

AWS Certified Solutions Architect Associate

By Stéphane Maarek



COURSE →



EXTRA PRACTICE EXAMS →

Disclaimer: These slides are copyrighted and strictly for personal use only

- This document is reserved for people enrolled into the [Ultimate AWS Solutions Architect Associate Course](#)
- Please do not share this document, it is intended for personal use and exam preparation only, thank you.
- If you've obtained these slides for free on a website that is not the course's website, please reach out to piracy@datacumulus.com. Thanks!
- Best of luck for the exam and happy learning!

AWS Certified Solutions Architect Associate Course

SAA-C03

Welcome! We're starting in 5 minutes



- We're going to prepare for the Solutions Architect exam - SAA-C03
- It's a challenging certification, so this course will be long and interesting
- Basic IT knowledge is necessary
- This course contains videos...
 - From the Cloud Practitioner, Developer and SysOps course - shared knowledge
 - Specific to the Solutions Architect exam - exciting ones on architecture!
- We will cover over 30 AWS services
- AWS / IT Beginners welcome! (but take your time, it's not a race)

My SAA-C03 certification: 96.1%

AWS Certified Solutions Architect - Associate	
Notice of Exam Results	
Candidate: Stephane MAAREK	Exam Date: Sep 02, 2022
Candidate ID: AWS [REDACTED]	Registration Number: [REDACTED]
Candidate Score: 961	Pass/Fail: PASS

About me

- I'm Stephane!
- Worked as in IT consultant and AWS Solutions Architect, Developer & SysOps
- Worked with AWS many years: built websites, apps, streaming platforms
- Veteran Instructor on AWS (Certifications, CloudFormation, Lambda, EC2...)
- You can find me on
 - GitHub: <https://github.com/simplesteph>
 - LinkedIn: <https://www.linkedin.com/in/stephanemaarek>
 - Medium: <https://medium.com/@stephane.maarek>
 - Twitter: <https://twitter.com/stephanemaarek>



★ 4.7 Instructor Rating

➲ 473,642 Reviews

➲ 1,553,489 Students

➲ 39 Courses

What's AWS?



- AWS (Amazon Web Services) is a Cloud Provider
- They provide you with servers and services that you can use on demand and scale easily
- AWS has revolutionized IT over time
- AWS powers some of the biggest websites in the world
 - Amazon.com
 - Netflix

What we'll learn in this course (and more!)



Amazon EC2



Amazon ECR



Amazon ECS



AWS Elastic Beanstalk



AWS Lambda



Auto Scaling



IAM



AWS KMS



Amazon S3



Amazon SES



Amazon RDS



Amazon Aurora



Amazon DynamoDB



Amazon ElastiCache



Amazon SQS



Amazon SNS



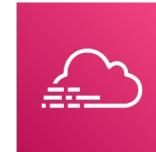
AWS Step Functions



Amazon CloudWatch



AWS CloudFormation



AWS CloudTrail



Amazon API Gateway



Elastic Load Balancing



Amazon CloudFront



Amazon Kinesis



Amazon Route 53

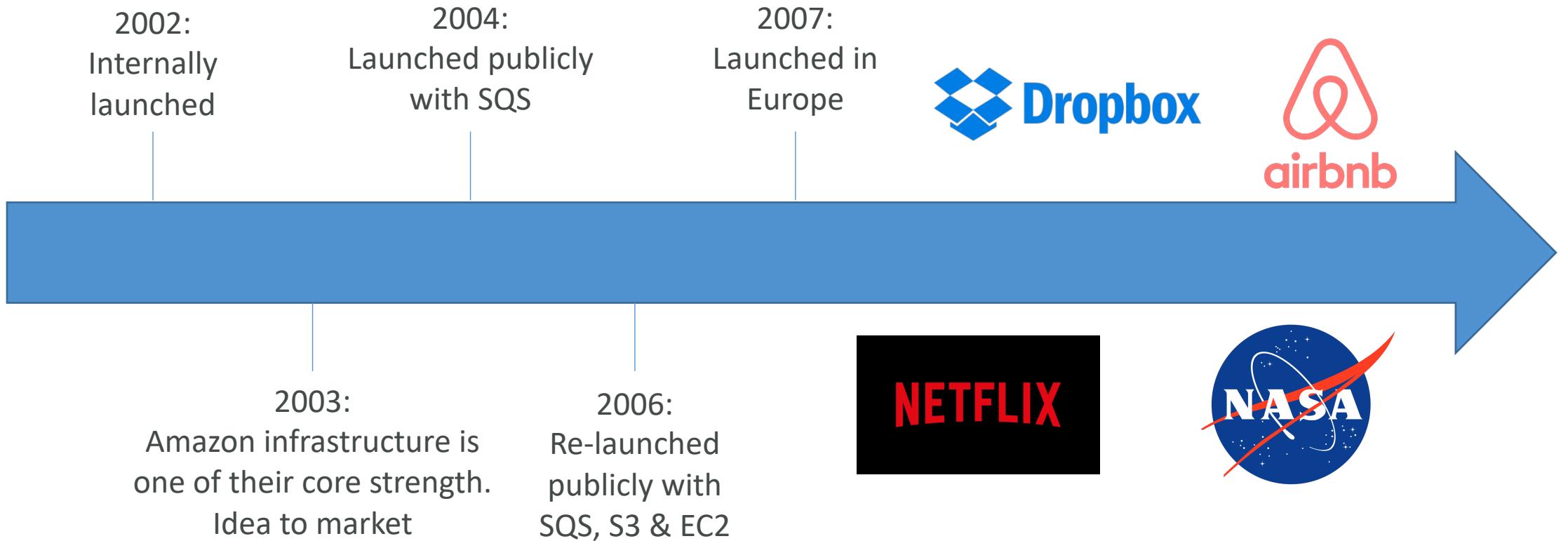
Navigating the AWS spaghetti bowl



Udemy Tips

Getting started with AWS

AWS Cloud History



AWS Cloud Number Facts

- In 2019, AWS had \$35.02 billion in annual revenue
- AWS accounts for 47% of the market in 2019 (Microsoft is 2nd with 22%)
- Pioneer and Leader of the AWS Cloud Market for the 9th consecutive year
- Over 1,000,000 active users

Figure 1. Magic Quadrant for Cloud Infrastructure as a Service, Worldwide



Gartner Magic Quadrant

AWS Cloud Use Cases

- AWS enables you to build sophisticated, scalable applications
- Applicable to a diverse set of industries
- Use cases include
 - Enterprise IT, Backup & Storage, Big Data analytics
 - Website hosting, Mobile & Social Apps
 - Gaming



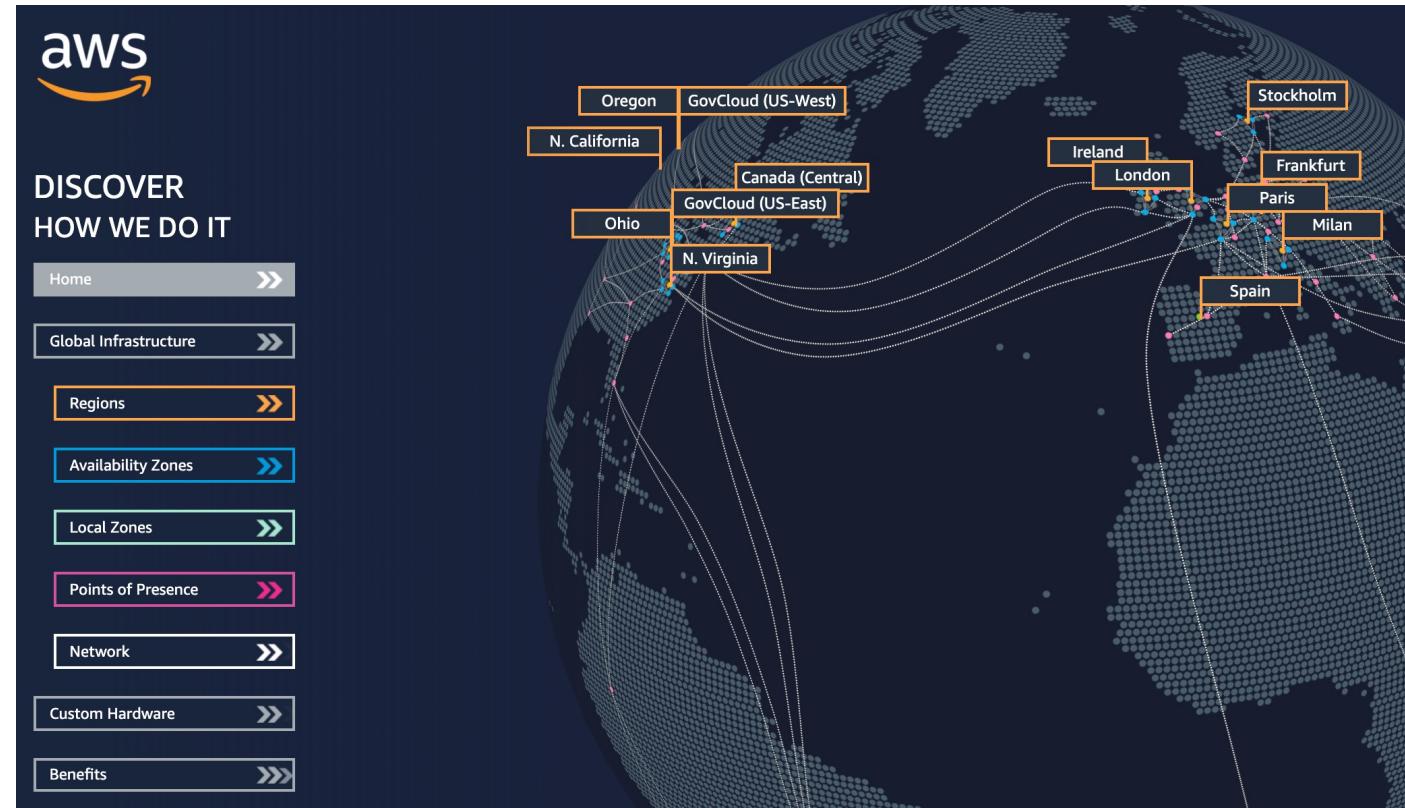
21ST
CENTURY
FOX

ACTIVISION



AWS Global Infrastructure

- AWS Regions
- AWS Availability Zones
- AWS Data Centers
- AWS Edge Locations / Points of Presence
- <https://infrastructure.aws/>



AWS Regions

- AWS has **Regions** all around the world
- Names can be us-east-1, eu-west-3...
- A region is a **cluster of data centers**
- Most AWS services are **region-scoped**



<https://aws.amazon.com/about-aws/global-infrastructure/>

US East (N. Virginia) us-east-1

US East (Ohio) us-east-2

US West (N. California) us-west-1

US West (Oregon) us-west-2

Africa (Cape Town) af-south-1

Asia Pacific (Hong Kong) ap-east-1

Asia Pacific (Mumbai) ap-south-1

Asia Pacific (Seoul) ap-northeast-2

Asia Pacific (Singapore) ap-southeast-1

Asia Pacific (Sydney) ap-southeast-2

Asia Pacific (Tokyo) ap-northeast-1

Canada (Central) ca-central-1

Europe (Frankfurt) eu-central-1

Europe (Ireland) eu-west-1

Europe (London) eu-west-2

Europe (Paris) eu-west-3

Europe (Stockholm) eu-north-1

Middle East (Bahrain) me-south-1

South America (São Paulo) sa-east-1

How to choose an AWS Region?

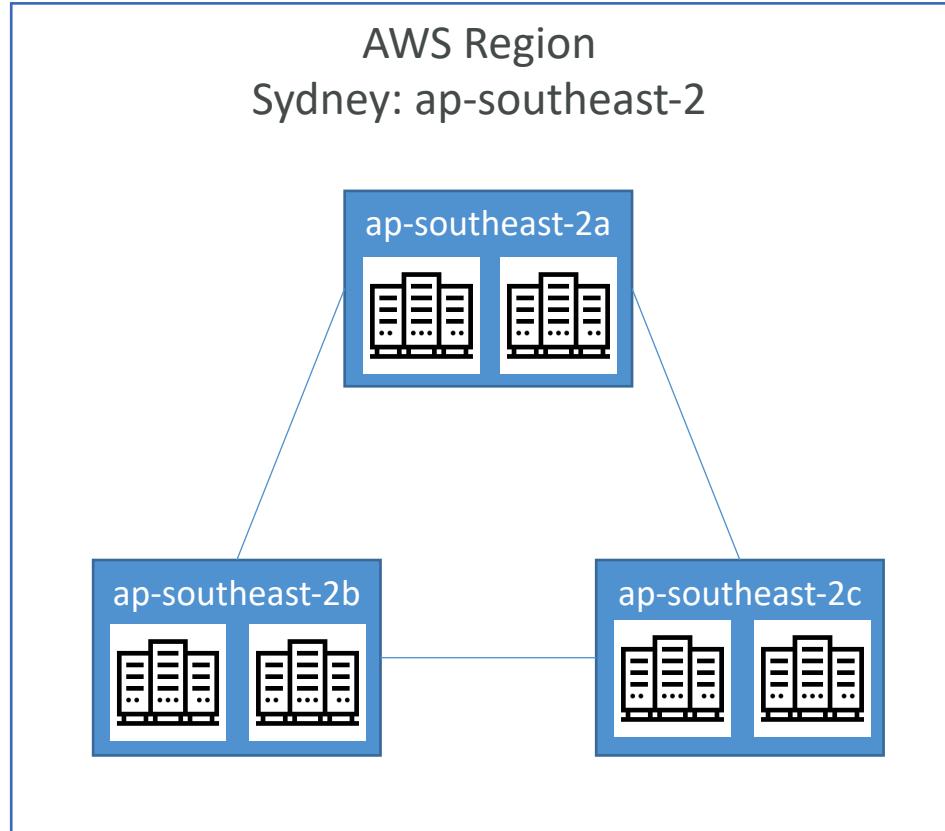
If you need to launch a new application,
where should you do it?



- **Compliance** with data governance and legal requirements: data never leaves a region without your explicit permission
- **Proximity** to customers: reduced latency
- **Available services** within a Region: new services and new features aren't available in every Region
- **Pricing**: pricing varies region to region and is transparent in the service pricing page

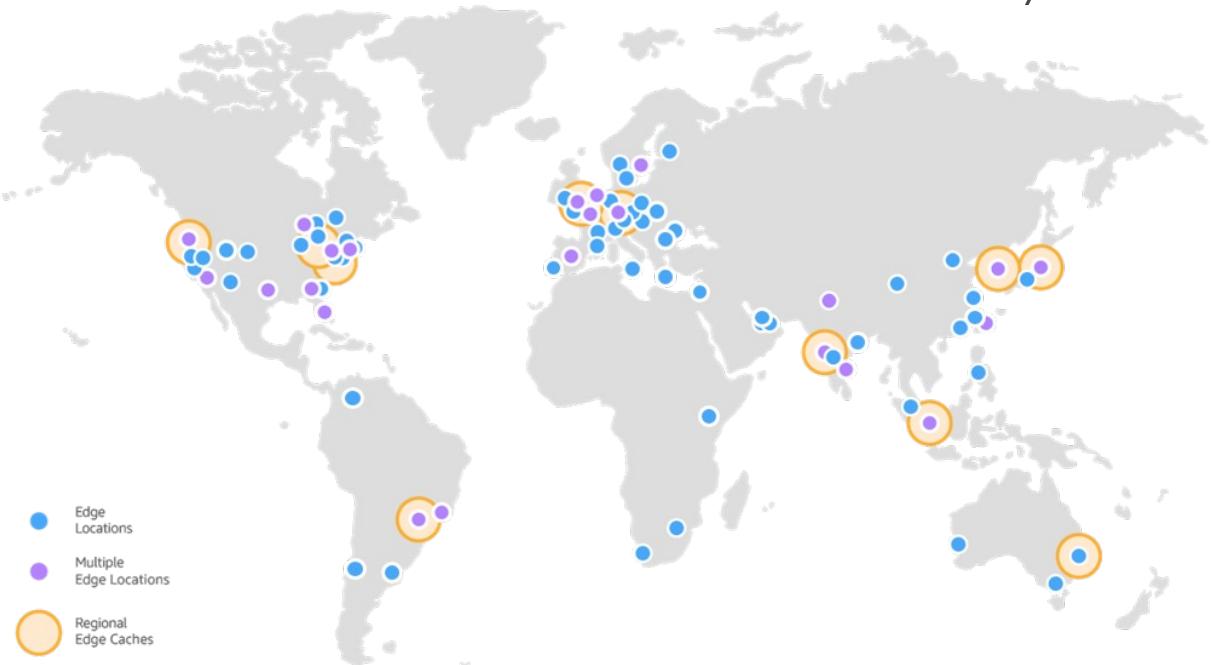
AWS Availability Zones

- Each region has many availability zones (usually 3, min is 3, max is 6). Example:
 - ap-southeast-2a
 - ap-southeast-2b
 - ap-southeast-2c
- Each availability zone (AZ) is one or more discrete data centers with redundant power, networking, and connectivity
- They're separate from each other, so that they're isolated from disasters
- They're connected with high bandwidth, ultra-low latency networking



AWS Points of Presence (Edge Locations)

- Amazon has 400+ Points of Presence (400+ Edge Locations & 10+ Regional Caches) in 90+ cities across 40+ countries
- Content is delivered to end users with lower latency

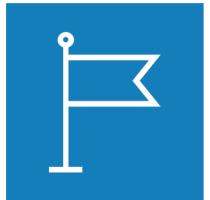


<https://aws.amazon.com/cloudfront/features/>

Tour of the AWS Console



- AWS has Global Services:
 - Identity and Access Management (IAM)
 - Route 53 (DNS service)
 - CloudFront (Content Delivery Network)
 - WAF (Web Application Firewall)
- Most AWS services are Region-scoped:
 - Amazon EC2 (Infrastructure as a Service)
 - Elastic Beanstalk (Platform as a Service)
 - Lambda (Function as a Service)
 - Rekognition (Software as a Service)
- Region Table: <https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services>



IAM Section

IAM: Users & Groups



- IAM = Identity and Access Management, **Global** service
- Root account created by default, shouldn't be used or shared
- **Users** are people within your organization, and can be grouped
- **Groups** only contain users, not other groups
- Users don't have to belong to a group, and user can belong to multiple groups



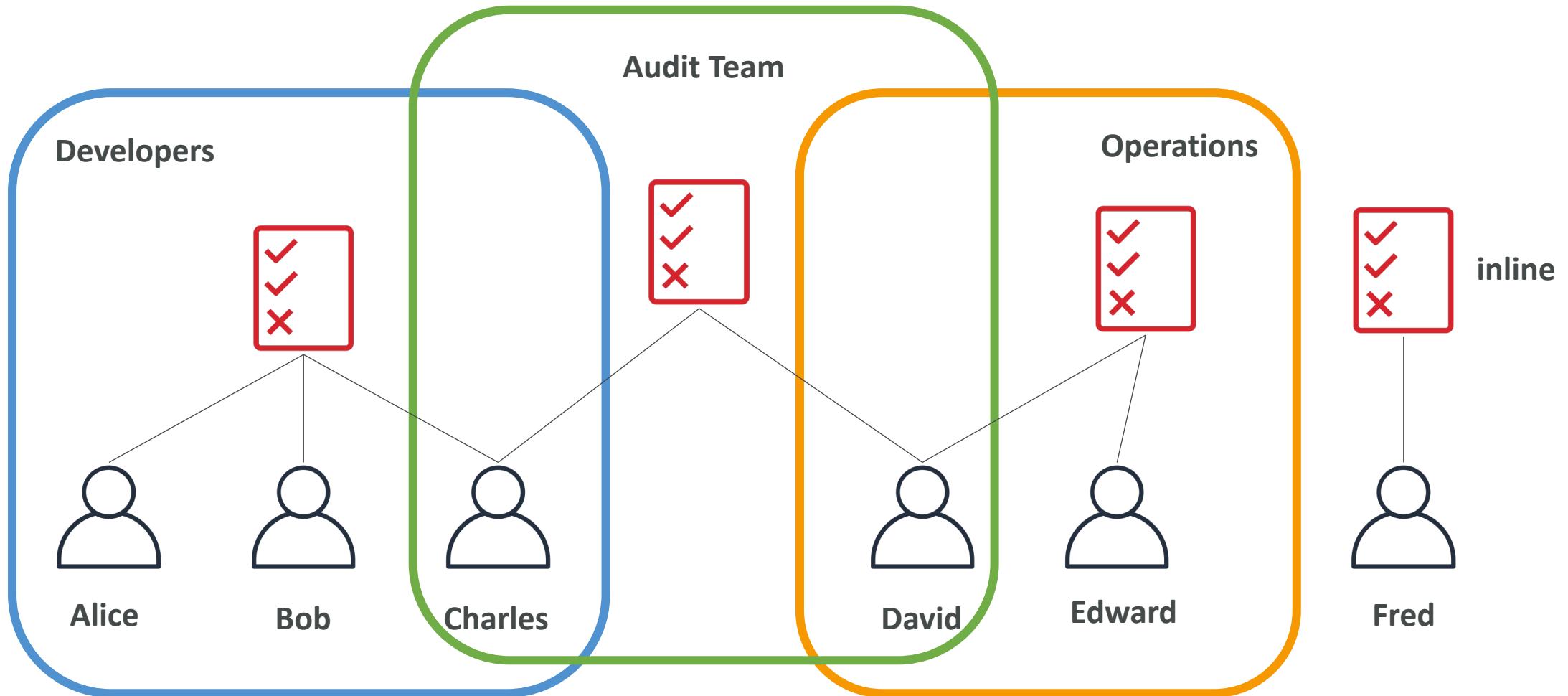
IAM: Permissions

- Users or Groups can be assigned JSON documents called policies
- These policies define the permissions of the users
- In AWS you apply the **least privilege principle**: don't give more permissions than a user needs

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "ec2:Describe*",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "elasticloadbalancing:Describe*",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "cloudwatch>ListMetrics",  
                "cloudwatch:GetMetricStatistics",  
                "cloudwatch:Describe"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```



IAM Policies inheritance



IAM Policies Structure

- Consists of
 - **Version:** policy language version, always include “2012-10-17”
 - **Id:** an identifier for the policy (optional)
 - **Statement:** one or more individual statements (required)
- Statements consists of
 - **Sid:** an identifier for the statement (optional)
 - **Effect:** whether the statement allows or denies access (Allow, Deny)
 - **Principal:** account/user/role to which this policy applied to
 - **Action:** list of actions this policy allows or denies
 - **Resource:** list of resources to which the actions applied to
 - **Condition:** conditions for when this policy is in effect (optional)

```
{  
  "Version": "2012-10-17",  
  "Id": "S3-Account-Permissions",  
  "Statement": [  
    {  
      "Sid": "1",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": ["arn:aws:iam::123456789012:root"]  
      },  
      "Action": [  
        "s3:GetObject",  
        "s3:PutObject"  
      ],  
      "Resource": ["arn:aws:s3:::mybucket/*"]  
    }  
  ]  
}
```

IAM – Password Policy

- Strong passwords = higher security for your account
- In AWS, you can setup a password policy:
 - Set a minimum password length
 - Require specific character types:
 - including uppercase letters
 - lowercase letters
 - numbers
 - non-alphanumeric characters
 - Allow all IAM users to change their own passwords
 - Require users to change their password after some time (password expiration)
 - Prevent password re-use

Multi Factor Authentication - MFA



- Users have access to your account and can possibly change configurations or delete resources in your AWS account
- You want to protect your Root Accounts and IAM users
- MFA = password you know + security device you own



- Main benefit of MFA:
if a password is stolen or hacked, the account is not compromised

MFA devices options in AWS

Virtual MFA device



Google Authenticator
(phone only)

Support for multiple tokens on a single device.



Authy
(multi-device)

Universal 2nd Factor (U2F) Security Key

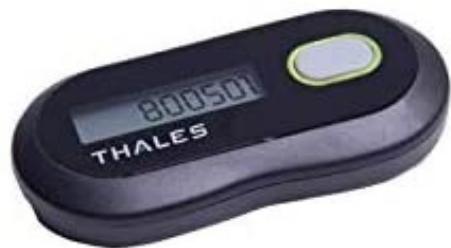


YubiKey by Yubico (3rd party)

Support for multiple root and IAM users using a single security key

MFA devices options in AWS

Hardware Key Fob MFA Device



Provided by Gemalto (3rd party)

Hardware Key Fob MFA Device for AWS GovCloud (US)



Provided by SurePassID (3rd party)

How can users access AWS ?



- To access AWS, you have three options:
 - AWS Management Console (protected by password + MFA)
 - AWS Command Line Interface (CLI): protected by access keys
 - AWS Software Developer Kit (SDK) - for code: protected by access keys
- Access Keys are generated through the AWS Console
- Users manage their own access keys
- Access Keys are secret, just like a password. Don't share them
- Access Key ID ~ = username
- Secret Access Key ~ = password

Example (Fake) Access Keys

Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. [Learn more](#)

[Create access key](#)

Access key ID	Created	Last used	Status	
AKIASK4E37PV4TU3RD6C	2020-05-25 15:13 UTC+0100	N/A	Active	Make inactive X

- Access key ID: AKIASK4E37PV4983d6C
- Secret Access Key: AZPN3z0jWozWCndljhB0Uh8239aIbzBzO5fqkZq
- Remember: don't share your access keys

What's the AWS CLI?

- A tool that enables you to interact with AWS services using commands in your command-line shell
- Direct access to the public APIs of AWS services
- You can develop scripts to manage your resources
- It's open-source <https://github.com/aws/aws-cli>
- Alternative to using AWS Management Console

```
→ ~ aws s3 cp myfile.txt s3://ccp-mybucket/myfile.txt
upload: ./myfile.txt to s3://ccp-mybucket/myfile.txt
→ ~ aws s3 ls s3://ccp-mybucket
2021-05-14 03:22:52          0 myfile.txt
→ ~ █
```

What's the AWS SDK?



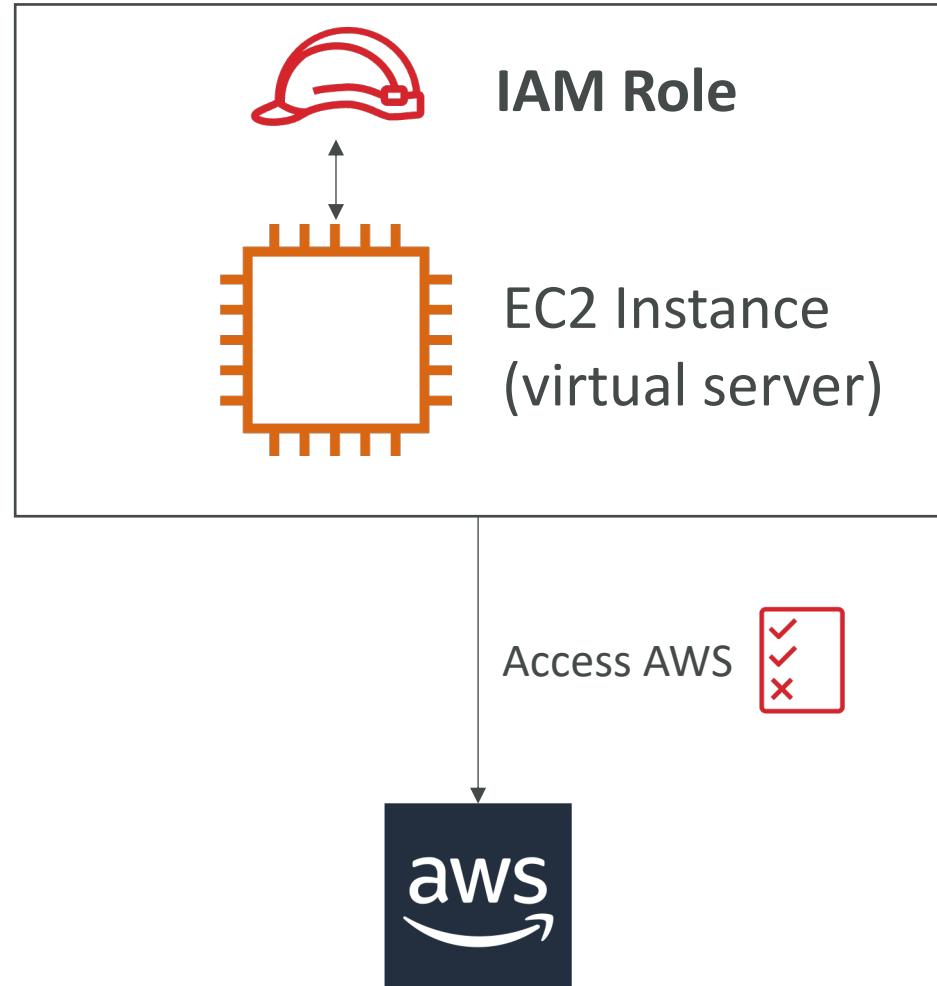
- AWS Software Development Kit (AWS SDK)
- Language-specific APIs (set of libraries)
- Enables you to access and manage AWS services programmatically
- Embedded within your application
- Supports
 - SDKs (JavaScript, Python, PHP, .NET, Ruby, Java, Go, Node.js, C++)
 - Mobile SDKs (Android, iOS, ...)
 - IoT Device SDKs (Embedded C, Arduino, ...)
- Example: AWS CLI is built on AWS SDK for Python



Your Application

IAM Roles for Services

- Some AWS service will need to perform actions on your behalf
- To do so, we will assign permissions to AWS services with IAM Roles
- Common roles:
 - EC2 Instance Roles
 - Lambda Function Roles
 - Roles for CloudFormation



IAM Security Tools

- **IAM Credentials Report (account-level)**
 - a report that lists all your account's users and the status of their various credentials
- **IAM Access Advisor (user-level)**
 - Access advisor shows the service permissions granted to a user and when those services were last accessed.
 - You can use this information to revise your policies.

IAM Guidelines & Best Practices



- Don't use the root account except for AWS account setup
- One physical user = One AWS user
- Assign users to groups and assign permissions to groups
- Create a strong password policy
- Use and enforce the use of Multi Factor Authentication (MFA)
- Create and use Roles for giving permissions to AWS services
- Use Access Keys for Programmatic Access (CLI / SDK)
- Audit permissions of your account using IAM Credentials Report & IAM Access Advisor
- Never share IAM users & Access Keys

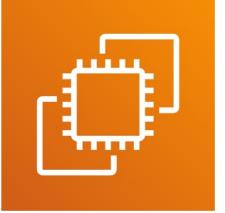
IAM Section – Summary



- **Users:** mapped to a physical user; has a password for AWS Console
- **Groups:** contains users only
- **Policies:** JSON document that outlines permissions for users or groups
- **Roles:** for EC2 instances or AWS services
- **Security:** MFA + Password Policy
- **AWS CLI:** manage your AWS services using the command-line
- **AWS SDK:** manage your AWS services using a programming language
- **Access Keys:** access AWS using the CLI or SDK
- **Audit:** IAM Credential Reports & IAM Access Advisor

EC2 Basics

Amazon EC2



- EC2 is one of the most popular of AWS' offering
- EC2 = Elastic Compute Cloud = Infrastructure as a Service
- It mainly consists in the capability of :
 - Renting virtual machines (EC2)
 - Storing data on virtual drives (EBS)
 - Distributing load across machines (ELB)
 - Scaling the services using an auto-scaling group (ASG)
- Knowing EC2 is fundamental to understand how the Cloud works

EC2 sizing & configuration options

- Operating System (OS): Linux, Windows or Mac OS
- How much compute power & cores (CPU)
- How much random-access memory (RAM)
- How much storage space:
 - Network-attached (EBS & EFS)
 - hardware (EC2 Instance Store)
- Network card: speed of the card, Public IP address
- Firewall rules: **security group**
- Bootstrap script (configure at first launch): EC2 User Data

EC2 User Data

- It is possible to bootstrap our instances using an [EC2 User data](#) script.
- [bootstrapping](#) means launching commands when a machine starts
- That script is [only run once](#) at the instance [first start](#)
- EC2 user data is used to automate boot tasks such as:
 - Installing updates
 - Installing software
 - Downloading common files from the internet
 - Anything you can think of
- The EC2 User Data Script runs with the root user

Hands-On: Launching an EC2 Instance running Linux

- We'll be launching our first virtual server using the AWS Console
- We'll get a first high-level approach to the various parameters
- We'll see that our web server is launched using EC2 user data
- We'll learn how to start / stop / terminate our instance.

EC2 Instance Types - Overview

- You can use different types of EC2 instances that are optimised for different use cases (<https://aws.amazon.com/ec2/instance-types/>)
- AWS has the following naming convention:

m5.2xlarge

- m: instance class
- 5: generation (AWS improves them over time)
- 2xlarge: size within the instance class

General Purpose

Compute Optimized

Memory Optimized

Accelerated Computing

Storage Optimized

Instance Features

Measuring Instance Performance

EC2 Instance Types – General Purpose

- Great for a diversity of workloads such as web servers or code repositories
- Balance between:
 - Compute
 - Memory
 - Networking
- In the course, we will be using the t2.micro which is a General Purpose EC2 instance

General Purpose

General purpose instances provide a balance of compute, memory and networking resources, and can be used for a variety of diverse workloads. These instances are ideal for applications that use these resources in equal proportions such as web servers and code repositories.

Mac	T4g	T3	T3a	T2	M6g	M5	M5a	M5n	M5zn	M4	A1
-----	-----	----	-----	----	-----	----	-----	-----	------	----	----

* this list will evolve over time, please check the AWS website for the latest information

EC2 Instance Types – Compute Optimized

- Great for compute-intensive tasks that require high performance processors:
 - Batch processing workloads
 - Media transcoding
 - High performance web servers
 - High performance computing (HPC)
 - Scientific modeling & machine learning
 - Dedicated gaming servers

Compute Optimized

Compute Optimized Instances are ideal for compute bound applications that benefit from high performance processors. Instances belonging to this family are well suited for batch processing workloads, media transcoding, high performance web servers, high performance computing (HPC), scientific modeling, dedicated gaming servers and ad server engines, machine learning inference and other compute intensive applications.

C6g C6gn C5 C5a C5n C4

* this list will evolve over time, please check the AWS website for the latest information

EC2 Instance Types – Memory Optimized

- Fast performance for workloads that process large data sets in memory
- Use cases:
 - High performance, relational/non-relational databases
 - Distributed web scale cache stores
 - In-memory databases optimized for BI (business intelligence)
 - Applications performing real-time processing of big unstructured data

Memory Optimized

Memory optimized instances are designed to deliver fast performance for workloads that process large data sets in memory.

R6g

R5

R5a

R5b

R5n

R4

X1e

X1

High Memory

z1d

* this list will evolve over time, please check the AWS website for the latest information

EC2 Instance Types – Storage Optimized

- Great for storage-intensive tasks that require high, sequential read and write access to large data sets on local storage
- Use cases:
 - High frequency online transaction processing (OLTP) systems
 - Relational & NoSQL databases
 - Cache for in-memory databases (for example, Redis)
 - Data warehousing applications
 - Distributed file systems

Storage Optimized

Storage optimized instances are designed for workloads that require high, sequential read and write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications.

I3 I3en D2 D3 D3en H1

* this list will evolve over time, please check the AWS website for the latest information

EC2 Instance Types: example

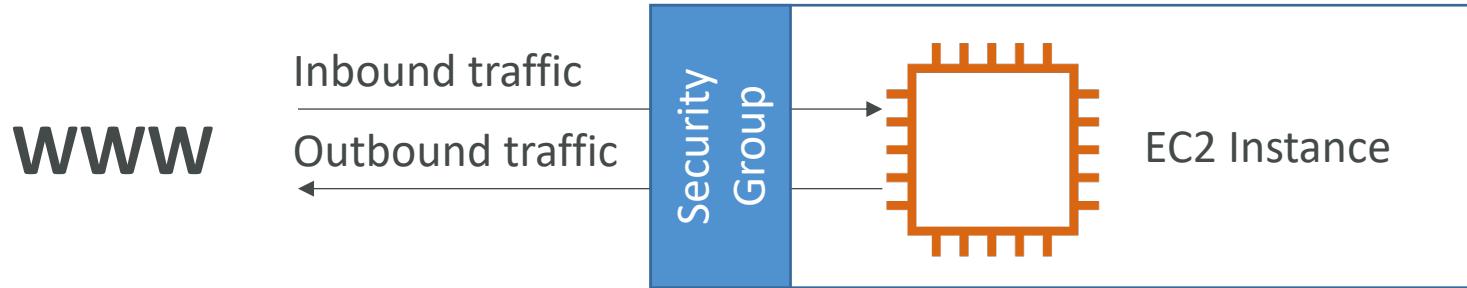
Instance	vCPU	Mem (GiB)	Storage	Network Performance	EBS Bandwidth (Mbps)
t2.micro	1	1	EBS-Only	Low to Moderate	
t2.xlarge	4	16	EBS-Only	Moderate	
c5d.4xlarge	16	32	1 x 400 NVMe SSD	Up to 10 Gbps	4,750
r5.16xlarge	64	512	EBS Only	20 Gbps	13,600
m5.8xlarge	32	128	EBS Only	10 Gbps	6,800

t2.micro is part of the AWS free tier (up to 750 hours per month)

Great website: <https://instances.vantage.sh>

Introduction to Security Groups

- Security Groups are the fundamental of network security in AWS
- They control how traffic is allowed into or out of our EC2 Instances.



- Security groups only contain **allow** rules
- Security groups rules can reference by IP or by security group

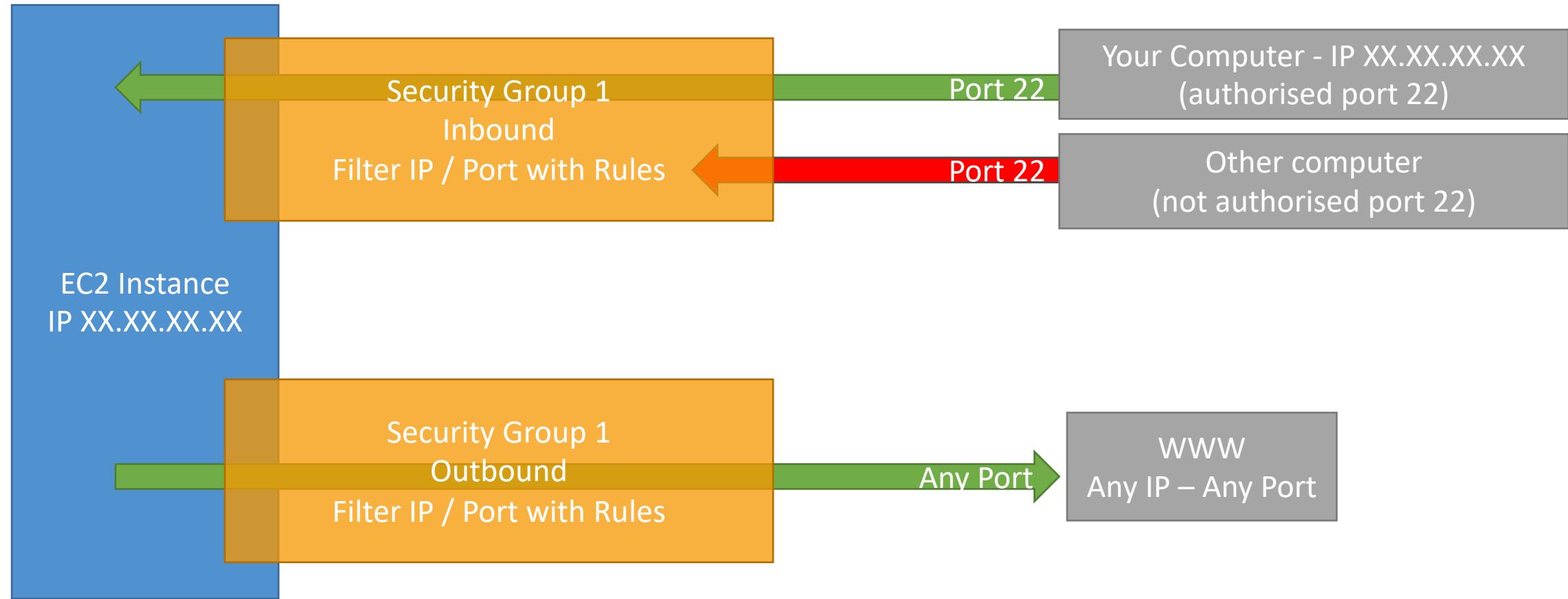
Security Groups

Deeper Dive

- Security groups are acting as a “firewall” on EC2 instances
- They regulate:
 - Access to Ports
 - Authorised IP ranges – IPv4 and IPv6
 - Control of inbound network (from other to the instance)
 - Control of outbound network (from the instance to other)

Type i	Protocol i	Port Range i	Source i	Description i
HTTP	TCP	80	0.0.0.0/0	test http page
SSH	TCP	22	122.149.196.85/32	
Custom TCP Rule	TCP	4567	0.0.0.0/0	java app

Security Groups Diagram



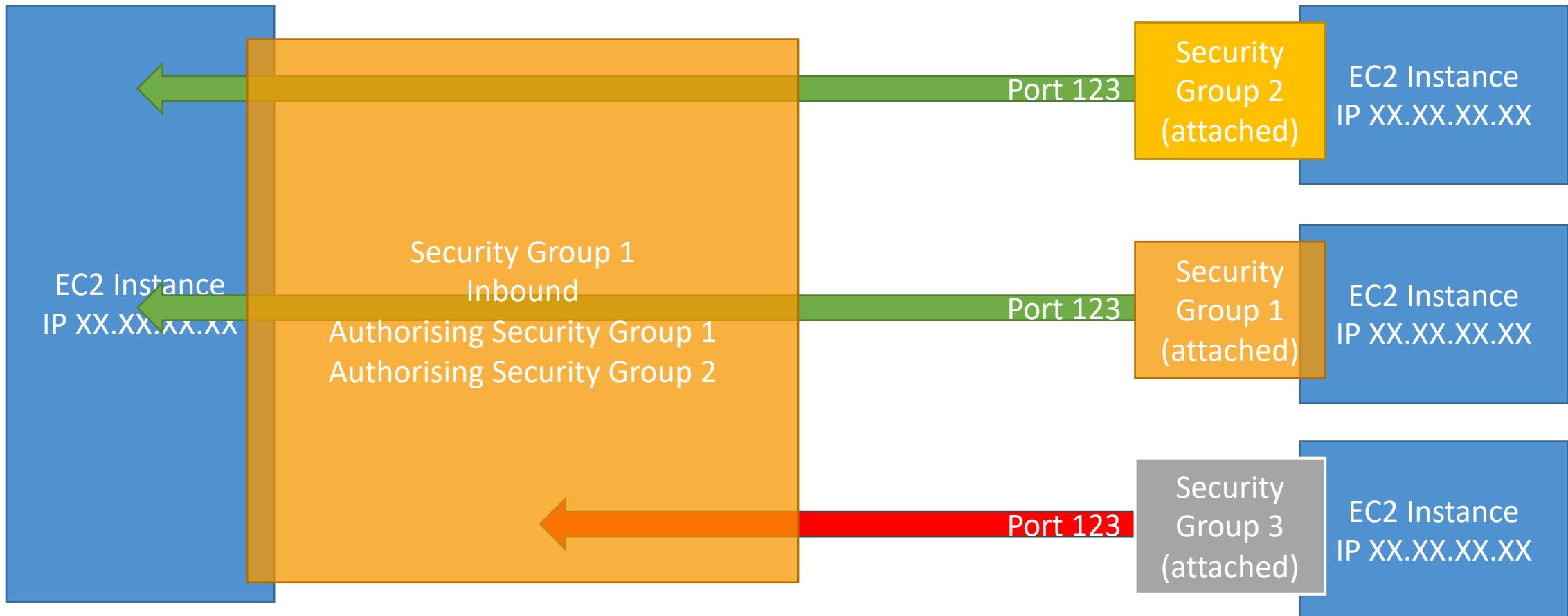
Security Groups

Good to know

- Can be attached to multiple instances
- Locked down to a region / VPC combination
- Does live “outside” the EC2 – if traffic is blocked the EC2 instance won’t see it
- It’s good to maintain one separate security group for SSH access
- If your application is not accessible (time out), then it’s a security group issue
- If your application gives a “connection refused” error, then it’s an application error or it’s not launched
- All inbound traffic is **blocked** by default
- All outbound traffic is **authorised** by default

Referencing other security groups

Diagram



Classic Ports to know

- 22 = SSH (Secure Shell) - log into a Linux instance
- 21 = FTP (File Transfer Protocol) – upload files into a file share
- 22 = SFTP (Secure File Transfer Protocol) – upload files using SSH
- 80 = HTTP – access unsecured websites
- 443 = HTTPS – access secured websites
- 3389 = RDP (Remote Desktop Protocol) – log into a Windows instance

SSH Summary Table

	SSH	Putty	EC2 Instance Connect
Mac	✓		✓
Linux	✓		✓
Windows < 10		✓	✓
Windows >= 10	✓	✓	✓

Which Lectures to watch

- Mac / Linux:
 - SSH on Mac/Linux lecture
- Windows:
 - Putty Lecture
 - If Windows 10: SSH on Windows 10 lecture
- All:
 - EC2 Instance Connect lecture

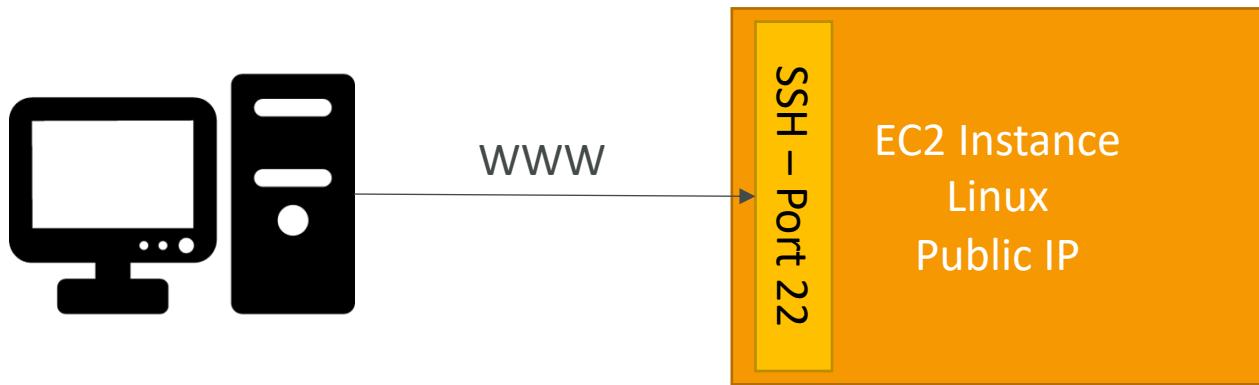
SSH troubleshooting

- Students have the most problems with SSH
- If things don't work...
 1. Re-watch the lecture. You may have missed something
 2. Read the troubleshooting guide
 3. Try EC2 Instance Connect
- If one method works (SSH, Putty or EC2 Instance Connect) you're good
- If no method works, that's okay, the course won't use SSH much

How to SSH into your EC2 Instance

Linux / Mac OS X

- We'll learn how to SSH into your EC2 instance using Linux / Mac
- SSH is one of the most important function. It allows you to control a remote machine, all using the command line.

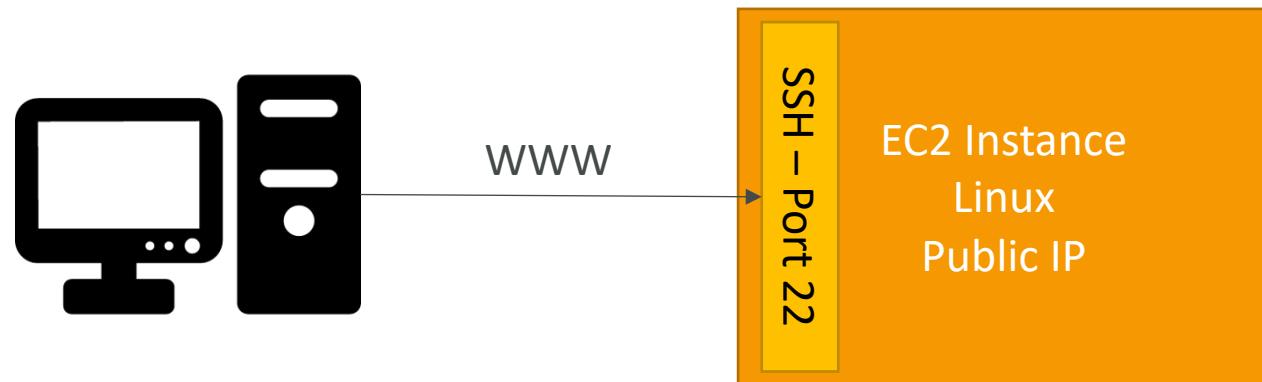


- We will see how we can configure OpenSSH `~/.ssh/config` to facilitate the SSH into our EC2 instances

How to SSH into your EC2 Instance

Windows

- We'll learn how to SSH into your EC2 instance using [Windows](#)
- SSH is one of the most important function. It allows you to control a remote machine, all using the command line.



- We will configure all the required parameters necessary for doing SSH on Windows using the free tool [Putty](#).

EC2 Instance Connect

- Connect to your EC2 instance within your browser
- No need to use your key file that was downloaded
- The “magic” is that a temporary key is uploaded onto EC2 by AWS
- Works only out-of-the-box with Amazon Linux 2
- Need to make sure the port 22 is still opened!

EC2 Instances Purchasing Options

- On-Demand Instances – short workload, predictable pricing, pay by second
- Reserved (1 & 3 years)
 - Reserved Instances – long workloads
 - Convertible Reserved Instances – long workloads with flexible instances
- Savings Plans (1 & 3 years) – commitment to an amount of usage, long workload
- Spot Instances – short workloads, cheap, can lose instances (less reliable)
- Dedicated Hosts – book an entire physical server, control instance placement
- Dedicated Instances – no other customers will share your hardware
- Capacity Reservations – reserve capacity in a specific AZ for any duration

EC2 On Demand

- Pay for what you use:
 - Linux or Windows - billing per second, after the first minute
 - All other operating systems - billing per hour
- Has the highest cost but no upfront payment
- No long-term commitment
- Recommended for **short-term** and **un-interrupted workloads**, where you can't predict how the application will behave

EC2 Reserved Instances

- Up to 72% discount compared to On-demand
- You reserve a specific instance attributes (Instance Type, Region, Tenancy, OS)
- Reservation Period – 1 year (+discount) or 3 years (+++discount)
- Payment Options – No Upfront (+), Partial Upfront (++) , All Upfront (+++)
- Reserved Instance's Scope – Regional or Zonal (reserve capacity in an AZ)
- Recommended for steady-state usage applications (think database)
- You can buy and sell in the Reserved Instance Marketplace
- Convertible Reserved Instance
 - Can change the EC2 instance type, instance family, OS, scope and tenancy
 - Up to 66% discount

Note: the % discounts are different from the video as AWS change them over time – the exact numbers are not needed for the exam. This is just for illustrative purposes ☺

EC2 Savings Plans

- Get a discount based on long-term usage (up to 72% - same as RIs)
- Commit to a certain type of usage (\$10/hour for 1 or 3 years)
- Usage beyond EC2 Savings Plans is billed at the On-Demand price
- Locked to a specific instance family & AWS region (e.g., M5 in us-east-1)
- Flexible across:
 - Instance Size (e.g., m5.xlarge, m5.2xlarge)
 - OS (e.g., Linux, Windows)
 - Tenancy (Host, Dedicated, Default)



EC2 Spot Instances

- Can get a **discount of up to 90%** compared to On-demand
- Instances that you can “lose” at any point of time if your max price is less than the current spot price
- The **MOST cost-efficient** instances in AWS
- Useful for workloads that are resilient to failure
 - Batch jobs
 - Data analysis
 - Image processing
 - Any **distributed** workloads
 - Workloads with a flexible start and end time
- Not suitable for critical jobs or databases

EC2 Dedicated Hosts

- A physical server with EC2 instance capacity fully dedicated to your use
- Allows you address **compliance requirements** and **use your existing server-bound software licenses** (per-socket, per-core, pe—VM software licenses)
- Purchasing Options:
 - **On-demand** – pay per second for active Dedicated Host
 - **Reserved** - 1 or 3 years (No Upfront, Partial Upfront, All Upfront)
- The most expensive option
- Useful for software that have complicated licensing model (BYOL – Bring Your Own License)
- Or for companies that have strong regulatory or compliance needs

EC2 Dedicated Instances

- Instances run on hardware that's dedicated to you
- May share hardware with other instances in same account
- No control over instance placement (can move hardware after Stop / Start)

Characteristic	Dedicated Instances	Dedicated Hosts
Enables the use of dedicated physical servers	X	X
Per instance billing (subject to a \$2 per region fee)	X	
Per host billing		X
Visibility of sockets, cores, host ID		X
Affinity between a host and instance		X
Targeted instance placement		X
Automatic instance placement	X	X
Add capacity using an allocation request		X

EC2 Capacity Reservations

- Reserve On-Demand instances capacity in a specific AZ for any duration
- You always have access to EC2 capacity when you need it
- **No time commitment** (create/cancel anytime), **no billing discounts**
- Combine with Regional Reserved Instances and Savings Plans to benefit from billing discounts
- You're charged at On-Demand rate whether you run instances or not
- Suitable for short-term, uninterrupted workloads that needs to be in a specific AZ

Which purchasing option is right for me?



- **On demand:** coming and staying in resort whenever we like, we pay the full price
- **Reserved:** like planning ahead and if we plan to stay for a long time, we may get a good discount.
- **Savings Plans:** pay a certain amount per hour for certain period and stay in any room type (e.g., King, Suite, Sea View, ...)
- **Spot instances:** the hotel allows people to bid for the empty rooms and the highest bidder keeps the rooms. You can get kicked out at any time
- **Dedicated Hosts:** We book an entire building of the resort
- **Capacity Reservations:** you book a room for a period with full price even you don't stay in it

Price Comparison

Example – m4.large – us-east-1

Price Type	Price (per hour)
On-Demand	\$0.10
Spot Instance (Spot Price)	\$0.038 - \$0.039 (up to 61% off)
Reserved Instance (1 year)	\$0.062 (No Upfront) - \$0.058 (All Upfront)
Reserved Instance (3 years)	\$0.043 (No Upfront) - \$0.037 (All Upfront)
EC2 Savings Plan (1 year)	\$0.062 (No Upfront) - \$0.058 (All Upfront)
Reserved Convertible Instance (1 year)	\$0.071 (No Upfront) - \$0.066 (All Upfront)
Dedicated Host	On-Demand Price
Dedicated Host Reservation	Up to 70% off
Capacity Reservations	On-Demand Price

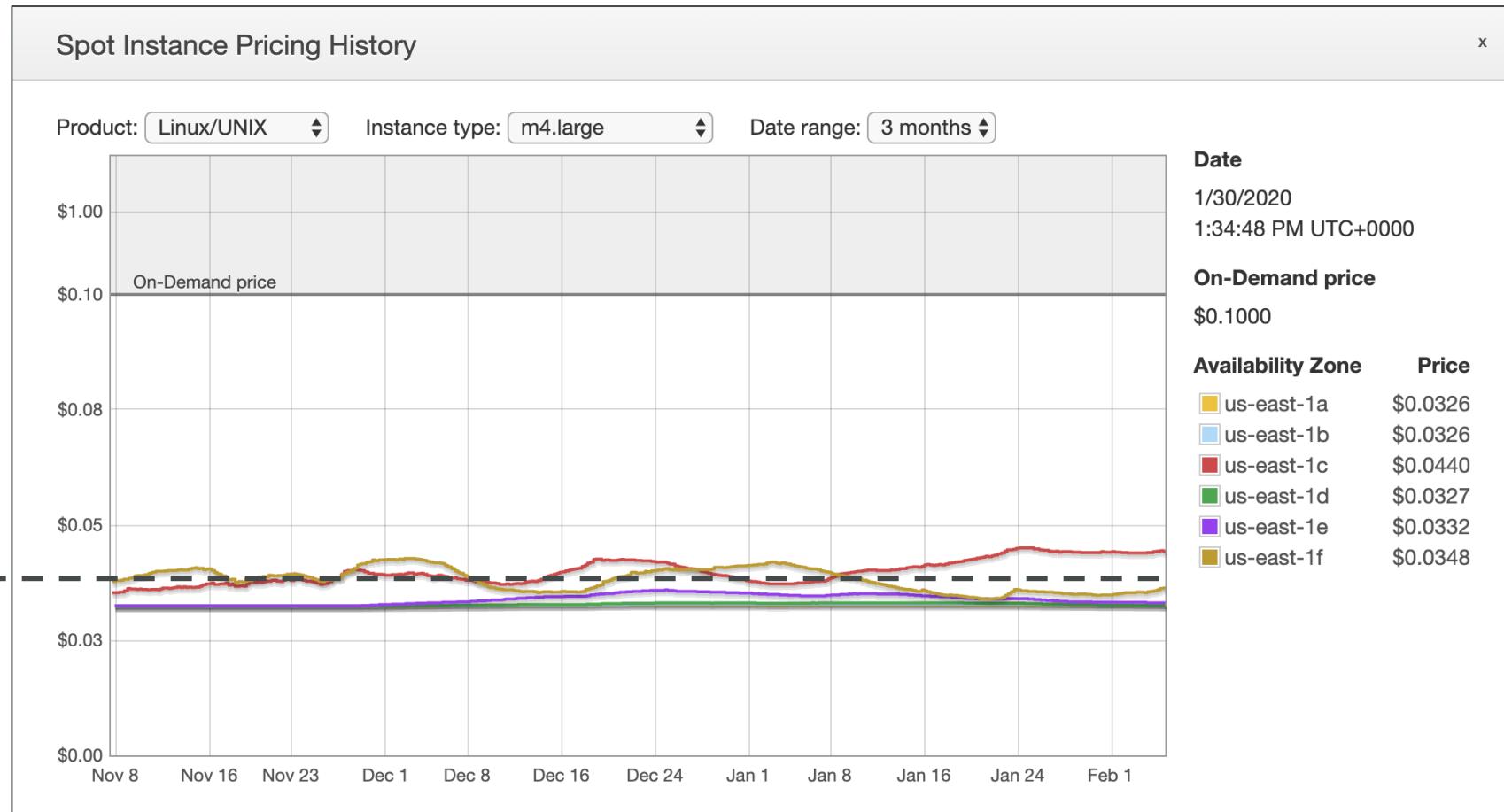


EC2 Spot Instance Requests

- Can get a discount of up to 90% compared to On-demand
- Define **max spot price** and get the instance while **current spot price < max**
 - The hourly spot price varies based on offer and capacity
 - If the current spot price > your max price you can choose to **stop** or **terminate** your instance with a 2 minutes grace period.
- Other strategy: **Spot Block**
 - “block” spot instance during a specified time frame (1 to 6 hours) without interruptions
 - In rare situations, the instance may be reclaimed
- Used for batch jobs, data analysis, or workloads that are resilient to failures.
- Not great for critical jobs or databases

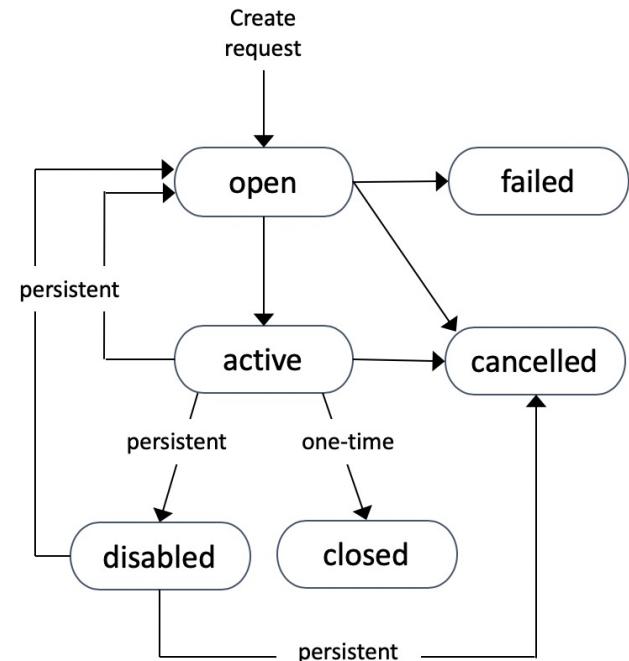
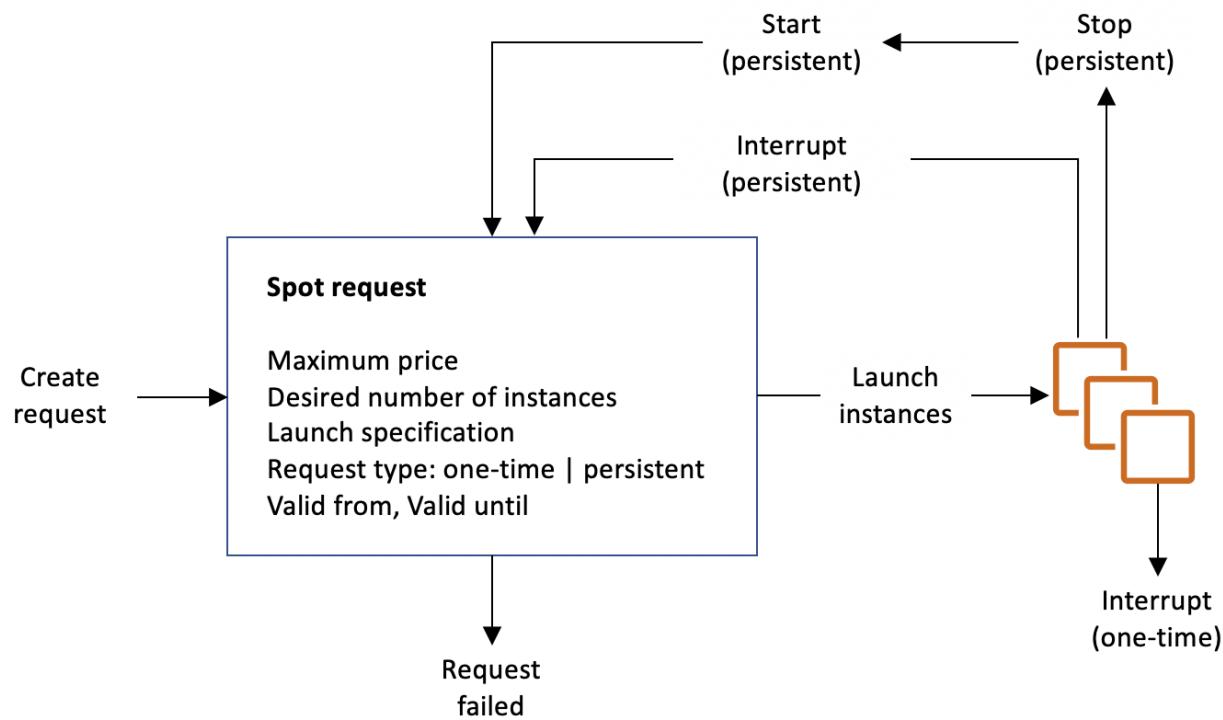
EC2 Spot Instances Pricing

User-defined max price



<https://console.aws.amazon.com/ec2sp/v1/spot/home?region=us-east-1#>

How to terminate Spot Instances?



You can only cancel Spot Instance requests that are **open, active, or disabled**.

Cancelling a Spot Request does not terminate instances

You must first cancel a Spot Request, and then terminate the associated Spot Instances

Spot Fleets

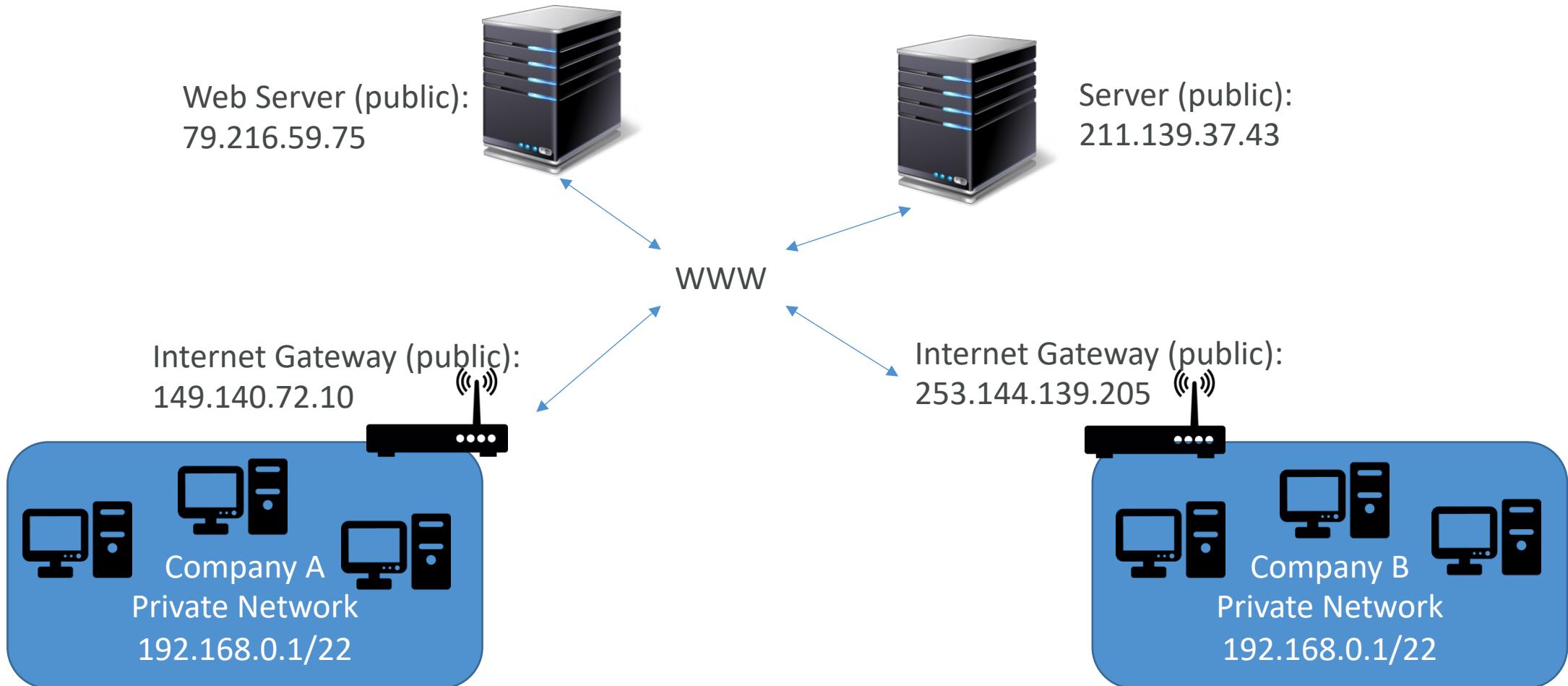
- Spot Fleets = set of Spot Instances + (optional) On-Demand Instances
- The Spot Fleet will try to meet the target capacity with price constraints
 - Define possible launch pools: instance type (m5.large), OS, Availability Zone
 - Can have multiple launch pools, so that the fleet can choose
 - Spot Fleet stops launching instances when reaching capacity or max cost
- Strategies to allocate Spot Instances:
 - **lowestPrice**: from the pool with the lowest price (cost optimization, short workload)
 - **diversified**: distributed across all pools (great for availability, long workloads)
 - **capacityOptimized**: pool with the optimal capacity for the number of instances
 - **priceCapacityOptimized (recommended)**: pools with highest capacity available, then select the pool with the lowest price (best choice for most workloads)
- Spot Fleets allow us to automatically request Spot Instances with the lowest price

EC2 – Associate

Private vs Public IP (IPv4)

- Networking has two sorts of IPs. IPv4 and IPv6:
 - IPv4: **1.160.10.240**
 - IPv6: **3ffe:1900:4545:3:200:f8ff:fe21:67cf**
- In this course, we will only be using IPv4.
- IPv4 is still the most common format used online.
- IPv6 is newer and solves problems for the Internet of Things (IoT).
- IPv4 allows for **3.7 billion** different addresses in the public space
- IPv4: [0-255].[0-255].[0-255].[0-255].

Private vs Public IP (IPv4) Example



Private vs Public IP (IPv4)

Fundamental Differences

- Public IP:
 - Public IP means the machine can be identified on the internet (WWW)
 - Must be unique across the whole web (not two machines can have the same public IP).
 - Can be geo-located easily
- Private IP:
 - Private IP means the machine can only be identified on a private network only
 - The IP must be unique across the private network
 - BUT two different private networks (two companies) can have the same IPs.
 - Machines connect to WWW using a NAT + internet gateway (a proxy)
 - Only a specified range of IPs can be used as private IP

Elastic IPs

- When you stop and then start an EC2 instance, it can change its public IP.
- If you need to have a fixed public IP for your instance, you need an Elastic IP
- An Elastic IP is a public IPv4 IP you own as long as you don't delete it
- You can attach it to one instance at a time

Elastic IP

- With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.
- You can only have 5 Elastic IP in your account (you can ask AWS to increase that).
- Overall, try to avoid using Elastic IP:
 - They often reflect poor architectural decisions
 - Instead, use a random public IP and register a DNS name to it
 - Or, as we'll see later, use a Load Balancer and don't use a public IP

Private vs Public IP (IPv4)

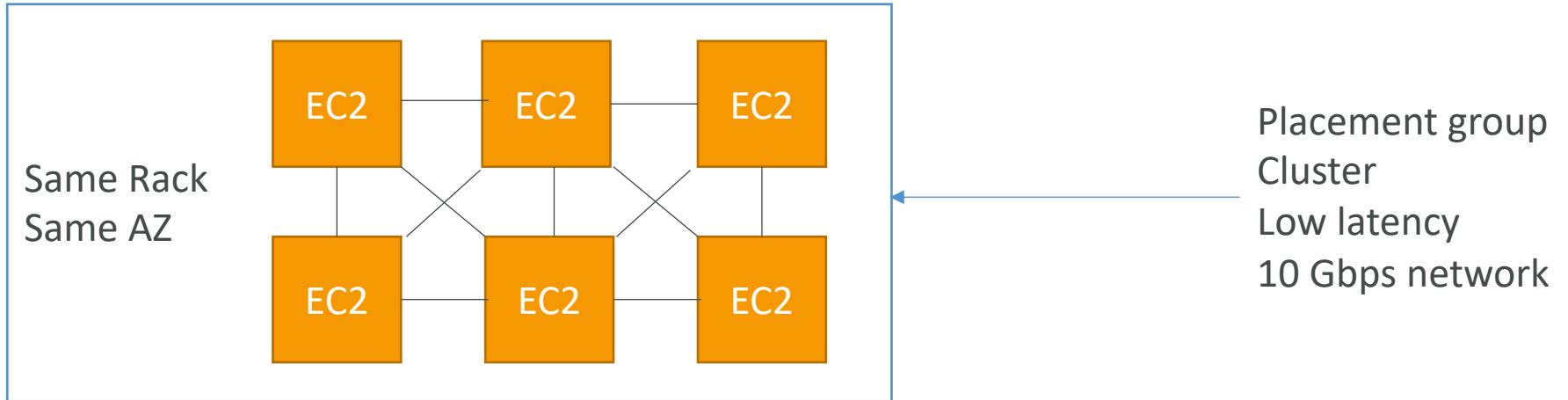
In AWS EC2 – Hands On

- By default, your EC2 machine comes with:
 - A private IP for the internal AWS Network
 - A public IP for the WWW.
- When we are doing SSH into our EC2 machines:
 - We can't use a private IP, because we are not in the same network
 - We can only use the public IP.
- If your machine is stopped and then started,
the public IP can change

Placement Groups

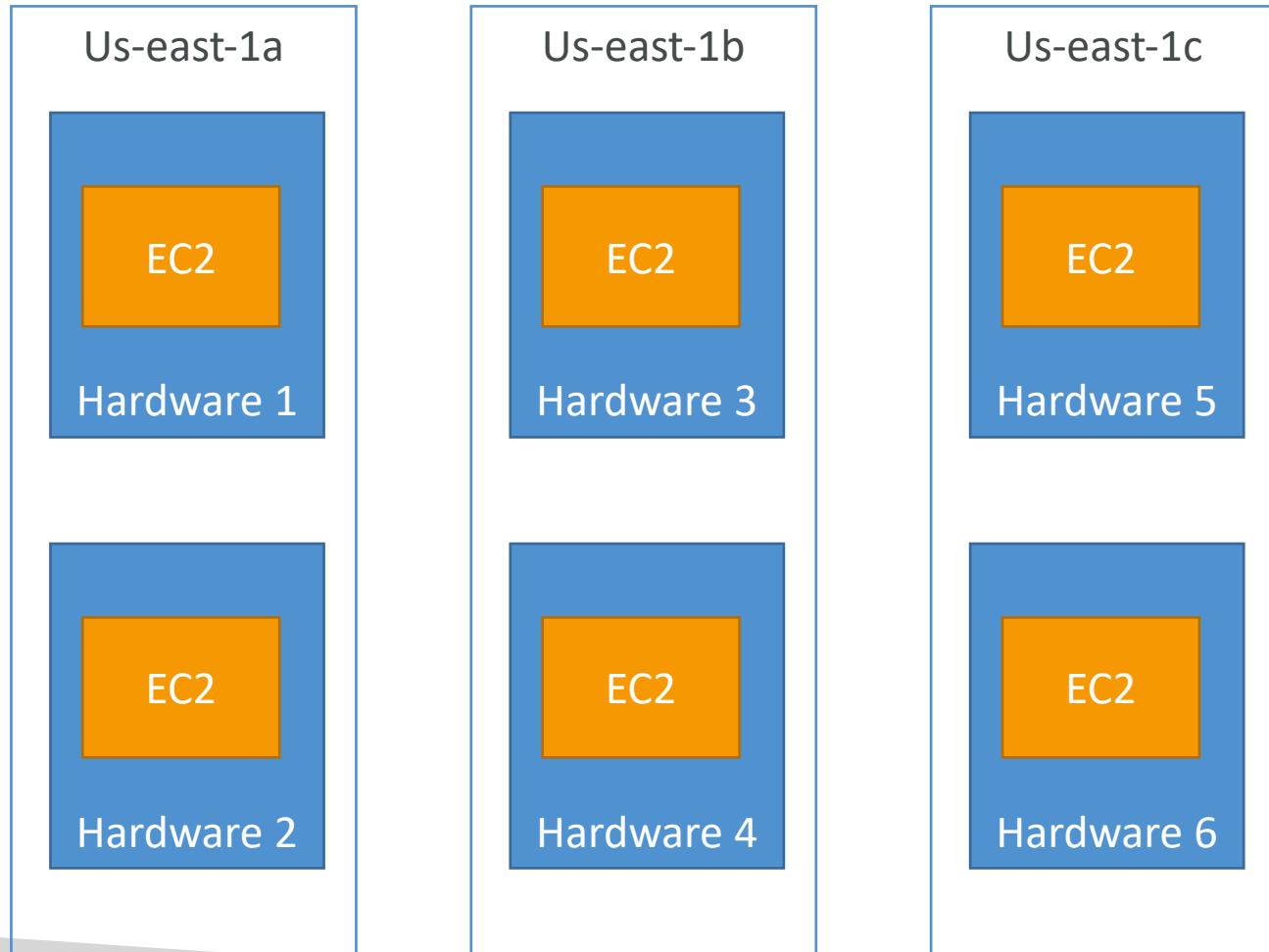
- Sometimes you want control over the EC2 Instance placement strategy
- That strategy can be defined using placement groups
- When you create a placement group, you specify one of the following strategies for the group:
 - *Cluster*—clusters instances into a low-latency group in a single Availability Zone
 - *Spread*—spreads instances across underlying hardware (max 7 instances per group per AZ)
 - *Partition*—spreads instances across many different partitions (which rely on different sets of racks) within an AZ. Scales to 100s of EC2 instances per group (Hadoop, Cassandra, Kafka)

Placement Groups Cluster



- Pros: Great network (10 Gbps bandwidth between instances with Enhanced Networking enabled - recommended)
- Cons: If the rack fails, all instances fail at the same time
- Use case:
 - Big Data job that needs to complete fast
 - Application that needs extremely low latency and high network throughput

Placement Groups Spread



- Pros:

- Can span across Availability Zones (AZ)
- Reduced risk of simultaneous failure
- EC2 Instances are on different physical hardware

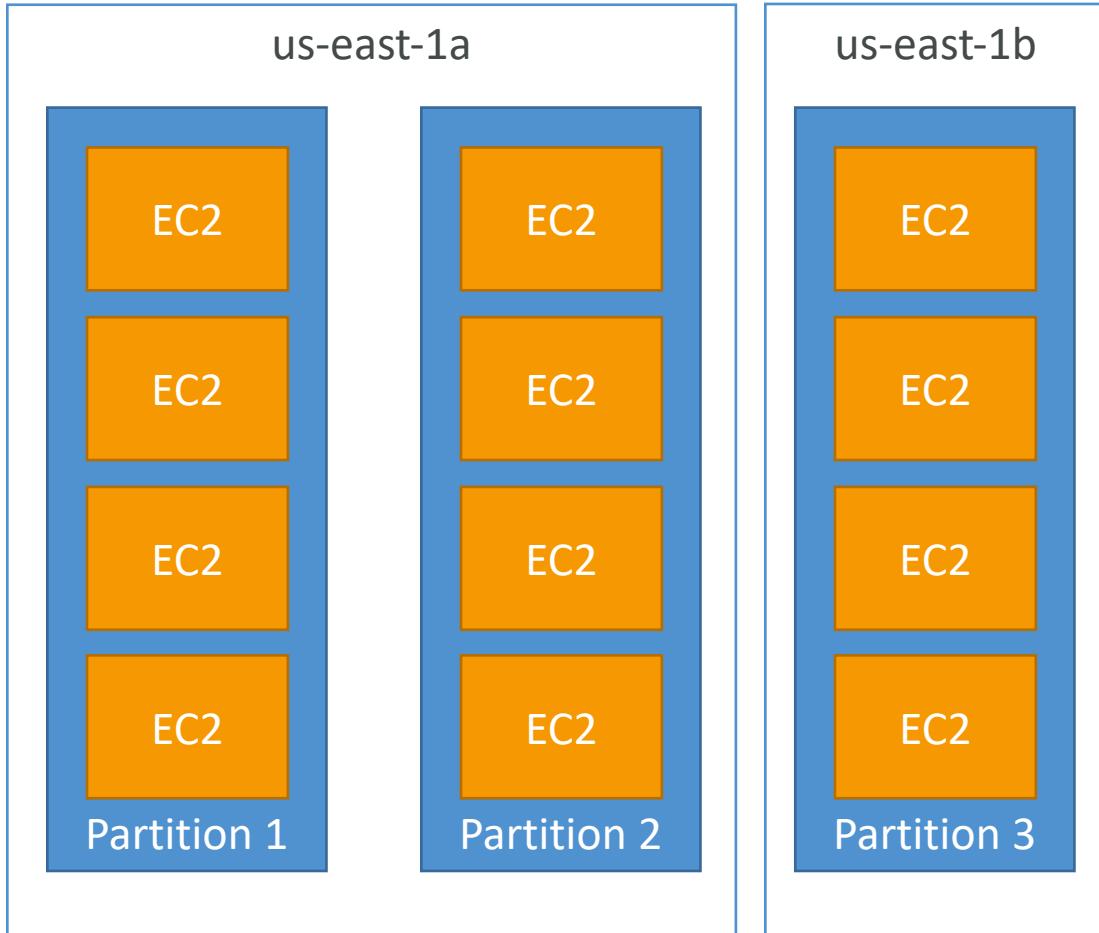
- Cons:

- Limited to 7 instances per AZ per placement group

- Use case:

- Application that needs to maximize high availability
- Critical Applications where each instance must be isolated from failure from each other

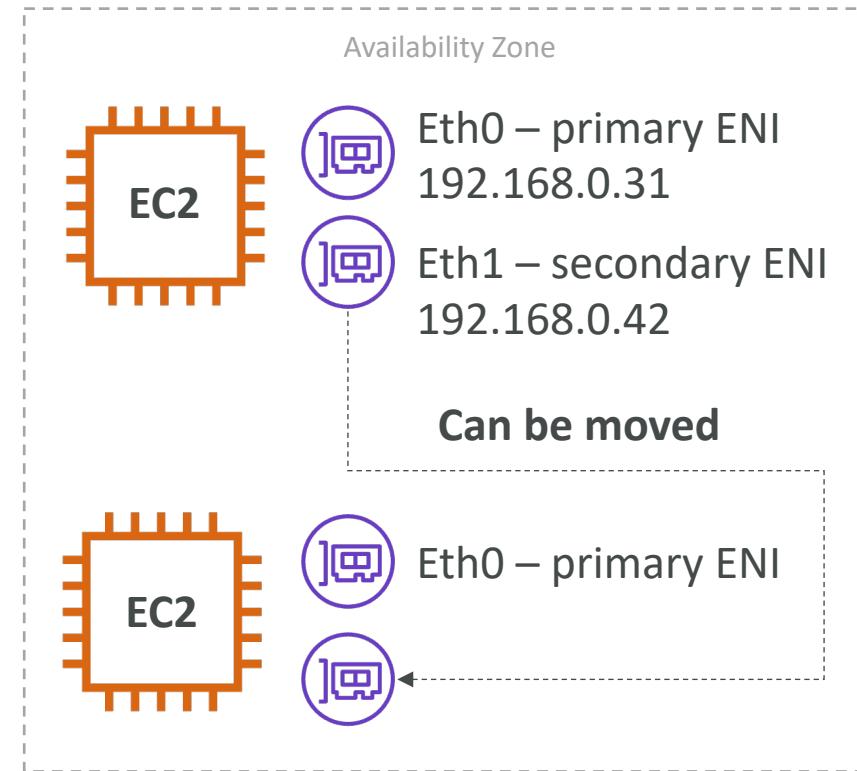
Placements Groups Partition



- Up to 7 partitions per AZ
- Can span across multiple AZs in the same region
- Up to 100s of EC2 instances
- The instances in a partition do not share racks with the instances in the other partitions
- A partition failure can affect many EC2 but won't affect other partitions
- EC2 instances get access to the partition information as metadata
- Use cases: HDFS, HBase, Cassandra, Kafka

Elastic Network Interfaces (ENI)

- Logical component in a VPC that represents a virtual network card
- The ENI can have the following attributes:
 - Primary private IPv4, one or more secondary IPv4
 - One Elastic IP (IPv4) per private IPv4
 - One Public IPv4
 - One or more security groups
 - A MAC address
- You can create ENI independently and attach them on the fly (move them) on EC2 instances for failover
- Bound to a specific availability zone (AZ)

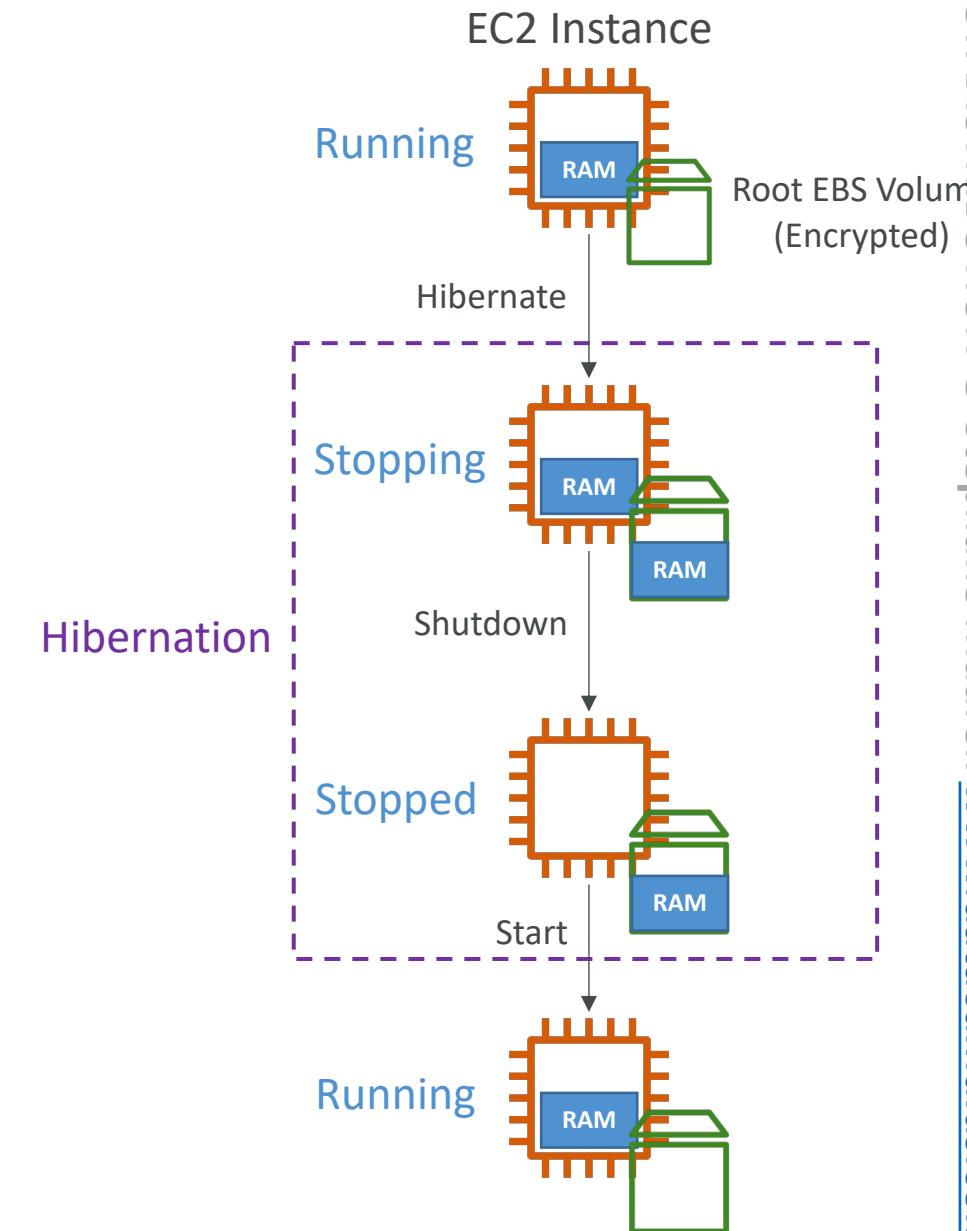


EC2 Hibernate

- We know we can stop, terminate instances
 - Stop – the data on disk (EBS) is kept intact in the next start
 - Terminate – any EBS volumes (root) also set-up to be destroyed is lost
- On start, the following happens:
 - First start: the OS boots & the EC2 User Data script is run
 - Following starts: the OS boots up
 - Then your application starts, caches get warmed up, and that can take time!

