

# AWS Essentials For Work

Tran Tuan Dat – Hong Tuan Quyen



# Welcome!

---

This course will last for **5 sessions (5 weeks)**, before going to **Cloud Practitioner Exam (CLF-C01)**

Each session will help you understand 1-2 main aspects when working with AWS infrastructure system in Runsystem's projects

Developer - perfect, BA/BrSE/PM/Tester/... with basic IT knowledge - helpful

**Learn by doing – key learning technique!**

$$Overall = \frac{\sum_{n=1}^5 (Quiz\ n) + Diligence + Final\ Project}{7}$$

Minimum score	Each Quiz	Diligence	Final Project	Overall
DEV	0	60	70	<b>70</b>
BrSE, BA	0	60	50	<b>70</b>
QA, Tester	0	100	0	<b>70</b>

Number of Quiz attempts: **2 - Average score** will be counted

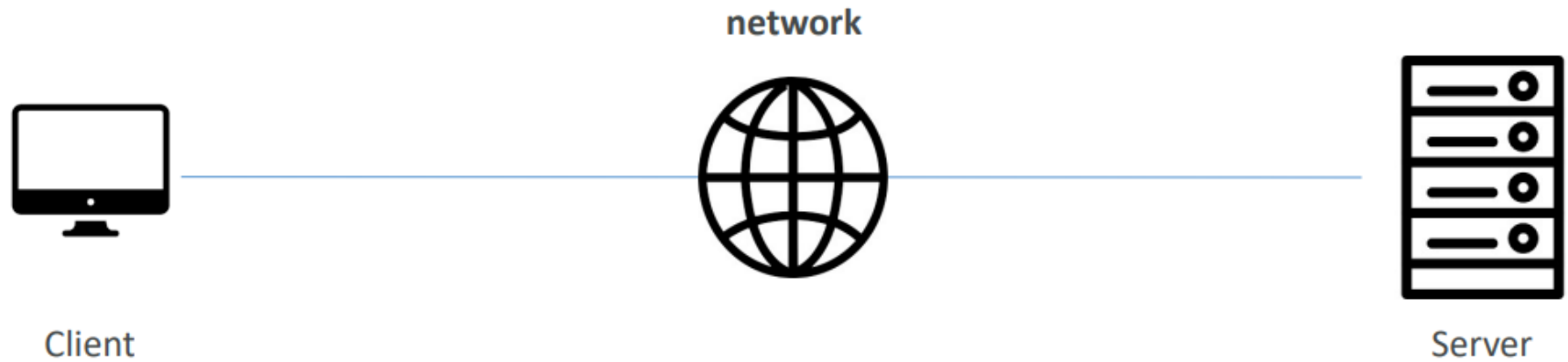
QA, Tester don't have Quiz 4,5 and Final Project => Total items to divide: 4

---

# What is Cloud Computing?

# How websites work?

---



Clients have IP addresses

Servers have IP addresses

# Like sending post mail

---

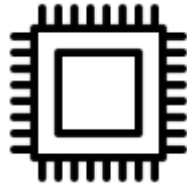


# What is a server composed of?

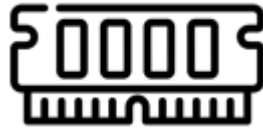
---

Compute: CPU

Memory: RAM



+



=



Storage: Data



Database: Store data in a structured way



Network: Routers, switch, DNS server



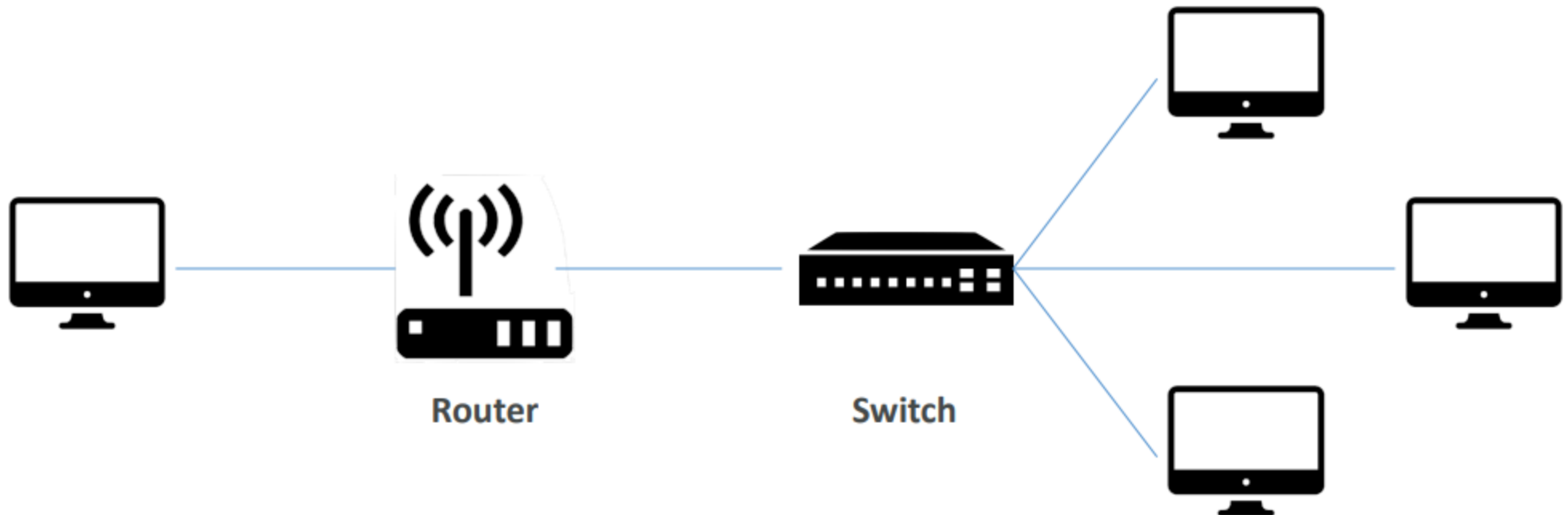
# IT Terminology

---

**Network:** cables, routers and servers connected with each other

**Router:** A networking device that forwards data packets between computer networks. They know where to send your packets on the internet

**Switch:** Takes a packet and send it to the correct server / client on your network





# Traditional infrastructure

---



Home or Garage



Office



Data center

# Problems with traditional IT approach

---

Pay for the rent for the data center

Pay for power supply, cooling, maintenance

Adding and replacing hardware takes time

Scaling is limited

Hire 24/7 team to monitor the infrastructure

Deal with disasters (earthquake, power shutdown, fire,...)

How to externalize all this?



# What is Cloud Computing?

---

Cloud Computing is the **on-demand delivery** of compute power, database storage, applications, and other IT resources

Through a cloud services playform with **pay-as-you-go-pricing**

You can **provision exactly the right type and size of computing** resources you need

You can access as many resources as you need, **almost instantly**

Simple way to access **servers, storage, databases** and a set of **application services**

Amazon Web Services owns and maintains the network-connected hardware required for these application services, while you provision and use what you need via a web application

# We are using some Cloud services

---



**Gmail**

E-mail cloud service

Pay for ONLY your emails stored (no infrastructure, etc.)



**Dropbox**

Cloud Storage Service

Originally built on AWS



**Netflix**

Built on AWS

Video on demand

# AWS Cloud Use Cases

---

AWS enables you to build sophisticated, scalable applications

Applicable to a diverse set of industries

Use cases include

Enterprise IT, Backup & Storage, Big Data analytics

Website hosting, Mobile & Social Apps

Gaming



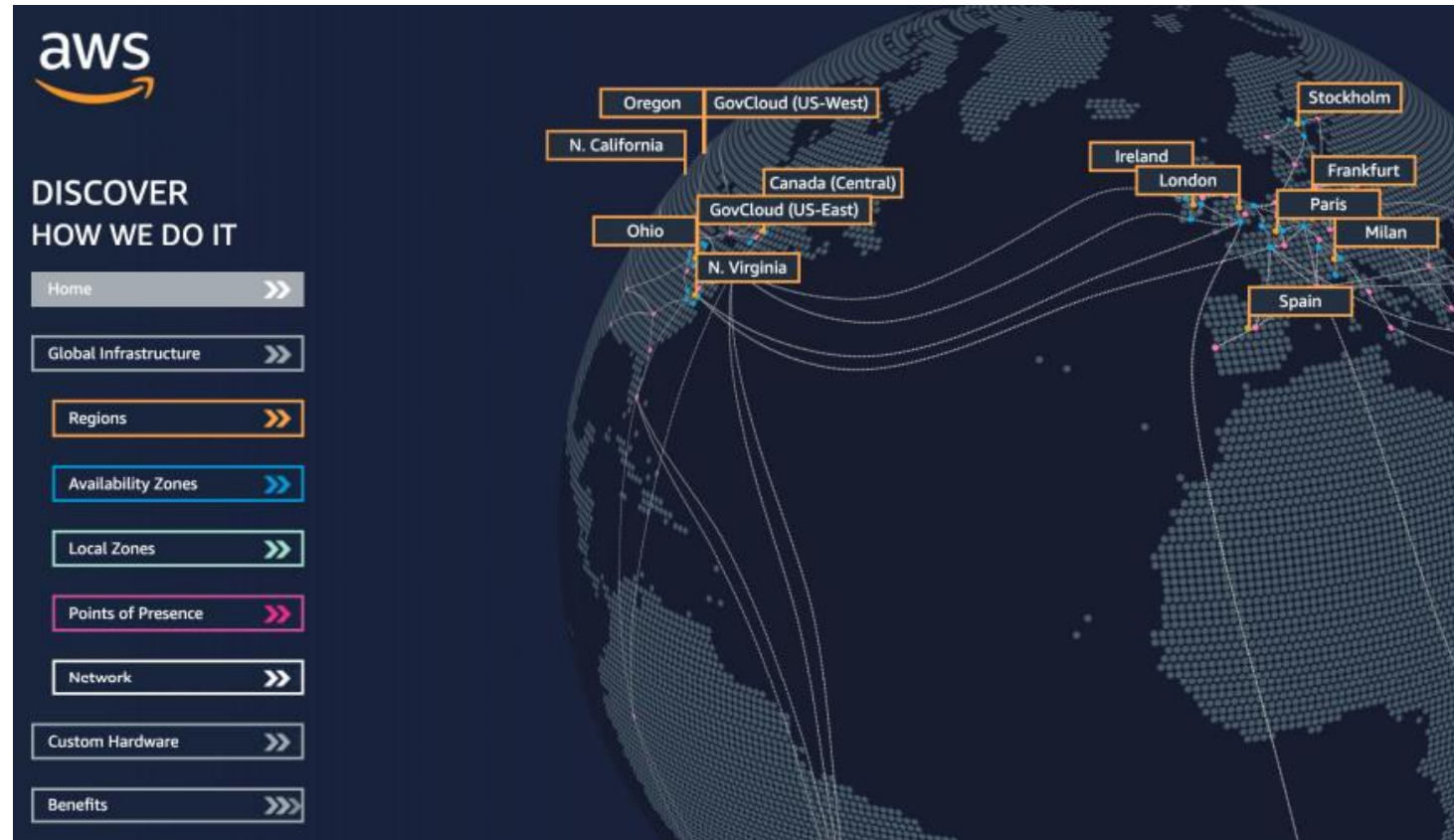
# AWS Global Infrastructure

Regions

Availability Zones

Data Centers

Edge Locations /  
Points of Presence



# 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/>

<b>US East (N. Virginia)</b>	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
<hr/>	
Africa (Cape Town)	af-south-1
<hr/>	
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
<hr/>	
Canada (Central)	ca-central-1
<hr/>	
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
<hr/>	
Middle East (Bahrain)	me-south-1
<hr/>	
South America (São Paulo)	sa-east-1

# How to choose an AWS Region?

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



**Compliance with data governance and legal requirements:** data never leaves a region without your explicit permission

**Proximity to customers:** reduced latency

**Available services within a Region:** new services and new features aren't available in every Region

**Pricing:** pricing varies region to region and is transparent in the service pricing page



# AWS Availability Zones

Each region has many availability zones (usually 3, min is 2, max is 6). Example:

ap-southeast-2a

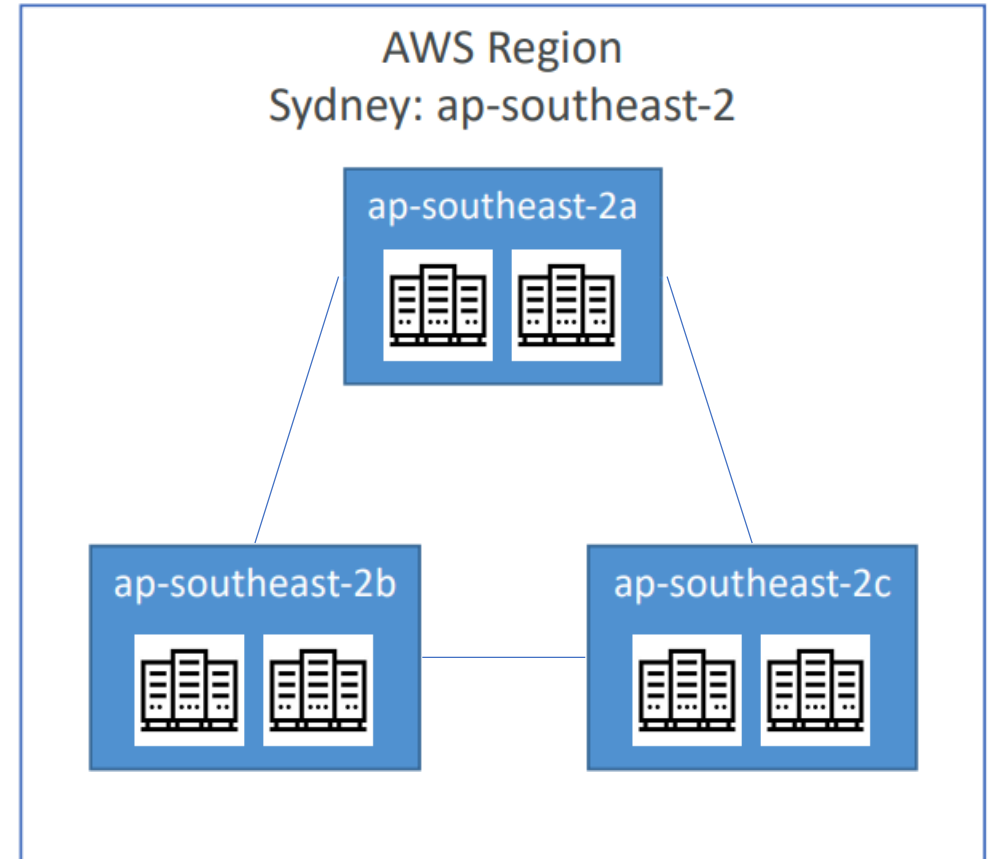
ap-southeast-2b

ap-southeast-2c

Each availability zone (AZ) is one or more discrete data centers with redundant power, networking, and connectivity

They're separate from each other, so that they're isolated from disasters

They're connected with high bandwidth, ultra-low latency networking

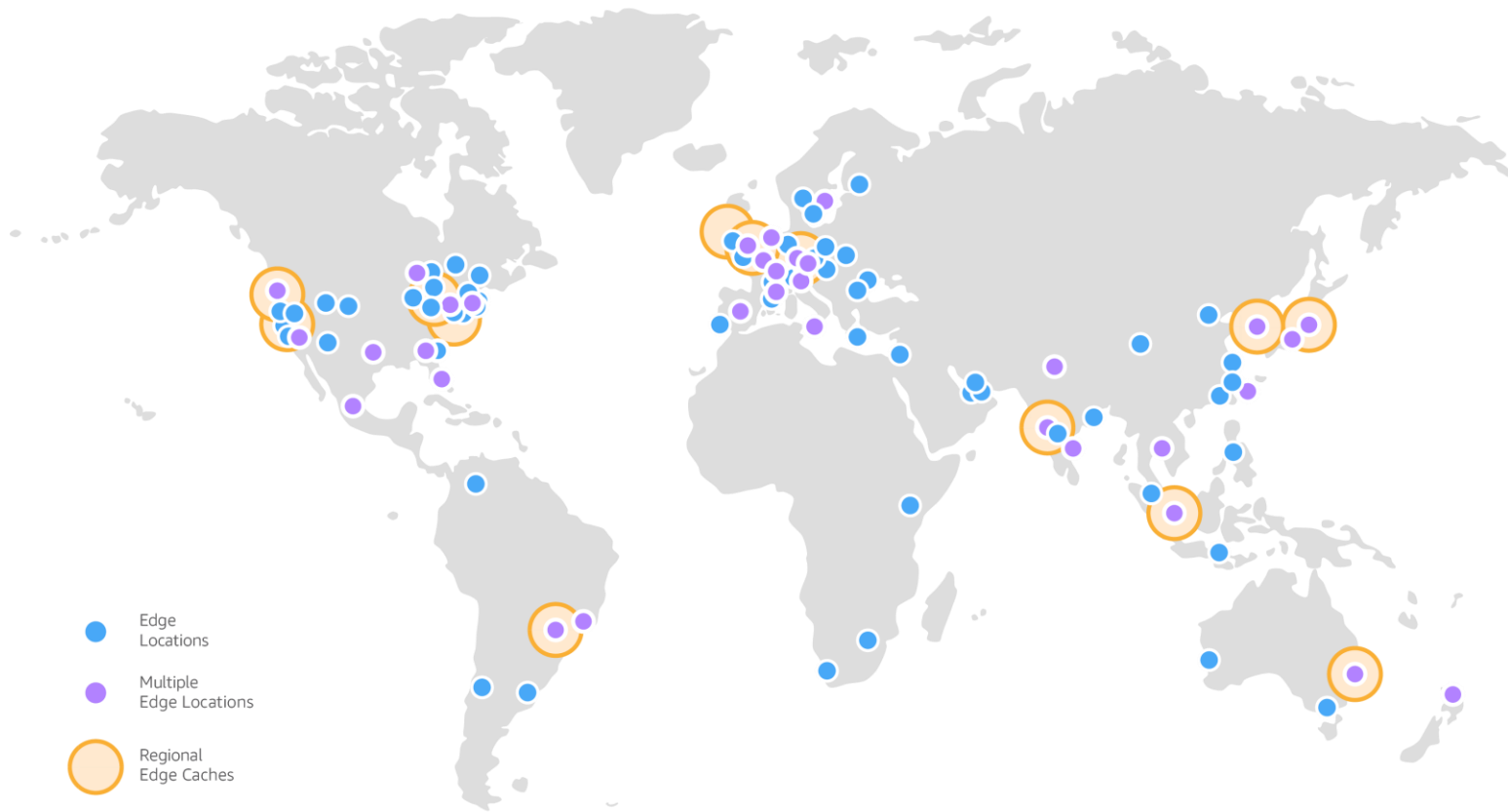


# AWS Points of Presence (Edge Locations)

---

Amazon has 310+ Points of Presence (300+ Edge Locations and 13 Regional Edge Caches)

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)
- Web Application Firewall (WAF)



