KubeCon | CloudNativeCon

Europe 2023

KubeCon | CloudNativeCon

Europe 2023

Cortex: How to Run a Rock Solid Multi-Tenant Prometheus

Friedrich Gonzalez, Adobe & Alan Protasio, AWS

Agenda

- Cortex Introduction
- What is the latest on cortex?
- Reliability with Users in mind
- Secret Sauce
- Coming up next!
- Questions

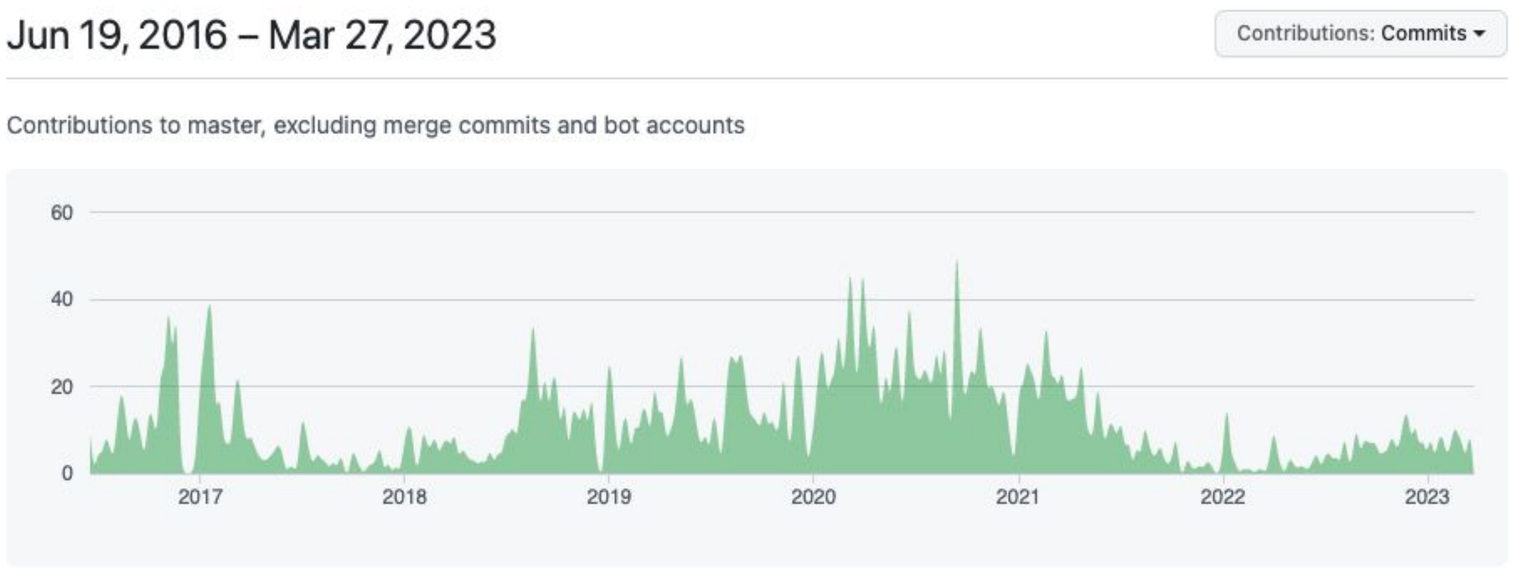
Cortex

cortex

Horizontally Scalable, highly available, multi-tenant, long term storage for Prometheus

Cortex

- Created in 2016
- CNCF Incubating project
- Apache 2 license


Jun 19, 2016 – Mar 27, 2023

Contributions to master, excluding merge commits and bot accounts

Speakers & Maintainers

Friedrich Gonzalez

Software Engineer
Adobe

@friedrichg

Alan Protasio

Senior Software
Development Engineer
AWS

@alanprot

Other Maintainers

Alvin Lin
Software Dev Manager
AWS

@alvinlin123

Ben Ye
Software Engineer
AWS

@yeya24

Helm Chart Maintainers

Tom Hayward
Infoblox

@kd7lxl

Niclas Schad
STACKIT
@nschad

Other Key Contributors

Harry John
@harry671003

Daniel Blando
@danielblando

Alex Le
@alexqyle

Xiaochao Dong
@damnever

Leon Wang
@wgliang

songjiang
@songjiang

Roy C
@roystchiang

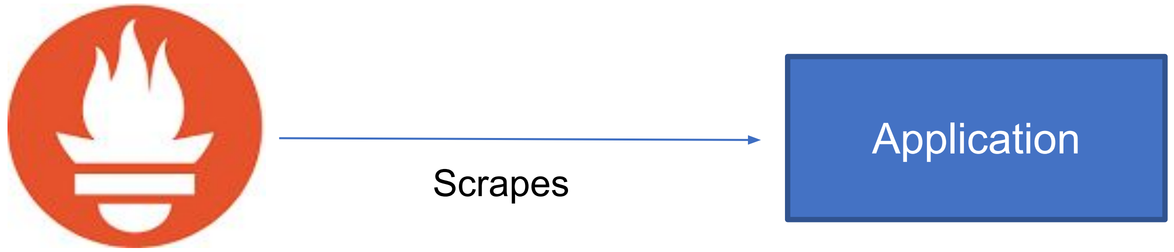
Kama Huang
@kama910

Mengmeng Yang
@mengmengy

Matthew Ames
@SuperMatt

Cortex: Why use Multitenancy?

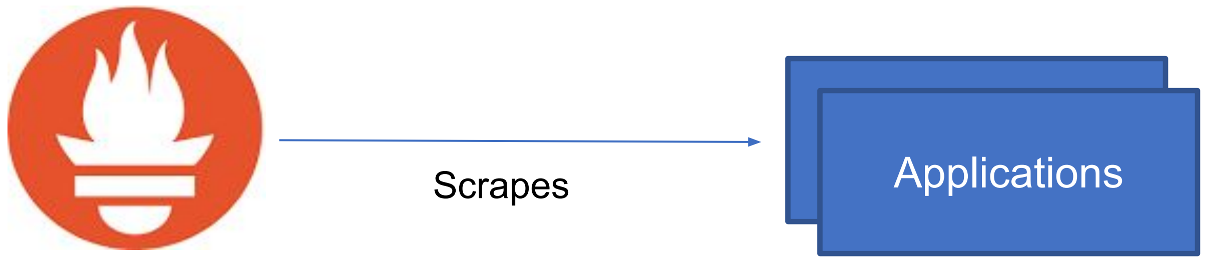
A. Prometheus scraping a single application (job)

