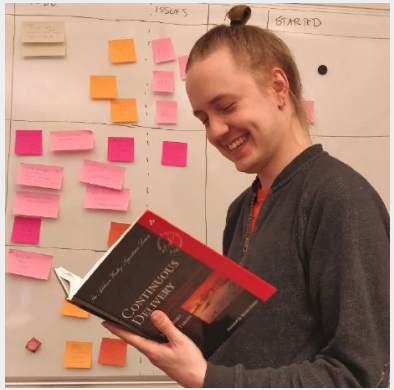

Unity®
Metrics, Tracing and bugs in production
2023

Nice to meet you!



Jami Kousa

Senior Software Engineer
Unity Technologies

Let’s get started

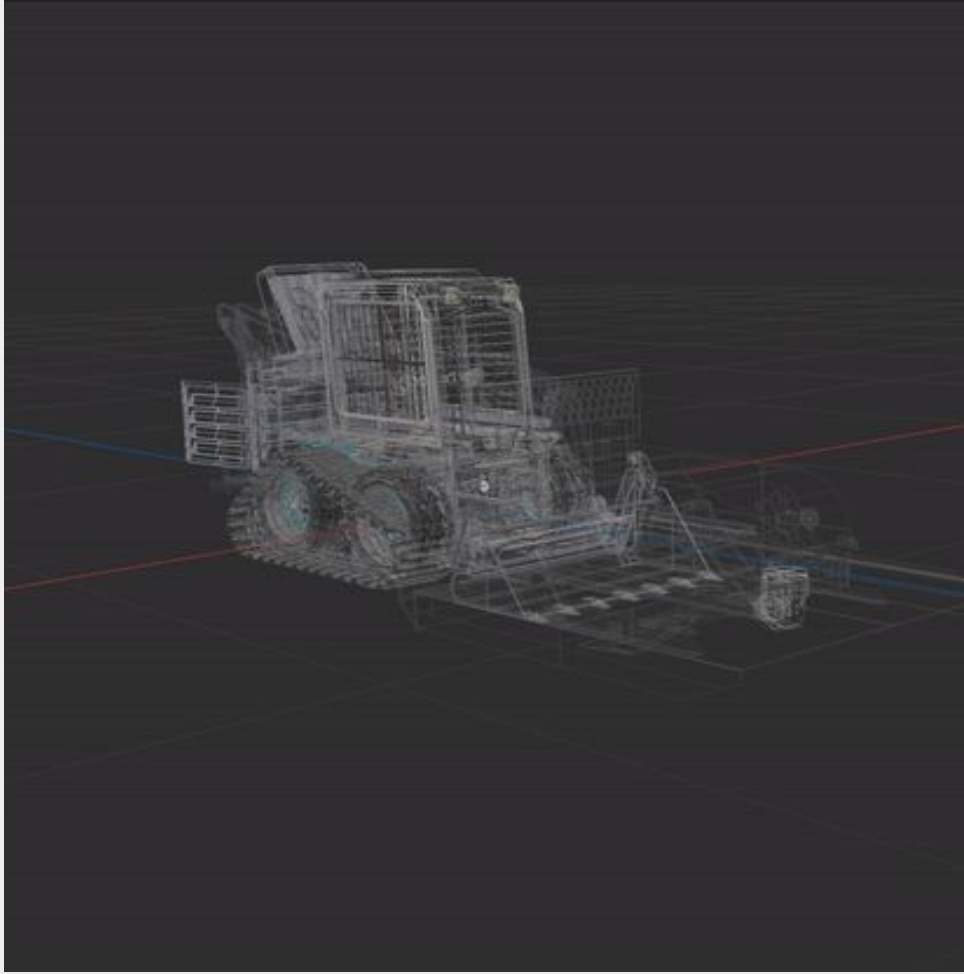
→ What is Unity?
→ What is Unity Cloud?

→ What do I do?
→ How do I know if my software works?


Unity®

Unity and real-time 3D

- → Video Games
  - – >39 000 games on Steam
  - – >80% of games on mobile
- → Industry
  - – Digital Twins
    - ● Cars



