

AWS  
re:Invent

DEV315-S

# Building SRE from scratch at Coinbase during hypergrowth

Niall O'Higgins  
**coinbase**



Daniel Maher  
**DATADOG**

“Our goal is to make Coinbase the most trusted and easiest to use digital currency exchange.”

**Brian Armstrong**  
**Co-founder & CEO of Coinbase**

# What is SRE?

- New field
- Definitions vary
- Many misconceptions



# Is this SRE?

- Endless firefighting?
- Being on-call?
- Operational toil?

Those are the **symptoms**; SRE is the **cure**.

# What about DevOps?

SRE satisfies many, if not all, of the operational and cultural elements of DevOps.



# Key SRE insight #1

Measure and improve human, organisational, and machine systems.



# Key SRE insight #2

Move from reactive to proactive. Go find sources of toil and eliminate them!





# Key SRE insight #3

Provide an organisational back-pressure mechanism.



# Set early expectations

- New language and fresh set of concepts.
- Takes time to absorb - no instant results!
- The best way to begin, is to begin.

# The Coinbase strategy

- Service Level Indicators
- The “Four Golden Signals”

# Metrics: the core of SLIs

- Natural tendency to over-engineer.
- Lots of data, none of it actionable or useful.

Optimise for KPIs, or high signal/noise.



# The Four Golden Signals

- Latency
- Traffic
- Errors
- Saturation



# Latency

- Direct impact on customer experience.
- Where and how you measure it is important.

# Traffic

- The amount of work being done - or attempted.
- Direct relationship with business value.

# Errors

- A nice, defined target to aim at.
- Direct impact on customer experience.

# Saturation

- Real Talk: this is a tricky one.
- Direct relationship to both scaling and capacity planning.



# Humans and the Four Golden Signals

- Latency
- Traffic
- Errors
- Saturation



# Start, then iterate

- Start with an initial specification - even if it's not ideal.
- Iterating on feedback is the key to getting it right.
- Keep it simple!

# Spreadsheets are simple, right?

Service	Latency	Errors	Saturation	Traffic
Foo	foo.latency	foo.error_rate	Disc space	TPS
Bar	bar.response_time	bar.error_rate	Memory	TPS



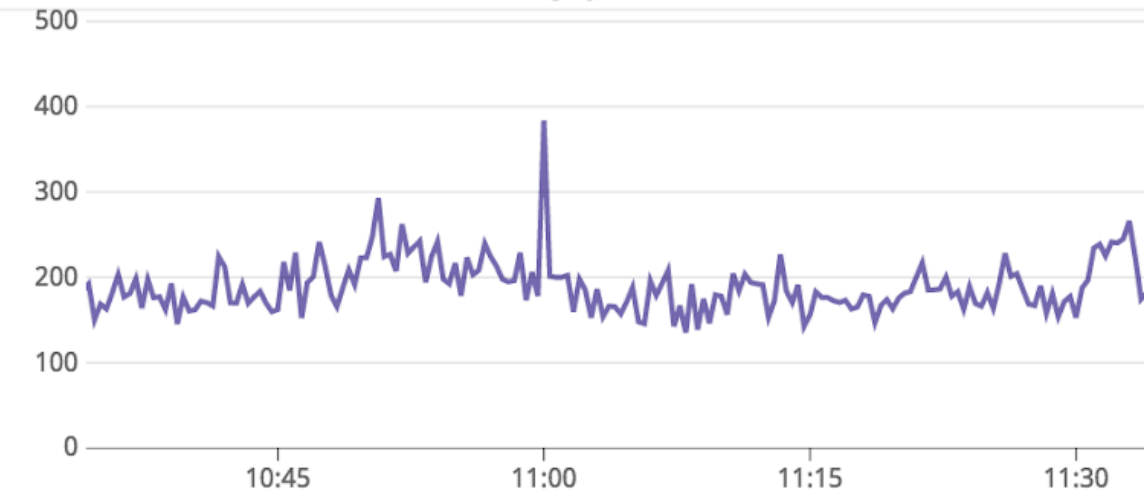
# SLIs: Defining “done”

- Per-service dashboard in Datadog with timeseries chart for each indicator.
- Document describing the indicators and why they are important.