EC2 Hibernate

- Introducing EC2 Hibernate:
 - The in-memory (RAM) state is preserved
 - The instance boot is much faster! (the OS is not stopped / restarted)
 - Under the hood: the RAM state is written to a file in the root EBS volume
 - The root EBS volume must be encrypted
- Use cases:
 - Long-running processing
 - Saving the RAM state
 - Services that take time to initialize

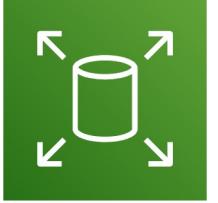


EC2 Hibernate – Good to know

- Supported Instance Families – C3, C4, C5, I3, M3, M4, R3, R4, T2, T3, ...
- Instance RAM Size – must be less than 150 GB.
- Instance Size – not supported for bare metal instances.
- AMI – Amazon Linux 2, Linux AMI, Ubuntu, RHEL, CentOS & Windows...
- Root Volume – must be EBS, encrypted, not instance store, and large
- Available for On-Demand, Reserved and Spot Instances
- An instance can NOT be hibernated more than 60 days

EC2 Instance Storage Section

What's an EBS Volume?

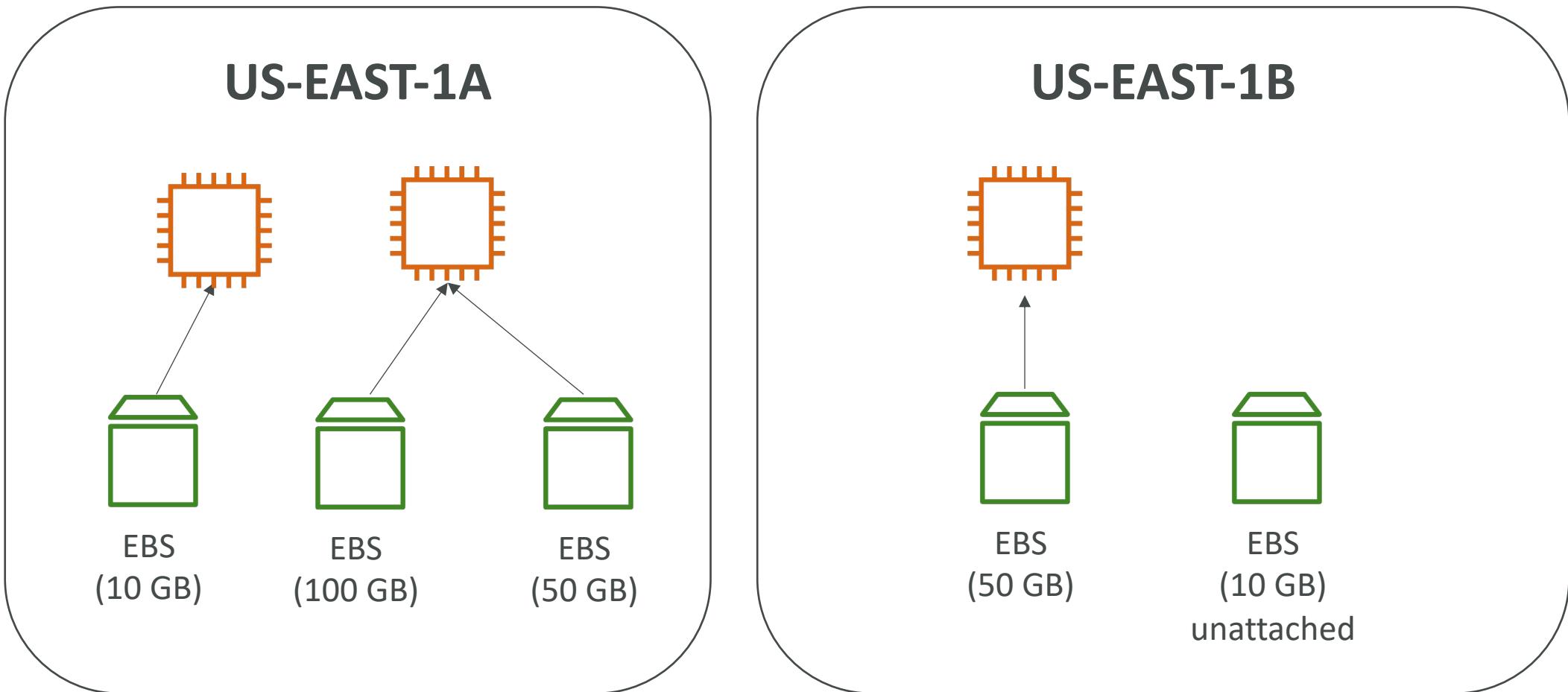


- An **EBS (Elastic Block Store) Volume** is a **network** drive you can attach to your instances while they run
- It allows your instances to persist data, even after their termination
- They can only be mounted to one instance at a time (at the CCP level)
- They are bound to a specific availability zone
- Analogy: Think of them as a “network USB stick”
- Free tier: 30 GB of free EBS storage of type General Purpose (SSD) or Magnetic per month

EBS Volume

- It's a network drive (i.e. not a physical drive)
 - It uses the network to communicate the instance, which means there might be a bit of latency
 - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
 - An EBS Volume in us-east-1a cannot be attached to us-east-1b
 - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
 - You get billed for all the provisioned capacity
 - You can increase the capacity of the drive over time

EBS Volume - Example



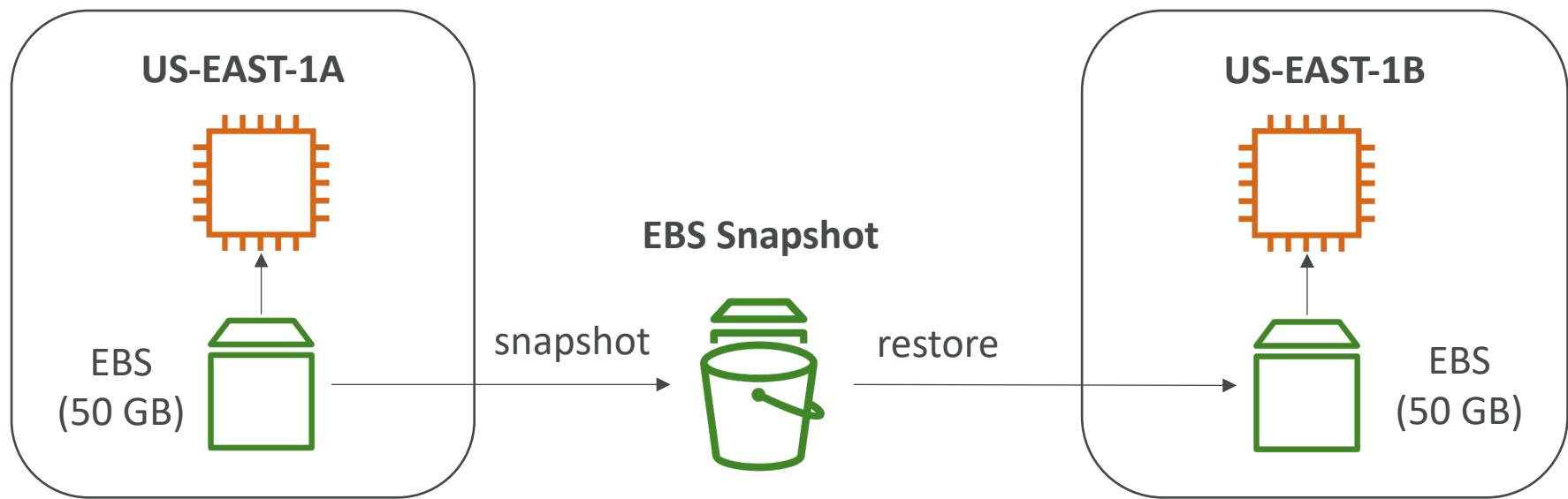
EBS – Delete on Termination attribute

Volume Type <small>i</small>	Device <small>i</small>	Snapshot <small>i</small>	Size (GiB) <small>i</small>	Volume Type <small>i</small>	IOPS <small>i</small>	Throughput (MB/s) <small>i</small>	Delete on Termination <small>i</small>	Encryption <small>i</small>
Root	/dev/xvda	snap-09f18f682fd23a1b1	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted ▾
EBS	/dev/sdb	Search (case-insensit	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input type="checkbox"/>	Not Encrypted ▾ X
Add New Volume								

- Controls the EBS behaviour when an EC2 instance terminates
 - By default, the root EBS volume is deleted (attribute enabled)
 - By default, any other attached EBS volume is not deleted (attribute disabled)
- This can be controlled by the AWS console / AWS CLI
- Use case: preserve root volume when instance is terminated

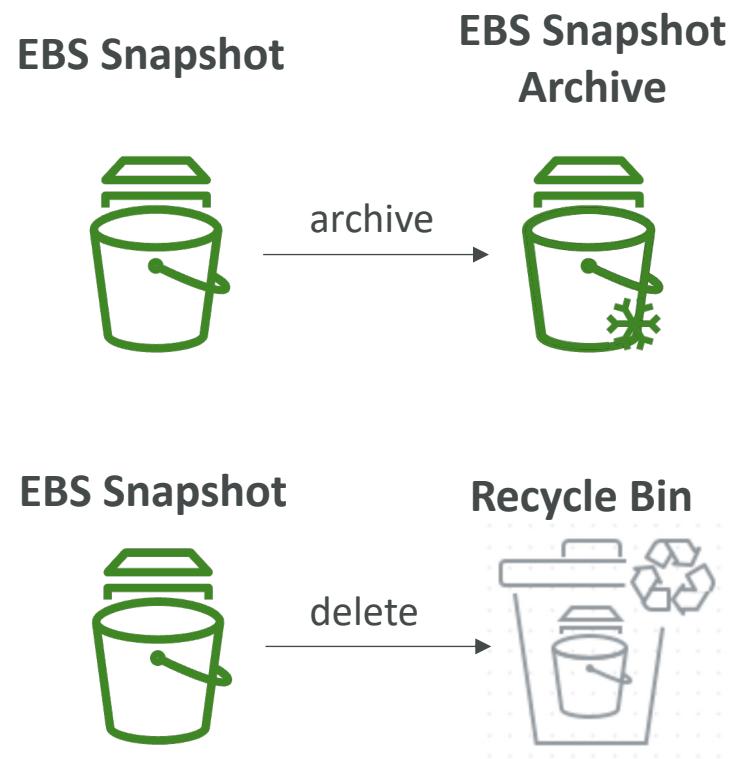
EBS Snapshots

- Make a backup (snapshot) of your EBS volume at a point in time
- Not necessary to detach volume to do snapshot, but recommended
- Can copy snapshots across AZ or Region



EBS Snapshots Features

- **EBS Snapshot Archive**
 - Move a Snapshot to an "archive tier" that is 75% cheaper
 - Takes within 24 to 72 hours for restoring the archive
- **Recycle Bin for EBS Snapshots**
 - Setup rules to retain deleted snapshots so you can recover them after an accidental deletion
 - Specify retention (from 1 day to 1 year)
- **Fast Snapshot Restore (FSR)**
 - Force full initialization of snapshot to have no latency on the first use (\$\$\$)



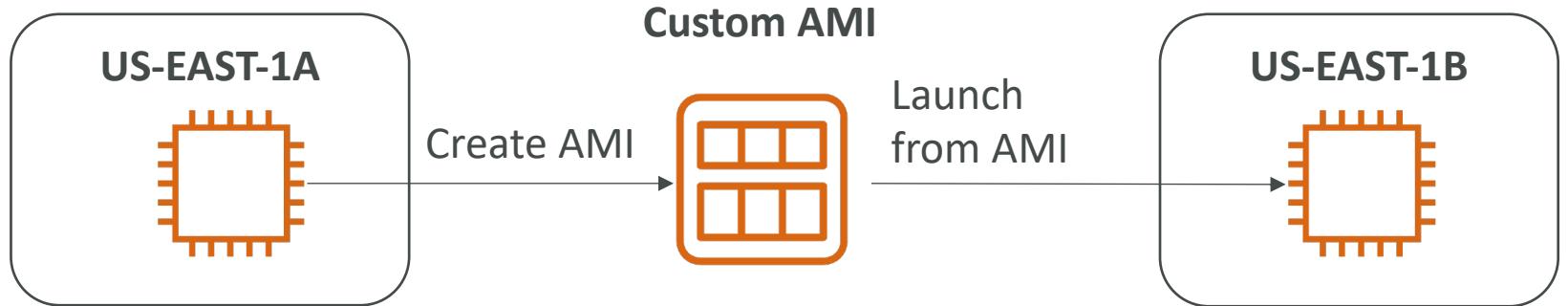


AMI Overview

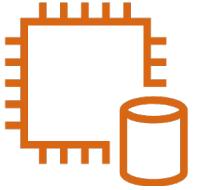
- AMI = Amazon Machine Image
- AMI are a **customization** of an EC2 instance
 - You add your own software, configuration, operating system, monitoring...
 - Faster boot / configuration time because all your software is pre-packaged
- AMI are built for a **specific region** (and can be copied across regions)
- You can launch EC2 instances from:
 - A **Public AMI**: AWS provided
 - **Your own AMI**: you make and maintain them yourself
 - An **AWS Marketplace AMI**: an AMI someone else made (and potentially sells)

AMI Process (from an EC2 instance)

- Start an EC2 instance and customize it
- Stop the instance (for data integrity)
- Build an AMI – this will also create EBS snapshots
- Launch instances from other AMIs



EC2 Instance Store



- EBS volumes are **network drives** with good but “limited” performance
- If you need a high-performance hardware disk, use EC2 Instance Store

- Better I/O performance
- EC2 Instance Store lose their storage if they're stopped (ephemeral)
- Good for buffer / cache / scratch data / temporary content
- Risk of data loss if hardware fails
- Backups and Replication are your responsibility

Local EC2 Instance Store

Very high IOPS

Instance Size	100% Random Read IOPS	Write IOPS
i3.large *	100,125	35,000
i3.xlarge *	206,250	70,000
i3.2xlarge	412,500	180,000
i3.4xlarge	825,000	360,000
i3.8xlarge	1.65 million	720,000
i3.16xlarge	3.3 million	1.4 million
i3.metal	3.3 million	1.4 million
i3en.large *	42,500	32,500
i3en.xlarge *	85,000	65,000
i3en.2xlarge *	170,000	130,000
i3en.3xlarge	250,000	200,000
i3en.6xlarge	500,000	400,000
i3en.12xlarge	1 million	800,000
i3en.24xlarge	2 million	1.6 million
i3en.metal	2 million	1.6 million

EBS Volume Types

- EBS Volumes come in 6 types
 - **gp2 / gp3 (SSD)**: General purpose SSD volume that balances price and performance for a wide variety of workloads
 - **io1 / io2 (SSD)**: Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads
 - **st1 (HDD)**: Low cost HDD volume designed for frequently accessed, throughput-intensive workloads
 - **scl (HDD)**: Lowest cost HDD volume designed for less frequently accessed workloads
- EBS Volumes are characterized in Size | Throughput | IOPS (I/O Ops Per Sec)
- When in doubt always consult the AWS documentation – it's good!
- Only gp2/gp3 and io1/io2 can be used as boot volumes

EBS Volume Types Use cases

General Purpose SSD

- Cost effective storage, low-latency
- System boot volumes, Virtual desktops, Development and test environments
- 1 GiB - 16 TiB
- gp3:
 - Baseline of 3,000 IOPS and throughput of 125 MiB/s
 - Can increase IOPS up to 16,000 and throughput up to 1000 MiB/s independently
- gp2:
 - Small gp2 volumes can burst IOPS to 3,000
 - Size of the volume and IOPS are linked, max IOPS is 16,000
 - 3 IOPS per GB, means at 5,334 GB we are at the max IOPS

EBS Volume Types Use cases

Provisioned IOPS (PIOPS) SSD

- Critical business applications with sustained IOPS performance
- Or applications that need more than 16,000 IOPS
- Great for **databases workloads** (sensitive to storage perf and consistency)
- io1/io2 (4 GiB - 16 TiB):
 - Max PIOPS: 64,000 for Nitro EC2 instances & 32,000 for other
 - Can increase PIOPS independently from storage size
 - io2 have more durability and more IOPS per GiB (at the same price as io1)
- io2 Block Express (4 GiB – 64 TiB):
 - Sub-millisecond latency
 - Max PIOPS: 256,000 with an IOPS:GiB ratio of 1,000:1
- Supports EBS Multi-attach

EBS Volume Types Use cases

Hard Disk Drives (HDD)

- Cannot be a boot volume
- 125 GiB to 16 TiB
- Throughput Optimized HDD (st1)
 - Big Data, Data Warehouses, Log Processing
 - Max throughput 500 MiB/s – max IOPS 500
- Cold HDD (sc1):
 - For data that is infrequently accessed
 - Scenarios where lowest cost is important
 - Max throughput 250 MiB/s – max IOPS 250

EBS – Volume Types Summary

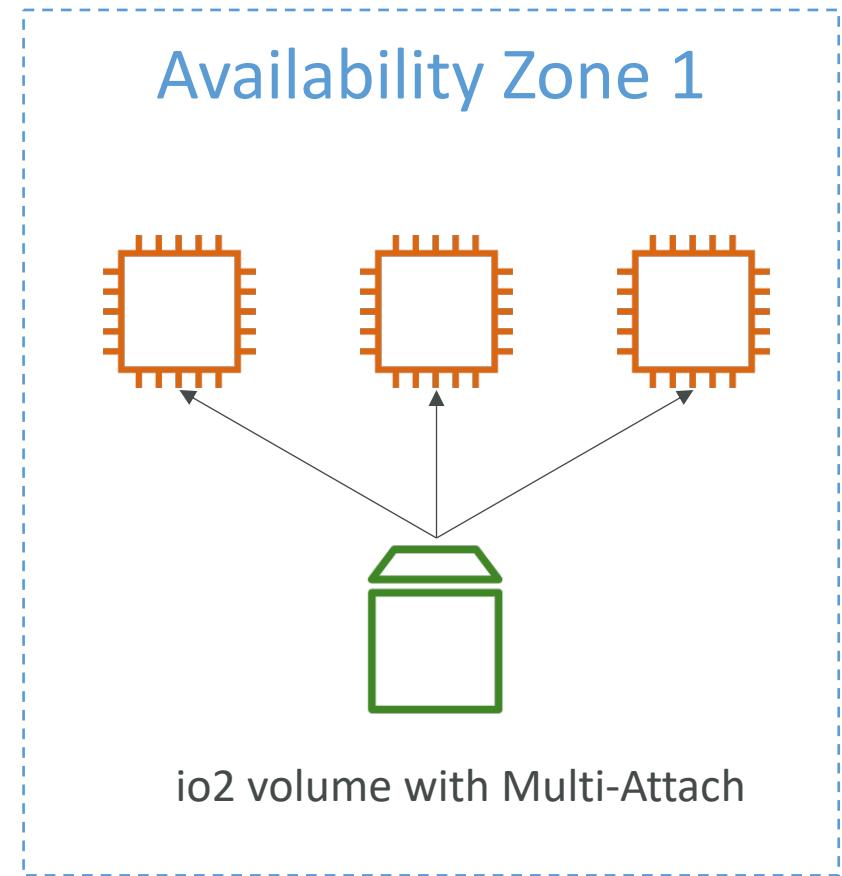
	General Purpose SSD		Provisioned IOPS SSD		
Volume type	gp3	gp2	io2 Block Express ‡	io2	io1
Durability	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.999% durability (0.001% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)
Use cases	<ul style="list-style-type: none"> Low-latency interactive apps Development and test environments 	Workloads that require sub-millisecond latency, and sustained IOPS performance or more than 64,000 IOPS or 1,000 MiB/s of throughput	<ul style="list-style-type: none"> Workloads that require sustained IOPS performance or more than 16,000 IOPS I/O-intensive database workloads 		
Volume size	1 GiB - 16 TiB	4 GiB - 64 TiB	4 GiB - 16 TiB		
Max IOPS per volume (16 KiB I/O)	16,000	256,000	64,000 †		

	Throughput Optimized HDD	Cold HDD
Volume type	st1	sc1
Durability	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)
Use cases	<ul style="list-style-type: none"> Big data Data warehouses Log processing 	<ul style="list-style-type: none"> Throughput-oriented storage for data that is infrequently accessed Scenarios where the lowest storage cost is important
Volume size	125 GiB - 16 TiB	125 GiB - 16 TiB
Max IOPS per volume (1 MiB I/O)	500	250
Max throughput per volume	500 MiB/s	250 MiB/s
Amazon EBS Multi-attach	Not supported	Not supported
Boot volume	Not supported	Not supported

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html#solid-state-drives>

EBS Multi-Attach – io1/io2 family

- Attach the same EBS volume to multiple EC2 instances in the same AZ
- Each instance has full read & write permissions to the high-performance volume
- Use case:
 - Achieve **higher application availability** in clustered Linux applications (ex: Teradata)
 - Applications must manage concurrent write operations
- **Up to 16 EC2 Instances at a time**
- Must use a file system that's cluster-aware (not XFS, EXT4, etc...)



EBS Encryption

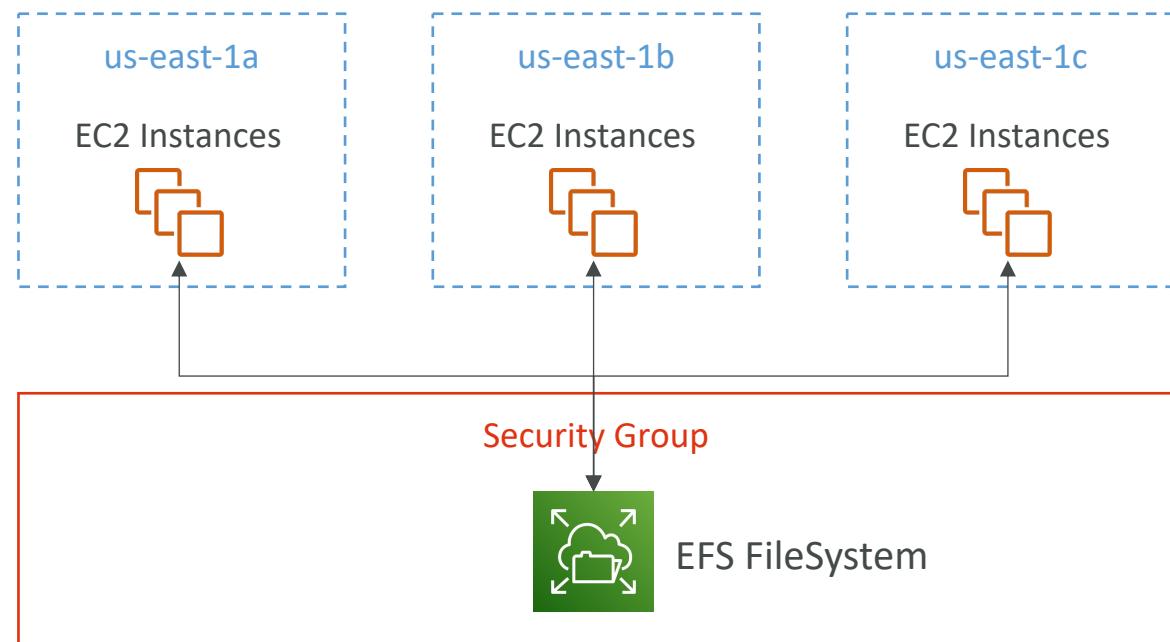
- When you create an encrypted EBS volume, you get the following:
 - Data at rest is encrypted inside the volume
 - All the data in flight moving between the instance and the volume is encrypted
 - All snapshots are encrypted
 - All volumes created from the snapshot
- Encryption and decryption are handled transparently (you have nothing to do)
- Encryption has a minimal impact on latency
- EBS Encryption leverages keys from KMS (AES-256)
- Copying an unencrypted snapshot allows encryption
- Snapshots of encrypted volumes are encrypted

Encryption: encrypt an unencrypted EBS volume

- Create an EBS snapshot of the volume
- Encrypt the EBS snapshot (using copy)
- Create new ebs volume from the snapshot (the volume will also be encrypted)
- Now you can attach the encrypted volume to the original instance

Amazon EFS – Elastic File System

- Managed NFS (network file system) that can be mounted on many EC2 instances
- EFS works with EC2 instances in multi-AZ
- Highly available, scalable, expensive (3x gp2), pay per use



Amazon EFS – Elastic File System

- Use cases: content management, web serving, data sharing, Wordpress
- Uses NFSv4.1 protocol
- Uses security group to control access to EFS
- **Compatible with Linux based AMI (not Windows)**
- Encryption at rest using KMS

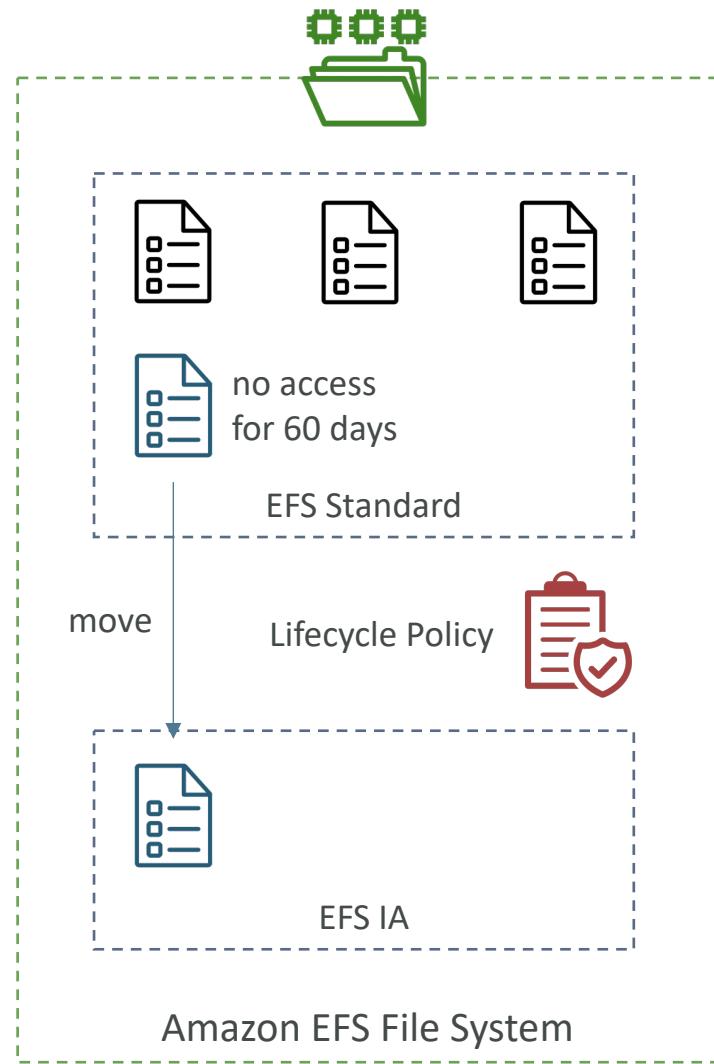
- POSIX file system (~Linux) that has a standard file API
- File system scales automatically, pay-per-use, no capacity planning!

EFS – Performance & Storage Classes

- EFS Scale
 - 1000s of concurrent NFS clients, 10 GB+ /s throughput
 - Grow to Petabyte-scale network file system, automatically
- Performance Mode (set at EFS creation time)
 - General Purpose (default) – latency-sensitive use cases (web server, CMS, etc...)
 - Max I/O – higher latency, throughput, highly parallel (big data, media processing)
- Throughput Mode
 - Bursting – 1 TB = 50MiB/s + burst of up to 100MiB/s
 - Provisioned – set your throughput regardless of storage size, ex: 1 GiB/s for 1 TB storage
 - Elastic – automatically scales throughput up or down based on your workloads
 - Up to 3GiB/s for reads and 1GiB/s for writes
 - Used for unpredictable workloads

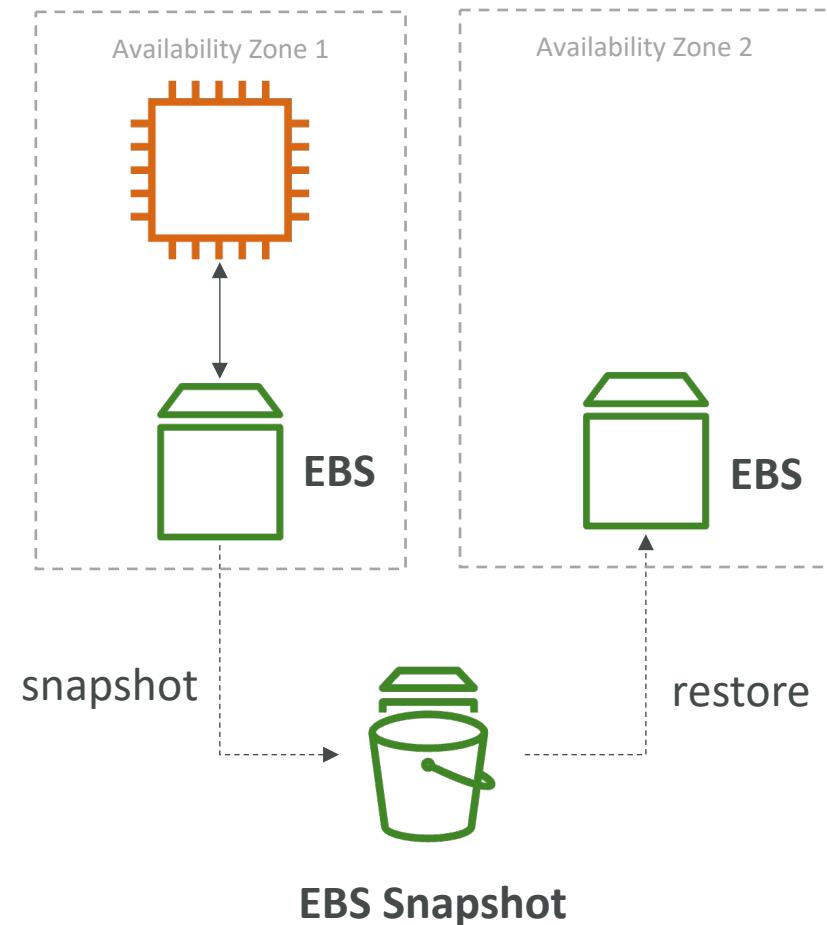
EFS – Storage Classes

- Storage Tiers (lifecycle management feature – move file after N days)
 - Standard: for frequently accessed files
 - Infrequent access (EFS-IA): cost to retrieve files, lower price to store. Enable EFS-IA with a Lifecycle Policy
- Availability and durability
 - Standard: Multi-AZ, great for prod
 - One Zone: One AZ, great for dev, backup enabled by default, compatible with IA (EFS One Zone-IA)
- Over 90% in cost savings



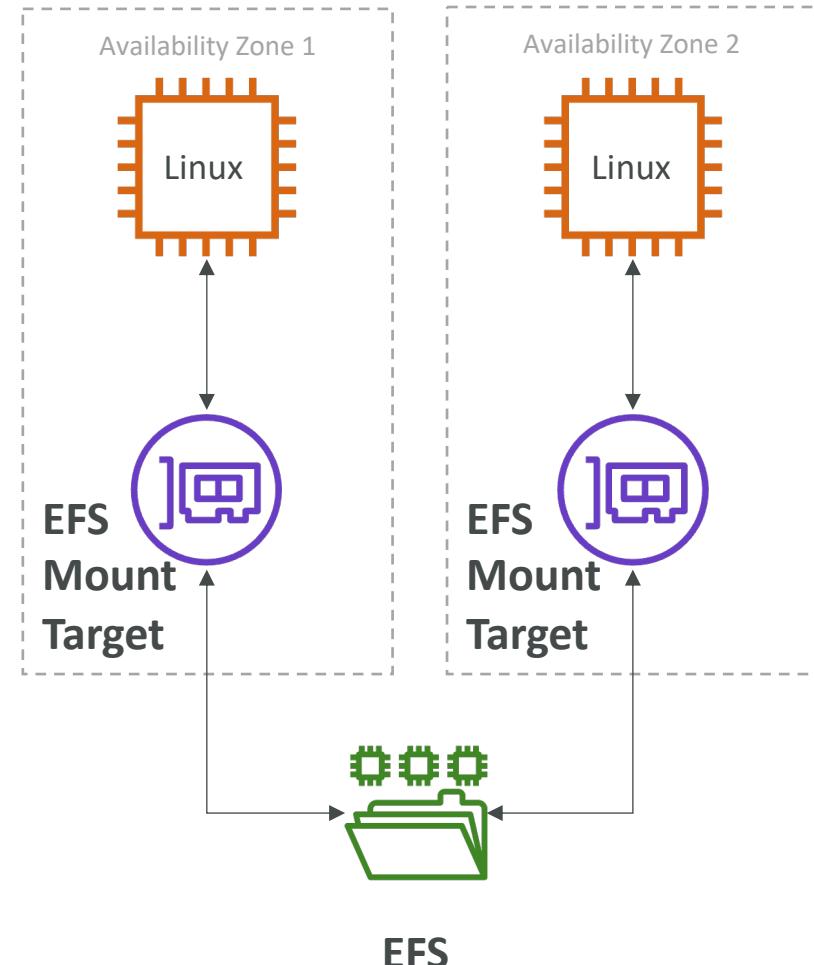
EBS vs EFS – Elastic Block Storage

- EBS volumes...
 - one instance (except multi-attach io1/io2)
 - are locked at the Availability Zone (AZ) level
 - gp2: IO increases if the disk size increases
 - io1: can increase IO independently
- To migrate an EBS volume across AZ
 - Take a snapshot
 - Restore the snapshot to another AZ
 - EBS backups use IO and you shouldn't run them while your application is handling a lot of traffic
- Root EBS Volumes of instances get terminated by default if the EC2 instance gets terminated. (you can disable that)



EBS vs EFS – Elastic File System

- Mounting 100s of instances across AZ
 - EFS share website files (WordPress)
 - Only for Linux Instances (POSIX)
-
- EFS has a higher price point than EBS
 - Can leverage EFS-IA for cost savings
-
- Remember: EFS vs EBS vs Instance Store



AWS Fundamentals – Part II

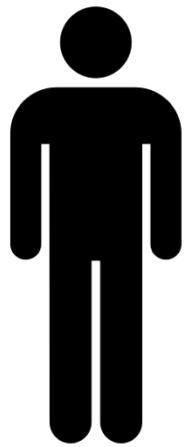
Load Balancing, Auto Scaling Groups and EBS Volumes

Scalability & High Availability

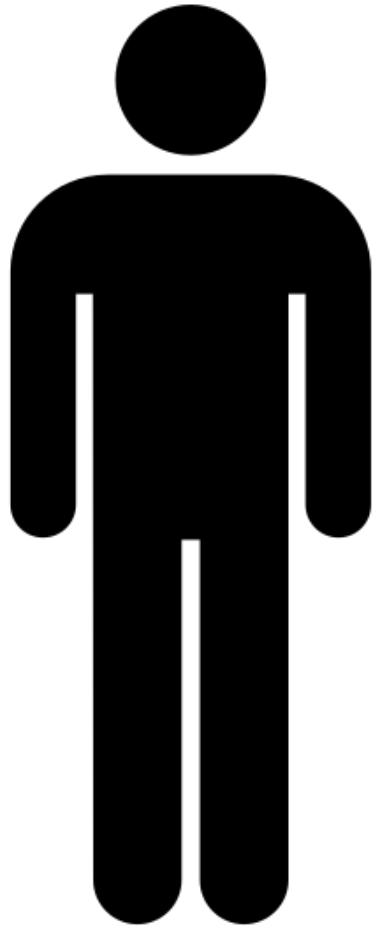
- Scalability means that an application / system can handle greater loads by adapting.
- There are two kinds of scalability:
 - Vertical Scalability
 - Horizontal Scalability (= elasticity)
- Scalability is linked but different to High Availability
- Let's deep dive into the distinction, using a call center as an example

Vertical Scalability

- Vertically scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database.
- RDS, ElastiCache are services that can scale vertically.
- There's usually a limit to how much you can vertically scale (hardware limit)



junior operator

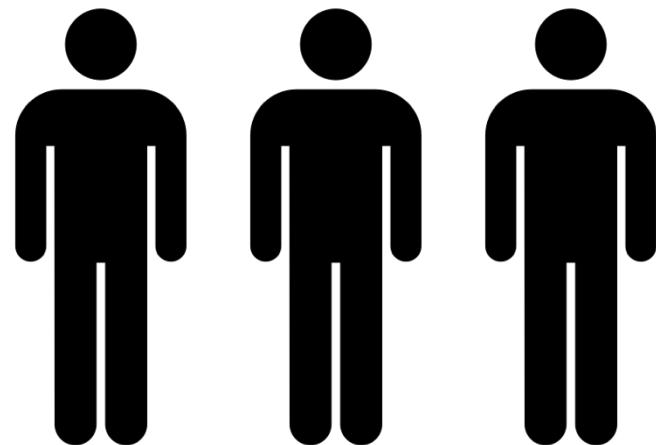
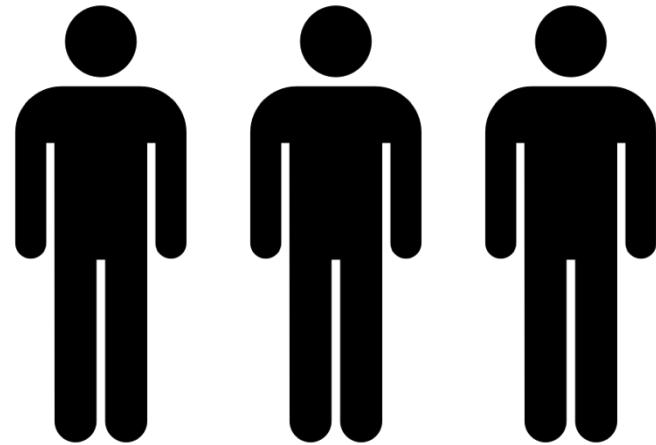


senior operator

Horizontal Scalability

- Horizontal Scalability means increasing the number of instances / systems for your application
- Horizontal scaling implies distributed systems.
- This is very common for web applications / modern applications
- It's easy to horizontally scale thanks the cloud offerings such as Amazon EC2

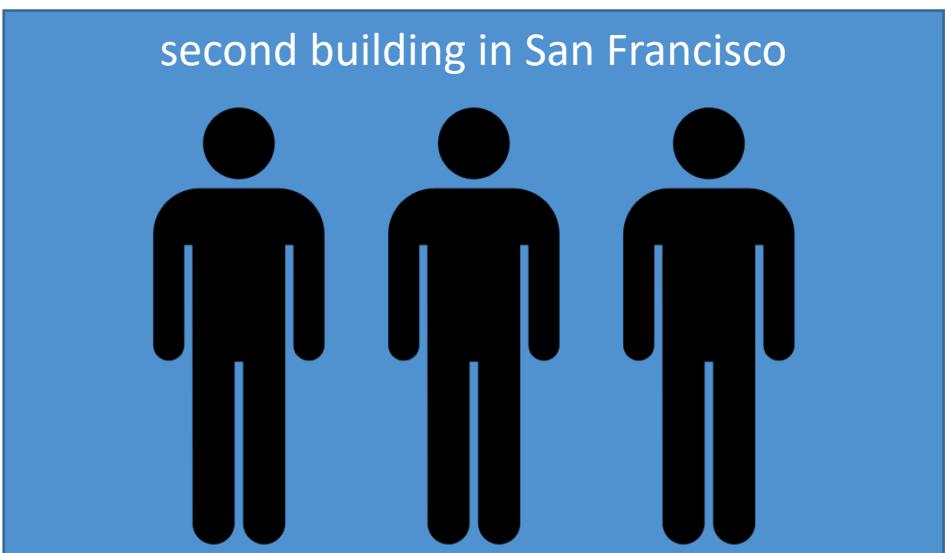
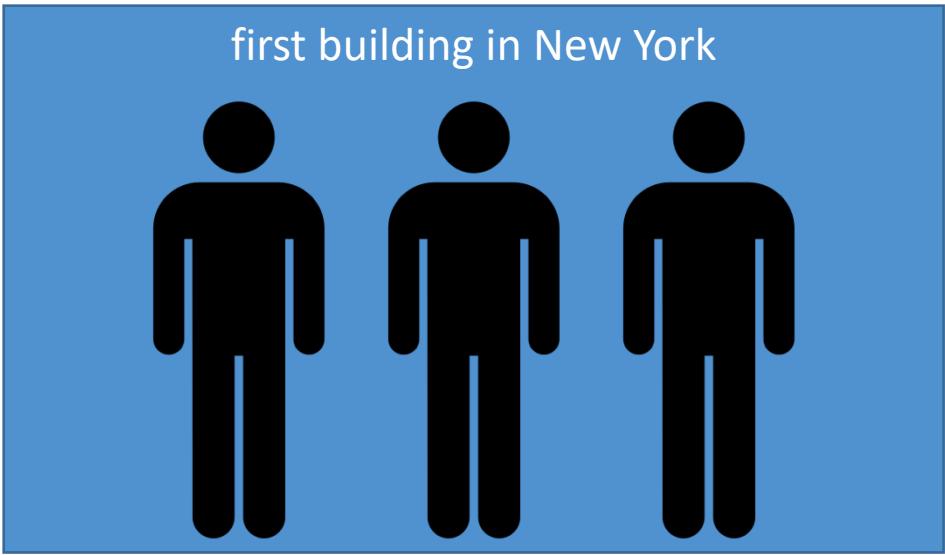
operator operator operator



operator operator operator

High Availability

- High Availability usually goes hand in hand with horizontal scaling
- High availability means running your application / system in at least 2 data centers (== Availability Zones)
- The goal of high availability is to survive a data center loss
- The high availability can be passive (for RDS Multi AZ for example)
- The high availability can be active (for horizontal scaling)

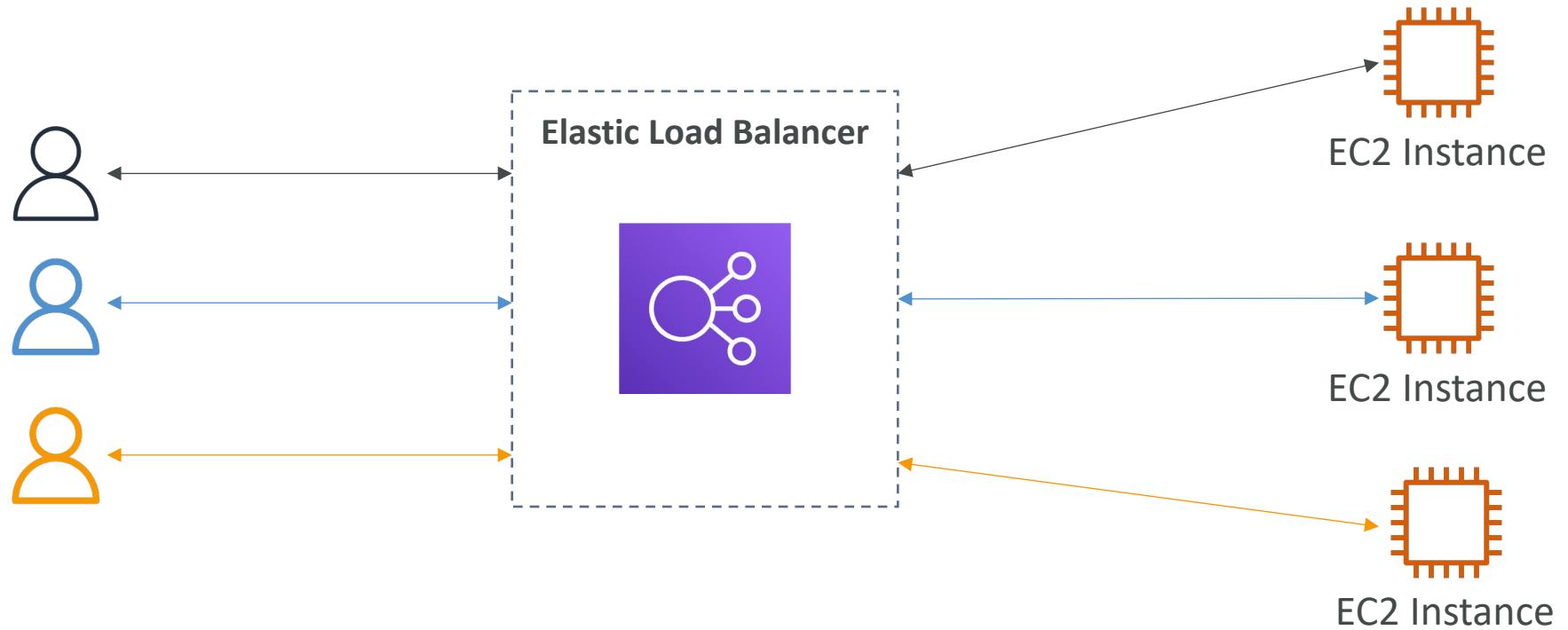


High Availability & Scalability For EC2

- Vertical Scaling: Increase instance size (= scale up / down)
 - From: t2.nano - 0.5G of RAM, 1 vCPU
 - To: u-12tbl.metal – 12.3 TB of RAM, 448 vCPUs
- Horizontal Scaling: Increase number of instances (= scale out / in)
 - Auto Scaling Group
 - Load Balancer
- High Availability: Run instances for the same application across multi AZ
 - Auto Scaling Group multi AZ
 - Load Balancer multi AZ

What is load balancing?

- Load Balances are servers that forward traffic to multiple servers (e.g., EC2 instances) downstream



Why use a load balancer?

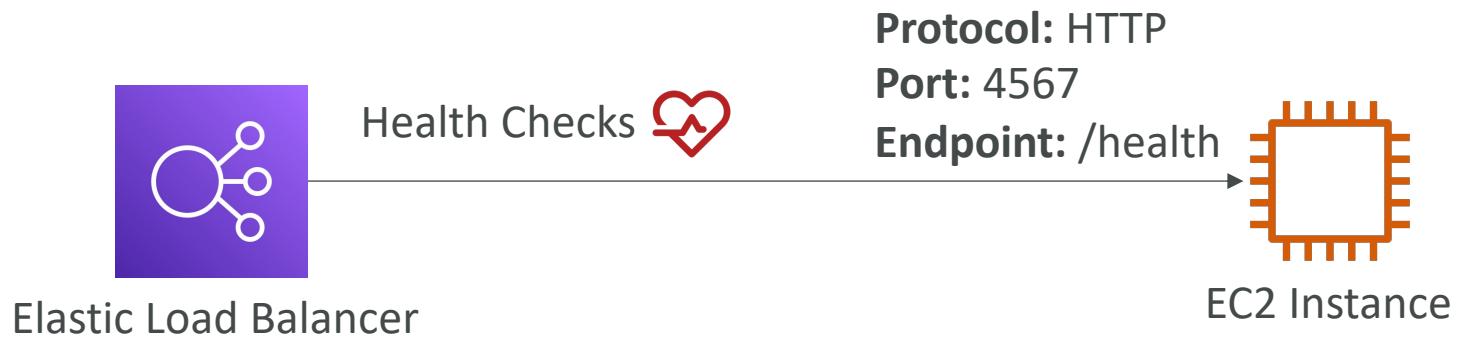
- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High availability across zones
- Separate public traffic from private traffic

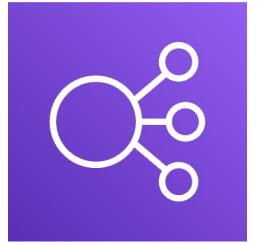
Why use an Elastic Load Balancer?

- An Elastic Load Balancer is a **managed load balancer**
 - AWS guarantees that it will be working
 - AWS takes care of upgrades, maintenance, high availability
 - AWS provides only a few configuration knobs
- It costs less to setup your own load balancer but it will be a lot more effort on your end
- It is integrated with many AWS offerings / services
 - EC2, EC2 Auto Scaling Groups, Amazon ECS
 - AWS Certificate Manager (ACM), CloudWatch
 - Route 53, AWS WAF, AWS Global Accelerator

Health Checks

- Health Checks are crucial for Load Balancers
- They enable the load balancer to know if instances it forwards traffic to are available to reply to requests
- The health check is done on a port and a route (/health is common)
- If the response is not 200 (OK), then the instance is unhealthy





Types of load balancer on AWS

- AWS has **4 kinds of managed Load Balancers**
- **Classic Load Balancer** (v1 - old generation) – 2009 – CLB
 - HTTP, HTTPS, TCP, SSL (secure TCP)
- **Application Load Balancer** (v2 - new generation) – 2016 – ALB
 - HTTP, HTTPS, WebSocket
- **Network Load Balancer** (v2 - new generation) – 2017 – NLB
 - TCP, TLS (secure TCP), UDP
- **Gateway Load Balancer** – 2020 – GWLB
 - Operates at layer 3 (Network layer) – IP Protocol
- Overall, it is recommended to use the newer generation load balancers as they provide more features
- Some load balancers can be setup as **internal** (private) or **external** (public) ELBs

Load Balancer Security Groups



Load Balancer Security Group:

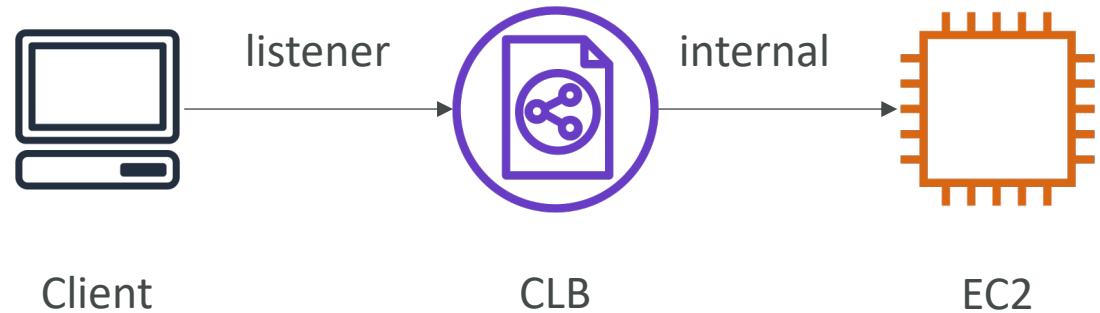
Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>	Description <small>i</small>
HTTP	TCP	80	0.0.0.0/0	Allow HTTP from an...
HTTPS	TCP	443	0.0.0.0/0	Allow HTTPS from a...

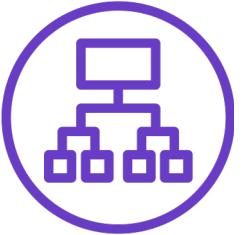
Application Security Group: Allow traffic only from Load Balancer

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>	Description <small>i</small>
HTTP	TCP	80	sg-054b5ff5ea02f2b6e (load-b	Allow Traffic only...

Classic Load Balancers (v1)

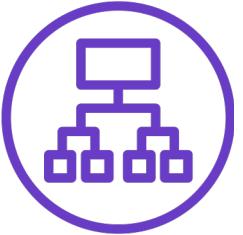
- Supports TCP (Layer 4), HTTP & HTTPS (Layer 7)
- Health checks are TCP or HTTP based
- Fixed hostname
XXX.region.elb.amazonaws.com





Application Load Balancer (v2)

- Application load balancers is Layer 7 (HTTP)
- Load balancing to multiple HTTP applications across machines (target groups)
- Load balancing to multiple applications on the same machine (ex: containers)
- Support for HTTP/2 and WebSocket
- Support redirects (from HTTP to HTTPS for example)

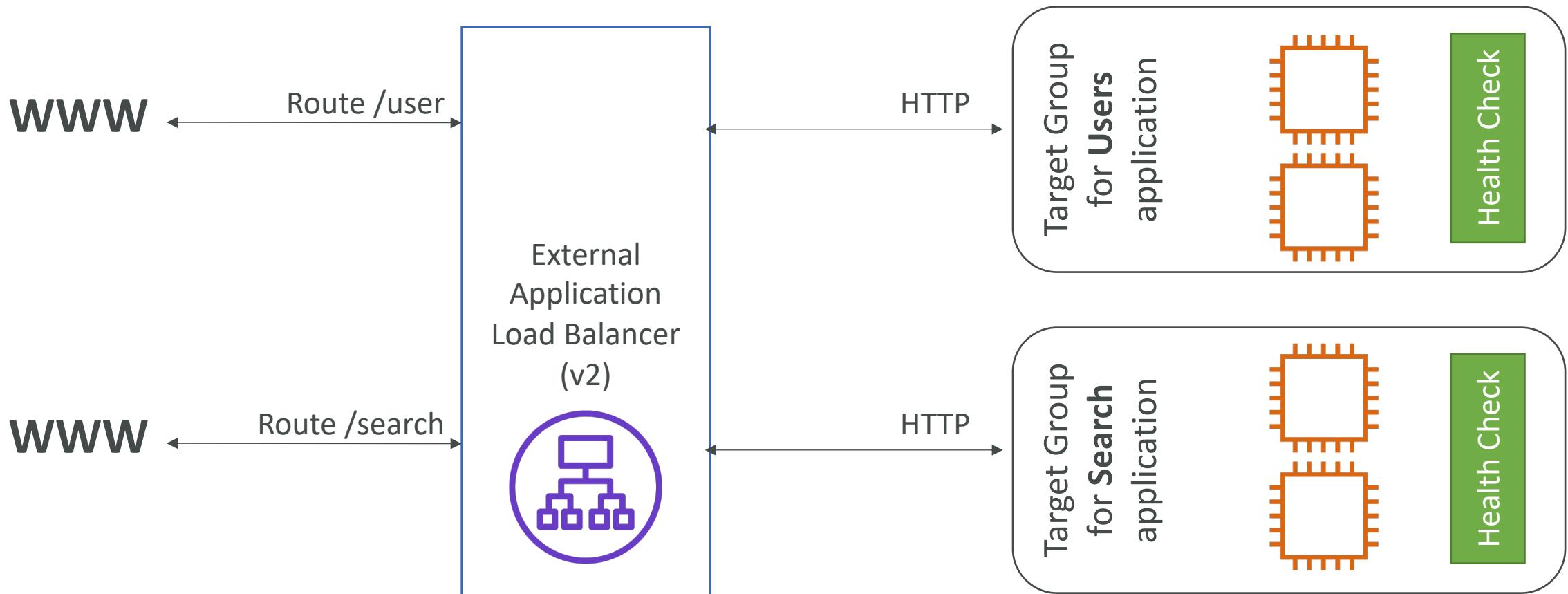


Application Load Balancer (v2)

- Routing tables to different target groups:
 - Routing based on path in URL (example.com/**users** & example.com/**posts**)
 - Routing based on hostname in URL (**one.example.com** & **other.example.com**)
 - Routing based on Query String, Headers
(example.com/users?id=123&order=false)
- ALB are a great fit for micro services & container-based application
(example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port in ECS
- In comparison, we'd need multiple Classic Load Balancer per application

Application Load Balancer (v2)

HTTP Based Traffic



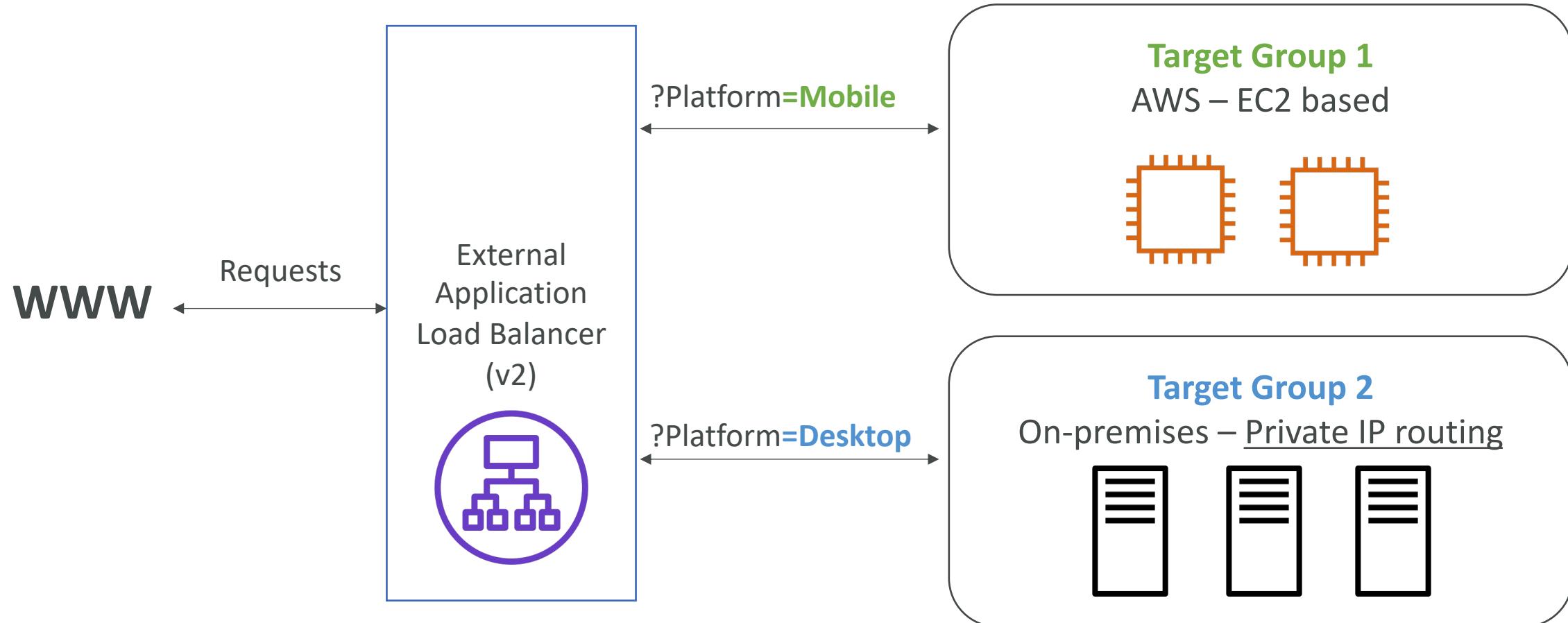
Application Load Balancer (v2)

Target Groups

- EC2 instances (can be managed by an Auto Scaling Group) – HTTP
 - ECS tasks (managed by ECS itself) – HTTP
 - Lambda functions – HTTP request is translated into a JSON event
 - IP Addresses – must be private IPs
-
- ALB can route to multiple target groups
 - Health checks are at the target group level

Application Load Balancer (v2)

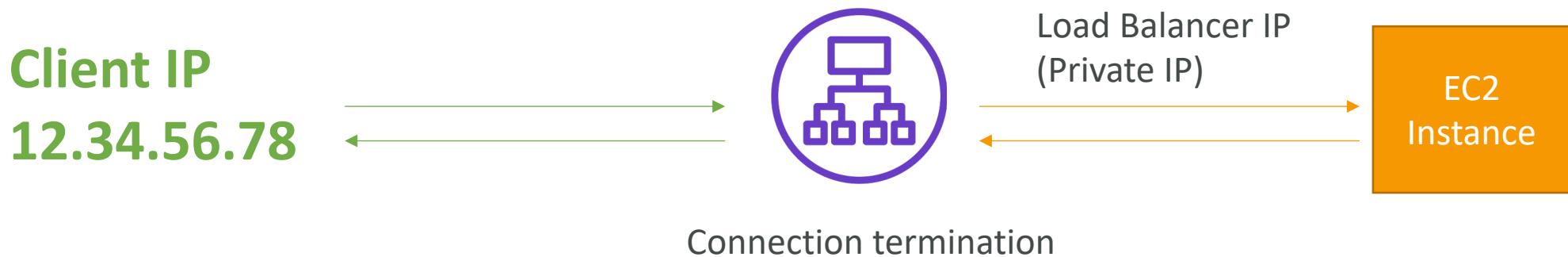
Query Strings/Parameters Routing

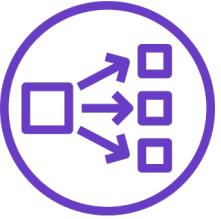


Application Load Balancer (v2)

Good to Know

- Fixed hostname (XXX.region.elb.amazonaws.com)
- The application servers don't see the IP of the client directly
 - The true IP of the client is inserted in the header X-Forwarded-For
 - We can also get Port (X-Forwarded-Port) and proto (X-Forwarded-Proto)



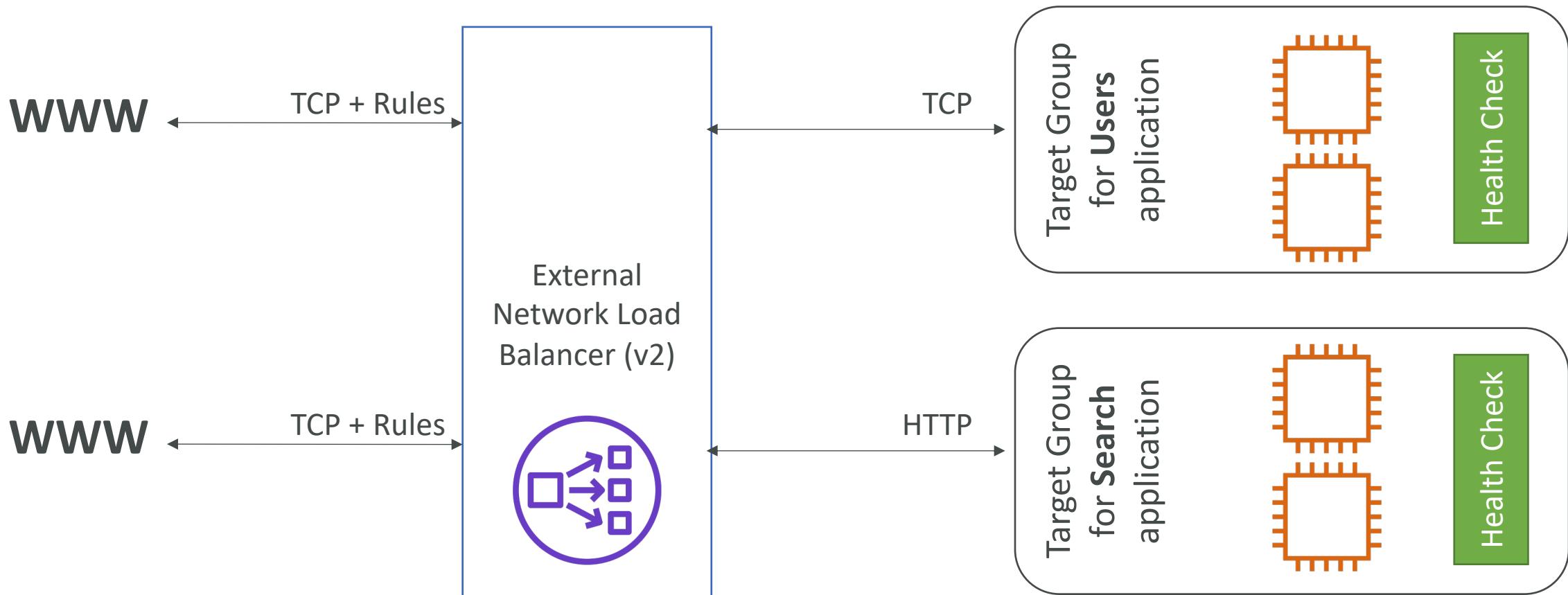


Network Load Balancer (v2)

- Network load balancers (Layer 4) allow to:
 - Forward TCP & UDP traffic to your instances
 - Handle millions of requests per second
 - Less latency ~100 ms (vs 400 ms for ALB)
- NLB has one static IP per AZ, and supports assigning Elastic IP (helpful for whitelisting specific IP)
- NLB are used for extreme performance, TCP or UDP traffic
- Not included in the AWS free tier

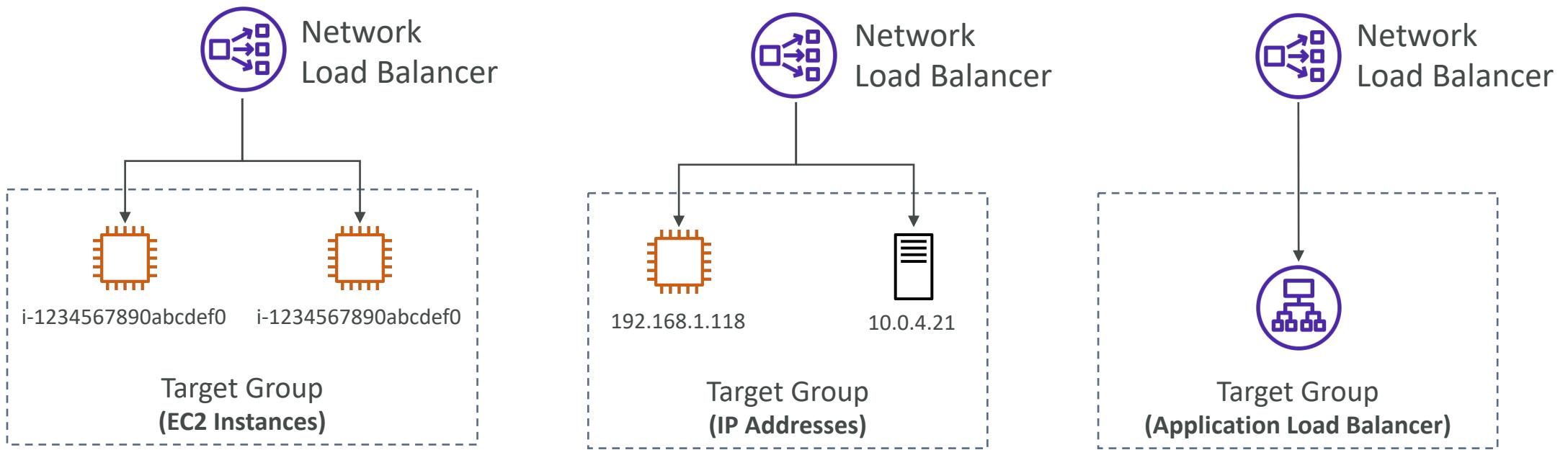
Network Load Balancer (v2)

TCP (Layer 4) Based Traffic



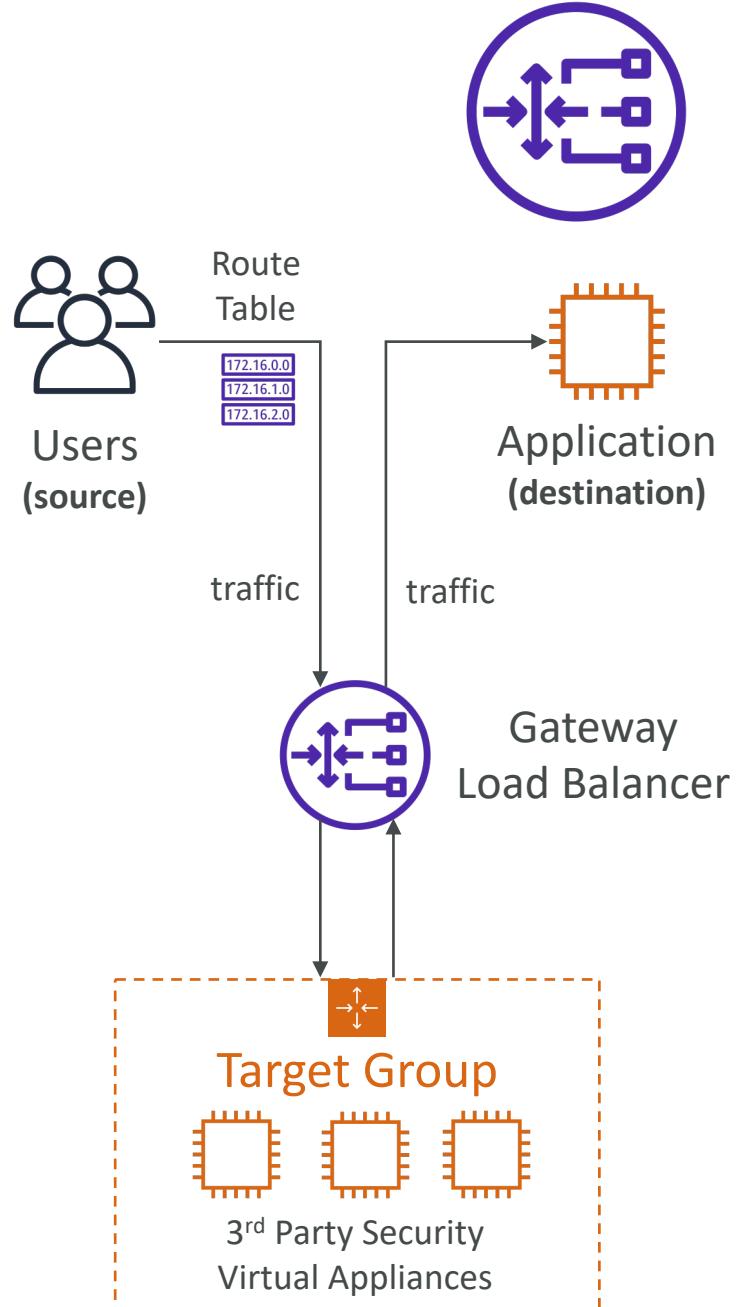
Network Load Balancer – Target Groups

- EC2 instances
- IP Addresses – must be private IPs
- Application Load Balancer
- Health Checks support the TCP, HTTP and HTTPS Protocols



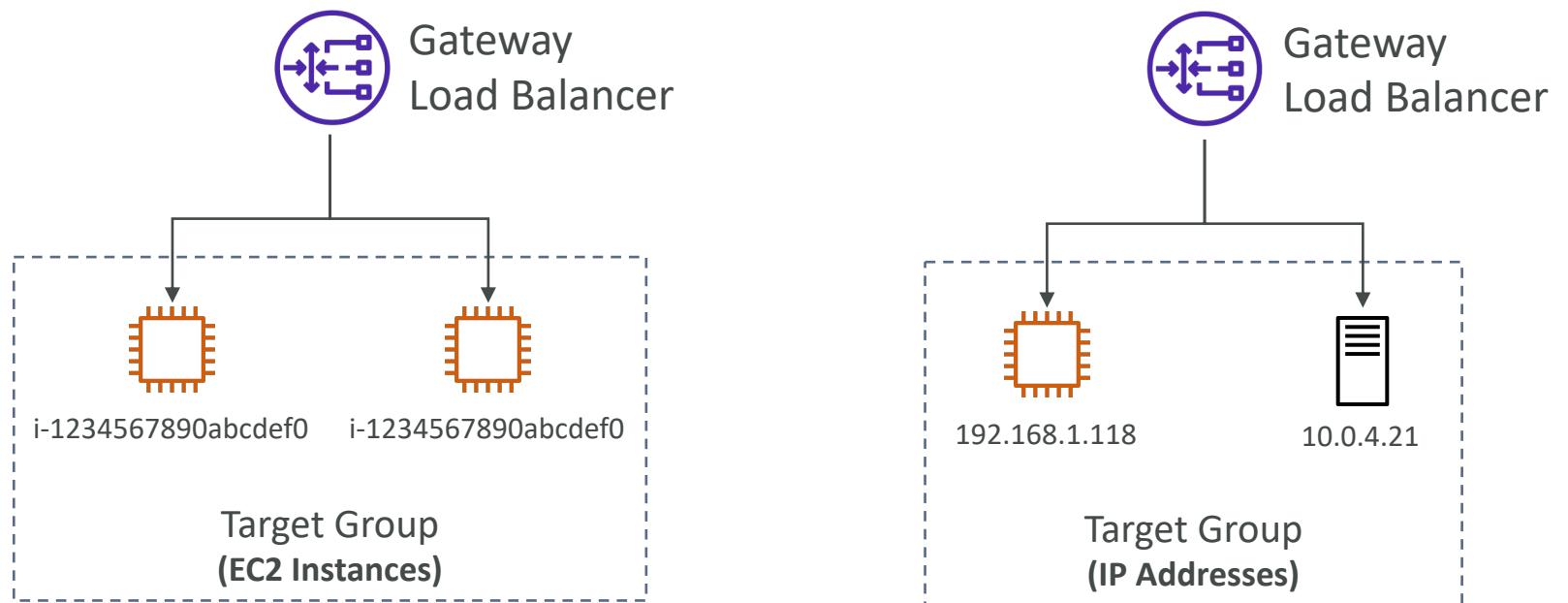
Gateway Load Balancer

- Deploy, scale, and manage a fleet of 3rd party network virtual appliances in AWS
- Example: Firewalls, Intrusion Detection and Prevention Systems, Deep Packet Inspection Systems, payload manipulation, ...
- Operates at Layer 3 (Network Layer) – IP Packets
- Combines the following functions:
 - **Transparent Network Gateway** – single entry/exit for all traffic
 - **Load Balancer** – distributes traffic to your virtual appliances
- Uses the **GENEVE** protocol on port 6081



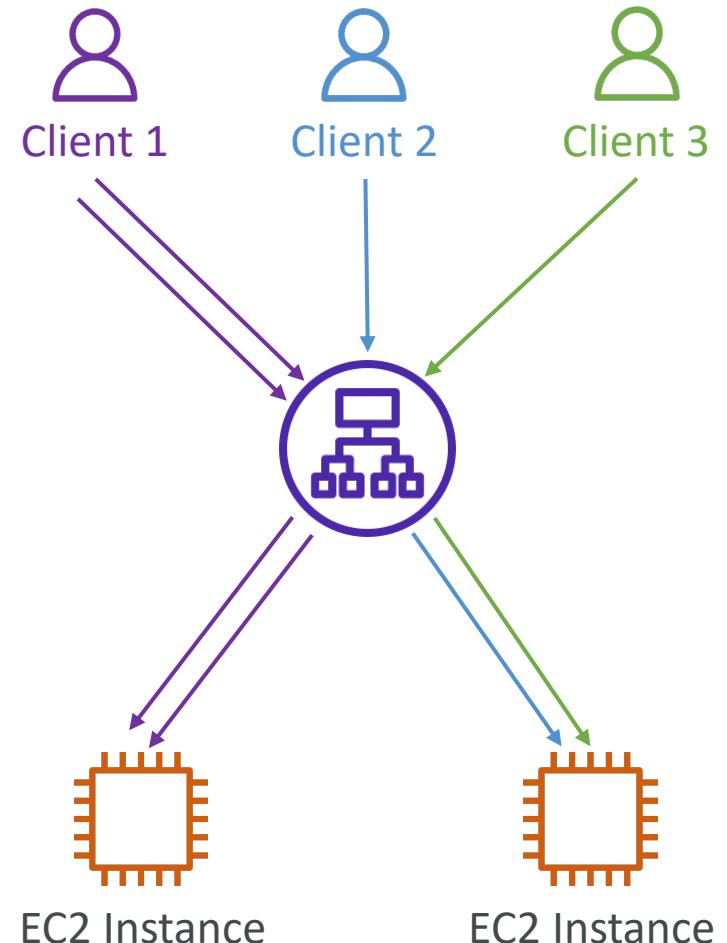
Gateway Load Balancer – Target Groups

- EC2 instances
- IP Addresses – must be private IPs



Sticky Sessions (Session Affinity)

- It is possible to implement stickiness so that the same client is always redirected to the same instance behind a load balancer
- This works for **Classic Load Balancer, Application Load Balancer, and Network Load Balancer**
- For both CLB & ALB, the “cookie” used for stickiness has an expiration date you control
- Use case: make sure the user doesn’t lose his session data
- Enabling stickiness may bring imbalance to the load over the backend EC2 instances



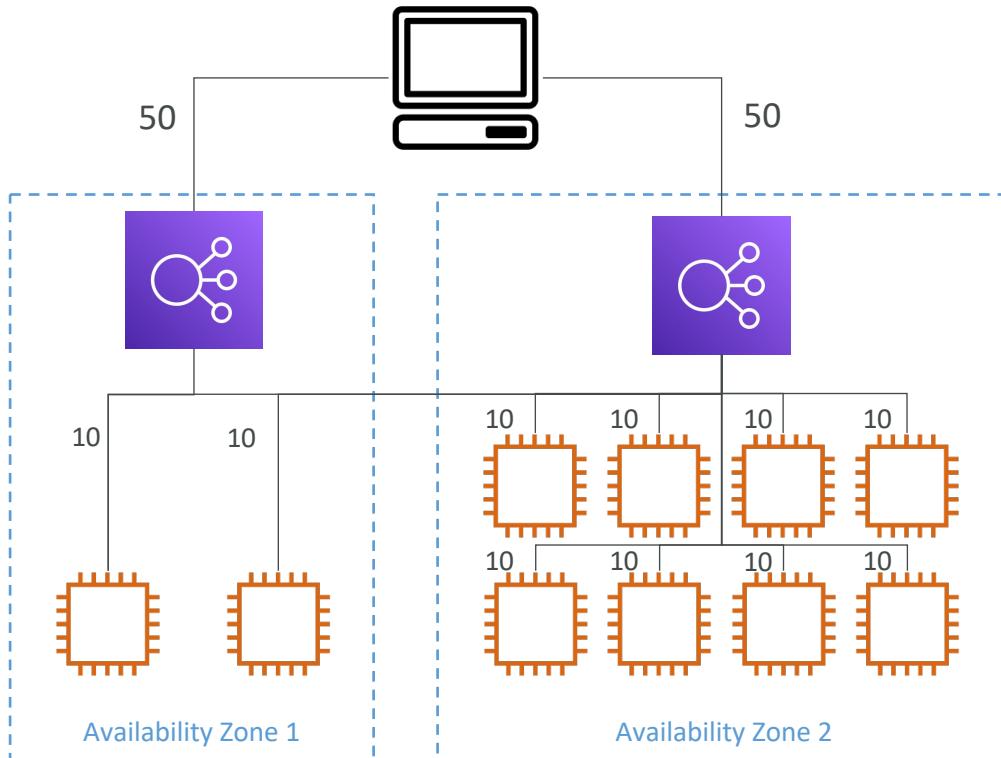
Sticky Sessions – Cookie Names

- Application-based Cookies
 - Custom cookie
 - Generated by the target
 - Can include any custom attributes required by the application
 - Cookie name must be specified individually for each target group
 - Don't use **AWSALB**, **AWSALBAPP**, or **AWSALBTG** (reserved for use by the ELB)
 - Application cookie
 - Generated by the load balancer
 - Cookie name is **AWSALBAPP**
- Duration-based Cookies
 - Cookie generated by the load balancer
 - Cookie name is **AWSALB** for ALB, **AWSELB** for CLB

