



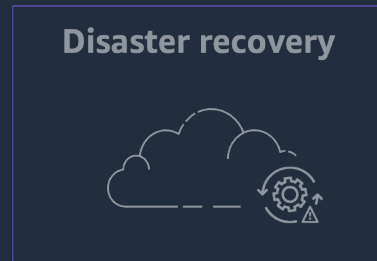
Well-Architected Reliability:

Best practices you can use today

Danny Wright (he/him)

Principal Technical Account Manager
AWS Enterprise Support

Continuous Resilience Roadmap



Best Practices

Well-Architected Framework

Resilience Modeling

Resilience Hub
Chaos Engineering

Recovery Planning

Disaster Recovery

Incident Analysis

Correction Of Errors

Agenda

Well-Architected Reliability

- Design principles
- Best practices

Key takeaways and helpful resources

Answer *your* questions

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

Werner Vogels
CTO, Amazon.com



Definitions

Reliability

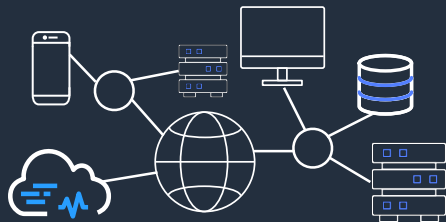
Ability of a workload to perform its required function correctly and consistently...

Resilience

Ability of a workload to recover from infrastructure or service disruptions...

– Reliability Pillar, AWS Well-Architected Framework

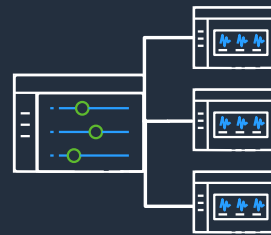
High Reliability is Now the New Normal



Complex ecosystems



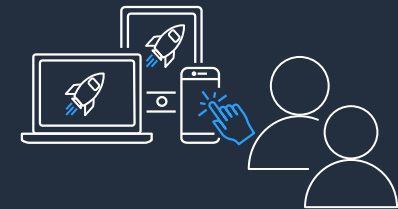
Multi-disciplinary teams



Incremental releases



Specialized services

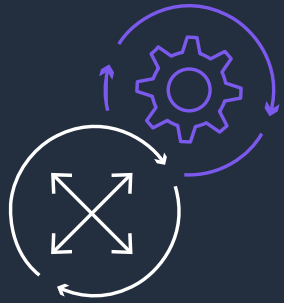


Expanding channels



Always on

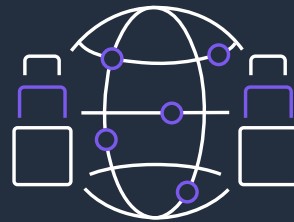
Six Pillars of AWS Well-Architected Framework



Operational
excellence



Security



Reliability



Performance
efficiency



Cost
optimization



Sustainability

Well-Architected is a Set of Best Practices



Design principles



Questions



Best Practices



Whitepaper



Labs

REL 12. How do you test reliability? [Info](#)

After you have designed your workload to be resilient to the stresses of production, testing is the only way to ensure that it will operate as designed, and deliver the resiliency you expect.

Question does not apply to this workload [Info](#)

Select from the following

- Use playbooks to investigate failures [Info](#)
- Perform post-incident analysis [Info](#)
- Test functional requirements [Info](#)
- Test scaling and performance requirements [Info](#)
- Test resiliency using chaos engineering [Info](#)
- Conduct game days regularly [Info](#)

Well-Architected Tool

AWS Well-Architected Tool

REL 12. How do you test reliability? [Info](#)

After you have designed your workload to be resilient, ensure that it will operate as designed, and deliver the required level of reliability.

Question does not apply to this workload

Select from the following

- Use playbooks to investigate failures [Info](#)
- Perform post-incident analysis [Info](#)
- Test functional requirements [Info](#)
- Test scaling and performance requirements
- Test resiliency using chaos engineering [Info](#)
- Conduct game days regularly [Info](#)

REL 11. How do you design your workload to withstand component failures? [Info](#)

Workloads with a requirement for high availability and low mean time to recovery (MTTR) must be architected for resiliency.

Question does not apply to this workload [Info](#)

Select from the following

- Monitor all components of the workload to detect failures [Info](#)
- Fail over to healthy resources [Info](#)
- Automate healing on all layers [Info](#)
- Use static stability to prevent bimodal behavior [Info](#)
- Send notifications when events impact availability [Info](#)

Best Practices: Well-Architected Reliability



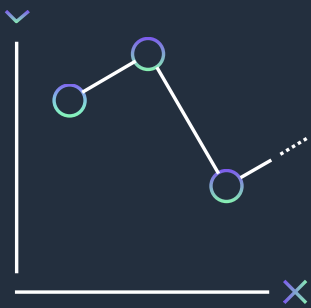
Four Areas | 13 “Questions”



