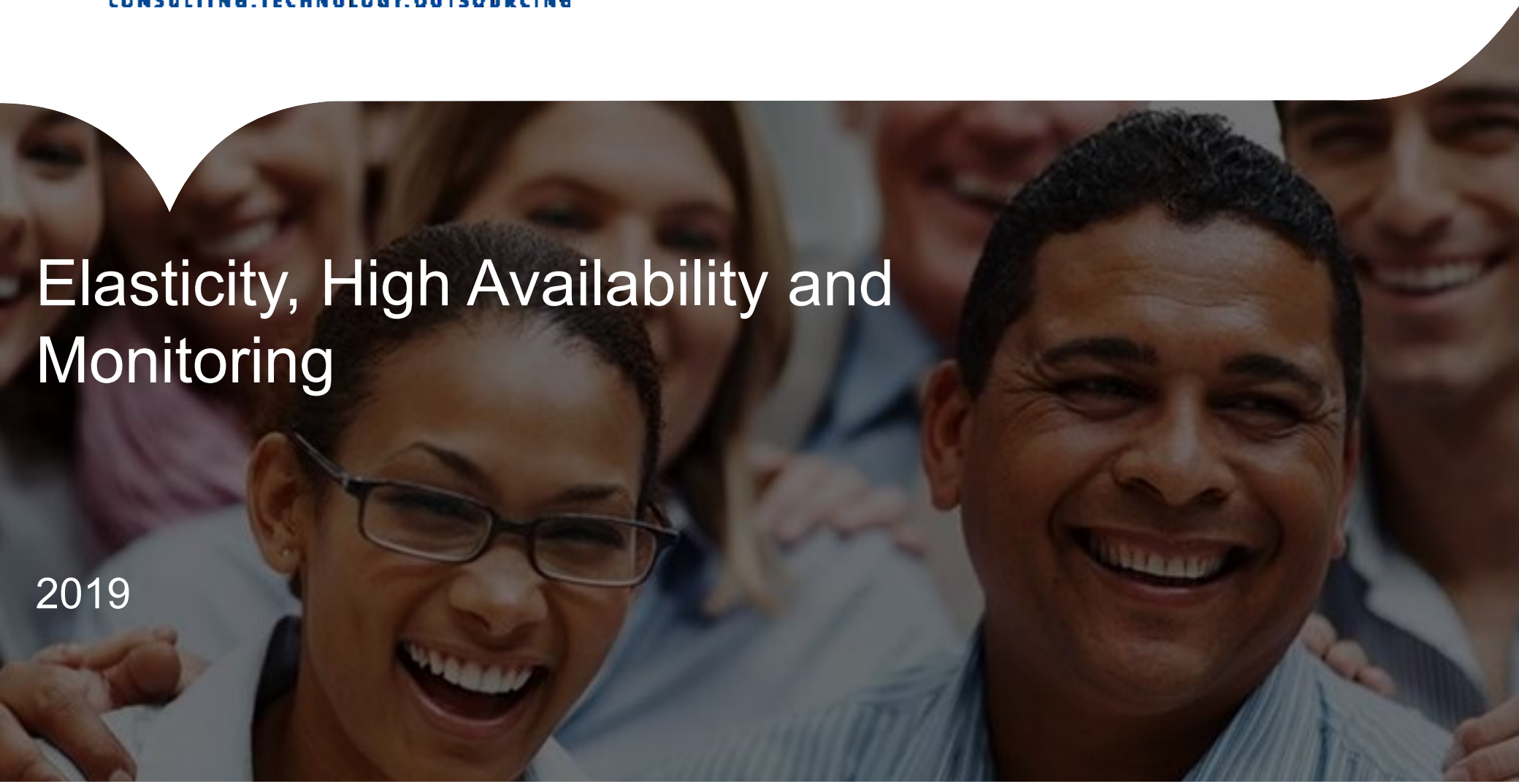
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

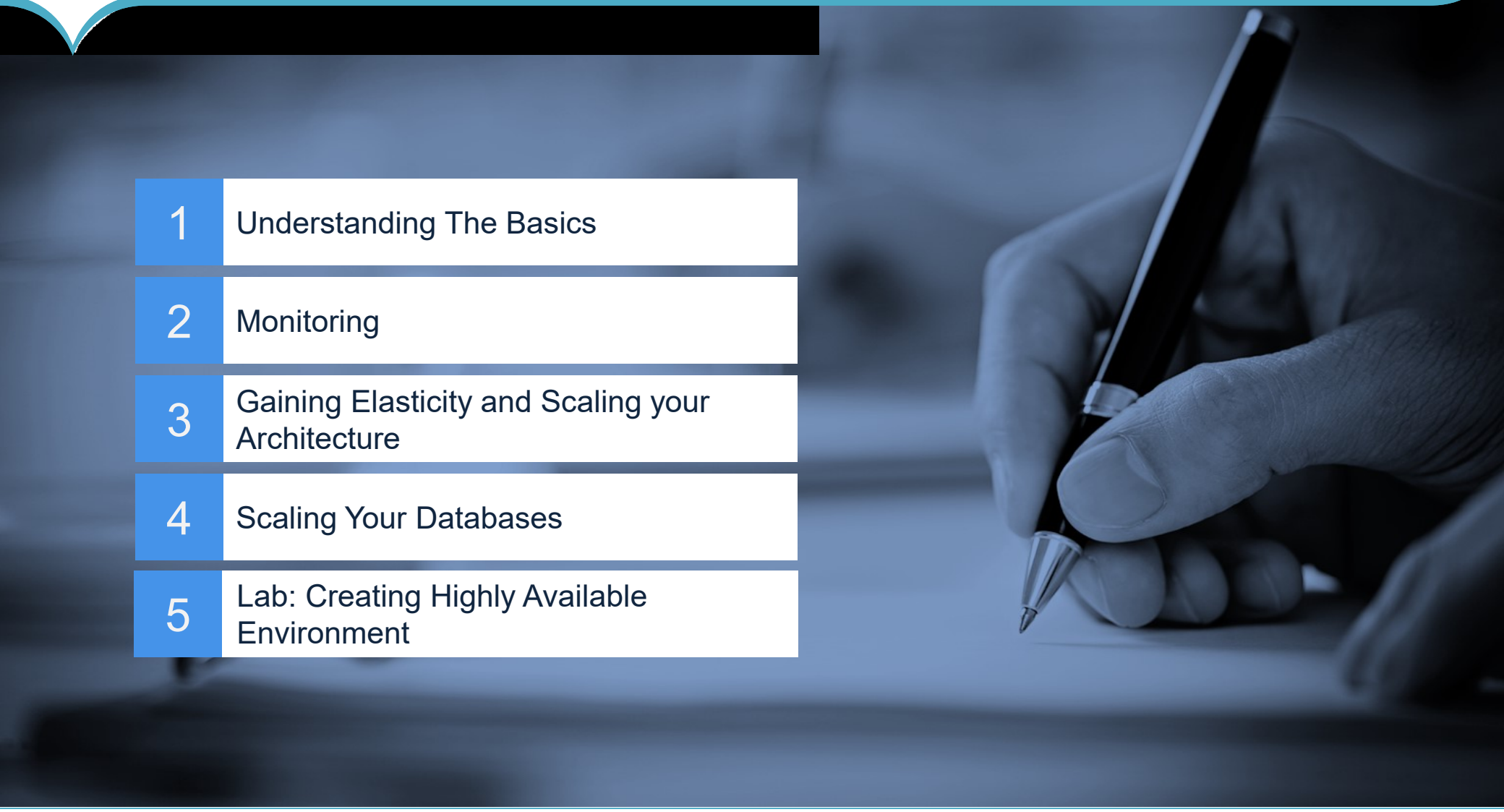


Elasticity, High Availability and Monitoring

2019

People matter, results count.

Agenda



1. Understanding The Basics
2. Monitoring
3. Gaining Elasticity and Scaling your Architecture
4. Scaling Your Databases
5. Lab: Creating Highly Available Environment

High Availability Factors

Fault tolerance

The **built-in redundancy** of an application's components

Recoverability

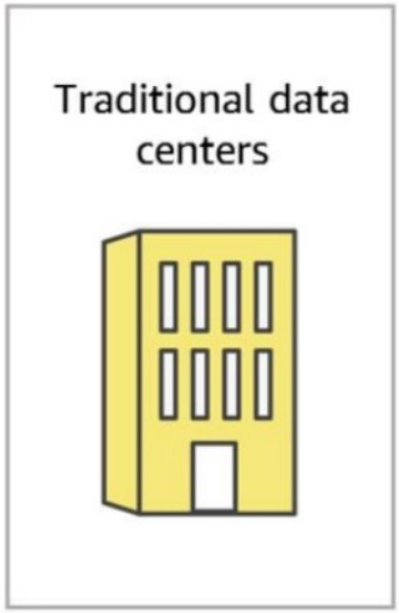
The process, policies and procedures related to **restoring service** after a catastrophic event

Scalability:

The ability of an application to **accommodate growth** without changing design

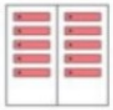
Understanding The Basics

What does Inelasticity look like?



