

SAA-1

AWS Global Infrastructure

- **AWS Regions**

- A Region is a cluster of data centers



-

- 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

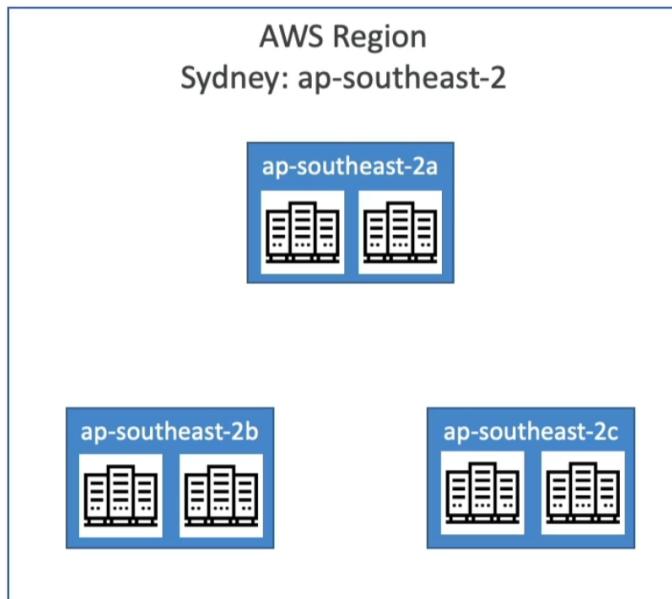
- Middle East (Bahrain) me-south-1

- South America (São Paulo) sa-east-1

-

- AWS Availability Zones**

- Each region has many availability zones(usually 3, min is 2,max is 6)
 - Each availability zone(ZA) is one or more discrete data centers with redundant power,networking, and connectivity
 - They are separate from each other, so that they are isolated from disasters



- **AWS Data Centers**
- **AWS Edge Locations/Points of Presence**
 - Amazon has 216 Points of Presence (205 Edge Locations & || Regional Caches) in 84 cities across 42 countries
 - Content is delivered to end users with lower latency

Console Home

IAM Section

IAM: Users & Groups

- IAM = **Identity and Access Management**, Global service
- **Root account** created by default, shouldn't be used or shared
- **User** 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



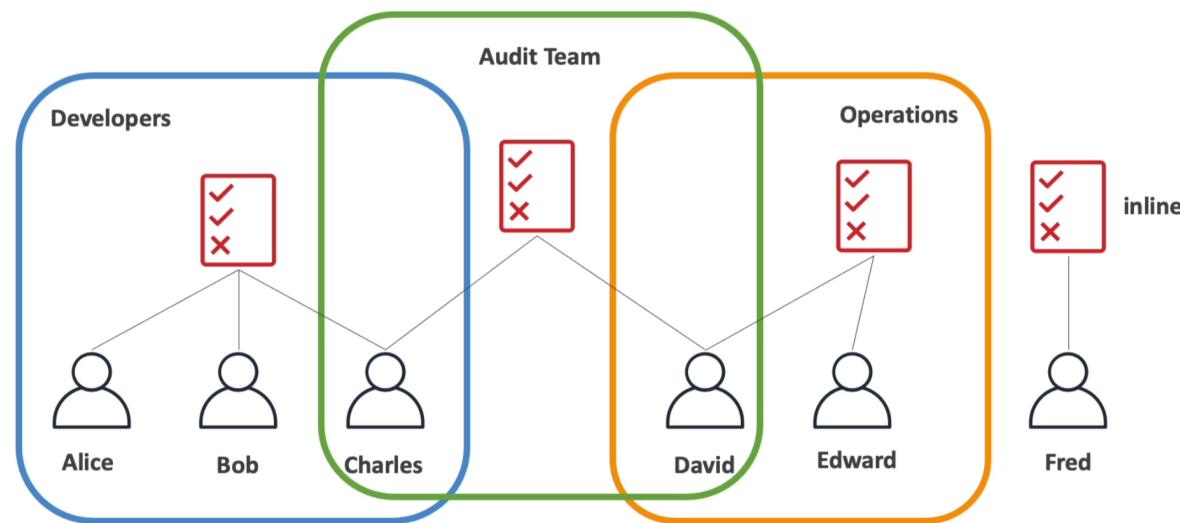
IAM: Permissions

- 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 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
 - Actions: 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

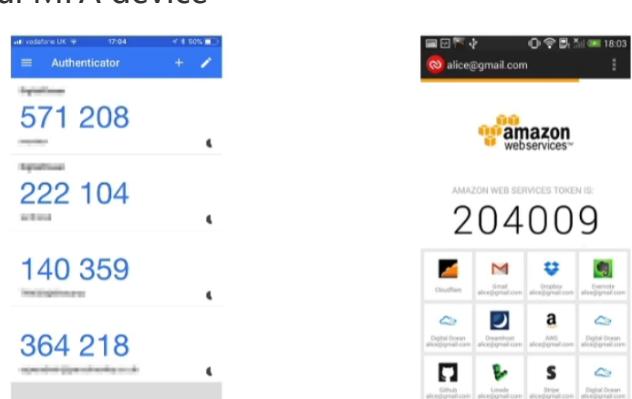
Multi Factor Authentication - MFA

- 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



MFA devices options in AWS

- Virtual MFA device



Google Authenticator
(phone only)

Authy
(multi-device)

Support for multiple tokens on a single device.

- Universal 2nd Factor(U2F) Security Key



YubiKey by Yubico (3rd party)

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

- Hardware Key Fob MFA Device



Provided by Gemalto (3rd party)

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



Provided by SurePassID (3rd party)

How can users access AWS?