Foundations



Workload architecture

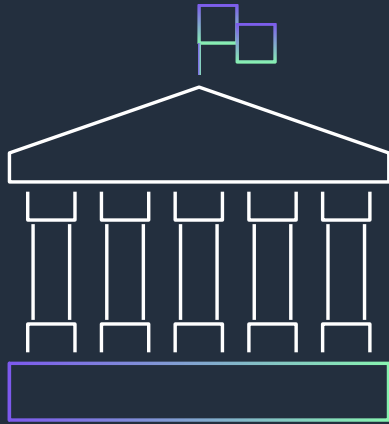


Change management



Failure management

Well-Architected Reliability Pillar



Foundations

REL 1

Service quotas and limits

- ⌘ Know your AWS resource quotas
- ⌘ Raise quotas where necessary
- ⌘ Work within hard limits

REL 2

Network topology

- ⌘ Highly available endpoints
- ⌘ Redundant connections
- ⌘ Size and layout of network

Manage Service Quotas

Dashboard

- Amazon Athena**
Total quotas: 4
- Amazon Dynamo**
Total quotas: 9
- Amazon Elastic Compute Cloud (Amazon EC2)**
Total quotas: 103
- Amazon Relation Service (Amazon**
Total quotas: 27
- AWS CloudFormation**
Total quotas: 24
- AWS Key Manage (AWS KMS)**
Total quotas: 52

Service Quotas > AWS services > AWS Lambda

AWS Lambda

Service quotas [Request quota increase](#)

Find quotas < 1 >

Quota name	Applied quota value	AWS default quota value	Adjustable
Asynchronous payload	Not available	256 kilobytes	No
Burst concurrency	Not available	1,000	No
Concurrent executions	1,000	1,000	Yes

Pending service quota requests (0) < 1 >

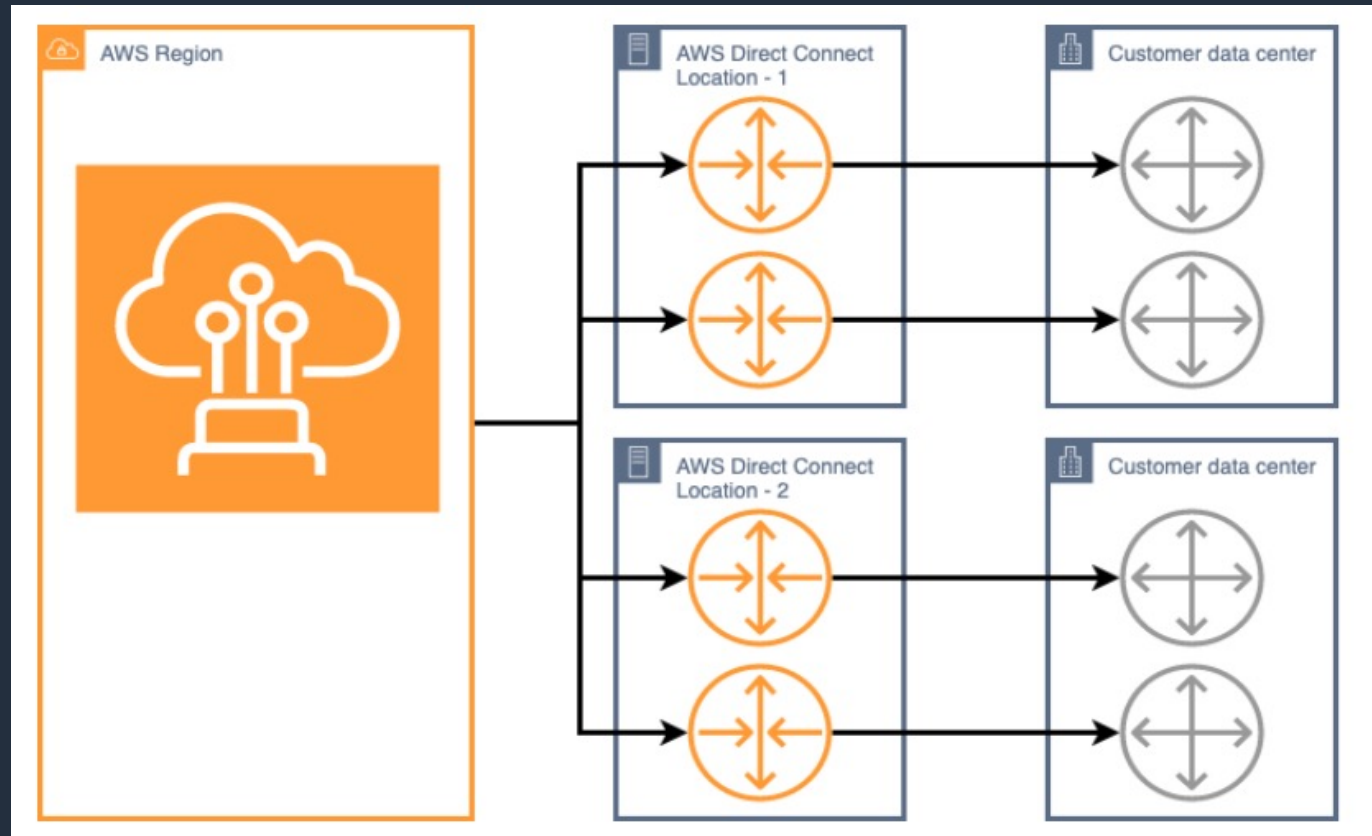
Quota name	Status	Last updated date
------------	--------	-------------------