## Most AWS services are Region-scoped

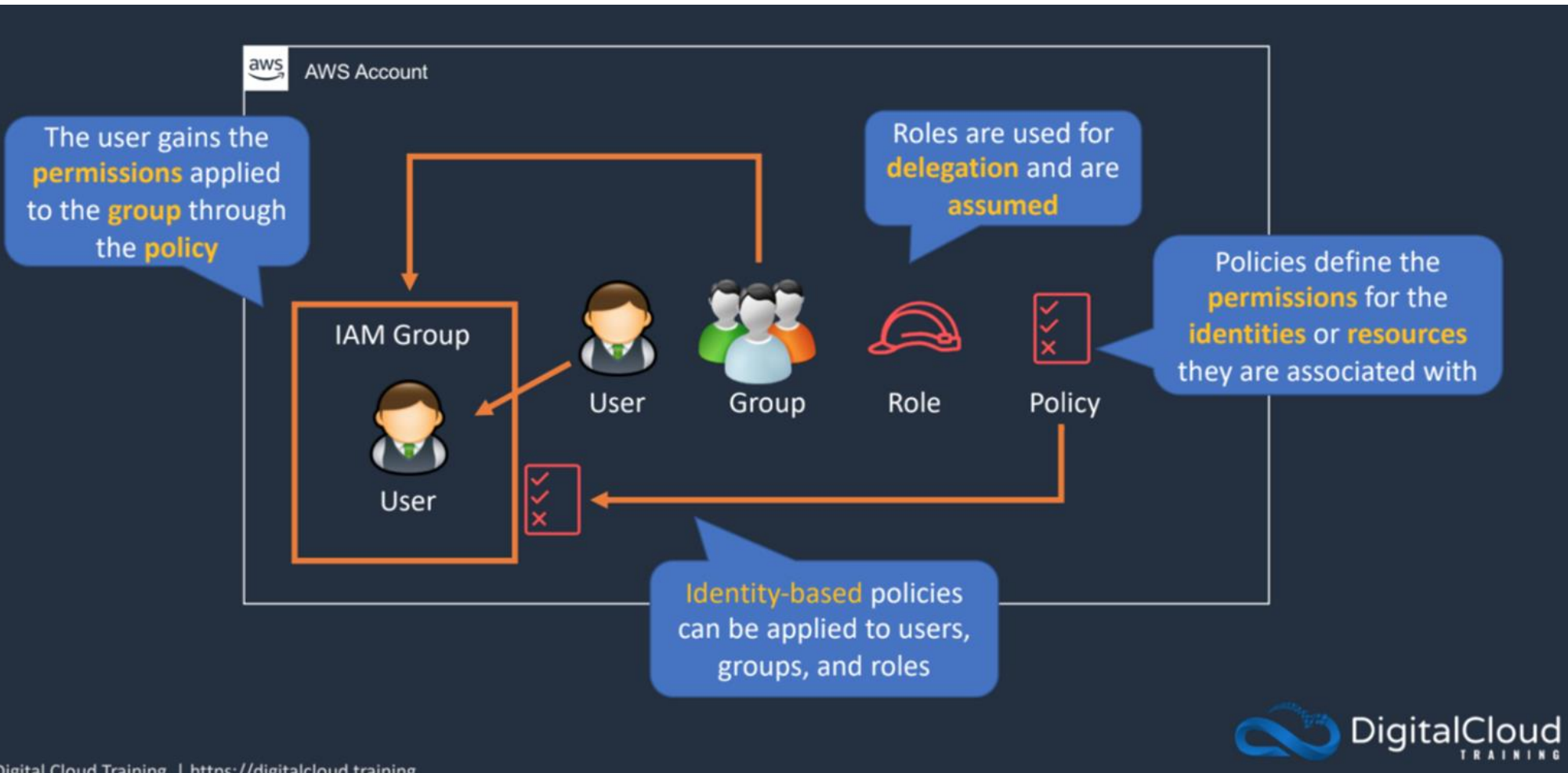
- Elastic Compute Cloud (EC2 – Infrastructure as a Service)
- Elastic Beanstalk (Platform as a Service)
- Lambda (Function as a Service)
- Rekognition (Software as a Service)

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

---

# Identity and Access Management (IAM)

# IAM Users, Groups, Policies, Roles



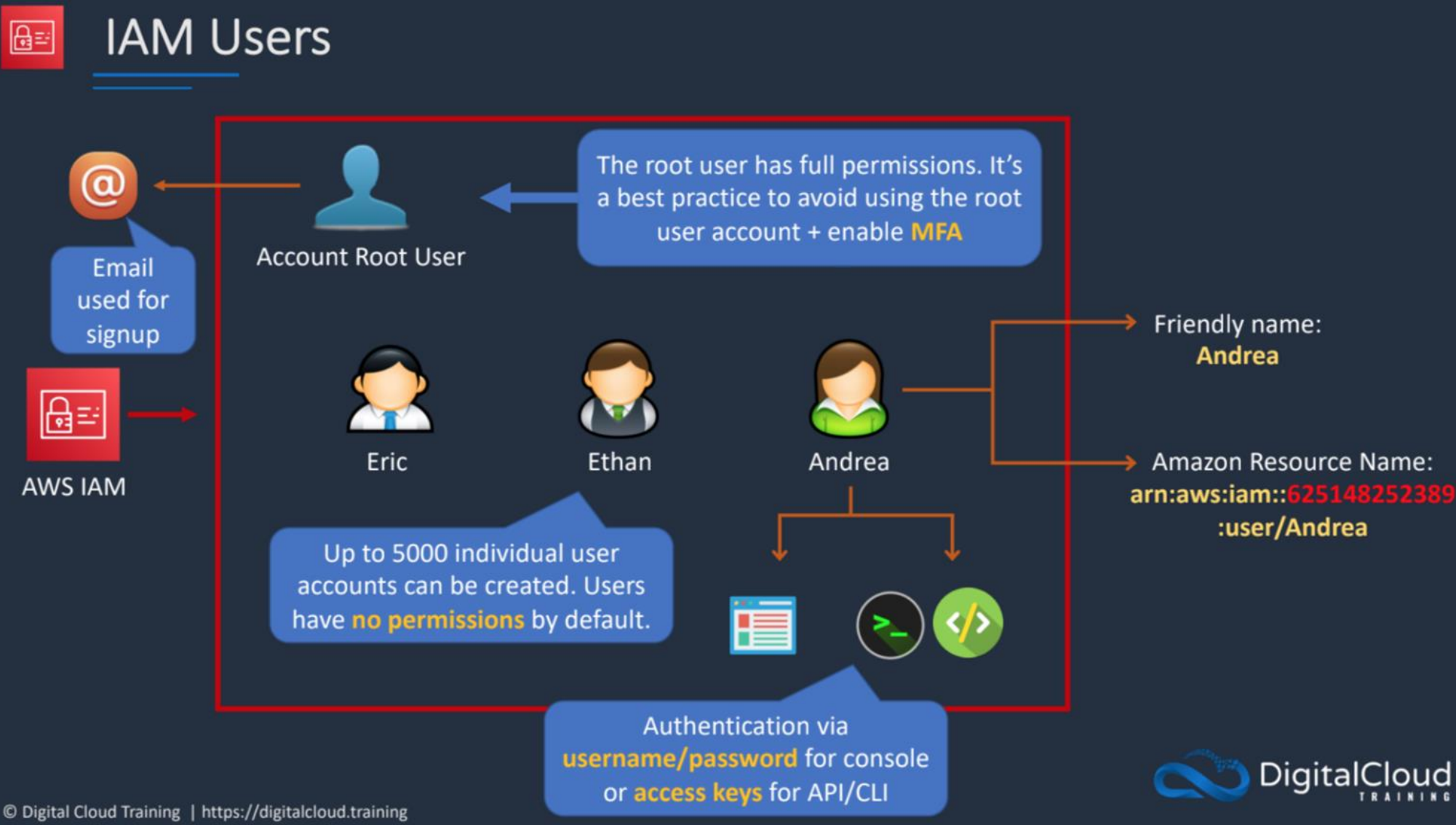
## Root user

- Full Access permissions
- Account khi đăng ký AWS

## IAM users

- Những người có thể thao tác được với AWS resources
- Mỗi user khi được tạo sẽ được cấp:
  - + Username và Password => Console
  - + Access keys => CLI, SDK (API)
- IAM user has no permissions by default

# IAM User

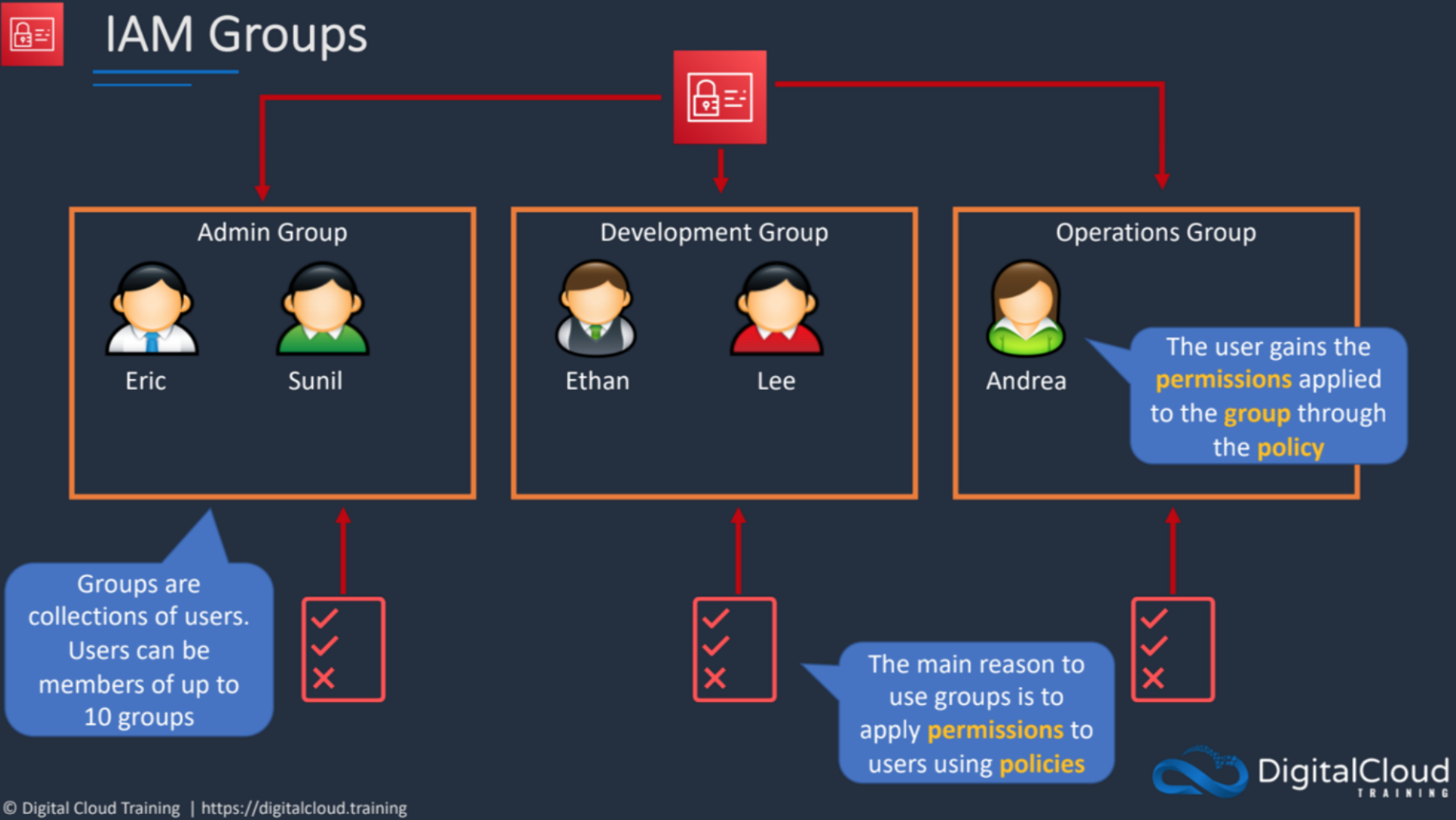


- Tập hợp các users có cùng chung các permissions
- Lợi ích
  - + Dễ quản lí permissions
  - + Dễ cấp quyền cho nhiều user cùng lúc:

Thay vì liệt kê từng user được cấp quyền thao tác cho 1 resource thì mình chỉ cần cấp quyền cho group đó là toàn bộ user trong group đó được apply
- Không được set group trong group



# IAM Group



## Root account

- Delete access keys
- Tạo admin IAM users để quản lí
- Tạo strong password
- Enable MFA

## IAM users

Khi tạo users:

- Set auto-generated password và bắt IAM user đặt lại password ở lần login đầu tiên
- Không set bất kì permissions nào cho đến khi user confirm login và đổi password thành công
- Nên set permissions với least privileges
- Đưa IAM user vào IAM group

# IAM Policy

---

- Là 1 file JSON define các permissions, mà ở đó các resources được chỉ định trong file được hoặc không được phép thực hiện actions nào đó
- Sử dụng AWS Policies hoặc User-defined Policies

VD: Policy AdministratorAccess

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": "*",  
7       "Resource": "*"  
8     }  
9   ]  
10 }
```

# IAM Policy

---

VD: S3 Read Only permissions

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": [
7          "s3:Get*",
8          "s3:List*",
9          "s3-object-lambda:Get*",
10         "s3-object-lambda:List*"
11        ],
12        "Resource": "arn:aws:s3:::bucket-quyen"
13      }
14    ]
15  }
```

# IAM Policy - Types

---

## Identity-based policy

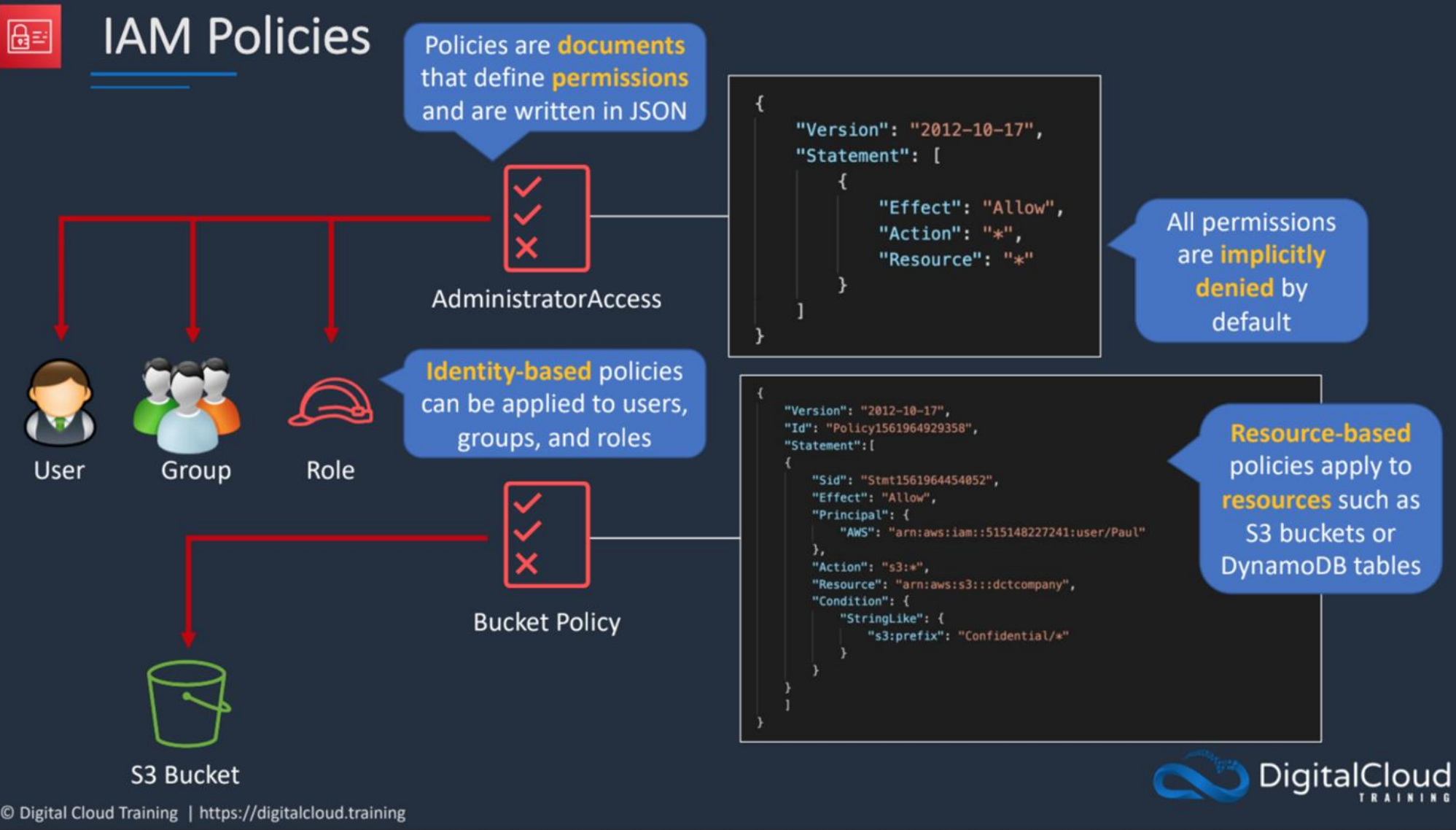
+ Allow / Deny Users / Groups / AWS Resources thực hiện actions đối với AWS resource nào đó

## Resource-based policy

+ Allow / Deny Users / Groups / AWS Resources **được** thực hiện actions nào đó lên resource đang được set policy

+ Phải có field Principal

# IAM Policy



# IAM Role

---

- Tập hợp các IAM Policy
- Assign cho AWS resources. VD: EC2
- Lưu ý: Ko thể assign role cho IAM users / groups mà thay vào đó là dùng IAM Policy

# Best practices

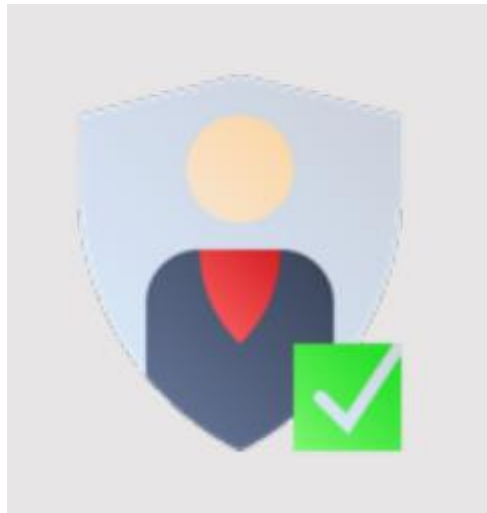
---

- Tránh viết quá nhiều permissions trong 1 IAM Policy mà thay vào đó, chia nhỏ ra và assign vào IAM Role, vì:
  - + Dễ quản lí
  - + Dễ tái sử dụng
- Grant least permissions: Cấp cho IAM users/groups hoặc aws resources (VD: ec2) permissions đủ dùng để thực hiện task, không được cấp permissions không liên quan đến task mà resources đó cần thực hiện



# Password Policy / MFA

---



# Password Policy

---

## Modify password policy

A password policy is a set of rules that define complexity requirements and mandatory rotation periods for your IAM users' passwords. [Learn more](#)

### Select your account password policy requirements:

☒ Enforce minimum password length

6

characters

☒ Require at least one uppercase letter from Latin alphabet (A-Z)

☒ Require at least one lowercase letter from Latin alphabet (a-z)

☒ Require at least one number

☒ Require at least one non-alphanumeric character (! @ # \$ % ^ & \* ( ) \_ + - = [ ] { } | ' )

☐ Enable password expiration

☐ Password expiration requires administrator reset

☒ Allow users to change their own password

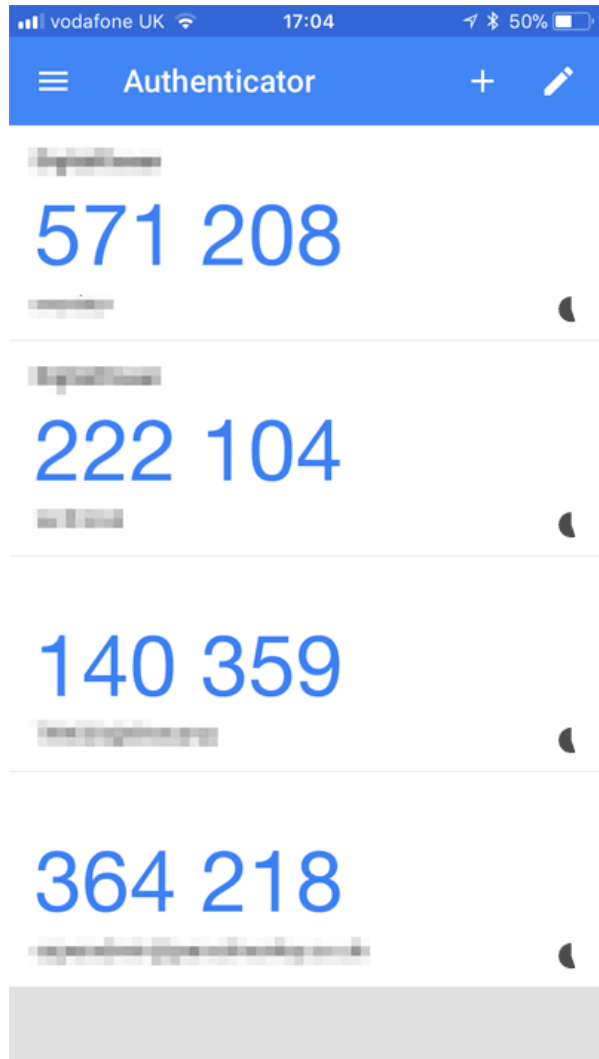
☐ Prevent password reuse

# Password Policy

---

- Ko apply cho root user và access key. Nếu password hết hạn thì dùng access key vẫn có thể access được
  - Hiệu lực
    - + Lần change password tiếp theo. VD: Thay đổi pass length hoặc thêm 1 số kí tự required, password của user sẽ apply policy mới này trong lần change pass tiếp theo
    - + Ngay lập tức. VD: Thay đổi expiration, nếu password nào expire thì sẽ apply cho lần sign in tới
  - Ko có "**lockout policy**": Ko thể khoá user sign in sau khi user nhập sai password
- => Sau khi change, nó sẽ apply cho toàn bộ IAM users

# MFA (Multi-factor Authentication)



Google Authenticator



U2F security key

# MFA (Multi-factor Authentication)

---



## Multi-factor Authentication

---

Please enter an MFA code to complete sign-in.

