AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

ARC301-R2

Testing resiliency of AWS workloads using chaos engineering

Seth Eliot
Principal Reliability Solutions Architect
AWS Well-Architected
AWS

Mahanth Jayadeva
Solutions Architect
AWS Well-Architected
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Automatically recover from failure
- Test recovery procedures
- Hands-on labs
- Making it real
- Bonus labs

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

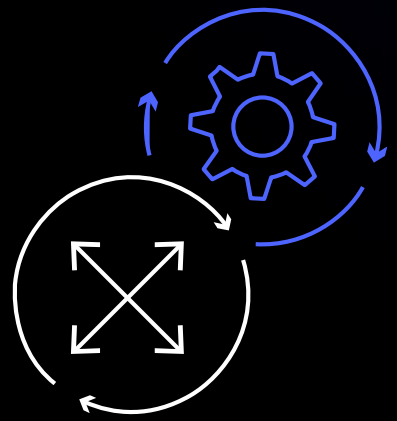


“We needed to build systems that embrace failure as a natural occurrence”

Werner Vogels
CTO
Amazon.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

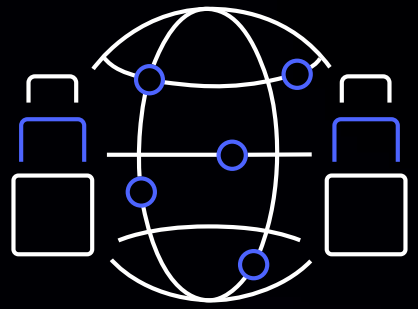
AWS Well-Architected



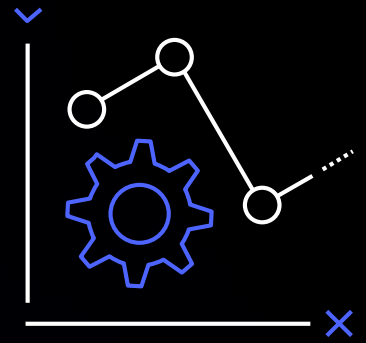
Operational excellence



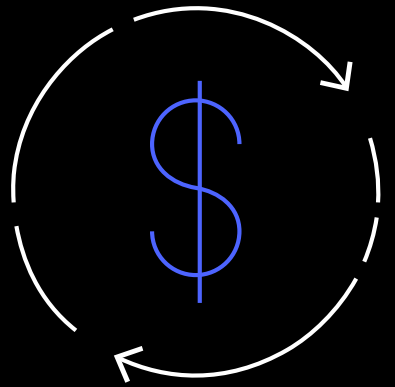
Security



Reliability



Performance efficiency



Cost optimization

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Testing resiliency

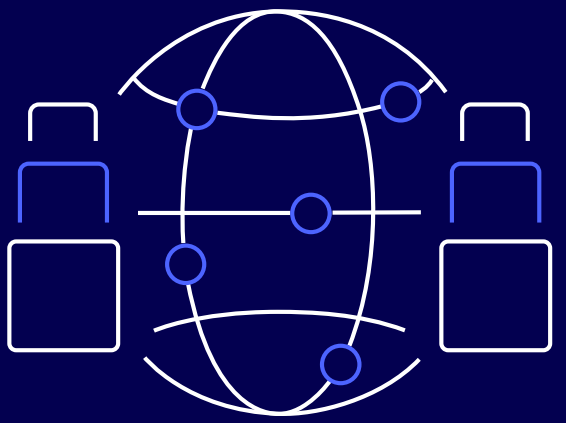
Resiliency

"The ability of a workload to recover from infrastructure or service disruptions . . ."

Design principles for reliability

- Automatically recover from failure
- Test recovery procedures

http://bit.ly/reliability-pillar



Automatically recover from failure

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fault isolation to protect your workload

AWS Well-Architected Framework

Add a link to your architectural design

REL 10. How do you use fault isolation to protect your workload? Info

Fault isolated boundaries limit the effect of a failure within a workload to a limited number of components. Components outside of the boundary are unaffected by the failure. Using multiple fault isolated boundaries, you can limit the impact on your workload.

Question does not apply to this workload Info

Select from the following

- [x] Deploy the workload to multiple locations Info
- [ ] Automate recovery for components constrained to a single location Info



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Regions and Availability Zones (AZs)

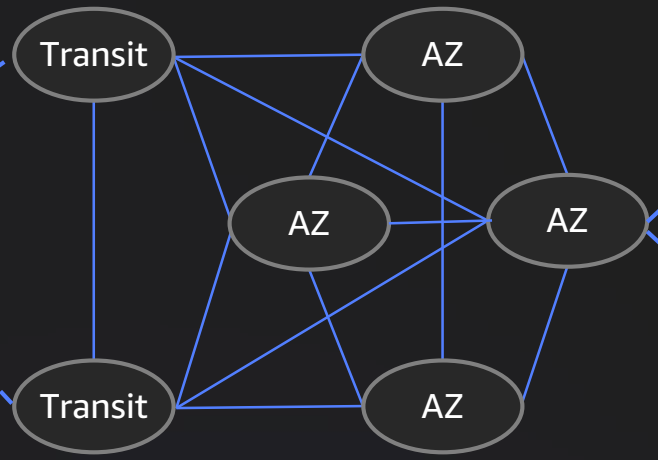
AWS REGIONS ARE PHYSICAL LOCATIONS AROUND THE WORLD WHERE WE CLUSTER DATA CENTERS