Recently resolved service quota requests (0) < 1 >

Quota name	Status	Last updated date
------------	--------	-------------------



Provision Redundant Connectivity



Well-Architected Reliability Pillar



Workload architecture

REL 3

Service architecture

- ⌘ Business domains
- ⌘ Right-sized, decoupled services

REL 4

Prevent failures

- ⌘ Loosely coupled dependencies
- ⌘ Idempotent, consistent

REL 5

Mitigate failures

- ⌘ Graceful degradation
- ⌘ Throttle, fail fast, limit retries

Choose How to Segment Your Workload



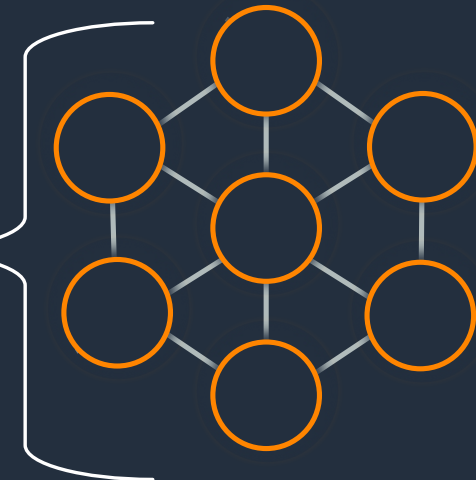
Monolithic application

- Does everything
- Shared release pipeline
- Rigid scaling
- High impact of change
- Hard to adopt new technologies



Service oriented

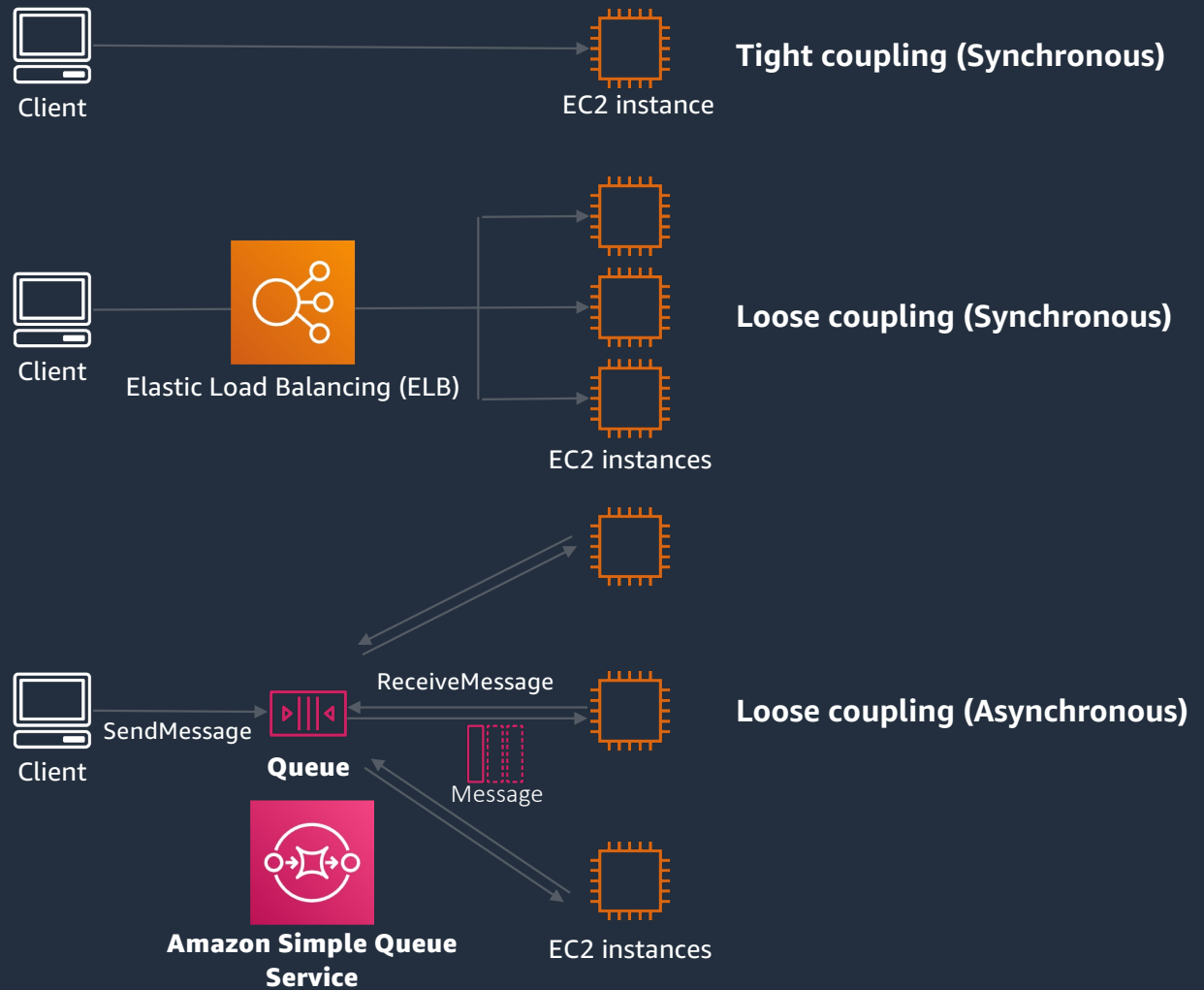
- Does some things
- Services surfaced via comms protocol
- Some coupled services



