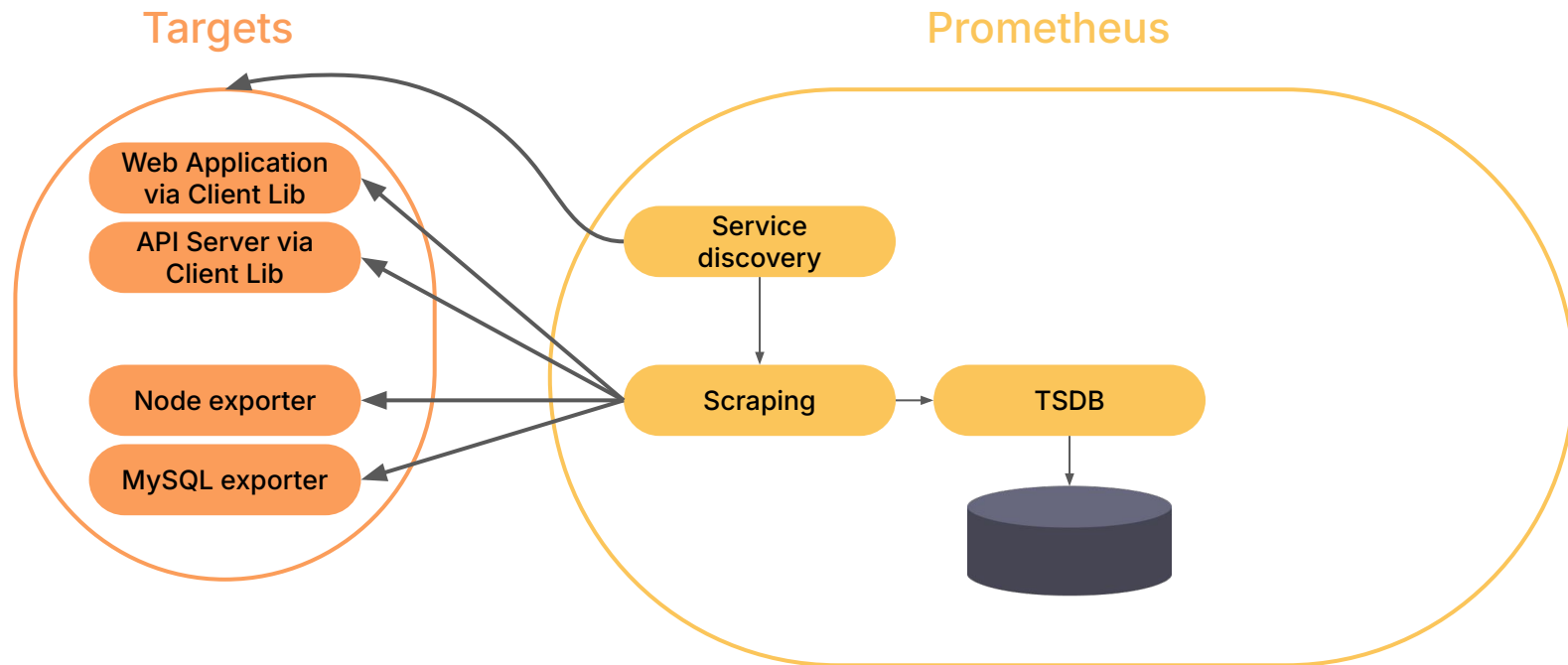
API Server via
Client Lib

Node exporter

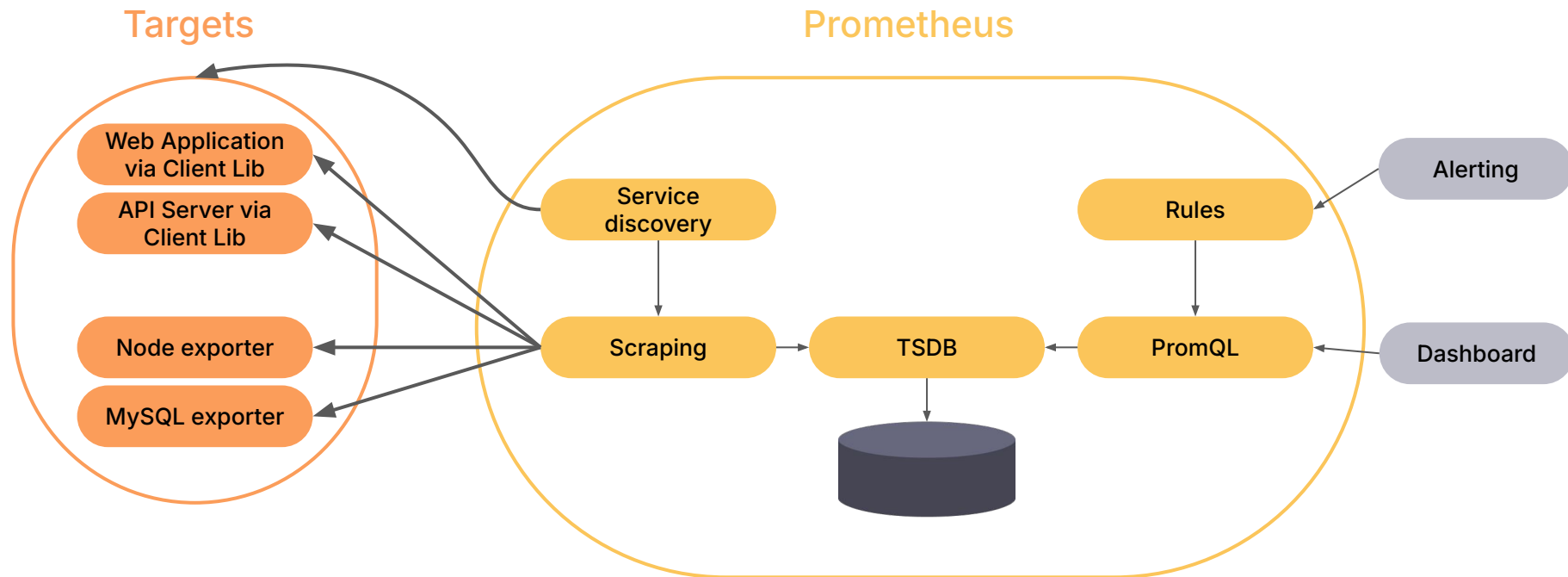
MySQL exporter



Architecture



Architecture



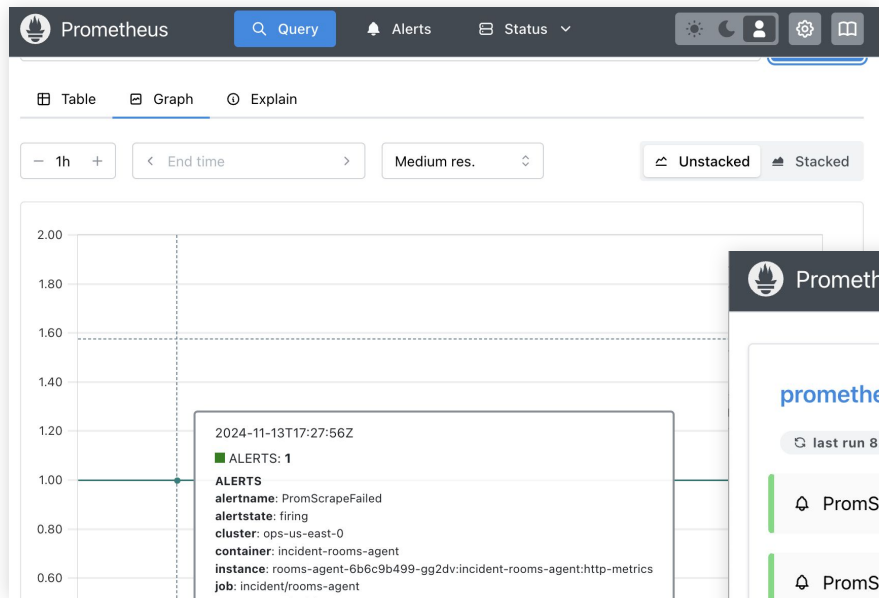


What's new?

Releases

- prometheus/prometheus
 - 4 main releases – 3.0 to 3.4
 - 1 LTS release – 2.45.6 to 2.55.1
- prometheus/alertmanager 0.28.1
- Too many exporter updates to list

New UI and UX



The screenshot shows the Prometheus UI with the 'Status > Rule health' tab selected. The page displays the rule health for the rule 'prometheus-extra' located at '/etc/prometheus/mixins/base/alerts.rules'. The rule is shown as 'last run 8.404s ago', 'took 2ms', and 'every 1m'. Below the rule information, a table lists the alerts:

Alert Name	Last Run	Duration	Status	Actions
PromScrapeFailed	8.405s ago	1ms	OK	▼
PromScrapeFlapping	8.404s ago	1ms	OK	▼
PromScrapeTooLong	8.403s ago	1ms	OK	▼

New UI and UX - Query explainer

The screenshot shows the Prometheus Query Editor interface. At the top, there's a navigation bar with 'Query', 'Alerts', and 'Status' tabs. The main area displays a query: `>_ sum by (version) (rate(prometheus_http_requests_total[5m]) * on (job) group_left(version) prometheus_build_info)`. Below the query, an execution plan is shown, breaking down the query into steps: `sum by(version)` (0 results - 83ms), `rate` (0 results - 76ms), `prometheus_http_requests_total[5m]` (0 results - 82ms), `* on(job) group_left(version)` (0 results - 76ms), and `prometheus_build_info` (0 results - 90ms). At the bottom, there's a 'Range vector selector' section with a 'Metric help' for 'Counter of HTTP requests' and a 'Metric type' of 'counter'. It explains that the query selects 5m of data going backward from the evaluation timestamp for any series that match all of the following criteria:

- The metric name is `prometheus_http_requests_total`.

 It also notes that if a series has no values in the last 5m, the series will not be returned.

