INNOVATE

FOR EVERY APPLICATION EDITION

25 August, 2022

Resiliency and availability design patterns for the cloud

Chandra Munibudha
Principal Solutions Architect
AISPL

Today's agenda

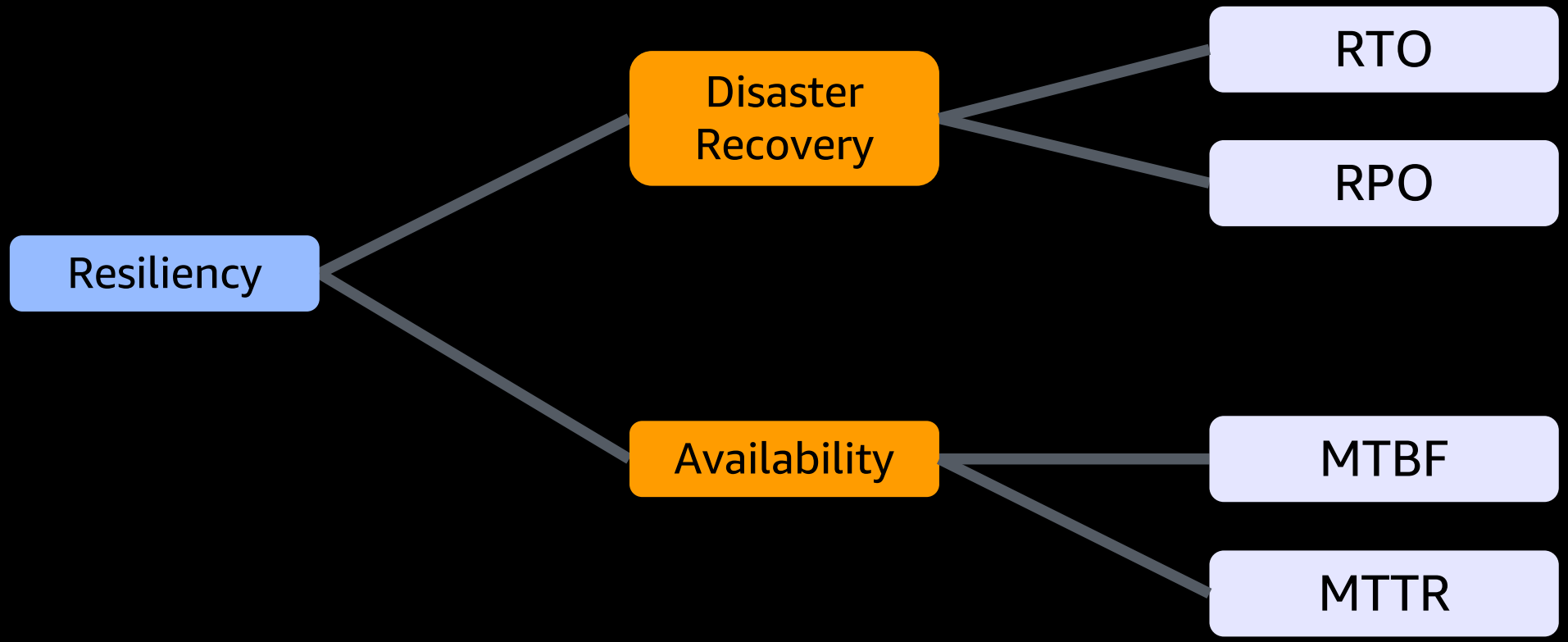
- Scope of Resiliency
- Resilient Architectures

  Timeouts, Retries with jitter
  Load Shedding
  Constant Work
  Static Stability
  Shuffle Sharding

What is resiliency?

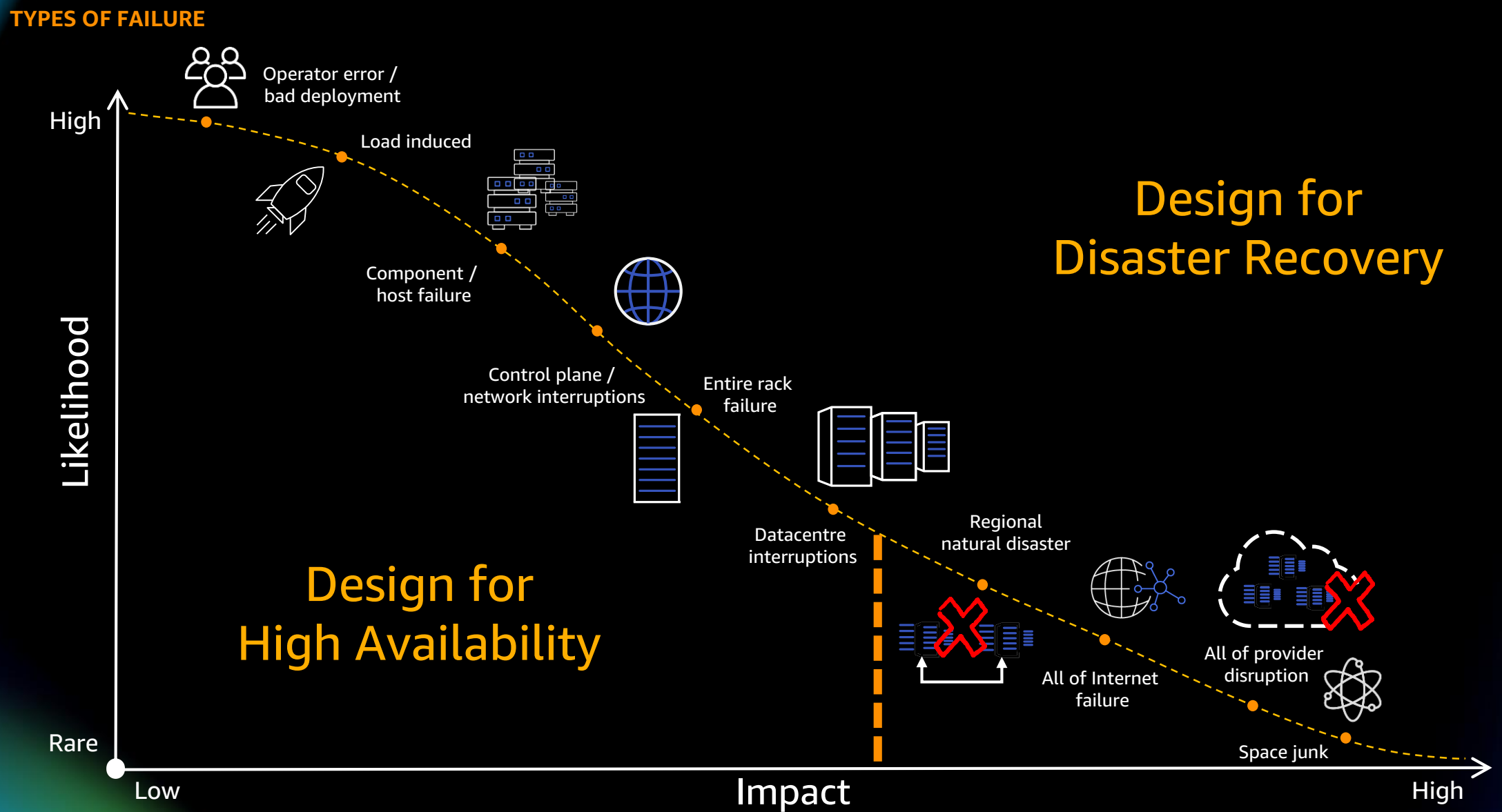
"**Resilience** is the ability of a system to **adapt** or **keep working** when challenges occur"

AWS whitepaper definition



- RTO (Recovery Time)
- RPO (Recovery Point)
- MTBF (Mean Time Between Failures)
- MTTR (Mean Time to Recover)