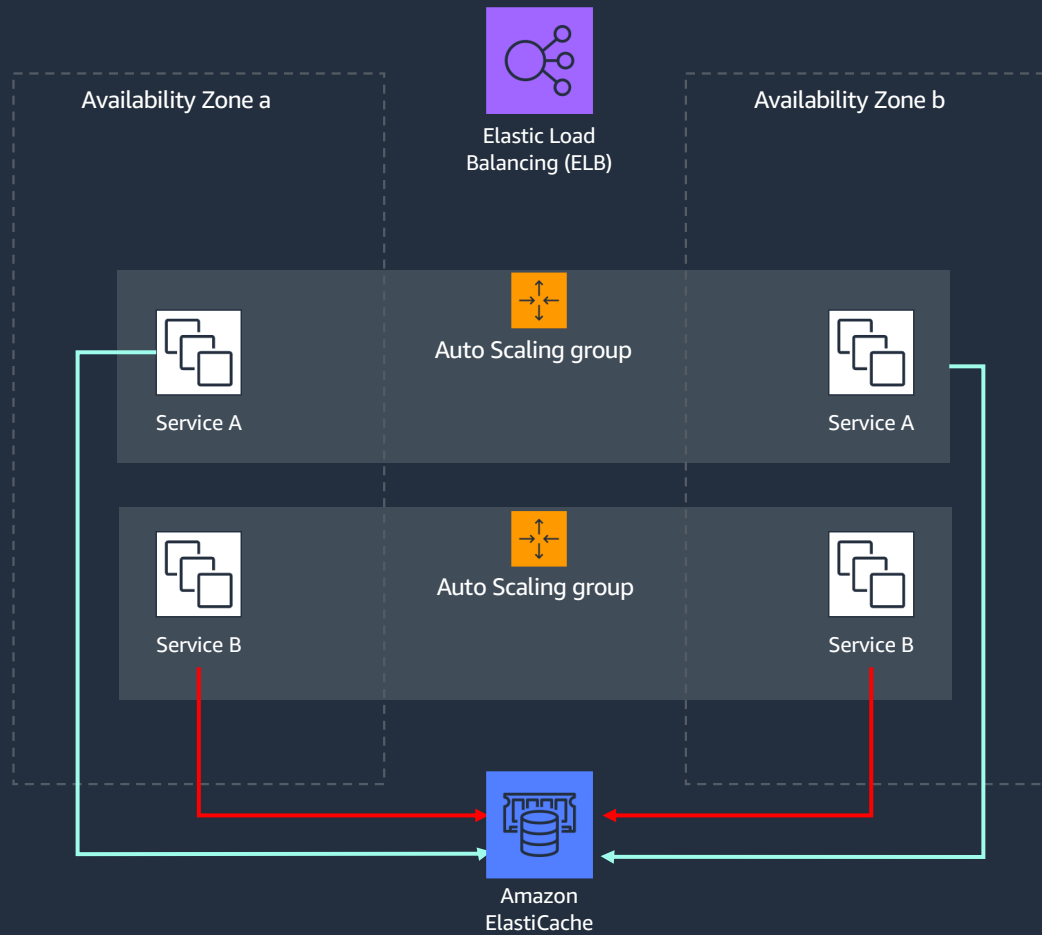
Microservices

- Does one thing
- Independent deployments
- Independent scaling
- Small impact of change
- Choice of technology

