
AWS
re:Invent

ARC309

How to scale beyond limits with cell-based architectures

Julien Lépine

Specialist SA Manager, EMEA

AWS

AWS re:Invent

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

Example Corp.

Smart buildings company

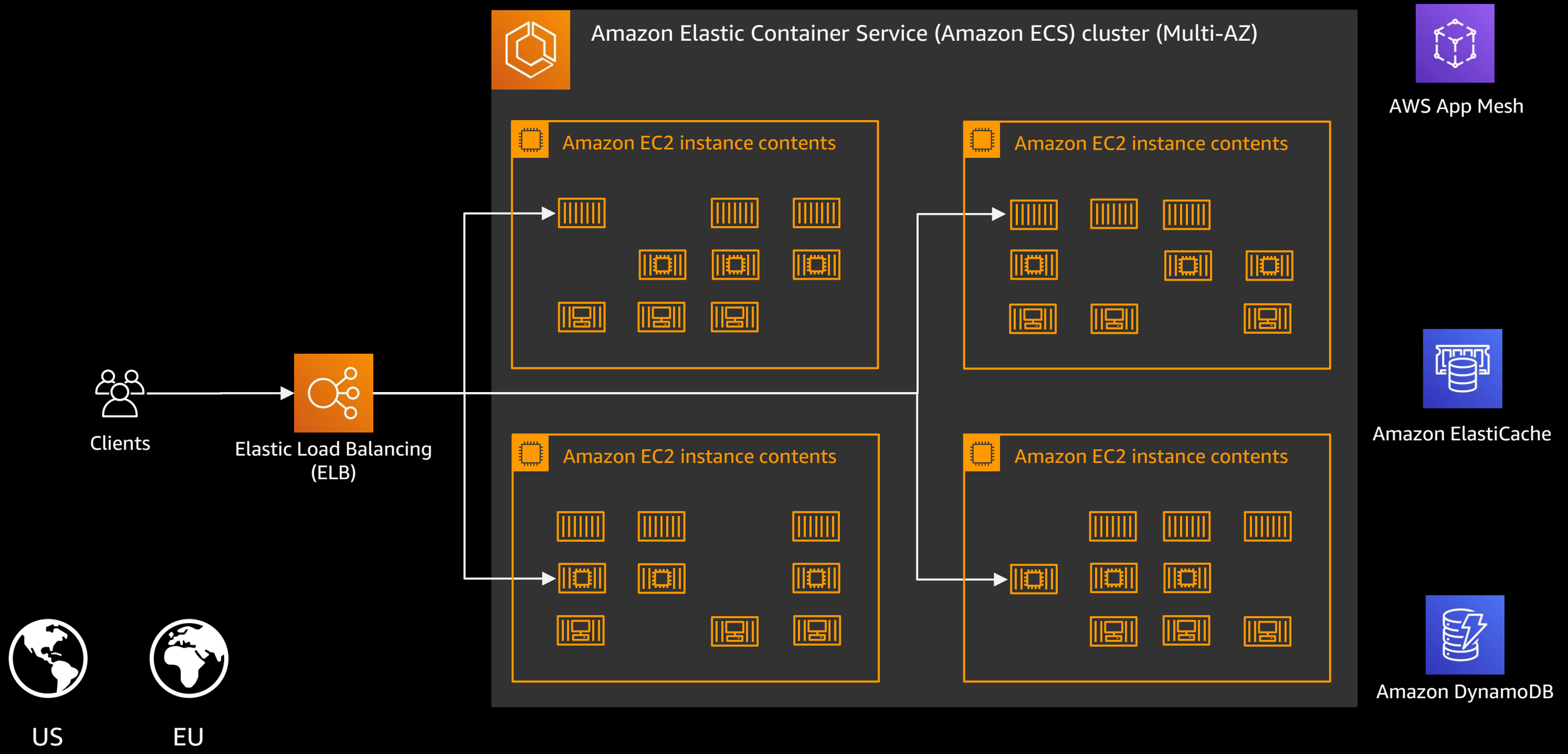
Worldwide customer base

Enterprise customers

Modern platform, 24 / 7



Example Corp. Smart Buildings Platform



“That day”

JANUARY 2 (PARIS TIME)

