Amazon Web Services

Ajay J. Singala

About Me

- I am Ajay.
- Enterprise Architect, DevOps, Trainer.
- ~30 years experience.
- Worked with global clients.
- TOGAF Certified Technical Architect.

Agenda

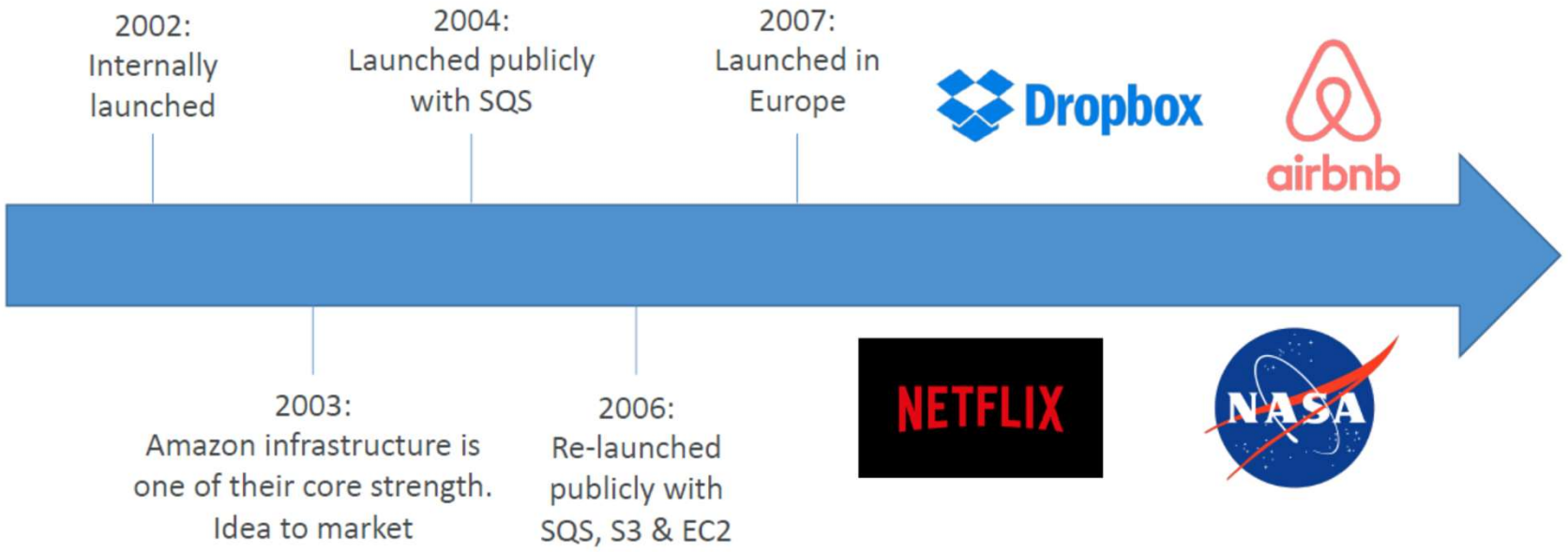
- AWS
- IAM
- EC2
- VPC
- EC2 Instance Storage (EBS)
- AMI
- S3
- Load Balancing & Auto Scaling
- RDS
- DynamoDB
- Route 53
- Serverless Computing
- CloudWatch

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

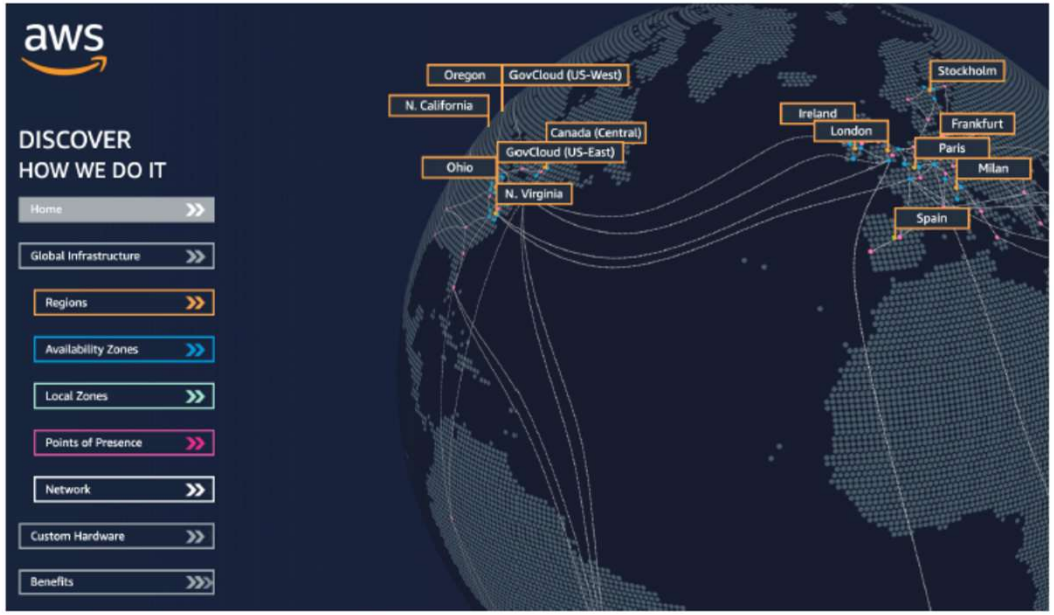
AWS Cloud History



AWS Global Infrastructure

- AWS Regions
- AWS Availability Zones
- AWS Data Centers
- AWS Edge Locations / Points of Presence

- https://infrastructure.aws/



AWS Regions

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

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

US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Africa (Cape Town)	af-south-1
Asia Pacific (Hong Kong)	ap-east-1
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-1
Middle East (Bahrain)	me-south-1
South America (São Paulo)	sa-east-1

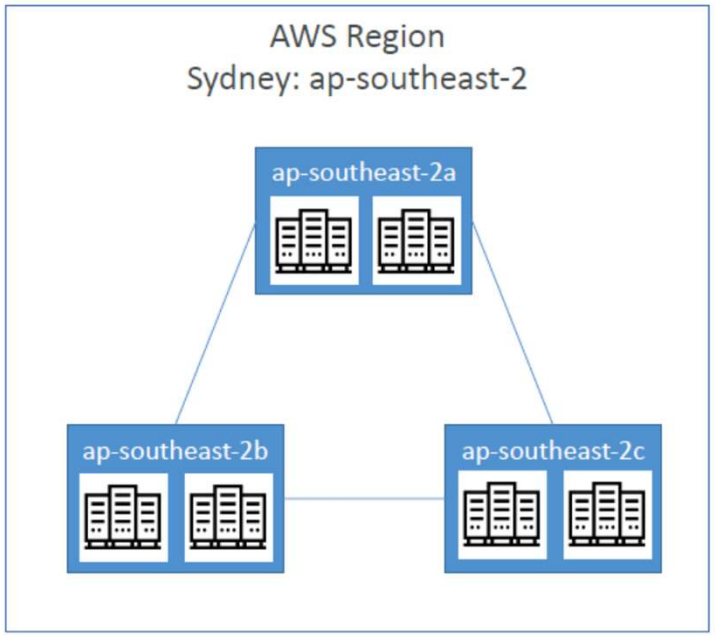
How to choose an AWS Region?

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

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

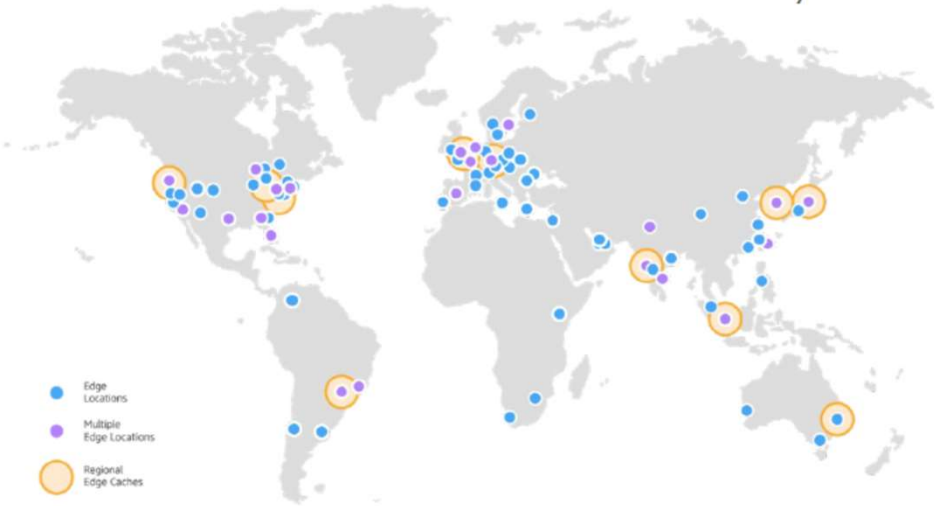
AWS Availability Zones

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



AWS Points of Presence (Edge Locations)

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



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

Tour of the AWS Console

- AWS has Global Services:
  - Identity and Access Management (IAM)
  - Route 53 (DNS service)
  - CloudFront (Content Delivery Network)
  - WAF (Web Application Firewall)

- Most AWS services are Region-scoped:
  - Amazon EC2 (Infrastructure as a Service)
  - Elastic Beanstalk (Platform as a Service)
  - Lambda (Function as a Service)
  - Rekognition (Software as a Service)

- Region Table: https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services

IAM Section

IAM: Users & Groups

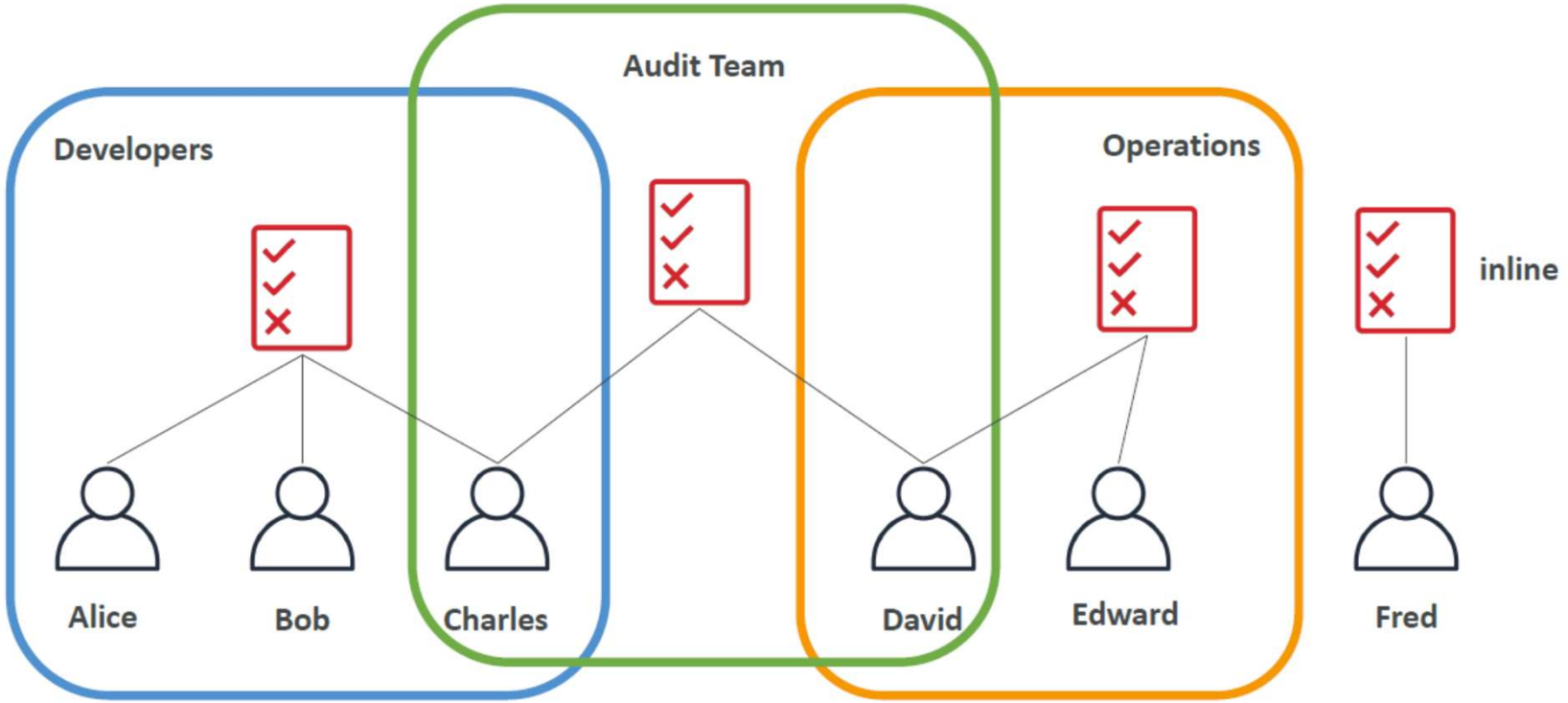
- IAM = Identity and Access Management, **Global** service
- **Root account** created by default, shouldn't be used or shared
- **Users** are people within your organization, and can be grouped
- **Groups** only contain users, not other groups
- Users don't have to belong to a group, and user can belong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

IAM: Permissions

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

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

IAM Policies inheritance



IAM Policies Structure

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

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

IAM – Password Policy

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

Multi Factor Authentication - MFA

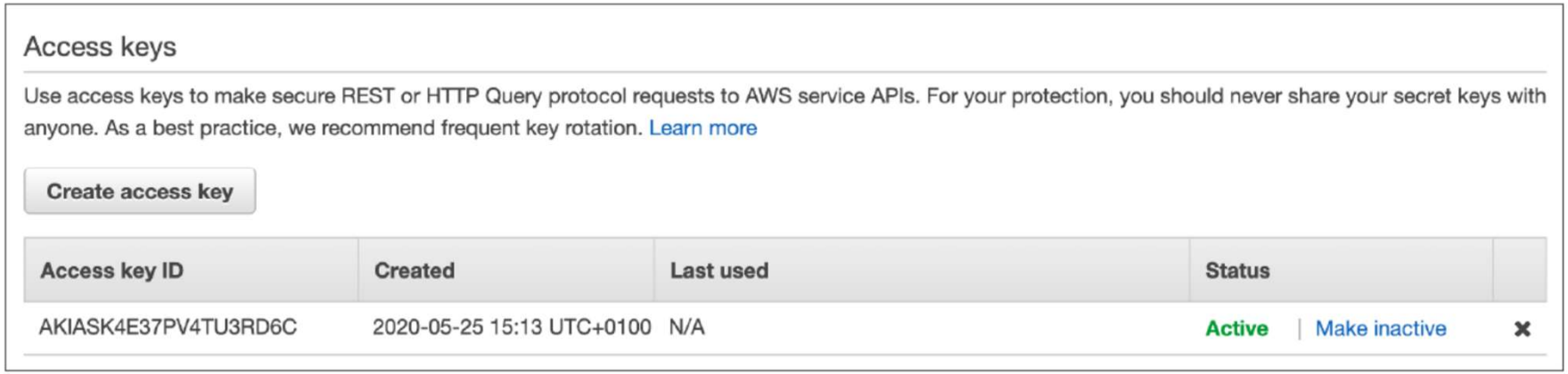
- Users have access to your account and can possibly change configurations or delete resources in your AWS account
- You want to protect your Root Accounts and IAM users
- MFA = password *you know* + security device *you own*

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

How can users access AWS ?

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

Example (Fake) Access Keys



Access keys

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

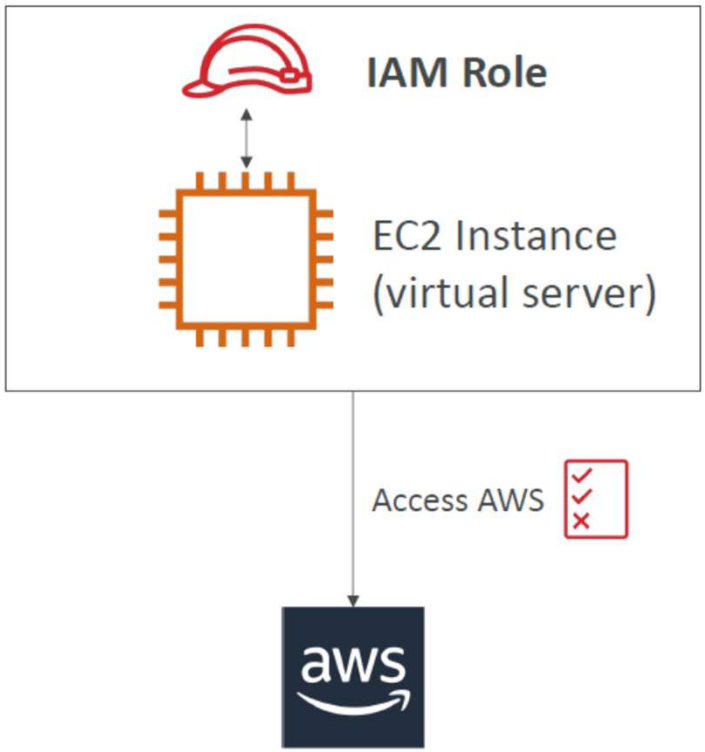
Create access key

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

- Access key ID: AKIASK4E37PV4983d6C
- Secret Access Key: AZPN3zojWozWCndljhB0Unh8239a1bzbzO5fqqkZq
- Remember: don't share your access keys

IAM Roles for Services

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



IAM Security Tools

- IAM Credentials Report (account-level)
  - a report that lists all your account's users and the status of their various credentials

- IAM Access Advisor (user-level)
  - Access advisor shows the service permissions granted to a user and when those services were last accessed.
  - You can use this information to revise your policies.

IAM Guidelines & Best Practices

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

IAM Section – Summary

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

EC2 Basics

Amazon EC2

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

- Knowing EC2 is fundamental to understand how the Cloud works

EC2 sizing & configuration options

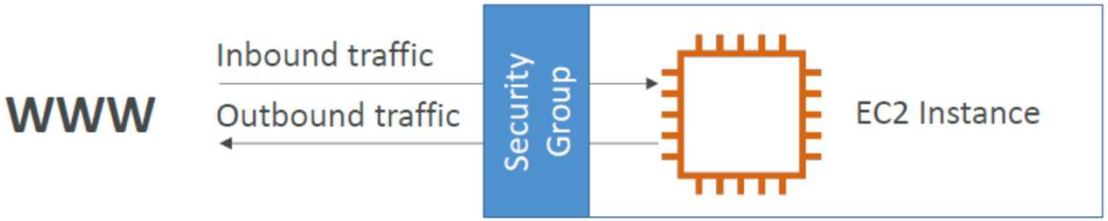
- Operating System (**OS**): Linux, Windows or Mac OS
- How much compute power & cores (**CPU**)
- How much random-access memory (**RAM**)
- How much storage space:
  - Network-attached (**EBS & EFS**)
  - hardware (**EC2 Instance Store**)
- Network card: speed of the card, Public IP address
- Firewall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

EC2 User Data

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

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

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

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 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

Which purchasing option is right for me?