Implement Loosely Coupled Dependencies



Make Services Stateless Where Possible



Well-Architected Reliability Pillar



REL 6

Monitor workload resources

- ⌘ You can't manage what you can't see
- ⌘ Implement observability

REL 7

Adapt to changes in demand

- ⌘ Scale up and down as needed
- ⌘ Use automation

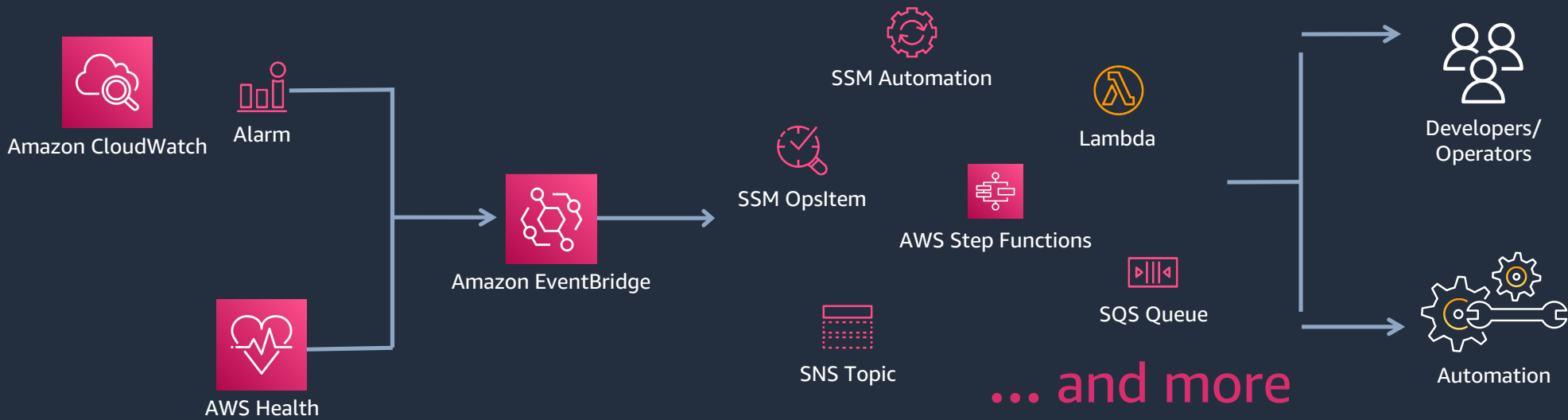
REL 8

Implement change

- ⌘ Deployments can be dangerous
- ⌘ Manage and reduce negative impacts

Real-time Processing and Alarming

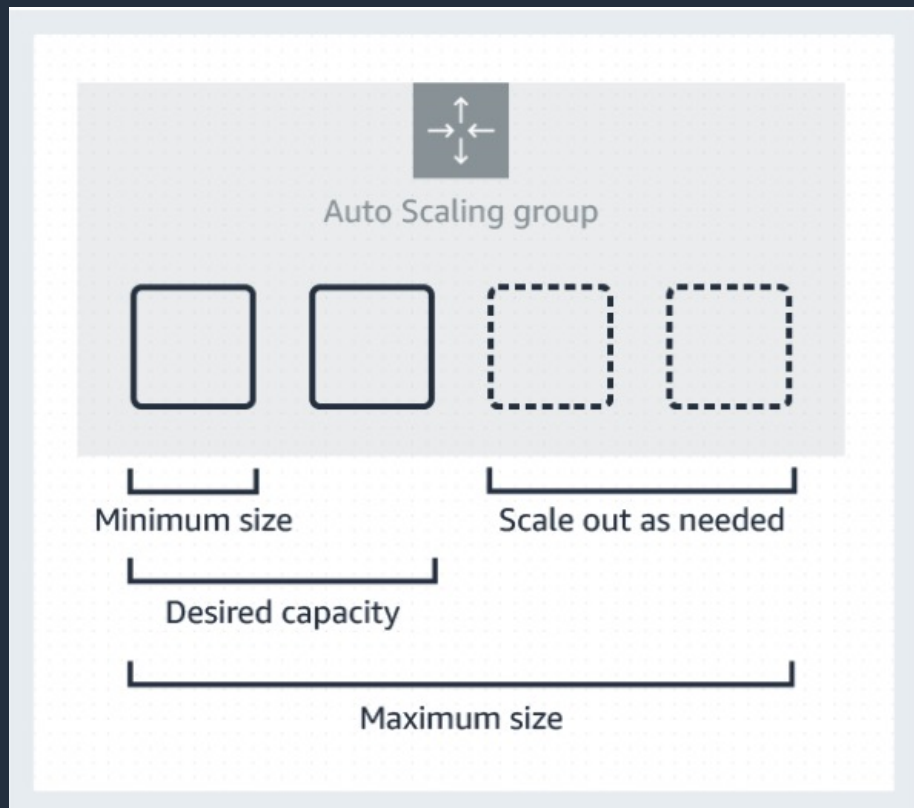
- Send notifications
- Automate responses



