
AWS
re:Invent

STG301

Architecting for high availability on Amazon S3

Eno Thereska

Principal Engineer

AWS

AWS re:Invent

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

Agenda

Measuring availability

End-to-end robustness in Amazon S3

Operational best practices

Building applications for high availability



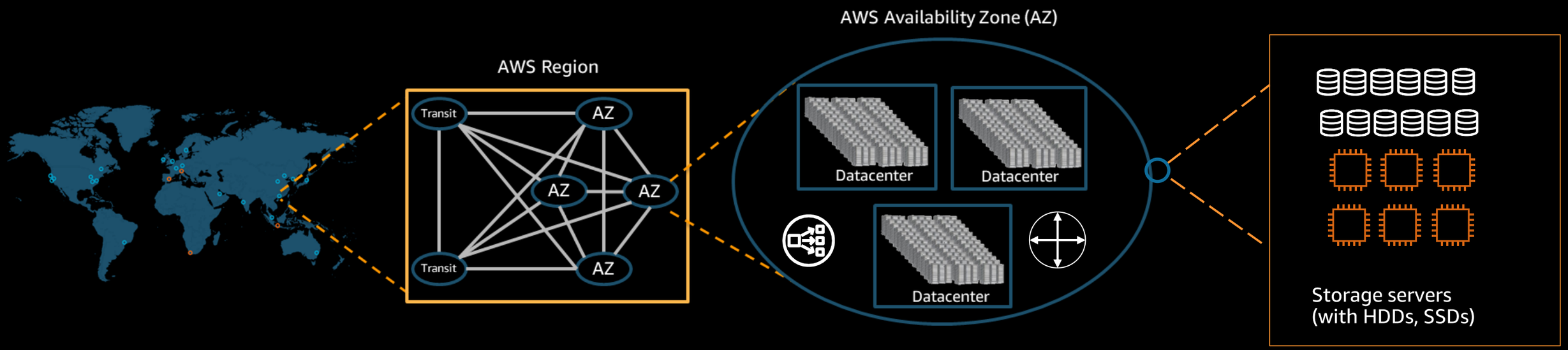
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Measuring availability



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

Amazon S3 internals: 10,000 ft view



Availability Zones, LBs, racks, switches, storage servers, disks…

Continuous deployment of 100s of microservices

Amazon S3 from the user's point of view



Is a highly available building block

Allows user to build robust applications

What is a good way to measure availability from the user's point of view?

What does availability mean to a customer?

It’s personal

- A user’s observed availability matters

It’s the system experience when it matters most

- When users send requests to Amazon S3 – they want them to succeed
- During a period of work when the user wants minimum downtime

