SKILup DAYS

by: DevOps Institute
ADVANCING THE HUMANS OF DEVOPS

Observability

A Socio- Engineering-Technology Problem

Name: Shivagami Gugan
DevOps Institute Ambassador
Head of SRE & Cloud
shivagamigugan@gmail.com

Agenda

A Practitioner's view

Observability : Why is it a Socio – Engineering – Technology Problem?

- **Technology Transformation Leader**, Aviation Technologist, **Head of Software Engineering** bootstrapped and built Software Architects and Engineers who deliver mission-critical software
- Led IT Digitisation, DevOps **and Head of Site Reliability engineering and Cloud** at Emirates Group IT
- DevOps Institute Ambassador and Middle East Chapter member

Shivagami Gugan

SKILup DAYS

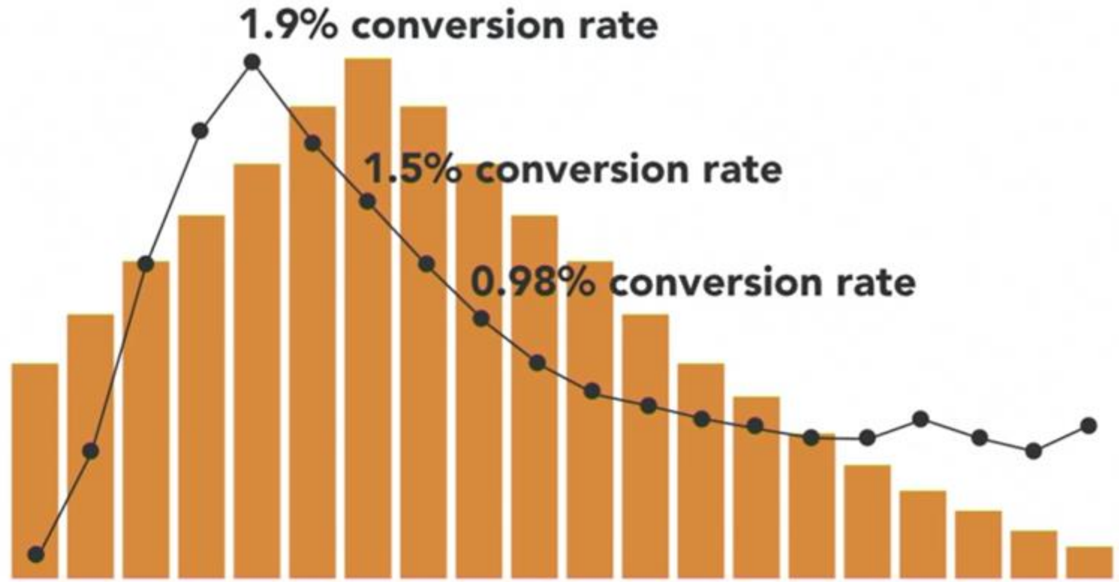
Key Takeaways

Why is Observability a BIGGER problem now ? What has changed?

Is Observability the missing link that will get you “the Zen of Performance” ?

Why is Observability such a Socio-Technology issue?

Performance Impacts the Business



1. Walmart found that for every 1 second improvement in page load time, conversions increased by 2%

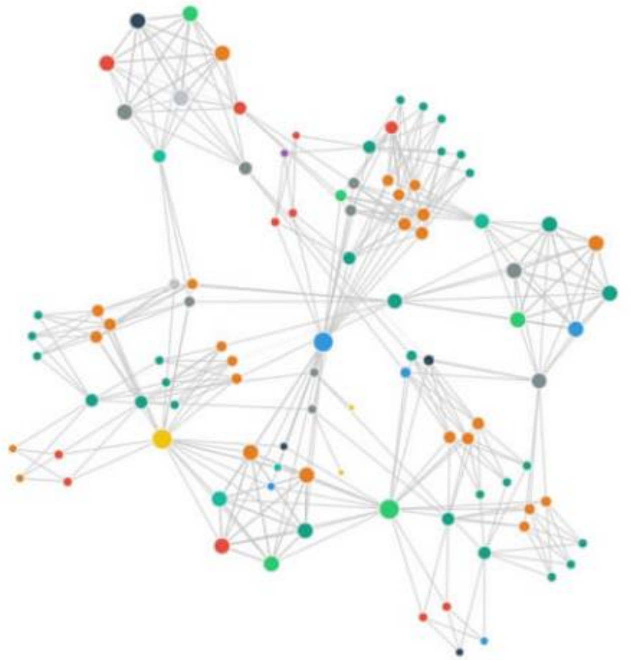
2. Mobify found that each 100ms improvement in their homepage's load time resulted in a 1.11% increase in conversion

SKILup DAYS

Performance in Complex Architectures

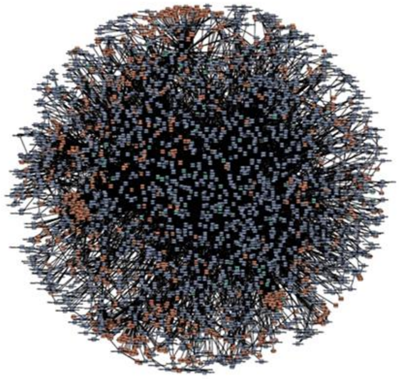
- Systems have become inherently very complex
- There is a whitespace in the area of “Integrated Visibility”

