Seeing RED: Monitoring and Observability in the Age of Microservices

Observability SKILup Day

Greg Leffler

23 September 2021

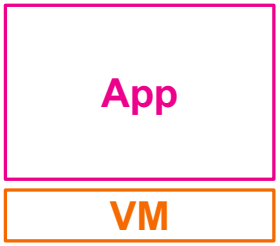
Monitoring

The times, they are a-changin'

TL;DR:
Monitoring Practices Matter

Applications are Changing

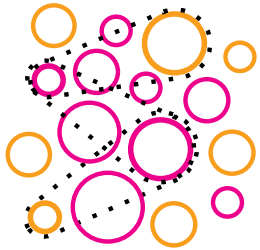
Legacy Monolithic Architecture



Monitored Environment

- Monolith App
- Single/Few hosts, On-Prem
- Single Language

New Microservices Architecture

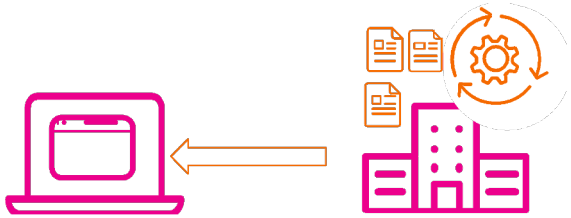


Monitored Environment

- Distributed Services (10s to 100s)
- Elastic environment scalable
- Frequent Code-Pushes (CI/CD)

Web Apps (Pages) are Also Changing

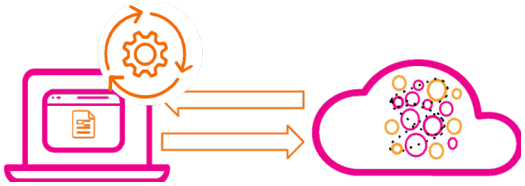
Multi-Page App



Mostly dependent on the backend

- Pages were rendered on the backend
- Page load were the automatic unit of measurement

Single Page App

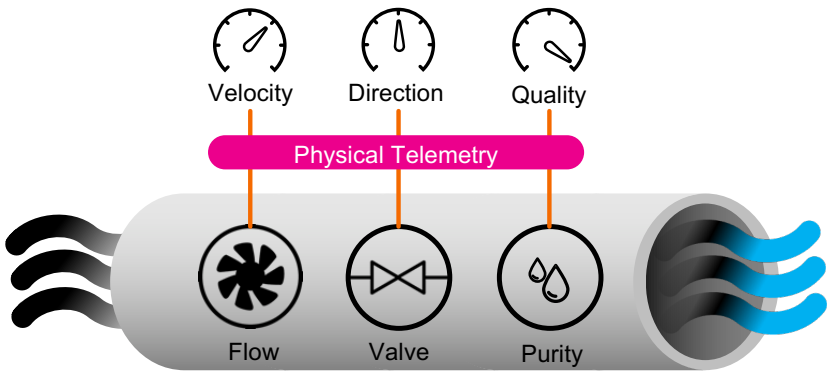


Front-end dependent

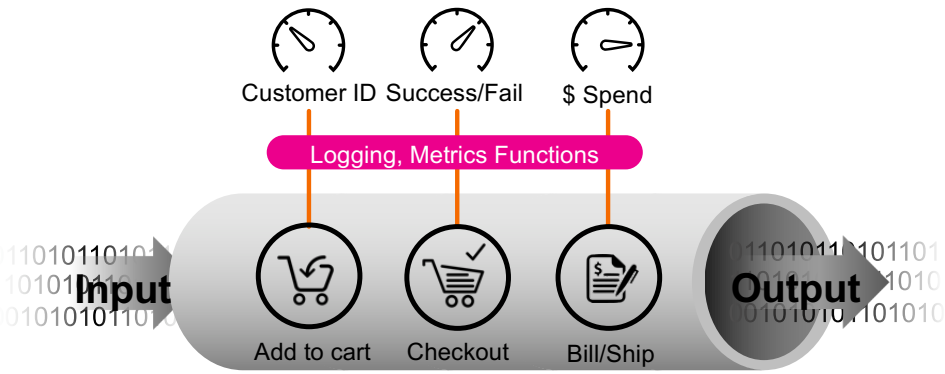
- Pages are rendered on the client side
- Multiple requests for additional data using XHR and API calls

No really. What is Observability?

