# Challenges, Best Practices, and Solutions for Monitoring and Alerting with Big Data

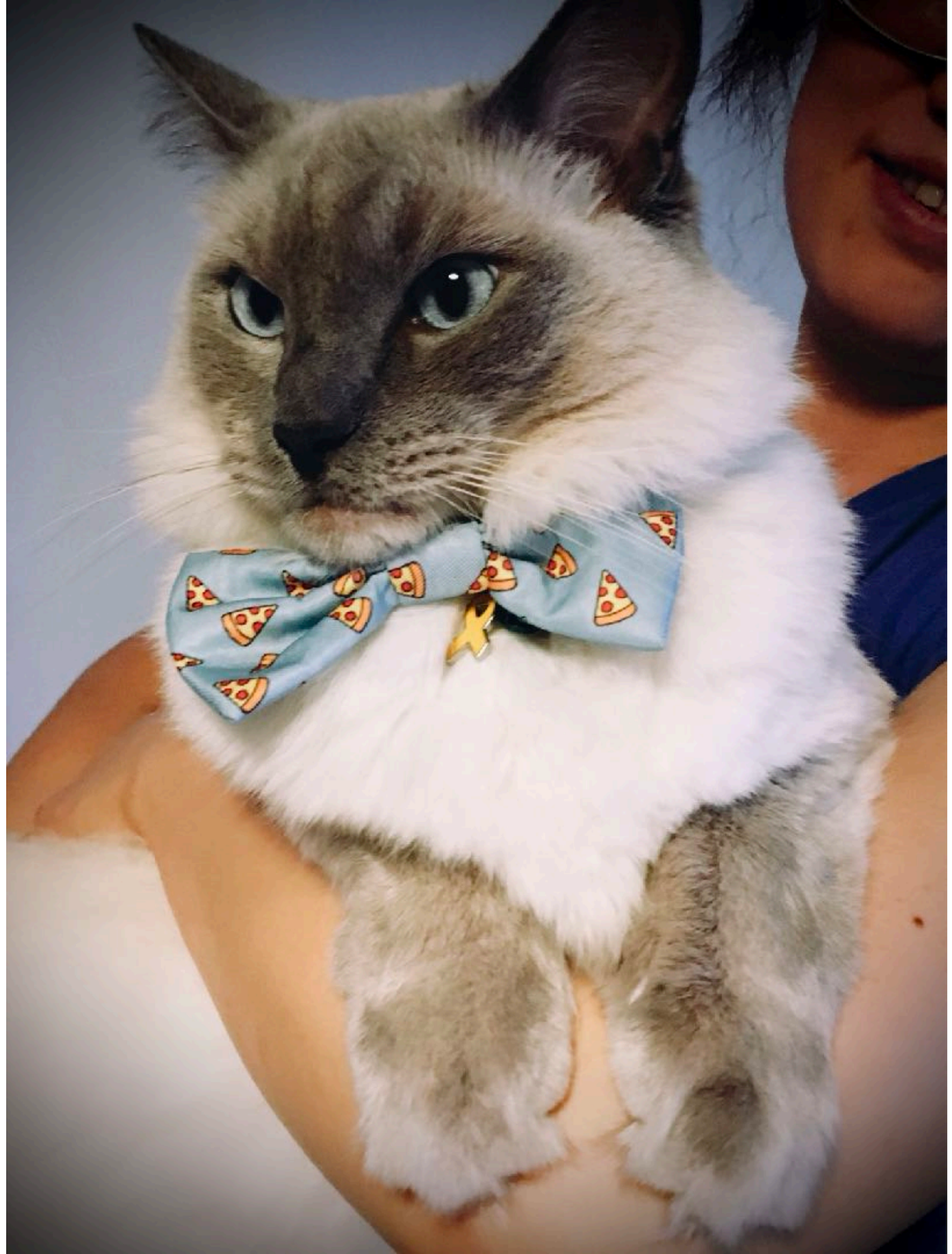
SRECon APAC 2022



**DANIEL O'DEA | SITE RELIABILITY ENGINEER | ATLASSIAN**

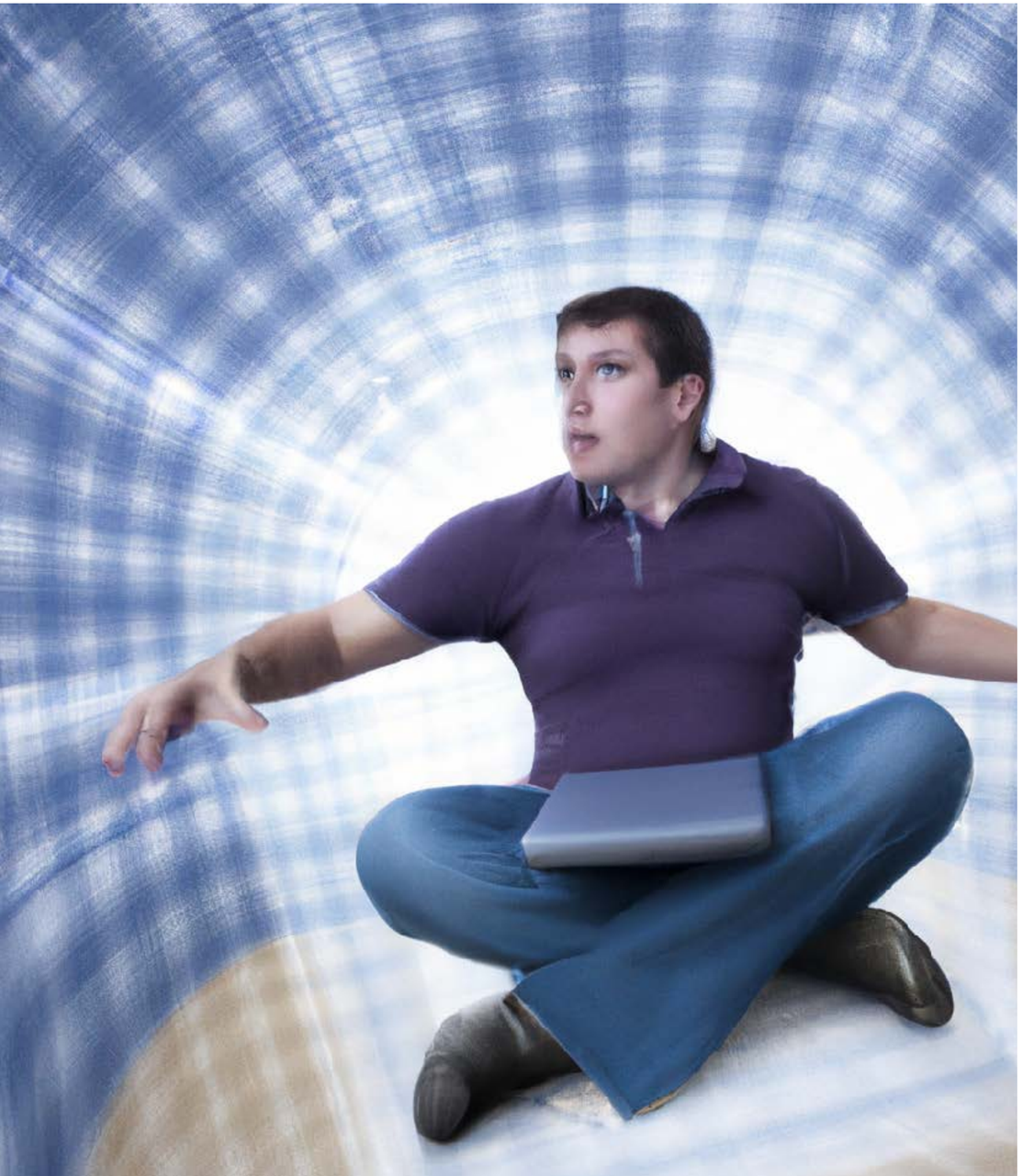
# RIP, TOBY.