Distributedness



Monitor does not go away

- Business metrics
- Demand
- Workload
  - Fault/Errors
  - Availability
  - Performance
- Resource metrics


amazon.com

NETFLIX

Correctness, Speed and Consistency of a Hairball Architecture makes Monitoring OUTDATED for complex Systems

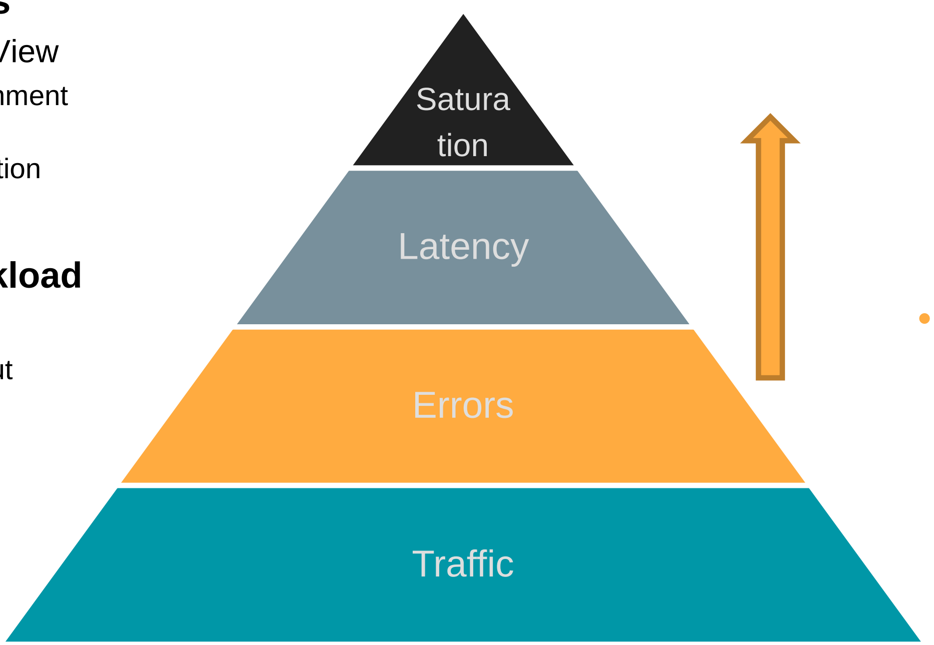
Logs, Events, Metrics and Tracing

Digital Business

- Business Metrics View
  - Checkout Abandonment
  - Customer Churn
  - Revenue per Location

Demand & Workload

- RED Metrics View
  - Request throughput
  - Errors
  - Duration (Latency, Response time)



Google's Golden Signals

Context

- Distributed Tracing
  - Dependency on downstream
  - Service Maps
  - End-to-End Transaction (hotspots, logic flaws)

