

ContribFest: Prometheus New Contributor Intro



Prometheus

ContribFest: Prometheus New Contributor Intro

Goal: Enable you to help us improve Prometheus



Prometheus

ContribFest: Prometheus New Contributor Intro

Goal: Enable you to help us improve Prometheus

Theme: Code Review



Prometheus



Who are we



Bryan Boreham
Grafana Labs

[@bboreham.bsky.social](https://bsky.social/@bboreham)



Ben Kochie
Reddit



Saswata Mukherjee
Red Hat



Arianna Vespri
OllyGarden



Bartłomiej Płotka
Google

Show of hands!



Q: Who's used (e.g. queried, deployed, configured) Prometheus?

Show of hands!



Q: Who contributed to Prometheus already (docs/issue/code)?

Show of hands!



Q: Who knows Go programming language?

Show of hands!



Q: Who use GenAI code tools in OSS (e.g. Claude Code)?

Show of hands!



Agenda

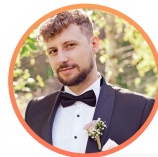
Prometheus Overview

Contribution Process

Why & How to Review

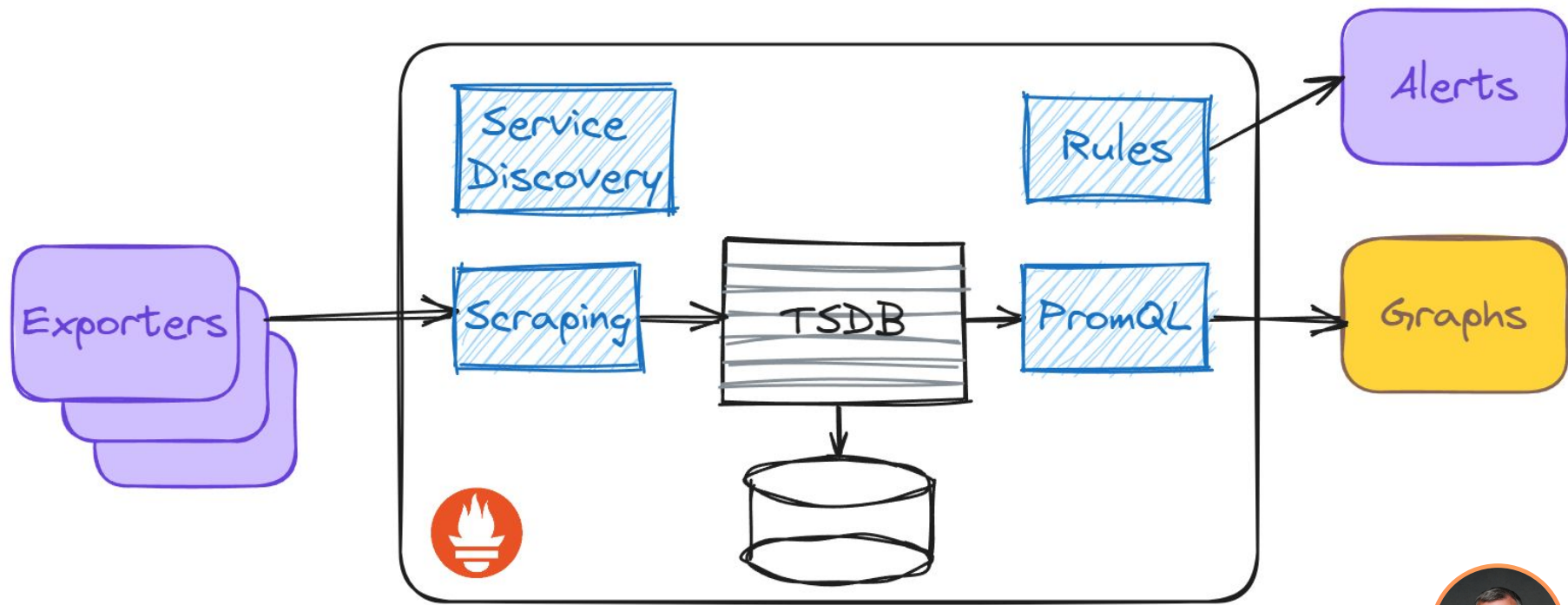
Review Workshop (start warming your laptops!)