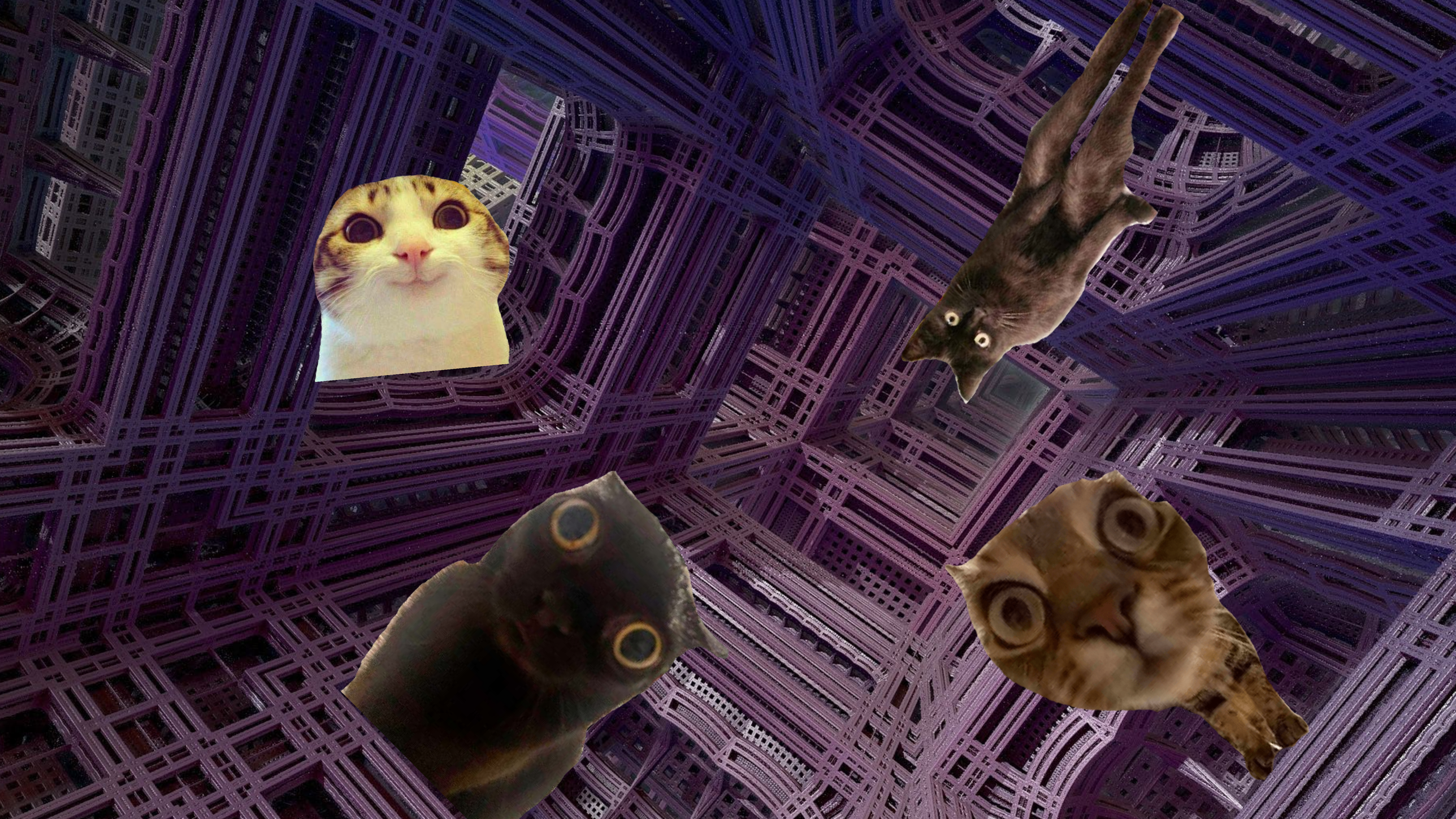
Our beloved boy, Toby *(2014-2022)*




Yakult
The Healthy Family Drink
FERMENTED MILK DRINK
Yakult

# There will be more AI-generated images, you have been warned

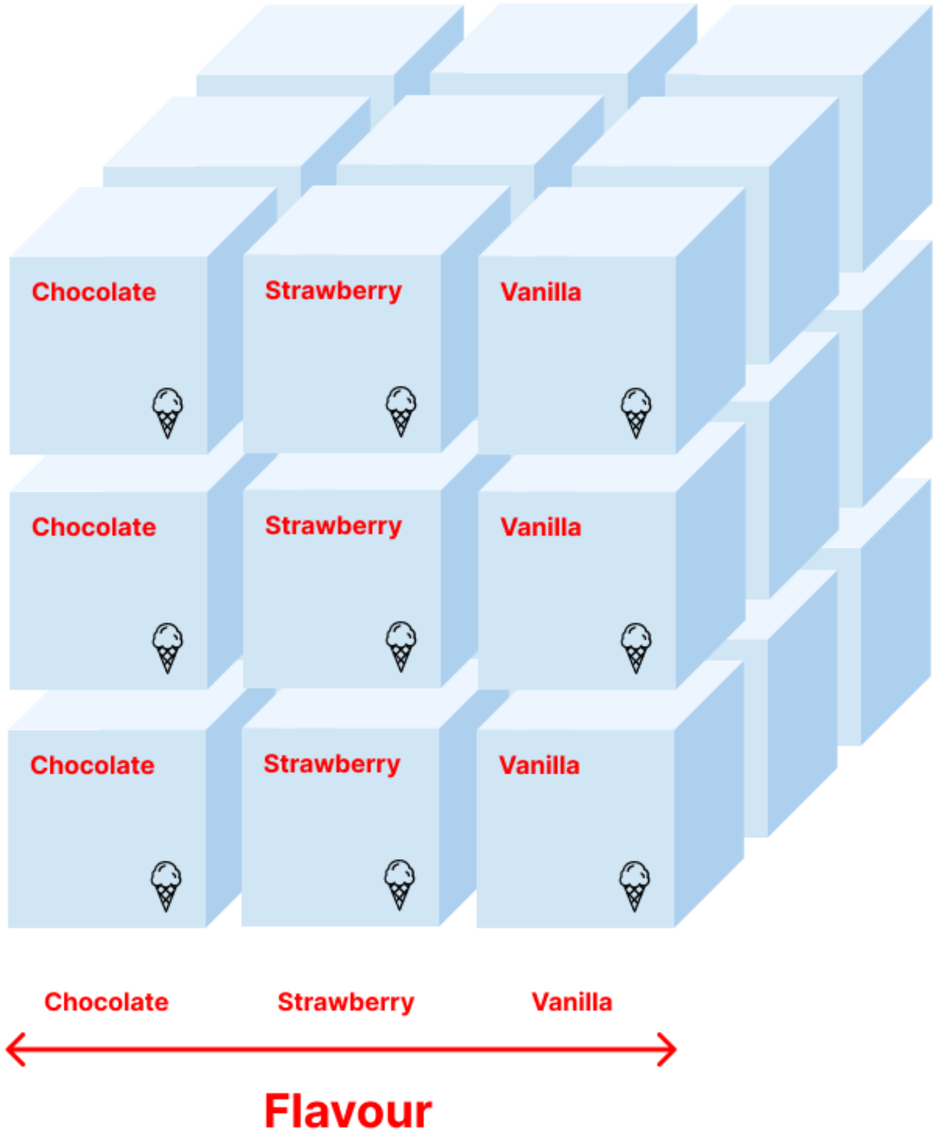




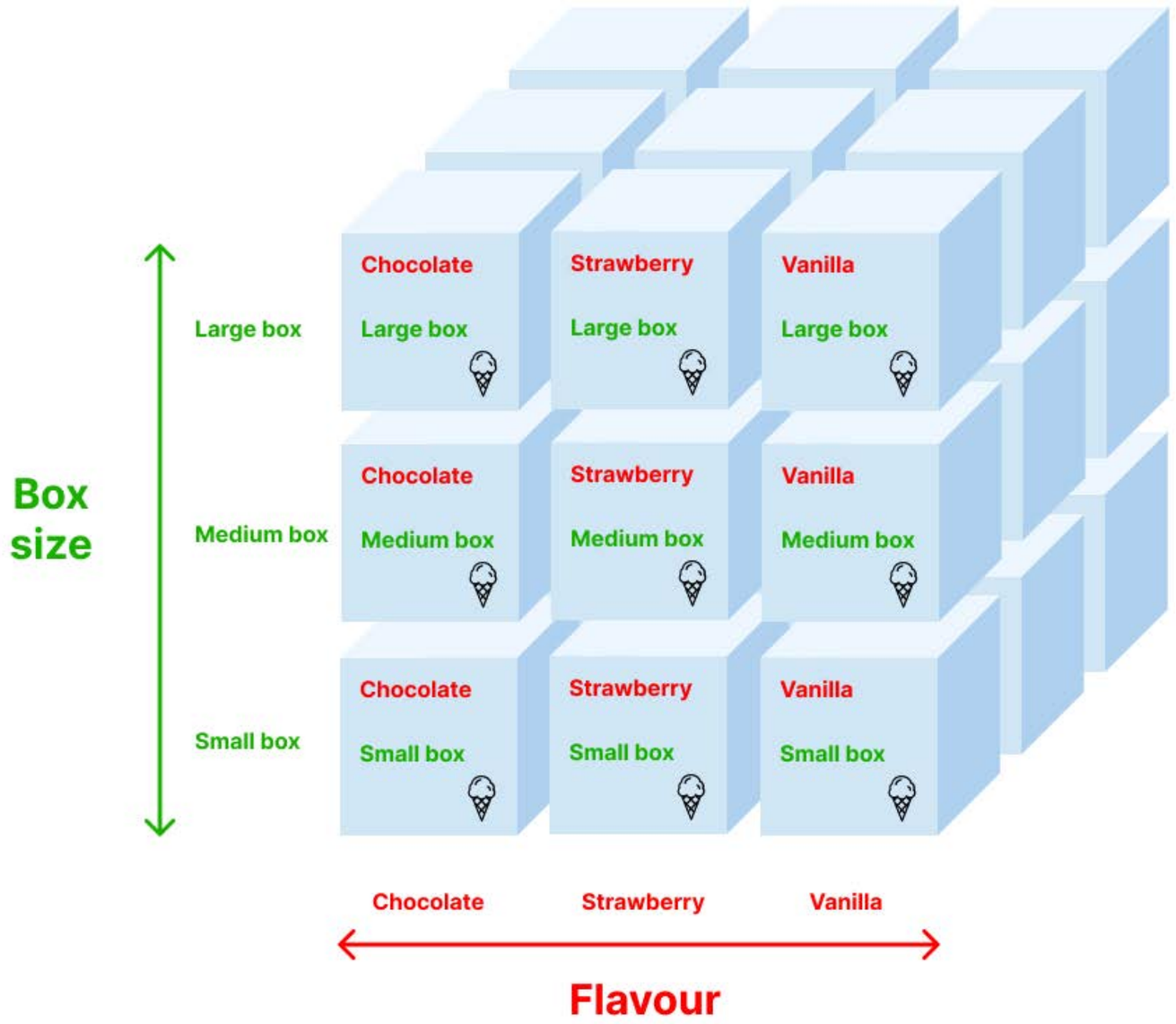
# Everyone gets the data!