The screenshot shows the 'Explain' tab of the Prometheus Query Explainer. It displays a 'Vector-to-vector binary operation' which calculates the result of applying the 'and' operator between the sample values of matching series from two sets of time series. The operation is defined as: `group_left()`: many-to-many match. Each series from the right-hand side is allowed to match with multiple series from the left-hand side.

The operation is visualized as follows:

Series 1	Series 2	Result
<code>{cluster="dev-us-central-0", component="discovery", grafana_slo_uid="GrafanaRulerGetByRuleGroup", namespace="grafana-ruler"}</code>	<code>{cluster="dev-us-central-0", component="discovery", grafana_slo_uid="GrafanaRulerGetByRuleGroup", namespace="grafana-ruler"}</code>	<code>{}</code> 0.3076923076923077
<code>{cluster="dev-us-central-0", component="discovery", grafana_slo_uid="GrafanaRulerGetByRuleGroup", namespace="grafana-ruler"}</code>	<code>{cluster="dev-us-central-0", component="discovery", grafana_slo_uid="GrafanaRulerGetByRuleGroup", namespace="grafana-ruler-dedicated-0"}</code>	<code>{}</code> 505.2631578947368
no matching series	<code>{cluster="dev-us-central-0", component="discovery", grafana_slo_uid="GrafanaRulerGetByRuleGroup", namespace="grafana-ruler-dedicated-0"}</code>	<code>{}</code> 1137.8932225656465

New UI and UX - Query explainer

The screenshot displays the Prometheus Query Explainer interface. At the top, a query `* on(job) group_left(version)` is shown with an error message: "Error executing query: found duplicate series for the match group (job='prometheus-k8s') on the right hand-side of the operation: [{__name__='pr...". Below the query, the query type is `prometheus_build_info` and the execution time is 2 results - 11ms. The labels are listed as `instance: 2 pod: 2 branch: 1 container: 1 endpoint: 1 goarch: 1 goos: 1 goversion: 1 job: 1 namespace: 1 ...4 more...`.