25 AWS Regions worldwide



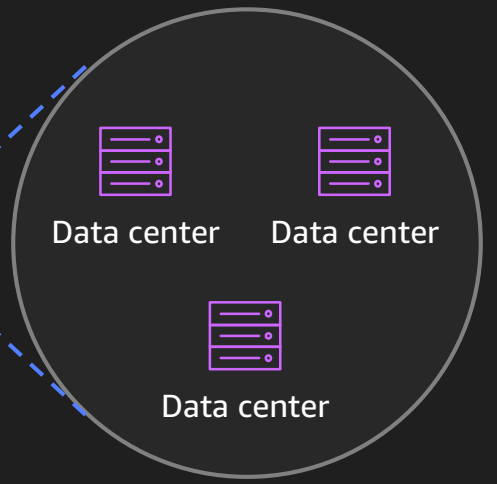
- AWS Regions
- Announced Regions

Each AWS Region has multiple AZs



A **Region** is a physical location in the world

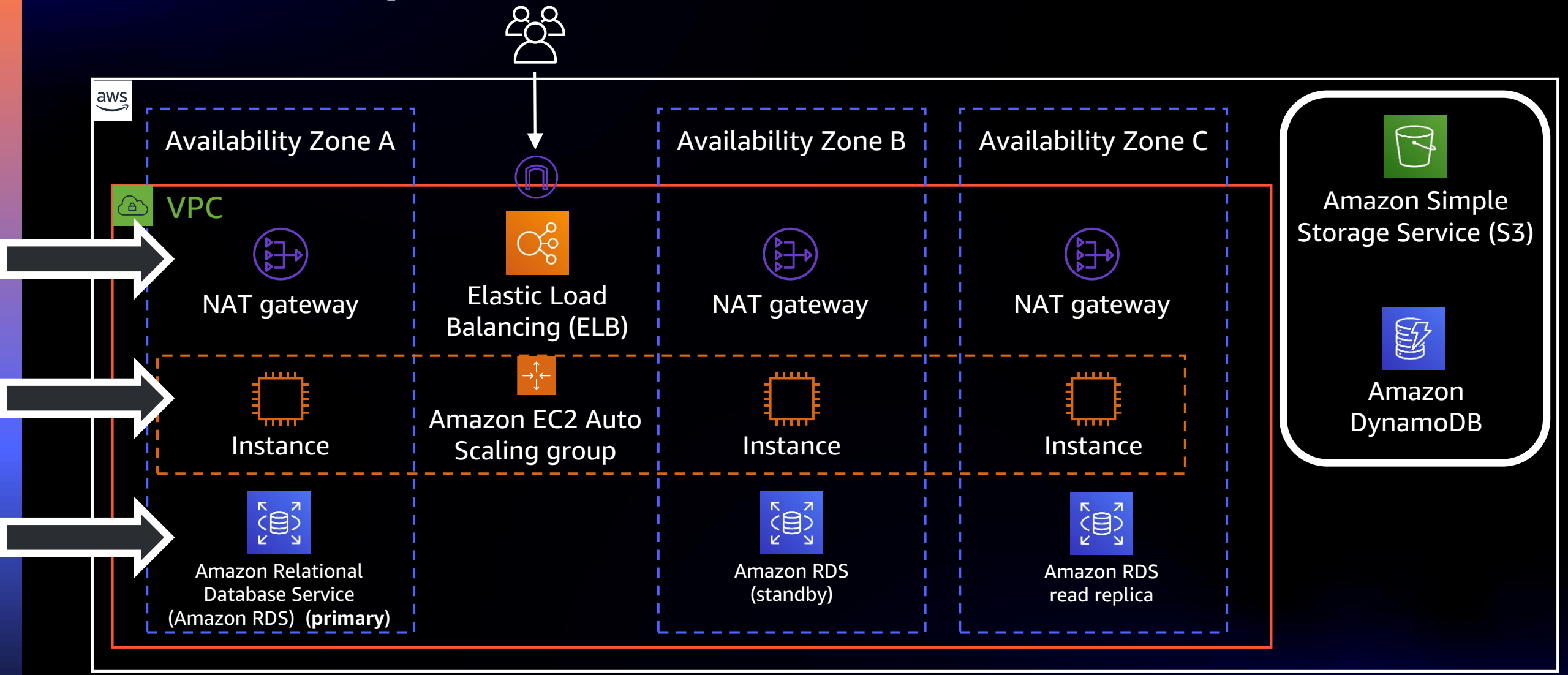
Each AZ is one or more discrete data centers



Data centers, each with redundant power, networking, and connectivity, are housed in separate facilities

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

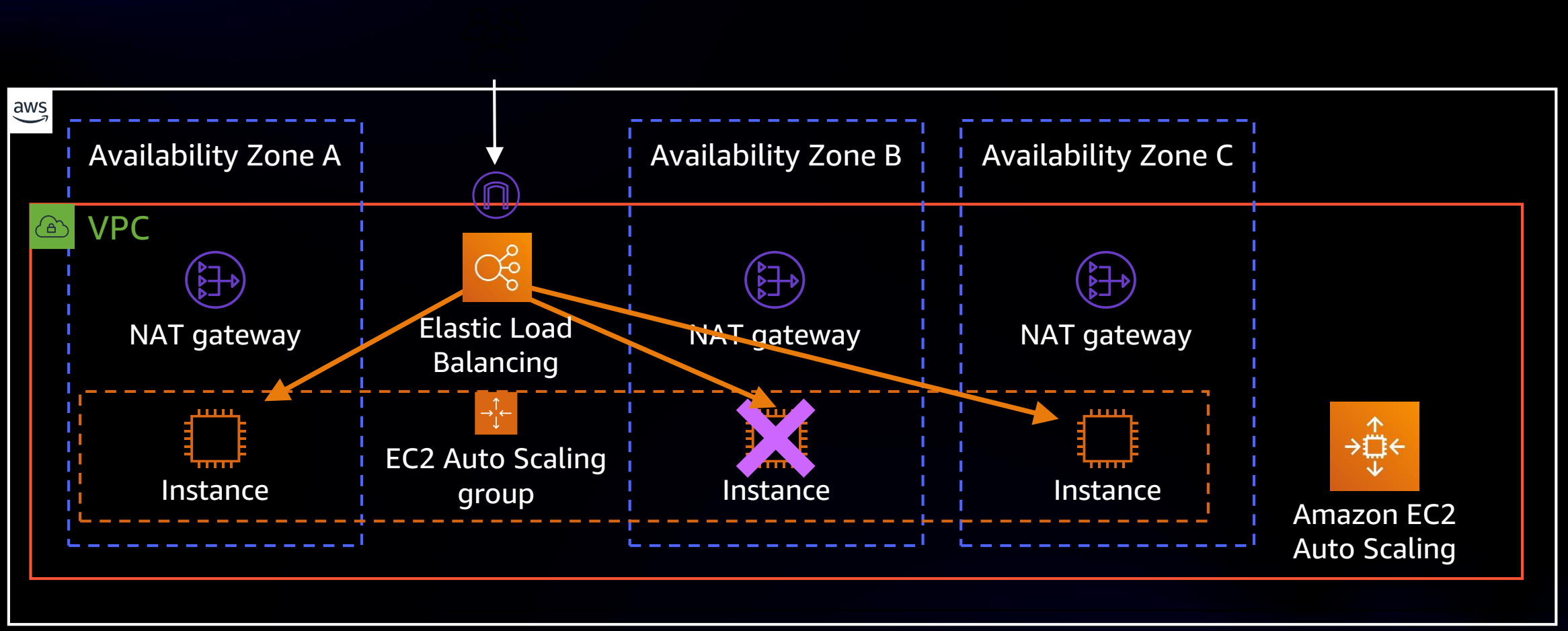
Three tiers, Multi-AZ



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Elastic Load Balancing and auto recovery

