Chronosphere User Guide

Document Level Classification

200

Introduction

Chronosphere is a new metrics storage system used by Epic. It is intended to be a replacement for New Relic Metrics use cases for APM, Infrastructure, Prometheus, and Custom metrics. You can think of it as a very large, scalable Prometheus server. It ingests metrics from our OTEL collectors and provides a PromQL interface to those metrics. We expose this as Prometheus datasources in Grafana, so you never need to interact with Chronosphere directly to make queries.

Chronosphere has a few key advantages over New Relic, and we believe you will have a better overall experience using it for metrics:

- **Significantly** faster query times. In an apples:apples test for the same data for the same app, Chronosphere was 4x to 12x faster than New Relic for loading individual panels in Grafana.
- Accuracy. Chronosphere is just a Prometheus implementation (M3) and implements the Prometheus spec. PromQL queries in New Relic are translated to NRQL, which can lead to inaccurate results, in particular for rate queries. Direct NRQL queries can have strange artifacts like random "spikes" or missing data points.
- Cost savings. Chronosphere has a lower overall cost and can help reduce costs further by helping reduce large metric footprints. The Data Platform team reduced 80% of its observability costs (around $550k/year) through a combination of migrating to Chronosphere and using Chronosphere's governance and aggregation tools to reduce its footprint.

As of June 2024, by default, the OTEL pipeline in **all substrate clusters** will dual write to New Relic and Chronosphere. Any metrics you send to the cluster-local OTLP collector, or export with Prometheus, will be delivered to both vendors

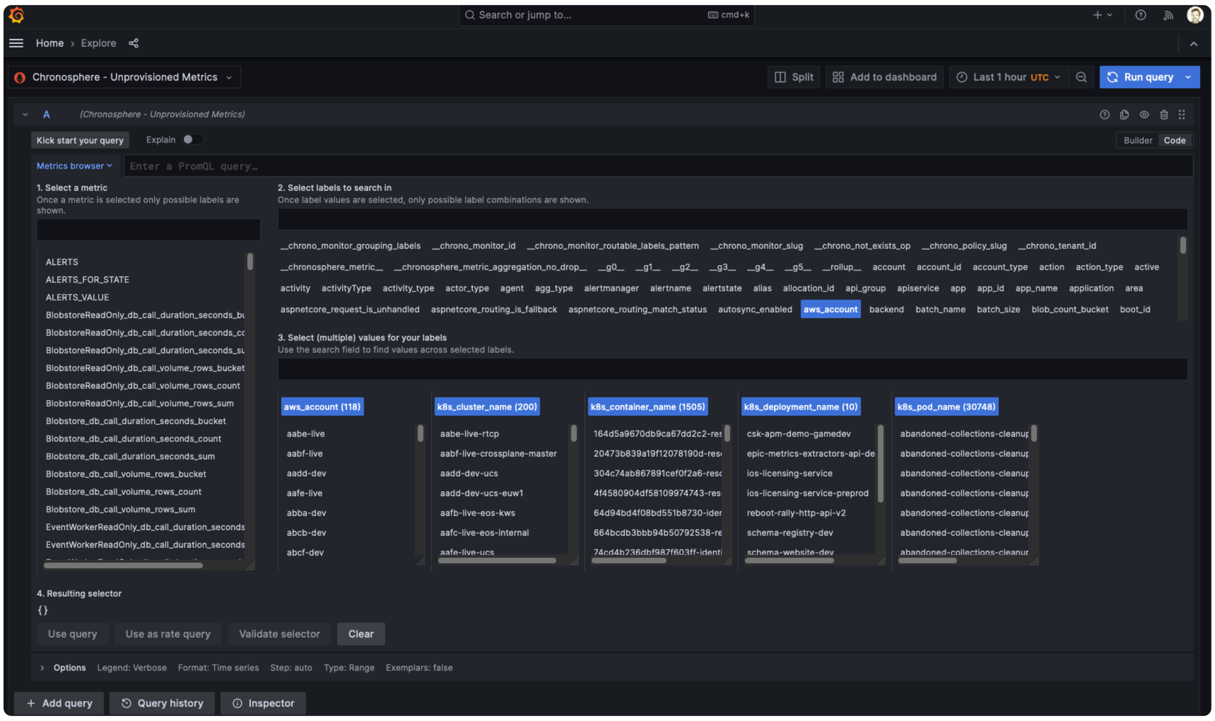
Finding Your Metrics

Chronosphere metrics are available in Grafana via Prometheus datasources starting with "Chronosphere". These are available in both

Grafana Core and Grafana EOS. There are two categories of Chronosphere datasources:

- "Chronosphere Tenant" - this is datasource is for metrics with specific capacity reserved
- "Chronosphere - Unprovisioned Metrics" - this is where everything else goes. If our account capacity limits are reached, these metrics are at risk of being dropped. Before using any metrics for production use cases, contact #ct-obs-support-ext to have your metrics moved to a tenant.

