



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

Discover Cortex: High Scalability Metrics in 2026

Friedrich Gonzalez & Charlie Le, Apple



Agenda

Introduction

Key improvements from the latest 1.20 release

Protocol Improvements

Prometheus Remote Write 2.0 + Native Histograms

OTLP Ingestion

What is coming for 1.21 release

Overrides API

Are my metrics queried at all?

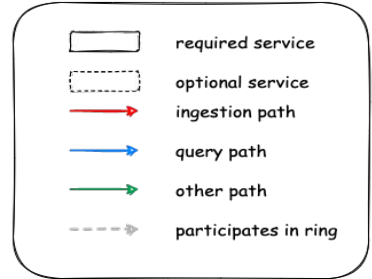
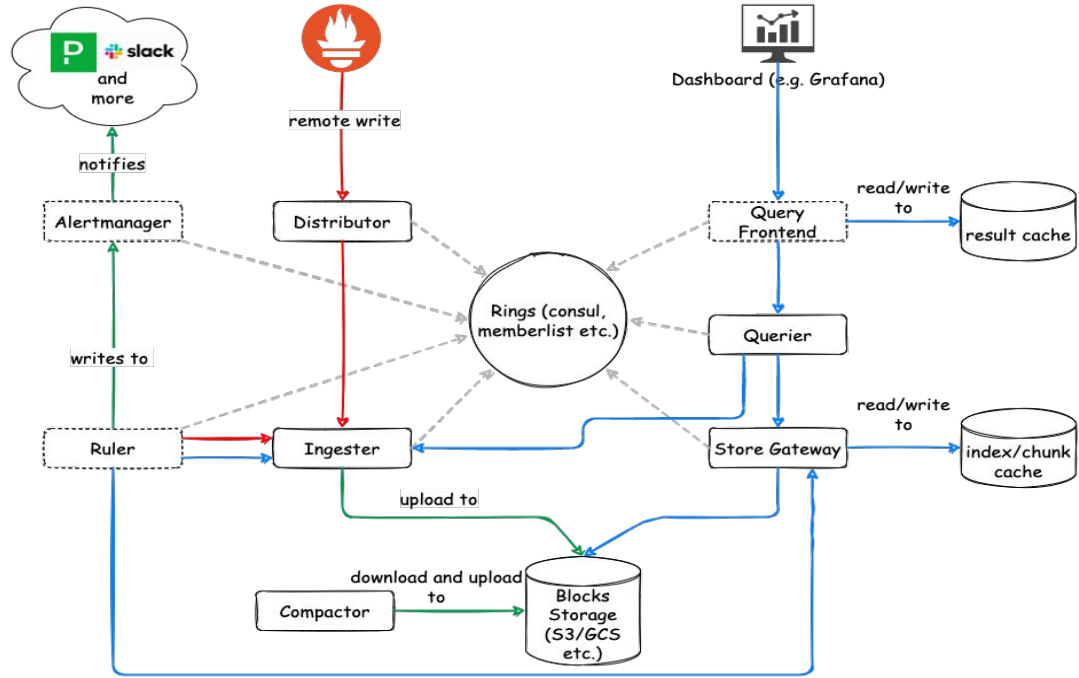
CNCF road to Graduation

Q&A with core maintainers

Get Involved

Introduction

Horizontally scalable,
highly available,
multi-tenant, long term
storage solution for
Prometheus and
OpenTelemetry Metrics.



Remote Write & Ingestion

Prometheus Remote Write 2.0 - Experimental support

gRPC Stream Push - Experimental stream-based push between distributors and ingesters for improved efficiency

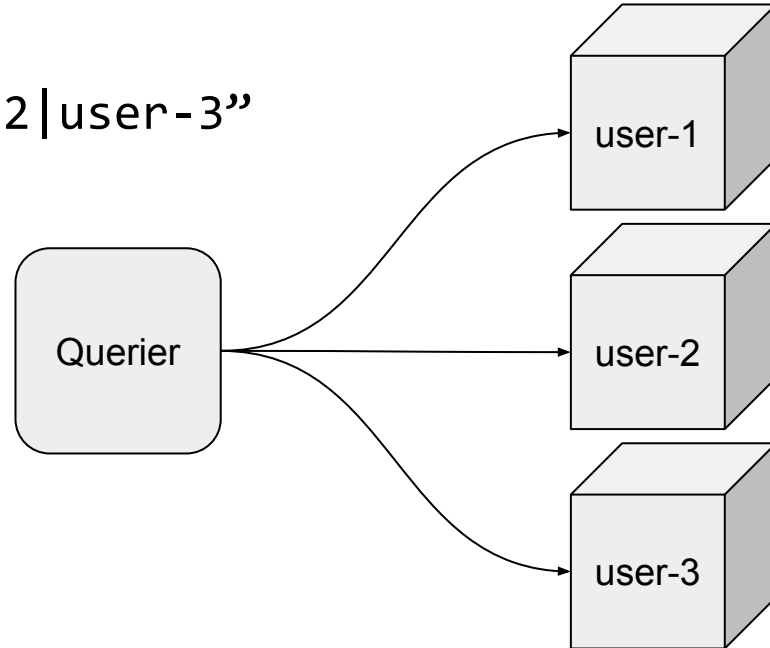
Native Histograms - Out-of-order ingestion support, per-tenant config, active series metrics, and ingestion rate limits

