

# 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](mailto:piracy@datacumulus.com). Thanks!
- Best of luck for the exam and happy learning!

# AWS Certified Solutions Architect Associate Course

## SAA-C02

# Welcome! We're starting in 5 minutes



- We're going to prepare for the Solutions Architect exam - SAA-C02
- 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 certification: 98.2%

## AWS Certified Solutions Architect - Associate

### Notice of Exam Results

Candidate: Stephane Maarek	Exam Date: January 30, 2019
Candidate ID: AWS00614912	Registration Number: 513425
Candidate Score: 982	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  
 243,574 Reviews  
 782,072 Students  
 38 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



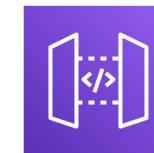
Amazon CloudWatch



AWS CloudFormation



AWS CloudTrail



Amazon API Gateway



Elastic Load Balancing



Amazon CloudFront



Amazon Kinesis



Amazon Route 53

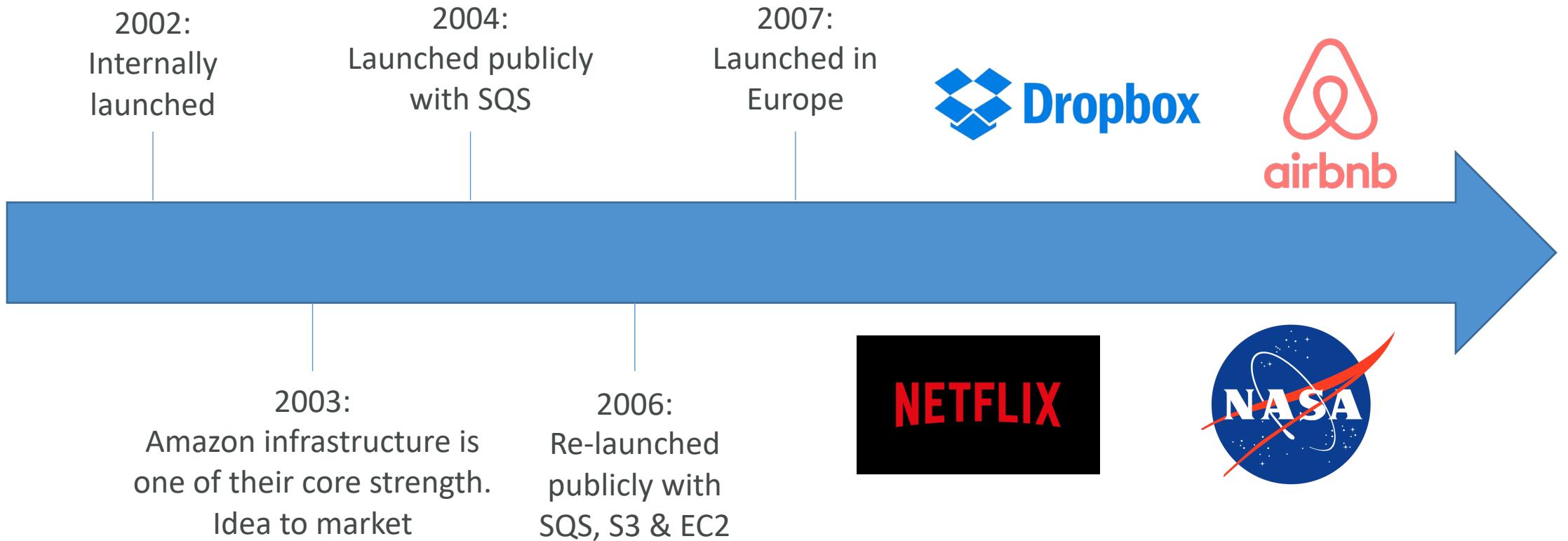
# Navigating the AWS spaghetti bowl



# 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



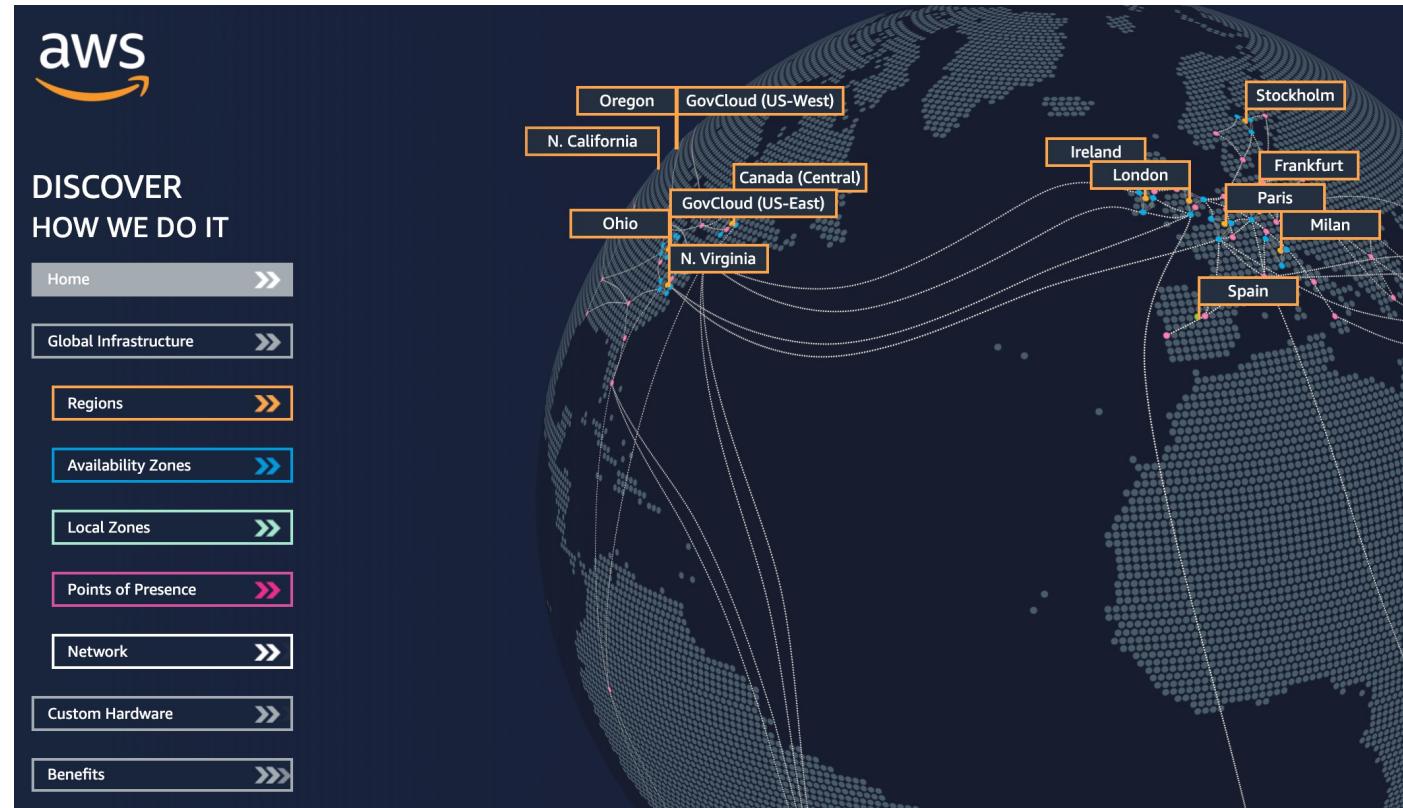
21ST  
CENTURY  
FOX

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?

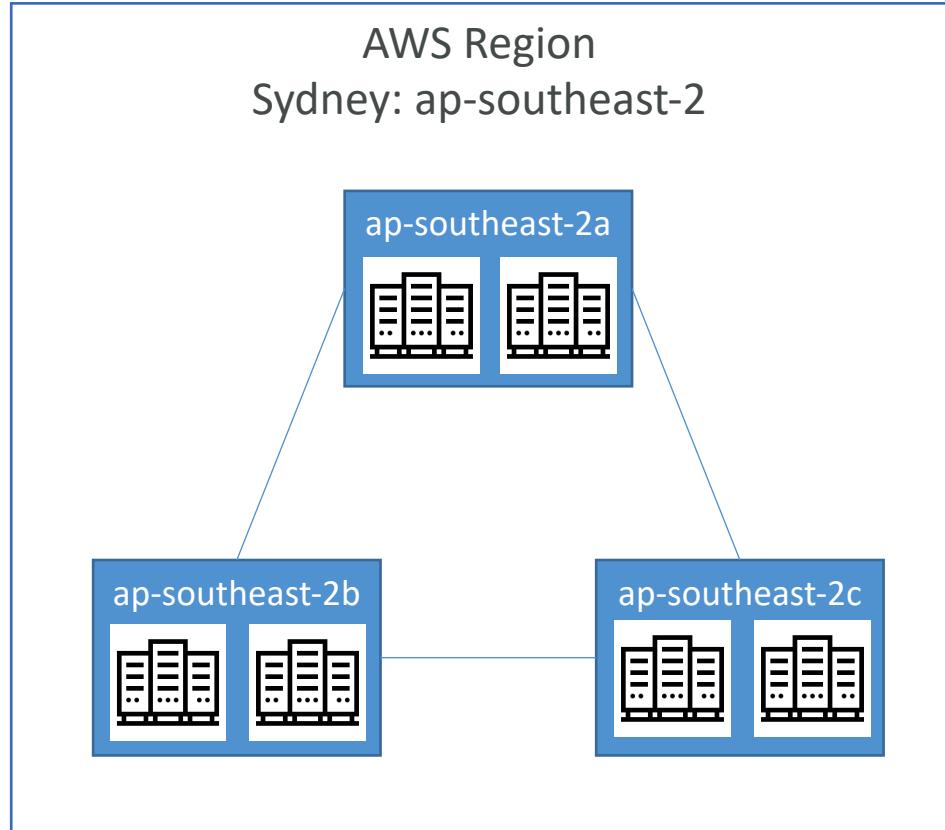
If you need to launch a new application,  
where should you do it?



- **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

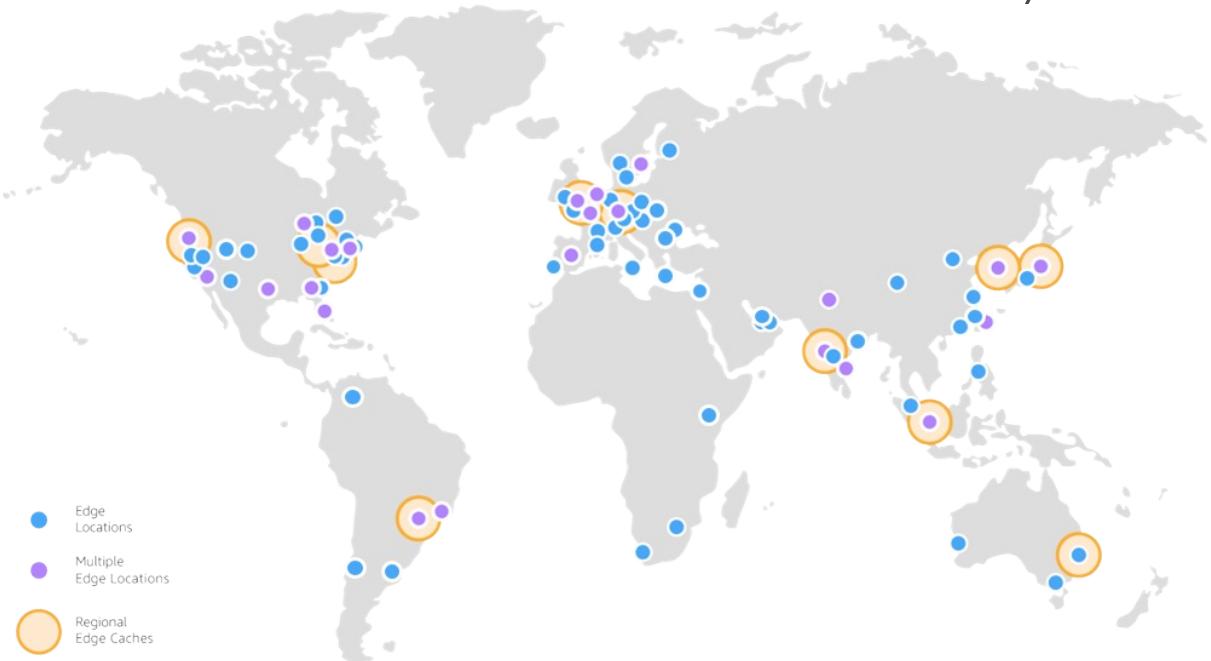
# AWS Availability Zones

- Each region has many availability zones (usually 3, min is 2, 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 216 Points of Presence (205 Edge Locations & 11 Regional Caches) in 84 cities across 42 countries
- Content is delivered to end users with lower latency

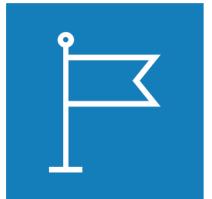


<https://aws.amazon.com/cloudfront/features/>

# 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>

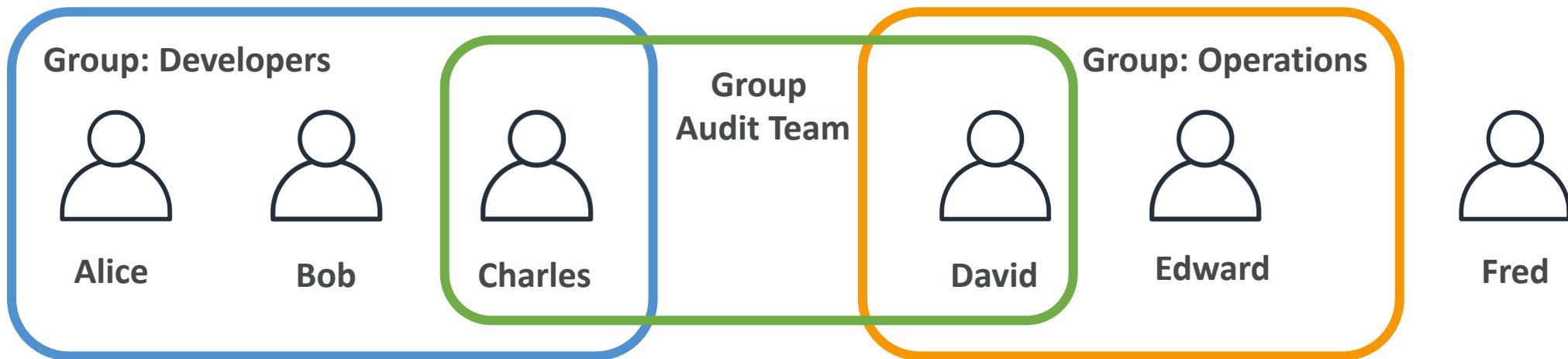


# IAM Section

# 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



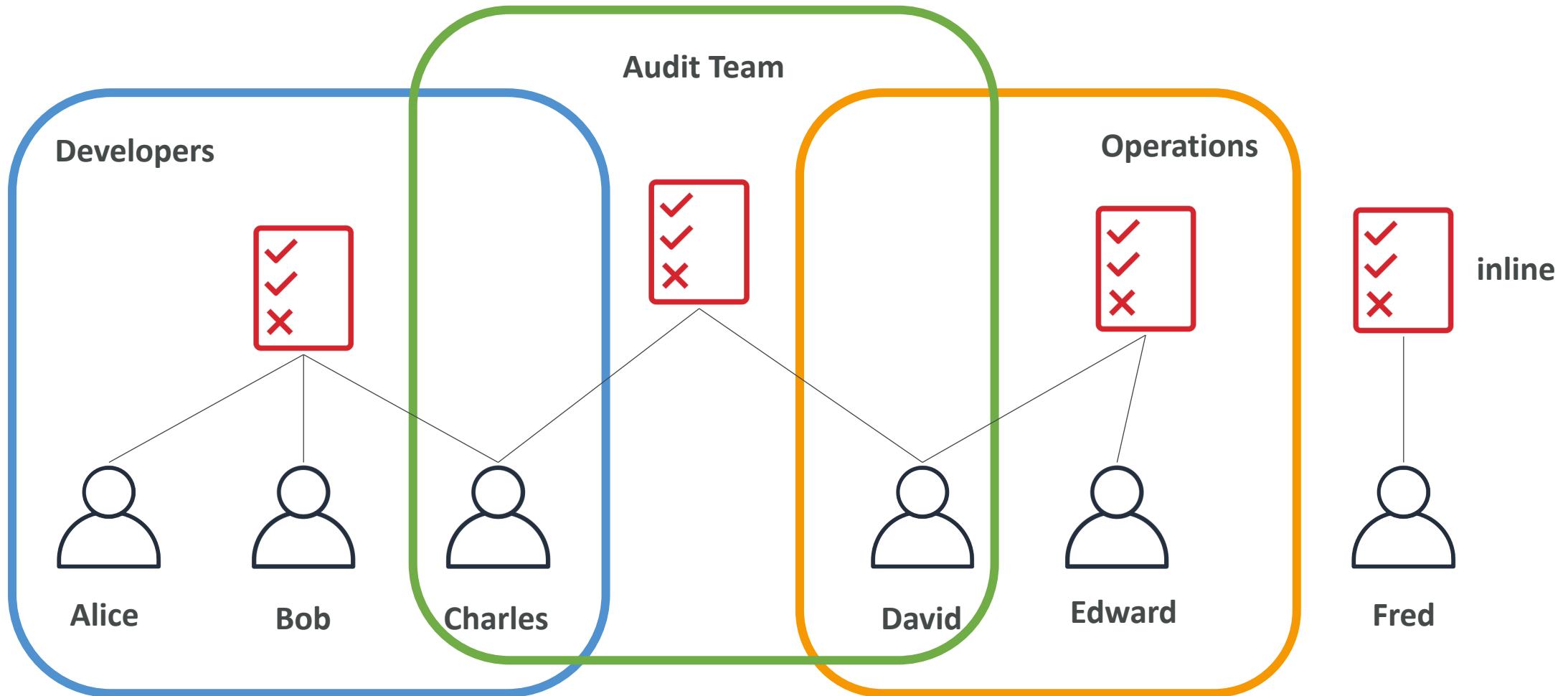
# IAM: Permissions

- 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 (3<sup>rd</sup> party)

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

# MFA devices options in AWS

## Hardware Key Fob MFA Device



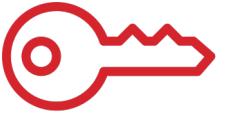
Provided by Gemalto (3<sup>rd</sup> party)

## Hardware Key Fob MFA Device for AWS GovCloud (US)



Provided by SurePassID (3<sup>rd</sup> 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	<a href="#">Make inactive</a> <a href="#">X</a>

- 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?



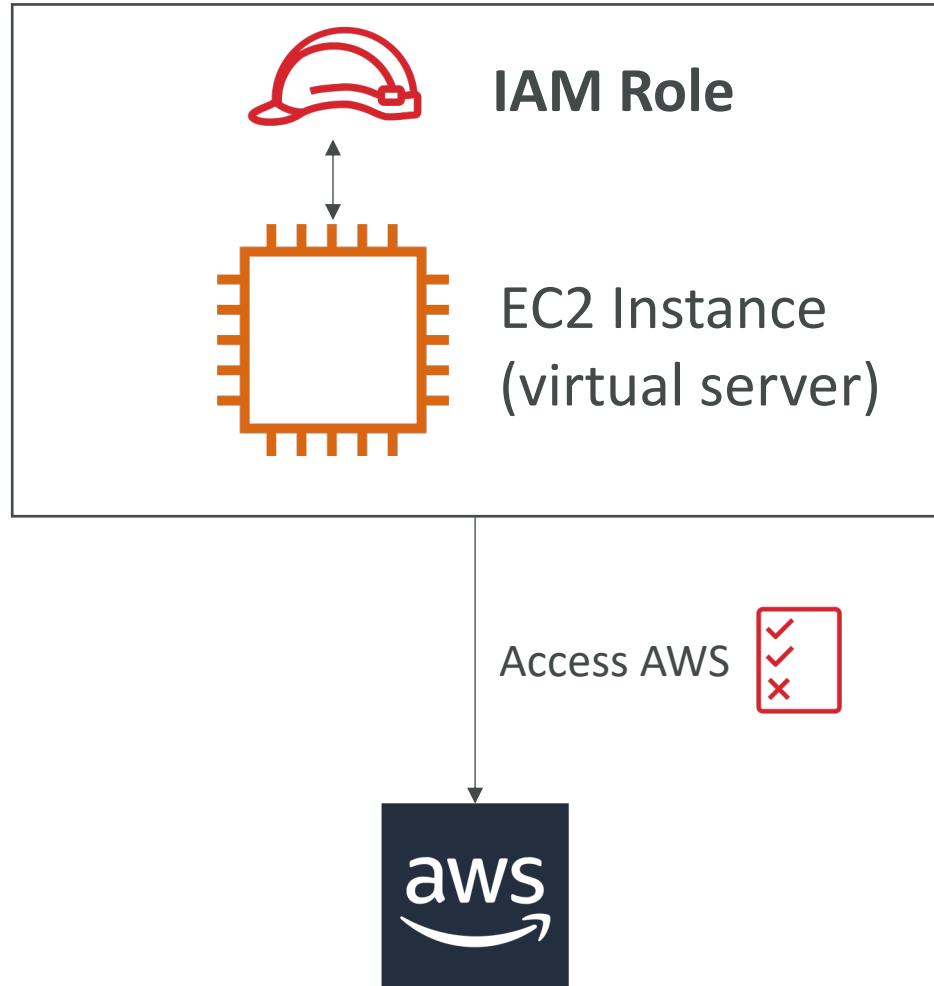
- AWS Software Development Kit (AWS SDK)
- Language-specific APIs (set of libraries)
- Enables you to access and manage AWS services programmatically
- Embedded within your application
- Supports
  - SDKs (JavaScript, Python, PHP, .NET, Ruby, Java, Go, Node.js, C++)
  - Mobile SDKs (Android, iOS, ...)
  - 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 with the IAM Credentials Report
- Never share IAM users & Access Keys

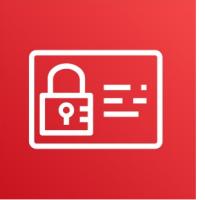
# Shared Responsibility Model for IAM



You

- Infrastructure (global network security)
- Configuration and vulnerability analysis
- Compliance validation
- Users, Groups, Roles, Policies management and monitoring
- Enable MFA on all accounts
- Rotate all your keys often
- Use IAM tools to apply appropriate permissions
- Analyze access patterns & review permissions

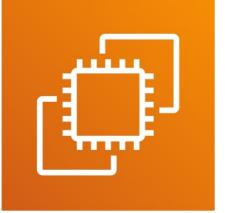
# IAM Section – Summary



- **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
- **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

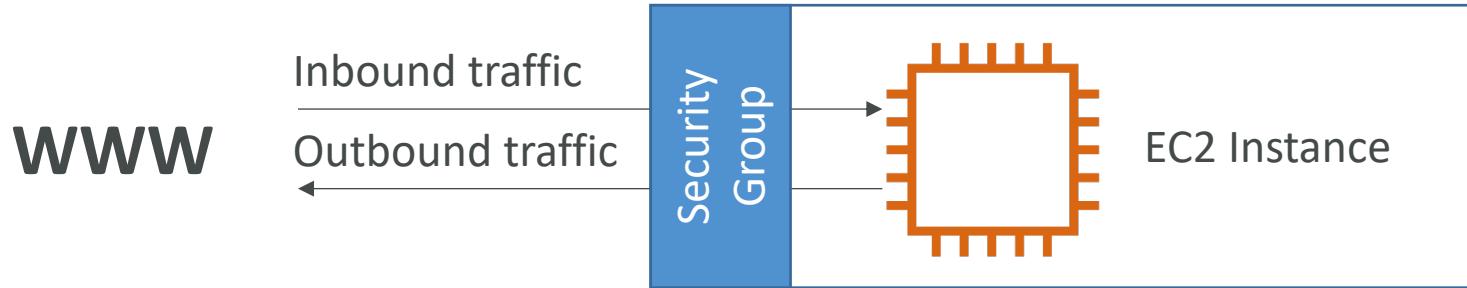
Instance	vCPU	Mem (GiB)	Storage	Network Performance	EBS Bandwidth (Mbps)
t2.micro	1	1	EBS-Only	Low to Moderate	
t2.xlarge	4	16	EBS-Only	Moderate	
c5d.4xlarge	16	32	1 x 400 NVMe SSD	Up to 10 Gbps	4,750
r5.16xlarge	64	512	EBS Only	20 Gbps	13,600
m5.8xlarge	32	128	EBS Only	10 Gbps	6,800

**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

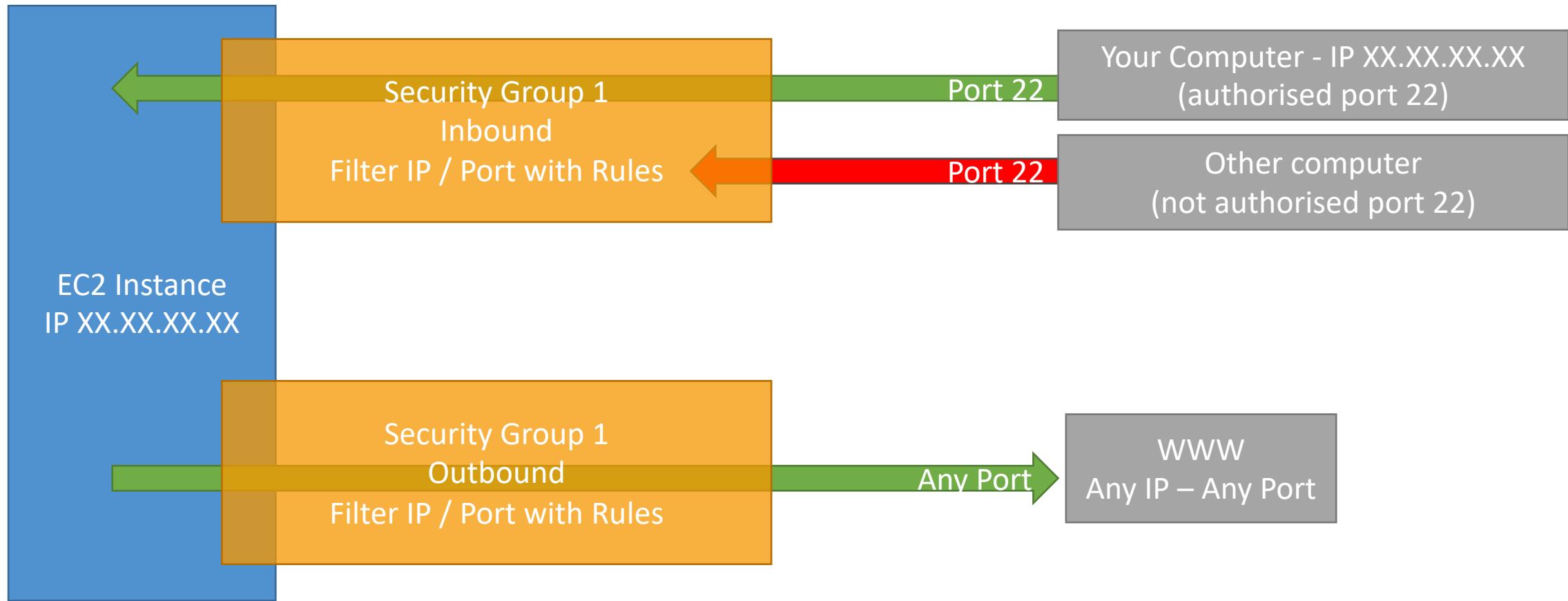
# Security Groups

## 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 <span>i</span>	Protocol <span>i</span>	Port Range <span>i</span>	Source <span>i</span>	Description <span>i</span>
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



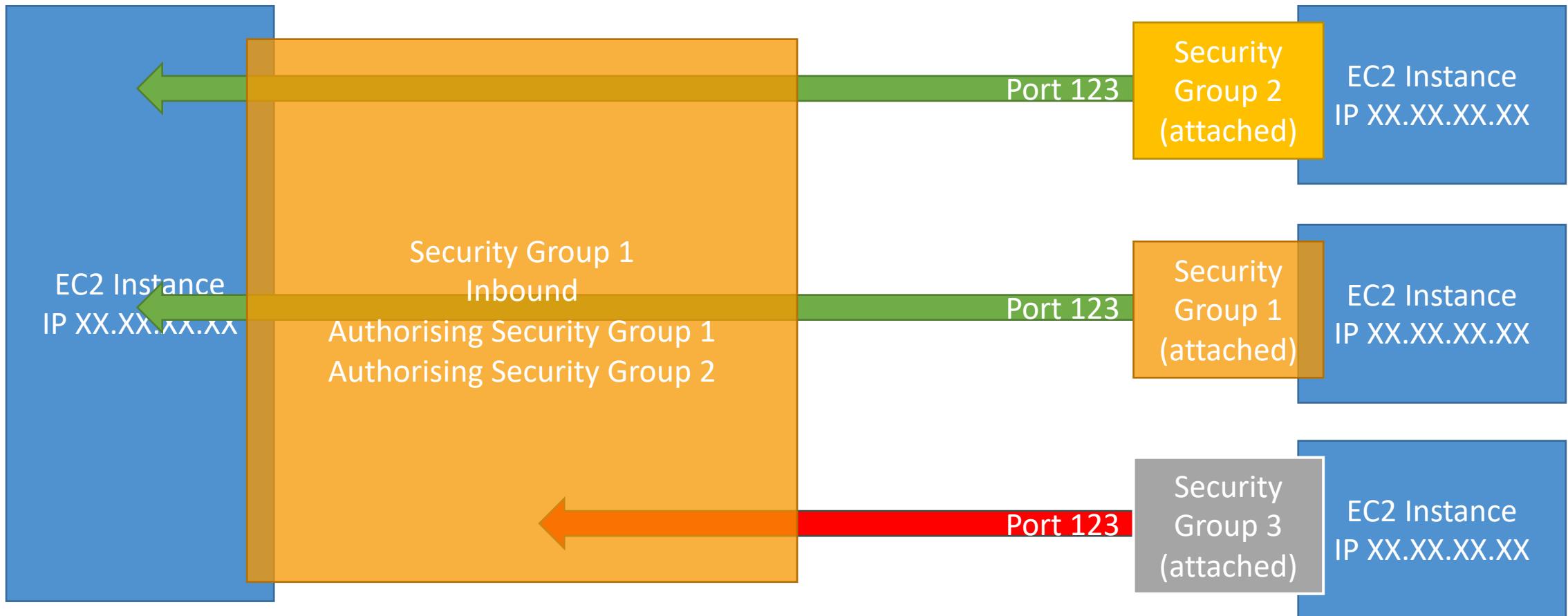
# Security Groups

## 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

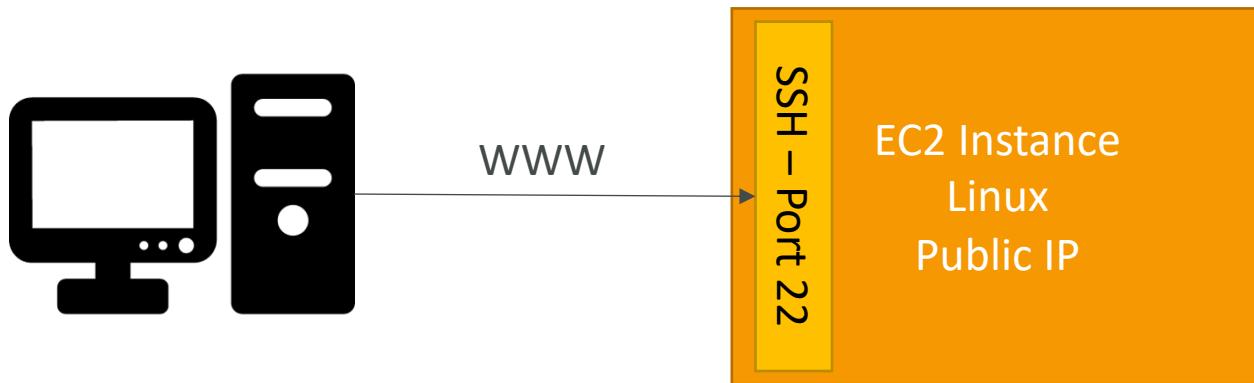
# SSH troubleshooting

- 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.

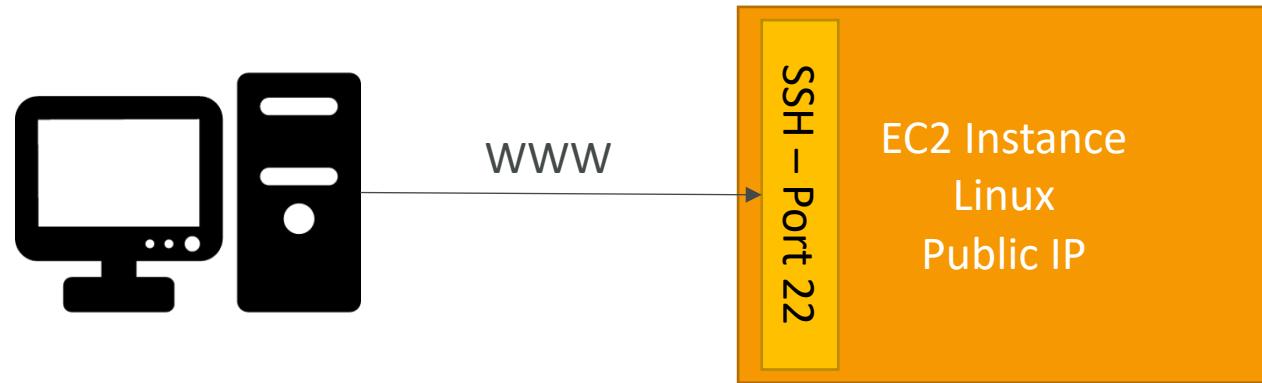


- We will see how we can configure OpenSSH `~/.ssh/config` to facilitate the SSH into our EC2 instances

# 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 <b>Convertible</b> 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

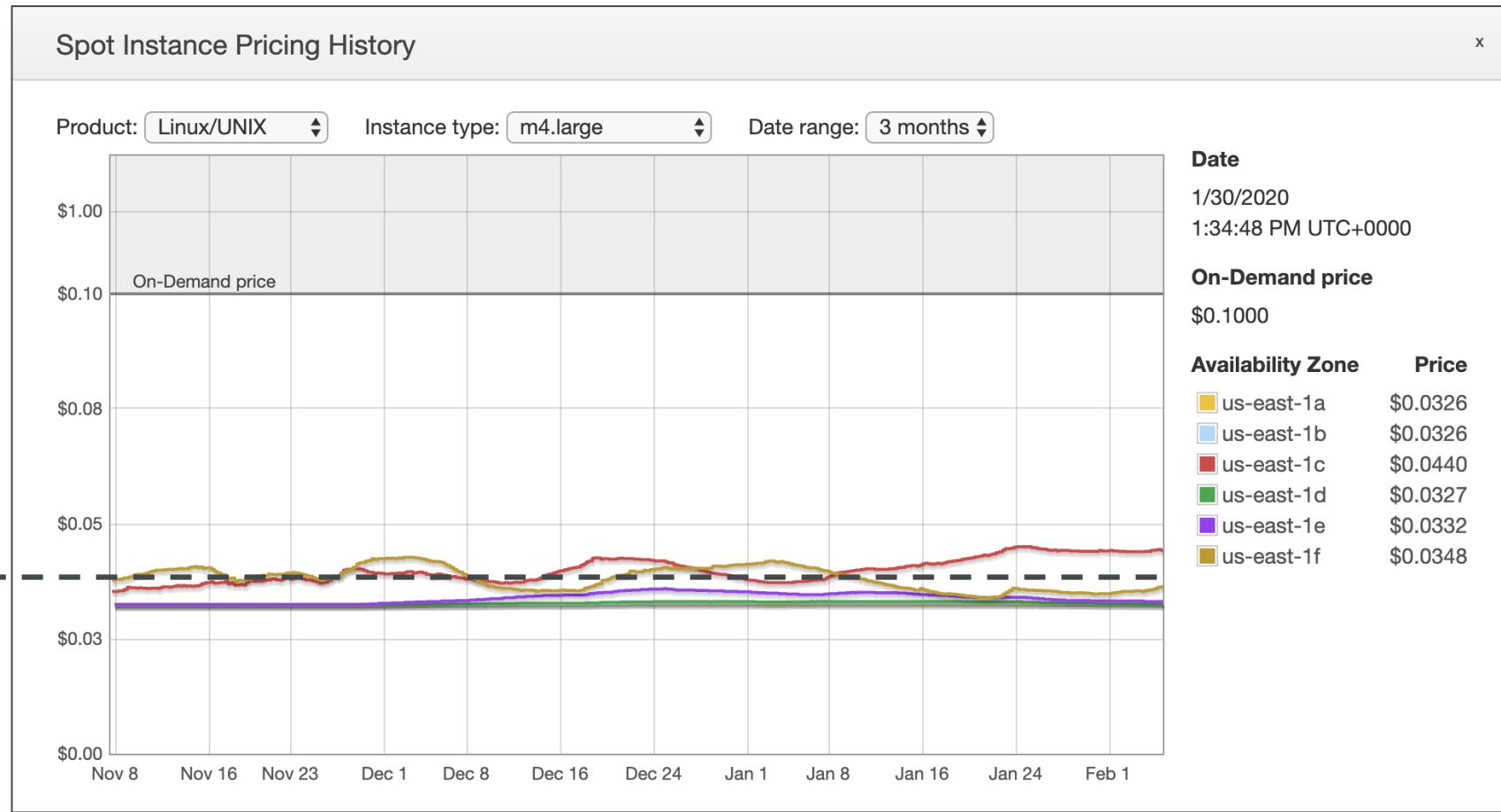


# EC2 Spot Instance Requests

- 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

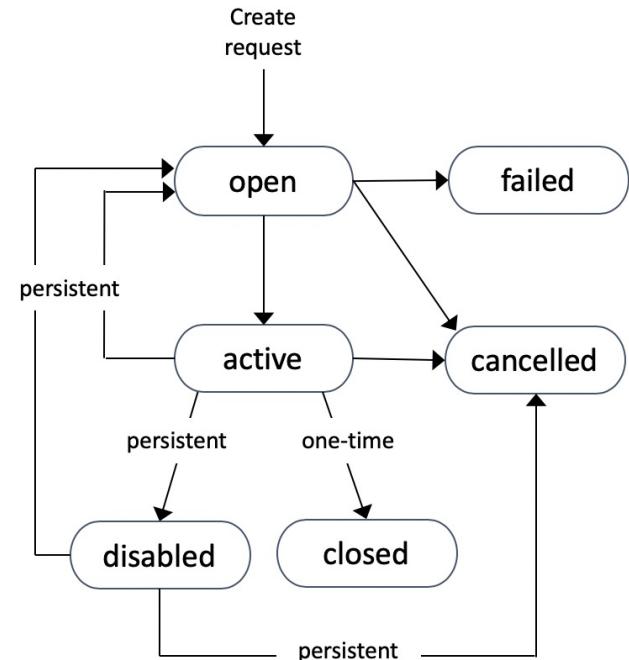
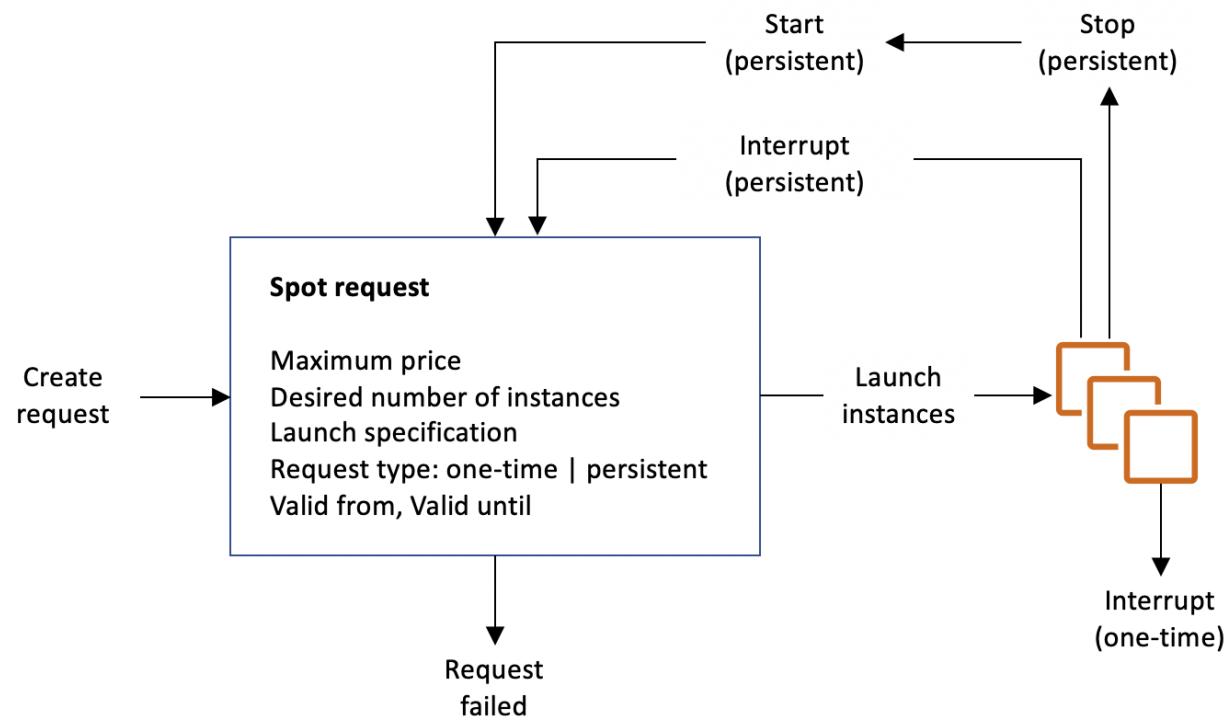
# EC2 Spot Instances Pricing

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**.

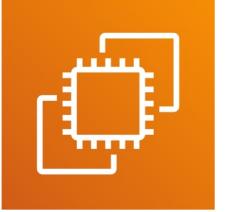
Cancelling a Spot Request does not terminate instances

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
- Spot Fleets allow us to automatically request Spot Instances with the lowest price

# EC2 Section – Summary



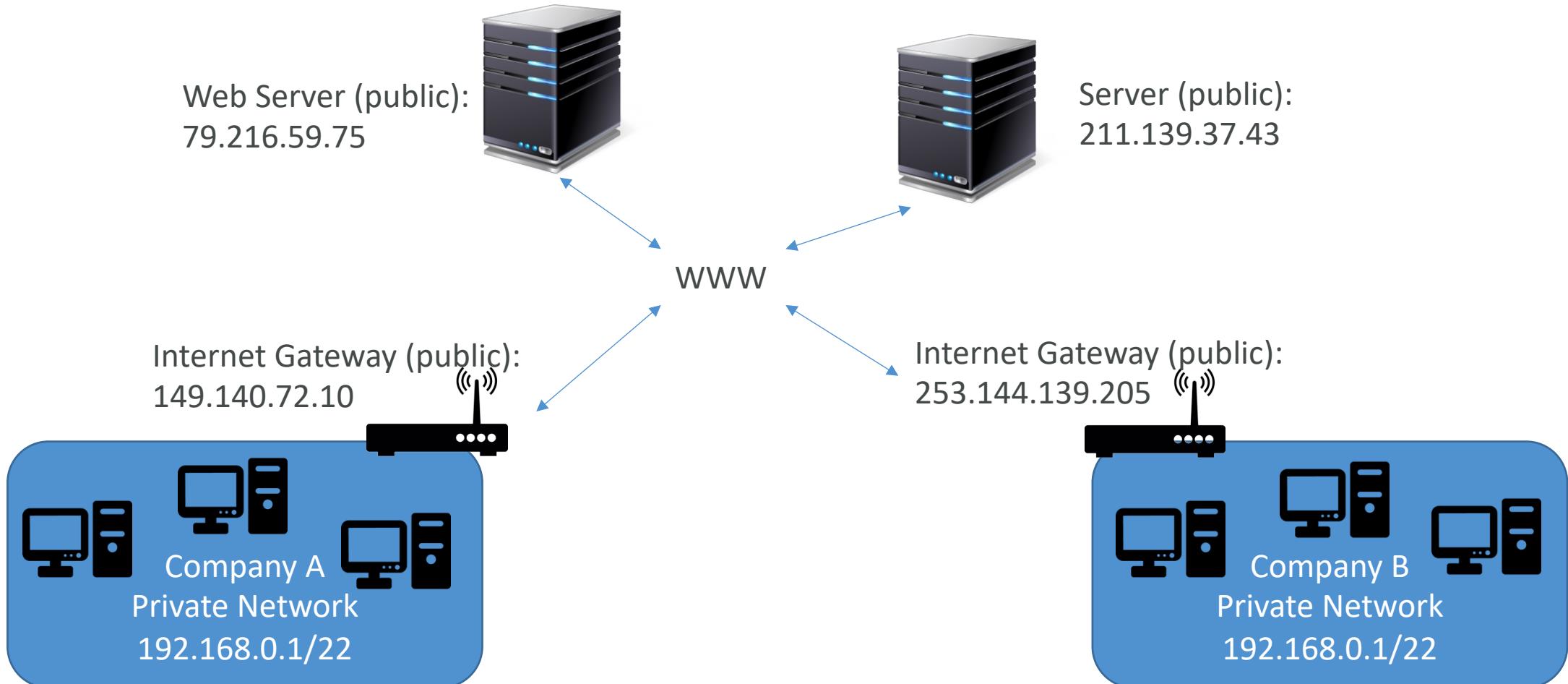
- **EC2 Instance:** AMI (OS) + Instance Size (CPU + RAM) + Storage + security groups + EC2 User Data
- **Security Groups:** Firewall attached to the EC2 instance
- **EC2 User Data:** Script launched at the first start of an instance
- **SSH:** start a terminal into our EC2 Instances (port 22)
- **EC2 Instance Role:** link to IAM roles
- **Purchasing Options:** On-Demand, Spot, Reserved (Standard + Convertible + Scheduled), Dedicated Host, Dedicated Instance

# 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)

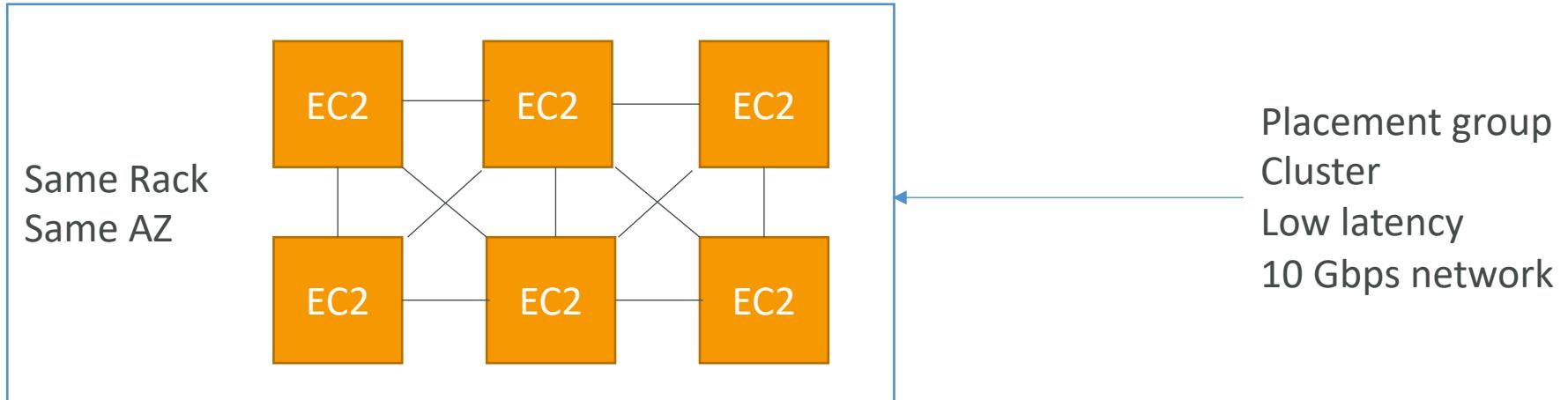
## In AWS EC2 – Hands On

- 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

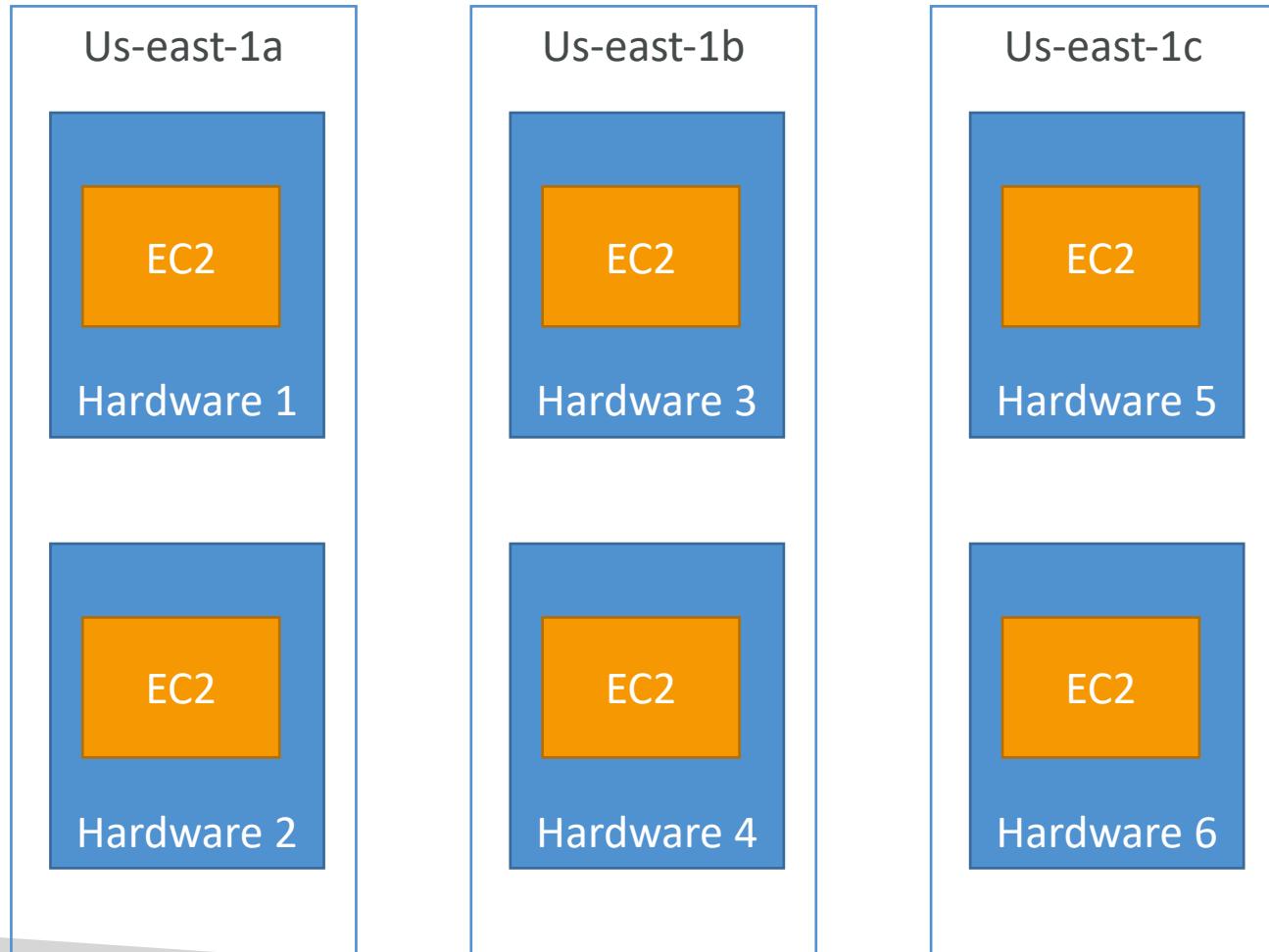
- Sometimes you want control over the EC2 Instance placement strategy
- That strategy can be defined using placement groups
- When you create a placement group, you specify one of the following strategies for the group:
  - *Cluster*—clusters instances into a low-latency group in a single Availability 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:

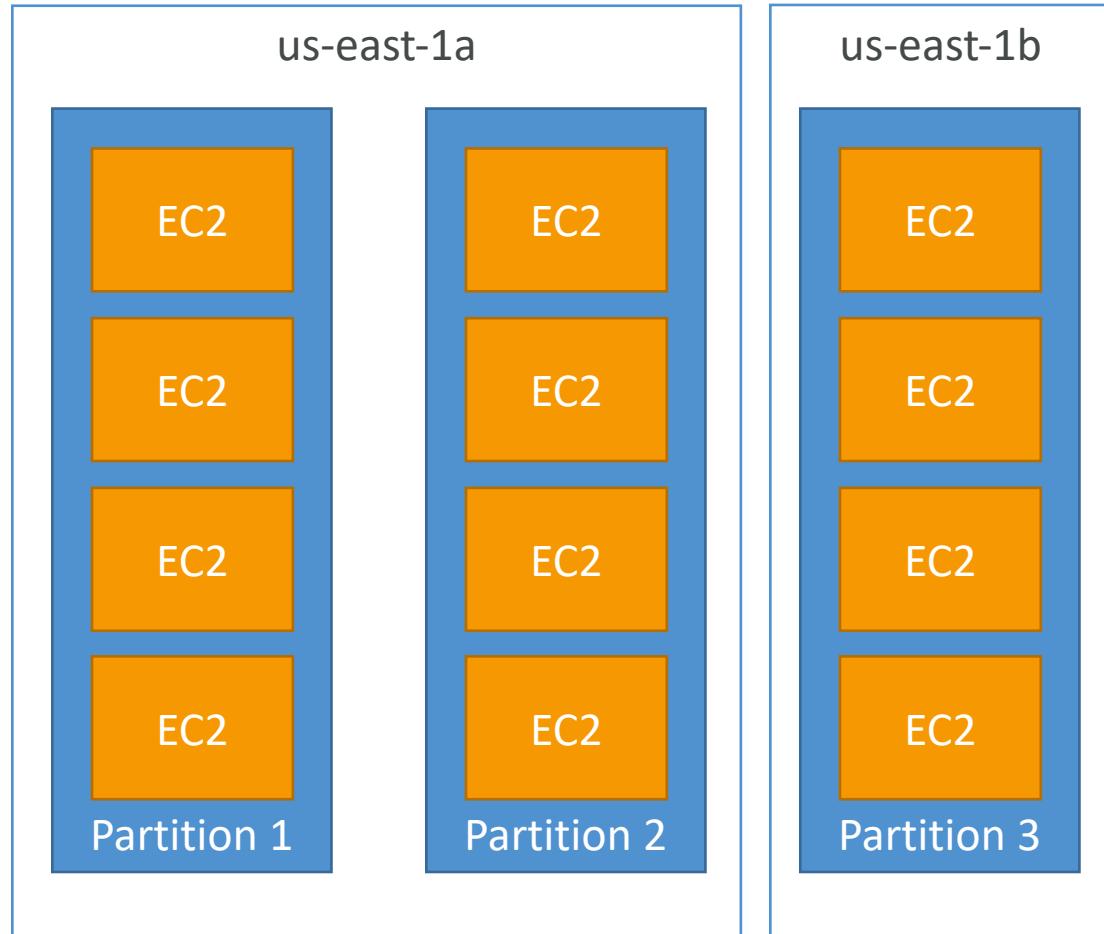
- Limited to 7 instances per AZ per placement group

- Use case:

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

# Placements Groups

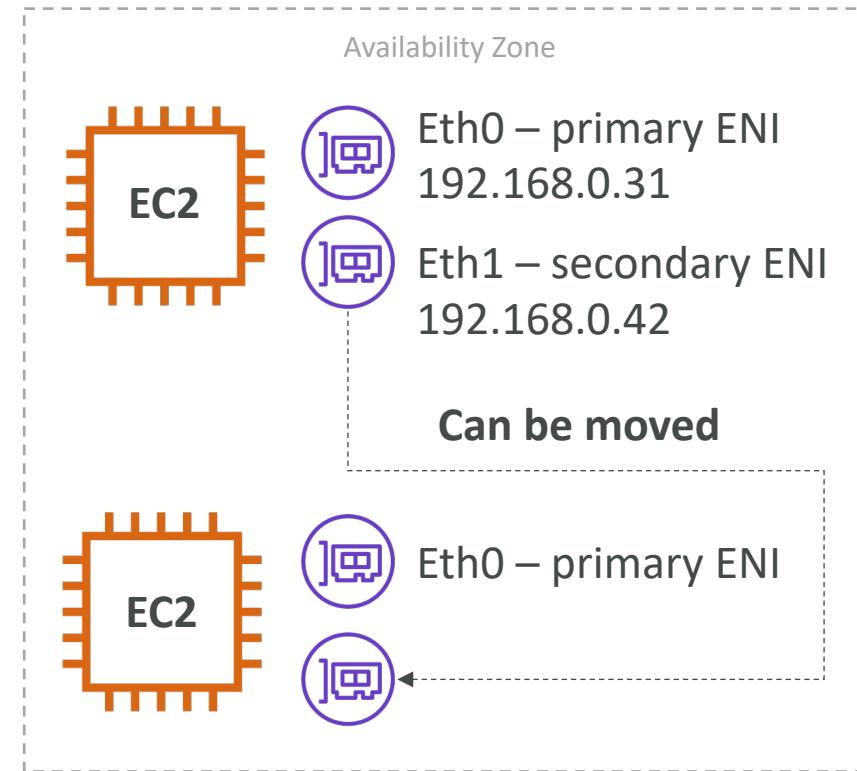
## Partition



- 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)

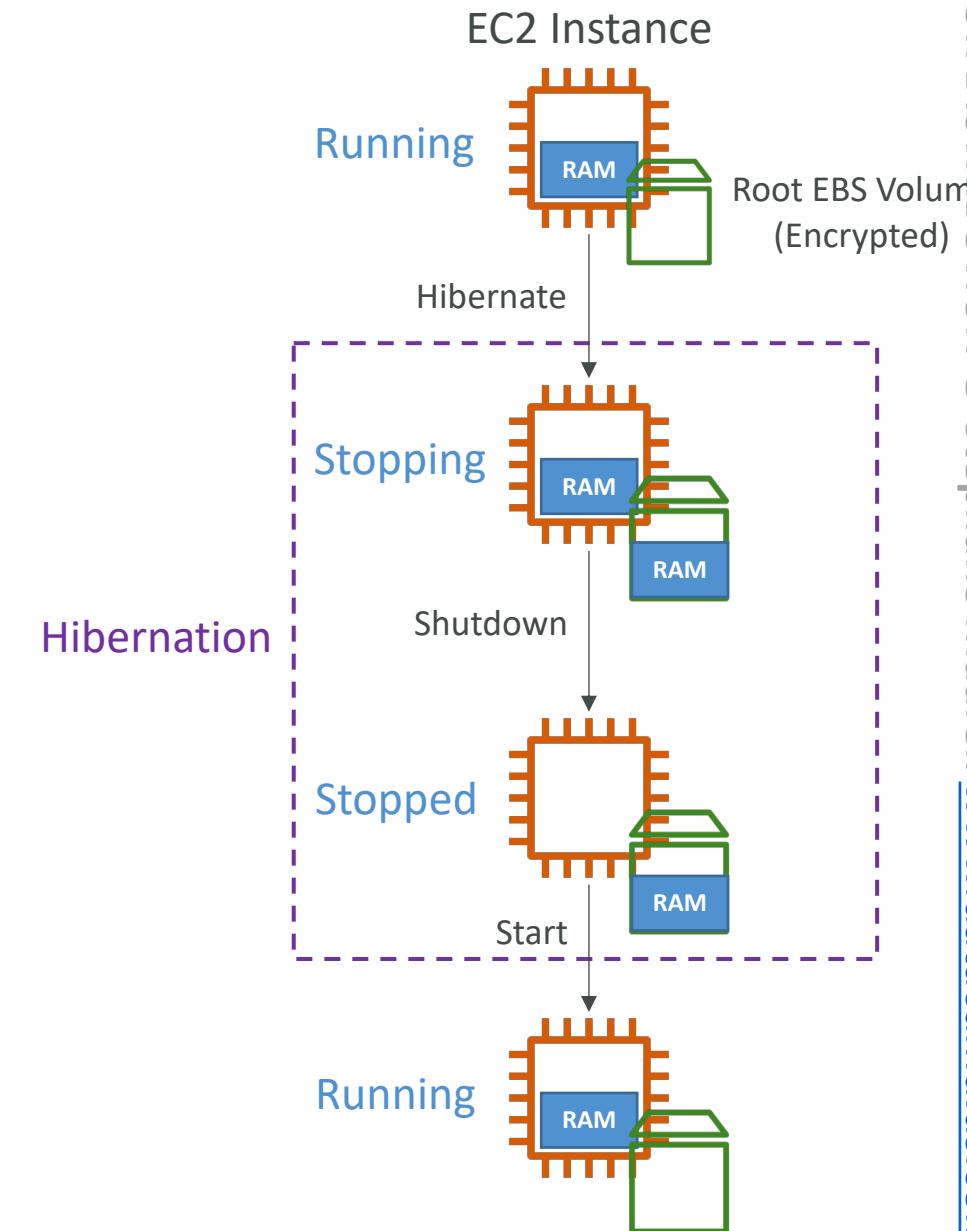


# EC2 Hibernate

- 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

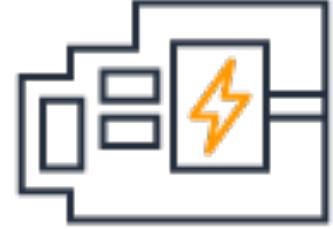
- Introducing EC2 Hibernate:
  - The in-memory (RAM) state is preserved
  - The instance boot is much faster! (the OS is not stopped / restarted)
  - Under the hood: the RAM state is written to a file in the root EBS volume
  - The root EBS volume must be encrypted
- Use cases:
  - Long-running processing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

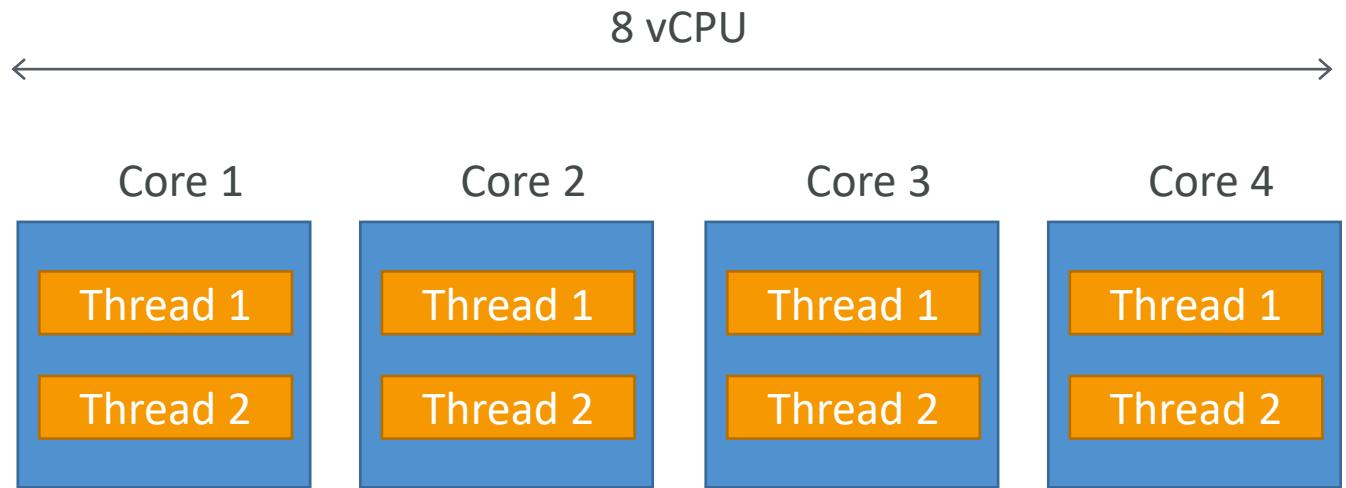
# EC2 Nitro



- Underlying Platform for the next generation of EC2 instances
- New virtualization technology
- Allows for better performance:
  - Better networking options (enhanced networking, HPC, IPv6)
  - **Higher Speed EBS** (Nitro is necessary for 64,000 EBS IOPS – max 32,000 on non-Nitro)
- Better underlying security
- Instance types example:
  - Virtualized: `A1`, `C5`, `C5a`, `C5ad`, `C5d`, `C5n`, `C6g`, `C6gd`, `C6gn`, `D3`, `D3en`, `G4`, `I3en`, `Infl`, `M5`, `M5a`, `M5ad`, `M5d`, `M5dn`, `M5n`, ....
  - Bare metal: `a1.metal`, `c5.metal`, `c5d.metal`, `c5n.metal`, `c6g.metal`, `c6gd.metal`...

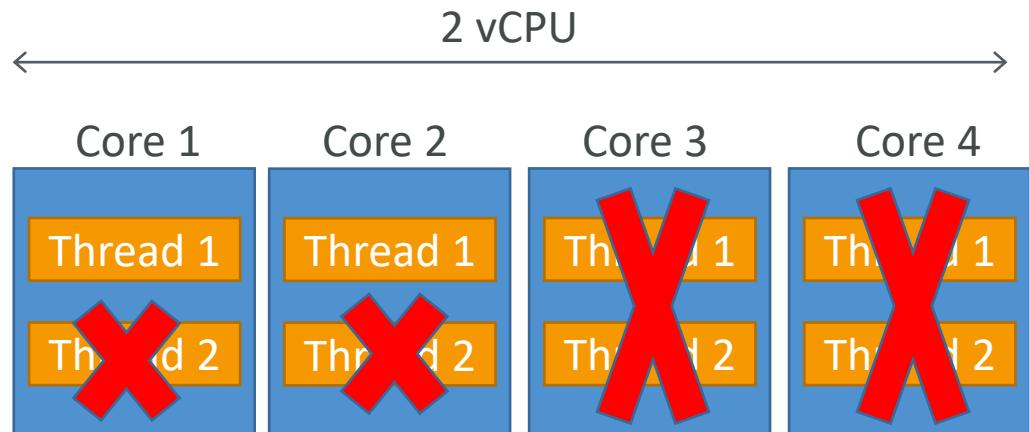
# EC2 – Understanding vCPU

- Multiple threads can run on one CPU (multithreading)
- Each thread is represented as a virtual CPU (vCPU)
- Example: m5.2xlarge
  - 4 CPU
  - 2 threads per CPU
  - => 8 vCPU in total



# EC2 – Optimizing CPU options

- EC2 instances come with a combination of RAM and vCPU
- But in some cases, you may want to change the vCPU options:
  - **# of CPU cores:** you can decrease it (helpful if you need high RAM and low number of CPU) – to decrease licensing costs
  - **# of threads per core:** disable multithreading to have 1 thread per CPU – helpful for high performance computing (HPC) workloads
- Only specified during instance launch



Instance type	Default vCPUs	Default CPU cores	Default threads per core	Valid CPU cores	Valid threads per core
r4.2xlarge	8	4	2	1, 2, 3, 4	1, 2

CPU options i  Specify CPU options

Core count	4
Threads per core	2
Number of vCPUs	8

# EC2 – Capacity Reservations

- Capacity Reservations ensure you have EC2 Capacity when needed
- Manual or planned end-date for the reservation
- No need for 1 or 3-year commitment
- Capacity access is immediate, you get billed as soon as it starts
- Specify:
  - The Availability Zone in which to reserve the capacity (only one)
  - The number of instances for which to reserve capacity
  - The instance attributes, including the instance type, tenancy, and platform/OS
- Combine with Reserved Instances and Savings Plans to do cost saving

## Instance Type

m5.xlarge

### EBS-optimized

If the selected instance type is EBS-optimized by default, you will not be able to uncheck this option.

### Instance store

Temporary block-level storage. Data persists only during the life of the instance.

## Platform

Linux/UNIX

## Availability Zone

us-east-1a

## Tenancy

Default - run a shared hardware instance

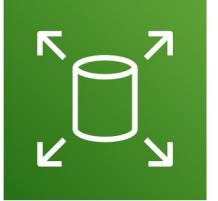
## Quantity

1

Up to X instances, per your current limits

# EC2 Instance Storage Section

# What's an EBS Volume?

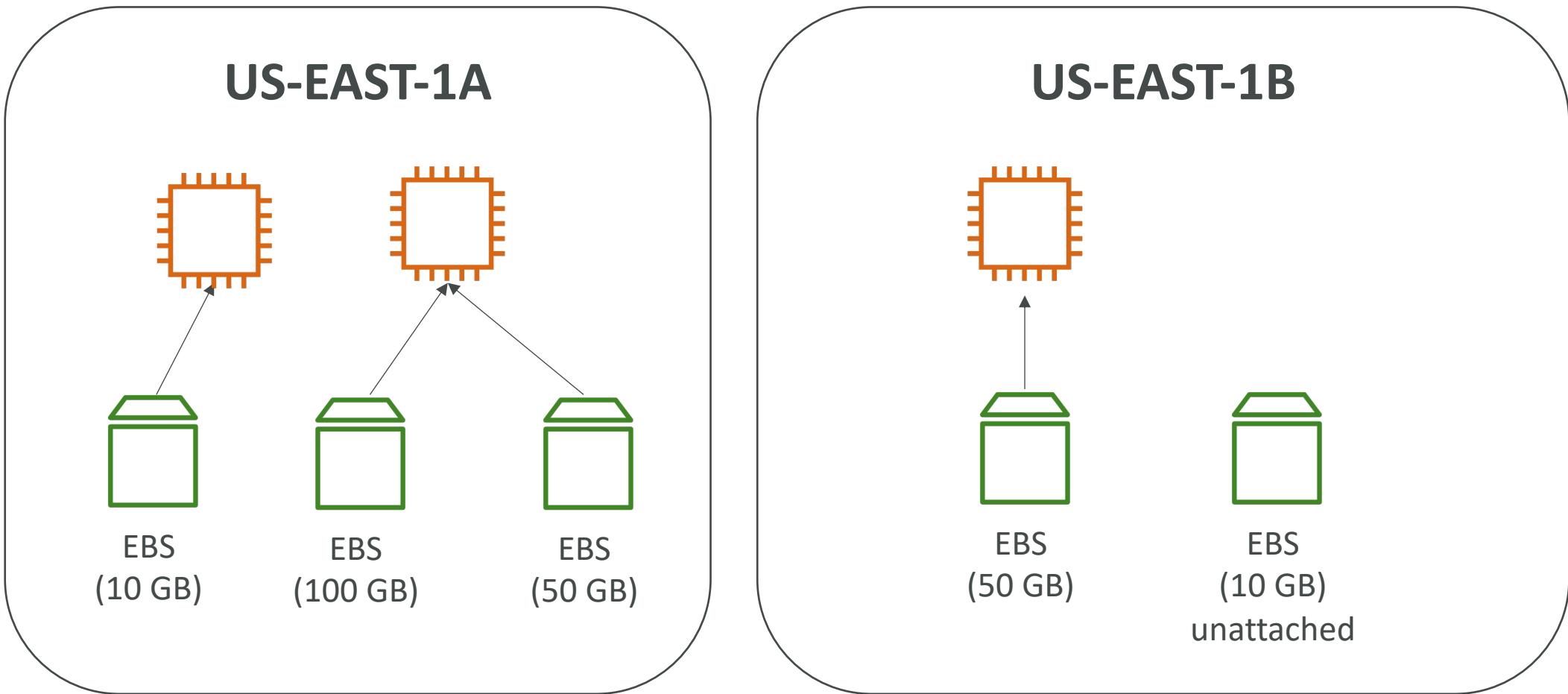


- An **EBS (Elastic Block Store) Volume** is a **network** drive you can attach to your instances while they run
- It allows your instances to persist data, even after their termination
- They can only be mounted to one instance at a time (at the CCP level)
- They are bound to a specific availability 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



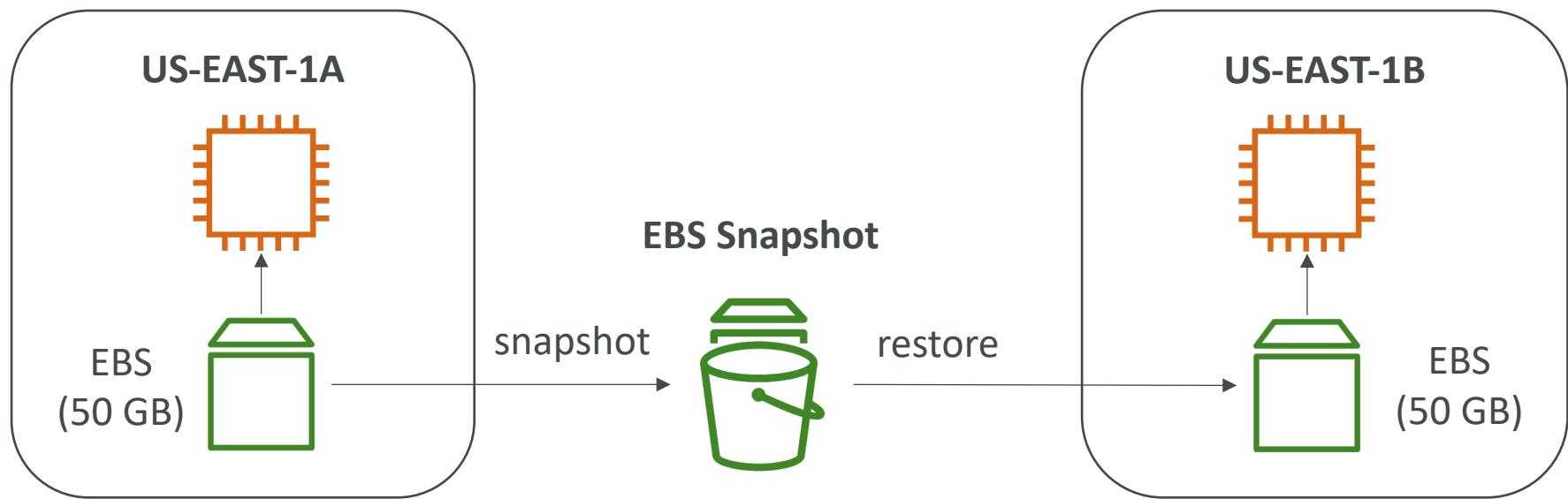
# EBS – Delete on Termination attribute

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 ▾ <span style="color:red;">X</span>
<a href="#">Add New Volume</a>								

- 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

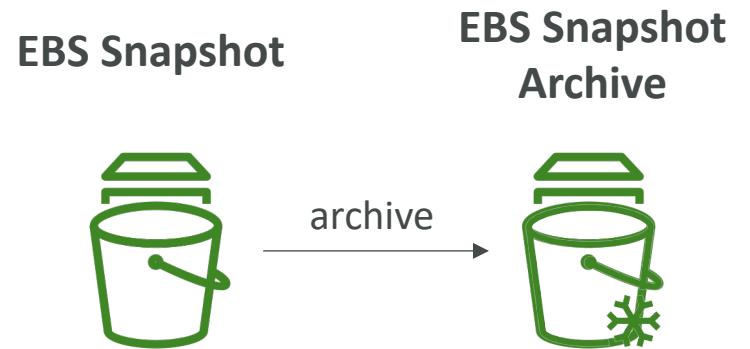
- Make a backup (snapshot) of your EBS volume at a point in time
- Not necessary to detach volume to do snapshot, but recommended
- Can copy snapshots across AZ or Region



# EBS Snapshots Features

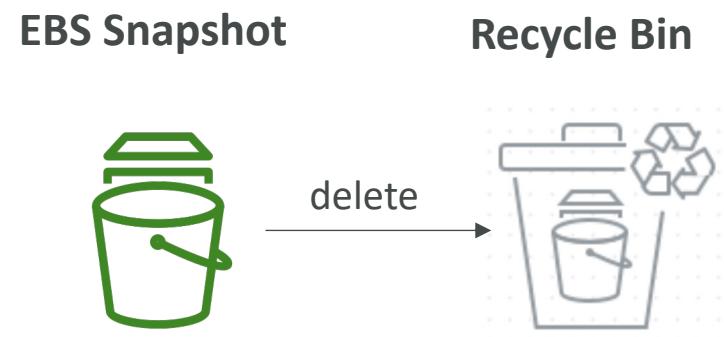
- **EBS Snapshot Archive**

- 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)



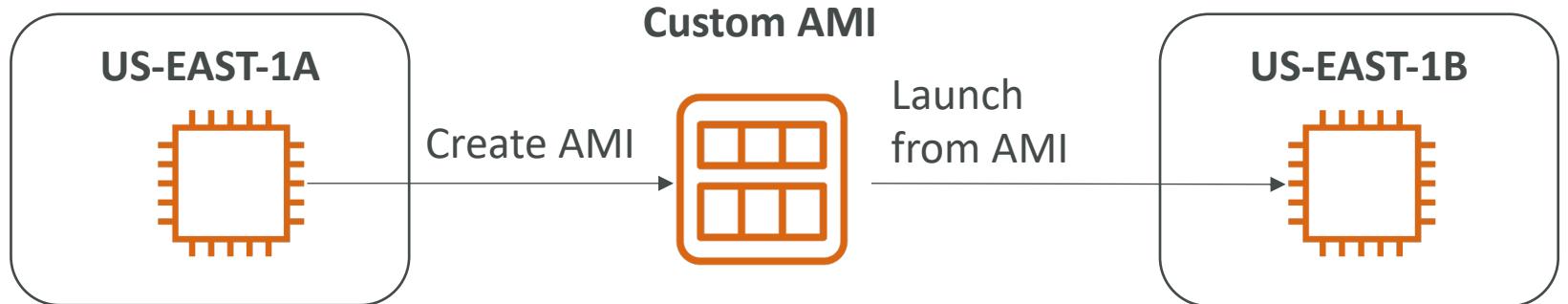


# AMI Overview

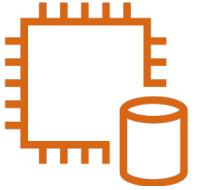
- AMI = Amazon Machine Image
- AMI are a **customization** of an EC2 instance
  - You add your own software, configuration, operating system, monitoring...
  - Faster boot / configuration time because all your software is pre-packaged
- AMI are built for a **specific region** (and can be copied across 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

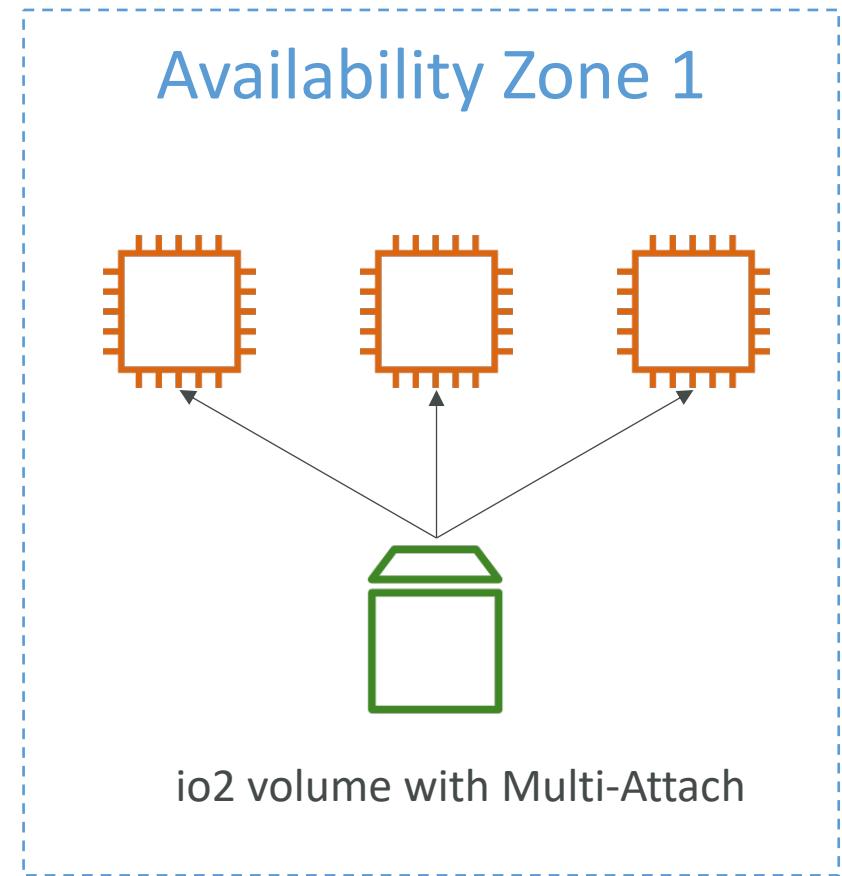
	General Purpose SSD		Provisioned IOPS SSD		
Volume type	gp3	gp2	io2 Block Express ‡	io2	io1
Durability	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.999% durability (0.001% annual failure rate)	99.8% - 99.9% durability (0.1% - 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"> <li>Low-latency interactive apps</li> <li>Development and test environments</li> </ul>	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"> <li>Workloads that require sustained IOPS performance or more than 16,000 IOPS</li> <li>I/O-intensive database workloads</li> </ul>		
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"> <li>Big data</li> <li>Data warehouses</li> <li>Log processing</li> </ul>	<ul style="list-style-type: none"> <li>Throughput-oriented storage for data that is infrequently accessed</li> <li>Scenarios where the lowest storage cost is important</li> </ul>
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 volume
- Use case:
  - Achieve higher application availability in clustered Linux applications (ex:Teradata)
  - Applications must manage concurrent write operations
- Must use a file system that's cluster-aware (not XFS, EX4, etc...)



# EBS Encryption

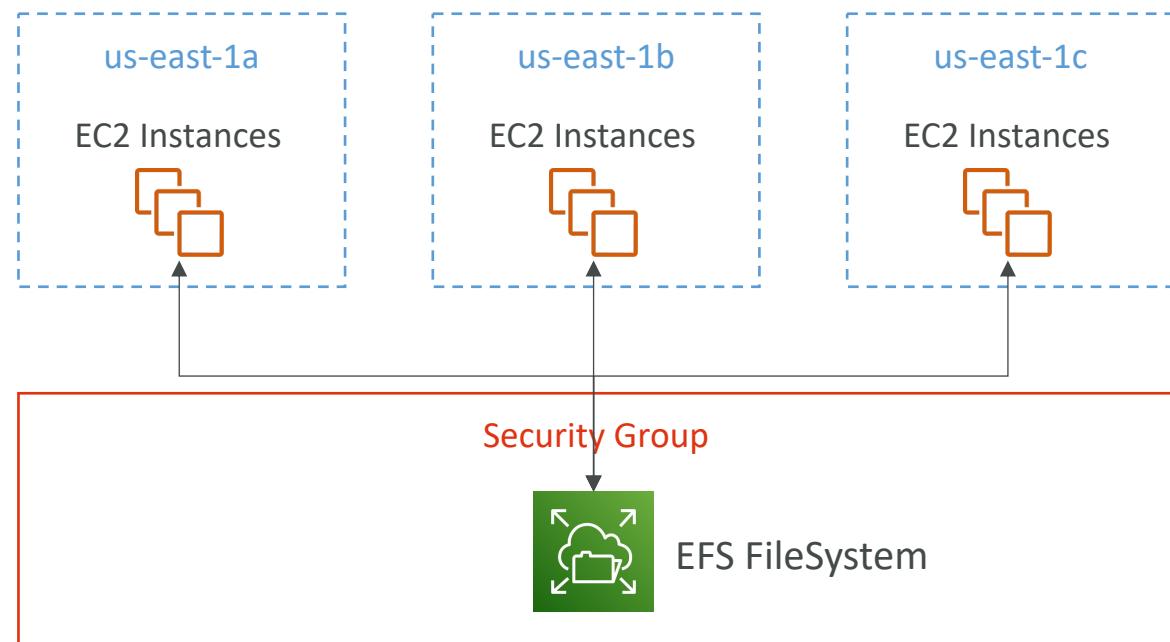
- When you create an encrypted EBS volume, you get the following:
  - Data at rest is encrypted inside the volume
  - All the data in flight moving between the instance and the volume is encrypted
  - All snapshots are encrypted
  - All volumes created from the snapshot
- Encryption and decryption are 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

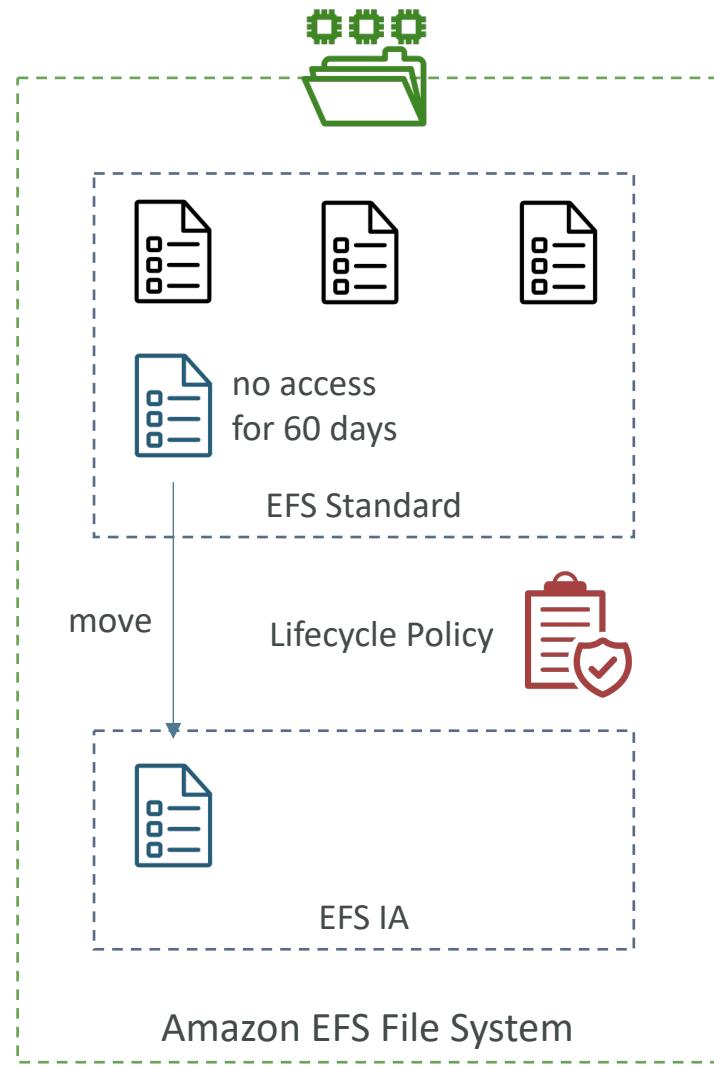
- Use cases: content management, web serving, data sharing, Wordpress
- Uses NFSv4.1 protocol
- Uses security group to control access to EFS
- **Compatible with Linux based AMI (not Windows)**
- Encryption at rest using KMS
  
- POSIX file system (~Linux) that has a standard file API
- File system scales 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

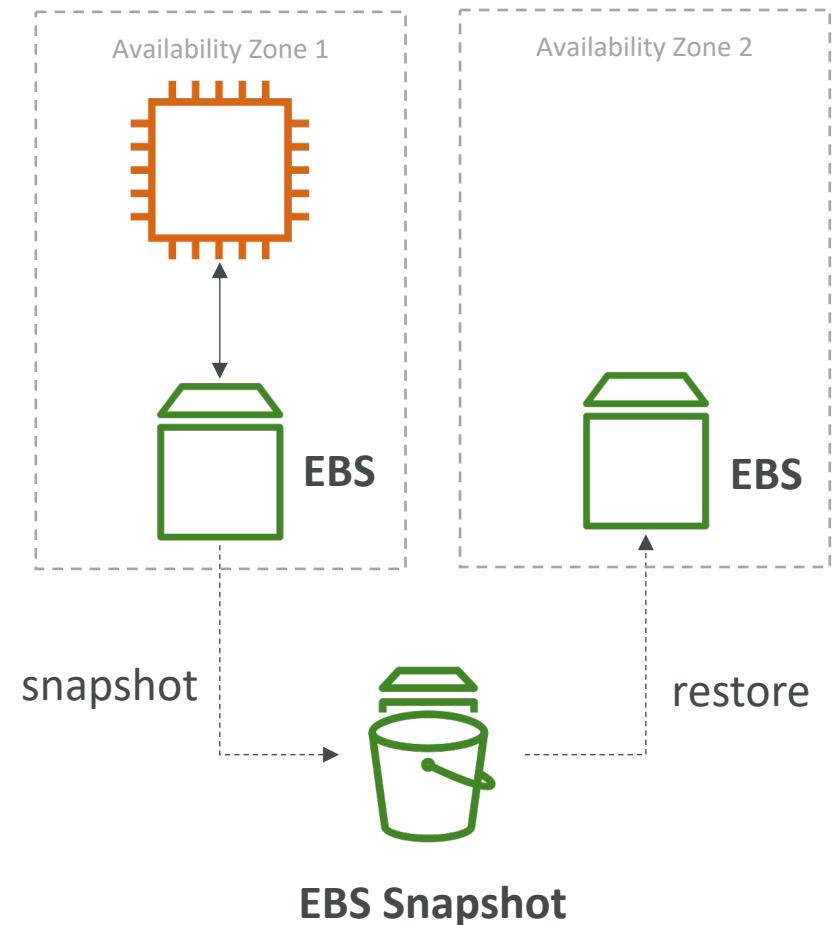
# EFS – Storage Classes

- 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
  - Regional: 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



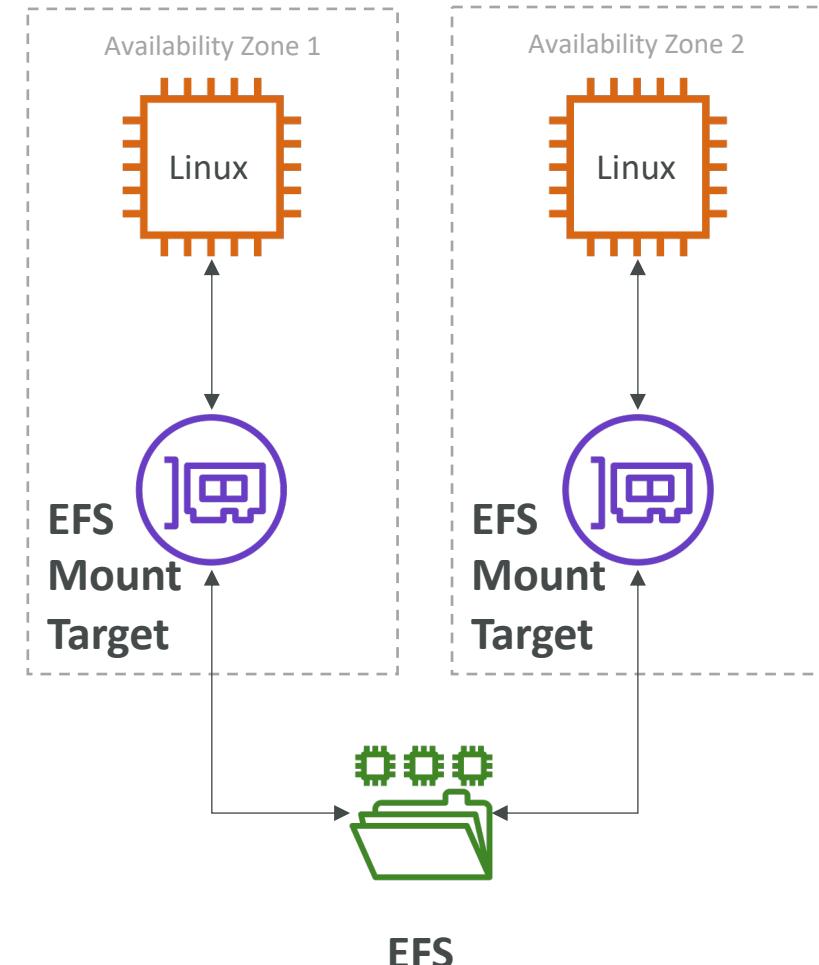
# EBS vs EFS – Elastic Block Storage

- EBS volumes...
  - can be attached to only one instance at a time
  - 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

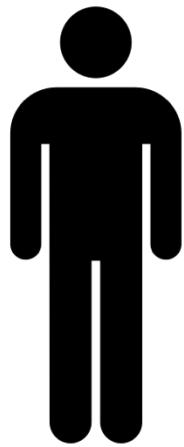
Load Balancing, Auto Scaling Groups and EBS Volumes

# Scalability & High Availability

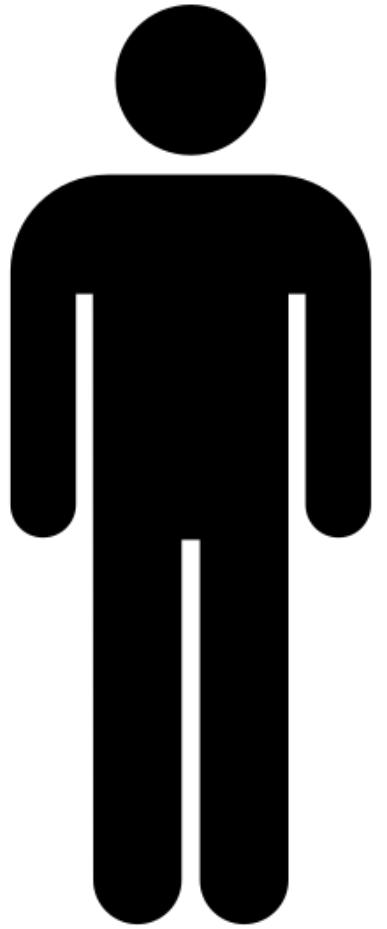
- Scalability means that an application / system can handle greater loads by adapting.
- There are two kinds of scalability:
  - Vertical Scalability
  - Horizontal Scalability (= elasticity)
- Scalability is linked but different to High Availability
- Let's deep dive into the distinction, using a call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

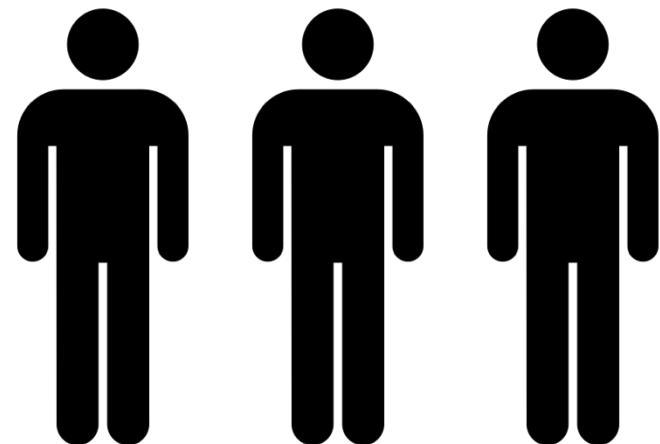
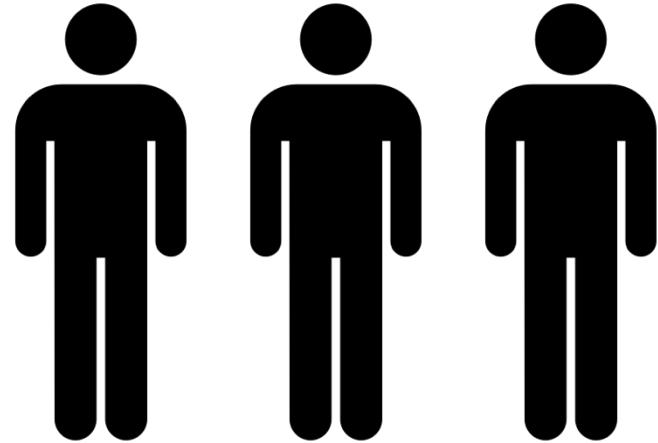


senior operator

# 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

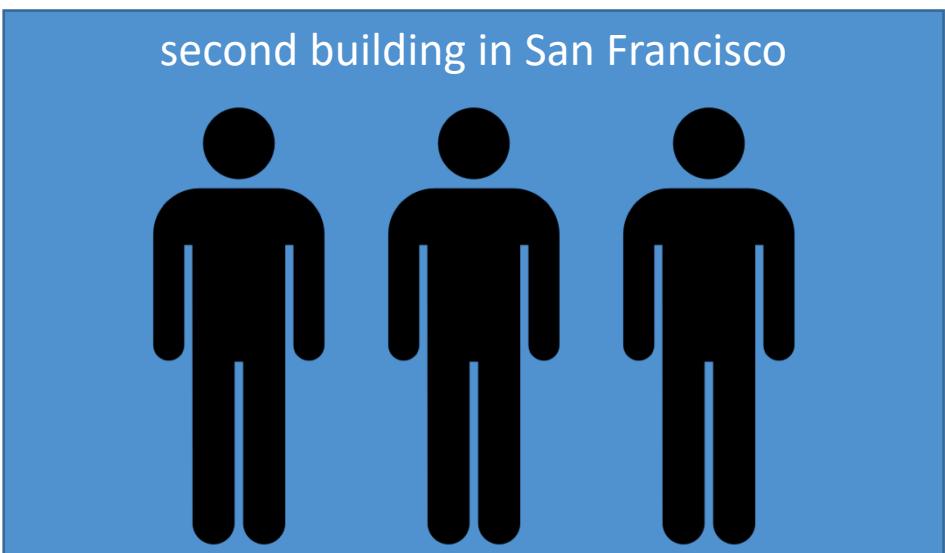
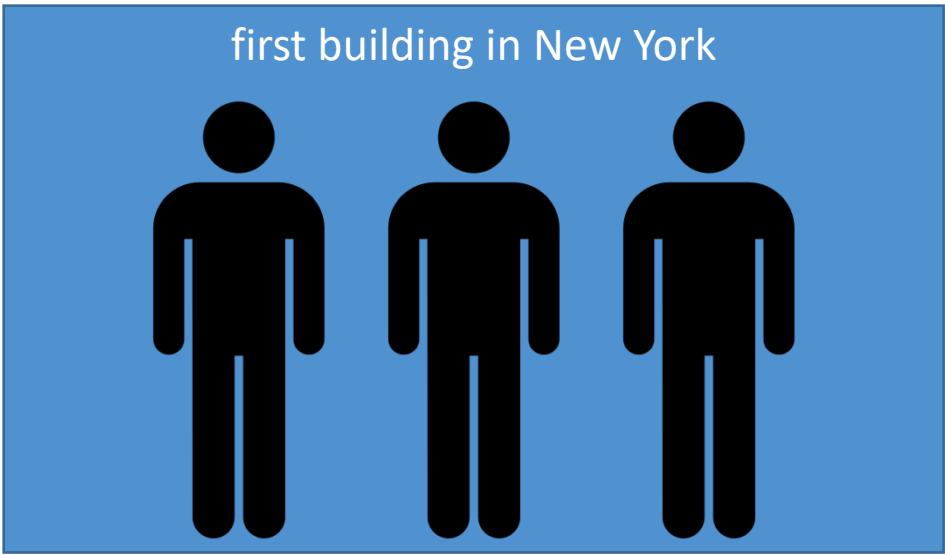
operator operator operator



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)

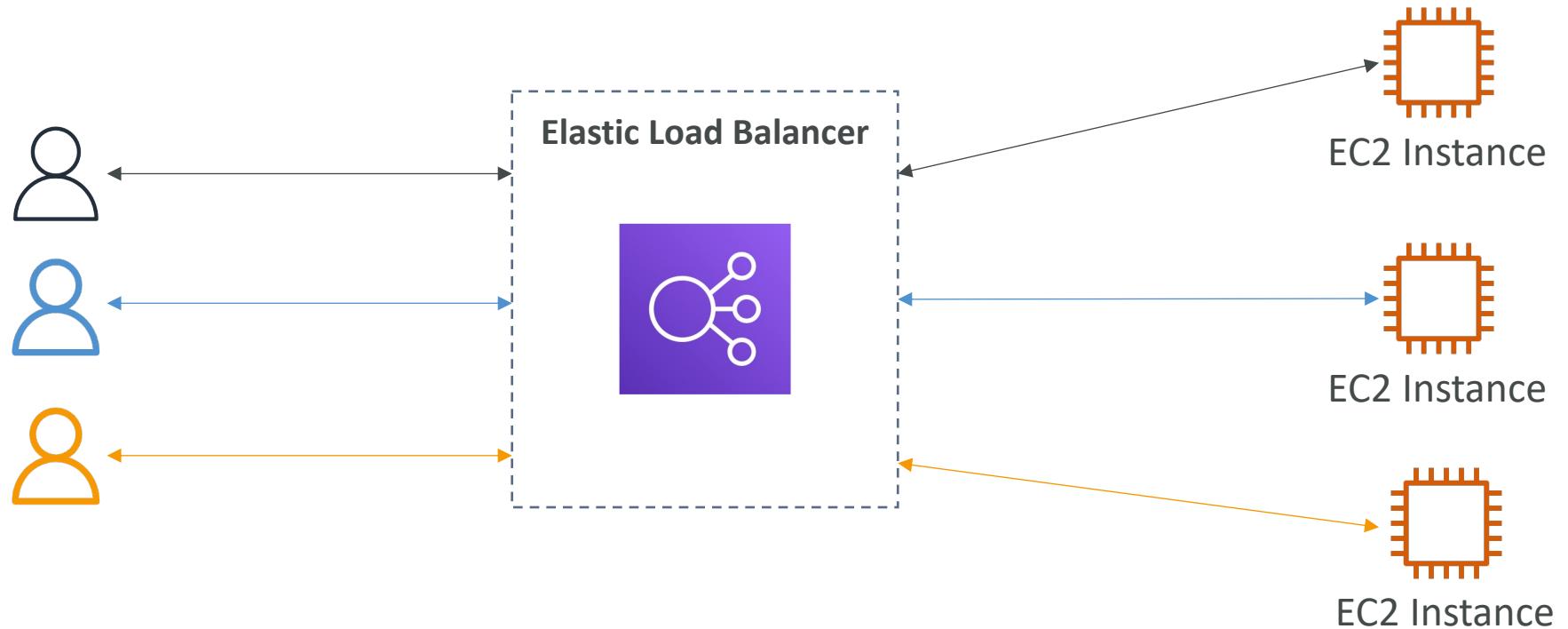


# High Availability & Scalability For EC2

- 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?

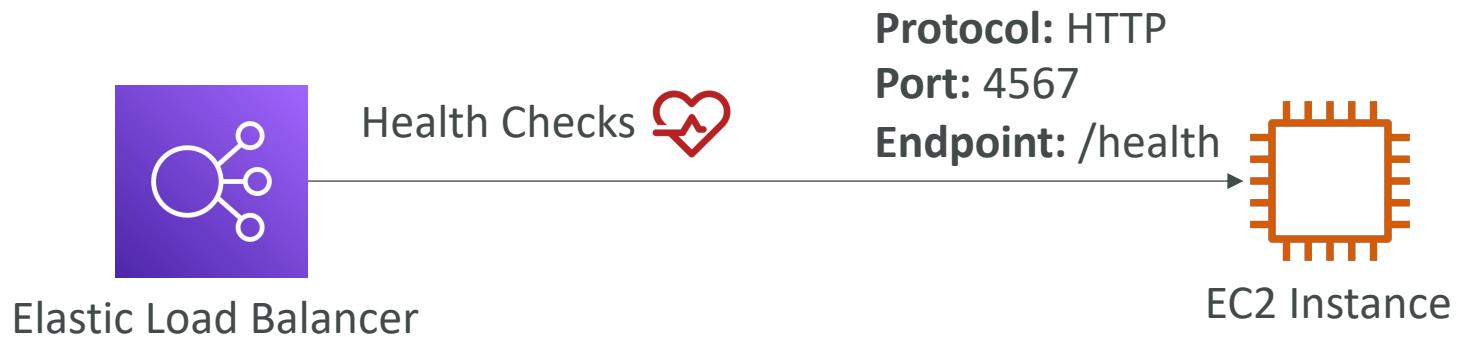
- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High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

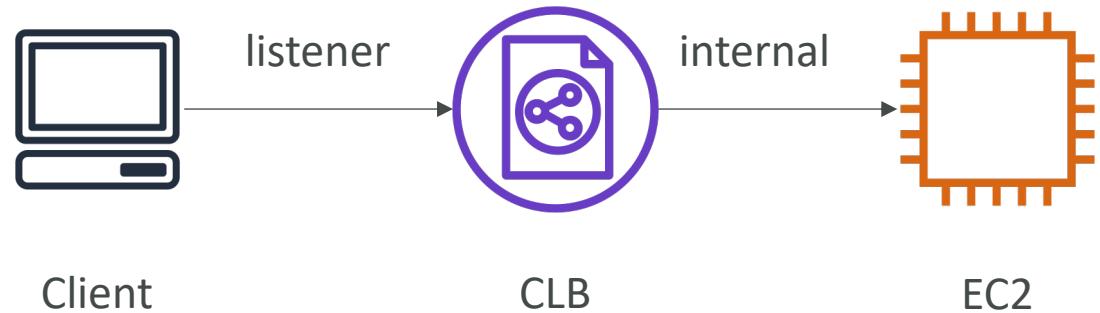
Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>	Description <small>i</small>
HTTP	TCP	80	0.0.0.0/0	Allow HTTP from an...
HTTPS	TCP	443	0.0.0.0/0	Allow HTTPS from a...

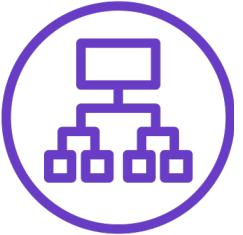
## Application Security Group: Allow traffic only from Load Balancer

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)

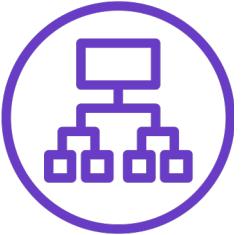
- Supports TCP (Layer 4), HTTP & HTTPS (Layer 7)
- Health checks are TCP or HTTP based
- Fixed hostname  
XXX.region.elb.amazonaws.com





# 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)

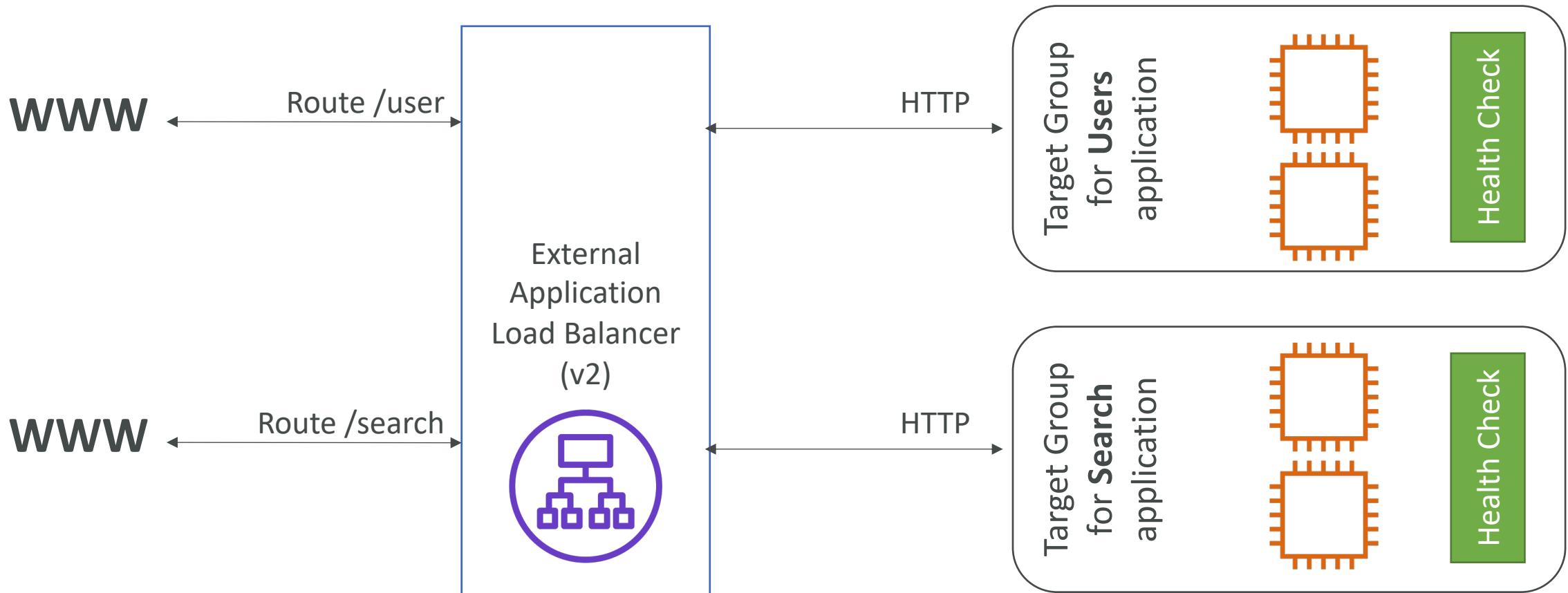


# Application Load Balancer (v2)

- 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



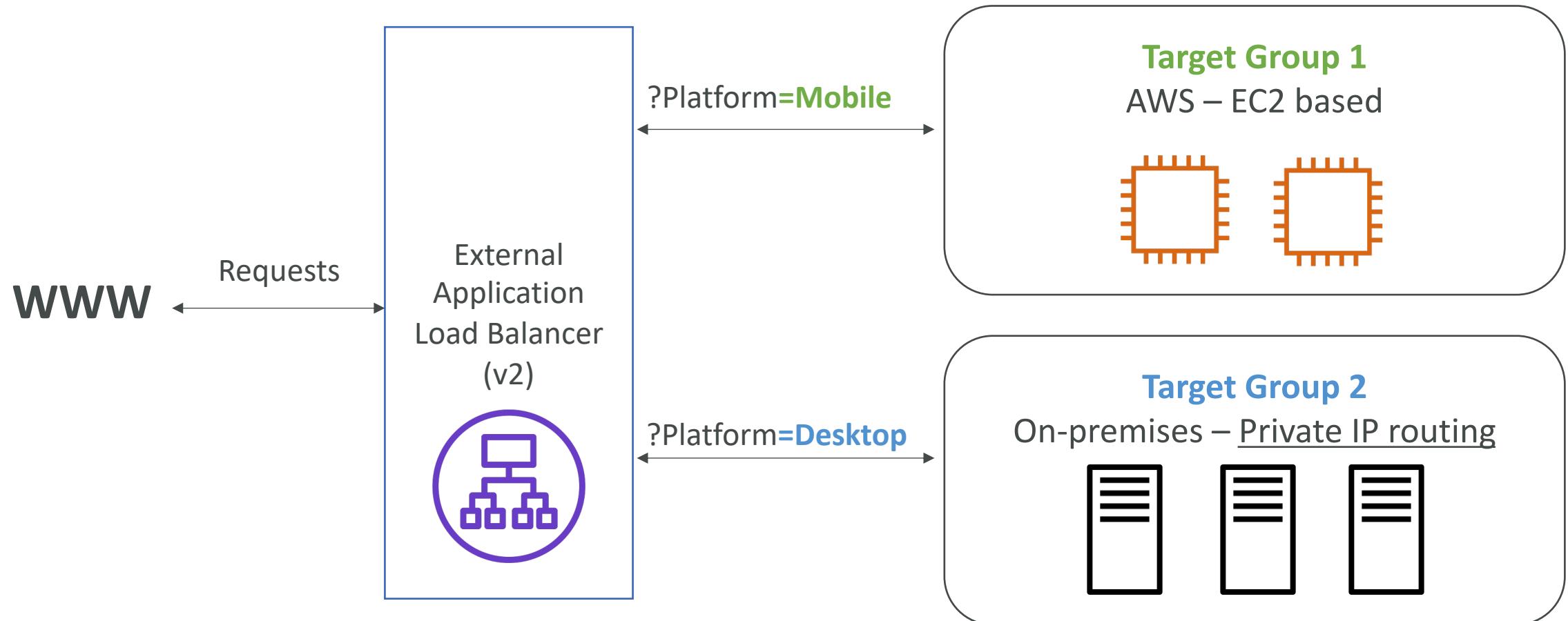
# Application Load Balancer (v2)

## 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)

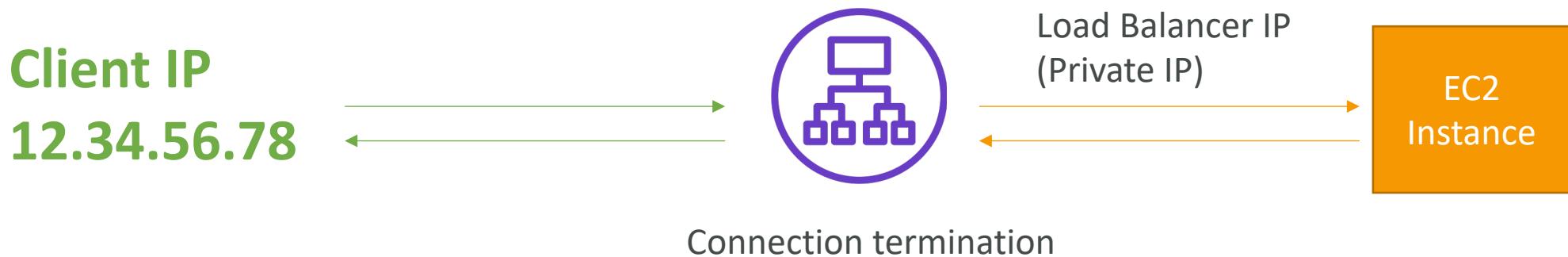
## Query Strings/Parameters Routing

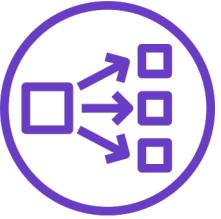


# Application Load Balancer (v2)

## 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)



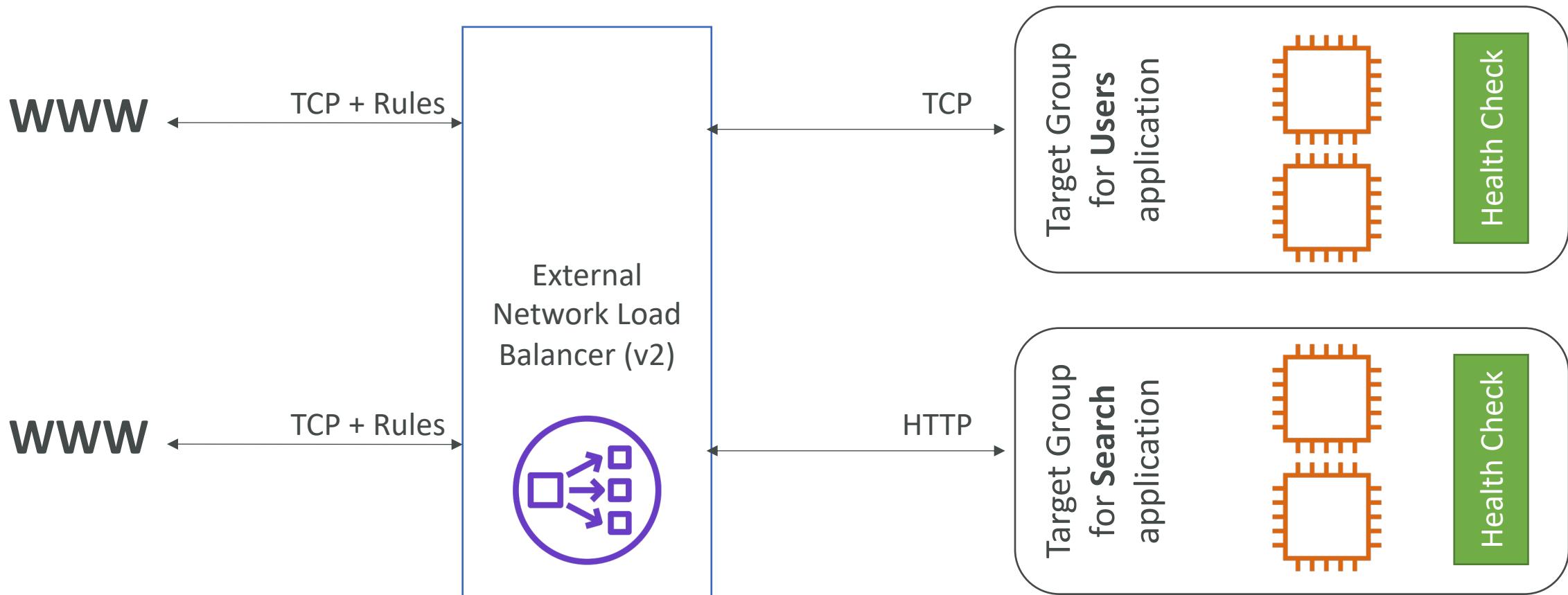


# Network Load Balancer (v2)

- 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

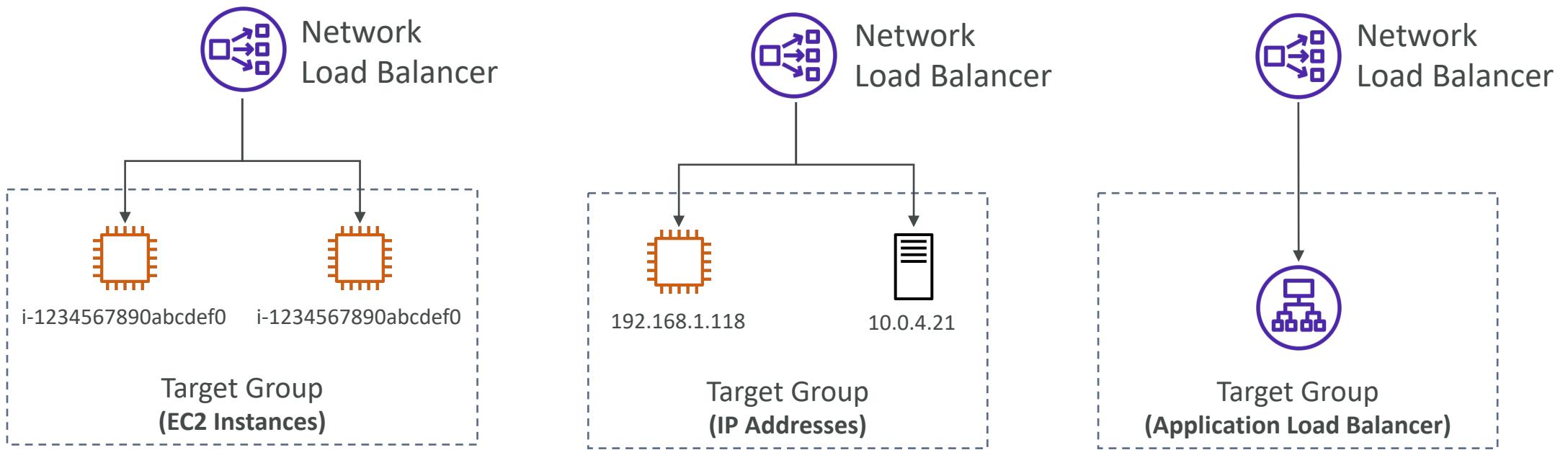
# Network Load Balancer (v2)

## TCP (Layer 4) Based Traffic



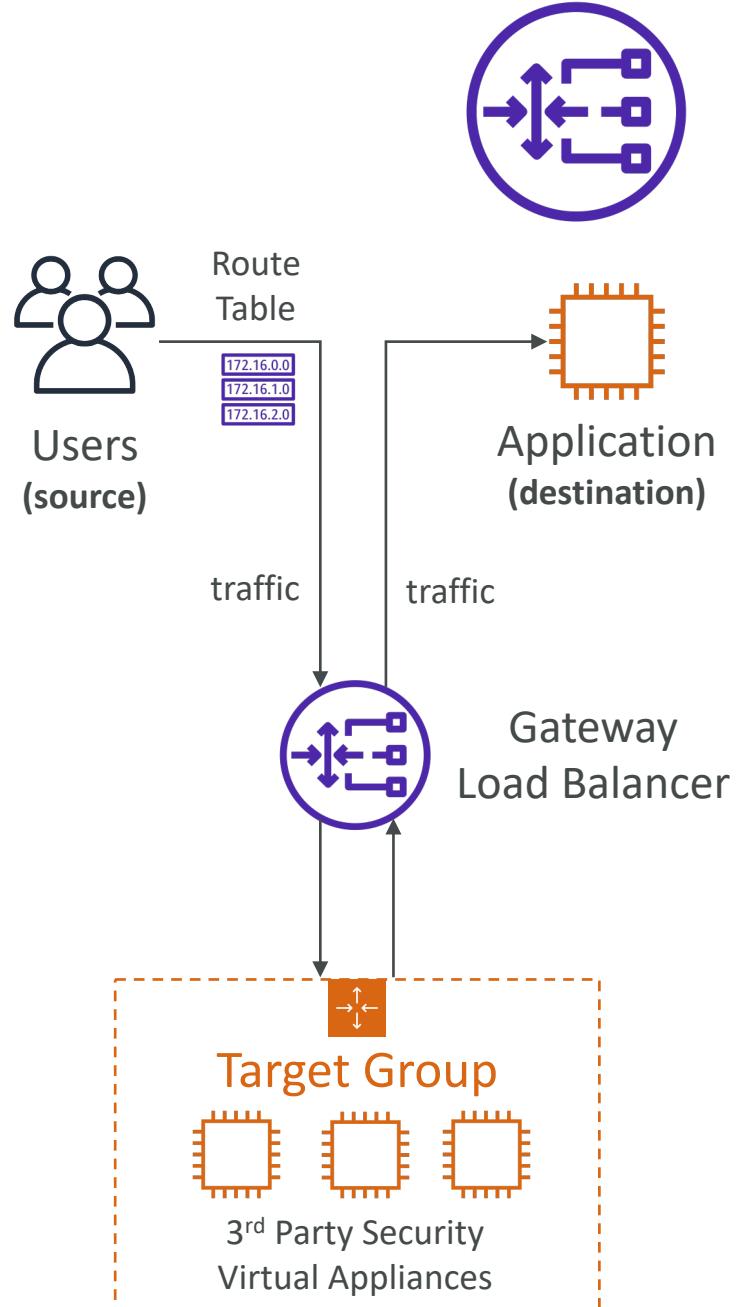
# Network Load Balancer – Target Groups

- EC2 instances
- IP Addresses – must be private IPs
- Application Load Balancer



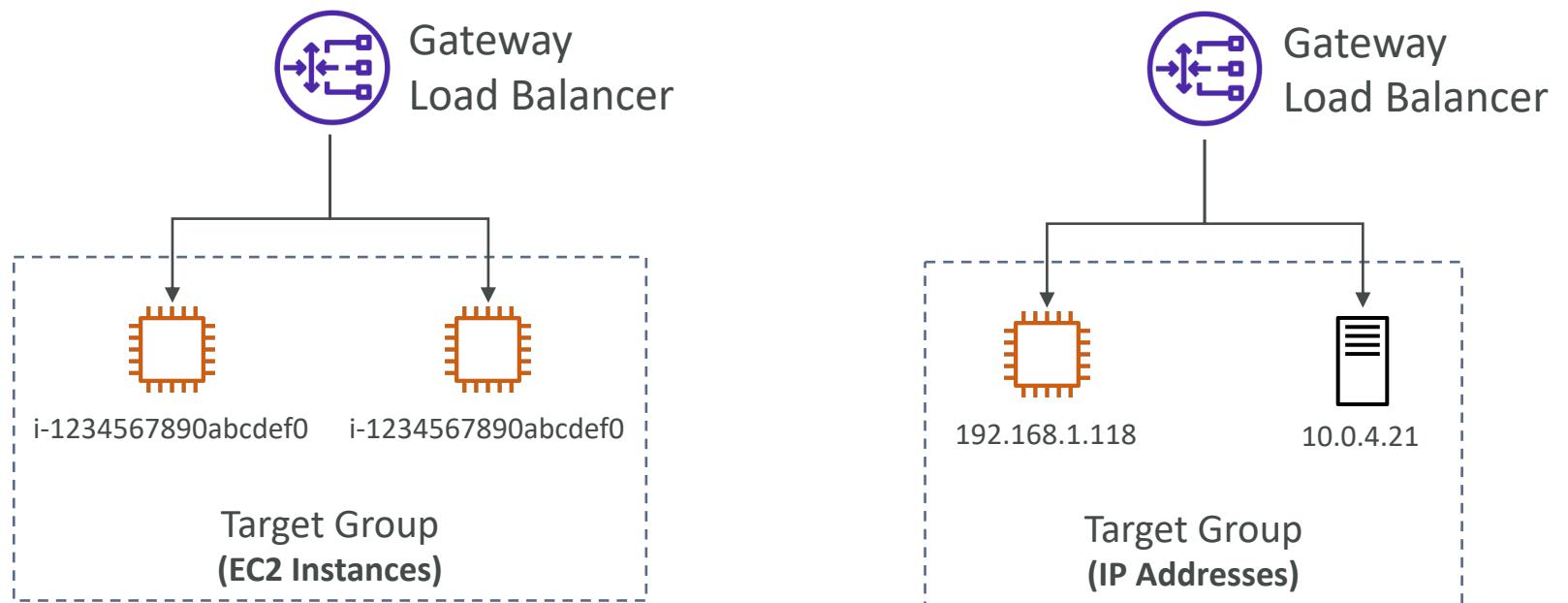
# Gateway Load Balancer

- Deploy, scale, and manage a fleet of 3<sup>rd</sup> 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



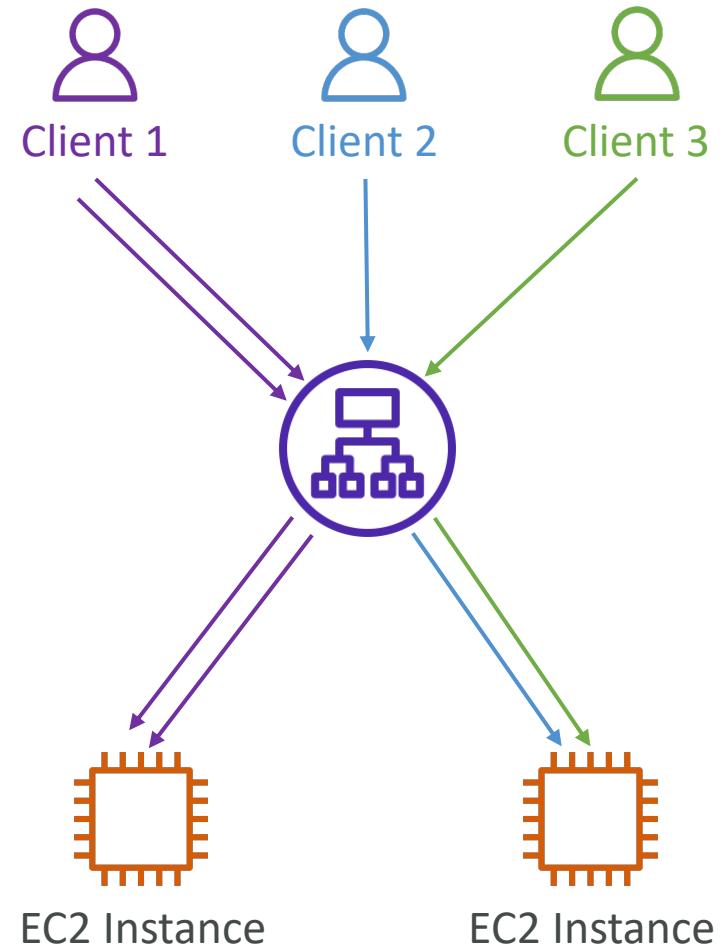
# Gateway Load Balancer – Target Groups

- 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 Balancers & Application Load Balancers
- 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



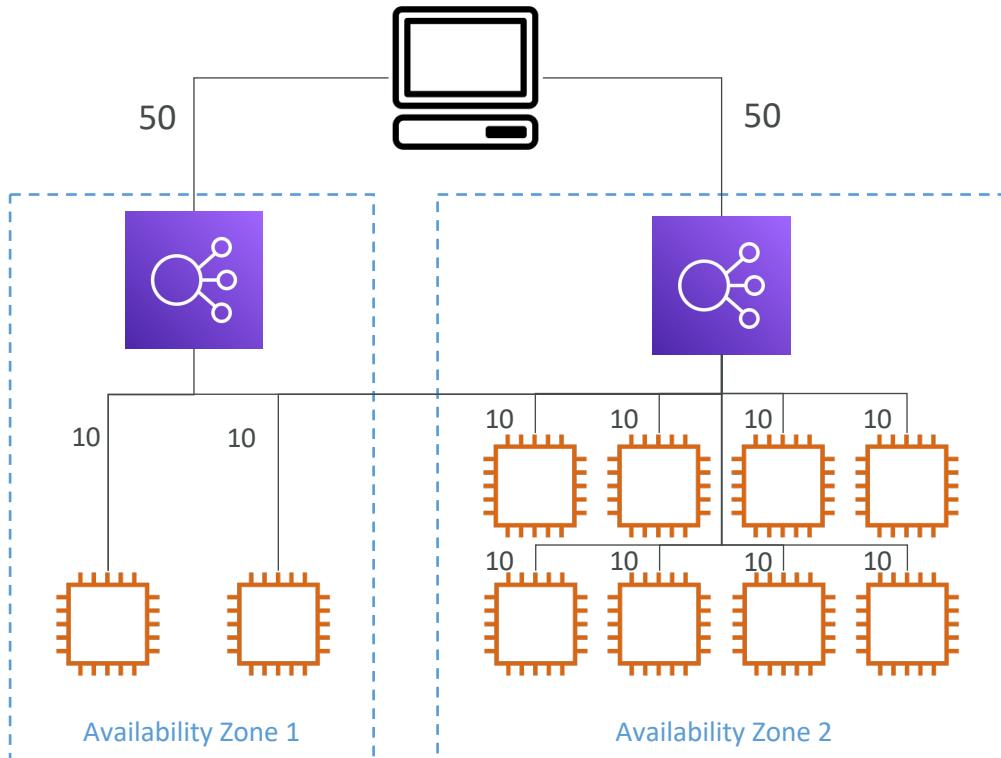
# Sticky Sessions – Cookie Names

- 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

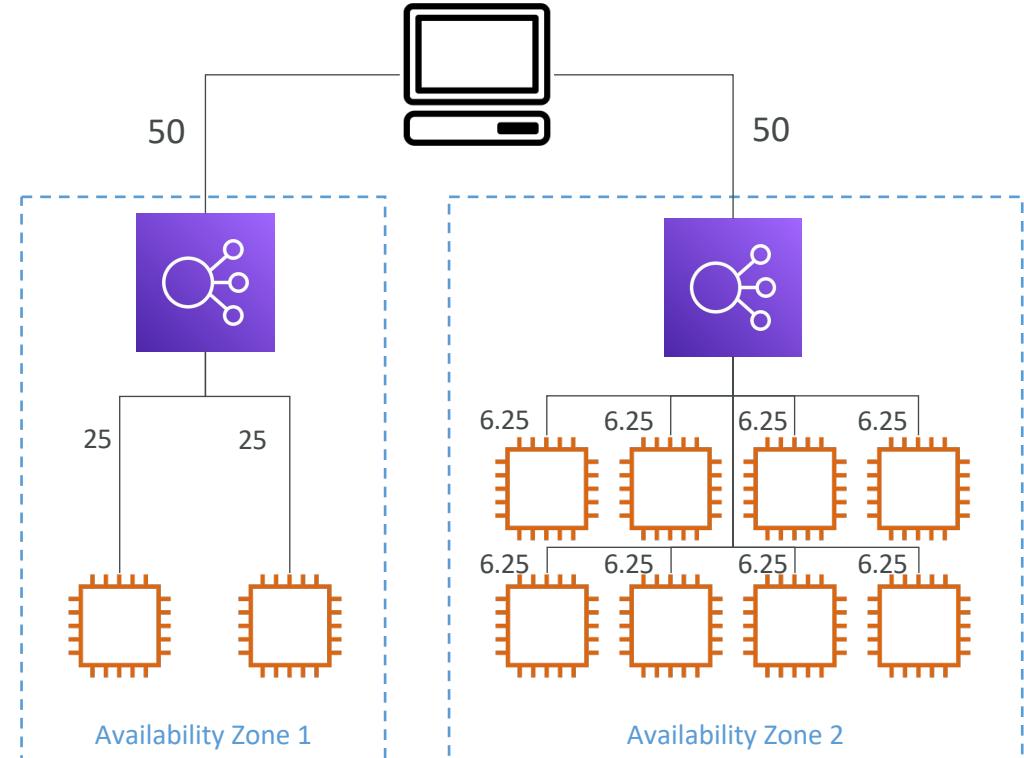
## With Cross Zone Load Balancing:

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
  - Always on (can't be disabled)
  - No charges for inter AZ data
- Network 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



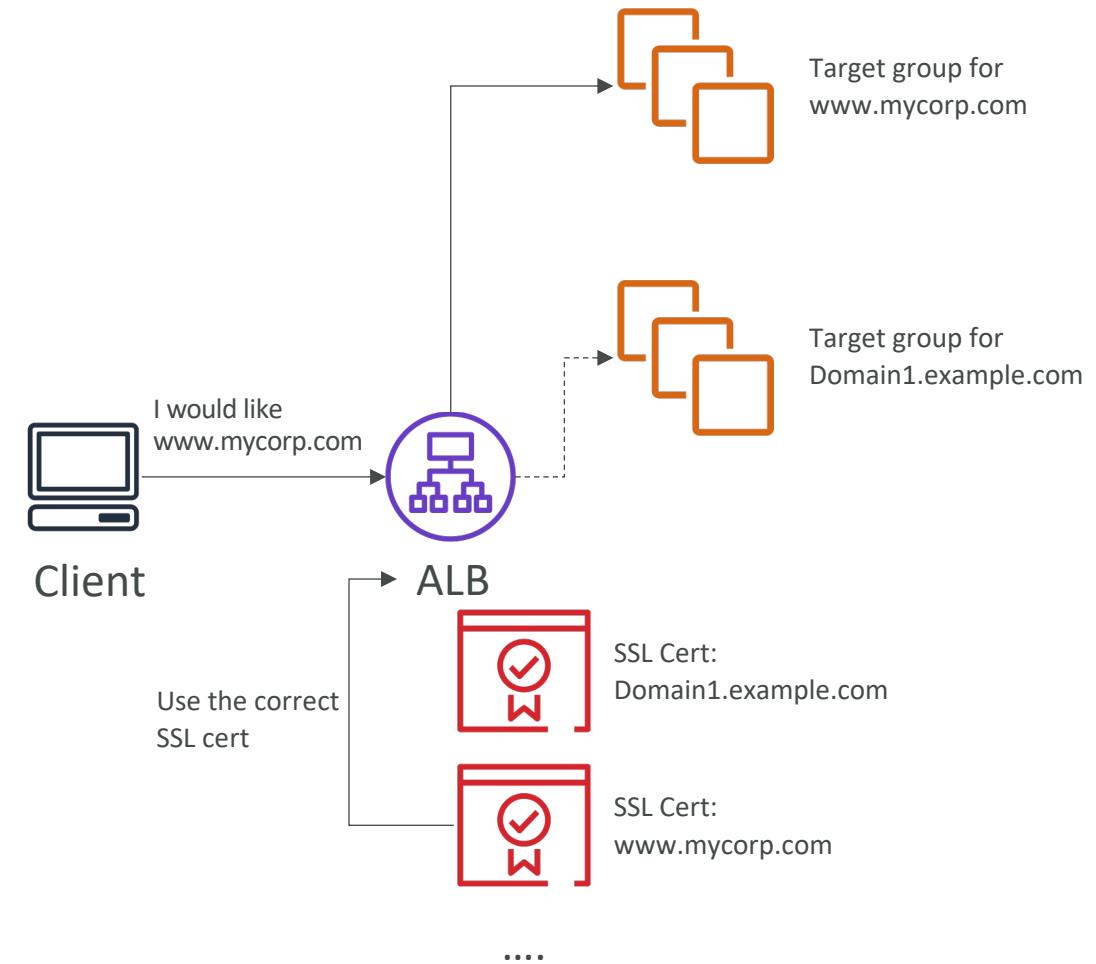
- The load balancer uses an X.509 certificate (SSL/TLS server certificate)
- You can manage certificates using ACM (AWS Certificate Manager)
- You can create/upload your own certificates alternatively
- HTTPS listener:
  - You must specify a default certificate
  - You can add an optional list of certs 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)

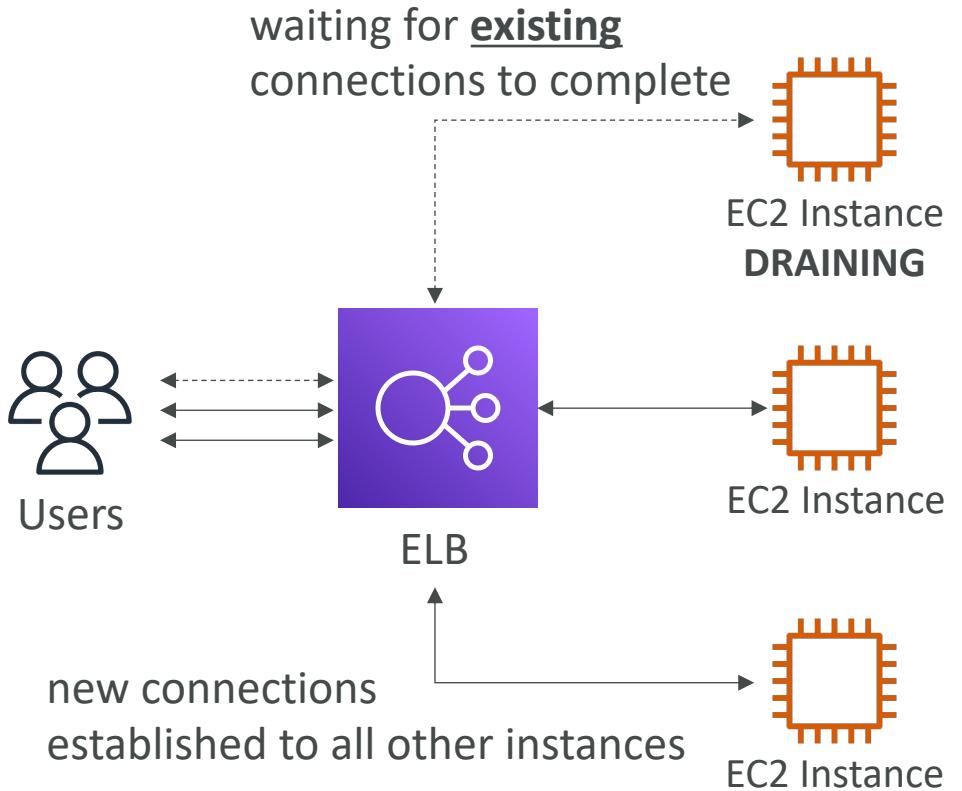


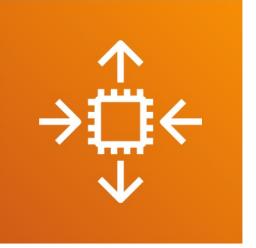
# Elastic Load Balancers – SSL Certificates

- **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

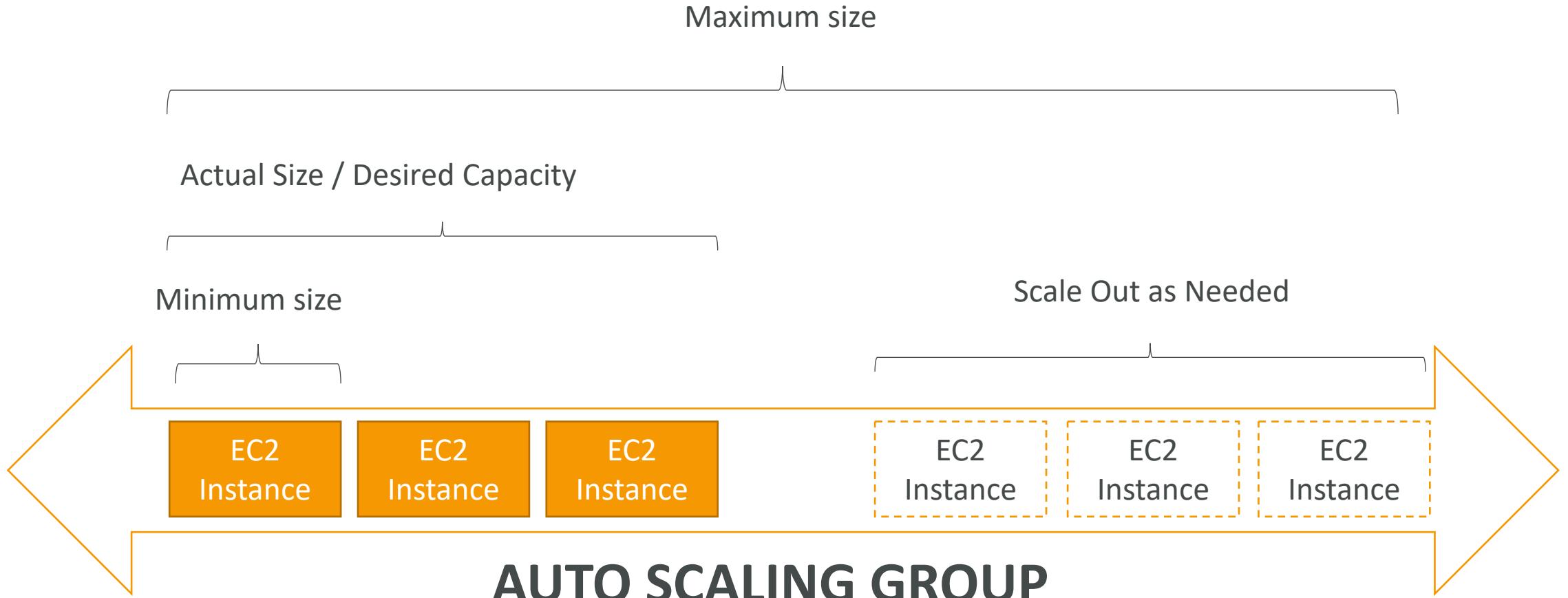




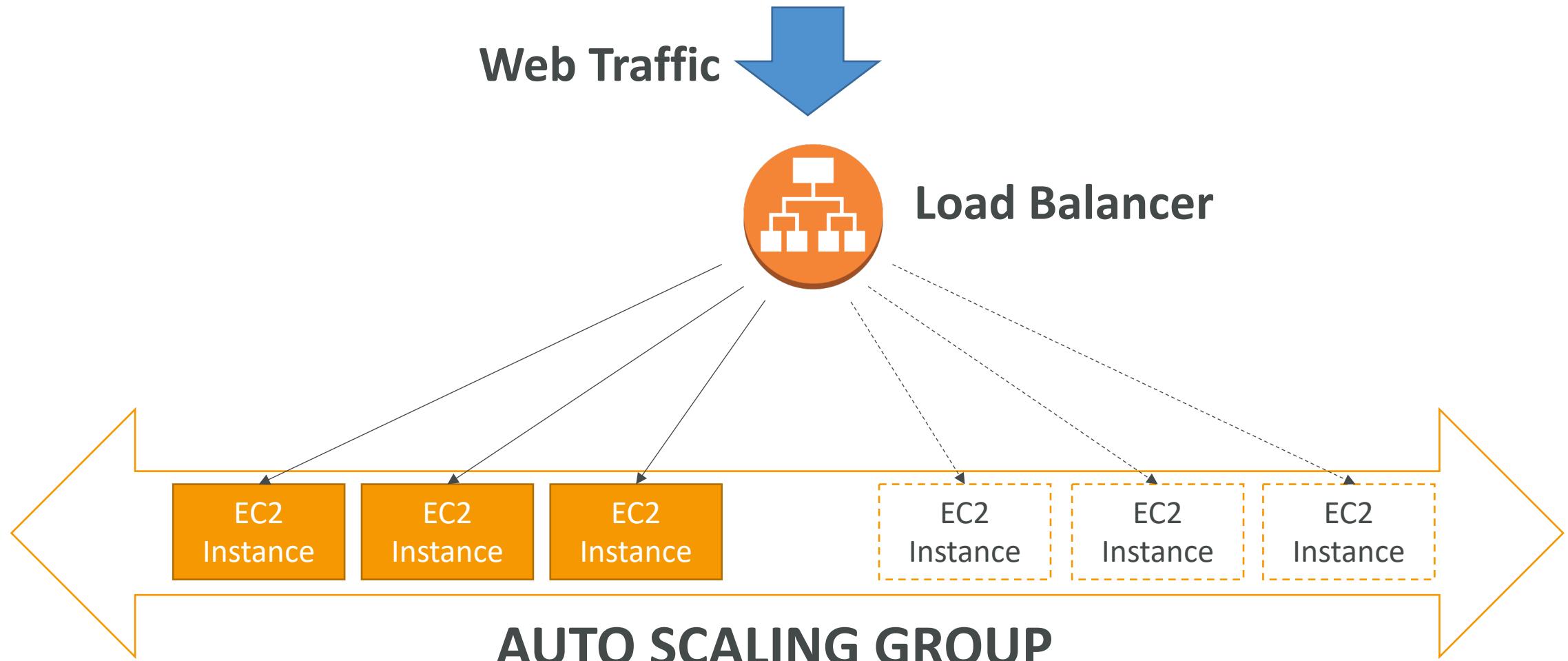
# What's an Auto Scaling Group?