Observability is the ability to measure the internal states of a system by examining its outputs. It's all about the data.



in Industrial Systems



in Software Systems

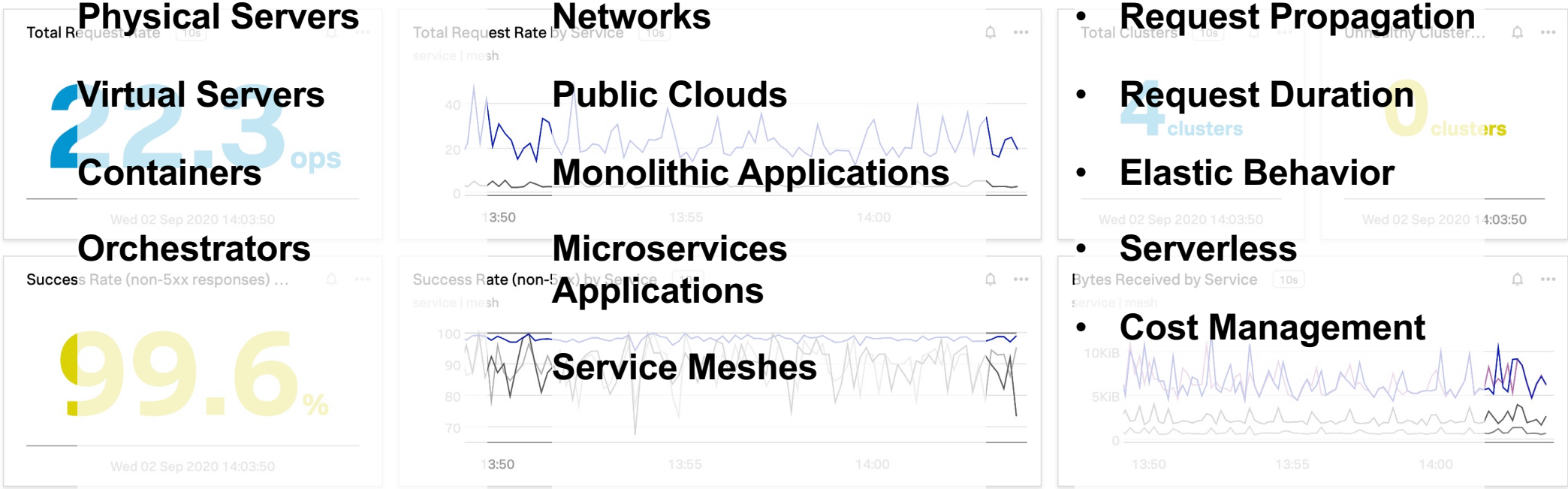
Data is the driving factor for Observability

But data is only useful if you can aggregate it, analyze and visualize it and respond to it.

Monitoring

Simple Name, Complex Problems

- Physical Servers
- Virtual Servers
- Containers
- Orchestrators
- Networks
- Public Clouds
- Monolithic Applications
- Microservices
- Applications
- Service Meshes
- Request Propagation
- Request Duration
- Elastic Behavior
- Serverless
- Cost Management



Thought Question:
How do you collect your data for monitoring today? Will it work for observability?

Challenges in Monitoring Microservices

Application and Infrastructure Monitoring

A microservices architecture will have 10s, 100s, 1000s, maybe even 10,000s of individual services:

- How do you know if an individual service is healthy?
- How do you measure the performance of an individual service?
- How do you troubleshoot and debug an individual service?

