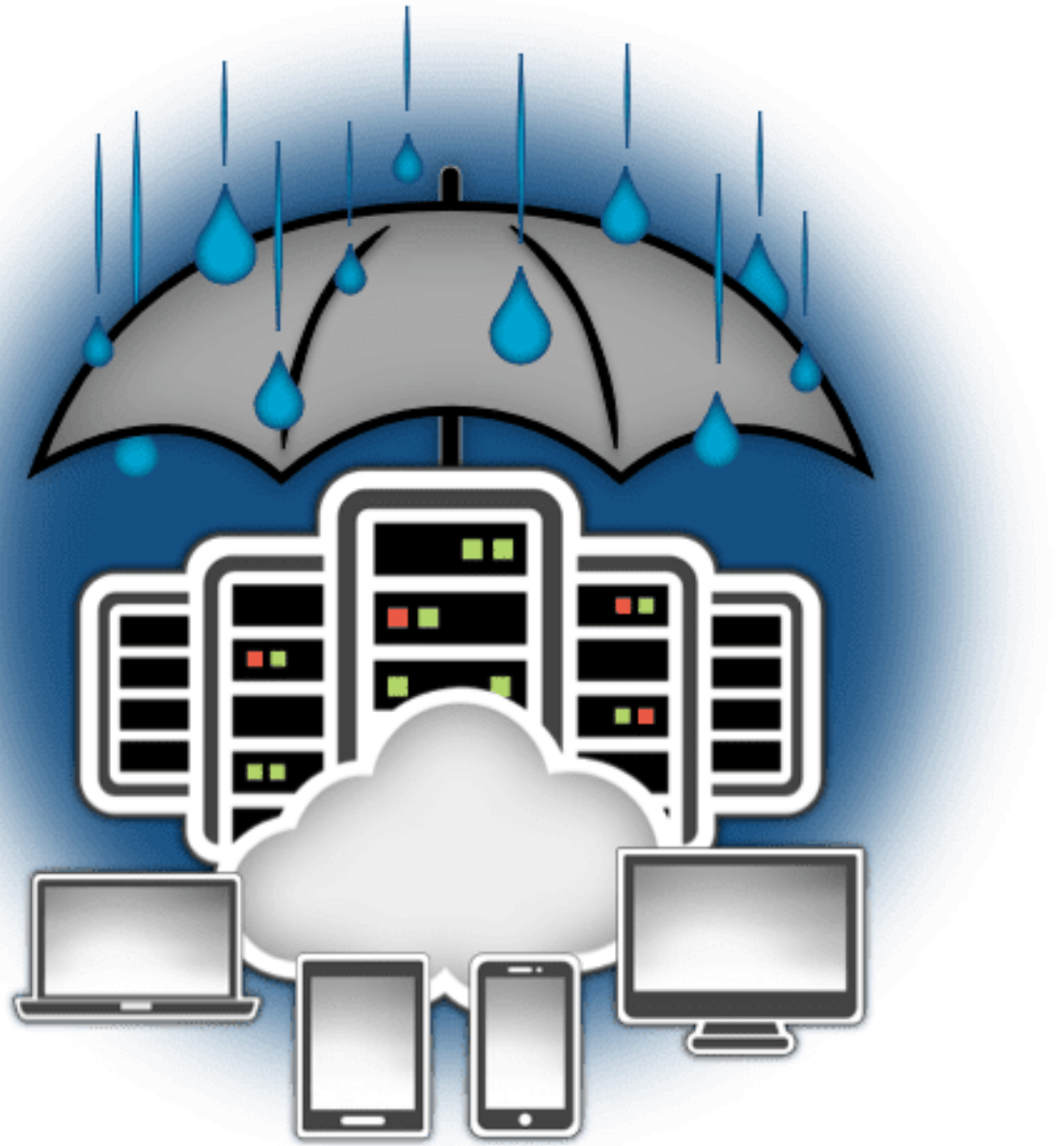
Designing Resilient, Scalable, Fault Tolerant Architectures



Resilience

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

High Availability

Resistance to common failures through design and operational mechanisms at a primary site

Core services, design goals to meet availability goals

Disaster Recovery

Returning to normal operation within specific targets at a recovery site for failures that cannot be handled by HA

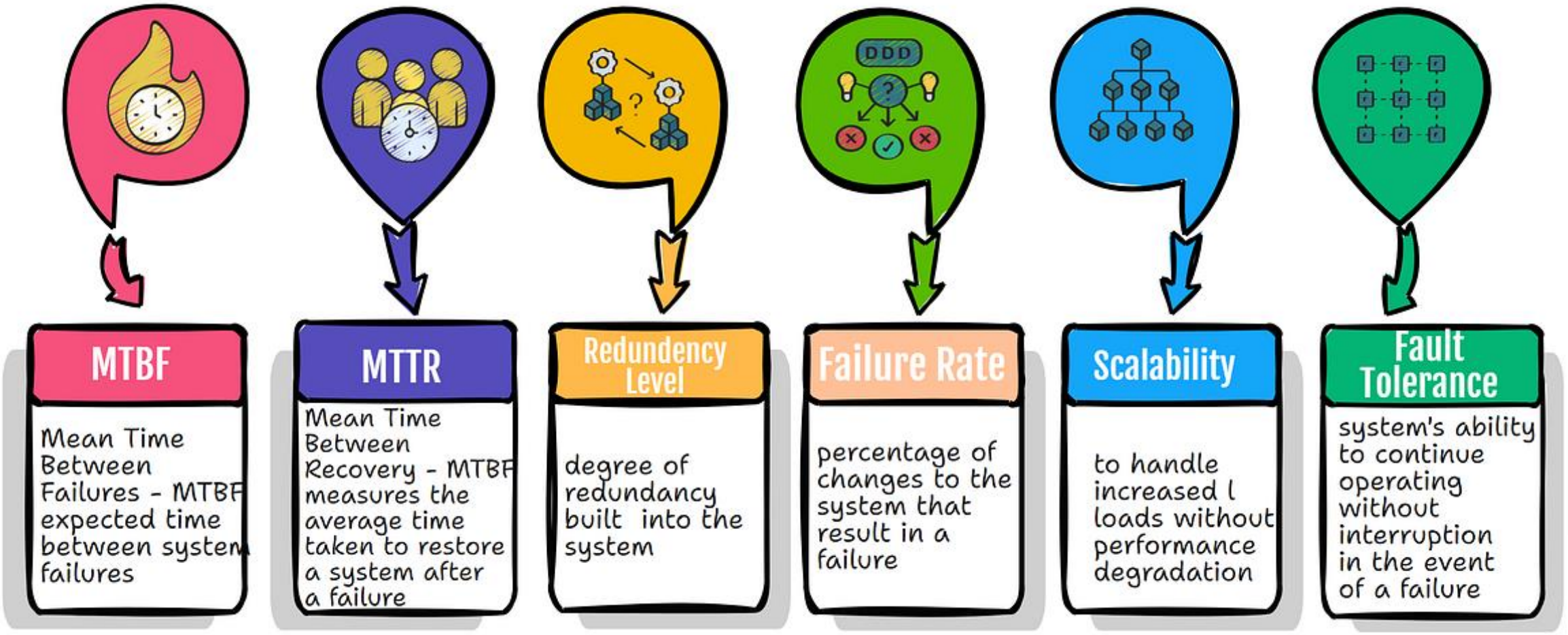
Backup & Recovery, Data Bunkering, Managed recovery objectives

Impacts of Resilient Software Systems

- **Business Impact** - tarnishing the brand's image, lose customer trust
- **Financial Impact** - lost transactions, lost customers, remediation costs, data loss
- **User Experience** – inconsistent and poor user experience, latency issues, unpredictable application response

Resilient System KPIs

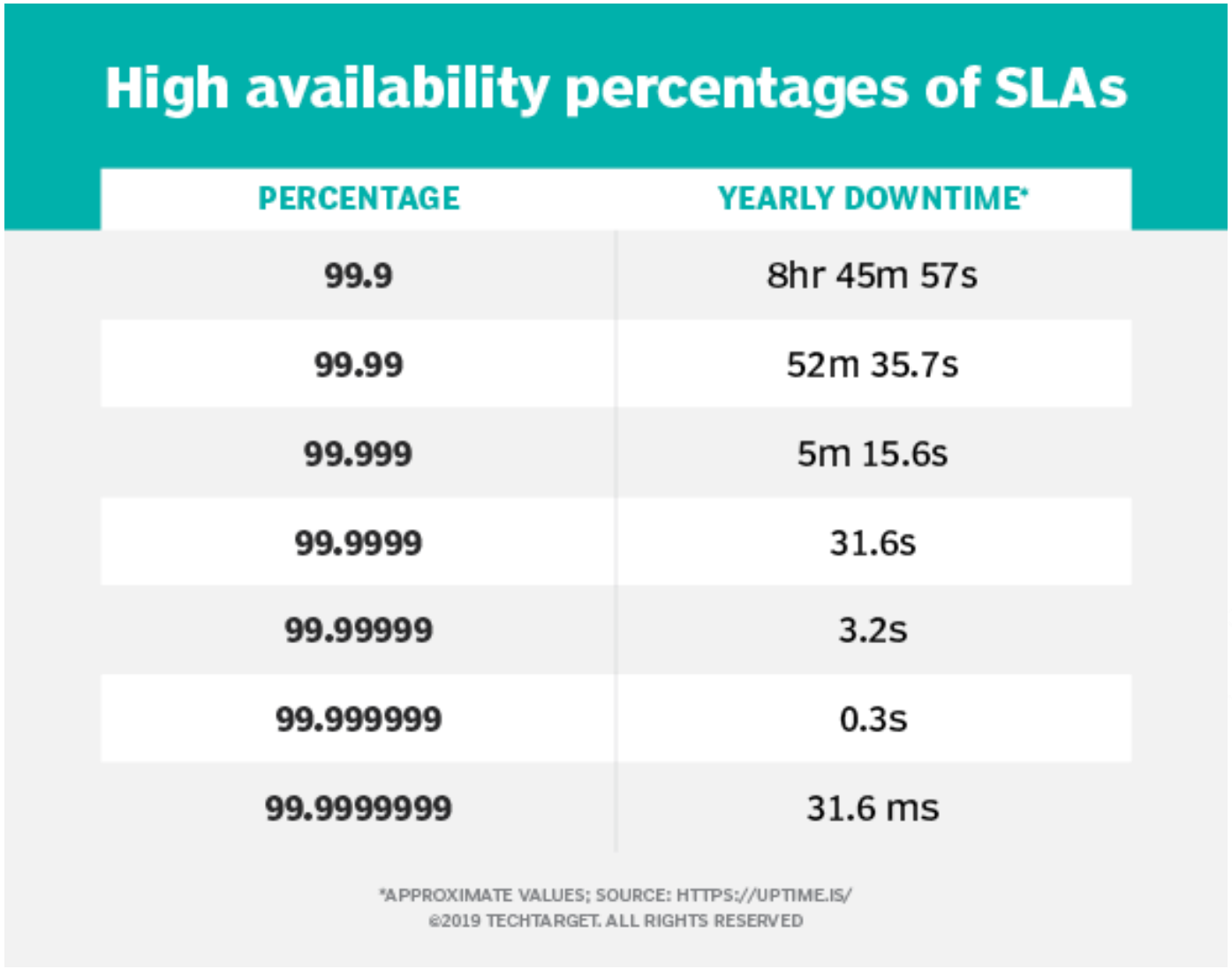
BeSA
Become a Solutions Architect



Resilience and related Software Quality Attributes

- **Security** - security measures protect against malicious attacks that could cause those failures. Example Rate limiting WAF, DDOS protection by Shield
- **Maintainability** – easy to update & repair and do not introduce additional risks Example – Systems Manager Automation, Patch Manager
- **Scalability** – ability to allocate more resources to handle increased demand without impacting system resilience Example - ASG
- **Performance** – ability of the system to perform well under normal conditions and even when under stress. Finding a fine balance. Redundancy often impacts latency. Example – Elastic Load Balancer
- **Availability** – closely linked, design for redundancy and seamless failover processes, quickly recovering by ensuring that alternative resources are available during a disruption/failure. Example – RDS Multi-AZ deployment, Read Replicas