1:37 p.m. – Monitoring alerts

2:15 p.m. – US Region unavailable

3:12 p.m. – Failover EU Region online

3:58 p.m. – EU Region unavailable

9:29 p.m. – Service restored



Root cause analysis

TIMELINE OF THE EVENT

December 5 – New feature for new customers

January 2 – Feature used by existing customer

1:00 p.m. – US / EU configuration conflict

Authentication cache corrupted

Throttling on source data tables

Circuit breaker on downstream services

Cache expires; services unresponsive

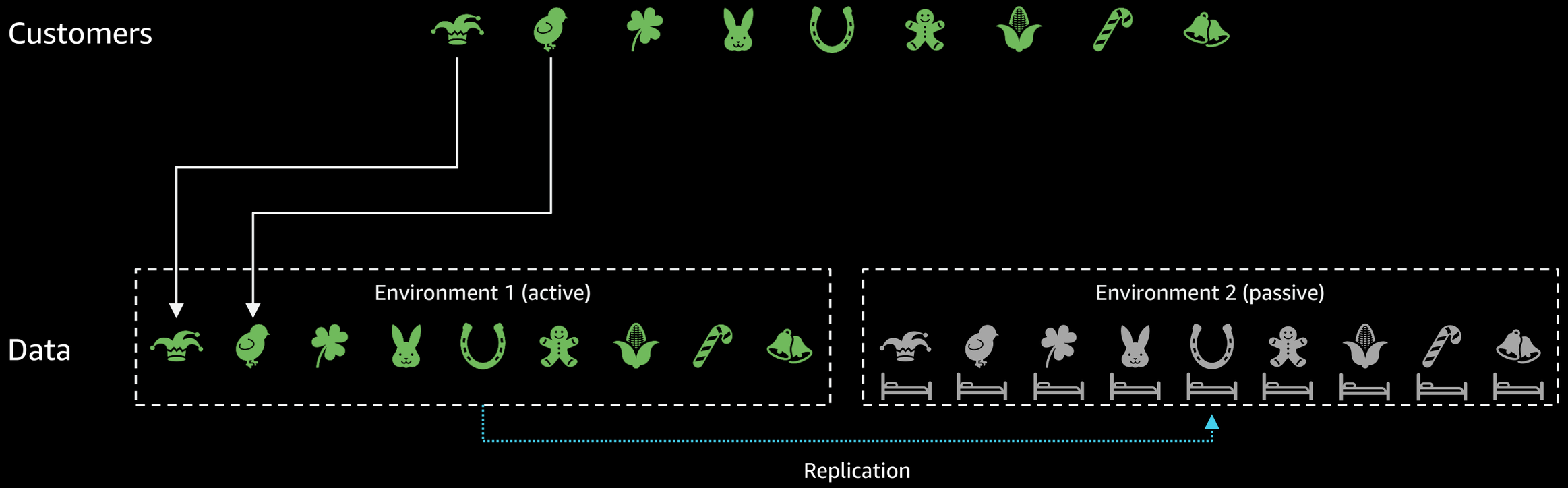
3:12 p.m. – Failover EU Region online

Behavior repeats

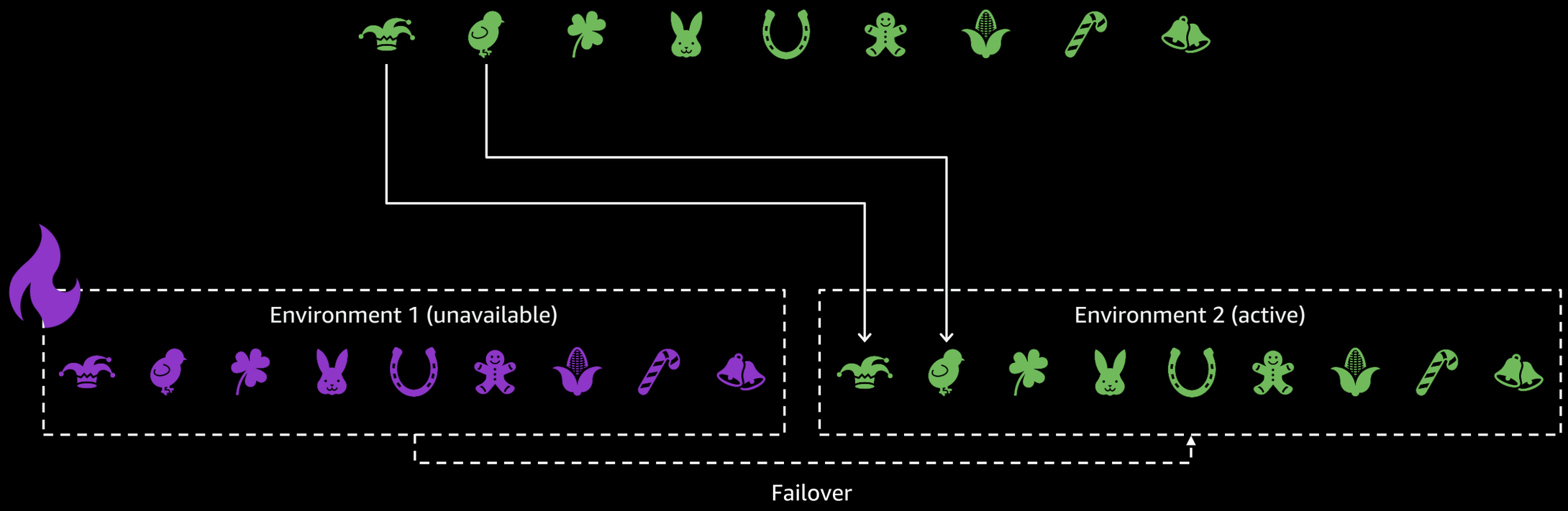
Patch to production services built; service restored at **9:29 p.m.**



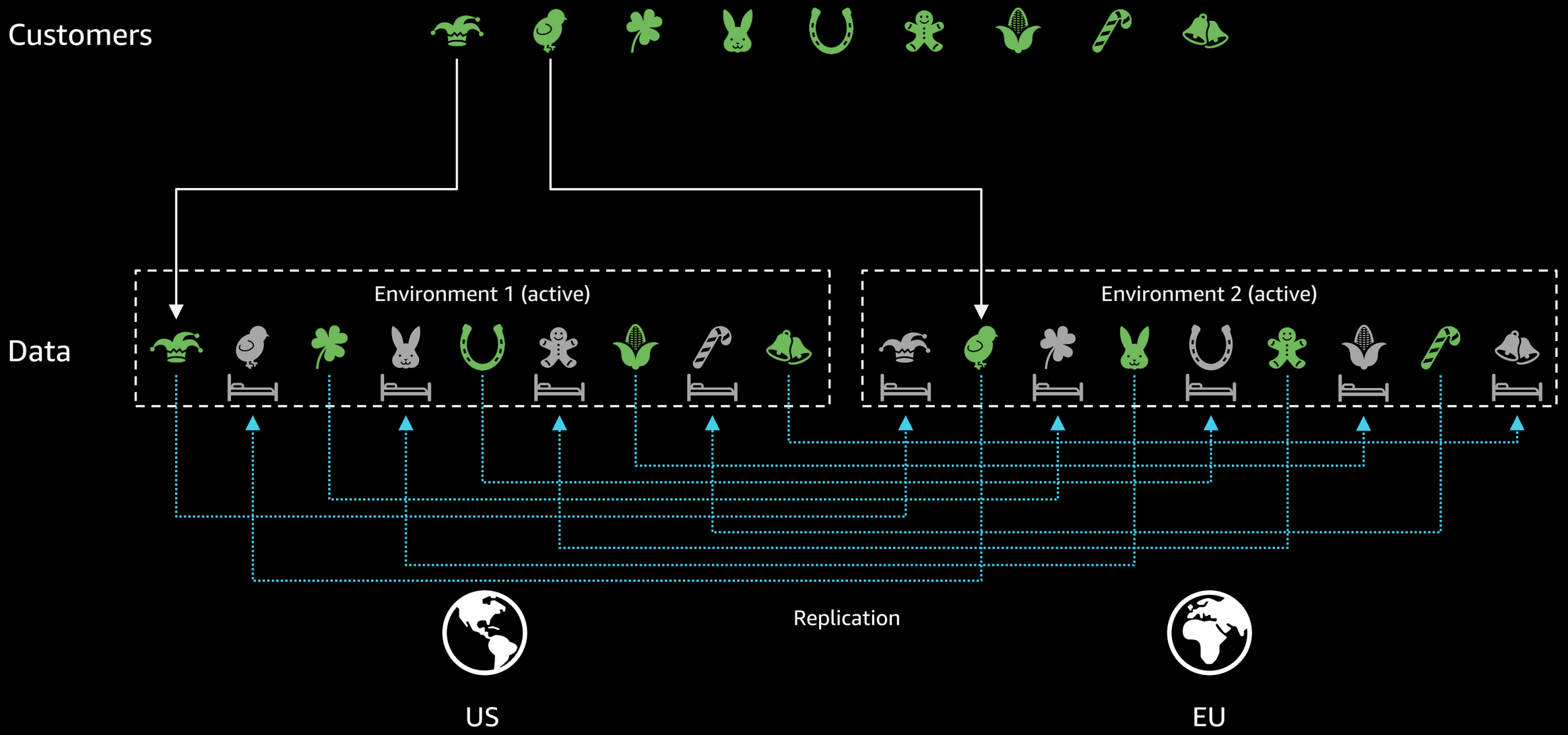
“Traditional” active / passive environments