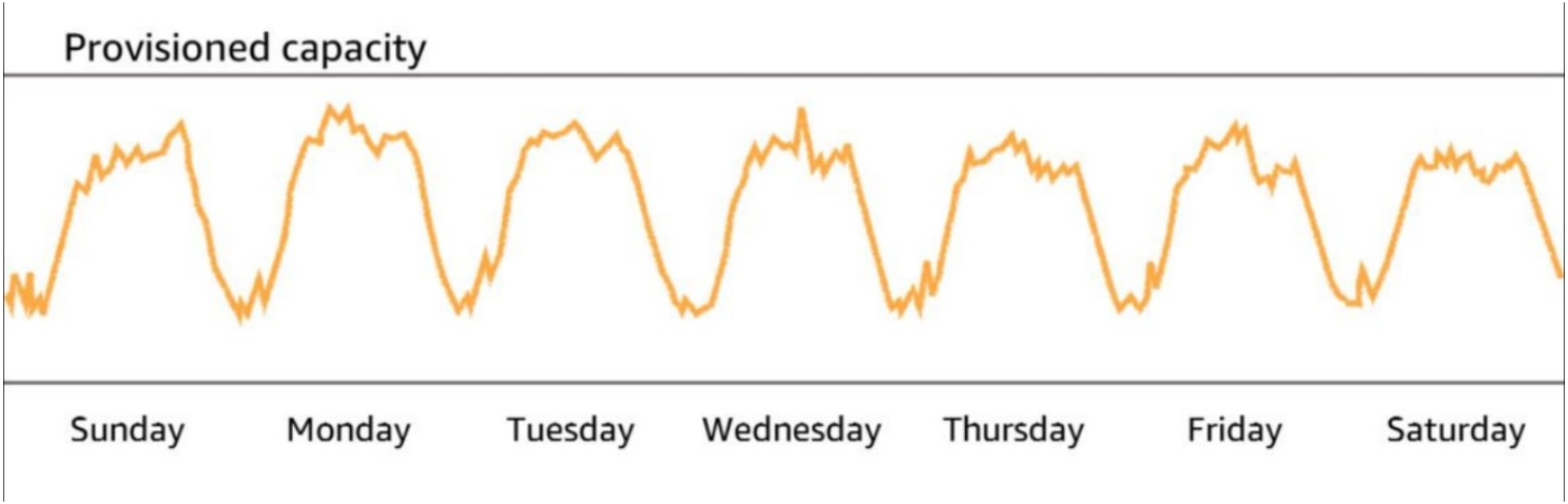
Pay for your resources up front and hope they cover your demand

Or

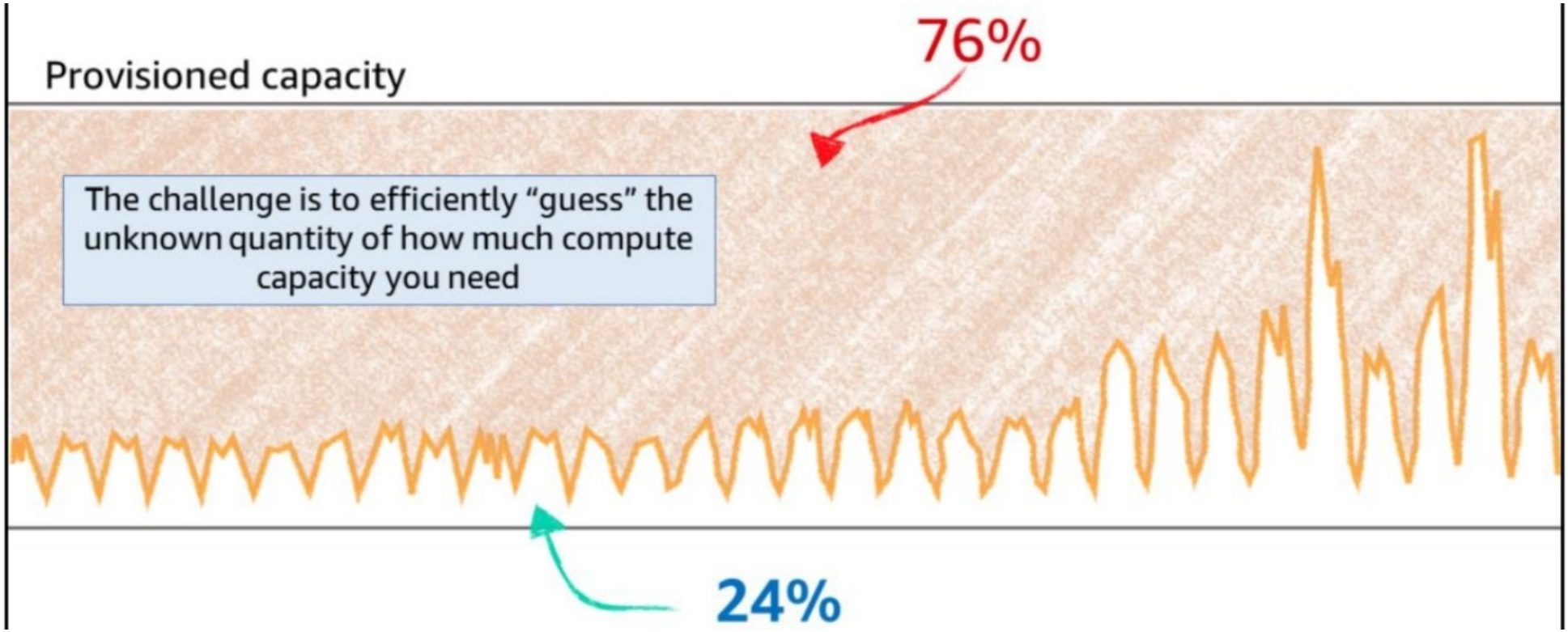


Too many extra resources, wasting money and burning electricity

Example: Amazon.com



November Traffic to Amazon.com



What is Elasticity?

An elastic infrastructure can intelligently expand and contract as its capacity need change.

Examples:

- Increasing the number of web servers when traffic spikes
- Lowering write capacity on your database when traffic goes down
- Handling the day-to-day fluctuation of demand throughout your architecture

Two Types of Elasticity

Time-Based

- Turning off resources when they are not being used (Dev and Test environments)

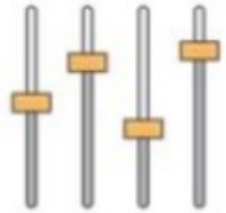
Volume-Based

- Matching scale to the intensity of your demand (making sure you have enough compute power)

Monitoring

The Reasons For Monitoring

Operational Health



Resource Utilization

Application Performance

Security Auditing

Monitoring to Understand Cost

To Create more flexible and elastic architecture,

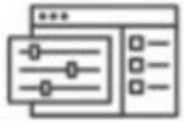
you should know where you are spending money.

Cost Explorer

Generates reports

13 months of data

Provides estimates



See patterns in your spending

Monitoring Infrastructure with Amazon CloudWatch

- Collects and tracks metrics for your resources
- Enables you to create alarm and send notifications
- Can trigger changes in capacity in a resource, based on rules that you set