Measure

Per-customer availability

Metric

Request error rate

Duration

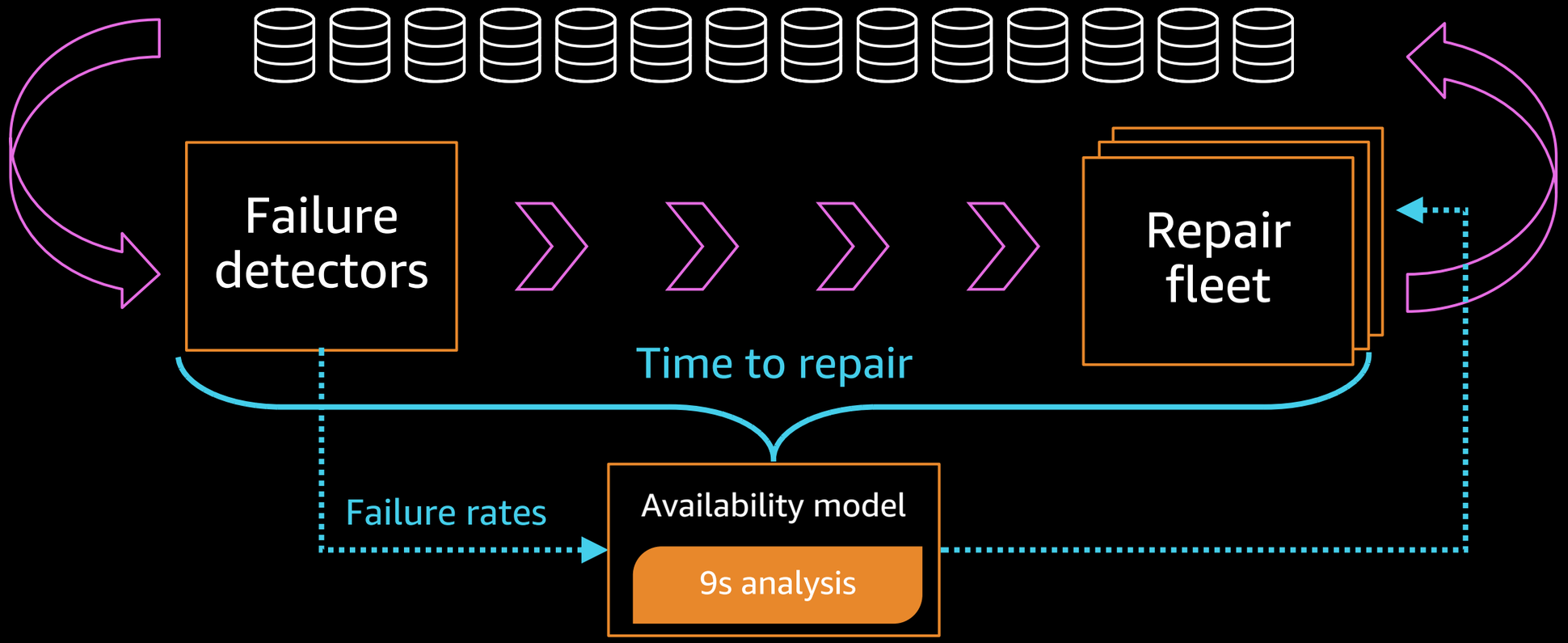
Fine-grained time interval

How does Amazon S3 measure availability?

Measure
Per-customer availability

Metric
Request error rate

Duration
Fine-grained time interval



End-to-end robustness in Amazon S3



AWS re:Invent

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

“Everything fails all the time.”