If you're unsure of where to get started, begin with the Grafana Explorer UI. This will show you all metrics currently being sent to a specific Chronosphere tenant.



PromQL

If you are new to PromQL, have a look at the Prometheus Query Basics documentation. There is a bit of a learning curve if you are coming from NRQL, but Prometheus is fast which makes it fairly easy to iterate and learn. Here are some practical examples of queries that work in NRQL, translated to PromQL.

Computing Request Rate by Route

NRQL

```
SELECT rate(count(%.server%duration), 1 second) as `req/s` FROM Metric W
```

PromQL

```
(sum(rate(http_server_duration_milliseconds_count{service_name="$servic
```

P99 Performance by Route

NRQL

```
SELECT percentile(%.server%duration, 0.99) FROM Metric WHERE service.na
```

PromQL

```
histogram_quantile(0.99, sum by (le, http_method, http_route) (rate(htt
```

Tenants and Quotas

Chronosphere capacity is provisioned up-front. This is by design to discourage run-away growth in metrics that lead to high observability costs. Our account capacity is provisioned into multiple metrics pools. The relationship between metric sources and the pool they go into is something we call a tenant. Tenants have a metrics datapoint-per-second (DPPS) quota, and all metrics going to a the same tenant are grouped under the same datasource. To illustrate with a practical example, let's look at how the Data org tenants work:

data-org-internal

- sources: fdbe-dev, cfef-live, edec-dev, fbfb-live (these are the AWS accounts emitting the metrics)
- quota: 40k DPPS
- grafana instance: grafana core
- datasource name: Chronosphere Tenant - data-org-internal

data-org-eos

- sources: eded-dev, dadb-live
- quota: 20k DPPS
- grafana instance: grafana eos
- datasource name: Chronosphere Tenant - data-org-eos

By default, your metrics go to a default pool that is de-prioritized and at risk at being dropped if account limits are reached. Before using Chronosphere metrics for production, reach out to #ct-obs-support-ext to have a tenant configured. The observability team will work with you to determine which tenant your metrics should go into, or whether a new one will be created. Typically new tenants will be created based on either organizational boundaries (Fortnite, Ecosec, Data, Mediatonic, etc) or when explicit separation is required (EOS for example). This mapping is done "on the fly" using dynamic labels and be reconfigured at any time, so it's possible to re-define what goes into a tenant in the future without losing any previous data. In other words, if we get it wrong, or there's a re-org in the future, we can change this later.

Quota Enforcement

Quotas are enforced when the account exceeds its DPPS capacity. As of June 2024, that capacity is 225,000 DPPS. Enforcement works like this:

- Once account capacity is reached, metrics in the default "unprovisioned" pool are dropped first. This is why you should have your metrics put in a tenant before relying on them for production.
- If capacity is still exceeded, tenants that are in violation of their quota are affected next
    - EKS dev cluster metrics are dropped first