The Ways CloudWatch Responds

Metrics

Logs

Alarms

Events

Rules

Targets

CloudWatch Metrics

Metrics

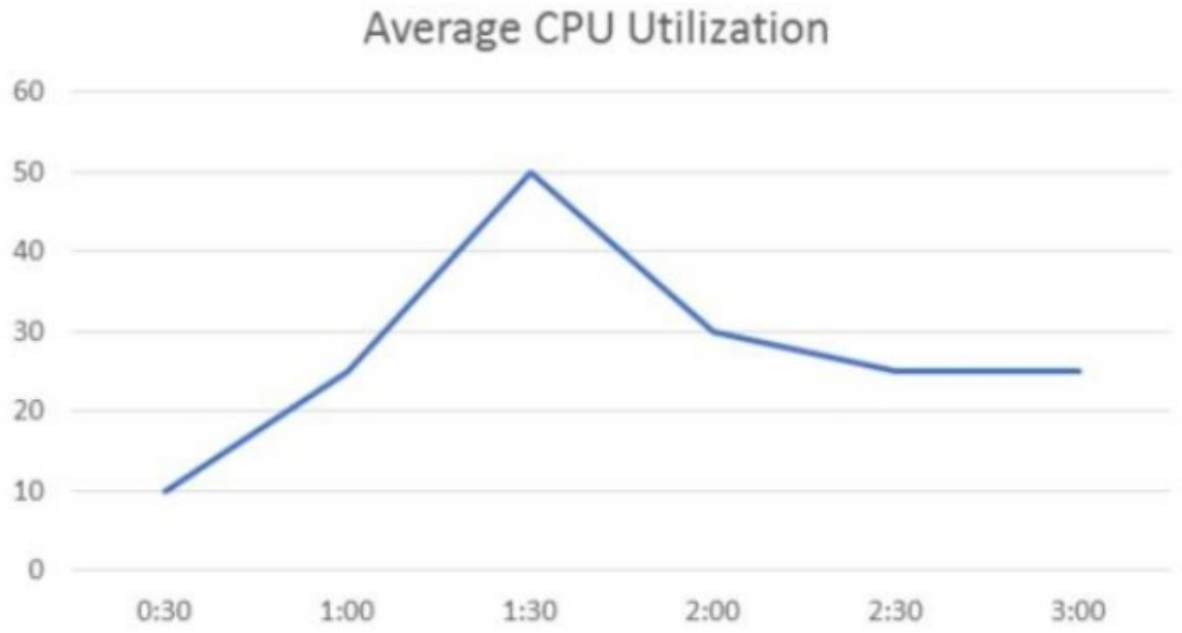
Logs

Alarms

Events

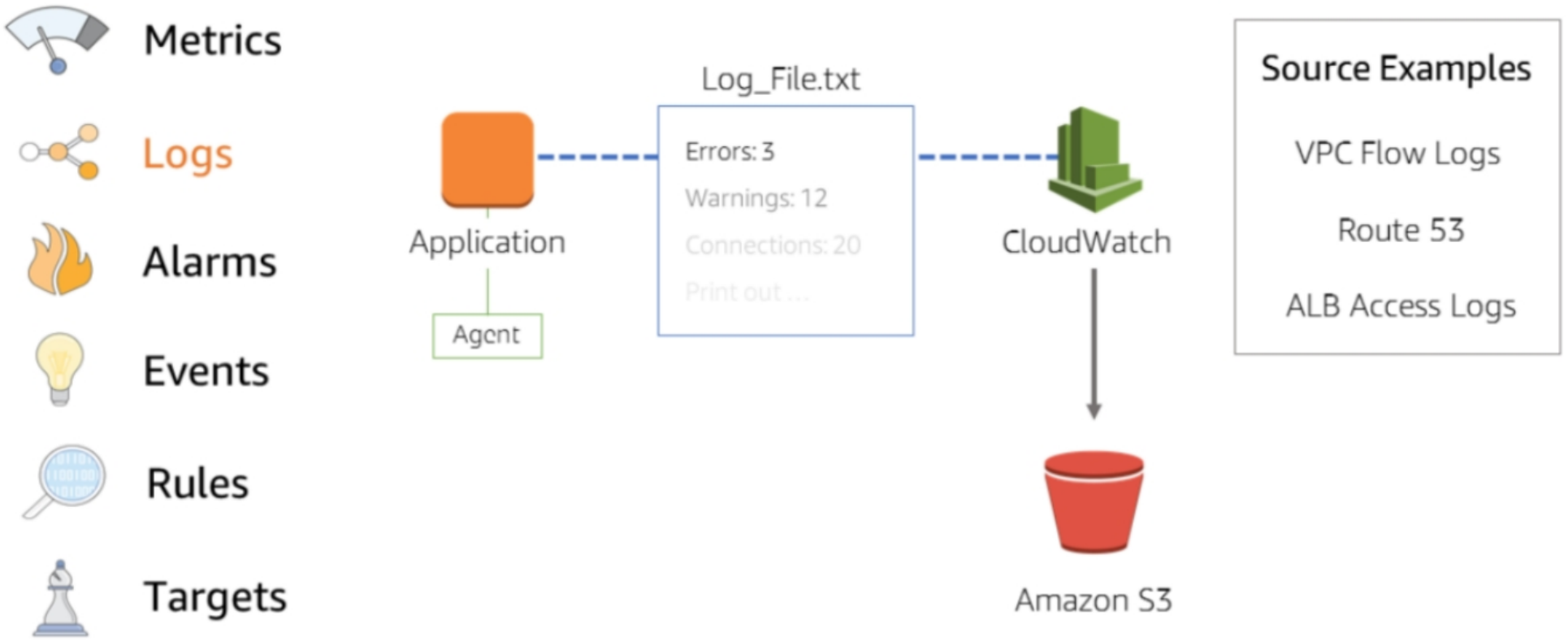
Rules

Targets

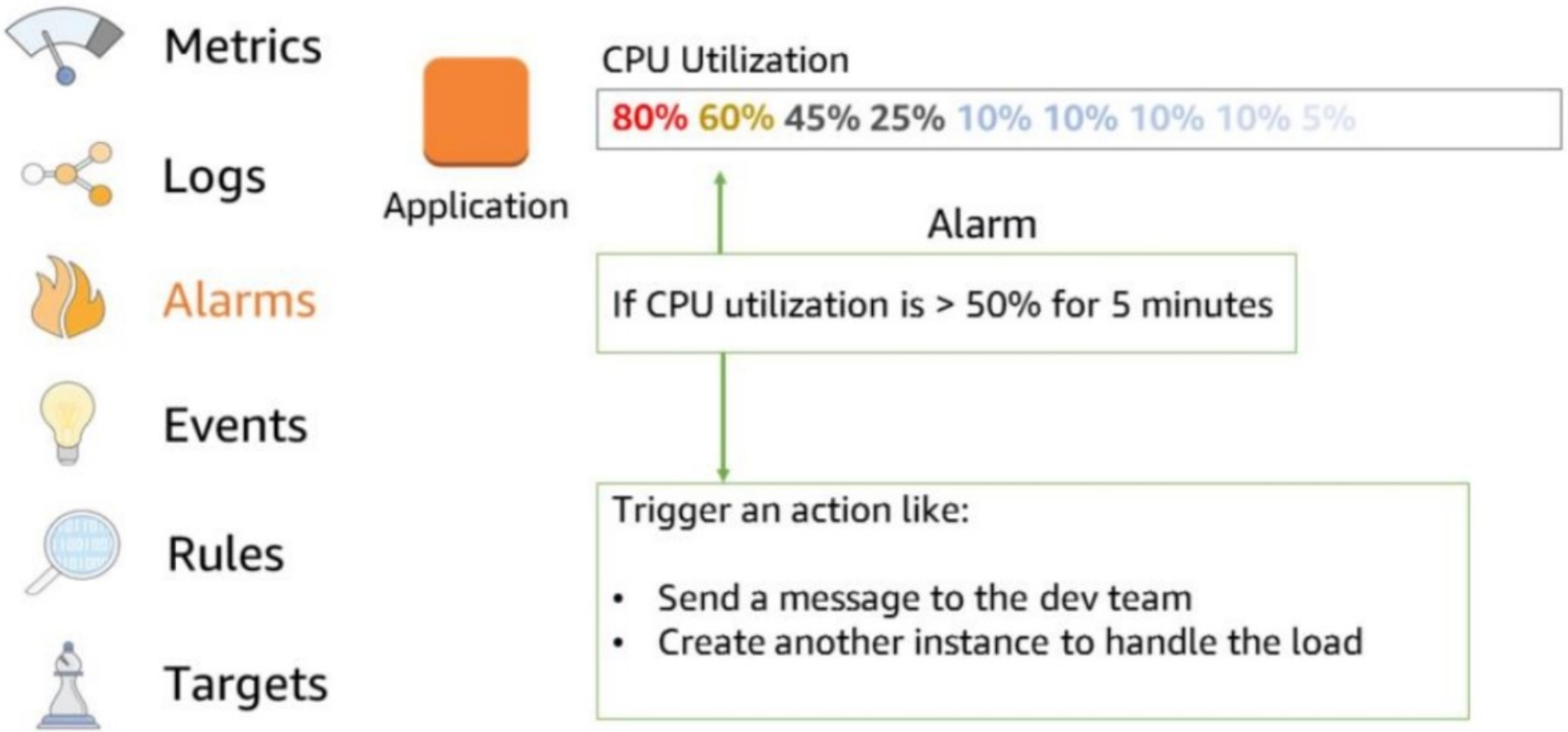


Metric data is kept for a period of 15 months

CloudWatch Logs



CloudWatch Alarms



CloudWatch Events

Metrics

Logs

Alarms

Events

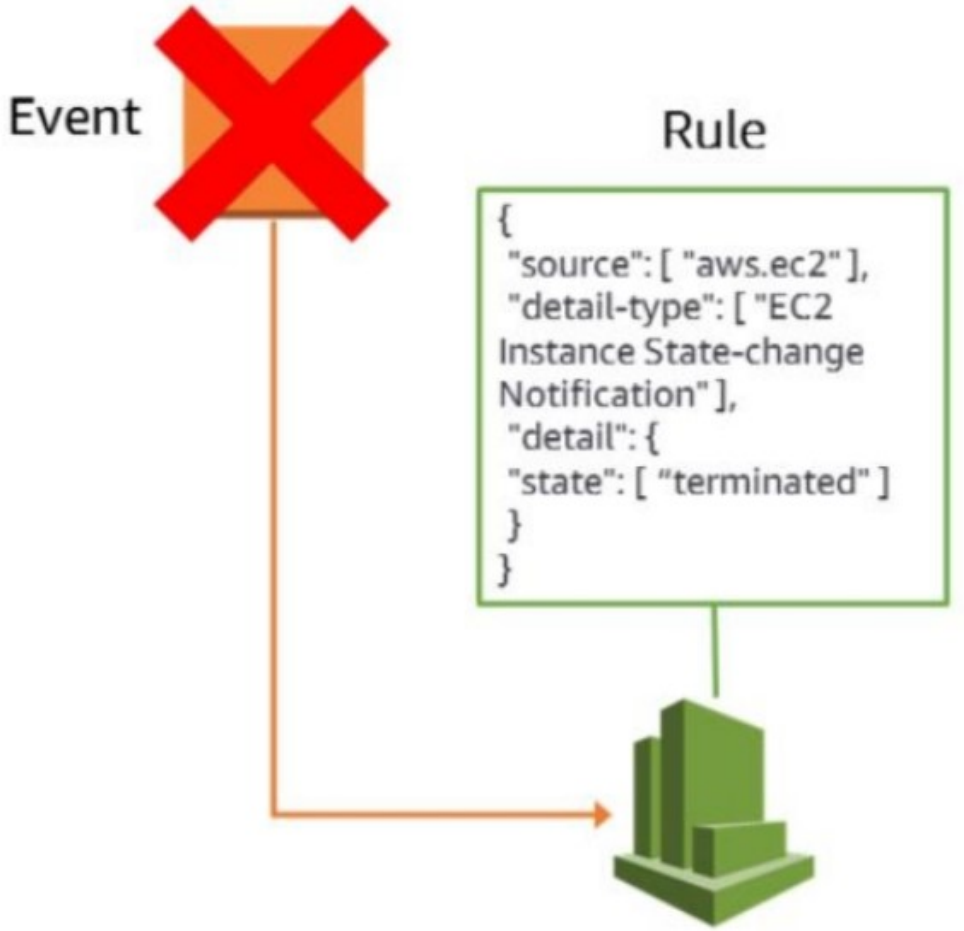
Rules

Targets

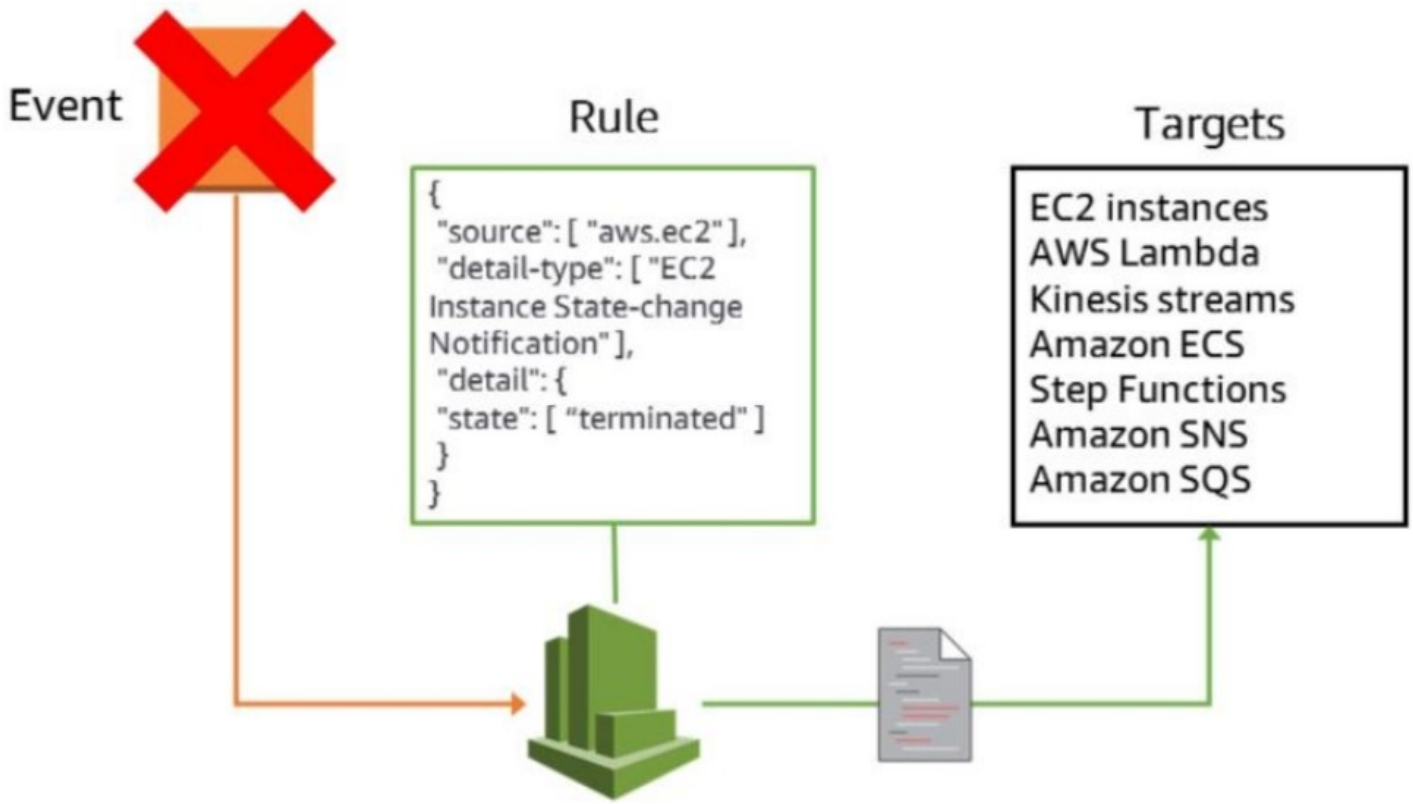
Event Examples

Console sign-in
Auto Scaling state change
EC2 instance state change
EBS volume creation
Any API call