The interface has three tabs: Table, Graph, and Explain. The Explain tab is selected, showing the "Vector-to-vector binary operation" section. This section explains that the node calculates the result of applying the "*" operator between the sample values of matching series from two sets of time series.

- `on(job)`: series on both sides are matched on the labels `job`
- `group_left(version)`: many-to-one match. Each series from the right-hand side is allowed to match with multiple series from the left-hand side. Any `version` labels found on the right-hand side are propagated into the result, in addition to the match group's labels.

A toggle switch labeled "Show sample values" is located to the right of the explanation.

A yellow warning box states: "Found matching issues in 1 match group. See below for per-group error details."

A red error box titled "Error in match group below" provides further details: "You requested **many-to-one matching** via `group_left()`, but we also found **multiple series on the right side** of the match group. Make sure that the right side only contains a single series." It lists two possible fixes:

- Update your matching parameters:** Consider including more differentiating labels in your matching modifiers (via `on()` / `ignoring()`) to split multiple series into distinct match groups.
- Aggregate the input:** Consider aggregating away the extra labels that create multiple series per group before applying the binary operation.

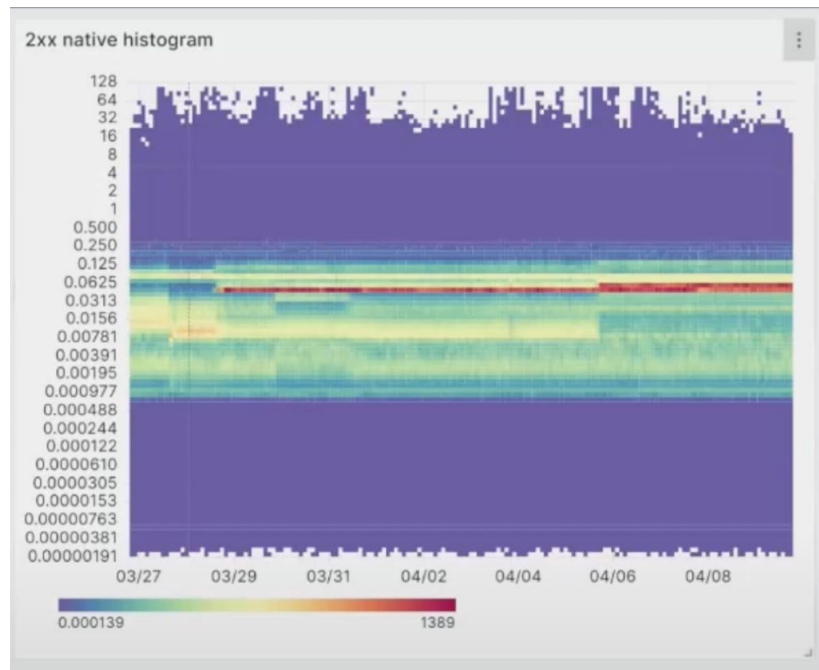
Native Histograms

```
// newInstrumentMiddlewareMetrics makes a new instrumentMiddlewareMetrics.  
func newInstrumentMiddlewareMetrics(registerer prometheus.Registerer) *instrumentMiddl  
    return &instrumentMiddlewareMetrics{  
        duration: promauto.With(registerer).NewHistogramVec(prometheus.Histogr  
            Namespace: "cortex",  
            Name:      "frontend_query_range_duration_seconds",  
            Help:      "Total time spent in seconds doing query range requ  
            Buckets:   prometheus.DefBuckets,  
        }, []string{"method", "status_code"}),  
    }  
}
```

Native Histograms

```
uncompressedBodySize: promauto.With(reg).NewHistogramVec(prometheus  
    Name: "cortex_distributor_uncom  
|    Help: "Size of uncompressed rec  
    NativeHistogramBucketFactor: 1.1,  
    NativeHistogramMinResetDuration: 1 * time.Hour,  
    NativeHistogramMaxBucketNumber: 100,  
, []string{"user"}),
```

Native Histograms



Remote Write 2.0

The screenshot shows the Prometheus website's navigation bar with links for DOCS, DOWNLOAD, COMMUNITY, SUPPORT & TRAINING, and BLOG. A search bar is also present. The left sidebar contains a list of navigation items: INTRODUCTION, CONCEPTS, PROMETHEUS SERVER, VISUALIZATION, INSTRUMENTING, OPERATING, ALERTMANAGER, BEST PRACTICES, GUIDES, TUTORIALS, and SPECIFICATIONS. The main content area is titled 'PROMETHEUS REMOTE-WRITE SPECIFICATION'. It includes a metadata section with 'Version: 2.0-rc.3', 'Status: Experimental', and 'Date: May 2024'. The text explains that the Remote-Write specification documents the standard for how Prometheus and Prometheus Remote-Write compatible senders send data to Prometheus or Prometheus Remote-Write compatible receivers. It also states that this document defines a second version of the Prometheus Remote-Write API with minor changes to protocol and semantics, adding a new Protobuf Message with new features for more use cases and wider adoption. The document deprecates the previous Protobuf Message from a 1.0 Remote-Write specification and adds mandatory X-Prometheus-Remote-Write-* HTTP response headers for reliability. Finally, it outlines how to implement backwards-compatible senders and receivers (even under a single endpoint) using existing basic content negotiation request headers. More advanced, automatic content negotiation mechanisms might come in a future minor version if needed. For the rationales behind the 2.0 specification, see the formal proposal. A key words section at the bottom states that 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' are to be interpreted as described in RFC 2119. A right sidebar contains a table of contents with links to Introduction, Background, Glossary, Definitions, Protocol, Response, Retries & Backoff, Backward and Forward Compatibility, Protobuf Message, io.prometheus.write.v2.Request, Out of Scope, Future Plans, Related, and FAQ.