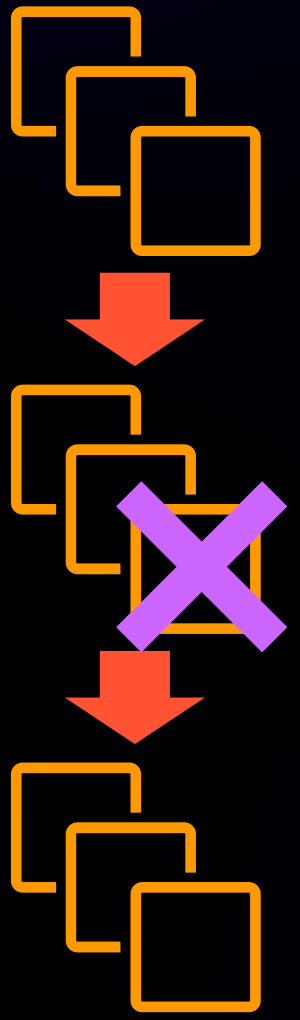


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon EC2 Auto Scaling and auto recovery



Amazon EC2
Auto Scaling



REL 5. How do you design interactions in a distributed system to mitigate or withstand failures? Info

Distributed systems rely on communications networks to interconnect components (such as servers or services). Your workload must operate reliably despite data loss or latency over these networks. Components of the distributed system must operate in a way that does not negatively impact other components or the workload. These best practices enable workloads to withstand stresses or failures, more quickly recover from them, and mitigate the impact of such impairments. The result is improved mean time to recovery (MTTR).

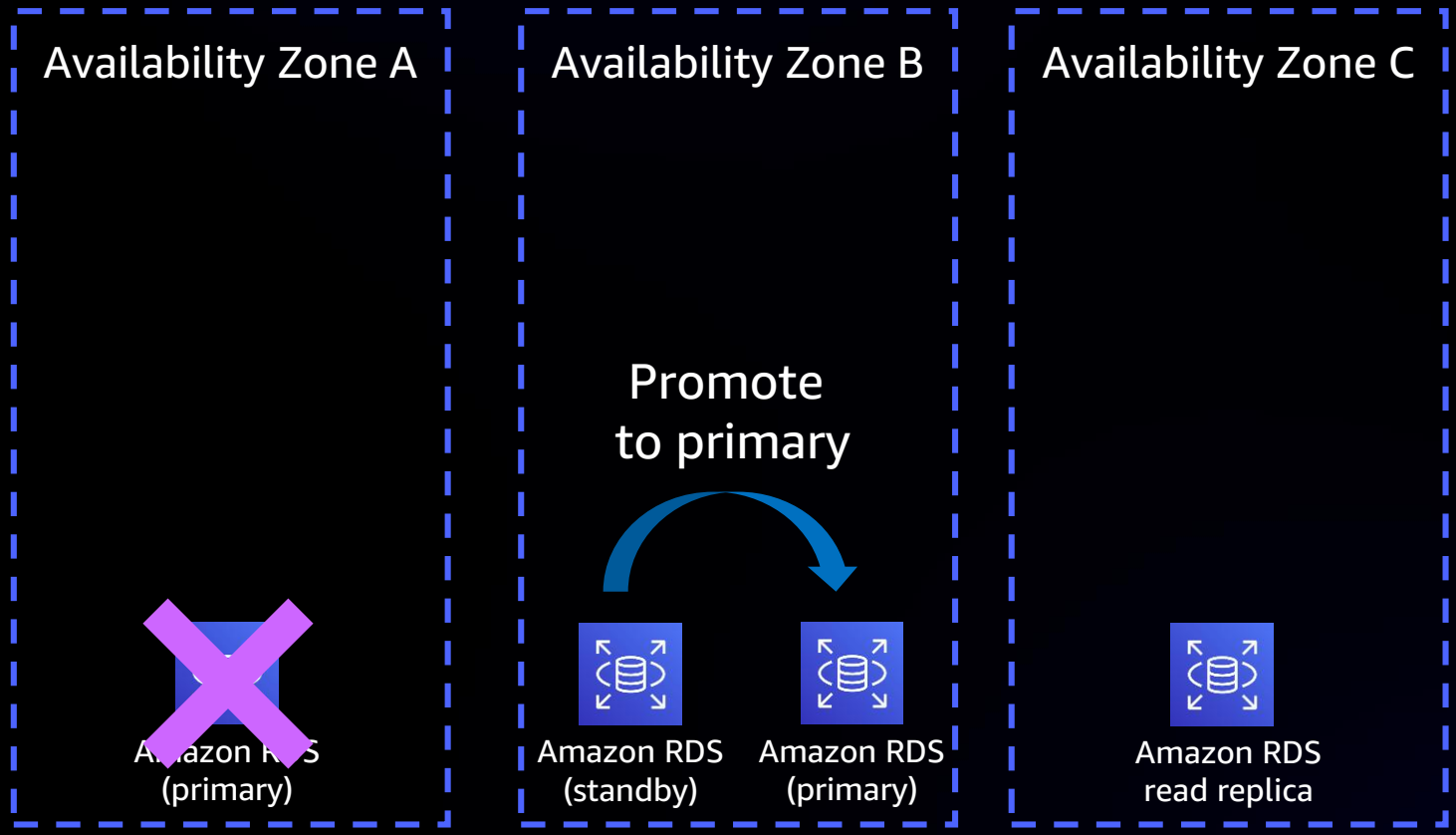
Question does not apply to this workload Info

Select from the following

- [x] Implement graceful degradation to transform applicable hard dependencies into soft dependencies Info
- [x] Throttle requests Info
- [x] Control and limit retry calls Info
- [x] Fail fast and limit queues Info
- [x] Set client timeouts Info
- [x] Make services stateless where possible Info

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon RDS and auto recovery



Amazon RDS

Amazon Aurora

Does not have standby

Can promote *any* read replica to primary

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Test recovery procedures with chaos engineering