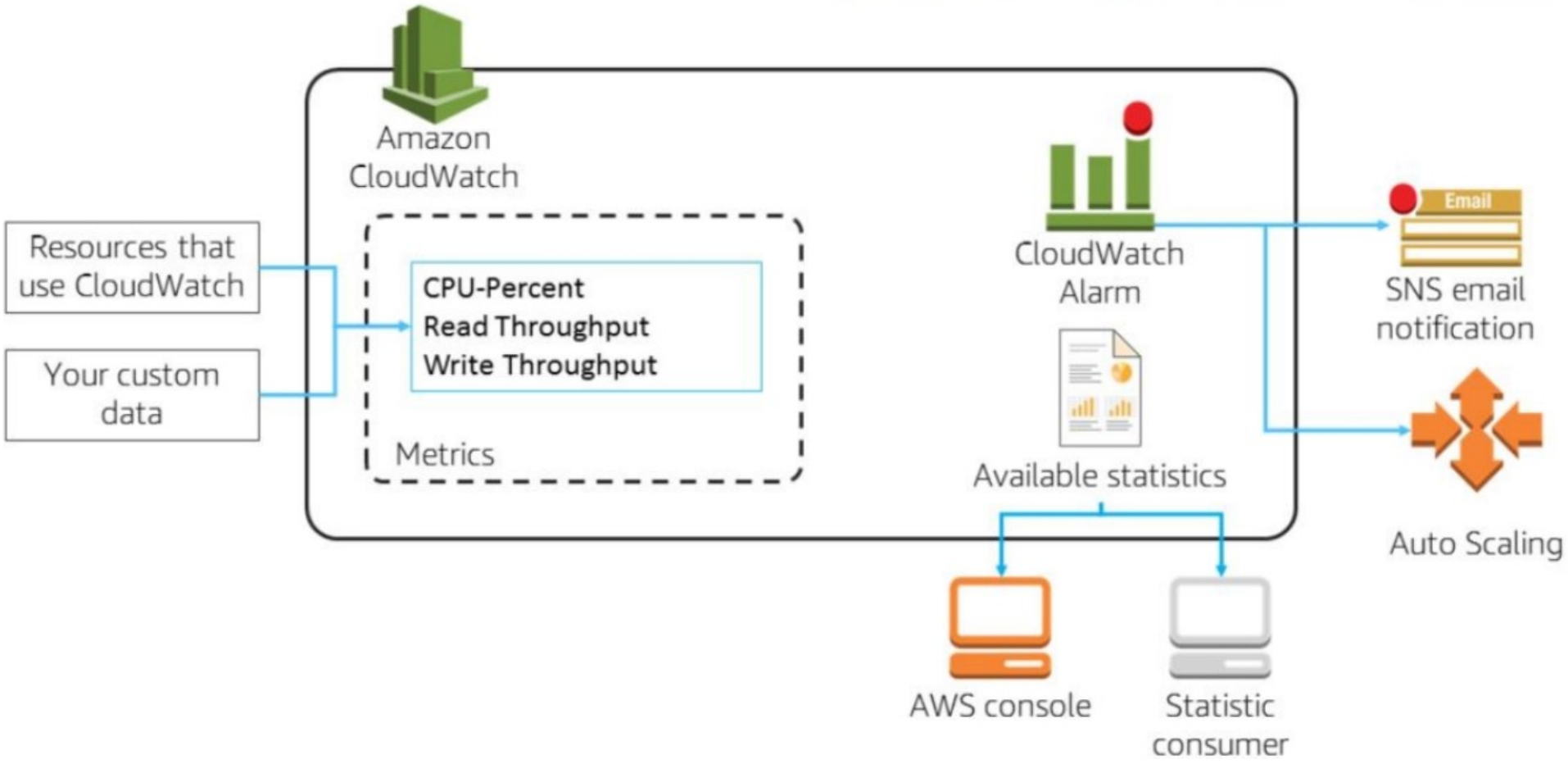
CloudWatch Rules



CloudWatch Targets

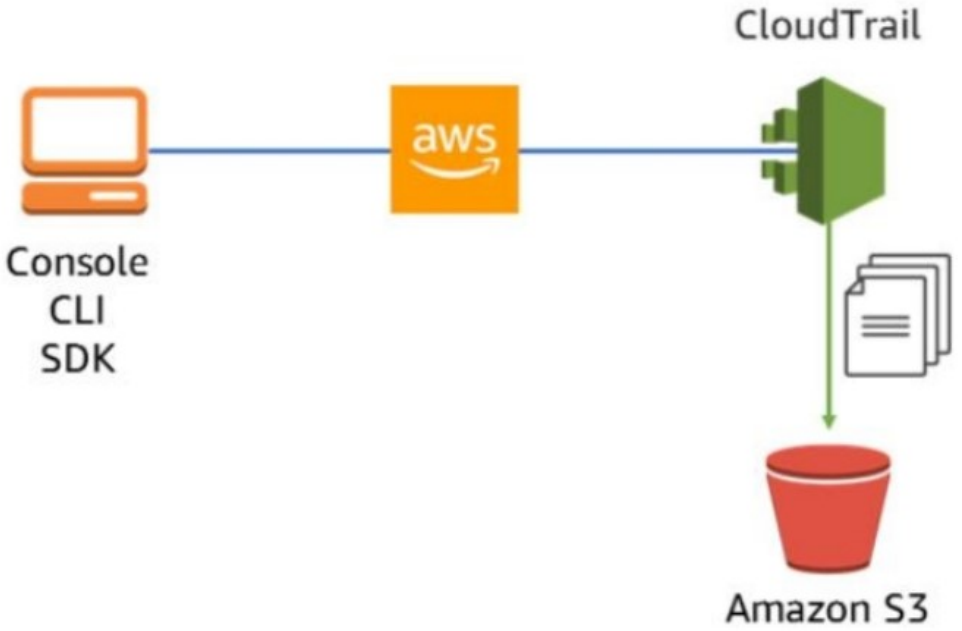


Visualizing CloudWatch



Monitoring Your Users with AWS CloudTrail

CloudTrail records all API calls made in your account and saves logs in your designated Amazon S3 bucket.



Monitoring your Network with VPC Flow Logs

VPC Flow Logs

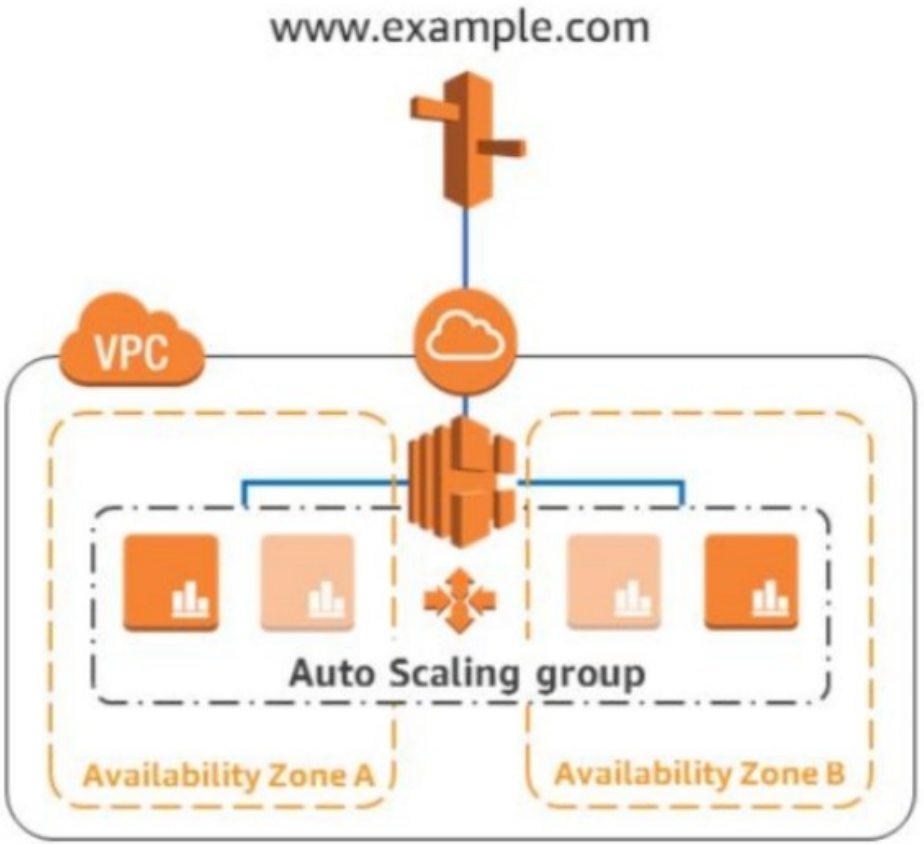
- Captures **traffic flow details** in your VPC
- Accepted, rejected or all traffic
- Can be enabled for **VPCs, subnets and ENIs**
- Logs published to **CloudWatch Logs**