Availability Table

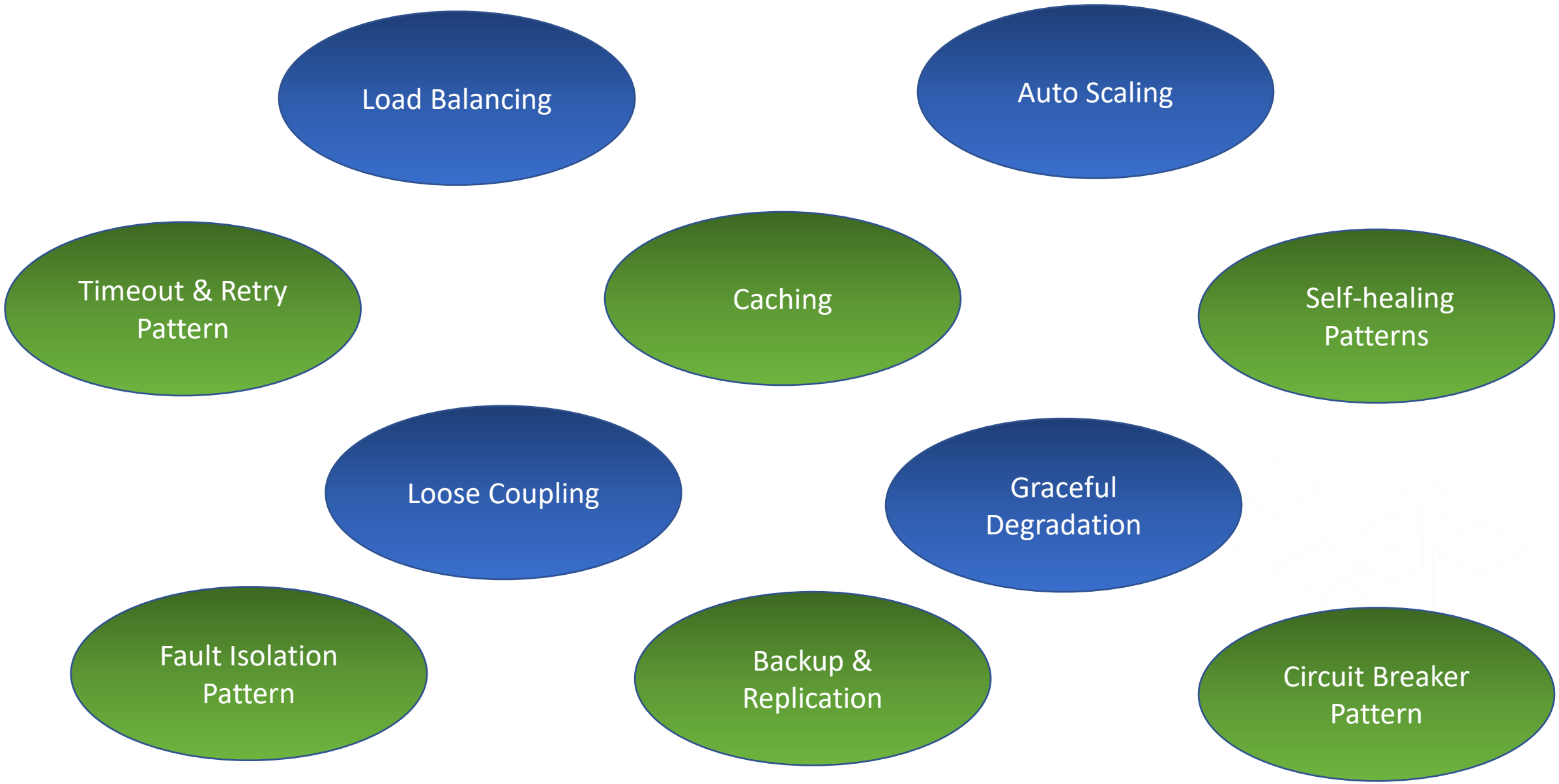


High availability percentages of SLAs

PERCENTAGE	YEARLY DOWNTIME*
99.9	8hr 45m 57s
99.99	52m 35.7s
99.999	5m 15.6s
99.9999	31.6s
99.99999	3.2s
99.999999	0.3s
99.9999999	31.6 ms

*APPROXIMATE VALUES; SOURCE: HTTPS://UPTIME.IS/

Techniques to build Resilience



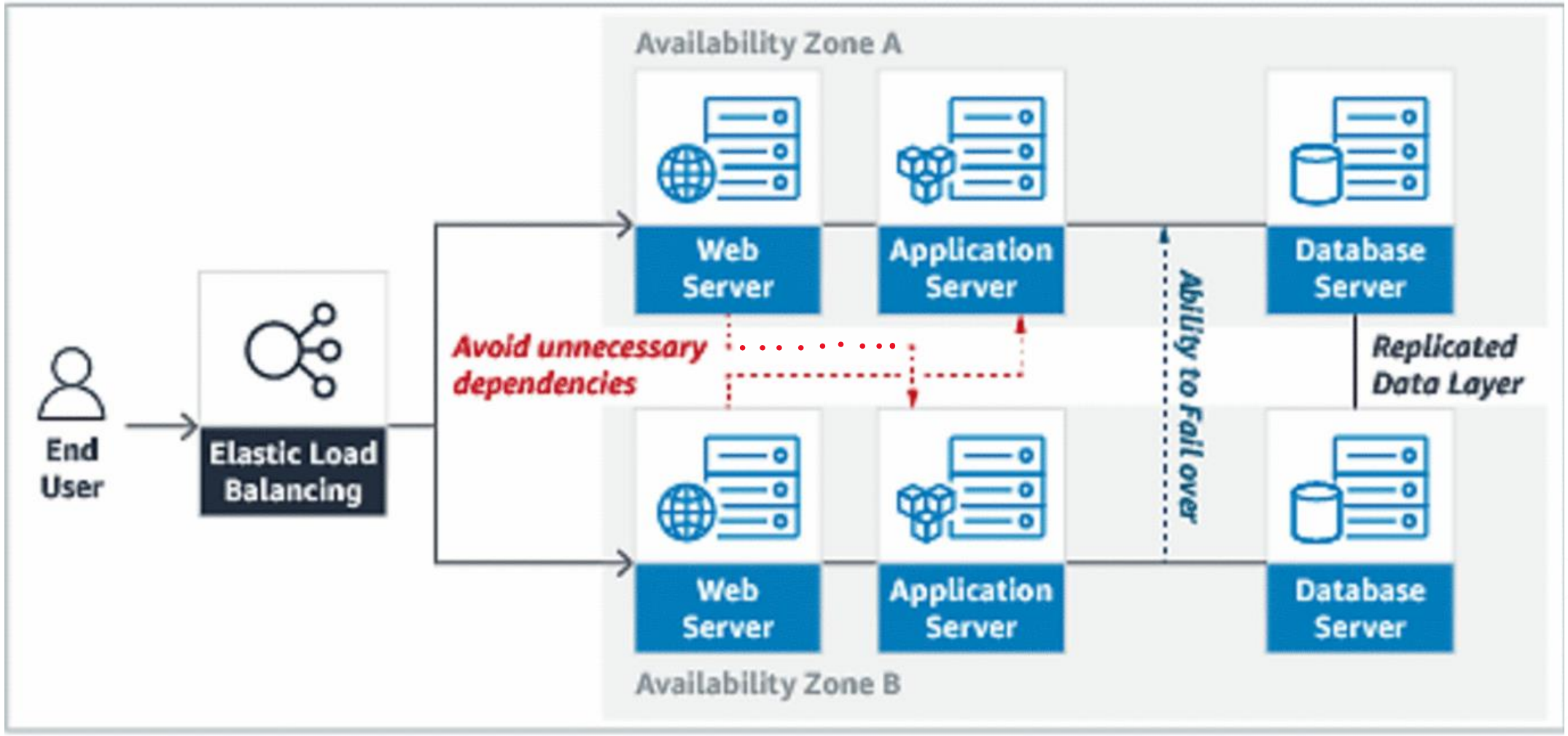
Resilient Software Design Patterns and Principles

- **Redundancy**: By avoiding single points of failure and introducing backup components, systems can continue to operate even when parts fail.
- **Load Balancing**: Distributing incoming network traffic across multiple servers ensures no single server is overwhelmed, enhancing both resilience and performance.
- **Failover Strategies**: These strategies allow systems to switch over automatically to a standby system or component upon failure.
- **Circuit Breaker Pattern**: halting operations temporarily, much like an electrical circuit breaker.
- **Bulkhead Pattern**: Isolating failures by dividing the application into isolated partitions, ensuring a failure in one does not impact the others.
- **Timeout and Retry Strategies**: These strategies define how systems should behave when awaiting responses, ensuring they don’t hang indefinitely.

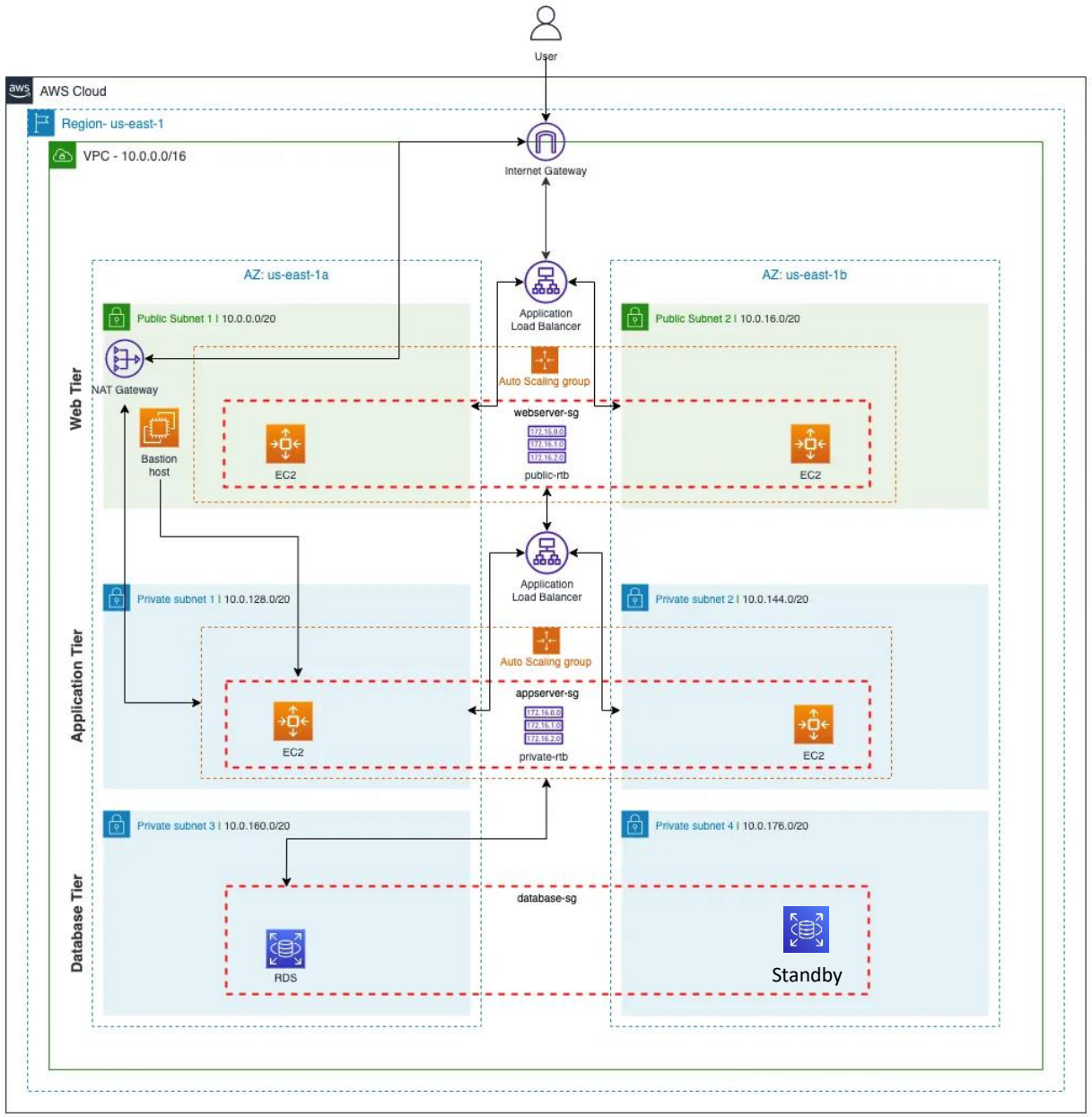
Resilient Design at every Layer - Tier

- **Infrastructure:** resilience through redundancy (having backup servers), failover mechanisms (automatically switching to a backup system during failures), and geographic distribution (spreading resources across various regions to mitigate localized issues).
- **Application:** resilience through effective error handling, graceful degradation (system continuity albeit at a reduced capacity, during failures), and persisting application state
- **Database:** resilience involves strategies like replication (maintaining copies of data), sharding (distributing data across multiple databases), and regular backups