**MFA Code:**

**Submit**

[Cancel](#)





<https://aws.amazon.com/free>

# Billing - Price Calculator

---



<https://calculator.aws/>



# AWS Budget

---



---

# Virtual Private Cloud (VPC)

# VPC & Subnet

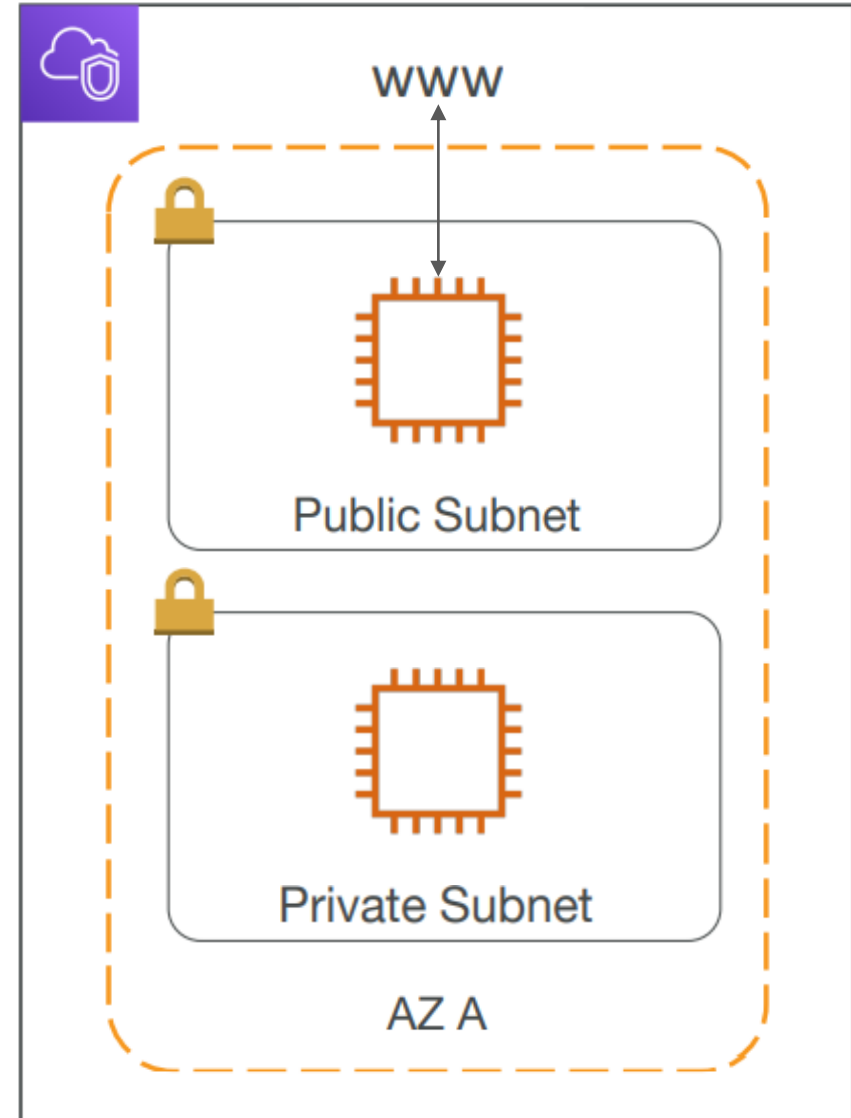
**VPC - Virtual Private Cloud:** private network to deploy your resources (regional resource)

**Subnets** allow you to partition your network inside your VPC (Availability Zone resource)

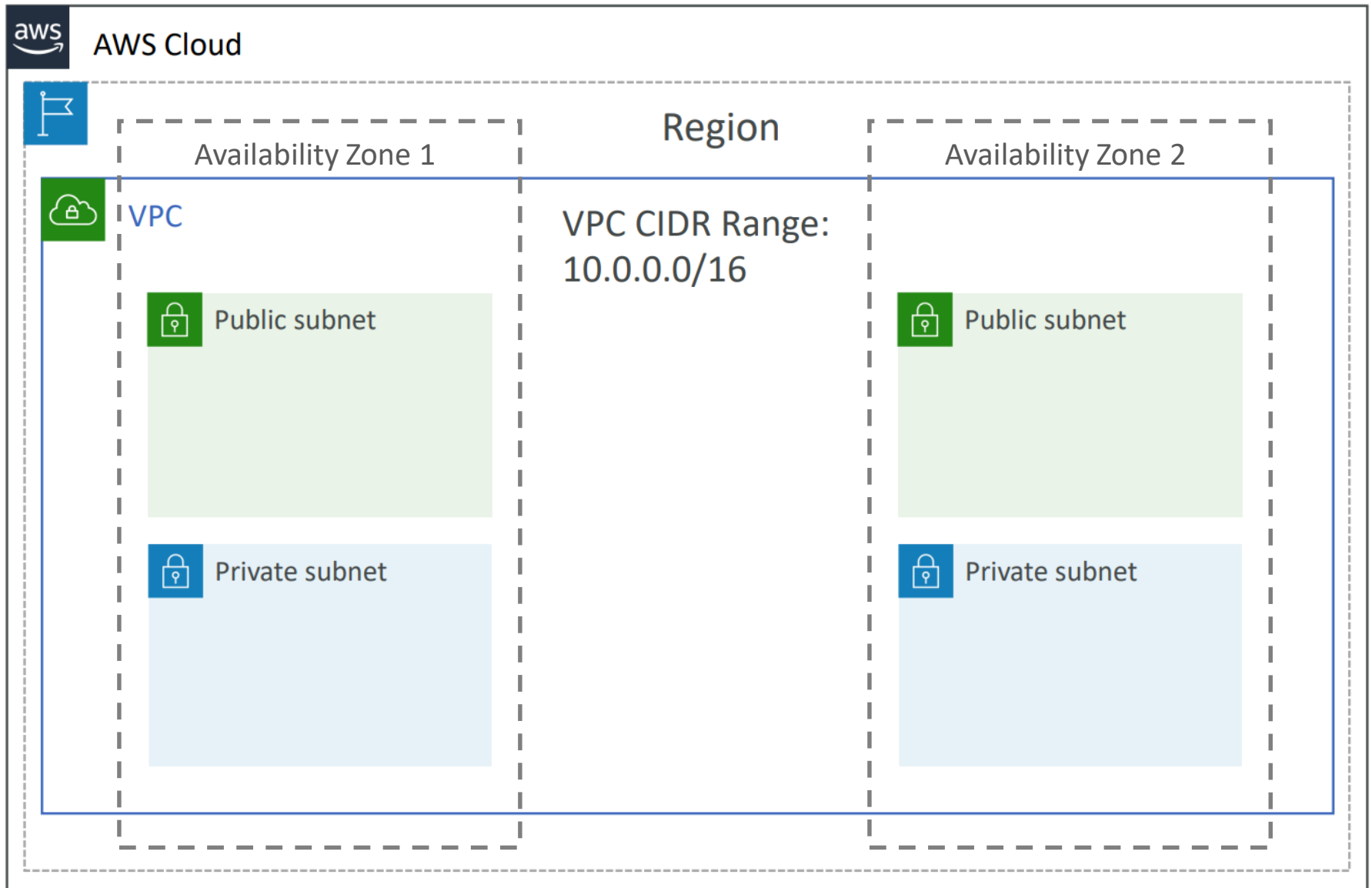
A **public subnet** is a subnet that is accessible from the internet

A **private subnet** is a subnet that is not accessible from the internet

To define access to the internet and between subnets, we use **Route Tables**



# VPC Diagram

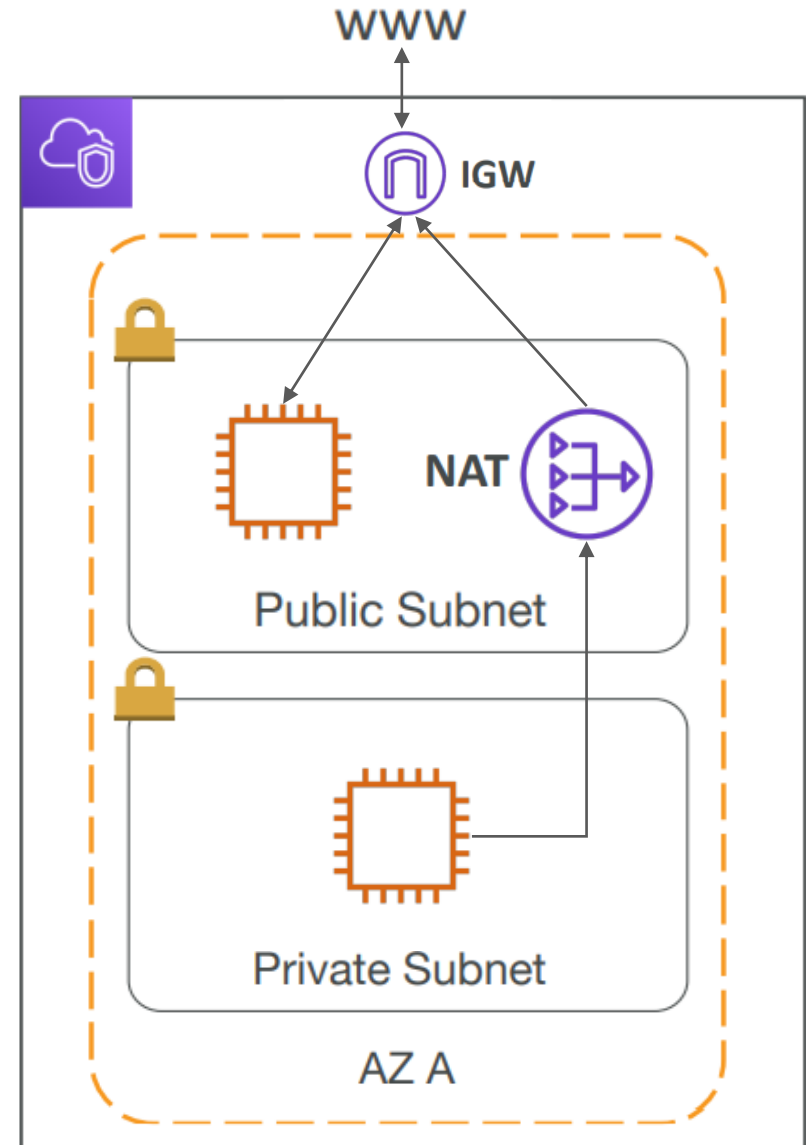


# Internet Gateway & NAT Gateways

**Internet Gateways** helps our VPC instances connect with the internet

Public Subnets have a route to the internet gateway

**NAT Gateways** (AWS-managed) & **NAT Instances** (self-managed) allow your instances in your **Private Subnets** to access the internet while remaining private



# Security Groups & Network Access Control List (NACL)

## NACL (Network ACL)

A firewall which controls traffic from and to subnet

Can have ALLOW and DENY rules

Are attached at the **Subnet** level

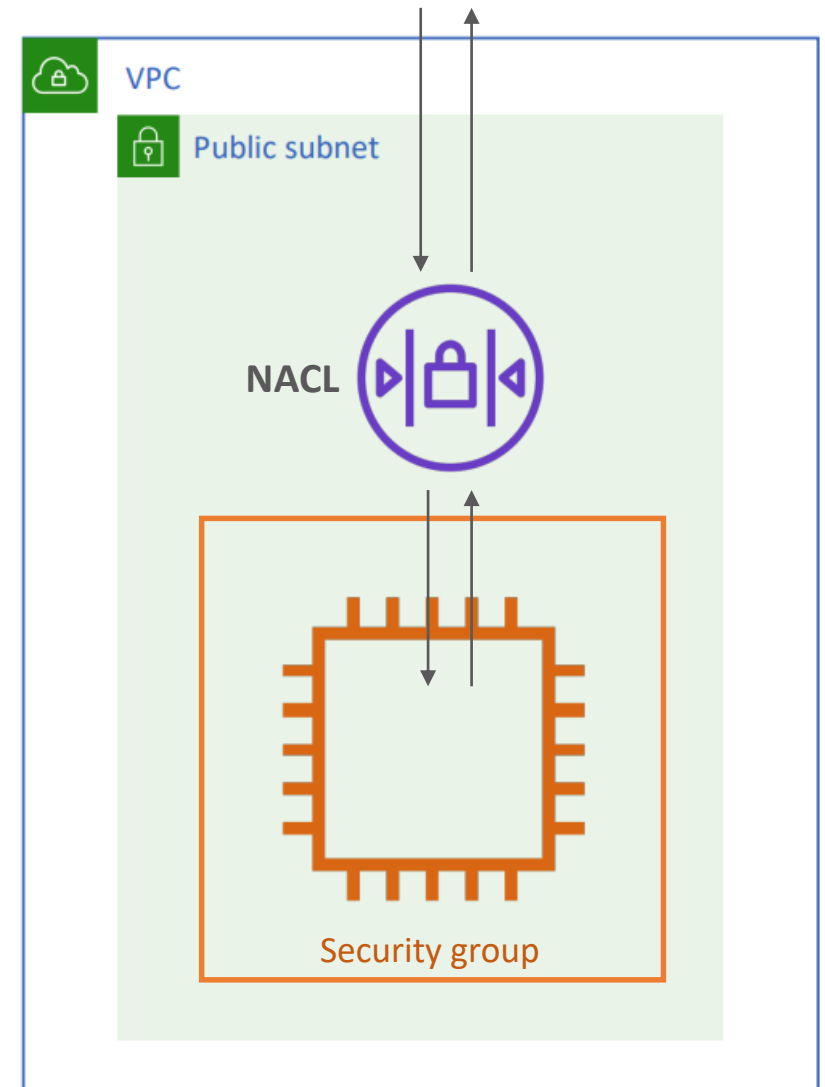
Rules only include IP addresses

## Security Groups

A firewall that controls traffic to and from **an ENI / an EC2 Instance**

Can have only ALLOW rules

Rules include IP addresses and other security groups



# Understanding CIDR – IPv4

**Classless Inter-Domain Routing** – a method for allocating IP addresses

Used in **Security Groups** rules and AWS networking in general

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

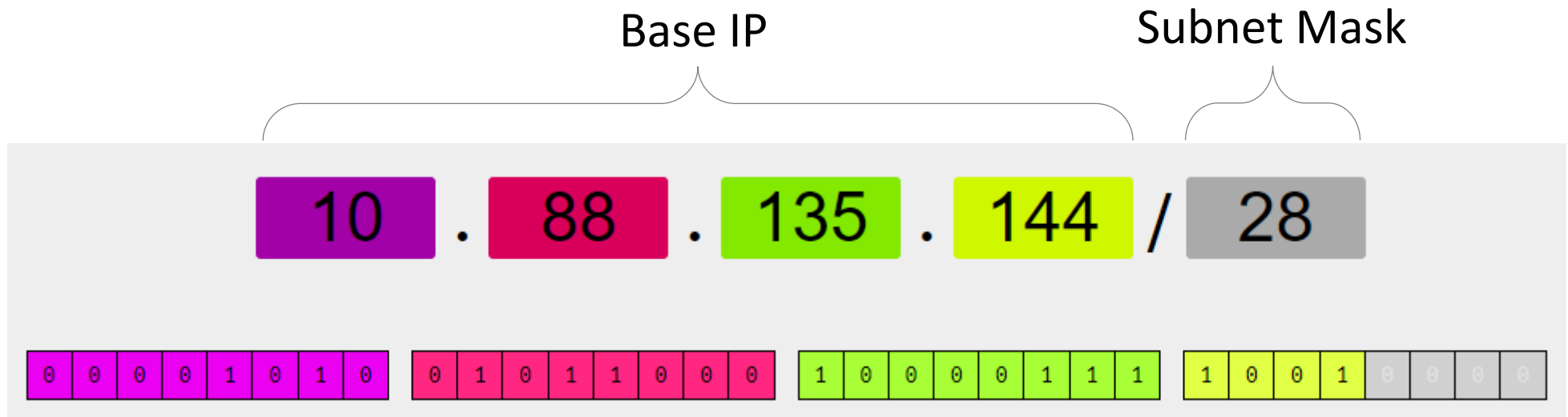
They help to define an IP address range:

We've seen xx.xx.xx.xx/32 → one IP

We've seen 0.0.0.0/0 → all IPs

But we can define: 192.168.0.0/26 → 192.168.0.0 – 192.168.0.63 (64 IP addresses)

# Understanding CIDR – IPv4



192.168.0.0 /32 → allows for **1** IP ( $2^0$ )

————→ 192.168.0.0

192.168.0.0 /31 → allows for **2** IP ( $2^1$ )

————→ 192.168.0.0 → 192.168.0.1

192.168.0.0 /30 → allows for **4** IP ( $2^2$ )

————→ 192.168.0.0 → 192.168.0.3

192.168.0.0 /24 → allows for **256** IP ( $2^8$ )

————→ 192.168.0.0 → 192.168.0.255

192.168.0.0 /16 → allows for **65,536** IP ( $2^{16}$ )

————→ 192.168.0.0 → 192.168.255.255

0.0.0.0 /0 → allows for **All** IPs

————→ 0.0.0.0 → 255.255.255.255



# Subnet CIDR

---

AWS reserves **5 IP addresses (first 4 & last 1)** in each subnet

These 5 IP addresses are not available for use and can't be assigned to an EC2 instance