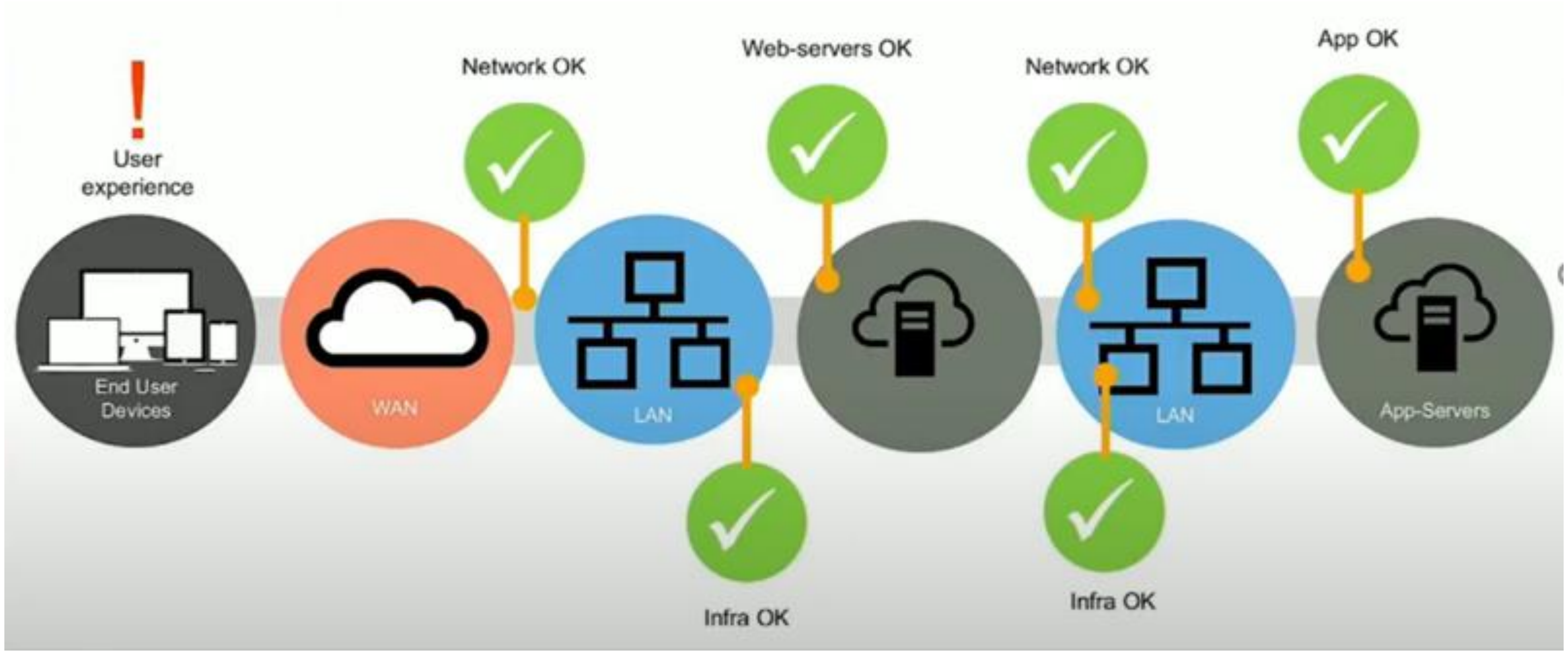
Resources

- USE Metrics View
  - Utilization
  - Saturation
  - Errors

As applications become more distributed, multiple dependencies, and ephemeral

BUILD BETTER INSIGHT INTO YOUR SYSTEM

Perspective bias



Law of requisite variety

“If a system is to be stable, the number of states of its control mechanism must be greater than or equal to the number of states in the system being controlled”

- W. Ross Ashby

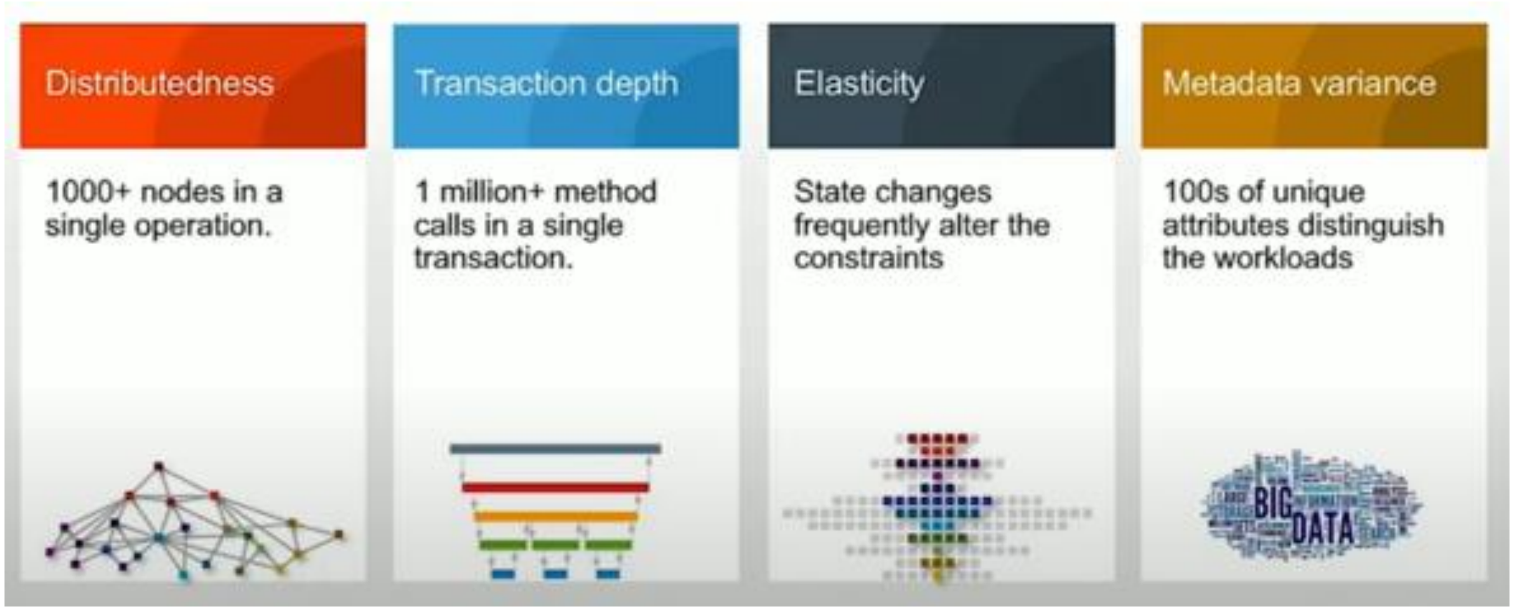
What are the Varieties?

***Version changes**: deployed upgrades of service versions*

***Topological changes**:* new components that appear and disappear in the system landscape and affect dependencies between existing running components.

***Component property changes**:* changing labels and tags of components

Observability of Complex Systems



Distributedness

1000+ nodes in a single operation.

Transaction depth

1 million+ method calls in a single transaction.

Elasticity

State changes frequently alter the constraints

Metadata variance

100s of unique attributes distinguish the workloads

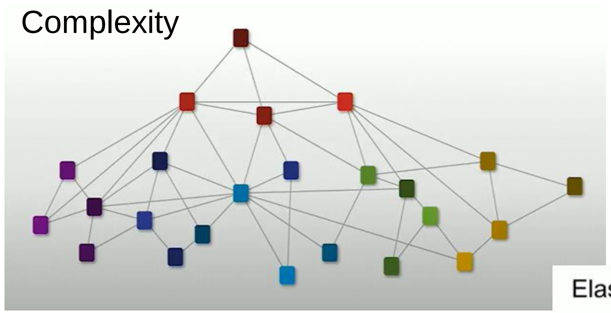
Cardinality and Dimensionality

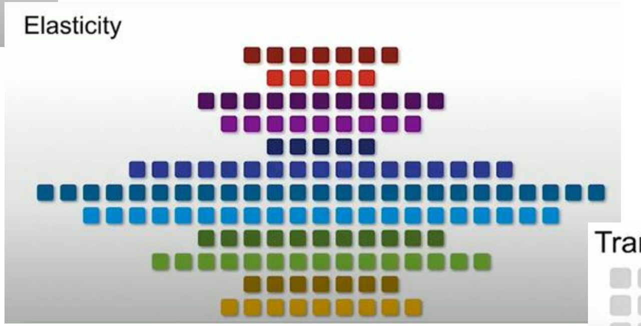
- System workload is many-dimensional data, not just one-dimensional values over time; and very high-cardinality.