... and more

```
{
  "source": ["aws.health"],
  "detail-type": ["AWS Health Event"],
  "detail": {
    "service": ["S3"],
    "eventTypeCategory": ["issue"],
    "eventTypeCode": ["AWS_S3_INCREASED_GET_API_ERROR_RATES", "AWS_S3_INCREASED_PUT_API_ERROR_RATES"]
  }
}
```

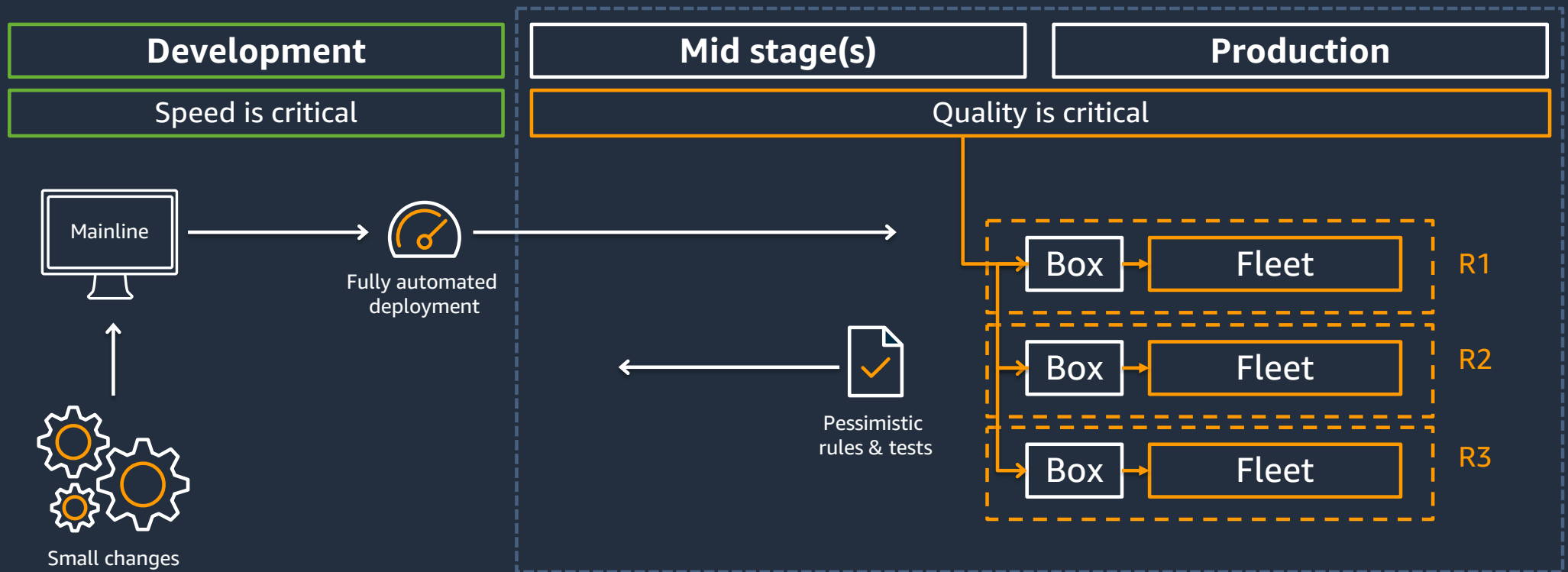
Use Automation When Obtaining or Scaling Resources



- EC2 instances
- Aurora read-replicas
- Aurora storage
- DynamoDB throughput
- ECS tasks
- Lambda concurrent instances

Deploy Changes with Automation

Also...Integrate testing as part of your deployment



Well-Architected Reliability Pillar



Failure management

REL 9

Back up data

- ⌘ Identify data to back up
- ⌘ Automate and secure back up

REL 10

Fault isolation

- ⌘ Use multiple sites
- ⌘ AWS Region, Availability Zone, etc

REL 11

Withstand component failures

- ⌘ Automate recovery
- ⌘ Notify when failures occur

Deploy the Workload to Multiple Locations

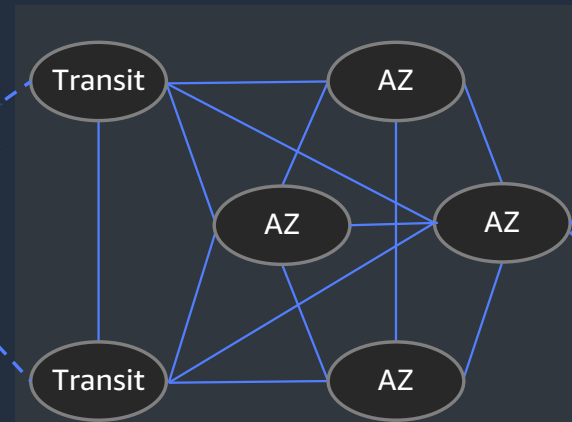
AWS REGIONS AND AVAILABILITY ZONES (AZS)

