Cloud Computing

Course plan

Learning session 5

Pricing

Monitoring and Observability

Pricing

Sometimes you pay not for what you use, but for what you forgot to turn off

There is no Garbage Collection

Pricing

- Snowflake pricing
- Pay-as-you-go vs pre-paid vs spot instances
- Storage pricing
- Traffic pricing
- Cost analysis
- Calculator

Pricing

The pricing story could be entirely different for different resources in the same cloud

Pricing

Pay-as-you-go (on-demand) – pay according to consumption

Pricing

Consumption could be calculated:

- Memory/cpu/disk
- Seconds/minutes/hours
- Custom units

Pricing

Pre-paid (reserved) – pay in advance

Sometimes can reduce price by >50%

Pricing

Spot instance – propose your price for resource

Pricing

Spot instances

- Cloud unused capacity
- Best suitable for batch or non-critical workloads
- You cannot know when resources will be available and when cloud takes them away
- Sometimes can reduce prices by 80%

Pricing

In the cloud, network traffic could be expensive

- Try not to cross a single data-center boundary
- Network traffic between data-centers is (often) billed
- Internet traffic is (often) billed

Pricing

Storage pricing

Cross-zone availability or backups could be included into cloud-managed service

You will pay for it if you implement the same functionality using basic Compute/Storage/Network primitives

Pricing

Regularly use cost analysis tools and set alerts on crossing your budget

Pricing

To estimate spending use Calculator:

- Azure
- AWS
- GCP

Pricing

Demo

Pricing

Tips

- Shut-down unused resources (automation is your friend)
- Consider changing resource size (often it’s easy-peasy, but could be only one direction - up)
- Reserved and spot instances to the rescue
- Autoscaling
- Regularly review your architecture (new services and features may arise)
- Set budget and ALERTS