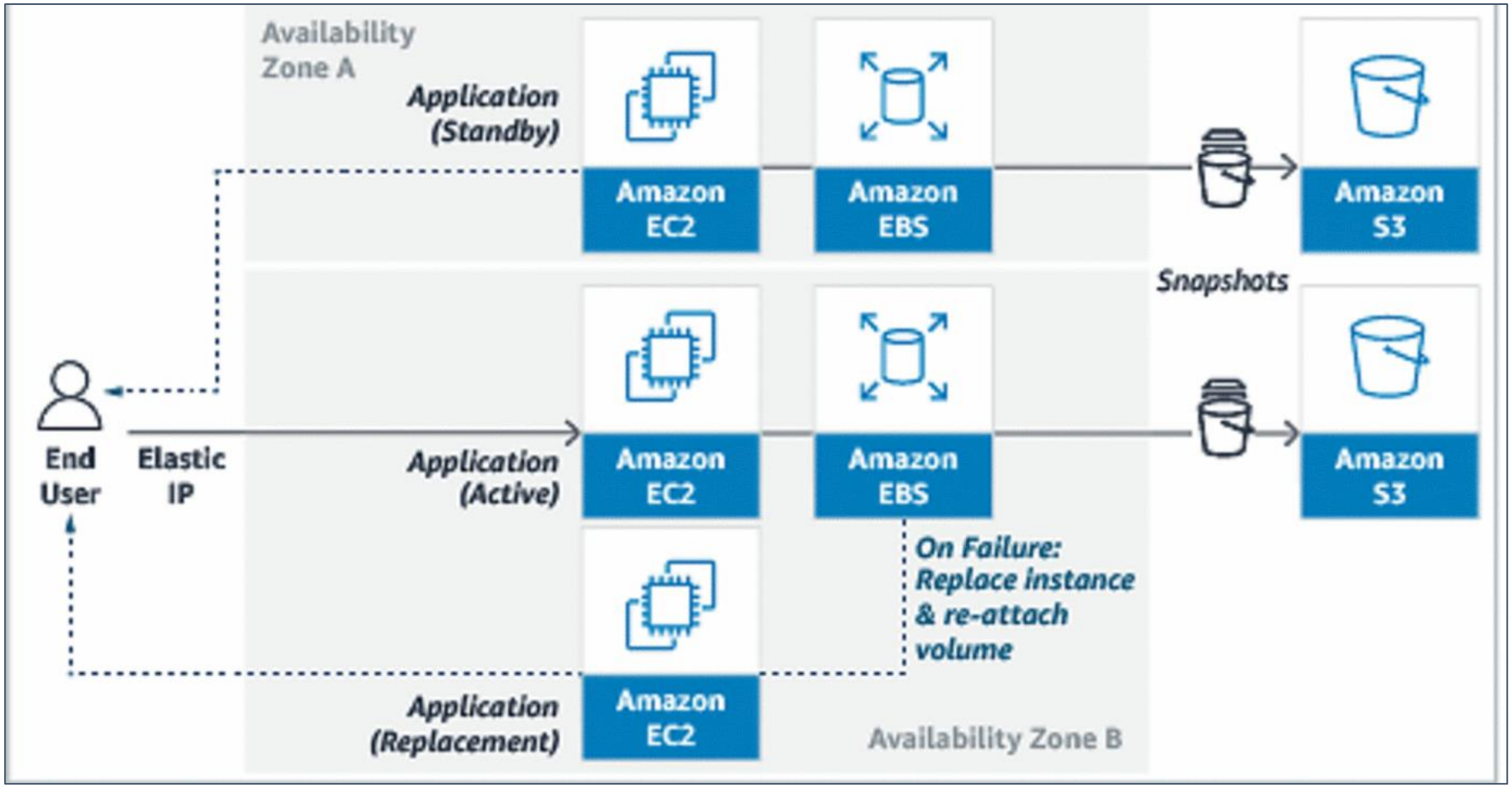
Resilient Architecture – Example 1



Resilient Architecture – Example 2



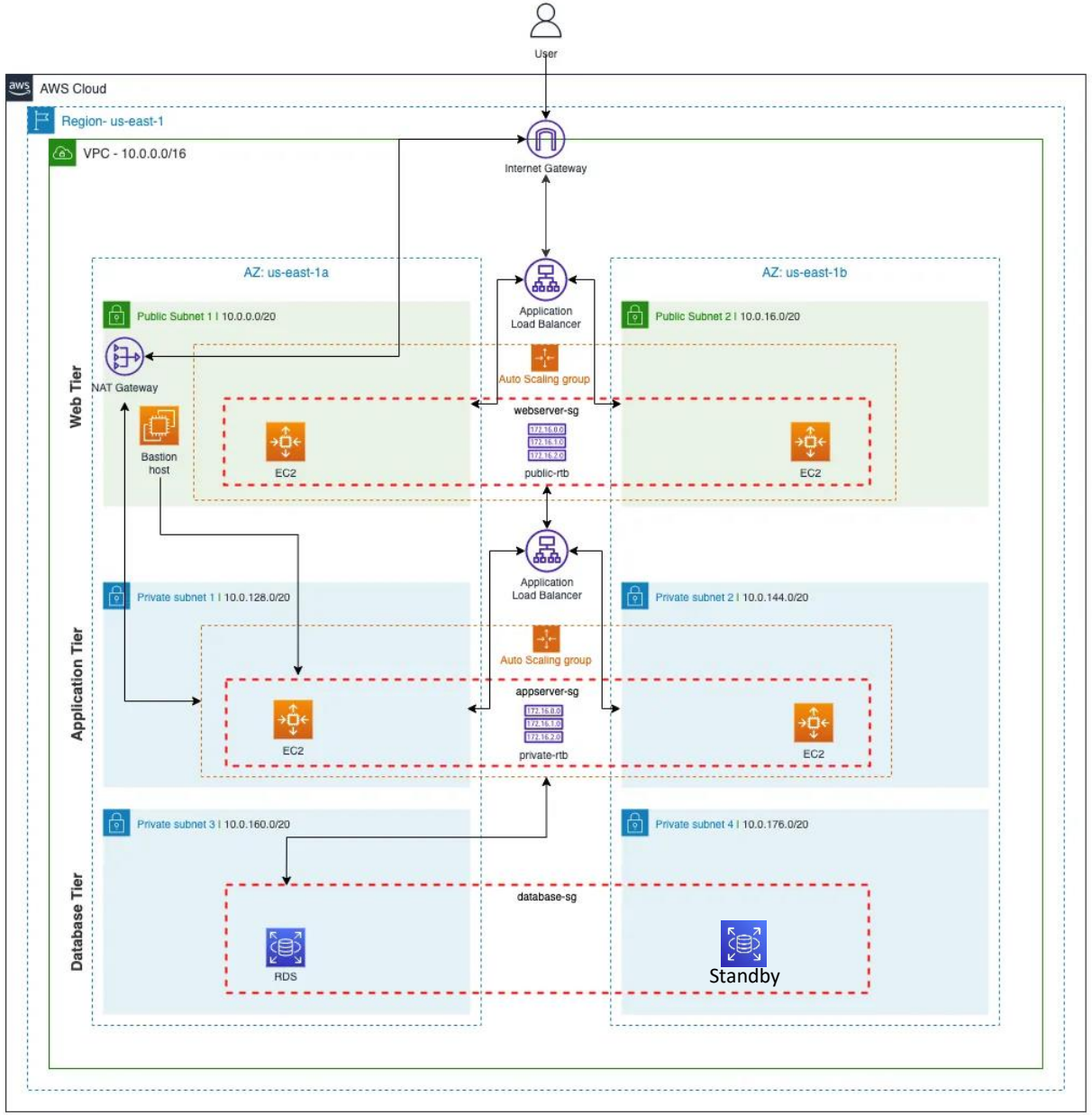
Resilient Architecture Example 3



Multi Zonal High Availability

MITIGATED

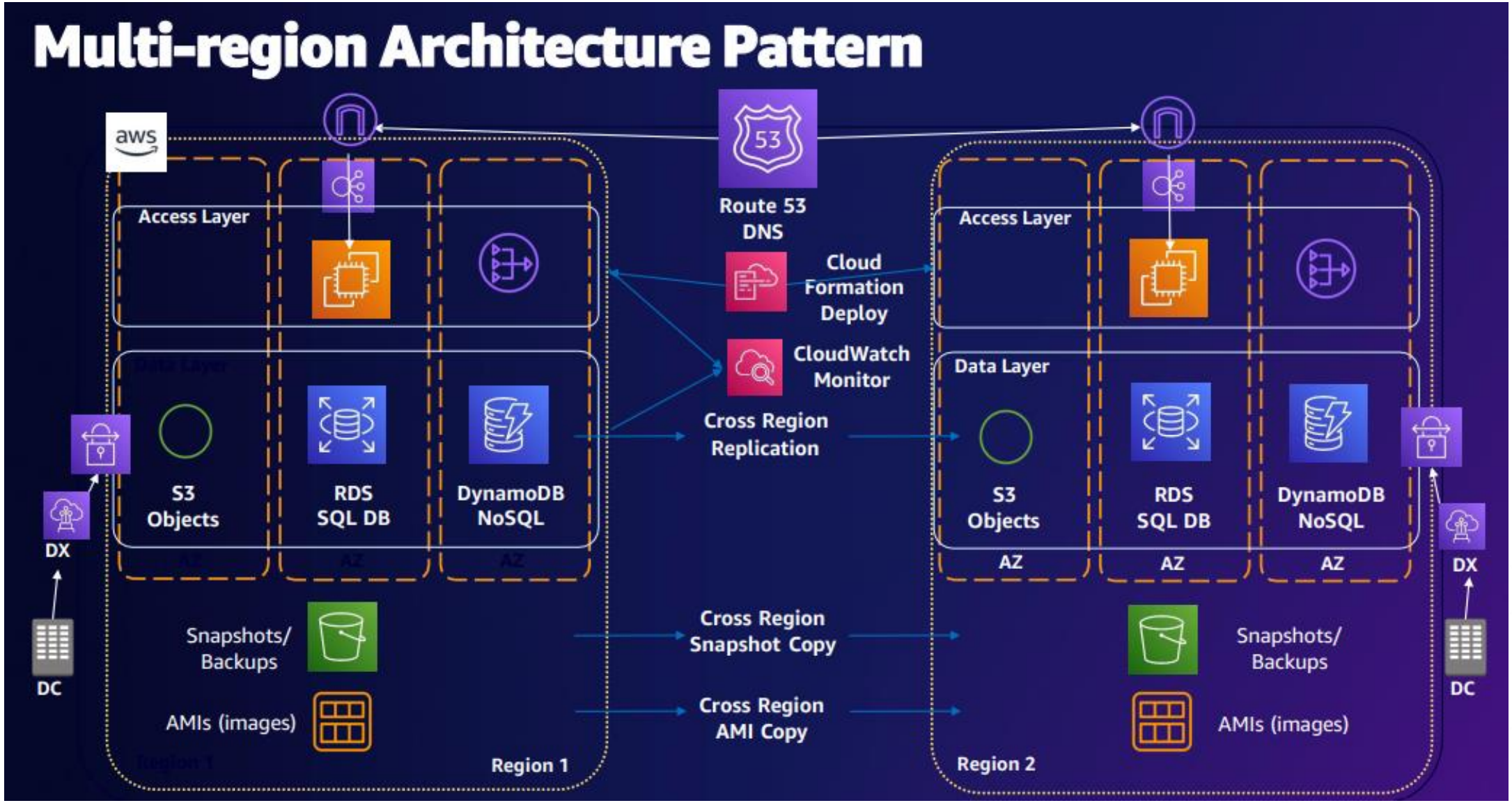
- Load Induced
- Component / Host failure
- network interruptions
- Entire Rack Failure
- Datacentre interruptions