# Datadog Dashboards

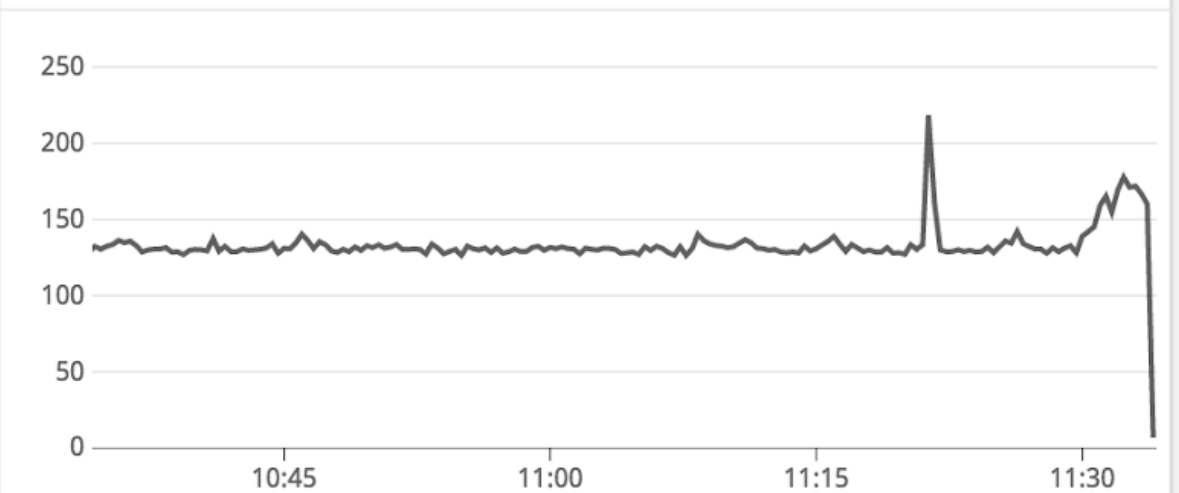
## Latency

consumer bulk write latency p95



## Traffic

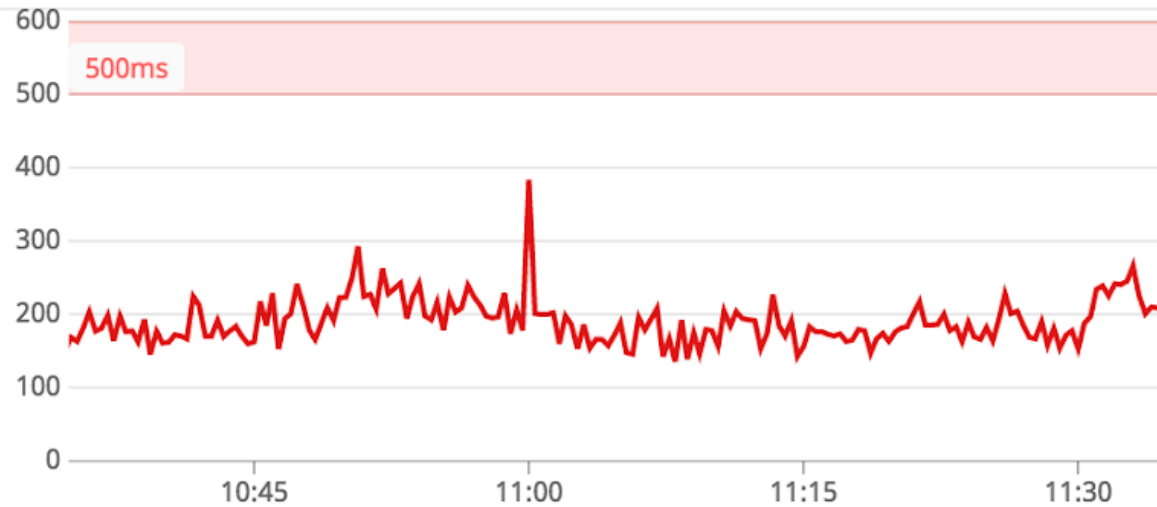
bulk write from consumers (writes/s)