product.icecream.count

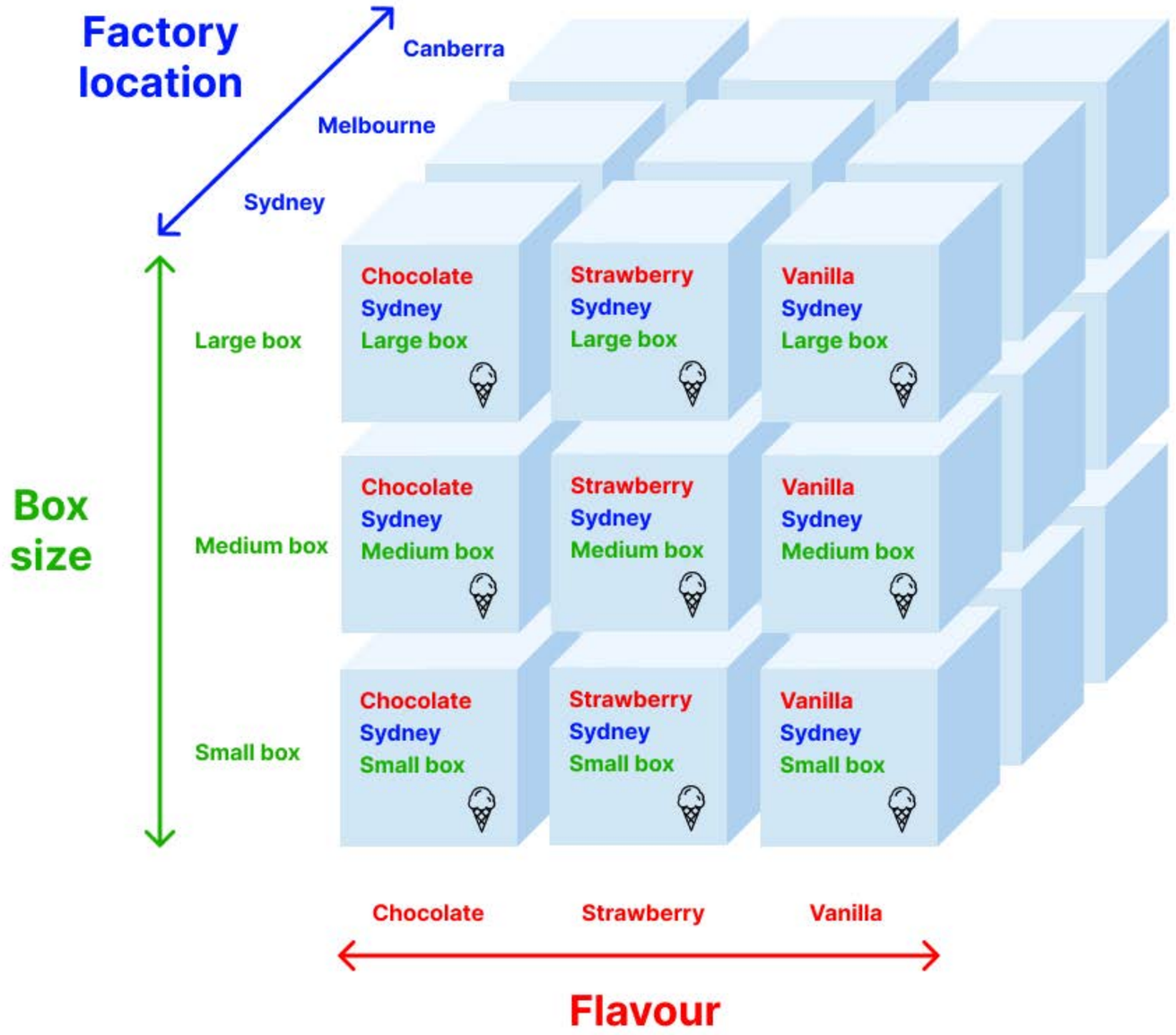
# product.icecream.count



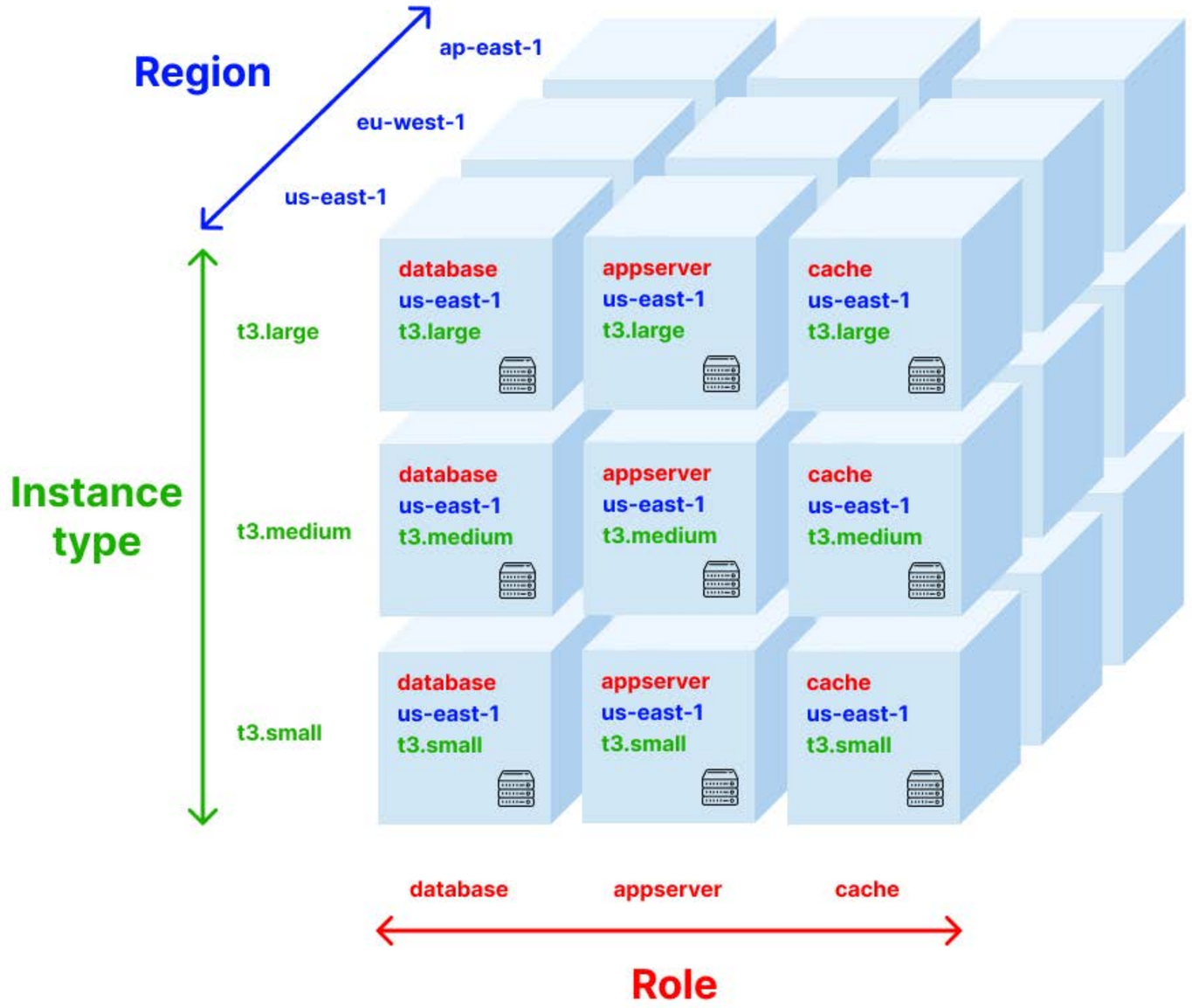
# product.icecream.count



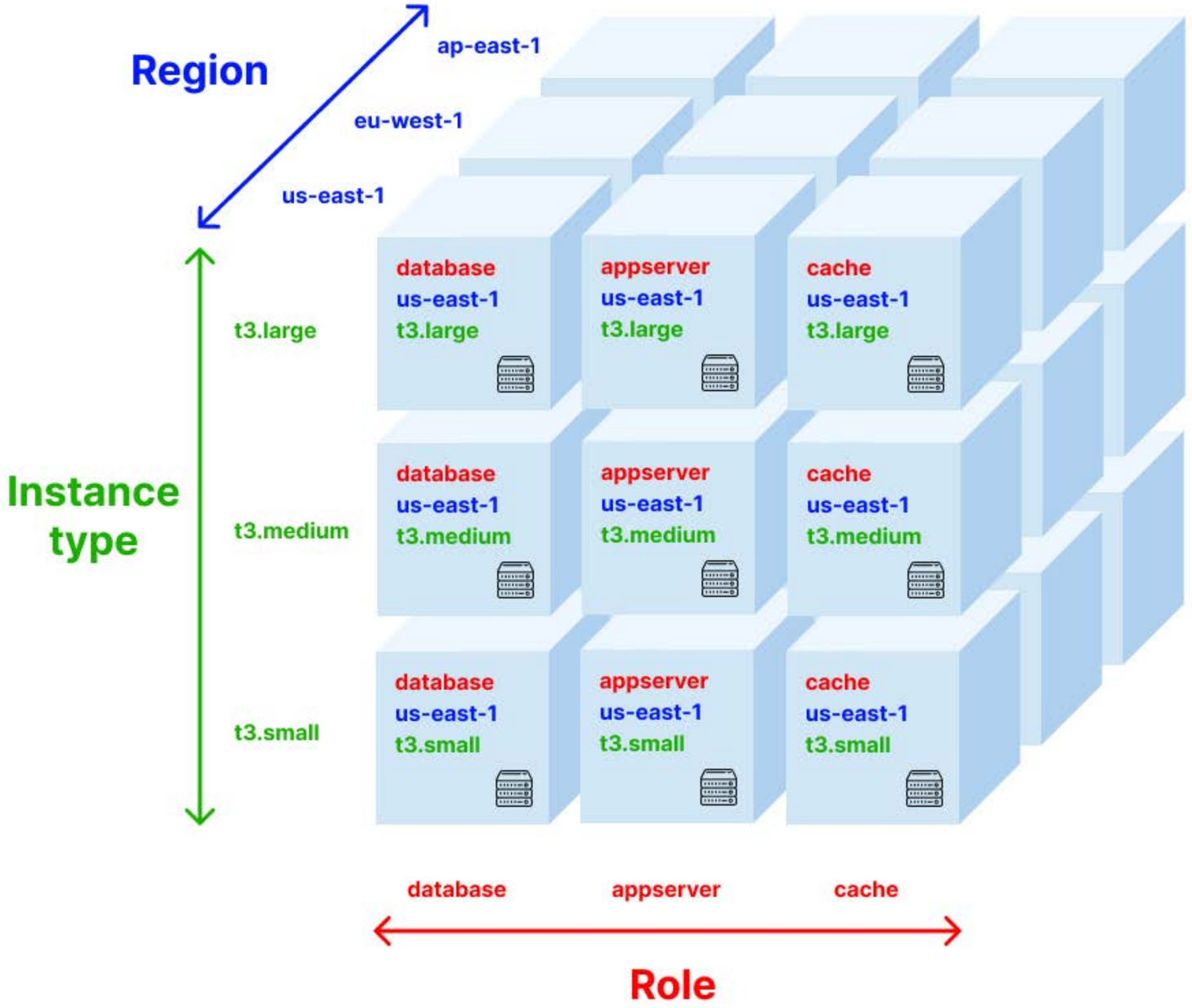
# product.icecream.count



# app.task.count



app.task.count

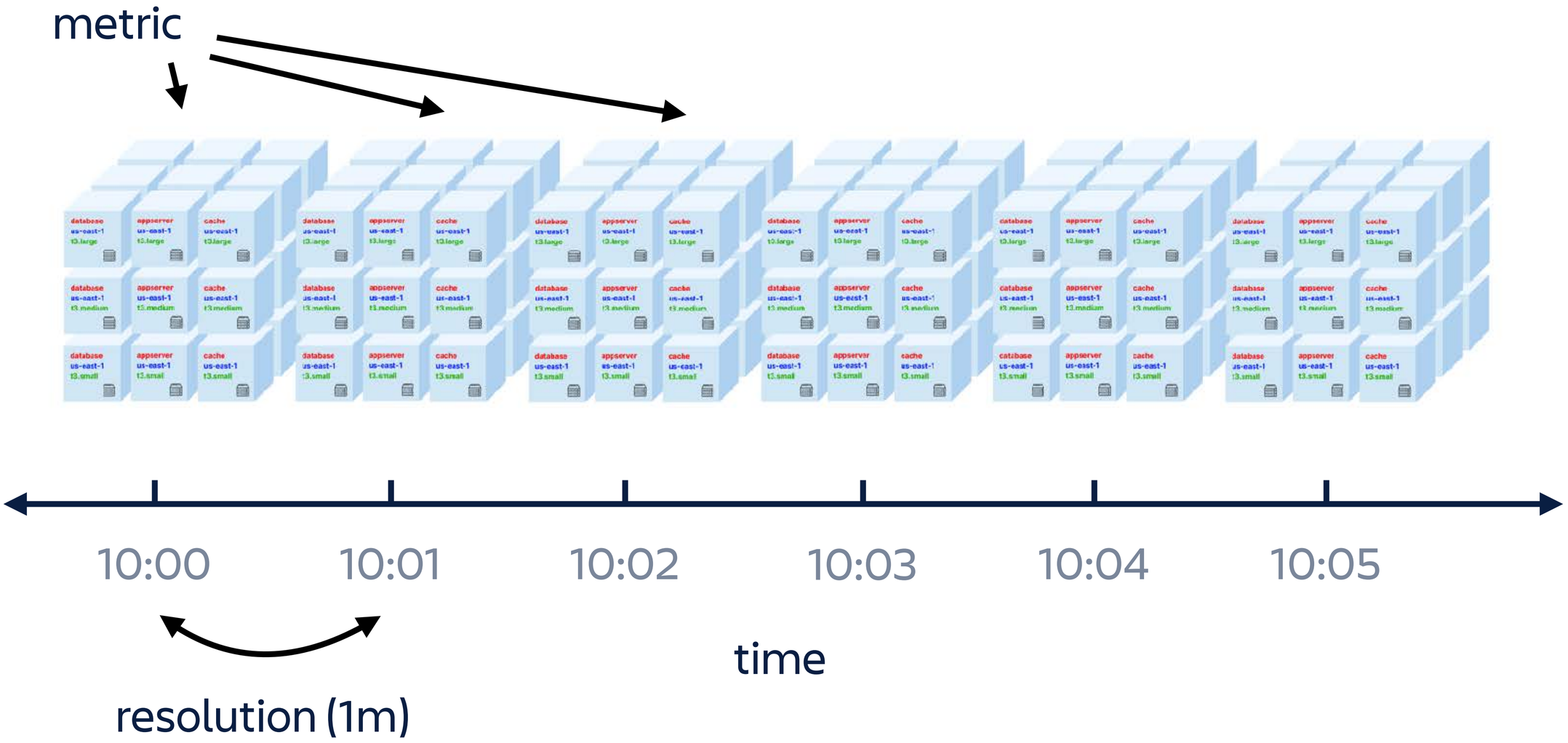



The **cardinality** of a metric is the product of the number of unique values within each tag/dimension.

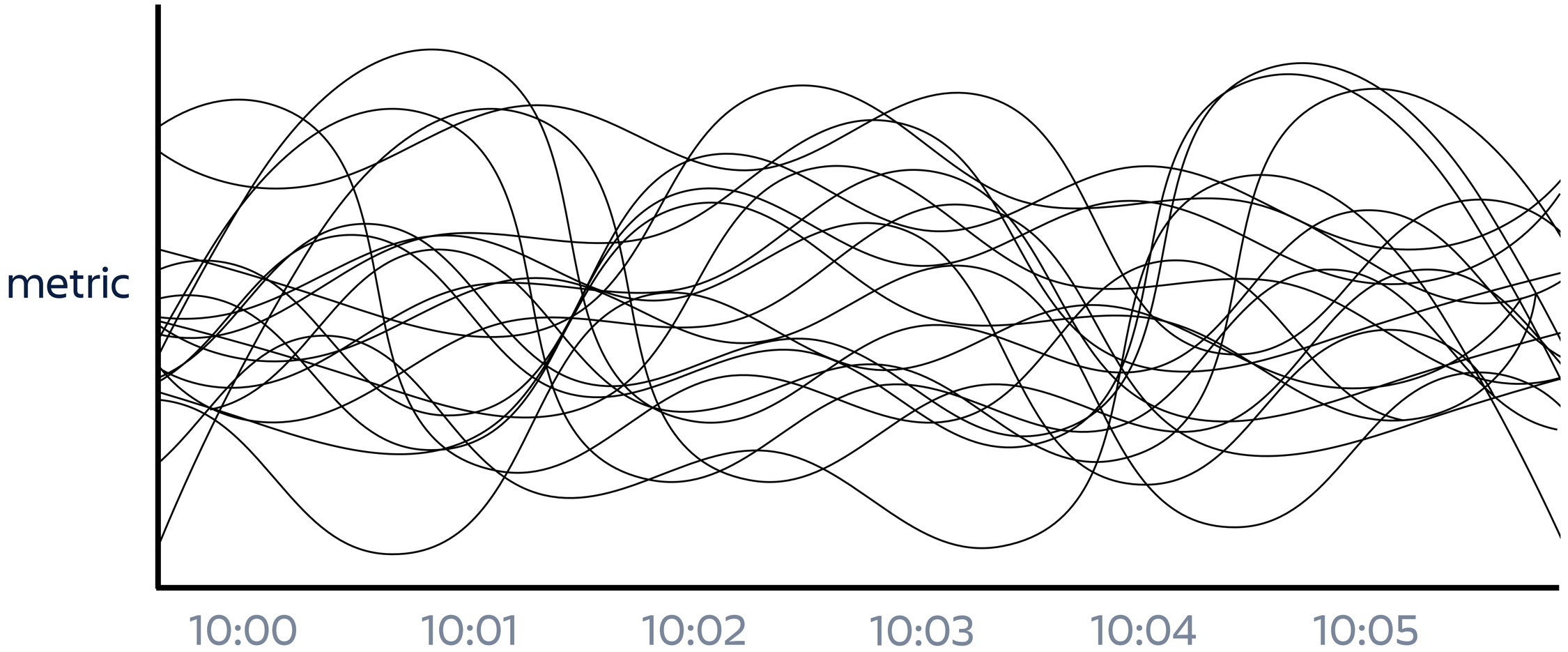
```
app.task.count {
  role: database,
  instance_type: t3.large,
  region: us-east-1
}
```

*Cardinality of* app.task.count*:*
3 x 3 x 3 = 27

# METRIC TIME SERIES (MTS)



# METRIC TIME SERIES (MTS)



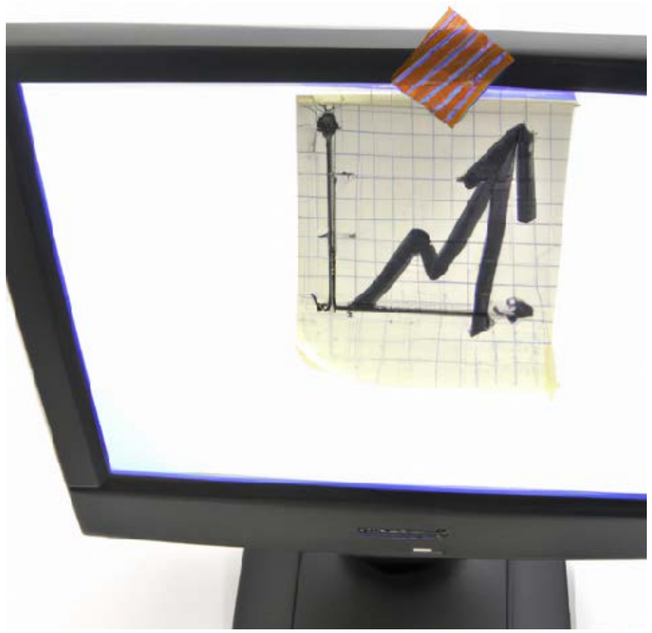
# WHAT'S THE (DATA) POINT?

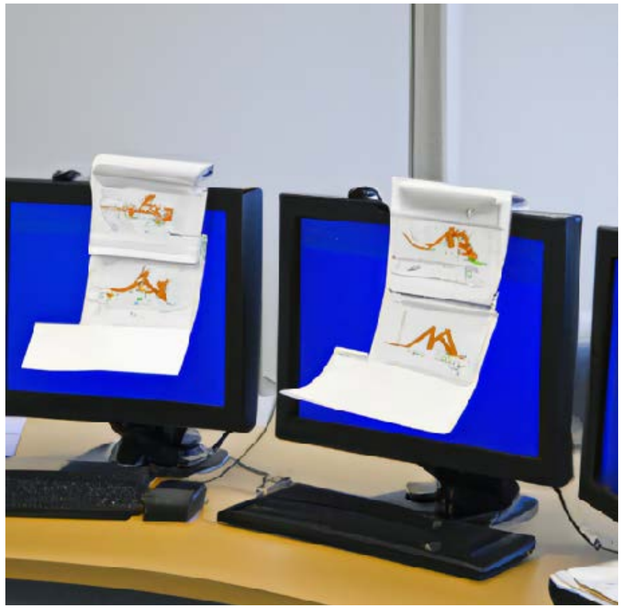
1. Improve the system
2. Track SLAs

# 1. IMPROVE THE SYSTEM

# 1. IMPROVE THE SYSTEM

Your monitoring setup, courtesy of Sam