Gaining Elasticity and Scaling your Architecture

Using Auto Scaling to Provide Elasticity

Amazon EC2 Auto Scaling

- **Launches or terminates instances** based on specified conditions
- Automatically **registers new instances** with load balancers when specified
- Can launch **across Availability Zones**



Ways To Auto Scale

Scheduled

Good for predictable workloads

Scale based on time or day

Use case: Turning off your Dev and Test instances at night

Ways To Auto Scale

Scheduled

Good for predictable workloads

Scale based on time or day

Use case: Turning off your Dev and Test instances at night

Dynamic

Excellent for general scaling

Supports target tracking

Use case: Scaling based on CPU utilization

Ways To Auto Scale

Scheduled

Good for predictable workloads

Scale based on time or day

Use case: Turning off your Dev and Test instances at night

Dynamic

Excellent for general scaling

Supports target tracking

Use case: Scaling based on CPU utilization

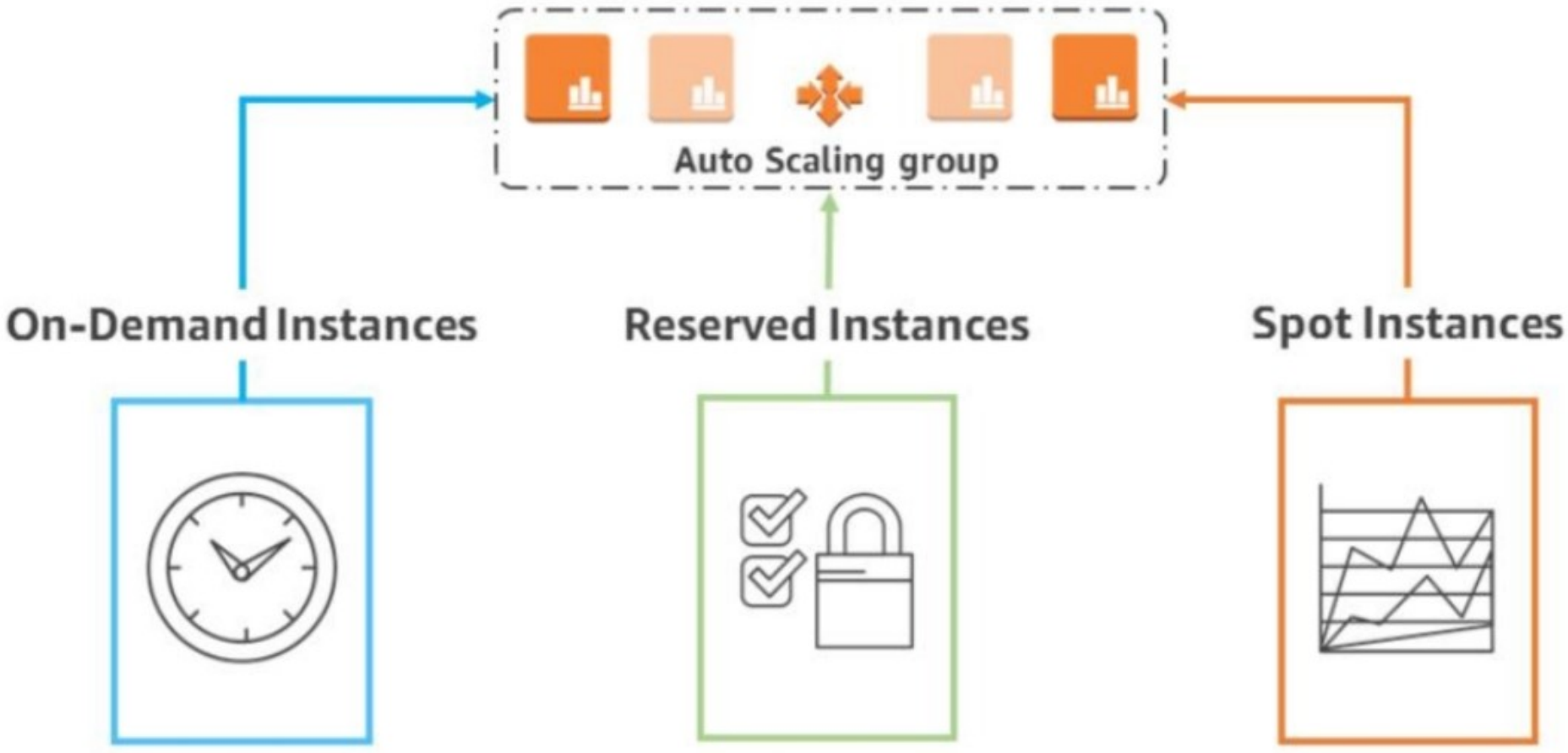
Predictive

Easiest to use

Machine learning based scaling

Use case: No longer need to manually adjust rules

Auto Scaling – Purchasing Options



Auto Scaling Minimum Capacity

Auto Scaling group defines:

- Desired Capacity
- Minimum Capacity
- Maximum Capacity

What would be a good **minimum** capacity to set it to?

What would be a good **maximum** capacity to set it to?

Auto Scaling Considerations

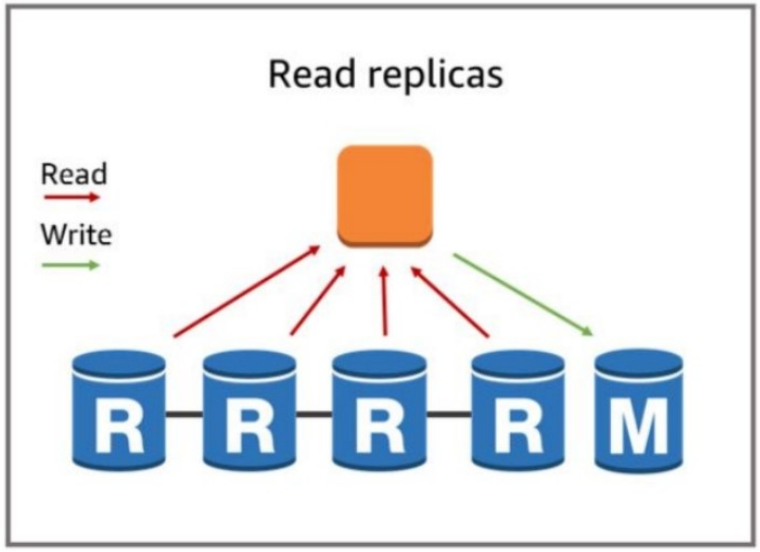
- You might need to combine **multiple** types of autoscaling
- Your architecture might require more hands scaling using: **Step Scaling**
- Some architectures need to **scale on two or more metrics** (e.g. not just CPU)
- Try to **scale out early and fast**, while **scaling in slowly** over time
- Use **lifecycle hooks**

  Perform custom action as Auto Scaling launches or terminates instances

Remember: Instances can take several minutes after launch to be fully usable.

Scaling Your Databases

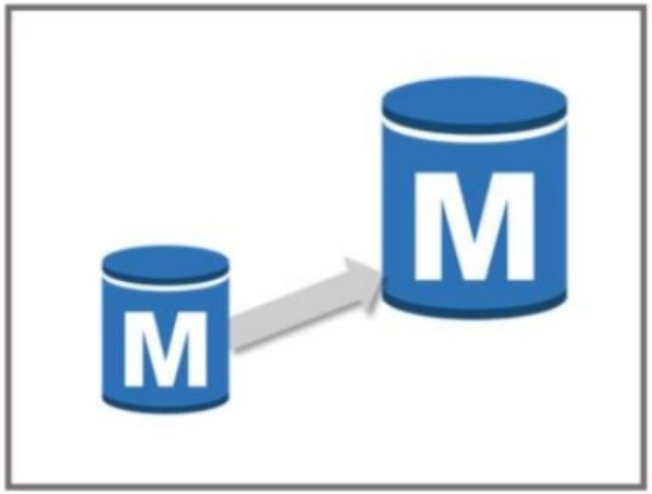
Horizontal Scaling with Read Replicas: Amazon RDS



- Horizontally scale for read-heavy workloads
- Offload reporting
- Keep in mind:
  - Replication is asynchronous
  - Currently available for: Amazon Aurora, MySQL, Maria DB and PostgresSQL