- 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 machines running
  - Automatically Register new instances to a load balancer

# Auto Scaling Group in AWS



# Auto Scaling Group in AWS With Load Balancer

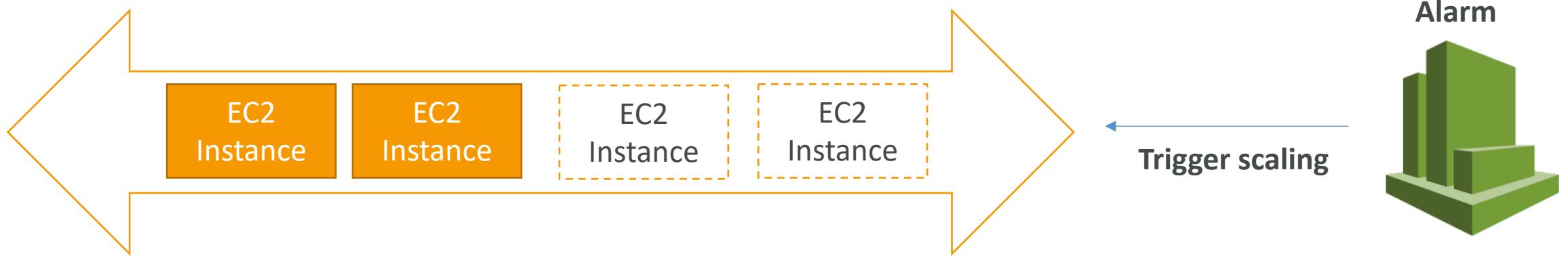


# ASGs have the following attributes

- A launch configuration
  - AMI + Instance Type
  - EC2 User Data
  - EBS Volumes
  - Security Groups
  - SSH Key Pair
- Min Size / Max Size / Initial Capacity
- Network + Subnets Information
- Load Balancer Information
- Scaling Policies

# Auto Scaling Alarms

- It is possible to scale an ASG based on CloudWatch alarms
- An Alarm monitors a metric (such as Average CPU)
- Metrics 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 New Rules

- It is now possible to define "better" auto scaling rules that are directly managed by EC2
  - Target Average CPU Usage
  - Number of requests on the ELB per instance
  - Average Network In
  - Average Network Out
- These rules are easier to set up and can make more sense

# Auto Scaling Custom Metric

- We can auto scale based on a custom metric (ex: number of connected users)
- 1. Send custom metric from application on EC2 to CloudWatch (PutMetric API)
- 2. Create CloudWatch alarm to react to low / high values
- 3. Use the CloudWatch alarm as the scaling policy for ASG

# ASG Brain Dump

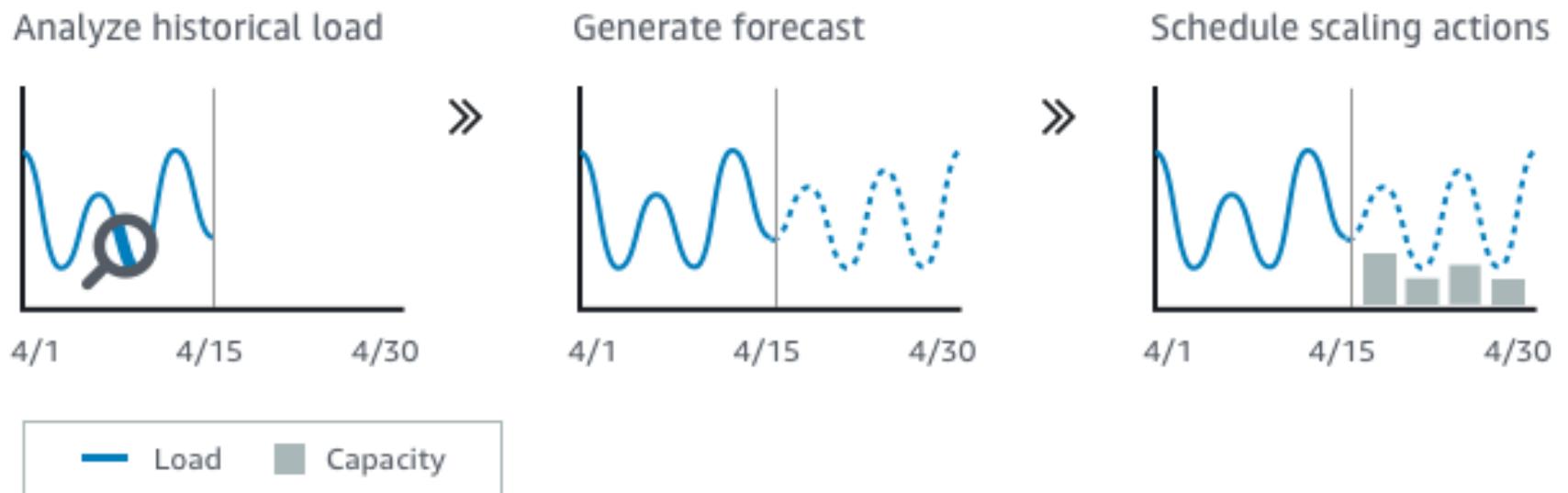
- Scaling policies can be on CPU, Network... and can even be on custom metrics or based on a schedule (if you know your visitors patterns)
- ASGs use Launch configurations or Launch Templates (newer)
- To update an ASG, you must provide a new launch configuration / launch template
- IAM roles attached to an ASG will get assigned to EC2 instances
- ASG are free. You pay for the underlying resources being launched
- Having instances under an ASG means that if they get terminated for whatever reason, the ASG will automatically **create new ones as a replacement**. Extra safety!
- ASG can terminate instances marked as unhealthy by an LB (and hence replace them)

# 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

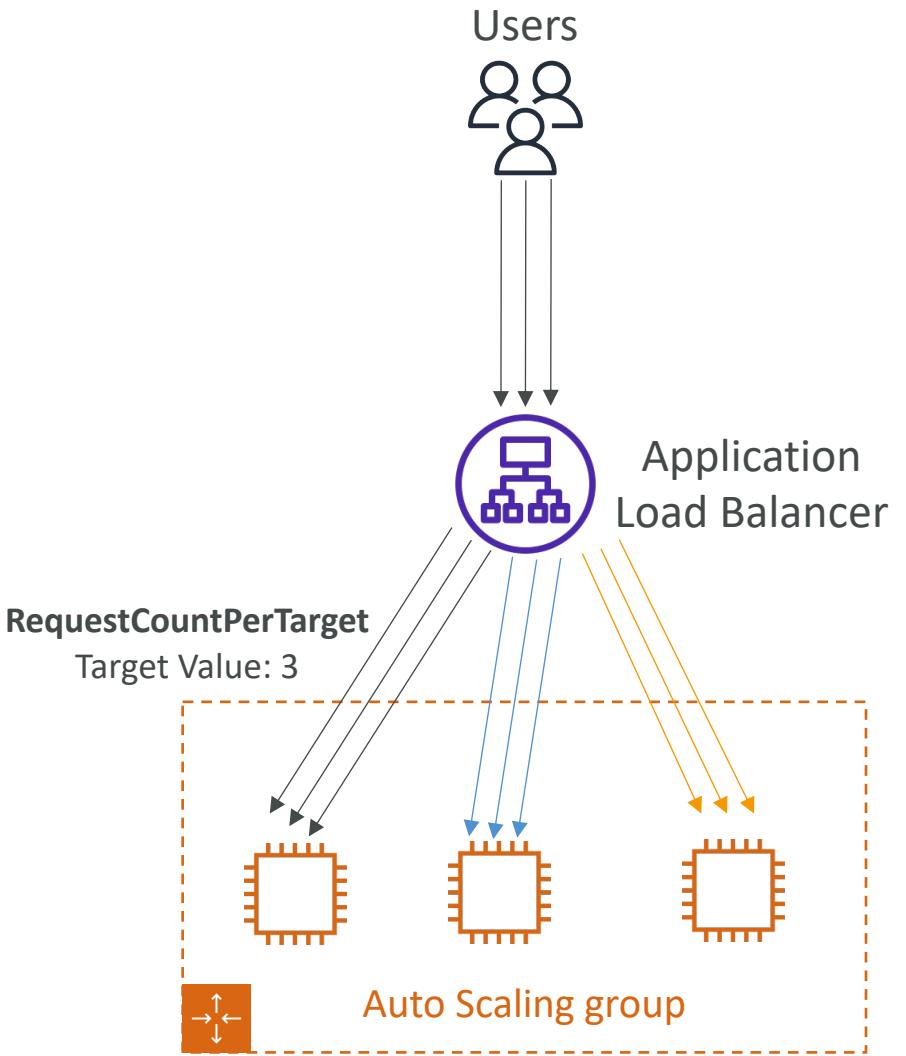
# Auto Scaling Groups – Predictive Scaling

- Predictive scaling: continuously forecast load and schedule scaling ahead



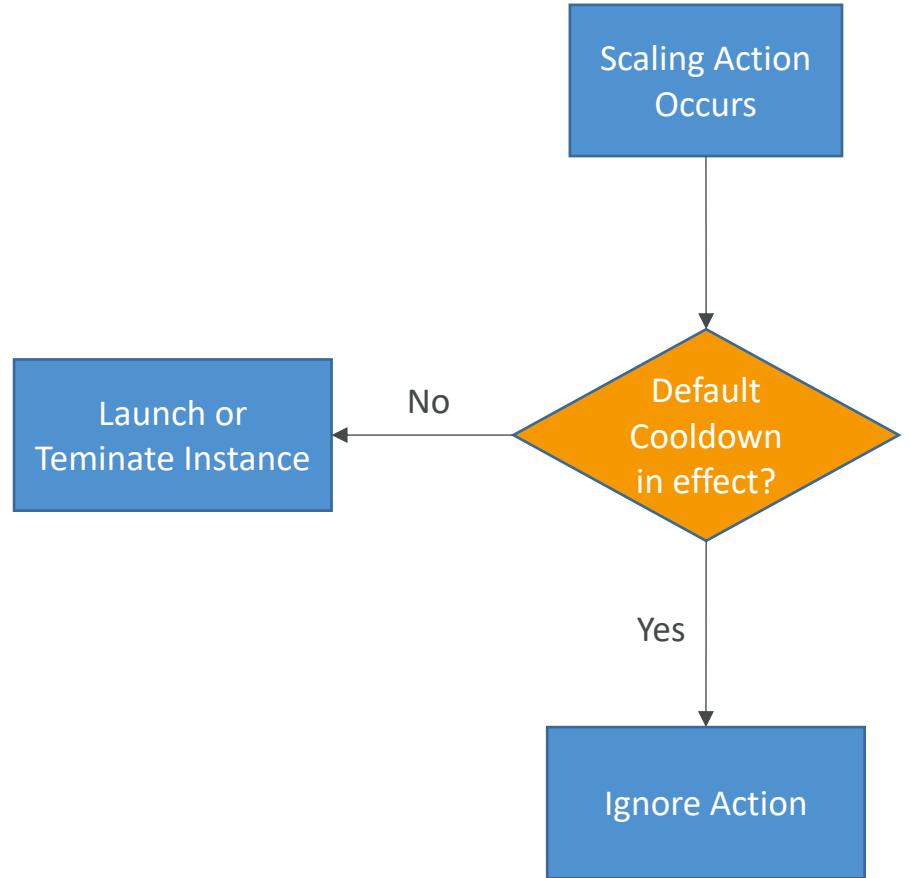
# Good metrics to scale on

- **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)



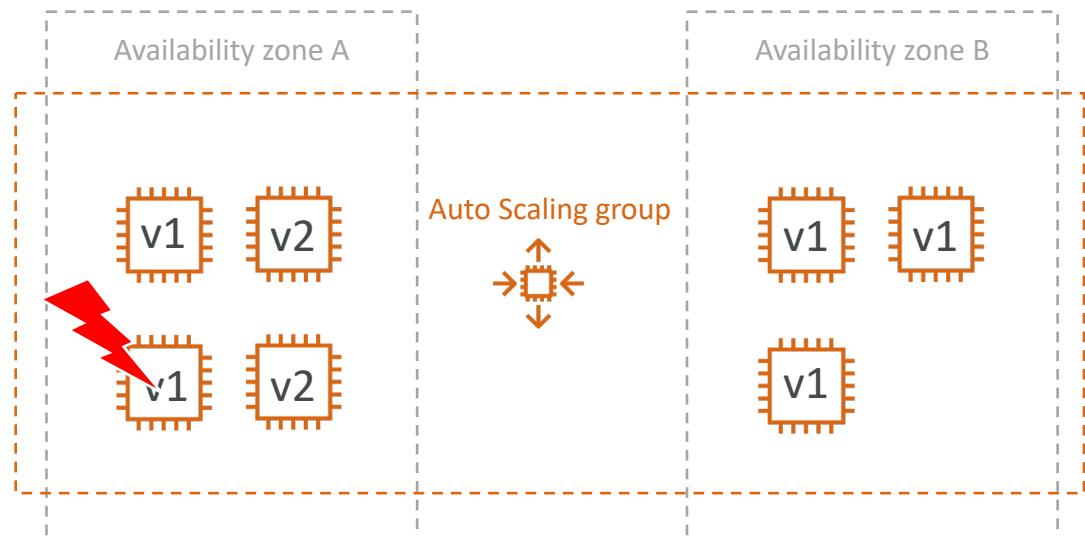
# Auto Scaling Groups - Scaling Cooldowns

- 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



# ASG for Solutions Architects

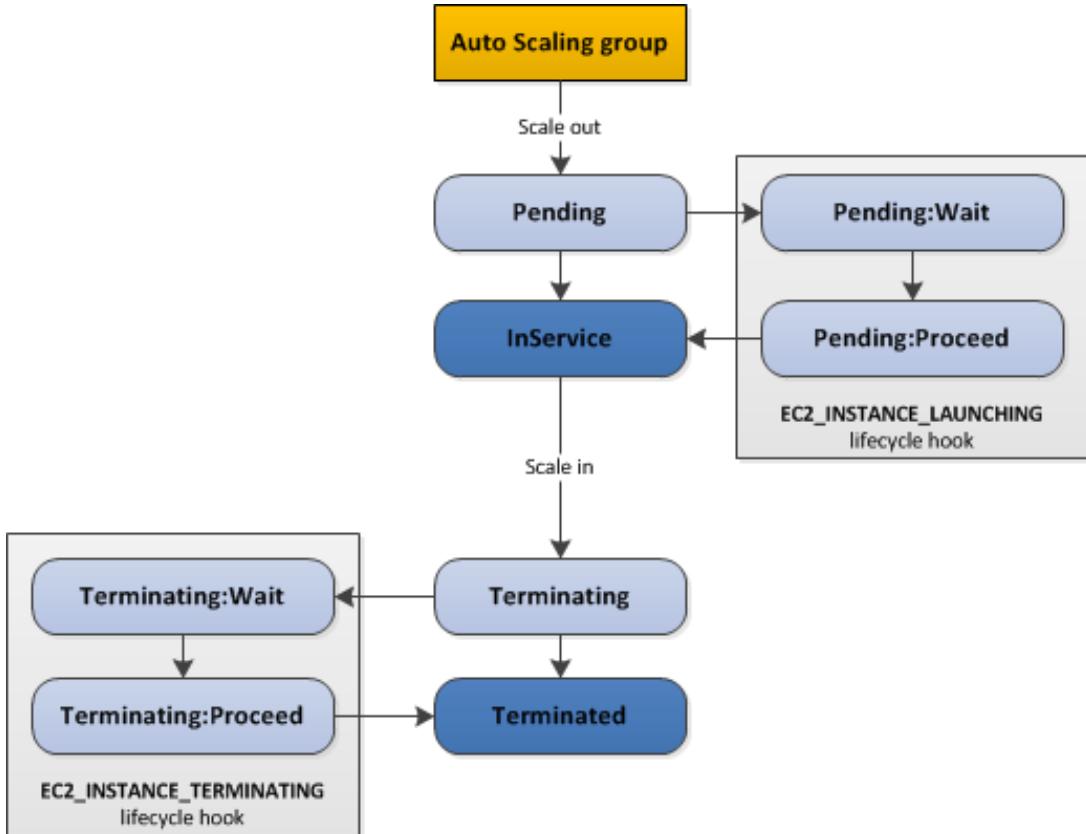
- ASG Default Termination Policy (simplified version):
  1. Find the AZ which has the most number of instances
  2. If there are multiple instances in the AZ to choose from, delete the one with the oldest launch configuration
- ASG tries to balance the number of instances across AZ by default



# ASG for Solutions Architects

## Lifecycle Hooks

- By default as soon as an instance is launched in an ASG it's in service.
- You have the ability to perform extra steps before the instance goes in service (Pending state)
- You have the ability to perform some actions before the instance is terminated (Terminating state)



<https://docs.aws.amazon.com/autoscaling/ec2/userguide/lifecycle-hooks.html>

# ASG for Solutions Architect

## Launch Template vs Launch Configuration

- Both:
  - ID of the Amazon Machine Image (AMI), the instance type, a key pair, security groups, and the other parameters that you use to launch EC2 instances (tags, EC2 user-data...)
- **Launch Configuration (legacy):**
  - Must be re-created every time
- **Launch Template (newer):**
  - Can have multiple versions
  - Create parameters subsets (partial configuration for re-use and inheritance)
  - Provision using both On-Demand and Spot instances (or a mix)
  - Can use T2 unlimited burst feature
  - Recommended by AWS going forward

# RDS, Aurora & ElastiCache

# AWS 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

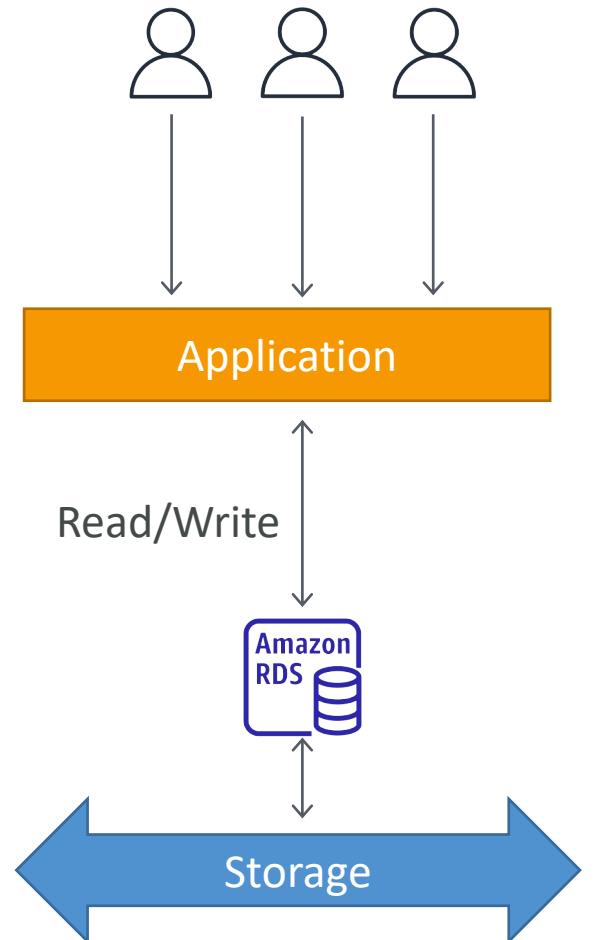
- RDS is a managed service:
  - Automated provisioning, OS patching
  - Continuous backups and restore to specific timestamp (Point in Time Restore)!
  - Monitoring dashboards
  - Read replicas for improved read performance
  - Multi AZ setup for DR (Disaster Recovery)
  - Maintenance windows for upgrades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

# RDS Backups

- Backups are automatically enabled in RDS
- Automated backups:
  - Daily full backup of the database (during the maintenance 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)
  - 7 days retention (can be increased to 35 days)
- DB Snapshots:
  - Manually triggered by the user
  - Retention of backup for as long as you want

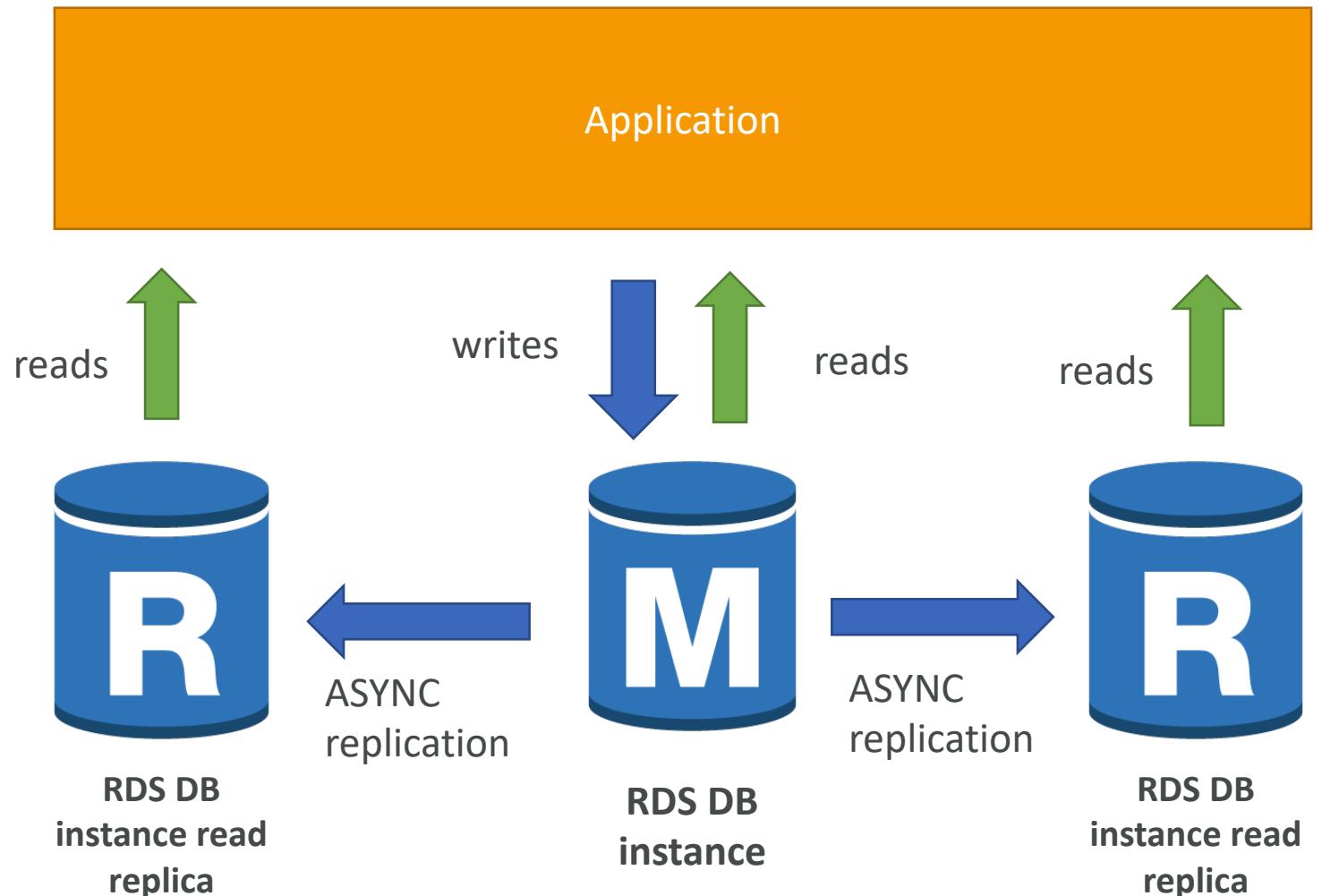
# RDS – Storage Auto Scaling

- 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)



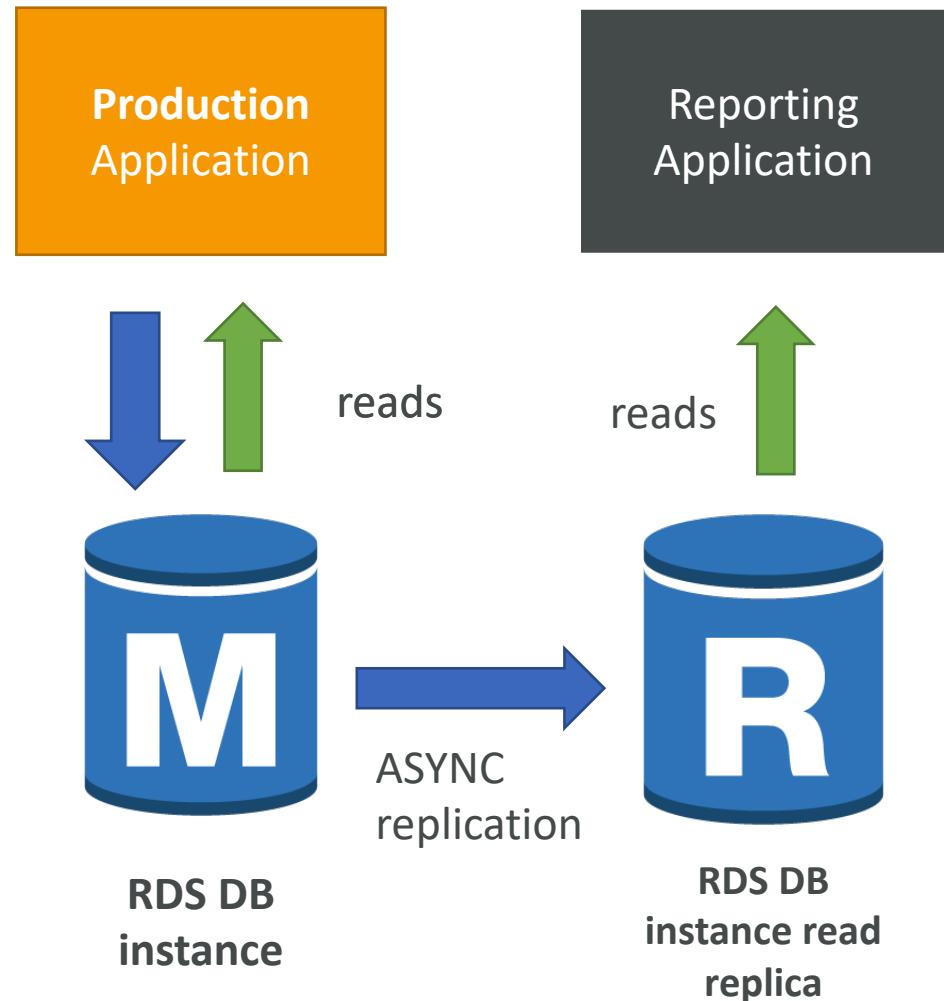
# RDS Read Replicas for read scalability

- Up to 5 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



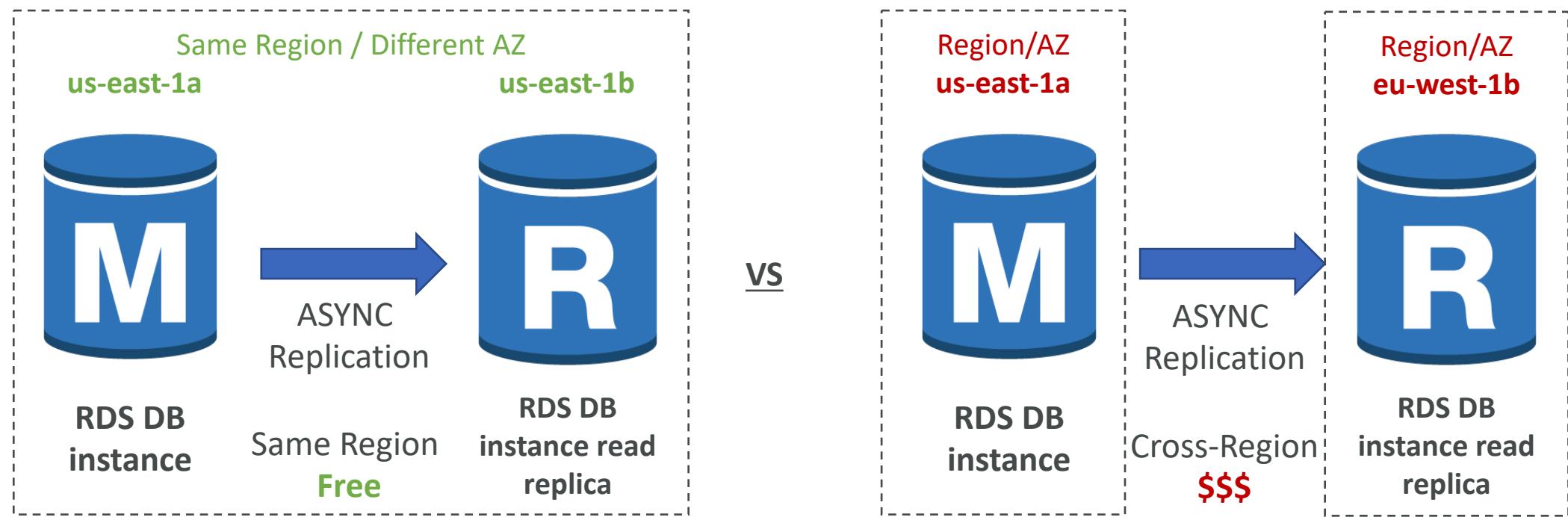
# RDS Read Replicas – Use Cases

- 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)



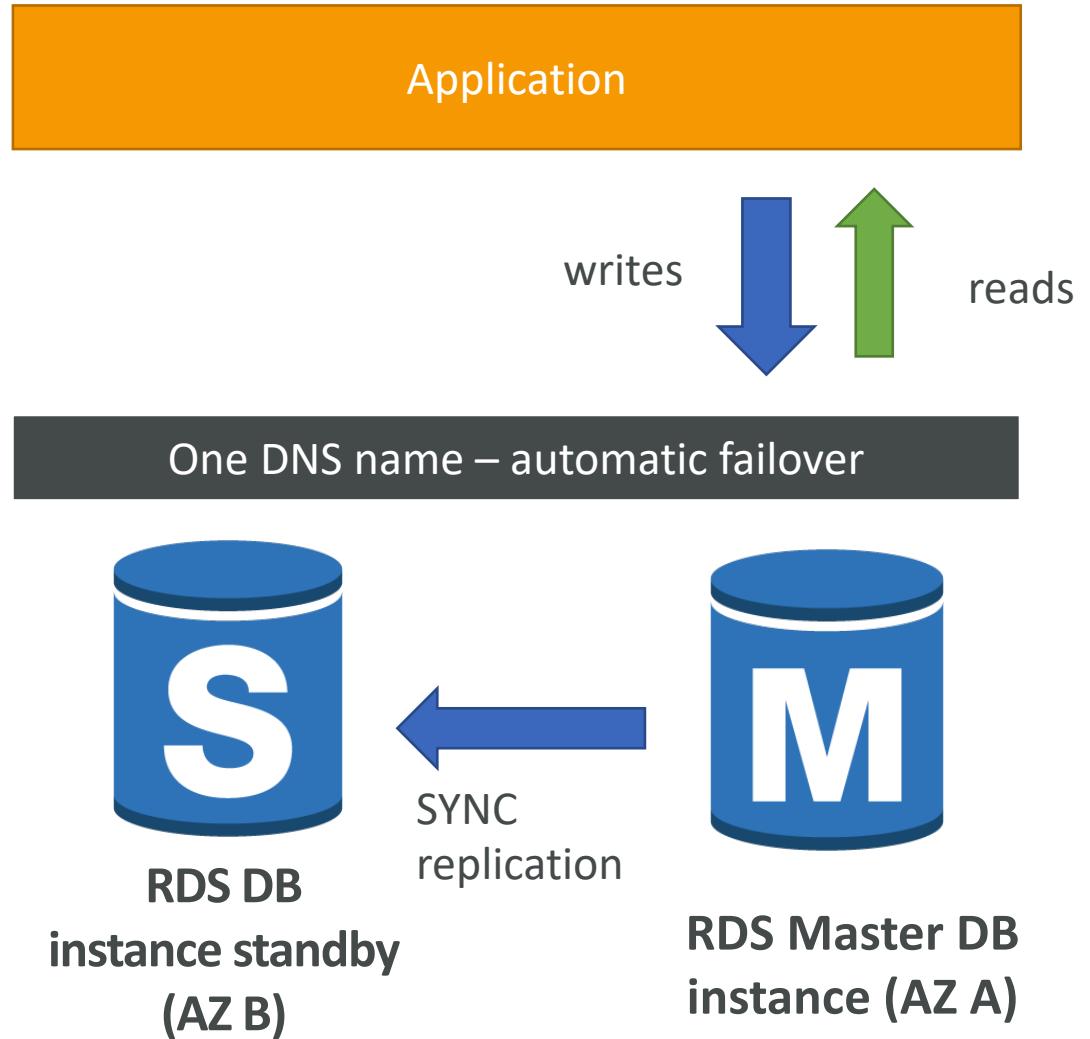
# RDS Read Replicas – Network Cost

- 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



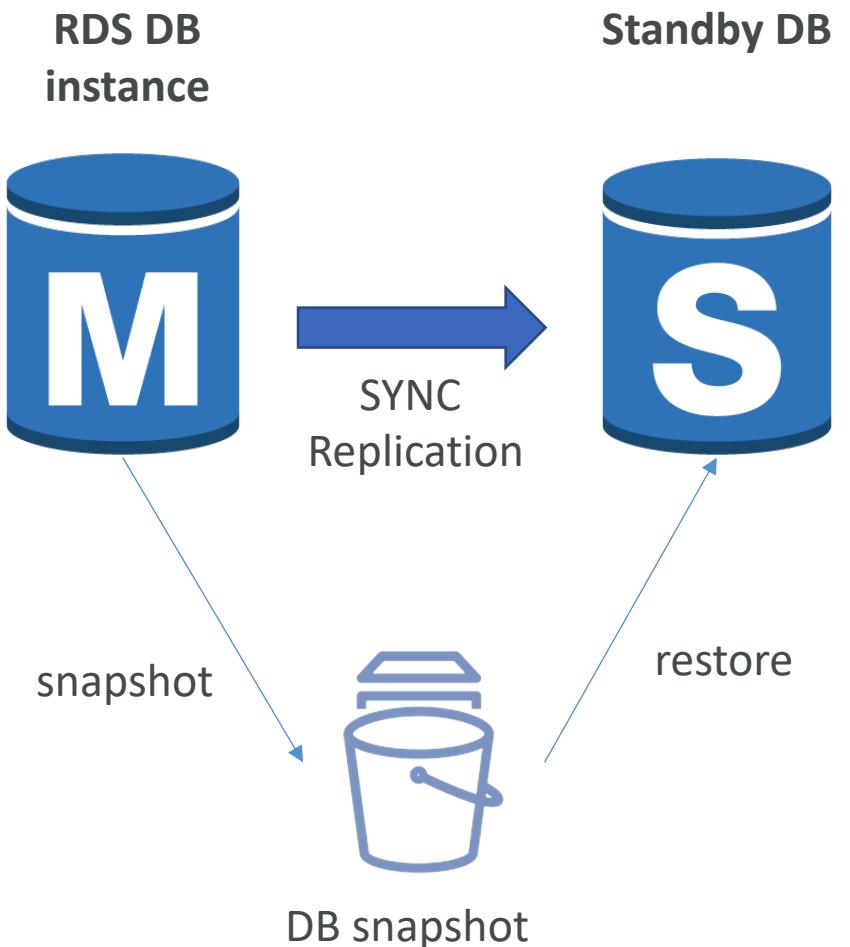
# RDS Multi AZ (Disaster Recovery)

- 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
- Multi-AZ replication is free
- 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 Security - Encryption

- At rest encryption
  - Possibility to encrypt the master & read replicas with AWS KMS - AES-256 encryption
  - Encryption has to be defined at launch time
  - **If the master is not encrypted, the read replicas cannot be encrypted**
  - Transparent Data Encryption (TDE) available for Oracle and SQL Server
- In-flight encryption
  - SSL certificates to encrypt data to RDS in flight
  - Provide SSL options with trust certificate when connecting to database
  - To enforce SSL:
    - PostgreSQL: rds.force\_ssl=1 in the AWS RDS Console (Parameter Groups)
    - MySQL: Within the DB:  
GRANT USAGE ON \*.\* TO 'mysqluser'@'%' REQUIRE SSL;

# RDS Encryption Operations

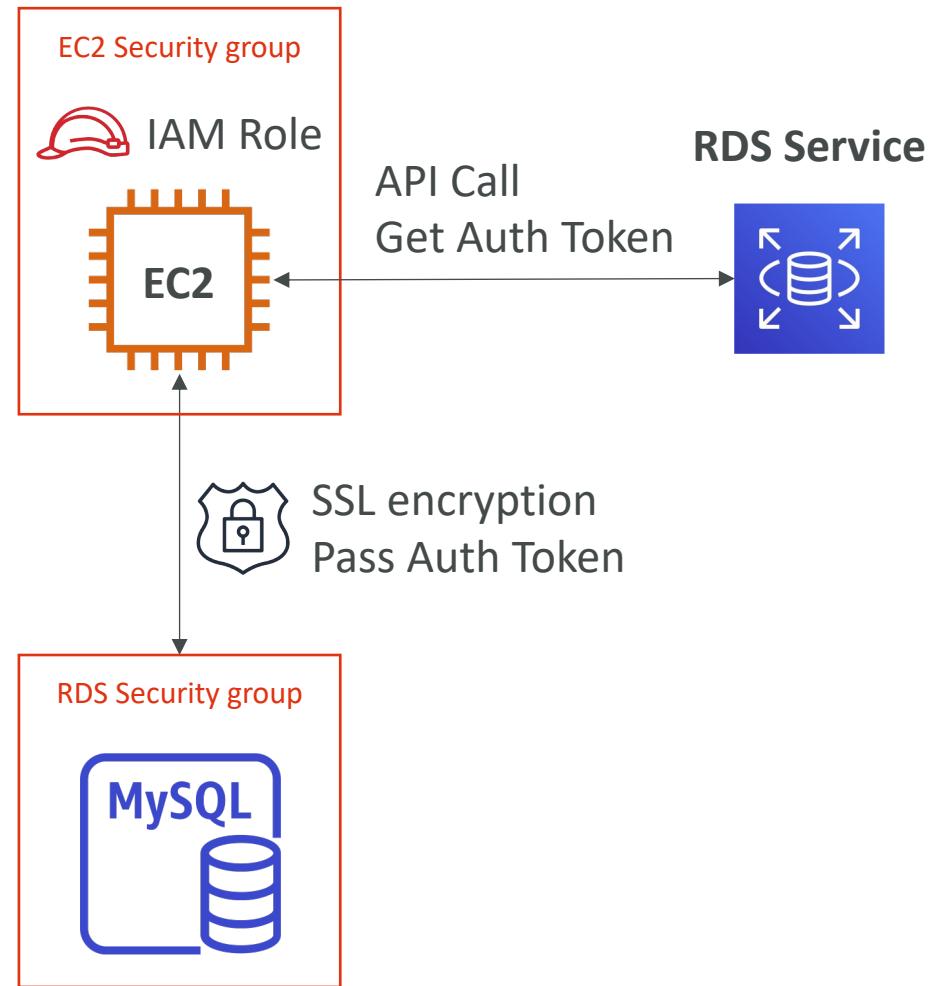
- Encrypting RDS backups
  - Snapshots of un-encrypted RDS databases are un-encrypted
  - Snapshots of encrypted RDS databases are encrypted
  - Can copy a snapshot into an encrypted one
- To encrypt an un-encrypted RDS database:
  - Create a snapshot of the un-encrypted database
  - Copy the snapshot and enable encryption for the snapshot
  - Restore the database from the encrypted snapshot
  - Migrate applications to the new database, and delete the old database

# RDS Security – Network & IAM

- Network Security
  - RDS databases are usually deployed within a private subnet, not in a public one
  - RDS security works by leveraging security groups (the same concept as for EC2 instances) – it controls which IP / security group can **communicate** with RDS
- Access Management
  - IAM policies help control who can **manage** AWS RDS (through the RDS API)
  - Traditional Username and Password can be used to **login** into the database
  - IAM-based authentication can be used to login into RDS MySQL & PostgreSQL

# RDS - IAM Authentication

- IAM database authentication works with MySQL and PostgreSQL
- You don't need a password, just an authentication token obtained through IAM & RDS API calls
- Auth token has a lifetime of 15 minutes
- Benefits:
  - Network in/out must be encrypted using SSL
  - IAM to centrally manage users instead of DB
  - Can leverage IAM Roles and EC2 Instance profiles for easy integration



# RDS Security – Summary

- Encryption at rest:
  - Is done only when you first create the DB instance
  - or: unencrypted DB => snapshot => copy snapshot as encrypted => create DB from snapshot
- Your responsibility:
  - Check the ports / IP / security group inbound rules in DB's SG
  - In-database user creation and permissions or manage through IAM
  - Creating a database with or without public access
  - Ensure parameter groups or DB is configured to only allow SSL connections
- AWS responsibility:
  - No SSH access
  - No manual DB patching
  - No manual OS patching
  - No way to audit the underlying instance

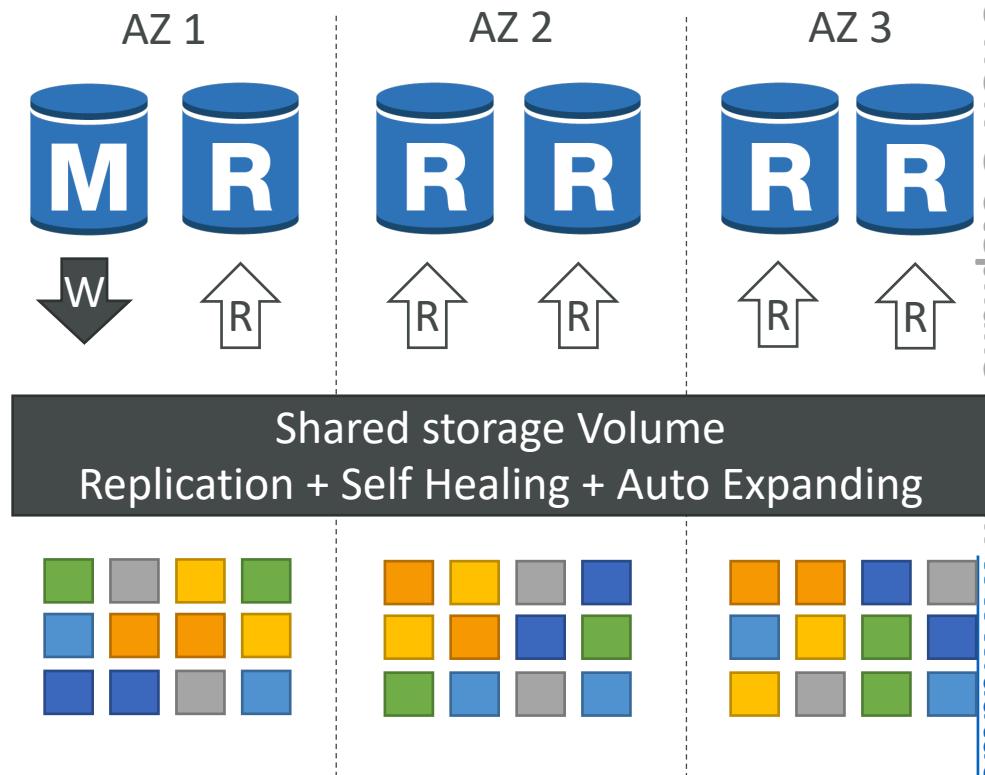


# Amazon Aurora

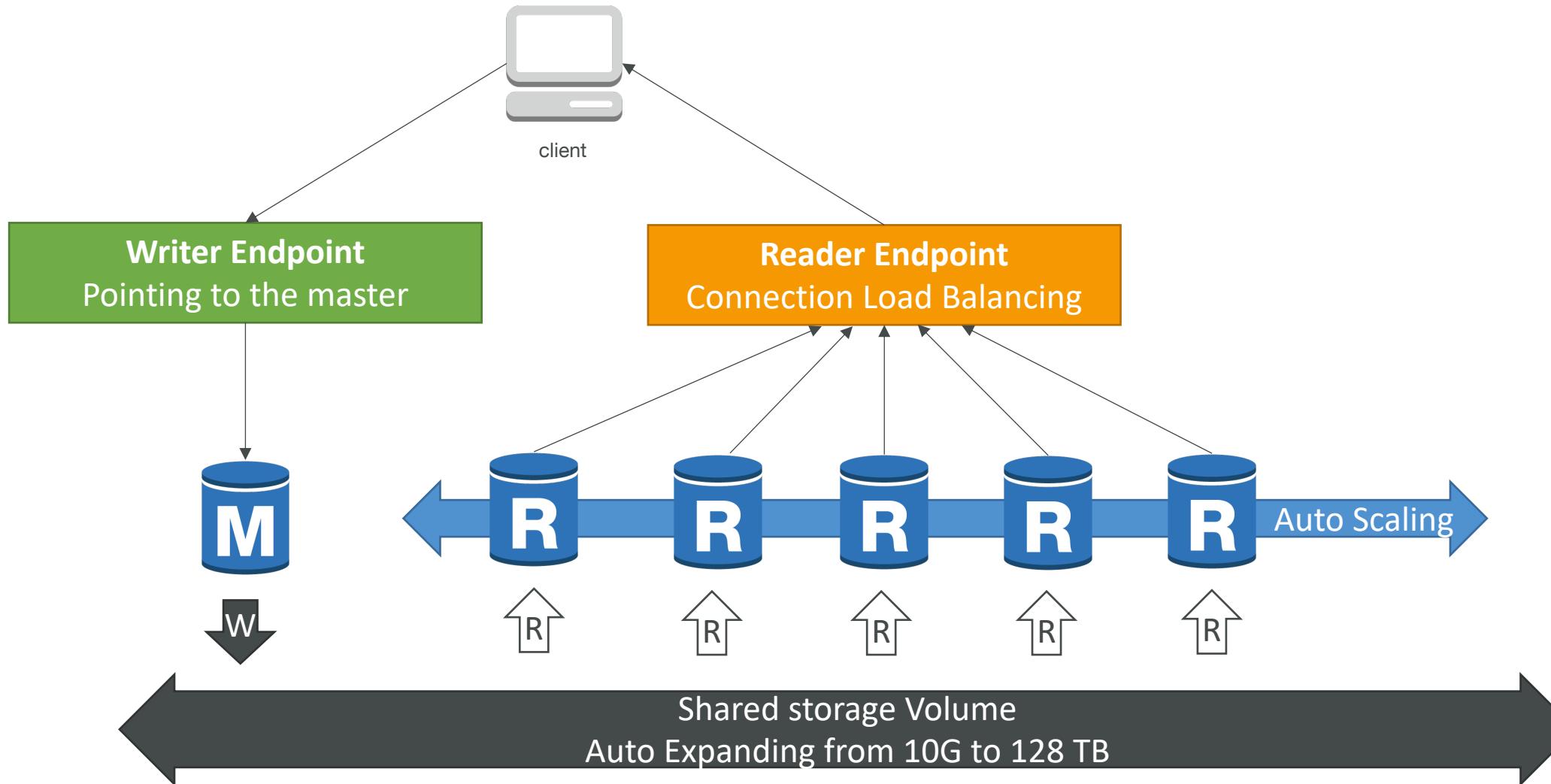
- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is “AWS cloud optimized” and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 128 TB.
- Aurora can have 15 replicas while MySQL has 5, and the replication process is faster (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



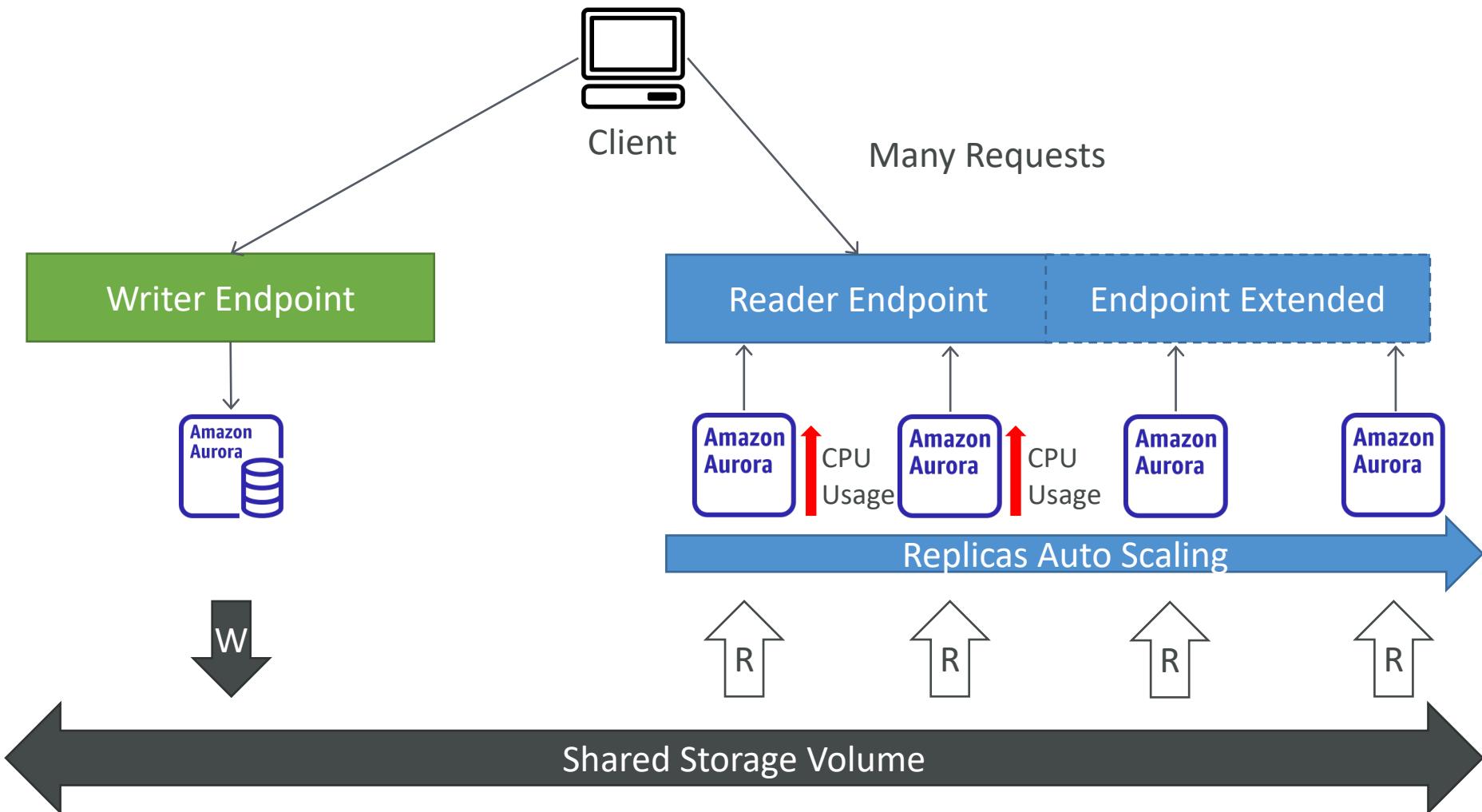
# Features of Aurora

- 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 Security

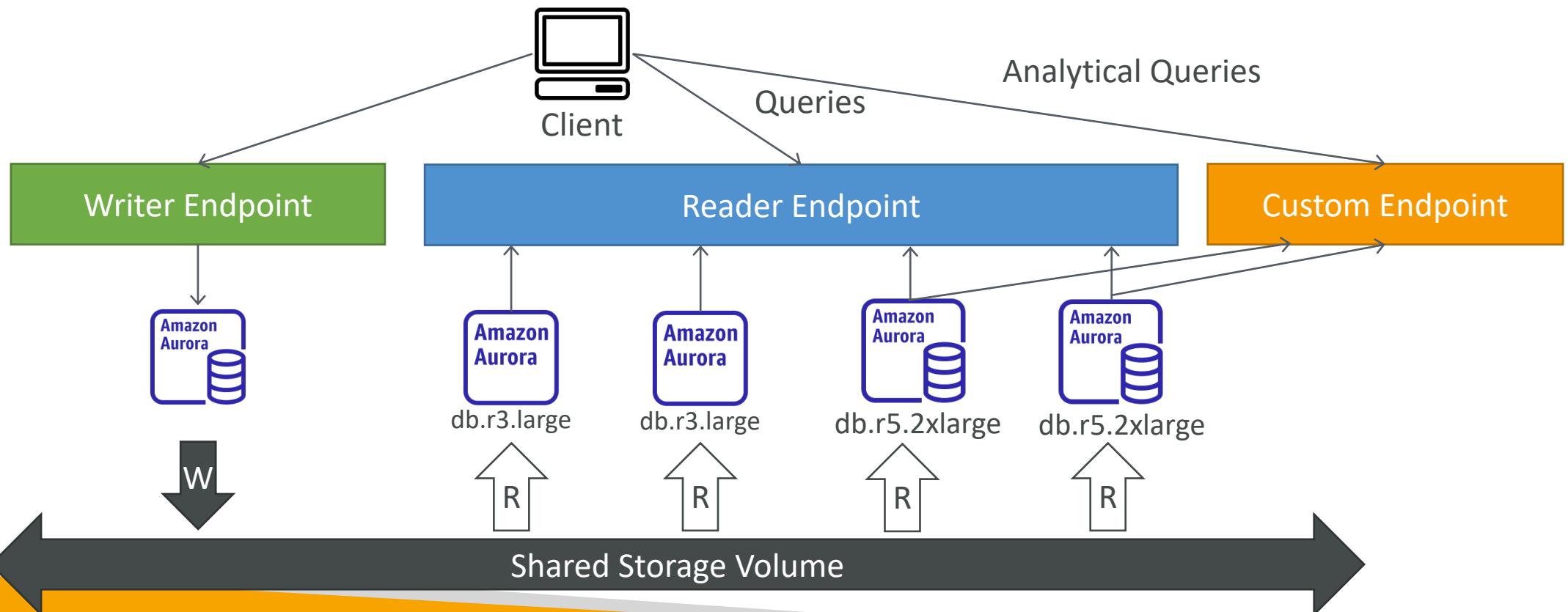
- Similar to RDS because uses the same engines
- Encryption at rest using KMS
- Automated backups, snapshots and replicas are also encrypted
- Encryption in flight using SSL (same process as MySQL or Postgres)
- **Possibility to authenticate using IAM token (same method as RDS)**
- You are responsible for protecting the instance with security groups
- You can't SSH

# Aurora Replicas - Auto Scaling



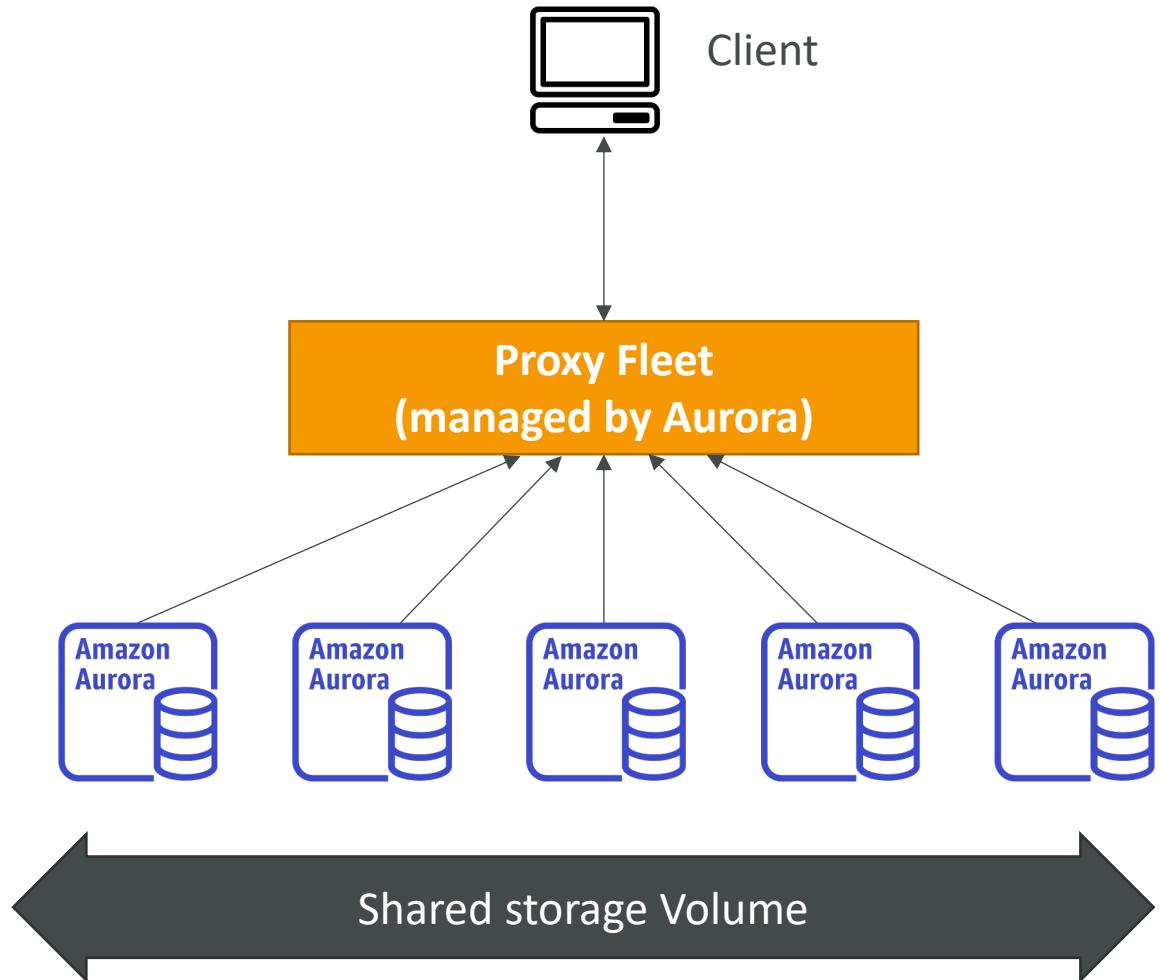
# Aurora – Custom Endpoints

- 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



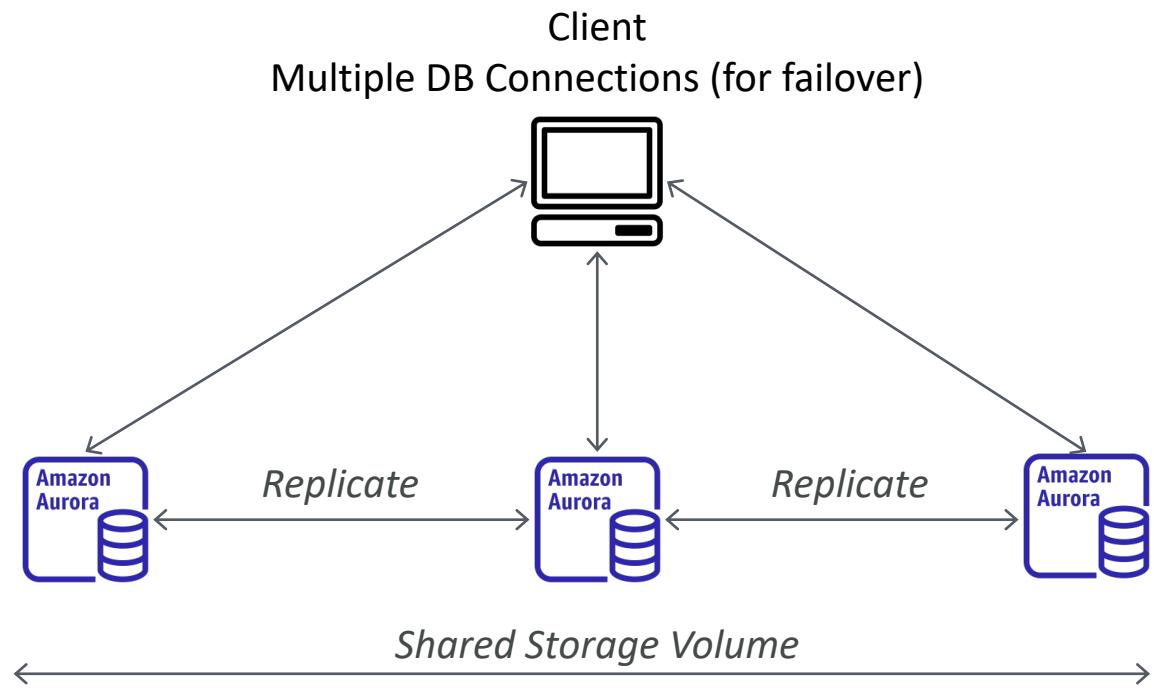
# Aurora Serverless

- 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



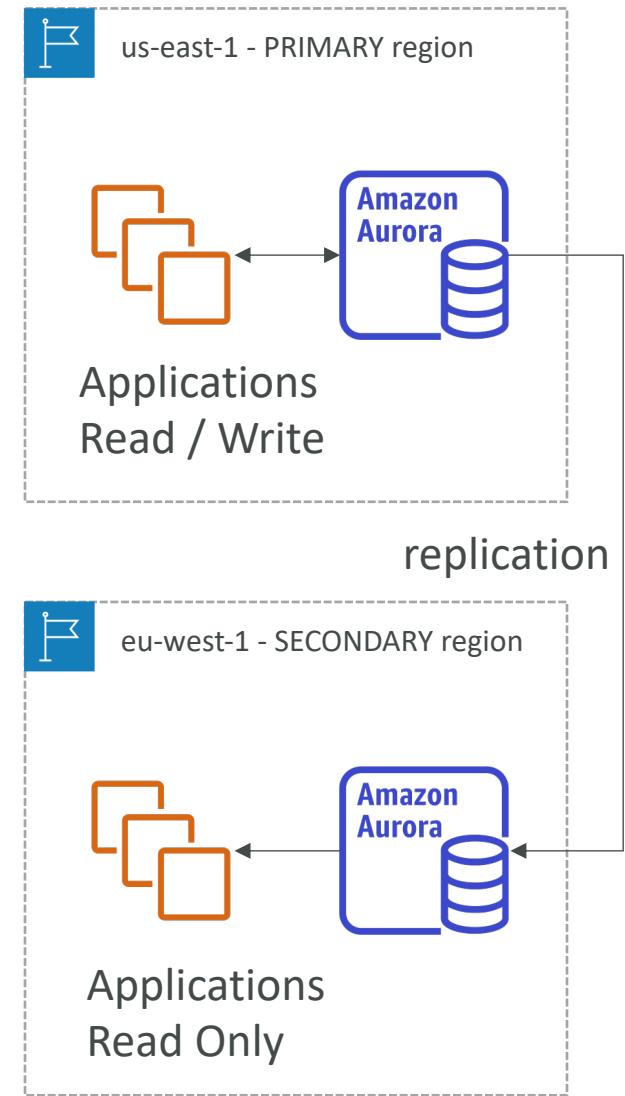
# Aurora Multi-Master

- In case you want immediate failover for write node (HA) –
- Every node does R/W - vs promoting a RR 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



# 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





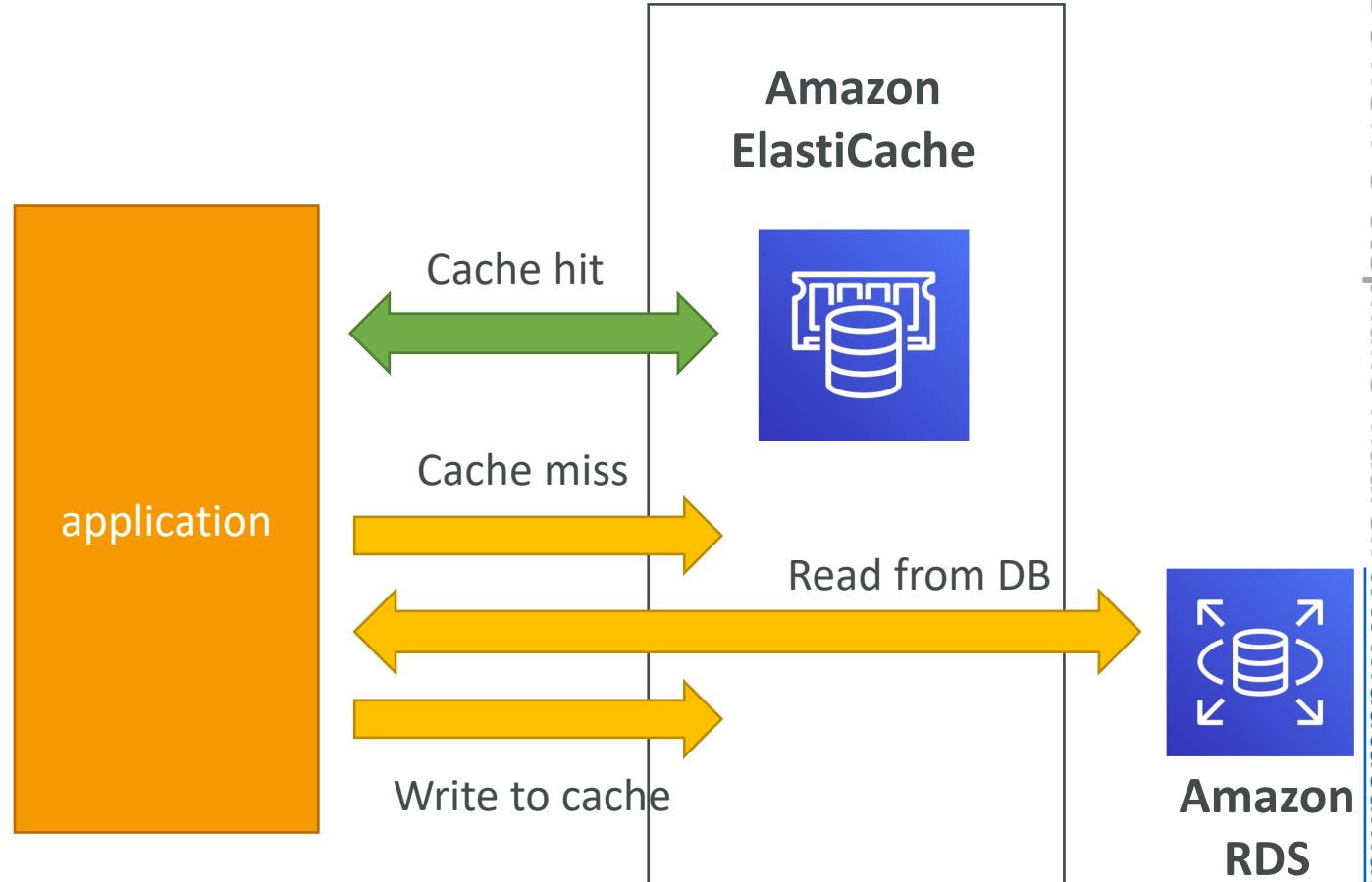
# Amazon ElastiCache Overview

- 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

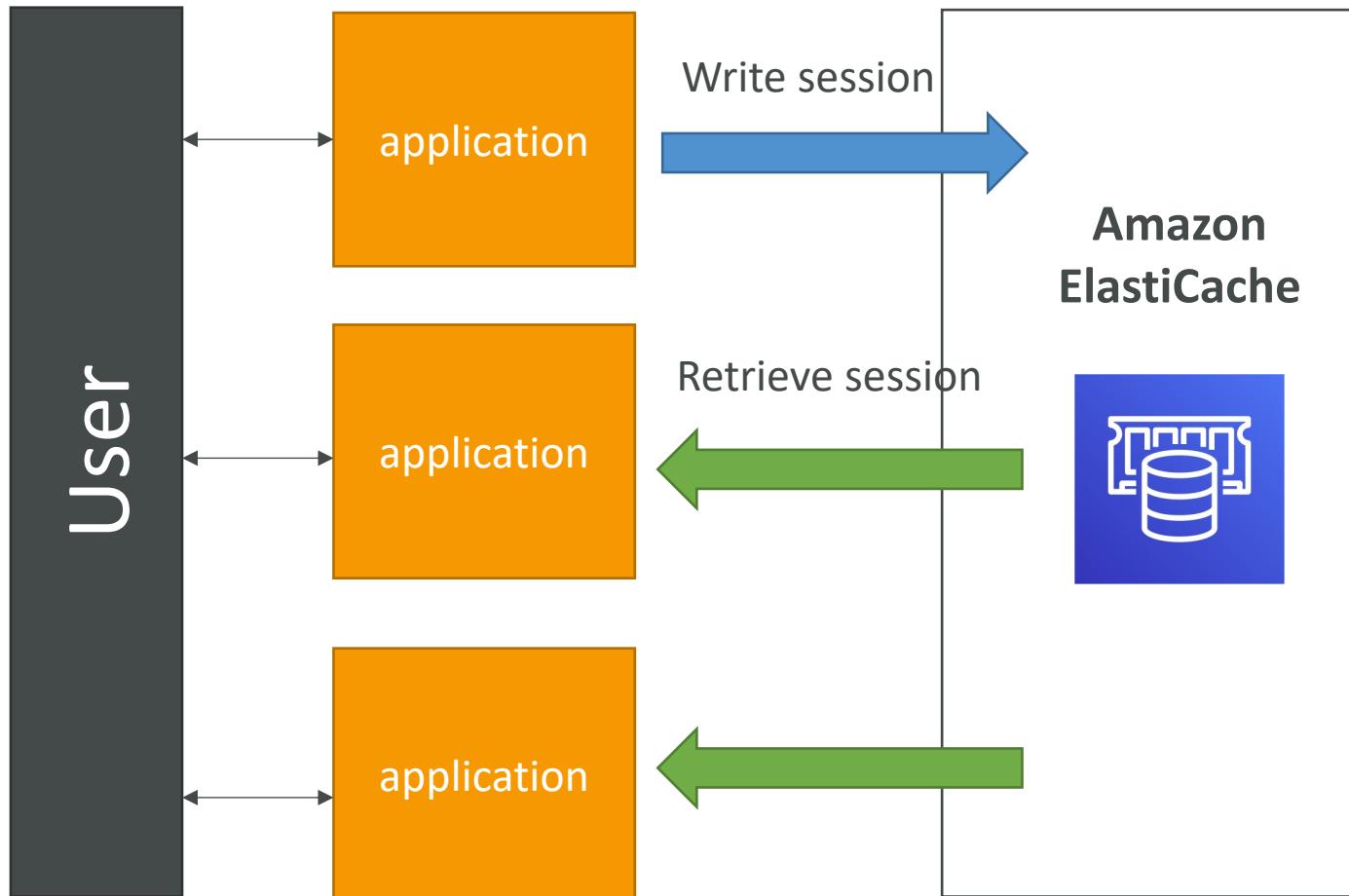
- Applications queries ElastiCache, if not available, get from RDS and store in ElastiCache.
- Helps relieve load in RDS
- Cache must have an invalidation strategy to make sure only the most current data is used in there.



# 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



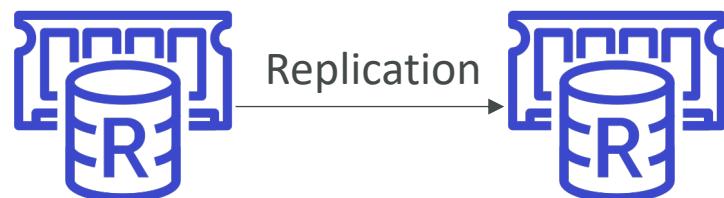
# ElastiCache – Redis vs Memcached

## REDIS

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

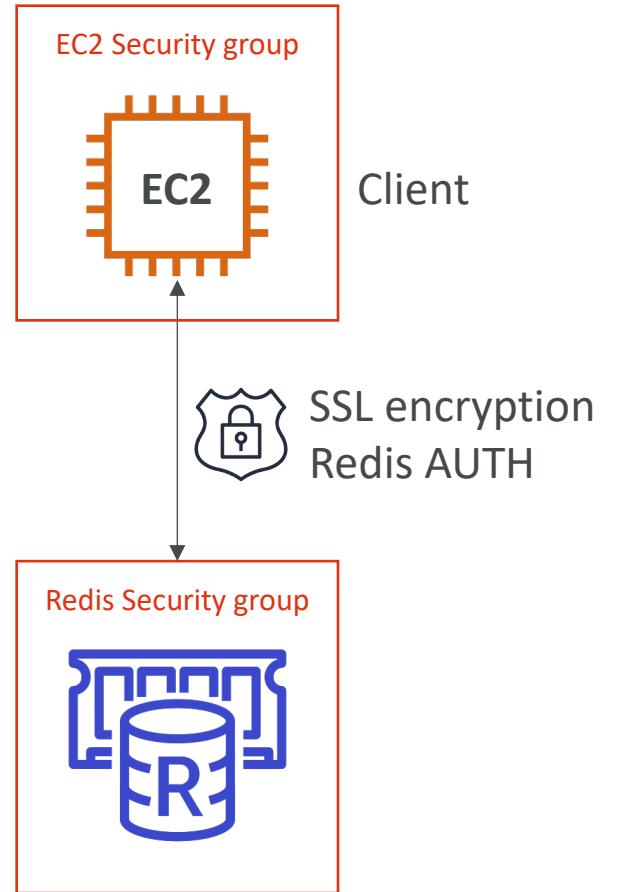
## MEMCACHED

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



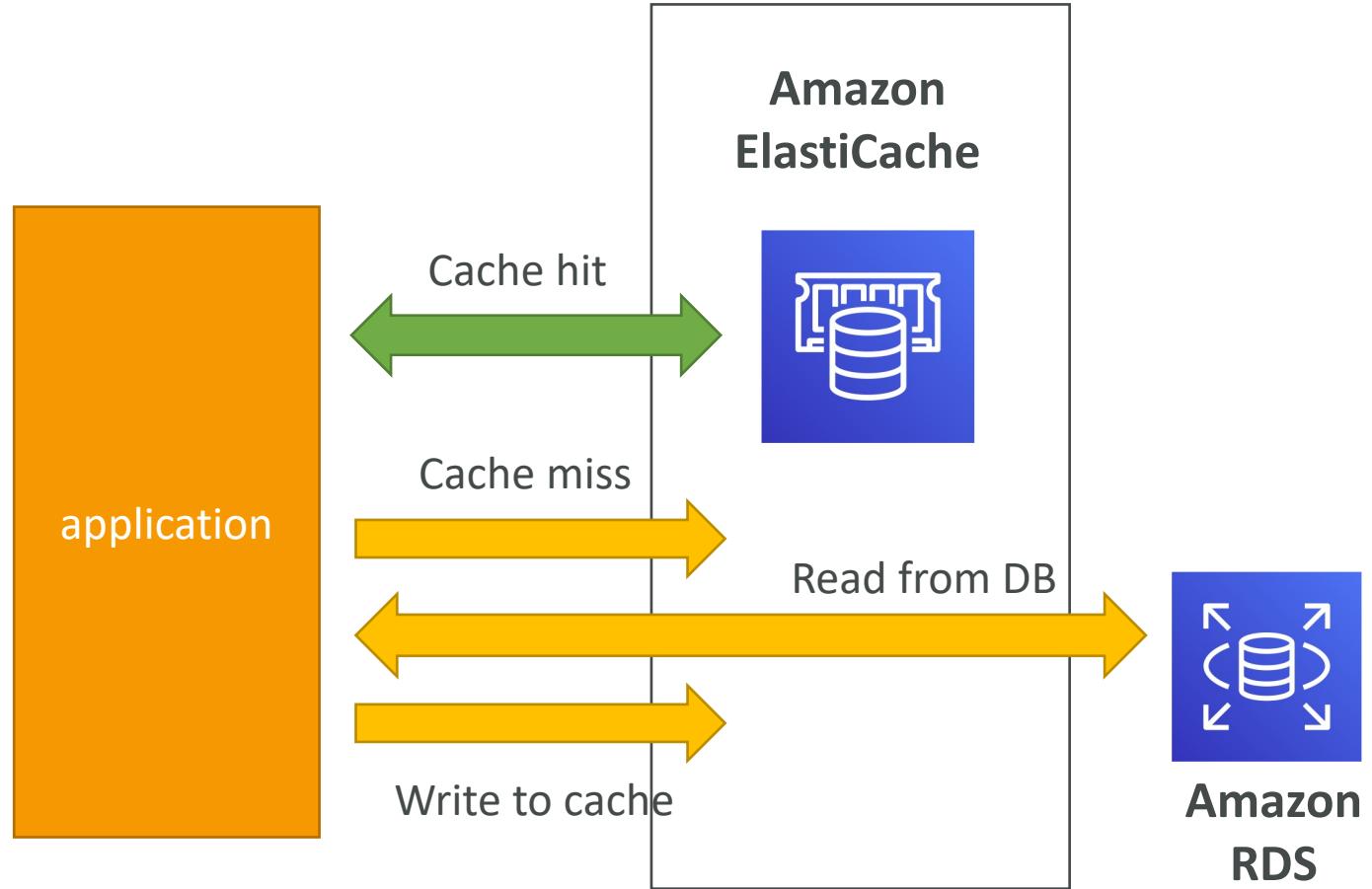
# ElastiCache – Cache Security

- All caches in ElastiCache:
  - Do not support IAM authentication
  - 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

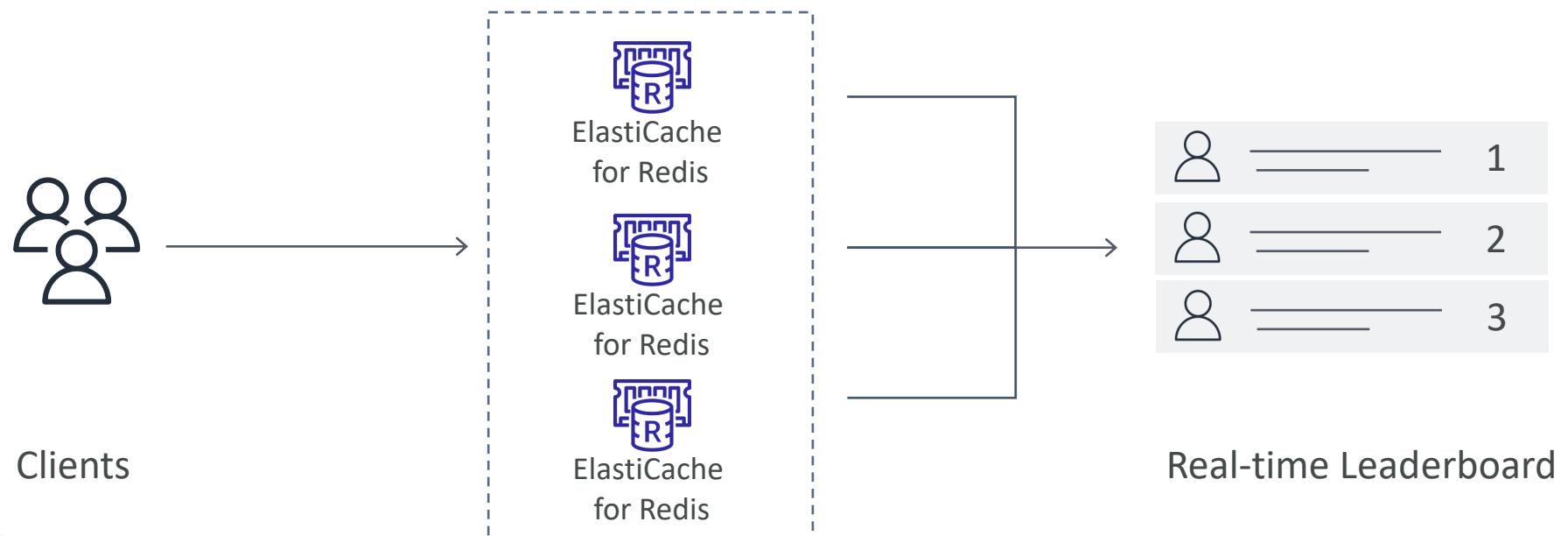
- **Lazy Loading:** all the read data is cached, data can become stale in cache
- **Write Through:** Adds or update data in the cache when written to a DB (no stale data)
- **Session Store:** store temporary session data in a cache (using TTL features)
- **Quote:** There are only two hard things in Computer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

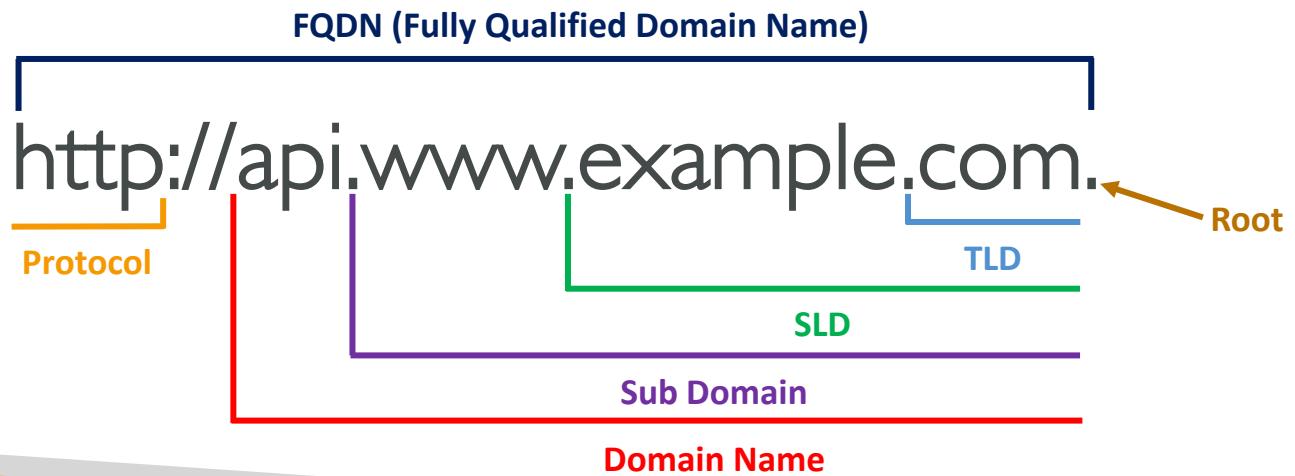
example.com

www.example.com

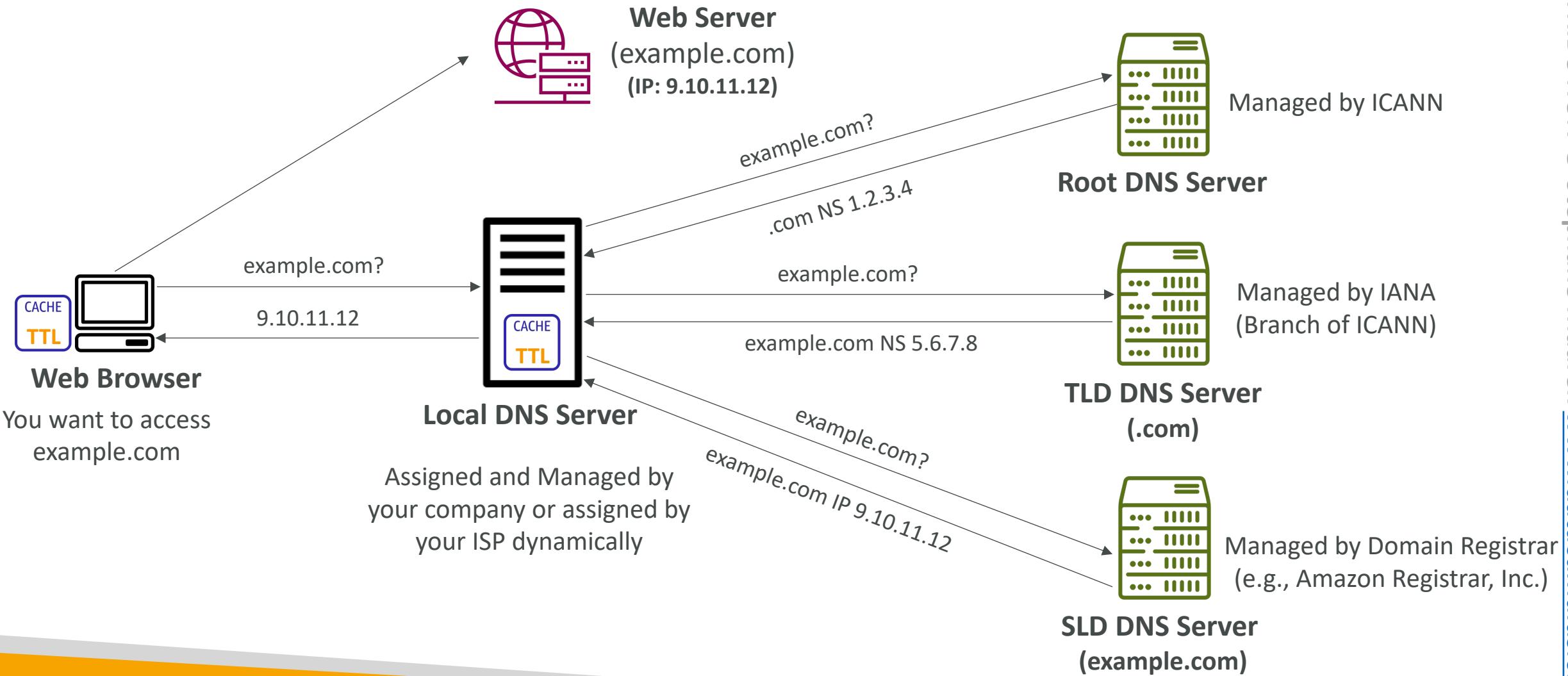
api.example.com

# 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, ...

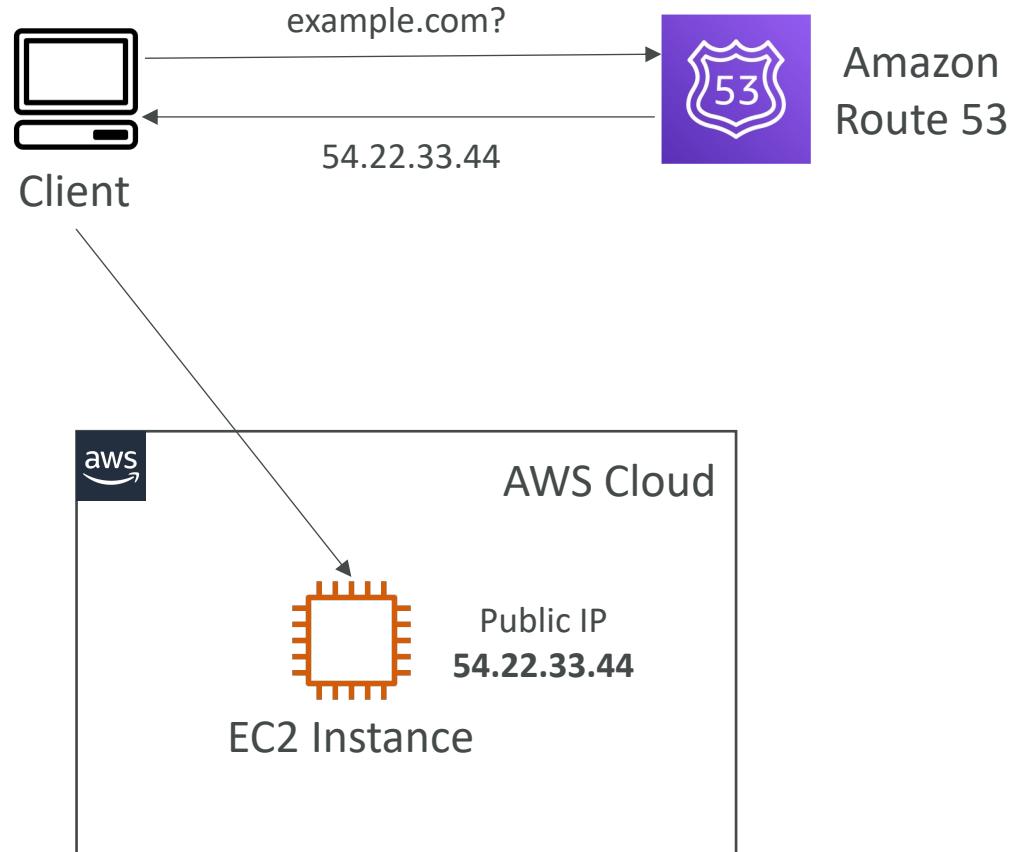


# How DNS Works



# 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

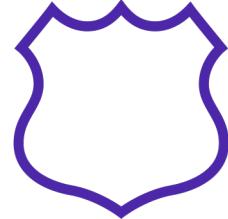


# Route 53 – Records

- 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

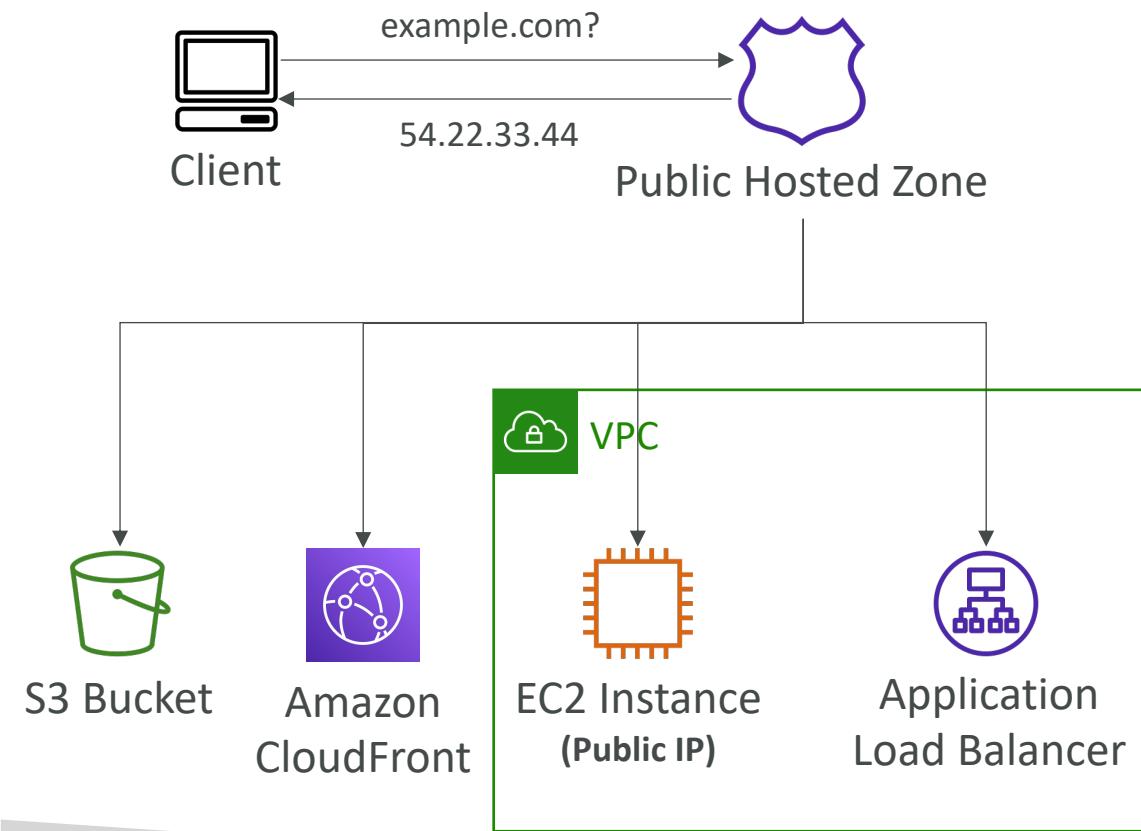


# Route 53 – Hosted Zones

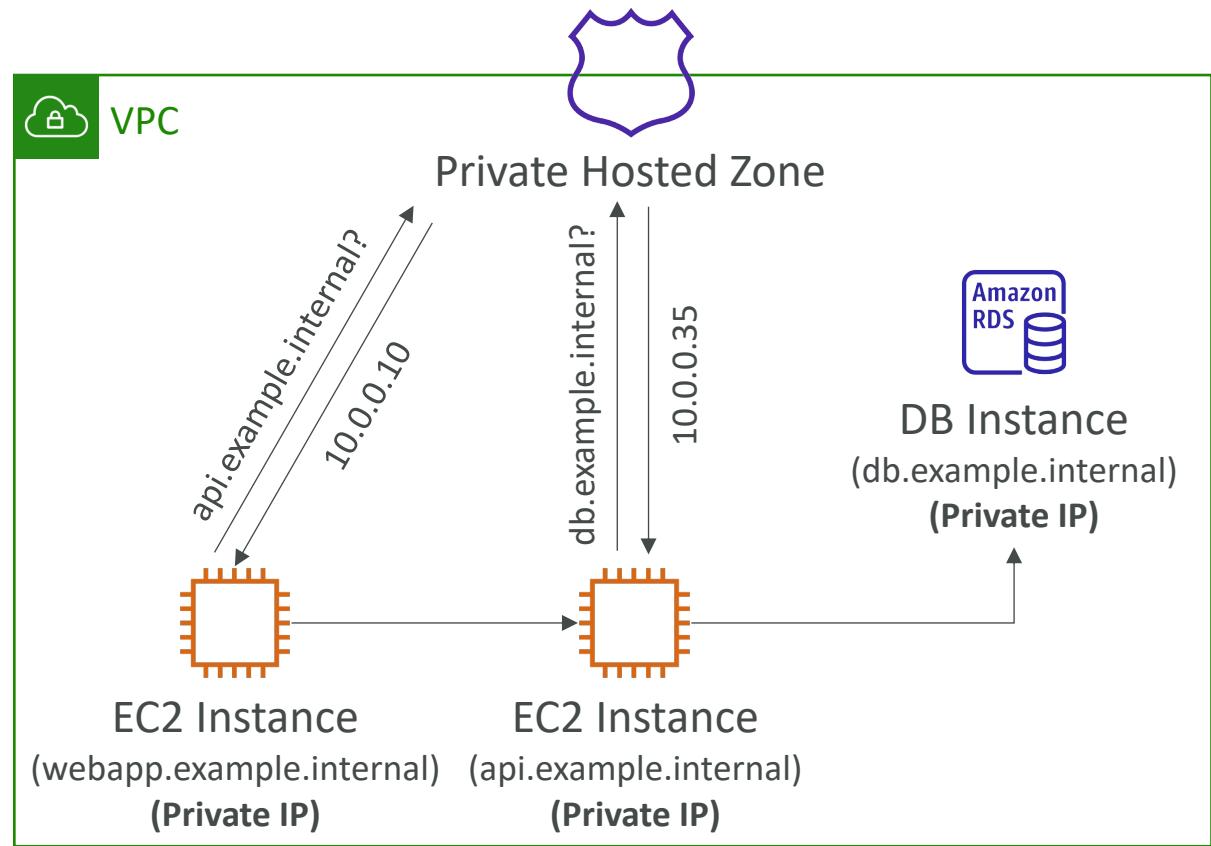
- A container for records that define how to route traffic to a domain and its subdomains
- **Public Hosted Zones** – contains records that specify how to route traffic on the Internet (public domain names)  
[application1.mypublicdomain.com](http://application1.mypublicdomain.com)
- **Private Hosted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

# Route 53 – Public vs. Private Hosted Zones

## Public Hosted Zone

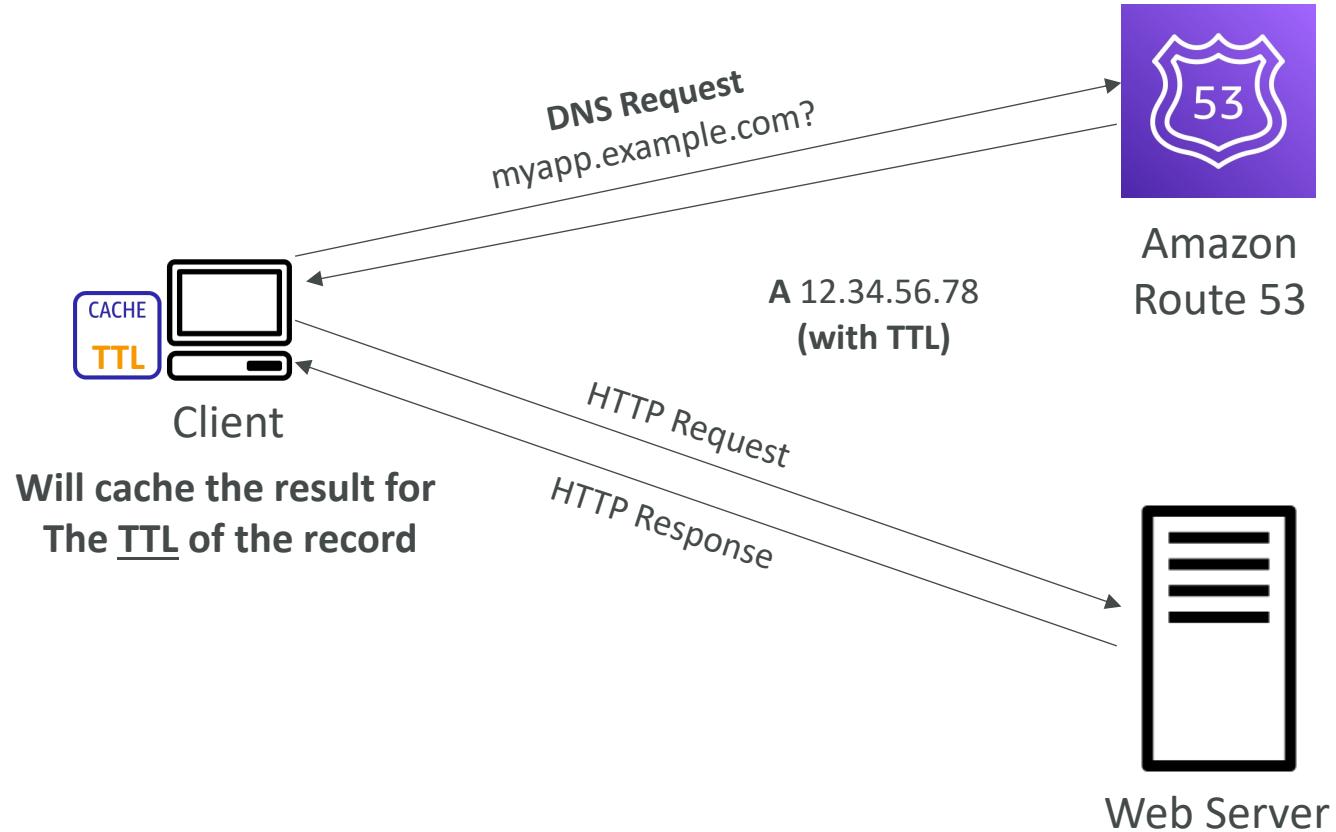


## Private Hosted Zone



# 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:
  - [lb-1234.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

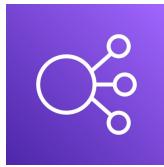
# Route 53 – Alias Records

- 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

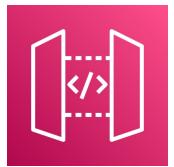
- Elastic Load Balancers
- CloudFront Distributions
- API Gateway
- Elastic Beanstalk environments
- S3 Websites
- VPC Interface Endpoints
- Global Accelerator accelerator
- Route 53 record in the same hosted zone
- You cannot set an ALIAS record for an EC2 DNS name



Elastic  
Load Balancer



Amazon  
CloudFront



Amazon  
API Gateway



Elastic Beanstalk



S3 Websites



VPC Interface  
Endpoints



Global Accelerator



Route 53 Record  
(same Hosted Zone)

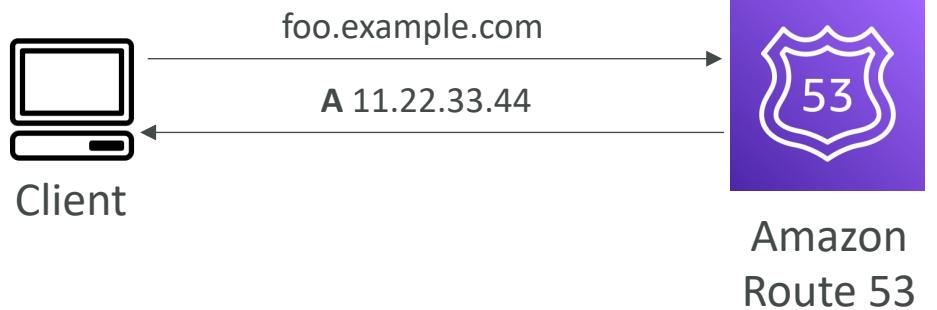
# Route 53 – Routing Policies

- 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)

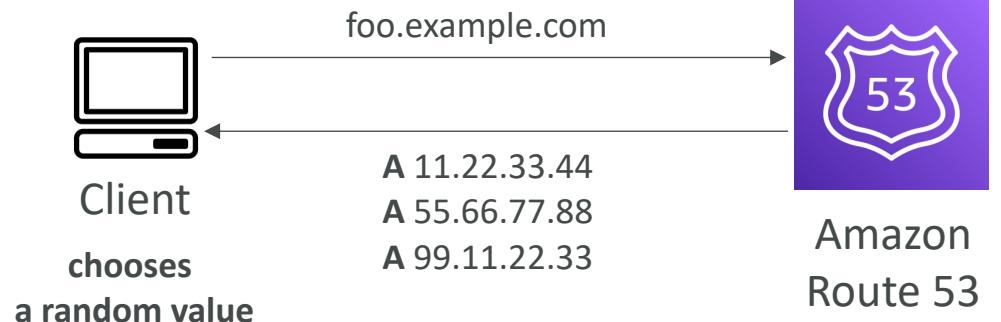
# Routing Policies – Simple

- 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

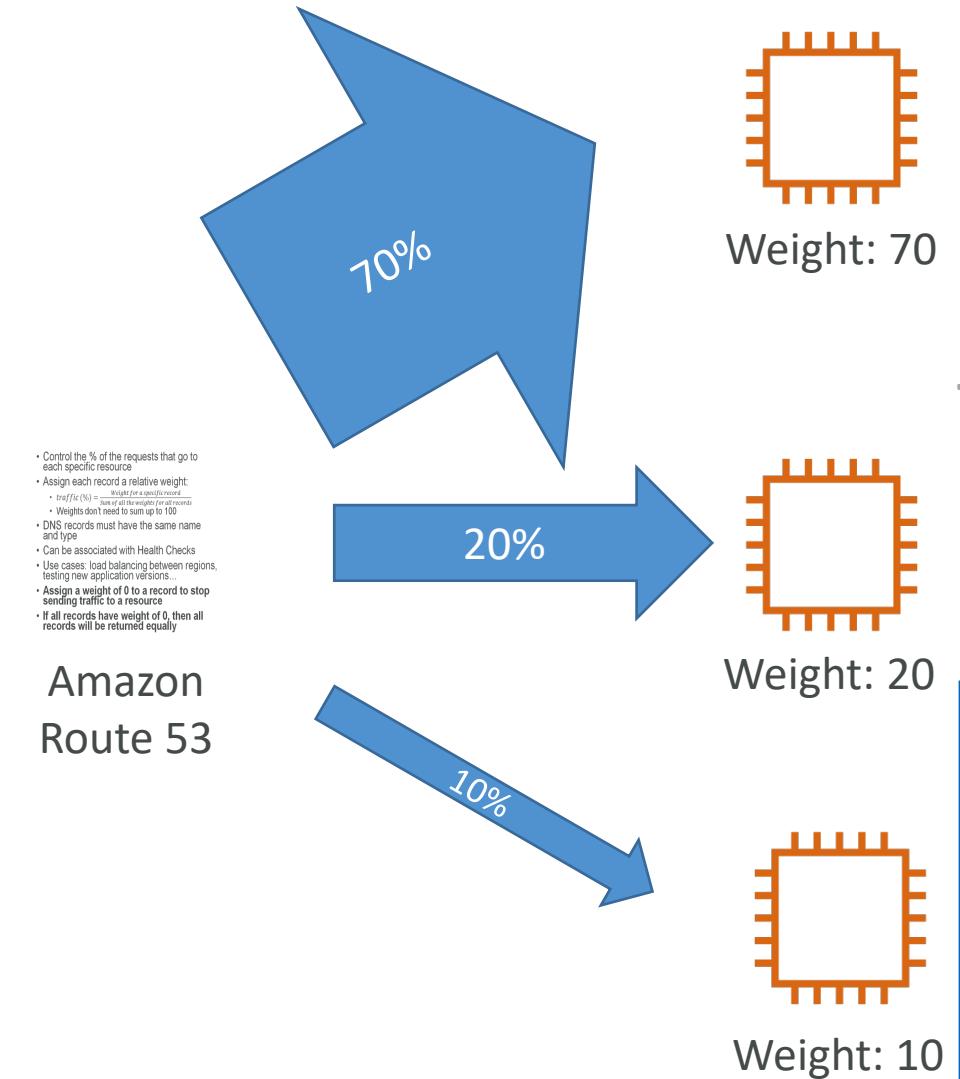


## Multiple Value



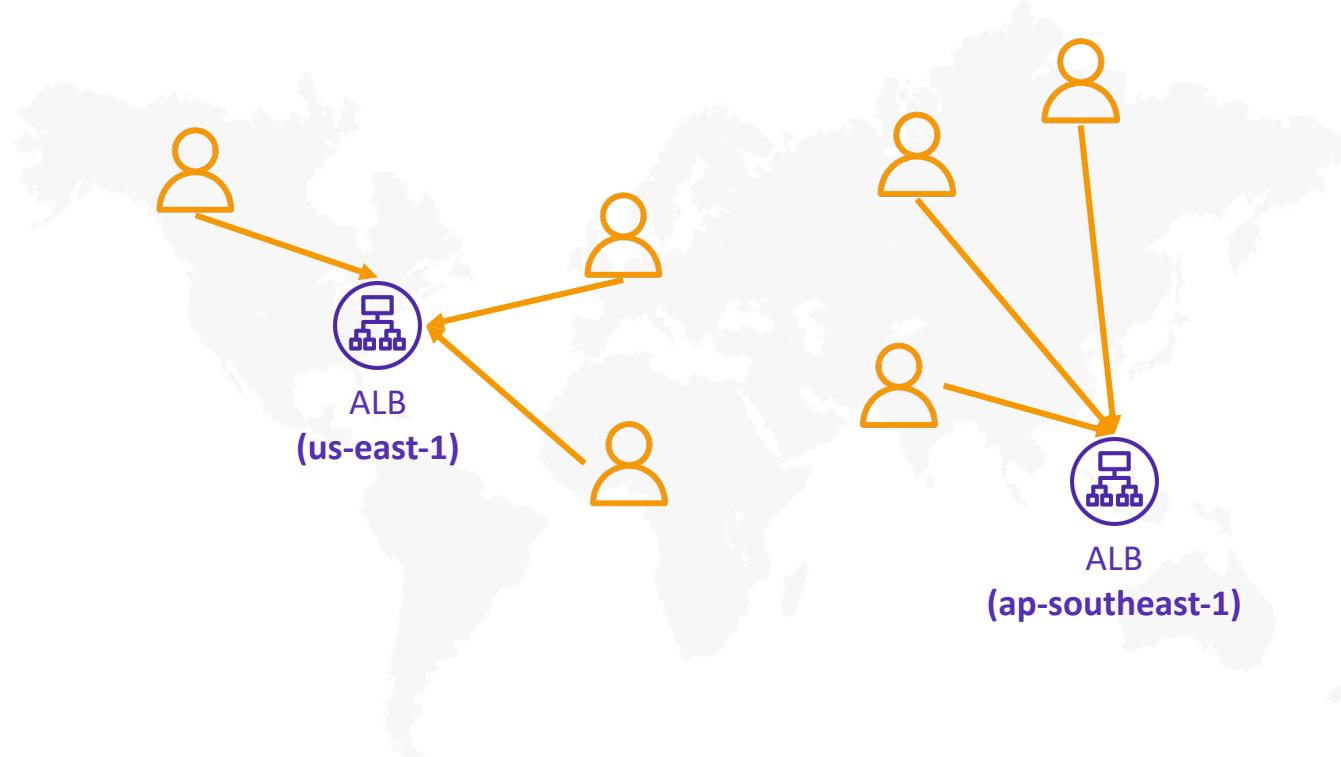
# Routing Policies – Weighted

- 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



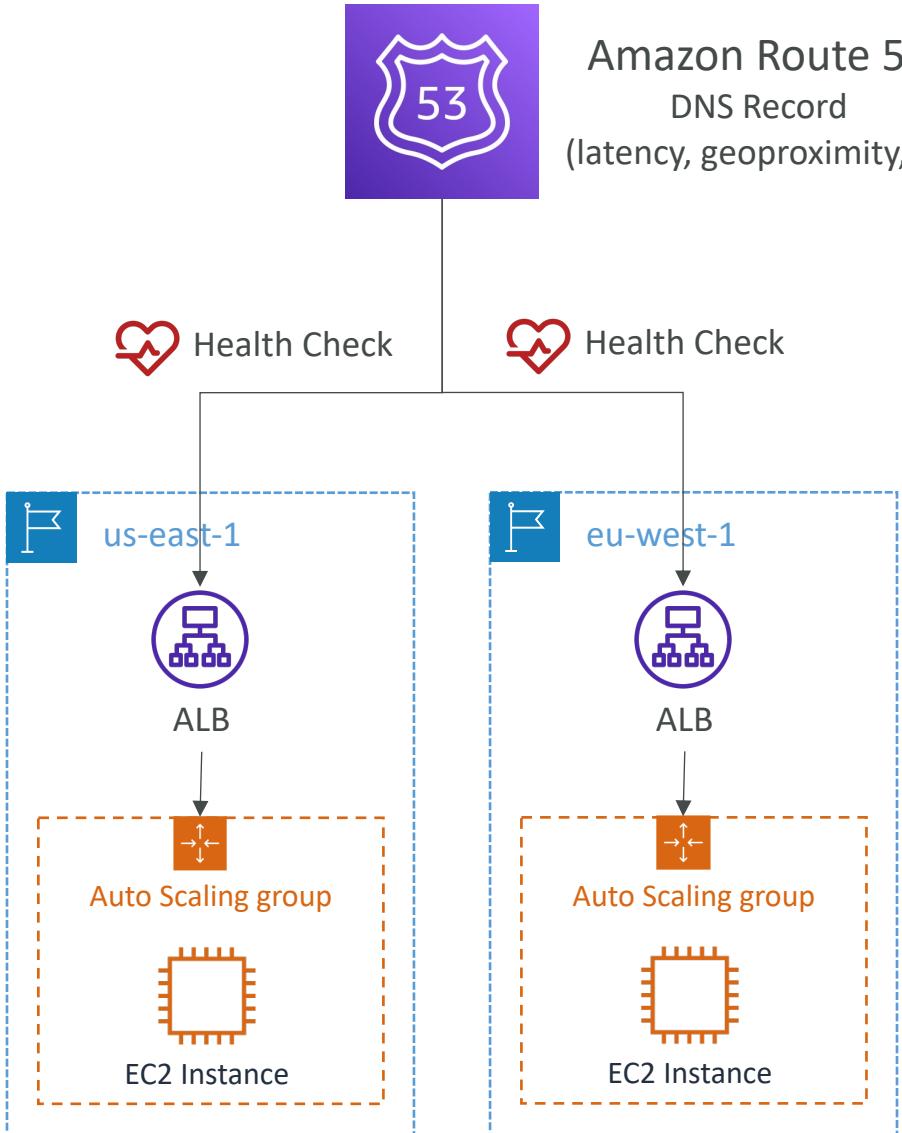
# Routing Policies – Latency-based

- 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)



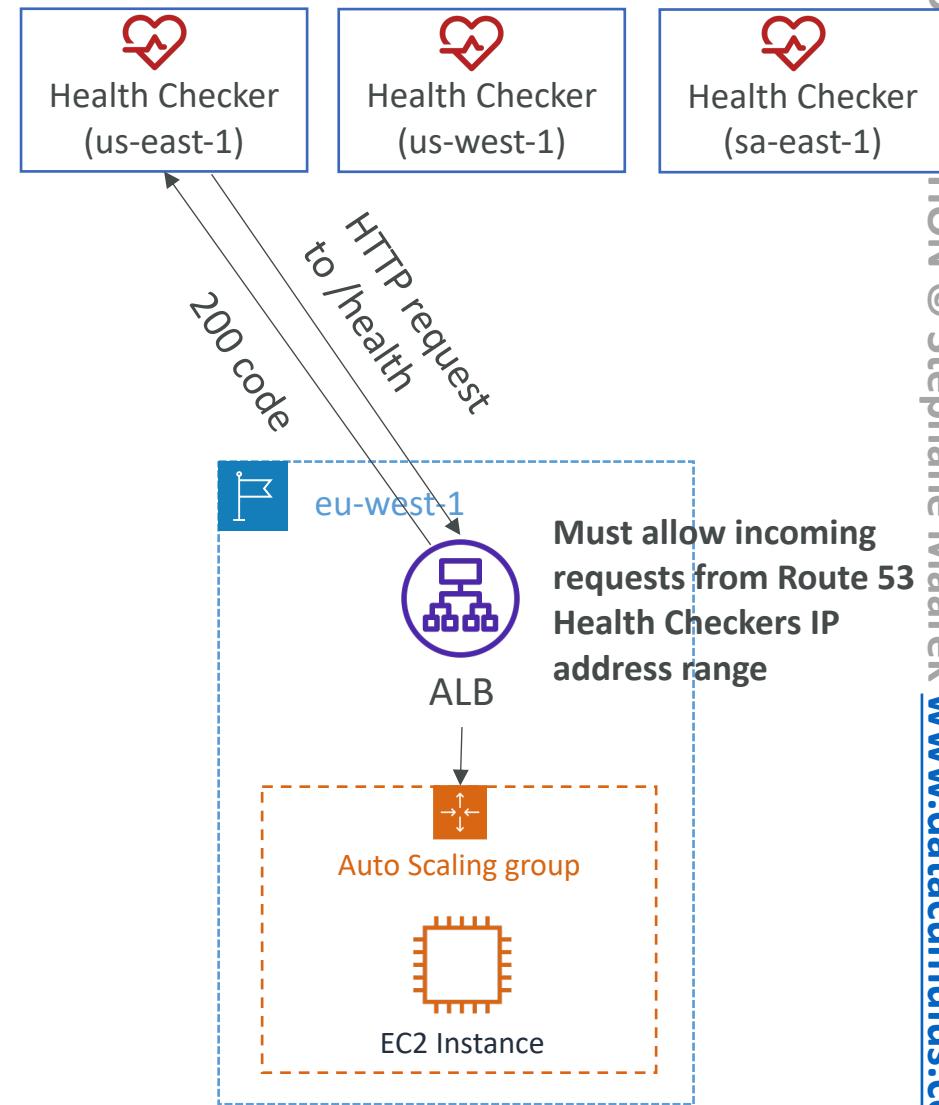
# Route 53 – Health Checks

- 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



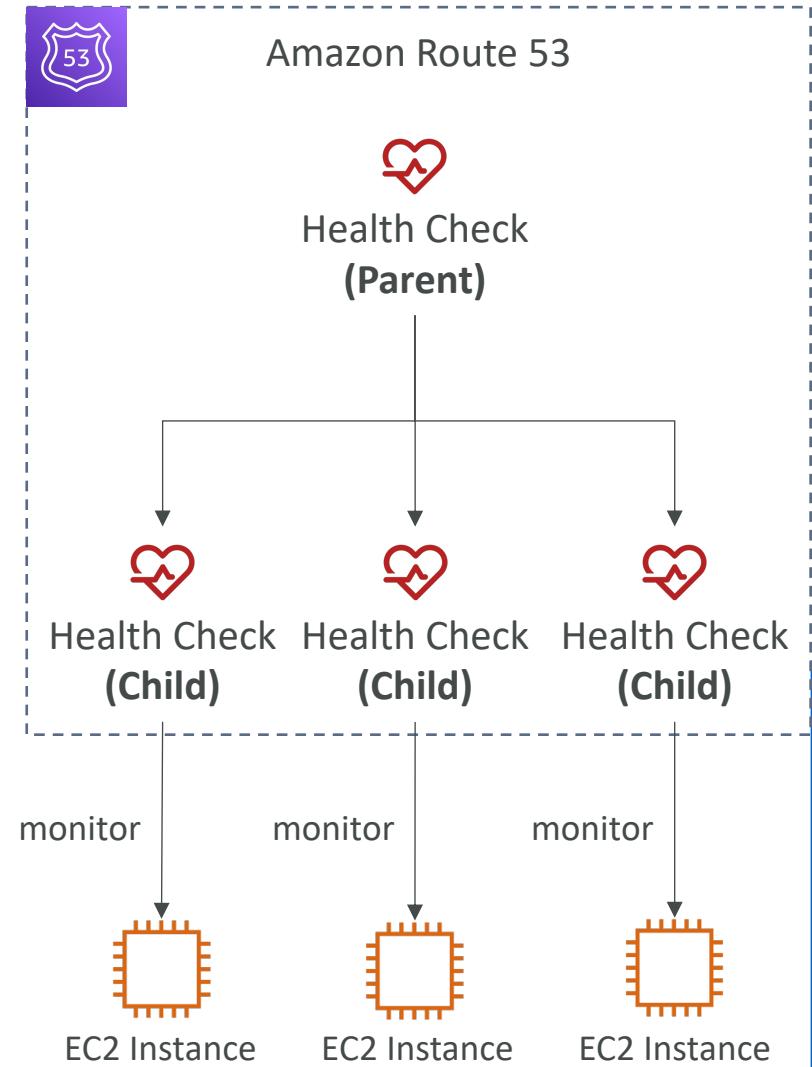
# Health Checks – Monitor an Endpoint

- 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



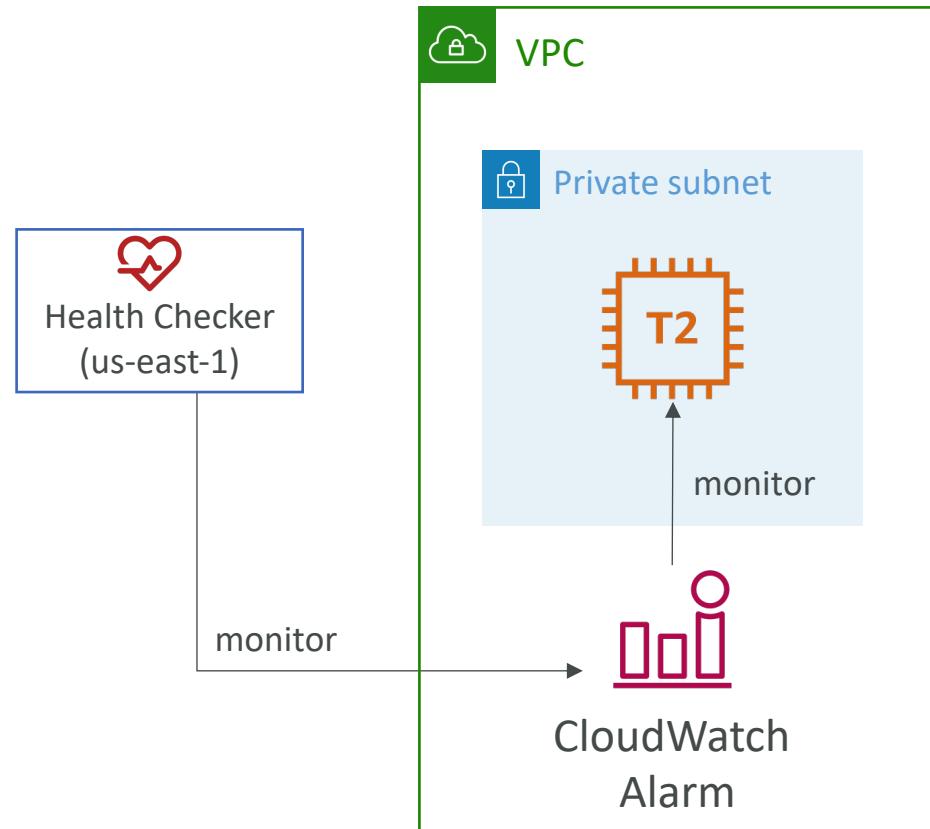
# Route 53 – Calculated Health Checks

- 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

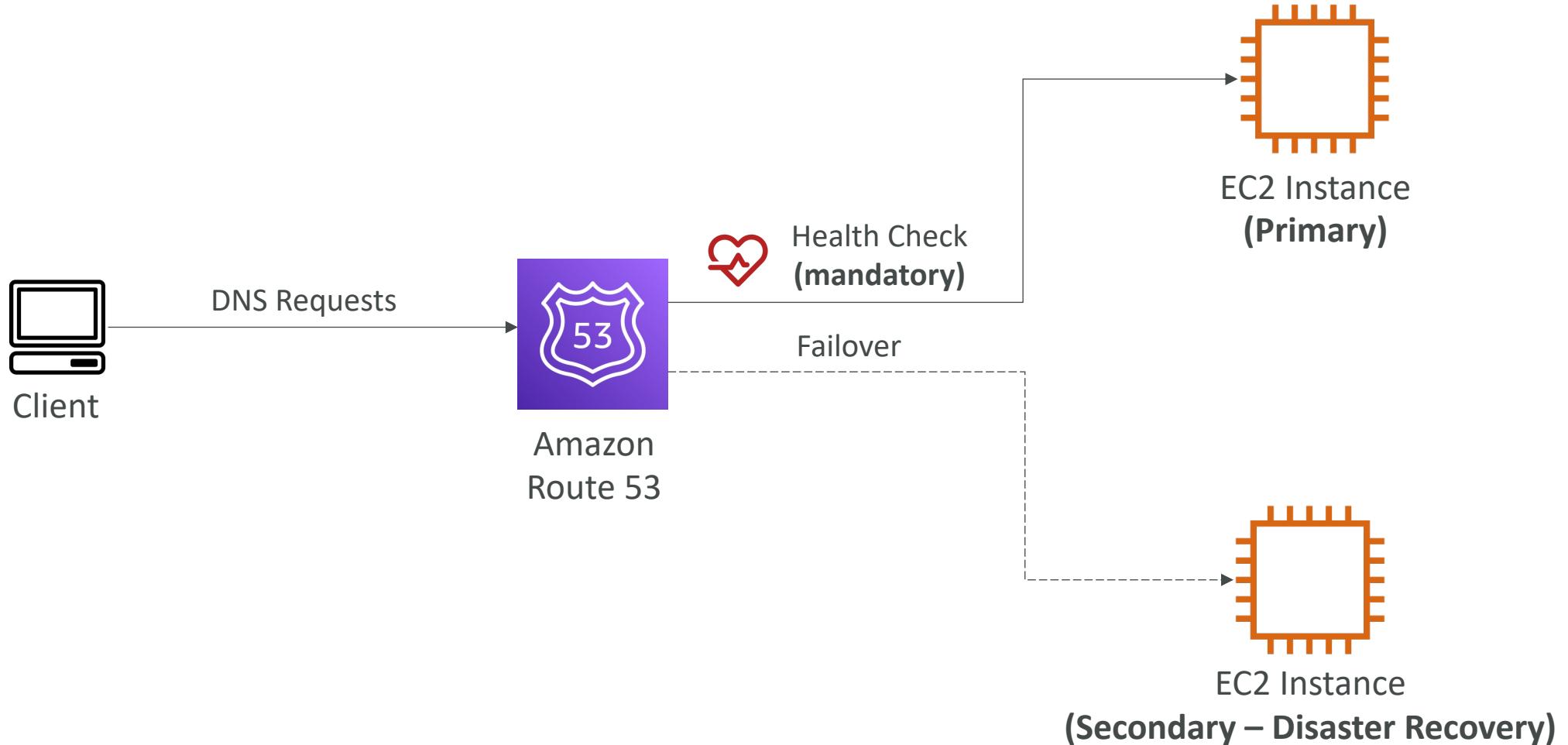


# Health Checks – Private Hosted Zones

- 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

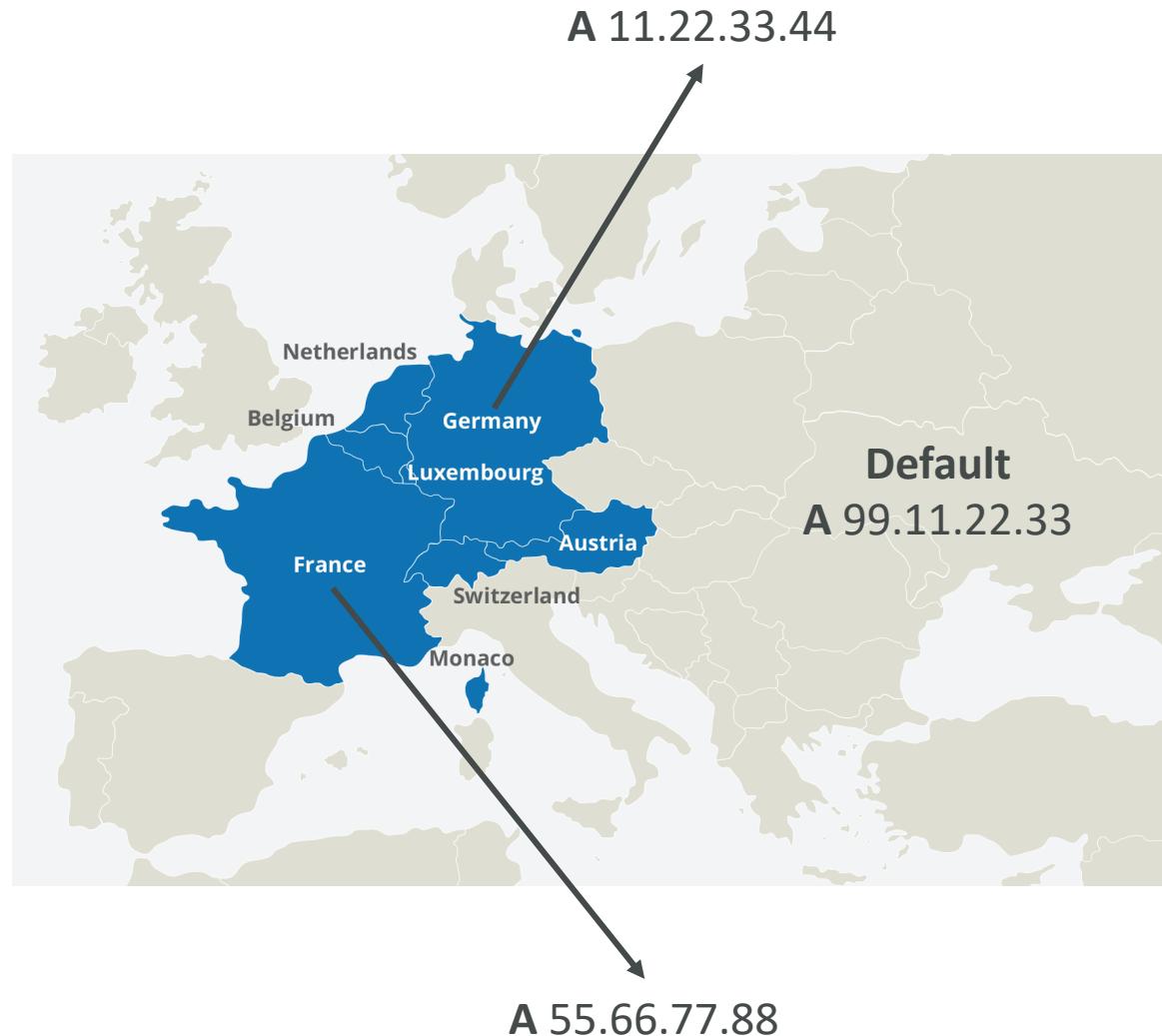


# Routing Policies – Failover (Active-Passive)



# Routing Policies – Geolocation

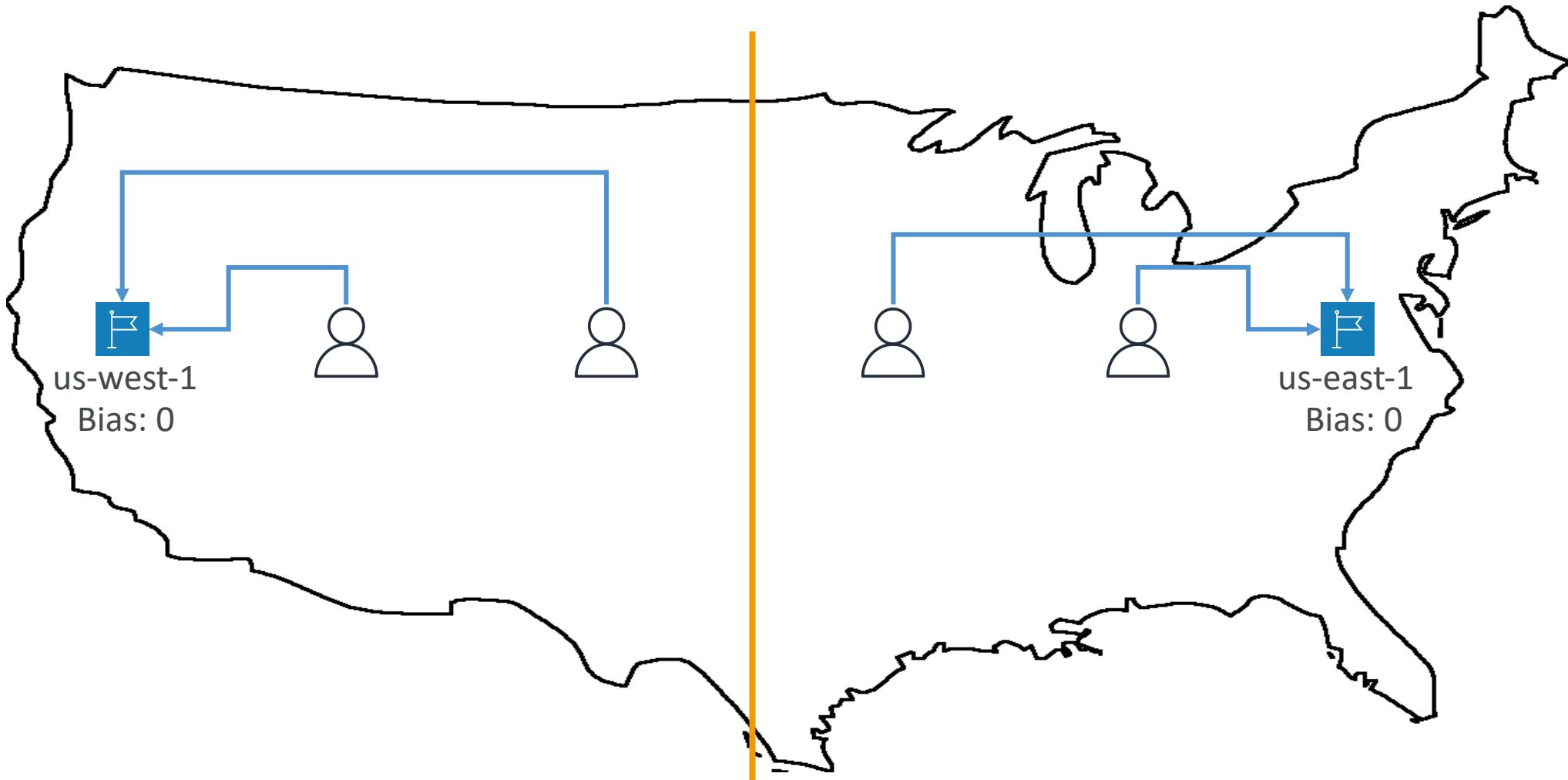
- Different from Latency-based!
- This routing is based on user location
- Specify location by Continent, Country or by US State (if there's overlapping, most precise location selected)
- Should create a “Default” record (in case there's no match on location)
- Use cases: website localization, restrict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



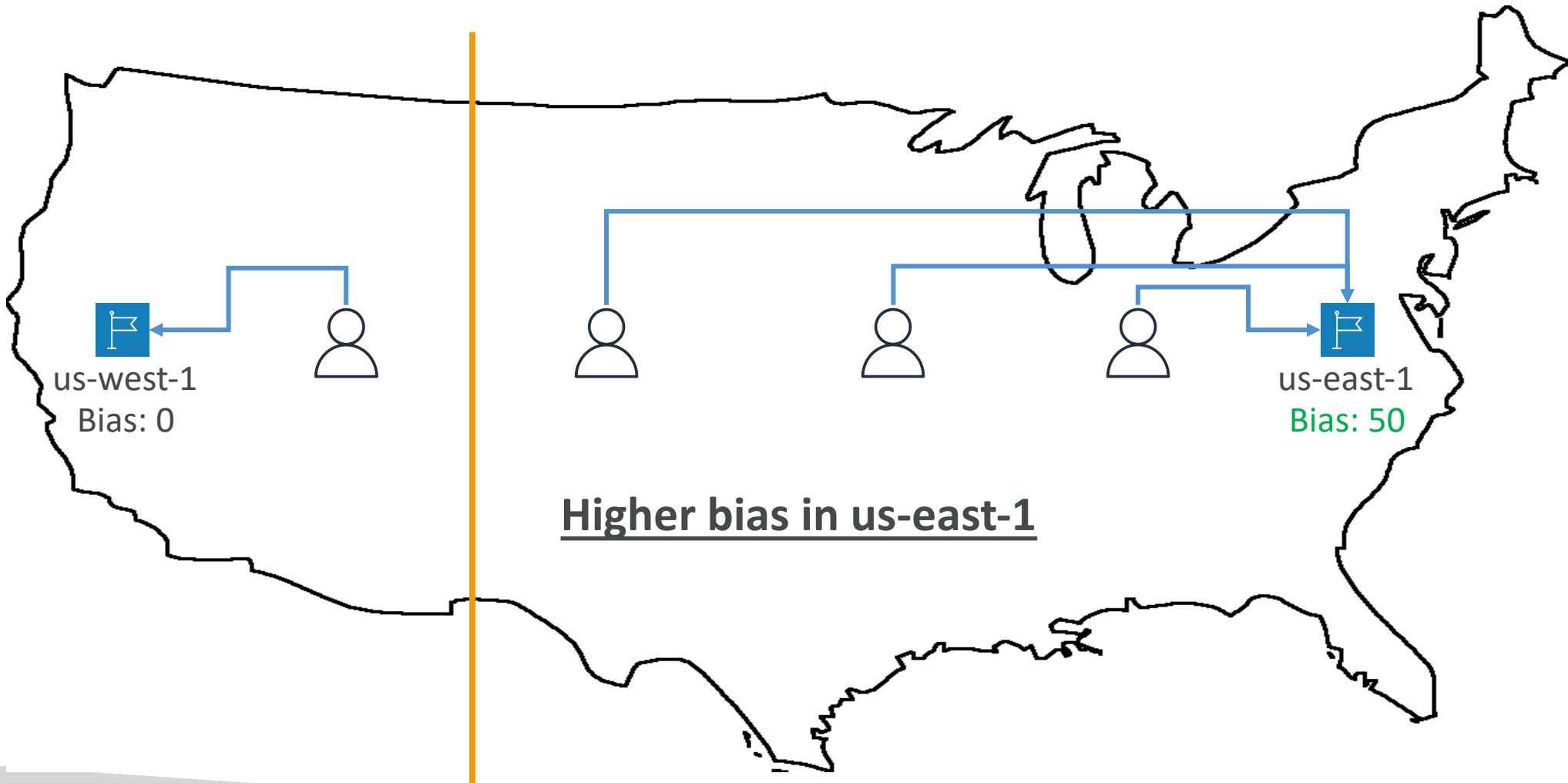
# Routing Policies – Geoproximity

- 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

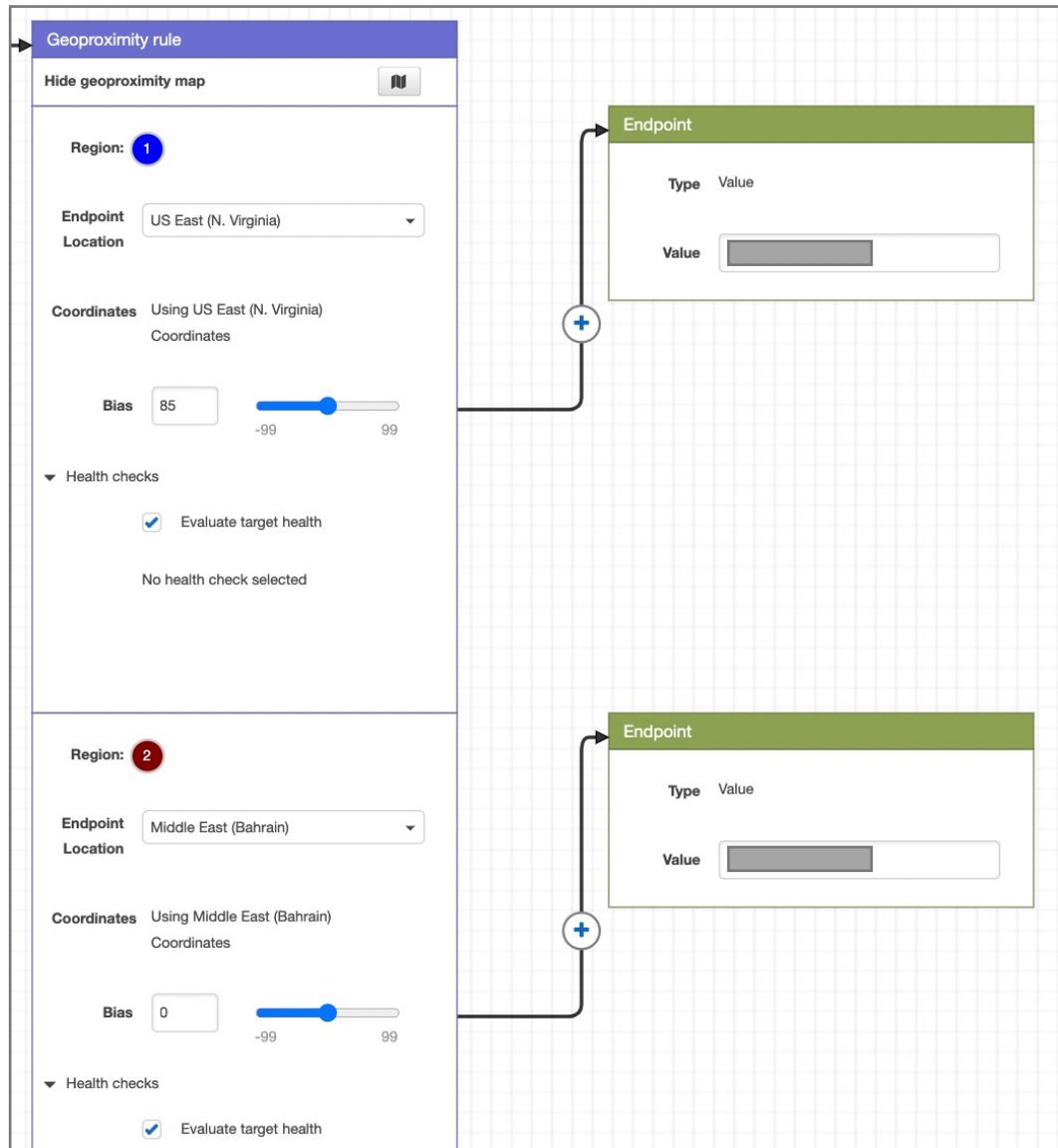


# Routing Policies – Geoproximity



# Route 53 – Traffic flow

- Simplify the process of creating and maintaining records in large and complex configurations
- Visual editor to manage complex routing decision trees
- Configurations can be saved as **Traffic Flow Policy**
  - Can be applied to different Route 53 Hosted Zones (different domain names)
  - Supports versioning



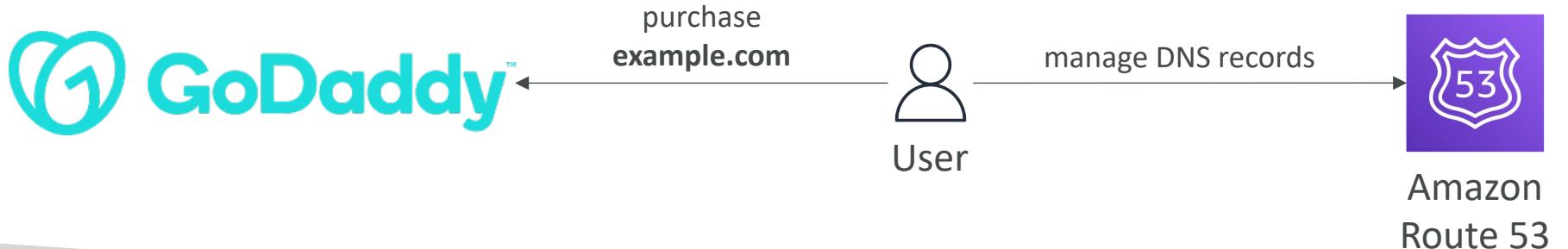
# Routing Policies – Multi-Value

- 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		

# 3<sup>rd</sup> Party Registrar with Amazon Route 53

- If you buy your domain on a 3<sup>rd</sup> 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 3<sup>rd</sup> 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

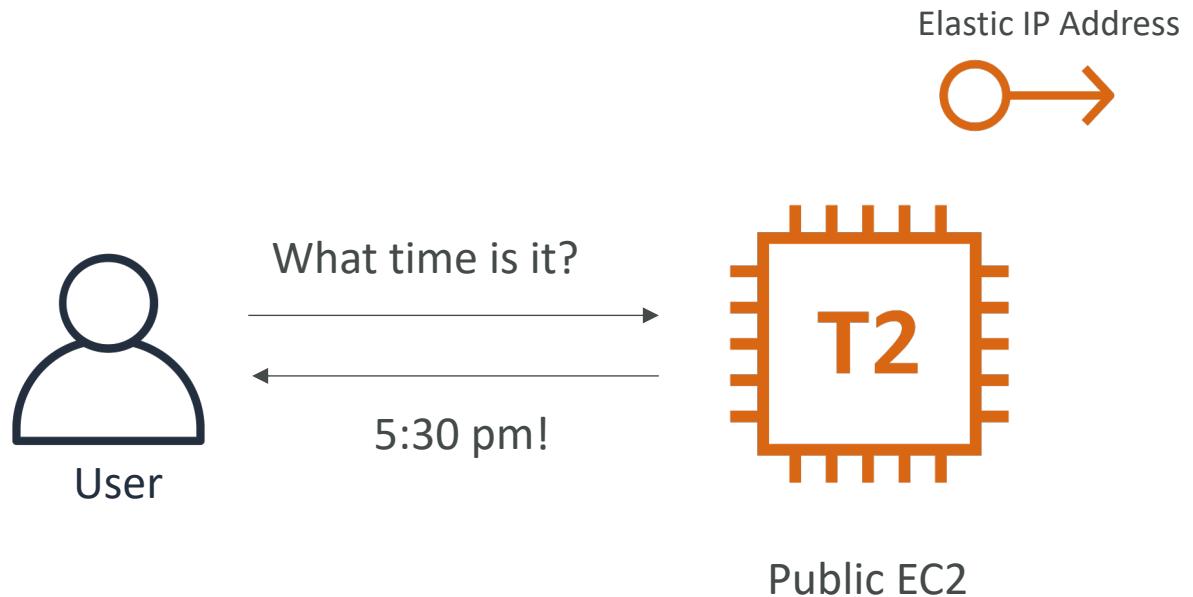
- These solutions architectures are the best part of this course
- Let's understand how all the technologies we've seen work together
- This is a section you need to be 100% comfortable with
- We'll see the progression of a Solution's architect mindset through many sample case studies:
  - WhatIsTheTime.Com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

# Stateless Web App: WhatIsTheTime.com

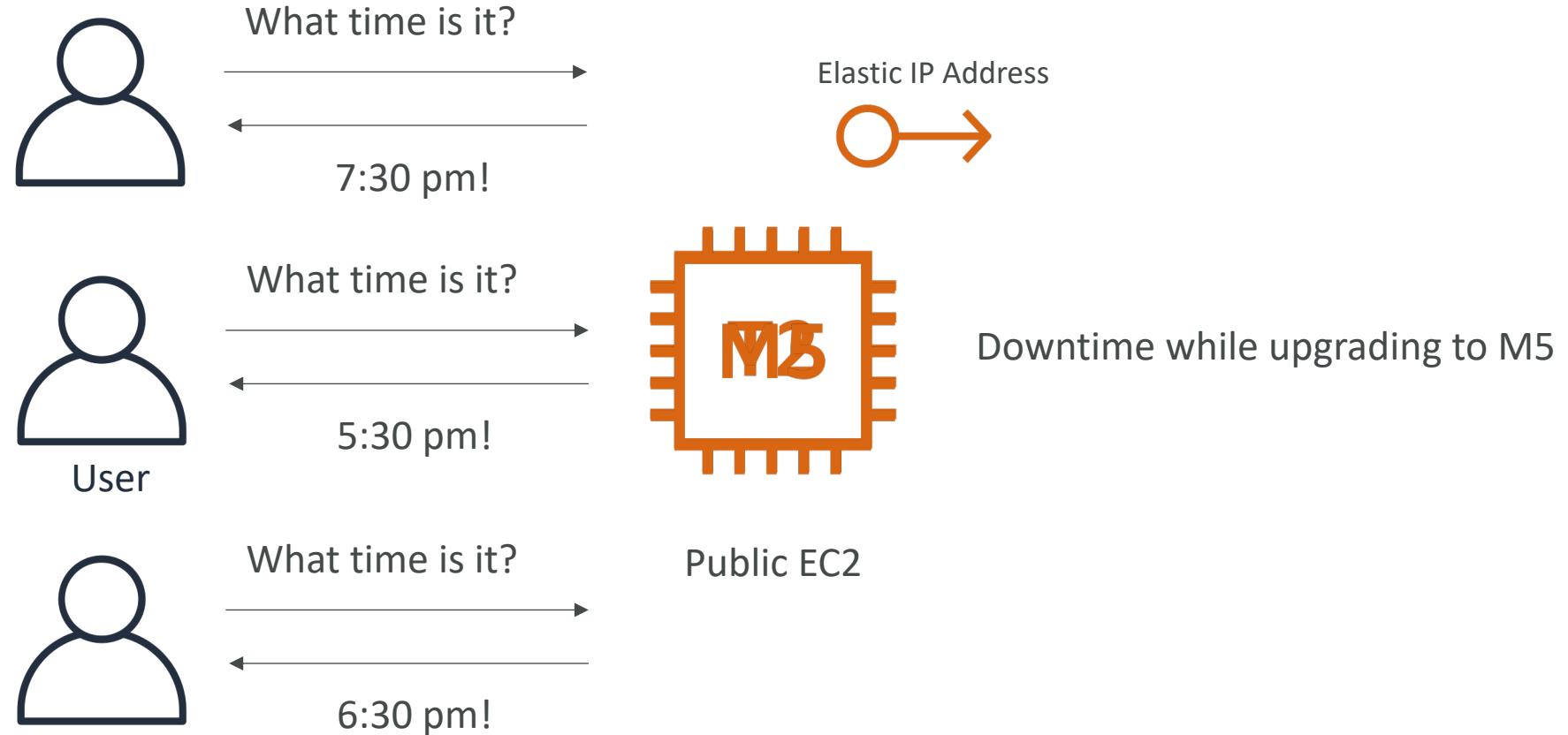
- WhatIsTheTime.com allows people to know what time it is
- We don't need a database
- We want to start small and can accept downtime
- We want to fully scale vertically and horizontally, no downtime
- Let's go through the Solutions Architect journey for this app
  
- Let's see how we can proceed!

# Stateless web app: What time is it?

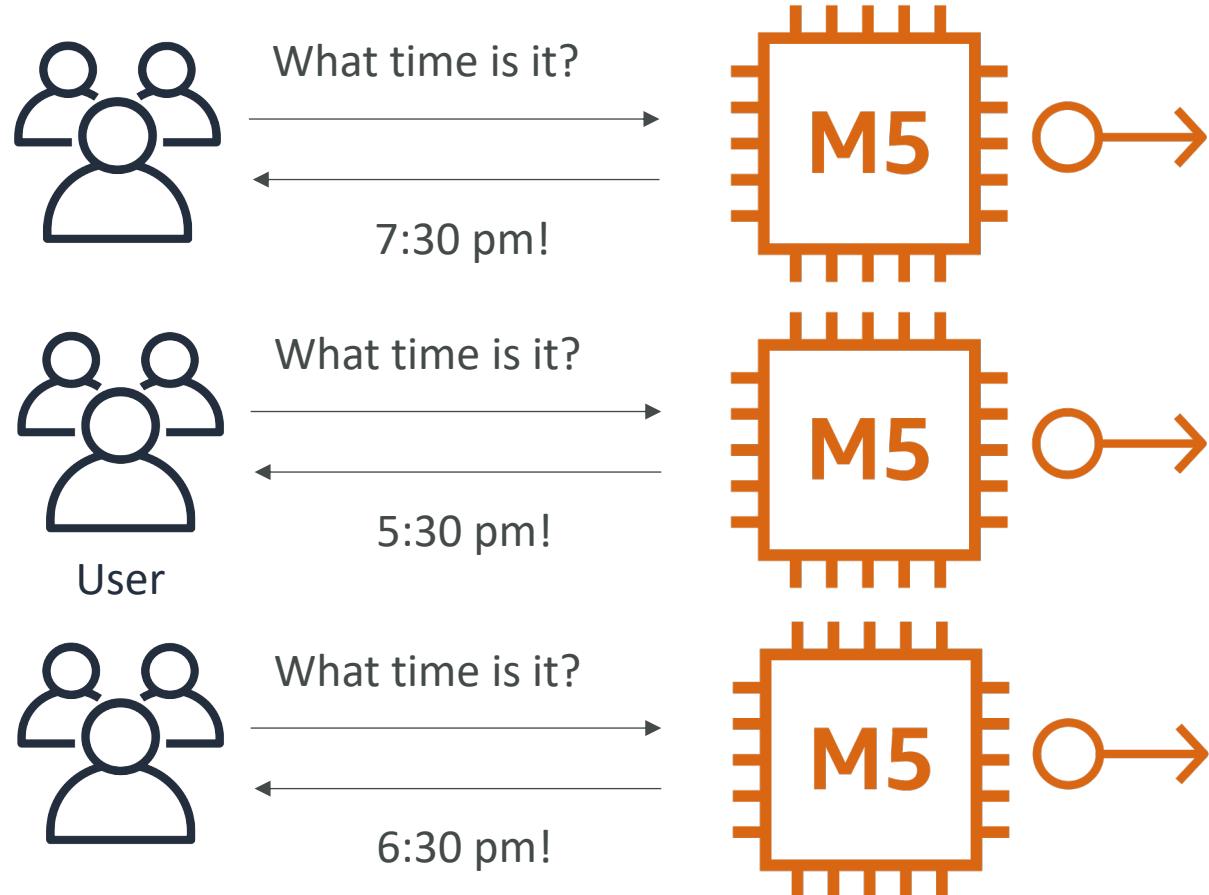
## Starting simple



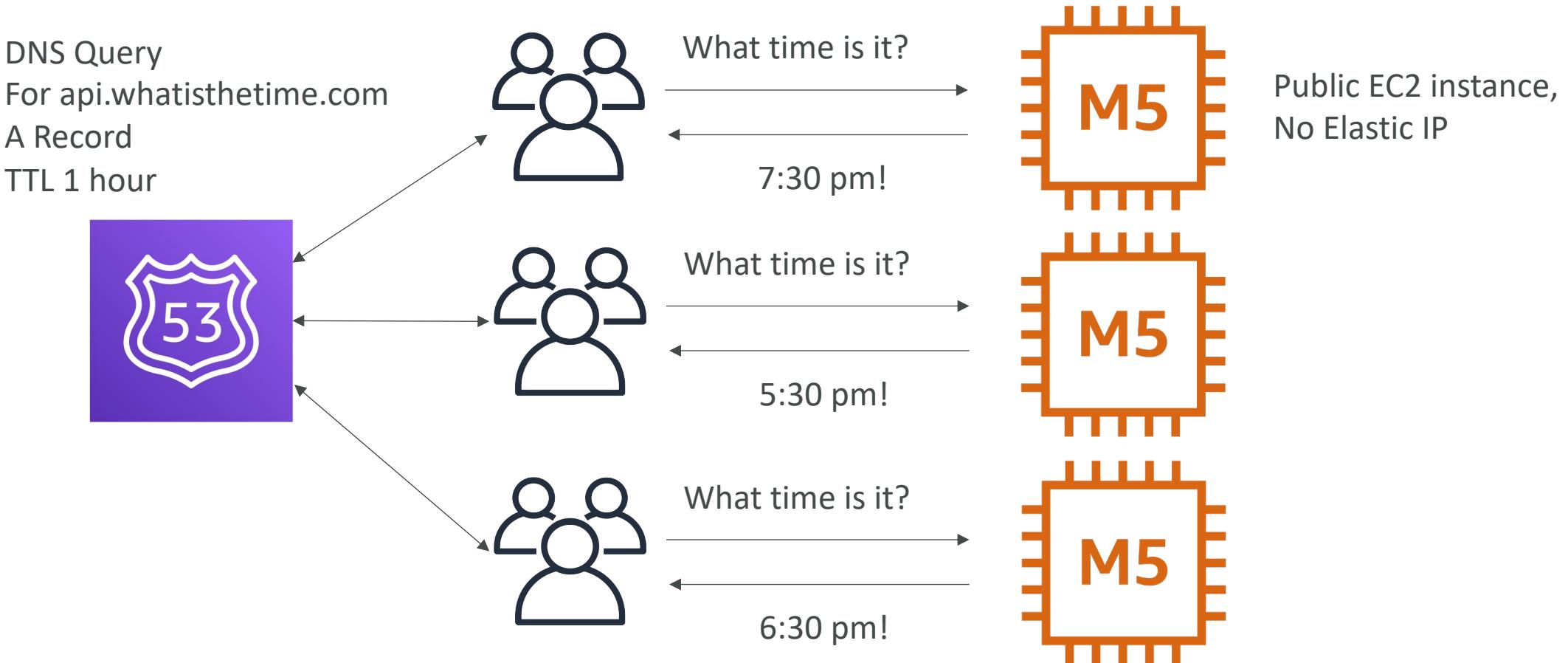
# Stateless web app: What time is it? Scaling vertically



# Stateless web app: What time is it? Scaling horizontally

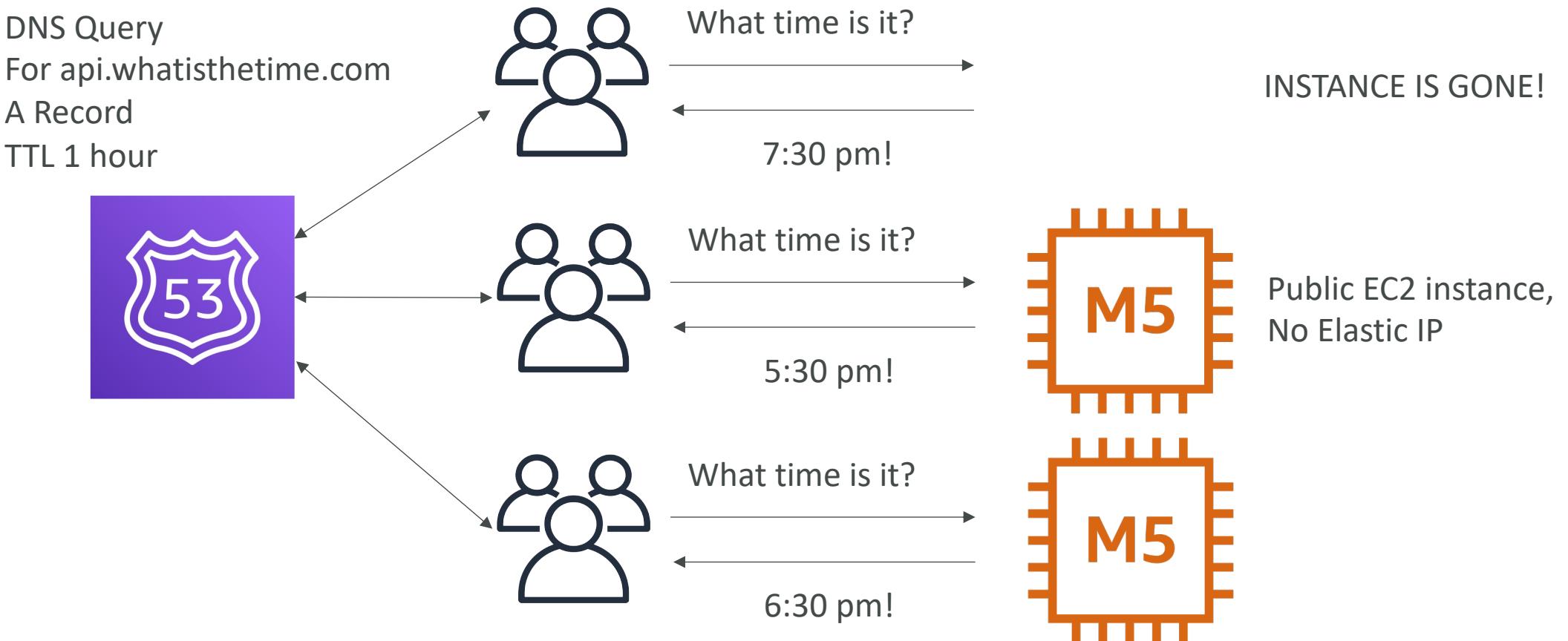


# Stateless web app: What time is it? Scaling horizontally

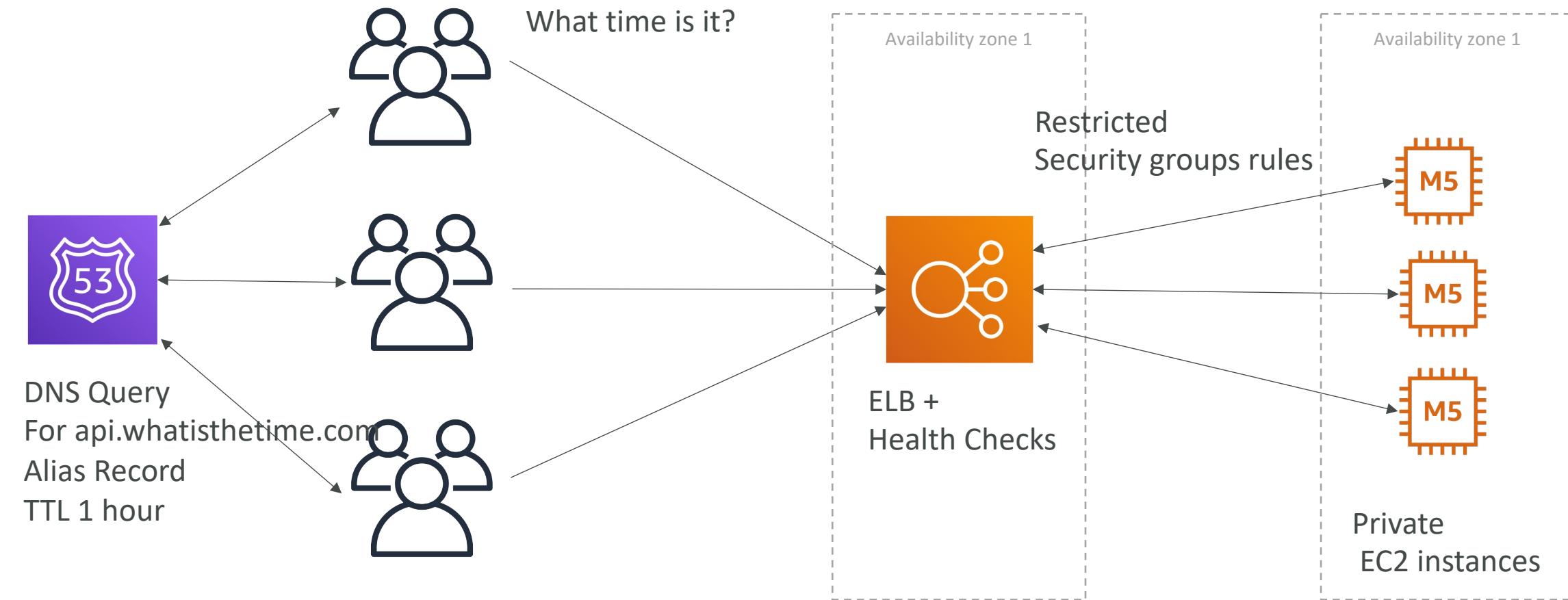


# Stateless web app: What time is it?

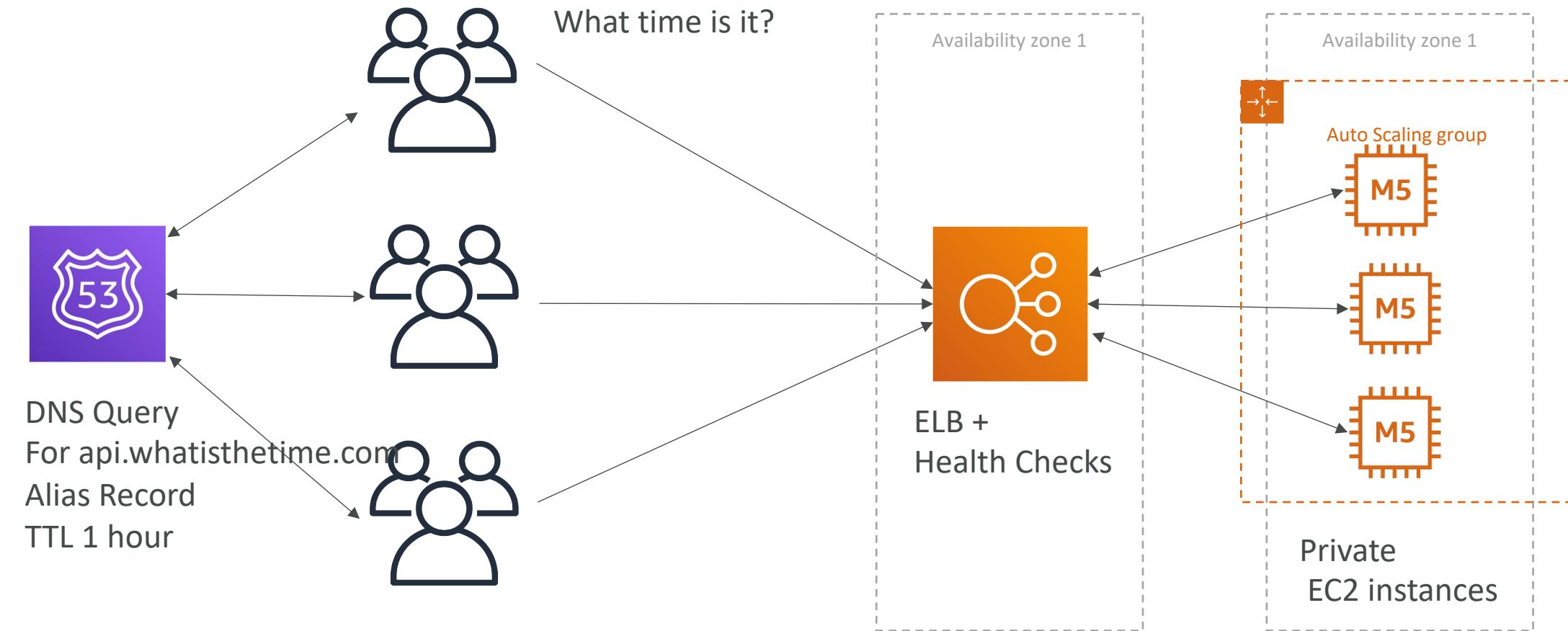
## Scaling horizontally, adding and removing instances



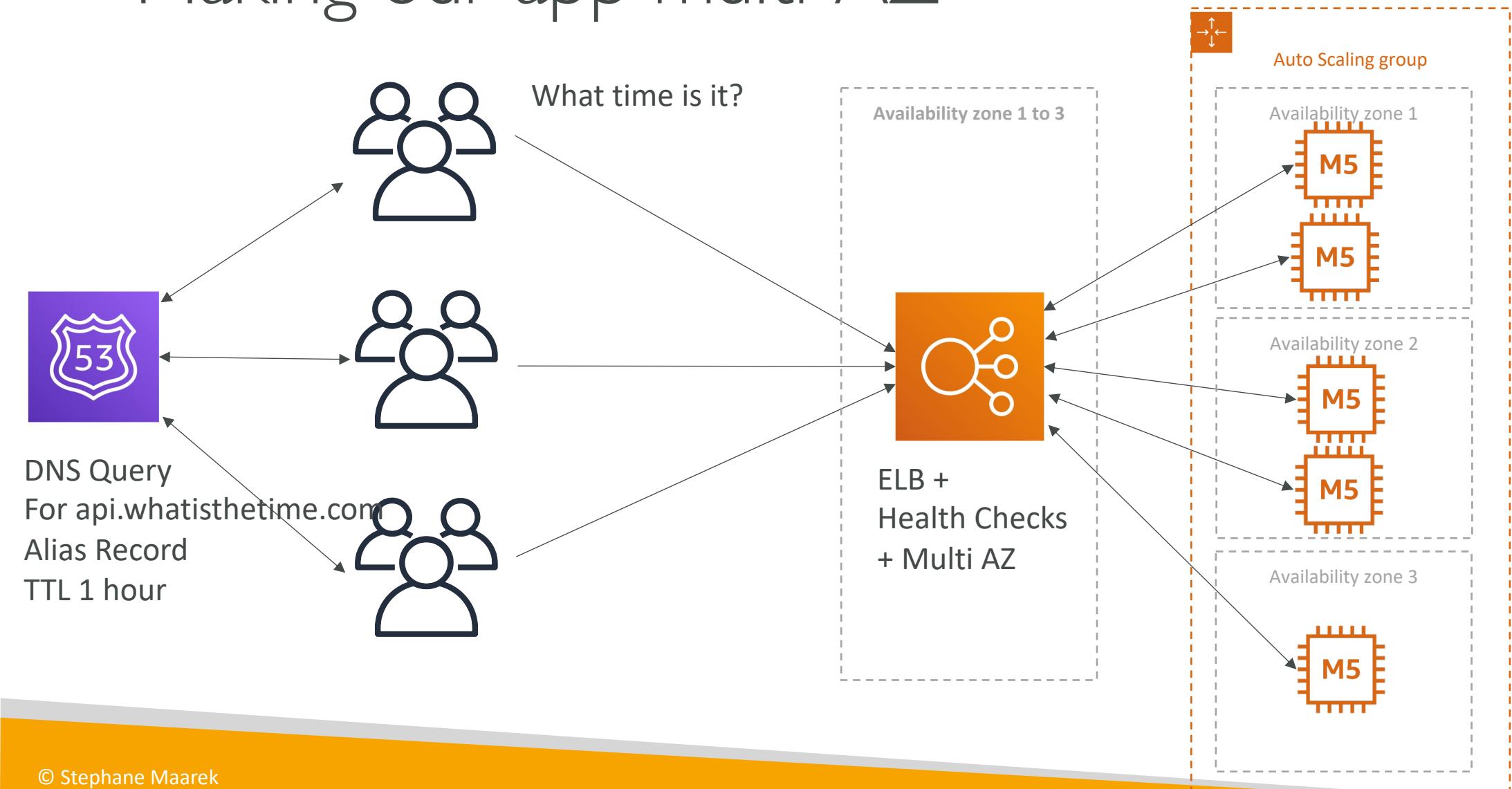
# Stateless web app: What time is it? Scaling horizontally, with a load balancer



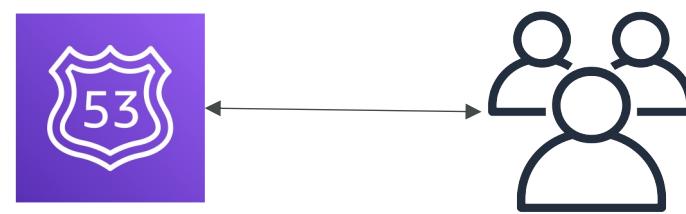
# Stateless web app: What time is it? Scaling horizontally, with an auto-scaling group



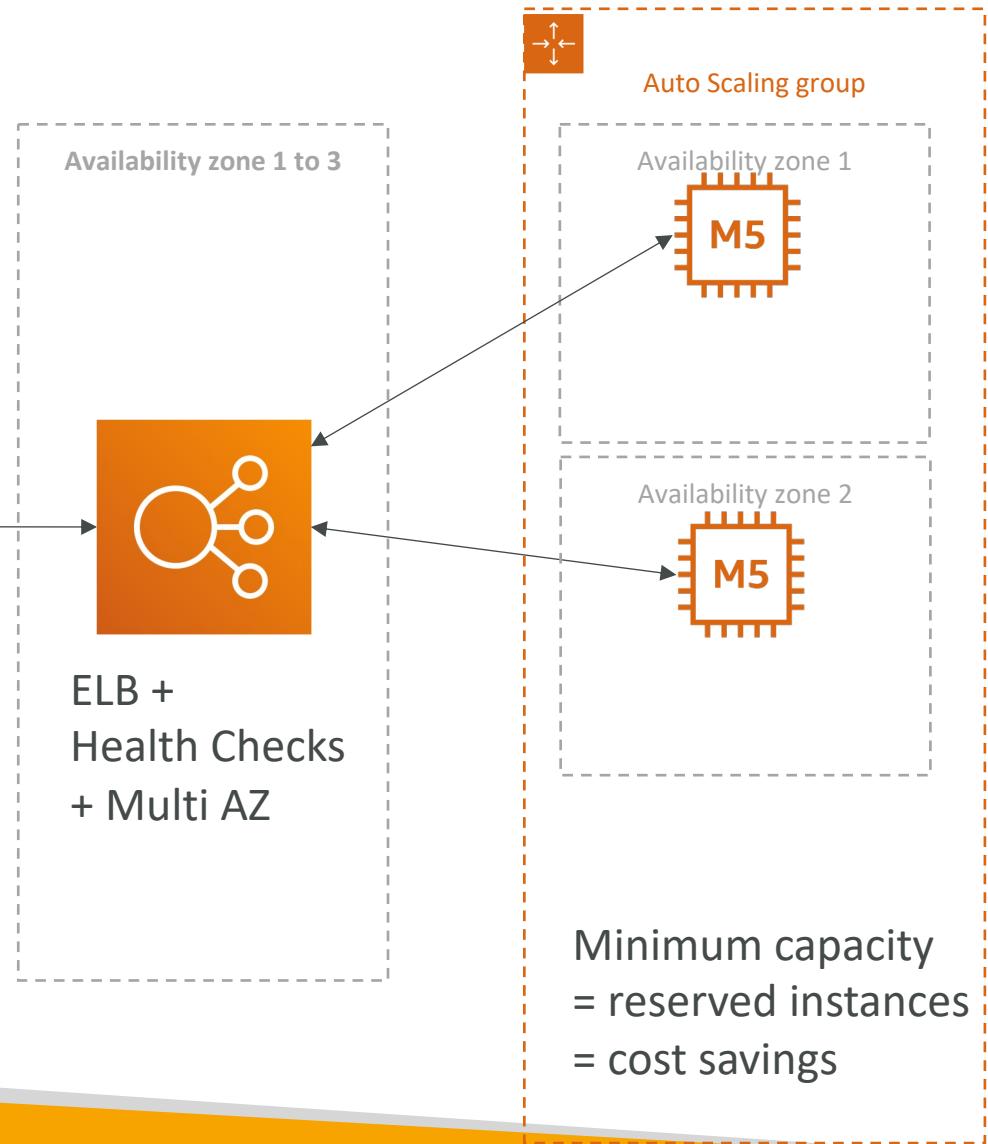
# Stateless web app: What time is it? Making our app multi-AZ



# Minimum 2 AZ => Let's reserve capacity



DNS Query  
For api.whatisthetime.com  
Alias Record  
TTL 1 hour



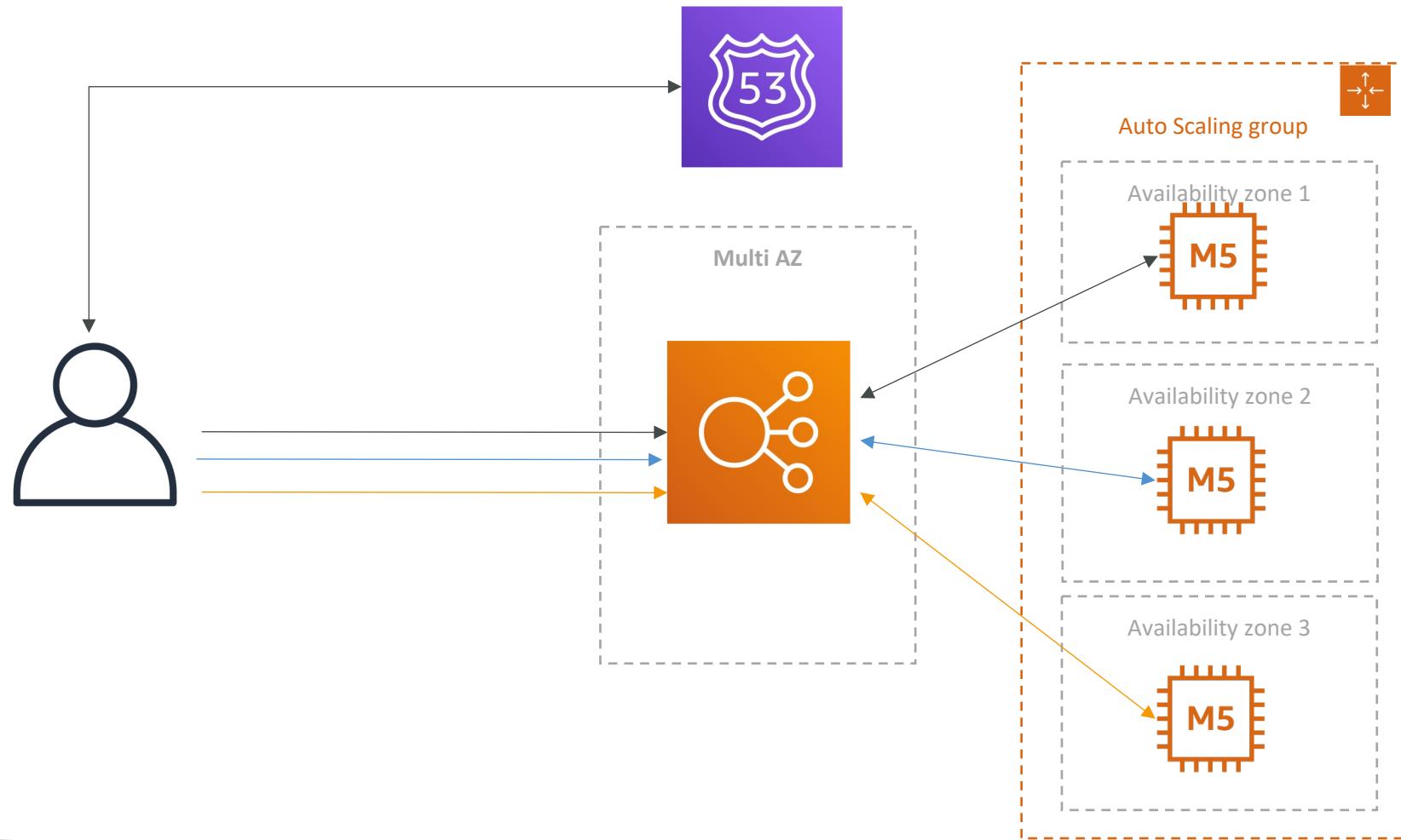
# In this lecture we've discussed...