Prometheus

DOCS DOWNLOAD COMMUNITY SUPPORT & TRAINING BLOG Search

INTRODUCTION

CONCEPTS

PROMETHEUS SERVER

VISUALIZATION

INSTRUMENTING

OPERATING

ALERTMANAGER

BEST PRACTICES

GUIDES

TUTORIALS

SPECIFICATIONS

PROMETHEUS REMOTE-WRITE SPECIFICATION

- Version: 2.0-rc.3
- Status: **Experimental**
- Date: May 2024

The Remote-Write specification, in general, is intended to document the standard for how Prometheus and Prometheus Remote-Write compatible senders send data to Prometheus or Prometheus Remote-Write compatible receivers.

This document is intended to define a second version of the [Prometheus Remote-Write](#) API with minor changes to protocol and semantics. This second version adds a new Protobuf Message with new features enabling more use cases and wider adoption on top of performance and cost savings. The second version also deprecates the previous Protobuf Message from a [1.0 Remote-Write specification](#) and adds mandatory `X-Prometheus-Remote-Write-*` [HTTP response headers](#) for reliability purposes. Finally, this spec outlines how to implement backwards-compatible senders and receivers (even under a single endpoint) using existing basic content negotiation request headers. More advanced, automatic content negotiation mechanisms might come in a future minor version if needed. For the rationales behind the 2.0 specification, see [the formal proposal](#).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#).

- Introduction
 - Background
 - Glossary
- Definitions
 - Protocol
 - Response
 - Retries & Backoff
 - Backward and Forward Compatibility
- Protobuf Message
 - `io.prometheus.write.v2.Request`
- Out of Scope
- Future Plans
- Related
 - FAQ

Prometheus Remote-Write 2.0 EXPERIMENTAL

Remote Write 2.0 - How?

http_requests_total{code="200"} 2
http_requests_total{code="404"} 1

```
{
  "labels": [
    { "name": "__name__", "value": "http_requests_total" },
    { "name": "code", "value": "200" }
  ],
  "samples": [
    {
      "value": 2,
      "timestamp": 1825839765
    }
  ]
},
{
  "labels": [
    { "name": "__name__", "value": "http_requests_total" },
    { "name": "code", "value": "404" }
  ],
  "samples": [
    {
      "value": 1,
      "timestamp": 1825839765
    }
  ]
}
```

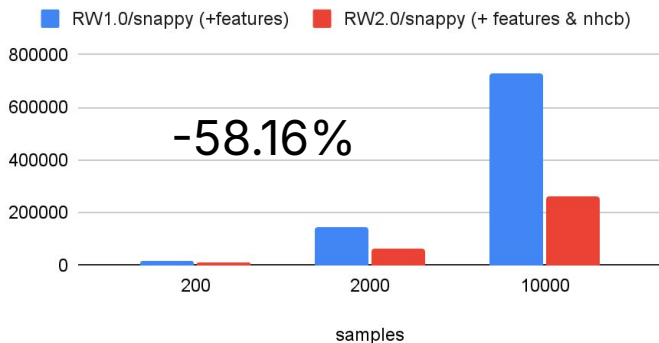
Remote Write 2.0 - How?

http_requests_total{code="200"} 2
http_requests_total{code="404"} 1

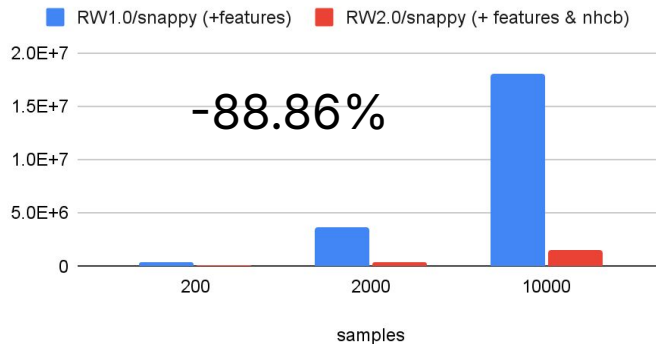
```
[__name__, http_requests_total, code, 200, 404]
timeseries: [
  {
    labels_ref: [0, 1, 2, 3],
    samples: [{
      "value": 2,
      "timestamp": 1825839765
    }]
  },
  {
    labels_ref: [0, 1, 2, 4],
    samples: [{
      "value": 1,
      "timestamp": 1825839765
    }]
  }
]
```

Remote Write 2.0

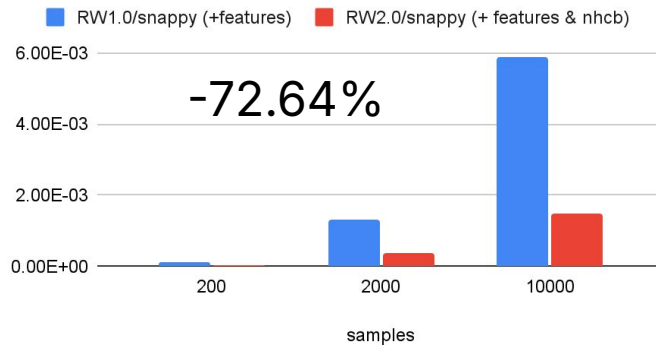
Message bytes on the wire (lower is better)



Serialization bytes allocated (lower is better)



Serialization CPU nanoseconds (lower is better)