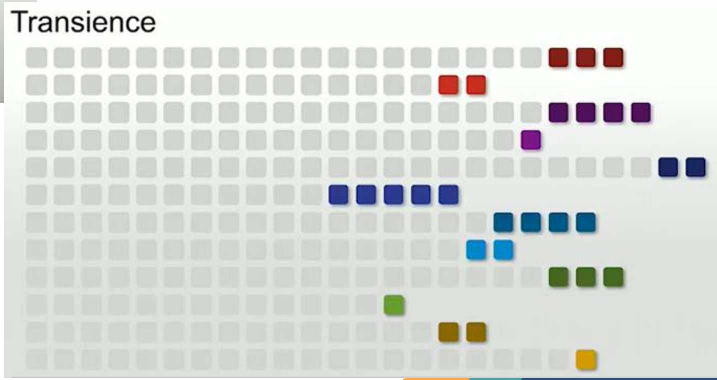
- Traditional time series databases were designed with a system-centric worldview and thus weren’t architected to store *or* query workload data. If Pre-aggregation happens before storing data, there is a fundamental problem.

- Using traditional tools to measure, inspect, and troubleshoot customers’ experiences is basically impossible because of pre-aggregation and cardinality limitations.

Practitioner’s view of Observability





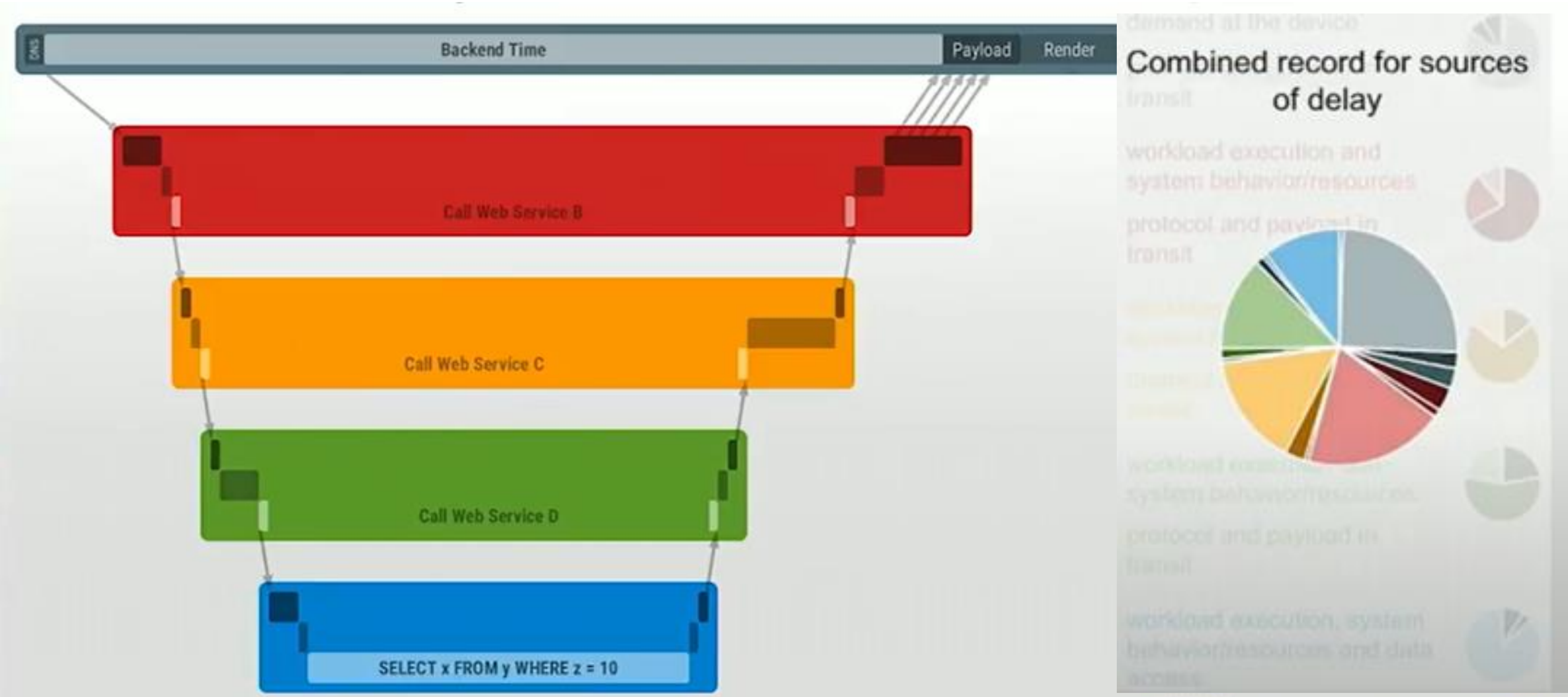


If you miss the State changes, you will not know which workload is being serviced by which resource.