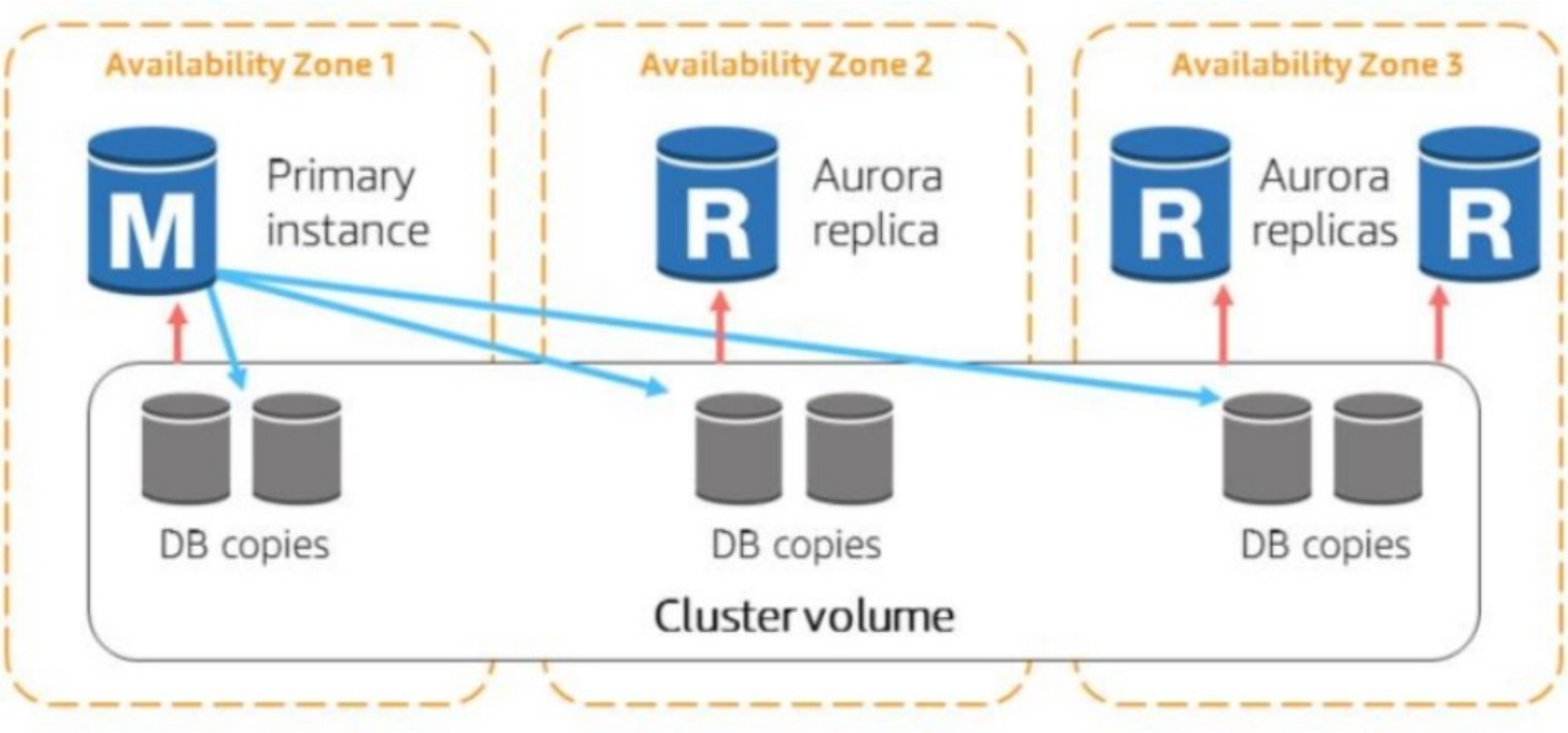
Scaling Amazon RDS: Push-Button Scaling

- Scale nodes **vertically** up or down
- From **micro to 8xlarge** and everything in-between
- Scale vertical often with **no downtime***



Aurora DB Cluster

Each Aurora DB cluster can have up to 15 Aurora replicas



Aurora Serverless

Responds to your application automatically:

- Scales capacity
- Shut down
- Start up

Pay for number of ACUs used

Good for spiky, unpredictable workloads.

Scaling Amazon RDS Writes with Database Sharding

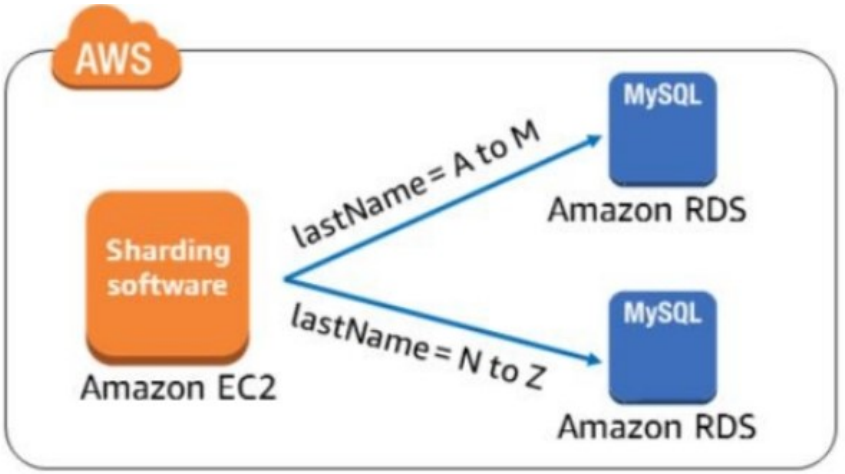
Without shards, all data resides in **one partition**

- Example: Users by last name, A to Z, in one database

With **sharding**, split your data into **large chunks** (shards)

- Example: Users by last name, A through M, in one database; N through Z in another database

In many circumstances, sharding gives you **higher performance** and **better operating efficiency**



DynamoDB Scaling – Two Ways

Auto Scaling

Default for all new tables

Specify upper and lower bounds

Use case: General scaling, great solution for most applications.

DynamoDB Scaling – Two Ways

Auto Scaling

Default for all new tables

Specify upper and lower bounds

Use case: General scaling, great solution for most applications.

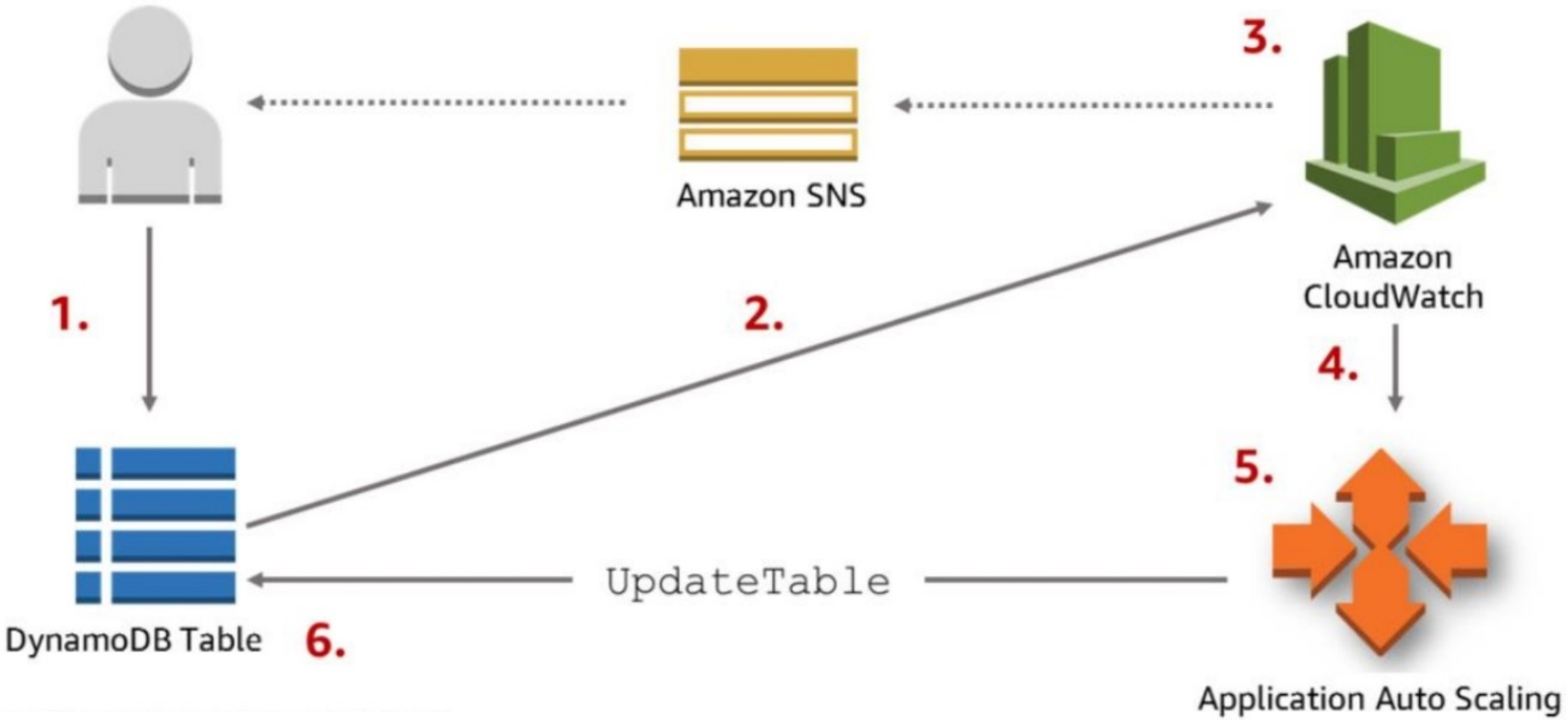
On-Demand

Pay per request

No more provisioning

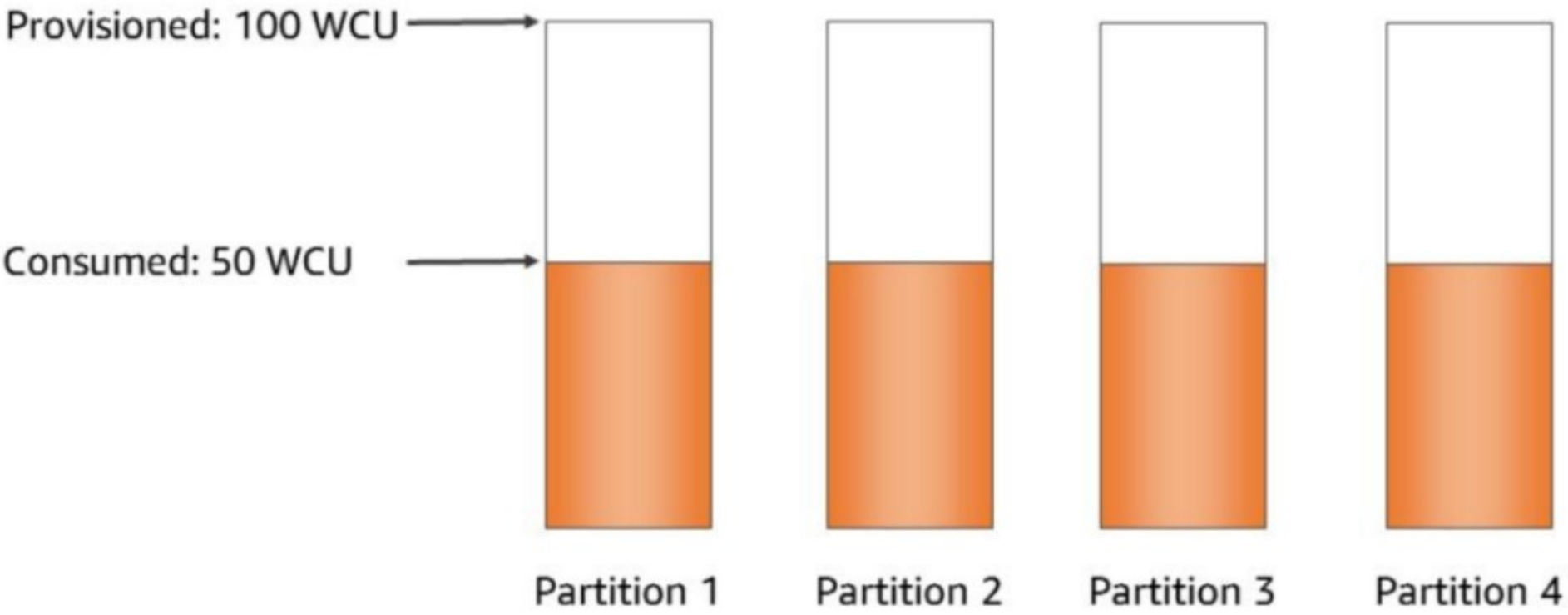
Use case: Spiky, unpredictable workloads. Rapidly accommodates to need.