# 1. IMPROVE THE SYSTEM

The faulty vanilla ice cream machines :(

# 2. TRACK SLAS

The children are in distress, and they’re letting us know.

# WHY DO WE HAVE HIGH CARDINALITY DATA?

# WHY DO WE HAVE HIGH CARDINALITY DATA?

Immutable infrastructure

# WHY DO WE HAVE HIGH CARDINALITY DATA?

Immutable infrastructure

Blue/green deploys

# WHY DO WE HAVE HIGH CARDINALITY DATA?

Immutable infrastructure

(forbidden choc-mould flavour)

Blue/green deploys

Long-term capacity planning

# WHY DO WE HAVE HIGH CARDINALITY DATA?

Immutable infrastructure

Blue/green deploys

(forbidden choc-mould flavour)

Long-term capacity planning

Per-customer metrics

# Jira

Micros

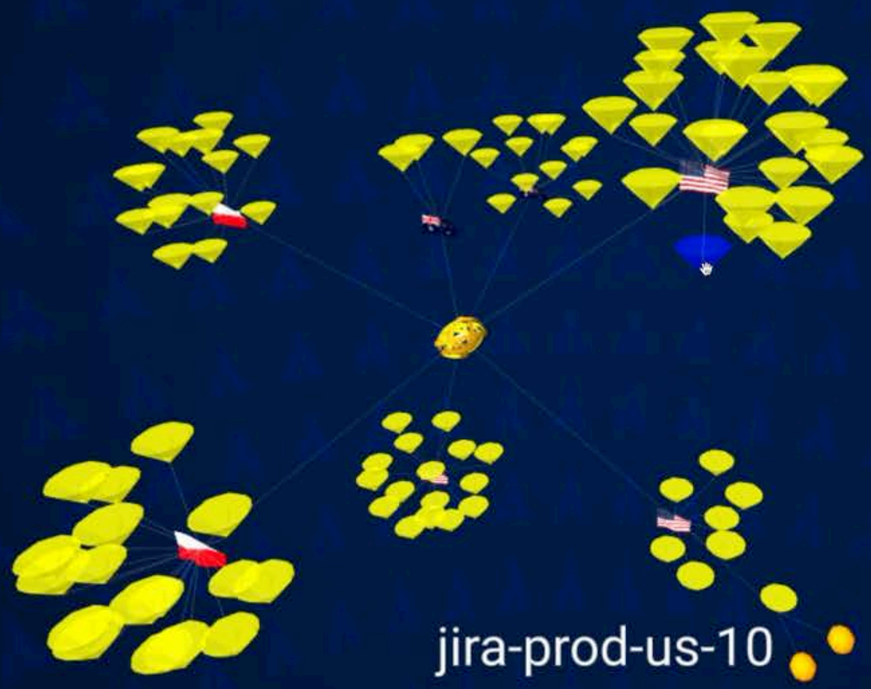
Micros: Micros abstracts AWS infrastucture so that we can run Jira internationally in different regions