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

Price Comparison
Example – m4.large – us-east-1

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

Virtual Private Connection (VPC)

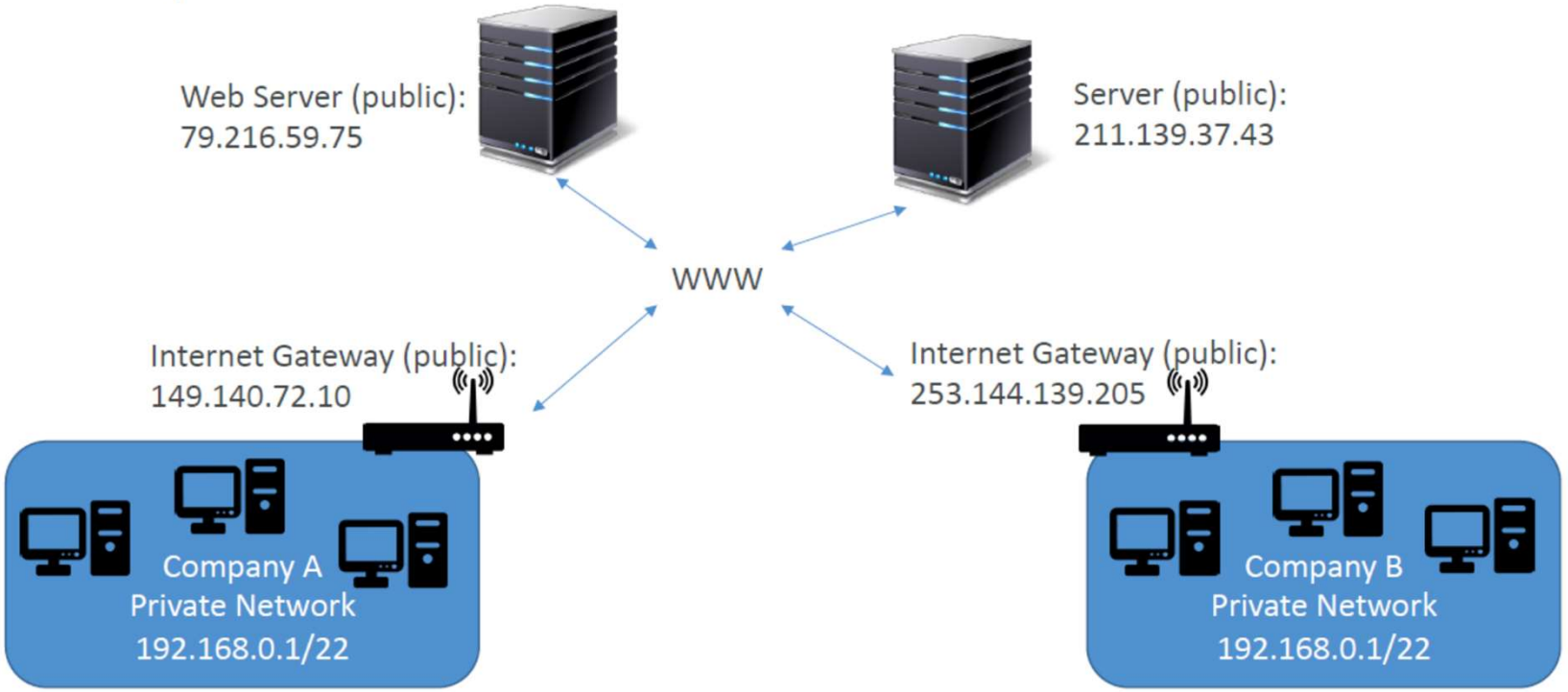
Private vs Public IP (IPv4)

- 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

EC2 Instance Storage Section

What's an EBS Volume?

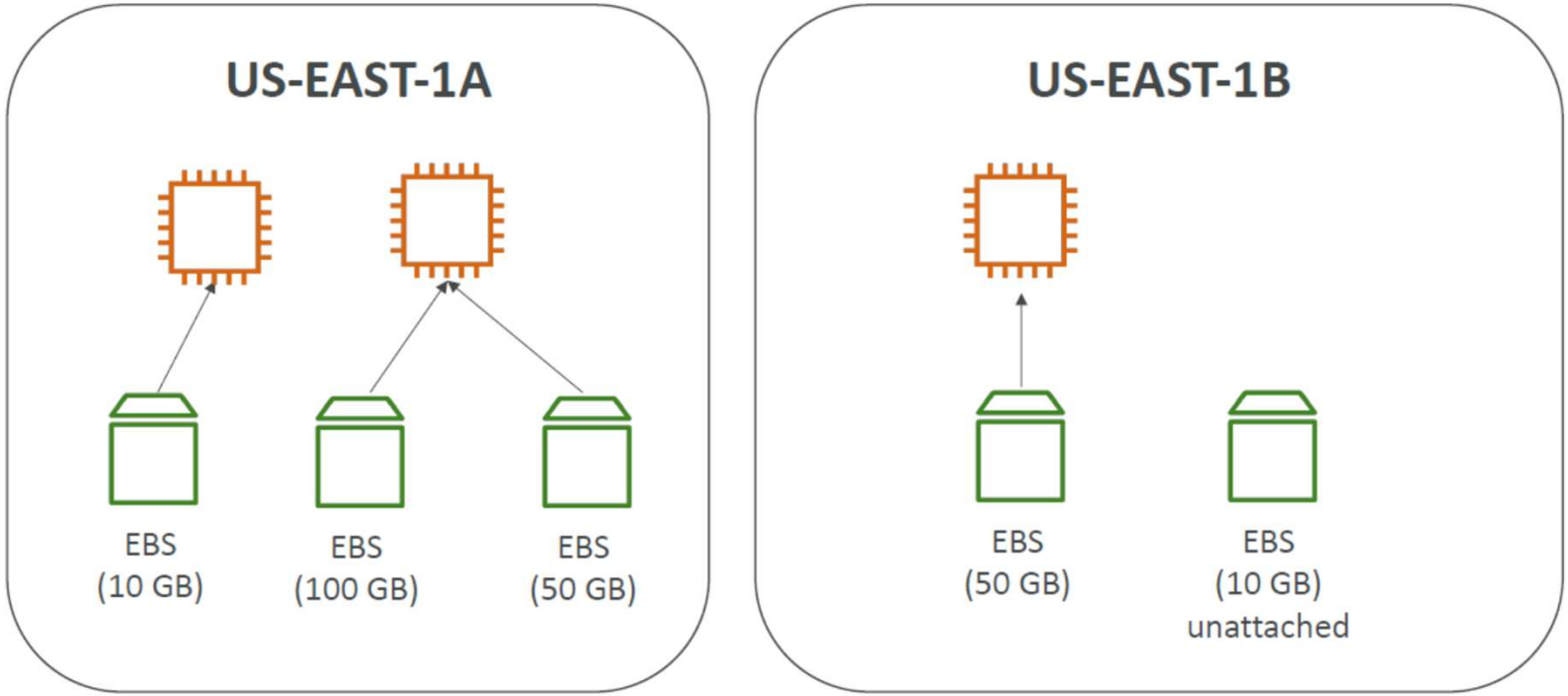
- An EBS (Elastic Block Store) Volume is a network drive you can attach to your instances while they run
- It allows your instances to persist data, even after their termination
- **They can only be mounted to one instance at a time** (at the CCP level)
- They are bound to **a specific availability zone**

- 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

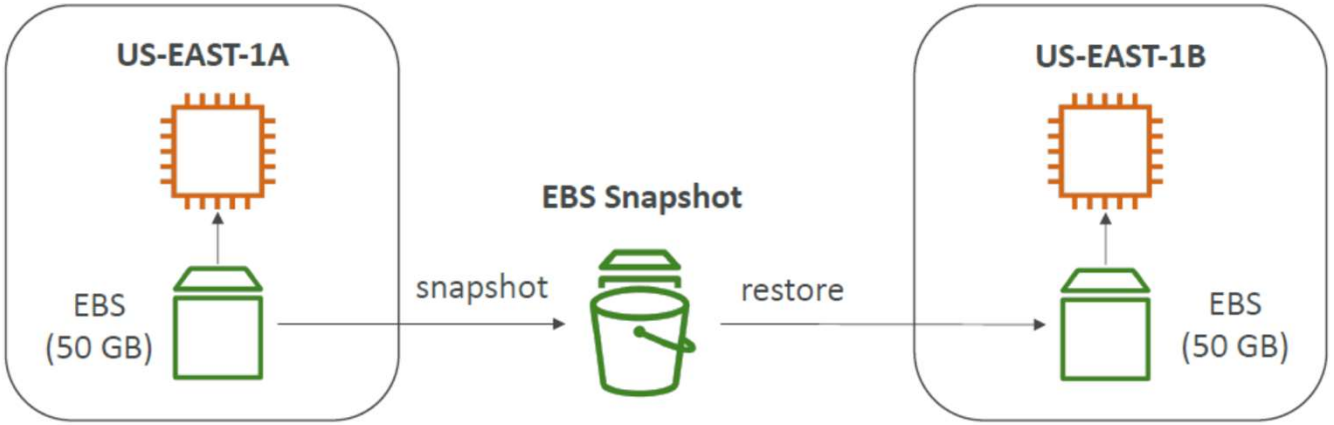


EBS – Delete on Termination attribute

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

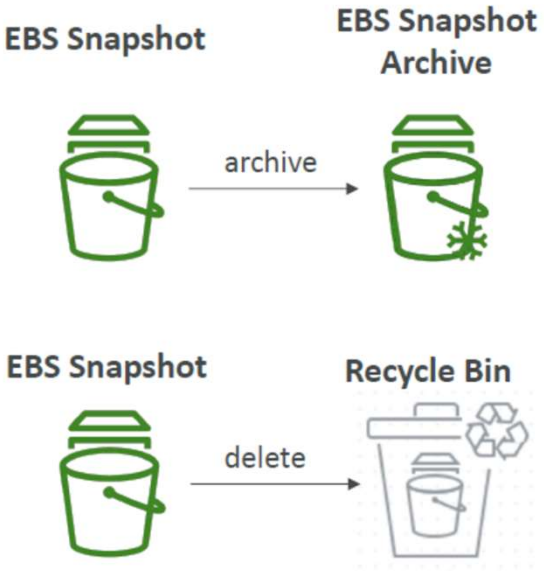
EBS Snapshots

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



EBS Snapshots Features

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



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
  - sc1 (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

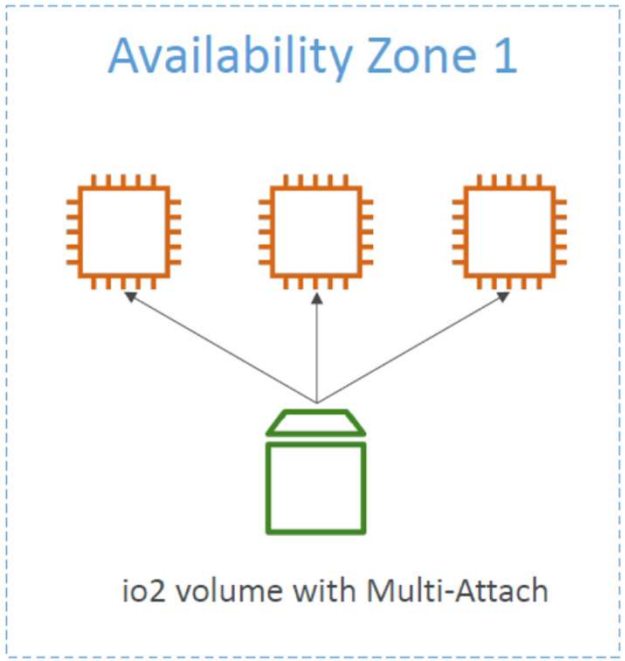
EBS Volume Types Use cases

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 Multi-Attach – io1/io2 family

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



EBS Encryption

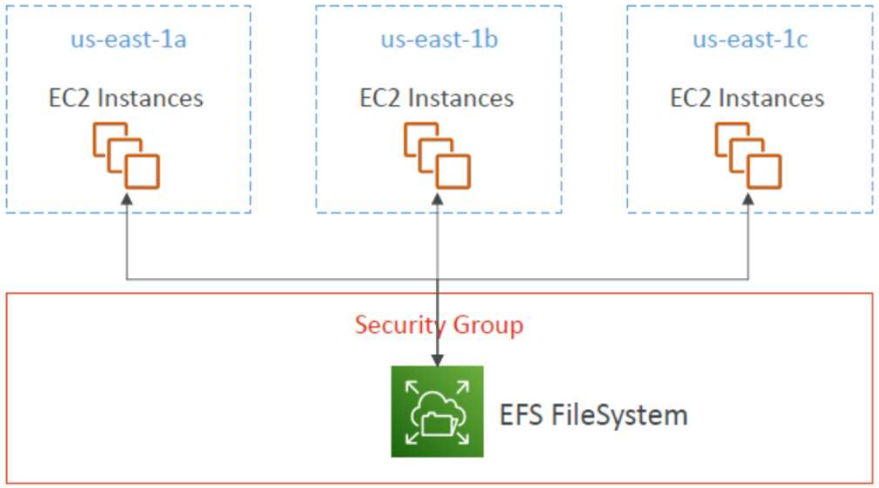
- When you create an encrypted EBS volume, you get the following:
  - Data at rest is encrypted inside the volume
  - All the data in flight moving between the instance and the volume is encrypted
  - All snapshots are encrypted
  - All volumes created from the snapshot
- Encryption and decryption are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

Encryption: encrypt an unencrypted EBS volume

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

Amazon EFS – Elastic File System

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



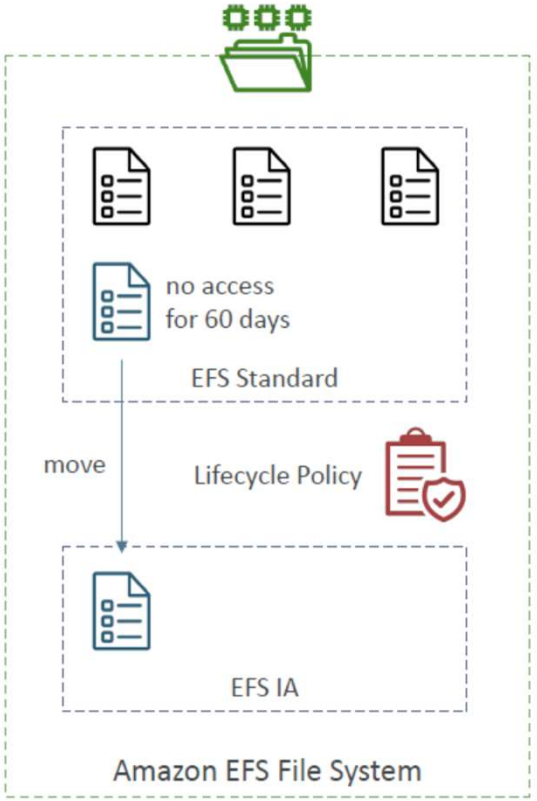
Amazon EFS – Elastic File System

- 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!

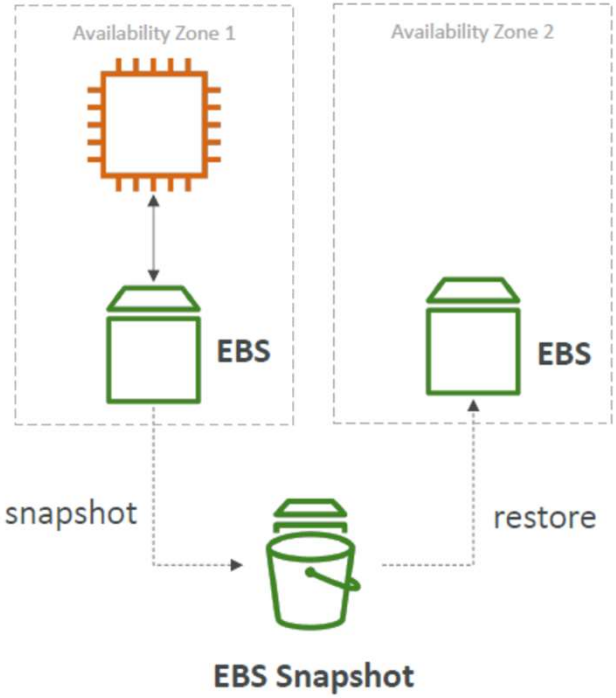
EFS – Storage Classes

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



EBS vs EFS – Elastic Block Storage

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

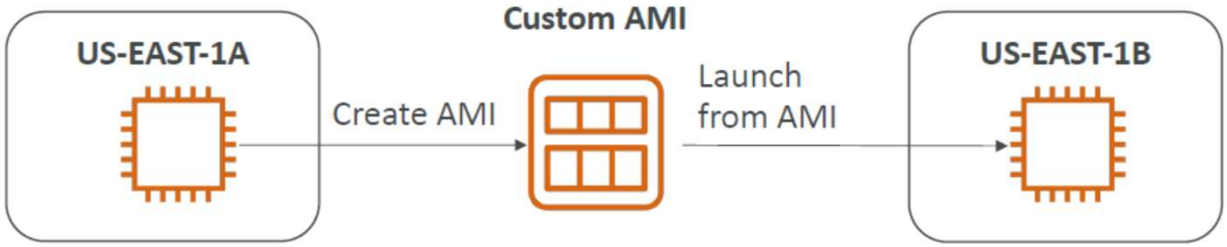


AMI Overview

- 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



Amazon S3 Section

Section introduction

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

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

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

Amazon S3 - Buckets

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

S3 Bucket

Amazon S3 - Objects

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

Object

S3 Bucket with Objects

Amazon S3 – Objects (cont.)

- Object values are the content of the body:
  - Max. Object Size is 5TB (5000GB)
  - If uploading more than 5GB, must use "multi-part upload"

- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

Amazon S3 – Security

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

S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Effect: Allow / Deny
  - Actions: Set of API to Allow or Deny
  - Principal: The account or user to apply the policy to

- Use S3 bucket for policy to:
  - Grant public access to the bucket
  - Force objects to be encrypted at upload
  - Grant access to another account (Cross Account)

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

Amazon S3 – Static Website Hosting

- S3 can host static websites and have them accessible on the Internet

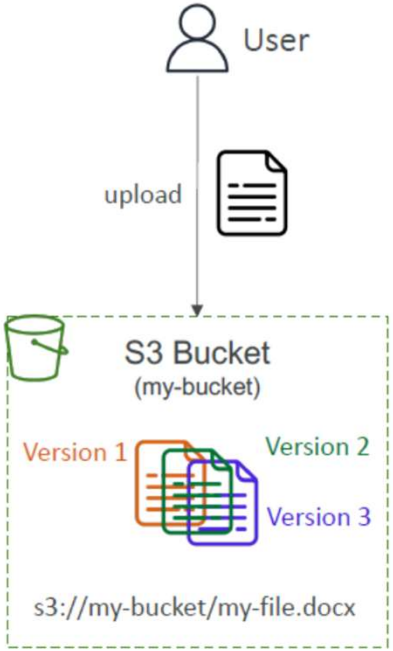
- The website URL will be (depending on the region)
  - http://*bucket-name*.s3-website-*aws-region*.amazonaws.com

  OR
  - http://*bucket-name*.s3-website.*aws-region*.amazonaws.com

- If you get a **403 Forbidden** error, make sure the bucket policy allows public reads!