DynamoDB - Auto Scaling In Action



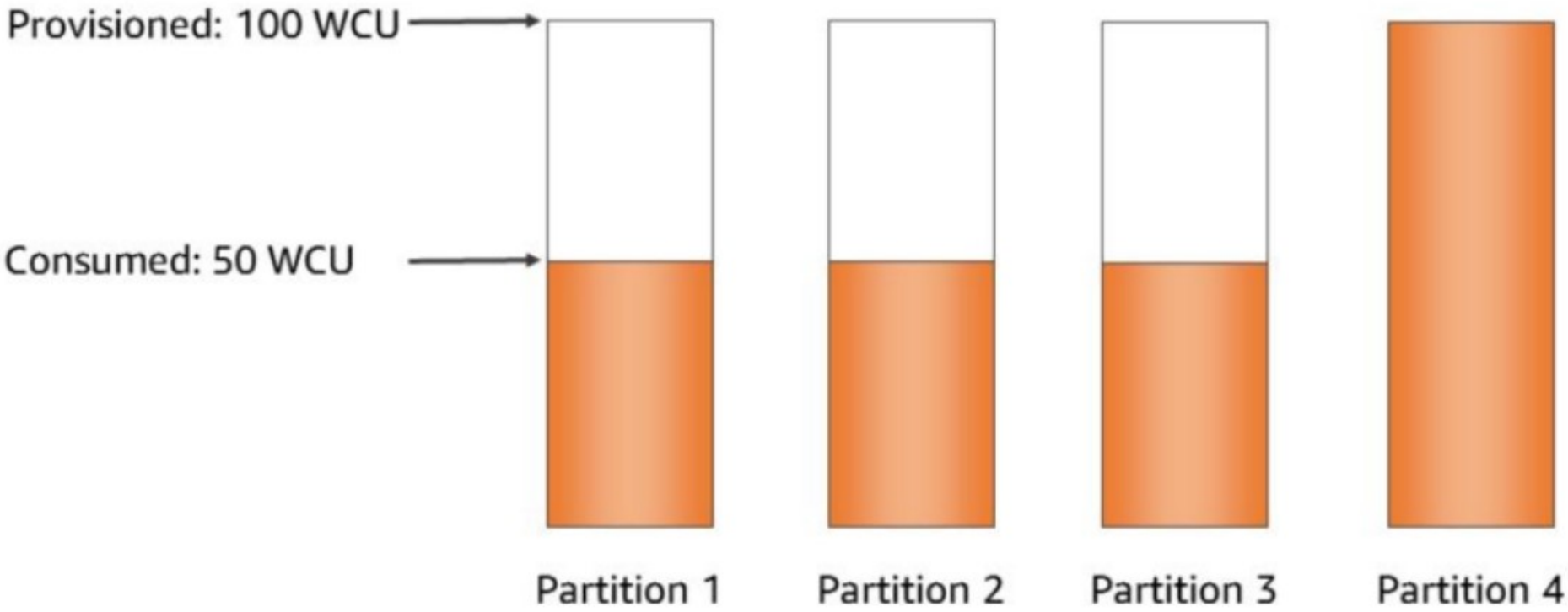
DynamoDB Adaptive Capacity

Example table with adaptive capacity
Total provisioned capacity = 400 WCUs
Total consumed capacity = 200 WCUs



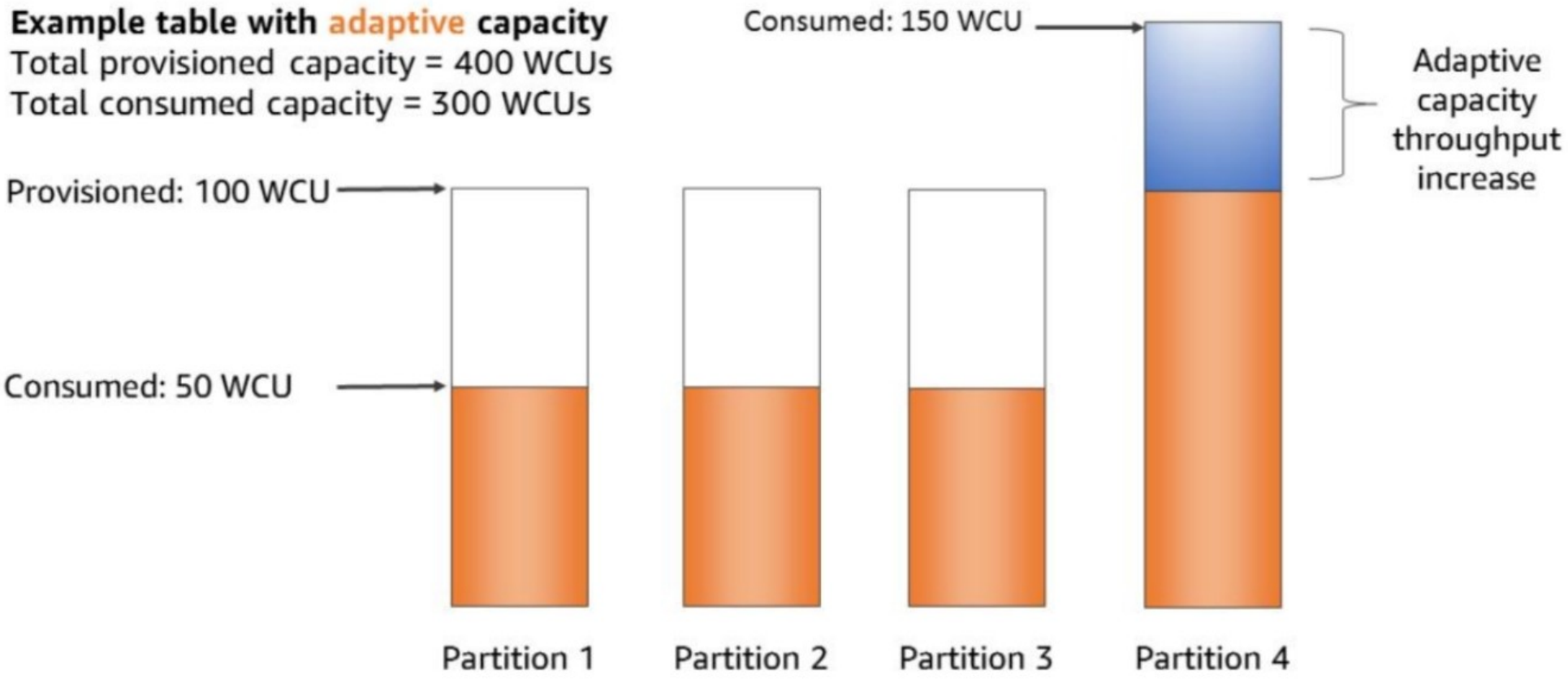
DynamoDB Adaptive Capacity

Example table with adaptive capacity
Total provisioned capacity = 400 WCUs
Total consumed capacity = 250 WCUs



DynamoDB Adaptive Capacity

Example table with adaptive capacity
Total provisioned capacity = 400 WCUs
Total consumed capacity = 300 WCUs



Adaptive Capacity Does Not Fix Hot Keys and Hot Partitions

Partition key value	Uniformity
User ID, where the application has many users	Good
Status code, where there are only a few possible status codes	Bad
Item creation date, rounded to the nearest time period (for example, day, hour, or minute)	Bad
Device ID, where each device accesses data at relatively similar intervals	Good
Device ID, where even if there are many devices being tracked, one is by far more popular than all of the others	Bad

Lab: Creating Highly Available Environment

Lab: Creating Highly Available Environment

"I Want resilient infrastructure“

Technologies used:

- Amazon VPC
- Application Load Balancer
- Amazon EC2 Auto Scaling group
- Amazon RDS

Lab: Creating Highly Available Environment

"I Want resilient infrastructure“

Technologies used:

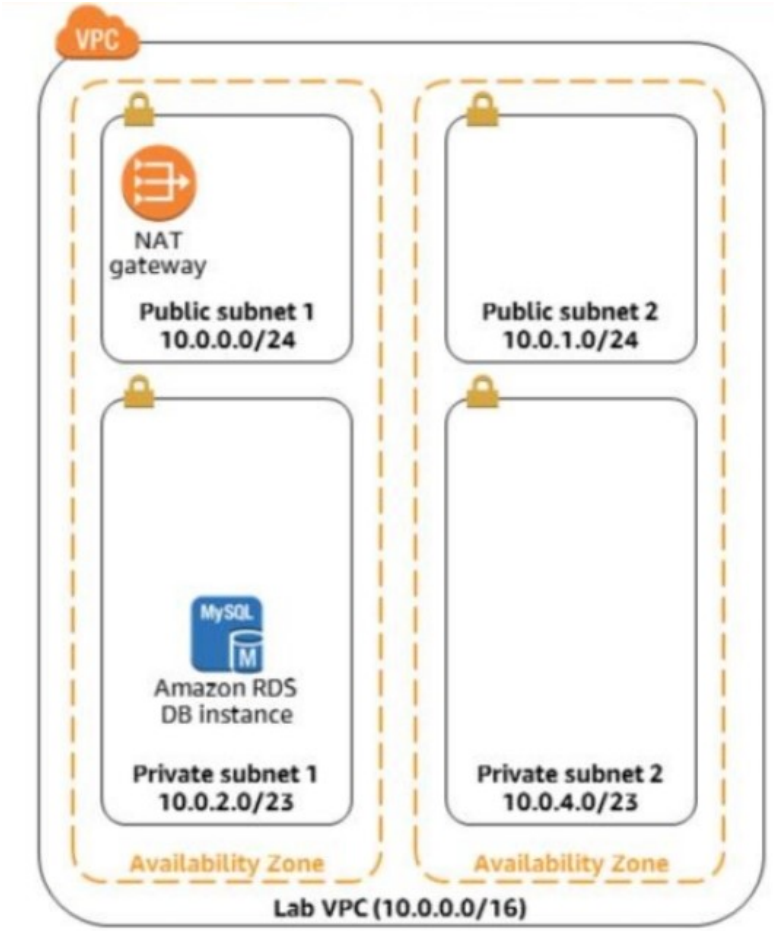
- Amazon VPC
- Application Load Balancer
- Amazon EC2 Auto Scaling group
- Amazon RDS