# Jira

prod-east

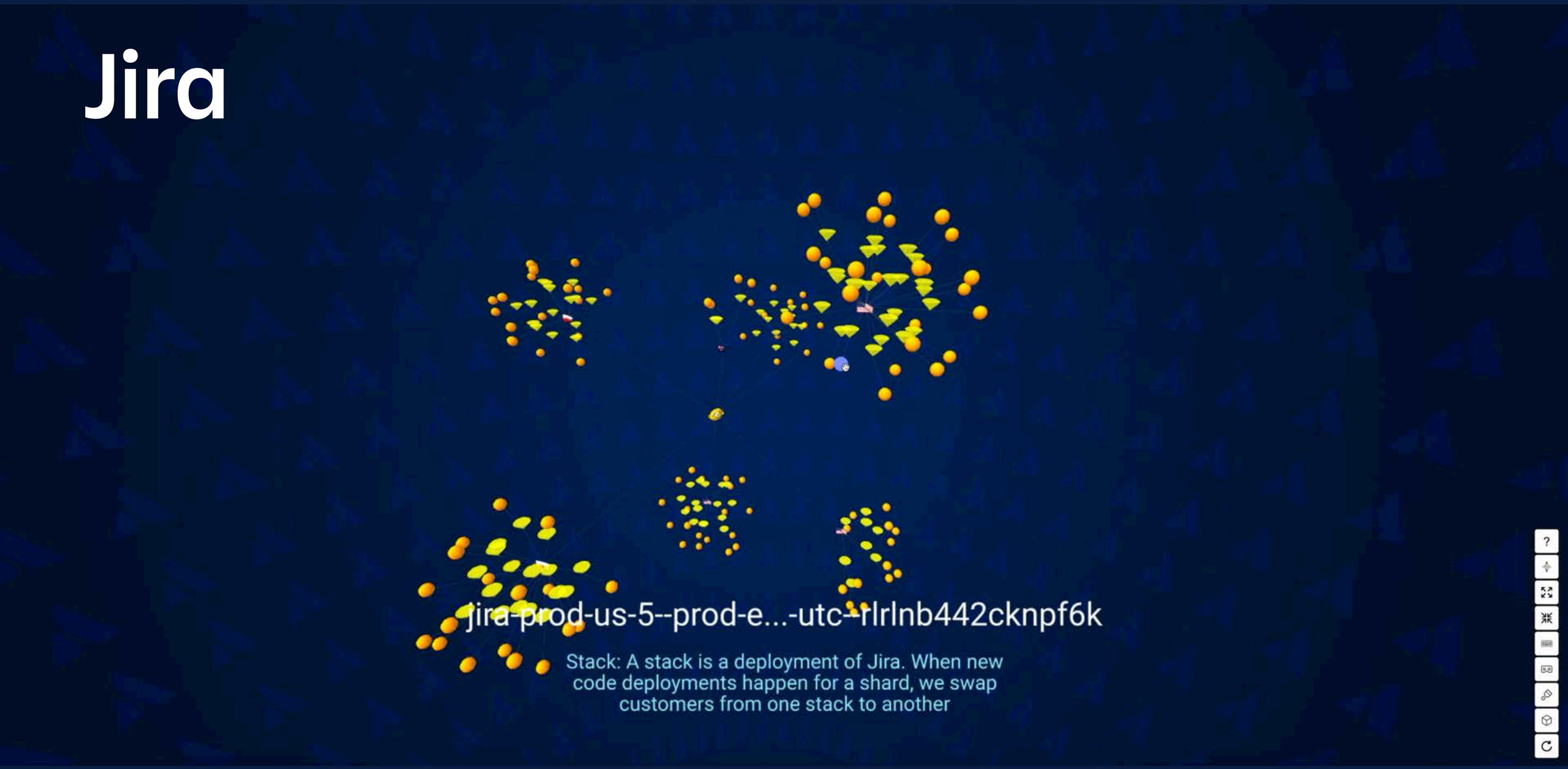
Micros Region: The AWS region holding resources

# Jira

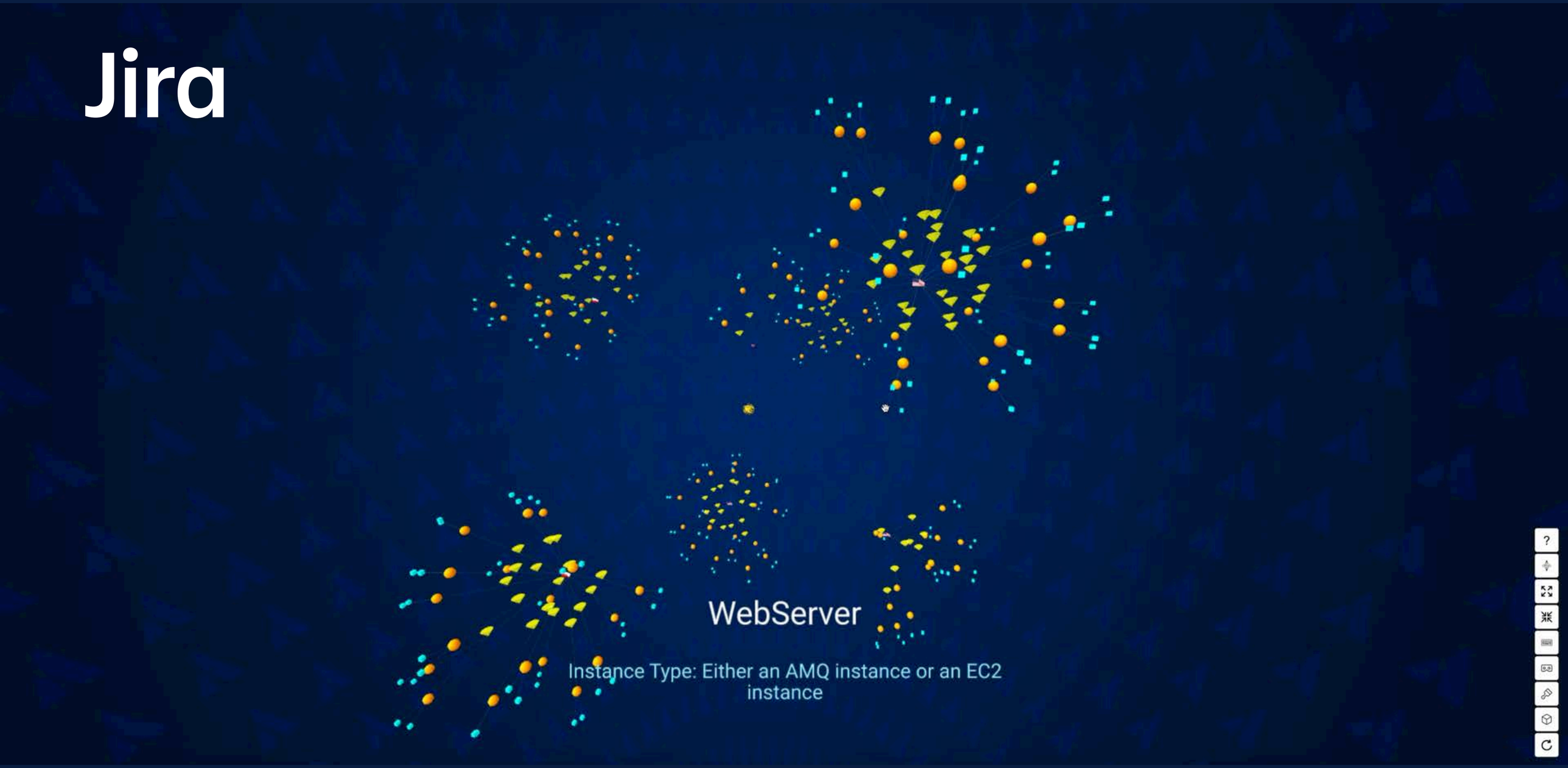



Shard: A shard groups a collection of customer deployments we call tenants

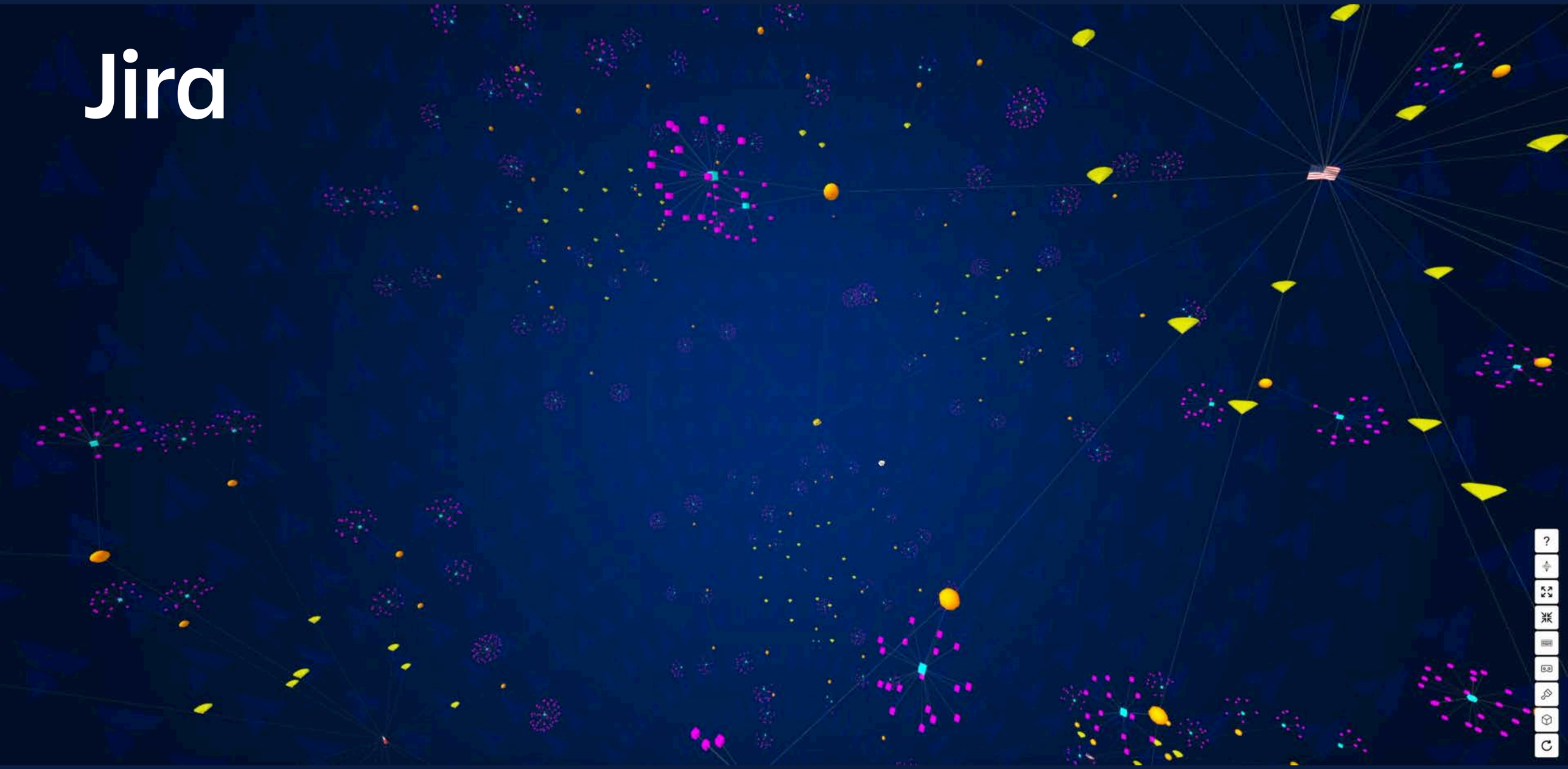
# Jira


jira-prod-us-5--prod-e...-utc-rlrlnb442cknpf6k
Stack: A stack is a deployment of Jira. When new code deployments happen for a shard, we swap customers from one stack to another