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

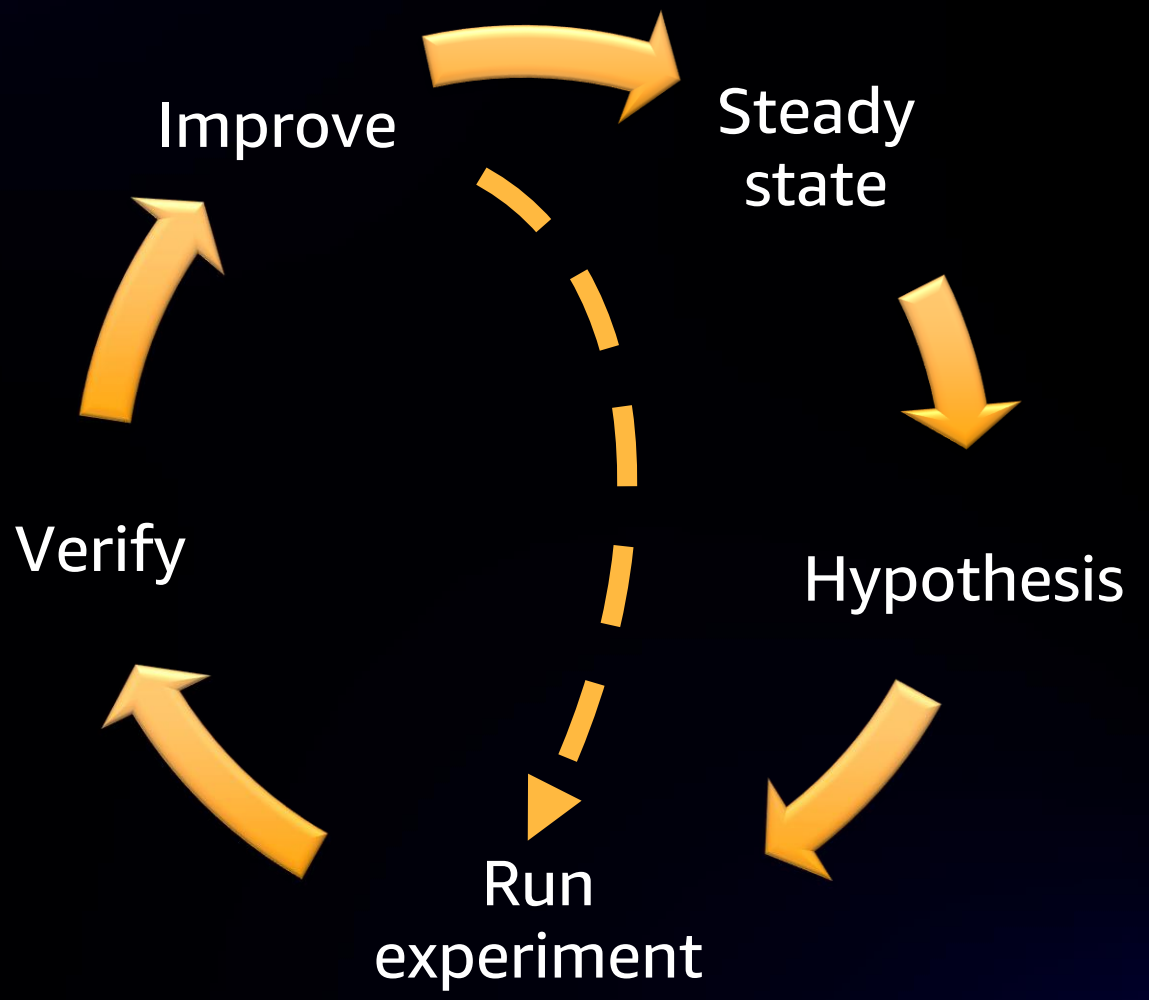
“Chaos Engineering is the discipline of experimenting on a system in order to build confidence in the system’s capability to withstand turbulent conditions in production.”

“Principles of Chaos Engineering”
http://principlesofchaos.org

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

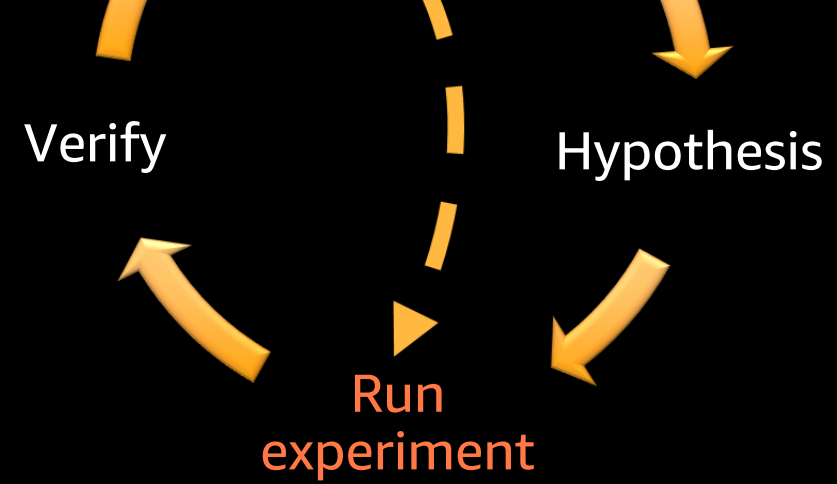
Chaos engineering

A SCIENTIFIC METHOD



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Chaos experiment



Inject events that simulate

- Hardware failures, such as servers dying
- Software failures, such as malformed responses
- Nonfailure events, such as spikes in traffic or scaling events

Any event capable of disrupting steady state





aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Chaos engineering is not new

2005

Amazon Game Days

Train how to respond to disaster



2010

Netflix Chaos Monkey



2016

Principles of Chaos

Chaos Engineering



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tools for chaos engineering



This workshop



Script it yourself



AWS Fault Injection Simulator (AWS FIS)







Open source and commercial products

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Managed service or custom scripts?

Managed service

License/usage fees apply

Provider builds and maintains

Benefit of provider experience

Features driven by customer demand

Custom scripts

No license/usage fees

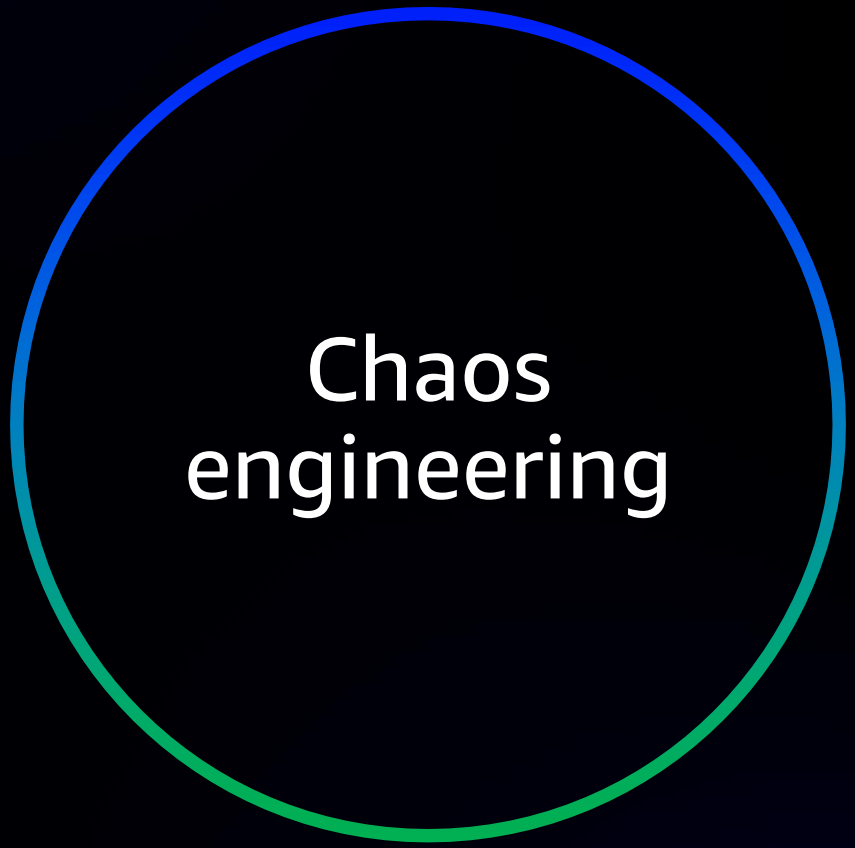
You build and maintain

Reliant on your experience

Complete control of development

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Managed service or custom scripts?



