Application Monitoring and Tracing in Kubernetes Avoiding Microservice Hell

David vonThenen
Cloud Native Engineer

Twitter: @dvonthenen
GitHub: dvonthenen
Blog: http://dvonthenen.com

vmware®

Open Source Summit Japan 2018
Thursday June 21, 2018

Agenda

Why do we care?

Introduction to Metrics

Introduction to Tracing

Demo

Q&A

Why do we care?

Microservices Are Awesome!

Discrete Set of Functionality

Resilient / Tolerates Failure

Distributed / Highly Scalable

Technology Freedom

Autonomy of Dev Teams

Enables Continuous Delivery

ANYONE CAN BE COOL
BUT AWESOME TAKES PRACTICE
more awesome pictures at THEMETAPICTURE.COM

Can Be Your Worst Nightmare!

Complex to Build

Decentralized Nature

Interface / Docs Required

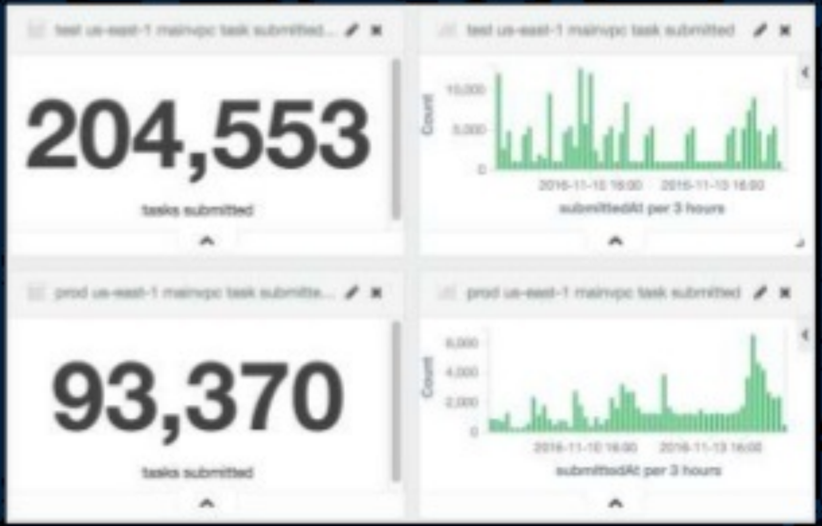
Operational Complexity

Transaction Management

Visibility is Difficult

Microservices at Scale

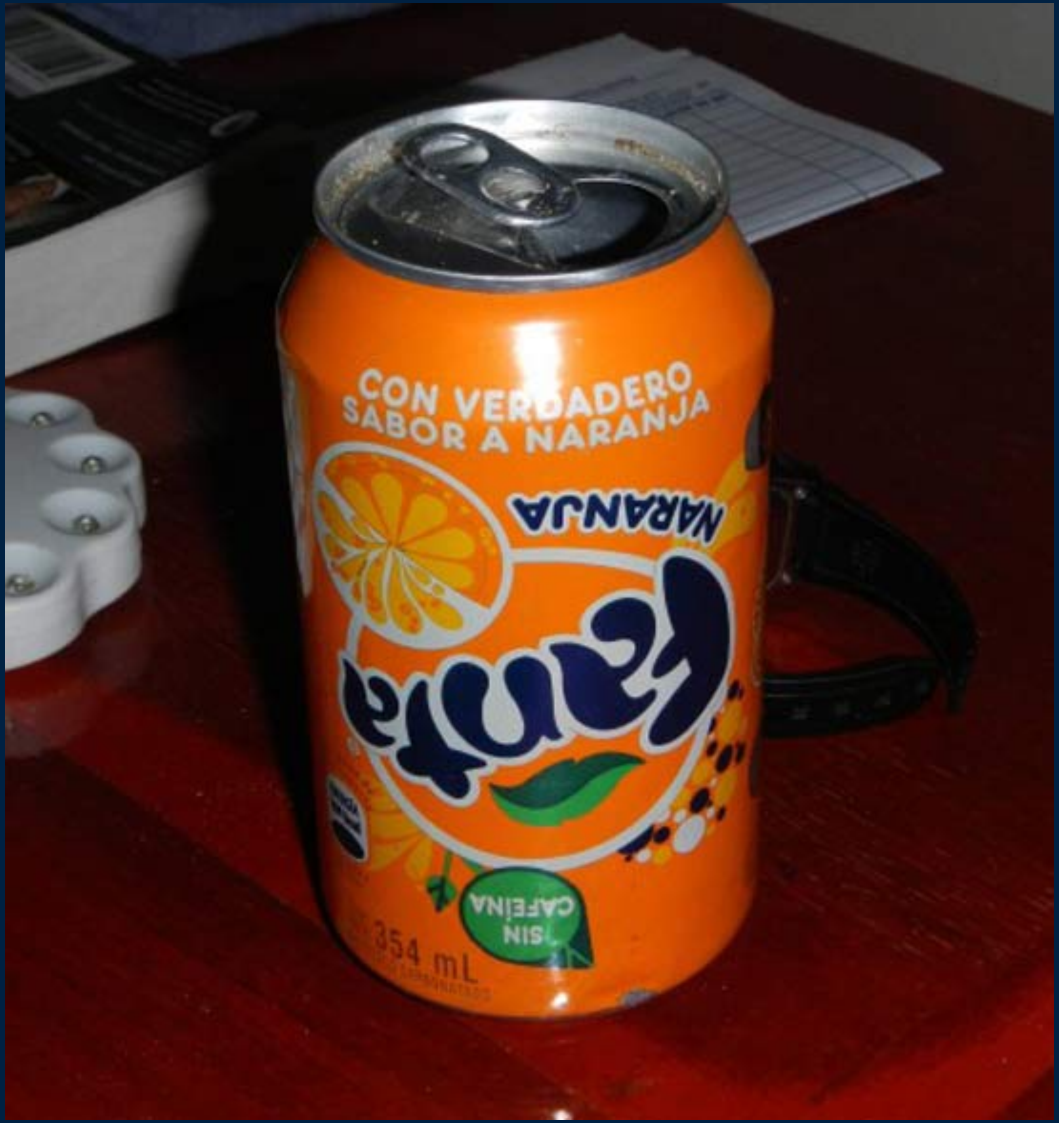
Titus Batch Usage (Week of 11/7)



- Started ~ 300,000 containers during the week
- Peak of 1000 containers per minute
- Peak of 3,000 instances (mix of r3.8xls and m4.4xls)