# Datadog Dashboards

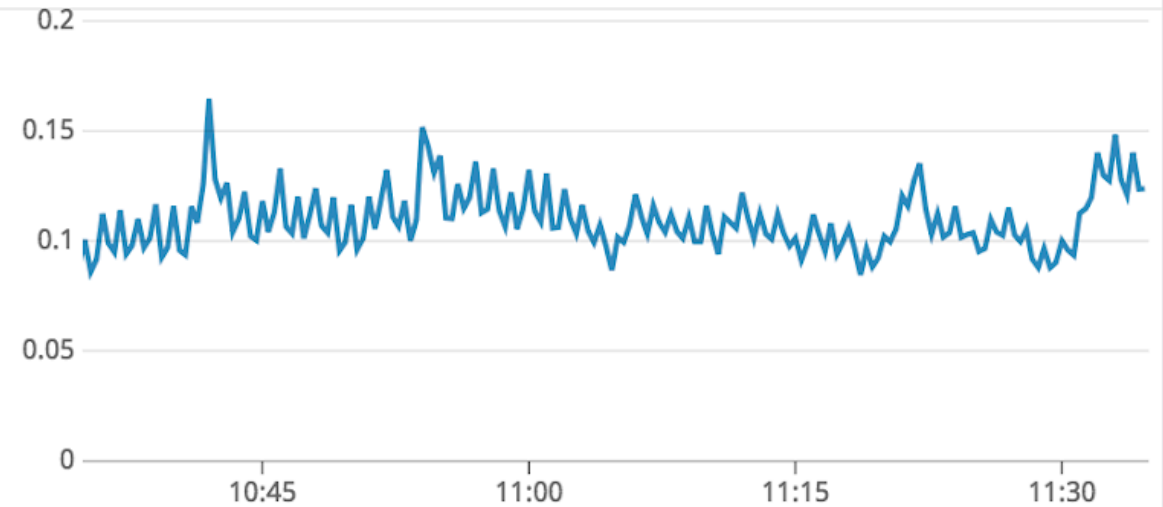
## Errors

consumer bulk write latency p95 over threshold



## Saturation

cluster-wide load1.norm avg



# Specification Documentation

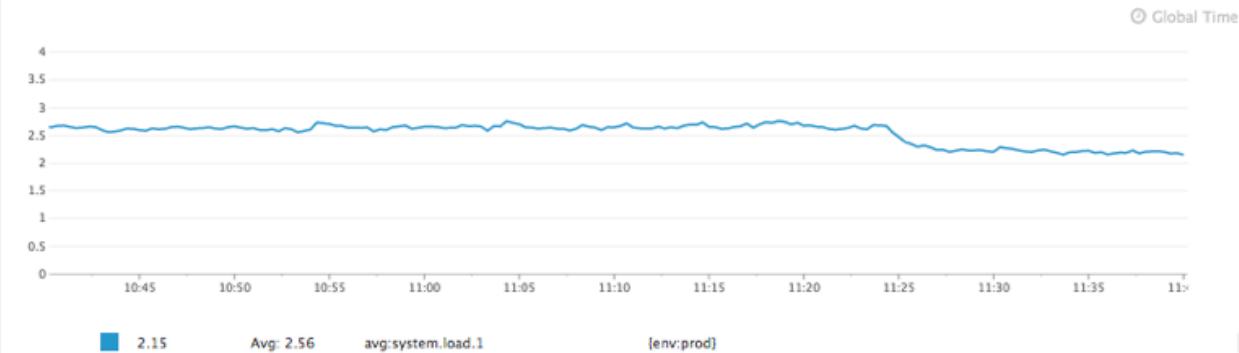
- Spec vs. implementation
- Where do you want to instrument?
- Where is it easy to instrument?

## Welcome to the Datadog Example Notebook.

Notebooks are designed to help you tell stories with your data. We've all heard the adage, *a picture is worth a thousand words*, and it certainly rings true when sharing metrics information. From reporting on infrastructure changes to sharing incident retrospectives, visualizing data allows you to communicate better.

Notebooks are composed of sequential markdown and graph cells. This is a markdown cell and below you'll see a graph cell. You can easily edit any cell by clicking on it.