Cross-Zone Load Balancing

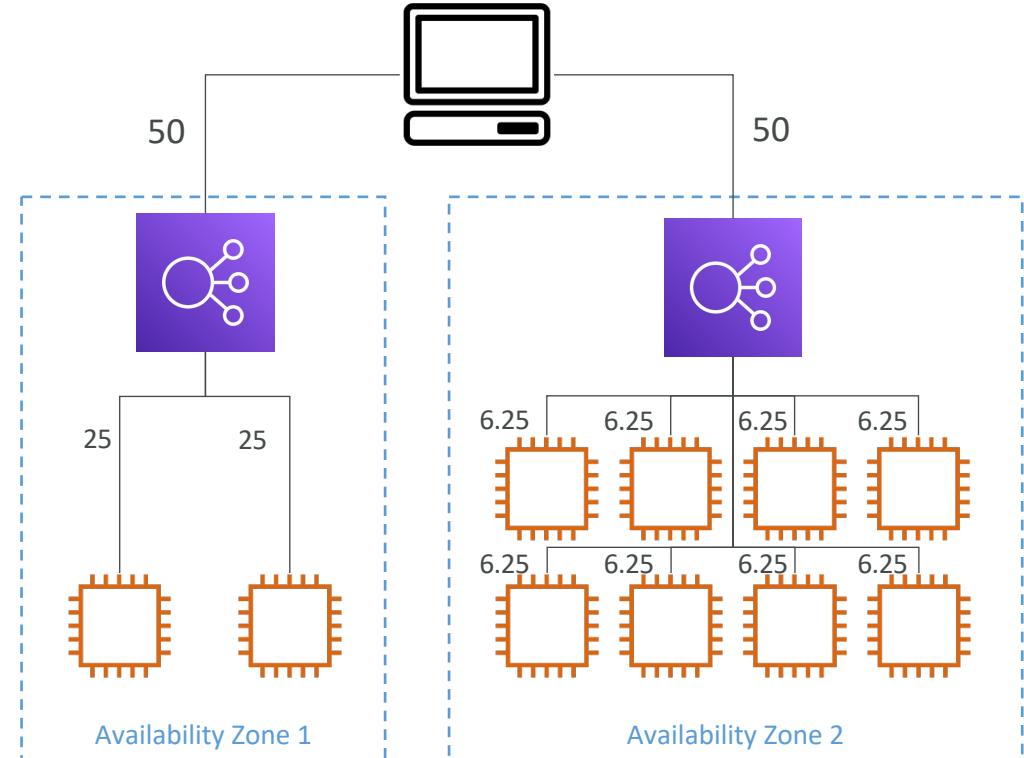
With Cross Zone Load Balancing:

each load balancer instance distributes evenly across all registered instances in all AZ



Without Cross Zone Load Balancing:

Requests are distributed in the instances of the node of the Elastic Load Balancer



Cross-Zone Load Balancing

- Application Load Balancer
 - Enabled by default (can be disabled at the Target Group level)
 - No charges for inter AZ data
- Network Load Balancer & Gateway Load Balancer
 - Disabled by default
 - You pay charges (\$) for inter AZ data if enabled
- Classic Load Balancer
 - Disabled by default
 - No charges for inter AZ data if enabled

SSL/TLS - Basics

- An SSL Certificate allows traffic between your clients and your load balancer to be encrypted in transit (in-flight encryption)
- SSL refers to Secure Sockets Layer, used to encrypt connections
- TLS refers to Transport Layer Security, which is a newer version
- Nowadays, **TLS certificates are mainly used**, but people still refer as SSL
- Public SSL certificates are issued by Certificate Authorities (CA)
- Comodo, Symantec, GoDaddy, GlobalSign, DigiCert, LetsEncrypt, etc...
- SSL certificates have an expiration date (you set) and must be renewed

Load Balancer - SSL Certificates



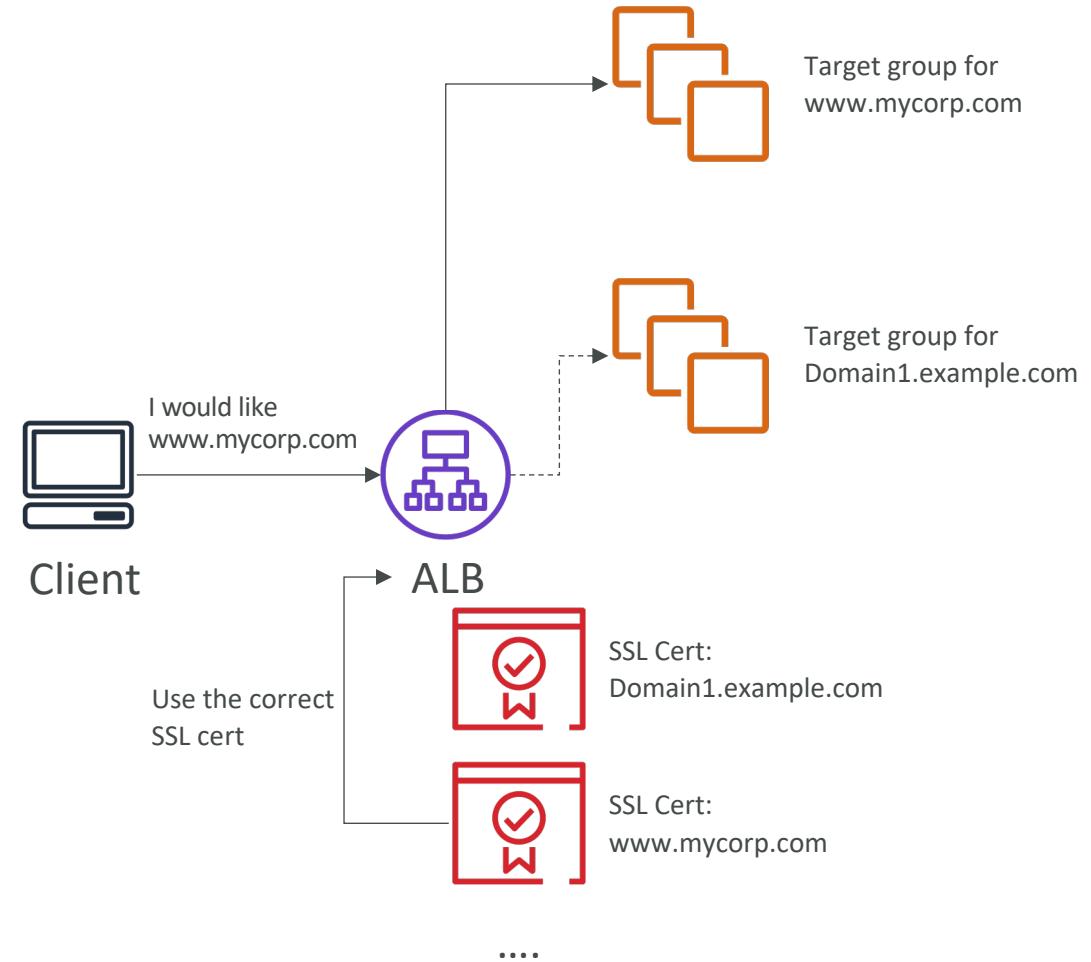
- The load balancer uses an X.509 certificate (SSL/TLS server certificate)
- You can manage certificates using ACM (AWS Certificate Manager)
- You can create/upload your own certificates alternatively
- HTTPS listener:
 - You must specify a default certificate
 - You can add an optional list of certs to support multiple domains
 - **Clients can use SNI (Server Name Indication) to specify the hostname they reach**
 - Ability to specify a security policy to support older versions of SSL / TLS (legacy clients)

SSL – Server Name Indication (SNI)

- SNI solves the problem of loading **multiple SSL certificates onto one web server** (to serve multiple websites)
- It's a “newer” protocol, and requires the client to **indicate** the hostname of the target server in the initial SSL handshake
- The server will then find the correct certificate, or return the default one

Note:

- Only works for ALB & NLB (newer generation), CloudFront
- Does not work for CLB (older gen)

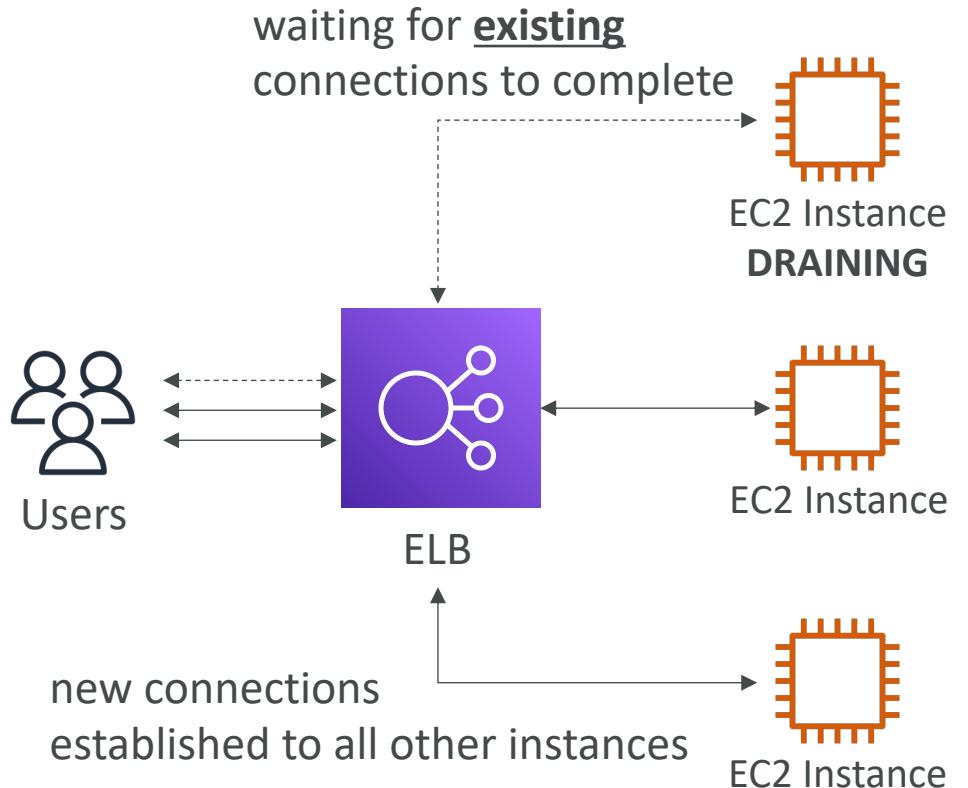


Elastic Load Balancers – SSL Certificates

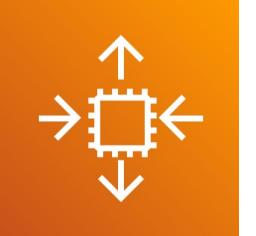
- **Classic Load Balancer (v1)**
 - Support only one SSL certificate
 - Must use multiple CLB for multiple hostname with multiple SSL certificates
- **Application Load Balancer (v2)**
 - Supports multiple listeners with multiple SSL certificates
 - Uses Server Name Indication (SNI) to make it work
- **Network Load Balancer (v2)**
 - Supports multiple listeners with multiple SSL certificates
 - Uses Server Name Indication (SNI) to make it work

Connection Draining

- Feature naming
 - Connection Draining – for CLB
 - Deregistration Delay – for ALB & NLB
- Time to complete “in-flight requests” while the instance is de-registering or unhealthy
- Stops sending new requests to the EC2 instance which is de-registering
- Between 1 to 3600 seconds (default: 300 seconds)
- Can be disabled (set value to 0)
- Set to a low value if your requests are short

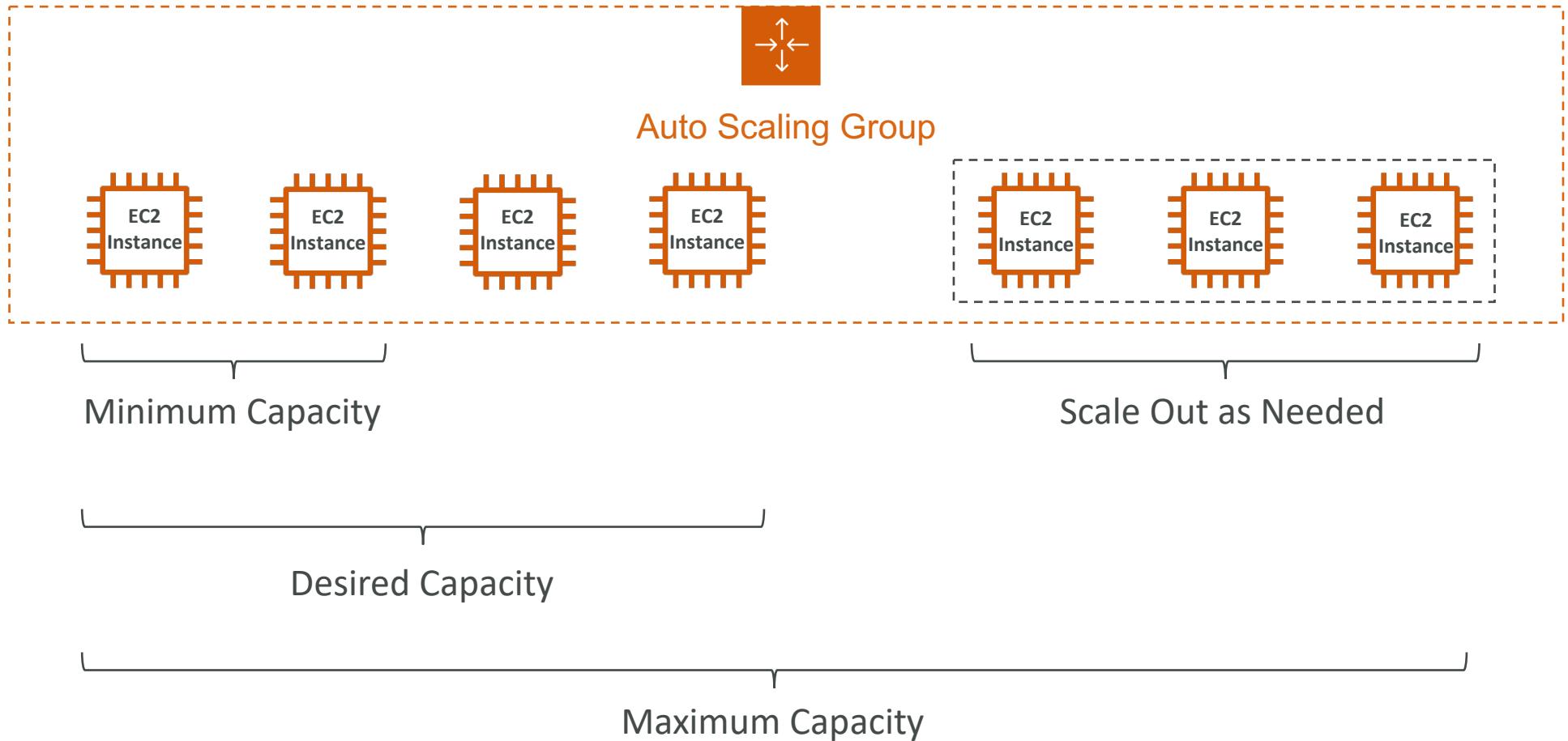


What's an Auto Scaling Group?

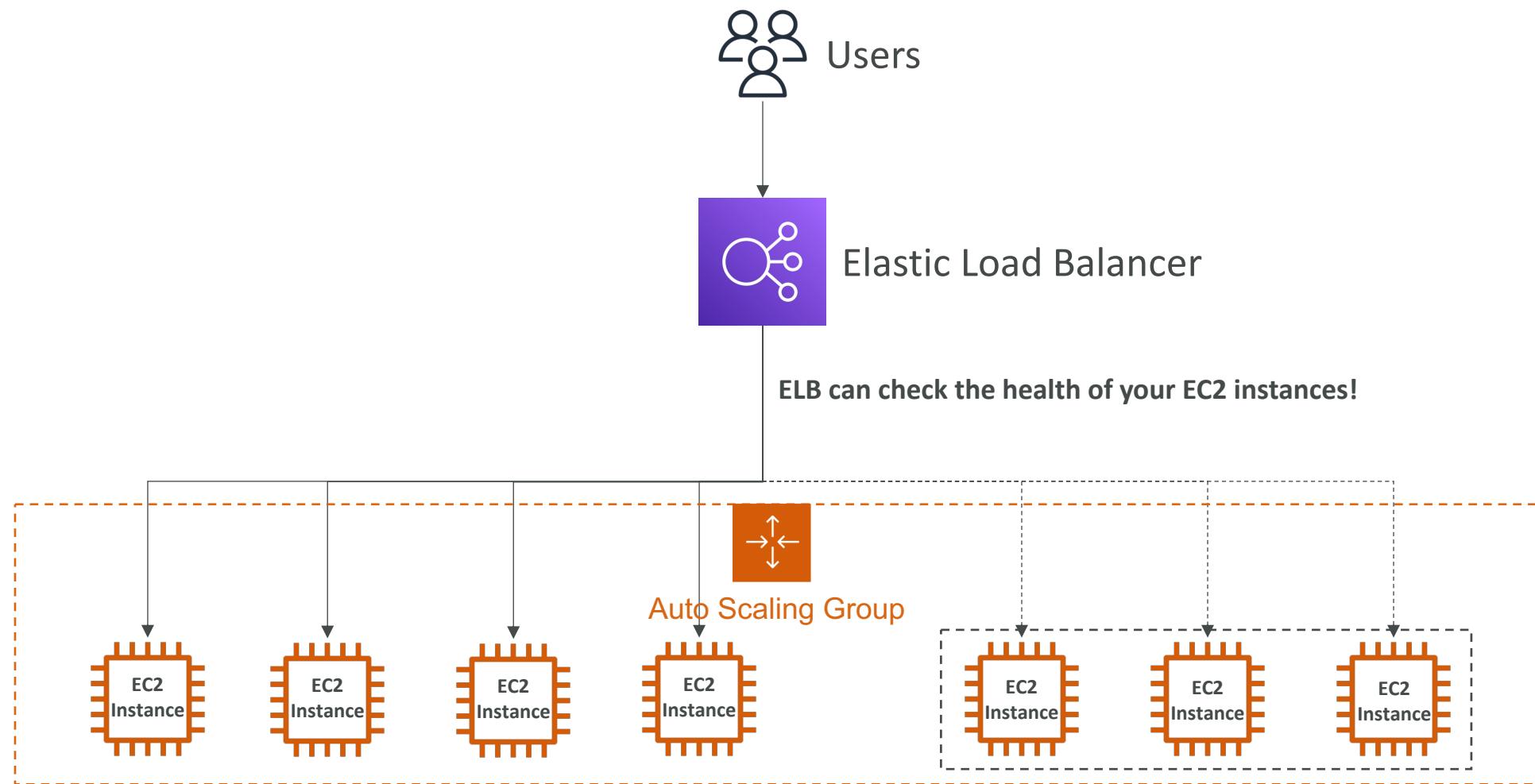


- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly
- The goal of an Auto Scaling Group (ASG) is to:
 - Scale out (add EC2 instances) to match an increased load
 - Scale in (remove EC2 instances) to match a decreased load
 - Ensure we have a minimum and a maximum number of EC2 instances running
 - Automatically register new instances to a load balancer
 - Re-create an EC2 instance in case a previous one is terminated (ex: if unhealthy)
- ASG are free (you only pay for the underlying EC2 instances)

Auto Scaling Group in AWS

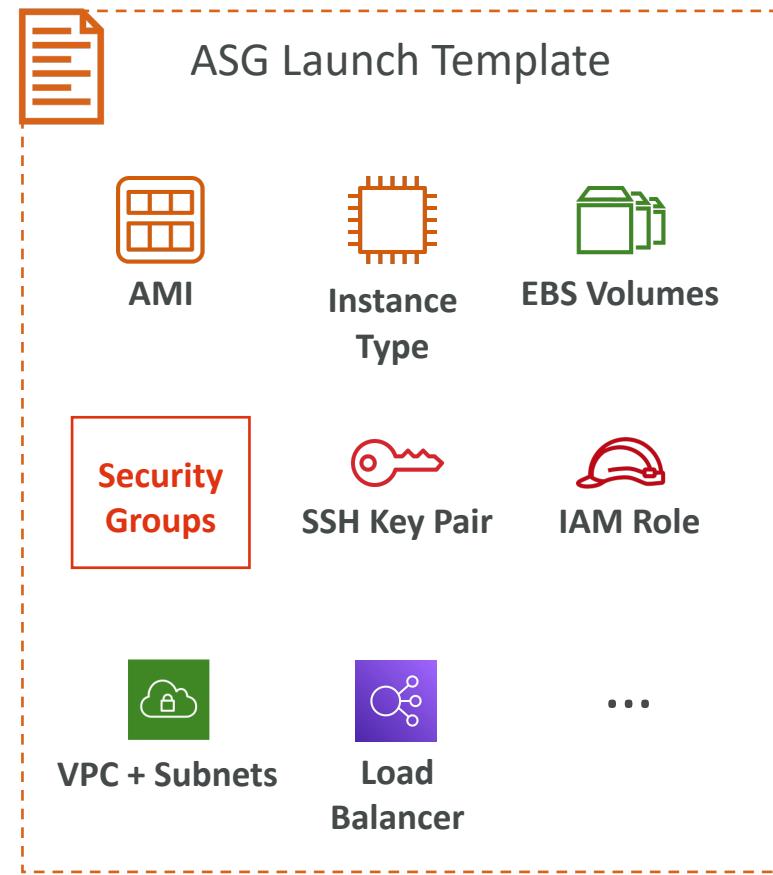


Auto Scaling Group in AWS With Load Balancer



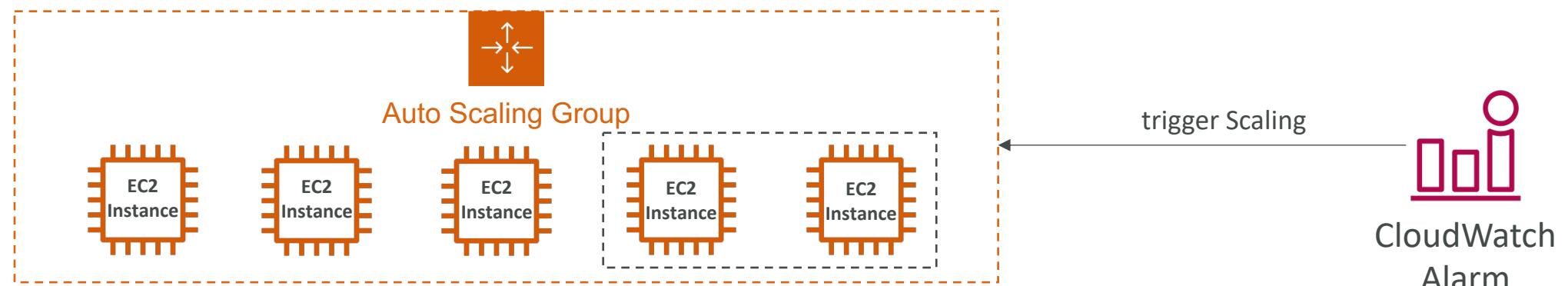
Auto Scaling Group Attributes

- A **Launch Template** (older “Launch Configurations” are deprecated)
 - AMI + Instance Type
 - EC2 User Data
 - EBS Volumes
 - Security Groups
 - SSH Key Pair
 - IAM Roles for your EC2 Instances
 - Network + Subnets Information
 - Load Balancer Information
- Min Size / Max Size / Initial Capacity
- Scaling Policies



Auto Scaling - CloudWatch Alarms & Scaling

- It is possible to scale an ASG based on CloudWatch alarms
- An alarm monitors a metric (such as **Average CPU**, or a **custom metric**)
- Metrics such as Average CPU are computed for the overall ASG instances
- Based on the alarm:
 - We can create scale-out policies (increase the number of instances)
 - We can create scale-in policies (decrease the number of instances)

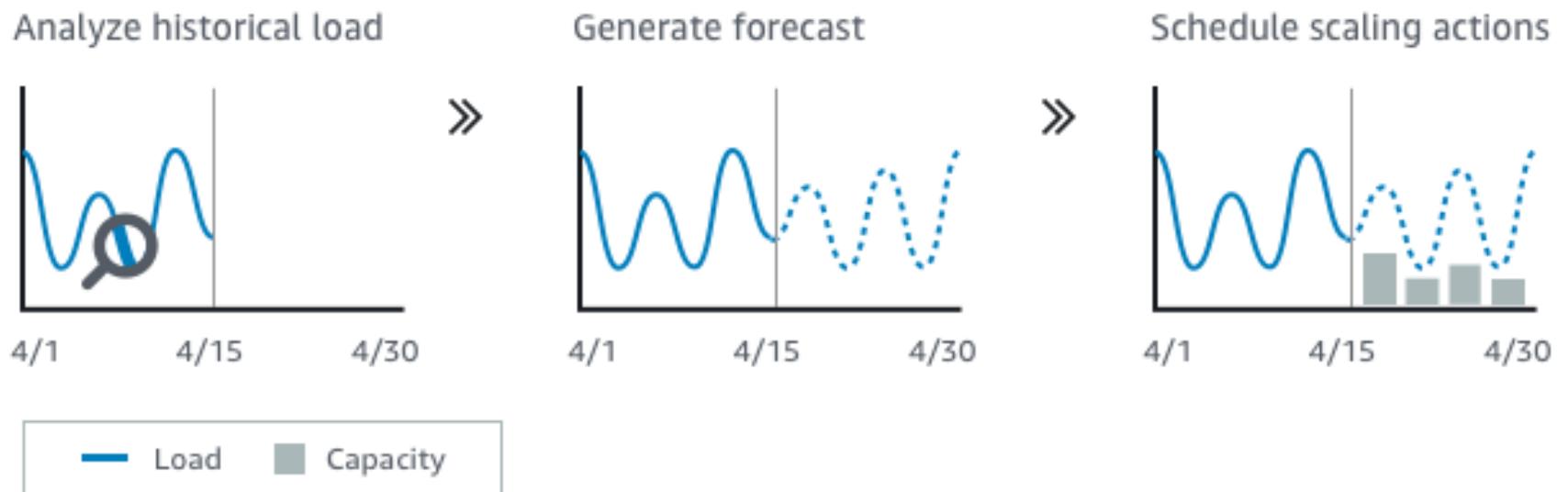


Auto Scaling Groups – Dynamic Scaling Policies

- Target Tracking Scaling
 - Most simple and easy to set-up
 - Example: I want the average ASG CPU to stay at around 40%
- Simple / Step Scaling
 - When a CloudWatch alarm is triggered (example CPU > 70%), then add 2 units
 - When a CloudWatch alarm is triggered (example CPU < 30%), then remove 1
- Scheduled Actions
 - Anticipate a scaling based on known usage patterns
 - Example: increase the min capacity to 10 at 5 pm on Fridays

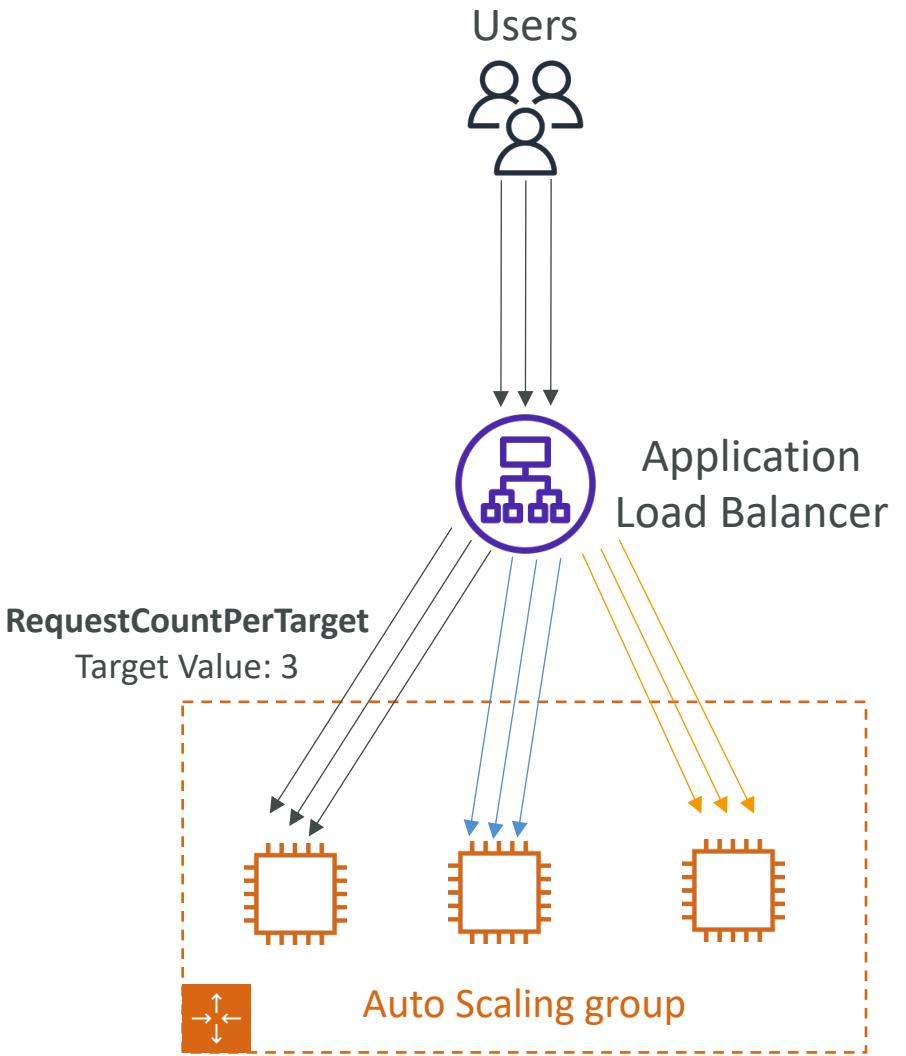
Auto Scaling Groups – Predictive Scaling

- Predictive scaling: continuously forecast load and schedule scaling ahead



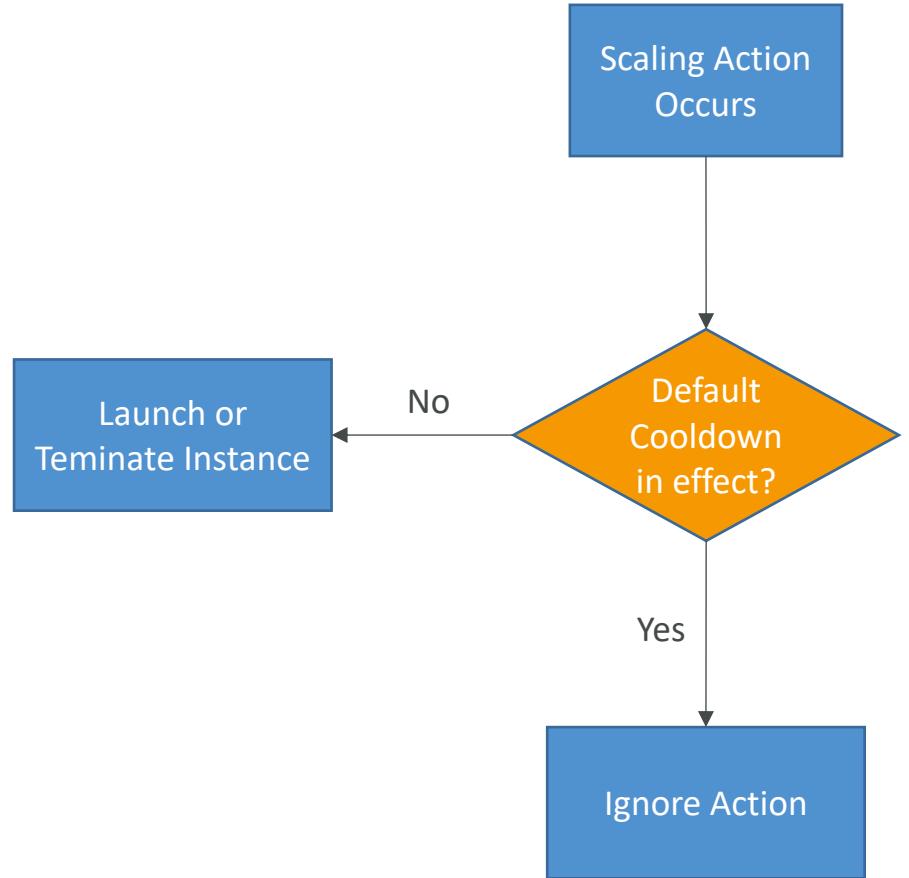
Good metrics to scale on

- **CPUUtilization:** Average CPU utilization across your instances
- **RequestCountPerTarget:** to make sure the number of requests per EC2 instances is stable
- **Average Network In / Out** (if you're application is network bound)
- Any custom metric (that you push using CloudWatch)



Auto Scaling Groups - Scaling Cooldowns

- After a scaling activity happens, you are in the cooldown period (default 300 seconds)
- During the cooldown period, the ASG will not launch or terminate additional instances (to allow for metrics to stabilize)
- Advice: Use a ready-to-use AMI to reduce configuration time in order to be serving request faster and reduce the cooldown period



RDS, Aurora & ElastiCache

Amazon RDS Overview



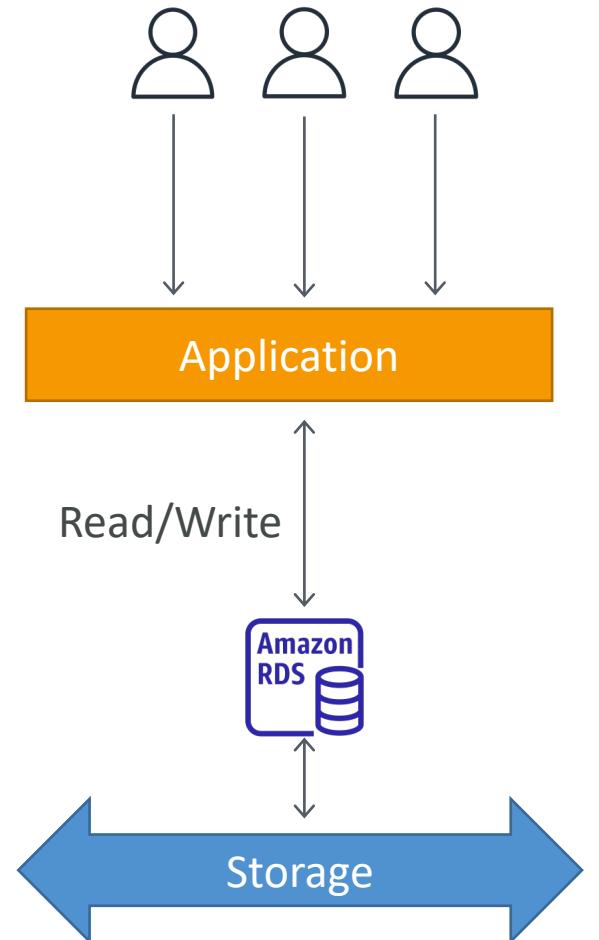
- RDS stands for Relational Database Service
- It's a managed DB service for DB use SQL as a query language.
- It allows you to create databases in the cloud that are managed by AWS
 - Postgres
 - MySQL
 - MariaDB
 - Oracle
 - Microsoft SQL Server
 - Aurora (AWS Proprietary database)

Advantage over using RDS versus deploying DB on EC2

- RDS is a managed service:
 - Automated provisioning, OS patching
 - Continuous backups and restore to specific timestamp (Point in Time Restore)!
 - Monitoring dashboards
 - Read replicas for improved read performance
 - Multi AZ setup for DR (Disaster Recovery)
 - Maintenance windows for upgrades
 - Scaling capability (vertical and horizontal)
 - Storage backed by EBS (gp2 or io1)
- BUT you can't SSH into your instances

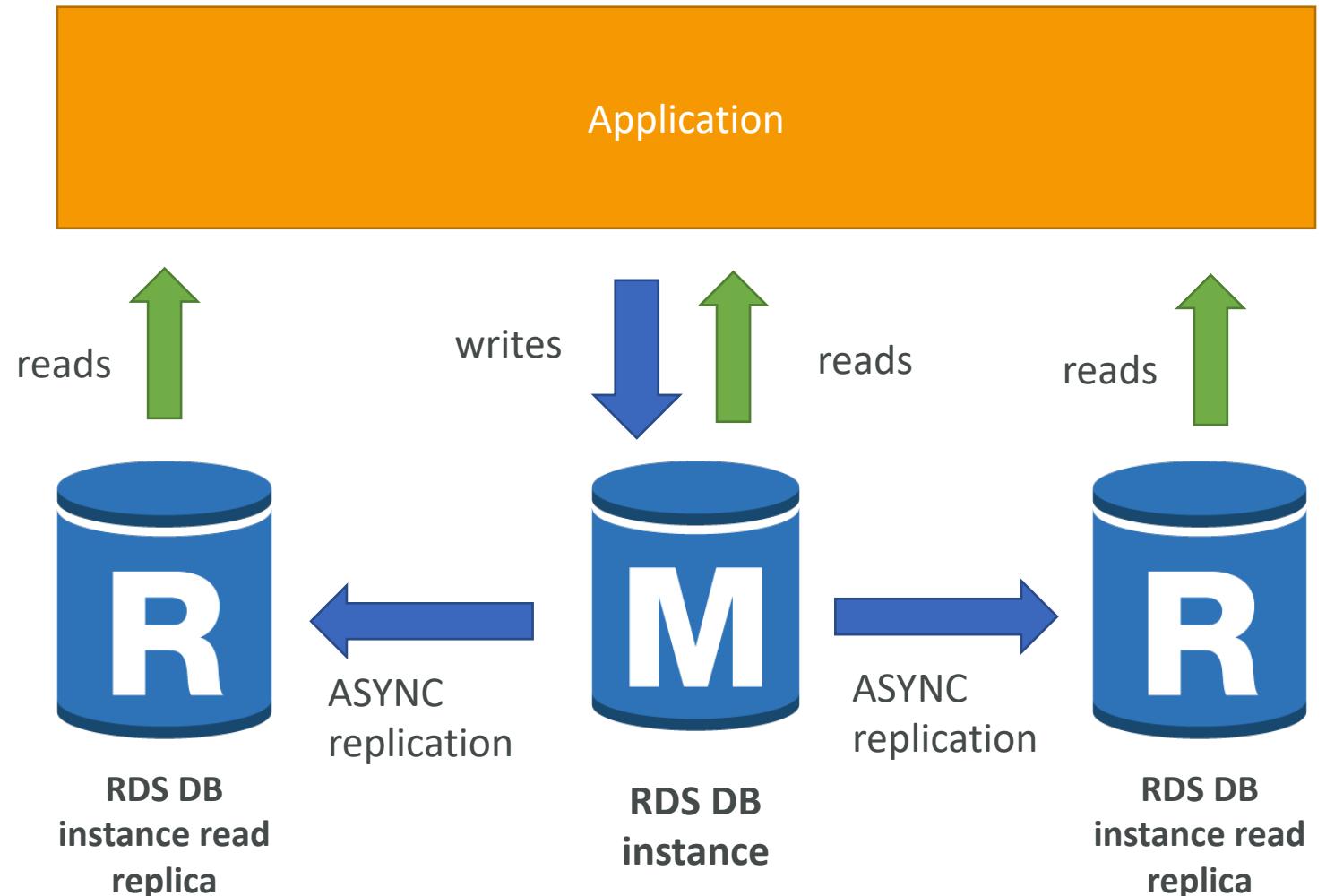
RDS – Storage Auto Scaling

- Helps you increase storage on your RDS DB instance dynamically
- When RDS detects you are running out of free database storage, it scales automatically
- Avoid manually scaling your database storage
- You have to set **Maximum Storage Threshold** (maximum limit for DB storage)
- Automatically modify storage if:
 - Free storage is less than 10% of allocated storage
 - Low-storage lasts at least 5 minutes
 - 6 hours have passed since last modification
- Useful for applications with **unpredictable workloads**
- Supports all RDS database engines (MariaDB, MySQL, PostgreSQL, SQL Server, Oracle)



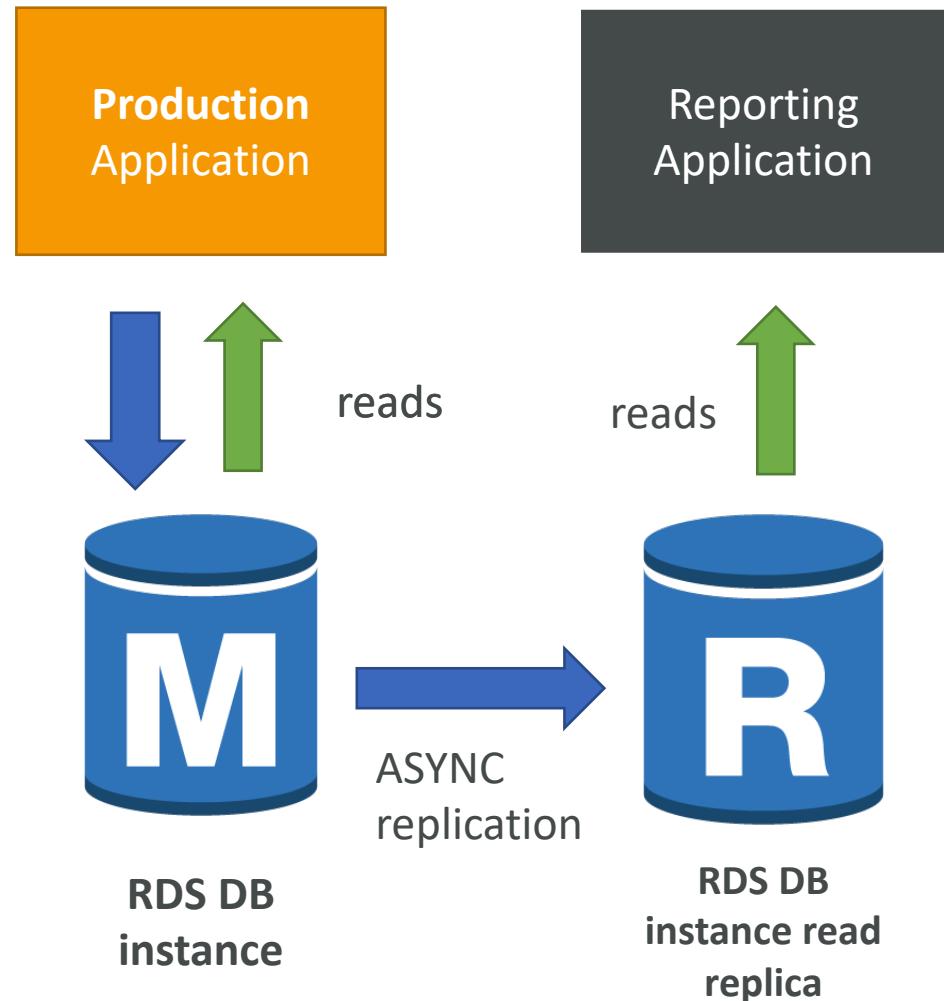
RDS Read Replicas for read scalability

- Up to 15 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is **ASYNC**, so reads are eventually consistent
- Replicas can be promoted to their own DB
- Applications must update the connection string to leverage read replicas



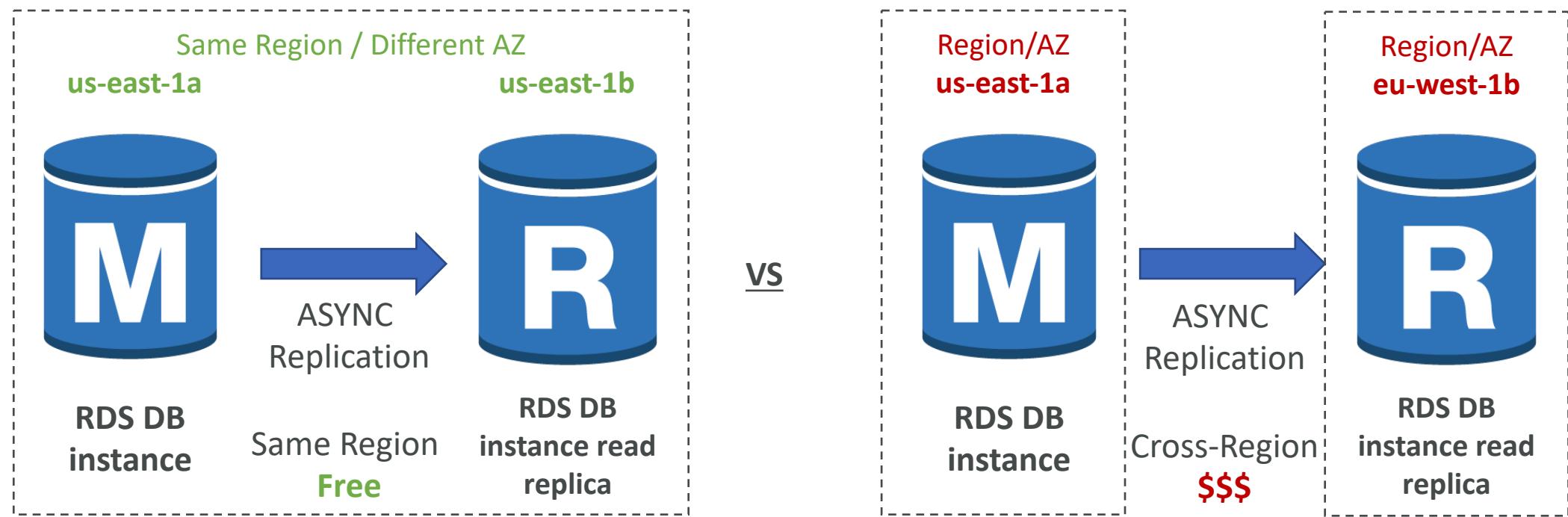
RDS Read Replicas – Use Cases

- You have a production database that is taking on normal load
- You want to run a reporting application to run some analytics
- You create a Read Replica to run the new workload there
- The production application is unaffected
- Read replicas are used for SELECT (=read) only kind of statements (not INSERT, UPDATE, DELETE)



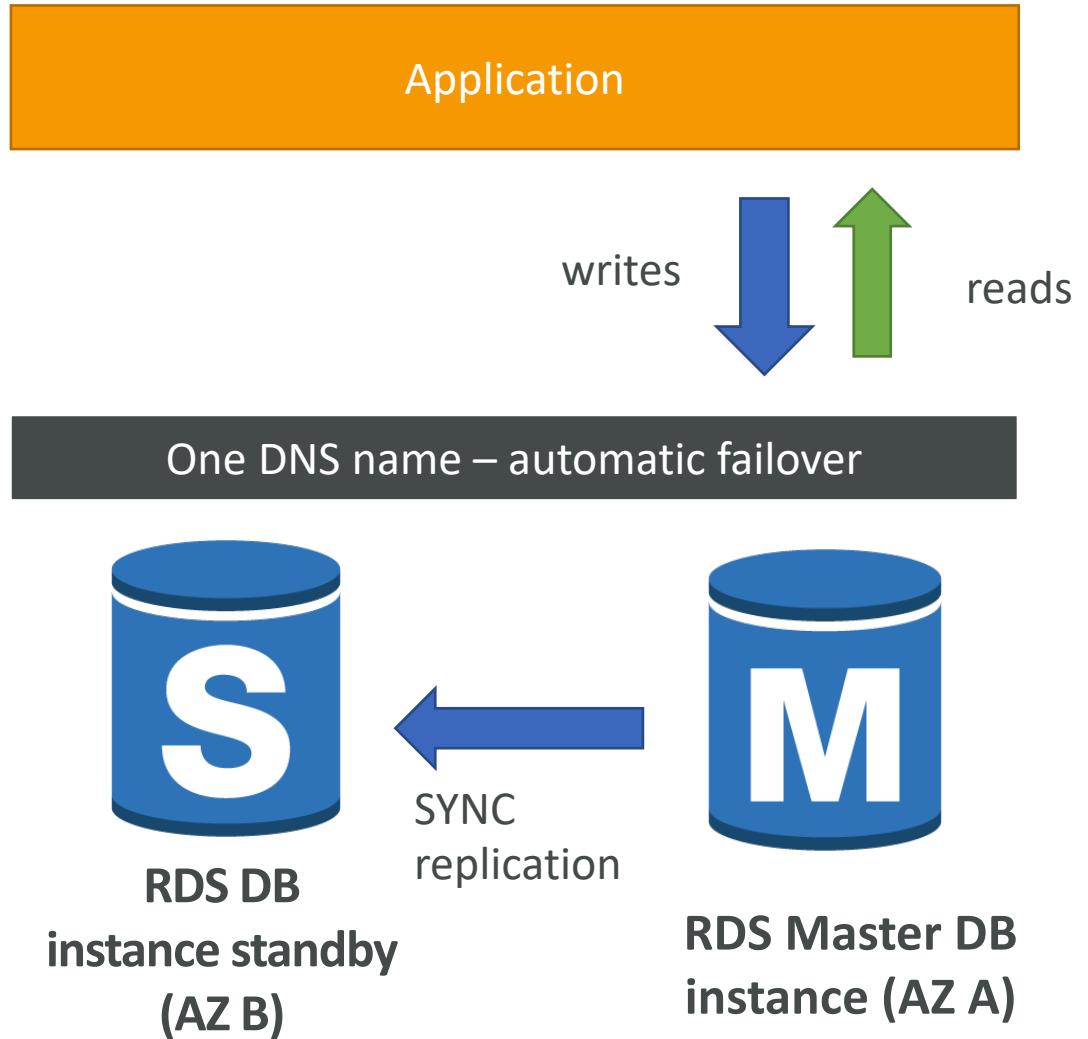
RDS Read Replicas – Network Cost

- In AWS there's a network cost when data goes from one AZ to another
- For RDS Read Replicas within the same region, you don't pay that fee



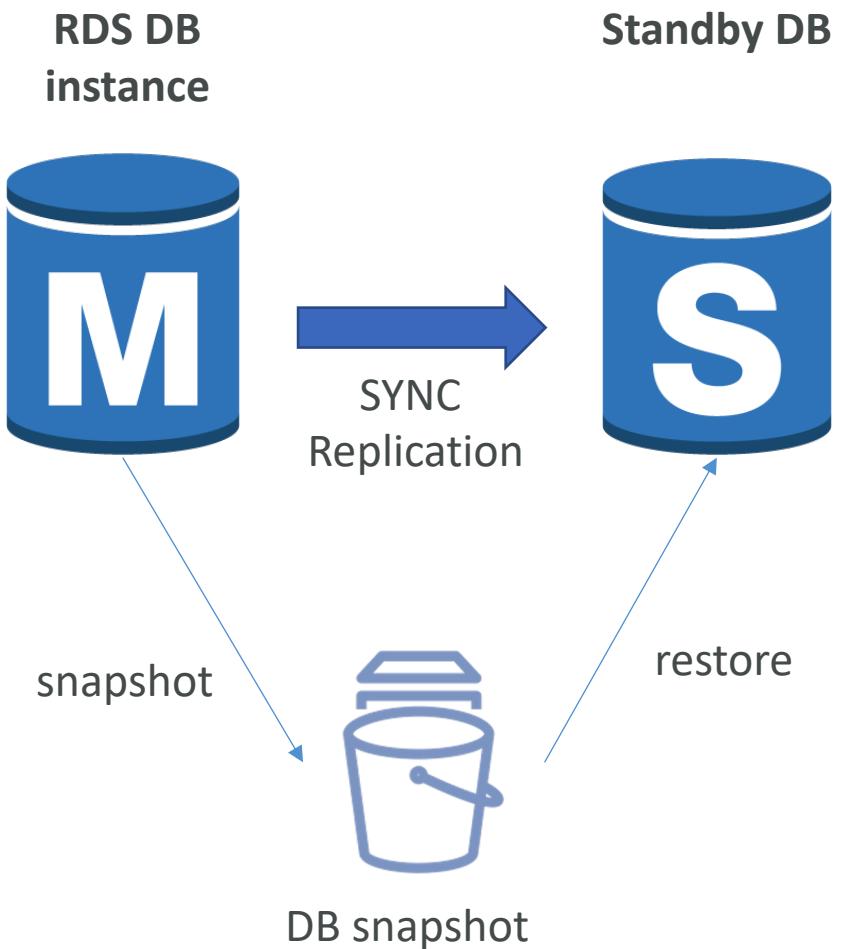
RDS Multi AZ (Disaster Recovery)

- SYNC replication
- One DNS name – automatic app failover to standby
- Increase availability
- Failover in case of loss of AZ, loss of network, instance or storage failure
- No manual intervention in apps
- Not used for scaling
- Note: The Read Replicas be setup as Multi AZ for Disaster Recovery (DR)



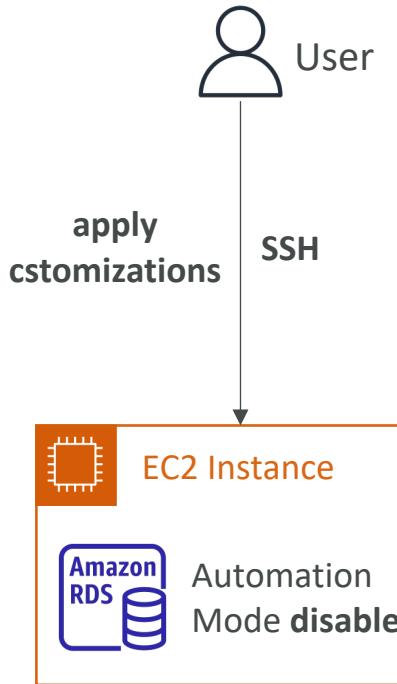
RDS – From Single-AZ to Multi-AZ

- Zero downtime operation (no need to stop the DB)
- Just click on “modify” for the database
- The following happens internally:
 - A snapshot is taken
 - A new DB is restored from the snapshot in a new AZ
 - Synchronization is established between the two databases



RDS Custom

- Managed Oracle and Microsoft SQL Server Database with OS and database customization
- RDS: Automates setup, operation, and scaling of database in AWS
- Custom: access to the underlying database and OS so you can
 - Configure settings
 - Install patches
 - Enable native features
 - Access the underlying EC2 Instance using **SSH** or **SSM Session Manager**
- **De-activate Automation Mode** to perform your customization, better to take a DB snapshot before
- RDS vs. RDS Custom
 - RDS: entire database and the OS to be managed by AWS
 - RDS Custom: full admin access to the underlying OS and the database



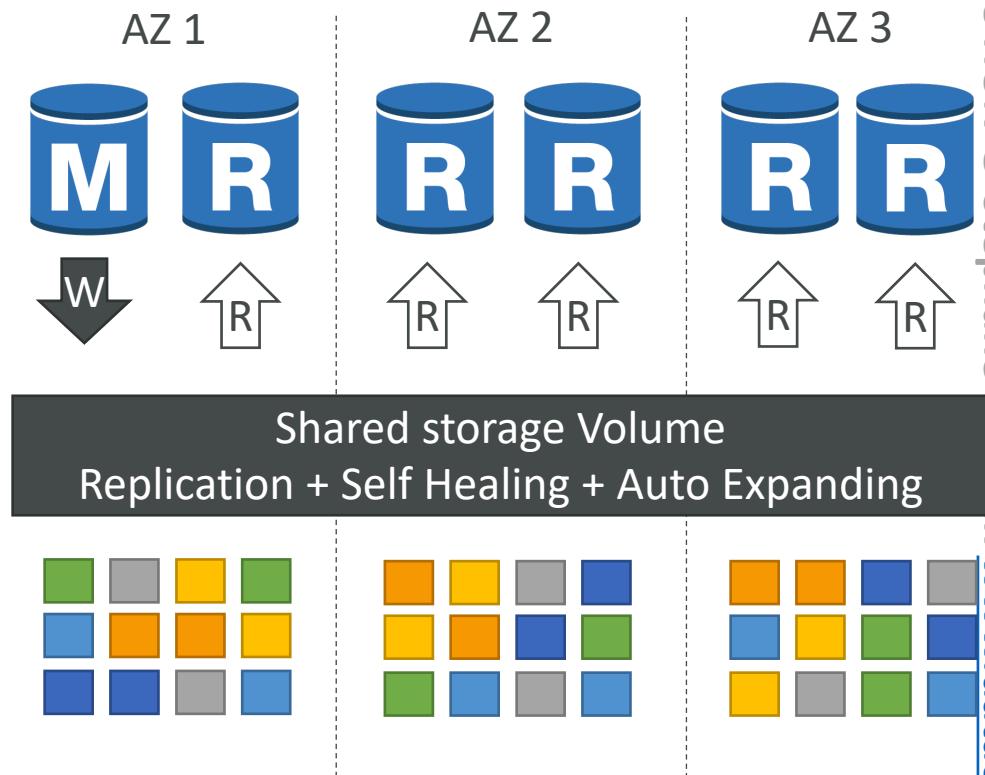


Amazon Aurora

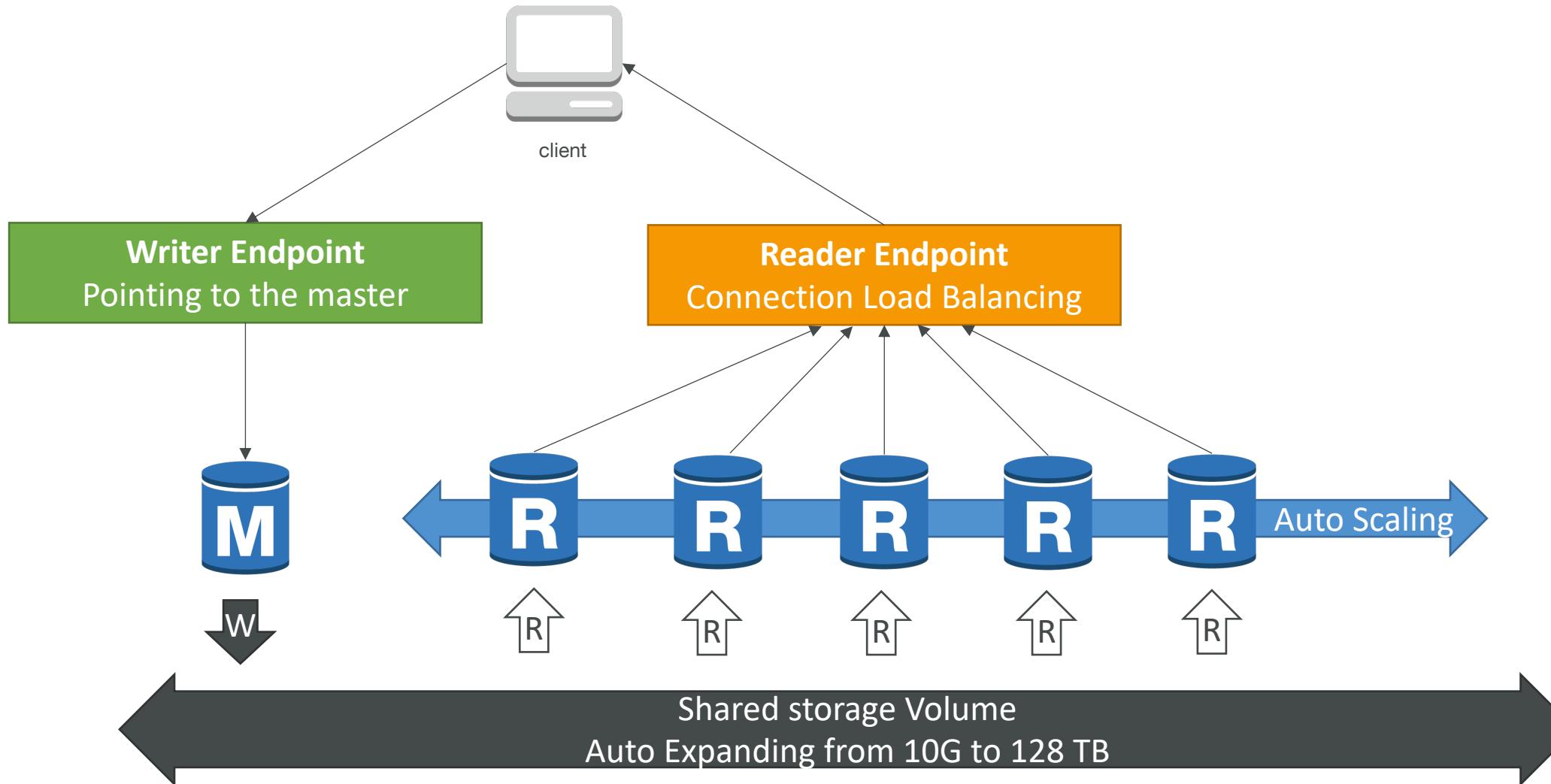
- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is “AWS cloud optimized” and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 128 TB.
- Aurora can have up to 15 replicas and the replication process is faster than MySQL (sub 10 ms replica lag)
- Failover in Aurora is instantaneous. It’s HA (High Availability) native.
- Aurora costs more than RDS (20% more) – but is more efficient

Aurora High Availability and Read Scaling

- 6 copies of your data across 3 AZ:
 - 4 copies out of 6 needed for writes
 - 3 copies out of 6 need for reads
 - Self healing with peer-to-peer replication
 - Storage is striped across 100s of volumes
- One Aurora Instance takes writes (master)
- Automated failover for master in less than 30 seconds
- Master + up to 15 Aurora Read Replicas serve reads
- Support for Cross Region Replication



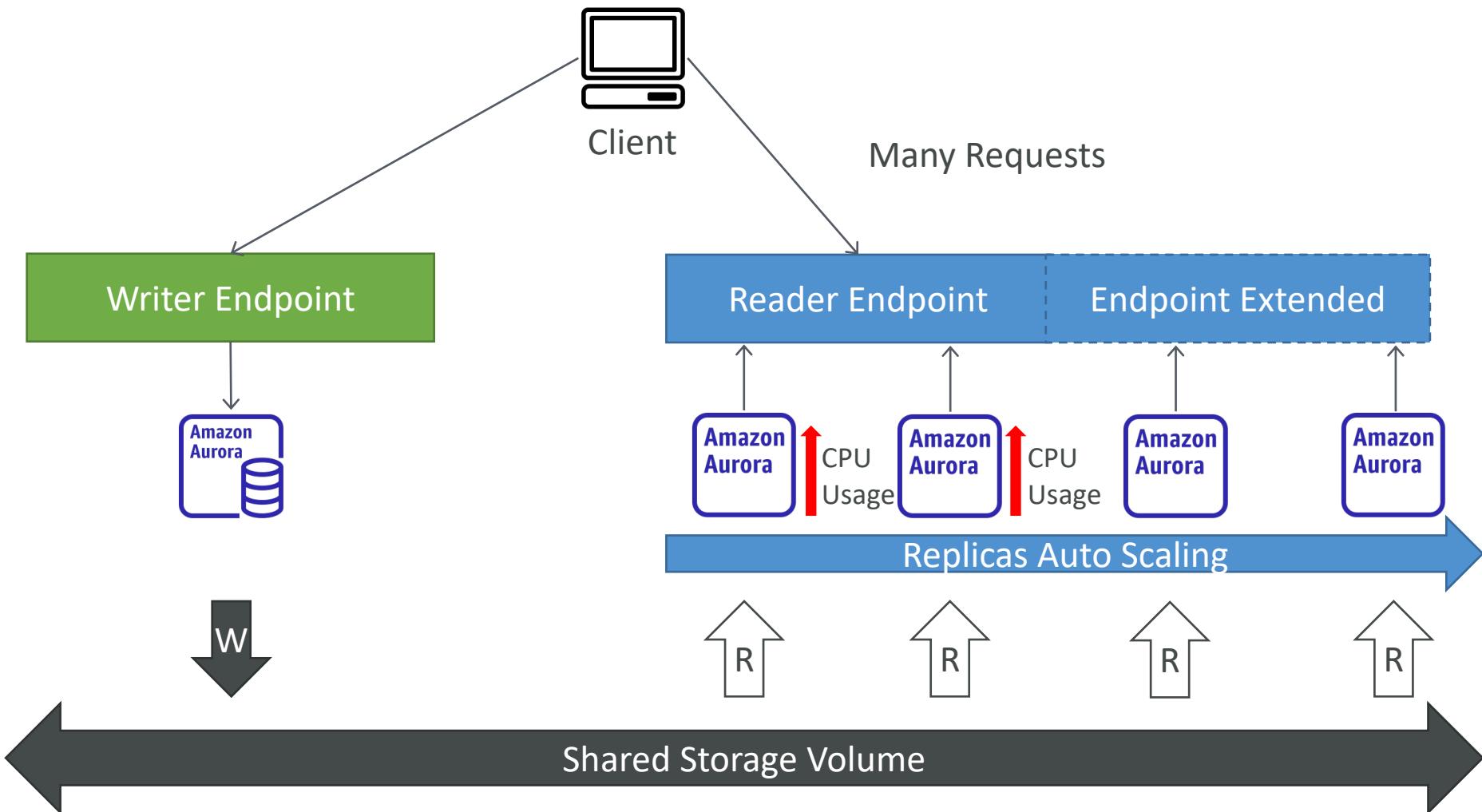
Aurora DB Cluster



Features of Aurora

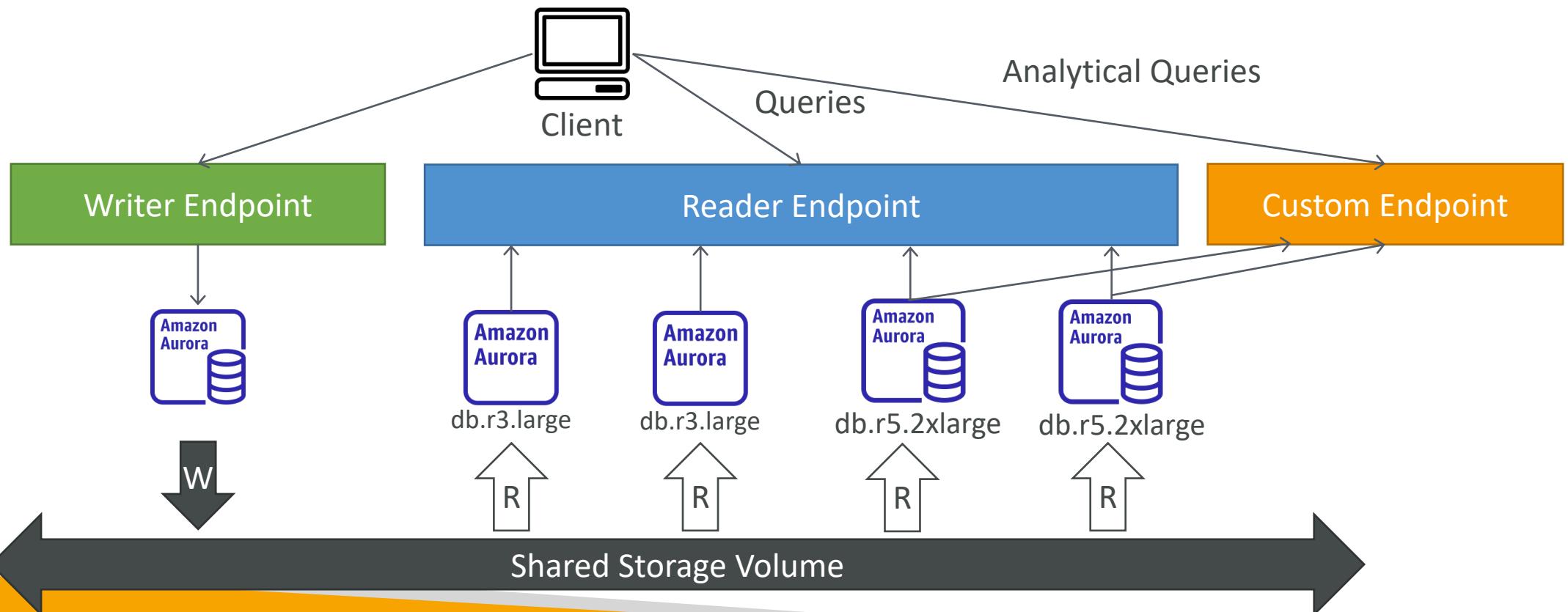
- Automatic fail-over
- Backup and Recovery
- Isolation and security
- Industry compliance
- Push-button scaling
- Automated Patching with Zero Downtime
- Advanced Monitoring
- Routine Maintenance
- Backtrack: restore data at any point of time without using backups

Aurora Replicas - Auto Scaling



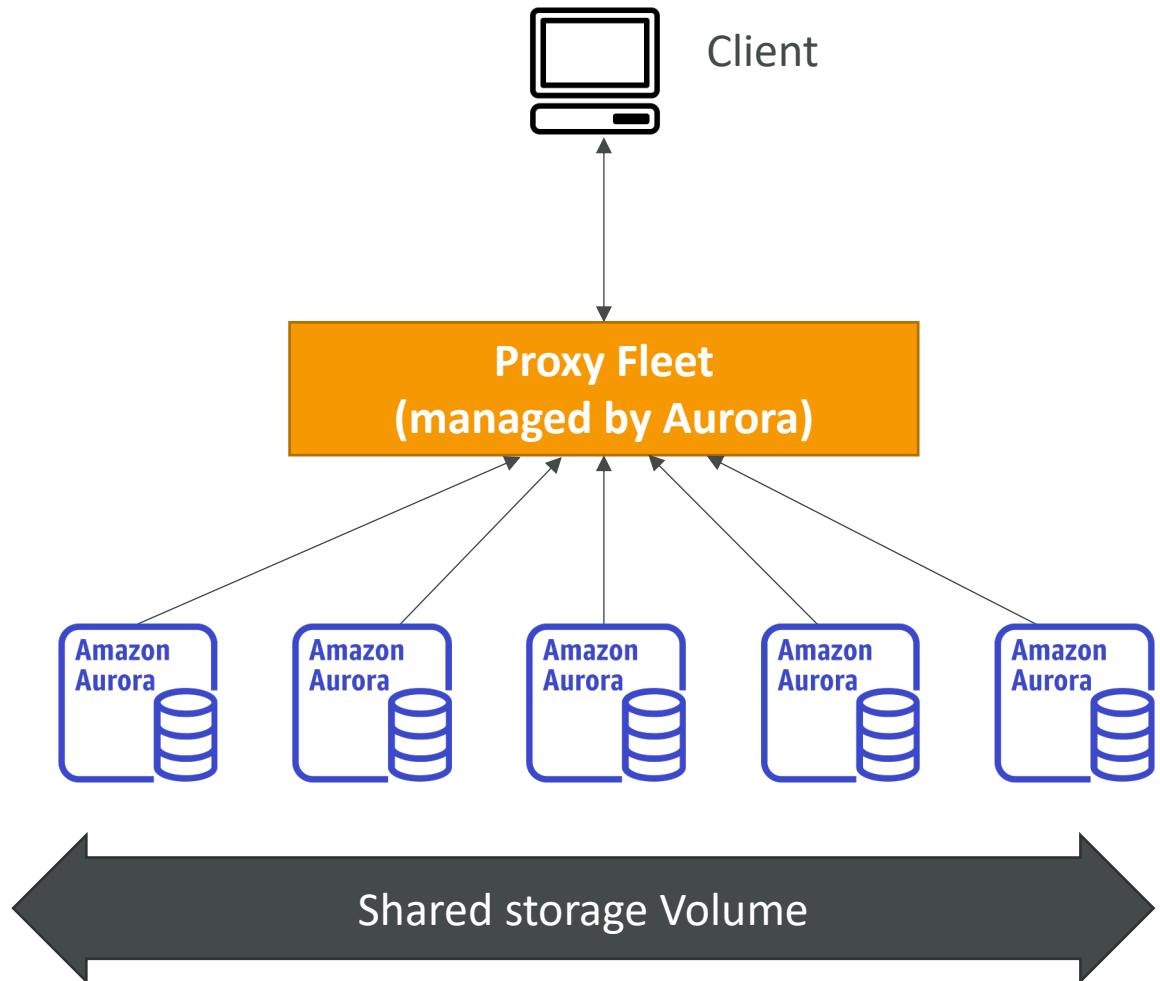
Aurora – Custom Endpoints

- Define a subset of Aurora Instances as a Custom Endpoint
- Example: Run analytical queries on specific replicas
- The Reader Endpoint is generally not used after defining Custom Endpoints



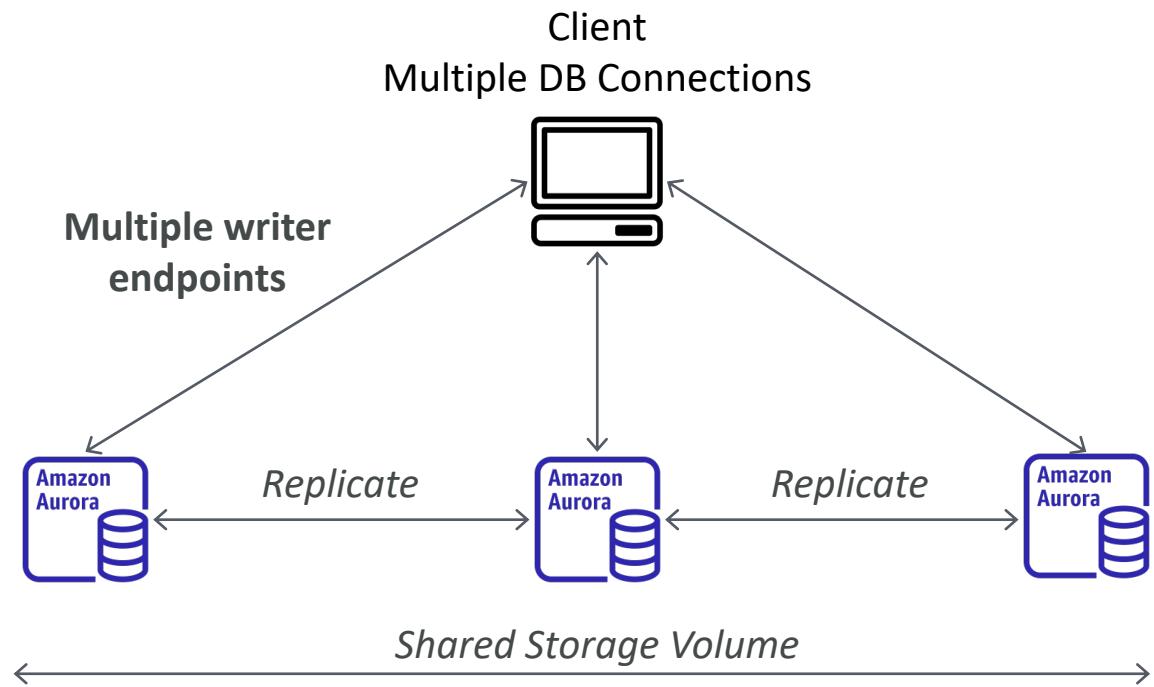
Aurora Serverless

- Automated database instantiation and auto-scaling based on actual usage
- Good for infrequent, intermittent or unpredictable workloads
- No capacity planning needed
- Pay per second, can be more cost-effective



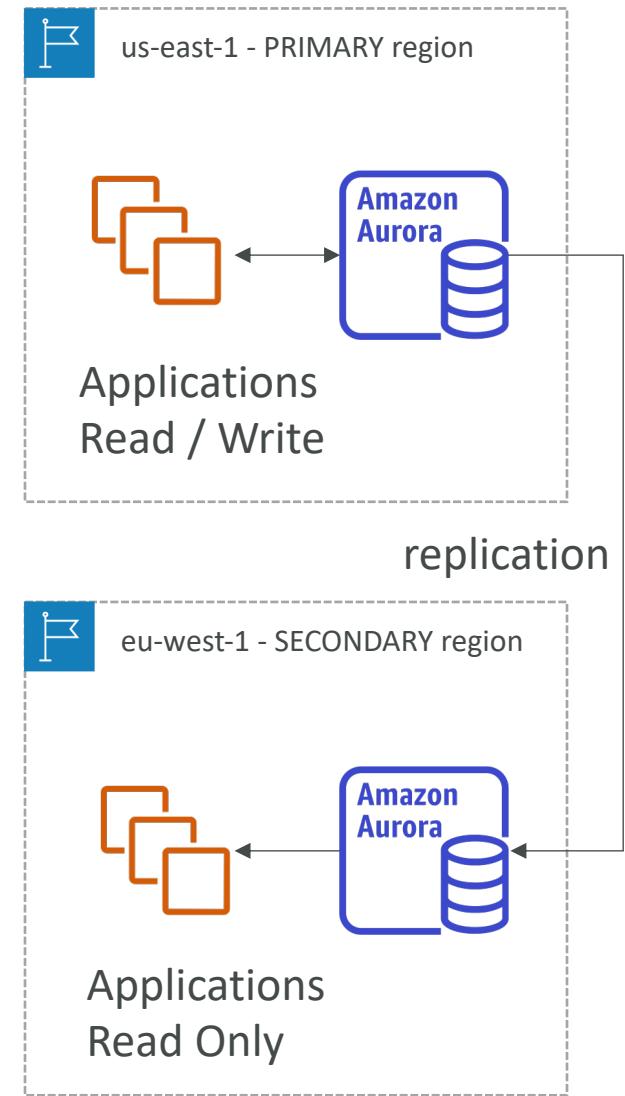
Aurora Multi-Master

- In case you want **continuous write availability** for the writer nodes
- Every node does R/W - vs promoting a Read Replica as the new master



Global Aurora

- Aurora Cross Region Read Replicas:
 - Useful for disaster recovery
 - Simple to put in place
- Aurora Global Database (recommended):
 - 1 Primary Region (read / write)
 - Up to 5 secondary (read-only) regions, replication lag is less than 1 second
 - Up to 16 Read Replicas per secondary region
 - Helps for decreasing latency
 - Promoting another region (for disaster recovery) has an RTO of < 1 minute
 - Typical cross-region replication takes less than 1 second

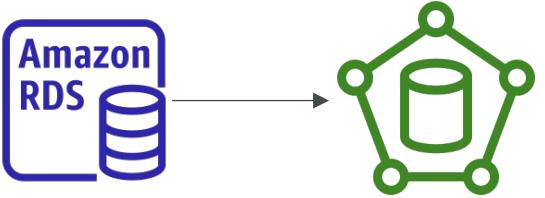


Aurora Machine Learning

- Enables you to add ML-based predictions to your applications via SQL
- Simple, optimized, and secure integration between Aurora and AWS ML services
- Supported services
 - Amazon SageMaker (use with any ML model)
 - Amazon Comprehend (for sentiment analysis)
- You don't need to have ML experience
- Use cases: fraud detection, ads targeting, sentiment analysis, product recommendations

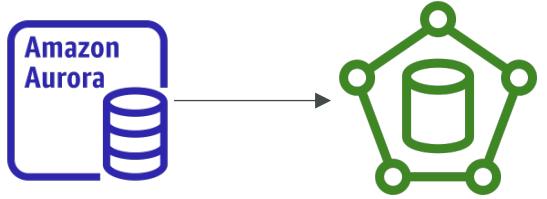


RDS Backups



- Automated backups:
 - Daily full backup of the database (during the backup window)
 - Transaction logs are backed-up by RDS every 5 minutes
 - => ability to restore to any point in time (from oldest backup to 5 minutes ago)
 - 1 to 35 days of retention, set 0 to disable automated backups
- Manual DB Snapshots
 - Manually triggered by the user
 - Retention of backup for as long as you want
- Trick: in a stopped RDS database, you will still pay for storage. If you plan on stopping it for a long time, you should snapshot & restore instead

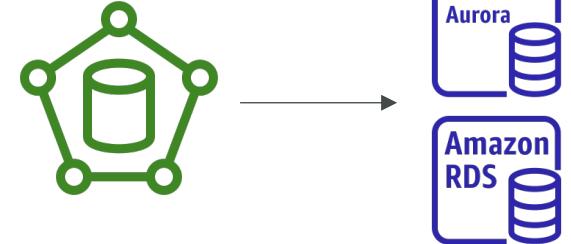
Aurora Backups



- Automated backups
 - 1 to 35 days (cannot be disabled)
 - point-in-time recovery in that timeframe
- Manual DB Snapshots
 - Manually triggered by the user
 - Retention of backup for as long as you want