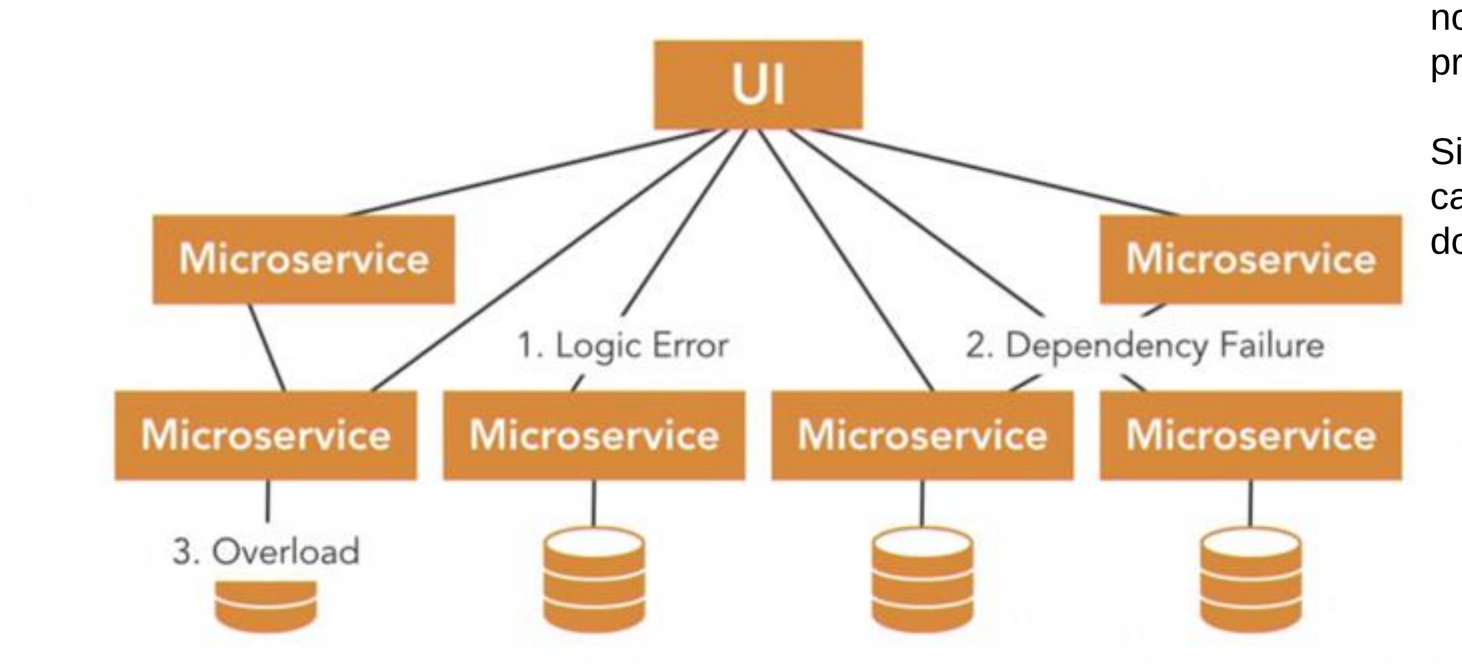
With Transience, with every spin up of resources, entity changes with every state change

Remember, Aggregation is the biggest enemy that will “kill” variety, making the information totally useless

Measure every element in the Request lifecycle



Distributed Tracing



Provides Context

Logs and Metrics will not show the real problem

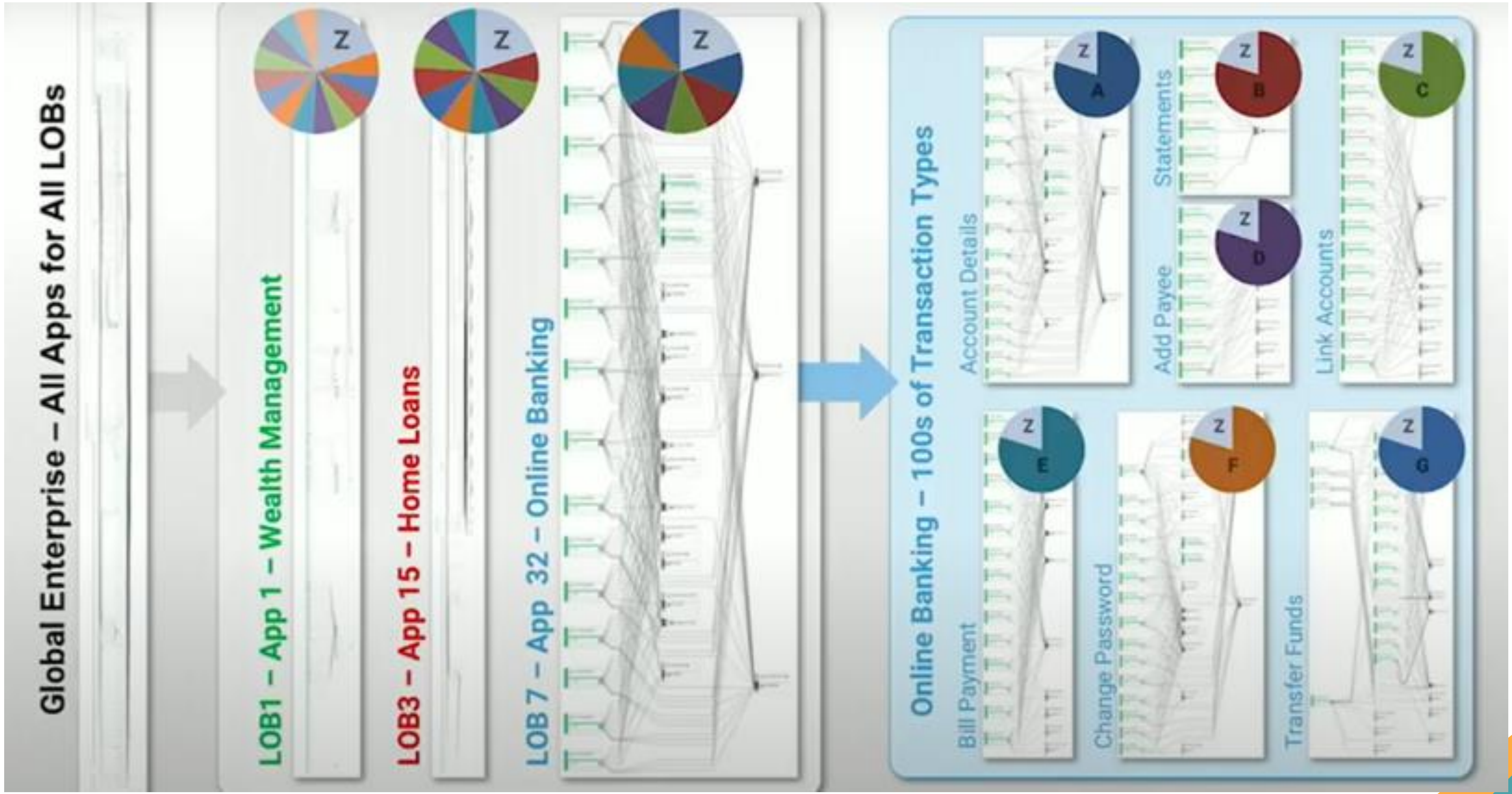
Single request may cause too many downstream requests