Multi-Tenancy

Regex Tenant Resolver - Pattern matching in X-Scope-OrgID header for flexible multi-tenant federation queries

X-Scope-OrgID: “user-1|user-2|user-3”

X-Scope-OrgID: “user-[1-3]”



Dynamic Query Splitting

Query frontend auto-adjusts split intervals to maintain shard and duration limits

```
query_range:  
  split_queries_by_interval: 24h  
  dynamic_query_splits:  
    max_shards_per_query: 100  
    max_fetched_data_duration_per_query: 8760h # 365 day  
    enable_dynamic_vertical_sharding: true
```

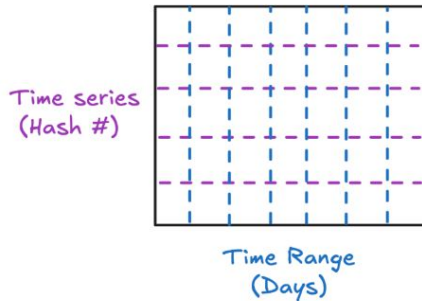
```
limits:  
  query_vertical_shard_size: 4
```

Static Splitting

The query time range is split by a fixed interval and time series are sharded into fixed number of subsets.

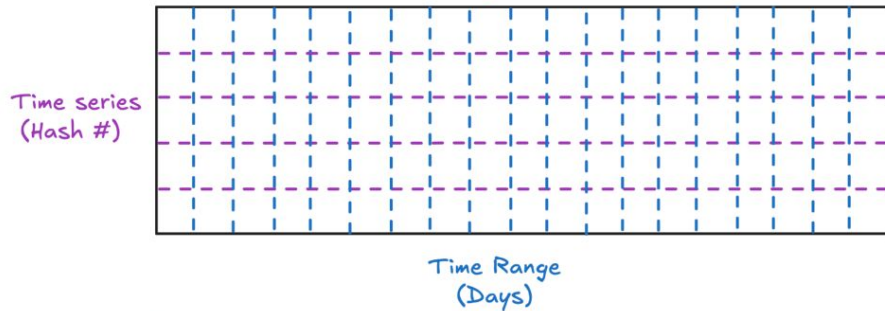
Short Range Query

The query has a **healthy** amount of sharding



Long Range Query

The query results in **too much sharding** given a static interval and vertical shard count

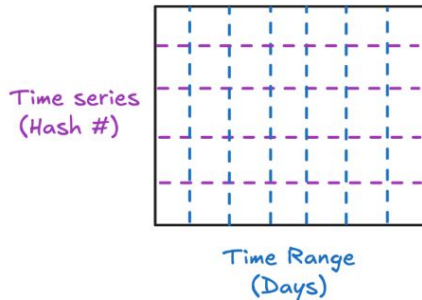


Static Splitting

The query time range is split by a fixed interval and time series are sharded into fixed number of subsets.

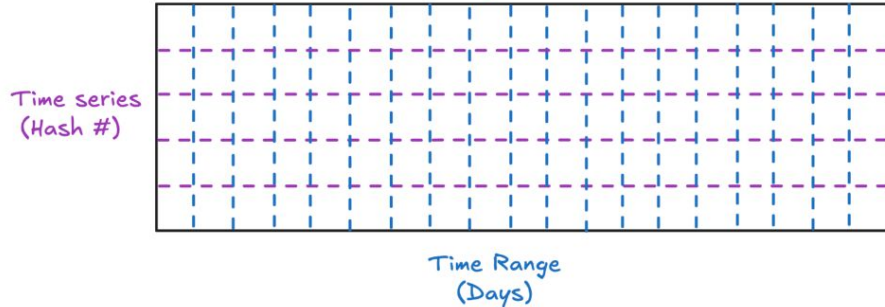
Short Range Query

The query has a **healthy** amount of sharding



Long Range Query

The query results in **too much sharding** given a static interval and vertical shard count