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why use scripts today?



Script it yourself

- Lab scripts are *examples*
- Show how *you* can write it yourself
- Gain confidence. Reduce fear
- Good place to start

Lab includes examples in

- Bash shell, Python, Java, PowerShell, and C#

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why use AWS FIS today?



AWS Fault Injection Simulator (AWS FIS)

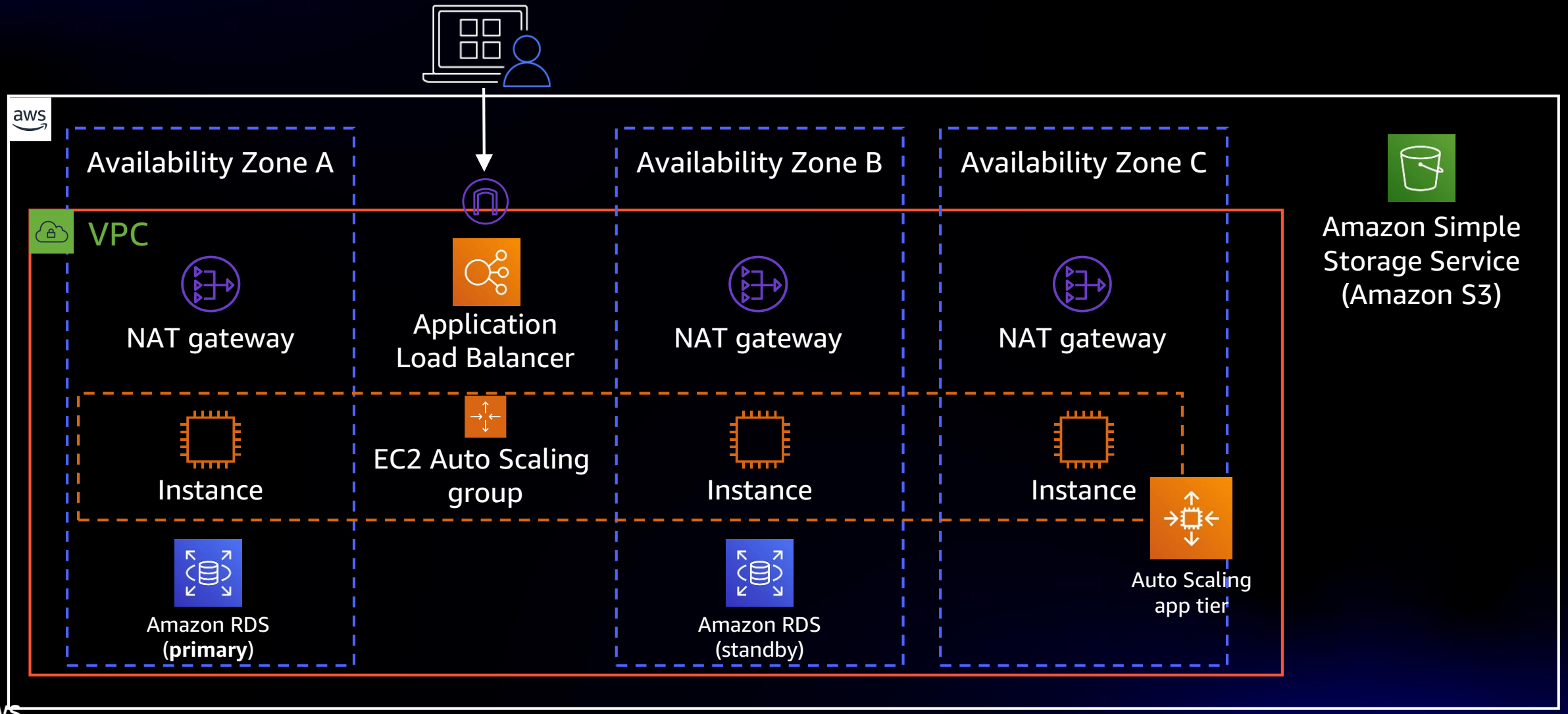
- AWS native integration
- AWS integrated security
- Prebuilt events/actions
- Define when to stop or roll back

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Chaos engineering experiment

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Our architecture for this workshop



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Website demo



Welcome to the Resiliency Workshop!

Data from the metadata API

account_id: [redacted]
ami_id: ami-00dfe2c7ce89a450b
availability_zone: us-east-2a
instance_id: i-00b9245c5b85828bb
instance_type: t2.micro
private_hostname: ip-10-0-55-56.us-east-2.compute.internal
private_ipv4: 10.0.55.56

click here to go to other page



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Roadmap for the rest of this session

Hands-on labs

- Amazon EC2 failure experiment
- Amazon RDS failure experiment

Presentation

- Making it real

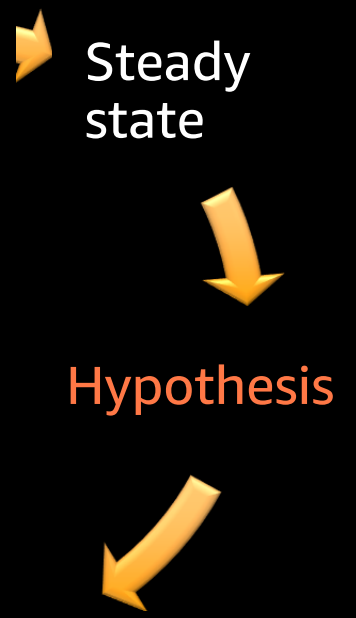
Hands-on labs (bonus)

- Application failure experiment
- AZ failure experiment – do this *last*



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Component failure experiments