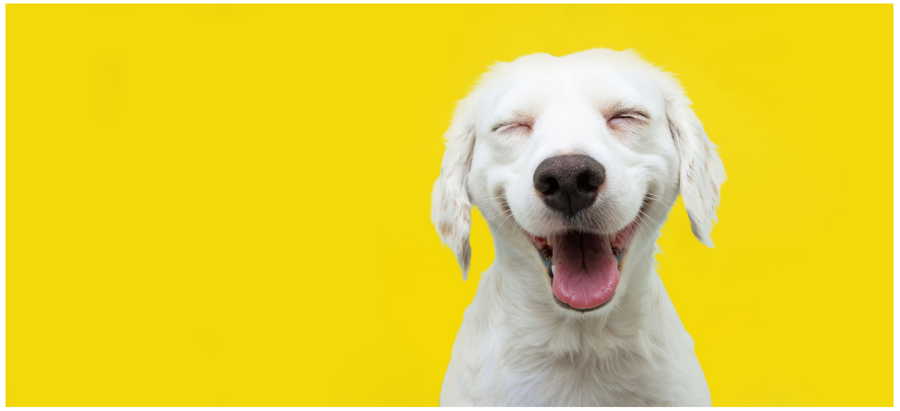


Scenarios:
- High cardinality
- High churn
- Increasing Data Retention

Cortex: Why use Multitenancy?

B. Prometheus scraping multiple applications (multiple jobs) for single owner

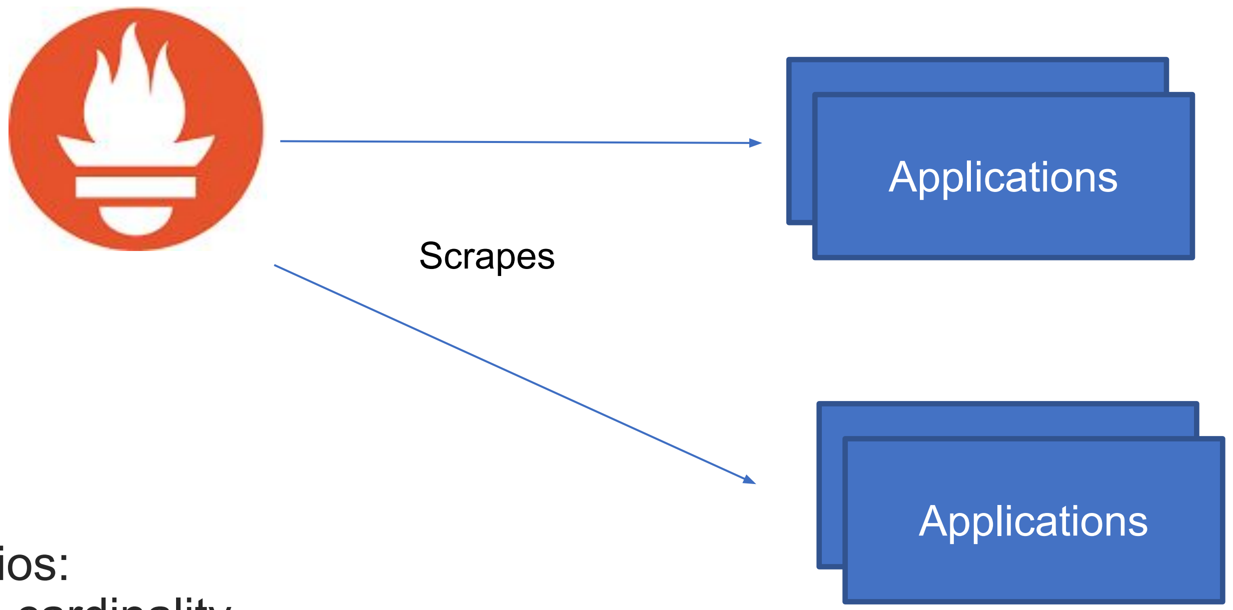




Scenarios:
- High cardinality
- High churn
- Increasing Data Retention

Cortex: Why use Multitenancy?

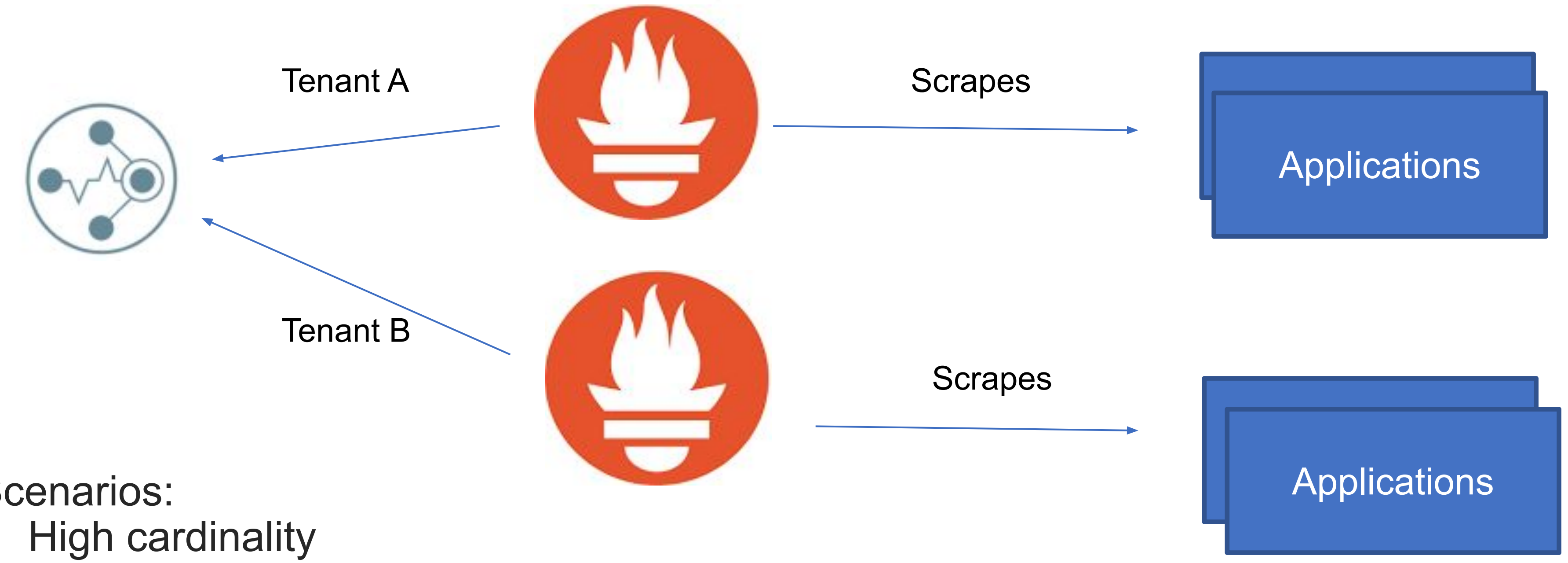
C. Prometheus scraping multiple applications from different teams (multiple jobs with different owners)