# Jira

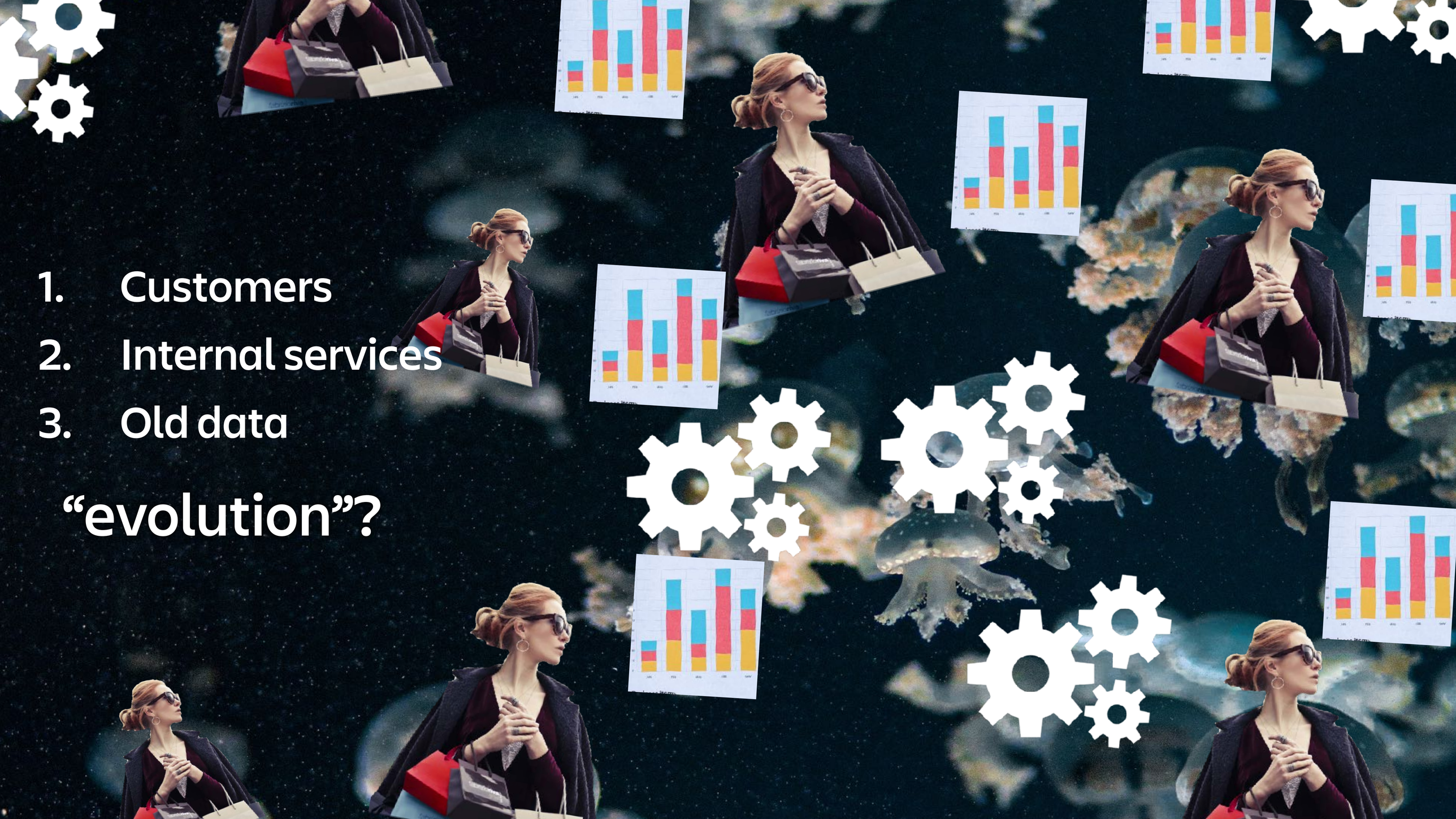

WebServer
Instance Type: Either an AMQ instance or an EC2 instance

# Jira




1. Customers


1. Customers
2. Internal services


1. Customers
2. Internal services
3. Old data
“evolution”?

## SQL

e.g. TimescaleDB

(◡‿◡✿)

## NO-SQL & SQL-LIKE

e.g. InfluxDB, OpenTSDB, Prometheus

??? oh

# TSDB CATEGORIES

1. Dependent 
2. Independent 
3. Proprietary 

# TSDB Categories

Dependent

Independent

Proprietary

## Dependent

Storage requirement on an existing DMBS for both short-term and long-term storage

- OpenTSDB
- TimescaleDB

# TSDB Categories

Dependent

Independent

Proprietary

## Independent

Optional storage requirement on an existing DMBS for long-term storage only

- Prometheus
- InfluxDB

# TSDB Categories

Dependent

Independent

**Proprietary**

## Proprietary $

$1-$∞. closed-source

- Splunk Infrastructure Monitoring splunk> 
- Splunk splunk>
- Gorilla 
- ByteSeries 
- Monarch 
- kdb+ kx

# WHY IS HIGH CARDINALITY DANGEROUS?

Query times

Storage requirements

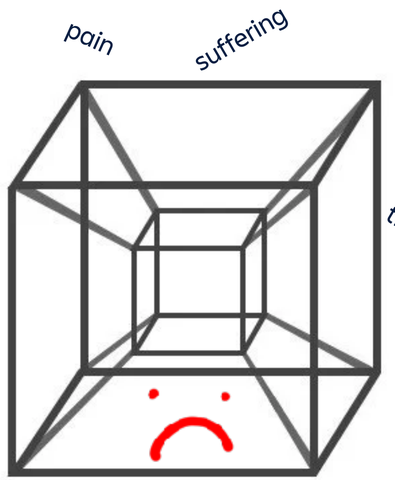

# In-memory TSDBs

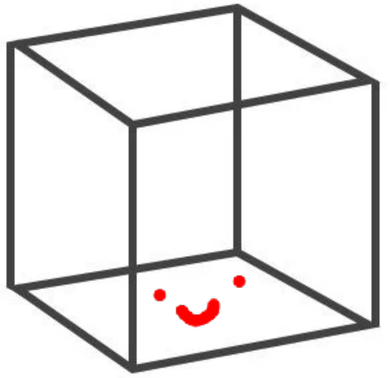
Fast but expensive.

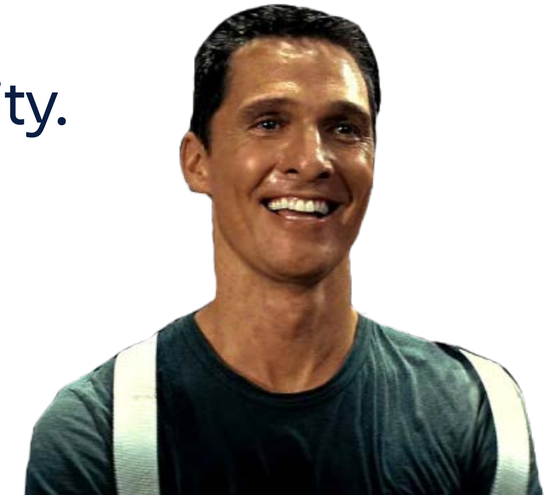

This could happen to you.

# HELP, HOW DO I GET OUT OF THE TESSERACT??

1. Track the high-cardinality metrics.
2. Only add dimensions when necessary.
3. Periodically delete or downsample old data.
4. Collect data less often.
5. Use sharding to get at least some visibility.
6. Split metrics up (sometimes).



# SPLITTING UP METRICS

```
metric_A: instance_id, region, response_code, routes
  |metric_A| = 100*10*10*10 = 100,000
```

This metric could do with some *slicing*.

```
metric_B: instance_id, region, response_code
  |metric_B| = 100*10*10 = 10,000

metric_C: region, response_code, routes
  |metric_C| = 10*10*10 = 1,000
```

Total system cardinality ***before***: |metric_A| = **100,000**
Total system cardinality ***after***: |metric_B| + |metric_C| = **11,000**

## VALUE-BASED DELTA COMPRESSION

# Proof of concept only!

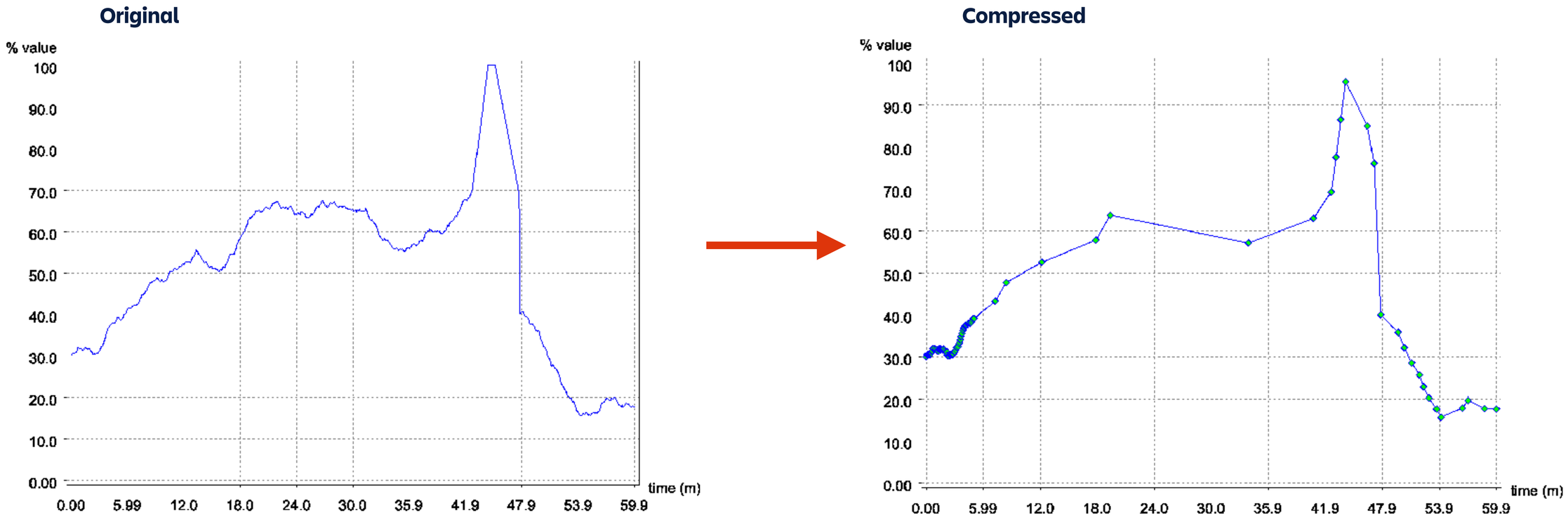
Proceed at your own risk. Experiment results may vary. A lot. Like really a lot, conduct your own tests be

# CONTENT-AWARE SCALE

# VALUE-BASED DELTA COMPRESSION

Only record data points if there is a statistically significant enough change in the data.