2K

Software Engineers and Developers

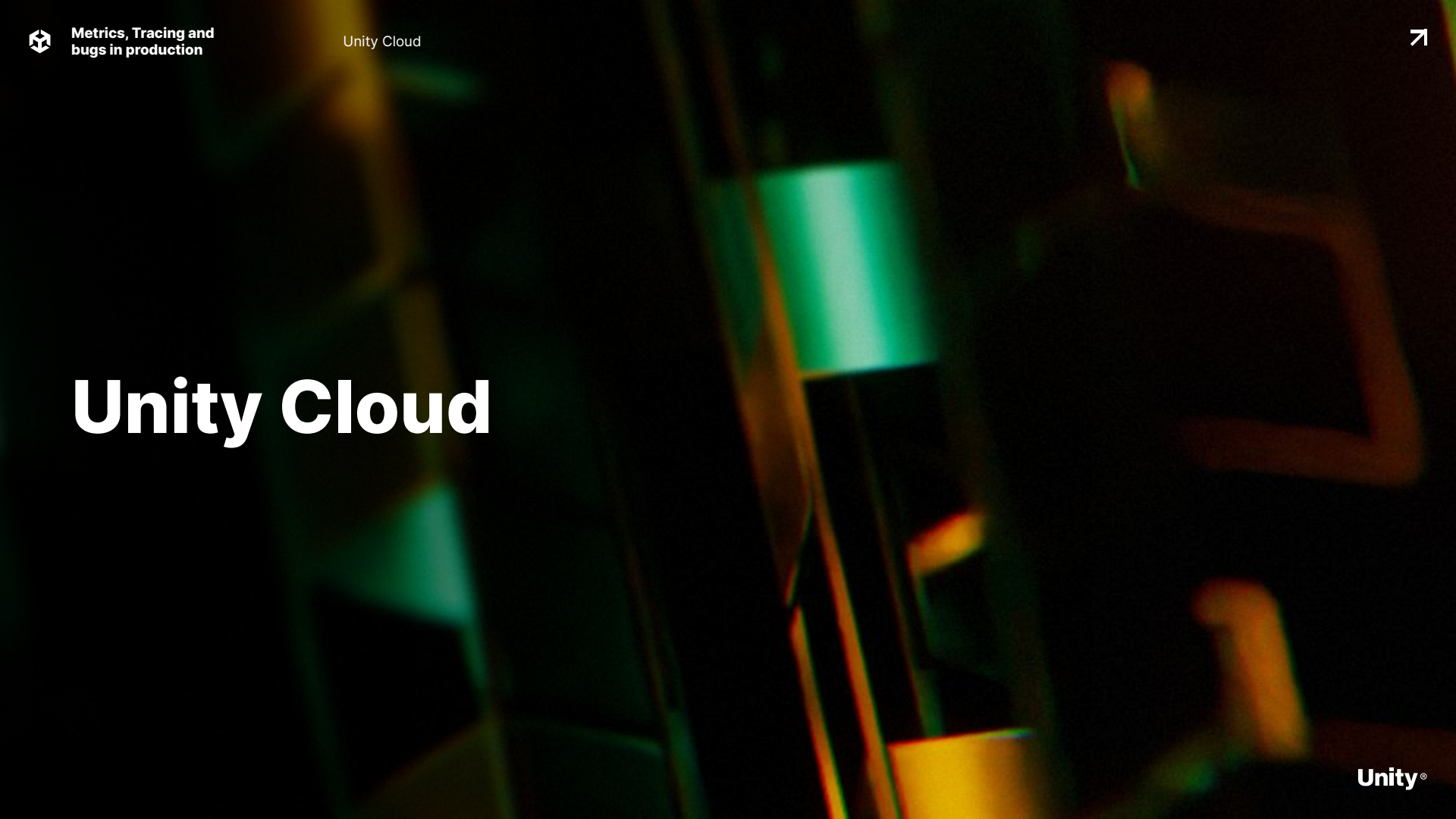
23

Office locations around the globe

2B

Monthly Users

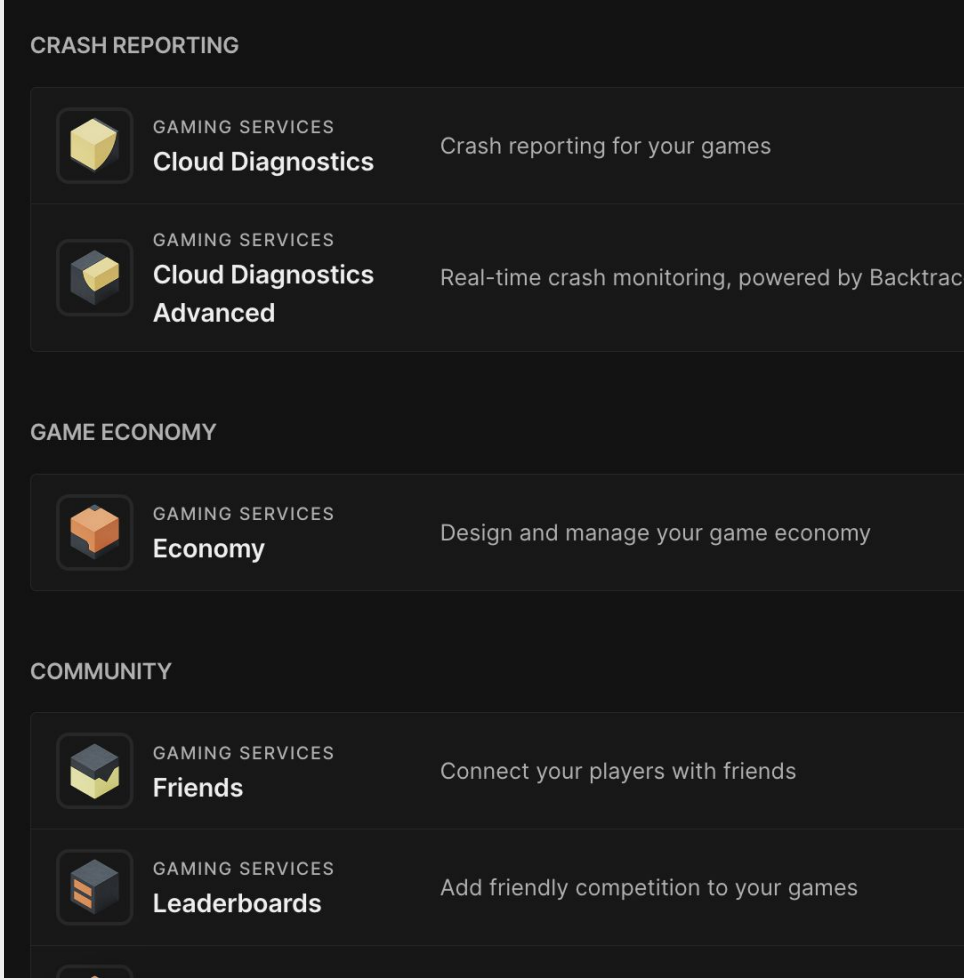
Unity Cloud



Unity Cloud

→ Most things that are available here:
- https://cloud.unity.com

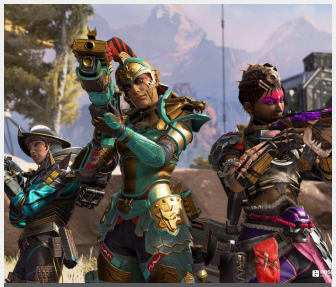
→ Services such as:
- Unity Gaming Services
- Unity DevOps
- Asset Manager



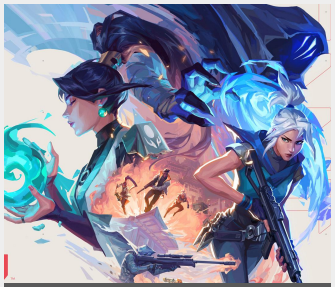
Games relying on our services

Among Us

Marvel Snap



Apex Legends



Valorant

Fortnite

Unity Analytics

Dashboards and visualization tools to help make the right decisions.