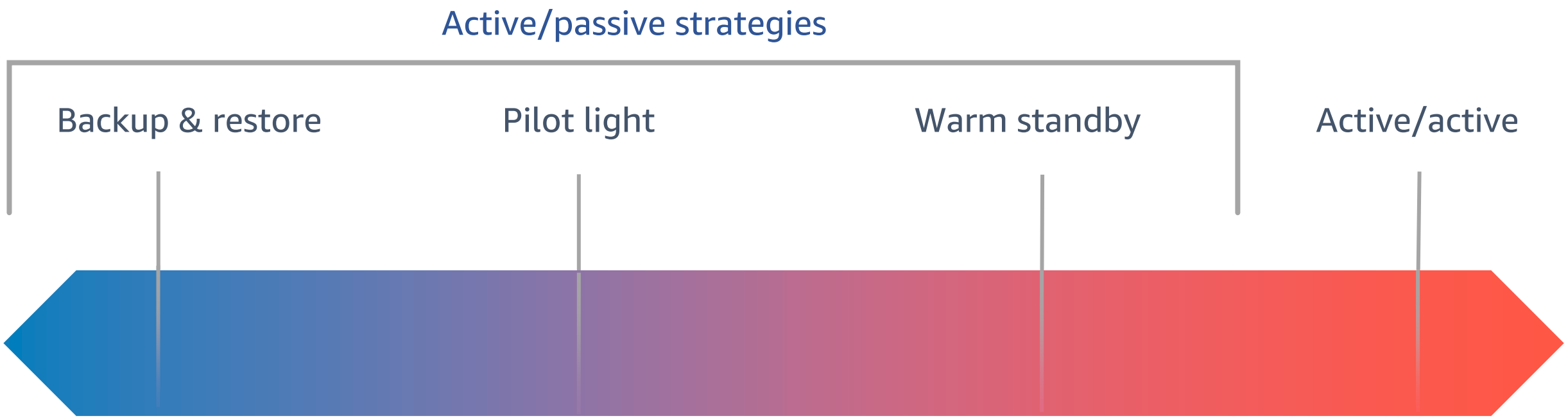
NOT MITIGATED

- Operator error / bad deployment
- Regional Failure / Natural Disaster
- All of Internet Failure
- All of provider disruption

Multi Region Resilient Architecture – Example 4



Strategies for disaster recovery



RPO / RTO: Hours

- Lower-priority use cases
- Provision all AWS resources after event
- Restore backups after event
- Cost $

RPO / RTO: 10s of minutes

- Data live
- Services idle
- Provision some AWS resources and scale after event
- Cost: $$

RPO / RTO: Minutes

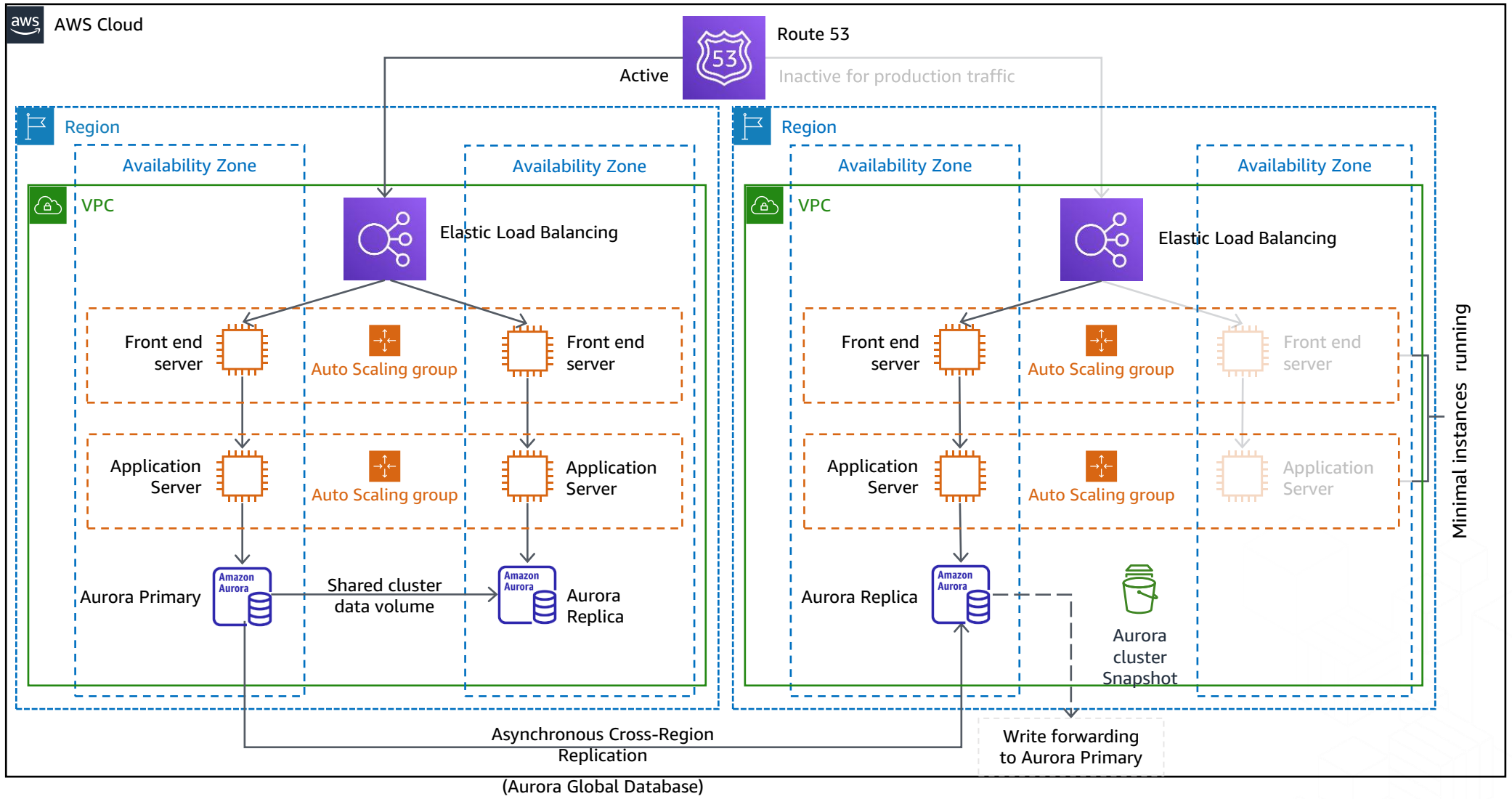
- Always running, but smaller
- Business critical
- Scale AWS resources after event
- Cost $$$

