
AWS
re:Invent

BLD204

Designing with simplicity at Amazon

Colm MacCárthaigh

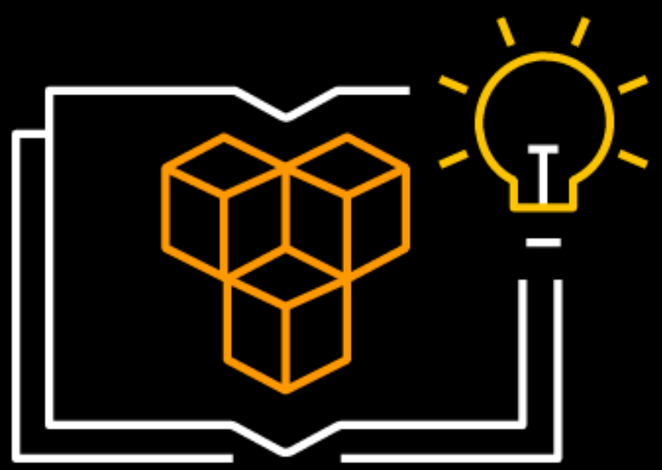
Sr. Principal Engineer
AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The Amazon Builders' Library

How Amazon builds and operates software



Architecture, software delivery, and operations

By Amazon's senior technical executives and engineers

Real-world practices with detailed explanations

Content available for free on the website

The Amazon Builders' Library

ARCHITECTURE

LEVEL 300



Leader election in distributed systems

Author: Marc Brooker

Improving efficiency, reducing coordination, and simplifying architectures by using leader election.

SOFTWARE DELIVERY AND OPERATIONS

LEVEL 300



Going faster with continuous delivery

Author: Mark Mansour

Automating the software testing and deployment process for speed and reliability

SOFTWARE DELIVERY AND OPERATIONS

LEVEL 400



Implementing health checks

Author: David Yanacek

Automatically detecting and mitigating server failures without unintended consequences from fleet-wide false positives.

ARCHITECTURE

LEVEL 400



Workload isolation using shuffle-sharding

Author: Colm MacCarthaigh

Shuffle Sharding is one of our core techniques for drastically limiting the scope of impact of operational issues

Many more at aws.amazon.com/builders-library

Invent and Simplify

Amazon Leadership Principles



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Invent and Simplify

Leaders expect and require innovation and invention from their teams and always find ways to simplify. They are externally aware, look for new ideas from everywhere, and are not limited by “not invented here.” As we do new things, we accept that we may be misunderstood for long periods of time.

Amazon Leadership Principles



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is simple?