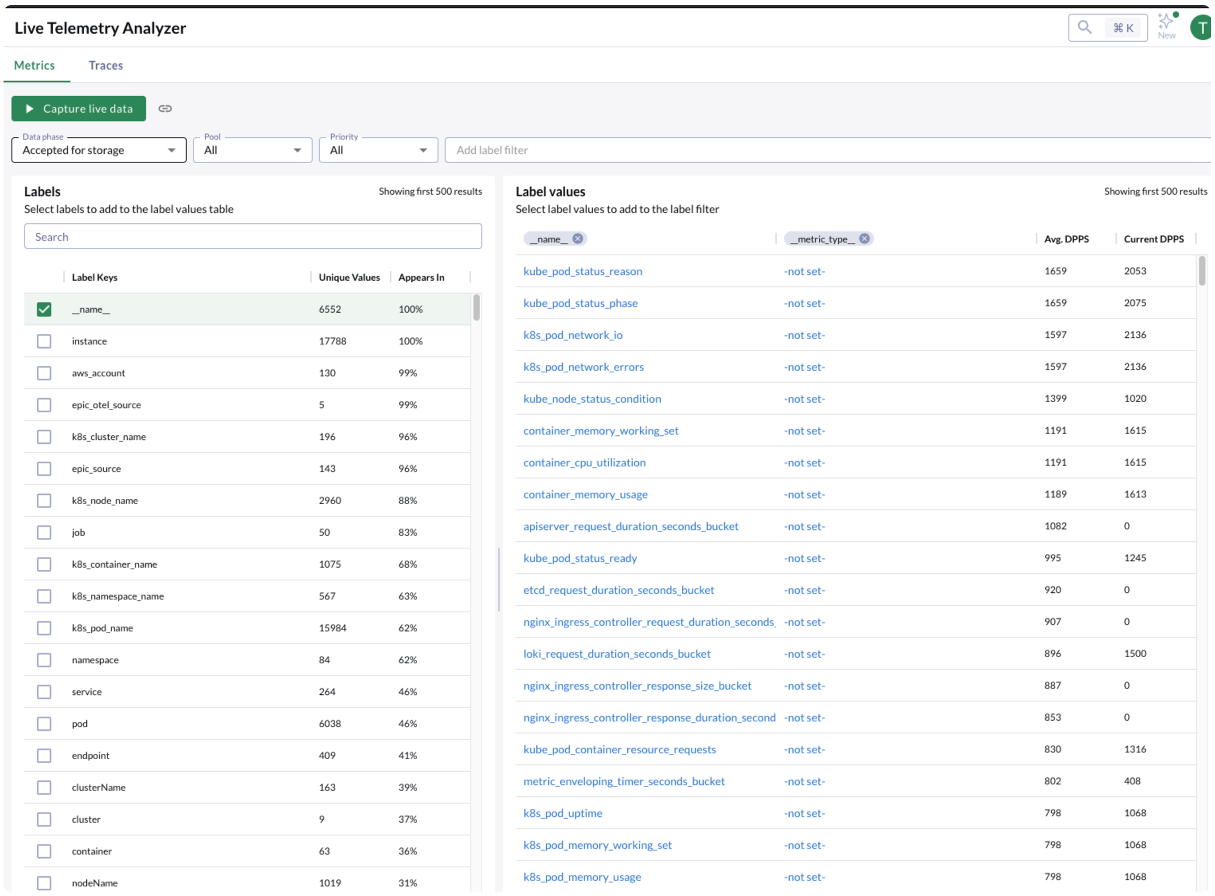
        - all other metrics not associated with an EKS live cluster are dropped after that
        - if still in violation of the quota at this point, live cluster metrics will be dropped
    - Tenants not in violation of their quota will not be affected by any enforcement mechanisms.

Advanced Features

Chronosphere has some advanced functionality for teams that want to get their metric costs under control. These are features only accessible in the Chronosphere UI. Since we think the typical user experience should start and end in Grafana, we don't grant this access by default, but if you need to use them please reach out. These are available as "power user" features.

Live Telemetry Analyzer

This allows us to peek at the "firehose" of metrics coming in and see large sources of writes, high cardinality metrics, etc. It's the first tool we use when asking "who or what is sending so much data?"