OpenTelemetry & Prometheus

PUSH

The screenshot shows the Prometheus website with a dark header containing navigation links: DOCS, DOWNLOAD, COMMUNITY, SUPPORT & TRAINING, and BLOG. A search bar is on the right. The left sidebar lists various guides: INTRODUCTION, CONCEPTS, PROMETHEUS SERVER, VISUALIZATION, INSTRUMENTING, OPERATING, ALERTMANAGER, BEST PRACTICES, and GUIDES. The main content area is titled 'USING PROMETHEUS AS YOUR OPENTELEMETRY BACKEND'. It explains that Prometheus supports OTLP (OpenTelemetry Protocol) ingestion through HTTP. A section 'Enable the OTLP receiver' states that by default, the OTLP receiver is disabled and provides instructions to enable it using the CLI flag `--web.enable-otlp-receiver`. A code block shows the command: `$ prometheus --web.enable-otlp-receiver`. A list of configuration steps is provided: Enable the OTLP receiver, Send OpenTelemetry Metrics to the Prometheus Server, Configuring Prometheus (with sub-points for out-of-order ingestion and resource attributes), Including resource attributes at query time, UTF-8, and Delta Temporality. The next section, 'Send OpenTelemetry Metrics to the Prometheus Server', explains that the source of the OTLP metrics traffic must be told about the Prometheus endpoint and that HTTP mode should be used (gRPC is the default). It concludes by stating that OpenTelemetry SDKs can be configured via standard environment variables.

Prometheus

DOCS DOWNLOAD COMMUNITY SUPPORT & TRAINING BLOG Search

INTRODUCTION

CONCEPTS

PROMETHEUS SERVER

VISUALIZATION

INSTRUMENTING

OPERATING

ALERTMANAGER

BEST PRACTICES

GUIDES

Basic auth

Monitoring Docker container metrics using cAdvisor

USING PROMETHEUS AS YOUR OPENTELEMETRY BACKEND

Prometheus supports [OTLP](#) (aka "OpenTelemetry Protocol") ingestion through [HTTP](#).

Enable the OTLP receiver

By default, the OTLP receiver is disabled, similarly to the Remote Write receiver. This is because Prometheus can work without any authentication, so it would not be safe to accept incoming traffic unless explicitly configured.

To enable the receiver you need to toggle the CLI flag `--web.enable-otlp-receiver`. This will cause Prometheus to serve OTLP metrics receiving on HTTP `/api/v1/otlp/v1/metrics` path.

```
$ prometheus --web.enable-otlp-receiver
```

- [Enable the OTLP receiver](#)
- [Send OpenTelemetry Metrics to the Prometheus Server](#)
- [Configuring Prometheus](#)
 - [Enable out-of-order ingestion](#)
 - [Promoting resource attributes](#)
- [Including resource attributes at query time](#)
- [UTF-8](#)
- [Delta Temporality](#)

Send OpenTelemetry Metrics to the Prometheus Server

Generally you need to tell the source of the OTLP metrics traffic about Prometheus endpoint and the fact that the [HTTP](#) mode of OTLP should be used (gRPC is usually a default).

OpenTelemetry SDKs and instrumentation libraries can be usually configured via [standard environment variables](#). The following are the OpenTelemetry variables needed to send OpenTelemetry metrics to a Prometheus server.

OpenTelemetry & Prometheus

```
otlp:
# Recommended attributes to be promoted
promote_resource_attributes:
- service.instance.id
- service.name
- service.namespace
- cloud.availability_zone
- cloud.region
- container.name
- deployment.environment.name
- k8s.cluster.name
- k8s.container.name
- k8s.cronjob.name
- k8s.daemonset.name
- k8s.deployment.name
- k8s.job.name
- k8s.namespace.name
- k8s.pod.name
- k8s.replicaset.name
- k8s.statefulset.name
# Ingest OTLP data keeping UTF-8
translation_strategy: NoUTF8Escaping

storage:
# OTLP is a push-based protocol,
tsdb:
  out_of_order_time_window: 30m
```

Metric: `http.client.active_requests` #

Status: [Experimental](#)

This metric is optional.

Name	Instrument Type	Unit (UCUM)	Description	Stability
http.client.active_requests	UpDownCounter	{request}	Number of active HTTP requests.	experimental

Attribute	Type	Description	Examples	Requirement Level	Stability
server.address	string	Server domain name if available without reverse DNS lookup; otherwise, IP address or Unix domain socket name. [1]	example.com ; 10.1.2.80 ; /tmp/my.sock	Required	stable

 Prometheus

Query

Alerts

Status

>_ {"http.client.active_requests", "server.⚠️"}

Execute

Table

Graph

Explain

< Evaluation time >

Load time: 17ms Result series: 1

{"http.client.active_requests", instance="my-laptop", "server.address"="top-secret🔒", "server.⚠️"="🔥", "service.instance.id"="my-laptop"}

75

OpenTelemetry & Prometheus

Quoting in exposition format



```
# HELP "http.requests.total" The total number of HTTP requests.  
# TYPE "http.requests.total" counter  
{"http.requests.total", code="200"} 1021 1565133713.989  
{"http.requests.total", code="400"} 1 1565133713.990
```

Quoting in PromQL



```
{"http.requests.total"}  
  
sum by ("service.name") ({"http.requests.total", "service.name"="my service"})
```

Why Not Just Dots?

UTF-8 support in Prometheus 3.0



Owen Williams
Senior Software Engineer

Sep 2024



OpenTelemetry & Prometheus

SOON

```

75
76 // metadata represents the metadata associated with the given series' samples
77 Metadata metadata = 5 [(gogoproto.nullable) = false];
78
79 // created_timestamp represents an optional created timestamp associated with
80 // this series' samples in ms format, typically for counter or histogram type
81 // metrics. Created timestamp represents the time when the counter started
82 // counting (sometimes referred to as start timestamp), which can increase
83 // the accuracy of query results.
84 //
85 // Note that some receivers might require this and in return
86 // ingest such samples within the Request.
87 //
88 // For Go, see github.com/prometheus/prometheus/model/timestamp
89 // for conversion from/to time.Time to Prometheus timestamp.
90 //
91 // Note that the "optional" keyword is omitted due to
92 // https://cloud.google.com/apis/design/design_patterns.md#optional
93 // Zero value means value not set. If you need to use exactly
94 // the timestamp, use 1 millisecond before or after.
95 int64 created_timestamp = 6;
96 }
97

```

Open

2 of 13 tasks

[Meta Issue] Created Timestamp Plan #14217
 bwplotka opened this issue on Jun 5 · 0 comments

- more accurate rates
- Support [OTLP protocol](#) one day.
- Support backends which require CT for accuracy e.g. Google Cloud Monitoring.