Cloud Content Delivery

Build and release game updates with content delivery via the cloud.

Multiplay

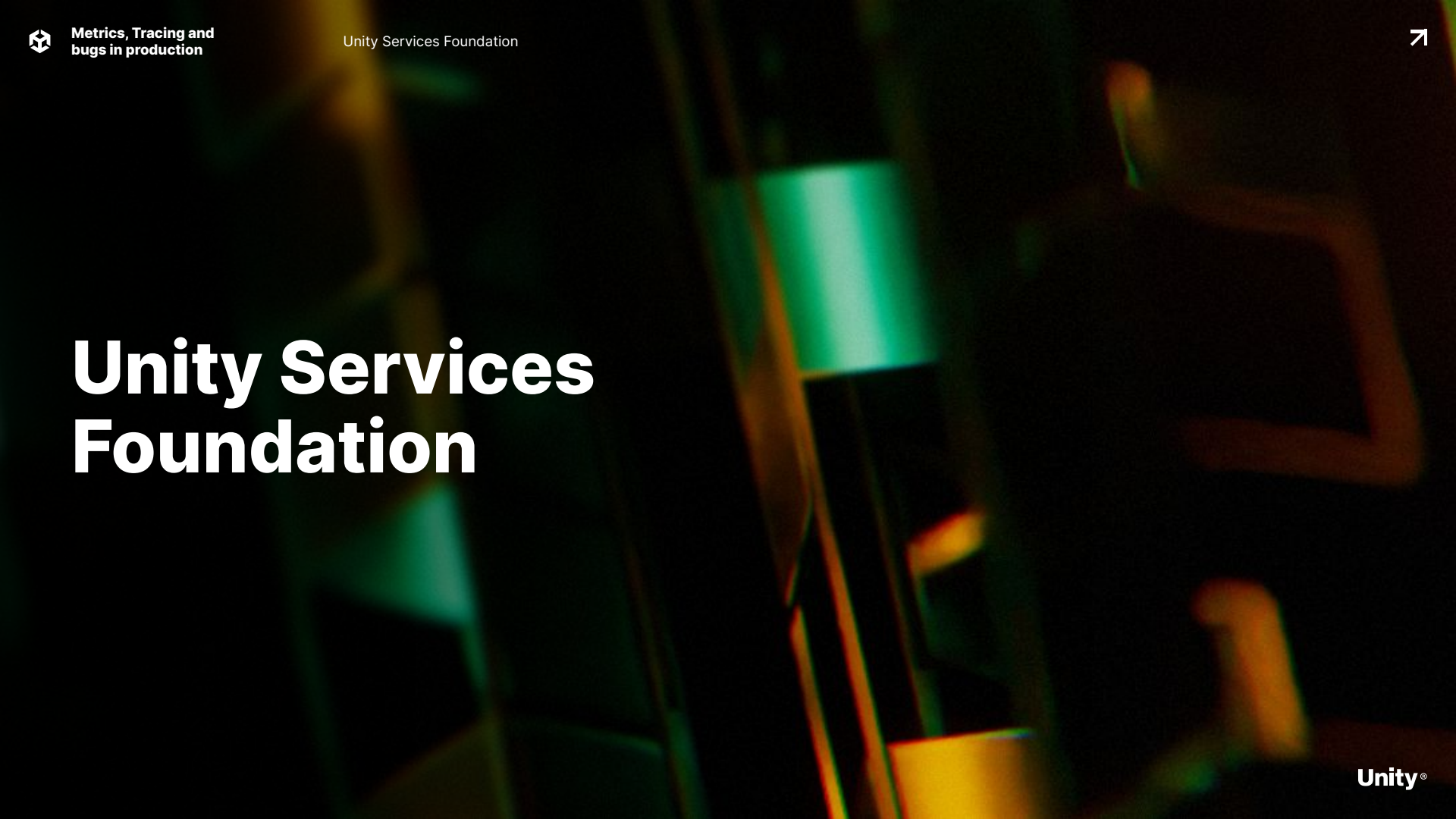
Orchestration solution for hosting and scaling your game.

Vivox

Service for both text and voice chat, with all the necessary controls you would expect.