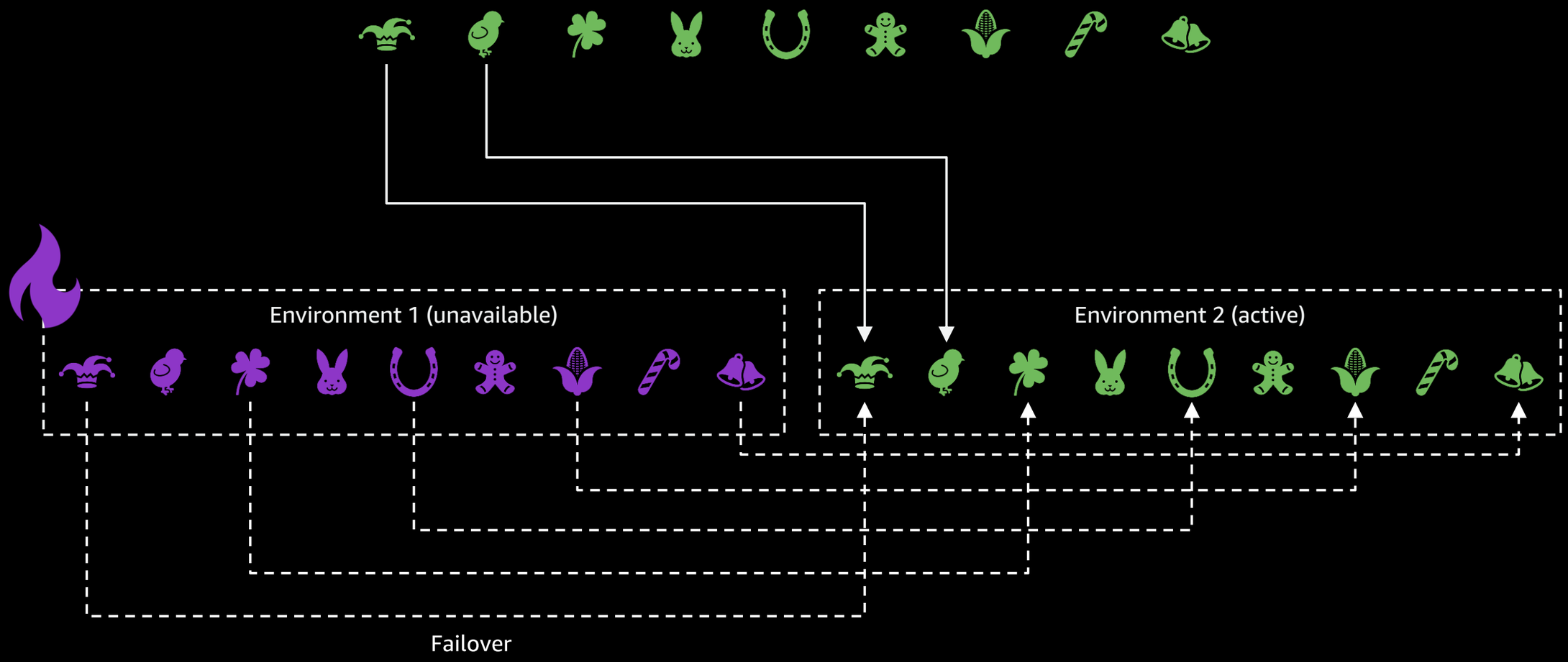
Environment operational event



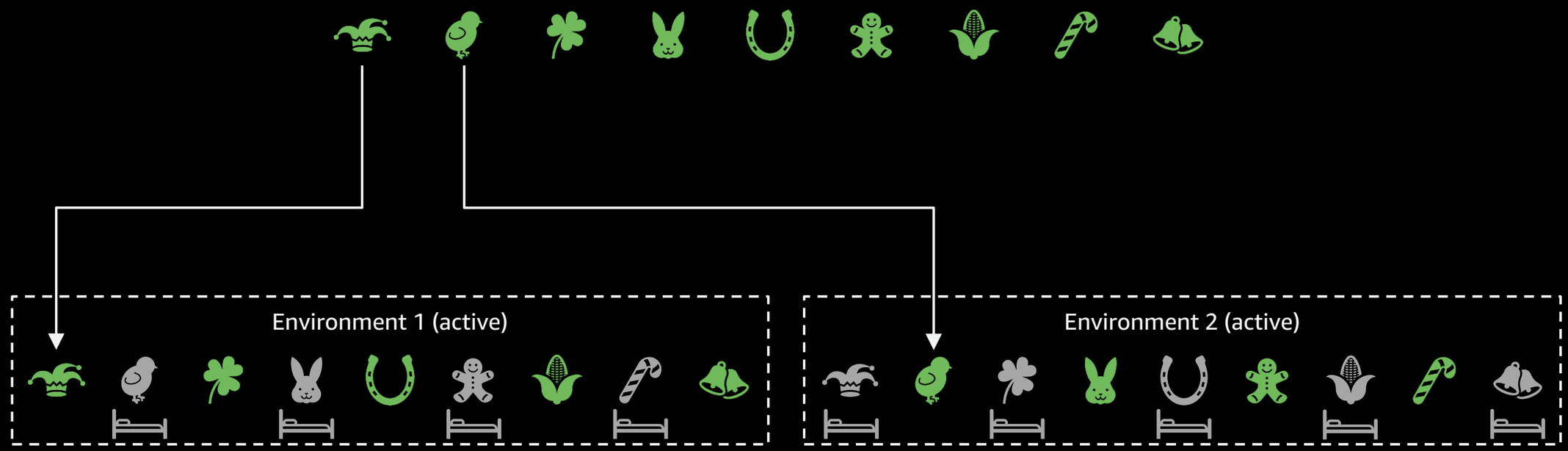
Active / active environments



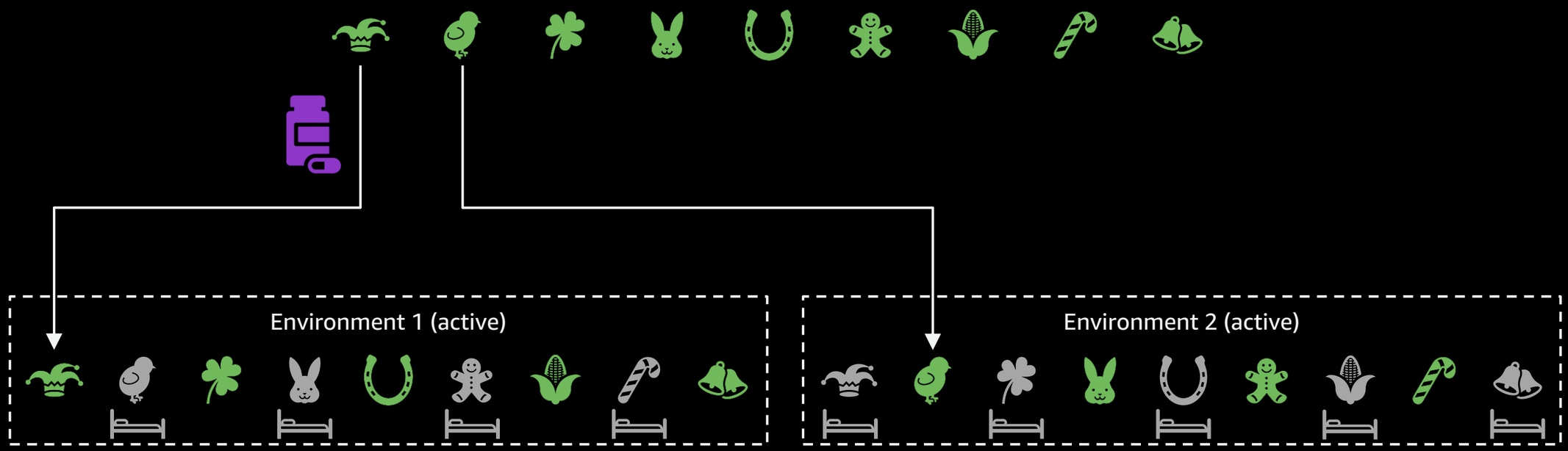
Environment operational event



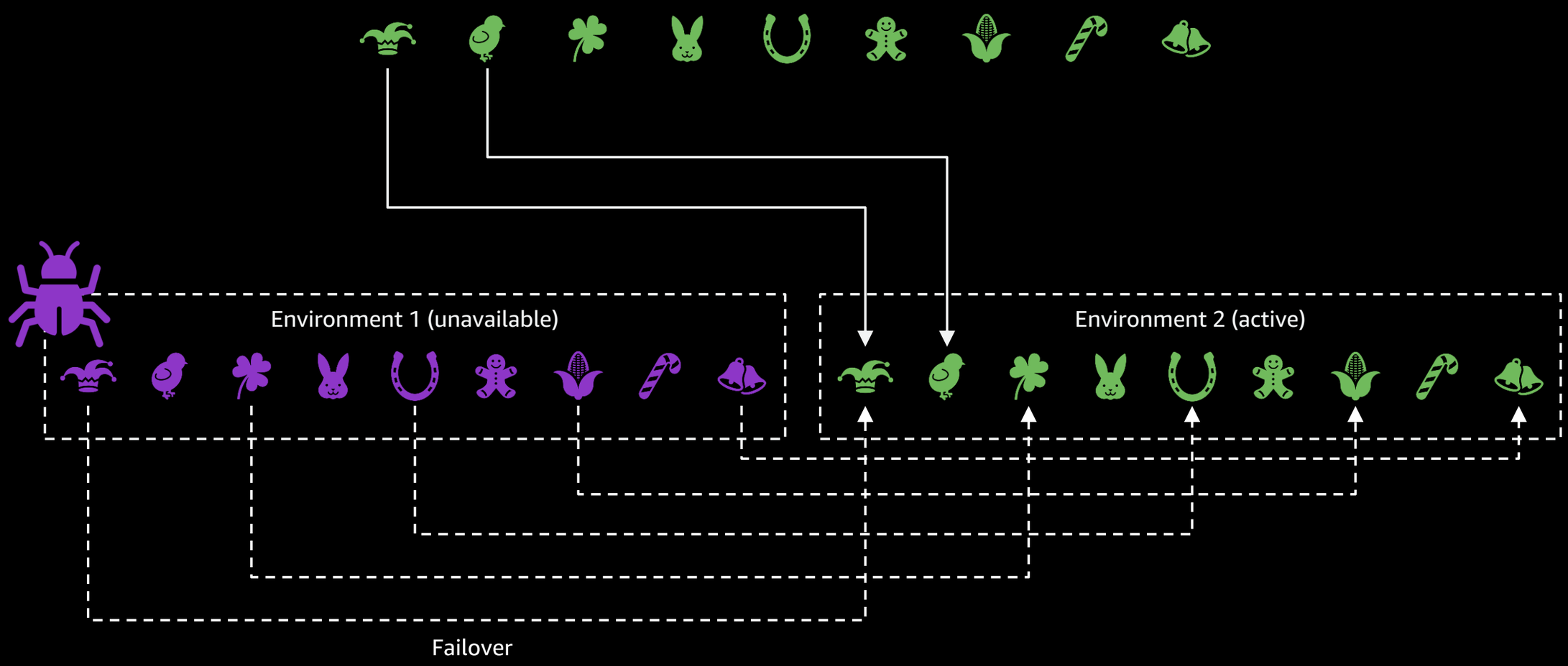
Poison pill



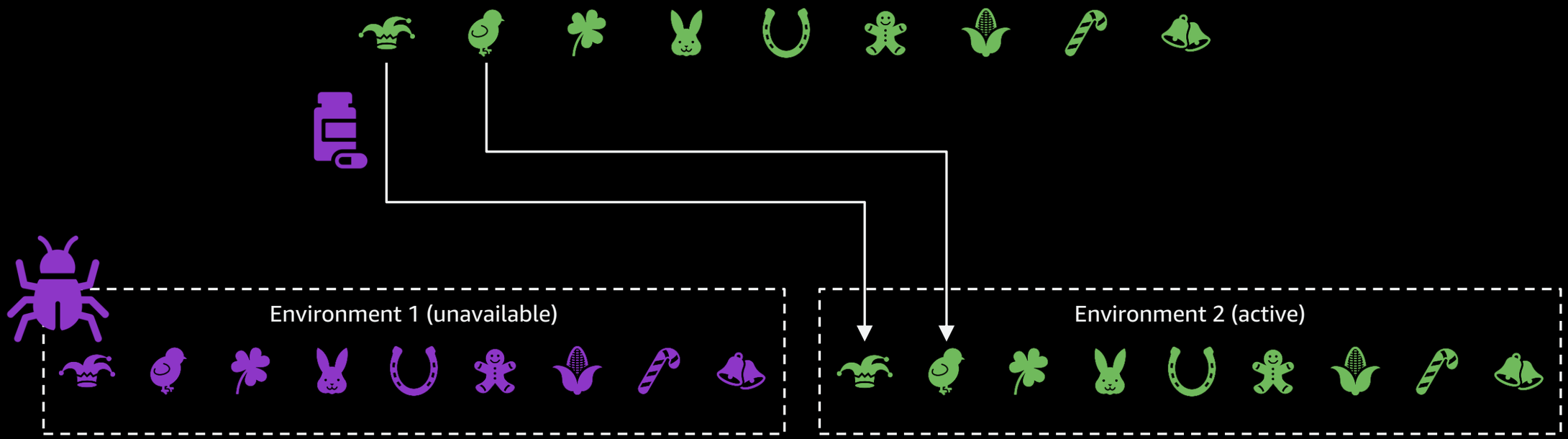
Poison pill



Poison pill / failover

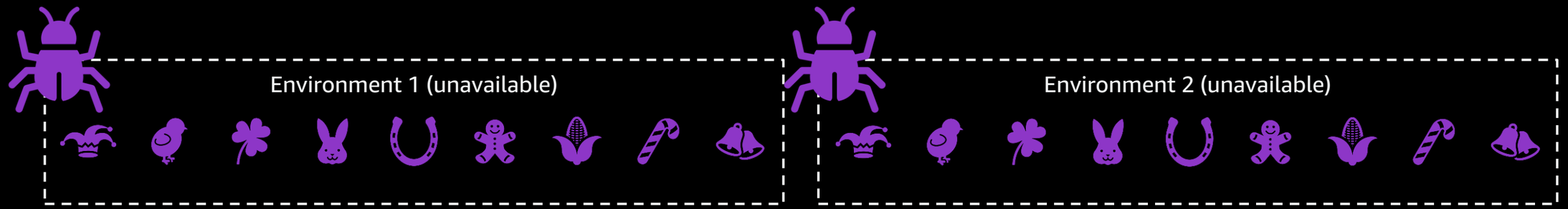


Poison pill on failover



Poison pill / global impact





Shuffle sharding

Accepts failure

- Resiliency and reliability