- 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

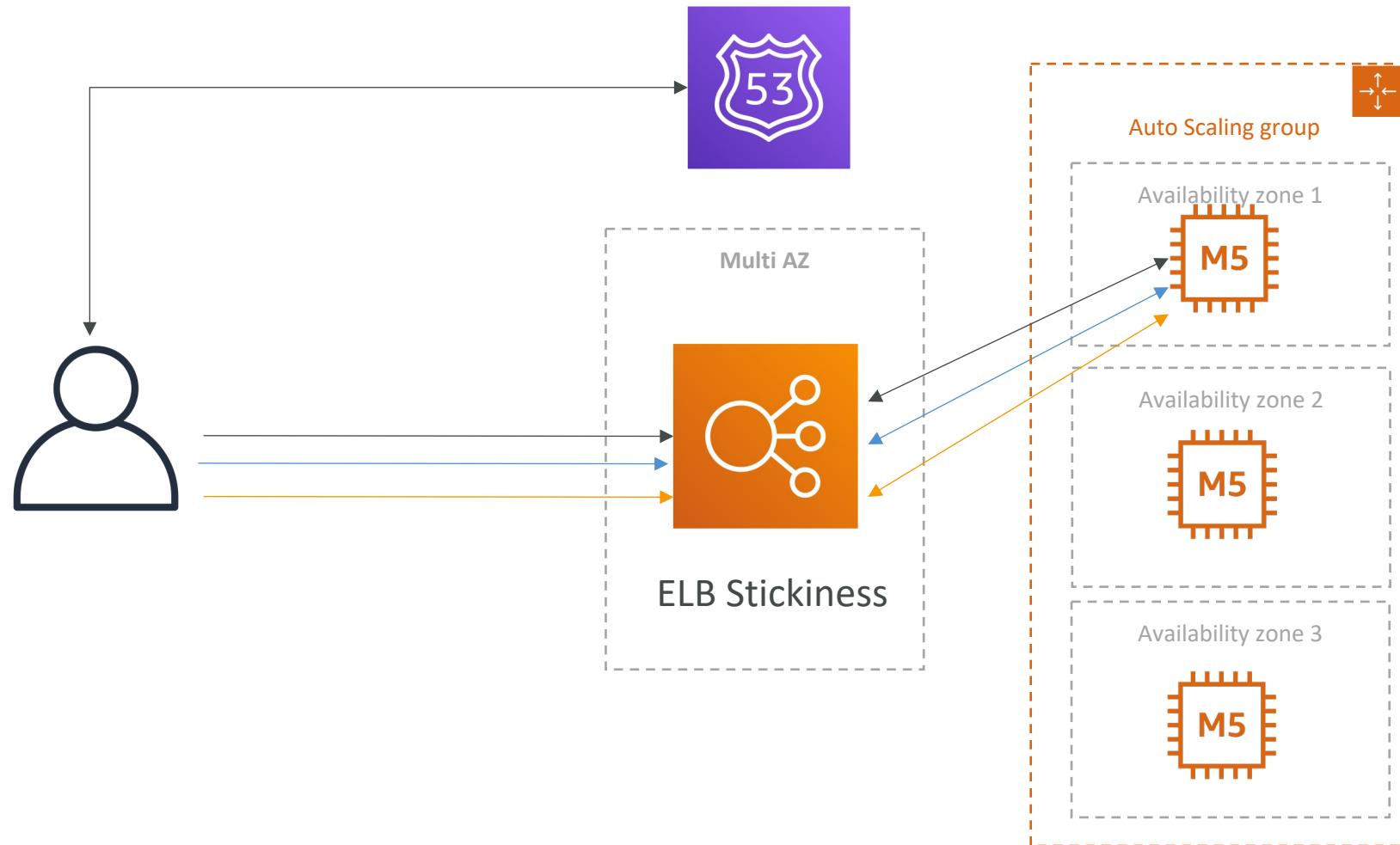
- MyClothes.com allows people to buy clothes online.
- There's a shopping cart
- Our website is having hundreds of users at the same time
- We need to scale, maintain horizontal scalability and keep our web application as stateless as possible
- Users should not lose their shopping cart
- Users should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

# Stateful Web App: MyClothes.com



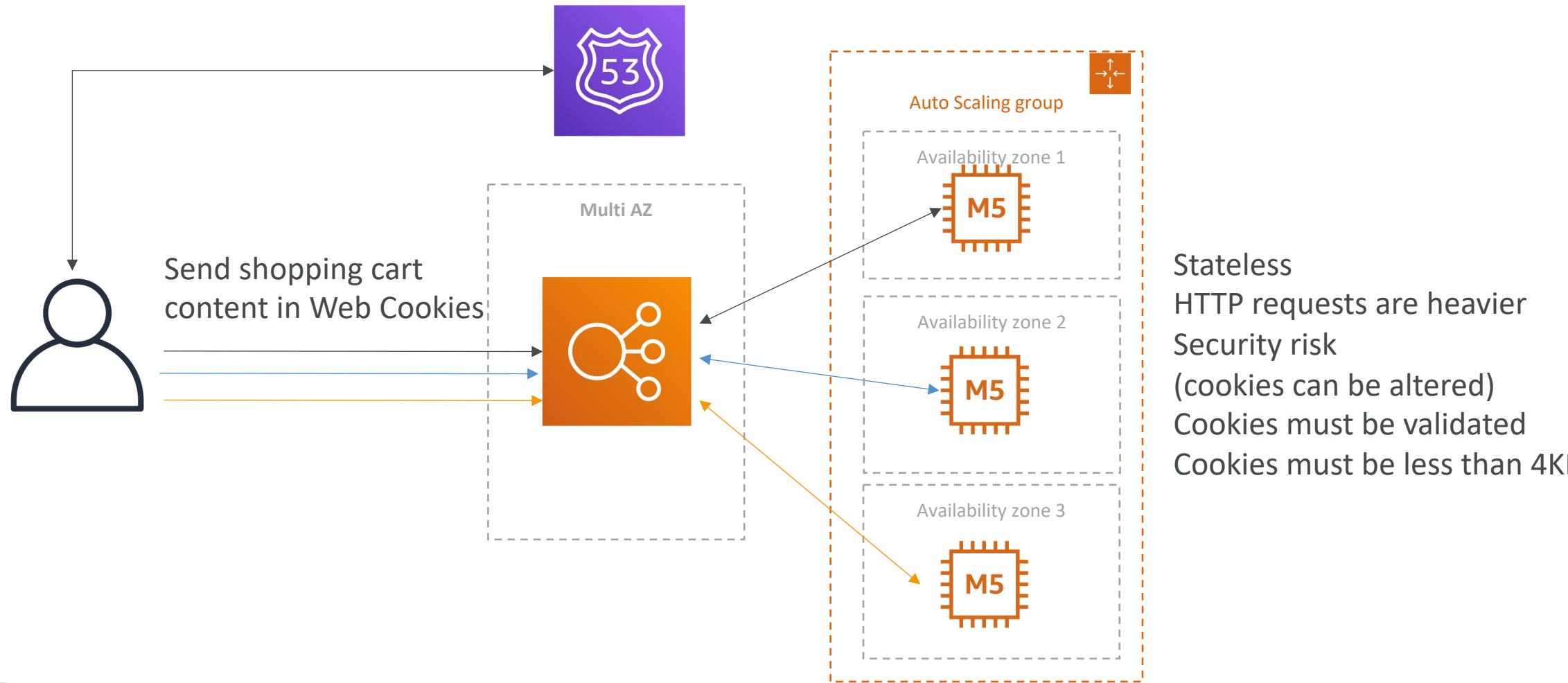
# Stateful Web App: MyClothes.com

## Introduce Stickiness (Session Affinity)



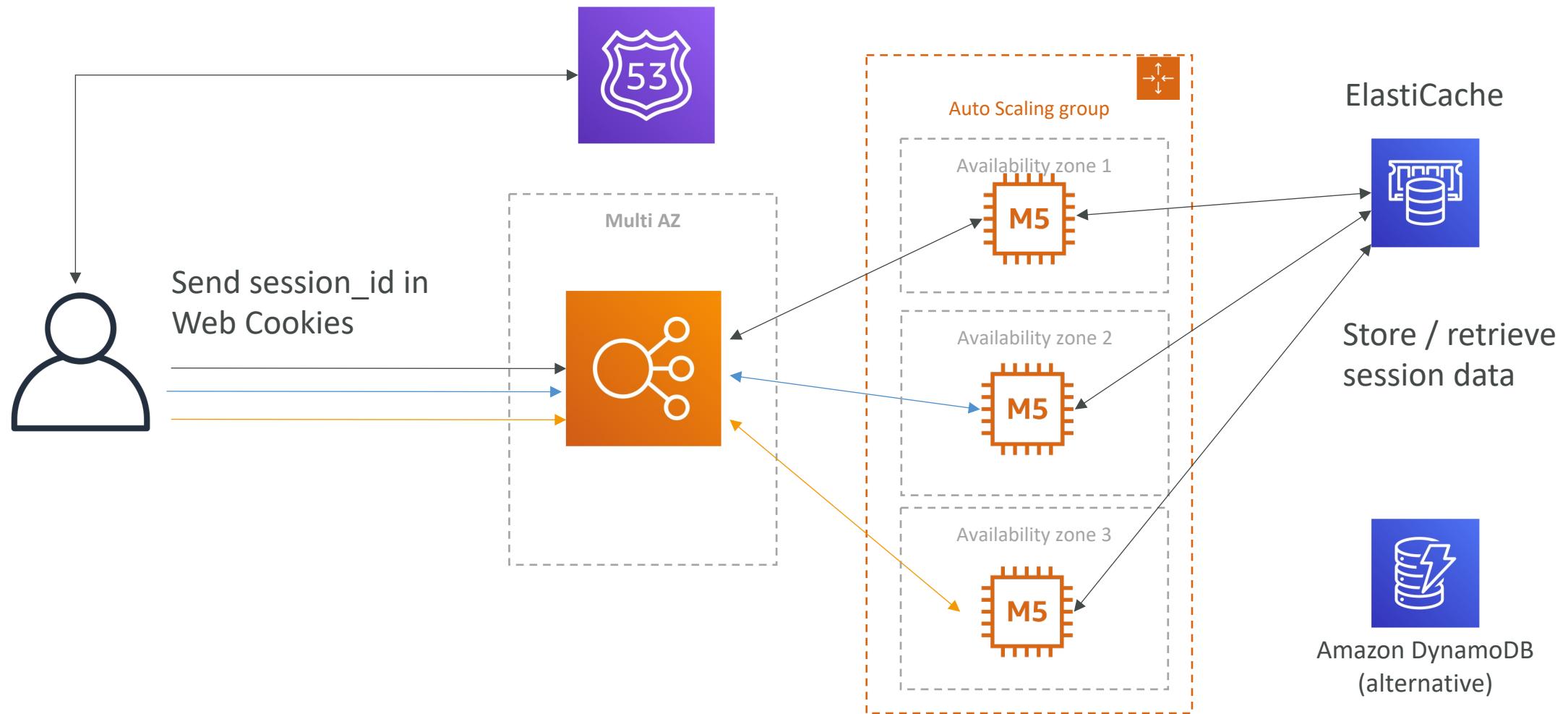
# Stateful Web App: MyClothes.com

## Introduce User Cookies



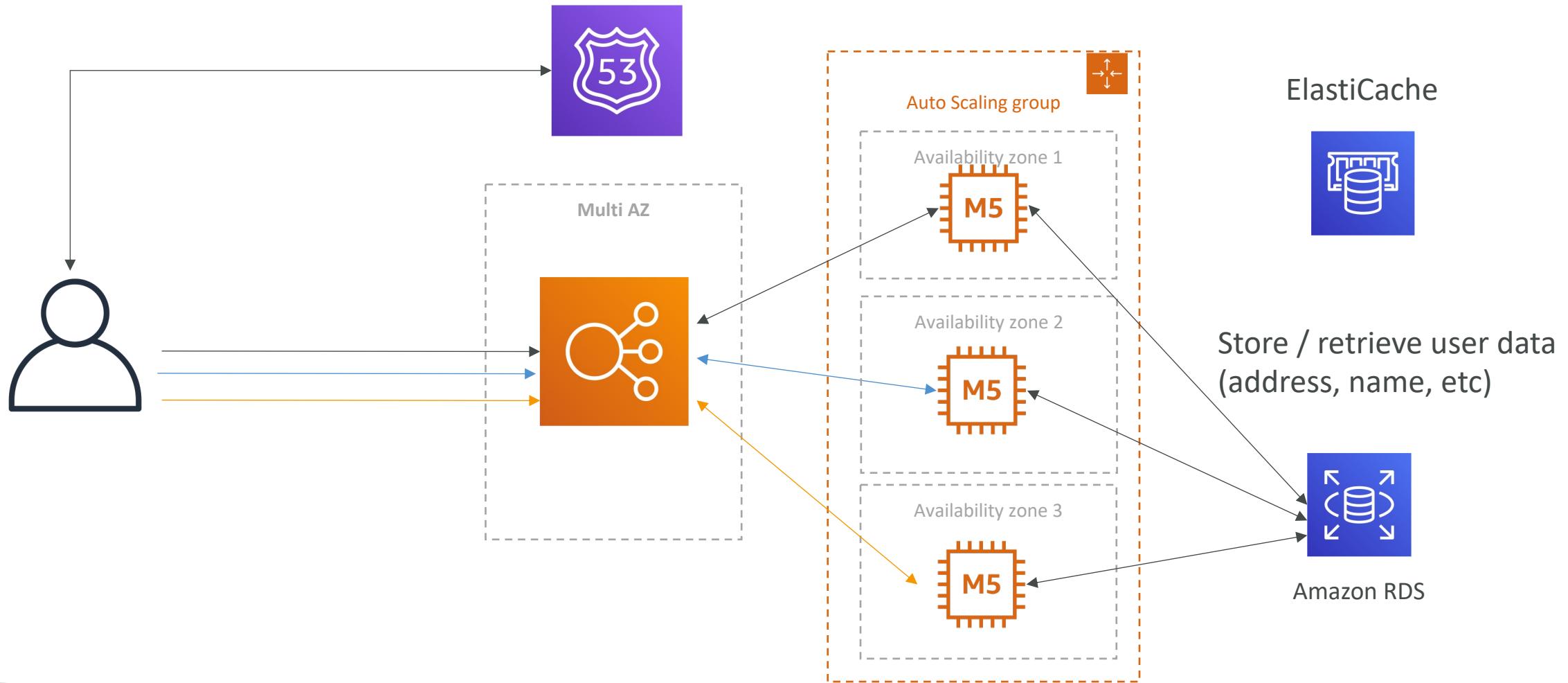
# Stateful Web App: MyClothes.com

## Introduce Server Session



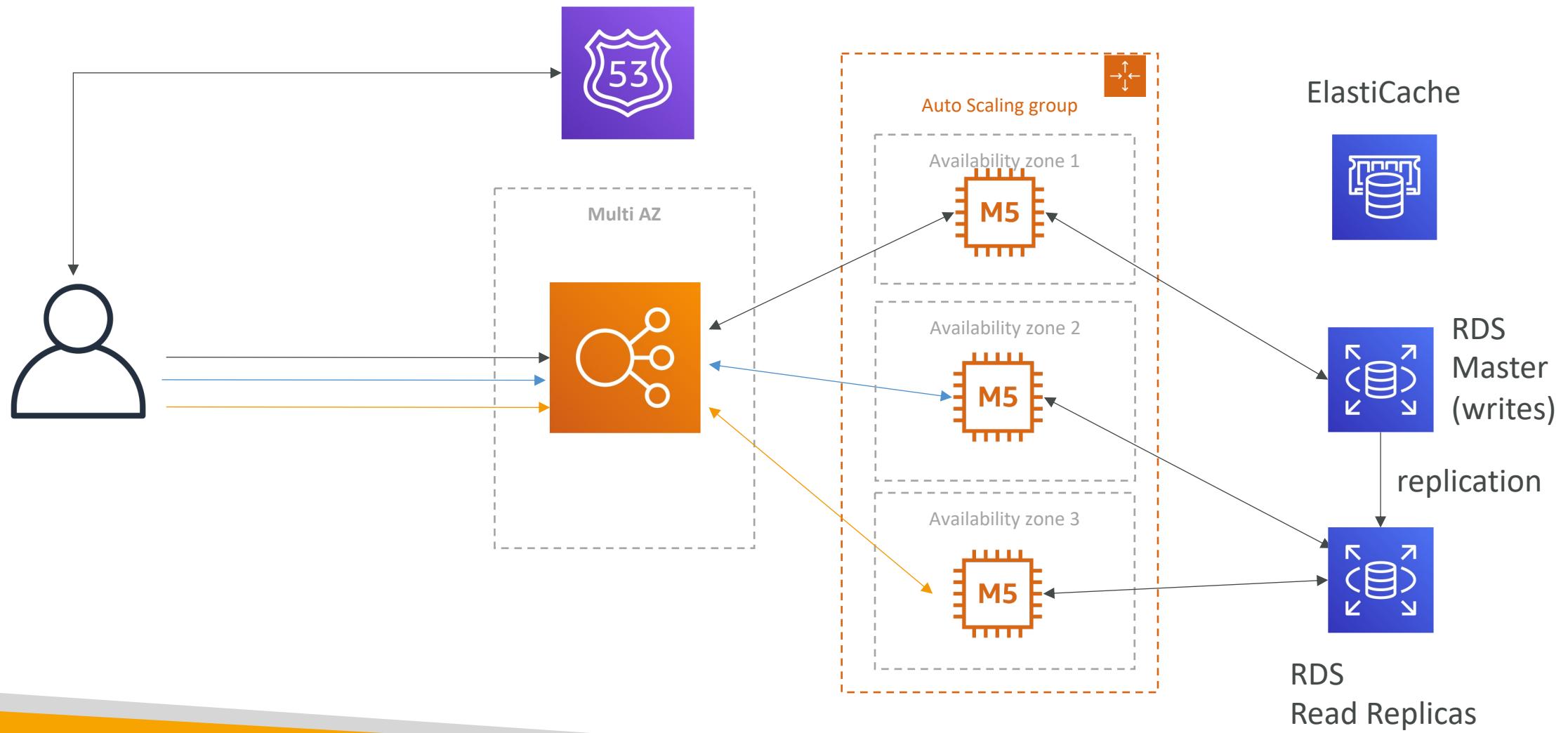
# Stateful Web App: MyClothes.com

## Storing User Data in a database



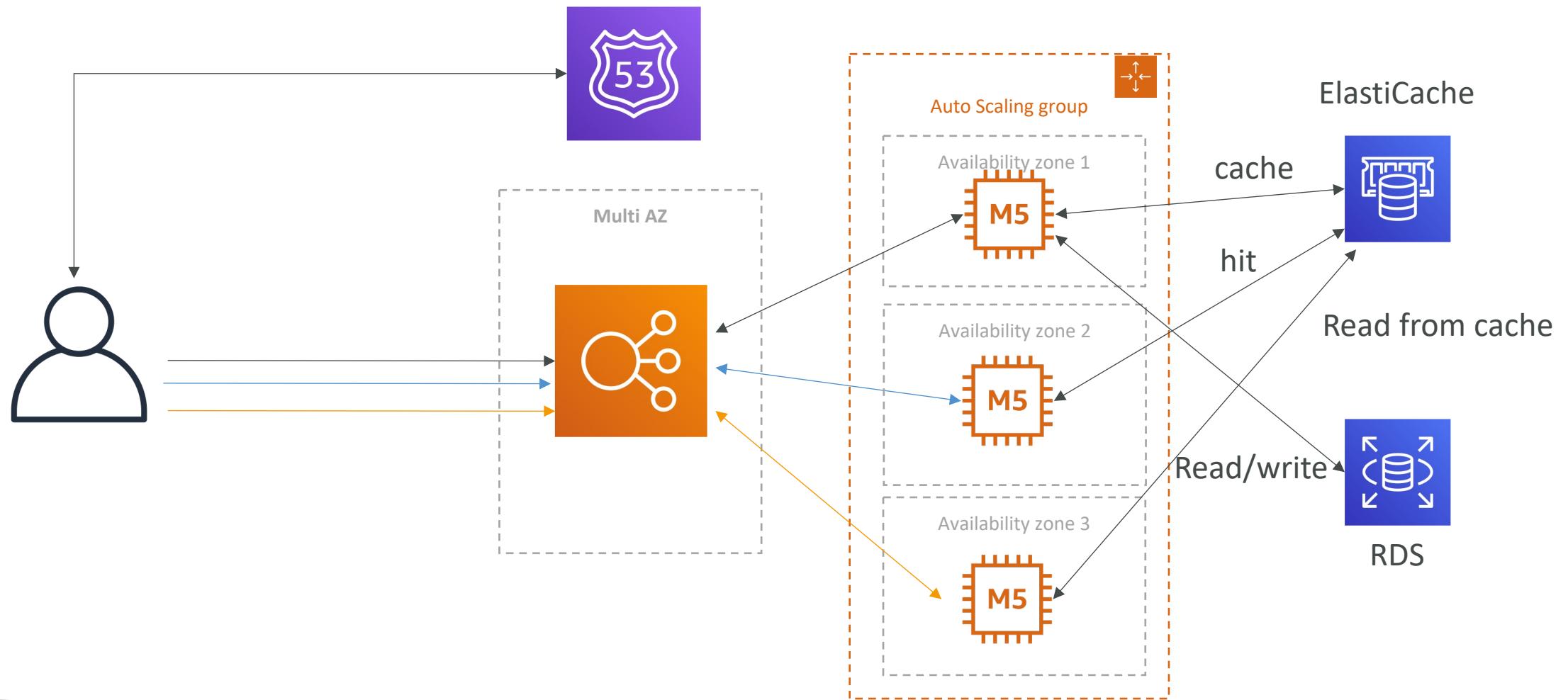
# Stateful Web App: MyClothes.com

## Scaling Reads



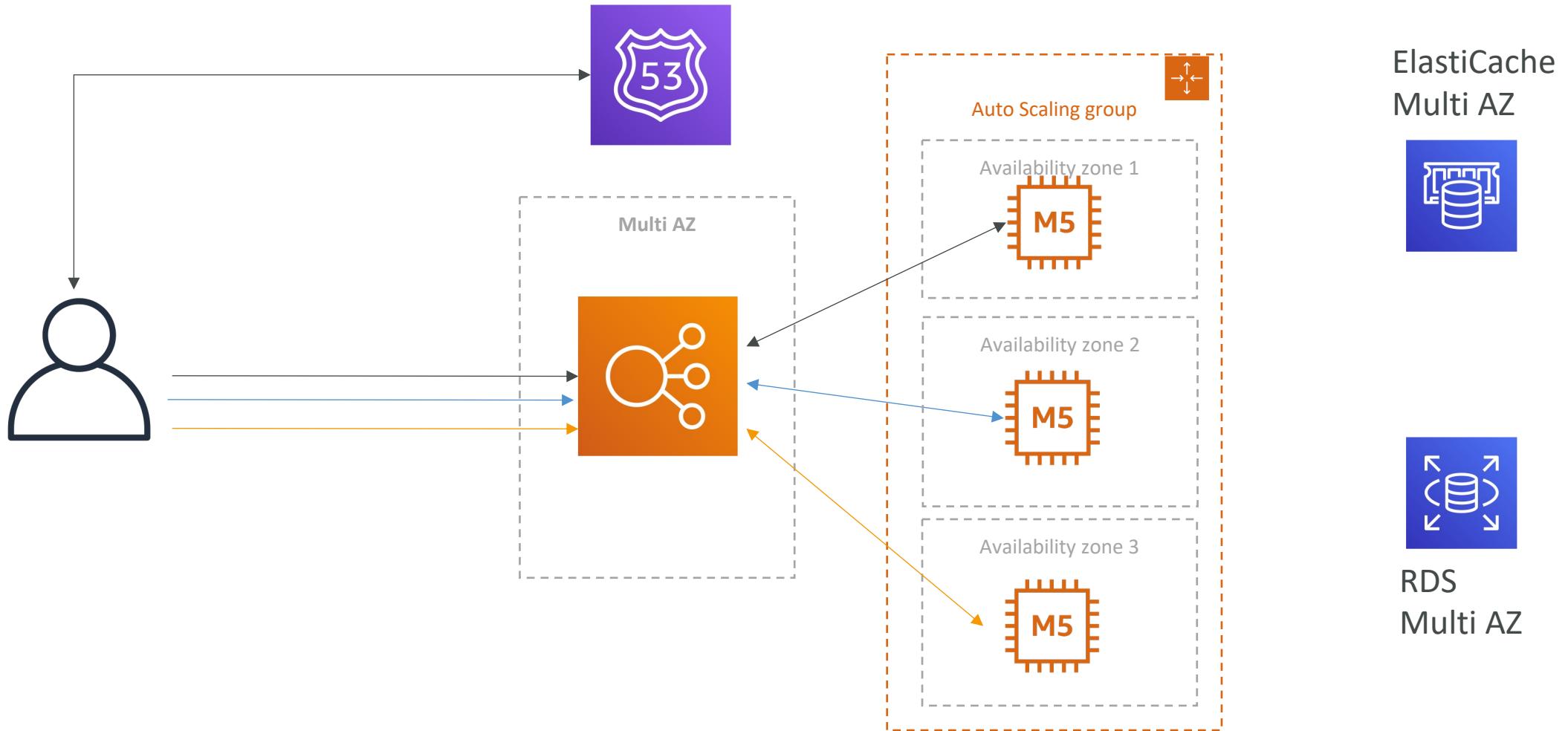
# Stateful Web App: MyClothes.com

## Scaling Reads (Alternative) – Write Through



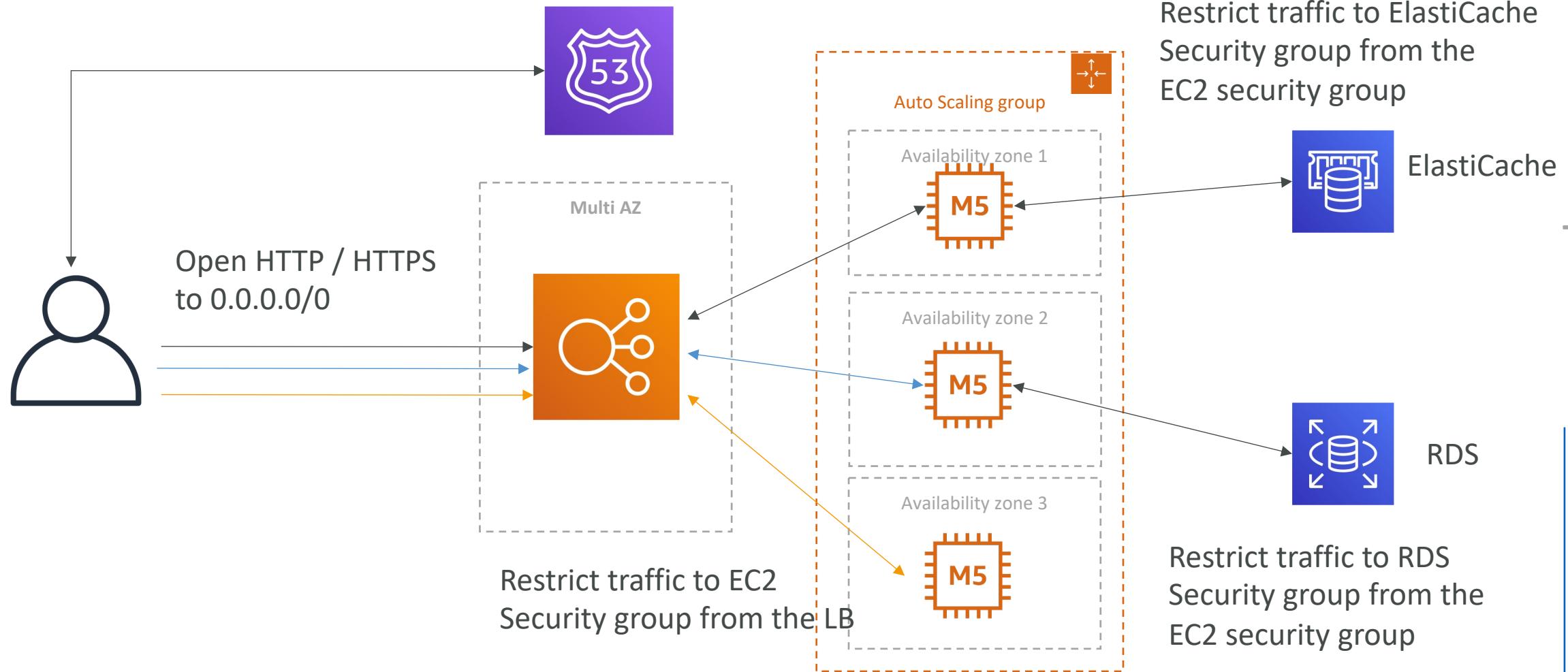
# Stateful Web App: MyClothes.com

## Multi AZ – Survive disasters



# Stateful Web App: MyClothes.com

## Security Groups



# In this lecture we've discussed... 3-tier architectures for web applications

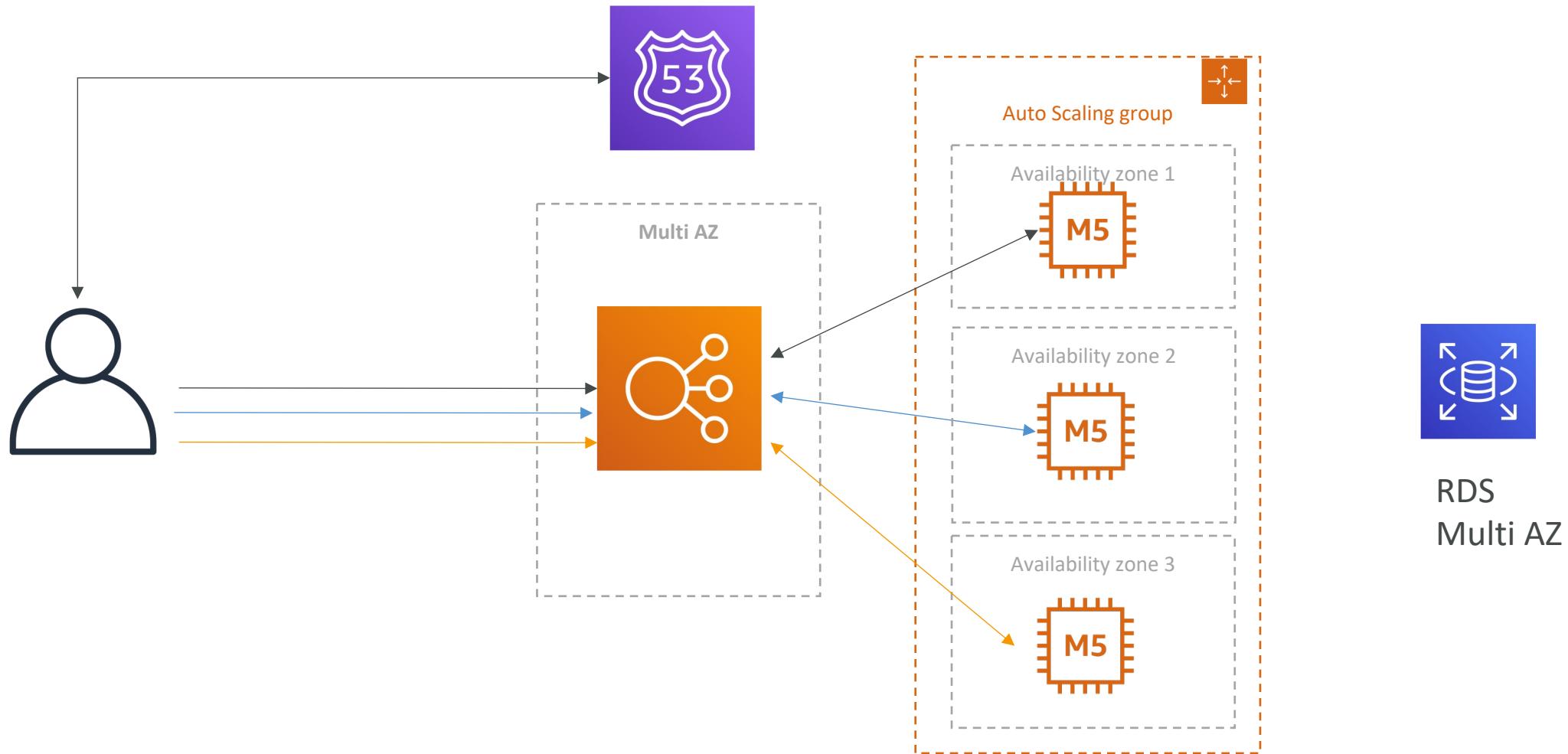
- ELB sticky sessions
- Web clients for storing cookies and making our web app stateless
- ElastiCache
  - For storing sessions (alternative: DynamoDB)
  - For caching data from RDS
  - Multi AZ
- RDS
  - For storing user data
  - Read replicas for scaling reads
  - Multi AZ for disaster recovery
- Tight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

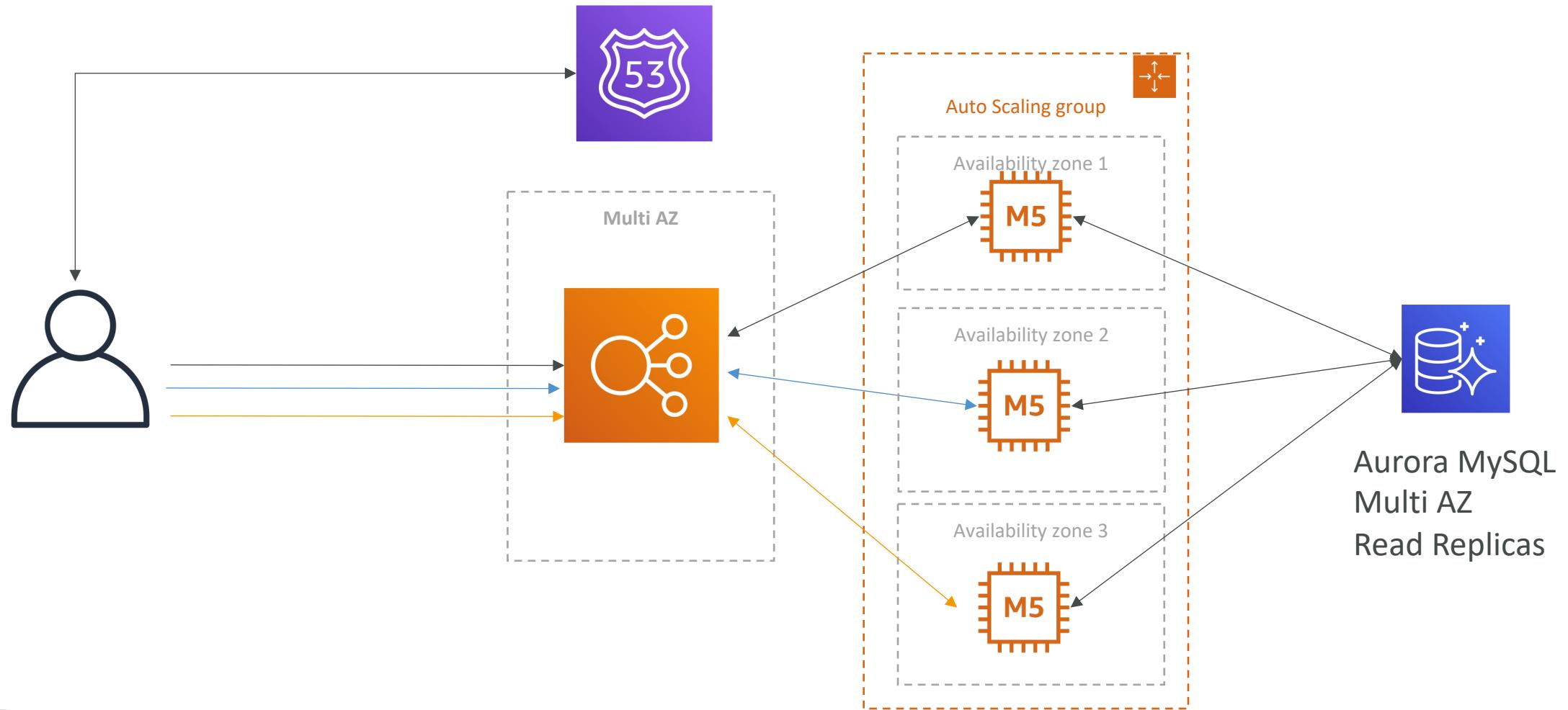
# Stateful Web App: MyWordPress.com

## RDS layer



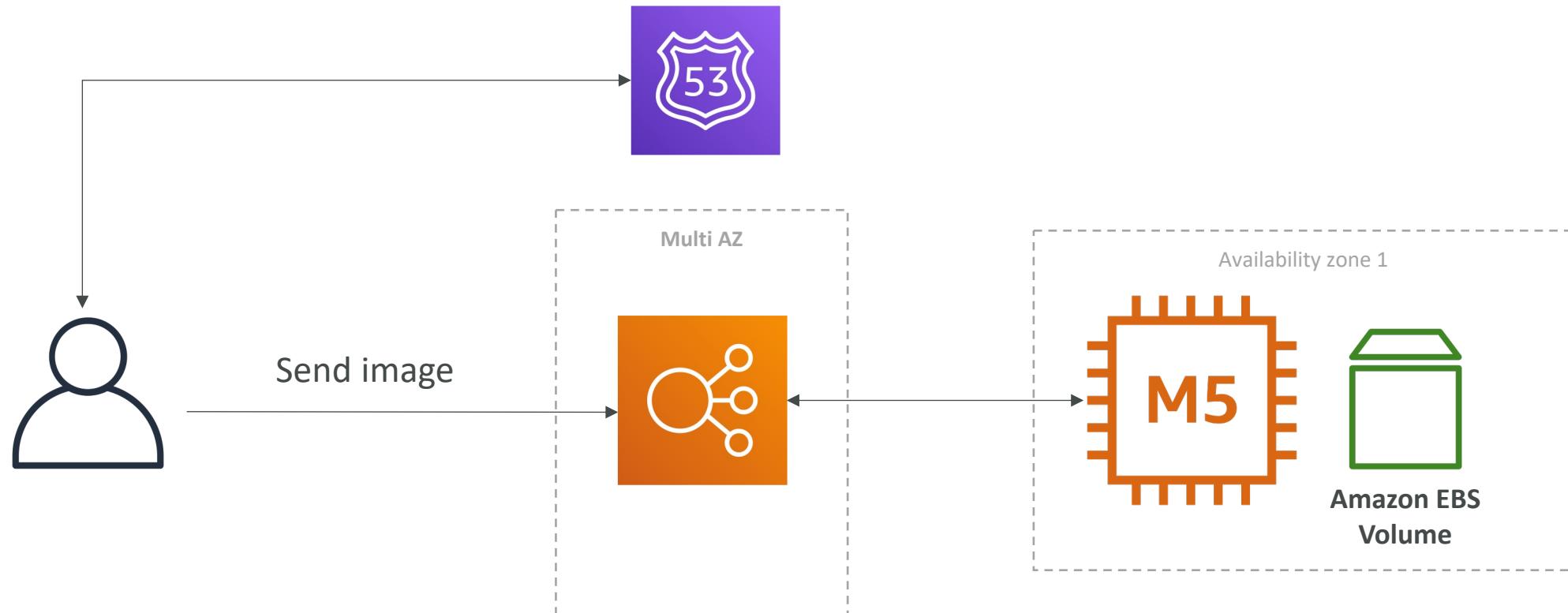
# Stateful Web App: MyWordPress.com

## Scaling with Aurora: Multi AZ & Read Replicas



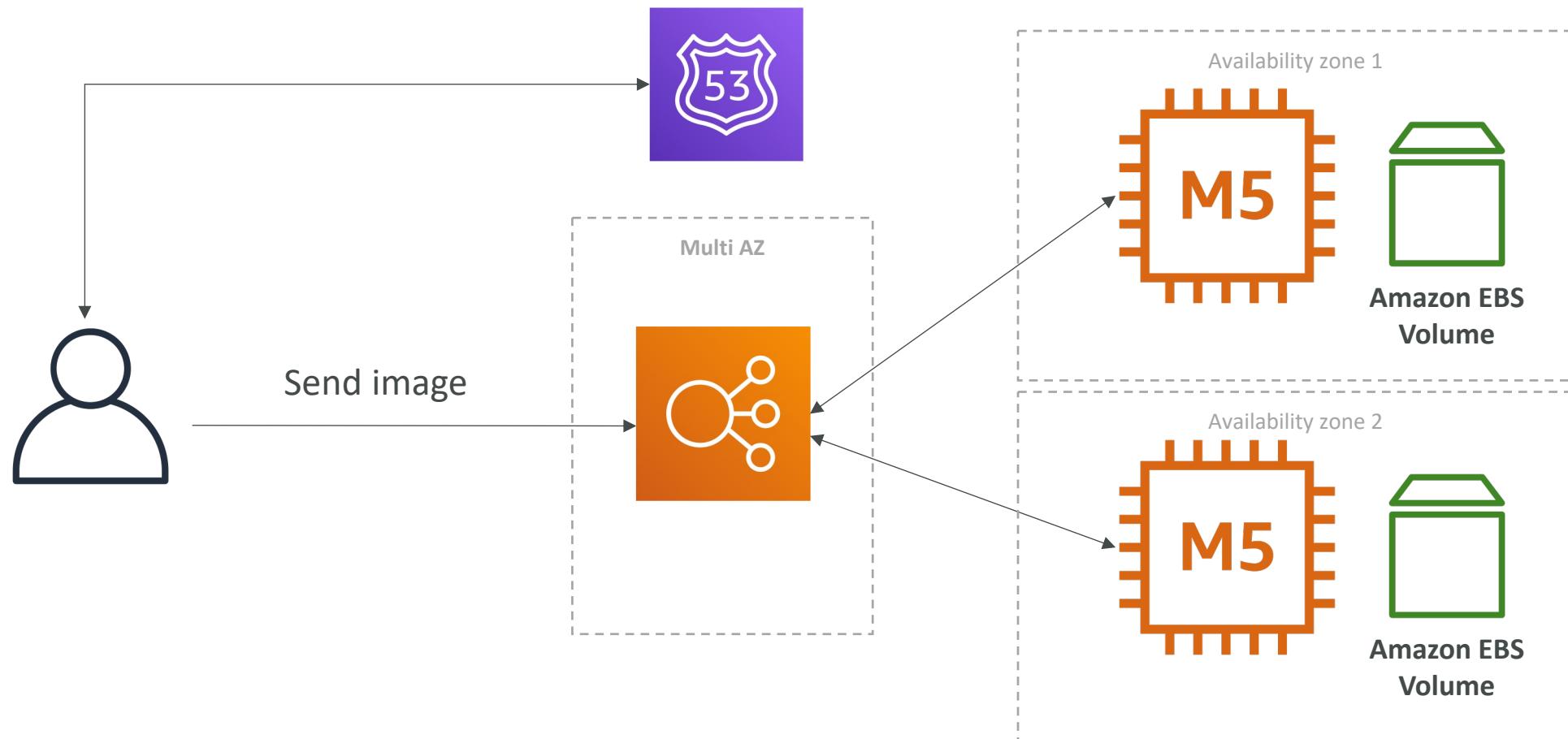
# Stateful Web App: MyWordPress.com

## Storing images with EBS



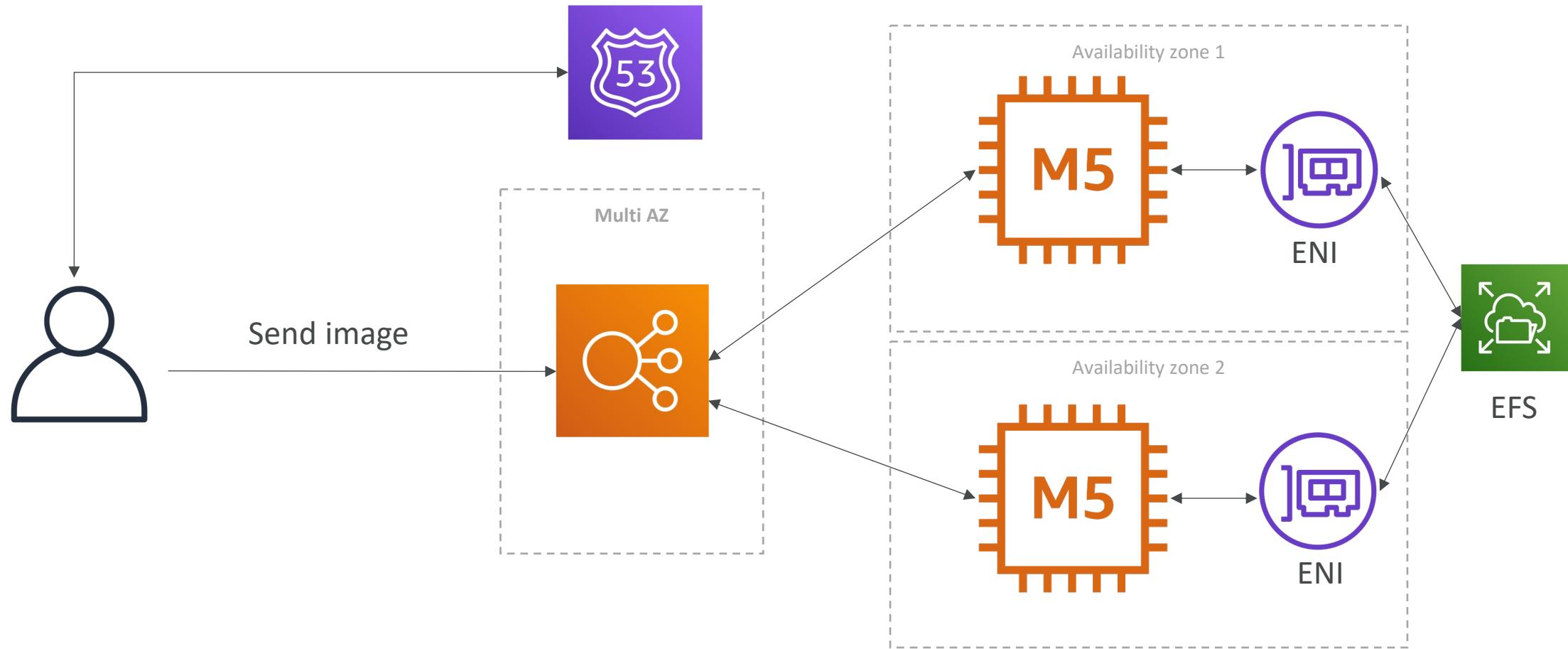
# Stateful Web App: MyWordPress.com

## 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)

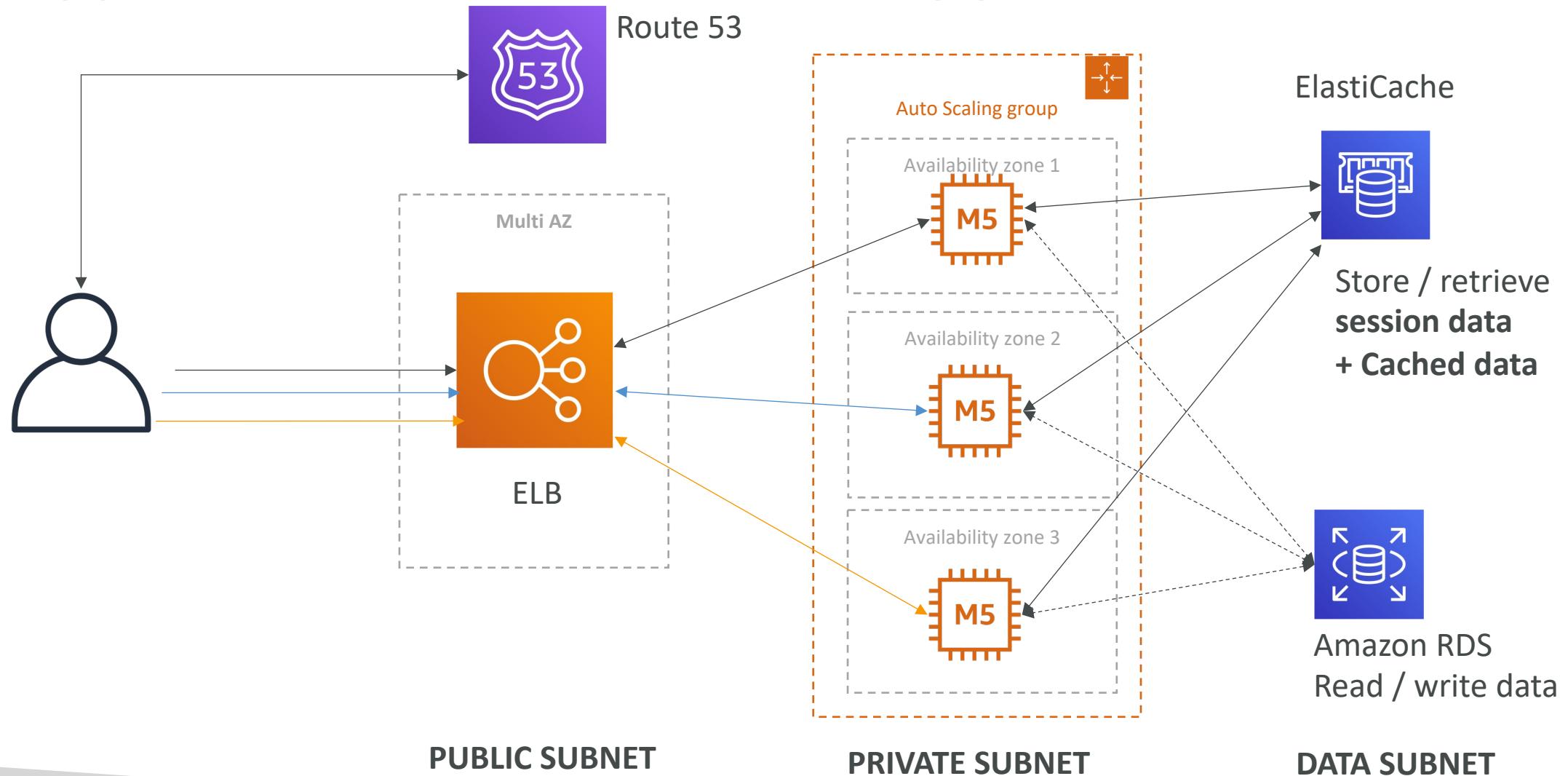
# Instantiating Applications quickly

- 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

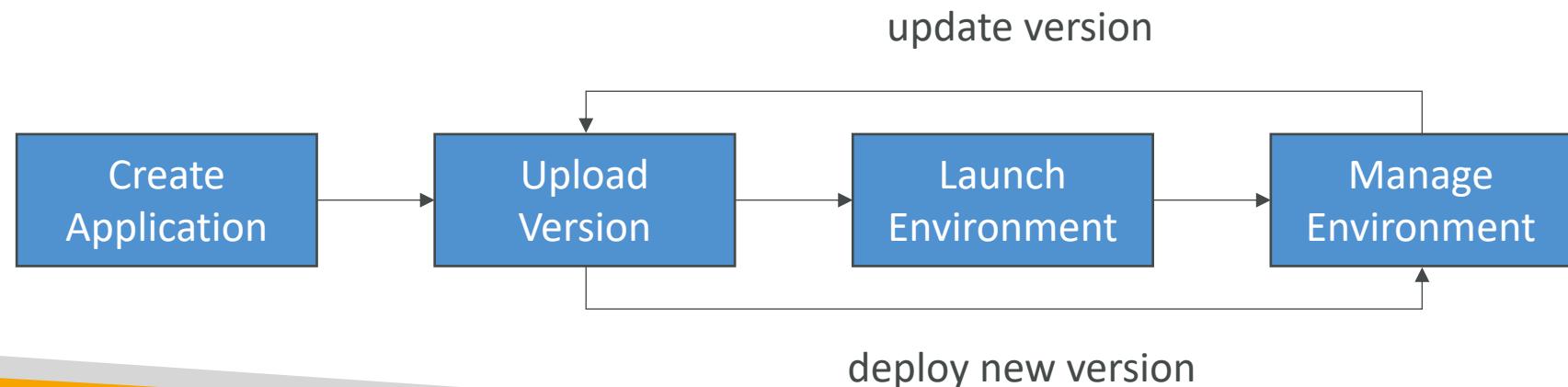
# Elastic Beanstalk – Overview



- 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

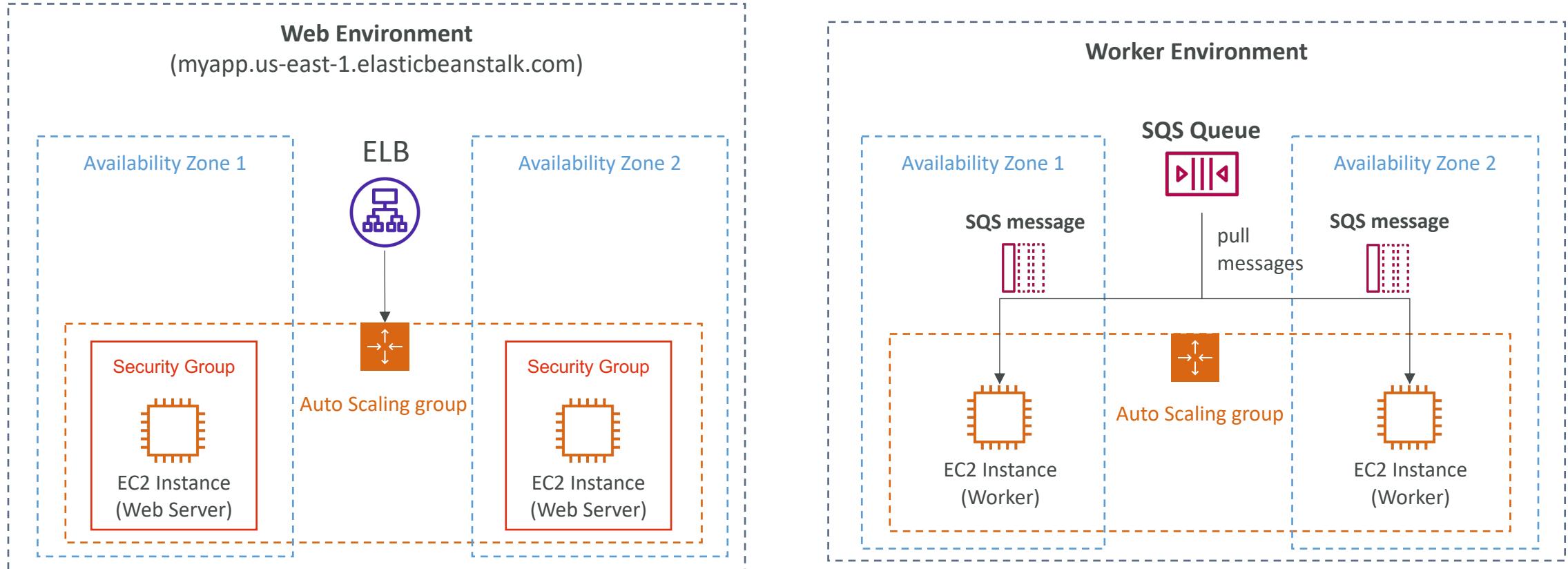
- **Application:** collection of Elastic Beanstalk components (environments, versions, configurations, ...)
- **Application Version:** an iteration of your application code
- **Environment**
  - Collection of AWS resources running an application version (only one application version at a time)
  - **Tiers:** 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

# S3 Storage and Data Management

# Section introduction



- Amazon S3 is one of the main building blocks of AWS
  - It's advertised as "infinitely scaling" storage
  - It's widely popular and deserves its own section
- 
- Many websites use Amazon S3 as a backbone
  - Many AWS services uses Amazon S3 as an integration as well
- 
- We'll have a step-by-step approach to S3

# Amazon S3 Overview - Buckets

- Amazon S3 allows people to store objects (files) in “buckets” (directories)
- Buckets must have a **globally unique name**
- Buckets are defined at the region level
- Naming convention
  - No uppercase
  - No underscore
  - 3-63 characters long
  - Not an IP
  - Must start with lowercase letter or number



# Amazon S3 Overview - 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 ("/")



# Amazon S3 Overview – Objects (continued)

- 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 - Versioning



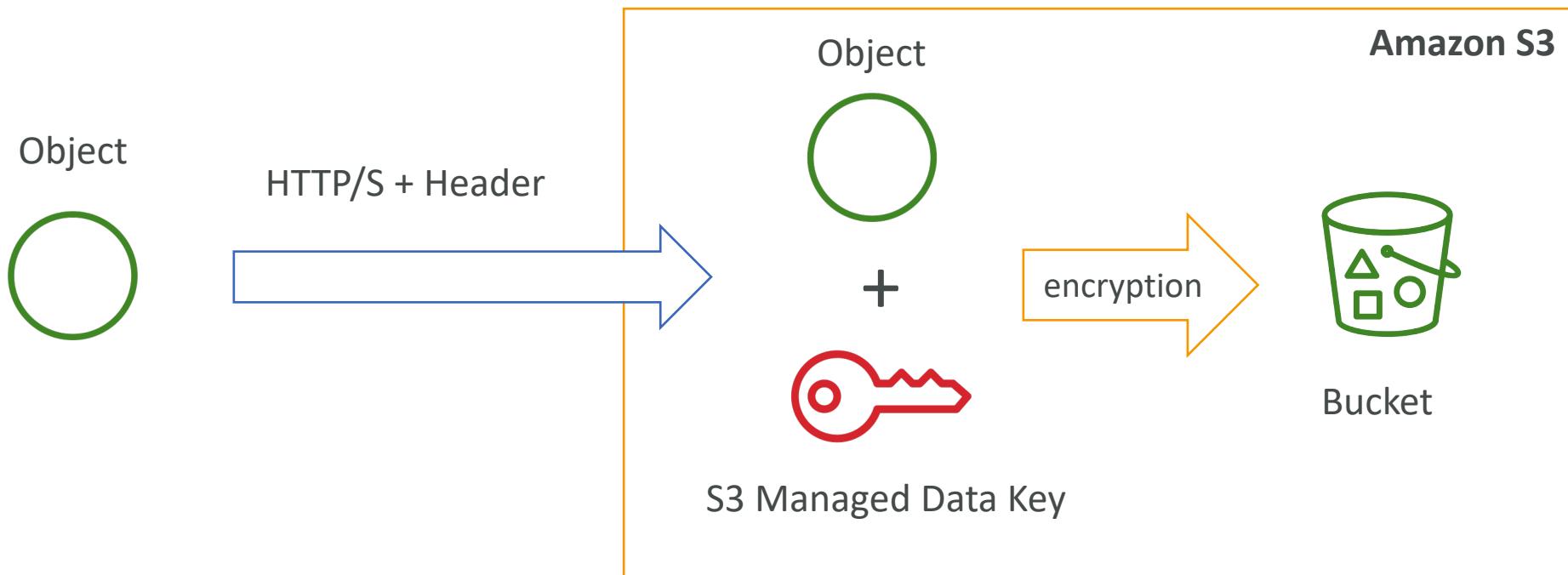
- You can version your files in Amazon S3
- It is enabled at the **bucket level**
- Same key overwrite will increment the “version”: 1, 2, 3....
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
-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

# S3 Encryption for Objects

- There are 4 methods of encrypting objects in S3
  - SSE-S3: encrypts S3 objects using keys handled & managed by AWS
  - SSE-KMS: leverage AWS Key Management Service to manage encryption keys
  - SSE-C: when you want to manage your own encryption keys
  - Client Side Encryption
- It's important to understand which ones are adapted to which situation for the exam

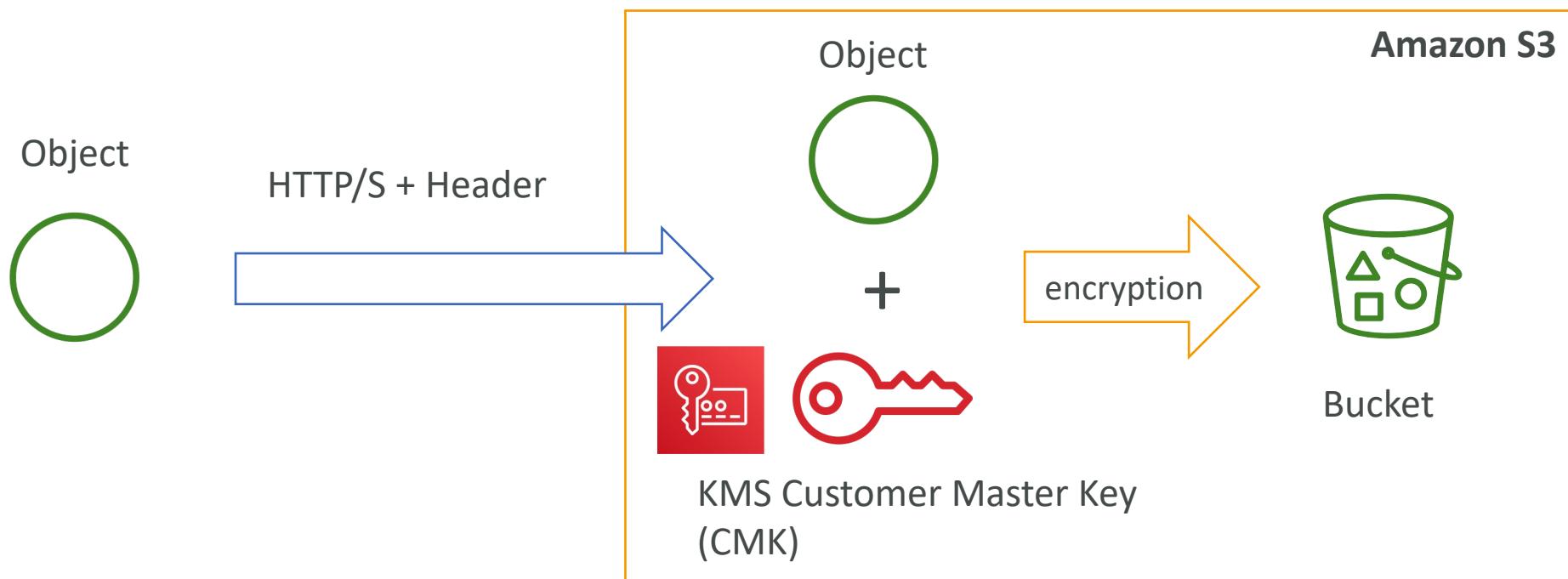
# SSE-S3

- SSE-S3: encryption using keys handled & managed by Amazon S3
- Object is encrypted server side
- AES-256 encryption type
- Must set header: "x-amz-server-side-encryption": "AES256"



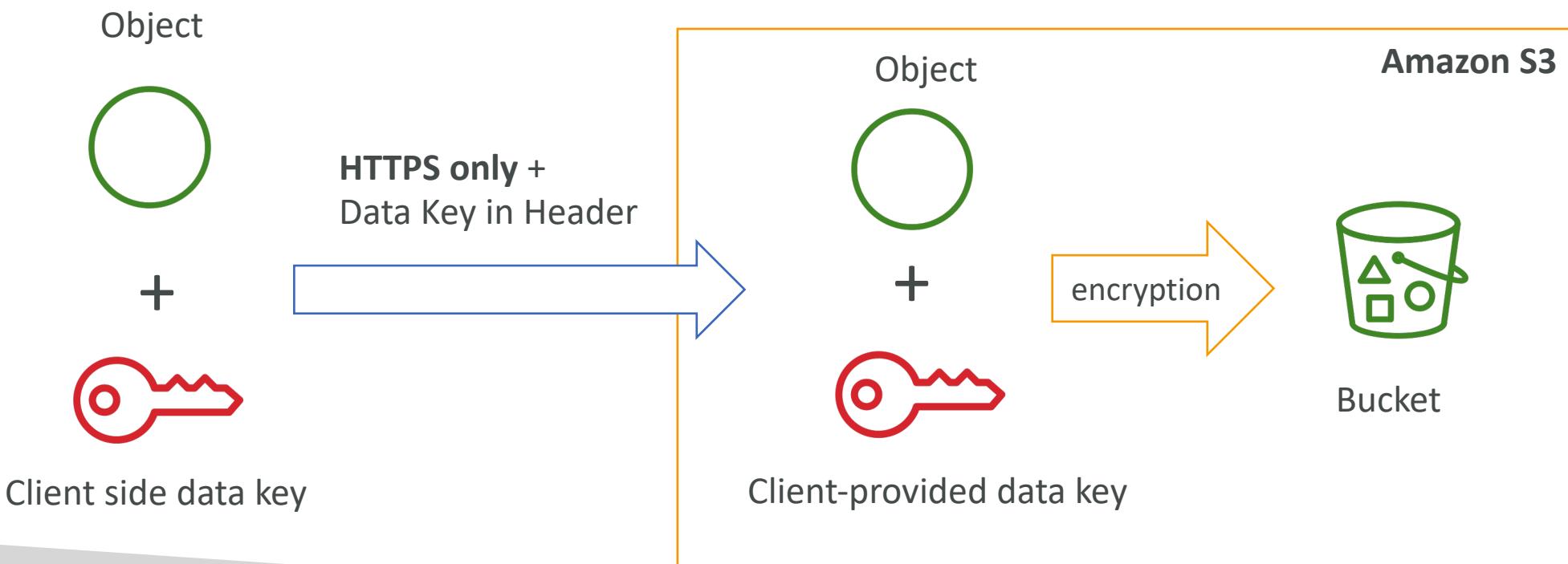
# SSE-KMS

- SSE-KMS: encryption using keys handled & managed by KMS
- KMS Advantages: user control + audit trail
- Object is encrypted server side
- Must set header: "x-amz-server-side-encryption": "aws:kms"



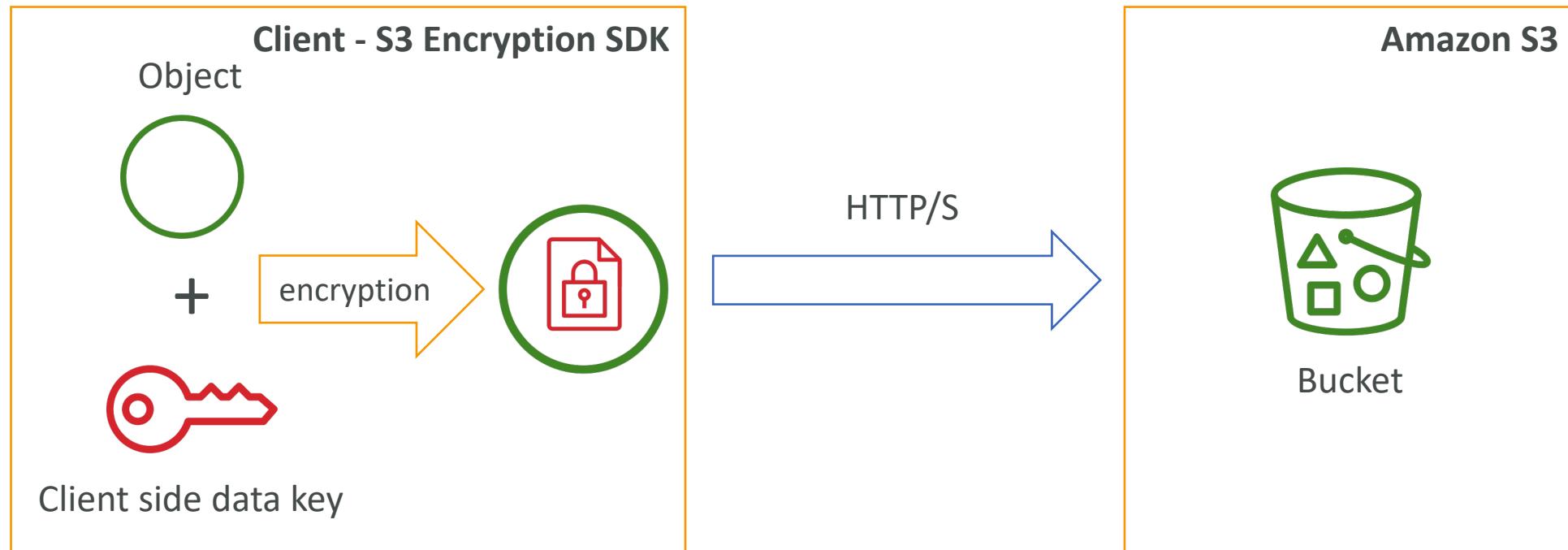
# SSE-C

- SSE-C: server-side encryption using data 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



# Client Side Encryption

- Client library such as the Amazon S3 Encryption Client
- Clients must encrypt data themselves before sending to S3
- Clients must decrypt data themselves when retrieving from S3
- Customer fully manages the keys and encryption cycle



# Encryption in transit (SSL/TLS)



- Amazon S3 exposes:
  - HTTP endpoint: non encrypted
  - HTTPS endpoint: encryption in flight
- You're free to use the endpoint you want, but HTTPS is recommended
- Most clients would use the HTTPS endpoint by default
- HTTPS is mandatory for SSE-C
- Encryption in flight is also called SSL / TLS

# S3 Security

- **User based**
  - IAM policies - which API calls should be allowed for a specific user from IAM console
- **Resource Based**
  - Bucket Policies - bucket wide rules from the S3 console - allows cross account
  - Object Access Control List (ACL) – finer grain
  - Bucket Access Control List (ACL) – less common
- **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

# S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Actions: Set of API to Allow or Deny
  - Effect: Allow / 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/*"  
      ]  
    }  
  ]  
}
```

# Bucket settings for Block Public Access

- Block public access to buckets and objects granted through
  - new access control lists (ACLs)
  - *any* access control lists (ACLs)
  - new public bucket or access point policies
- Block public and cross-account access to buckets and objects through *any* public bucket or access point policies
- 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

# S3 Security - Other

- Networking:
  - Supports VPC Endpoints (for instances in VPC without www internet)
- Logging and Audit:
  - S3 Access Logs can be stored in other S3 bucket
  - API calls can be logged in AWS CloudTrail
- User Security:
  - MFA Delete: MFA (multi factor authentication) can be required in versioned buckets to delete objects
  - Pre-Signed URLs: URLs that are valid only for a limited time (ex: premium video service for logged in users)

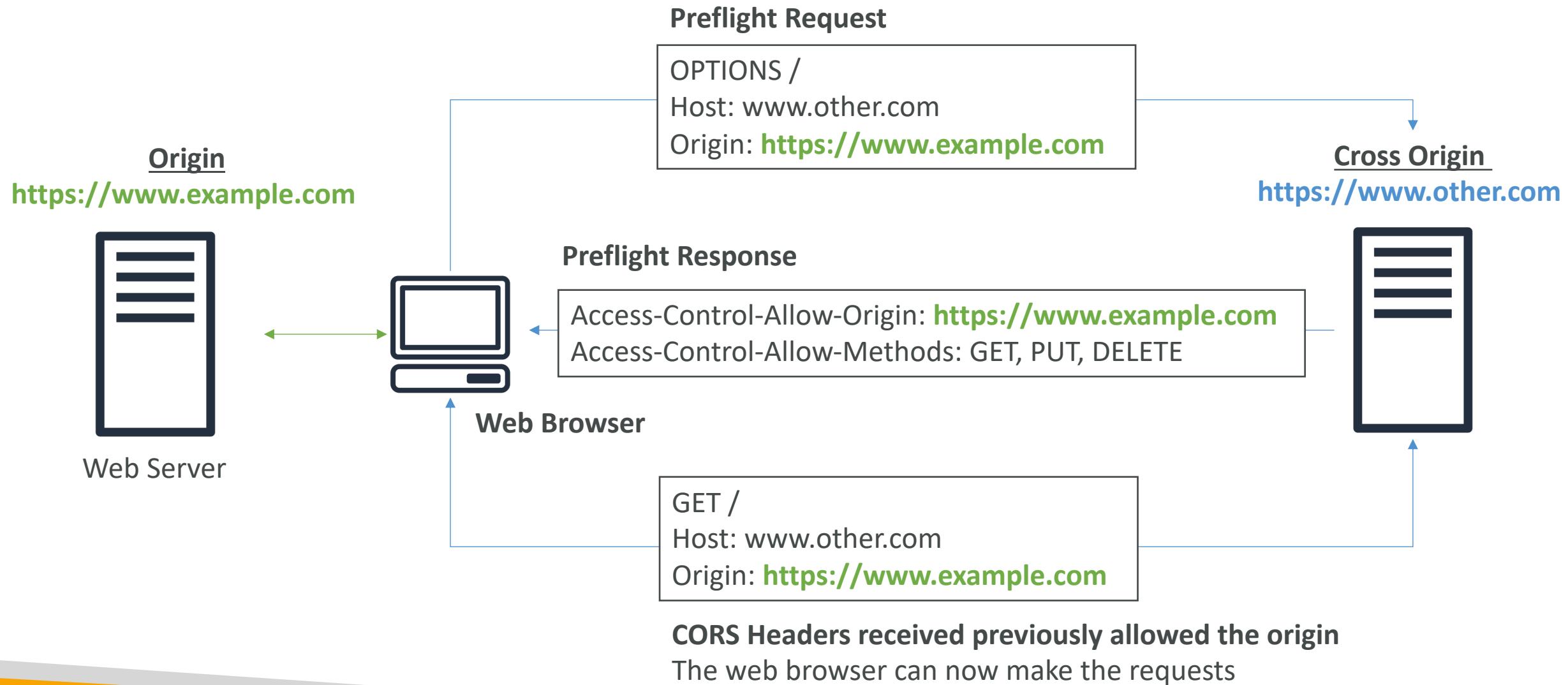
# S3 Websites

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

# CORS - Explained

- An **origin** is a scheme (protocol), host (domain) and port
  - E.g.: <https://www.example.com> (implied port is 443 for HTTPS, 80 for HTTP)
- CORS means Cross-Origin Resource Sharing
- 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** (ex: Access-Control-Allow-Origin)

# CORS – Diagram



# S3 CORS

- If a client does 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 - Consistency Model

- Strong consistency as of December 2020:
- After a:
  - successful write of a new object (new PUT)
  - or an overwrite or delete of an existing object (overwrite PUT or DELETE)
- ...any:
  - subsequent read request immediately receives the latest version of the object (read after write consistency)
  - subsequent list request immediately reflects changes (list consistency)
- Available at no additional cost, without any performance impact

# Developing on AWS

# AWS EC2 Instance Metadata

- AWS EC2 Instance Metadata is powerful but one of the least known features to developers
- It allows AWS EC2 instances to "learn about themselves" without using an IAM Role for that purpose.
- The URL is <http://169.254.169.254/latest/meta-data>
- You can retrieve the IAM Role name from the metadata, but you CANNOT retrieve the IAM Policy.
- Metadata = Info about the EC2 instance
- Userdata = launch script of the EC2 instance
- Let's practice and see what we can do with it!

# AWS SDK Overview

- What if you want to perform actions on AWS directly from your applications code ? (without using the CLI).
- You can use an SDK (software development kit) !
- Official SDKs are...
  - Java
  - .NET
  - Node.js
  - PHP
  - Python (named boto3 / botocore)
  - Go
  - Ruby
  - C++

# AWS SDK Overview

- We have to use the AWS SDK when coding against AWS Services such as DynamoDB
- Fun fact... the AWS CLI uses the Python SDK (boto3)
- The exam expects you to know when you should use an SDK
- We'll practice the AWS SDK when we get to the Lambda functions
- Good to know: if you don't specify or configure a default region, then us-east-1 will be chosen by default

# Advanced S3

S3, Glacier, Athena

# S3 MFA-Delete

- MFA (multi factor authentication) forces user to generate a code on a device (usually a mobile phone or hardware) before doing important operations on S3
- To use MFA-Delete, enable Versioning on the S3 bucket
- You will need MFA to
  - permanently delete an object version
  - suspend versioning on the bucket
- You won't need MFA for
  - enabling versioning
  - listing deleted versions
- Only the bucket owner (root account) can enable/disable MFA-Delete
- MFA-Delete currently can only be enabled using the CLI

# S3 Default Encryption vs Bucket Policies

- One way to “force encryption” is to use a bucket policy and refuse any API call to PUT an S3 object without encryption headers:

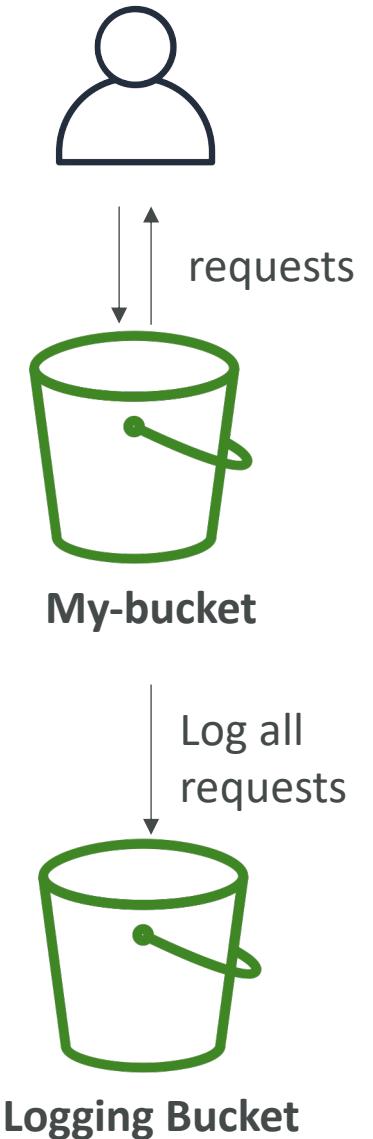
```
{  
    "Version": "2012-10-17",  
    "Id": "PutObjPolicy",  
    "Statement": [  
        {  
            "Sid": "DenyIncorrectEncryptionHeader",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::<bucket_name>/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-server-side-encryption": "AES256"  
                }  
            }  
        }  
    ],  
}.
```

```
{  
    "Sid": "DenyUnEncryptedObjectUploads",  
    "Effect": "Deny",  
    "Principal": "*",  
    "Action": "s3:PutObject",  
    "Resource": "arn:aws:s3:::<bucket_name>/*",  
    "Condition": {  
        "Null": {  
            "s3:x-amz-server-side-encryption": true  
        }  
    }  
}
```

- Another way is to use the “default encryption” option in S3
- Note: Bucket Policies are evaluated before “default encryption”

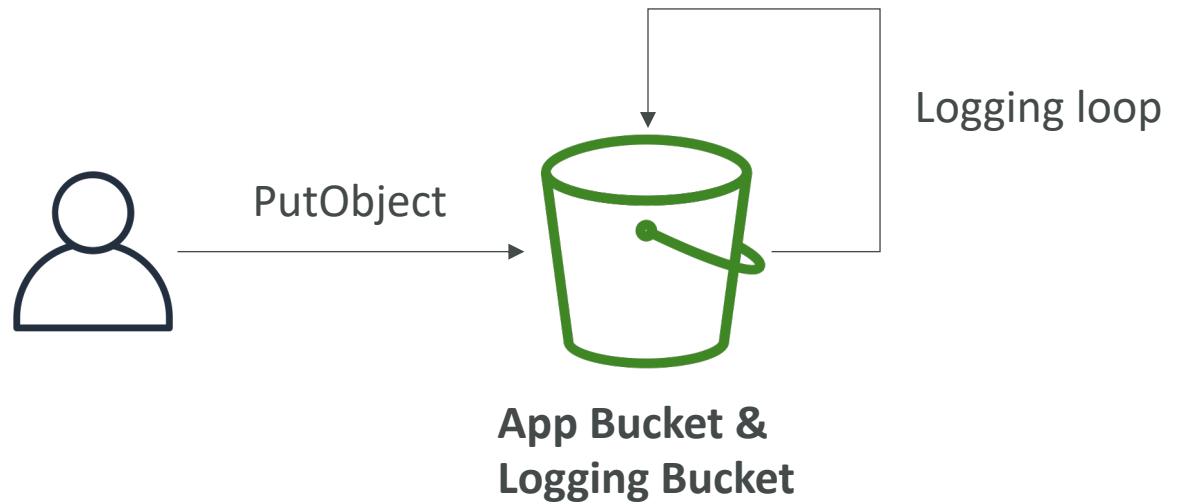
# S3 Access Logs

- 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...
- Or Amazon Athena as we'll see later in this section!
- The log format is at:  
<https://docs.aws.amazon.com/AmazonS3/latest/dev/LogFileFormat.html>



# S3 Access Logs: Warning

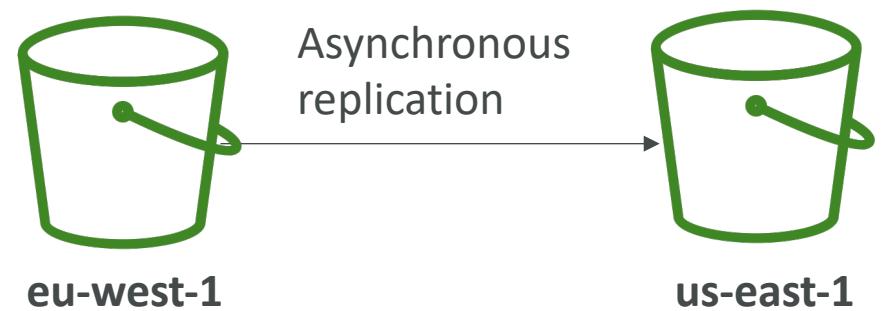
- Do not set your logging bucket to be the monitored bucket
- It will create a logging loop, and your bucket will grow in size exponentially



Do not try this at home 😊

# S3 Replication (CRR & SRR)

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



# S3 Replication – Notes

- After activating, 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 Pre-Signed URLs

- Can generate pre-signed URLs using SDK or CLI
  - For downloads (easy, can use the CLI)
  - For uploads (harder, must use the SDK)
- Valid for a default of 3600 seconds, can change timeout with --expires-in [TIME\_BY\_SECONDS] argument
- Users given a pre-signed URL inherit the permissions of the person who generated the URL for GET / PUT
- Examples :
  - Allow only logged-in users to download a premium video on 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 our bucket

# 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

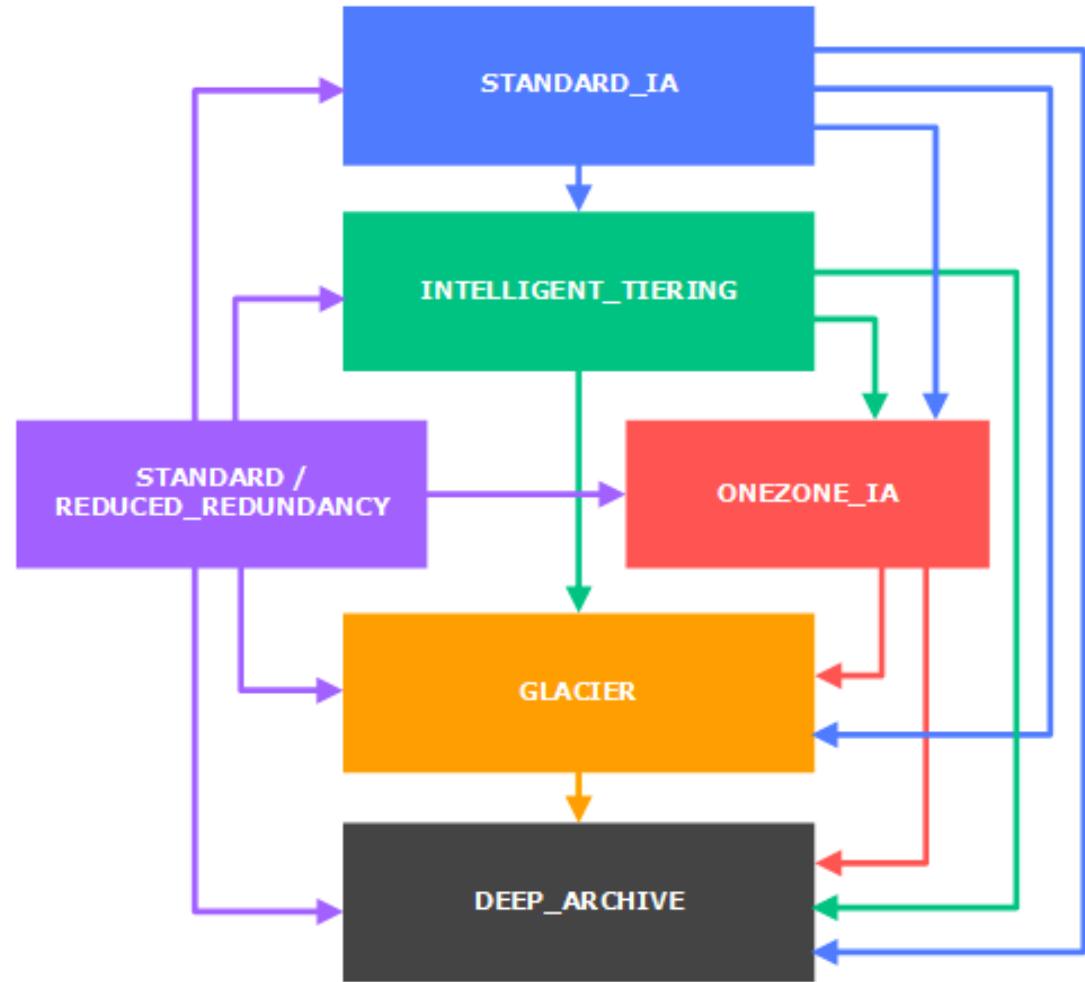
Example: us-east-1

	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)	<b>GET:</b> \$0.0004 <b>POST:</b> \$0.005	<b>GET:</b> \$0.0004 <b>POST:</b> \$0.005	<b>GET:</b> \$0.001 <b>POST:</b> \$0.01	<b>GET:</b> \$0.001 <b>POST:</b> \$0.01	<b>GET:</b> \$0.01 <b>POST:</b> \$0.02	<b>GET:</b> \$0.0004 <b>POST:</b> \$0.03  <b>Expedited:</b> \$10 <b>Standard:</b> \$0.05 <b>Bulk:</b> free	<b>GET:</b> \$0.0004 <b>POST:</b> \$0.05  <b>Standard:</b> \$0.10 <b>Bulk:</b> \$0.025
Retrieval Time	Instantaneous						<b>Expedited</b> (1 – 5 mins) <b>Standard</b> (3 – 5 hours) <b>Bulk</b> (5 – 12 hours)
Monitoring Cost (pet 1000 objects)		\$0.0025					

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

# S3 – Moving between storage classes

- You can transition objects between storage classes
- For infrequently accessed object, move them to STANDARD\_IA
- For archive objects you don't need in real-time, GLACIER or DEEP\_ARCHIVE
- Moving objects can be automated using a **lifecycle configuration**



# S3 Lifecycle Rules

- **Transition actions:** It defines when objects are transitioned 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 (ex - s3://mybucket/mp3/\*)
- Rules can be created for certain objects tags (ex - Department: Finance)

# 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 45 days. The source images should be able to be immediately retrieved for these 45 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 45 days.
- S3 thumbnails can be on ONEZONE\_IA, with a lifecycle configuration to expire them (delete them) after 45 days.

# S3 Lifecycle Rules – Scenario 2

- A rule in your company states that you should be able to recover your deleted S3 objects immediately for 15 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.
- You need to 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
- You can transition these “noncurrent versions” of the object to S3\_IA
- You can transition afterwards these “noncurrent versions” to DEEP\_ARCHIVE

# S3 Analytics – Storage Class Analysis

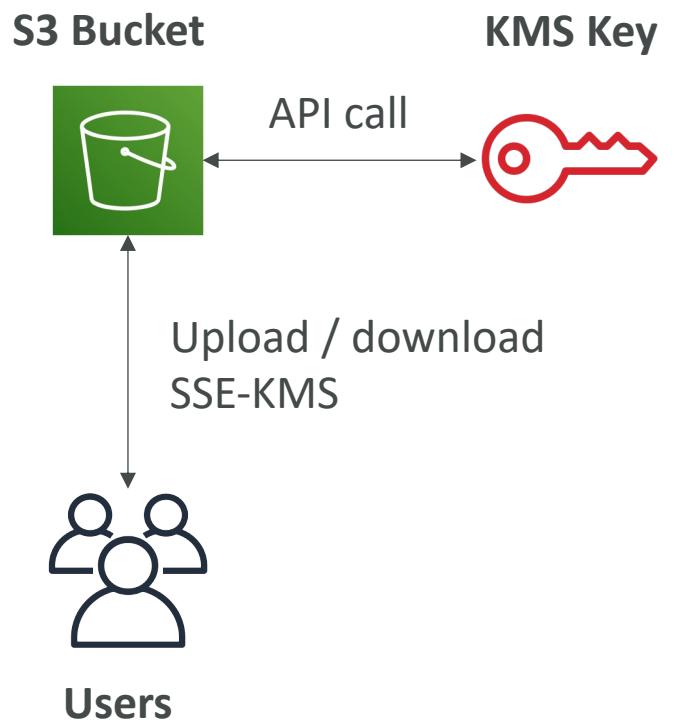
- You can setup S3 Analytics to help determine when to transition objects from Standard to Standard\_IA
- Does not work for ONEZONE\_IA or GLACIER
- Report is updated daily
- Takes about 24h to 48h hours to first start
- Good first step to put together Lifecycle Rules (or improve them)!

# 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 and 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

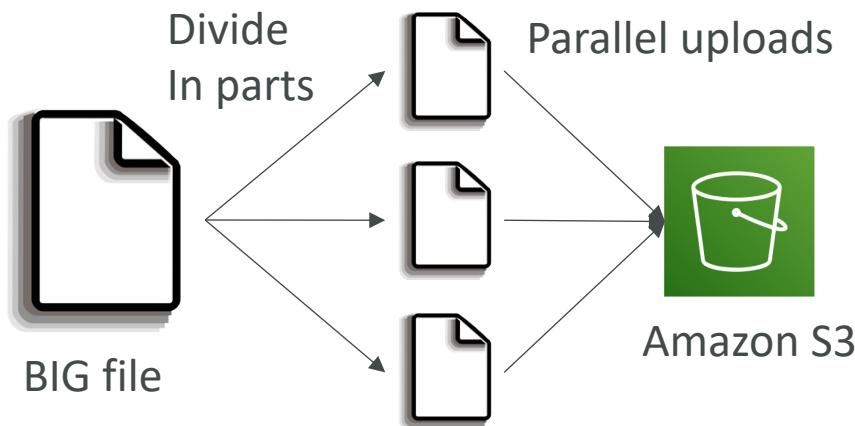
# S3 – KMS Limitation

- 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



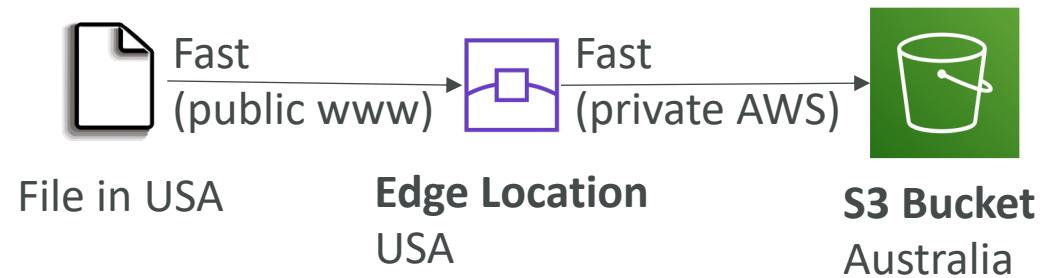
# S3 Performance

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



- S3 Transfer Acceleration

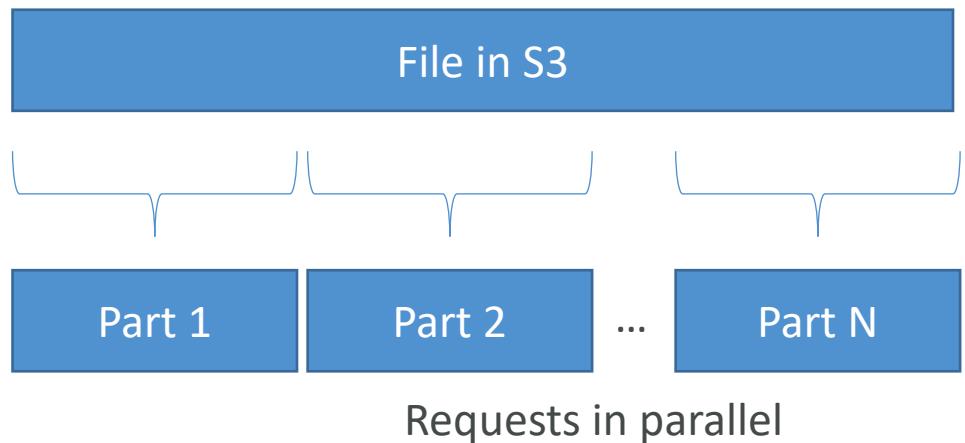
- Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region
- Compatible with multi-part upload



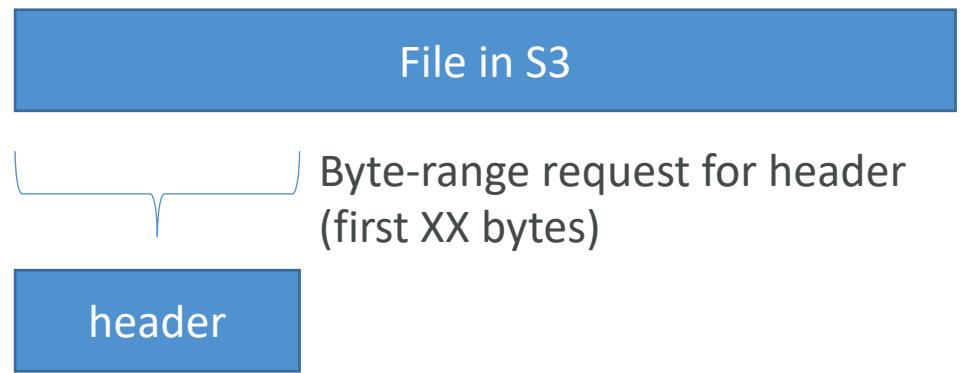
# S3 Performance – S3 Byte-Range Fetches

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

Can be used to speed up downloads

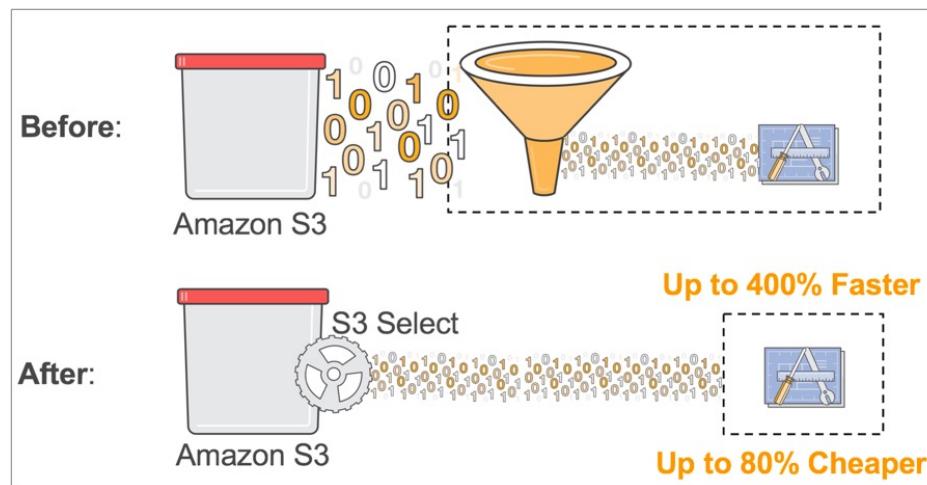


Can be used to retrieve only partial data (for example the head of a file)



# 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

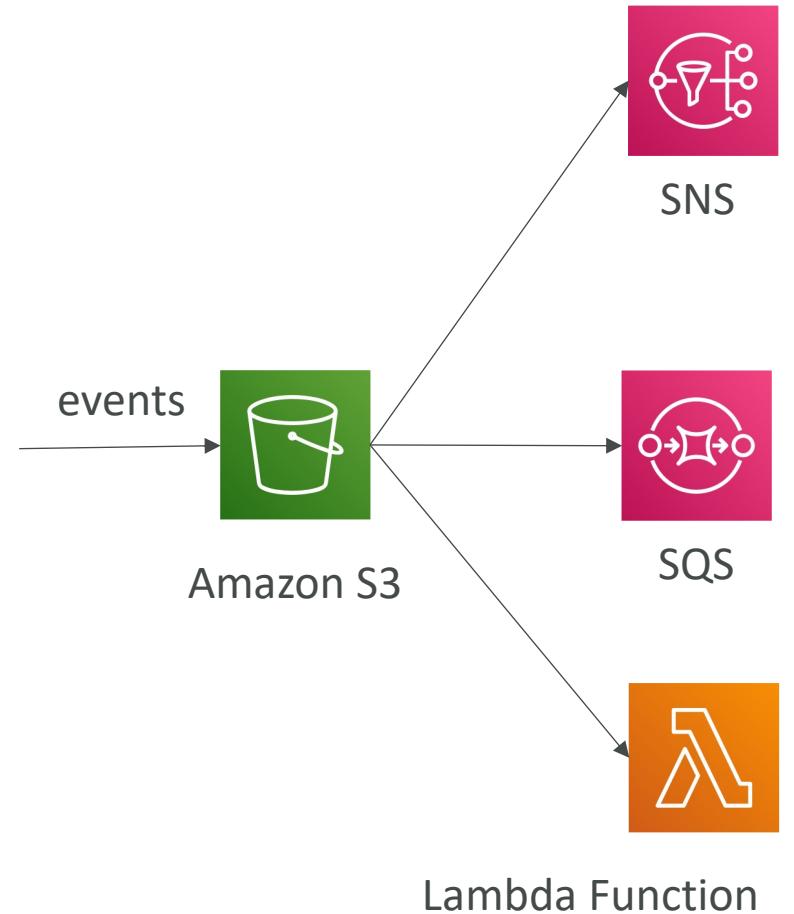


<https://aws.amazon.com/blogs/aws/s3-glacier-select/>

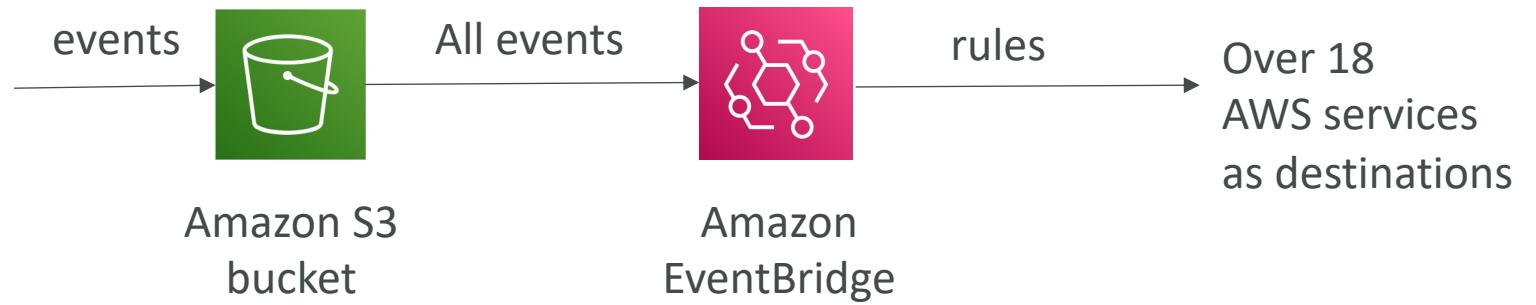


# S3 Event Notifications

- 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 with Amazon EventBridge

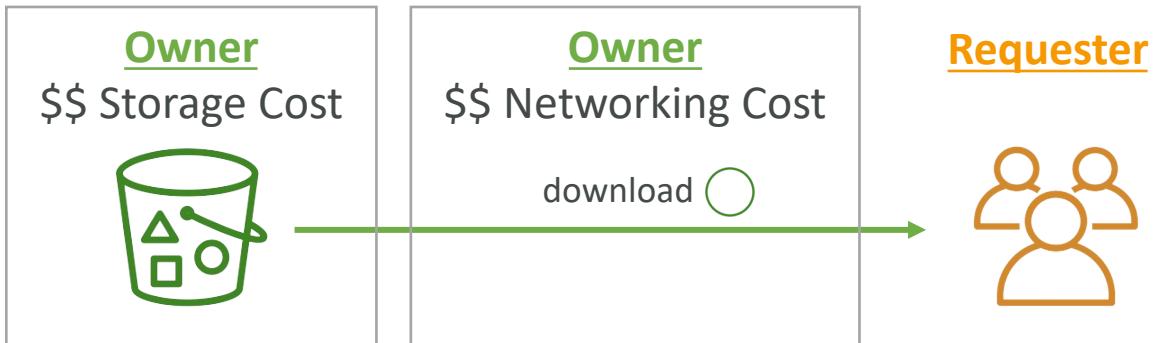


- Advanced filtering options with JSON rules (metadata, object size, name...)
- Multiple Destinations – ex Step Functions, Kinesis Streams / Firehose...
- EventBridge Capabilities – Archive, Replay Events, Reliable delivery

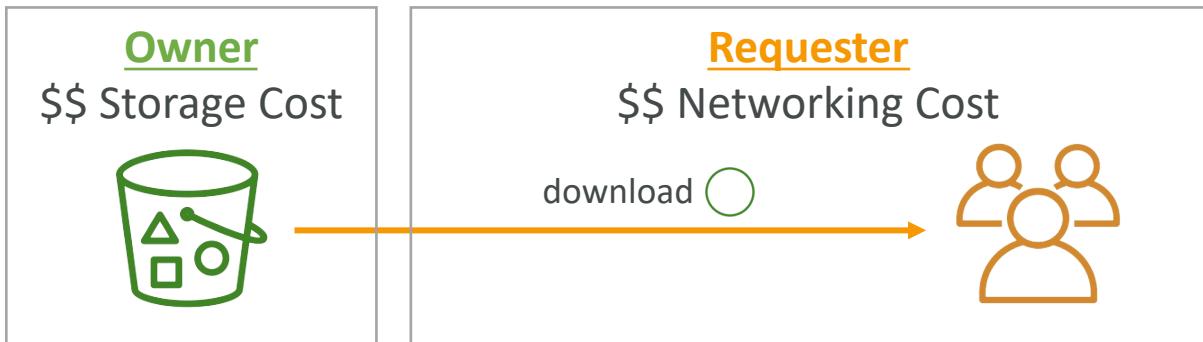
# S3 – Requester Pays

- 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**

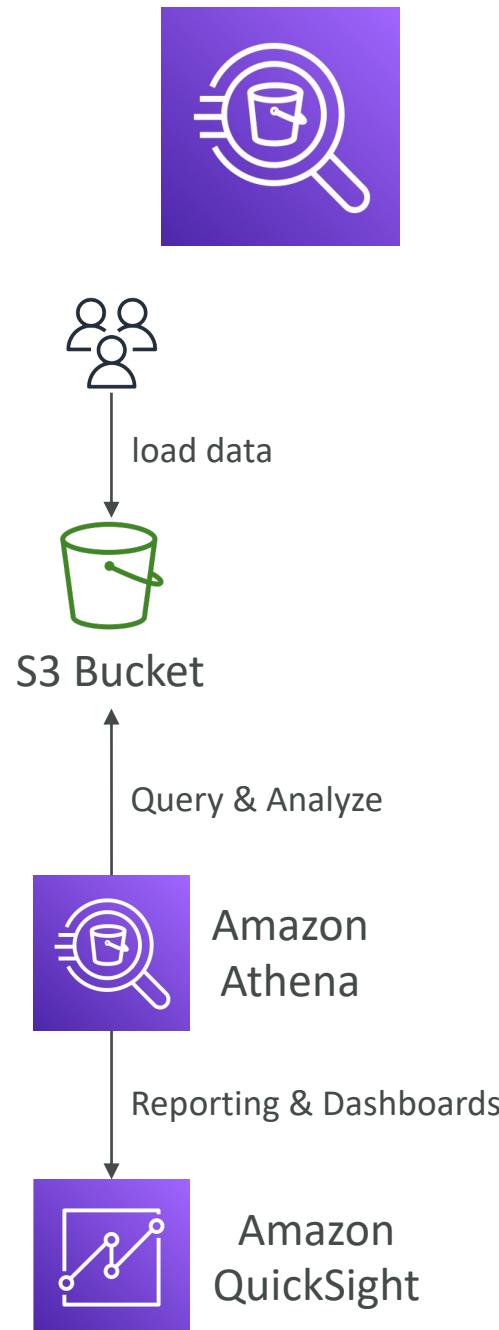


**Requester Pays Bucket**



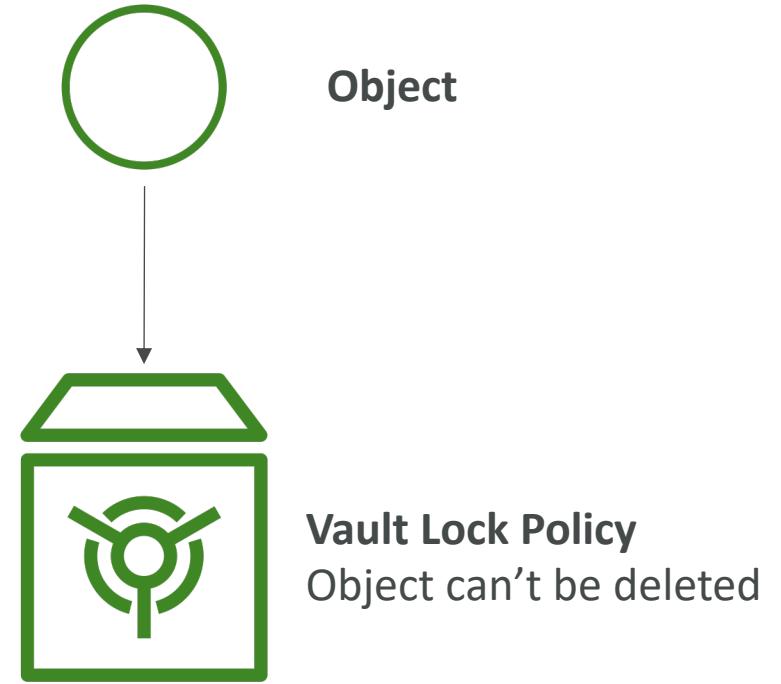
# Amazon Athena

- Serverless query service to **perform analytics against S3 objects**
- Uses standard SQL language to query the files
- Supports CSV, JSON, ORC, Avro, and Parquet (built on Presto)
- Pricing: \$5.00 per TB of data scanned
- Use compressed or columnar data for cost-savings (less scan)
- Use cases: Business intelligence / analytics / reporting, analyze & query VPC Flow Logs, ELB Logs, CloudTrail trails, etc...
- Exam Tip: analyze data in S3 using serverless SQL, use Athena



# Glacier Vault Lock

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



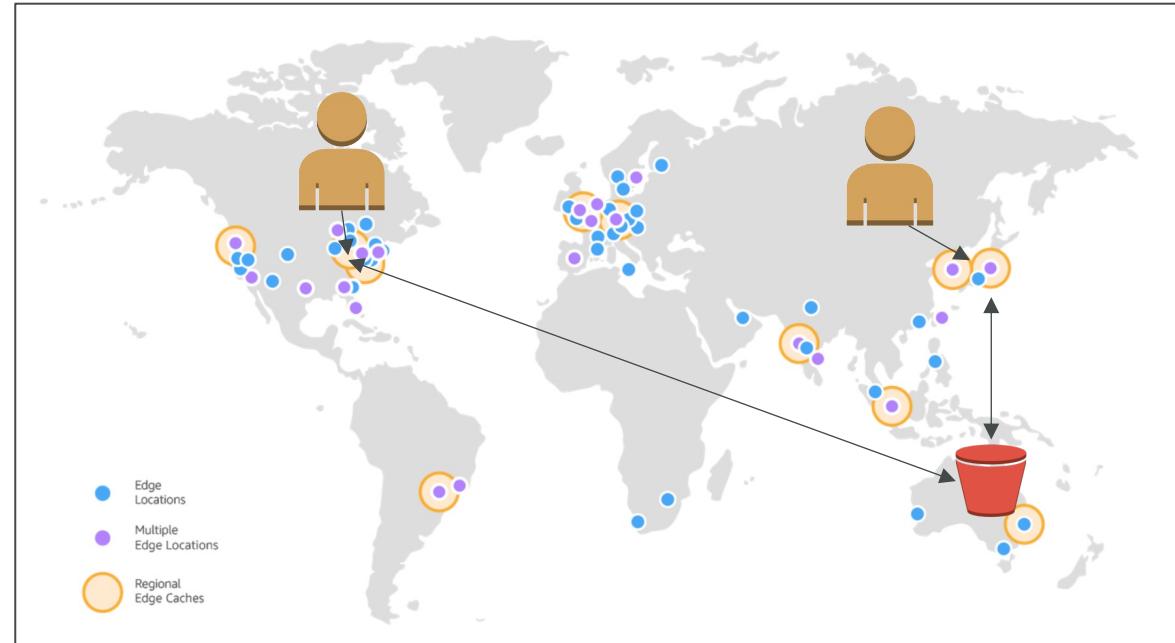
# S3 Object Lock (versioning must be enabled)

- Adopt a WORM (Write Once Read Many) model
- Block an object version deletion for a specified amount of time
- Object retention:
  - **Retention Period:** specifies a fixed period
  - **Legal Hold:** same protection, no expiry date
- Modes:
  - **Governance mode:** users can't overwrite or delete an object version or alter its lock settings unless they have special permissions
  - **Compliance mode:** a protected object version can't be overwritten or deleted by any user, including the root user in your AWS account. When an object is locked in compliance mode, its retention mode can't be changed, and its retention period can't be shortened.

# AWS CloudFront



- Content Delivery Network (CDN)
- Improves read performance, content is cached at the edge
- 216 Point of Presence globally (edge locations)
- DDoS protection, integration with Shield, AWS Web Application Firewall
- Can expose external HTTPS and can talk to internal HTTPS backends

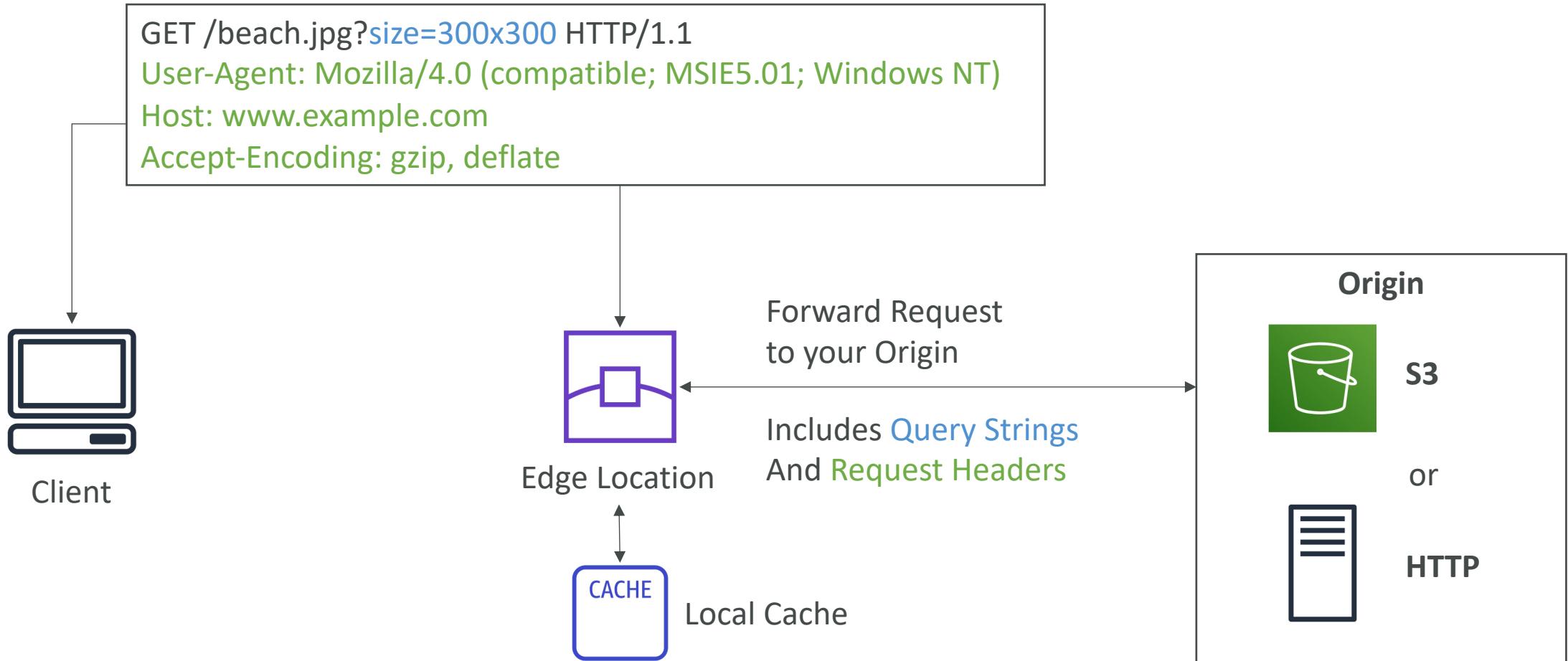


Source: <https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2>

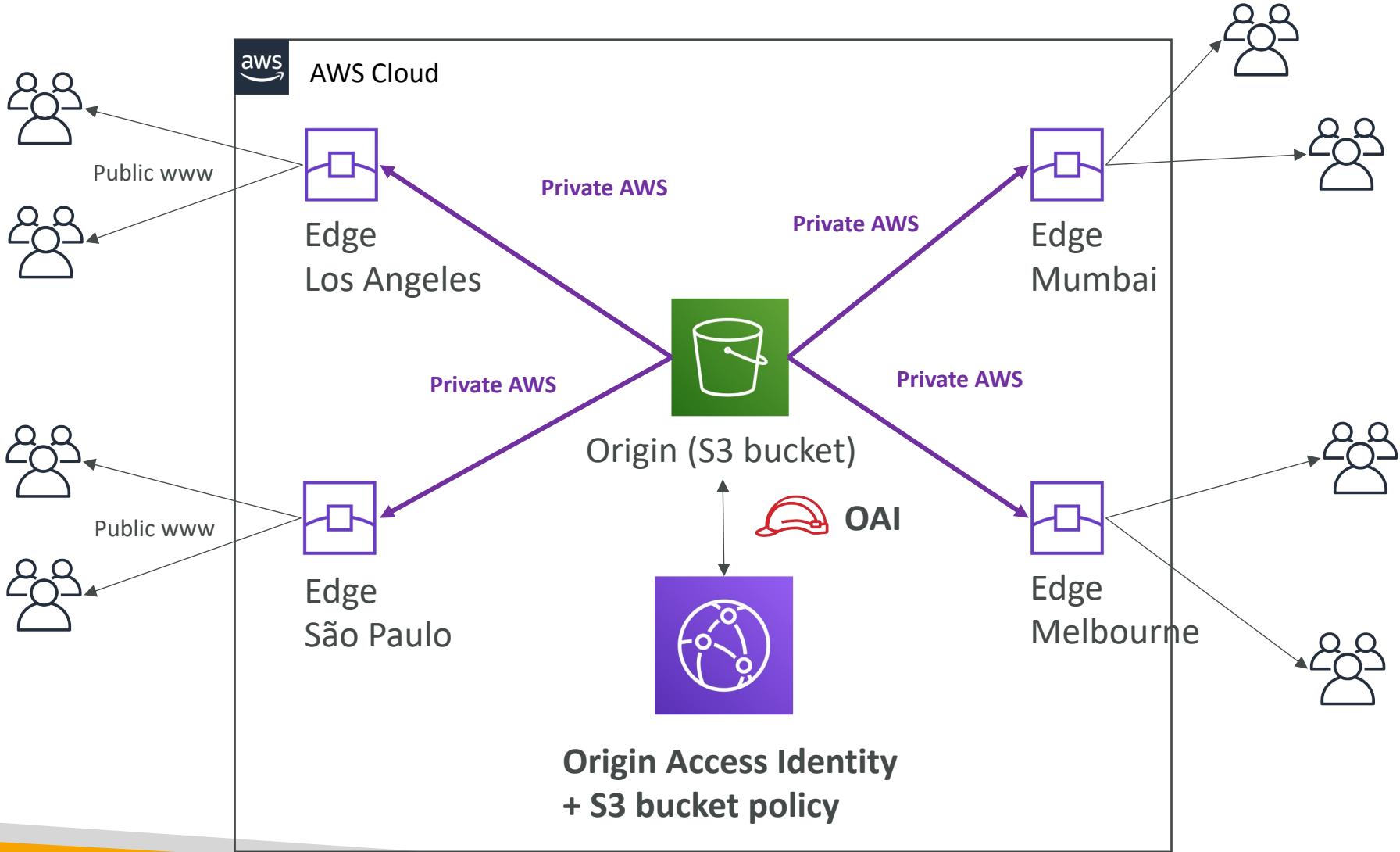
# CloudFront – Origins

- S3 bucket
  - For distributing files and caching them at the edge
  - Enhanced security with CloudFront Origin Access 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

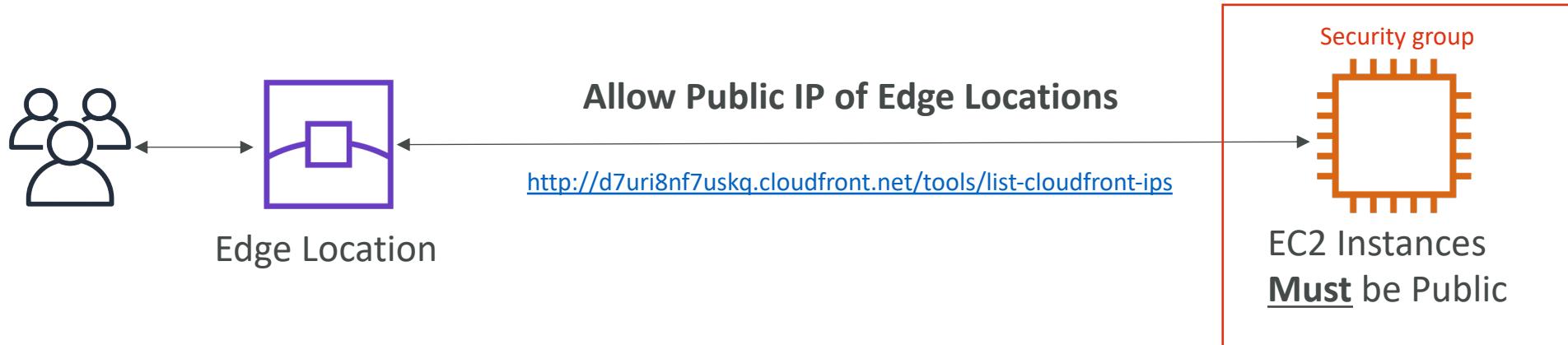
# CloudFront at a high level



# CloudFront – S3 as an Origin



# CloudFront – ALB or EC2 as an origin



# CloudFront Geo Restriction

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

# 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

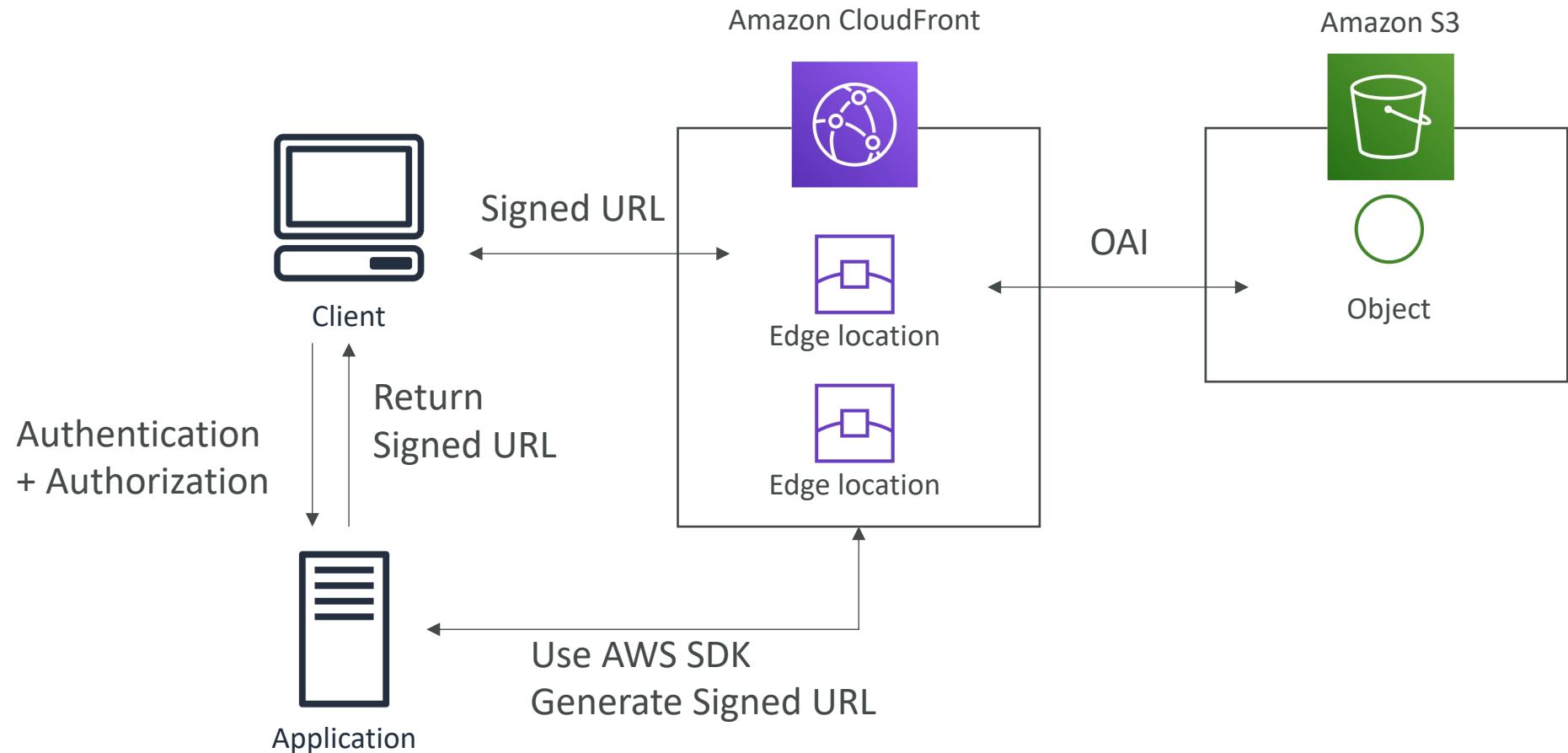
# AWS CloudFront Hands On

- We'll create an S3 bucket
- We'll create a CloudFront distribution
- We'll create an Origin Access Identity
- We'll limit the S3 bucket to be accessed only using this identity

# CloudFront Signed URL / Signed Cookies

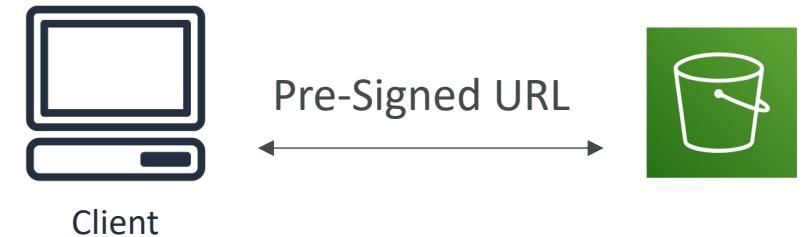
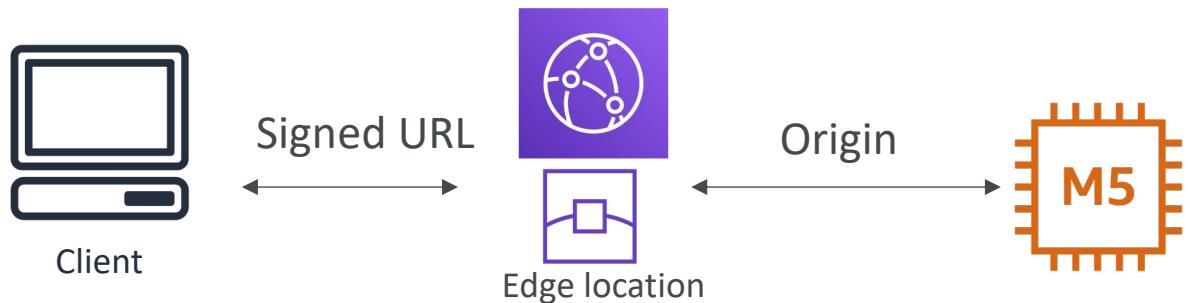
- You want to distribute paid shared content to premium users over the world
- We can use CloudFront Signed URL / Cookie. We attach a policy with:
  - Includes URL expiration
  - Includes IP ranges to access the data from
  - Trusted signers (which AWS accounts can create signed URLs)
- How long should the URL be valid for?
  - Shared content (movie, music): make it short (a few minutes)
  - Private content (private to the user): you can make it last for years
- Signed URL = access to individual files (one signed URL per file)
- Signed Cookies = access to multiple files (one signed cookie for many files)

# CloudFront Signed URL Diagram



# CloudFront Signed URL vs S3 Pre-Signed URL

- CloudFront Signed URL:
  - Allow access to a path, no matter the origin
  - Account wide key-pair, only the root can manage it
  - Can filter by IP, path, date, expiration
  - Can leverage caching features
- S3 Pre-Signed URL:
  - Issue a request as the person who pre-signed the URL
  - Uses the IAM key of the signing IAM principal
  - Limited lifetime



# 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

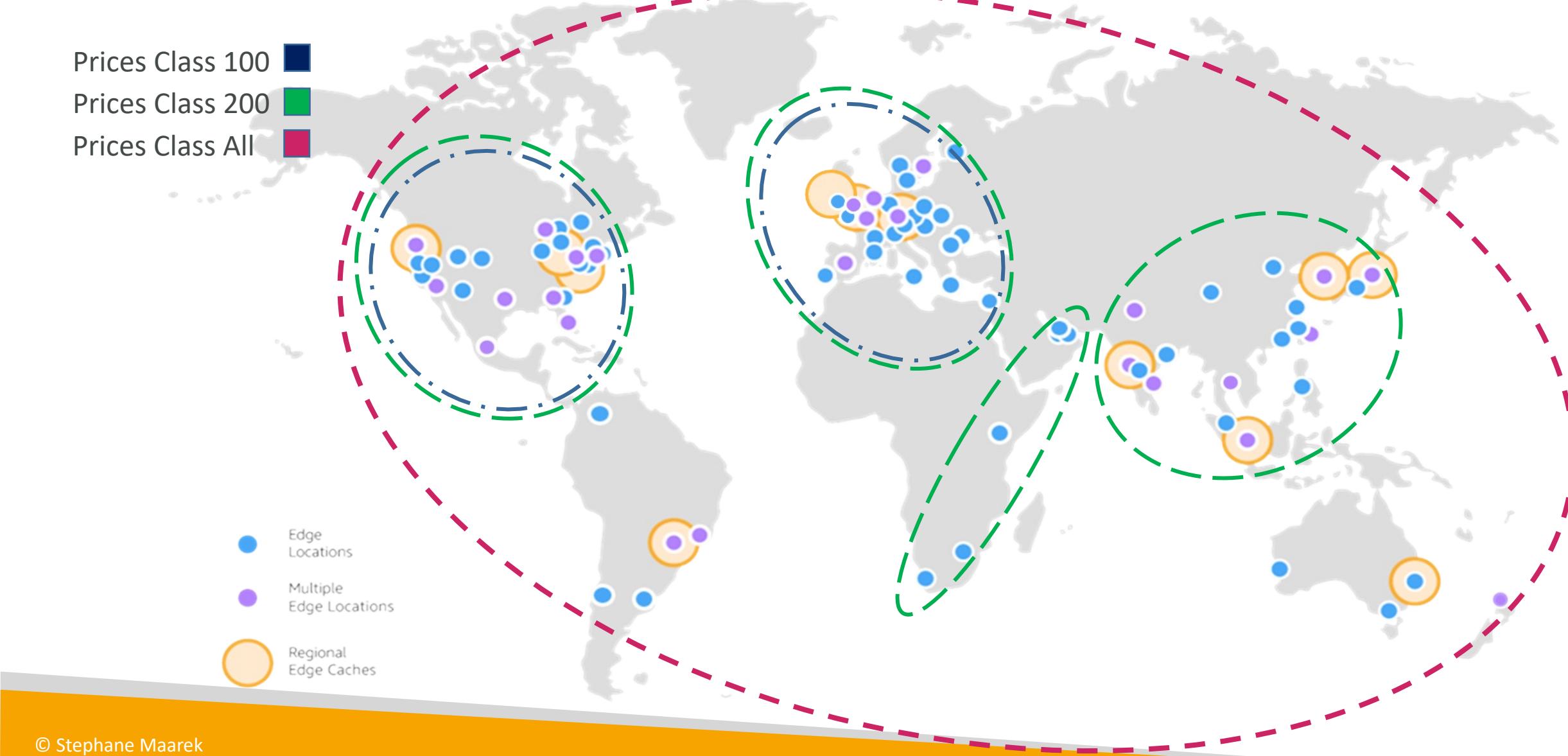
Edge Locations Included Within	United States, Mexico, & Canada	Europe & Israel	South Africa, Kenya, & Middle East	South America	Japan	Australia & New Zealand	Hong Kong, Philippines, Singapore, South Korea, Taiwan, & Thailand	India
Price Class All	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Price Class 200	Yes	Yes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

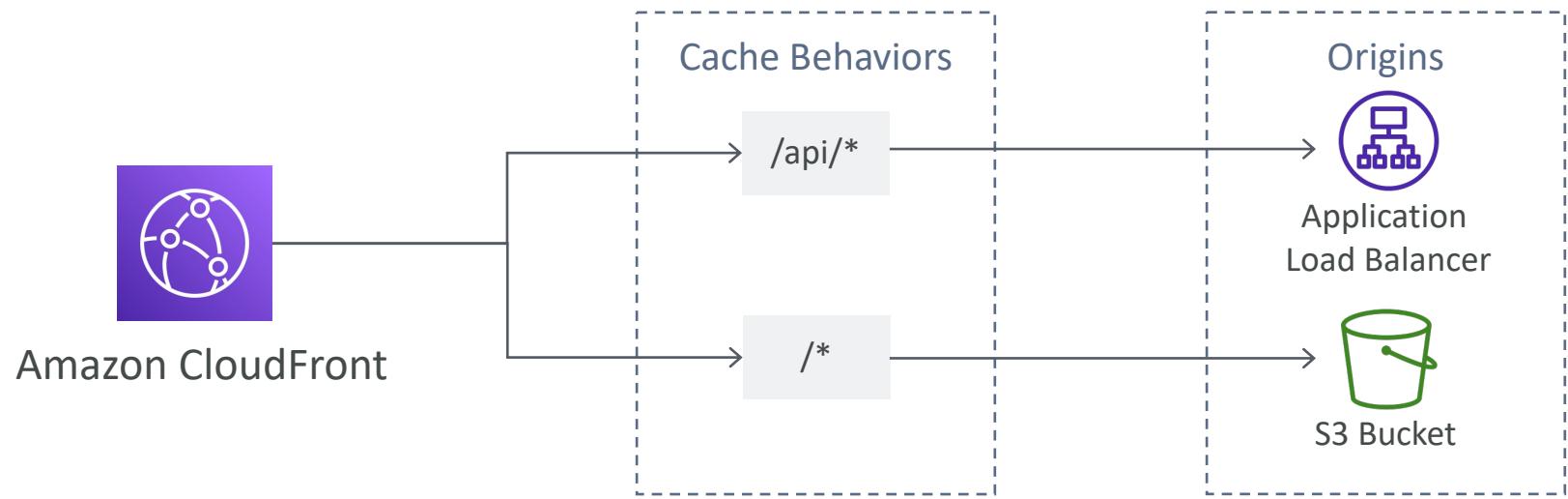


- Edge Locations
- Multiple Edge Locations
- Regional Edge Caches



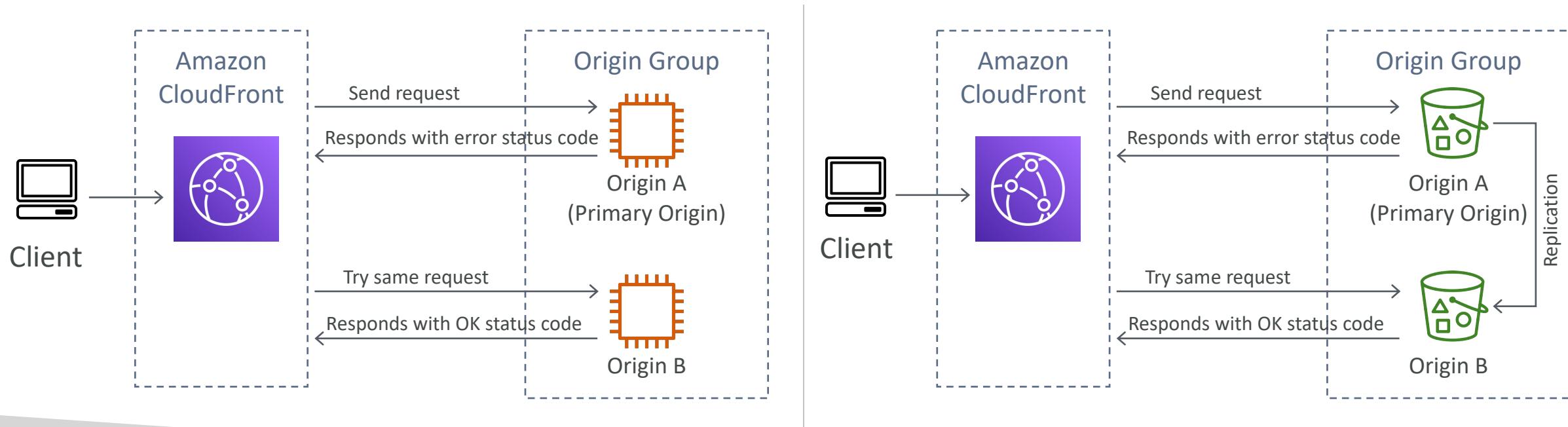
# CloudFront – Multiple Origin

- To route to different kind of origins based on the content type
- Based on path pattern:
  - /images/\*
  - /api/\*
  - /\*



# CloudFront – Origin Groups

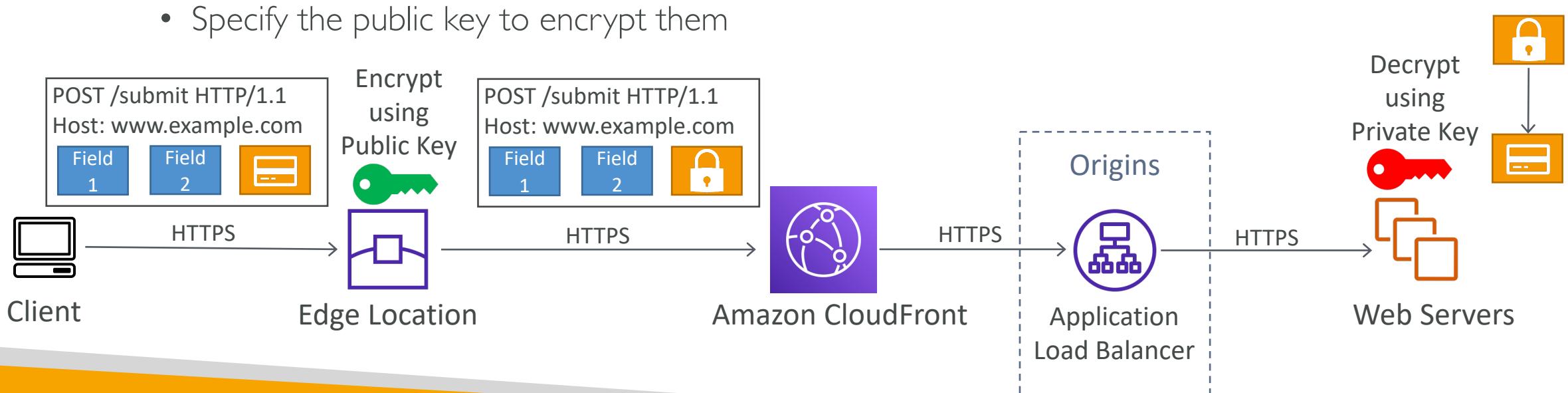
- To increase high-availability and do failover
- Origin Group: one primary and one secondary origin
- If the primary origin fails, the second one is used



S3 + CloudFront – Region-level High Availability

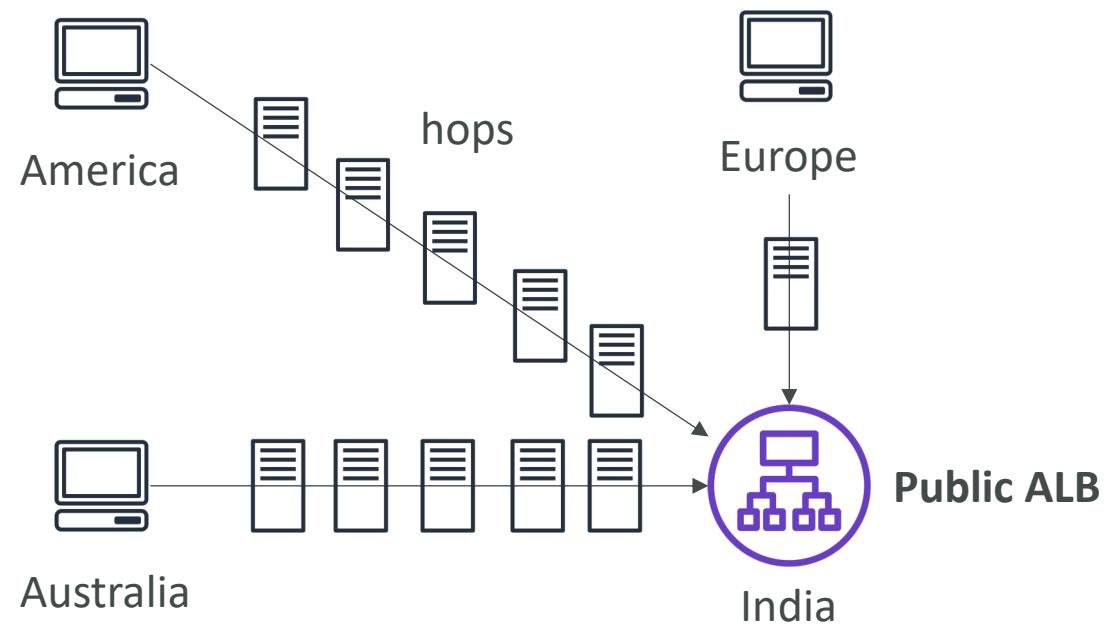
# CloudFront – Field Level Encryption

- Protect user sensitive information through application stack
- Adds an additional layer of security along with HTTPS
- Sensitive information encrypted at the edge close to user
- Uses asymmetric encryption
- Usage:
  - Specify set of fields in POST requests that you want to be encrypted (up to 10 fields)
  - Specify the public key to encrypt them



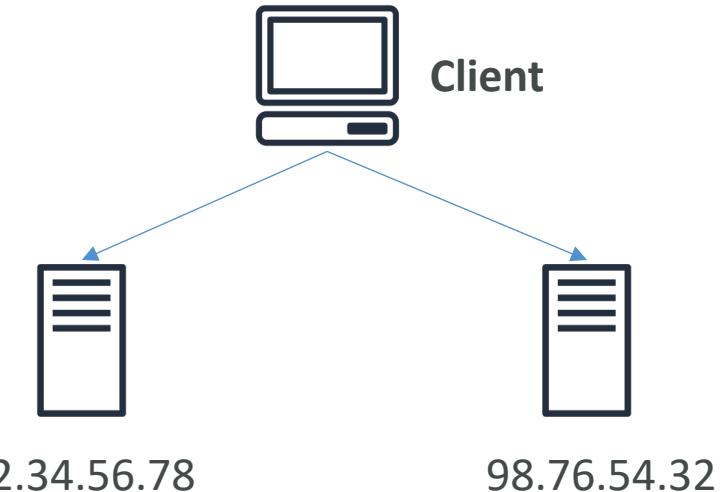
# Global users for our application

- 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

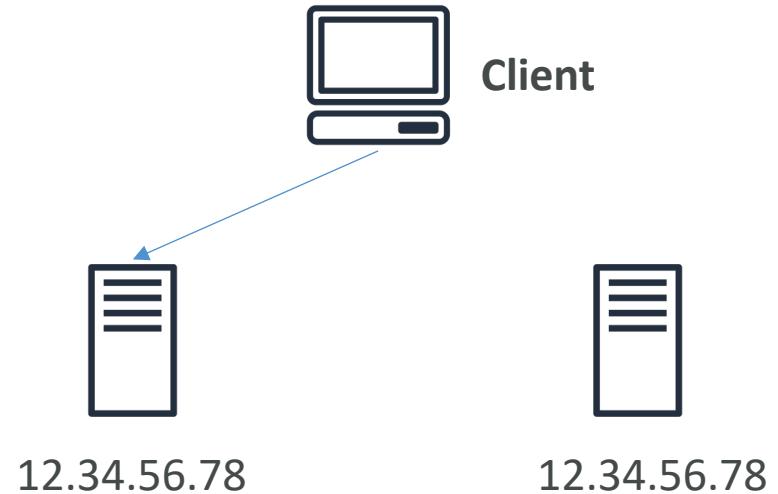


# Unicast IP vs Anycast IP

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



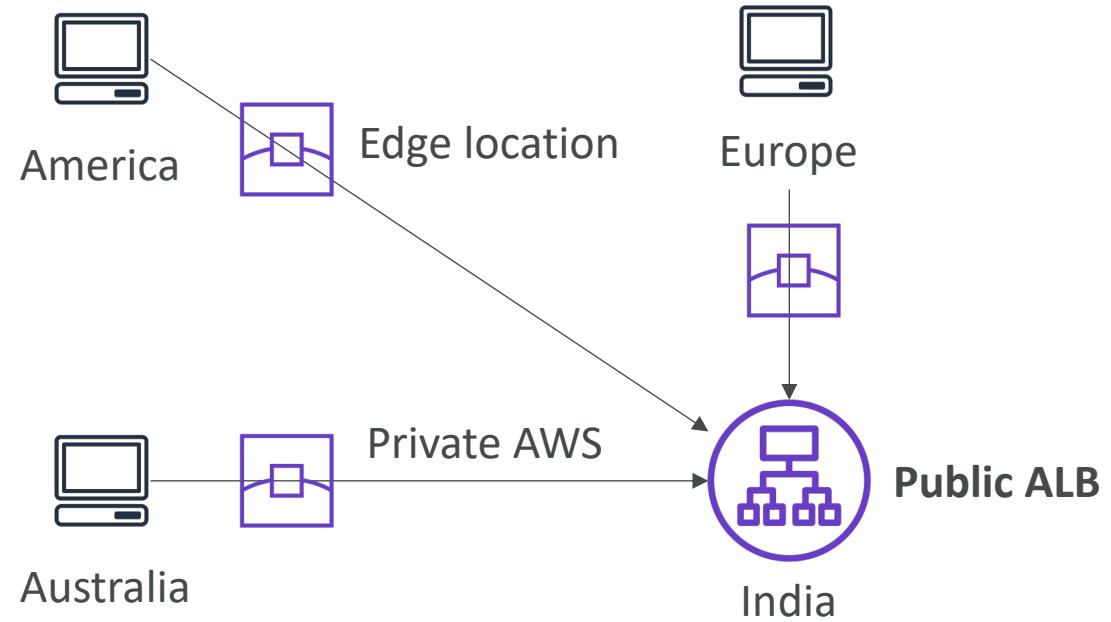
- **Anycast IP:** all servers hold the same IP address and the client is routed to the nearest one



# 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

# 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

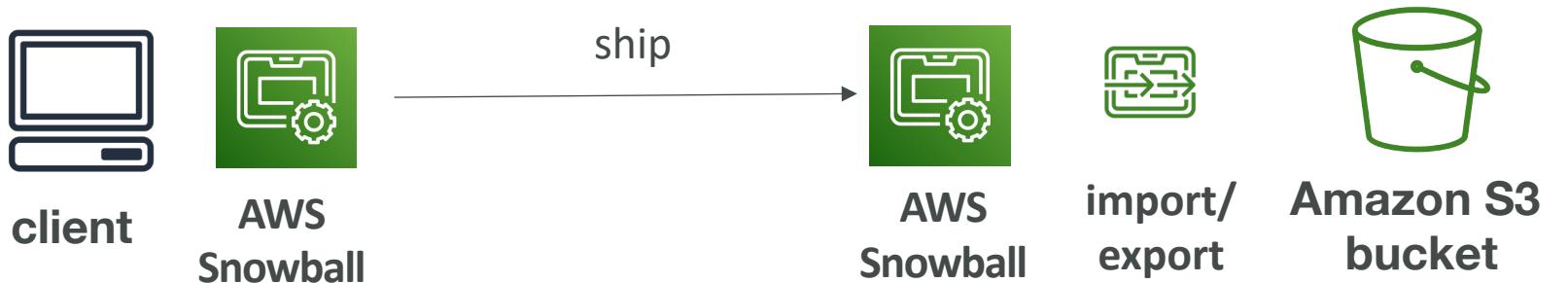
If it takes more than a week to transfer over the network, use Snowball devices!

# 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 capacity for block volume and S3 compatible object storage
- Use cases: large data cloud migrations, DC decommission, disaster recovery



# AWS Snowcone



- 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
- **8 TBs of usable 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**

	<b>Snowcone</b>	<b>Snowball Edge Storage Optimized</b>	<b>Snowmobile</b>
Storage Capacity	8 TB usable	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)

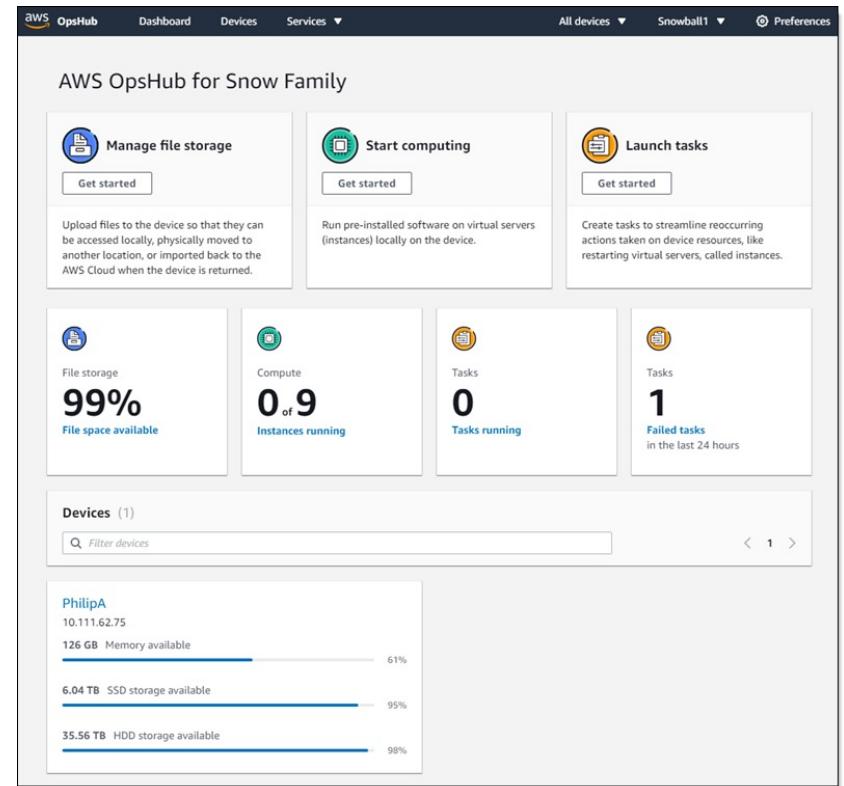
# Snow Family – Edge Computing

- Snowcone (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
  - 52 vCPUs, 208 GiB of RAM
  - Optional GPU (useful for video processing or machine learning)
  - 42 TB usable storage
- Snowball Edge – Storage Optimized
  - Up to 40 vCPUs, 80 GiB of RAM
  - 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**

# Amazon FSx for Windows (File Server)



- *EFS* is a shared POSIX system for Linux systems.
- **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
- 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)

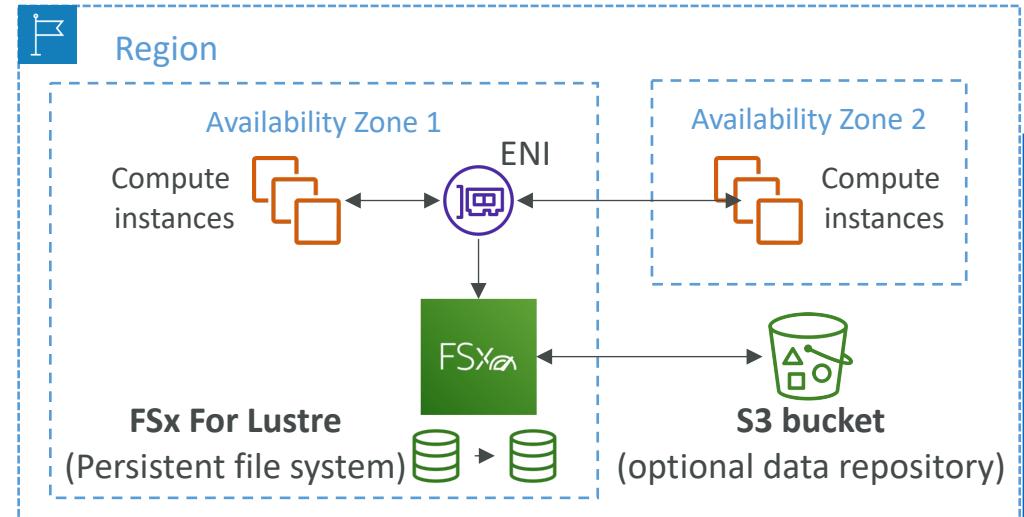
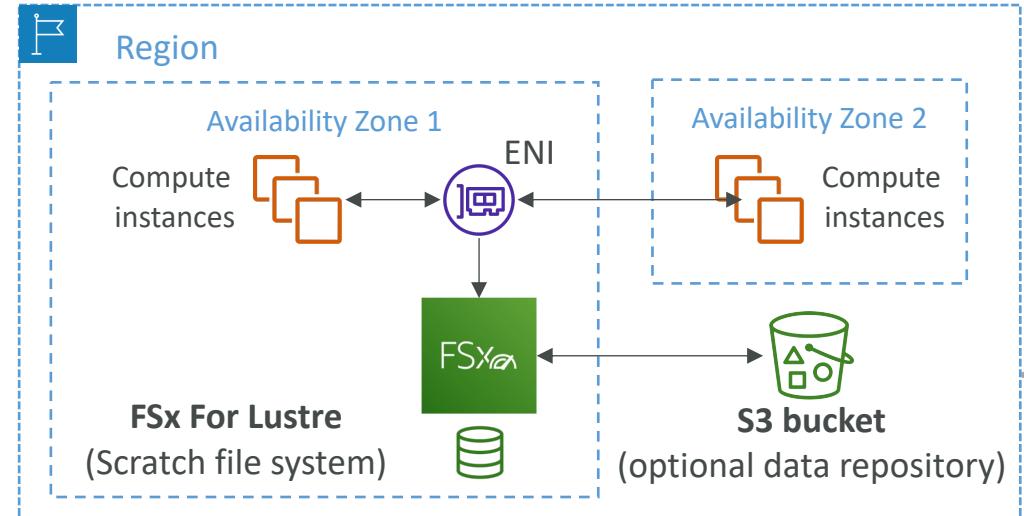
# FSx File System Deployment Options

## • 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



# 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



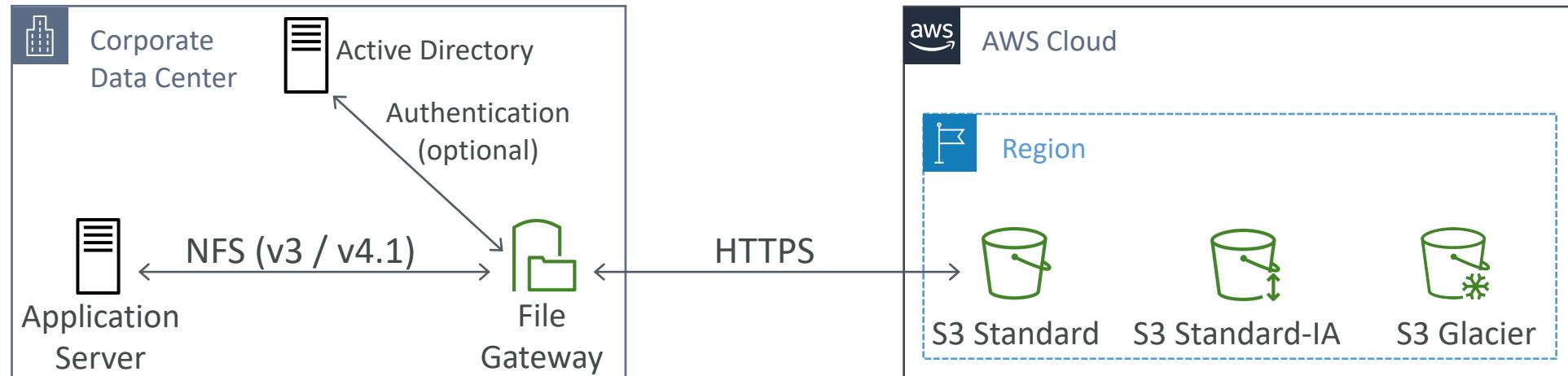
# AWS Storage Gateway

- Bridge between on-premises data and cloud data in S3
- Use cases: disaster recovery, backup & restore, tiered storage
- 3 types of Storage Gateway:
  - File Gateway
  - Volume Gateway
  - Tape Gateway
- Exam Tip: You need to know the differences between all 3!



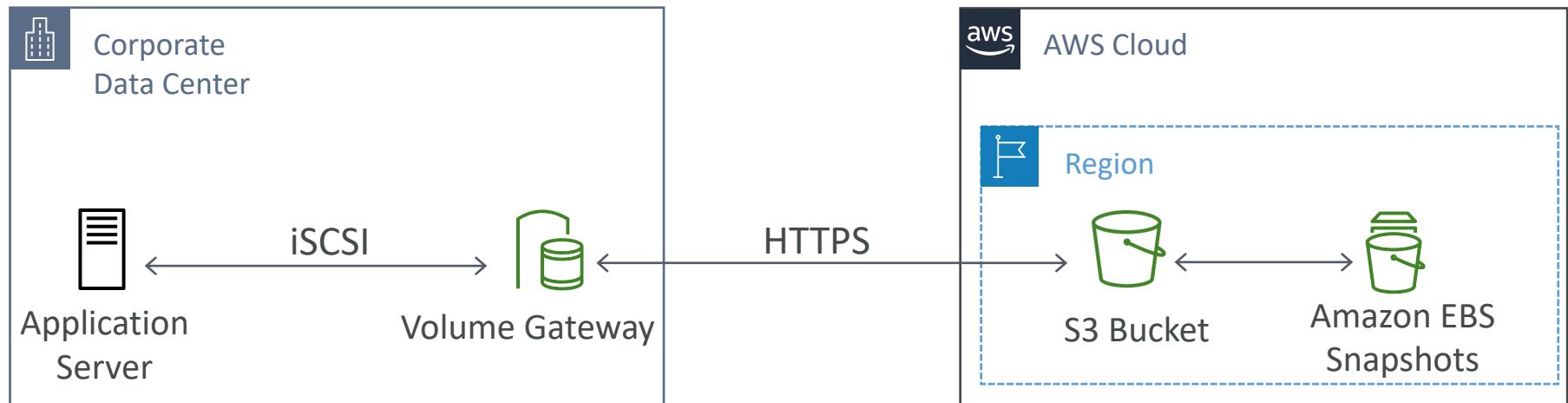
# File Gateway

- Configured S3 buckets are accessible using the NFS and SMB protocol
- Supports S3 standard, S3 IA, S3 One Zone IA
- Bucket access using IAM roles for each File Gateway
- Most recently used data is cached in the file gateway
- Can be mounted on many servers
- Integrated with Active Directory (AD) for user authentication



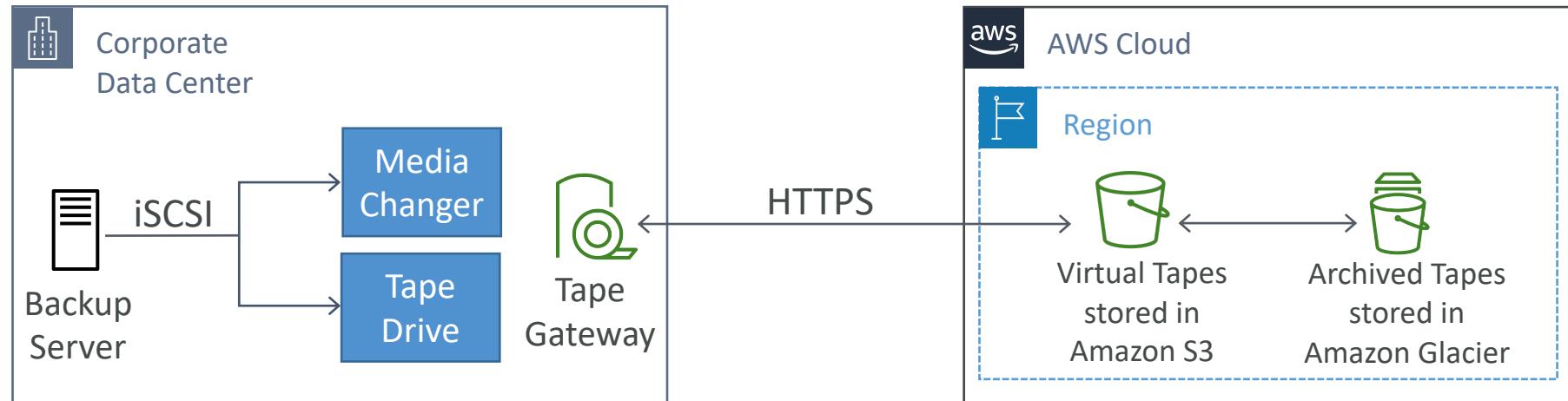
# Volume Gateway

- 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](#)

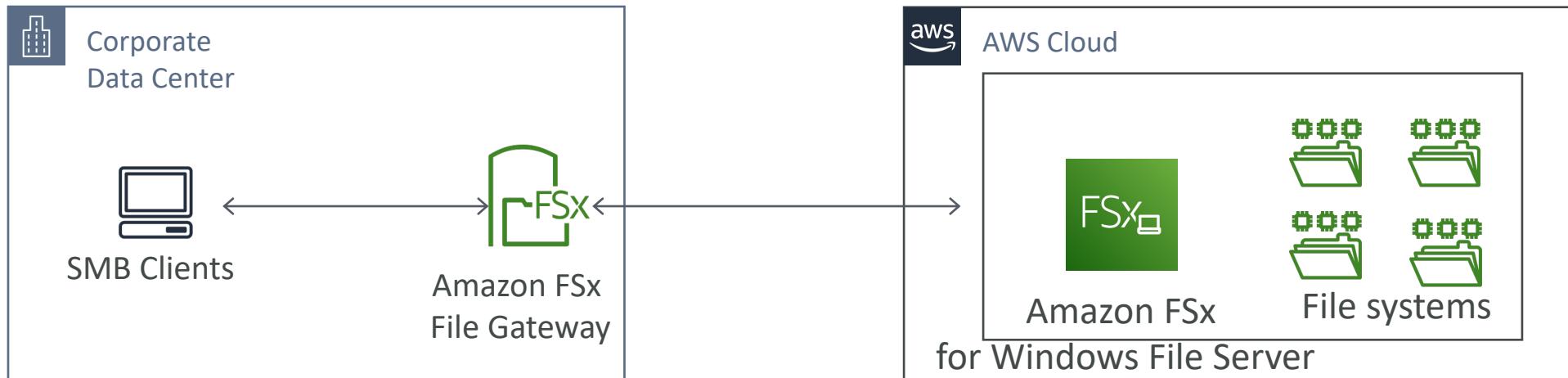


# AWS Storage Gateway Summary

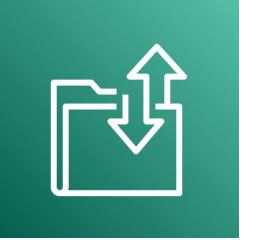
- Exam tip: Read the question well, it will hint at which gateway to use
- On-premises data to the cloud => Storage Gateway
- File access / NFS – user auth with Active Directory => File Gateway (backed by S3)
- Volumes / Block Storage / iSCSI => Volume gateway (backed by S3 with EBS snapshots)
- VTL Tape solution / Backup with iSCSI => Tape Gateway (backed by S3 and Glacier)
- No on-premises virtualization => Hardware Appliance

# 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

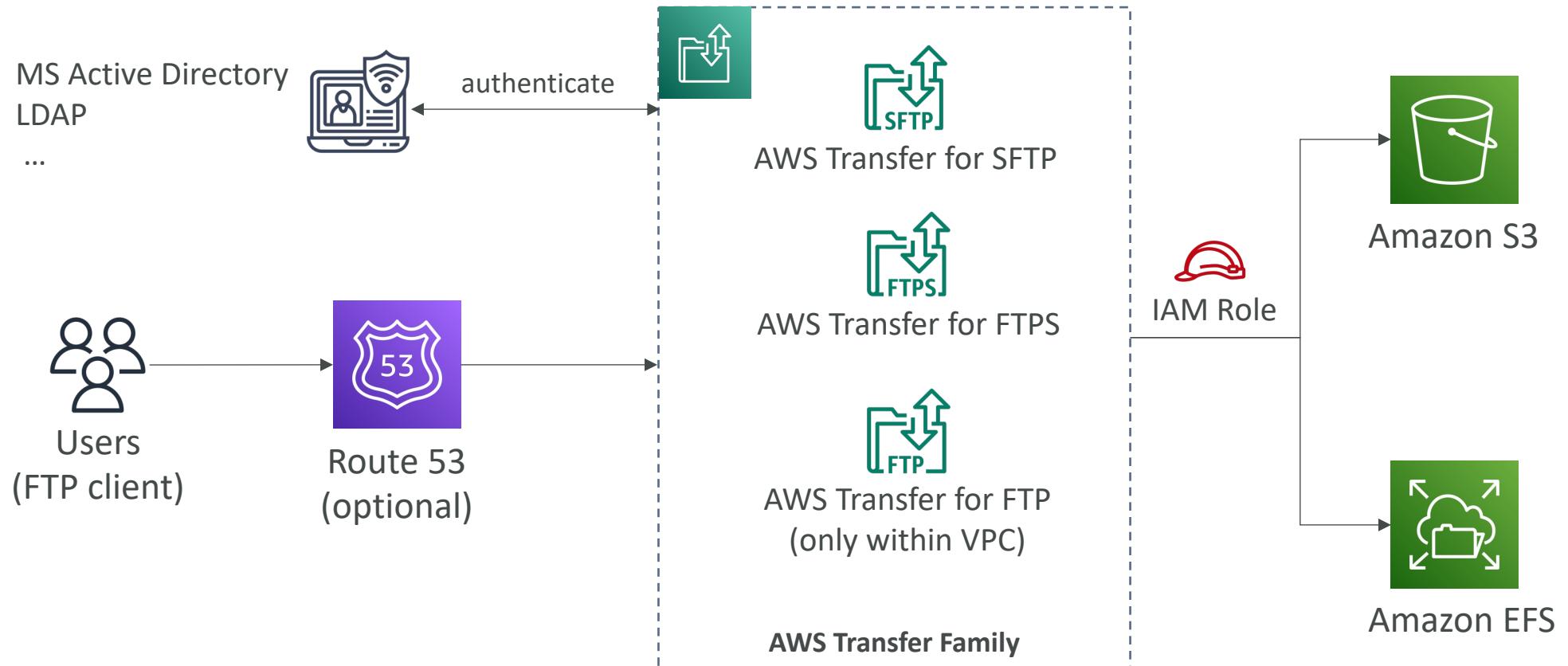


# AWS Transfer Family



- 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



# Storage Comparison

- **S3:** Object Storage
- **Glacier:** Object Archival
- **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
- **EBS volumes:** Network storage for one EC2 instance at a time
- **Instance Storage:** Physical storage for your EC2 instance (high IOPS)
- **Storage Gateway:** File Gateway, Volume Gateway (cache & stored), Tape Gateway
- **Snowball / Snowmobile:** to move large amount of data to the cloud, physically
- **Database:** for specific workloads, usually with indexing and querying

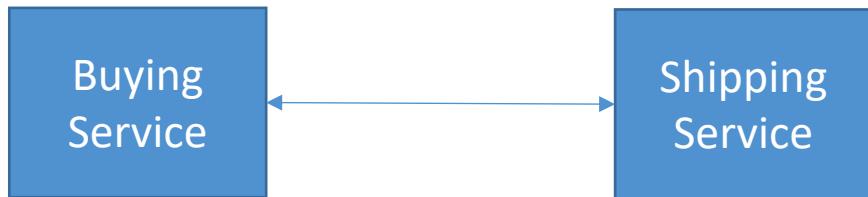
# AWS Integration & Messaging

SQS, SNS & Kinesis

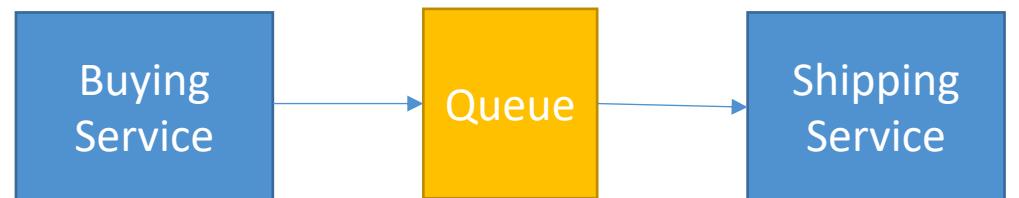
# Section Introduction

- 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)**

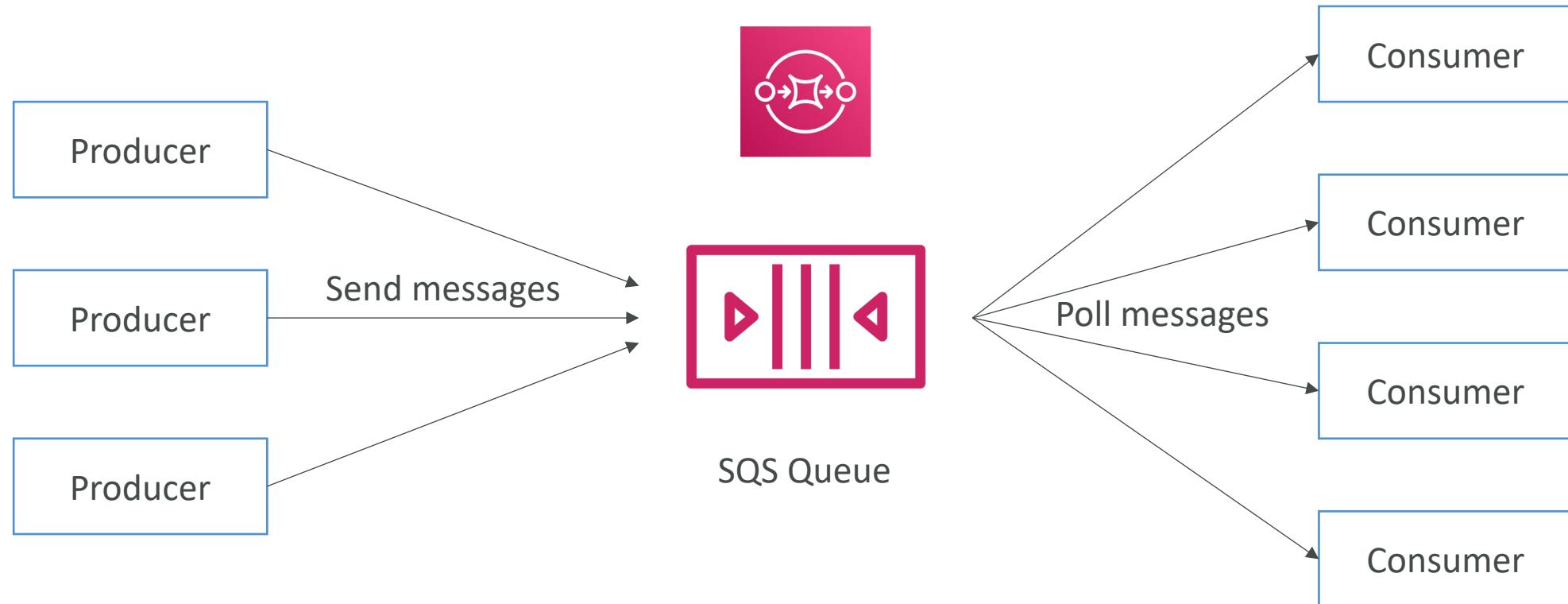


# Section Introduction

- 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?



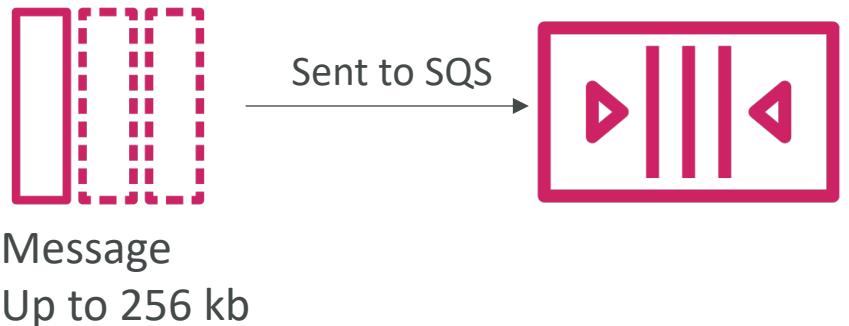
# Amazon SQS – Standard Queue



- 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)

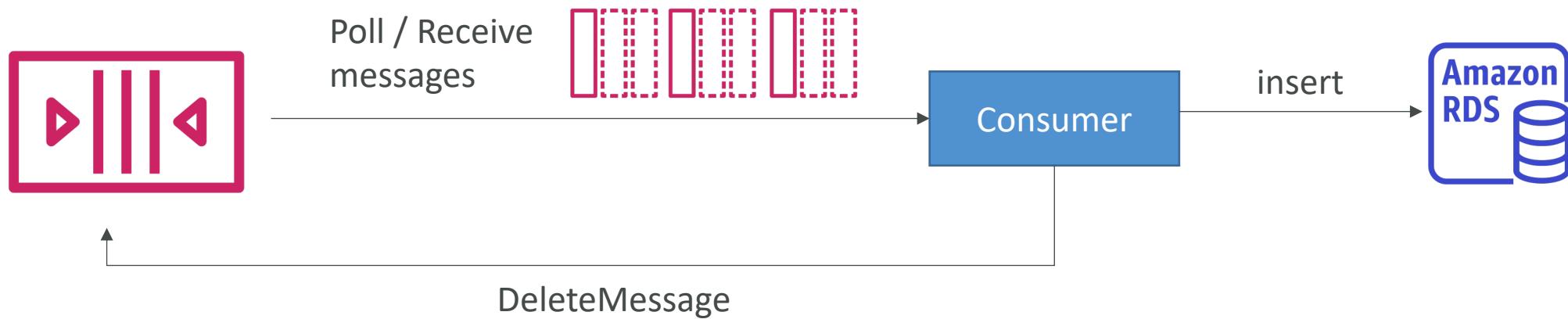
# SQS – Producing Messages

- 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

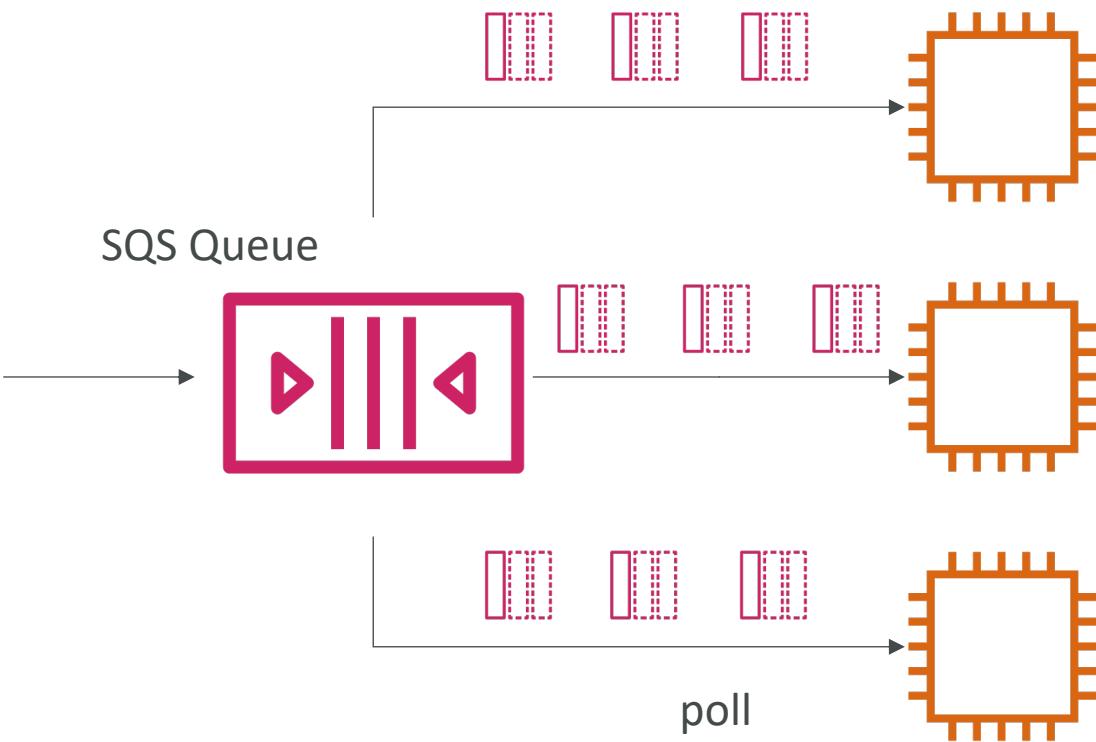


# SQS – Consuming Messages

- 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

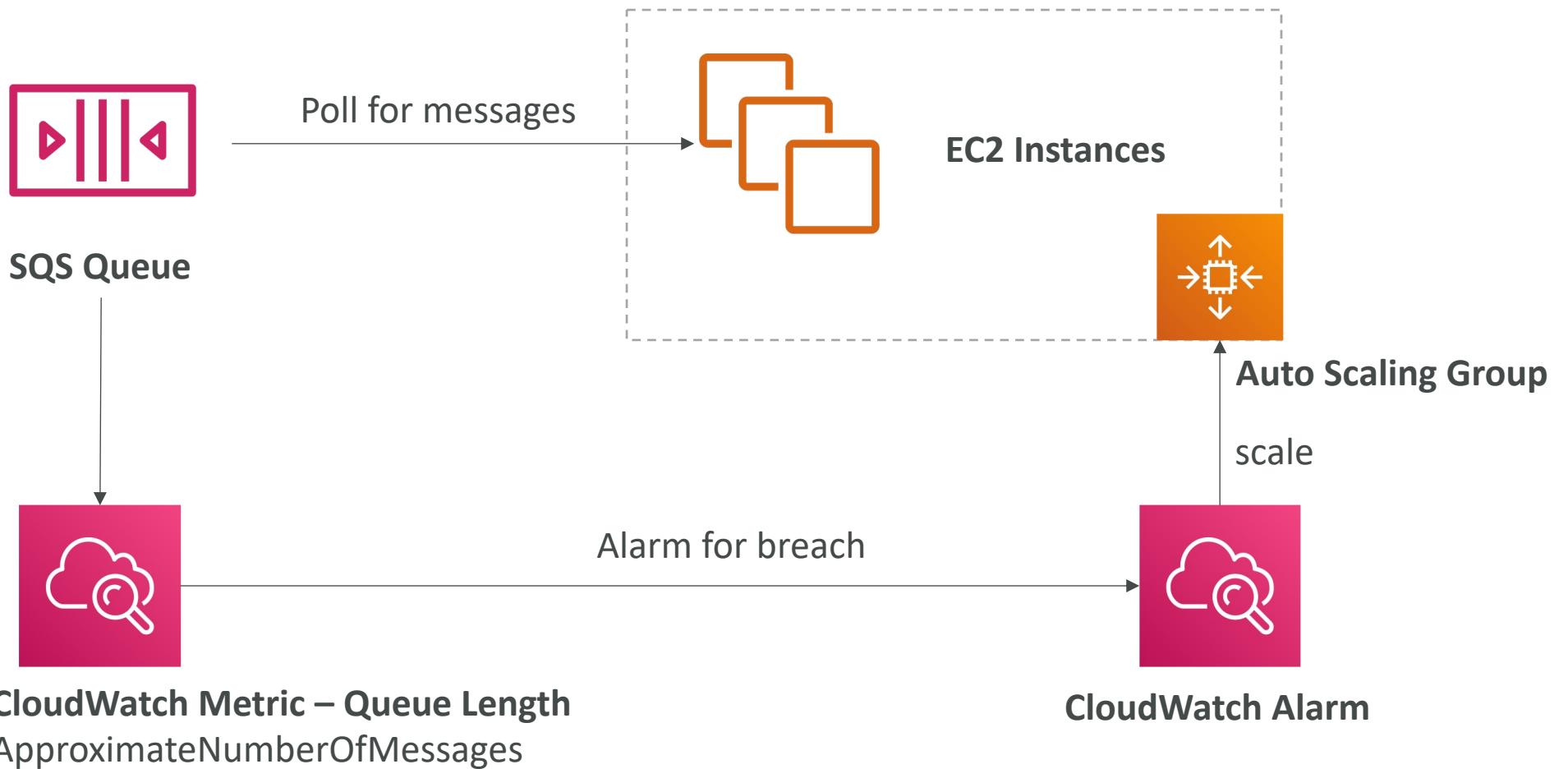


# SQS – Multiple EC2 Instances Consumers

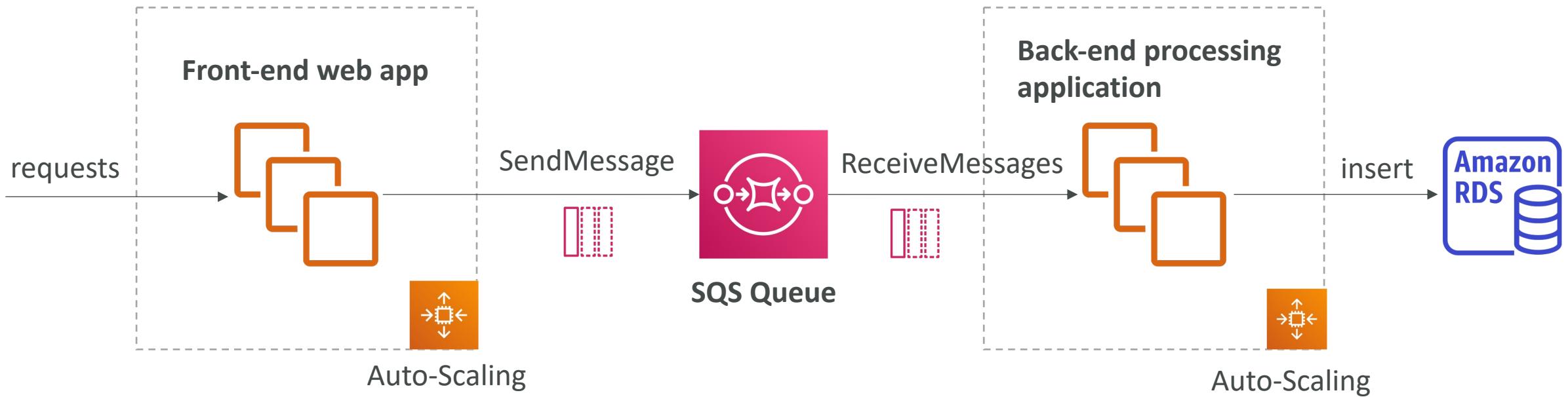


- Consumers receive and process messages in parallel
- At least once delivery
- Best-effort message ordering
- Consumers delete messages after processing them
- We can scale consumers horizontally to improve throughput of processing

# SQS with Auto Scaling Group (ASG)



# SQS to decouple between application tiers

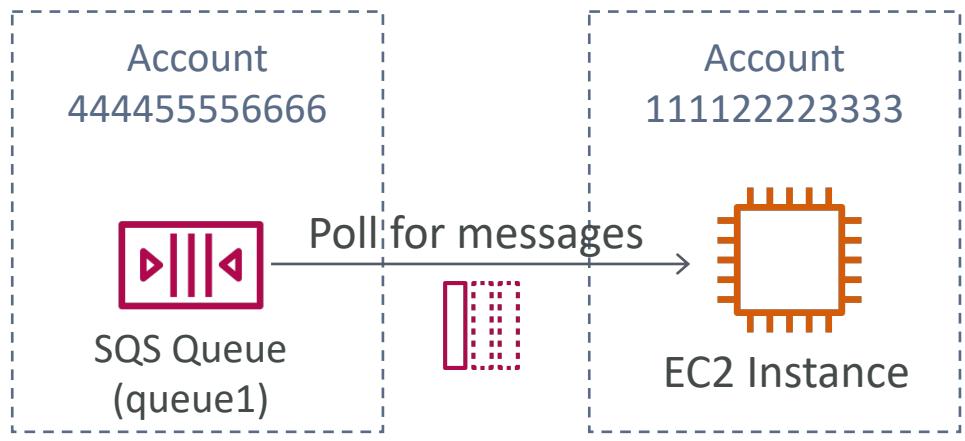


# Amazon SQS - Security

- **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 Queue Access Policy

## Cross Account Access



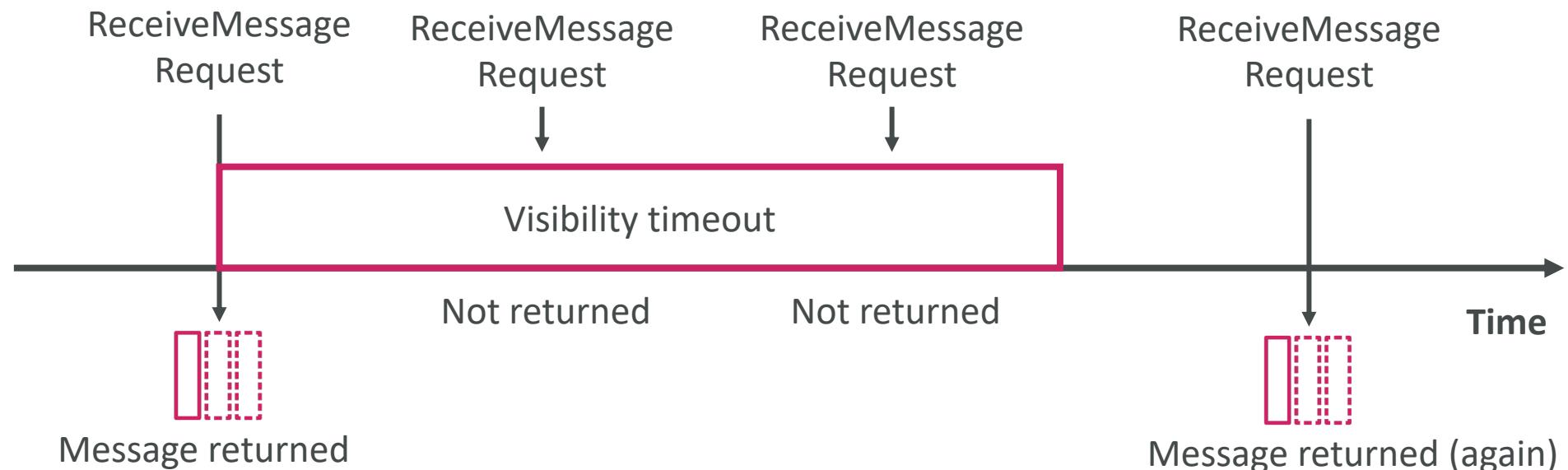
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": { "AWS": [ "111122223333" ] },  
      "Action": [ "sns:ReceiveMessage" ],  
      "Resource": "arn:aws:sns:us-east-1:444455556666:queue1"  
    }  
  ]}
```