Example: if CIDR block 10.0.0.0/24, then reserved IP addresses are:

- 10.0.0.0 – Network Address

- 10.0.0.1 – reserved by AWS for the VPC router

- 10.0.0.2 – reserved by AWS for mapping to Amazon-provided DNS

- 10.0.0.3 – reserved by AWS for future use

- 10.0.0.255 – Network Broadcast Address

# Public vs Private IP (IPv4)

---

The Internet Assigned Numbers Authority (IANA) established certain blocks of IPv4 addresses for the use of private (LAN) and public (Internet) addresses

**Private IP** can only allow certain values

- 10.0.0.0 – 10.255.255.255 (10.0.0.0/8) ← in big networks
- 172.16.0.0 – 172.31.255.255 (172.16.0.0/12) ← AWS default VPC range
- 192.168.0.0 – 192.168.255.255 (192.168.0.0/16) ← home networks

All the rest of the IP addresses on the Internet are **Public**

---

# Elastic Compute Cloud (EC2)

# Amazon EC2

---

EC2 is one of the most popular of AWS' offering

EC2 = Elastic Compute Cloud = Infrastructure as a Service

It mainly consists in the capability of:

- Renting virtual machines (EC2)

- Storing data on virtual drives (EBS)

- Distributing load across machines (ELB)

- Scaling the services using an auto-scaling group (ASG)

Knowing EC2 is **fundamental** to understand how the Cloud works

# EC2 sizing & configuration options

---

Operating System (**OS**): Linux, Windows or Mac OS

How much compute power & cores (**CPU**)

How much random-access memory (**RAM**)

How much storage space

- Network-attached (**EBS & EFS**)

- Hardware (**EC2 Instance Store**)

Network card: speed of the card, Public IP address

Firewall rules: **security group**

Bootstrap script (configure at first launch): EC2 User Data

# EC2 User Data

---

It is possible to bootstrap our instances using an **EC2 User data** script  
**bootstrapping** means launching commands when a machine starts

That script is **only run once** at the instance **first start**

EC2 user data is used to automate boot tasks such as:

- Installing updates

- Installing software

- Downloading common files from the internet

- Anything you can think of

The EC2 User Data Script runs with the root user

# Hands-On: Launching an EC2 Instance running Linux

---

We'll be launching our first virtual server using the AWS Console

We'll get a first high-level approach to the various parameters

We'll see that our web server is launched using EC2 user data

We'll learn how to start / stop / terminate our instance

# EC2 Instances Types - Overview

---

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

AWS has the following naming convention:

m5.2xlarge

m: instance class

5: generation (AWS improves them over time)

2xlarge: size within the instance class

General Purpose

Compute Optimized

Memory Optimized

Accelerated Computing

Storage Optimized

Instance Features

Measuring Instance  
Performance



# EC2 instance types: example

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

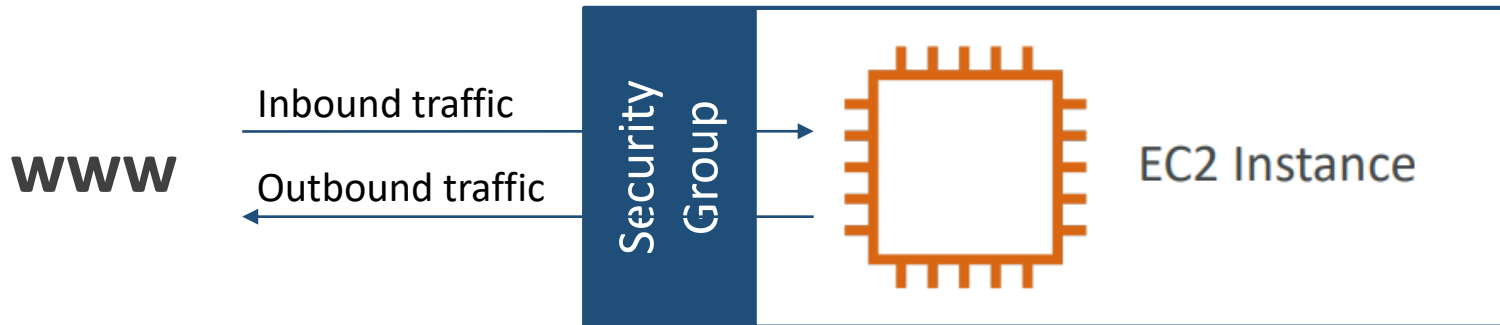
**t2.micro** is part of the AWS free tier  
(up to **750 hours** per month during the **first year**)

EC2 instances info: <https://instances.vantage.sh/>

# Introduction to Security Groups

---

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



Security groups only contain **allow** rules

Security groups rules can reference by IP or by security group

# Security Groups Deeper Dive

Security groups are acting as a “firewall” on EC2 instances

They regulate:

- Access to Ports

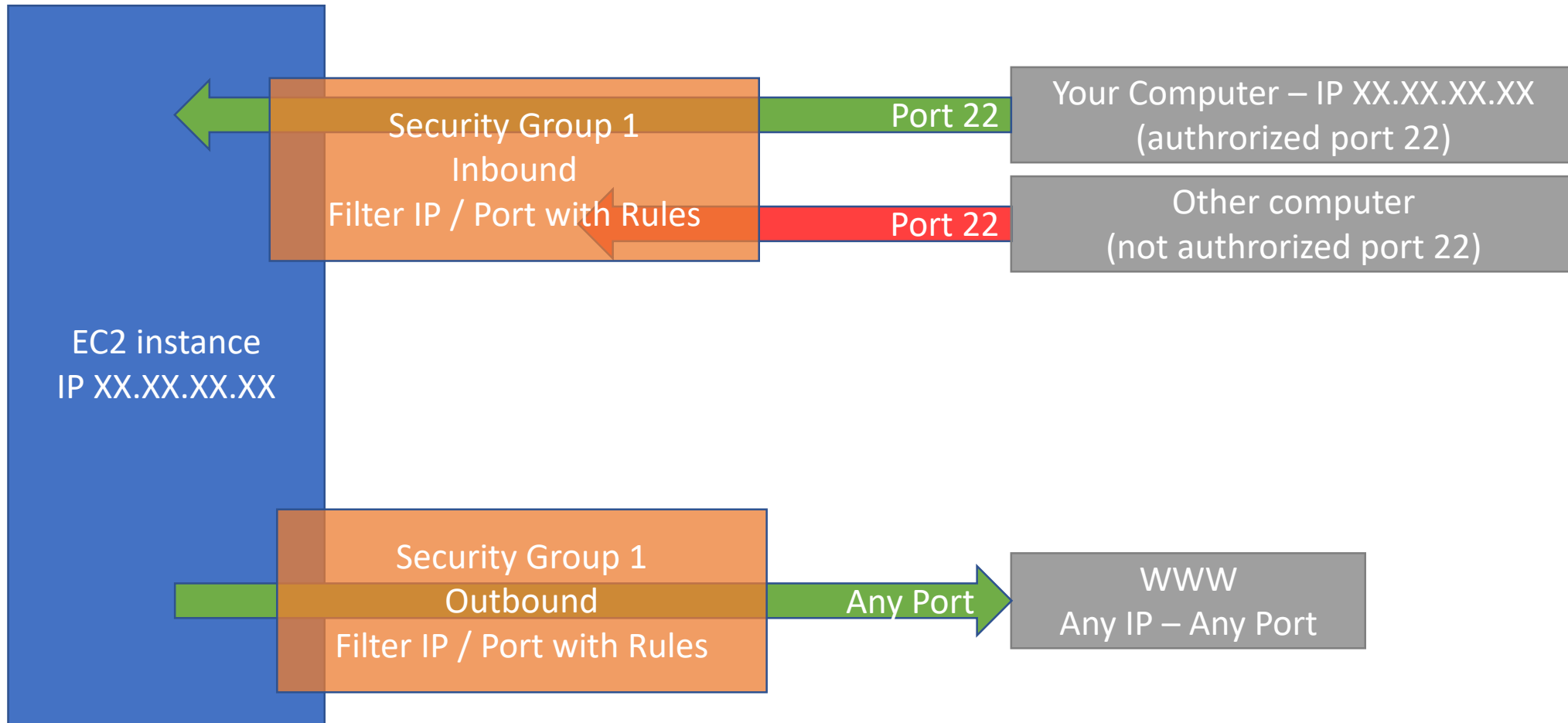
- Authorised IP ranges – IPv4 and IPv6

- Control of inbound network (from other to the instance)

- Control of outbound network (from the instance to other)

Type ⓘ	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 Diagram



# Security Groups Good to know

---

Can be attached to multiple instances

Locked down to a region / VPC combination

Lives “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

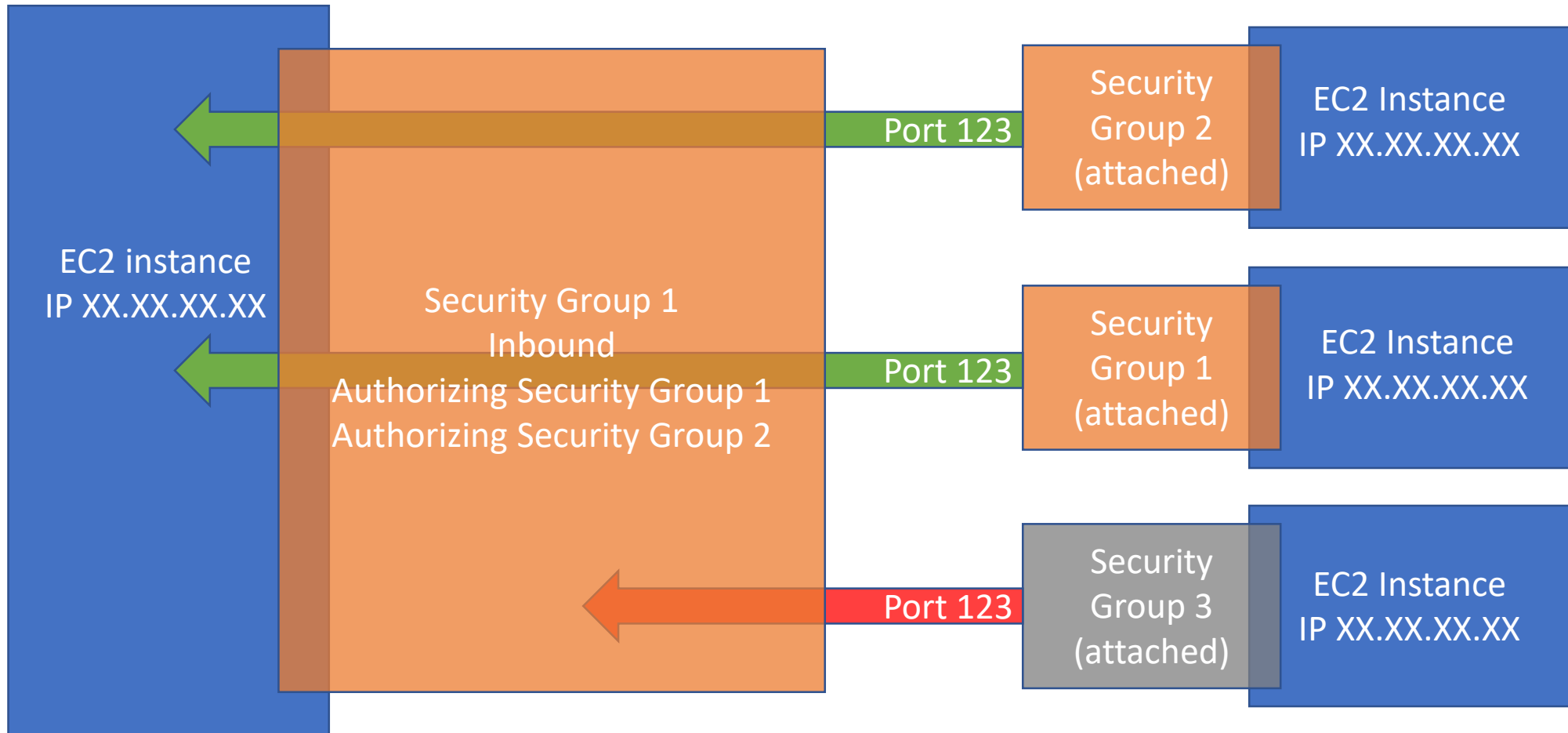
If your application is not accessible (timeout), then it’s a security group issue

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 **authorized** by default

# Referencing other security groups Diagram



# Classic Ports to know

---

22 = SSH (Secure Shell) – log into a Linux instance

21 = FTP (File Transfer Protocol) – upload files into a file share

22 = SFTP (Secure File Transfer Protocol) – upload files using SSH

80 = HTTP – access unsecured websites

443 = HTTPS – access secured websites

3389 = RDP (Remote Desktop Protocol) – log into a Windows instance

# SSH Summary Table

---

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

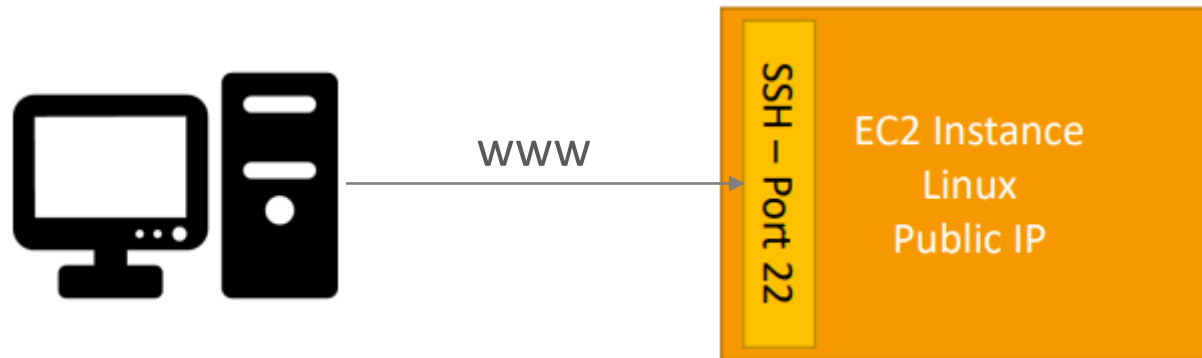


# How to SSH into your EC2 Instance – Windows

---

We'll learn how to SSH into your EC2 instance using [Windows](#)

SSH is one of the most important function which allows you to control a remote machine, all using the command line



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

# EC2 Instances Purchasing Options

---

**On-Demand Instances:** short workload, predictable pricing

No up-front payment, no long-term commitment, for short-term and un-interrupted workloads

**Reserved:** 1 year **or** 3 years = +++ discount, up to **72%**

**Reserved Instances:** long, steady-state workloads (database)

**Convertible Reserved Instances:** long workloads with flexible instances

**Scheduled Reserved Instances:** example – every Thursday between 3 and 6 pm

**Saving Plans:** **new**, similar to Reserved, based on committed **spend**

**Spot Instances:** short workloads, cheapest, can lose instances (less reliable)

**Dedicated Hosts:** book an entire physical server, control instance placement

Need **compliance requirements**, allows you to use your **server-bound software licences**

**Dedicated Instances:** no other customers will share your hardware

# 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

**Saving Plans:** planning ahead and deposit an amount of money that you commit to stay, if you need more rooms or bigger rooms and the deposit runs out, come back to on-demand

**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 and install more stuff

---

# Storage

# What is 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

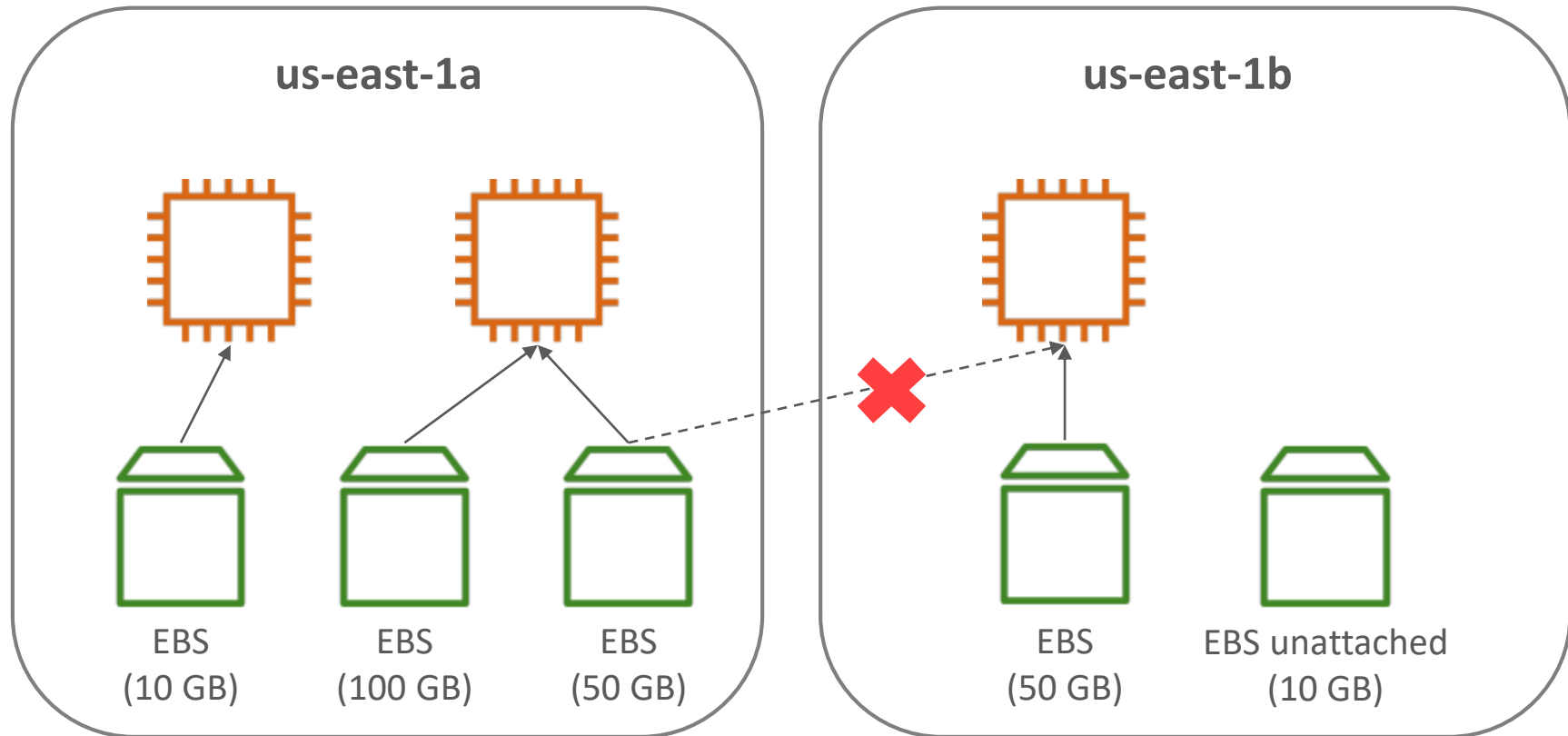
One EBS can only be mounted to one instance at a time

They are bound to a specific availability zone

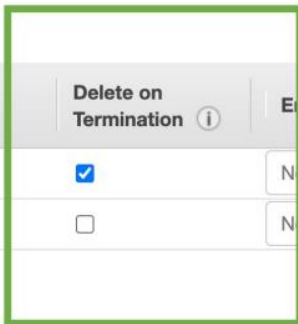
Analogy: think of them as a **network USB stick**