Like downsampling, but based on values *and* time, not just time.

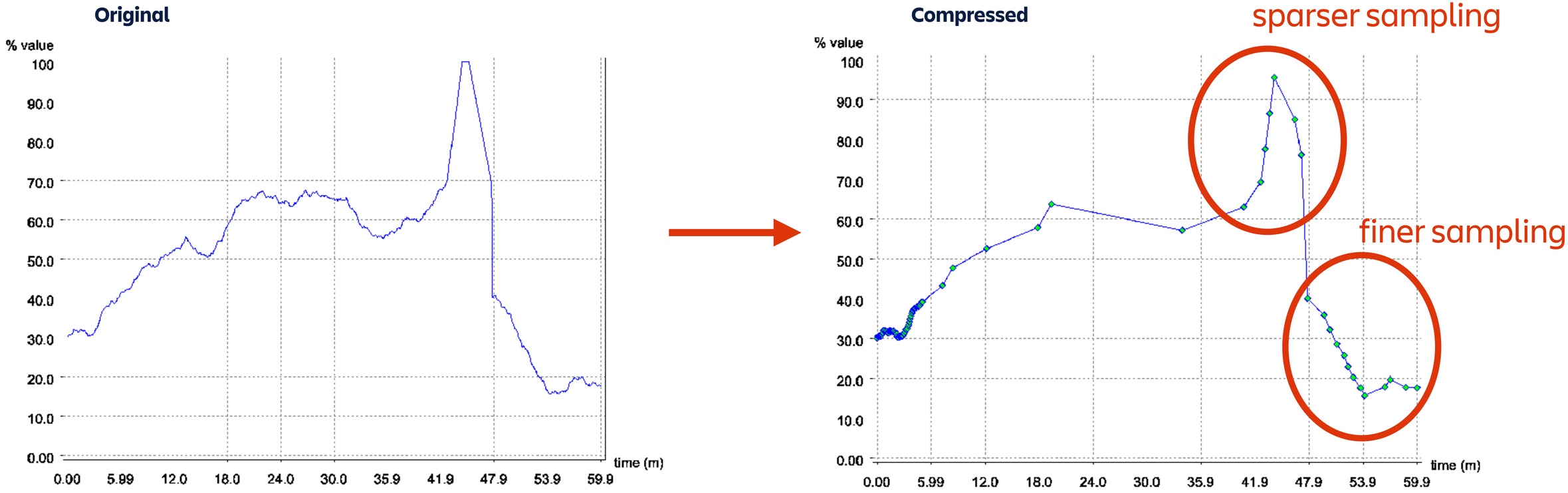



Example CPU usage % graph over 1 hour, 5-second source resolution

# VALUE-BASED DELTA COMPRESSION

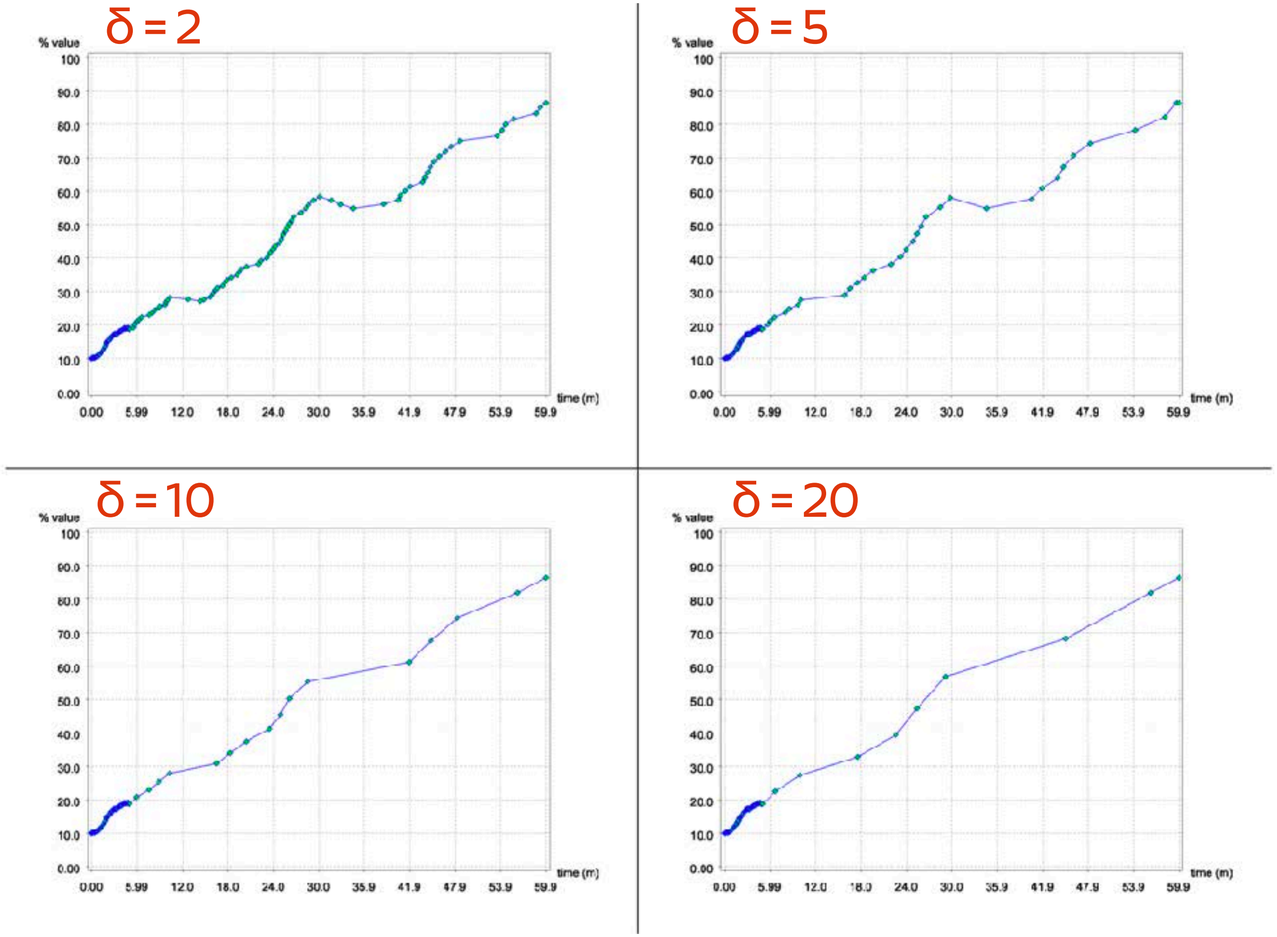
Only record data points if there is a statistically significant enough change in the data.

Like downsampling, but based on values *and* time, not just time.