Categories of failure



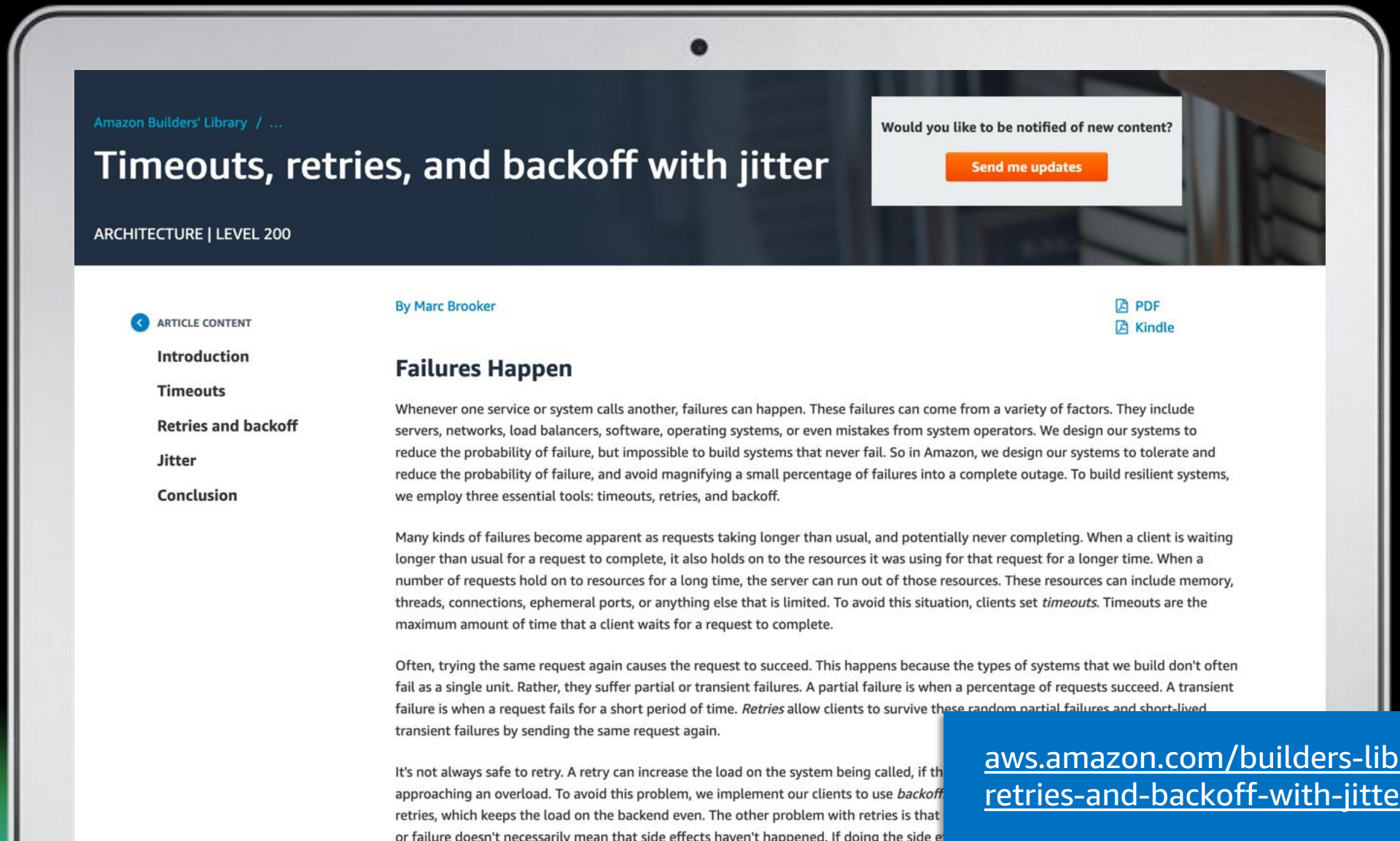
Resilient Design Patterns

Client-side patterns

- Timeouts - maximum amount of time that a client (or calling dependency) waits for a request to complete.
- *Retries* - survive random partial failures and short-lived transient failures
- Use *backoff* – *Increase* the time between subsequent retries, which keeps the load on the backend even
- *Jitter* - This is a random amount of time before retrying a request to help prevent large simultaneous bursts

AWS Resilience Workshop: https://tinyurl.com/bhjbsk67

Timeouts, retries, and backoff with jitter



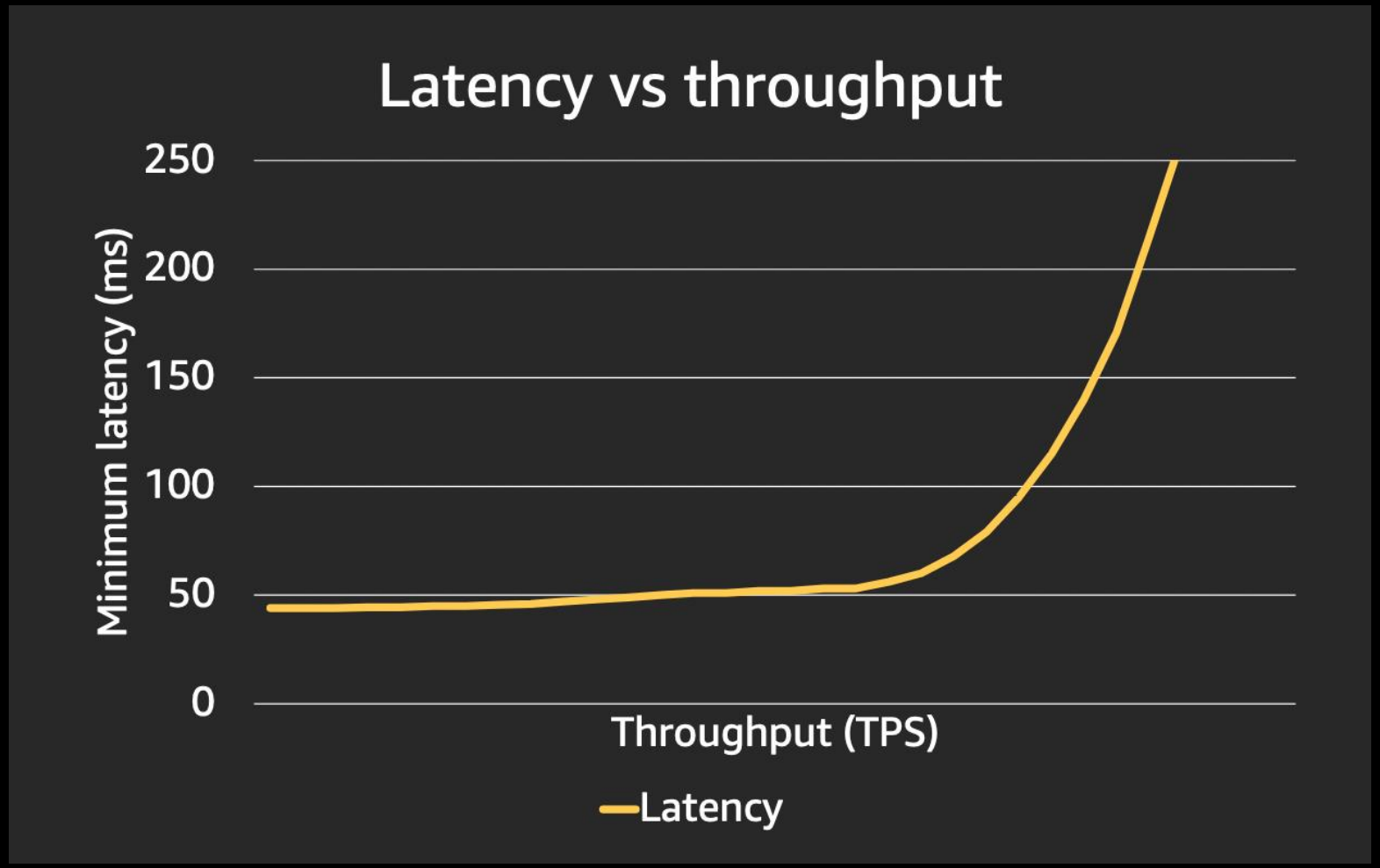
aws.amazon.com/builders-library/timeouts-retries-and-backoff-with-jitter/