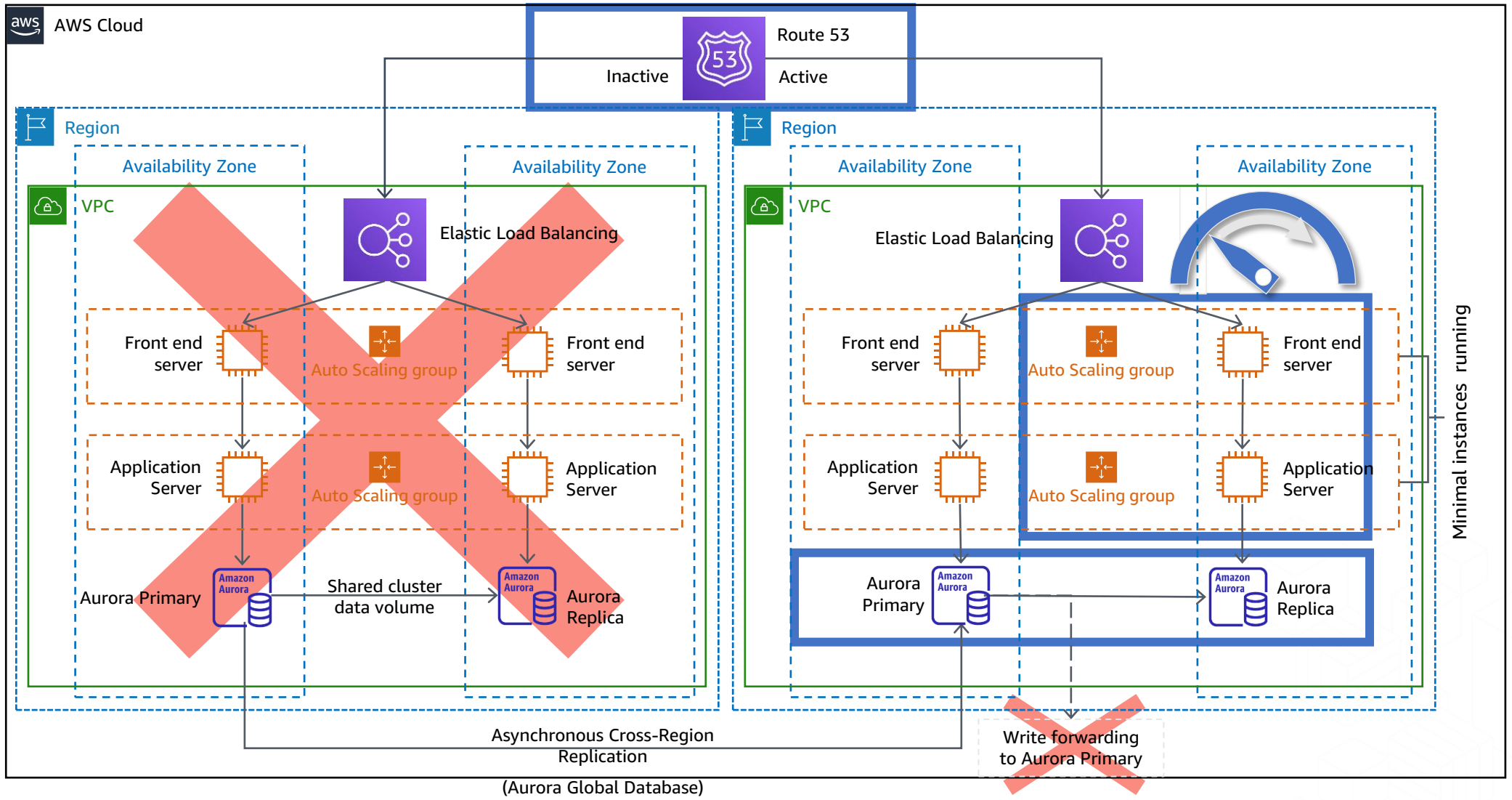
RPO / RTO: Real-time

- Zero downtime
- Near-zero data loss
- Mission Critical Services
- Cost $$$$

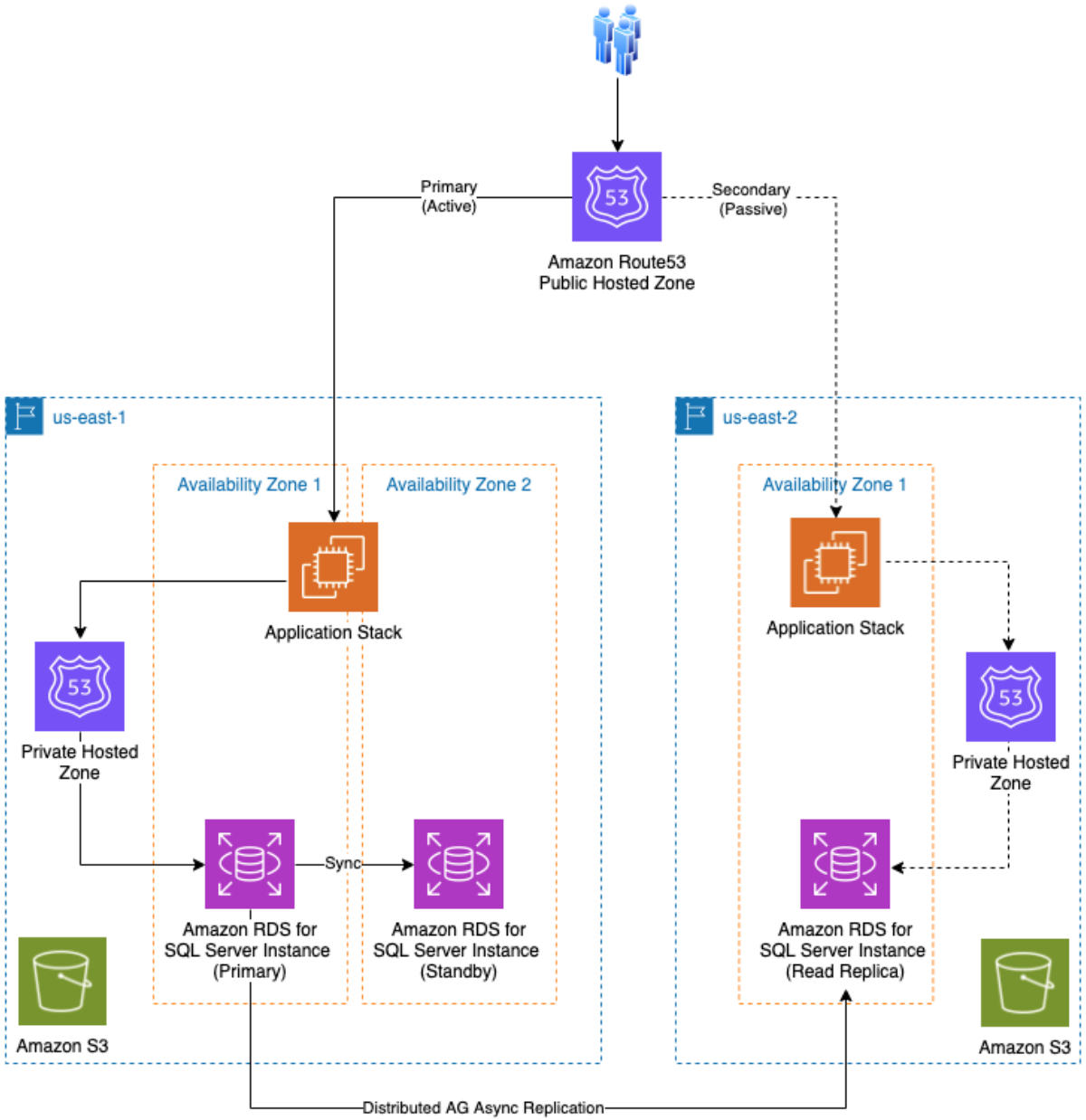
Disaster Recovery SLAs

Multi Region

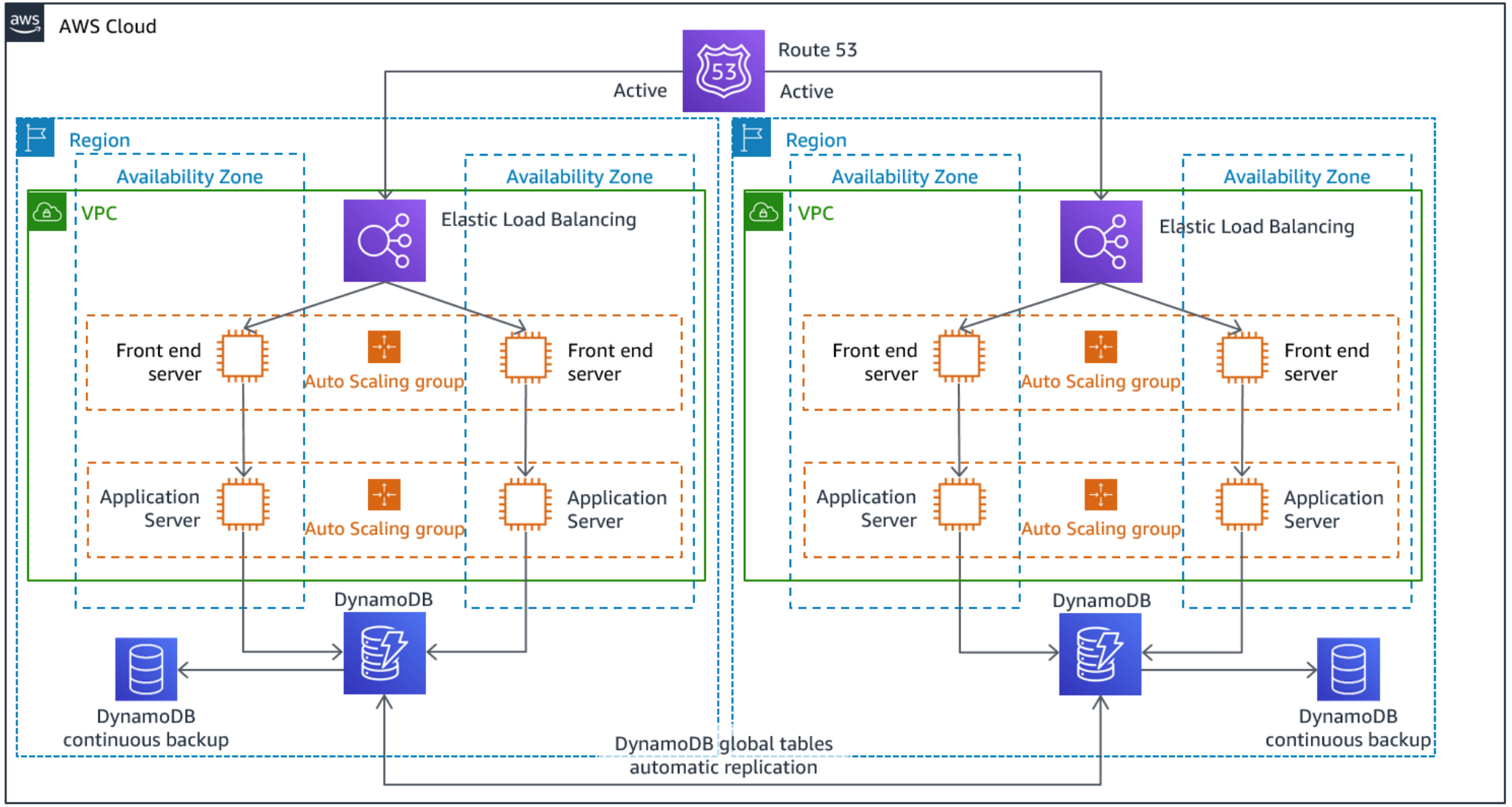



AWS Cloud
Route 53
Inactive
Active
Region
Region
Availability Zone
Availability Zone
Availability Zone
Availability Zone
VPC
VPC
Elastic Load Balancing
Elastic Load Balancing
Front end server
Front end server
Front end server
Auto Scaling group
Auto Scaling group
Application Server
Application Server
Application Server
Auto Scaling group
Auto Scaling group
Aurora Primary
Shared cluster data volume
Aurora Replica
Aurora Primary
Aurora Replica
Minimal instances running
Asynchronous Cross-Region Replication
(Aurora Global Database)
Write forwarding to Aurora Primary

RDS - Cross Region Replication

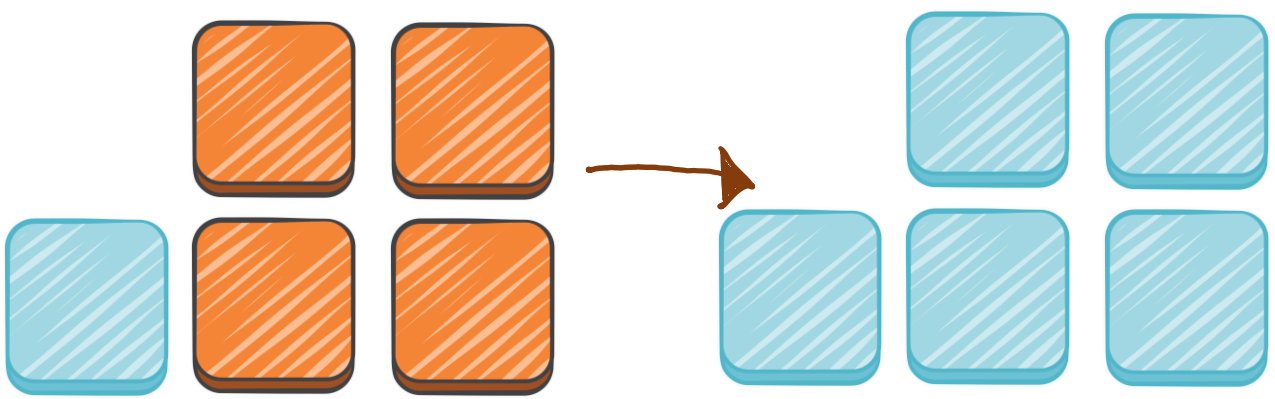


DynamoDB – Active - Active with Global Tables

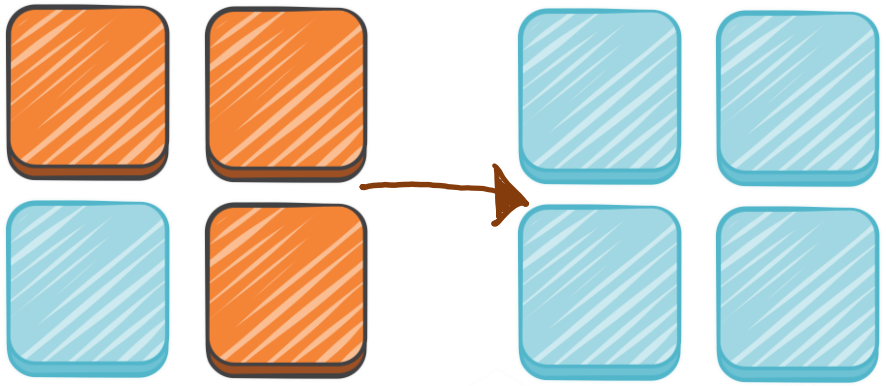


Fractional deployments minimize impact

Blue/Green or Canary deployments

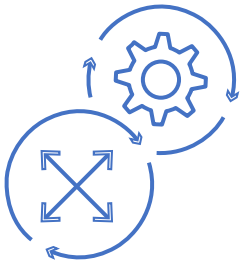


All at once

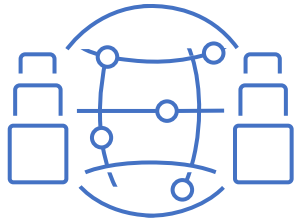


Rolling

AWS Well Architected Pillars for Resilience



Operational Excellence



Reliability

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.