**Free tier:** 30 GB of free EBS storage of type General Purpose (SSD) or Magnetic per month

# EBS Volume - Example



# EBS – Delete on Termination attribute



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

Add New Volume

Controls the EBS behaviour when an EC2 instance terminates

By default, the root EBS volume is deleted (attribute enabled)

By default, any other attached EBS volume is not deleted (attribute disabled)

This can be controlled by the AWS console / AWS CLI

Use case: preserve root volume when instance is terminated

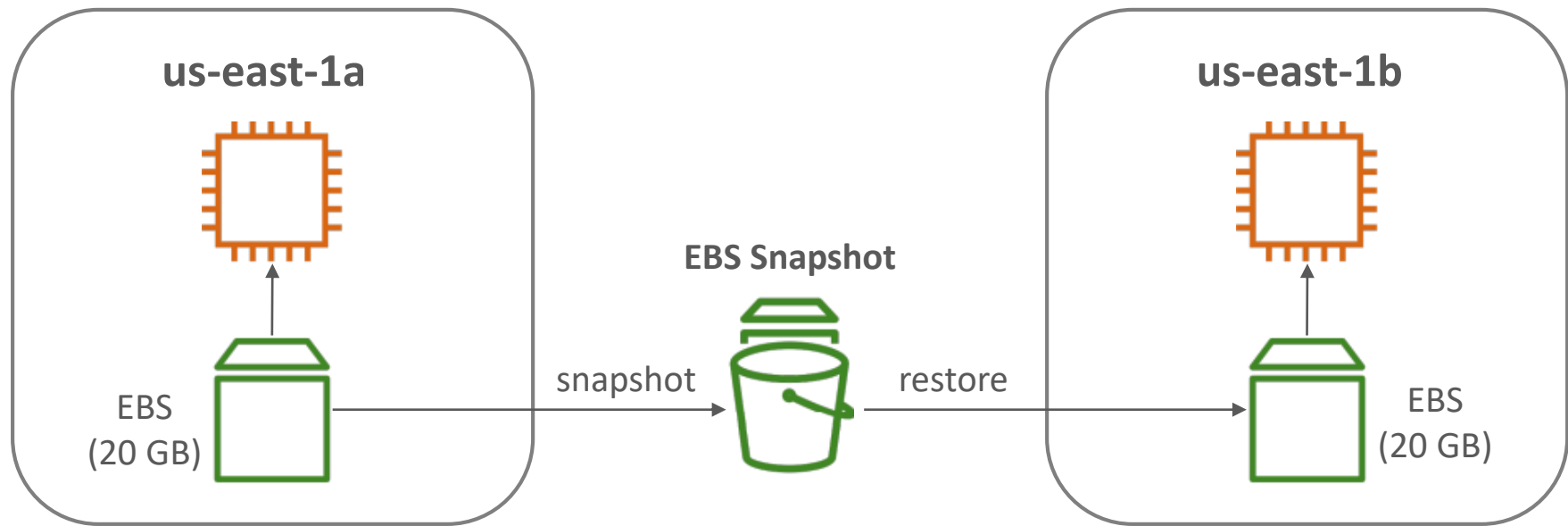
# EBS Snapshots

---

Make a backup (snapshot) of your EBS volume at a point in time

Not necessary to detach volume to do snapshot, but recommended

Can copy snapshots across AZ or Region





# 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)

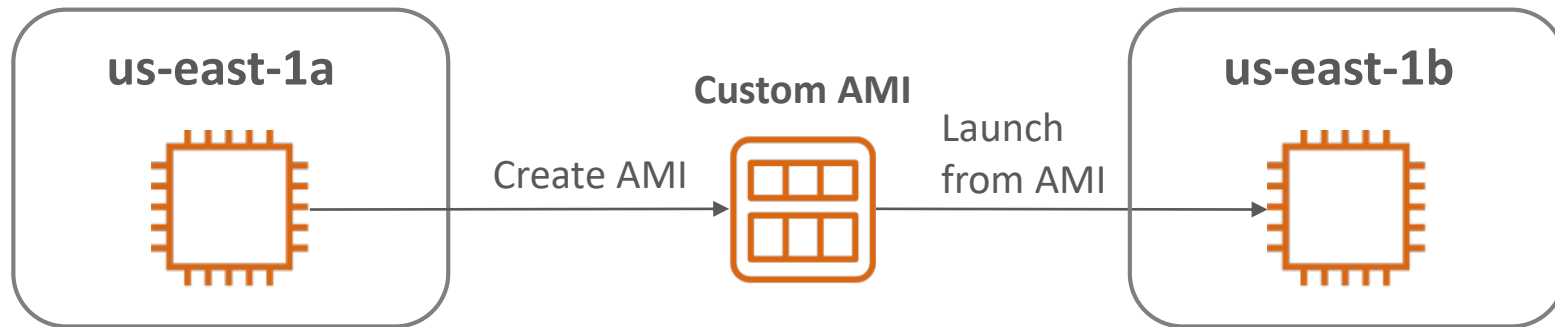
---

Start an EC2 instance and customize it

Stop the instance (for data integrity) – **not obligatory but recommended**

Build an AMI – this will also create EBS snapshots

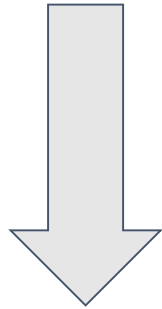
Launch instances from other AMIs



# AWS S3, EFS, CloudWatch

---

**Storage**

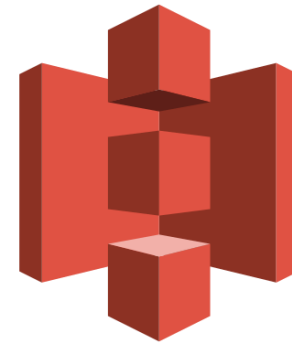
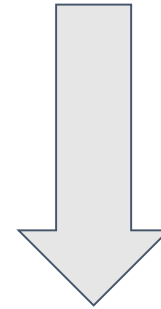


**S3**



**EFS**

**Monitor**



**CloudWatch**



Simple Storage Service

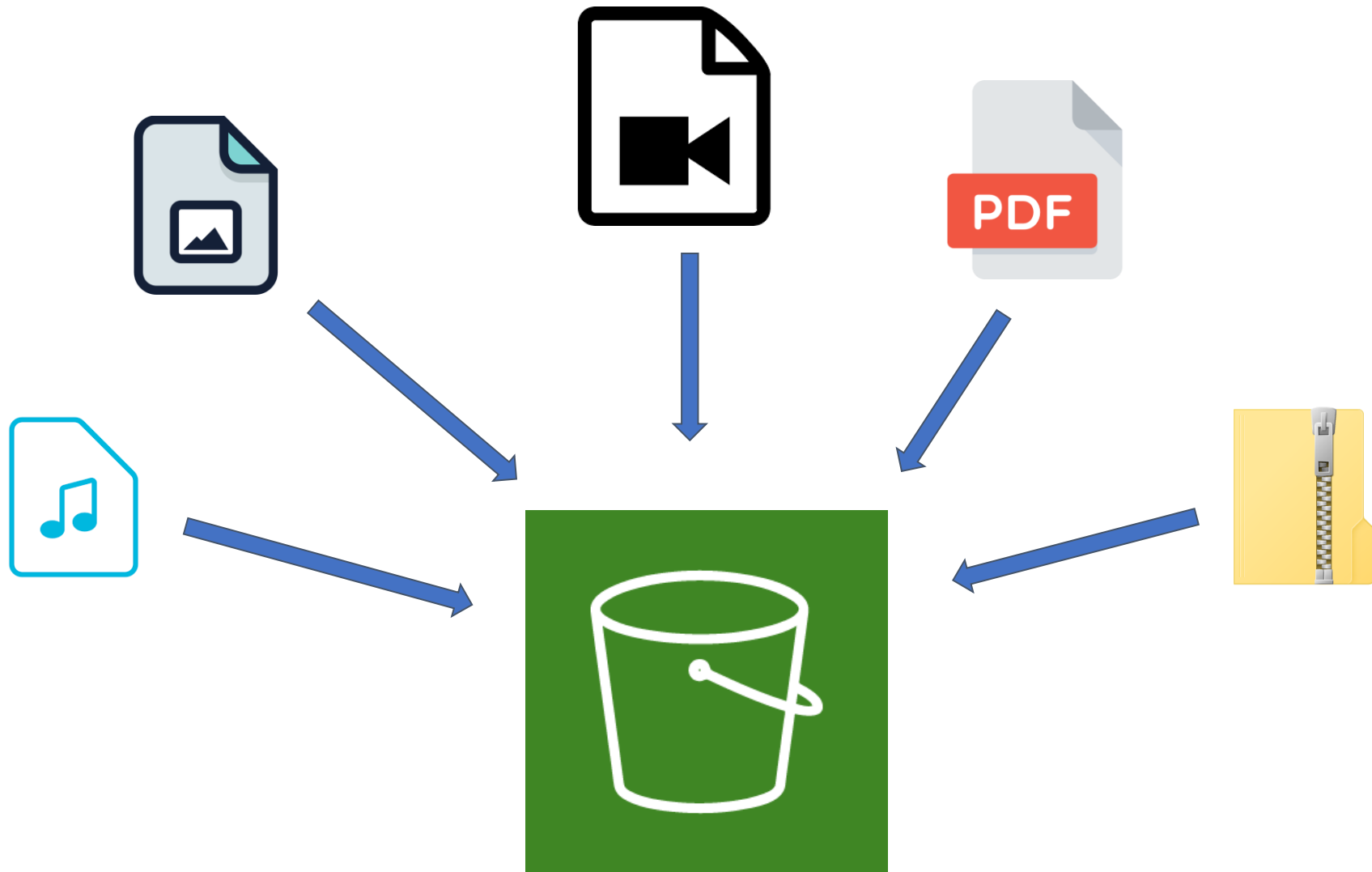
# S3 - Overview

---

- 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 uses Amazon S3 as an integration as well

# S3 - Overview

---



# S3 - Bucket

---

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

# S3 - Object

---

- An object is a file and any metadata that describes that file
- An object consists of following:
  - Key: The name that you assign to an object.
  - Value: The content that you are storing.
  - Metadata: A set of name-value pairs with which you can store information regarding the object.
  - Version ID: Within a bucket, a key and version ID uniquely identify an object.
- Object values are the content of the body:
  - Max Object Size is 5TB (5000GB)
  - If uploading more than 5GB, must use “multi-part upload”



# S3 - Storage Classes

---

- Amazon S3 Standard - General Purpose
- Amazon S3 Standard-Infrequent Access (IA)
- Amazon S3 One Zone-Infrequent Access
- Amazon S3 Intelligent Tiering
- Amazon Glacier
- Amazon Glacier Deep Archive

# S3 Standard - General Purpose

---

- Designed for durability of 99.999999999% of objects across multiple Availability Zones
- Supports SSL for data in transit and encryption of data at rest
- Low latency and high throughput performance (Frequently accessed data)
- Use Cases: dynamic websites, content distribution, mobile and gaming applications, and big data analytics.

## S3 Standard – Infrequent Access (IA)

---

- Suitable for data that is **less frequently accessed**, but requires rapid access when needed
- High durability (99.999999999%) of objects across multiple AZs
- 99.9% Availability
- Supports SSL for data in transit and encryption of data at rest
- Has a lower cost per-GB to store data (about 50% of S3 Standard), but a high cost to store and retrieve items. In addition to charging double (or more) to store and retrieve files, there is also a per-GB charge for retrieving data
- Use Cases: long-term storage, backups, and as a data store for disaster recovery files

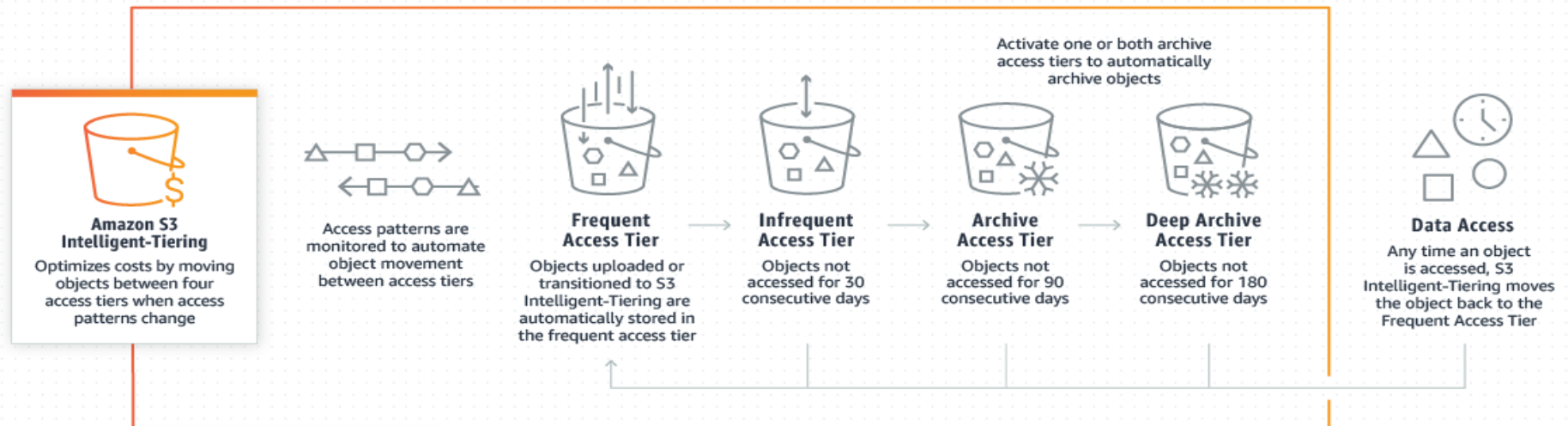
## S3 One Zone - Infrequent Access (IA)

---

- Same as IA but data is stored in a single AZ
- High durability (99.999999999%) of objects in a single AZ; data lost when AZ is destroyed
- 99.5% Availability
- Low latency and high throughput performance
- Supports SSL for data at transit and encryption at rest
- Low cost compared to IA (by 20%)
- Use Cases: For infrequently accessed data that is **re-creatable**:
  - Copies of on-premises backups
  - For storage that is already replicated in another AWS Region for compliance or disaster recovery purposes

# S3 Intelligent Tiering

- Small monthly monitoring and auto-tiering fee
- Automatically moves objects between two access tiers based on changing access patterns
- Designed for durability of 99.999999999% of objects across multiple Availability Zones
- Designed for 99.9% availability over a given year



# S3 Glacier

---

- Purpose-built for data **archiving**
  - Designed for durability of 99.999999999% of objects across multiple Availability Zones
  - Designed for 99.9% data availability in a given year
  - 128 KB minimum object size
- 3 types:
  - Amazon S3 Glacier Instant Retrieval storage class
  - Amazon S3 Glacier Flexible Retrieval (Formerly S3 Glacier) storage class
  - Amazon S3 Glacier Deep Archive (S3 Glacier Deep Archive)

# Amazon S3 Glacier Instant Retrieval

---

- Data retrieval in milliseconds with the same performance as S3 Standard
- Save up to 68% on storage costs compared to using the S3 Standard-Infrequent Access (S3 Standard-IA) storage class, when your data is accessed once per quarter
- Delivers the fastest access to archive storage, with the same throughput and milliseconds access as the S3 Standard and S3 Standard-IA storage classes
- Use case: For archive data that needs immediate access, such as medical images, news media assets, or user-generated content archives

# Amazon S3 Glacier Flexible Retrieval (Formerly S3 Glacier)

---

- Delivers low-cost storage, up to 10% lower cost (than S3 Glacier Instant Retrieval), for archive data that is accessed 1—2 times per year
- Flexible retrieval options that balance cost with access times ranging from minutes to hours and with free bulk retrievals
- Use case: For backup, disaster recovery, and for when some data occasionally need to be retrieved in minutes, and you don't want to worry about costs.



# Amazon S3 Glacier Deep Archive

---

- Lowest-cost storage class and supports long-term retention and digital preservation for data that may be accessed once or twice in a year.
- Lowest cost storage class designed for long-term retention of data that will be retained for 7-10 years
- Retrieval time within 12 hours
- Use cases:
  - Designed for customers—particularly those in highly-regulated industries, such as financial services, healthcare, and public sectors—that retain data sets for 7—10 years or longer to meet regulatory compliance requirements
  - Backup and disaster recovery

# S3 - Storage classes comparison

	S3 Standard	S3 IntelligentTiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier Instant Retrieval	S3 Glacier Flexible Retrieval	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.9999999% 99% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.9999999% 99% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128 KB	128 KB	128 KB	40 KB	40 KB
Minimum storage duration charge	N/A	N/A	30 days	30 days	90 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved

# Lifecycle - Expiration

---

January 26  
2016



Day 0



Object  
Uploaded



February 9  
2016



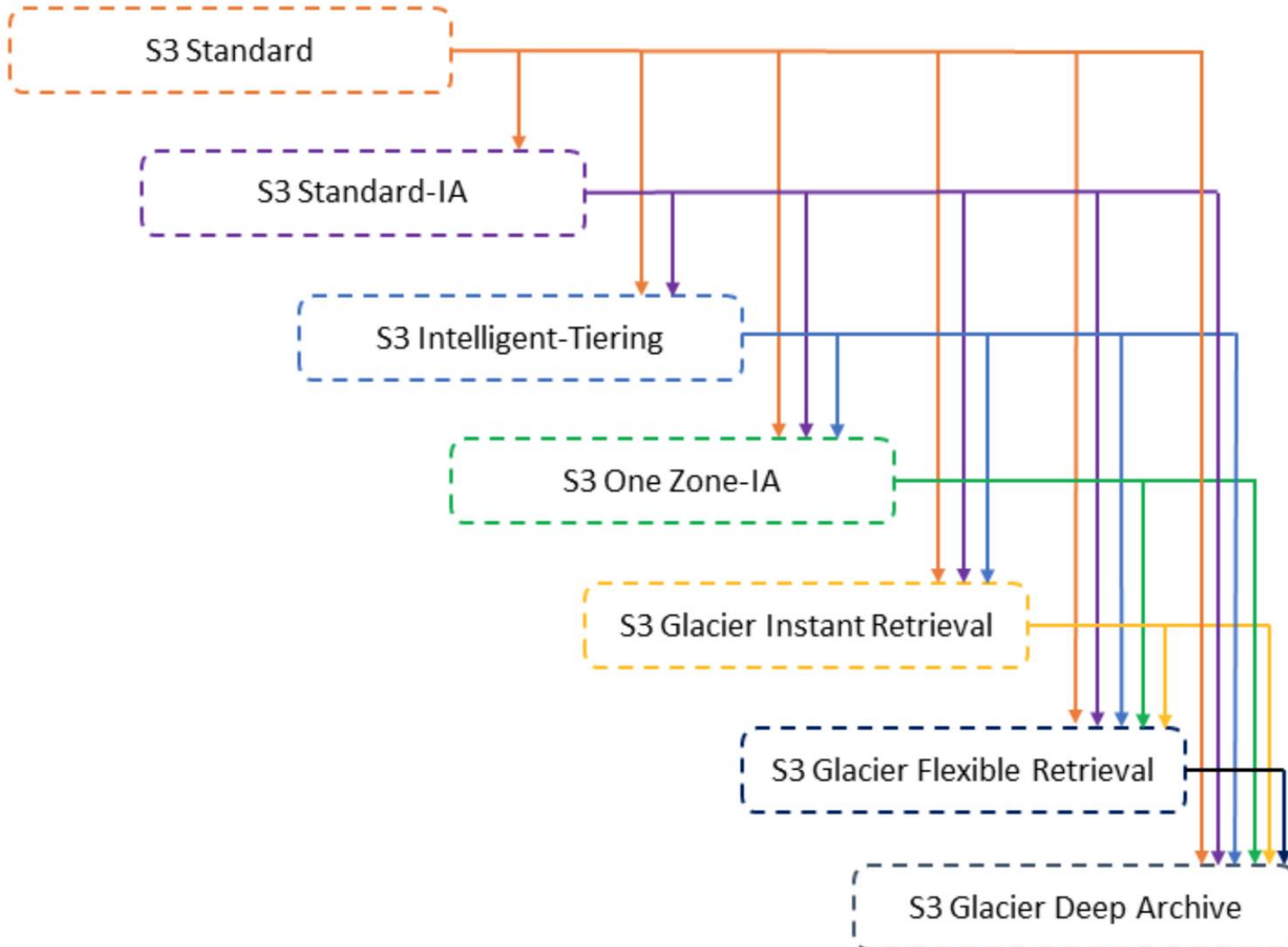
Day 14



Rule:  
Expire

Object Deleted

# Lifecycle - Transition



# Lifecycle - Notes

---

## Expiration action

- If you create an S3 Lifecycle expiration rule that causes objects that have been in S3 Standard-IA or S3 One Zone-IA storage for less than 30 days to expire, you are charged for 30 days.
- If you create a Lifecycle expiration rule that causes objects that have been in S3 Glacier Flexible Retrieval storage for less than 90 days to expire, you are charged for 90 days.
- If you create a Lifecycle expiration rule that causes objects that have been in S3 Glacier Deep Archive storage for less than 180 days to expire, you are charged for 180 days.