The Golden Signals

Google's Golden Signals

Latency, Saturation, Errors, Traffic

USE Monitoring

Utilization, Saturation, Errors

RED Monitoring

Rate, Errors, Duration

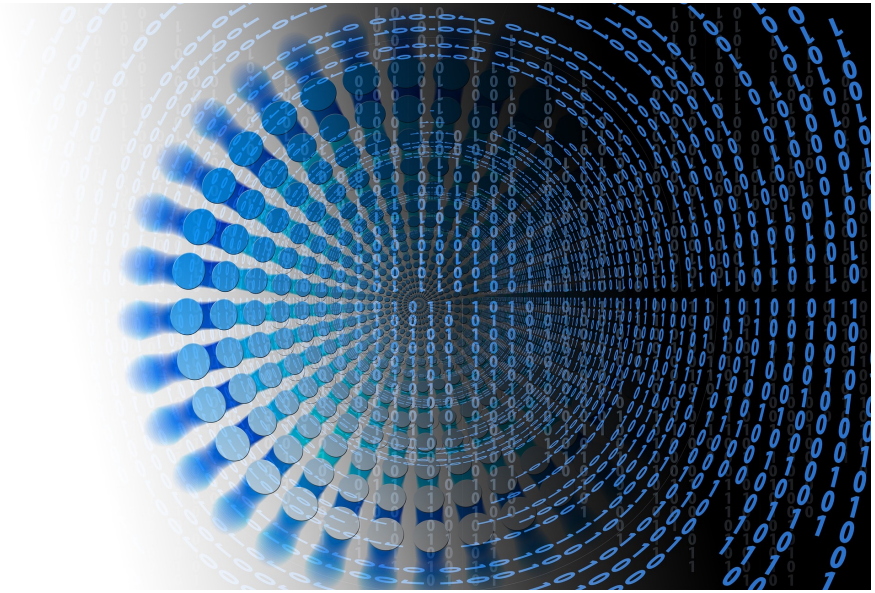
How about RED?

- A subset of Google's Golden Signals (SRE-related)
  H/T to Tom Wilkie
- Made up of rate, errors, duration
- Designed for request-driven systems, microservices

Service	Req/sec	Error Rate	P50 Duration	P90 Duration
api	9.9	51%	96ms	98ms
catalog	0.70	29%	74ms	75ms
checkout	9.3	8.5%	74ms	75ms
mangoDB	9.3	8.5%	32ms	50ms
payment	7.5	55%	50ms	51ms

Why RED

- Complexity matters
  - Lots of moving items
  - Lots of interrelations
  - Lots of “Not there now”
- We need simplicity and abstraction to resolve clutter
- We need to retain complexity for “Gotchas” and “A-ha’s”

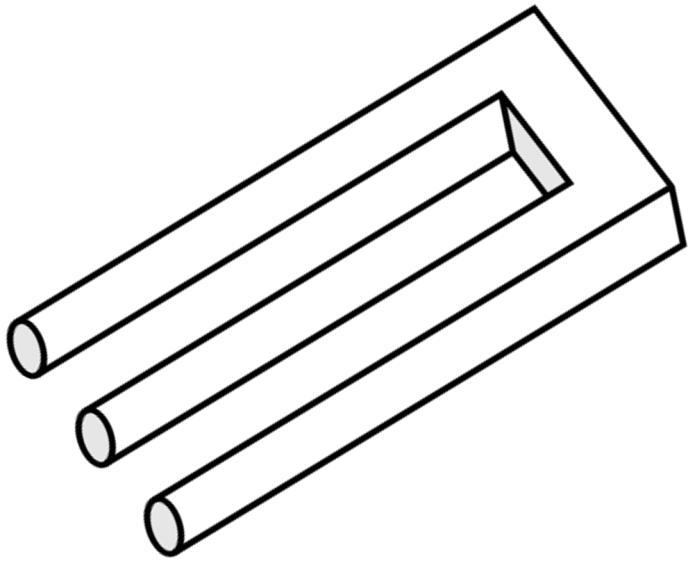


Rate

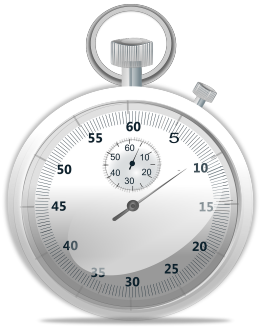
- Rate: number/size of requests on network and system
  - HTTP, SOAP, REST
  - Middleware messaging/queuing
  - API calls
  - Overhead of control structures like service meshes
- Any environment that can fail on peak traffic is a target for rate monitoring