Scenarios:
- High cardinality
- High churn
- Increasing Data Retention

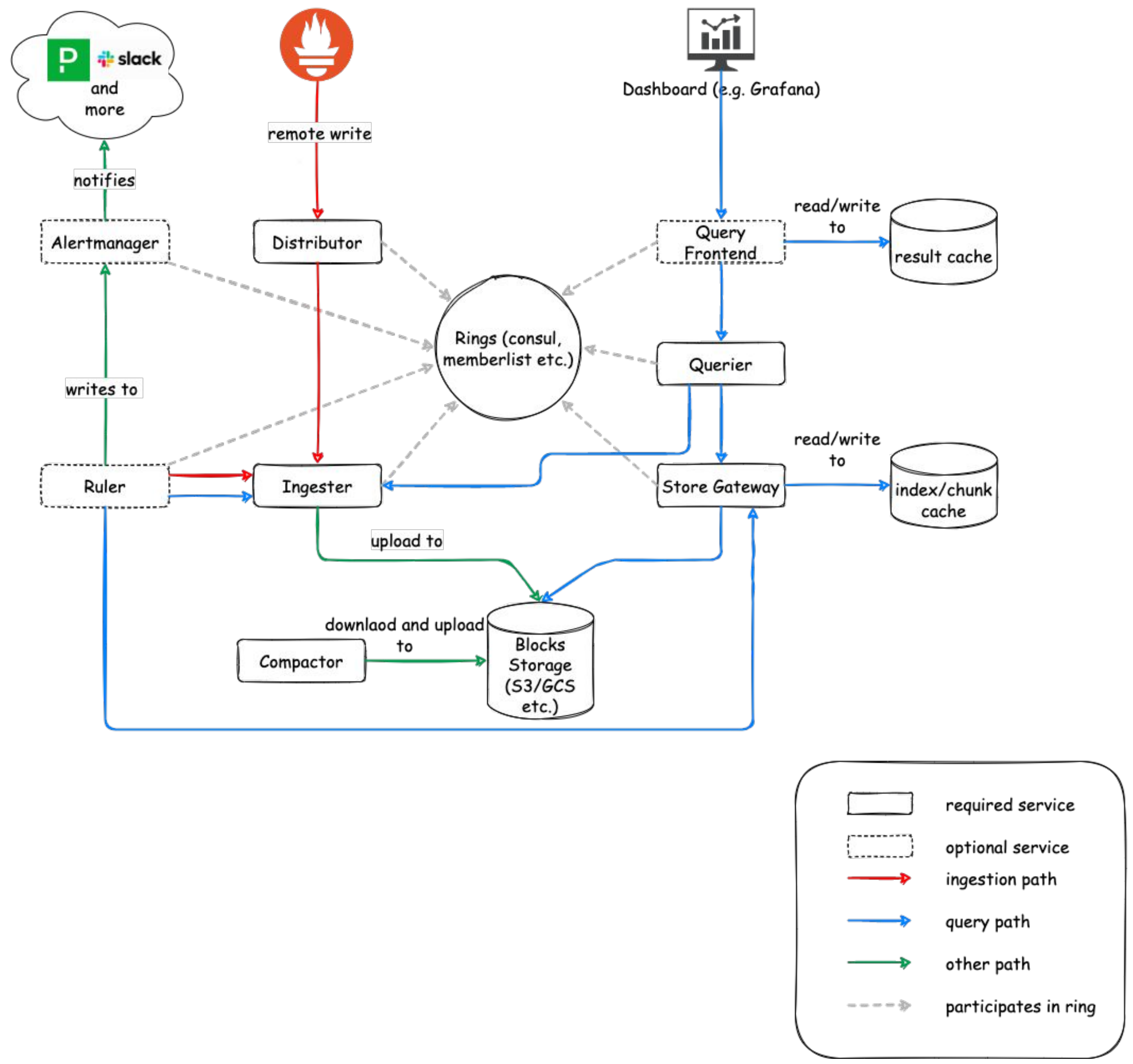
Cortex: Why use Multitenancy?

D. Cortex provides long term retention while providing tenant isolation

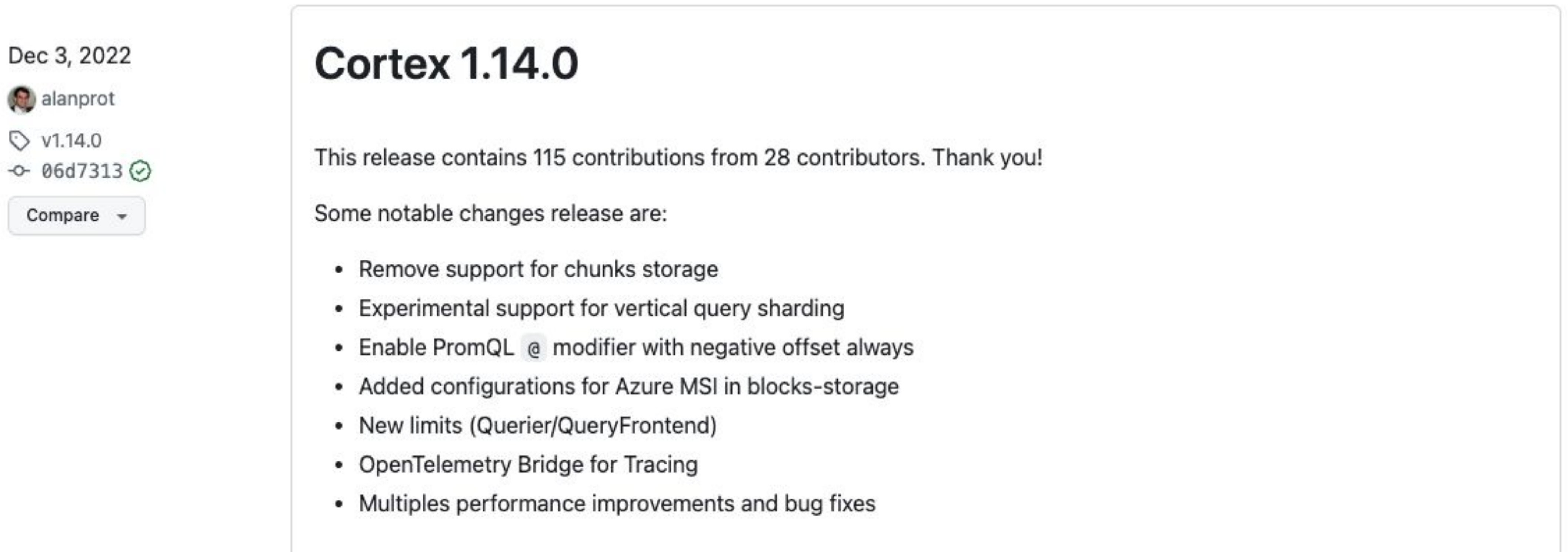


Scenarios:
- High cardinality
- High churn
- Increasing Data Retention

Cortex’s Architecture



What is the latest on cortex?