# Lifecycle - Notes

---


## Transition action

- If there are more than 1 rule, the order will be from top to bottom:  
EX: If transition 1 is Glacier, transition 2 must be Deep Archive.
- Constraints:
  - Transitioning to Standard-IA or One zone-IA will take at least 30 days

Storage class transitions

Days after object creation

Remove transition

 A minimum of 30 days is required before transitioning to Standard-IA.

Storage class transitions

Days after object creation

Remove transition

 A minimum of 30 days is required before transitioning to One Zone-IA.

# Lifecycle - Notes

---

- Transition days will determined based on Minimum storage duration charge

EX:

1. Transition 1 is Standard-IA with 30 days, Transition 2 must be One zone - IA with 60 days
2. Transition 1 is Glacier with 10 days, Transition 2 must be Glacier Deep Archive with 100 days

# S3 - Versioning

---

- It is enabled at the bucket level
- Same key overwrite will increment the “version”: 1, 2, 3....
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Notes:
  - Any file that is not versioned prior to enabling versioning will have version “null”
  - Once enabled, can not be disabled, only **suspended**.
  - Suspending versioning does not delete the previous versions
  - Increase the size of bucket to store file versions
  - Default permission of new version is PRIVATE. Permission of new version is independent of old version.



## S3 - Bucket policy

---

```
{  
  "Version": "2012-10-17",  
  "Id": "Policy1611277539797",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Principal": {  
        "AWS": "arn:aws:iam::217578211887:user/htquyen-1"  
      },  
      "Action": "s3:*",  
      "Resource": "arn:aws:s3:::bucket-quyen/*"  
    }  
  ]  
}
```

# S3 - Encryption

---

- **Server-side encryption**

- SSE-S3 (Amazon S3-managed encryption keys)
- SSE-KMS (Server-Side Encryption with CMKs Stored in AWS Key Management Service)
- SSE-C (server-side encryption with customer-provided encryption keys)

- **Client-side encryption**

# Server-side encryption

---

- **SSE-S3 (Amazon S3-managed encryption keys)**

Algorithm: AES-256

Header: **x-amz-server-side-encryption**

- **SSE-KMS (Server-Side Encryption with CMKs Stored in AWS Key Management Service)**

- Algorithm: AES-256

- Header:

- + **s3:x-amz-server-side-encryption: "aws:kms"**

- + **s3:x-amz-server-side-encryption-aws-kms-key-id: "{key id}"**

- **SSE-C (server-side encryption with customer-provided encryption keys)**

Algorithm: AES-256

Header:

- + **x-amz-server-side-encryption-customer-algorithm**: Encrypted algorithm

- + **x-amz-server-side-encryption-customer-key**: 256-bit, base64-encoded key

- + **x-amz-server-side-encryption-customer-key-MD5**: for check sum

=> Must be encrypted in transit (using HTTPS)

# Default encryption

---

- If enabled, all objects uploaded to bucket will be automatically SSE-S3 or SSE-KMS encrypted.
  - If objects are encrypted before sending to bucket, objects WILL NOT be encrypted by default encryption.
- => Order: **Object encryption** > **Default encryption**

## Note:

- If Bucket policy is set required PUT action with SSE-S3 or SSE-KMS encryption, even though Default Encryption is enabled with SSE-S3 or SSE-KMS, PUT action without required header will be **denied**.

**DEMO**

# S3 - Pricing

---



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

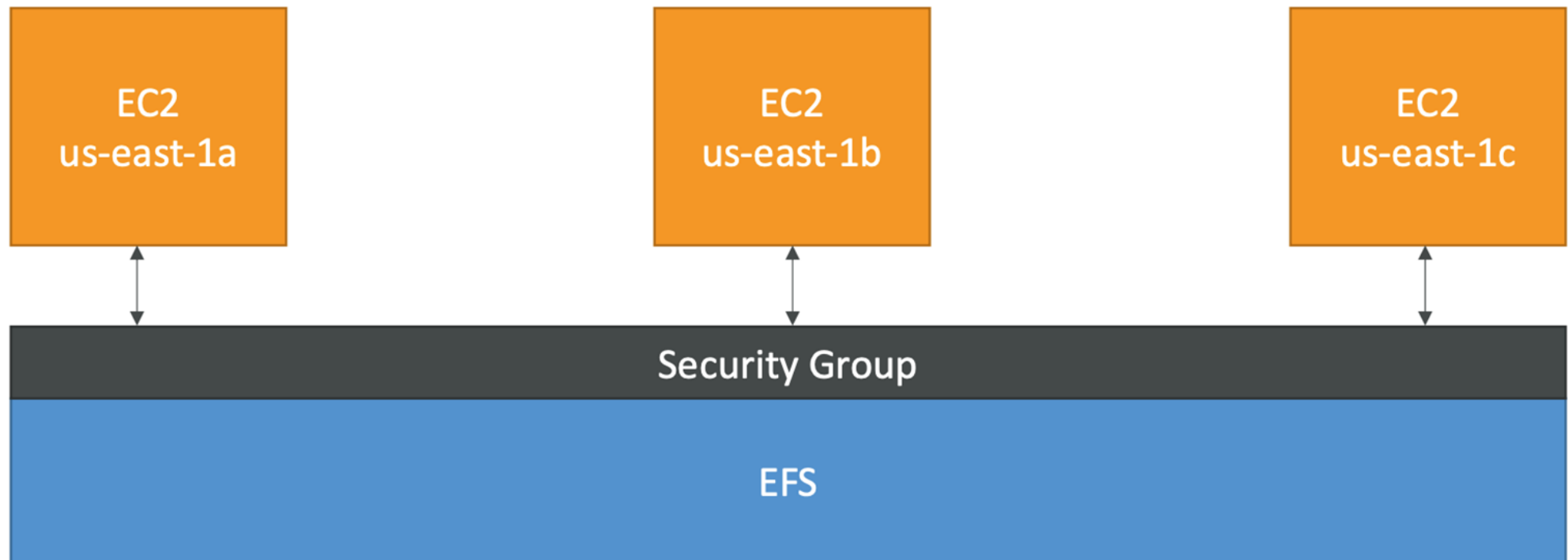


## Elastic File System

# EFS - Overview

---

- 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





# EFS - Overview

---

- Uses NFSv4.1 protocol
- **Uses security group** to control access to EFS
- Compatible with **Linux based AMI** (not Windows)
- Encryption at rest using KMS
- File system **scales automatically, pay-per-use, no capacity planning**

# EFS - Performance and Storage Tiers

---

- **Performance mode (set at EFS creation time)**

- General purpose (default): latency-sensitive use cases (web server, content management systems, home directories, and general file serving)
- Max I/O – higher latency, throughput, highly parallel (big data, media processing)
  - => No support One-zone tier

- **Storage Tiers**

- Standard: for frequently accessed files
    - + Standard
    - + One-zone
  - Infrequent access (EFS-IA): cost to retrieve files, lower price to store
    - + Standard - IA (Per GB retrieval fees apply)
    - + One-zone IA (Per GB retrieval fees apply)
- => Unable to choose Standard-IA and One-zone IA while creating file but through Life cycle.



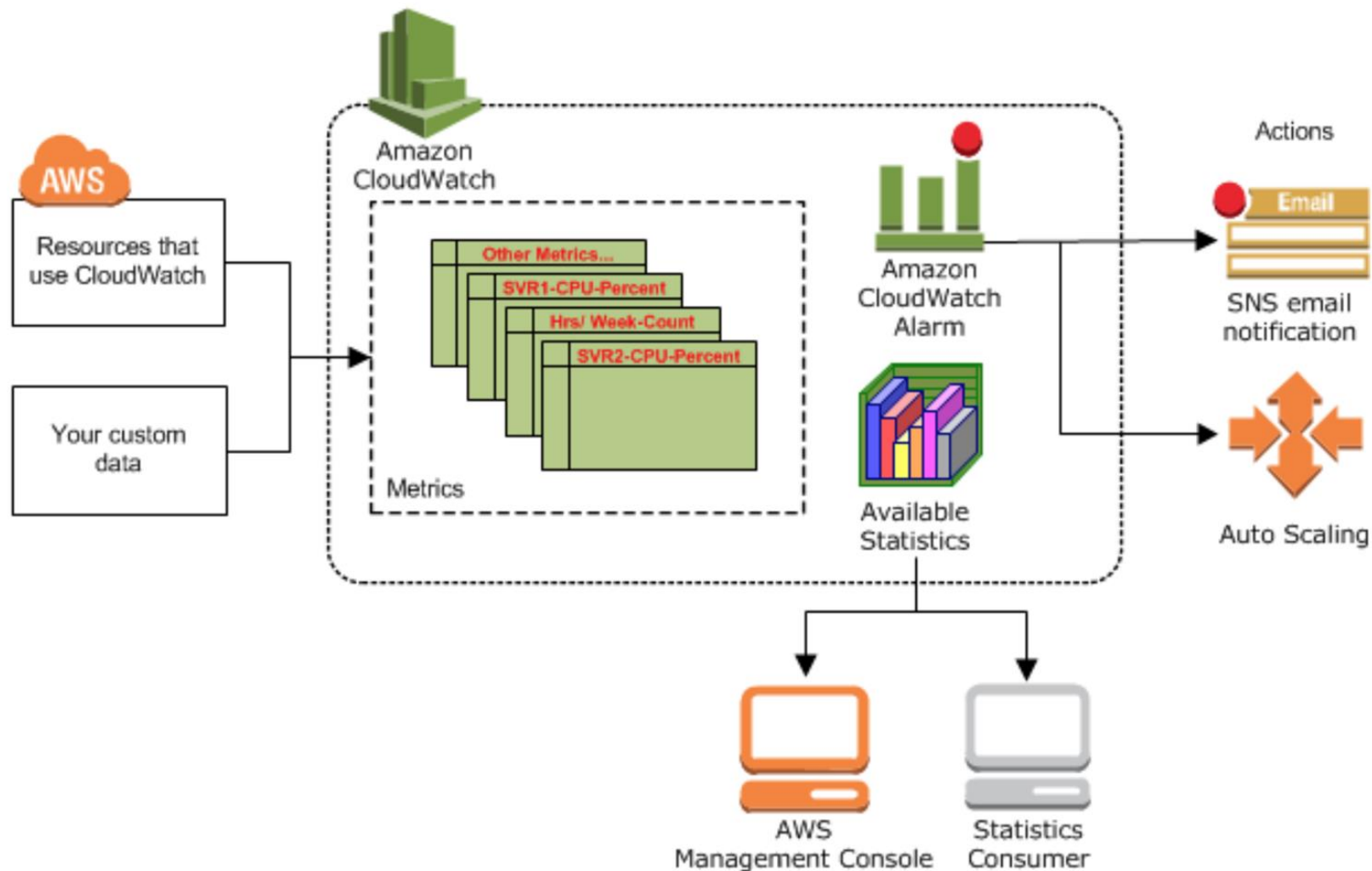
## Amazon CloudWatch

# Overview

---

- Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time
- You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.
- 4 Types:
  - Metrics: Collect and track key metrics
  - Logs: Collect, monitor, analyze and store log files
  - Events: Send notifications when certain events happen in your AWS
  - Alarms: React in real-time to metrics

# CloudWatch Metrics and Alarms



---

# Elastic Load Balancing & Auto Scaling Groups

# Scalability & High Availability (HA)

---

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

---

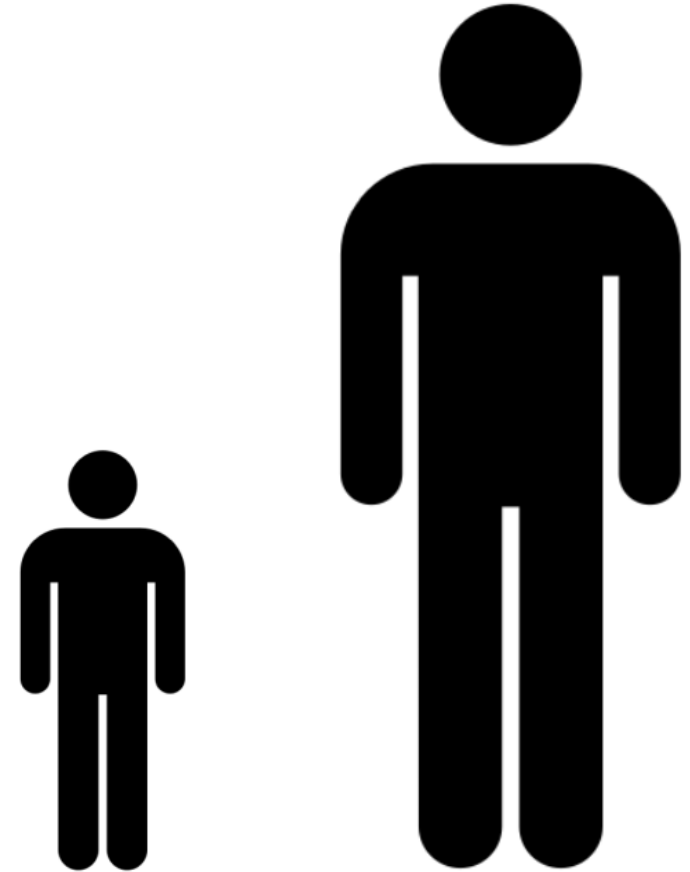
Vertical Scalability means increasing the size of the instance

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

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



junior operator

senior operator



# Horizontal Scalability

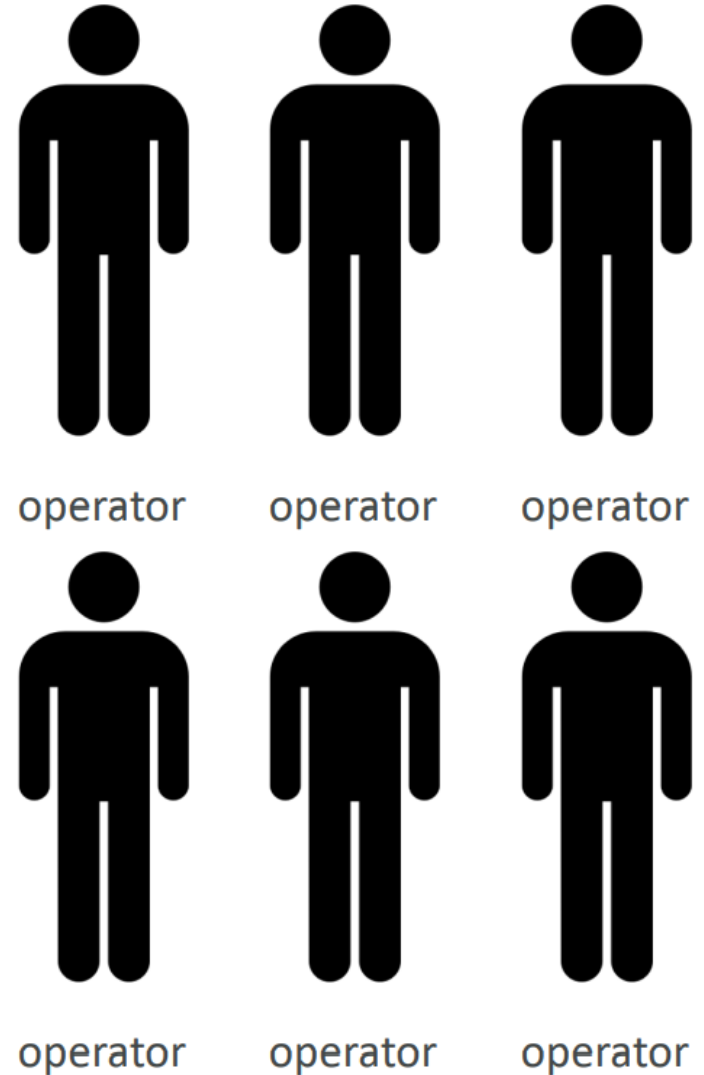
---

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 to the cloud offerings such as Amazon EC2



# High Availability

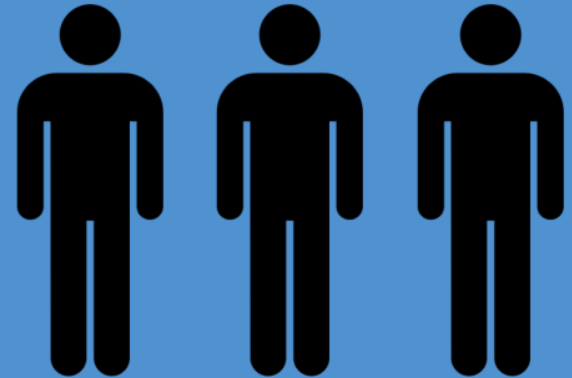
---

High availability usually goes hand in hand with horizontal scaling

High availability means running your application / system in at least 2 Availability Zones

The goal of high availability is to survive a data center loss (disaster)

first building in New York



second building in San Francisco



# High Availability & Scalability For EC2

---

Vertical Scaling: increase instance size (= scale up / down)

From: t2.nano - 0.5G of RAM, 1 vCPU

To: u-12tb1.metal – 12.3 TB of RAM, 448 vCPUs

Horizontal Scaling: increase number of instances (= scale out / in)