31 AWS Regions worldwide



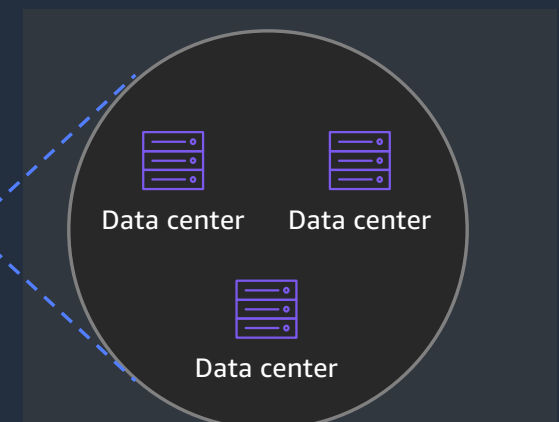
- AWS Regions
- Announced Regions

Each AWS Region has multiple AZs



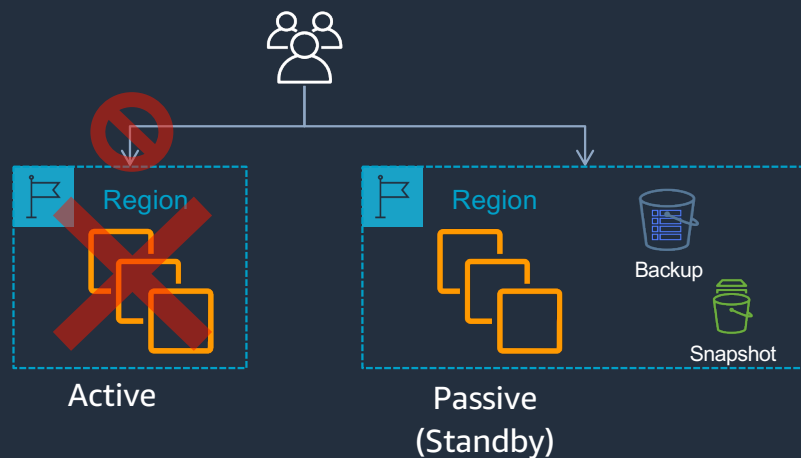
A Region is a physical location in the world

Each AZ includes one or more discrete data centers



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

Rely on the Data Plane and Not the Control Plane During Recovery



If control plane fails....

What will work	What may not work
DNS resolution and health checks	Updates to routing policies



Amazon Route 53



Route 53 Application recovery controller

Changes to routing controls	CRUDL routing controls
-----------------------------	------------------------