...and countless more

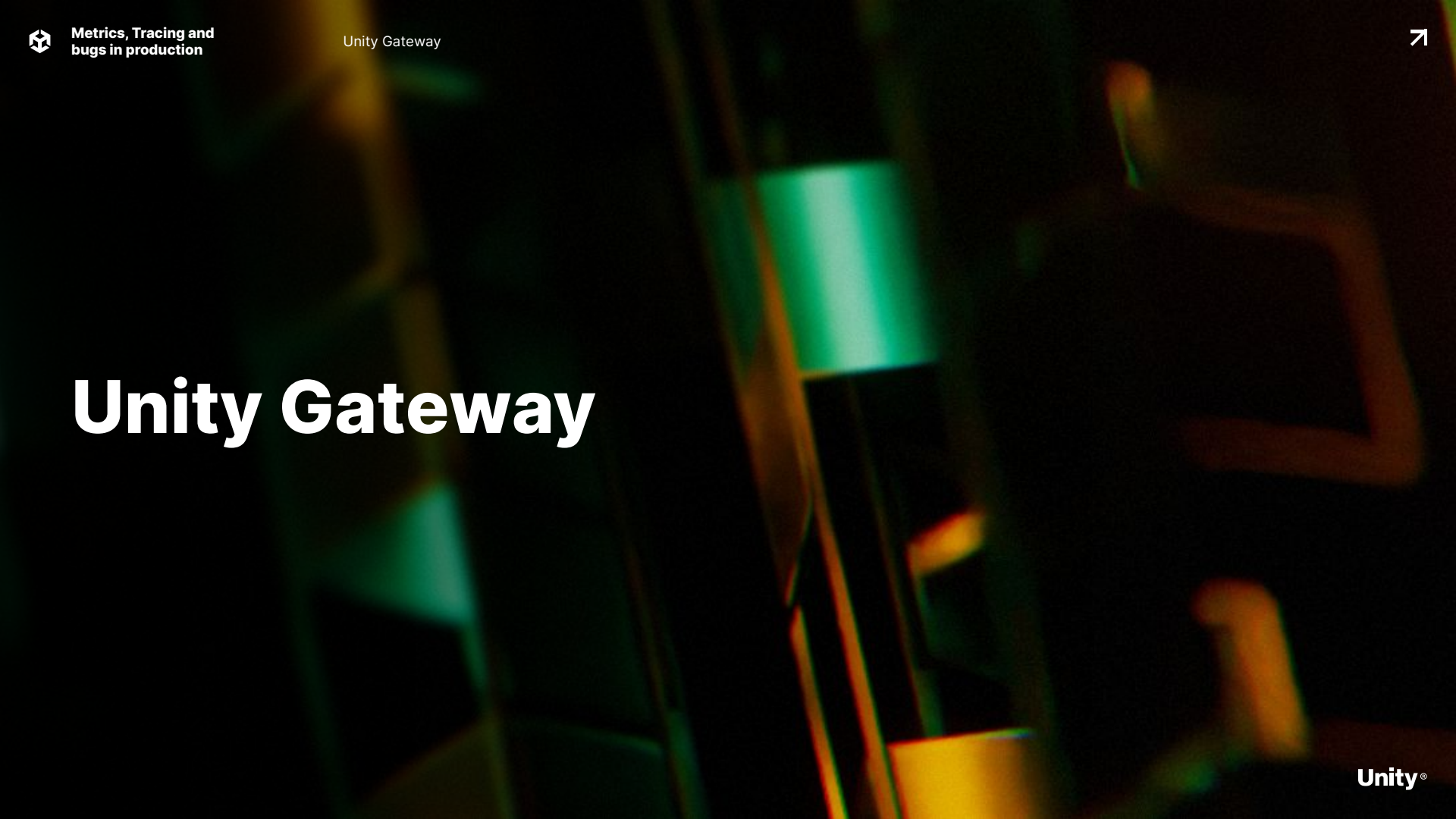
Unity Services Foundation



Next week..