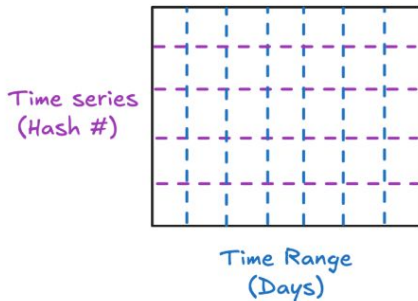


Dynamic Splitting

An optimal combination of split interval and vertical shard count is used to achieve maximum parallelism within a configured target

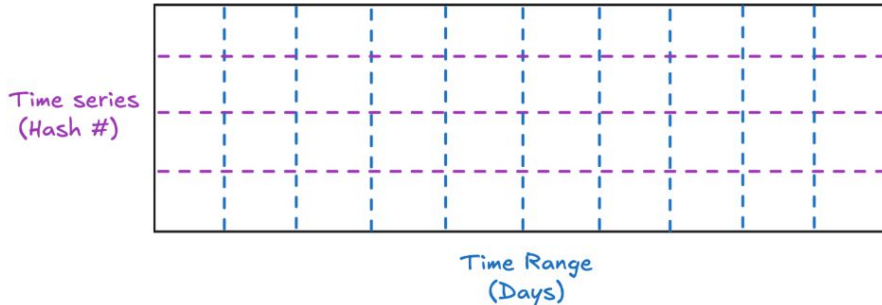
Short Range Query

The query has a **healthy** amount of sharding



Long Range Query

Split interval and vertical shard count are adjusted to maintain a **healthy** amount of sharding



Blog

[READONLY Ingestor Scaling](#)[Query Priority in Cortex](#)**[Dynamic Query Splitting](#)**[Query Rejection in Cortex](#)[Optimizing PromQL queries](#)[Hello Cortex!](#)

Efficient Query Parallelism in Cortex with Dynamic Query Splitting

This article explores the motivations behind adapting dynamic query splitting in Cortex, how the dynamic model works, and how to configure it for more efficient scalable PromQL query execution.

By **Ahmed Hassan (@afhassan)** | Tuesday, August 12, 2025

Tags: [blog](#) [cortex](#) [query](#) [optimization](#)

Categories: [blog](#)

Introduction

Cortex traditionally relied on **static query splitting** and **static vertical sharding** to optimize the execution of long-range PromQL queries. Static query splitting divides a query into fixed time intervals, while vertical sharding—when applicable—splits the query across subsets of time series. These techniques offered improved parallelism and reduced query latency but were limited by their one-size-fits-all approach. They did not account for differences in query range, lookback behavior, and cardinality—leading to inefficiencies like over-sharding, redundant data fetches, and storage pressure in large or complex queries.

To address those gaps, Cortex introduced **dynamic query splitting** and **dynamic vertical sharding**—two adaptive mechanisms that intelligently adjust how queries are broken down based on query semantics.

This article explores the motivations behind this evolution, how the dynamic model works, and how to configure it for more efficient scalable PromQL query execution.

Query Splitting

Query splitting breaks a single long-range query into smaller subqueries based on a configured time interval. For example, given this configuration, a 30-day range query will be split into 30 individual 1-day subqueries:

```
query_range:  
  split_queries_by_interval: 24h
```

RSS

[Edit this page](#)[Create documentation issue](#)[Create project issue](#)[Introduction](#)[Query Splitting](#)[Vertical Sharding](#)[Introducing Dynamic Query](#)[Splitting](#)[1. Queuing and Merge](#)[Bottlenecks from Over-Splitting](#)[2. Parallelism Cost with Query](#)[Lookback](#)[How to Configure Dynamic Query](#)[Splitting](#)[Example Configuration](#)[Query #1](#)[Query #2](#)[Conclusion](#)

Our Projects

[cortex 6](#)

Tag Cloud

[blog 6](#) [community 1](#)[cortex 6](#) [ingester 1](#)[optimization 3](#) [query 4](#)[rejection 1](#) [scaling 1](#)[updates 1](#)

Reliability

- ResourceMonitor
- ResourceBasedLimiter

New components to prevent service overload under high demand

```
target: ingester
```

```
monitored_resources: cpu,heap
```

```
ingester:
```

```
  query_protection:
```

```
    rejection:
```

```
      enabled: true
```