Issues

Following work is planned. A bit of rationales are provided for prioritization purposes:

- ☒ PRW spec change: [remote write 2.0: Add support for created timestamp propagation \[spec\]](#) #13368
- ☐ Fix final known limitations of OM text CT parsing (`TestCTParseFailures`)
- ☐ Optimize OM text CT parsing [Created Timestamp: Optimize memory use for OM Text CT parsing \(with created-timestamp-zero-ingestion feature flag\)](#) #14823
- ☐ Scrape loop test and skipping CT metrics when CT injection is set on <https://github.com/prometheus/prometheus/pull/13506/files#diff-94ac40a66b06f7a86e45c1415b90cece99f310db590a8dd2b1310a5a81c21996R719>
 - Solves [Handle OpenMetrics ..._created_lines](#) #6541
- ☒ Parse CT from OM Text: [Parse Created timestamps from OpenMetrics text format](#) #12980
 - This also solves: [Store metadata for OpenMetrics _created metrics](#) #10164
- ☐ CT zero ingestion for native histograms: [tsdb: need histogram support for created-timestamp handling](#) #13384
- ☐ Propose alternative CT attachment to OM Text: [Propose alternative Created Timestamp handling for OpenMetrics](#)

<https://github.com/prometheus/prometheus/issues/14217>

OpenTelemetry & Prometheus

Support OTel delta temporality #12763

Open

gouthamve opened this issue on Aug 29, 2023 · 3 comments



gouthamve commented on Aug 29, 2023

Member

Proposal

I've outlined my thoughts here:
<https://docs.google.com/document/d/1vMtFKEkxRiwkr0JvVOrUrNTogVvHlcEWaWgZlqsY7Q/edit>

But I think supporting ingesting deltas instead of cumulatives will open us up to new use-cases.

4

5

gouthamve added component/remote storage component/promql not-as-easy-as-it-looks labels on Aug 29, 2023

gouthamve added this to the OTEL Support milestone on Aug 29, 2023

aknuds1 mentioned this issue on Nov 22, 2023

OTLP/HTTP Endpoint Invalid Temporality When Using AggregationTemporality.DELTA grafana/mimir#6696

carriedwards added this to Prometheus 3.0 on Nov 2, 2023

carriedwards moved this to Todo in Prometheus 3.0 on Jan 9

Assignees

No one—[assign yourself](#)

Labels

component/promql

component/remote storage

help wanted

not-as-easy-as-it-looks

Projects

Prometheus 3.0

Status: Todo

+2 more

Milestone

OTEL Support

Development

Create a branch for this issue or link a pull request.

Notifications

Subscribe

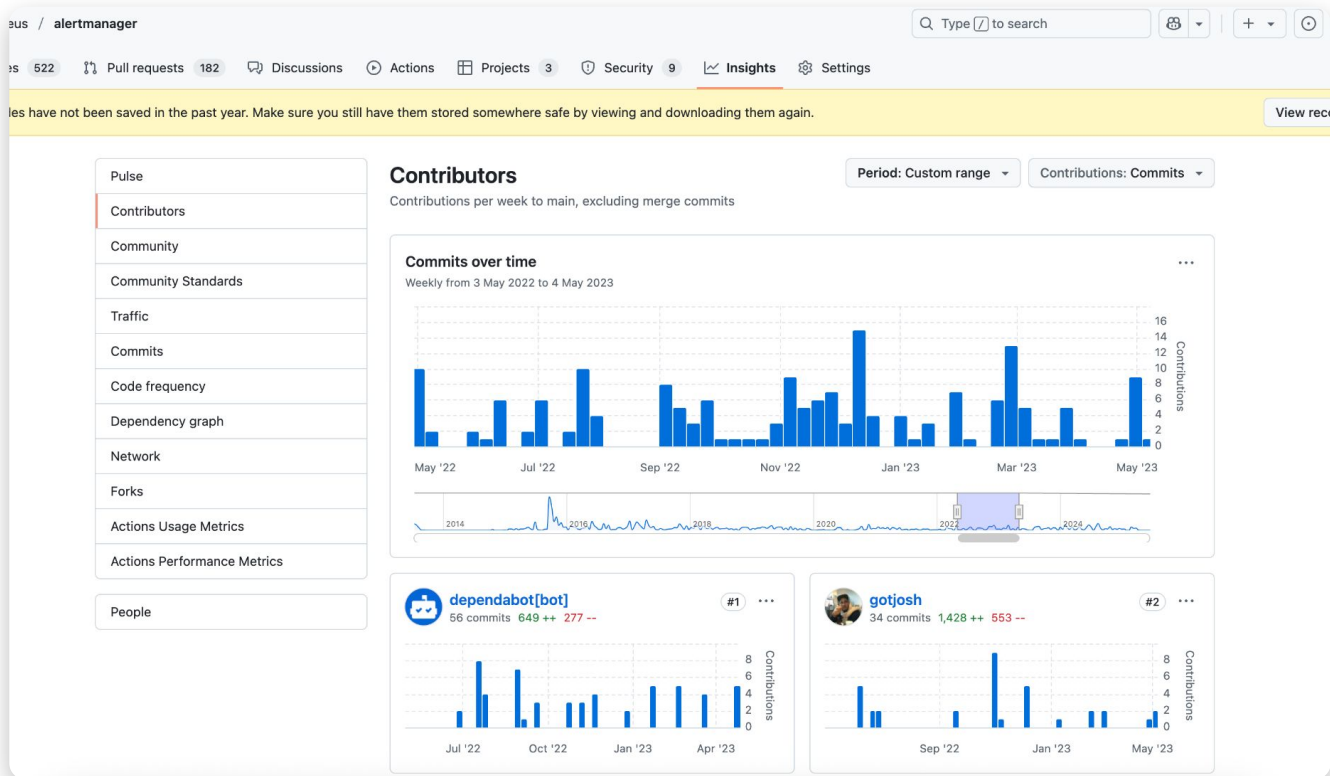
You're not receiving notifications from this thread.