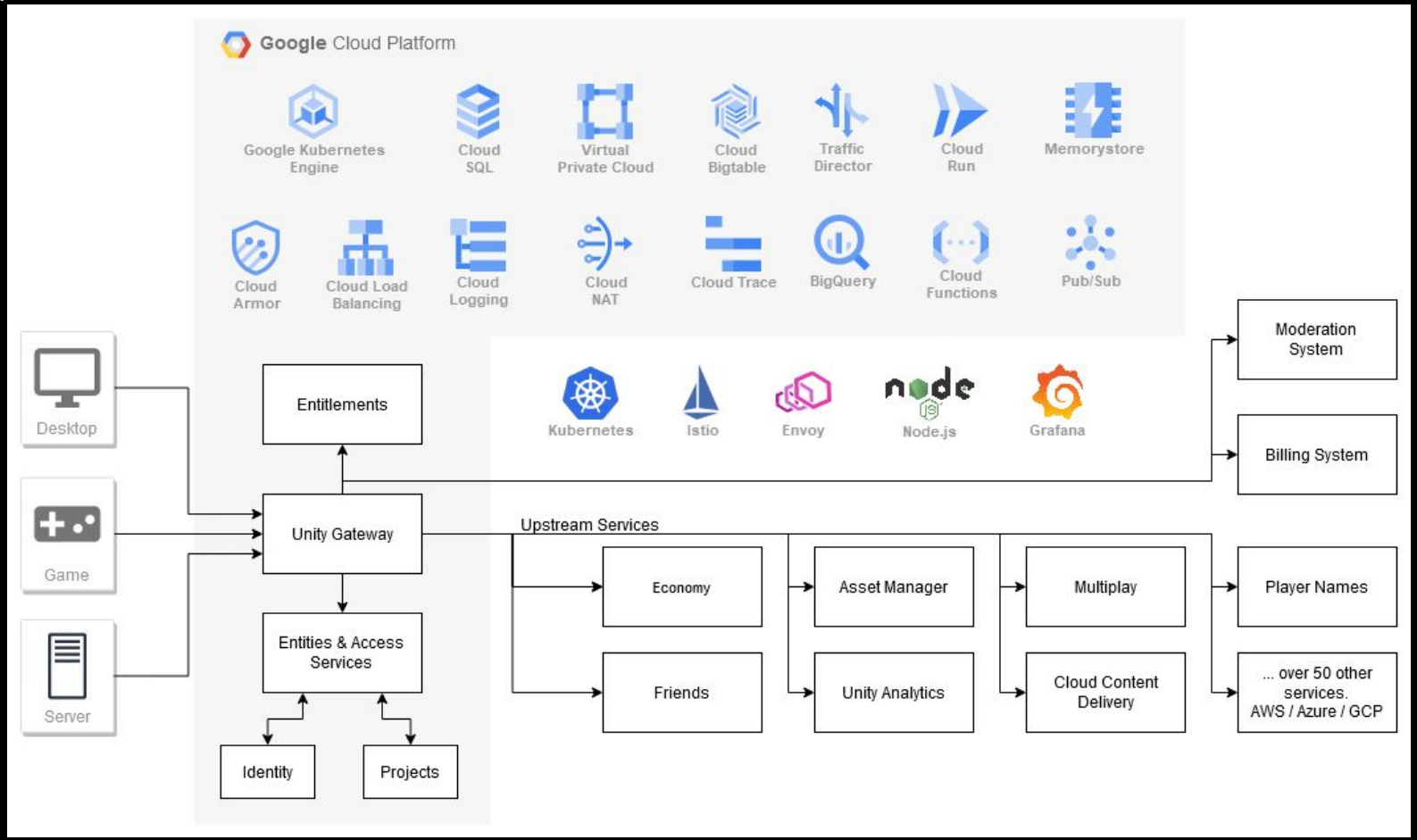
Kasper Hirvikoski

Director of Engineering
Unity Technologies



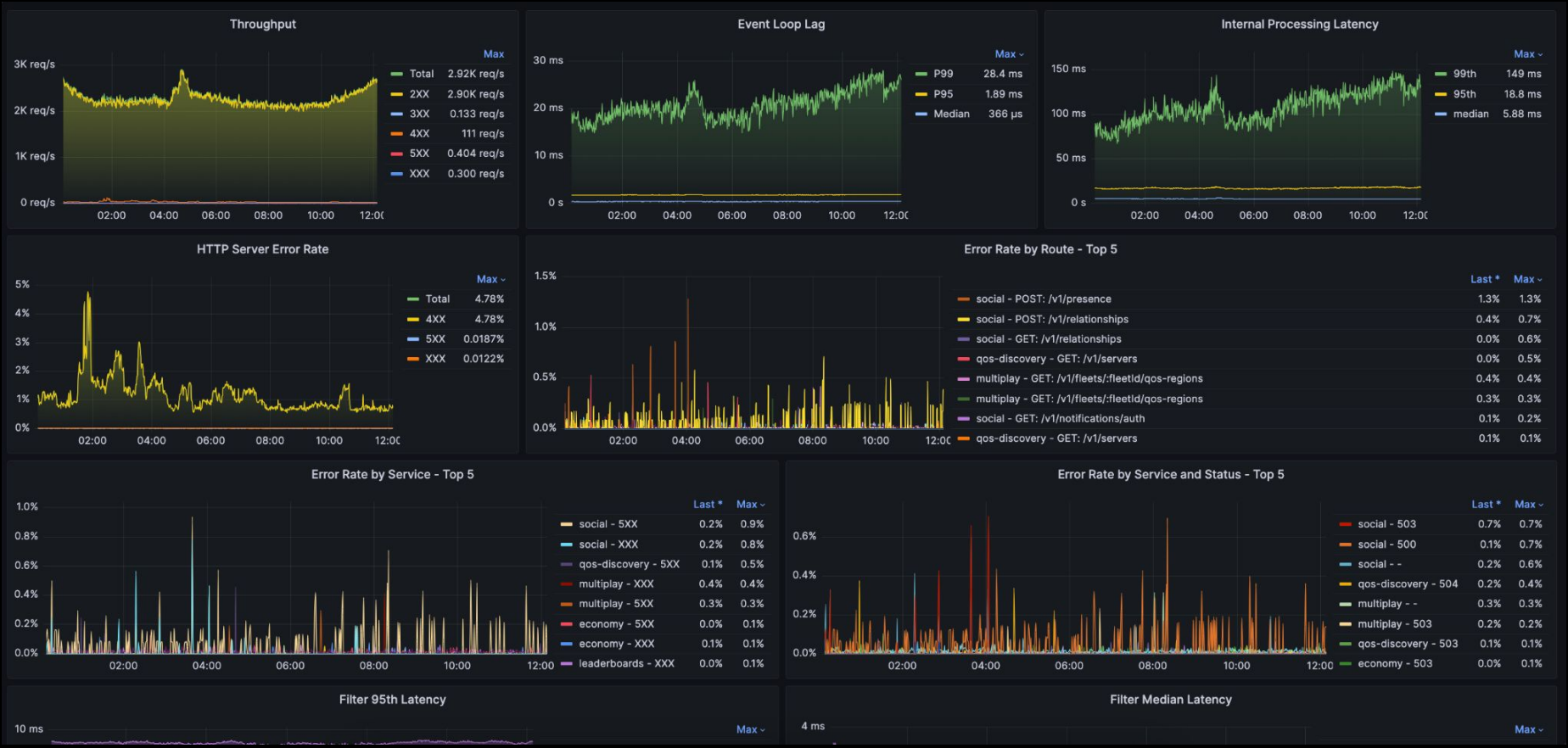
Unity Gateway

Unity Gateway

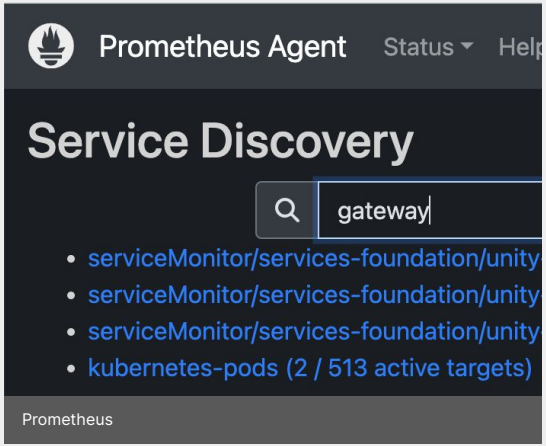


Maintaining

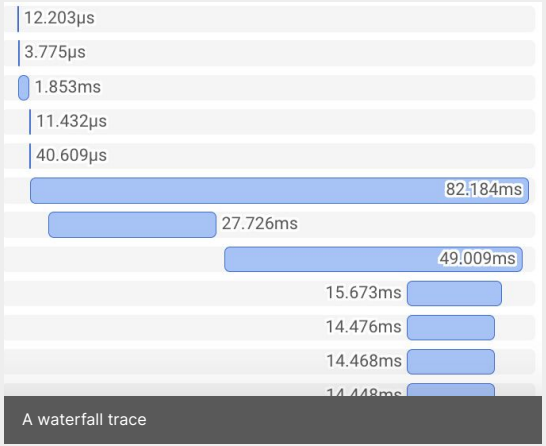
- → Documentation
  - – Game Developer Docs
  - – Unity Developer Docs
  - – Gateway Developer Docs
- → How do I maintain my sanity and a stress-free work environment?
  - – Tests
    - ● End-to-end tests
    - ● Integration tests
    - ● Unit tests
  - – Observability


Throughput
3K req/s
2K req/s
1K req/s
0 req/s
02:00 04:00 06:00 08:00 10:00 12:0
Max
Total 2.92K req/s
2XX 2.90K req/s
3XX 0.133 req/s
4XX 111 req/s
5XX 0.404 req/s
XXX 0.300 req/s
Event Loop Lag
30 ms
20 ms
10 ms
0 s
02:00 04:00 06:00 08:00 10:00 12:0
Max
P99 28.4 ms
P95 1.89 ms
Median 366 µs
Internal Processing Latency
150 ms
100 ms
50 ms
0 s
02:00 04:00 06:00 08:00 10:00 12:0
Max
99th 149 ms
95th 18.8 ms
median 5.88 ms
HTTP Server Error Rate
5%
4%
3%
2%
1%
0%
02:00 04:00 06:00 08:00 10:00 12:0
Max
Total 4.78%
4XX 4.78%
5XX 0.0187%
XXX 0.0122%
Error Rate by Route - Top 5
1.5%
1.0%
0.5%
0.0%
02:00 04:00 06:00 08:00 10:00 12:0
Last * Max
social - POST: /v1/presence 1.3% 1.3%
social - POST: /v1/relationships 0.4% 0.7%
social - GET: /v1/relationships 0.0% 0.6%
qos-discovery - GET: /v1/servers 0.0% 0.5%
multiplay - GET: /v1/fleets/:fleetId/qos-regions 0.4% 0.4%
multiplay - GET: /v1/fleets/:fleetId/qos-regions 0.3% 0.3%