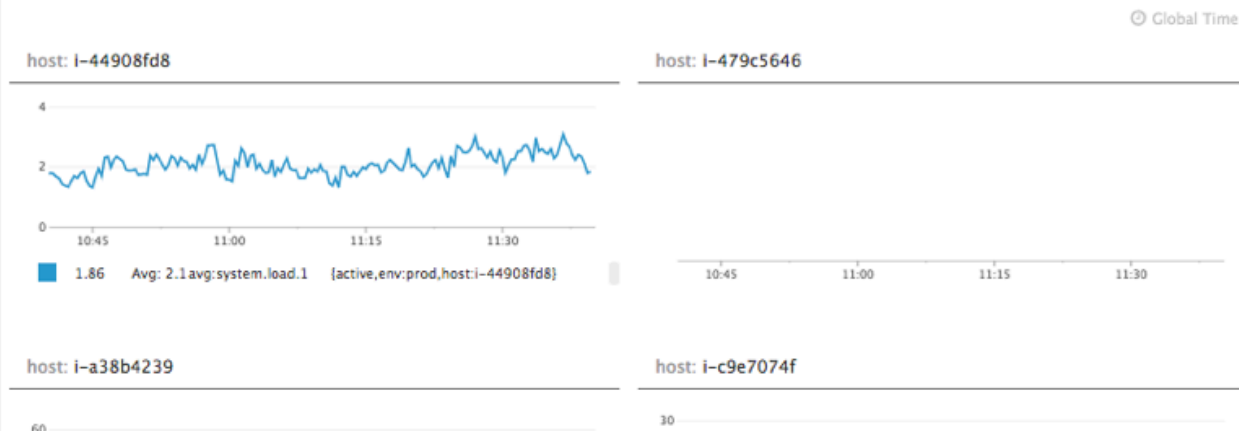
`avg:system.load.1{env:prod}`



The graph above is your average system load over the past hour from your environments that have been tagged as "prod". Note that it is following *Global Time* as indicated by the text on the right, above the graph. This means that it's following the time indicated at the top of the page.

The time indicator should look familiar. It's similar to the time controls in your other Datadog dashboards. You can select different durations and use the arrow buttons to move forward or backward in time.

`avg:system.load.1{active,env:prod}`



# First SLIs, then Promises

- Plain-language statements; easy to parse, easy to understand.
- Plenty of potential stakeholders.
- Start simple!

“You can rely on us to buy or sell crypto whenever you want.”