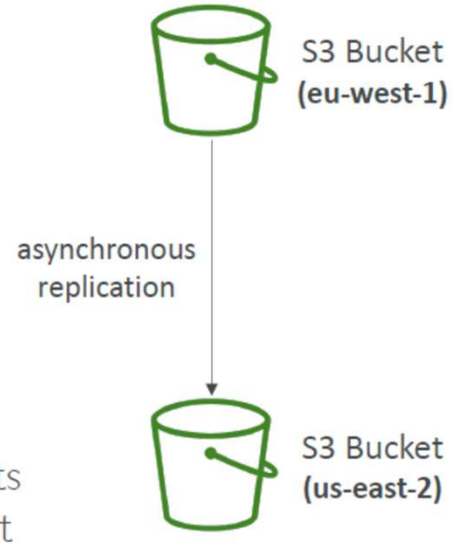
Amazon S3 - Versioning

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



Amazon S3 – Replication (CRR & SRR)

- **Must enable Versioning** in source and destination buckets
- **Cross-Region Replication (CRR)**
- **Same-Region Replication (SRR)**
- Buckets can be in different AWS accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3



- Use cases:
  - **CRR** – compliance, lower latency access, replication across accounts
  - **SRR** – log aggregation, live replication between production and test accounts

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 (99.999999999%, 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 (99.999999999%) in a single AZ; data lost when AZ is destroyed
  - 99.5% Availability
  - Use Cases: Storing secondary backup copies of on-premises data, or data you can recreate

Amazon S3 Glacier Storage Classes

- Low-cost object storage meant for archiving / backup
- Pricing: price for storage + object retrieval cost

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

S3 Intelligent-Tiering

- Small monthly monitoring and auto-tiering fee
- Moves objects automatically between Access Tiers based on usage
- There are no retrieval charges in S3 Intelligent-Tiering

- *Frequent Access tier (automatic):* default tier
- *Infrequent Access tier (automatic)*: objects not accessed for 30 days
- *Archive Instant Access tier (automatic):* objects not accessed for 90 days
- *Archive Access tier (optional):* configurable from 90 days to 700+ days
- *Deep Archive Access tier (optional):* config. from 180 days to 700+ days

S3 Storage Classes Comparison

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

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

S3 Storage Classes – Price Comparison Example: us-east-1

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Storage Cost (per GB per month)	$0.023	$0.0025 - $0.023	%0.0125	$0.01	$0.004	$0.0036	$0.00099
Retrieval Cost (per 1000 request)	**GET:** $0.0004 **POST:** $0.005	**GET:** $0.0004 **POST:** $0.005	**GET:** $0.001 **POST:** $0.01	**GET:** $0.001 **POST:** $0.01	**GET:** $0.01 **POST:** $0.02	**GET:** $0.0004 **POST:** $0.03 **Expedited:** $10 **Standard:** $0.05 **Bulk:** free	**GET:** $0.0004 **POST:** $0.05 **Standard:** $0.10 **Bulk:** $0.025
Retrieval Time	Instantaneous					**Expedited** (1 – 5 mins) **Standard** (3 – 5 hours) Bulk (5 – 12 hours)	**Standard** (12 hours) **Bulk** (48 hours)
Monitoring Cost (pet 1000 objects)		$0.0025					

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

Load Balancing and Auto Scaling

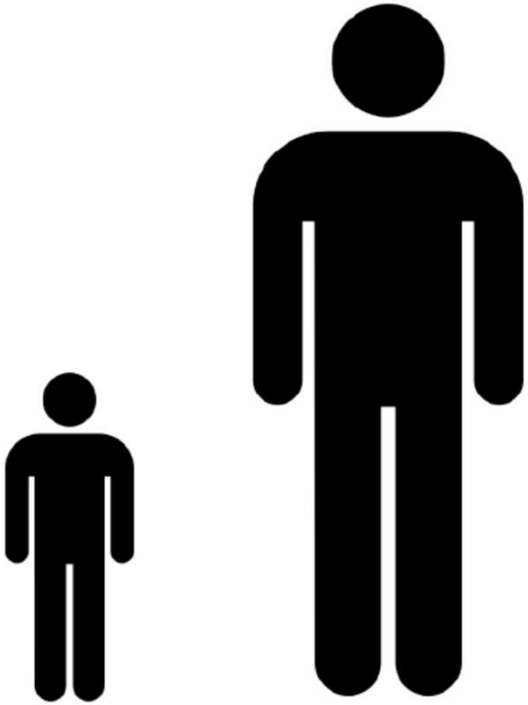
Scalability & High Availability

- 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)



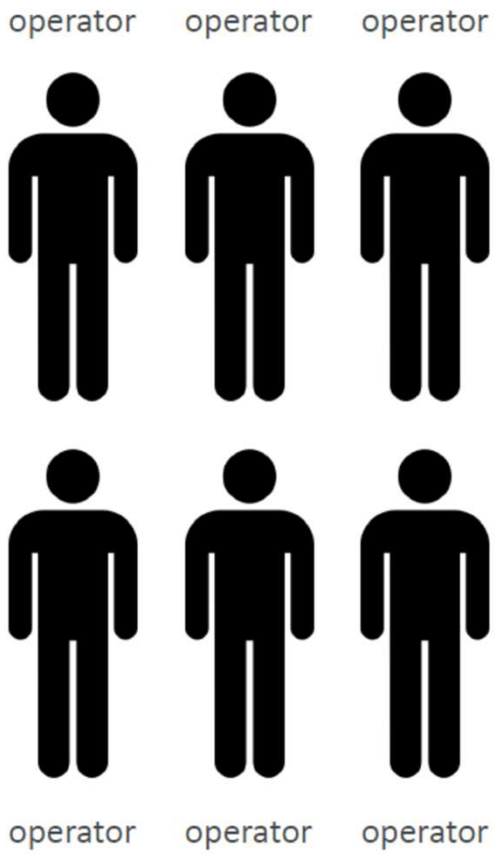
junior operator senior operator

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

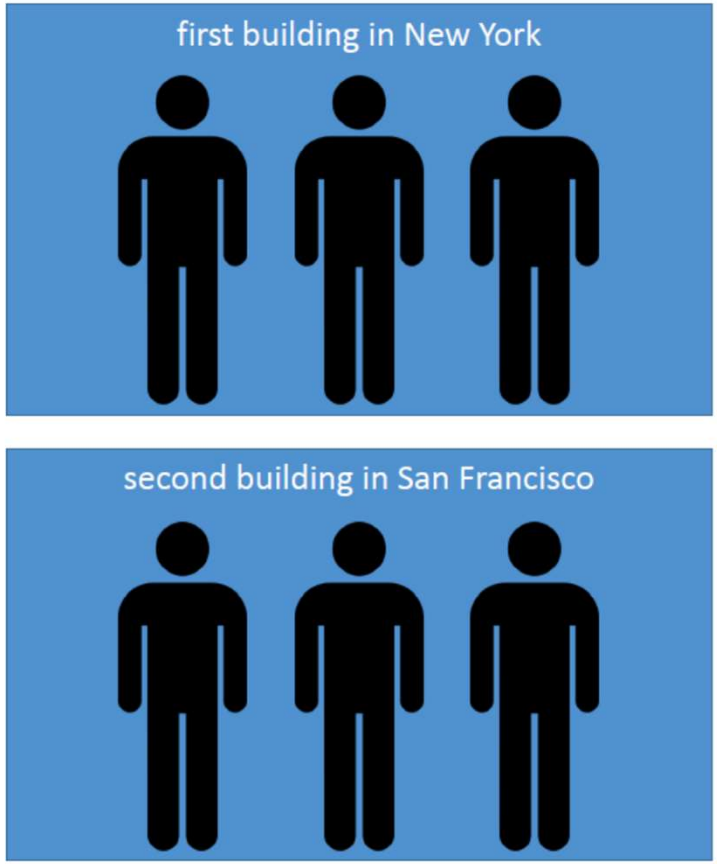
- It's easy to horizontally scale thanks the cloud offerings such as Amazon EC2



High Availability

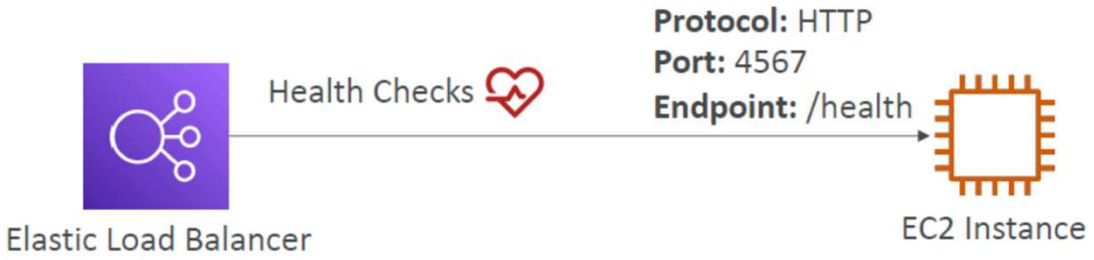
- High Availability usually goes hand in hand with horizontal scaling
- High availability means running your application / system in at least 2 data centers (== Availability Zones)
- The goal of high availability is to survive a data center loss

- The high availability can be passive (for RDS Multi AZ for example)
- The high availability can be active (for horizontal scaling)



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



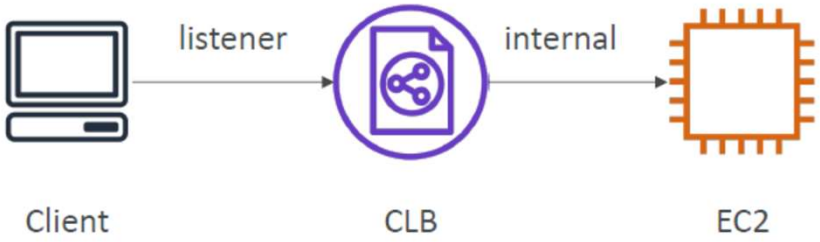
Types of load balancer on AWS

- 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

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

Network Load Balancer (v2)

- Network load balancers (Layer 4) allow to:
  - Forward TCP & UDP traffic to your instances
  - Handle millions of request per seconds
  - Less latency ~100 ms (vs 400 ms for ALB)

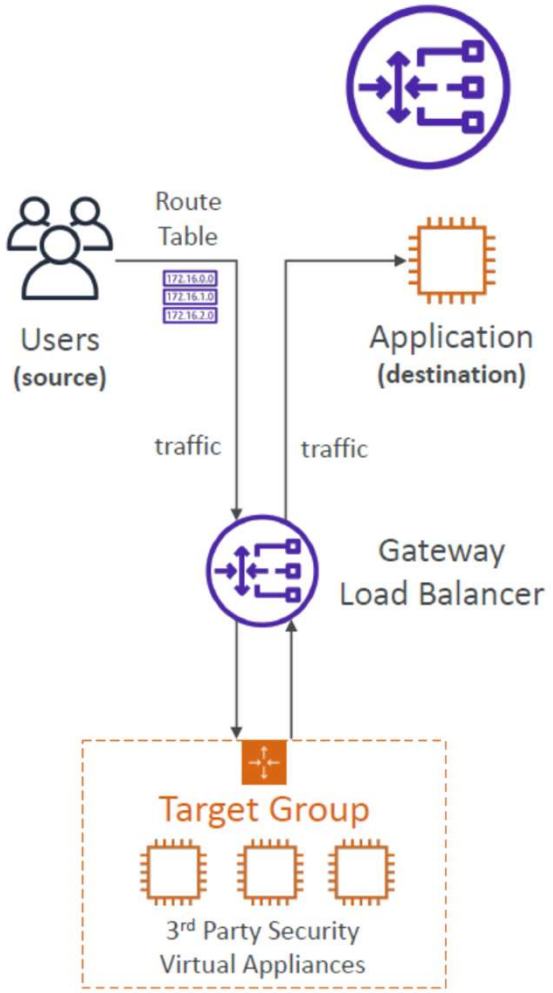
- NLB has one static IP per AZ, and supports assigning Elastic IP (helpful for whitelisting specific IP)

- NLB are used for extreme performance, TCP or UDP traffic
- Not included in the AWS free tier

Gateway Load Balancer

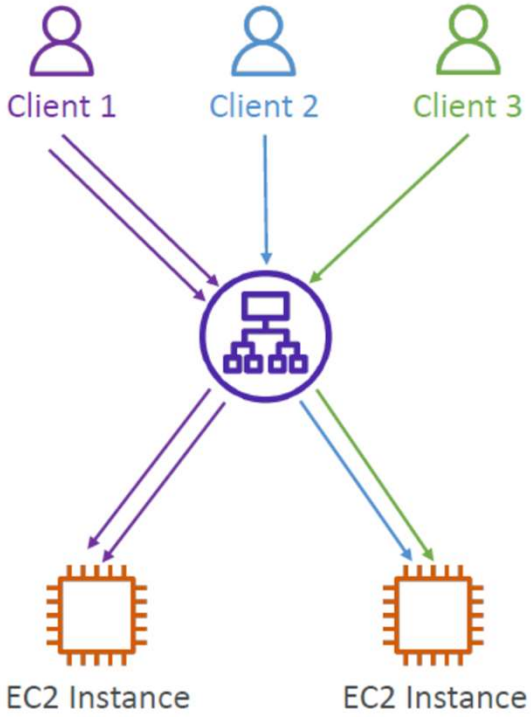
- Deploy, scale, and manage a fleet of 3rd party network virtual appliances in AWS
- Example: Firewalls, Intrusion Detection and Prevention Systems, Deep Packet Inspection Systems, payload manipulation, …

- Operates at Layer 3 (Network Layer) – IP Packets
- Combines the following functions:
  - **Transparent Network Gateway** – single entry/exit for all traffic
  - **Load Balancer** – distributes traffic to your virtual appliances
- Uses the **GENEVE** protocol on port **6081**



Sticky Sessions (Session Affinity)

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



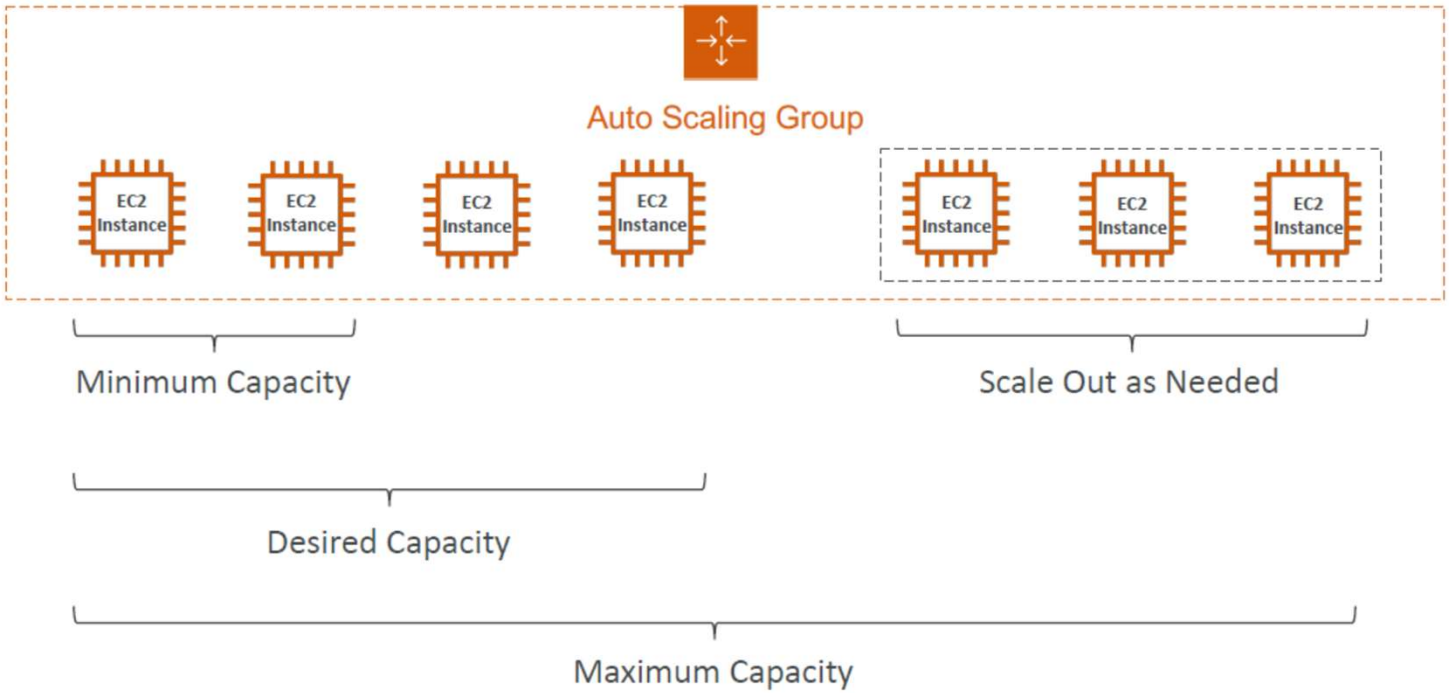
What's an Auto Scaling Group?

- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly

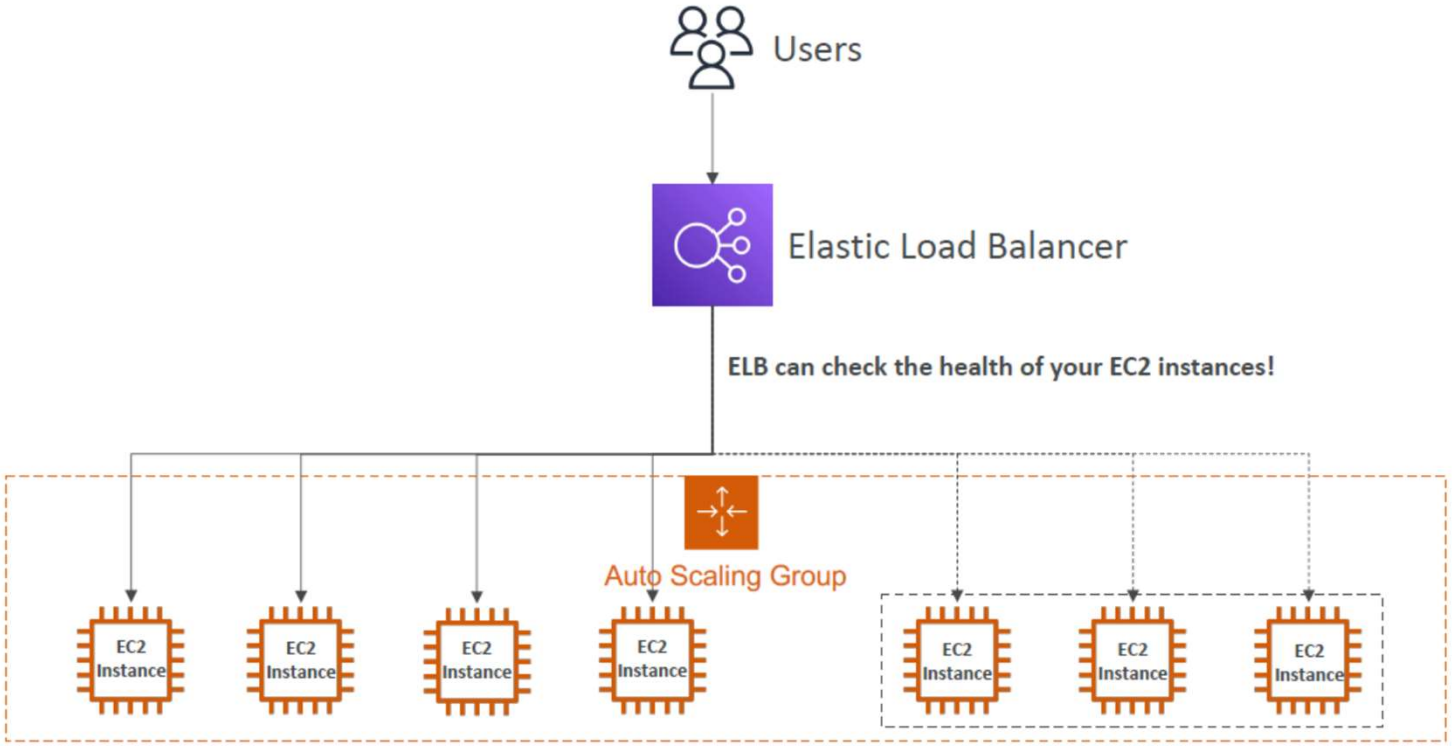
- The goal of an Auto Scaling Group (ASG) is to:
  - Scale out (add EC2 instances) to match an increased load
  - Scale in (remove EC2 instances) to match a decreased load
  - Ensure we have a minimum and a maximum number of EC2 instances running
  - Automatically register new instances to a load balancer
  - Re-create an EC2 instance in case a previous one is terminated (ex: if unhealthy)

- ASG are free (you only pay for the underlying EC2 instances)

Auto Scaling Group in AWS

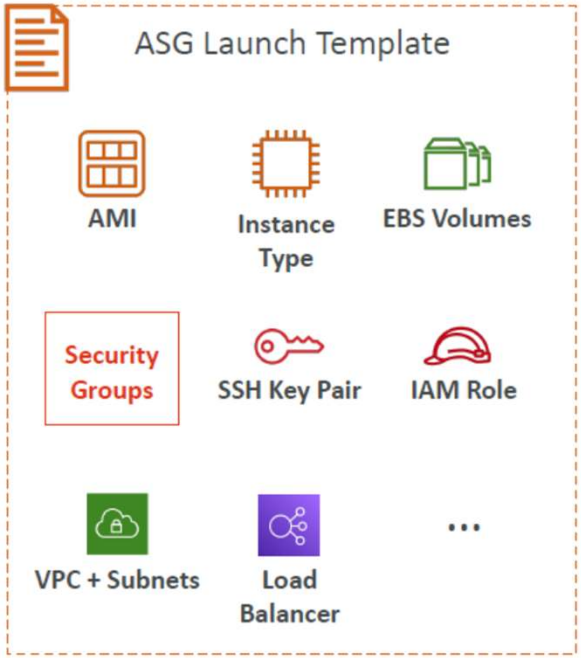


Auto Scaling Group in AWS With Load Balancer



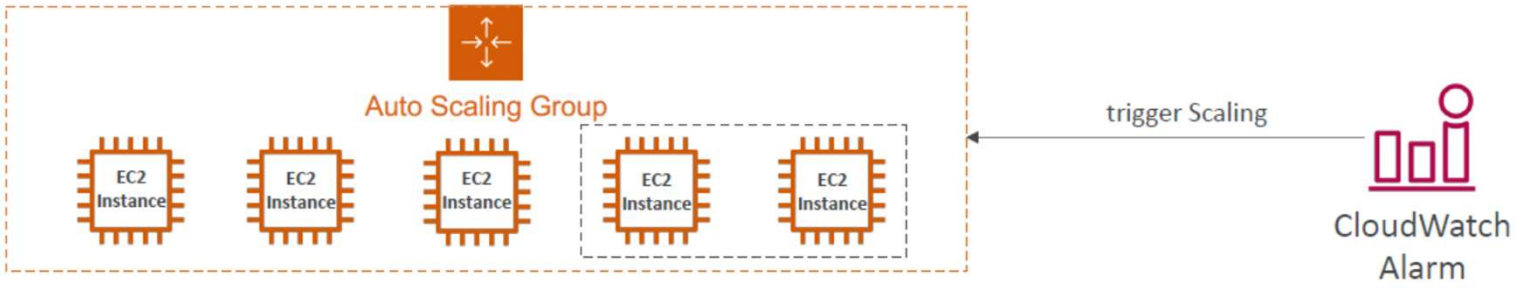
Auto Scaling Group Attributes

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



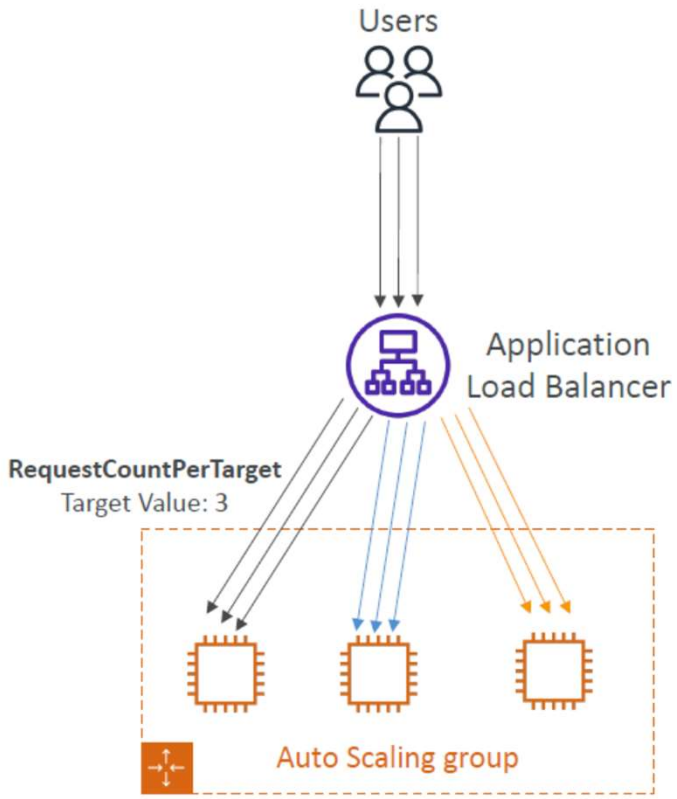
Auto Scaling - CloudWatch Alarms & Scaling

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



Good metrics to scale on

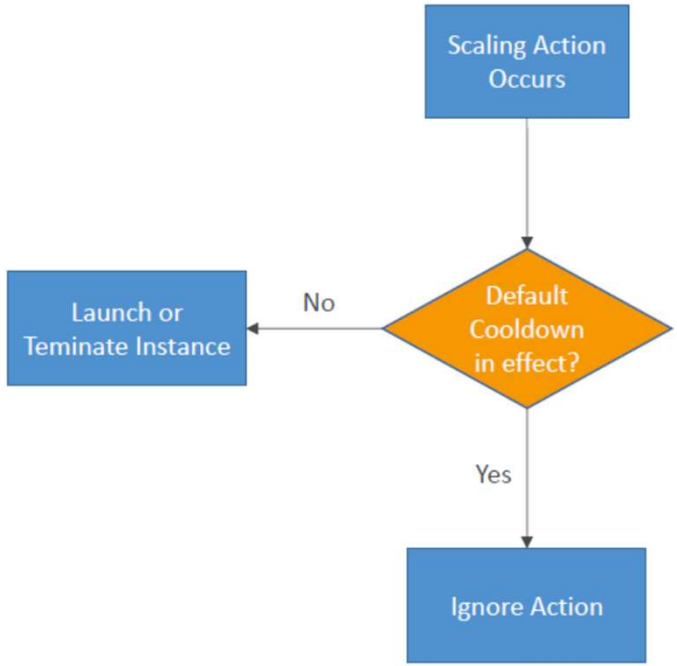
- **CPUUtilization**: Average CPU utilization across your instances
- **RequestCountPerTarget**: to make sure the number of requests per EC2 instances is stable
- **Average Network In / Out** (if you're application is network bound)
- **Any custom metric** (that you push using CloudWatch)



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 fasters and reduce the cooldown period



RDS (Relational Database Service)

Amazon RDS Overview

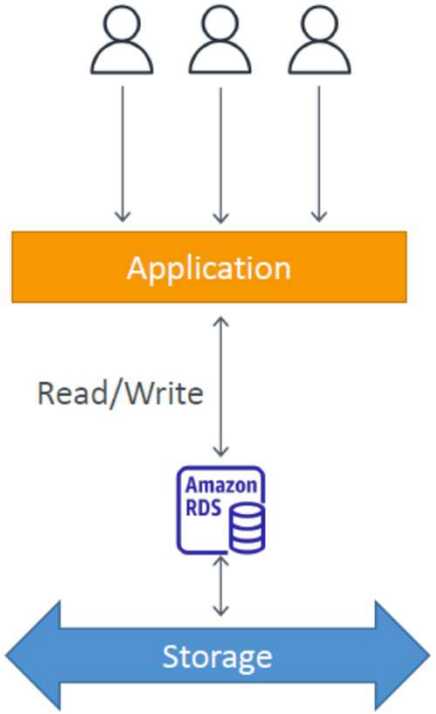
- RDS stands for Relational Database Service
- It's a managed DB service for DB use SQL as a query language.
- It allows you to create databases in the cloud that are managed by AWS
  - Postgres
  - MySQL
  - MariaDB
  - Oracle
  - Microsoft SQL Server
  - Aurora (AWS Proprietary database)

Advantage over using RDS versus deploying DB on EC2

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

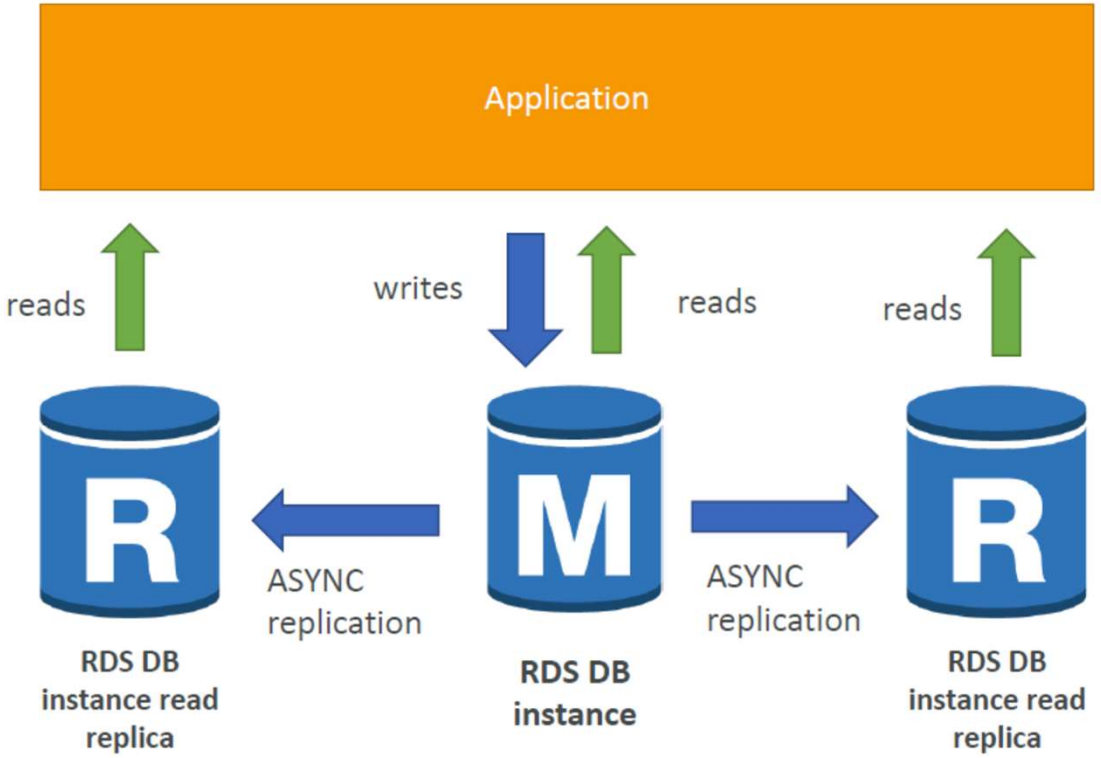
RDS – Storage Auto Scaling

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



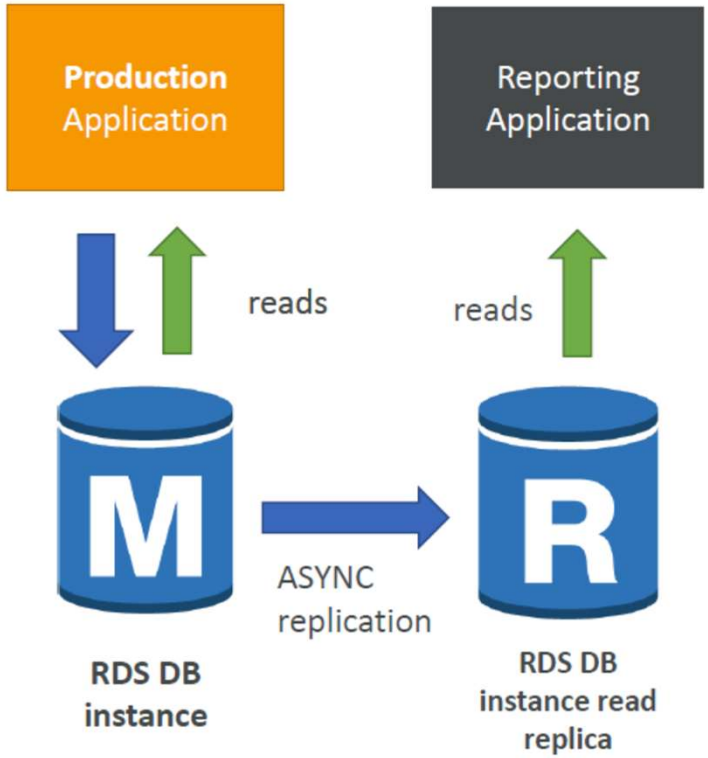
RDS Read Replicas for read scalability

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



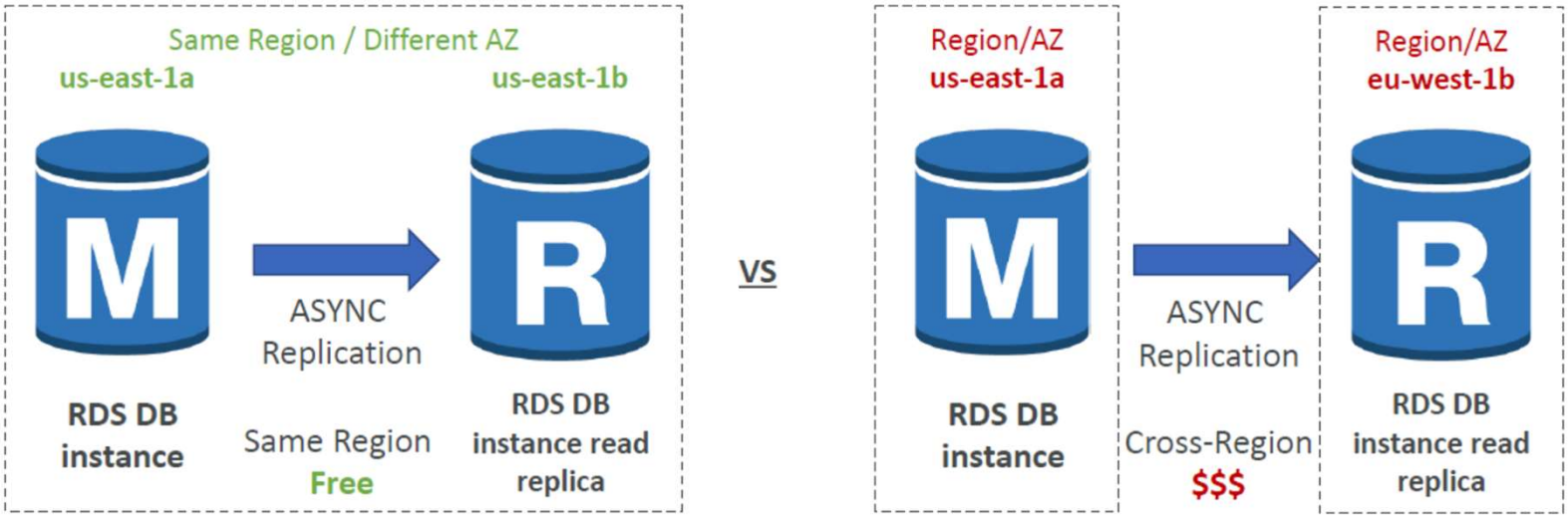
RDS Read Replicas – Use Cases

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



RDS Read Replicas – Network Cost

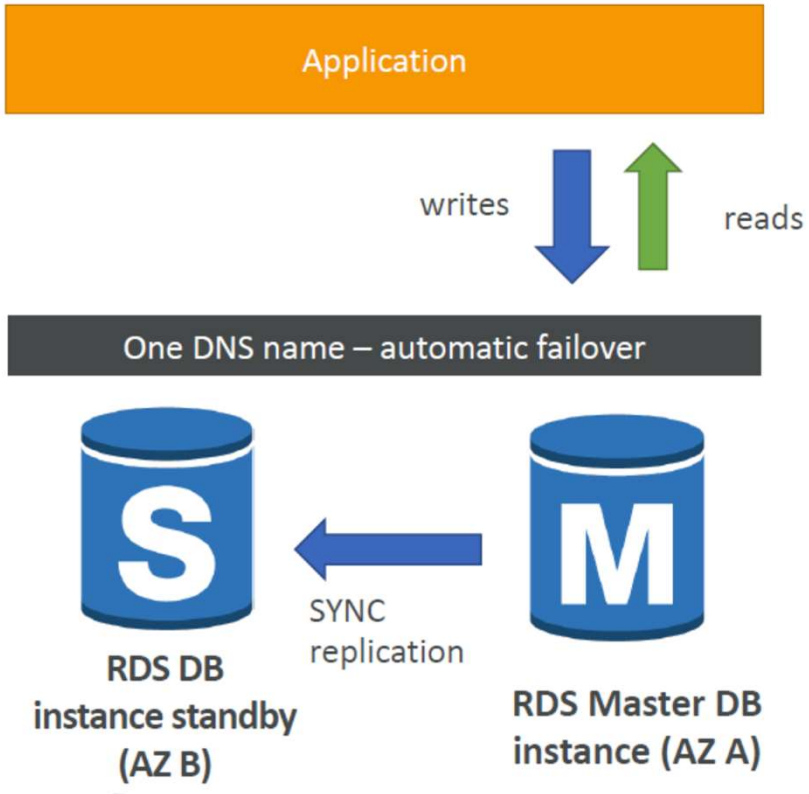
- In AWS there's a network cost when data goes from one AZ to another
- For RDS Read Replicas within the same region, you don't pay that fee



RDS Multi AZ (Disaster Recovery)