<https://github.com/prometheus/prometheus/issues/12763>

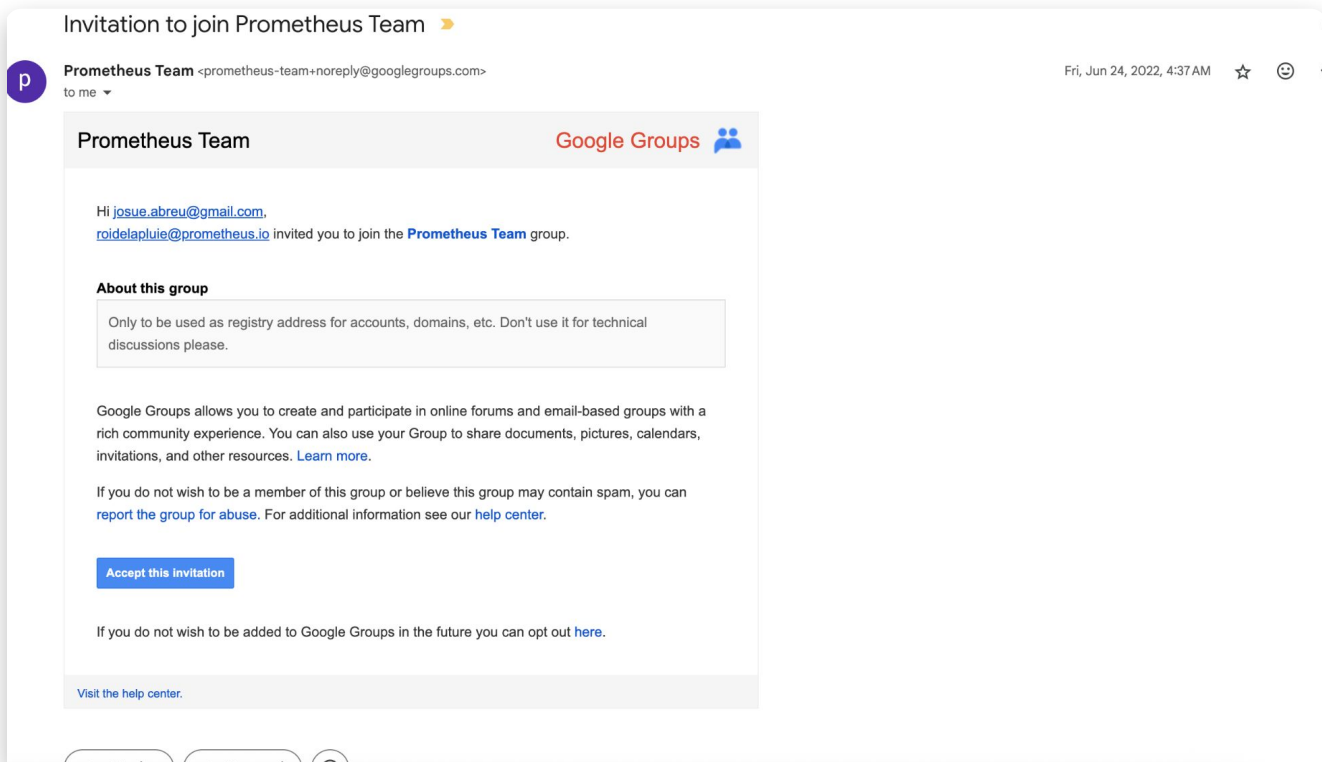


Governance

How I became a "Team" Member



How I became a "Team" Member



See a problem?



Governance 1.0

The screenshot shows a GitHub pull request interface for the repository 'prometheus / docs'. The pull request is titled 'Add governance #950' and is in a 'Merged' state. It was merged by 'juliusv' on January 21, 2018, merging 8 commits from the 'add-governance' branch into the 'master' branch. The interface includes a navigation bar with links to Issues (144), Pull requests (144), Discussions, Actions, Projects, Security, Insights, and Settings. A yellow warning banner at the top states: 'Every code has not been saved in the past year. Make sure you still have them stored somewhere safe by viewing and downloading them again.' Below the title, statistics show 62 conversations, 8 commits, 0 checks, and 4 files changed. A comment from 'matthiasr' (Member) dated January 9, 2018, explains that this PR replaces 'prometheus/prometheus#2930' and adds separate commits for any changes since then. The comment also notes that this brings the governance document to prometheus.io, allowing the team list to be placed directly on the page instead of linking back to the community page. At the bottom, a commit summary indicates that 'Matthias Rampke' added 5 commits 8 years ago.

prometheus / docs

Issues 144 Pull requests 144 Discussions Actions Projects Security Insights Settings

Every code has not been saved in the past year. Make sure you still have them stored somewhere safe by viewing and downloading them again.

Add governance #950

Merged juliusv merged 8 commits into `master` from `add-governance` on Jan 21, 2018

Conversation 62 Commits 8 Checks 0 Files changed 4

matthiasr commented on Jan 9, 2018 • edited

This PR replaces [prometheus/prometheus#2930](#). I added separate commits for any changes since then.

Since this brings the governance document to prometheus.io, the team list can be right there instead of linking back to the community page. The community page links to the governance. I removed the version from the title, since changes are tracked in git. Finally, I had to fiddle with the layouts a bit since this is the first non-docs, non-blog page in Markdown.