Reflections & Q&A





Prometheus Overview



Different parts of Prometheus

Under <https://github.com/prometheus>:

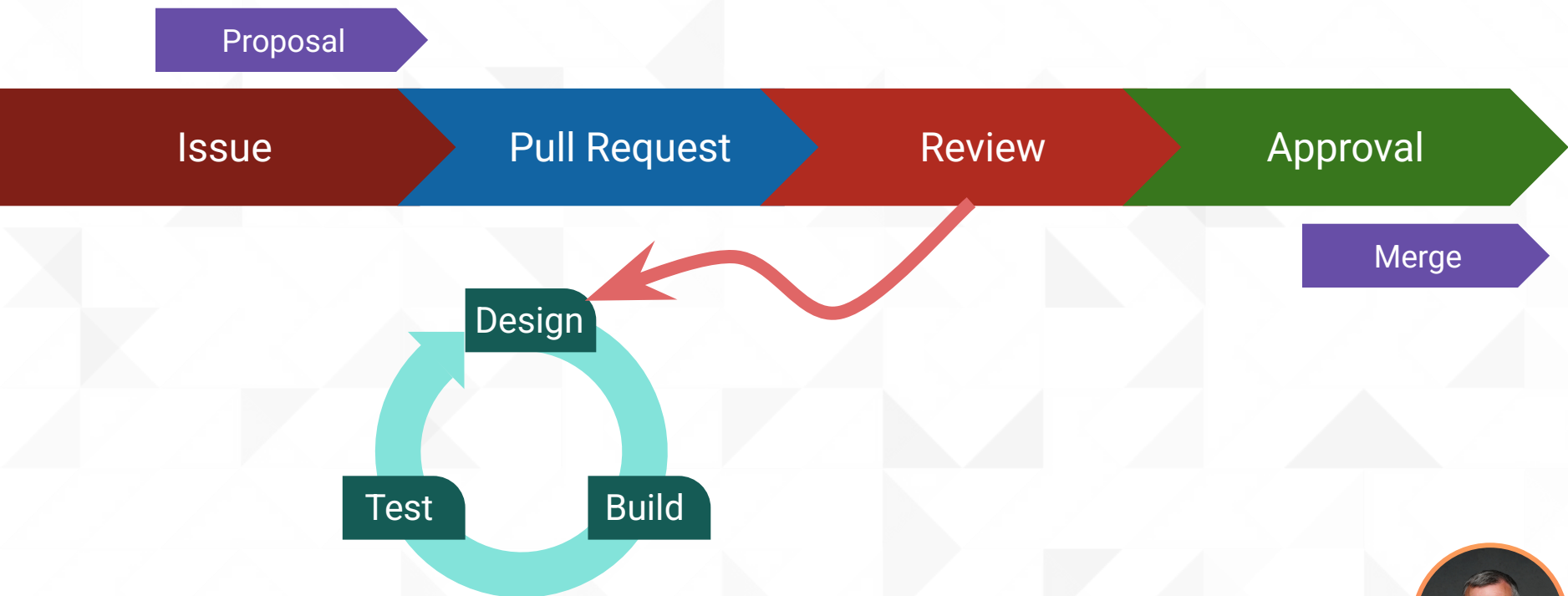
- prometheus
- alertmanager
- common
- client_golang, client_java, etc.
- exporters - node_exporter, blackbox_exporter, pushgateway, ...
- docs
- build infra - golang-builder, promu, test-infra, ...

Also <https://github.com/prometheus-community>





Contribution Lifecycle





Contributor Ladder

<https://github.com/prometheus/governance>

Steering Committee Member

Maintainer

Member

Contributor! <- **You are here!**

(Code author | Docs author | Reviewer)





The impact of AI

A modern AI coding agent (for example Cursor, Claude Code) can quickly generate a PR that looks impressive.