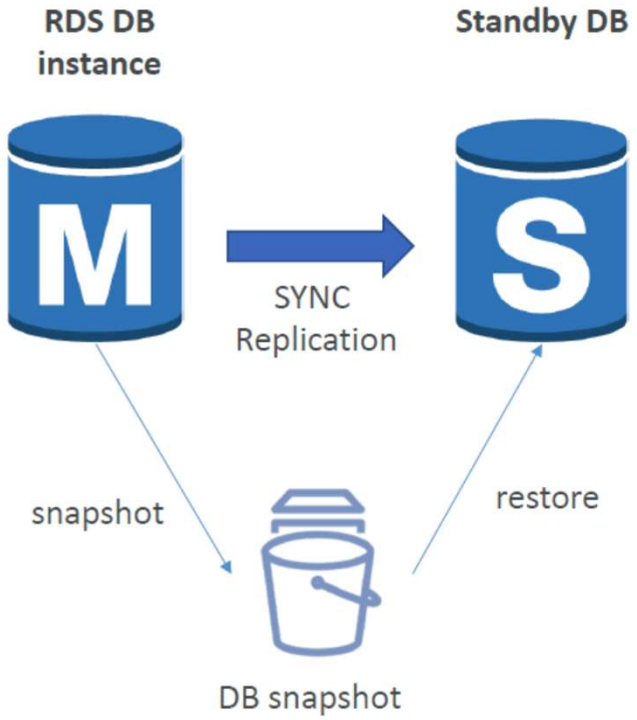
- SYNC replication
- One DNS name – automatic app failover to standby
- Increase availability
- Failover in case of loss of AZ, loss of network, instance or storage failure
- No manual intervention in apps
- Not used for scaling

- Note: The Read Replicas be setup as Multi AZ for Disaster Recovery (DR)



RDS – From Single-AZ to Multi-AZ

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

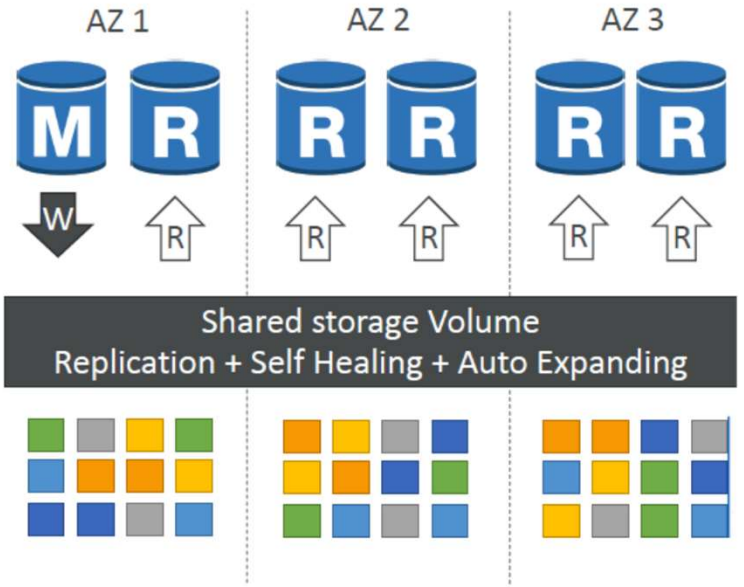


Amazon Aurora

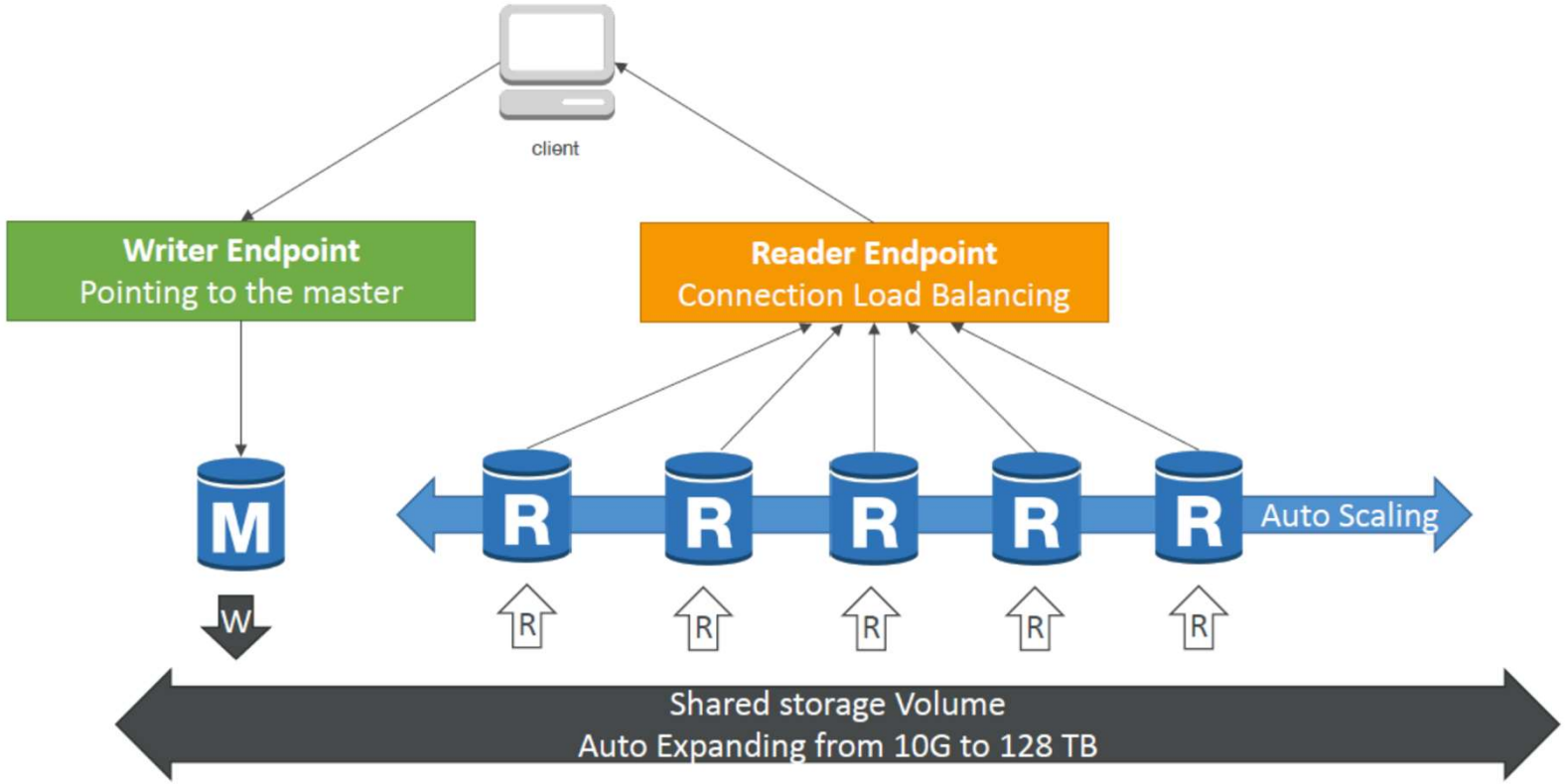
- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is "AWS cloud optimized" and claims 5x performance improvement over MySQL on RDS, over 3x the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

Aurora High Availability and Read Scaling

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



Aurora DB Cluster



Features of Aurora

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

RDS Backups

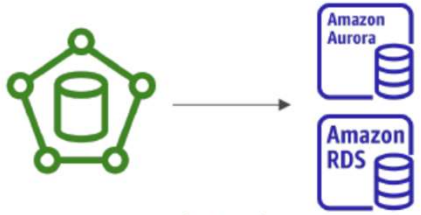
- Automated backups:
  - Daily full backup of the database (during the backup window)
  - Transaction logs are backed-up by RDS every 5 minutes
  - => ability to restore to any point in time (from oldest backup to 5 minutes ago)
  - 1 to 35 days of retention, set 0 to disable automated backups
- Manual DB Snapshots
  - Manually triggered by the user
  - Retention of backup for as long as you want

- Trick: in a stopped RDS database, you will still pay for storage. If you plan on stopping it for a long time, you should snapshot & restore instead

Aurora Backups

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

RDS & Aurora Restore options



- **Restoring a RDS / Aurora backup or a snapshot** creates a new database

- **Restoring MySQL RDS database from S3**
  - Create a backup of your on-premises database
  - Store it on Amazon S3 (object storage)
  - Restore the backup file onto a new RDS instance running MySQL

- **Restoring MySQL Aurora cluster from S3**
  - Create a backup of your on-premises database using Percona XtraBackup
  - Store the backup file on Amazon S3
  - Restore the backup file onto a new Aurora cluster running MySQL

DynamoDB

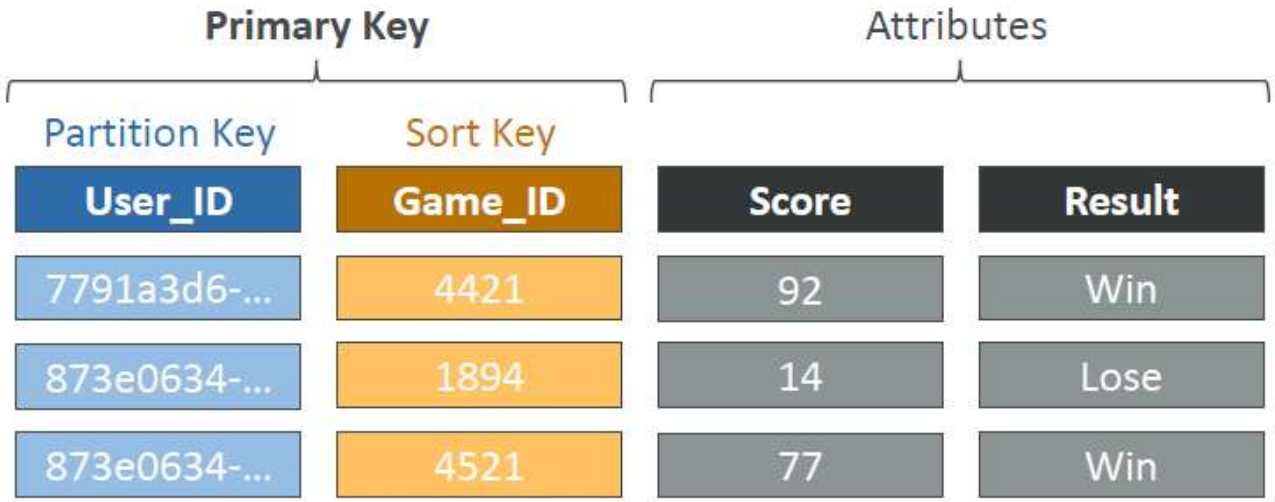
Amazon DynamoDB

- Fully managed, highly available with replication across multiple AZs
- NoSQL database - not a relational database - with transaction support
- Scales to massive workloads, distributed database
- Millions of requests per seconds, trillions of row, 100s of TB of storage
- Fast and consistent in performance (single-digit millisecond)
- Integrated with IAM for security, authorization and administration
- Low cost and auto-scaling capabilities
- No maintenance or patching, always available
- 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
- Therefore, in DynamoDB you can rapidly evolve schemas

DynamoDB – Table example



DynamoDB – Backups for disaster recovery

- Continuous backups using point-in-time recovery (PITR)
  - Optionally enabled for the last 35 days
  - Point-in-time recovery to any time within the backup window
  - The recovery process creates a new table

- On-demand backups
  - Full backups for long-term retention, until explicitely deleted
  - Doesn't affect performance or latency
  - Can be configured and managed in AWS Backup (enables cross-region copy)
  - The recovery process creates a new table

Route 53 Section

What is DNS?

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

.com

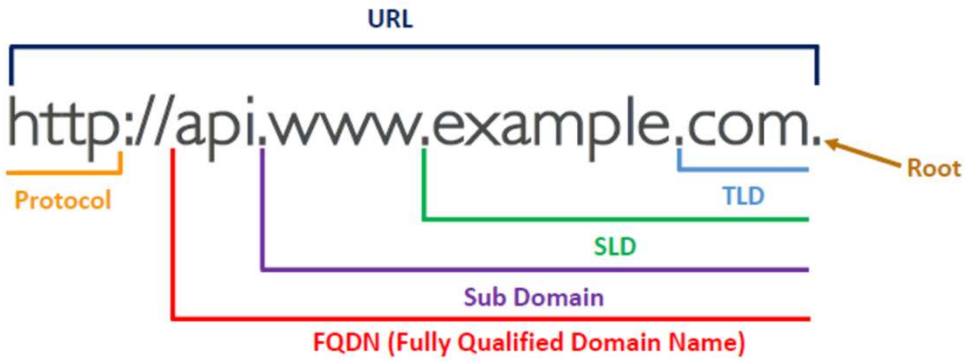
example.com

www.example.com

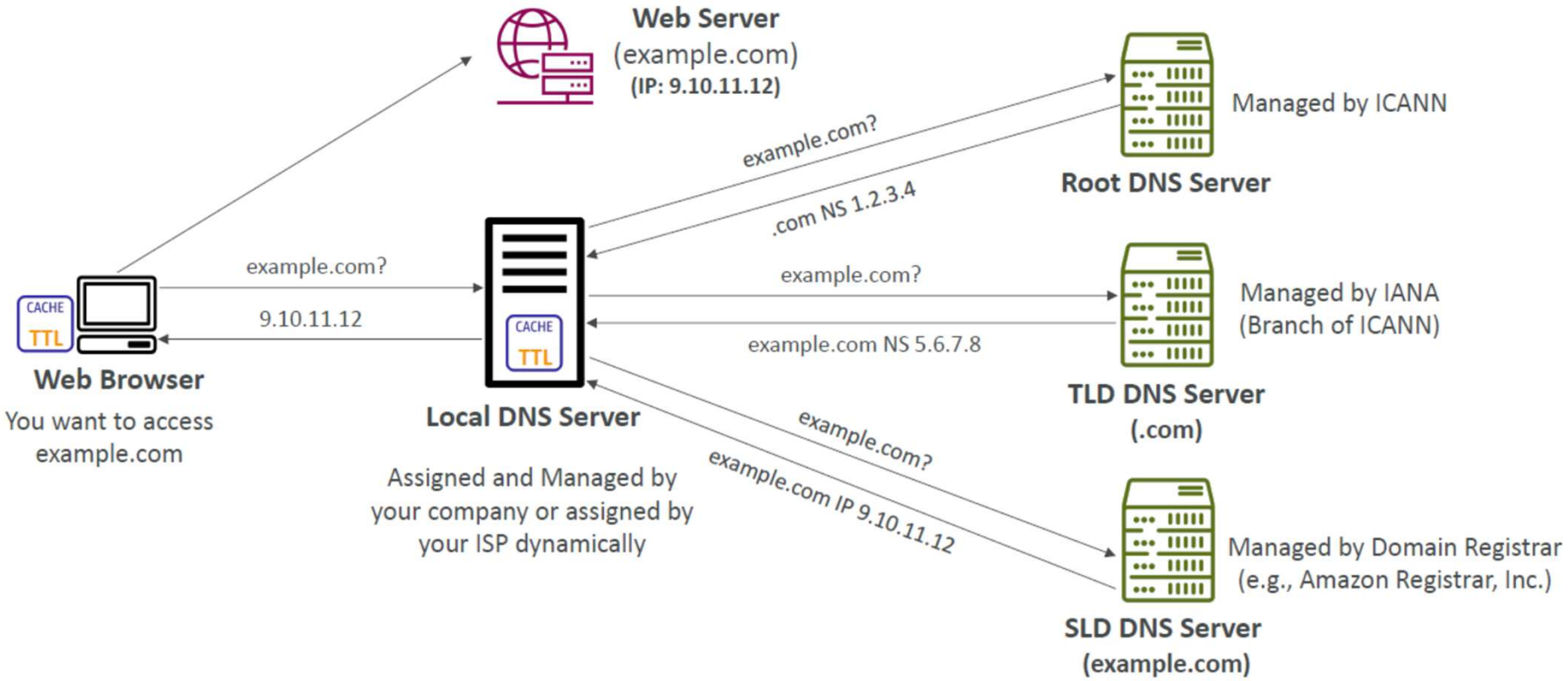
api.example.com

DNS Terminologies

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

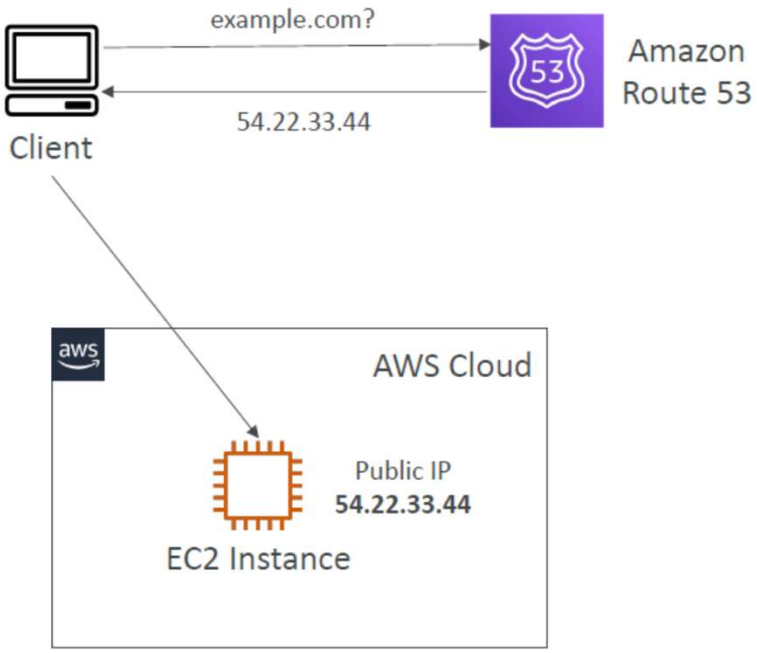


How DNS Works



Amazon Route 53

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



Route 53 – Records

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

Route 53 – Record Types

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

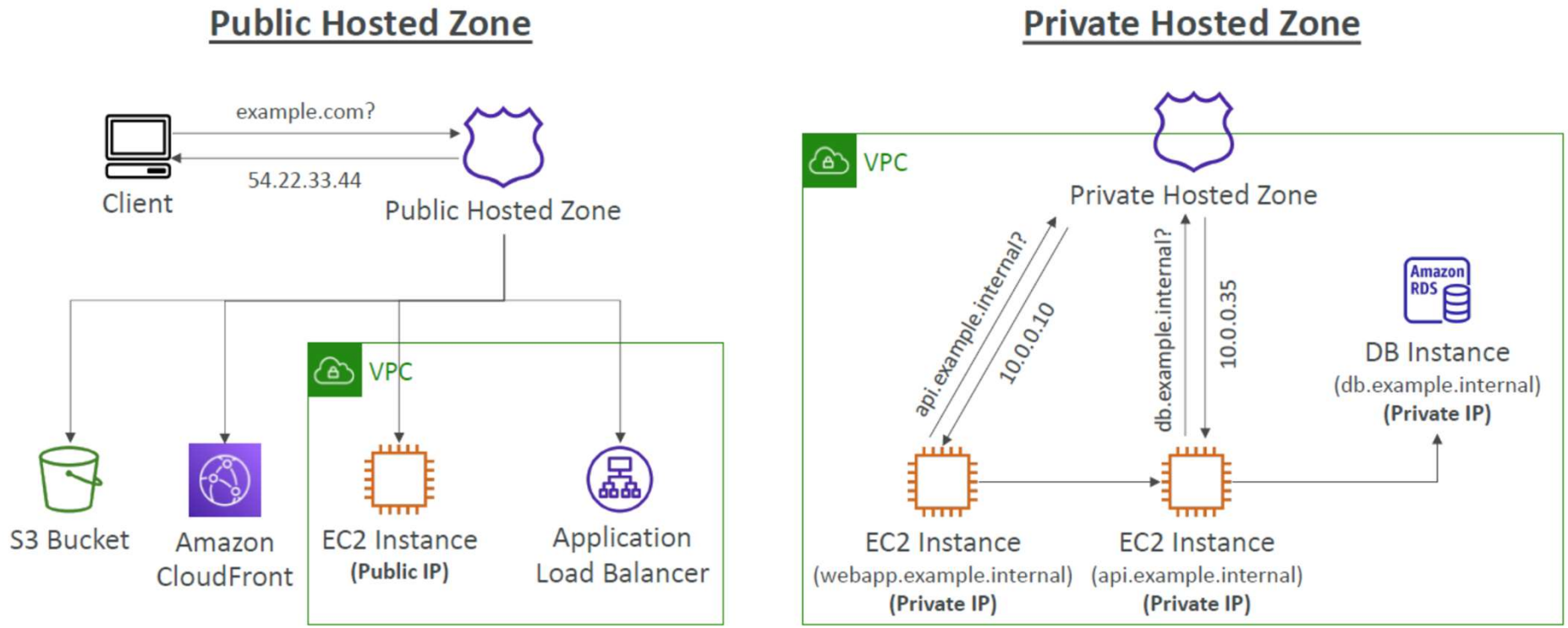
Route 53 – Hosted Zones

- A container for records that define how to route traffic to a domain and its subdomains