## Publish S3 Event Notifications To SQS Queue

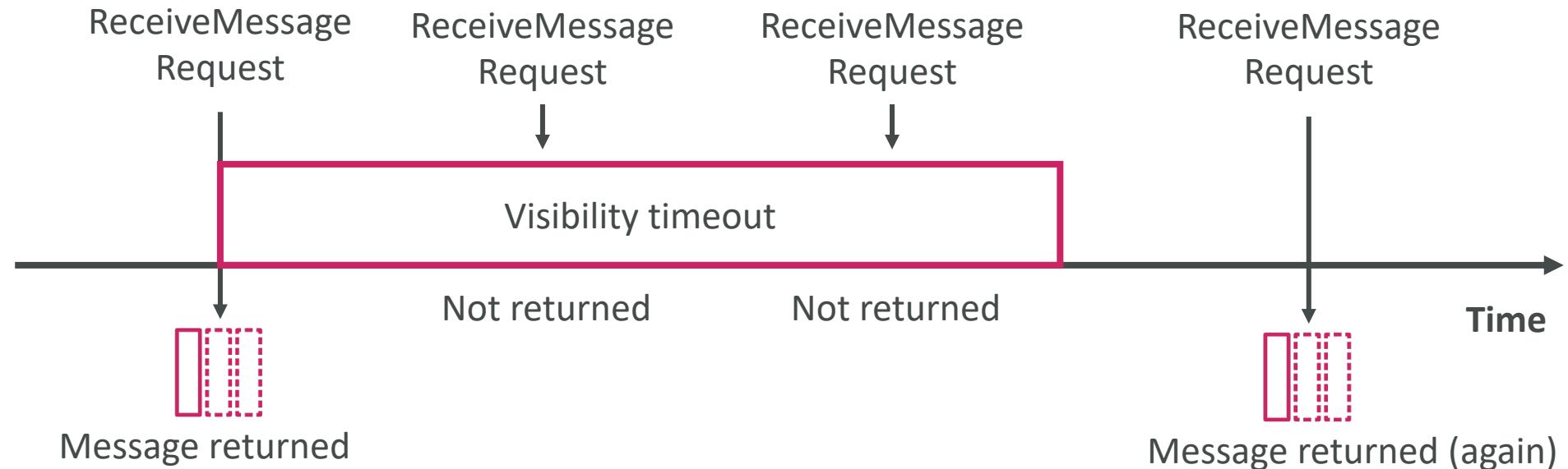


# SQS – Message Visibility Timeout

- 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



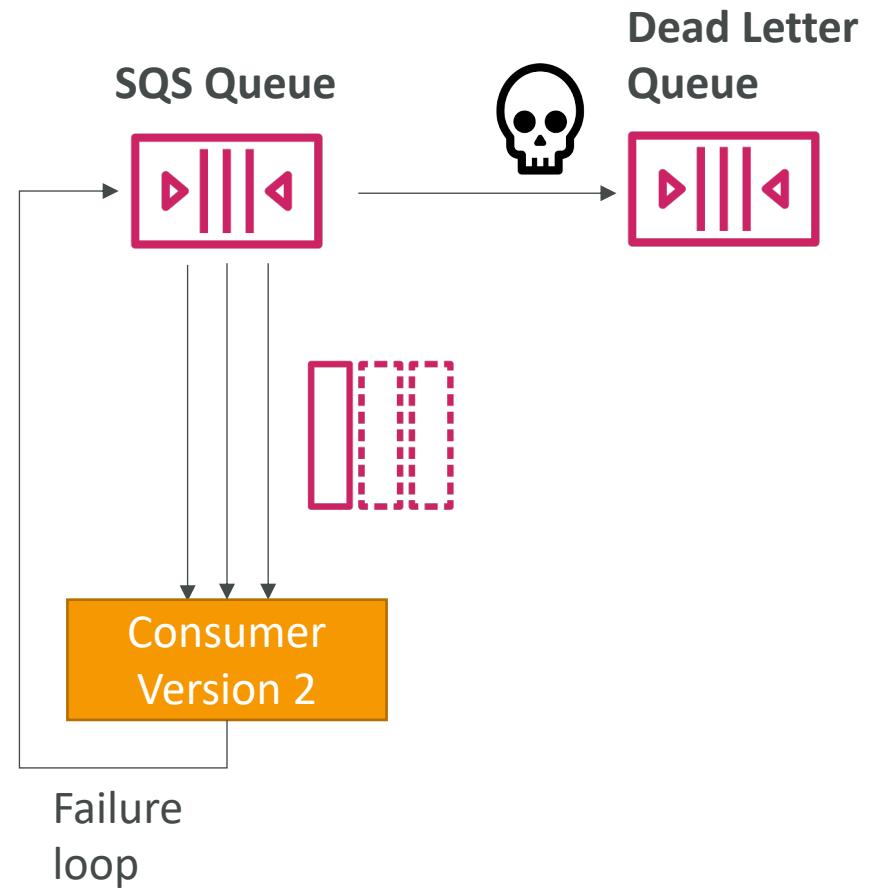
# SQS – Message Visibility Timeout



- 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

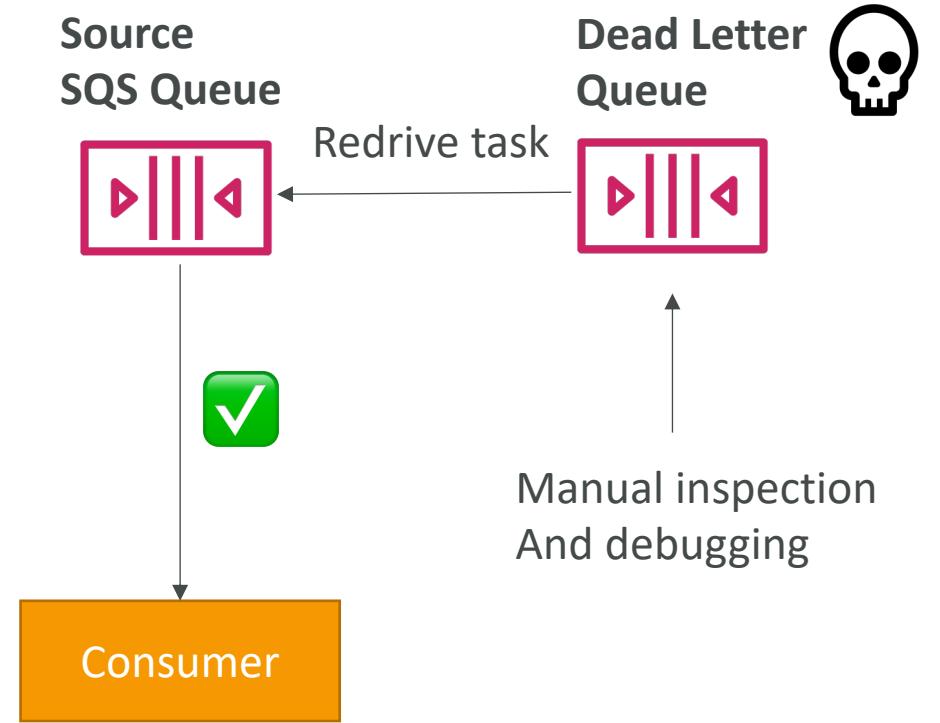
# Amazon SQS – Dead Letter Queue

- If a consumer fails to process a message within the Visibility Timeout...  
the message goes back to the queue!
- We can set a threshold of how many times a message can go back to the queue
- After the **MaximumReceives** threshold is exceeded, the message goes into a dead letter queue (DLQ)
  
- Useful for debugging!
- Make sure to process the messages in the DLQ before they expire:
  - Good to set a retention of 14 days in the DLQ



# SQS DLQ – Redrive to Source

- Feature to help consume messages in the DLQ to understand what is wrong with them
- When our code is fixed, we can redrive the messages from the DLQ back into the source queue (or any other queue) in batches without writing custom code



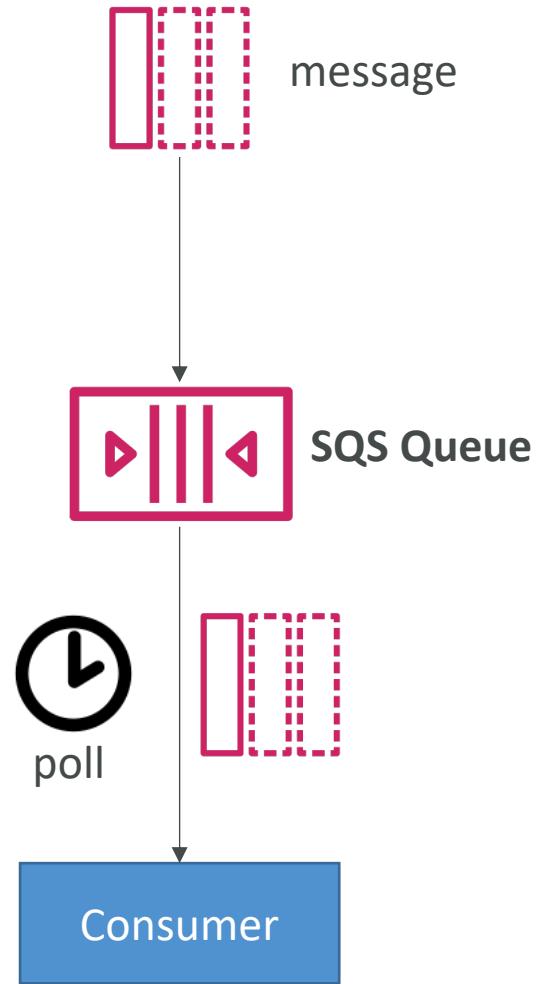
# Amazon SQS – Delay Queue

- Delay a message (consumers don't see it immediately) up to 15 minutes
- Default is 0 seconds (message is available right away)
- Can set a default at queue level
- Can override the default on send using the `DelaySeconds` parameter

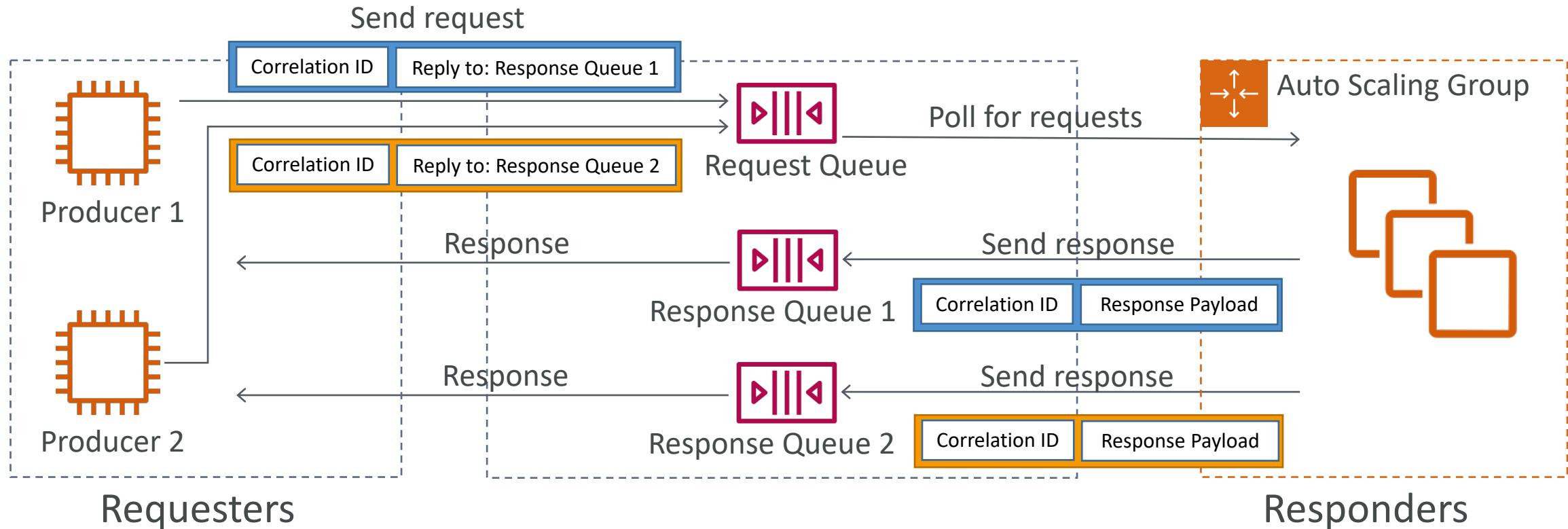


# Amazon SQS - Long Polling

- 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`



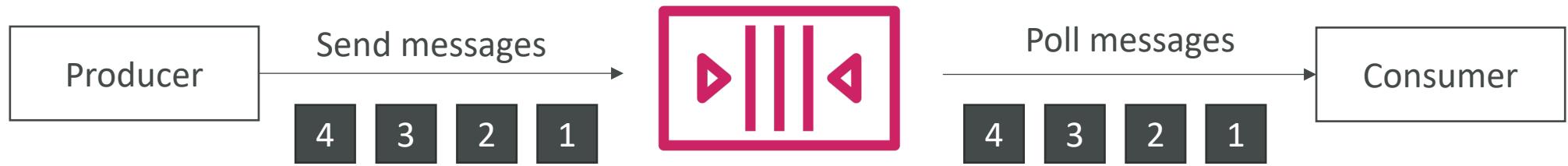
# SQS – Request-Response Systems



- To implement this pattern: use the [SQS Temporary Queue Client](#)
- It leverages virtual queues instead of creating / deleting SQS queues (cost-effective)

# Amazon SQS – FIFO Queue

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

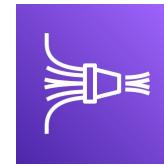


- Limited throughput: 300 msg/s without batching, 3000 msg/s with
- Exactly-once send capability (by removing duplicates)
- Messages are processed in order by the consumer

# Kinesis Overview

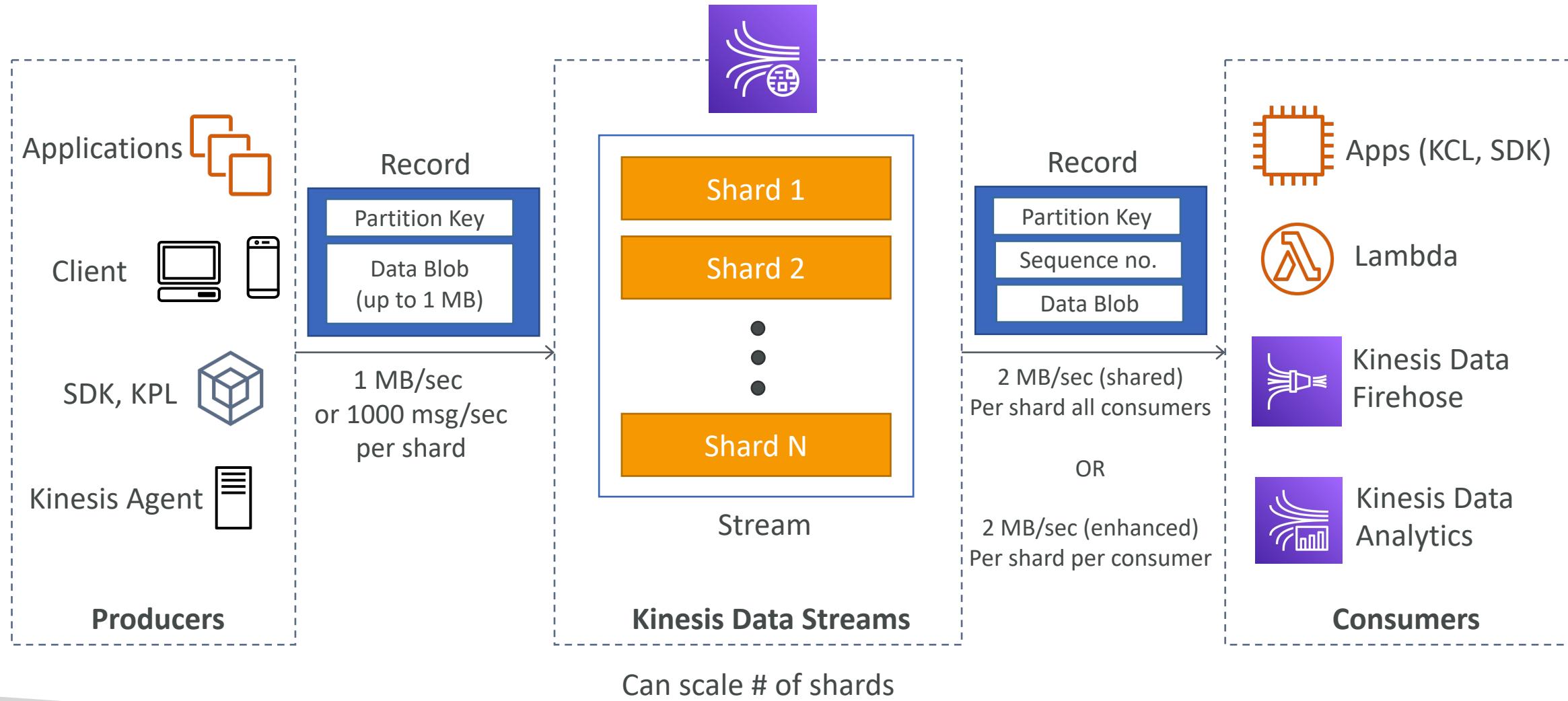


- Makes it easy to **collect, process, and analyze** streaming data in real-time
- Ingest real-time data such as: Application logs, Metrics, Website clickstreams, IoT telemetry data...



- **Kinesis Data Streams:** capture, process, and store data streams
- **Kinesis Data Firehose:** load data streams into AWS data stores
- **Kinesis Data Analytics:** analyze data streams with SQL or Apache Flink
- **Kinesis Video Streams:** capture, process, and store video streams

# Kinesis Data Streams





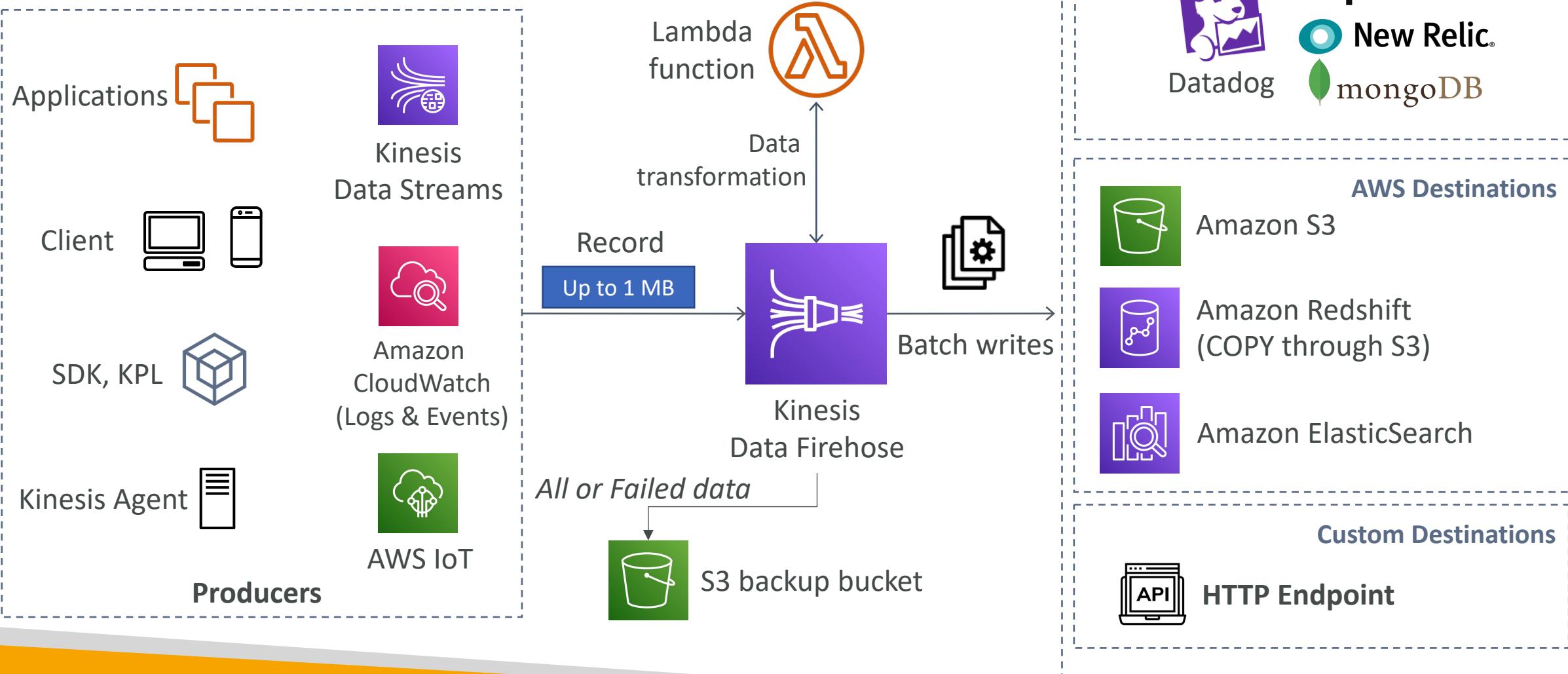
# Kinesis Data Streams

- Retention between 1 day to 365 days
- Ability to reprocess (replay) data
- Once data is inserted in Kinesis, it can't be deleted (immutability)
- Data that shares the same partition goes to the same shard (ordering)
- Producers: AWS SDK, Kinesis Producer Library (KPL), Kinesis Agent
- Consumers:
  - Write your own: Kinesis Client Library (KCL), AWS SDK
  - Managed: AWS Lambda, Kinesis Data Firehose, Kinesis Data Analytics,

# Kinesis Data Streams – Capacity Modes

- **Provisioned mode:**
  - You choose the number of shards provisioned, scale manually or using API
  - Each shard gets 1MB/s in (or 1000 records per second)
  - Each shard gets 2MB/s out (classic or enhanced fan-out consumer)
  - You pay per shard provisioned per hour
- **On-demand mode:**
  - No need to provision or manage the capacity
  - Default capacity provisioned (4 MB/s in or 4000 records per second)
  - Scales automatically based on observed throughput peak during the last 30 days
  - Pay per stream per hour & data in/out per GB

# Kinesis Data Firehose





# Kinesis Data Firehose

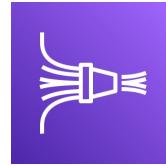
- Fully Managed Service, no administration, automatic scaling, serverless
  - AWS: Redshift / Amazon S3 / ElasticSearch
  - 3rd party partner: Splunk / MongoDB / DataDog / NewRelic / ...
  - Custom: send to any HTTP endpoint
- Pay for data going through Firehose
- **Near Real Time**
  - 60 seconds latency minimum for non full batches
  - Or minimum 32 MB of data at a time
- Supports many data formats, conversions, transformations, compression
- Supports custom data transformations using AWS Lambda
- Can send failed or all data to a backup S3 bucket

# Kinesis Data Streams vs Firehose



## Kinesis Data Streams

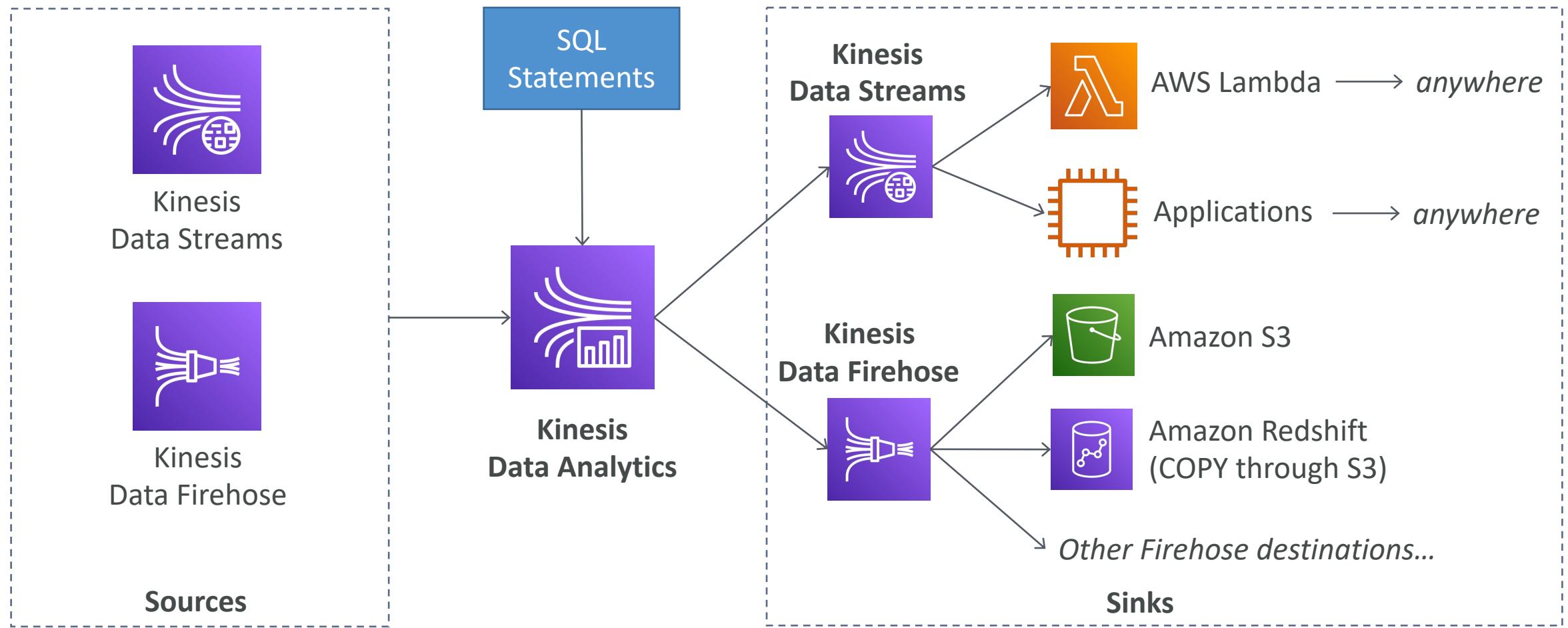
- Streaming service for ingest at scale
- Write custom code (producer / consumer)
- Real-time (~200 ms)
- Manage scaling (shard splitting / merging)
- Data storage for 1 to 365 days
- Supports replay capability



## Kinesis Data Firehose

- Load streaming data into S3 / Redshift / ES / 3<sup>rd</sup> party / custom HTTP
- Fully managed
- Near real-time (buffer time min. 60 sec)
- Automatic scaling
- No data storage
- Doesn't support replay capability

# Kinesis Data Analytics (SQL application)



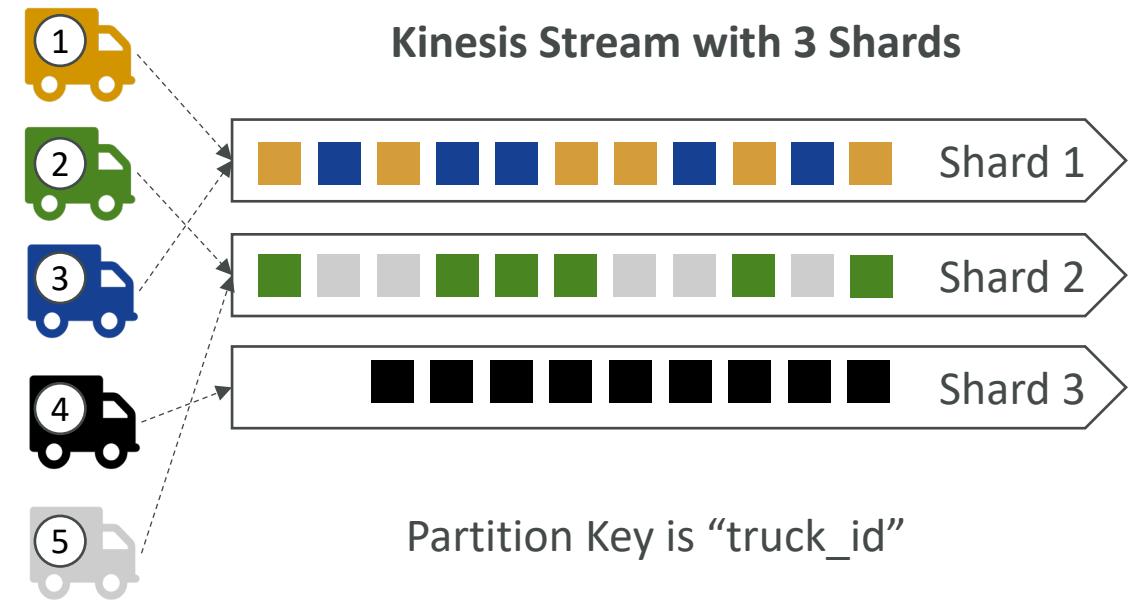


# Kinesis Data Analytics (SQL application)

- Perform real-time analytics on Kinesis Streams using SQL
- Fully managed, no servers to provision
- Automatic scaling
- Real-time analytics
- Pay for actual consumption rate
- Can create streams out of the real-time queries
- Use cases:
  - Time-series analytics
  - Real-time dashboards
  - Real-time metrics

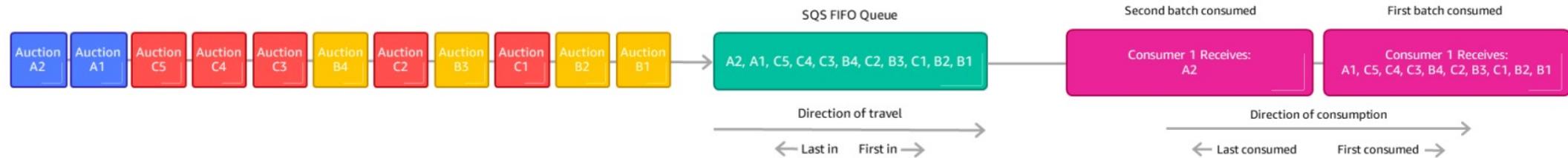
# Ordering data into Kinesis

- Imagine you have 100 trucks (truck\_1, truck\_2, ... truck\_100) on the road sending their GPS positions regularly into AWS.
- You want to consume the data in order for each truck, so that you can track their movement accurately.
- How should you send that data into Kinesis?
- Answer: send using a “Partition Key” value of the “truck\_id”
- The same key will always go to the same shard

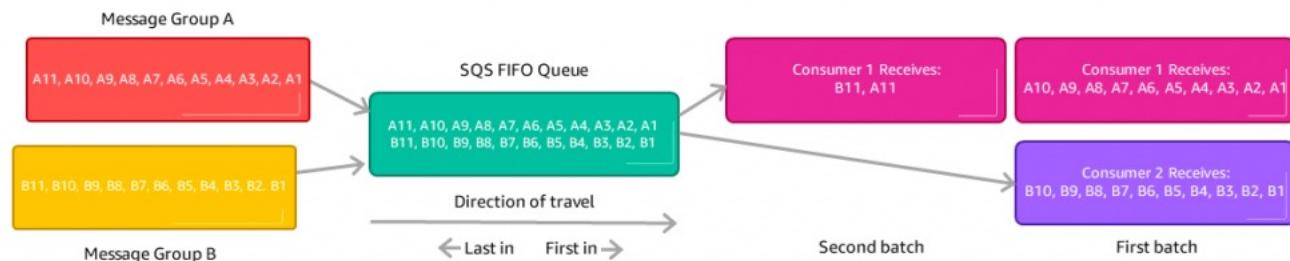


# Ordering data into SQS

- For SQS standard, there is no ordering.
- For SQS FIFO, if you don't use a Group ID, messages are consumed in the order they are sent, with **only one consumer**



- You want to scale the number of consumers, but you want messages to be “grouped” when they are related to each other
- Then you use a Group ID (similar to Partition Key in Kinesis)

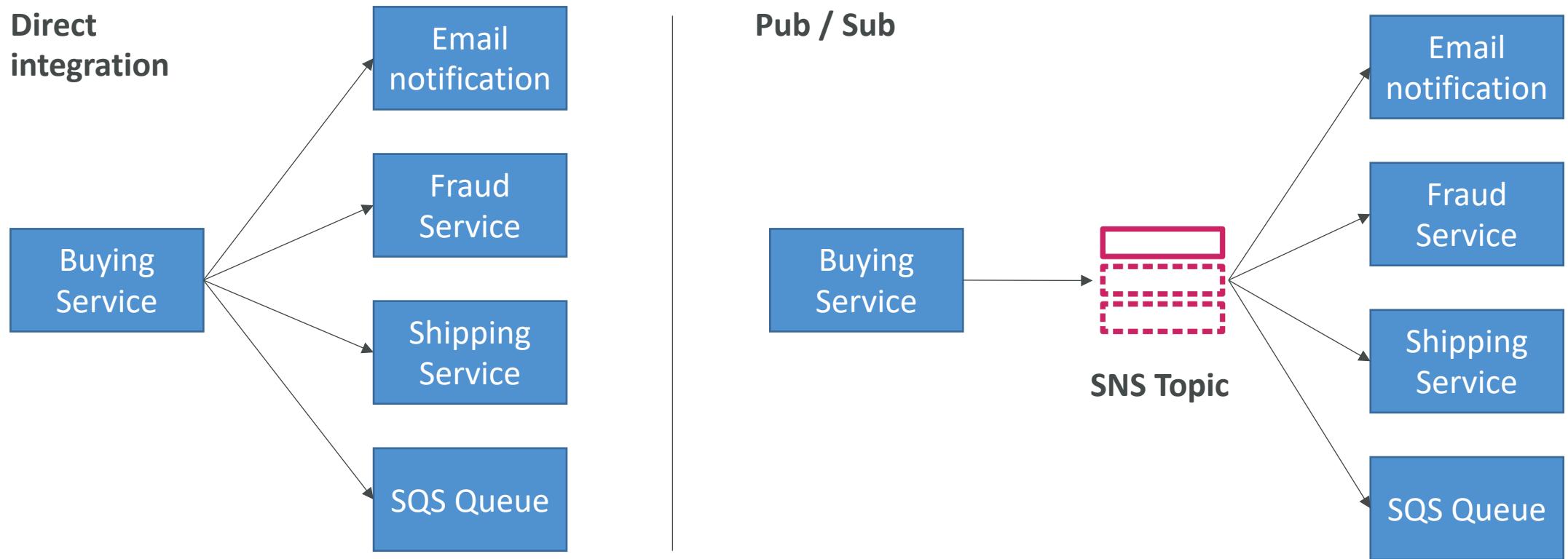


# Kinesis vs SQS ordering

- Let's assume 100 trucks, 5 kinesis shards, 1 SQS FIFO
- Kinesis Data Streams:
  - On average you'll have 20 trucks per shard
  - Trucks will have their data ordered within each shard
  - The maximum amount of consumers in parallel we can have is 5
  - Can receive up to 5 MB/s of data
- SQS FIFO
  - You only have one SQS FIFO queue
  - You will have 100 Group ID
  - You can have up to 100 Consumers (due to the 100 Group ID)
  - You have up to 300 messages per second (or 3000 if using batching)

# Amazon SNS

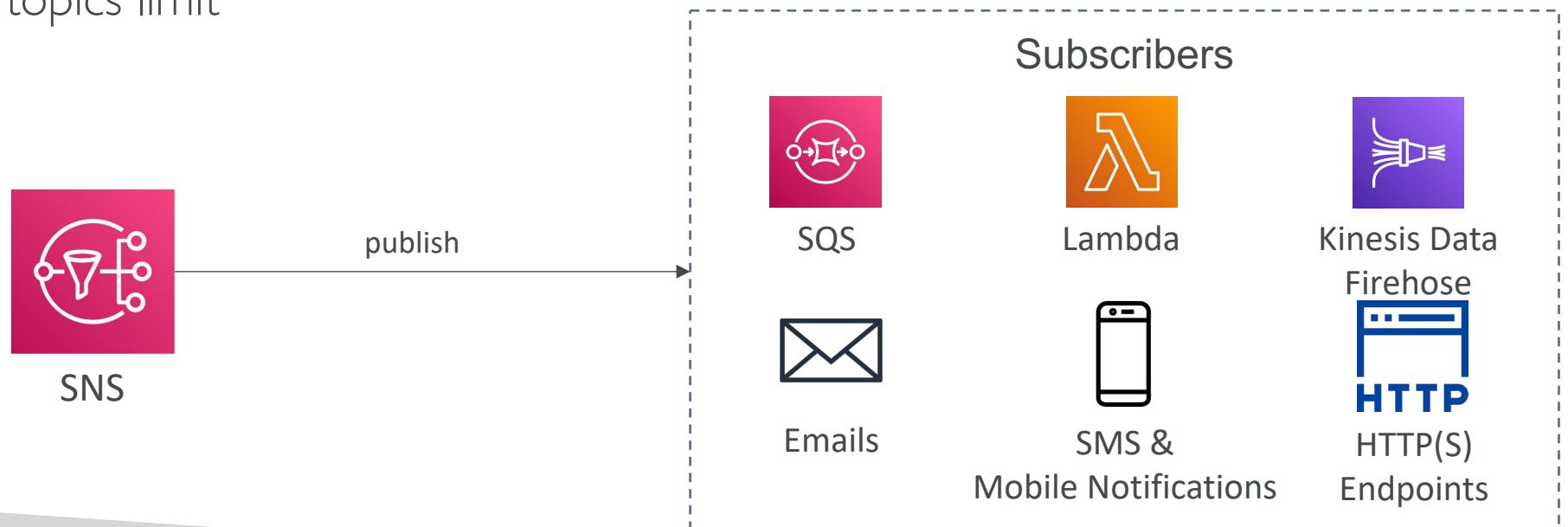
- What if you want to send one message to many receivers?



# Amazon SNS

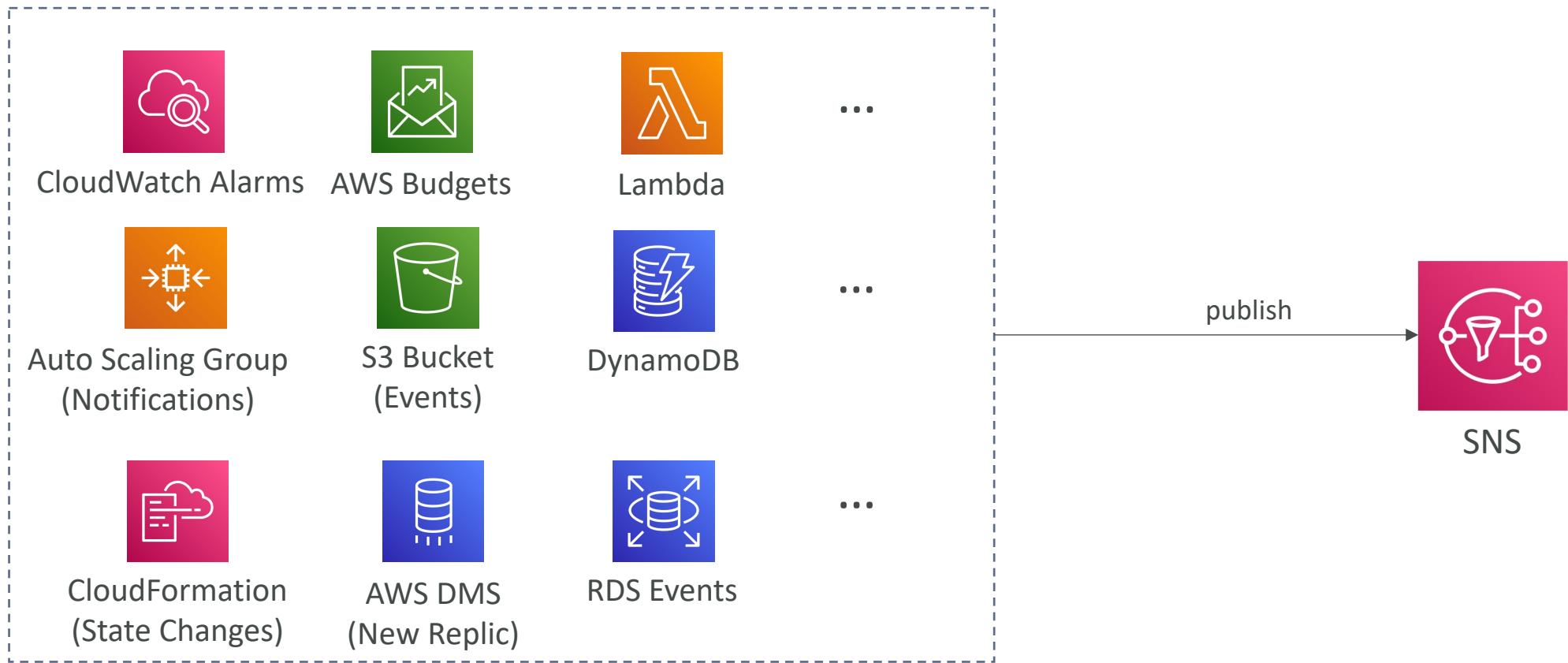


- The “event producer” only sends message to one SNS topic
- As many “event receivers” (subscriptions) as we want to listen to the SNS topic notifications
- Each subscriber to the topic will get all the messages (note: new feature to filter messages)
- Up to 12,500,000 subscriptions per topic
- 100,000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



# SNS integrates with a lot of AWS services

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



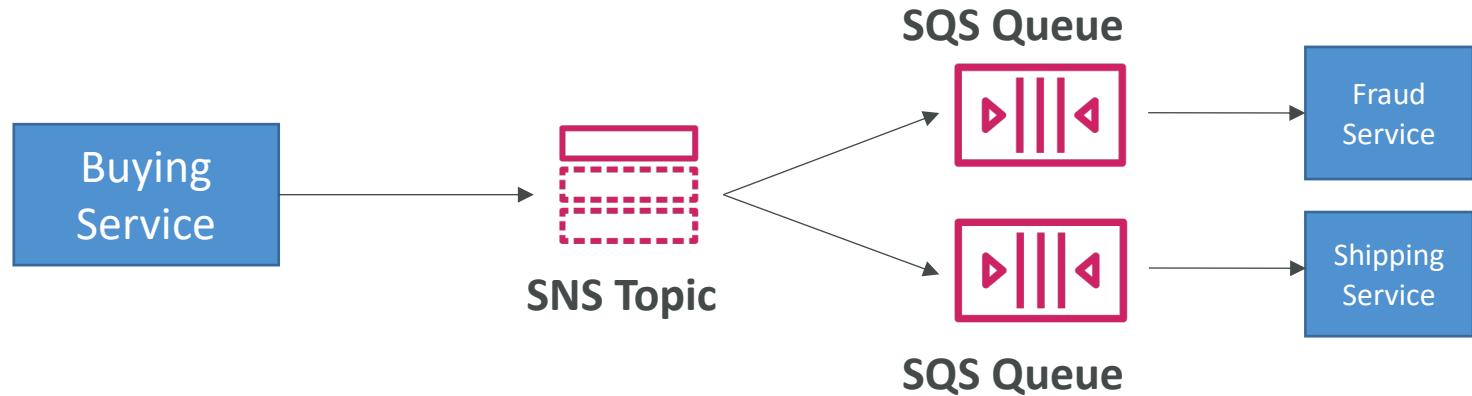
# Amazon SNS – How to publish

- 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

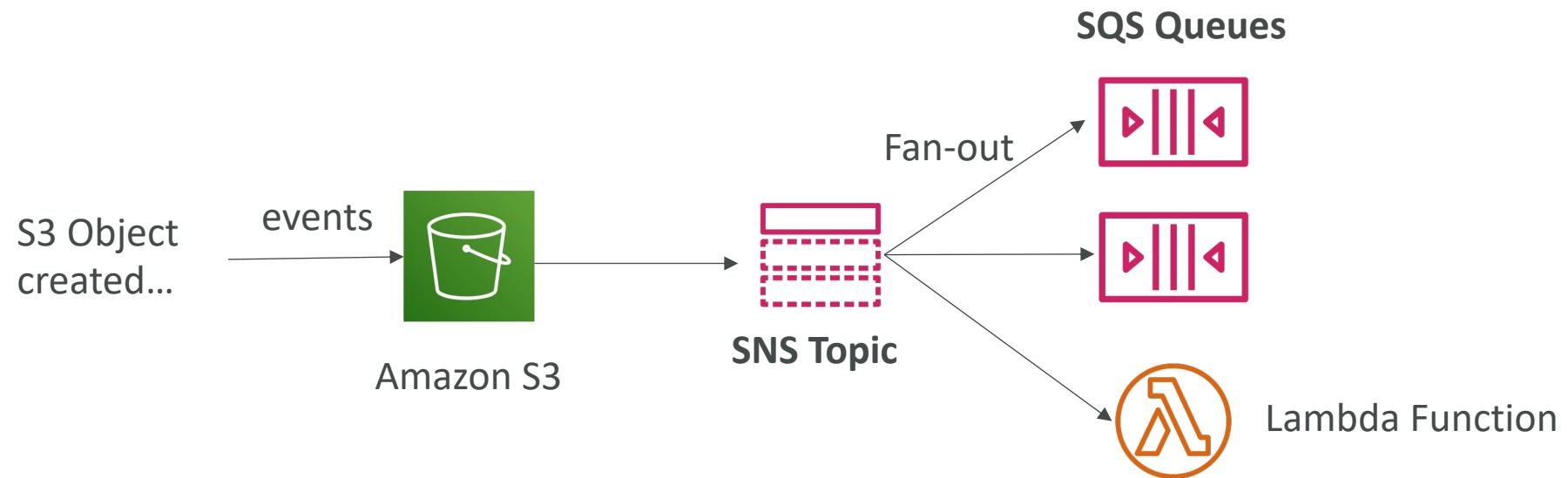
# SNS + SQS: Fan Out



- 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

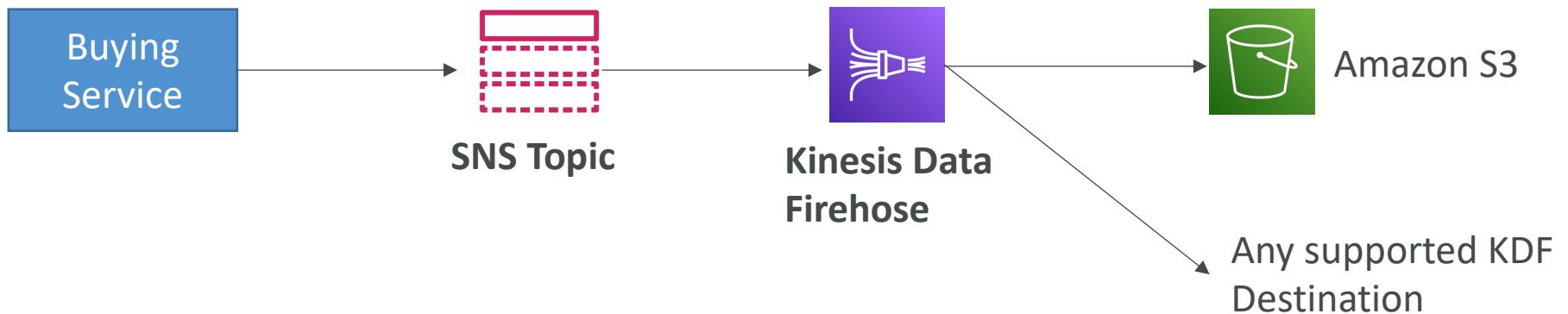
# Application: S3 Events to multiple queues

- 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:



# Amazon SNS – FIFO Topic

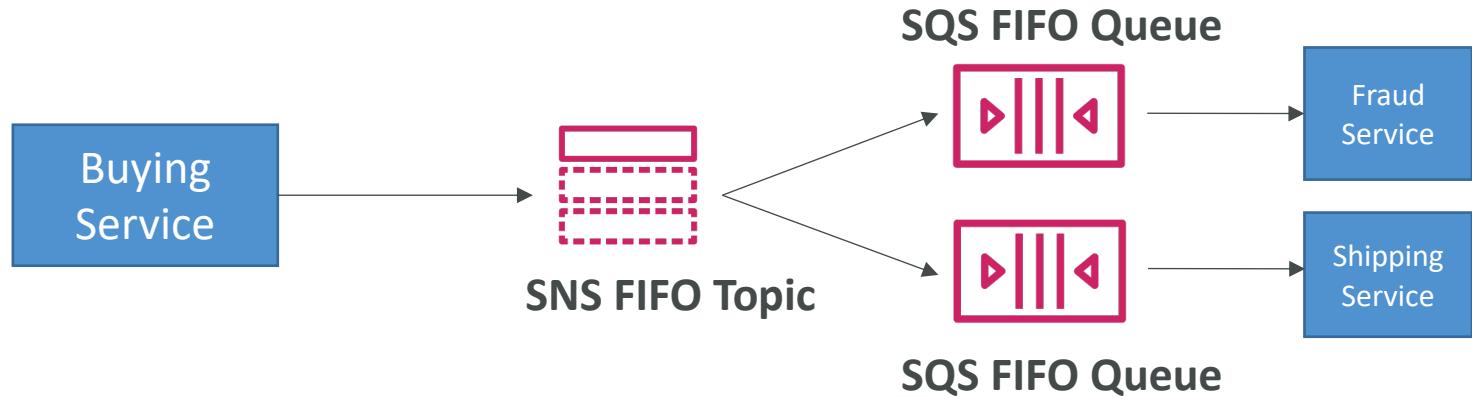
- FIFO = First In First Out (ordering of messages in the topic)



- Similar features as SQS FIFO:
  - Ordering by Message Group ID (all messages in the same group are ordered)
  - Deduplication using a Deduplication ID or Content Based Deduplication
- Can only have SQS FIFO queues as subscribers
- Limited throughput (same throughput as SQS FIFO)

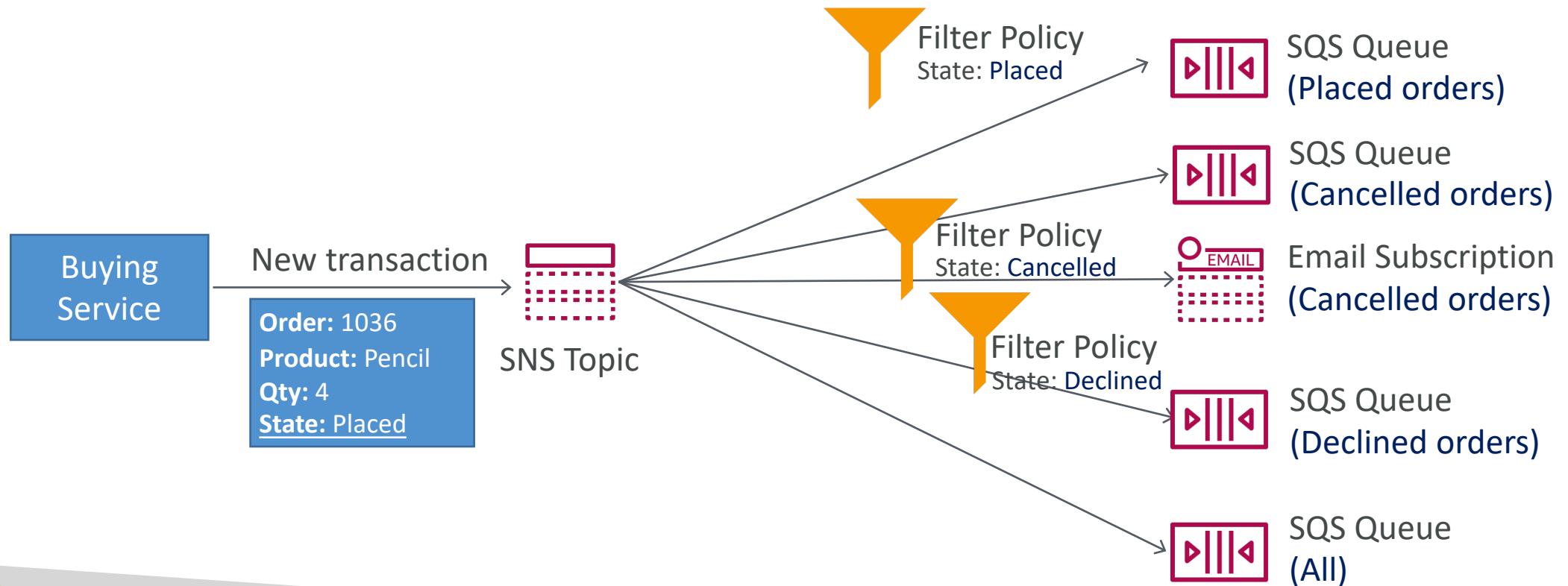
# SNS FIFO + SQS FIFO: Fan Out

- In case you need fan out + ordering + deduplication



# SNS – Message Filtering

- JSON policy used to filter messages sent to SNS topic's subscriptions
- If a subscription doesn't have a filter policy, it receives every message



# SQS vs SNS vs Kinesis

## SQS:

- Consumer “pull data”
- Data is deleted after being consumed
- Can have as many workers (consumers) as we want
- No need to provision throughput
- Ordering guarantees only on FIFO queues
- Individual message delay capability



## SNS:

- Push data to many subscribers
- Up to 12,500,000 subscribers
- Data is not persisted (lost if not delivered)
- Pub/Sub
- Up to 100,000 topics
- No need to provision throughput
- Integrates with SQS for fan-out architecture pattern
- FIFO capability for SQS FIFO

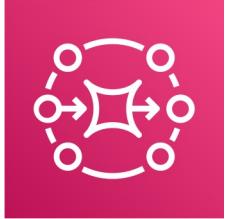


## Kinesis:

- Standard: pull data
  - 2 MB per shard
- Enhanced-fan out: push data
  - 2 MB per shard per consumer
- Possibility to replay data
- Meant for real-time big data, analytics and ETL
- Ordering at the shard level
- Data expires after X days
- Provisioned mode or on-demand capacity mode

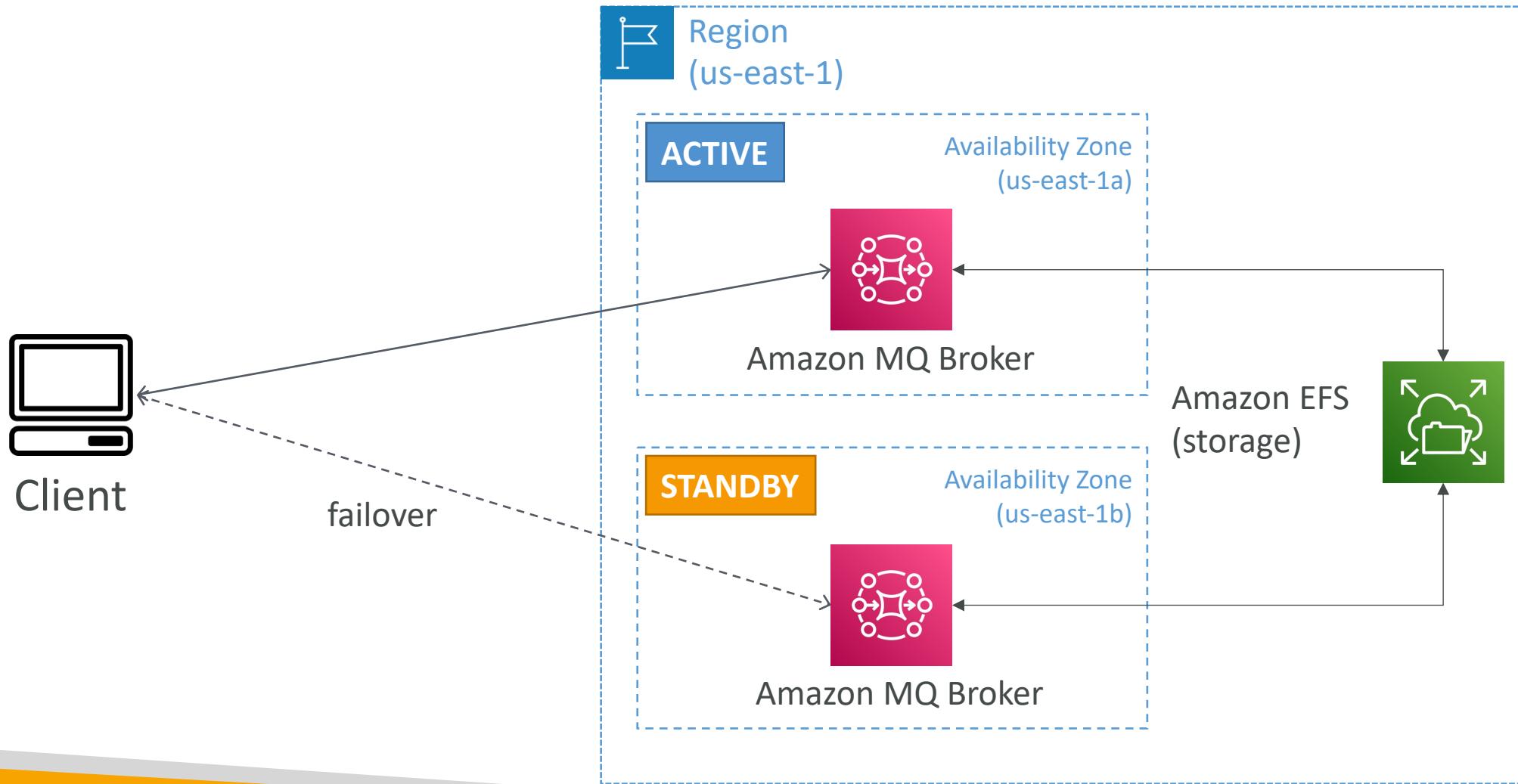


# Amazon MQ



- SQS, SNS are “cloud-native” services, and they’re using proprietary protocols from AWS.
  - Traditional applications running from on-premises may use open protocols such as: MQTT, AMQP, STOMP, Openwire, WSS
  - When migrating to the cloud, instead of re-engineering the application to use SQS and SNS, we can use Amazon MQ
  - Amazon MQ = managed Apache ActiveMQ
- 
- Amazon MQ doesn’t “scale” as much as SQS / SNS
  - Amazon MQ runs on a dedicated machine, can run in HA with failover
  - Amazon MQ has both queue feature (~SQS) and topic features (~SNS)

# Amazon MQ – High Availability



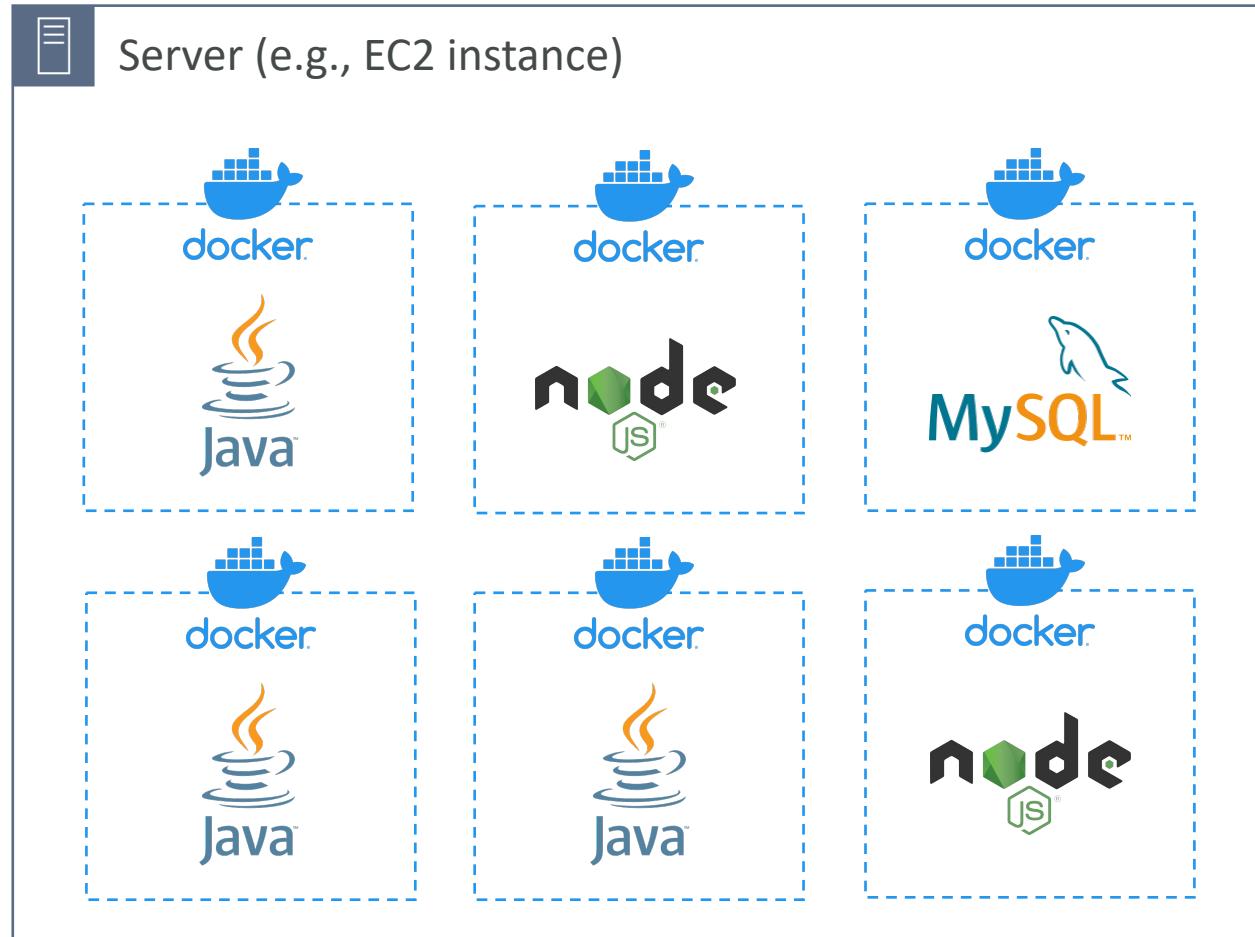
# Container Section



# What is Docker?

- Docker is a software development platform to deploy apps
- Apps are packaged in **containers** that can be run on any OS
- Apps run the same, regardless of where they're run
  - Any machine
  - No compatibility issues
  - Predictable behavior
  - Less work
  - Easier to maintain and deploy
  - Works with any language, any OS, any technology
- Use cases: microservices architecture, lift-and-shift apps from on-premises to the AWS cloud, ...

# Docker on an OS

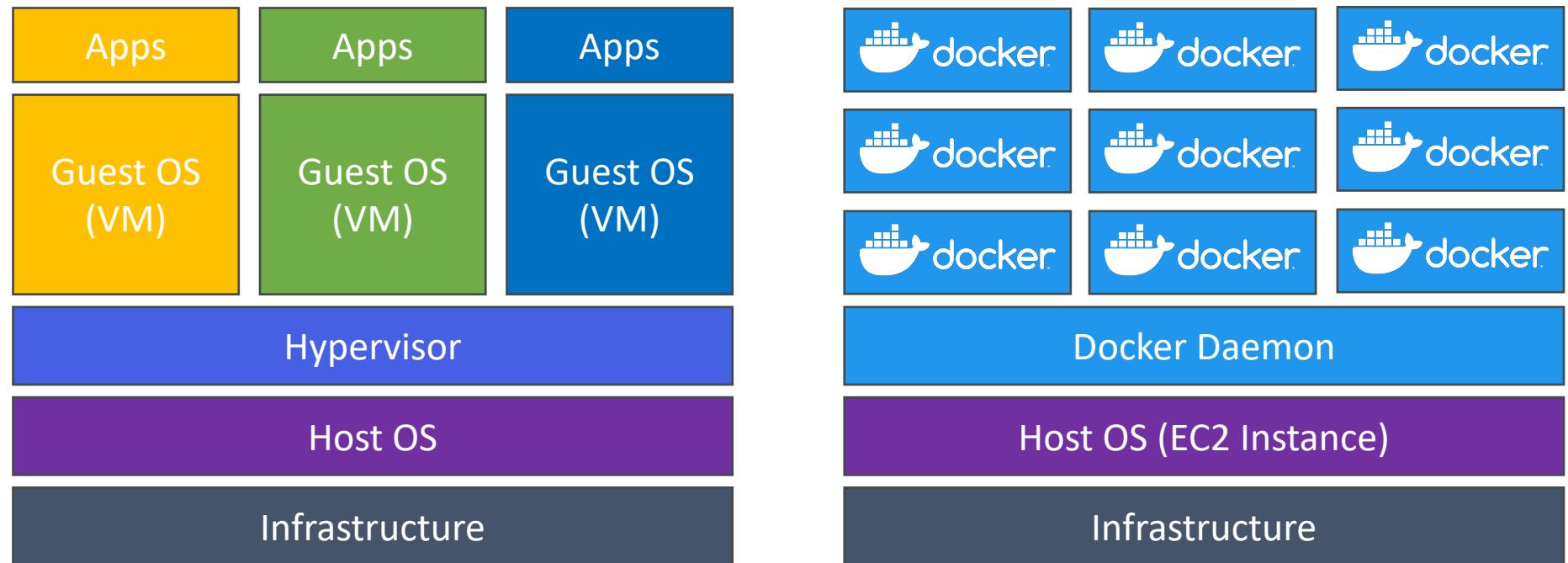


# Where are Docker images stored?

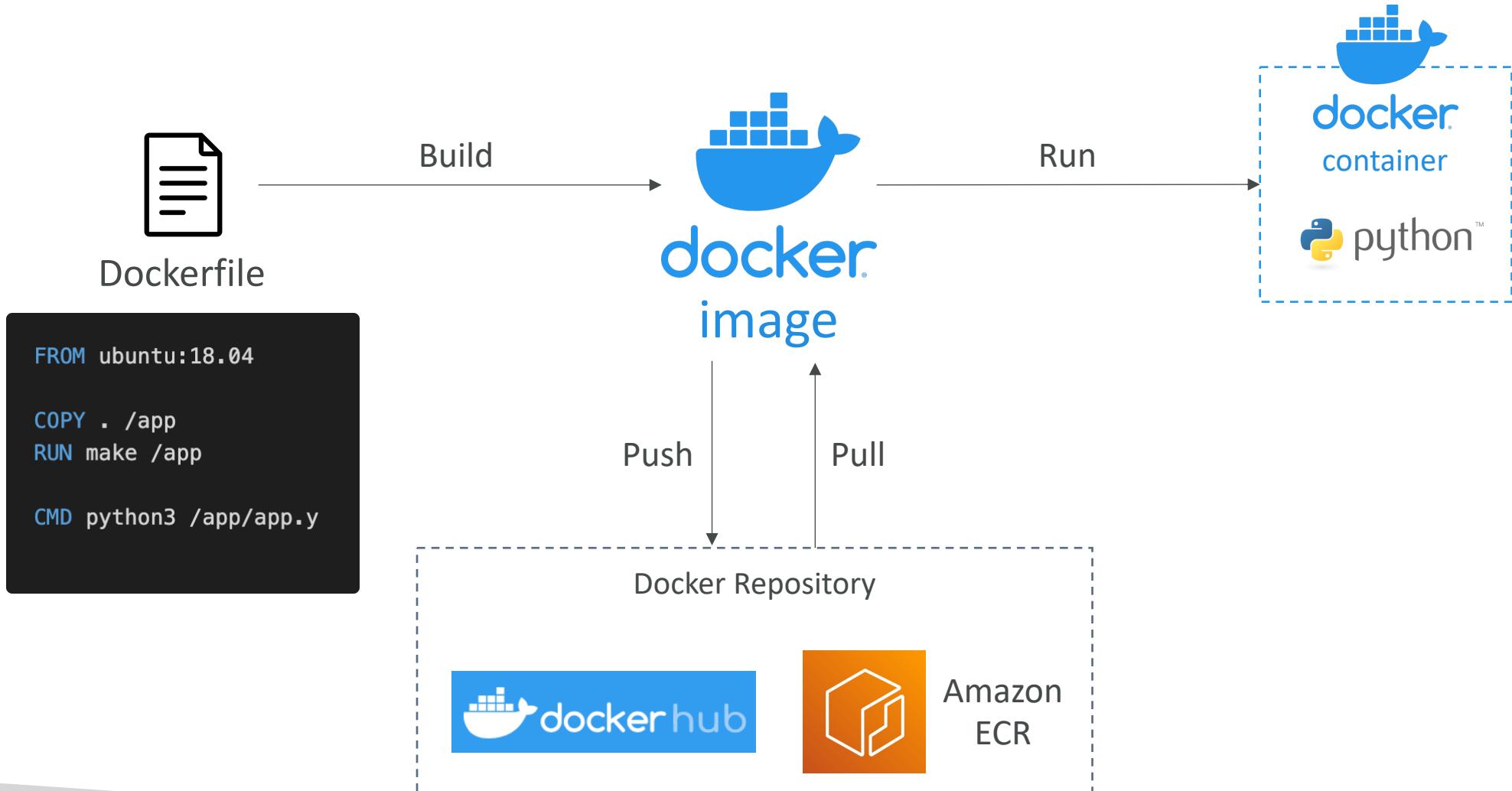
- Docker images are stored in Docker Repositories
- Docker Hub (<https://hub.docker.com>)
  - Public repository
  - Find base images for many technologies or OS (e.g., Ubuntu, MySQL, ...)
- Amazon ECR (Amazon Elastic Container Registry)
  - Private repository
  - Public repository (Amazon ECR Public Gallery <https://gallery.ecr.aws>)

# Docker vs. Virtual Machines

- Docker is "sort of" a virtualization technology, but not exactly
- Resources are shared with the host => many containers on one server



# Getting Started with Docker



# Docker Containers Management on AWS

- Amazon Elastic Container Service (Amazon ECS)
  - Amazon's own container platform
- Amazon Elastic Kubernetes Service (Amazon EKS)
  - Amazon's managed Kubernetes (open source)
- AWS Fargate
  - Amazon's own Serverless container platform
  - Works with ECS and with EKS
- Amazon ECR:
  - Store container images



Amazon ECS



Amazon EKS



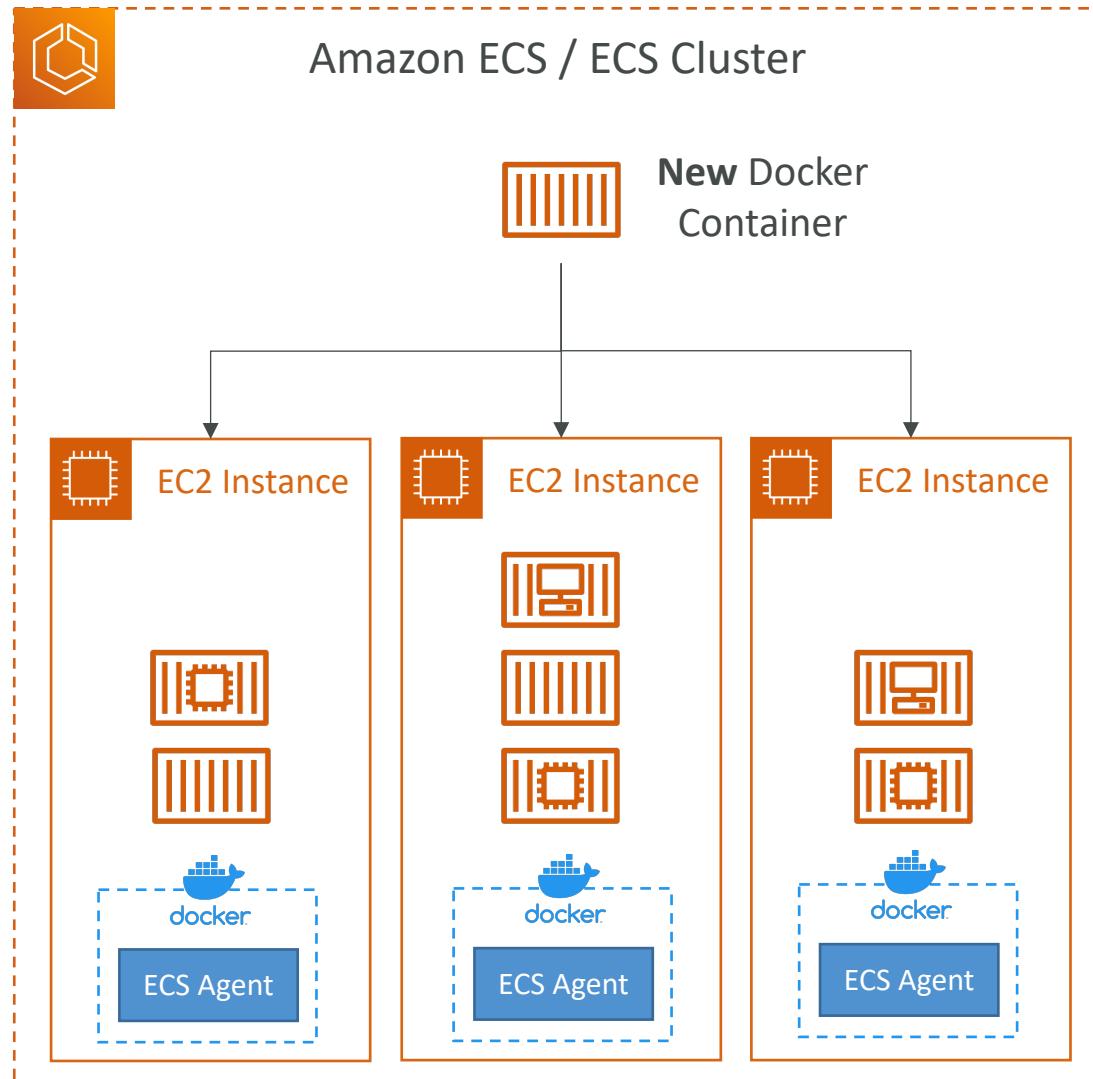
AWS Fargate



Amazon ECR

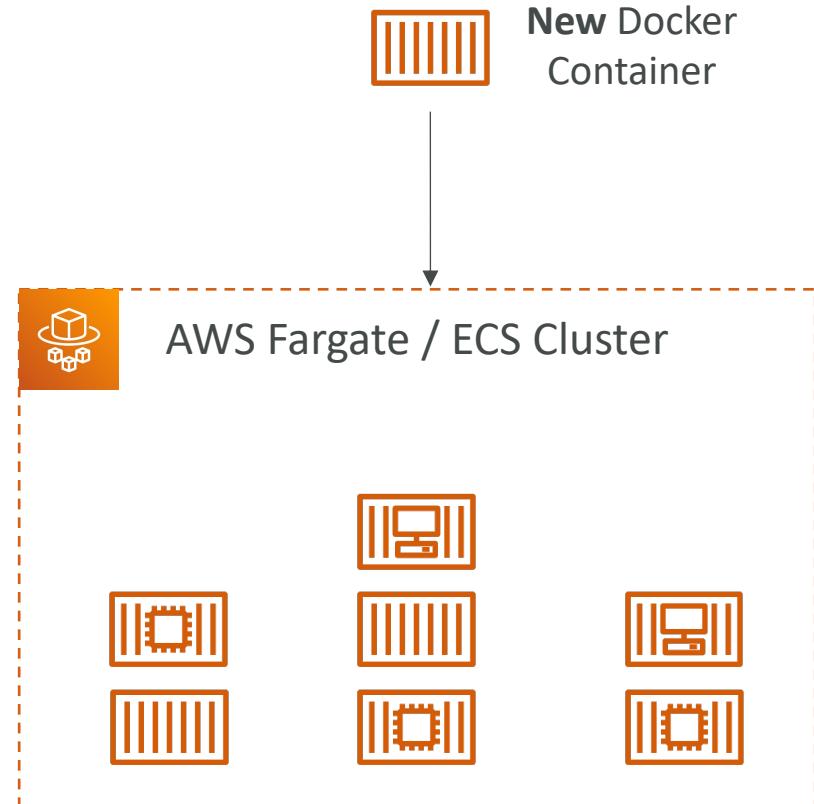
# Amazon ECS - EC2 Launch Type

- ECS = Elastic Container Service
- Launch Docker containers on AWS = Launch **ECS Tasks** on ECS Clusters
- **EC2 Launch Type:** you must provision & maintain the infrastructure (the EC2 instances)
- Each EC2 Instance must run the ECS Agent to register in the ECS Cluster
- AWS takes care of starting / stopping containers



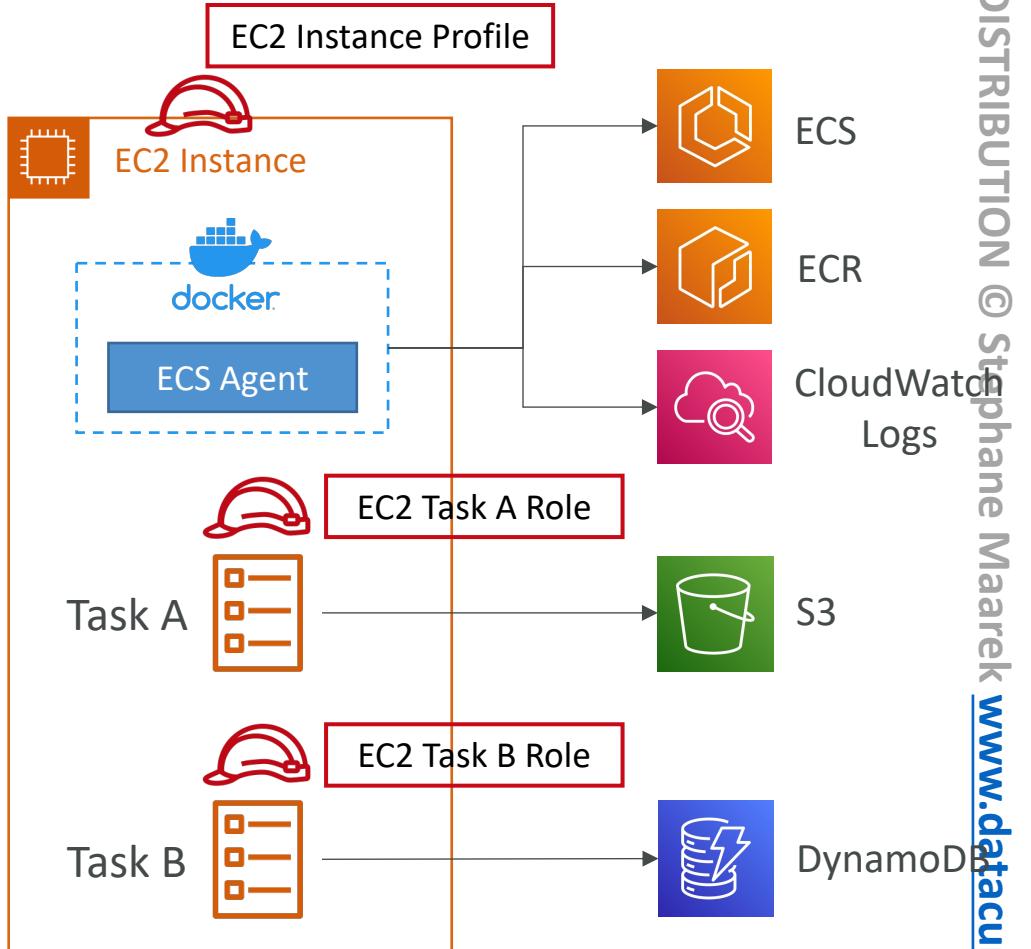
# Amazon ECS – Fargate Launch Type

- Launch Docker containers on AWS
- You do not provision the infrastructure  
(no EC2 instances to manage)
- It's all Serverless!
- You just create task definitions
- AWS just runs ECS Tasks for you based  
on the CPU / RAM you need
- To scale, just increase the number of  
tasks. Simple - no more EC2 instances



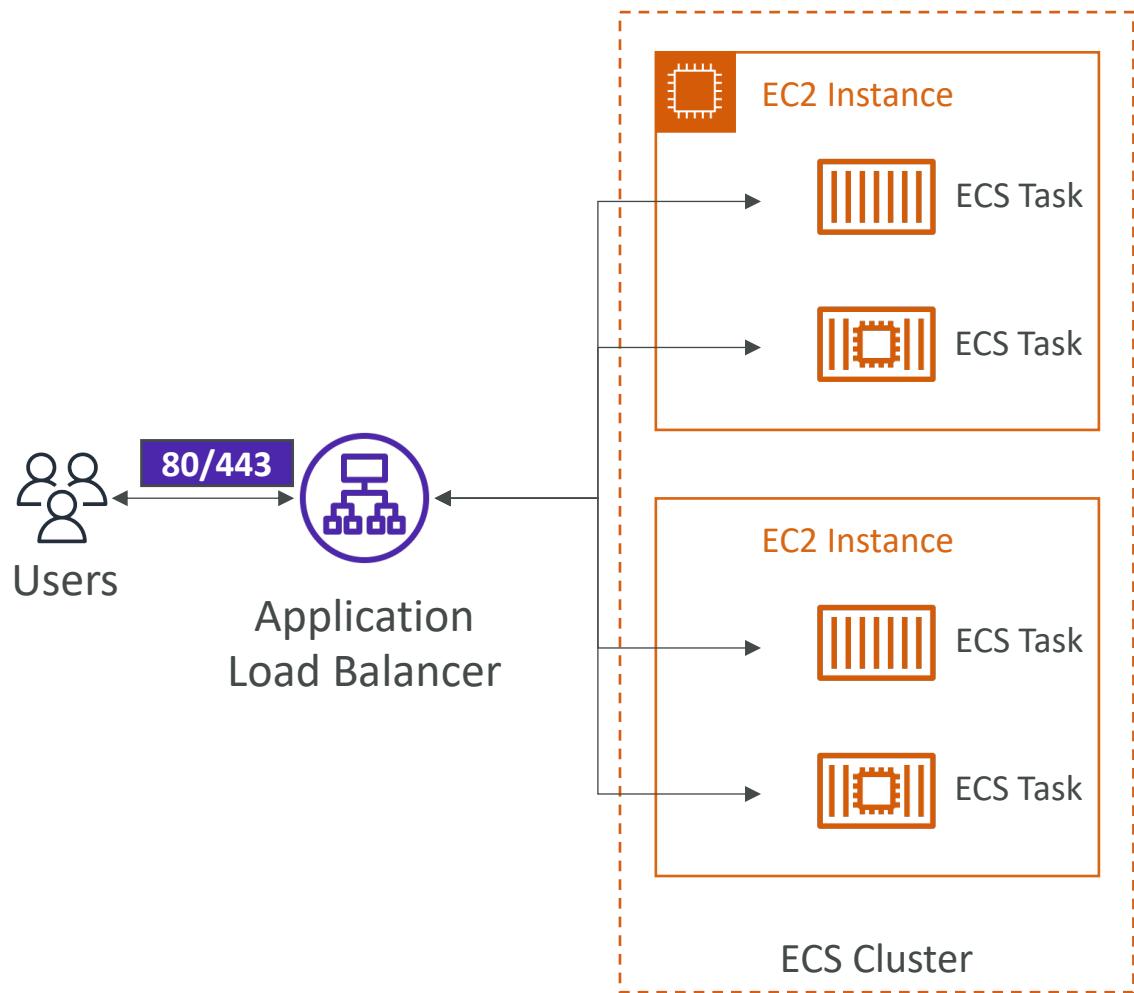
# Amazon ECS – IAM Roles for ECS

- **EC2 Instance Profile (EC2 Launch Type only):**
  - Used by the ECS agent
  - Makes API calls to ECS service
  - Send container logs to CloudWatch Logs
  - Pull Docker image from ECR
  - Reference sensitive data in Secrets Manager or SSM Parameter Store
- **ECS Task Role:**
  - Allows each task to have a specific role
  - Use different roles for the different ECS Services you run
  - Task Role is defined in the task definition



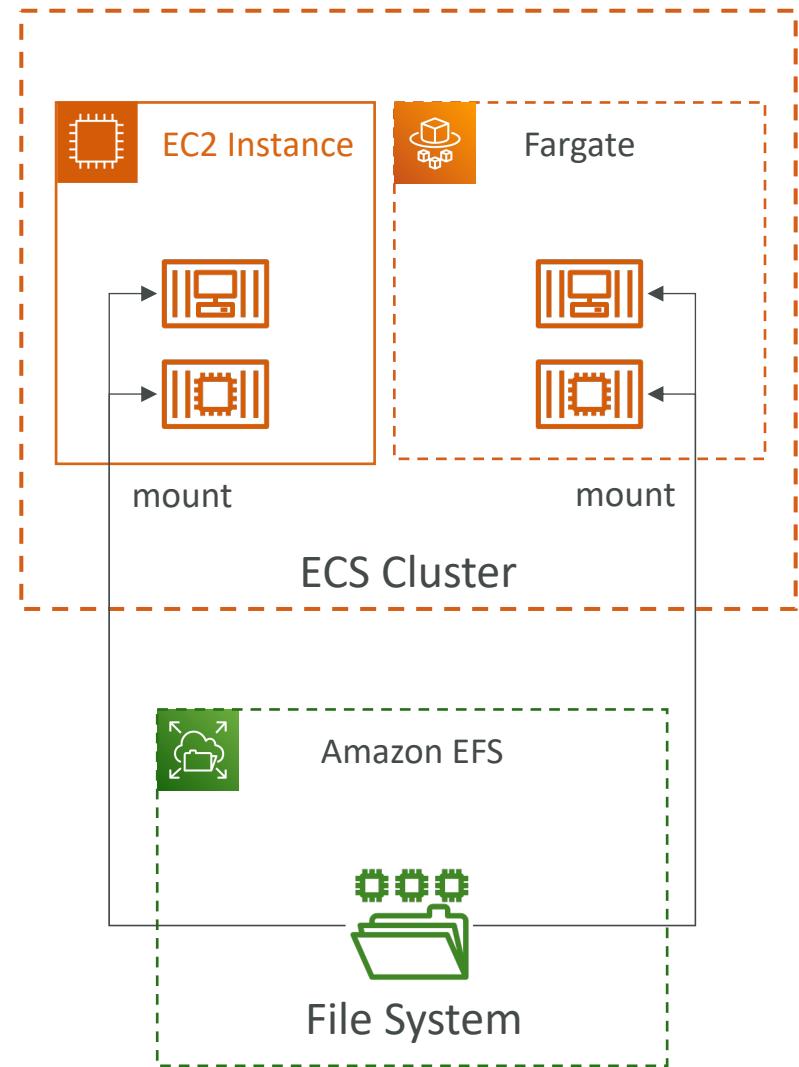
# Amazon ECS – Load Balancer Integrations

- **Application Load Balancer** supported and works for most use cases
- **Network Load Balancer** recommended only for high throughput / high performance use cases, or to pair it with AWS Private Link
- **Elastic Load Balancer** supported but not recommended (no advanced features – no Fargate)

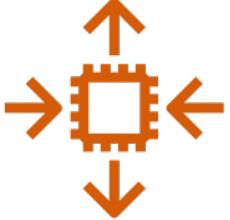


# Amazon ECS – Data Volumes (EFS)

- Mount EFS file systems onto ECS tasks
- Works for both **EC2** and **Fargate** launch types
- Tasks running in any AZ will share the same data in the EFS file system
- **Fargate + EFS = Serverless**
- Use cases: persistent multi-AZ shared storage for your containers
- Note:
  - FSx For Lustre & Windows not supported
  - Amazon S3 cannot be mounted as a file system



# ECS Service Auto Scaling

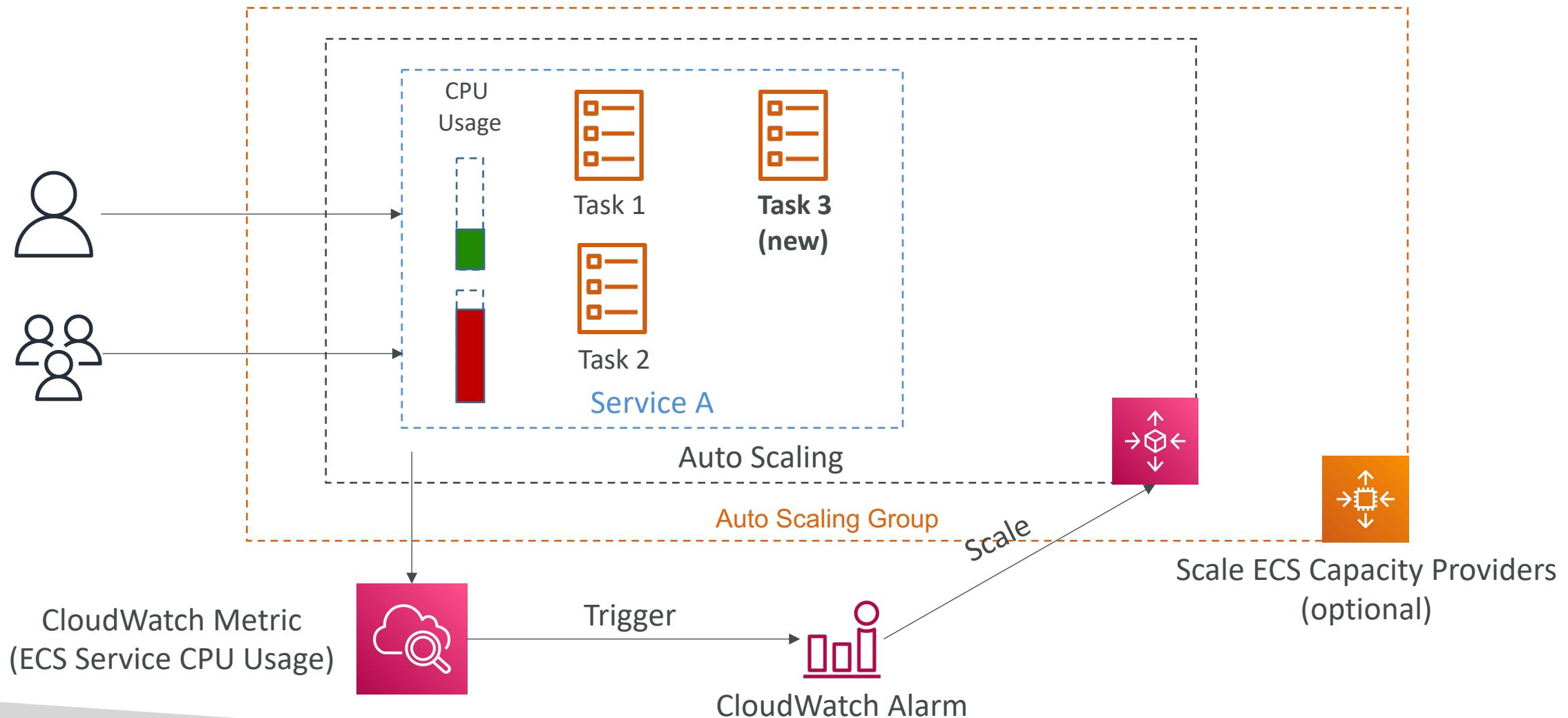


- Automatically increase/decrease the desired number of ECS tasks
- Amazon ECS Auto Scaling uses **AWS Application Auto Scaling**
  - ECS Service Average CPU Utilization
  - ECS Service Average Memory Utilization - Scale on RAM
  - ALB Request Count Per Target – metric coming from the ALB
- **Target Tracking** – scale based on target value for a specific CloudWatch metric
- **Step Scaling** – scale based on a specified CloudWatch Alarm
- **Scheduled Scaling** – scale based on a specified date/time (predictable changes)
- ECS Service Auto Scaling (task level) **≠** EC2 Auto Scaling (EC2 instance level)
- Fargate Auto Scaling is much easier to setup (because **Serverless**)

# EC2 Launch Type – Auto Scaling EC2 Instances

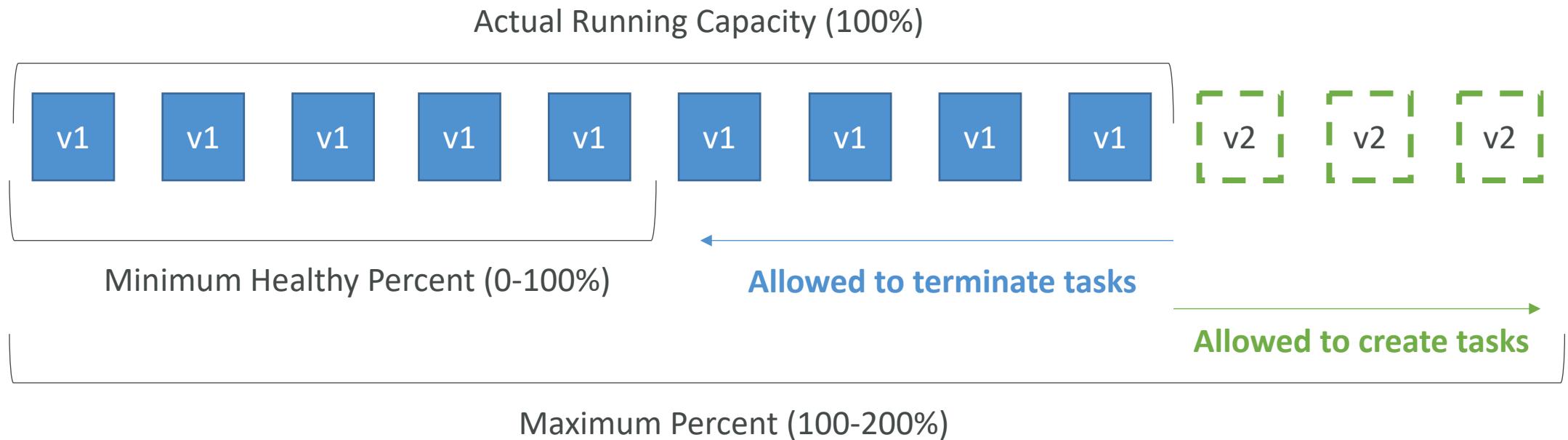
- Accommodate ECS Service Scaling by adding underlying EC2 Instances
- **Auto Scaling Group Scaling**
  - Scale your ASG based on CPU Utilization
  - Add EC2 instances over time
- **ECS Cluster Capacity Provider**
  - Used to automatically provision and scale the infrastructure for your ECS Tasks
  - Capacity Provider paired with an Auto Scaling Group
  - Add EC2 Instances when you're missing capacity (CPU, RAM...)

# ECS Scaling – Service CPU Usage Example



# ECS Rolling Updates

- When updating from v1 to v2, we can control how many tasks can be started and stopped, and in which order

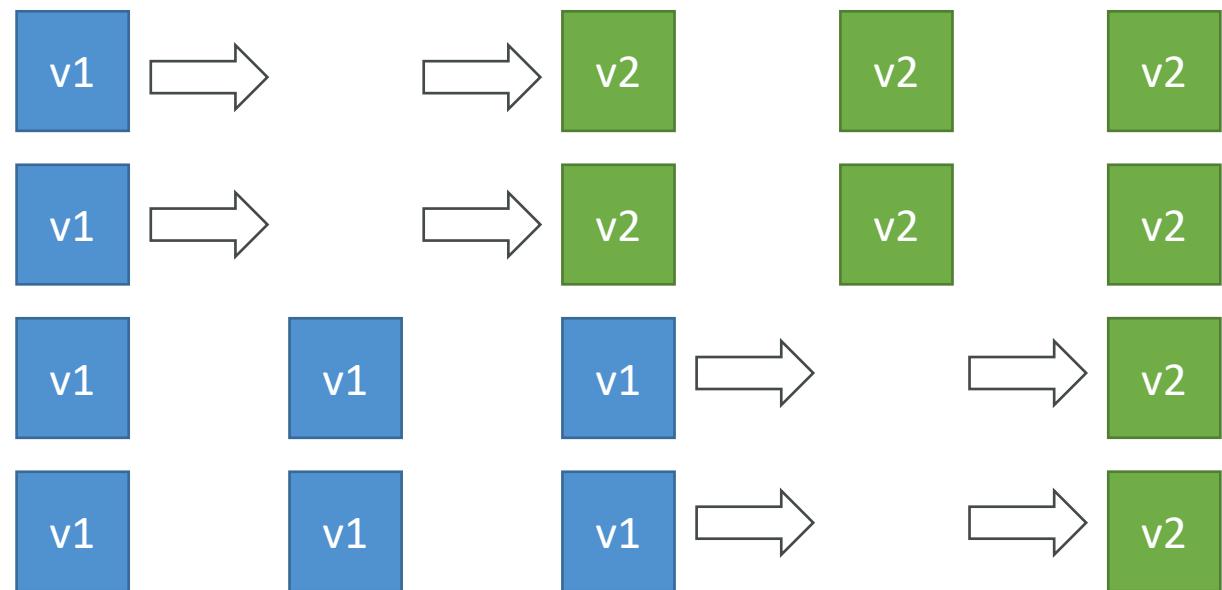


ECS Service update screen

Minimum healthy percent	<input type="text" value="100"/> <small>i</small>
Maximum percent	<input type="text" value="200"/> <small>i</small>

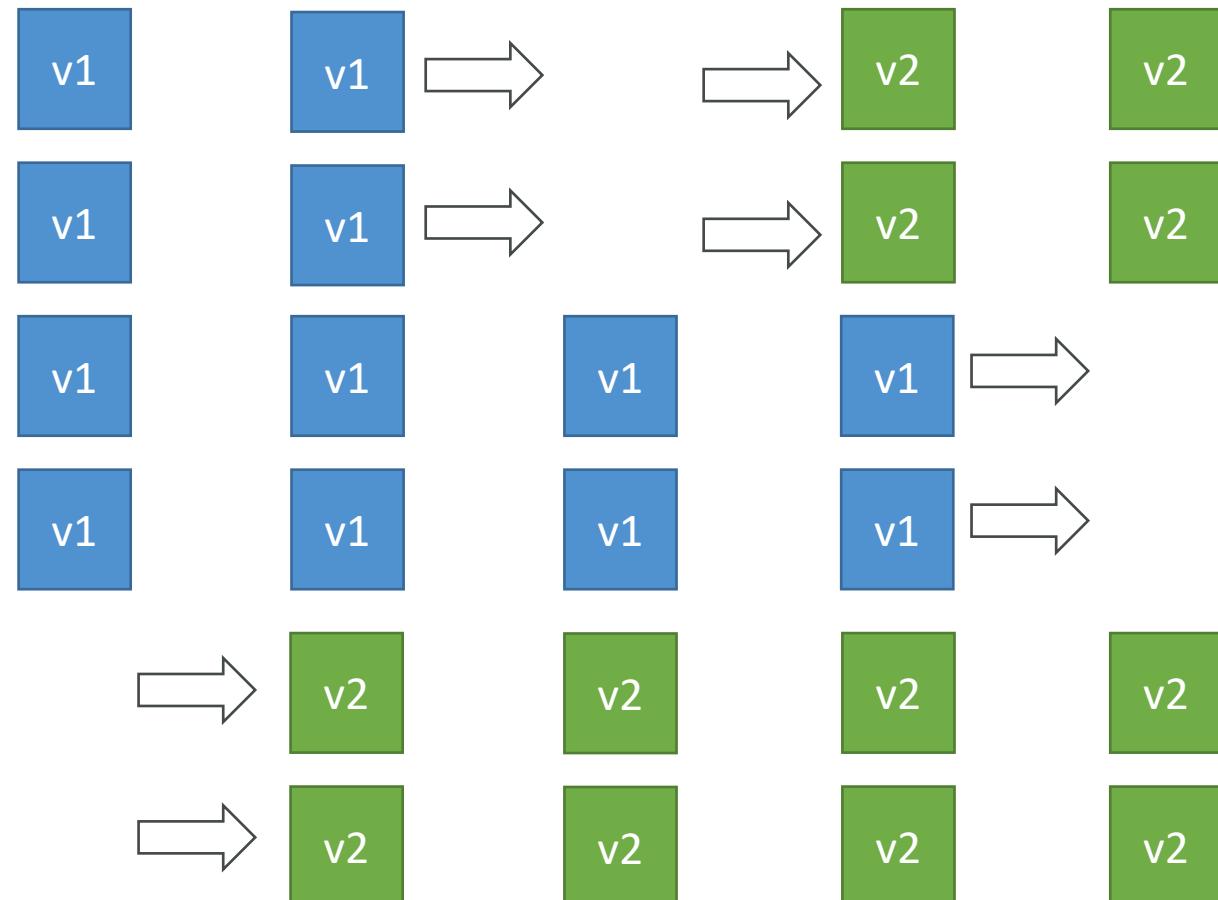
# ECS Rolling Update – Min 50%, Max 100%

- Starting number of tasks: 4

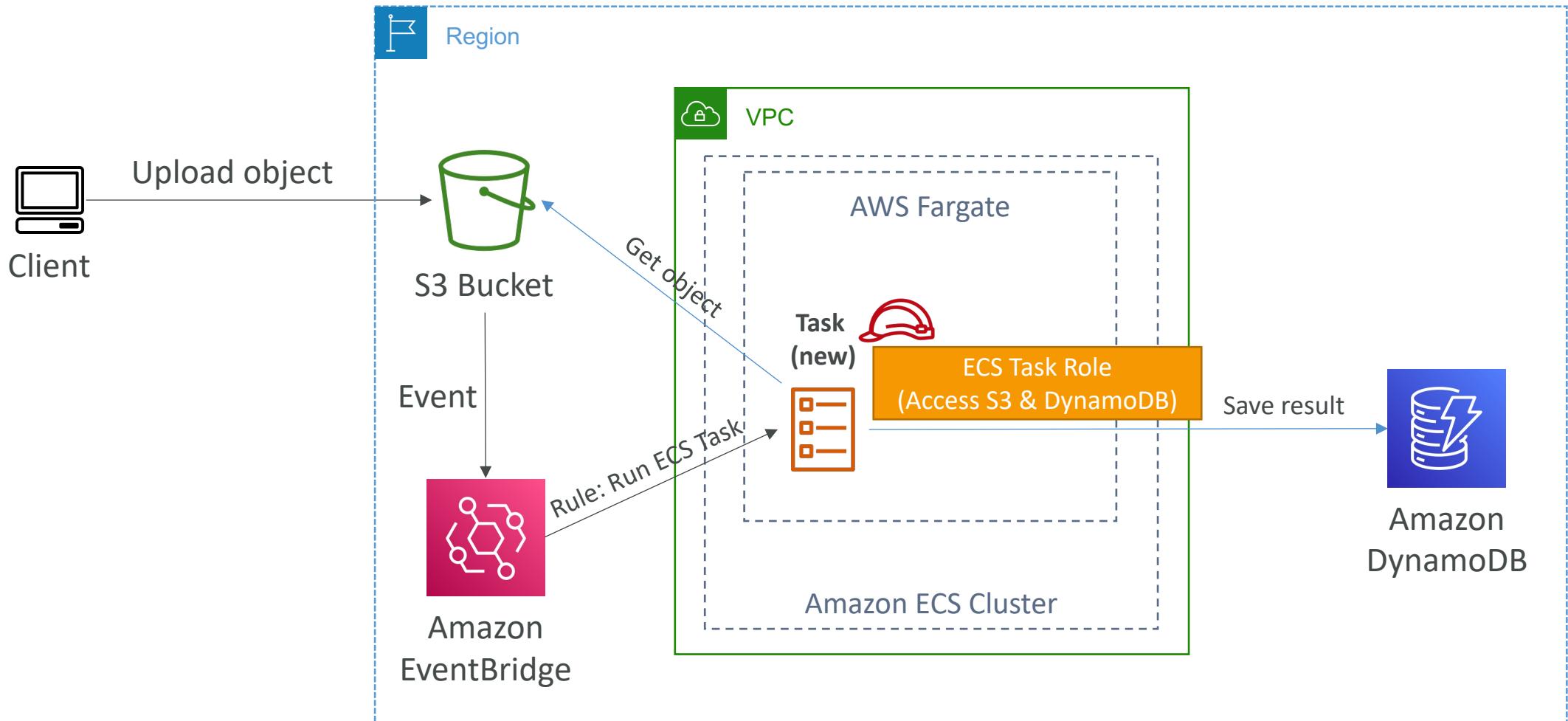


# ECS Rolling Update – Min 100%, Max 150%

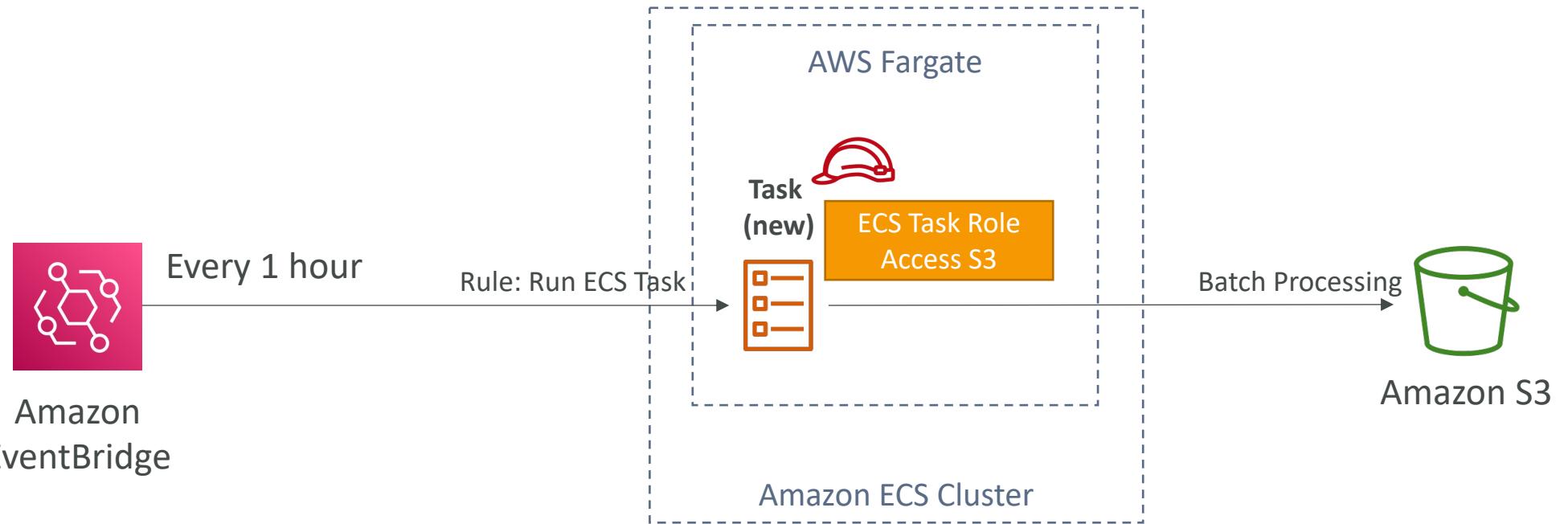
- Starting number of tasks: 4



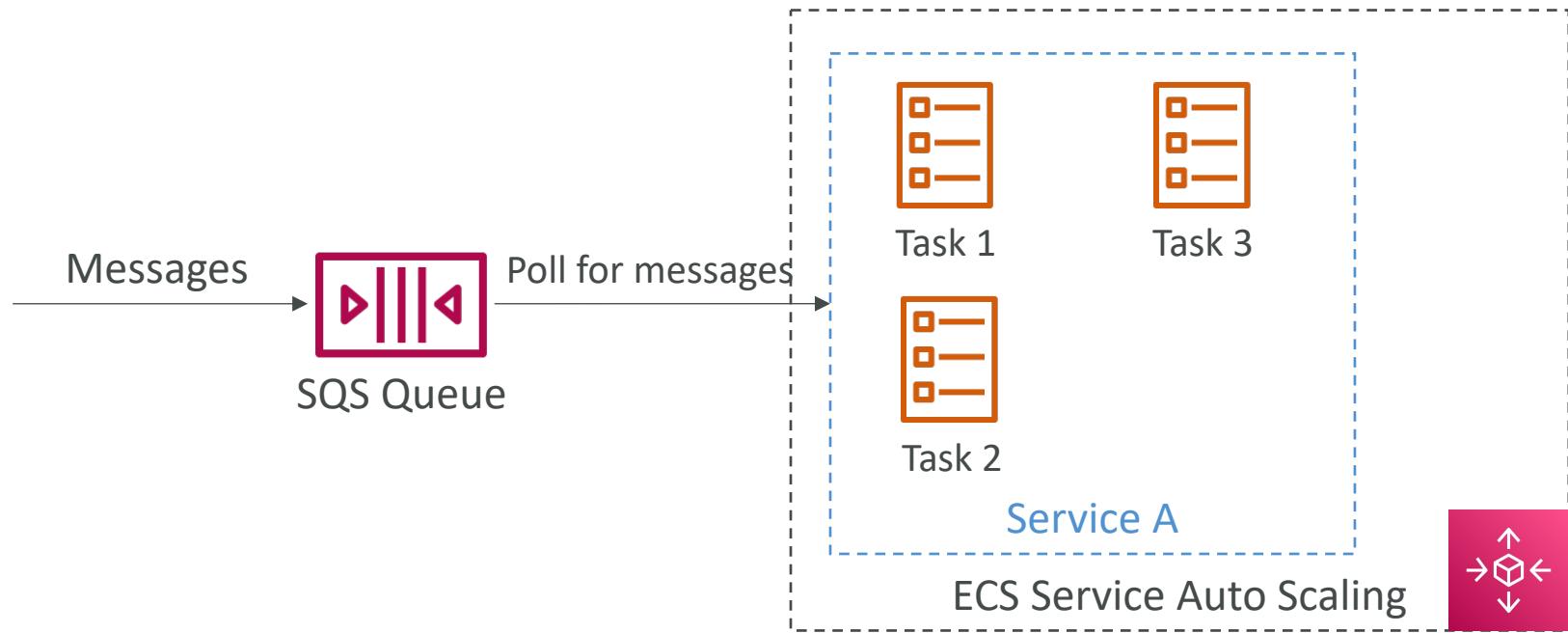
# ECS tasks invoked by Event Bridge



# ECS tasks invoked by Event Bridge Schedule



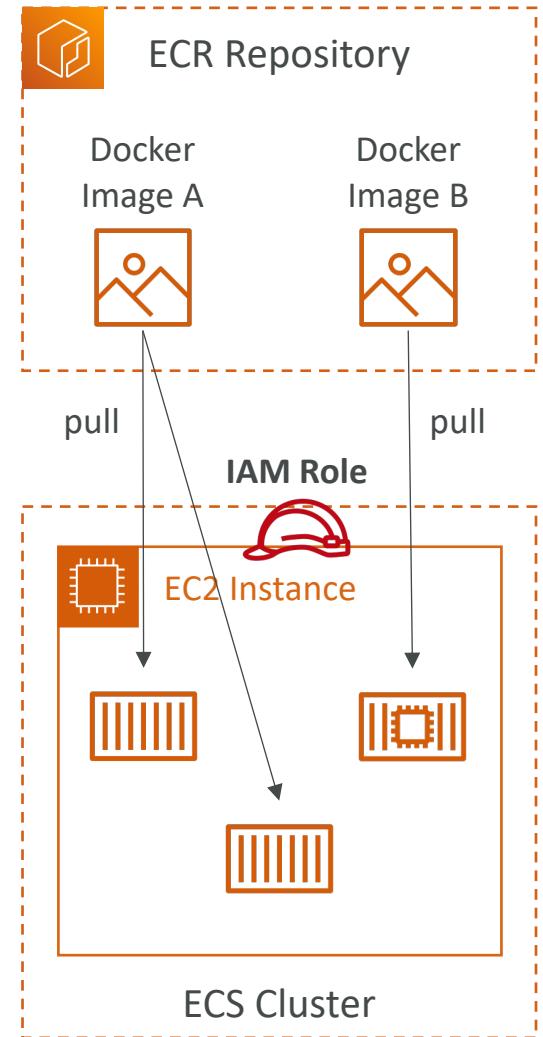
# ECS – SQS Queue Example



# Amazon ECR



- ECR = Elastic Container Registry
- Store and manage Docker images on AWS
- Private and Public repository (Amazon ECR Public Gallery <https://gallery.ecr.aws>)
- Fully integrated with ECS, backed by Amazon S3
- Access is controlled through IAM (permission errors => policy)
- Supports image vulnerability scanning, versioning, image tags, image lifecycle, ...

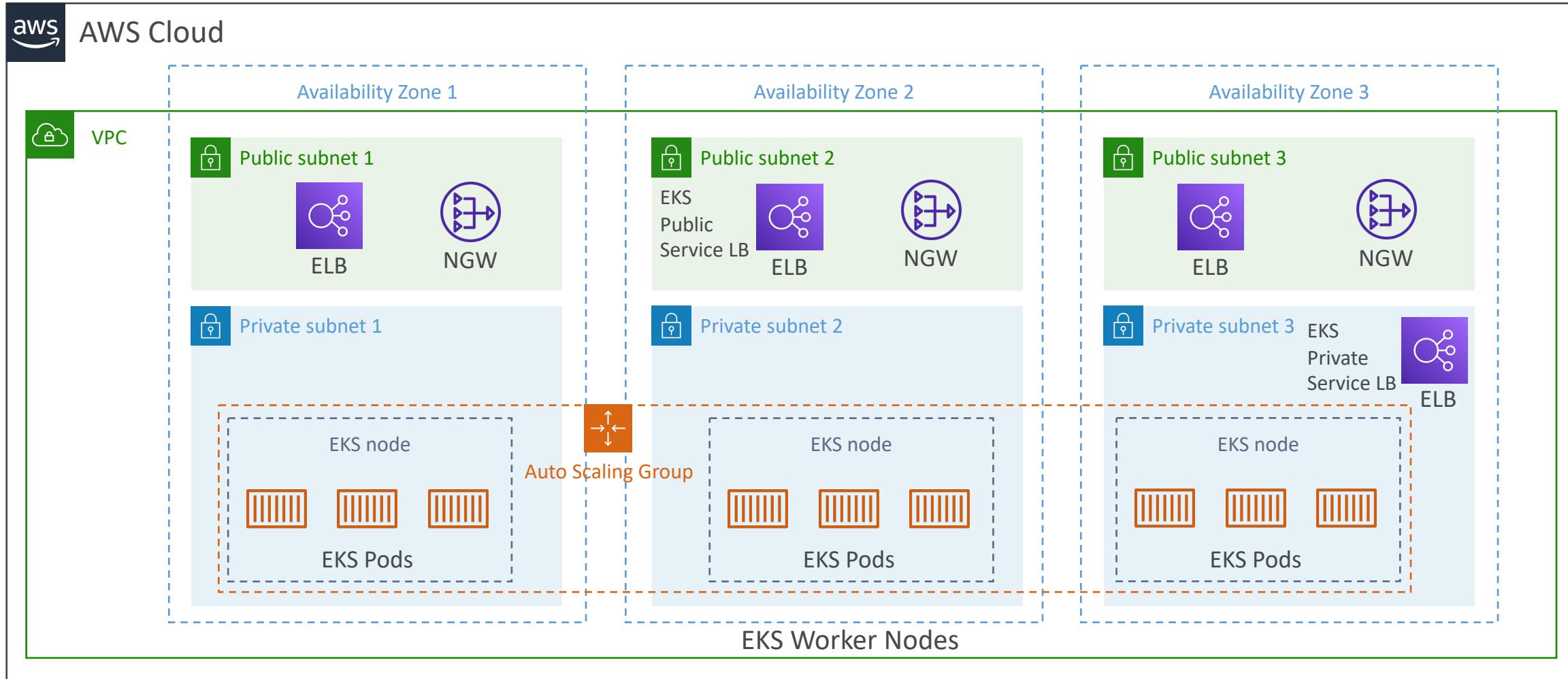


# Amazon EKS Overview



- Amazon EKS = Amazon Elastic **Kubernetes** Service
- It is a way to launch **managed Kubernetes clusters** on AWS
- Kubernetes is an **open-source system** for automatic deployment, scaling and management of containerized (usually Docker) application
- It's an alternative to ECS, similar goal but different API
- EKS supports **EC2** if you want to deploy worker nodes or **Fargate** to deploy serverless containers
- **Use case:** if your company is already using Kubernetes on-premises or in another cloud, and wants to migrate to AWS using Kubernetes
- **Kubernetes is cloud-agnostic** (can be used in any cloud – Azure, GCP...)
- For multiple regions, deploy one EKS cluster per region

# Amazon EKS - Diagram



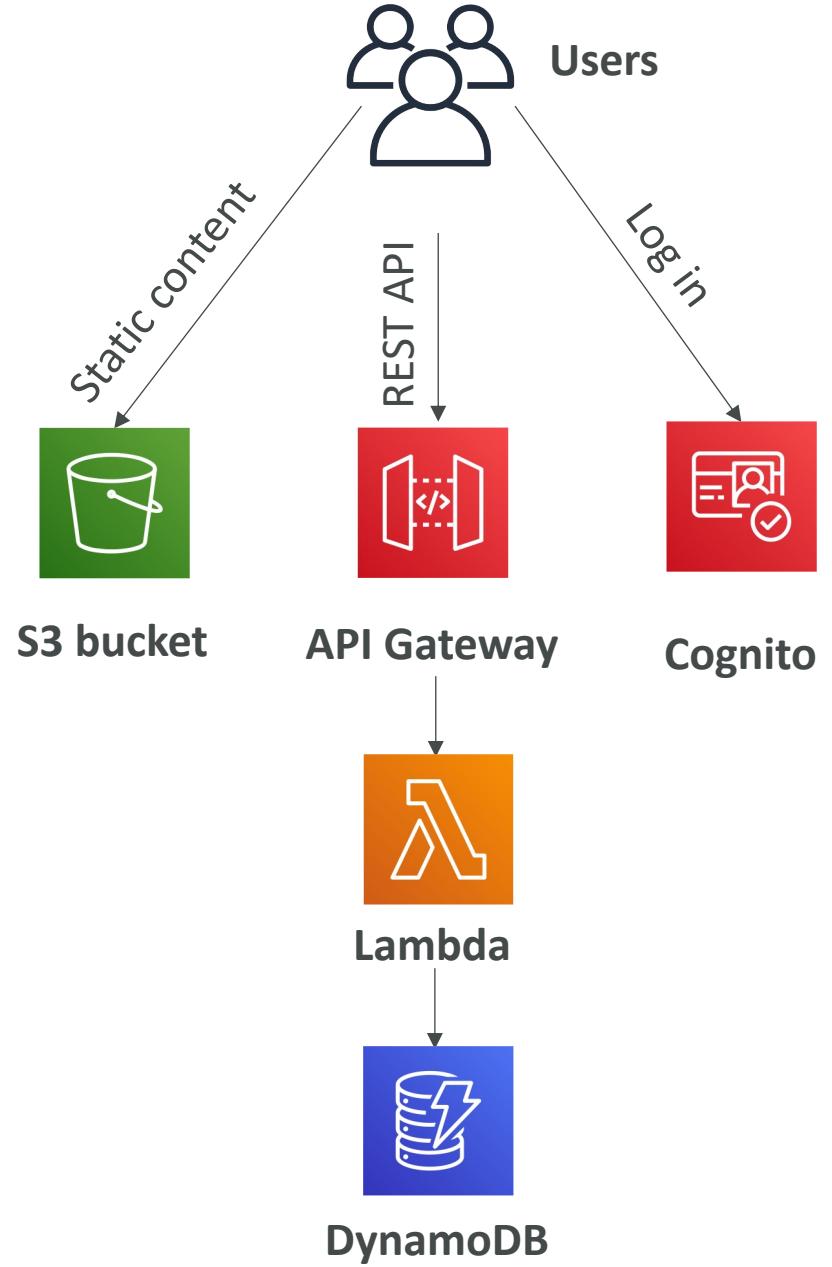
# Serverless Overview

# What's serverless?

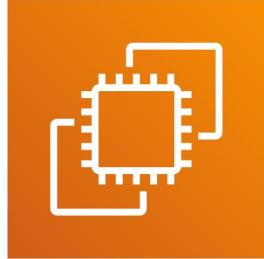
- Serverless is a new paradigm in which the developers don't have to manage servers anymore...
- They just deploy code
- They just deploy... functions !
- Initially... Serverless == FaaS (Function as a Service)
- Serverless was pioneered by AWS Lambda but now also includes anything that's managed: “databases, messaging, storage, etc.”
- **Serverless does not mean there are no servers...**  
it means you just don't manage / provision / see them

# Serverless in AWS

- AWS Lambda
- DynamoDB
- AWS Cognito
- AWS API Gateway
- Amazon S3
- AWS SNS & SQS
- AWS Kinesis Data Firehose
- Aurora Serverless
- Step Functions
- Fargate



# Why AWS Lambda



Amazon EC2

- Virtual Servers in the Cloud
  - Limited by RAM and CPU
  - Continuously running
  - Scaling means intervention to add / remove servers
- 



Amazon Lambda

- Virtual **functions** – no servers to manage!
- Limited by time - **short executions**
- Run **on-demand**
- **Scaling is automated!**

# Benefits of AWS Lambda

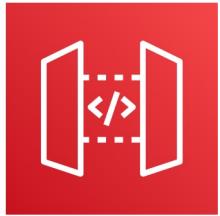
- Easy Pricing:
  - Pay per request and compute time
  - Free tier of 1,000,000 AWS Lambda requests and 400,000 GBs of compute time
- Integrated with the whole AWS suite of services
- Integrated with many programming languages
- Easy monitoring through AWS CloudWatch
- Easy to get more resources per functions (up to 10GB of RAM!)
- Increasing RAM will also improve CPU and network!