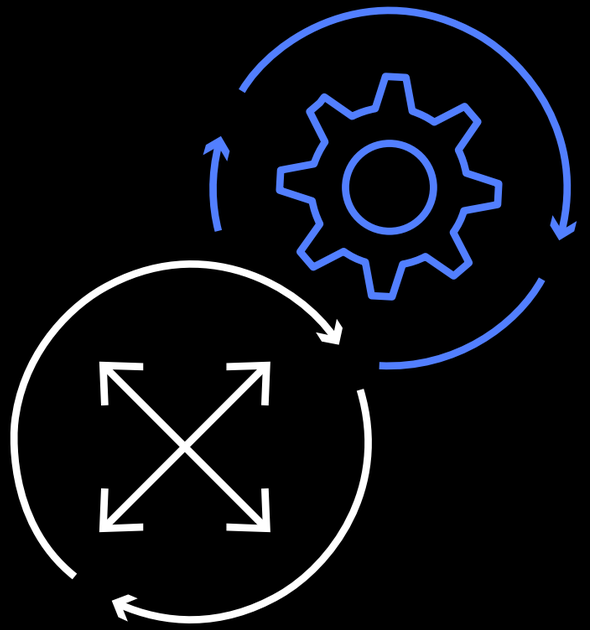


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

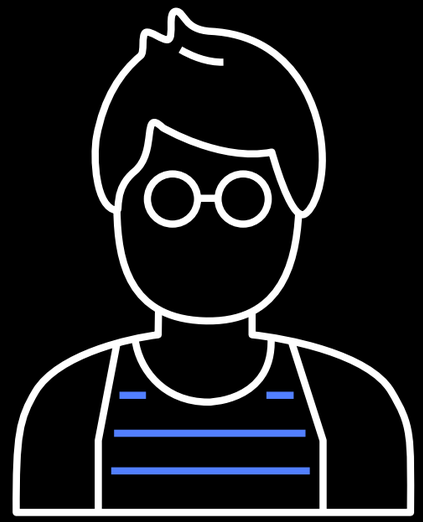
What is simple?