Deploy customer in **shards**

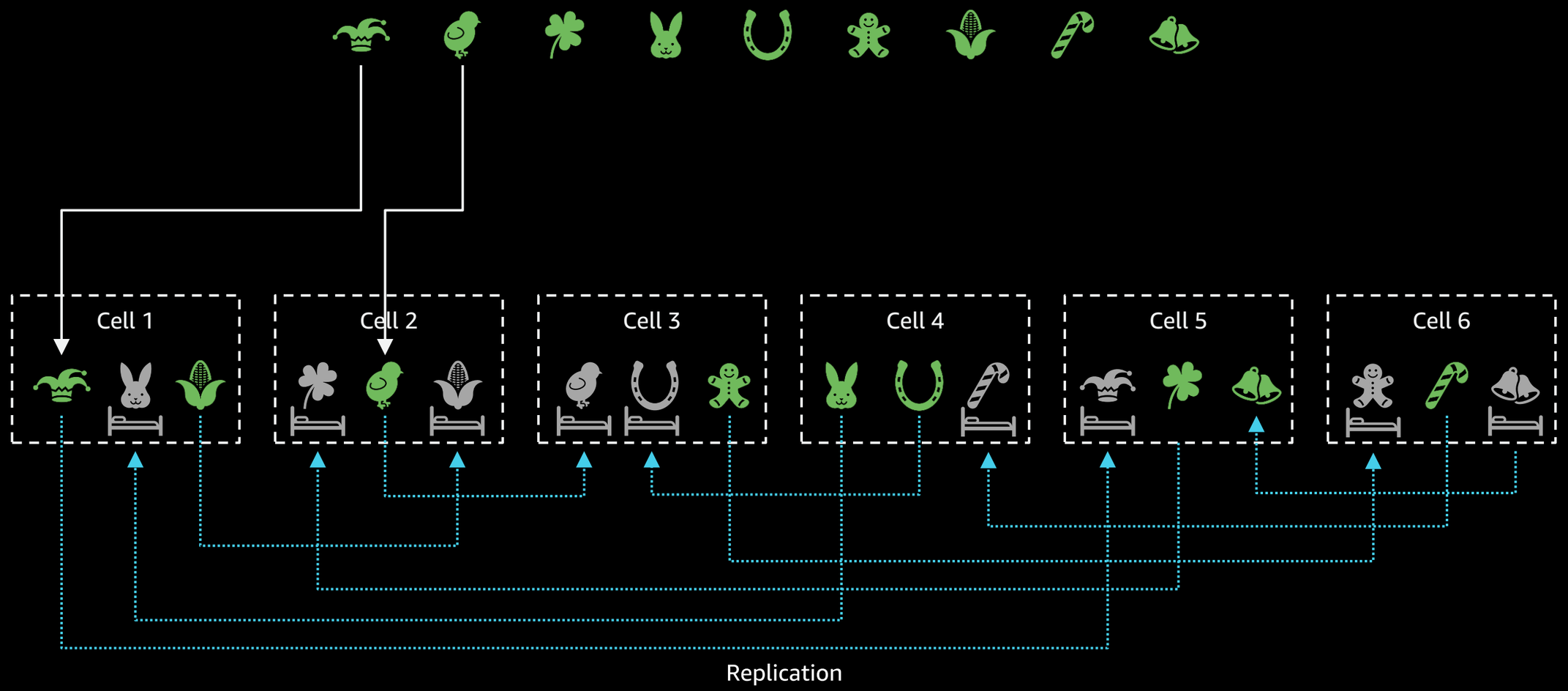
- Contains the blast radius

Random assignment to shards

- Limits customer impact



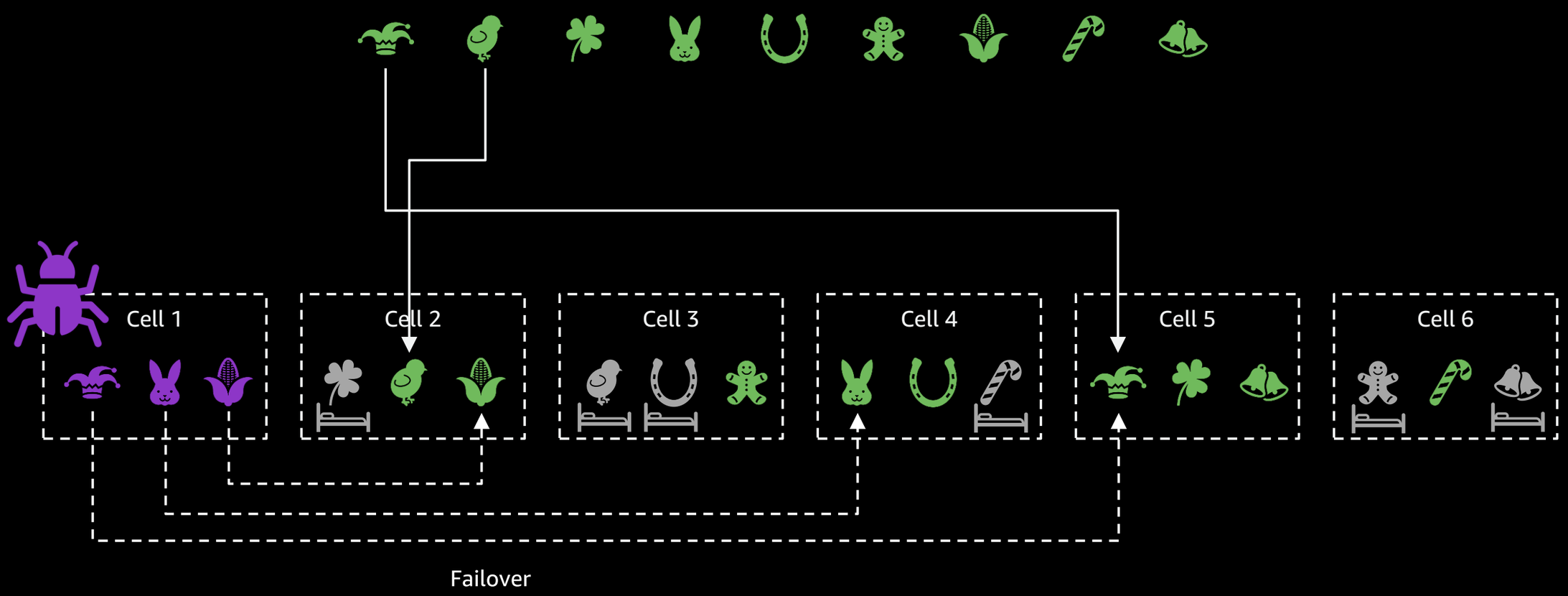
Shuffle sharding



Poison pill



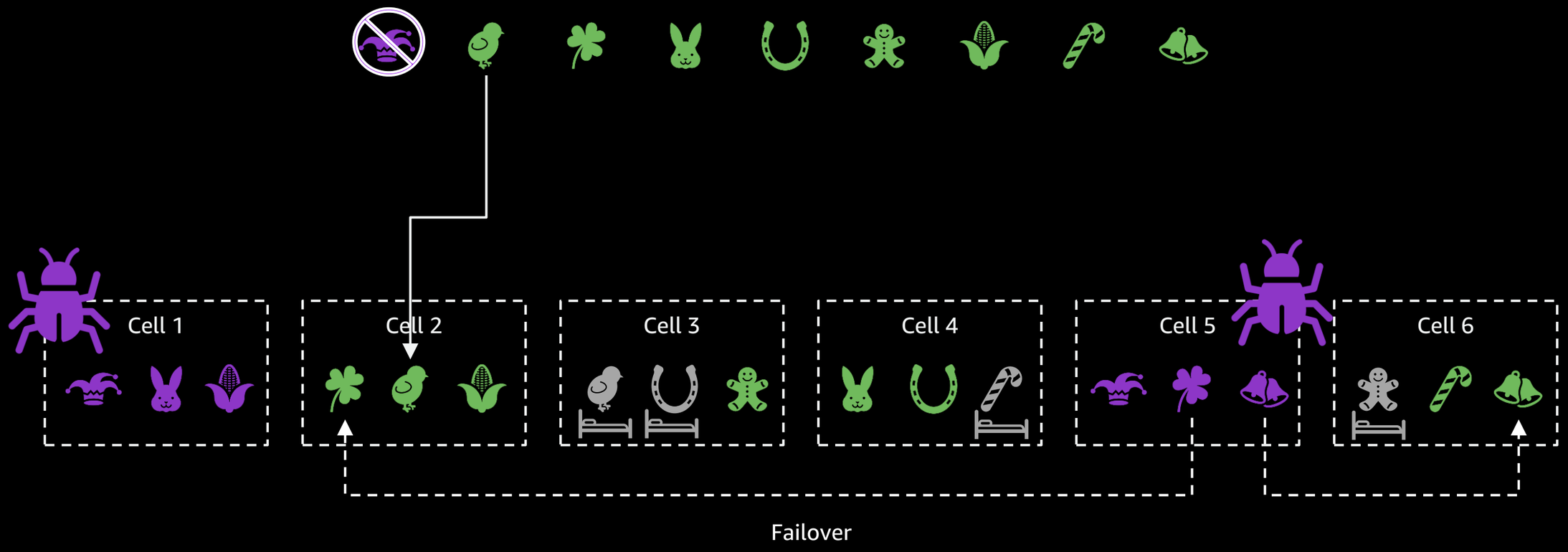
Poison pill / failover



Poison pill on failover



Poison pill / contained impact



Let’s do some math

$$\text{impact} = \frac{1}{C(|\text{cells}|,|\text{shards}|)}$$

Overlap	% impacted customers
0	53.6%
1	42.8%
2	3.6%

8 cells, 2 shards per customer