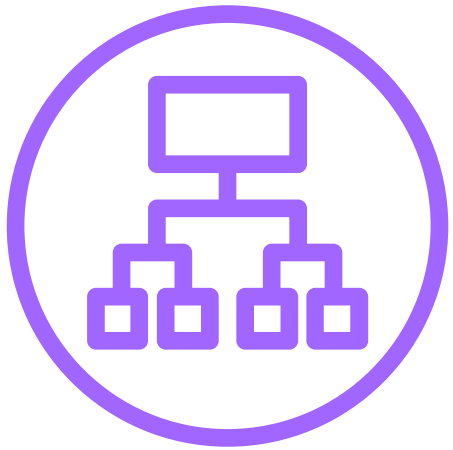
Question does not apply to this workload Info

Select from the following

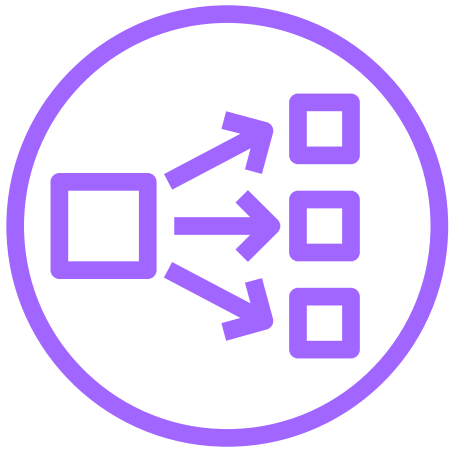
- [x] Monitor all components of the workload to detect failures Info
- [x] Fail over to healthy resources Info
- [x] Automate healing on all layers Info
- [x] Use static stability to prevent bimodal behavior Info
- [x] Send notifications when events impact availability Info

The Elastic Load Balancing family

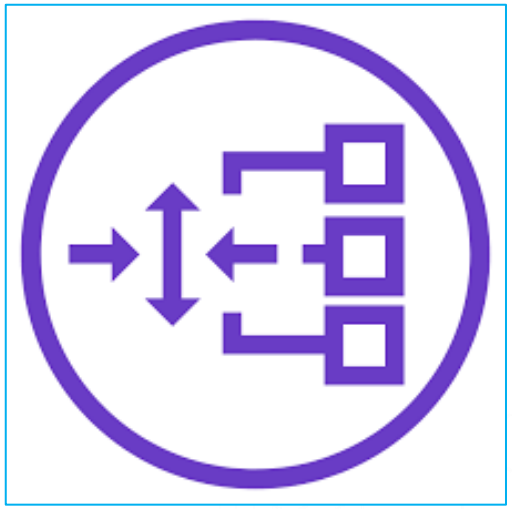
Application



Network



Gateway

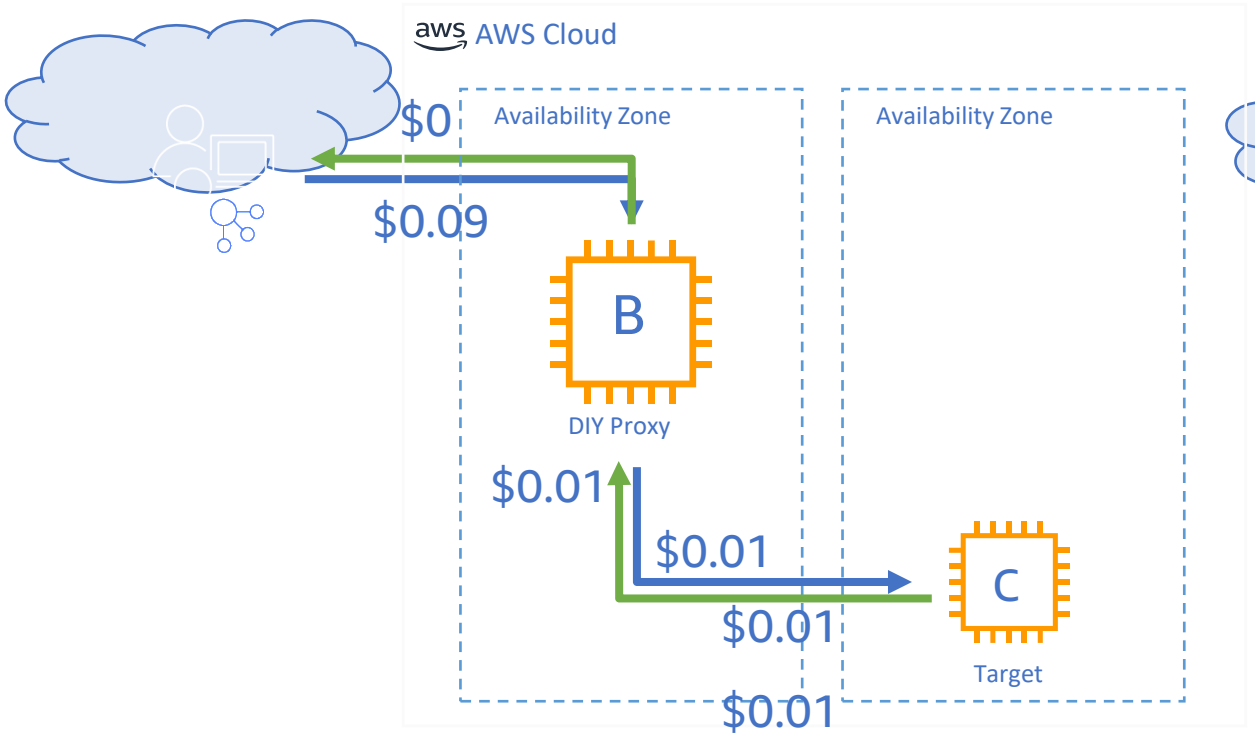


ALB vs NLB

Criteria	ALB	NLB
Security	Security Groups, TLS offload	Security Groups, TLS offload
Targets	EC2, ECS/EKS, IP, Lambda, ASG	EC2, ECS/EKS, IP, ASG
Scaling	100Mbps->100 Gbps, 2x every 5 mins	3Gbps->30 Gbps at 1Gbps per min per AZ
Latency	<10ms per request	<100us per connection
Performance per LB	Shard at 50 IPs Shard at 100 Gbps Shard at 1M new CPS/RPS/TLS Shard at 5M active Conns	Shard at 1M PPS per IP Shard at 30 Gbps per IP Shard at 50k new CPS per IP Shard at 1M active Conns per IP
Key AWS Integrations	VPC, ACM, IAM, DX, VPN, AGA, WAF, Cognito	VPC, ACM, IAM, DX, VPN, PrivateLink
Simplicity	Single DNS endpoint	Single DNS endpoint, Static IP
Cost	~$16/month (us-east-1) $0.008/LCU (us-east-1) No Cross Zone DTAZ	~$16/month (us-east-1) $0.006/LCU (us-east-1)

Cross Zone Data Transfer Costs with Public ALB

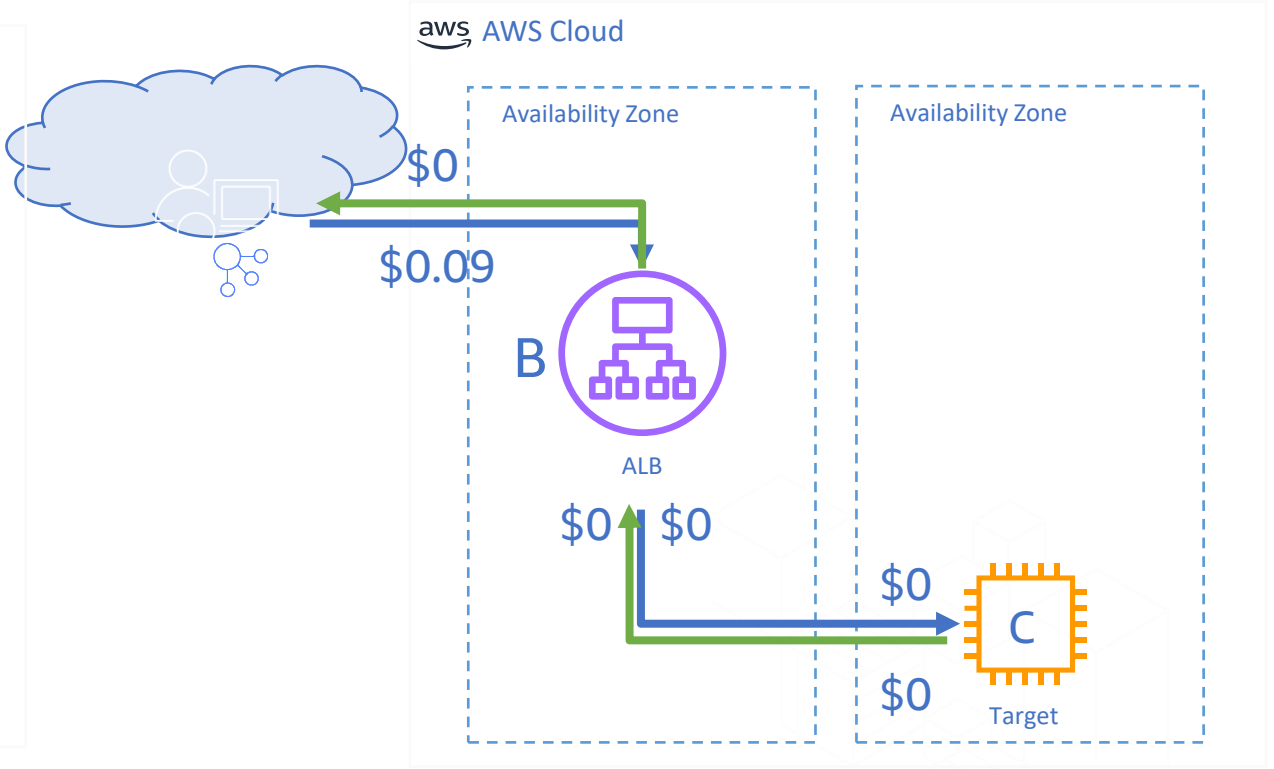
3rd Party DIY Proxy w/Cross Zone



Client IN (Request)
Target OUT (Response)