Lab: Creating Highly Available Environment

Provided at start of Lab:

- VPC across two Availability Zones
- 2x public subnets
- 2x private subnets
- 1x NAT gateway
- Amazon RDS DB instance

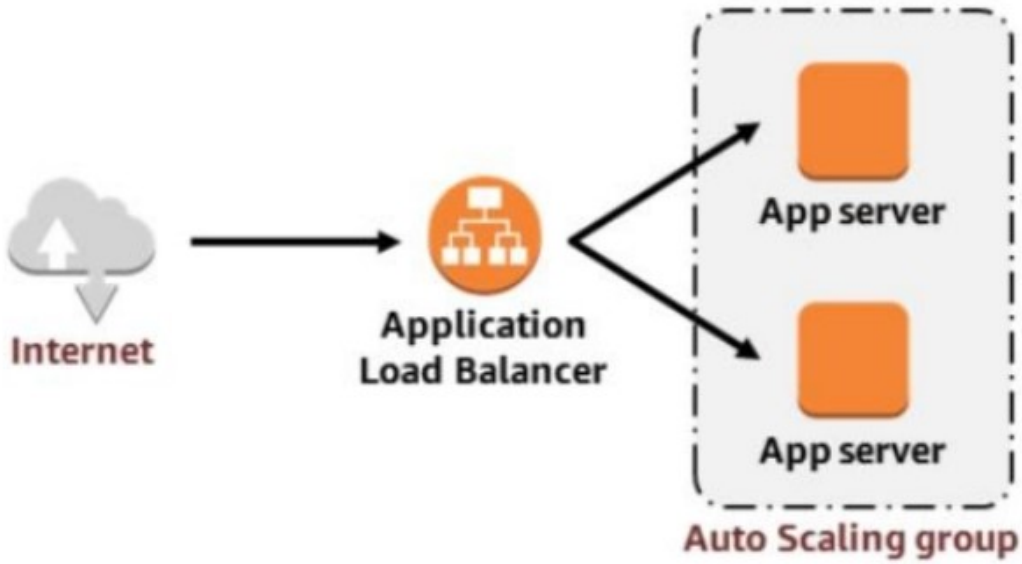
You will make this highly available!



Lab: Creating Highly Available Environment

To distribute requests across multiple servers, use:

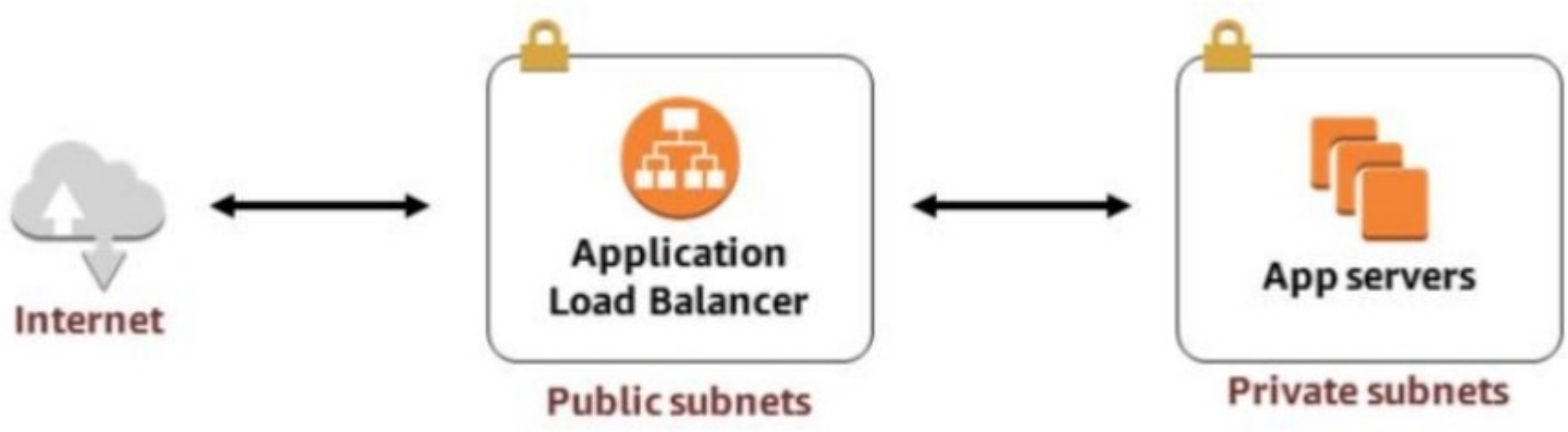
- Amazon EC2 Auto Scaling group
- Load Balancer



Lab: Creating Highly Available Environment

The Load Balancer is distributed across the public subnets

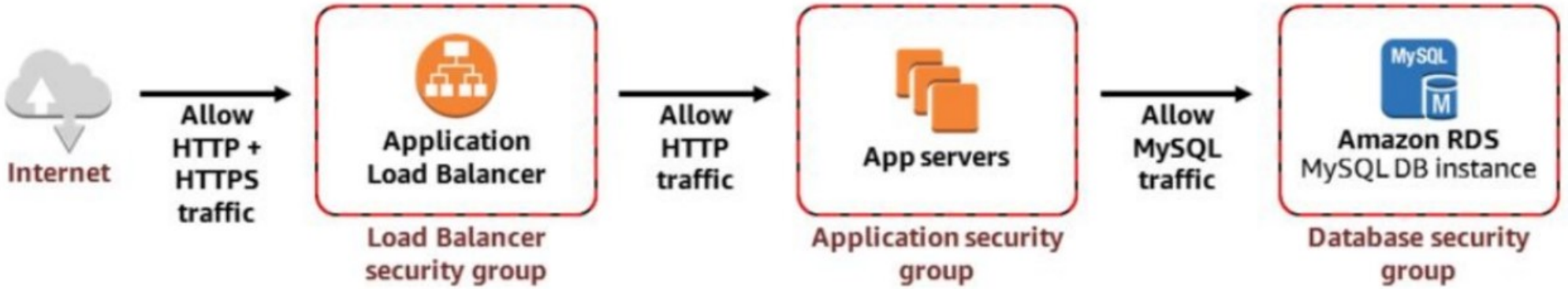
The application servers are in the private subnets



Lab: Creating Highly Available Environment

You will create a 3-tier architecture

Security groups provide additional security between each tier.



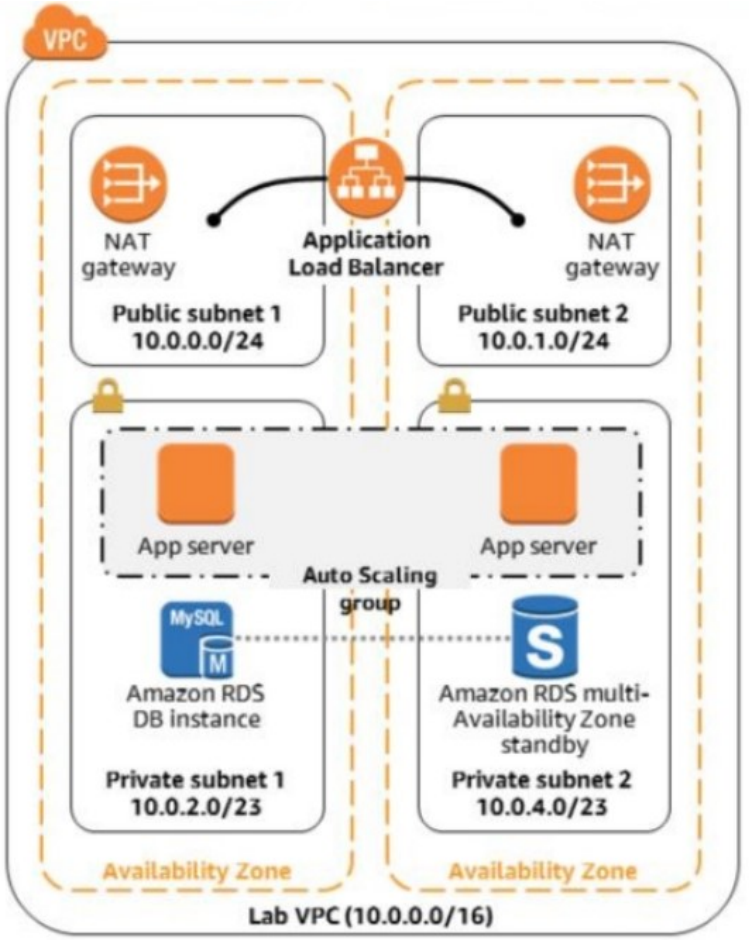
Lab: Creating Highly Available Environment

Final Configuration:

- Load balancer
- Multiple application servers
- multi-Availability one database
- NAT gateway in each Availability Zone

Scalable, reliable, high available

Duration : 40m



Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

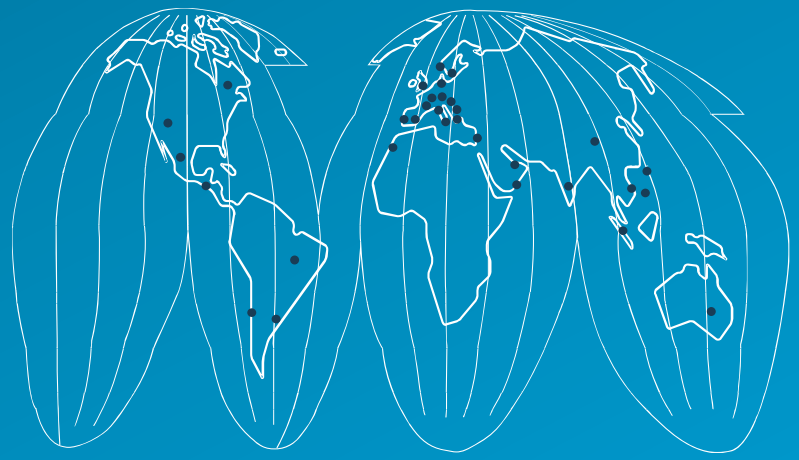
People matter, results count.

About Capgemini

With more than 145,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.5 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini



www.capgemini.com