Overlap	% impacted customers
0	77%
1	21%
2	1.8%
3	0.06%
4	0.0006%
5	0.0000013%

100 cells, 5 shards per customer



Amazon Route 53

2048 cells, 4 shards per domain

1 chance in 730 billion

Placement

CONTROLLING THE CUSTOMER EXPERIENCE

Control cell allocation

Keeps **central repository** of usage

Understands customer constraints



Amazon Aurora



Amazon DynamoDB



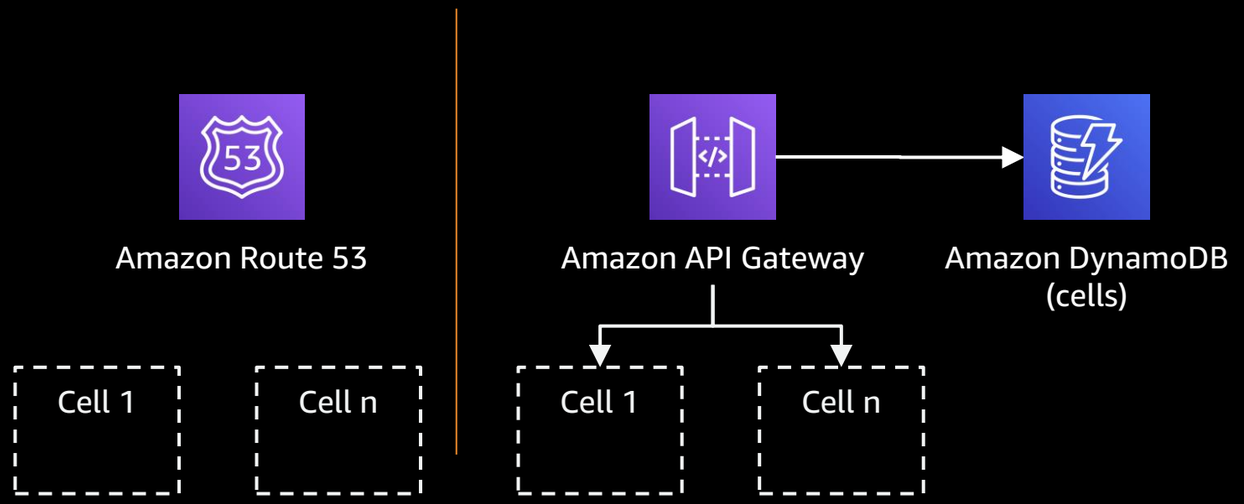
Router

THE THINNEST POSSIBLE LAYER

Discovery mechanism for clients

Fast and **reliable** (handles failures)