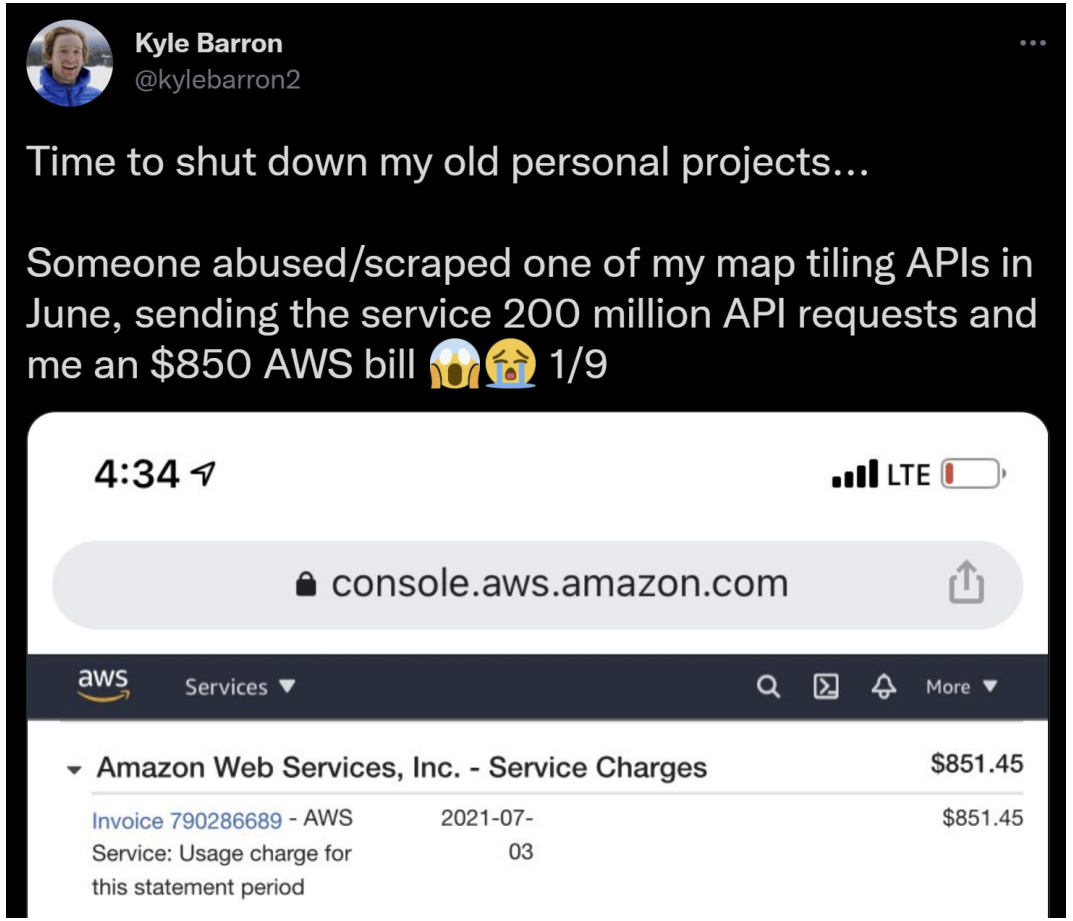
Pricing

All the rules apply even if you do …

Pricing

All the rules apply even if you do …

…a small pet-project



Pricing

All the rules apply even if you do …

…a big non-profit

[ua] translation

How I Got Pwned by My Cloud Costs

24 JANUARY 2022

I have been, and still remain, a massive proponent of "the cloud". I built Have I Been Pwned (HIBP) as a cloud-first service that took advantage of modern cloud paradigms such as Azure Table Storage to massively drive down costs at crazy levels of performance I never could have achieved before. I wrote many blog posts about doing big things for small dollars and did talks all over the world about the great success I'd had with these approaches. One such talk was How I Pwned My Cloud Costs so it seems apt that today, I write about the exact opposite: how my cloud costs pwned me.

It all started with my monthly Azure bill for December which was *way* over what it would normally be. It only took a moment to find the problem:

Purchases Charge Start Date - Charge End Date	Charges/ Credits (AUD)	GST	Tax Amount (AUD)	Total (including Tax) (AUD)
Networking 01/12/2021-31/12/2021	4,557.13	10.00%	455.72	5,012.85

Pricing

p.s.