# AWS Lambda language support

- Node.js (JavaScript)
- Python
- Java (Java 8 compatible)
- C# (.NET Core)
- Golang
- C# / Powershell
- Ruby
- Custom Runtime API (community supported, example Rust)
- Lambda Container Image
  - The container image must implement the Lambda Runtime API
  - ECS / Fargate is preferred for running arbitrary Docker images

# AWS Lambda Integrations

## Main ones



API Gateway



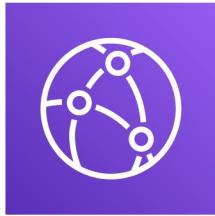
Kinesis



DynamoDB



S3



CloudFront



CloudWatch Events  
EventBridge



CloudWatch Logs



SNS

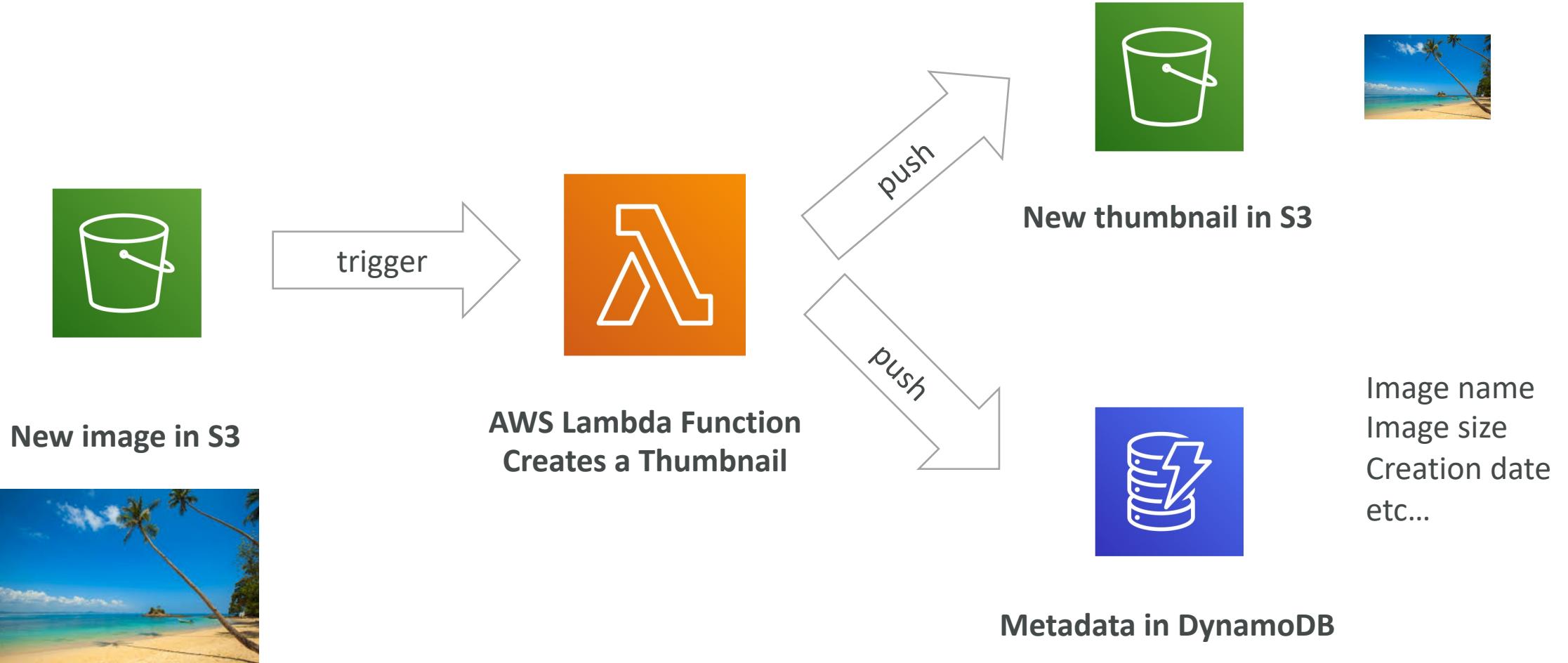


SQS



Cognito

# Example: Serverless Thumbnail creation



# Example: Serverless CRON Job



# AWS Lambda Pricing: example

- You can find overall pricing information here:  
<https://aws.amazon.com/lambda/pricing/>
- Pay per calls:
  - First 1,000,000 requests are free
  - \$0.20 per 1 million requests thereafter (\$0.0000002 per request)
- Pay per duration: (in increment of 1 ms)
  - 400,000 GB-seconds of compute time per month for FREE
  - == 400,000 seconds if function is 1 GB RAM
  - == 3,200,000 seconds if function is 128 MB RAM
  - After that \$1.00 for 600,000 GB-seconds
- It is usually very cheap to run AWS Lambda so it's very popular

# AWS Lambda Limits to Know - per region

- **Execution:**
  - Memory allocation: 128 MB – 10GB (1 MB increments)
  - Maximum execution time: 900 seconds (15 minutes)
  - Environment variables (4 KB)
  - Disk capacity in the “function container” (in /tmp): 512 MB
  - Concurrency executions: 1000 (can be increased)
- **Deployment:**
  - Lambda function deployment size (compressed .zip): 50 MB
  - Size of uncompressed deployment (code + dependencies): 250 MB
  - Can use the /tmp directory to load other files at startup
  - Size of environment variables: 4 KB

# Lambda@Edge

- You have deployed a CDN using CloudFront
- What if you wanted to run a global AWS Lambda alongside?
- Or how to implement request filtering before reaching your application?
- For this, you can use **Lambda@Edge**:  
deploy Lambda functions alongside your CloudFront CDN
  - Build more responsive applications
  - You don't manage servers, Lambda is deployed globally
  - Customize the CDN content
  - Pay only for what you use

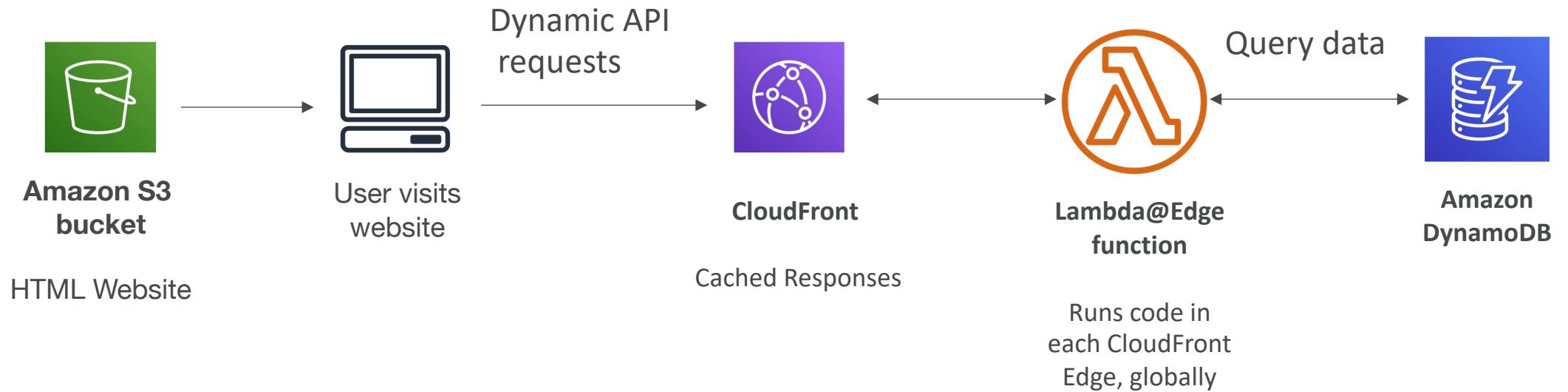
# Lambda@Edge

- You can use Lambda to change CloudFront requests and responses:
  - After CloudFront receives a request from a viewer (viewer request)
  - Before CloudFront forwards the request to the origin (origin request)
  - After CloudFront receives the response from the origin (origin response)
  - Before CloudFront forwards the response to the viewer (viewer response)



- You can also generate responses to viewers without ever sending the request to the origin

# Lambda@Edge: Global application



# Lambda@Edge: Use Cases

- Website Security and Privacy
- Dynamic Web Application at the Edge
- Search Engine Optimization (SEO)
- Intelligently Route Across Origins and Data Centers
- Bot Mitigation at the Edge
- Real-time Image Transformation
- A/B Testing
- User Authentication and Authorization
- User Prioritization
- User Tracking and Analytics

# Amazon DynamoDB



- Fully managed, highly available with replication across multiple AZs
- NoSQL database - not a relational database
- Scales to massive workloads, distributed database
- Millions of requests per seconds, trillions of row, 100s of TB of storage
- Fast and consistent in performance (low latency on retrieval)
- Integrated with IAM for security, authorization and administration
- Enables event driven programming with DynamoDB Streams
- Low cost and auto-scaling capabilities
- Standard & Infrequent Access (IA) Table Class

# DynamoDB - Basics

- DynamoDB is made of **Tables**
- Each table has a **Primary Key** (must be decided at creation time)
- Each table can have an infinite number of items (= rows)
- Each item has **attributes** (can be added over time – can be null)
- Maximum size of an item is **400KB**
- Data types supported are:
  - **Scalar Types** – String, Number, Binary, Boolean, Null
  - **Document Types** – List, Map
  - **Set Types** – String Set, Number Set, Binary Set

# DynamoDB – Table example

Primary Key		Attributes	
Partition Key	Sort Key	Score	Result
User_ID	Game_ID	Score	Result
7791a3d6...	4421	92	Win
873e0634...	1894	14	Lose
873e0634...	4521	77	Win

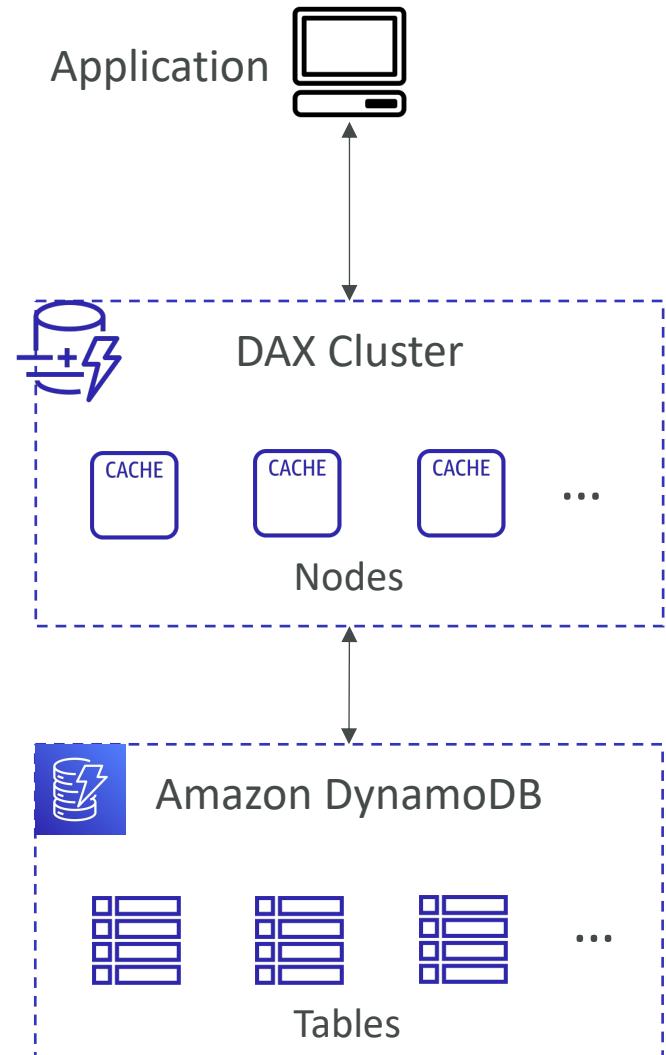
# DynamoDB – Read/Write Capacity Modes

- Control how you manage your table's capacity (read/write throughput)
- Provisioned Mode (default)
  - You specify the number of reads/writes per second
  - You need to plan capacity beforehand
  - Pay for provisioned Read Capacity Units (RCU) & Write Capacity Units (WCUs)
  - Possibility to add auto-scaling mode for RCU & WCU
- On-Demand Mode
  - Read/writes automatically scale up/down with your workloads
  - No capacity planning needed
  - Pay for what you use, more expensive (\$\$\$)
  - Great for unpredictable workloads

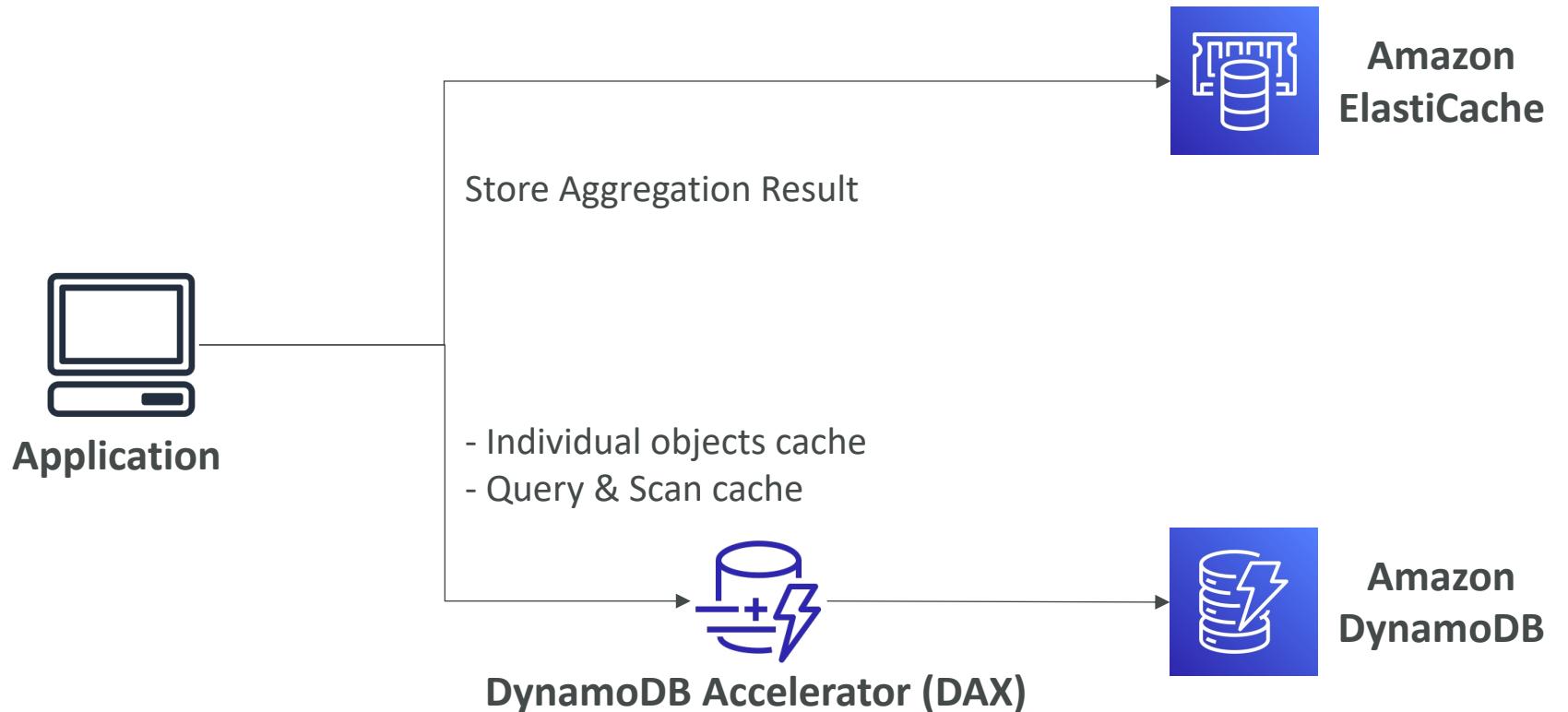
# DynamoDB Accelerator (DAX)



- Fully-managed, highly available, seamless in-memory cache for DynamoDB
- Help solve read congestion by caching
- Microseconds latency for cached data
- Doesn't require application logic modification (compatible with existing DynamoDB APIs)
- 5 minutes TTL for cache (default)



# DynamoDB Accelerator (DAX) vs. ElastiCache

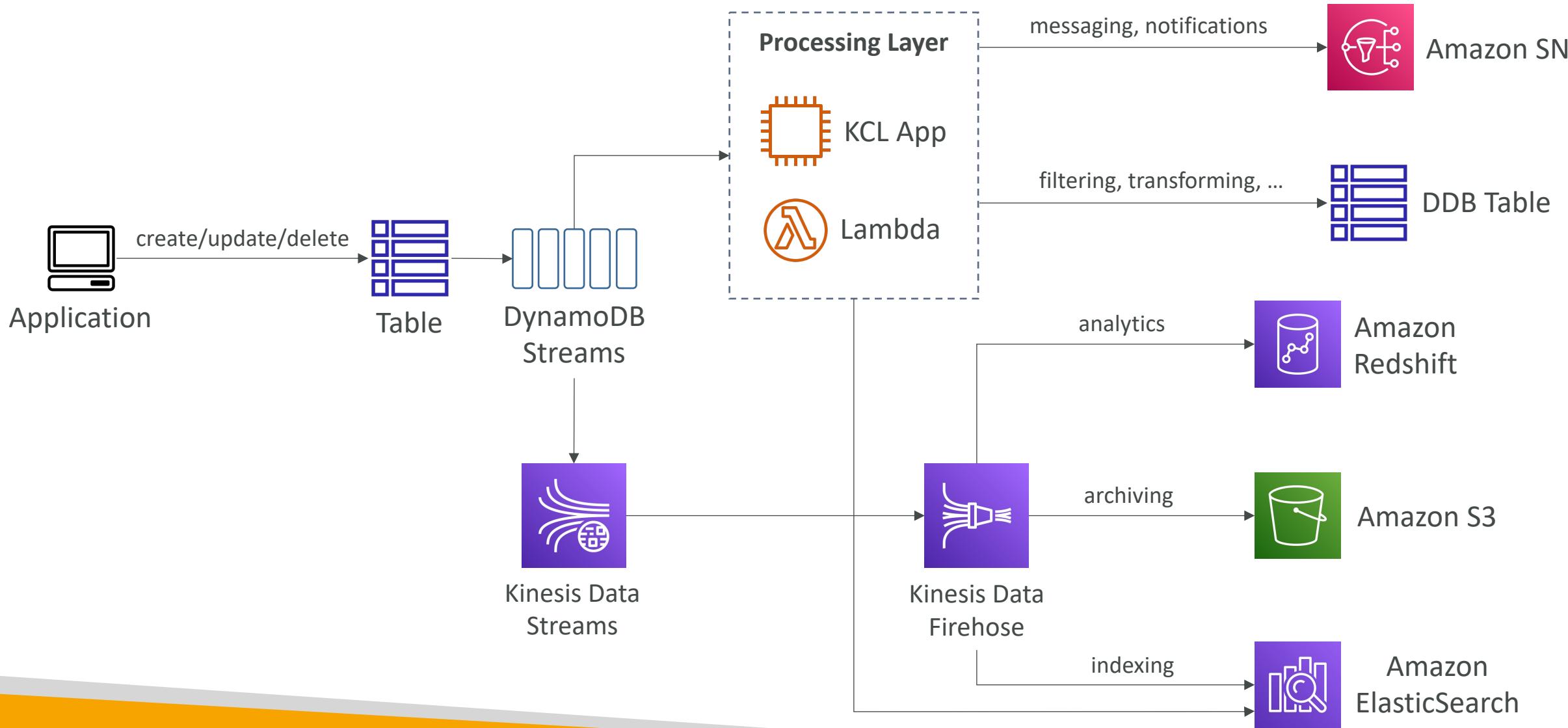




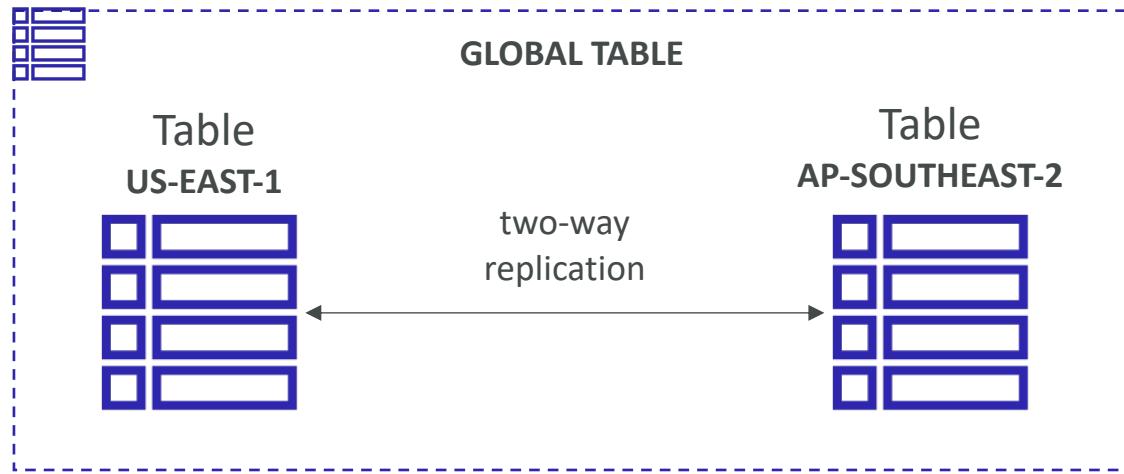
# DynamoDB Streams

- Ordered stream of item-level modifications (create/update/delete) in a table
- Stream records can be:
  - Sent to Kinesis Data Streams
  - Read by AWS Lambda
  - Read by Kinesis Client Library applications
- Data Retention for up to 24 hours
- Use cases:
  - react to changes in real-time (welcome email to users)
  - Analytics
  - Insert into derivative tables
  - Insert into ElasticSearch
  - Implement cross-region replication

# DynamoDB Streams



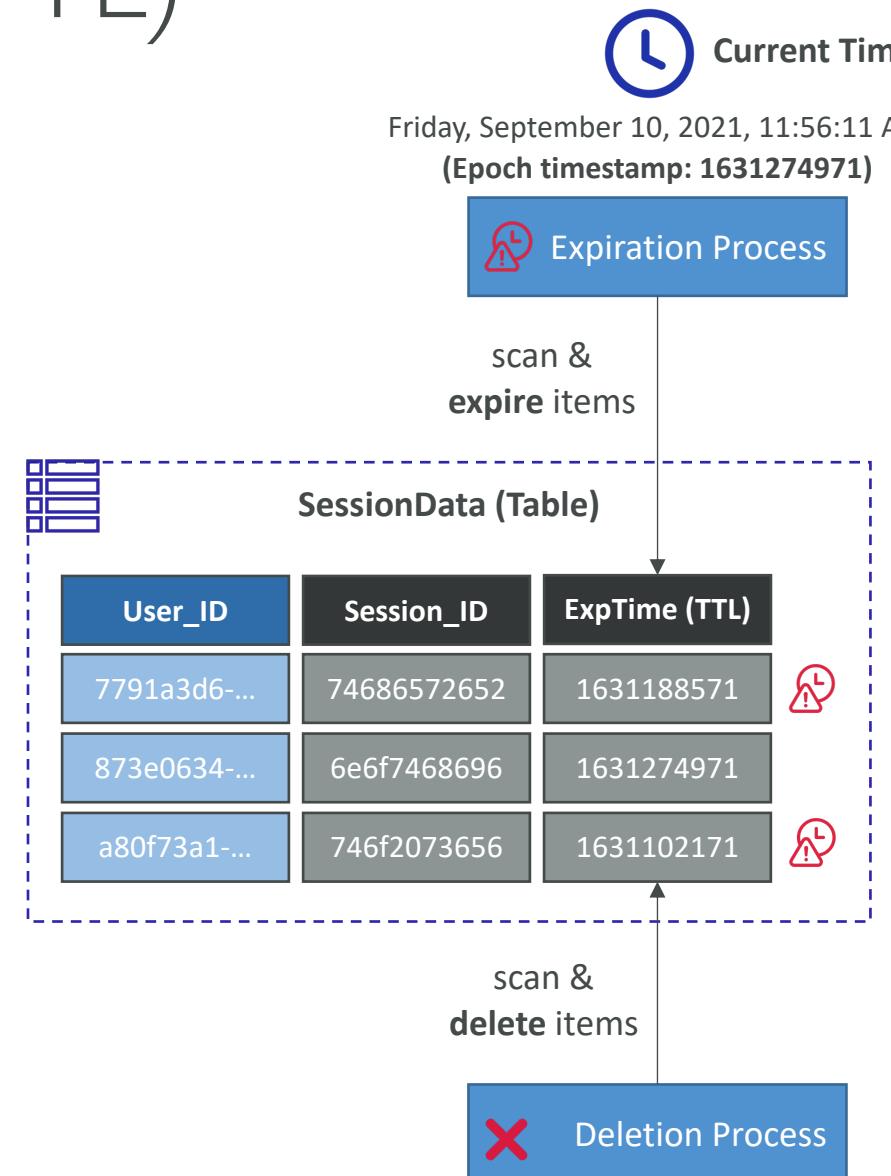
# DynamoDB Global Tables



- Make a DynamoDB table accessible with **low latency** in multiple-regions
- Active-Active replication
- Applications can **READ** and **WRITE** to the table in any region
- Must enable DynamoDB Streams as a pre-requisite

# DynamoDB – Time To Live (TTL)

- Automatically delete items after an expiry timestamp
- Use cases: reduce stored data by keeping only current items, adhere to regulatory obligations, ...



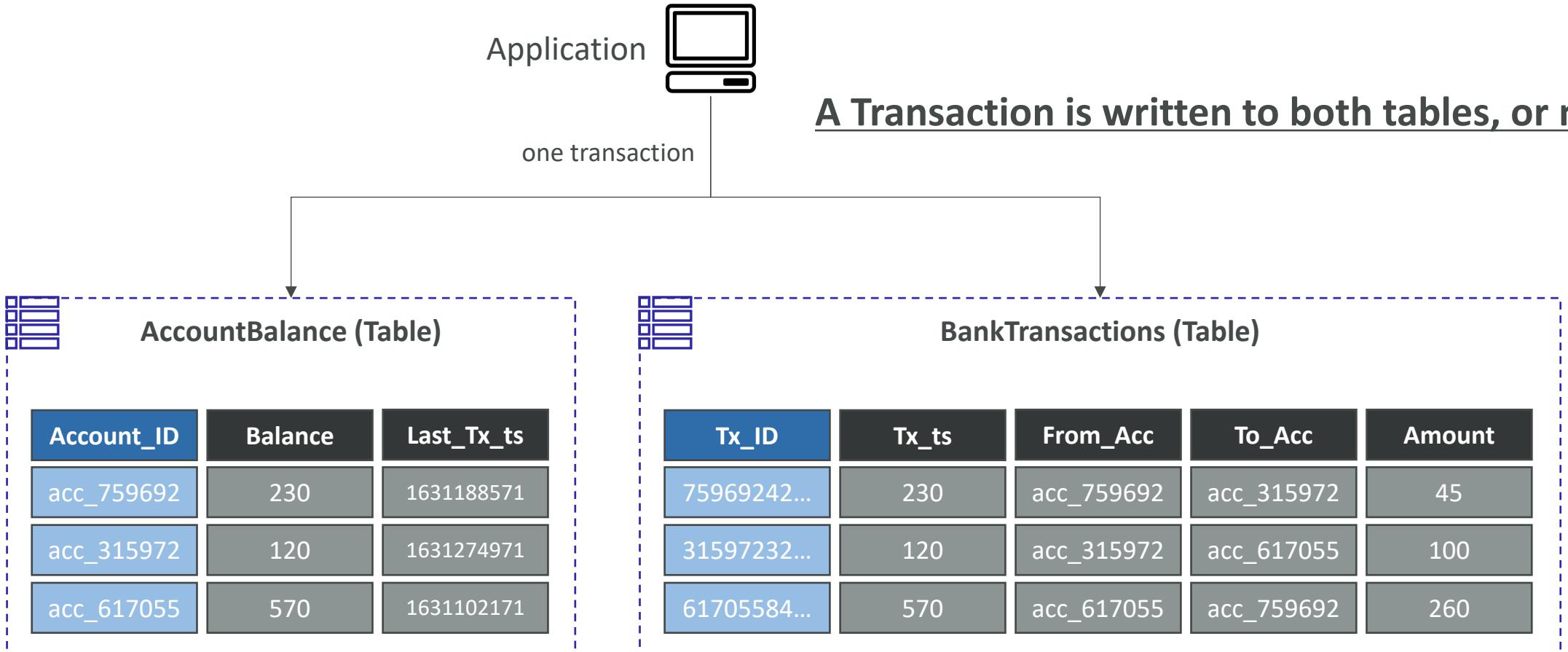
# DynamoDB - Indexes

- Global Secondary Indexes (GSI) & Local Secondary Indexes (LSI)
- High level: allow to query on attributes other than the Primary Key

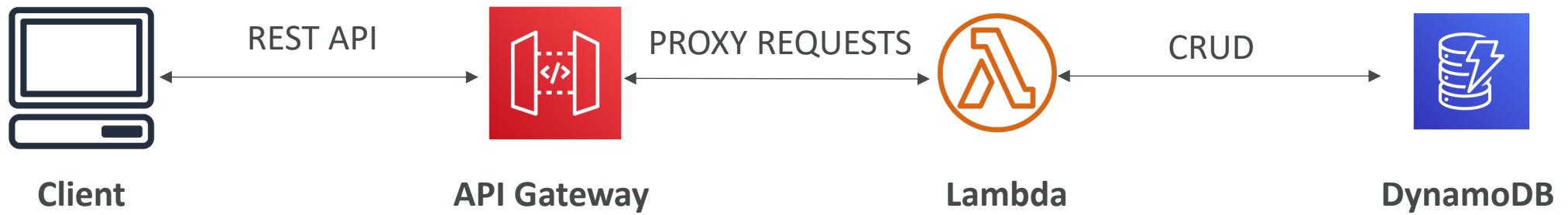
Primary Key		Attributes		
Partition Key	Sort Key	Game_TS	Score	Result
User_ID	Game_ID	Game_TS	Score	Result
7791a3d6...	4421	"2021-03-15T17:43:08"	92	Win
873e0634...	4521	"2021-06-20T19:02:32"		Lose
a80f73a1...	1894	"2021-02-11T04:11:31"	77	Win

- With Indexes, we can query by Game ID, Game\_TS, Score, Result, etc...

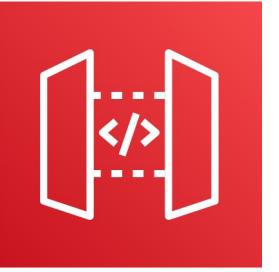
# DynamoDB - Transactions



# Example: Building a Serverless API



# AWS API Gateway



- AWS Lambda + API Gateway: No infrastructure to manage
- Support for the WebSocket Protocol
- Handle API versioning (v1, v2...)
- Handle different environments (dev, test, prod...)
- Handle security (Authentication and Authorization)
- Create API keys, handle request throttling
- Swagger / Open API import to quickly define APIs
- Transform and validate requests and responses
- Generate SDK and API specifications
- Cache API responses

# API Gateway – Integrations High Level

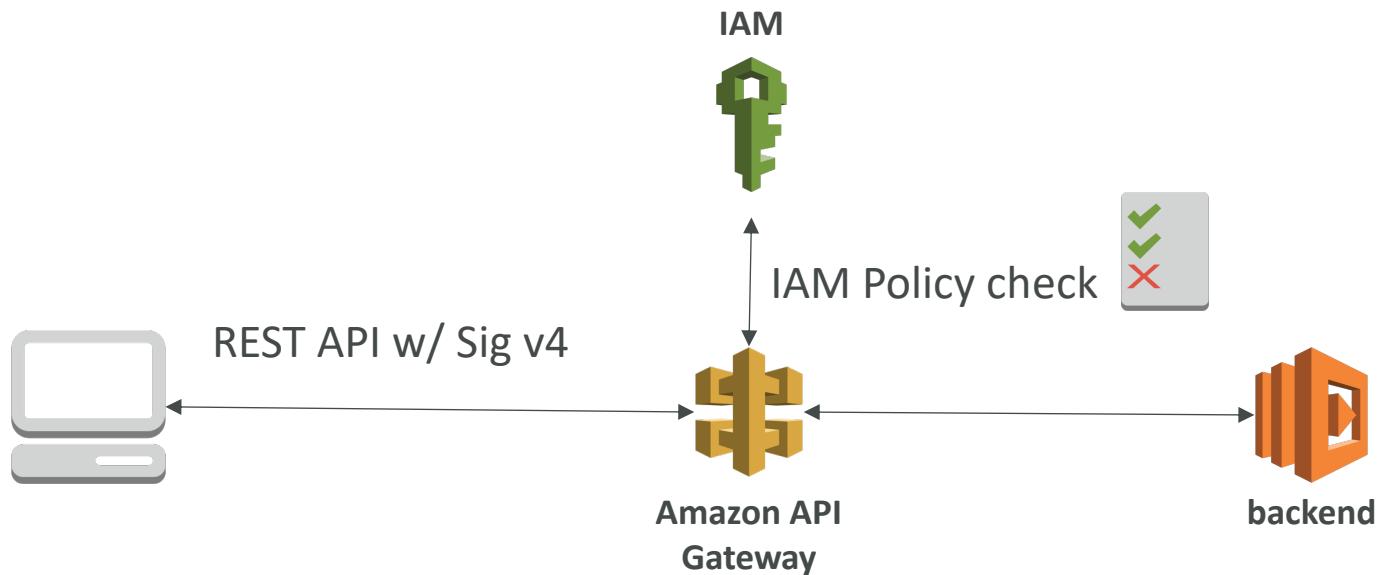
- Lambda Function
  - Invoke Lambda function
  - Easy way to expose REST API backed by AWS Lambda
- HTTP
  - Expose HTTP endpoints in the backend
  - Example: internal HTTP API on premise, Application Load Balancer...
  - Why? Add rate limiting, caching, user authentications, API keys, etc...
- AWS Service
  - Expose any AWS API through the API Gateway?
  - Example: start an AWS Step Function workflow, post a message to SQS
  - Why? Add authentication, deploy publicly, rate control...

# API Gateway - Endpoint Types

- **Edge-Optimized (default):** For global clients
  - Requests are routed through the CloudFront Edge locations (improves latency)
  - The API Gateway still lives in only one region
- **Regional:**
  - For clients within the same region
  - Could manually combine with CloudFront (more control over the caching strategies and the distribution)
- **Private:**
  - Can only be accessed from your VPC using an interface VPC endpoint (ENI)
  - Use a resource policy to define access

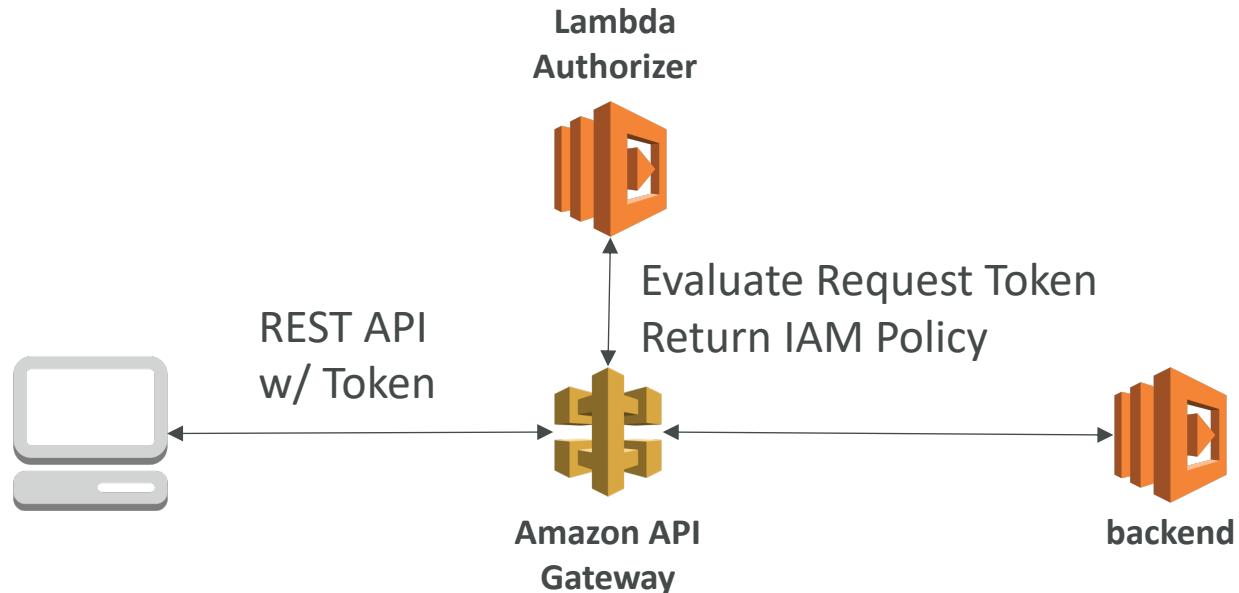
# API Gateway – Security IAM Permissions

- Create an IAM policy authorization and attach to User / Role
- API Gateway verifies IAM permissions passed by the calling application
- Good to provide access within your own infrastructure
- Leverages “Sig v4” capability where IAM credential are in headers



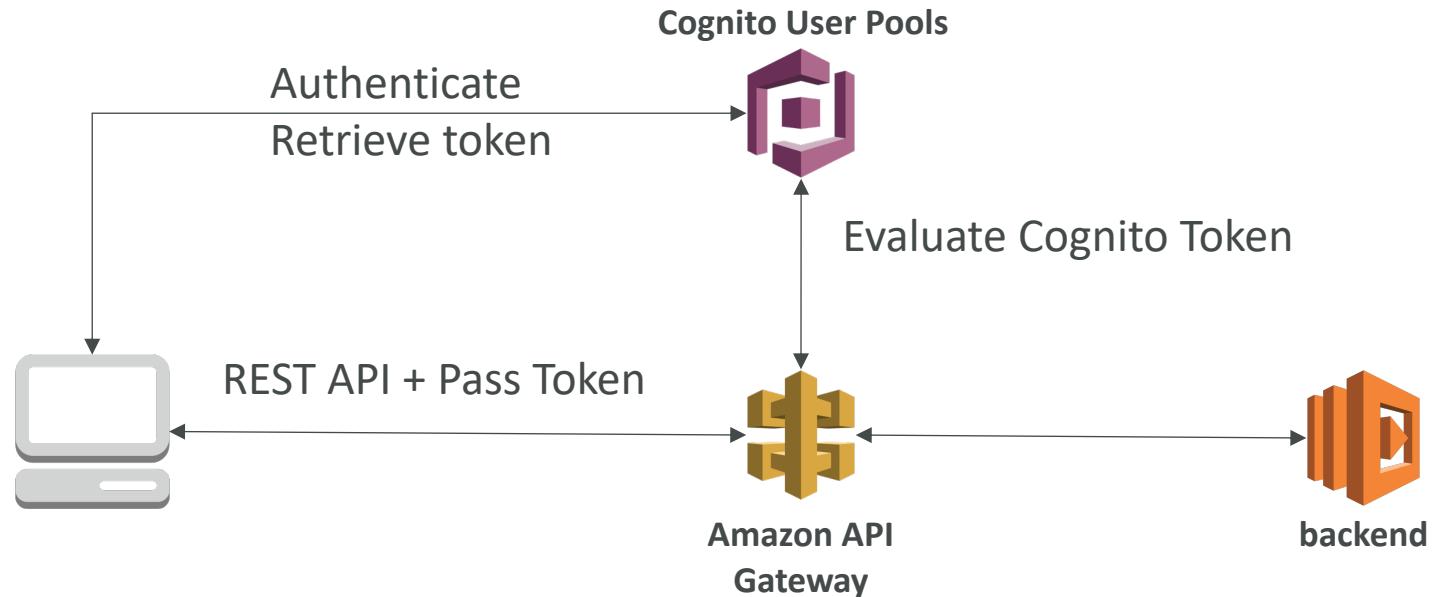
# API Gateway – Security Lambda Authorizer (formerly Custom Authorizers)

- Uses AWS Lambda to validate the token in header being passed
- Option to cache result of authentication
- Helps to use OAuth / SAML / 3<sup>rd</sup> party type of authentication
- Lambda must return an IAM policy for the user



# API Gateway – Security Cognito User Pools

- Cognito fully manages user lifecycle
- API gateway verifies identity automatically from AWS Cognito
- No custom implementation required
- Cognito only helps with authentication, not authorization



# API Gateway – Security – Summary

- **IAM:**
  - Great for users / roles already within your AWS account
  - Handle authentication + authorization
  - Leverages Sig v4
- **Custom Authorizer:**
  - Great for 3<sup>rd</sup> party tokens
  - Very flexible in terms of what IAM policy is returned
  - Handle Authentication + Authorization
  - Pay per Lambda invocation
- **Cognito User Pool:**
  - You manage your own user pool (can be backed by Facebook, Google login etc...)
  - No need to write any custom code
  - Must implement authorization in the backend

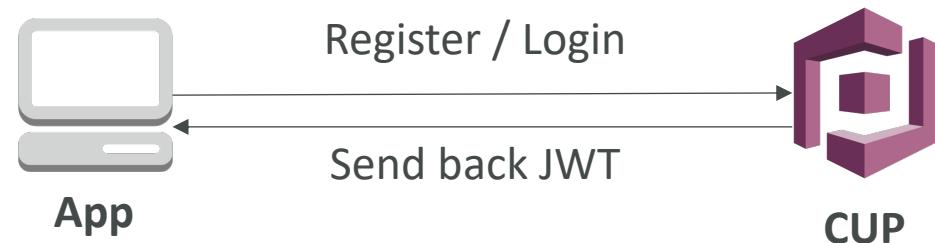


# AWS Cognito

- We want to give our users an identity so that they can interact with our application.
- **Cognito User Pools:**
  - Sign in functionality for app users
  - Integrate with API Gateway
- **Cognito Identity Pools (Federated Identity):**
  - Provide AWS credentials to users so they can access AWS resources directly
  - Integrate with Cognito User Pools as an identity provider
- **Cognito Sync:**
  - Synchronize data from device to Cognito.
  - May be deprecated and replaced by AppSync

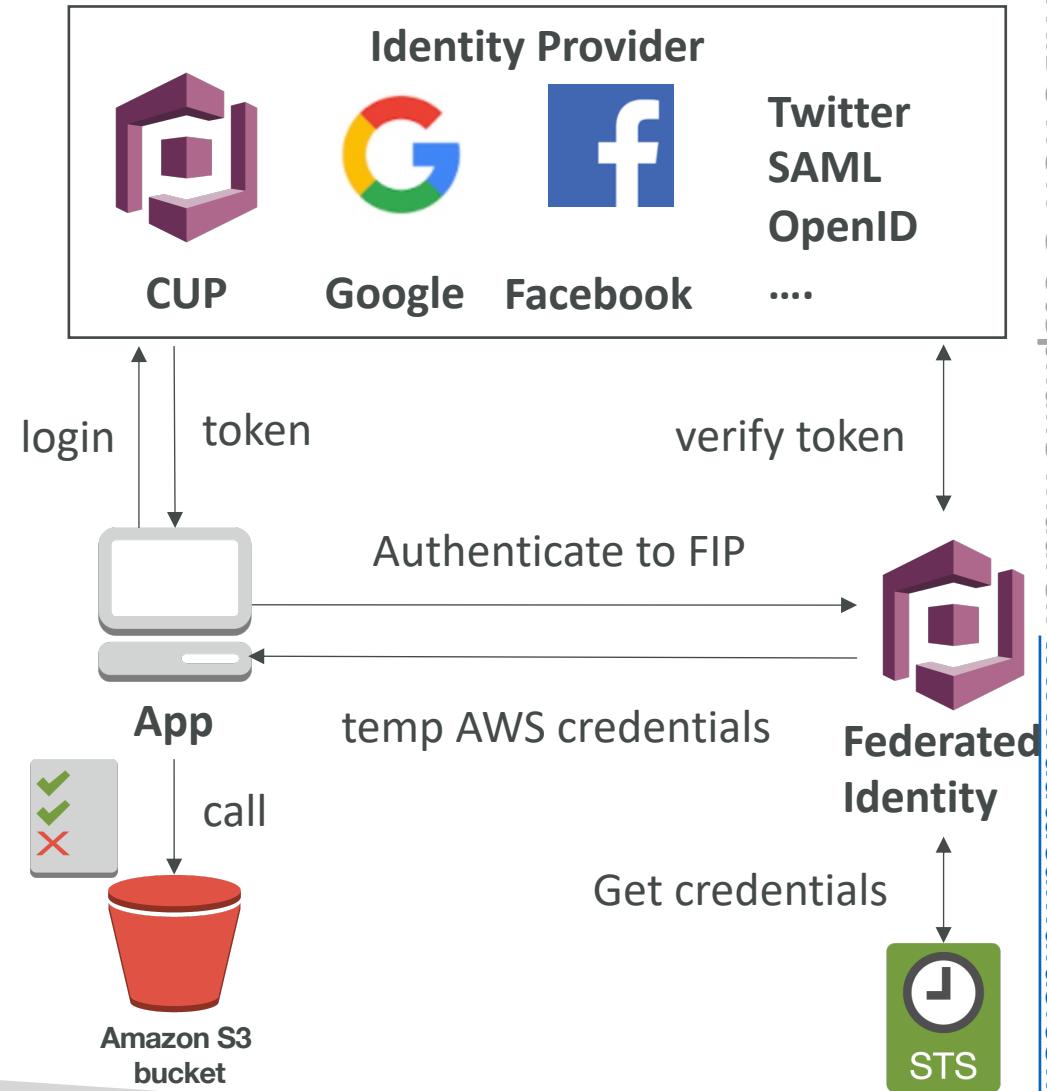
# AWS Cognito User Pools (CUP)

- Create a serverless database of user for your mobile apps
- Simple login: Username (or email) / password combination
- Possibility to verify emails / phone numbers and add MFA
- Can enable Federated Identities (Facebook, Google, SAML...)
- Sends back a JSON Web Tokens (JWT)
- Can be integrated with API Gateway for authentication



# AWS Cognito – Federated Identity Pools

- Goal:
  - Provide direct access to AWS Resources from the Client Side
- How:
  - Log in to federated identity provider – or remain anonymous
  - Get temporary AWS credentials back from the Federated Identity Pool
  - These credentials come with a pre-defined IAM policy stating their permissions
- Example:
  - provide (temporary) access to write to S3 bucket using Facebook Login



# AWS Cognito Sync

- Deprecated – use AWS AppSync now
- Store preferences, configuration, state of app
- Cross device synchronization (any platform – iOS, Android, etc...)
- Offline capability (synchronization when back online)
- Requires Federated Identity Pool in Cognito (not User Pool)
- Store data in datasets (up to 1MB)
- Up to 20 datasets to synchronise

# AWS SAM - Serverless Application Model



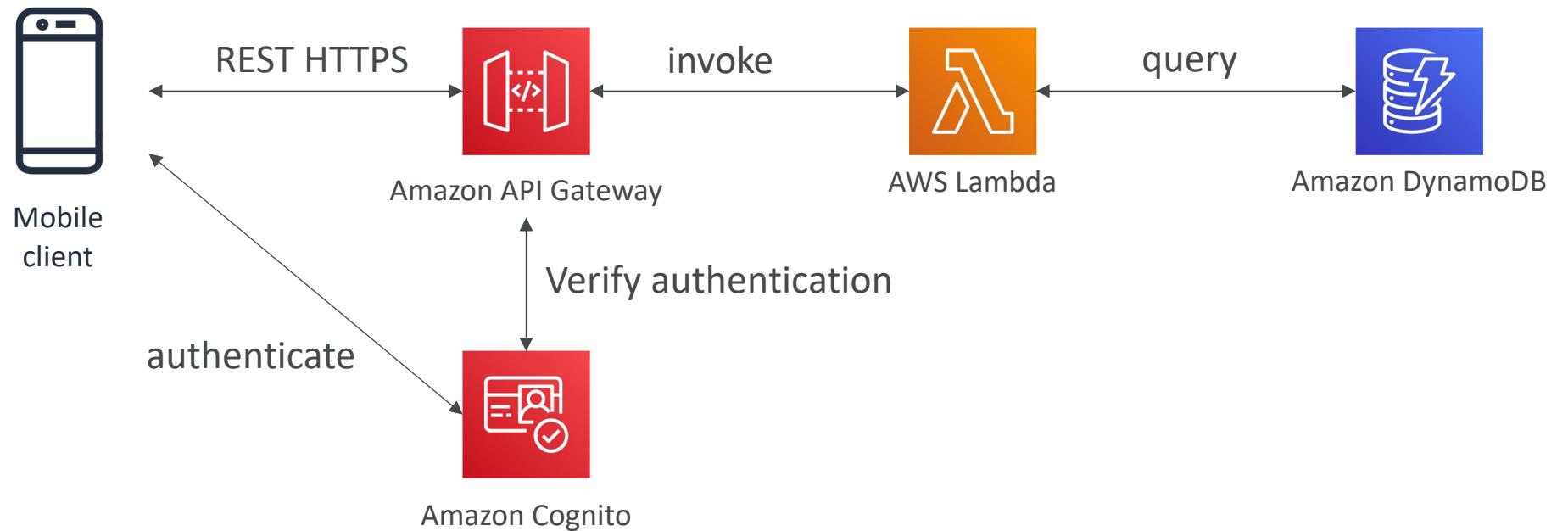
- SAM = Serverless Application Model
- Framework for developing and deploying serverless applications
- All the configuration is YAML code
  - Lambda Functions
  - DynamoDB tables
  - API Gateway
  - Cognito User Pools
- SAM can help you to run Lambda, API Gateway, DynamoDB locally
- SAM can use CodeDeploy to deploy Lambda functions

# Serverless Architectures

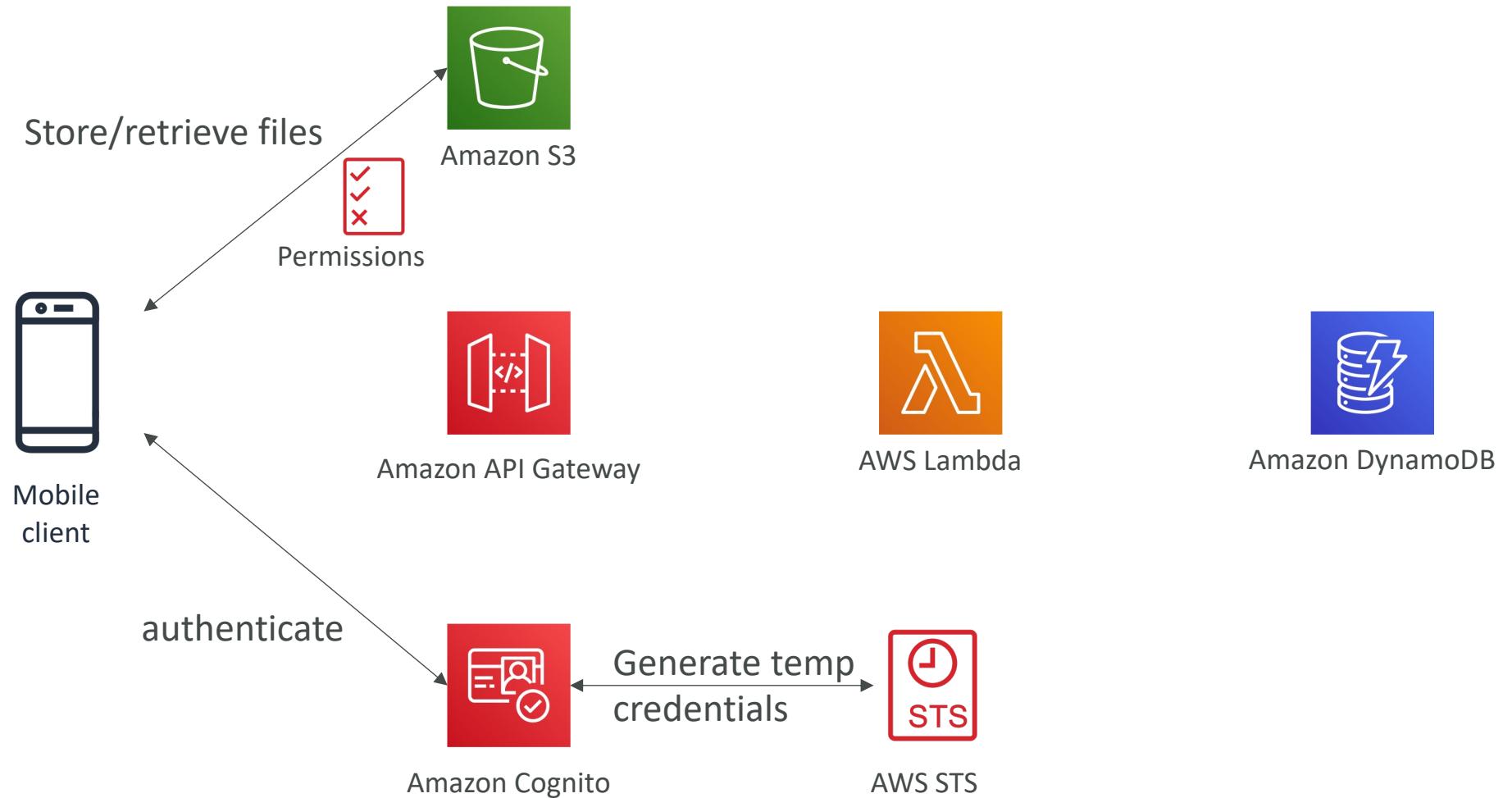
# Mobile application: MyTodoList

- We want to create a mobile application with the following requirements
- Expose as REST API with HTTPS
- Serverless architecture
- Users should be able to directly interact with their own folder in S3
- Users should authenticate through a managed serverless service
- The users can write and read to-dos, but they mostly read them
- The database should scale, and have some high read throughput

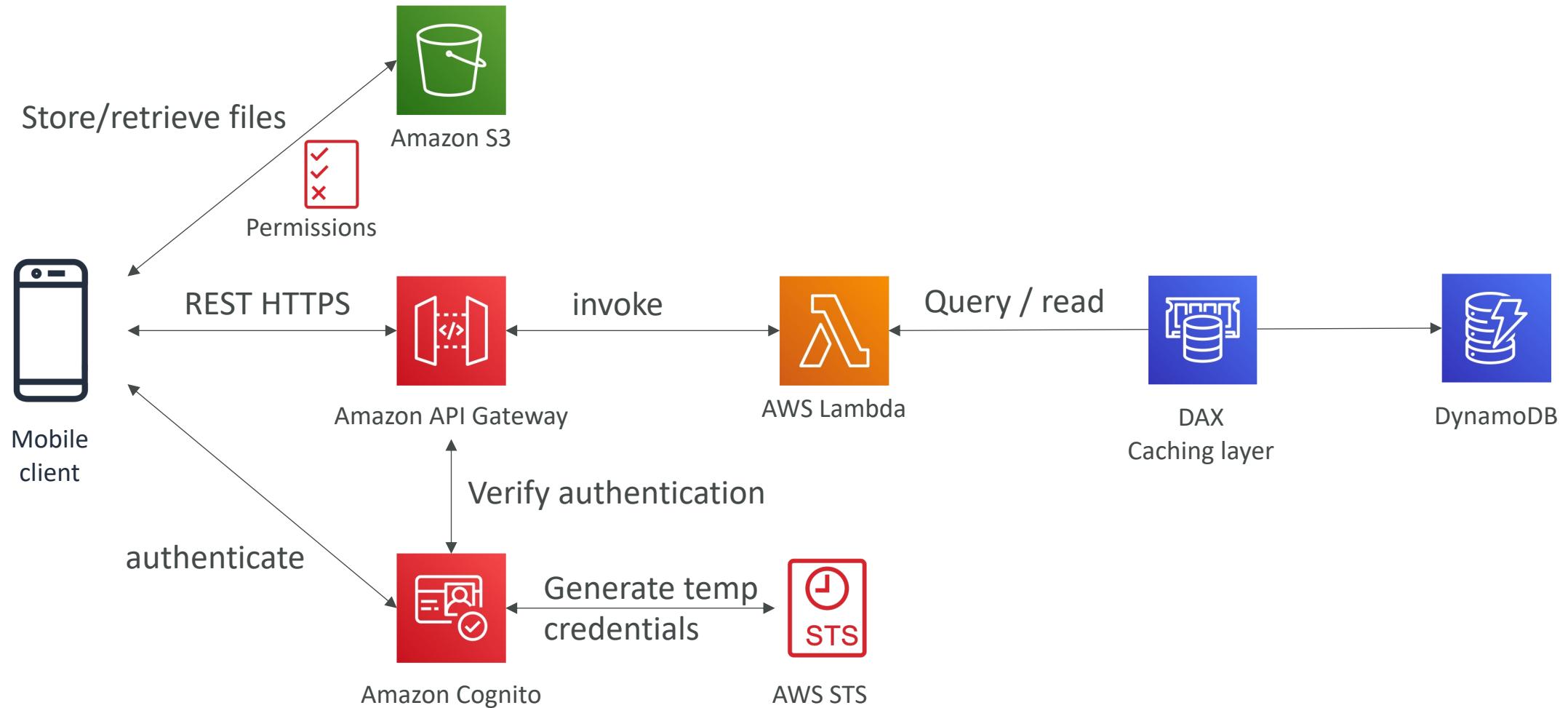
# Mobile app: REST API layer



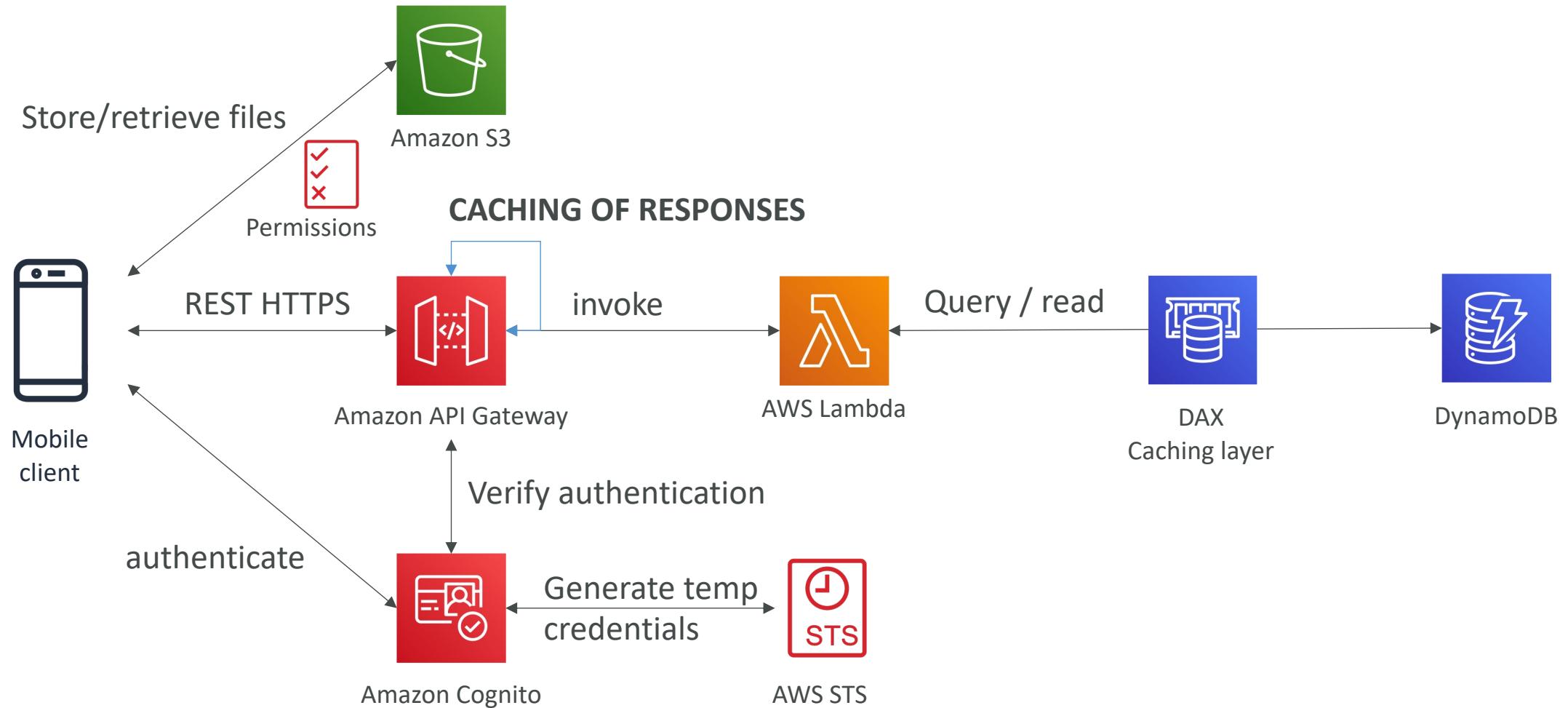
# Mobile app: giving users access to S3



# Mobile app: high read throughput, static data



# Mobile app: caching at the API Gateway



# In this lecture

- Serverless REST API: HTTPS, API Gateway, Lambda, DynamoDB
- Using Cognito to generate temporary credentials with STS to access S3 bucket with restricted policy. App users can directly access AWS resources this way. Pattern can be applied to DynamoDB, Lambda...
- Caching the reads on DynamoDB using DAX
- Caching the REST requests at the API Gateway level
- Security for authentication and authorization with Cognito, STS

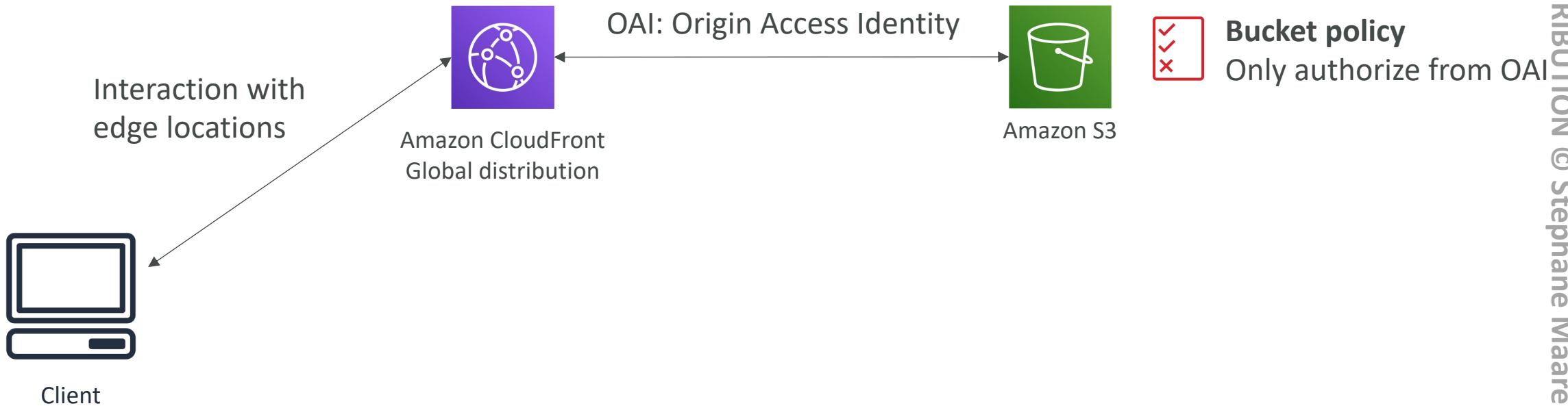
# Serverless hosted website: MyBlog.com

- This website should scale globally
- Blogs are rarely written, but often read
- Some of the website is purely static files, the rest is a dynamic REST API
- Caching must be implemented where possible
- Any new users that subscribe should receive a welcome email
- Any photo uploaded to the blog should have a thumbnail generated

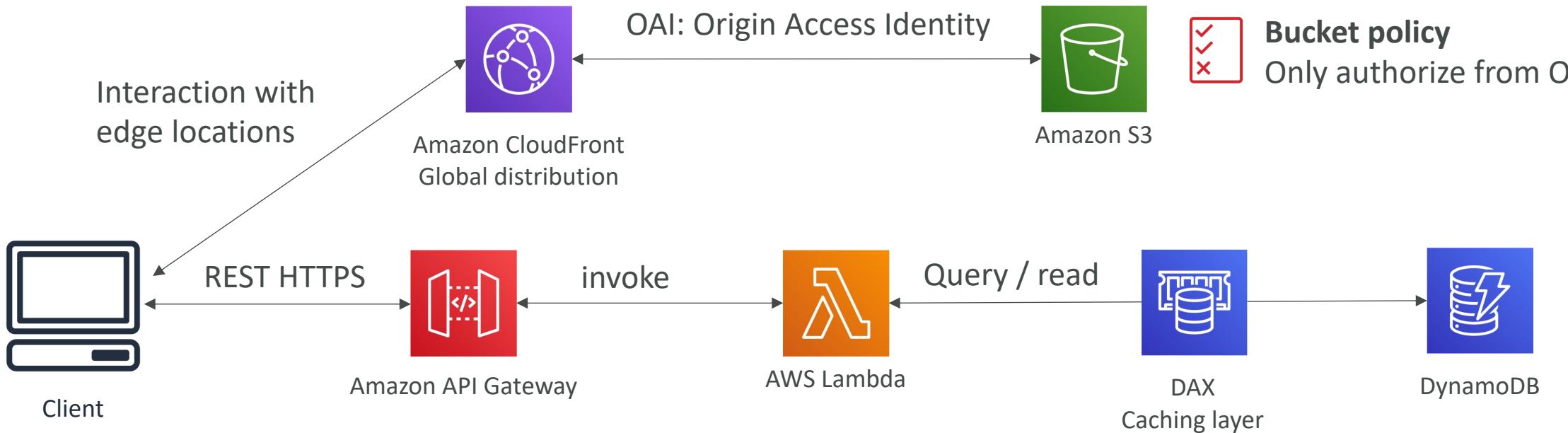
# Serving static content, globally



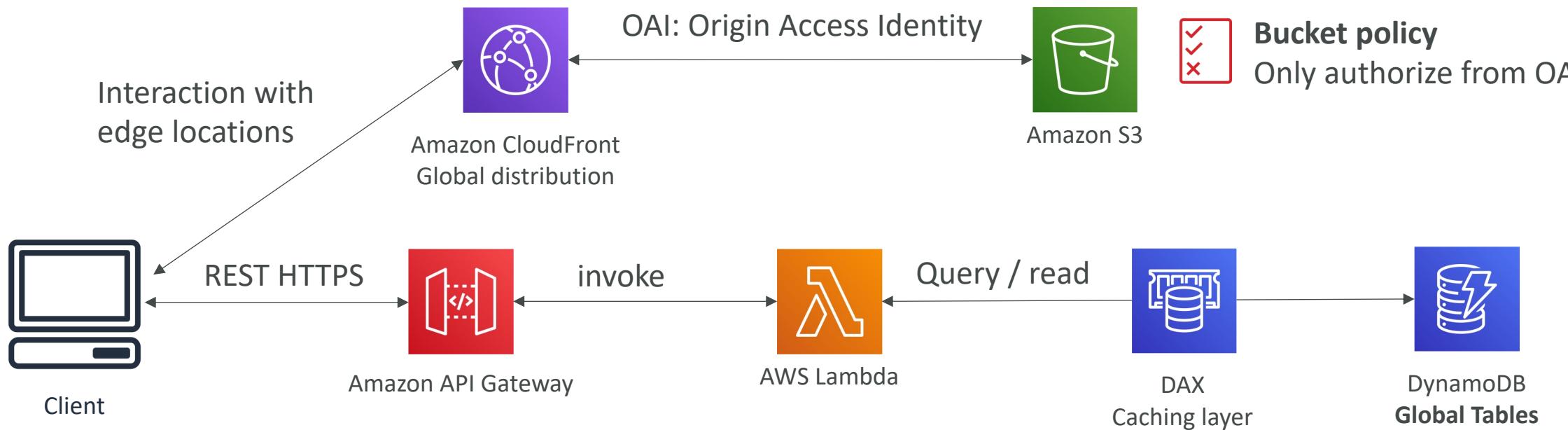
# Serving static content, globally, securely



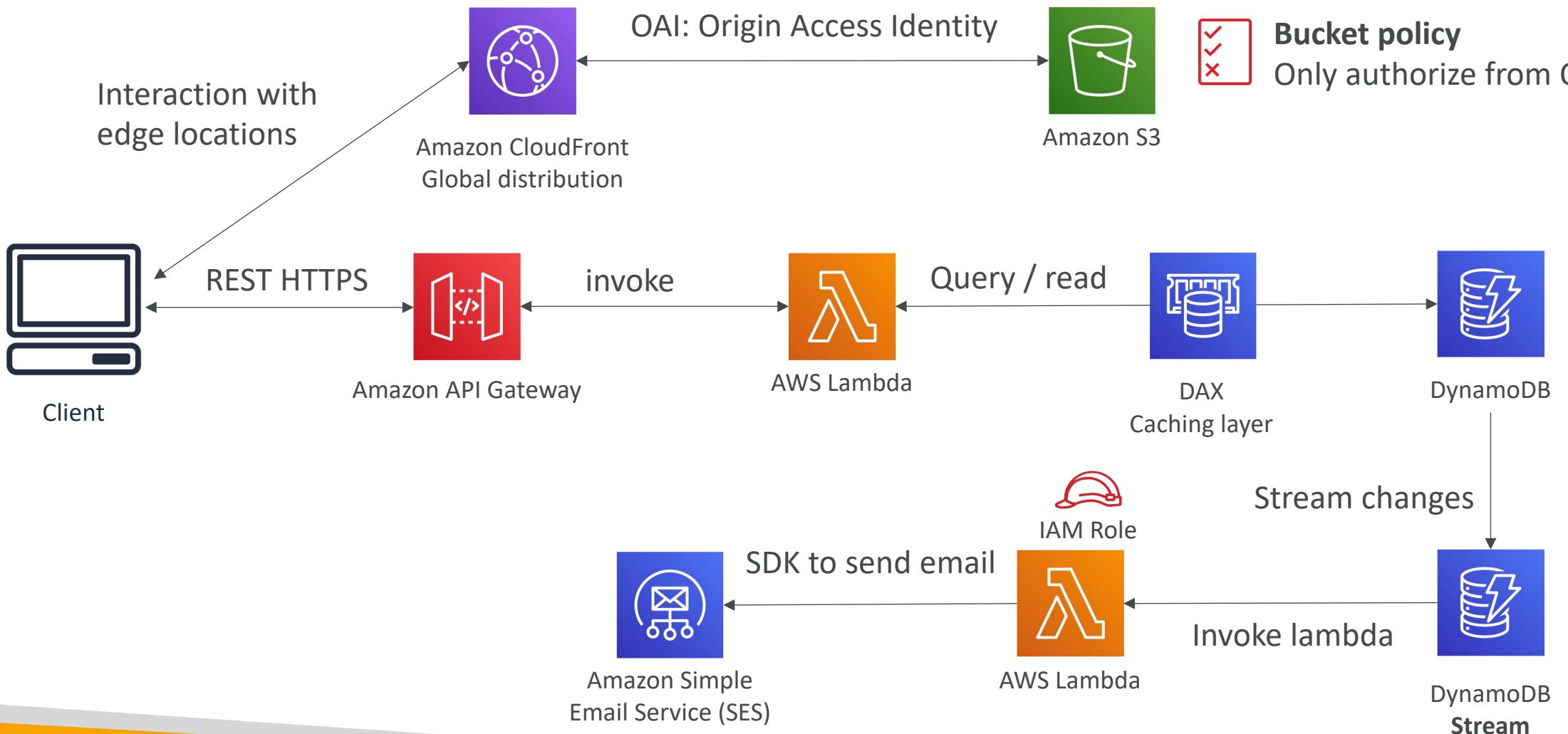
# Adding a public serverless REST API



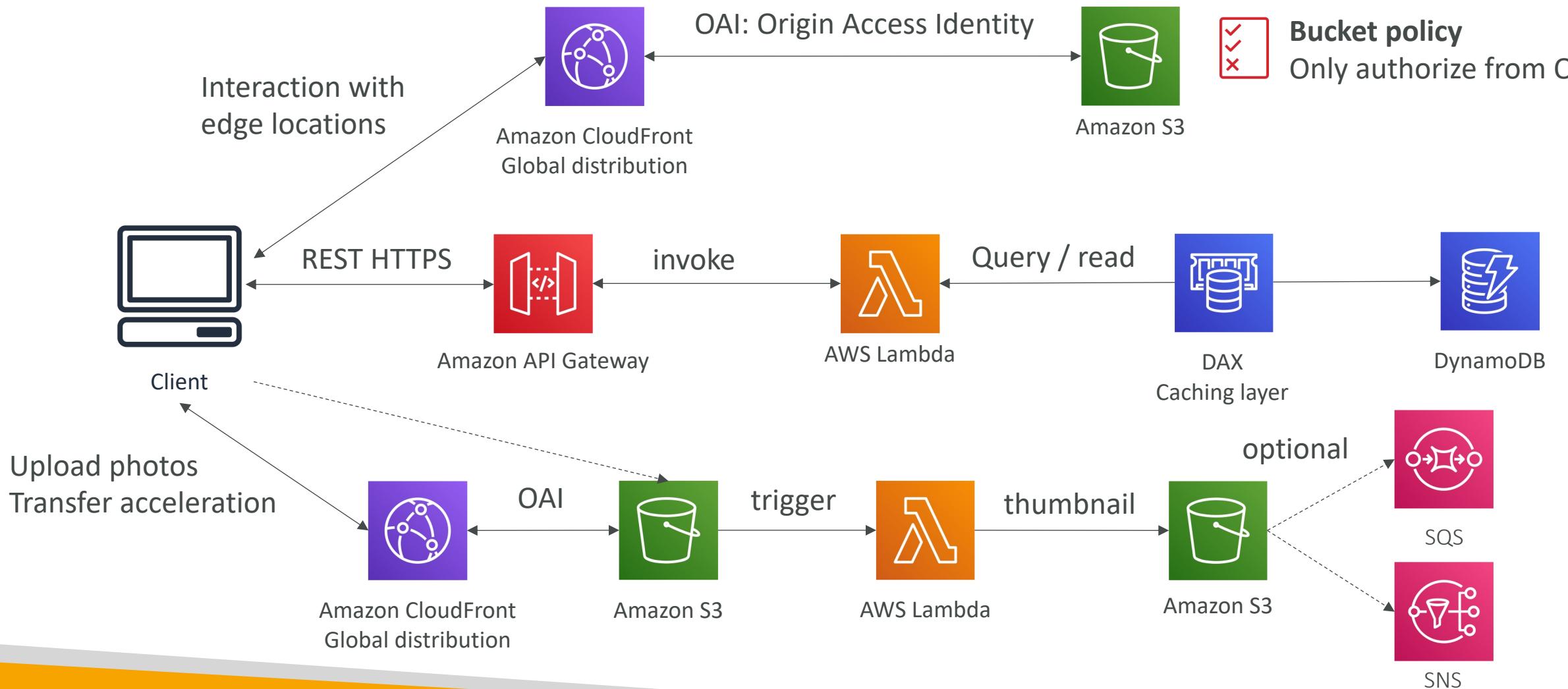
# Leveraging DynamoDB Global Tables



# User Welcome email flow



# Thumbnail Generation flow



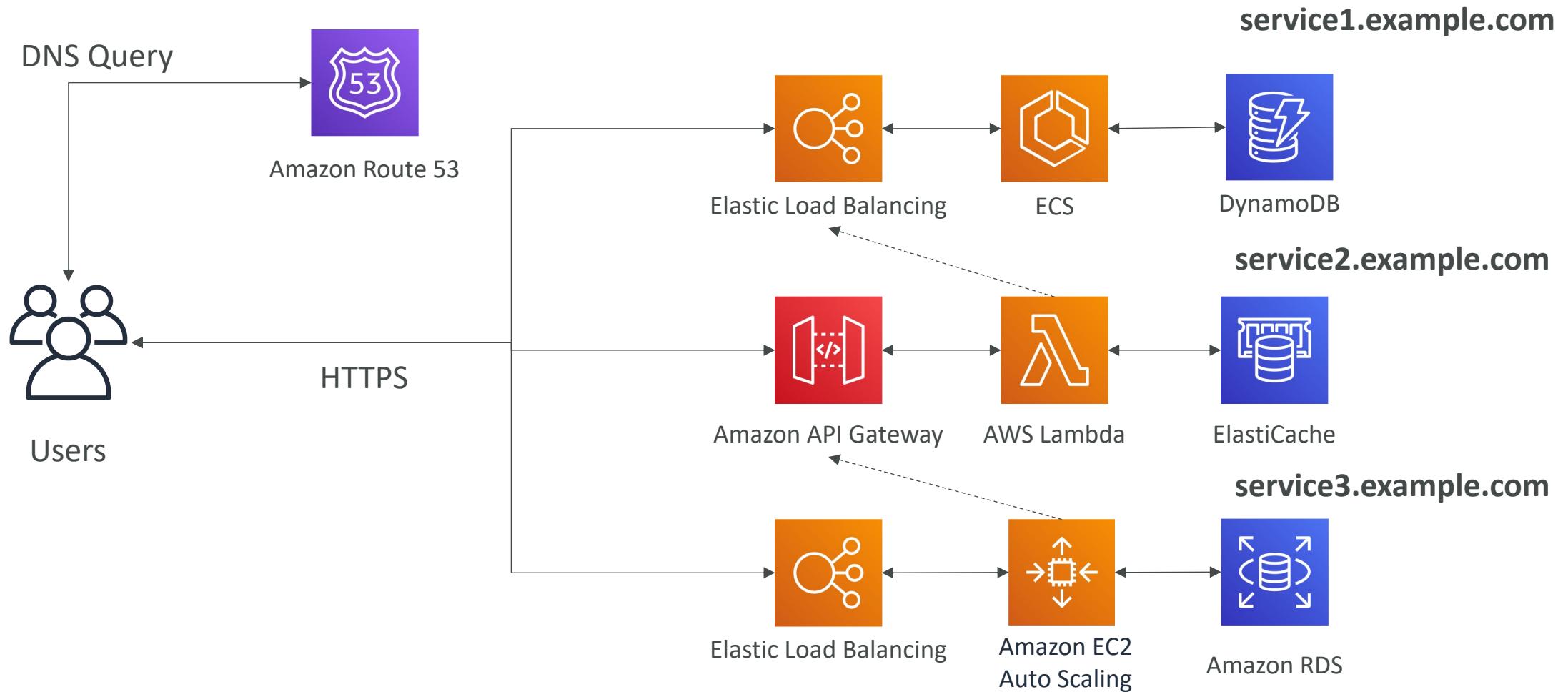
# AWS Hosted Website Summary

- We've seen static content being distributed using CloudFront with S3
- The REST API was serverless, didn't need Cognito because public
- We leveraged a Global DynamoDB table to serve the data globally
  - (we could have used Aurora Global Database)
- We enabled DynamoDB streams to trigger a Lambda function
- The lambda function had an IAM role which could use SES
- SES (Simple Email Service) was used to send emails in a serverless way
- S3 can trigger SQS / SNS / Lambda to notify of events

# Micro Services architecture

- We want to switch to a micro service architecture
  - Many services interact with each other directly using a REST API
  - Each architecture for each micro service may vary in form and shape
- 
- We want a micro-service architecture so we can have a leaner development lifecycle for each service

# Micro Services Environment



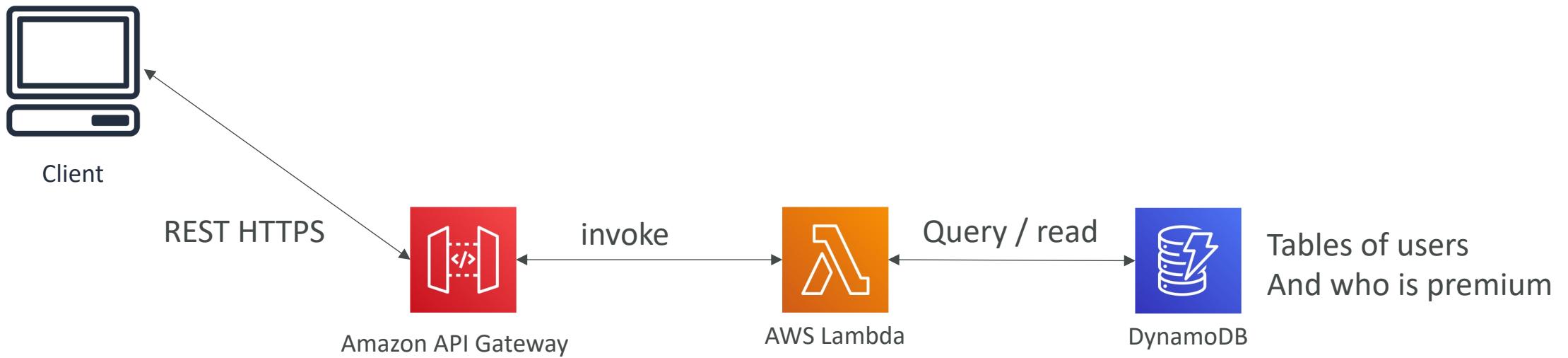
# Discussions on Micro Services

- You are free to design each micro-service the way you want
- Synchronous patterns: API Gateway, Load Balancers
- Asynchronous patterns: SQS, Kinesis, SNS, Lambda triggers (S3)
- Challenges with micro-services:
  - repeated overhead for creating each new microservice,
  - issues with optimizing server density/utilization
  - complexity of running multiple versions of multiple microservices simultaneously
  - proliferation of client-side code requirements to integrate with many separate services.
- Some of the challenges are solved by Serverless patterns:
  - API Gateway, Lambda scale automatically and you pay per usage
  - You can easily clone API, reproduce environments
  - Generated client SDK through Swagger integration for the API Gateway

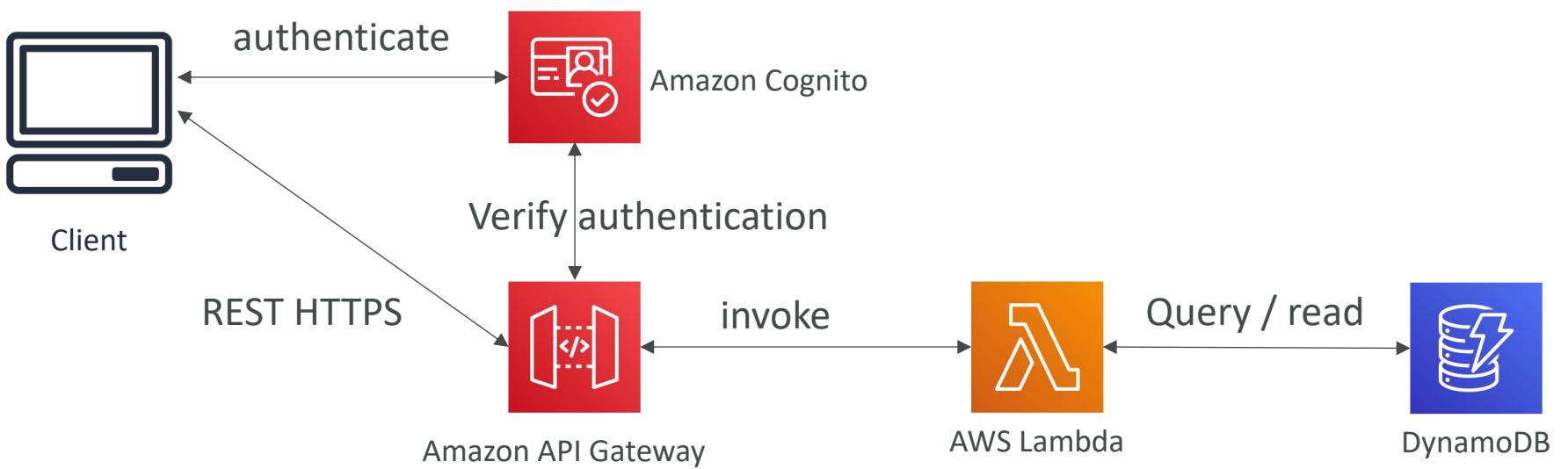
# Distributing paid content

- We sell videos online and users have to paid to buy videos
- Each videos can be bought by many different customers
- We only want to distribute videos to users who are premium users
- We have a database of premium users
- Links we send to premium users should be short lived
- Our application is global
- We want to be fully serverless

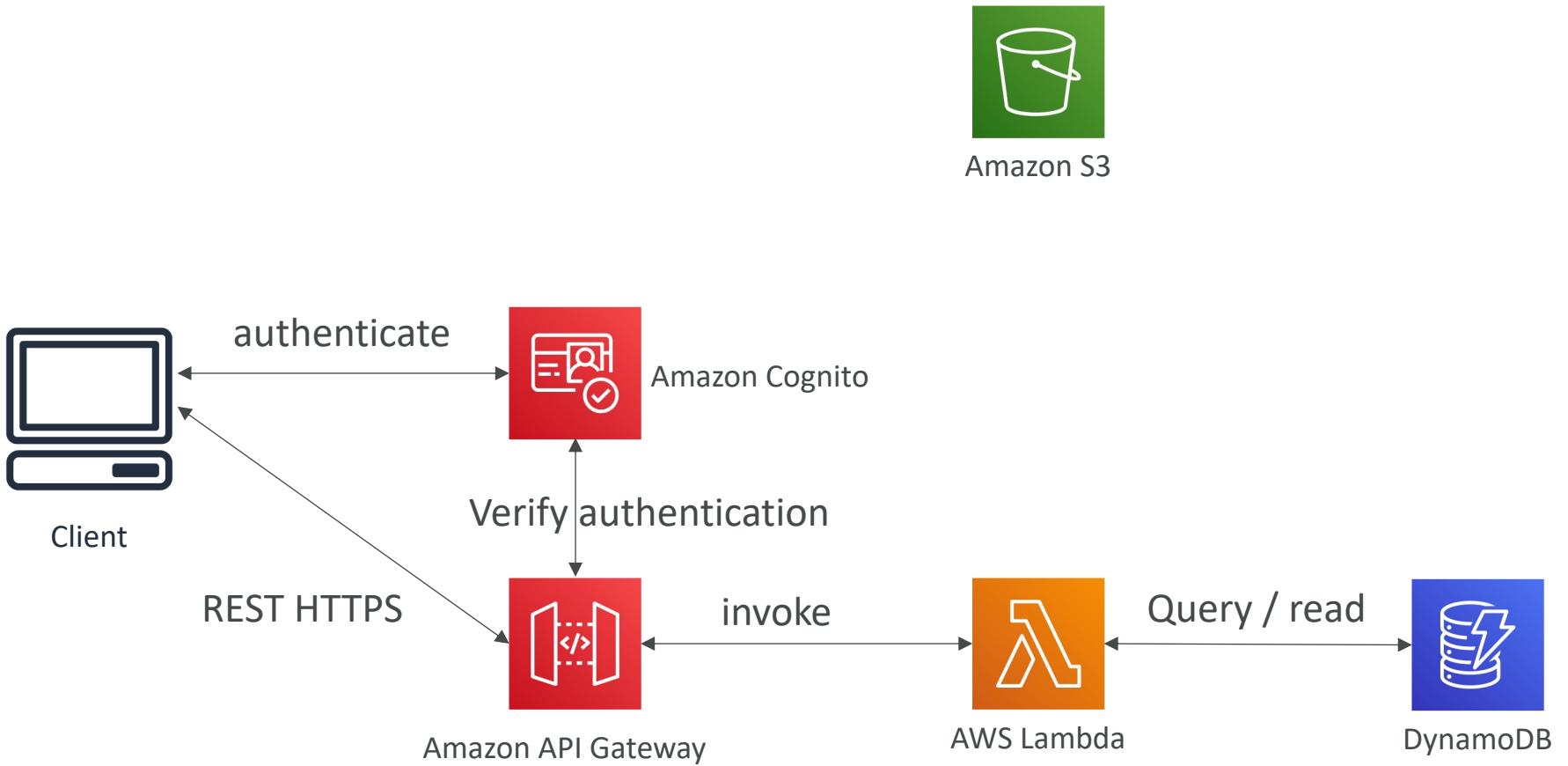
# Start simple, premium user service



# Add authentication

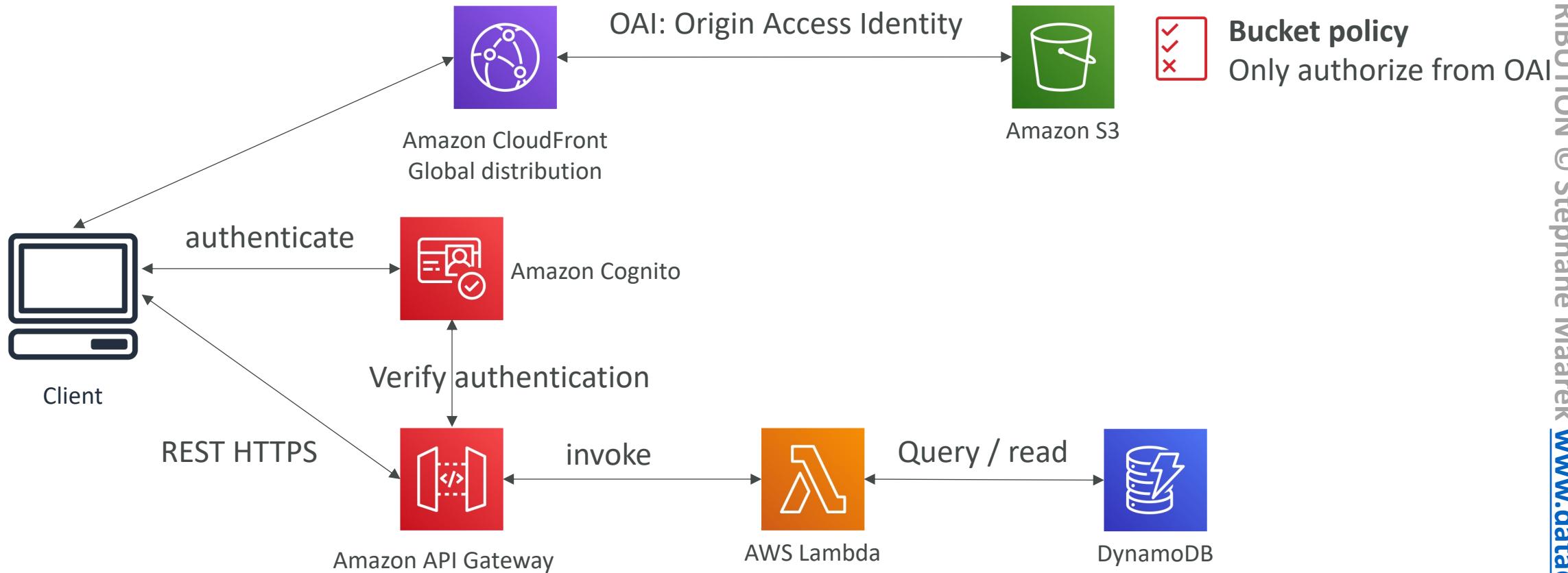


# Add Videos Storage Service

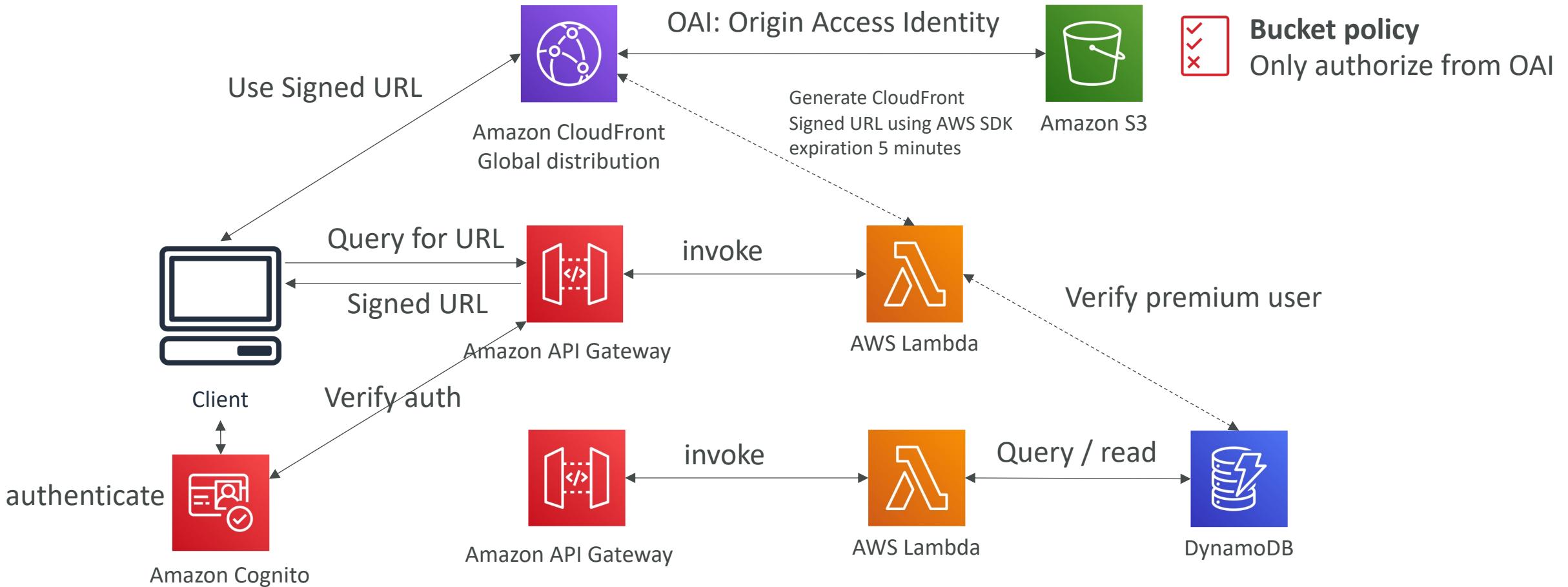


Amazon S3

# Distribute Globally and Secure



# Distribute Content only to premium users



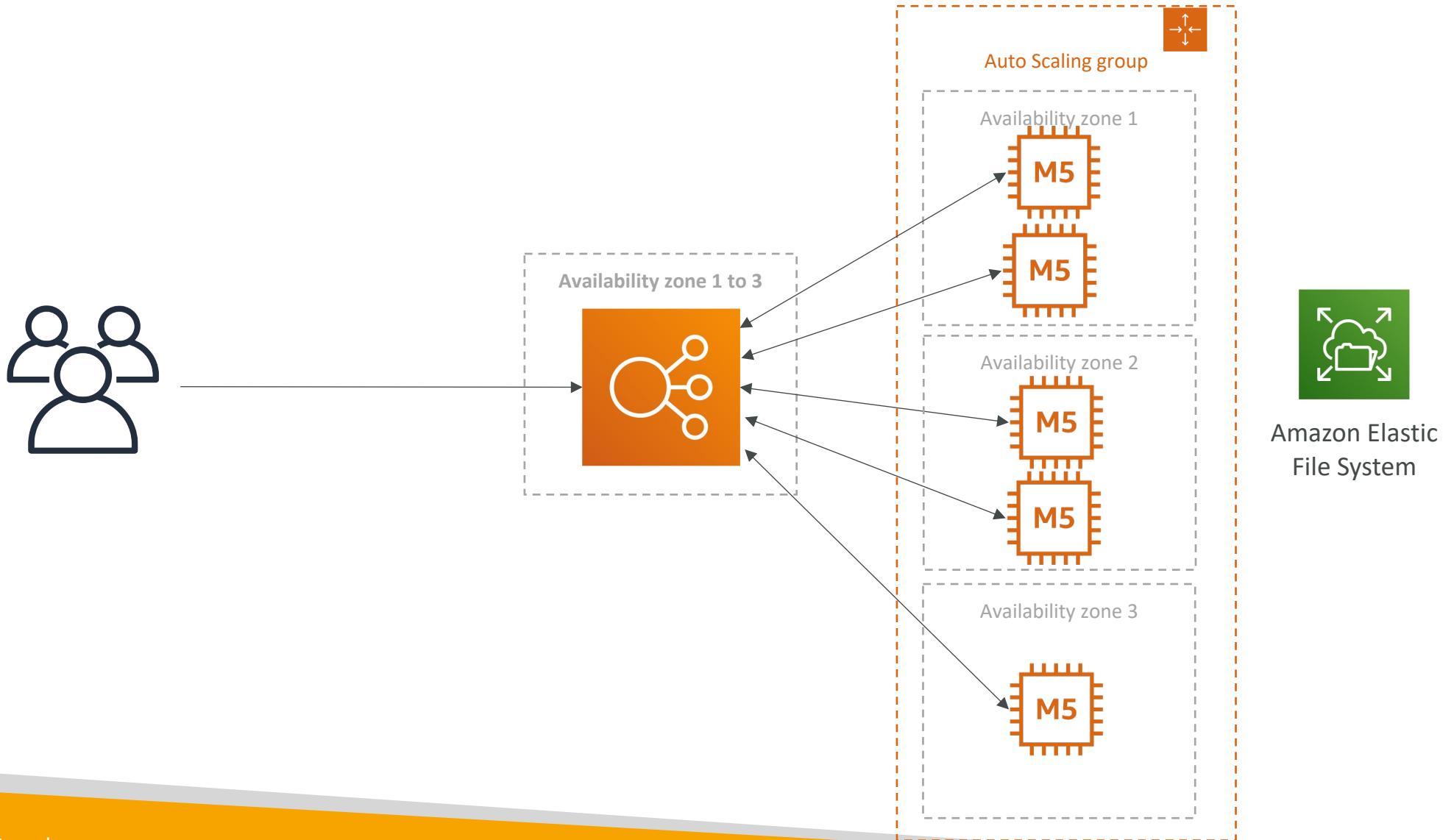
# Premium User Video service

- We have implemented a fully serverless solution:
  - Cognito for authentication
  - DynamoDB for storing users that are premium
  - 2 serverless applications
    - Premium User registration
    - CloudFront Signed URL generator
  - Content is stored in S3 (serverless and scalable)
  - Integrated with CloudFront with OAI for security (users can't bypass)
  - CloudFront can only be used using Signed URLs to prevent unauthorized users
  - What about S3 Signed URL? They're not efficient for global access

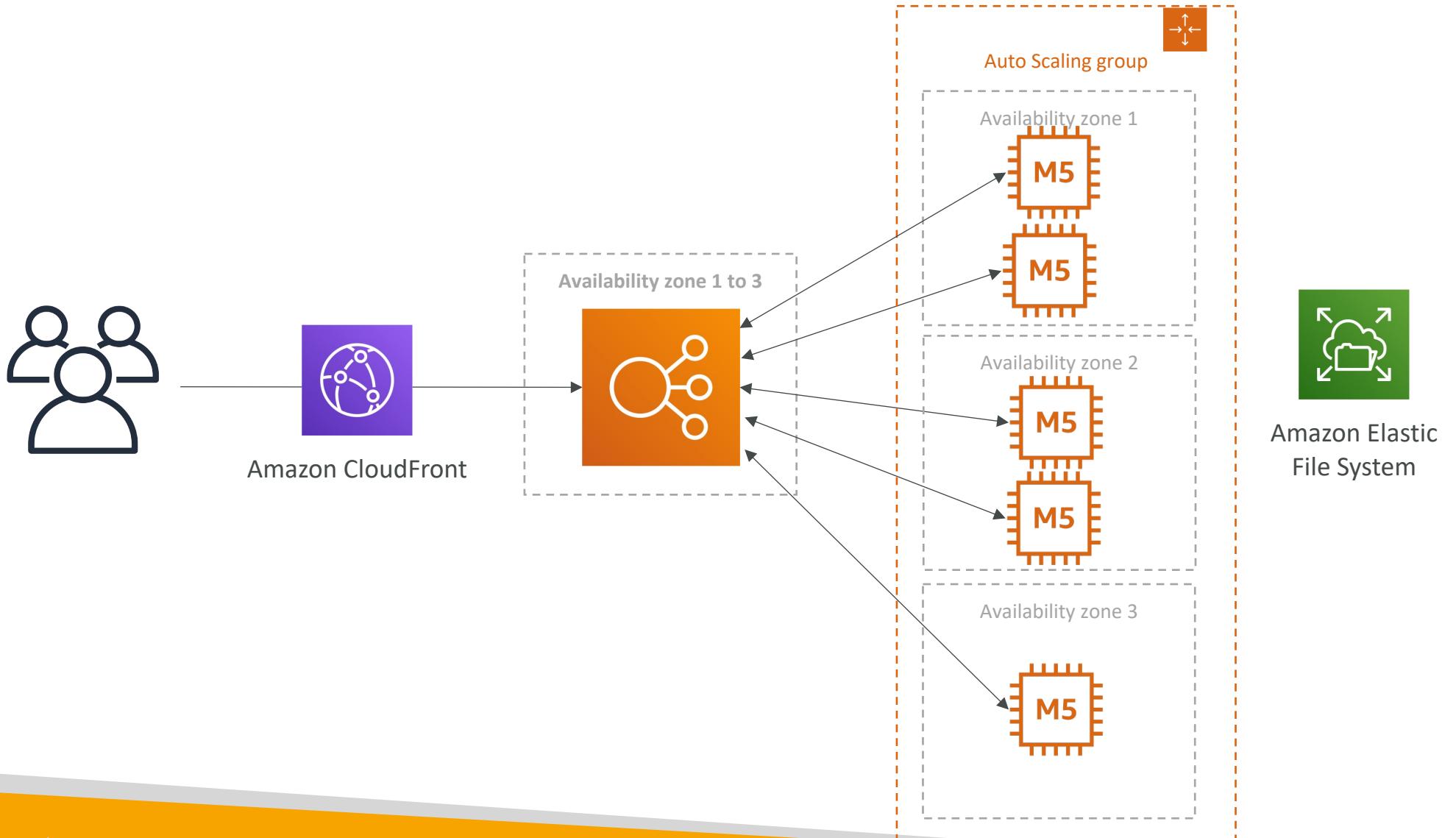
# Software updates offloading

- We have an application running on EC2, that distributes software updates once in a while
- When a new software update is out, we get a lot of request and the content is distributed in mass over the network. It's very costly
- We don't want to change our application, but want to optimize our cost and CPU, how can we do it?

# Our application current state



# Easy way to fix things!



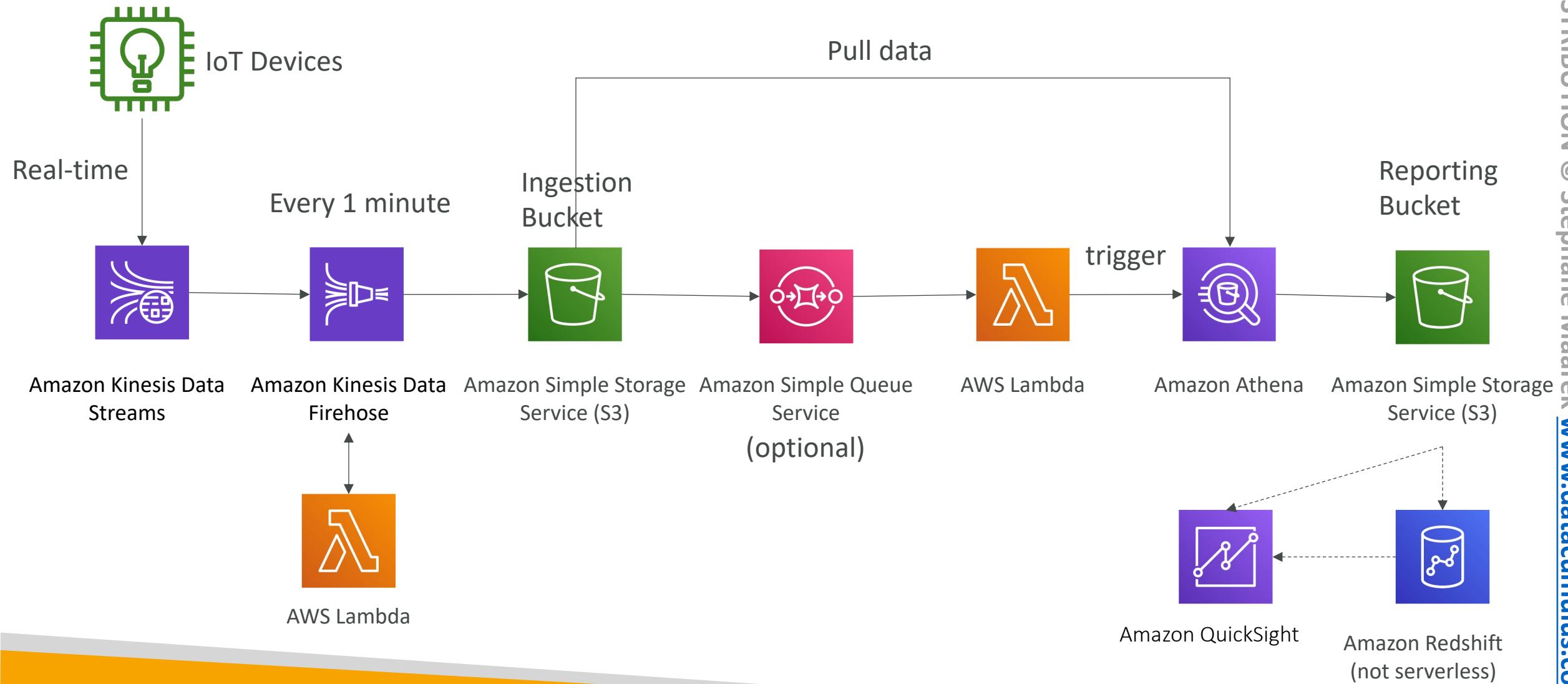
# Why CloudFront?

- No changes to architecture
- Will cache software update files at the edge
- Software update files are not dynamic, they're static (never changing)
- Our EC2 instances aren't serverless
- But CloudFront is, and will scale for us
- Our ASG will not scale as much, and we'll save tremendously in EC2
- We'll also save in availability, network bandwidth cost, etc
- Easy way to make an existing application more scalable and cheaper!

# Big Data Ingestion Pipeline

- We want the ingestion pipeline to be fully serverless
- We want to collect data in real time
- We want to transform the data
- We want to query the transformed data using SQL
- The reports created using the queries should be in S3
- We want to load that data into a warehouse and create dashboards

# Big Data Ingestion Pipeline



# Big Data Ingestion Pipeline discussion

- IoT Core allows you to harvest data from IoT devices
- Kinesis is great for real-time data collection
- Firehose helps with data delivery to S3 in near real-time (1 minute)
- Lambda can help Firehose with data transformations
- Amazon S3 can trigger notifications to SQS
- Lambda can subscribe to SQS (we could have connecter S3 to Lambda)
- Athena is a serverless SQL service and results are stored in S3
- The reporting bucket contains analyzed data and can be used by reporting tool such as AWS QuickSight, Redshift, etc...

# Databases

# Choosing the Right Database

- We have a lot of managed databases on AWS to choose from
- Questions to choose the right database based on your architecture:
  - Read-heavy, write-heavy, or balanced workload? Throughput needs? Will it change, does it need to scale or fluctuate during the day?
  - How much data to store and for how long? Will it grow? Average object size? How are they accessed?
  - Data durability? Source of truth for the data ?
  - Latency requirements? Concurrent users?
  - Data model? How will you query the data? Joins? Structured? Semi-Structured?
  - Strong schema? More flexibility? Reporting? Search? RDBMS / NoSQL?
  - License costs? Switch to Cloud Native DB such as Aurora?

# Database Types



- **RDBMS (= SQL / OLTP)**: RDS, Aurora – great for joins
- **NoSQL database**: DynamoDB (~JSON), ElastiCache (key / value pairs), Neptune (graphs) – no joins, no SQL
- **Object Store**: S3 (for big objects) / Glacier (for backups / archives)
- **Data Warehouse (= SQL Analytics / BI)**: Redshift (OLAP), Athena
- **Search**: ElasticSearch (JSON) – free text, unstructured searches
- **Graphs**: Neptune – displays relationships between data

# RDS Overview



- Managed PostgreSQL / MySQL / Oracle / SQL Server
  - Must provision an EC2 instance & EBS Volume type and size
  - Support for Read Replicas and Multi AZ
  - Security through IAM, Security Groups, KMS , SSL in transit
  - Backup / Snapshot / Point in time restore feature
  - Managed and Scheduled maintenance
  - Monitoring through CloudWatch
- 
- **Use case:** Store relational datasets (RDBMS / OLTP), perform SQL queries, transactional inserts / update / delete is available

# RDS for Solutions Architect

- **Operations:** small downtime when failover happens, when maintenance happens, scaling in read replicas / ec2 instance / restore EBS implies manual intervention, application changes
- **Security:** AWS responsible for OS security, we are responsible for setting up KMS, security groups, IAM policies, authorizing users in DB, using SSL
- **Reliability:** Multi AZ feature, failover in case of failures
- **Performance:** depends on EC2 instance type, EBS volume type, ability to add Read Replicas. Storage auto-scaling & manual scaling of instances
- **Cost:** Pay per hour based on provisioned EC2 and EBS

# Aurora Overview



- Compatible API for PostgreSQL / MySQL
- Data is held in 6 replicas, across 3 AZ
- Auto healing capability
- Multi AZ, Auto Scaling Read Replicas
- Read Replicas can be Global
- Aurora database can be Global for DR or latency purposes
- Auto scaling of storage from 10GB to 128 TB
- Define EC2 instance type for aurora instances
- Same security / monitoring / maintenance features as RDS
- Aurora Serverless – for unpredictable / intermittent workloads
- Aurora Multi-Master – for continuous writes failover
  
- **Use case:** same as RDS, but with less maintenance / more flexibility / more performance

# Aurora for Solutions Architect

- **Operations:** less operations, auto scaling storage
- **Security:** AWS responsible for OS security, we are responsible for setting up KMS, security groups, IAM policies, authorizing users in DB, using SSL
- **Reliability:** Multi AZ, highly available, possibly more than RDS, Aurora Serverless option, Aurora Multi-Master option
- **Performance:** 5x performance (according to AWS) due to architectural optimizations. Up to 15 Read Replicas (only 5 for RDS)
- **Cost:** Pay per hour based on EC2 and storage usage. Possibly lower costs compared to Enterprise grade databases such as Oracle

# ElastiCache Overview



- Managed Redis / Memcached (similar offering as RDS, but for caches)
- In-memory data store, sub-millisecond latency
- Must provision an EC2 instance type
- Support for Clustering (Redis) and Multi AZ, Read Replicas (sharding)
- Security through IAM, Security Groups, KMS, Redis Auth
- Backup / Snapshot / Point in time restore feature
- Managed and Scheduled maintenance
- Monitoring through CloudWatch
- **Use Case:** Key/Value store, Frequent reads, less writes, cache results for DB queries, store session data for websites, cannot use SQL.

# ElastiCache for Solutions Architect

- **Operations:** same as RDS
- **Security:** AWS responsible for OS security, we are responsible for setting up KMS, security groups, IAM policies, users (Redis Auth), using SSL
- **Reliability:** Clustering, Multi AZ
- **Performance:** Sub-millisecond performance, in memory, read replicas for sharding, very popular cache option
- **Cost:** Pay per hour based on EC2 and storage usage

# DynamoDB Overview



- AWS proprietary technology, managed NoSQL database
- Serverless, provisioned capacity, auto scaling, on demand capacity (Nov 2018)
- Can replace ElastiCache as a key/value store (storing session data for example)
- Highly Available, Multi AZ by default, Read and Writes are decoupled, DAX for read cache
- Reads can be eventually consistent or strongly consistent
- Security, authentication and authorization is done through IAM
- DynamoDB Streams to integrate with AWS Lambda
- Backup / Restore feature, Global Table feature
- Monitoring through CloudWatch
- Can only query on primary key, sort key, or indexes
- **Use Case:** Serverless applications development (small documents 100s KB), distributed serverless cache, doesn't have SQL query language available, has transactions capability from Nov 2018

# DynamoDB for Solutions Architect

- **Operations:** no operations needed, auto scaling capability, serverless
- **Security:** full security through IAM policies, KMS encryption, SSL in flight
- **Reliability:** Multi AZ, Backups
- **Performance:** single digit millisecond performance, DAX for caching reads, performance doesn't degrade if your application scales
- **Cost:** Pay per provisioned capacity and storage usage (no need to guess in advance any capacity – can use auto scaling)

# S3 Overview



- S3 is a... key / value store for objects
- Great for big objects, not so great for small objects
- Serverless, scales infinitely, max object size is 5 TB
- Strong consistency
- Tiers: S3 Standard, S3 IA, S3 One Zone IA, Glacier for backups
- Features: Versioning, Encryption, Cross Region Replication, etc...
- Security: IAM, Bucket Policies, ACL
- Encryption: SSE-S3, SSE-KMS, SSE-C, client side encryption, SSL in transit
- **Use Case:** static files, key value store for big files, website hosting

# S3 for Solutions Architect

- **Operations:** no operations needed
- **Security:** IAM, Bucket Policies, ACL, Encryption (Server/Client), SSL
- **Reliability:** 99.99999999% durability / 99.99% availability, Multi AZ, CRR
- **Performance:** scales to thousands of read / writes per second, transfer acceleration / multi-part for big files
- **Cost:** pay per storage usage, network cost, requests number

# Athena Overview



- Fully Serverless database with SQL capabilities
  - Used to query data in S3
  - Pay per query
  - Output results back to S3
  - Secured through IAM
- 
- **Use Case:** one time SQL queries, serverless queries on S3, log analytics

# Athena for Solutions Architect

- **Operations:** no operations needed, serverless
- **Security:** IAM + S3 security
- **Reliability:** managed service, uses Presto engine, highly available
- **Performance:** queries scale based on data size
- **Cost:** pay per query / per TB of data scanned, serverless

# Redshift Overview



- Redshift is based on PostgreSQL, but it's not used for OLTP
- It's OLAP – online analytical processing (analytics and data warehousing)
- 10x better performance than other data warehouses, scale to PBs of data
- Columnar storage of data (instead of row based)
- Massively Parallel Query Execution (MPP)
- Pay as you go based on the instances provisioned
- Has a SQL interface for performing the queries
- BI tools such as AWS Quicksight or Tableau integrate with it

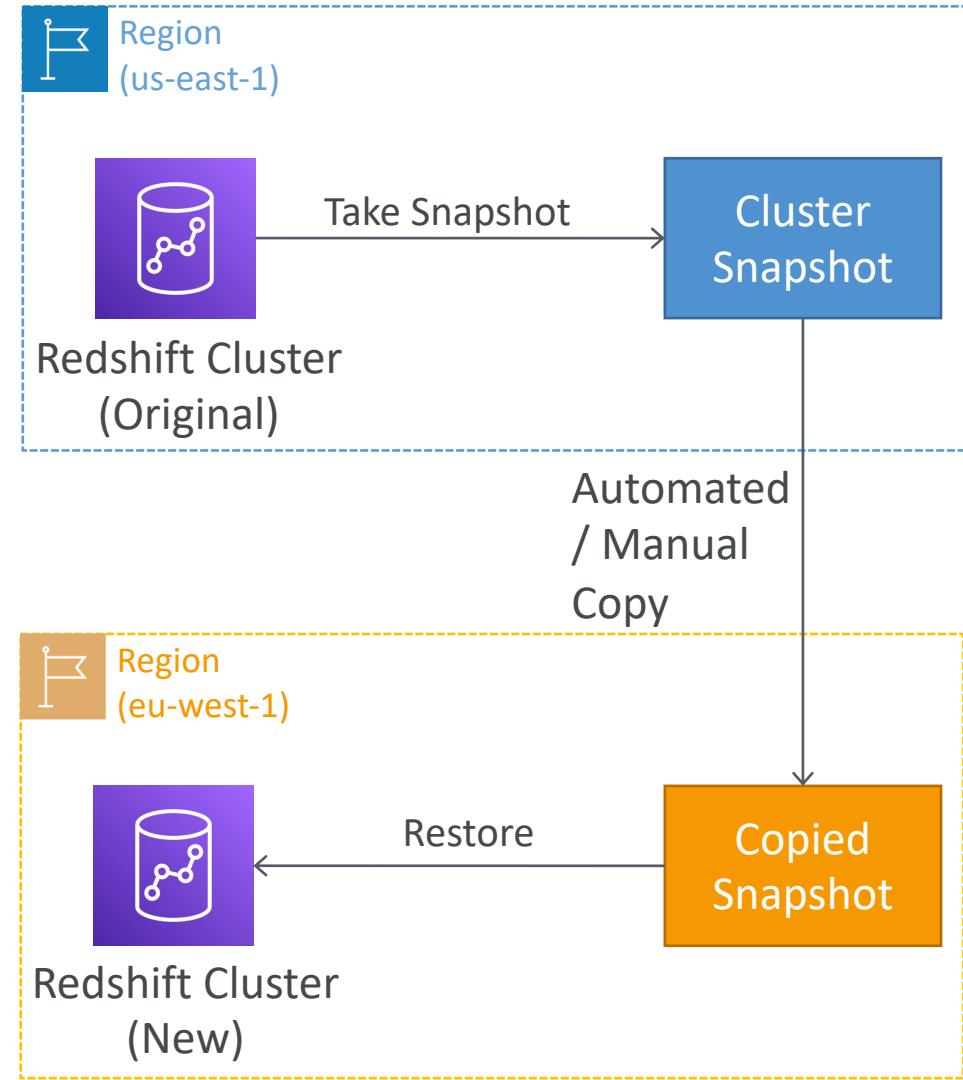
# Redshift Continued...



- Data is loaded from S3, DynamoDB, DMS, other DBs...
- From 1 node to 128 nodes, up to 128 TB of space per node
- Leader node: for query planning, results aggregation
- Compute node: for performing the queries, send results to leader
- Redshift Spectrum: perform queries directly against S3 (no need to load)
- Backup & Restore, Security VPC / IAM / KMS, Monitoring
- Redshift Enhanced VPC Routing: COPY / UNLOAD goes through VPC

# Redshift – Snapshots & DR

- Redshift has no “Multi-AZ” mode
- Snapshots are point-in-time backups of a cluster, stored internally in S3
- Snapshots are incremental (only what has changed is saved)
- You can restore a snapshot into a new cluster
- Automated: every 8 hours, every 5 GB, or on a schedule. Set retention
- Manual: snapshot is retained until you delete it
- You can configure Amazon Redshift to automatically copy snapshots (automated or manual) of a cluster to another AWS Region

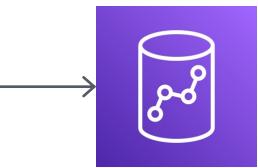


# Loading data into Redshift

## Amazon Kinesis Data Firehose

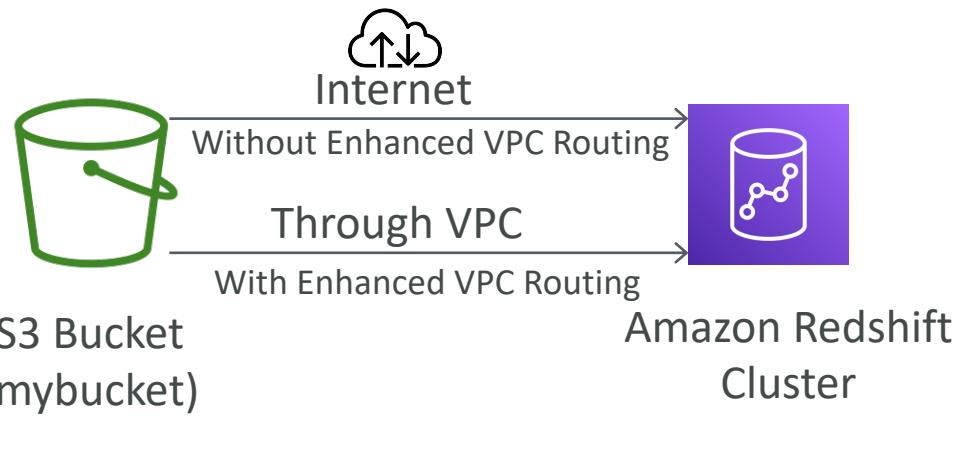


Amazon Kinesis Data Firehose



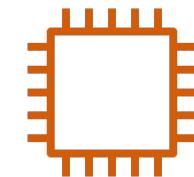
Amazon Redshift Cluster  
(through S3 copy)

## S3 using COPY command

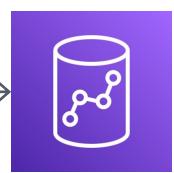


```
copy customer
from 's3://mybucket/mydata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

## EC2 Instance JDBC driver



EC2 Instance

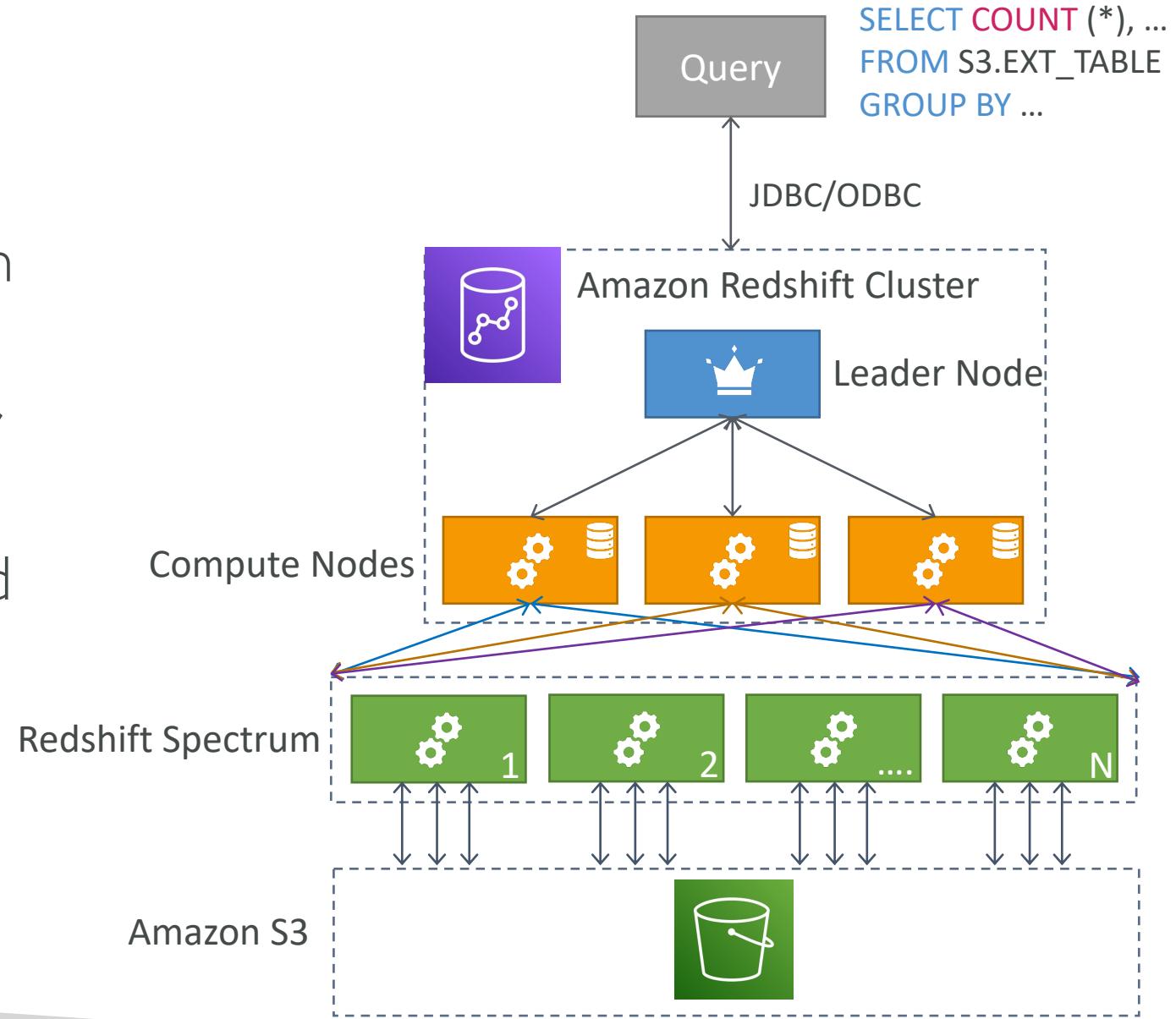


Amazon Redshift Cluster

*Better to write  
Data in batches*

# Redshift Spectrum

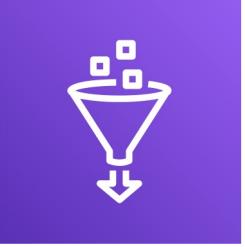
- Query data that is already in S3 without loading it
- Must have a Redshift cluster available to start the query
- The query is then submitted to thousands of Redshift Spectrum nodes



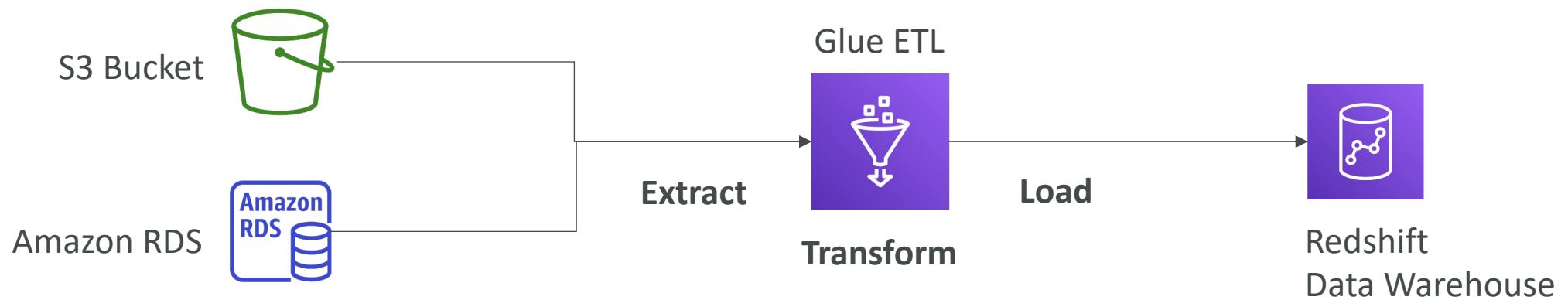
# Redshift for Solutions Architect

- Operations: like RDS
- Security: IAM, VPC, KMS, SSL (like RDS)
- Reliability: auto healing features, cross-region snapshot copy
- Performance: 10x performance vs other data warehousing, compression
- Cost: pay per node provisioned, 1/10<sup>th</sup> of the cost vs other warehouses
- vs Athena: faster queries / joins / aggregations thanks to indexes
- Remember: Redshift = Analytics / BI / Data Warehouse

# AWS Glue



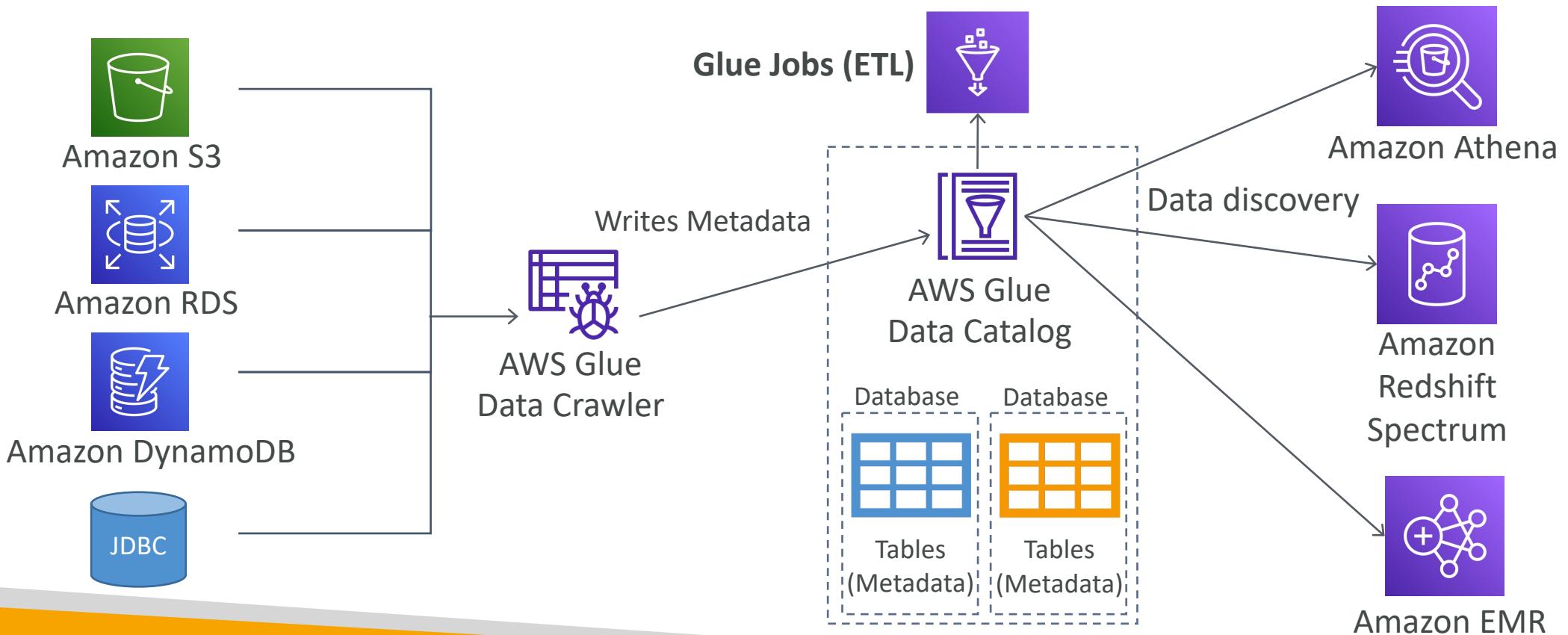
- Managed **extract, transform, and load (ETL)** service
- Useful to prepare and transform data for analytics
- Fully **serverless** service



# Glue Data Catalog



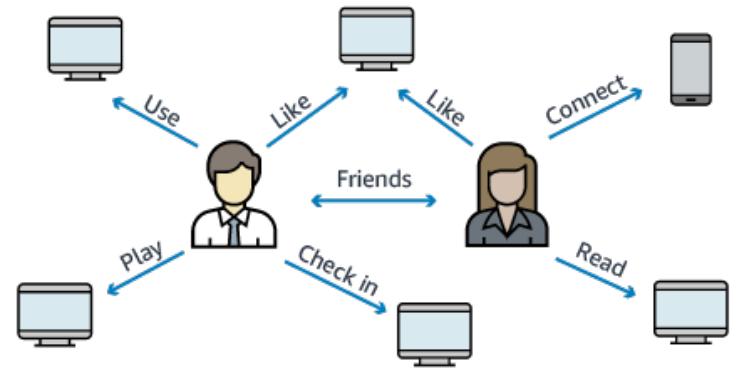
- Glue Data Catalog: catalog of datasets



# Neptune



- Fully managed graph database
- When do we use Graphs?
  - High relationship data
  - Social Networking: Users friends with Users, replied to comment on post of user and likes other comments.
  - Knowledge graphs (Wikipedia)
- Highly available across 3 AZ, with up to 15 read replicas
- Point-in-time recovery, continuous backup to Amazon S3
- Support for KMS encryption at rest + HTTPS



# Neptune for Solutions Architect

- **Operations:** similar to RDS
- **Security:** IAM, VPC, KMS, SSL (similar to RDS) + IAM Authentication
- **Reliability:** Multi-AZ, clustering
- **Performance:** best suited for graphs, clustering to improve performance
- **Cost:** pay per node provisioned (similar to RDS)
- **Remember:** Neptune = Graphs



# ElasticSearch

- Example: In DynamoDB, you can only find by primary key or indexes.
- With ElasticSearch, you can **search any field**, even partially matches
- It's common to use ElasticSearch as a complement to another database
- ElasticSearch also has some usage for Big Data applications
- You can provision a cluster of instances
- Built-in integrations: Amazon Kinesis Data Firehose, AWS IoT, and Amazon CloudWatch Logs for data ingestion
- Security through Cognito & IAM, KMS encryption, SSL & VPC
- Comes with Kibana (visualization) & Logstash (log ingestion) – ELK stack

# ElasticSearch for Solutions Architect

- **Operations:** similar to RDS
- **Security:** Cognito, IAM, VPC, KMS, SSL
- **Reliability:** Multi-AZ, clustering
- **Performance:** based on ElasticSearch project (open source), petabyte scale
- **Cost:** pay per node provisioned (similar to RDS)
- **Remember:** ElasticSearch = Search / Indexing

# AWS Monitoring, Audit and Performance

CloudWatch, CloudTrail & AWS Config

# AWS CloudWatch Metrics



- CloudWatch provides metrics for every services in AWS
- **Metric** is a variable to monitor (CPUUtilization, NetworkIn...)
- Metrics belong to **namespaces**
- Dimension is an attribute of a metric (instance id, environment, etc....).
- Up to 10 dimensions per metric
- Metrics have **timestamps**
- Can create CloudWatch dashboards of metrics

# EC2 Detailed monitoring

- EC2 instance metrics have metrics “every 5 minutes”
- With detailed monitoring (for a cost), you get data “every 1 minute”
- Use detailed monitoring if you want to scale faster for your ASG!
- The AWS Free Tier allows us to have 10 detailed monitoring metrics
- Note: EC2 Memory usage is by default not pushed (must be pushed from inside the instance as a custom metric)

# CloudWatch Custom Metrics

- Possibility to define and send your own custom metrics to CloudWatch
- Example: memory (RAM) usage, disk space, number of logged in users ...
- Use API call `PutMetricData`
- Ability to use dimensions (attributes) to segment metrics
  - `Instance.id`
  - `Environment.name`
- Metric resolution (`StorageResolution` API parameter – two possible value):
  - Standard: 1 minute (60 seconds)
  - High Resolution: 1/5/10/30 second(s) – Higher cost
- **Important:** Accepts metric data points two weeks in the past and two hours in the future (make sure to configure your EC2 instance time correctly)

# CloudWatch Dashboards

- Great way to setup custom dashboards for quick access to key metrics and alarms
- Dashboards are global
- Dashboards can include graphs from different AWS accounts and regions
- You can change the time zone & time range of the dashboards
- You can setup automatic refresh (10s, 1m, 2m, 5m, 15m)
- Dashboards can be shared with people who don't have an AWS account (public, email address, 3<sup>rd</sup> party SSO provider through Amazon Cognito)
- Pricing:
  - 3 dashboards (up to 50 metrics) for free
  - \$3/dashboard/month afterwards



# CloudWatch Logs

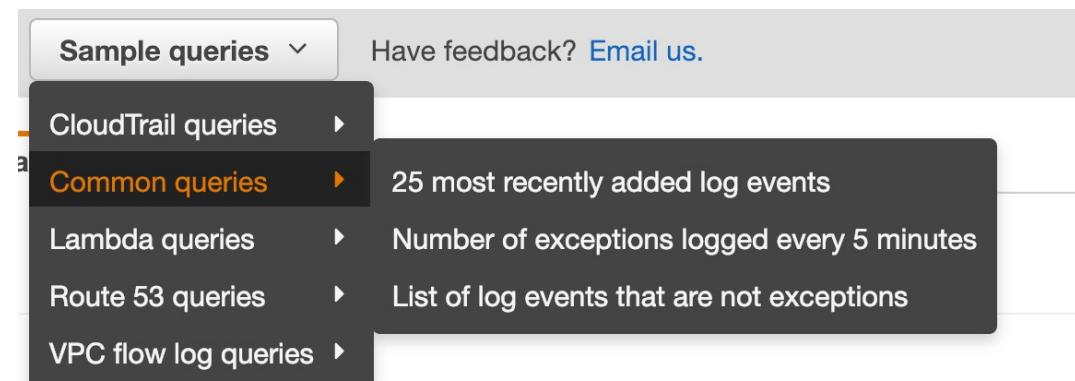
- **Log groups:** arbitrary name, usually representing an application
- **Log stream:** instances within application / log files / containers
- Can define log expiration policies (never expire, 30 days, etc..)
- **CloudWatch Logs can send logs to:**
  - Amazon S3 (exports)
  - Kinesis Data Streams
  - Kinesis Data Firehose
  - AWS Lambda
  - ElasticSearch

# CloudWatch Logs - Sources

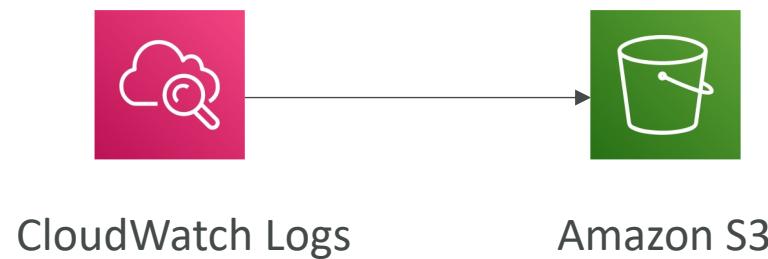
- SDK, CloudWatch Logs Agent, CloudWatch Unified Agent
- Elastic Beanstalk: collection of logs from application
- ECS: collection from containers
- AWS Lambda: collection from function logs
- VPC Flow Logs: VPC specific logs
- API Gateway
- CloudTrail based on filter
- Route53: Log DNS queries

# CloudWatch Logs Metric Filter & Insights

- CloudWatch Logs can use filter expressions
  - For example, find a specific IP inside of a log
  - Or count occurrences of “ERROR” in your logs
- Metric filters can be used to trigger CloudWatch alarms
- CloudWatch Logs Insights can be used to query logs and add queries to CloudWatch Dashboards

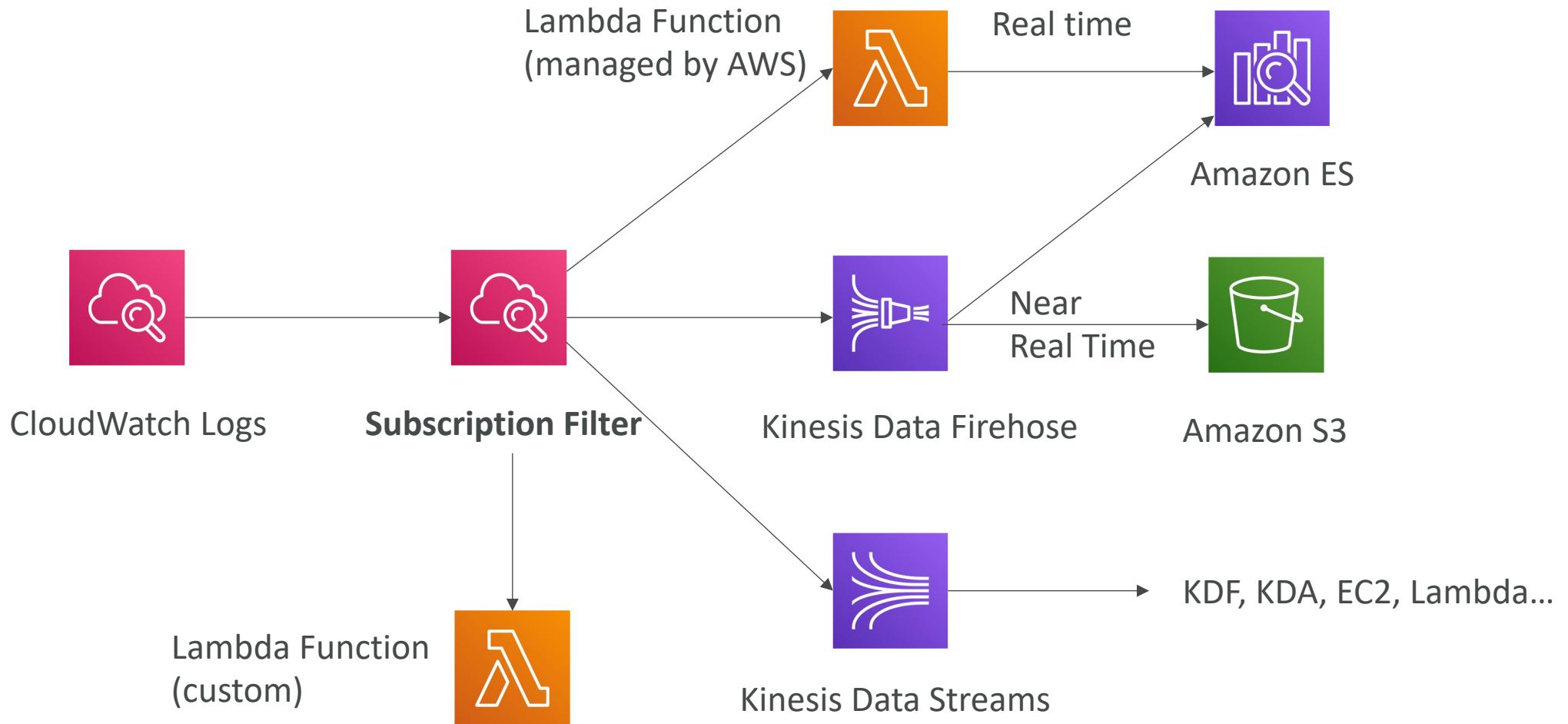


# CloudWatch Logs – S3 Export

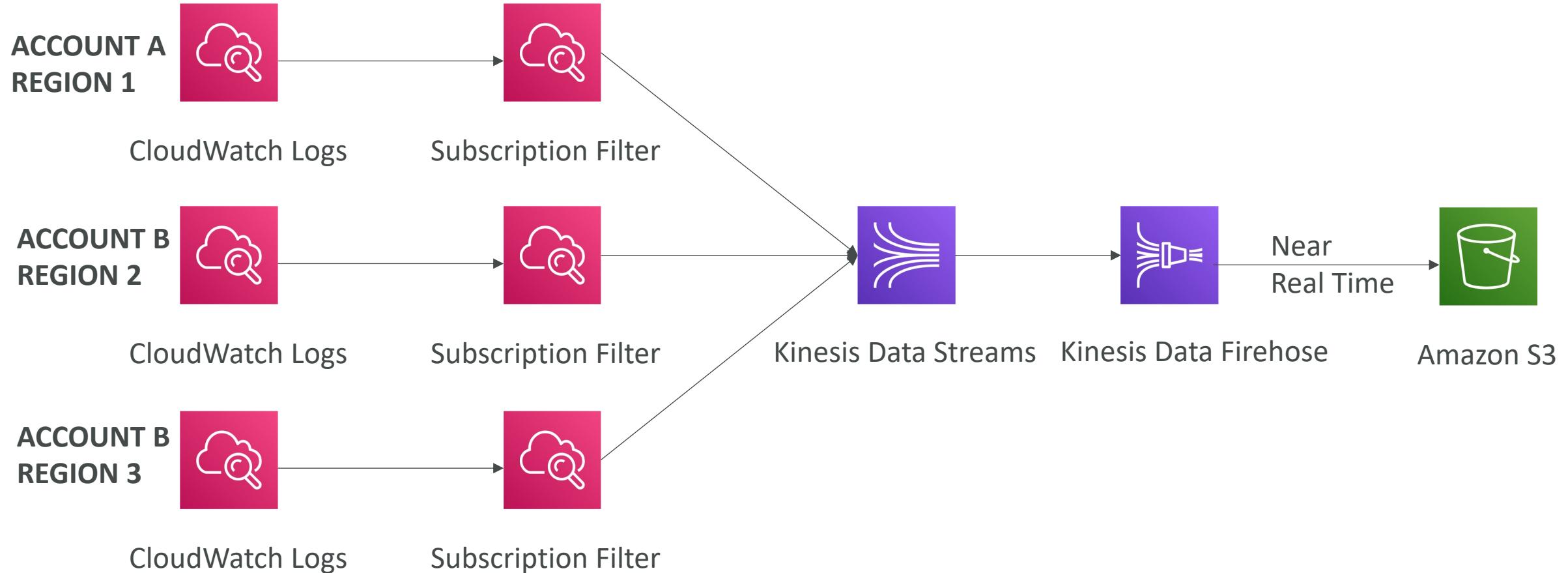


- Log data can take up to 12 hours to become available for export
- The API call is `CreateExportTask`
- Not near-real time or real-time... use Logs Subscriptions instead

# CloudWatch Logs Subscriptions

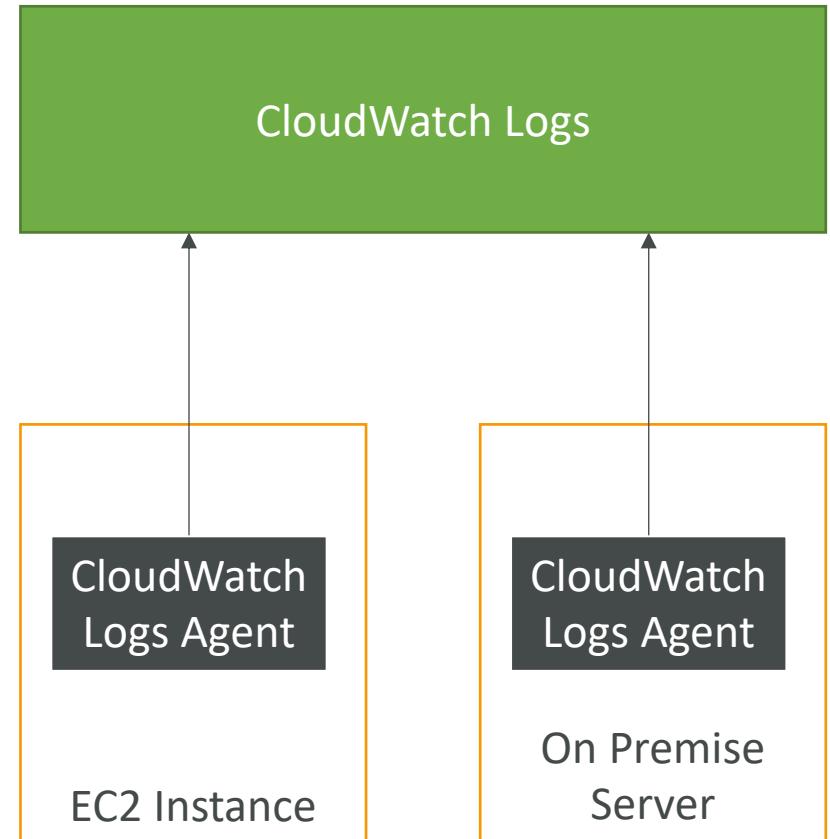


# CloudWatch Logs Aggregation Multi-Account & Multi Region



# CloudWatch Logs for EC2

- By default, no logs from your EC2 machine will go to CloudWatch
- You need to run a CloudWatch agent on EC2 to push the log files you want
- Make sure IAM permissions are correct
- The CloudWatch log agent can be setup on-premises too



# CloudWatch Logs Agent & Unified Agent

- For virtual servers (EC2 instances, on-premises servers...)
- **CloudWatch Logs Agent**
  - Old version of the agent
  - Can only send to CloudWatch Logs
- **CloudWatch Unified Agent**
  - Collect additional system-level metrics such as RAM, processes, etc...
  - Collect logs to send to CloudWatch Logs
  - Centralized configuration using SSM Parameter Store

# CloudWatch Unified Agent – Metrics

- Collected directly on your Linux server / EC2 instance
- CPU (active, guest, idle, system, user, steal)
- Disk metrics (free, used, total), Disk IO (writes, reads, bytes, iops)
- RAM (free, inactive, used, total, cached)
- Netstat (number of TCP and UDP connections, net packets, bytes)
- Processes (total, dead, bloqued, idle, running, sleep)
- Swap Space (free, used, used %)
- Reminder: out-of-the box metrics for EC2 – disk, CPU, network (high level)

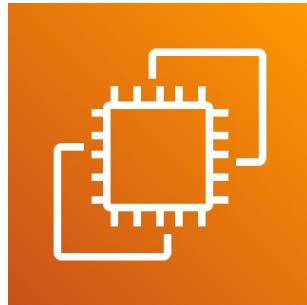
# CloudWatch Alarms



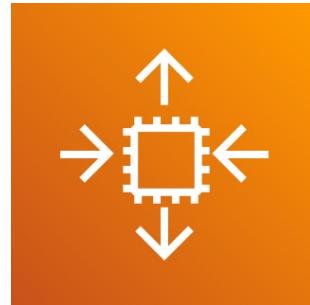
- Alarms are used to trigger notifications for any metric
- Various options (sampling, %, max, min, etc...)
- Alarm States:
  - OK
  - INSUFFICIENT\_DATA
  - ALARM
- Period:
  - Length of time in seconds to evaluate the metric
  - High resolution custom metrics: 10 sec, 30 sec or multiples of 60 sec

# CloudWatch Alarm Targets

- Stop, Terminate, Reboot, or Recover an EC2 Instance
- Trigger Auto Scaling Action
- Send notification to SNS (from which you can do pretty much anything)



Amazon EC2



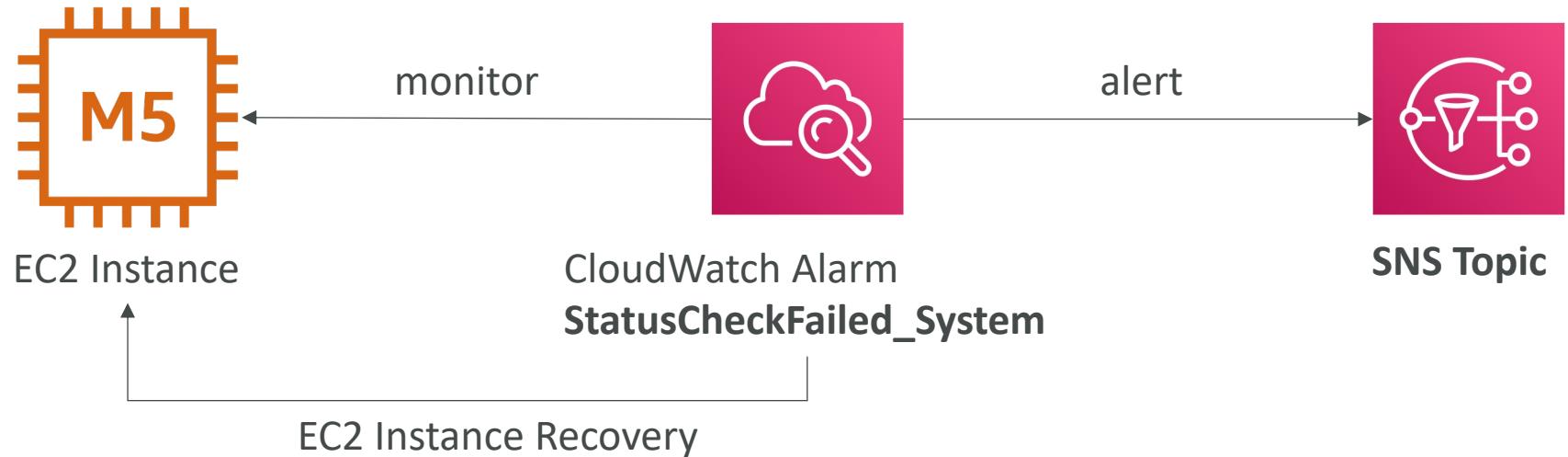
EC2 Auto Scaling



Amazon SNS

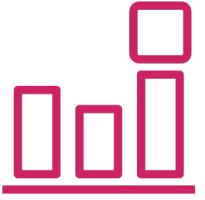
# EC2 Instance Recovery

- Status Check:
  - Instance status = check the EC2 VM
  - System status = check the underlying hardware



- Recovery: Same Private, Public, Elastic IP, metadata, placement group

# CloudWatch Events



- Event Pattern: Intercept events from AWS services (Sources)
  - Example sources: EC2 Instance Start, CodeBuild Failure, S3, Trusted Advisor
  - Can intercept any API call with CloudTrail integration
- Schedule or Cron (example: create an event every 4 hours)
- A JSON payload is created from the event and passed to a target...
  - **Compute:** Lambda, Batch, ECS task
  - **Integration:** SQS, SNS, Kinesis Data Streams, Kinesis Data Firehose
  - **Orchestration:** Step Functions, CodePipeline, CodeBuild
  - **Maintenance:** SSM, EC2 Actions



# Amazon EventBridge

- EventBridge is the next evolution of CloudWatch Events
- **Default Event Bus** – generated by AWS services (CloudWatch Events)
- **Partner Event Bus** – receive events from SaaS service or applications (Zendesk, DataDog, Segment, Auth0...)
- **Custom Event Buses** – for your own applications
- Event buses can be accessed by other AWS accounts
- You can **archive events** (all/filter) sent to an event bus (indefinitely or set period)
- Ability to **replay archived events**
- **Rules:** how to process the events (like CloudWatch Events)

# Amazon EventBridge – Schema Registry

- EventBridge can analyze the events in your bus and infer the **schema**
- The **Schema Registry** allows you to generate code for your application, that will know in advance how data is structured in the event bus
- Schema can be versioned

The screenshot shows the AWS Schema Registry interface. At the top, there's a header with the schema name: "aws.codepipeline@CodePipelineActionExecutionStateChange". Below the header, there's a section titled "Schema details" with the following information:

Schema name	Last modified	Schema ARN
aws.codepipeline@CodePipelineActionExecutionStateChange	Dec 1, 2019, 12:11 AM GMT	-
Description	Schema for event type CodePipelineActionExecutionStateChange, published by AWS service aws.codepipeline	Schema registry aws.events Number of versions 1 Schema type OpenAPI 3.0

Below the "Schema details" section, there's a "Version 1" section with a timestamp of "Created on Dec 1, 2019, 12:11 AM GMT". It includes an "Action" button and a "Download code bindings" button. The "Download code bindings" button is highlighted with an orange background. Below these buttons, the schema definition is shown as a JSON-like code block:

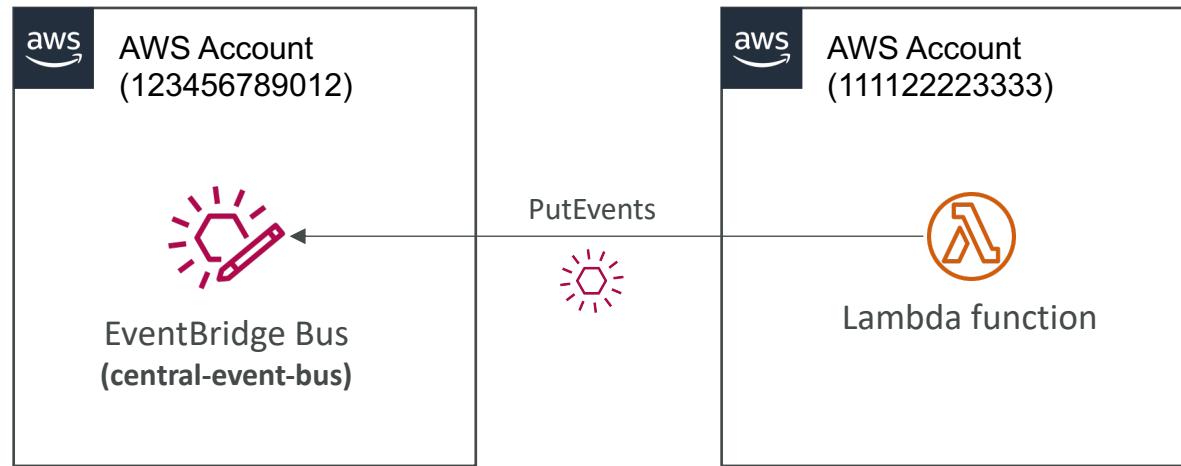
```
1 {
2   "openapi": "3.0.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "CodePipelineActionExecutionStateChange"
6   },
7   "paths": {},
8   "components": {
9     "schemas": {
10       "AWSEvent": {
```

# Amazon EventBridge – Resource-based Policy

- Manage permissions for a specific Event Bus
- Example: allow/deny events from another AWS account or AWS region
- Use case: aggregate all events from your AWS Organization in a single AWS account or AWS region

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "events:PutEvents",  
            "Principal": { "AWS": "111122223333" },  
            "Resource": "arn:aws:events:us-east-1:123456789012:  
event-bus/central-event-bus"  
        }  
    ]  
}
```

Allow **events** from another AWS account



# Amazon EventBridge vs CloudWatch Events

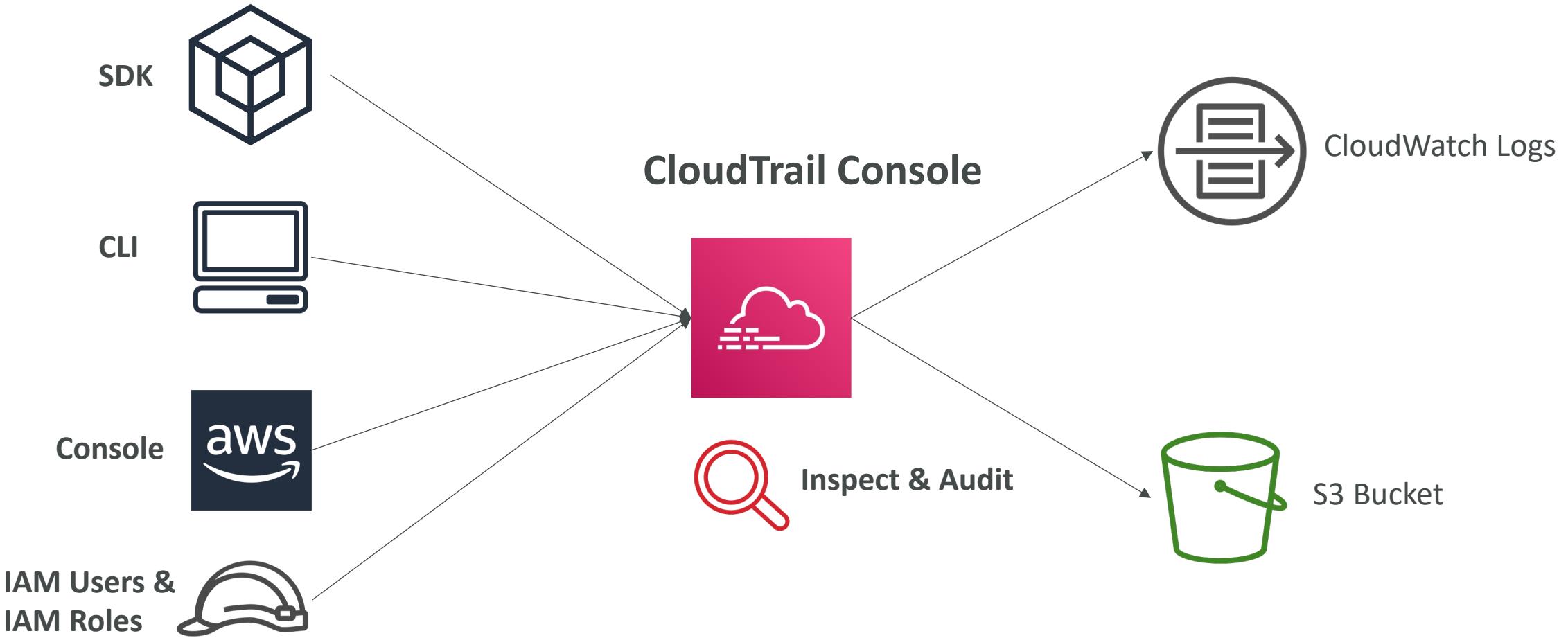
- Amazon EventBridge builds upon and extends CloudWatch Events.
  - It uses the same service API and endpoint, and the same underlying service infrastructure.
  - EventBridge allows extension to add event buses for your custom applications and your third-party SaaS apps.
  - Event Bridge has the Schema Registry capability
- 
- EventBridge has a different name to mark the new capabilities
  - Over time, the CloudWatch Events name will be replaced with EventBridge.



# AWS CloudTrail

- Provides governance, compliance and audit for your AWS Account
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
  - Console
  - SDK
  - CLI
  - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs or S3
- A trail can be applied to All Regions (default) or a single Region.
- If a resource is deleted in AWS, investigate CloudTrail first!