Cost per GB
A->B $0
B->C $0.02

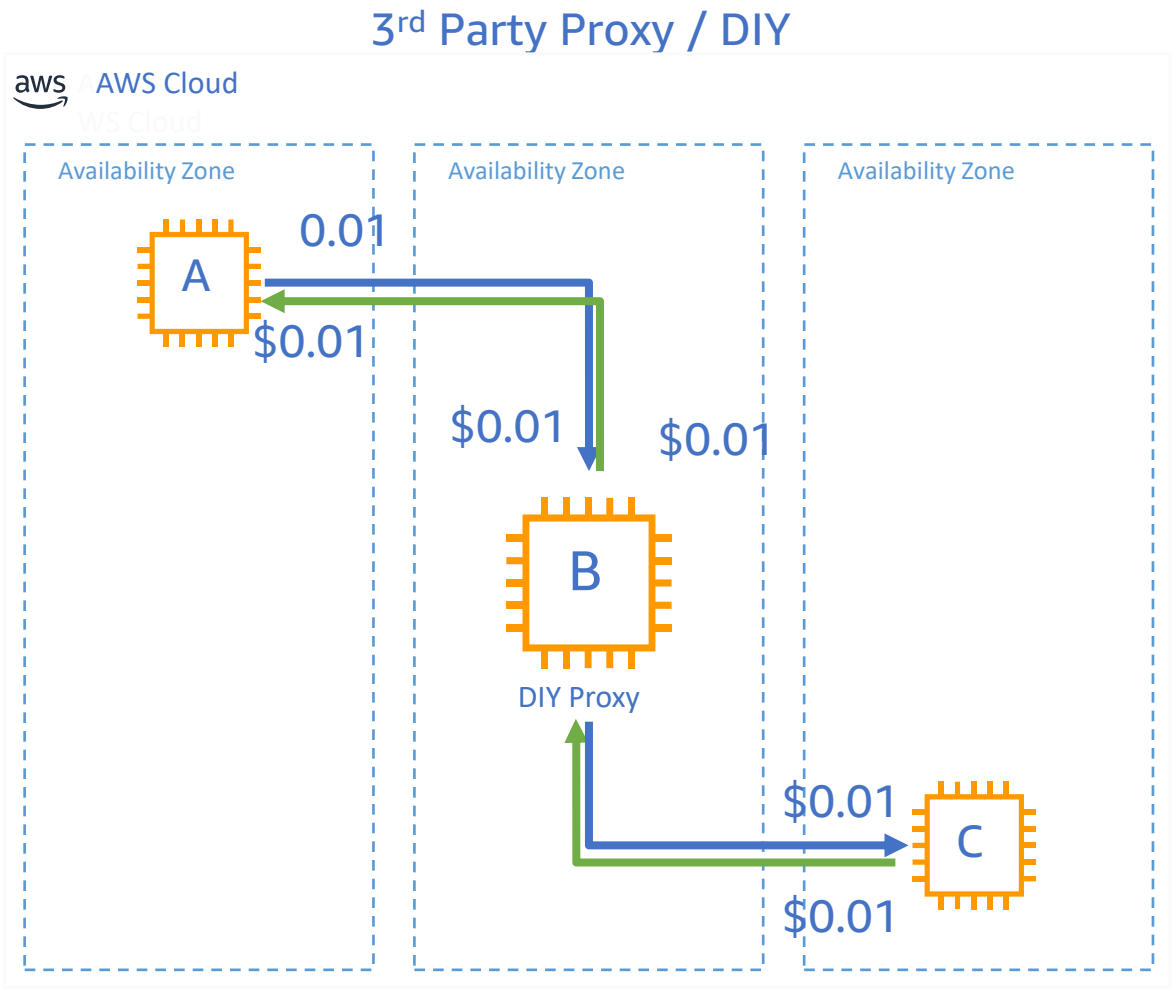
ALB

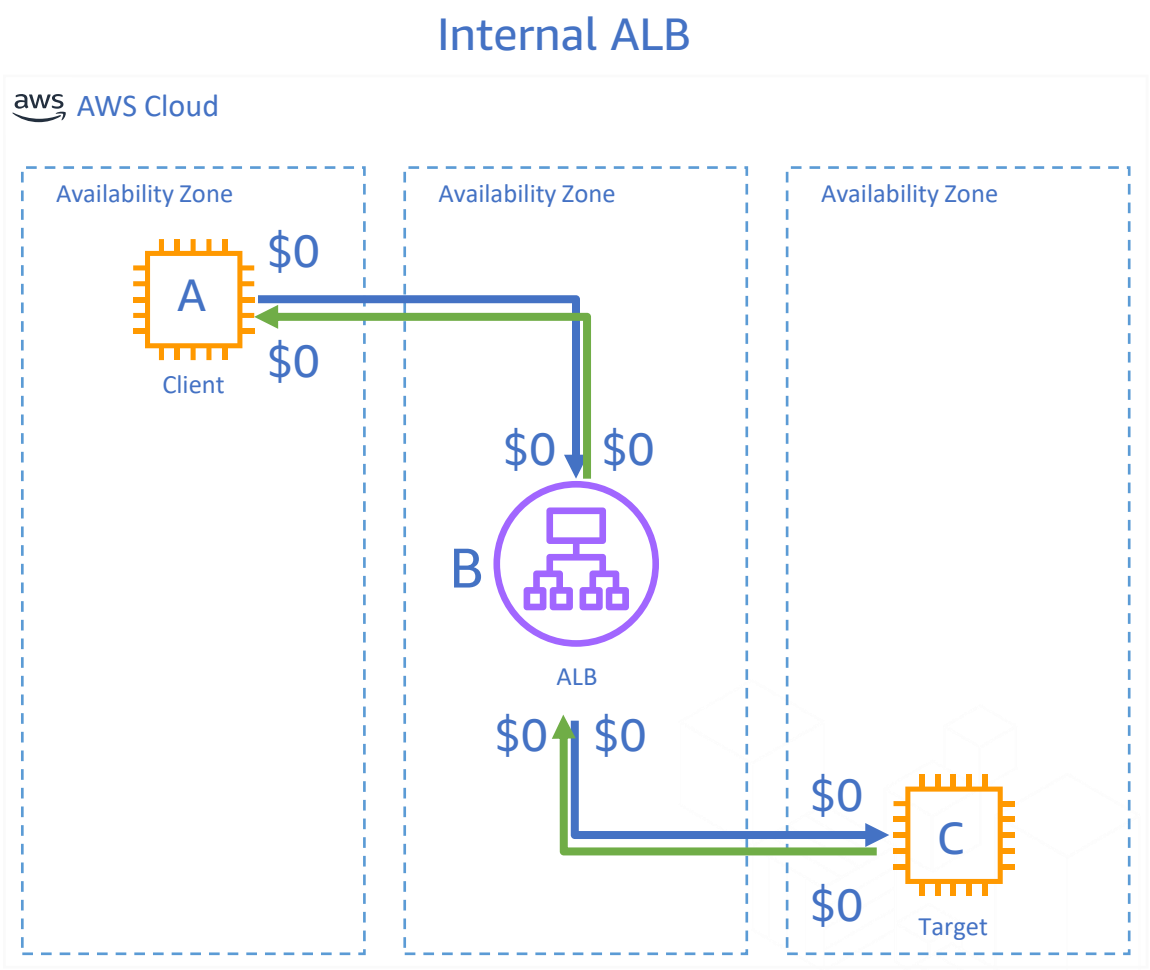


Client IN (Request)
Target OUT (Response)

Cost per GB
A->B $0
B->C $0

Cross Zone Data Transfer Cost with Internal ALB





Client IN (Request)
Target OUT (Response)

Cost per GB
A->B $0.02
B->C $0.02

Client IN (Request)
Target OUT (Response)

Cost per GB
A->B $0
B->C $0

Why use ELB?

Best Practices

- Secure by default
- Isolation of Control and Data Plane
- Health checks

Monitoring & Observability

- Capture request level info thru Access logs
- Monitoring via CloudWatch Metrics
- Visualization using Athena and QuickSight

AWS Integrations

- Targets include EC2, ECS, EKS, Lambda, and ASGs
- Security from VPC, WAF, ACM, IAM, and Cognito
- Infra Management using Elastic Beanstalk & CloudFormation

Where to use ELB in your architecture?

Front Door

- Handles connections from clients at Internet scale
- Highly Available entry point to backend stacks
- May use other AWS services

Microservice Gateway

- Fault tolerant endpoint to communicate across Microservices
- Routing traffic through 3rd party services
- Routing to microservices

Hybrid Connectivity

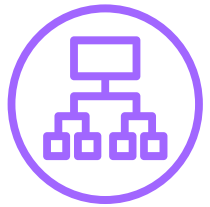
- Load balance targets on-prem
- Makes it easy to failover on-prem applications to the cloud