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

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
→ ~
```

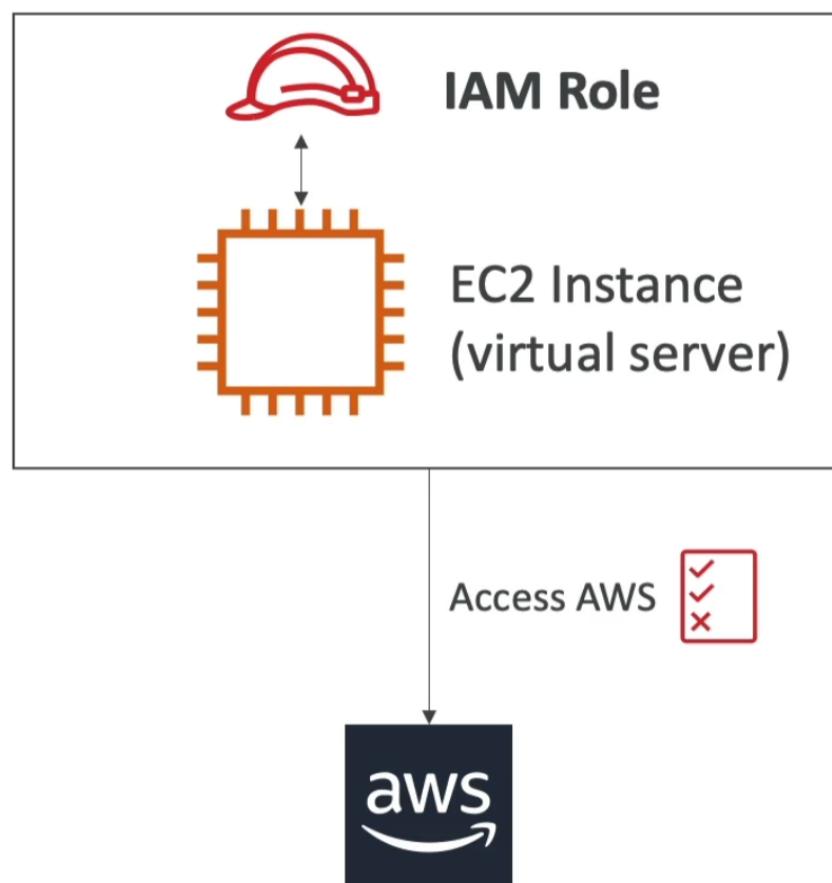
What's the AWS SDK?

- 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 (JavaScripts, 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

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 Role 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

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

AWS Budgets Setup

EC2 Section

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
 - 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

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



EC2 Instance Types - General Purpose

- Great for a diversity of workloads such as web servers or code repositories
- Balance between:
 - Compute
 - Memory
 - Networking

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
-----	-----	----	-----	----	-----	----	-----	-----	------	----	----

-

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
-----	------	----	-----	-----	----

-

EC2 Instance Types - Memory Optimized

- Fast performance for wordloads 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

•

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 application
 - 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

•

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.

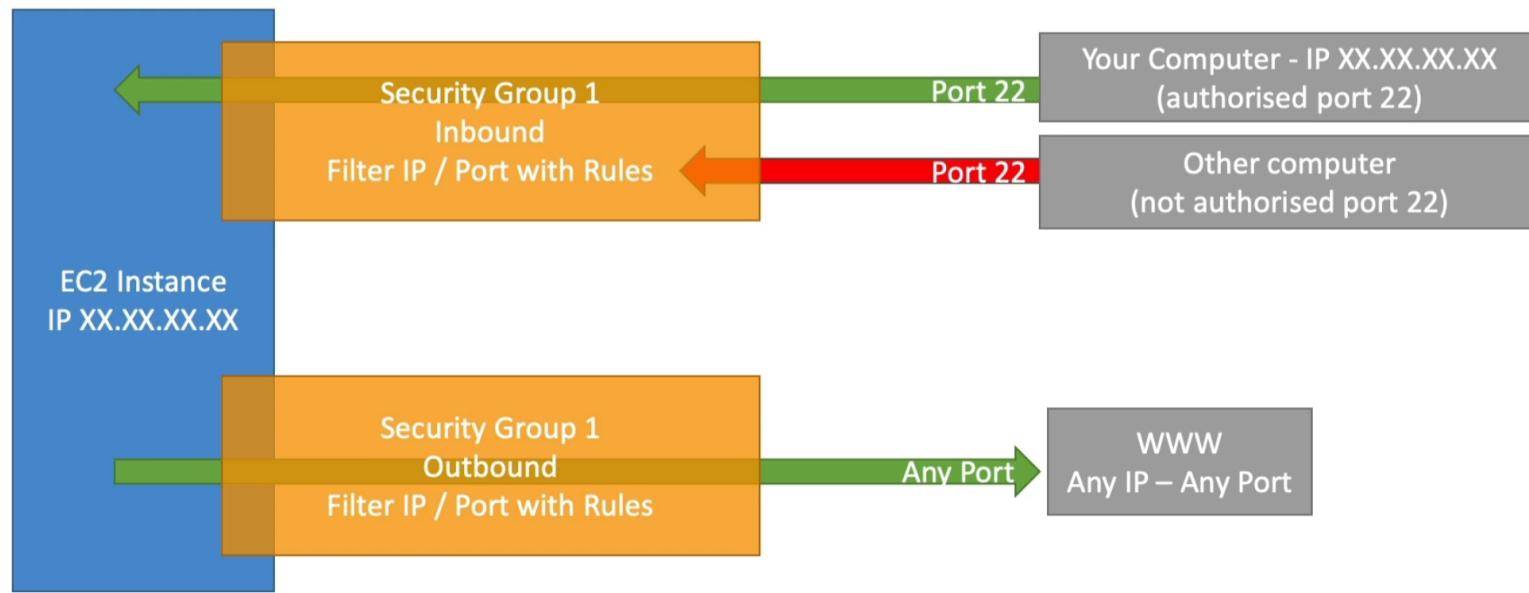


- Security groups only contain allow rules
- Security groups rules can reference by IP or by security group

Security Groups Deeper Dive

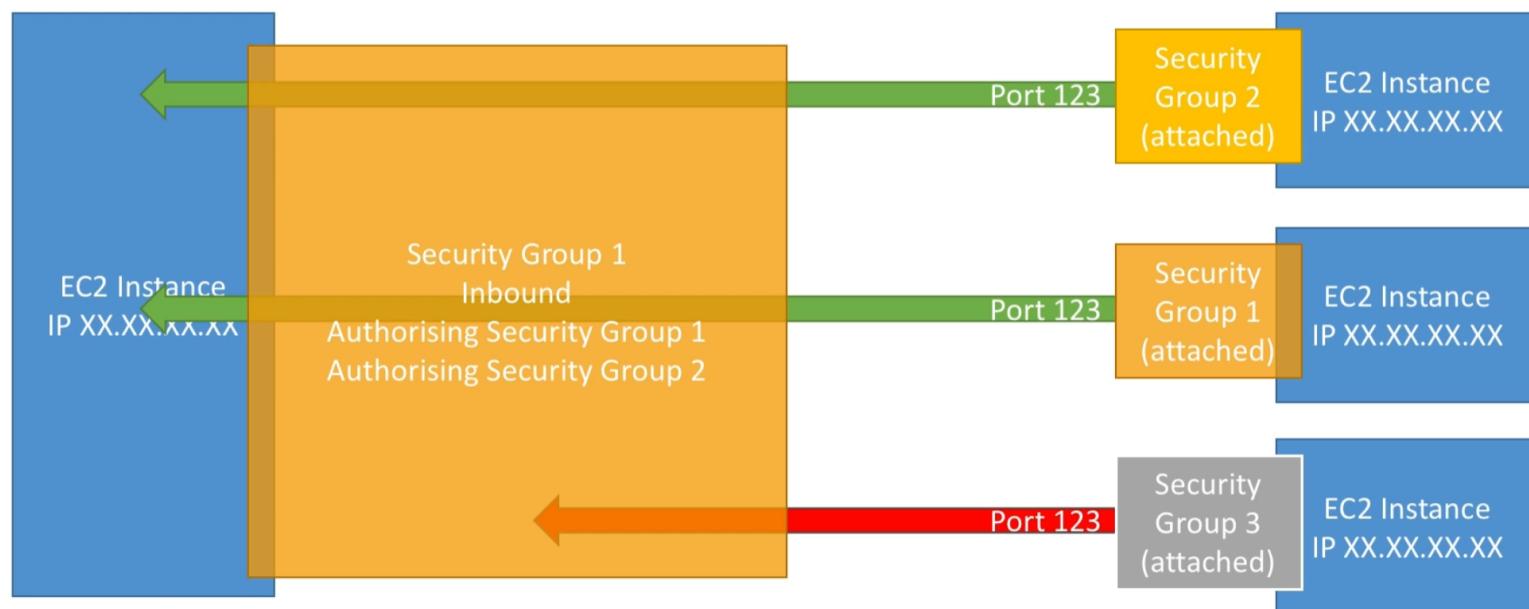
- Security groups are acting as a "firewall" on EC2 instances
- They regulate:
 - Access to Ports
 - Authorised IP ranges - IPv4 and IPv6
 - Control of inbound network (from other to the instance)
 - Control of outbound network (from the instance to other)

Type	Protocol	Port Range	Source	Description
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 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



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	✓	✓	✓

EC2 Instabce Roles

Actions -> Security -> Modify IAM role -> choose a IAM Role

EC2 Instances Purchasing Options

- On-Demand Instances - short workload, predictable pricing, pay by second
 - 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
- Reserved (1 & 3 years)
 - Reserved Instances - long workloads
 - 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 Instances - long workloads with flexible instances
 - Can change the EC2 instance type, instance family, OS, scope and tenancy
 - Up to 66% discount
- Savings Plans (1 & 3 years) - commitment to an amount of usage, long workload
 - 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 Saving 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)
- Spot Instance - short workloads, cheap, can lose Instances (less reliable)
 - 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**
- Dedicated Hosts - book an entire physical server, control instance placement
 - 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, per-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
- Dedicated Instances - no other customers will share your hardware
 - Instances run on hardware that's dedicated to you
 - May share hardware with other instance 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

- Capacity Reservations - reserve capacity in a specific AZ for any duration
 - 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 Saving Plans to benefit from billing discounts
 - You're charged at On-Demand rate whether you run instance or not
 - Suitable for short-term, uninterrupted workloads that needs to be in a specific AZ

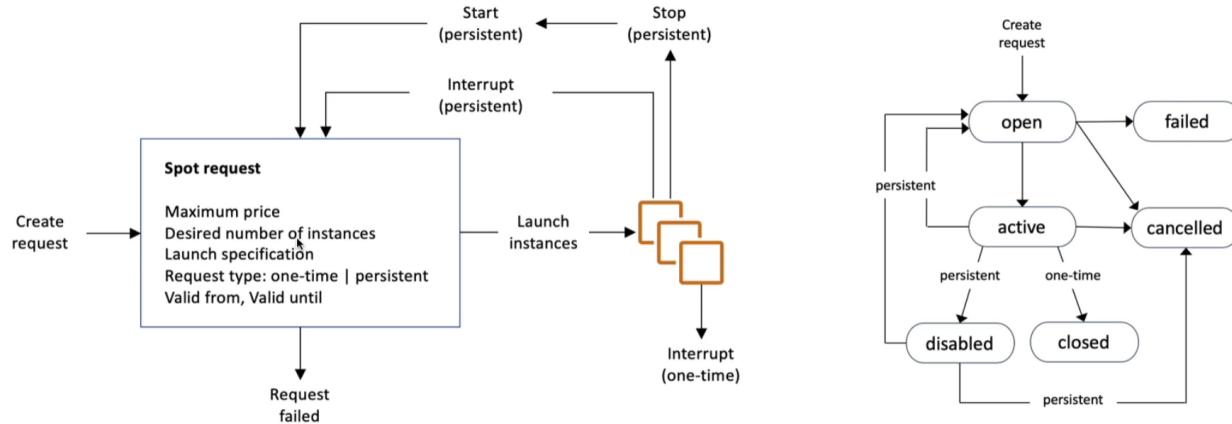
Which purchasing option is right for me? Using resort for an example

- **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 Host:** 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

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**

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

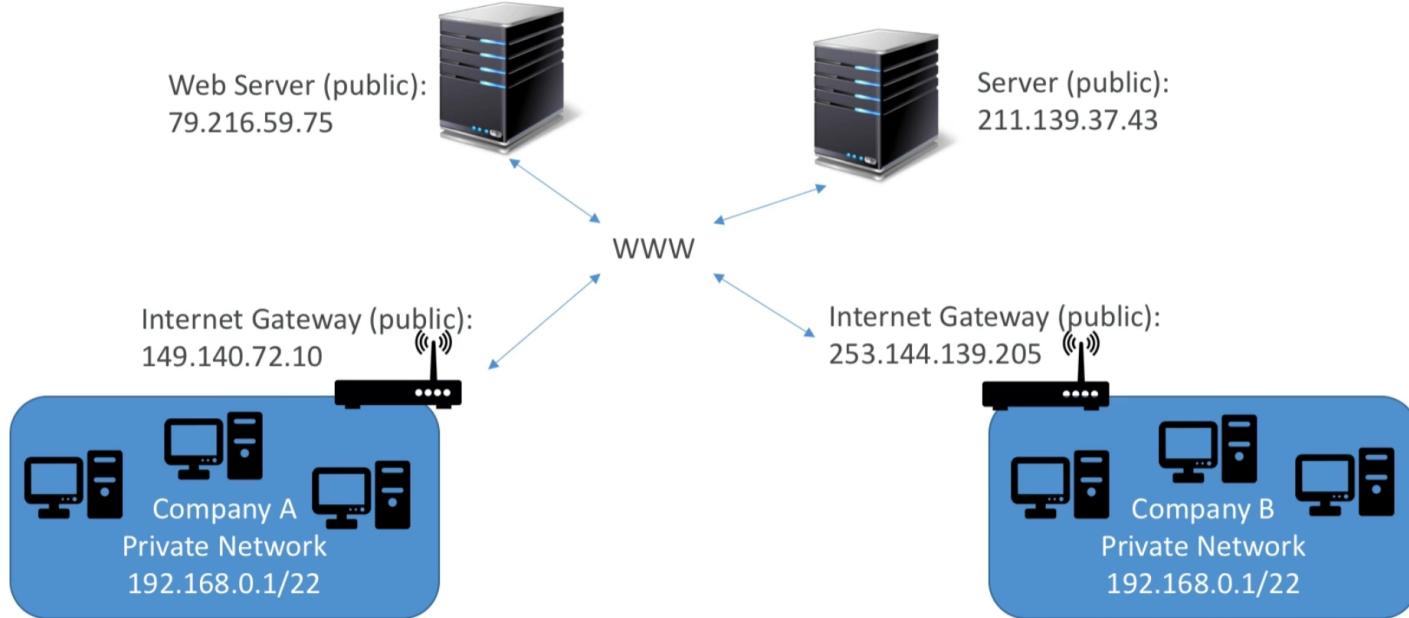
Spot Fleets