Often engineering team salary is >>> cloud bill

Self-managed service price also includes engineering time

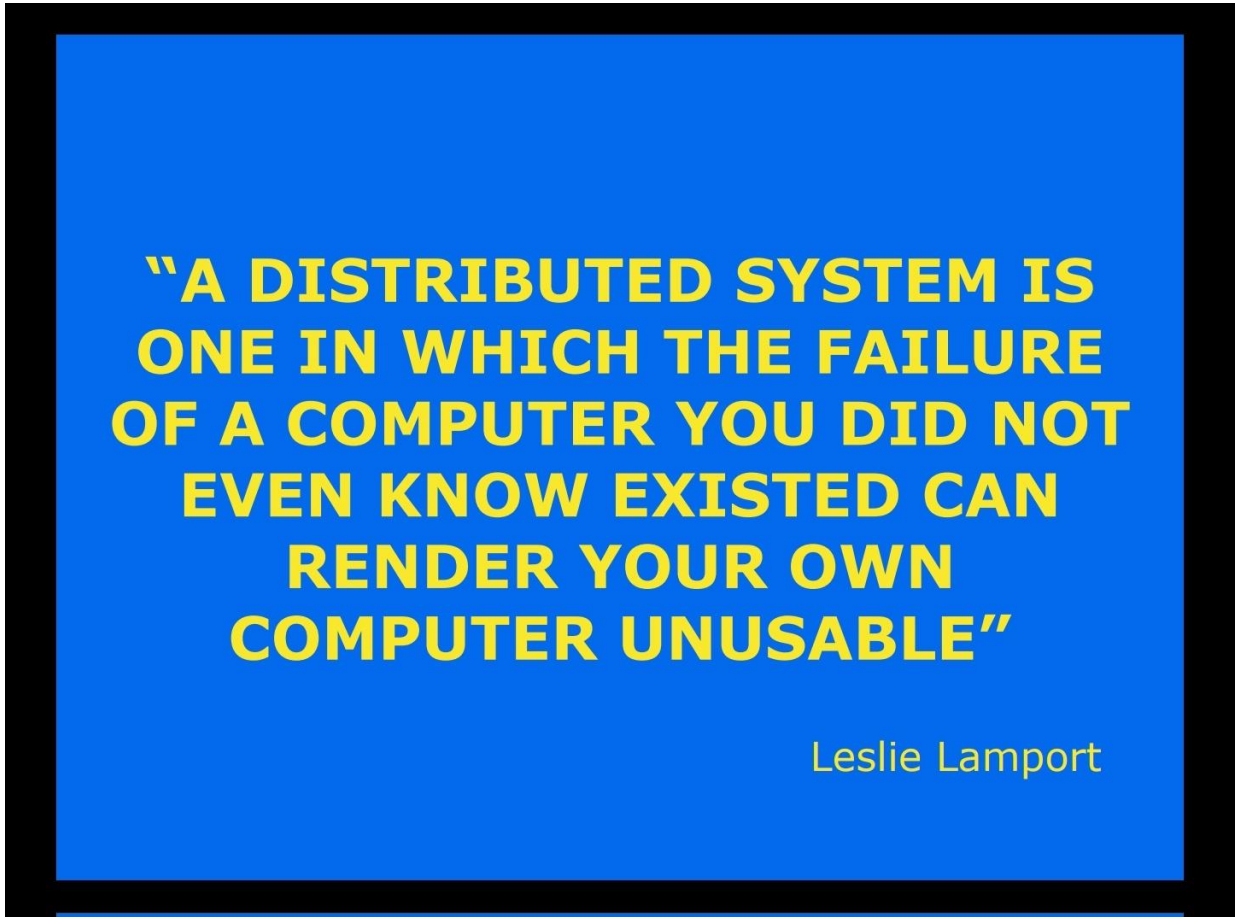
Course plan

Learning session 5

Pricing

Monitoring and Observability

Monitoring and Observability



Monitoring and Observability

- What is Monitoring and Observability
- Logs
- Metrics (and SLI/SLO/SLA)
- Traces
- Audit
- Alerts
- Available solutions

Monitoring and Observability

Any system eventually starts misbehaving

Monitoring and Observability

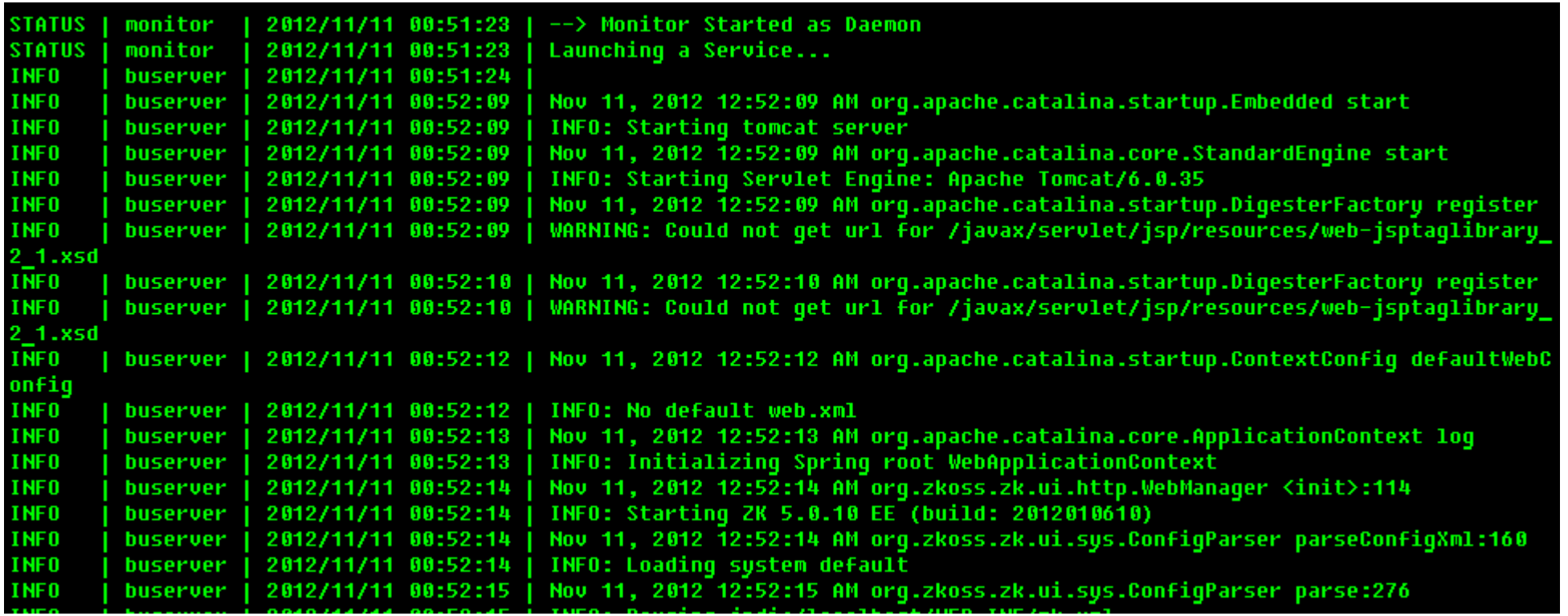
When system misbehaves on *my-machine* – I can debug it

Monitoring and Observability

When system misbehaves in-production – I can …

Monitoring and Observability

Logging – process for collecting, storing, retrieving, processing and visualizing log records/events.