Dec 3, 2022

alanprot

v1.14.0

06d7313

Compare

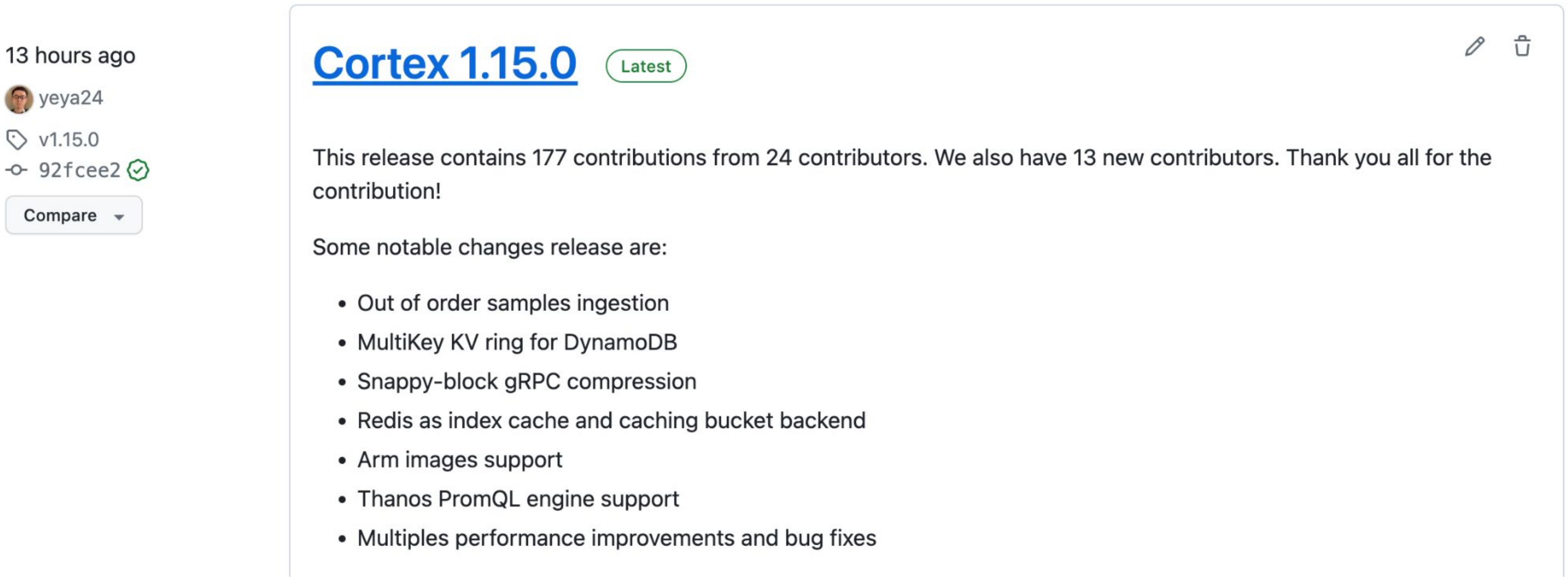
Cortex 1.14.0

This release contains 115 contributions from 28 contributors. Thank you!

Some notable changes release are:

- Remove support for chunks storage
- Experimental support for vertical query sharding
- Enable PromQL `@` modifier with negative offset always
- Added configurations for Azure MSI in blocks-storage
- New limits (Querier/QueryFrontend)
- OpenTelemetry Bridge for Tracing
- Multiples performance improvements and bug fixes

What is the latest on cortex?



13 hours ago

yeya24

v1.15.0

92fcee2

Compare

Cortex 1.15.0 Latest

This release contains 177 contributions from 24 contributors. We also have 13 new contributors. Thank you all for the contribution!

Some notable changes release are:

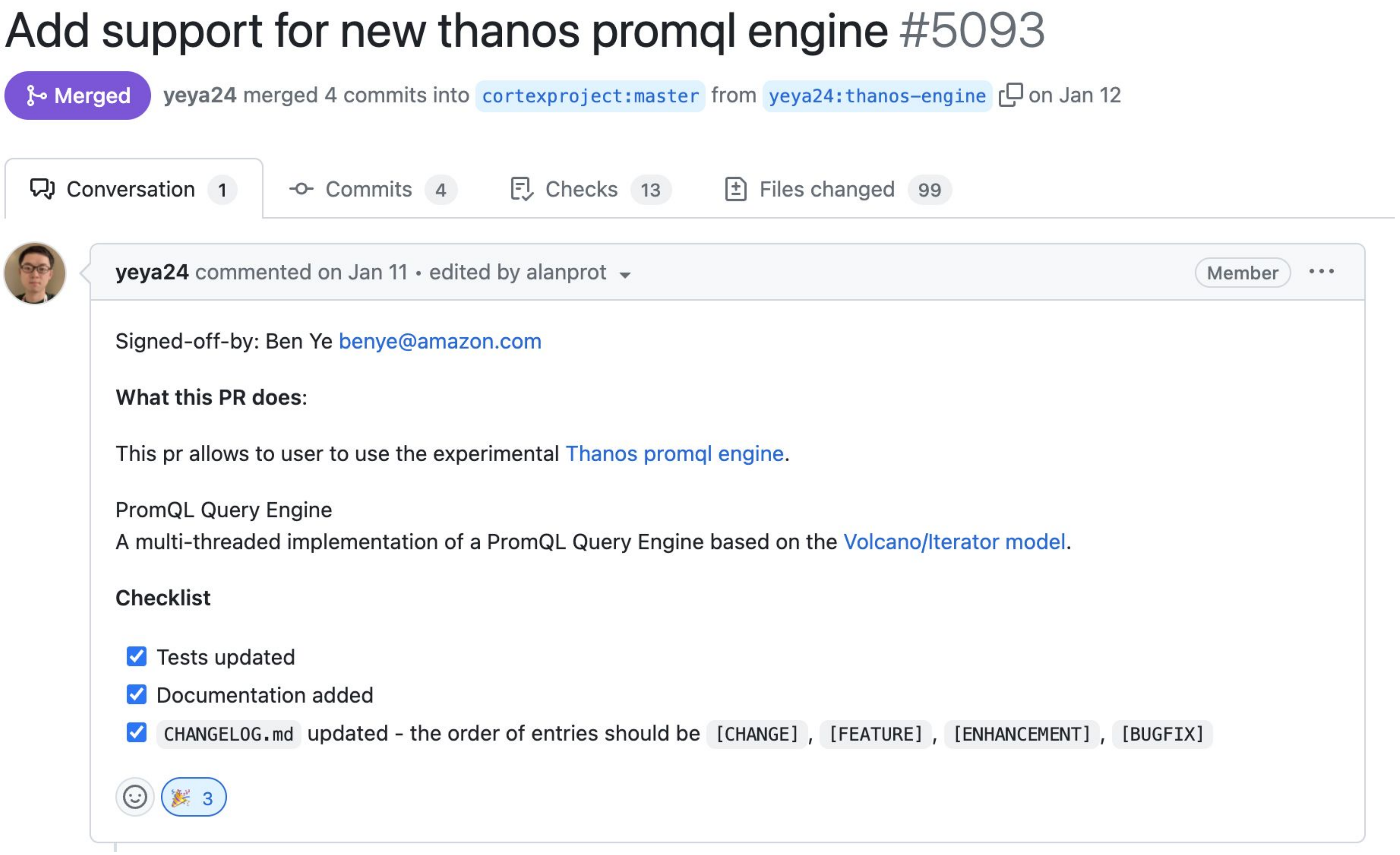
- Out of order samples ingestion
- MultiKey KV ring for DynamoDB
- Snappy-block gRPC compression
- Redis as index cache and caching bucket backend
- Arm images support
- Thanos PromQL engine support
- Multiples performance improvements and bug fixes

New Cortex Maintainer

Ben Ye
Software Engineer
AWS
Thanos Maintainer

@yeya24

What is the latest on cortex?



Add support for new thanos promql engine #5093

Merged yeya24 merged 4 commits into `cortexproject:master` from `yeya24:thanos-engine` on Jan 12

Conversation 1 | Commits 4 | Checks 13 | Files changed 99

yeya24 commented on Jan 11 • edited by alanprot | Member

Signed-off-by: Ben Ye benye@amazon.com

What this PR does:

This pr allows to user to use the experimental Thanos promql engine.

PromQL Query Engine
A multi-threaded implementation of a PromQL Query Engine based on the Volcano/Iterator model.

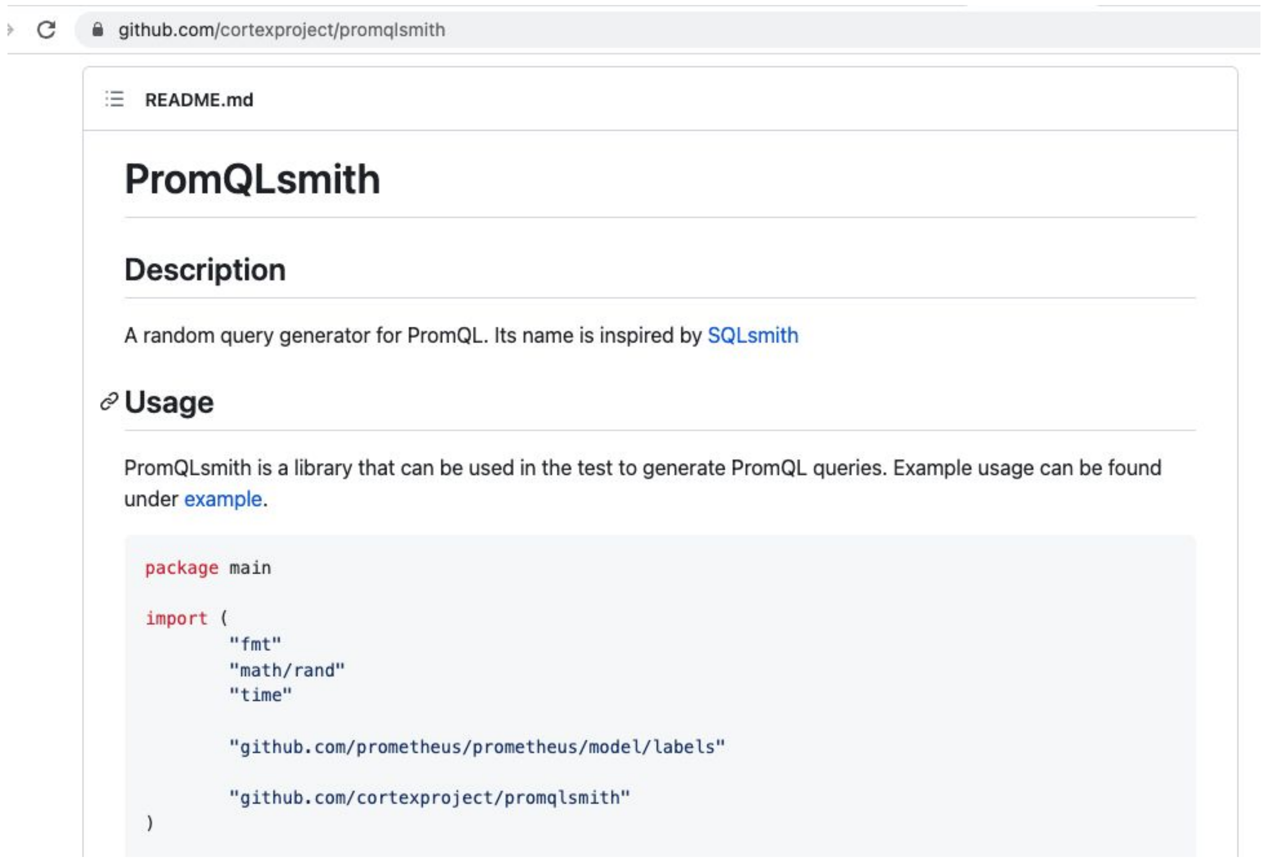
Checklist

- [x] Tests updated
- [x] Documentation added
- [x] `CHANGELOG.md` updated - the order of entries should be `[CHANGE]`, `[FEATURE]`, `[ENHANCEMENT]`, `[BUGFIX]`

🎉 3

What is latest on cortex?

https://github.com/cortexproject/promqlsmith



github.com/cortexproject/promqlsmith

README.md

PromQLsmith

Description

A random query generator for PromQL. Its name is inspired by SQLsmith

Usage

PromQLsmith is a library that can be used in the test to generate PromQL queries. Example usage can be found under example.

```
package main

import (
        "fmt"
        "math/rand"
        "time"

        "github.com/prometheus/prometheus/model/labels"

        "github.com/cortexproject/promqlsmith"
)
```

Reliability with Users in mind

What is the single thing users want from monitoring?