Errors

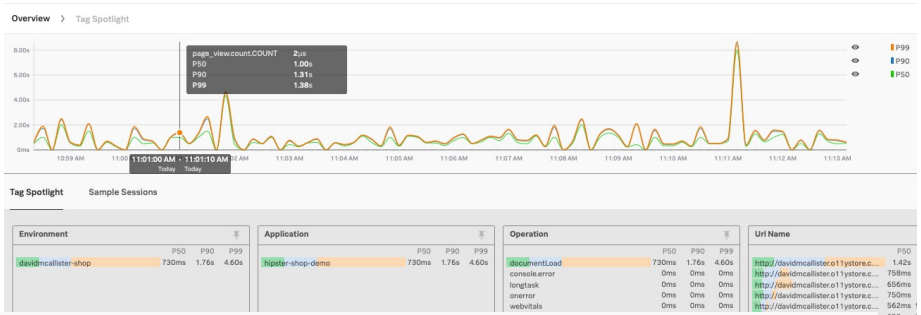
- Errors: problems that cause an incorrect, incomplete or unexpected result
  - Code failures
  - Production load bugs
  - Peak load bugs
  - Communication woes
- Errors need:
  - Rapid Responses
  - Point Specific responses
- Need deep dive, high-fidelity
- Need ASAP

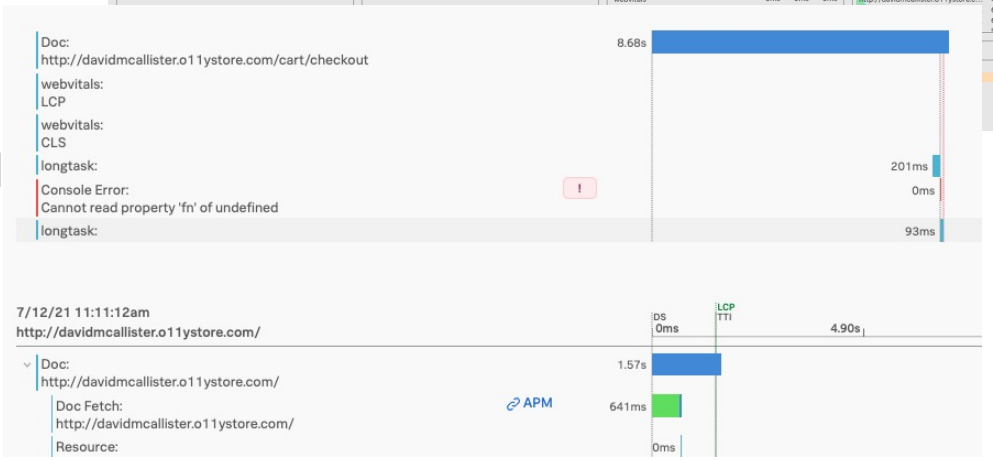


Duration



- It’s all about time
- Both client-side and server-sides are important
  - But client side maybe more
- Usually (now) the domain of distributed request tracing, RUM and APM
- **Bring events into causal order**

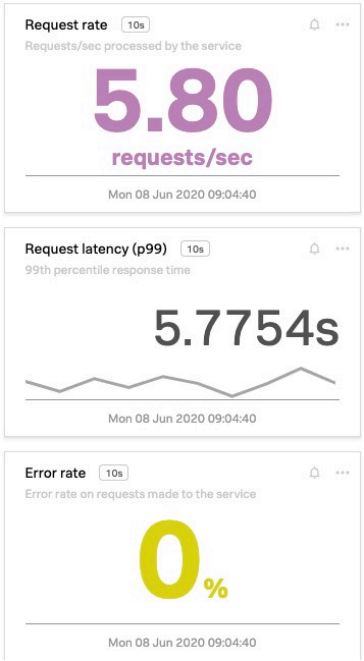




Why RED?