- 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 optimal capacity for the number of instances
- Spot Fleets allow us to automatically request Spot Instance with the lowest price

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
- 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) 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 an internet gateway (a proxy)
 - Only a specified range of IPs can be used as private IP

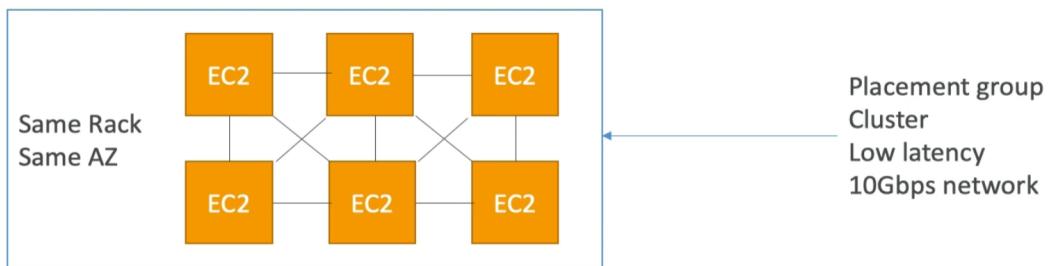
Elastic IPs

- 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
- 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

Placement Groups

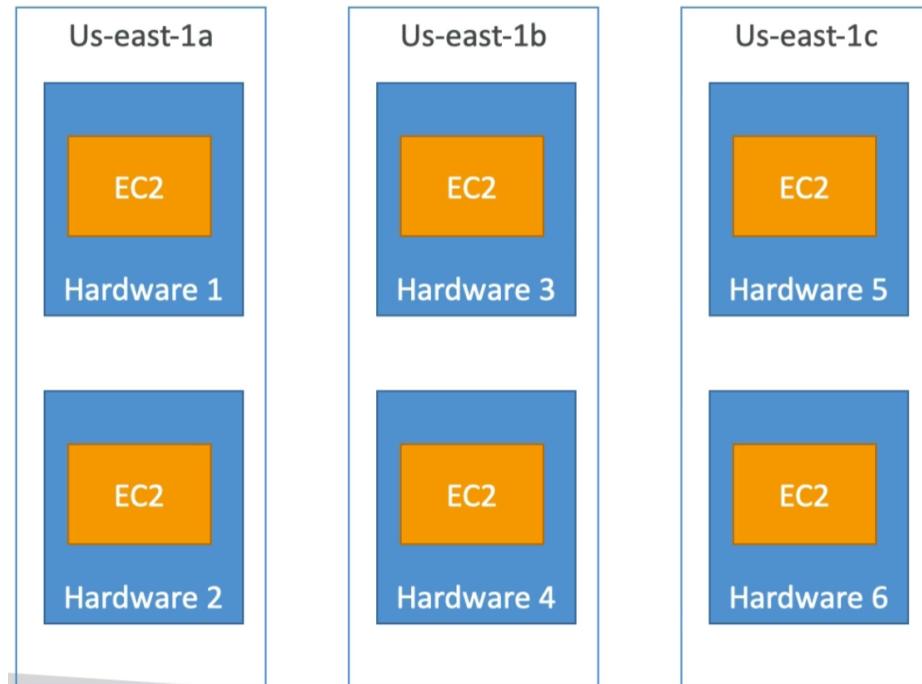
- Sometimes you want control over the EC2 Instance placement strategy
- That strategy can be defined using placement groups
- When you create a placement group, you specify one of the following strategies for the group:
 - Cluster - clusters instances into a low-latency group in a single Availability Zone
 - Spread - spreads instances across underlying hardware (max 7 instances per group per AZ) - critical applications
 - 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



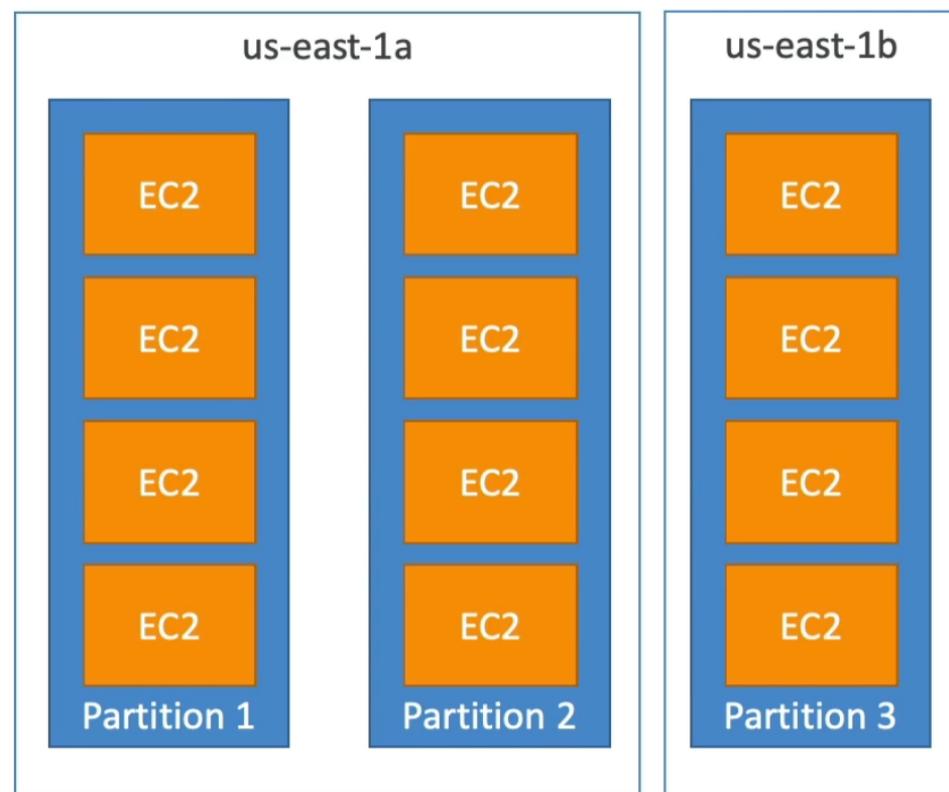
- Pros: Great network (10 Gbps bandwidth between instances)
- Cons: If the rack fails, all instances fail at the same time
- Use case:
 - Big Data job that needs to complete fast
 - Application that needs extremely low latency and high network throughput

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

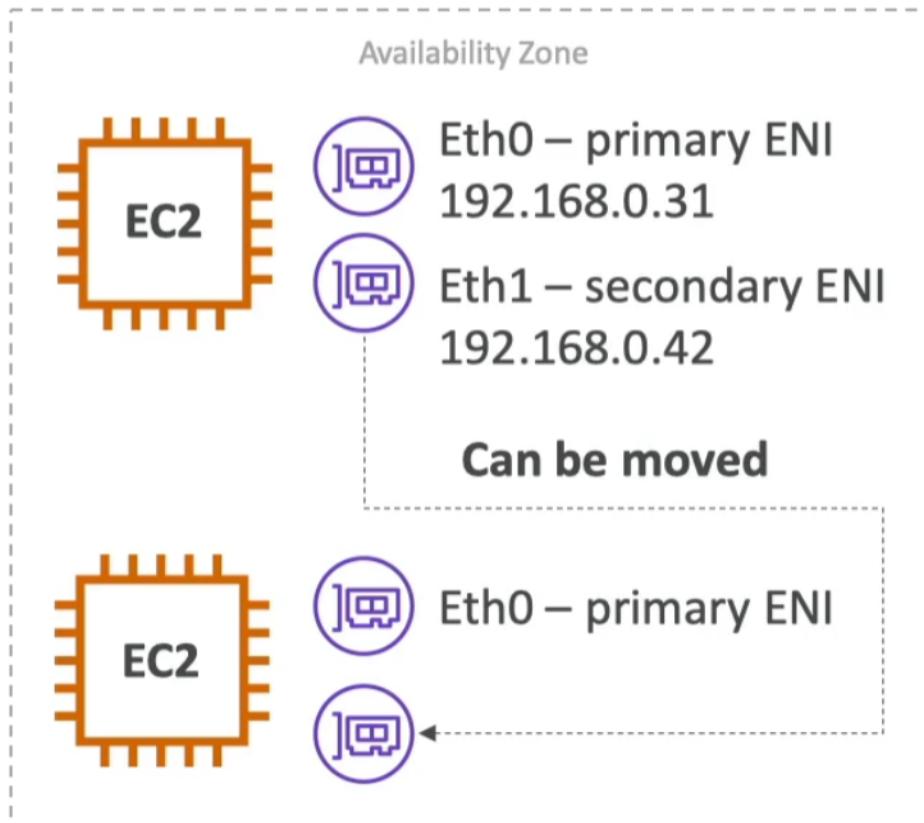
Placements Groups Partition



- 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)

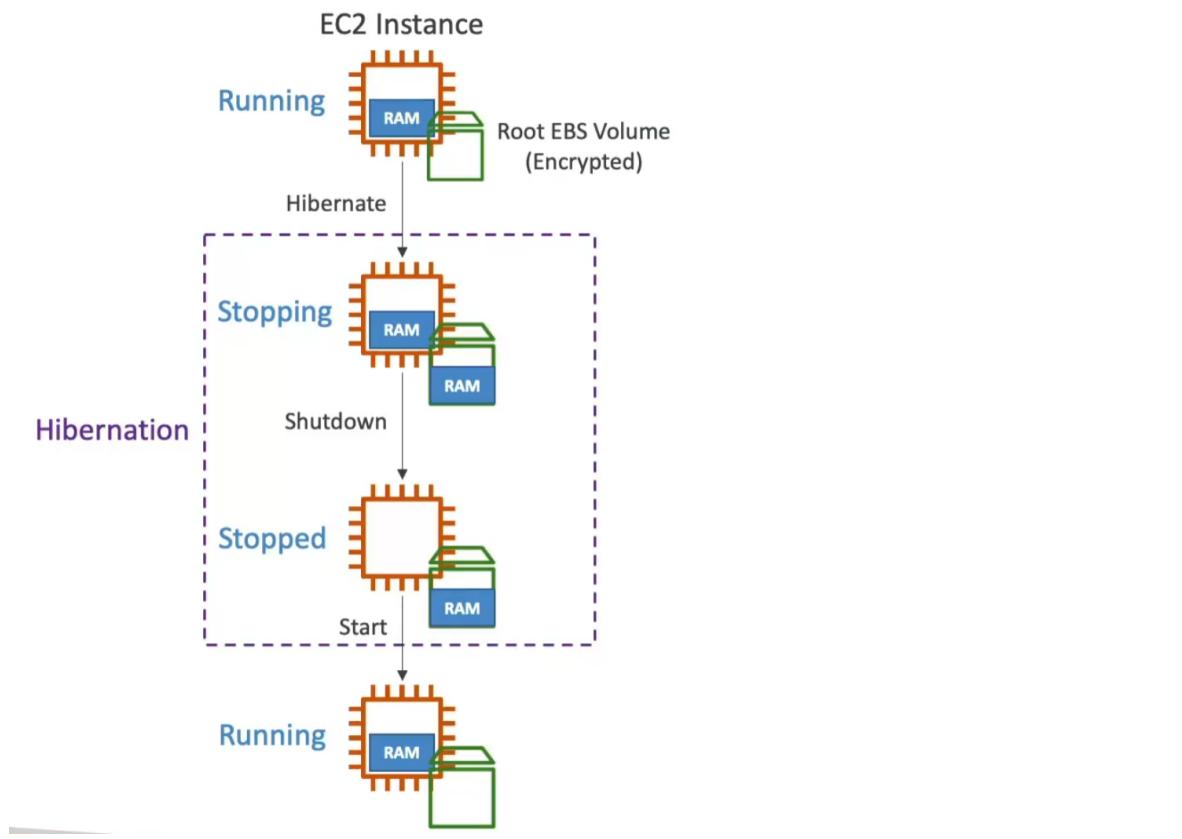


- Logical component in a VPC that represents a virtual network card
- The ENI can have the following attributes:
 - Primary private IPv4, one or more secondary IPv4
 - One Elastic IP ((IPv4) per private IPv4
 - One Public IPv4
 - One or more security groups
 - A MAC address
- You can create ENI independently 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 start: the OS boots up
 - Then your application starts, caches get warmed up, and that can take time!

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

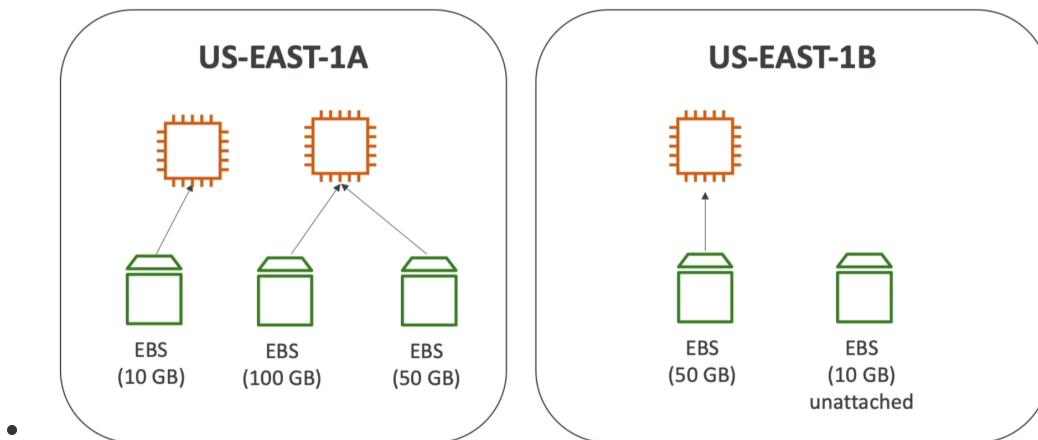
EC2 Instance Storage Section

what's an EBS Volume?

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

EBS Volume

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



EBS - Delete on Termination attribute

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)
Root	/dev/xvda	snap-09f18f682fd23a1b1	8	General Purpose SSD (gp2)	100 / 3000	N/A
EBS	/dev/sdb	Search (case-insensit)	8	General Purpose SSD (gp2)	100 / 3000	N/A
Add New Volume						

A green box highlights the "Delete on Termination" checkbox, which is checked by default.

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

EBS Snapshots

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



EBS Snapshots Features

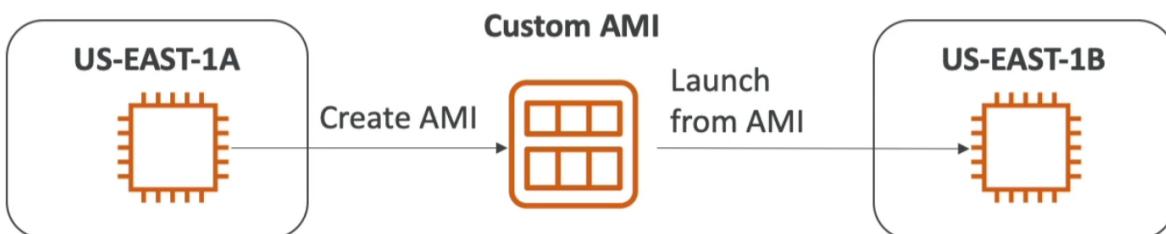
- EBS Snapshot Archive**
 - Move a Snapshot to an "archive tier" that is 75% cheaper
 - Takes within 24 to 72 hours for restoring the archive
- Recycle Bin for EBS Snapshots**
 - Setup rules to retain deleted snapshots so you can recover them after an accidental deletion
 - Specify retention (from 1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

AMI

- 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

EBS volume Types

- EBS Volumes come in 6 types
 - [gp2 / gp3 \(SSD\)](#): General purpose SSD volume that balance 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 - 16TiB
- gp3:
 - Baseline of 3000 IOPS and throughput of 125 MiB/s
 - Can increase IOPS up to 16000 and throughput up to 1000 MiB/s independently
- gp2:
 - Small gp2 volumes can burst IOPS to 3000
 - Size of the volume and IOPS are linked, max IOPS is 16000
 - 3 IOPS per GB, means at 5334 GB we are at the max IOPS

Provisioned IOPS (PIOPS) SSD

- Critical business applications with sustained IOPS performance
- Or applications that need more than 16000 IOPS
- Great for **databases workloads** (sensitive to storage perf and consistency)
- io1 / io2 (4 GiB - 16 TiB):