- **Public Hosted Zones** – contains records that specify how to route traffic on the Internet (public domain names)
  application1.mypublicdomain.com
- **Private Hosted Zones** – contain records that specify how you route traffic within one or more VPCs (private domain names)
  application1.company.internal

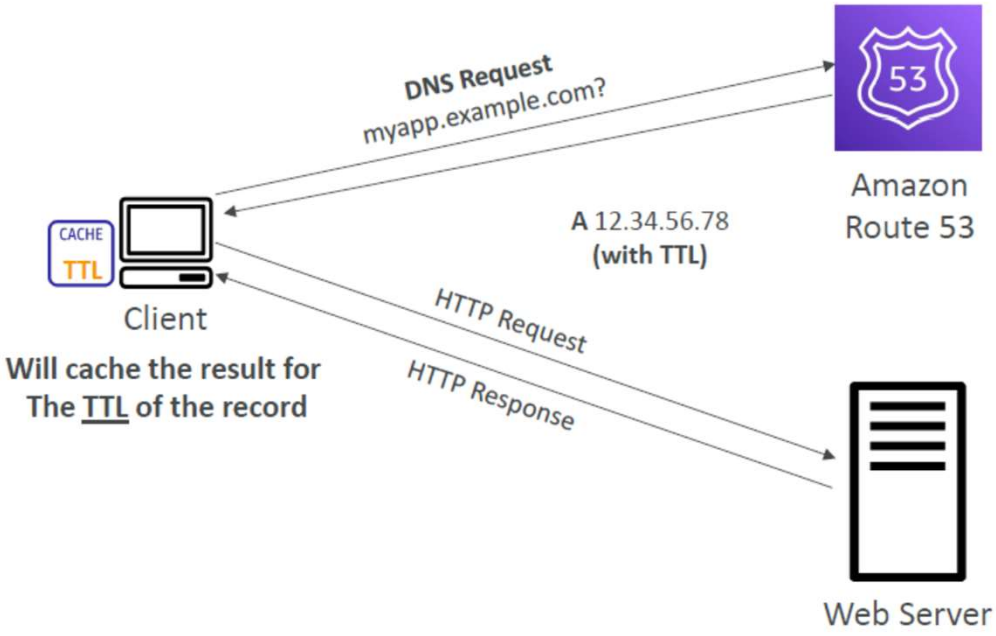
- You pay $0.50 per month per hosted zone

Route 53 – Public vs. Private Hosted Zones



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

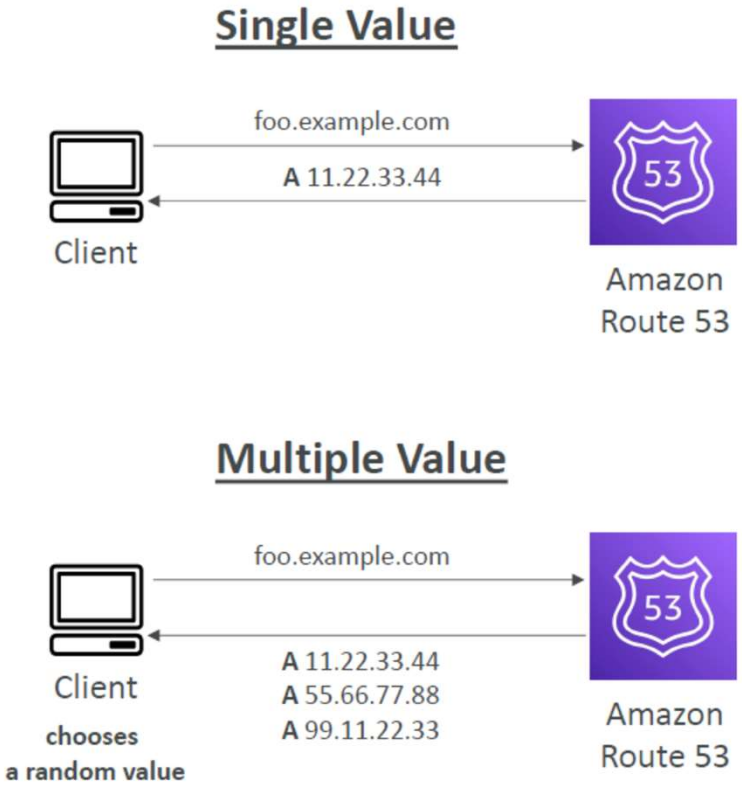


Route 53 – Routing Policies

- 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)

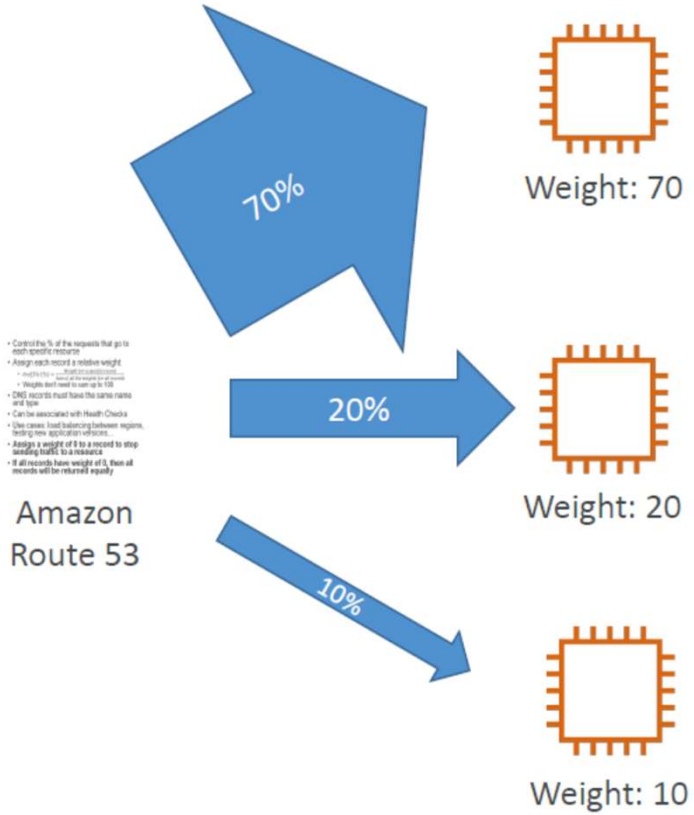
Routing Policies – Simple

- 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



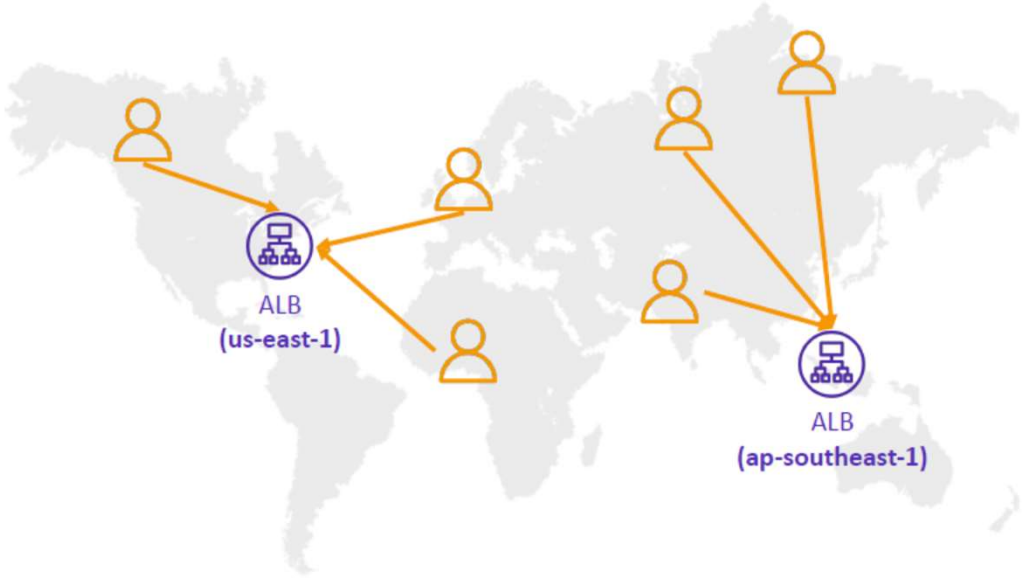
Routing Policies – Weighted

- Control the % of the requests that go to each specific resource
- Assign each record a relative weight:
  - $traffic\ (\%) = \frac{Weight\ for\ a\ specific\ record}{Sum\ of\ all\ the\ weights\ for\ all\ records}$
  - 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



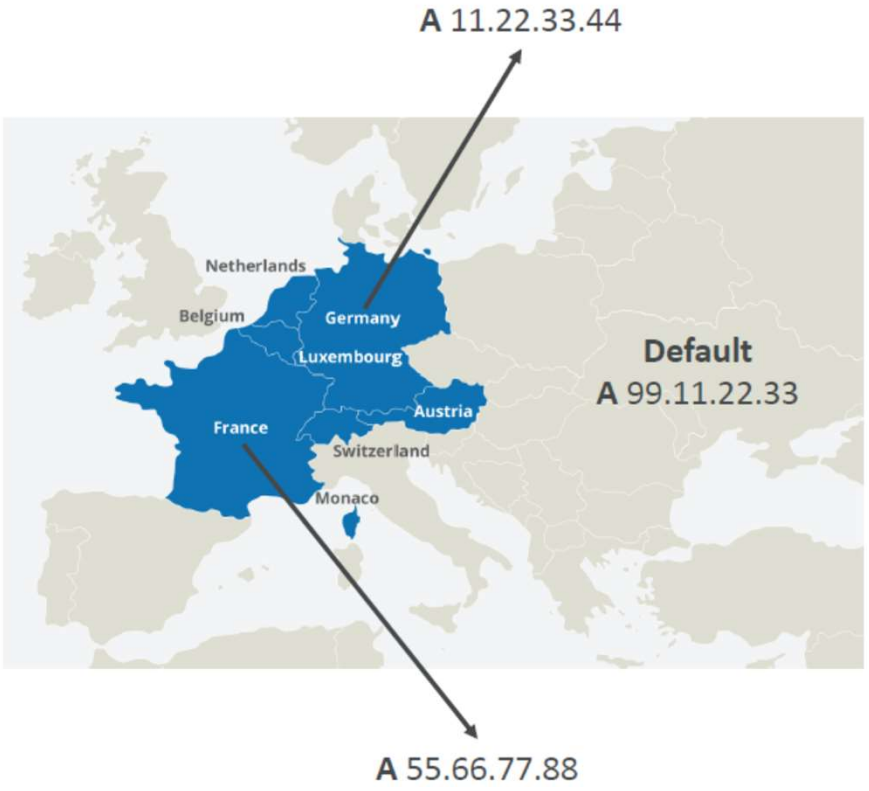
Routing Policies – Latency-based

- 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)



Routing Policies – Geolocation

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

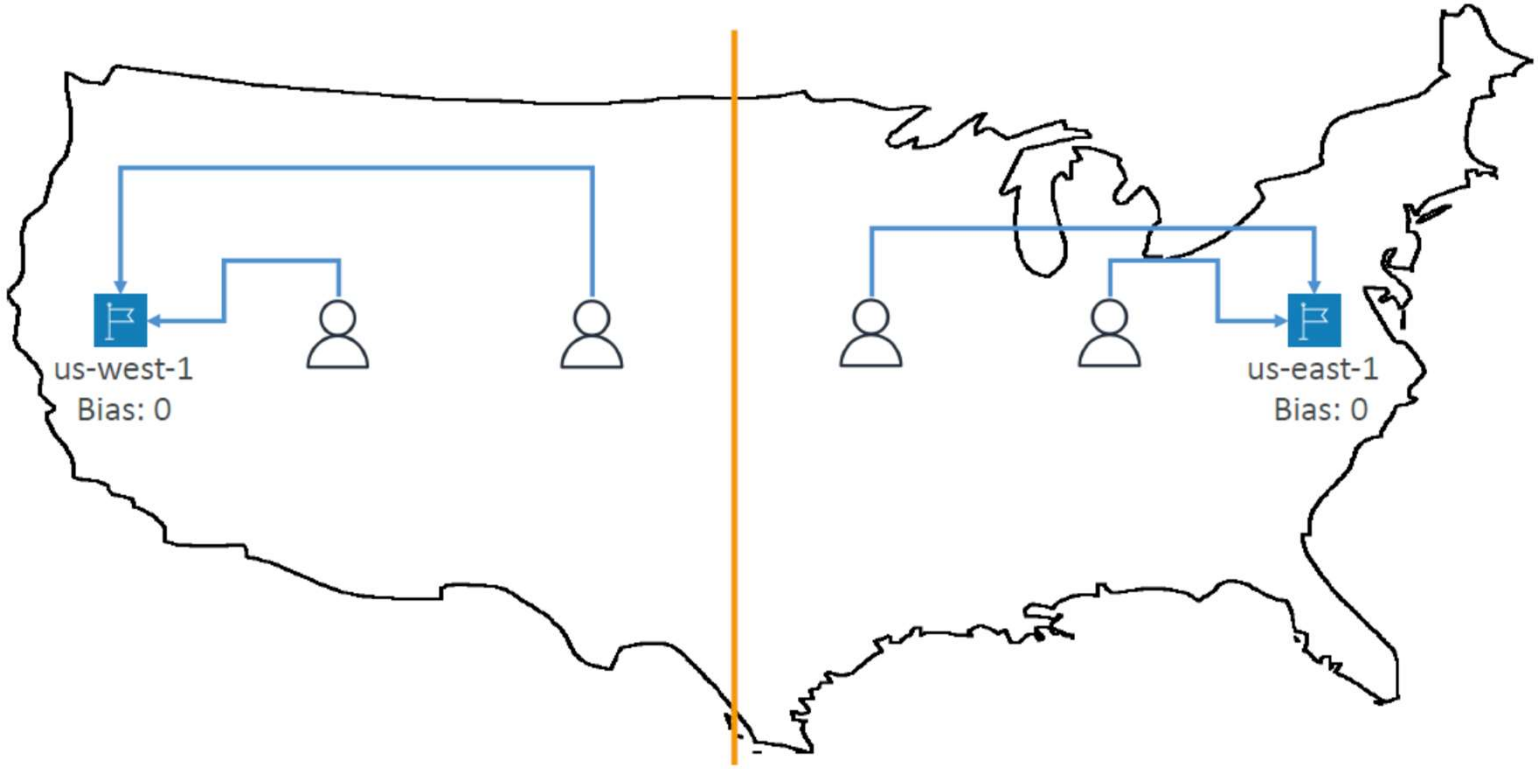


Routing Policies – Geoproximity

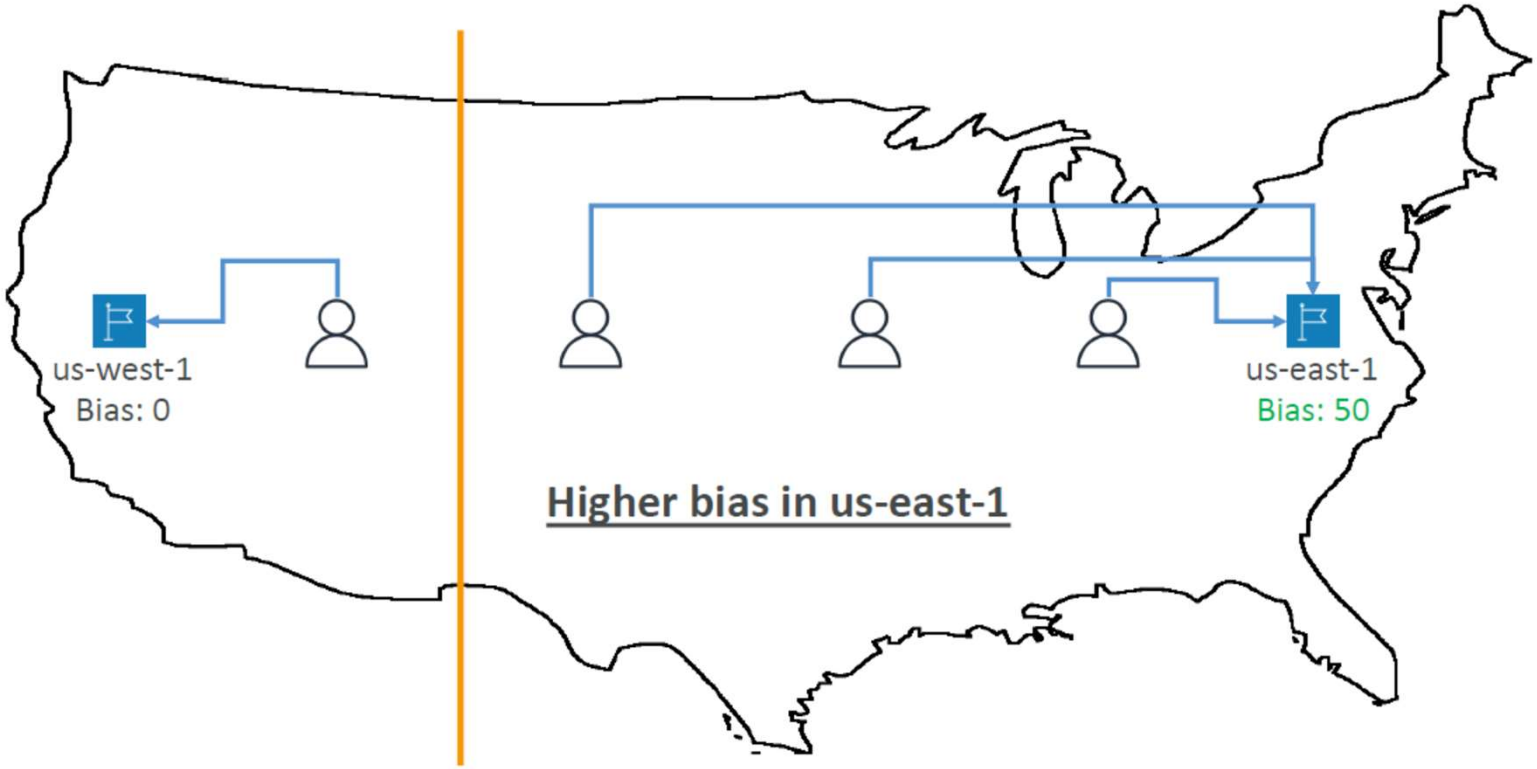
- Route traffic to your resources based on the geographic location of users and resources
- Ability **to shift more traffic to resources based** on the defined **bias**
- To change the size of the geographic region, specify **bias** values:
  - To expand (1 to 99) – more traffic to the resource
  - To shrink (-1 to -99) – less traffic to the resource