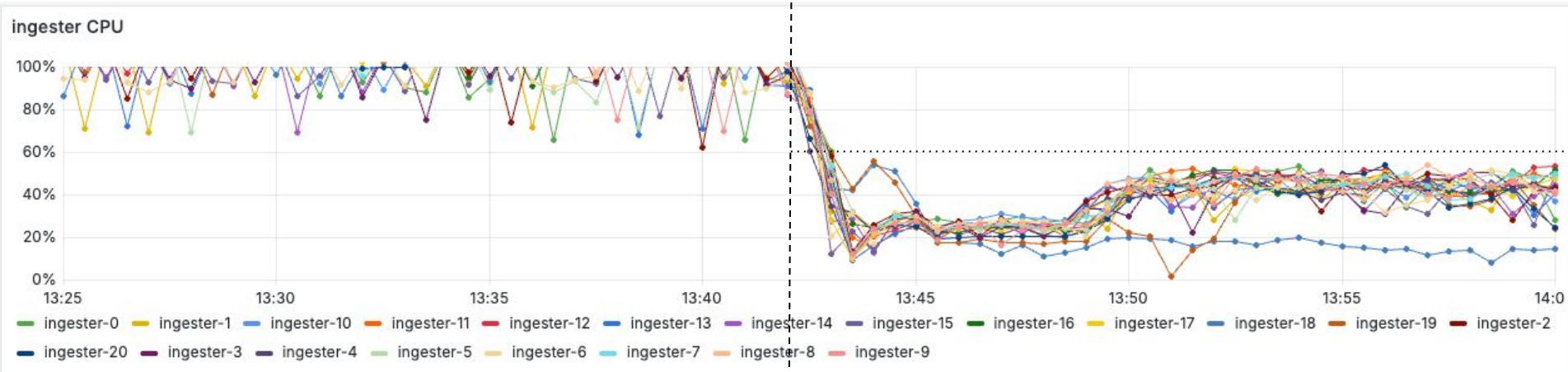
```
      threshold:
```

```
        cpu_utilization: 0.6
```

```
        heap_utilization: 0.8
```

No Limit

60% CPU Limit Applied



Running Cortex on
Kubernetes

Getting started with a
gossip ring cluster

Alertmanager
configuration

Authentication and
Authorisation

Capacity Planning

Config for horizontally
scaling the Ruler

Config for sending HA
Pairs data to Cortex

Encryption at Rest

Ingesters rolling updates

Ingesters scaling up and
down

Migration KV Store to
Memberlist

Resource-based throttling (Experimental)

Although the static limits are able to protect Cortex components from specific query patterns, they are not generic enough to cover different combinations of bad query patterns. For example, what if the query fetches relatively large postings, series and chunks that are slightly below the individual limits? For a more generic solution, you can enable resource-based throttling by setting CPU and heap utilization thresholds.

Currently, it only throttles incoming query requests with error code 429 (too many requests) when the resource usage breaches the configured thresholds.

For example, the following configuration will start throttling query requests if either CPU or heap utilization is above 80%, leaving 20% of room for inflight requests.

```
target: ingester
monitored_resources: cpu,heap
ingester:
  query_protection:
    rejection:
      enabled: true
      threshold:
        cpu_utilization: 0.8
        heap_utilization: 0.8
```

See https://cortexmetrics.io/docs/configuration/configuration-file/--:text=query_protection for details.

query

Max chunks fetched per (sharded)

query

Max samples fetched per
(sharded) query

Resource-based throttling
(Experimental)

Our Projects

cortex 6

Tag Cloud

blog 6 community 1

cortex 6 ingester 1

optimization 3 query 4

rejection 1 scaling 1

updates 1



cortex

Community

Slack

GitHub

Twitter

About


Cortex is an OSS licensed project as Apache License 2.0

Compatibility

UTF-8 Support - `-name-validation-scheme` flag enables UTF-8 metric name validation