Reliability with Users in mind

Measure it

- Service Level Indicators with errors and latency

```
  sum by (job) (
    rate(
      cortex_request_duration_seconds_bucket{le="1.0",route="/cortex.Ingester/Push",status_code!~"5.."}[5m]
    )
  )
/
  sum by (job) (
    rate(
      cortex_request_duration_seconds_count{route="/cortex.Ingester/Push"}[5m]
    )
  )
```

Reliability with Users in mind

Service Level Objectives

- Be realistic.
- Refine your promise over time
- Page on SLO breaches

2019-11-20 *How to Include Latency in SLO-based Alerting* KubeCon + CloudNativeCon, San Diego, CA, USA

Reliability with Users in mind

Empathize with users and help them achieve their objectives

Secret Sauce

- Kubernetes Ready
- Prometheus based
- Thanos based

Tenant overrides & limits

```
medium_user:: {

  max_series_per_query: 100000,

  max_global_series_per_user: 3000000,  // 3M
  max_global_series_per_metric: 300000,  // 300K

  ingestion_rate: 350000,  // 350K
  ingestion_burst_size: 3500000,  // 3.5M

  // 1800 rules
  ruler_max_rules_per_rule_group: 20,
  ruler_max_rule_groups_per_tenant: 90,
},
```

```
big_user:: {

  max_series_per_query: 100000,

  max_global_series_per_user: 6000000,  // 6M
  max_global_series_per_metric: 600000,  // 600K

  ingestion_rate: 700000,  // 700K
  ingestion_burst_size: 7000000,  // 7M

  // 2200 rules
  ruler_max_rules_per_rule_group: 20,
  ruler_max_rule_groups_per_tenant: 110,
},
```

Tenant overrides & limits

```
ingestion_rate: 700000
ingestion_rate_strategy: local
ingestion_burst_size: 7000000
accept_ha_samples: false
ha_cluster_label: cluster
ha_replica_label: __replica__
ha_max_clusters: 0
drop_labels: []
max_label_name_length: 1024
max_label_value_length: 2048
max_label_names_per_series: 30
max_labels_size_bytes: 0
max_metadata_length: 1024
reject_old_samples: false
reject_old_samples_max_age: 2w
creation_grace_period: 10m
enforce_metadata_metric_name: true
enforce_metric_name: true
ingestion_tenant_shard_size: 60
max_exemplars: 0
max_series_per_query: 100000
max_series_per_user: 0
max_series_per_metric: 0
max_global_series_per_user: 6000000
max_global_series_per_metric: 600000
max_metadata_per_user: 8000
max_metadata_per_metric: 10
max_global_metadata_per_user: 0
max_global_metadata_per_metric: 0
out_of_order_time_window: 0s
max_fetched_chunks_per_query: 0
max_fetched_series_per_query: 0
max_fetched_chunk_bytes_per_query: 0
max_fetched_data_bytes_per_query: 0
max_query_lookback: 0s
max_query_length: 0s
max_query_parallelism: 14
max_cache_freshness: 1m
max_queriers_per_tenant: 0
query_vertical_shard_size: 0
max_outstanding_requests_per_tenant: 100
ruler_evaluation_delay_duration: 0s
ruler_tenant_shard_size: 0
ruler_max_rules_per_rule_group: 20
ruler_max_rule_groups_per_tenant: 110
store_gateway_tenant_shard_size: 0
compactor_blocks_retention_period: 0s
compactor_tenant_shard_size: 0
s3_sse_type: ""
s3_sse_kms_key_id: ""
s3_sse_kms_encryption_context: ""
alertmanager_receivers_firewall_block_cidr_networks: ""
alertmanager_receivers_firewall_block_private_addresses: false
alertmanager_notification_rate_limit: 0
alertmanager_notification_rate_limit_per_integration: {}
alertmanager_max_config_size_bytes: 0
alertmanager_max_templates_count: 0
alertmanager_max_template_size_bytes: 0
alertmanager_max_dispatcher_aggregation_groups: 0
alertmanager_max_alerts_count: 0
alertmanager_max_alerts_size_bytes: 0
```

Instance limits - Ingesters

```
instance_limits:
  # Max ingestion rate (samples/sec) that ingester will accept. This limit is
  # per-ingester, not per-tenant. Additional push requests will be rejected.
  # Current ingestion rate is computed as exponentially weighted moving average,
  # updated every second. This limit only works when using blocks engine. 0 =
  # unlimited.
  # CLI flag: -ingester.instance-limits.max-ingestion-rate
  [max_ingestion_rate: <float> | default = 0]

  # Max users that this ingester can hold. Requests from additional users will
  # be rejected. This limit only works when using blocks engine. 0 = unlimited.
  # CLI flag: -ingester.instance-limits.max-tenants
  [max_tenants: <int> | default = 0]

  # Max series that this ingester can hold (across all tenants). Requests to
  # create additional series will be rejected. This limit only works when using
  # blocks engine. 0 = unlimited.
  # CLI flag: -ingester.instance-limits.max-series
  [max_series: <int> | default = 0]

  # Max inflight push requests that this ingester can handle (across all
  # tenants). Additional requests will be rejected. 0 = unlimited.
  # CLI flag: -ingester.instance-limits.max-inflight-push-requests
  [max_inflight_push_requests: <int> | default = 0]
```