- Resources can be:
  - AWS resources (specify AWS region)
  - Non-AWS resources (specify Latitude and Longitude)
- You must use Route 53 **Traffic Flow** to use this feature

Routing Policies – Geoproximity

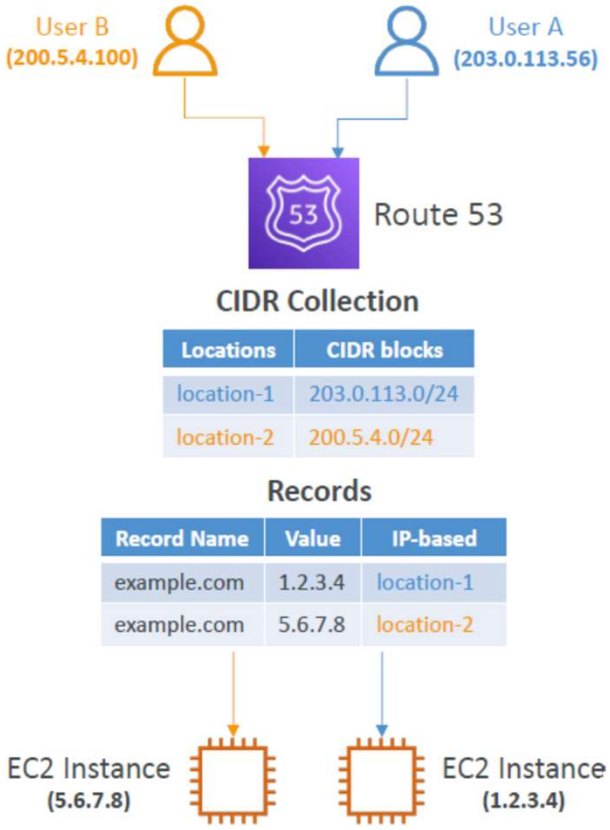


Routing Policies – Geoproximity



Routing Policies – IP-based Routing

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



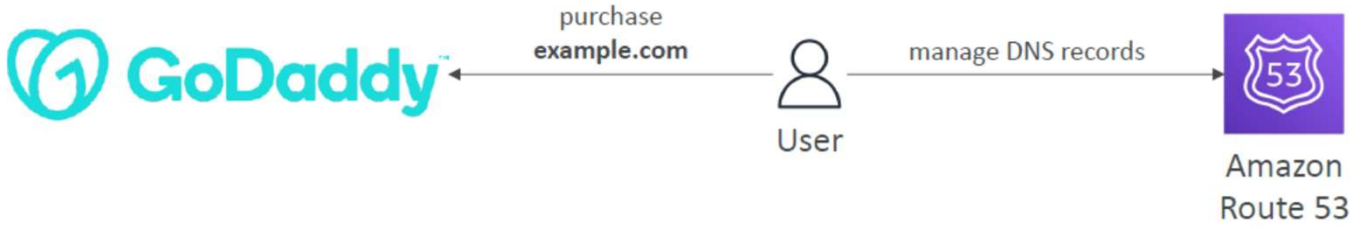
Routing Policies – Multi-Value

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

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

Domain Registar 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



Serverless Computing

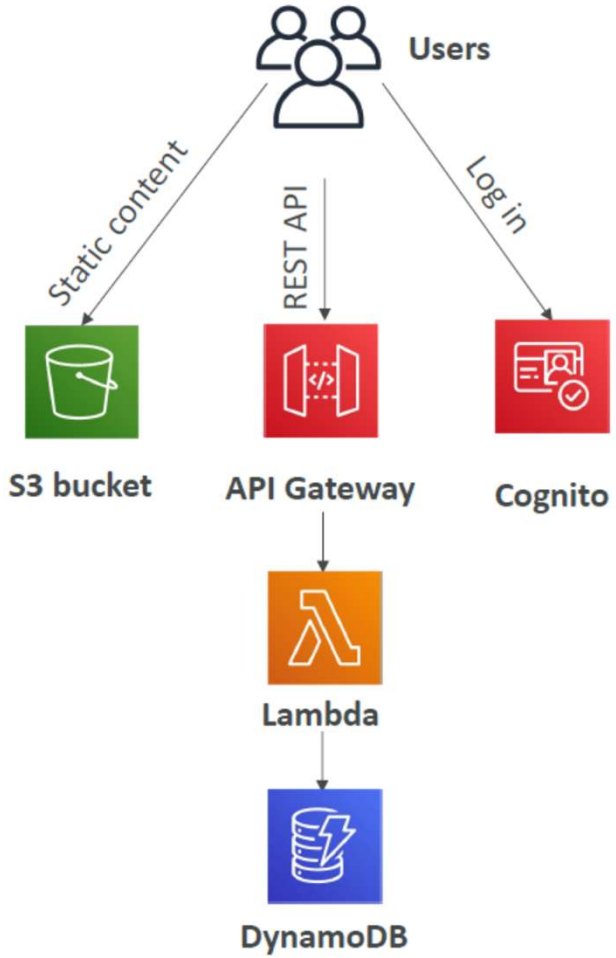
Lambda, API Gateway, SQS, SNS

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

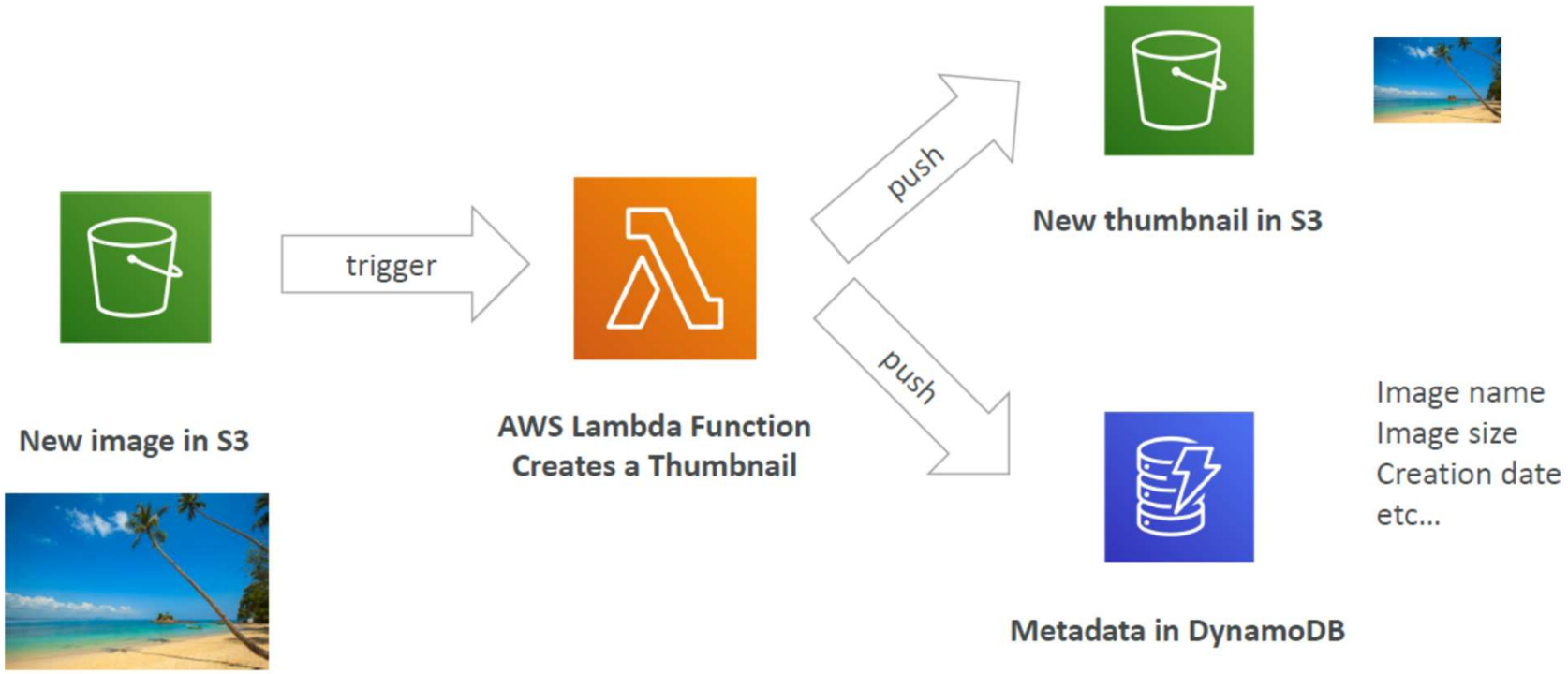
CloudWatch Logs

SNS

SQS

Cognito

Example: Serverless Thumbnail creation



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 1GB 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 to 10GB
  - 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

Example: Building a Serverless API

Integration Services

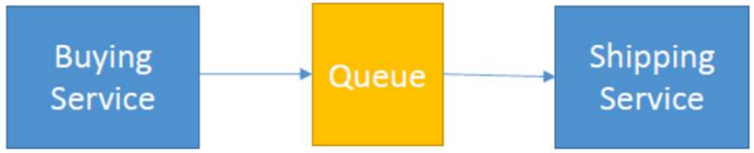
SQS, SNS

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)

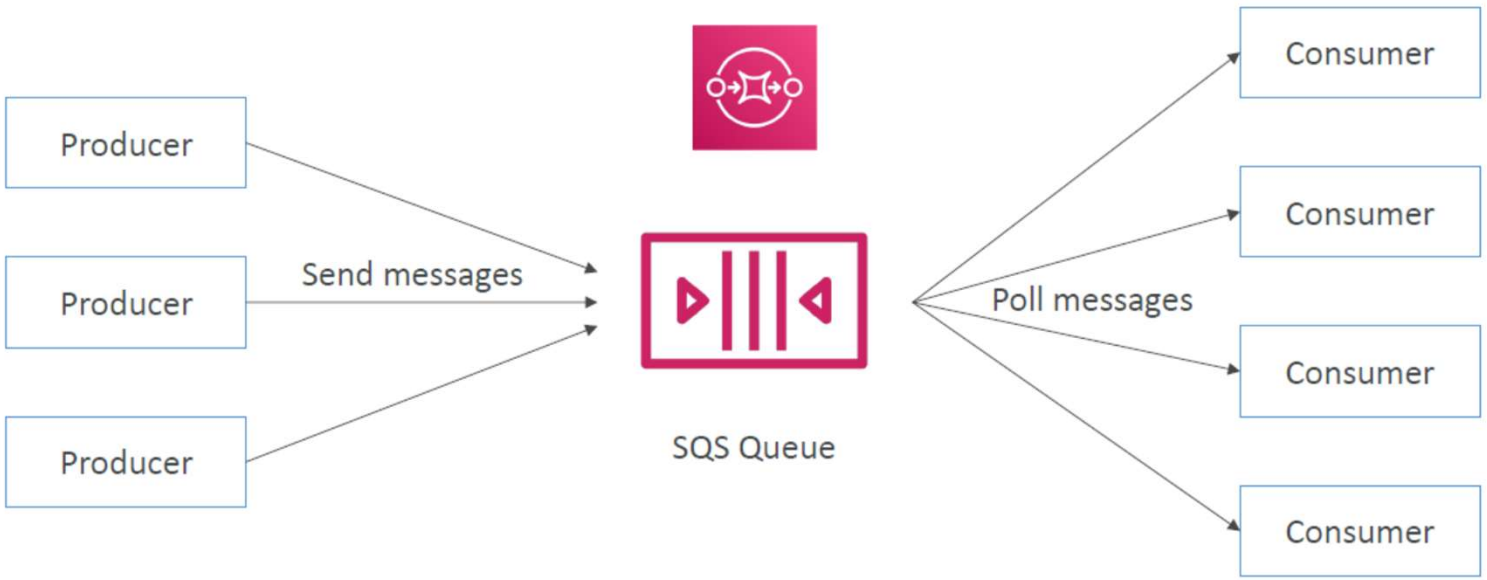


Section Introduction

- 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

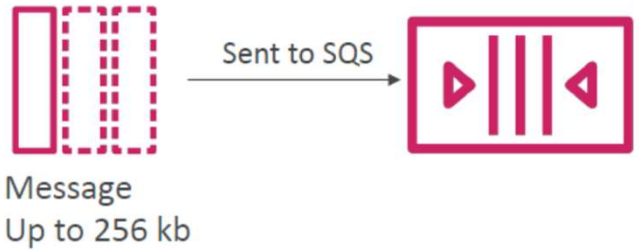
- Can have duplicate messages (at least once delivery, occasionally)
- Can have out of order messages (best effort ordering)

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

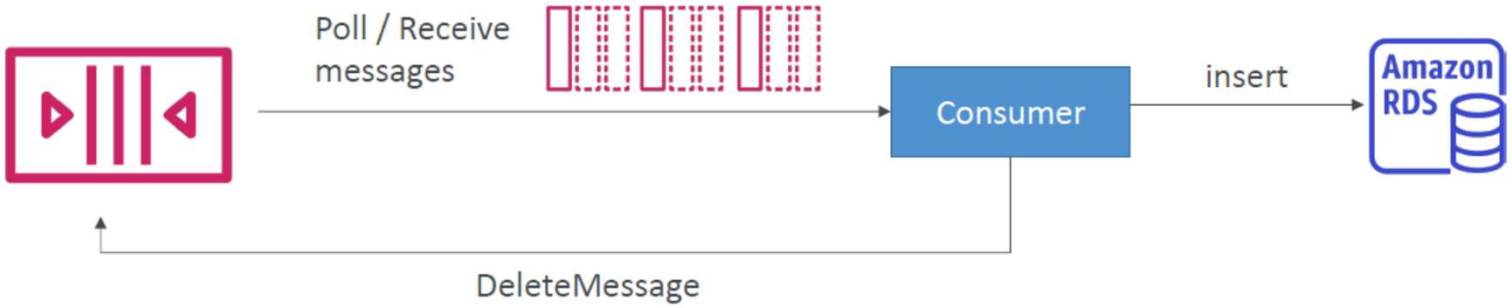
- Example: send an order to be processed
  - Order id
  - Customer id
  - Any attributes you want

- SQS standard: unlimited throughput

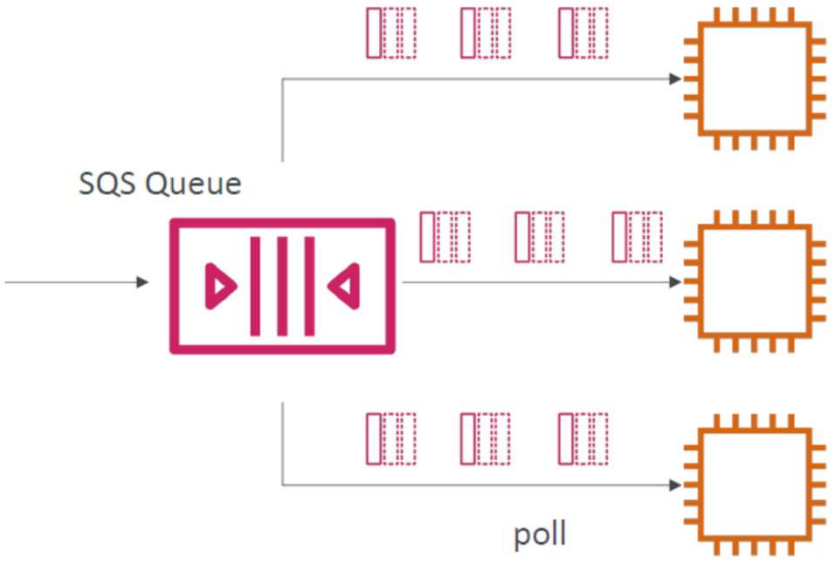


SQS – Consuming Messages

- 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

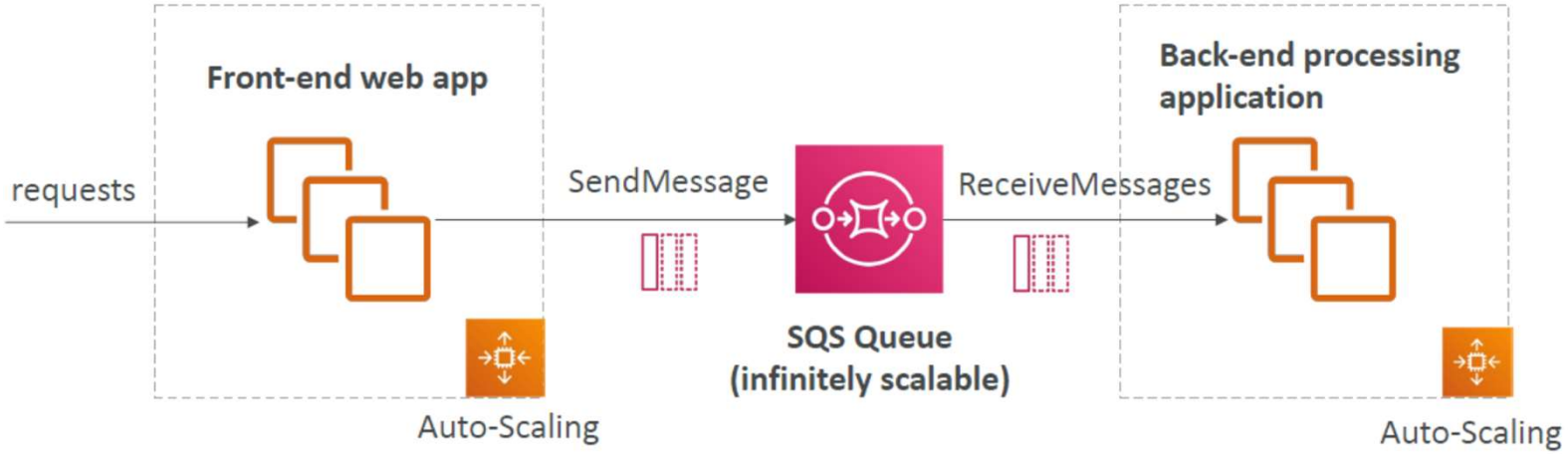


SQS – Multiple EC2 Instances Consumers



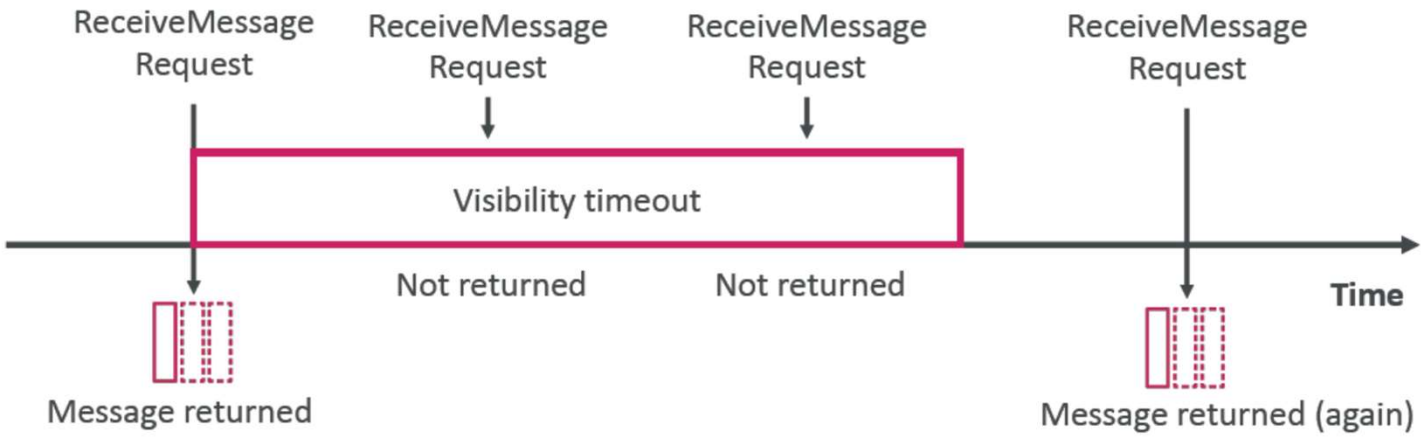
- Consumers receive and process messages in parallel
- At least once delivery
- Best-effort message ordering
- Consumers delete messages after processing them
- We can scale consumers horizontally to improve throughput of processing