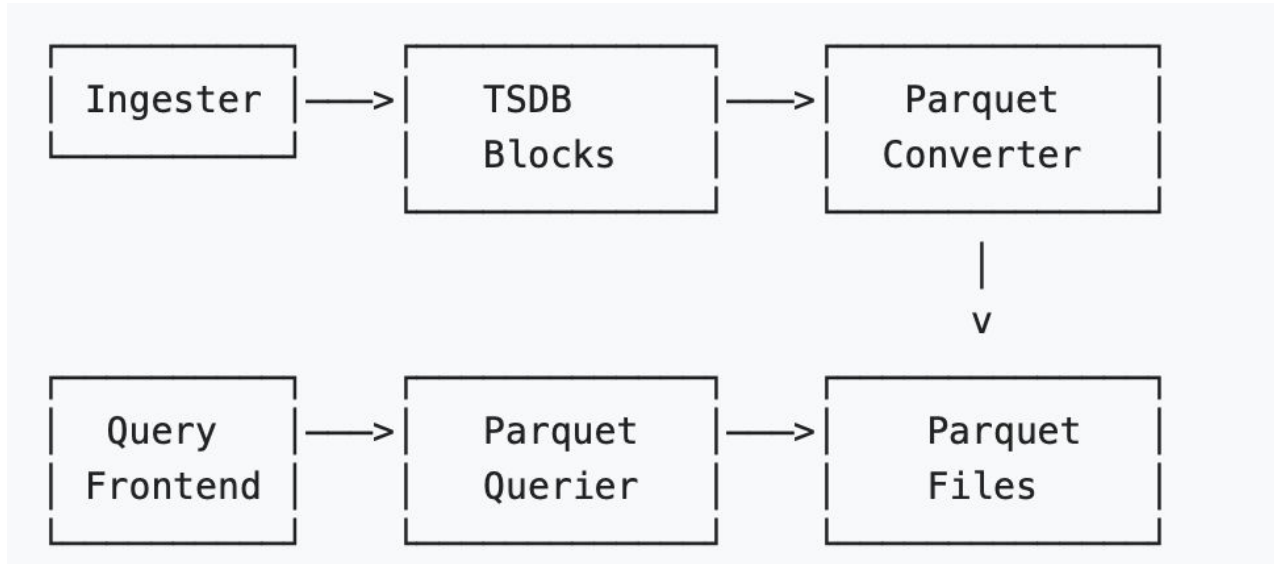
`metric_name{"my.label.name"="bar"}` 

`metric.name{"my.label.name"="bar"}` 

`{"metric.name", "my.label.name"="bar"}` 

Parquet Format

Service to convert TSDB blocks to parquet format, with experimental querier support



Documentation

[Getting Started](#)[Architecture](#)[Blocks Storage](#)[Configuration](#)[Guides](#)[Running Cortex on Kubernetes](#)[Getting started with a gossip ring cluster](#)[Alertmanager configuration](#)[Authentication and Authorisation](#)[Capacity Planning](#)[Config for horizontally scaling the Ruler](#)[Config for sending HA Pairs data to Cortex](#)[Encryption at Rest](#)[Ingesters rolling updates](#)[Ingesters scaling up and down](#)[Documentation](#) / [Guides](#) / [Parquet Mode](#)

Parquet Mode

Overview

Parquet mode in Cortex provides an experimental feature that converts TSDB blocks to Parquet format for improved query performance and storage efficiency on older data. This feature is particularly beneficial for long-term storage scenarios where data is accessed less frequently but needs to be queried efficiently.

The parquet mode consists of two main components:

- **Parquet Converter:** Converts TSDB blocks to Parquet format
- **Parquet Queryable:** Enables querying of Parquet files with fallback to TSDB blocks

Why Parquet Mode?

Traditional TSDB format and Store Gateway architecture face significant challenges when dealing with long-term data storage on object storage:

TSDB Format Limitations

- **Random Read Intensive:** TSDB index relies heavily on random reads, where each read becomes a separate request to object storage
- **Overfetching:** To reduce object storage requests, data that are close together are merged in a single request, leading to higher bandwidth usage and overfetching
- **High Cardinality Bottlenecks:** Index postings can become a major bottleneck for high cardinality data

Store Gateway Operational Challenges

- **Resource Intensive:** Requires significant local disk space for index headers and high memory usage

[Edit this page](#)[Create documentation issue](#)[Create project issue](#)[Overview](#)[Why Parquet Mode?](#)[TSDB Format Limitations](#)[Store Gateway Operational Challenges](#)[Parquet Advantages](#)[Architecture](#)[Configuration](#)[Enabling Parquet Converter](#)[Parquet Converter Configuration](#)[Per-Tenant Parquet Settings](#)[Enabling Parquet Queryable](#)[Query Limits for Parquet](#)[Cache Configuration](#)[Block Conversion Logic](#)[Querying Behavior](#)[Monitoring](#)[Parquet Converter Metrics](#)[Parquet Queryable Metrics](#)[Best Practices](#)[Deployment Recommendations](#)[Performance Tuning](#)[Fallback Configuration](#)[Limitations](#)[Migration Considerations](#)

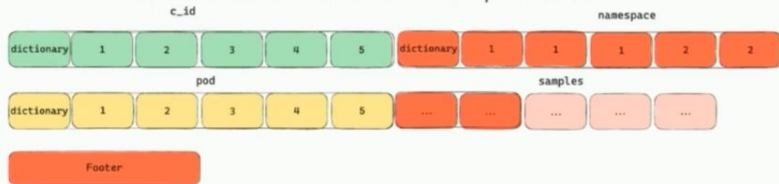
Data-access patterns

sum(kube_pod_info{namespace="monitoring"})