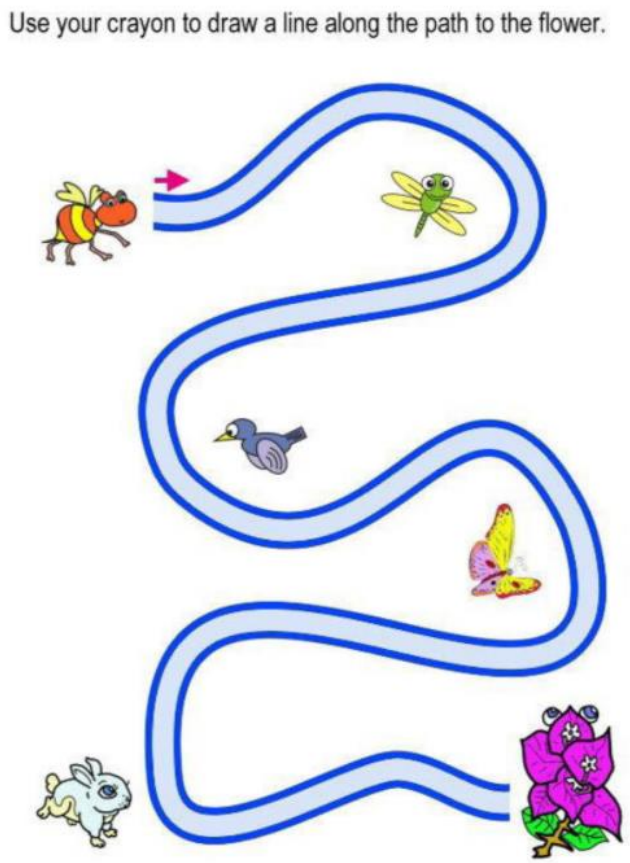
Monitoring and Observability

Monitoring – process for collecting, storing, retrieving, processing and visualizing state data

https://grafana.com/

Monitoring and Observability

Tracing – represents the entire path of a request: which services (or methods) it crosses, how long each step takes



Monitoring and Observability

Observability – a measure of how well internal states of a system can be inferred from knowledge of its external outputs

“Observability is about being able to ask arbitrary questions about your environment **without having to know ahead of time what you wanted to ask**” © Honeycomb

If you want to look fancy, you can write o11y instead of Observability (or squeeze it into a tweet limit)

Monitoring and Observability: Logging

Logs – events written to a file or storage (almost always time-stamped)

Monitoring and Observability: Logging

Plain-text (unstructured) log: just a string

"2010-01-01 12:34:56.0000 Info: Hello, world"

Monitoring and Observability: Logging

Plain-text log **pros**

"2010-01-01 12:34:56.0000 Info: Hello, world"

+ easy to read by humans
+ easy to implement
+ cheap to store

Monitoring and Observability: Logging

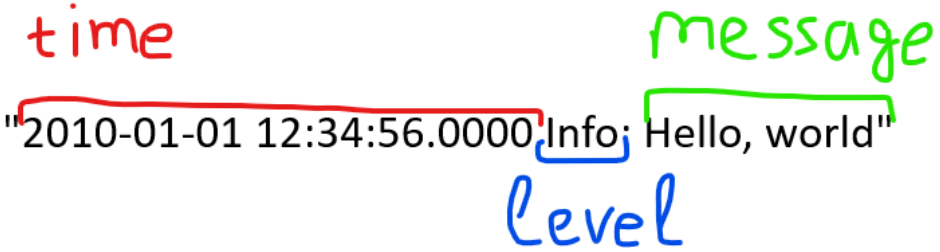
Plain-text log **cons**

"2010-01-01 12:34:56.0000 Info: Hello, world"

- limited filtering capabilities
- limited analysis options

Monitoring and Observability: Logging

Structured logs: all log events follow a defined structure.




```
{
"time": "2010-01-01 12:34:56.0000",
"level": "Info",
"message": "hello, world",
"user": "test-user",
"env":"prod",
"hostname":"vm-0001"
}
```

Monitoring and Observability: Logging

The most popular format for structured logs is json

Big companies often prefer binary encoding, for example, protobuf or avro

Monitoring and Observability: Logging

Structured logs **pros**:

+ unlimited filtering capabilities
+ unlimited analysis options
+ (could) enforce format
+ (could be) small messages

```
{
"time": "2010-01-01 12:34:56.0000",
"level": "Info",
"message": "hello, world",
"user": "test-user",
"env":"prod",
"hostname":"vm-0001"
}
```

Monitoring and Observability: Logging

Structured logs **cons**:

- hardly readable by humans
- harder to implement
- could be expensive to store
- requires infrastructure to process, aggregate, ingest, index


```
{
"time": "2010-01-01 12:34:56.0000",
"level": "Info",
"message": "hello, world",
"user": "test-user",
"env":"prod",
"hostname":"vm-0001"
}
```

Monitoring and Observability: Logging

Demo

Monitoring and Observability: Metrics

Metrics – numerical representation of data

Monitoring and Observability: Metrics

Popular metric types:

- Counter/Gauge – a number, which goes up or down
- Histogram – samples observations into buckets

Monitoring and Observability: Metrics

Often metrics are stored in *Time-Series Database (TSDB)*: software optimized to store ordered by time data-points

But could be also stored as structured records in any other storage type

Monitoring and Observability: Metrics

Often time-series record is identified by name + unique set of **dimensions** *<name>{<label_name>=<label value>, ...}*:

- `api_http_requests_total{method="POST", handler="/messages"}`
- `api_http_requests_total{method="GET", handler="/messages"}`

Monitoring and Observability: Metrics

Each unique label generates a new time-series, thus:

- Save one more measurement into **existing time-series** – *cheap*
- Save one more measurement into **new time-series** – *expensive*

Monitoring and Observability: Metrics

- Querying *the same* time-series– *cheap*
- Querying *multiple* time-series – *expensive*

Monitoring and Observability: Metrics

High-cardinality label – label that could have a lot of unique values

When you use TSDB, better not to use high-cardinality labels

Monitoring and Observability: Metrics

Good label:

- HTTP Verb and Status Code
- VM id

Bad label:

- UserId
- RequestId

Monitoring and Observability: Metrics

Service-Level Objective (**SLO**) – target availability: how long system can be unavailable.

Service-Level Agreement (**SLA**) – SLO promised to others (users)

Service-Level Indicator (**SLI**) – units of SLO

sli, sla, slo by Google

Monitoring and Observability: Metrics

Compute AWS Service Level Objective

AWS will use commercially reasonable efforts to make the Included Services each available for each AWS region with a Monthly Uptime Percentage of at least 99.99%

Monitoring and Observability: Metrics

Compute AWS Service Level Agreement

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.99% but equal to or greater than 99.0%	10%
Less than 99.0% but equal to or greater than 95.0%	30%
Less than 95.0%	100%