Server/Backend patterns

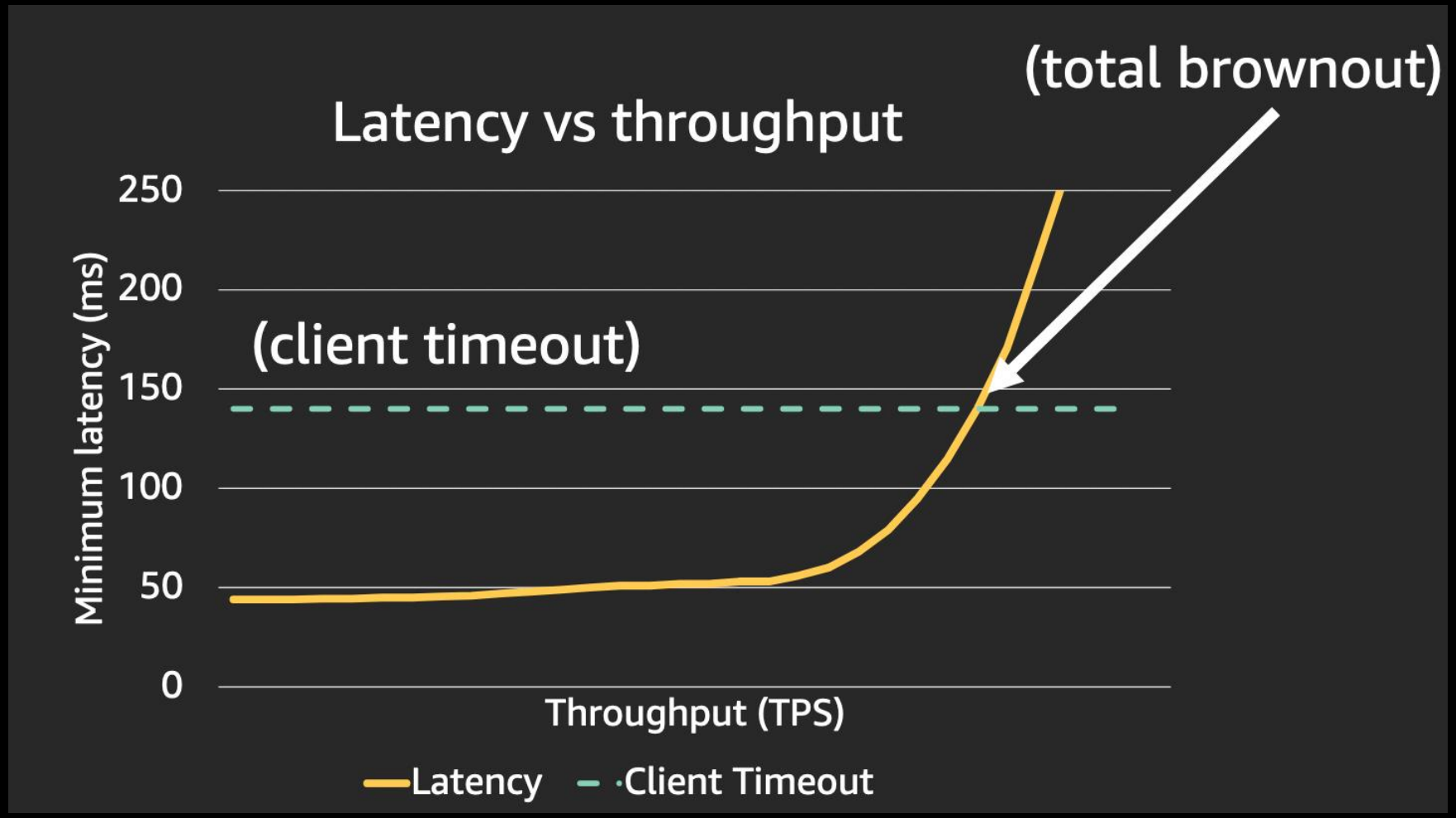
- Load Shedding – avoid brownout by rejecting excess load.
- Constant Work - survive random partial failures and short-lived transient failures.
- Static Stability – the overall system keeps working even when a dependency becomes impaired.
- Shuffle Sharding – *Isolating customers/resources to virtual shards and reduce overall impact of bad requests.*