Well-Architected Reliability Pillar



Failure management

REL 12

Test reliability

- ⌘ Simulate failure modes
- ⌘ Test scaling, performance, processes

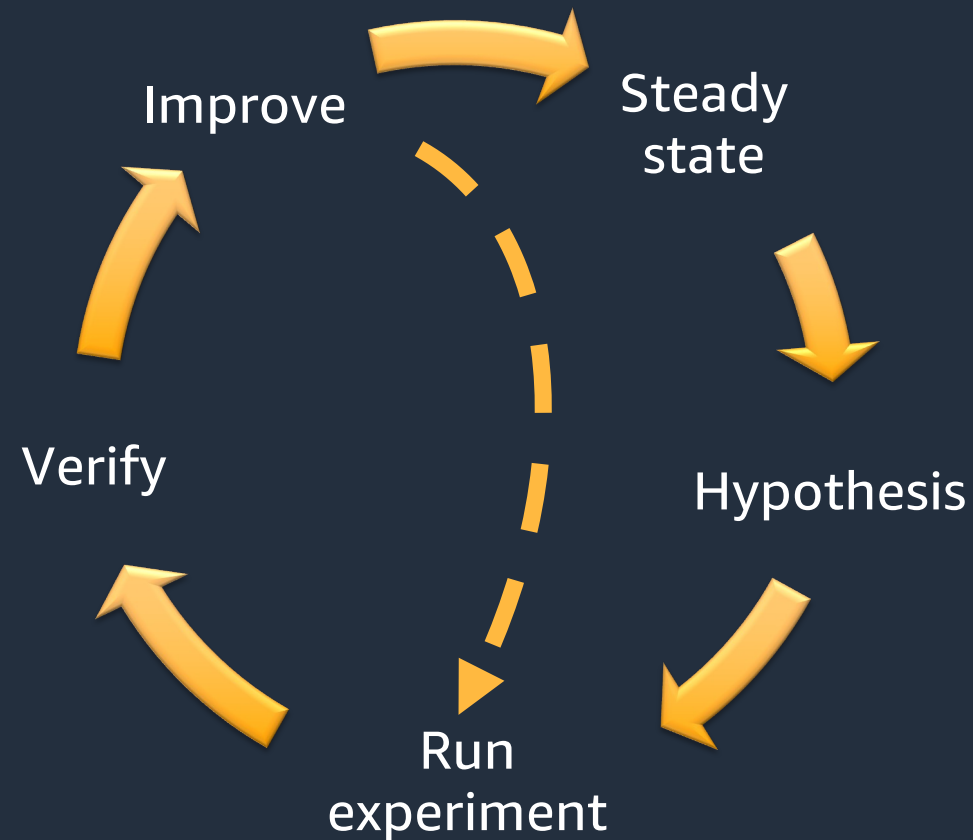
REL 13

Disaster recovery

- ⌘ Set objectives
- ⌘ Implement and test DR strategy

Test Resilience Using Chaos Engineering

A SCIENTIFIC METHOD



Conduct Game Days Regularly

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



People

Cross-discipline
team
Processes



Briefing

Overview
Roles



Planning

Preparation
Hypothesis



Execution

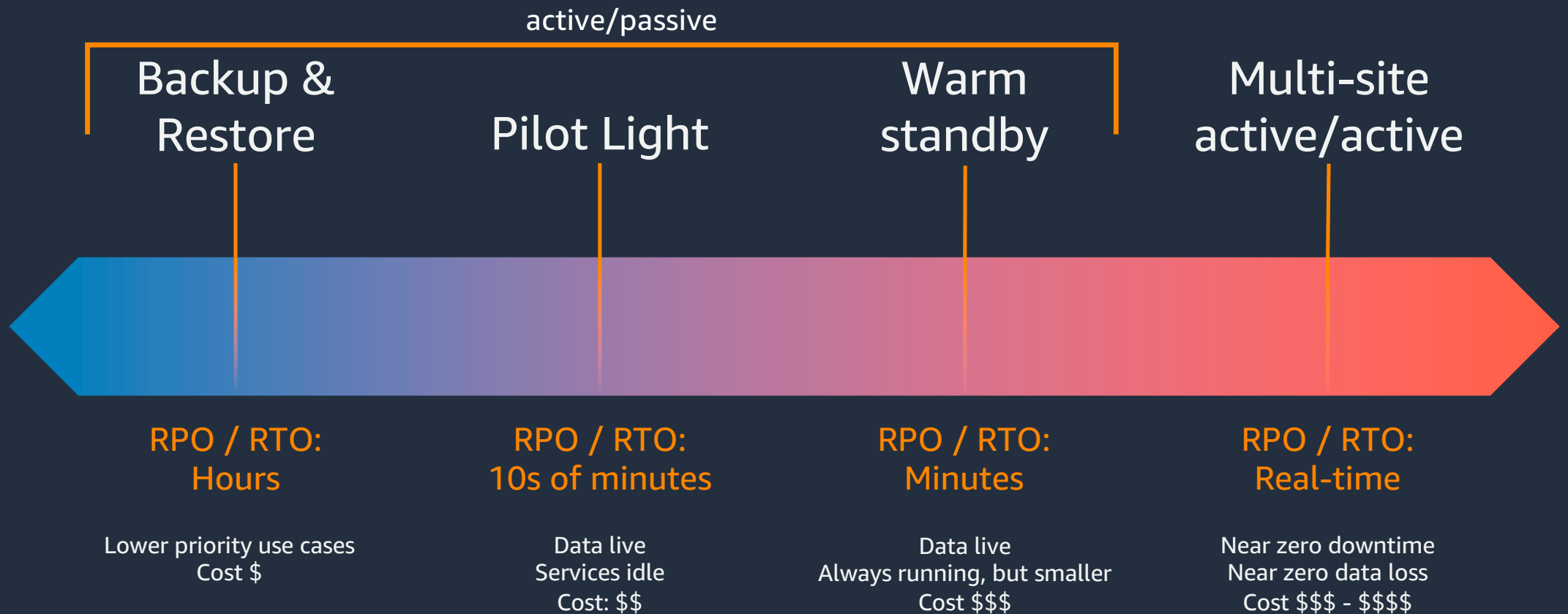
Run experiment



Analysis

Verify
Improve

Strategies for Disaster Recovery



Key Takeaways

- ⌘ Reliability is a continuous improvement process
- ⌘ Your system is not reliable and resilient unless you test that
- ⌘ Prepare for failure – evaluate risks
- ⌘ Call to action: get started with the best practices mentioned!!!

Resources

Whitepaper: Reliability Pillar: AWS Well-Architected Framework

bit.ly/reliability-pillar

Well-Architected hands-on labs

wellarchitectedlabs.com/reliability

Well-Architected Tool

aws.com/well-architected-tool/

AWS Resilience Hub

aws.amazon.com/resilience-hub/

go.aws/3FF5UA0 (Blog post)



Q & A

Danny Wright

Principal Technical Account Manager

AWS Enterprise Support

dwriamz@amazon.com

Please
complete the
session survey.



Well-Architected Reliability: Best
Practices You Can Use Today