Data phase: Accepted for storage
Pool: All
Priority: All
Add label filter

Labels
Select labels to add to the label values table
Showing first 500 results

Search

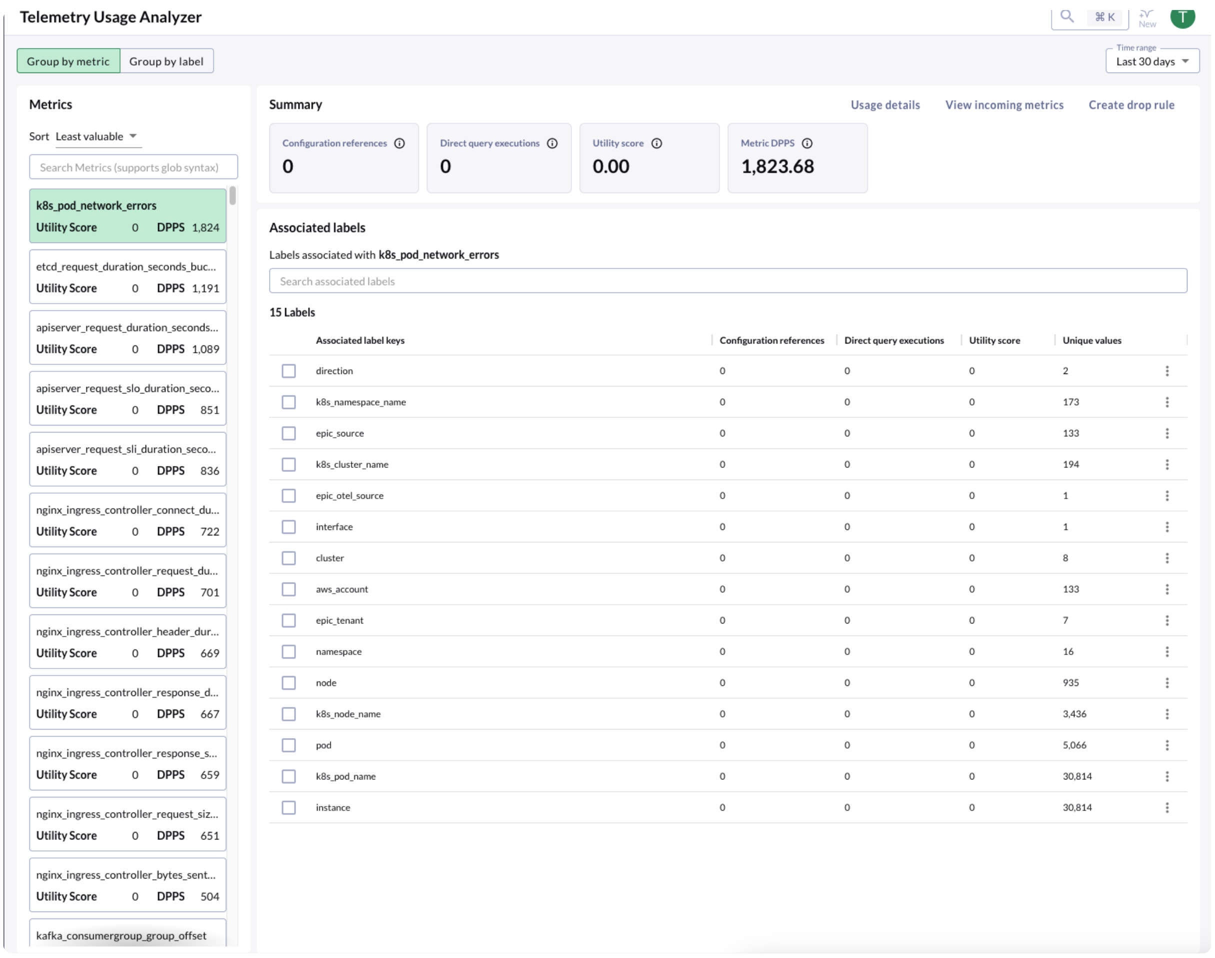
	Label Keys	Unique Values	Appears In
✓	__name__	6552	100%
☐	instance	17788	100%
☐	aws_account	130	99%
☐	epic_otel_source	5	99%
☐	k8s_cluster_name	196	96%
☐	epic_source	143	96%
☐	k8s_node_name	2960	88%
☐	job	50	83%
☐	k8s_container_name	1075	68%
☐	k8s_namespace_name	567	63%
☐	k8s_pod_name	15984	62%
☐	namespace	84	62%
☐	service	264	46%
☐	pod	6038	46%
☐	endpoint	409	41%
☐	clusterName	163	39%
☐	cluster	9	37%
☐	container	63	36%
☐	nodeName	1019	31%

Label values
Select label values to add to the label filter
Showing first 500 results

__name__	__metric_type__	Avg. DPPS	Current DPPS
kube_pod_status_reason	-not set-	1659	2053
kube_pod_status_phase	-not set-	1659	2075
k8s_pod_network_io	-not set-	1597	2136
k8s_pod_network_errors	-not set-	1597	2136
kube_node_status_condition	-not set-	1399	1020
container_memory_working_set	-not set-	1191	1615
container_cpu_utilization	-not set-	1191	1615
container_memory_usage	-not set-	1189	1613
apiserver_request_duration_seconds_bucket	-not set-	1082	0
kube_pod_status_ready	-not set-	995	1245
etcd_request_duration_seconds_bucket	-not set-	920	0
nginx_ingress_controller_request_duration_seconds_	-not set-	907	0
loki_request_duration_seconds_bucket	-not set-	896	1500
nginx_ingress_controller_response_size_bucket	-not set-	887	0
nginx_ingress_controller_response_duration_second	-not set-	853	0
kube_pod_container_resource_requests	-not set-	830	1316
metric_enveloping_timer_seconds_bucket	-not set-	802	408
k8s_pod_uptime	-not set-	798	1068
k8s_pod_memory_working_set	-not set-	798	1068
k8s_pod_memory_usage	-not set-	798	1068

Telemetry Usage Analyzer

This tool helps us see which metrics are being used. It takes a lot of guesswork out of "do I even need this?"