Monitoring and Observability: Metrics

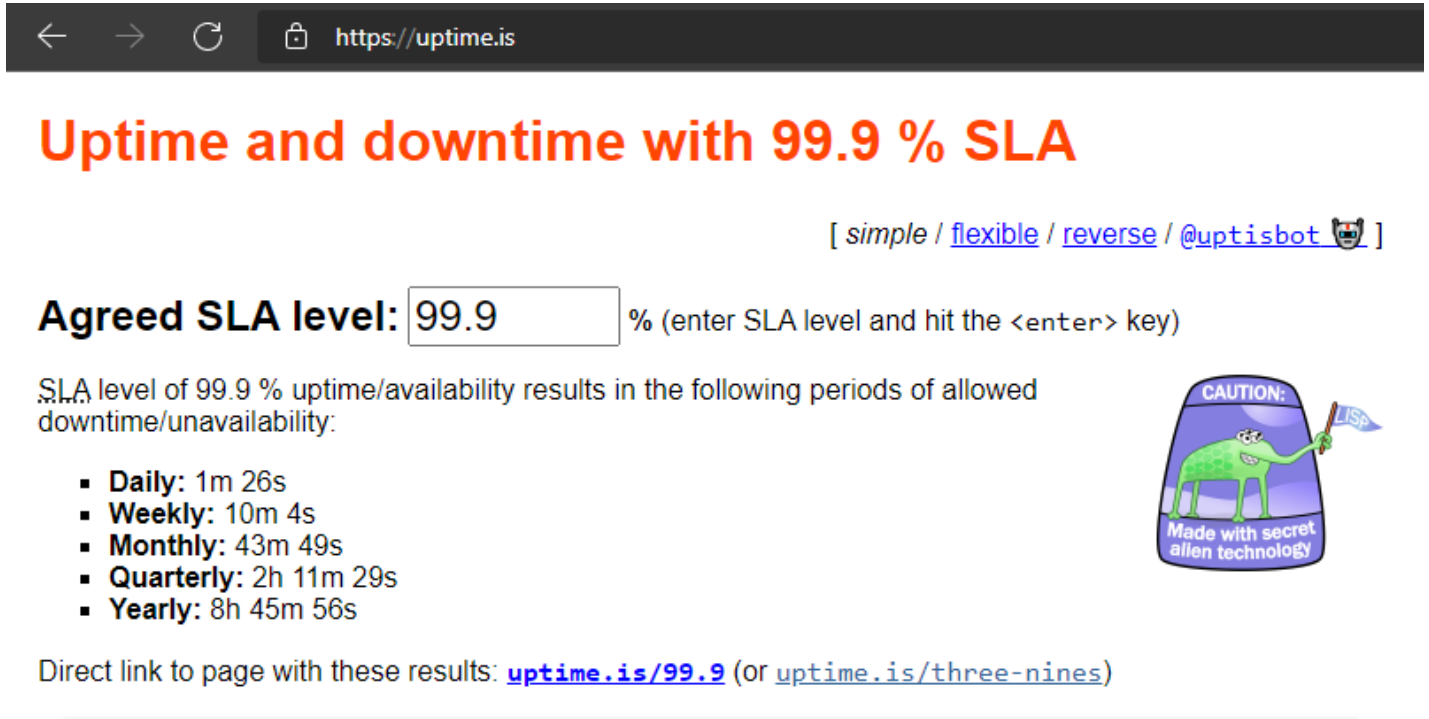
Compute AWS Service Level Indicator

*"Monthly Uptime Percentage" is calculated by subtracting from 100% the **percentage of minutes** during the month in which … [service] was **in the state of Unavailability***

*"Unavailable" and "Unavailability" mean … for Single EC2 Instances, when your Single EC2 Instance **has no external connectivity**.*

Monitoring and Observability: Metrics

Uptime calculator



Monitoring and Observability: Metrics

p.s.

Internal and External SLO often are different

Monitoring and Observability: Metrics

Demo

Monitoring and Observability: Traces

Finding logs and metrics associated with the same action/request could be tricky in a distributed system.

Monitoring and Observability: Traces

Tracing is an attempt to address this problem.

Trace represents the entire journey of a request/action as it moves through all the nodes/modules of a distributed system.

Monitoring and Observability: Traces

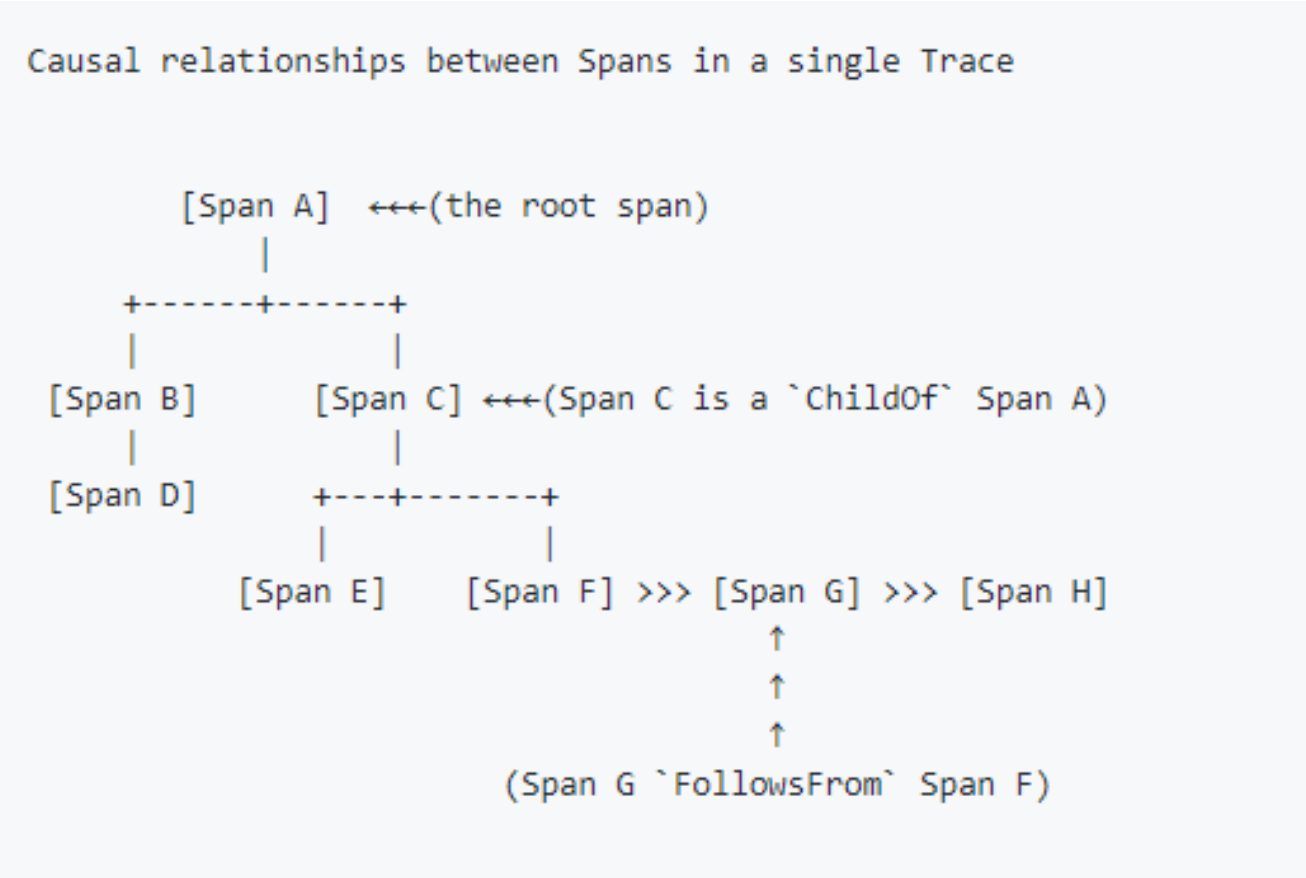
OpenTelemetry proposes vendor-neutral specification for trace model