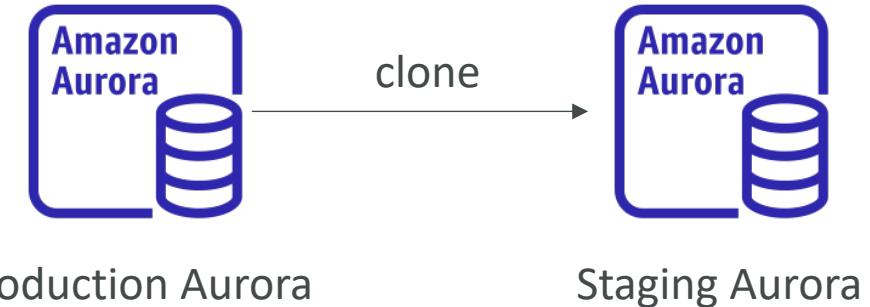
RDS & Aurora Restore options

- Restoring a RDS / Aurora backup or a snapshot creates a new database
- Restoring MySQL RDS database from S3
 - Create a backup of your on-premises database
 - Store it on Amazon S3 (object storage)
 - Restore the backup file onto a new RDS instance running MySQL
- Restoring MySQL Aurora cluster from S3
 - Create a backup of your on-premises database using Percona XtraBackup
 - Store the backup file on Amazon S3
 - Restore the backup file onto a new Aurora cluster running MySQL



Aurora Database Cloning

- Create a new Aurora DB Cluster from an existing one
- Faster than snapshot & restore
- Uses **copy-on-write** protocol
 - Initially, the new DB cluster uses the same data volume as the original DB cluster (fast and efficient – no copying is needed)
 - When updates are made to the new DB cluster data, then additional storage is allocated and data is copied to be separated
- Very fast & cost-effective
- Useful to create a “staging” database from a “production” database without impacting the production database



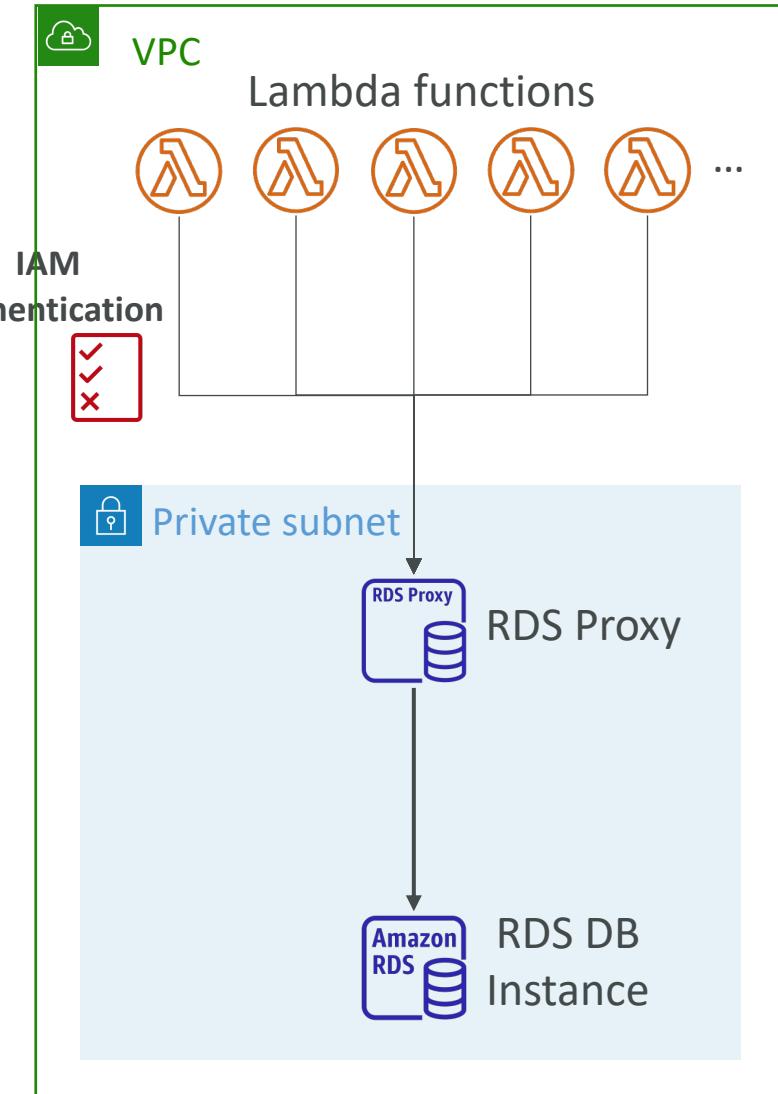
RDS & Aurora Security

- **At-rest encryption:**
 - Database master & replicas encryption using AWS KMS – must be defined as launch time
 - If the master is not encrypted, the read replicas cannot be encrypted
 - To encrypt an un-encrypted database, go through a DB snapshot & restore as encrypted
- **In-flight encryption:** TLS-ready by default, use the AWS TLS root certificates client-side
- **IAM Authentication:** IAM roles to connect to your database (instead of username/pw)
- **Security Groups:** Control Network access to your RDS / Aurora DB
- **No SSH available** except on RDS Custom
- **Audit Logs can be enabled** and sent to CloudWatch Logs for longer retention

Amazon RDS Proxy



- Fully managed database proxy for RDS
- Allows apps to pool and share DB connections established with the database
- Improving database efficiency by reducing the stress on database resources (e.g., CPU, RAM) and minimize open connections (and timeouts)
- Serverless, autoscaling, highly available (multi-AZ)
- Reduced RDS & Aurora failover time by up 66%
- Supports RDS (MySQL, PostgreSQL, MariaDB, MS SQL Server) and Aurora (MySQL, PostgreSQL)
- No code changes required for most apps
- Enforce IAM Authentication for DB, and securely store credentials in AWS Secrets Manager
- RDS Proxy is never publicly accessible (must be accessed from VPC)





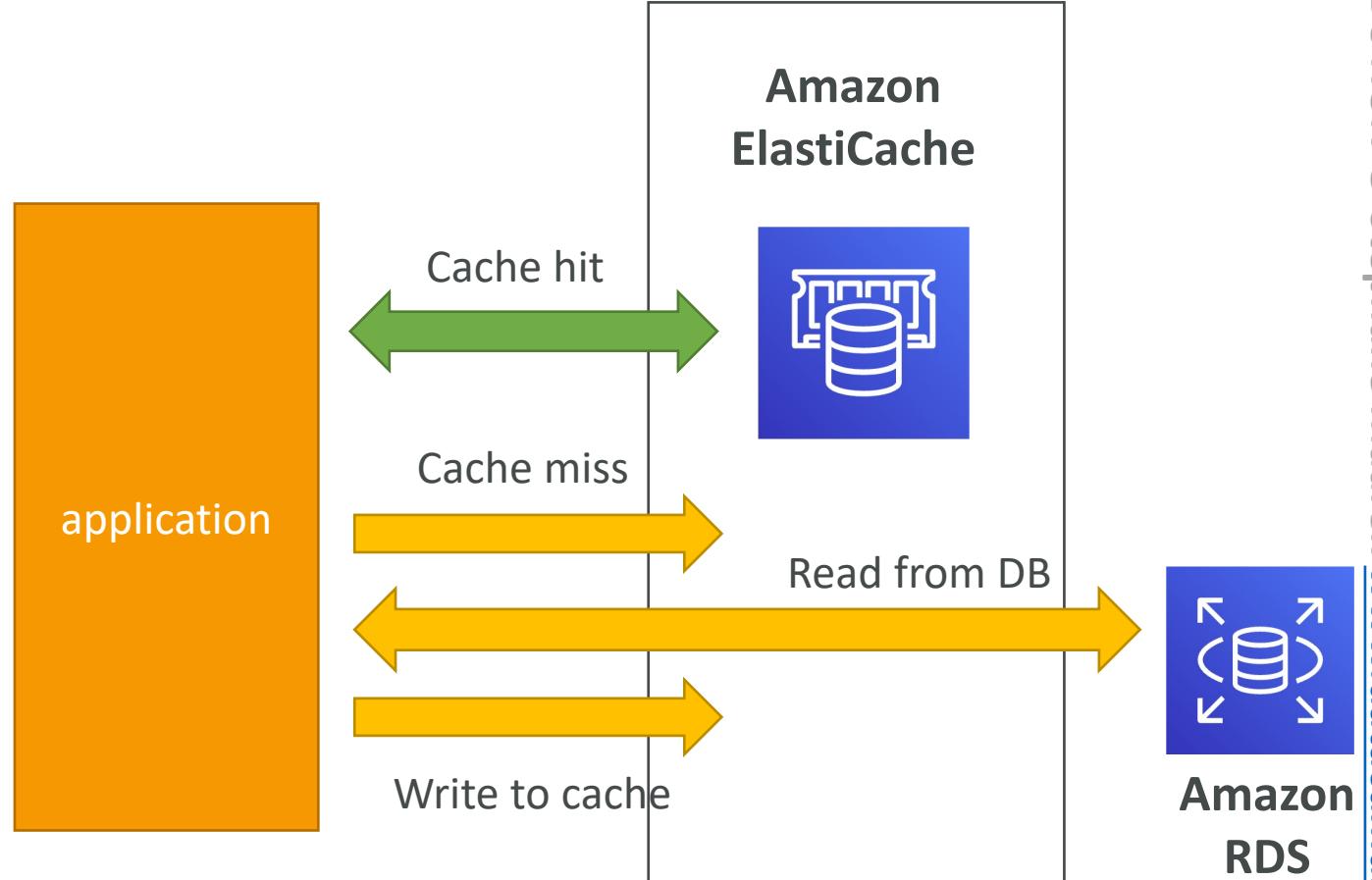
Amazon ElastiCache Overview

- The same way RDS is to get managed Relational Databases...
- ElastiCache is to get managed Redis or Memcached
- Caches are in-memory databases with really high performance, low latency
- Helps reduce load off of databases for read intensive workloads
- Helps make your application stateless
- AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups
- Using ElastiCache involves heavy application code changes

ElastiCache

Solution Architecture - DB Cache

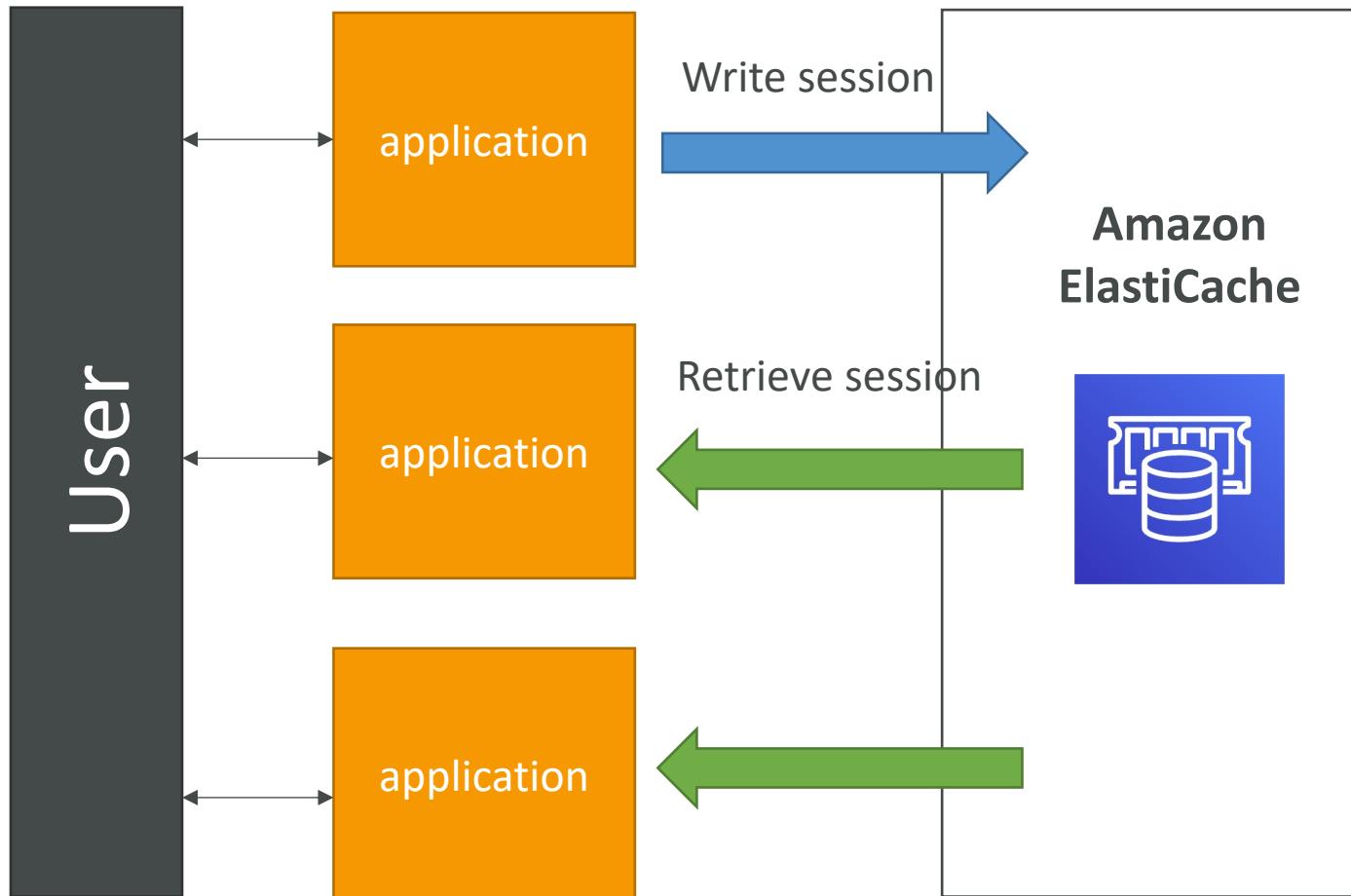
- Applications queries ElastiCache, if not available, get from RDS and store in ElastiCache.
- Helps relieve load in RDS
- Cache must have an invalidation strategy to make sure only the most current data is used in there.



ElastiCache

Solution Architecture – User Session Store

- User logs into any of the application
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in



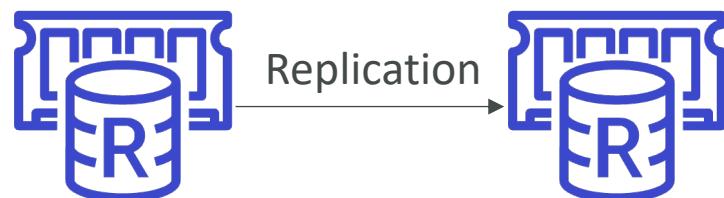
ElastiCache – Redis vs Memcached

REDIS

- Multi AZ with Auto-Failover
- Read Replicas to scale reads and have high availability
- Data Durability using AOF persistence
- Backup and restore features
- Supports Sets and Sorted Sets

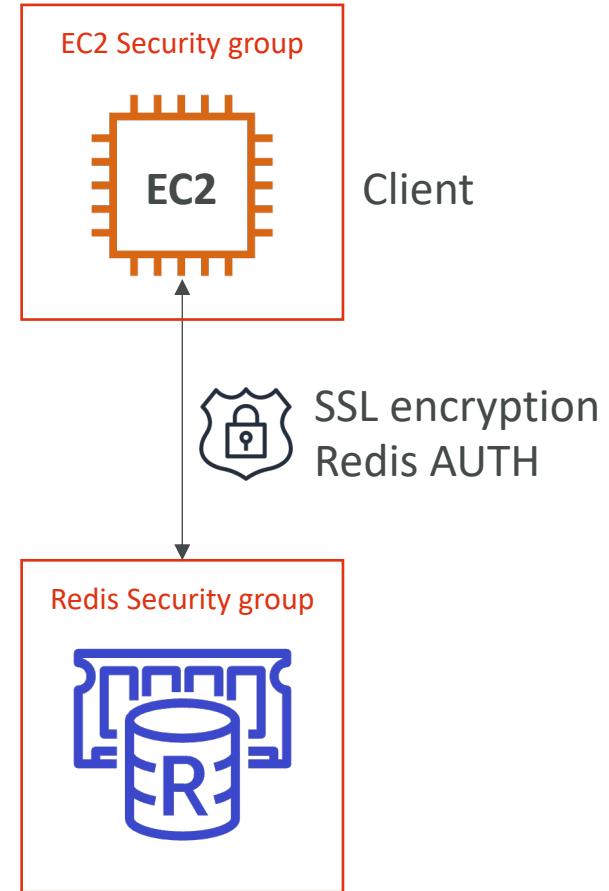
MEMCACHED

- Multi-node for partitioning of data (sharding)
- No high availability (replication)
- Non persistent
- No backup and restore
- Multi-threaded architecture



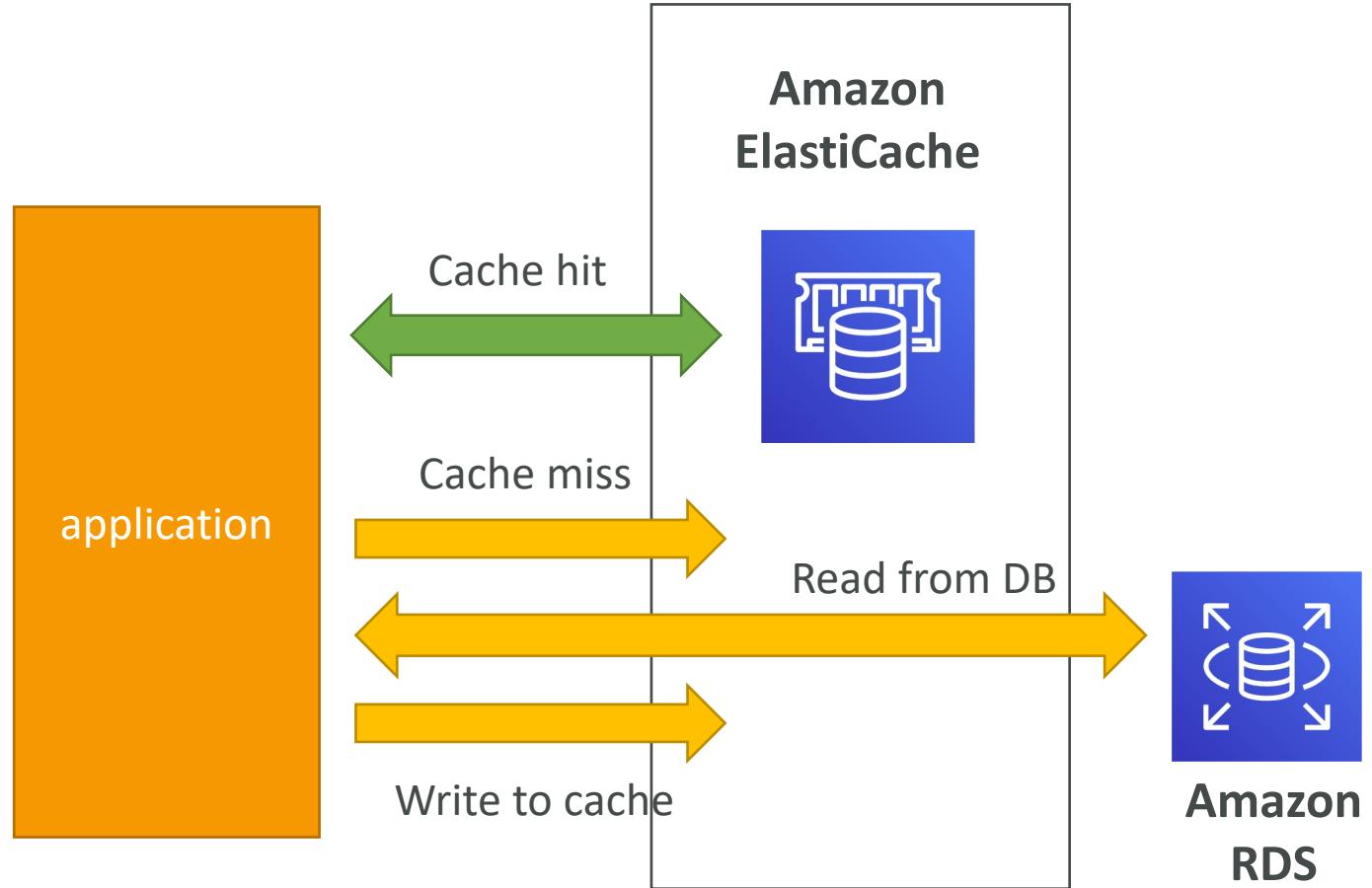
ElastiCache – Cache Security

- ElastiCache supports **IAM Authentication** for Redis
- IAM policies on ElastiCache are only used for AWS API-level security
- **Redis AUTH**
 - You can set a “password/token” when you create a Redis cluster
 - This is an extra level of security for your cache (on top of security groups)
 - Support SSL in flight encryption
- Memcached
 - Supports SASL-based authentication (advanced)



Patterns for ElastiCache

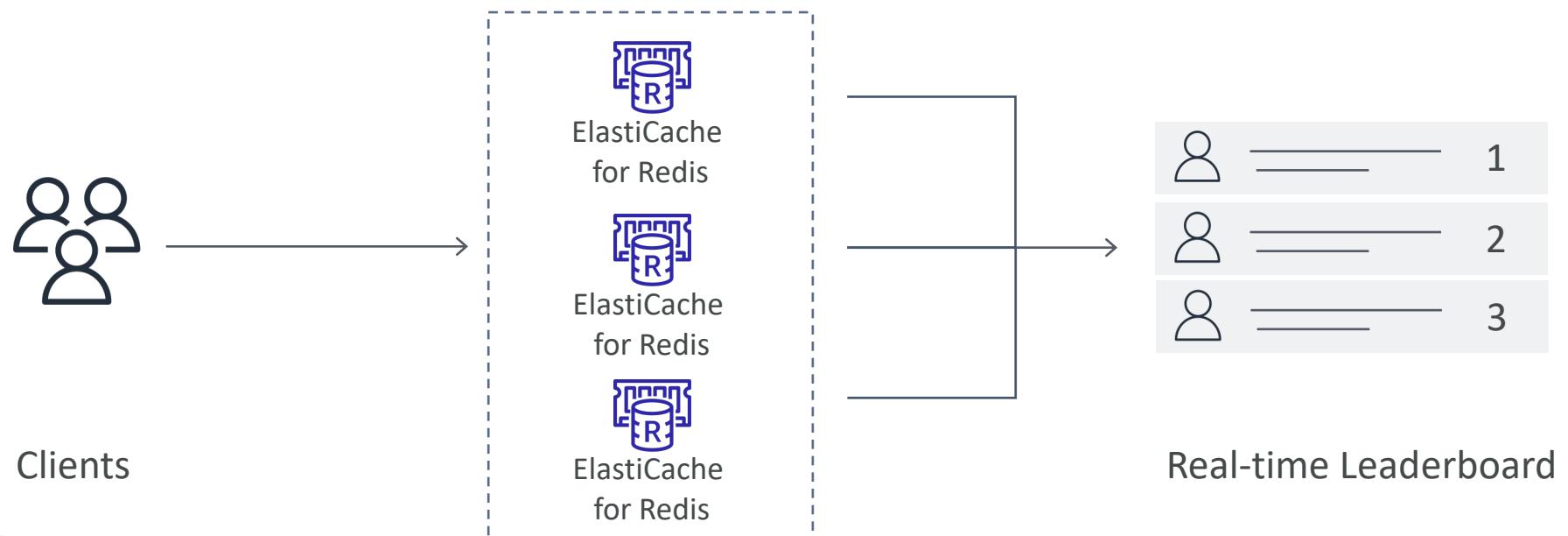
- **Lazy Loading:** all the read data is cached, data can become stale in cache
- **Write Through:** Adds or update data in the cache when written to a DB (no stale data)
- **Session Store:** store temporary session data in a cache (using TTL features)
- **Quote:** There are only two hard things in Computer Science: cache invalidation and naming things



Lazy Loading illustrated

ElastiCache – Redis Use Case

- Gaming Leaderboards are computationally complex
- **Redis Sorted sets** guarantee both uniqueness and element ordering
- Each time a new element added, it's ranked in real time, then added in correct order



Route 53 Section

What is DNS?

- Domain Name System which translates the human friendly hostnames into the machine IP addresses
- www.google.com => 172.217.18.36
- DNS is the backbone of the Internet
- DNS uses hierarchical naming structure

.com

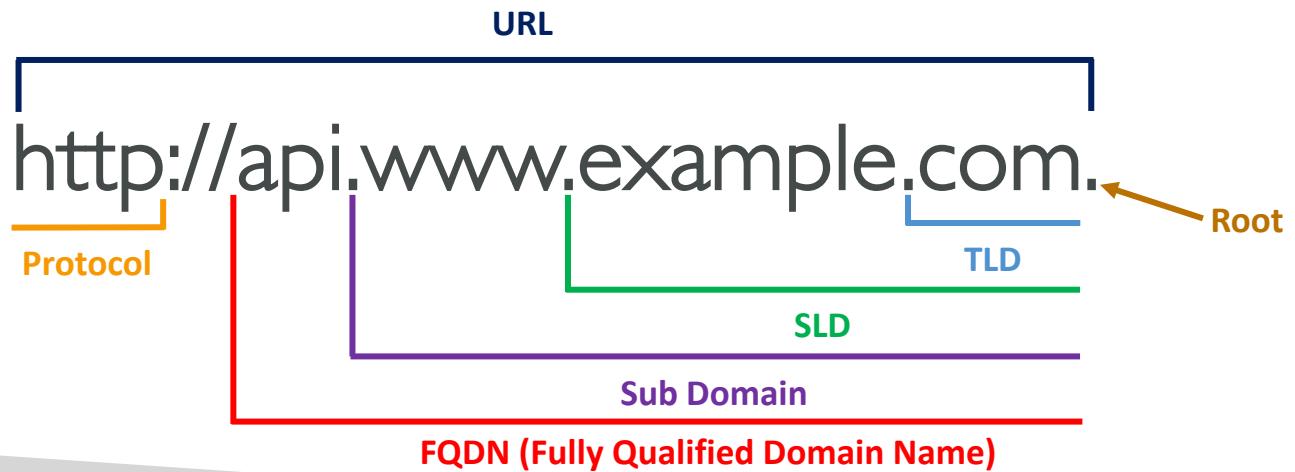
example.com

www.example.com

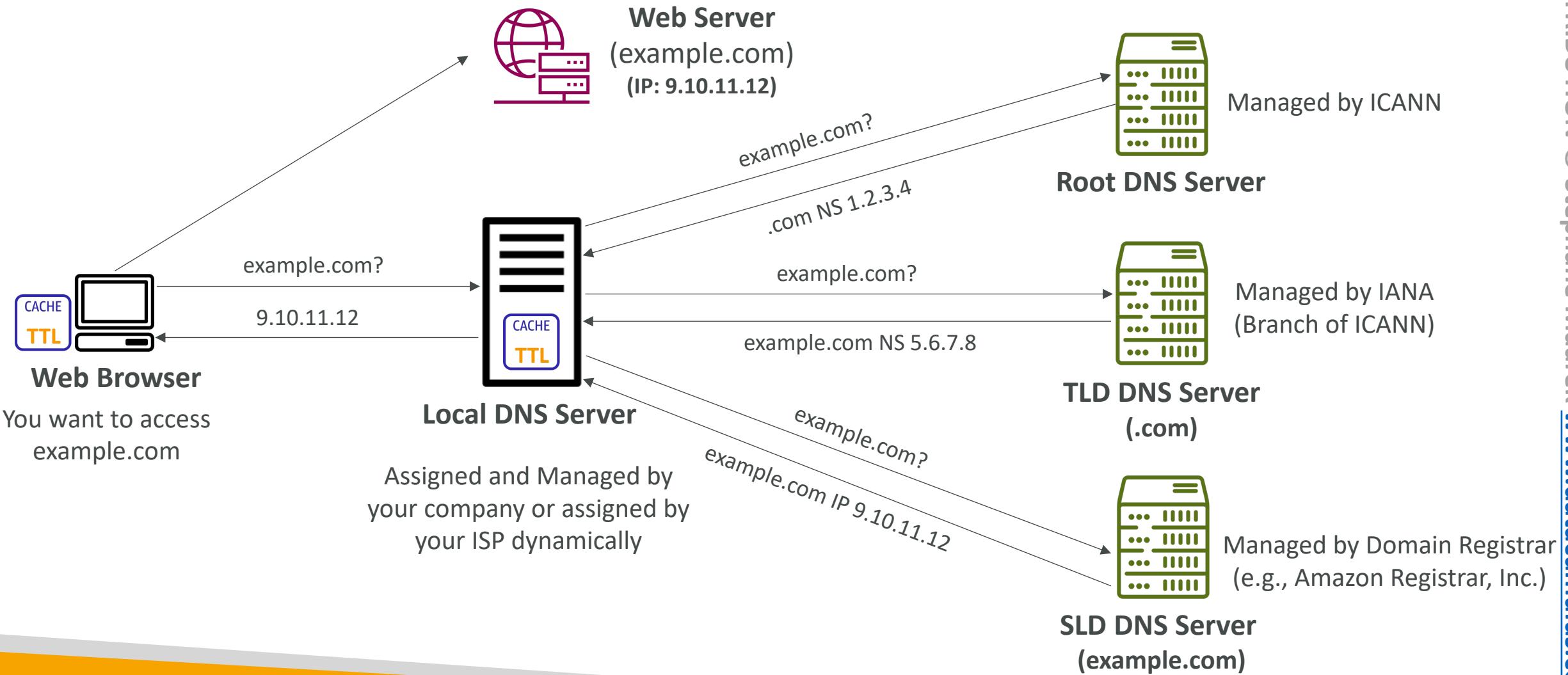
api.example.com

DNS Terminologies

- Domain Registrar: Amazon Route 53, GoDaddy, ...
- DNS Records: A, AAAA, CNAME, NS, ...
- Zone File: contains DNS records
- Name Server: resolves DNS queries (Authoritative or Non-Authoritative)
- Top Level Domain (TLD): .com, .us, .in, .gov, .org, ...
- Second Level Domain (SLD): amazon.com, google.com, ...

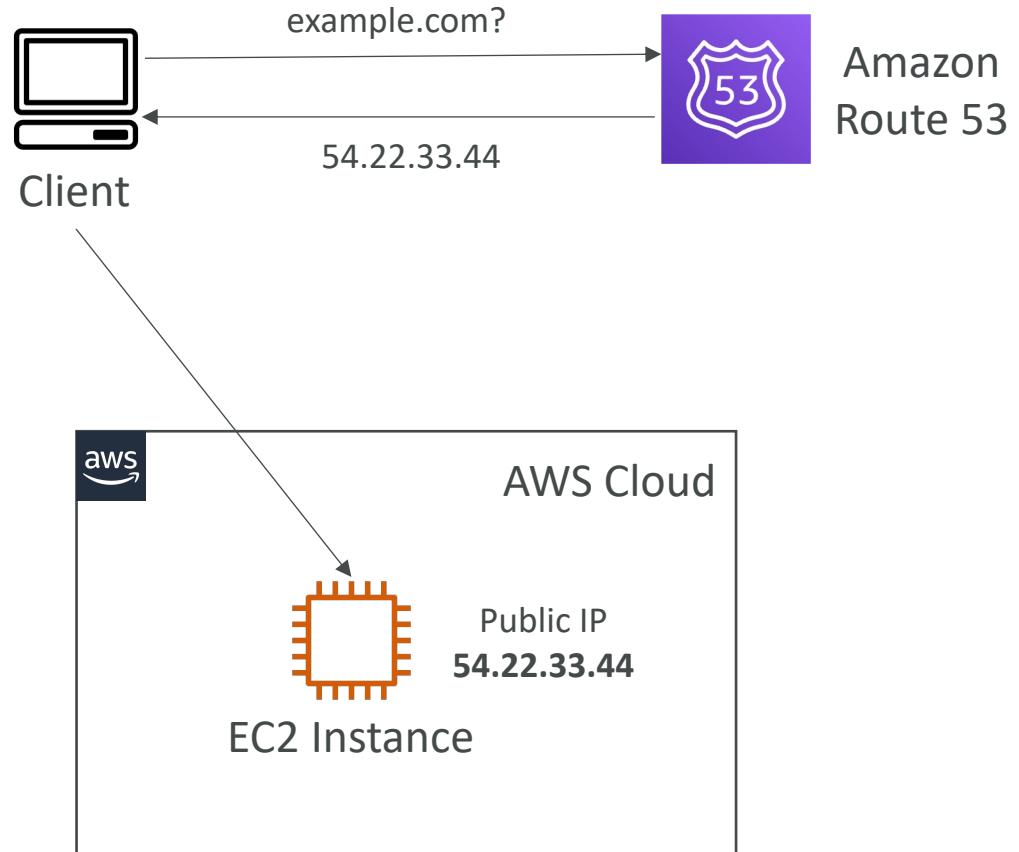


How DNS Works



Amazon Route 53

- A highly available, scalable, fully managed and Authoritative DNS
 - Authoritative = the customer (you) can update the DNS records
- Route 53 is also a Domain Registrar
- Ability to check the health of your resources
- The only AWS service which provides 100% availability SLA
- Why Route 53? 53 is a reference to the traditional DNS port

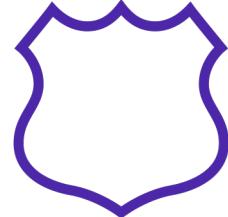


Route 53 – Records

- How you want to route traffic for a domain
- Each record contains:
 - Domain/subdomain Name – e.g., example.com
 - Record Type – e.g., A or AAAA
 - Value – e.g., 12.34.56.78
 - Routing Policy – how Route 53 responds to queries
 - TTL – amount of time the record cached at DNS Resolvers
- Route 53 supports the following DNS record types:
 - (must know) A / AAAA / CNAME / NS
 - (advanced) CAA / DS / MX / NAPTR / PTR / SOA / TXT / SPF / SRV

Route 53 – Record Types

- A – maps a hostname to IPv4
- AAAA – maps a hostname to IPv6
- CNAME – maps a hostname to another hostname
 - The target is a domain name which must have an A or AAAA record
 - Can't create a CNAME record for the top node of a DNS namespace (Zone Apex)
 - Example: you can't create for example.com, but you can create for www.example.com
- NS – Name Servers for the Hosted Zone
 - Control how traffic is routed for a domain

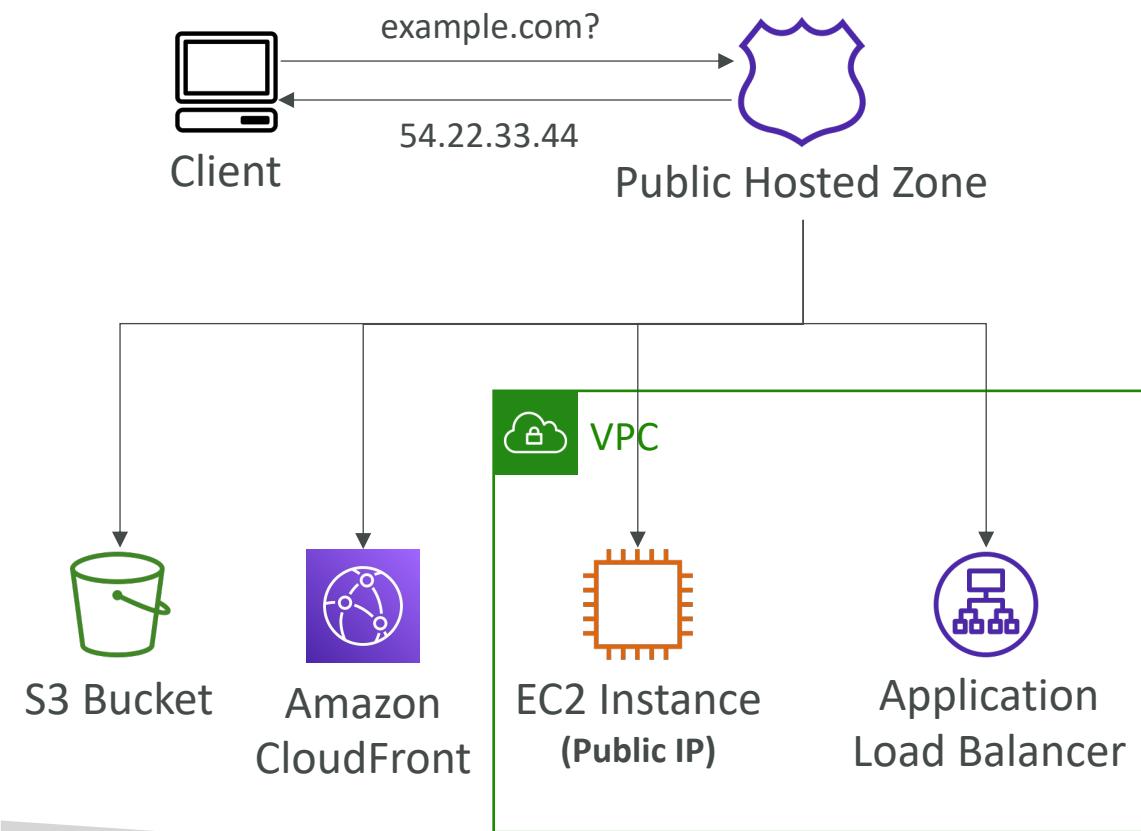


Route 53 – Hosted Zones

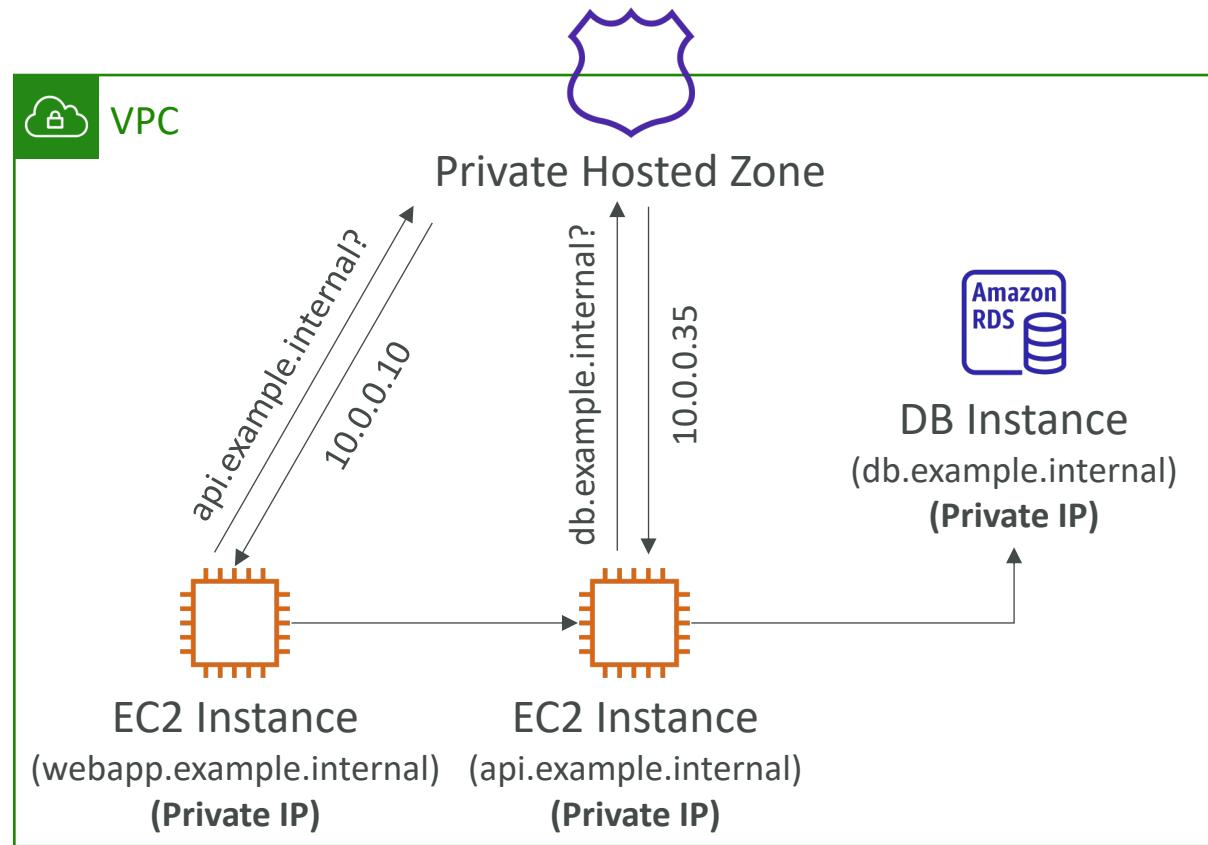
- A container for records that define how to route traffic to a domain and its subdomains
- **Public Hosted Zones** – contains records that specify how to route traffic on the Internet (public domain names)
application1.mypublicdomain.com
- **Private Hosted Zones** – contain records that specify how you route traffic within one or more VPCs (private domain names)
application1.company.internal
- You pay \$0.50 per month per hosted zone

Route 53 – Public vs. Private Hosted Zones

Public Hosted Zone

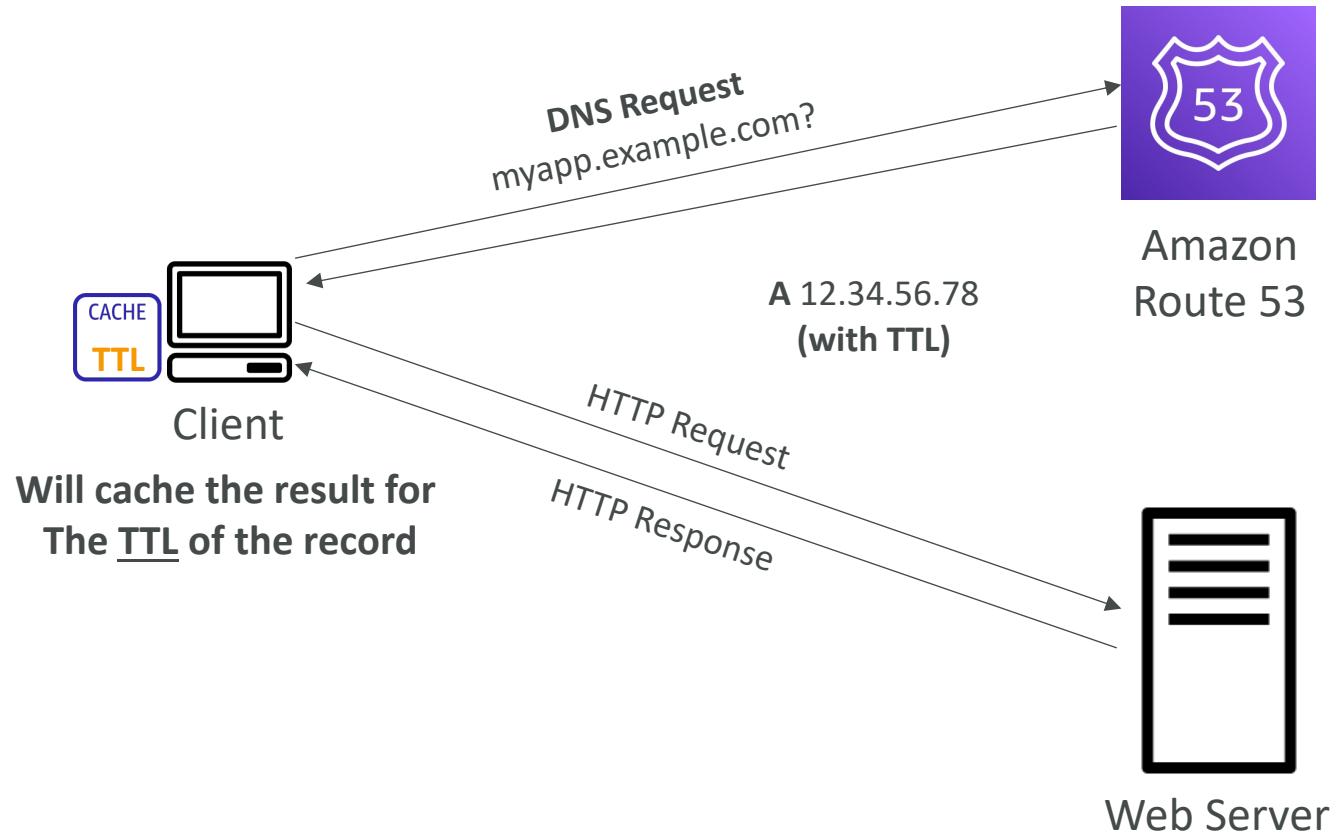


Private Hosted Zone



Route 53 – Records TTL (Time To Live)

- High TTL – e.g., 24 hr
 - Less traffic on Route 53
 - Possibly outdated records
- Low TTL – e.g., 60 sec.
 - More traffic on Route 53 (\$\$)
 - Records are outdated for less time
 - Easy to change records
- Except for Alias records, TTL is mandatory for each DNS record



CNAME vs Alias

- AWS Resources (Load Balancer, CloudFront...) expose an AWS hostname:
 - [lb1-l234.us-east-2.elb.amazonaws.com](https://l1-l234.us-east-2.elb.amazonaws.com) and you want myapp.mydomain.com
- CNAME:
 - Points a hostname to any other hostname. (app.mydomain.com => blabla.anything.com)
 - ONLY FOR NON ROOT DOMAIN (aka. something.mydomain.com)
- Alias:
 - Points a hostname to an AWS Resource (app.mydomain.com => blabla.amazonaws.com)
 - Works for ROOT DOMAIN and NON ROOT DOMAIN (aka mydomain.com)
 - Free of charge
 - Native health check

Route 53 – Alias Records

- Maps a hostname to an AWS resource
- An extension to DNS functionality
- Automatically recognizes changes in the resource's IP addresses
- Unlike CNAME, it can be used for the top node of a DNS namespace (Zone Apex), e.g.: example.com
- Alias Record is always of type A/AAAA for AWS resources (IPv4 / IPv6)
- You can't set the TTL



Route 53 – Alias Records Targets

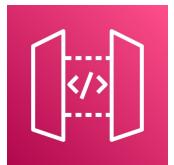
- Elastic Load Balancers
- CloudFront Distributions
- API Gateway
- Elastic Beanstalk environments
- S3 Websites
- VPC Interface Endpoints
- Global Accelerator accelerator
- Route 53 record in the same hosted zone
- You cannot set an ALIAS record for an EC2 DNS name



Elastic
Load Balancer



Amazon
CloudFront



Amazon
API Gateway



Elastic Beanstalk



S3 Websites



VPC Interface
Endpoints



Global Accelerator



Route 53 Record
(same Hosted Zone)

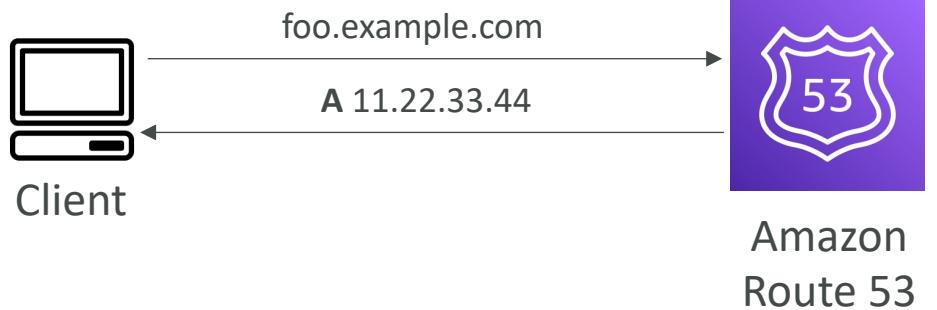
Route 53 – Routing Policies

- Define how Route 53 responds to DNS queries
- Don't get confused by the word "Routing"
 - It's not the same as Load balancer routing which routes the traffic
 - DNS does not route any traffic, it only responds to the DNS queries
- Route 53 Supports the following Routing Policies
 - Simple
 - Weighted
 - Failover
 - Latency based
 - Geolocation
 - Multi-Value Answer
 - Geoproximity (using Route 53 Traffic Flow feature)

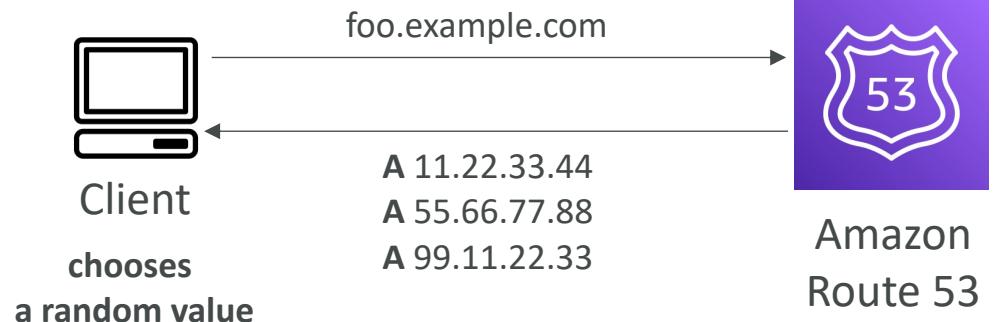
Routing Policies – Simple

- Typically, route traffic to a single resource
- Can specify multiple values in the same record
- If multiple values are returned, a random one is chosen by the client
- When Alias enabled, specify only one AWS resource
- Can't be associated with Health Checks

Single Value

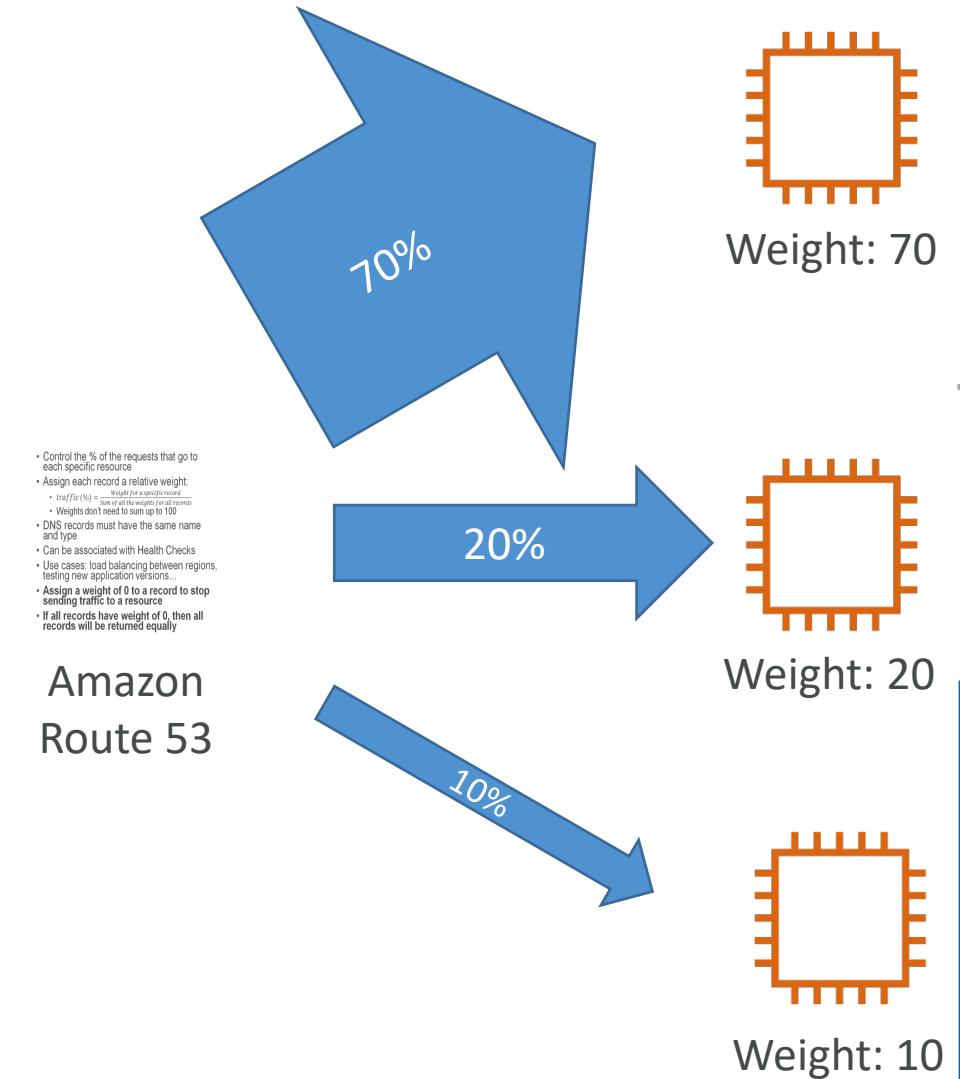


Multiple Value



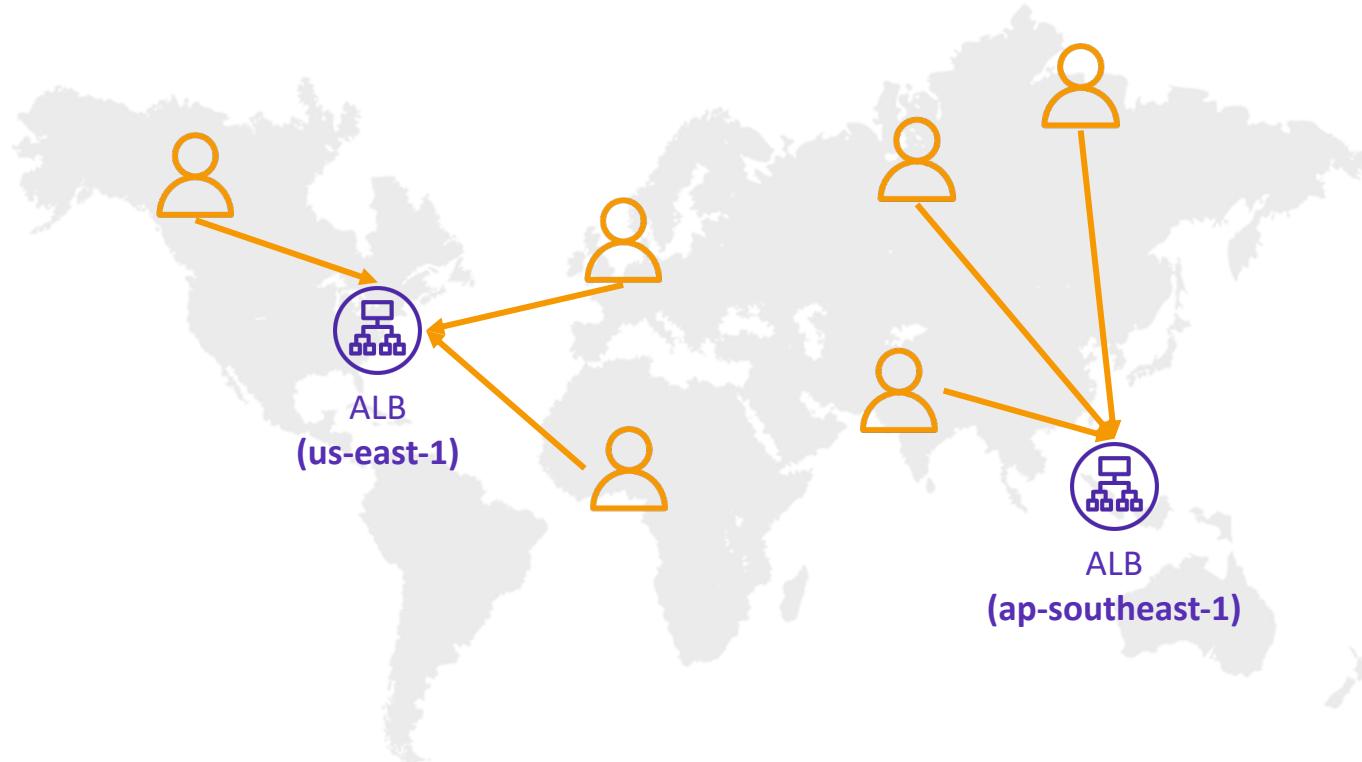
Routing Policies – Weighted

- Control the % of the requests that go to each specific resource
- Assign each record a relative weight:
 - $$\text{traffic (\%)} = \frac{\text{Weight for a specific record}}{\text{Sum of all the weights for all records}}$$
 - Weights don't need to sum up to 100
- DNS records must have the same name and type
- Can be associated with Health Checks
- Use cases: load balancing between regions, testing new application versions...
- Assign a weight of 0 to a record to stop sending traffic to a resource
- If all records have weight of 0, then all records will be returned equally



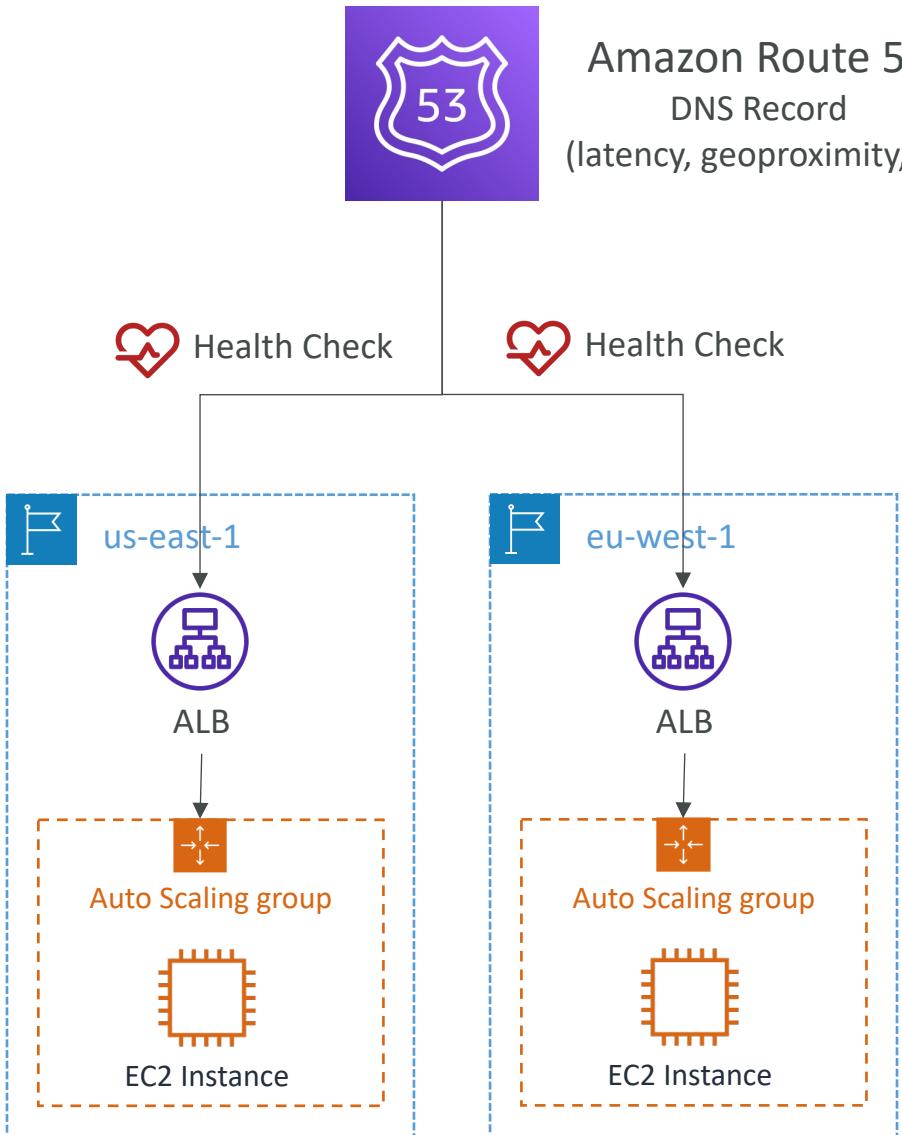
Routing Policies – Latency-based

- Redirect to the resource that has the least latency close to us
- Super helpful when latency for users is a priority
- Latency is based on traffic between users and AWS Regions
- Germany users may be directed to the US (if that's the lowest latency)
- Can be associated with Health Checks (has a failover capability)



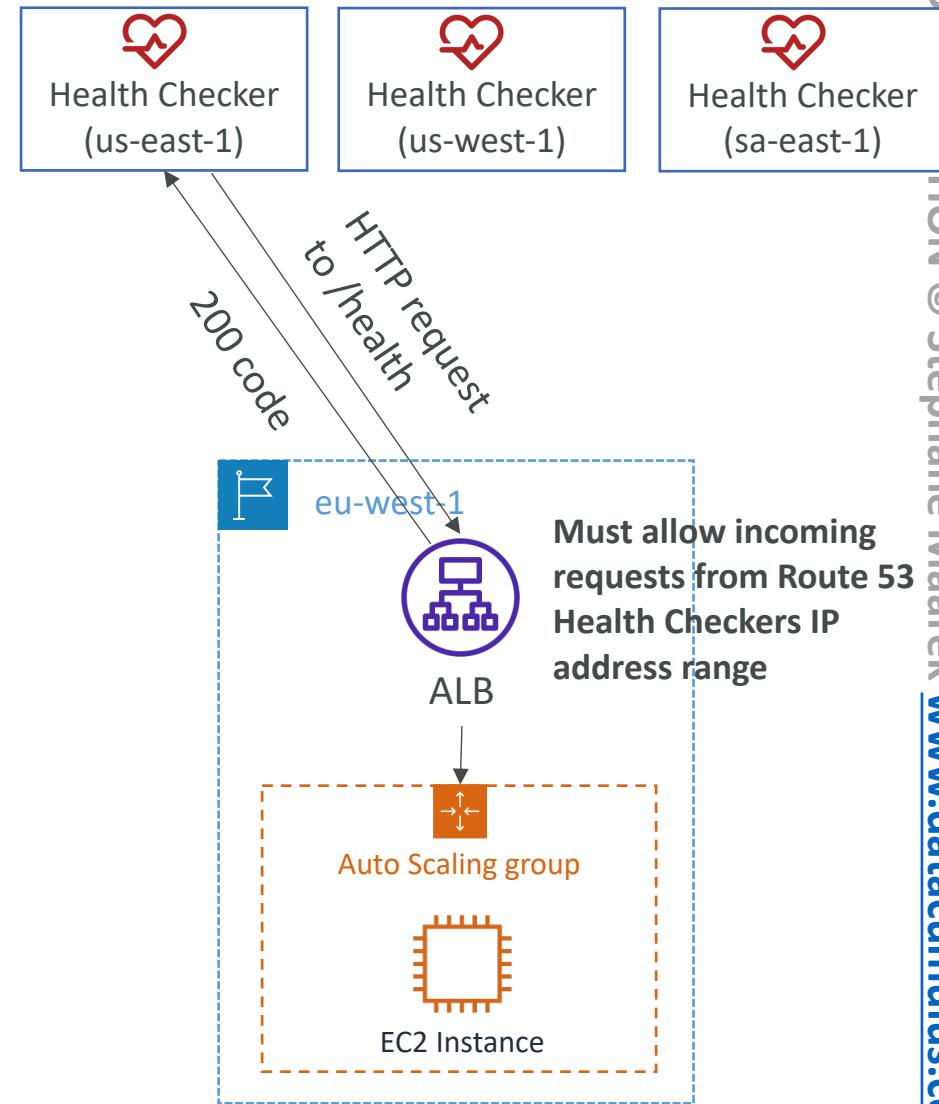
Route 53 – Health Checks

- HTTP Health Checks are only for **public resources**
- Health Check => Automated DNS Failover:
 1. Health checks that monitor an endpoint (application, server, other AWS resource)
 2. Health checks that monitor other health checks (Calculated Health Checks)
 3. Health checks that monitor CloudWatch Alarms (full control !!) – e.g., throttles of DynamoDB, alarms on RDS, custom metrics, ... (helpful for private resources)
- Health Checks are integrated with CW metrics



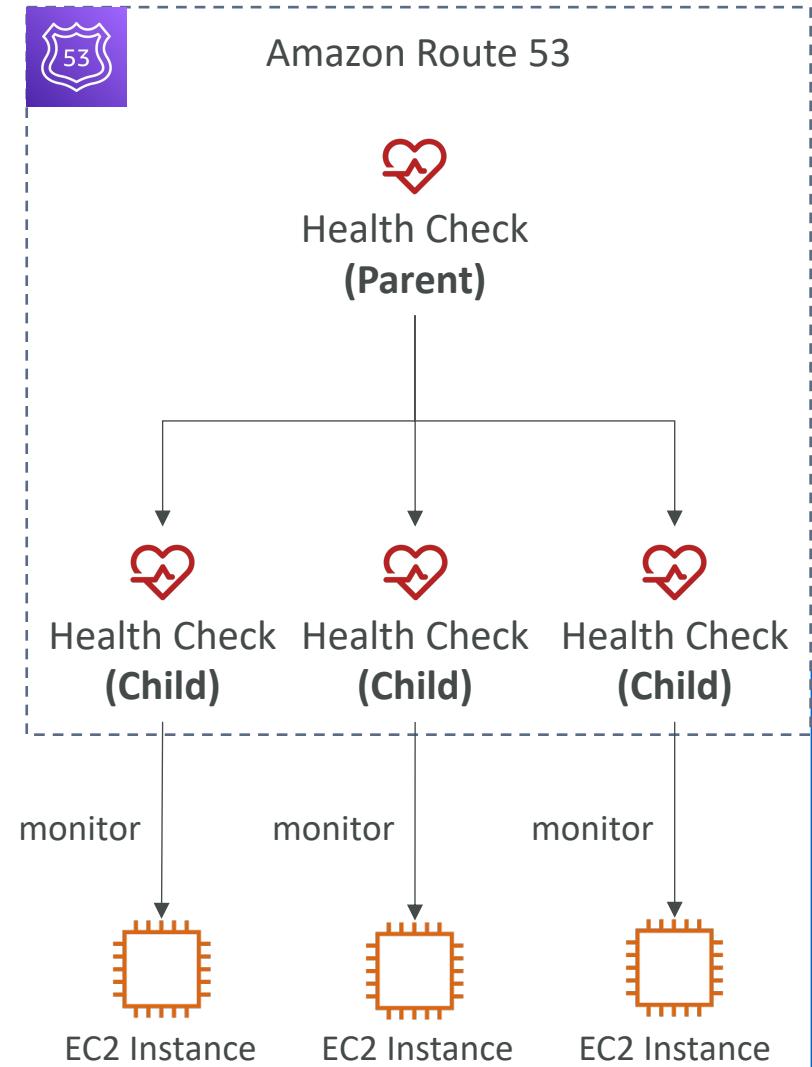
Health Checks – Monitor an Endpoint

- About 15 global health checkers will check the endpoint health
 - Healthy/Unhealthy Threshold – 3 (default)
 - Interval – 30 sec (can set to 10 sec – higher cost)
 - Supported protocol: HTTP, HTTPS and TCP
 - If > 18% of health checkers report the endpoint is healthy, Route 53 considers it **Healthy**. Otherwise, it's **Unhealthy**
 - Ability to choose which locations you want Route 53 to use
- Health Checks pass only when the endpoint responds with the 2xx and 3xx status codes
- Health Checks can be setup to pass / fail based on the text in the first **5120 bytes** of the response
- Configure your router/firewall to allow incoming requests from Route 53 Health Checkers



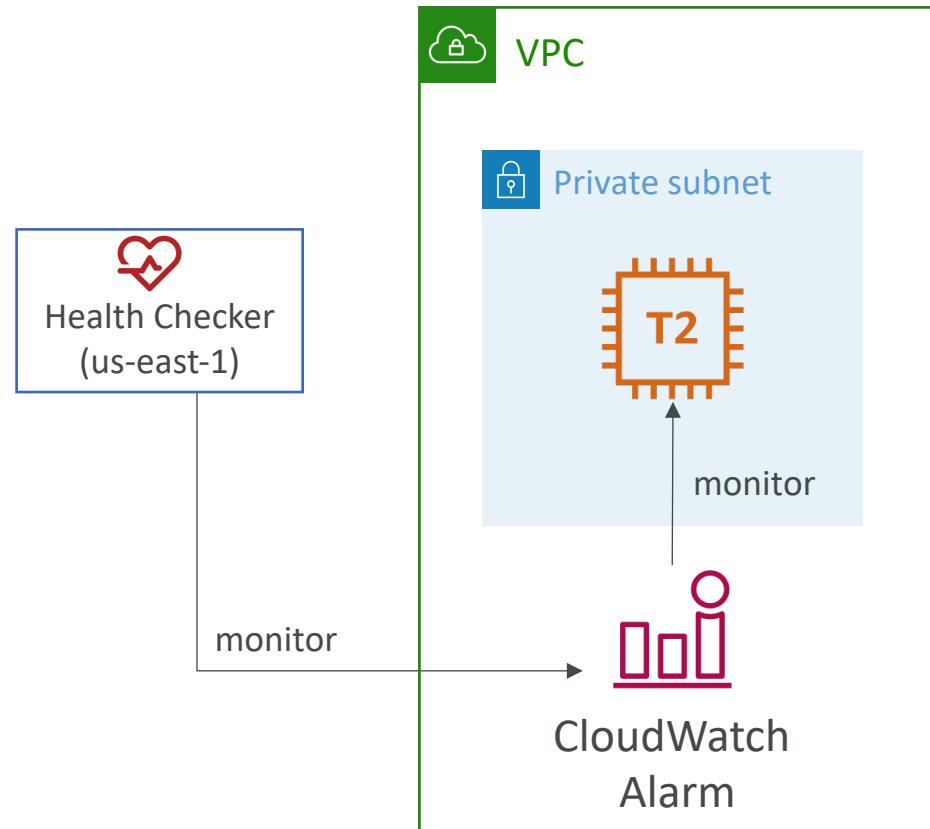
Route 53 – Calculated Health Checks

- Combine the results of multiple Health Checks into a single Health Check
- You can use **OR**, **AND**, or **NOT**
- Can monitor up to 256 Child Health Checks
- Specify how many of the health checks need to pass to make the parent pass
- Usage: perform maintenance to your website without causing all health checks to fail

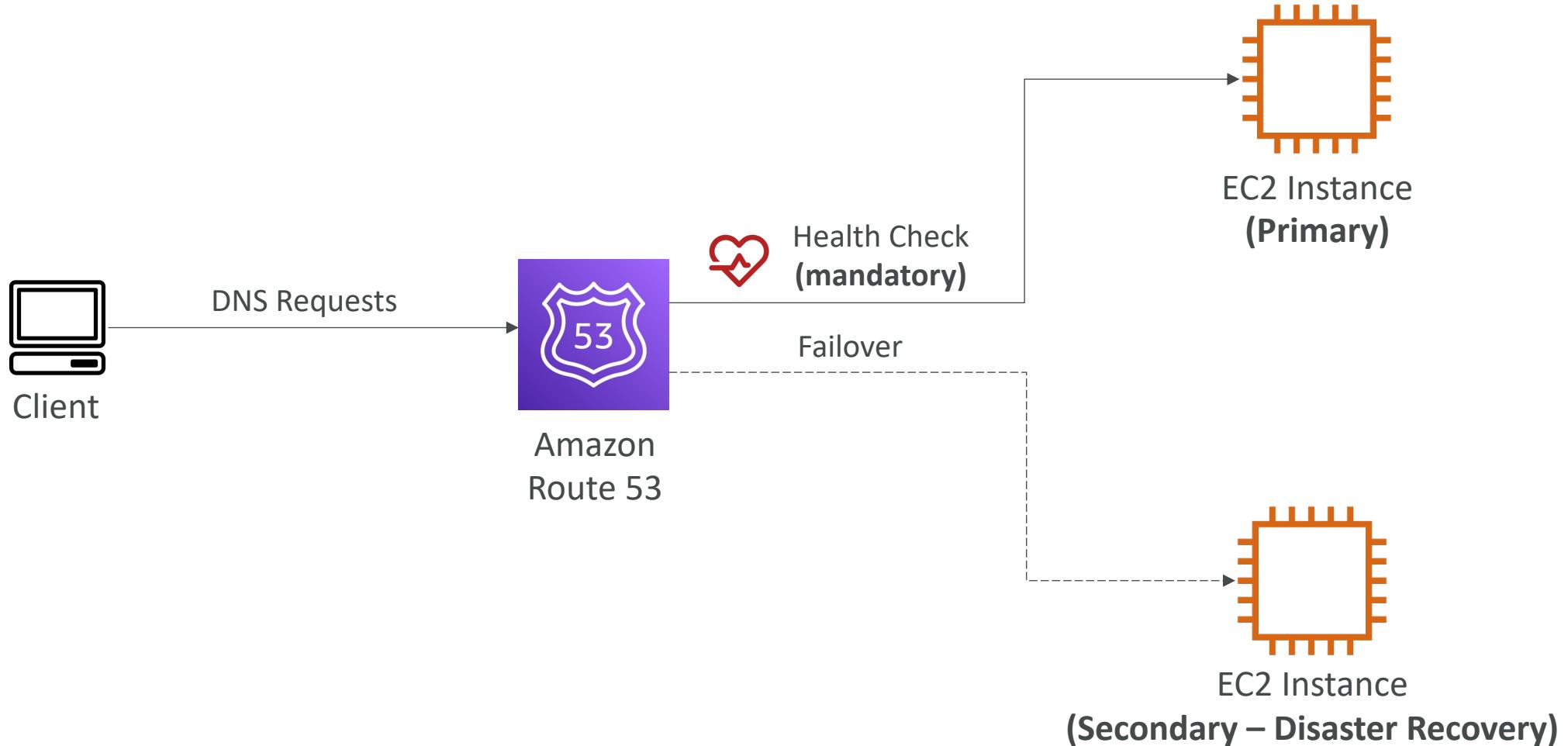


Health Checks – Private Hosted Zones

- Route 53 health checkers are outside the VPC
- They can't access **private** endpoints (private VPC or on-premises resource)
- You can create a CloudWatch Metric and associate a CloudWatch Alarm, then create a Health Check that checks the alarm itself

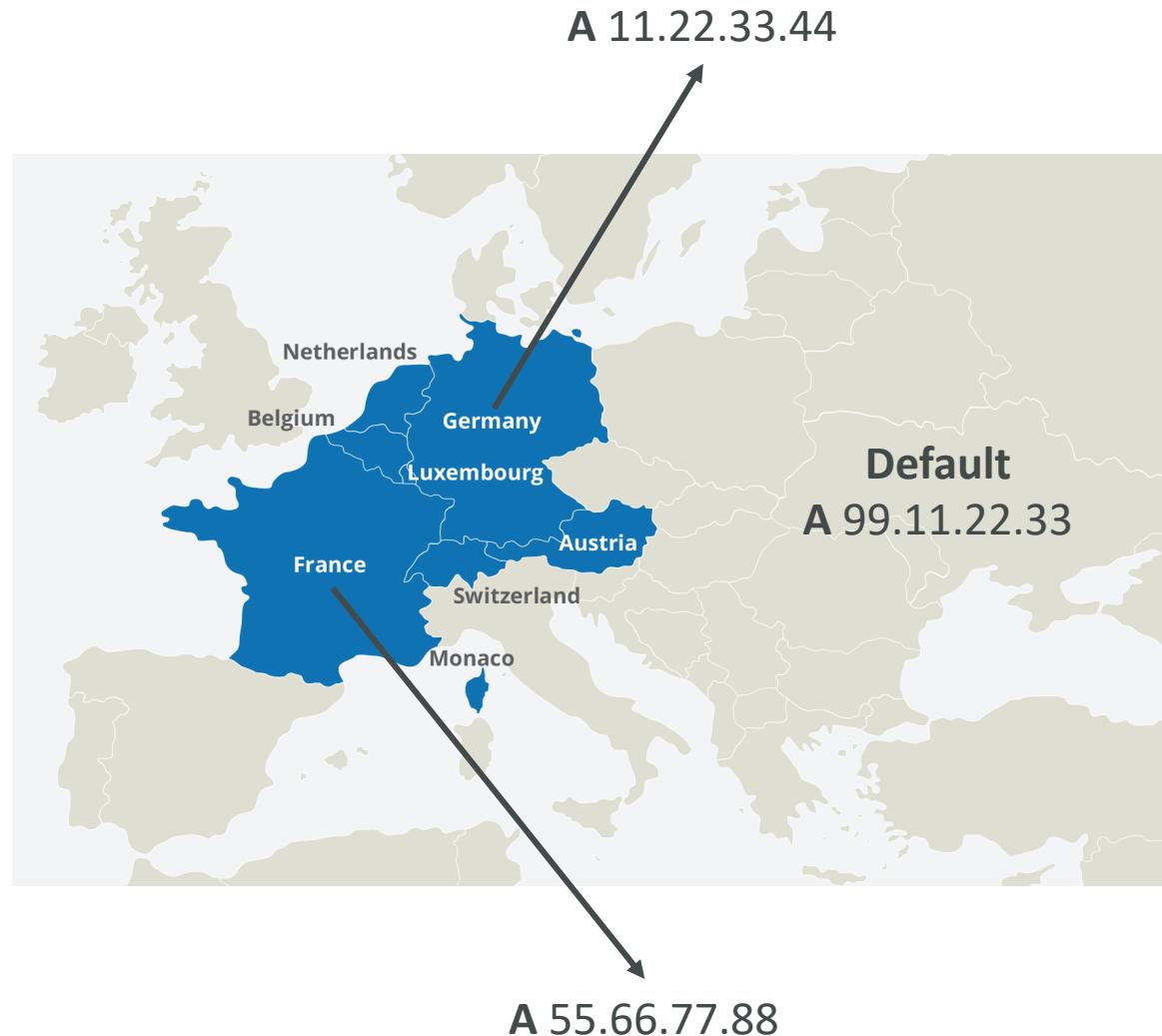


Routing Policies – Failover (Active-Passive)



Routing Policies – Geolocation

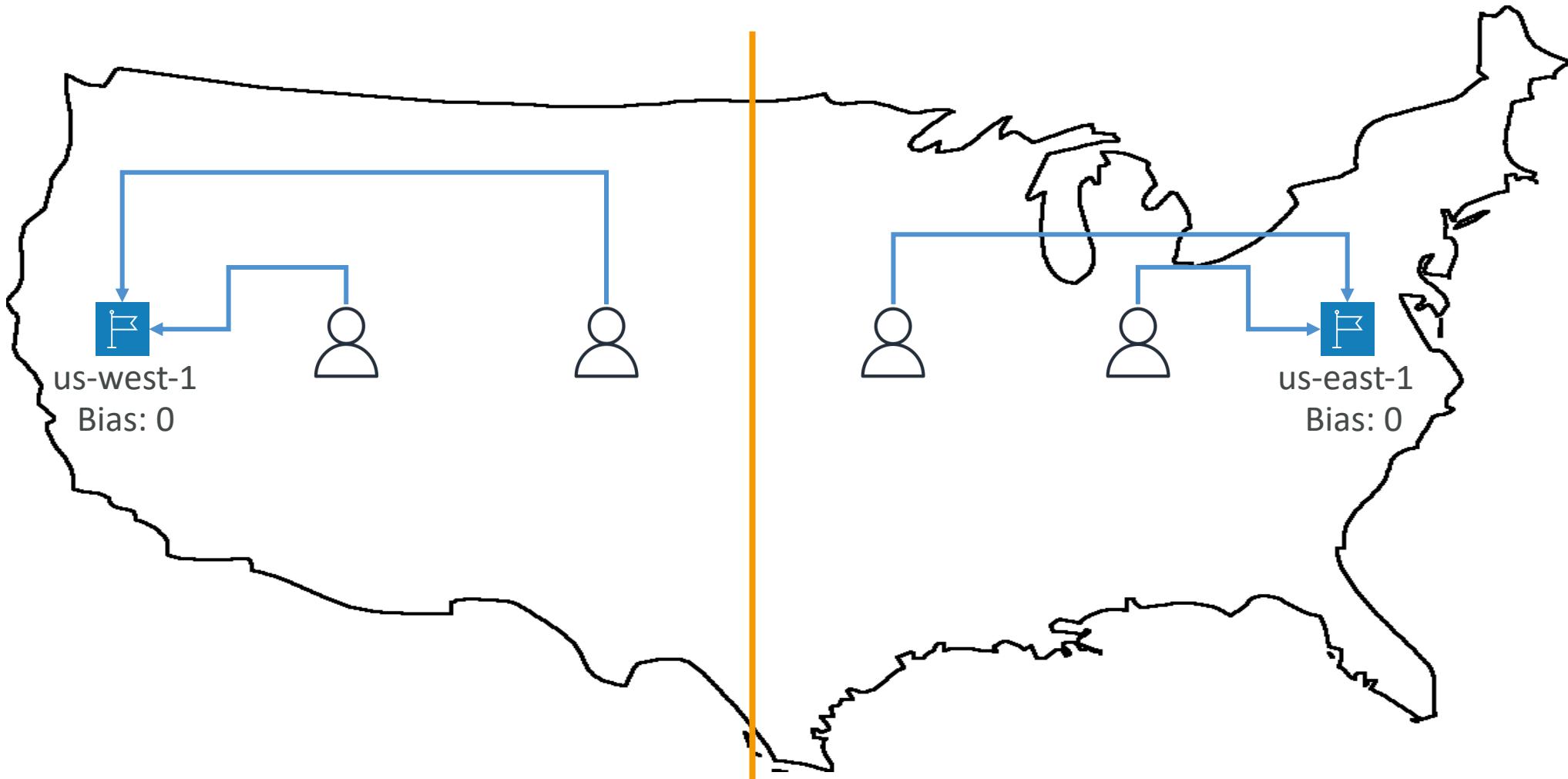
- Different from Latency-based!
- This routing is based on user location
- Specify location by Continent, Country or by US State (if there's overlapping, most precise location selected)
- Should create a “Default” record (in case there's no match on location)
- Use cases: website localization, restrict content distribution, load balancing, ...
- Can be associated with Health Checks



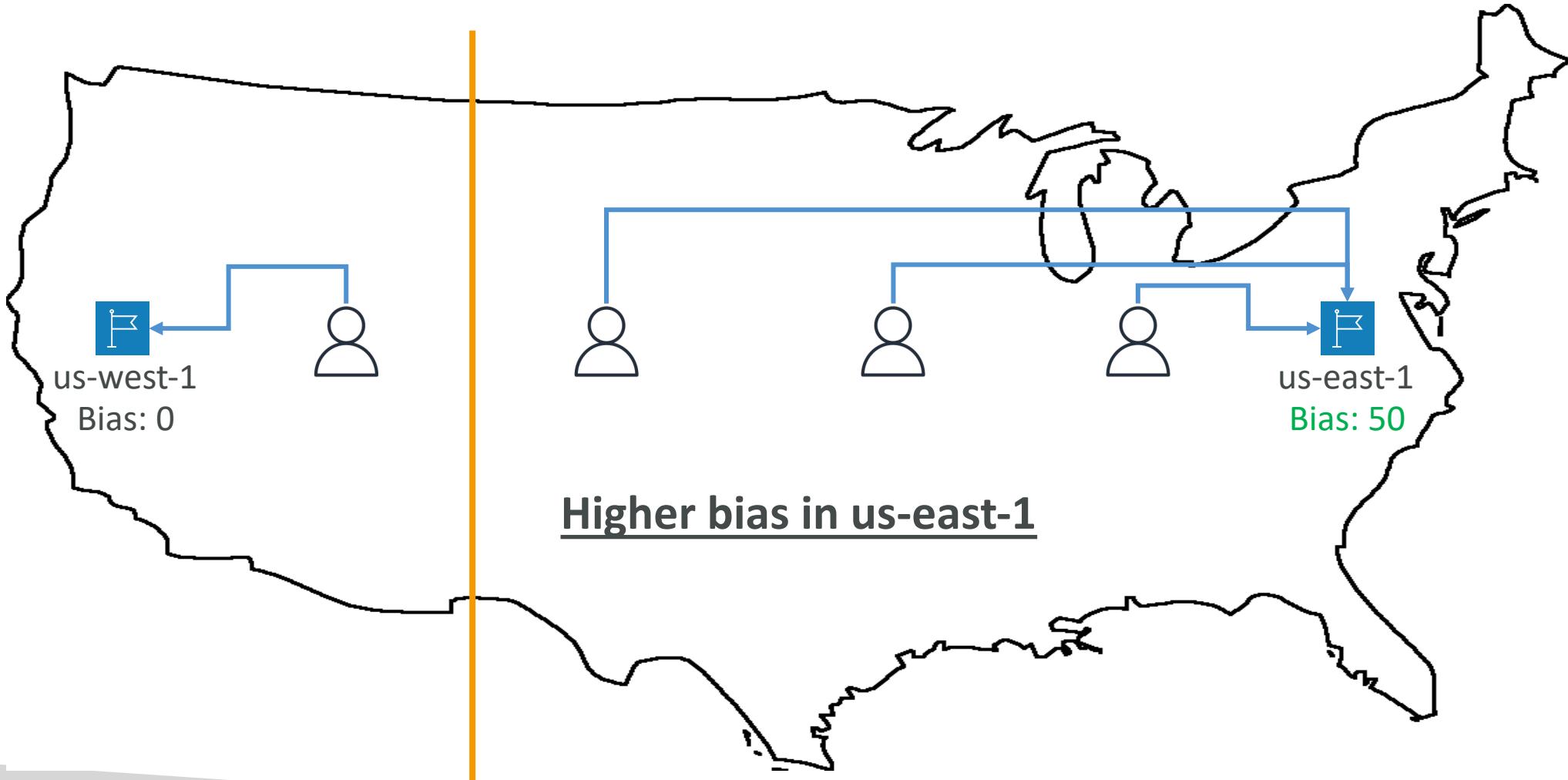
Routing Policies – Geoproximity

- Route traffic to your resources based on the geographic location of users and resources
- Ability **to shift more traffic to resources based** on the defined bias
- To change the size of the geographic region, specify **bias** values:
 - To expand (1 to 99) – more traffic to the resource
 - To shrink (-1 to -99) – less traffic to the resource
- Resources can be:
 - AWS resources (specify AWS region)
 - Non-AWS resources (specify Latitude and Longitude)
- You must use Route 53 Traffic Flow to use this feature

Routing Policies – Geoproximity

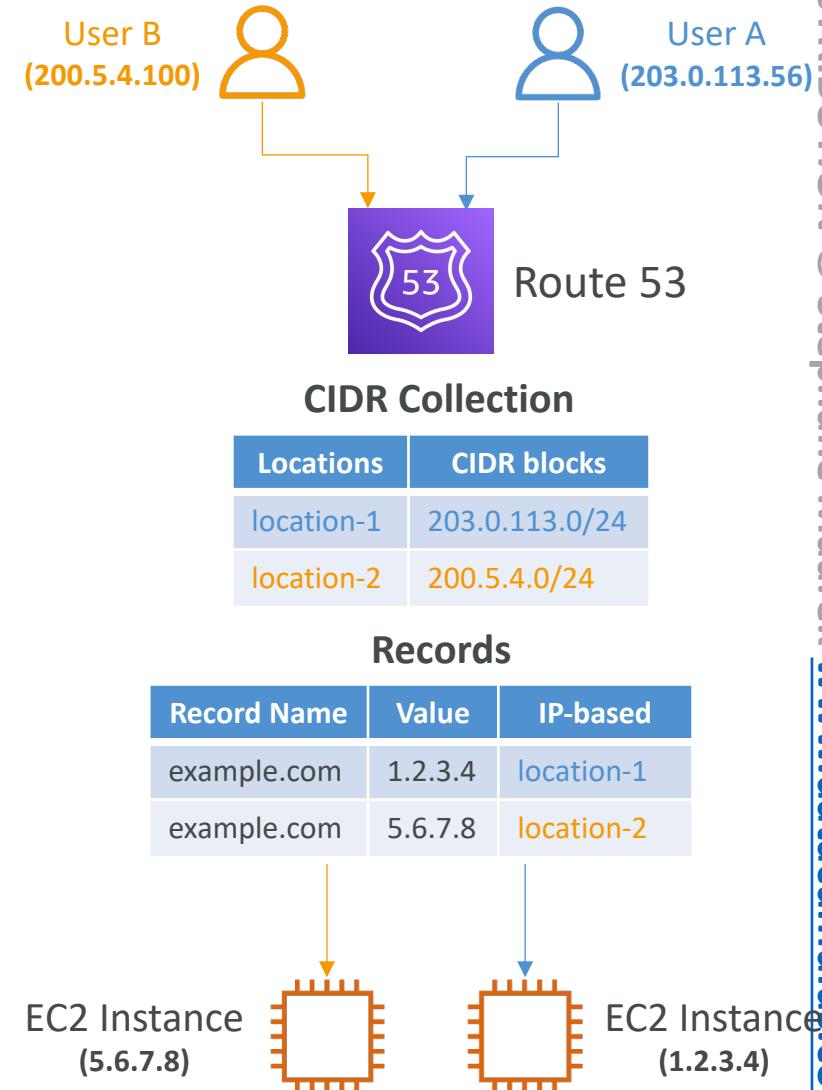


Routing Policies – Geoproximity



Routing Policies – IP-based Routing

- Routing is based on clients' IP addresses
- You provide a list of CIDRs for your clients and the corresponding endpoints/locations (user-IP-to-endpoint mappings)
- Use cases: Optimize performance, reduce network costs...
- Example: route end users from a particular ISP to a specific endpoint



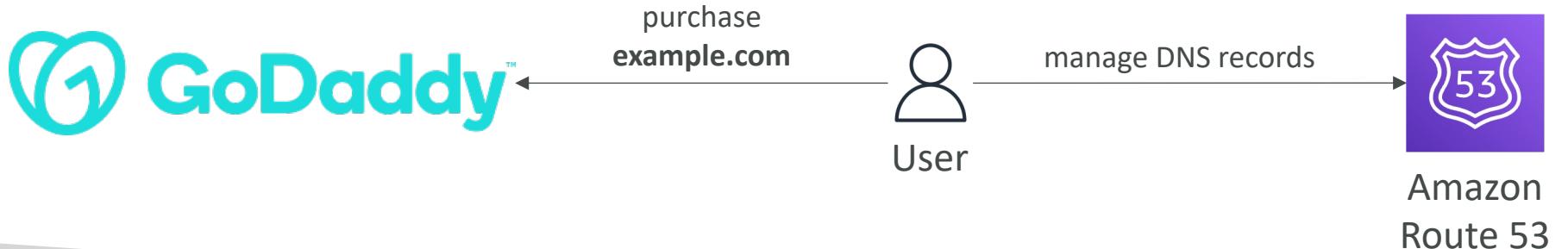
Routing Policies – Multi-Value

- Use when routing traffic to multiple resources
- Route 53 return multiple values/resources
- Can be associated with Health Checks (return only values for healthy resources)
- Up to 8 healthy records are returned for each Multi-Value query
- Multi-Value is not a substitute for having an ELB

Name	Type	Value	TTL	Set ID	Health Check
www.example.com	A Record	192.0.2.2	60	Web1	A
www.example.com	A Record	198.51.100.2	60	Web2	B
www.example.com	A Record	203.0.113.2	60	Web3	C

Domain Registrar vs. DNS Service

- You buy or register your domain name with a Domain Registrar typically by paying annual charges (e.g., GoDaddy, Amazon Registrar Inc., ...)
- The Domain Registrar usually provides you with a DNS service to manage your DNS records
- But you can use another DNS service to manage your DNS records
- Example: purchase the domain from GoDaddy and use Route 53 to manage your DNS records



GoDaddy as Registrar & Route 53 as DNS Service



Records

We can't display your DNS information because your nameservers aren't managed by us.