Hypothesis
If one EC2 instance dies, then availability will not be impacted

Hypothesis
If primary RDS instance dies, then availability will not be impacted

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Hands-on lab

https://bit.ly/wa-chaos

Throughout the lab, *always* follow the instructions for

- "If you are attending an in-person workshop . . ."
- Single region

Step 1. Log into your AWS account

2c24-1e6769fee4-57

Step 2. Configure environment

Java C# Python Bash PowerShell

Step 3. Prepare for failure injection

Step 4. EC2 failure injection

- Using script
- Skip Step 4.3 (FIS); it is optional

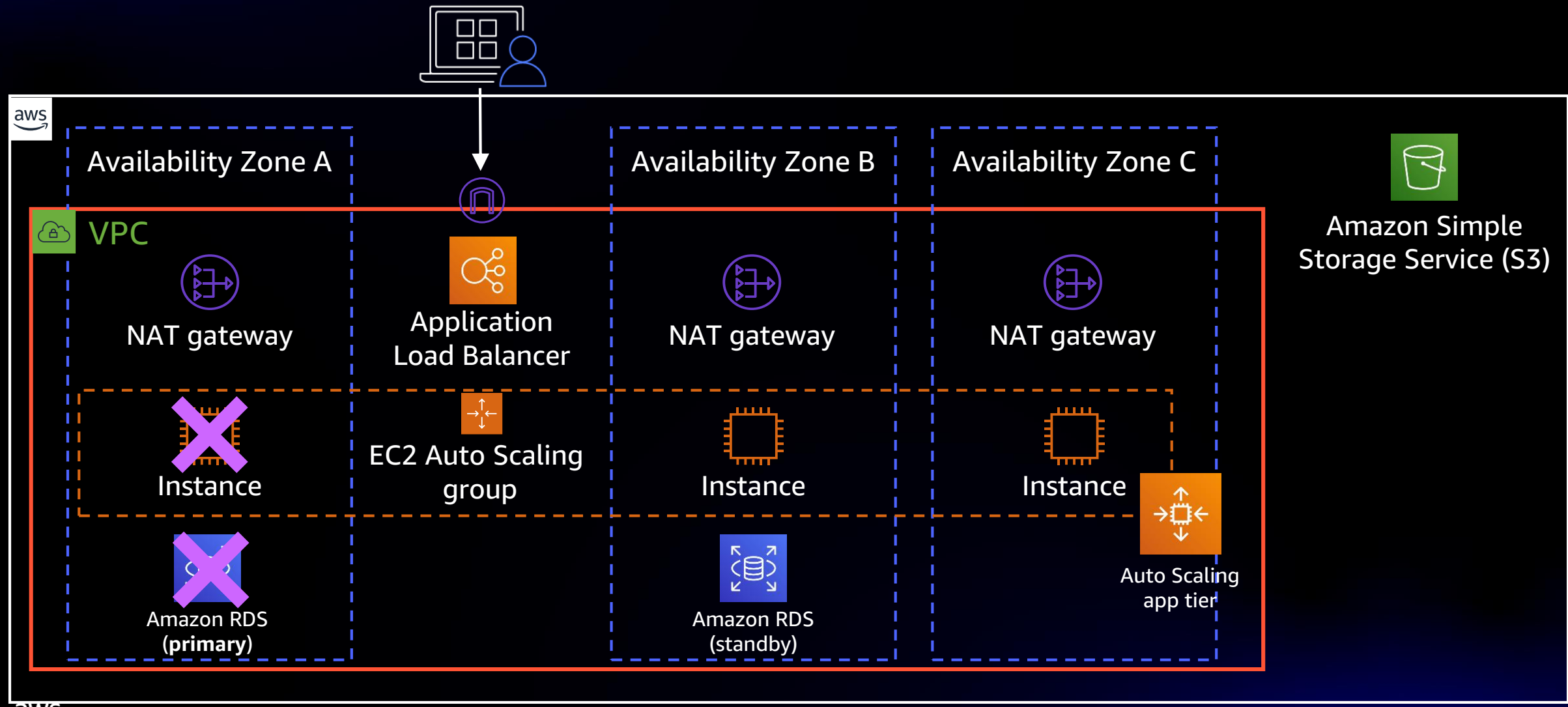
Step 5. RDS failure injection

- Using script
- Using FIS

aws

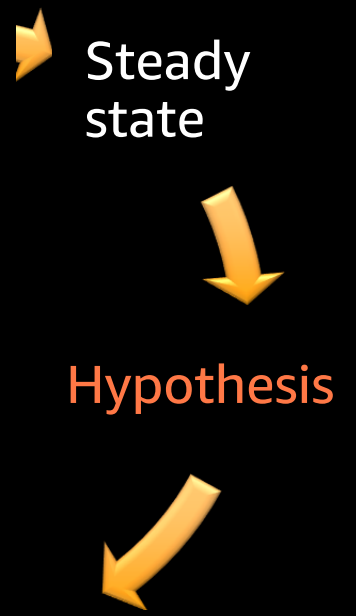
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Simulated fault experiments



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Component failure: Amazon EC2



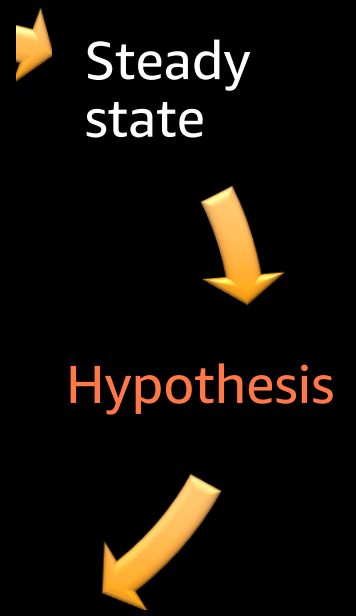
Hypothesis
If one EC2 instance dies, then availability will not be impacted



- ELB detects unhealthy instance and reroutes requests
- Load adequately served by remaining two EC2 instances
- Auto scaling will replace unhealthy instance in same AZ

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Component failure: Amazon RDS