COMMON ANSWERS

Fewer moving parts

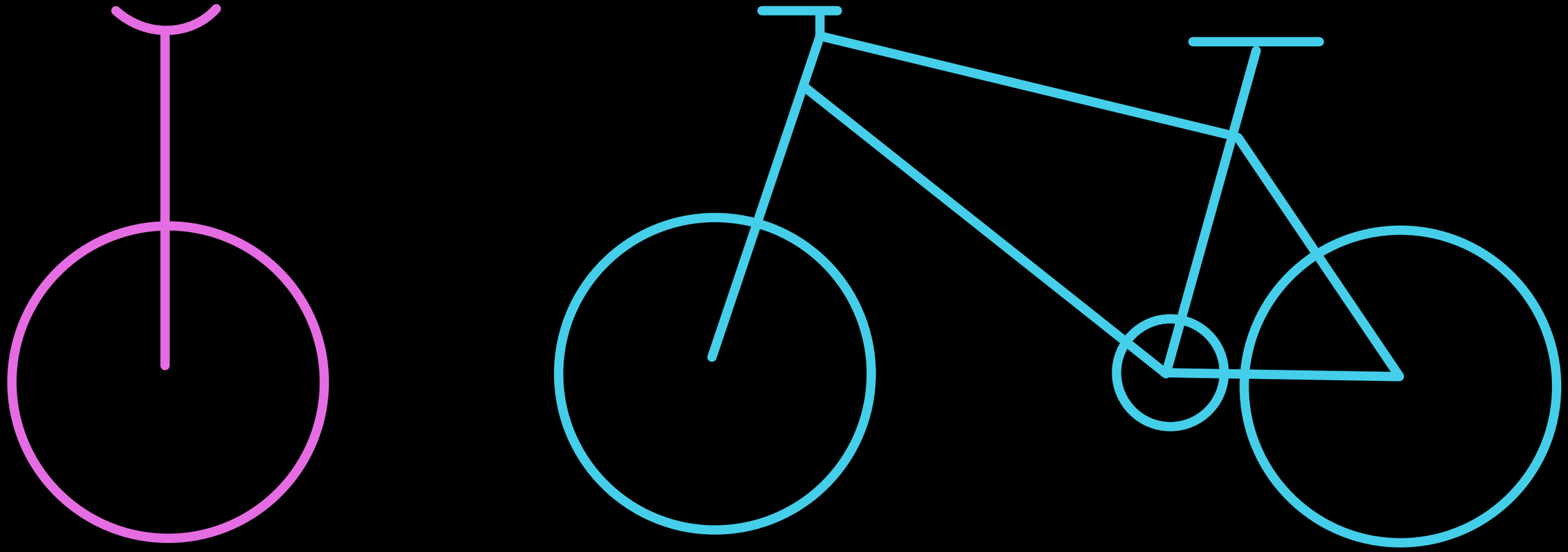


Easy to understand and use



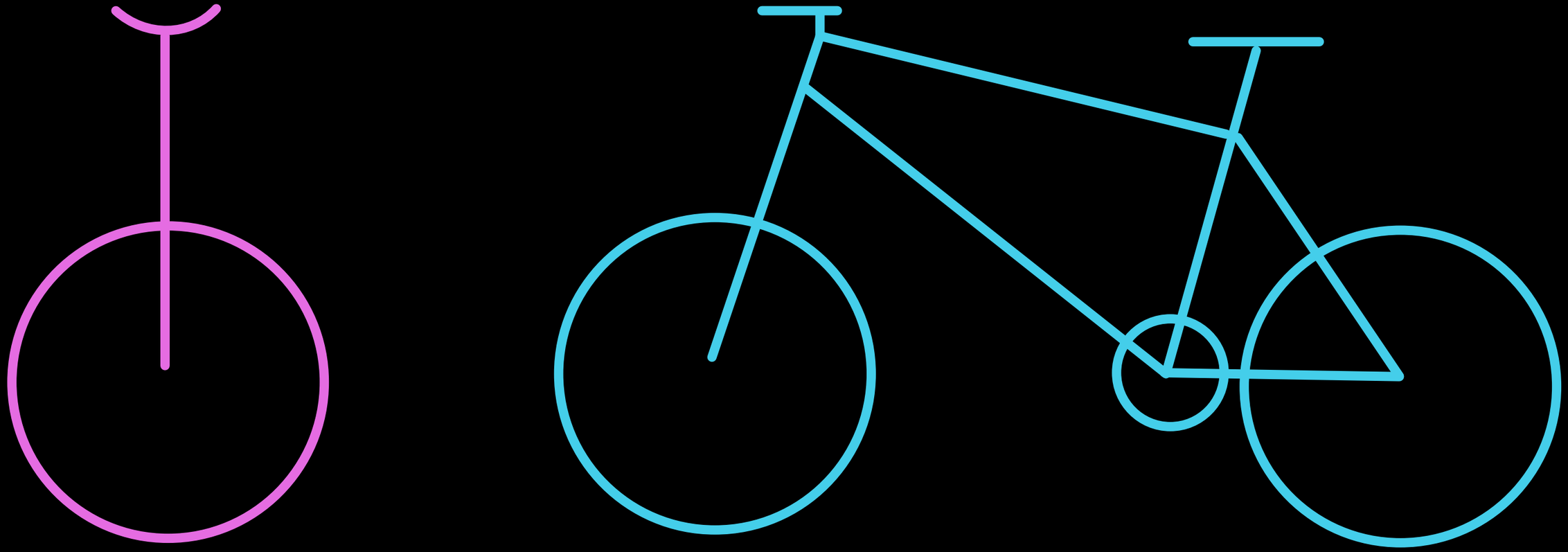
What is simple?

Is “fewer moving parts” really simpler?



What is simple?

Elegance can be deceptive



Why simple?



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why simple?

Security

Durability

Reliability

Cost

Manageability

Why simple?

We are still in the early days of the internet

Long-term success is built on components that prove their worth during times of stress

Long-term success also depends on usability and ownership of those components across generations of teams and management

Simple solutions can be understood and managed effectively by people who had nothing to do with their inceptions