Matthias Rampke added 5 commits 8 years ago

Governance 1.0

- *Lazy consensus* is the default mechanism
- No consensus = Vote by majority (1/2)
- Members. Voted in / out by super-majority (2/3rd)
- Team members = FULL ADMIN ON EVERYTHING

Governance 1.0

Team member status may be given to those who have made ongoing contributions to the Prometheus project for at least 3 months.

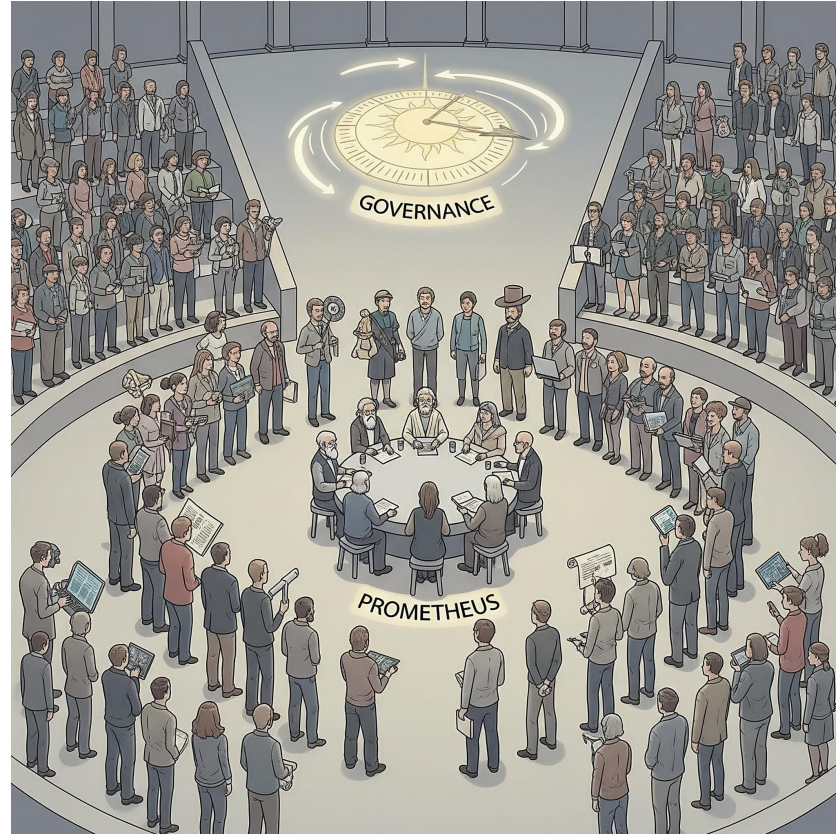
This is usually in the form of code improvements and/or notable work on documentation, but organizing events or user support could also be taken into account.

Top Problems

- No clear path to membership
- Hesitation to add new members
- No roles / organizing
- Grafana Labs over-representation
- Others you can read here:



Goal: Go from 30 to 300 diverse individuals



Proposal: New Governance

Roles

- Contributor
- Member
- Maintainer

Steering Committee

- 7 individuals
 - At most 2 from a company
- Voted for by members and maintainers

Proposal: New Governance

Roles

- Contributor
- Member
- Maintainer

Contributor is everyone who has ever contributed to Prometheus

Proposal: New Governance

Roles

- Contributor
- Member
- Maintainer

20 contributions in 12 months

Comments count, Meetups count,
Blog posts count

Proposal: New Governance

Roles

- Contributor
- Member
- Maintainer

Current maintainer nominates

No existing maintainer objects

Proposal: New Governance

How is the steering committee
defined?

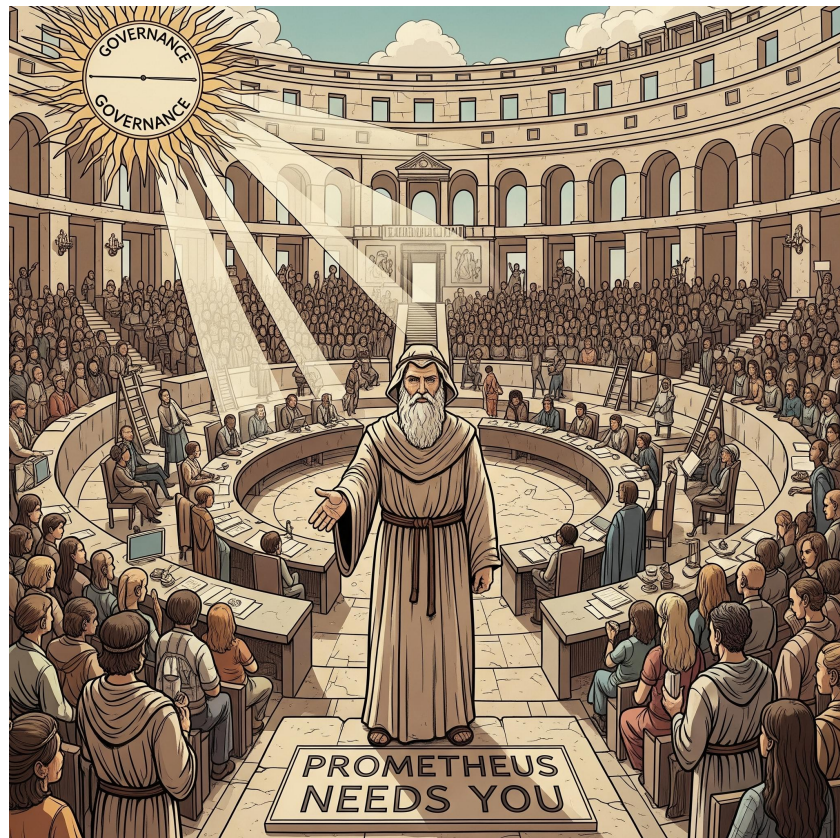
- Any *member* is eligible
- *Members* vote
- 2 year term

Goal: Go from 30 to 300 diverse individuals

Exporters

Docs

Issue triaging



SDKs

Alertmanager

Prometheus Server

Thank you 🙏

Questions?