Nameservers

Using custom nameservers [Change](#)

Nameserver
ns-1083.awsdns-07.org
ns-932.awsdns-52.net
ns-1911.awsdns-46.co.uk
ns-481.awsdns-60.com



Amazon
Route 53

Public Hosted Zone
stephanetheteacher.com

▼ Hosted zone details [Edit hosted zone](#)

Hosted zone ID	Type	Name servers
Z30IJCCWPKUV	Public hosted zone	ns-252.awsdns-31.com ns-1468.awsdns-55.org ns-633.awsdns-15.net ns-1800.awsdns-33.co.uk
Description	Record count	
HostedZone created by Route53 Registrar	22	
Query log		

3rd Party Registrar with Amazon Route 53

- If you buy your domain on a 3rd party registrar, you can still use Route 53 as the DNS Service provider
 - 1. Create a Hosted Zone in Route 53
 - 2. Update NS Records on 3rd party website to use Route 53 Name Servers
- Domain Registrar != DNS Service
- But every Domain Registrar usually comes with some DNS features

Classic Solutions Architecture

Section Introduction

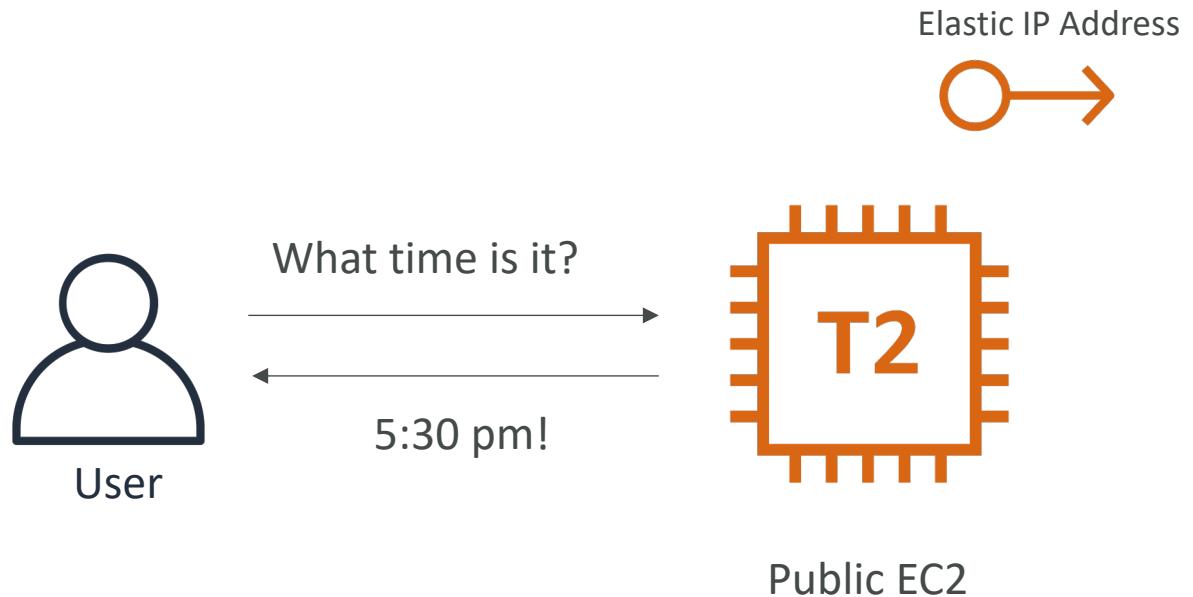
- These solutions architectures are the best part of this course
- Let's understand how all the technologies we've seen work together
- This is a section you need to be 100% comfortable with
- We'll see the progression of a Solution's architect mindset through many sample case studies:
 - WhatIsTheTime.Com
 - MyClothes.Com
 - MyWordPress.Com
 - Instantiating applications quickly
 - Beanstalk

Stateless Web App: WhatIsTheTime.com

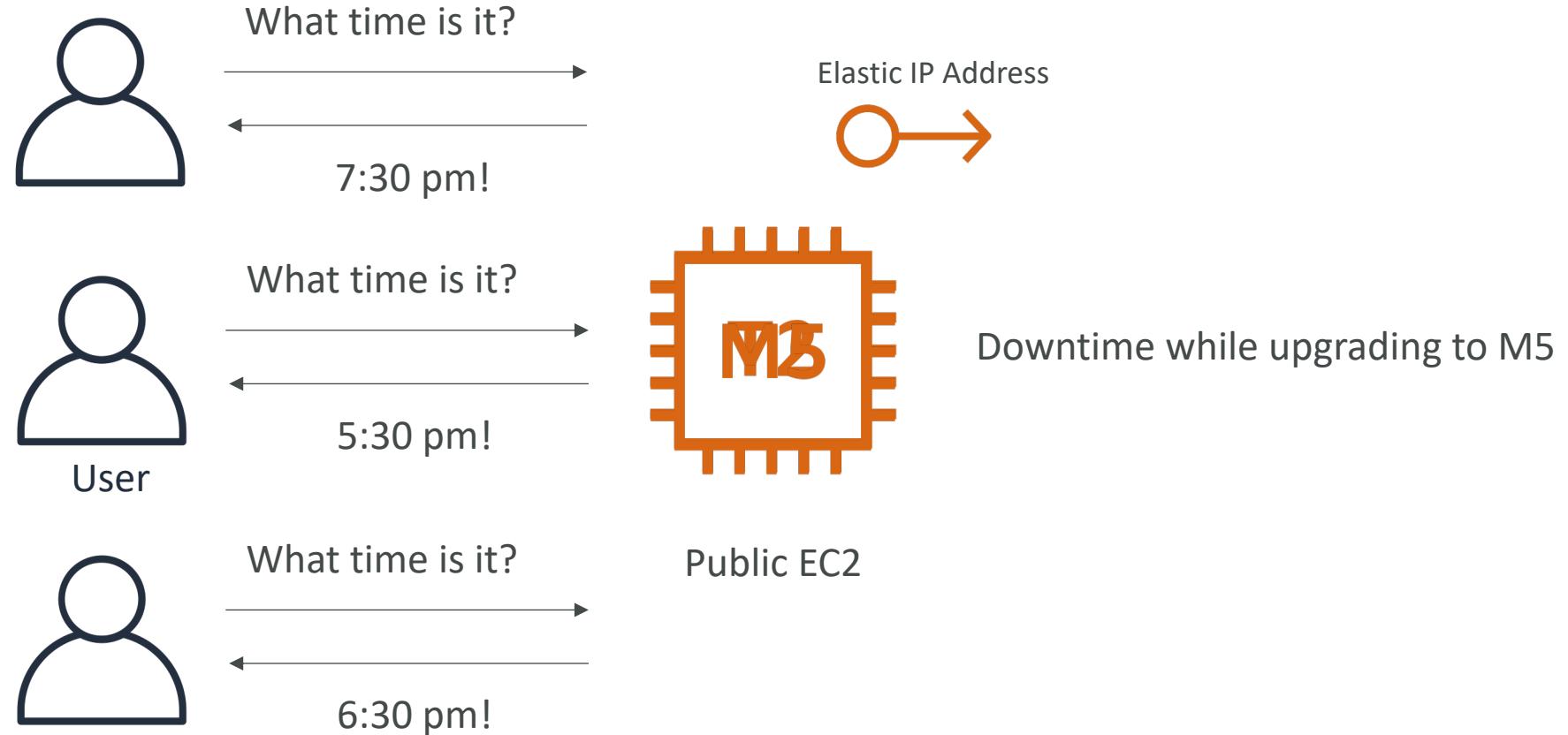
- WhatIsTheTime.com allows people to know what time it is
- We don't need a database
- We want to start small and can accept downtime
- We want to fully scale vertically and horizontally, no downtime
- Let's go through the Solutions Architect journey for this app
- Let's see how we can proceed!

Stateless web app: What time is it?

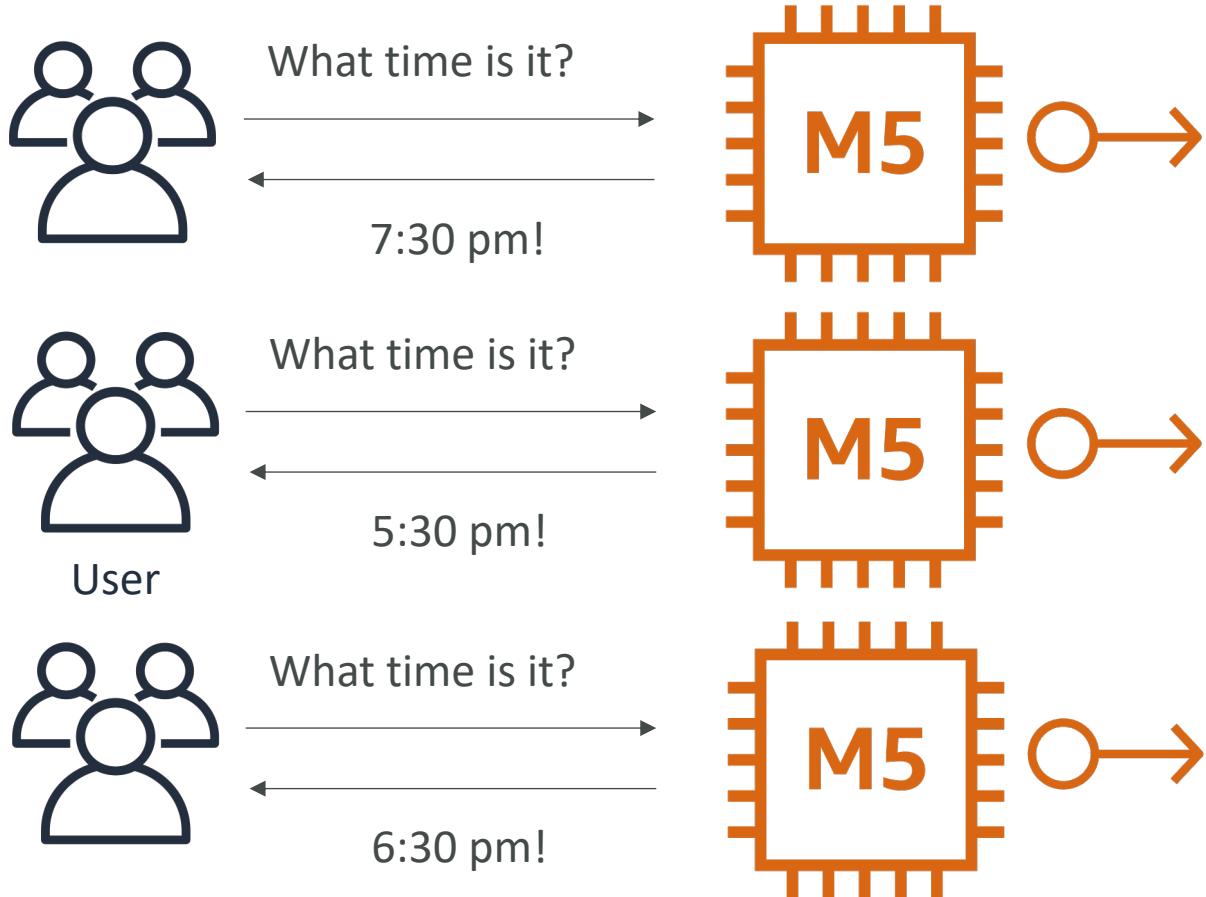
Starting simple



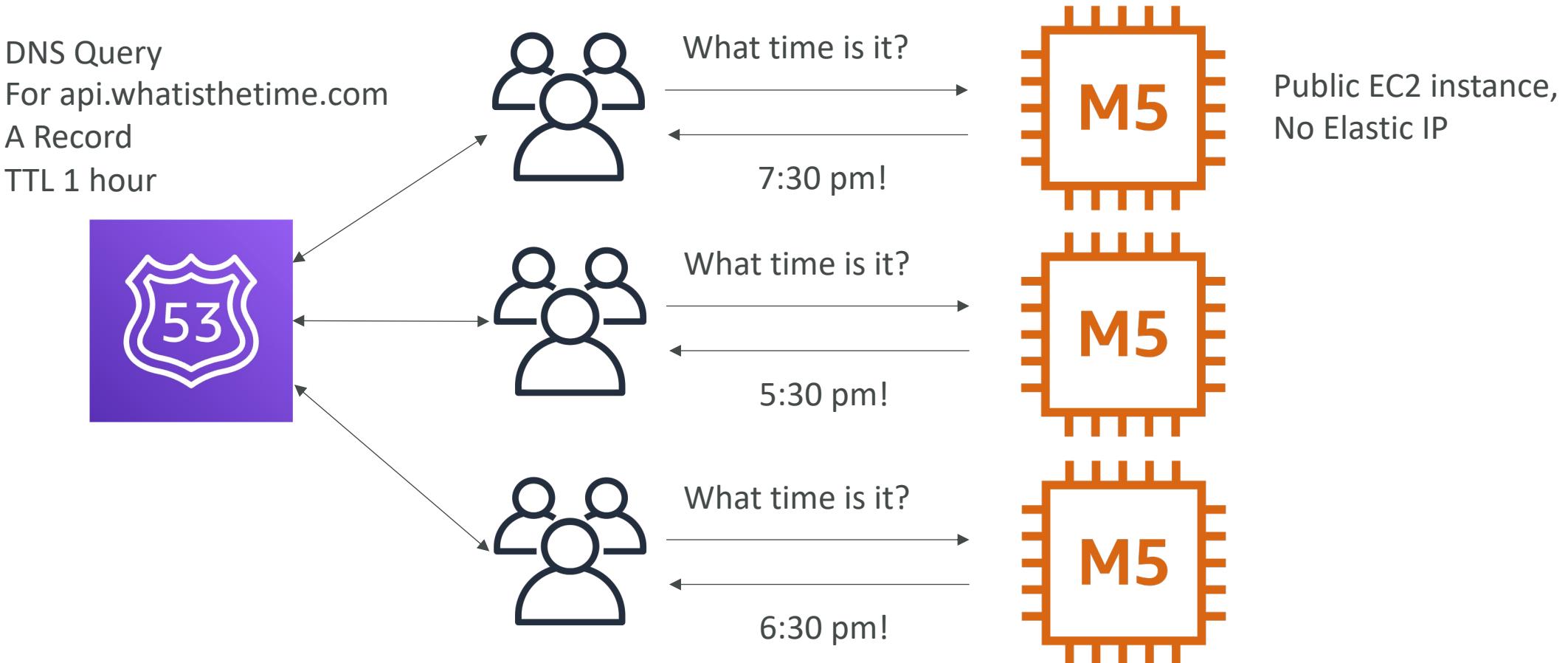
Stateless web app: What time is it? Scaling vertically



Stateless web app: What time is it? Scaling horizontally

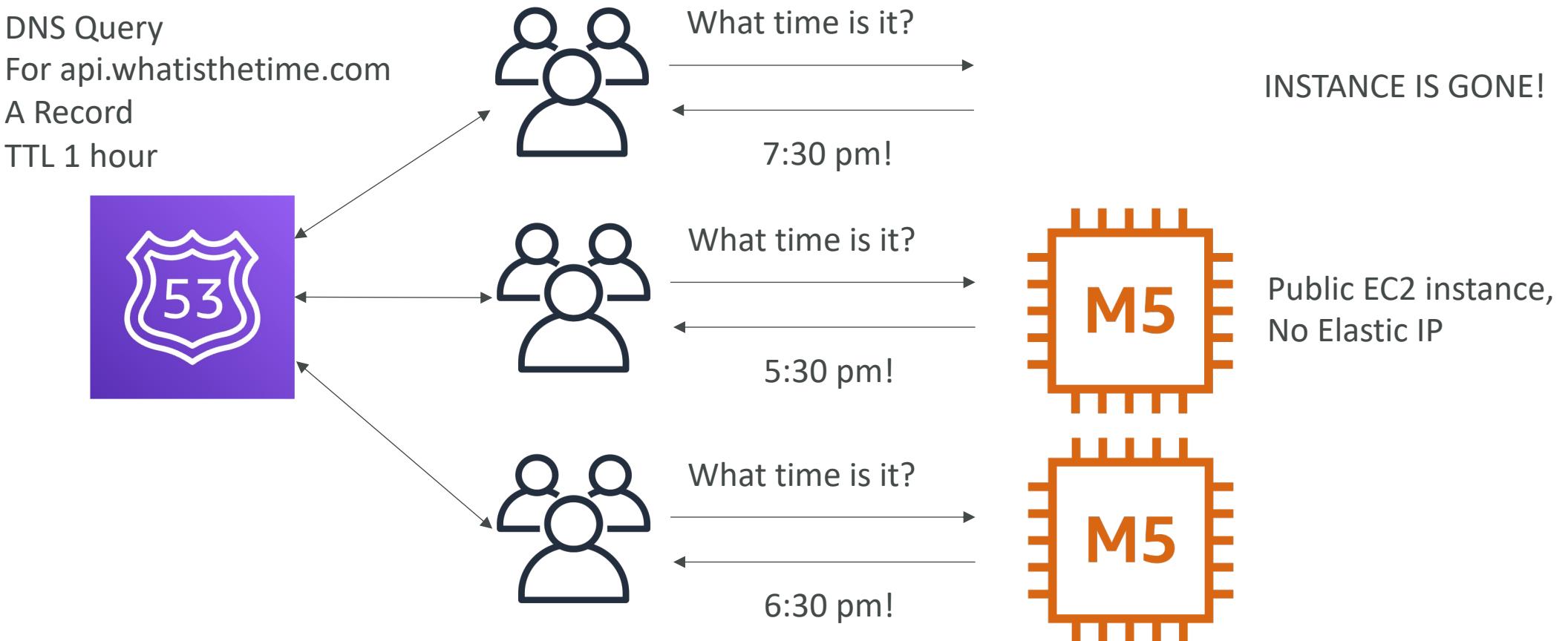


Stateless web app: What time is it? Scaling horizontally

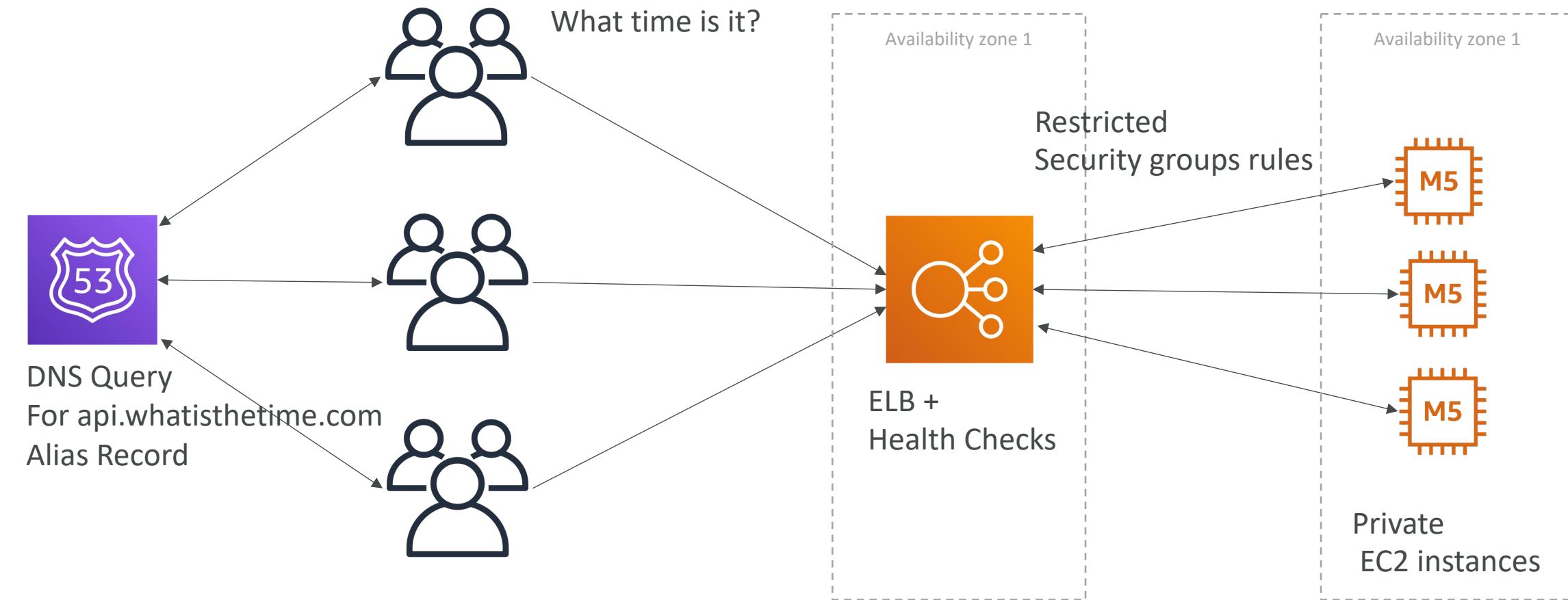


Stateless web app: What time is it?

Scaling horizontally, adding and removing instances

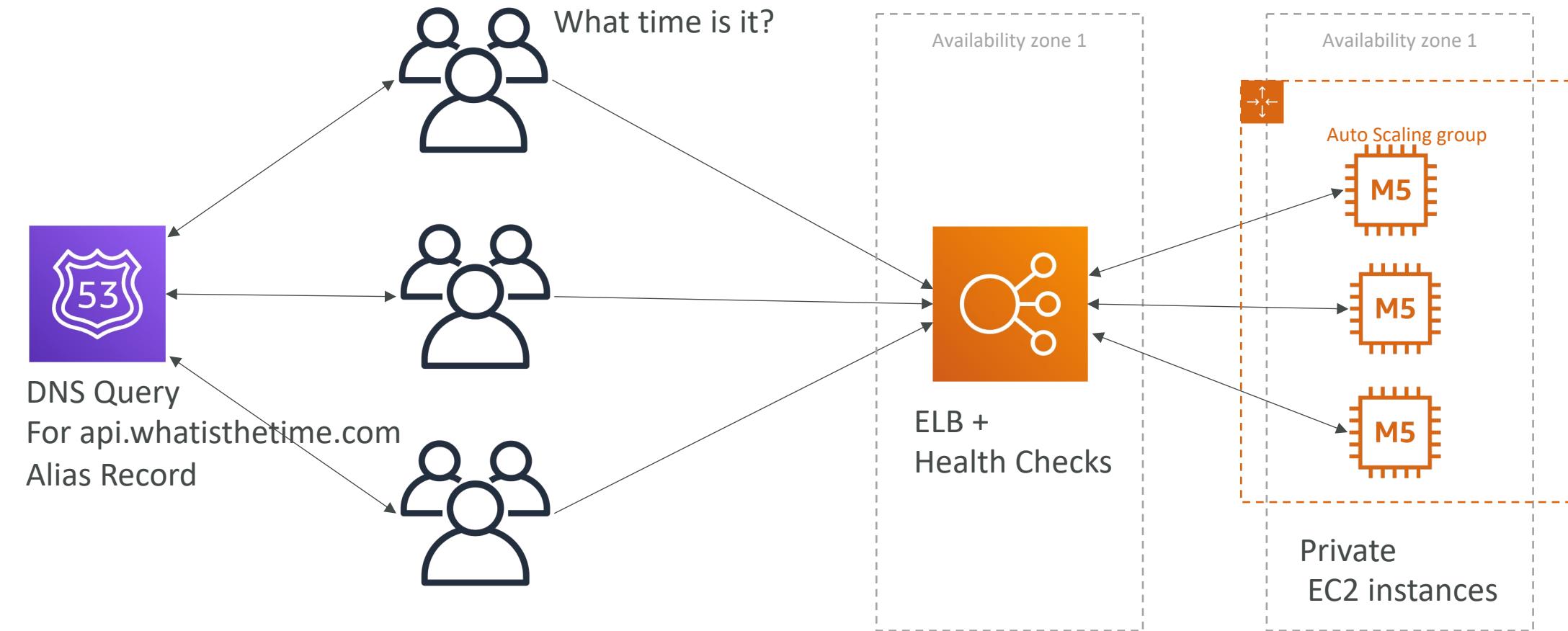


Stateless web app: What time is it? Scaling horizontally, with a load balancer



Stateless web app: What time is it?

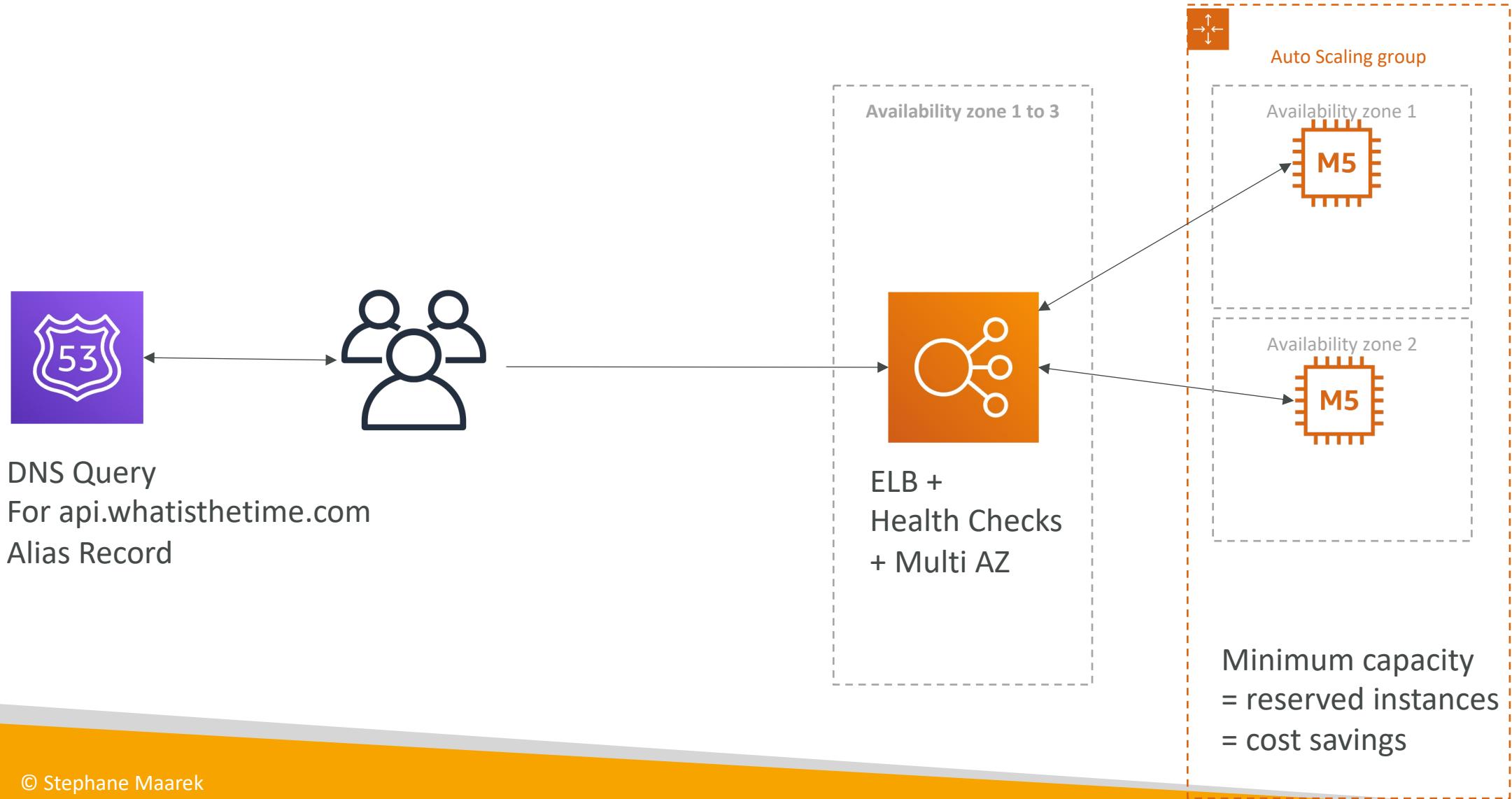
Scaling horizontally, with an auto-scaling group



Stateless web app: What time is it? Making our app multi-AZ



Minimum 2 AZ => Let's reserve capacity



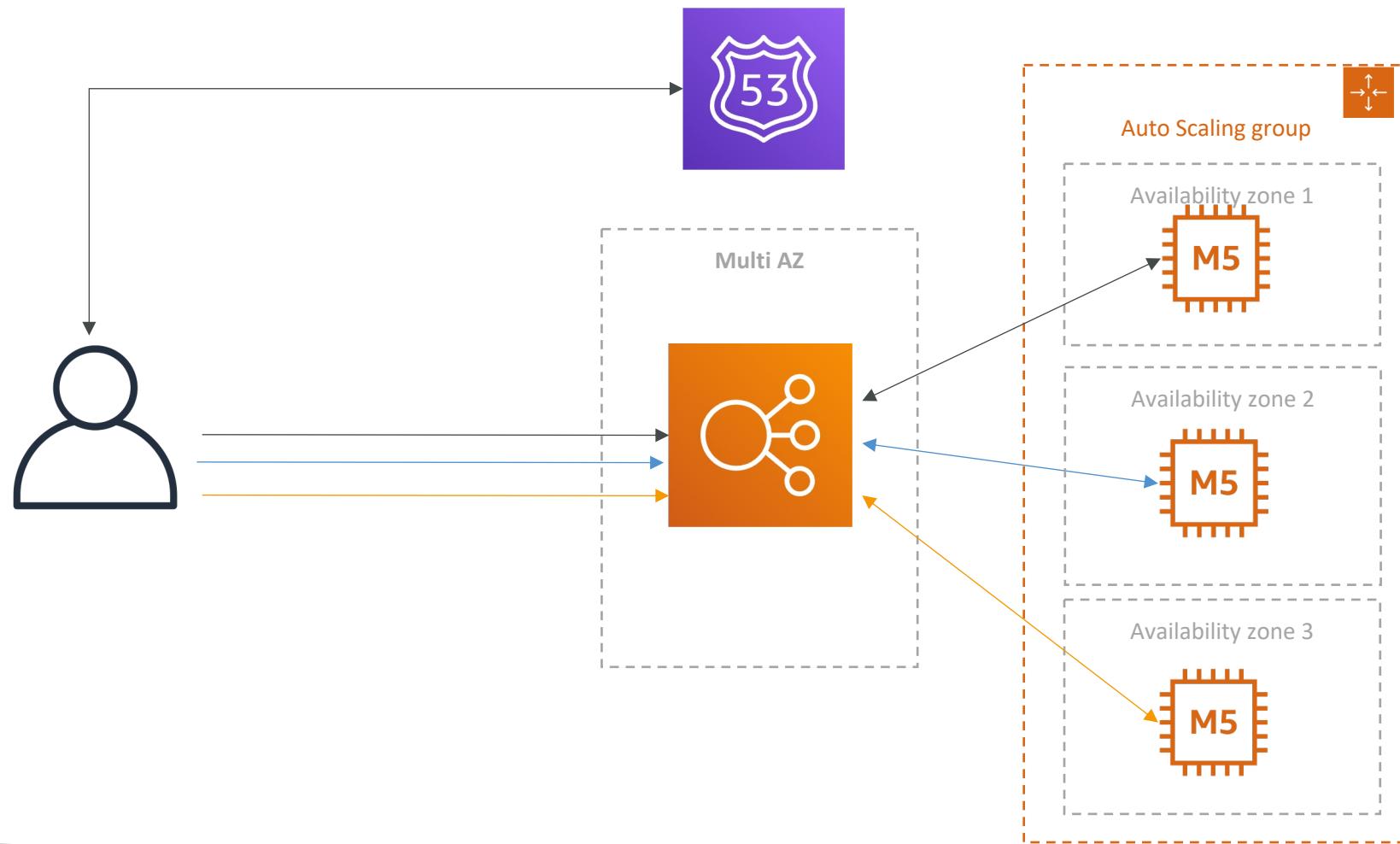
In this lecture we've discussed...

- Public vs Private IP and EC2 instances
- Elastic IP vs Route 53 vs Load Balancers
- Route 53 TTL, A records and Alias Records
- Maintaining EC2 instances manually vs Auto Scaling Groups
- Multi AZ to survive disasters
- ELB Health Checks
- Security Group Rules
- Reservation of capacity for costing savings when possible
- We're considering 5 pillars for a well architected application:
costs, performance, reliability, security, operational excellence

Stateful Web App: MyClothes.com

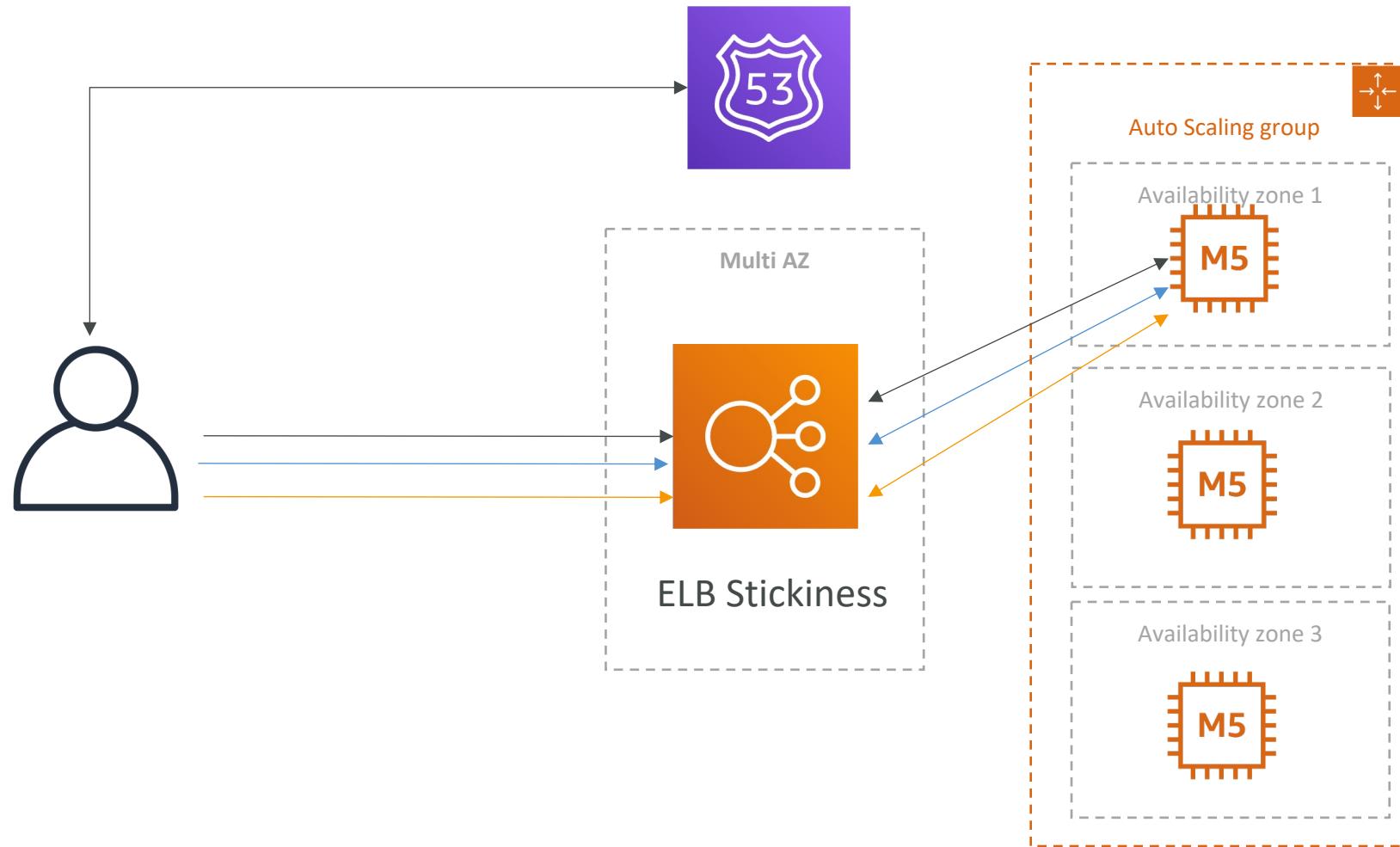
- MyClothes.com allows people to buy clothes online.
- There's a shopping cart
- Our website is having hundreds of users at the same time
- We need to scale, maintain horizontal scalability and keep our web application as stateless as possible
- Users should not lose their shopping cart
- Users should have their details (address, etc) in a database
- Let's see how we can proceed!

Stateful Web App: MyClothes.com



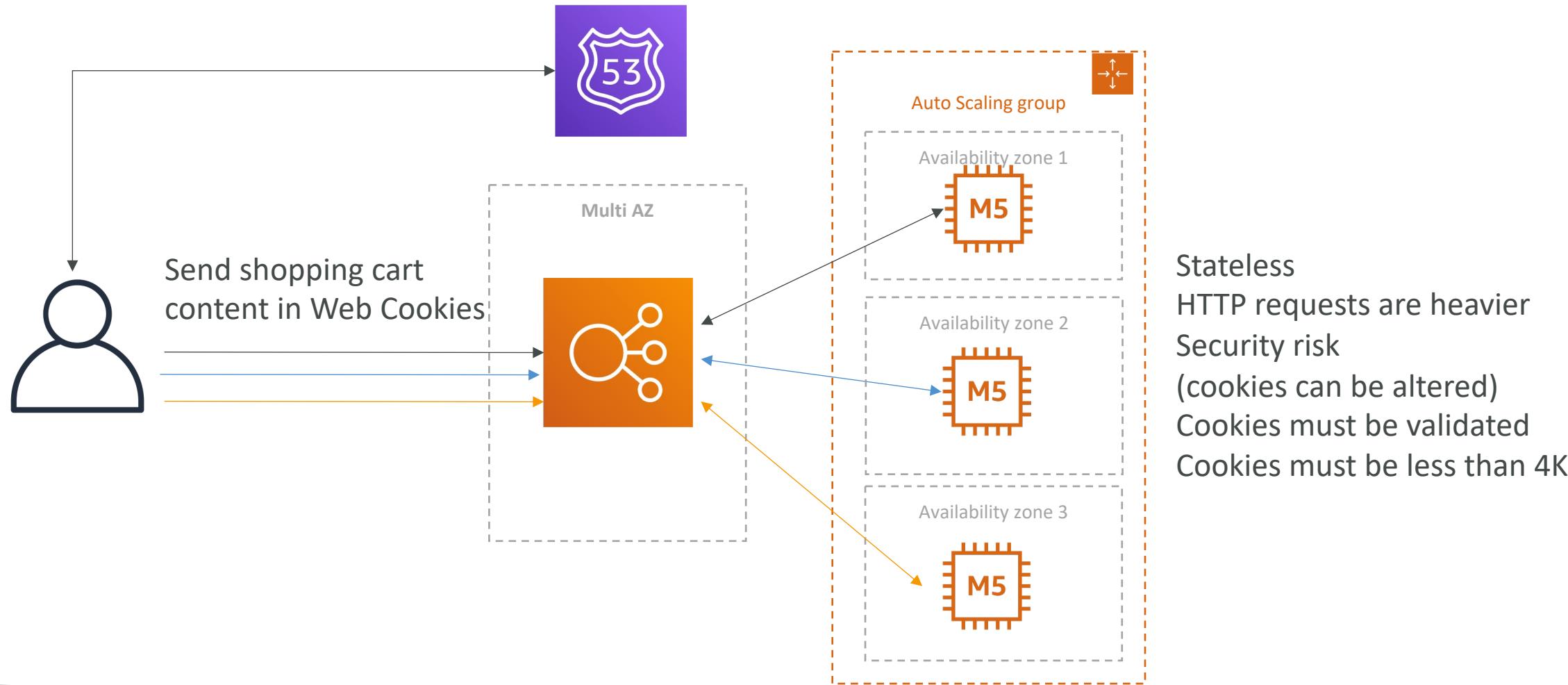
Stateful Web App: MyClothes.com

Introduce Stickiness (Session Affinity)



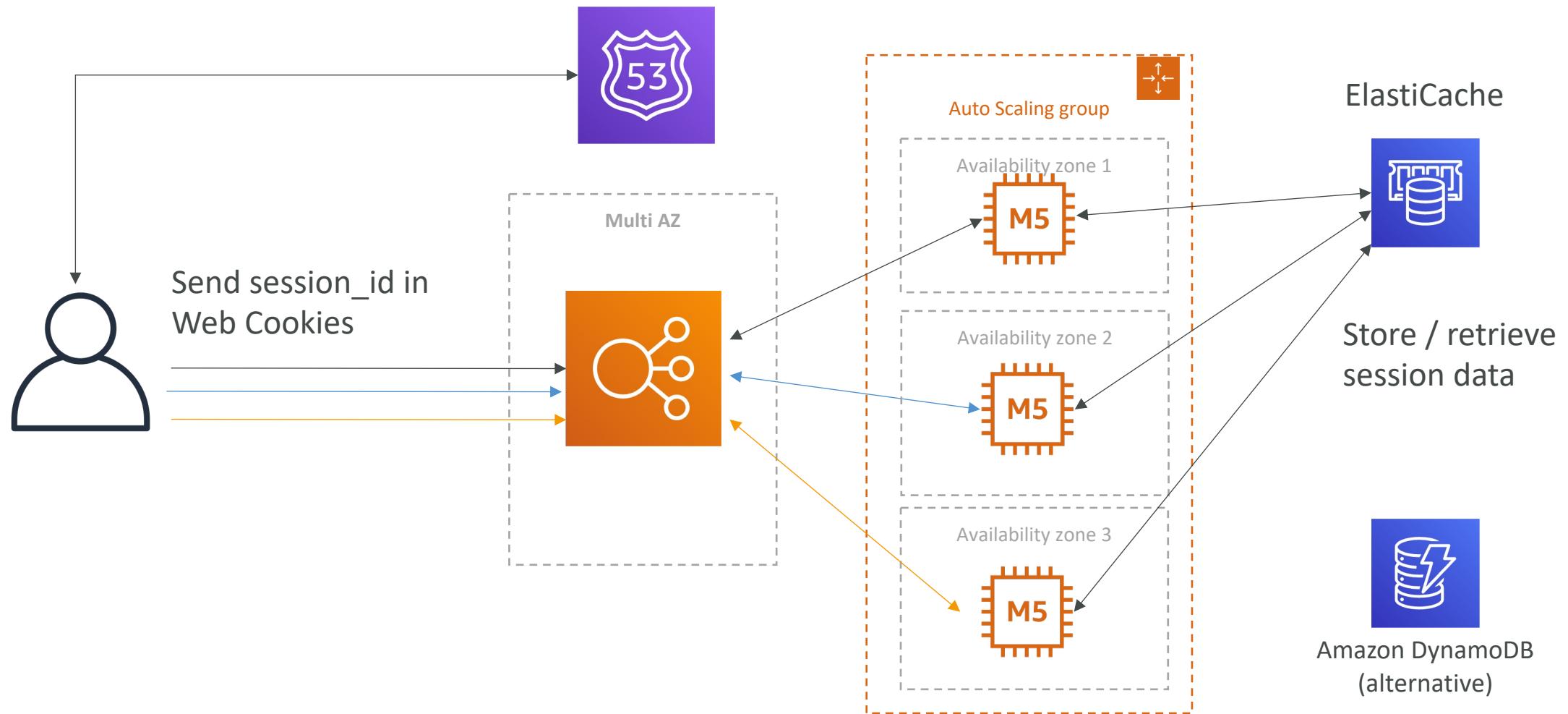
Stateful Web App: MyClothes.com

Introduce User Cookies



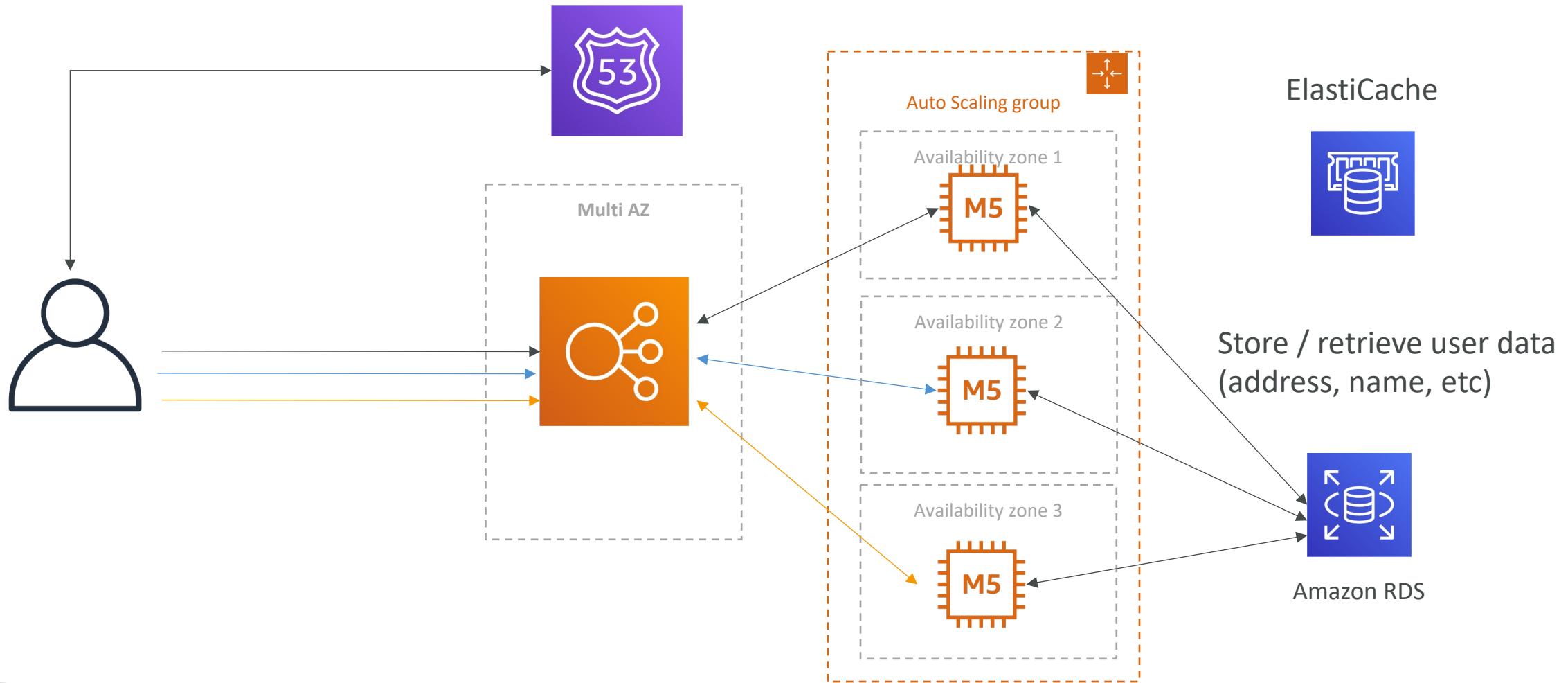
Stateful Web App: MyClothes.com

Introduce Server Session



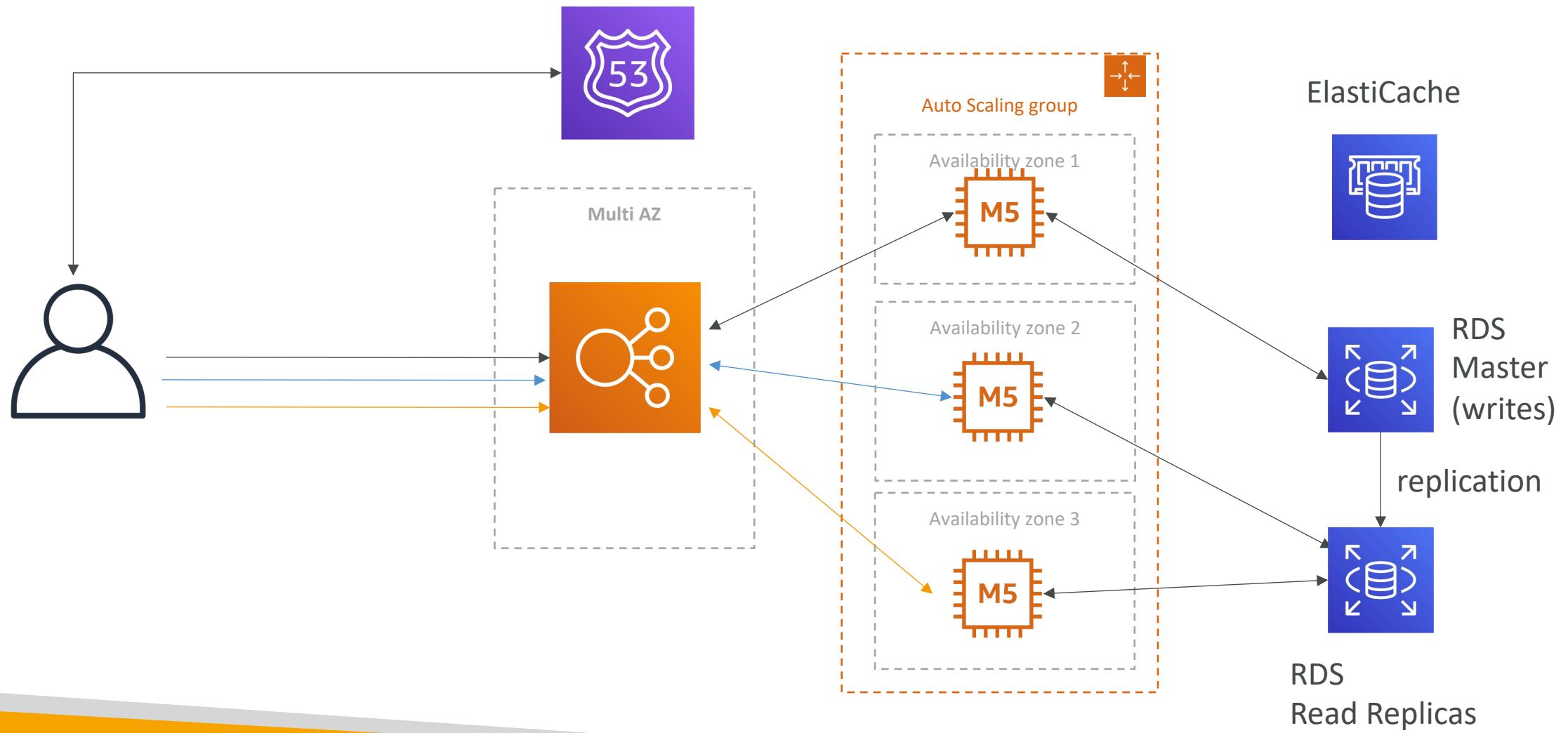
Stateful Web App: MyClothes.com

Storing User Data in a database



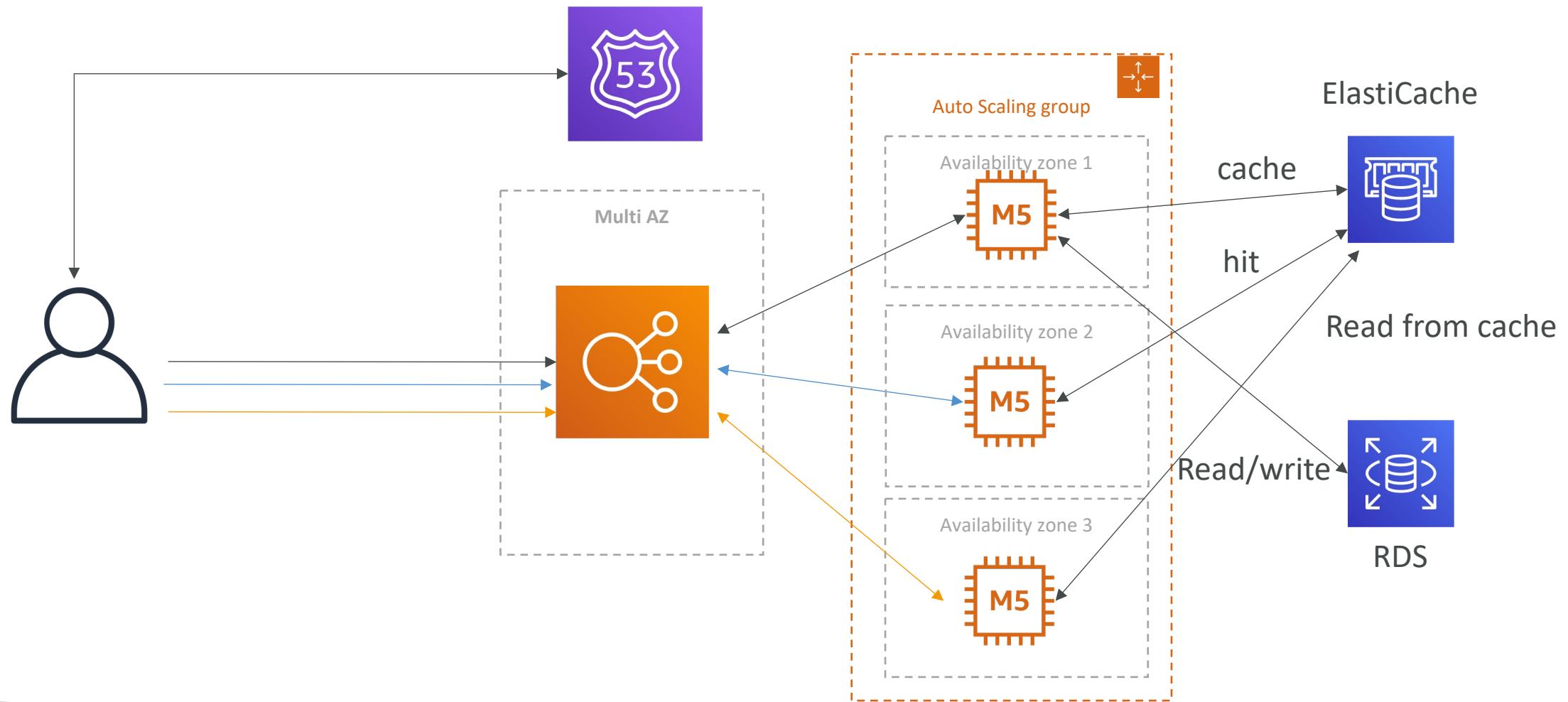
Stateful Web App: MyClothes.com

Scaling Reads



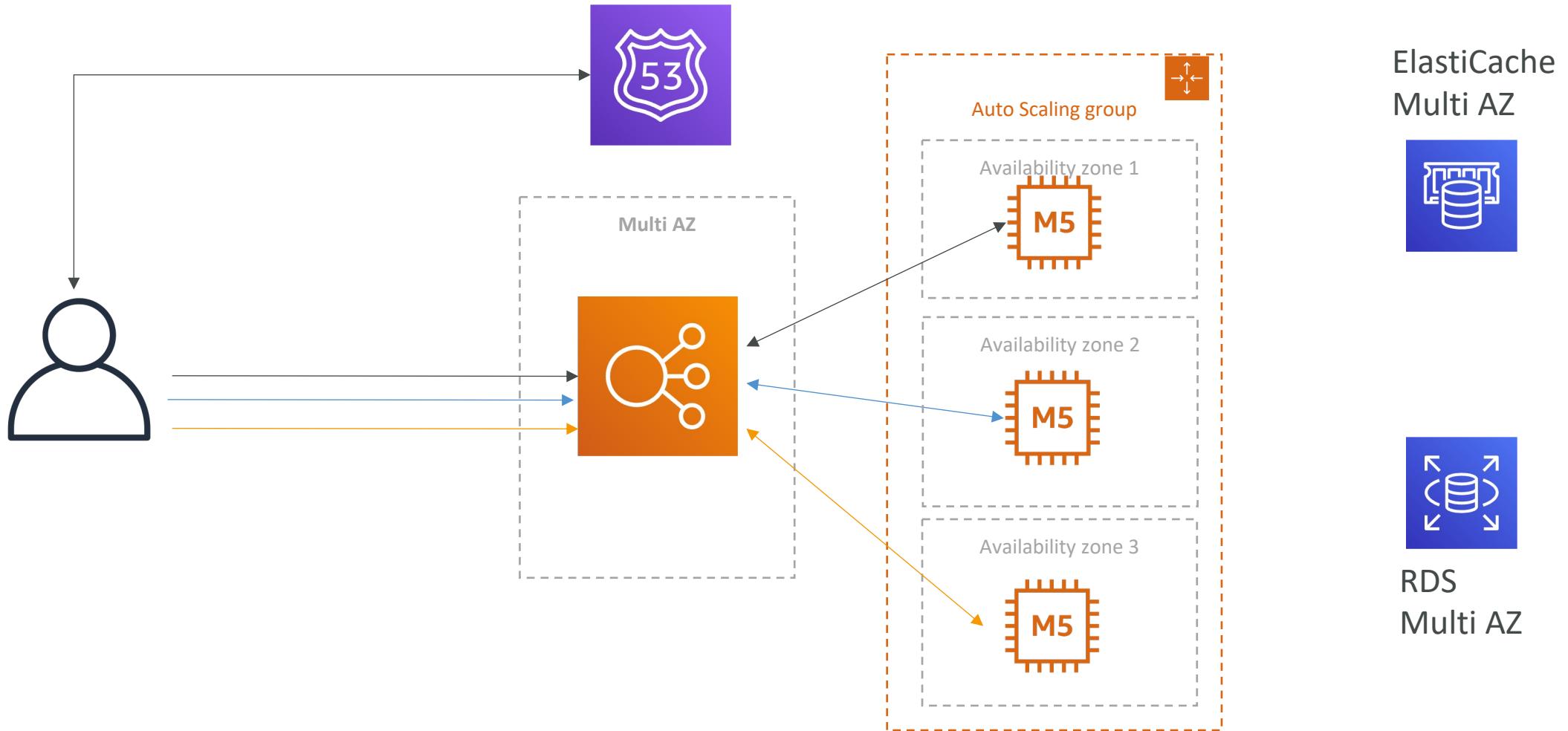
Stateful Web App: MyClothes.com

Scaling Reads (Alternative) – Lazy Loading



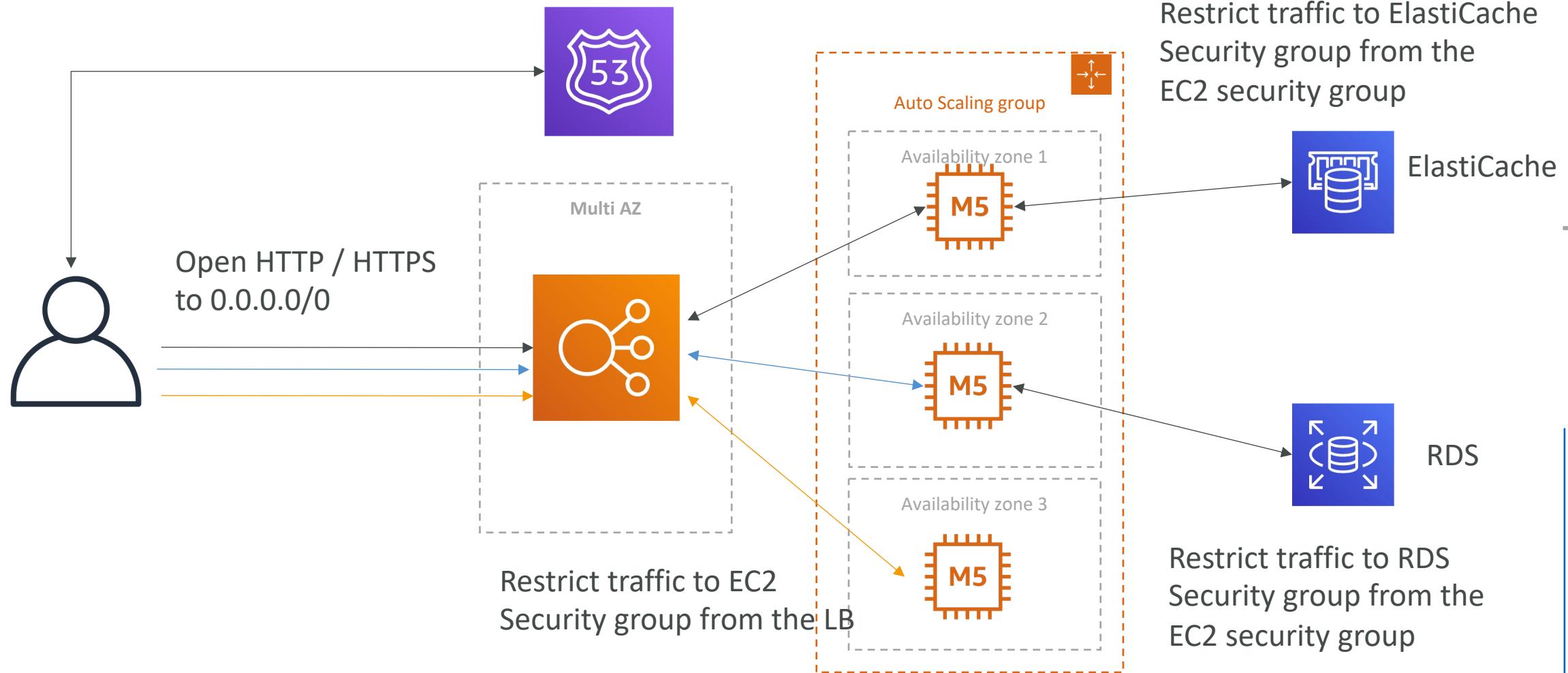
Stateful Web App: MyClothes.com

Multi AZ – Survive disasters



Stateful Web App: MyClothes.com

Security Groups



In this lecture we've discussed...

3-tier architectures for web applications

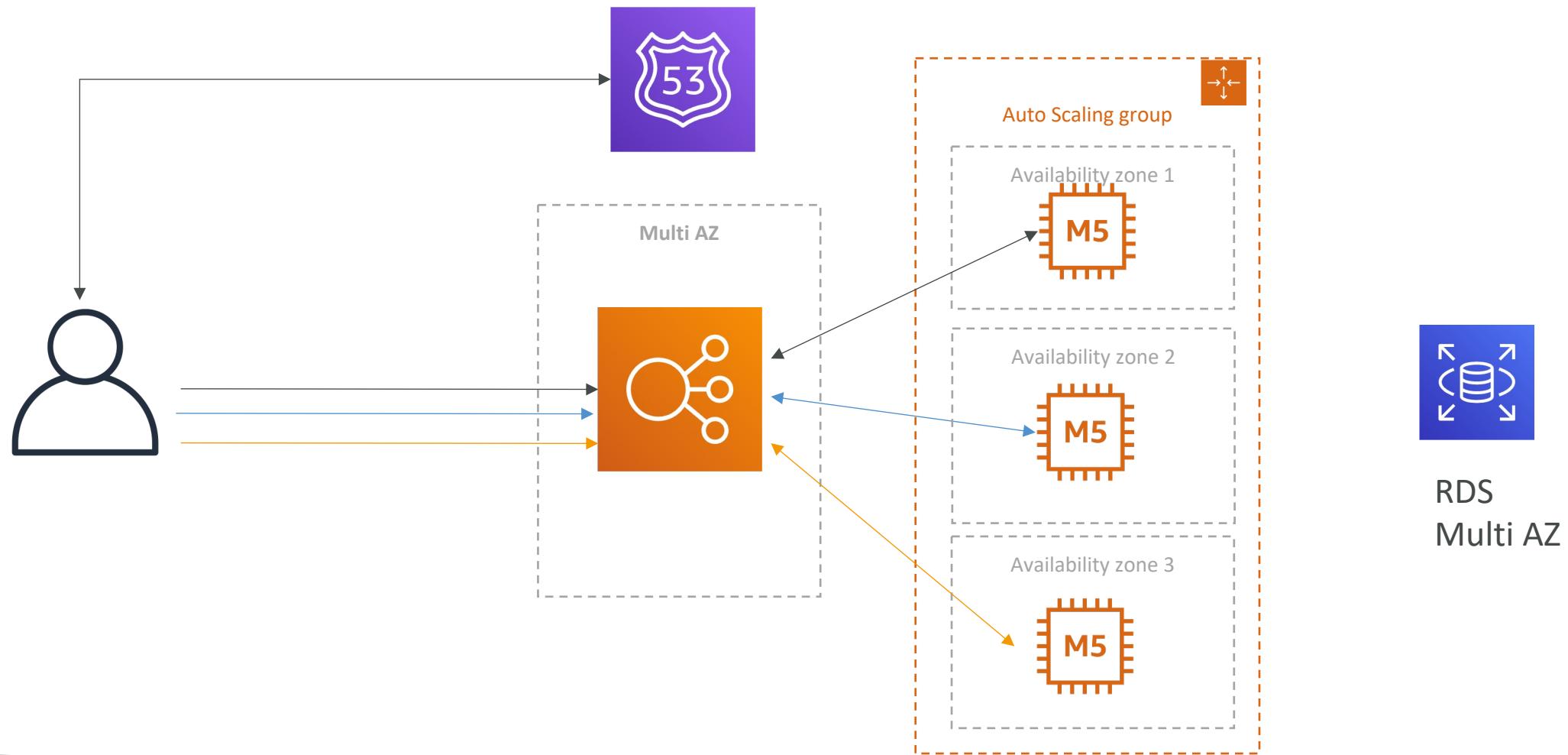
- ELB sticky sessions
- Web clients for storing cookies and making our web app stateless
- ElastiCache
 - For storing sessions (alternative: DynamoDB)
 - For caching data from RDS
 - Multi AZ
- RDS
 - For storing user data
 - Read replicas for scaling reads
 - Multi AZ for disaster recovery
- Tight Security with security groups referencing each other

Stateful Web App: MyWordPress.com

- We are trying to create a fully scalable WordPress website
 - We want that website to access and correctly display picture uploads
 - Our user data, and the blog content should be stored in a MySQL database.
-
- Let's see how we can achieve this!

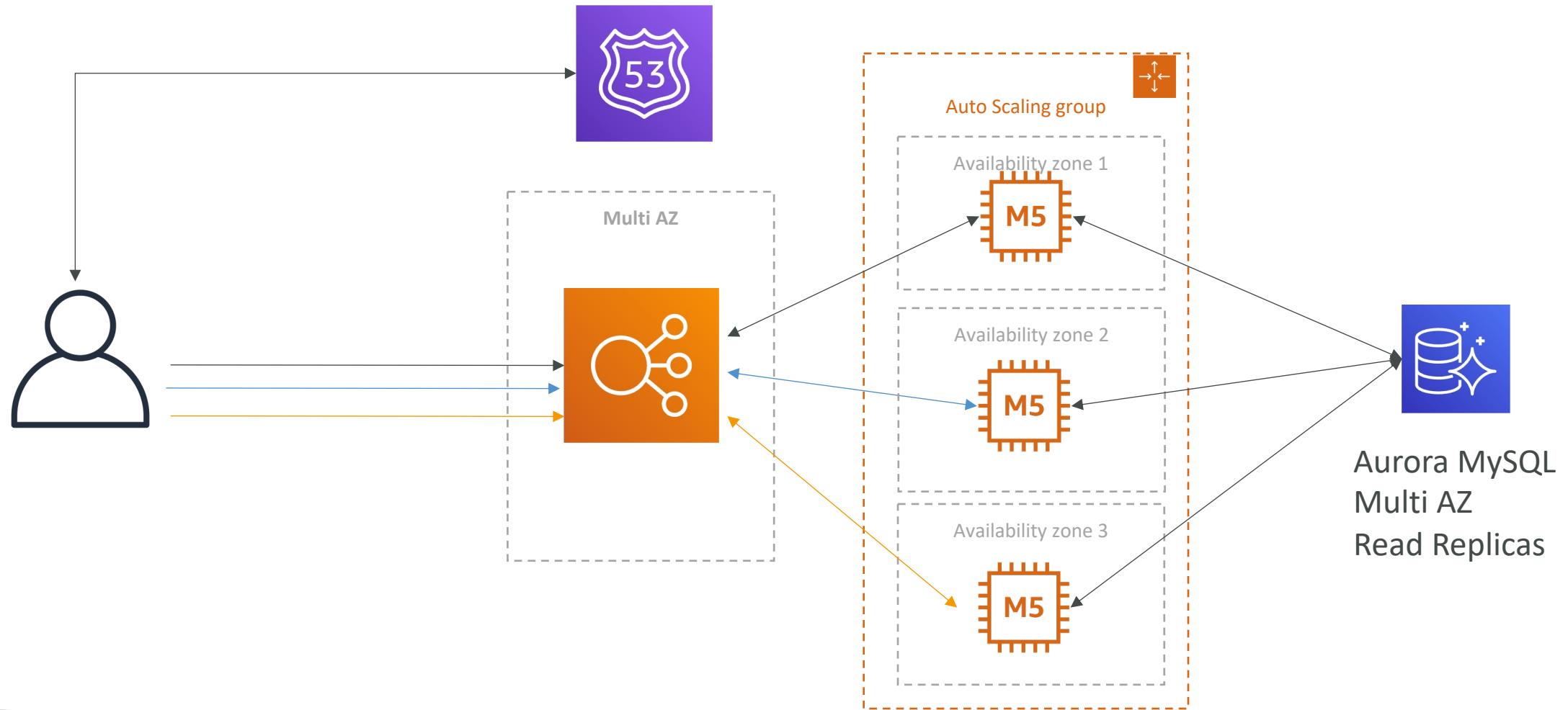
Stateful Web App: MyWordPress.com

RDS layer



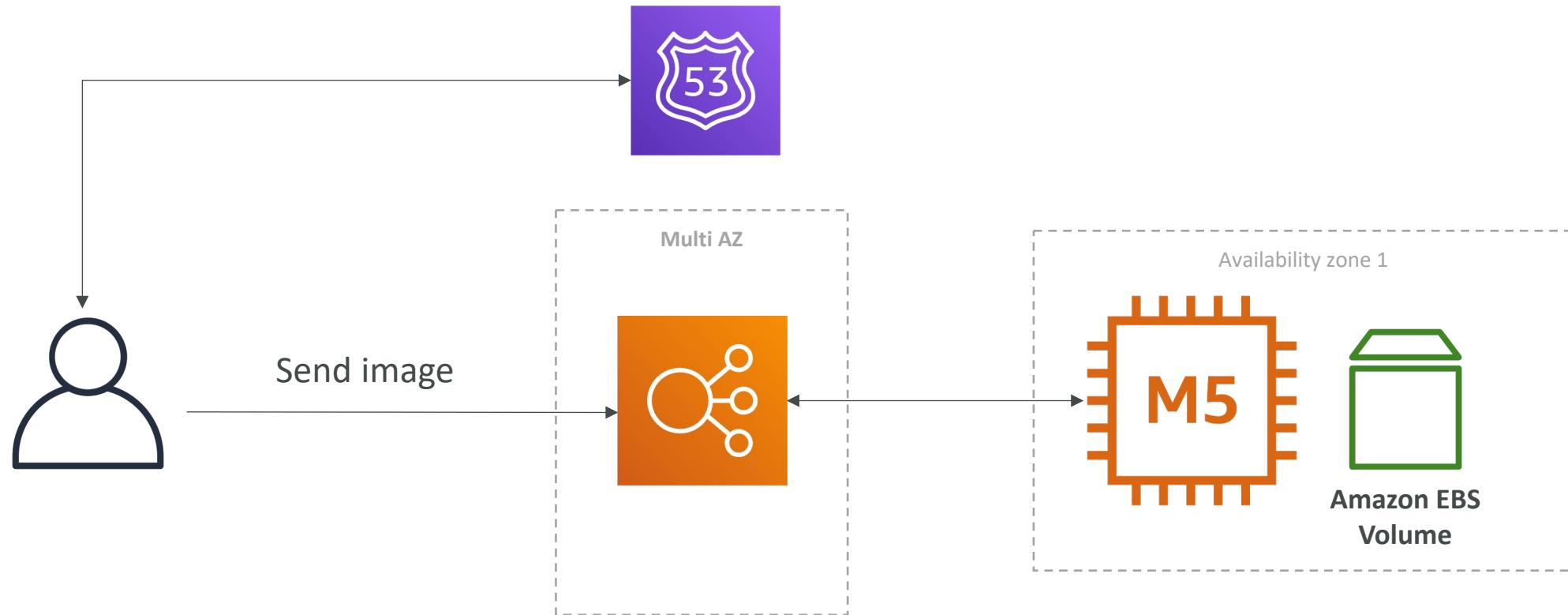
Stateful Web App: MyWordPress.com

Scaling with Aurora: Multi AZ & Read Replicas



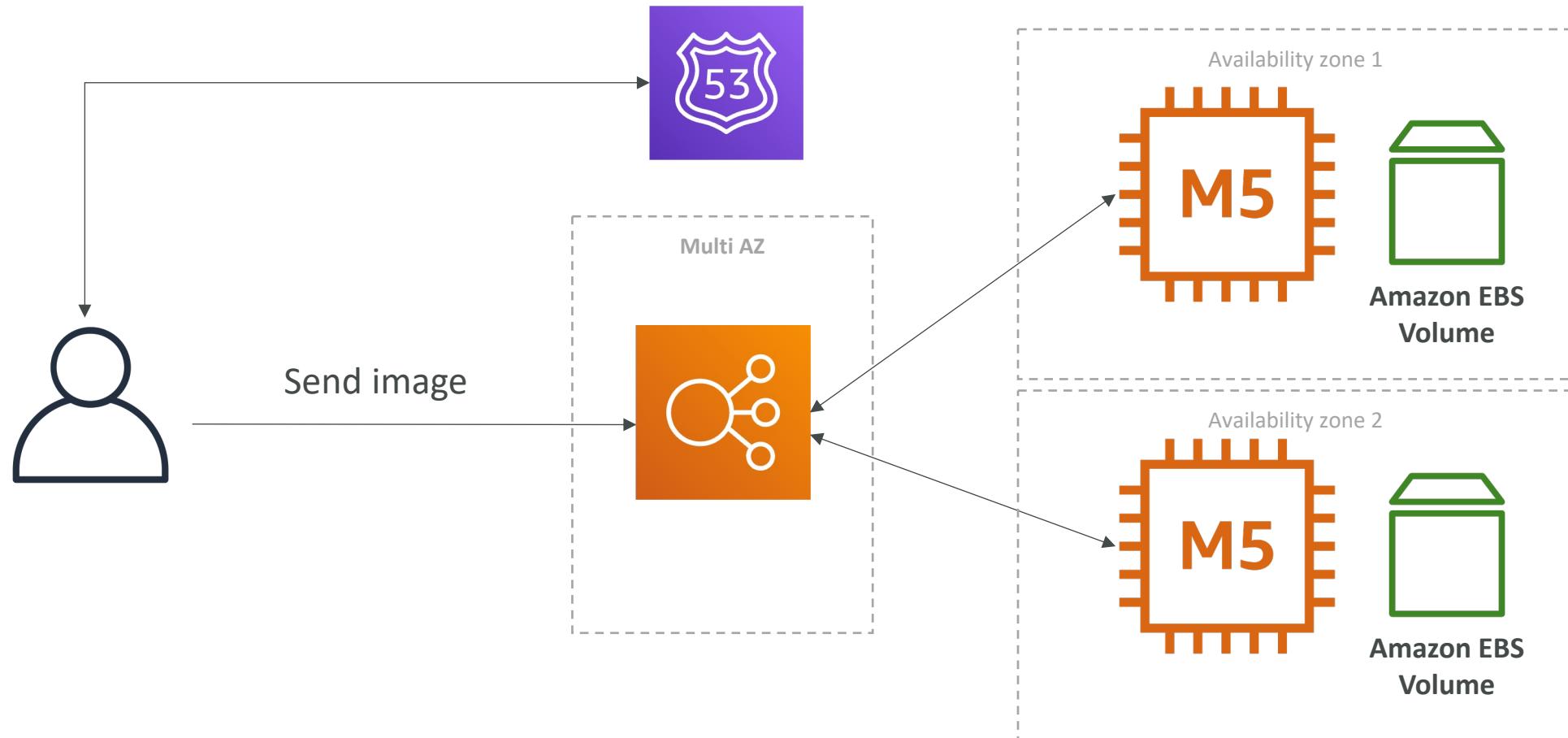
Stateful Web App: MyWordPress.com

Storing images with EBS



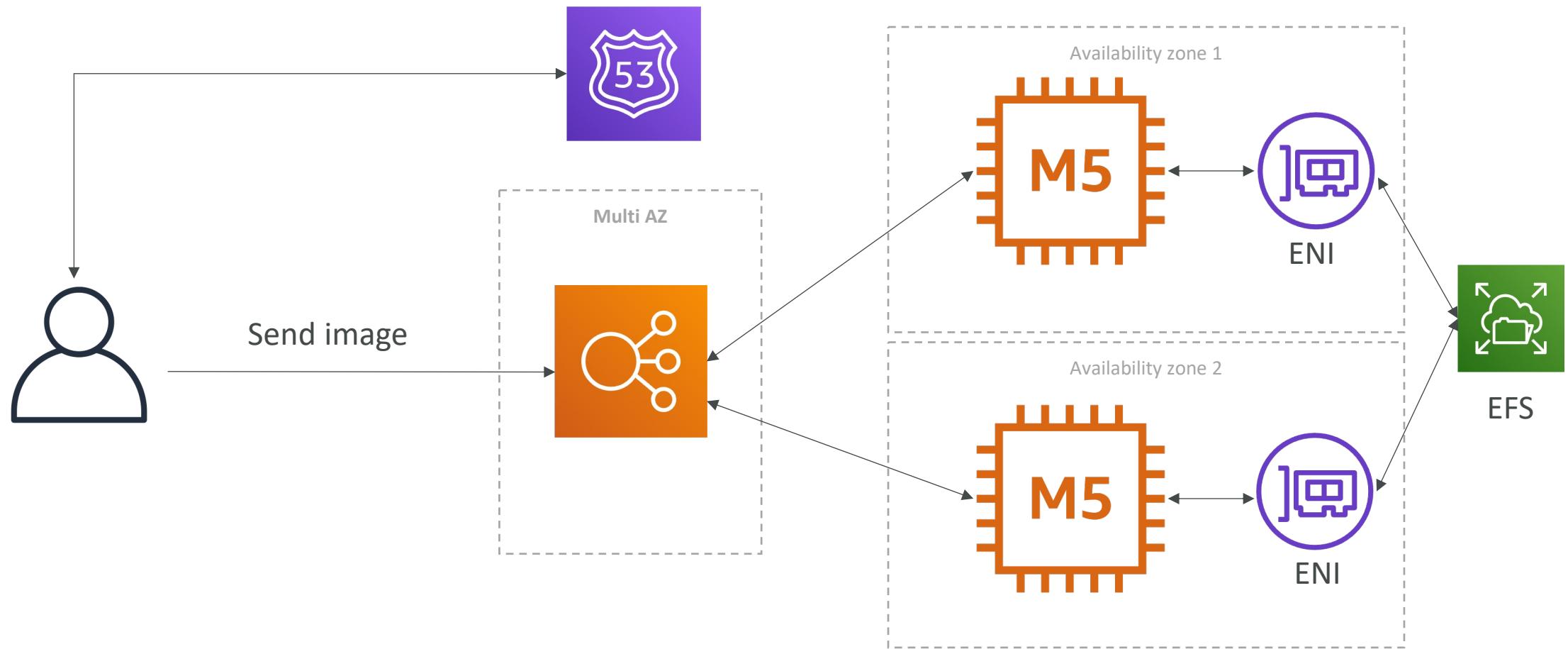
Stateful Web App: MyWordPress.com

Storing images with EBS



Stateful Web App: MyWordPress.com

Storing images with EFS



In this lecture we've discussed...