Telemetry Usage Analyzer

Group by metric | Group by label

Time range: Last 30 days

Metrics

Sort: Least valuable

Search Metrics (supports glob syntax)

Metric	Utility Score	DPPS
k8s_pod_network_errors	0	1,824
etcd_request_duration_seconds_buc...	0	1,191
apiserver_request_duration_seconds...	0	1,089
apiserver_request_slo_duration_seco...	0	851
apiserver_request_sli_duration_seco...	0	836
nginx_ingress_controller_connect_du...	0	722
nginx_ingress_controller_request_du...	0	701
nginx_ingress_controller_header_dur...	0	669
nginx_ingress_controller_response_d...	0	667
nginx_ingress_controller_response_s...	0	659
nginx_ingress_controller_request_siz...	0	651
nginx_ingress_controller_bytes_sent...	0	504
kafka_consumergroup_group_offset		

Summary

Usage details | View incoming metrics | Create drop rule

Configuration references	Direct query executions	Utility score	Metric DPPS
0	0	0.00	1,823.68

Associated labels

Labels associated with **k8s_pod_network_errors**

Search associated labels

15 Labels

Associated label keys	Configuration references	Direct query executions	Utility score	Unique values
direction	0	0	0	2
k8s_namespace_name	0	0	0	173
epic_source	0	0	0	133
k8s_cluster_name	0	0	0	194
epic_otel_source	0	0	0	1
interface	0	0	0	1
cluster	0	0	0	8
aws_account	0	0	0	133
epic_tenant	0	0	0	7
namespace	0	0	0	16
node	0	0	0	935
k8s_node_name	0	0	0	3,436
pod	0	0	0	5,066
k8s_pod_name	0	0	0	30,814
instance	0	0	0	30,814

Metrics Aggregations

Metrics aggregations can be used for a number of purposes but a major one is to reduce high cardinality metrics and high DPPS by dropping unneeded dimensions. This is very useful, for example, when you have no control over the exported metrics and it's sending high cardinality labels that you don't want. To illustrate the concept, imagine you have some metrics coming in like this:

```
http_request_count{service="web", k8s_pod_name="foo", http_status_code=
http_request_count{service="web", k8s_pod_name="bar", http_status_code=
http_request_count{service="web", k8s_pod_name="xyz", http_status_code=
http_request_count{service="web", k8s_pod_name="foo", http_status_code=
http_request_count{service="web", k8s_pod_name="bar", http_status_code=
http_request_count{service="web", k8s_pod_name="xyz", http_status_code=
```

You have a lot of pods emitting these metrics but you don't care about this metric at the pod level. Rather than store one metric per pod, you can use an aggregation rule to sum up these metrics and drop the pod name. This would produce a metric like:

```
http_request_count{service="web", http_status_code="200"} 4
http_request_count{service="web", http_status_code="204"} 12
```

Processing metrics like this is not free but it is substantially cheaper (about 1/3rd the price) of ingesting the raw metrics. Aggregation rules are managed in this terraform repository. If you would like help setting aggregation rules, please reach out to #ct-obs-support-ext.

Drop Rules

Drop rules let you drop noisy metrics in a pinch at the ingestion layer at Chronosphere. While it's more efficient to simply not emit the metric in the first place, the ability to drop metrics without changing any code or re-deploying is useful especially in an emergency. Drop rules are managed in this terraform repository. If you would like help setting drop rules, please reach out to #ct-obs-support-ext.

FAQ

How do quotas work during large events?

Quotas are not enforced if the account-wide limit has not been reached. A head of a large-scale event, the observability team will request additional capacity from Chronosphere, which will effectively raise any quota you currently have. **tl;dr** you do not need to worry about micromanaging your quotas ahead of an event.

How do I see my quota limit and utilization?

See the Chronosphere Quota Utilization dashboard here. For EOS grafana, the dashboard is here. We do not yet have limit values in these dashboards, so to inspect your limit, see this terraform file for now.

How do I request a quota increase?

Just contact #ct-obs-support-ext. We can work with you to raise your quota.

My quota is $X DPPS, what is this in real dollar terms?

This will likely change in the future as we get better bulk pricing, but currently each 10000 DPPS allocated to your tenant quota runs approximately $15000/year.

Glossary

Tenant - a logical grouping of related metric sources that roll up under the same quota and are accessed through the same grafana datasource. These are usually aligned with a particular business unit or organization.

DPPS - datapoints per second, or the number of metrics being ingested per second

---

Page Information:
Page ID: 81068700
Space: Cloud Developer Platform
Downloaded: 2025-07-12 04:08:44