Load Shedding

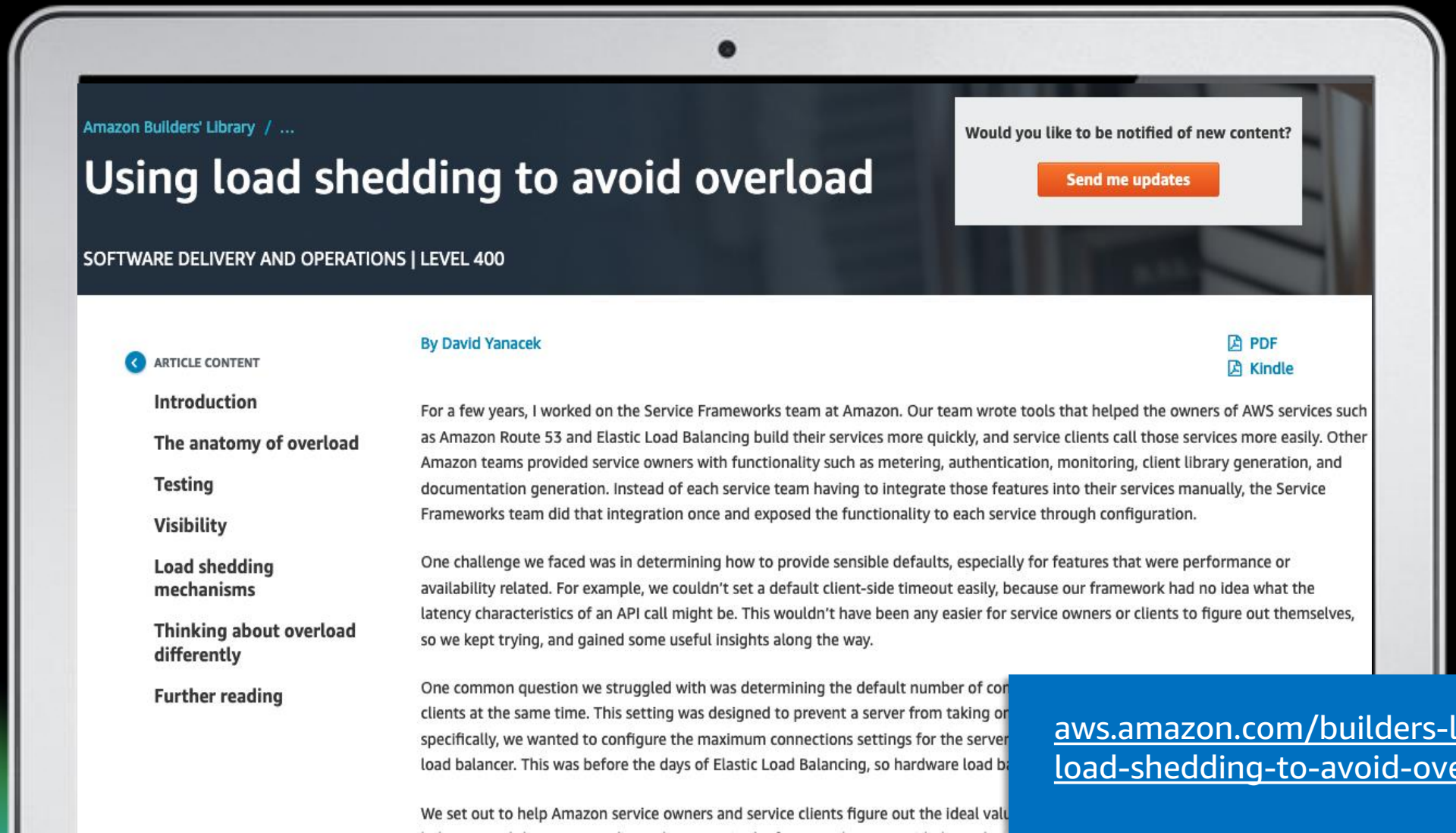
Load Shedding



Load Shedding



Load Shedding



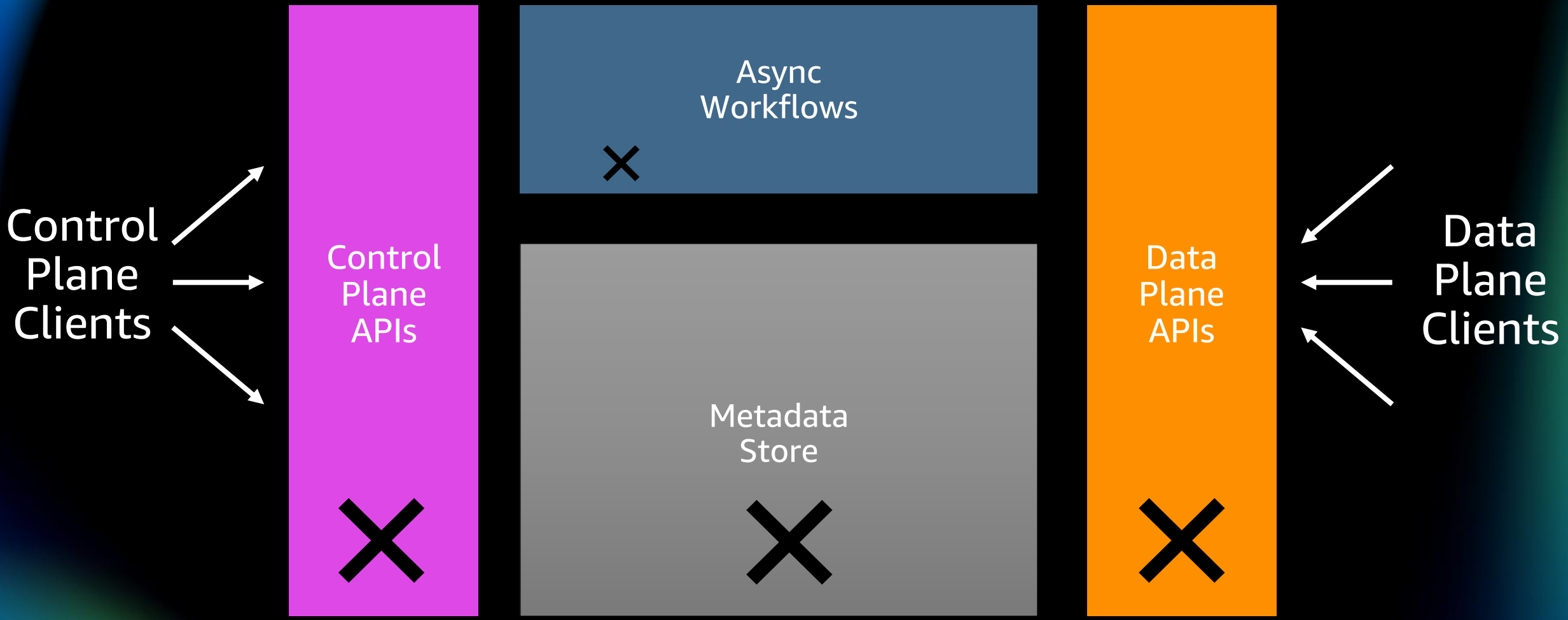
aws.amazon.com/builders-library/using-load-shedding-to-avoid-overload/

Static stability

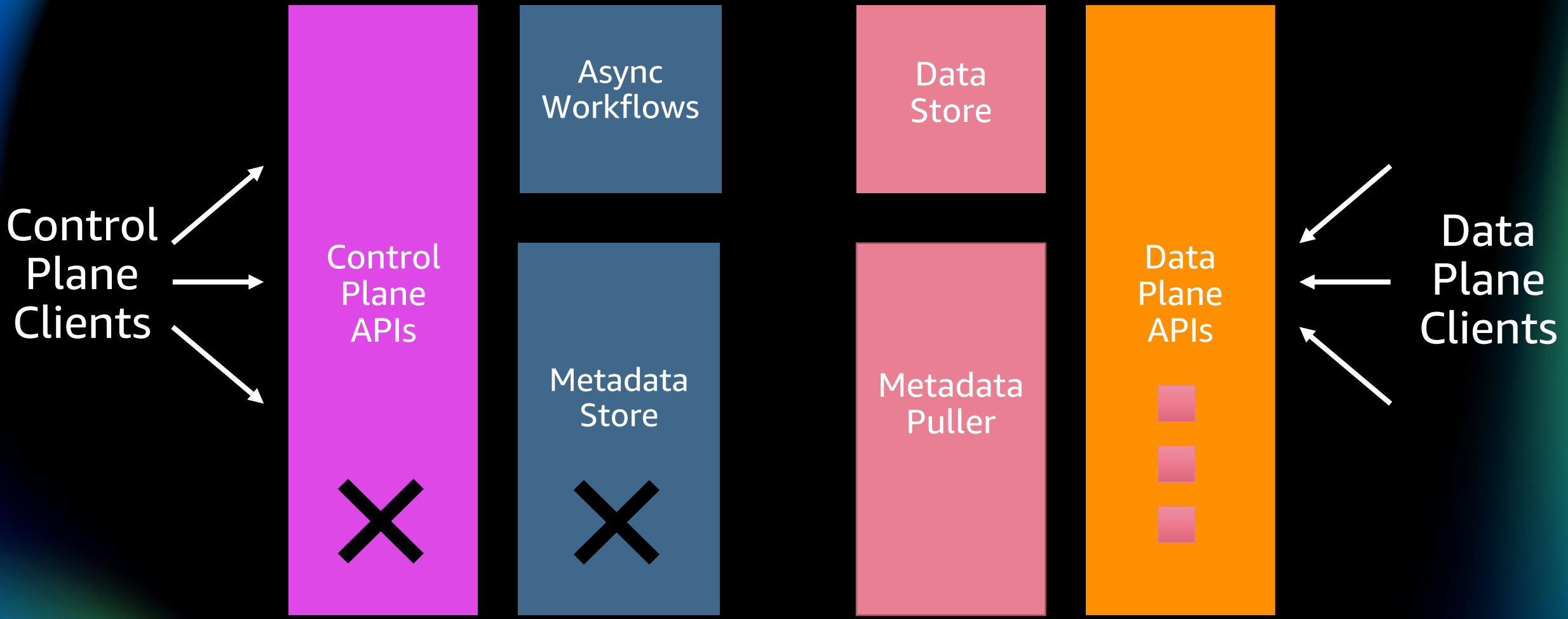
Control Plane and Data Plane

MODULAR SEPARATION CREATES STATIC STABILITY

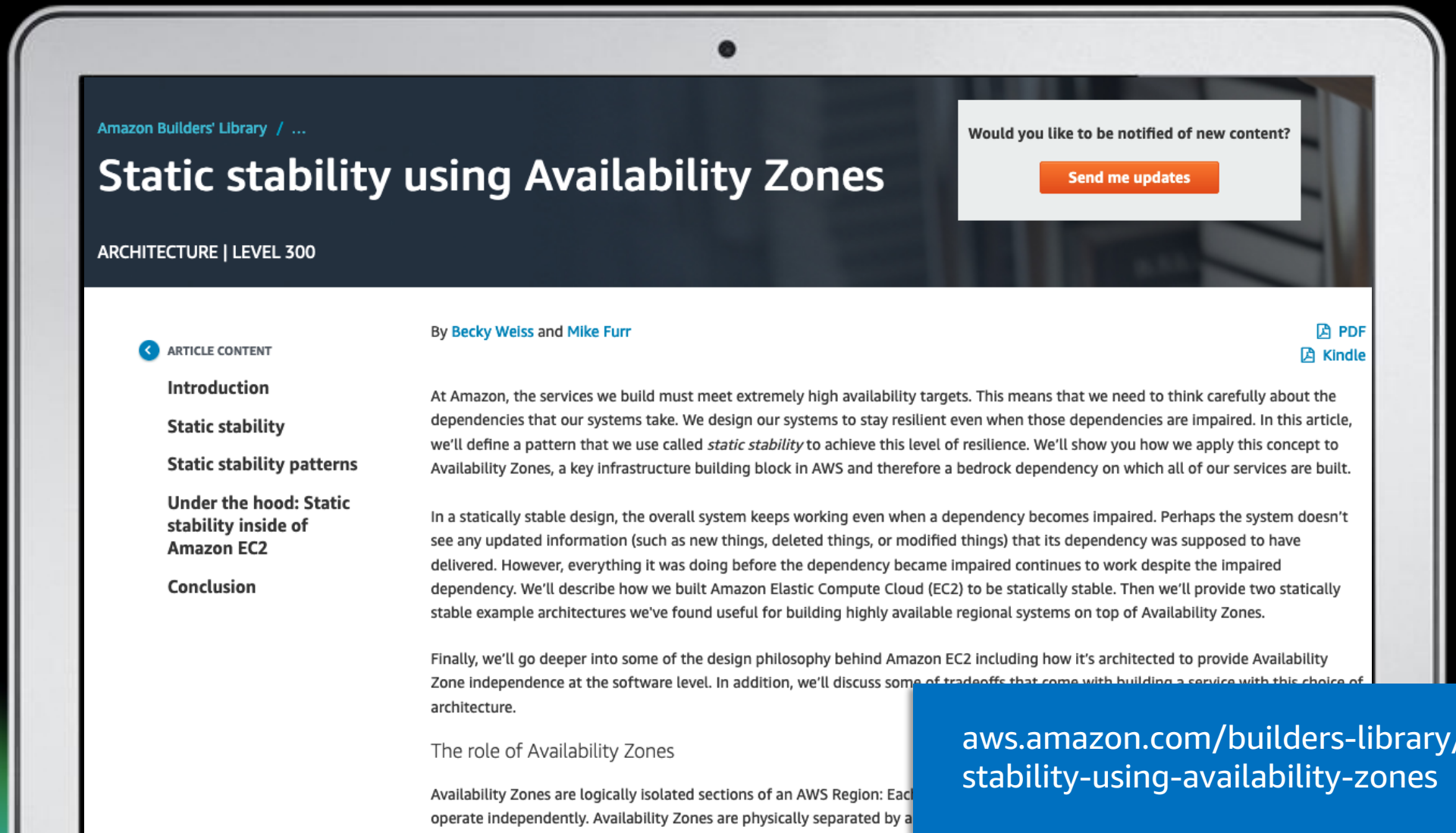
Static Stability



Static Stability



Static Stability



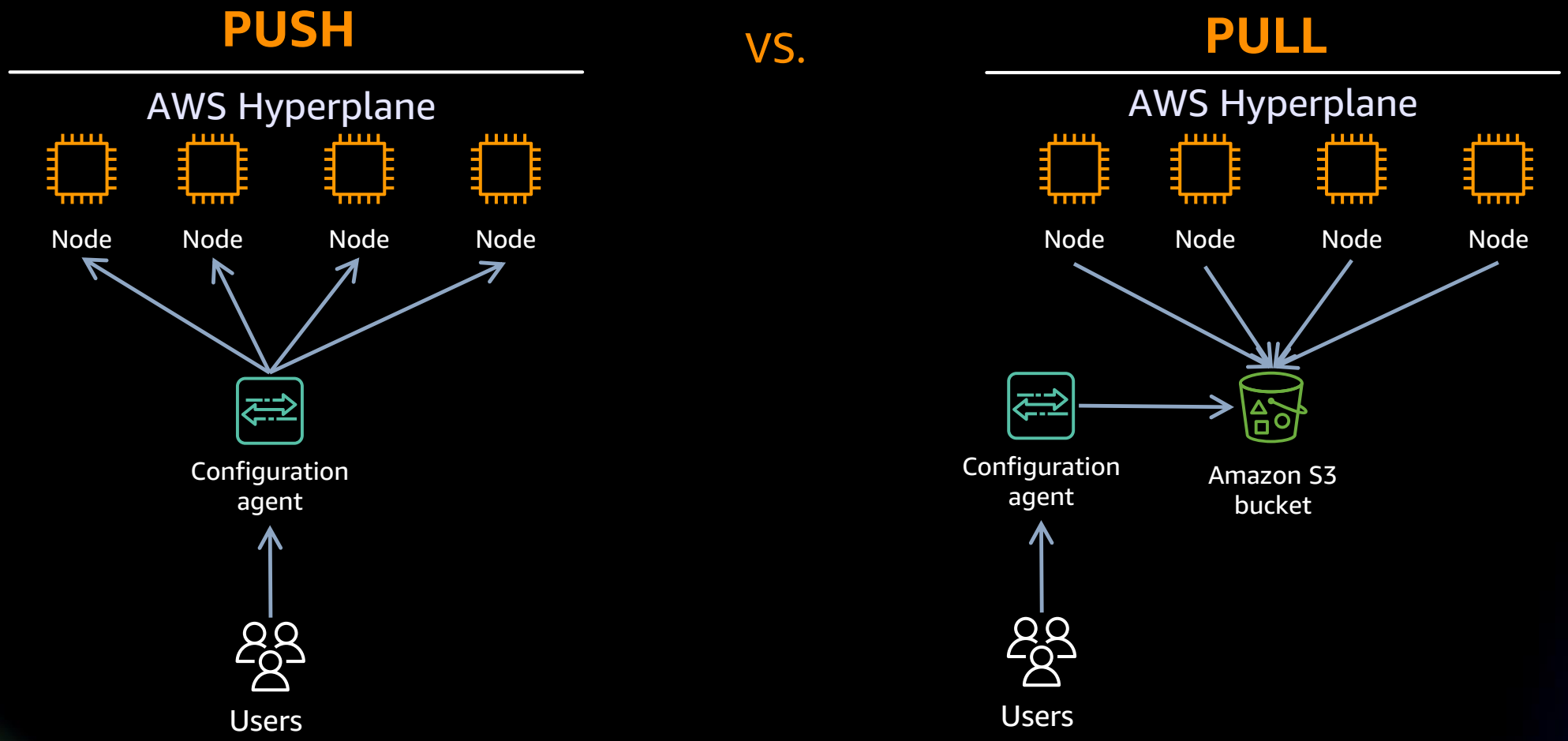
aws.amazon.com/builders-library/static-stability-using-availability-zones

Simple designs and constant work

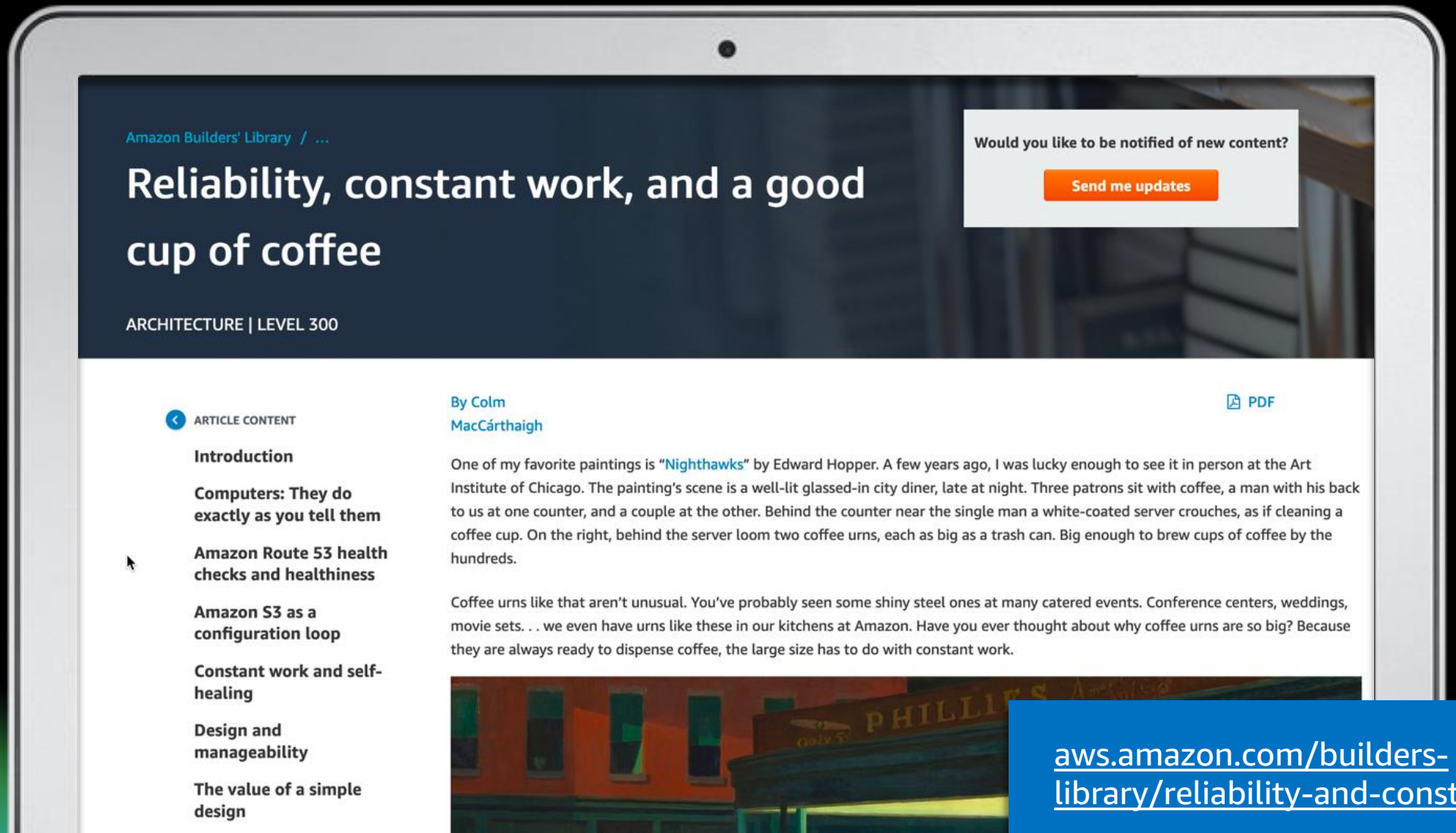
Simple designs and constant work

- Risk is often proportionate to rates of change in systems
- Example: a spike in load can slow down a system, which can cause knock-on and cascading effects
- Reducing dynamism in systems is a great way to make them simpler
- A counter-intuitive solution is to run the system at “maximum” load all the time, every time