 - Max PIOPS:64000 for Nitro EC2 instances & 32000 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

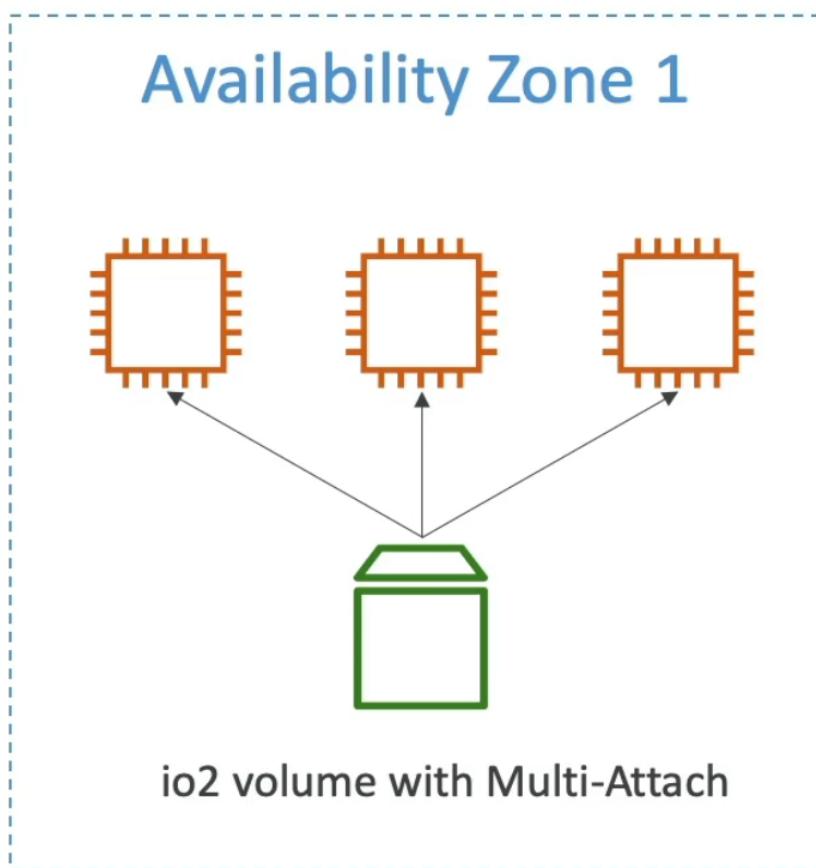
- Max PIOPS: 256000 with an IOPS:GiB ratio of 1000:1
- Supports EBS Multi-attach

Hard Disk Drives (HDD)

- Cannot be a boot volume
- 125 MiB to 16 TiB
- Throughput Optimized HDD (stl)
 - Big Data, Data Warehousing, Log Processing
 - **Max throughput** 500 MiB/s - max IOPS 500
- Cold HDD (scl):
 - 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, EX4, etc...)



EBS Encryption

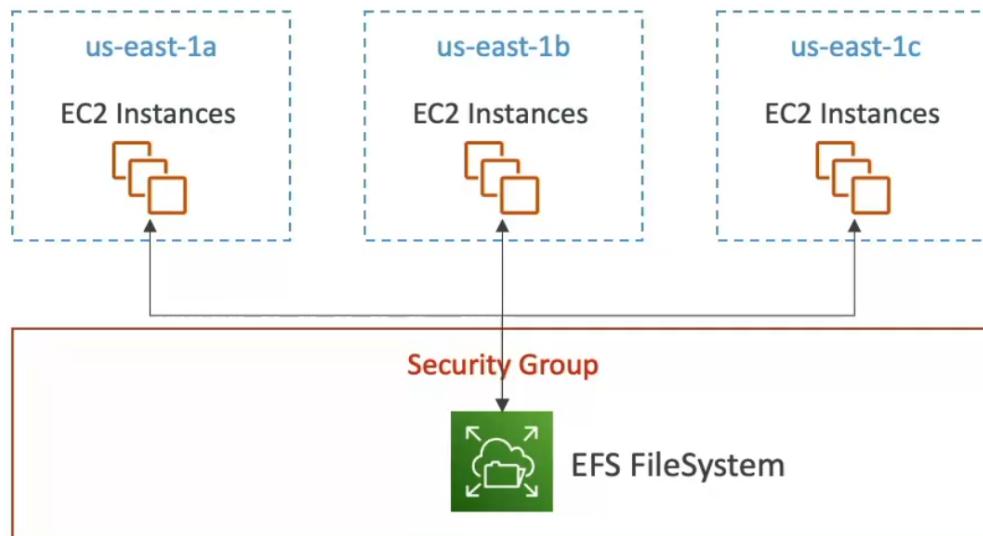
- When you create an encrypted EBS volume, you get the following:
 - Data at rest is encrypted inside the volume
 - All the data in flight moving between the instance and the volume is encrypted
 - All snapshots are encrypted
 - All volumes created from the snapshot
- Encryption and decryption are handled 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
- EFS works with EC2 instances in multi-AZ
- Highly available, scalable, expensive (3x gp2), pay per use
- **Use cases:** content management, web serving, data sharing, Wordpress
- Use 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!

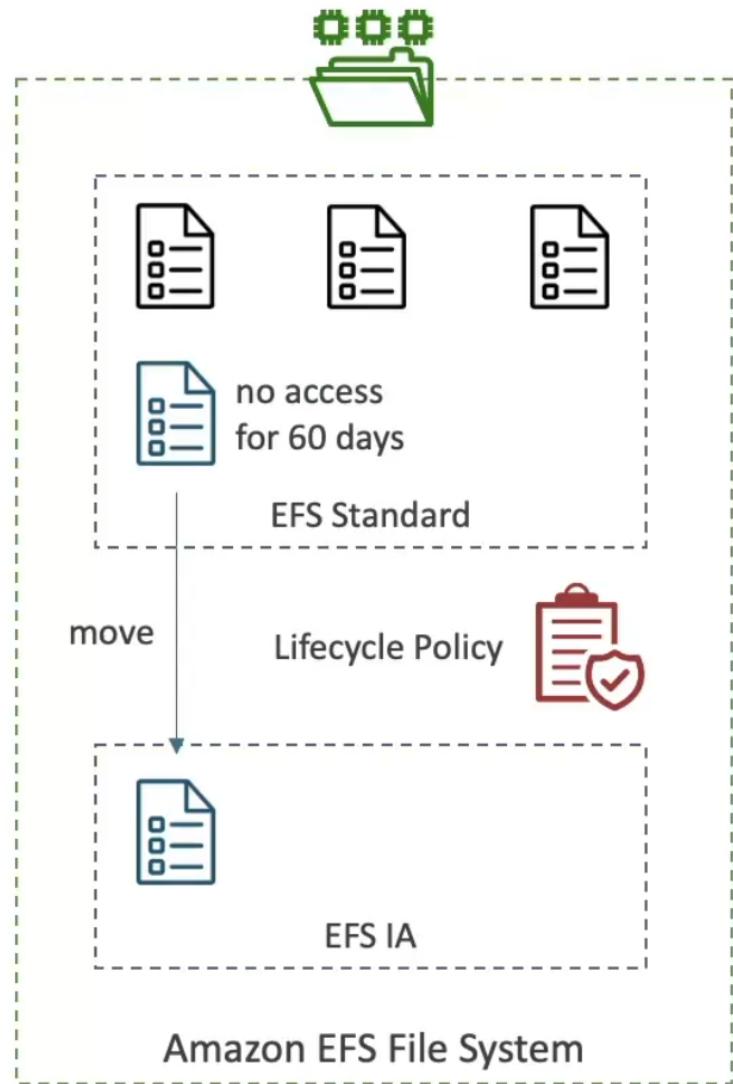
EFS - Performance & Storage Classes

- 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

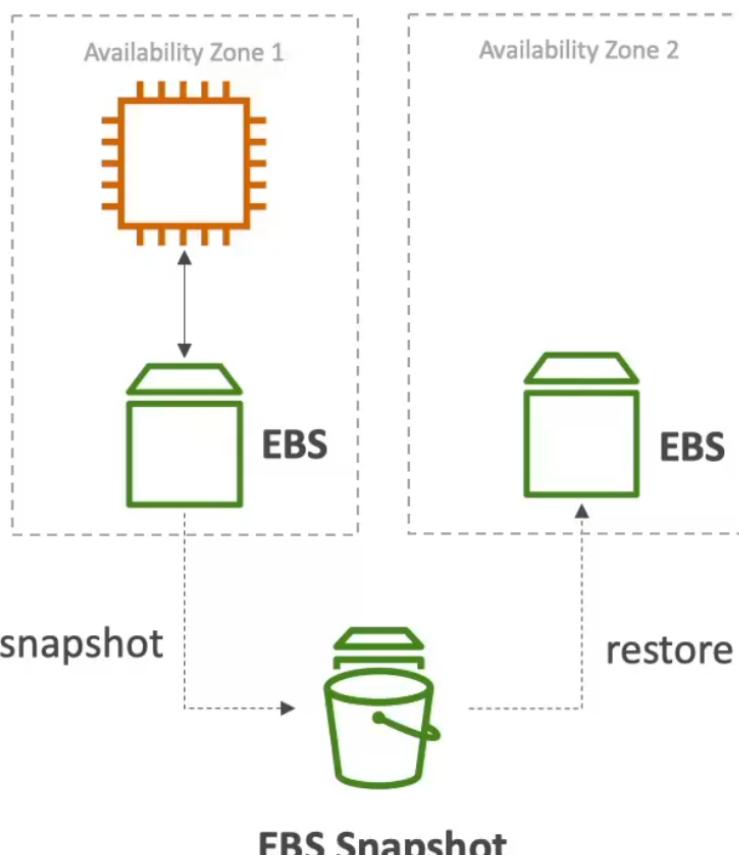
- Storage Tiers (lifecycle management feature - move file after N days)
 - Standard: for frequently accessed files
 - Infrequent access (EFS-IA): cost to retrieve files, lower price to store. Enable EFS-IA with a Lifecycle Policy
- Availability and durability
 - Standard: Multi-AZ, great for prod
 -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

- Over 90% in cost savings



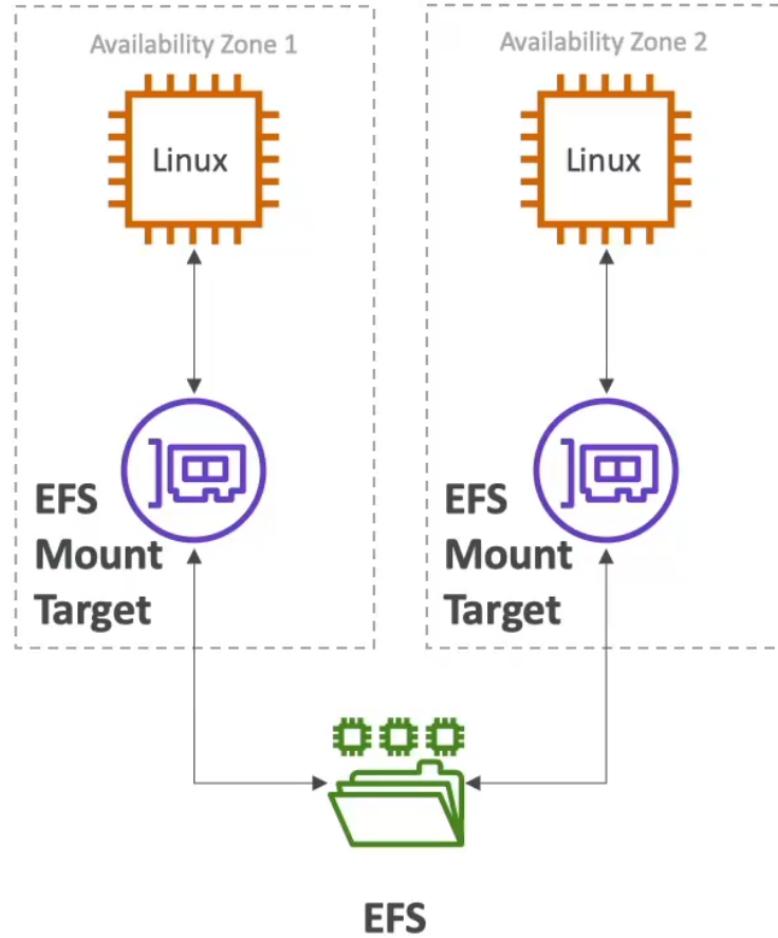
EBS vs EFS

- 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)



- EFS

- 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



Scalability & High Availability

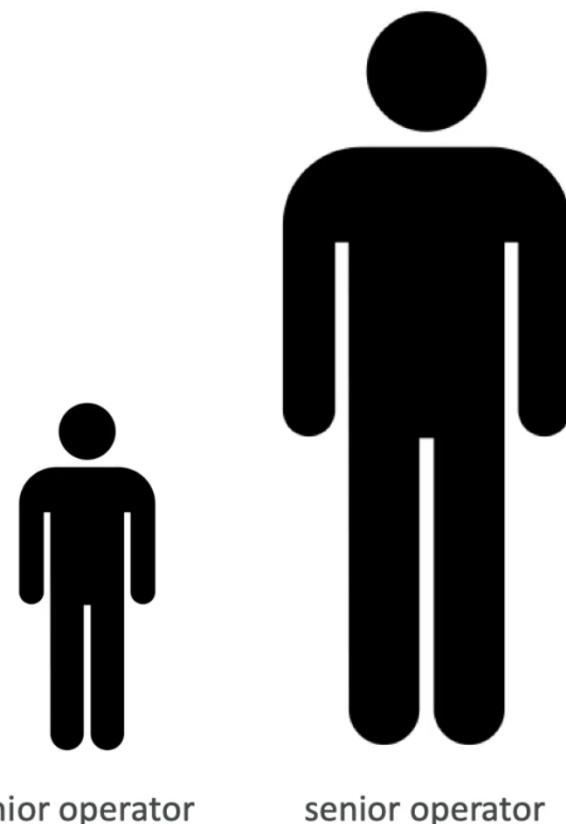
- Scalability means that an application / system can handle greater loads by adapting
- There are two kinds of scalability:
- Vertical Scalability
- Horizontal Scalability (= elasticity)
- **Scalability is linked but different to High Availability**

Let's deep dive into the distinction, using a call center as an example

Vertical Scalability

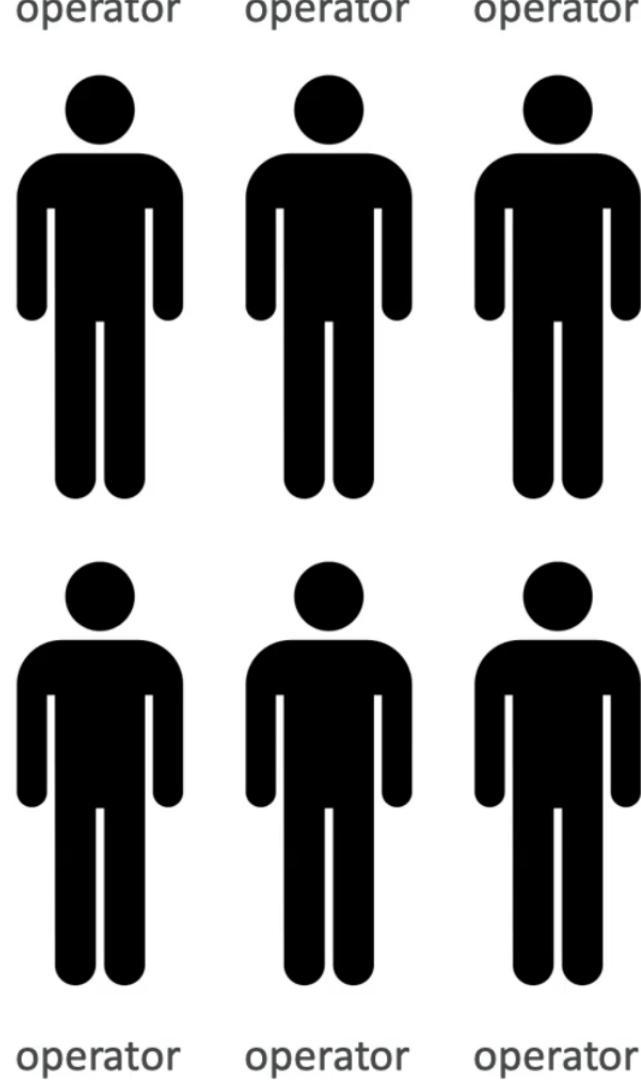
- Vertically scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database
- RDS, ElastiCache are services that can scale vertically

- There's usually a limit to how much you can vertically scale (hardware limit)



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)

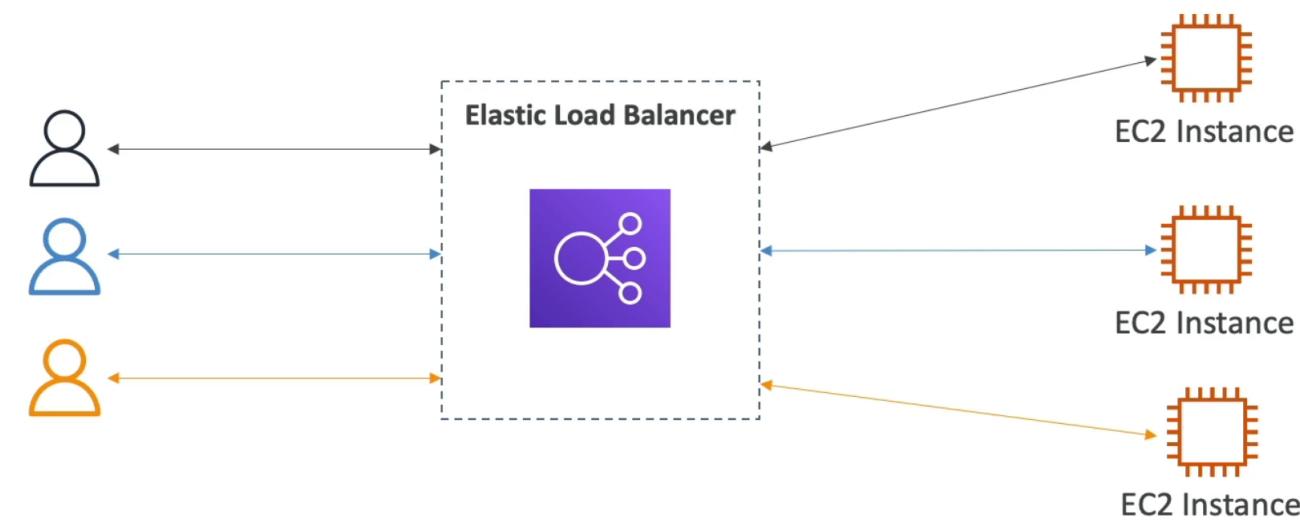


High Availability & Scalability For EC2

- Vertical Scaling: Increase instance size (= scale up / down)
 - From:t2.nano - 0.5G of RAM, 1 vCPU
 - To: u-1 2tb 1.metal - 12.3TB 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 Balancers 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

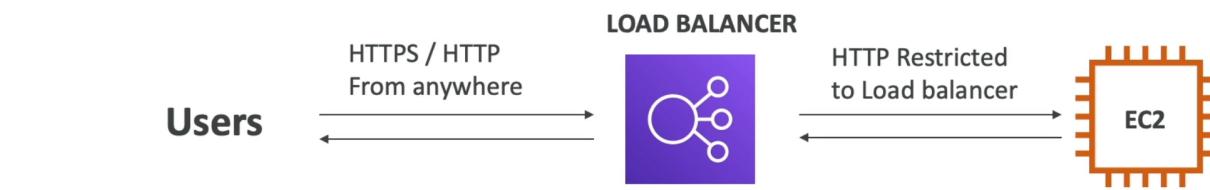
AWS has 4 kinds of managed Load Balancers

- Classic Load Balancer (v1 - old generation) - 2009 - CLB
 - HTTP, HTTPS, TCP, SSL(secure TCP)
 - Supports TCP (Layer 4), HTTP & HTTPS (Layer 7)
 - Fixed hostname
 - XXX.region.elb.amazonaws.com



- Client
- CLB
- EC2
- Application Load Balancer (v2 - new generation) - 2016 - ALB
 - HTTP, HTTPS, WebSocket
- Network Load Balancer (v2 - new generation) - 2017 - NLB
 - TCPTLS (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:

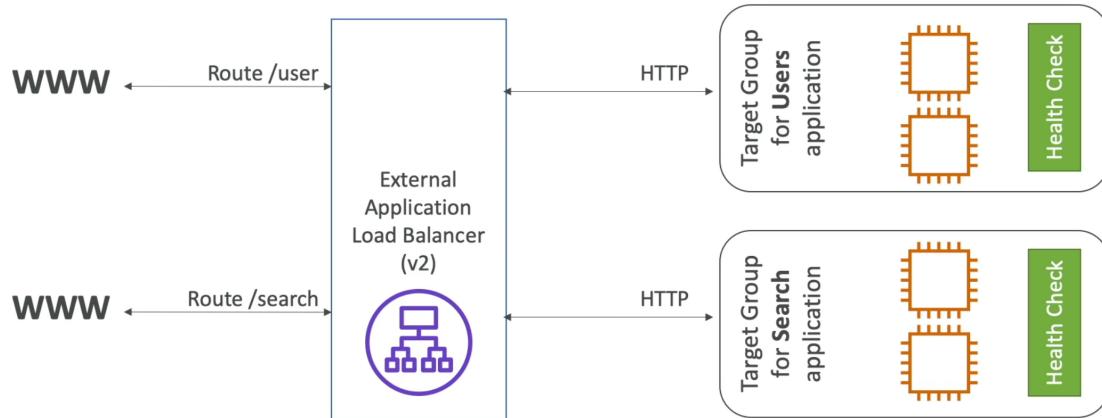
Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Allow HTTP from an...
HTTPS	TCP	443	0.0.0.0/0	Allow HTTPS from a...

Application Security Group: Allow traffic only from Load Balancer

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

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)
- 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)



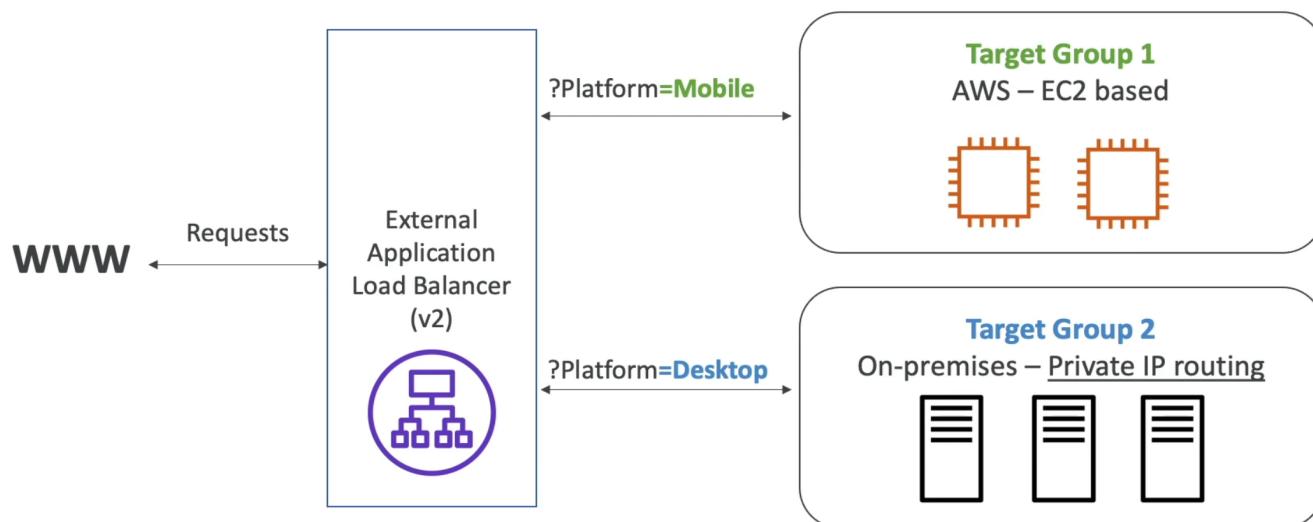
- ALB are a great fit for micro services & container-based application(example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port ECS