Clients need to be fault tolerant





Replication

KEEPING YOUR SHARDS UP TO DATE

Data driven



Amazon DynamoDB Streams

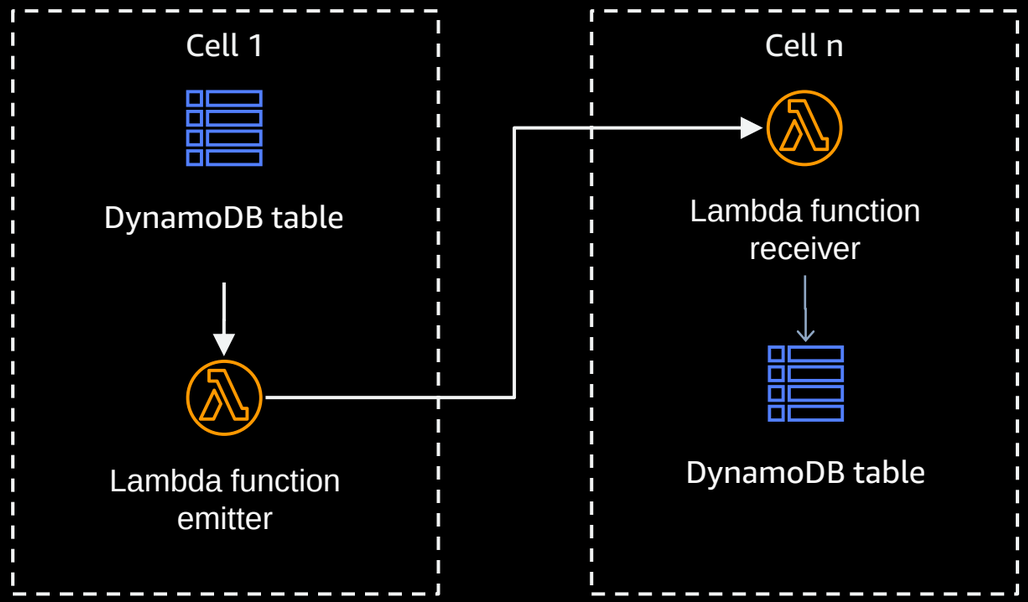
Event driven

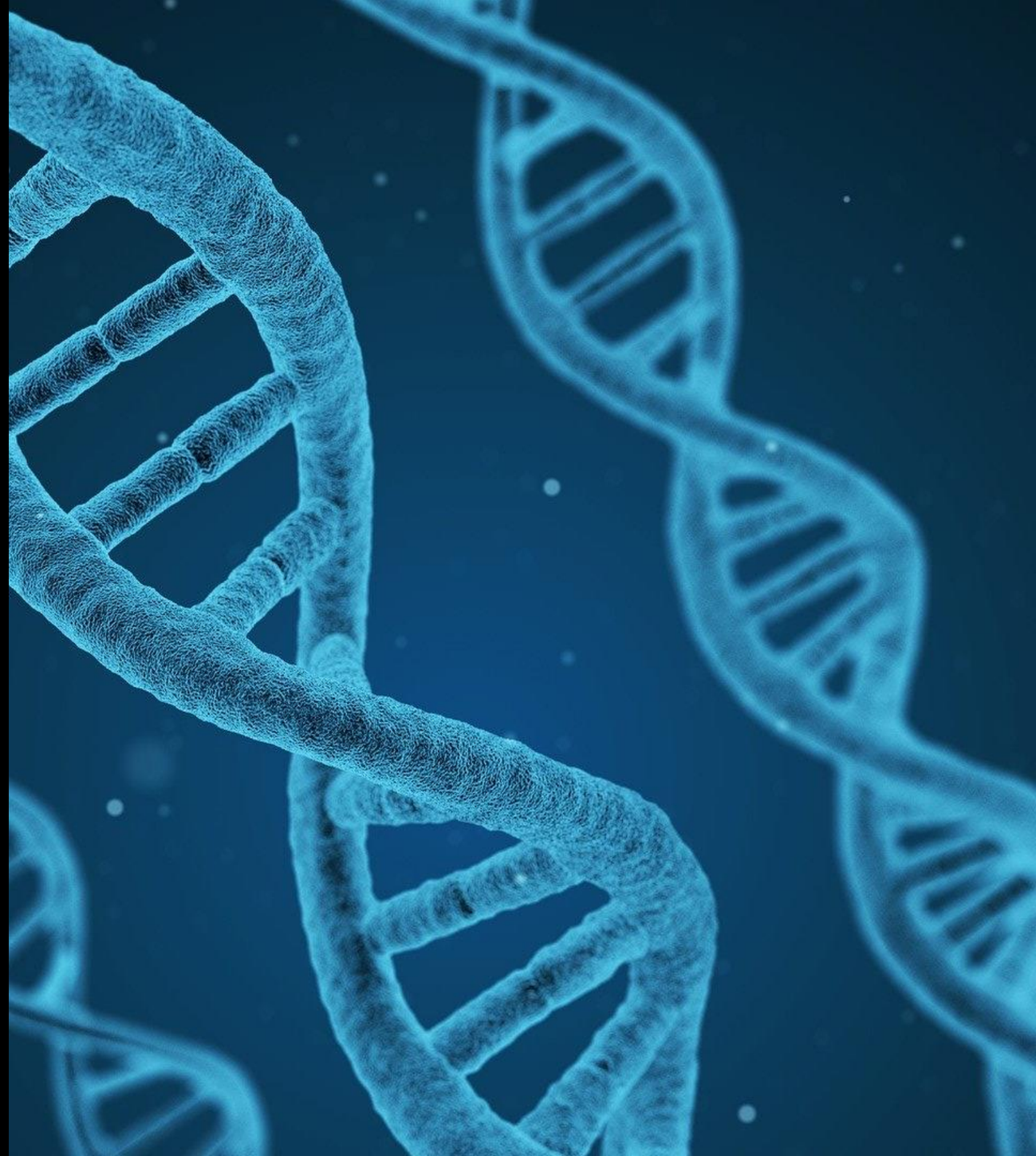


Amazon Quantum Ledger Database (Amazon QLDB)



Amazon EventBridge





Migration

MAINTAINING THE PLATFORM OPERATION

Noisy neighbor / “Whale”

Ongoing **optimization** strategies

Failure **recovery**



Control plane

PLATFORM COCKPIT

Controls **CRUD** and monitoring

API driven, no manual action

Deployment, CI / CD, A / B testing



AWS Cloud Development Kit (AWS CDK)