Five examples of simple



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

1. Availability Zones



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Availability Zones

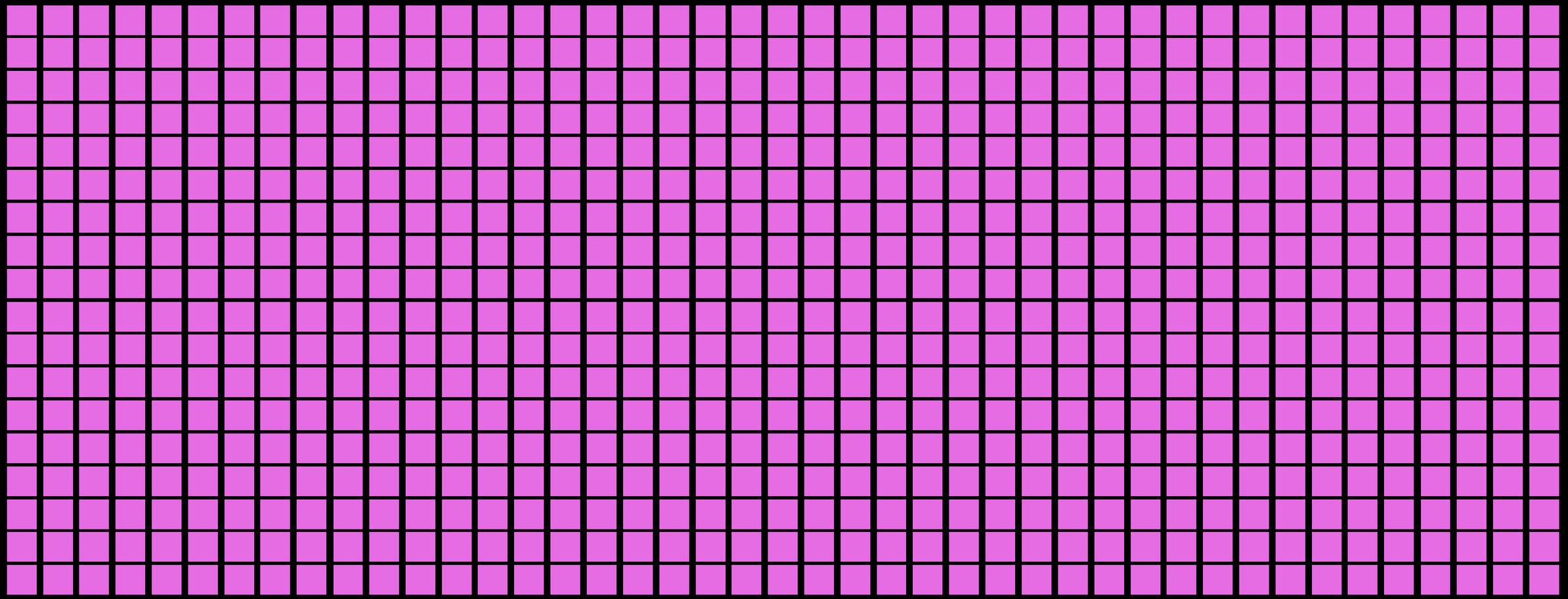
Builders need meaningful resilience and redundancy