- In comparison, we'd need multiple Classic Load Balancer per application

Target Groups

- EC2 instances (can be managed by an Auto Scaling Group) - HTTP
- EC2 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

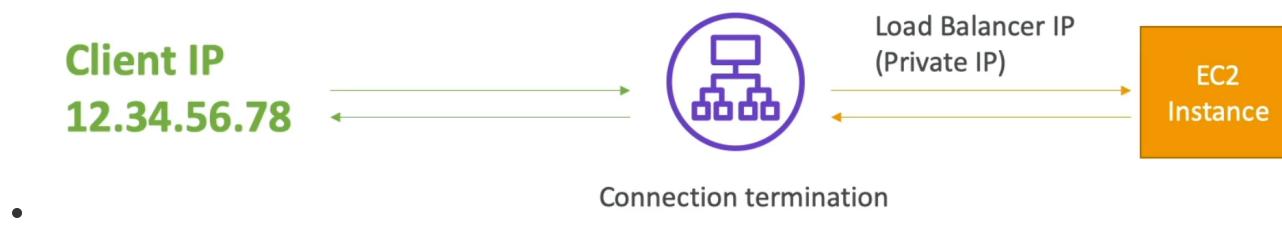
Query Strings/Parameters Routing

example



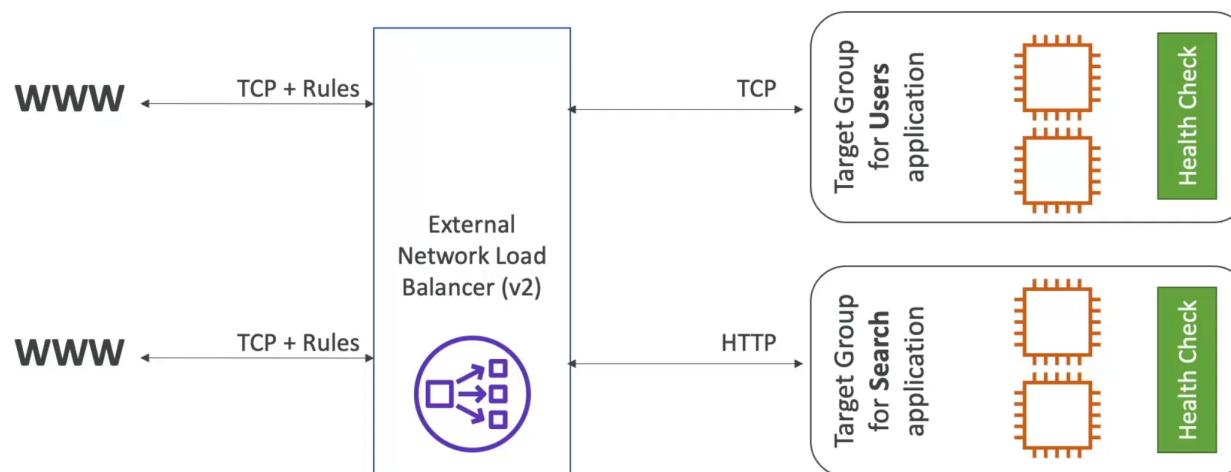
Good to Know

- 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)

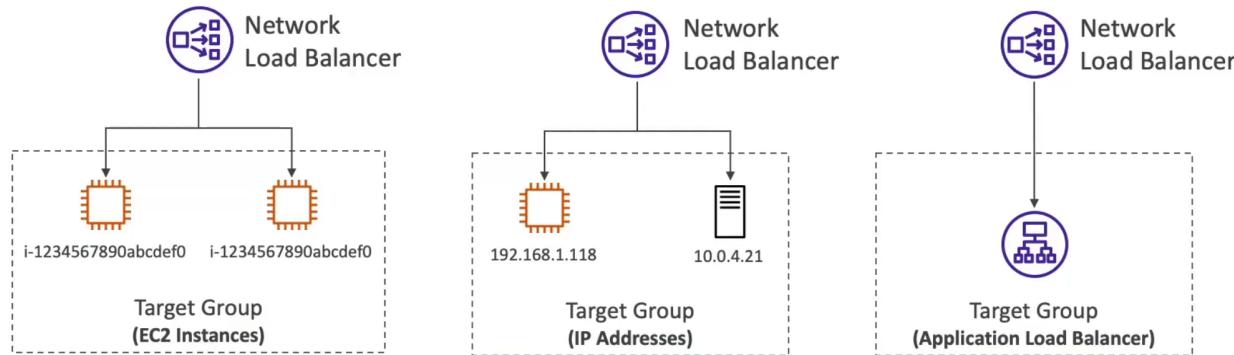
- Network load balancers (Layer 4) allow to:
 - **Forward TCP & UDP traffic to your instances**
 - Handle millions of requests per second
 - Less latency ~100 ms (vs 400 ms for ALB)
- NLB has one static IP per AZ, and supports assigning Elastic IP (helpful for whitelisting specific IP)
- NLB are used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



Target Groups

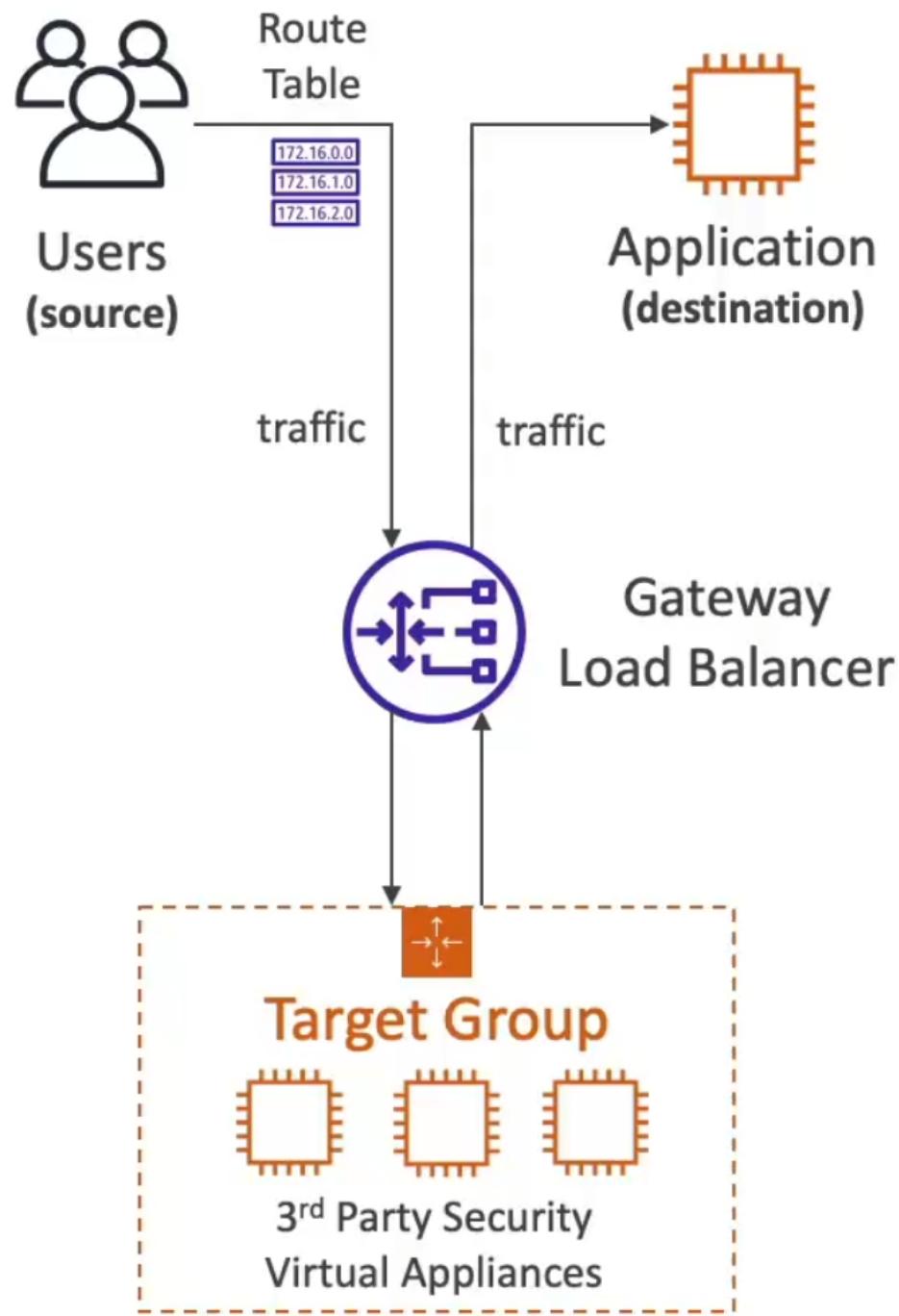
- EC2 instances
- IP Addresses - must be private IPs
- **Application Load Balancer**

- Health Checks support the **TCP, HTTP and HTTPS Protocols**



Gateway Load Balancer

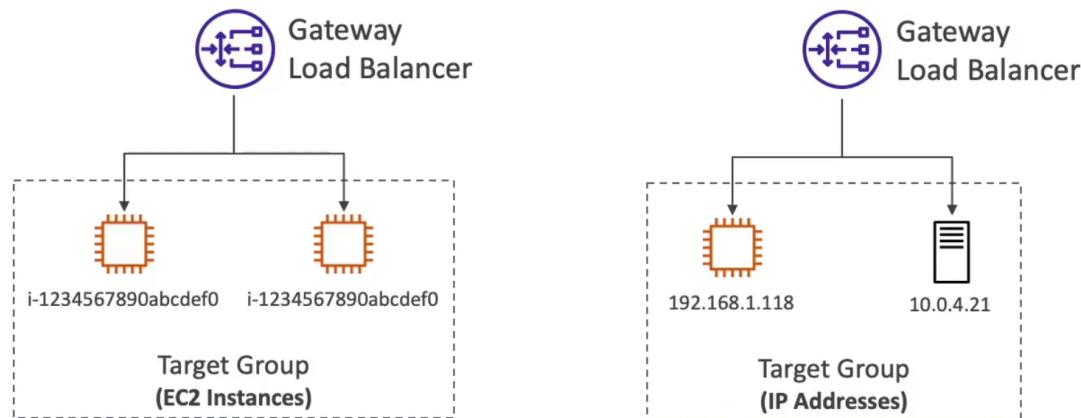
- Deploy, scale, and manage a fleet of 3rd party network virtual appliances in AWS
- Example: Firewalls, Intrusion Detection and Prevention Systems, Deep Packet Inspection Systems, payload manipulation...
- Operates at Layer 3 (Network Layer) - IP Packets
- Combines the following functions:
 - **Transparent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



Target Groups

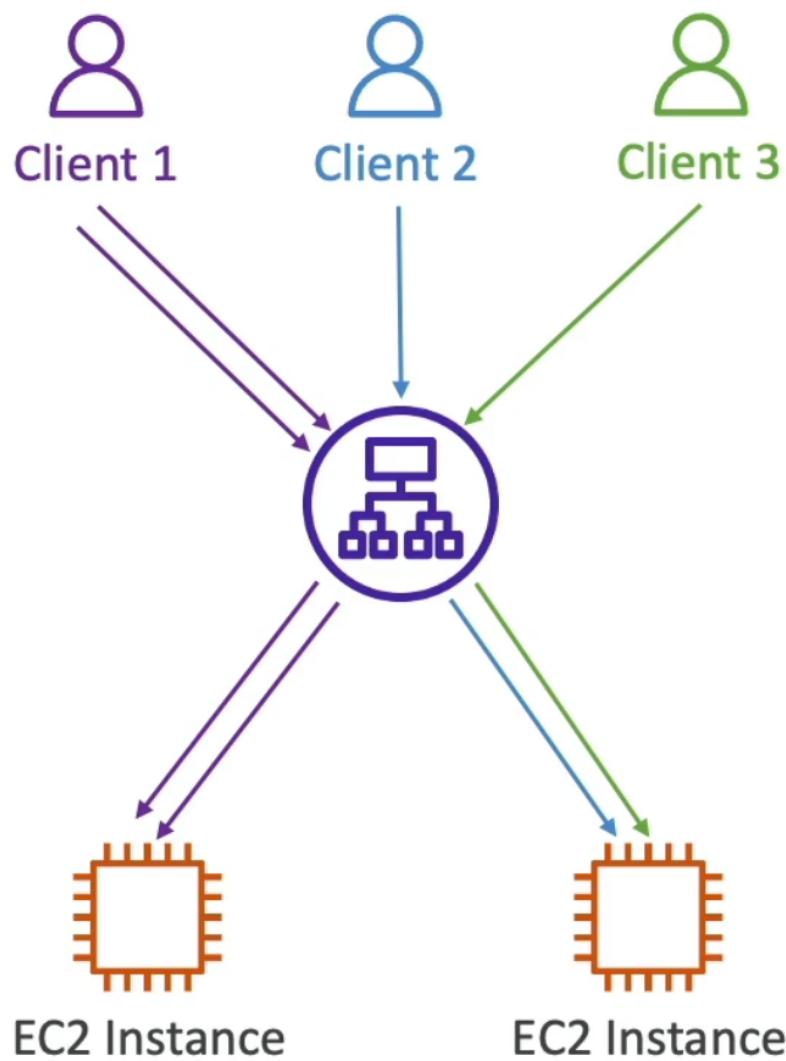
- EC2 Instances

- IP Addresses - must be private IPs



Sticky Session (Session Affinity)

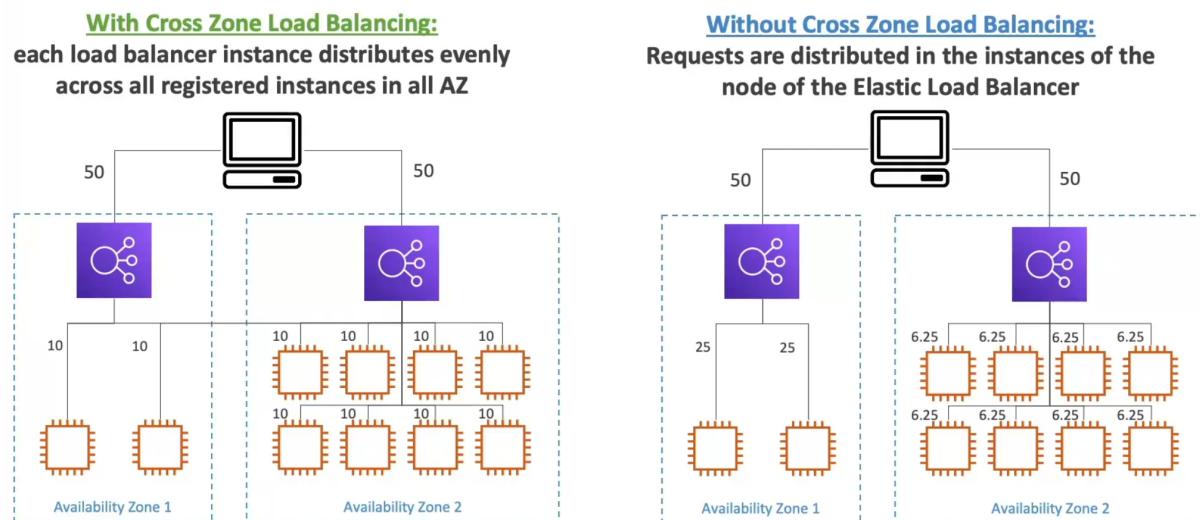
- It is possible to implement stickiness so that the same client is always redirected to the same instance behind a load balancer
- This works for Classic Load Balancers & Application Load Balancers
- The "cookie" used for stickiness has an expiration date you control
- Use case: make sure the user doesn't 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 cookied
 - 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



- **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 enable

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

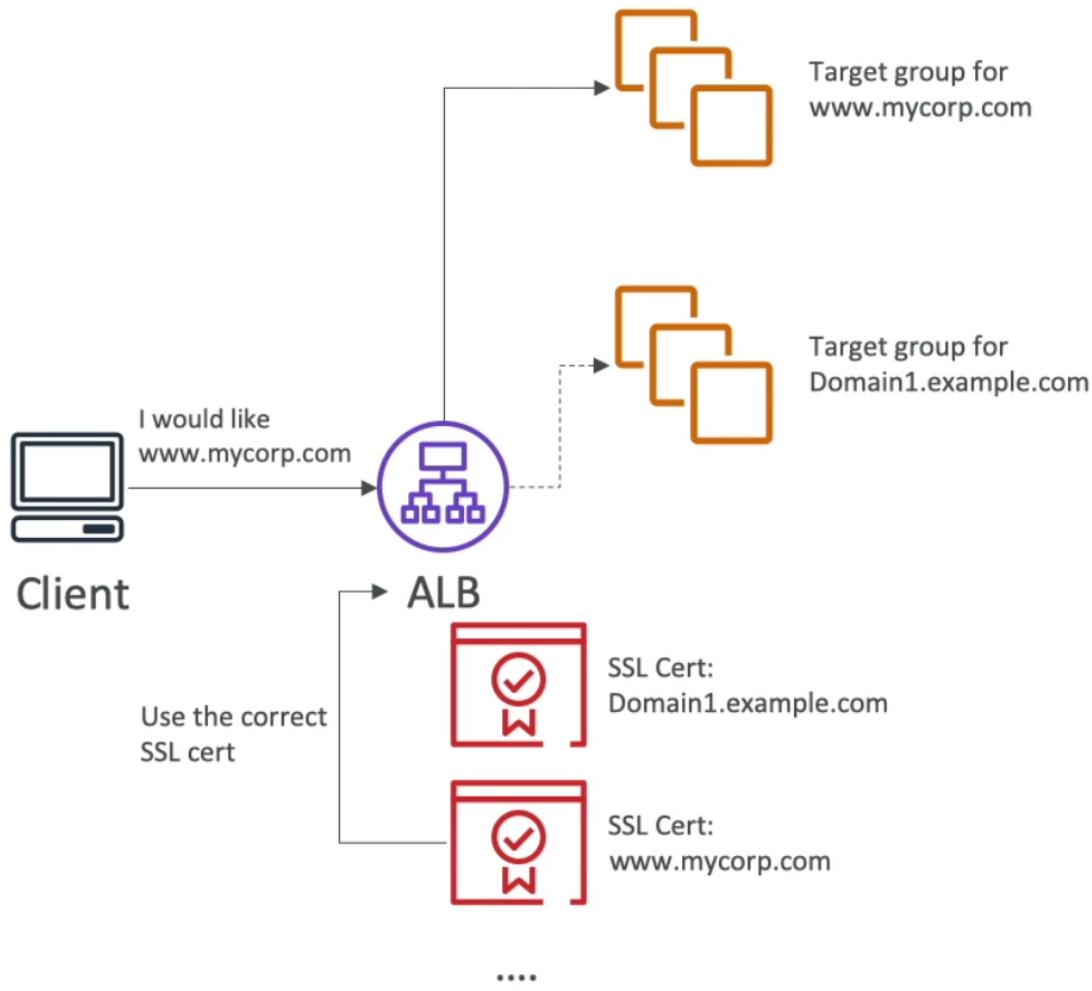


- The load balancer uses an X.509 certificate (SSL/TLS server certificate)
- You can manage certificates using ACM (AWS Certificate Manager)
- You can create/upload your own certificates alternatively
- HTTPS listener:
 - You must specify a default certificate
 - You can add an optional list of certs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

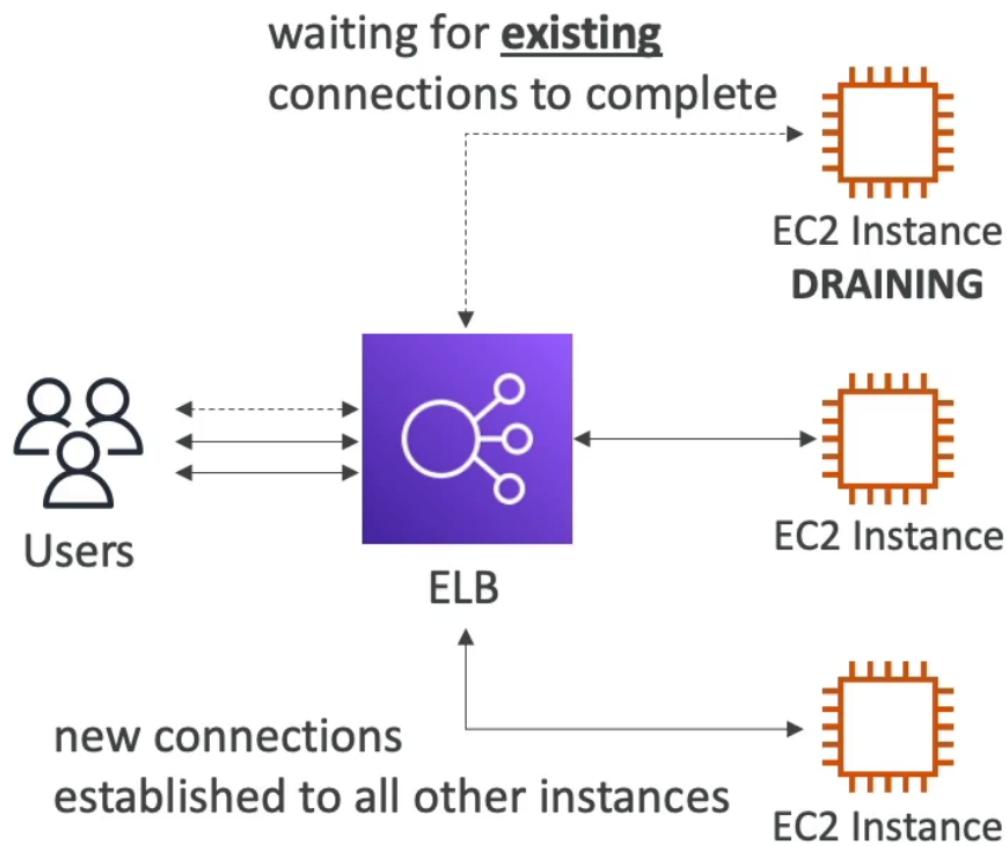
Elastic Load Balancers - SSL Certificates

- 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
 - User Server Name Indication (SNI) to name it work
- Network Load Balancer (v2)
 - Supports multiple listeners with multiple SSL certificates
 - Users 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)

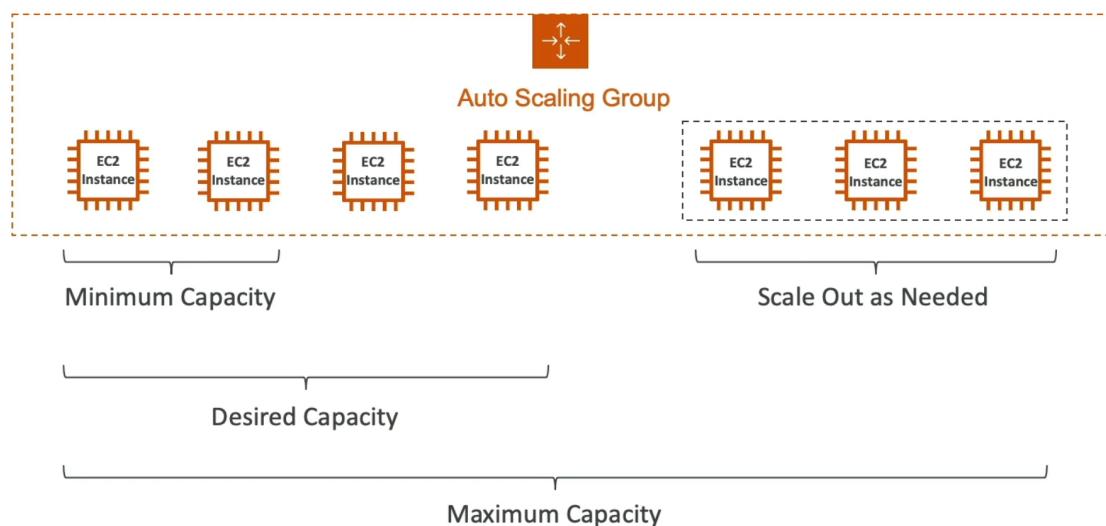
- Set to a low value if your request are short



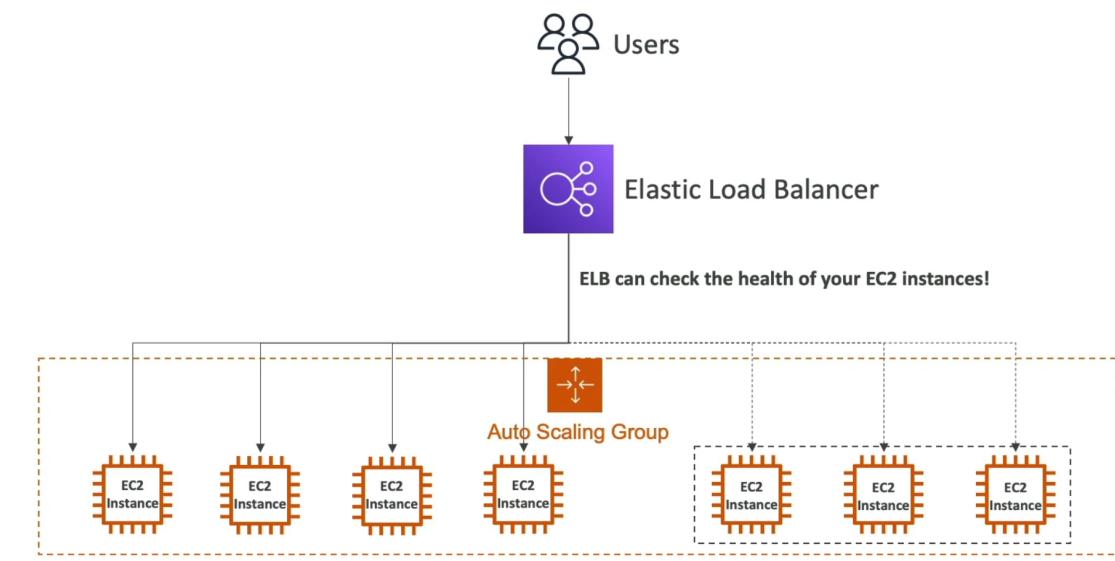
What's an Auto Scaling Group?

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

Auto Scaling Group in AWS

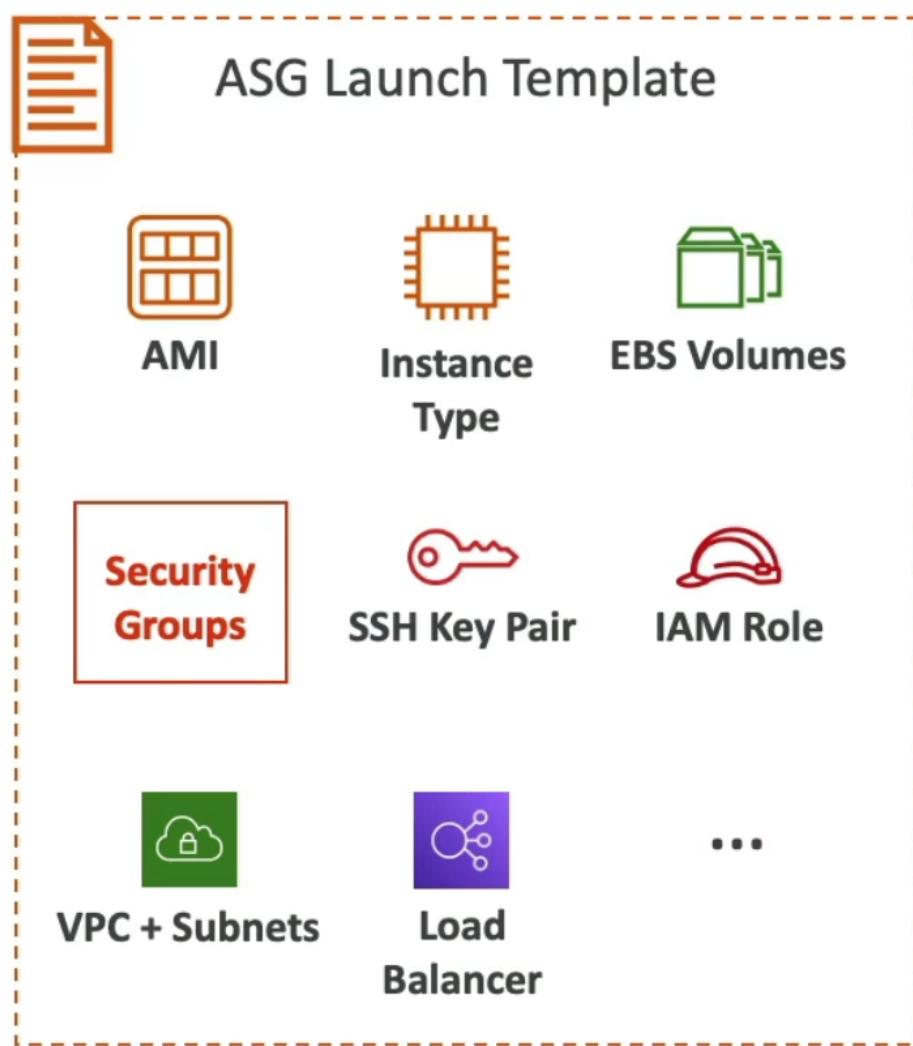


Auto Scaling Group in AWS With Load Balancer



Auto Scaling Group Attributes

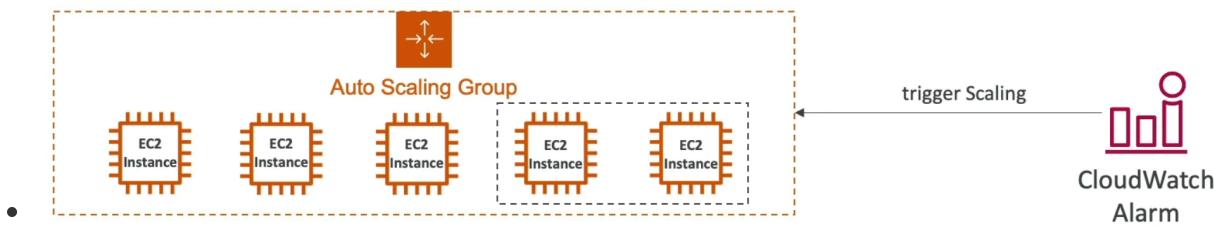
- A **Launch Template** (older "Launch Configurations" are deprecated)
 - AMI + Instance Type
 - EC2 User Data
 - EBS Volumes
 - Security Groups
 - SSH Key Pair
 - IAM Roles for your EC2 Instances
 - Network + Subnets Information
 - Load Balancer Information



- Min Size / Max Size / Initial Capacity

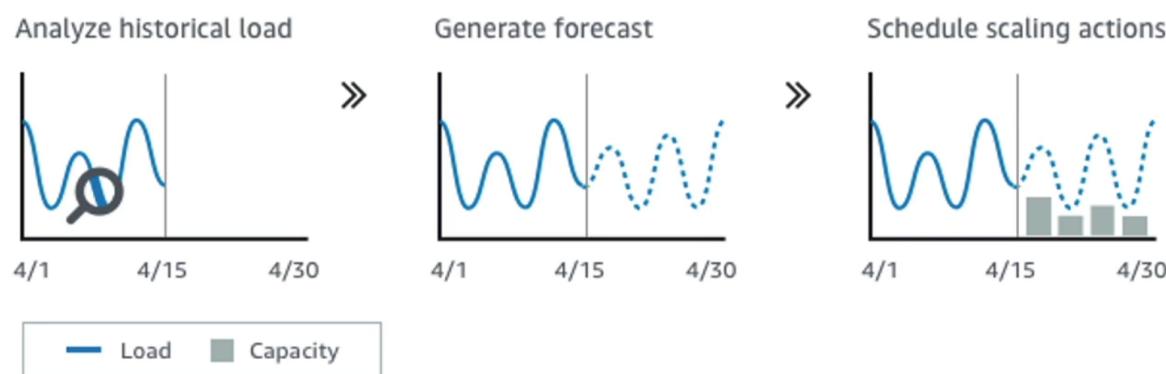
Auto Scaling - CloudWatch Alarm & Scaling

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



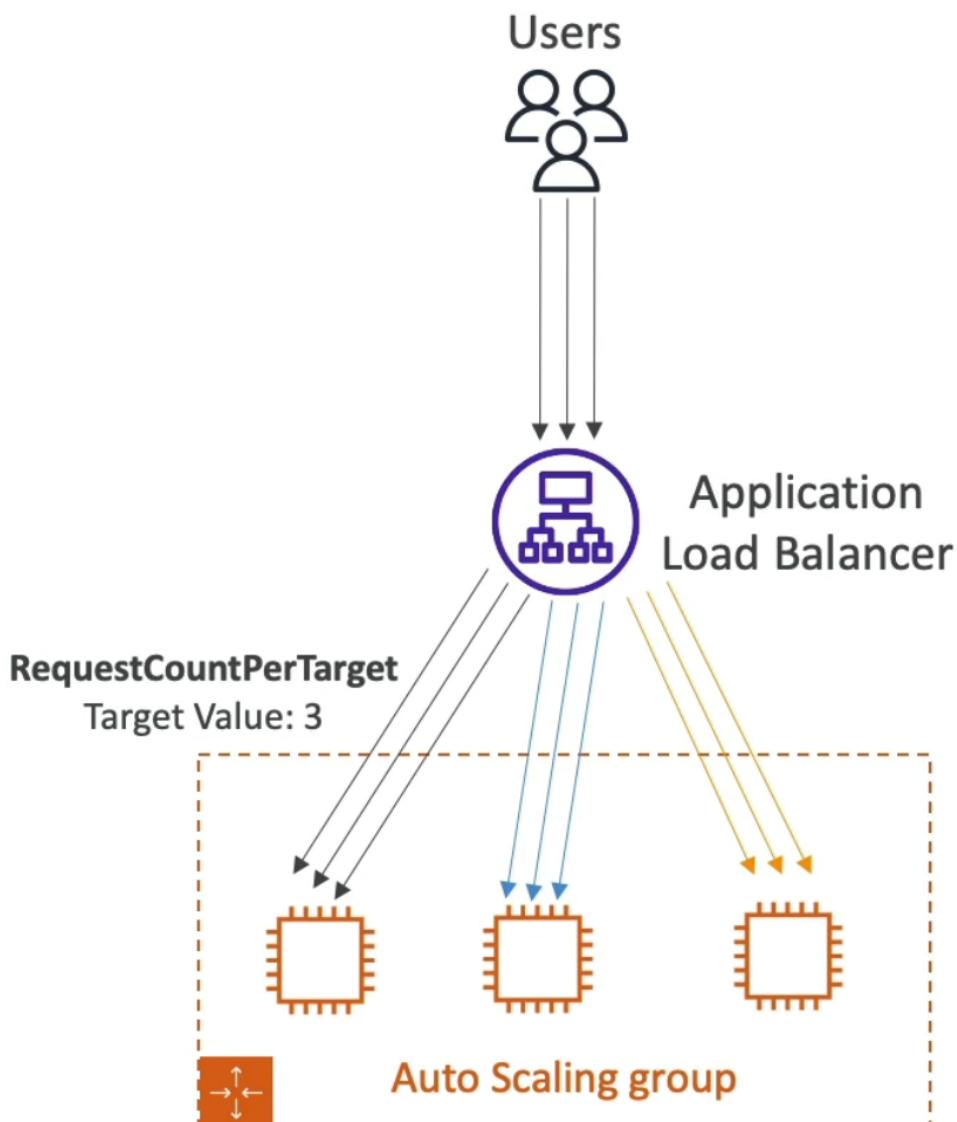
Auto Scaling Groups - Dynamic Scaling Policies

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



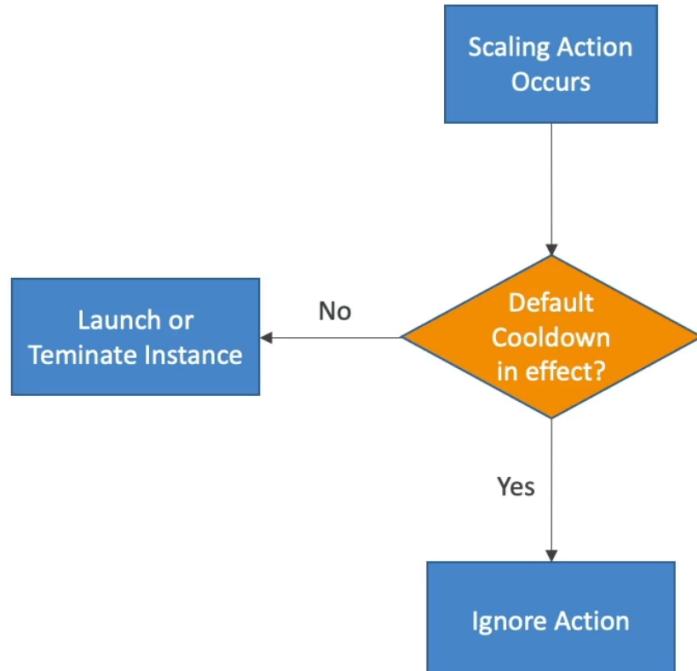
Good metrics to scale on

- **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 your 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