# CloudTrail Diagram





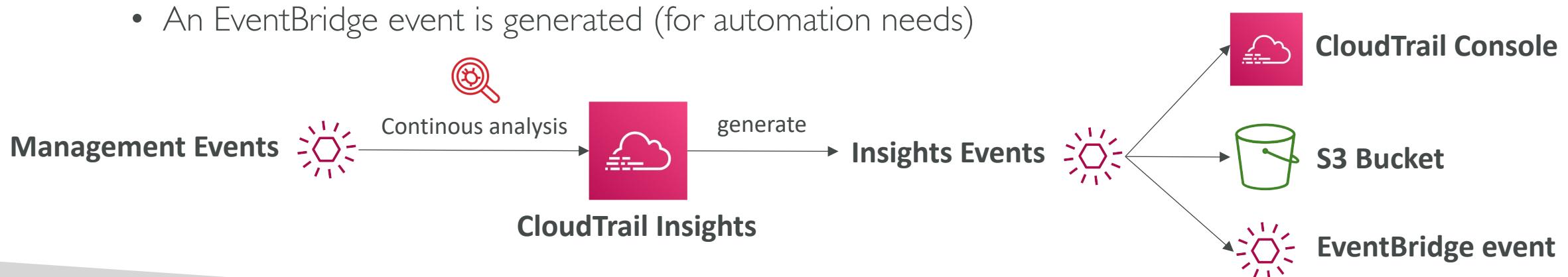
# CloudTrail Events

- Management Events:
  - Operations that are performed on resources in your AWS account
  - Examples:
    - Configuring security (IAM `AttachRolePolicy`)
    - Configuring rules for routing data (Amazon EC2 `CreateSubnet`)
    - Setting up logging (AWS CloudTrail `CreateTrail`)
  - By default, trails are configured to log management events.
  - Can separate Read Events (that don't modify resources) from Write Events (that may modify resources)
- Data Events:
  - By default, data events are not logged (because high volume operations)
  - Amazon S3 object-level activity (ex: `GetObject`, `DeleteObject`, `PutObject`): can separate Read and Write Events
  - AWS Lambda function execution activity (the `Invoke` API)
- CloudTrail Insights Events:
  - See next slide ☺



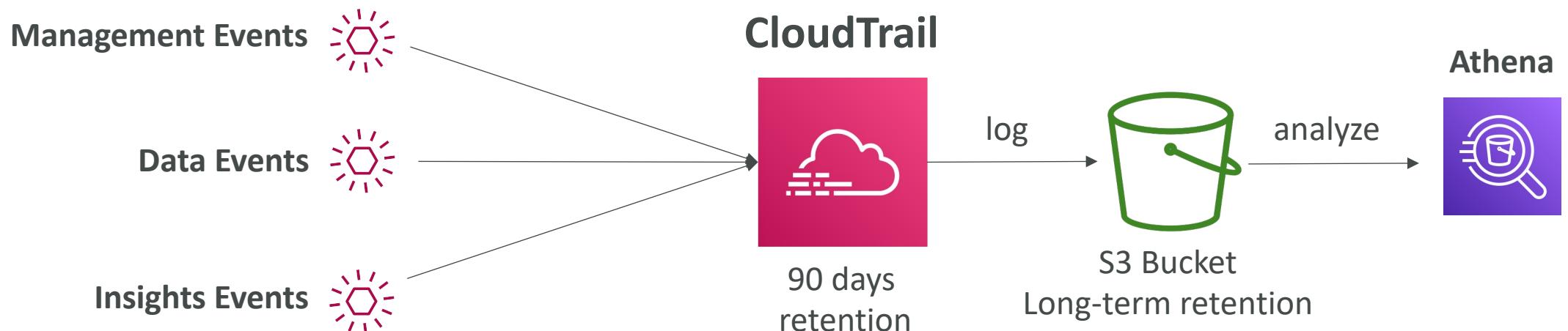
# CloudTrail Insights

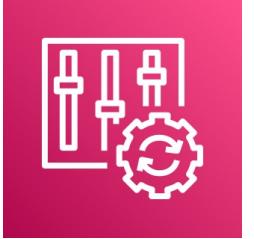
- Enable CloudTrail Insights to detect unusual activity in your account:
  - inaccurate resource provisioning
  - hitting service limits
  - Bursts of AWS IAM actions
  - Gaps in periodic maintenance activity
- CloudTrail Insights analyzes normal management events to create a baseline
- And then continuously analyzes write events to detect unusual patterns
  - Anomalies appear in the CloudTrail console
  - Event is sent to Amazon S3
  - An EventBridge event is generated (for automation needs)



# CloudTrail Events Retention

- Events are stored for 90 days in CloudTrail
- To keep events beyond this period, log them to S3 and use Athena





# AWS Config

- Helps with auditing and recording **compliance** of your AWS resources
- Helps record configurations and changes over time
- Questions that can be solved by AWS Config:
  - Is there unrestricted SSH access to my security groups?
  - Do my buckets have any public access?
  - How has my ALB configuration changed over time?
- You can receive alerts (SNS notifications) for any changes
- AWS Config is a per-region service
- Can be aggregated across regions and accounts
- Possibility of storing the configuration data into S3 (analyzed by Athena)

# Config Rules

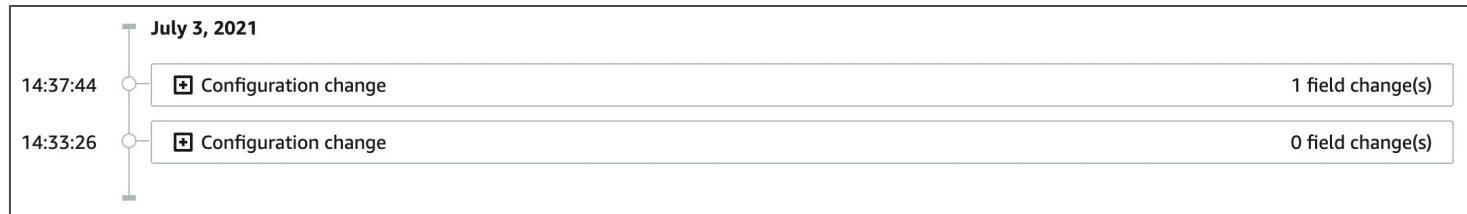
- Can use AWS managed config rules (over 75)
- Can make custom config rules (must be defined in AWS Lambda)
  - Ex: evaluate if each EBS disk is of type gp2
  - Ex: evaluate if each EC2 instance is t2.micro
- Rules can be evaluated / triggered:
  - For each config change
  - And / or: at regular time intervals
- AWS Config Rules does not prevent actions from happening (no deny)
- Pricing: no free tier, \$0.003 per configuration item recorded per region, \$0.001 per config rule evaluation per region

# AWS Config Resource

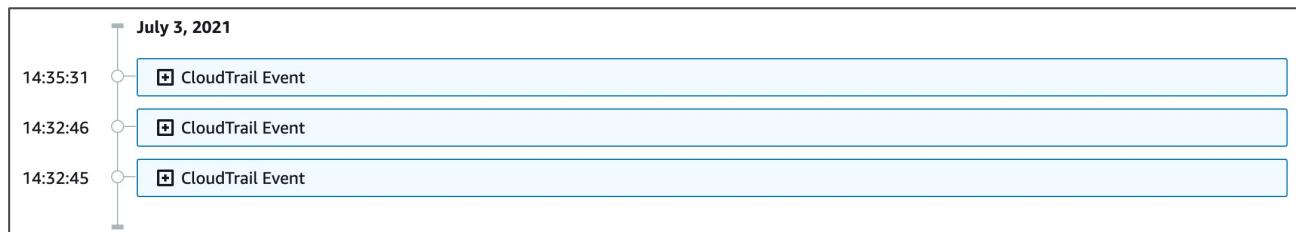
- View compliance of a resource over time

<input type="radio"/> sg-077b425b1649da83e	EC2 SecurityGroup	 Compliant
<input type="radio"/> sg-0831434f1876c0c74	EC2 SecurityGroup	 Noncompliant
<input type="radio"/> sg-09f10ed254d464f30	EC2 SecurityGroup	 Compliant

- View configuration of a resource over time

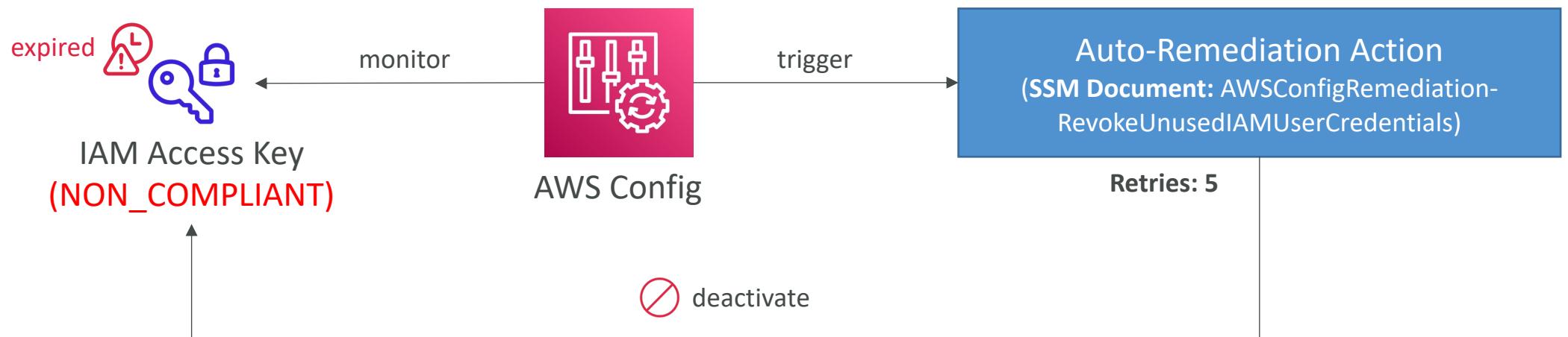


- View CloudTrail API calls of a resource over time



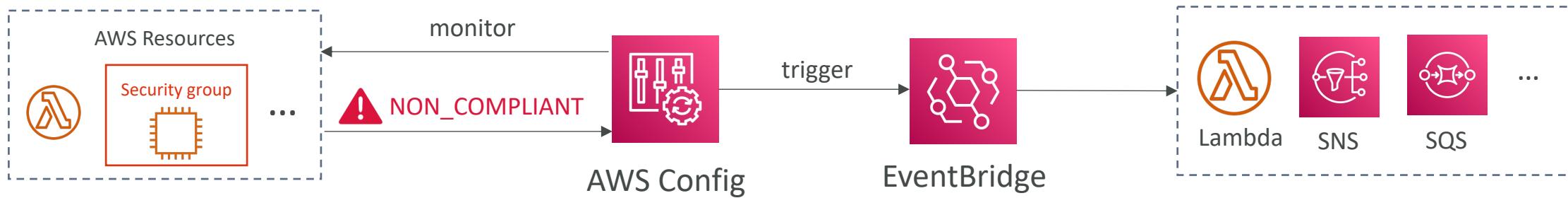
# Config Rules – Remediations

- Automate remediation of non-compliant resources using SSM Automation Documents
- Use AWS-Managed Automation Documents or create custom Automation Documents
  - Tip: you can create custom Automation Documents that invokes Lambda function
- You can set **Remediation Retries** if the resource is still non-compliant after auto-remediation

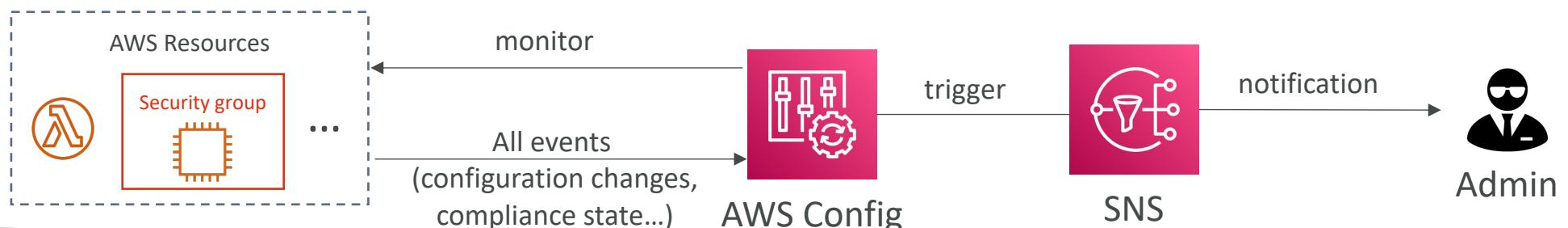


# Config Rules – Notifications

- Use EventBridge to trigger notifications when AWS resources are non-compliant



- Ability to send configuration changes and compliance state notifications to SNS (all events – use SNS Filtering or filter at client-side)



# CloudWatch vs CloudTrail vs Config

- CloudWatch
  - Performance monitoring (metrics, CPU, network, etc...) & dashboards
  - Events & Alerting
  - Log Aggregation & Analysis
- CloudTrail
  - Record API calls made within your Account by everyone
  - Can define trails for specific resources
  - Global Service
- Config
  - Record configuration changes
  - Evaluate resources against compliance rules
  - Get timeline of changes and compliance

# For an Elastic Load Balancer

- CloudWatch:
  - Monitoring Incoming connections metric
  - Visualize error codes as % over time
  - Make a dashboard to get an idea of your load balancer performance
- Config:
  - Track security group rules for the Load Balancer
  - Track configuration changes for the Load Balancer
  - Ensure an SSL certificate is always assigned to the Load Balancer (compliance)
- CloudTrail:
  - Track who made any changes to the Load Balancer with API calls

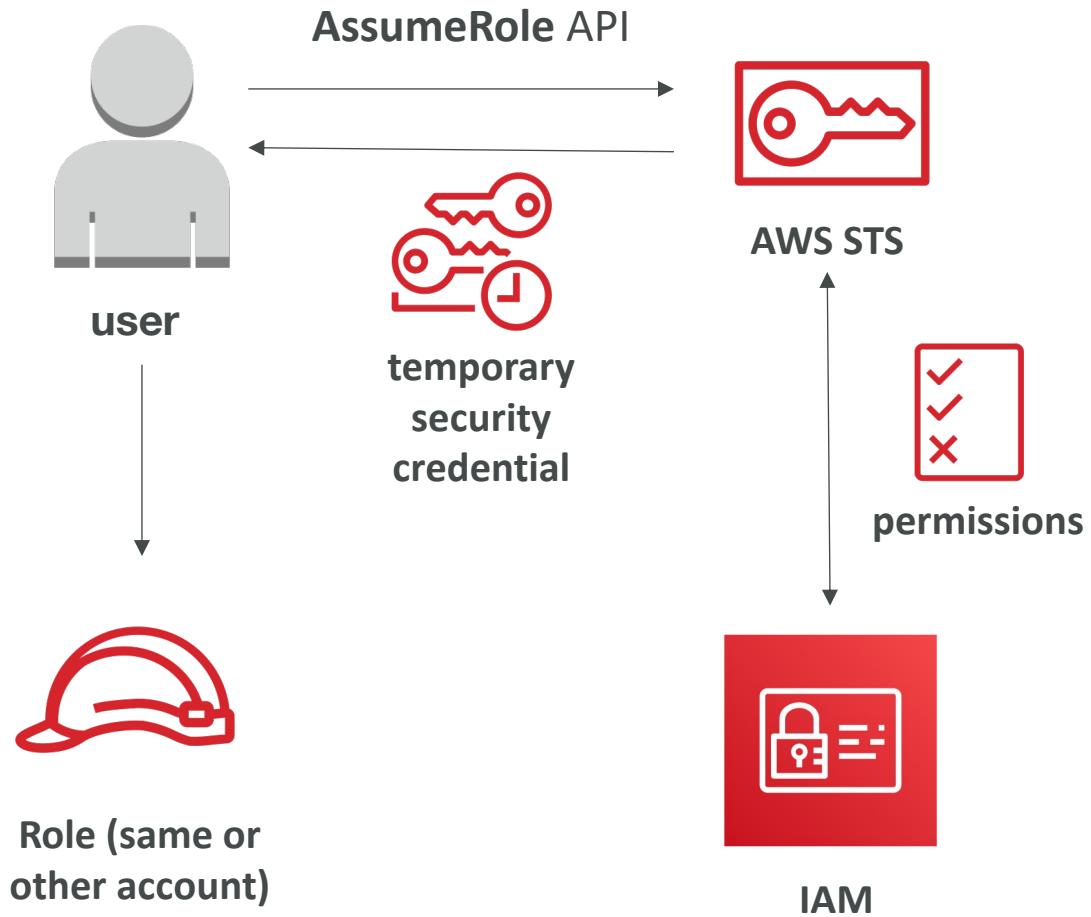


# AWS STS – Security Token Service

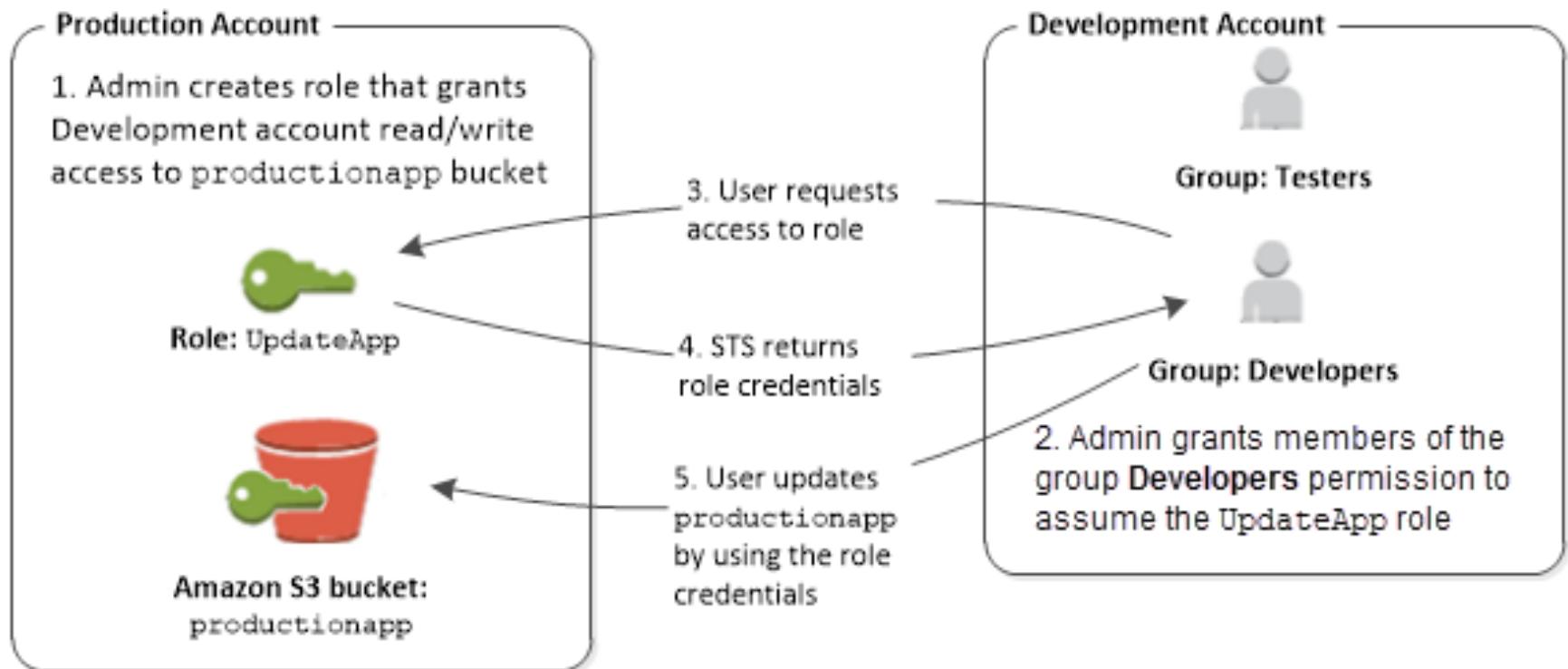
- Allows to grant limited and temporary access to AWS resources.
- Token is valid for up to one hour (must be refreshed)
- **AssumeRole**
  - Within your own account: for enhanced security
  - Cross Account Access: assume role in target account to perform actions there
- **AssumeRoleWithSAML**
  - return credentials for users logged with SAML
- **AssumeRoleWithWebIdentity**
  - return creds for users logged with an IdP (Facebook Login, Google Login, OIDC compatible...)
  - AWS recommends against using this, and using **Cognito** instead
- **GetSessionToken**
  - for MFA, from a user or AWS account root user

# Using STS to Assume a Role

- Define an IAM Role within your account or cross-account
- Define which principals can access this IAM Role
- Use AWS STS (Security Token Service) to retrieve credentials and impersonate the IAM Role you have access to (`AssumeRole API`)
- Temporary credentials can be valid between 15 minutes to 1 hour



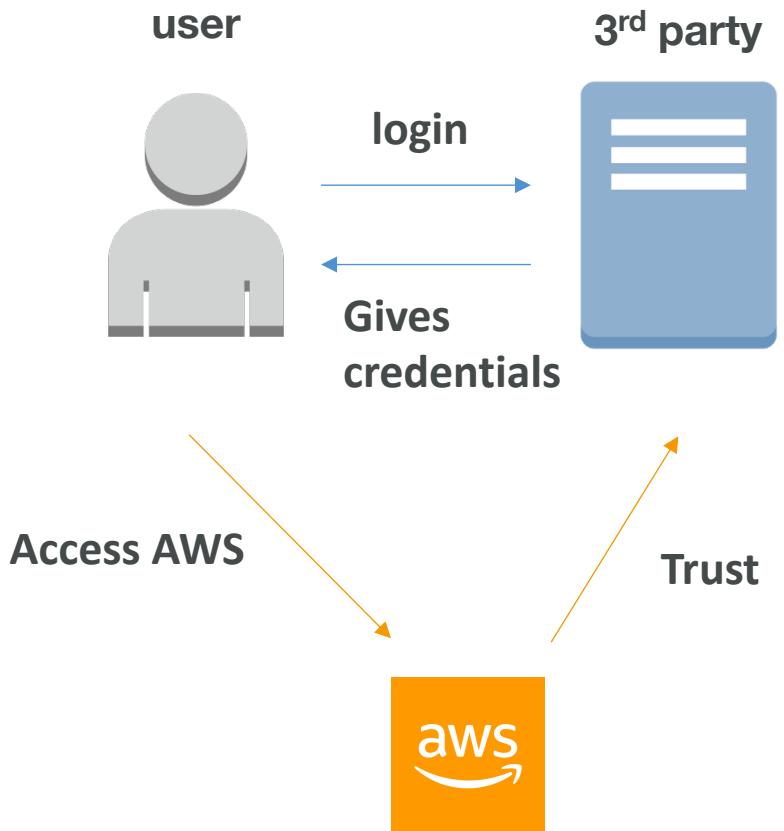
# Cross account access with STS



[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_common-scenarios\\_awssaccounts.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_common-scenarios_awssaccounts.html)

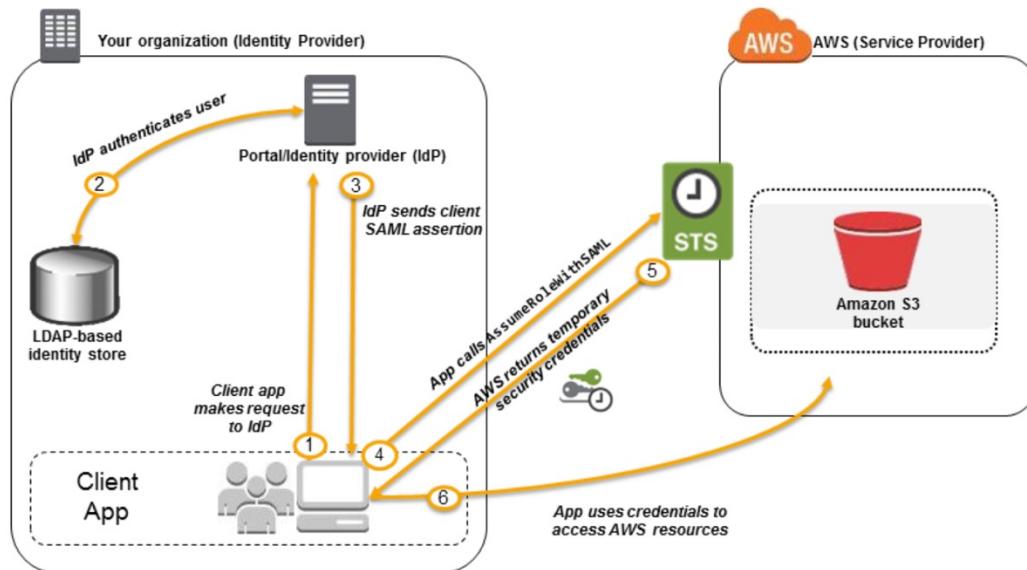
# Identity Federation in AWS

- Federation lets users outside of AWS to assume temporary role for accessing AWS resources.
- These users assume identity provided access role.
- Federations can have many flavors:
  - SAML 2.0
  - Custom Identity Broker
  - Web Identity Federation with Amazon Cognito
  - Web Identity Federation without Amazon Cognito
  - Single Sign On
  - Non-SAML with AWS Microsoft AD
- Using federation, you don't need to create IAM users  
(user management is outside of AWS)

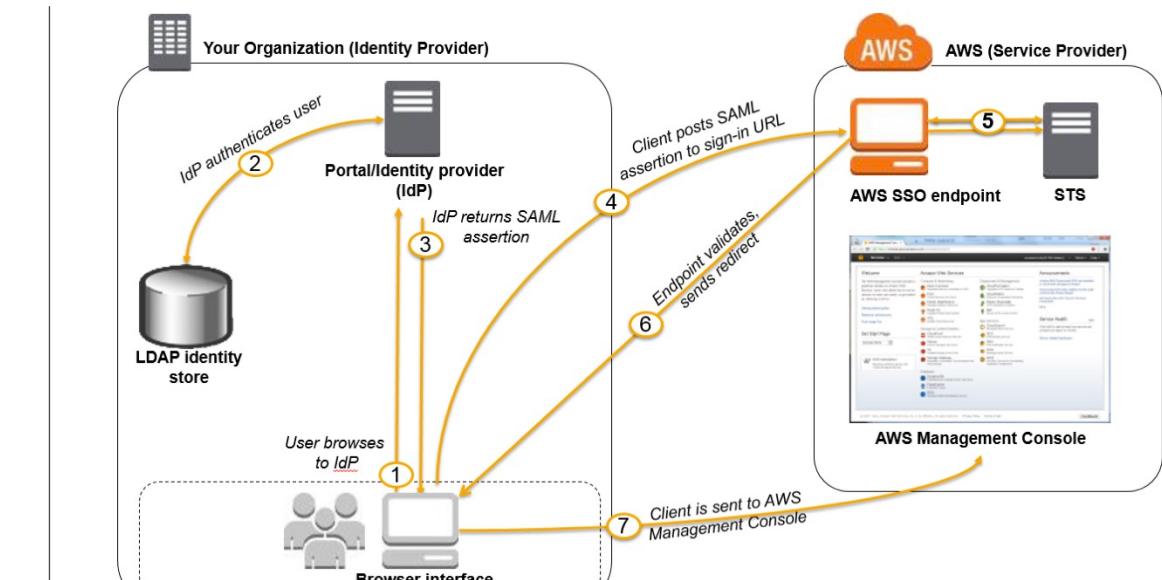


# SAML 2.0 Federation

- To integrate Active Directory / ADFS with AWS (or any SAML 2.0)
- Provides access to AWS Console or CLI (through temporary creds)
- No need to create an IAM user for each of your employees



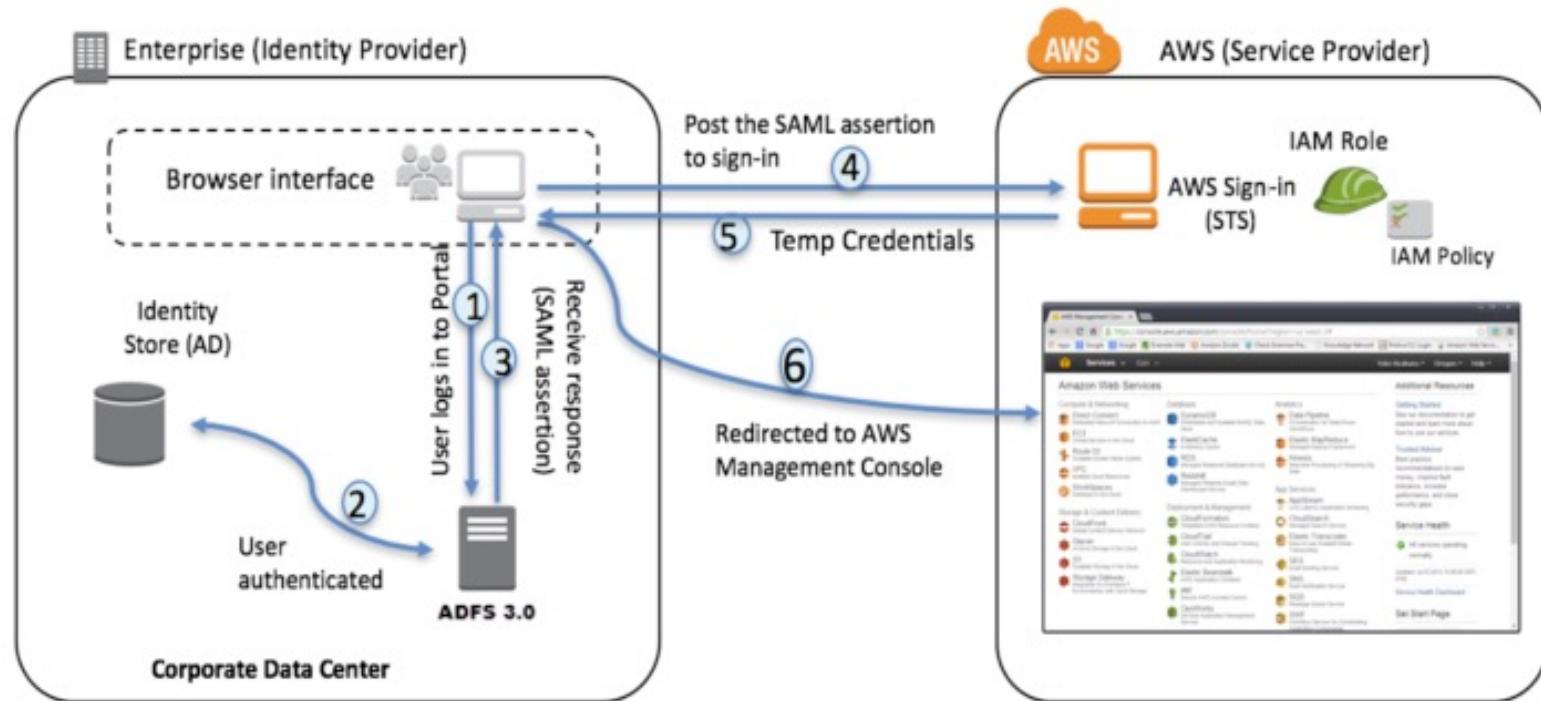
[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_providers\\_saml.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_saml.html)



[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_providers\\_enable-console-saml.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_enable-console-saml.html)

# SAML 2.0 Federation – Active Directory FS

- Same process as with any SAML 2.0 compatible IdP



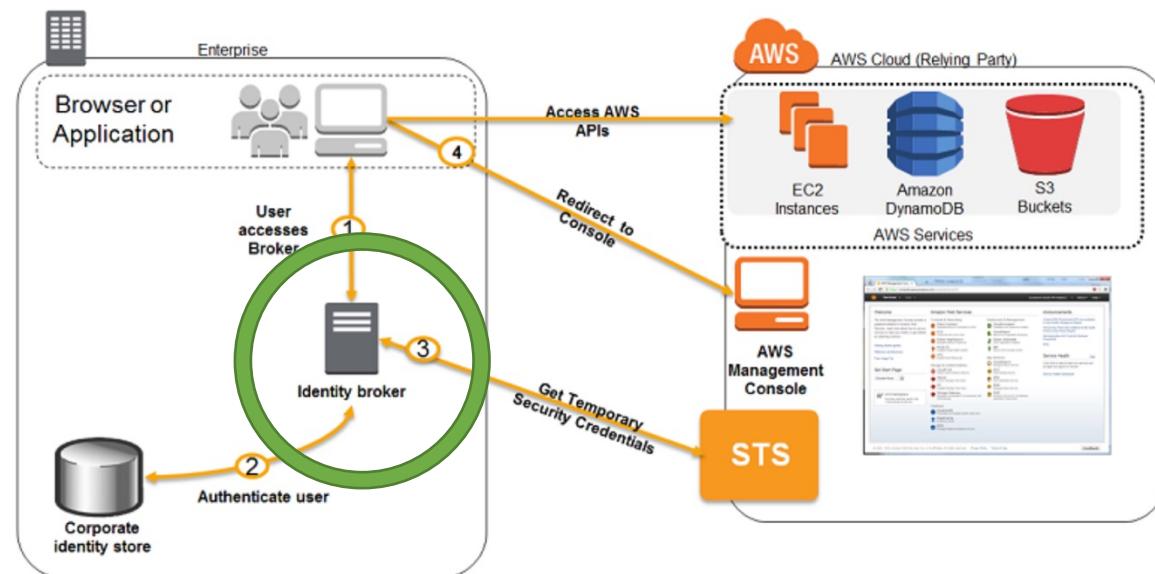
<https://aws.amazon.com/blogs/security/aws-federated-authentication-with-active-directory-federation-services-ad-fs/>

# SAML 2.0 Federation

- Needs to setup a trust between AWS IAM and SAML (both ways)
- SAML 2.0 enables web-based, cross domain SSO
- Uses the STS API: AssumeRoleWithSAML
- Note federation through SAML is the “old way” of doing things
- **Amazon Single Sign On (SSO) Federation** is the new managed and simpler way
  - Read more here: <https://aws.amazon.com/blogs/security/enabling-federation-to-aws-using-windows-active-directory-adfs-and-saml-2-0/>

# Custom Identity Broker Application

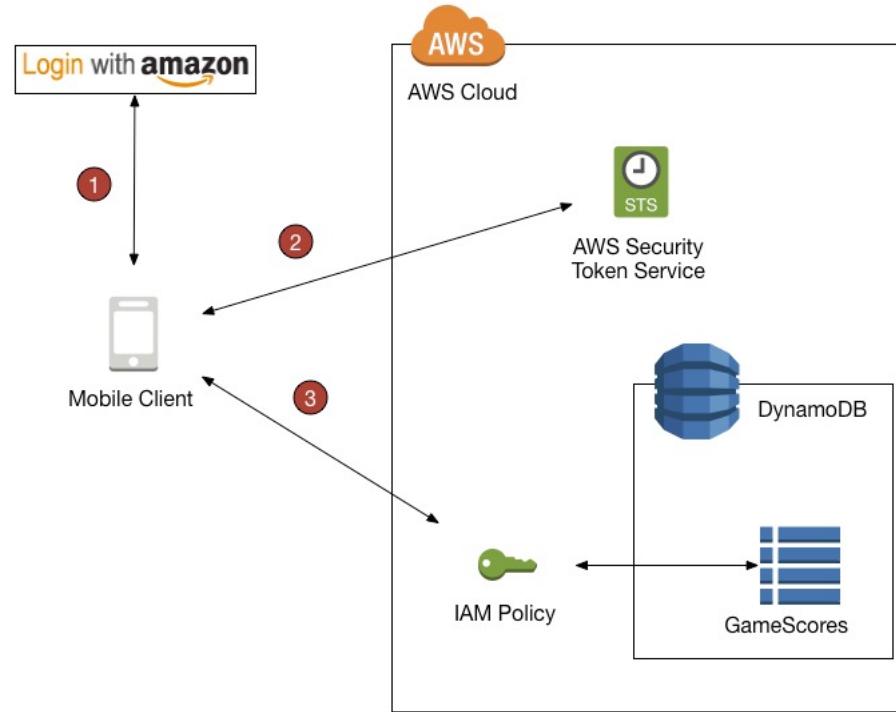
- Use only if identity provider is not compatible with SAML 2.0
- The identity broker must determine the appropriate IAM policy
- Uses the STS API: AssumeRole or GetFederationToken



[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles-common-scenarios\\_federated-users.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles-common-scenarios_federated-users.html)

# Web Identity Federation – AssumeRoleWithWebIdentity

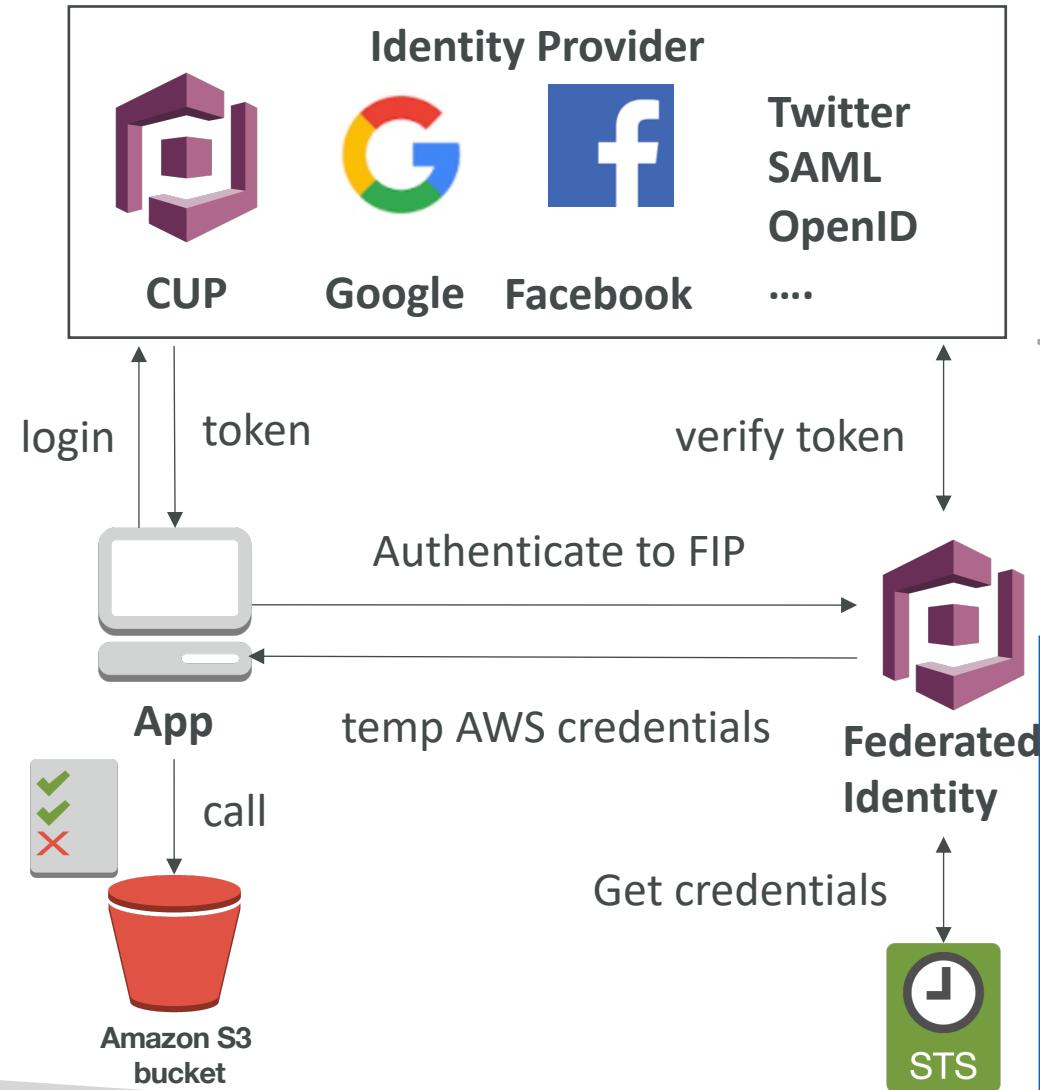
- Not recommended by AWS – use Cognito instead (allows for anonymous users, data synchronization, MFA)



[https://docs.amazonaws.cn/en\\_us/amazondynamodb/latest/developerguide/WIF.html](https://docs.amazonaws.cn/en_us/amazondynamodb/latest/developerguide/WIF.html)

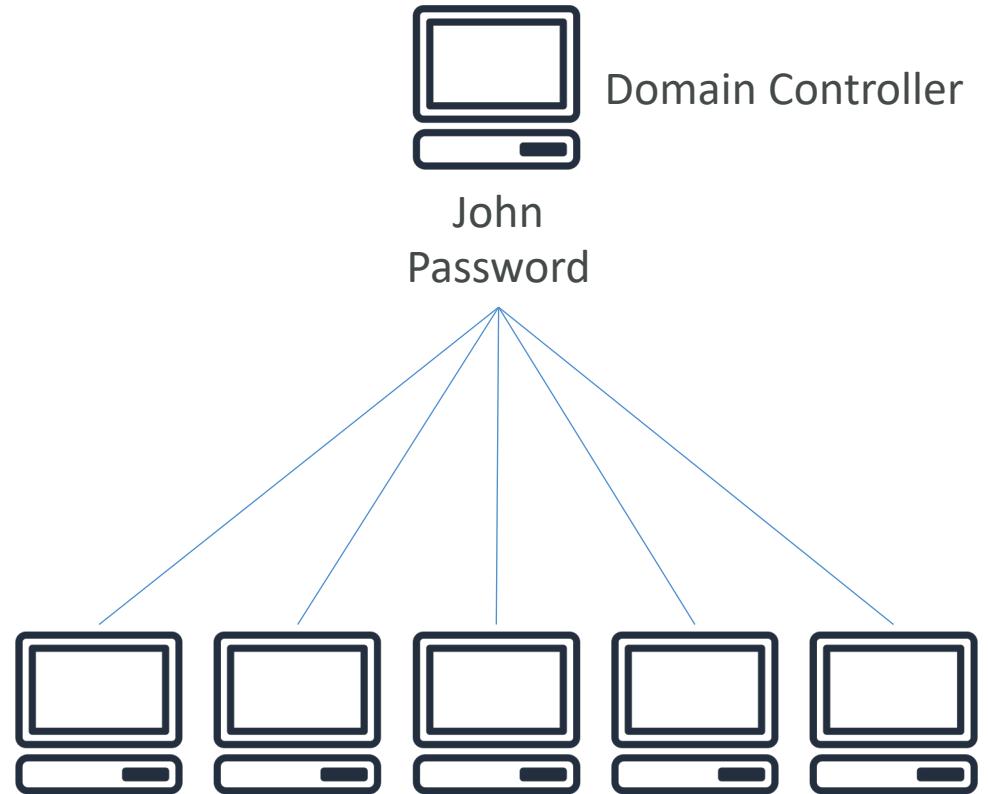
# AWS Cognito

- **Goal:**
  - Provide direct access to AWS Resources from the Client Side (mobile, web app)
- **Example:**
  - provide (temporary) access to write to S3 bucket using Facebook Login
- **Problem:**
  - We don't want to create IAM users for our app users
- **How:**
  - Log in to federated identity provider – or remain anonymous
  - Get temporary AWS credentials back from the Federated Identity Pool
  - These credentials come with a pre-defined IAM policy stating their permissions



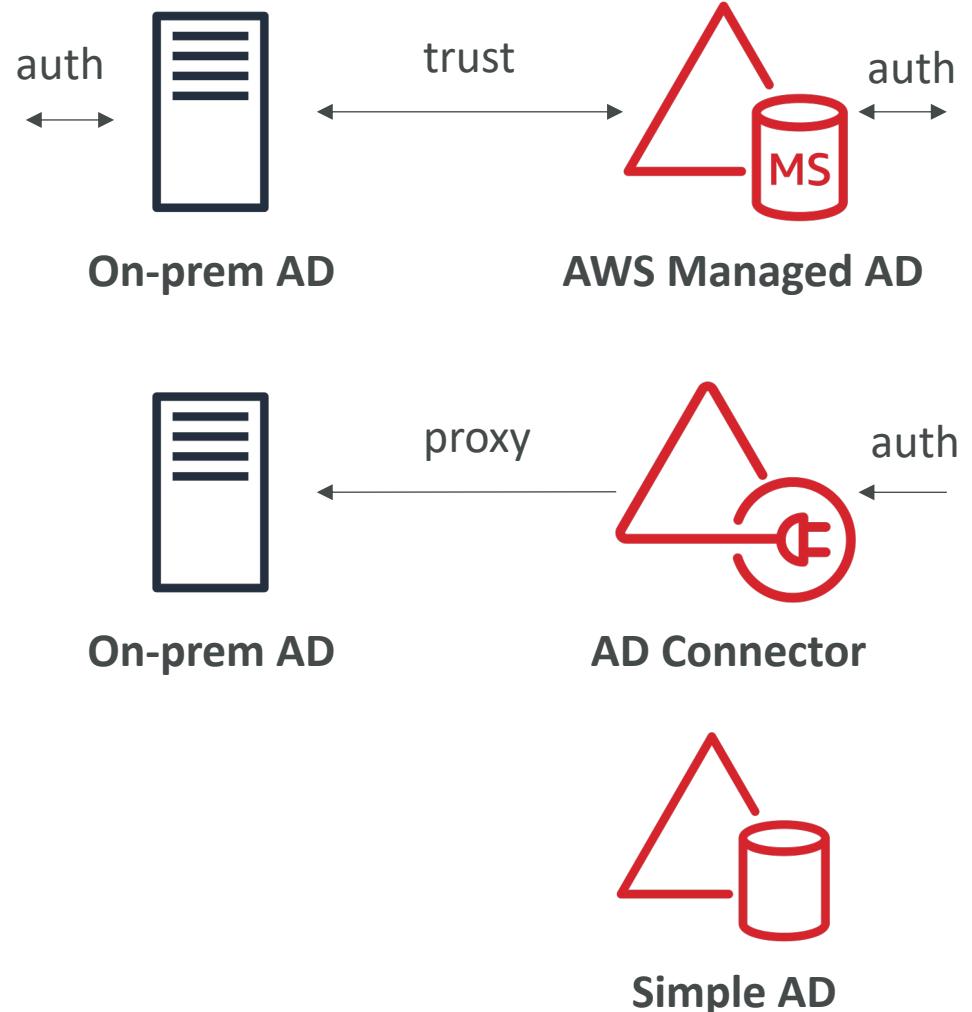
# What is Microsoft Active Directory (AD)?

- Found on any Windows Server with AD Domain Services
- Database of **objects**: User Accounts, Computers, Printers, File Shares, Security Groups
- Centralized security management, create account, assign permissions
- Objects are organized in **trees**
- A group of trees is a **forest**

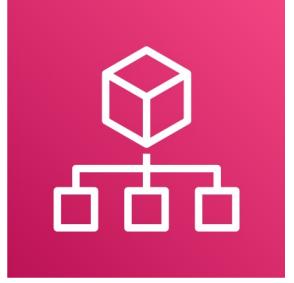


# AWS Directory Services

- AWS Managed Microsoft AD
  - Create your own AD in AWS, manage users locally, supports MFA
  - Establish “trust” connections with your on-premises AD
- AD Connector
  - Directory Gateway (proxy) to redirect to on-premises AD, supports MFA
  - Users are managed on the on-premises AD
- Simple AD
  - AD-compatible managed directory on AWS
  - Cannot be joined with on-premises AD



# AWS Organizations



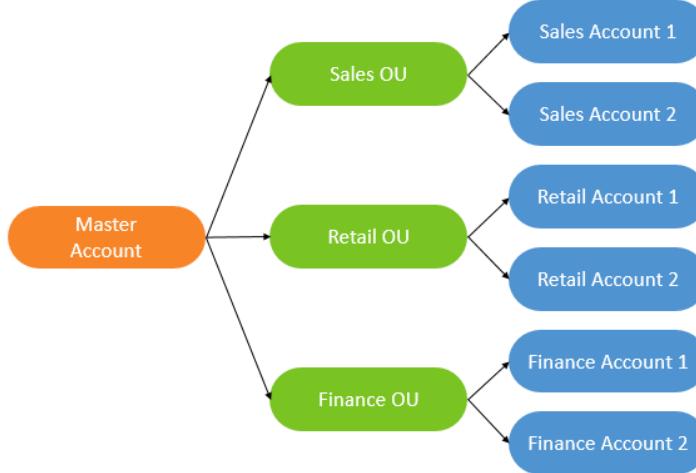
- Global service
- Allows to manage multiple AWS accounts
- The main account is the master account – you can't change it
- Other accounts are member accounts
- Member accounts can only be part of one organization
- Consolidated Billing across all accounts - single payment method
- Pricing benefits from aggregated usage (volume discount for EC2, S3...)
- API is available to automate AWS account creation

# Multi Account Strategies

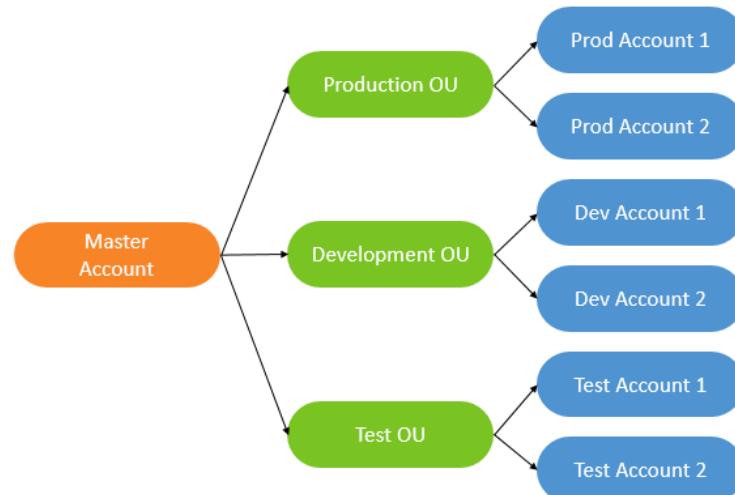
- Create accounts per department, per cost center, per dev / test / prod, based on regulatory restrictions (using SCP), for better resource isolation (ex:VPC), to have separate per-account service limits, isolated account for logging
- Multi Account vs One Account Multi VPC
- Use tagging standards for billing purposes
- Enable CloudTrail on all accounts, send logs to central S3 account
- Send CloudWatch Logs to central logging account
- Establish Cross Account Roles for Admin purposes

# Organizational Units (OU) - Examples

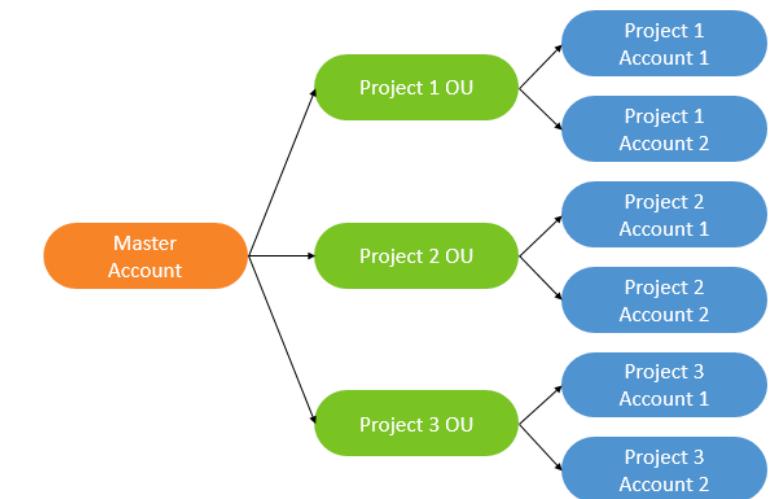
**Business Unit**



**Environmental Lifecycle**

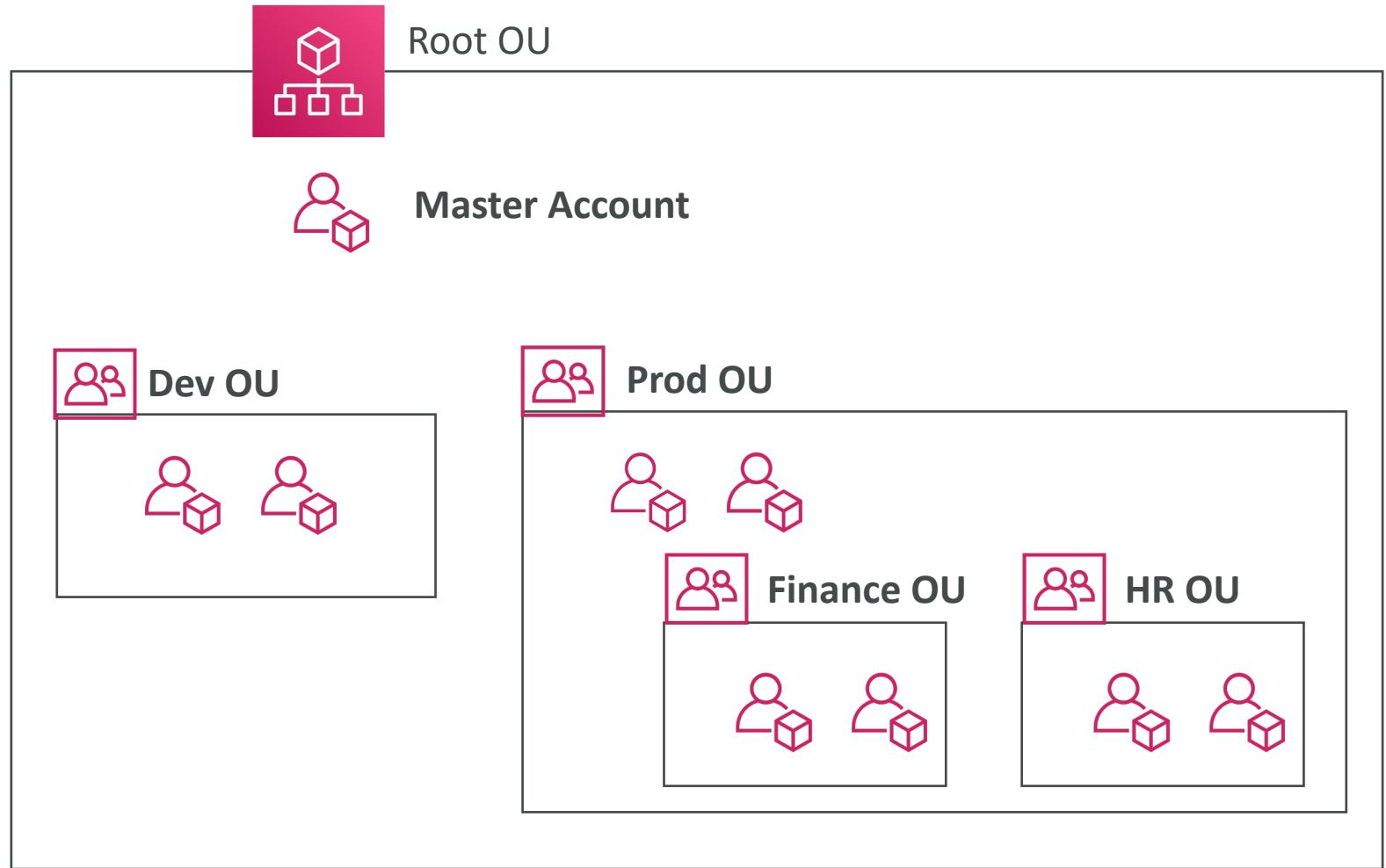


**Project-based**



<https://aws.amazon.com/answers/account-management/aws-multi-account-billing-strategy/>

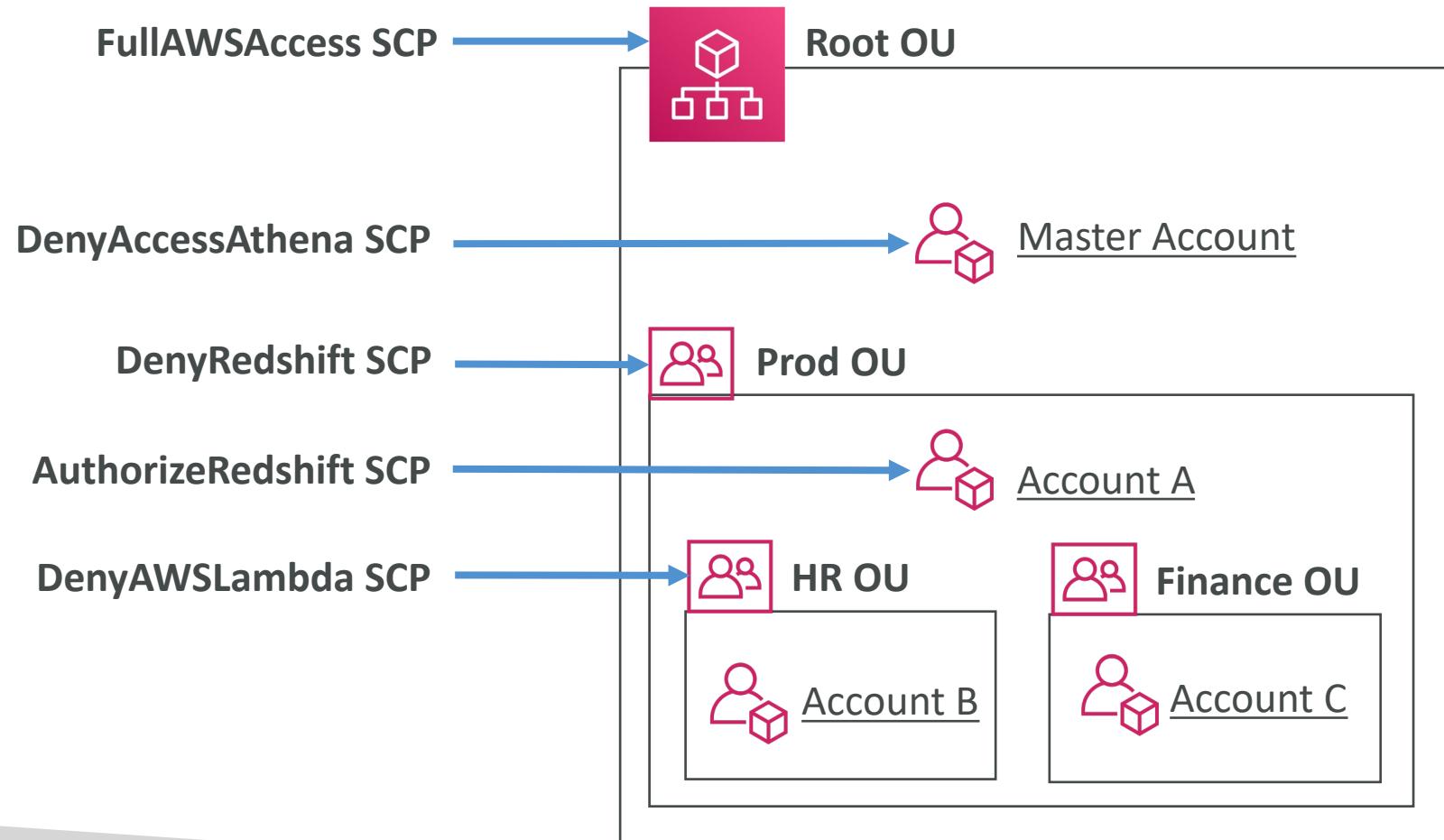
# AWS Organization



# Service Control Policies (SCP)

- Whitelist or blacklist IAM actions
- Applied at the **OU** or **Account** level
- Does not apply to the Master Account
- SCP is applied to all the **Users** and **Roles** of the Account, including Root user
- The SCP does not affect service-linked roles
  - Service-linked roles enable other AWS services to integrate with AWS Organizations and can't be restricted by SCPs.
- SCP must have an explicit Allow (does not allow anything by default)
- Use cases:
  - Restrict access to certain services (for example: can't use EMR)
  - Enforce PCI compliance by explicitly disabling services

# SCP Hierarchy



- Master Account
  - Can do anything
  - (no SCP apply)
- Account A
  - Can do anything
  - EXCEPT access Redshift (explicit Deny from OU)
- Account B
  - Can do anything
  - EXCEPT access Redshift (explicit Deny from Prod OU)
  - EXCEPT access Lambda (explicit Deny from HR OU)
- Account C
  - Can do anything
  - EXCEPT access Redshift (explicit Deny from Prod OU)

# SCP Examples

## Blacklist and Whitelist strategies

```
Version": "2012-10-17",
"Statement": [
    {
        "Sid": "AllowsAllActions",
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
    },
    {
        "Sid": "DenyDynamoDB",
        "Effect": "Deny",
        "Action": "dynamodb:*",
        "Resource": "*"
    }
]
```

```
Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "ec2:*",
            "cloudwatch:*
```

More examples: [https://docs.aws.amazon.com/organizations/latest/userguide/orgs\\_manage\\_policies\\_example-scps.html](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_example-scps.html)

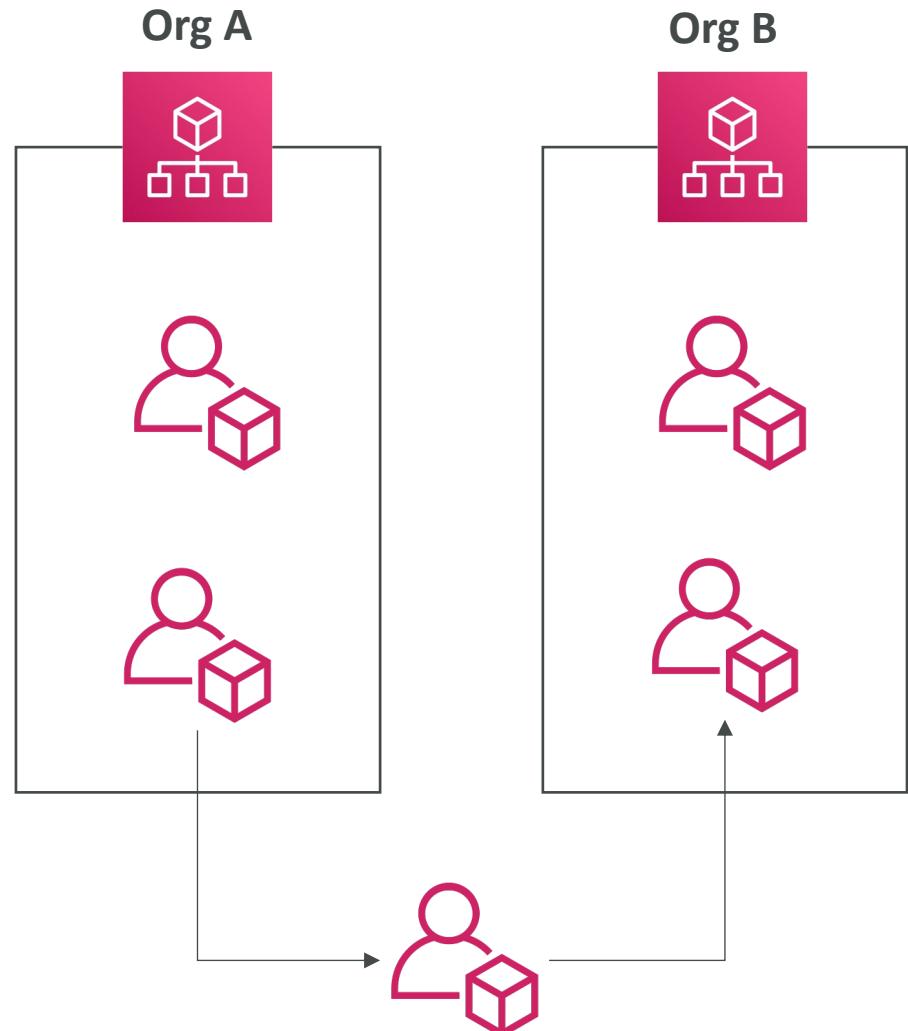
# AWS Organization – Moving Accounts

To migrate accounts from one organization to another

1. Remove the member account from the old organization
2. Send an invite to the new organization
3. Accept the invite to the new organization from the member account

If you want the master account of the old organization to also join the new organization, do the following:

1. Remove the member accounts from the organizations using procedure above
2. Delete the old organization
3. Repeat the process above to invite the old master account to the new org



# IAM Conditions

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "NotIpAddress": {
        "aws:SourceIp": [
          "192.0.2.0/24",
          "203.0.113.0/24"
        ]
      },
    }
  }
}
```

**aws:SourceIP:** restrict the client IP from which the API calls are being made

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowOnlyInsideEU",
      "Effect": "Allow",
      "Action": [
        "ec2:*",
        "rds:*",
        "dynamodb:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": [
            "eu-central-1",
            "eu-west-1"
          ]
        }
      }
    }
  ]
}
```

**Aws:RequestedRegion:** restrict the region The API calls are made to

# IAM Conditions

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "StartStopIfTags",  
            "Effect": "Allow",  
            "Action": [  
                "ec2:StartInstances",  
                "ec2:StopInstances",  
                "ec2:DescribeTags"  
            ],  
            "Resource": "arn:aws:ec2:region:account-id:instance/*",  
            "Condition": {  
                "StringEquals": {  
                    "ec2:ResourceTag/Project                    "aws:PrincipalTag/Department                }  
            }  
        }  
    ]  
}
```

Restrict based on tags

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowAllActionsForEC2",  
            "Effect": "Allow",  
            "Action": "ec2:*",  
            "Resource": "*"  
        },  
        {  
            "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",  
            "Effect": "Deny",  
            "Action": [  
                "ec2:StopInstances",  
                "ec2:TerminateInstances"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "BoolIfExists": {"aws:MultiFactorAuthPresent": false}  
            }  
        }  
    ]  
}
```

Force MFA

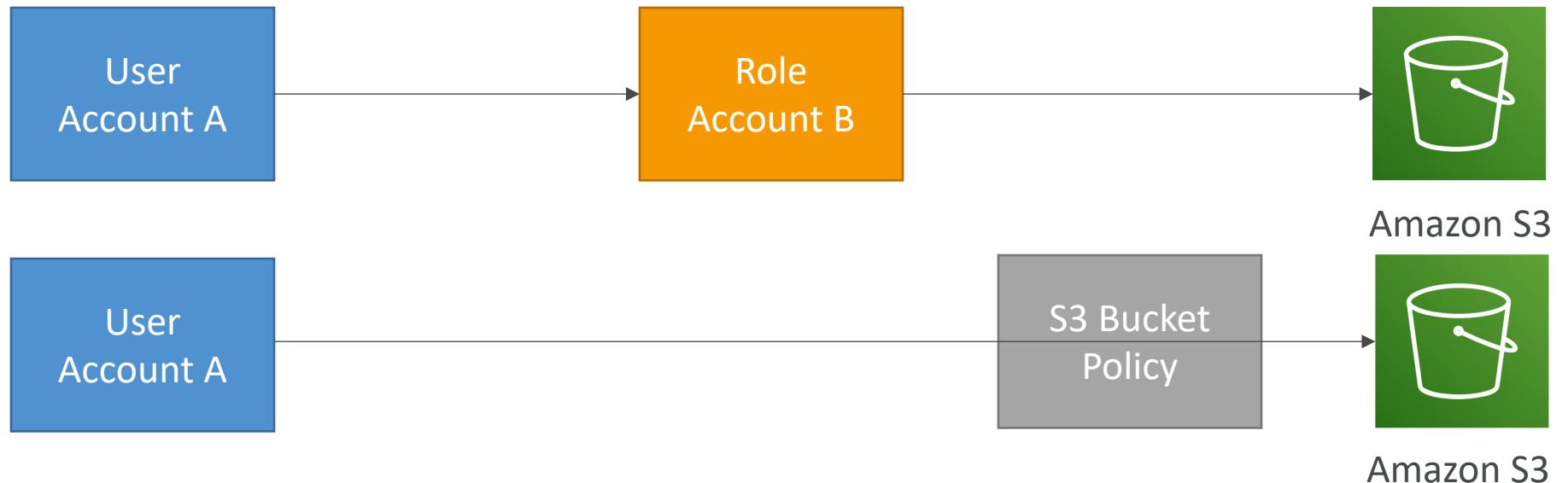
# IAM for S3

- ListBucket permission applies to  
arn:aws:s3:::test
- => bucket level permission
- GetObject, PutObject,  
DeleteObject applies to  
arn:awn:s3:::test/\*
- => object level permission

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["s3>ListBucket"],  
            "Resource": ["arn:aws:s3:::test"]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3>PutObject",  
                "s3>GetObject",  
                "s3>DeleteObject"  
            ],  
            "Resource": ["arn:aws:s3:::test/*"]  
        }  
    ]  
}
```

# IAM Roles vs Resource Based Policies

- Attach a policy to a resource (example: S3 bucket policy) versus attaching of a using a role as a proxy



# IAM Roles vs Resource Based Policies

- When you assume a role (user, application or service), you give up your original permissions and take the permissions assigned to the role
- When using a resource based policy, the principal doesn't have to give up his permissions
- Example: User in account A needs to scan a DynamoDB table in Account A and dump it in an S3 bucket in Account B.
- Supported by: Amazon S3 buckets, SNS topics, SQS queues, etc...

# IAM Permission Boundaries

- IAM Permission Boundaries are supported for users and roles (not groups)
- Advanced feature to use a managed policy to set the maximum permissions an IAM entity can get.

**Example:**

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:*",  
                "cloudwatch:*",  
                "ec2:*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```



**IAM Permission Boundary**

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iam>CreateUser",  
            "Resource": "*"  
        }  
    ]  
}
```

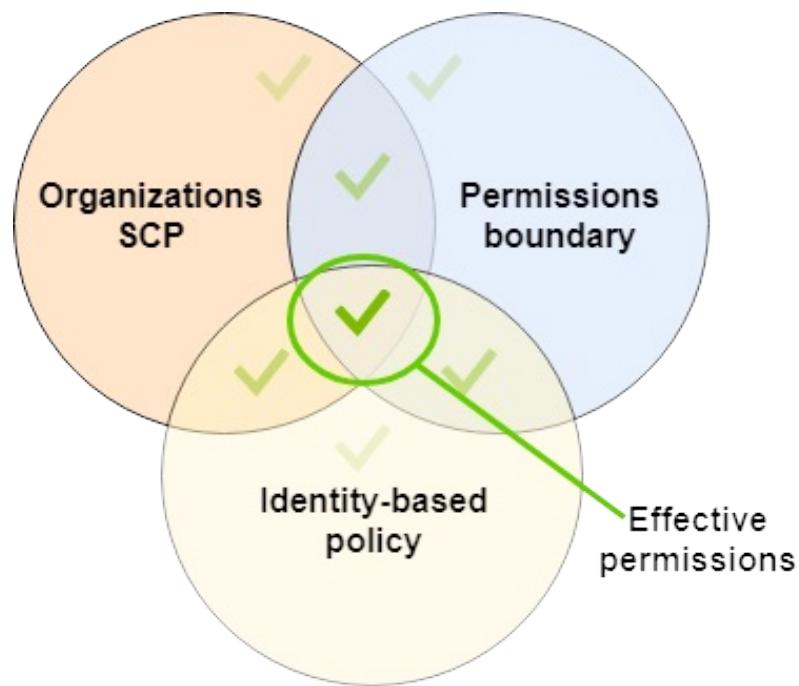
**IAM Permissions  
Through IAM Policy**



**No Permissions**

# IAM Permission Boundaries

- Can be used in combinations of AWS Organizations SCP

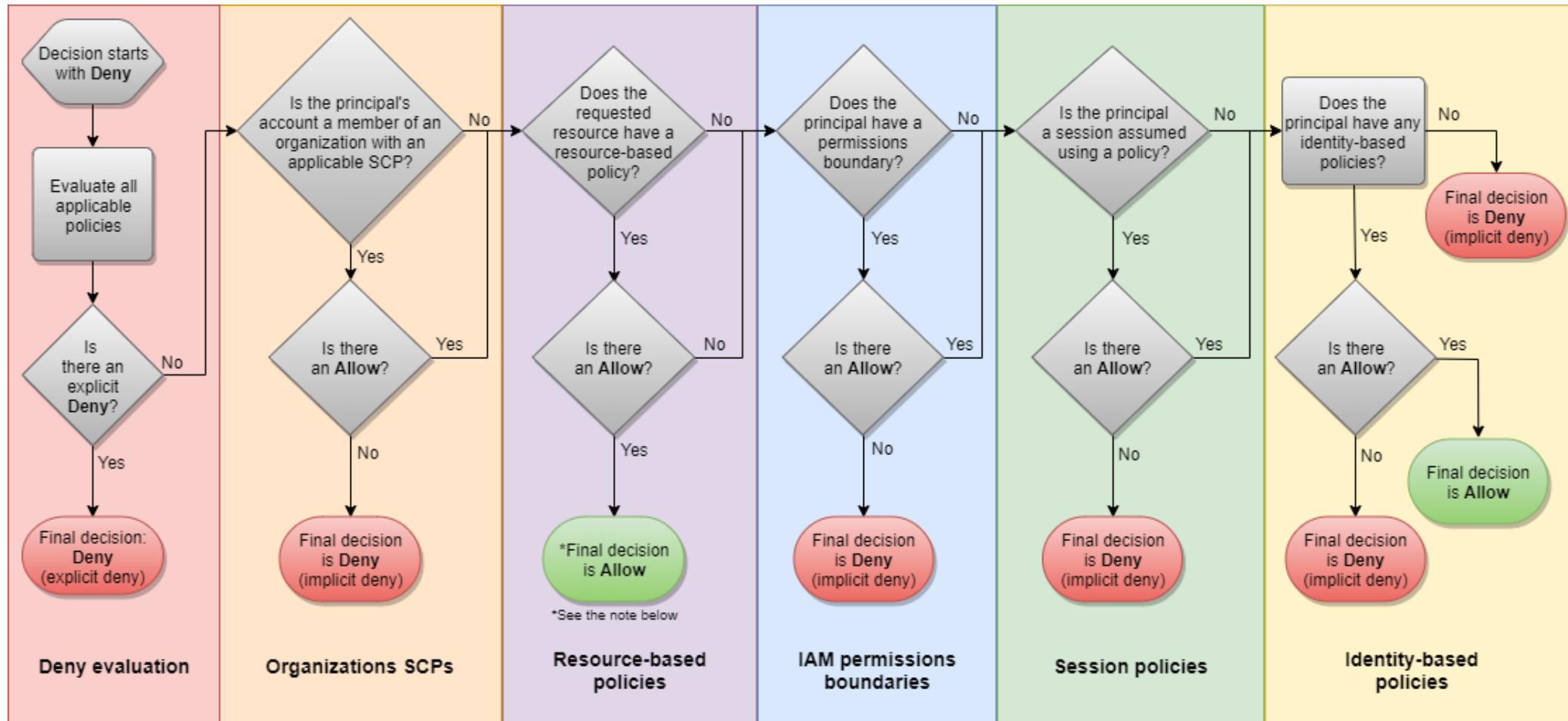


[https://docs.aws.amazon.com/IAM/latest/UserGuide/access\\_policies\\_boundaries.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html)

## Use cases

- Delegate responsibilities to non administrators within their permission boundaries, for example create new IAM users
- Allow developers to self-assign policies and manage their own permissions, while making sure they can't "escalate" their privileges (= make themselves admin)
- Useful to restrict one specific user (instead of a whole account using Organizations & SCP)

# IAM Policy Evaluation Logic



[https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_evaluation-logic.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html)

# Example IAM Policy

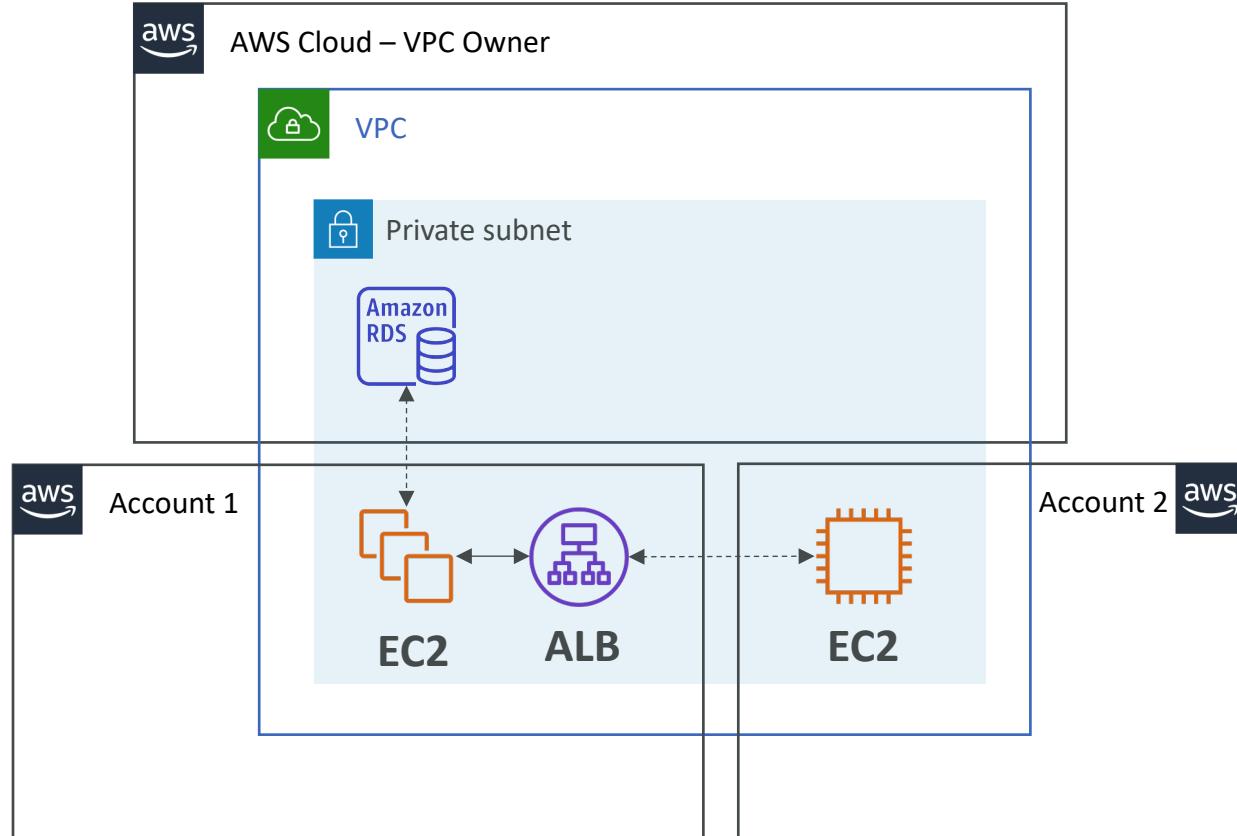
- Can you perform sqs:CreateQueue?
- Can you perform sqs:DeleteQueue?
- Can you perform ec2:DescribeInstances?

```
"Version": "2012-10-17",
"Statement": [
  {
    "Action": "sts:AssumeRole",
    "Effect": "Allow",
    "Resource": "*"
  },
  {
    "Action": [
      "sts:AssumeRole"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
```

# AWS Resource Access Manager (RAM)

- Share AWS resources that you own with other AWS accounts
- Share with any account or within your Organization
- Avoid resource duplication!
- **VPC Subnets:**
  - allow to have all the resources launched in the same subnets
  - must be from the same AWS Organizations.
  - Cannot share security groups and default VPC
  - Participants can manage their own resources in there
  - Participants can't view, modify, delete resources that belong to other participants or the owner
- AWS Transit Gateway
- Route53 Resolver Rules
- License Manager Configurations

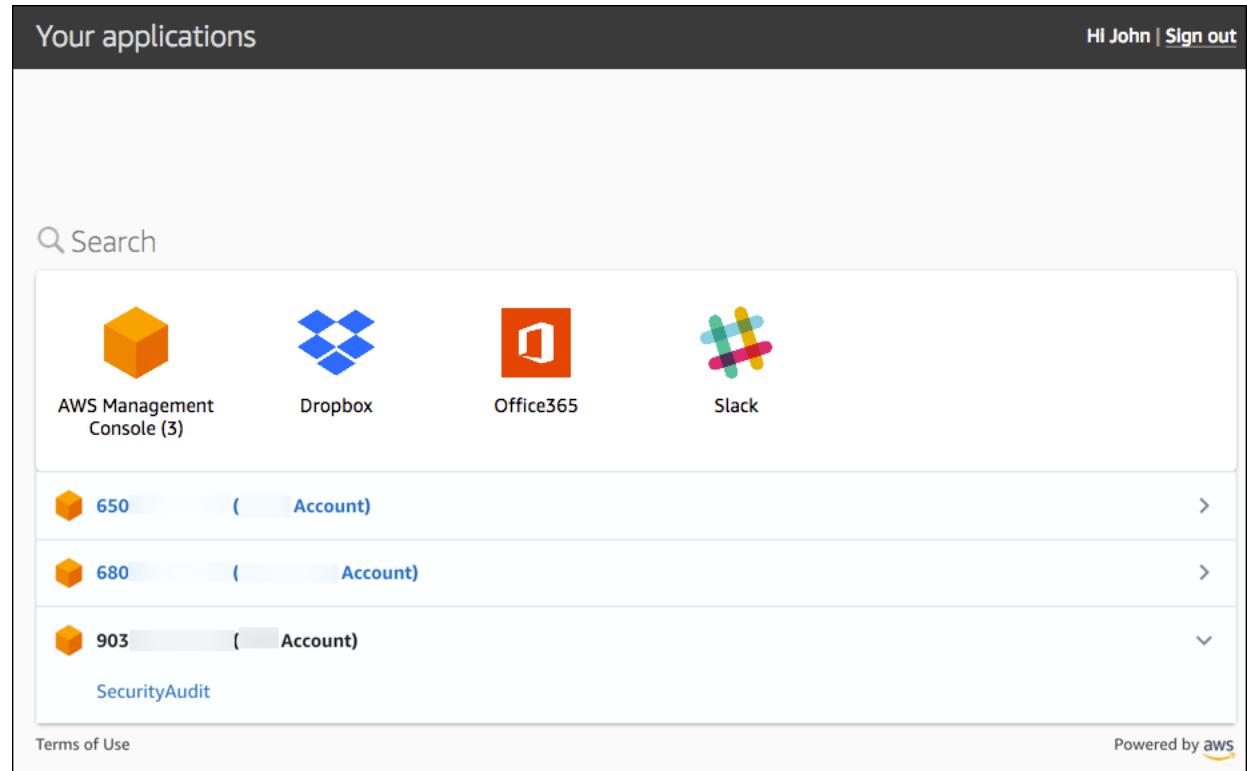
# Resource Access Manager – VPC example



- Each account...
  - is responsible for its own resources
  - cannot view, modify or delete other resources in other accounts
- Network is shared so...
  - Anything deployed in the VPC can talk to other resources in the VPC
  - Applications are accessed easily across accounts, using private IP!
  - Security groups from other accounts can be referenced for maximum security

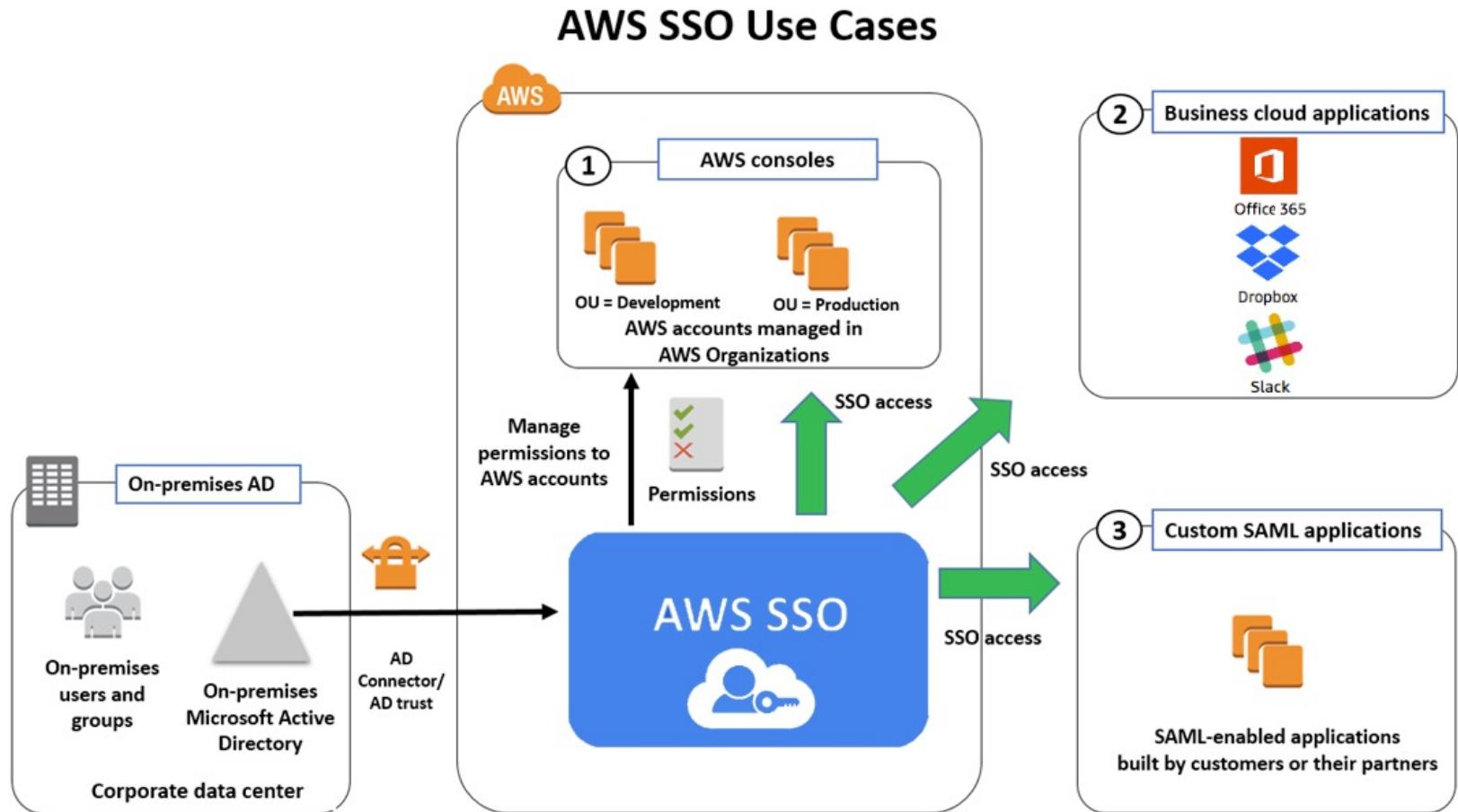
# AWS Single Sign-On (SSO)

- Centrally manage Single Sign-On to access **multiple accounts** and **3<sup>rd</sup>-party business applications**.
- Integrated with AWS Organizations
- Supports SAML 2.0 markup
- Integration with on-premises **Active Directory**
- Centralized permission management
- Centralized auditing with CloudTrail

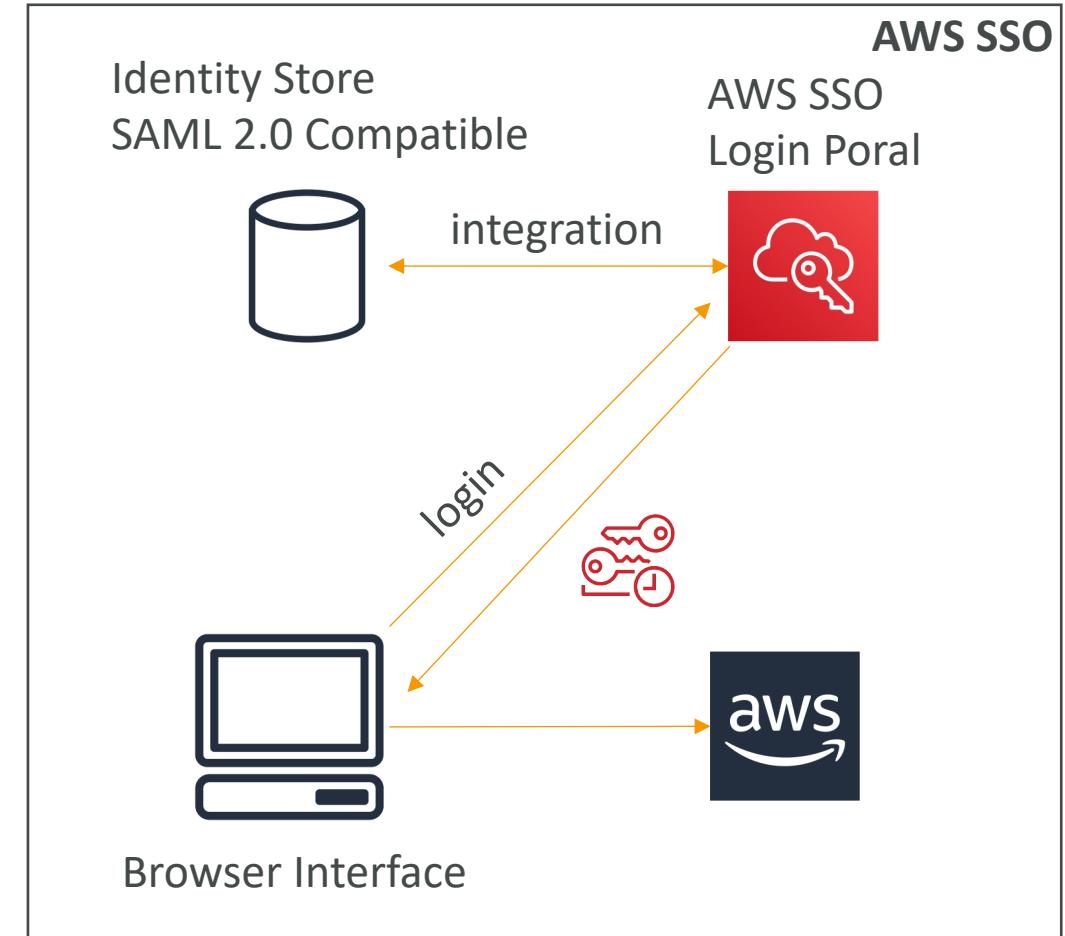
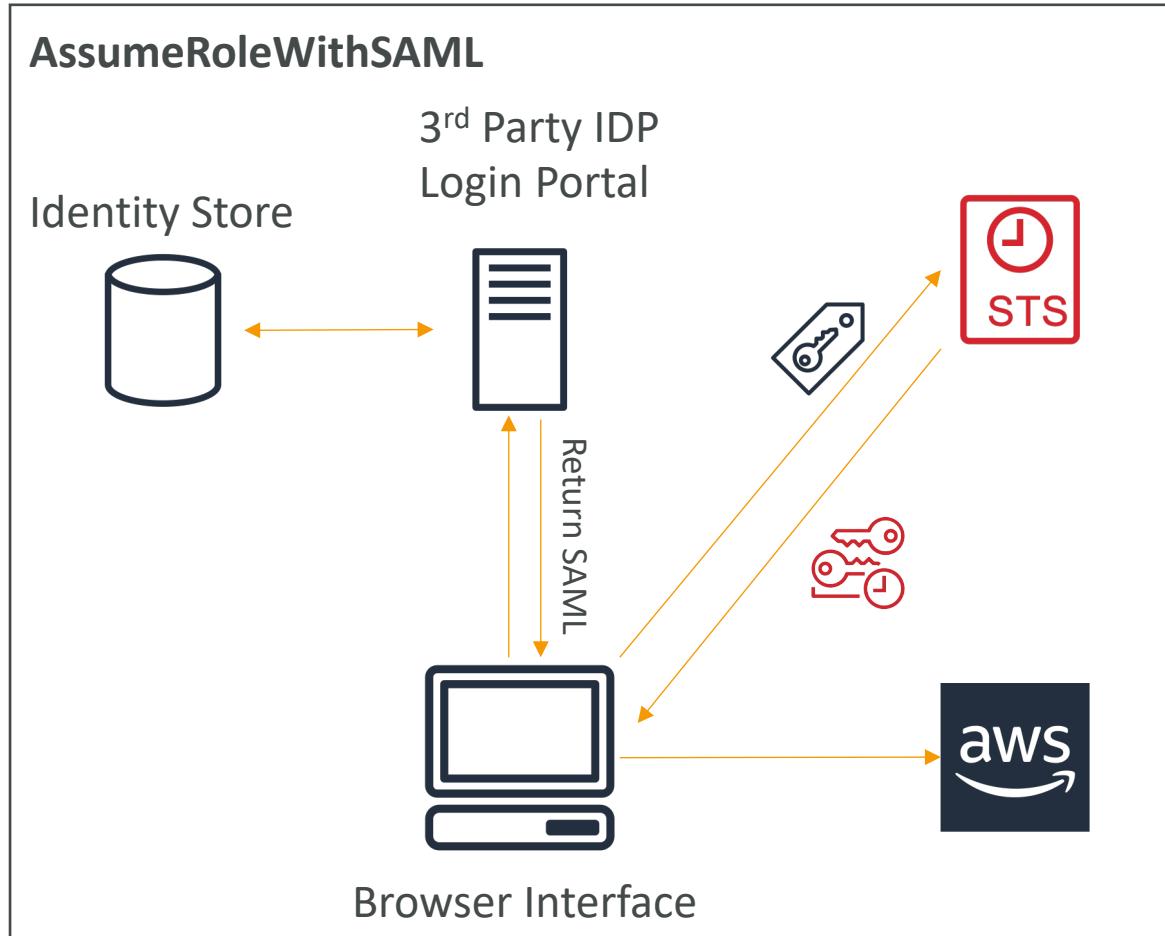


<https://aws.amazon.com/blogs/security/introducing-aws-single-sign-on/>

# AWS Single Sign-On (SSO) – Setup with AD



# SSO – vs AssumeRoleWithSAML



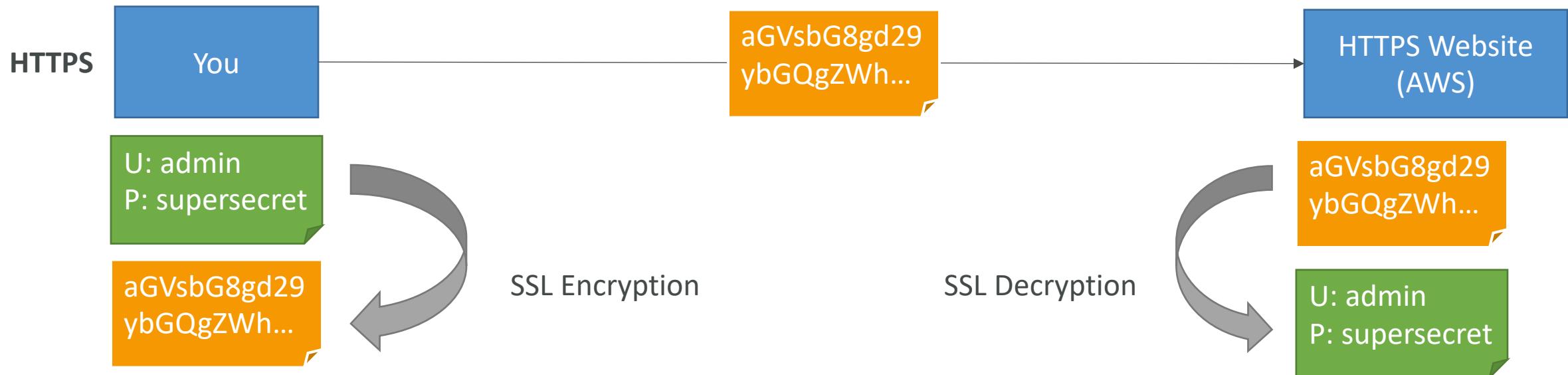
# AWS Security & Encryption

KMS, Encryption SDK, SSM Parameter Store

# Why encryption?

## Encryption in flight (SSL)

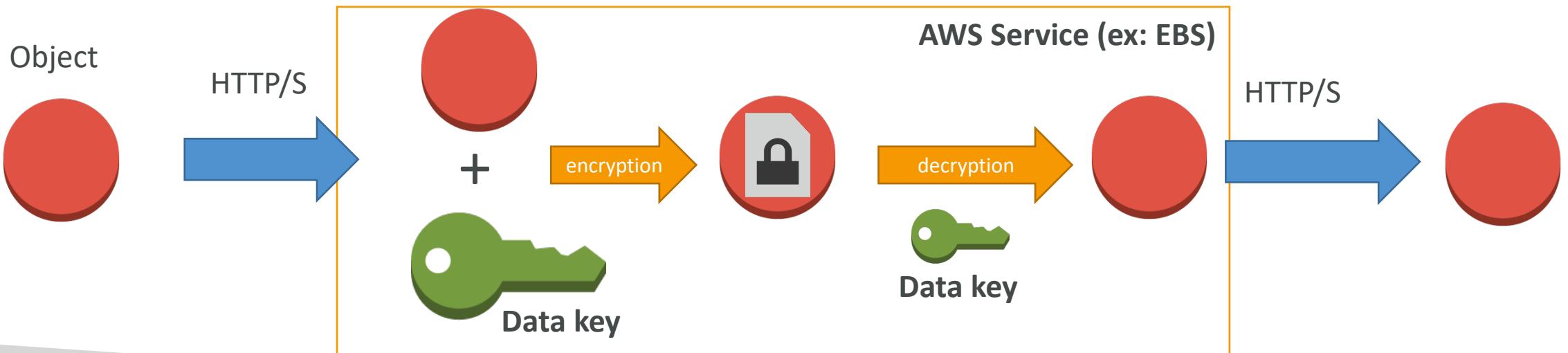
- Data is encrypted before sending and decrypted after receiving
- SSL certificates help with encryption (HTTPS)
- Encryption in flight ensures no MITM (man in the middle attack) can happen



# Why encryption?

## Server side encryption at rest

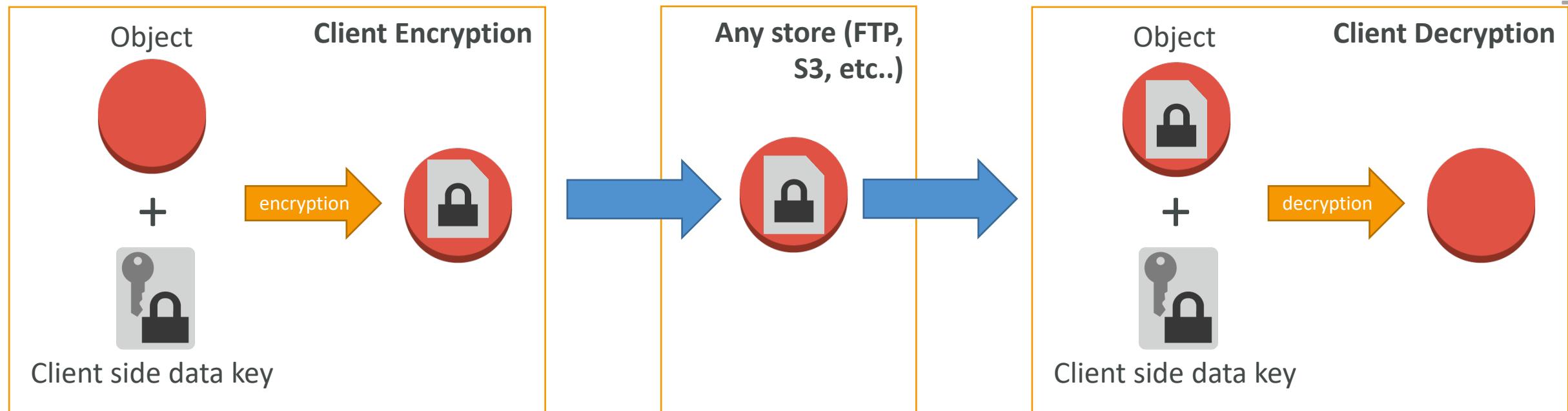
- Data is encrypted after being received by the server
- Data is decrypted before being sent
- It is stored in an encrypted form thanks to a key (usually a data key)
- The encryption / decryption keys must be managed somewhere and the server must have access to it



# Why encryption?

## Client side encryption

- Data is encrypted by the client and never decrypted by the server
- Data will be decrypted by a receiving client
- The server should not be able to decrypt the data
- Could leverage Envelope Encryption





# AWS KMS (Key Management Service)

- Anytime you hear “encryption” for an AWS service, it’s most likely KMS
- Easy way to control access to your data, AWS manages keys for us
- Fully integrated with IAM for authorization
- Seamlessly integrated into:
  - Amazon EBS: encrypt volumes
  - Amazon S3: Server side encryption of objects
  - Amazon Redshift: encryption of data
  - Amazon RDS: encryption of data
  - Amazon SSM: Parameter store
  - Etc...
- But you can also use the CLI / SDK

# KMS – Customer Master Key (CMK) Types

- **Symmetric (AES-256 keys)**
  - First offering of KMS, single encryption key that is used to Encrypt and Decrypt
  - AWS services that are integrated with KMS use Symmetric CMKs
  - Necessary for envelope encryption
  - You never get access to the Key unencrypted (must call KMS API to use)
- **Asymmetric (RSA & ECC key pairs)**
  - Public (Encrypt) and Private Key (Decrypt) pair
  - Used for Encrypt/Decrypt, or Sign/Verify operations
  - The public key is downloadable, but you can't access the Private Key unencrypted
  - Use case: encryption outside of AWS by users who can't call the KMS API

# AWS KMS (Key Management Service)

- Able to fully manage the keys & policies:
  - Create
  - Rotation policies
  - Disable
  - Enable
- Able to audit key usage (using CloudTrail)
- Three types of Customer Master Keys (CMK):
  - AWS Managed Service Default CMK: **free**
  - User Keys created in KMS: **\$1 / month**
  - User Keys imported (must be 256-bit symmetric key): **\$1 / month**
- + pay for API call to KMS (**\$0.03 / 10000 calls**)

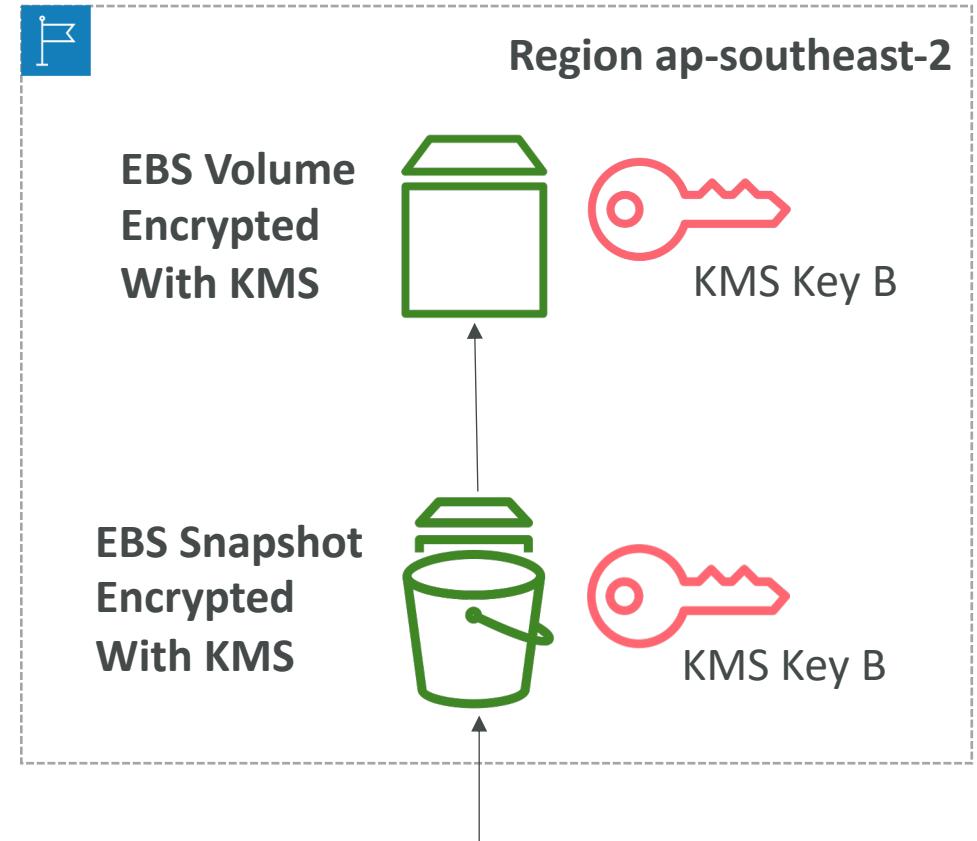
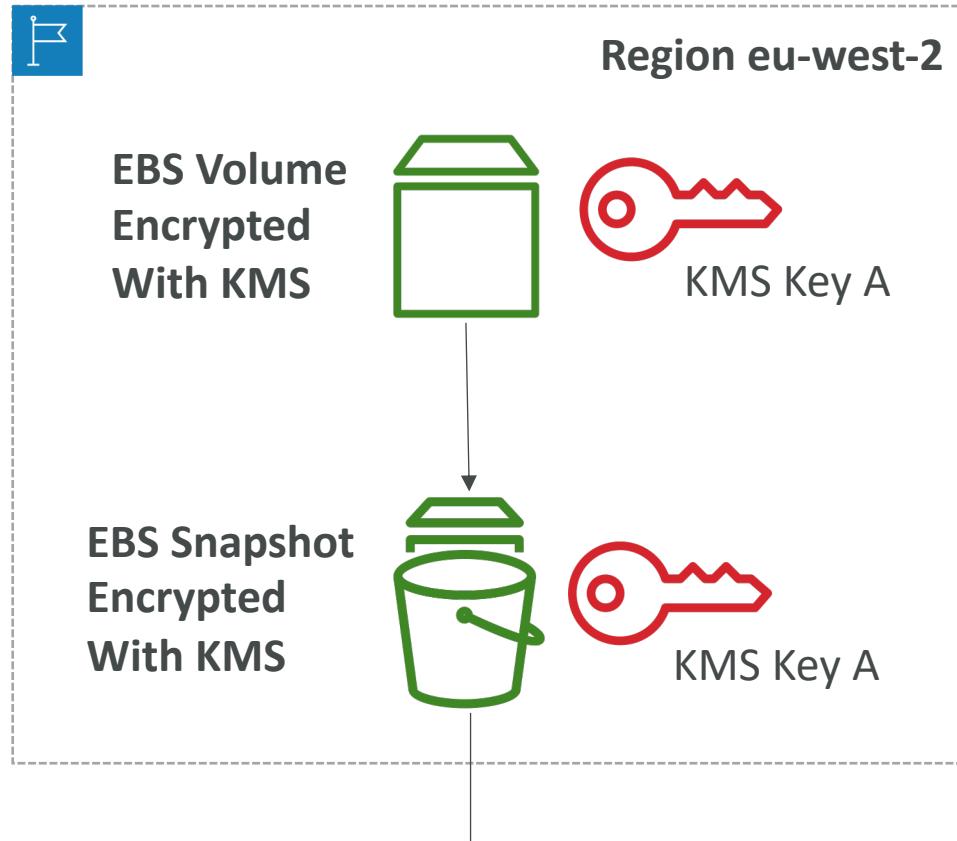
# AWS KMS 101

- Anytime you need to share sensitive information... use KMS
  - Database passwords
  - Credentials to external service
  - Private Key of SSL certificates
- The value in KMS is that the CMK used to encrypt data can never be retrieved by the user, and the CMK can be rotated for extra security

# AWS KMS 101

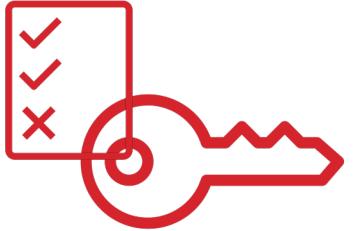
- Never ever store your secrets in plaintext, especially in your code!
- Encrypted secrets can be stored in the code / environment variables
- KMS can only help in encrypting up to 4KB of data per call
- If data > 4 KB, use envelope encryption
- To give access to KMS to someone:
  - Make sure the Key Policy allows the user
  - Make sure the IAM Policy allows the API calls

# Copying Snapshots across regions



KMS ReEncrypt with KMS Key B

# KMS Key Policies



- Control access to KMS keys, “similar” to S3 bucket policies
- Difference: you cannot control access without them
- **Default KMS Key Policy:**
  - Created if you don't provide a specific KMS Key Policy
  - Complete access to the key to the root user = entire AWS account
  - Gives access to the IAM policies to the KMS key
- **Custom KMS Key Policy:**
  - Define users, roles that can access the KMS key
  - Define who can administer the key
  - Useful for cross-account access of your KMS key

# Copying Snapshots across accounts

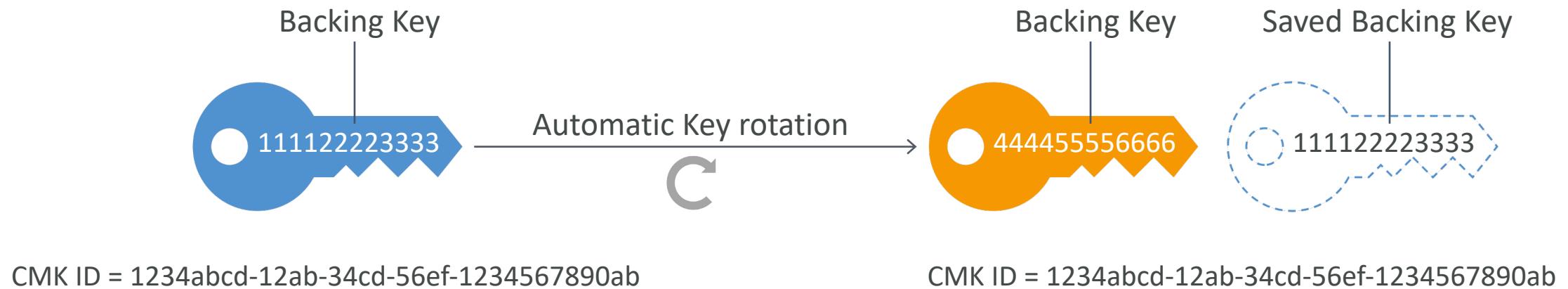
1. Create a Snapshot, encrypted with your own CMK
2. Attach a KMS Key Policy to authorize cross-account access
3. Share the encrypted snapshot
4. (in target) Create a copy of the Snapshot, encrypt it with a KMS Key in your account
5. Create a volume from the snapshot

```
{  
    "Sid": "Allow use of the key with destination account",  
    "Effect": "Allow",  
    "Principal": {  
        "AWS": "arn:aws:iam::TARGET-ACCOUNT-ID:role/ROLENAMESPACE"  
    },  
    "Action": [  
        "kms:Decrypt",  
        "kms>CreateGrant"  
    ],  
    "Resource": "*",  
    "Condition": {  
        "StringEquals": {  
            "kms:ViaService": "ec2.REGION.amazonaws.com",  
            "kms:CallerAccount": "TARGET-ACCOUNT-ID"  
        }  
    }  
}
```

KMS Key Policy

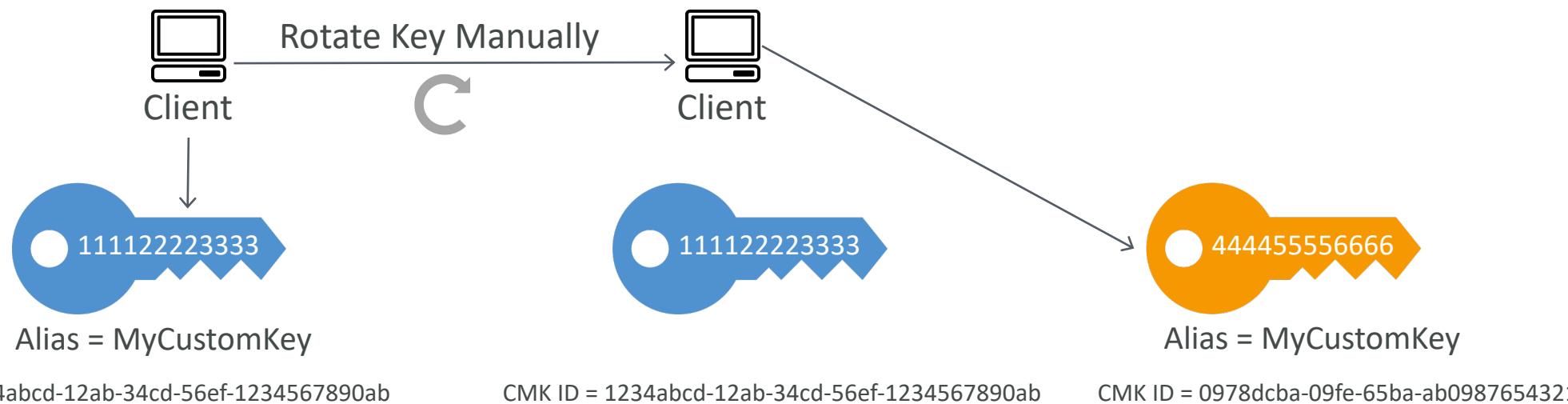
# KMS Automatic Key Rotation

- For Customer-managed CMK (not AWS managed CMK)
- If enabled: automatic key rotation happens every 1 year
- Previous key is kept active so you can decrypt old data
- New Key has the same CMK ID (only the backing key is changed)



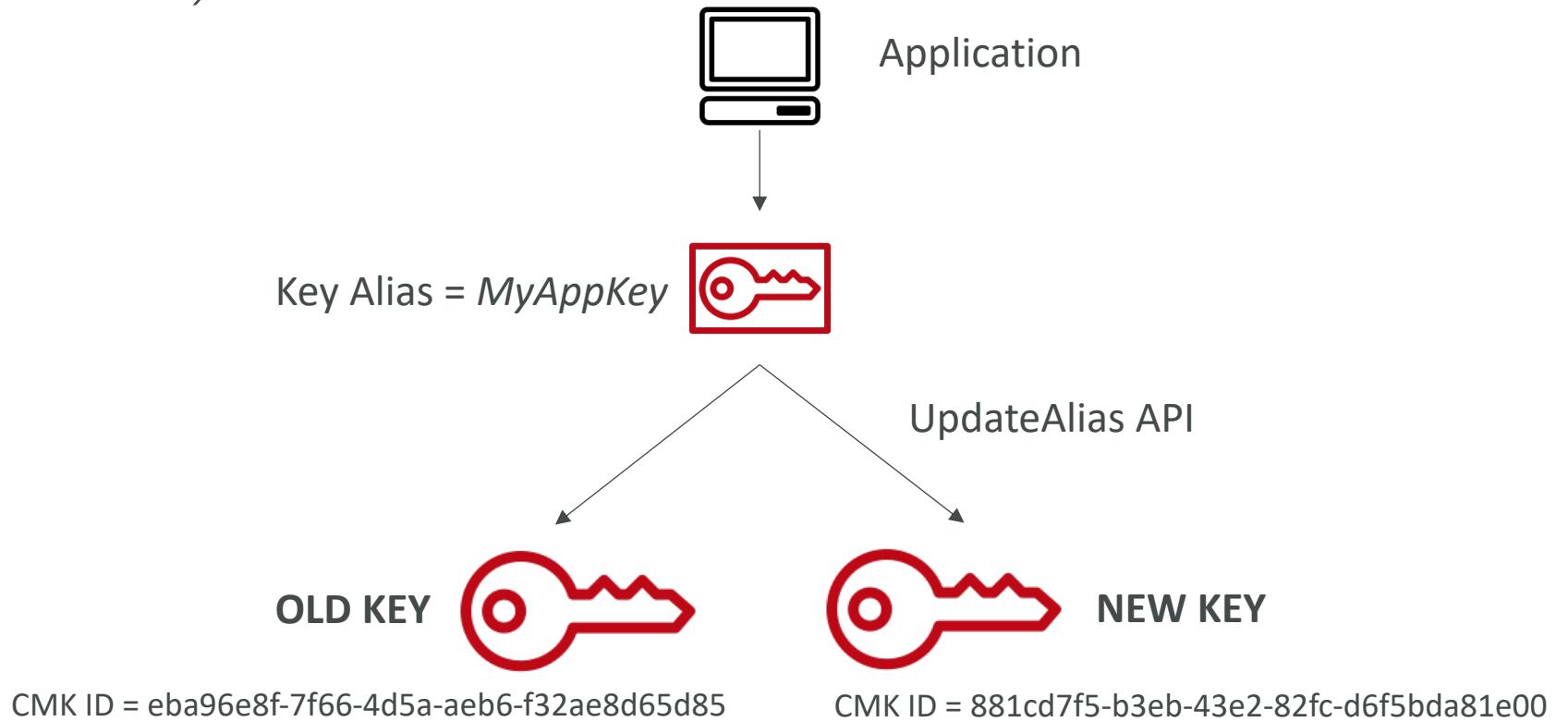
# KMS Manual Key Rotation

- When you want to rotate key every 90 days, 180 days, etc...
- New Key has a different CMK ID
- Keep the previous key active so you can decrypt old data
- Better to use aliases in this case (to hide the change of key for the application)
- Good solution to rotate CMK that are not eligible for automatic rotation (like asymmetric CMK)



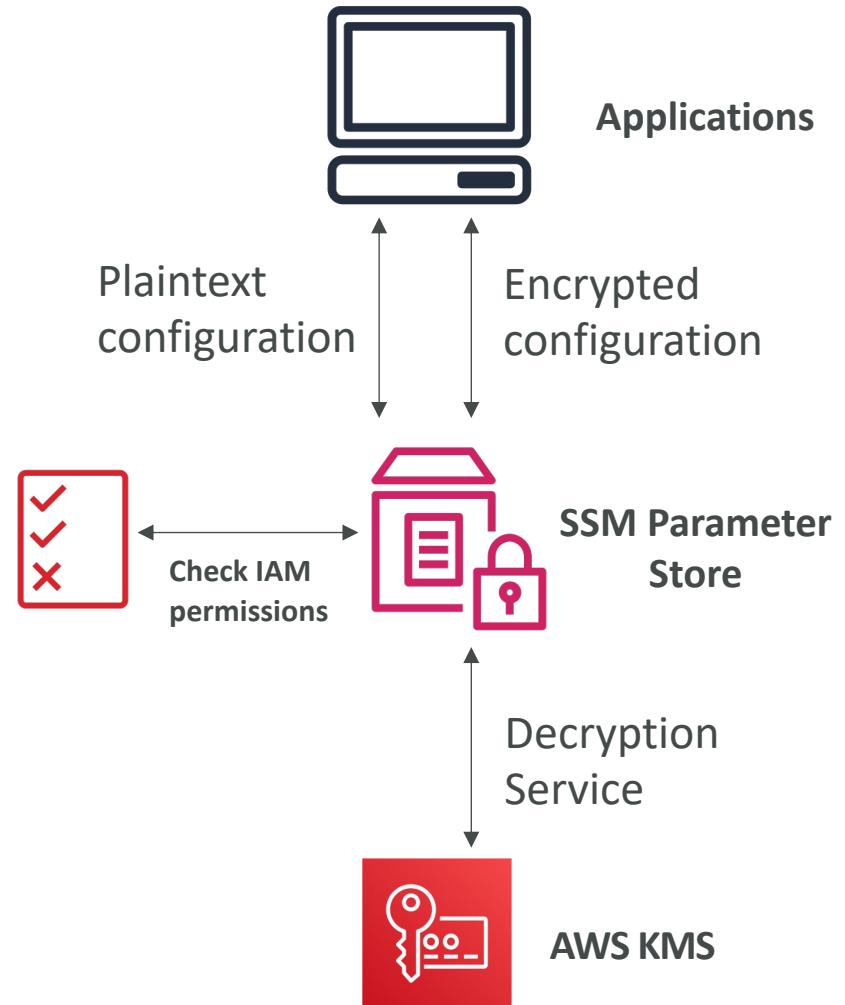
# KMS Alias Updating

- Better to use aliases in this case (to hide the change of key for the application)



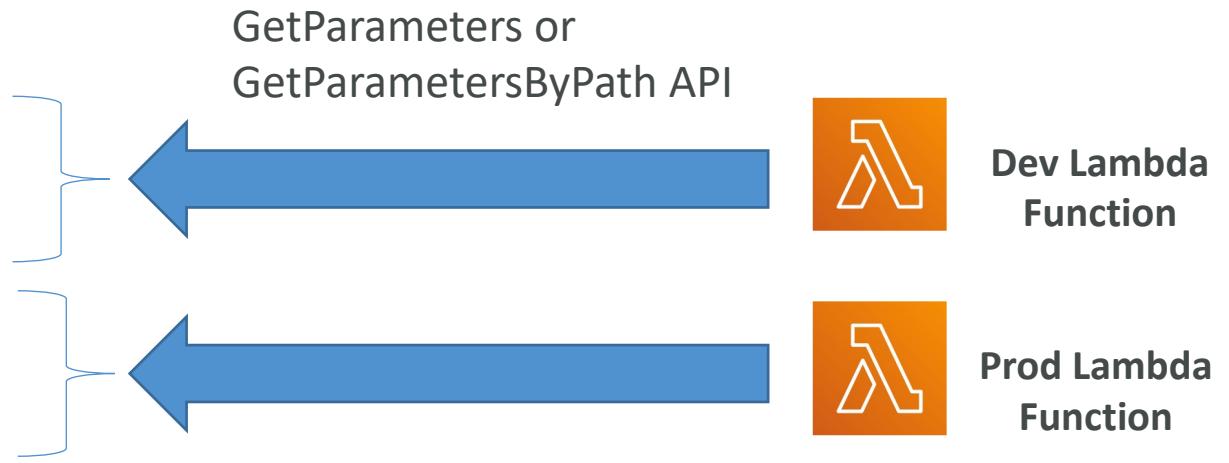
# SSM Parameter Store

- Secure storage for configuration and secrets
- Optional Seamless Encryption using KMS
- Serverless, scalable, durable, easy SDK
- Version tracking of configurations / secrets
- Configuration management using path & IAM
- Notifications with CloudWatch Events
- Integration with CloudFormation



# SSM Parameter Store Hierarchy

- /my-department/
  - my-app/
    - dev/
      - db-url
      - db-password
    - prod/
      - db-url
      - db-password
  - other-app/
  - /other-department/
  - /aws/reference/secretsmanager/secret\_ID\_in\_Secrets\_Manager
  - /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86\_64-gp2



# Standard and advanced parameter tiers

	Standard	Advanced
Total number of parameters allowed (per AWS account and Region)	10,000	100,000
Maximum size of a parameter value	4 KB	8 KB
Parameter policies available	No	Yes
Cost	No additional charge	Charges apply
Storage Pricing	Free	\$0.05 per advanced parameter per month
API Interaction Pricing (higher throughput = up to 1000 Transactions per second)	Standard Throughput: free Higher Throughput: \$0.05 per 10,000 API interactions	Standard Throughput: \$0.05 per 10,000 API interactions Higher Throughput: \$0.05 per 10,000 API interactions

# Parameters Policies (for advanced parameters)

- Allow to assign a TTL to a parameter (expiration date) to force updating or deleting sensitive data such as passwords
- Can assign multiple policies at a time

**Expiration (to delete a parameter)**

```
{  
  "Type": "Expiration",  
  "Version": "1.0",  
  "Attributes": {  
    "Timestamp": "2020-12-02T21:34:33.000Z"  
  }  
}
```

**ExpirationNotification (CW Events)**

```
{  
  "Type": "ExpirationNotification",  
  "Version": "1.0",  
  "Attributes": {  
    "Before": "15",  
    "Unit": "Days"  
  }  
}
```

**NoChangeNotification (CW Events)**

```
{  
  "Type": "NoChangeNotification",  
  "Version": "1.0",  
  "Attributes": {  
    "After": "20",  
    "Unit": "Days"  
  }  
}
```

# AWS Secrets Manager



- Newer service, meant for storing secrets
- Capability to force **rotation of secrets** every X days
- Automate generation of secrets on rotation (uses Lambda)
- Integration with **Amazon RDS** (MySQL, PostgreSQL, Aurora)
- Secrets are encrypted using KMS
- Mostly meant for RDS integration

# AWS Shield



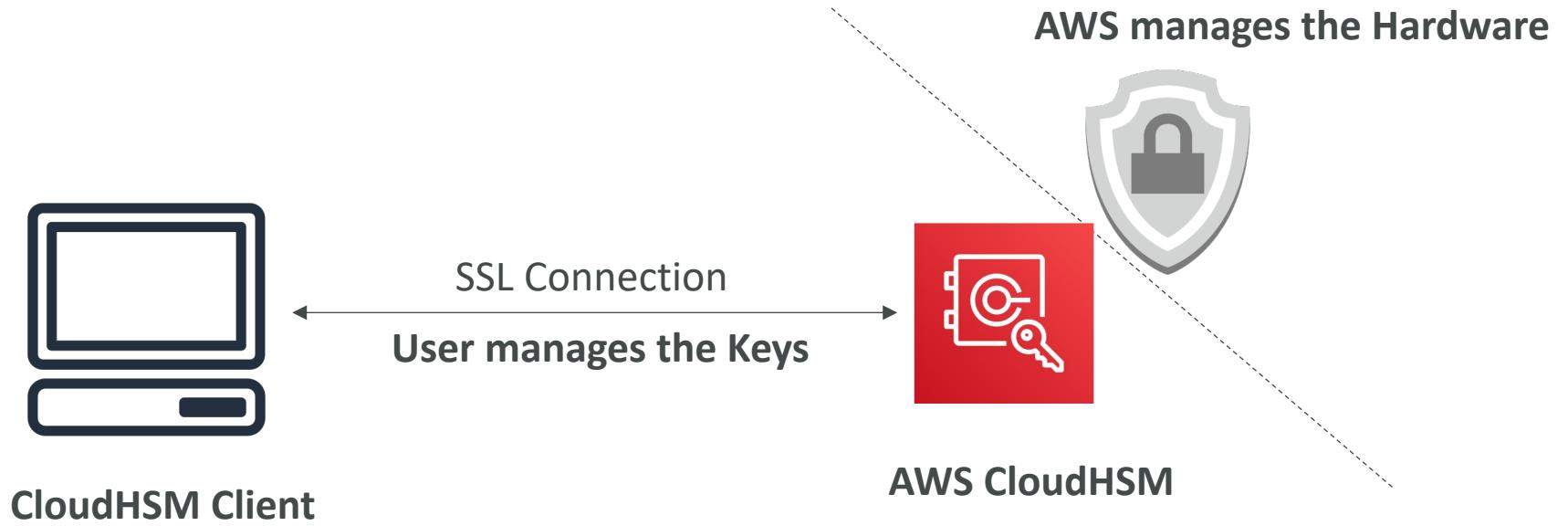
- AWS Shield Standard:
  - Free service that is activated for every AWS customer
  - Provides protection from attacks such as SYN/UDP Floods, Reflection attacks and other layer 3/layer 4 attacks
- AWS Shield Advanced:
  - Optional DDoS mitigation service (\$3,000 per month per organization)
  - Protect against more sophisticated attack on [Amazon EC2](#), [Elastic Load Balancing \(ELB\)](#), [Amazon CloudFront](#), [AWS Global Accelerator](#), and [Route 53](#)
  - 24/7 access to AWS DDoS response team (DRP)
  - Protect against higher fees during usage spikes due to DDoS

# CloudHSM



- KMS => AWS manages the software for encryption
- CloudHSM => AWS provisions encryption **hardware**
- Dedicated Hardware (HSM = Hardware Security Module)
- You manage your own encryption keys entirely (not AWS)
- HSM device is tamper resistant, FIPS 140-2 Level 3 compliance
- Supports both symmetric and **asymmetric** encryption (SSL/TLS keys)
- No free tier available
- Must use the CloudHSM Client Software
- Redshift supports CloudHSM for database encryption and key management
- **Good option to use with SSE-C encryption**

# CloudHSM Diagram



## IAM permissions:

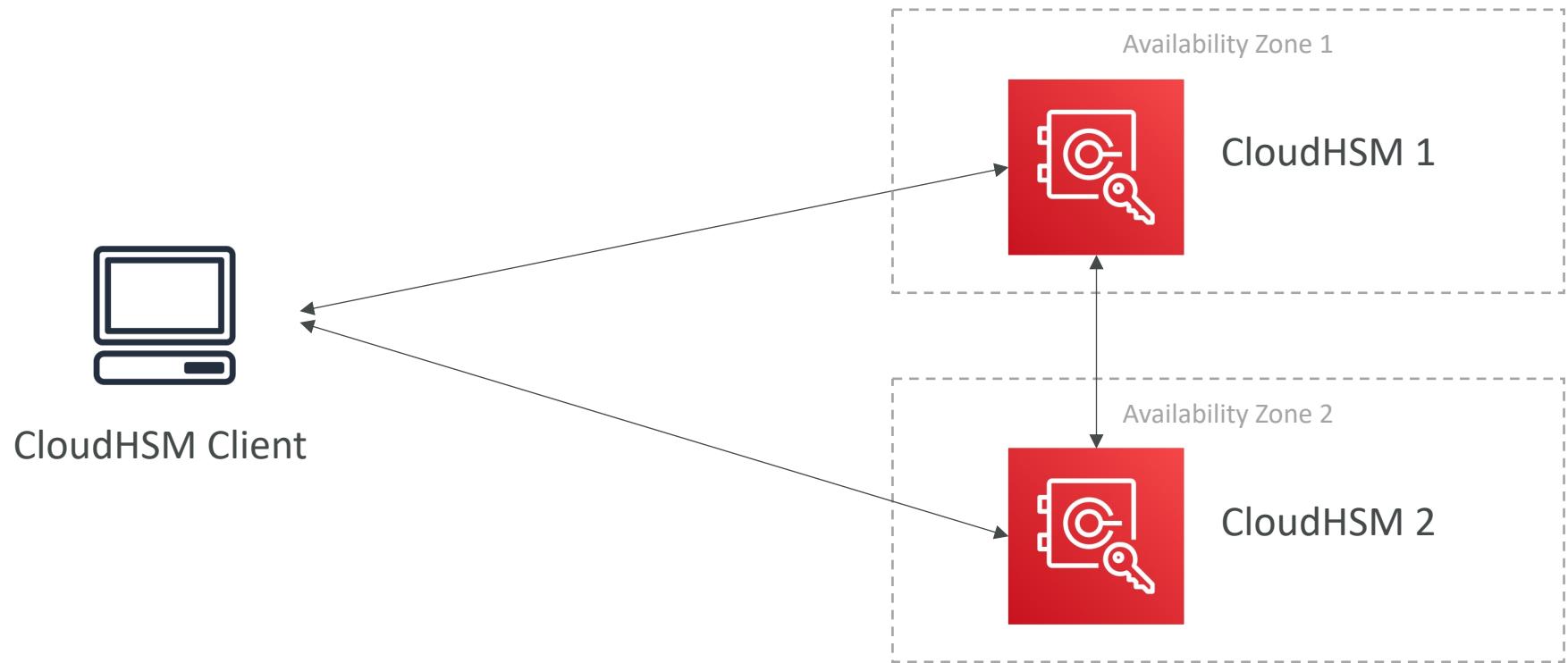
- CRUD an HSM Cluster

## CloudHSM Software:

- Manage the Keys
- Manage the Users

# CloudHSM – High Availability

- CloudHSM clusters are spread across Multi AZ (HA)
- Great for availability and durability



# CloudHSM vs. KMS

Feature	AWS KMS	AWS CloudHSM
Tenancy	Multi-Tenant	Single-Tenant
Standard	FIPS 140-2 Level 2	FIPS 140-2 Level 3
Master Keys	<ul style="list-style-type: none"><li>• AWS Owned CMK</li><li>• AWS Managed CMK</li><li>• Customer Managed CMK</li></ul>	Customer Managed CMK
Key Types	<ul style="list-style-type: none"><li>• Symmetric</li><li>• Asymmetric</li><li>• Digital Signing</li></ul>	<ul style="list-style-type: none"><li>• Symmetric</li><li>• Asymmetric</li><li>• Digital Signing &amp; Hashing</li></ul>
Key Accessibility	Accessible in multiple AWS regions (can't access keys outside the region it's created in)	<ul style="list-style-type: none"><li>• Deployed and managed in a VPC</li><li>• Can be shared across VPCs (VPC Peering)</li></ul>
Cryptographic Acceleration	None	<ul style="list-style-type: none"><li>• SSL/TLS Acceleration</li><li>• Oracle TDE Acceleration</li></ul>
Access & Authentication	AWS IAM	You create users and manage their permissions

# CloudHSM vs. KMS

Feature	AWS KMS	AWS CloudHSM
High Availability	AWS Managed Service	Add multiple HSMs over different AZs
Audit Capability	<ul style="list-style-type: none"><li>• CloudTrail</li><li>• CloudWatch</li></ul>	<ul style="list-style-type: none"><li>• CloudTrail</li><li>• CloudWatch</li><li>• MFA support</li></ul>
Free Tier	Yes	No

# AWS WAF – Web Application Firewall



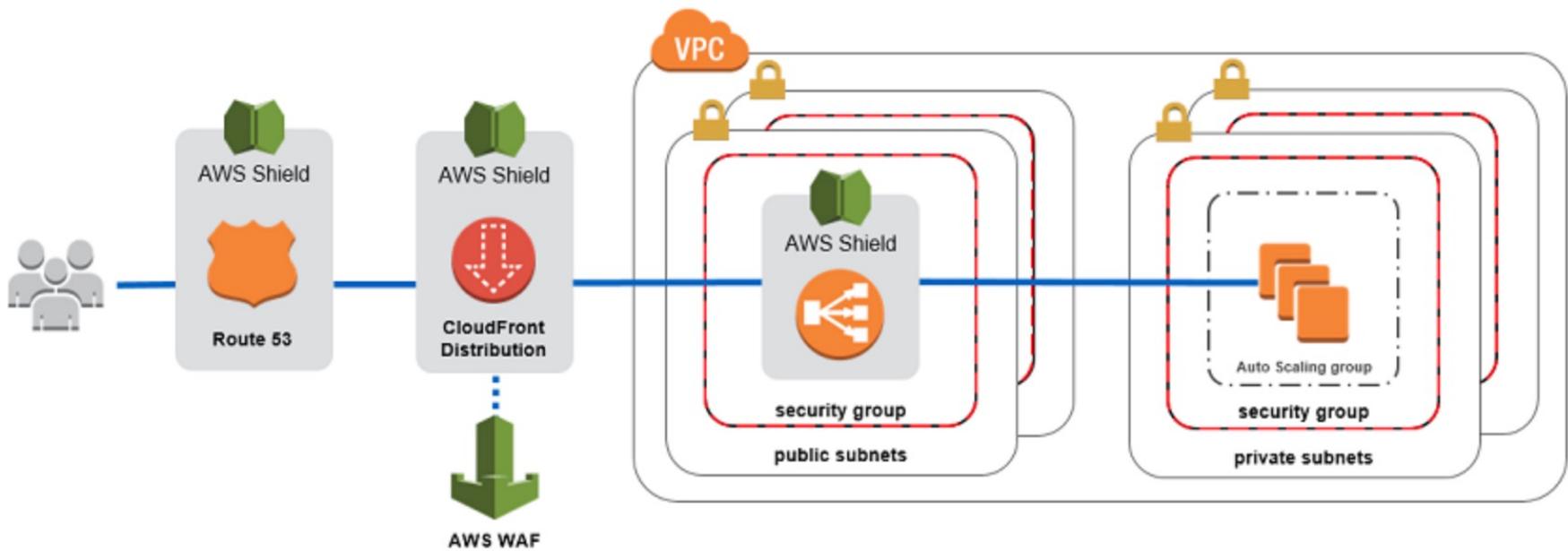
- Protects your web applications from common web exploits (Layer 7)
- Layer 7 is HTTP (vs Layer 4 is TCP)
- Deploy on Application Load Balancer, API Gateway, CloudFront
- Define Web ACL (Web Access Control List):
  - Rules can include: IP addresses, HTTP headers, HTTP body, or URI strings
  - Protects from common attack - SQL injection and Cross-Site Scripting (XSS)
  - Size constraints, geo-match (block countries)
  - Rate-based rules (to count occurrences of events) – for DDoS protection

# AWS Firewall Manager



- Manage rules in all accounts of an AWS Organization
- Common set of security rules
- WAF rules (Application Load Balancer, API Gateways, CloudFront)
- AWS Shield Advanced (ALB, CLB, Elastic IP, CloudFront)
- Security Groups for EC2 and ENI resources in VPC

# Sample Reference Architecture for DDoS Protection



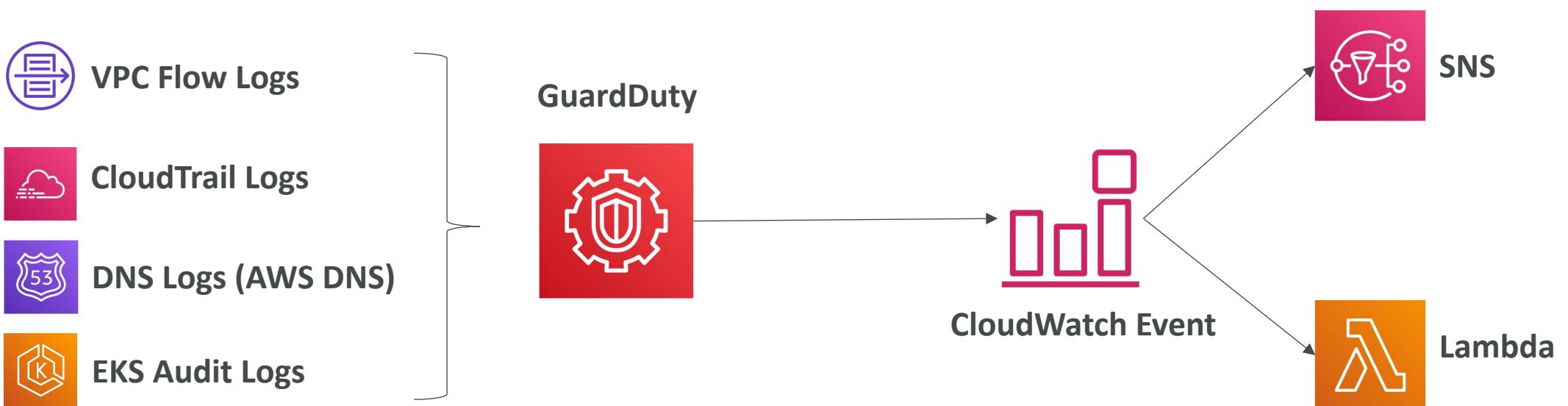
<https://aws.amazon.com/answers/networking/aws-ddos-attack-mitigation/>



# Amazon GuardDuty

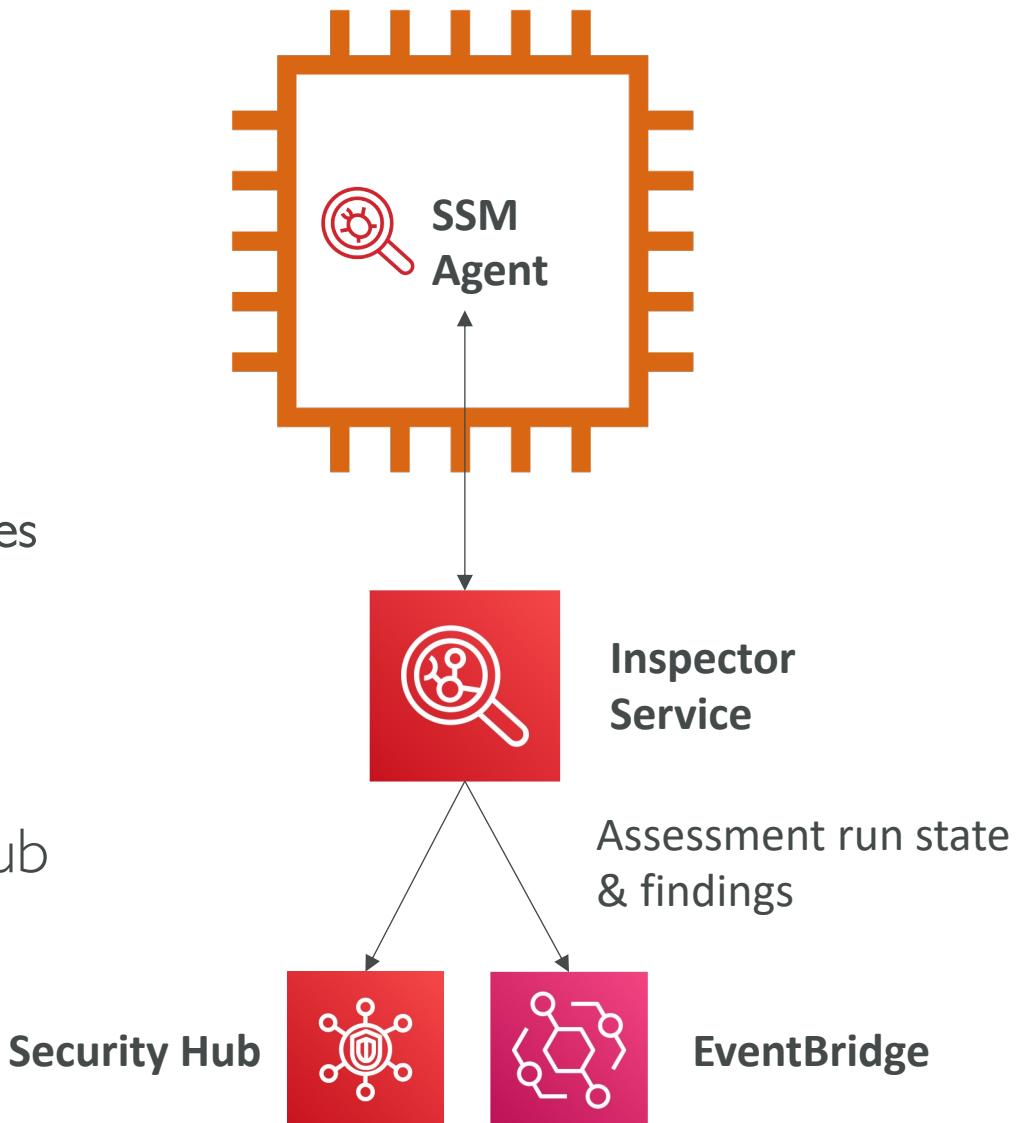
- Intelligent Threat discovery to Protect AWS Account
- Uses Machine Learning algorithms, anomaly detection, 3<sup>rd</sup> party data
- One click to enable (30 days trial), no need to install software
- Input data includes:
  - **CloudTrail Events Logs** – unusual API calls, unauthorized deployments
    - CloudTrail Management Events – create VPC subnet, create trail, ...
    - CloudTrail S3 Data Events – get object, list objects, delete object, ...
  - **VPC Flow Logs** – unusual internal traffic, unusual IP address
  - **DNS Logs** – compromised EC2 instances sending encoded data within DNS queries
  - **Kubernetes Audit Logs** – suspicious activities and potential EKS cluster compromises
- Can setup **CloudWatch Event rules** to be notified in case of findings
- CloudWatch Events rules can target AWS Lambda or SNS
- **Can protect against CryptoCurrency attacks** (has a dedicated “finding” for it)

# Amazon GuardDuty



# Amazon Inspector

- Automated Security Assessments
- For EC2 instances
  - Leveraging the AWS System Manager (SSM) agent
  - Analyze against unintended network accessibility
  - Analyze the running OS against known vulnerabilities
- For Containers push to Amazon ECR
  - Assessment of containers as they are pushed
- Reporting & integration with AWS Security Hub
- Send findings to Amazon Event Bridge



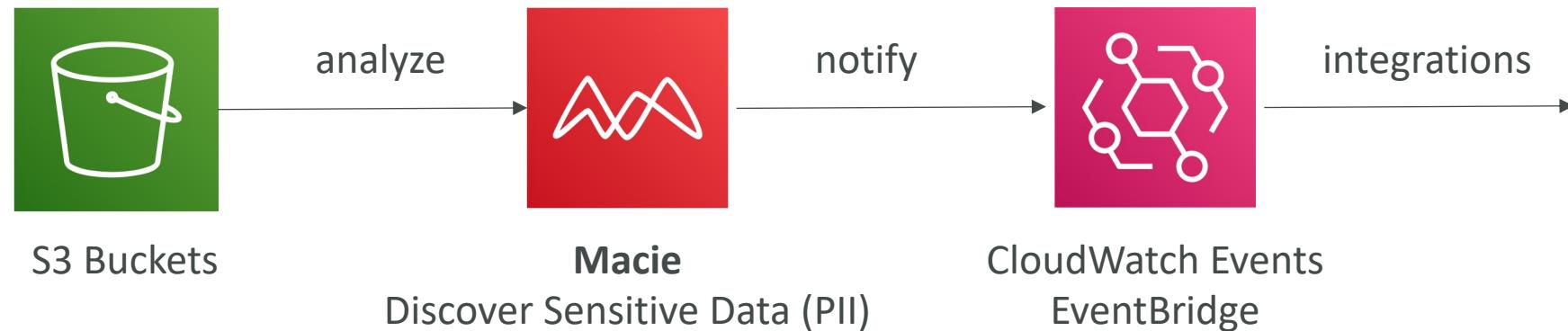
# What does AWS Inspector evaluate?

- Remember: only for EC2 instances and container infrastructure
- Continuous scanning of the infrastructure, only when needed
- Package vulnerabilities (EC2 & ECR) – database of CVE
- Network reachability (EC2)
- A risk score is associated with all vulnerabilities for prioritization

# Amazon Macie



- Amazon Macie is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover and protect your sensitive data in AWS.
- Macie helps identify and alert you to sensitive data, such as personally identifiable information (PII)



# AWS Shared Responsibility Model

- AWS responsibility - Security **of** the Cloud
  - Protecting infrastructure (hardware, software, facilities, and networking) that runs all the AWS services
  - Managed services like S3, DynamoDB, RDS, etc.
- Customer responsibility - Security **in** the Cloud
  - For EC2 instance, customer is responsible for management of the guest OS (including security patches and updates), firewall & network configuration, IAM
  - Encrypting application data
- Shared controls:
  - Patch Management, Configuration Management, Awareness & Training

# Example, for RDS



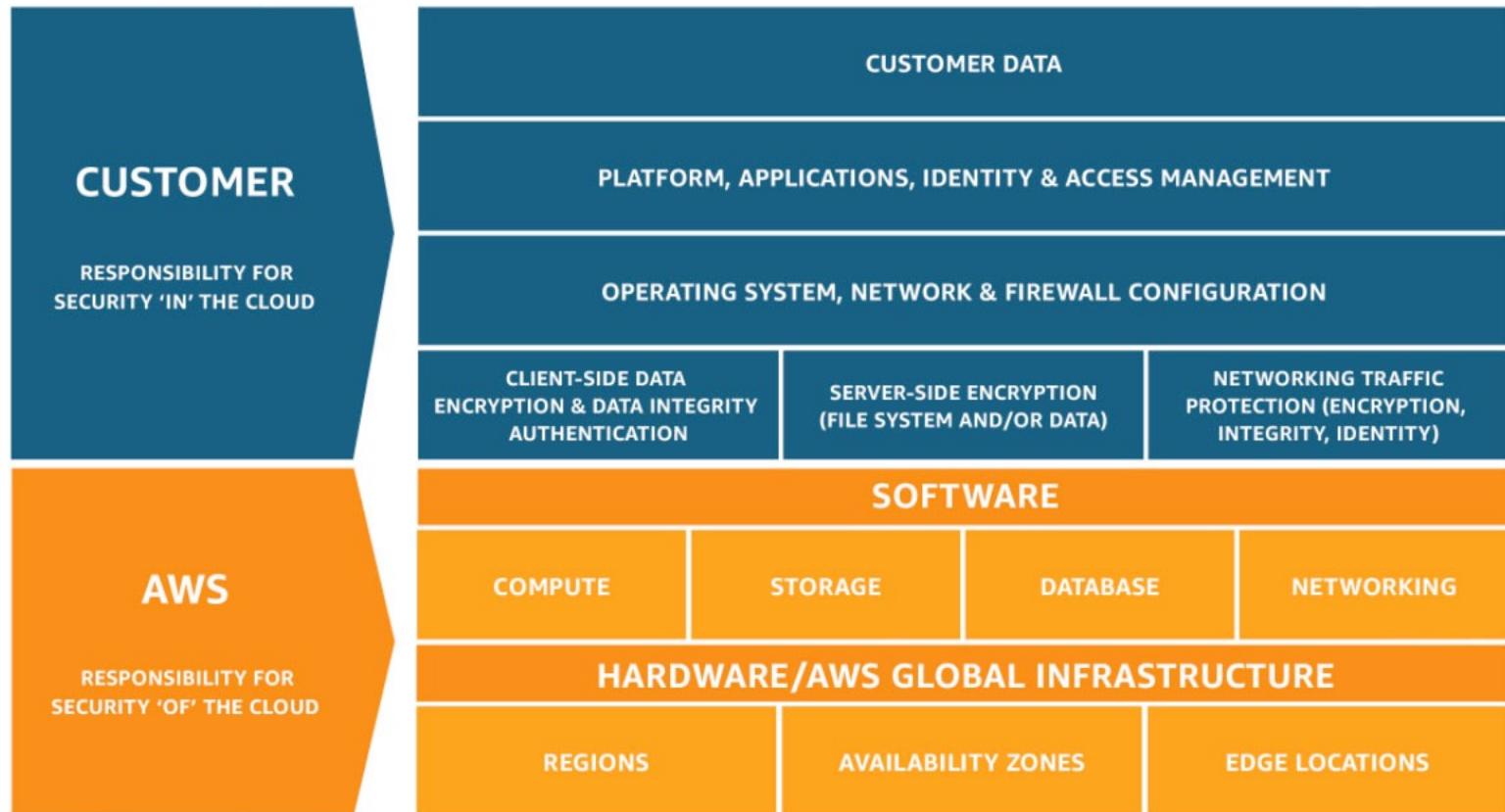
- AWS responsibility:
  - Manage the underlying EC2 instance, disable SSH access
  - Automated DB patching
  - Automated OS patching
  - Audit the underlying instance and disks & guarantee it functions
- Your responsibility:
  - Check the ports / IP / security group inbound rules in DB's SG
  - In-database user creation and permissions
  - Creating a database with or without public access
  - Ensure parameter groups or DB is configured to only allow SSL connections
  - Database encryption setting



# Example, for S3

- AWS responsibility:
  - Guarantee you get unlimited storage
  - Guarantee you get encryption
  - Ensure separation of the data between different customers
  - Ensure AWS employees can't access your data
- Your responsibility:
  - Bucket configuration
  - Bucket policy / public setting
  - IAM user and roles
  - Enabling encryption

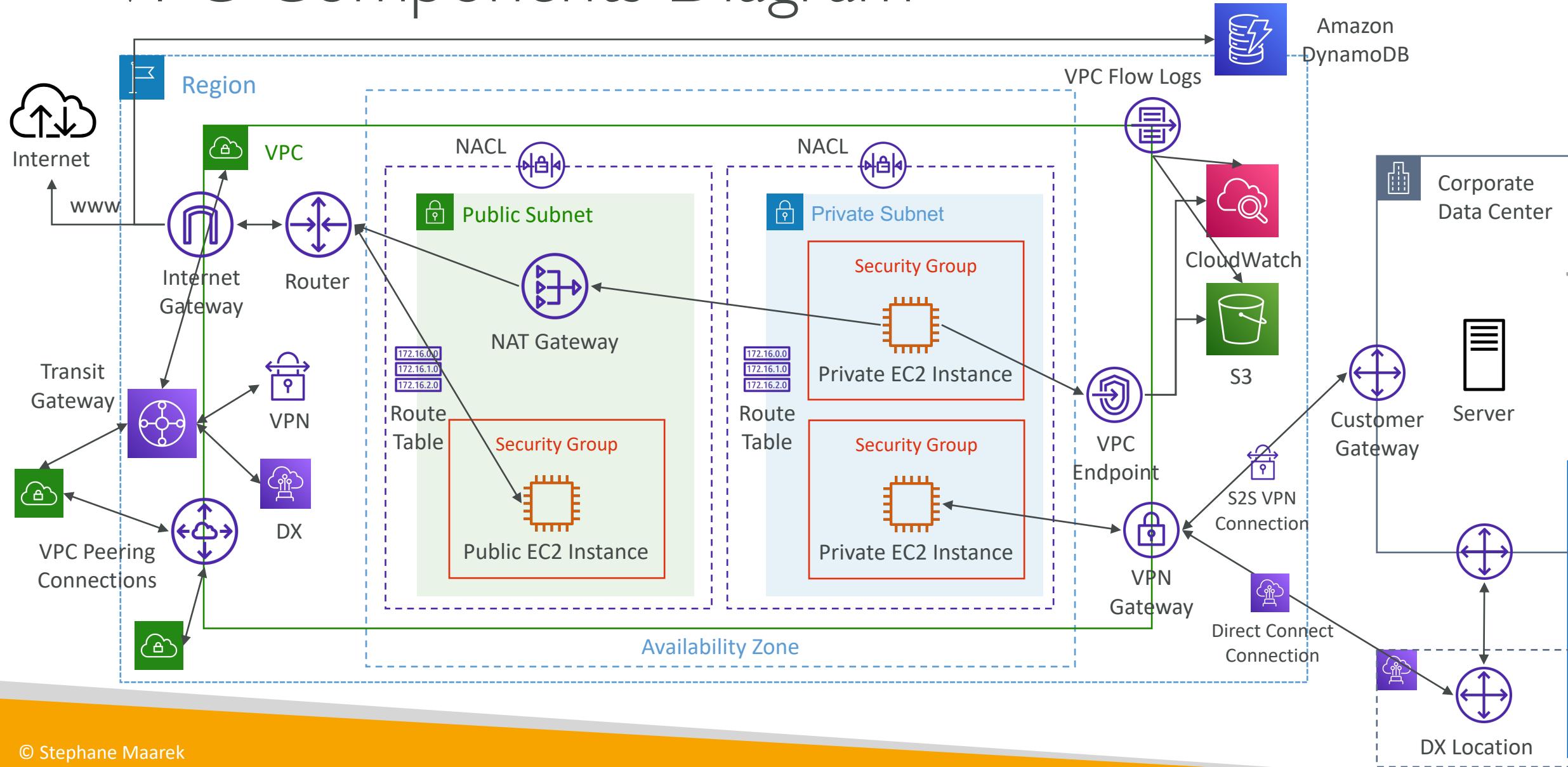
# Shared Responsibility Model diagram



<https://aws.amazon.com/compliance/shared-responsibility-model/>

# Virtual Private Cloud (VPC)

# VPC Components Diagram



# Understanding CIDR – IPv4

- Classless Inter-Domain Routing – a method for allocating IP addresses
- Used in **Security Groups** rules and AWS networking in general

IP version	Type	Protocol	Port range	Source	Description
IPv4	SSH	TCP	22	122.149.196.85/32	-
IPv4	HTTP	TCP	80	0.0.0.0/0	-

- They help to define an IP address range:
  - We've seen WW.XX.YY.ZZ/32 => one IP
  - We've seen 0.0.0.0/0 => all IPs
  - But we can define: 192.168.0.0/26 => 192.168.0.0 – 192.168.0.63 (64 IP addresses)

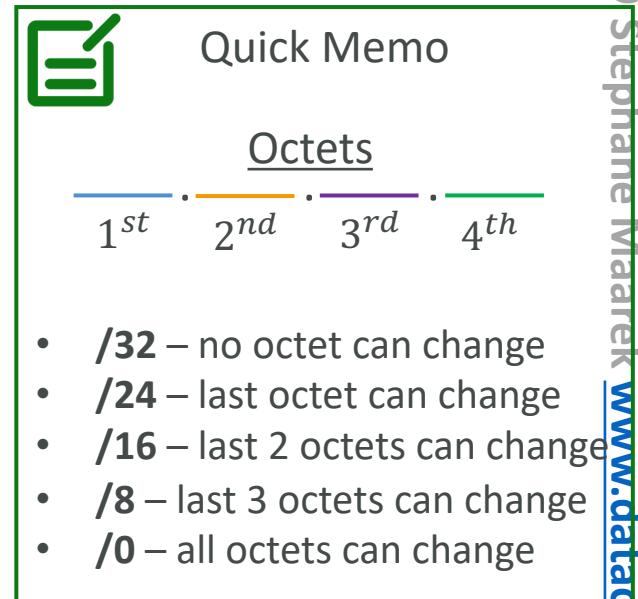
# Understanding CIDR – IPv4

- A CIDR consists of two components
- Base IP
  - Represents an IP contained in the range (XX.XX.XX.XX)
  - Example: 10.0.0.0, 192.168.0.0, ...
- Subnet Mask
  - Defines how many bits can change in the IP
  - Example: /0, /24, /32
  - Can take two forms:
    - /8  $\Leftrightarrow$  255.0.0.0
    - /16  $\Leftrightarrow$  255.255.0.0
    - /24  $\Leftrightarrow$  255.255.255.0
    - /32  $\Leftrightarrow$  255.255.255.255

# Understanding CIDR – Subnet Mask

- The Subnet Mask basically allows part of the underlying IP to get additional next values from the base IP

192	. 168 . 0 . 0 /32 => allows for <b>1</b> IP ( $2^0$ )	→ 192.168.0.0
192	. 168 . 0 . 0 /31 => allows for <b>2</b> IP ( $2^1$ )	→ 192.168.0.0 -> 192.168.0.1
192	. 168 . 0 . 0 /30 => allows for <b>4</b> IP ( $2^2$ )	→ 192.168.0.0 -> 192.168.0.3
192	. 168 . 0 . 0 /29 => allows for <b>8</b> IP ( $2^3$ )	→ 192.168.0.0 -> 192.168.0.7
192	. 168 . 0 . 0 /28 => allows for <b>16</b> IP ( $2^4$ )	→ 192.168.0.0 -> 192.168.0.15
192	. 168 . 0 . 0 /27 => allows for <b>32</b> IP ( $2^5$ )	→ 192.168.0.0 -> 192.168.0.31
192	. 168 . 0 . 0 /26 => allows for <b>64</b> IP ( $2^6$ )	→ 192.168.0.0 -> 192.168.0.63
192	. 168 . 0 . 0 /25 => allows for <b>128</b> IP ( $2^7$ )	→ 192.168.0.0 -> 192.168.0.127
192	. 168 . 0 . 0 /24 => allows for <b>256</b> IP ( $2^8$ )	→ 192.168.0.0 -> 192.168.0.255
...		
192	. 168 . 0 . 0 /16 => allows for <b>65,536</b> IP ( $2^{16}$ )	→ 192.168.0.0 -> 192.168.255.255
...		
192	. 168 . 0 . 0 /0 => allows for All IPs	→ 0.0.0.0 -> 255.255.255.255



# Understanding CIDR – Little Exercise

- $192.168.0.0/24 = \dots ?$ 
  - $192.168.0.0 - 192.168.0.255$  (256 IPs)
- $192.168.0.0/16 = \dots ?$ 
  - $192.168.0.0 - 192.168.255.255$  (65,536 IPs)
- $134.56.78.123/32 = \dots ?$ 
  - Just  $134.56.78.123$
- $0.0.0.0/0$ 
  - All IPs!
- When in doubt, use this website <https://www.ipaddressguide.com/cidr>

# Public vs. Private IP (IPv4)

- The Internet Assigned Numbers Authority (IANA) established certain blocks of IPv4 addresses for the use of private (LAN) and public (Internet) addresses
- **Private IP** can only allow certain values:
  - 10.0.0 – 10.255.255.255 (10.0.0/8) ↵ in big networks
  - 172.16.0.0 – 172.31.255.255 (172.16.0.0/12) ↵ AWS default VPC in that range
  - 192.168.0.0 – 192.168.255.255 (192.168.0.0/16) ↵ e.g., home networks
- All the rest of the IP addresses on the Internet are Public

# Default VPC Walkthrough

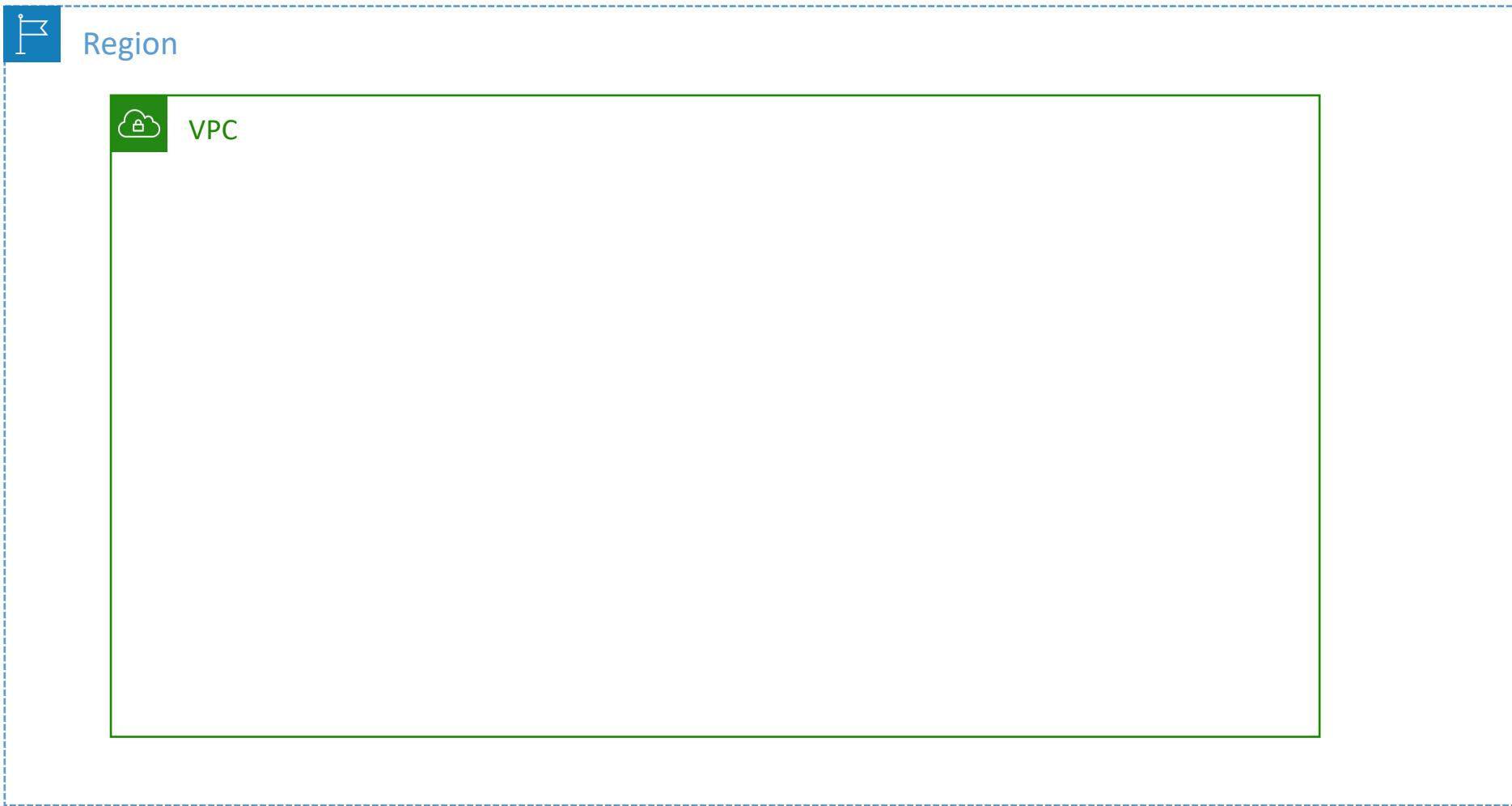
- All new AWS accounts have a default VPC
- New EC2 instances are launched into the default VPC if no subnet is specified
- Default VPC has Internet connectivity and all EC2 instances inside it have public IPv4 addresses
- We also get a public and a private IPv4 DNS names

# VPC in AWS – IPv4

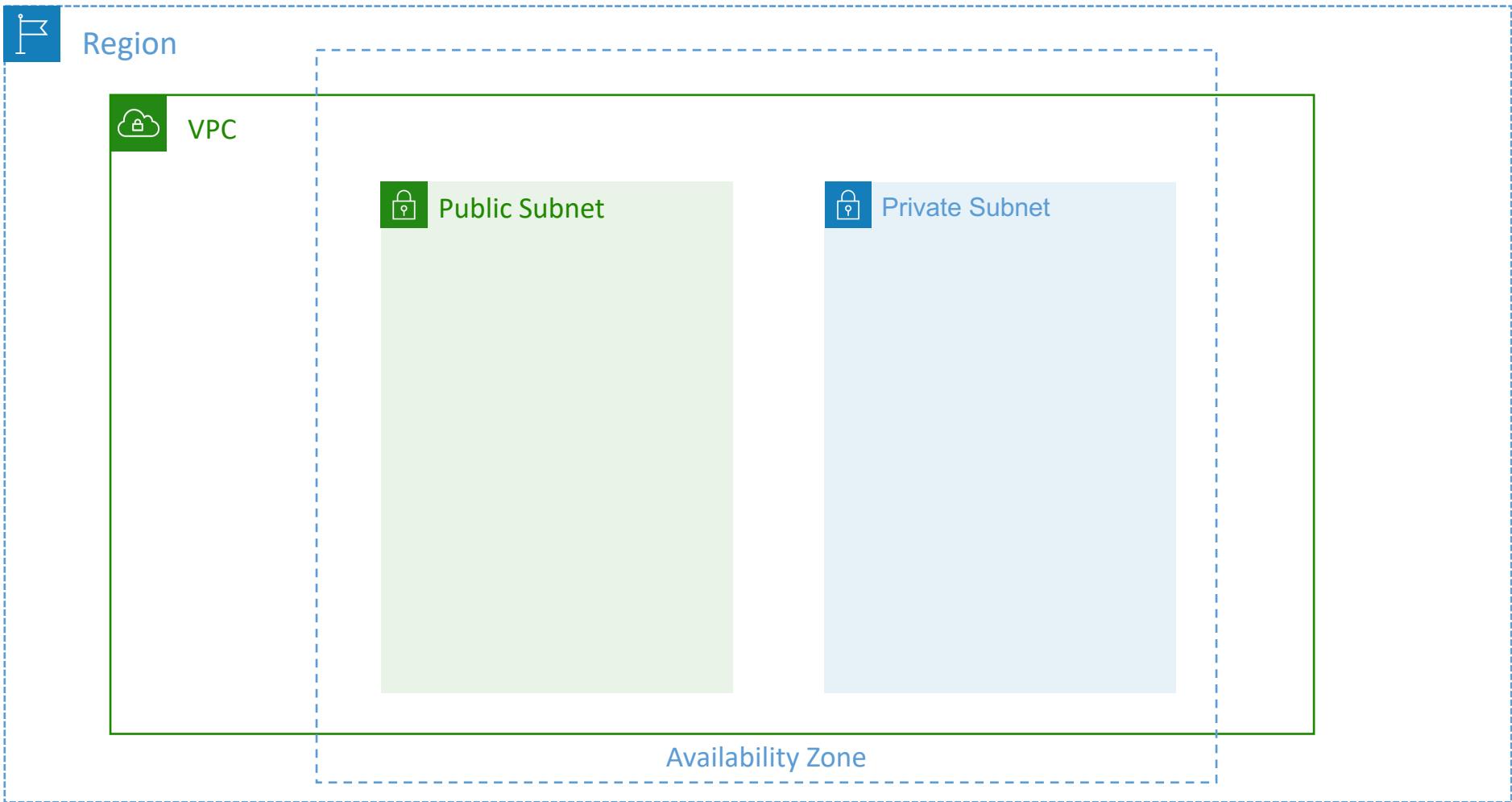


- VPC = Virtual Private Cloud
- You can have multiple VPCs in an AWS region (max. 5 per region – soft limit)
- Max. CIDR per VPC is 5, for each CIDR:
  - Min. size is /28 (16 IP addresses)
  - Max. size is /16 (65536 IP addresses)
- Because VPC is private, only the Private IPv4 ranges are allowed:
  - 10.0.0.0 – 10.255.255.255 (10.0.0.0/8)
  - 172.16.0.0 – 172.31.255.255 (172.16.0.0/12)
  - 192.168.0.0 – 192.168.255.255 (192.168.0.0/16)
- Your VPC CIDR should NOT overlap with your other networks (e.g., corporate)

# State of Hands-on



# Adding Subnets



# VPC – Subnet (IPv4)



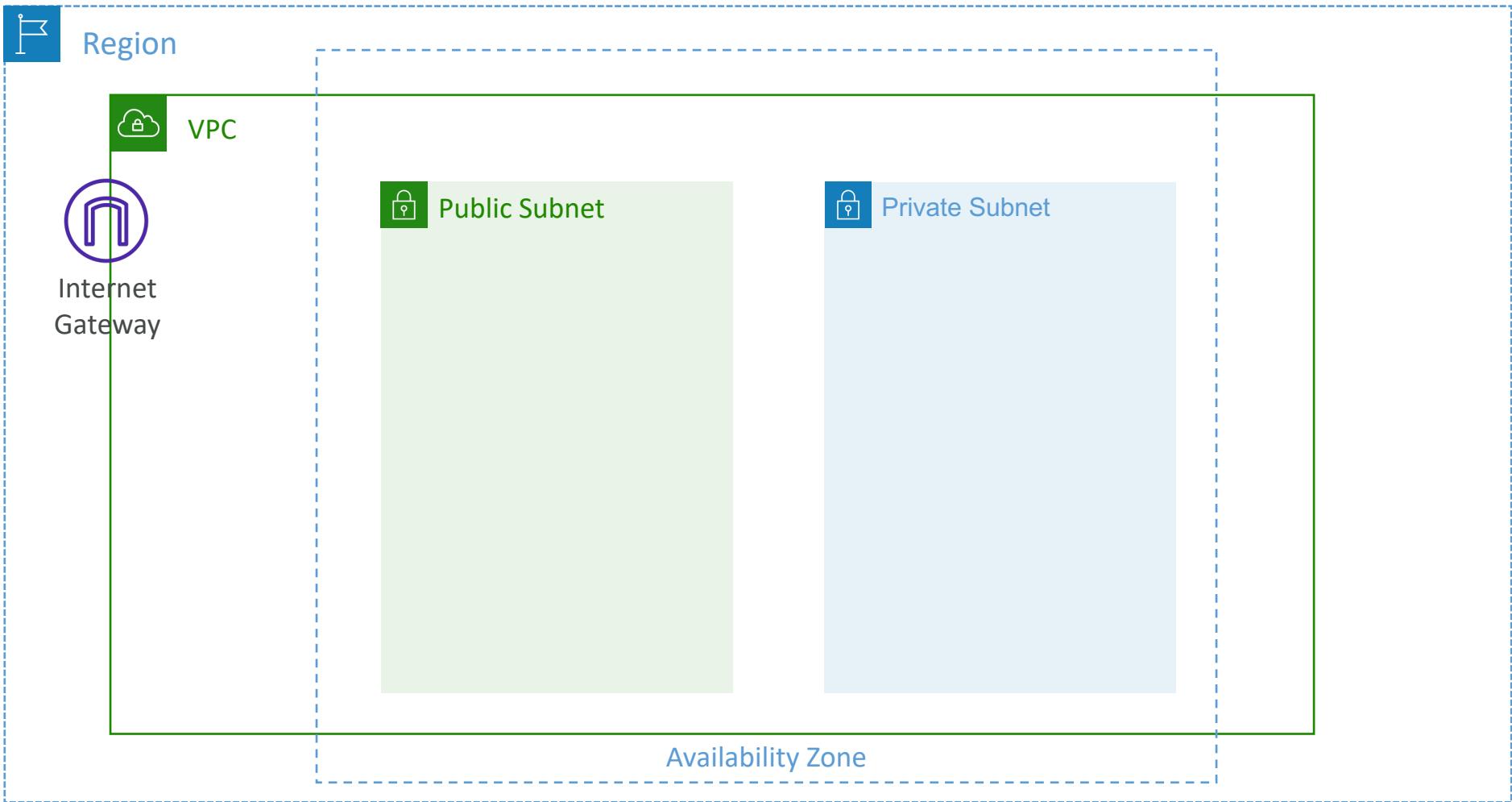
- AWS reserves **5 IP addresses (first 4 & last 1)** in each subnet
- These 5 IP addresses are not available for use and can't be assigned to an EC2 instance
- Example: if CIDR block 10.0.0.0/24, then reserved IP addresses are:
  - 10.0.0.0 – Network Address
  - 10.0.0.1 – reserved by AWS for the VPC router
  - 10.0.0.2 – reserved by AWS for mapping to Amazon-provided DNS
  - 10.0.0.3 – reserved by AWS for future use
  - 10.0.0.255 – Network Broadcast Address. AWS does not support broadcast in a VPC, therefore the address is reserved
- **Exam Tip**, if you need 29 IP addresses for EC2 instances:
  - You can't choose a subnet of size /27 (32 IP addresses,  $32 - 5 = 27 < 29$ )
  - You need to choose a subnet of size /26 (64 IP addresses,  $64 - 5 = 59 > 29$ )



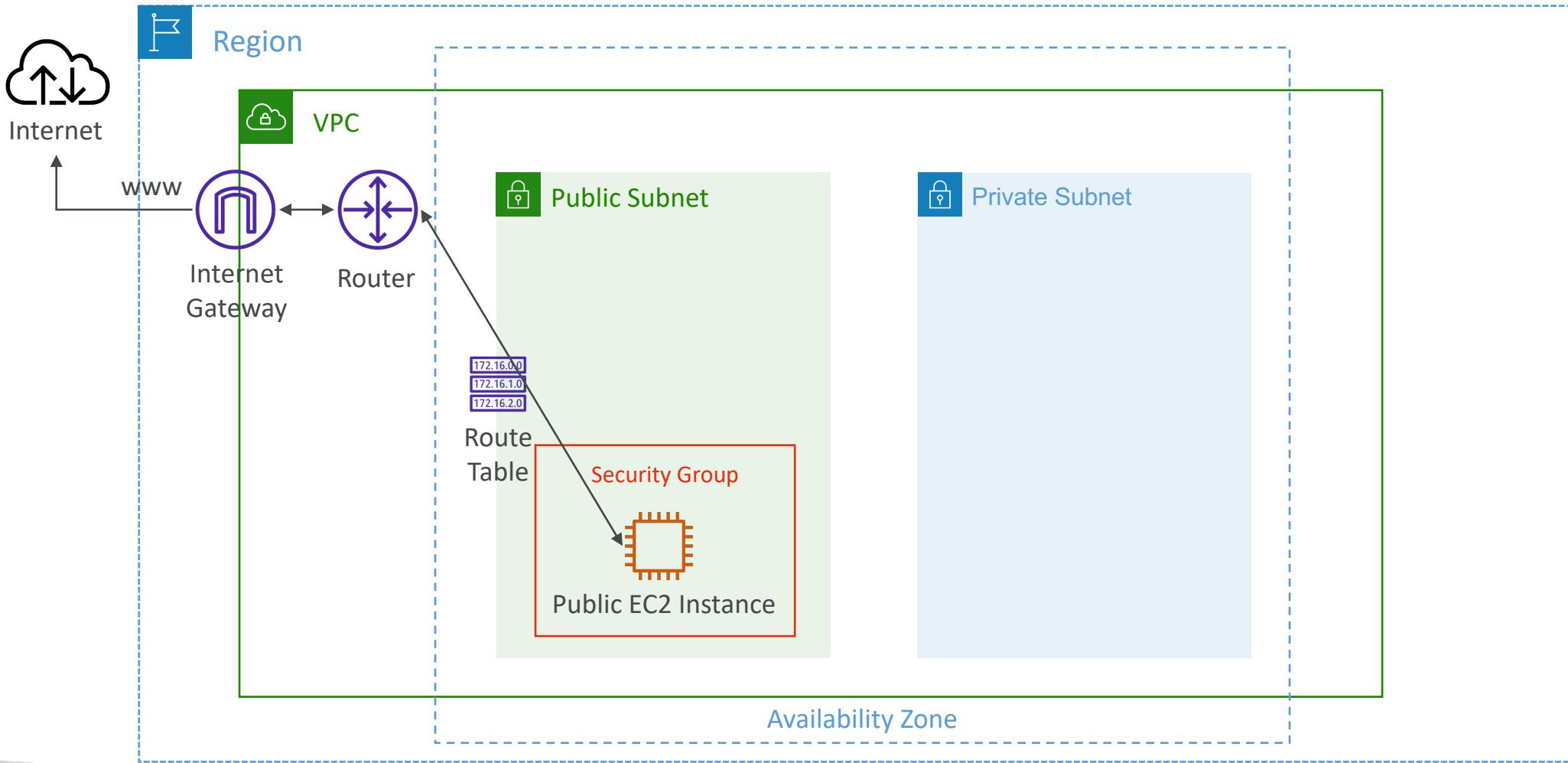
# Internet Gateway (IGW)

- Allows resources (e.g., EC2 instances) in a VPC connect to the Internet
- It scales horizontally and is highly available and redundant
- Must be created separately from a VPC
- One VPC can only be attached to one IGW and vice versa
  
- Internet Gateways on their own do not allow Internet access...
- Route tables must also be edited!