- Aurora Database to have easy Multi-AZ and Read-Replicas
- Storing data in EBS (single instance application)
- Vs Storing data in EFS (distributed application)

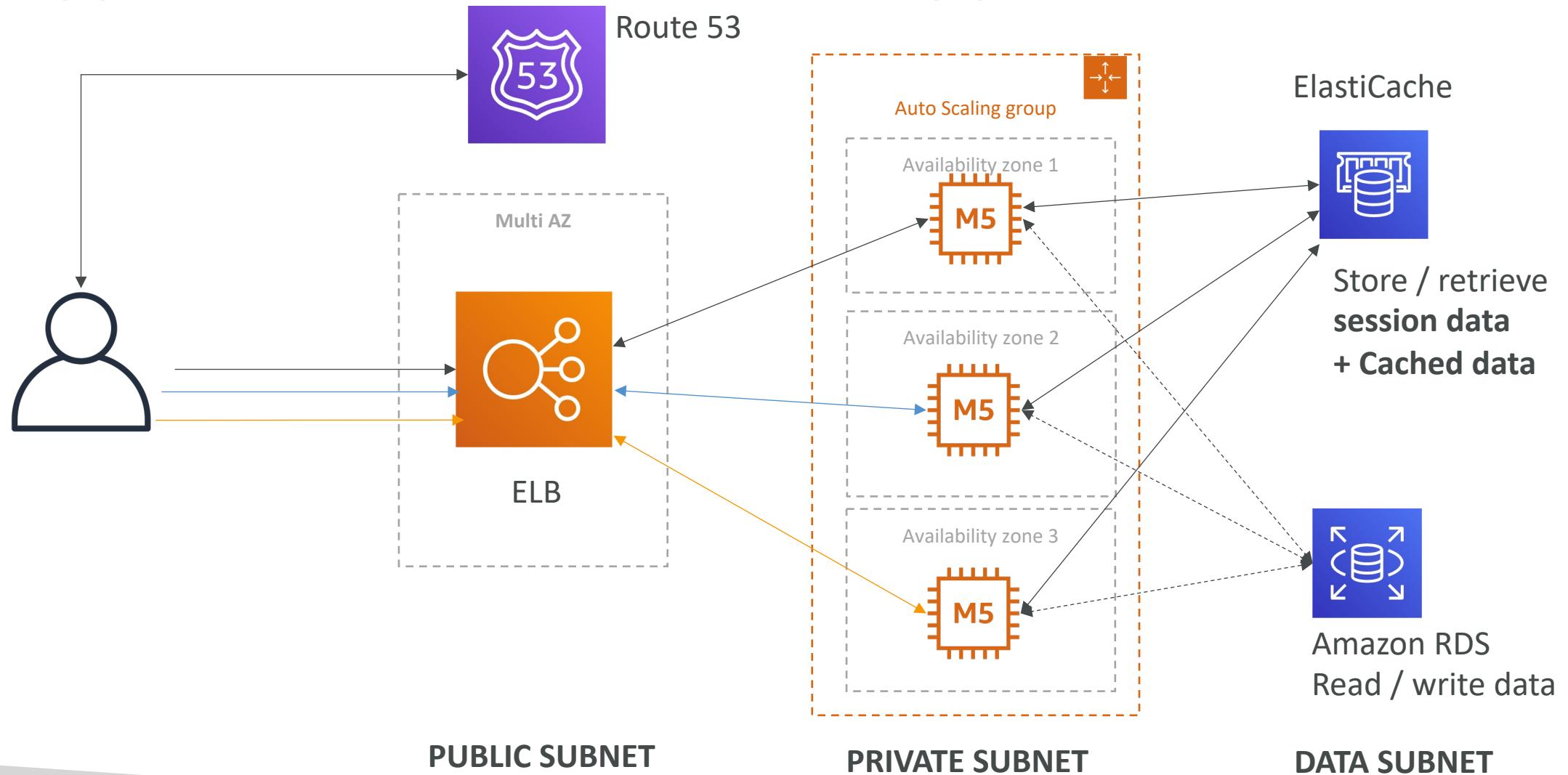
Instantiating Applications quickly

- When launching a full stack (EC2, EBS, RDS), it can take time to:
 - Install applications
 - Insert initial (or recovery) data
 - Configure everything
 - Launch the application
- We can take advantage of the cloud to speed that up!

Instantiating Applications quickly

- EC2 Instances:
 - **Use a Golden AMI:** Install your applications, OS dependencies etc.. beforehand and launch your EC2 instance from the Golden AMI
 - **Bootstrap using User Data:** For dynamic configuration, use User Data scripts
 - **Hybrid:** mix Golden AMI and User Data (Elastic Beanstalk)
- RDS Databases:
 - Restore from a snapshot: the database will have schemas and data ready!
- EBS Volumes:
 - Restore from a snapshot: the disk will already be formatted and have data!

Typical architecture: Web App 3-tier



Developer problems on AWS

- Managing infrastructure
 - Deploying Code
 - Configuring all the databases, load balancers, etc
 - Scaling concerns
-
- Most web apps have the same architecture (ALB + ASG)
 - All the developers want is for their code to run!
 - Possibly, consistently across different applications and environments

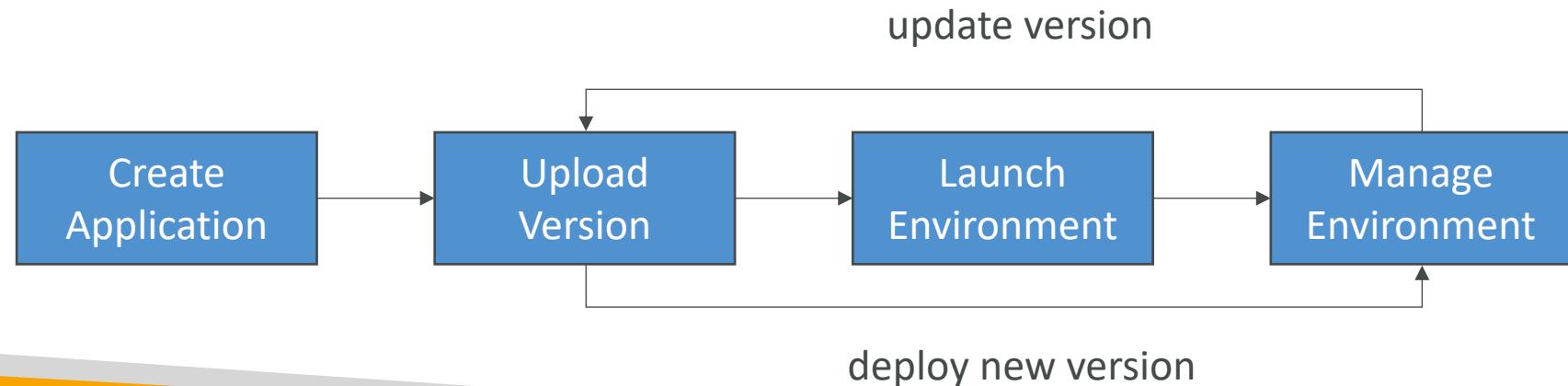
Elastic Beanstalk – Overview



- Elastic Beanstalk is a developer centric view of deploying an application on AWS
- It uses all the component's we've seen before: EC2, ASG, ELB, RDS, ...
- Managed service
 - Automatically handles capacity provisioning, load balancing, scaling, application health monitoring, instance configuration, ...
 - Just the application code is the responsibility of the developer
- We still have full control over the configuration
- Beanstalk is free but you pay for the underlying instances

Elastic Beanstalk – Components

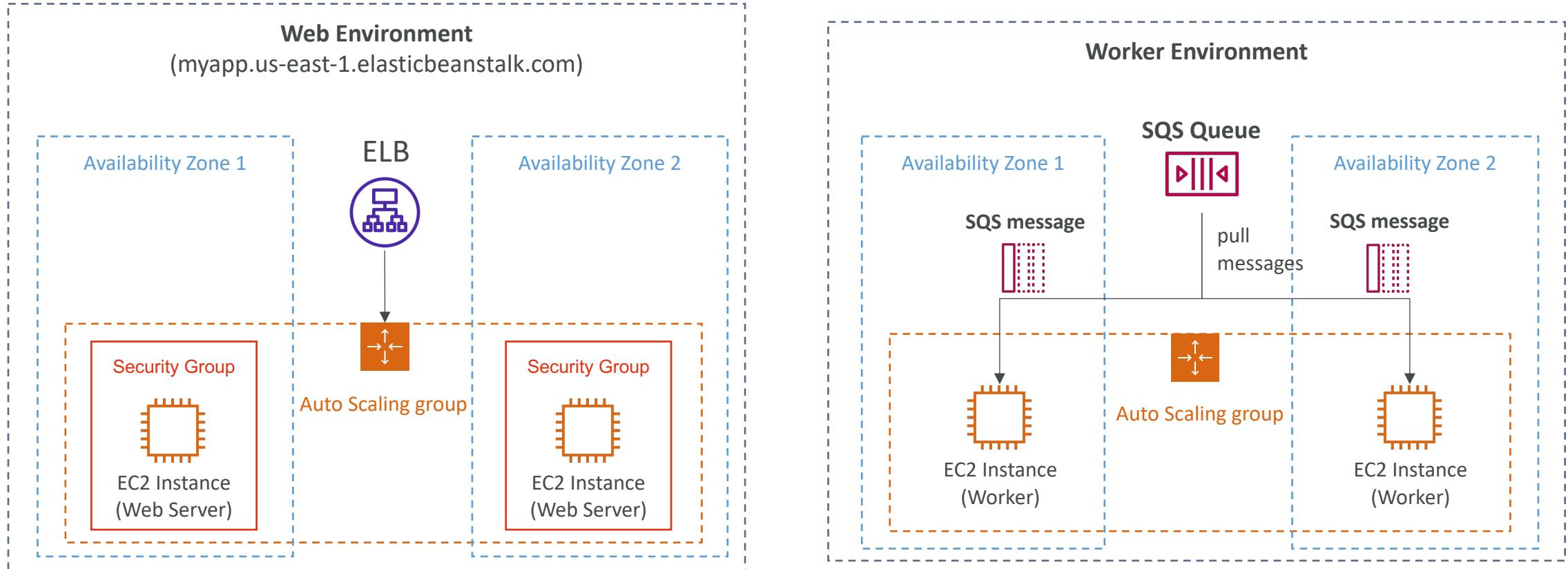
- **Application:** collection of Elastic Beanstalk components (environments, versions, configurations, ...)
- **Application Version:** an iteration of your application code
- **Environment**
 - Collection of AWS resources running an application version (only one application version at a time)
 - **Tiers:** Web Server Environment Tier & Worker Environment Tier
 - You can create multiple environments (dev, test, prod, ...)



Elastic Beanstalk – Supported Platforms

- Go
- Java SE
- Java with Tomcat
- .NET Core on Linux
- .NET on Windows Server
- Node.js
- PHP
- Python
- Ruby
- Packer Builder
- Single Container Docker
- Multi-container Docker
- Preconfigured Docker
- If not supported, you can write your custom platform (advanced)

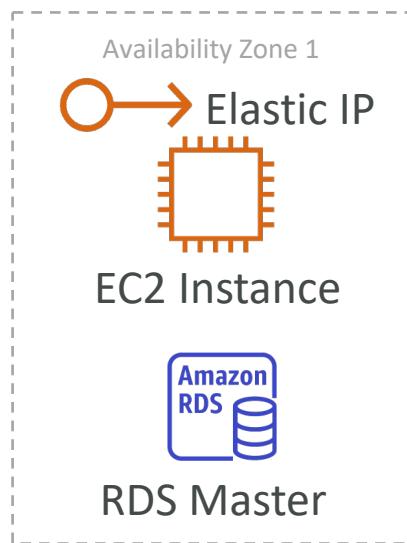
Web Server Tier vs. Worker Tier



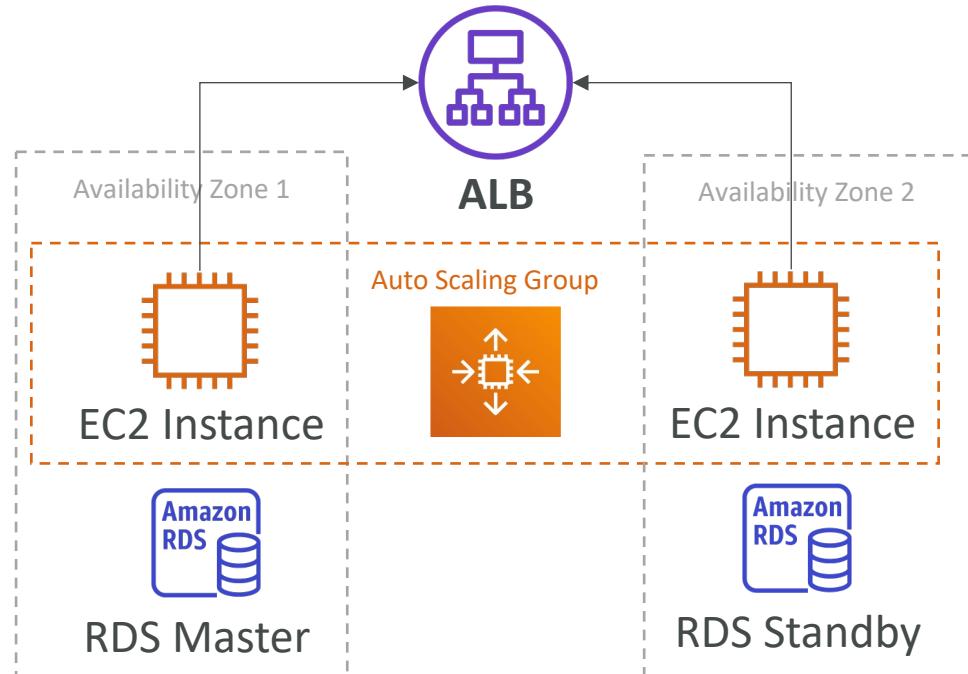
- Scale based on the number of SQS messages
- Can push messages to SQS queue from another Web Server Tier

Elastic Beanstalk Deployment Modes

Single Instance
Great for dev



High Availability with Load Balancer
Great for prod



Amazon S3 Section



Section introduction

- Amazon S3 is one of the main building blocks of AWS
- It's advertised as "infinitely scaling" storage

- Many websites use Amazon S3 as a backbone
- Many AWS services use Amazon S3 as an integration as well

- We'll have a step-by-step approach to S3

Amazon S3 Use cases

- Backup and storage
- Disaster Recovery
- Archive
- Hybrid Cloud storage
- Application hosting
- Media hosting
- Data lakes & big data analytics
- Software delivery
- Static website



Nasdaq stores 7 years of data into S3 Glacier



Sysco runs analytics on its data and gain business insights

Amazon S3 - Buckets

- Amazon S3 allows people to store objects (files) in “buckets” (directories)
- Buckets must have a **globally unique name** (across all regions all accounts)
- Buckets are defined at the region level
- S3 looks like a global service but buckets are created in a region
- Naming convention
 - No uppercase, No underscore
 - 3-63 characters long
 - Not an IP
 - Must start with lowercase letter or number
 - Must NOT start with the prefix `xn--`
 - Must NOT end with the suffix `-s3alias`



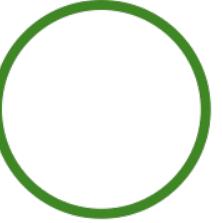
S3 Bucket

Amazon S3 - Objects

- Objects (files) have a Key
- The **key** is the **FULL** path:
 - s3://my-bucket/**my_file.txt**
 - s3://my-bucket/**my_folder1/another_folder/my_file.txt**
- The key is composed of **prefix** + **object name**
 - s3://my-bucket/**my_folder1/another_folder**/**my_file.txt**
- There's no concept of "directories" within buckets (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes ("/")



with Objects



Amazon S3 – Objects (cont.)

- Object values are the content of the body:
 - Max. Object Size is 5TB (5000GB)
 - If uploading more than 5GB, must use “multi-part upload”
- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

Amazon S3 – Security

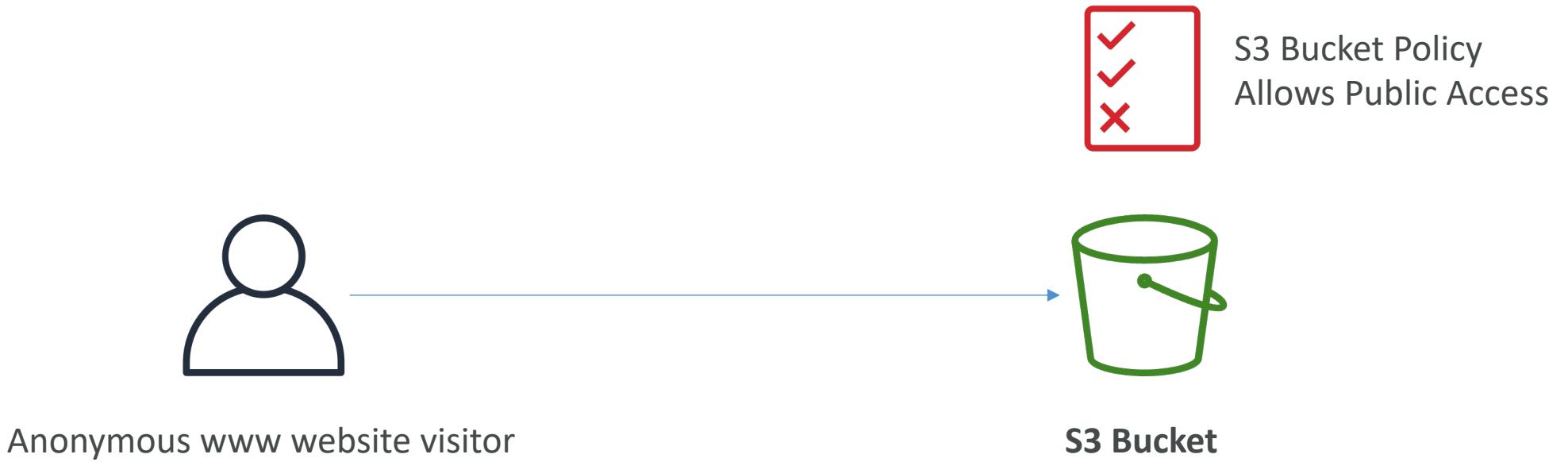
- User-Based
 - IAM Policies – which API calls should be allowed for a specific user from IAM
- Resource-Based
 - Bucket Policies – bucket wide rules from the S3 console - allows cross account
 - Object Access Control List (ACL) – finer grain (can be disabled)
 - Bucket Access Control List (ACL) – less common (can be disabled)
- **Note:** an IAM principal can access an S3 object if
 - The user IAM permissions ALLOW it OR the resource policy ALLOWS it
 - AND there's no explicit DENY
- **Encryption:** encrypt objects in Amazon S3 using encryption keys

S3 Bucket Policies

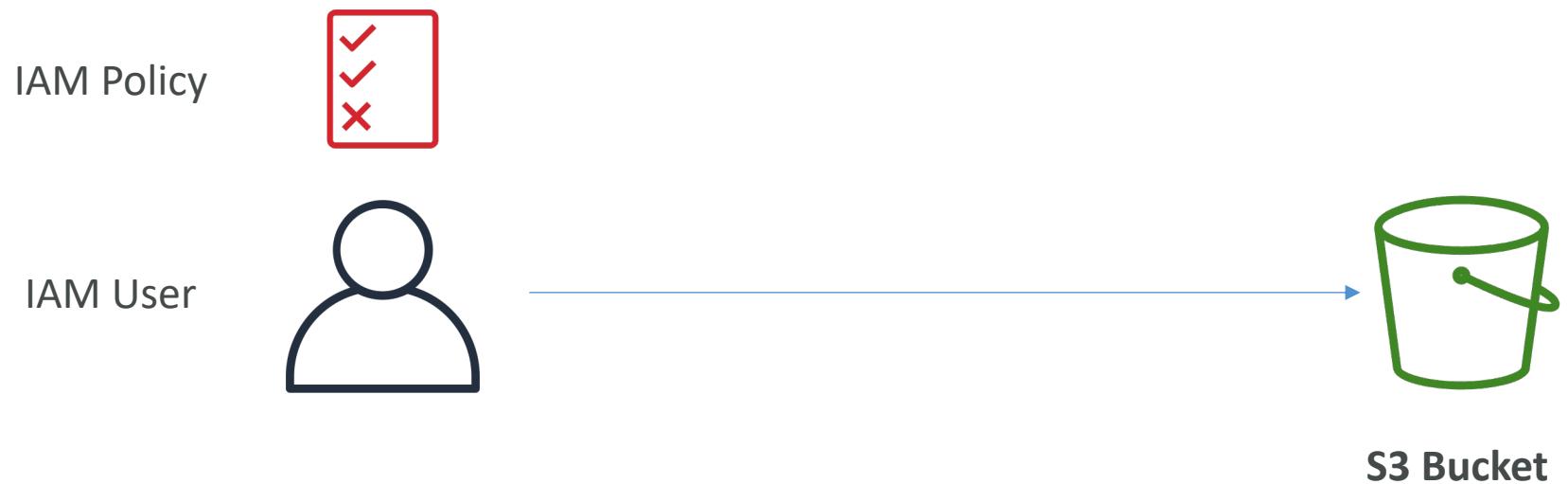
- JSON based policies
 - Resources: buckets and objects
 - Effect: Allow / Deny
 - Actions: Set of API to Allow or Deny
 - Principal: The account or user to apply the policy to
- Use S3 bucket for policy to:
 - Grant public access to the bucket
 - Force objects to be encrypted at upload
 - Grant access to another account (Cross Account)

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "PublicRead",  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": [  
        "s3:GetObject"  
      ],  
      "Resource": [  
        "arn:aws:s3:::examplebucket/*"  
      ]  
    }  
  ]  
}
```

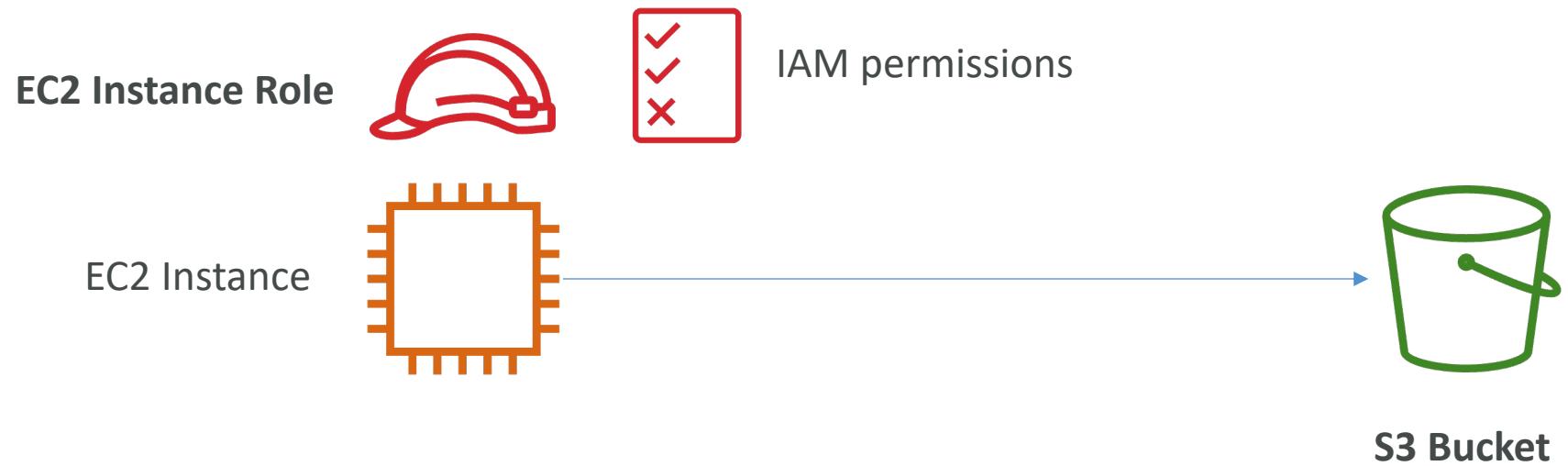
Example: Public Access - Use Bucket Policy



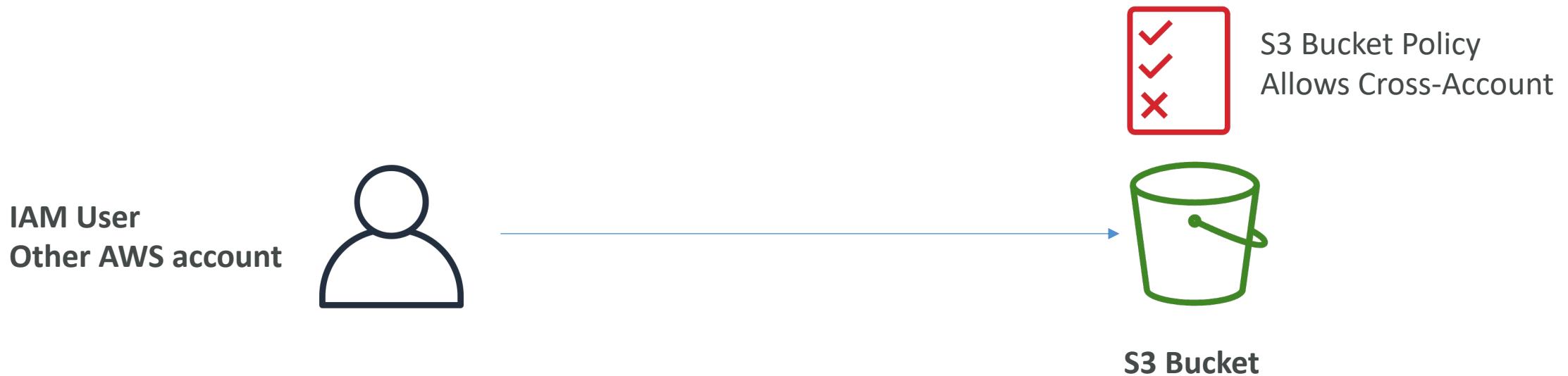
Example: User Access to S3 – IAM permissions



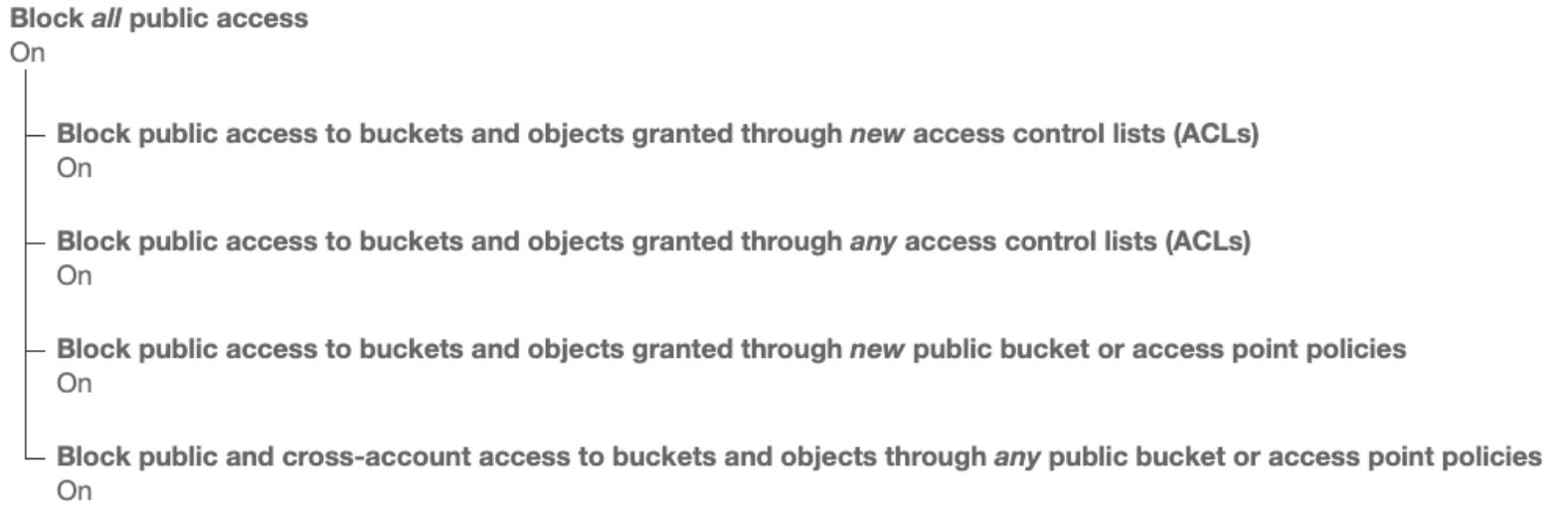
Example: EC2 instance access - Use IAM Roles



Advanced: Cross-Account Access – Use Bucket Policy



Bucket settings for Block Public Access



- These settings were created to prevent company data leaks
- If you know your bucket should never be public, leave these on
- Can be set at the account level

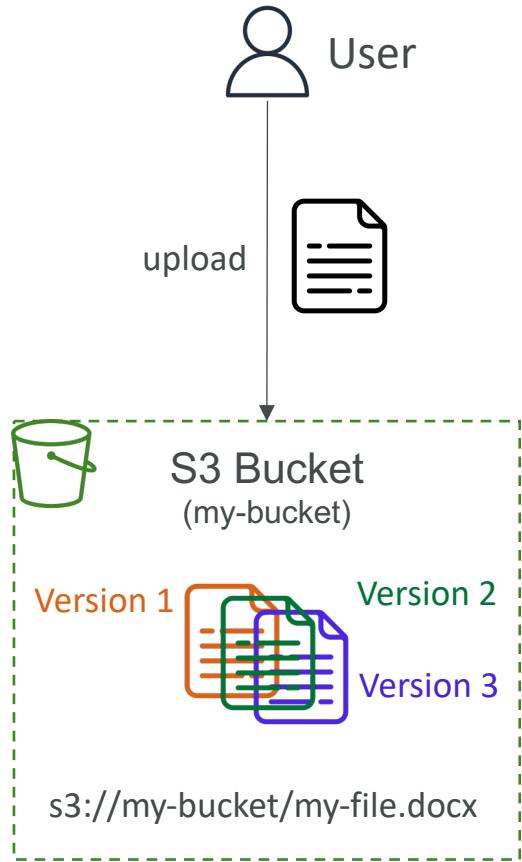
Amazon S3 – Static Website Hosting

- S3 can host static websites and have them accessible on the Internet
- The website URL will be (depending on the region)
 - `http://bucket-name.s3-website-aws-region.amazonaws.com`
 - OR
 - `http://bucket-name.s3-website.aws-region.amazonaws.com`
- If you get a 403 Forbidden error, make sure the bucket policy allows public reads!



Amazon S3 - Versioning

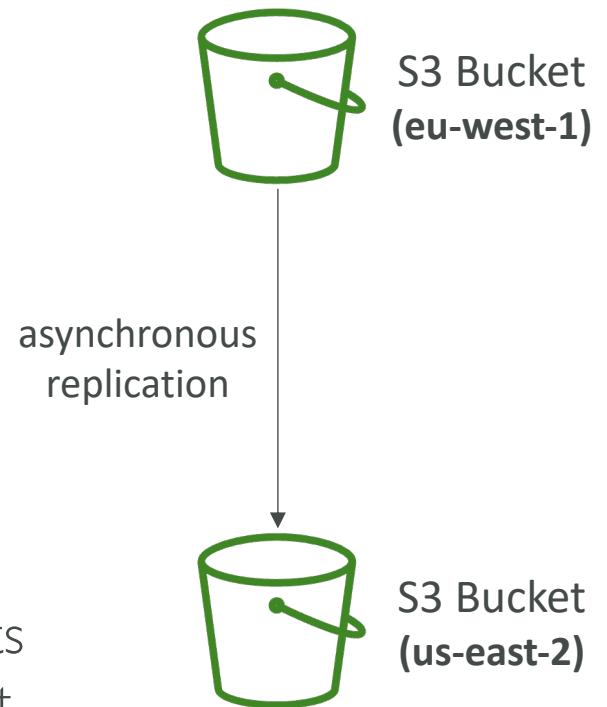
- You can version your files in Amazon S3
- It is enabled at the **bucket level**
- Same key overwrite will change the “version”: 1, 2, 3....
- It is best practice to version your buckets
 - Protect against unintended deletes (ability to restore a version)
 - Easy roll back to previous version
- Notes:
 - Any file that is not versioned prior to enabling versioning will have version “null”
 - Suspending versioning does not delete the previous versions



Amazon S3 – Replication (CRR & SRR)



- Must enable Versioning in source and destination buckets
- Cross-Region Replication (CRR)
- Same-Region Replication (SRR)
- Buckets can be in different AWS accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3
- Use cases:
 - CRR – compliance, lower latency access, replication across accounts
 - SRR – log aggregation, live replication between production and test accounts



Amazon S3 – Replication (Notes)

- After you enable Replication, only new objects are replicated
- Optionally, you can replicate existing objects using **S3 Batch Replication**
 - Replicates existing objects and objects that failed replication
- For DELETE operations
 - Can replicate delete markers from source to target (optional setting)
 - Deletions with a version ID are not replicated (to avoid malicious deletes)
- There is no “chaining” of replication
 - If bucket 1 has replication into bucket 2, which has replication into bucket 3
 - Then objects created in bucket 1 are not replicated to bucket 3

S3 Storage Classes

- Amazon S3 Standard - General Purpose
- Amazon S3 Standard-Infrequent Access (IA)
- Amazon S3 One Zone-Infrequent Access
- Amazon S3 Glacier Instant Retrieval
- Amazon S3 Glacier Flexible Retrieval
- Amazon S3 Glacier Deep Archive
- Amazon S3 Intelligent Tiering
- Can move between classes manually or using S3 Lifecycle configurations

S3 Durability and Availability

- Durability:
 - High durability (99.99999999%, 11 9's) of objects across multiple AZ
 - If you store 10,000,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000 years
 - Same for all storage classes
- Availability:
 - Measures how readily available a service is
 - Varies depending on storage class
 - Example: S3 standard has 99.99% availability = not available 53 minutes a year



S3 Standard – General Purpose

- 99.99% Availability
 - Used for frequently accessed data
 - Low latency and high throughput
 - Sustain 2 concurrent facility failures
-
- Use Cases: Big Data analytics, mobile & gaming applications, content distribution...

S3 Storage Classes – Infrequent Access

- For data that is less frequently accessed, but requires rapid access when needed
- Lower cost than S3 Standard
- Amazon S3 Standard-Infrequent Access (S3 Standard-IA)
 - 99.9% Availability
 - Use cases: Disaster Recovery, backups
- Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA)
 - High durability (99.99999999%) in a single AZ; data lost when AZ is destroyed
 - 99.5% Availability
 - Use Cases: Storing secondary backup copies of on-premises data, or data you can recreate



Amazon S3 Glacier Storage Classes

- Low-cost object storage meant for archiving / backup
- Pricing: price for storage + object retrieval cost
- **Amazon S3 Glacier Instant Retrieval**
 - Millisecond retrieval, great for data accessed once a quarter
 - Minimum storage duration of 90 days
- **Amazon S3 Glacier Flexible Retrieval** (formerly Amazon S3 Glacier):
 - Expedited (1 to 5 minutes), Standard (3 to 5 hours), Bulk (5 to 12 hours) – free
 - Minimum storage duration of 90 days
- **Amazon S3 Glacier Deep Archive** – for long term storage:
 - Standard (12 hours), Bulk (48 hours)
 - Minimum storage duration of 180 days





S3 Intelligent-Tiering

- Small monthly monitoring and auto-tiering fee
 - Moves objects automatically between Access Tiers based on usage
 - There are no retrieval charges in S3 Intelligent-Tiering
-
- *Frequent Access tier (automatic)*: default tier
 - *Infrequent Access tier (automatic)*: objects not accessed for 30 days
 - *Archive Instant Access tier (automatic)*: objects not accessed for 90 days
 - *Archive Access tier (optional)*: configurable from 90 days to 700+ days
 - *Deep Archive Access tier (optional)*: config. from 180 days to 700+ days

S3 Storage Classes Comparison

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 3	>= 3	>= 3	1	>= 3	>= 3	>= 3
Min. Storage Duration Charge	None	None	30 Days	30 Days	90 Days	90 Days	180 Days
Min. Billable Object Size	None	None	128 KB	128 KB	128 KB	40 KB	40 KB
Retrieval Fee	None	None	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved

<https://aws.amazon.com/s3/storage-classes/>

S3 Storage Classes – Price Comparison

Example: us-east-1

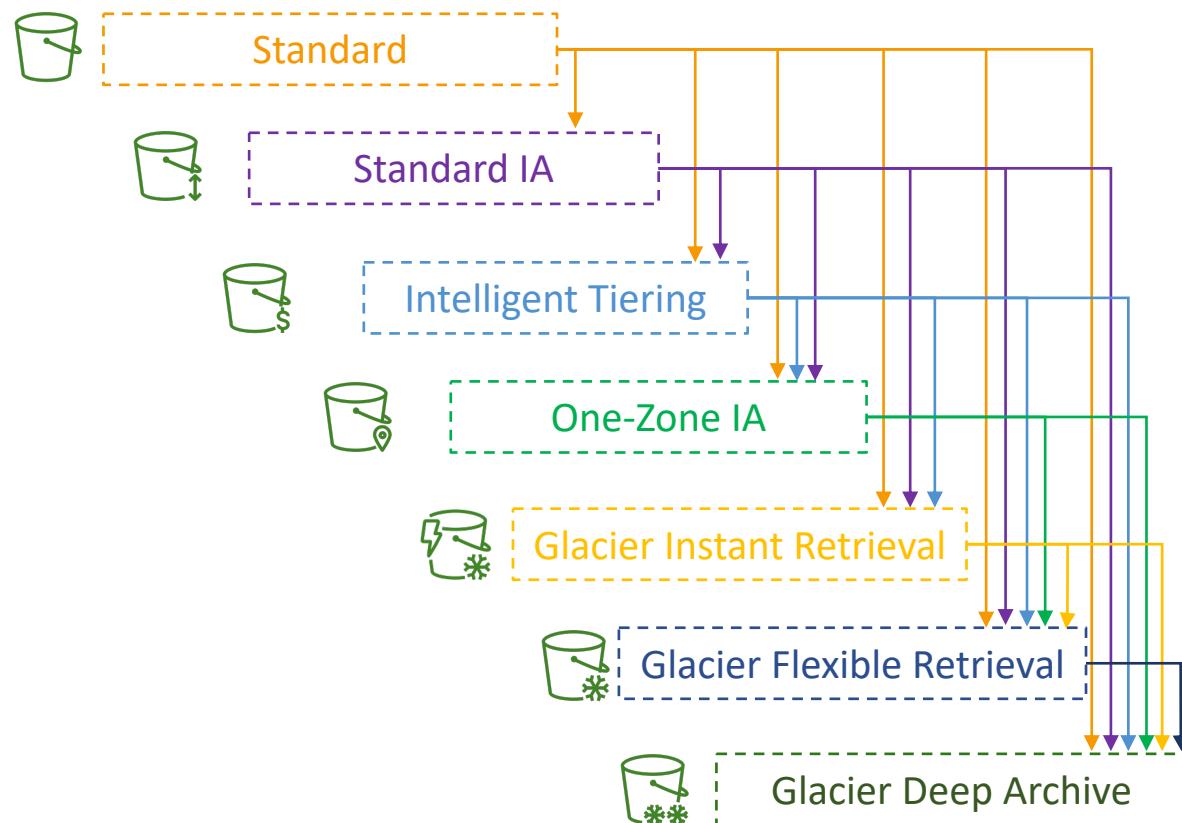
	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Storage Cost (per GB per month)	\$0.023	\$0.0025 - \$0.023	\$0.0125	\$0.01	\$0.004	\$0.0036	\$0.00099
Retrieval Cost (per 1000 request)	GET: \$0.0004 POST: \$0.005	GET: \$0.0004 POST: \$0.005	GET: \$0.001 POST: \$0.01	GET: \$0.001 POST: \$0.01	GET: \$0.01 POST: \$0.02	GET: \$0.0004 POST: \$0.03 Expedited: \$10 Standard: \$0.05 Bulk: free	GET: \$0.0004 POST: \$0.05 Standard: \$0.10 Bulk: \$0.025
Retrieval Time	Instantaneous						Expedited (1 – 5 mins) Standard (3 – 5 hours) Bulk (5 – 12 hours)
Monitoring Cost (pet 1000 objects)		\$0.0025					

<https://aws.amazon.com/s3/pricing/>

Advanced S3

Amazon S3 – Moving between Storage Classes

- You can transition objects between storage classes
- For infrequently accessed object, move them to **Standard IA**
- For archive objects that you don't need fast access to, move them to **Glacier or Glacier Deep Archive**
- Moving objects can be automated using a **Lifecycle Rules**





Amazon S3 – Lifecycle Rules

- **Transition Actions** – configure objects to transition to another storage class
 - Move objects to Standard IA class 60 days after creation
 - Move to Glacier for archiving after 6 months
- **Expiration actions** – configure objects to expire (delete) after some time
 - Access log files can be set to delete after a 365 days
 - **Can be used to delete old versions of files (if versioning is enabled)**
 - Can be used to delete incomplete Multi-Part uploads
- Rules can be created for a certain prefix (example: s3://mybucket/mp3/*)
- Rules can be created for certain objects Tags (example: Department: Finance)

Amazon S3 – Lifecycle Rules (Scenario I)

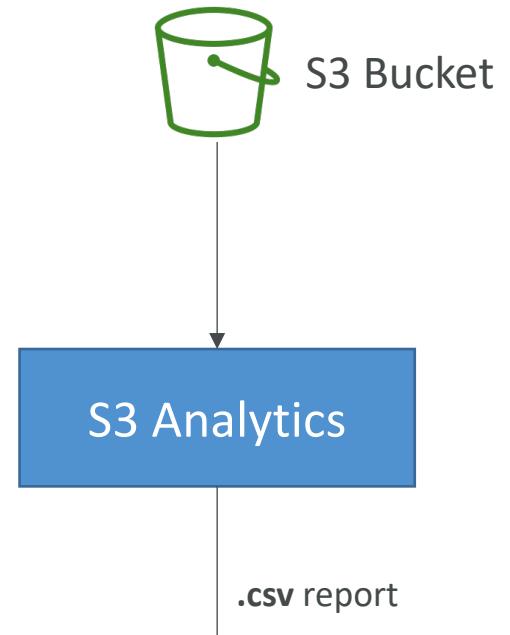
- Your application on EC2 creates images thumbnails after profile photos are uploaded to Amazon S3. These thumbnails can be easily recreated, and only need to be kept for 60 days. The source images should be able to be immediately retrieved for these 60 days, and afterwards, the user can wait up to 6 hours. How would you design this?
- S3 source images can be on **Standard**, with a lifecycle configuration to transition them to **Glacier** after 60 days
- S3 thumbnails can be on **One-Zone IA**, with a lifecycle configuration to expire them (delete them) after 60 days

Amazon S3 – Lifecycle Rules (Scenario 2)

- A rule in your company states that you should be able to recover your deleted S3 objects immediately for 30 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.
- Enable **S3 Versioning** in order to have object versions, so that “deleted objects” are in fact hidden by a “delete marker” and can be recovered
- Transition the “noncurrent versions” of the object to **Standard IA**
- Transition afterwards the “noncurrent versions” to **Glacier Deep Archive**

Amazon S3 Analytics – Storage Class Analysis

- Help you decide when to transition objects to the right storage class
- Recommendations for **Standard** and **Standard IA**
 - Does NOT work for One-Zone IA or Glacier
- Report is updated daily
- 24 to 48 hours to start seeing data analysis
- Good first step to put together Lifecycle Rules (or improve them)!

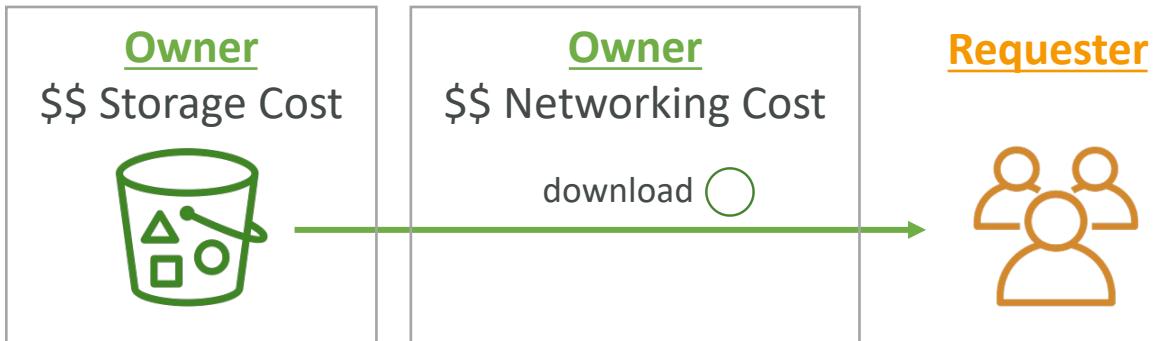


Date	StorageClass	ObjectAge
8/22/2022	STANDARD	000-014
8/25/2022	STANDARD	030-044
9/6/2022	STANDARD	120-149

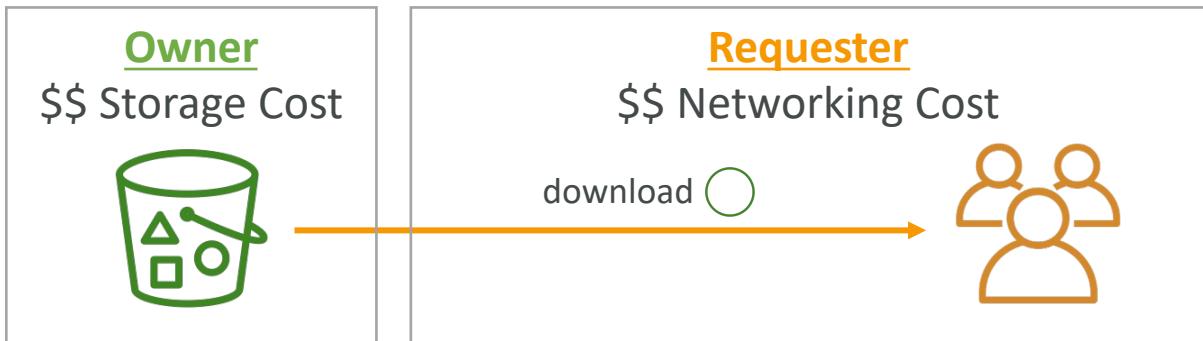
S3 – Requester Pays

- In general, bucket owners pay for all Amazon S3 storage and data transfer costs associated with their bucket
- With Requester Pays buckets, the requester instead of the bucket owner pays the cost of the request and the data download from the bucket
- Helpful when you want to share large datasets with other accounts
- The requester must be authenticated in AWS (cannot be anonymous)

Standard Bucket

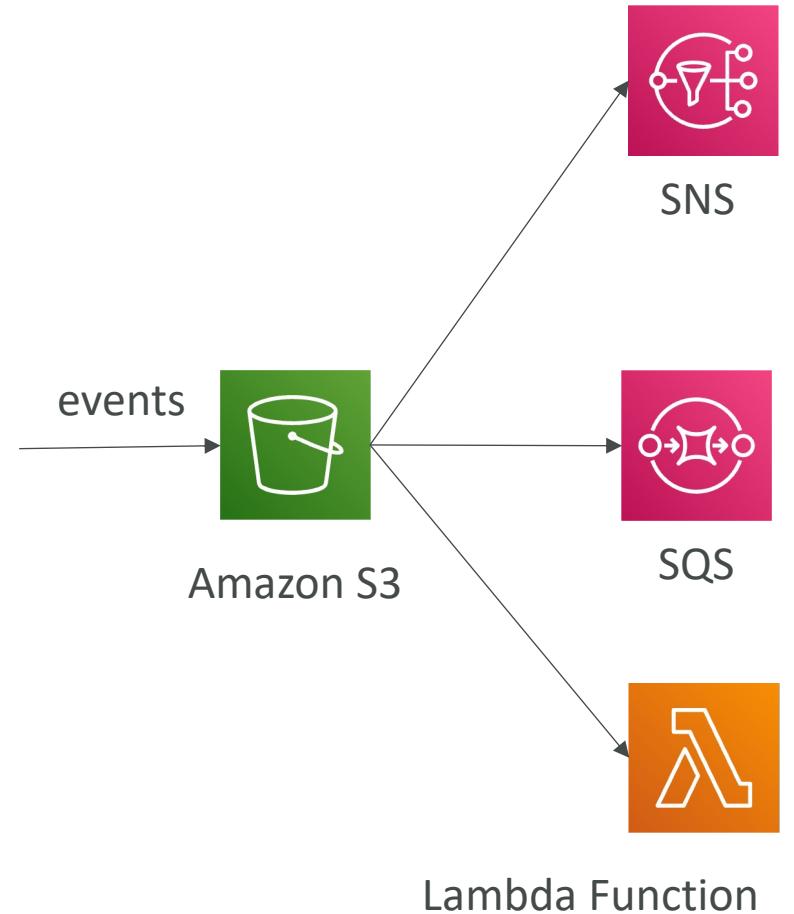


Requester Pays Bucket

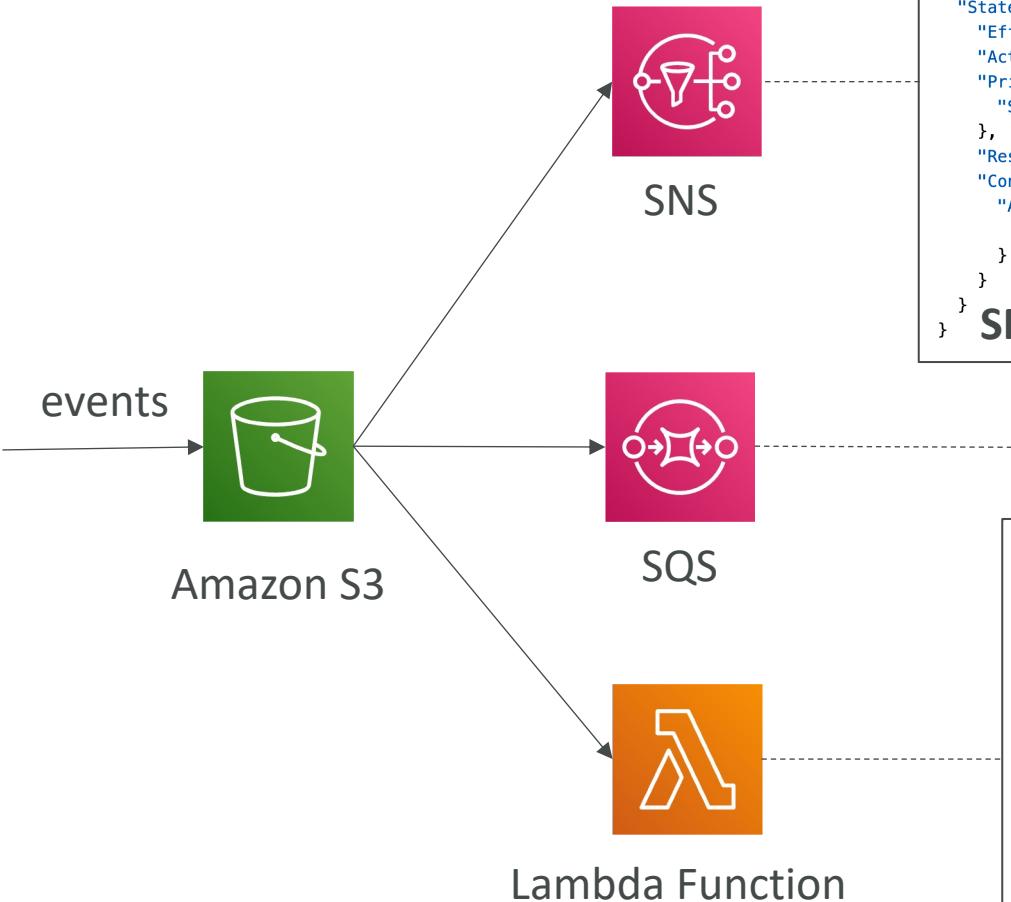


S3 Event Notifications

- S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication...
- Object name filtering possible (*.jpg)
- Use case: generate thumbnails of images uploaded to S3
- Can create as many “S3 events” as desired
- S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer



S3 Event Notifications – IAM Permissions



```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "SNS:Publish",  
        "Principal": {  
            "Service": "s3.amazonaws.com"  
        },  
        "Resource": "arn:aws:sns:us-east-1:123456789012:MyTopic",  
        "Condition": {  
            "ArnLike": {  
                "aws:SourceArn": "arn:aws:s3:::MyBucket"  
            }  
        }  
    }  
}
```

SNS Resource (Access) Policy

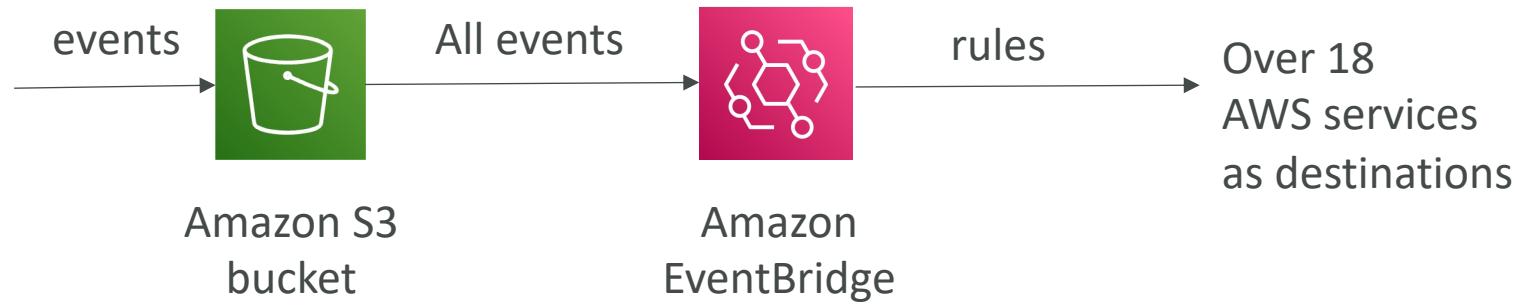
```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "SQS:SendMessage",  
        "Principal": {  
            "Service": "s3.amazonaws.com"  
        },  
        "Resource": "arn:aws:sqs:us-east-1:123456789012:MyQueue",  
        "Condition": {  
            "ArnLike": {  
                "aws:SourceArn": "arn:aws:s3:::MyBucket"  
            }  
        }  
    }  
}
```

SQS Resource (Access) Policy

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "lambda:InvokeFunction",  
        "Principal": {  
            "Service": "s3.amazonaws.com"  
        },  
        "Resource": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",  
        "Condition": {  
            "ArnLike": {  
                "AWS:SourceArn": "arn:aws:s3:::MyBucket"  
            }  
        }  
    }  
}
```

Lambda Resource Policy

S3 Event Notifications with Amazon EventBridge



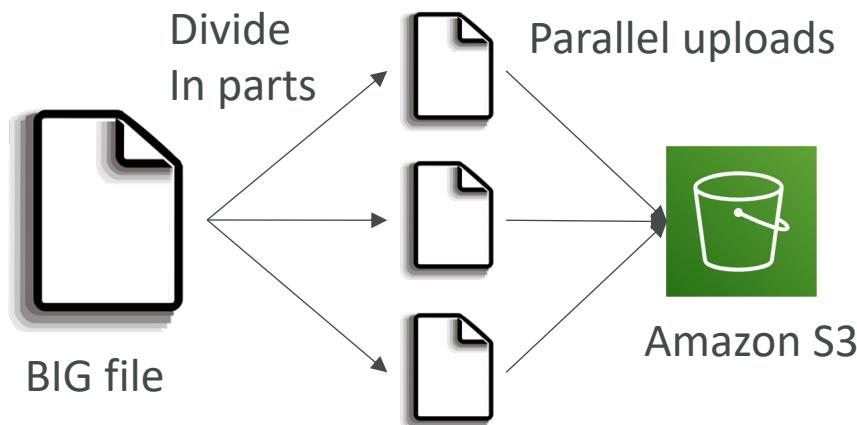
- Advanced filtering options with JSON rules (metadata, object size, name...)
- Multiple Destinations – ex Step Functions, Kinesis Streams / Firehose...
- EventBridge Capabilities – Archive, Replay Events, Reliable delivery

S3 – Baseline Performance

- Amazon S3 automatically scales to high request rates, latency 100-200 ms
- Your application can achieve at least 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in a bucket.
- There are no limits to the number of prefixes in a bucket.
- Example (object path => prefix):
 - bucket/folder1/sub1/file => /folder1/sub1/
 - bucket/folder1/sub2/file => /folder1/sub2/
 - bucket/1/file => /1/
 - bucket/2/file => /2/
- If you spread reads across all four prefixes evenly, you can achieve 22,000 requests per second for GET and HEAD

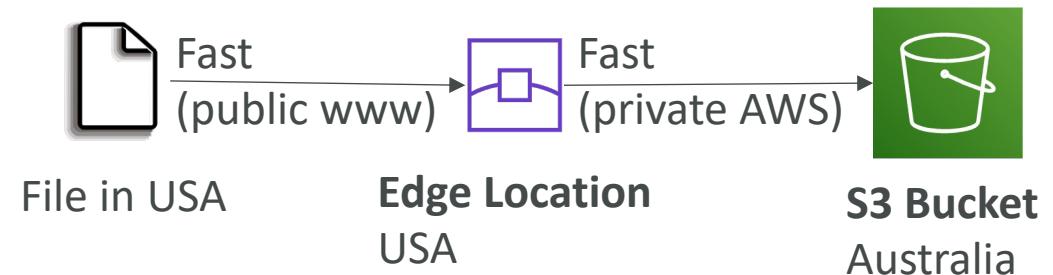
S3 Performance

- Multi-Part upload:
 - recommended for files > 100MB, must use for files > 5GB
 - Can help parallelize uploads (speed up transfers)



- S3 Transfer Acceleration

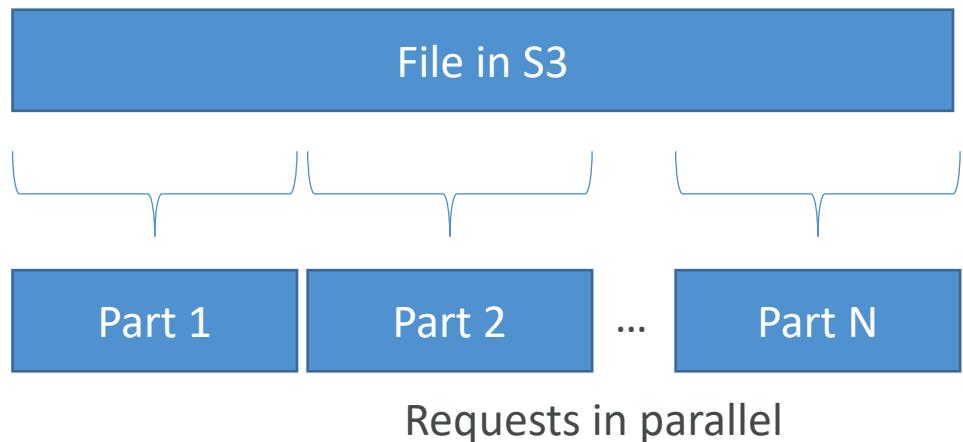
- Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region
- Compatible with multi-part upload



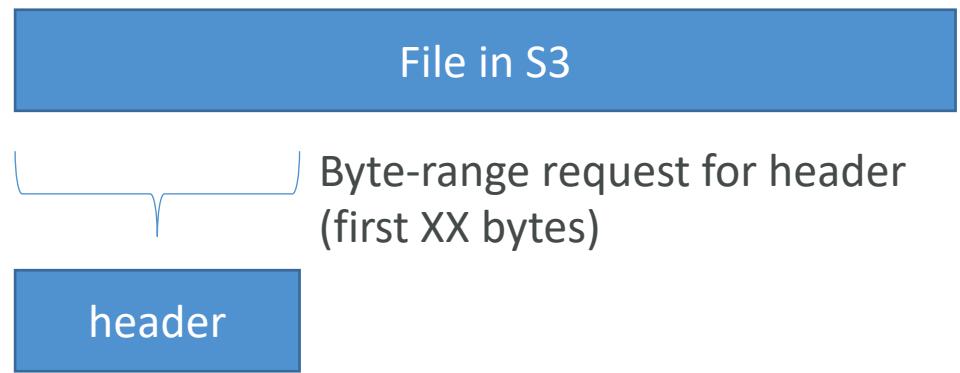
S3 Performance – S3 Byte-Range Fetches

- Parallelize GETs by requesting specific byte ranges
- Better resilience in case of failures

Can be used to speed up downloads

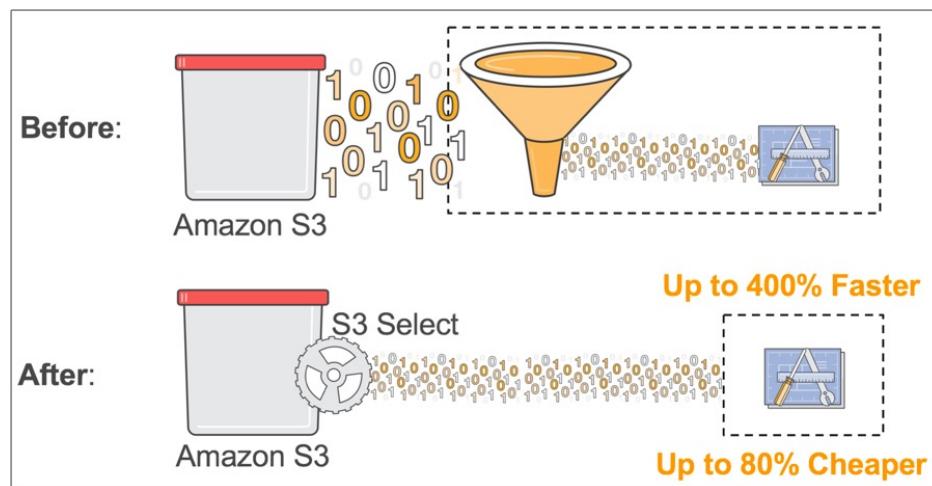


Can be used to retrieve only partial data (for example the head of a file)



S3 Select & Glacier Select

- Retrieve less data using SQL by performing **server-side filtering**
- Can filter by rows & columns (simple SQL statements)
- Less network transfer, less CPU cost client-side

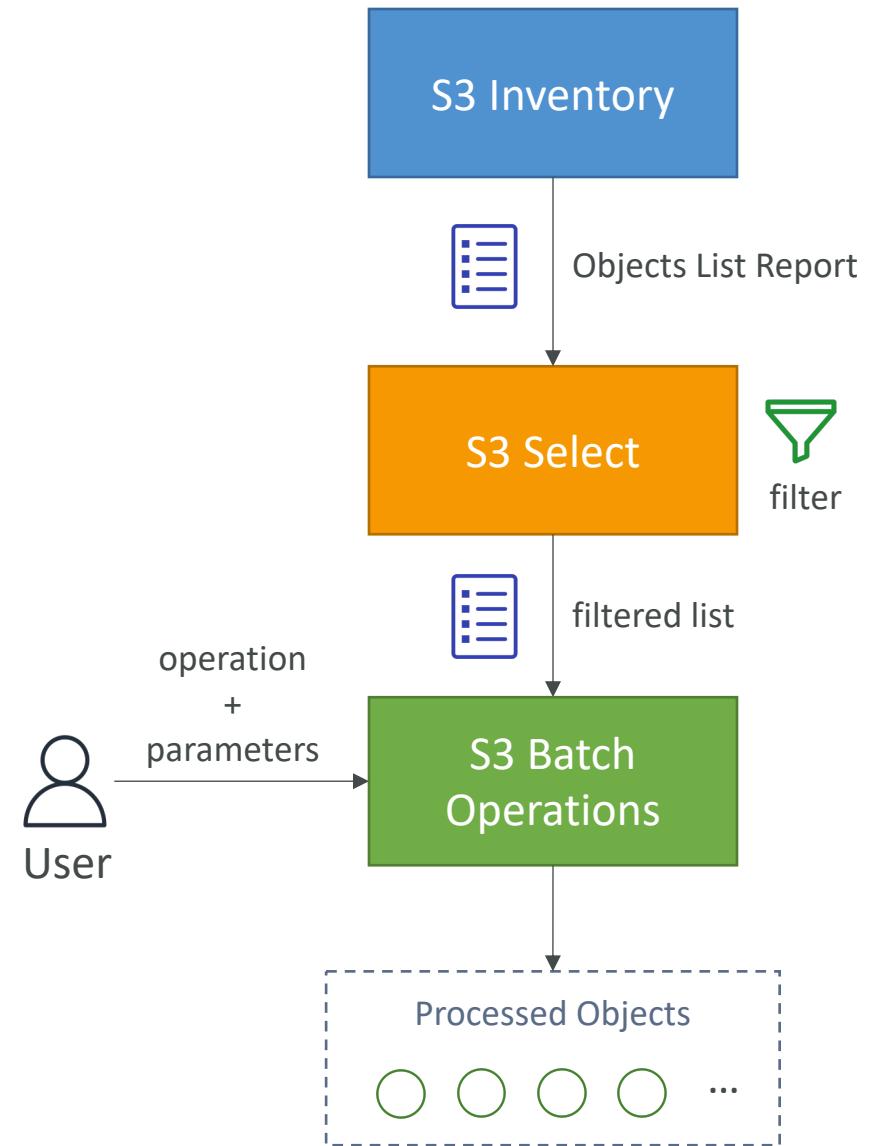


<https://aws.amazon.com/blogs/aws/s3-glacier-select/>



S3 Batch Operations

- Perform bulk operations on existing S3 objects with a single request, example:
 - Modify object metadata & properties
 - Copy objects between S3 buckets
 - **Encrypt un-encrypted objects**
 - Modify ACLs, tags
 - Restore objects from S3 Glacier
 - Invoke Lambda function to perform custom action on each object
- A job consists of a list of objects, the action to perform, and optional parameters
- S3 Batch Operations manages retries, tracks progress, sends completion notifications, generate reports ...
- You can use S3 Inventory to get object list and use S3 Select to filter your objects



Amazon S3 Security

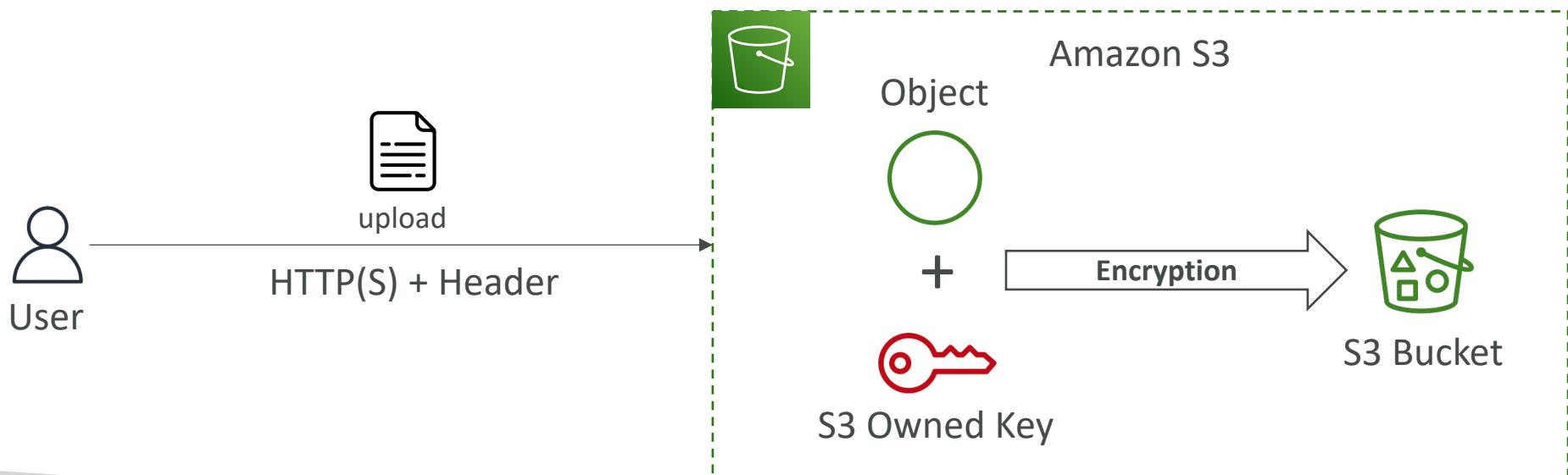


Amazon S3 – Object Encryption

- You can encrypt objects in S3 buckets using one of 4 methods
- Server-Side Encryption (SSE)
 - Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) – Enabled by Default
 - Encrypts S3 objects using keys handled, managed, and owned by AWS
 - Server-Side Encryption with KMS Keys stored in AWS KMS (SSE-KMS)
 - Leverage AWS Key Management Service (AWS KMS) to manage encryption keys
 - Server-Side Encryption with Customer-Provided Keys (SSE-C)
 - When you want to manage your own encryption keys
- Client-Side Encryption
- It's important to understand which ones are for which situation for the exam

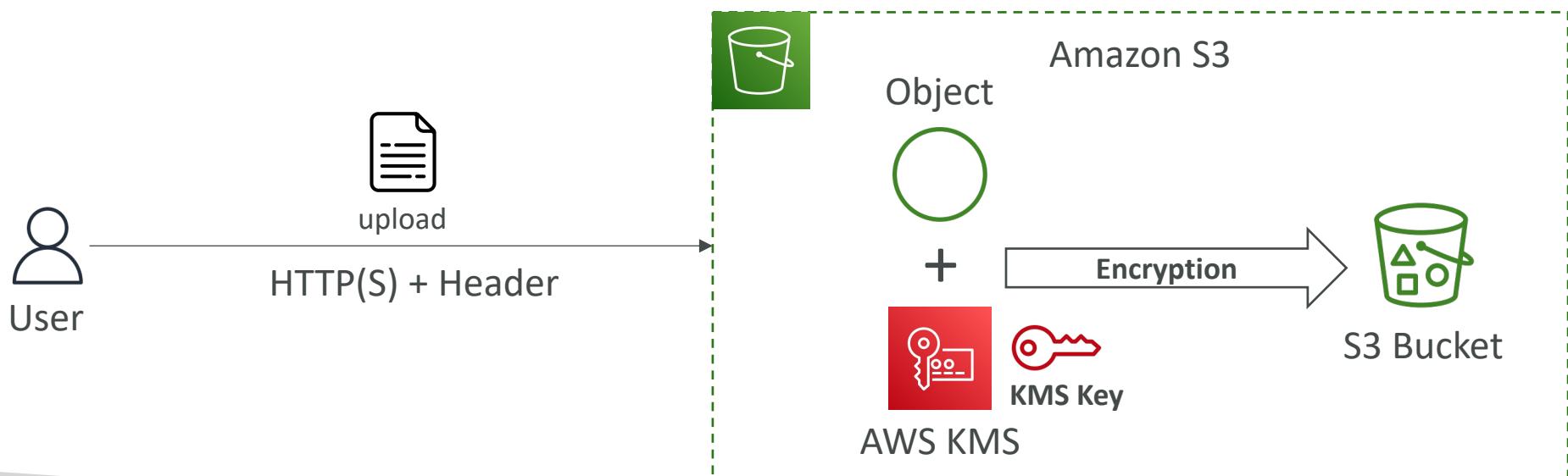
Amazon S3 Encryption – SSE-S3

- Encryption using keys handled, managed, and owned by AWS
- Object is encrypted server-side
- Encryption type is AES-256
- Must set header "x-amz-server-side-encryption": "AES256"
- Enabled by default for new buckets & new objects



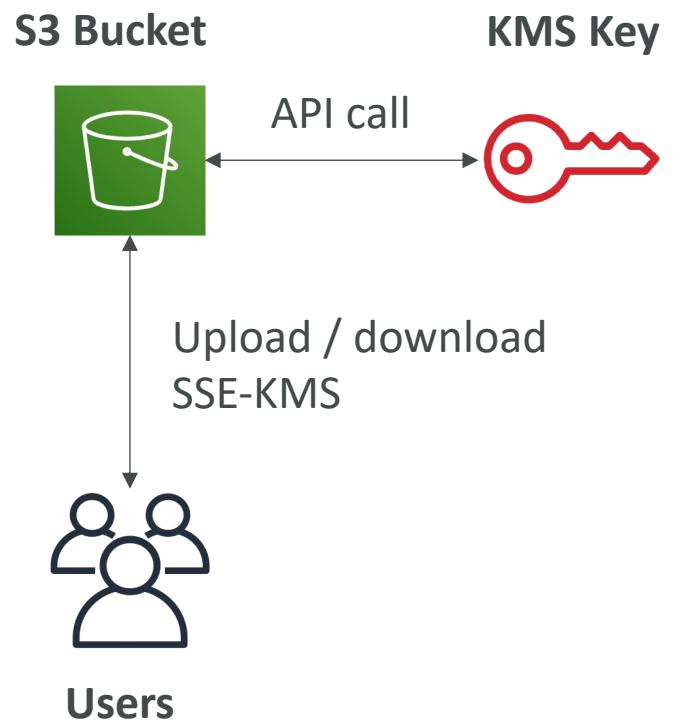
Amazon S3 Encryption – SSE-KMS

- Encryption using keys handled and managed by AWS KMS (Key Management Service)
- KMS advantages: user control + audit key usage using CloudTrail
- Object is encrypted server side
- Must set header "x-amz-server-side-encryption": "aws:kms"



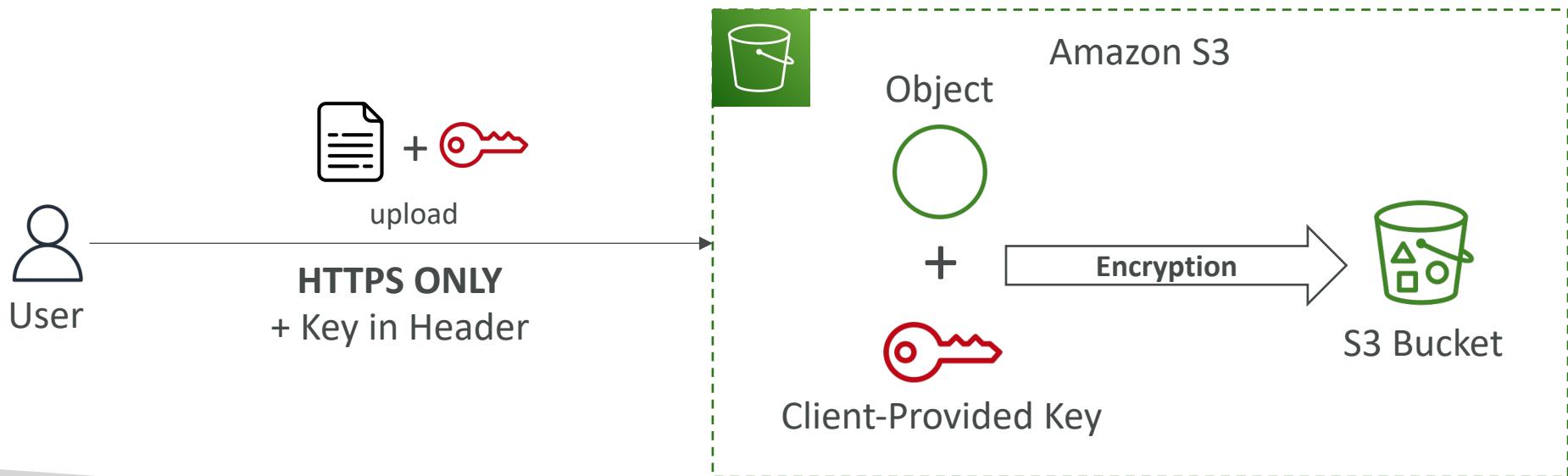
SSE-KMS Limitation

- If you use SSE-KMS, you may be impacted by the KMS limits
- When you upload, it calls the **GenerateDataKey** KMS API
- When you download, it calls the **Decrypt** KMS API
- Count towards the KMS quota per second (5500, 10000, 30000 req/s based on region)
- You can request a quota increase using the Service Quotas Console



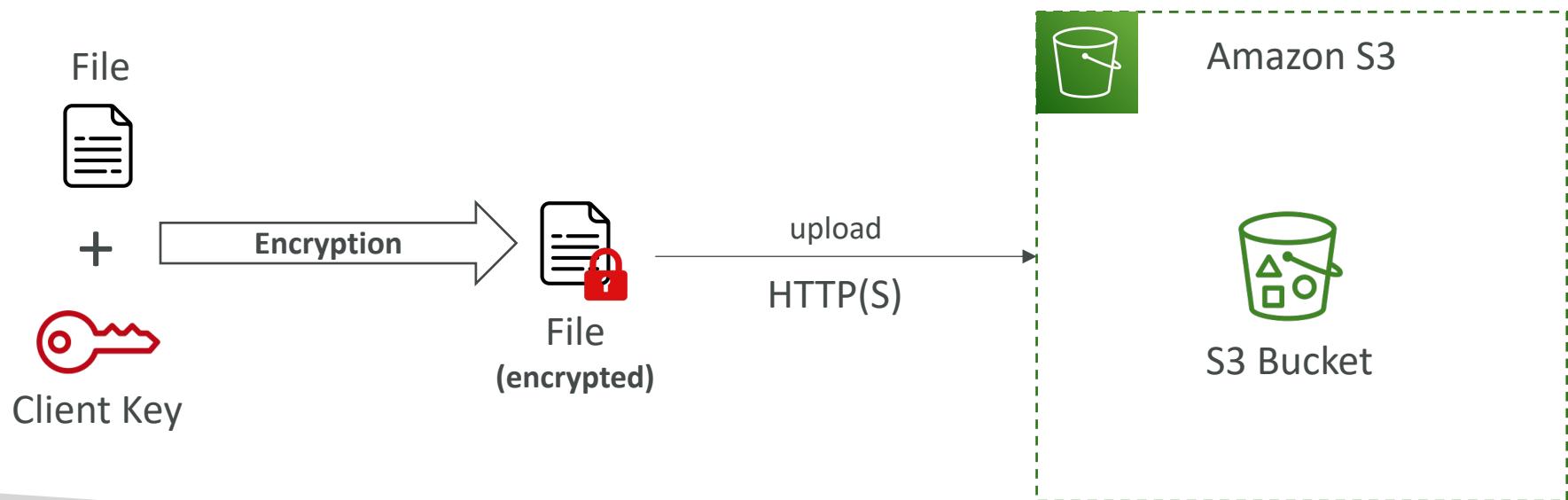
Amazon S3 Encryption – SSE-C

- Server-Side Encryption using keys fully managed by the customer outside of AWS
- Amazon S3 does **NOT** store the encryption key you provide
- **HTTPS must be used**
- Encryption key must provided in HTTP headers, for every HTTP request made



Amazon S3 Encryption – Client-Side Encryption

- Use client libraries such as [Amazon S3 Client-Side Encryption Library](#)
- Clients must encrypt data themselves before sending to Amazon S3
- Clients must decrypt data themselves when retrieving from Amazon S3
- Customer fully manages the keys and encryption cycle



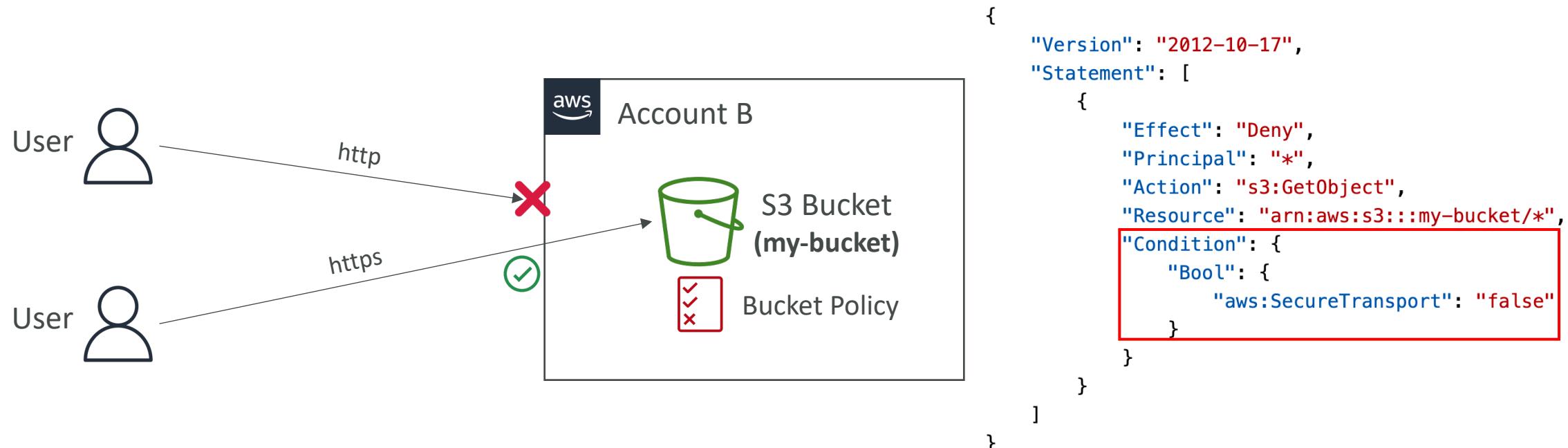
Amazon S3 – Encryption in transit (SSL/TLS)

- Encryption in flight is also called SSL/TLS
- Amazon S3 exposes two endpoints:
 - HTTP Endpoint – non encrypted
 - HTTPS Endpoint – encryption in flight
- HTTPS is recommended
- HTTPS is mandatory for SSE-C
- Most clients would use the HTTPS endpoint by default



Amazon S3 – Force Encryption in Transit

aws:SecureTransport



Amazon S3 – Default Encryption vs. Bucket Policies

- SSE-S3 encryption is automatically applied to new objects stored in S3 bucket
- Optionally, you can “force encryption” using a bucket policy and refuse any API call to PUT an S3 object without encryption headers (SSE-KMS or SSE-C)

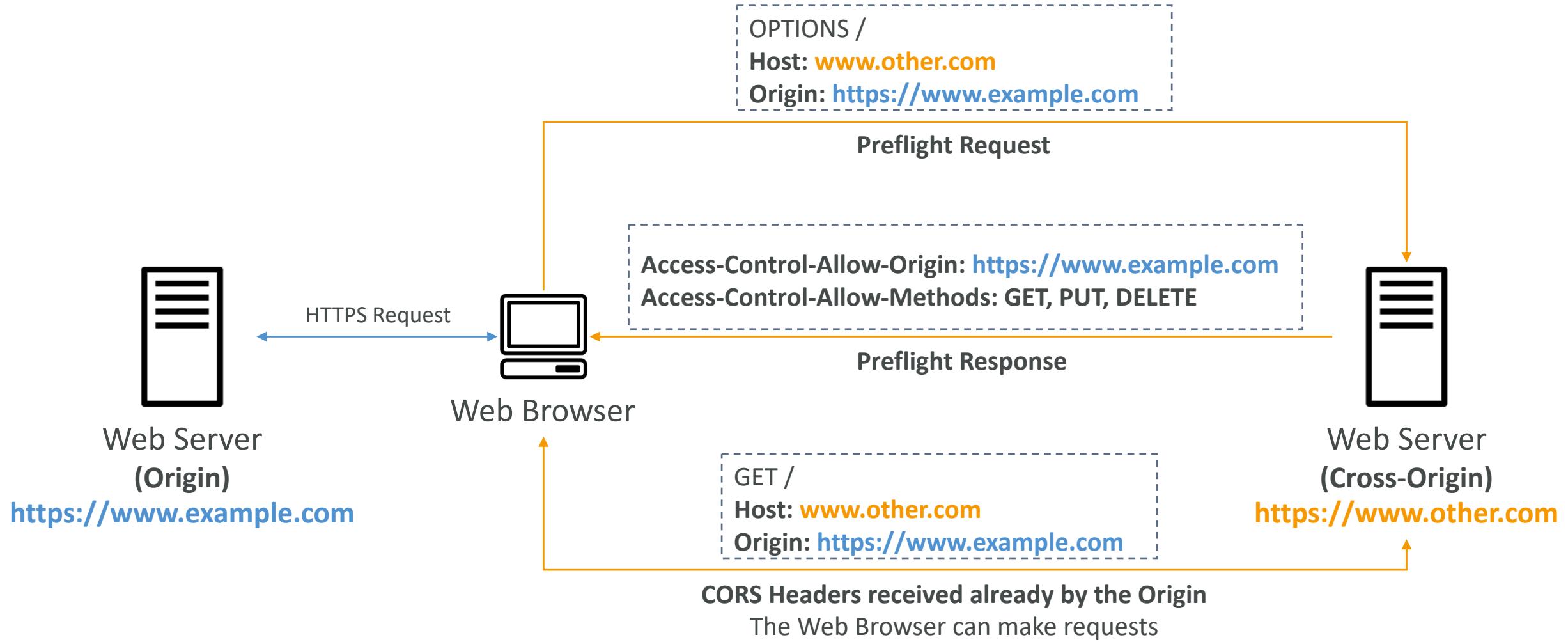
```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "s3:PutObject",  
            "Principal": "*",  
            "Resource": "arn:aws:s3:::my-bucket/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-server-side-encryption": "aws:kms"  
                }  
            }  
        }  
    ]  
}  
  
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "s3:PutObject",  
            "Principal": "*",  
            "Resource": "arn:aws:s3:::my-bucket/*",  
            "Condition": {  
                "Null": {  
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"  
                }  
            }  
        }  
    ]  
}
```

- Note: Bucket Policies are evaluated before “Default Encryption”

What is CORS?