Builders need a way to reason about the shared fate of infrastructure

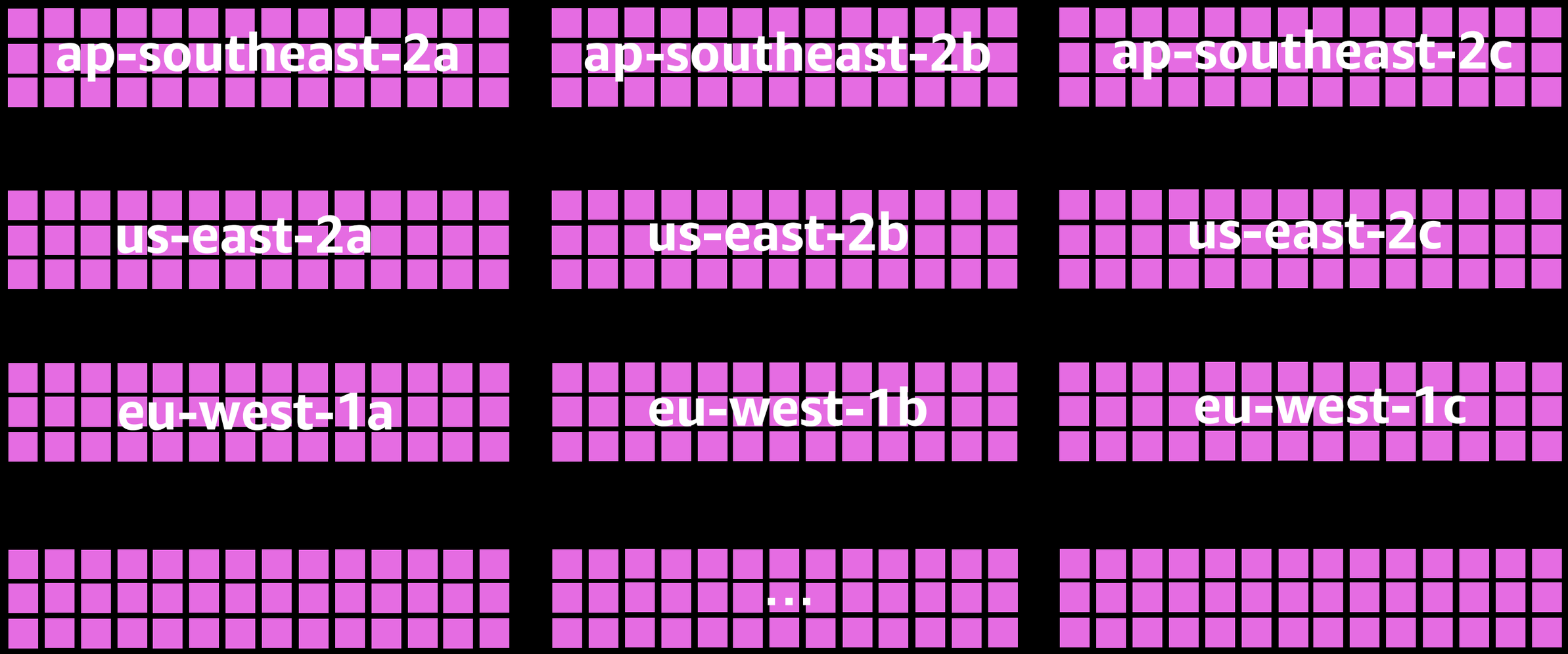
Even when you have redundant power and networking inside a building, there are still reasons why they may experience a problem at the same time