```
sudo amazon-linux-extras install epel -y  
sudo yum install stress -y
```

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)

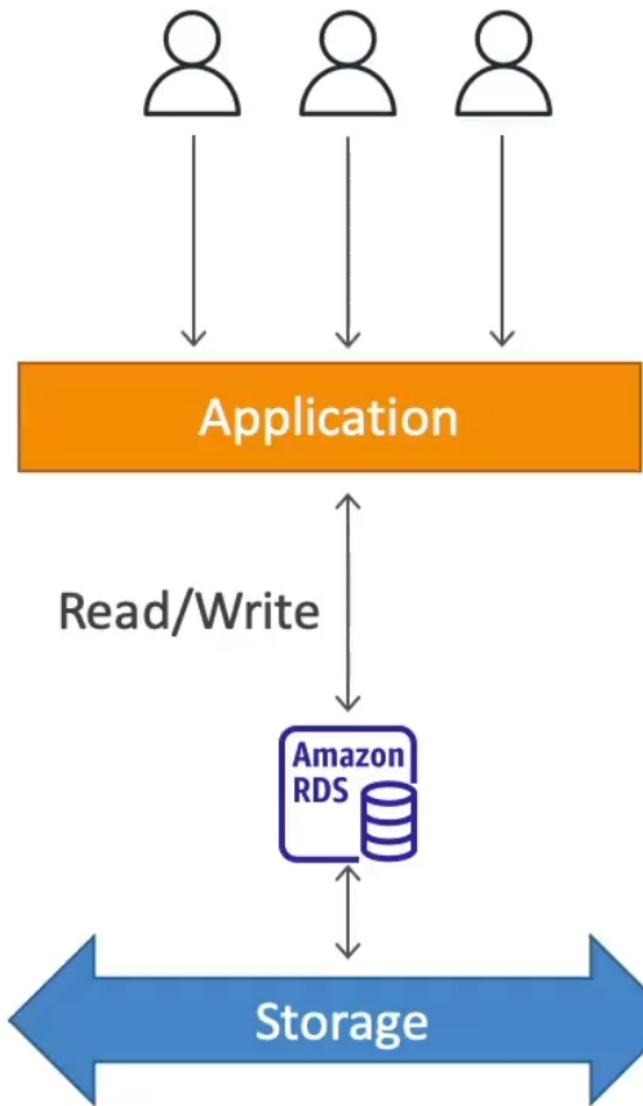
Advantage over using RDS versus deploying DB on EC2

- 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

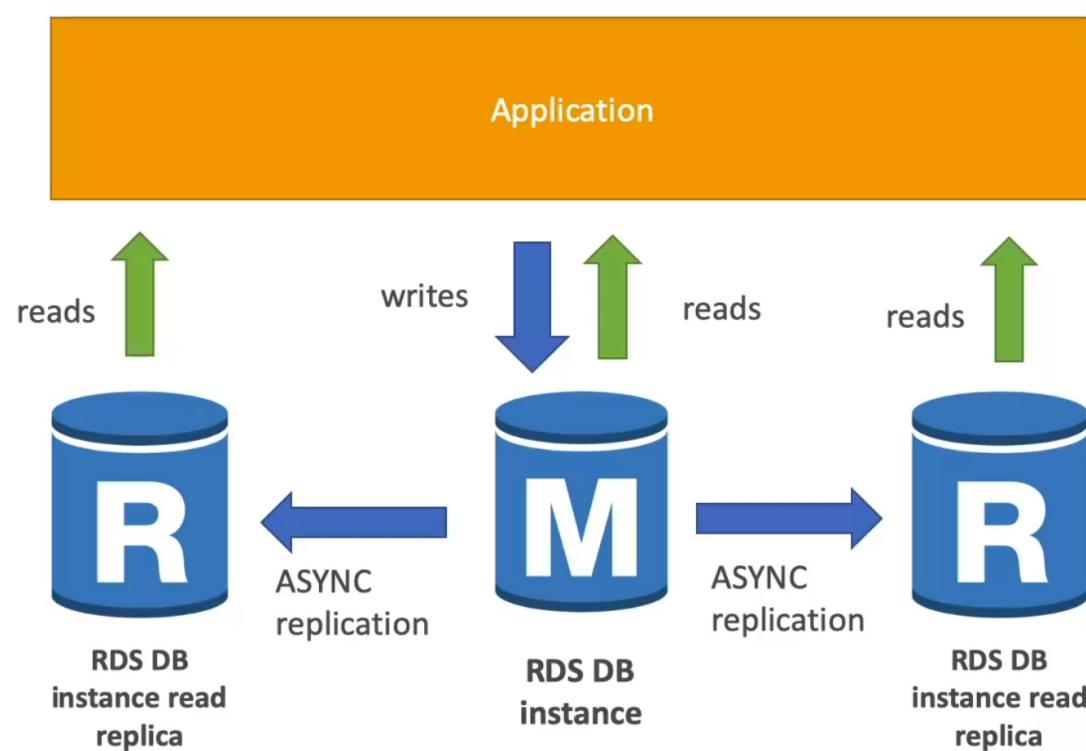
- Helps you increase storage on your RDS DB instance dynamically
- When RDS detects you are running out of free database storage, it scales automatically
- Avoid manually scaling your database storage
- You have to set Maximum Storage Threshold (maximum limit for DB storage)
- Automatically modify storage 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

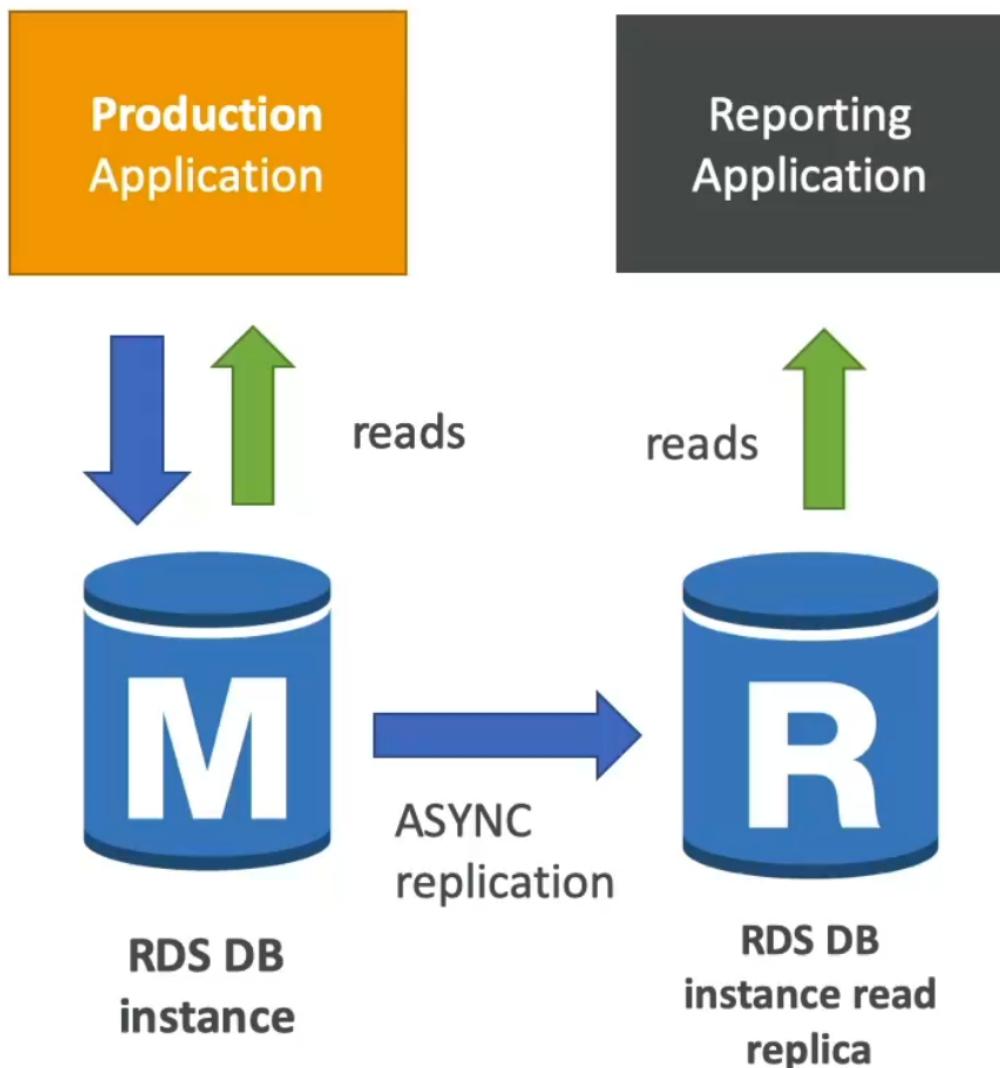
- Up to 5 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is ASYNC, so reads are eventually consistent
- Replicas can be promoted to their own DB
- Applications must update the connection string to leverage read replicas



RDS Read Replicas - Use Cases

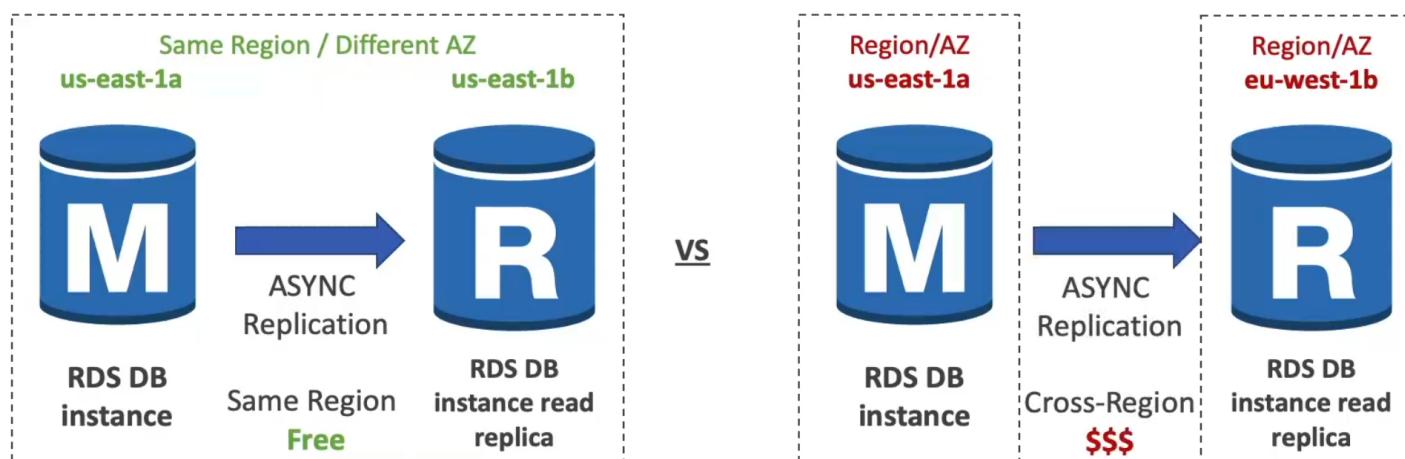
- You have a production database that is taking on normal load
- You want to run a reporting application to run some analytics
- You create a Read Replica to run the new workload there

- The production application is unaffected
- Read replicas are used for SELECT (=read) only kind of statements (not INSERT,UPDATE,DELETE)



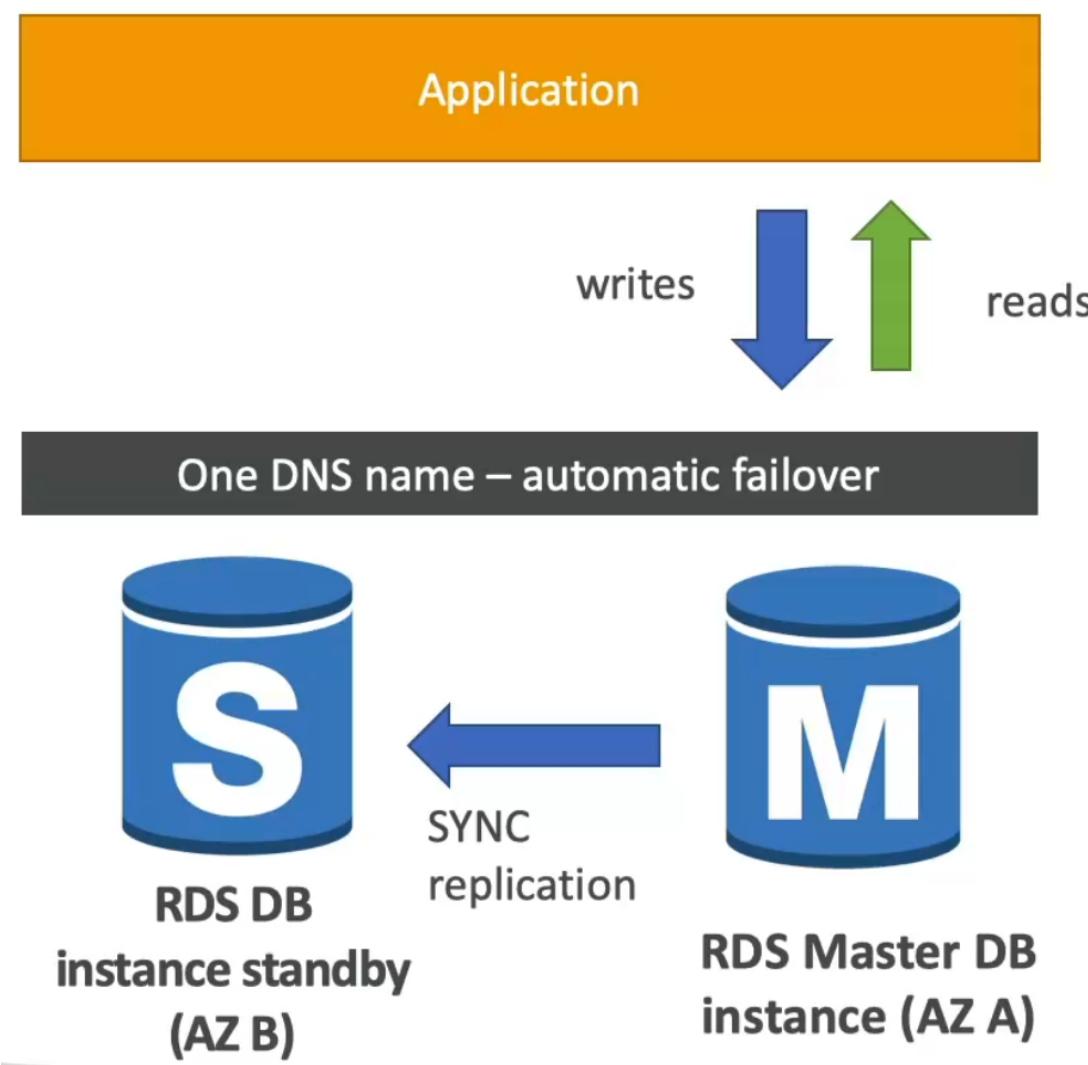
RDS Read Replicas - Network Cost

- 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**



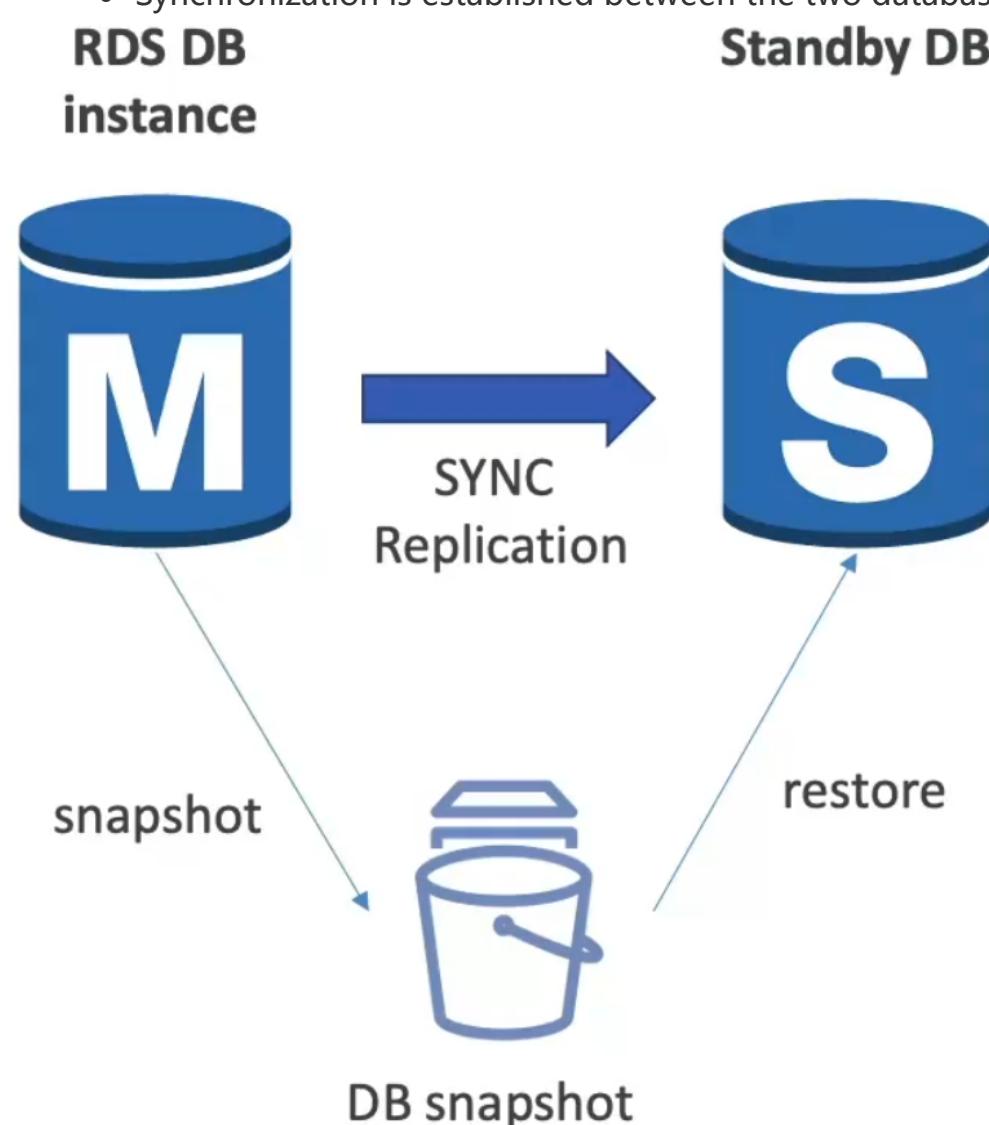
RDS Multi AZ (Disaster Recovery)

- SYSNC** replication
- One DNS name - automatic app failover to standby
- Increase **availability**
- Failover in case of loss of AZ, loss of net work, 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

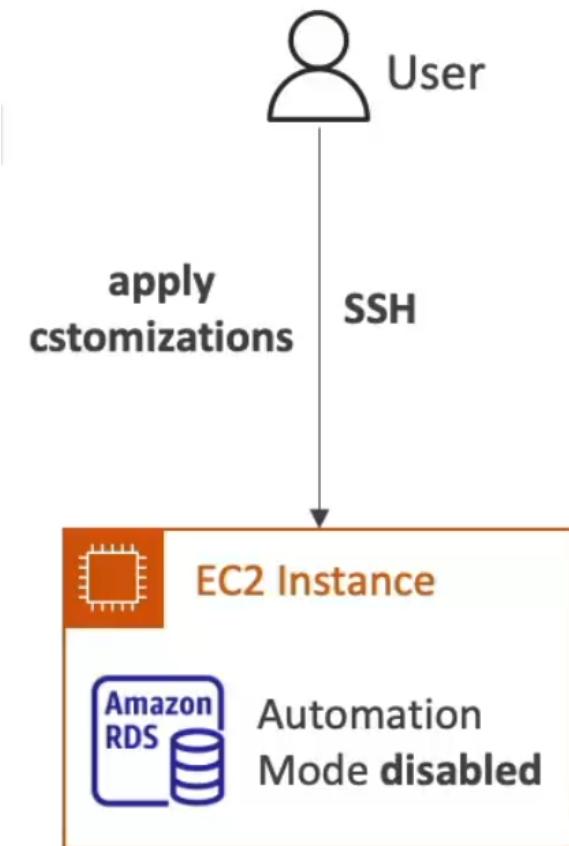
- Zero downtime operation (no need to stop the DB)
- Just click on "modify" for the database
- The following happens internally:
 - A snapshot is taken
 - A new DB is restored from the snapshot in new AZ
 - Synchronization is established between the two databases



RDS Custom

- Managed Oracle and Microsoft SQL Server Database with OS and database customization

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

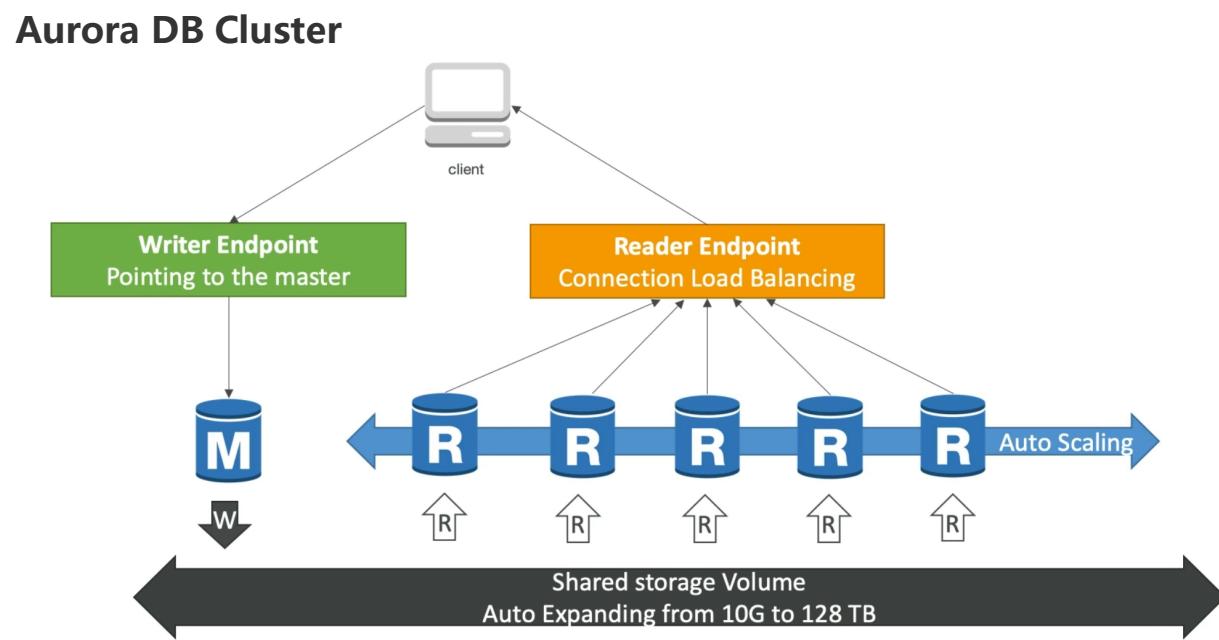
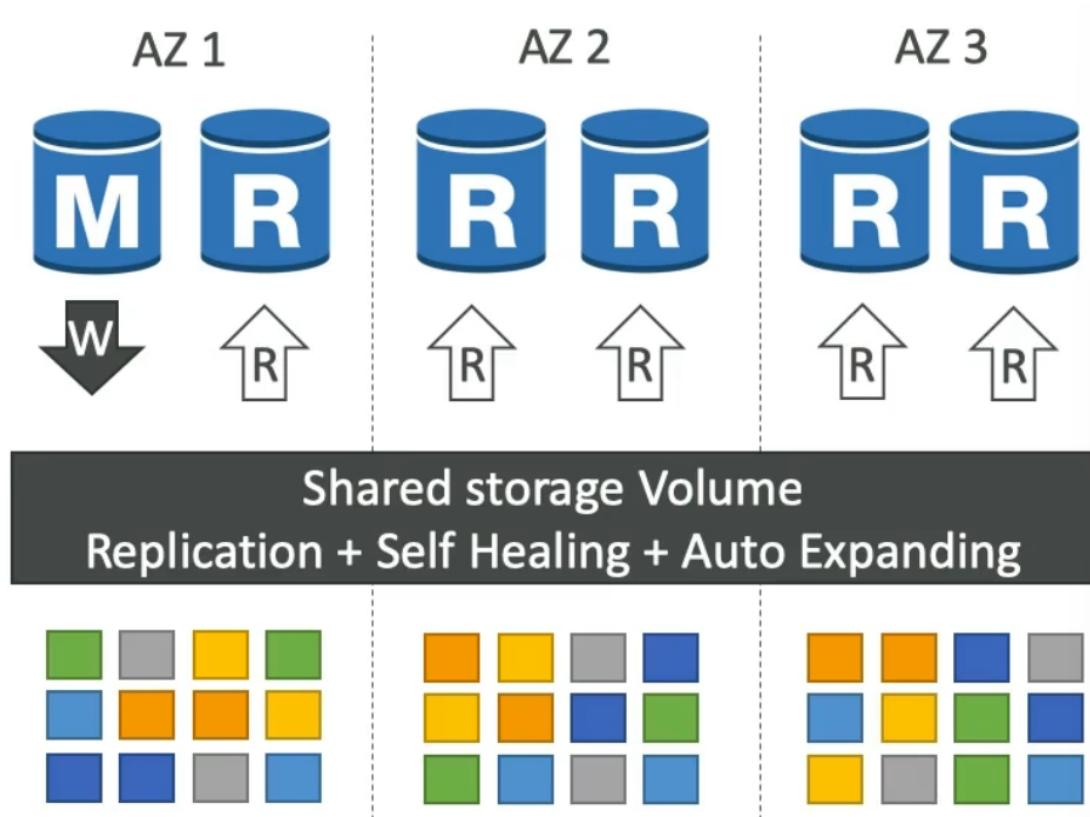


Amazon Aurora

- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is "AWS cloud optimized" and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 128 TB.
- Aurora can have 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 native.
- Aurora costs more than RDS (20% more) - but is more efficient

Aurora High Availability and Read Scaling

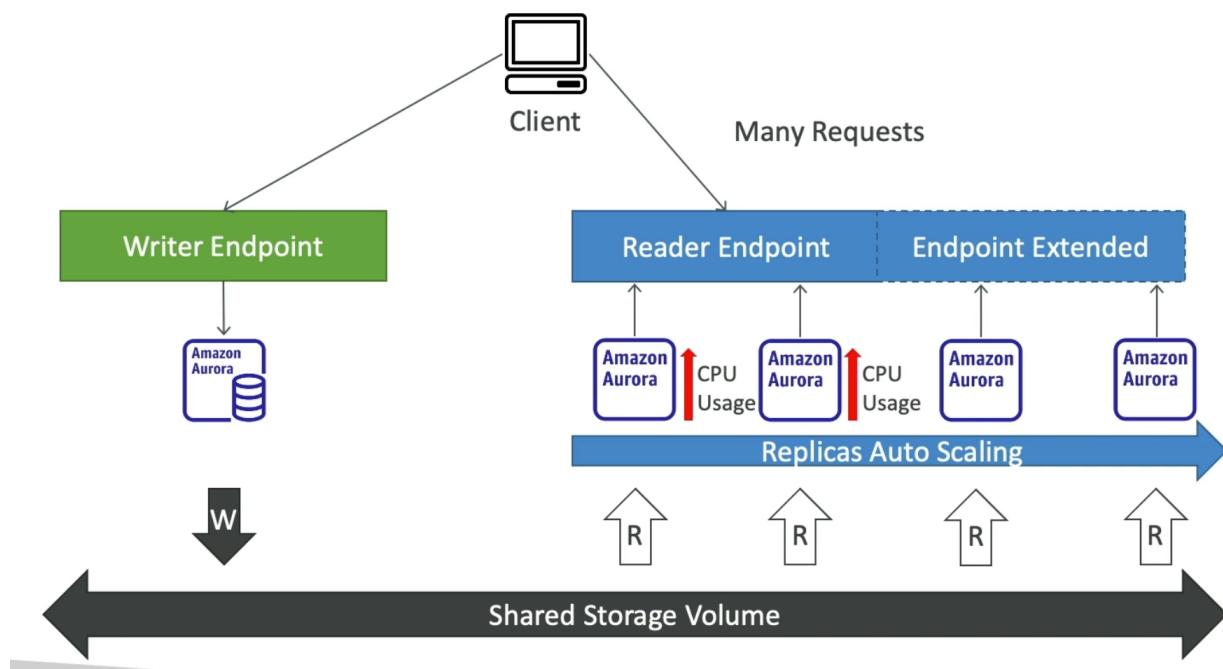
- 6 copies of your data across 3 AZ:
 - 4 copies out of 6 needed for writes
 - 3 copies out of 6 need for reads
 - Self healing with peer-to-peer replication
 - Storage is striped across 100s of volumes
- One Aurora Instance takes writes (master)
- Automated failover for master in less than 30 seconds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



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 Relicas - Auto Scaling

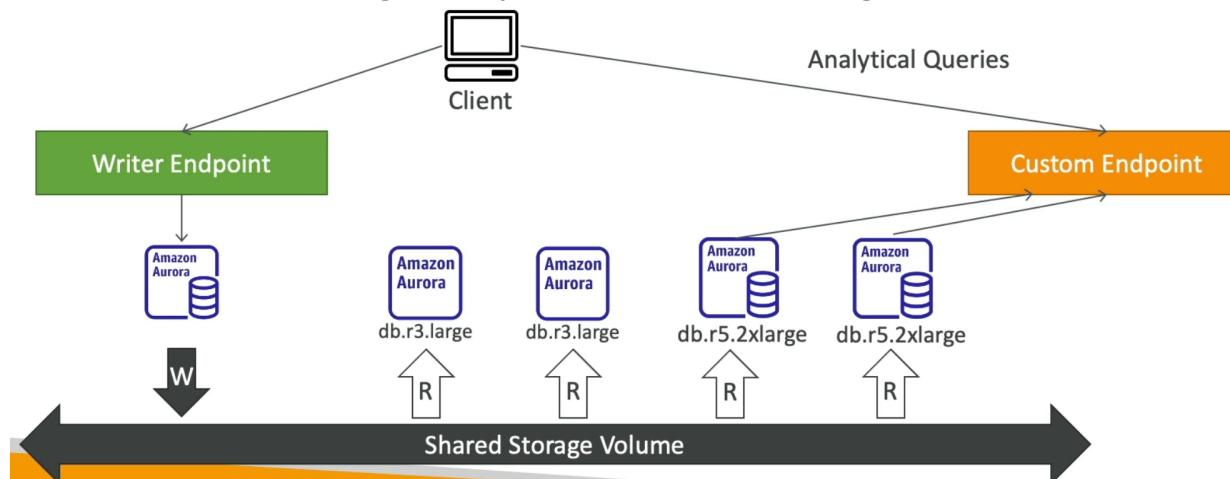


Aurora - Custom Endpoints

Define a subnet of Aurora Instances as a Custom Endpoint

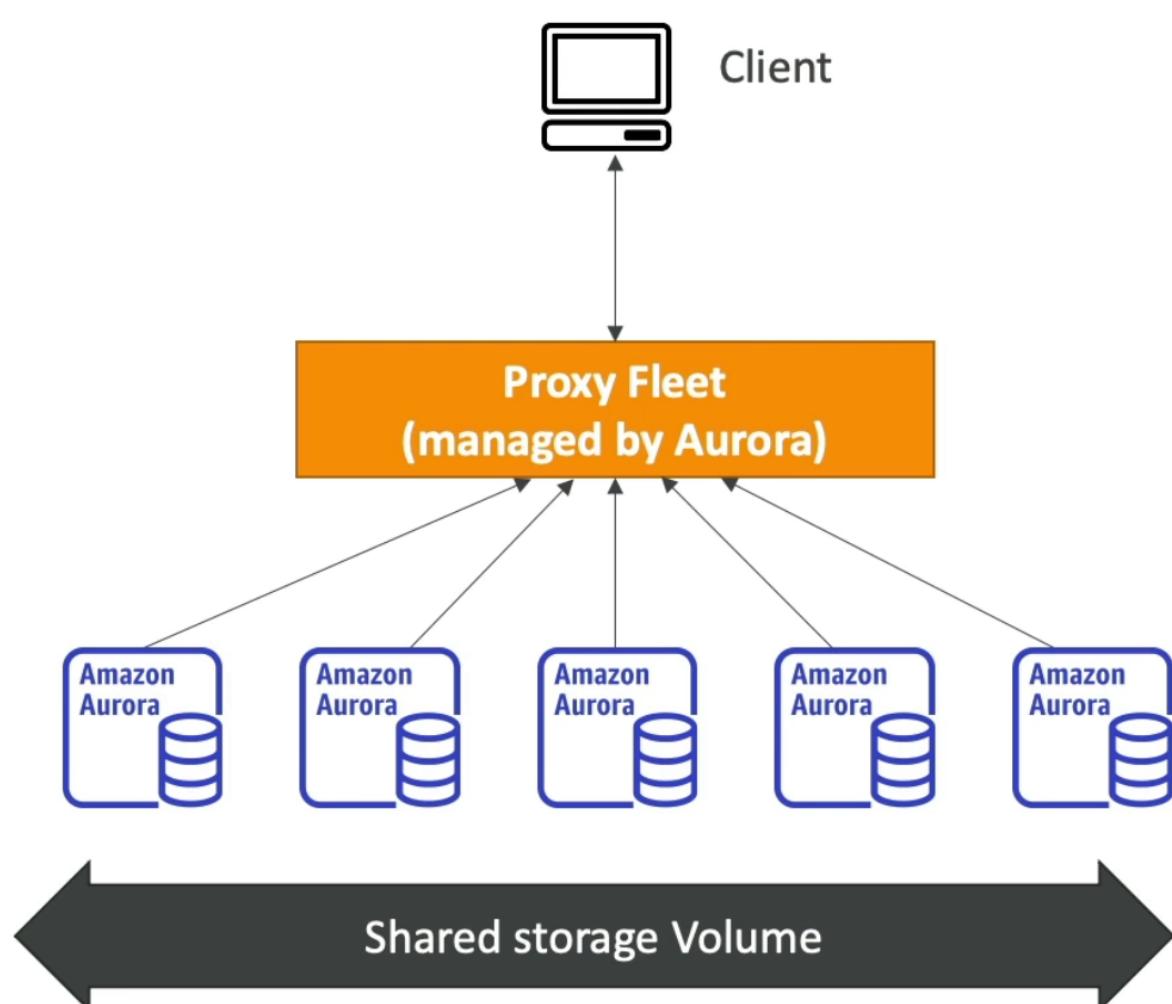
Example: Run analytical queries on specific replicas

The Reader Endpoint is generally not used after defining Custom Endpoints



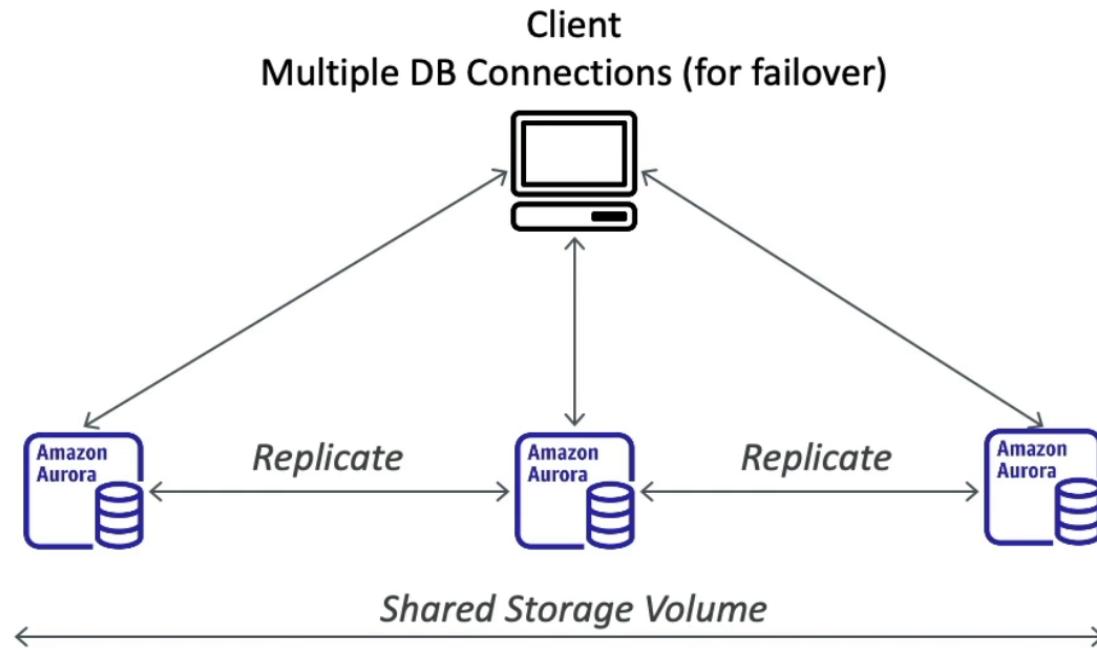
Aurora Serverless

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



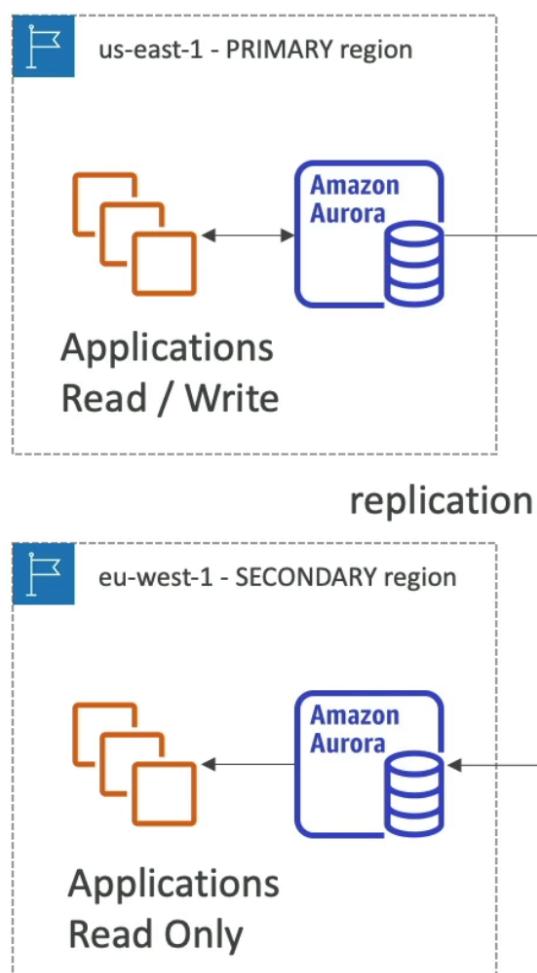
Aurora Multi-Master

- 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

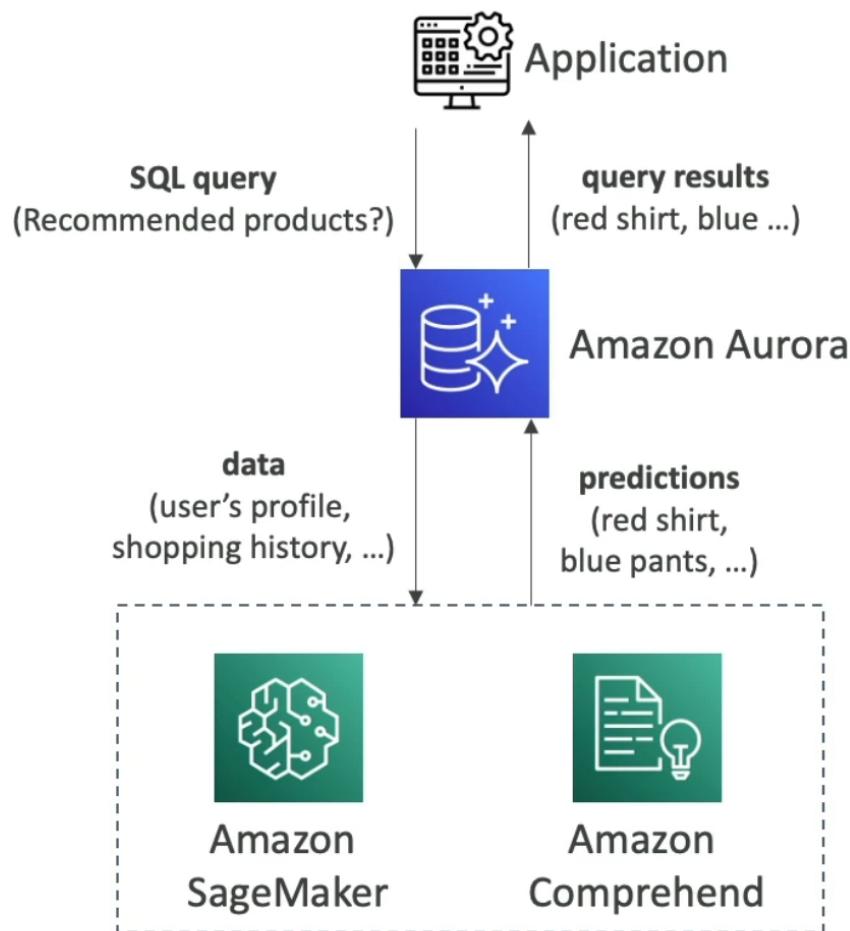
- Aurora Cross Region Read Replicas:
 - Useful for disaster recovery
 - Simple to put in place
- Aurora Global Database (recommended):
 - 1 Primary Region (read/write)
 - Up to 5 secondary (read-only) regions, replication lag is less than 1 second
 - Up to 16 Read Replicas per secondary region
 - Helps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



Aurora Machine Learning

- Enables you to add ML-based predictions to your applications via SQL