# Adding Internet Gateway

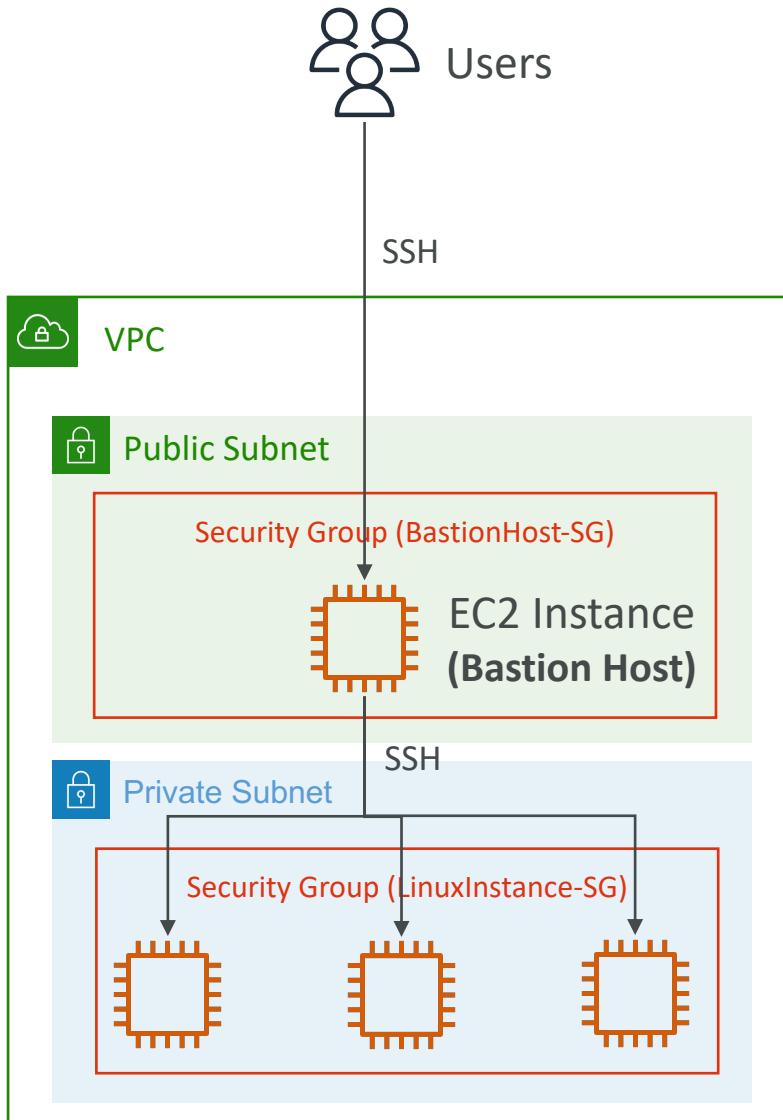


# Editing Route Tables



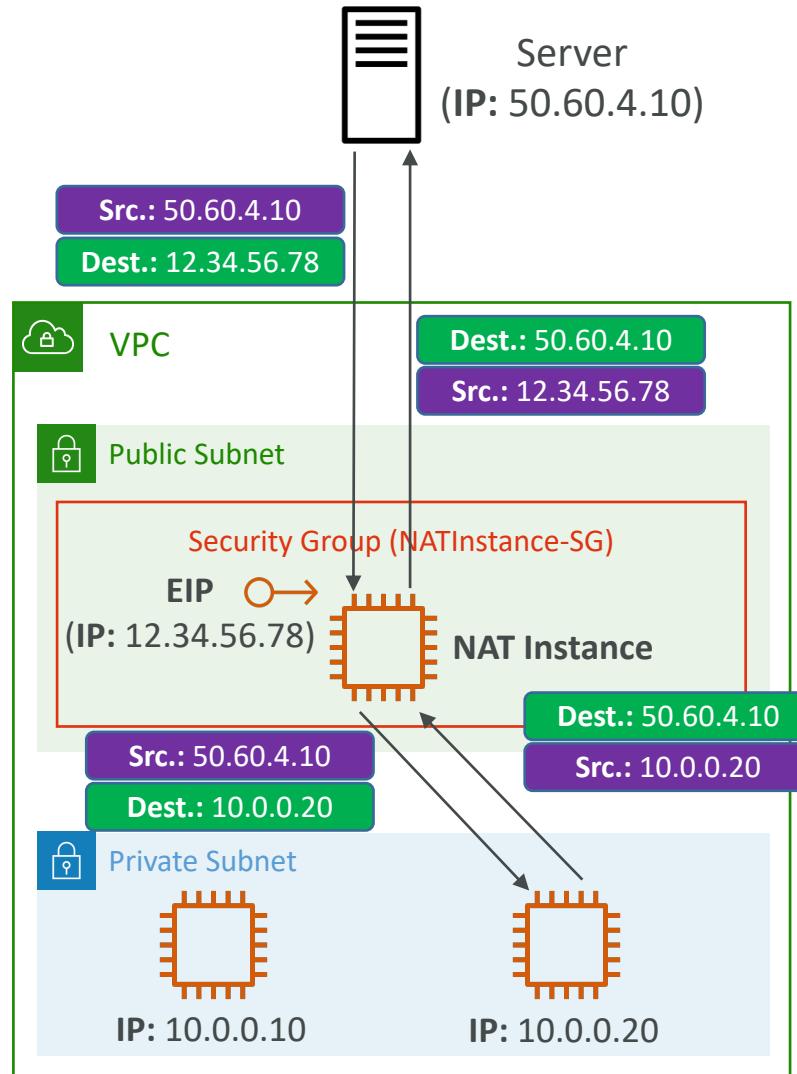
# Bastion Hosts

- We can use a Bastion Host to SSH into our private EC2 instances
- The bastion is in the public subnet which is then connected to all other private subnets
- Bastion Host security group must be tightened
- Exam Tip: Make sure the bastion host only has port 22 traffic from the IP address you need, not from the security groups of your other EC2 instances

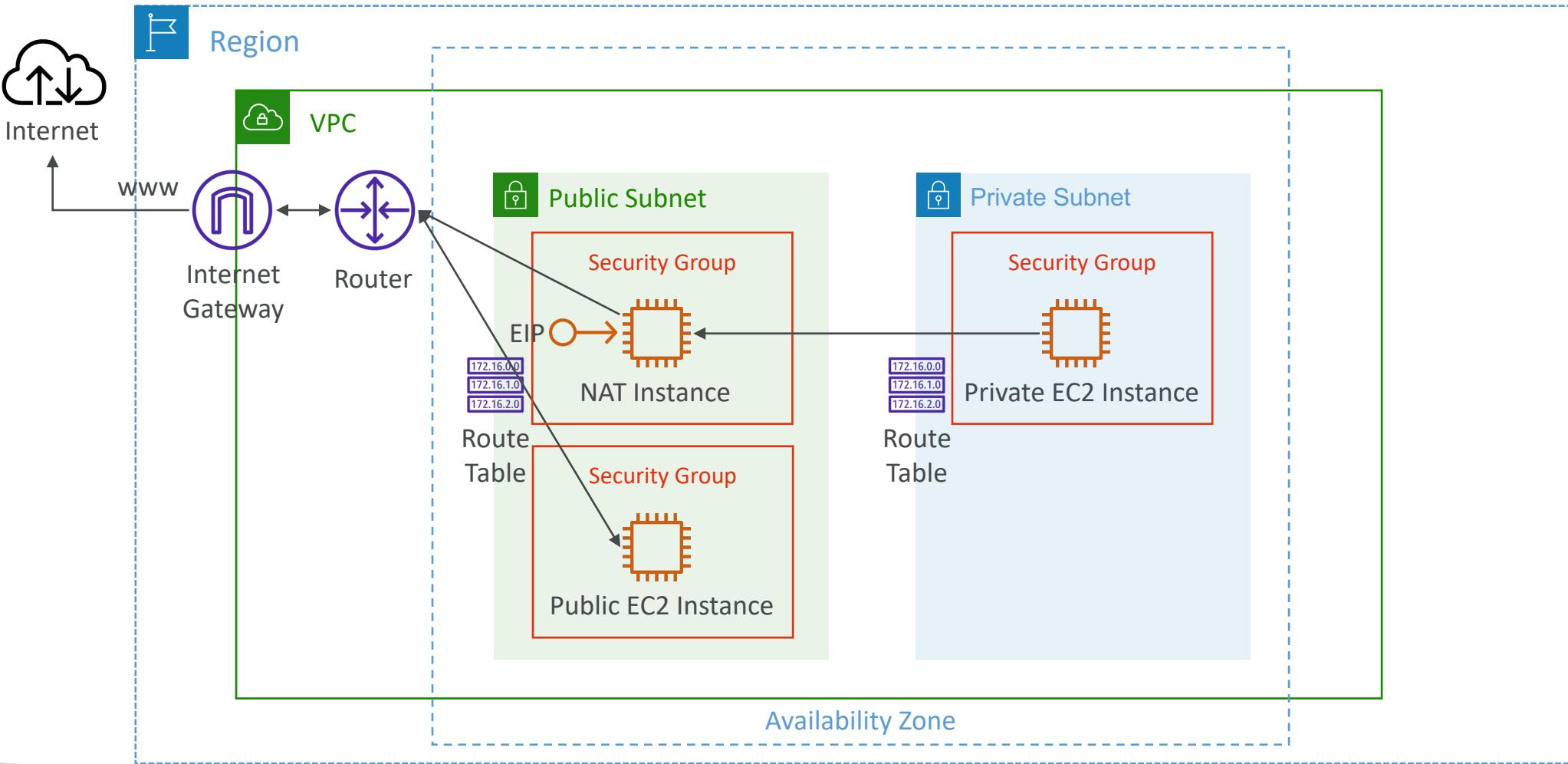


# NAT Instance (outdated, but still at the exam)

- NAT = Network Address Translation
- Allows EC2 instances in private subnets to connect to the Internet
- Must be launched in a public subnet
- Must disable EC2 setting: **Source / destination Check**
- Must have Elastic IP attached to it
- Route Tables must be configured to route traffic from private subnets to the NAT Instance

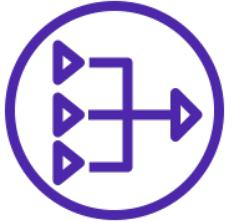


# NAT Instance



# NAT Instance – Comments

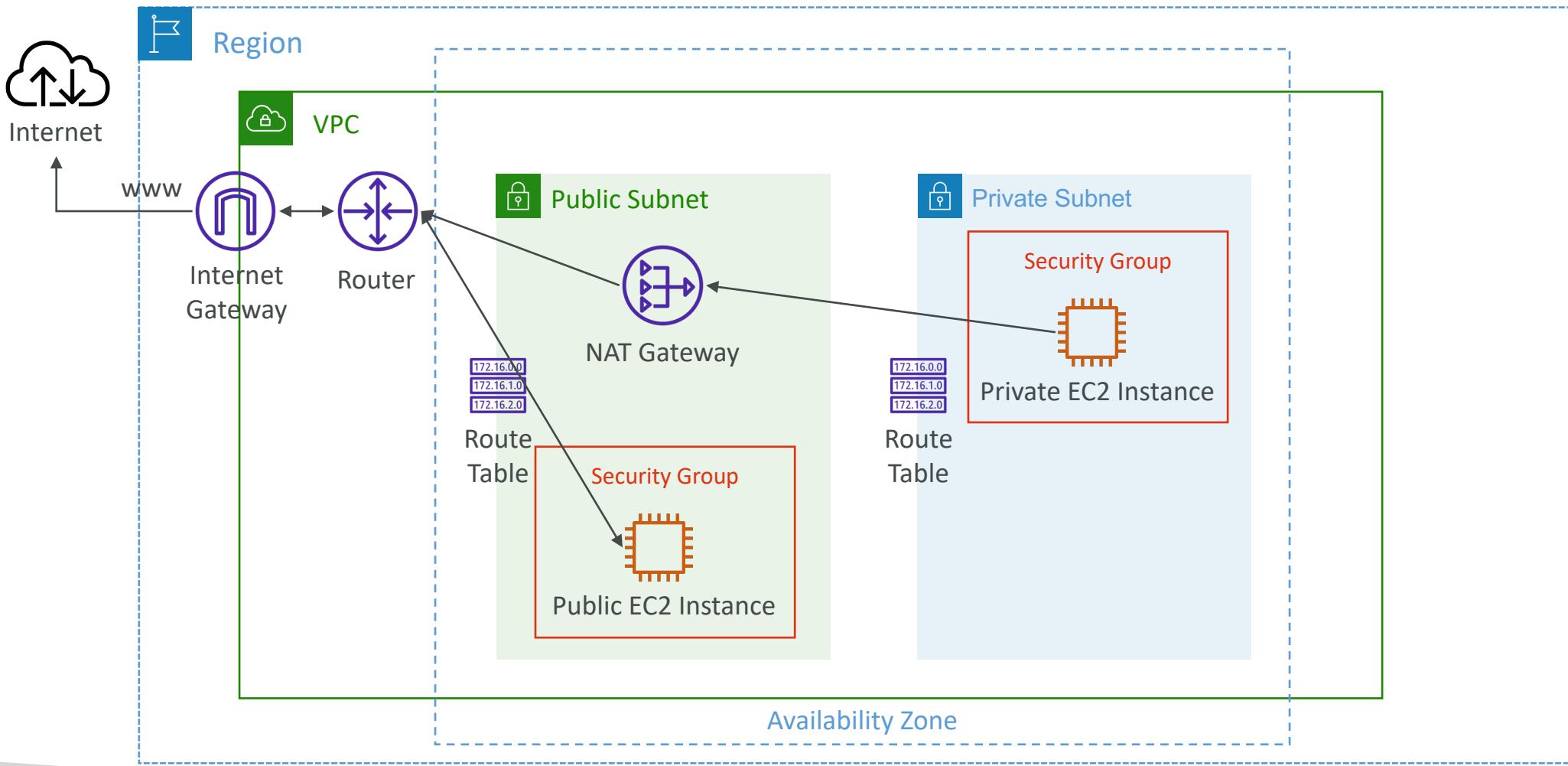
- Pre-configured Amazon Linux AMI is available
  - Reached the end of standard support on December 31, 2020
- Not highly available / resilient setup out of the box
  - You need to create an ASG in multi-AZ + resilient user-data script
- Internet traffic bandwidth depends on EC2 instance type
- You must manage Security Groups & rules:
  - Inbound:
    - Allow HTTP / HTTPS traffic coming from Private Subnets
    - Allow SSH from your home network (access is provided through Internet Gateway)
  - Outbound:
    - Allow HTTP / HTTPS traffic to the Internet



# NAT Gateway

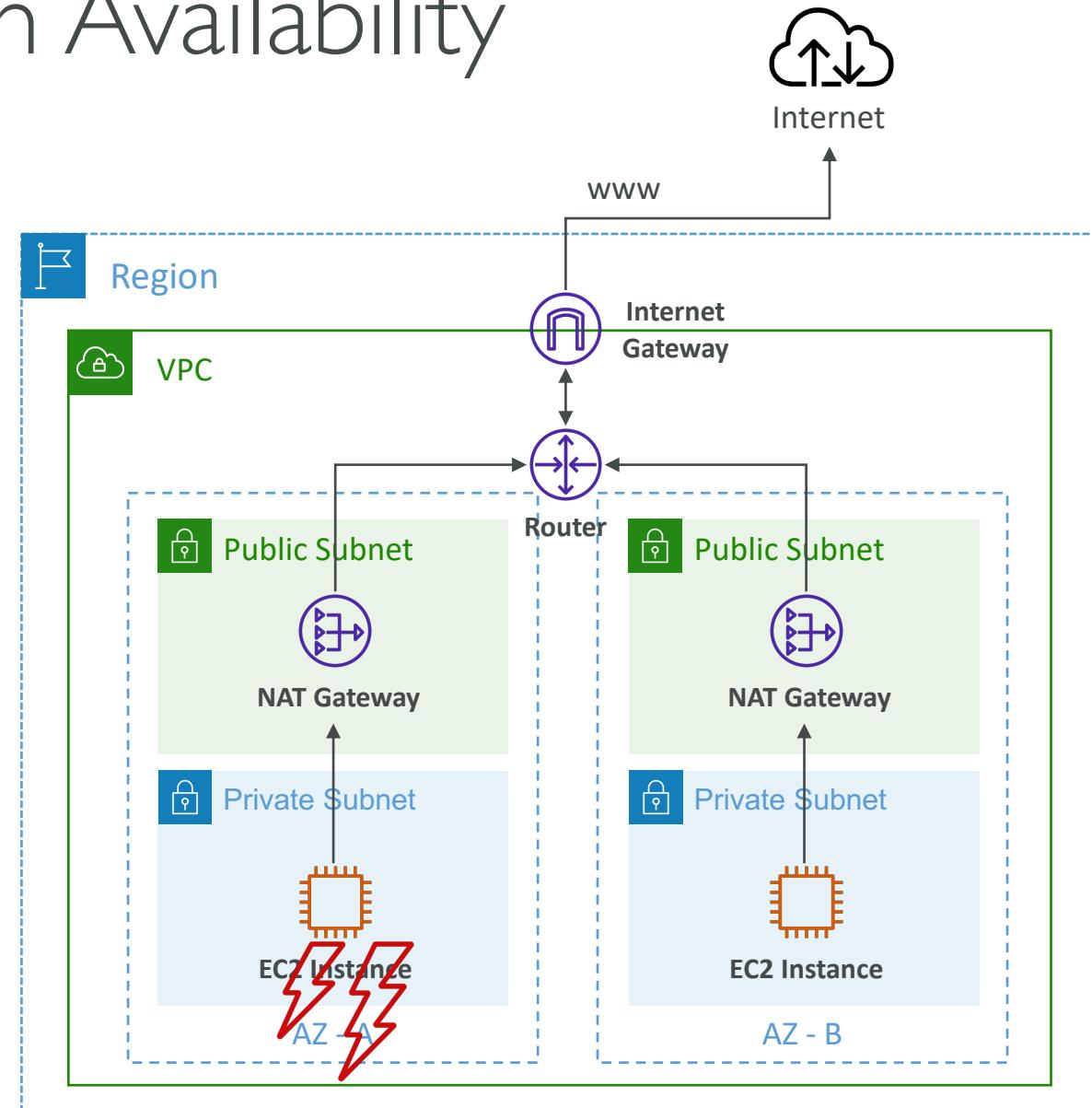
- AWS-managed NAT, higher bandwidth, high availability, no administration
- Pay per hour for usage and bandwidth
- NATGW is created in a specific Availability Zone, uses an Elastic IP
- Can't be used by EC2 instance in the same subnet (only from other subnets)
- Requires an IGW (Private Subnet => NATGW => IGW)
- 5 Gbps of bandwidth with automatic scaling up to 45 Gbps
- No Security Groups to manage / required

# NAT Gateway



# NAT Gateway with High Availability

- NAT Gateway is resilient within a single Availability Zone
- Must create multiple NAT Gateways in multiple AZs for fault-tolerance
- There is no cross-AZ failover needed because if an AZ goes down it doesn't need NAT



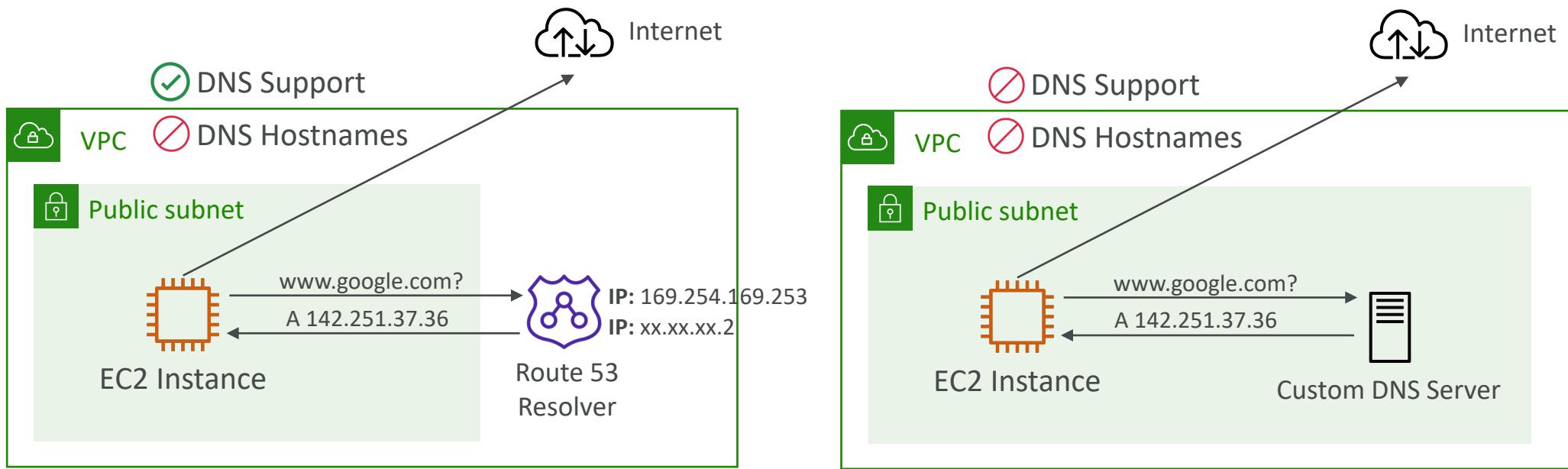
# NAT Gateway vs. NAT Instance

	NAT Gateway	NAT Instance
<b>Availability</b>	Highly available within AZ (create in another AZ)	Use a script to manage failover between instances
<b>Bandwidth</b>	Up to 45 Gbps	Depends on EC2 instance type
<b>Maintenance</b>	Managed by AWS	Managed by you (e.g., software, OS patches, ...)
<b>Cost</b>	Per hour & amount of data transferred	Per hour, EC2 instance type and size, + network \$
<b>Public IPv4</b>	✓	✓
<b>Private IPv4</b>	✓	✓
<b>Security Groups</b>	✗	✓
<b>Use as Bastion Host?</b>	✗	✓

More at: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html>

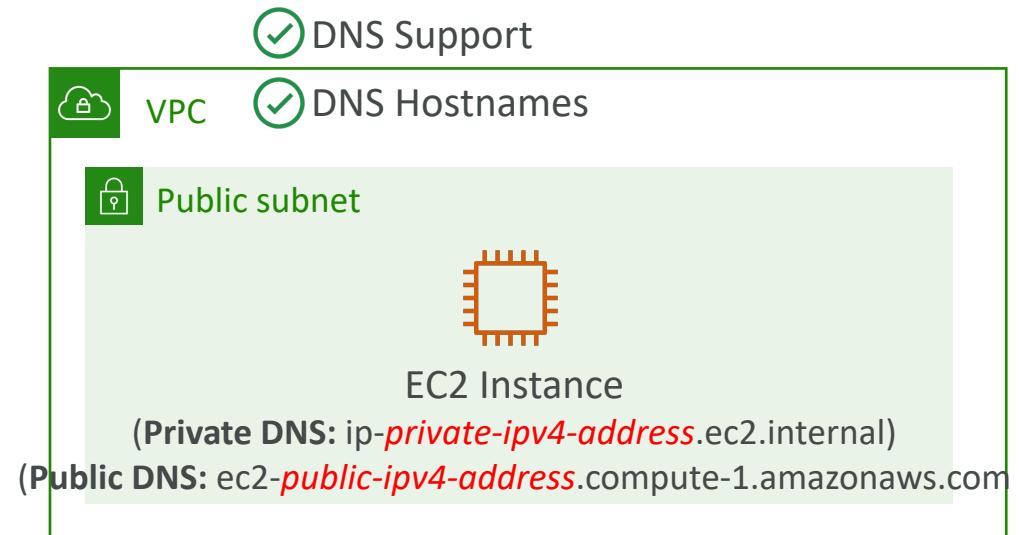
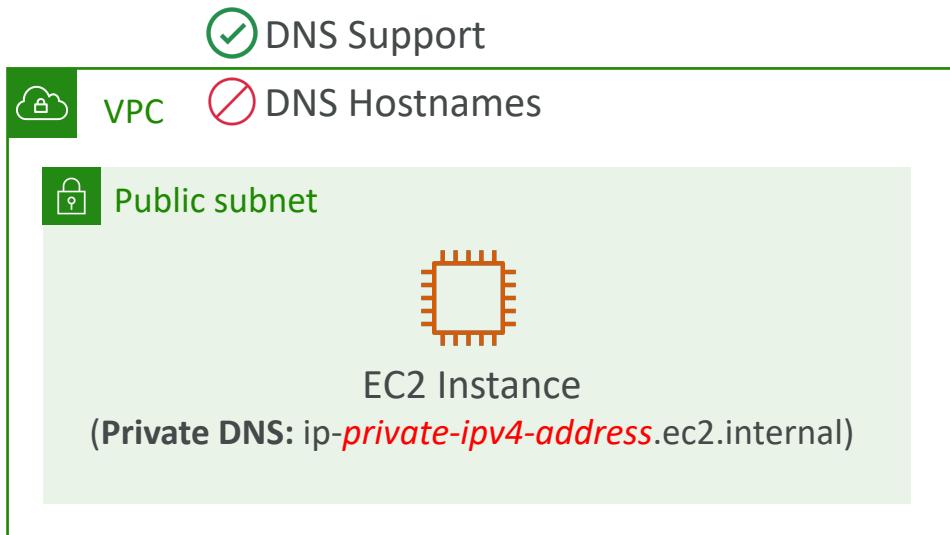
# DNS Resolution in VPC

- DNS Resolution (`enableDnsSupport`)
  - Decides if DNS resolution from Route 53 Resolver server is supported for the VPC
  - True (default): it queries the Amazon Provider DNS Server at 169.254.169.253 or the reserved IP address at the base of the VPC IPv4 network range plus two (.2)



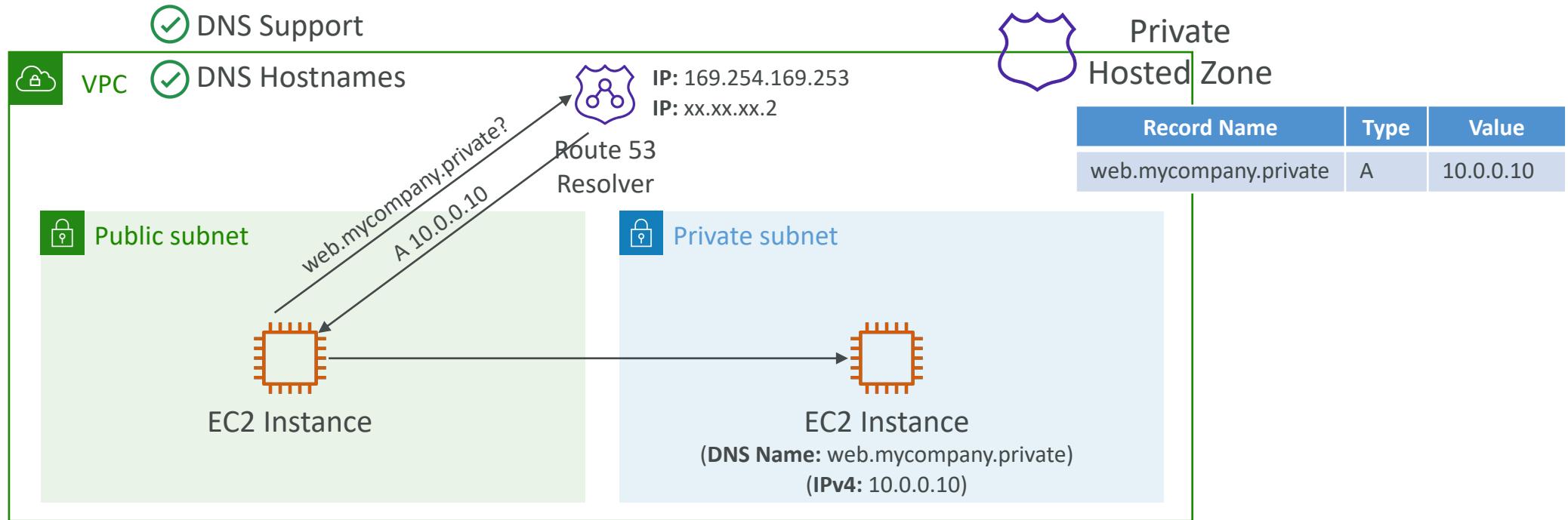
# DNS Resolution in VPC

- DNS Hostnames (enableDnsHostnames)
  - By default,
    - True => default VPC
    - False => newly created VPCs
  - Won't do anything unless enableDnsSupport=true
  - IfTrue, assigns public hostname to EC2 instance if it has a public IPv4



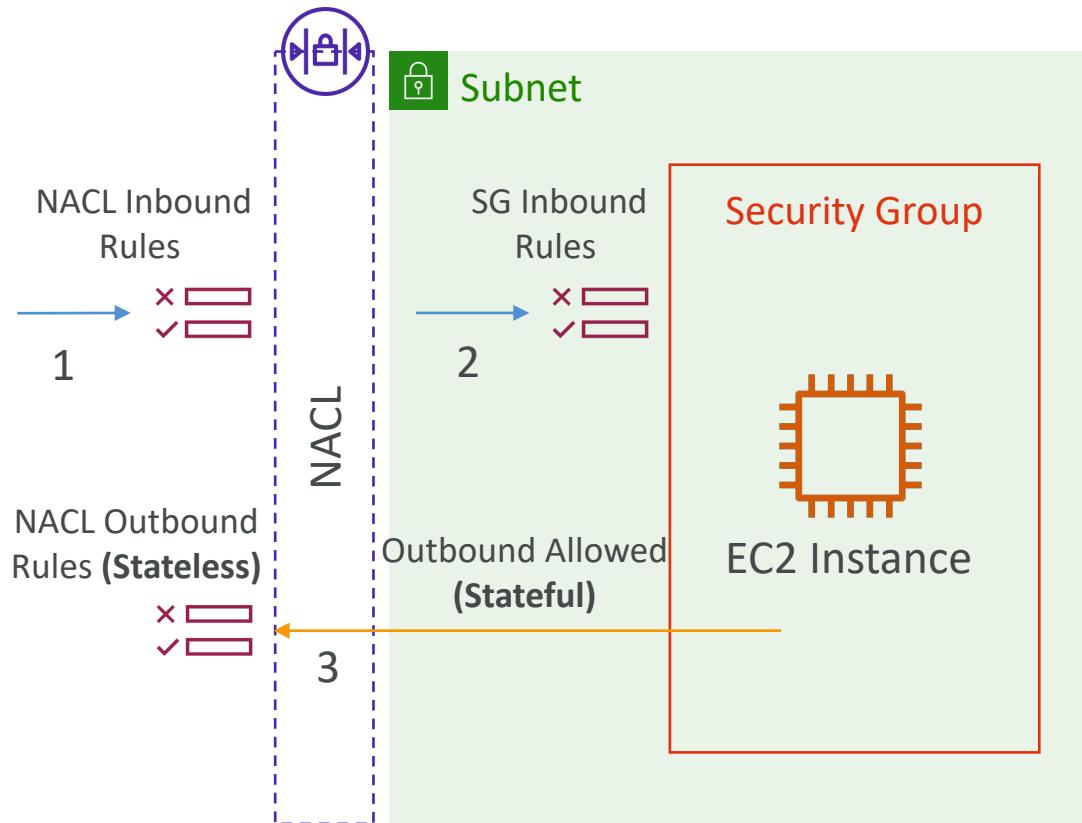
# DNS Resolution in VPC

- If you use custom DNS domain names in a Private Hosted Zone in Route 53, you must set both these attributes (enableDnsSupport & enableDnsHostname) to true

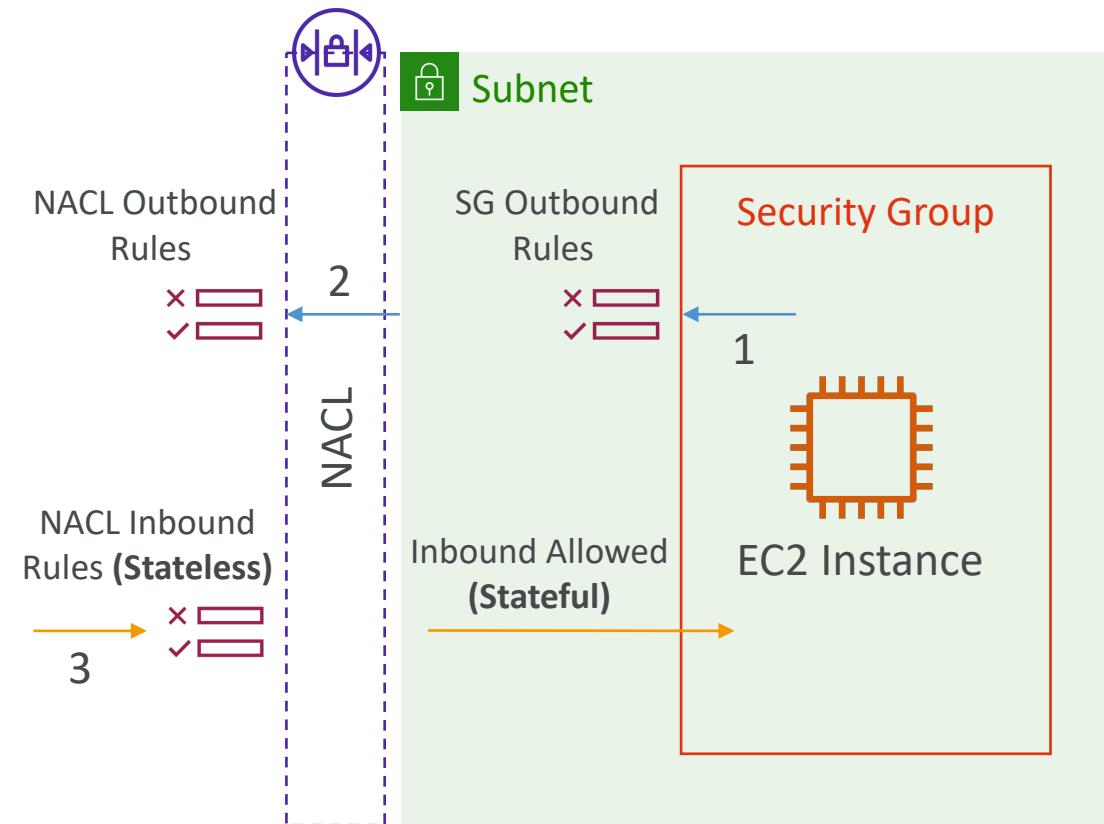


# Security Groups & NACLs

## Incoming Request



## Outgoing Request

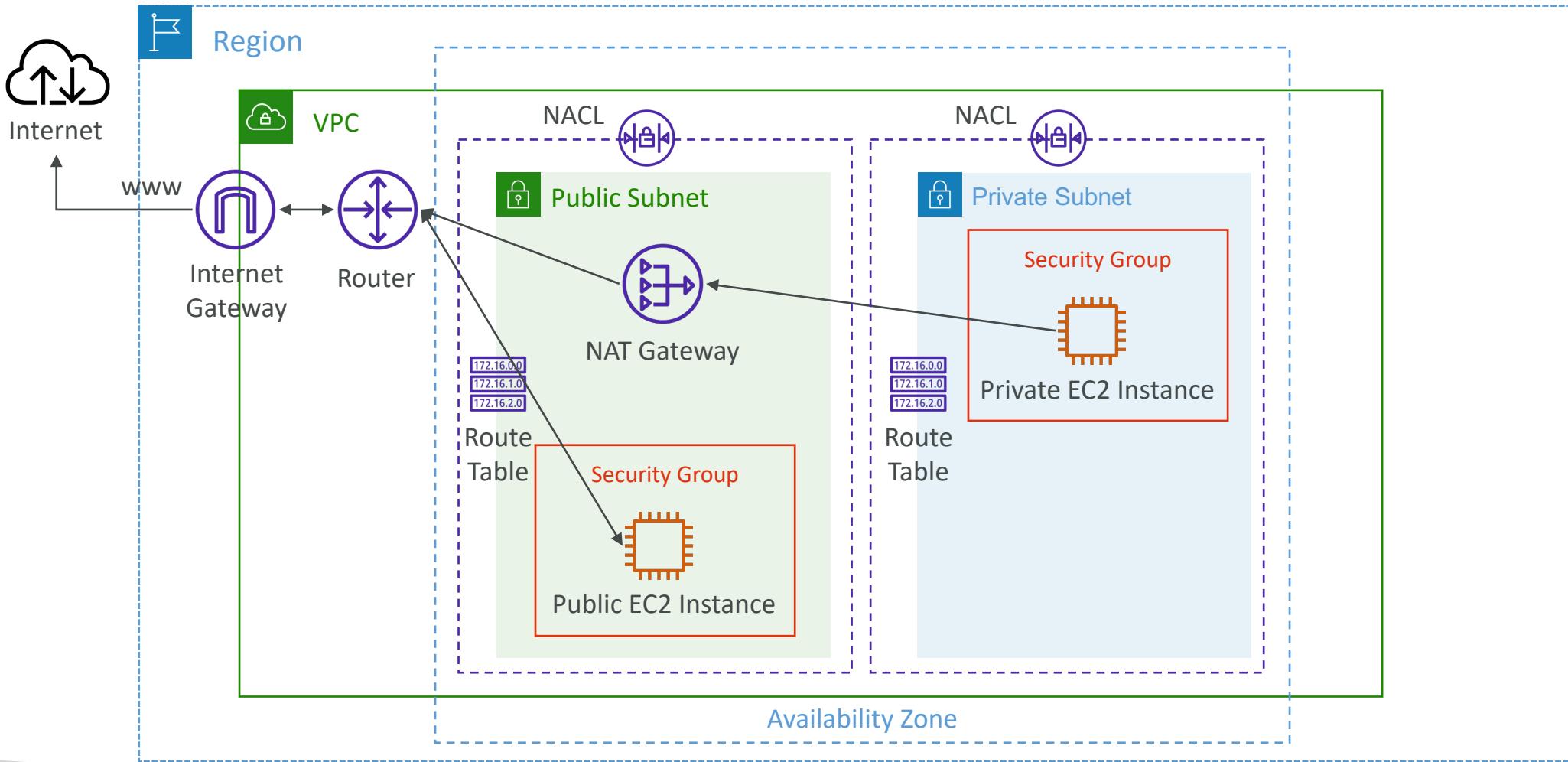


# Network Access Control List (NACL)



- NACL are like a firewall which control traffic from and to subnets
- One NACL per subnet, new subnets are assigned the Default NACL
- You define NACL Rules:
  - Rules have a number (1-32766), higher precedence with a lower number
  - First rule match will drive the decision
  - Example: if you define #100 ALLOW 10.0.0.10/32 and #200 DENY 10.0.0.10/32, the IP address will be allowed because 100 has a higher precedence over 200
  - The last rule is an asterisk (\*) and denies a request in case of no rule match
  - AWS recommends adding rules by increment of 100
- Newly created NACLs will deny everything
- NACL are a great way of blocking a specific IP address at the subnet level

# NACLs



# Default NACL

- Accepts everything inbound/outbound with the subnets it's associated with
- Do NOT modify the Default NACL, instead create custom NACLs



Default NACL for a VPC that supports IPv4

## Inbound Rules

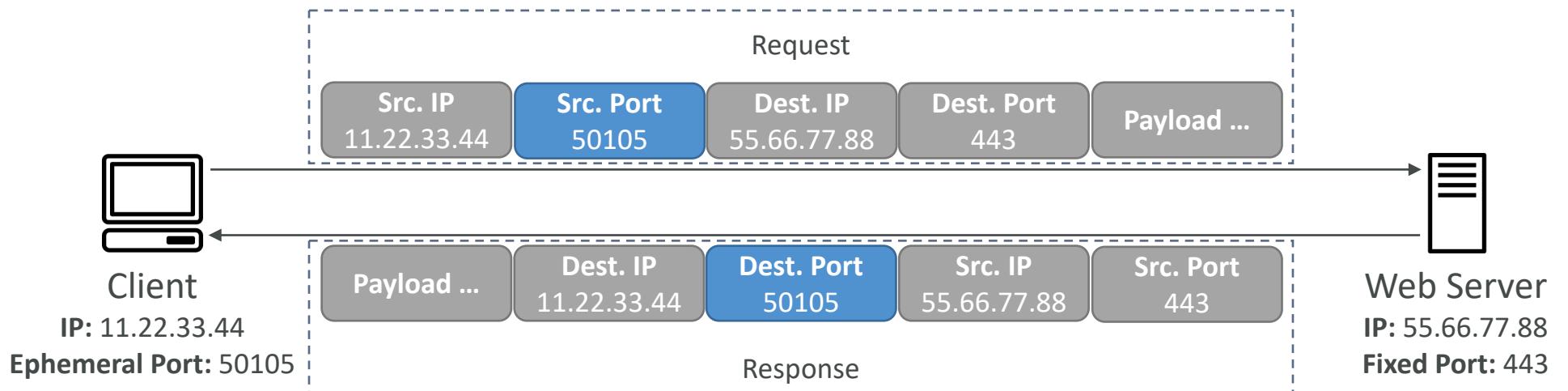
Rule #	Type	Protocol	Port Range	Source	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

## Outbound Rules

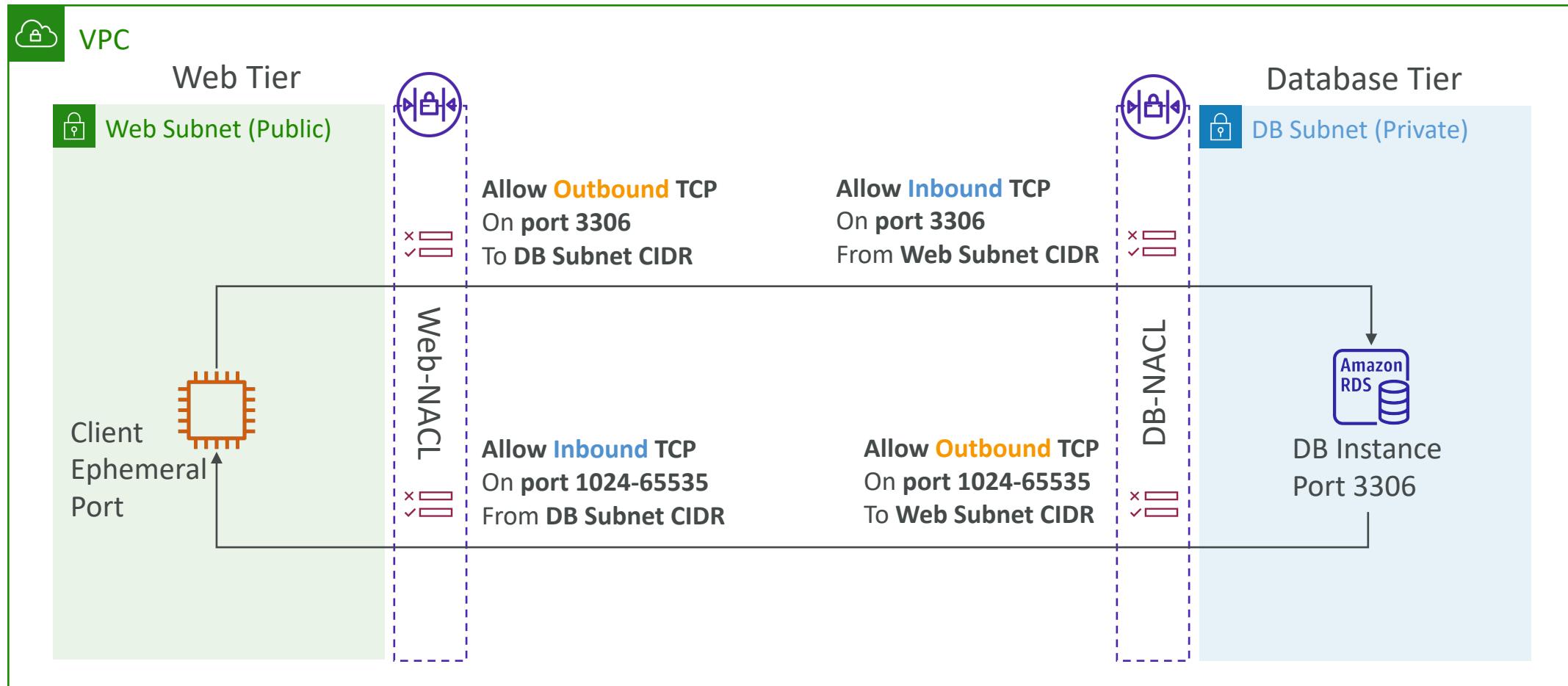
Rule #	Type	Protocol	Port Range	Destination	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

# Ephemeral Ports

- For any two endpoints to establish a connection, they must use ports
- Clients connect to a **defined port**, and expect a response on an **ephemeral port**
- Different Operating Systems use different port ranges, examples:
  - IANA & MS Windows 10 → 49152 – 65535
  - Many Linux Kernels → 32768 – 60999

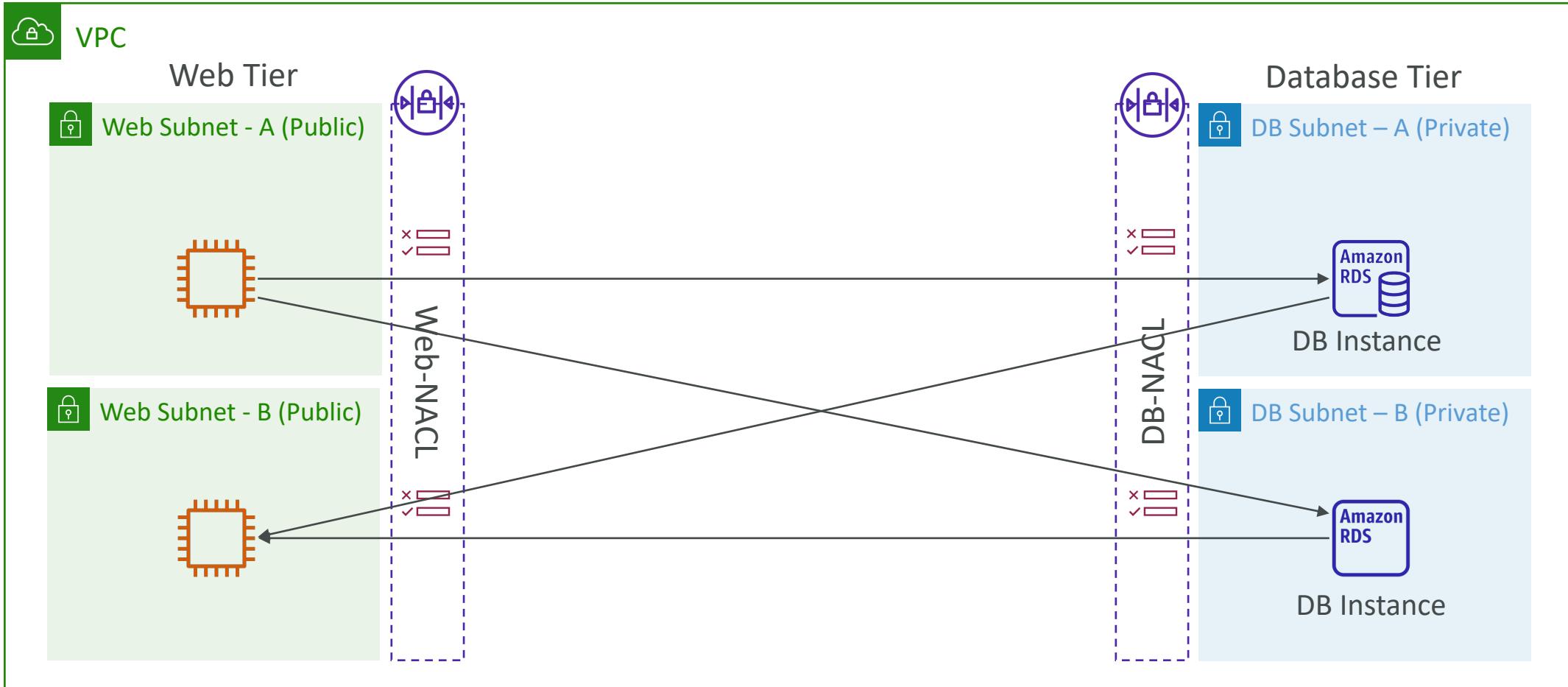


# NACL with Ephemeral Ports



<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html#nacl-ephemeral-ports>

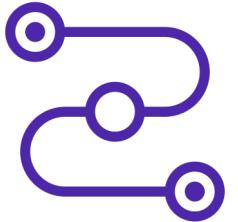
# Create NACL rules for each target subnets CIDR



# Security Group vs. NACLs

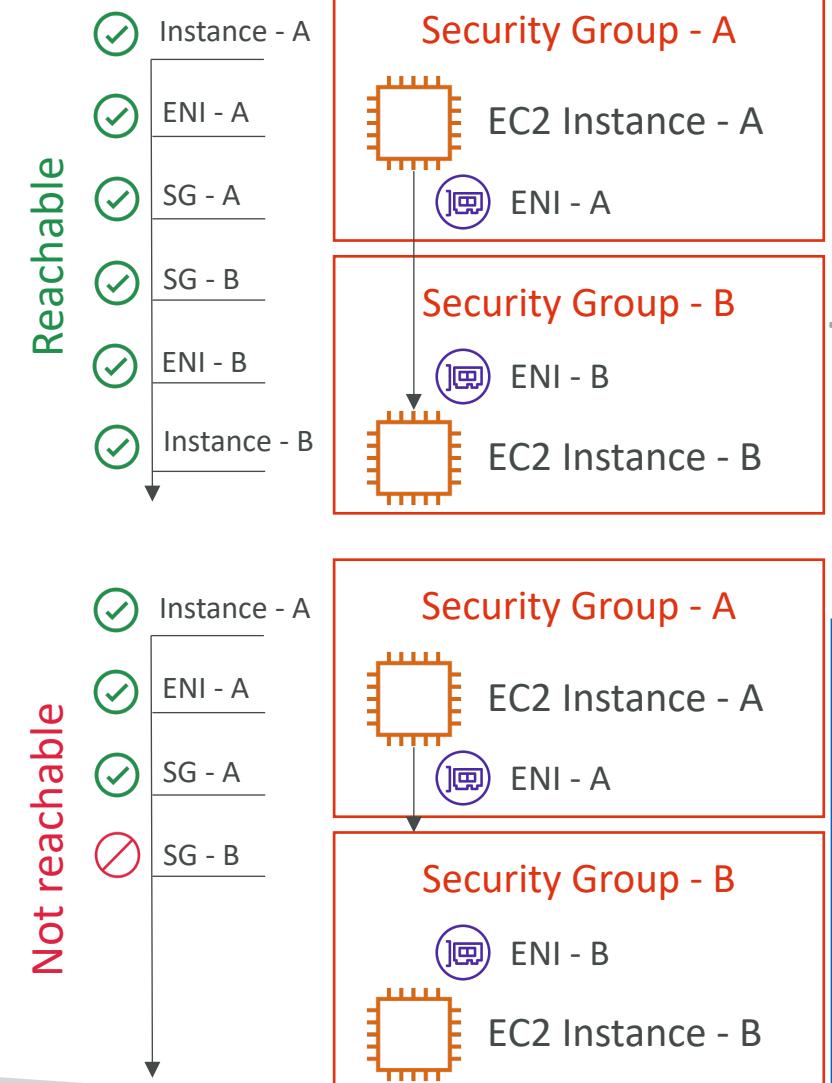
Security Group	NACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
<b>Stateful:</b> return traffic is automatically allowed, regardless of any rules	<b>Stateless:</b> return traffic must be explicitly allowed by rules (think of ephemeral ports)
All rules are evaluated before deciding whether to allow traffic	Rules are evaluated in order (lowest to highest) when deciding whether to allow traffic, first match wins
Applies to an EC2 instance when specified by someone	Automatically applies to all EC2 instances in the subnet that it's associated with

NACL Examples: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html>



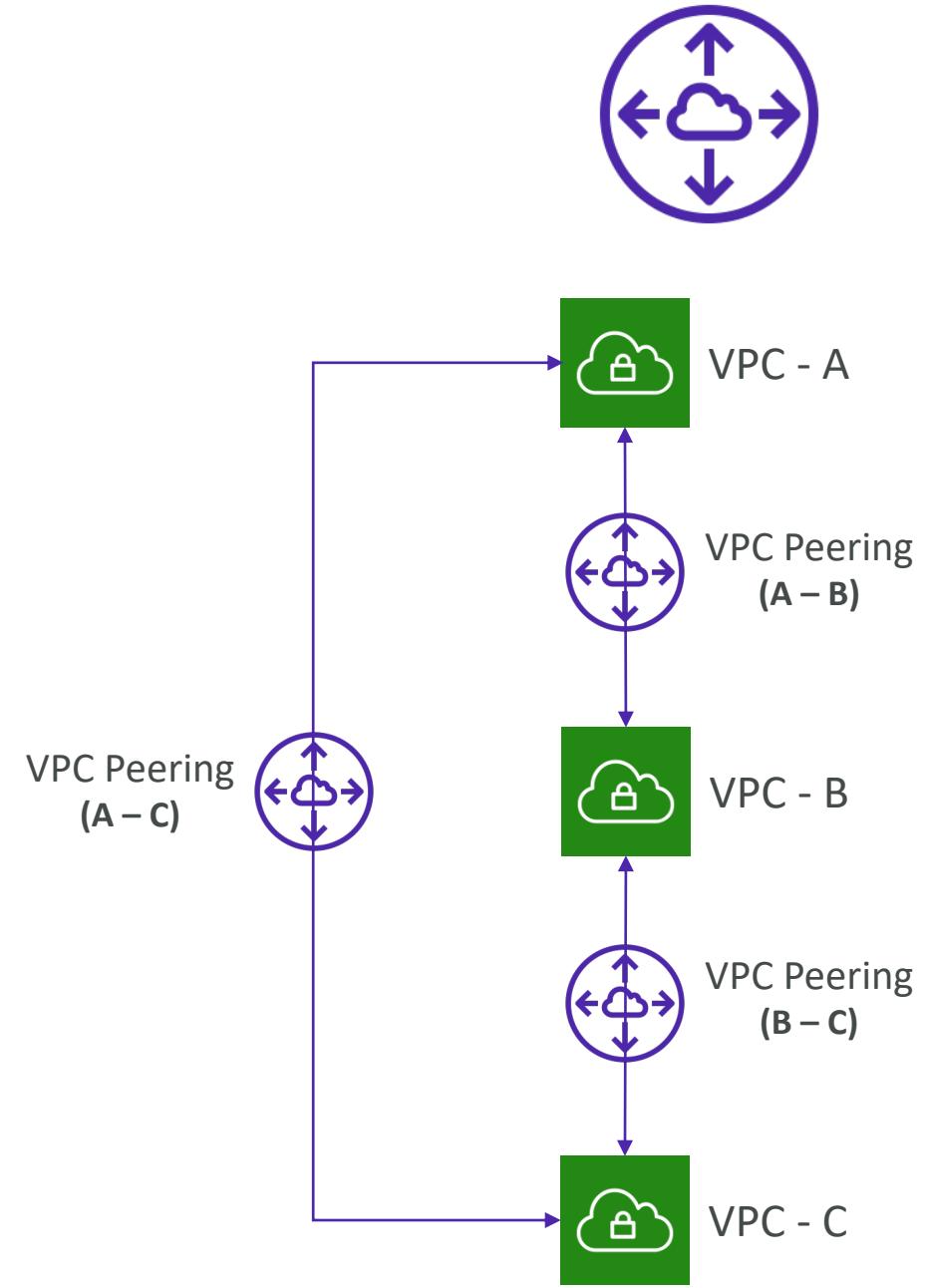
# VPC – Reachability Analyzer

- A network diagnostics tool that troubleshoots network connectivity between two endpoints in your VPC(s)
- It builds a model of the network configuration, then checks the reachability based on these configurations (**it doesn't send packets**)
- When the destination is
  - **Reachable** – it produces hop-by-hop details of the virtual network path
  - **Not reachable** – it identifies the blocking component(s) (e.g., configuration issues in SGs, NACLs, Route Tables, ...)
- Use cases: troubleshoot connectivity issues, ensure network configuration is as intended, ...



# VPC Peering

- Privately connect two VPCs using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDRs
- VPC Peering connection is **NOT** transitive (must be established for each VPC that need to communicate with one another)
- You must update route tables in each VPC's subnets to ensure EC2 instances can communicate with each other



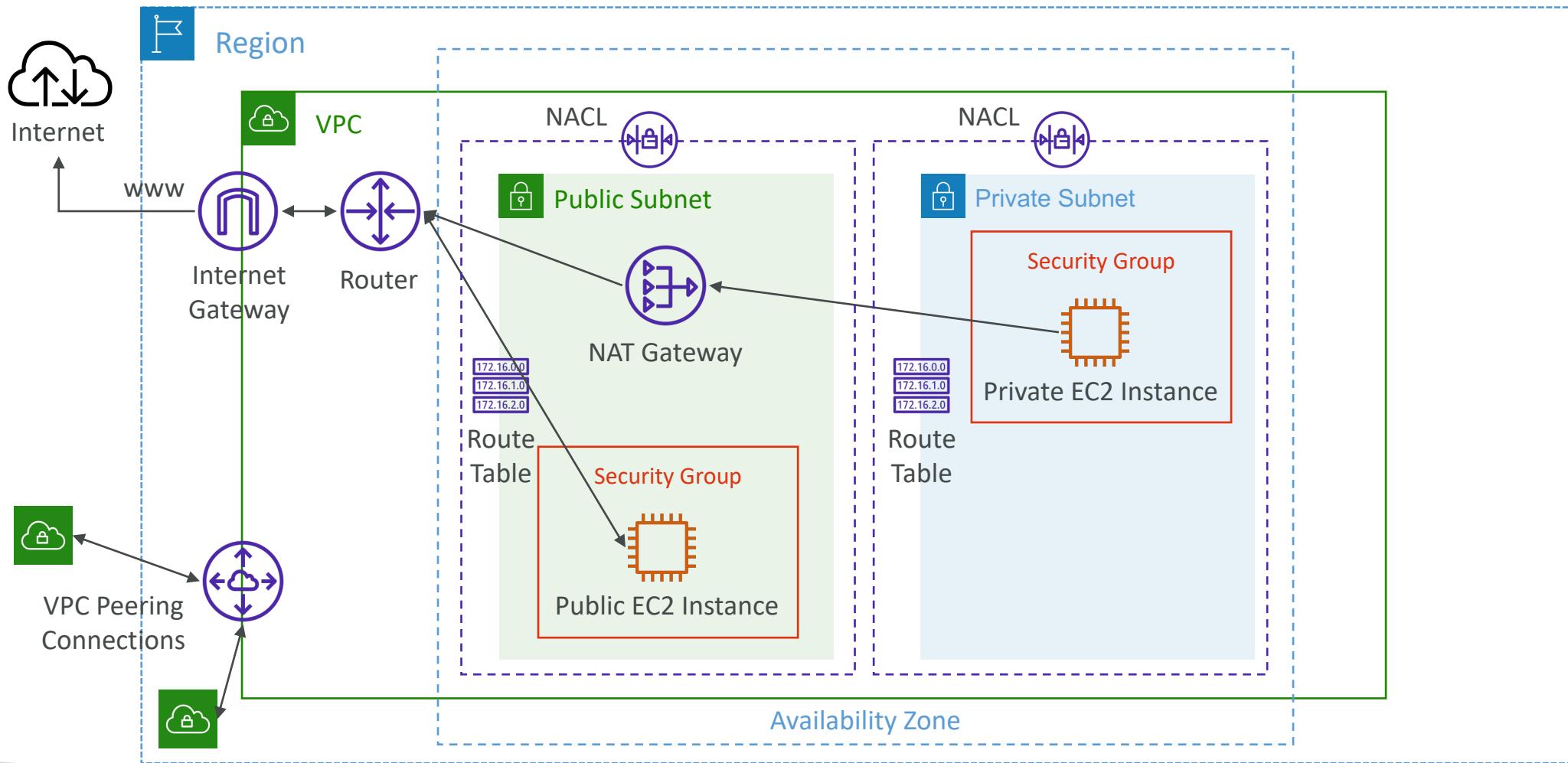
# VPC Peering – Good to know

- You can create VPC Peering connection between VPCs in different AWS accounts/regions
- You can reference a security group in a peered VPC (works cross accounts – same region)

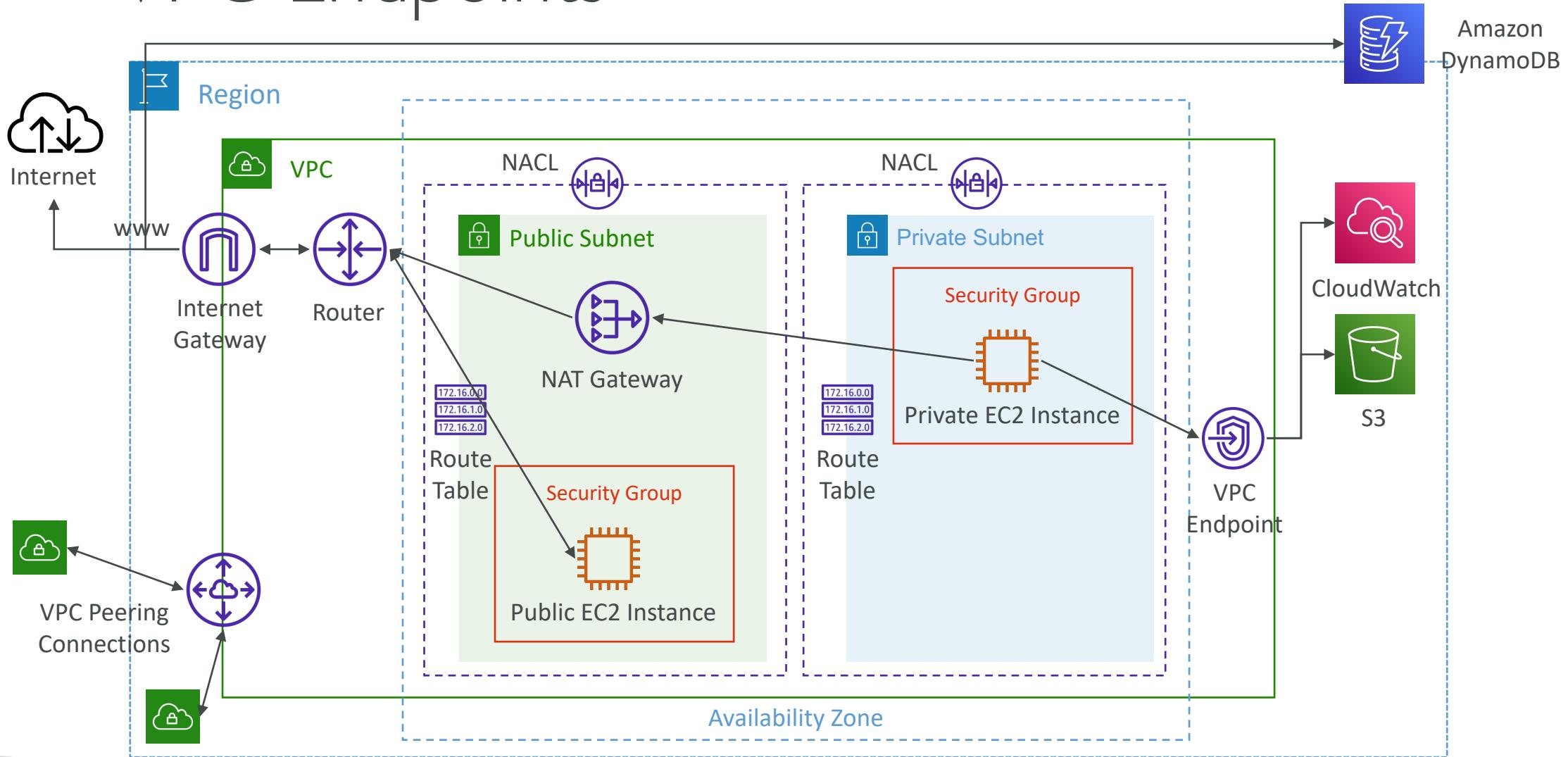
Type	Protocol	Port range	Source
HTTP	TCP	80	sg-04991f9af3473b939 / default
HTTP	TCP	80	[REDACTED] / sg-027ad1f7865d4be76

↑  
Account ID

# VPC Peering



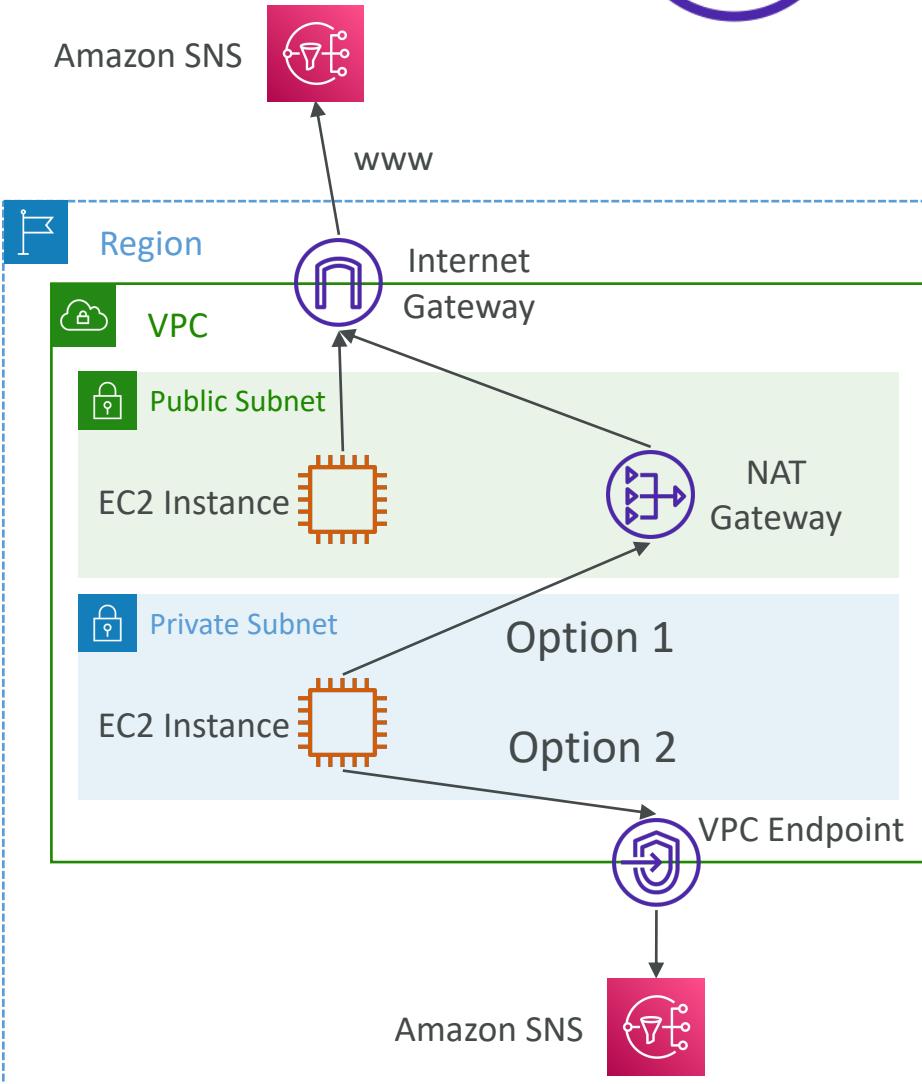
# VPC Endpoints



# VPC Endpoints (AWS PrivateLink)



- Every AWS service is publicly exposed (public URL)
- VPC Endpoints (powered by AWS PrivateLink) allows you to connect to AWS services using a private network instead of using the public Internet
- They're redundant and scale horizontally
- They remove the need of IGW, NATGW, ... to access AWS Services
- In case of issues:
  - Check DNS Setting Resolution in your VPC
  - Check Route Tables



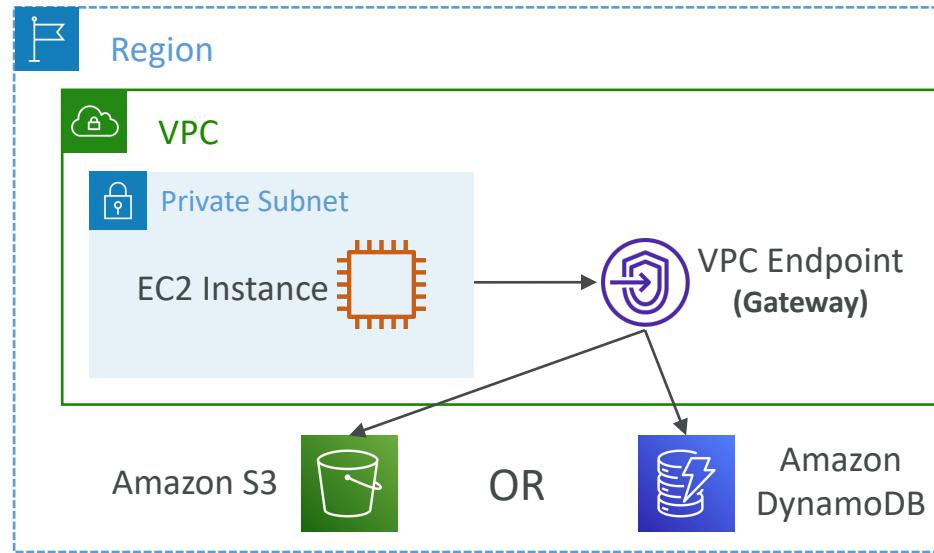
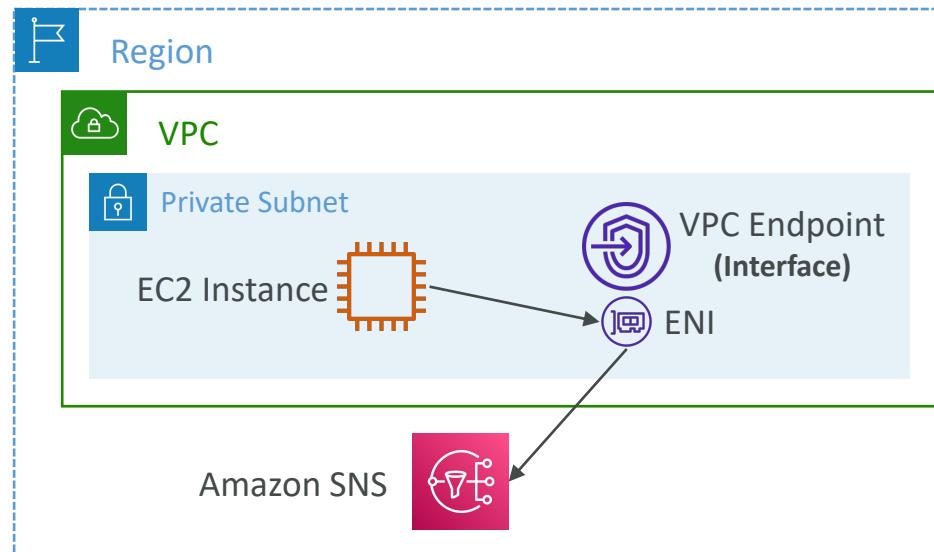
# Types of Endpoints

- Interface Endpoints

- Provisions an ENI (private IP address) as an entry point (must attach a Security Group)
- Supports most AWS services

- Gateway Endpoints

- Provisions a gateway and must be used as a target in a route table
- Supports both S3 and DynamoDB

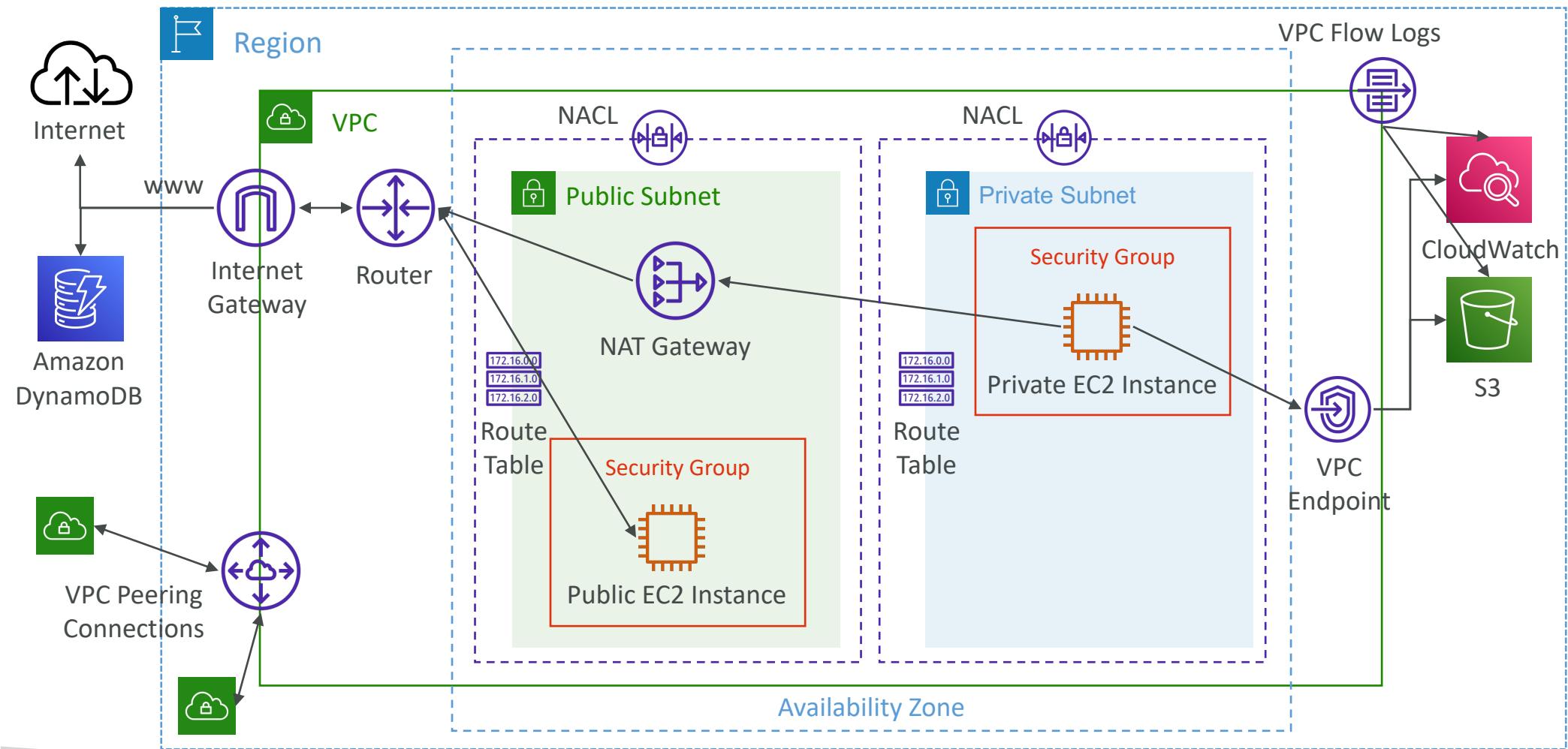




# VPC Flow Logs

- Capture information about IP traffic going into your interfaces:
  - VPC Flow Logs
  - Subnet Flow Logs
  - Elastic Network Interface (ENI) Flow Logs
- Helps to monitor & troubleshoot connectivity issues
- Flow logs data can go to S3 / CloudWatch Logs
- Captures network information from AWS managed interfaces too: ELB, RDS, ElastiCache, Redshift, WorkSpaces, NATGW, Transit Gateway...

# VPC Flow Logs



# VPC Flow Logs Syntax

version	interface-id	dstaddr	dstport	packets	start	action
2	123456789010	eni-1235b8ca123456789	172.31.16.139	172.31.16.21	20641	ACCEPT OK
2	123456789010	eni-1235b8ca123456789	172.31.9.69	172.31.9.12	49761	REJECT OK
account-id	srcaddr	srcport	protocol	bytes	end	log-status

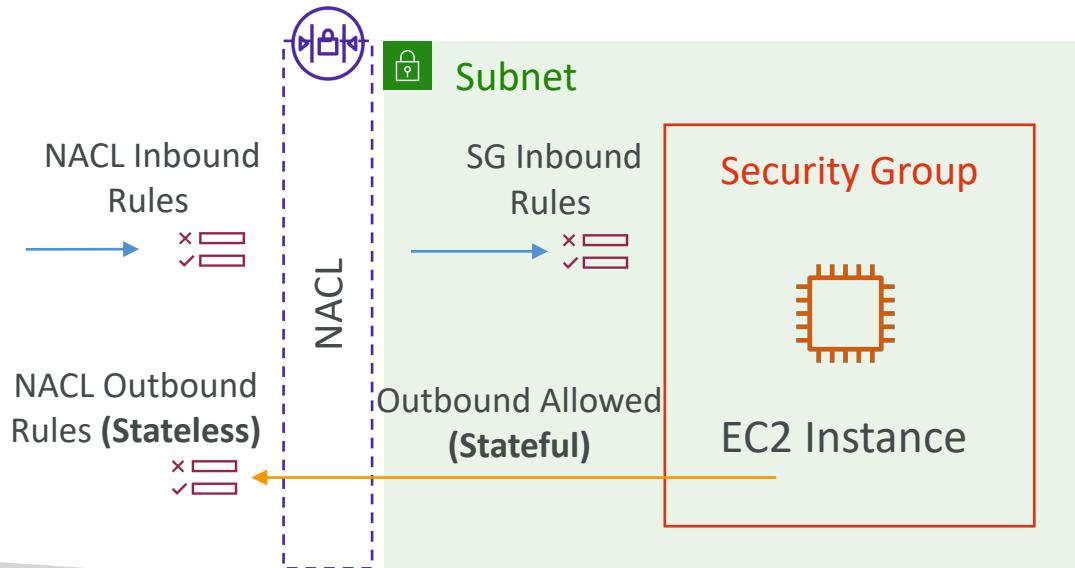
- srcaddr & dstaddr – help identify problematic IP
- srcport & dstport – help identify problematic ports
- Action – success or failure of the request due to Security Group / NACL
- Can be used for analytics on usage patterns, or malicious behavior
- Query VPC flow logs using Athena on S3 or CloudWatch Logs Insights
- Flow Logs examples: <https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs-records-examples.html>

# VPC Flow Logs – Troubleshoot SG & NACL issues

Look at the “ACTION” field

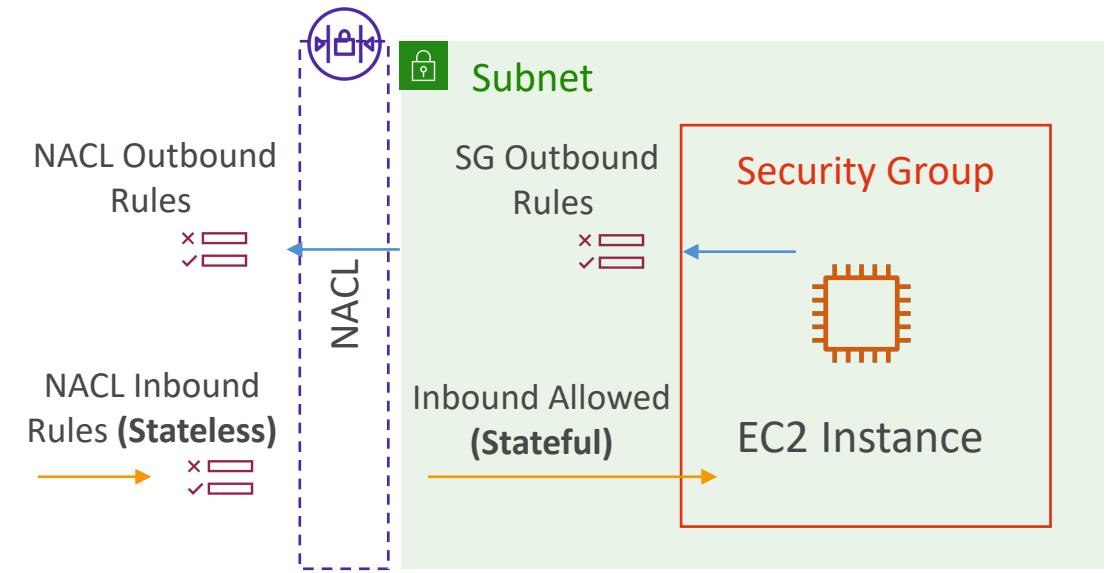
## Incoming Requests

- Inbound REJECT => NACL or SG
- Inbound ACCEPT, Outbound REJECT => NACL

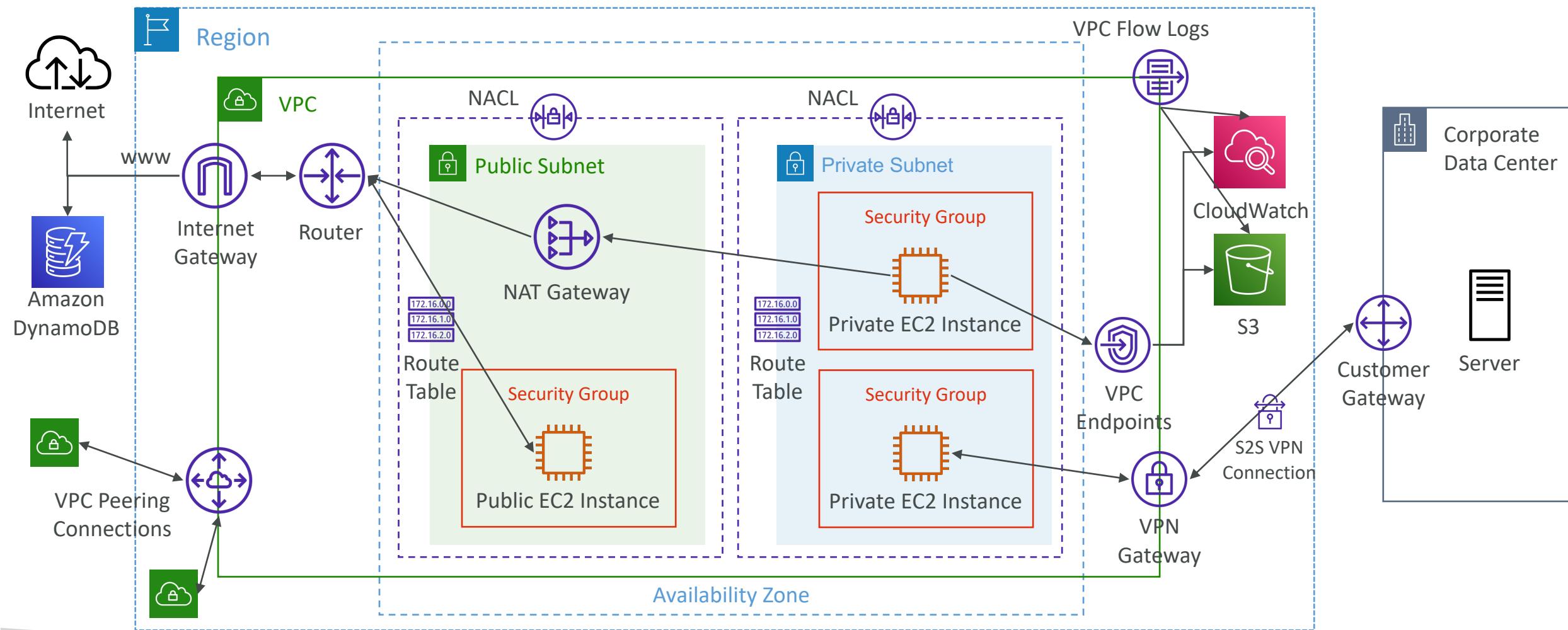


## Outgoing Requests

- Outbound REJECT => NACL or SG
- Outbound ACCEPT, Inbound REJECT => NACL



# AWS Site-to-Site VPN



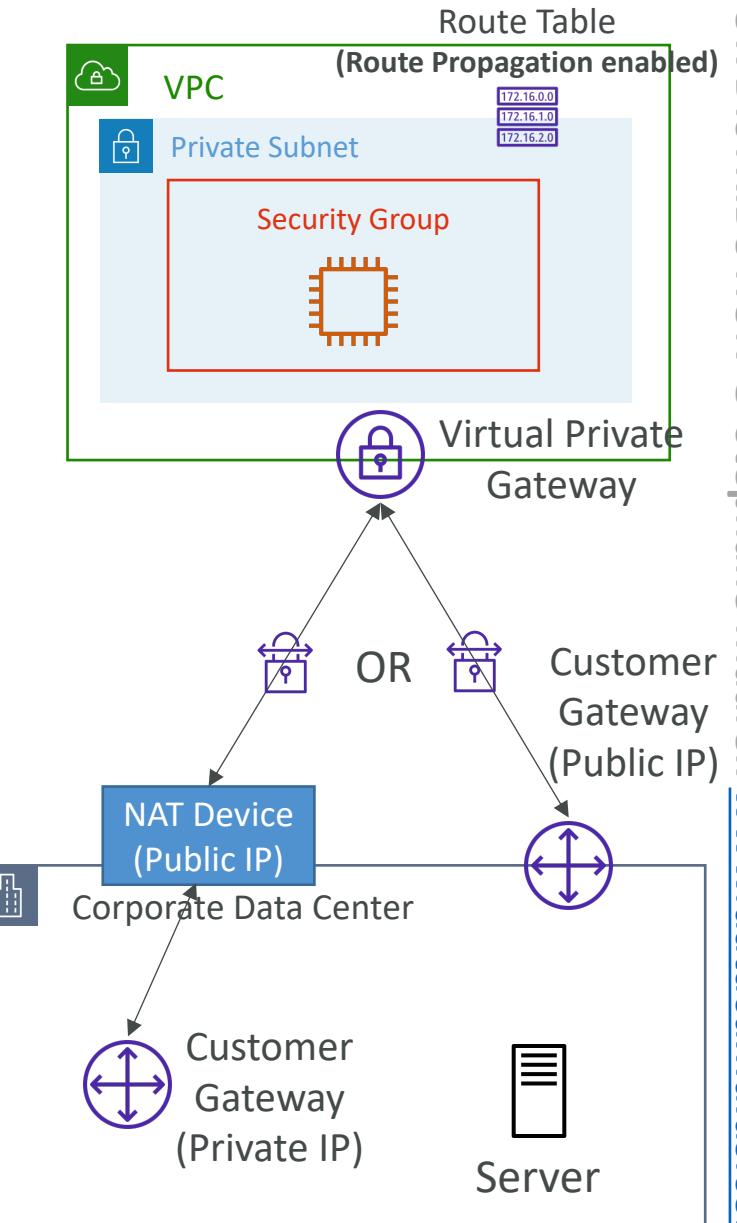
# AWS Site-to-Site VPN



- **Virtual Private Gateway (VGW)**
  - VPN concentrator on the AWS side of the VPN connection
  - VGW is created and attached to the VPC from which you want to create the Site-to-Site VPN connection
  - Possibility to customize the ASN (Autonomous System Number)
- **Customer Gateway (CGW)**
  - Software application or physical device on customer side of the VPN connection
  - <https://docs.aws.amazon.com/vpn/latest/s2svpn/your-cgw.html#DevicesTested>

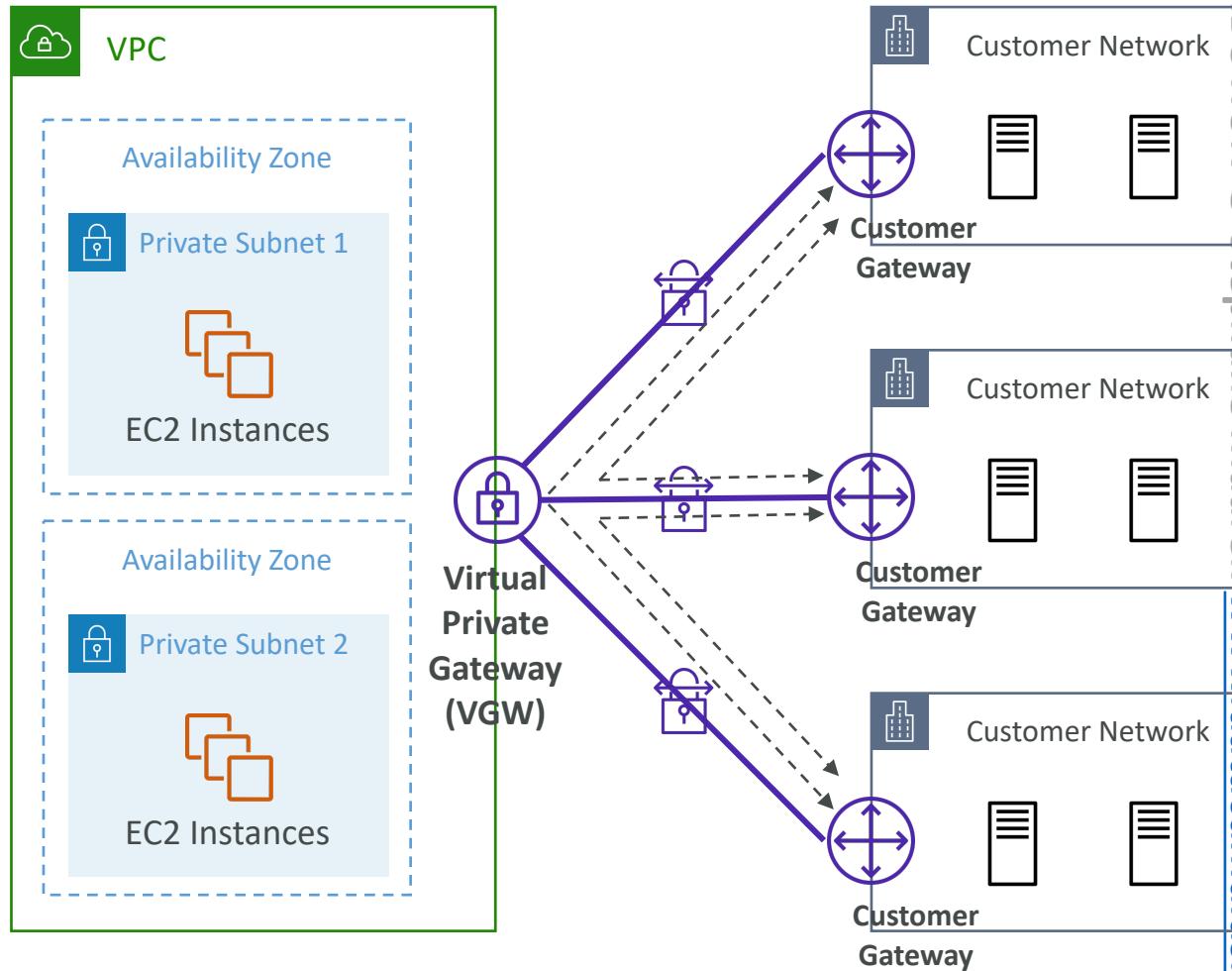
# Site-to-Site VPN Connections

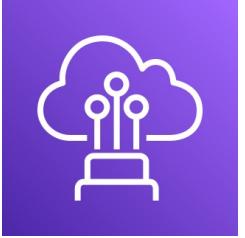
- Customer Gateway Device (On-premises)
  - What IP address to use?
    - Public Internet-routable IP address for your Customer Gateway device
    - If it's behind a NAT device that's enabled for NAT traversal (NAT-T), use the public IP address of the NAT device
- Important step: enable Route Propagation for the Virtual Private Gateway in the route table that is associated with your subnets
- If you need to ping your EC2 instances from on-premises, make sure you add the ICMP protocol on the inbound of your security groups



# AWS VPN CloudHub

- Provide secure communication between multiple sites, if you have multiple VPN connections
- Low-cost hub-and-spoke model for primary or secondary network connectivity between different locations (VPN only)
- It's a VPN connection so it goes over the public Internet
- To set it up, connect multiple VPN connections on the same VGW, setup dynamic routing and configure route tables

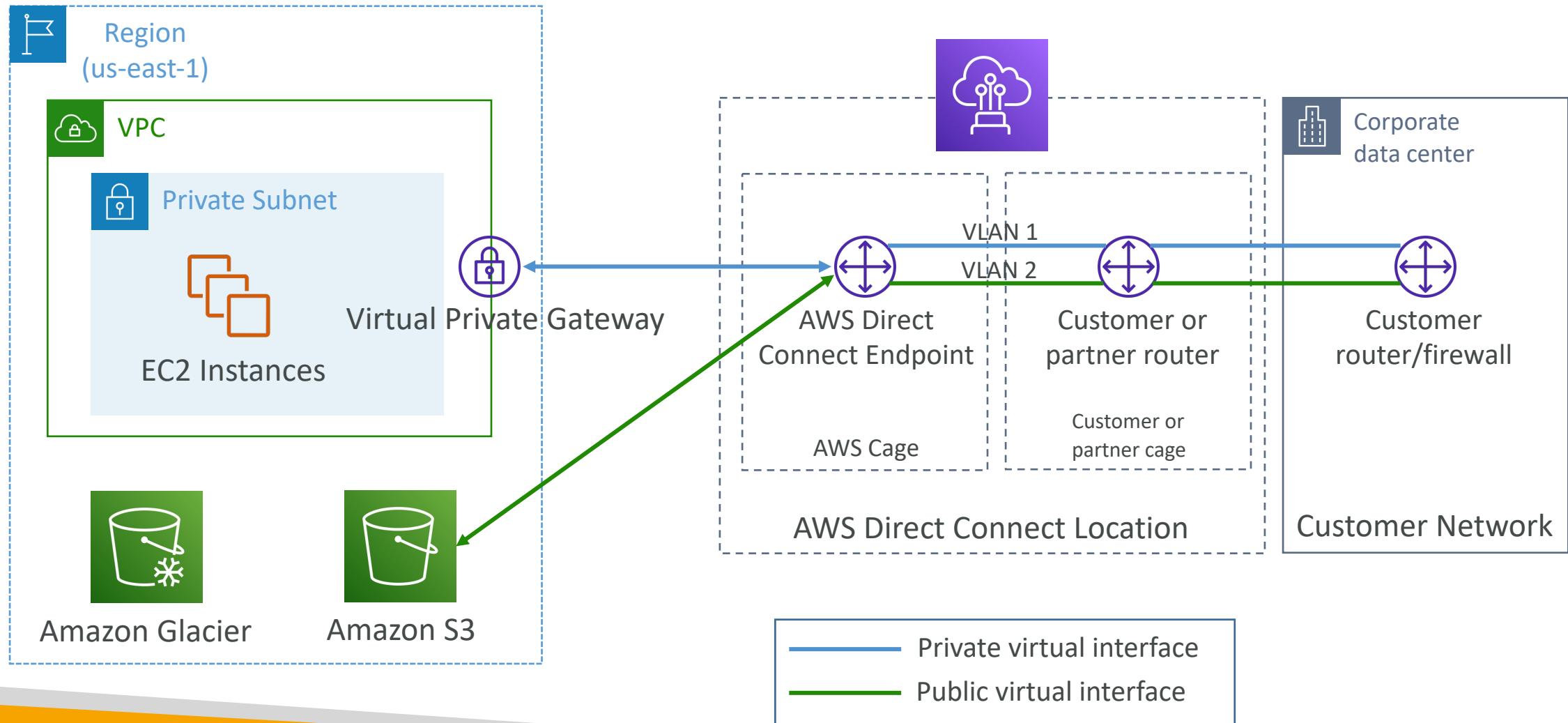




# Direct Connect (DX)

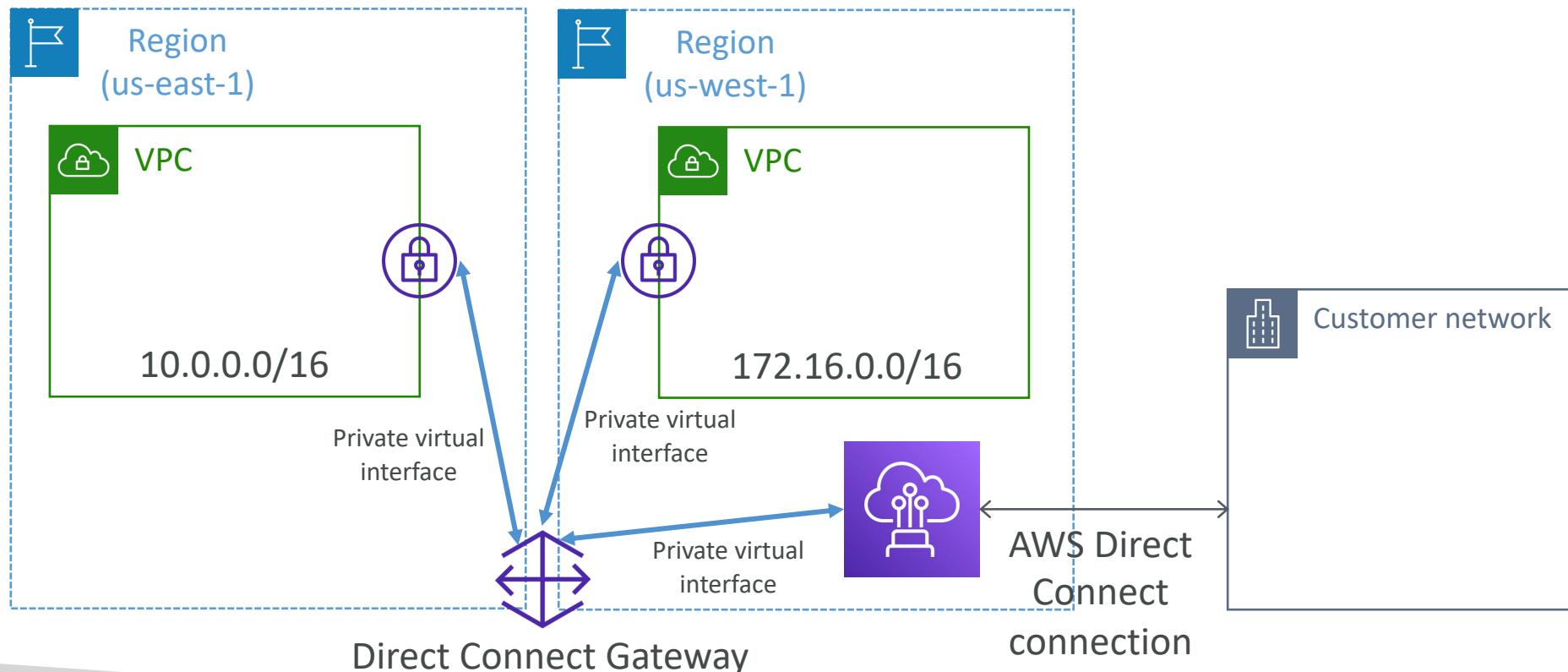
- Provides a dedicated private connection from a remote network to your VPC
- Dedicated connection must be setup between your DC and AWS Direct Connect locations
- You need to setup a Virtual Private Gateway on your VPC
- Access public resources (S3) and private (EC2) on same connection
- Use Cases:
  - Increase bandwidth throughput - working with large data sets – lower cost
  - More consistent network experience - applications using real-time data feeds
  - Hybrid Environments (on prem + cloud)
- Supports both IPv4 and IPv6

# Direct Connect Diagram



# Direct Connect Gateway

- If you want to setup a Direct Connect to one or more VPC in many different regions (same account), you must use a Direct Connect Gateway

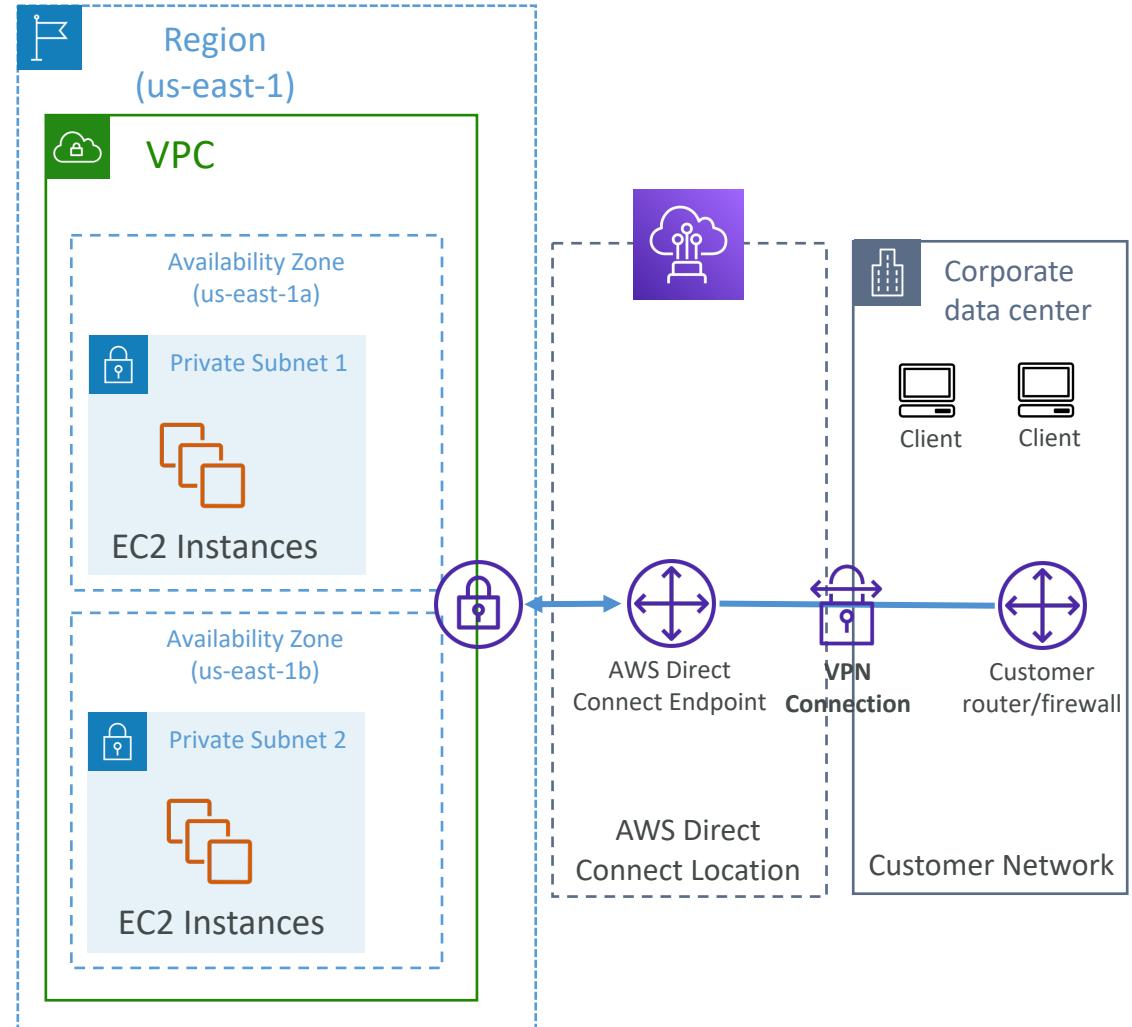


# Direct Connect – Connection Types

- **Dedicated Connections:** 1 Gbps and 10 Gbps capacity
  - Physical ethernet port dedicated to a customer
  - Request made to AWS first, then completed by AWS Direct Connect Partners
- **Hosted Connections:** 50Mbps, 500 Mbps, to 10 Gbps
  - Connection requests are made via AWS Direct Connect Partners
  - Capacity can be **added or removed on demand**
  - 1, 2, 5, 10 Gbps available at select AWS Direct Connect Partners
- Lead times are often longer than 1 month to establish a new connection

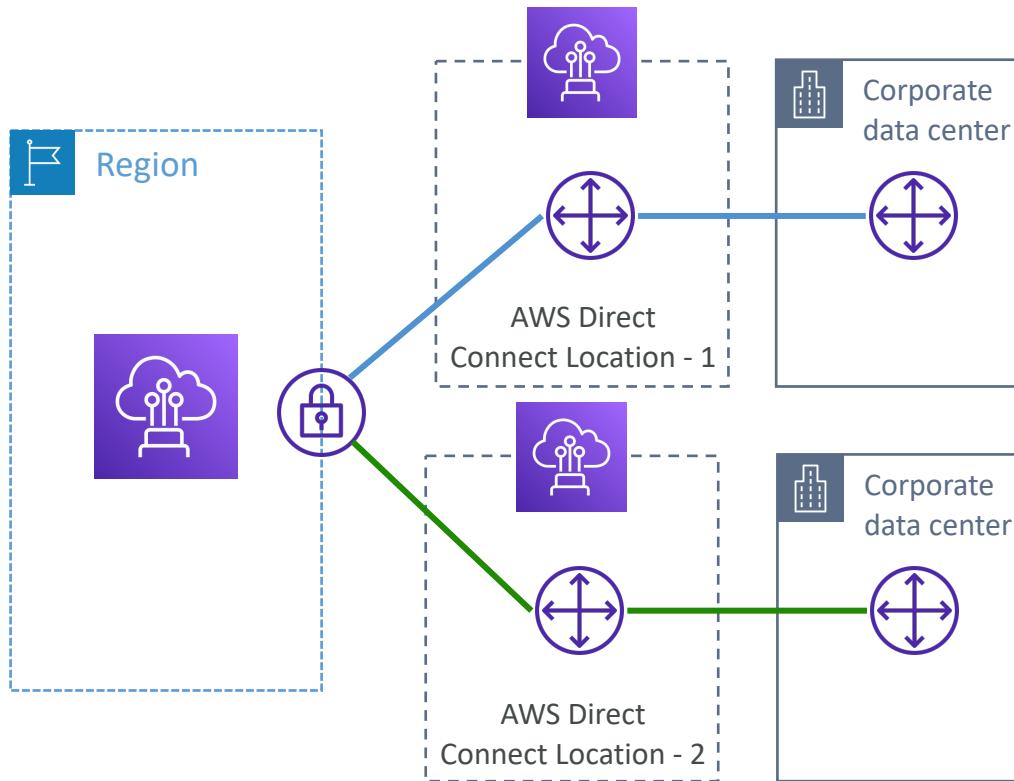
# Direct Connect – Encryption

- Data in transit is not encrypted but is private
- AWS Direct Connect + VPN provides an IPsec-encrypted private connection
- Good for an extra level of security, but slightly more complex to put in place



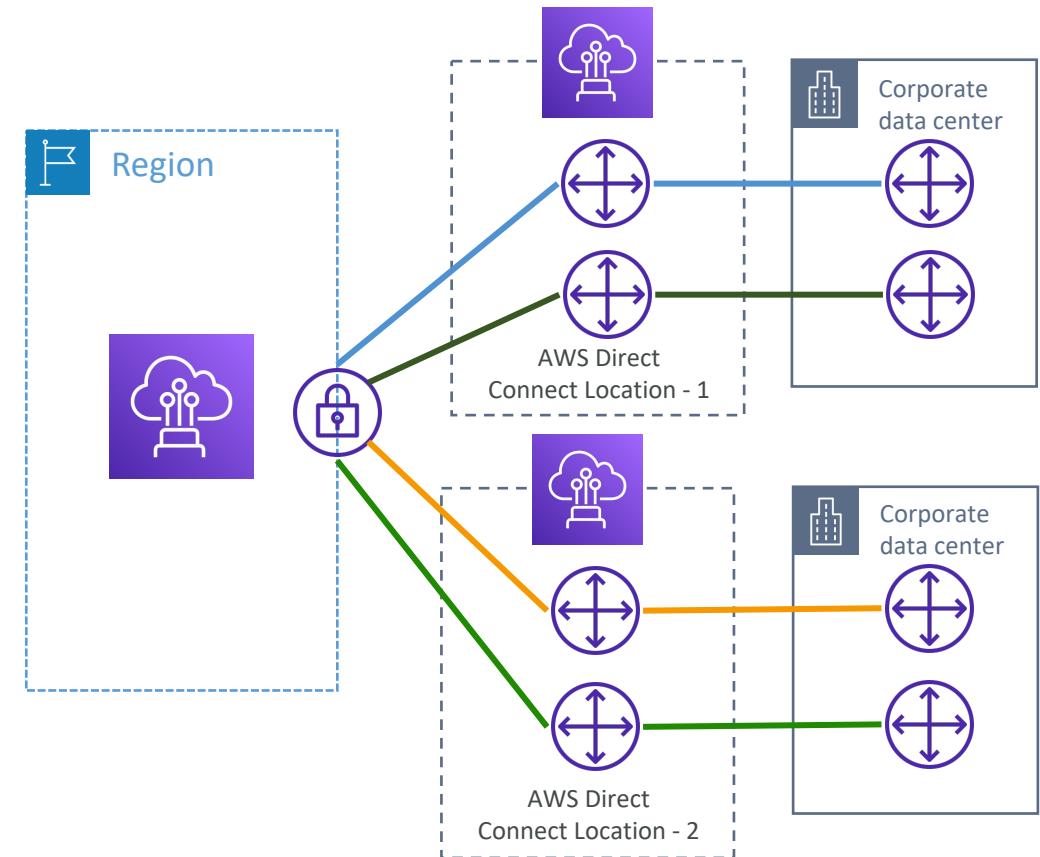
# Direct Connect - Resiliency

High Resiliency for Critical Workloads



One connection at multiple locations

Maximum Resiliency for Critical Workloads

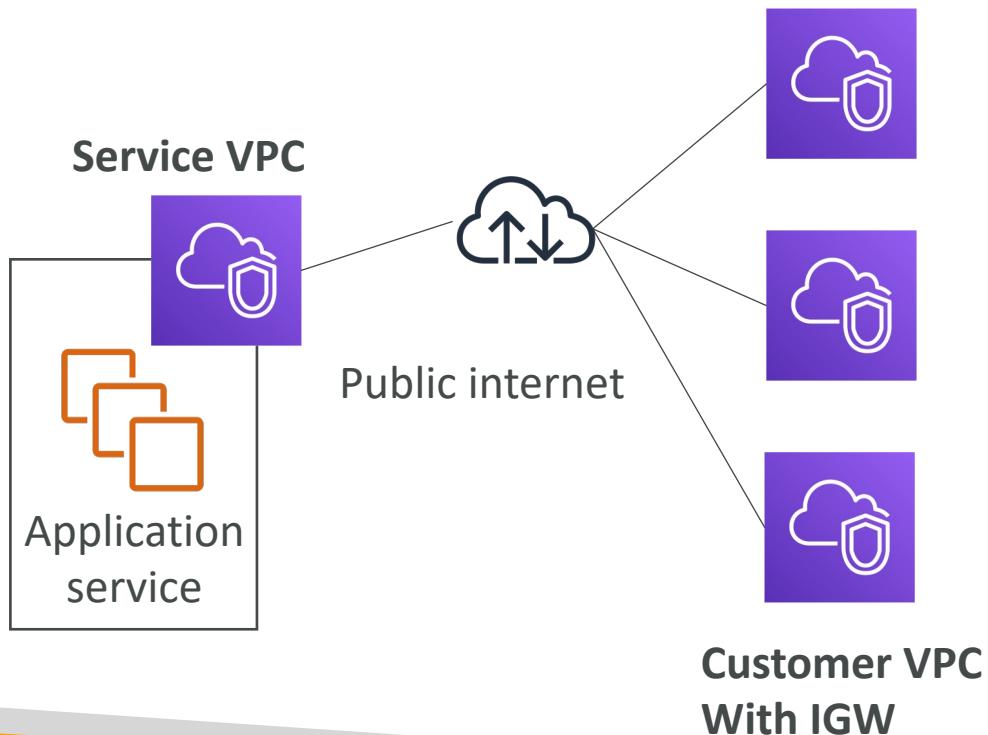


Maximum resilience is achieved by separate connections terminating on separate devices in more than one location.

# Exposing services in your VPC to other VPC

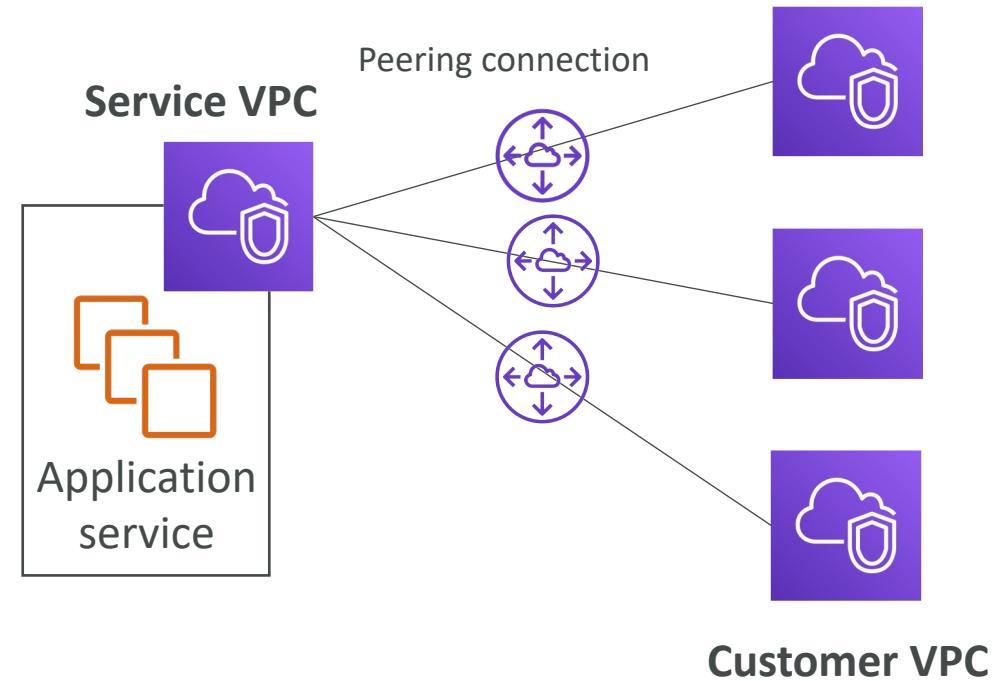
- Option 1: make it public

- Goes through the public www
- Tough to manage access



- Option 2: VPC peering

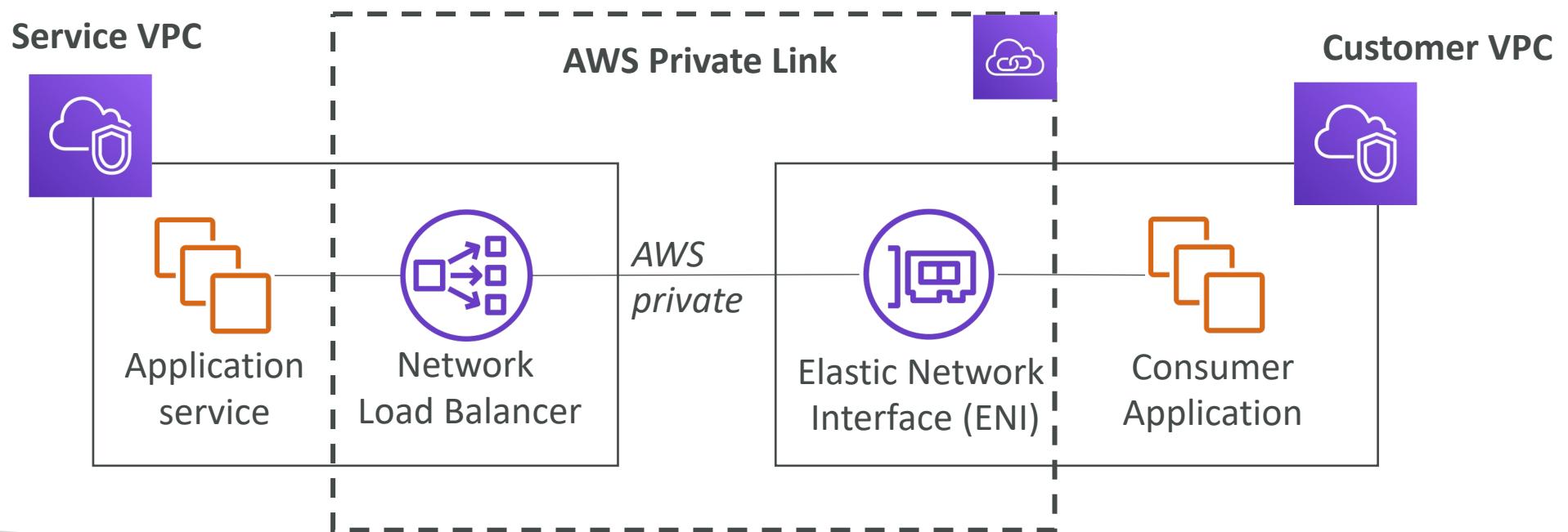
- Must create many peering relations
- Opens the **whole** network



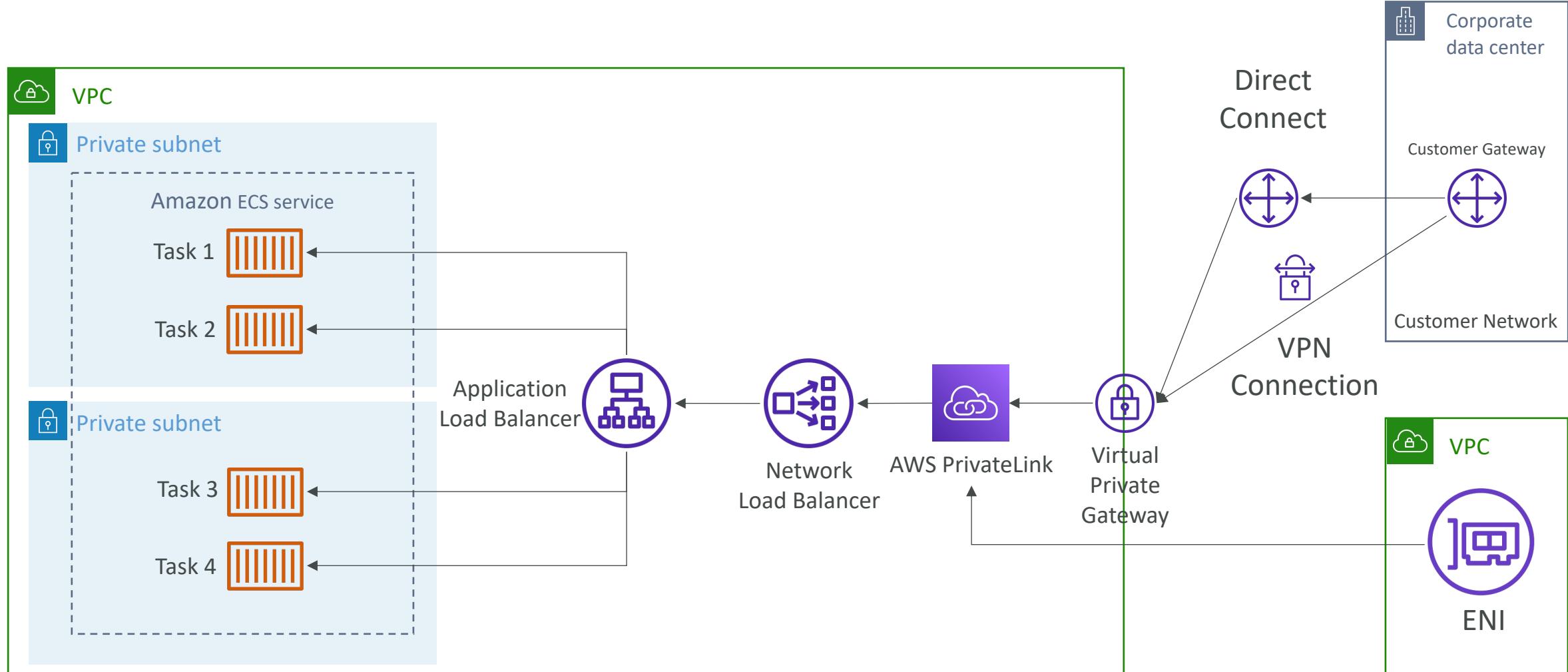
# AWS PrivateLink (VPC Endpoint Services)



- Most secure & scalable way to expose a service to 1000s of VPC (own or other accounts)
- Does not require VPC peering, internet gateway, NAT, route tables...
- Requires a network load balancer (Service VPC) and ENI (Customer VPC) or GWLB
- If the NLB is in multiple AZ, and the ENIs in multiple AZ, the solution is fault tolerant!



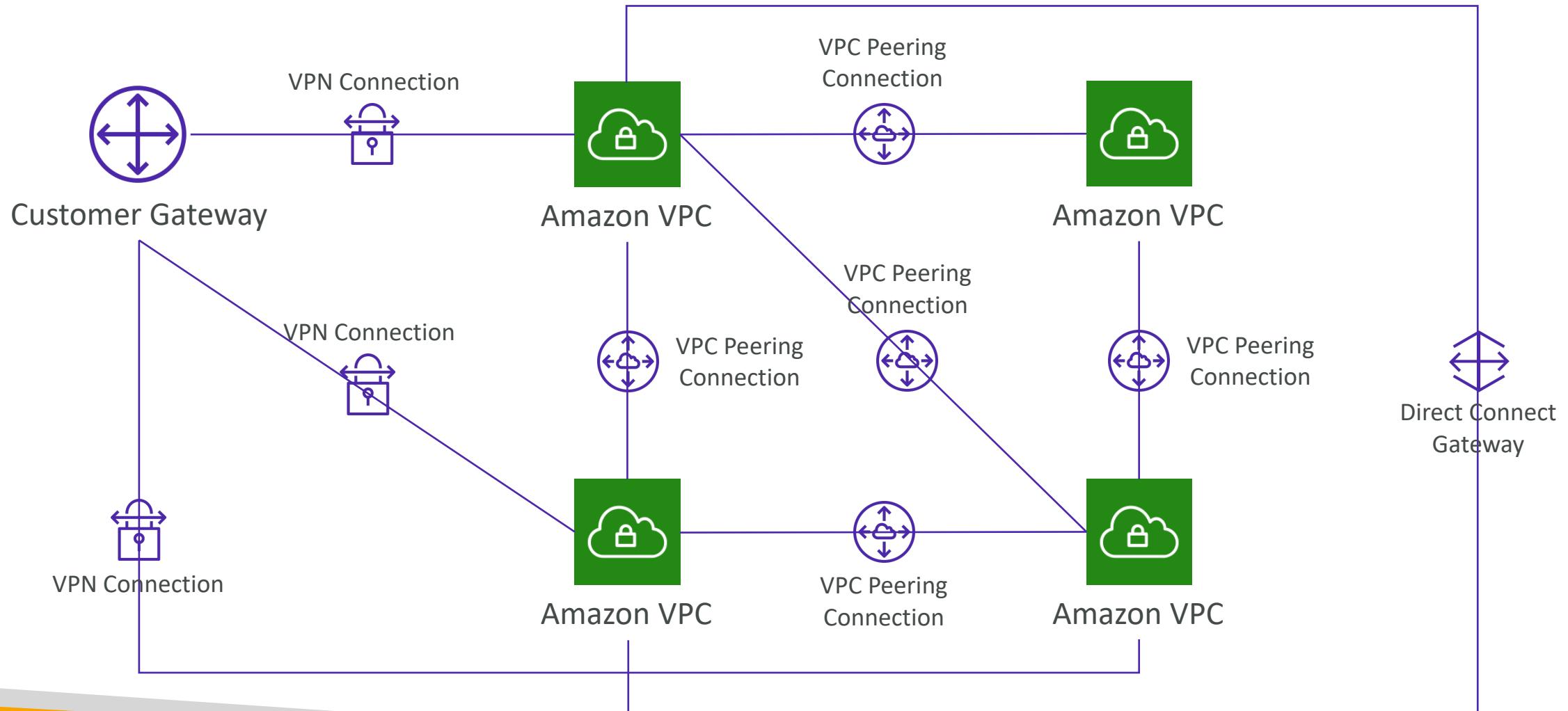
# AWS Private Link & ECS



# EC2-Classic & AWS ClassicLink (deprecated)

- EC2-Classic: instances run in a single network shared with other customers
- Amazon VPC: your instances run logically isolated to your AWS account
- ClassicLink allows you to link EC2-Classic instances to a VPC in your account
  - Must associate a security group
  - Enables communication using private IPv4 addresses
  - Removes the need to make use of public IPv4 addresses or Elastic IP addresses
- Likely to be distractors at the exam

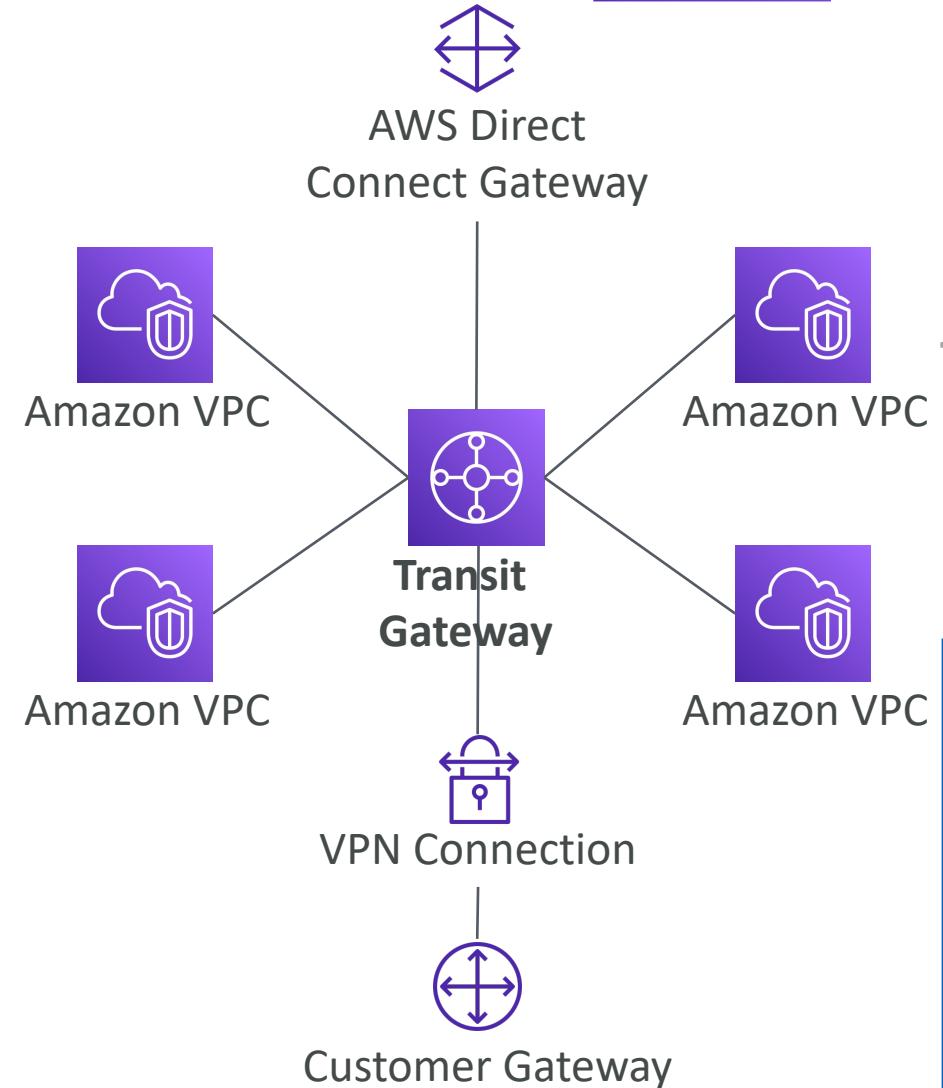
# Network topologies can become complicated



# Transit Gateway

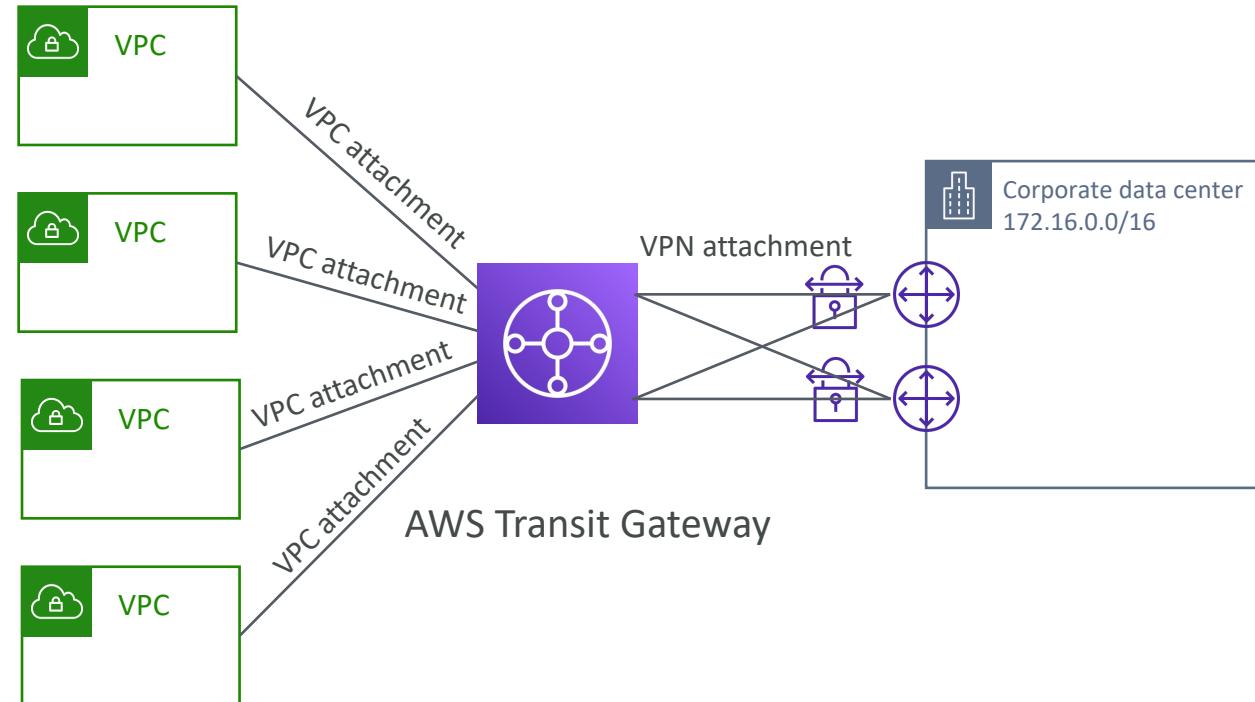


- For having transitive peering between thousands of VPC and on-premises, hub-and-spoke (star) connection
- Regional resource, can work cross-region
- Share cross-account using Resource Access Manager (RAM)
- You can peer Transit Gateways across regions
- Route Tables: limit which VPC can talk with other VPC
- Works with Direct Connect Gateway, VPN connections
- Supports IP Multicast (not supported by any other AWS service)



# Transit Gateway: Site-to-Site VPN ECMP

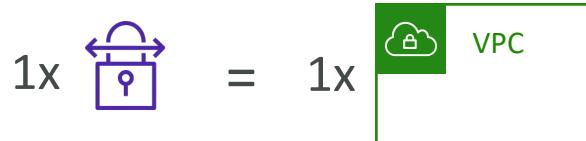
- ECMP = Equal-cost multi-path routing
- Routing strategy to allow to forward a packet over multiple best path
- Use case: create multiple Site-to-Site VPN connections to increase the bandwidth of your connection to AWS



# Transit Gateway: throughput with ECMP



VPN to virtual private gateway



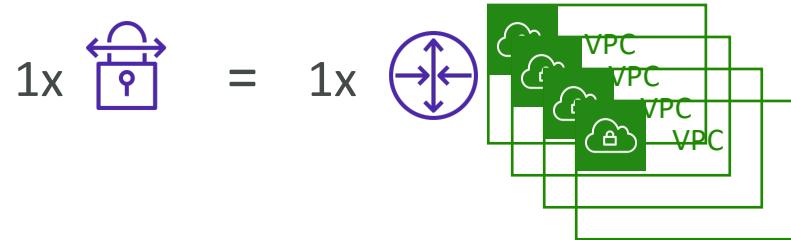
1x = 1.25 Gbps



VPN connection  
(2 tunnels)



VPN to transit gateway



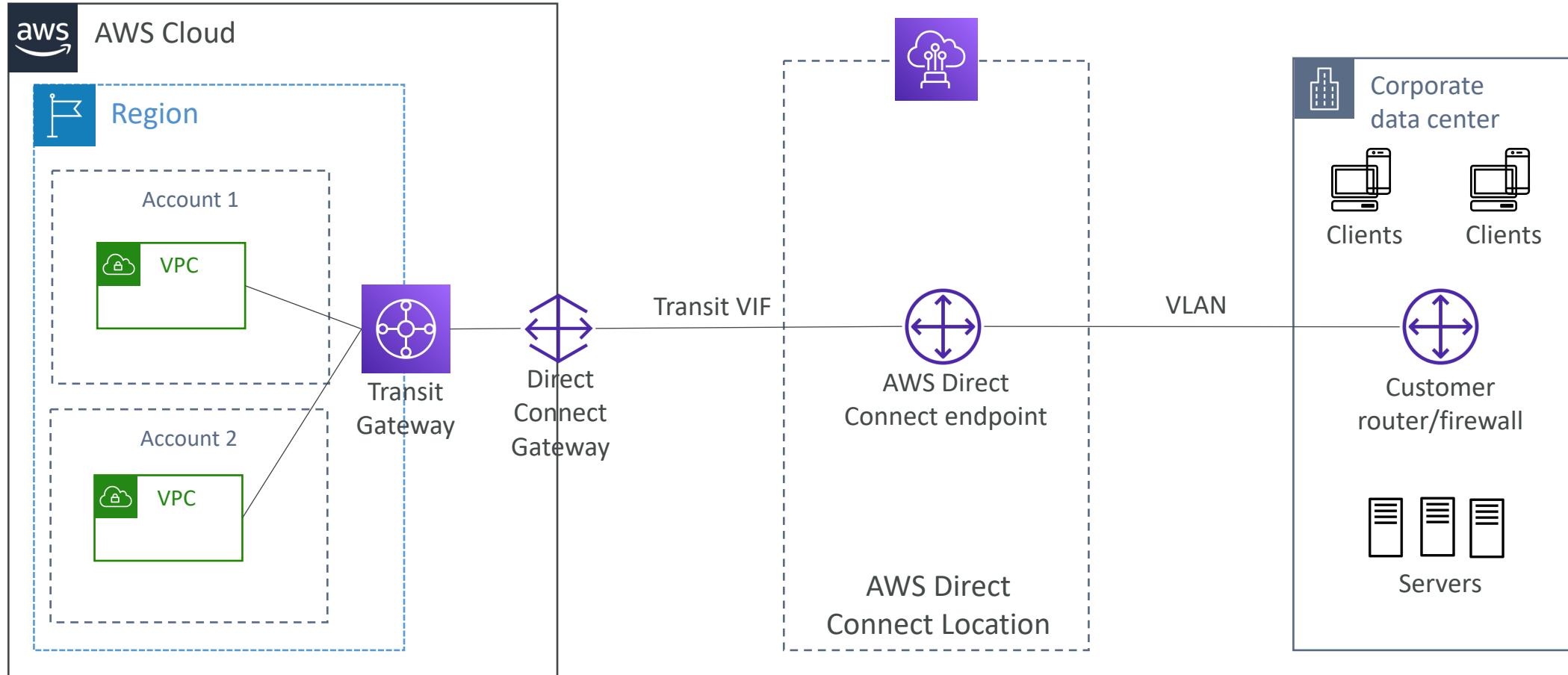
1x = 2.5 Gbps (ECMP) – 2 tunnels used

2x = 5.0 Gbps (ECMP)

3x = 7.5 Gbps (ECMP)

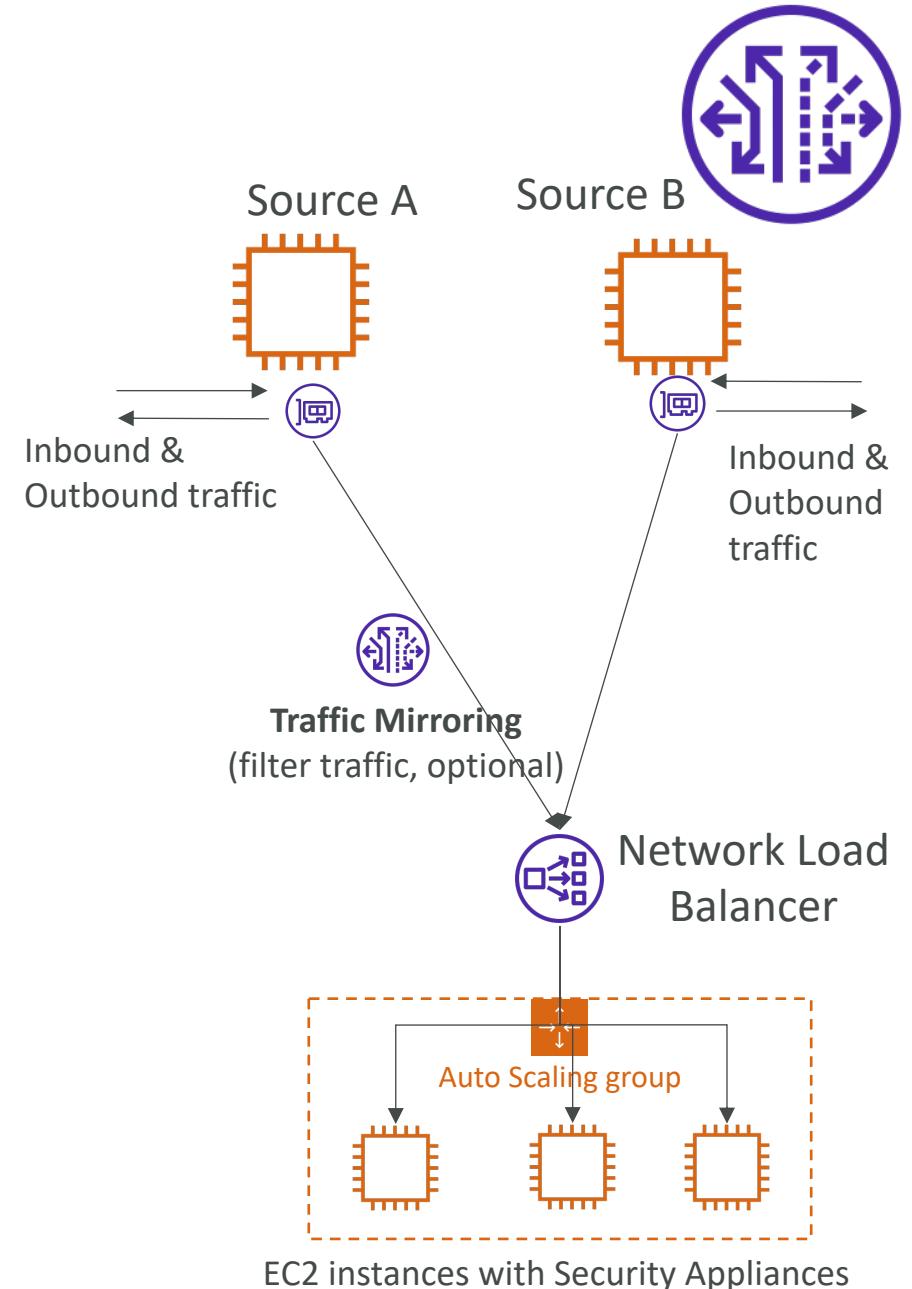
+\$\$ per GB of TGW  
processed data

# Transit Gateway – Share Direct Connect between multiple accounts



# VPC – Traffic Mirroring

- Allows you to capture and inspect network traffic in your VPC
- Route the traffic to security appliances that you manage
- Capture the traffic
  - From (Source) – ENIs
  - To (Targets) – an ENI or a Network Load Balancer
- Capture all packets or capture the packets of your interest (optionally, truncate packets)
- Source and Target can be in the same VPC or different VPCs (VPC Peering)
- Use cases: content inspection, threat monitoring, troubleshooting, ...