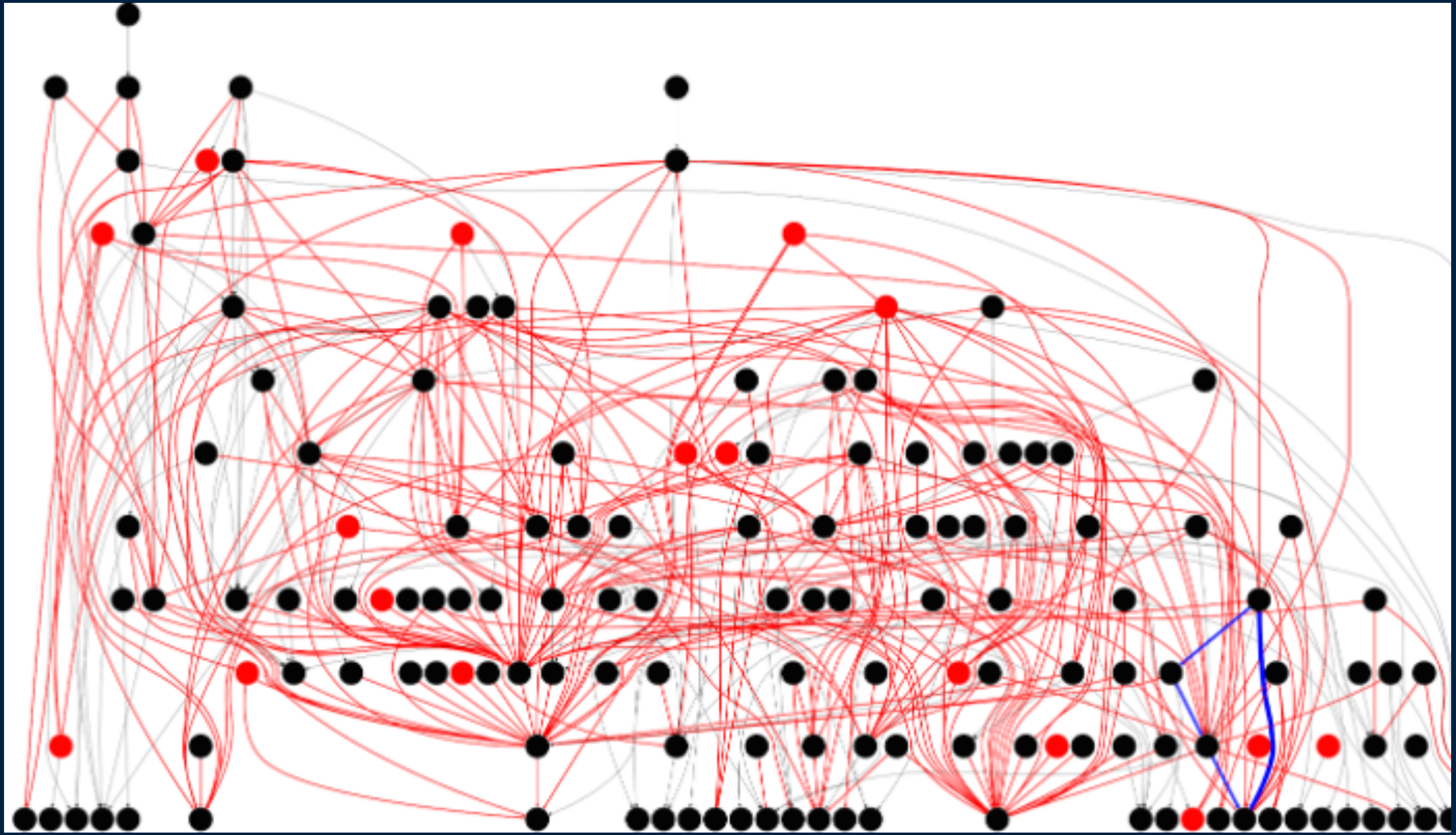
https://www.slideshare.net/aspyker/reinvent-2016-container-scheduling-execution-and-aws-integration

Simple Failures



Complex Failures

Who is Talking to Who?



One Bad Apple...

Logs Aren’t Enough

Gain Visibility Now!

The Answer is…

Metrics/Instrumentation

- Measure properties of a given system
- Alarms and Notifications

Tracing

- Observe interactions at a request level
- Measure work in time

Introduction to Metrics

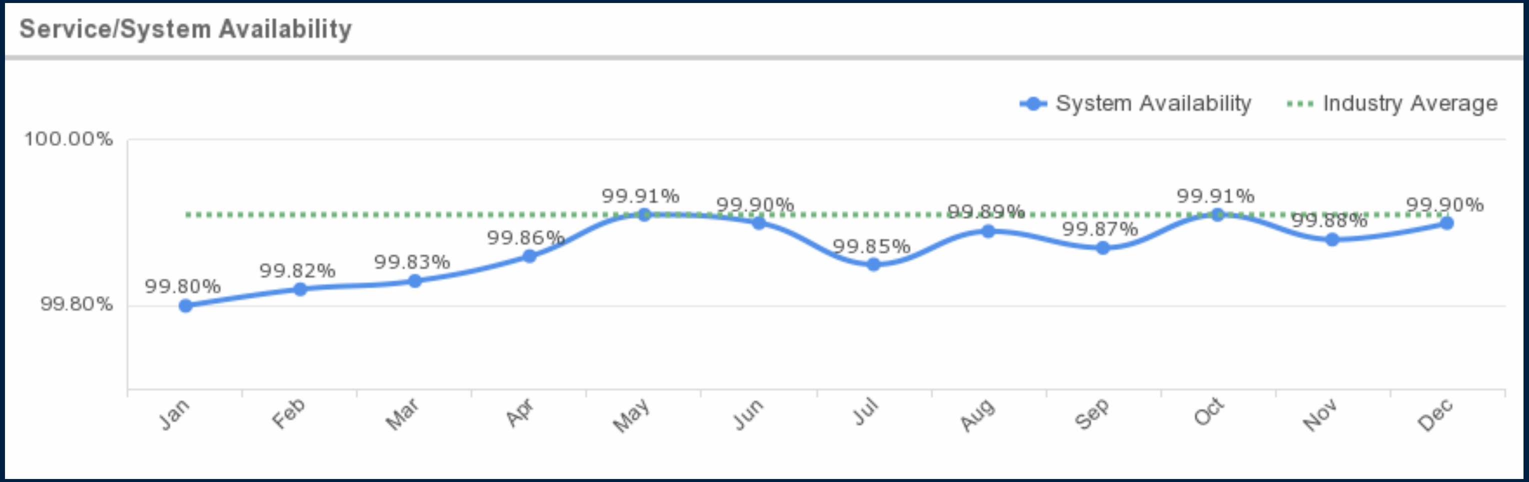
What are Metrics?

Metrics are a quantifiable set of measurements of a property for a given system, process, or component.

- Performance counters
- Instrumentation

Observe behavior

React to changes



Prometheus

Open-source systems monitoring and alerting project

Cloud Native Compute Foundation (CNCF) hosted project

Originally built by SoundCloud

Data model with time series data

https://github.com/prometheus/prometheus

Types of Measurments

Counter – only increases in value

Gauge – value goes up or down over time

Histogram – samples observations and counts them over buckets

Summary – histogram plus a summation of value

Alerts

Create rules based on observed metrics

Alerts trigger actions to be taken

- Email
- Slack
- Webhooks

Why do we care?

- Enables dynamic scale up and down

Prometheus Language Bindings

15 official and community supported libraries

- Go, Java, Python, Ruby, C++, etc

https://prometheus.io/docs/instrumenting/clientlibs/

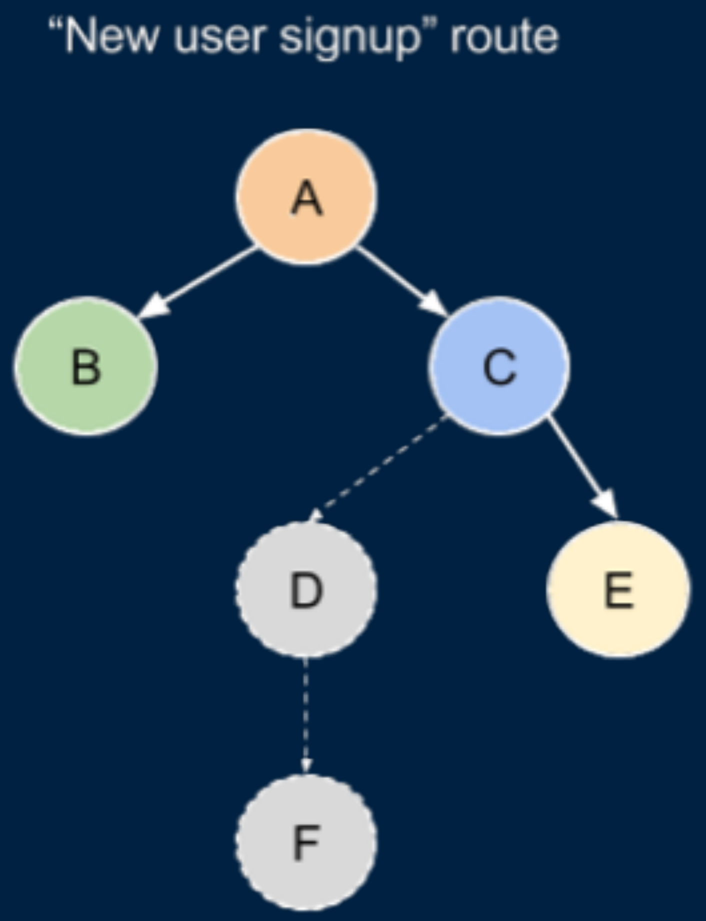
Introduction to Tracing

What is Tracing?

Enables observability of a given transaction as it moves through a (distributed) system

Allows visualization of which microservice instances are involved

Tracks the path through the software stack + time metrics



Jaeger

Open-source distributed tracing system

CNCF hosted project

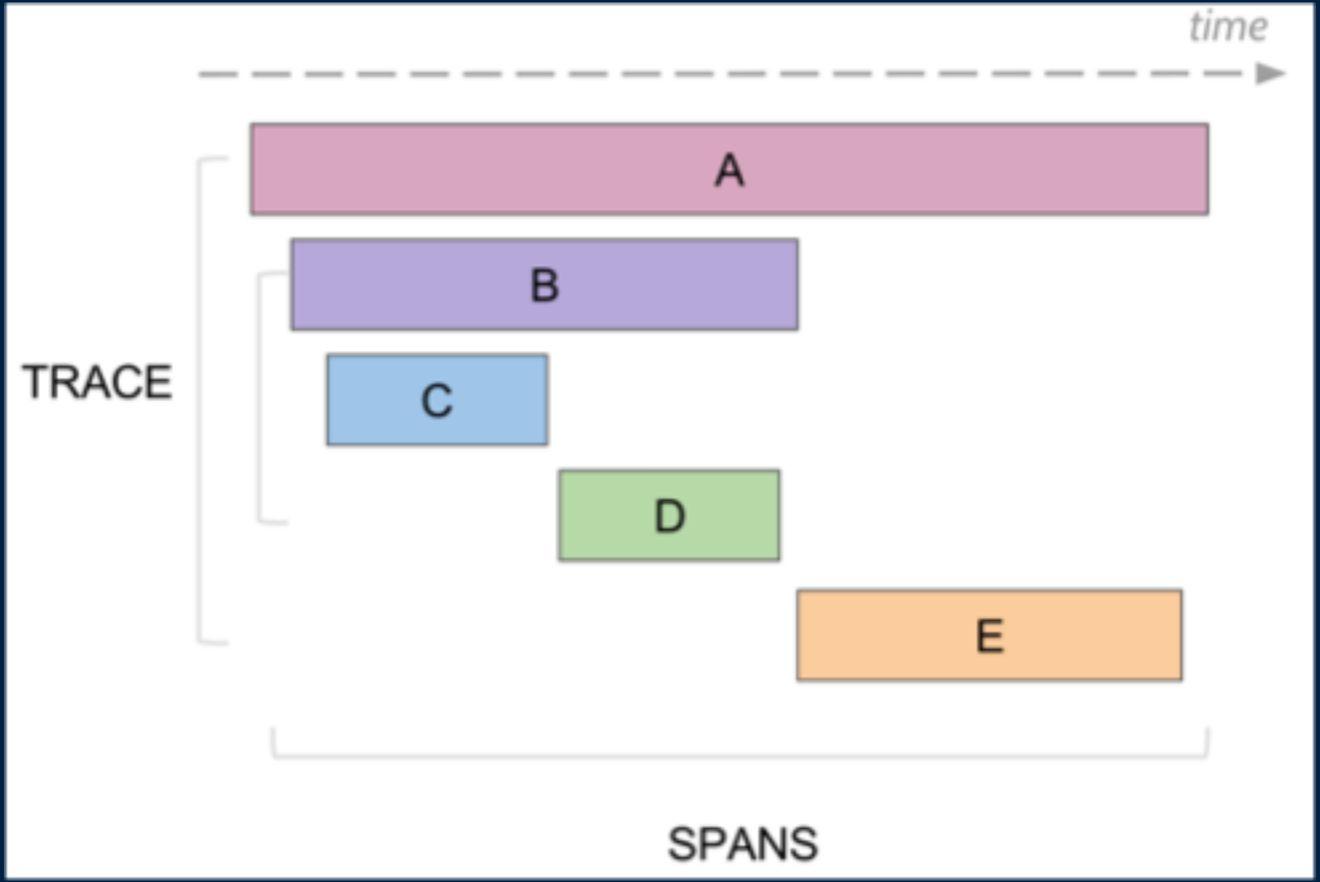
Originally built by Uber

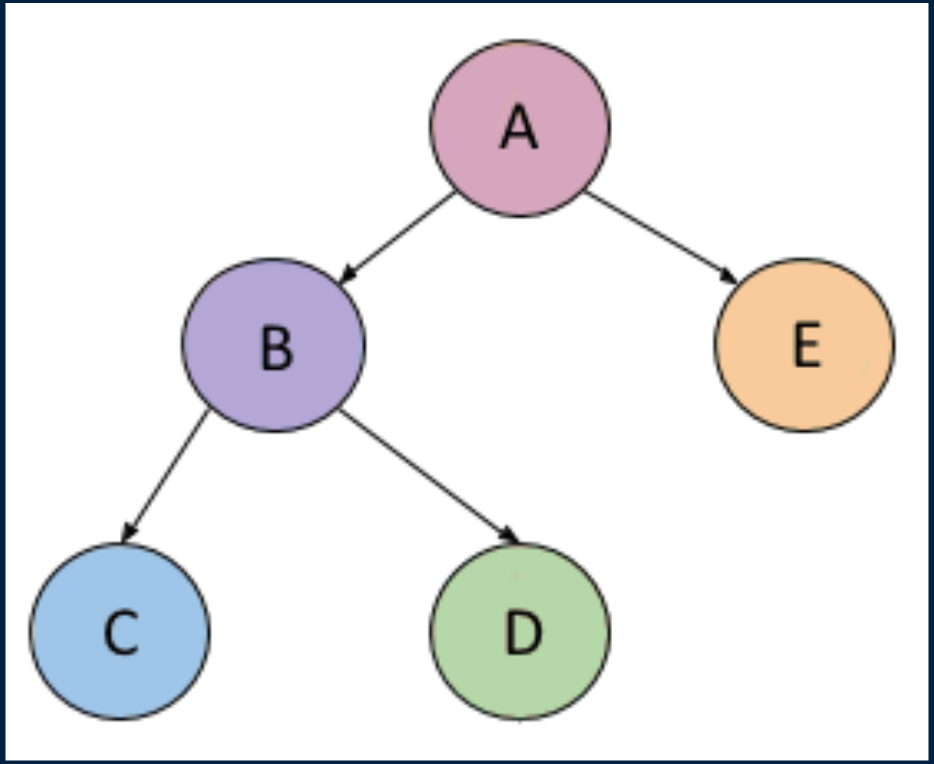
OpenTracing compatible

Root cause and observe performance

https://github.com/jaegertracing/jaeger

Traces and Spans





Jaeger Language Bindings

5 official and bunch of community supported libraries

- Go, Java, Python, node, C++
- http://jaeger.readthedocs.io/en/latest/client_libraries/

Metrics vs Tracing

Metrics

- Gives a singular per node, instance, or component view of the world
- Health checks, performance monitoring, etc
- Alerts and reaction to change

Tracing

- Follows a single transaction, API call, etc through a given system or application
- Think what a stack trace provides except tracing is doing it in a distributed fashion

Demo

Demo

Demo Configuration

Kubernetes 1.9

Prometheus 2.2

Jaeger 1.5

How-to:
https://github.com/dvonthenen/proposals/tree/master/2018_OSS_JAPAN

Thank You / Q&A

vmware®