- Simple, optimized, and secure integration between Aurora and AWS ML services

- Supported services
 - Amazon SageMaker (use with any ML model)
 - Amazon Comprehend (for sentiment analysis)
- You don't need to have ML experience
- Use cases: fraud detection, ads targeting, sentiment analysis, product recommendations



RDS Backups

- Automated backups:
 - Daily full backup of the database (during the 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)
 - 1 to 35 days of retention, set 0 to disable automated backups
- Manual DB Snapshots
 - Manually triggered by the user
 - Retention of backup for as long as you want
- Trick: in a stopped RDS database, you will still pay for storage. If you plan on stopping it for a long time, you should snapshot & restore instead

Aurora Backups

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

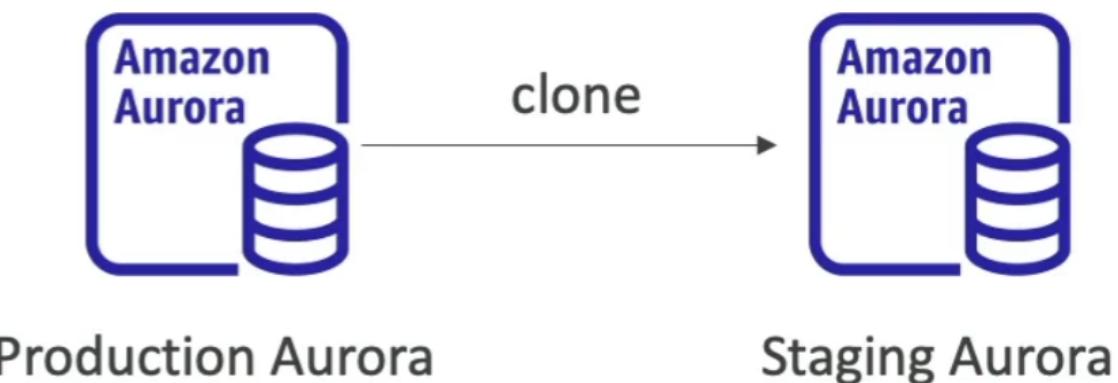
RDS & Aurora Restore options

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

- Store the backup file on Amazon S3
- Restore the backup file onto a new Aurora cluster running MySQL

Aurora Database Cloning

- Create a new Aurora DB Cluster from an existing one
- Faster than snapshot & restore
- The new DB cluster uses the same cluster volume and data as the original but will change when data updates are made
- Very fast & cost-effective Useful to create a "staging" database from a "production" database without impacting the production database



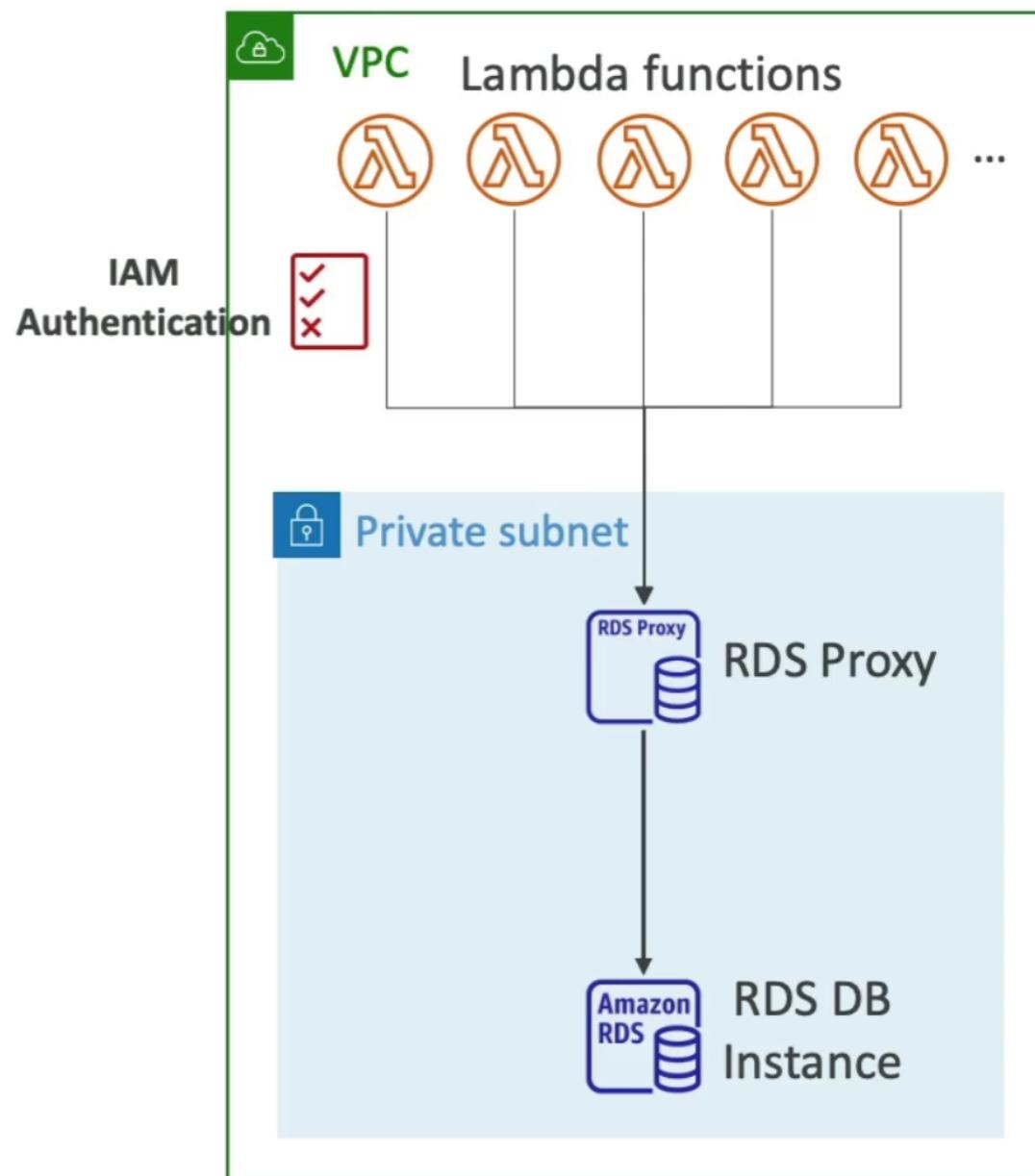
RDS & Aurora Security

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

Amazon RDS Proxy

- Fully managed database proxy for RDS
- Allows apps to pool and share DB connections established with the database
- **Improving database efficiency by reducing the stress on database resources (e.g., CPU, RAM) and minimize open connections (and timeouts)**
- Serverless, autoscaling, highly available (multi-AZ)
- **Reduced RDS & Aurora failover time by up 66%**
- Supports RDS (MySQL, PostgreSQL, MariaDB) and Aurora (MySQL, PostgreSQL)
- No code changes required for most apps
- **Enforce IAM Authentication for DB, and securely store credentials in AWS Secrets Manager**

- RDS Proxy is never publicly accessible (must be accessed from VPC)



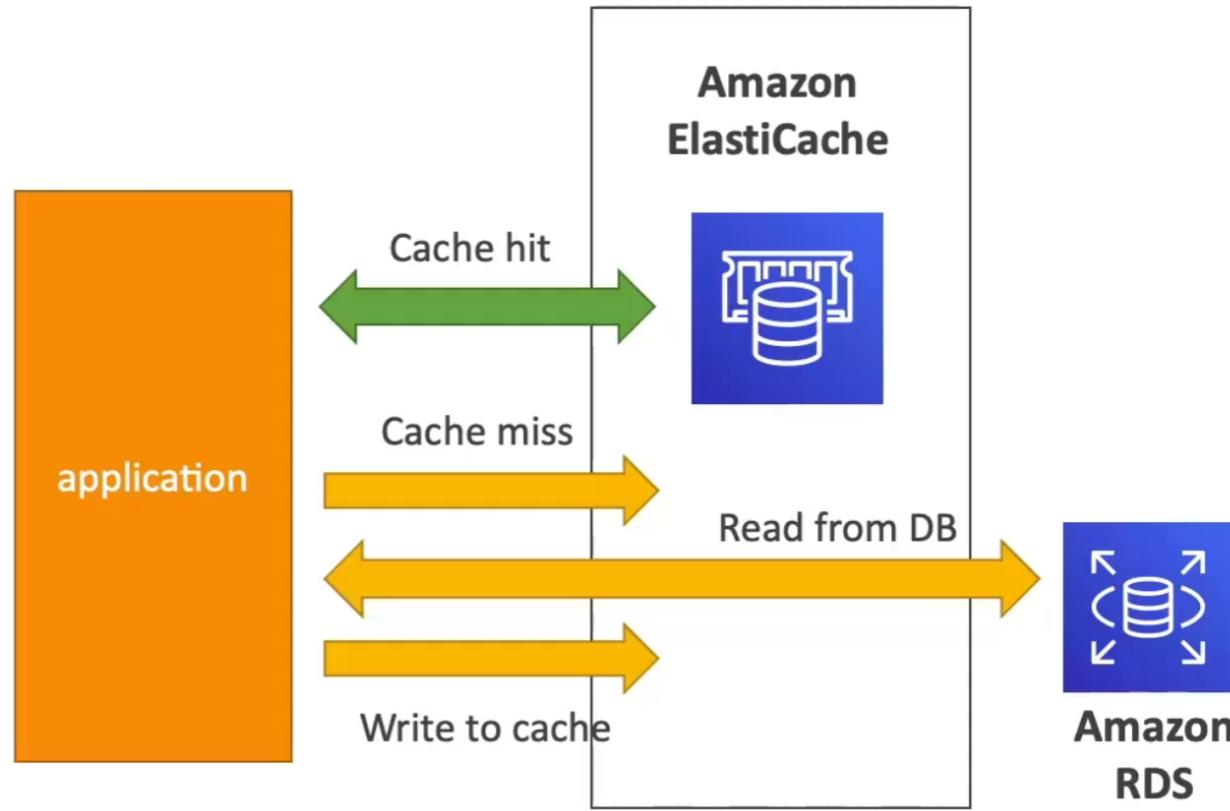
Amazon ElastiCache Overview

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

ElastiCache Solution Architecture - DB Cache

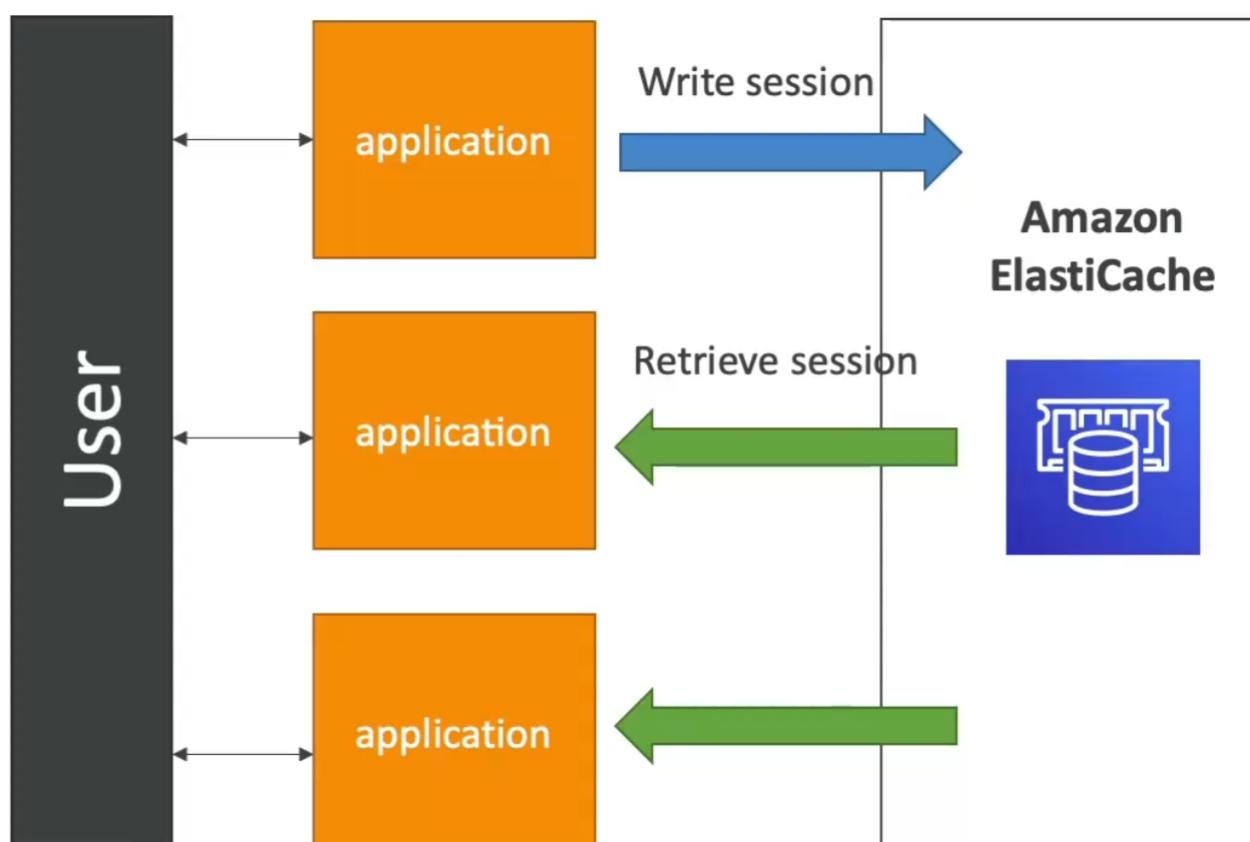
- Applications queries ElastiCache, if not available, get from RDS and store in ElastiCache.
- Helps relieve load in RDS

- Cache must have an invalidation strategy to make sure only the most current data is used in there.



ElastiCache Solution Architecture - User Session Store

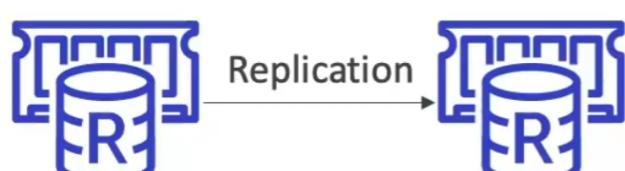
- User logs into any of the application
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in



ElasticCache - 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

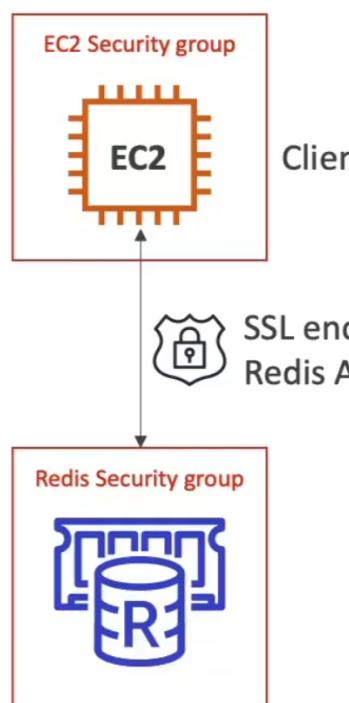
- Multi-node for partitioning of data (sharding)

- No high availability (replication)
- Non persistent
- No backup and restore
- Multi-threaded architecture



ElastiCache - Cache Security

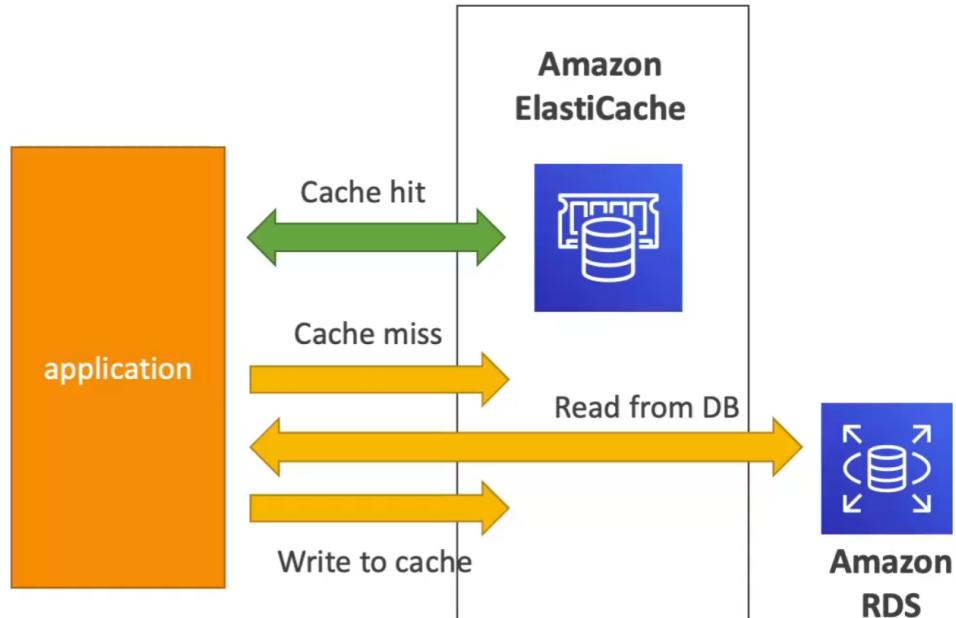
- All caches in ElastiCache:
 - Do not support IAM authentication
 - IAM policies on ElastiCache are only used for AWS API-level security
- Redis AUTH
 - You can set a "password/token" when you create a Redis cluster
 - This is an extra level of security for your cache (on top of security groups)
 - Support SSL in flight encryption
- Memcached
 - Supports SASL-based authentication (advanced)



Pattern 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)

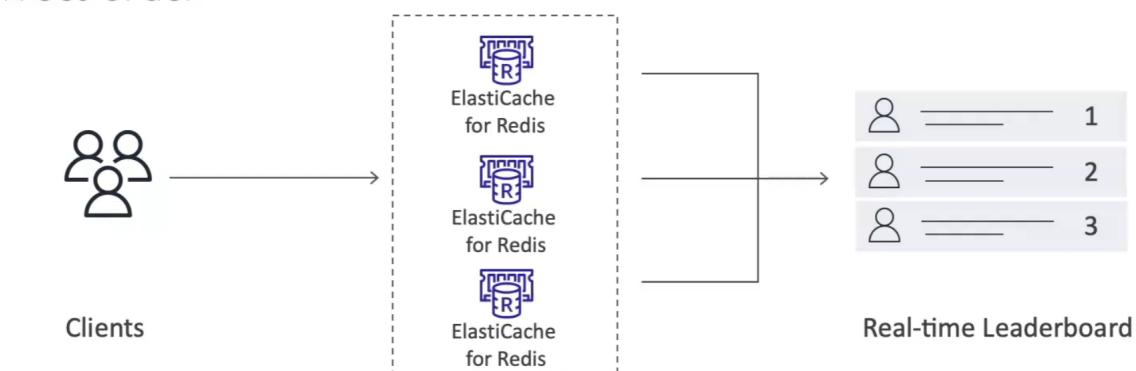
- Quote: There are only two hard things in Computer Science: cache invalidation and naming things



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



What is DNS?

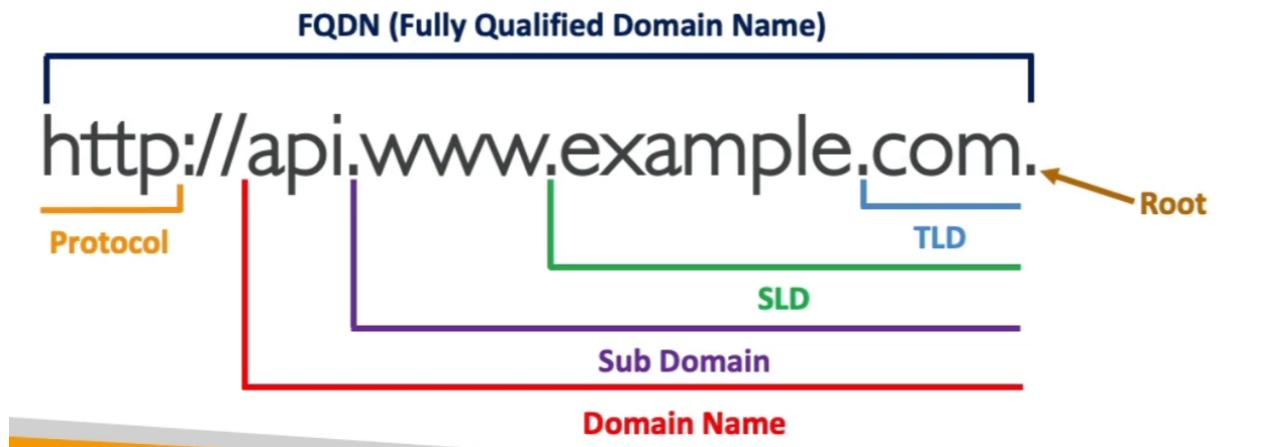
- Domain Name System which translates the human friendly hostnames into the machine IP addresses
- `www.google.com` => `172.217.18.36`
- DNS is the backbone of the Internet
- DNS uses hierarchical naming structure