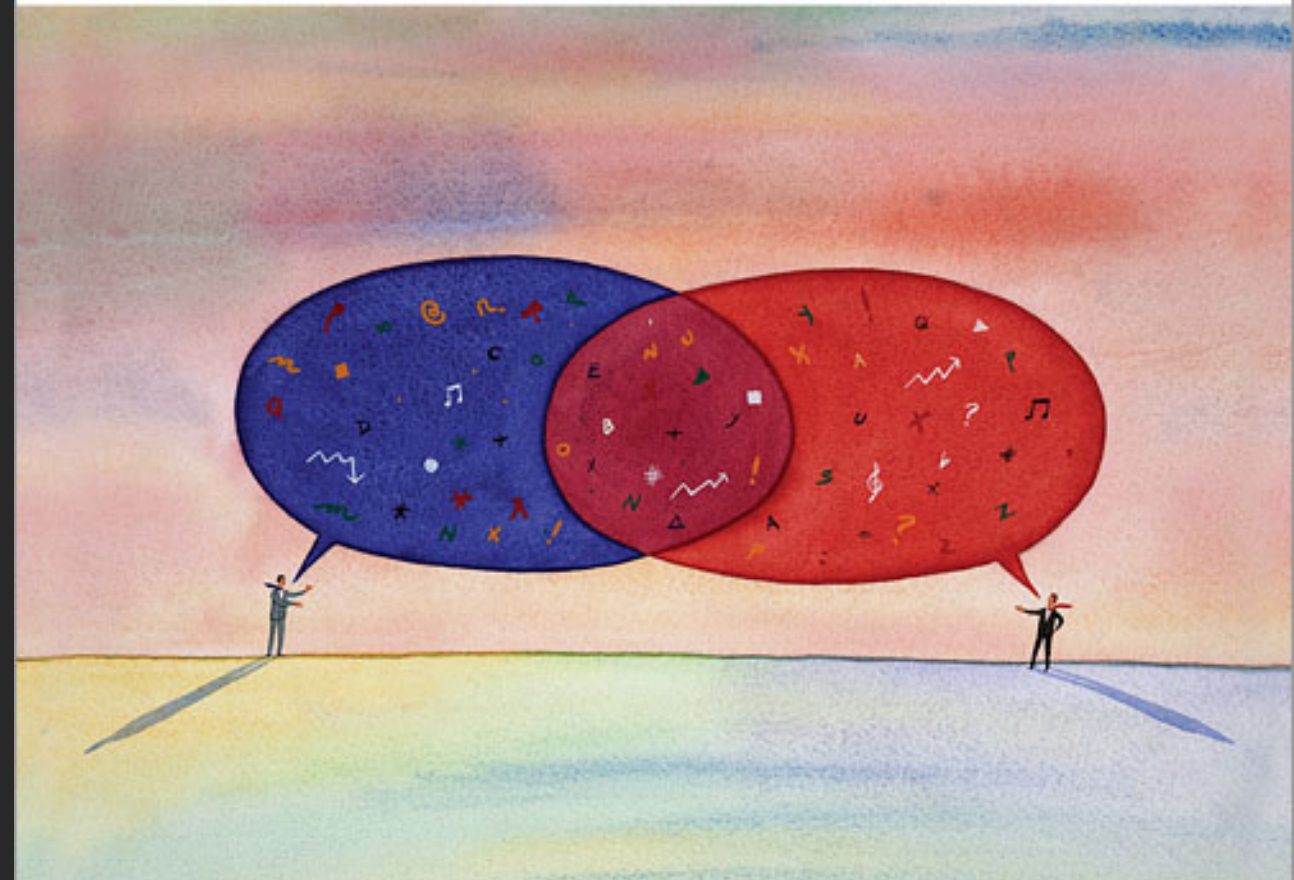
**Coinbase’s “Prime Promise”**

# Essential Reading

## Thinking in Promises by Mark Burgess

# THINKING IN **PROMISES**

DESIGNING SYSTEMS FOR COOPERATION



# Concerning Promises

- Promises have two parties.
- Promises can be human to machine, human to human, or machine to machine.



# Example Promises

- You service promises to respond to clients within 50ms.
- A service you depend on promises that its error rate will be  $< 1\%$ .
- On-call promises they will engage an incident within 15 minutes.

# Promise enumeration

- Each team must formalise the promises they are willing to keep.
- They must also understand the promises they rely upon to function.
- When a promise you rely upon is broken, what should you do? Who should you contact?

# Promises: defining “done”

Promises are done when they have a Datadog monitor (alert).

## 1 Pick hosts by name or tag

role:cassandra x

excluding

## 2 Set alert conditions

Check Alert

Cluster Alert

An alert is triggered when the % of checks in a status exceeds your threshold.

Ungrouped ▾

Calculate status percentage across all active hosts.

### % Missing Data to Alert



### % Missing Data to Warn



The alert will...

- **trigger** when more than 50% of hosts are missing data
- **warn** when more than 30% of hosts are missing data
- **resolve** in all other cases

Notify if data is missing for more than  minutes.

[Never] ▾

automatically resolve this event from a triggered state.