Werner Vogels

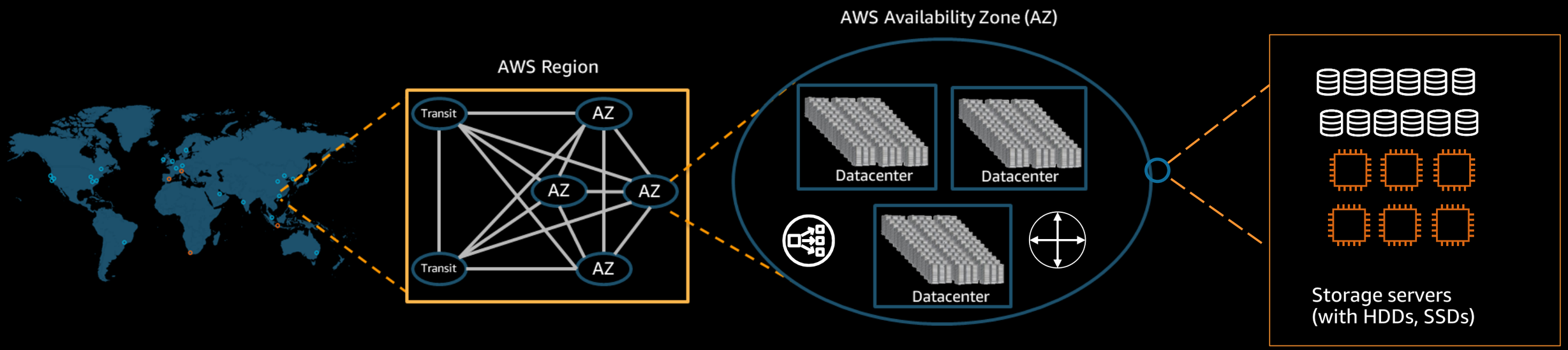
CTO AWS



AWS re:Invent

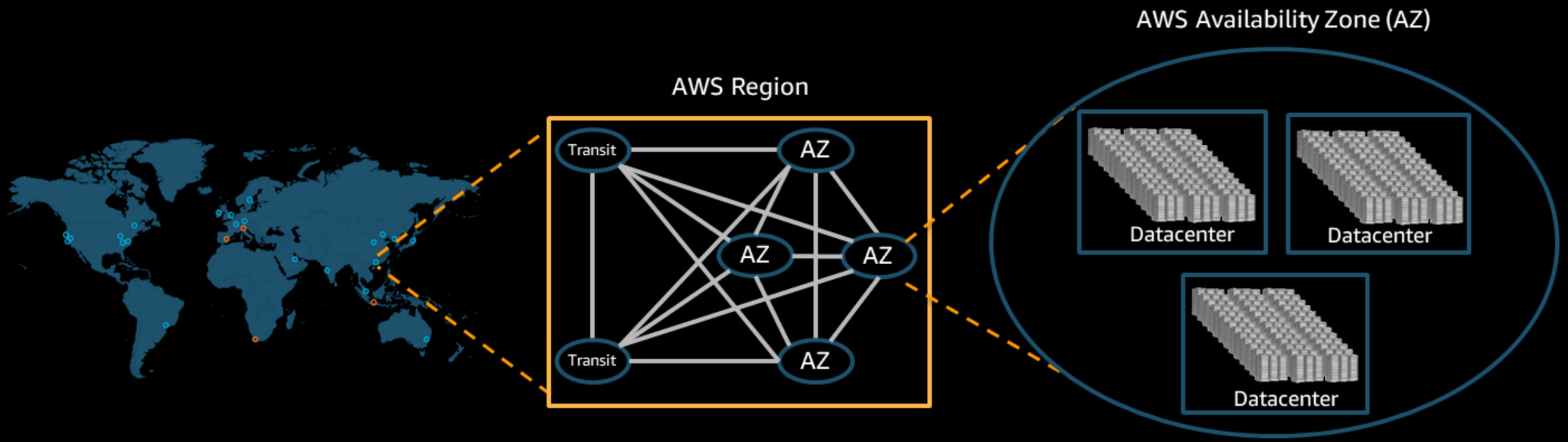
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Hardware infrastructure and failures



Availability zones, LBs, racks, switches, storage servers, disks, etc., can all fail permanently or transiently

Facility failover and recovery



Goal – availability in face of unexpected loss of entire facility

Mechanism – facility isolation and facility-aware data placement

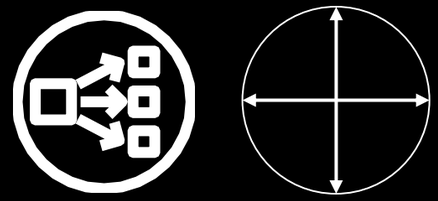
Hardware failure and recovery

GOALS

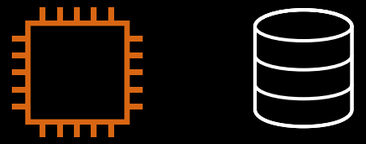
Goal – protect data while tolerating low rate of component failures

Assume arbitrary failures of servers or drives, hardware defects, and correlation of failures across devices