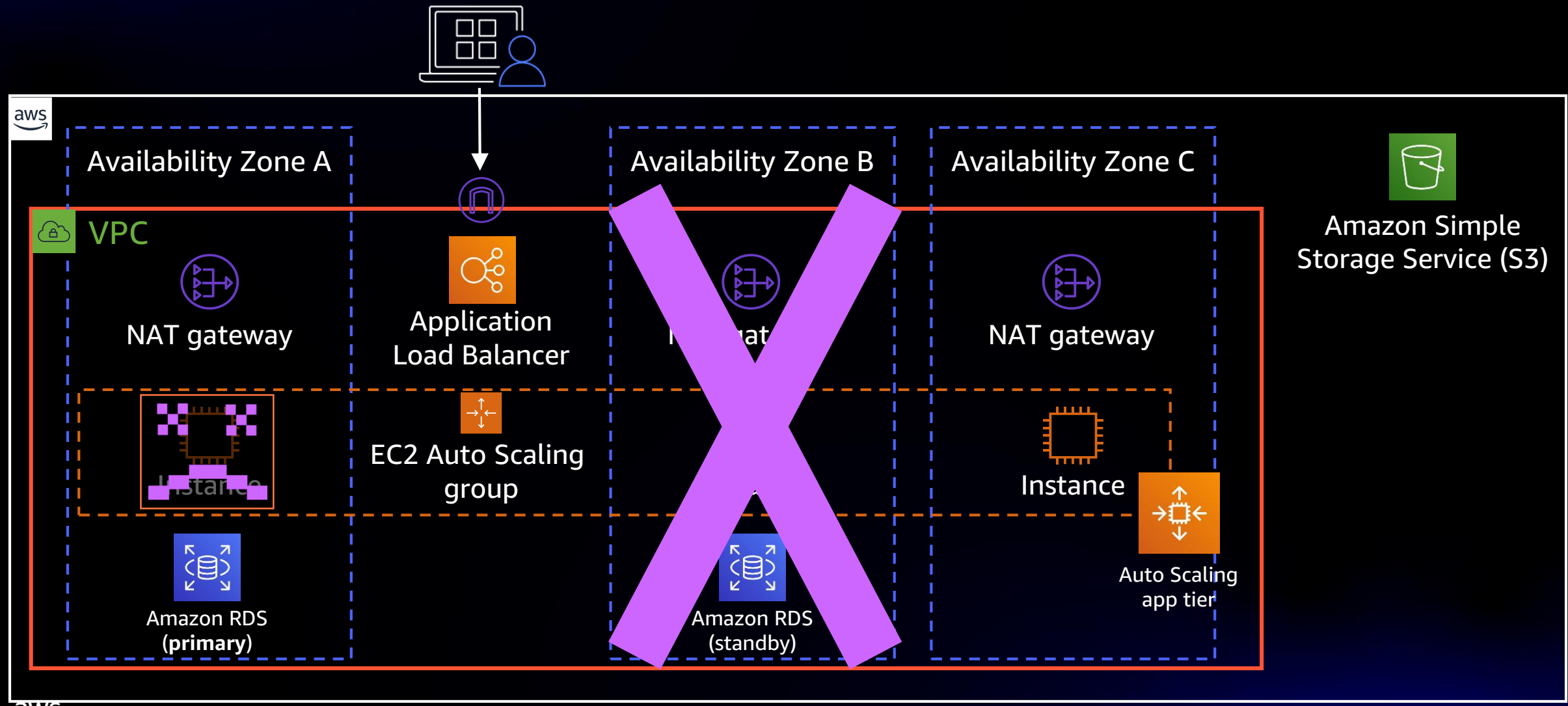


Hypothesis
If primary RDS instance dies, then availability will not be impacted

- RDS detects unhealthy instance and promotes standby instance
- Application connects to newly promoted primary

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Later: Bonus simulated fault experiments



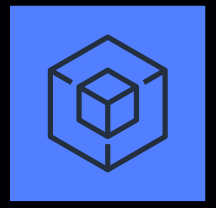
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How the scripts work to simulate failures

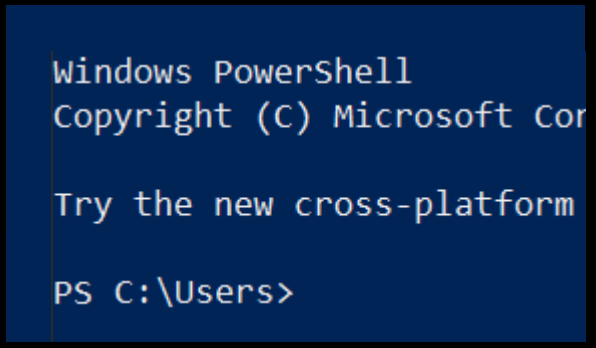
AWS Command Line Interface (AWS CLI)



AWS SDK



AWS Tools for PowerShell



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Making it real

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“Reality . . . what a concept.”

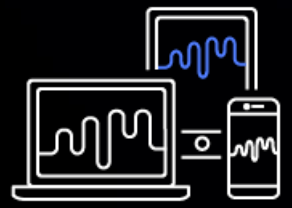
Robin Williams

Actor and comedian

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reality is production with actual users

- Your recovery design is based on *assumptions*
- Reality may differ
- Test close to production



Traffic patterns

Record/replay

Shadow testing

Test in production



Environment

Cloud-deployed

Test in production



Events

Are simulations

Learn from production

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Game days

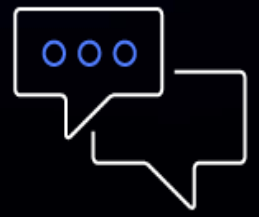
SIMULATE FAILURE OR EVENT TO TEST SYSTEMS' RESILIENCY, PROCESSES, AND TEAM RESPONSES