- Easy to remember
- Reduces decision fatigue
- Drives standardization and consistency
- Helps with automation
- Serves as a proxy for user happiness

A Customer Happiness Proxy



- External (customer's) view is singular
  - Request, and its latency and success
- Operator's view is over a workload
  - Requests latency, rates, and concurrency
  - System resources/ components

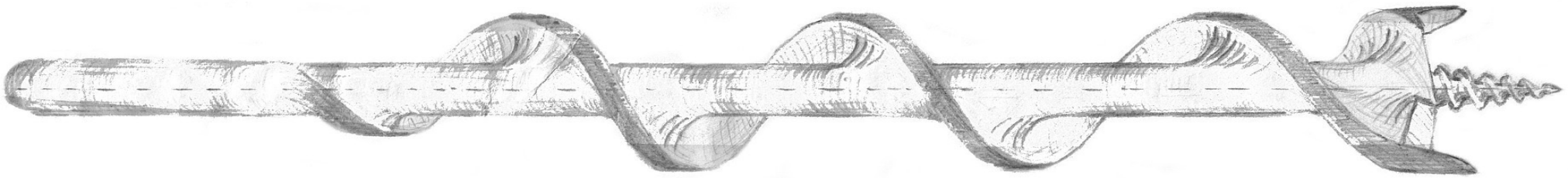
splunk> turn data into doing

Thought Question:
Do you need to track user happiness?

How will you approach this?

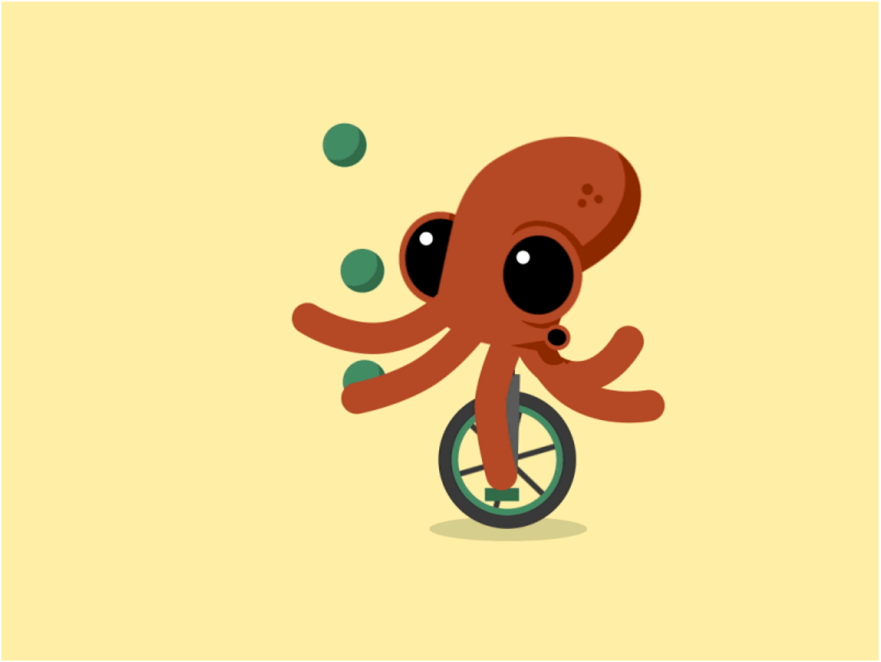
Monitoring Challenges

- It’s not Infrastructure OR applications - **It’s both**
- In complex environments tools should help you out
- If you can’t drill-down, don’t bother



Key Takeaways

- A common structure for monitoring allows clarity in separate teams
- Understand that you need to tailor focus to your needs
- Find the right tool to give you clarity and insight



Thank You!