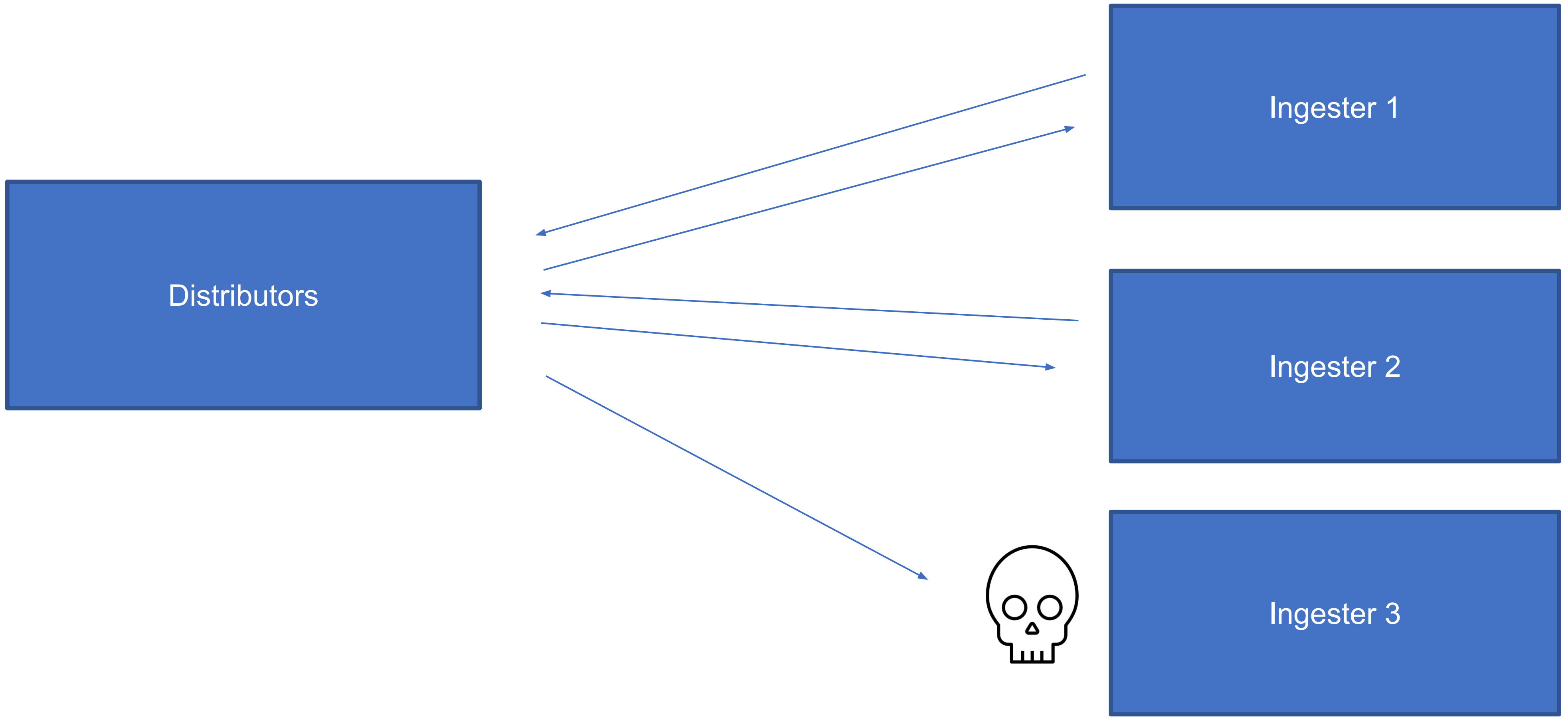
Instance limits - Distributors

```
instance_limits:
  # Max ingestion rate (samples/sec) that this distributor will accept. This
  # limit is per-distributor, not per-tenant. Additional push requests will be
  # rejected. Current ingestion rate is computed as exponentially weighted
  # moving average, updated every second. 0 = unlimited.
  # CLI flag: -distributor.instance-limits.max-ingestion-rate
  [max_ingestion_rate: <float> | default = 0]

  # Max inflight push requests that this distributor can handle. This limit is
  # per-distributor, not per-tenant. Additional requests will be rejected. 0 =
  # unlimited.
  # CLI flag: -distributor.instance-limits.max-inflight-push-requests
  [max_inflight_push_requests: <int> | default = 0]
```

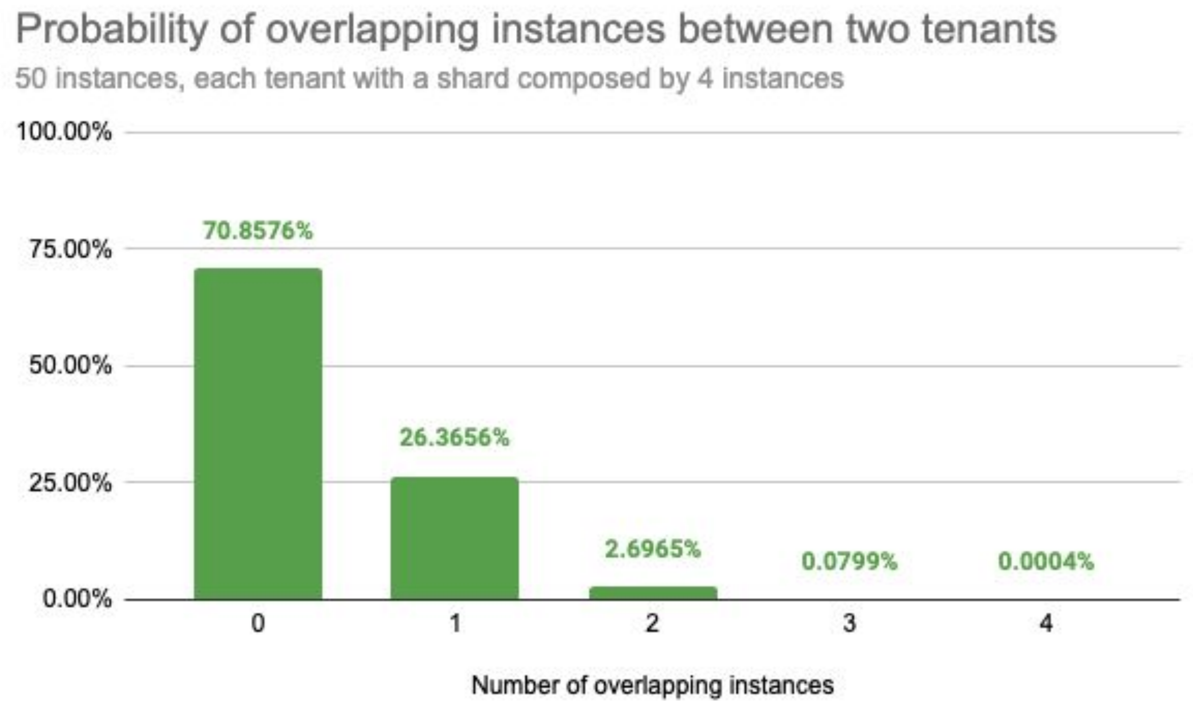
Replication Factor and Quorum

- Succeed write request when receiving at least 2 out of 3 positive responses.
- Succeed read when receiving 2 out of 3 positive responses.



Shuffle Sharding

- Known best practice for workload isolation
- Reduce the blast radius of an outage
- Reduce impact of a misbehaving tenant affecting other tenants.



Consistent Hashing: Why?

- Ease scaling: Add new nodes without much rebalancing
- Requires a key/value database.

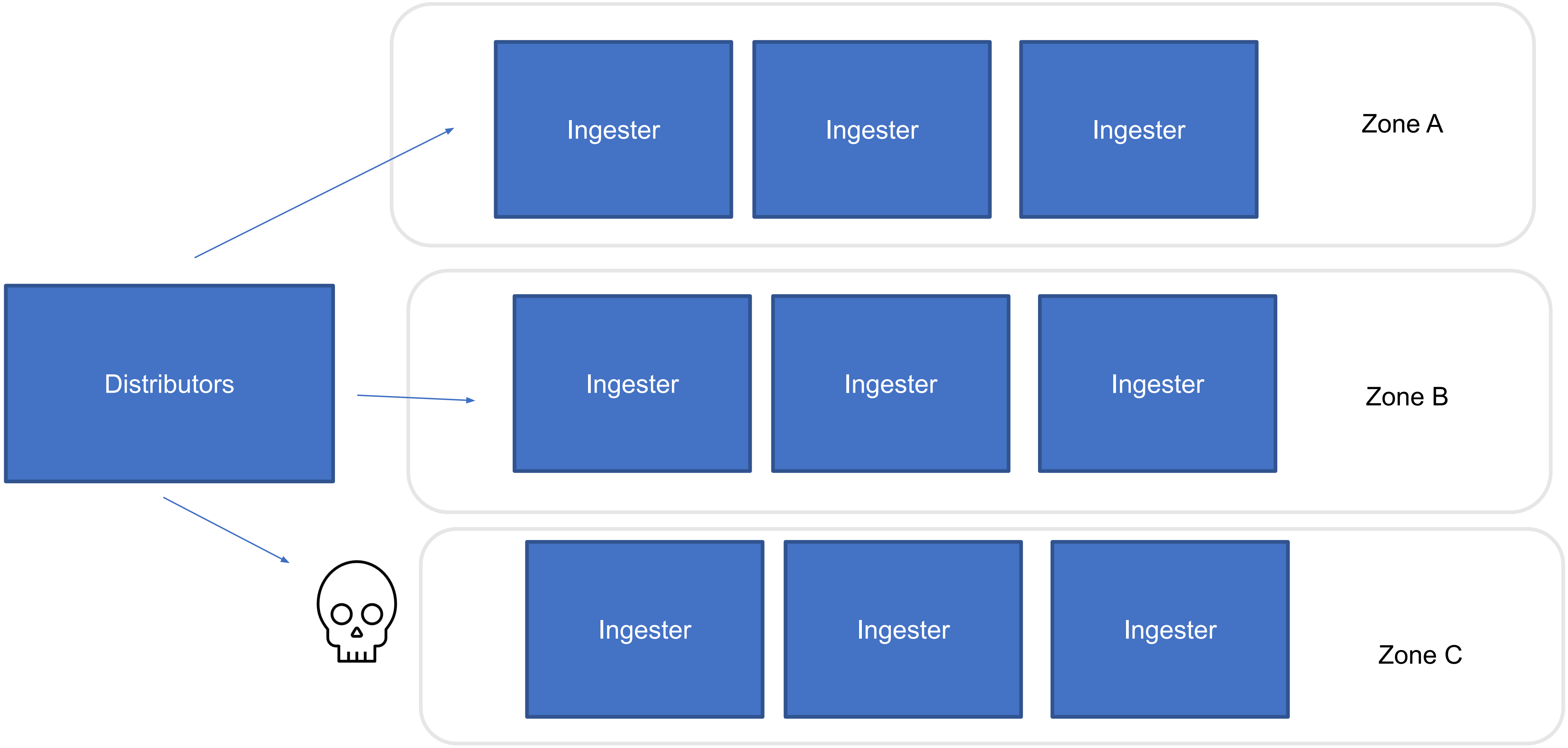
Rings:

- Consul (2016)
- Etcd (2019)
- Memberlist (2019)
- Dynamodb (2023)

Now also with multikey the hot key problem is gone

Zone aware replication

One replication zone can fail and ingestion and reading stays unaffected

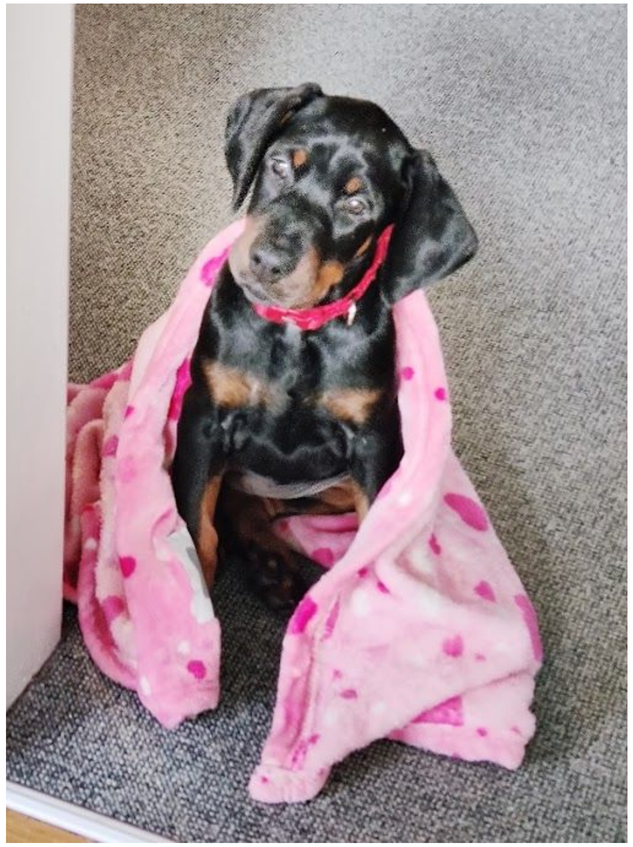


Coming Up Soon

- Auth-Gateway (LFX mentoring project)
- Importing Prometheus TSDB (LFX mentoring project)
- Downsampling
- Federated rules
- Native Histograms

Community

- Join Cortex CNCF channel and let's talk
- Vote on issues, tell us what is important to you
- Help is welcomed!



Please scan the QR Code above
to leave feedback on this session