Monitoring and Observability: Traces

OpenTelemetry introduces data model:

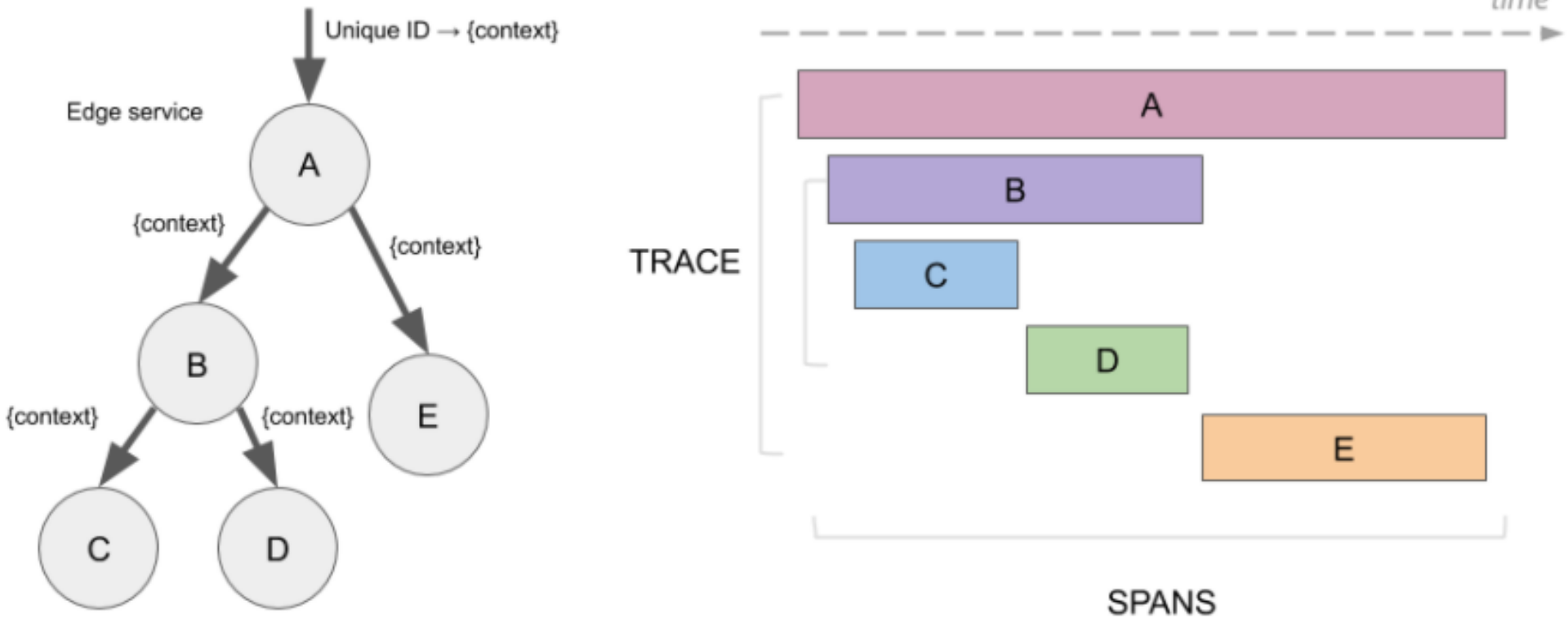
- *Trace* is directed acyclic graph (DAG)
- Each edge of this graph is named *Span*
- *Spans* could have two relation types: *ChildOf* and *FollowsFrom*
- *Spans* can have associated *Timestamp, Attributes, Links*, *Context*