social - GET: /v1/notifications/auth 0.1% 0.2%
qos-discovery - GET: /v1/servers 0.1% 0.1%
Error Rate by Service - Top 5
1.0%
0.8%
0.6%
0.4%
0.2%
0.0%
02:00 04:00 06:00 08:00 10:00 12:00
Last * Max
social - 5XX 0.2% 0.9%
social - XXX 0.2% 0.8%
qos-discovery - 5XX 0.1% 0.5%
multiplay - XXX 0.4% 0.4%
multiplay - 5XX 0.3% 0.3%
economy - 5XX 0.0% 0.1%
economy - XXX 0.1% 0.1%
leaderboards - XXX 0.0% 0.1%
Error Rate by Service and Status - Top 5
0.6%
0.4%
0.2%
0.0%
02:00 04:00 06:00 08:00 10:00 12:00
Last * Max
social - 503 0.7% 0.7%
social - 500 0.1% 0.7%
social - - 0.2% 0.6%
qos-discovery - 504 0.2% 0.4%
multiplay - - 0.3% 0.3%
multiplay - 503 0.2% 0.2%
qos-discovery - 503 0.1% 0.1%
economy - 503 0.0% 0.1%
Filter 95th Latency
10 ms
Max
Filter Median Latency
4 ms
Max

Observability through telemetry



Prometheus



A waterfall trace

Jotta varmasti ymmärtäisimme, miten ohjelma toimii, lisätään koodiin muutama `console.log`

```
const App = () => {
  const [counter, setCounter] =
useState(0)
  console.log('rendering with
```

console.log

METRICS

Basically numbers. Prometheus collects and stores its metrics as time series data, i.e. metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels.

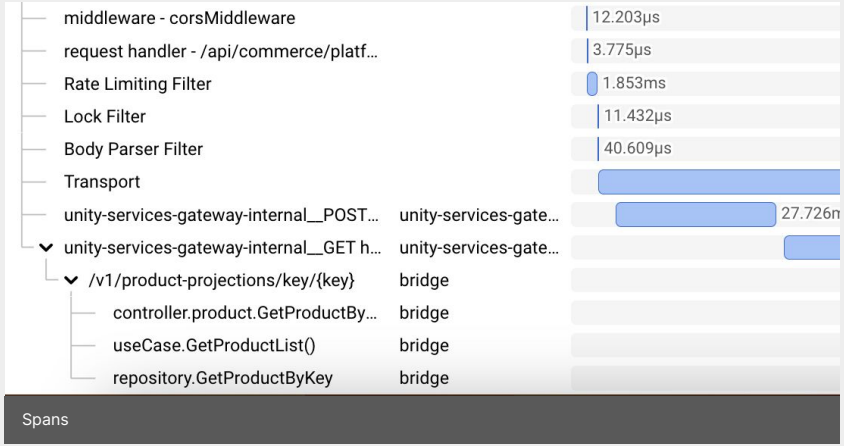
TRACES

A spans records a piece of information about how the application handles an action, such as a request. Spans are the construction blocks of a trace. OpenTelemetry is used to collect trace data from the application.