Network components



Storage server components



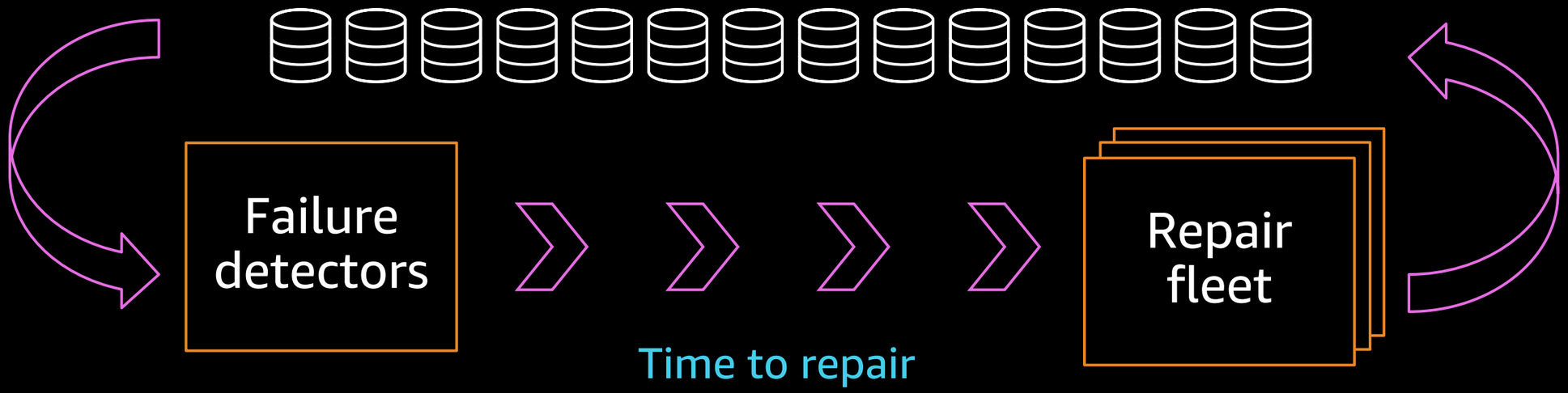
Power components



Hardware failure and recovery

MECHANISMS

- Have redundancy in everything
- Continuously improve failure detection
- Continuously improve recovery times



Transient failures

Some failures are transient, not permanent (e.g., component queues could be temporarily overloaded and shed traffic or DNS resolving might fail)

- Mechanisms – appropriate error codes (e.g., 5xx server or throttling), exponential backoffs with jitter, bounded number of retries and DNS re-resolving
- Tuned internally in Amazon S3
- Keep AWS SDK up-to-date
- Follow best practices to re-resolve DNS

Operational best practices



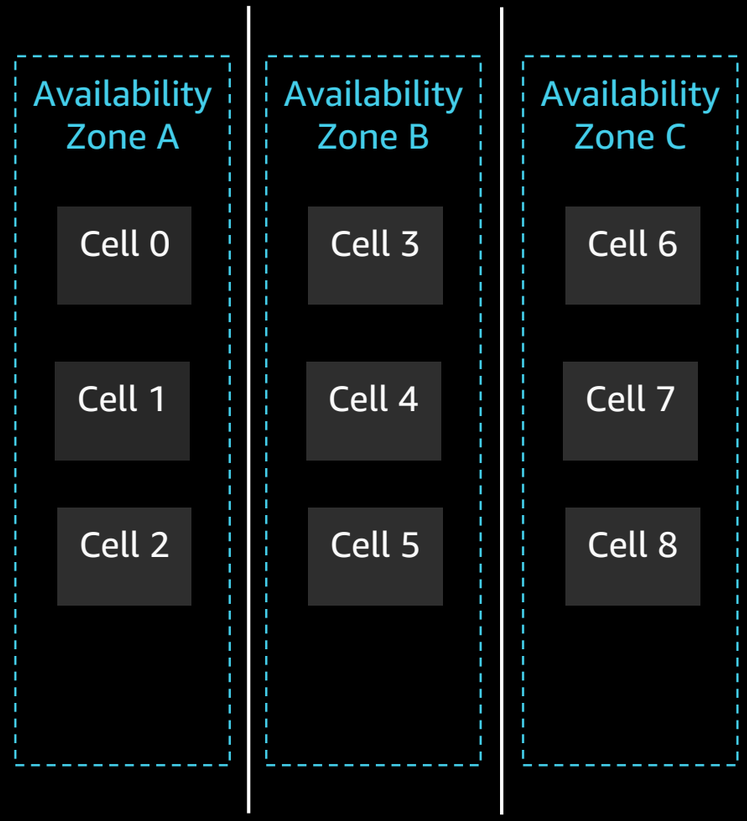
AWS re:Invent

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

Cell-based architecture to reduce blast radius

Goal – do not damage multiple cells at the same time

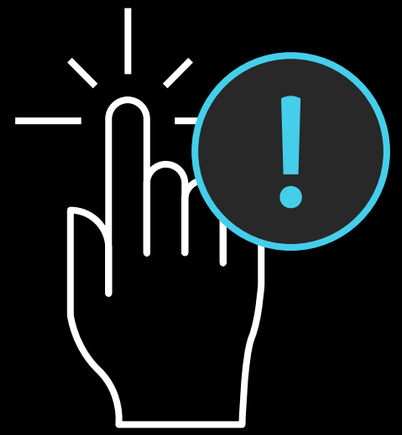
- Deploy gradually but continuously
- Deploy to one cell and measure, then roll back or continue
- Roll back first; diagnose next