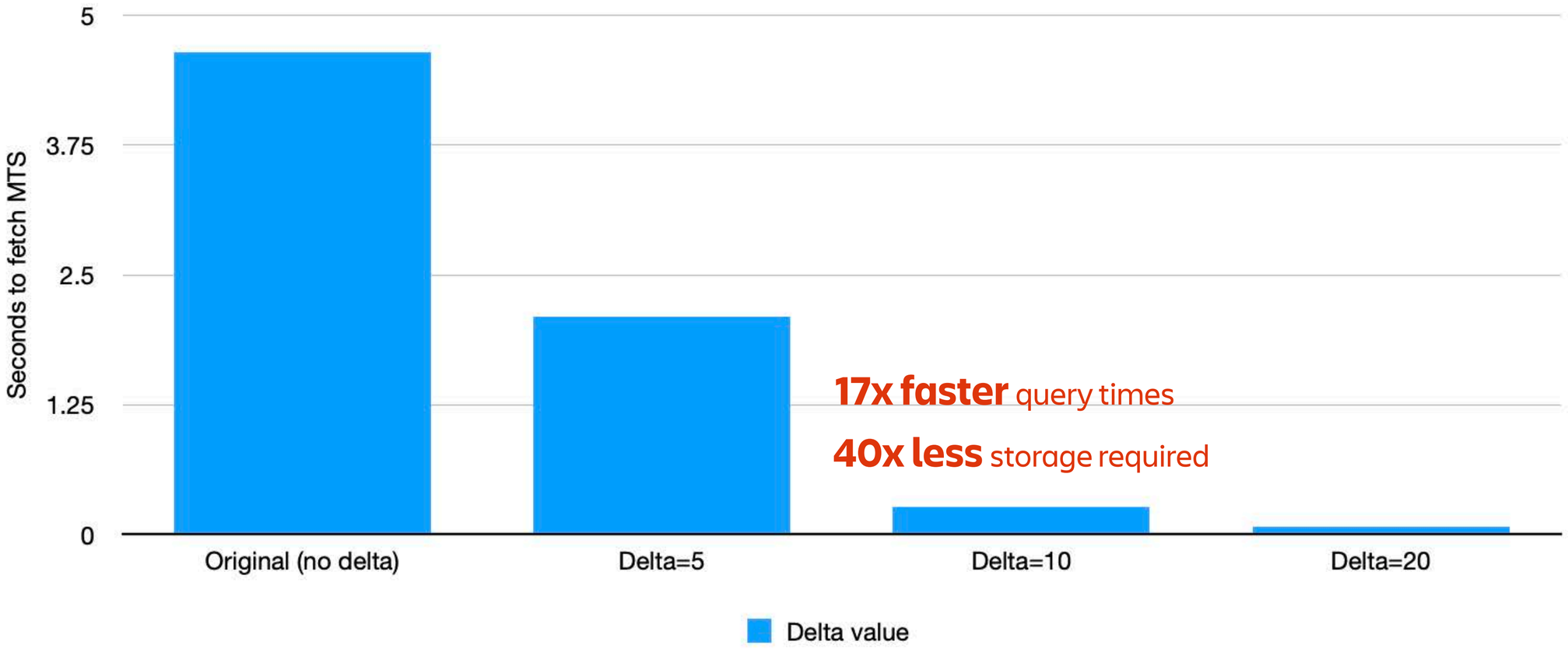

Example CPU usage % graph over 1 hour, 5-second source resolution

# VALUE-BASED DELTA COMPRESSION



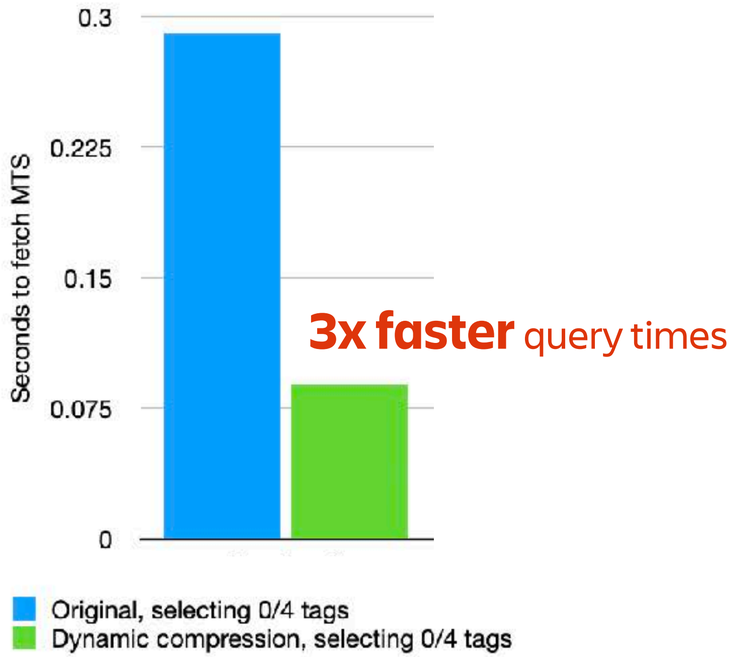

Example CPU usage % graph over 1 hour, 5-second source resolution

# VALUE-BASED DELTA COMPRESSION - SPEED-UP



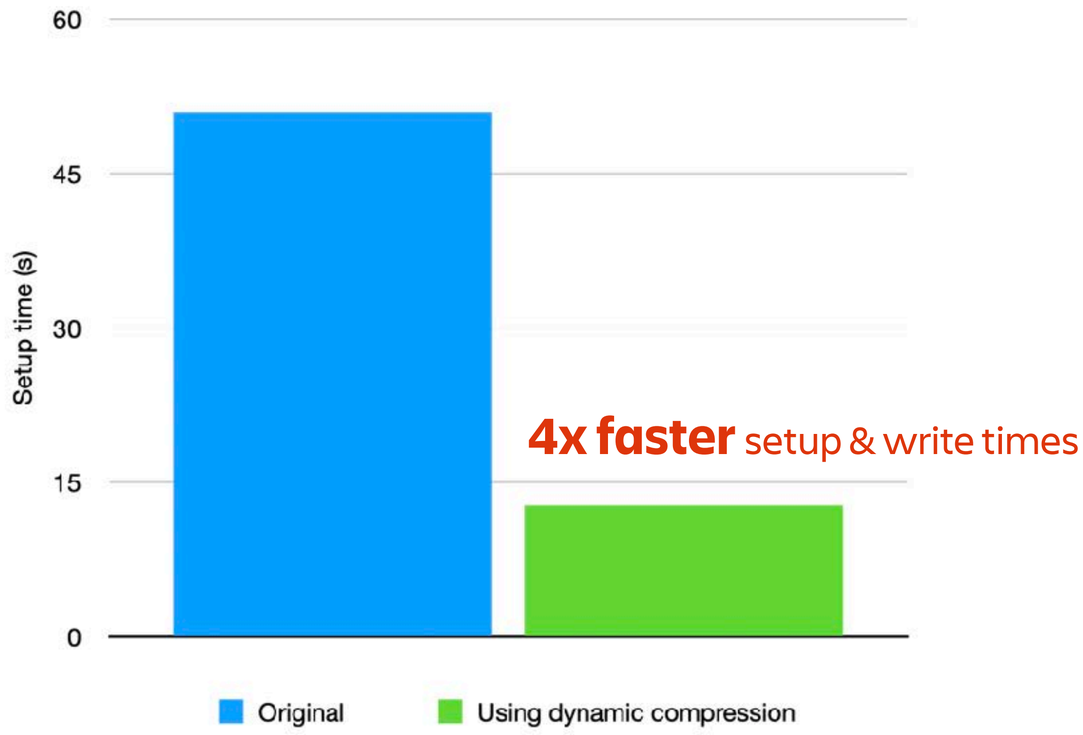

Query times for source data of 5s resolution over 1hr

# VALUE-BASED DELTA COMPRESSION - SPEED-UP




Query times for source data of 60s resolution over 1hr

# VALUE-BASED DELTA COMPRESSION - SPEED-UP

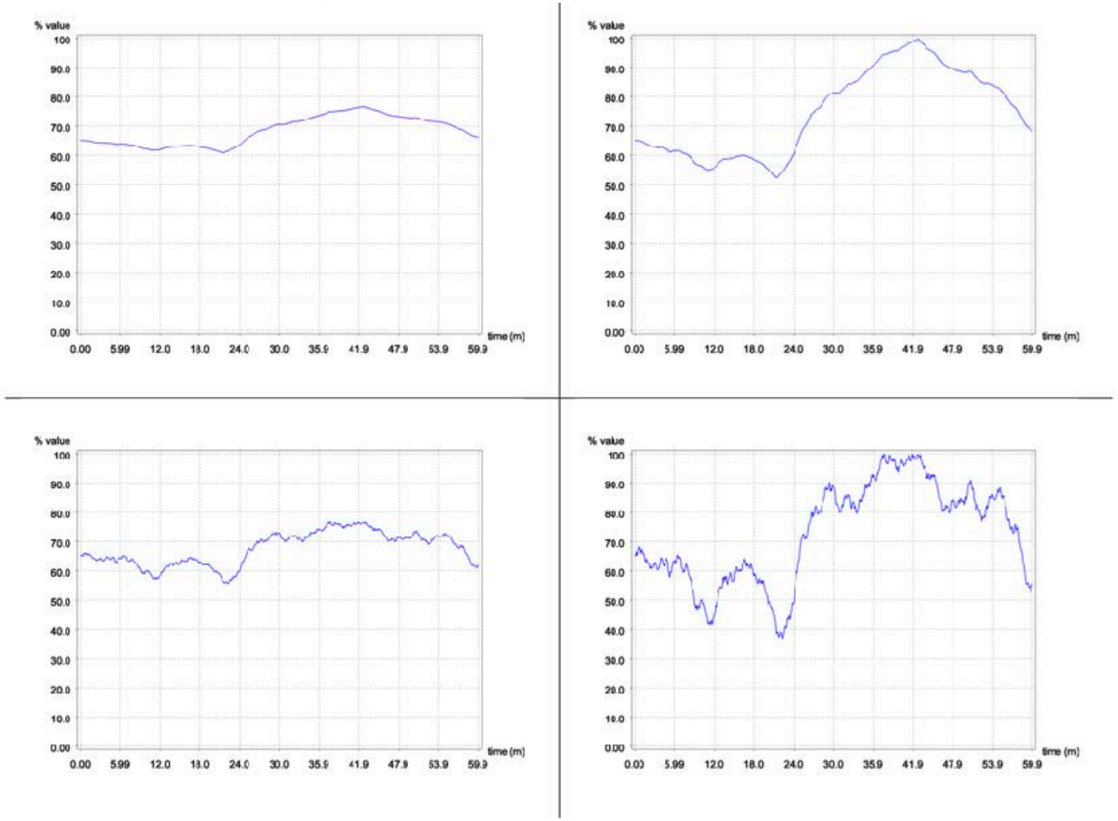



**4x faster** setup & write times

Query times for source data of 60s resolution over 1hr

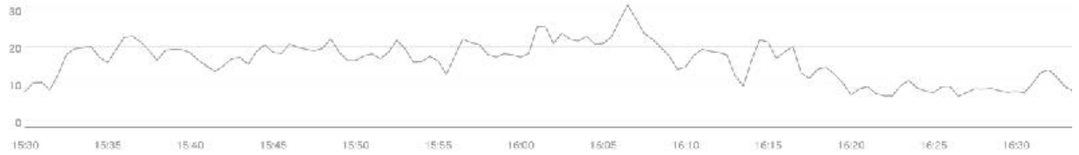
# TEST DATA - MIDPOINT DISPLACEMENT ALGORITHM

## Generated

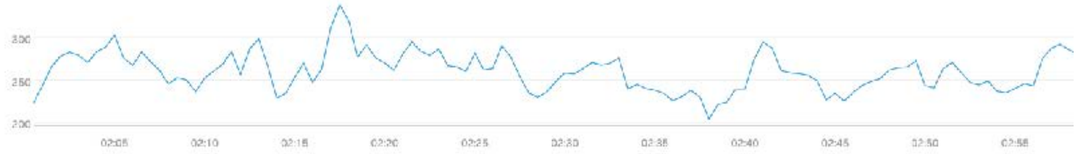



Midpoint displacement graph demonstrating input variables, for vertical displacement v and smoothness s, graphing an hour's worth of percentage data with points every 15 seconds. Top-left: v = 50,s = 1.2. Top-right: v = 150, s = 1.2. Bottom-left: v = 50, s = 0.7. Bottom-right: v = 150, s = 0.7

## Reference *(Atlassian)*



% CPU utilisation for an internal microservice, for 1 hour with a 30-second resolution



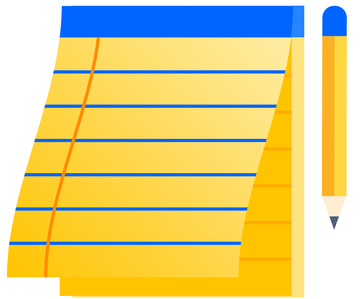
User CPU time, as opposed to the system CPU time, for Jira in prod-east, for 1 hour, for a particular shard, with a 30-second resolution

# TESTING ENVIRONMENT

- Implemented & tested with Yuvi
  - Lightweight proof-of-concept TSDB by Pinterest

# CAVEATS

- Delta of deltas timestamp encoding becomes ineffective.
- Less effective for highly variable data



# THE END - Q&A

#22apac-day2-track2
(or later if I ran out of time)