SQS to **decouple** between application tiers

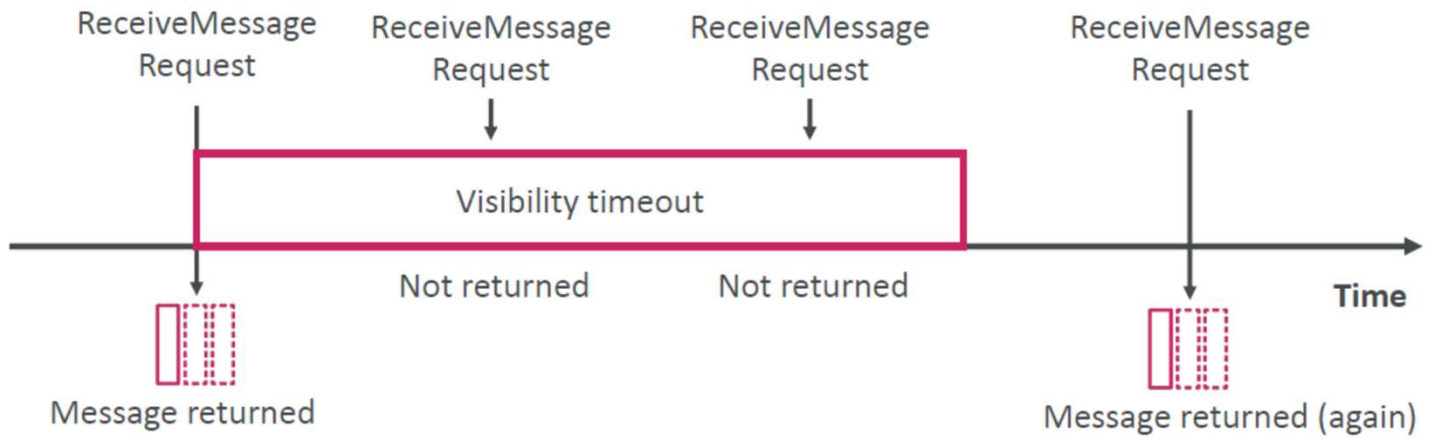


SQS – Message Visibility Timeout

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



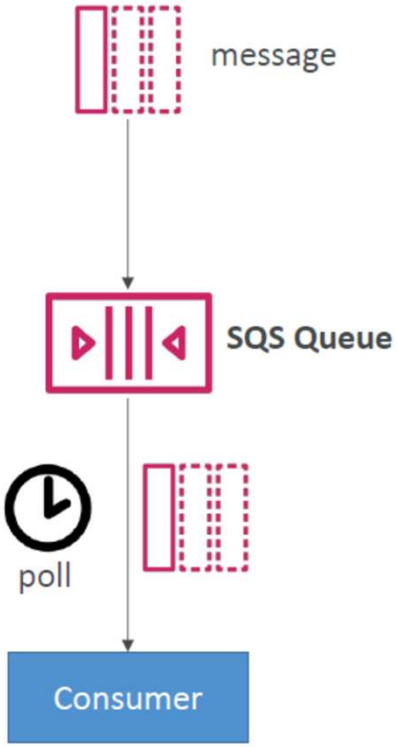
SQS – Message Visibility Timeout



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

Amazon SQS - Long Polling

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



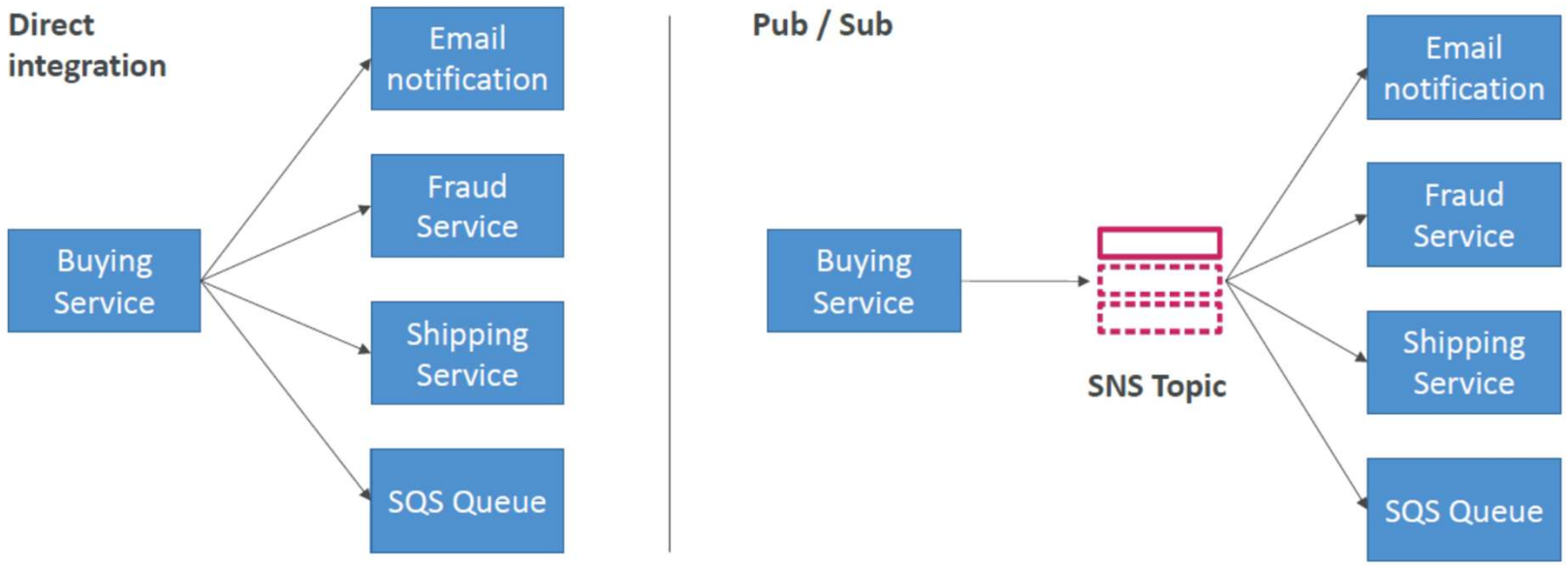
Amazon SQS – FIFO Queue

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

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

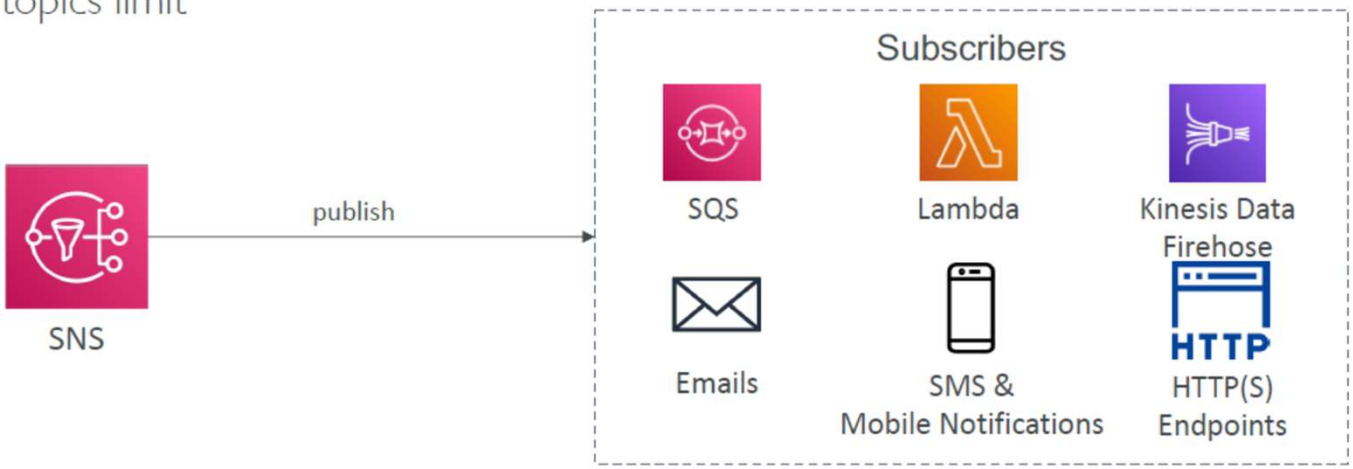
Amazon SNS

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



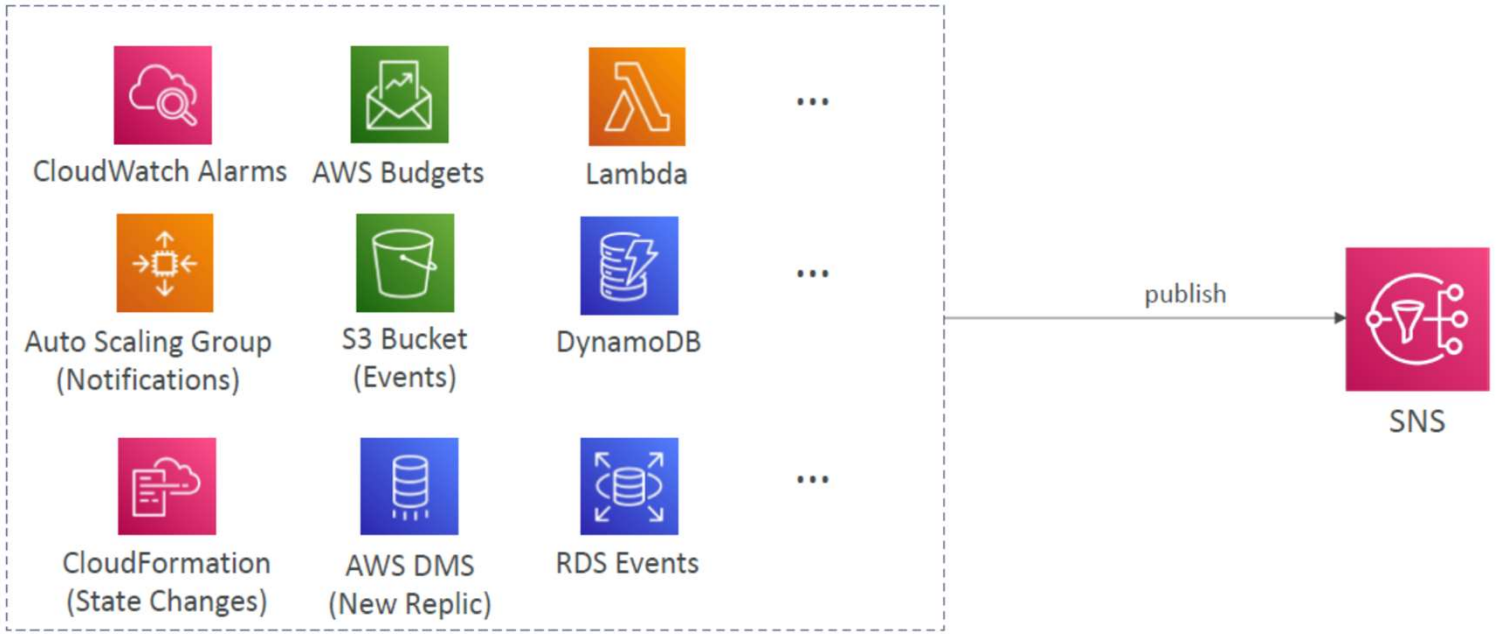
Amazon SNS

- 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…

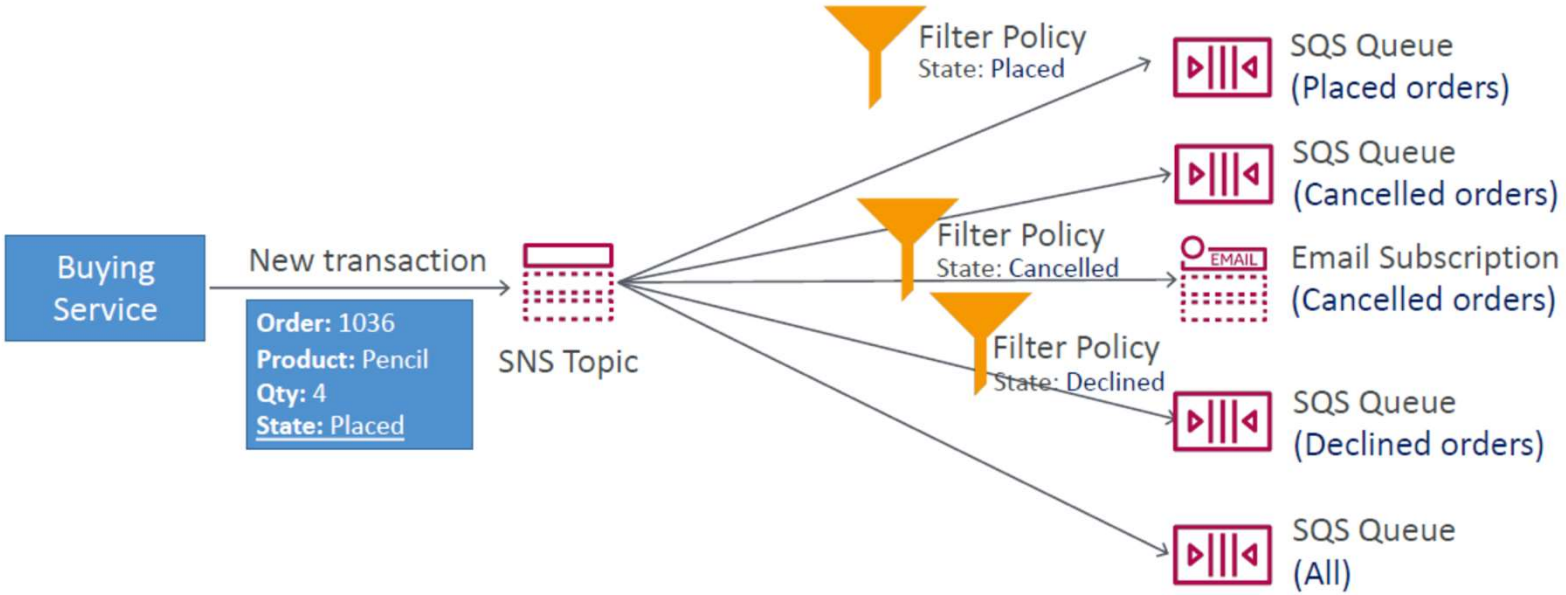
Amazon SNS – FIFO Topic

- 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 – 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



Amazon CloudWatch

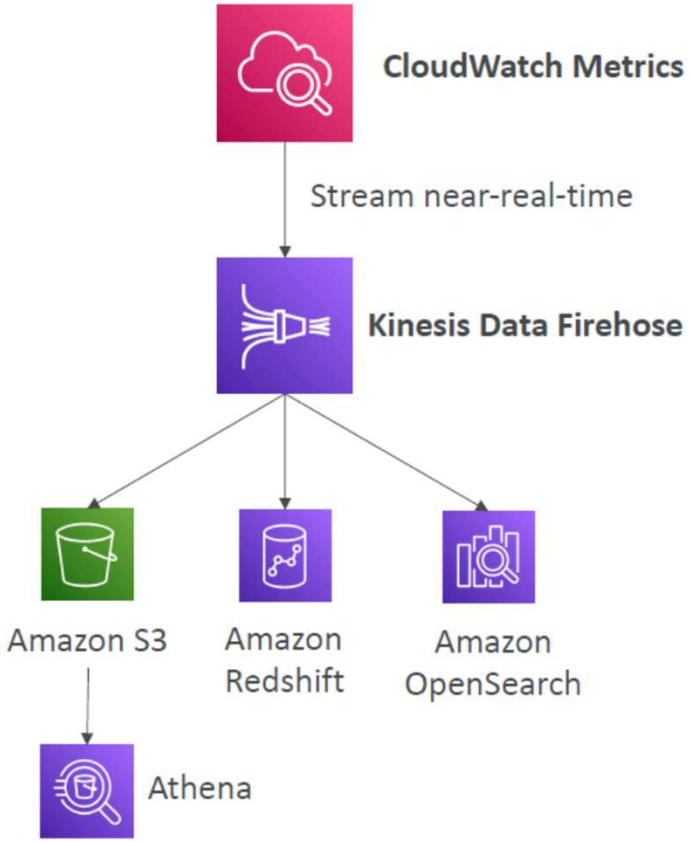
Monitoring

Amazon 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 30 dimensions per metric
- Metrics have **timestamps**
- Can create CloudWatch dashboards of metrics
- Can create **CloudWatch Custom Metrics** (for the RAM for example)

CloudWatch Metric Streams

- Continually stream CloudWatch metrics to a destination of your choice, with **near-real-time delivery** and low latency.
  - Amazon Kinesis Data Firehose (and then its destinations)
  - 3rd party service provider: Datadog, Dynatrace, New Relic, Splunk, Sumo Logic…
- Option to **filter metrics** to only stream a subset of them



CloudWatch Logs

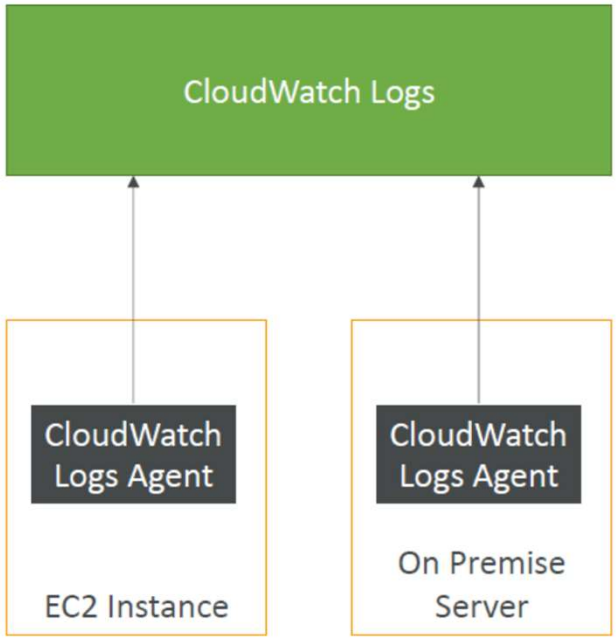
- **Log groups:** arbitrary name, usually representing an application
- **Log stream:** instances within application / log files / containers
- Can define log expiration policies (never expire, 30 days, etc..)
- **CloudWatch Logs can send logs to:**
  - Amazon S3 (exports)
  - Kinesis Data Streams
 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

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 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

Thank you