LOGS

Text. Structured text is usually more useful than unstructured.

Tracing



Spans

Grafana

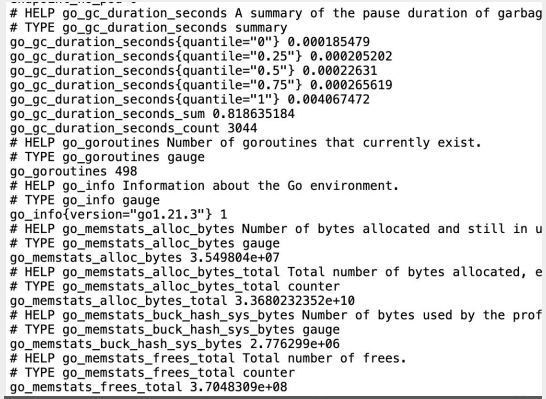
REQUEST TRACING

Each step of a request the Unity Gateway receives is tracked as a separate “Span” these spans are then collected into a complete trace of the request, telling the entire story of the request processing.

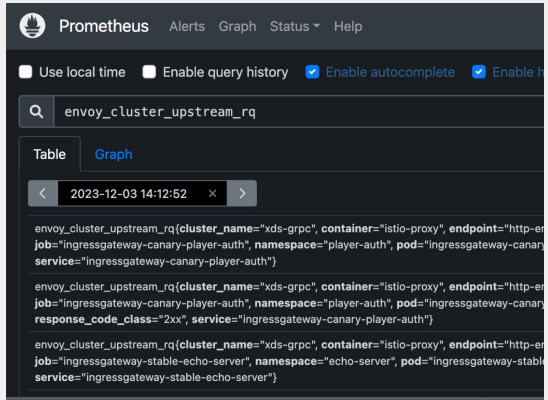
VISUALIZATION

We use Grafana to visualize the traces. An additional tool which offers us ways to query and retrieve traces. This way us humans can look at pretty pictures and try to come up with actions.

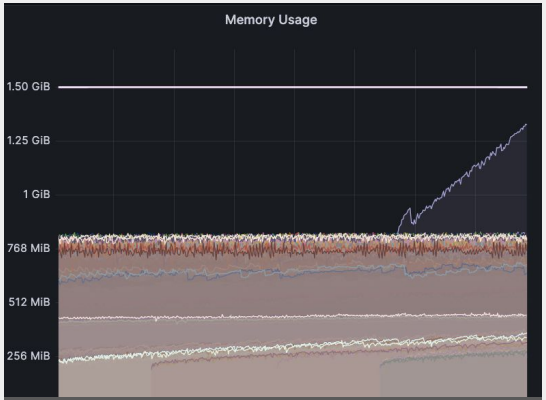
Metrics



A metrics endpoint



Prometheus dashboard



Grafana panel

WHY

One of the best part of metrics is being able to define alerts. Alerts then allow you to know if something is not going as expected.

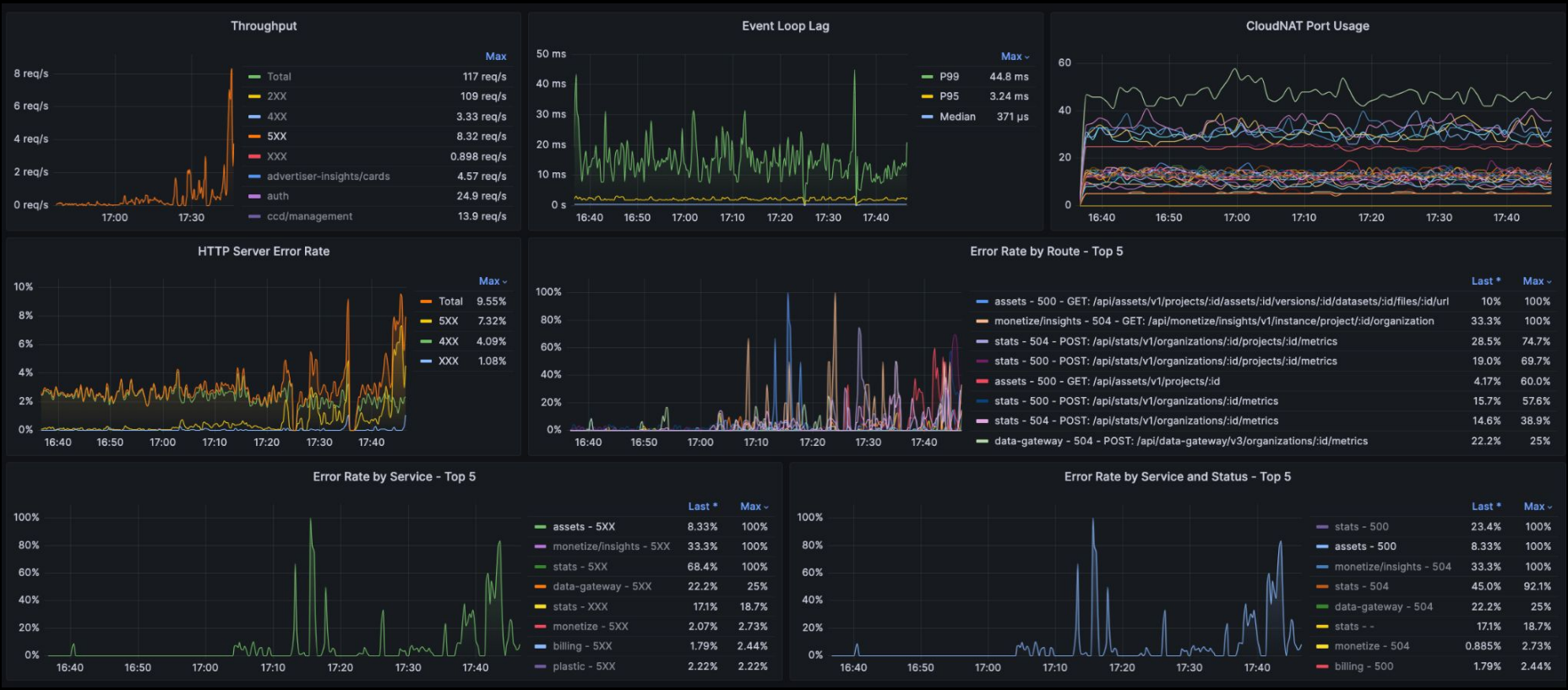
Other reason is to report those metrics to someone who cares.