Robustness in face of operator mistakes

Humans make mistakes

Goal is to ensure mistakes do not lead to unavailability

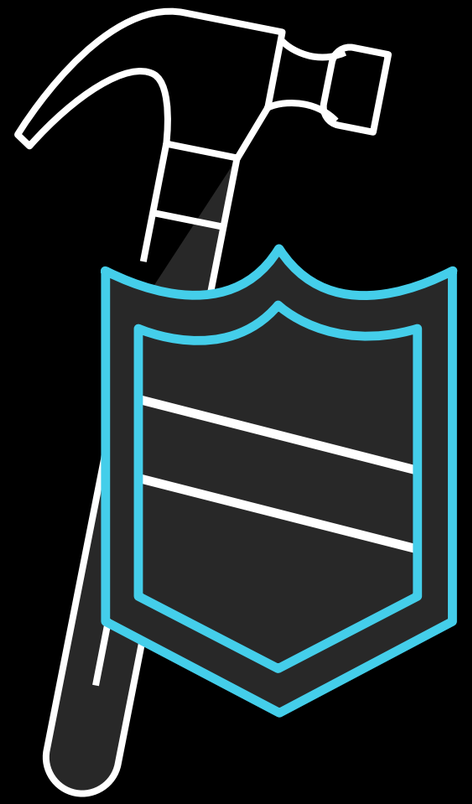


Automation, automation, automation

Principle of least privilege

Good tooling

- Prioritize tool improvements
- Don’t require unnecessary input that can be inferred
- Perform safety / sanity checks, and fail safely
- Don’t avoid misuse; make misuse safe



Organizational culture

Root-cause analysis reviews

- Identify root cause, prioritize improvements
- Are non-judgmental

Typical review agenda	
What happened?	What do we need to change?
What was the user impact?	Action items
Why did it happen? (5 whys)	Owners and dates

Applying lessons to your own applications



AWS re:Invent

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

Applying these approaches

High-availability thinking for Amazon S3 customers



What parts of my application need extra robustness?

What are your business's other availability concerns given that Amazon S3 is high availability already?



What can go wrong?

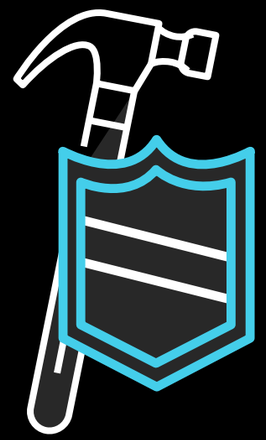


Consider this through the eyes of a hypothetical user

Availability Case Study 1

Goal

Robustness in face of compute instance failures (transient or permanent)



Mechanisms

Use multiple availability zones and AWS Auto Scaling

If using Amazon S3, state is already high availability

Focus on fast recovery for rest of state

“We want our customers to continue to be served with no downtime even when one or more of our compute instances are down”

Availability Case Study 2

Goal

Robustness in face of transient network glitches (e.g., DNS lookup failure)



Mechanisms

Ensure AWS SDK is kept up-to-date

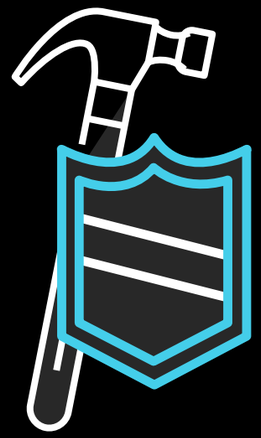
Configure DNS to re-resolve periodically (e.g., for Java, Sun's JVM caches DNS lookups forever by default and you need to change that)

“We want our applications to continue to operate in face of transient network failures”

Availability Case Study 3

Goal

Deploy new software with minimal or no downtime



Mechanisms

Reduce blast radius through cell-based architectures

Deploy, measure, roll back

Guard against operator errors (e.g., disallow SSH)

“We want to deploy bug fixes and new features more frequently but worry that our deployments will result in customer downtime”

Summary

Measuring availability

End-to-end robustness in Amazon S3

Operational best practices

Building applications for high availability

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.