ELB front door integrations

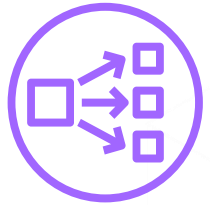
Amazon Route 53

Amazon CloudFront

AWS Global Accelerator

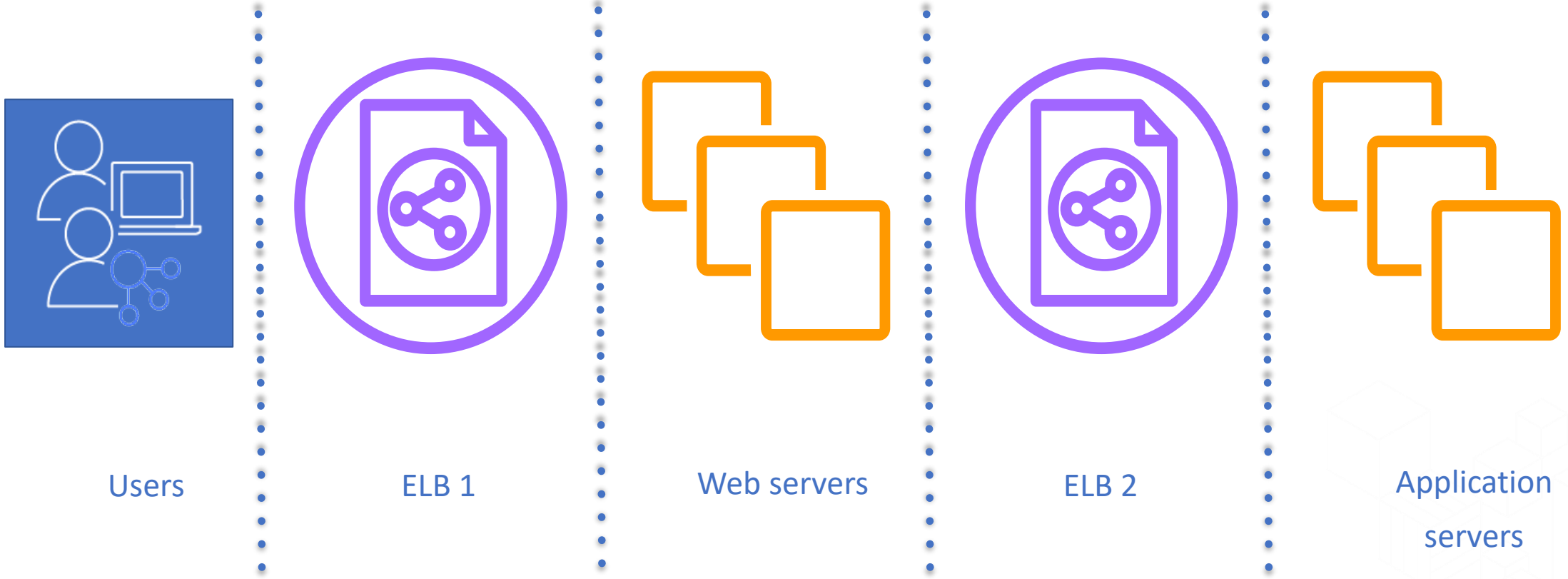


Application Load Balancer



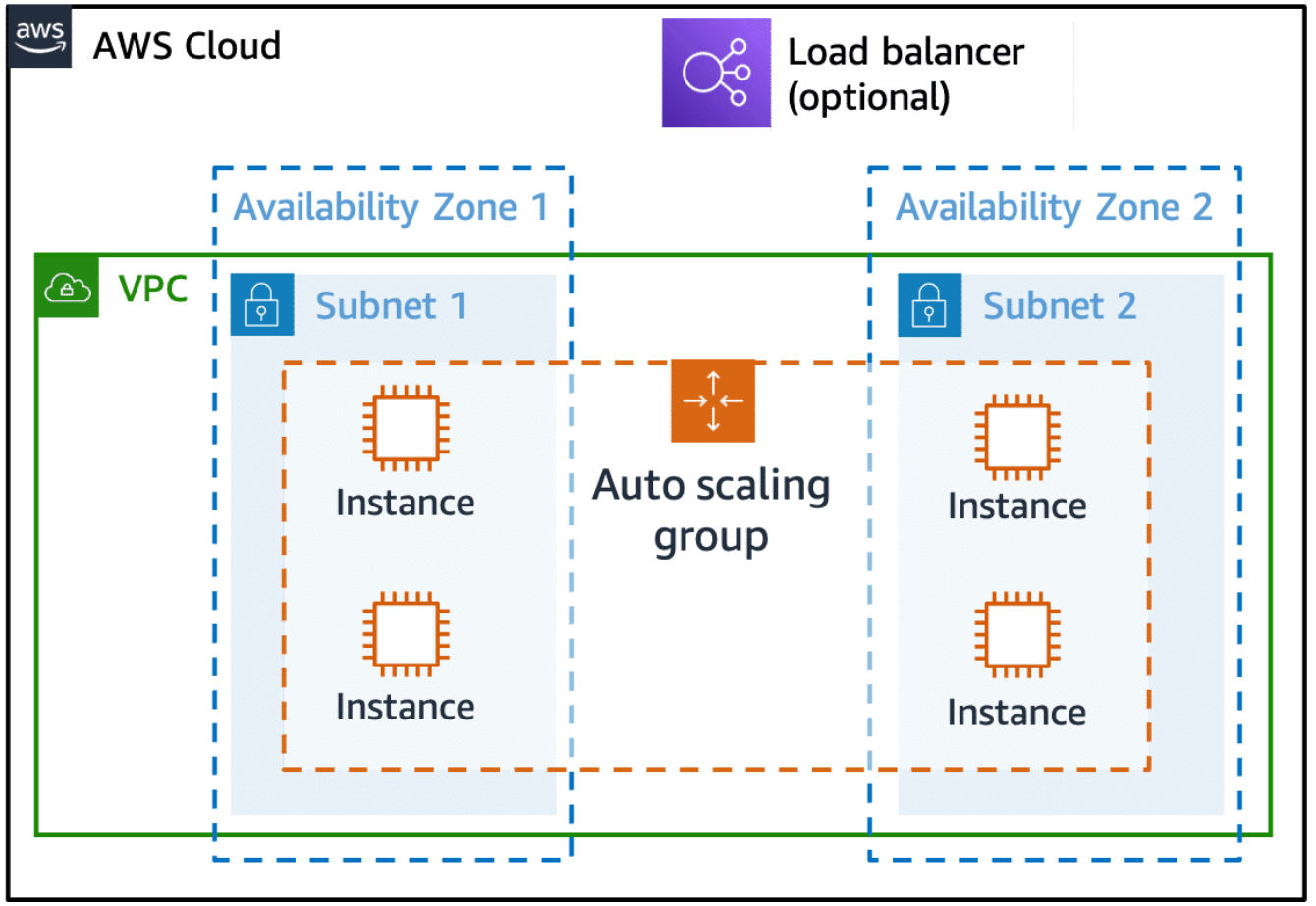
Network Load Balancer

Architectural ELB Sandwich



Auto Scaling Group

"How can we dynamically increase and decrease capacity to meet changing demand?"

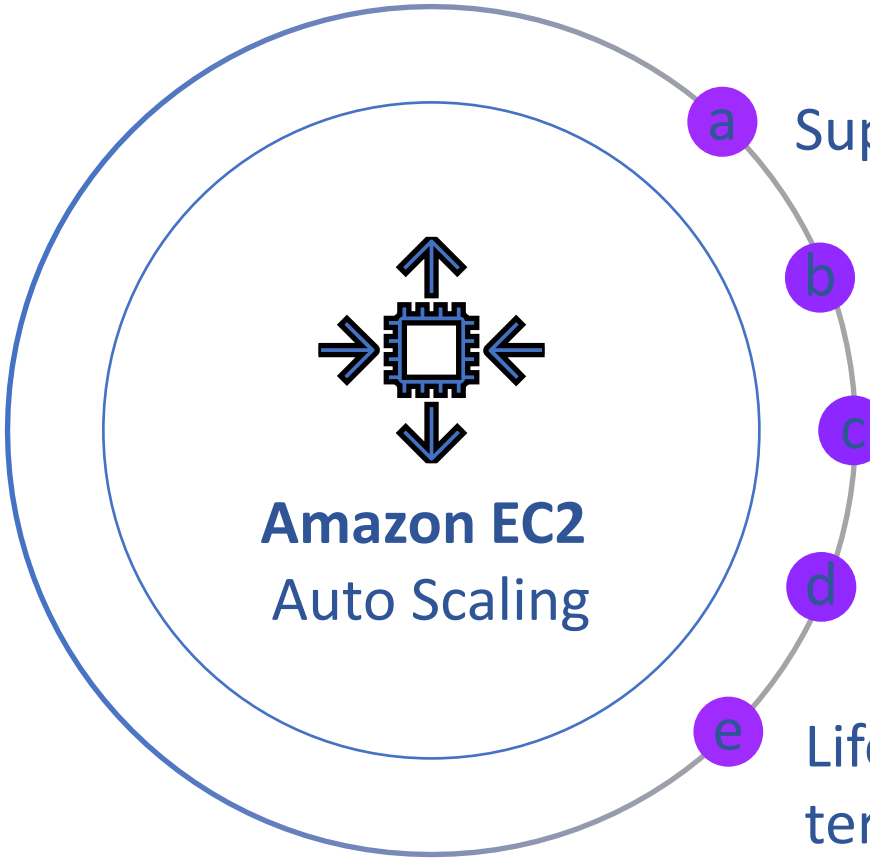


EC2 Auto Scaling

Launch Template	Auto Scaling Group	Scaling Policy
What resources do you need?	Where and how many do you need?	When and for how long do you need them?

• AMI
• Instance type
• Security groups
• Roles
	• VPC and subnets
	• Load balancer
	• Define:
	  ◦ Minimum instances
	  ◦ Maximum instances
	  ◦ Desired capacity (optional)
		• Scheduled
		• On-demand
		• Predictive auto scaling
		• Scale-out policy
		• Scale-in policy

EC2 Auto Scaling Group Features



a. Support for mixed instance and purchase types

b. Replacement of unhealthy & interrupted instances

c. Balanced capacity across availability zones

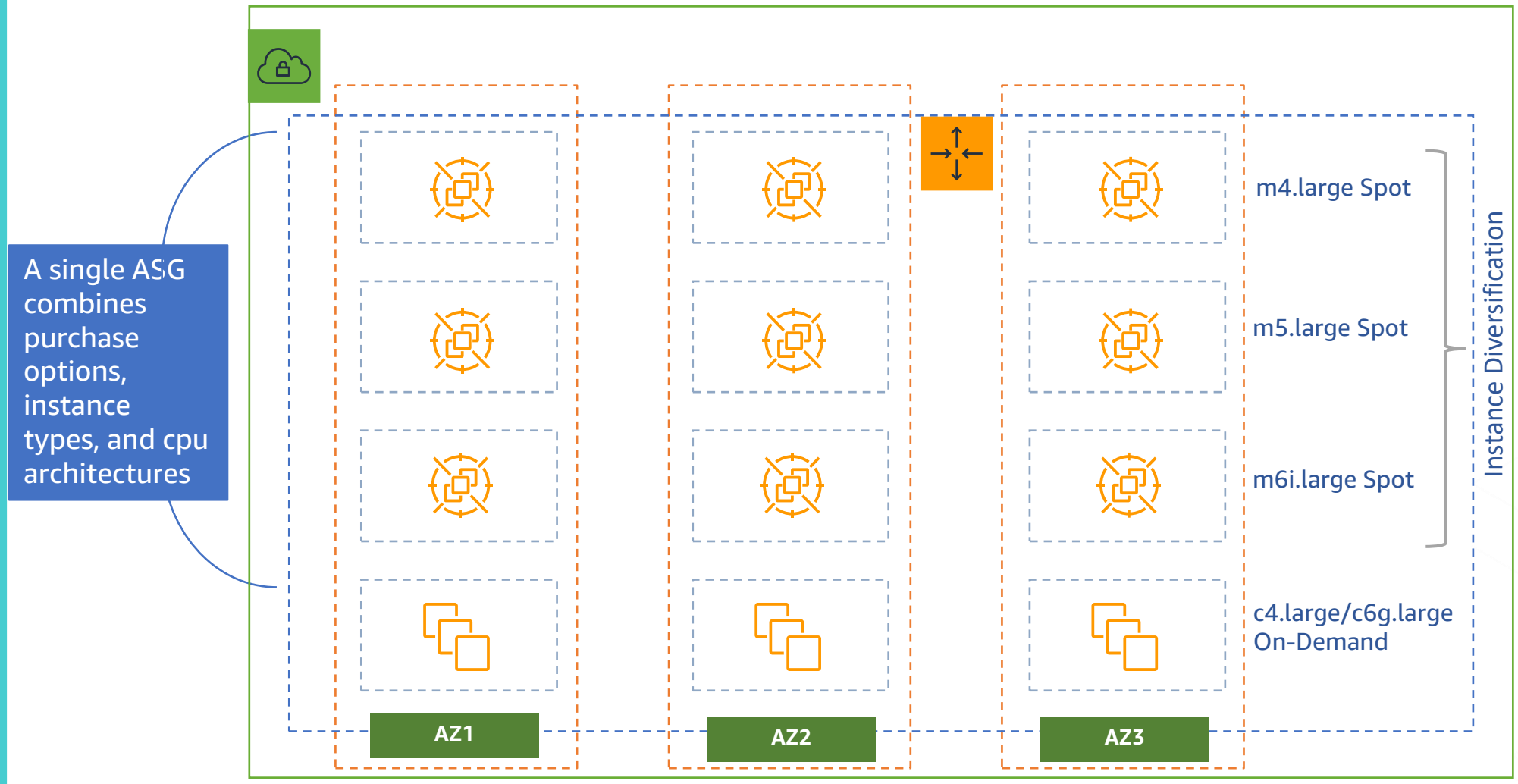
d. Elastic Load Balancer integration

e. Lifecycle hooks, termination policies, instance termination protection

EC2 ASG Features

- **Warm Pools -** pool of pre-initialized EC2 instances that sits alongside an Auto Scaling group to ensure instances are ready to quickly start serving application traffic
- **Capacity Rebalancing -** Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions
- **Life Cycle Hooks -** When a scale-out event occurs runs a script to download and install the needed software packages for your application, making sure that your instance is fully ready before it starts receiving traffic. When a scale-in event occurs, you can invoke an AWS Lambda function or connect to the instance to download logs or other data before the instance is fully terminated.
- **Instance refresh –** if AMI or user data changes, instance types needs to be changed
- **Maximum Instance Lifetime -** requirement to replace your instances on a schedule because of internal security policies or external compliance controls.
- **Attribute based instance type selection** - Provision capacity based on instance attributes depending on compute requirements

Multiple Instance Types and Purchase Types and Architecture



- Purchase Options : EC2 Spot, Savings Plan and Reserved Instances
- Multiple instance types in a group
- Multiple CPU Architecture in a group

Launch template in YAML format

```yaml
AutoScalingGroupName: my-asg
CapacityRebalance: true
DesiredCapacity: 4
MinSize: 4
MaxSize: 100
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 3
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: capacity-optimized
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceRequirements:
        VCpuCount:
          Min: 2
          Max: 4
        MemoryMiB:
          Min: 2048
        CpuManufacturers:
        - intel
VPCZoneIdentifier: subnet-1,subnet-2,subnet-3
```

5. Allow EC2 Auto Scaling to proactively rebalance Spot Instances with elevated risk of interruption to increase application availability

1. Allow EC2 Auto Scaling to provision a combination of On-Demand Instances and Spot Instances across multiple instance types

3. Set a base capacity for On-Demand Instances

4. Use the price-capacity-optimized allocation strategy to find the optimal spare capacity to reduce cost of interruptions while lowering costs (shown is capacity-optimized)

2. Provide Instance Attributes and Availability Zones to allow EC2 Auto Scaling to pick the right instances, optimize for performance, cost, and availability

Thank You