Examples: Switchover systems, fire safety, extreme weather events, tectonic events

Availability Zones



Availability Zones



Availability Zones

We reuse the Availability Zone concept over and over

Example: We phase software deployments and configuration changes by Availability Zone

Example: We sometimes align hardware types with Availability Zones

Over time, we learned to simplify even more; customers now get the same Availability Zone names

2. SIGv4 authentication



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SIGv4 authentication protocol

~~SIGv1~~ ~~SIGv2~~ ~~SIGv3~~ SIGv4

Virtually all AWS API calls are authenticated using SIGv4

SIGv4 is also available as a feature on Amazon API Gateway

Based on fast symmetric cryptography

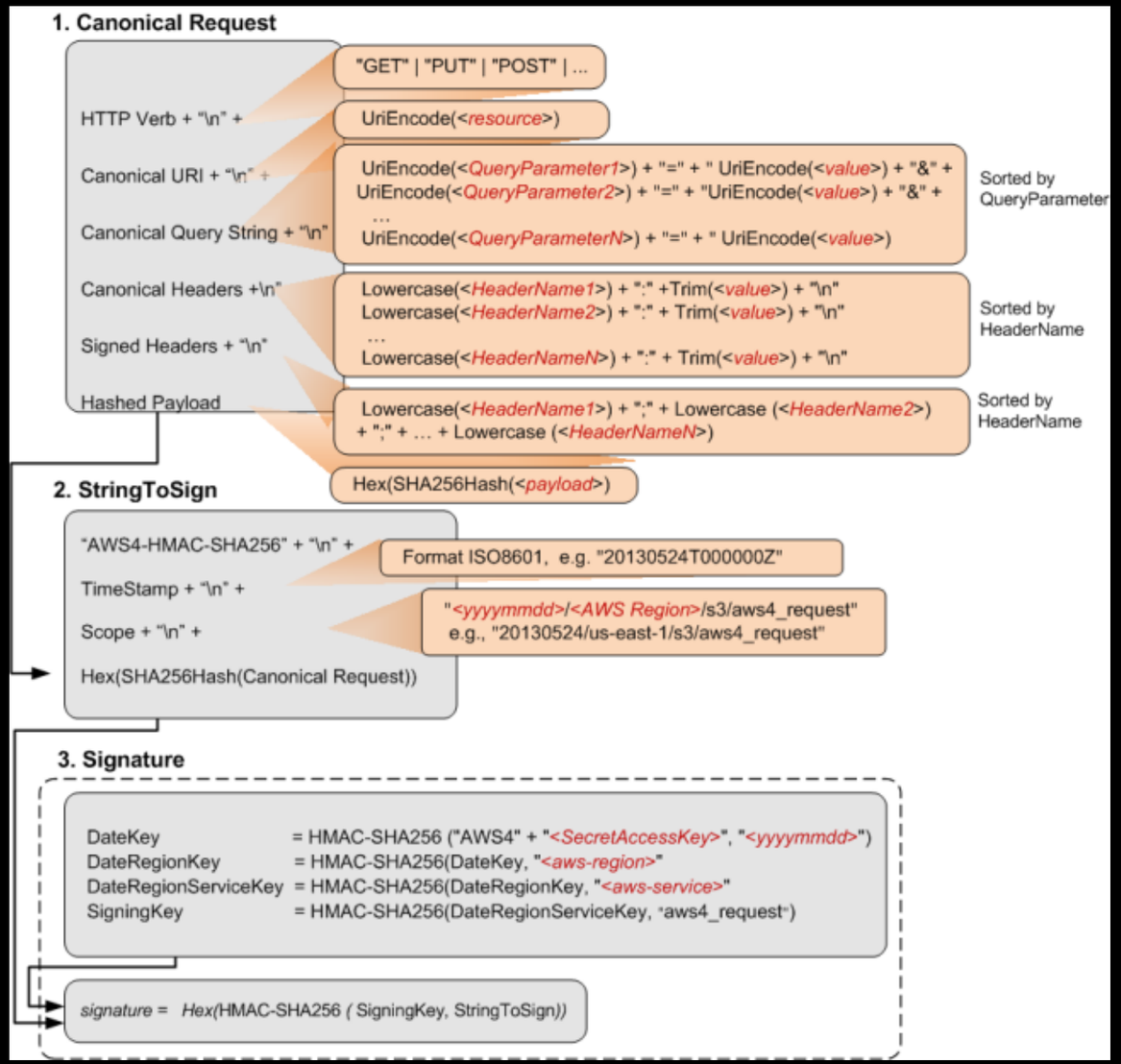
Can be implemented in an incredibly small amount of code

SIGv4 authentication protocol

Canonicalize the request headers

Derive an ephemeral time, Region, and service-specific key from an AWS access key