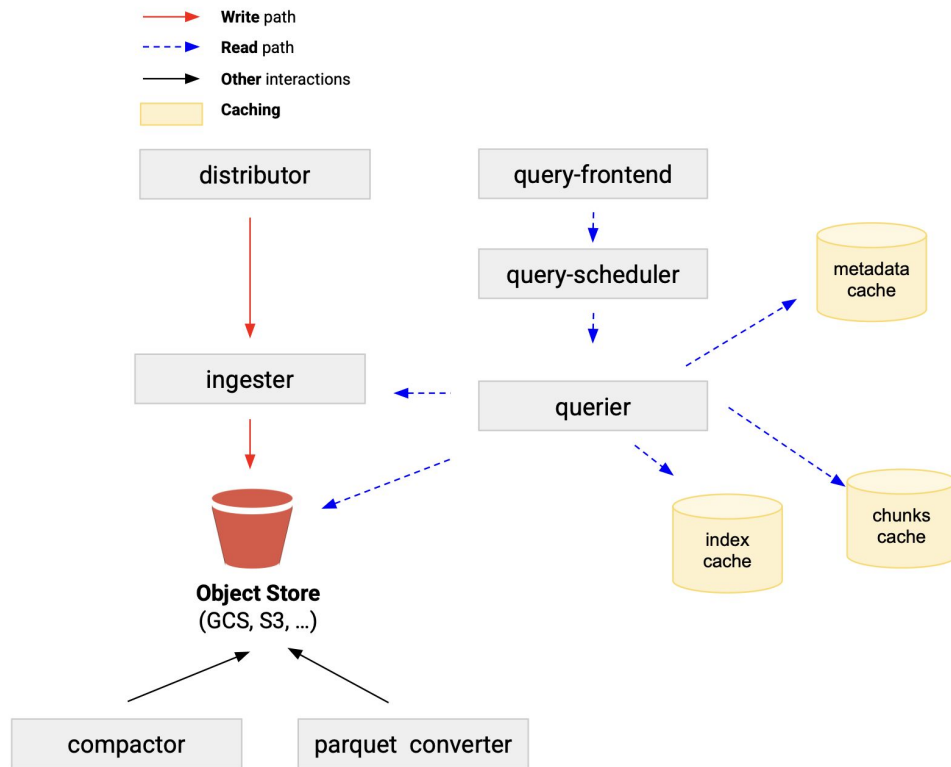
c_id	namespace	pod	samples
1	1	1	...
2	1	2	...
3	1	3	...
4	2	4	...
5	2	5	...

1: monitoring 1: thanos
2: kube-system 2: prometheus
3: statsd 3: statsd
4: core-dns 4: core-dns
5: kubelet 5: kubelet

Column-oriented access - 3 IO operations




Parquet mode for Querier (v1.20)



Parquet mode for Store-Gateway (v1.21)

Parquet Store gateway #7046

 Merged yeya24 merged 17 commits into [cortexproject:master](#) from [SungJin1212:parquet-gateway-poc](#)  on Nov 6, 2025

 Conversation 15  Commits 17  Checks 18  Files changed 18



SungJin1212 commented on Oct 10, 2025 · edited 

Member 

This PR introduces a new **parquet** mode for the Store Gateway. In this initial version, the mode operates statelessly and does not perform operations such as block syncing. The primary improvement is to enhance cache utilization by leveraging the Store Gateway's block affinity.

Which issue(s) this PR fixes:

Fixes [#6940](#)

Checklist

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



What does it bring?

- Re-use store gateway block affinity to retrieve parquet files
- Improves cache utilization over in memory querier caching