Simple designs and constant work



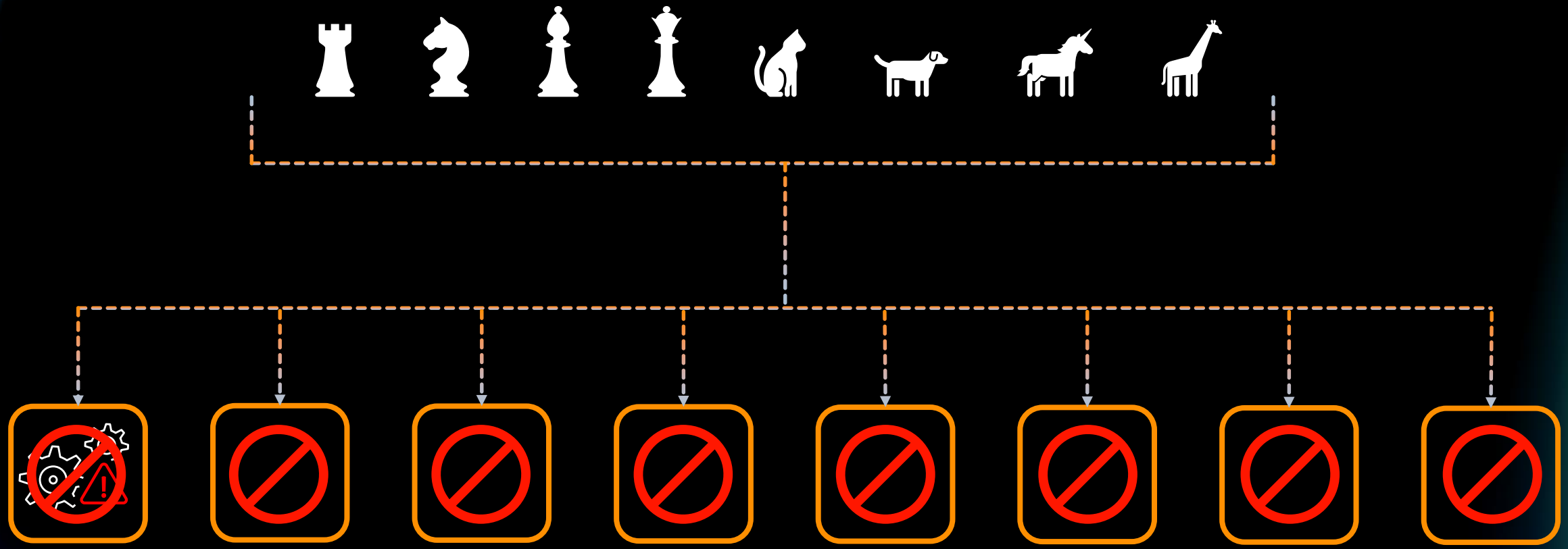
Simple designs and constant work



aws.amazon.com/builders-library/reliability-and-constant-work

Shuffle Sharding

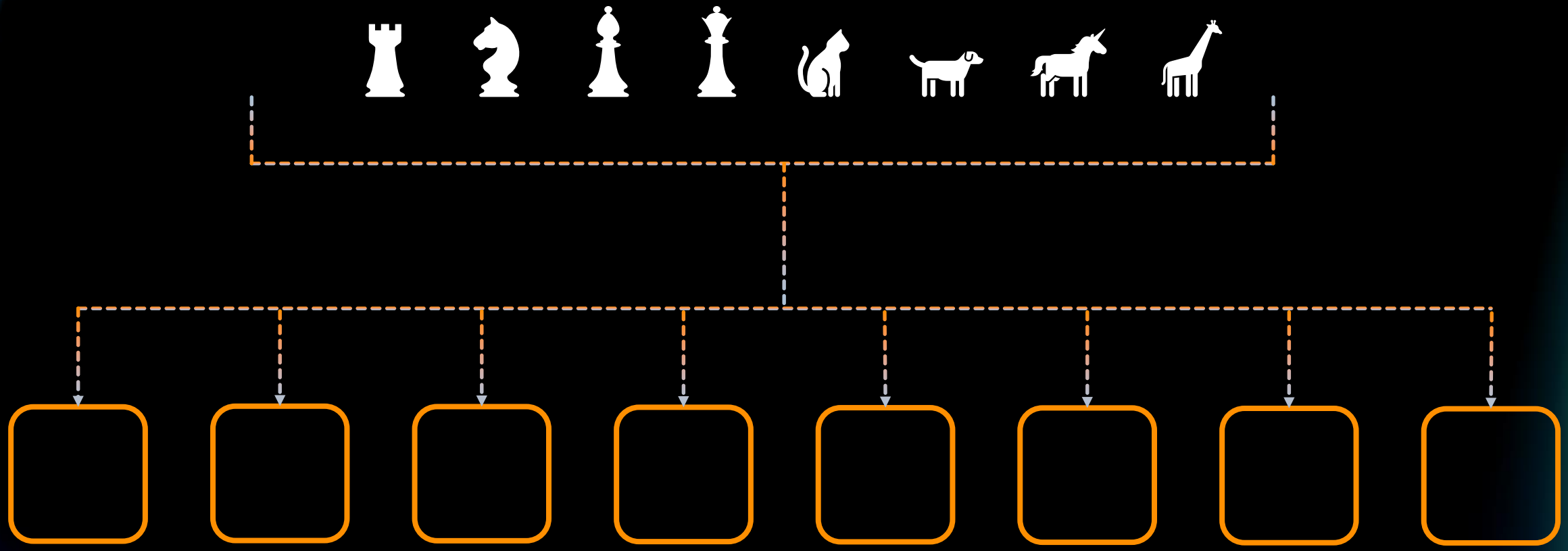
Traditional architecture



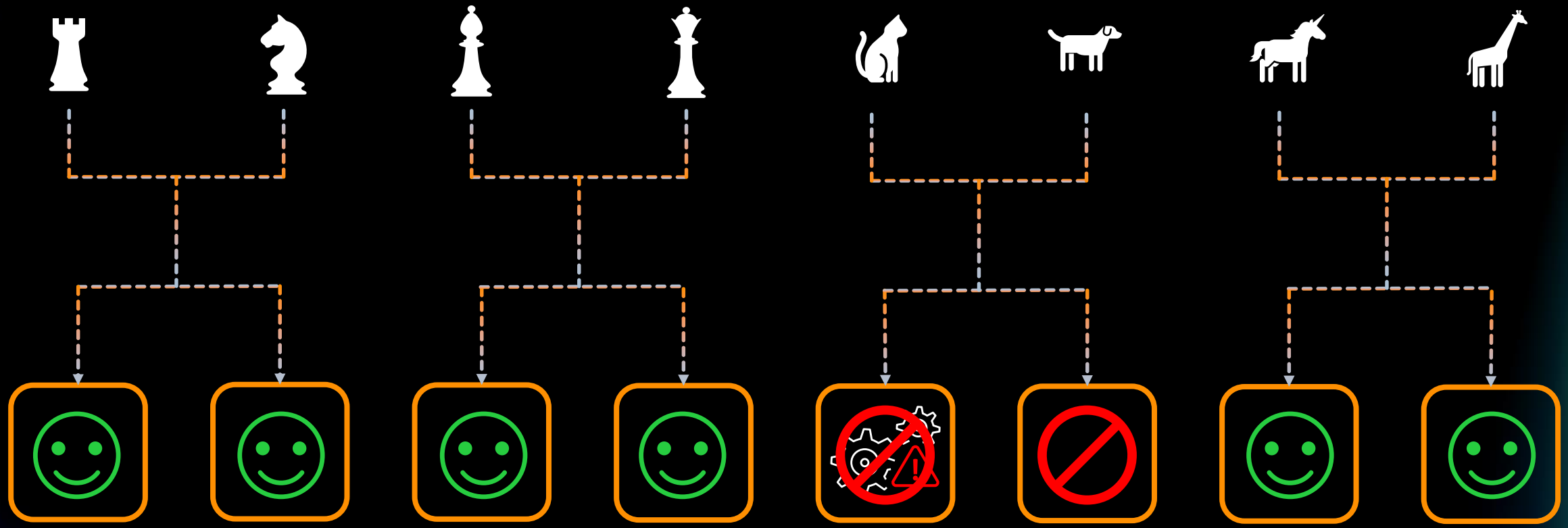
Scope of impact = All customers

Combining Partitioning and Replication

Sharding

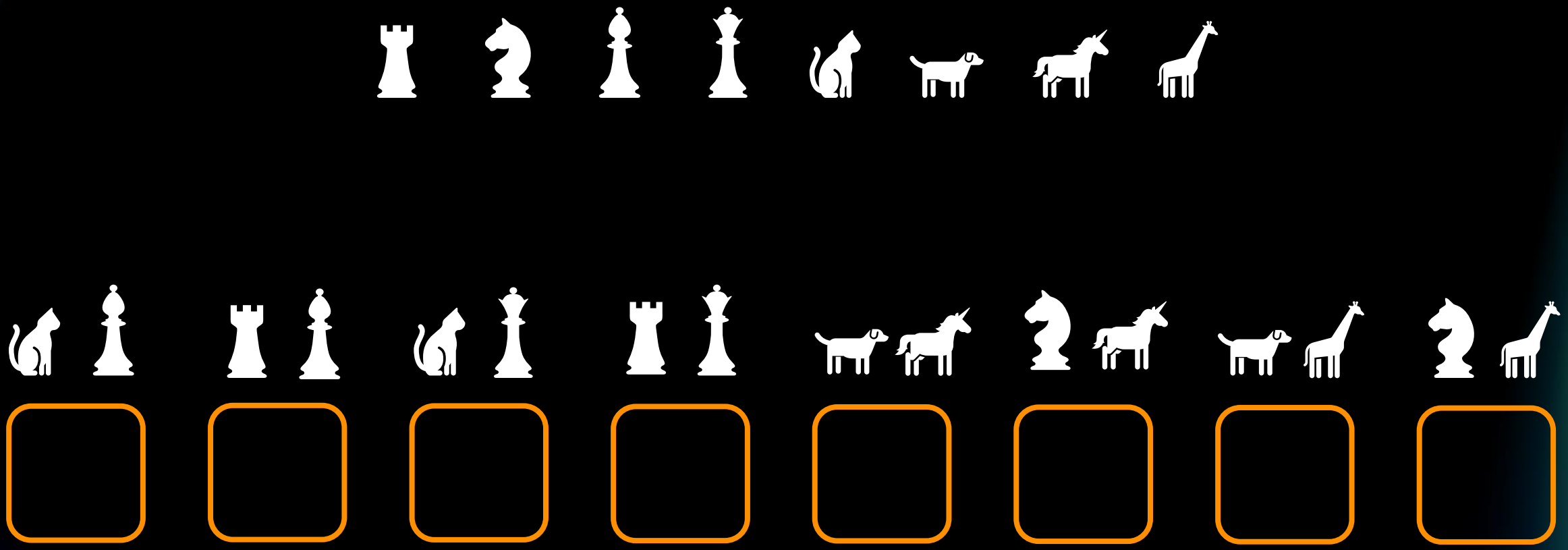


Sharding



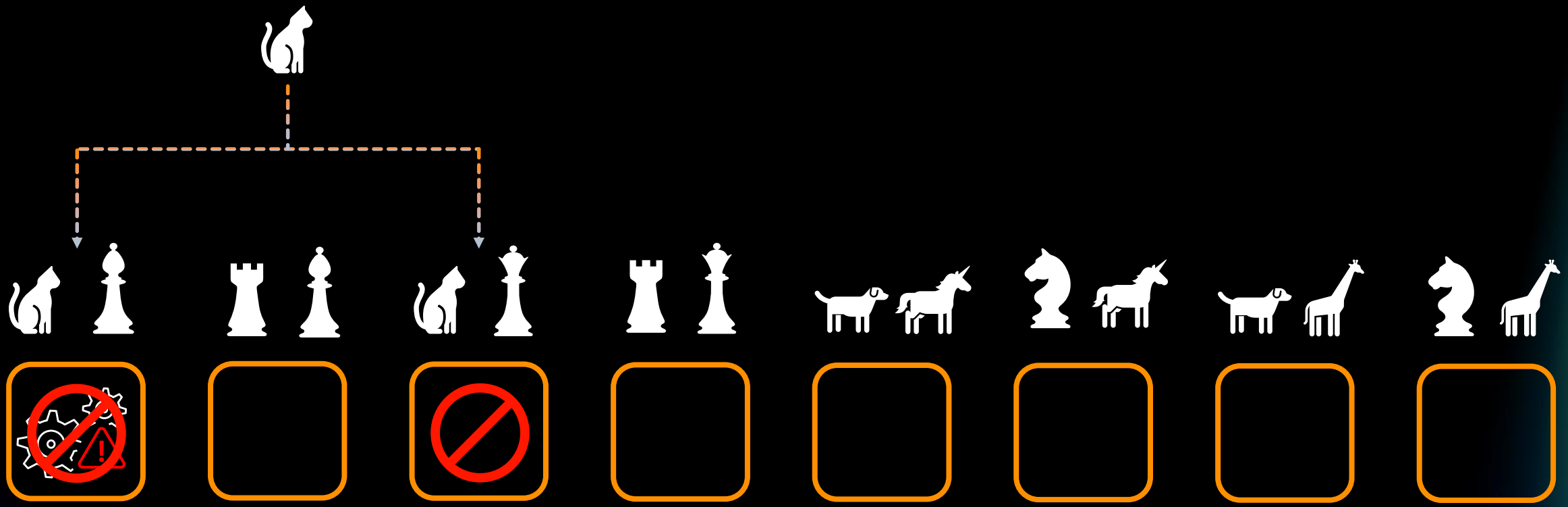
$$\text{Scope of impact} = \frac{\text{Customers}}{\text{Shards}}$$

Shuffle sharding

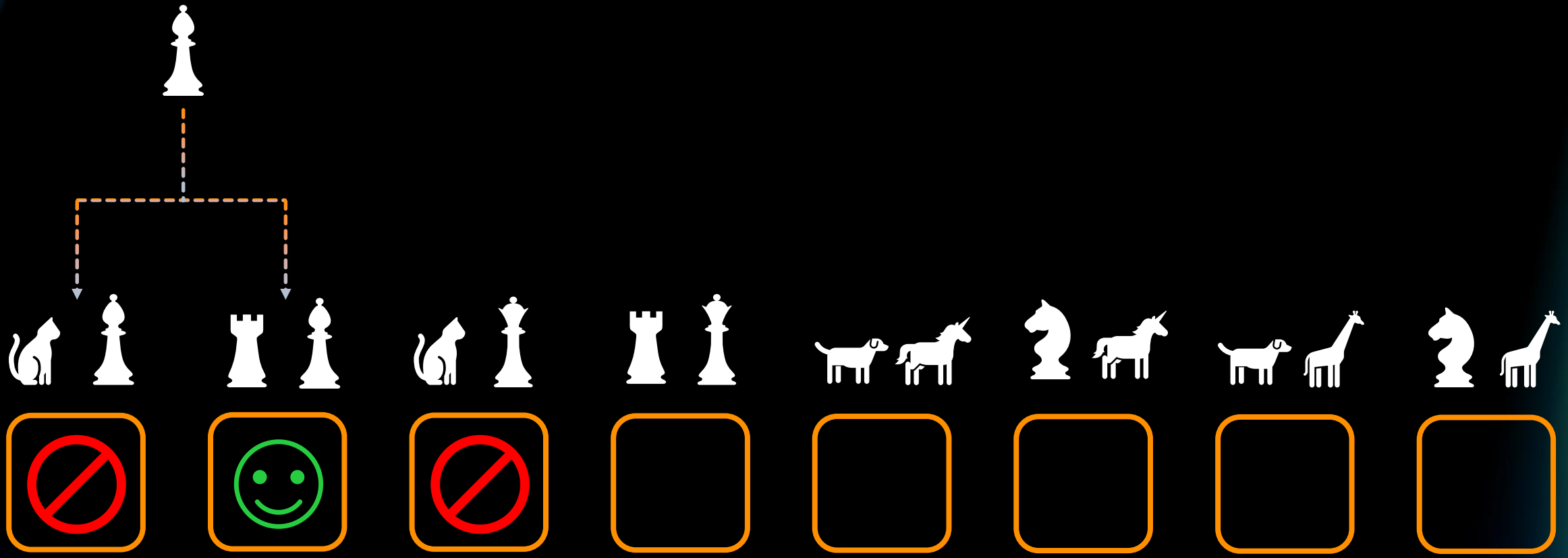


By choosing two instances from eight there are 56 potential shuffle shards, much more than the four simple shards we had before.