HOW TO MEASURE

The modern tool for this is Prometheus. Prometheus is used to scrape metrics information from the application.

HOW TO DISPLAY

Grafana gives us humans visibility into the metrics and allows creation of the cool dashboards.


Throughput
Max
8 req/s
6 req/s
4 req/s
2 req/s
0 req/s
17:00
17:30
Total 117 req/s
2XX 109 req/s
4XX 3.33 req/s
5XX 8.32 req/s
XXX 0.898 req/s
advertiser-insights/cards 4.57 req/s
auth 24.9 req/s
ccd/management 13.9 req/s
Event Loop Lag
Max
50 ms
40 ms
30 ms
20 ms
10 ms
0 s
16:40
16:50
17:00
17:10
17:20
17:30
17:40
P99 44.8 ms
P95 3.24 ms
Median 371 µs
CloudNAT Port Usage
60
40
20
0
16:40
16:50
17:00
17:10
17:20
17:30
17:40
HTTP Server Error Rate
Max
10%
8%
6%
4%
2%
0%
16:40
16:50
17:00
17:10
17:20
17:30
17:40
Total 9.55%
5XX 7.32%
4XX 4.09%
XXX 1.08%
Error Rate by Route - Top 5
Last *
Max
100%
80%
60%
40%
20%
0%
16:40
16:50
17:00
17:10
17:20
17:30
17:40
assets - 500 - GET: /api/assets/v1/projects/:id/assets/:id/versions/:id/datasets/:id/files/:id/url 10% 100%
monetize/insights - 504 - GET: /api/monetize/insights/v1/instance/project/:id/organization 33.3% 100%
stats - 504 - POST: /api/stats/v1/organizations/:id/projects/:id/metrics 28.5% 74.7%
stats - 500 - POST: /api/stats/v1/organizations/:id/projects/:id/metrics 19.0% 69.7%
assets - 500 - GET: /api/assets/v1/projects/:id 4.17% 60.0%
stats - 500 - POST: /api/stats/v1/organizations/:id/metrics 15.7% 57.6%
stats - 504 - POST: /api/stats/v1/organizations/:id/metrics 14.6% 38.9%
data-gateway - 504 - POST: /api/data-gateway/v3/organizations/:id/metrics 22.2% 25%
Error Rate by Service - Top 5
Last *
Max
100%
80%
60%
40%
20%
0%
16:40
16:50
17:00
17:10
17:20
17:30
17:40
assets - 5XX 8.33% 100%
monetize/insights - 5XX 33.3% 100%
stats - 5XX 68.4% 100%
data-gateway - 5XX 22.2% 25%
stats - XXX 17.1% 18.7%
monetize - 5XX 2.07% 2.73%
billing - 5XX 1.79% 2.44%
plastic - 5XX 2.22% 2.22%
Error Rate by Service and Status - Top 5
Last *
Max
100%
80%
60%
40%
20%
0%
16:40
16:50
17:00
17:10
17:20
17:30
17:40
stats - 500 23.4% 100%
assets - 500 8.33% 100%
monetize/insights - 504 33.3% 100%
stats - 504 45.0% 92.1%
data-gateway - 504 22.2% 25%
stats - - 17.1% 18.7%
monetize - 504 0.885% 2.73%
billing - 500 1.79% 2.44%

When an alert triggers

Investigate

Gather the data from all sources, such as logs messages, traces and metrics to gain an understanding of what is happening.

Evaluate

Make the decision to start an **incident** if this warrants immediate action.

Alerts can be caused from multiple reasons. There might be a need to contact another team, or save everyone from a DDoS attack. Or maybe the alert was faulty.

Fix

Silence the alert in one way or another. Depending on the evaluation.

Incident management process

In our slack “/incident start” creates a new channel, notifies relevant teams and assigns you as an “incident commander”

Incidents always have a commander, and the commander is responsible for driving the incident toward resolution.

Commander duty can be re-assigned to someone who knows better.

After it’s been resolved, a postmortem document is created with things such as: the start time, the end time, what happened, what was the problem, how was it found and how to prevent if from happening again.

Incident levels

- P0 Critical Issue
  - Significant revenue or data loss or outage of a critical service impacting many teams.
- P1 Service Degradation
  - Revenue loss, critical service difficult to use, or a non-critical service outage.
- P2 Unintended Behavior
  - A product or a service that has lost some functionality. No revenue loss.
- P3 Known Issue
  - A product or service with unexpected but acceptable behavior.

Q&A

Unity®

Thank you

2023


Unity®