# When Promises are broken...

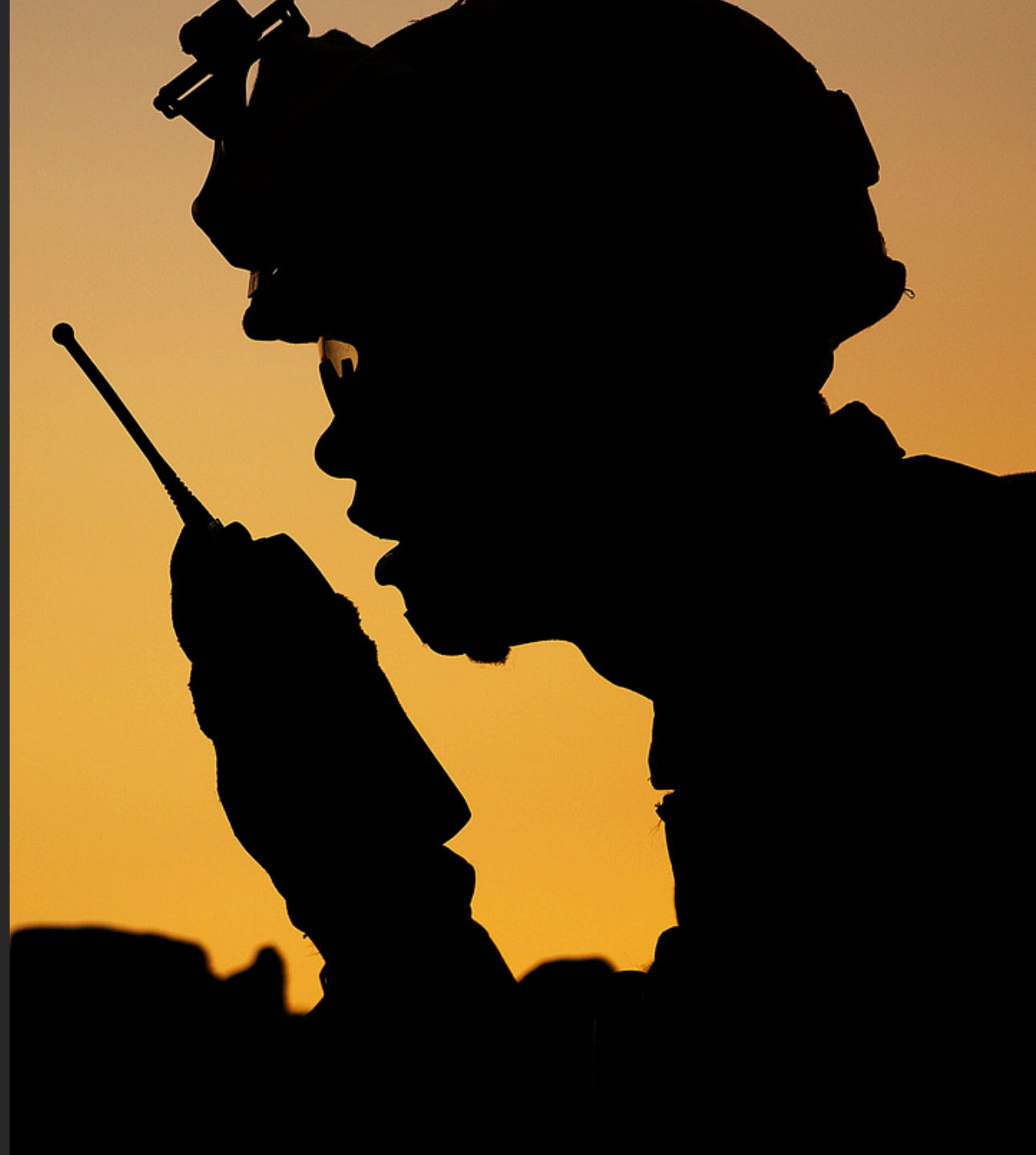
- It is inevitable that a promises will be broken at some point.
- What to do when that happens?

# Blameless post-mortems

- Is “blameless post-mortem” a real thing?
  - What about data-driven post-mortems?
  - How does this relate to promises - specifically broken ones?
- 
- [https://v.gd/jpr\\_post\\_mortems](https://v.gd/jpr_post_mortems)
  - [https://v.gd/jyee\\_datadriven](https://v.gd/jyee_datadriven)

# Interpreting incidents

- Build a shared language.
- Practise communicating.
- Understand that incidents and outages are broken promises.



# Measuring incident response

- Quantify and measure the quality of your incident responses.
- Quantitative: Time to detect, time to engage, time to fix.
- Qualitative: Quality of communication.

# The end-game

- Have a clear answer for “why SRE?”
- Start with instrumentation – keep it simple to start.
- Enumerate your promises.
- Measure your response when promises are broken.
- Transparency
- Understanding
- Confidence



# Thank you!

Niall O'Higgins  
**coinbase**



Daniel Maher  
**DATADOG**



Please complete the session  
survey in the mobile app.