AWS CloudFormation



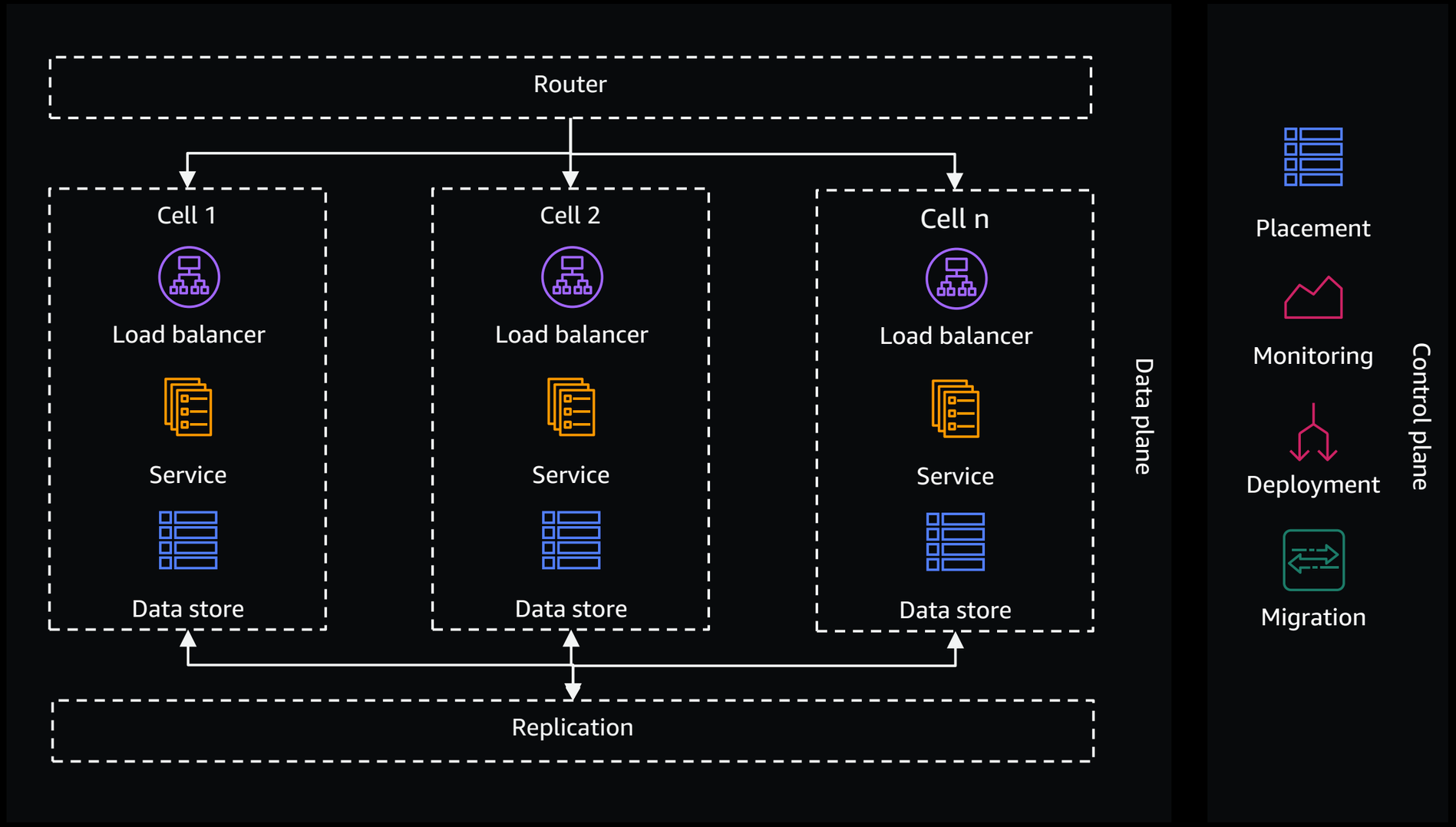
AWS CodePipeline



Amazon CloudWatch



Cell-based architectures, all in one

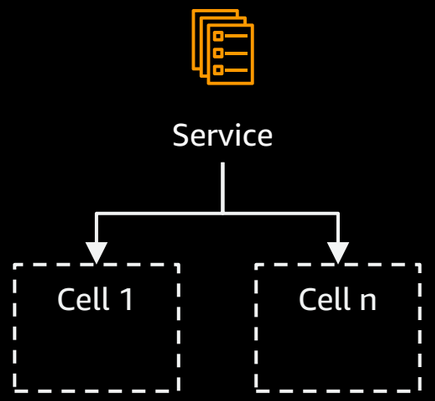


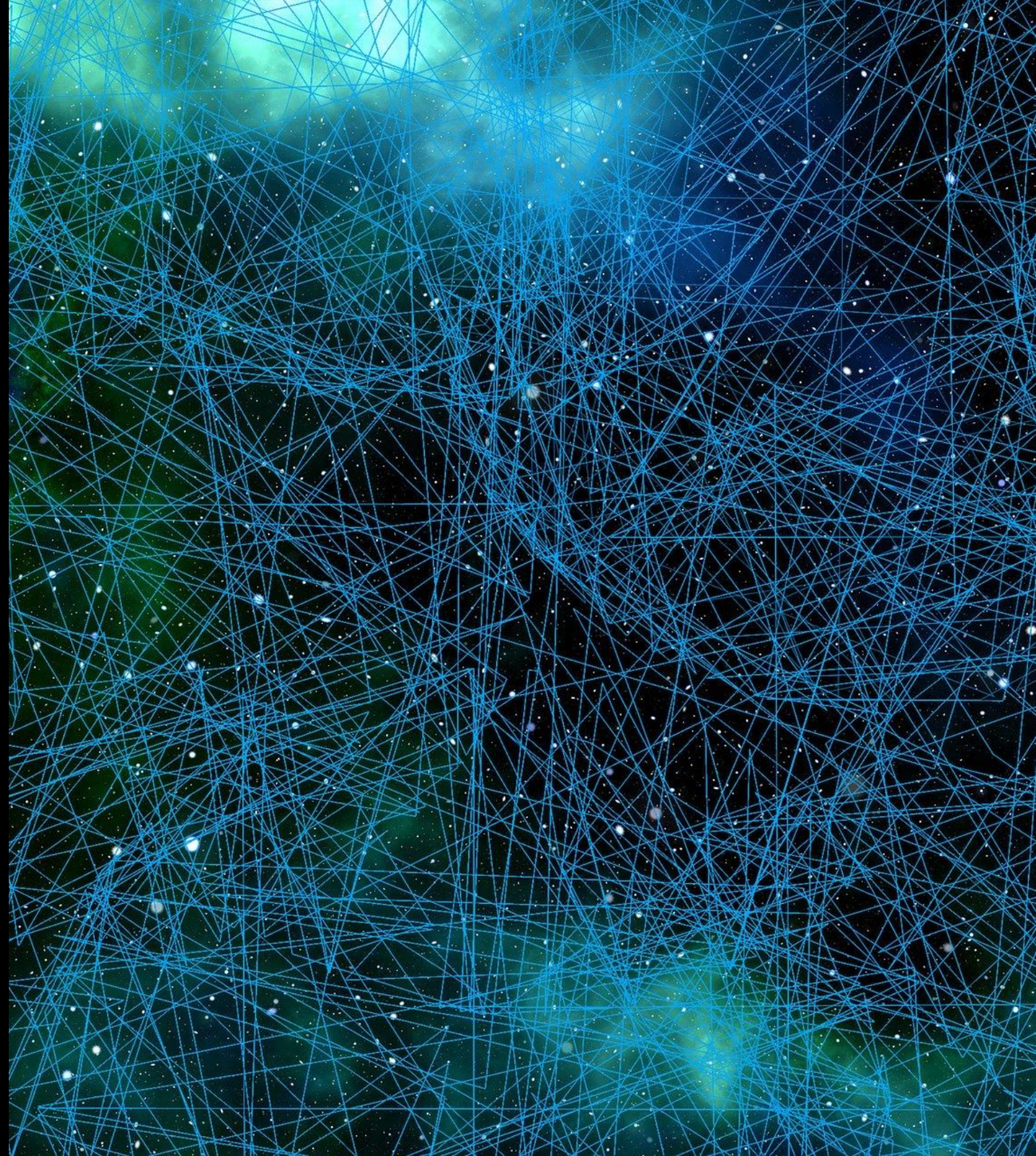
Multi-cell requests

Map reduce approach

Call the cells **external** endpoints

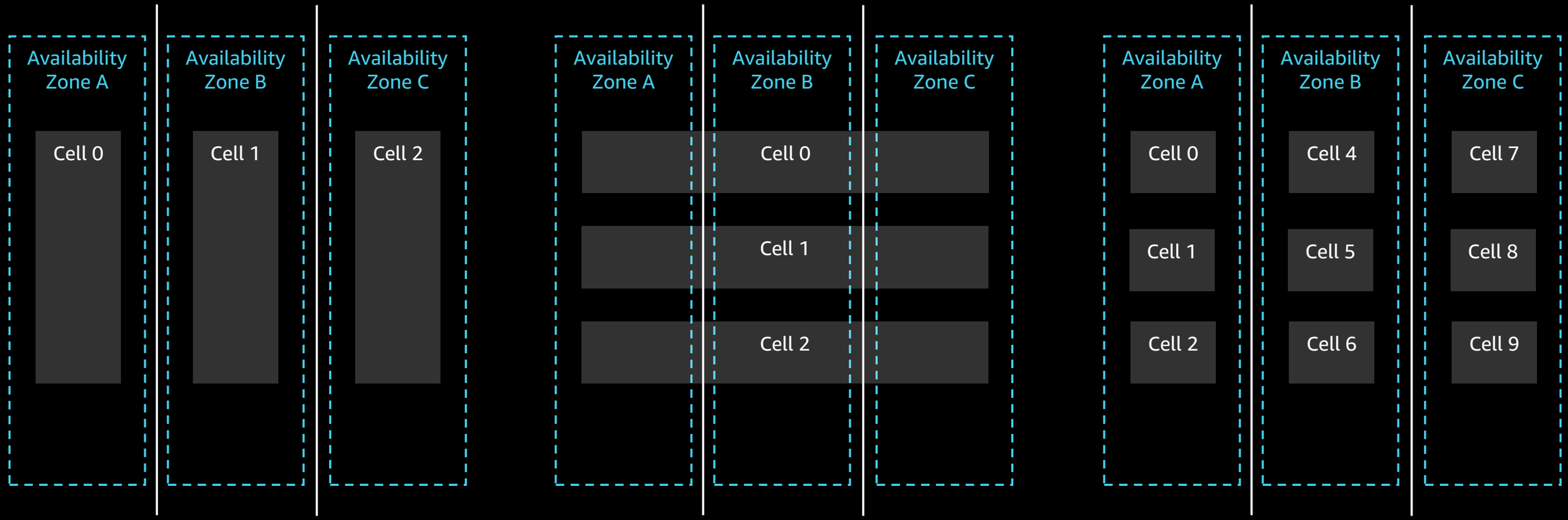
Service exists **outside** the cells





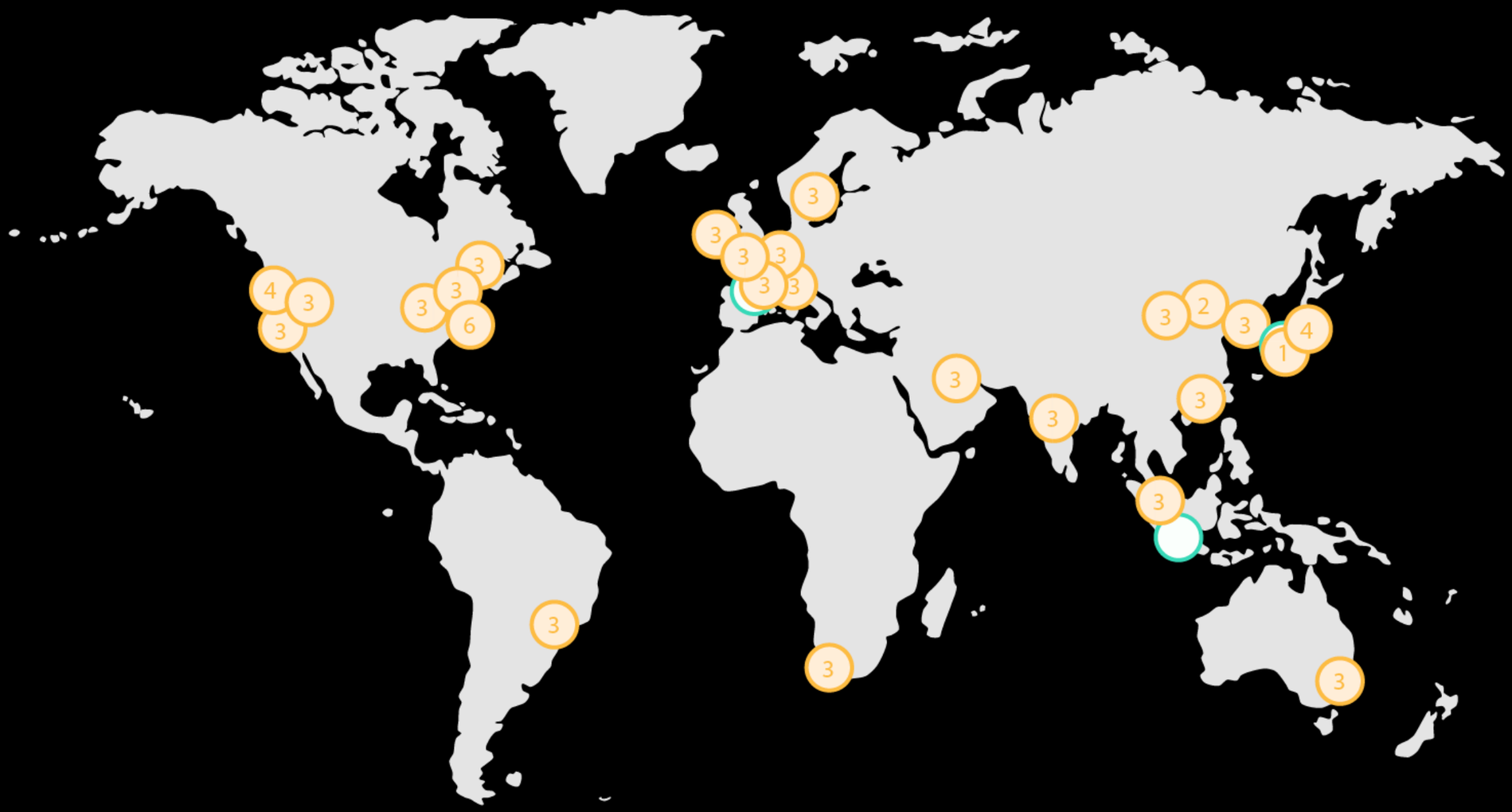
How does AWS help me?

BUILDING CELLS IN AN AWS REGION



How does AWS help me?

BUILDING CELLS ACROSS THE AWS REGIONS



Amazon Route 53

Amazon CloudFront

Amazon Aurora

Amazon DynamoDB

Next steps

MOVING FORWARD

Smaller cells and more shards

- **Optimizing** the platform

Cells and microservices

- Defining the “**grain**” of the cell

Multi-account strategies

- **Securing** cells

Cells and continuous deployment

- Deployment **pipeline**



Benefits

WHY CELL-BASED ARCHITECTURES?

Reliability **and** resiliency

Control of **scale**

Easier **error detection** (signal vs. noise)

Contained **blast radius**

More **configurable**

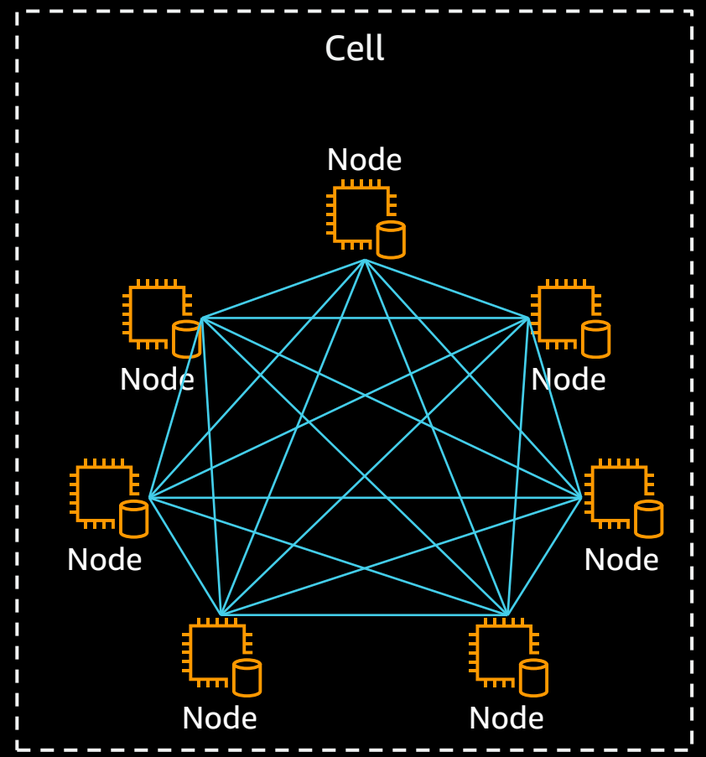


Extreme cells: Physalia

Created for Amazon Elastic Block Store (Amazon EBS)

Millions of cells / databases (one per volume)

Consistent, available, and fault tolerant



Diving deeper

Amazon Builder's Library: Workload isolation using shuffle-sharding

How Amazon Route 53 achieves resiliency and provides a 100% SLA using shuffle sharding

https://amzn.to/3naLjKo

Amazon EBS / Physalia: Millions of Tiny Databases

The design of Physalia, the backend data store build for Amazon EBS

https://bit.ly/2G1KOSg

Amazon Builder's Library: Timeouts, retries, and backoff with jitter

How to design client applications to gracefully handle downstream failures

https://amzn.to/3lhyImV

Architecture resources

AWS Well-Architected Framework

The official best practices for architecting in the AWS Cloud

https://aws.amazon.com/architecture/well-architected

AWS Well-Architected Labs

Hands-on labs to help you learn, measure, and build using architectural best practices

https://wellarchitectedlabs.com

AWS Architecture Center

Official AWS repository for all architecture resources

https://aws.amazon.com/architecture

AWS Solutions Library

Vetted reference implementations and well-architected patterns

https://aws.amazon.com/solutions

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.