People

Cross-discipline team

Processes



Briefing

Overview

Roles



Planning

Preparation

Hypothesis



Execution

Run experiment



Analysis

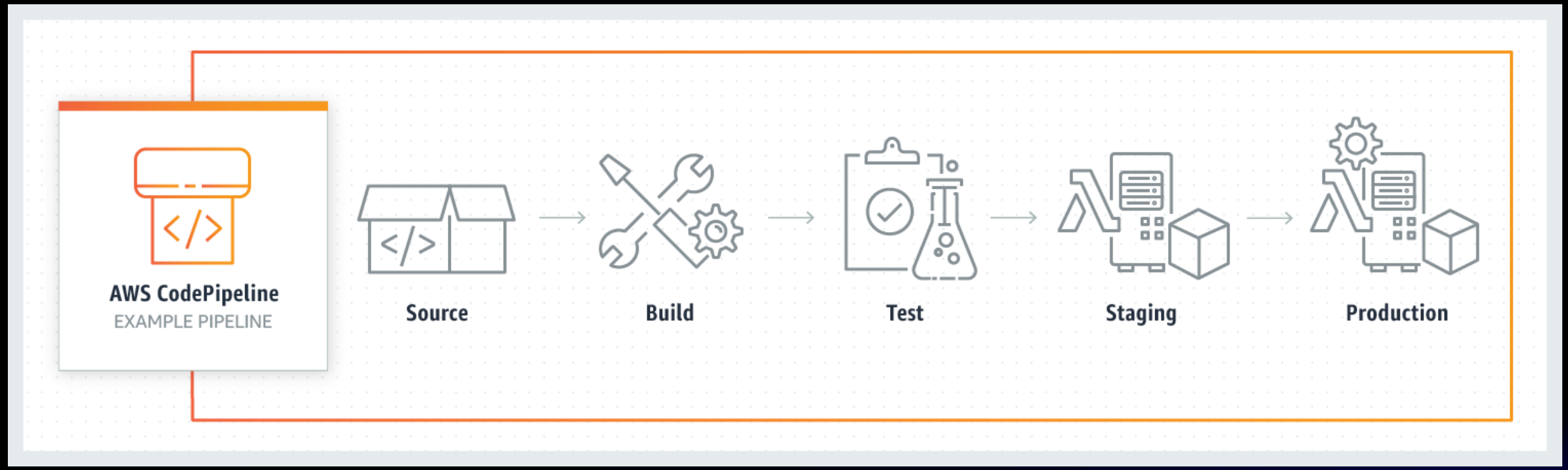
Verify

Improve

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

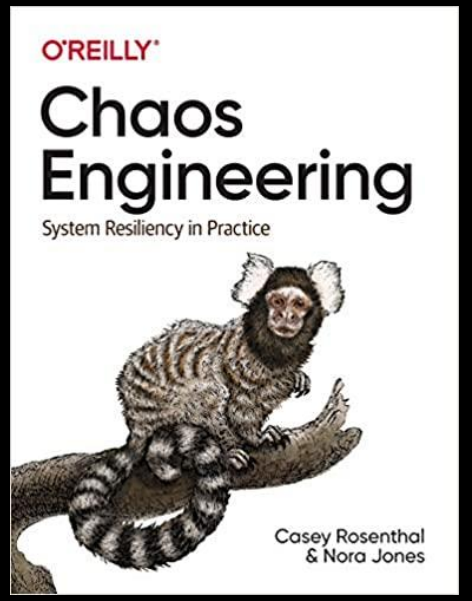
Run these tests regularly

- As part of CI/CD pipeline
- Start with staging (test, pre-production) environments
- Consider which tests to run in production



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“This isn’t about creating chaos. It is about making the chaos inherent in the system visible.”



Casey Rosenthal and Nora Jones

Chaos Engineering: System Resiliency in Practice

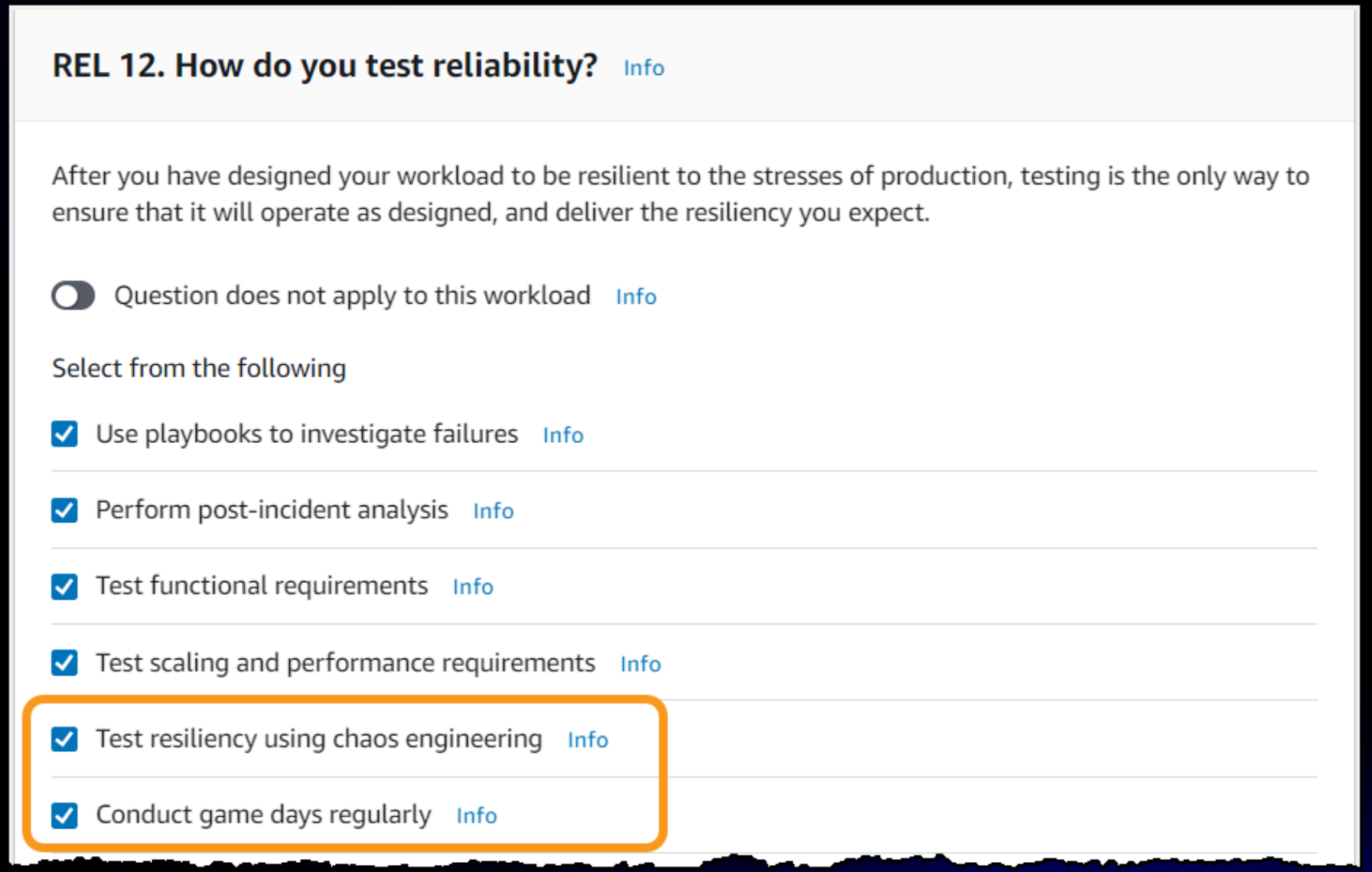
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Next steps

You *can* do this

You *should* do this



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Well-Architected labs are open source

Contributions welcome

Go to the lab and click on this link



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Resources



AWS Well-Architected

ARC304 [Chalk Talk]: Automation safety: How to avoid breaking things at scale

FSI315 [Builders' Session]: Test resiliency in financial services with chaos engineering

BOA310 [Workshop]: Resilient and well-architected apps with chaos engineering

DOP303 [Builders' Session]: Assessing your application resiliency using chaos engineering

Whitepaper: Reliability Pillar - AWS Well-Architected Framework

bit.ly/reliability-pillar

Well-Architected hands-on labs

wellarchitectedlabs.com/reliability

Principles of chaos engineering

https://principlesofchaos.org/

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Final steps



With any remaining time, feel free to run these bonus labs

Step 6. Application failure injection

Step 7. AZ failure injection

Do this last

No need to clean up – just log out

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Seth Eliot

Twitter: @setheliot

Mahanth Jayadeva

Twitter: @MahanthJayadeva

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.