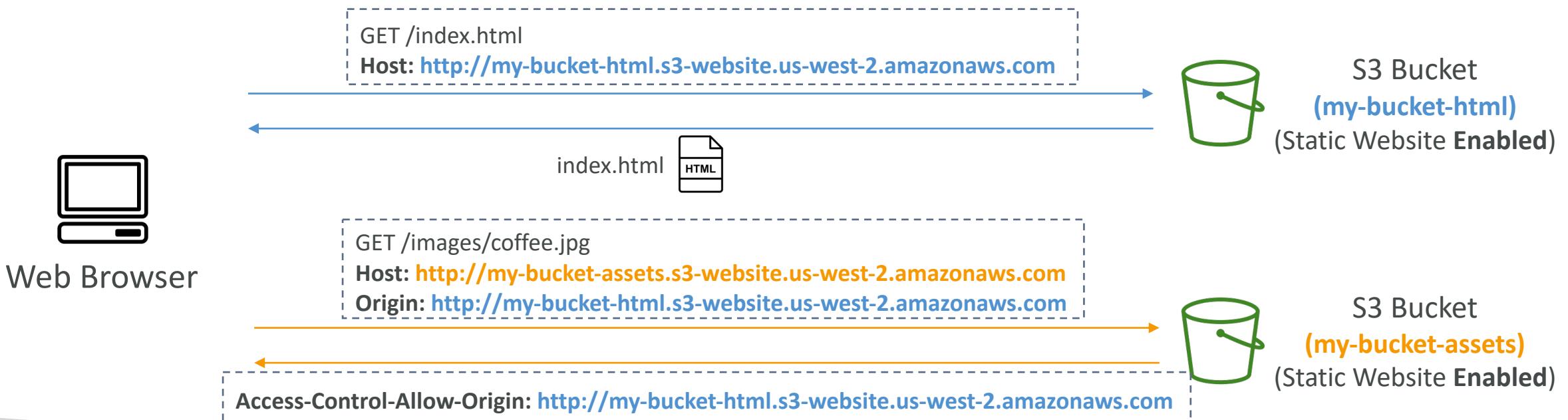
- Cross-Origin Resource Sharing (CORS)
- Origin = scheme (protocol) + host (domain) + port
 - example: <https://www.example.com> (implied port is 443 for HTTPS, 80 for HTTP)
- Web Browser based mechanism to allow requests to other origins while visiting the main origin
- Same origin: <http://example.com/app1> & <http://example.com/app2>
- Different origins: <http://www.example.com> & <http://other.example.com>
- The requests won't be fulfilled unless the other origin allows for the requests, using CORS Headers (example: Access-Control-Allow-Origin)

What is CORS?



Amazon S3 – CORS

- If a client makes a cross-origin request on our S3 bucket, we need to enable the correct CORS headers
- It's a popular exam question
- You can allow for a specific origin or for * (all origins)



Amazon S3 – MFA Delete

- **MFA (Multi-Factor Authentication)** – force users to generate a code on a device (usually a mobile phone or hardware) before doing important operations on S3
- MFA will be required to:
 - Permanently delete an object version
 - Suspend Versioning on the bucket
- MFA won't be required to:
 - Enable Versioning
 - List deleted versions
- To use MFA Delete, Versioning must be enabled on the bucket
- Only the bucket owner (root account) can enable/disable MFA Delete



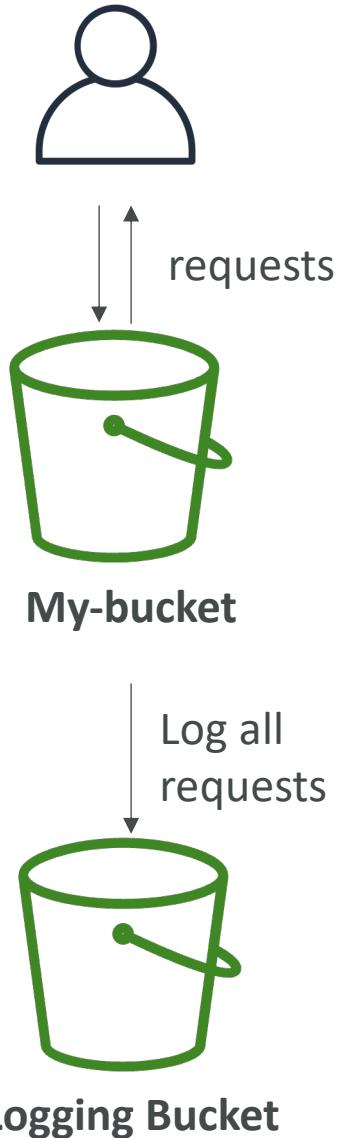
Google Authenticator



MFA Hardware Device

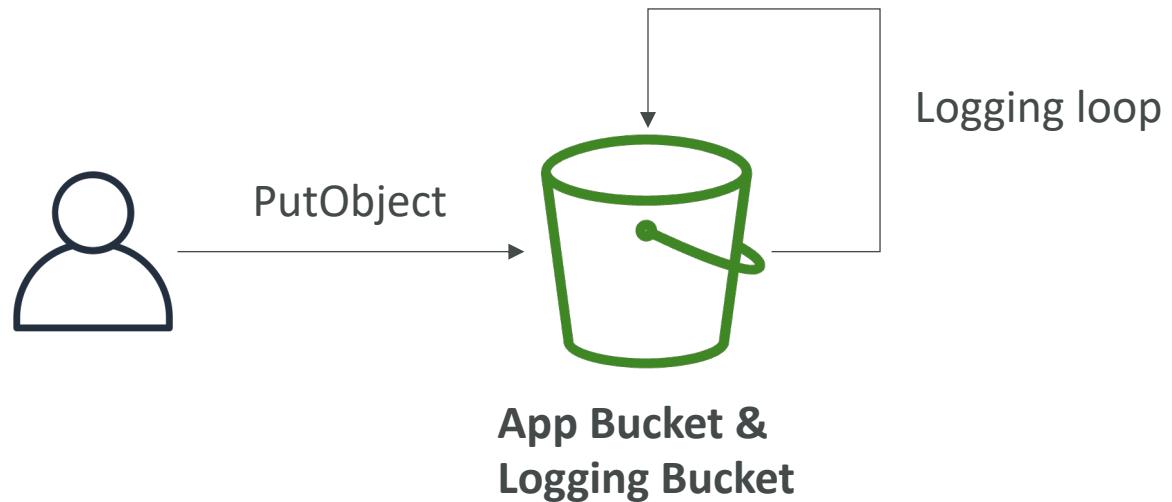
S3 Access Logs

- For audit purpose, you may want to log all access to S3 buckets
 - Any request made to S3, from any account, authorized or denied, will be logged into another S3 bucket
 - That data can be analyzed using data analysis tools...
 - The target logging bucket must be in the same AWS region
-
- The log format is at:
<https://docs.aws.amazon.com/AmazonS3/latest/dev/LogFormat.html>



S3 Access Logs: Warning

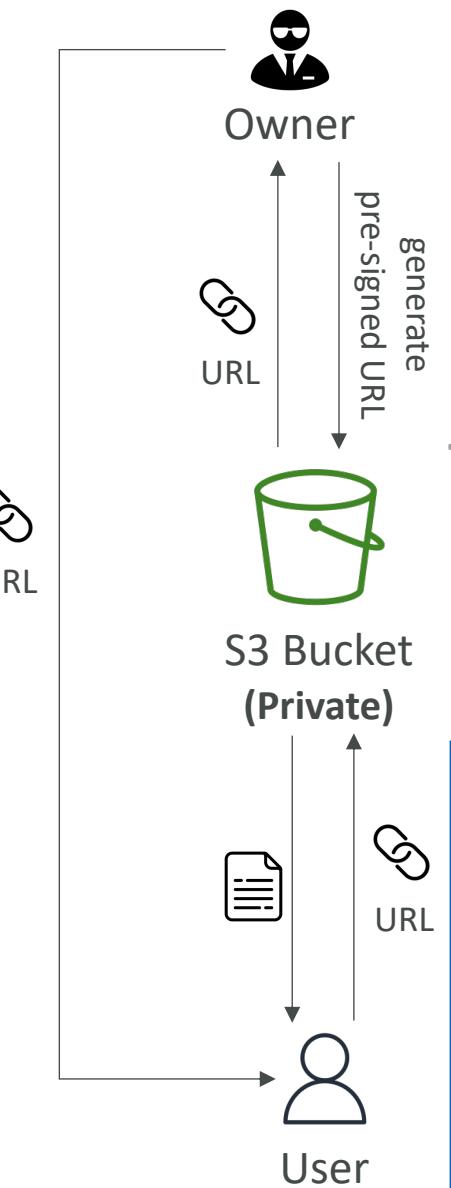
- Do not set your logging bucket to be the monitored bucket
- It will create a logging loop, and **your bucket will grow exponentially**



Do not try this at home 😊

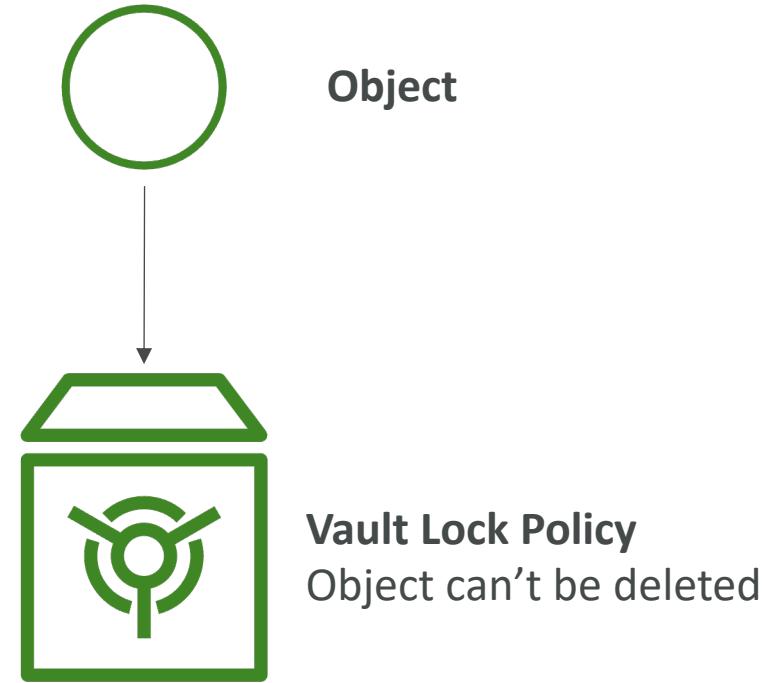
Amazon S3 – Pre-Signed URLs

- Generate pre-signed URLs using the **S3 Console, AWS CLI or SDK**
- **URL Expiration**
 - S3 Console – 1 min up to 720 mins (12 hours)
 - AWS CLI – configure expiration with `--expires-in` parameter in seconds (default 3600 secs, max. 604800 secs ~ 168 hours)
- Users given a pre-signed URL inherit the permissions of the user that generated the URL for GET / PUT
- Examples:
 - Allow only logged-in users to download a premium video from your S3 bucket
 - Allow an ever-changing list of users to download files by generating URLs dynamically
 - Allow temporarily a user to upload a file to a precise location in your S3 bucket



S3 Glacier Vault Lock

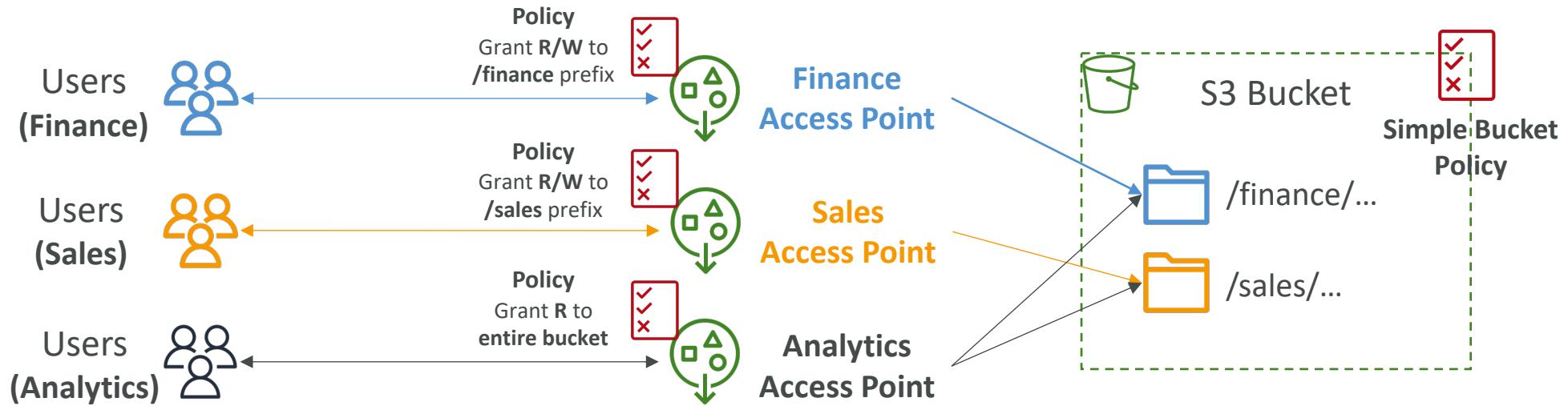
- Adopt a WORM (Write Once Read Many) model
- Create a Vault Lock Policy
- Lock the policy for future edits (can no longer be changed or deleted)
- Helpful for compliance and data retention



S3 Object Lock (versioning must be enabled)

- Adopt a WORM (Write Once Read Many) model
- Block an object version deletion for a specified amount of time
- **Retention mode - Compliance:**
 - Object versions can't be overwritten or deleted by any user, including the root user
 - Objects retention modes can't be changed, and retention periods can't be shortened
- **Retention mode - Governance:**
 - Most users can't overwrite or delete an object version or alter its lock settings
 - Some users have special permissions to change the retention or delete the object
- **Retention Period:** protect the object for a fixed period, it can be extended
- **Legal Hold:**
 - protect the object indefinitely, independent from retention period
 - can be freely placed and removed using the `s3:PutObjectLegalHold` IAM permission

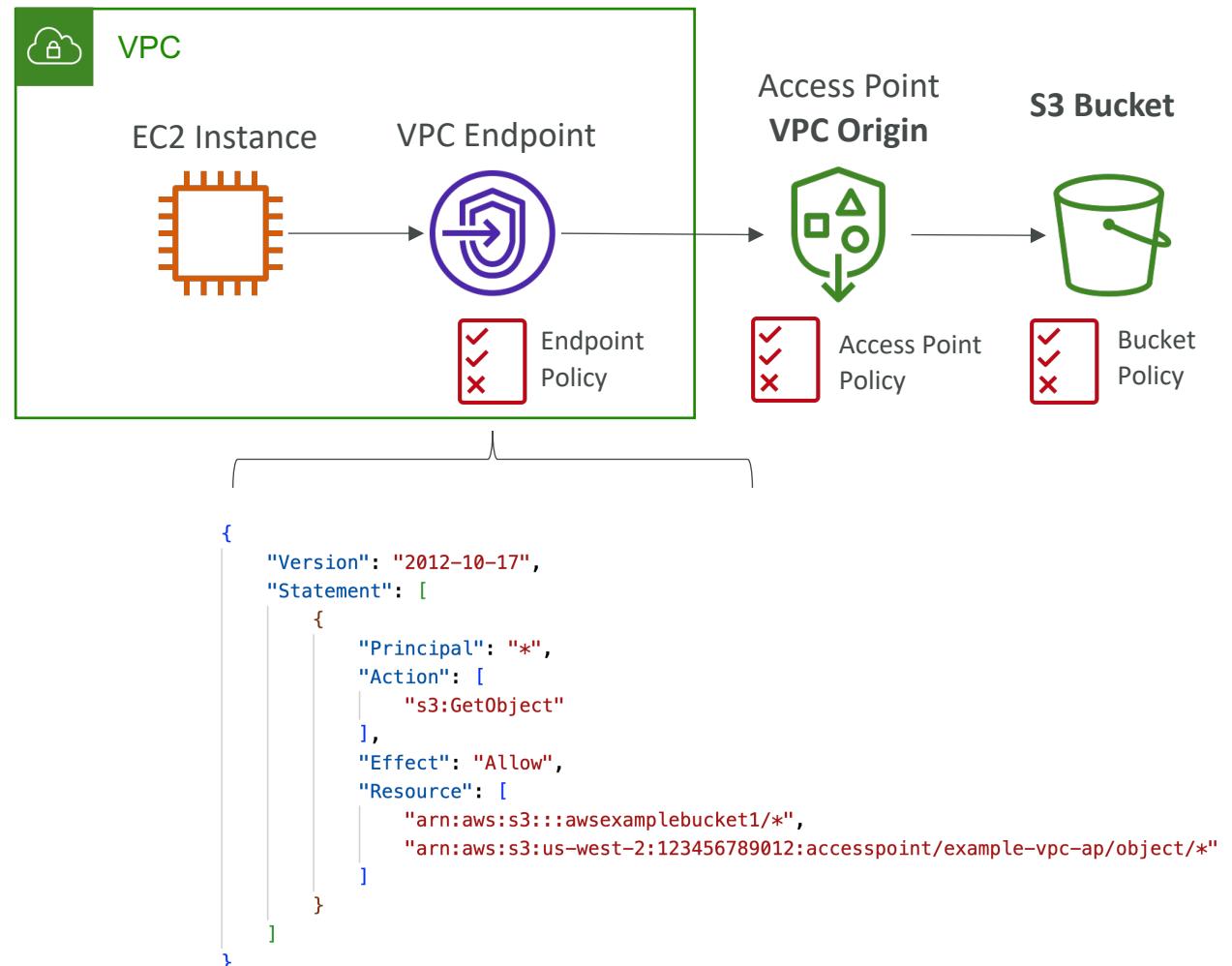
S3 – Access Points



- Access Points simplify security management for S3 Buckets
- Each Access Point has:
 - its own DNS name (Internet Origin or VPC Origin)
 - an access point policy (similar to bucket policy) – manage security at scale

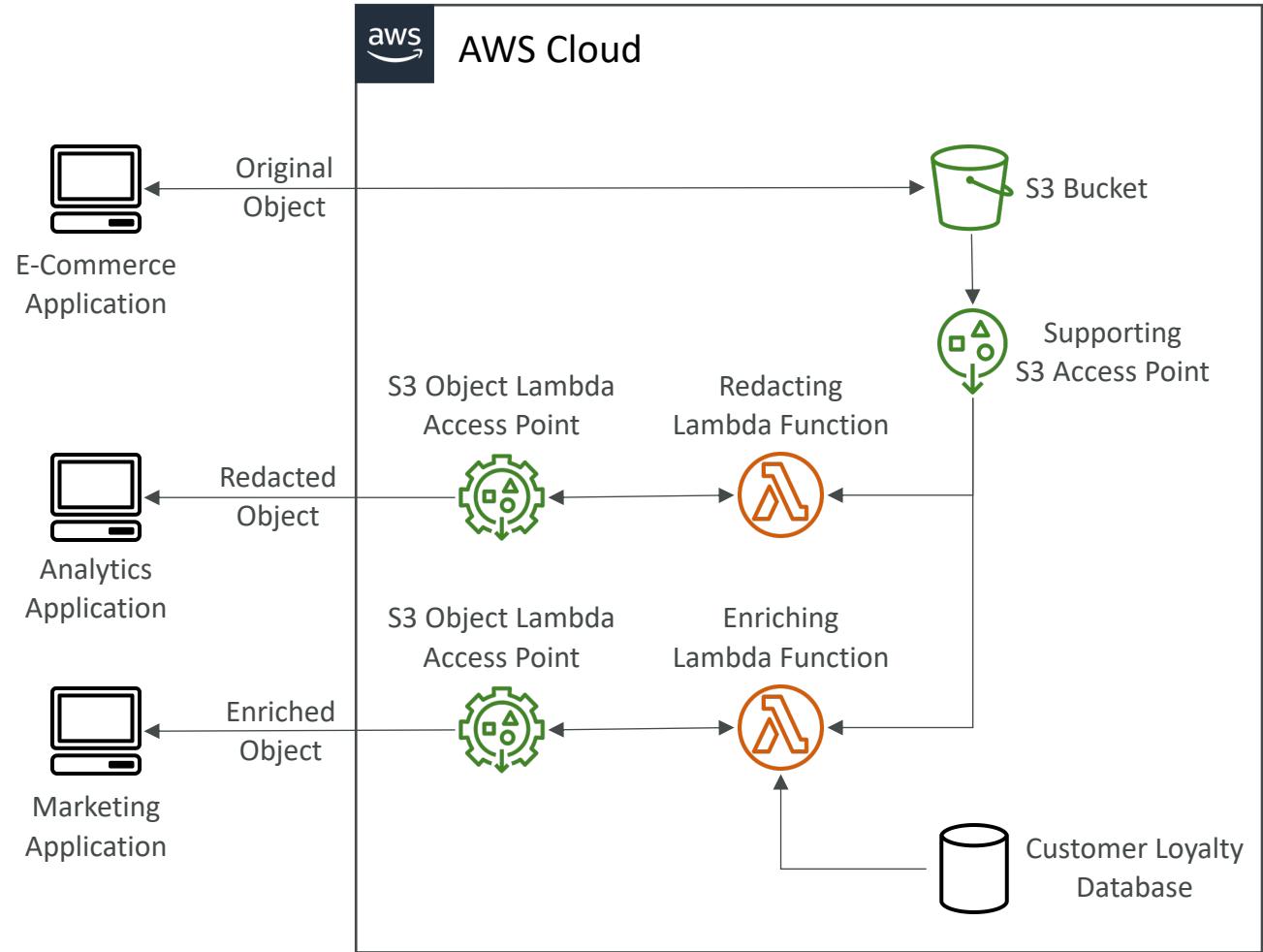
S3 – Access Points – VPC Origin

- We can define the access point to be accessible only from within the VPC
- You must create a VPC Endpoint to access the Access Point (Gateway or Interface Endpoint)
- The VPC Endpoint Policy must allow access to the target bucket and Access Point



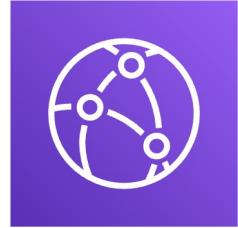
S3 Object Lambda

- Use AWS Lambda Functions to change the object before it is retrieved by the caller application
- Only one S3 bucket is needed, on top of which we create **S3 Access Point** and **S3 Object Lambda Access Points**.
- Use Cases:
 - Redacting personally identifiable information for analytics or non-production environments.
 - Converting across data formats, such as converting XML to JSON.
 - Resizing and watermarking images on the fly using caller-specific details, such as the user who requested the object.

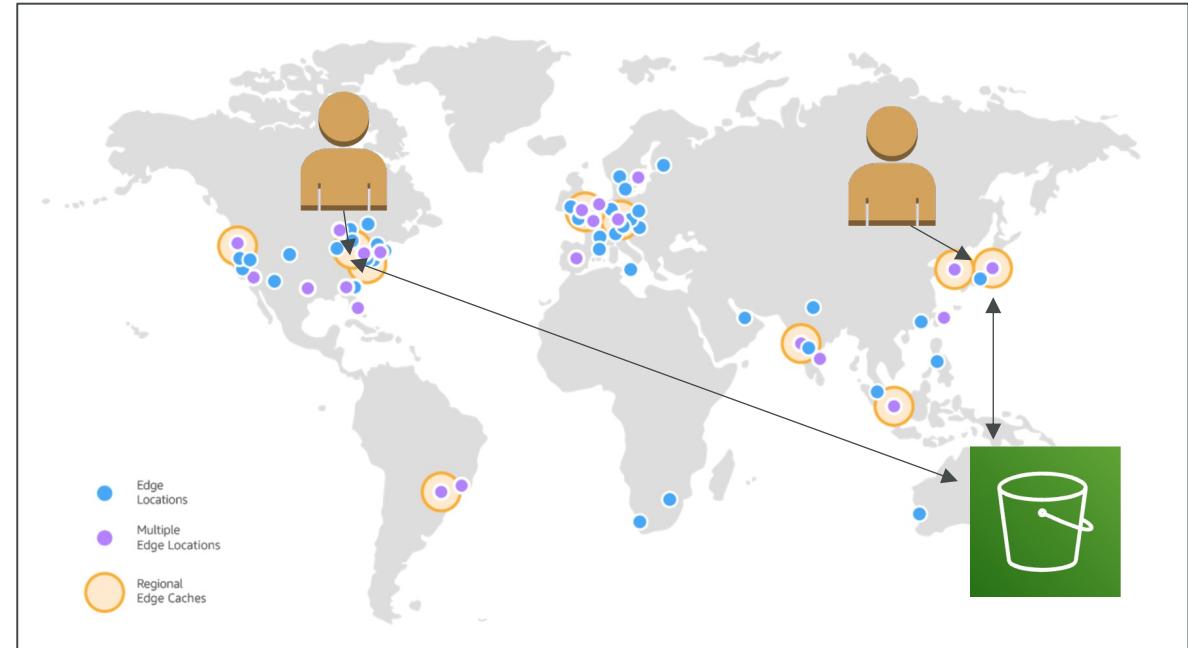


Global Infrastructure

Amazon CloudFront



- Content Delivery Network (CDN)
- Improves read performance, content is cached at the edge
- Improves users experience
- 216 Point of Presence globally (edge locations)
- DDoS protection (because worldwide), integration with Shield, AWS Web Application Firewall

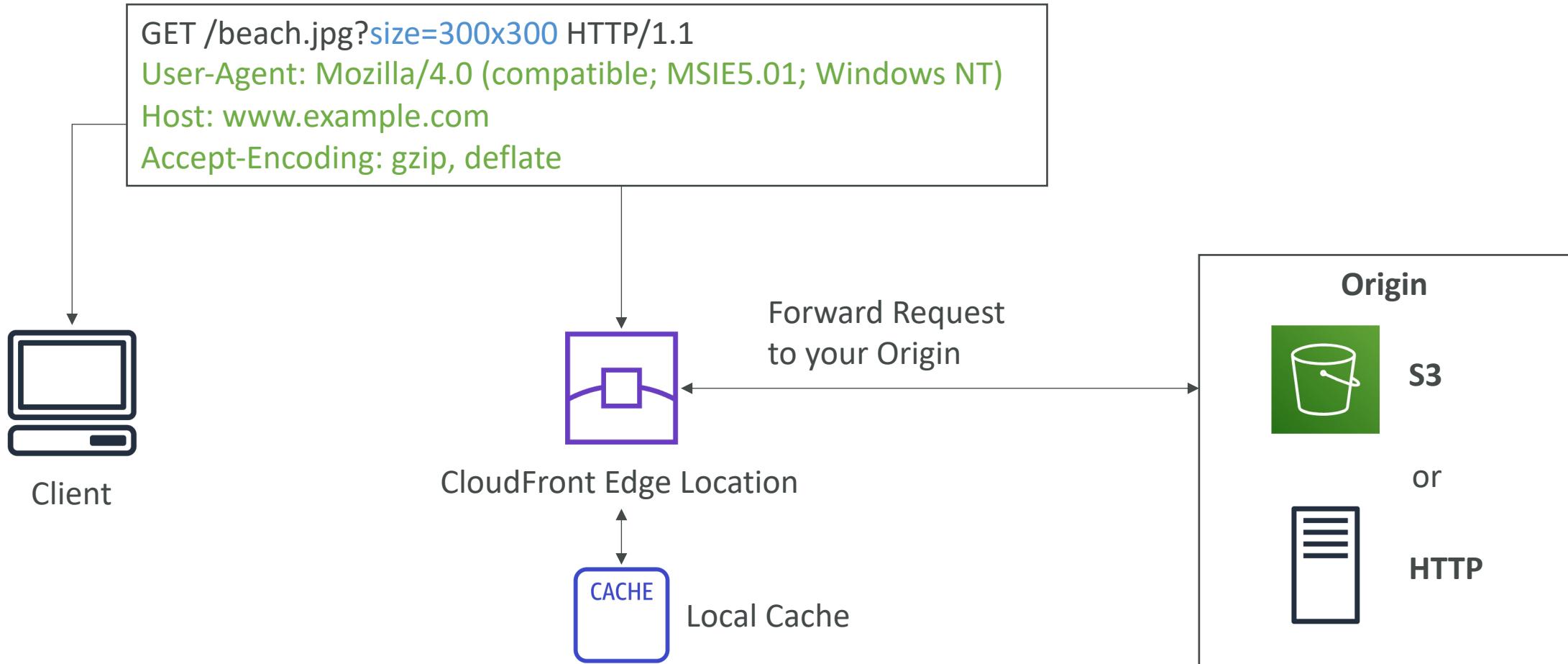


Source: <https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2>

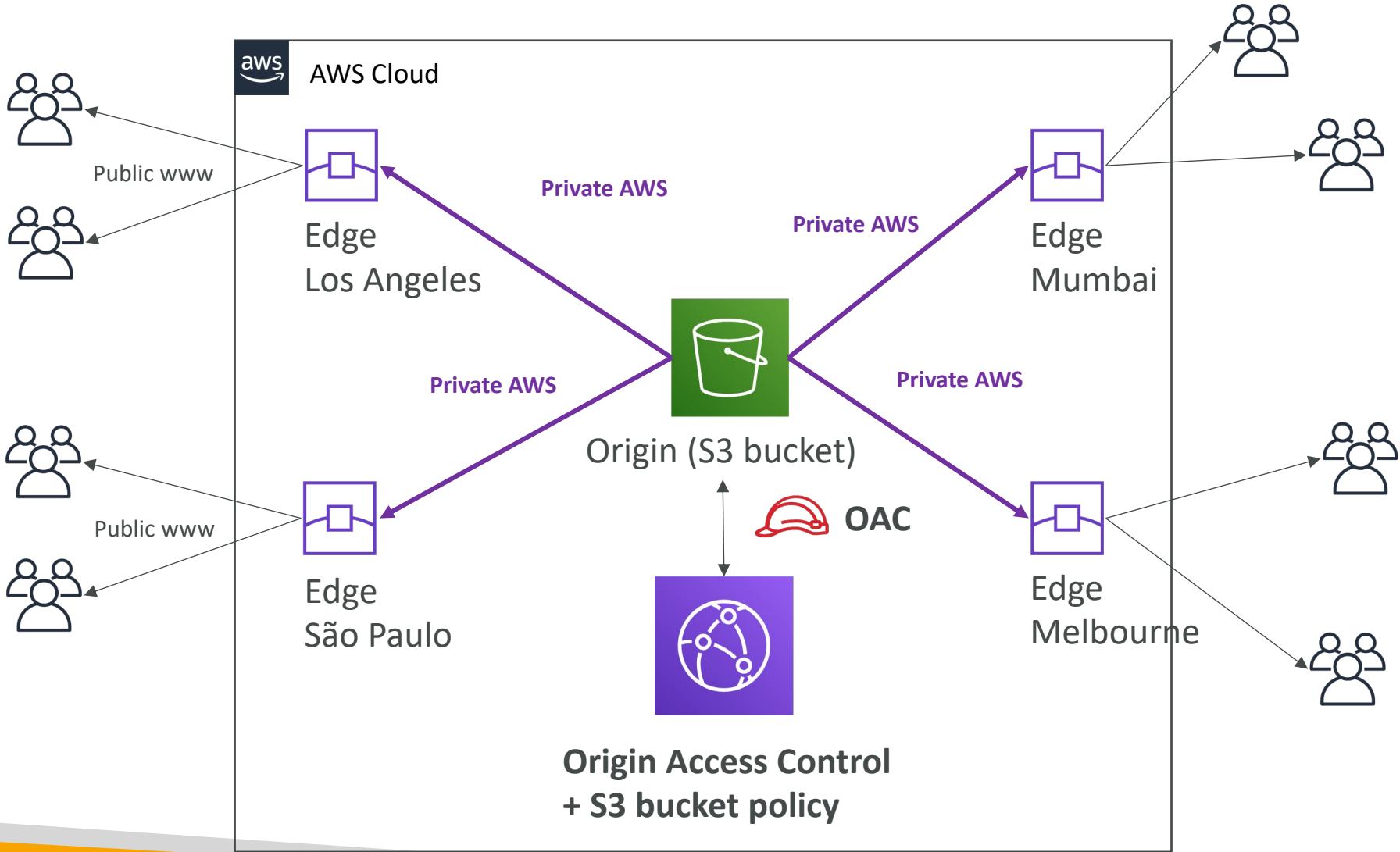
CloudFront – Origins

- **S3 bucket**
 - For distributing files and caching them at the edge
 - Enhanced security with CloudFront Origin Access Control (OAC)
 - OAC is replacing Origin Access Identity (OAI)
 - CloudFront can be used as an ingress (to upload files to S3)
- **Custom Origin (HTTP)**
 - Application Load Balancer
 - EC2 instance
 - S3 website (must first enable the bucket as a static S3 website)
 - Any HTTP backend you want

CloudFront at a high level



CloudFront – S3 as an Origin



CloudFront vs S3 Cross Region Replication

- CloudFront:
 - Global Edge network
 - Files are cached for a TTL (maybe a day)
 - Great for static content that must be available everywhere
- S3 Cross Region Replication:
 - Must be setup for each region you want replication to happen
 - Files are updated in near real-time
 - Read only
 - Great for dynamic content that needs to be available at low-latency in few regions

CloudFront – ALB or EC2 as an origin



CloudFront Geo Restriction

- You can restrict who can access your distribution
 - **Allowlist:** Allow your users to access your content only if they're in one of the countries on a list of approved countries.
 - **Blocklist:** Prevent your users from accessing your content if they're in one of the countries on a list of banned countries.
- The “country” is determined using a 3rd party Geo-IP database
- Use case: Copyright Laws to control access to content

CloudFront - Pricing

- CloudFront Edge locations are all around the world
- The cost of data out per edge location varies

Per Month	United States, Mexico, & Canada	Europe & Israel	South Africa, Kenya, & Middle East	South America	Japan	Australia & New Zealand	Hong Kong, Philippines, Singapore, South Korea, Taiwan, & Thailand	India
First 10TB	\$0.085	\$0.085	\$0.110	\$0.110	\$0.114	\$0.114	\$0.140	\$0.170
Next 40TB	\$0.080	\$0.080	\$0.105	\$0.105	\$0.089	\$0.098	\$0.135	\$0.130
Next 100TB	\$0.060	\$0.060	\$0.090	\$0.090	\$0.086	\$0.094	\$0.120	\$0.110
Next 350TB	\$0.040	\$0.040	\$0.080	\$0.080	\$0.084	\$0.092	\$0.100	\$0.100
Next 524TB	\$0.030	\$0.030	\$0.060	\$0.060	\$0.080	\$0.090	\$0.080	\$0.100
Next 4PB	\$0.025	\$0.025	\$0.050	\$0.050	\$0.070	\$0.085	\$0.070	\$0.100
Over 5PB	\$0.020	\$0.020	\$0.040	\$0.040	\$0.060	\$0.080	\$0.060	\$0.100

lower higher

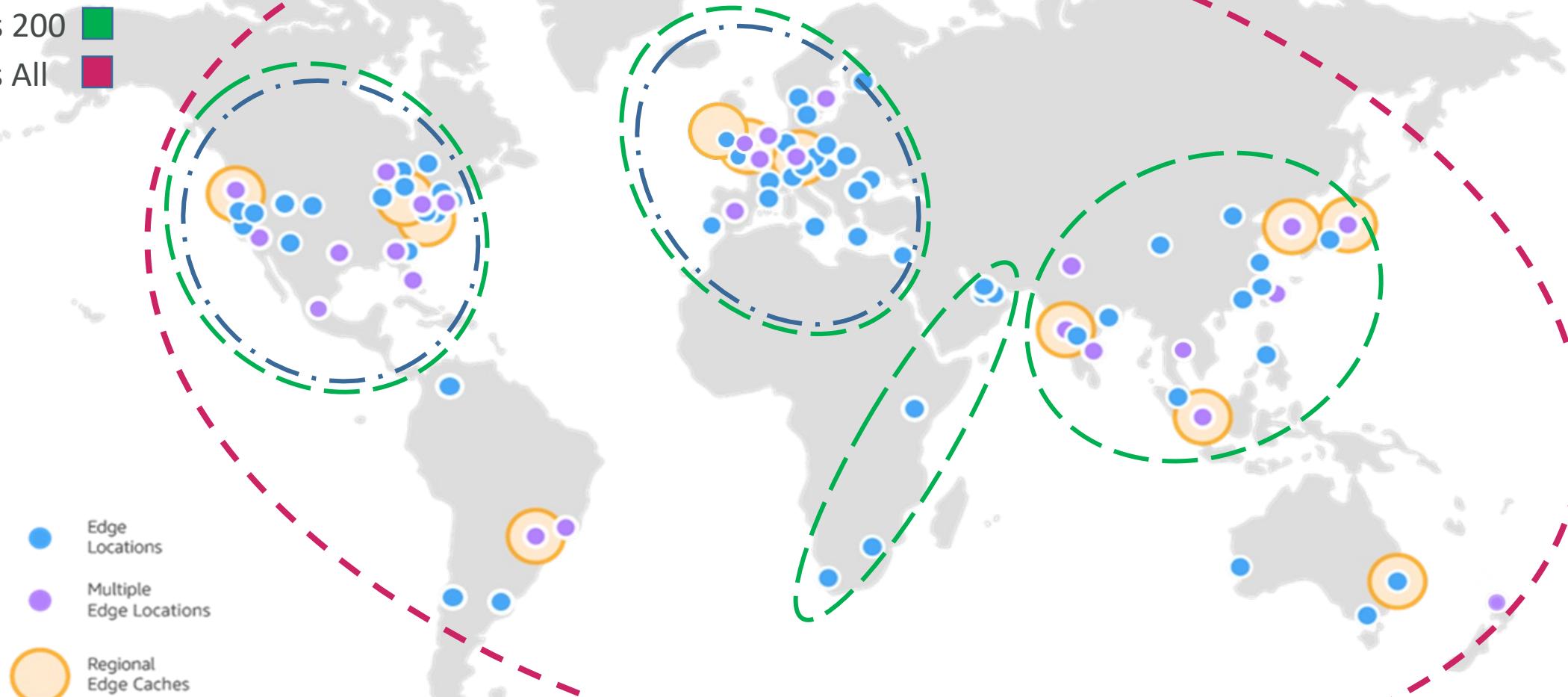

CloudFront – Price Classes

- You can reduce the number of edge locations for cost reduction
- Three price classes:
 1. Price Class All: all regions – best performance
 2. Price Class 200: most regions, but excludes the most expensive regions
 3. Price Class 100: only the least expensive regions

Edge Locations Included Within	United States, Mexico, & Canada	Europe & Israel	South Africa, Kenya, & Middle East	South America	Japan	Australia & New Zealand	Hong Kong, Philippines, Singapore, South Korea, Taiwan, & Thailand	India
Price Class All	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Price Class 200	Yes	Yes	Yes	x	Yes	x	Yes	Yes
Price Class 100	Yes	Yes	x	x	x	x	x	x

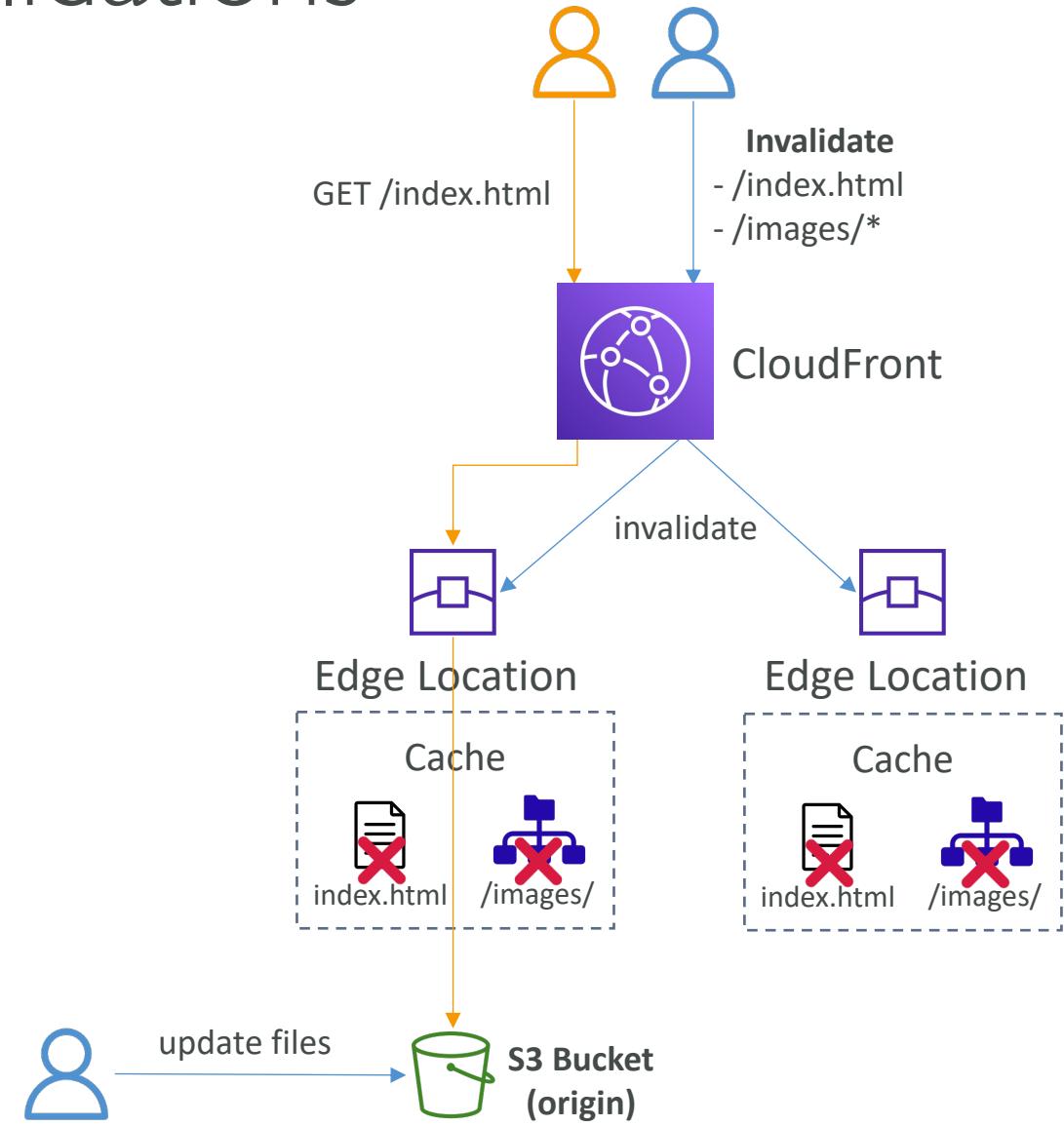
CloudFront - Price Class

Prices Class 100 ■
Prices Class 200 ■
Prices Class All ■



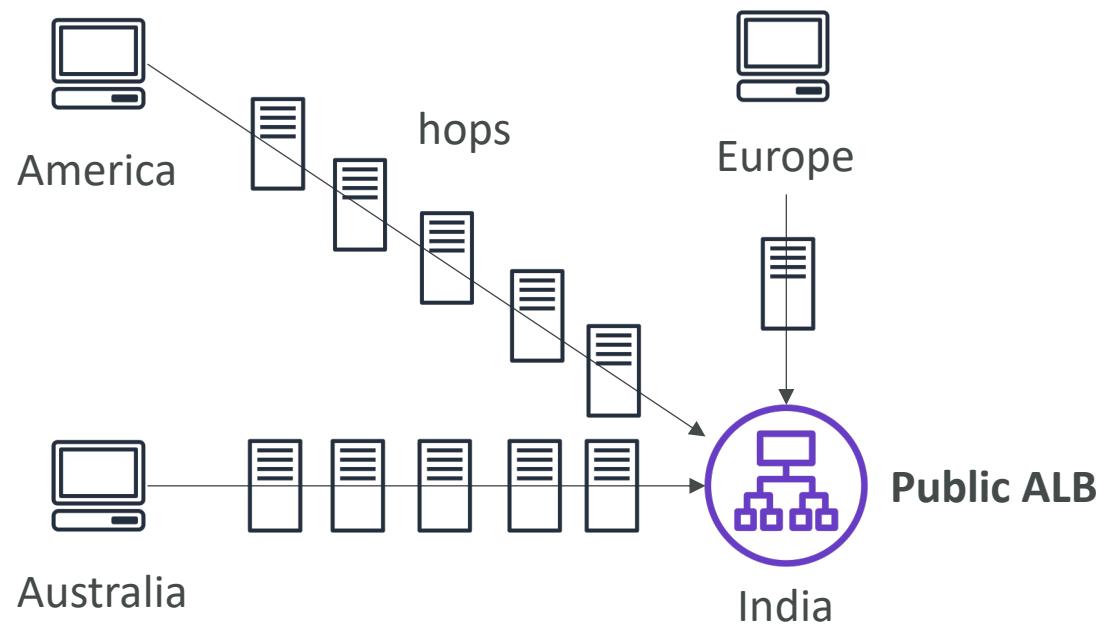
CloudFront – Cache Invalidations

- In case you update the back-end origin, CloudFront doesn't know about it and will only get the refreshed content after the TTL has expired
- However, you can force an entire or partial cache refresh (thus bypassing the TTL) by performing a **CloudFront Invalidation**
- You can invalidate all files (*) or a special path (/images/*)



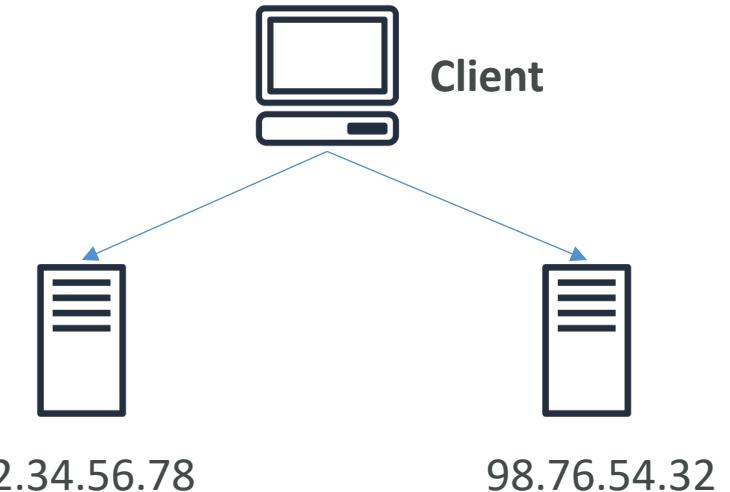
Global users for our application

- You have deployed an application and have global users who want to access it directly.
- They go over the public internet, which can add a lot of latency due to many hops
- We wish to go as fast as possible through AWS network to minimize latency

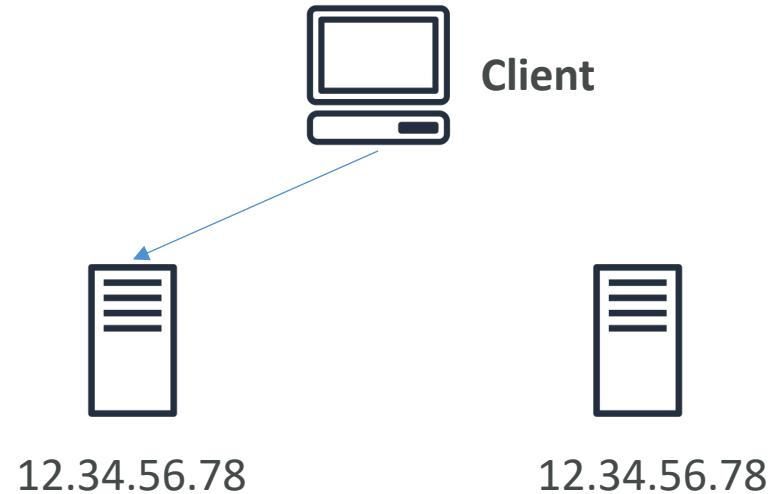


Unicast IP vs Anycast IP

- **Unicast IP:** one server holds one IP address



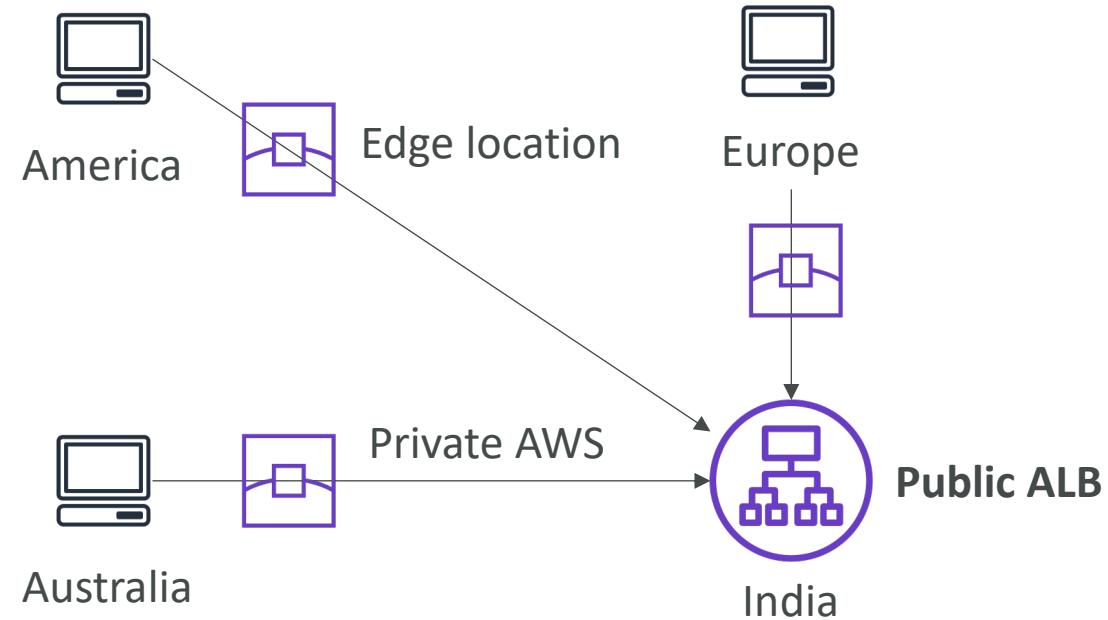
- **Anycast IP:** all servers hold the same IP address and the client is routed to the nearest one



AWS Global Accelerator



- Leverage the AWS internal network to route to your application
- 2 Anycast IP are created for your application
- The Anycast IP send traffic directly to Edge Locations
- The Edge locations send the traffic to your application



AWS Global Accelerator

- Works with Elastic IP, EC2 instances, ALB, NLB, public or private
- Consistent Performance
 - Intelligent routing to lowest latency and fast regional failover
 - No issue with client cache (because the IP doesn't change)
 - Internal AWS network
- Health Checks
 - Global Accelerator performs a health check of your applications
 - Helps make your application global (failover less than 1 minute for unhealthy)
 - Great for disaster recovery (thanks to the health checks)
- Security
 - only 2 external IP need to be whitelisted
 - DDoS protection thanks to AWS Shield

AWS Global Accelerator vs CloudFront

- They both use the AWS global network and its edge locations around the world
- Both services integrate with AWS Shield for DDoS protection.
- **CloudFront**
 - Improves performance for both cacheable content (such as images and videos)
 - Dynamic content (such as API acceleration and dynamic site delivery)
 - Content is served at the edge
- **Global Accelerator**
 - Improves performance for a wide range of applications over TCP or UDP
 - Proxying packets at the edge to applications running in one or more AWS Regions.
 - Good fit for non-HTTP use cases, such as gaming (UDP), IoT (MQTT), or Voice over IP
 - Good for HTTP use cases that require static IP addresses
 - Good for HTTP use cases that required deterministic, fast regional failover

Advanced Storage on AWS

AWS Snow Family

- Highly-secure, portable devices to collect and process data at the edge, and migrate data into and out of AWS

- Data migration:



Snowcone



Snowball Edge



Snowmobile

- Edge computing:



Snowcone



Snowball Edge

Data Migrations with AWS Snow Family

	Time to Transfer		
	100 Mbps	1Gbps	10Gbps
10 TB	12 days	30 hours	3 hours
100 TB	124 days	12 days	30 hours
1 PB	3 years	124 days	12 days

Challenges:

- Limited connectivity
- Limited bandwidth
- High network cost
- Shared bandwidth (can't maximize the line)
- Connection stability

AWS Snow Family: offline devices to perform data migrations

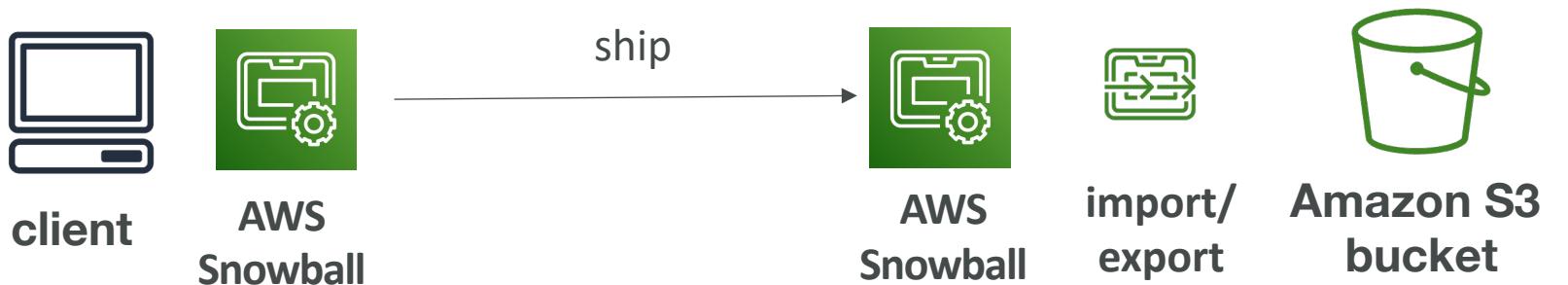
If it takes more than a week to transfer over the network, use Snowball devices!

Diagrams

- Direct upload to S3:



- With Snow Family:



Snowball Edge (for data transfers)



- Physical data transport solution: move TBs or PBs of data in or out of AWS
- Alternative to moving data over the network (and paying network fees)
- Pay per data transfer job
- Provide block storage and Amazon S3-compatible object storage
- **Snowball Edge Storage Optimized**
 - 80 TB of HDD capacity for block volume and S3 compatible object storage
- **Snowball Edge Compute Optimized**
 - 42 TB of HDD or 28TB NVMe capacity for block volume and S3 compatible object storage
- Use cases: large data cloud migrations, DC decommission, disaster recovery



AWS Snowcone & Snowcone SSD



- Small, portable computing, anywhere, rugged & secure, withstands harsh environments
- Light (4.5 pounds, 2.1 kg)
- Device used for edge computing, storage, and data transfer
- Snowcone – 8 TB of HDD Storage
- Snowcone SSD – 14 TB of SSD Storage
- Use Snowcone where Snowball does not fit (space-constrained environment)
- Must provide your own battery / cables
- Can be sent back to AWS offline, or connect it to internet and use **AWS DataSync** to send data



AWS Snowmobile



- Transfer exabytes of data (1 EB = 1,000 PB = 1,000,000 TBs)
- Each Snowmobile has 100 PB of capacity (use multiple in parallel)
- High security: temperature controlled, GPS, 24/7 video surveillance
- Better than Snowball if you transfer more than 10 PB

AWS Snow Family for Data Migrations



Snowcone



Snowball Edge



Snowmobile

	Snowcone & Snowcone SSD	Snowball Edge Storage Optimized	Snowmobile
Storage Capacity	8 TB HDD 14 TB SSD	80 TB usable	< 100 PB
Migration Size	Up to 24 TB, online and offline	Up to petabytes, offline	Up to exabytes, offline
DataSync agent	Pre-installed		
Storage Clustering		Up to 15 nodes	

Snow Family – Usage Process

1. Request Snowball devices from the AWS console for delivery
2. Install the snowball client / AWS OpsHub on your servers
3. Connect the snowball to your servers and copy files using the client
4. Ship back the device when you're done (goes to the right AWS facility)
5. Data will be loaded into an S3 bucket
6. Snowball is completely wiped

What is Edge Computing?

- Process data while it's being created on **an edge location**
 - A truck on the road, a ship on the sea, a mining station underground...



- These locations may have
 - Limited / no internet access
 - Limited / no easy access to computing power
- We setup a **Snowball Edge / Snowcone** device to do edge computing
- Use cases of Edge Computing:
 - Preprocess data
 - Machine learning at the edge
 - Transcoding media streams
- Eventually (if need be) we can ship back the device to AWS (for transferring data for example)

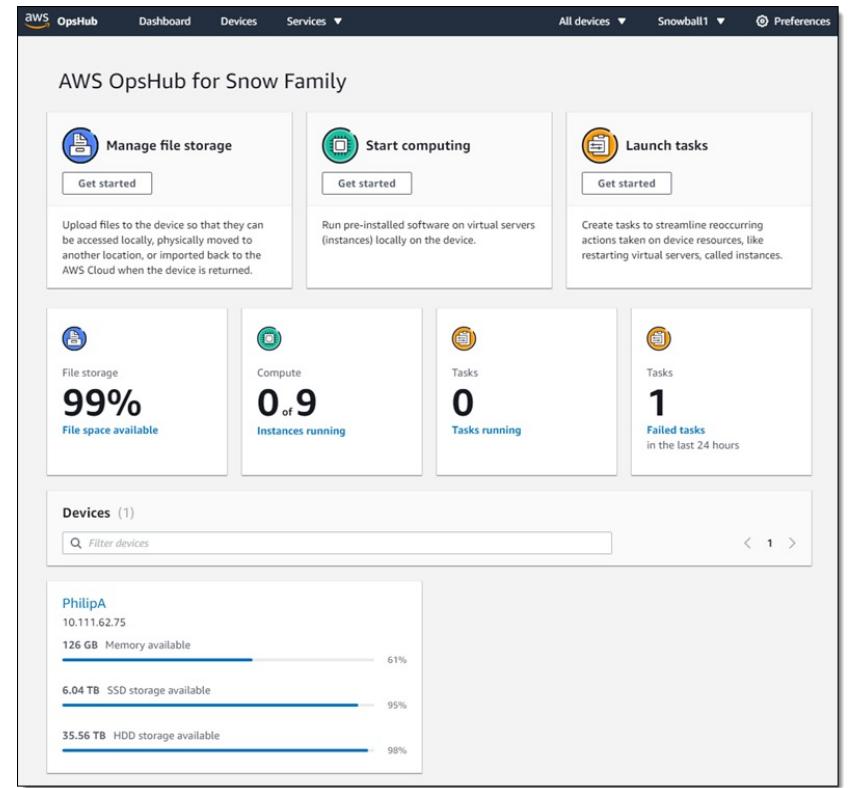
Snow Family – Edge Computing

- Snowcone & Snowcone SSD (smaller)
 - 2 CPUs, 4 GB of memory, wired or wireless access
 - USB-C power using a cord or the optional battery
- Snowball Edge – Compute Optimized
 - 104 vCPUs, 416 GiB of RAM
 - Optional GPU (useful for video processing or machine learning)
 - 28 TB NVMe or 42TB HDD usable storage
- Snowball Edge – Storage Optimized
 - Up to 40 vCPUs, 80 GiB of RAM, 80 TB storage
 - Object storage clustering available
- All: Can run EC2 Instances & AWS Lambda functions (using AWS IoT Greengrass)
- Long-term deployment options: 1 and 3 years discounted pricing



AWS OpsHub

- Historically, to use Snow Family devices, you needed a CLI (Command Line Interface tool)
- Today, you can use **AWS OpsHub** (a software you install on your computer / laptop) to manage your Snow Family Device
 - Unlocking and configuring single or clustered devices
 - Transferring files
 - Launching and managing instances running on Snow Family Devices
 - Monitor device metrics (storage capacity, active instances on your device)
 - Launch compatible AWS services on your devices (ex: Amazon EC2 instances, AWS DataSync, Network File System (NFS))



<https://aws.amazon.com/blogs/aws/aws-snowball-edge-update/>

Solution Architecture: Snowball into Glacier

- Snowball cannot import to Glacier directly
- You must use Amazon S3 first, in combination with an S3 lifecycle policy



Amazon FSx – Overview



- Launch 3rd party high-performance file systems on AWS
- Fully managed service



FSx for Lustre



FSx for
Windows
File Server



FSx for
NetApp ONTAP



FSx for
OpenZFS

Amazon FSx for Windows (File Server)



- FSx for Windows is a fully managed Windows file system share drive
- Supports SMB protocol & Windows NTFS
- Microsoft Active Directory integration, ACLs, user quotas
- Can be mounted on Linux EC2 instances
- Supports Microsoft's Distributed File System (DFS) Namespaces (group files across multiple FS)
- Scale up to 10s of GB/s, millions of IOPS, 100s PB of data
- Storage Options:
 - SSD – latency sensitive workloads (databases, media processing, data analytics, ...)
 - HDD – broad spectrum of workloads (home directory, CMS, ...)
- Can be accessed from your on-premises infrastructure (VPN or Direct Connect)
- Can be configured to be Multi-AZ (high availability)
- Data is backed-up daily to S3

Amazon FSx for Lustre



- Lustre is a type of parallel distributed file system, for large-scale computing
- The name Lustre is derived from “Linux” and “cluster”
- Machine Learning, **High Performance Computing (HPC)**
- Video Processing, Financial Modeling, Electronic Design Automation
- Scales up to 100s GB/s, millions of IOPS, sub-ms latencies
- Storage Options:
 - SSD – low-latency, IOPS intensive workloads, small & random file operations
 - HDD – throughput-intensive workloads, large & sequential file operations
- **Seamless integration with S3**
 - Can “read S3” as a file system (through FSx)
 - Can write the output of the computations back to S3 (through FSx)
- **Can be used from on-premises servers (VPN or Direct Connect)**

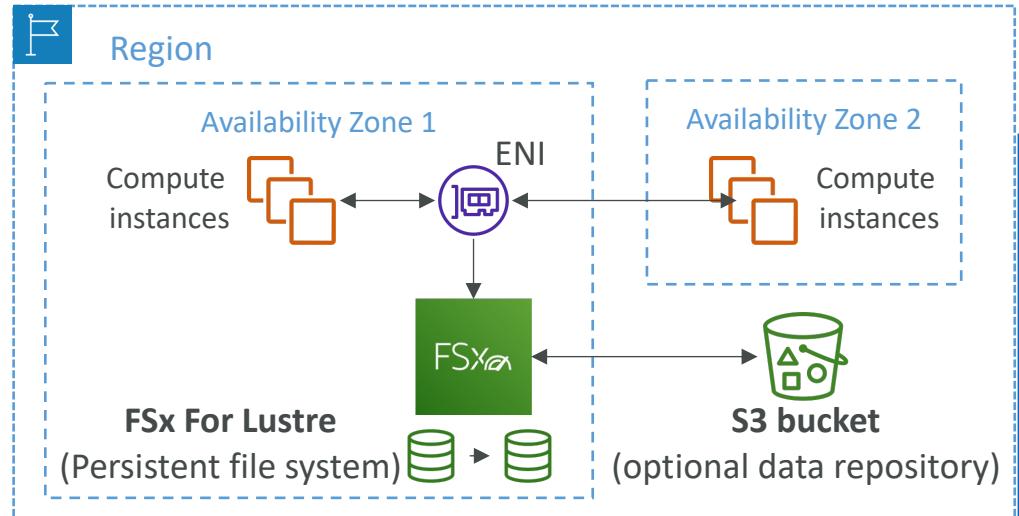
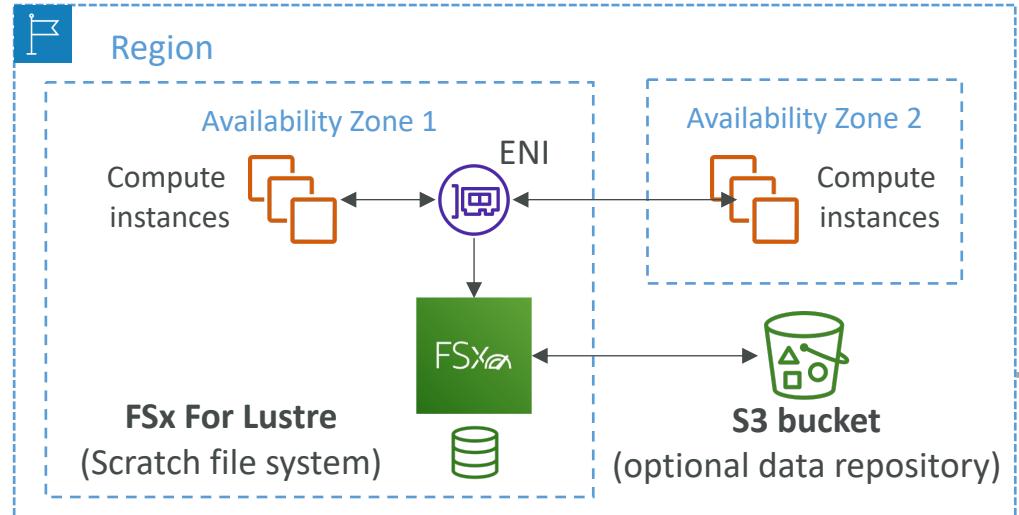
FSx Lustre - File System Deployment Options

• Scratch File System

- Temporary storage
- Data is not replicated (doesn't persist if file server fails)
- High burst (6x faster, 200MBps per TiB)
- Usage: short-term processing, optimize costs

• Persistent File System

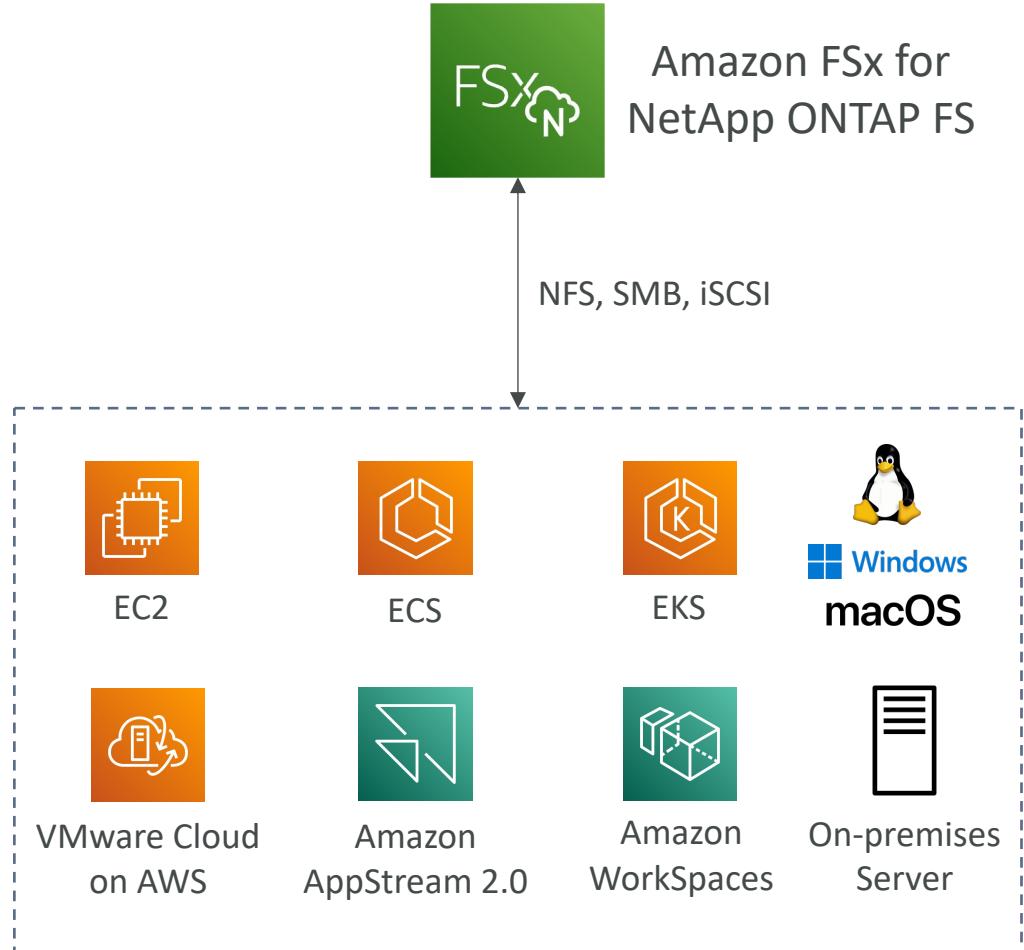
- Long-term storage
- Data is replicated within same AZ
- Replace failed files within minutes
- Usage: long-term processing, sensitive data



Amazon FSx for NetApp ONTAP



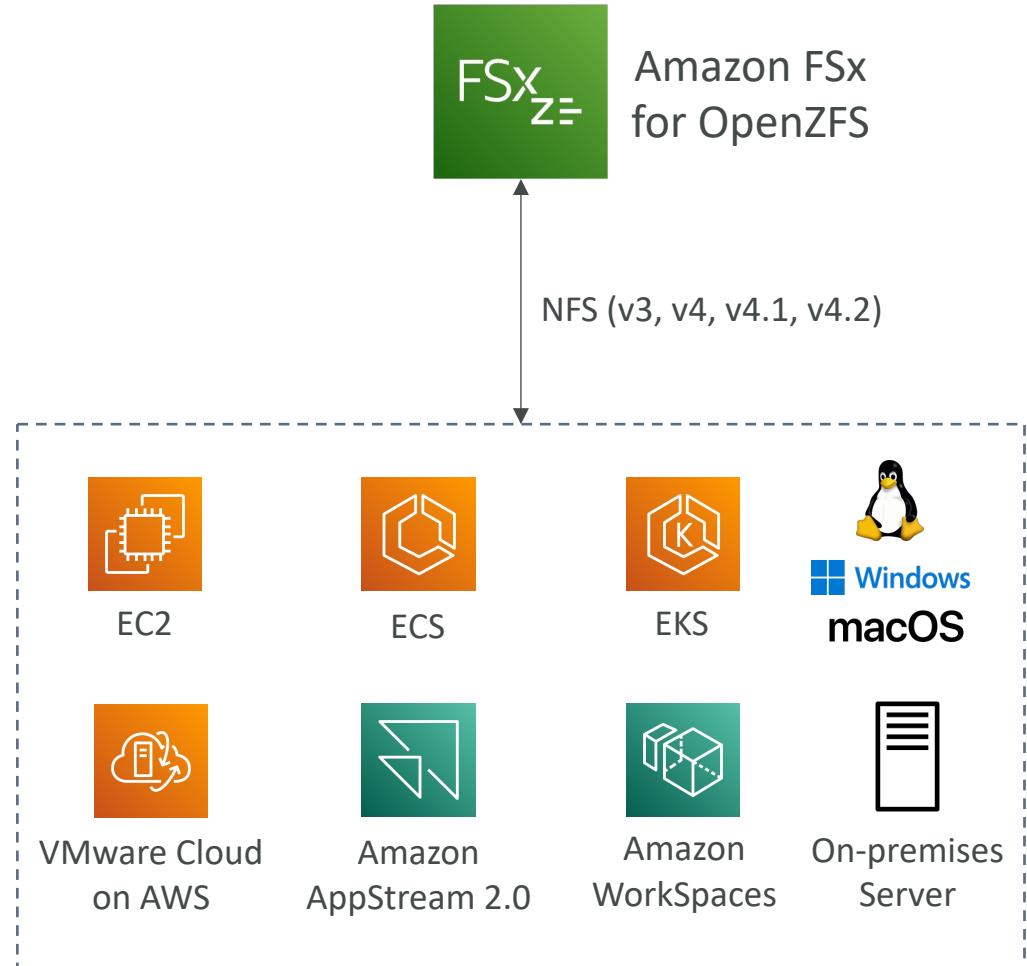
- Managed NetApp ONTAP on AWS
- File System compatible with NFS, SMB, iSCSI protocol
- Move workloads running on ONTAP or NAS to AWS
- Works with:
 - Linux
 - Windows
 - MacOS
 - VMware Cloud on AWS
 - Amazon Workspaces & AppStream 2.0
 - Amazon EC2, ECS and EKS
- Storage shrinks or grows automatically
- Snapshots, replication, low-cost, compression and data de-duplication
- Point-in-time instantaneous cloning (helpful for testing new workloads)



Amazon FSx for OpenZFS



- Managed OpenZFS file system on AWS
- File System compatible with NFS (v3, v4, v4.1, v4.2)
- Move workloads running on ZFS to AWS
- Works with:
 - Linux
 - Windows
 - MacOS
 - VMware Cloud on AWS
 - Amazon Workspaces & AppStream 2.0
 - Amazon EC2, ECS and EKS
- Up to 1,000,000 IOPS with < 0.5ms latency
- Snapshots, compression and low-cost
- **Point-in-time instantaneous cloning (helpful for testing new workloads)**



Hybrid Cloud for Storage

- AWS is pushing for "hybrid cloud"
 - Part of your infrastructure is on the cloud
 - Part of your infrastructure is on-premises
- This can be due to
 - Long cloud migrations
 - Security requirements
 - Compliance requirements
 - IT strategy
- S3 is a proprietary storage technology (unlike EFS / NFS), so how do you expose the S3 data on-premises?
- AWS Storage Gateway!