Observability-driven development

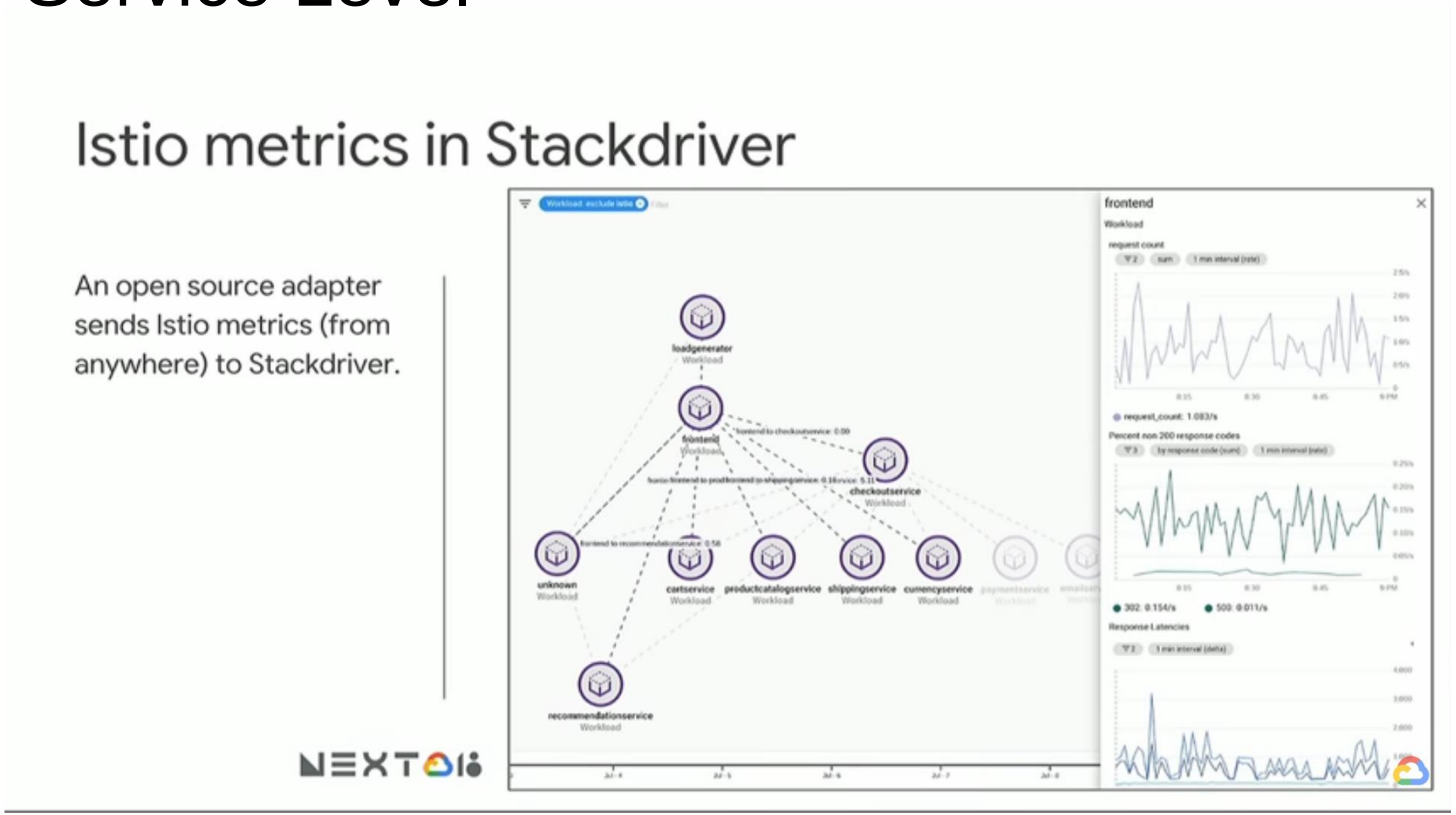
- Dev and Ops war will go only one way, the Dev way
- Give Developers the privilege to “ You Build, You Run, You Monitor”
- Merge will happen only when proper Observability hooks are baked in the code
- Never accept a PR until you learn the instrumentation
- Technology should enable distributed tracing, and tracing the breadcrumbs built in the system