However they have no understanding. This generally leads to frustration.

You can use whatever tools you like to work on Prometheus, but you must understand the PR that you submit.





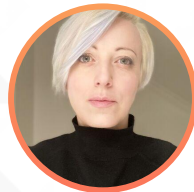
What is reviewing?

Reviewing is a lot more work than you might think.

The purpose is to maintain quality in the Prometheus code-base.

Quality means the code does what it is supposed to do, fairly efficiently, doesn't crash or lose your data.

We also care about maintainability - can the next person understand the code so they can extend or fix it?





Be kind.

Comment on what is written, don't assume intent.

- "In line 45, the code may use a nil pointer"
- vs "You forgot to check for a nil pointer".

Understandability - is it clear what change is being made?

Consistency - does the title cover what the change does?

Coherence - does the PR make more than one unrelated change?

We use linters to identify many common pitfalls in Go coding; reviewers should not focus on minor points of style.





Reviewing tips

Checkout the code locally!

```
git clone https://github.com/prometheus-community/prometheus
```

```
git fetch origin pull/ID/head:BRANCH_NAME
```

- Check CI to see what test fails!
- Look at broader code context from the diff and failing tests.
- Investigate and begin your review from there :)





Best Practices

We want contributions to the Prometheus ecosystem to follow best practices.

- https://prometheus.io/docs/instrumenting/writing_exporters/
- <https://prometheus.io/docs/practices/naming/>
- <https://prometheus.io/docs/practices/instrumentation/>

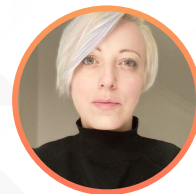
There is also shared libraries and infrastructure to make building and publishing consistent.

- <https://github.com/prometheus/exporter-toolkit>
- <https://github.com/prometheus/golang-builder>
- <https://github.com/prometheus/promci>



What is special about Prometheus?

- Second oldest CNCF project
- Vast ecosystem
- 24/7 critical infrastructure
- Upgrade without fear





Now You Try! [30 min]

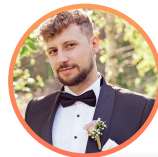
(A) For total beginners: Pair with others!

(B) For beginners: Pick github.com/prometheus-community/prometheus workshop PR (we will distribute now) and **review it**. Find **all the problems** that in your opinion **prevents** this fix to be merged into Prometheus main.

After 25m we will **do the review together** and **compare vs what you found!**

(C) For experienced developers: Feel free to pick **any prometheus/prometheus** PR to review and we can **check your review after** you (today or async). Put *"Review created as part of KubeCon EU 2026 ContribFest"* in your review.

There are lots of Prometheus maintainers in the room:
If you need help, raise your hand!





About the example PRs

Set and check (and document) a global `label_value_length_limit` [#16525](#)

Proposal

Prometheus has `label_name_length_limit` and `label_value_length_limit`, currently defaulted to no limit.

After [#16069](#), Prometheus will crash if you give it a name or value longer than 16MB, so we should at least have a default limit lower than that.

I think that a reasonable limit would be more like 200 bytes, but I am aware of people doing exciting things like storing a bitmap in a label. Maybe we could go with something like 1MB?

The code checking limits while scraping would need to check before constructing the `Labels` data structure, and we should check in other places such as the remote-write receiver, `label_replace`, `label_join`.

Let's start reviewing!

Reflection Time!

What have you found?



Things you could have found

config/config.go line 194: Consts are inconsistent.

config/config.go line 707: Value limit is not defaulted

config/config_test.go line 2934: Should be Name then Value

promql/functions.go line 2003: Hard-codes limit; should use configured value

(also line promql/functions.go 2057, scrape/scrape.go line 1718)

storage/remote/write_handler_test.go: No test cases use specific limits, all 0



Thank You!

Summary:

Our GitHub handles:

@bboreham? @bwplotka, @superq, @saswata @ves... ?



Prometheus user survey:

<https://forms.gle/uuEsawKm7u9wCT4T8>