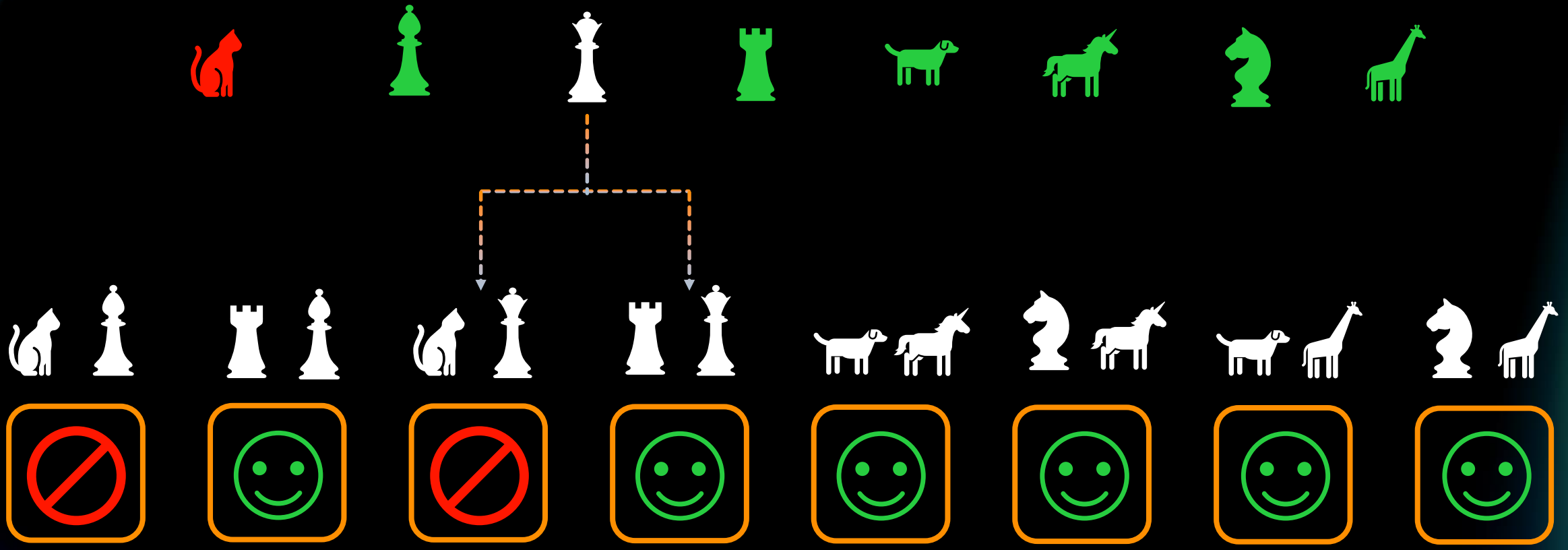
Shuffle sharding



Shuffle sharding



Shuffle sharding



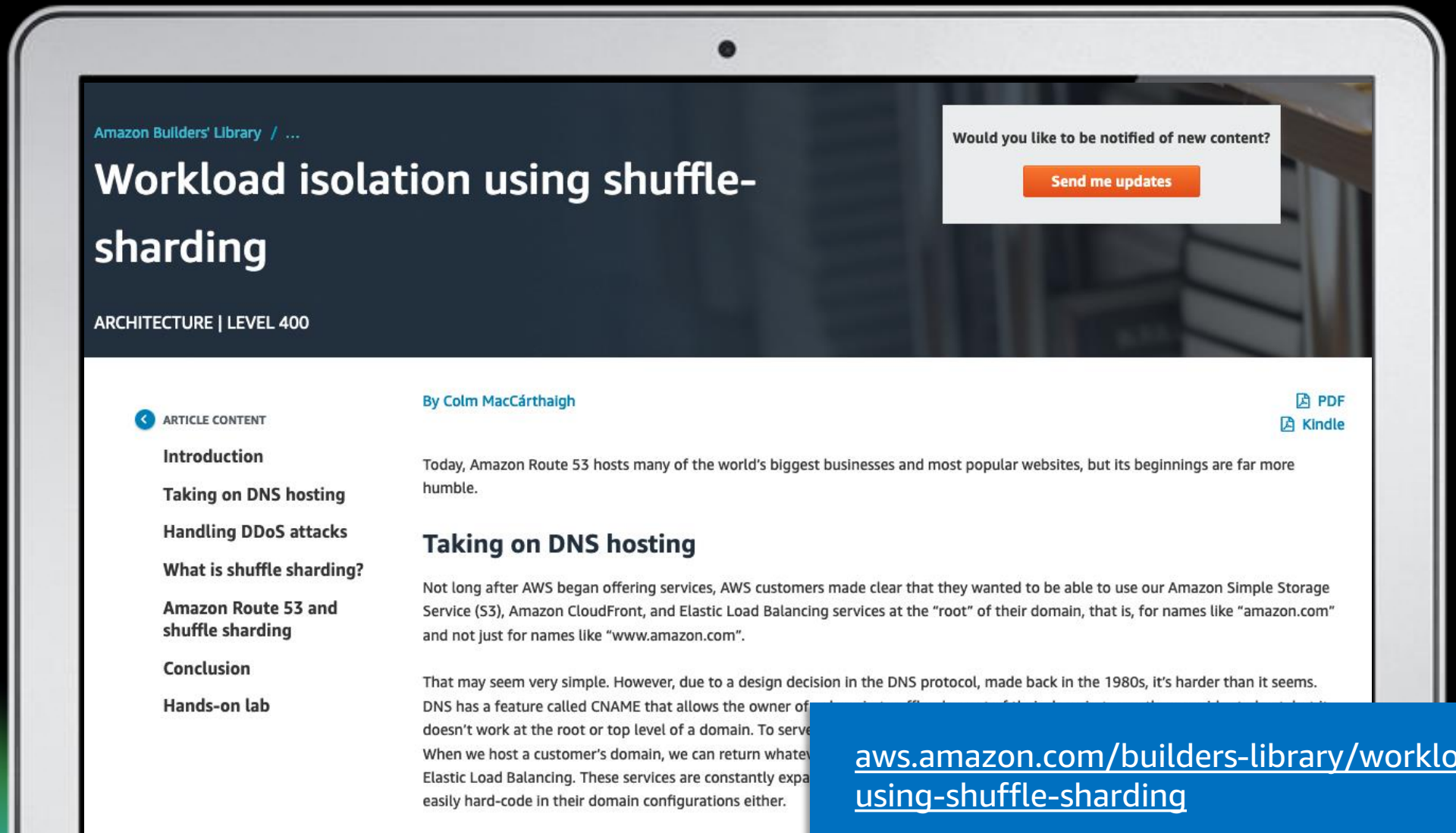
$$\text{Scope of impact} = \frac{\text{Customers}}{\text{Combinations}}$$

*retry logic in the client to try every endpoint in a Shuffle Shard, until one succeeds

Shuffle Sharding

- Needs a client that retires or is fault tolerant
- Needs a routing mechanism – per customer DNS name, per resource DNS name of Shuffle sharding aware router
- Sharding strategy - customer id, customer-resource-operation-type, multidimensional sharding

Shuffle Sharding



aws.amazon.com/builders-library/workload-isolation-using-shuffle-sharding

Summary

1. Timeouts, Retries – Use timeouts on any remote or inter-process calls. Think of idempotency while retrying.

2. Load Shedding - Avoid overloading. Understand the cost of rejecting a request. Categorize your incoming request types and decide which ones to shed.

3. Static Stability – Build some redundancy into your data plane so that they can continue to work despite control plane failure.

4. Constant Work – Work that a system does should be idempotent in nature and not get affected by variations in load or stress.

5. Shuffle Sharding – Applying fault isolation to traditional horizontal scaling.

https://aws.amazon.com/builders-library/

Thank you for attending AWS Innovate – For Every Application Edition

We hope you found it interesting! A kind reminder to **complete the survey.**
Let us know what you thought of today's event and how we can improve the event experience for you in the future.


aws-apj-marketing@amazon.com


twitter.com/AWSCloud


facebook.com/AmazonWebServices


youtube.com/user/AmazonWebServices


slideshare.net/AmazonWebServices


twitch.tv/aws

Thank you!