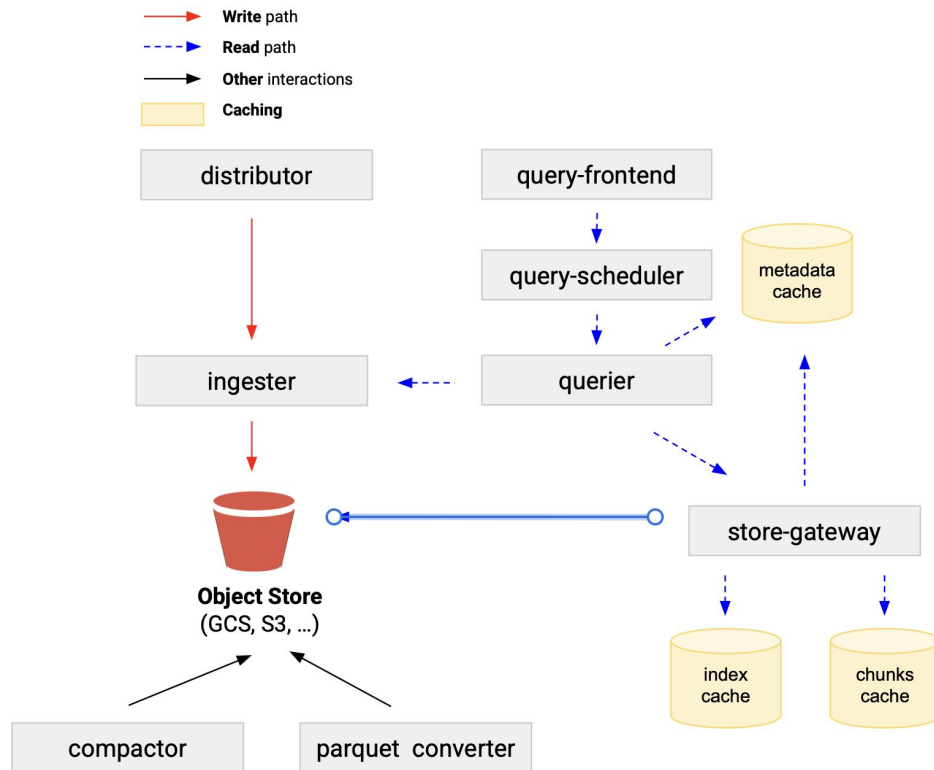
KubeCon



CloudNativeCon

Europe 2026

Parquet mode for Store-Gateway (v1.21)



What is in the v1.21 release

This release contains 164 contributions from 29 contributors. We also have 12 new contributors. Thank you all for the contributions!

Release	v1.18.0	v1.19.0	v1.20.0	v1.21.0
Date	2024-08-19	2025-01-22	2025-10-31	2026-03-09
Changes	11	7	4	5
Features	11	12	18	10
Enhancements	23	41	59	68
Bugfixes	8	8	25	25
Total	53	68	106	108

What is in the v1.21 release

Production readiness

- Ruler API
- Alertmanager API/sharding
- Tenant federation
- FIFO/Redis cache
- Instance limits
- Memcached DNS-based service discovery

What is in the v1.21 release

HATracker: Add experimental support for `memberlist` and `multi` as a KV store backend. #7284 (Feature)

[Documentation](#) / [Guides](#) / [Config for sending HA Pairs data to Cortex](#)

Config for sending HA Pairs data to Cortex

Context [↔](#)

You can have more than a single Prometheus monitoring and ingesting the same metrics for redundancy. Cortex already does replication for redundancy, and it doesn't make sense to ingest the same data twice. So in Cortex, we made sure we can dedupe the data we receive from HA Pairs of Prometheus. We do this via the following:


What is in the v1.21 release



Are my metrics queried at all???







What is in the v1.21 release

Now you can have metrics for number of series queried. Useful to understand query pattern and resource usage.

Add metrics for number of series queried #7173 

 Merged [yeya24](#) merged 4 commits into [cortexproject:master](#) from [yeya24:active-queried-series](#)  on Dec 23, 2025

Conversation 1 Commits 4 Checks 27 Files changed 28 +2,989 -10 

 **yeya24** commented on Dec 23, 2025 · edited  Member 




What this PR does:


This PR introduces a new feature to keep track of number of series queried in Ingestor per tenant and expose it as metrics. Number of series queried can be tracked via configurable time windows so that we can keep track of number of series queried for the past 2h, 12h or 24h. This can be useful to understand query pattern and resource usage.


Here are some key points of this implementation:


- [HyperLogLog](#) or [HLL](#) is used to keep track of the cardinality of queried series. This is a probabilistic data structure to estimate cardinality with minimum memory usage.
- Each tenant will maintain a circular buffer of HLLs. Each HLL keeps track of the number of series queried for a given time window. If the time window size is configured to be 30m, to calculate the usage for the past 2h, we just need to merge HLLs from the past 4 time windows and estimate the final cardinality.
- Users can configure different time windows to keep track of the metric. For example, 2h and 24h. With 30m time window, the circular buffer will have 48 HLLs in total. When calculating 2h metric, 4 HLLs are merged. When calculating 24h metric, 48 HLLs are merged. The actual implementation utilized caching as only the latest HLL is changing and previous HLLs' merged results are cached.
- The hash of queried series is used to be added to HLLs to estimate cardinality. Sampling can be enabled via `ingester.active-queried-series-metrics-sample-rate` to reduce resource usage as queries go to ingesters are likely to be repetitive
- Updating HLL with series hash is implemented using an async producer consumer pattern using channels. This avoids adding additional latency to the Ingestor query path.


Which issue(s) this PR fixes:
Fixes [#7151](#)


Reviewers 
 **alanprot** 

Assignees 
No one—[assign yourself](#)

Labels 
[component/ingester](#) [lgtm](#) [size/XXL](#)
[type/feature](#)

Projects 
None yet

Milestone 
No milestone


Development 
Successfully merging this pull request may close these issues.
[Expose metrics for number of active series ...](#)


Notifications [Customize](#)


What is in the v1.21 release

Configurable OTLP metric suffixes via `-distributor.otlp.add-metric-suffixes`

[FEATURE] Distributor: Make add-metric-suffixes configurable #7286 

 Merged [friedrich](#) merged 8 commits into `cortexproject:master` from `siddharthahuja1:siddharthahuja1/make-add-metric-suffixes-configurable`  2 weeks ago