Sign using HMAC-SHA256



3. DNS health checks



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

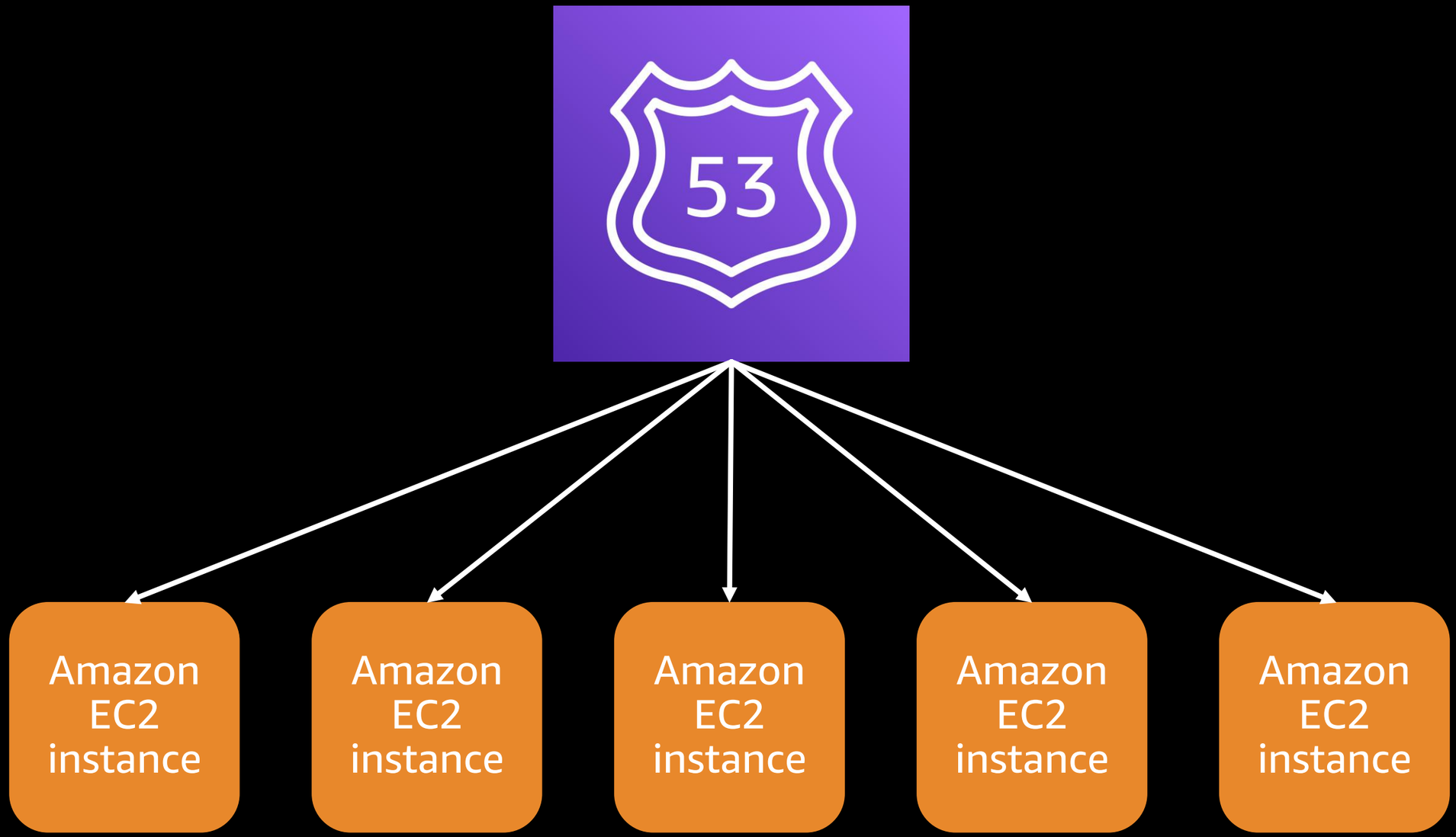
DNS health checks

DNS health checks are a feature built into Amazon Route 53

Also used extensively by AWS services for high availability and load balancing

Amazon Elastic Load Balancing, Amazon Relational Database Service, and more all use Route 53 health checks for high availability

DNS health checks



4. Rollback



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Rollback

Code review and testing are the first vanguards of reliability – Catching problems before they get to production is always the goal

Still, the changes introduced by software deployments into production can trigger issues

We phase and stagger deployments, starting small and increasing deployment footprints as confidence increases

We’ve learned that fast, reliable rollback at every stage is a key to success

Rollback

Hit a problem? Roll it back! Don’t need to spend time on root cause diagnosis or live debugging

This takes preparation and effort – Teams have to consider that every change is eligible for rollback and organize their changes that way

Example: A schema change is often 2 separate changes

1. Add the new schema, but still write to the old one
2. Stop writing to the old one

5. Static stability



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Static stability

If a system loses power or networking, it should come back to the state it was previously in when power or networking is restored

This is harder than it seems

Static stability implies that we need to be careful about the dependencies that we take – What if a dependency is not available when power is restored?

Example: Encrypting durable data

Static stability

If you’re encrypting durable data on a machine, it’s usually not secure to keep a durable copy of the encryption key locally

This is important for security against physical theft or confiscation

If you retrieve the key on startup, now you have a remote dependency to worry about – What if it’s not available? And how does that system have static stability?

Solution: AWS Nitro security chip with secure local key storage

Takeaways



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Takeaways

Lean in to a humanistic meaning of “simple”

What really matters is that it is easy to understand and easy to use

. . . filtered through survival of the fittest

Study examples anywhere you can find them

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.