Monitoring and Observability: Traces



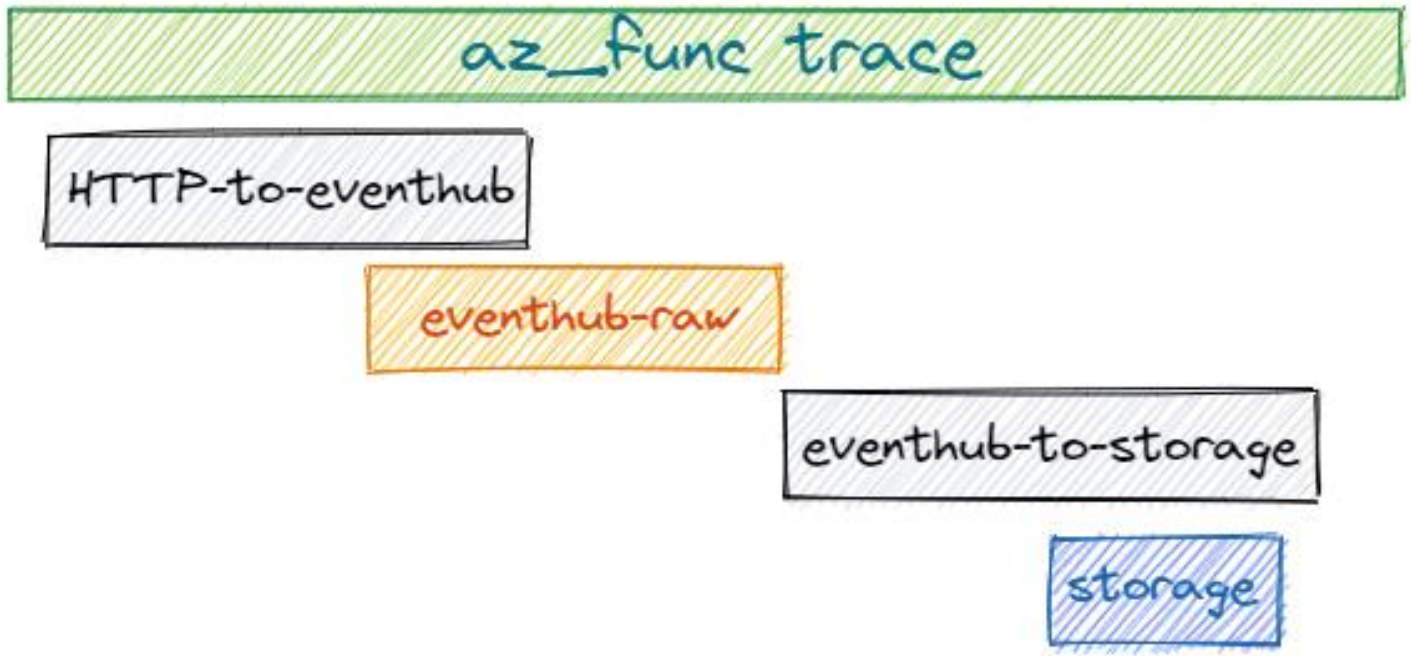
https://github.com/opentracing/specification/blob/master/specification.md#the-opentracing-data-model

Monitoring and Observability: Traces



https://www.jaegertracing.io/docs/1.22/architecture/

Monitoring and Observability: Traces



Monitoring and Observability: Traces

To build a trace:

- Propagate *trace-id* with each request
- Write events with attached *trace-id*
- Aggregate events from all the services into a single storage
- Group all events from the storage that have the same *trace-id*

Monitoring and Observability: Traces

Demo

Monitoring and Observability: Alerts

Alert – action based on a criteria

Monitoring and Observability: Alerts

Alert could be based on

- Logs
- Metrics
- Billing

Monitoring and Observability: Alerts

Alert, often, is a notification: a message sent to a communication channel

Monitoring and Observability: Alerts

But instead of sending a message you can make a request or invoke an API

Monitoring and Observability: Alerts

With alerts you can:

- post a warning to messenger (for example, Slack or email)
- make a phone call or send SMS
- invoke a Lambda function

Monitoring and Observability: Alerts

Demo

- Billing alert
- Metric alert
- Logs alert

Monitoring and Observability

All-in-one solutions:

- Azure Monitor; Amazon CloudWatch; Google Stackdriver
- Datadog, Splunk, Elastic, New Relic, Dynatrace, Honeycomb

Metrics-only: Prometheus, Graphite, Victoria-Metrics

Traces-only: Jaeger, Zipkin

Logs-only: Elasticsearch (ELK), Grafana Loki

…and many-many-many others

Monitoring and Observability: Audit

Audit log – records/events answering the question: *“**who** did **what** and **when**?”*

Monitoring and Observability: Audit

Some audit logs are enabled by default

Some – you must enable explicitly and route to a long-term storage

Monitoring and Observability: Audit

Audit log is crucial in terms of Security

Monitoring and Observability: Audit

Demo

Additional resources

- (articles + java samples) Terse logback: github and webpage
- (article) SRE fundamentals: SLIs, SLAs and SLOs
- (article) Distributed Tracing

Additional resources

(course summary ☺) Serverless to speed-up observability servers