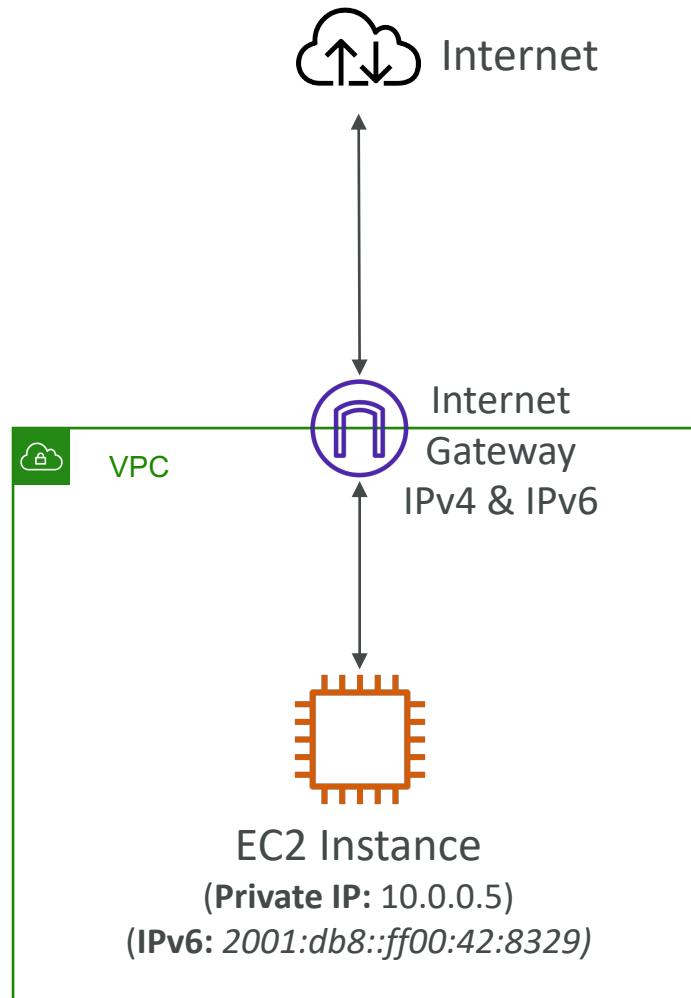


# What is IPv6?

- IPv4 designed to provide 4.3 Billion addresses (they'll be exhausted soon)
- IPv6 is the successor of IPv4
- IPv6 is designed to provide  $3.4 \times 10^{38}$  unique IP addresses
- Every IPv6 address is public and Internet-routable (no private range)
- Format → x.x.x.x.x.x.x.x (x is hexadecimal, range can be from 0000 to ffff)
- Examples:
  - 2001:db8:3333:4444:5555:6666:7777:8888
  - 2001:db8:3333:4444:cccc:dddd:eeee:ffff
  - :: → all 8 segments are zero
  - 2001:db8:: → the last 6 segments are zero
  - ::1234:5678 → the first 6 segments are zero
  - 2001:db8::1234:5678 → the middle 4 segments are zero

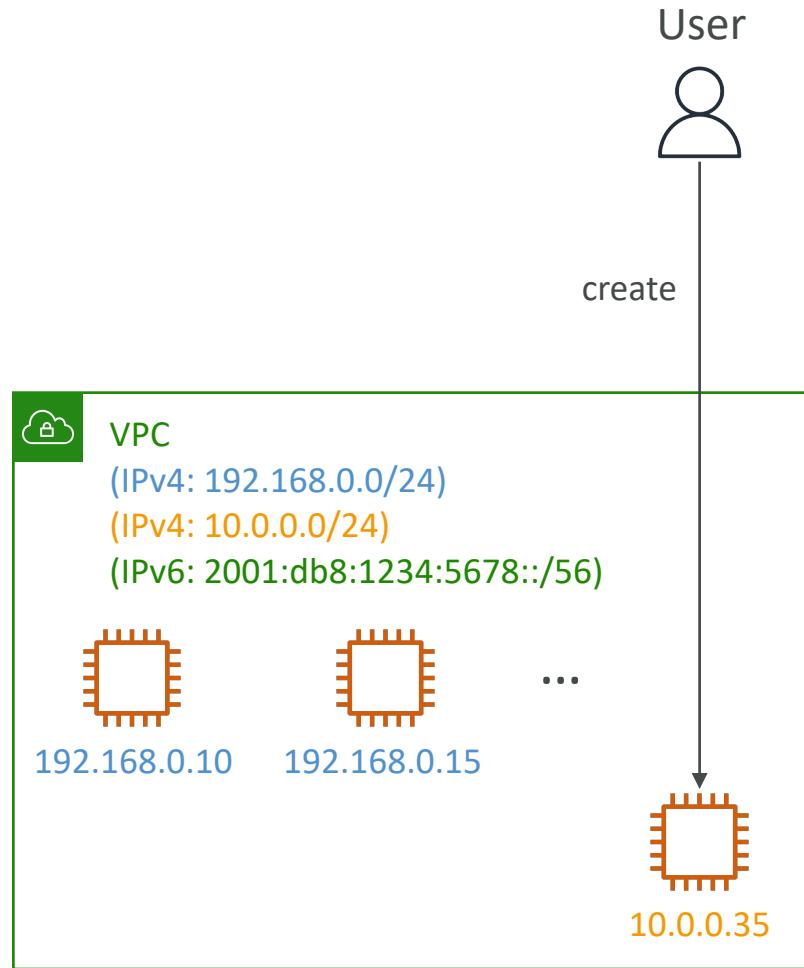
# IPv6 in VPC

- IPv4 cannot be disabled for your VPC and subnets
- You can enable IPv6 (they're public IP addresses) to operate in dual-stack mode
- Your EC2 instances will get at least a private internal IPv4 and a public IPv6
- They can communicate using either IPv4 or IPv6 to the internet through an Internet Gateway

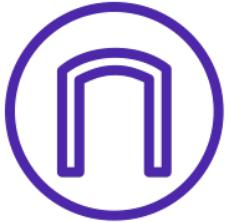


# IPv6 Troubleshooting

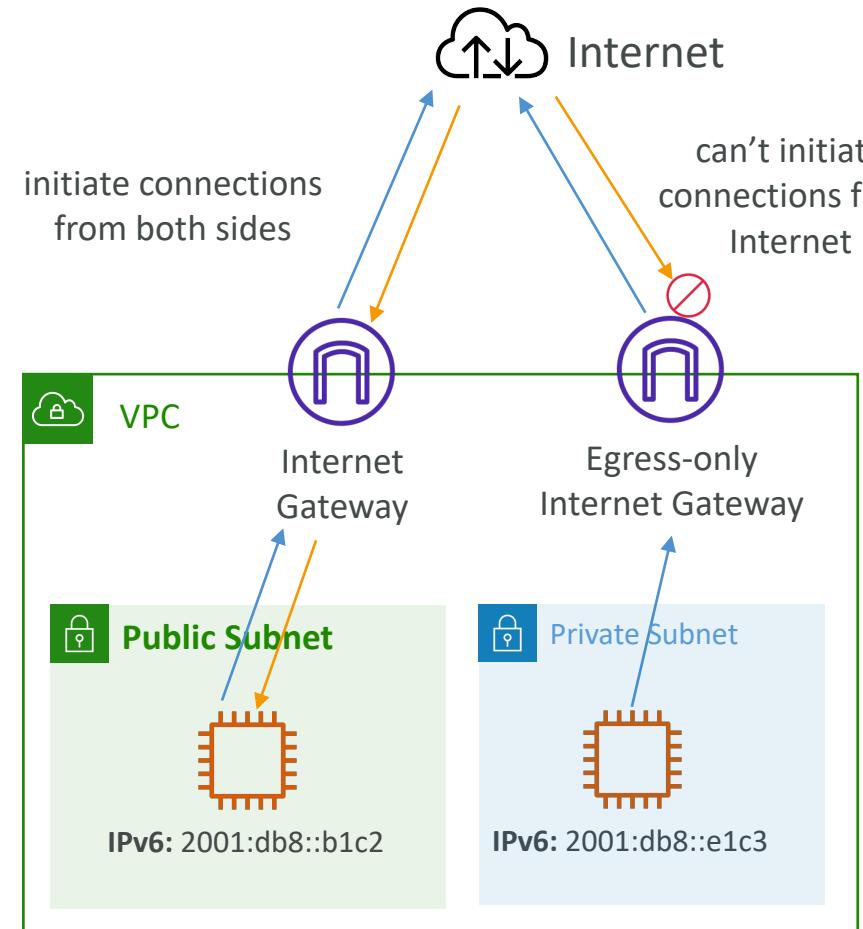
- IPv4 cannot be disabled for your VPC and subnets
- So, if you cannot launch an EC2 instance in your subnet
  - It's not because it cannot acquire an IPv6 (the space is very large)
  - It's because there are no available IPv4 in your subnet
- Solution: create a new IPv4 CIDR in your subnet



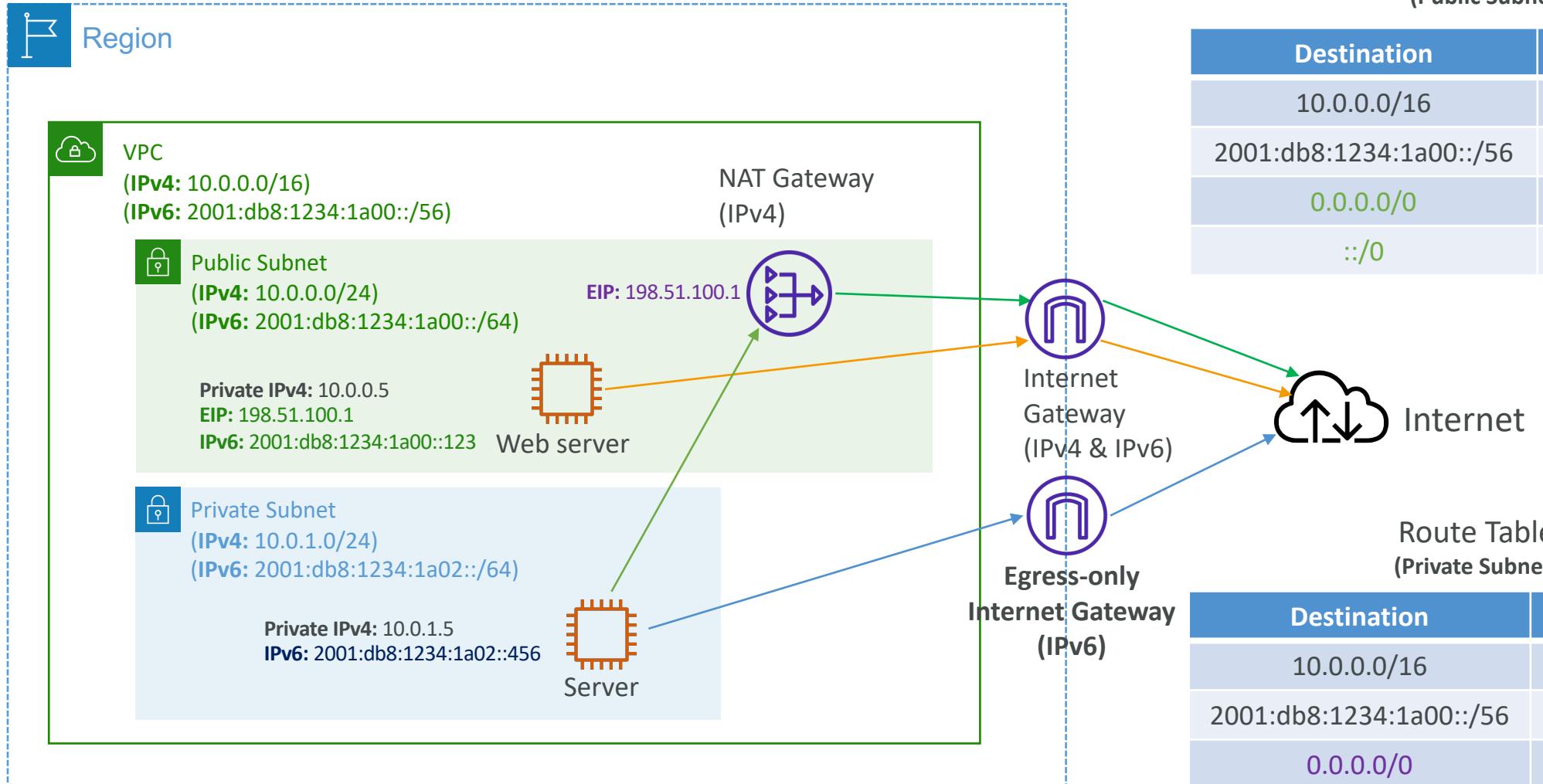
# Egress-only Internet Gateway



- Used for IPv6 only
- (similar to a NAT Gateway but for IPv6)
- Allows instances in your VPC outbound connections over IPv6 while preventing the internet to initiate an IPv6 connection to your instances
- You must update the Route Tables



# IPv6 Routing



# VPC Section Summary (1/3)

- CIDR – IP Range
- VPC – Virtual Private Cloud => we define a list of IPv4 & IPv6 CIDR
- Subnets – tied to an AZ, we define a CIDR
- Internet Gateway – at the VPC level, provide IPv4 & IPv6 Internet Access
- Route Tables – must be edited to add routes from subnets to the IGW,VPC Peering Connections,VPC Endpoints, ...
- Bastion Host – public EC2 instance to SSH into, that has SSH connectivity to EC2 instances in private subnets
- NAT Instances – gives Internet access to EC2 instances in private subnets. Old, must be setup in a public subnet, disable Source / Destination check flag
- NAT Gateway – managed by AWS, provides scalable Internet access to private EC2 instances, IPv4 only
- Private DNS + Route 53 – enable DNS Resolution + DNS Hostnames (VPC)

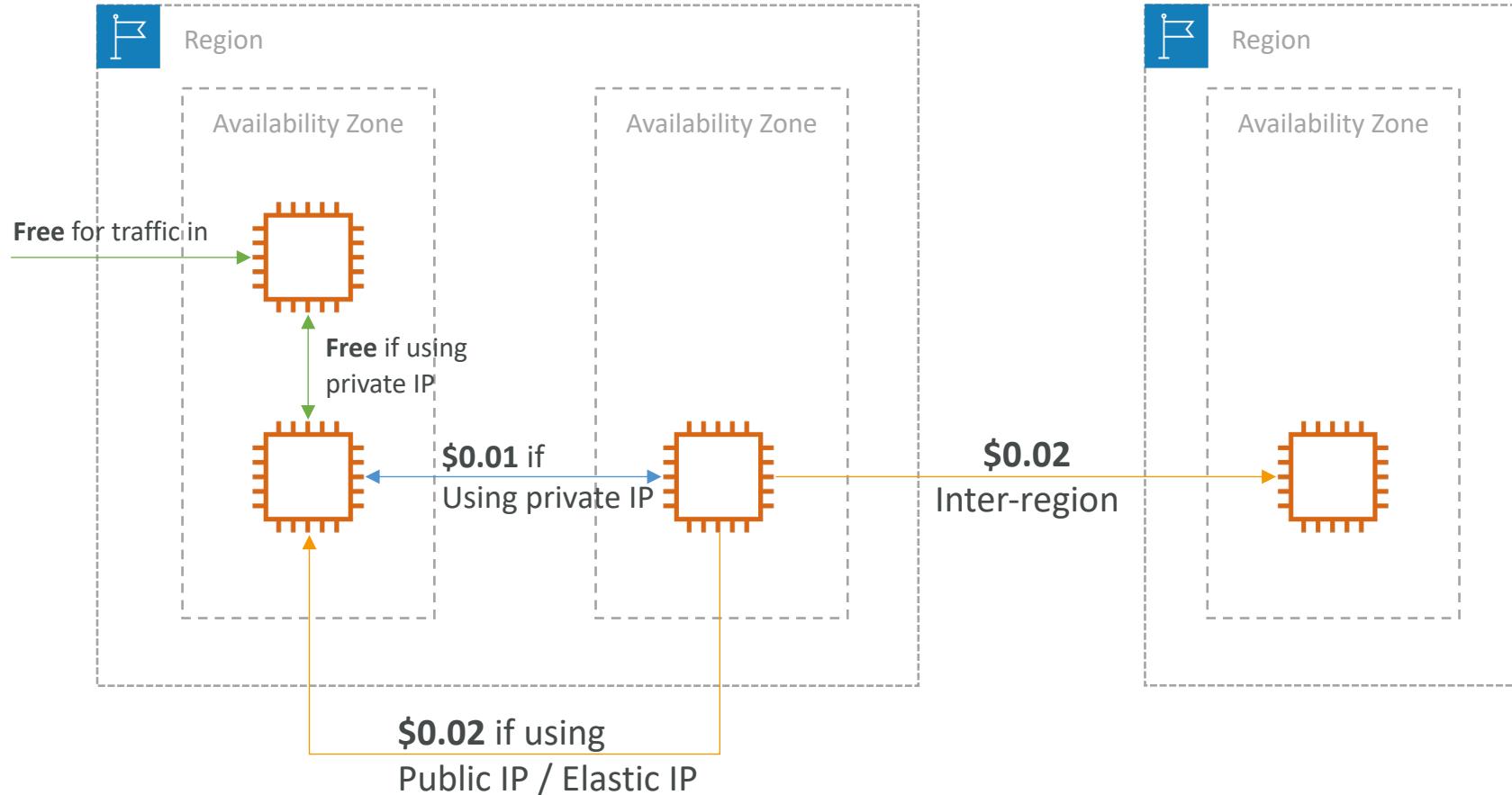
# VPC Section Summary (2/3)

- **NACL** – stateless, subnet rules for inbound and outbound, don't forget Ephemeral Ports
- **Security Groups** – stateful, operate at the EC2 instance level
- **Reachability Analyzer** – perform network connectivity testing between AWS resources
- **VPC Peering** – connect two VPCs with non overlapping CIDR, non-transitive
- **VPC Endpoints** – provide private access to AWS Services (S3, DynamoDB, CloudFormation, SSM) within a VPC
- **VPC Flow Logs** – can be setup at the VPC / Subnet / ENI Level, for ACCEPT and REJECT traffic, helps identifying attacks, analyze using Athena or CloudWatch Logs Insights
- **Site-to-Site VPN** – setup a Customer Gateway on DC, a Virtual Private Gateway on VPC, and site-to-site VPN over public Internet
- **AWS VPN CloudHub** – hub-and-spoke VPN model to connect your sites

# VPC Section Summary (3/3)

- **Direct Connect** – setup a Virtual Private Gateway on VPC, and establish a direct private connection to an AWS Direct Connect Location
- **Direct Connect Gateway** – setup a Direct Connect to many VPCs in different AWS regions
- **AWS PrivateLink / VPC Endpoint Services:**
  - Connect services privately from your service VPC to customers VPC
  - Doesn't need VPC Peering, public Internet, NAT Gateway, Route Tables
  - Must be used with Network Load Balancer & ENI
- **ClassicLink** – connect EC2-Classic EC2 instances privately to your VPC
- **Transit Gateway** – transitive peering connections for VPC, VPN & DX
- **Traffic Mirroring** – copy network traffic from ENIs for further analysis
- **Egress-only Internet Gateway** – like a NAT Gateway, but for IPv6

# Networking Costs in AWS per GB - Simplified

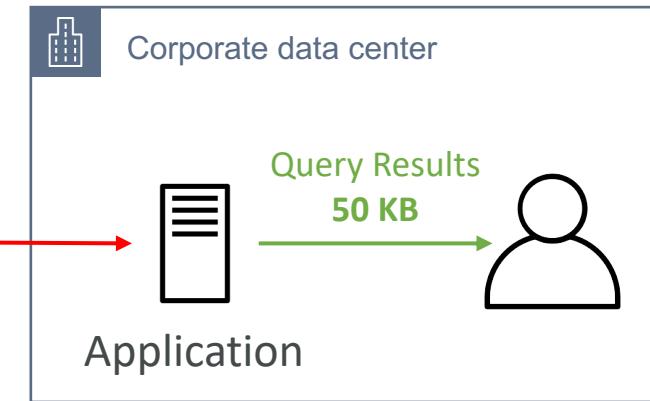
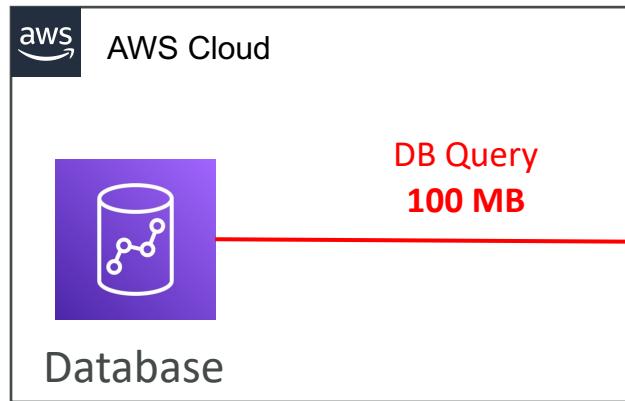


- Use Private IP instead of Public IP for good savings and better network performance
- Use same AZ for maximum savings (at the cost of high availability)

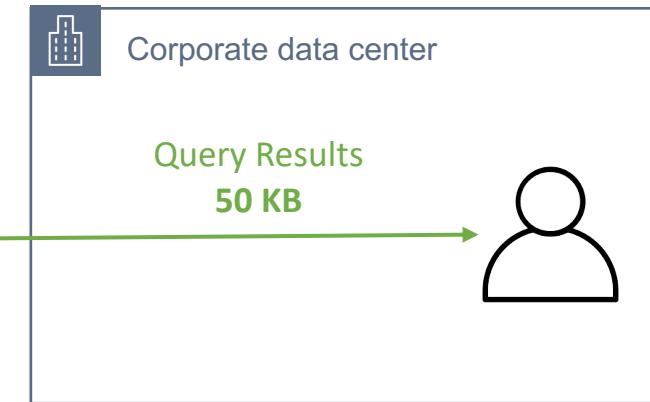
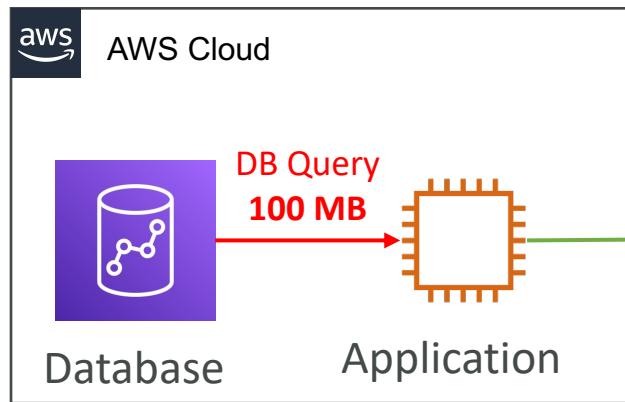
# Minimizing egress traffic network cost

- Egress traffic: outbound traffic (from AWS to outside)
- Ingress traffic: inbound traffic - from outside to AWS (typically free)
- Try to keep as much internet traffic within AWS to minimize costs
- Direct Connect locations that are co-located in the same AWS Region result in lower cost for egress network

**Egress cost is high**

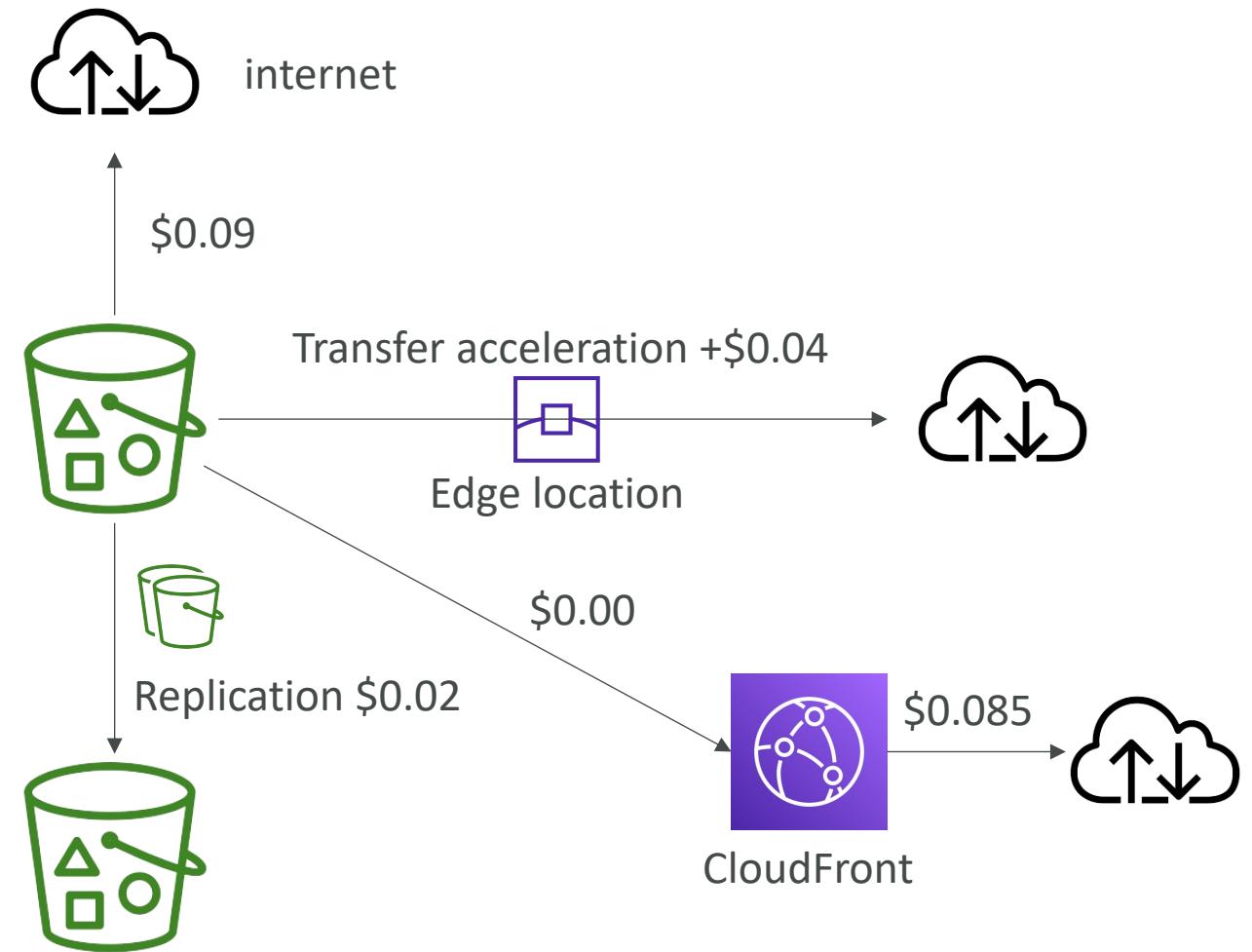


**Egress cost is minimized**



# S3 Data Transfer Pricing – Analysis for USA

- S3 ingress: free
- S3 to Internet: \$0.09 per GB
- S3 Transfer Acceleration:
  - Faster transfer times (50 to 500% better)
  - Additional cost on top of Data Transfer Pricing: +\$0.04 to \$0.08 per GB
- S3 to CloudFront: \$0.00 per GB
- CloudFront to Internet: \$0.085 per GB (slightly cheaper than S3)
  - Caching capability (lower latency)
  - Reduce costs associated with S3 Requests Pricing (7x cheaper with CloudFront)
- S3 Cross Region Replication: \$0.02 per GB



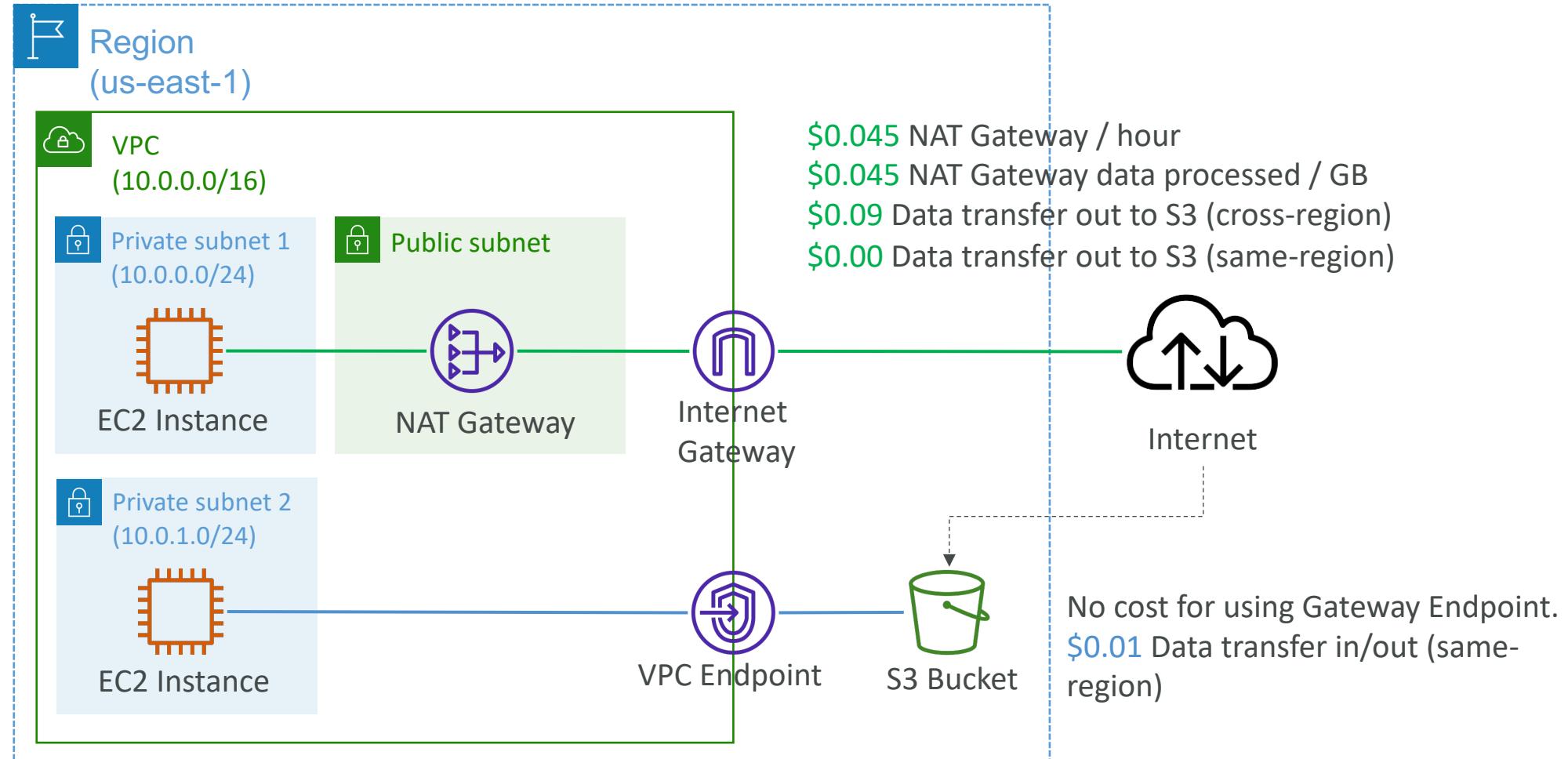
# Pricing: NAT Gateway vs Gateway VPC Endpoint

Subnet 1 route table

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-id

Subnet 2 route table

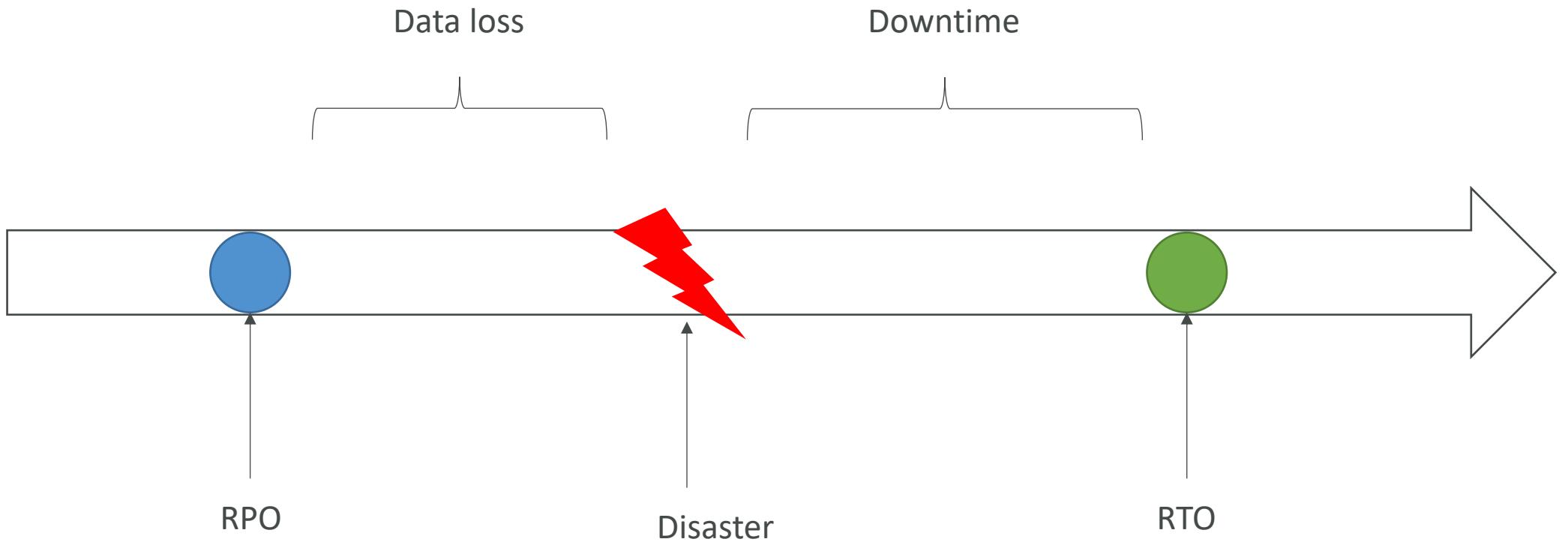
Destination	Target
10.0.0.0/16	Local
pl-id for Amazon S3	vpce-id



# Disaster Recovery Overview

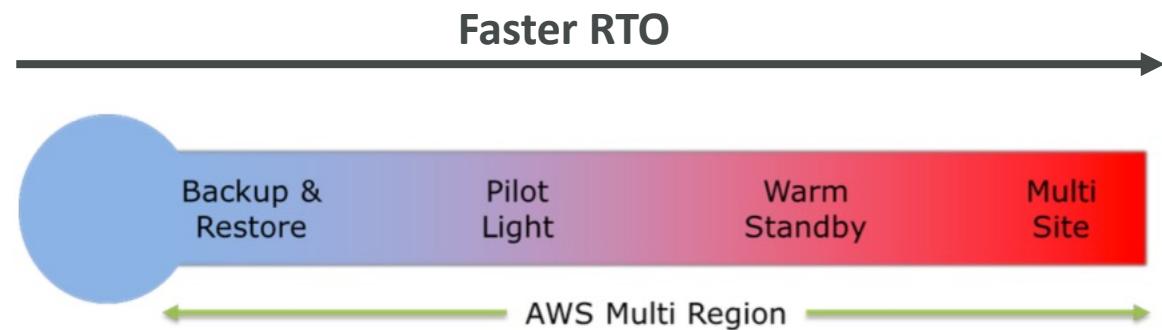
- Any event that has a negative impact on a company's business continuity or finances is a disaster
- Disaster recovery (DR) is about preparing for and recovering from a disaster
- What kind of disaster recovery?
  - On-premise => On-premise: traditional DR, and very expensive
  - On-premise => AWS Cloud: hybrid recovery
  - AWS Cloud Region A => AWS Cloud Region B
- Need to define two terms:
  - RPO: Recovery Point Objective
  - RTO: Recovery Time Objective

# RPO and RTO

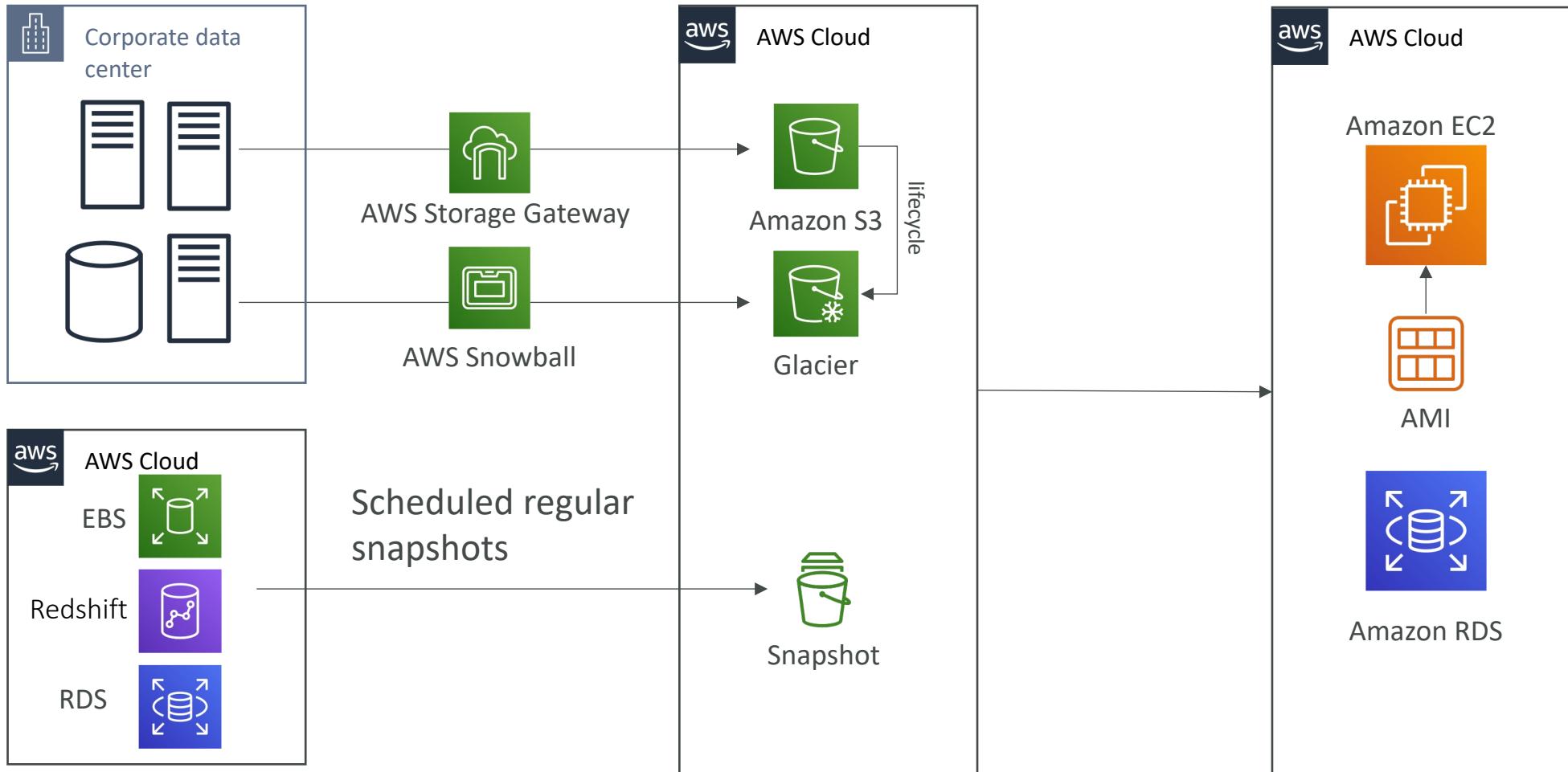


# Disaster Recovery Strategies

- Backup and Restore
- Pilot Light
- Warm Standby
- Hot Site / Multi Site Approach

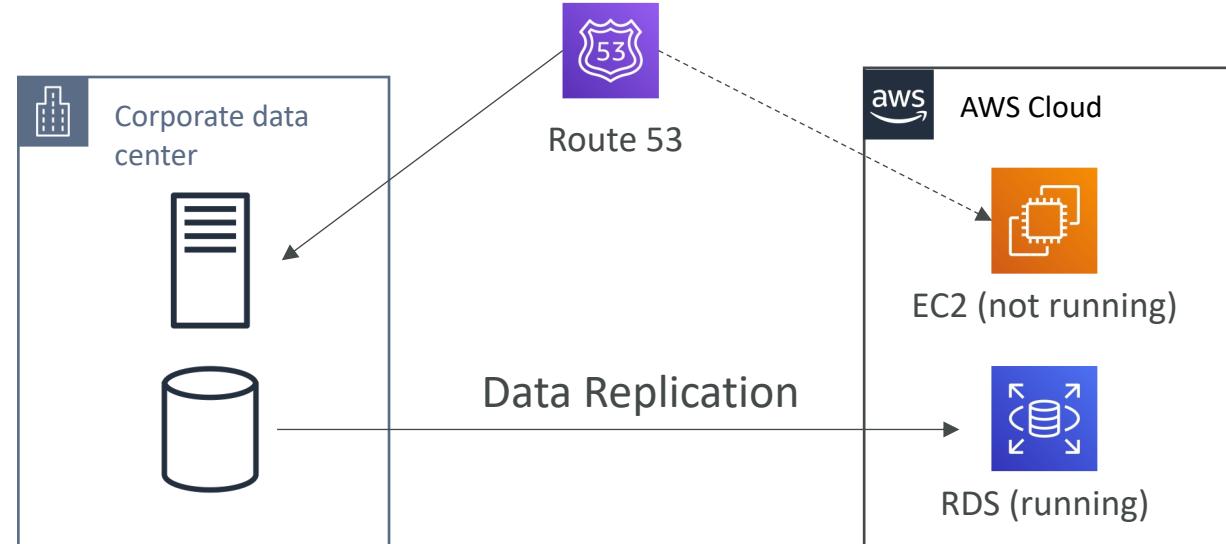


# Backup and Restore (High RPO)



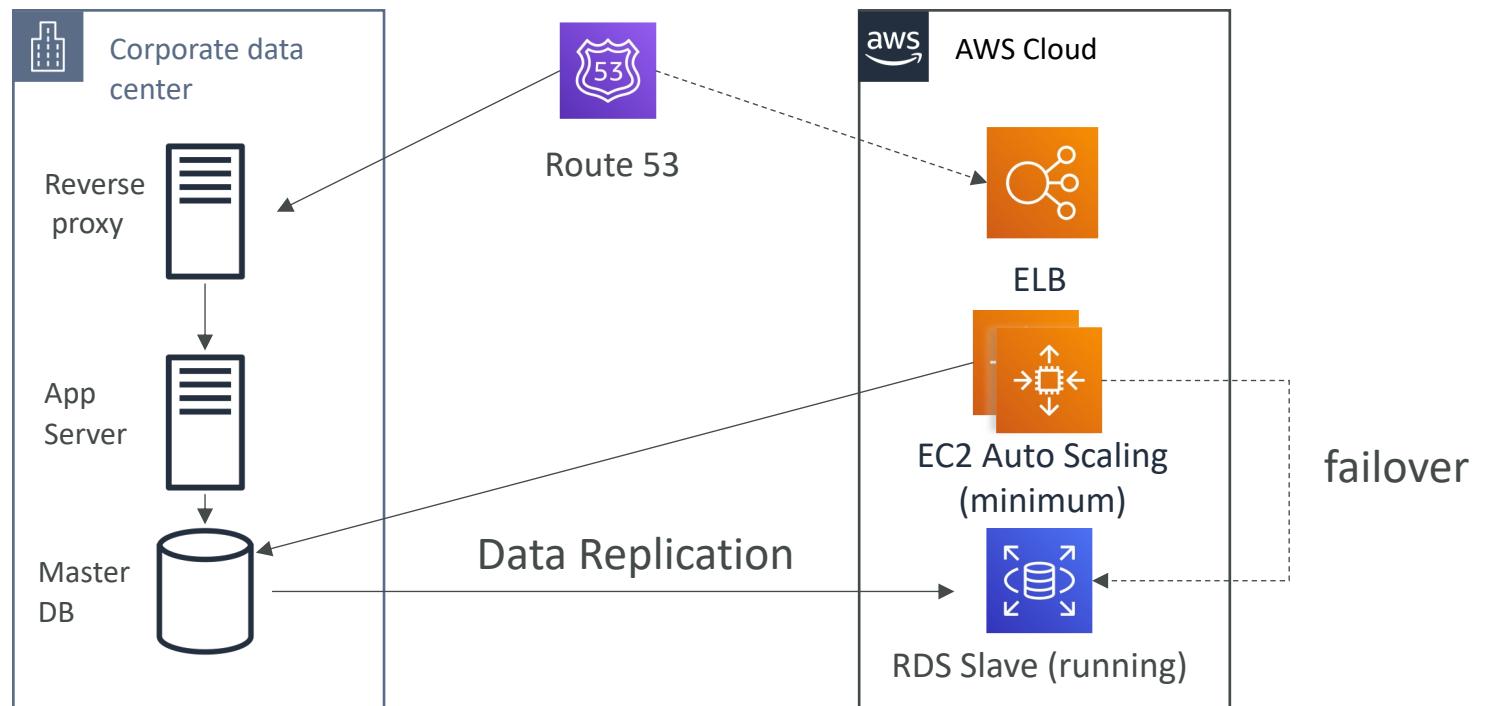
# Disaster Recovery – Pilot Light

- A small version of the app is always running in the cloud
- Useful for the critical core (pilot light)
- Very similar to Backup and Restore
- Faster than Backup and Restore as critical systems are already up



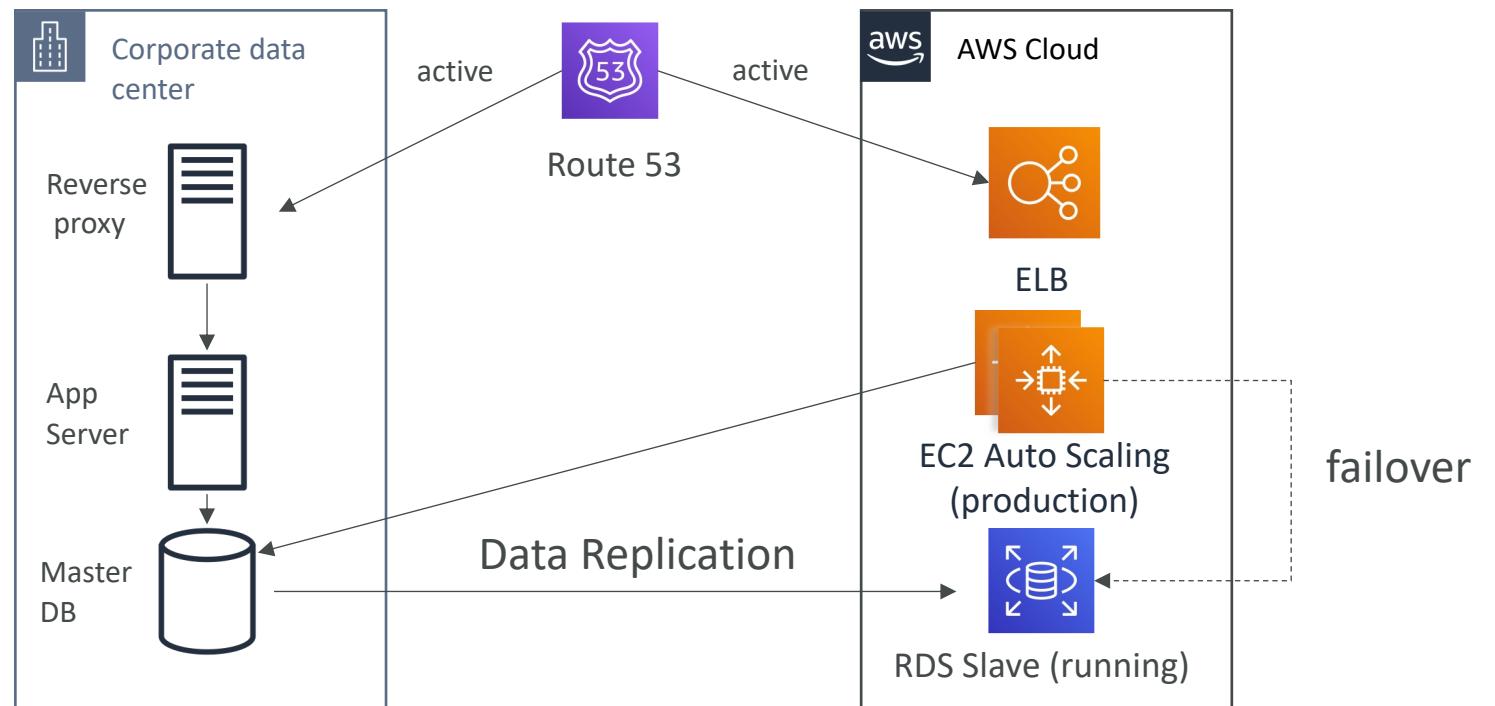
# Warm Standby

- Full system is up and running, but at minimum size
- Upon disaster, we can scale to production load

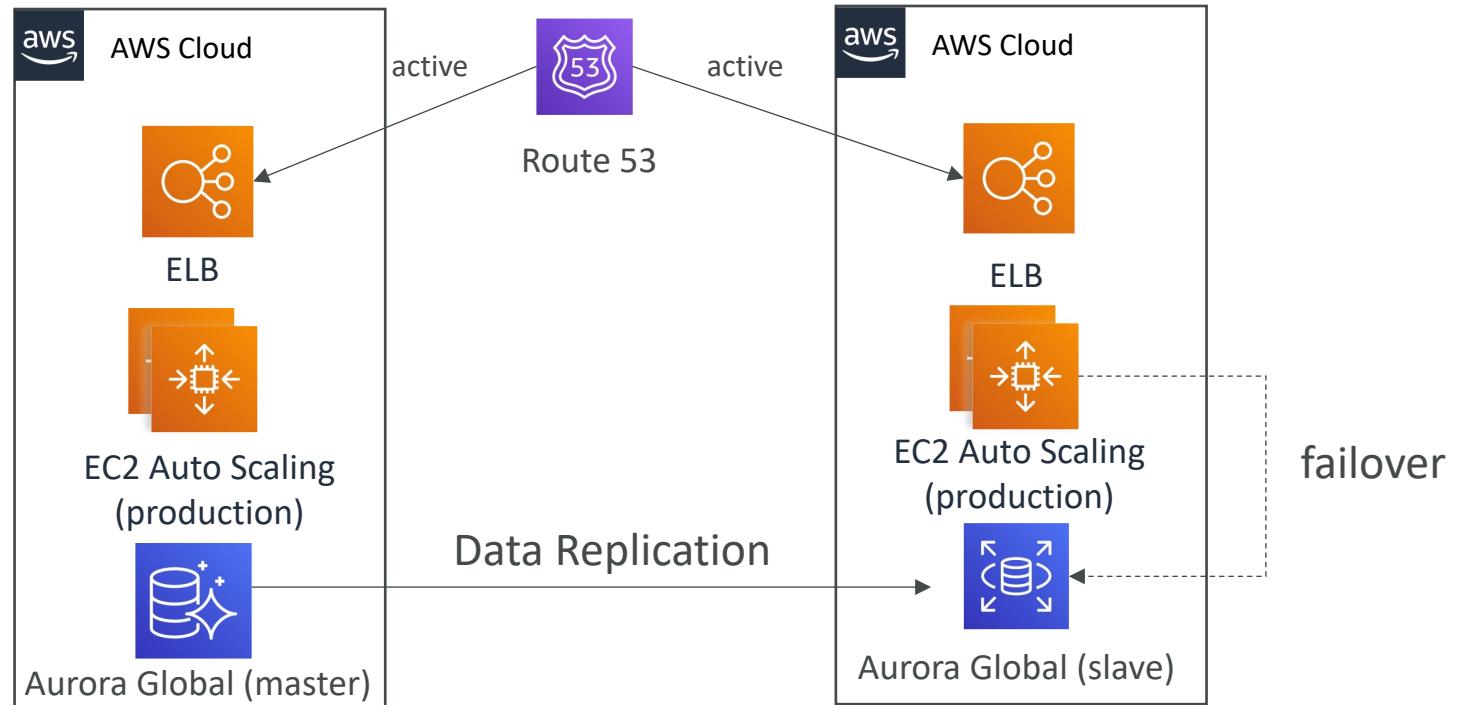


# Multi Site / Hot Site Approach

- Very low RTO (minutes or seconds) – very expensive
- Full Production Scale is running AWS and On Premise



# All AWS Multi Region



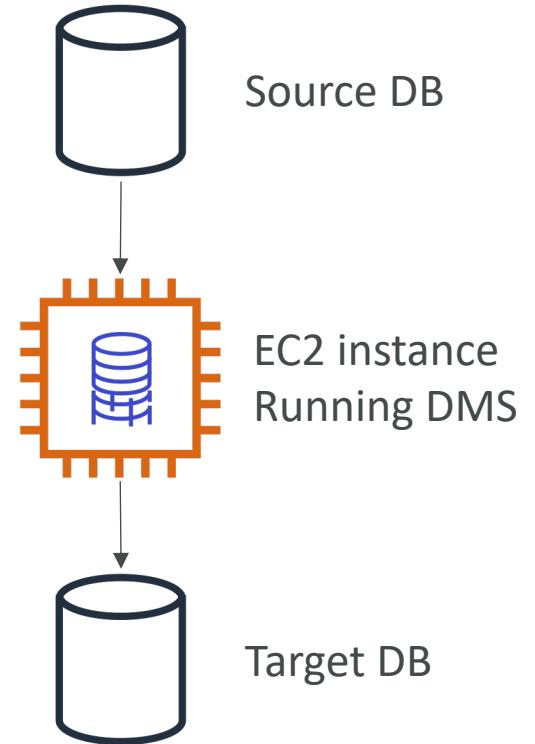
# Disaster Recovery Tips

- **Backup**
  - EBS Snapshots, RDS automated backups / Snapshots, etc...
  - Regular pushes to S3 / S3 IA / Glacier, Lifecycle Policy, Cross Region Replication
  - From On-Premise: Snowball or Storage Gateway
- **High Availability**
  - Use Route53 to migrate DNS over from Region to Region
  - RDS Multi-AZ, ElastiCache Multi-AZ, EFS, S3
  - Site to Site VPN as a recovery from Direct Connect
- **Replication**
  - RDS Replication (Cross Region), AWS Aurora + Global Databases
  - Database replication from on-premises to RDS
  - Storage Gateway
- **Automation**
  - CloudFormation / Elastic Beanstalk to re-create a whole new environment
  - Recover / Reboot EC2 instances with CloudWatch if alarms fail
  - AWS Lambda functions for customized automations
- **Chaos**
  - Netflix has a “simian-army” randomly terminating EC2



# DMS – Database Migration Service

- Quickly and securely migrate databases to AWS, resilient, self healing
- The source database remains available during the migration
- Supports:
  - Homogeneous migrations: ex Oracle to Oracle
  - Heterogeneous migrations: ex Microsoft SQL Server to Aurora
- Continuous Data Replication using CDC
- You must create an EC2 instance to perform the replication tasks



# DMS Sources and Targets

## SOURCES:

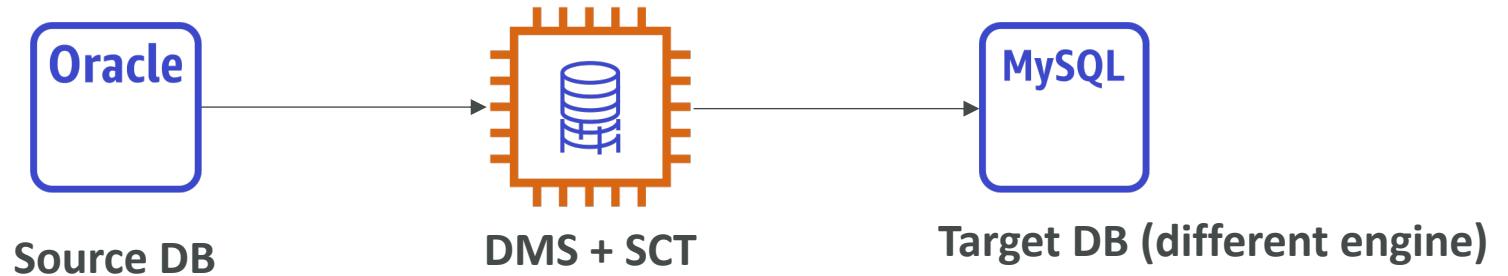
- On-Premise and EC2 instances databases: Oracle, MS SQL Server, MySQL, MariaDB, PostgreSQL, MongoDB, SAP, DB2
- Azure: Azure SQL Database
- Amazon RDS: all including Aurora
- Amazon S3

## TARGETS:

- On-Premise and EC2 instances databases: Oracle, MS SQL Server, MySQL, MariaDB, PostgreSQL, SAP
- Amazon RDS
- Amazon Redshift
- Amazon DynamoDB
- Amazon S3
- ElasticSearch Service
- Kinesis Data Streams
- DocumentDB

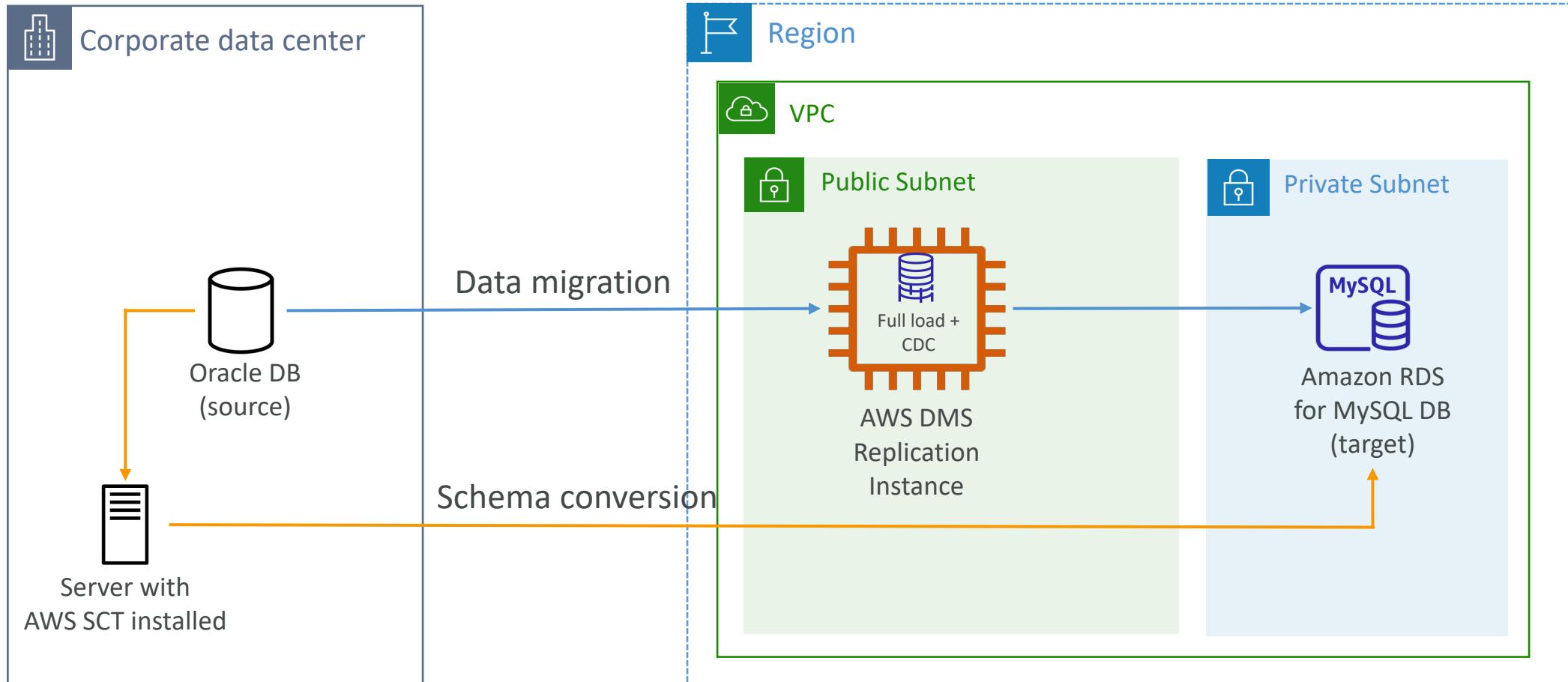
# AWS Schema Conversion Tool (SCT)

- Convert your Database's Schema from one engine to another
- Example OLTP: (SQL Server or Oracle) to MySQL, PostgreSQL, Aurora
- Example OLAP: (Teradata or Oracle) to Amazon Redshift
- Prefer compute-intensive instances to optimize data conversions



- You do not need to use SCT if you are migrating the same DB engine
  - Ex: On-Premise PostgreSQL => RDS PostgreSQL
  - The DB engine is still PostgreSQL (RDS is the platform)

# DMS - Continuous Replication



# On-Premise strategy with AWS

- Ability to download Amazon Linux 2 AMI as a VM (.iso format)
  - VMWare, KVM, VirtualBox (Oracle VM), Microsoft Hyper-V
- VM Import / Export
  - Migrate existing applications into EC2
  - Create a DR repository strategy for your on-premises VMs
  - Can export back the VMs from EC2 to on-premises
- AWS Application Discovery Service
  - Gather information about your on-premises servers to plan a migration
  - Server utilization and dependency mappings
  - Track with AWS Migration Hub
- AWS Database Migration Service (DMS)
  - replicate On-premise => AWS , AWS => AWS, AWS => On-premise
  - Works with various database technologies (Oracle, MySQL, DynamoDB, etc..)
- AWS Server Migration Service (SMS)
  - Incremental replication of on-premises live servers to AWS

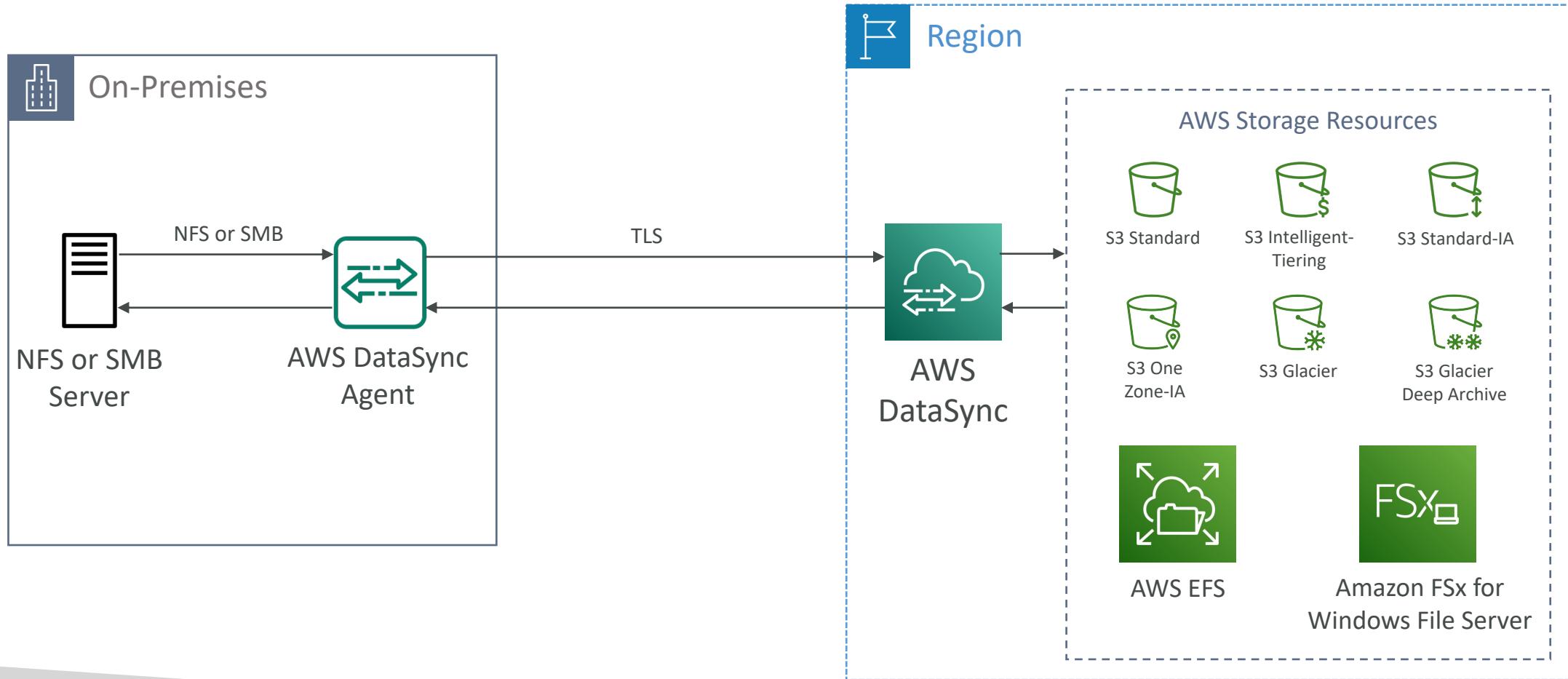
# AWS DataSync



- Move large amount of data from on-premises to AWS
- Can synchronize to: Amazon S3 (any storage classes – including Glacier), Amazon EFS, Amazon FSx for Windows
- Move data from your NAS or file system via NFS or SMB
- Replication tasks can be scheduled hourly, daily, weekly
- Leverage the DataSync agent to connect to your systems
- Can setup a bandwidth limit

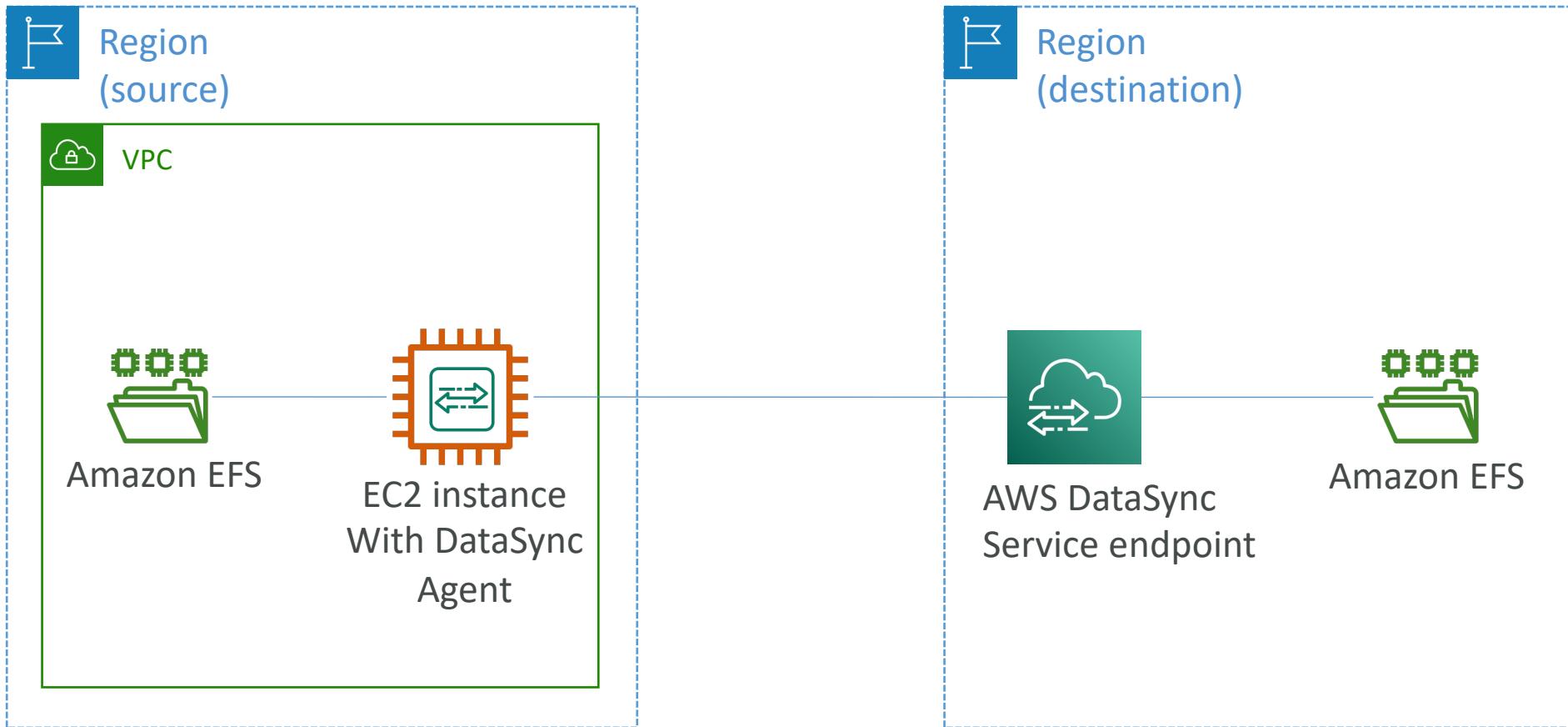
# AWS DataSync

NFS / SMB to AWS (S3, EFS, FSx for Windows)



# AWS DataSync

## EFS to EFS





# AWS Backup

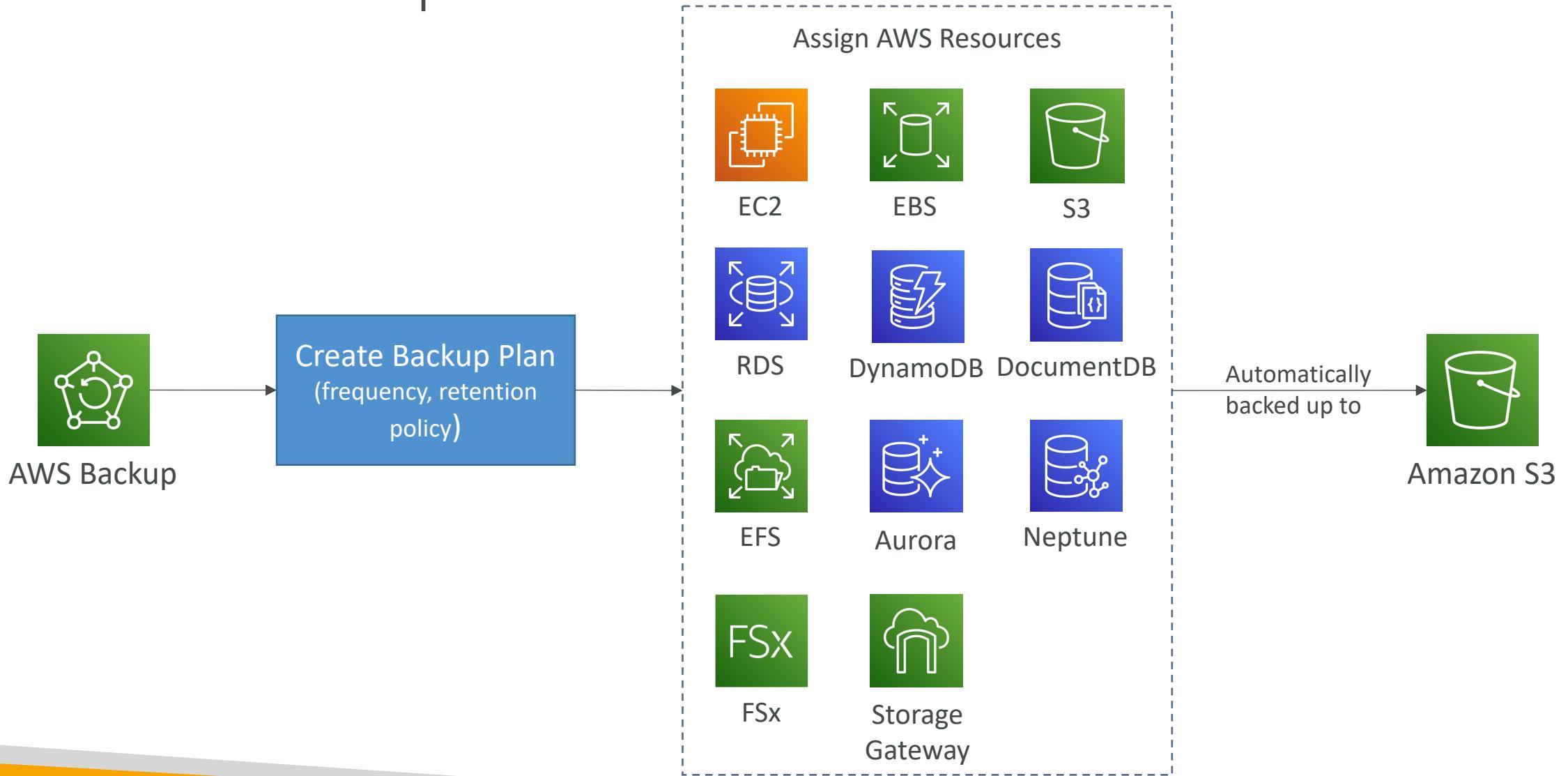
- Fully managed service
- Centrally manage and automate backups across AWS services
- No need to create custom scripts and manual processes
- Supported services:
  - Amazon EC2 / Amazon EBS
  - Amazon S3
  - Amazon RDS (all DBs engines) / Amazon Aurora / Amazon DynamoDB
  - Amazon DocumentDB / Amazon Neptune
  - Amazon EFS / Amazon FSx (Lustre & Windows File Server)
  - AWS Storage Gateway (Volume Gateway)
- Supports cross-region backups
- Supports cross-account backups

# AWS Backup



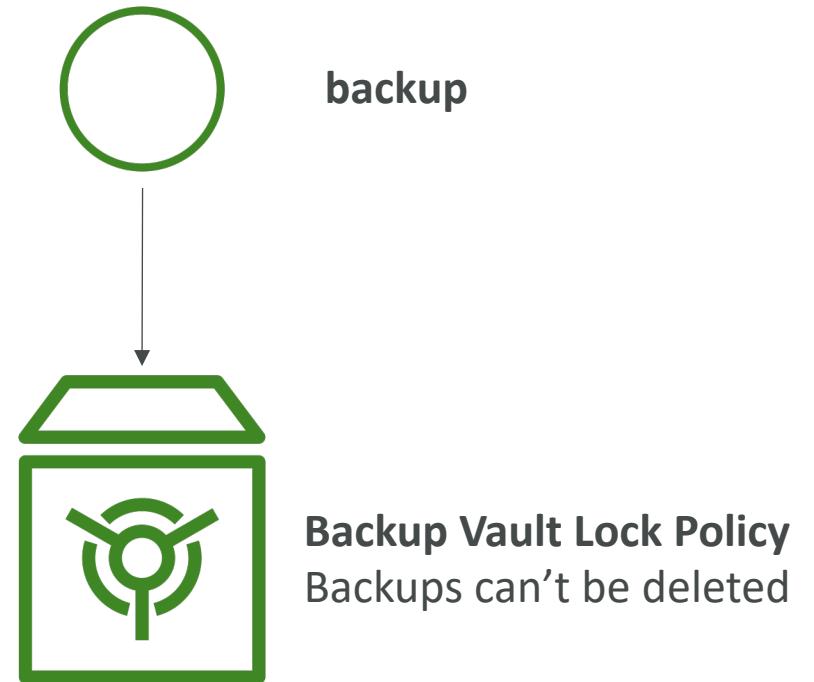
- Supports PITR for supported services
- On-Demand and Scheduled backups
- Tag-based backup policies
- You create backup policies known as **Backup Plans**
  - Backup frequency (every 12 hours, daily, weekly, monthly, cron expression)
  - Backup window
  - Transition to Cold Storage (Never, Days, Weeks, Months, Years)
  - Retention Period (Always, Days, Weeks, Months, Years)

# AWS Backup



# AWS Backup Vault Lock

- Enforce a WORM (Write Once Read Many) state for all the backups that you store in your AWS Backup Vault
- Additional layer of defense to protect your backups against:
  - Inadvertent or malicious delete operations
  - Updates that shorten or alter retention periods
- Even the root user cannot delete backups when enabled

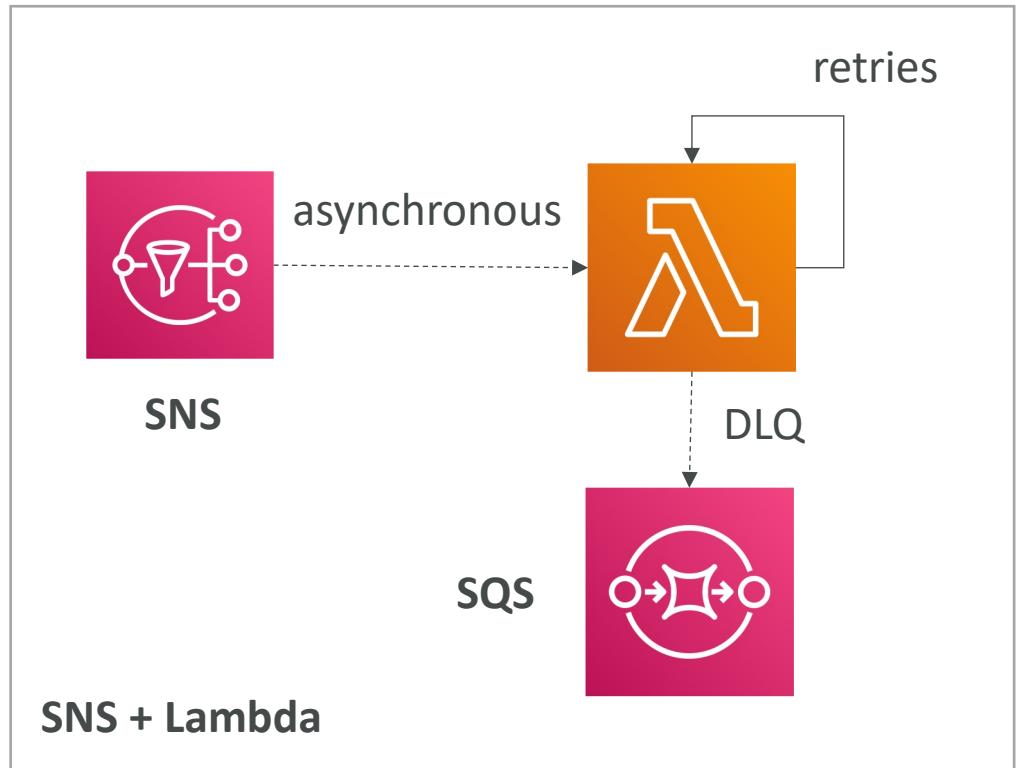
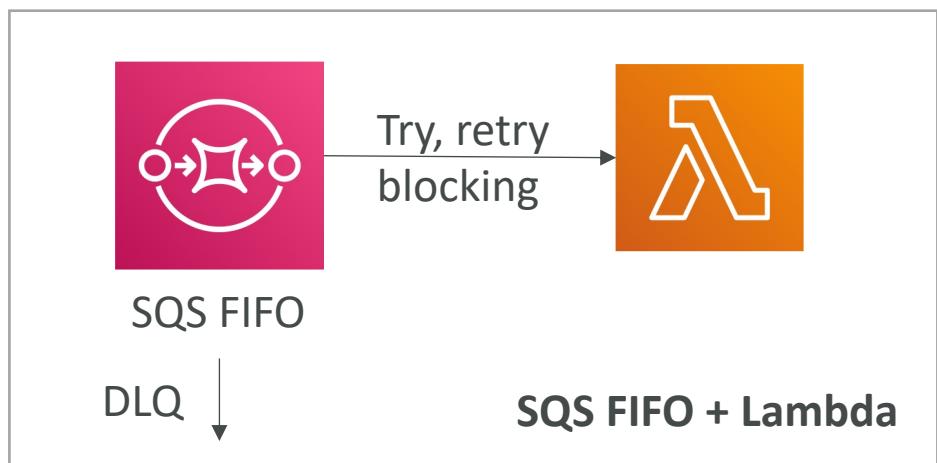
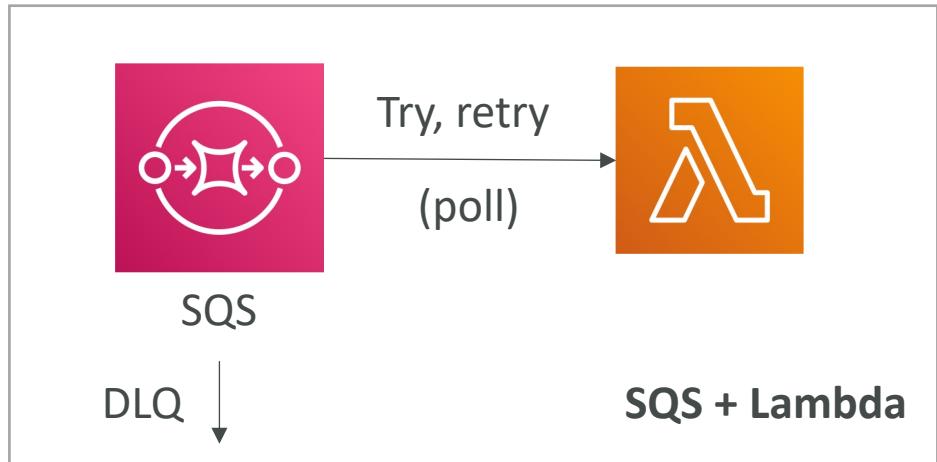


# Transferring large amount of data into AWS

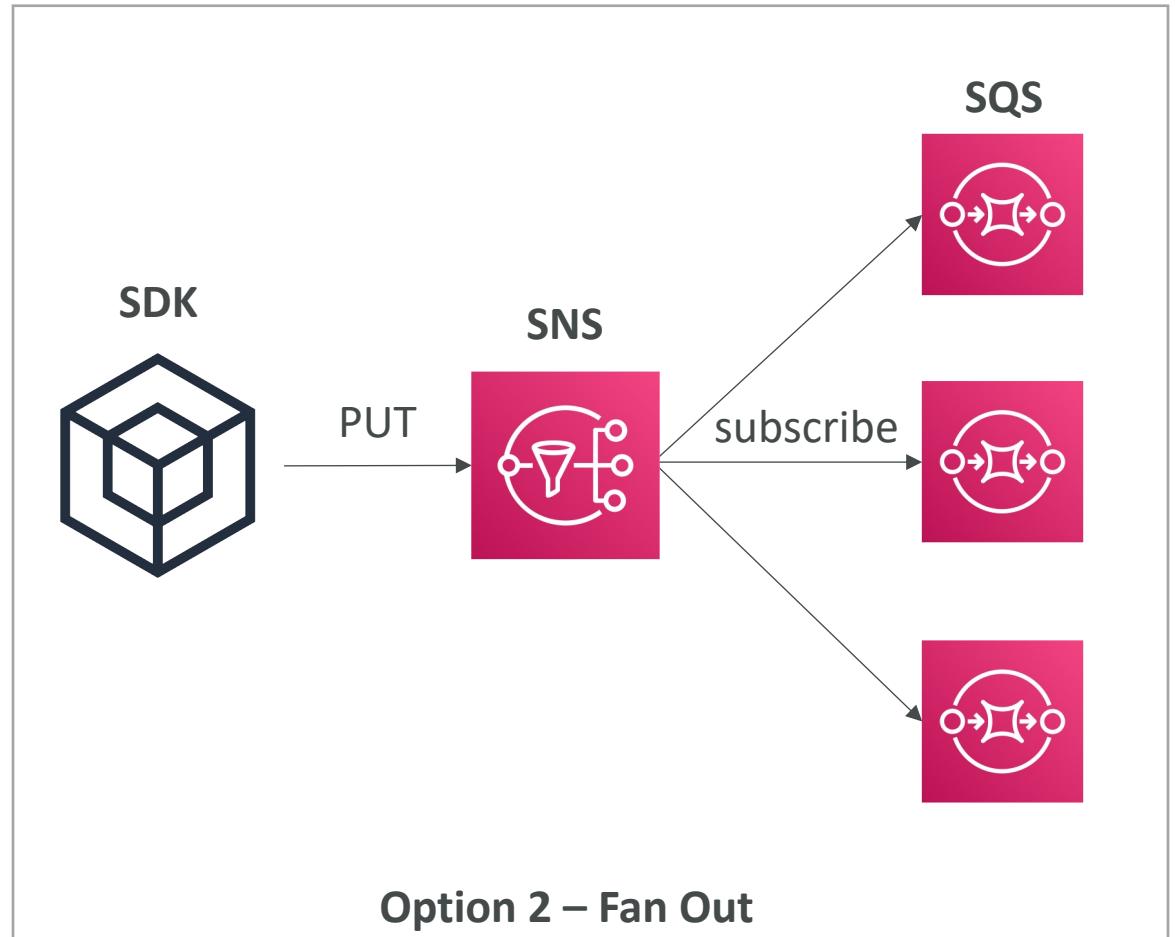
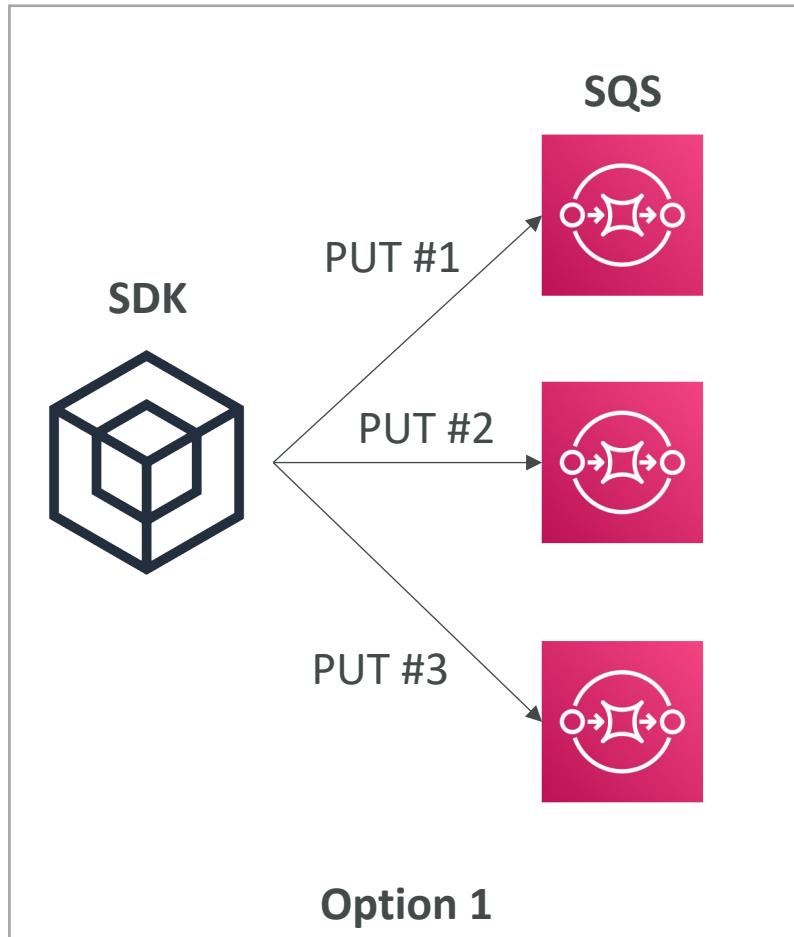
- Example: transfer 200 TB of data in the cloud. We have a 100 Mbps internet connection.
- Over the internet / Site-to-Site VPN:
  - Immediate to setup
  - Will take  $200(\text{TB}) * 1000(\text{GB}) * 1000(\text{MB}) * 8(\text{Mb}) / 100 \text{ Mbps} = 16,000,000 \text{s} = 185 \text{d}$
- Over direct connect 1 Gbps:
  - Long for the one-time setup (over a month)
  - Will take  $200(\text{TB}) * 1000(\text{GB}) * 8(\text{Gb}) / 1 \text{ Gbps} = 1,600,000 \text{s} = 18.5 \text{d}$
- Over Snowball:
  - Will take 2 to 3 snowballs in parallel
  - Takes about 1 week for the end-to-end transfer
  - Can be combined with DMS
- For on-going replication / transfers: Site-to-Site VPN or DX with DMS or DataSync

# Extra Solution Architecture discussions

# Lambda, SNS & SQS

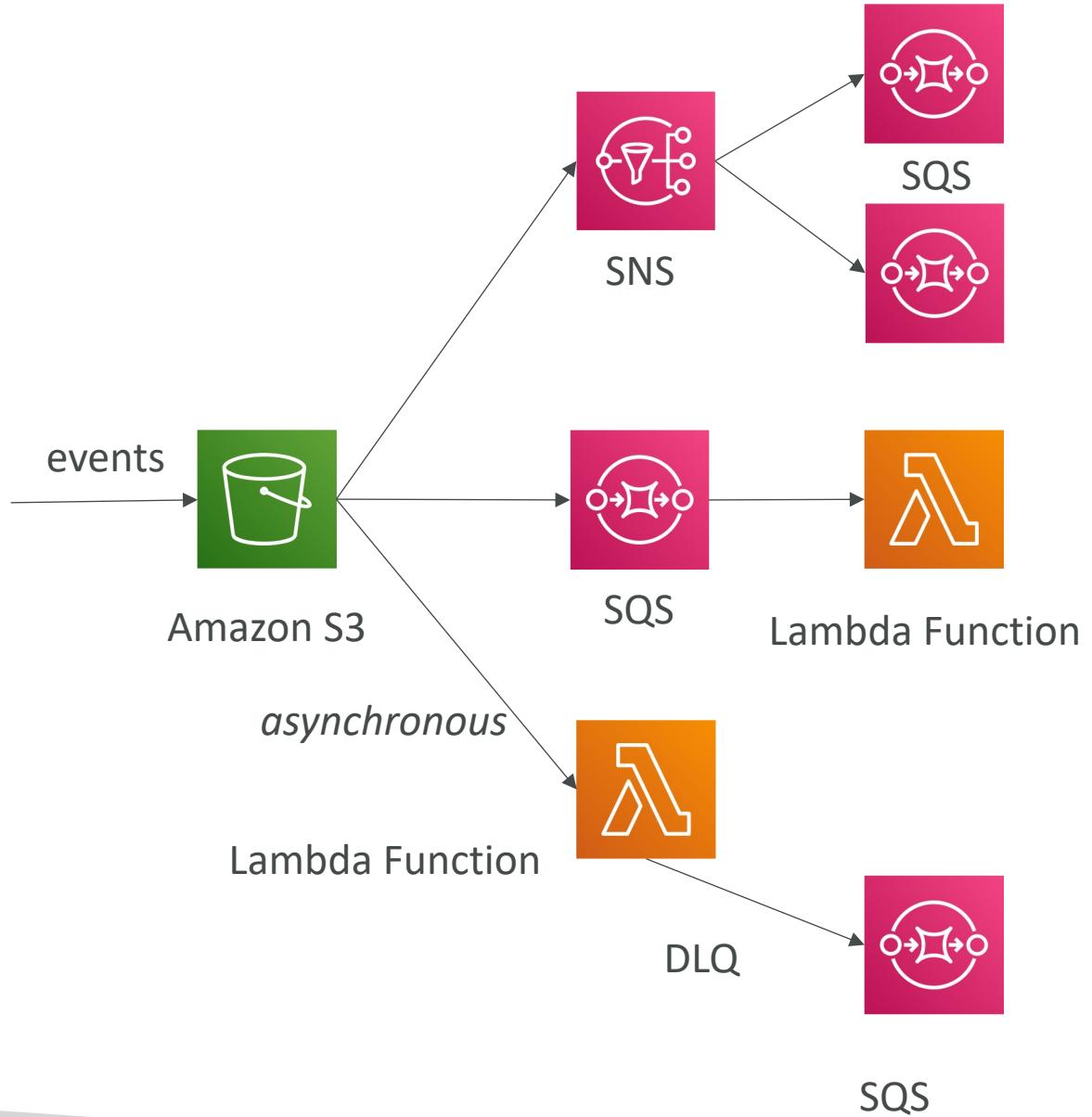


# Fan Out Pattern: deliver to multiple SQS

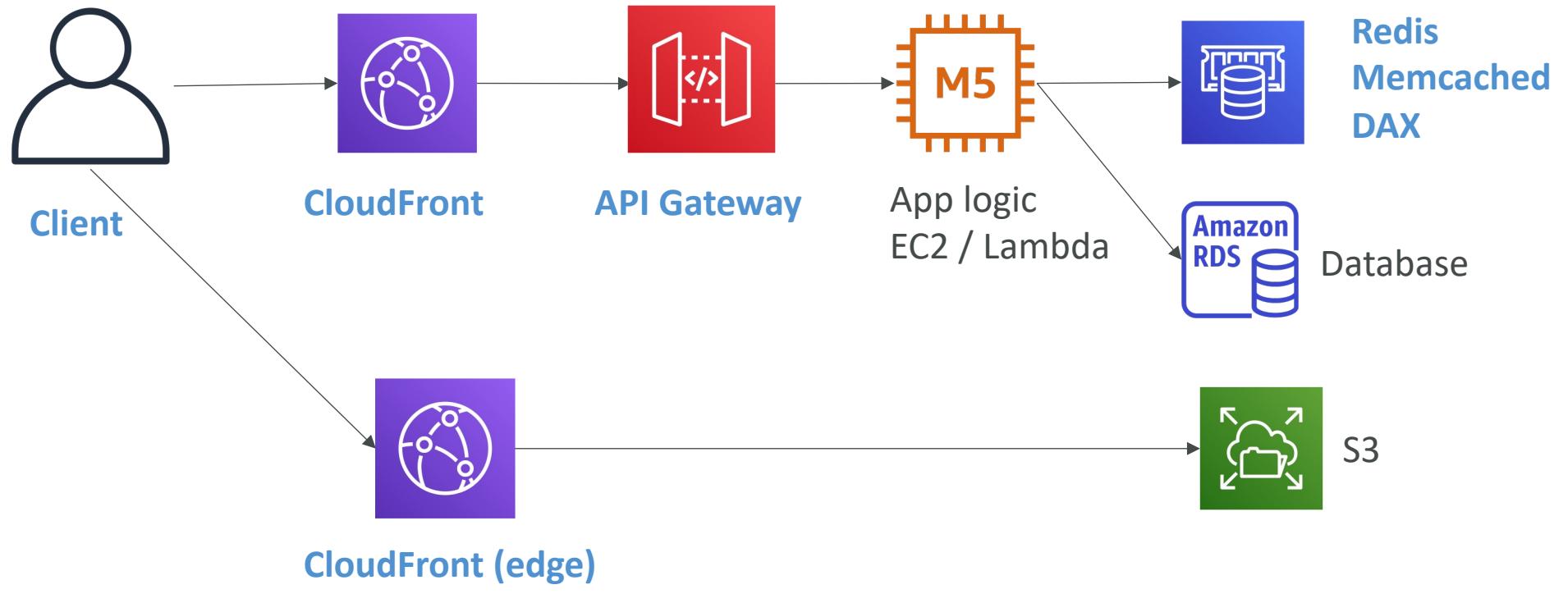


# S3 Events

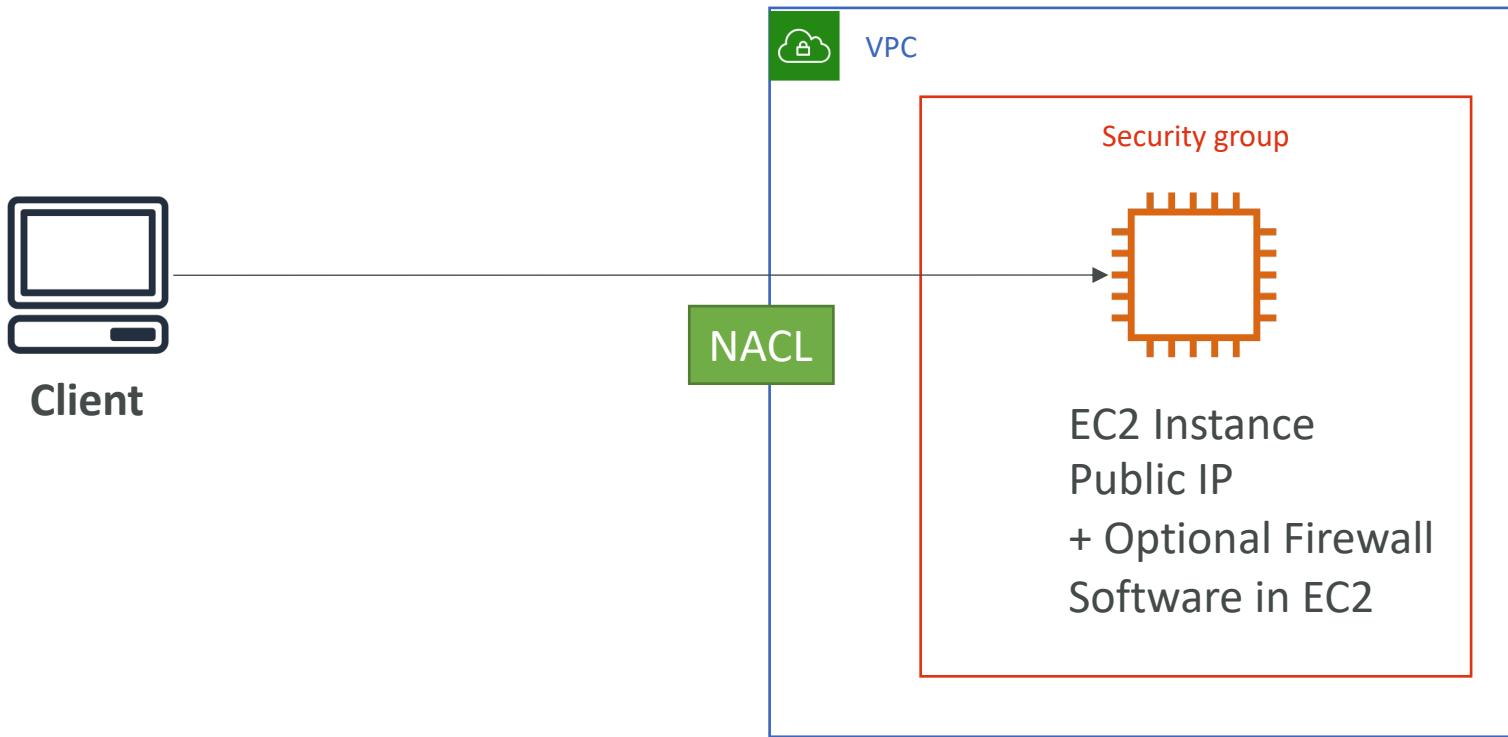
- S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication...
- Object name filtering possible (\*.jpg)
- Use case: generate thumbnails of images uploaded to S3
- Can create as many “S3 events” as desired
- S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer
- If two writes are made to a single non-versioned object at the same time, it is possible that only a single event notification will be sent
- If you want to ensure that an event notification is sent for every successful write, you can enable versioning on your bucket.



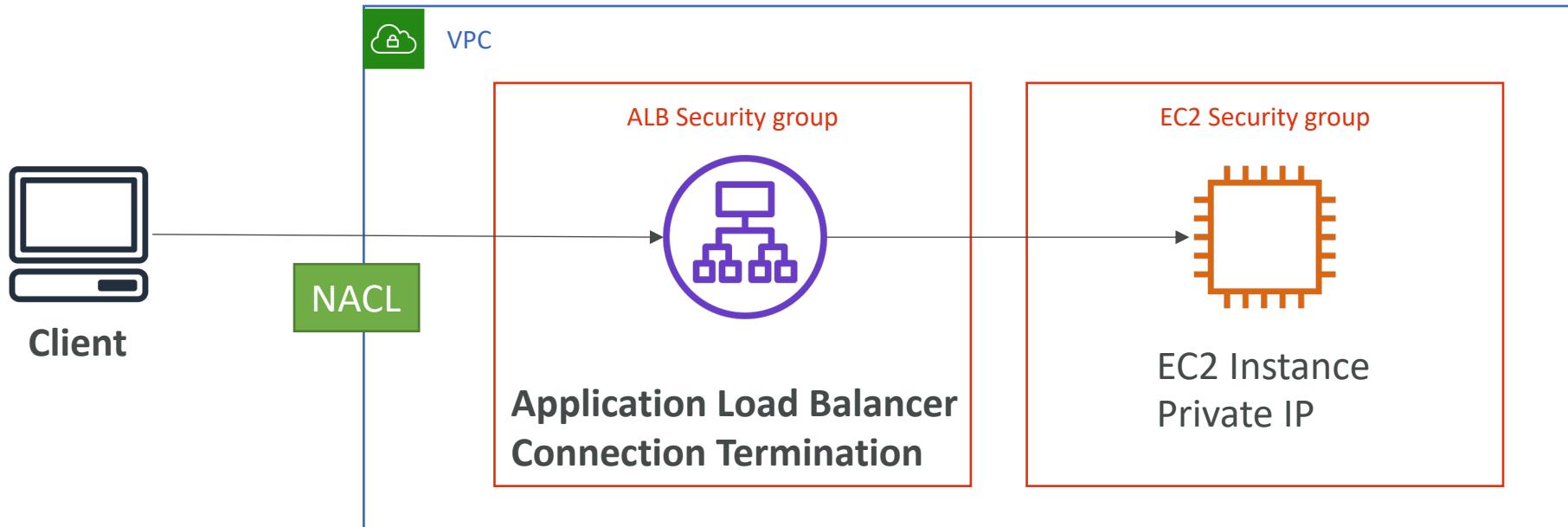
# Caching Strategies



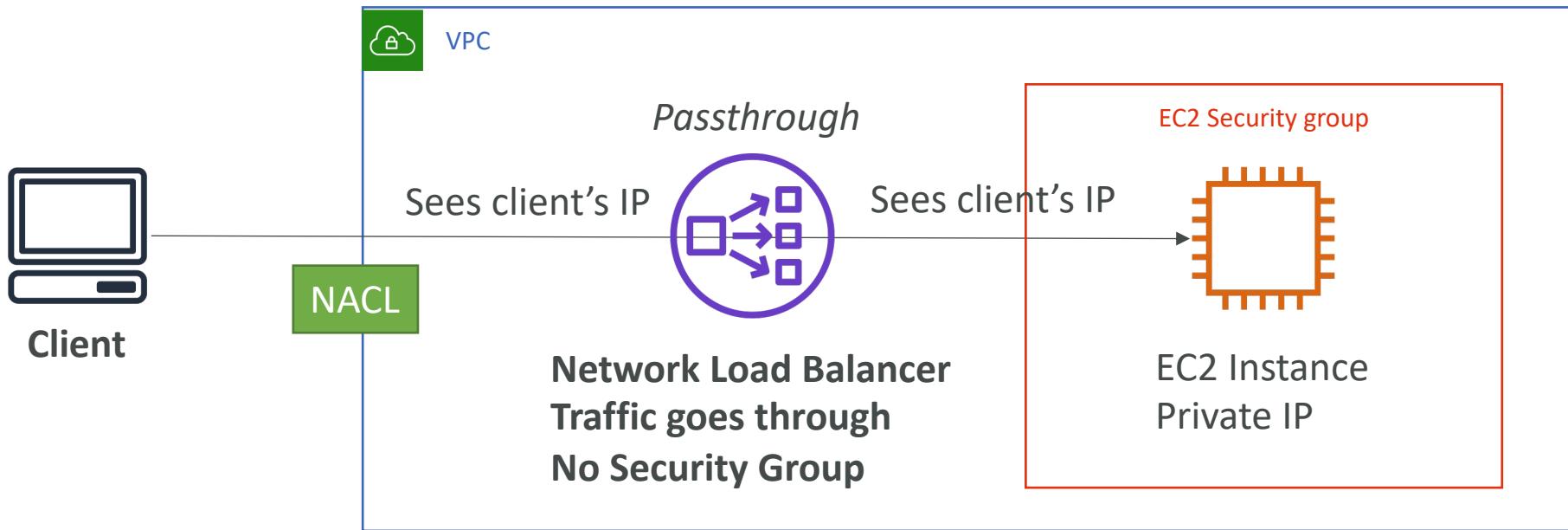
# Blocking an IP address



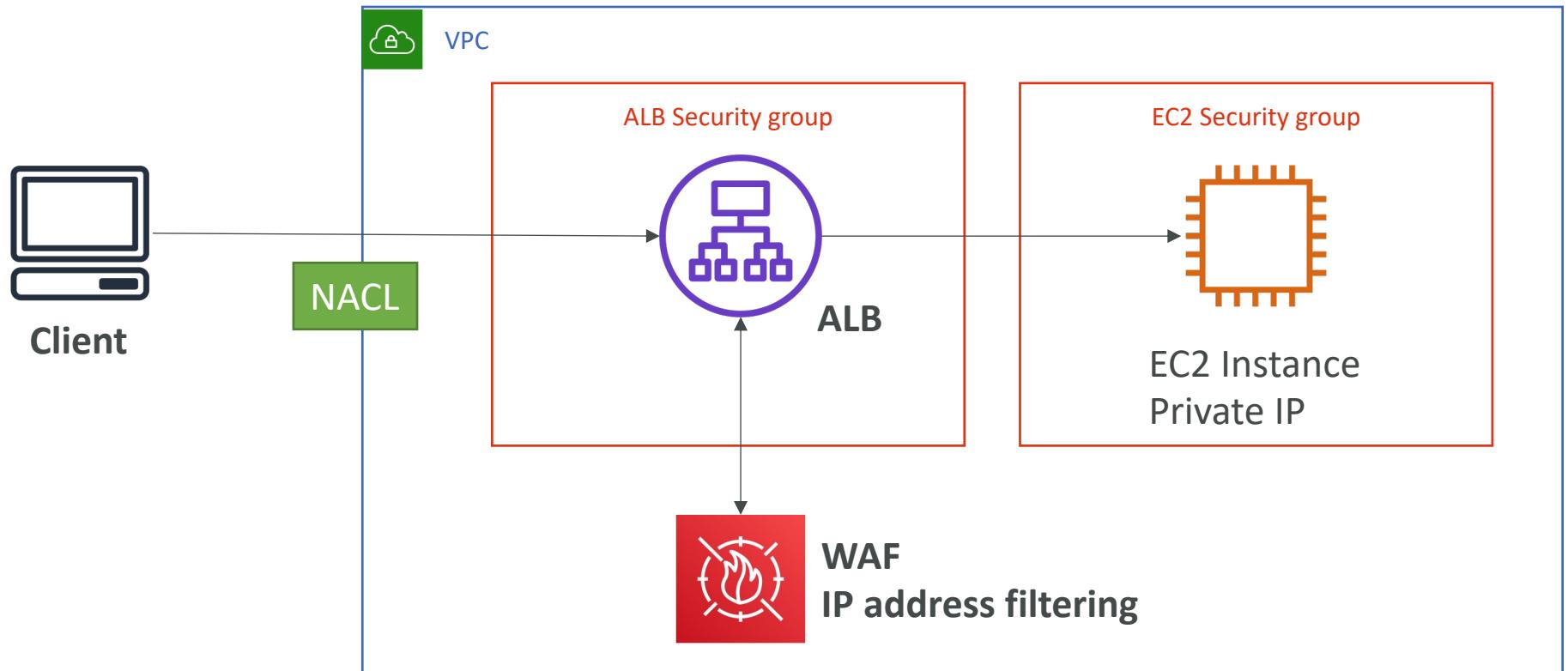
# Blocking an IP address – with an ALB



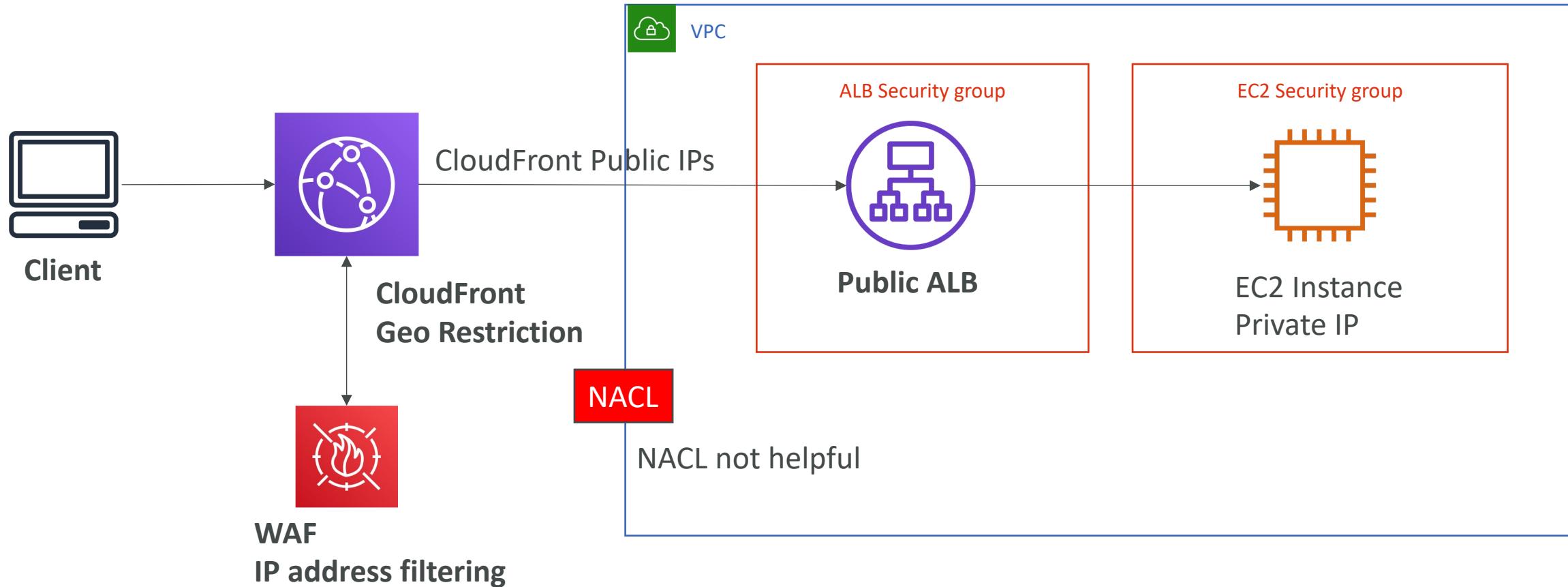
# Blocking an IP address – with an NLB



# Blocking an IP address – ALB + WAF



# Blocking an IP address – ALB, CloudFront WAF



# High Performance Computing (HPC)

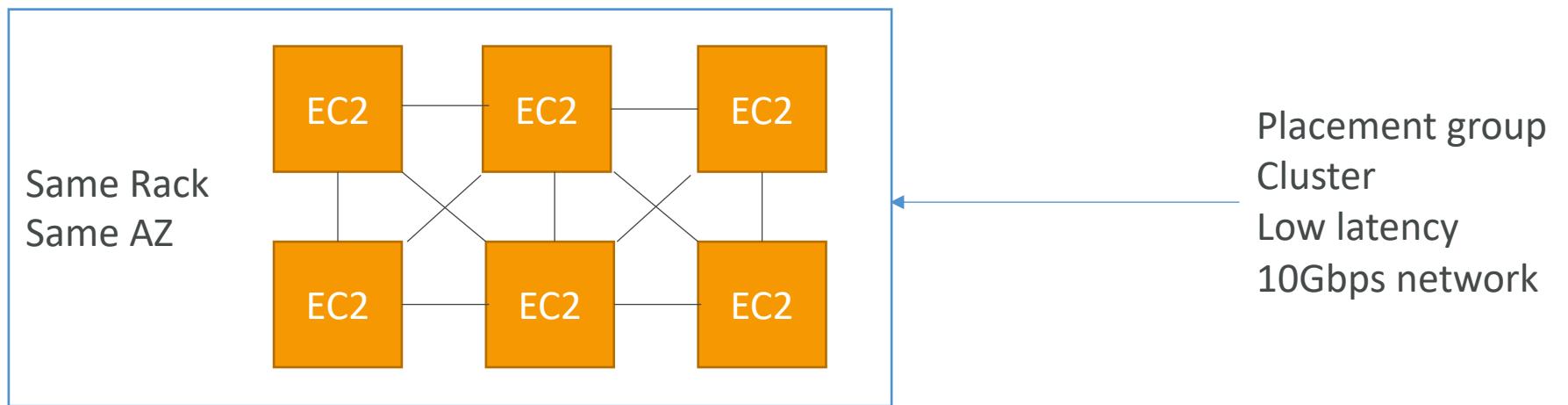
- The cloud is the perfect place to perform HPC
- You can create a very high number of resources in no time
- You can speed up time to results by adding more resources
- You can pay only for the systems you have used
- Perform genomics, computational chemistry, financial risk modeling, weather prediction, machine learning, deep learning, autonomous driving
- Which services help perform HPC?

# Data Management & Transfer

- AWS Direct Connect:
  - Move GB/s of data to the cloud, over a private secure network
- Snowball & Snowmobile
  - Move PB of data to the cloud
- AWS DataSync
  - Move large amount of data between on-premises and S3, EFS, FSx for Windows

# Compute and Networking

- EC2 Instances:
  - CPU optimized, GPU optimized
  - Spot Instances / Spot Fleets for cost savings + Auto Scaling
- EC2 Placement Groups: **Cluster** for good network performance



# Compute and Networking

- EC2 Enhanced Networking (SR-IOV)
  - Higher bandwidth, higher PPS (packet per second), lower latency
  - Option 1: Elastic Network Adapter (ENA) up to 100 Gbps
  - Option 2: Intel 82599 VF up to 10 Gbps – LEGACY
- Elastic Fabric Adapter (EFA)
  - Improved ENA for HPC, only works for Linux
  - Great for inter-node communications, **tightly coupled workloads**
  - Leverages Message Passing Interface (MPI) standard
  - Bypasses the underlying Linux OS to provide low-latency, reliable transport

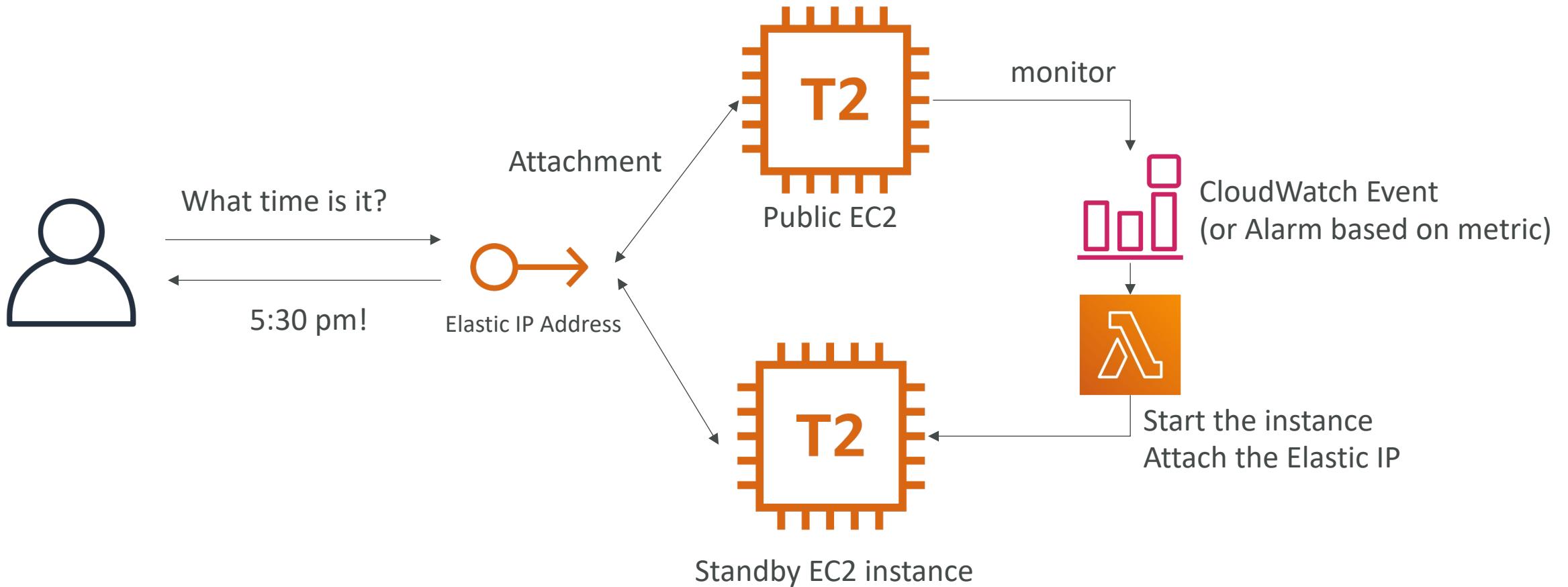
# Storage

- Instance-attached storage:
  - EBS: scale up to 256,000 IOPS with io2 Block Express
  - Instance Store: scale to millions of IOPS, linked to EC2 instance, low latency
- Network storage:
  - Amazon S3: large blob, not a file system
  - Amazon EFS: scale IOPS based on total size, or use provisioned IOPS
  - Amazon FSx for Lustre:
    - HPC optimized distributed file system, millions of IOPS
    - Backed by S3

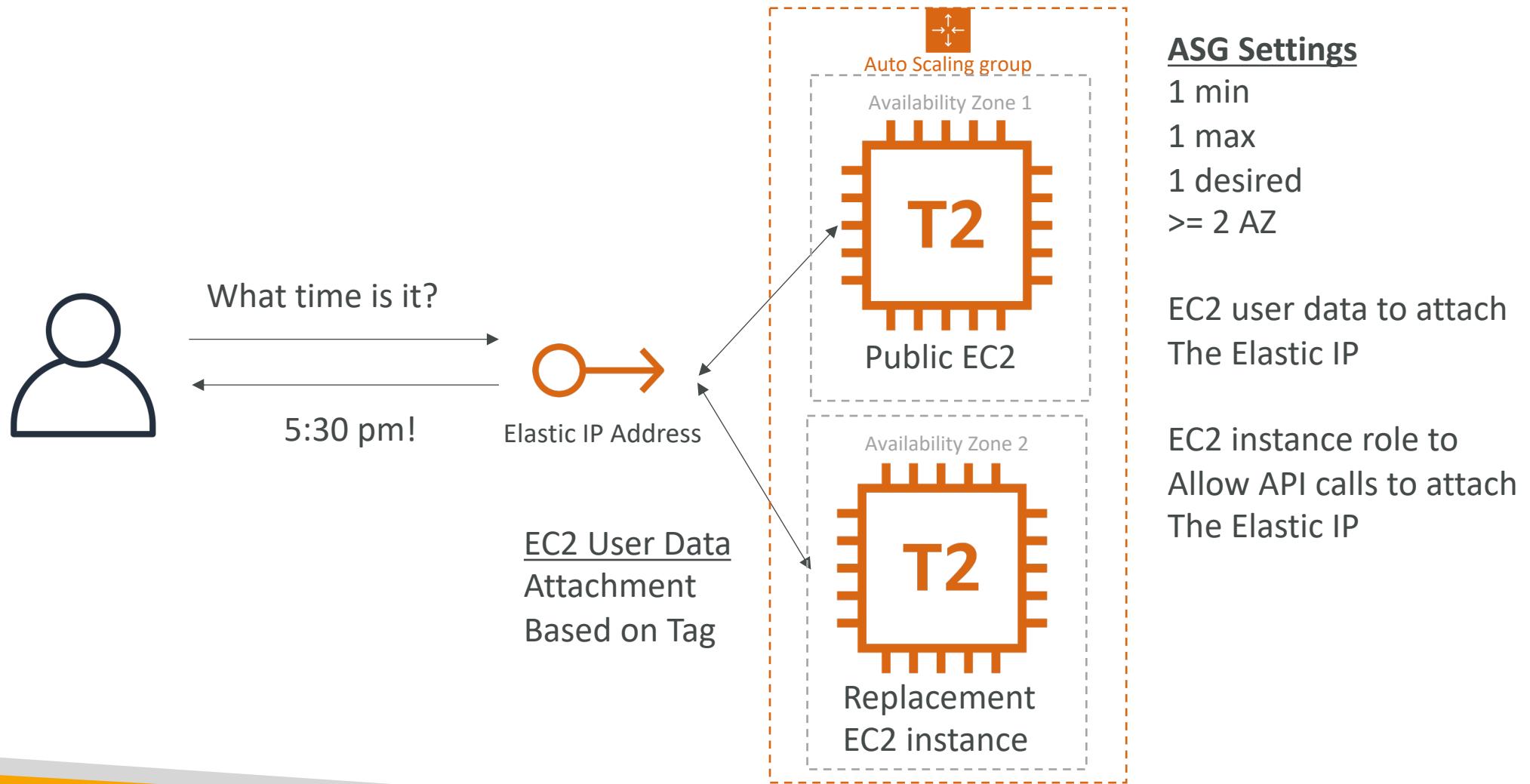
# Automation and Orchestration

- AWS Batch
  - AWS Batch supports multi-node parallel jobs, which enables you to run single jobs that span multiple EC2 instances.
  - Easily schedule jobs and launch EC2 instances accordingly
- AWS ParallelCluster
  - Open-source cluster management tool to deploy HPC on AWS
  - Configure with text files
  - Automate creation of VPC, Subnet, cluster type and instance types
  - Ability to enable EFA on the cluster (improves network performance)

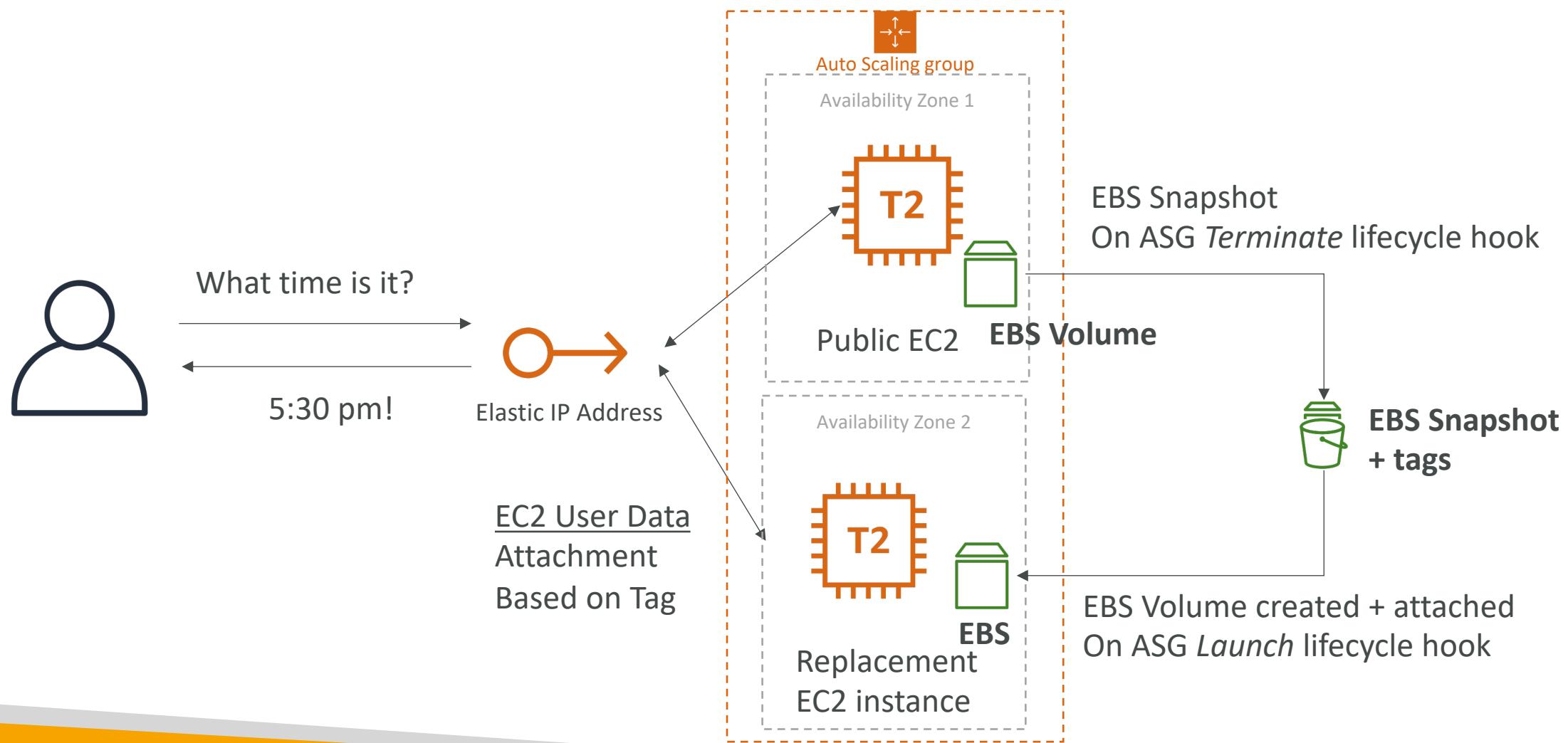
# Creating a highly available EC2 instance



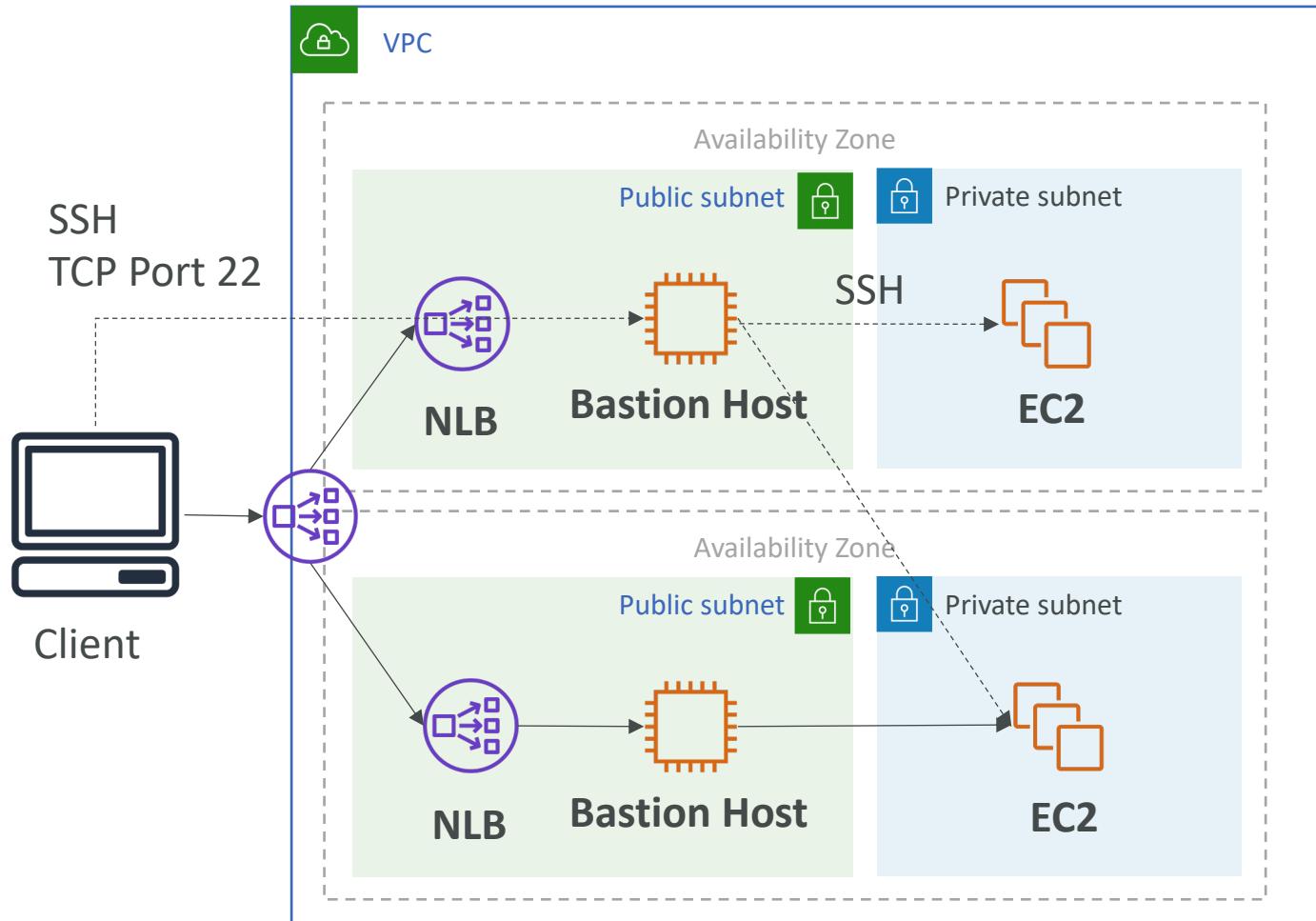
# Creating a highly available EC2 instance With an Auto Scaling Group



# Creating a highly available EC2 instance With ASG + EBS



# High Availability for a Bastion Host



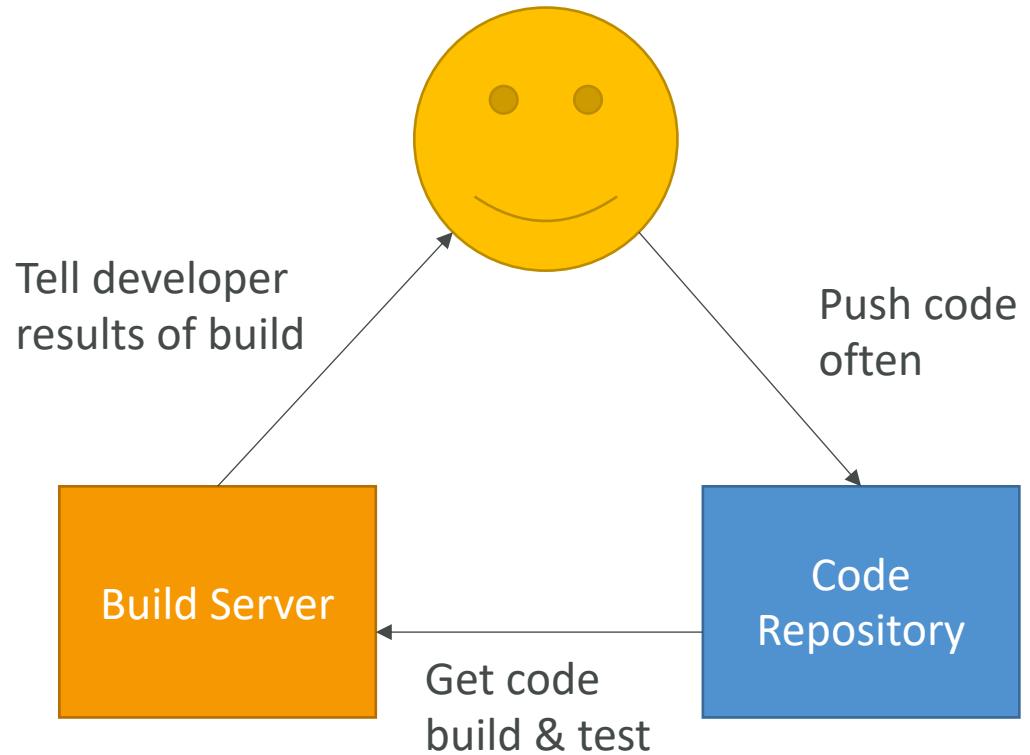
- HA options for the bastion host
  - Run 2 across 2 AZ
  - Run 1 across 2 AZ with 1 ASG 1:1:1
- Routing to the bastion host
  - If 1 bastion host, use an elastic IP with ec2 user-data script to access it
  - If 2 bastion hosts, use an Network Load Balancer (layer 4) deployed in multiple AZ
  - If NLB, the bastion hosts can live in the private subnet directly
- Note: Can't use ALB as the ALB is layer 7 (HTTP protocol)

# Other Services

Overview of Services that might come up in a few questions

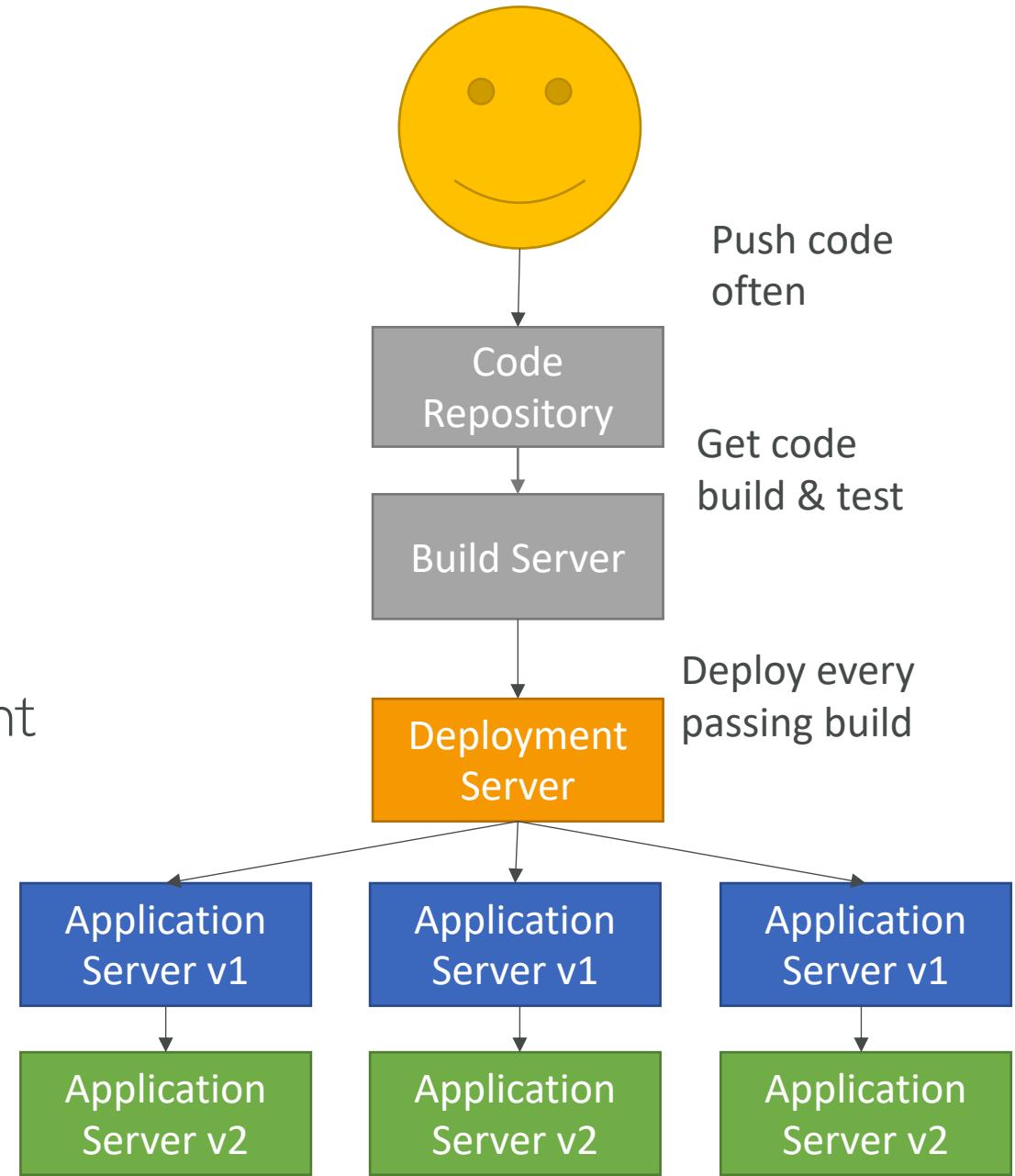
# Continuous Integration

- Developers push the code to a code repository often (GitHub / CodeCommit / Bitbucket / etc...)
- A testing / build server checks the code as soon as it's pushed (CodeBuild / Jenkins CI / etc...)
- The developer gets feedback about the tests and checks that have passed / failed
- Find bugs early, fix bugs
- Deliver faster as the code is tested
- Deploy often
- Happier developers, as they're unblocked

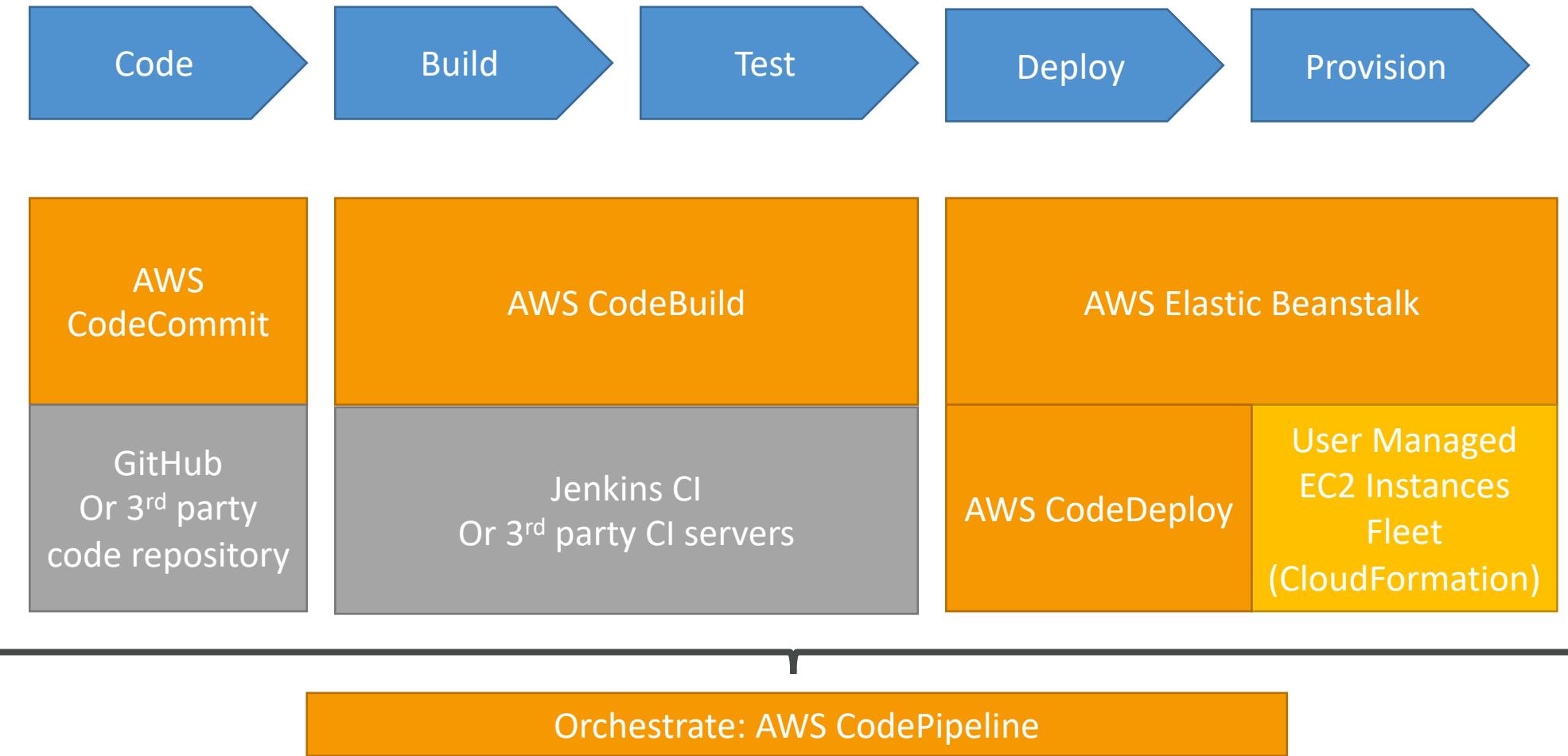


# Continuous Delivery

- Ensure that the software can be released reliably whenever needed.
- Ensures deployments happen often and are quick
- Shift away from “one release every 3 months” to “5 releases a day”
- That usually means automated deployment
  - CodeDeploy
  - Jenkins CD
  - Spinnaker
  - Etc...



# Technology Stack for CICD



# Infrastructure as Code

- Currently, we have been doing a lot of manual work
- All this manual work will be very tough to reproduce:
  - In another region
  - in another AWS account
  - Within the same region if everything was deleted
- Wouldn't it be great, if all our infrastructure was... code?
- That code would be deployed and create / update / delete our infrastructure

# What is CloudFormation



- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
  - I want a security group
  - I want two EC2 machines using this security group
  - I want two Elastic IPs for these EC2 machines
  - I want an S3 bucket
  - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

# Benefits of AWS CloudFormation (1/2)

- Infrastructure as code
  - No resources are manually created, which is excellent for control
  - The code can be version controlled for example using git
  - Changes to the infrastructure are reviewed through code
- Cost
  - Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
  - You can estimate the costs of your resources using the CloudFormation template
  - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

# Benefits of AWS CloudFormation (2/2)

- Productivity
  - Ability to destroy and re-create an infrastructure on the cloud on the fly
  - Automated generation of Diagram for your templates!
  - Declarative programming (no need to figure out ordering and orchestration)
- Separation of concern: create many stacks for many apps, and many layers. Ex:
  - VPC stacks
  - Network stacks
  - App stacks
- Don't re-invent the wheel
  - Leverage existing templates on the web!
  - Leverage the documentation

# How CloudFormation Works

- Templates have to be uploaded in S3 and then referenced in CloudFormation
- To update a template, we can't edit previous ones. We have to re-upload a new version of the template to AWS
- Stacks are identified by a name
- Deleting a stack deletes every single artifact that was created by CloudFormation.

# Deploying CloudFormation templates

- Manual way:
  - Editing templates in the CloudFormation Designer
  - Using the console to input parameters, etc
- Automated way:
  - Editing templates in a YAML file
  - Using the AWS CLI (Command Line Interface) to deploy the templates
  - Recommended way when you fully want to automate your flow

# CloudFormation Building Blocks

## Templates components

1. Resources: your AWS resources declared in the template (**MANDATORY**)
2. Parameters: the dynamic inputs for your template
3. Mappings: the static variables for your template
4. Outputs: References to what has been created
5. Conditionals: List of conditions to perform resource creation
6. Metadata

## Templates helpers:

1. References
2. Functions

Note:

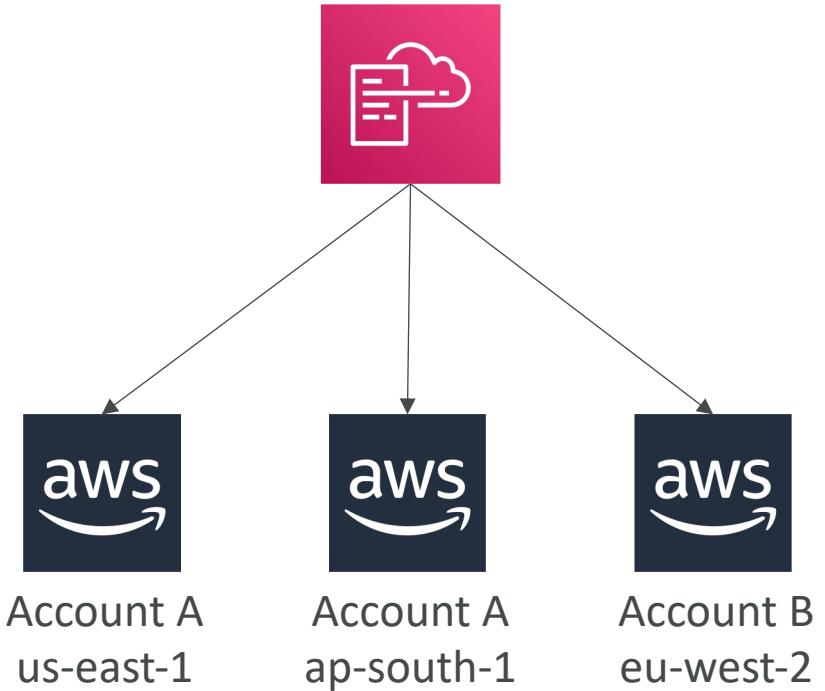
# This is an introduction to CloudFormation

- It can take over 3 hours to properly learn and master CloudFormation
- This lecture is meant so you get a good idea of how it works
- The exam expects you to understand how to read CloudFormation

# CloudFormation - StackSets

- Create, update, or delete stacks across **multiple accounts and regions** with a single operation
- Administrator account to create StackSets
- Trusted accounts to create, update, delete stack instances from StackSets
- When you update a stack set, *all* associated stack instances are updated throughout all accounts and regions.

CloudFormation **StackSet**  
Admin Account

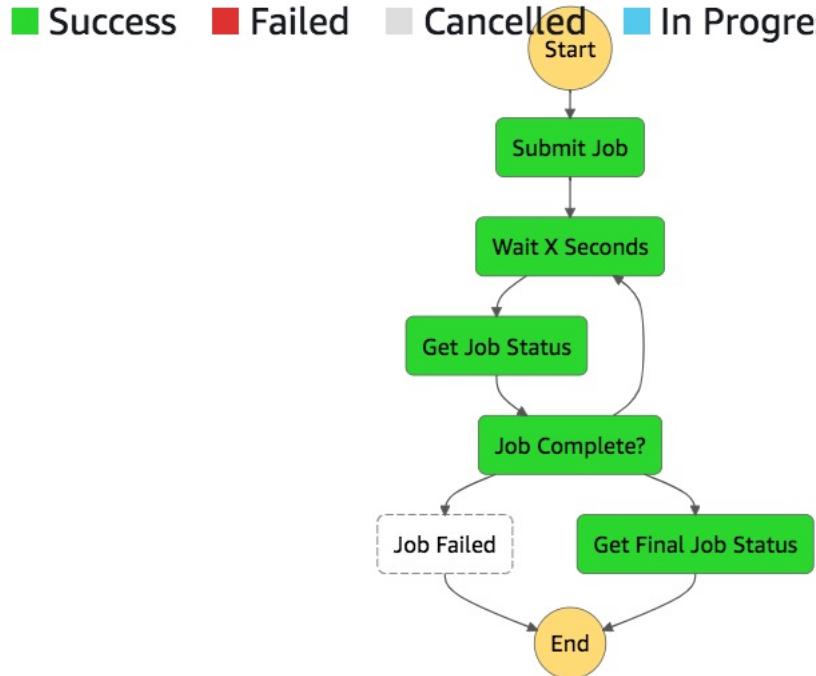
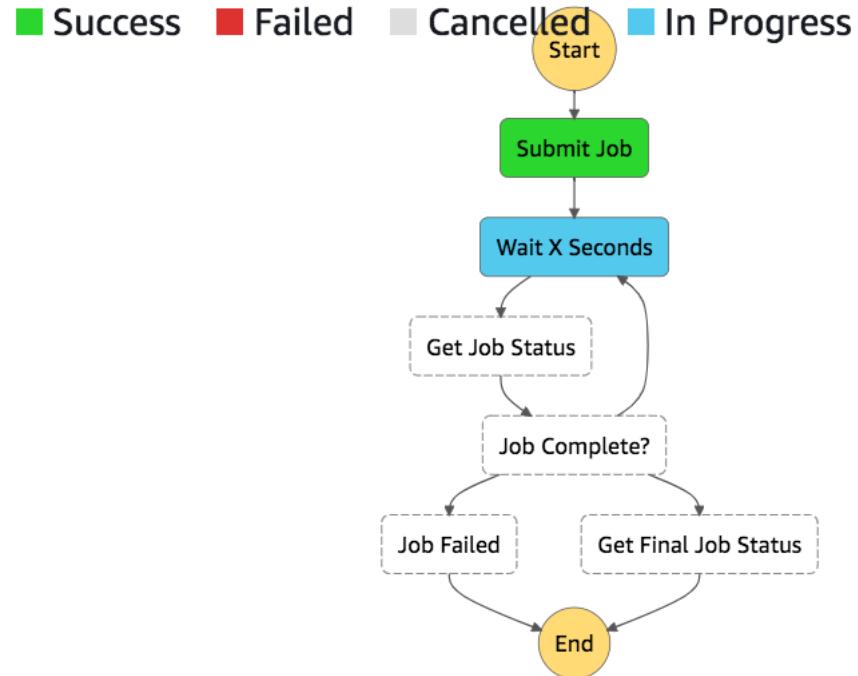
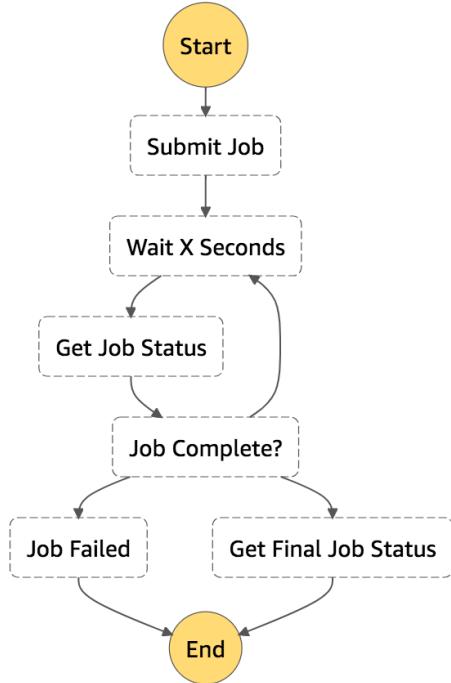




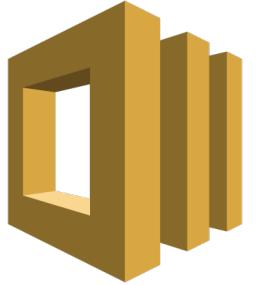
# AWS Step Functions

- Build serverless visual workflow to orchestrate your Lambda functions
- Represent flow as a **JSON state machine**
- Features: sequence, parallel, conditions, timeouts, error handling...
- Can also integrate with EC2, ECS, On premise servers, API Gateway
- Maximum execution time of 1 year
- Possibility to implement human approval feature
- Use cases:
  - Order fulfillment
  - Data processing
  - Web applications
  - Any workflow

# Visual workflow in Step Functions



# AWS SWF – Simple Workflow Service



- Coordinate work amongst applications
- Code runs on EC2 (not serverless)
- 1 year max runtime
- Concept of “activity step” and “decision step”
- Has built-in “human intervention” step
- Example: order fulfilment from web to warehouse to delivery
- **Step Functions is recommended to be used for new applications, except:**
  - If you need external signals to intervene in the processes
  - If you need child processes that return values to parent processes

# Amazon EMR



- EMR stands for “Elastic MapReduce”
- EMR helps creating **Hadoop clusters (Big Data)** to analyze and process vast amount of data
- The clusters can be made of hundreds of EC2 instances
- Also supports Apache Spark, HBase, Presto, Flink...
- EMR takes care of all the provisioning and configuration
- Auto-scaling and integrated with Spot instances
- Use cases: data processing, machine learning, web indexing, big data...

# AWS Opsworks



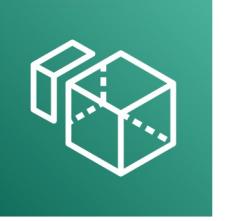
- Chef & Puppet help you perform server configuration automatically, or repetitive actions
- They work great with EC2 & On Premise VM
- AWS Opsworks = Managed Chef & Puppet
- It's an alternative to AWS SSM
- No hands on here, no knowledge of chef and puppet needed
- In the exam: Chef & Puppet needed => AWS Opsworks



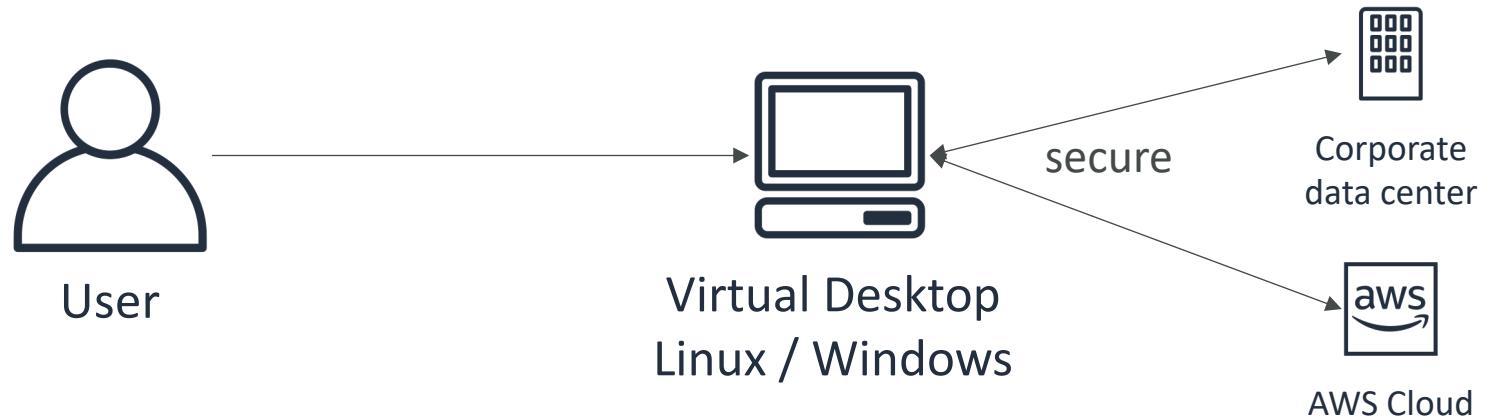
# Quick word on Chef / Puppet

- They help with managing configuration as code
- Helps in having consistent deployments
- Works with Linux / Windows
- Can automate: user accounts, cron, ntp, packages, services...
- They leverage “Recipes” or “Manifests”
- Chef / Puppet have similarities with SSM / Beanstalk / CloudFormation but they’re open-source tools that work cross-cloud

# AWS WorkSpaces



- Managed, Secure Cloud Desktop
- Great to eliminate management of on-premises VDI (Virtual Desktop Infrastructure)
- On Demand, pay per usage
- Secure, Encrypted, Network Isolation
- Integrated with Microsoft Active Directory



# AWS AppSync



- Store and sync data across mobile and web apps in real-time
- Makes use of GraphQL (mobile technology from Facebook)
- Client Code can be generated automatically
- Integrations with DynamoDB / Lambda
- Real-time subscriptions
- Offline data synchronization (replaces Cognito Sync)
- Fine Grained Security

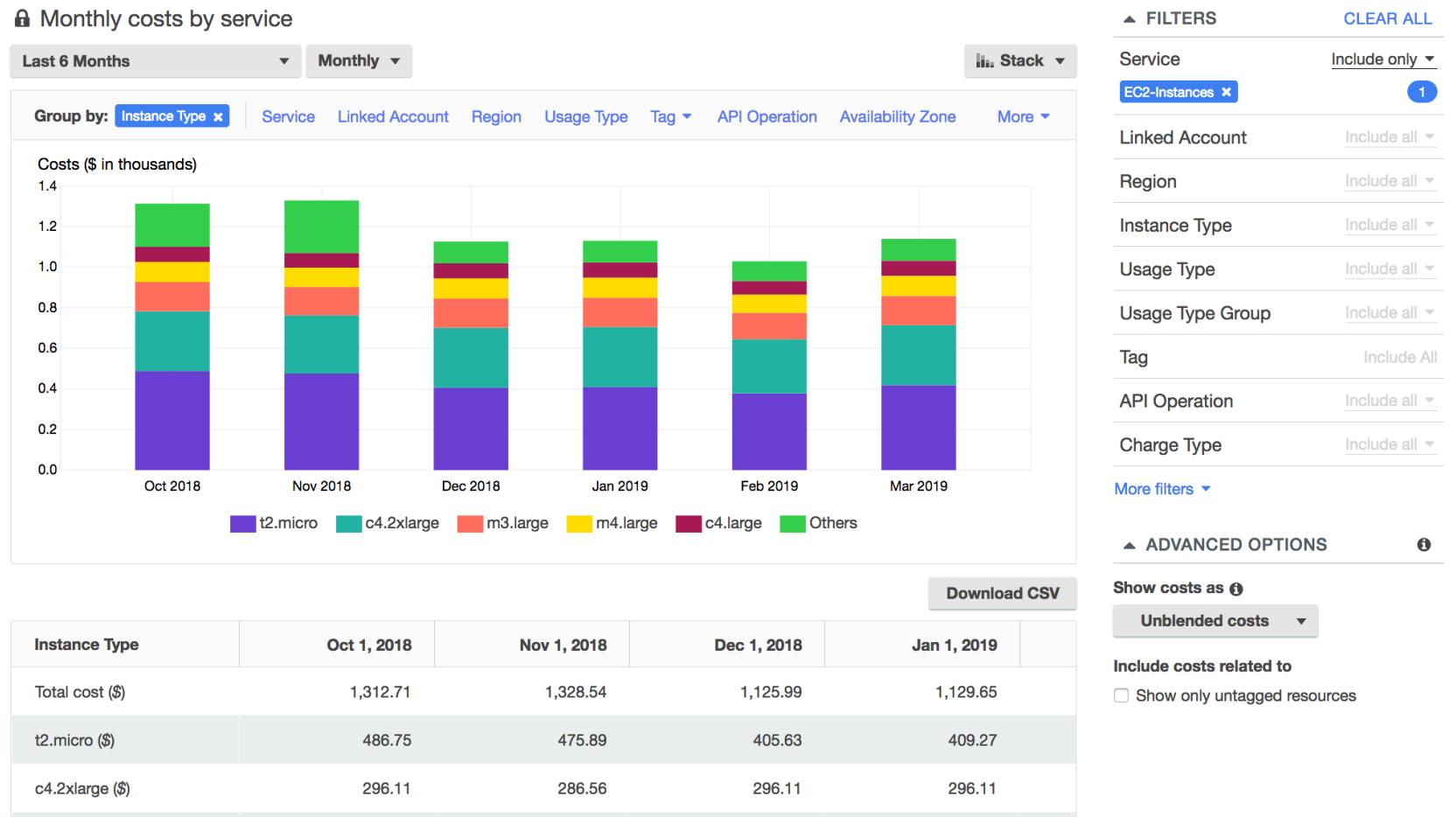


# Cost Explorer

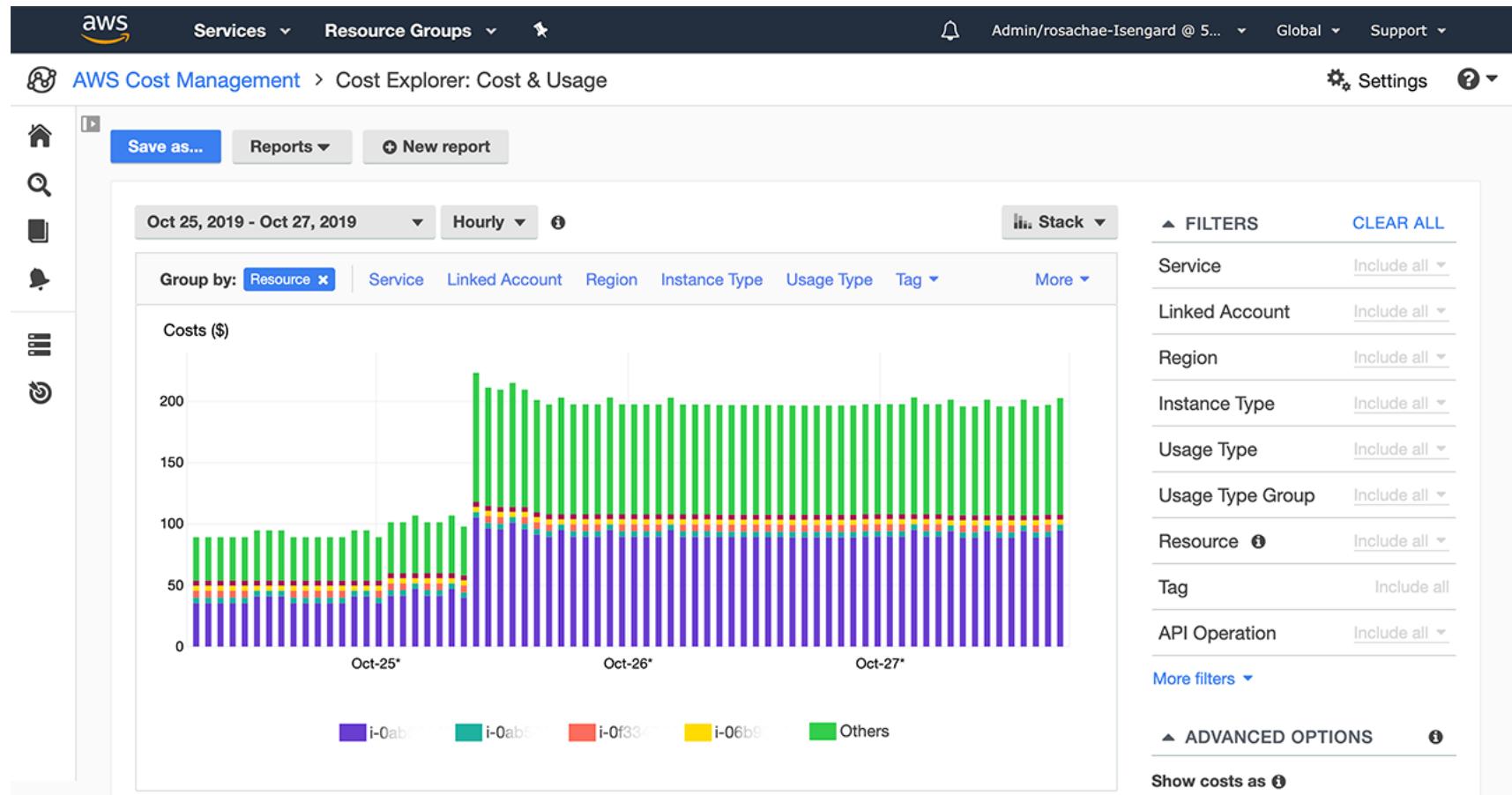


- Visualize, understand, and manage your AWS costs and usage over time
- Create custom reports that analyze cost and usage data.
- Analyze your data at a high level: total costs and usage across all accounts
- Or Monthly, hourly, resource level granularity
- Choose an optimal **Savings Plan** (to lower prices on your bill)
- Forecast usage up to 12 months based on previous usage

# Cost Explorer – Monthly Cost by AWS Service



# Cost Explorer– Hourly & Resource Level



# Cost Explorer – Savings Plan Alternative to Reserved Instances

Recommendation options

Savings Plans type <input checked="" type="radio"/> Compute <input type="radio"/> EC2 Instance	Savings Plans term <input type="radio"/> 1-year <input checked="" type="radio"/> 3-year	Payment option <input checked="" type="radio"/> All upfront <input type="radio"/> Partial upfront <input type="radio"/> No upfront	Based on the past <input type="radio"/> 7 days <input type="radio"/> 30 days <input checked="" type="radio"/> 60 days
--	---	---	--

Recommendation: Purchase a Compute Savings Plan at a commitment of \$2.40/hour

You could save an estimated **\$1,173** monthly by purchasing the recommended Compute Savings Plan.

Based on your past **60 days** of usage, we recommend purchasing a Savings Plan with a commitment of **\$2.40/hour** for a **3-year term**. With this commitment, we project that you could save an average of **\$1.61/hour** - representing a **40%** savings compared to On-Demand. To account for variable usage patterns, this recommendation maximizes your savings by leaving an average **\$0.04/hour** of On-Demand spend.

Before recommended purchase	After recommended purchase (based on your past 60 days of usage)
Monthly On-Demand spend <small> ⓘ</small> <b>\$2,955</b> (\$4.05/hour) Based on your On-Demand spend over the past 60 days	Estimated monthly spend <small> ⓘ</small> <b>\$1,782</b> (\$2.44/hour) Your recommended \$2.40/hour Savings Plans commitment + an average \$0.04/hour of On-Demand spend Estimated monthly savings <small> ⓘ</small> <b>\$1,173</b> (\$1.61/hour) 40% monthly savings over On-Demand \$2,955 - \$1,782 = \$1,173

This recommendation examines your usage over the past 60 days (including your existing Savings Plans and EC2 Reserved Instances) and calculates what your costs would have been had you purchased the recommended Savings Plans. See applicable rates for Savings Plans [here](#). To generate this recommendation, AWS simulates your bill for different commitment amounts and recommends the commitment amount that provides the greatest estimated savings. [Learn more](#)

Recommended Compute Savings Plans [Download CSV](#) [Add selected Savings Plan\(s\) to cart](#)

x	Term	Payment option	Recommended commitment	Estimated hourly savings <small> ⓘ</small>
<input checked="" type="checkbox"/>	3-year	All upfront	\$2.40/hour	\$1.61 (40%)

\*Average hourly spend and minimum hourly spend based on your current on-demand spend for the given instance family.

# Cost Explorer – Forecast Usage



# White Papers and Architectures

Well Architected Framework, Disaster Recovery, etc...

# Section Overview

- Well Architected Framework Whitepaper
- Well Architected Tool
- AWS Trusted Advisor
- Reference architectures resources (for real-world)
- Disaster Recovery on AWS Whitepaper

# Well Architected Framework

## General Guiding Principles

- <https://aws.amazon.com/architecture/well-architected>
- Stop guessing your capacity needs
- Test systems at production scale
- Automate to make architectural experimentation easier
- Allow for evolutionary architectures
  - Design based on changing requirements
- Drive architectures using data
- Improve through game days
  - Simulate applications for flash sale days

# Well Architected Framework

## 6 Pillars

- 1) Operational Excellence
  - 2) Security
  - 3) Reliability
  - 4) Performance Efficiency
  - 5) Cost Optimization
  - 6) Sustainability
- 
- They are not something to balance, or trade-offs, they're a synergy

# AWS Well-Architected Tool



- Free tool to **review your architectures** against the 6 pillars Well-Architected Framework and **adopt architectural best practices**
- How does it work?
  - Select your workload and answer questions
  - Review your answers against the 6 pillars
  - Obtain advice: get videos and documentations, generate a report, see the results in a dashboard
- Let's have a look: <https://console.aws.amazon.com/wellarchitected>

The screenshot shows the AWS Well-Architected Tool interface. At the top, there is a navigation bar with links for 'Well-Architected Tool' and 'Workloads'. Below the navigation is a search bar labeled 'Search by workload name'. To the right of the search bar are buttons for 'Generate report', 'View details', 'Edit', 'Delete', and 'Define workload'. A pagination control shows page 1 of 1. The main area is a table titled 'Workloads' with columns: Name, Overall status, High risks, Medium risks, Improvement status, and Last updated. The table lists five workloads:

Name	Overall status	High risks	Medium risks	Improvement status	Last updated
Internal Employee Portal	Answered	13	2	None	Nov 24, 2018 3:40 PM UTC-8
Mobile app - Android	Answered	9	1	None	Nov 24, 2018 3:43 PM UTC-8
Mobile app - iOS	Answered	0	1	None	Nov 24, 2018 3:49 PM UTC-8
Retail Website- EU	Unanswered	0	0	None	Nov 24, 2018 3:52 PM UTC-8
Retail Website- North America	Unanswered	0	0	None	Nov 24, 2018 3:19 PM UTC-8

At the bottom of the page, there is a footer with links for '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.

<https://aws.amazon.com/blogs/aws/new-aws-well-architected-tool-review-workloads-against-best-practices/>



# Trusted Advisor

- No need to install anything – high level AWS account assessment
- Analyze your AWS accounts and provides recommendation:

**Cost Optimization**



**Performance**



**Security**



**Fault Tolerance**



**Service Limits**



- Core Checks and recommendations – all customers
- Can enable weekly email notification from the console
- Full Trusted Advisor – Available for **Business & Enterprise** support plans
  - Ability to set CloudWatch alarms when reaching limits
  - Programmatic Access using AWS Support API



# Trusted Advisor Checks Examples

- Cost Optimization:
  - low utilization EC2 instances, idle load balancers, under-utilized EBS volumes...
  - Reserved instances & savings plans optimizations,
- Performance:
  - High utilization EC2 instances, CloudFront CDN optimizations
  - EC2 to EBS throughput optimizations, Alias records recommendations
- Security:
  - MFA enabled on Root Account, IAM key rotation, exposed Access Keys
  - S3 Bucket Permissions for public access, security groups with unrestricted ports
- Fault Tolerance:
  - EBS snapshots age, Availability Zone Balance
  - ASG Multi-AZ, RDS Multi-AZ, ELB configuration...
- Service Limits

# More Architecture Examples

- We've explored the most important architectural patterns:
  - **Classic:** EC2, ELB, RDS, ElastiCache, etc...
  - **Serverless:** S3, Lambda, DynamoDB, CloudFront, API Gateway, etc...
- If you want to see more AWS architectures:
- <https://aws.amazon.com/architecture/>
- <https://aws.amazon.com/solutions/>

# Exam Review & Tips

# State of learning checkpoint

- Let's look how far we've gone on our learning journey
- <https://aws.amazon.com/certification/certified-solutions-architect-associate/>

# Practice makes perfect

- If you're new to AWS, take a bit of AWS practice thanks to this course before rushing to the exam
  - The exam recommends you to have one or more years of hands-on experience on AWS
  - Practice makes perfect!
- 
- If you feel overwhelmed by the amount of knowledge you just learned, just go through it one more time

# Proceed by elimination

- Most questions are going to be scenario based
  - For all the questions, rule out answers that you know for sure are wrong
  - For the remaining answers, understand which one makes the most sense
- 
- There are very few trick questions
  - Don't over-think it
  - If a solution seems feasible but highly complicated, it's probably wrong

# Skim the AWS Whitepapers

- You can read about some AWS White Papers here:
  - Architecting for the Cloud: AWS Best Practices
  - AWS Well-Architected Framework
  - AWS Disaster Recovery (<https://aws.amazon.com/disaster-recovery/>)
- Overall we've explored all the most important concepts in the course
- It's never bad to have a look at the whitepapers you think are interesting!

# Read each service's FAQ

- FAQ = Frequently asked questions
- Example: <https://aws.amazon.com/vpc/faqs/>
- FAQ cover a lot of the questions asked at the exam
- They help confirm your understanding of a service

# Get into the AWS Community

- Help out and discuss with other people in the course Q&A
  - Review questions asked by other people in the Q&A
  - Do the practice test in this section
- 
- Read forums online
  - Read online blogs
  - Attend local meetups and discuss with other AWS engineers
  - Watch re-invent videos on Youtube (AWS Conference)

# How will the exam work?

- You'll have to register online at <https://www.aws.training/>
- Fee for the exam is 150 USD
- Provide two identity documents (ID, Credit Card, details are in emails sent to you)
- No notes are allowed, no pen is allowed, no speaking
- 65 questions will be asked in 130 minutes
- At the end you can optionally review all the questions / answers
  
- You will know right away if you passed / failed the exams
- You will not know which answers were right / wrong
- You will know the overall score a few days later (email notification)
- To pass you need a score of at least 720 out of 1000
- If you fail, you can retake the exam again 14 days later

# Congratulations!

# Congratulations!

- Congrats on finishing the course!
- I hope you will pass the exam without a hitch ☺
- If you haven't done so yet, I'd love a review from you!
- If you passed, I'll be more than happy to know I've helped
  - Post it in the Q&A to help & motivate other students. Share your tips!
  - Post it on LinkedIn and tag me!
- Overall, I hope you learned how to use AWS and that you will be a tremendously good AWS Solutions Architect