Auto Scaling Group

Load Balancer

High Availability: Run instances for the same application across multi AZ

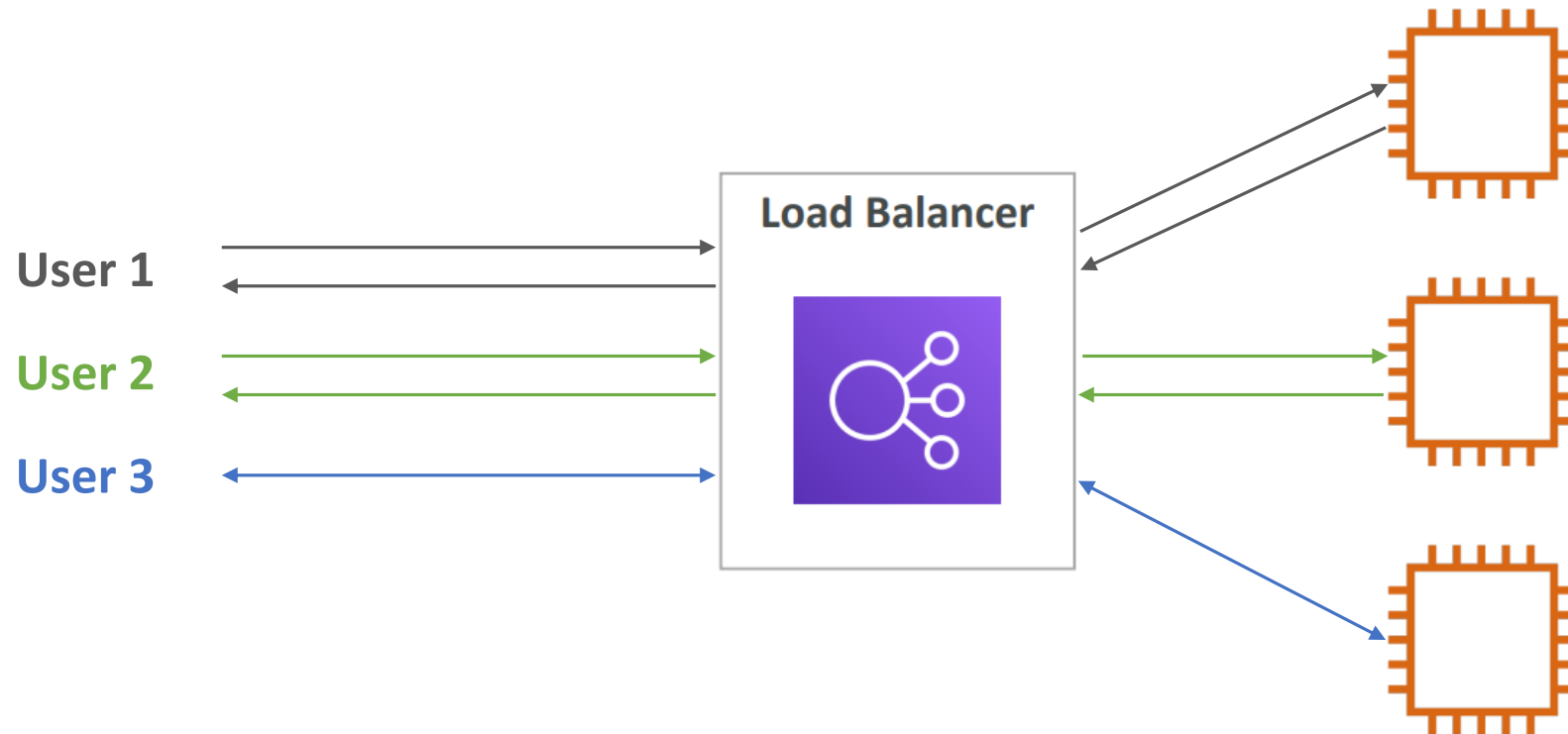
Auto Scaling Group multi AZ

Load Balancer multi AZ

# What is load balancing?

---

Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream



# Why use an Elastic Load Balancer?

---

An ELB (Elastic Load Balancer) is a managed load balancer

Expose a single point of access (DNS) to your application

Do regular health checks to your instances

Provide SSL termination (HTTPS) for your websites

It costs less to setup your own load balancer but it will be a lot more effort on your end (maintenance, integrations)

4 kinds of load balancers offered by AWS

- Application Load Balancer (HTTP / HTTPS only) – layer 7

- Network Load Balancer (ultra-high performance, allows TCP) – layer 4

- Gateway Load Balancer (new)

- Classic Load Balancer (slowly retiring) – layer 4 & 7

# What is 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 increase load

- Scale in (remove EC2 instances) to match a decrease load

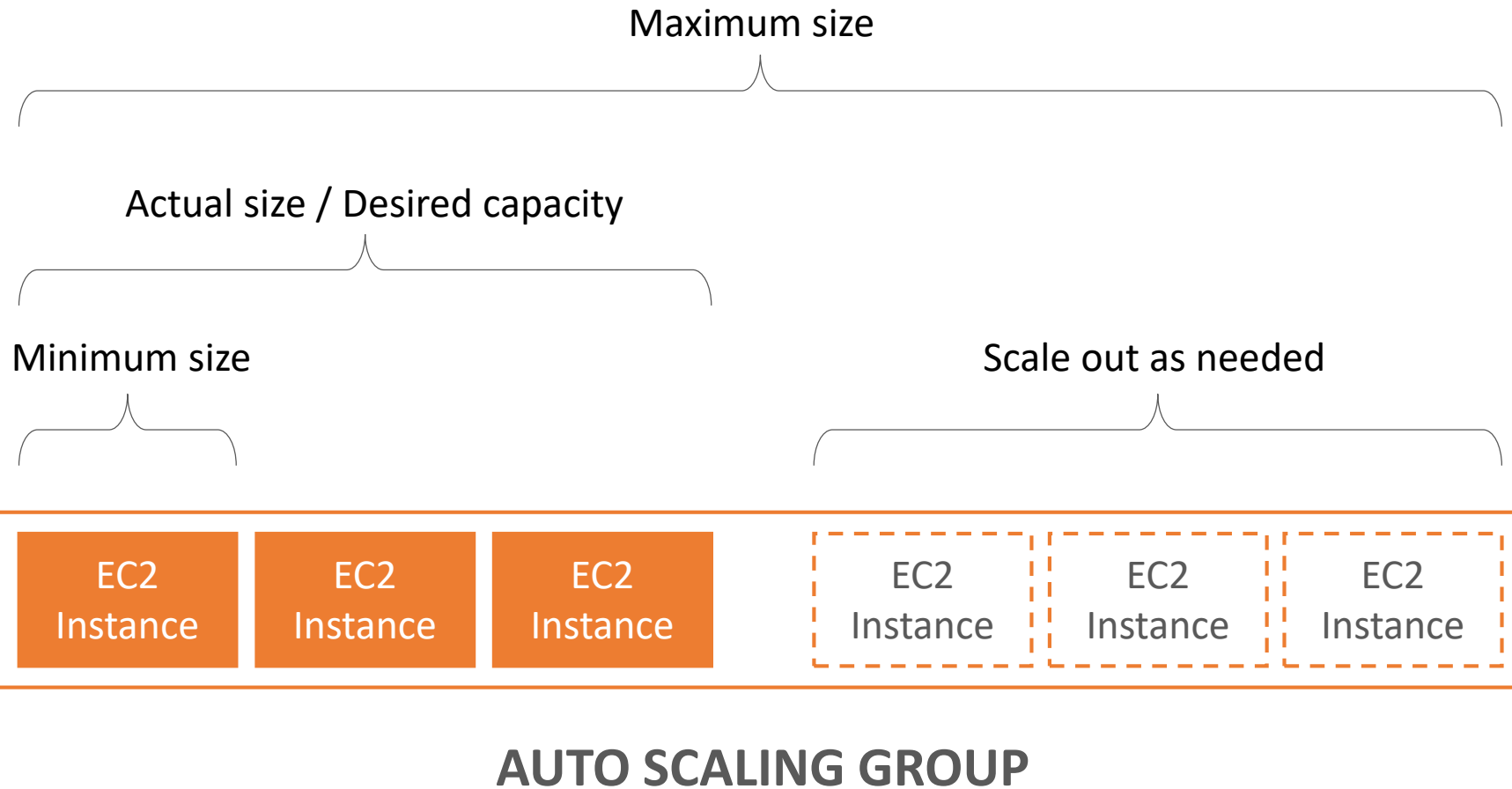
- Ensure we have a minimum and a maximum number of machines running

- Automatically register new instances to a load balancer

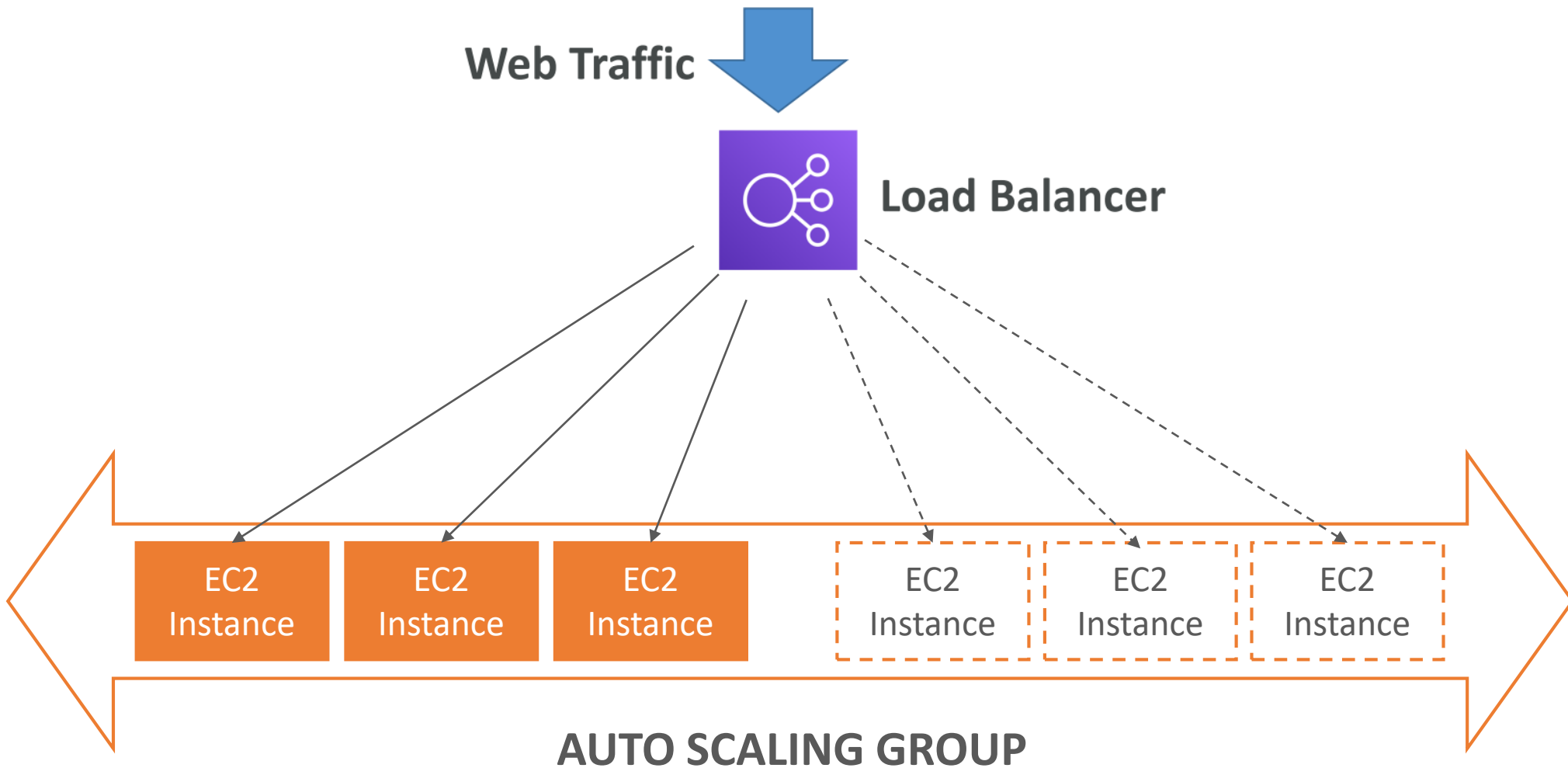
- Replace unhealthy instances

Cost Savings: only run at an optimal capacity (principle of the cloud)

# Auto Scaling Group in AWS



# Auto Scaling Group with Load Balancer





# Auto Scaling Groups – Scaling Strategies

---

**Manual Scaling:** Update the size of an ASG manually

**Dynamic Scaling:** Respond to changing demand

## **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

## **Target Tracking Scaling**

Example: I want the average ASG CPU to stay at around 40%

## **Scheduled Scaling**

Anticipate a scaling based on known usage patterns

Example: increase the min. capacity to 10 at 5 pm on Fridays

## **Predictive Scaling**

Uses Machine Learning to predict future traffic ahead of time

---

# Database

# Databases Intro

---

Storing data on disk (EFS, EBS, EC2 Instance Store, S3) can have its limits

Sometimes, you want to store data in a database...

You can **structure** the data

You build **indexes** to efficiently **query** / **search** through the data

You define **relationships** between your **datasets**

Databases are **optimized for a purpose** and come with different features, shapes and constraints

# Relational Databases

Looks just like Excel spreadsheets, with links between them!

Can use the SQL language to perform queries / lookups

Students

Student ID	Dept ID	Name	Email
1	M01	Joe Miller	joe@abc.com
2	B01	Sarah T	sarah@abc.com

Departments

Dept ID	SPOC	Email	Phone
M01	Kelly Jones	kelly@abc.com	+1234567890
B01	Satish Kumar	satish@abc.com	+1234567891

Subjects

Student ID	Subject
1	Physics
1	Chemistry
1	Math
2	History
2	Geography
2	Economics

# NoSQL Databases

---

NoSQL = non-SQL = non relational databases

NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications

## Benefits

- Flexibility: easy to evolve data model

- Scalability: designed to scale-out by using distributed clusters

- High-performance: optimized for a specific data model

- Highly functional: types optimized for the data model

Examples: Key-value, document, graph, in-memory, search databases

# NoSQL data example: JSON

---

JSON = JavaScript Object Notation

JSON is a common form of data that fits into a NoSQL model

Data can be **nested**

Fields can **change** over time

Support for new types: arrays, etc...

```
{  
  "name": "John",  
  "age": 30,  
  "cars": [  
    "Ford",  
    "BMW",  
    "Fiat"  
  ],  
  "address": {  
    "type": "house",  
    "number": 23,  
    "street": "Dream Road"  
  }  
}
```

# Databases & Shared Responsibility on AWS

---

AWS offers use to **manage** different databases

Benefits include:

- Quick Provisioning, High Availability, Vertical and Horizontal Scaling

- Automated Backup & Restore, Operations, Upgrades

- Operating System Patching is handled by AWS

- Monitoring, alerting

Note: many databases technologies could be run on EC2, but you must handle yourself the resiliency, backup, patching, high availability, fault tolerance, scaling...

# 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)

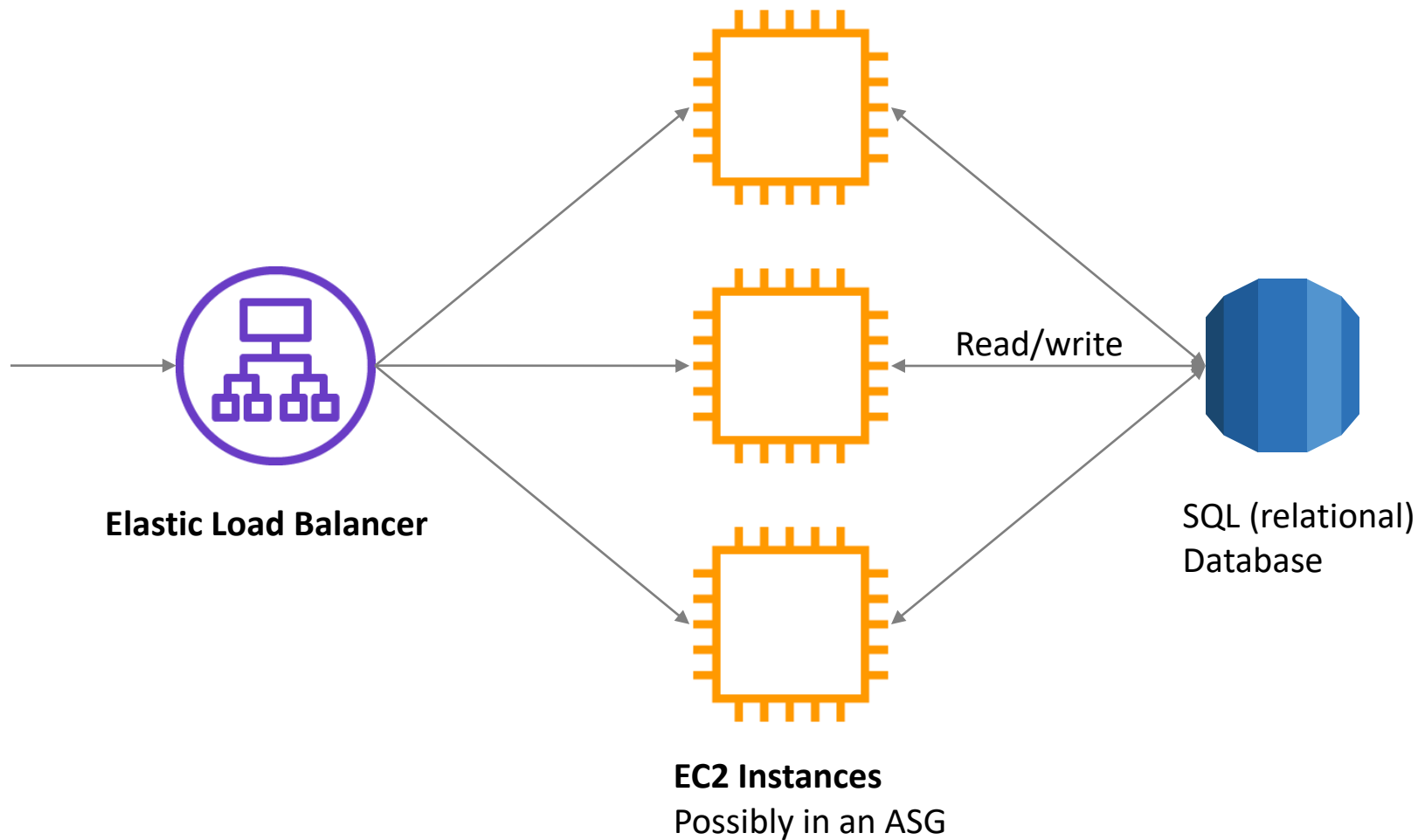
- Maintenance windows for upgrades

- Scaling capability (vertical and horizontal)

- Storage backed by EBS (gp2 or io1)

BUT you can't SSH into your instances

# RDS Solution Architecture



# Amazon Aurora

---

Aurora is a proprietary technology from AWS (not open sourced)

**PostgreSQL and MySQL** are both supported as Aurora DB

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 64 TB

Aurora costs more than RDS (20% more) – but is more efficient

Not in the free tier



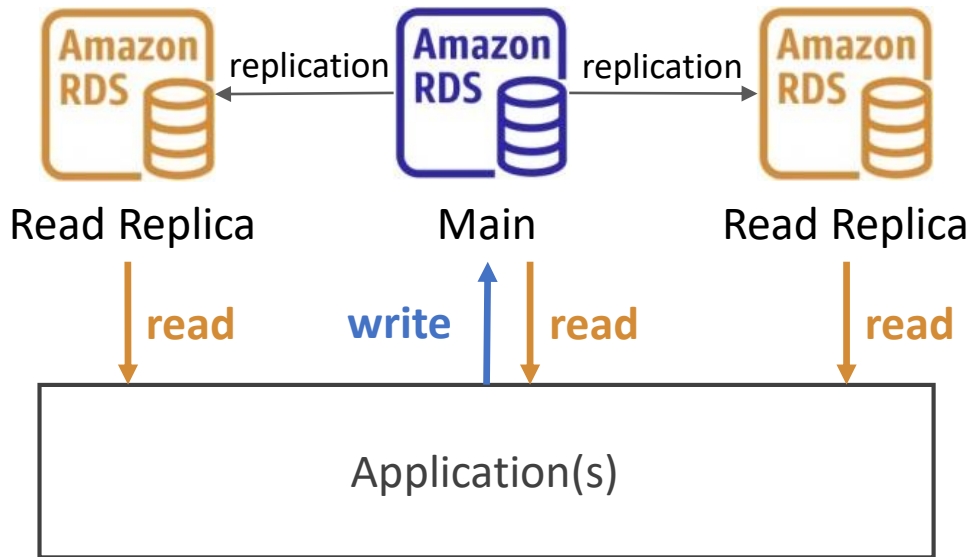
# RDS Deployment Options

## Read Replicas

**Scale** the read workload of your DB

Can create up to 5 Read Replicas

Data is only written to the main DB

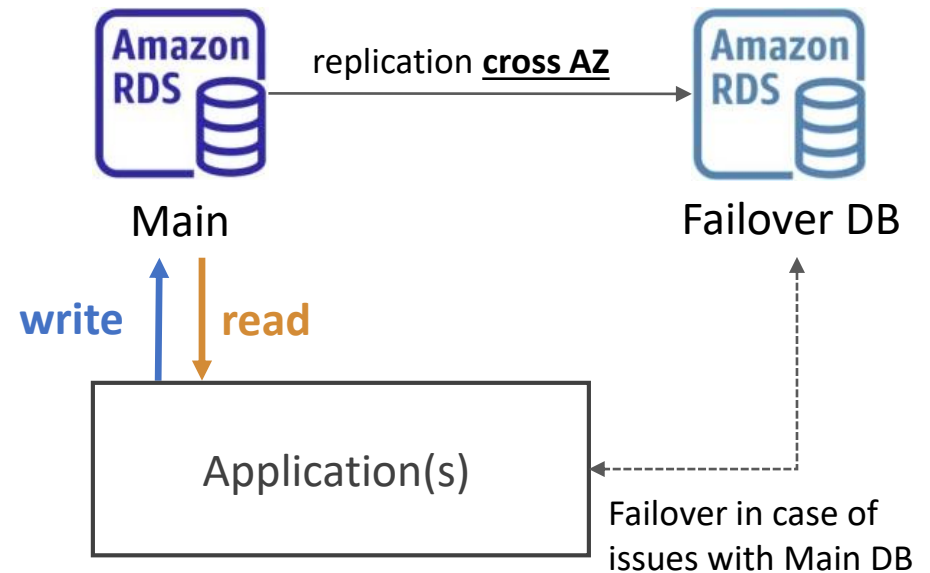


## Multi AZ

**Failover** in case of AZ outage (high availability)

Data is only read/written to the main database

Can only have 1 other AZ as failover



# Amazon ElastiCache Overview

---

The same way RDS is to get managed Relational Databases...

