

# **The road to SRE**

**@spanneberg @instanaHQ**



**What actually is this  
SRE thing ... ?**

# **What actually is this SRE thing ... ?**

***“SRE is what happens when you ask a  
software engineer to design an operations  
team”*** *Ben Treynor, VP Engineering Google*

***“Google’s approach to Service  
Management”*** *SRE book*

# **What actually is this SRE thing ... ?**

**SLI/O/As**

**Error  
Budgets**

**Blameless  
Postmortems**

**Capacity  
Planning**

**Being  
On-Call**

# **But what is it for the rest of us ... ?**

**SLI/O/As**

**Error  
Budgets**

**Blameless  
Postmortems**

**Capacity  
Planning**

**Being  
On-Call**

# **But what is it for the rest of us ... ?**

SLI/O/As

Automation

Eliminating  
Toil

System  
Engineering

Error  
Budgets

Blameless  
Postmortems

DB

Operations

Cost  
Planning

Releases

Developer  
Support

Capacity  
Planning

Networking

Software  
Engineering

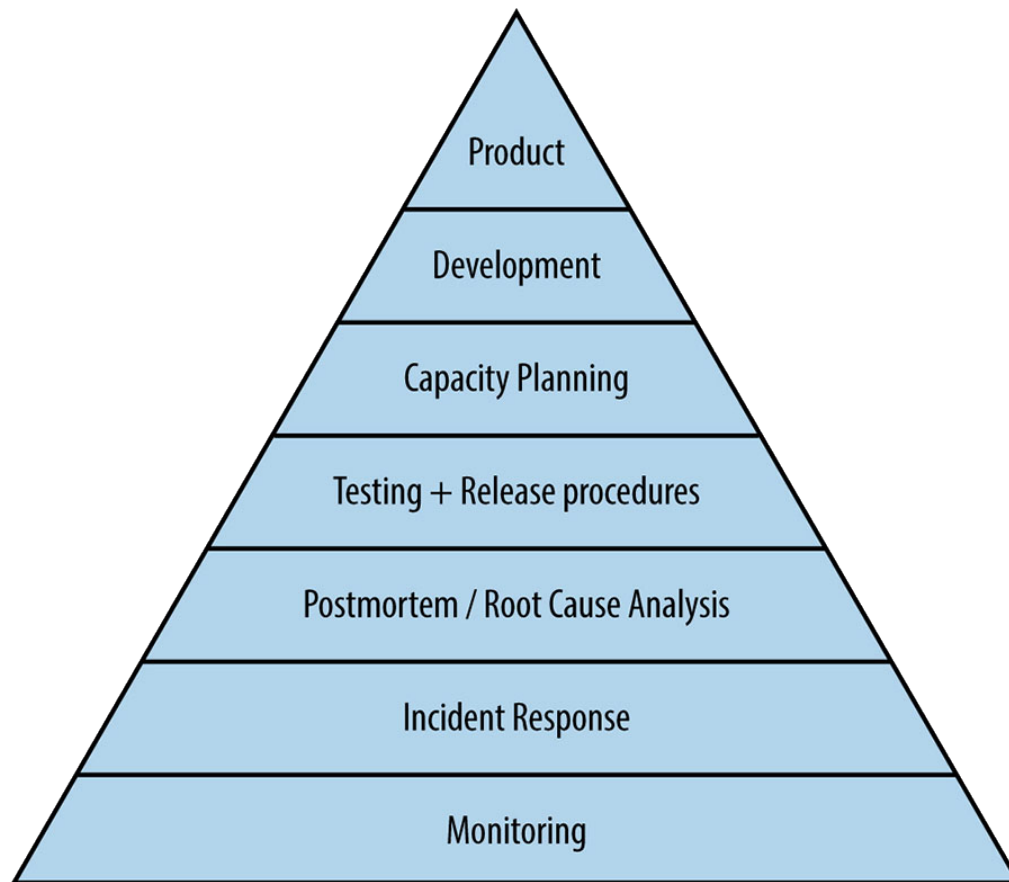
Internal  
Infrastructure

Being  
On-Call

Basic  
Operational  
Tasks

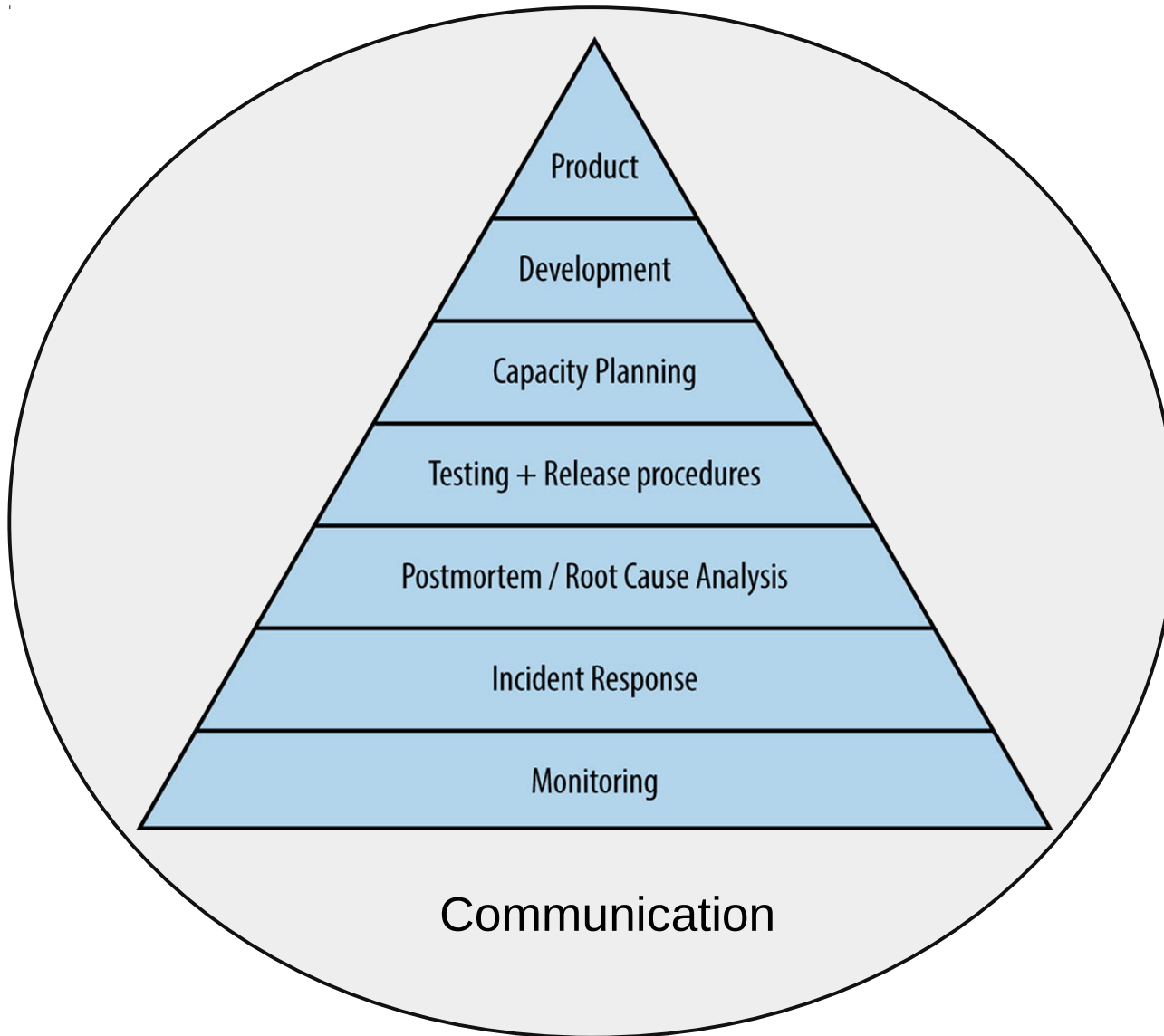
# The SRE Pyramid

## aka Dickersons Hierarchy of Reliability



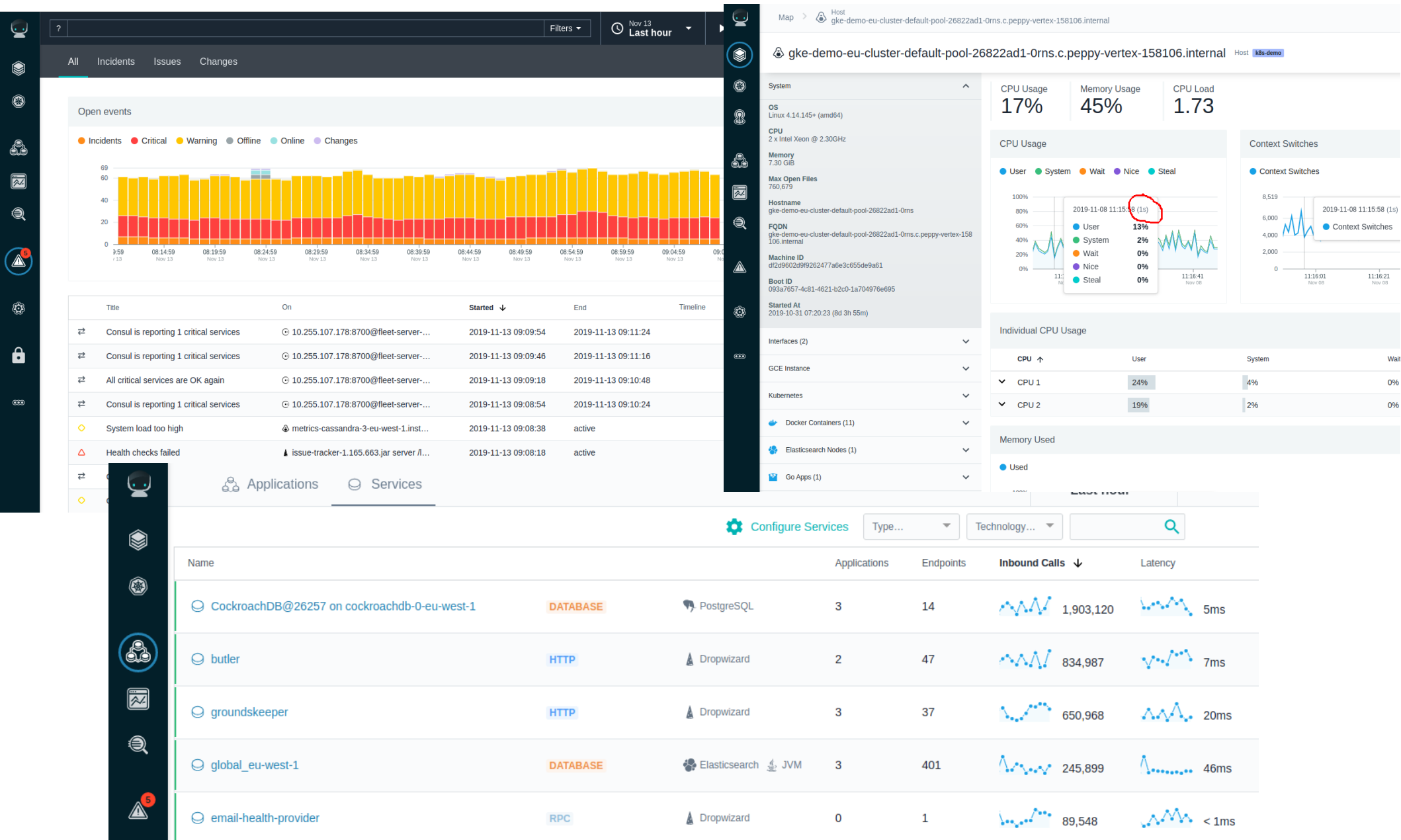
# The SRE Pyramid

## aka Dickersons Hierarchy of Reliability



# Setting the stage

## What is Instana?



# The early days

< 20 people. Mostly engineers + founders.  
Sales roles just starting.

Family + friends customers.

~~SRE~~ Ops/DevOps.

Everybody could touch anything.

Focus on product.

Failures did not matter that much (yet)

Solid but limited platform (Ansible, Docker, EC2).

Simple HC-based alerting

Founding  
2015

Joined  
Apr  
2016



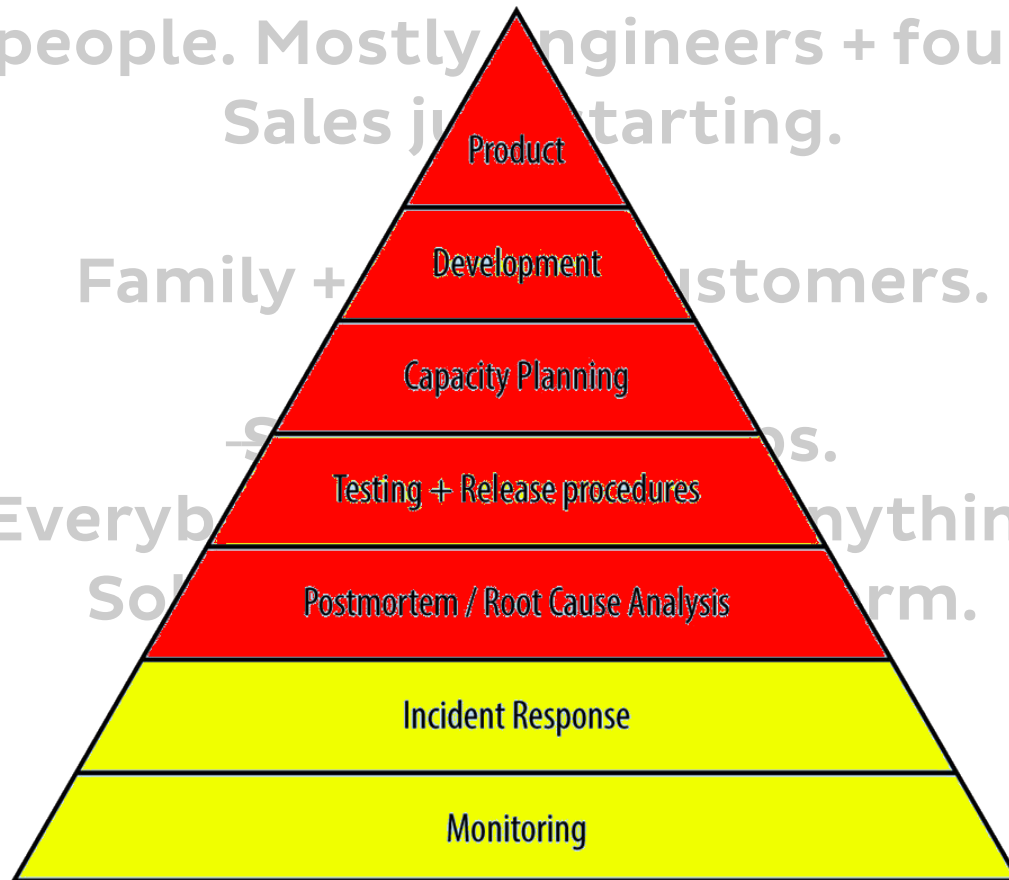
# The early days

< 20 people. Mostly engineers + founders.  
Sales just starting.

Family + friends + customers.

Everyone doing everything.  
Solving problems.

Failures did not matter that much (yet)



Founding  
2015

Joined  
Apr  
2016

# **Making things harder ...**

**Enter: On-Prem!**

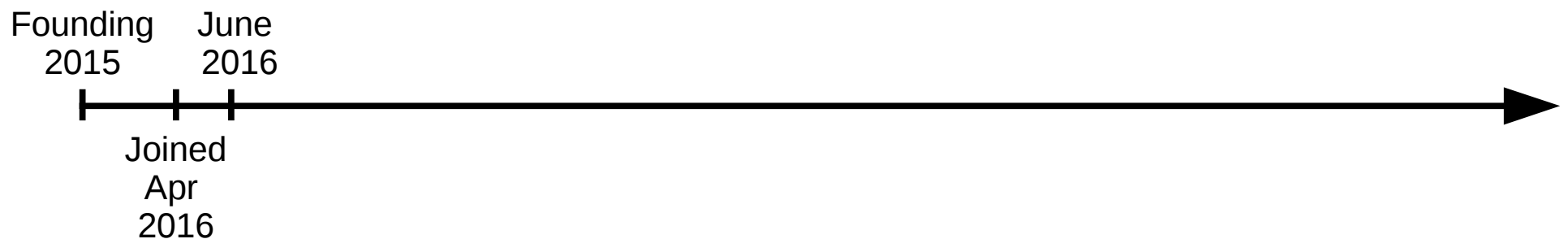
**Business need.**

**Ops/SRE team best fit.**

**Container-based approach w/ docker-compose.**

**Need to handle different release streams.**

**Customer support.**



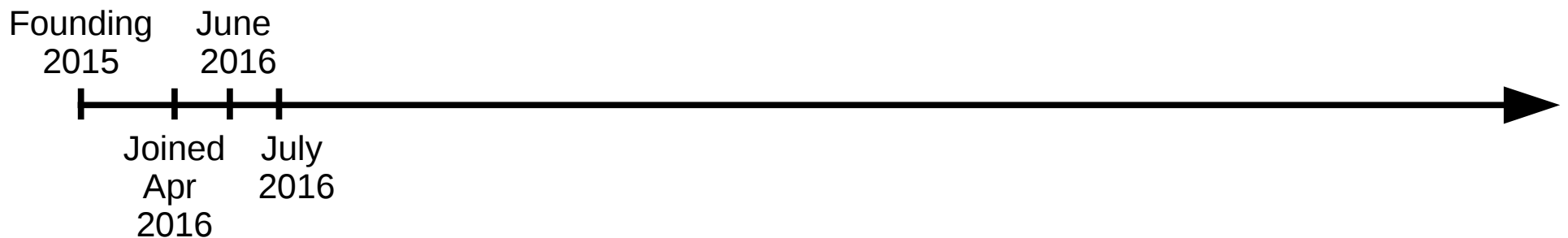
# **... and picking up traction**

**More customers.**

**Need for better on-call coverage.**

**First US colleague.**

**Prepare for scale.**

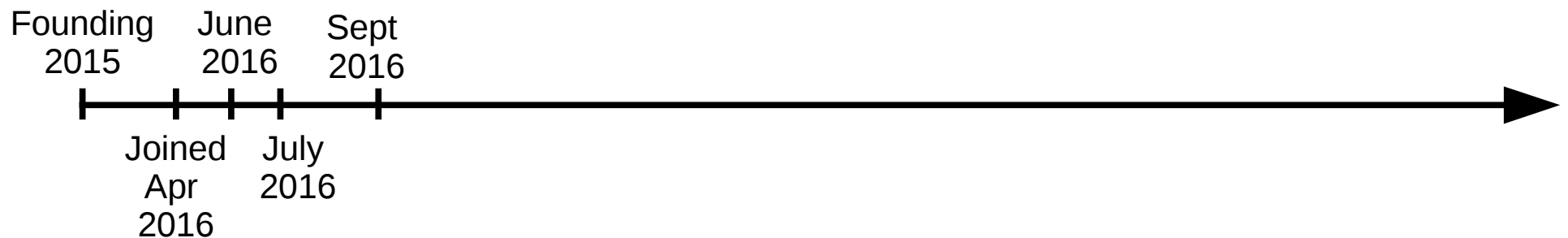


# Next steps

Platform migration → Consul/Nomad  
Proper failover. Multi-AZ.

Increase utilization. Lower cost.

More separation of concerns for the teams.



# Next steps

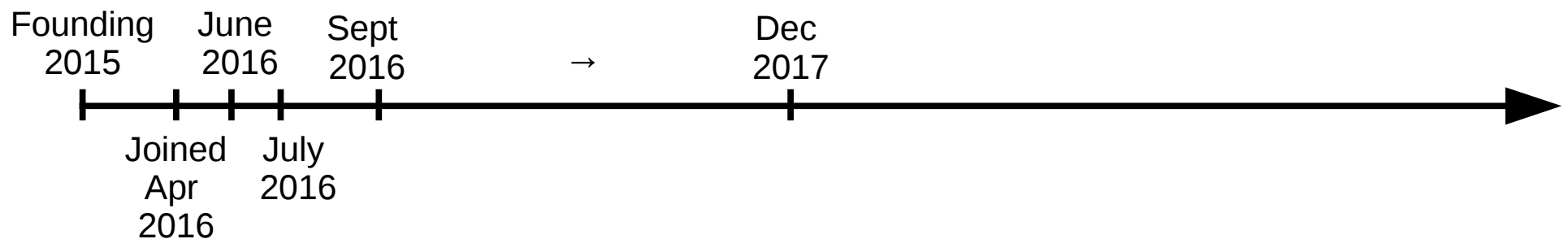
Platform migration → Consul/Nomad  
Proper failover. Multi-AZ.

Increase utilization. Lower cost.

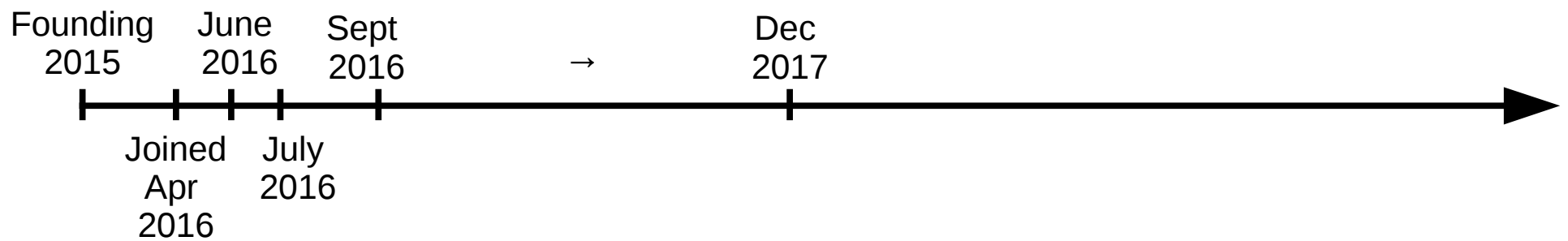
More separation of concerns for the teams.

Re-work on-prem (package-based).

Eliminate parts that did not work/scale.  
(Neo4J, Redis (Cluster), ...)



# Lessons learned?

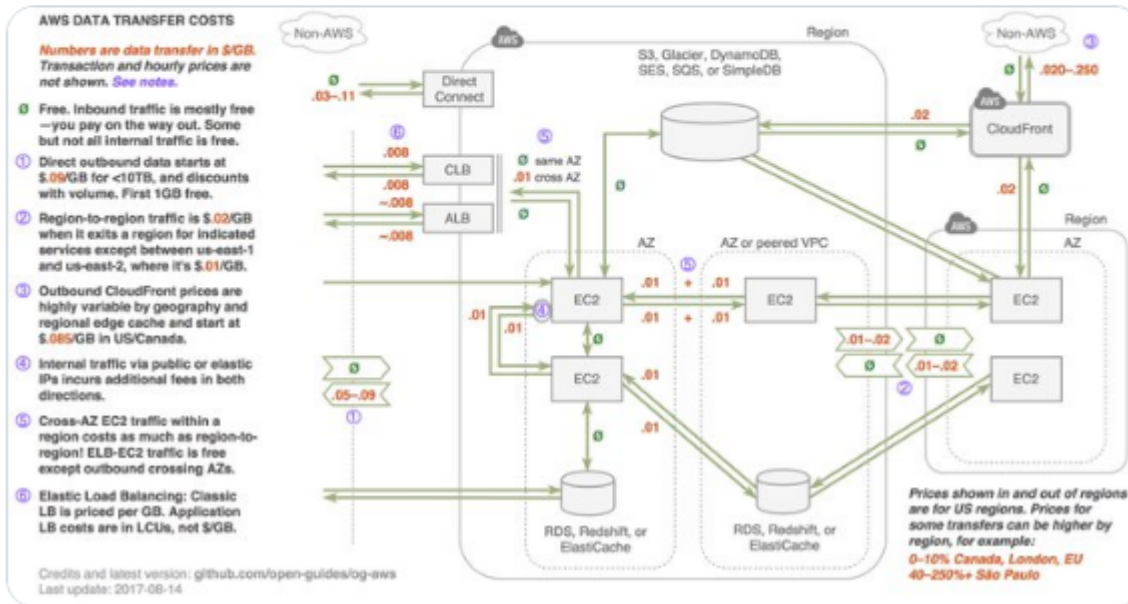


# Lessons learned?



Corey Quinn  
@QuinnyPig

I'm saying it a bit louder every time: [@awscloud](#)'s data transfer pricing is predatory garbage.



**Reliability costs money!**

**And effort.**

→ **Architecture changes.**

→ **Maintenance**

Founding 2015      June 2016      Sept 2016      →      Dec 2017

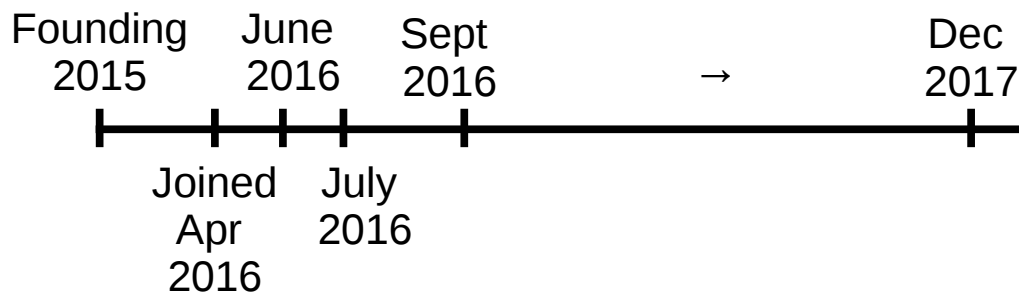
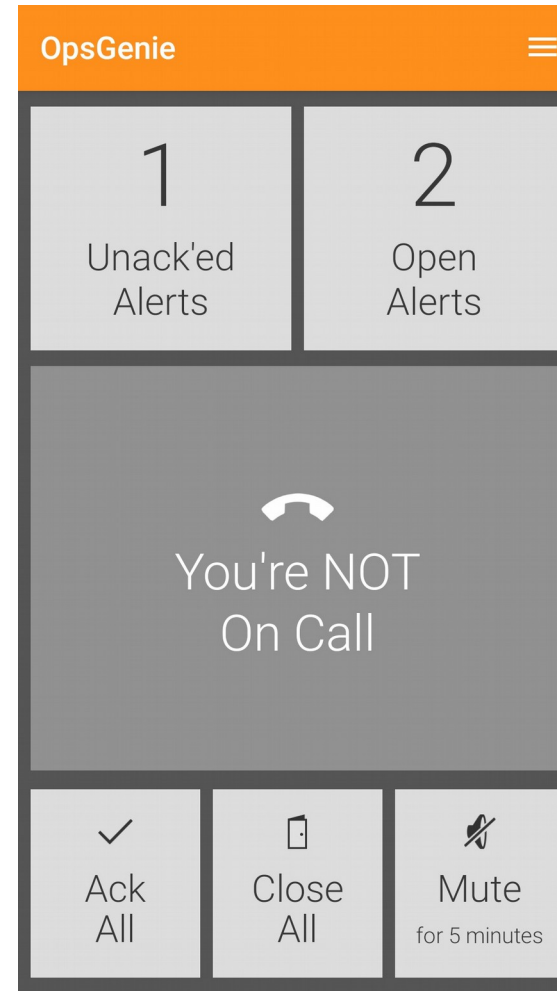
Joined Apr 2016      July 2016

# Lessons learned?

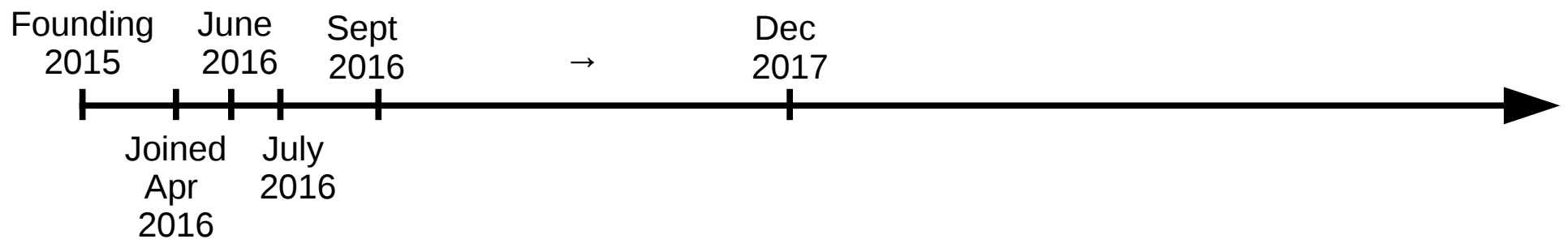
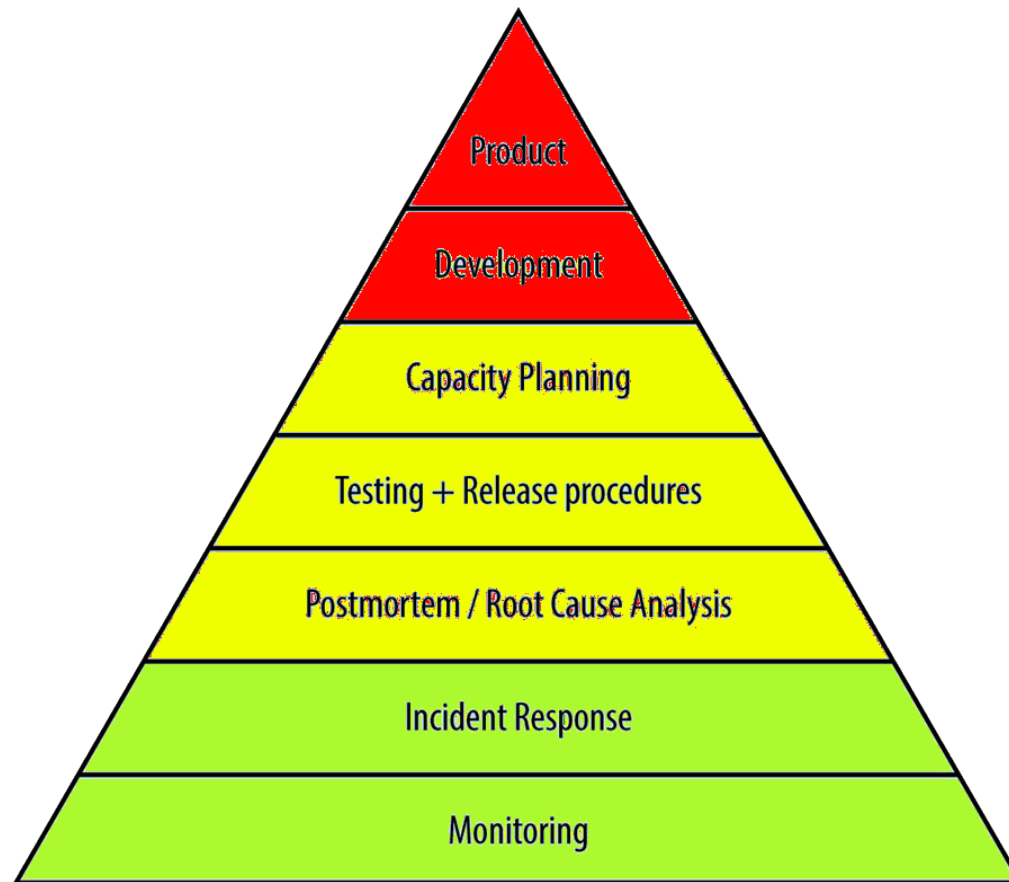
**Beware of unhealthy  
on-call schedules!**

**Implement rules to  
define who deals with  
what and when**

**Not everything is  
urgent**



# So where are we now?



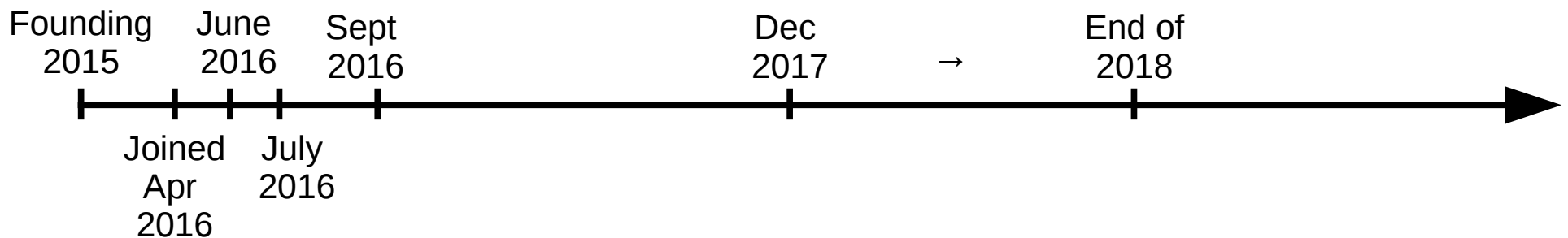
# More growth – more changes

A lot more non-engineers join.

Communication becoming more important.

→ Learn how to deal with and avoid panic ;)  
(re-visited Slack structure)

Provide RCAs to Customer Success to enable them to properly communicate with customers.



# More growth – more changes

△ [experimental SLO] Kafka consumer lag larger than one minute

×

Started  
2019-11-12  
12:55:30

Ended  
2019-11-12  
12:57:01

Duration  
1m

## Description

On:  processor-1.165.646.jar server /local/config.yaml

### Detail

At least one kafka consumer in this component has accumulated a lag greater than one minute.

To find out which consumer (resp. which topic) is lagging, check the `kafka-consumer.<topic>.time-lag-ms` metrics.

Normally all our components start dropping when the lag is getting too high. So a high lag can be an indicator of the following problems:

- a bug, e.g. missing backpressure handling - check the code and other dropwizard metrics to find out if we start dropping on high lag.
- CPU usage on the host too high (consumer thread is not getting enough CPU time) - check cpu metrics on the host and reschedule components to different hosts.
- network overloaded - check network metrics on the host and rescheduled components to different hosts.
- kafka broker overloaded - check metrics on the kafka broker (cpu, network, idle time) and rebalance topics or scale out kafka broker.

Rule: `metrics.gauges.kafka-consumer.max-time-lag-ms > 60000` (avg over 60000ms)

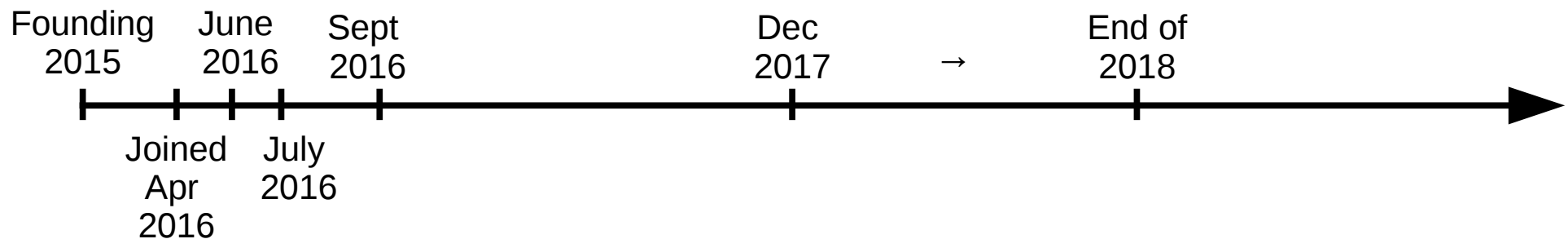
[View Custom Event](#)

## Metrics

● `metrics.gauges.kafka-consumer.max-time-lag-ms`

More work with  
SLOs to ensure  
platform QoS.

Constantly  
re-visiting  
usefulness  
of SLOs



# More growth – more changes

△ [experimental SLO] Kafka consumer lag larger than one minute

Started  
2019-11-12  
12:55:30

Ended  
2019-11-12  
12:57:01

## Description

On:  processor-1.165.646.jar server /local/config.yaml

## Detail

At least one kafka consumer in this component has accumulated a lag greater than one minute.

To find out which consumer (resp. which topic) is lagging, check the `kafka-consumer.<topic>.time-lag-ms`

Normally all our components start dropping when the lag is getting too high. So a high lag can be an indicator of

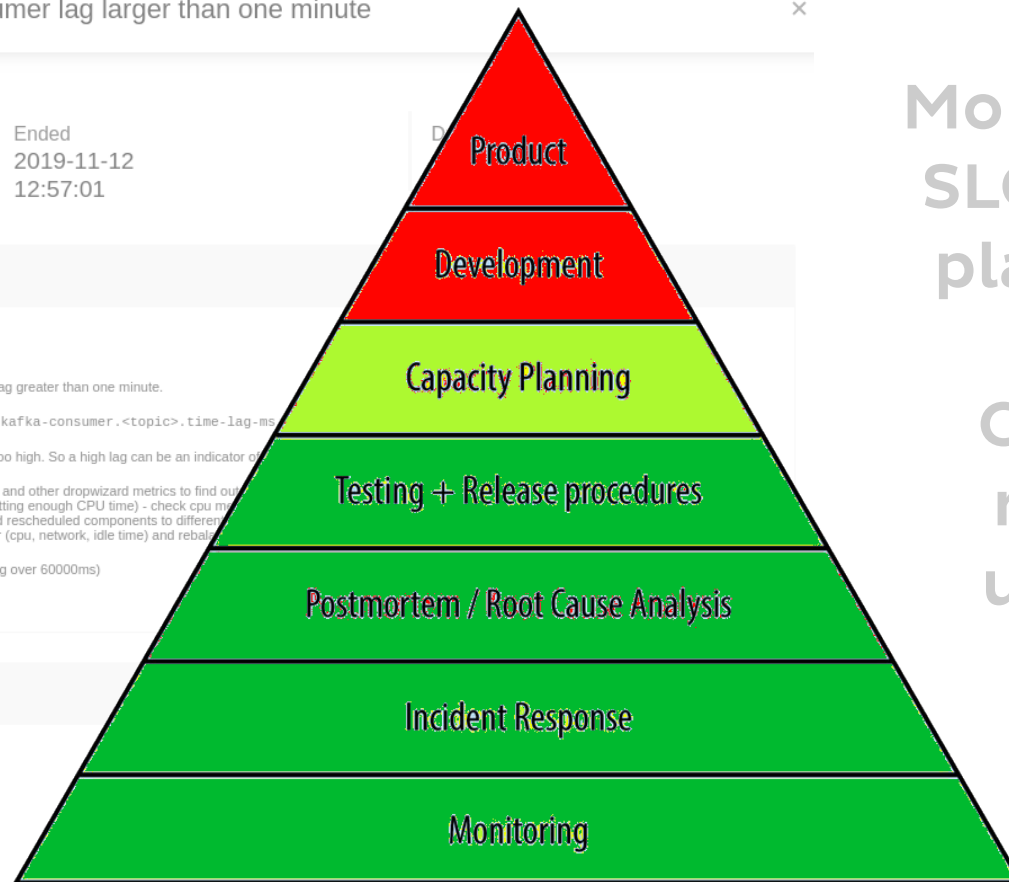
- a bug, e.g. missing backpressure handling - check the code and other dropwizard metrics to find out
- CPU usage on the host too high (consumer thread is not getting enough CPU time) - check `cpu` metrics
- network overloaded - check network metrics on the host and rescheduled components to different hosts
- kafka broker overloaded - check metrics on the kafka broker (`cpu`, `network`, `idle` time) and rebalance

Rule: `metrics.gauges.kafka-consumer.max-time-lag-ms > 60000` (avg over 60000ms)

[View Custom Event](#)

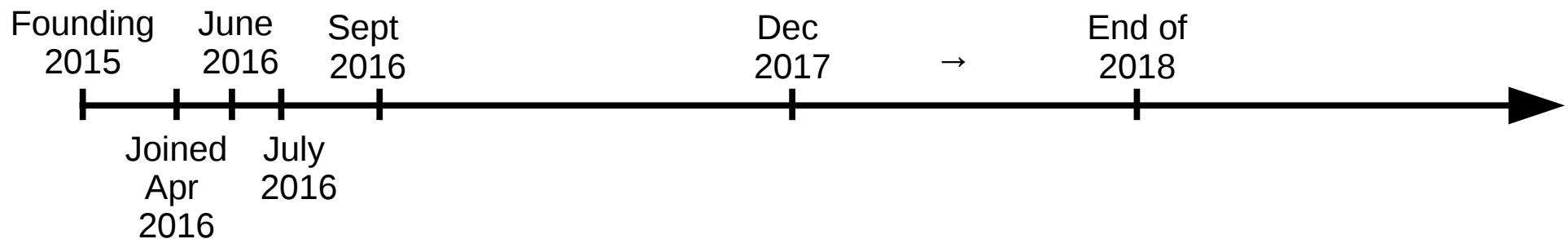
## Metrics

● `metrics.gauges.kafka-consumer.max-time-lag-ms`



More work with  
SLOs to ensure  
platform QoS.

Constantly  
re-visiting  
usefulness  
of SLOs



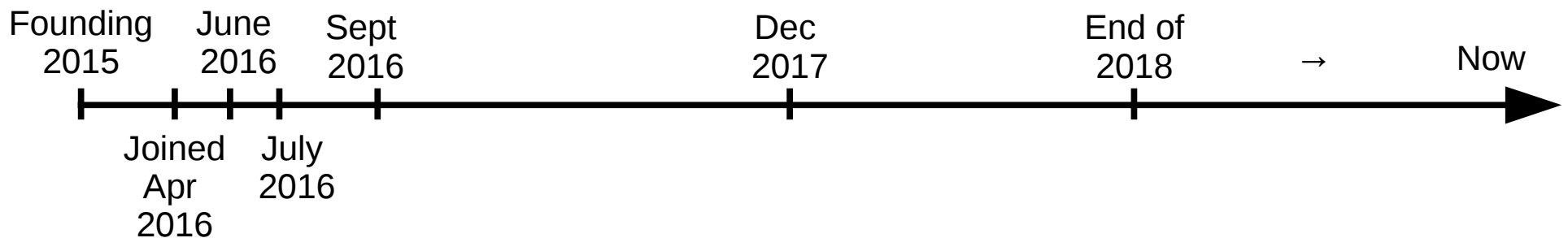
# What's next ?

## Consolidate Tooling

Next platform migration, replacing Consul/Nomad with Kubernetes.

In preparation for multi-cloud deployments.

Based on internal tooling written in Go.  
→ Replacing current legacy automation code



# **What's next ?**

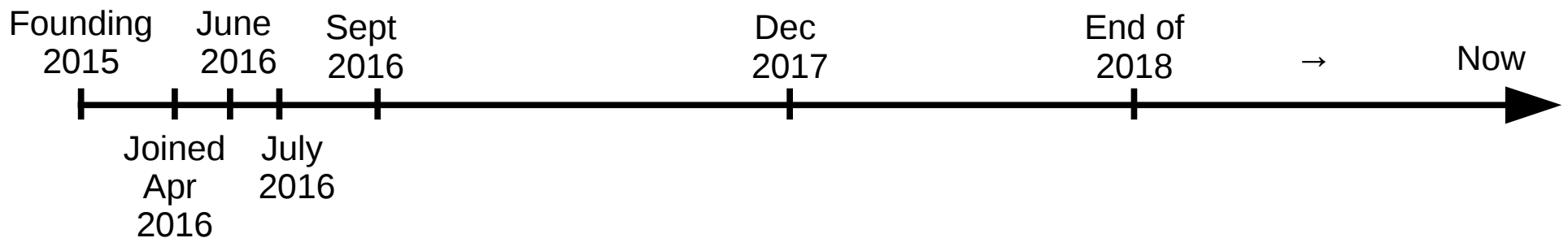
## **Sustainability**

**Expand SRE team based on on-call needs**  
→ first colleague in Australia

**Move non-core topics into other teams**  
→ Dev Support and On-Prem

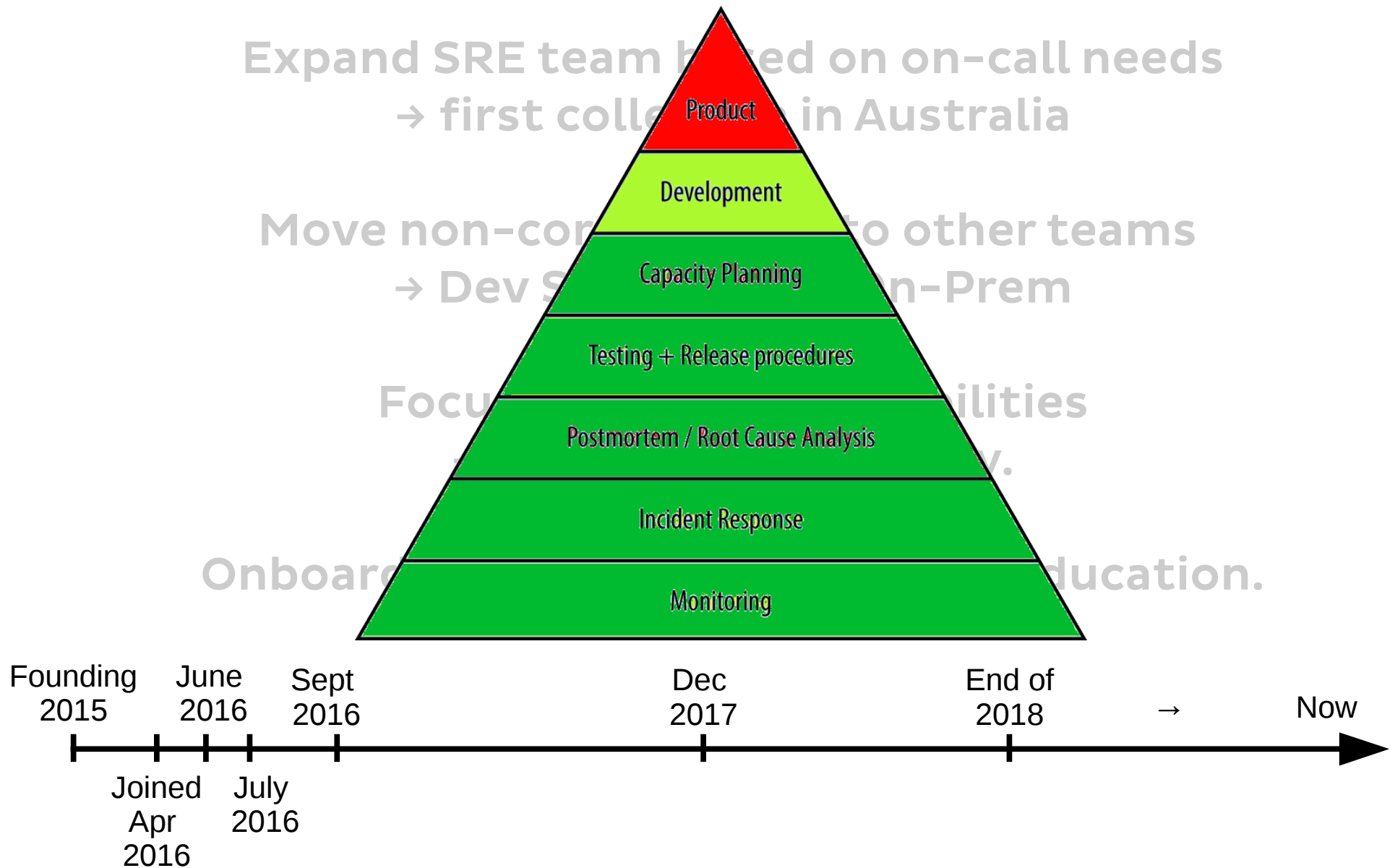
**Focus on core responsibilities**  
→ QoS. Cost. Scalability.

**Onboarding. Knowledge Sharing. Education.**



# What's next ?

## Sustainability



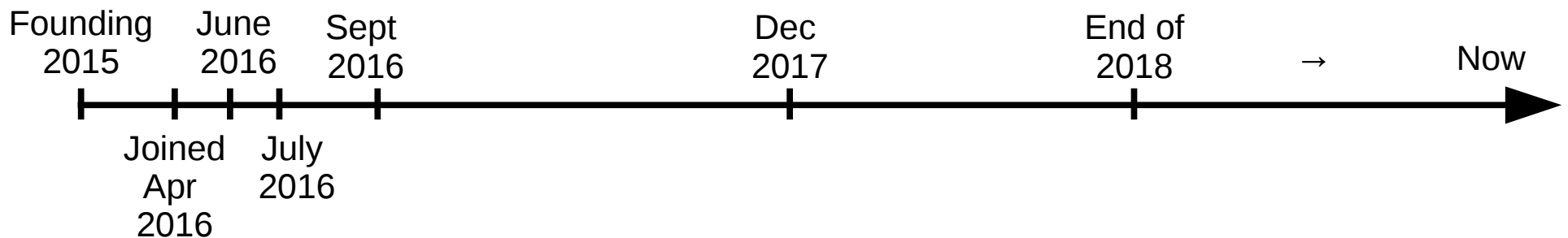
# Takeaways

**SRE is not a tool you use or a switch you turn on.  
SRE is a mindset and requires constant adjustment**

**Try (to learn) to do the right thing at the right time.  
Don't be afraid to break things.**

**You probably cannot avoid politics.  
→ Communication becomes more and more  
important as you grow!**

**It's all about customer satisfaction!**





**Thank you!**  
**Questions ?**

**@spanneberg**

**@instanaHQ**