 Conversation 2  Commits 8  Checks 33  Files changed 7 +247 -1 


 [siddharthahuja1](#) commented on Feb 20 · edited  Contributor 


What this PR does:
This PR introduces a configurable parameter (`add-metric-suffixes`) to control addition of suffixes to the metrics for name normalization.
Addresses the feature request: [#7282](#)


Which issue(s) this PR fixes:
Fixes #

Checklist

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




Reviewers 
 [friedrich](#) 

Assignees 
No one—[assign yourself](#)

Labels 
`component/distributor` `lgtm` `size/L`
`type/feature`

Projects 
None yet

Milestone 
No milestone

Overrides API

The User Overrides API provides a RESTful interface for managing tenant-specific limit overrides at runtime without requiring manual edits to the runtime configuration file or service restarts.

Add Overrides API component and rename old overrides to overrides-configs

#6975

Edit

<> Code

Merged

friedrichg merged 33 commits into cortexproject:master from bogdan-st:overrides_api last week

Conversation 51

Commits 33

Checks 33

Files changed 15

+2,257 -58



bogdan-st commented on Aug 15, 2025

Contributor

Adds initial draft for the User Overrides API, described in <https://cortexmetrics.io/docs/proposals/overrides-api/>

Checklist

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



Reviewers

friedrichg

+2 more reviewers

dsabsay

araiu

Assignees

No one—[assign yourself](#)

Labels

Road to CNCF Graduation

Road to Graduation Add status update Insights Workflows 4

View + New view

Filter by keyword or by field View

- Todo 1** This item hasn't been started
 - cortex #6905 Complete prep doc on graduation
- In Progress 1** This is actively being worked on
 - cortex #6683 Third party security review
- Done 3** This has been completed
 - cortex #6684 Roadmap change process
 - cortex #6685 Make sure release process is documented
 - cortex #6904 Confirm cortex adopters



Cortex security audit

Technical & Financial proposal

OSTIF

Get Involved



KubeCon



CloudNativeCon

Europe 2026

Cortex Community

About the community

SIG Meetings: every 4 weeks on Thursdays at 1700 UTC

Slack: <https://cloud-native.slack.com/messages/cortex/>

Issues: <https://github.com/cortexproject/cortex/issues>

Good first issues

Special topics

Enhancement Proposal Process

CNCF Code of Conduct

Q&A with core maintainers

Parquet?

Architecture & Scaling?

Multi-Tenancy?

Remote Write?

Community & Governance?

Operations?



KubeCon



CloudNativeCon

Europe 2026





KubeCon



CloudNativeCon

Europe 2026





KubeCon



CloudNativeCon

Europe 2026





KubeCon



CloudNativeCon

Europe 2026



Improved OTLP/HTTP Support

OTLP (OpenTelemetry Protocol) allows applications instrumented with OpenTelemetry SDKs to send traces, metrics, and logs directly to backend platforms or via the OpenTelemetry Collector. This vendor-neutral protocol enables direct-to-backend ingestion or intermediate collection, reducing dependence on specific agents.

Improvements:

- Supports Delta Temporality
- Flag to enable to types and units
- Metadata Ingestion

Connecting Your Laptop to Present

If you plan to use speaker notes, please follow the steps below to ensure your laptop is set up correctly. This will allow you to view your notes while the audience sees only your slides.

Before Connecting

- Close all unnecessary apps & tabs
- Turn off notifications
- Disable Night Mode / TrueTone
- Don't enter full-screen presentation mode yet

At the podium

- Plug in the HDMI cable
- If needed, use your USB-C to HDMI adapter
- On Mac, click “Allow” if prompted

USE “EXTENDED DISPLAY” MODE (NOT MIRRORED)

Using MAC

- Go to System Settings > Displays
- Click the “Arrangement” tab
- Make sure “Mirror Displays” is unchecked

Using PC

- Press \square Win + P, then select “Extend”
- Or: Right-click on the Desktop > Display Settings
- Under Multiple Displays, select “Extend these displays”

Connecting Your Laptop to Present



KubeCon



CloudNativeCon

Europe 2026

When Using PowerPoint

- Start slideshow: ⌘Cmd + Enter (Mac) or F5 (PC)
- Or: Click Slide Show > Play from Start
- If notes show on the wrong screen, click “Swap Displays”
- <https://www.youtube.com/watch?v=gQ3D4m-5pww>

Google Slides

- Use Google Chrome as your browser
- Click the dropdown next to ‘Slideshow’ > select Presentation Display Options
- Check “Presenter View” and “Full Screen”
- If prompted, select "Allow"
- Choose the external display (may have a strange name like PanasonicXYZ)
- Click Start Slideshow
- <https://www.youtube.com/watch?v=-GT7WCvPcys>