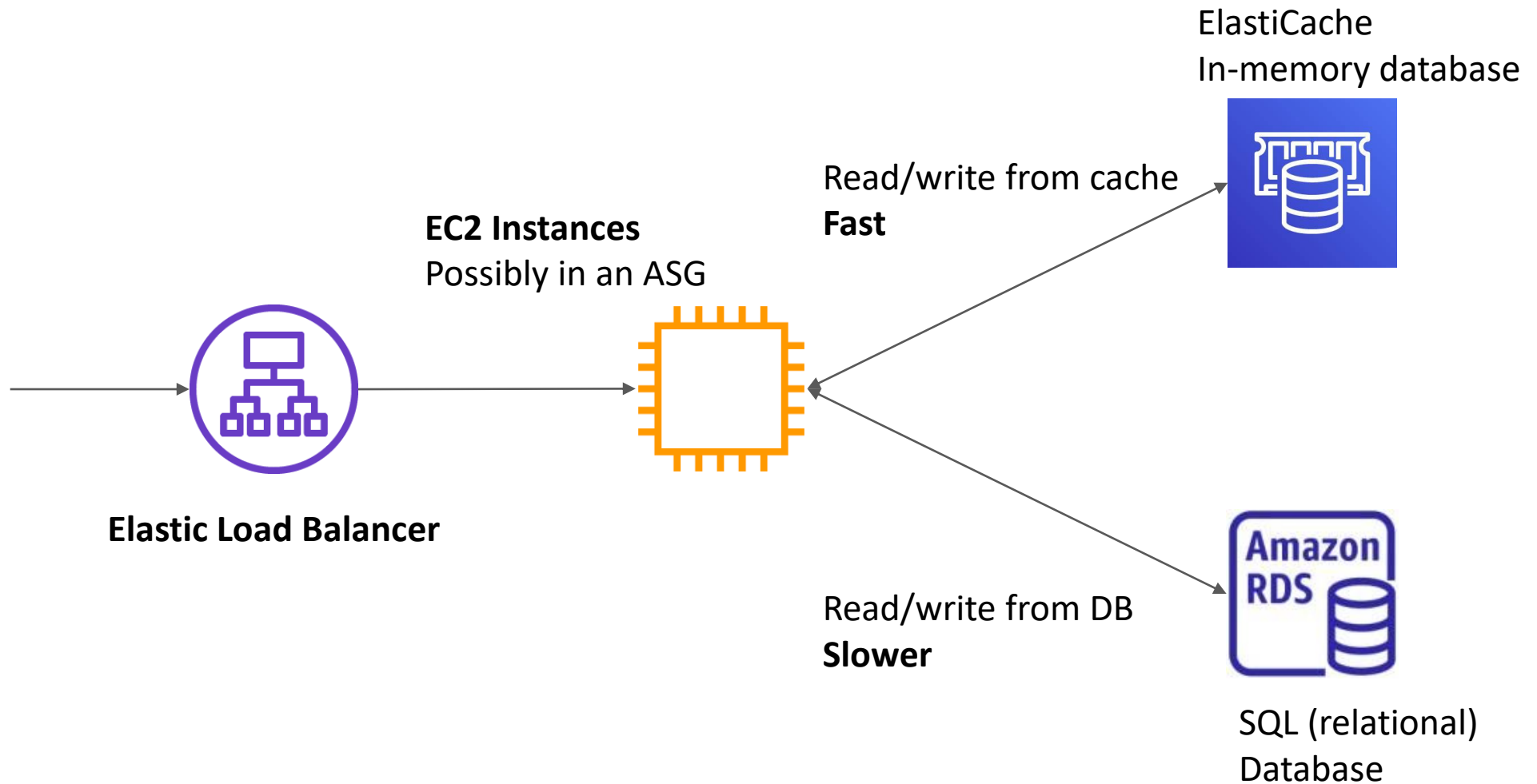
ElastiCache is to get managed Redis or Memcached

Caches are **in-memory databases** with high performance, low latency

Helps **reduce load off databases for read intensive workloads**

AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups

# ElastiCache Solution Architecture



---

# Route 53 & CloudFront

# Why make a global application?

---

A **global application** is an application deployed in **multiple geographies**

On AWS: this could be **Regions** and / or **Edge Locations**

## Decreased Latency

Latency is the time it takes for a network packet to reach a server

It takes time for a packet from Asia to reach the US

Deploy your applications closer to your users to decrease latency, better experience

## Disaster Recovery (DR)

If an AWS region goes down (earthquake, storms, power shutdown, politics)...

You can fail-over to another region and have your application still working

A DR plan is important to increase the availability of your application

**Attack protection:** distributed global infrastructure is harder to attack



# Global Application in AWS

---

## Global DNS: Route 53

Great to route users to the closest deployment with least latency

Great for disaster recovery strategies



## Global Content Delivery Network (CDN): CloudFront

Replicate part of your application to AWS Edge Locations – decrease latency

Cache common requests – improved user experience and decreased latency



## S3 Transfer Acceleration

Accelerate global uploads & downloads into Amazon S3



## AWS Global Accelerator

Improve global application availability and performance using the AWS global network



# Amazon Route 53 Overview

---

Route53 is a Managed DNS (Domain Name System)

DNS is a collection of rules and records which helps clients understand how to reach a server through URLs

In AWS, the most common records are:

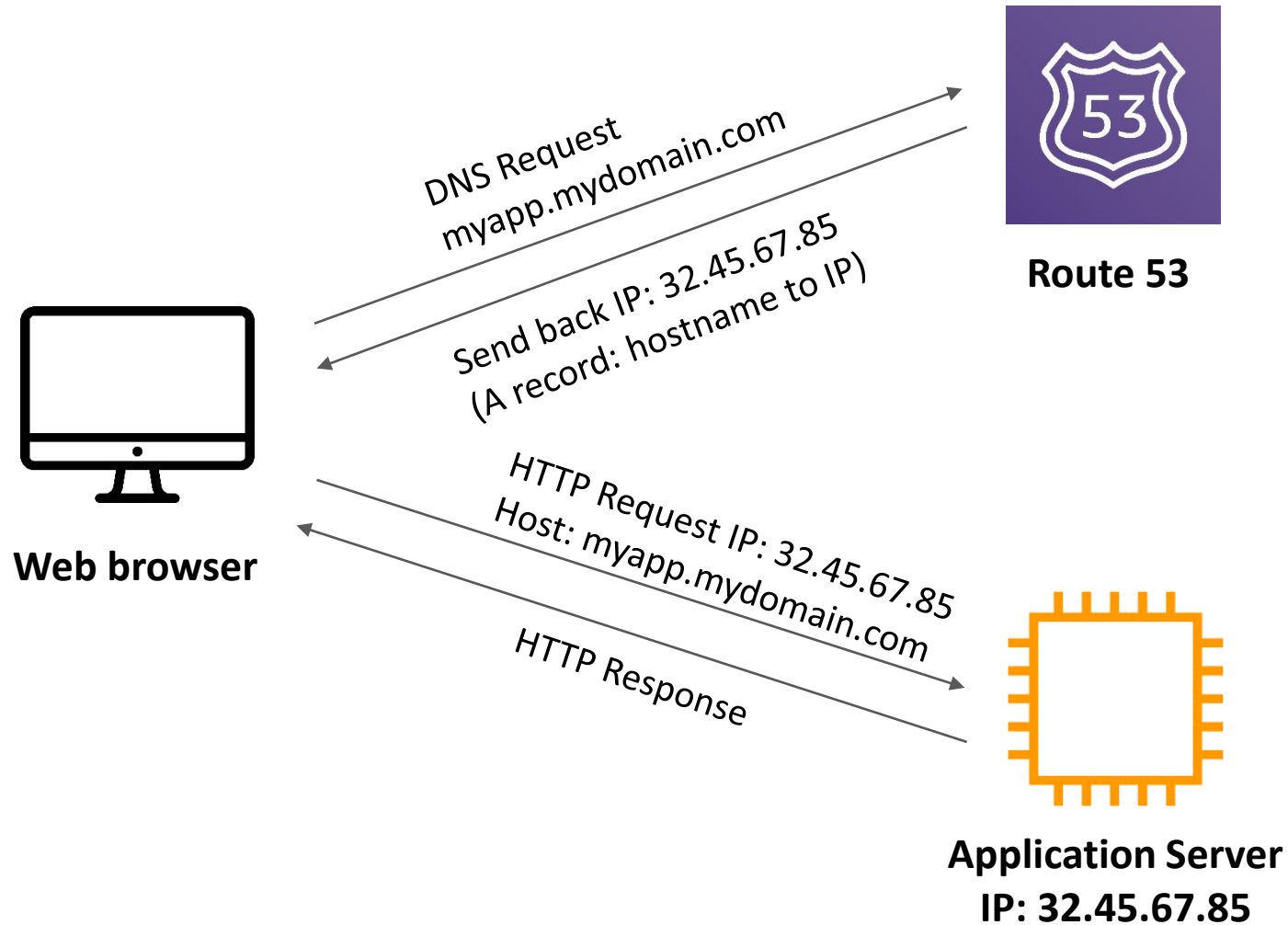
www.google.com => 12.34.56.78 == **A record (IPv4)**

www.google.com => 2001:0db8:85a3:0000:0000:8a2e:0370:7334 == **AAAA IPv6**

search.google.com => www.google.com == **CNAME: hostname to hostname**

example.com => AWS resource == **Alias (ex: ELB, CloudFront, S3, RDS, etc...)**

# Route 53 – Diagram for A record



# Amazon CloudFront Overview

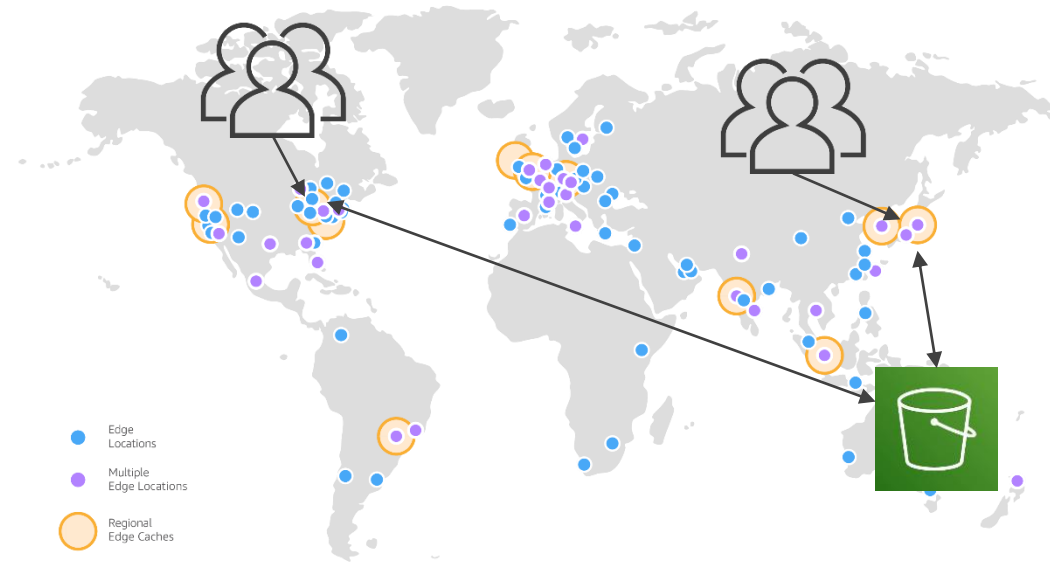
Content Delivery Network (CDN)

**Improves read performance,**  
content is cached at the edge

Improves users experience

+310 Point of Presence globally  
(edge locations)

**DDoS** protection (because  
worldwide), integration with Shield,  
AWS Web Application Firewall



# CloudFront Origins

---

## S3 bucket

For distributing files and caching them at the edge

Enhanced security with CloudFront **Origin Access Identity (OAI)**

CloudFront can be used as an ingress (to upload files to S3)

## Custom Origin (HTTP)

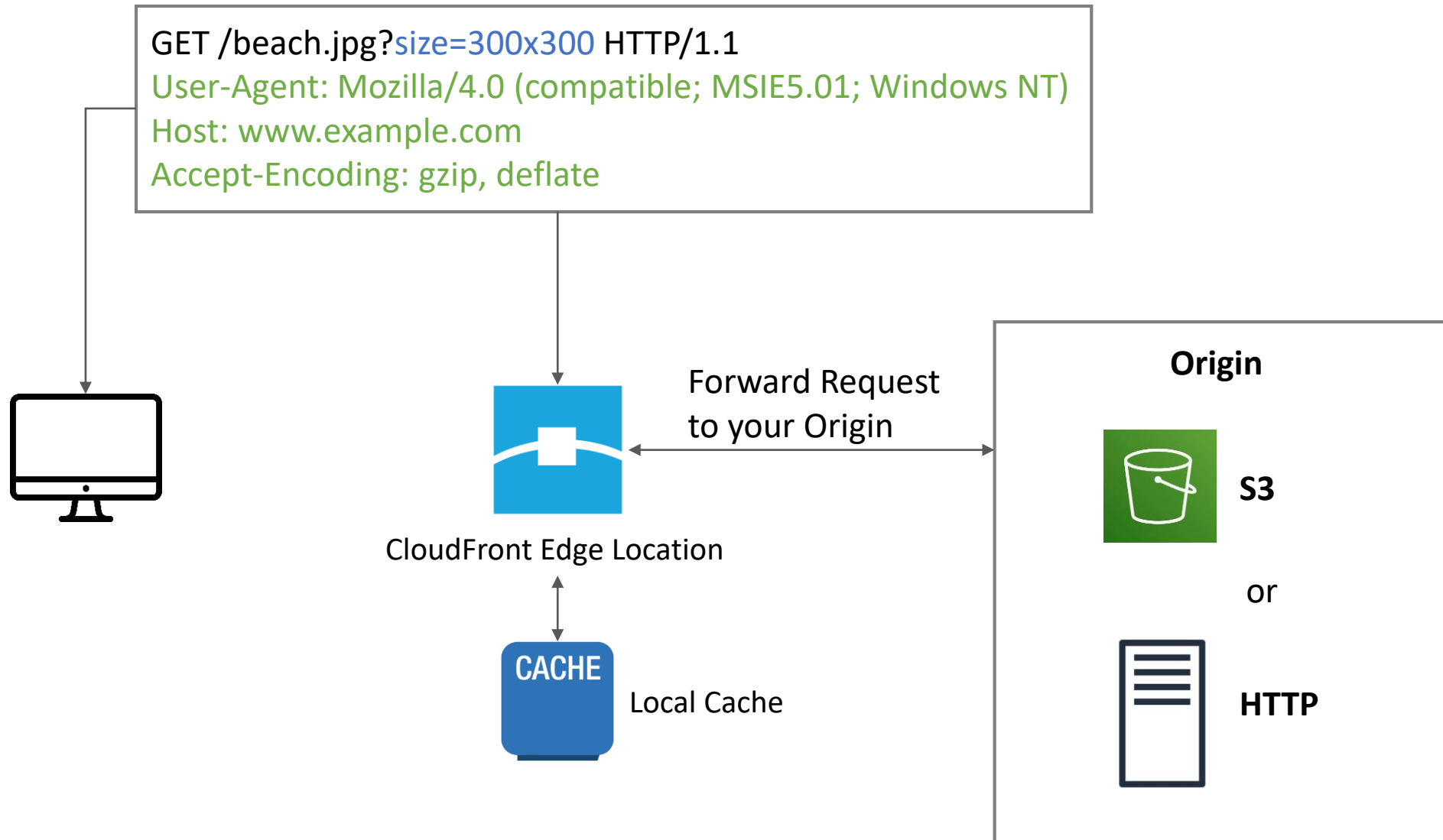
Application Load Balancer

EC2 instance

S3 website (must first enable the bucket as a static S3 website)

Any HTTP backend you want

# CloudFront at a high level



# Head Office and Branches



## HEAD OFFICE – HA NOI

6<sup>th</sup> and 12<sup>th</sup> Floor Ocean Park Building, No.01 Dao Duy Anh Street, Phuong Mai Ward, Dong Da District, Ha Noi

Tel.: +84-24-3772-4304

Fax: +84-24-3772-5204

Email: [info@runsystem.net](mailto:info@runsystem.net)

## BRANCH – HO CHI MINH

7<sup>th</sup> and 8<sup>th</sup> Floor, Ha Do Airport Building  
No.02 Hong Ha, Ward 2, Tan Binh District, Ho Chi Minh

Tel.: +84-28-7308-6086

## BRANCH – DA NANG

1<sup>st</sup> Floor, Central Vietnam Helicopter Company Building,  
Nguyen Van Linh, Thac Gian, Thanh Khe, Da Nang

Tel.: +84-236-388-6066

Fax: +84-236-388-6065

## BRANCH – TOKYO

7<sup>th</sup> Floor Cerulean Tower, 26-1 Sakuragaoka-cho,  
Shibuya-ku, Tokyo, Japan

Tel.: +81-3-5962-0056

Fax: +81-3-3780-4077

## BRANCH – FUKUOKA

1-3-41 Puriodaimyou 2<sup>nd</sup> Floor, Daimyo, Fukuoka Shi  
Chuo Ku, Fukuoka Ken, 810-0041, Japan

Tel.: +81-70-1061-3143

Fax: +81-3-3780-4077

# Thank you!