PID loops and the art of keeping systems stable

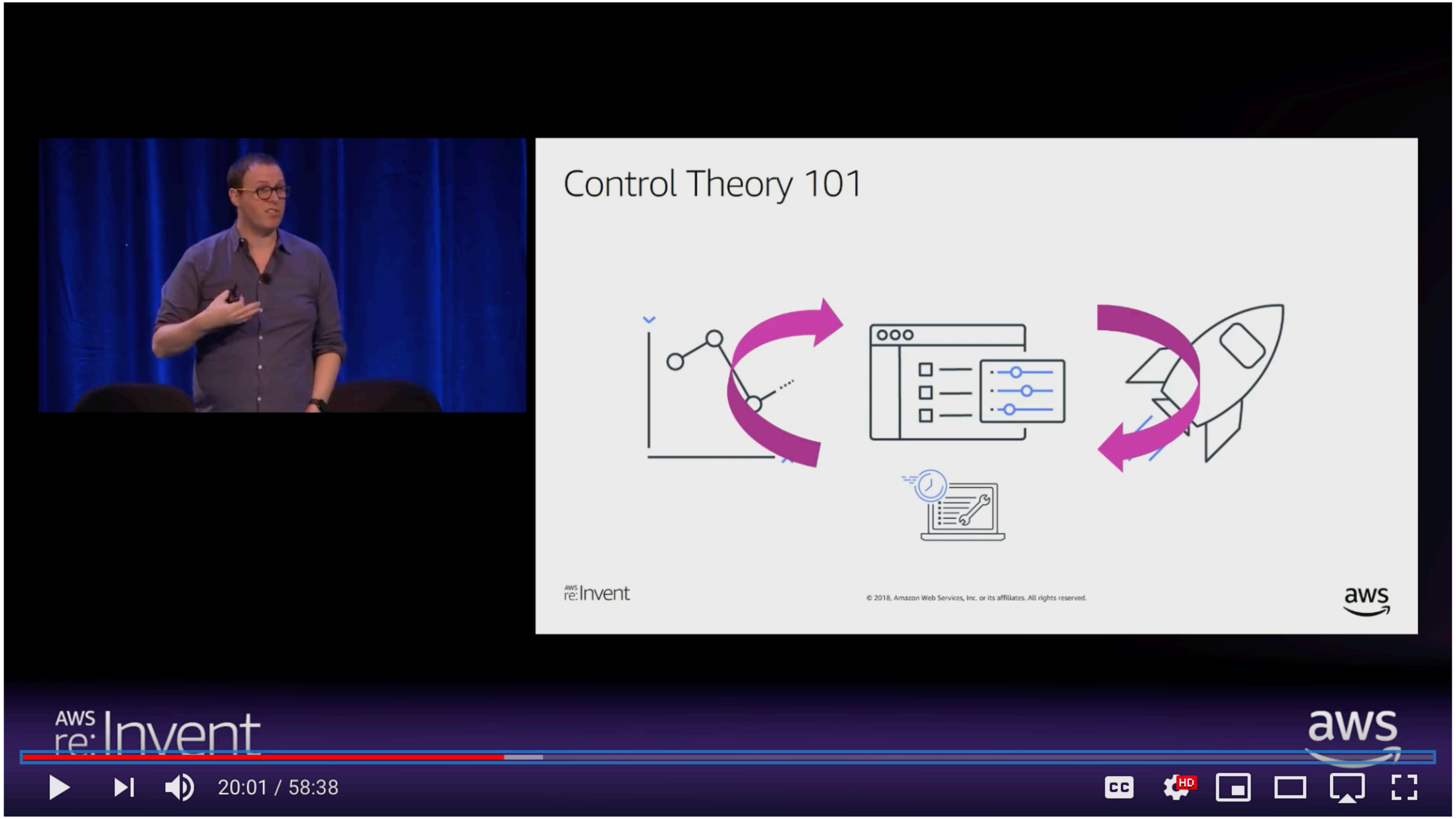
@colmmacc

Colm MacCárthaigh

2019-06-24

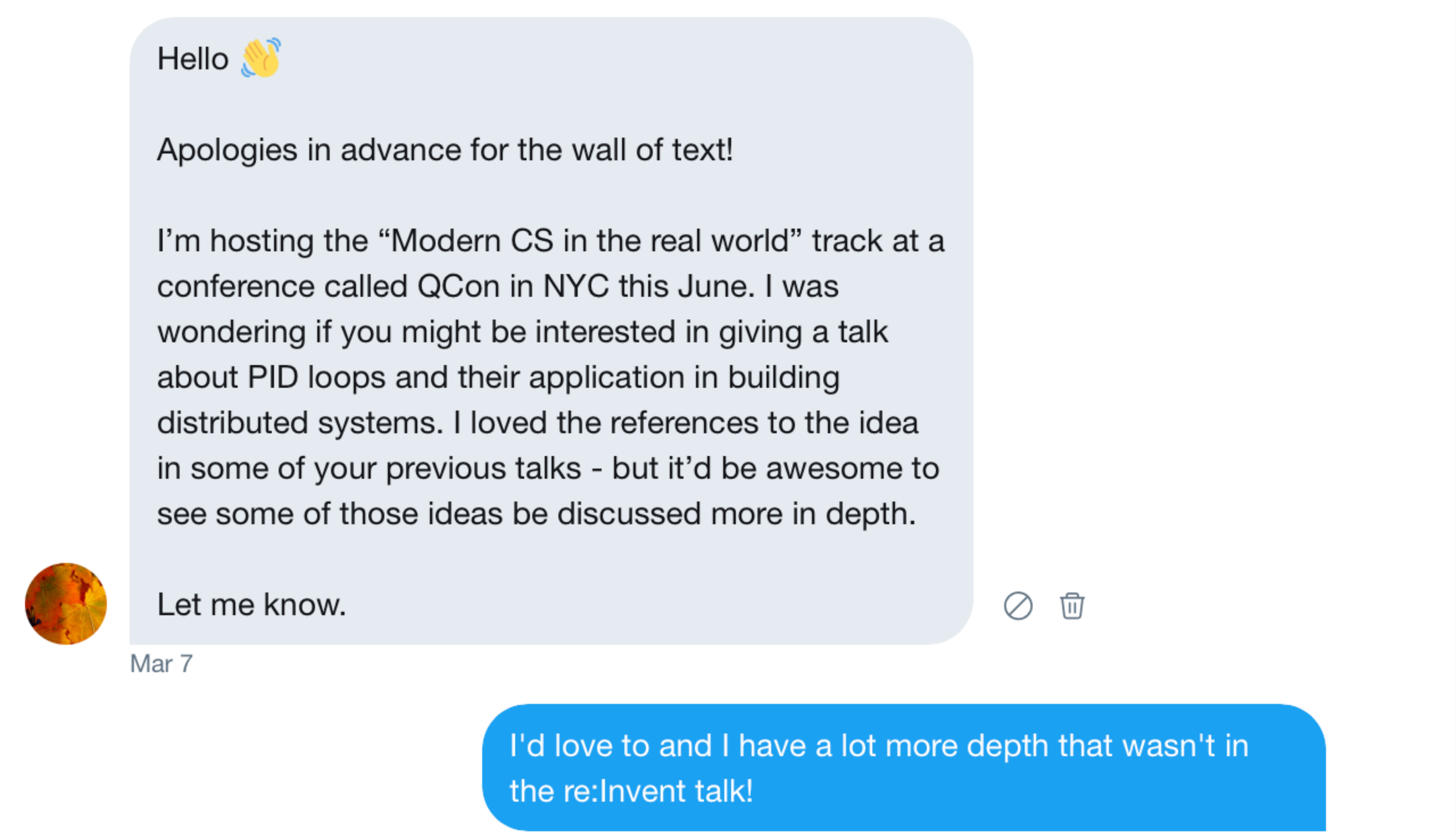
Control Theory: Where the fruit is hanging so low IT IS TOUCHING THE GROUND

META



Control Theory 101
re:Invent
© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
aws
AWS re:Invent
aws
20:01 / 58:38

AWS re:Invent 2018: Close Loops & Opening Minds: How to Take Control of Systems, Big & Small ARC337



Hello 👋

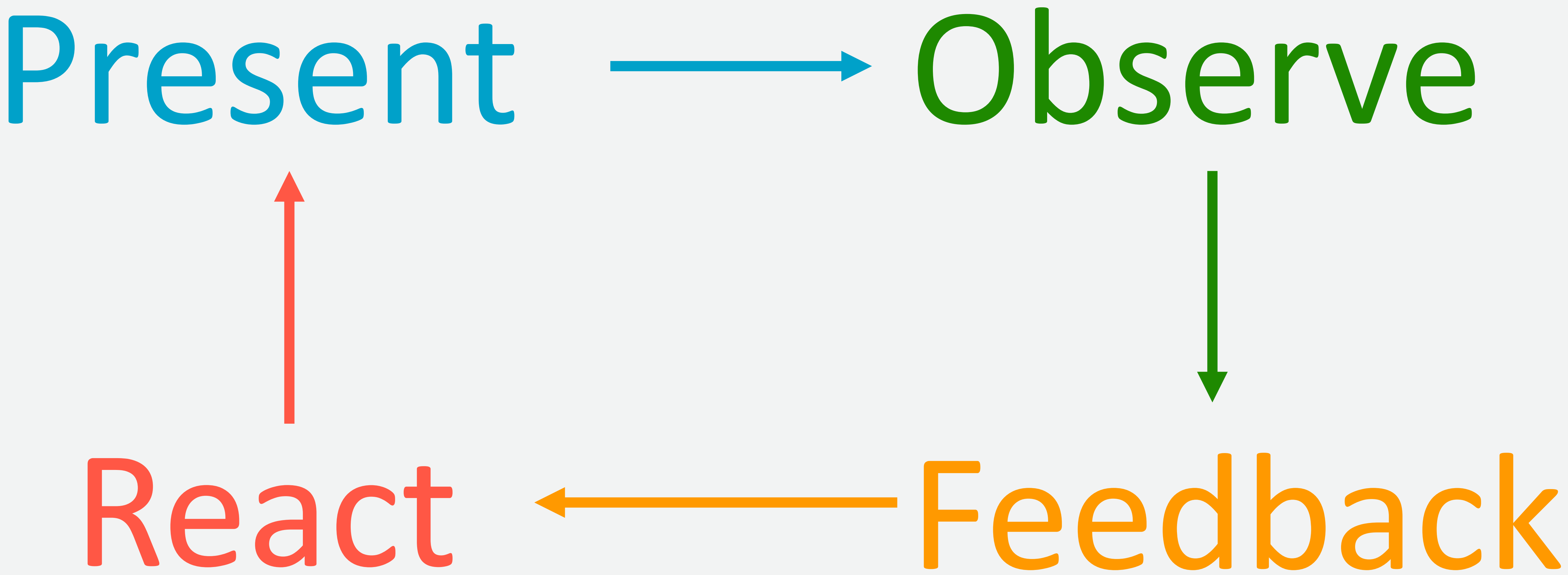
Apologies in advance for the wall of text!

I’m hosting the “Modern CS in the real world” track at a conference called QCon in NYC this June. I was wondering if you might be interested in giving a talk about PID loops and their application in building distributed systems. I loved the references to the idea in some of your previous talks - but it’d be awesome to see some of those ideas be discussed more in depth.

Let me know.

Mar 7

I'd love to and I have a lot more depth that wasn't in the re:Invent talk!



Present

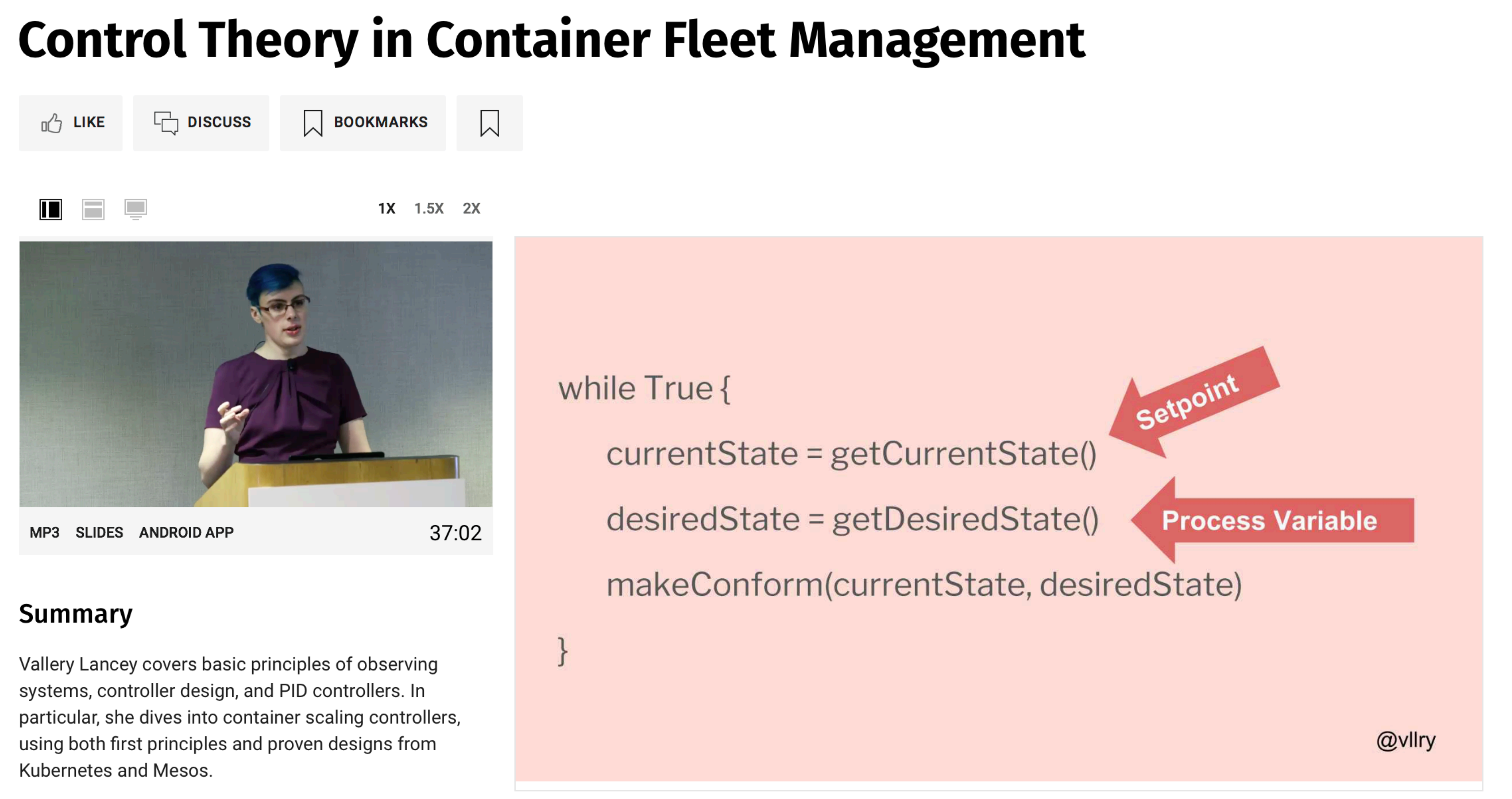
Observe

Feedback

React

Control Theory

Prior Art



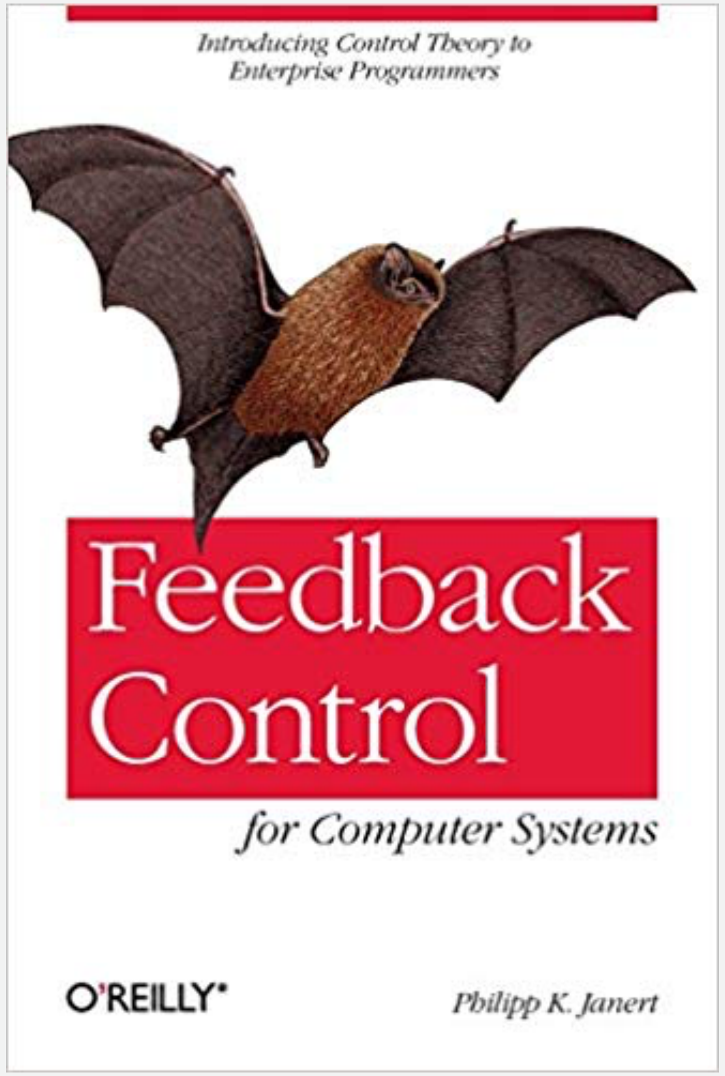
Control Theory in Container Fleet Management

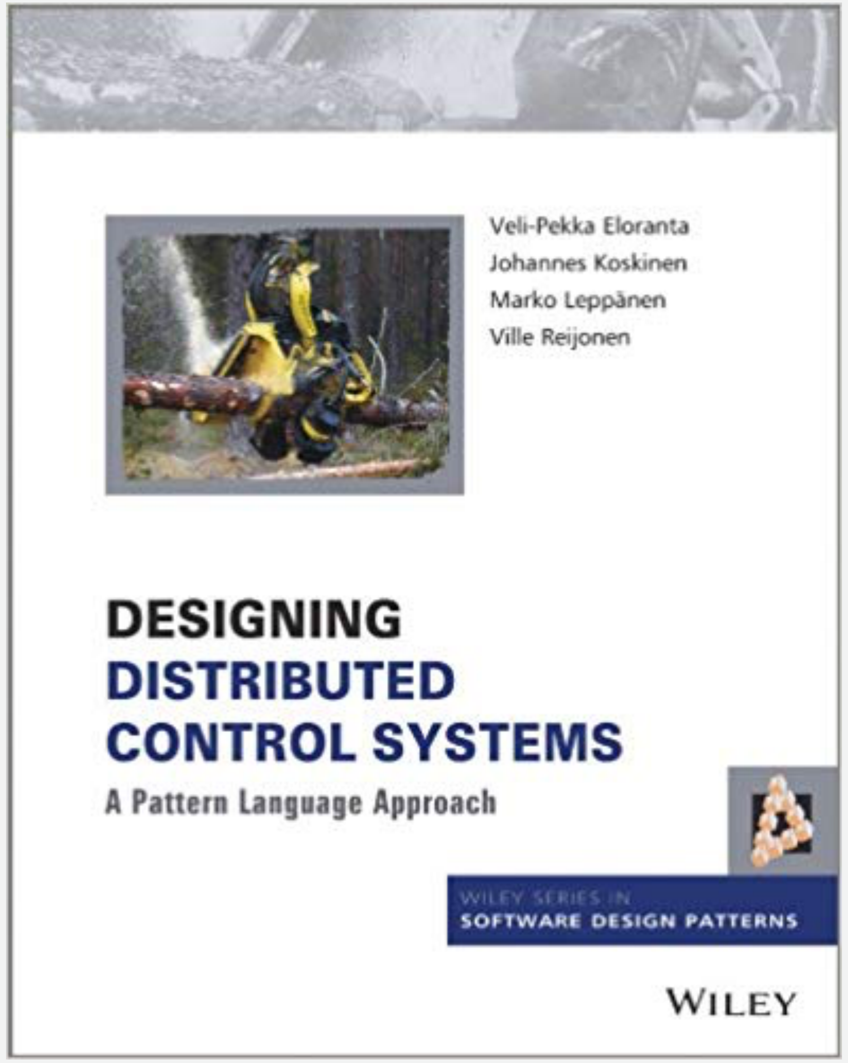
```
while True {
    currentState = getCurrentState()
    desiredState = getDesiredState()
    makeConform(currentState, desiredState)
}
```

Summary

Vallery Lancey covers basic principles of observing systems, controller design, and PID controllers. In particular, she dives into container scaling controllers, using both first principles and proven designs from Kubernetes and Mesos.

Prior Art


Introducing Control Theory to Enterprise Programmers
Feedback Control
for Computer Systems
O'REILLY®
Philipp K. Janert


Veli-Pekka Eloranta
Johannes Koskinen
Marko Leppänen
Ville Reijonen
DESIGNING
DISTRIBUTED
CONTROL SYSTEMS
A Pattern Language Approach
WILEY SERIES IN
SOFTWARE DESIGN PATTERNS
WILEY

Control Theory and PID loops

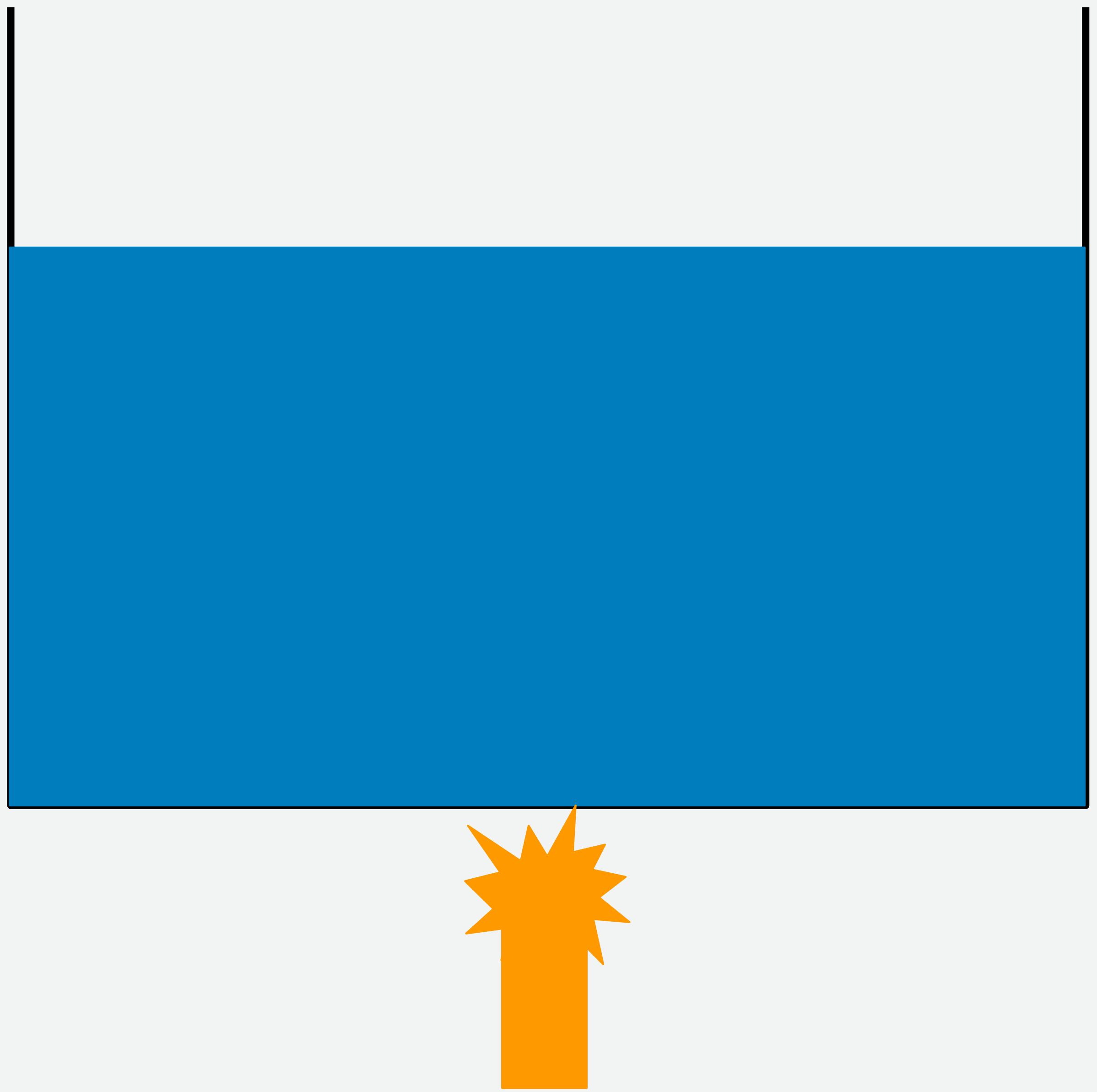
Comes up in the context of …

Autoscaling and placement: Instances, Storage, Network, etc.

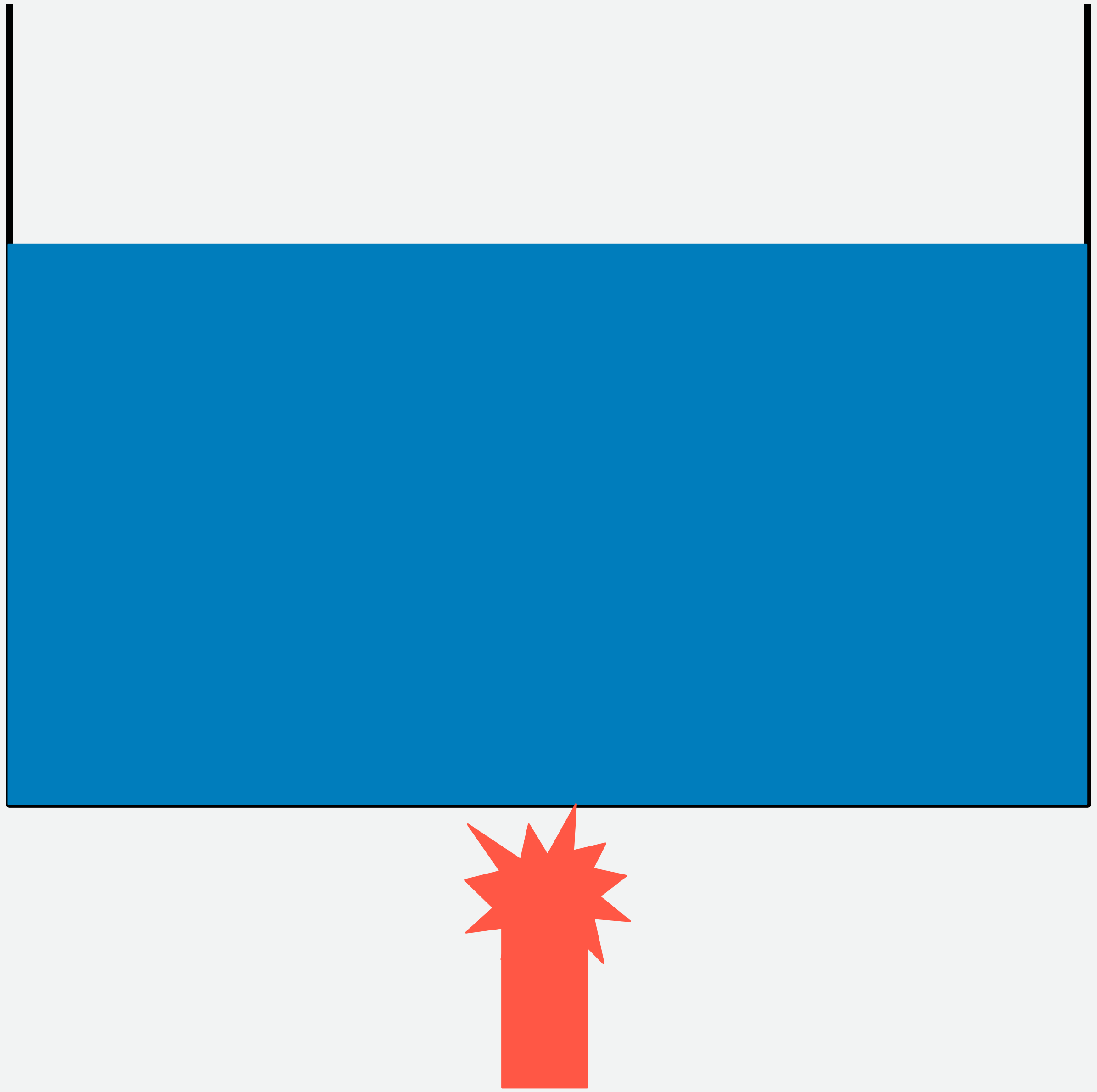
Fairness algorithms: TCP, Queues, Throttling

Systems stability

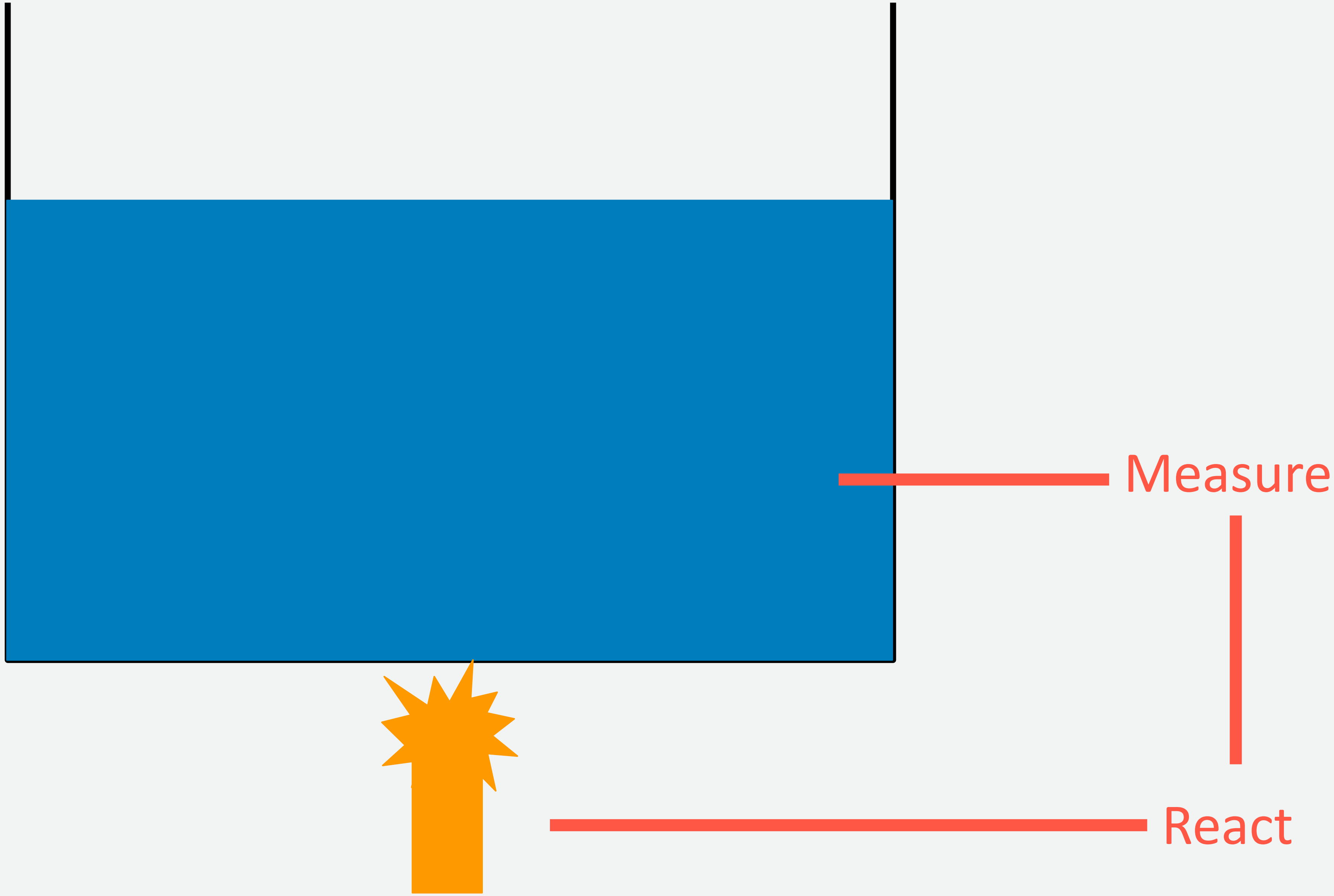
The Furnace



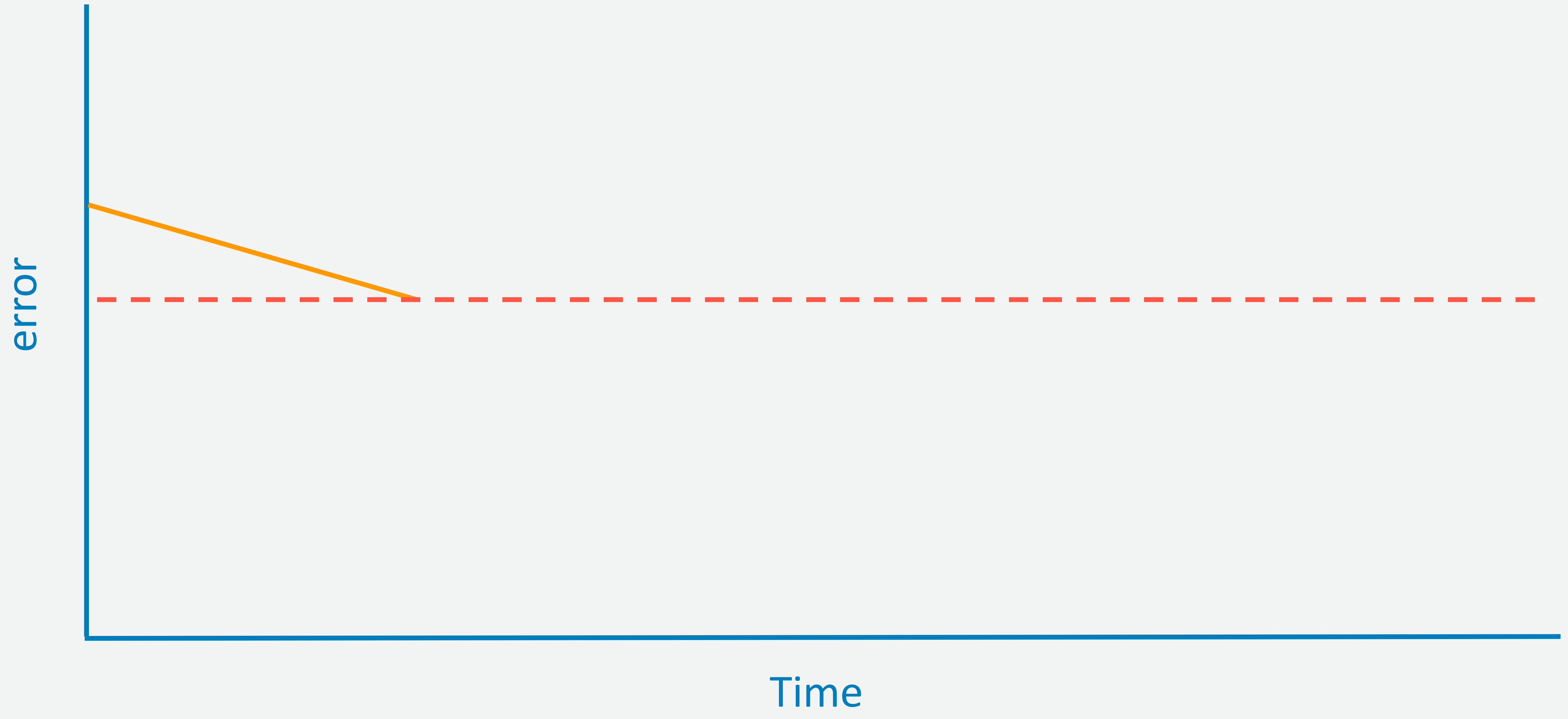
The Furnace



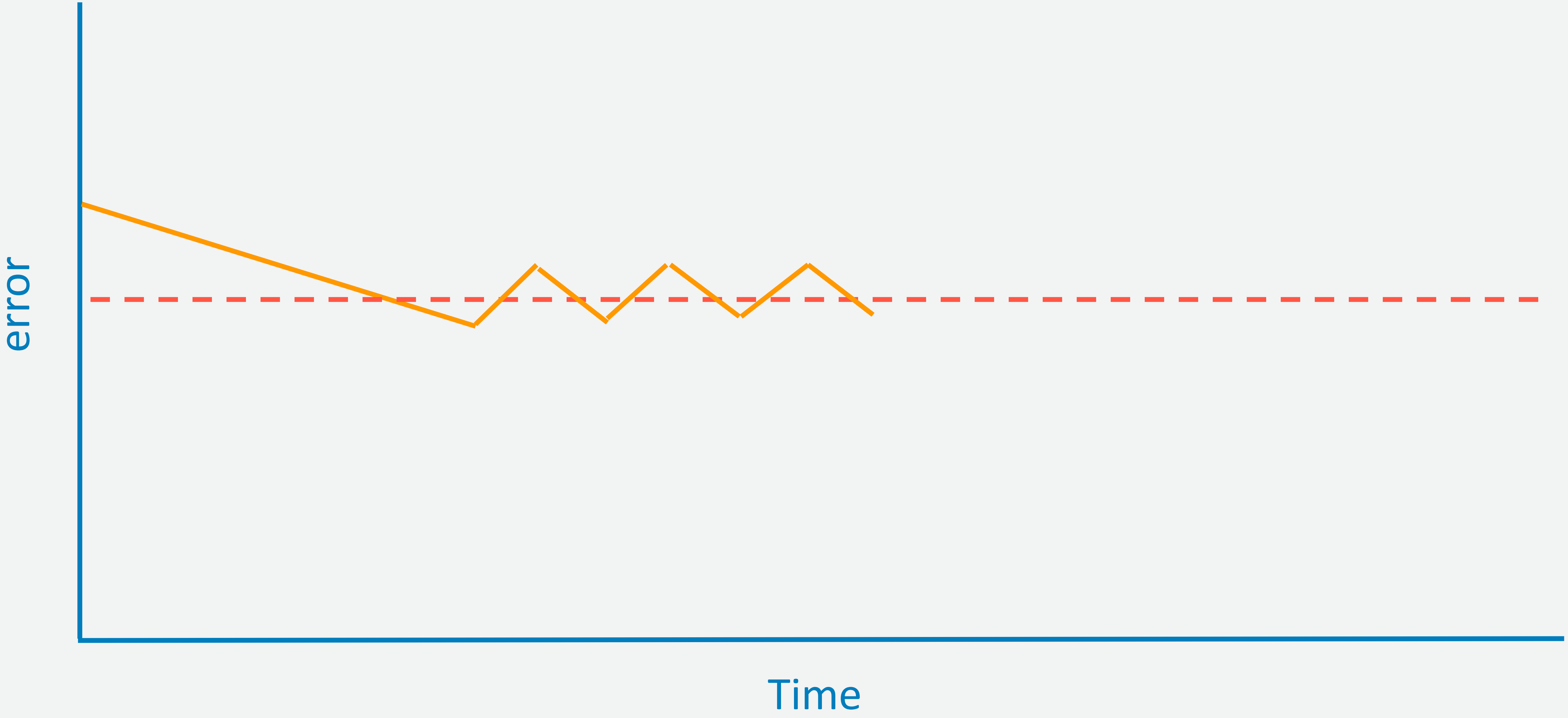
The Furnace



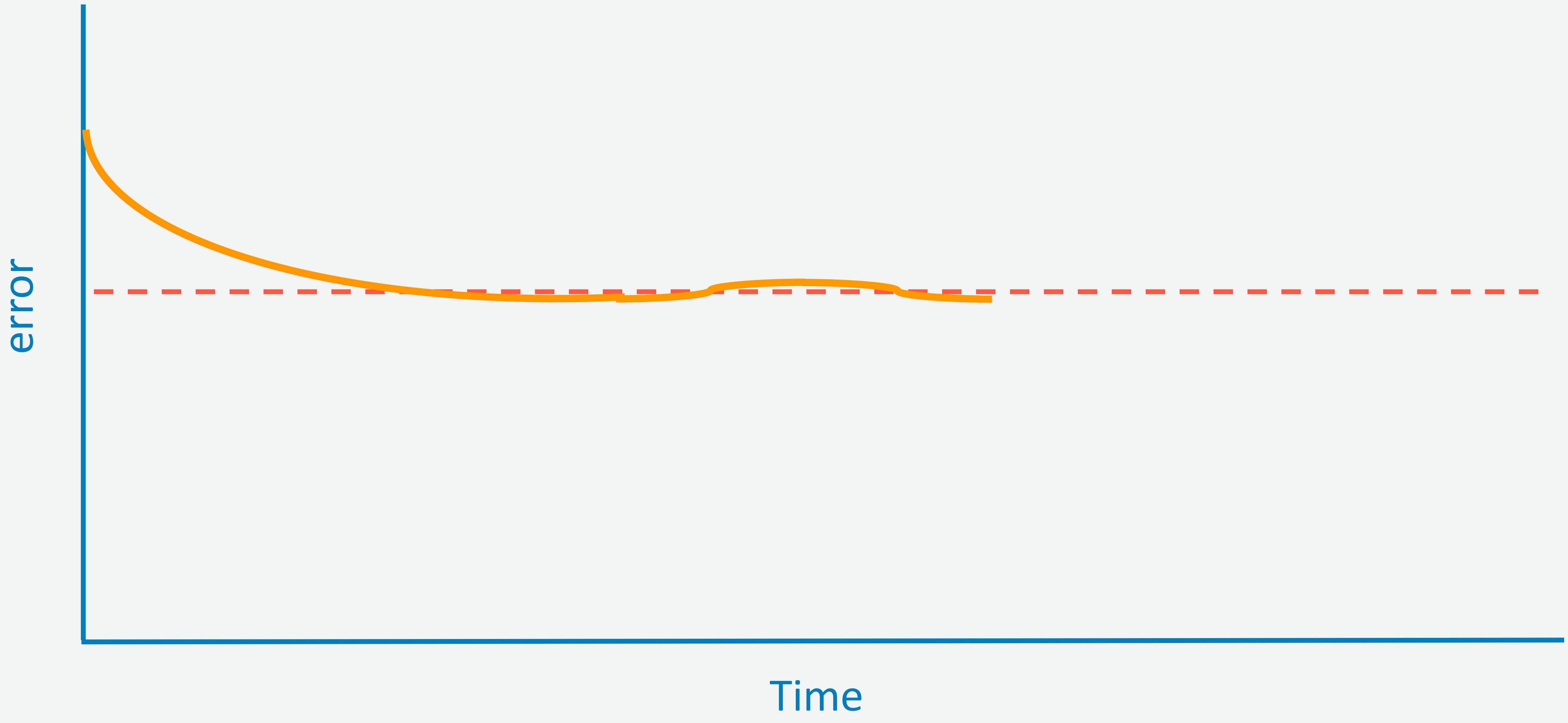
The Furnace



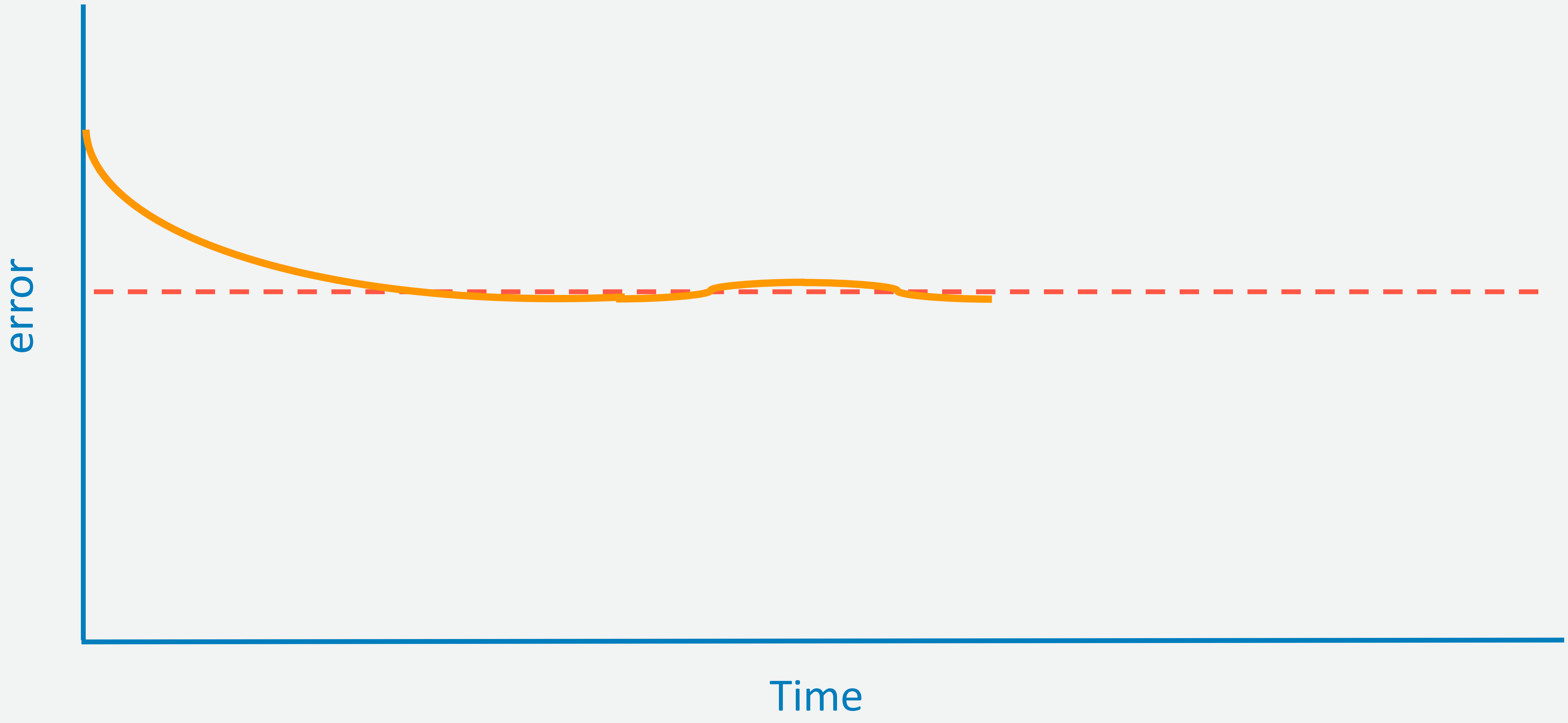
The Furnace



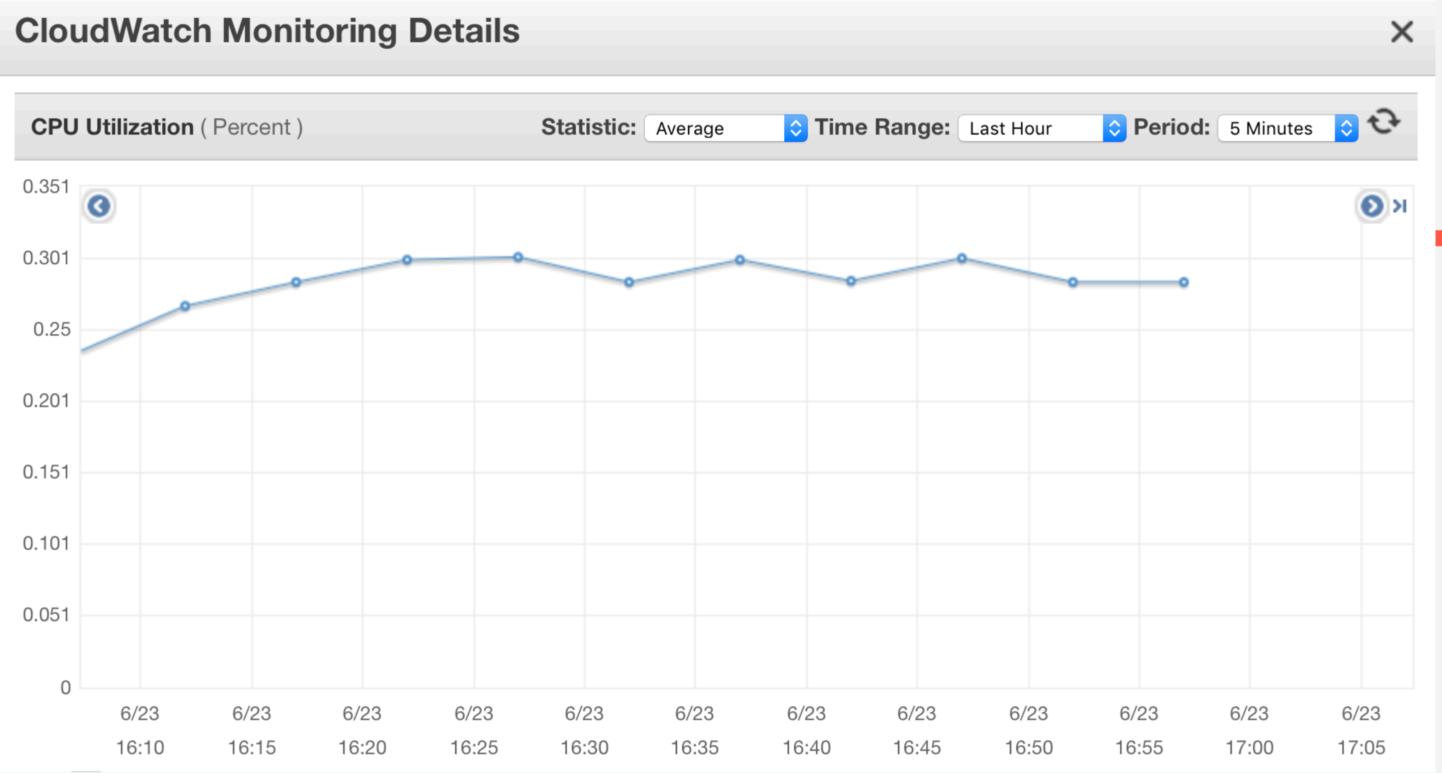
The Furnace



Autoscaling



Autoscaling



CloudWatch Monitoring Details

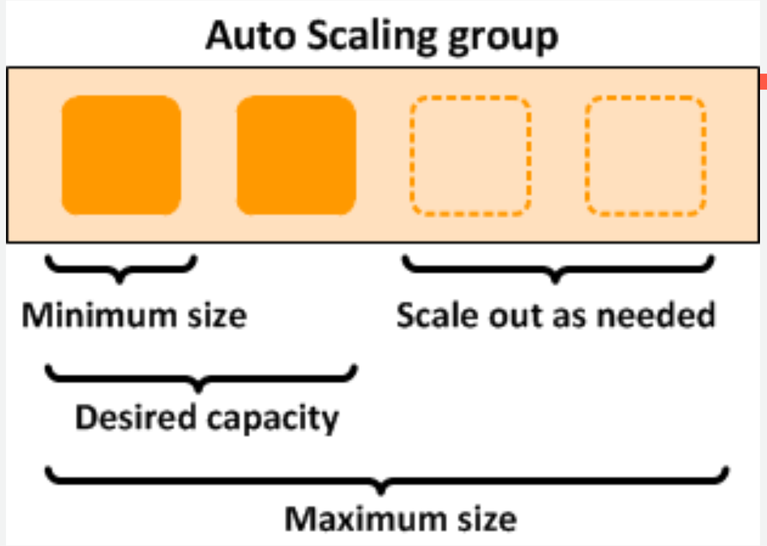
CPU Utilization (Percent)

Statistic: Average

Time Range: Last Hour

Period: 5 Minutes

Measure



React

Autoscaling : forecasting and fancy integrals!

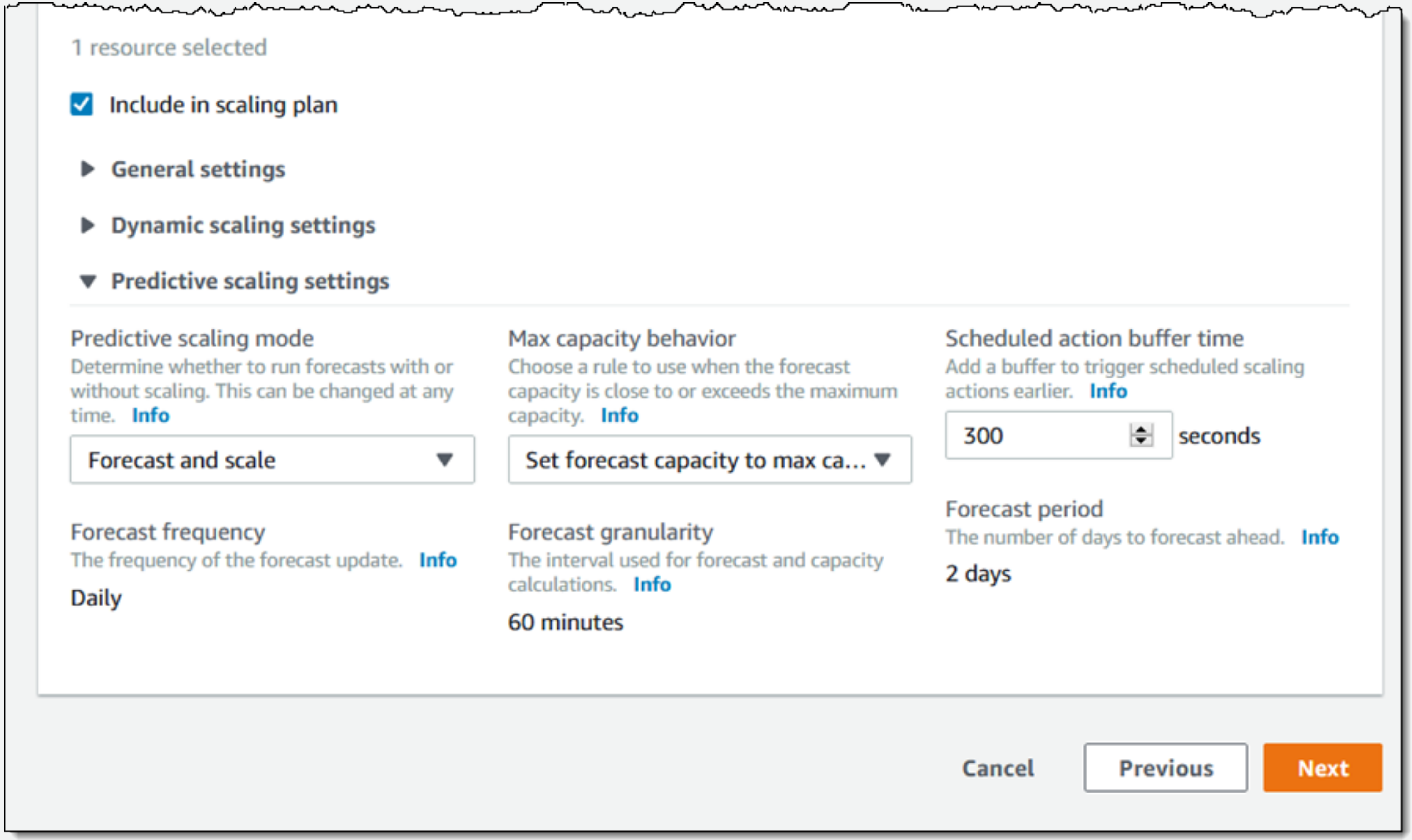
Any signal can be processed with Fourier Analysis to find underlying constituent frequencies

Real-world operational systems often have strong daily, weekly, annual cycles, etc.

Holt-Winters Forecasting can simulate these cycles into the future

Machine Learning can do even better!

Autoscaling : forecasting and fancy integrals!



1 resource selected

☑ Include in scaling plan

▶ General settings

▶ Dynamic scaling settings

▼ Predictive scaling settings

Predictive scaling mode
Determine whether to run forecasts with or without scaling. This can be changed at any time. Info
Forecast and scale ▼

Max capacity behavior
Choose a rule to use when the forecast capacity is close to or exceeds the maximum capacity. Info
Set forecast capacity to max ca... ▼

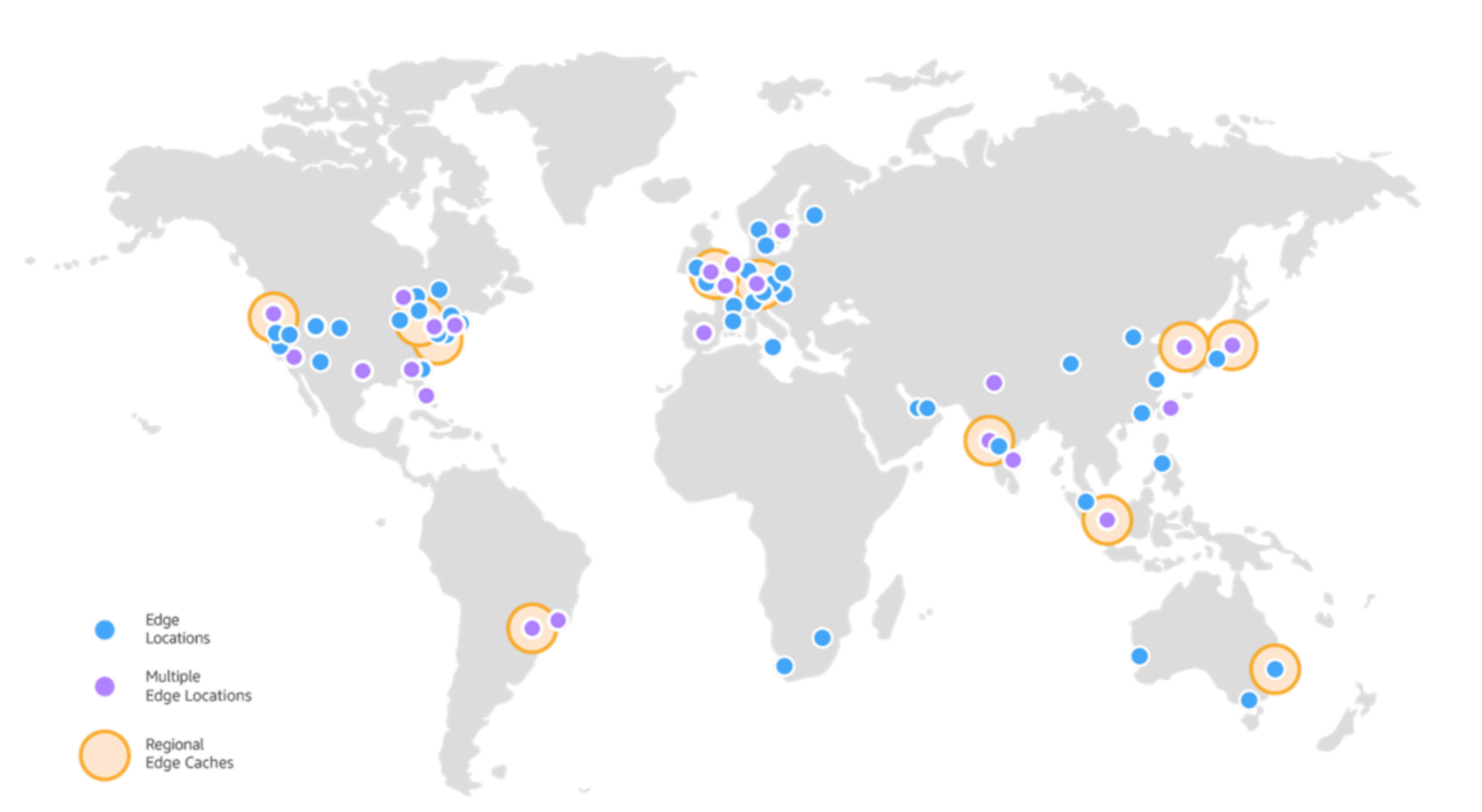
Scheduled action buffer time
Add a buffer to trigger scheduled scaling actions earlier. Info
300 seconds

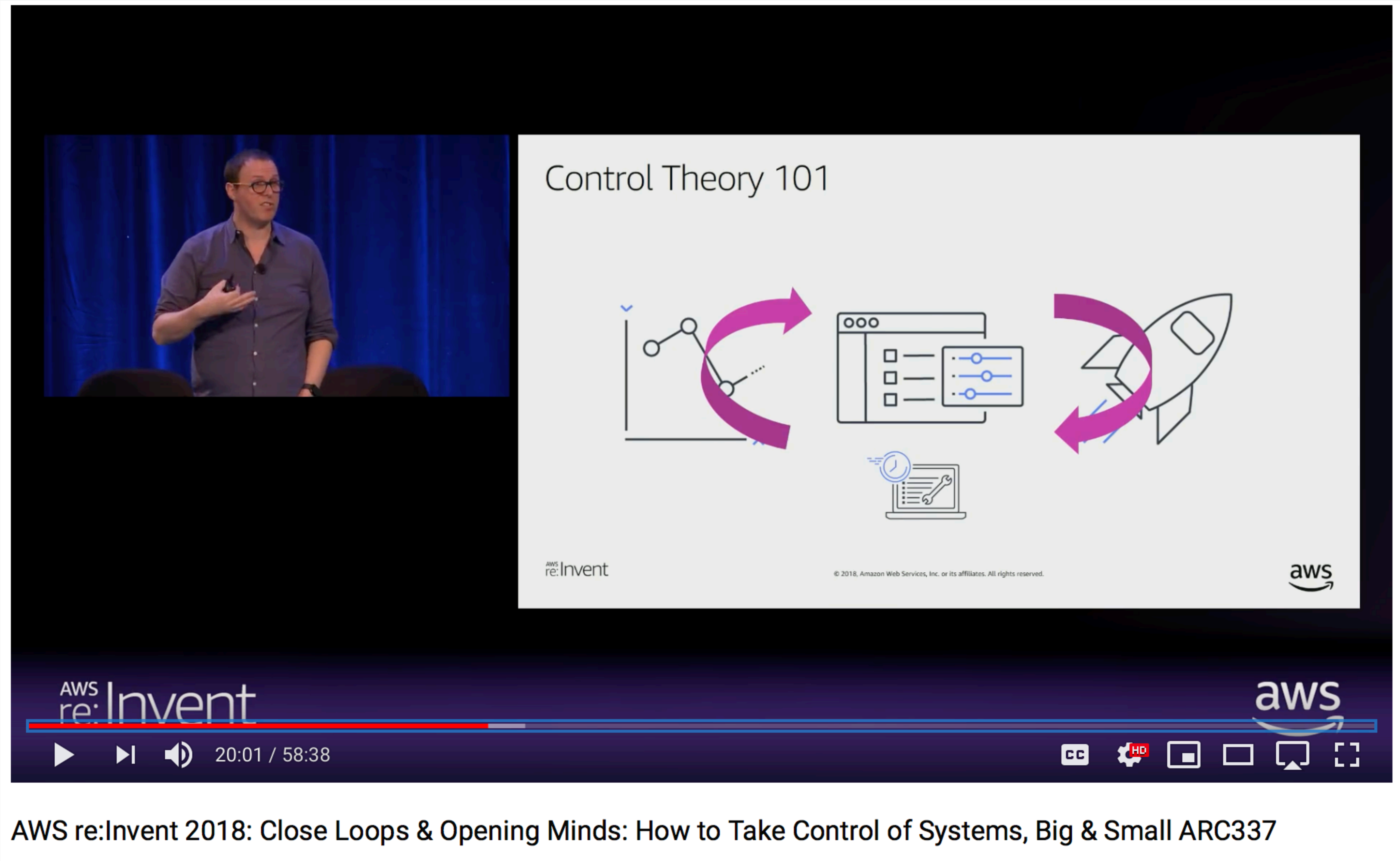
Forecast frequency
The frequency of the forecast update. Info
Daily

Forecast granularity
The interval used for forecast and capacity calculations. Info
60 minutes

Forecast period
The number of days to forecast ahead. Info
2 days

Placement and fairness




Control Theory 101

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS re:Invent 2018: Close Loops & Opening Minds: How to Take Control of Systems, Big & Small ARC337

1

X-Ray Vision: Open Loops

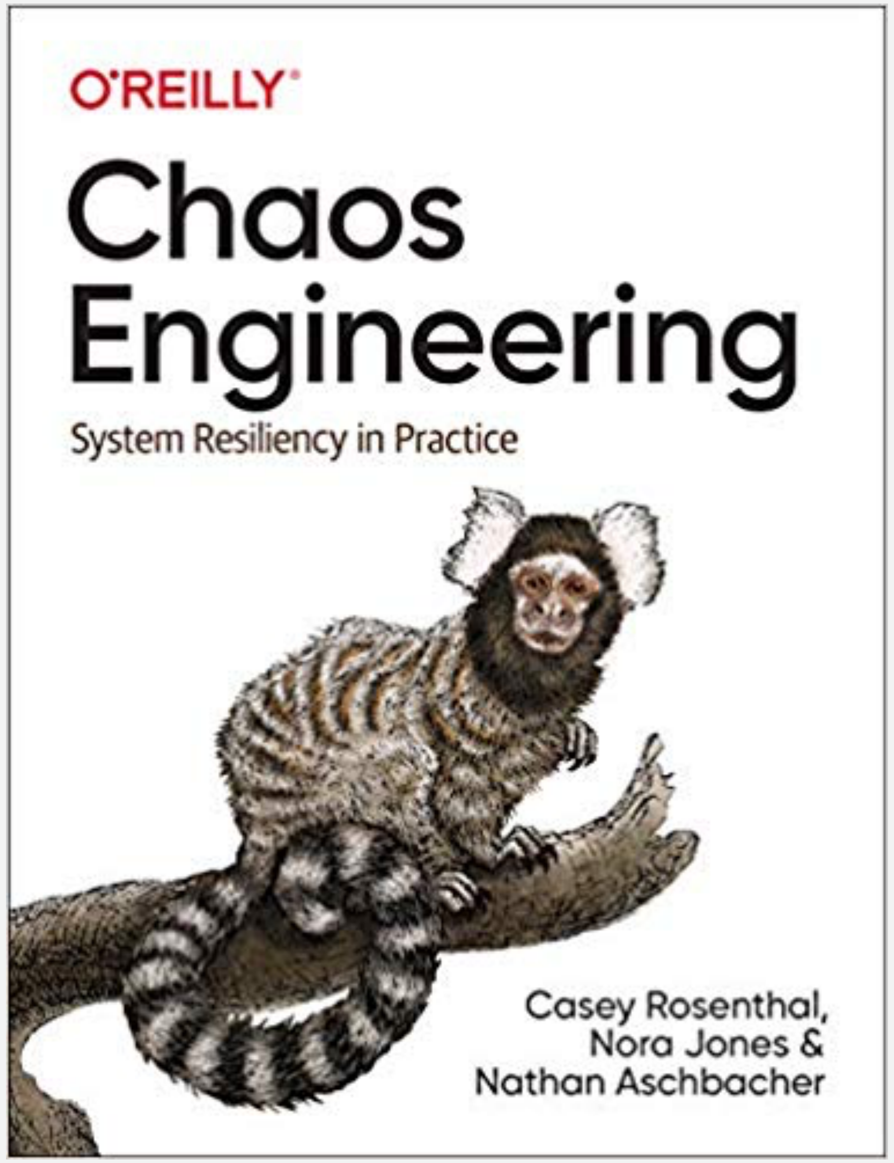
X-Ray Vision: Open Loops

```
# launch 10 Instances
for (i = 0; i < 10; i++)
    instance[i] = ec2_launch_instance()

# wait a minute
sleep(60);

# Register the instances
for (i = 0; i < 10; i++)
    register_instance(instance[i]);
```

X-Ray Vision: Open Loops

- A surprising number of real-world systems are Open Loops
- Potential reasons:
    - Organic out-growth from scripts
    - Imperative programming “*Do this, then do this*” is very natural
    - Infrastructure is very very reliable these days
    - Infrequent actions

X-Ray Vision: Open Loops

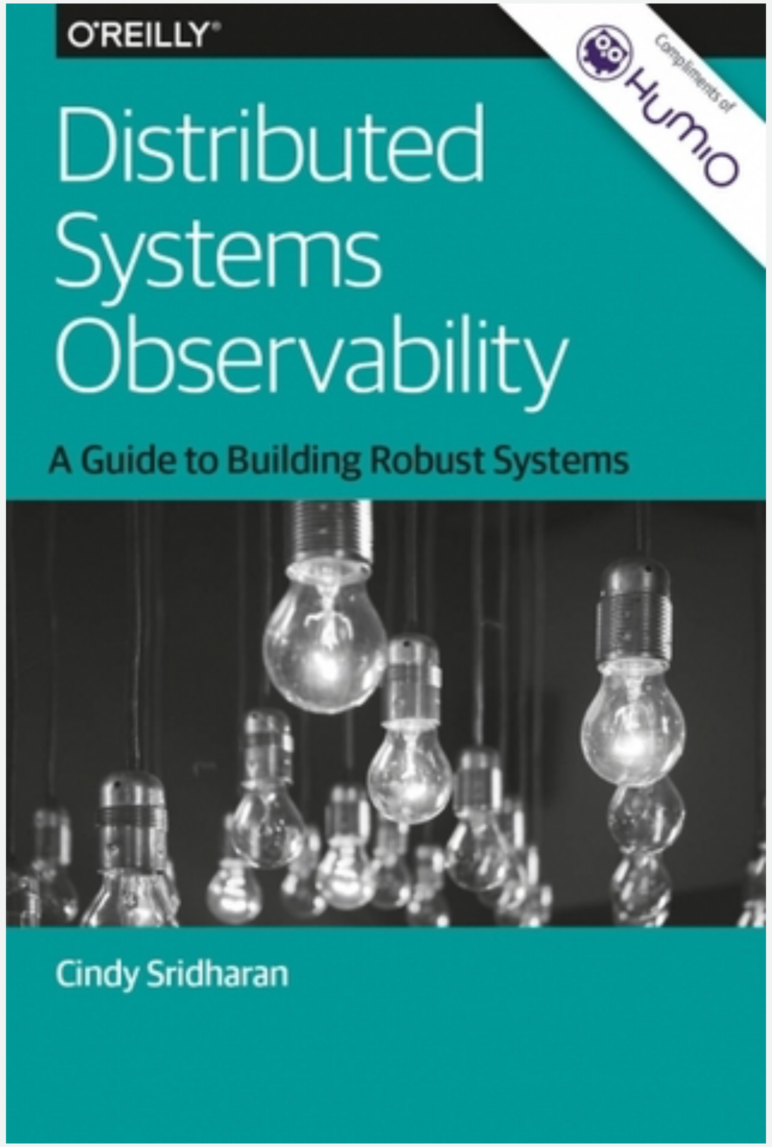


O'REILLY

Chaos Engineering

System Resiliency in Practice

Casey Rosenthal,
Nora Jones &
Nathan Aschbacher



O'REILLY

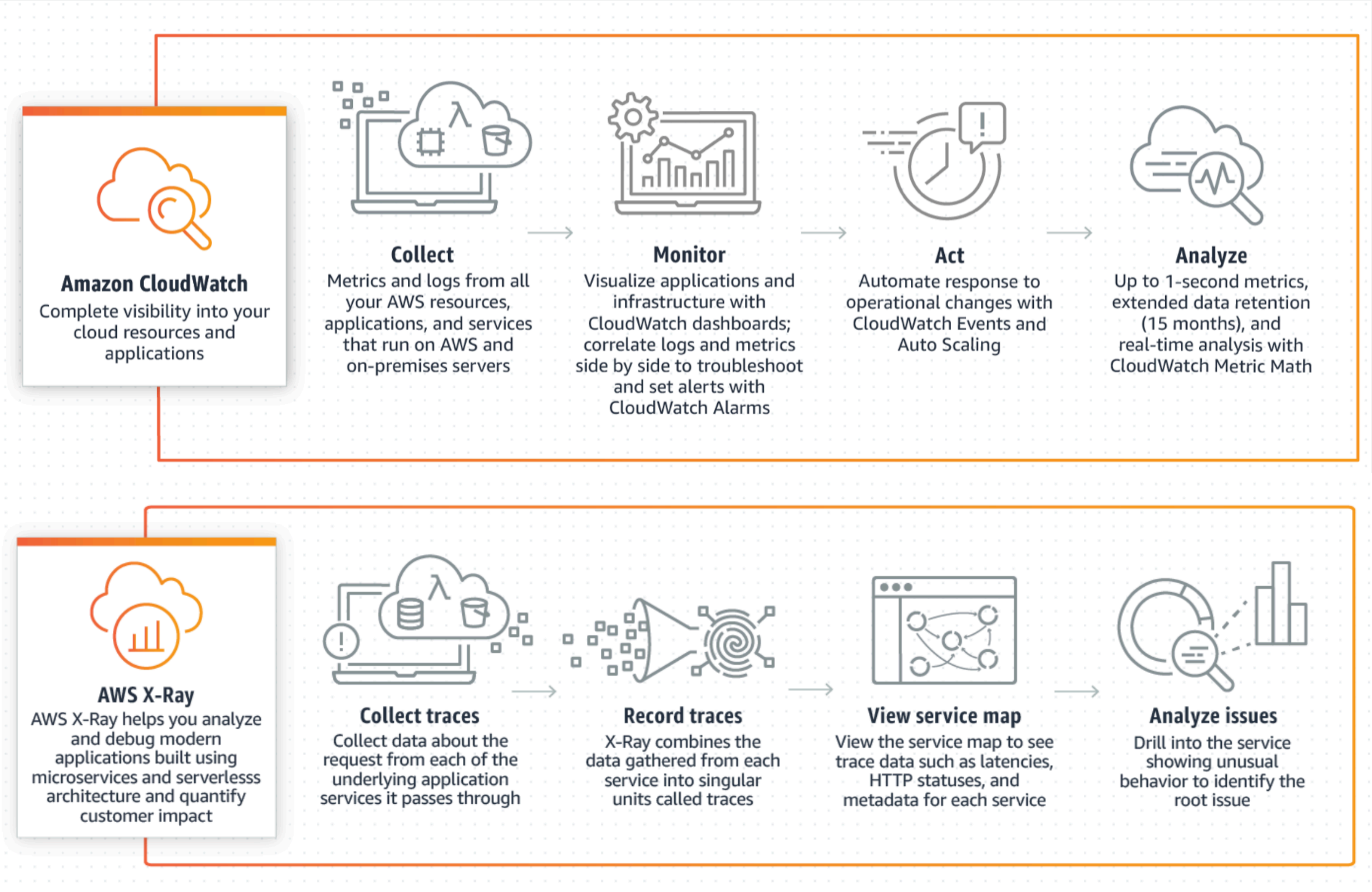
Compliments of Humio

Distributed Systems Observability

A Guide to Building Robust Systems

Cindy Sridharan

X-Ray Vision: Open Loops



Amazon CloudWatch
Complete visibility into your cloud resources and applications

Collect
Metrics and logs from all your AWS resources, applications, and services that run on AWS and on-premises servers

→

Monitor
Visualize applications and infrastructure with CloudWatch dashboards; correlate logs and metrics side by side to troubleshoot and set alerts with CloudWatch Alarms

→

Act
Automate response to operational changes with CloudWatch Events and Auto Scaling

→

Analyze
Up to 1-second metrics, extended data retention (15 months), and real-time analysis with CloudWatch Metric Math

AWS X-Ray
AWS X-Ray helps you analyze and debug modern applications built using microservices and serverlesss architecture and quantify customer impact

Collect traces
Collect data about the request from each of the underlying application services it passes through

→

Record traces
X-Ray combines the data gathered from each service into singular units called traces

→

View service map
View the service map to see trace data such as latencies, HTTP statuses, and metadata for each service

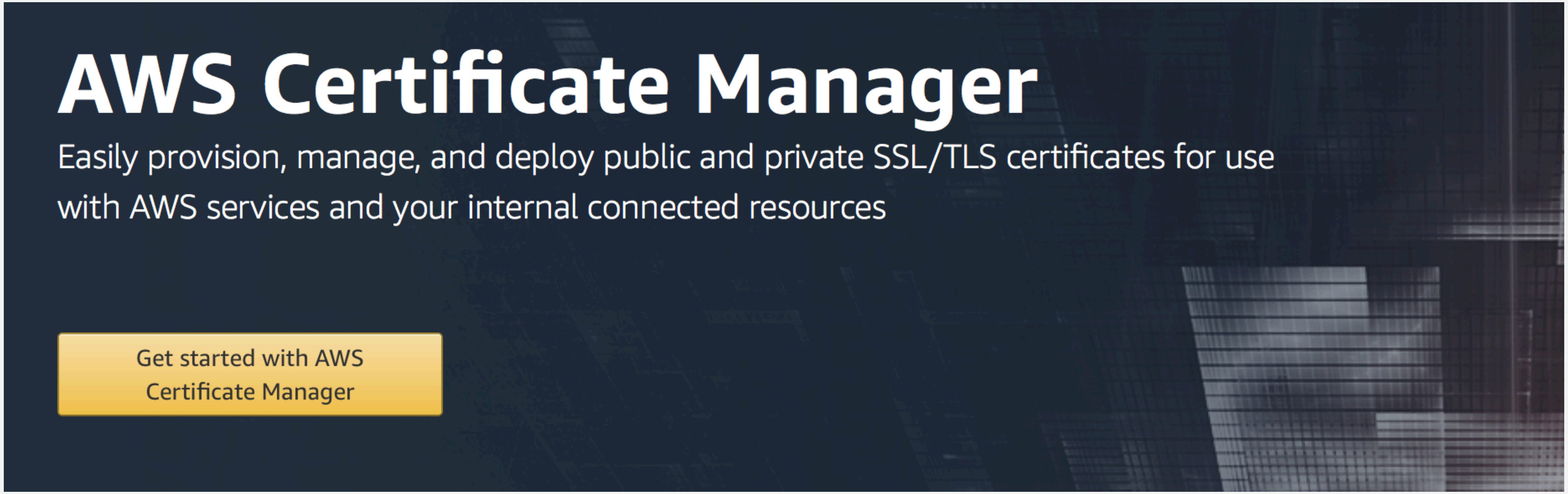
→

Analyze issues
Drill into the service showing unusual behavior to identify the root issue

X-Ray Vision: Open Loops

- Closing loops:
    - Embrace “*Measure first. Then react.*”
    - Measure a lot of things. Check everything you can think to.
    - Avoid infrequent operations – make them more frequent where possible.

X-Ray Vision: Open Loops



AWS Certificate Manager

Easily provision, manage, and deploy public and private SSL/TLS certificates for use with AWS services and your internal connected resources

X-Ray Vision: Open Loops

- Measure-first systems tend to be more naturally declarative

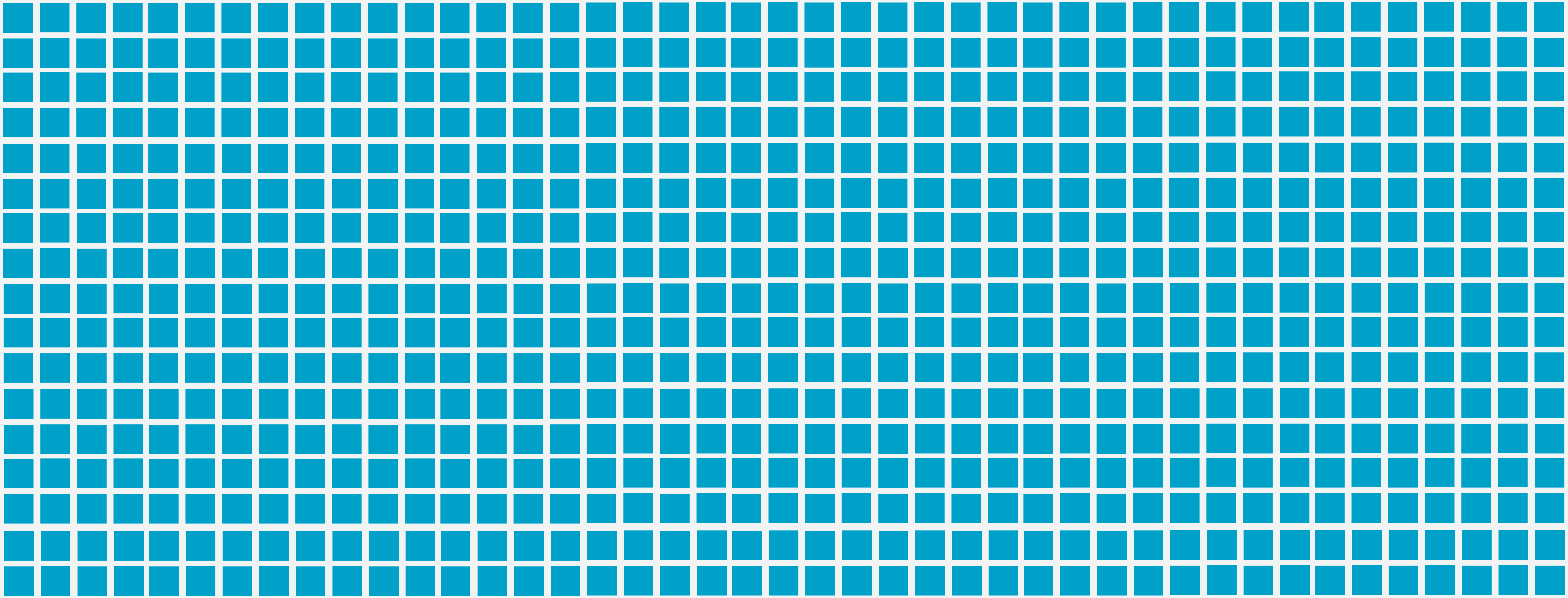
- In general, declarative are easily to formally verify

- TLA+ , F*, CoQ, SAW/Cryptol

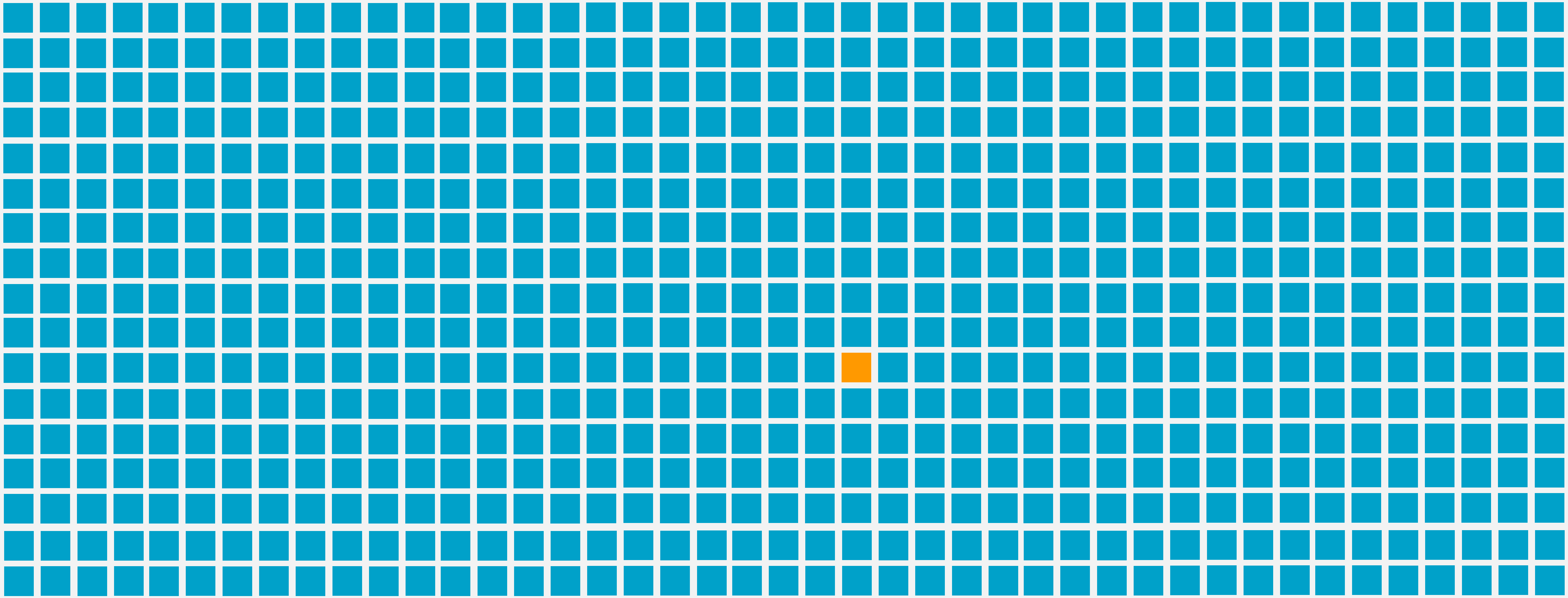
2

X-Ray Vision: Power Laws

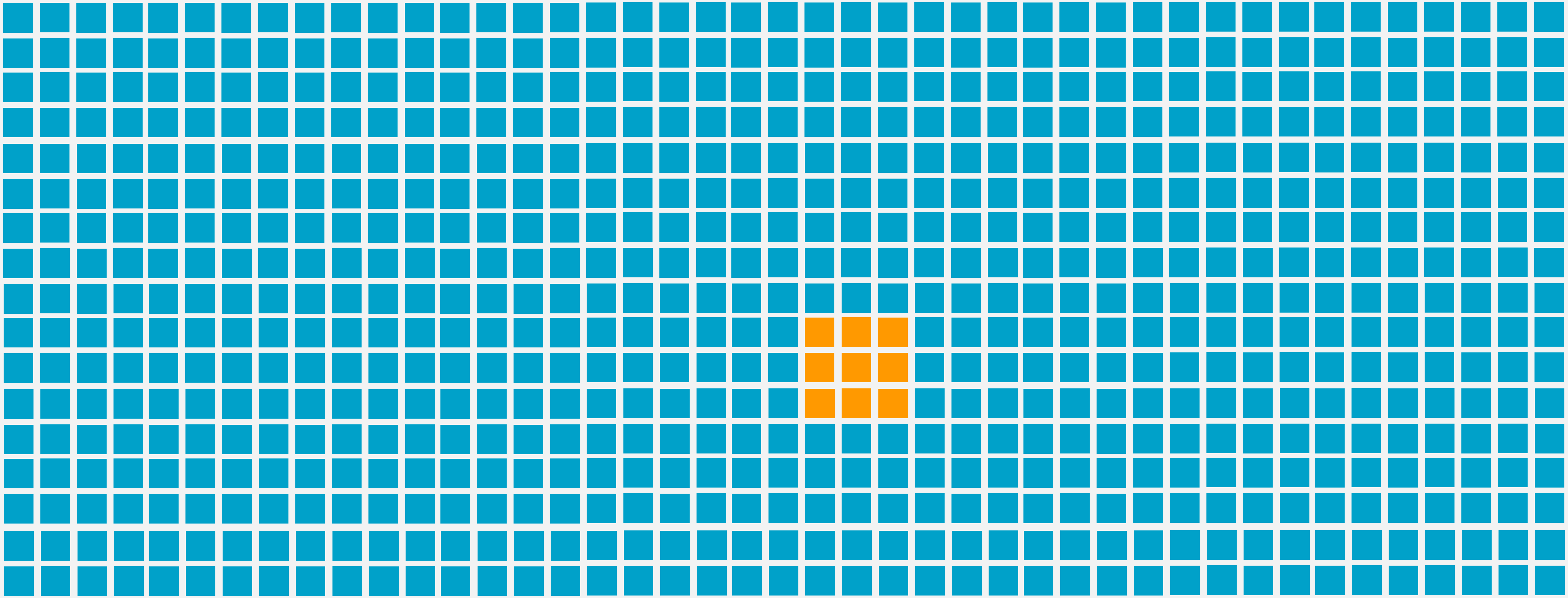
Power Laws



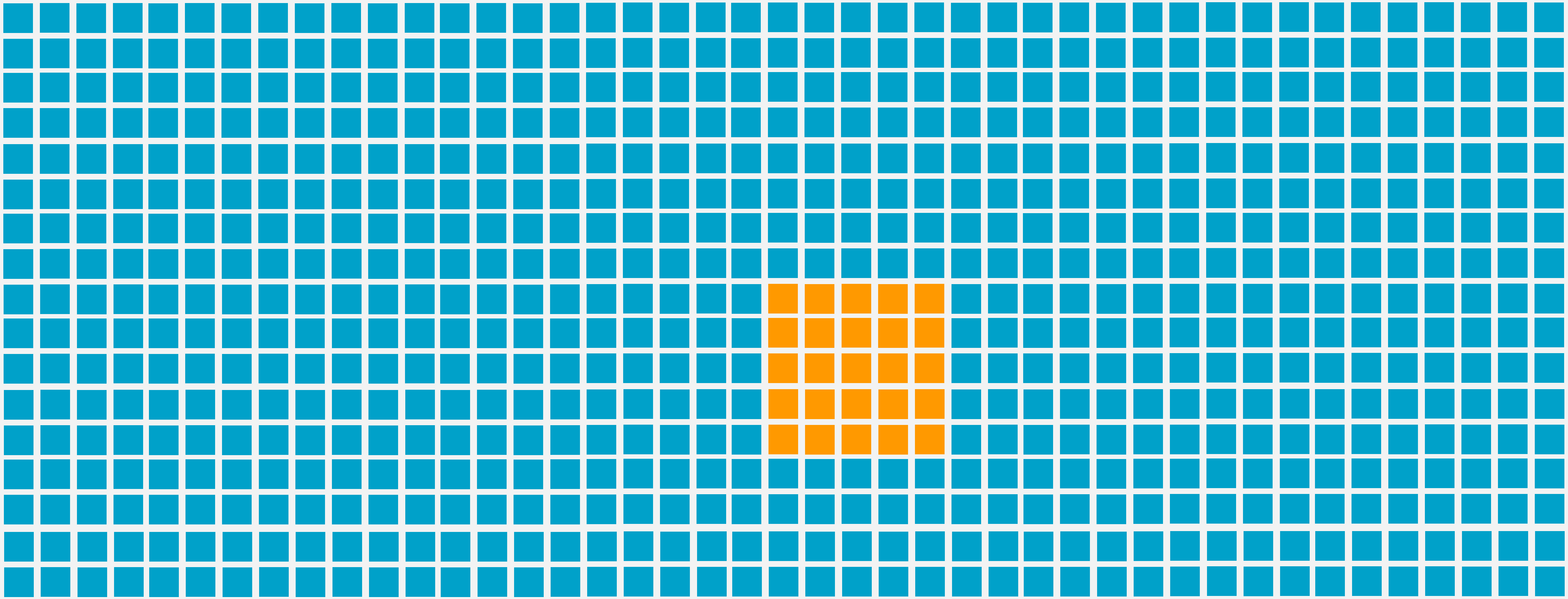
Power Laws



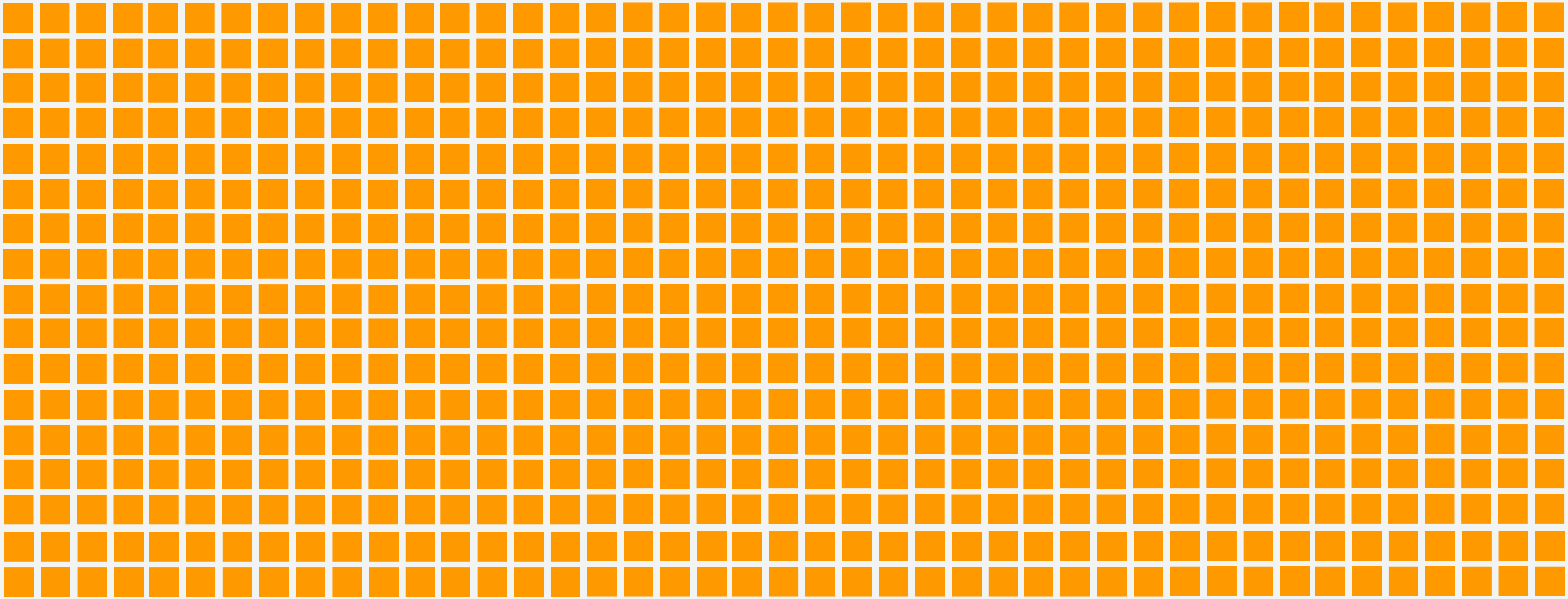
Power Laws



Power Laws



Power Laws



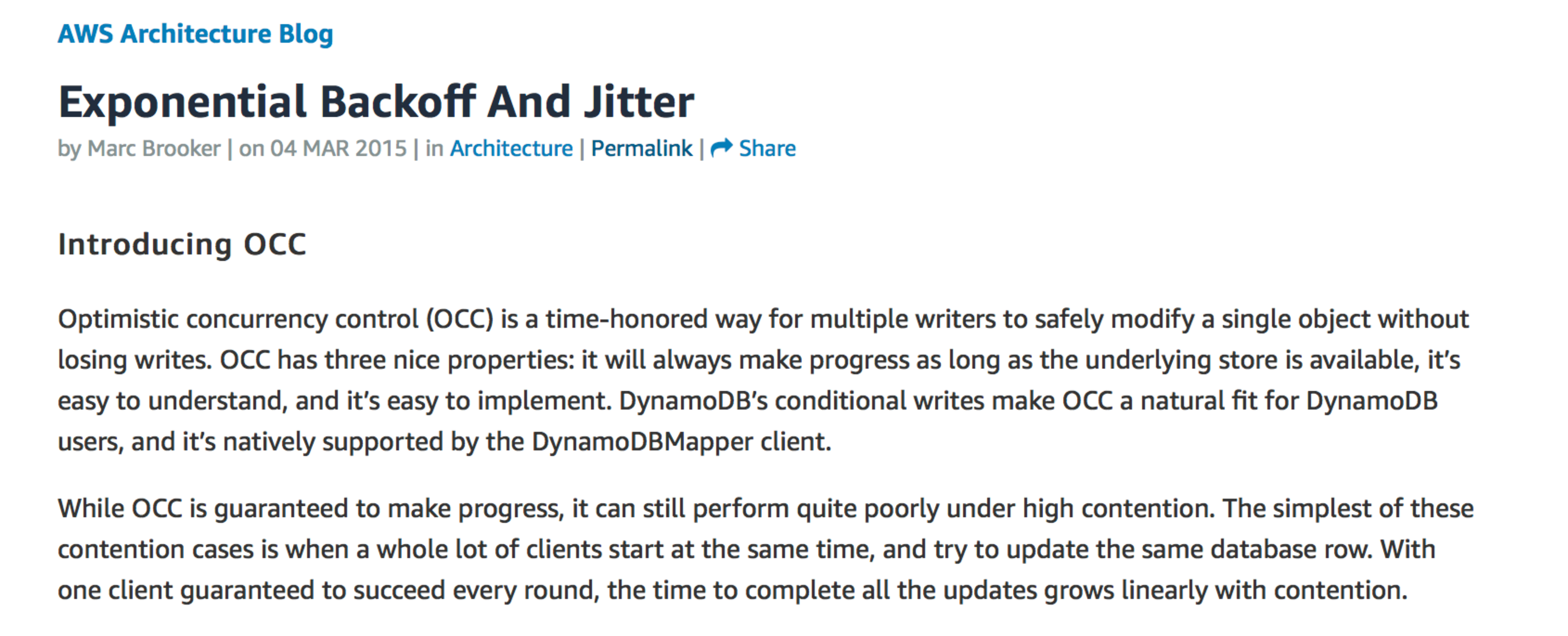
Power Laws

- First, compartmentalize.
- More compartments means relatively smaller blast radius.
- Many real-world control systems reflect this lesson of scale.
- What next?

Power Laws

- Exponential Back-off
    - Brings our own power-law to the table
- Rate-limiters
    - Simple token buckets can be incredibly effective
- Working Backpressure
    - AWS SDK retry strategy = Token buckets + Rate-Limiters + persistent state

Power Laws



AWS Architecture Blog

Exponential Backoff And Jitter

by Marc Brooker | on 04 MAR 2015 | in Architecture | Permalink | ➦ Share

Introducing OCC

Optimistic concurrency control (OCC) is a time-honored way for multiple writers to safely modify a single object without losing writes. OCC has three nice properties: it will always make progress as long as the underlying store is available, it's easy to understand, and it's easy to implement. DynamoDB's conditional writes make OCC a natural fit for DynamoDB users, and it's natively supported by the DynamoDBMapper client.

While OCC is guaranteed to make progress, it can still perform quite poorly under high contention. The simplest of these contention cases is when a whole lot of clients start at the same time, and try to update the same database row. With one client guaranteed to succeed every round, the time to complete all the updates grows linearly with contention.

3

X-Ray Vision: Liveness and Lag

X-Ray Vision: Liveness and Lag

- Operating on old information can be worse than operating on no information
- Simple example: system gets very busy and workflows and metrics pipelines can build up
- Ephemeral “shocks” such as spiky loads or brief outages can end up taking very long to recover

X-Ray Vision: Liveness and Lag

- Strive for O(1) scaling as much as possible

- Provision everything, every time

- Report everything, every time

- Do everything, every time

X-Ray Vision: Liveness and Lag

- If you need to use a bus or queue, think carefully about limits on the size of that queue

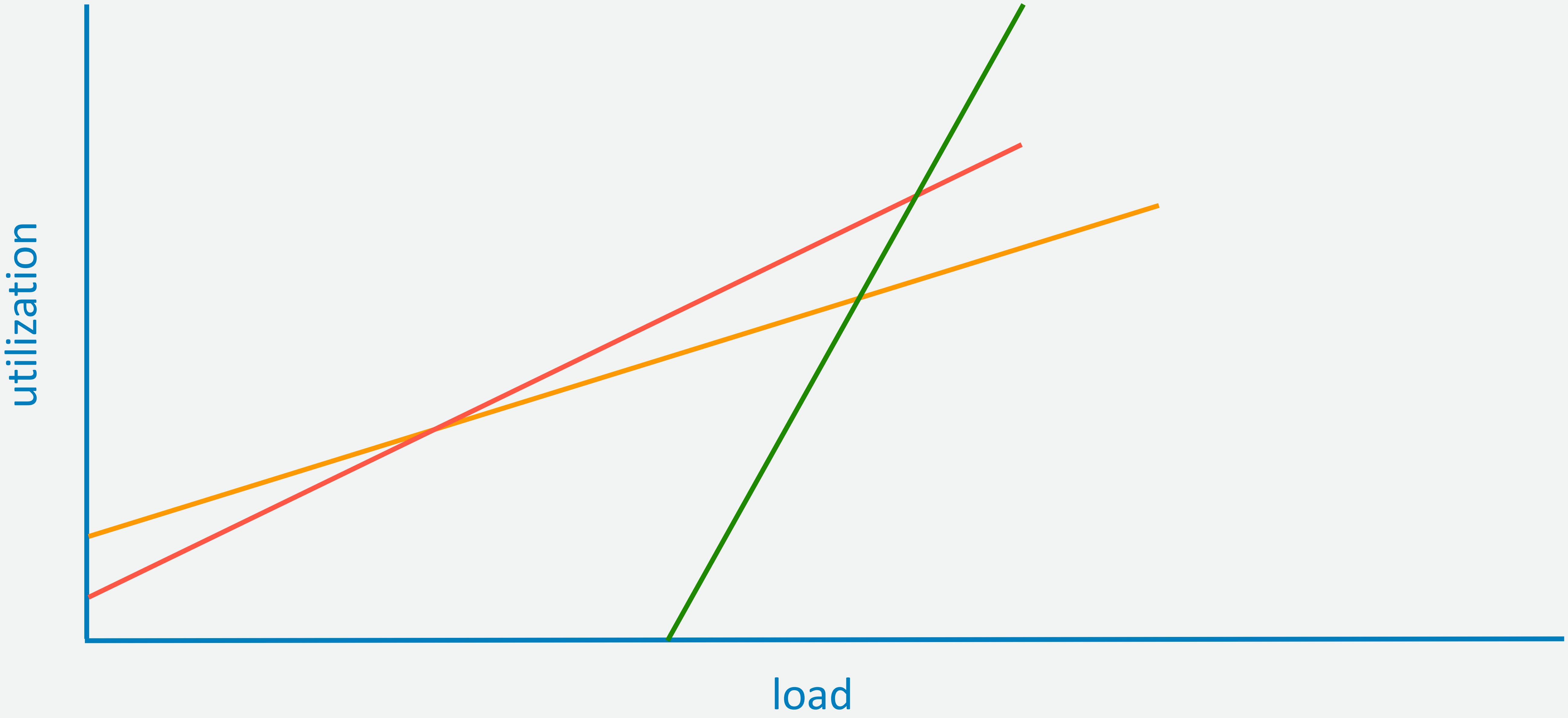
- In general: short queues are safer

- LIFO queues can be a great strategy for information channels
    - Naturally prioritizes recent state
    - Out of order back-fill for any “catching up”

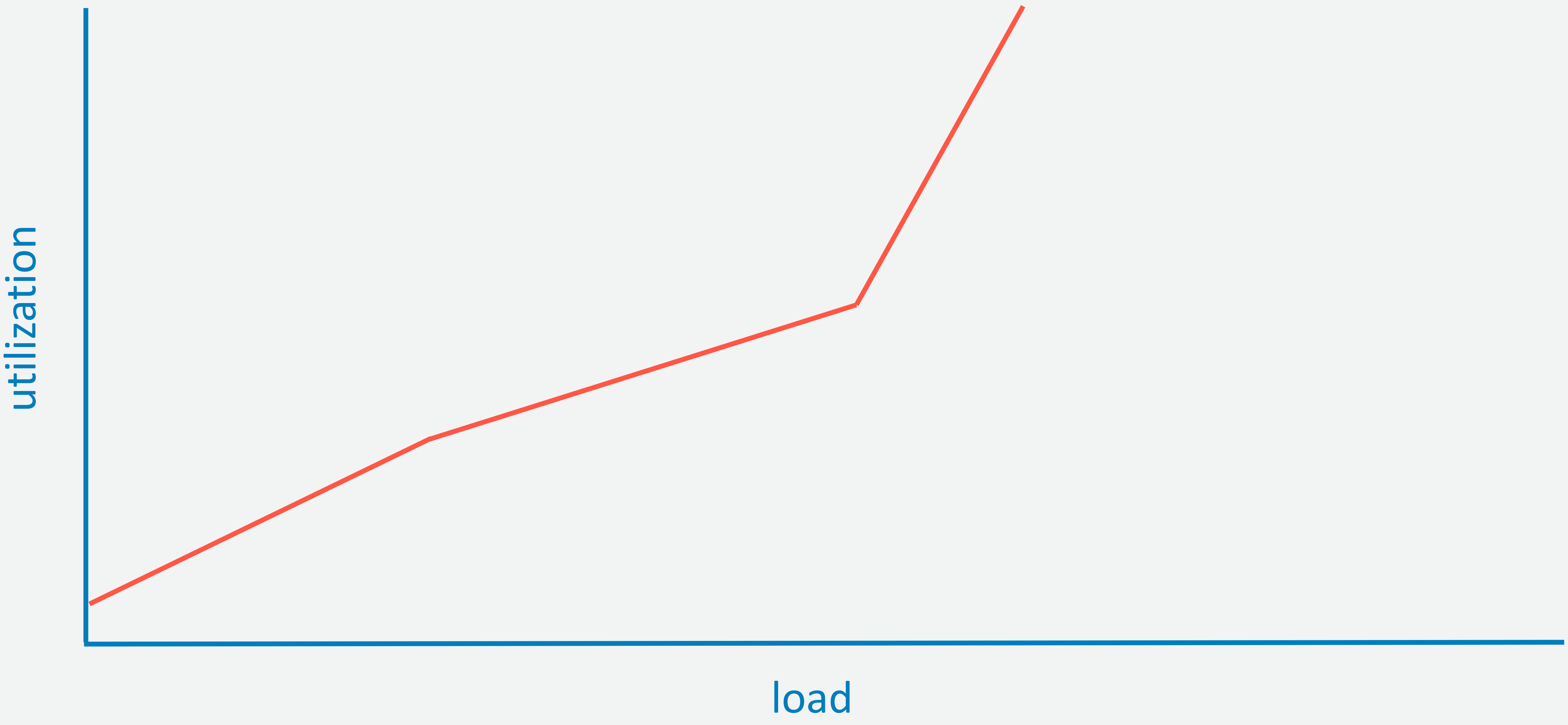
4

X-Ray Vision: False Functions

False functions



False functions



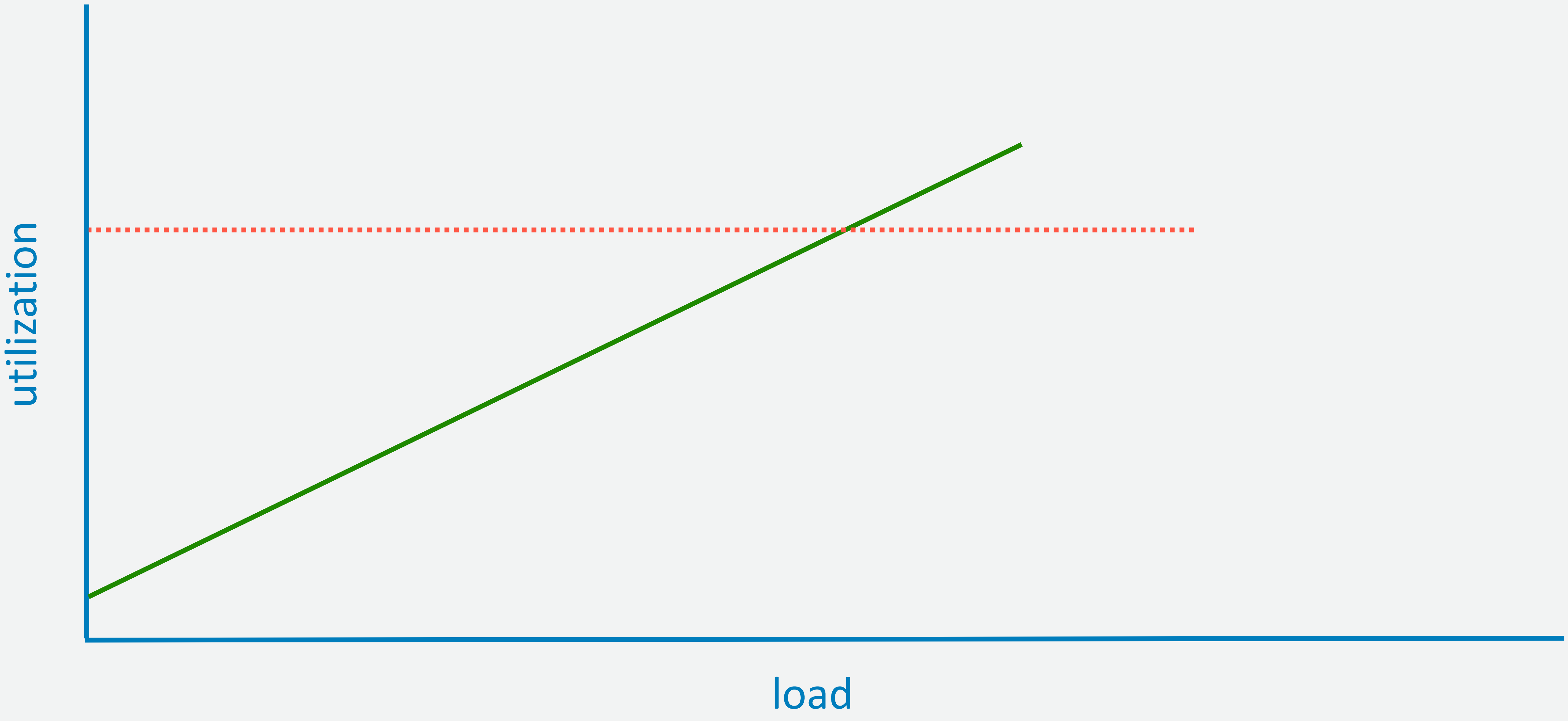
False functions

- Hall of fame false function: Unix load
- Runners-up: system latency, network latency
- Hard to predict Garbage Collector behavior can be confounding
- CPU can be surprisingly effective

5

X-Ray Vision: Edge Triggering

Edge Triggering



Edge Triggering

- Edge Triggering invites modal behavior
- Often the new mode kicks in at a time of high-stress
- Edge Triggering often associated with the “Deliver exactly once” problem
- O.k. for alerting humans but usually an anti-pattern for control systems

Summary

Summary

- “Measure first” and “Integrate feedback” are deeply rewarding concepts
- Right now, this knowledge is highly leveraged
- We can think of distributed systems in terms of control theory, with 100 years of powerful mental models available
- Control Theory can help us formally analyze the stability of systems

Q&A

Colm MacCárthaigh

Thank you!