```
.com
example.com
www.example.com
api.example.com
```

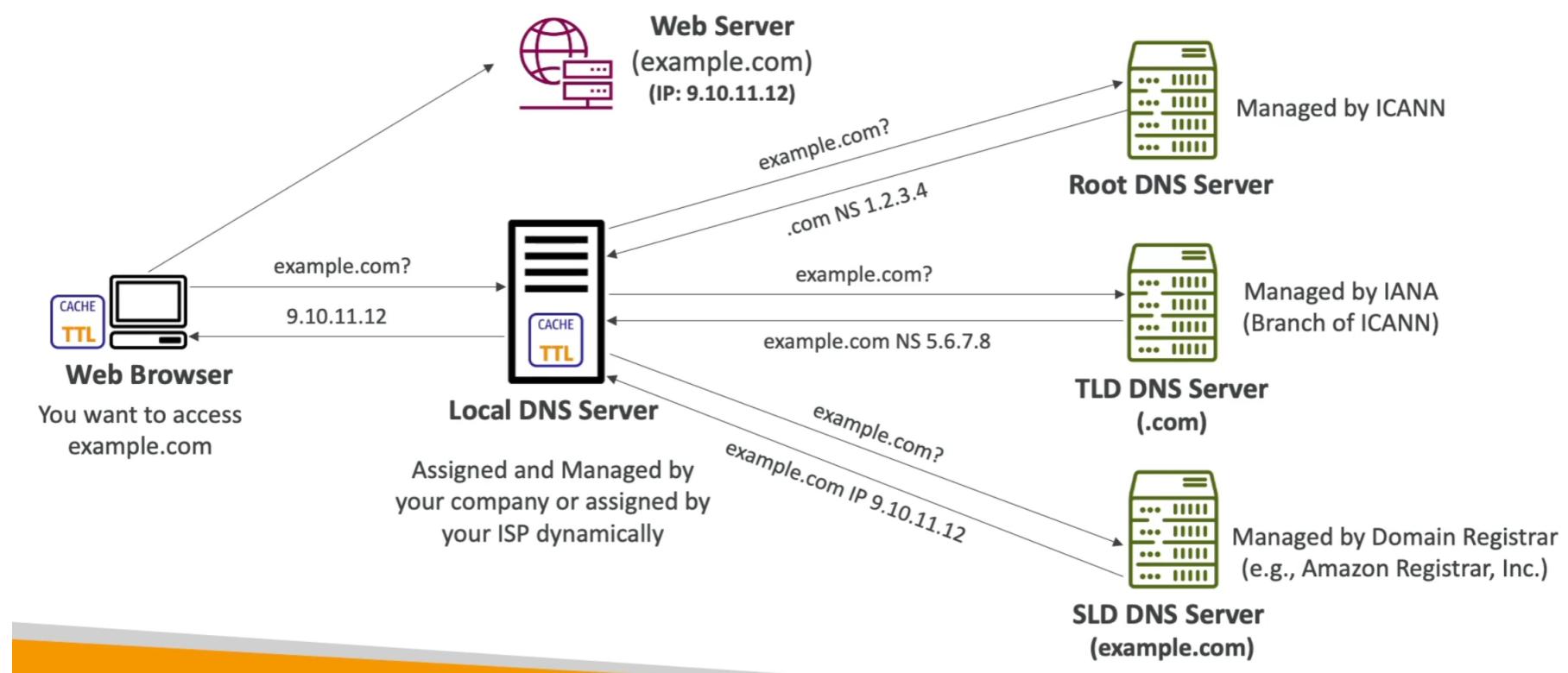
DNS Terminologies

- 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, ...

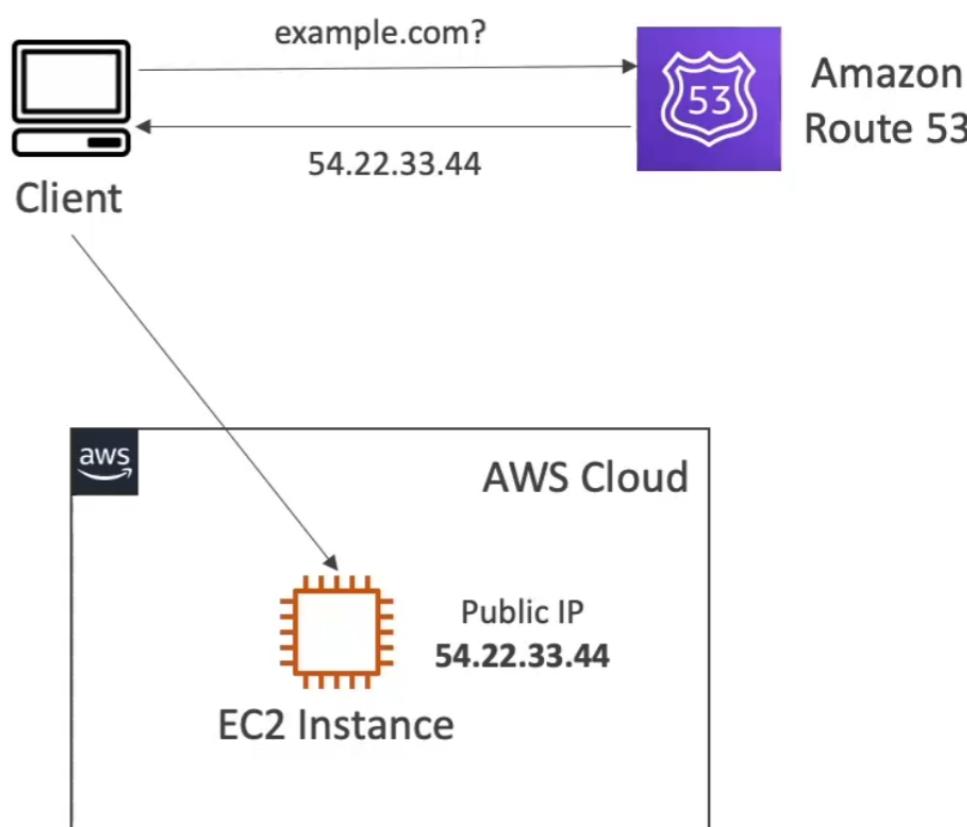


How DNS Works



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

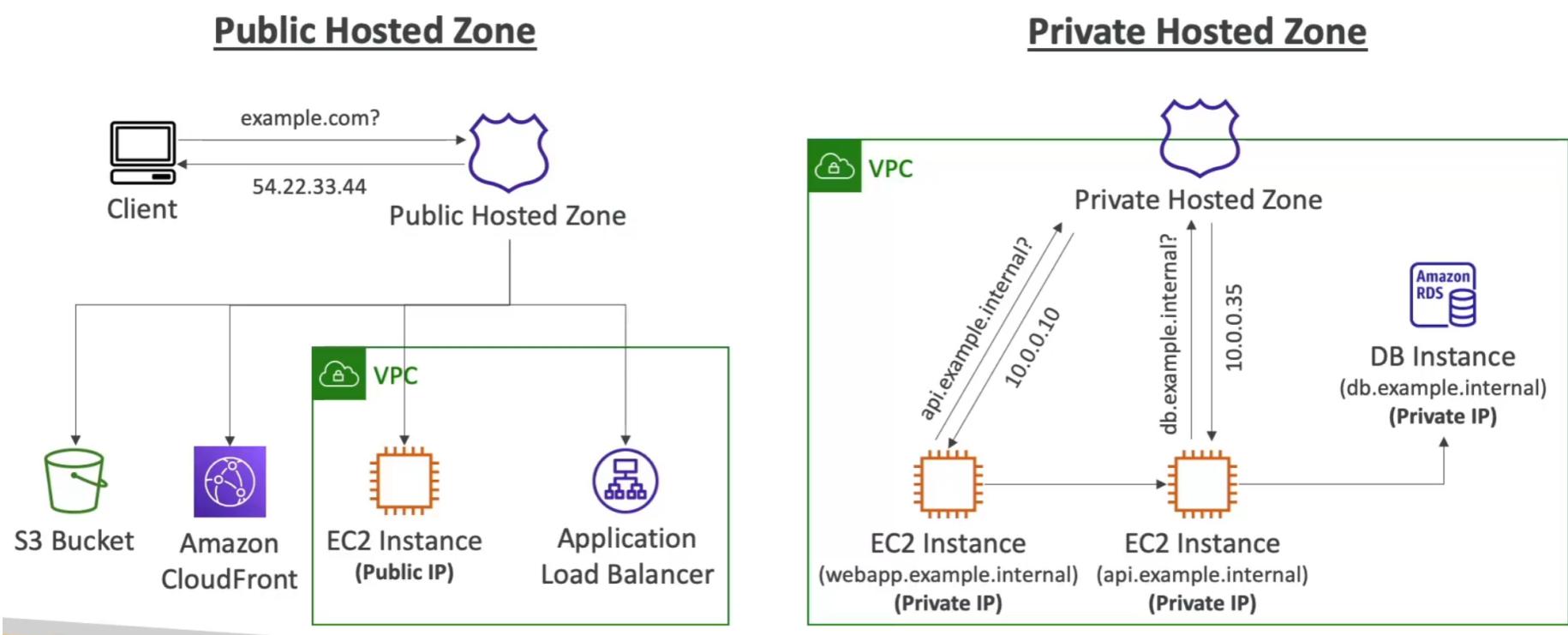
The screenshot shows the 'Create record' page for a hosted zone named 'stephanetheteacher.com'. The 'Record name' field contains 'demo' and the 'Record type' dropdown is set to 'A – Routes traffic to an IPv4 address and so...'. The 'Value' field contains '3.70.14.253'. The 'TTL (seconds)' field is set to '120'. The 'Routing policy' dropdown is set to 'Simple routing'. At the bottom, there are buttons for 'Cancel' and 'Create records'.

- **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

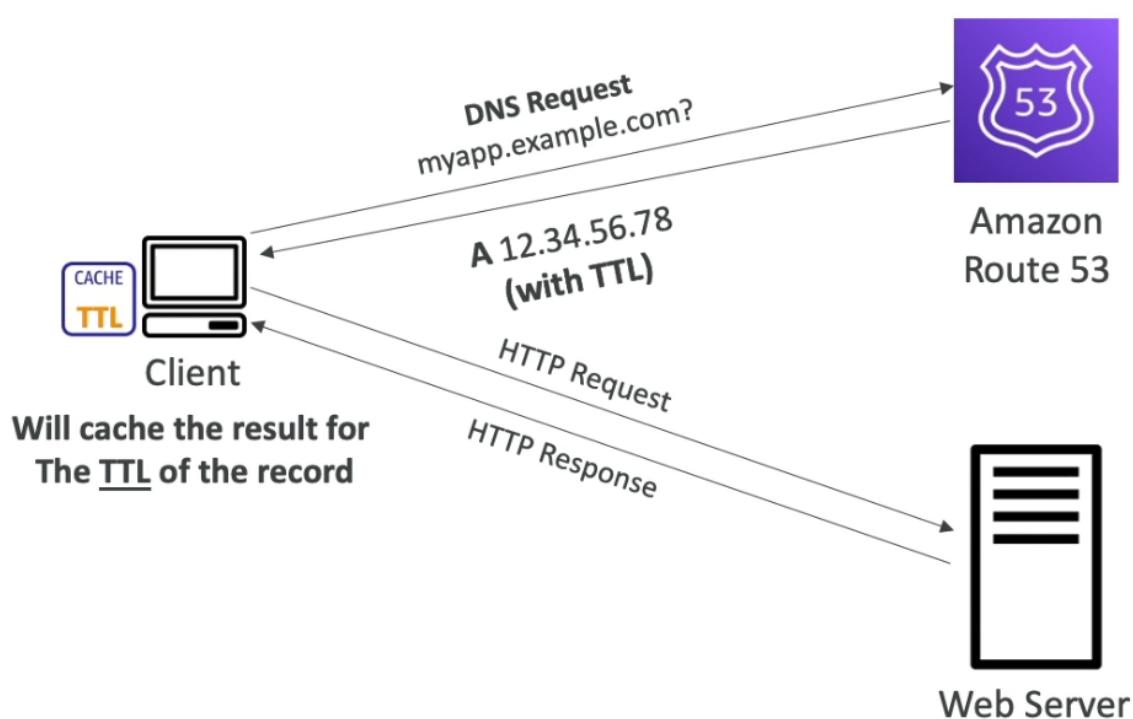
- A container for records that define how to route traffic to a domain and its subdomains
- **Public Hosted Zones** - contains records that specify how to route traffic on the Internet (public domain names)
[application1.mypublicdomain.com](#)
- **Private Hosted Zones** - contain records that specify how you route traffic within one or more VPCs (private domain names)
[application1.company.internal](#)
- You pay \$0.5 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
- **Expect for Alias records, TTL is mandatory for each DNS record**



CNAME vs Alias

- AWS Resources (Load Balancer, CloudFront...) expose an AWS hostname:
 - [Ib 1-1234.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](#))

■ ROOT DOMAIN

The screenshot shows a modal dialog with an error message: "Error occurred Bad request. (InvalidChangeBatch 400: RRSet of type CNAME with DNS name stephanetheteacher.com. is not permitted at apex in zone stephanetheteacher.com.)". Below the dialog is the Route 53 "Create record" form for a CNAME record named "blog" pointing to "demo-alb-route-53-1648128207.eu-central-1.elb.amazonaws.com". The TTL is set to 300 seconds.

■ NON ROOT DOMAIN

The screenshot shows the "Create record" form for a CNAME record named "myapp" pointing to "demo-alb-route-53-1648128207.eu-central-1.elb.amazonaws.com". The TTL is set to 300 seconds. The "Create records" button is visible at the bottom right.

- o

- Alias:

- o Point a hostname to an AWS Resource ([app.mydomain.com](#) => [blabla.amazonaws.com](#))
- o Works for ROOT DOMAIN and NON ROOT DOMAIN ([aka.mydomain.com](#))

■ ROOT DOMAIN

The screenshot shows the "Create record" form for an A alias record named "blog" pointing to "ral-1.elb.amazonaws.com". The "Route traffic to" section is set to "Alias" and "Alias to Application and CloudFront". The "Evaluate target health" option is set to "Yes". The "Create records" button is visible at the bottom right.

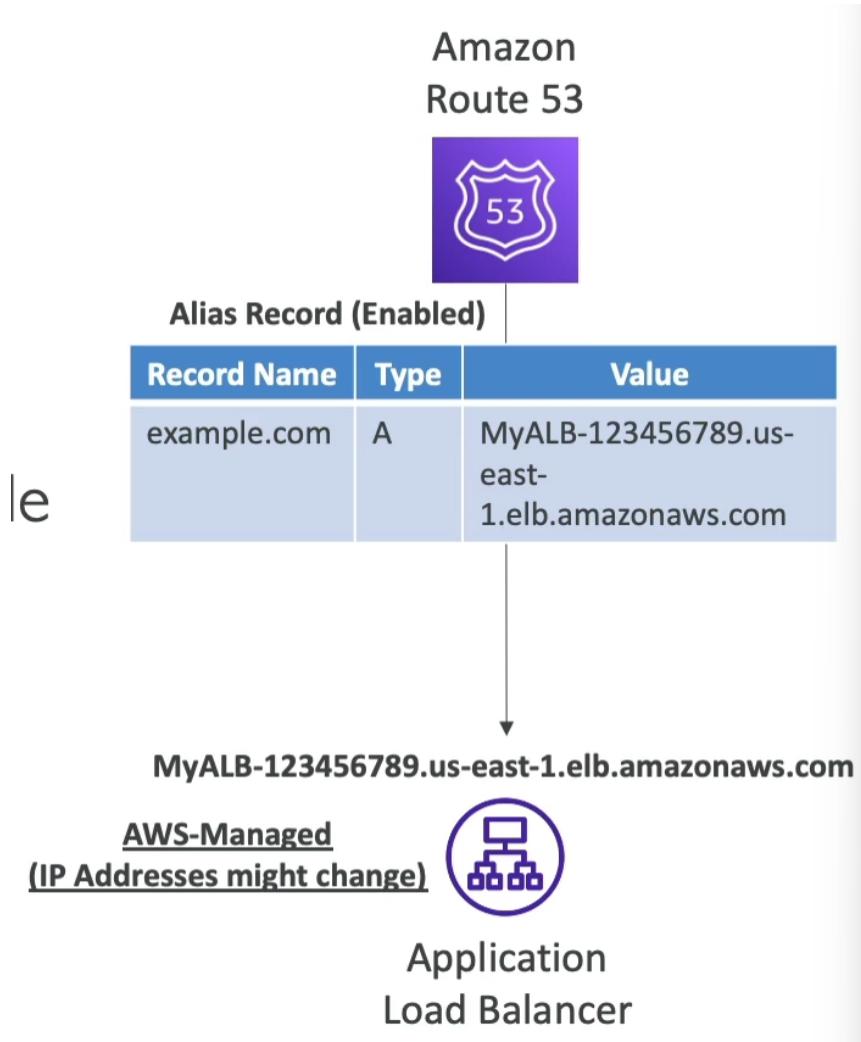
■ NON ROOT DOMAIN

The screenshot shows the "Create record" form for an A alias record named "myalias" pointing to "ral-1.elb.amazonaws.com". The "Route traffic to" section is set to "Alias" and "Alias to Application and CloudFront". The "Evaluate target health" option is set to "Yes". The "Create records" button is visible at the bottom right.

- 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**