- This is making DevOps fuller -> Each developer needs to own their code, with the ability to deploy it and debug it in production

The slowest constraint


Global Enterprise – All Apps for All LOBs
LOB1 – App 1 – Wealth Management
LOB3 – App 15 – Home Loans
LOB 7 – App 32 – Online Banking
Z
Online Banking – 100s of Transaction Types
Account Details
Statements
Add Payee
Link Accounts
Bill Payment
Change Password
Transfer Funds
A
B
C
D
E
F
G

Do it like an SRE: Observability has to be at the Service Level



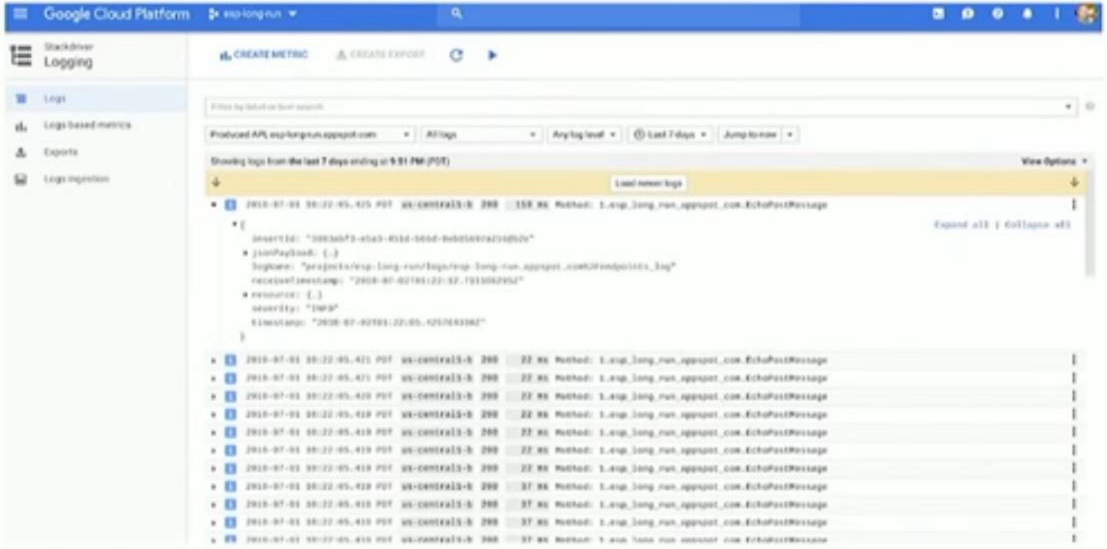
Istio metrics in Stackdriver

An open source adapter sends Istio metrics (from anywhere) to Stackdriver.

Logs + BigQuery = Ultimate power

Big Data/Big Analysis

Creating an export sink sends your logs to BigQuery, giving you the same power of Dremel that our SREs have.



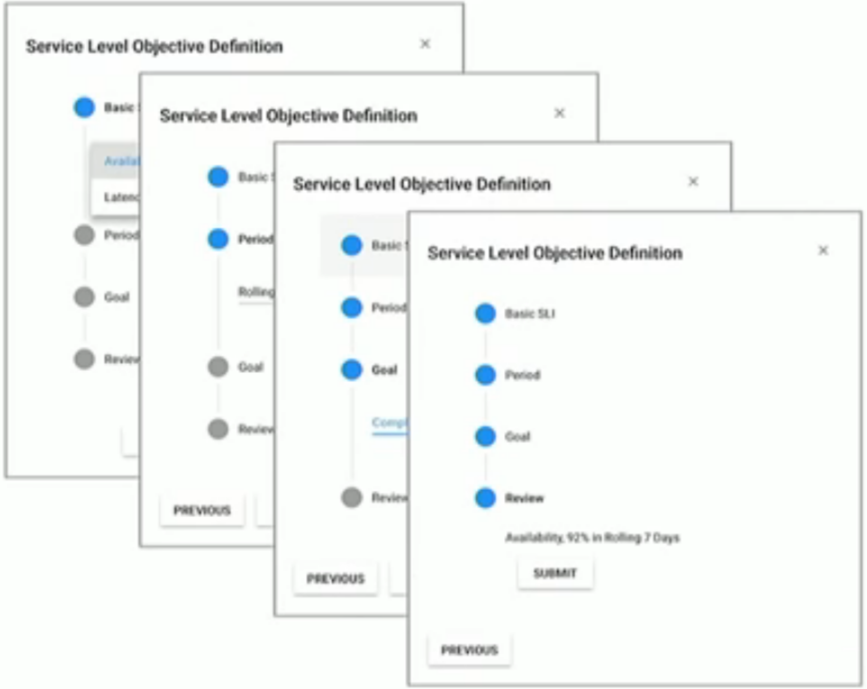
Showing logs from the last 7 days ending at 9:51 PM (PDT)

```
▾ 2018-07-01 18:22:05.425 PDT  us-central1-b  200   159 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
  ▾ {
     insertId: "[illegible]"
   ▸ jsonPayload: {…}
     logName: "projects/esp-long-run/logs/esp-long-run.appspot.com%2Fendpoints_log"
     receiveTimestamp: "2018-07-02T01:22:12.[illegible]Z"
   ▸ resource: {…}
     severity: "INFO"
     timestamp: "2018-07-02T01:22:05.[illegible]Z"
    }
▸ 2018-07-01 18:22:05.421 PDT  us-central1-b  200    22 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.421 PDT  us-central1-b  200    22 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.419 PDT  us-central1-b  200    22 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.418 PDT  us-central1-b  200    22 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.419 PDT  us-central1-b  200    22 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.419 PDT  us-central1-b  200    22 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.419 PDT  us-central1-b  200    22 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.418 PDT  us-central1-b  200    37 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.418 PDT  us-central1-b  200    37 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.418 PDT  us-central1-b  200    37 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
▸ 2018-07-01 18:22:05.418 PDT  us-central1-b  200    37 ms  Method: 1.esp_long_run_appspot_com.EchoPostMessage
```

SLO Monitoring

Monitor like an SRE

- Monitor customer-visible behavior
- Validate promises to user
- Error budget lets you balance velocity vs. reliability
- Alert only when promises are broken / on path to being broken



Service Level Objective Definition

Basic SLI

Period

Goal

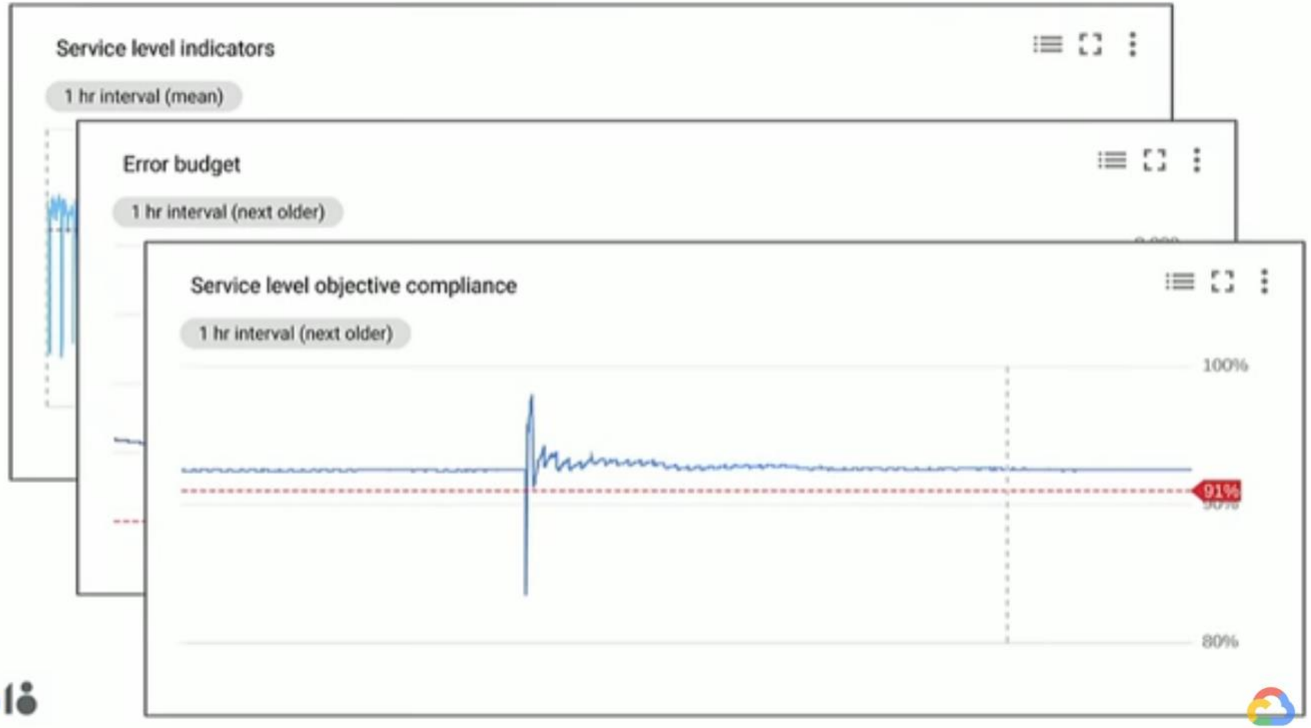
Review

Availability, 92% in Rolling 7 Days

SLO Monitoring

Monitor like an SRE

- Monitor customer-visible behavior
- Validate promises to user
- Error budget lets you balance velocity vs. reliability
- Alert only when promises are broken / on path to being broken


Service level indicators
1 hr interval (mean)
Error budget
1 hr interval (next older)
Service level objective compliance
1 hr interval (next older)
100%
91%
80%

It’s a Socio-Engineering-Technology problem

- Observability-driven-development (ODD)
- Incentivise the developer to capture everything
- Observability is the 1st step of the new world good coding practices
- SLI guided approach across multiple services
- Technology that will allow high Cardinality with little or no Aggregation
- Health checks, Logs, Metrics, Distributed, Request end to end tracing
- Not Manual/Toil – have a SRE approach
- ODD leads to true DevSecOps (for e.g. threat modelling)
- Leading to Autonomous AI

THANK YOU!

Meet Me in the Network Chat Lounge for Questions

Shivagami Gugan

SKILup DAYS