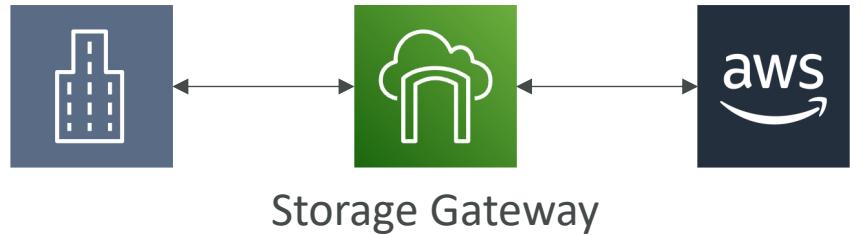
AWS Storage Cloud Native Options



AWS Storage Gateway

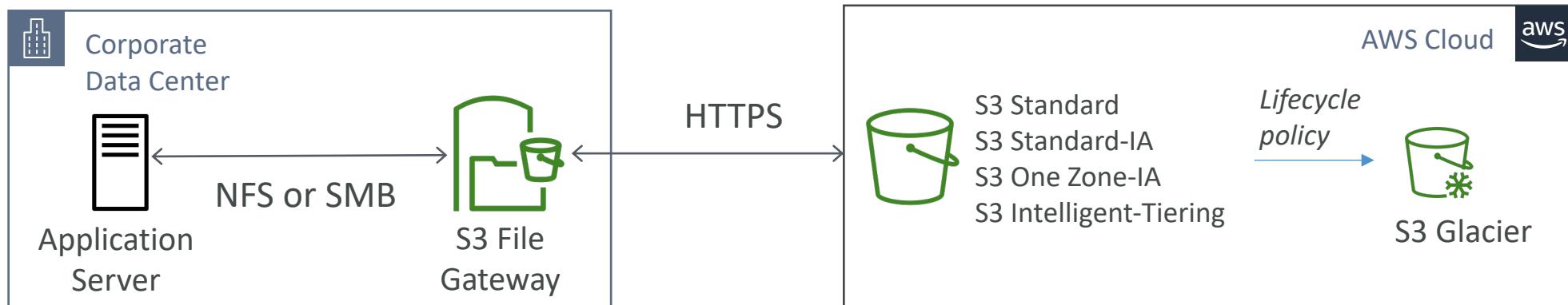


- Bridge between on-premises data and cloud data
- Use cases:
 - disaster recovery
 - backup & restore
 - tiered storage
 - on-premises cache & low-latency files access
- Types of Storage Gateway:
 - S3 File Gateway
 - FSx File Gateway
 - Volume Gateway
 - Tape Gateway



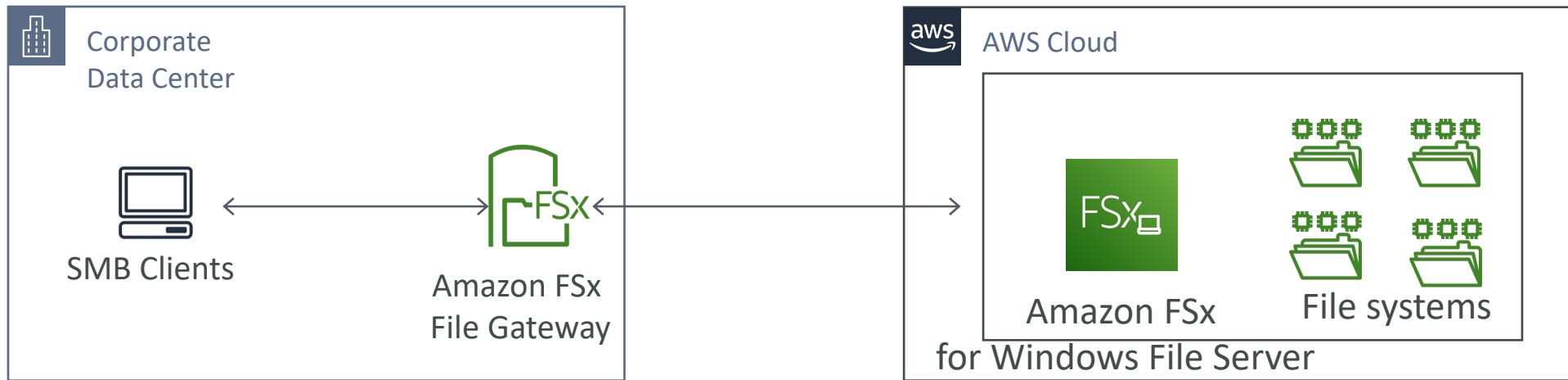
Amazon S3 File Gateway

- Configured S3 buckets are accessible using the NFS and SMB protocol
- Most recently used data is cached in the file gateway
- Supports S3 Standard, S3 Standard IA, S3 One Zone A, S3 Intelligent Tiering
- Transition to S3 Glacier using a Lifecycle Policy
- Bucket access using IAM roles for each File Gateway
- SMB Protocol has integration with Active Directory (AD) for user authentication



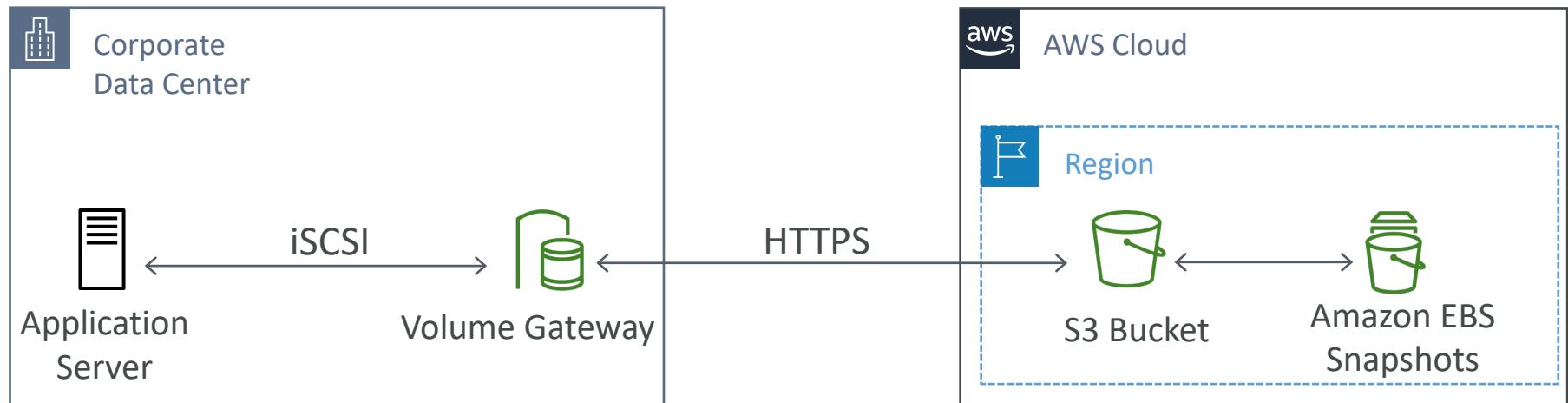
Amazon FSx File Gateway

- Native access to Amazon FSx for Windows File Server
- Local cache for frequently accessed data
- Windows native compatibility (SMB, NTFS, Active Directory...)
- Useful for group file shares and home directories



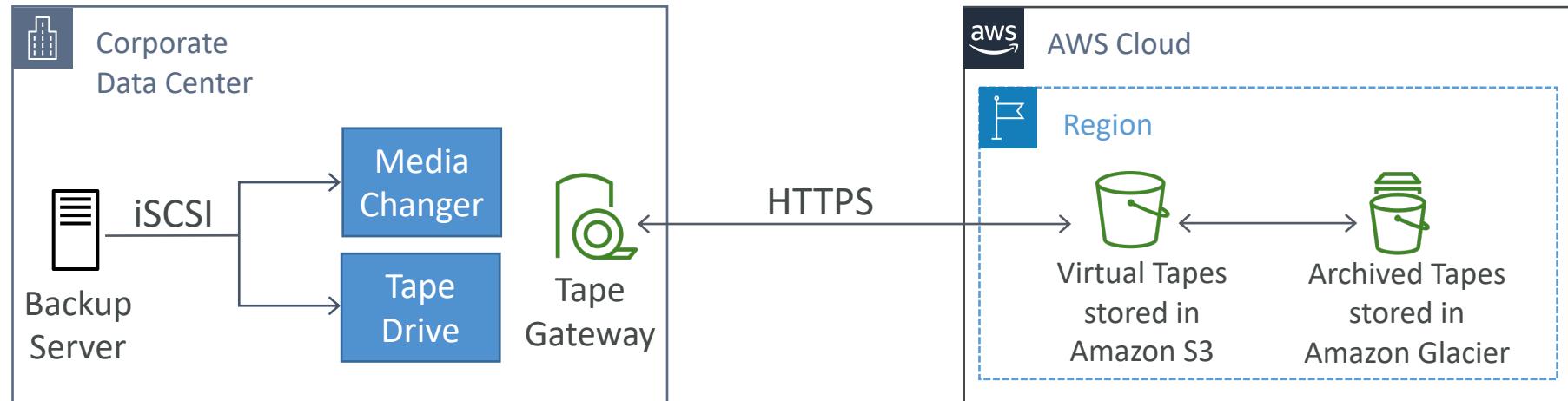
Volume Gateway

- Block storage using iSCSI protocol backed by S3
- Backed by EBS snapshots which can help restore on-premises volumes!
- **Cached volumes:** low latency access to most recent data
- **Stored volumes:** entire dataset is on premise, scheduled backups to S3



Tape Gateway

- Some companies have backup processes using physical tapes (!)
- With Tape Gateway, companies use the same processes but, in the cloud
- Virtual Tape Library (VTL) backed by Amazon S3 and Glacier
- Back up data using existing tape-based processes (and iSCSI interface)
- Works with leading backup software vendors



Storage Gateway – Hardware appliance

- Using Storage Gateway means you need on-premises virtualization
- Otherwise, you can use a **Storage Gateway Hardware Appliance**
- You can buy it on amazon.com

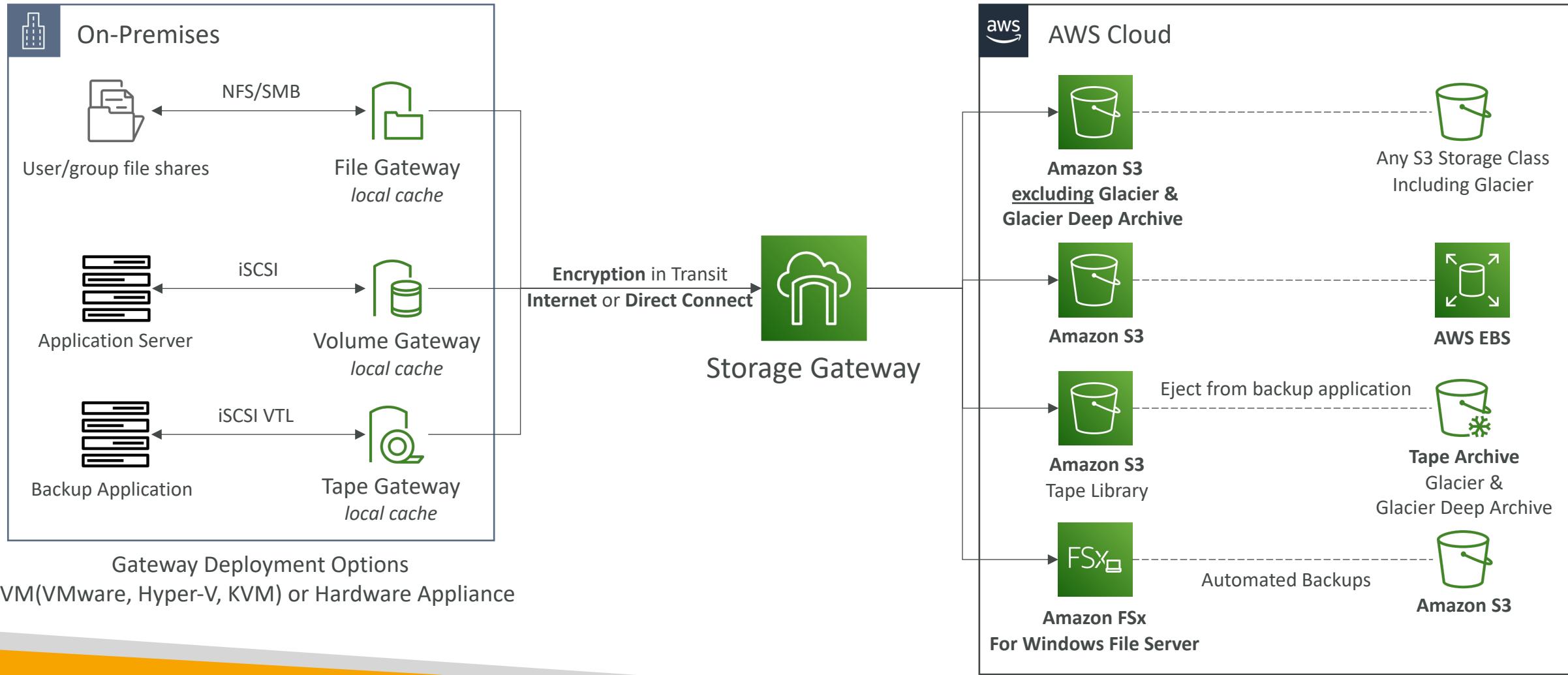
- Works with File Gateway, Volume Gateway, Tape Gateway
- Has the required CPU, memory, network, SSD cache resources
- Helpful for daily NFS backups in small data centers

Select host platform

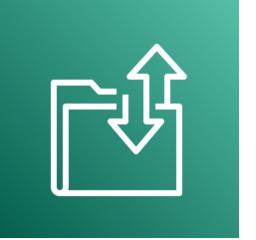
- VMware ESXi
- Microsoft Hyper-V 2012R2/2016
- Linux KVM
- Amazon EC2
- Hardware Appliance

[Buy on Amazon](#)[Activate Appliance](#)

AWS Storage Gateway

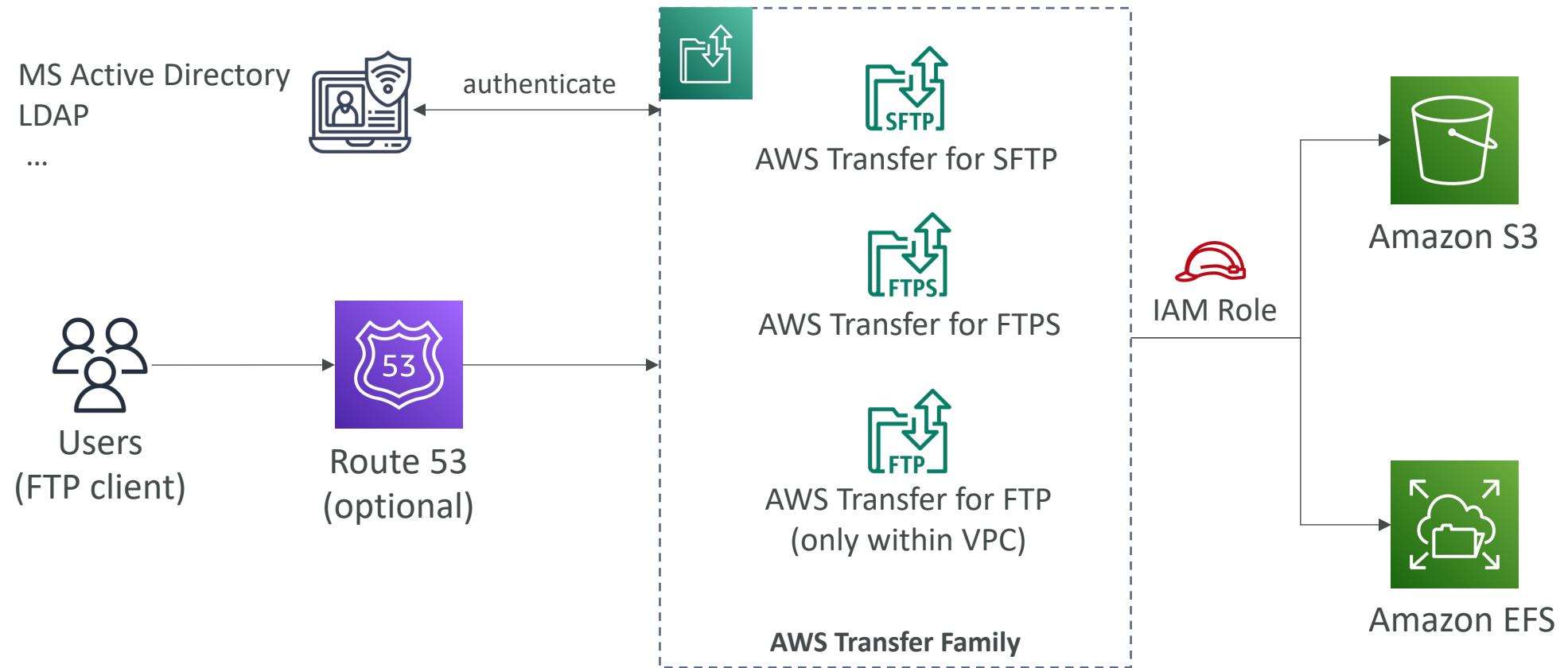


AWS Transfer Family



- A fully-managed service for file transfers into and out of Amazon S3 or Amazon EFS using the FTP protocol
- Supported Protocols
 - AWS Transfer for FTP (File Transfer Protocol (FTP))
 - AWS Transfer for FTPS (File Transfer Protocol over SSL (FTPS))
 - AWS Transfer for SFTP (Secure File Transfer Protocol (SFTP))
- Managed infrastructure, Scalable, Reliable, Highly Available (multi-AZ)
- Pay per provisioned endpoint per hour + data transfers in GB
- Store and manage users' credentials within the service
- Integrate with existing authentication systems (Microsoft Active Directory, LDAP, Okta, Amazon Cognito, custom)
- Usage: sharing files, public datasets, CRM, ERP, ...

AWS Transfer Family



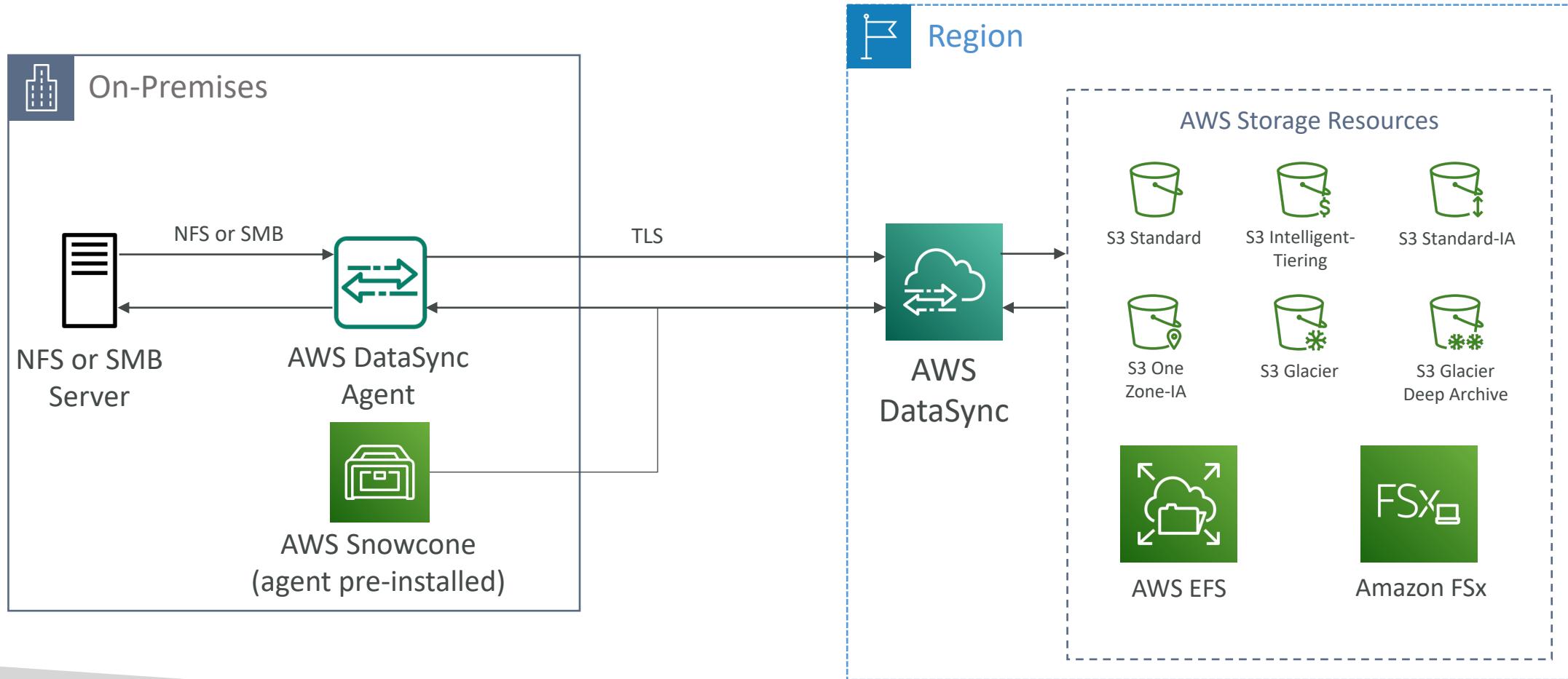


AWS DataSync

- Move large amount of data to and from
 - On-premises / other cloud to AWS (NFS, SMB, HDFS, S3 API...) – needs agent
 - AWS to AWS (different storage services) – no agent needed
- Can synchronize to:
 - Amazon S3 (any storage classes – including Glacier)
 - Amazon EFS
 - Amazon FSx (Windows, Lustre, NetApp, OpenZFS...)
- Replication tasks can be scheduled hourly, daily, weekly
- File permissions and metadata are preserved (NFS POSIX, SMB...)
- One agent task can use 10 Gbps, can setup a bandwidth limit

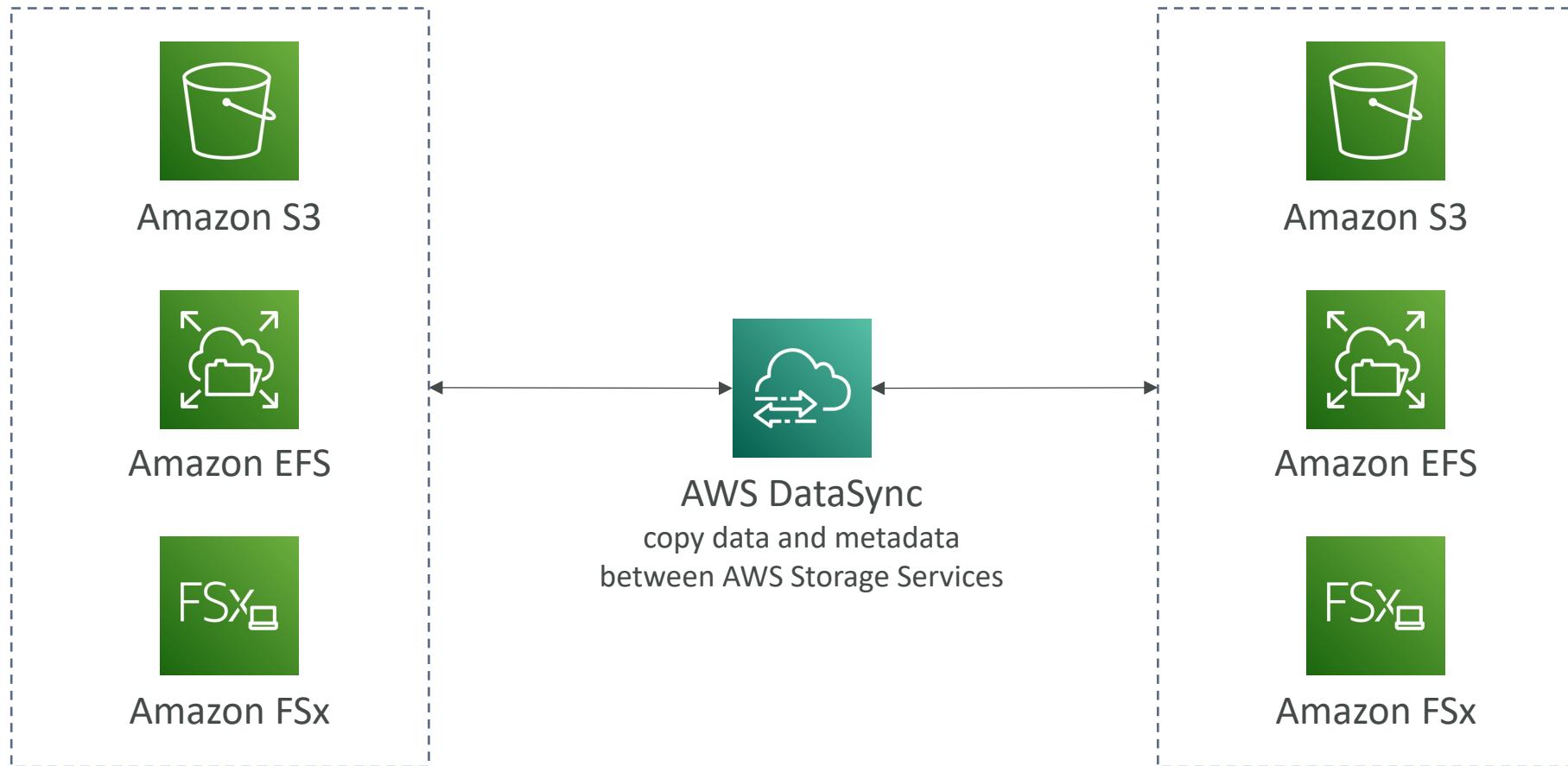
AWS DataSync

NFS / SMB to AWS (S3, EFS, FSx...)



AWS DataSync

Transfer between AWS storage services



Storage Comparison

- S3: Object Storage
- S3 Glacier: Object Archival
- EBS volumes: Network storage for one EC2 instance at a time
- Instance Storage: Physical storage for your EC2 instance (high IOPS)
- EFS: Network File System for Linux instances, POSIX filesystem
- FSx for Windows: Network File System for Windows servers
- FSx for Lustre: High Performance Computing Linux file system
- FSx for NetApp ONTAP: High OS Compatibility
- FSx for OpenZFS: Managed ZFS file system
- Storage Gateway: S3 & FSx File Gateway, Volume Gateway (cache & stored), Tape Gateway
- Transfer Family: FTP, FTPS, SFTP interface on top of Amazon S3 or Amazon EFS
- DataSync: Schedule data sync from on-premises to AWS, or AWS to AWS
- Snowcone / Snowball / Snowmobile: to move large amount of data to the cloud, physically
- Database: for specific workloads, usually with indexing and querying

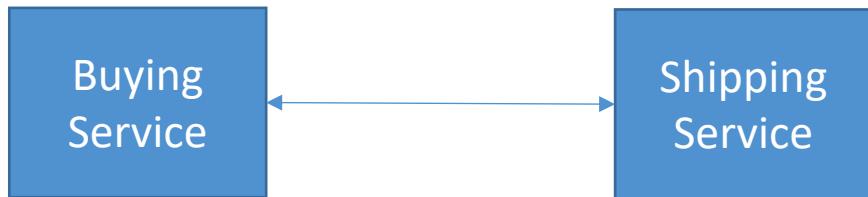
AWS Integration & Messaging

SQS, SNS & Kinesis

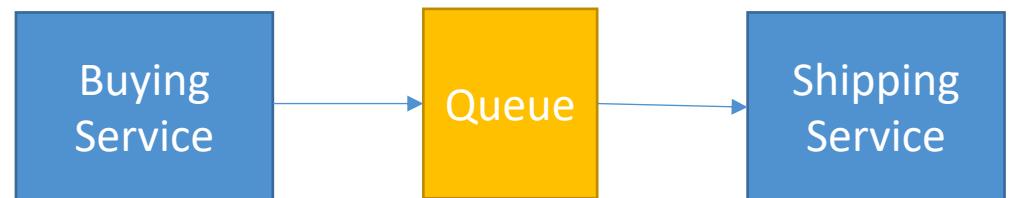
Section Introduction

- When we start deploying multiple applications, they will inevitably need to communicate with one another
- There are two patterns of application communication

**1) Synchronous communications
(application to application)**



**2) Asynchronous / Event based
(application to queue to application)**

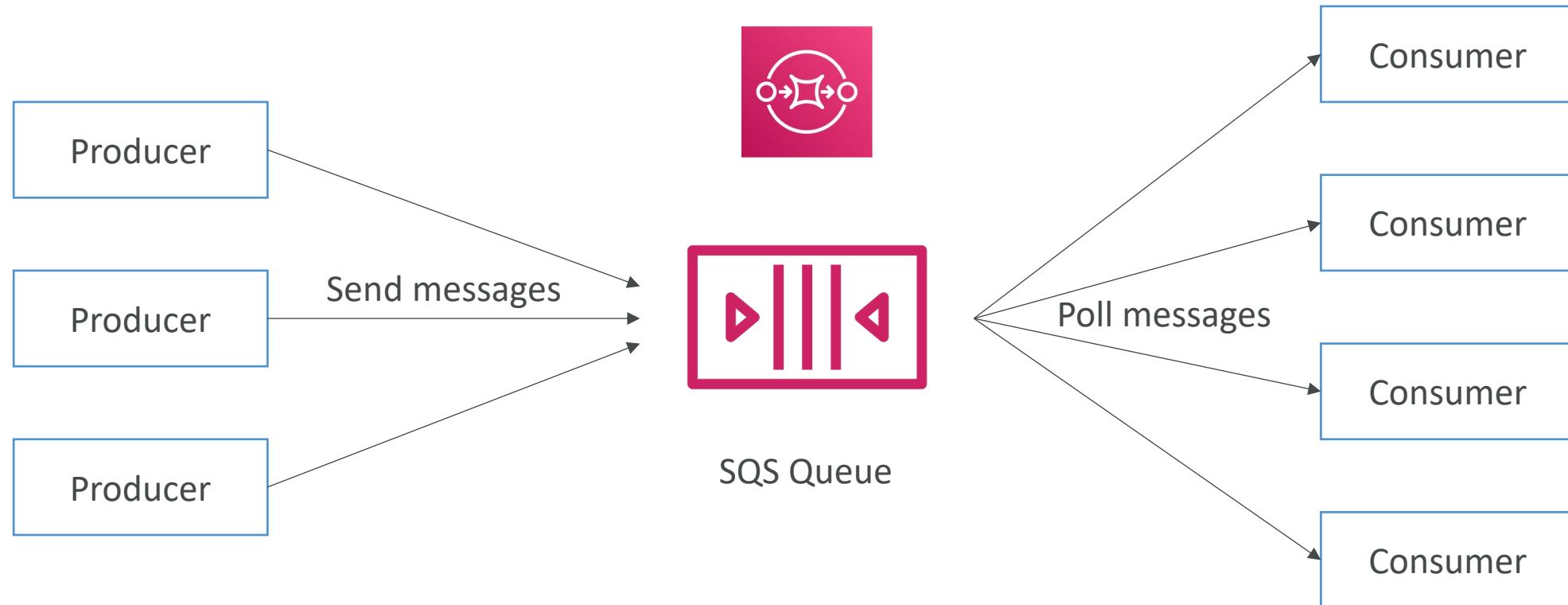


Section Introduction

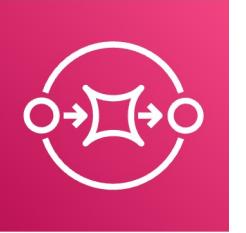
- Synchronous between applications can be problematic if there are sudden spikes of traffic
- What if you need to suddenly encode 1000 videos but usually it's 10?
- In that case, it's better to **decouple** your applications,
 - using SQS: queue model
 - using SNS: pub/sub model
 - using Kinesis: real-time streaming model
- These services can scale independently from our application!

Amazon SQS

What's a queue?



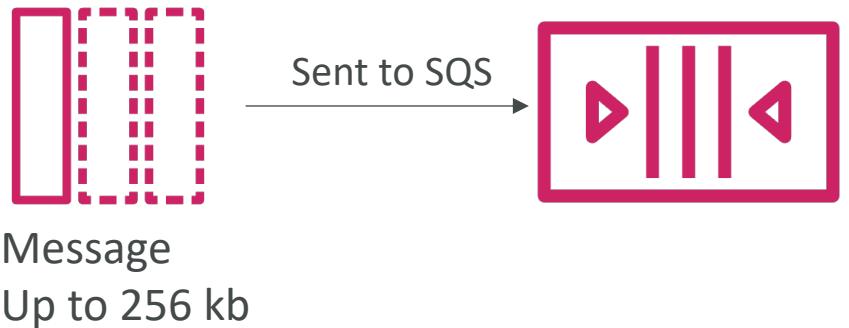
Amazon SQS – Standard Queue



- Oldest offering (over 10 years old)
- Fully managed service, used to **decouple applications**
- Attributes:
 - Unlimited throughput, unlimited number of messages in queue
 - Default retention of messages: 4 days, maximum of 14 days
 - Low latency (<10 ms on publish and receive)
 - Limitation of 256KB per message sent
- Can have duplicate messages (at least once delivery, occasionally)
- Can have out of order messages (best effort ordering)

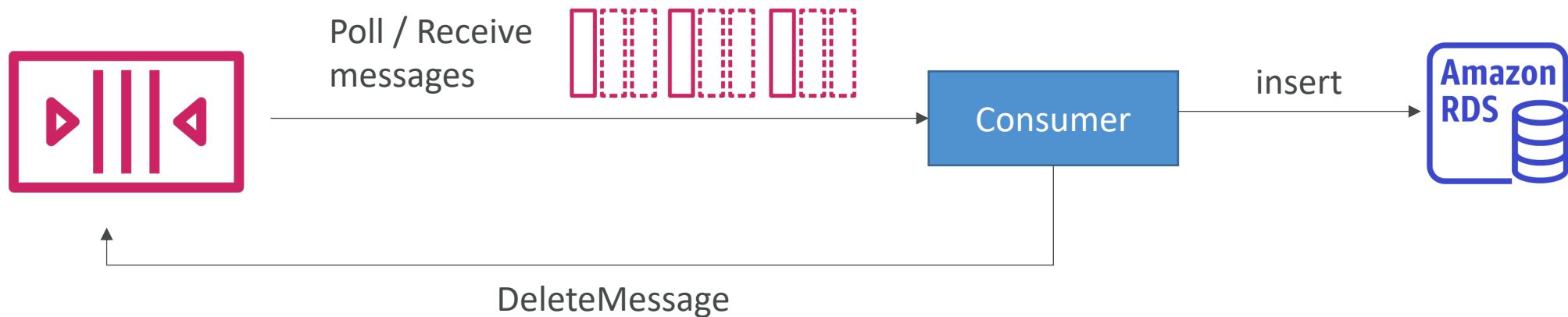
SQS – Producing Messages

- Produced to SQS using the SDK (SendMessage API)
- The message is **persisted** in SQS until a consumer deletes it
- Message retention: default 4 days, up to 14 days
- Example: send an order to be processed
 - Order id
 - Customer id
 - Any attributes you want
- SQS standard: unlimited throughput

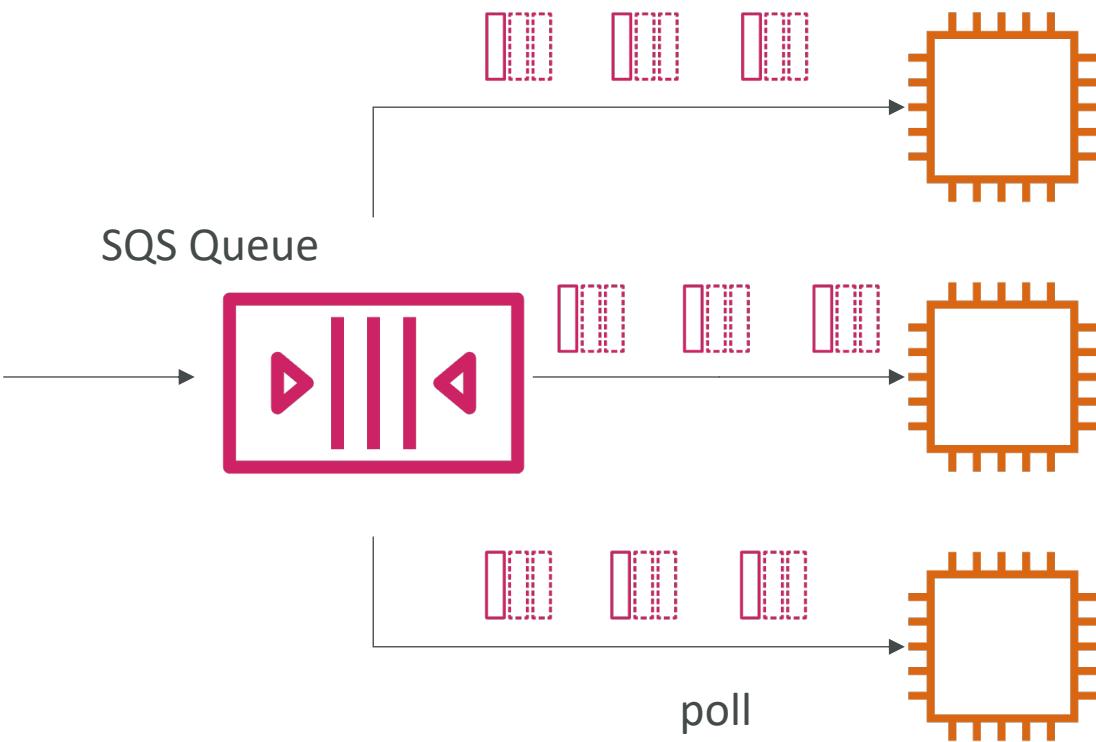


SQS – Consuming Messages

- Consumers (running on EC2 instances, servers, or AWS Lambda)...
- Poll SQS for messages (receive up to 10 messages at a time)
- Process the messages (example: insert the message into an RDS database)
- Delete the messages using the DeleteMessage API

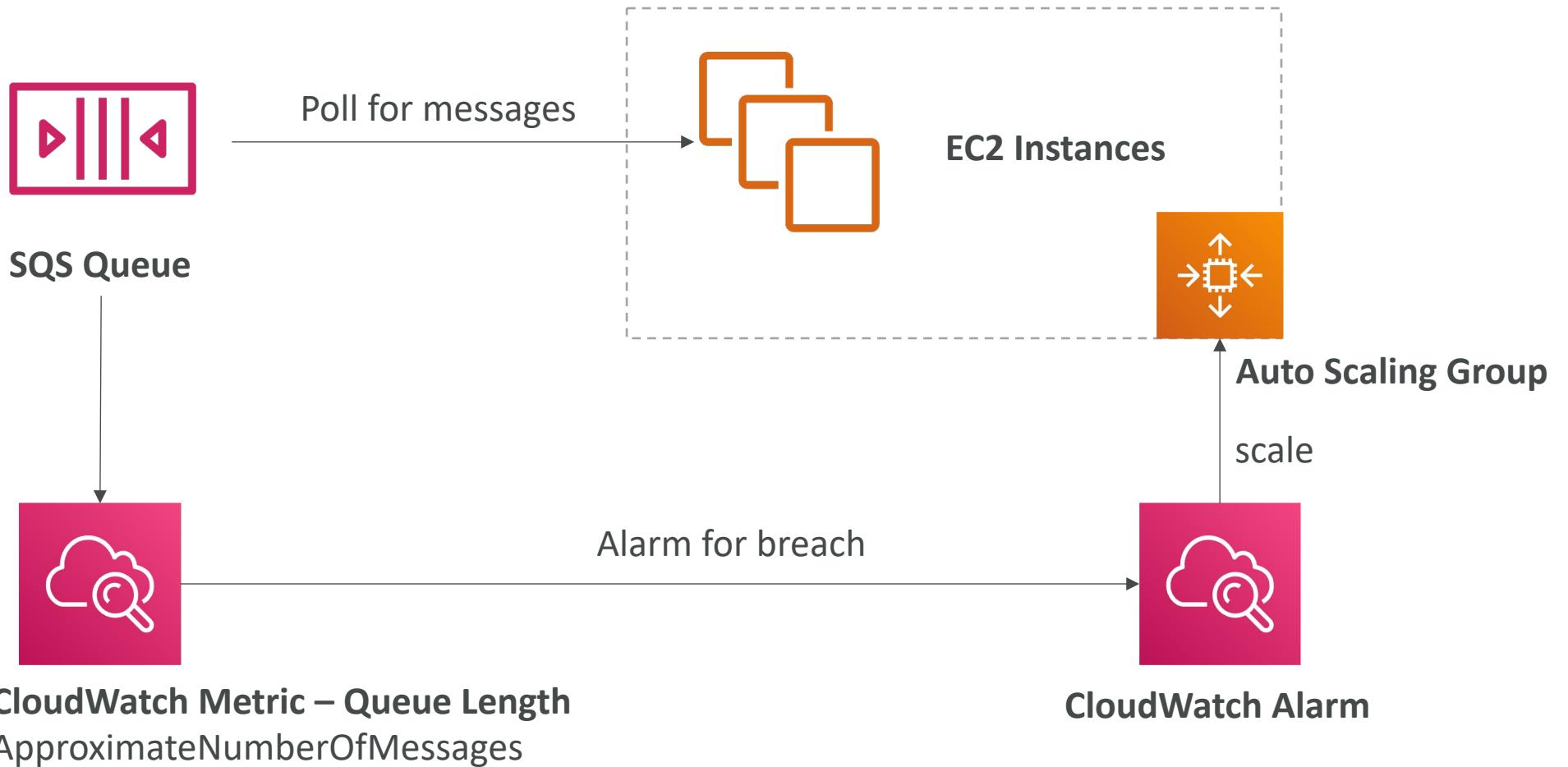


SQS – Multiple EC2 Instances Consumers

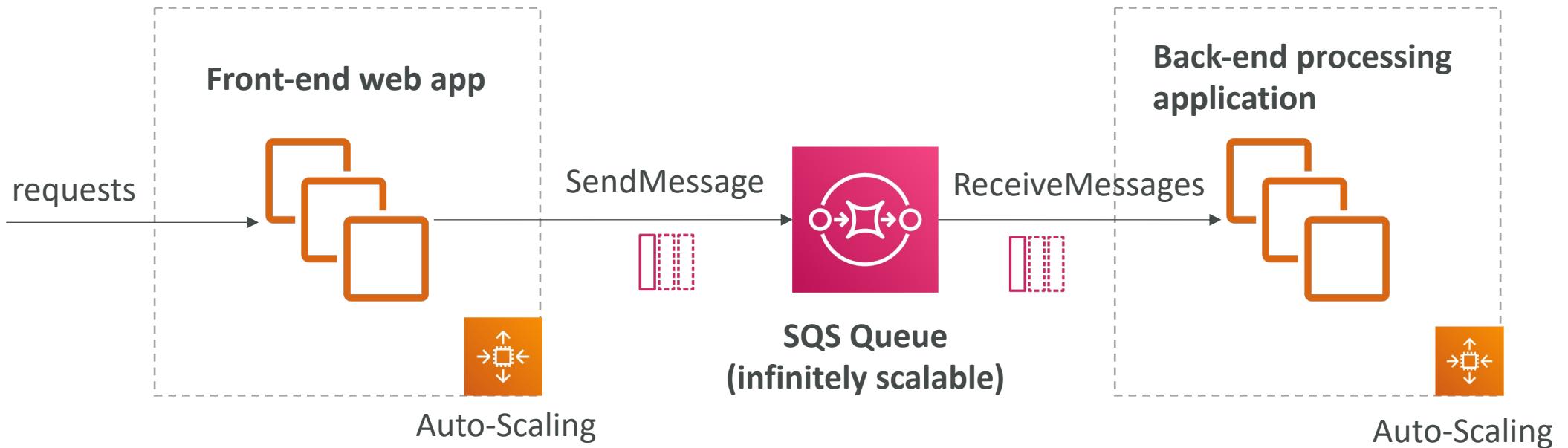


- Consumers receive and process messages in parallel
- At least once delivery
- Best-effort message ordering
- Consumers delete messages after processing them
- We can scale consumers horizontally to improve throughput of processing

SQS with Auto Scaling Group (ASG)



SQS to decouple between application tiers

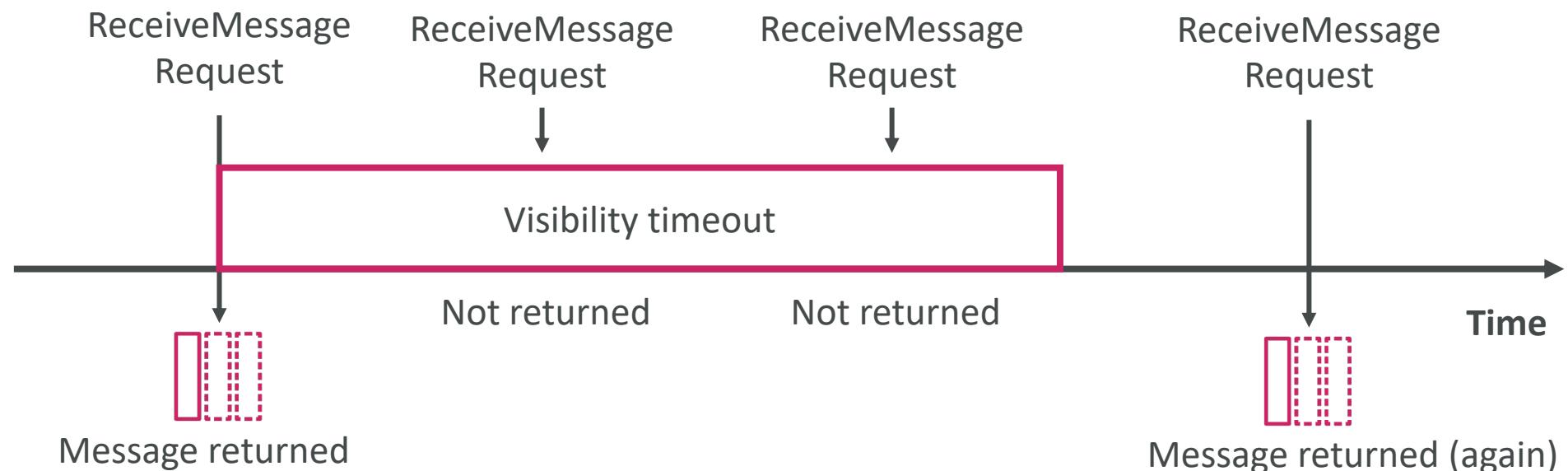


Amazon SQS - Security

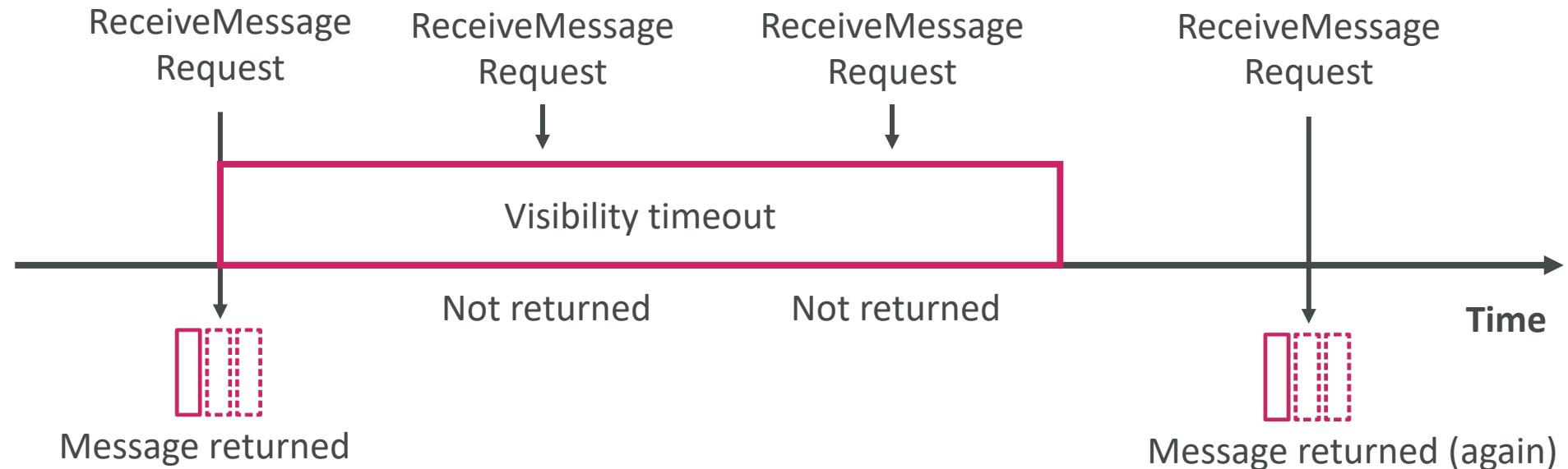
- **Encryption:**
 - In-flight encryption using HTTPS API
 - At-rest encryption using KMS keys
 - Client-side encryption if the client wants to perform encryption/decryption itself
- **Access Controls:** IAM policies to regulate access to the SQS API
- **SQS Access Policies** (similar to S3 bucket policies)
 - Useful for cross-account access to SQS queues
 - Useful for allowing other services (SNS, S3...) to write to an SQS queue

SQS – Message Visibility Timeout

- After a message is polled by a consumer, it becomes **invisible** to other consumers
- By default, the “message visibility timeout” is **30 seconds**
- That means the message has 30 seconds to be processed
- After the message visibility timeout is over, the message is “visible” in SQS



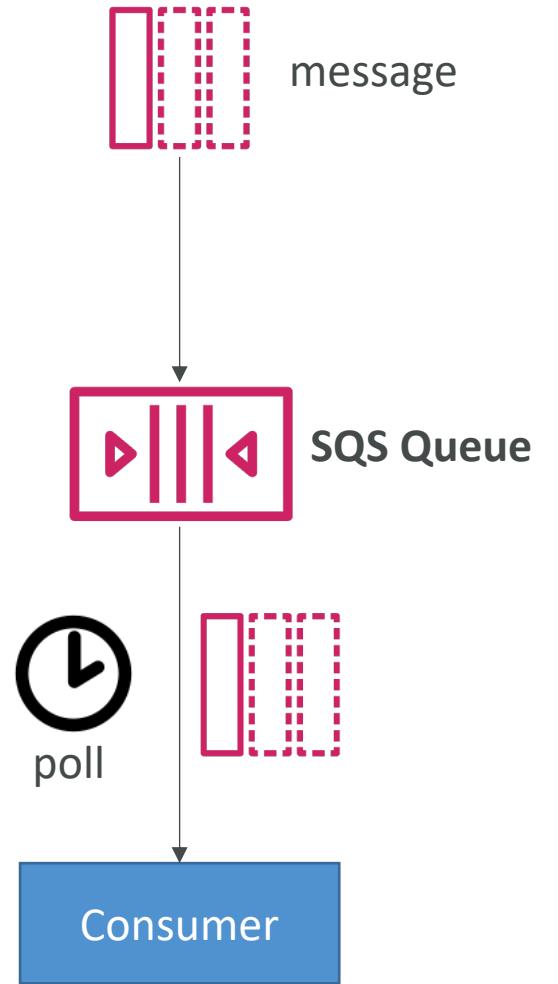
SQS – Message Visibility Timeout



- If a message is not processed within the visibility timeout, it will be processed **twice**
- A consumer could call the **ChangeMessageVisibility** API to get more time
- If visibility timeout is high (hours), and consumer crashes, re-processing will take time
- If visibility timeout is too low (seconds), we may get duplicates

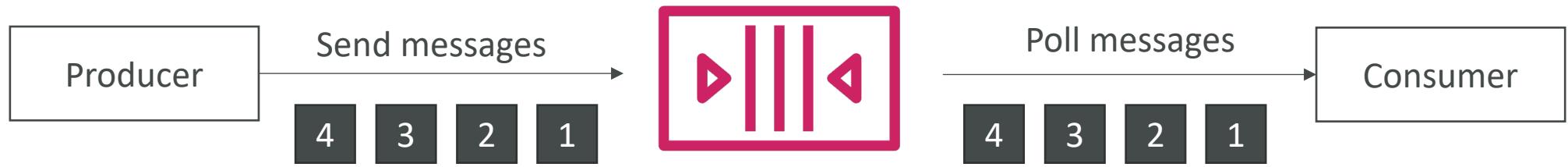
Amazon SQS - Long Polling

- When a consumer requests messages from the queue, it can optionally “wait” for messages to arrive if there are none in the queue
- This is called Long Polling
- LongPolling decreases the number of API calls made to SQS while increasing the efficiency and reducing latency of your application
- The wait time can be between 1 sec to 20 sec (20 sec preferable)
- Long Polling is preferable to Short Polling
- Long polling can be enabled at the queue level or at the API level using `WaitTimeSeconds`



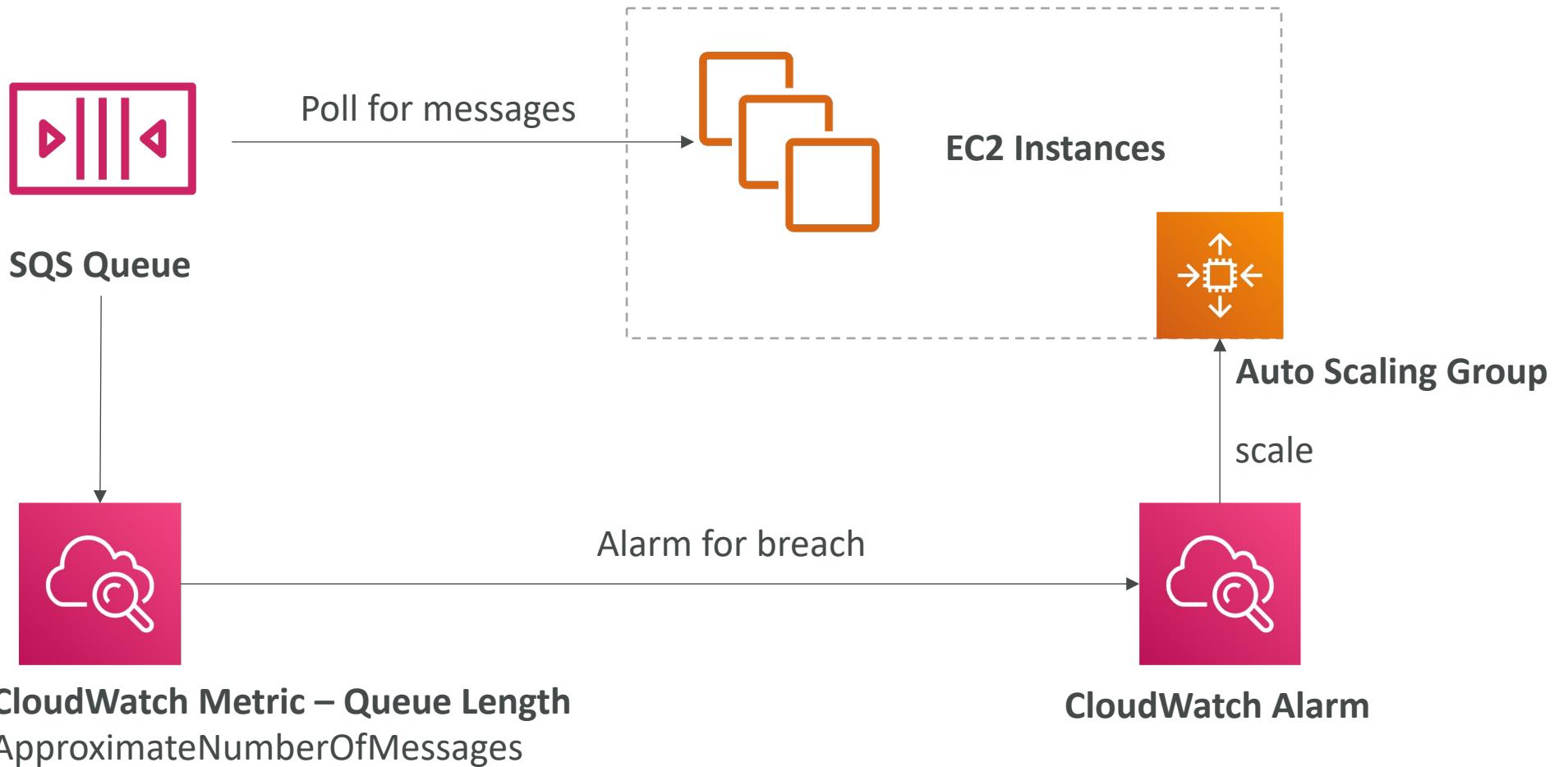
Amazon SQS – FIFO Queue

- FIFO = First In First Out (ordering of messages in the queue)



- Limited throughput: 300 msg/s without batching, 3000 msg/s with
- Exactly-once send capability (by removing duplicates)
- Messages are processed in order by the consumer

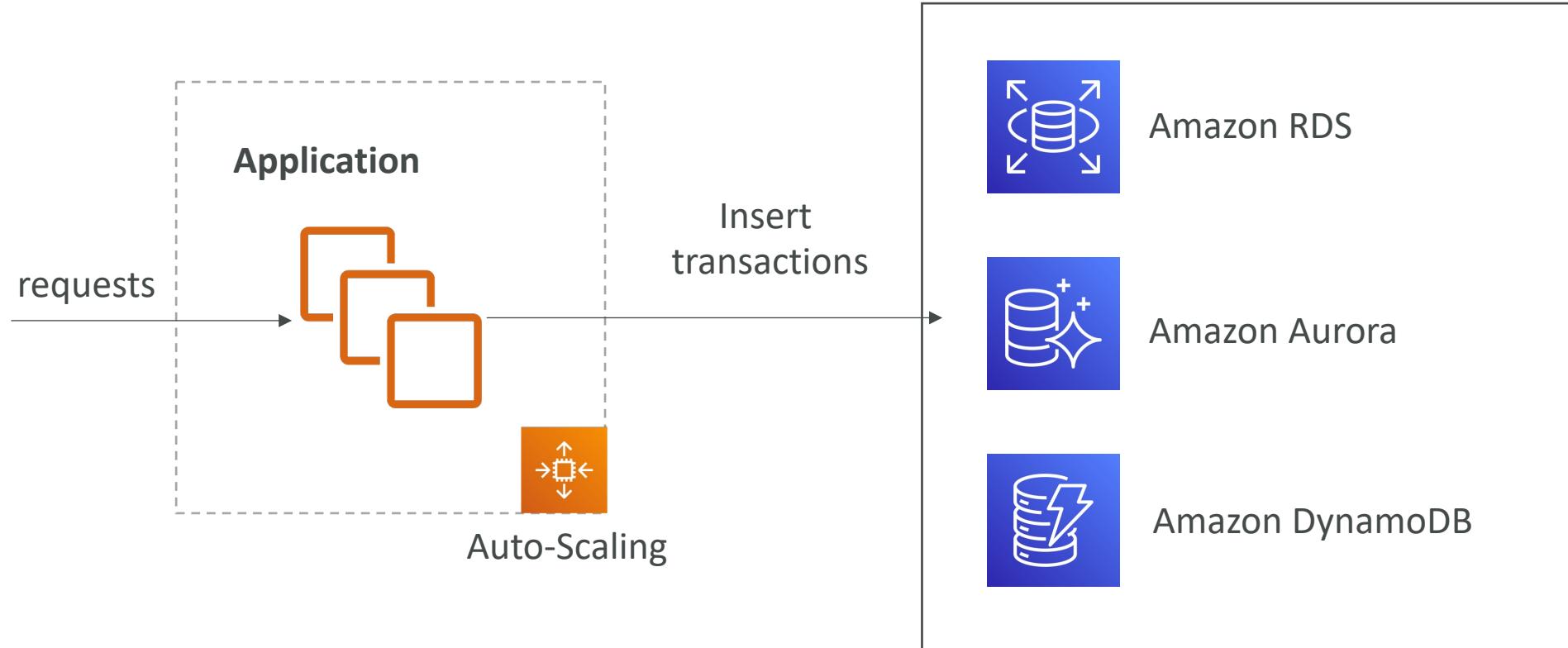
SQS with Auto Scaling Group (ASG)



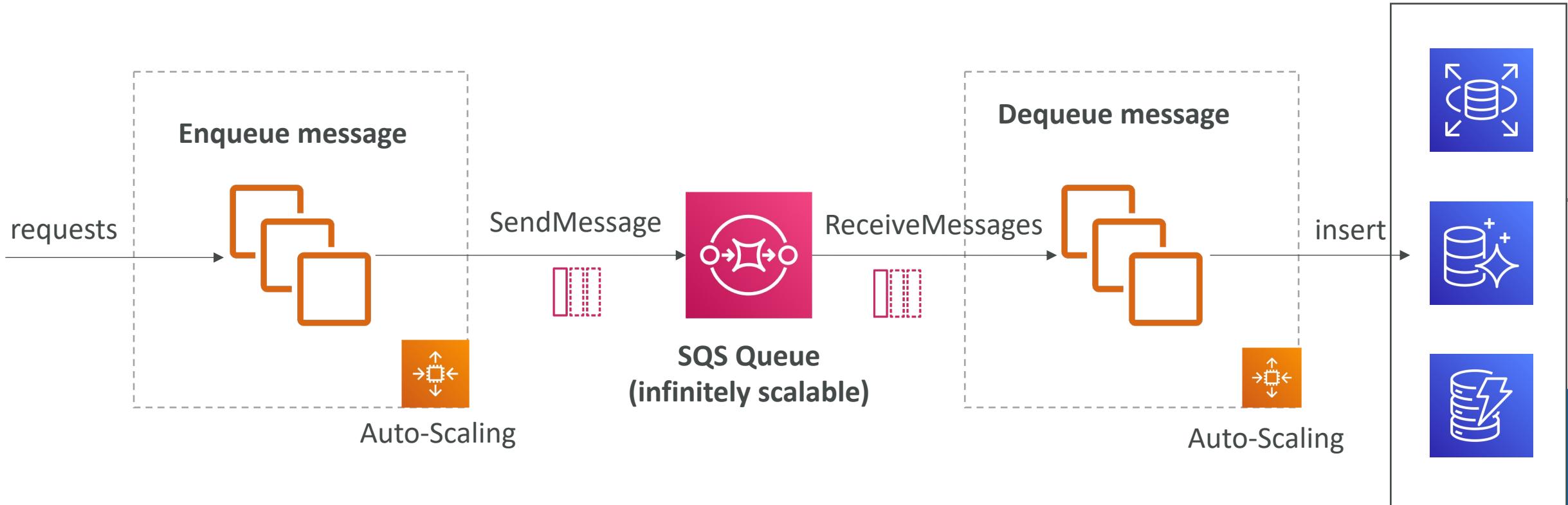
CloudWatch Metric – Queue Length
ApproximateNumberOfMessages

CloudWatch Alarm

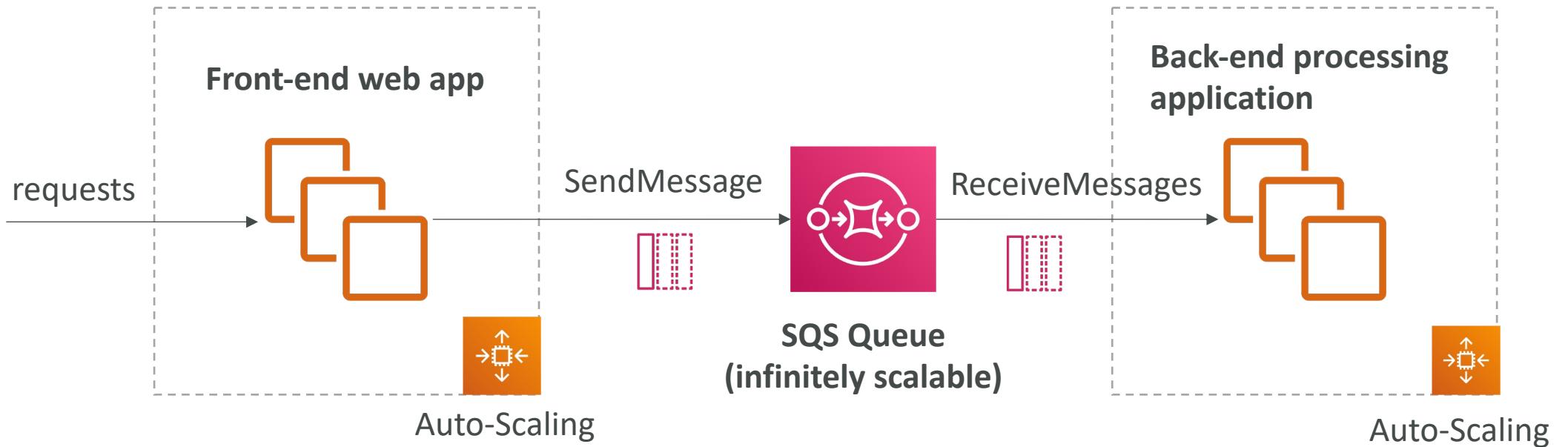
If the load is too big,
some transactions may be lost



SQS as a buffer to database writes

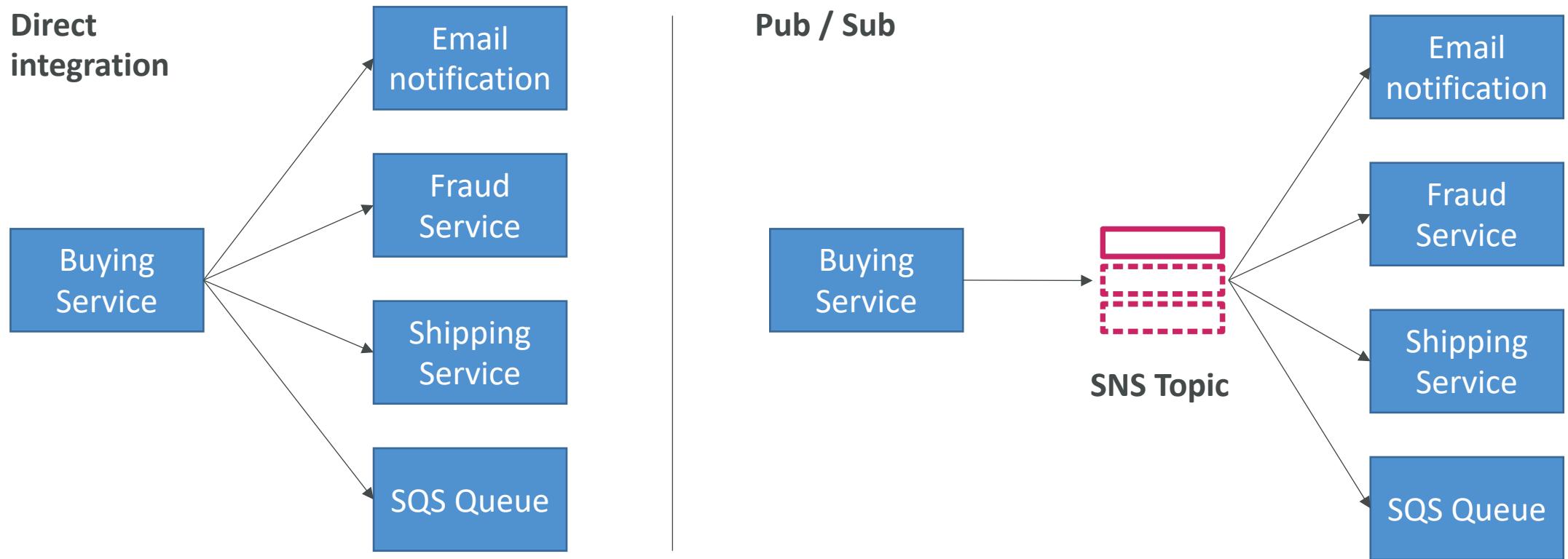


SQS to decouple between application tiers



Amazon SNS

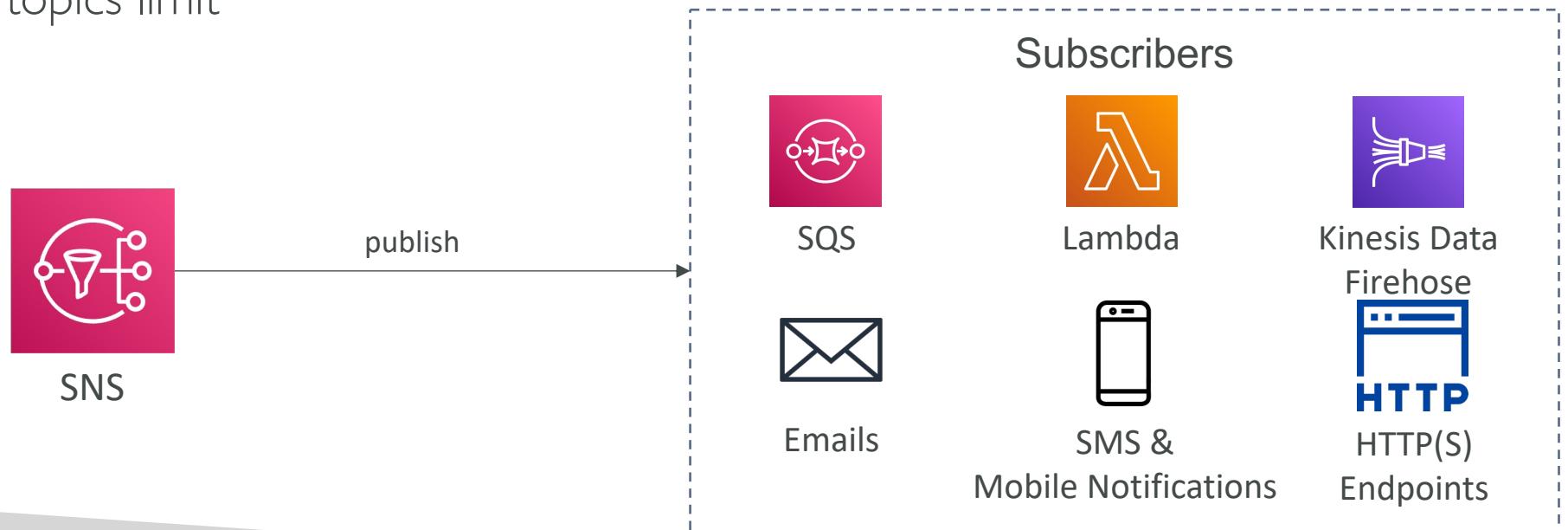
- What if you want to send one message to many receivers?



Amazon SNS

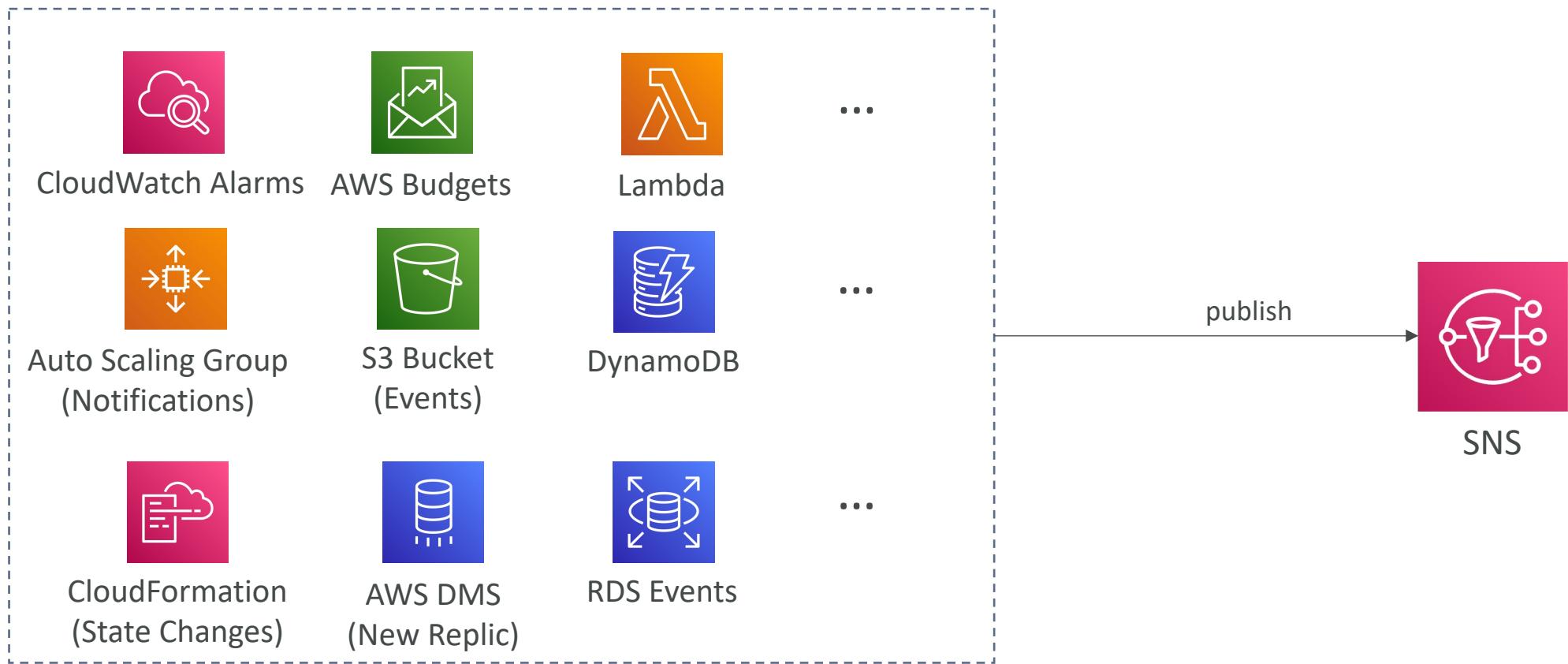


- The “event producer” only sends message to one SNS topic
- As many “event receivers” (subscriptions) as we want to listen to the SNS topic notifications
- Each subscriber to the topic will get all the messages (note: new feature to filter messages)
- Up to 12,500,000 subscriptions per topic
- 100,000 topics limit



SNS integrates with a lot of AWS services

- Many AWS services can send data directly to SNS for notifications



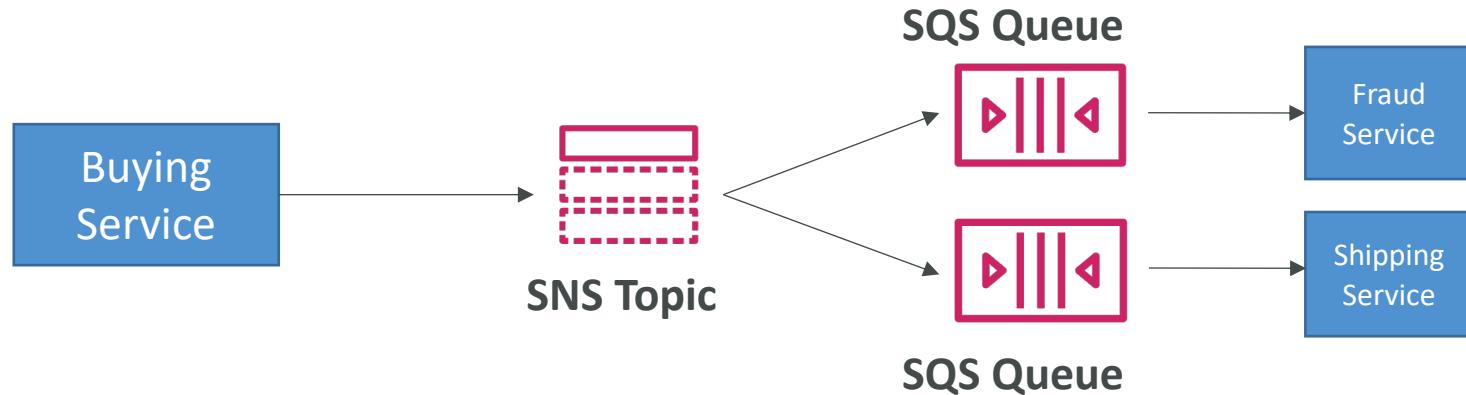
Amazon SNS – How to publish

- Topic Publish (using the SDK)
 - Create a topic
 - Create a subscription (or many)
 - Publish to the topic
- Direct Publish (for mobile apps SDK)
 - Create a platform application
 - Create a platform endpoint
 - Publish to the platform endpoint
 - Works with Google GCM, Apple APNS, Amazon ADM...

Amazon SNS – Security

- **Encryption:**
 - In-flight encryption using HTTPS API
 - At-rest encryption using KMS keys
 - Client-side encryption if the client wants to perform encryption/decryption itself
- **Access Controls:** IAM policies to regulate access to the SNS API
- **SNS Access Policies** (similar to S3 bucket policies)
 - Useful for cross-account access to SNS topics
 - Useful for allowing other services (S3...) to write to an SNS topic

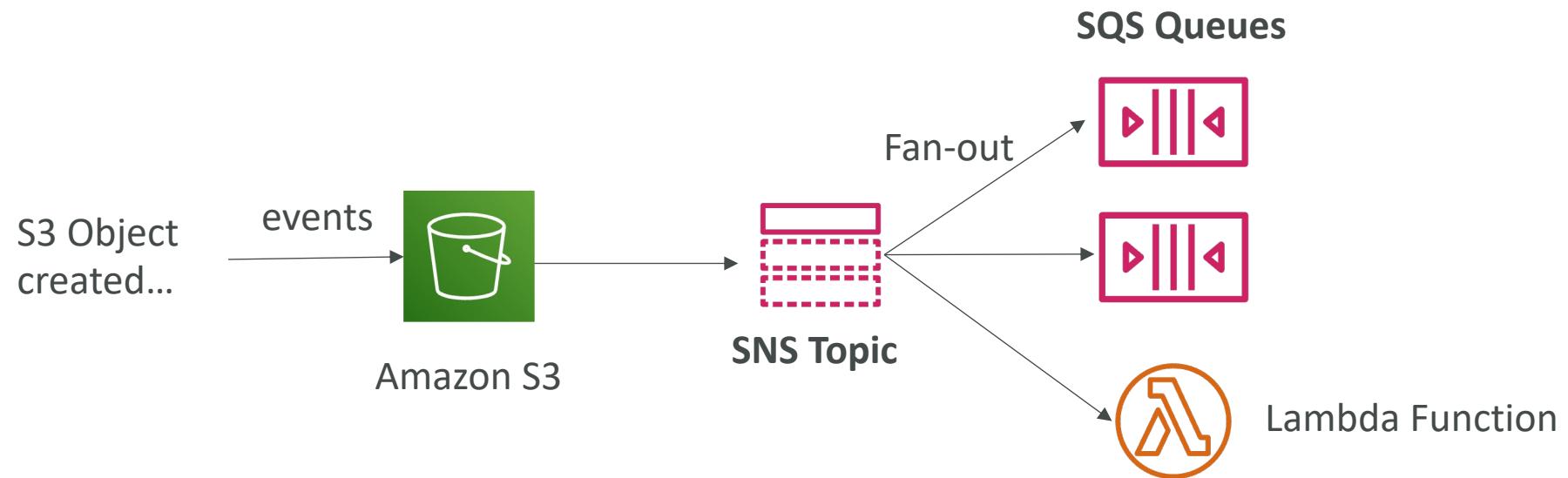
SNS + SQS: Fan Out



- Push once in SNS, receive in all SQS queues that are subscribers
- Fully decoupled, no data loss
- SQS allows for: data persistence, delayed processing and retries of work
- Ability to add more SQS subscribers over time
- Make sure your SQS queue **access policy** allows for SNS to write
- Cross-Region Delivery: works with SQS Queues in other regions

Application: S3 Events to multiple queues

- For the same combination of: **event type** (e.g. object create) and **prefix** (e.g. images/) you can only have one S3 Event rule
- If you want to send the same S3 event to many SQS queues, use fan-out



Application: SNS to Amazon S3 through Kinesis Data Firehose

- SNS can send to Kinesis and therefore we can have the following solutions architecture:

