

# AWS EC2 & IAM

*CS516 – Cloud Computing*

*Computer Science Department*

*Maharishi International University*

# Maharishi International University - Fairfield, Iowa

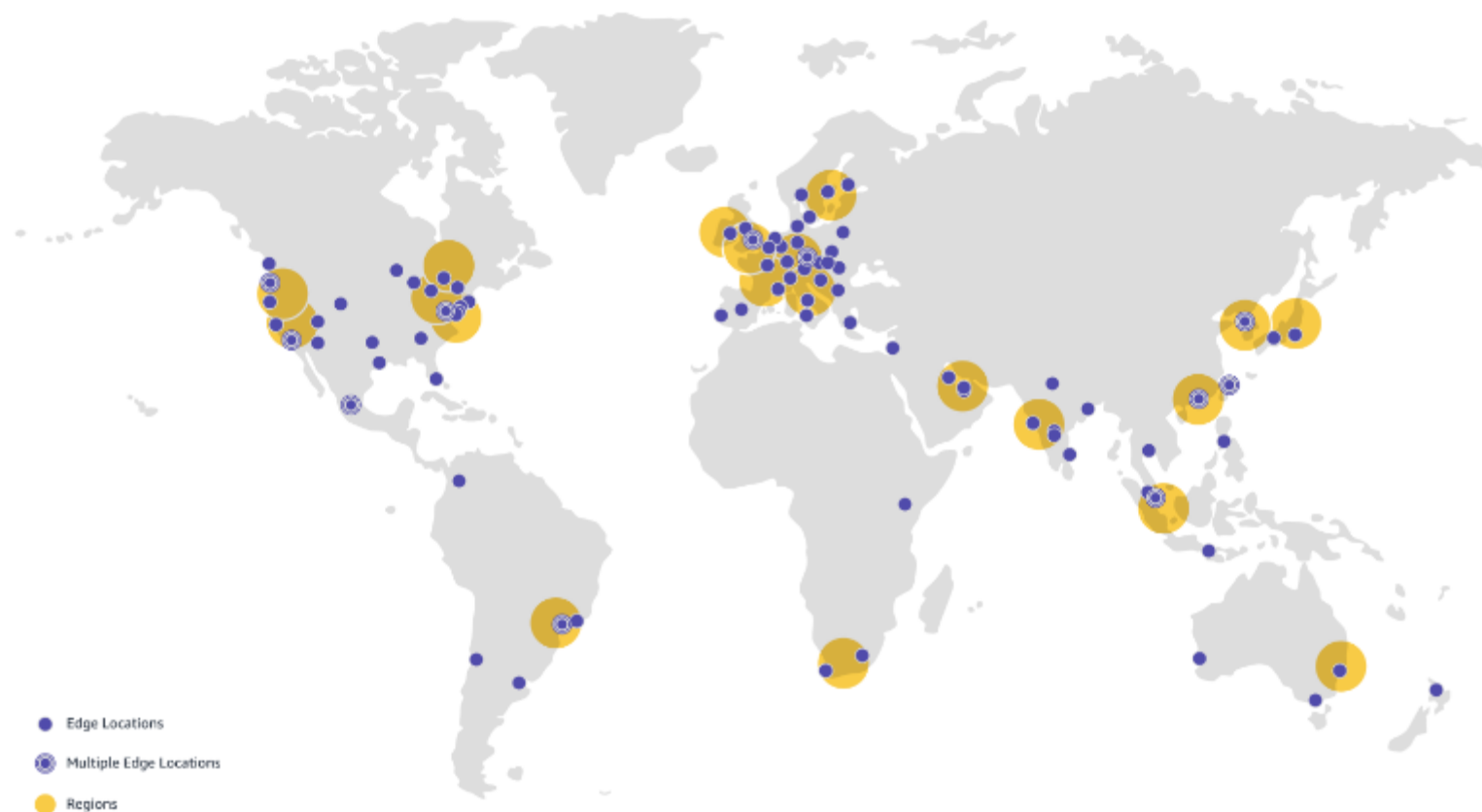


All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi International University.

# Main points

- The big picture – High level architecture of the AWS Cloud. Concepts include
  - Physical – AZs, edge servers
  - Logical – regions, VPC, subnets, and IP addresses.
- Virtual machine (EC2) – Its components such as AMI, EBS, Snapshots, SG.
- Identity access management (IAM) – users, roles, permissions, and STS assume role.

# AWS Global Infrastructure Map

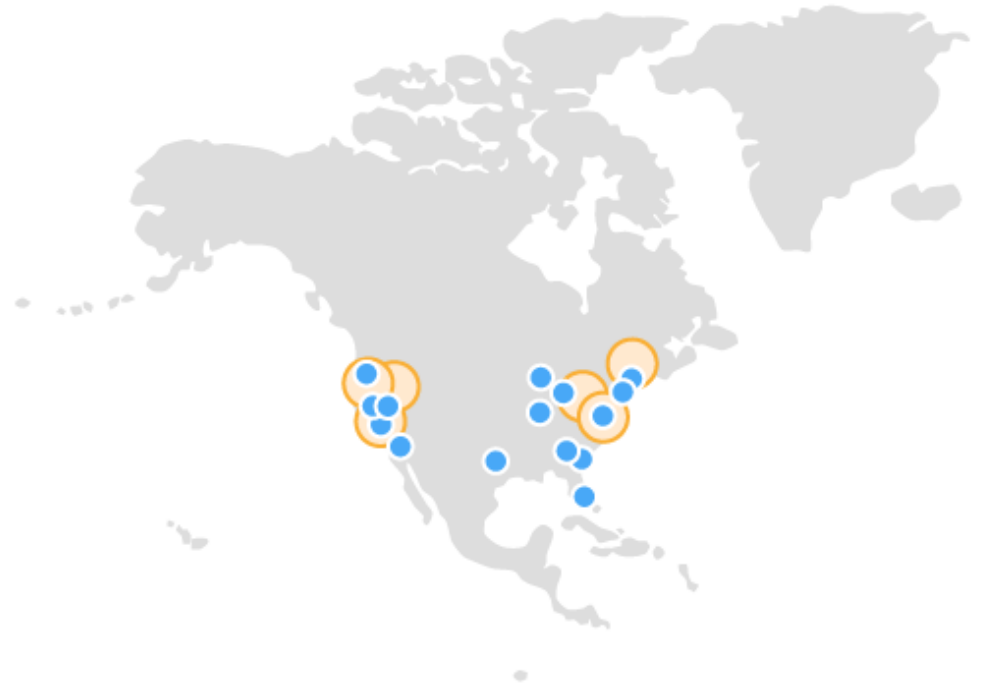


Read more about: [AWS Global Infrastructure](#) and [AWS Global Accelerator features](#)

# AWS Global Infrastructure

**Regions** (Logical) - A physical location around the world where AWS ***clusters*** data centers. One region has multiple data centers.

**Availability Zones** (Physical) - Geographical isolated data centers within a region.



# Regions

- Followings vary depending on the region.
  - Service and service feature availability – Check out [AWS announcement page](#) to see what services and features are newly added in that region.
  - Limit – See next slide
  - Pricing – The same servers in the US region tend to cost less than the ones in Asia.
- Most services are regional. To build and deploy a global app, you have to deploy the same stack to every region. Most companies target 3 regions in 3 different continents.
- If the region goes down, all apps in that region are down as well. So, architects consider a multi-regional deployment strategy.

Read More about: [Regions and Zones](#)

Regions	Transactions per second
<ul style="list-style-type: none"> <li>•US East (N. Virginia) – The biggest. Global resources live in this region.</li> <li>•US West (Oregon)</li> <li>•Europe (Ireland) – One of 2 major regions in the EU</li> </ul>	<a href="#">PutEvents</a> has a soft limit of 10,000 requests per second by default in these Regions.
<ul style="list-style-type: none"> <li>•US East (Ohio)</li> <li>•Europe (Frankfurt) – One of 2 major regions in the EU</li> </ul>	<a href="#">PutEvents</a> has a soft limit of 2,400 requests per second by default in these Regions.
Asia Pacific (Tokyo) – One of 2 major regions in Asia <ul style="list-style-type: none"> <li>•Asia Pacific (Singapore) – One of 2 major regions in Asia</li> <li>•US West (N. California)</li> <li>•Europe (London)</li> <li>•Asia Pacific (Sydney)</li> </ul>	<a href="#">PutEvents</a> has a soft limit of 1,200 requests per second by default in these Regions.
<ul style="list-style-type: none"> <li>•Canada (Central)</li> <li>•Europe (Paris)</li> <li>•South America (São Paulo)</li> <li>•Asia Pacific (Seoul)</li> <li>•Asia Pacific (Mumbai)</li> <li>•Asia Pacific (Hong Kong) – Not even a region. Most services are not available.</li> </ul>	<a href="#">PutEvents</a> has a soft limit of 600 requests per second by default in these Regions.
<ul style="list-style-type: none"> <li>•AWS GovCloud (US-West)</li> <li>•China (Beijing)</li> <li>•Asia Pacific (Osaka)</li> <li>•Africa (Cape Town)</li> </ul>	<a href="#">PutEvents</a> has a soft limit of 400 requests per second by default in these Regions.

# AWS Region/AZ

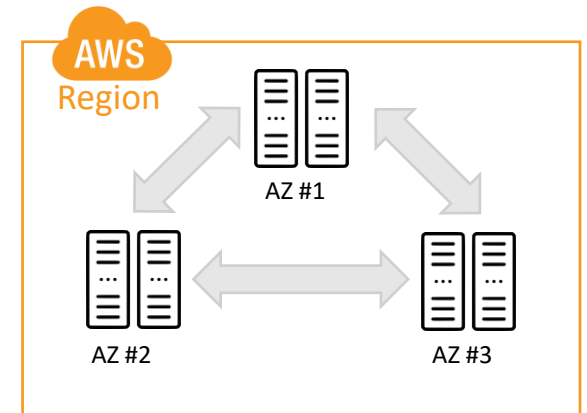
Each Region has multiple, isolated locations known as *Availability Zones*.

**Availability Zones** provide redundancy for AWS resources in that region, highly available, fault tolerant, and more scalability. AZs have low latency, high-bandwidth network connection, and support near real-time replication between AZs. All traffic is encrypted.

**High Availability** is creating an architecture in such a way that the system is always available or has the least amount of downtime as possible. If the app is running on 2 or more AZs, it means the app is highly available. Availability is normally expressed in 9's.

- 5 nines uptime means only 5 min downtime is allowed a year
- 4 nines uptime means less than an hour downtime is allowed a year

**Fault Tolerant** is the ability of your system to withstand failures in one or more of its components and still remain available. Asynchrony and decoupling using SQS increase fault tolerance.





# Edge servers

AWS has servers in 247+ Countries and Territories. Those servers are not many in that area to form an AZ or a region. Those servers are called **edge servers**. Edge servers play the following roles:

- AWS Global Accelerator takes advantage of edge servers and routes requests efficiently in the AWS network without bouncing on the internet.
- CloudFront caches your static contents all over the world on edge servers.
- You can run small code on Lambda@Edge which is a feature of CloudFront.

AWS Global Accelerator is a networking service that improves the availability and performance of the applications that you offer to your global users. (Source: [aws.amazon.com](https://aws.amazon.com/global-accelerator/)) Before:



After:



AWS Accelerator is taking advantage of edge servers to improve performance.  
[Building a global app with AWS Global Accelerator](#)

VPC



# Virtual Private Cloud (VPC)

VPC is an isolated virtual network where non-public AWS resources run. You have complete control over your networking such as the selection of your own local **IP** address range, the creation of **subnets**, and other networking components.

In other words, VPC is simply how you manage networking in the cloud. VPC is similar to traditional networking. VPC is mostly managed by a networking guy or the DevOps team. You just need to know its high level, big picture concepts.

## 2 types of resources

1. Public – It is just like a third-party API that you can call directly such as S3, DynamoDB, SNS, SQS, and so on (FaaS services). If valid authorization tokens are present, the API call is a success.
2. Non-public – You must launch those resources in VPC so that it gets an IP address so others can connect to it. It includes EC2, RDS, Load Balancers, and so on (IaaS and PaaS services).

# Subnets

A subnet is a sub-section of a network. Generally, it includes all the computers in a specific location like zip code for addressing houses.

A VPC includes many subnets. **A subnet is associated with an AZ.**

There are 2 types of subnets:

- Anyone can access to resources in **public** subnet directly from the internet.
- A **private** subnet is a safe environment where you can run back-end servers and databases securely. The internet (outsiders) cannot directly access resources in private subnet. The only way to access private resources is through the other resource in the public subnet. All resources in VPC talk to one another using private IPs even if they all have public IPs.

Subnets are written in [CIDR format](#).

Read more about [Subnets](#)

# CIDR ranges for the private network

## Most popular private IPs

CIDR	Starting and ending IPs	Total number of hosts
10.0.0.0/8	10.0.0.0 – 10.255.255.255	16,777,216
172.16.0.0/12	172.16.0.0 – 172.31.255.255	1,048,576
192.168.0.0/16	192.168.0.0 – 192.168.255.255	65,536

### Total address space

**200.100.10.0/24**  
(256 addresses)

200.100.10.0	200.100.10.1
200.100.10.2	200.100.10.3
200.100.10.4	200.100.10.5
200.100.10.6	200.100.10.7
⋮	⋮
200.100.10.252	200.100.10.253
200.100.10.254	200.100.10.255

**Before Subnetting**

### Partial address spaces

**200.100.10.0/25**  
(128 addresses)

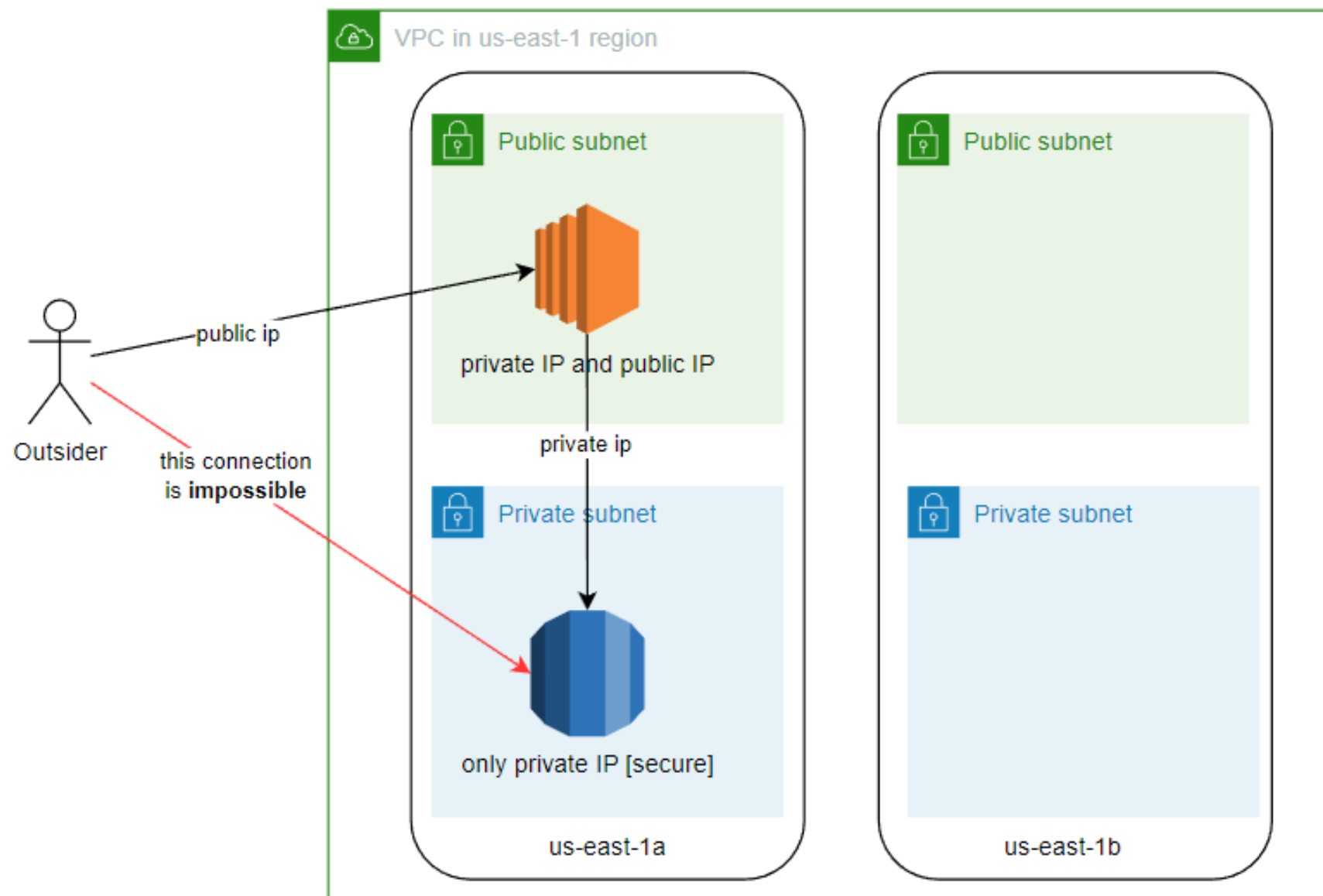
200.100.10.0	200.100.10.1
⋮	⋮
200.100.10.126	200.100.10.127

**200.100.10.128/25**  
(128 addresses)

200.100.10.128	200.100.10.129
⋮	⋮
200.100.10.254	200.100.10.255

**After Subnetting**





# VPC Security Layers

The VPC has 2 layers of security:

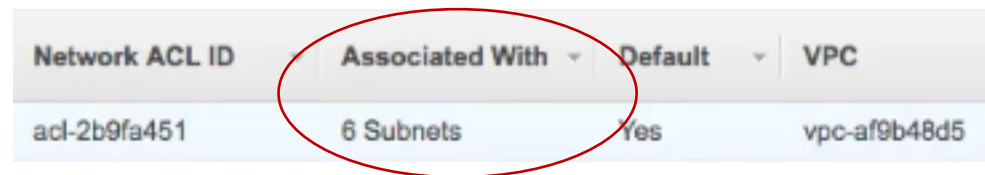
- Security Groups (SG) is a firewall on a resource level.
- Network Access Control Lists (NACL) is a firewall on a subnet layer. In one subnet, there are many resources.

To access your resource in the VPC, you must allow access on both layers.

You don't normally touch the NACL whereas the SG is the most important concept like IAM that developers work daily basis. If the resource is not responding, most likely, it is an issue with your SG that doesn't allow incoming access.

# Network Access Control Lists - NACL

NACL acts as a firewall for controlling traffic in and out of one or more subnets. Your default VPC already has an NACL in place and is associated with all default subnets.



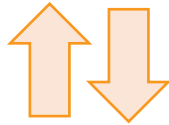
Network ACL ID	Associated With	Default	VPC
acl-2b9fa451	6 Subnets	Yes	vpc-af9b48d5

To access an EC2 instance from the internet, the request must pass the NACL security layer. If there is a rule that denies the request on the way, you cannot access the instance. Once it passed the NACL layer, then the request also has to pass Security Groups (SG) layer associated with the instance.

Read more about [Network ACLs](#)



The default NACL allows all traffic, both inbound and outbound



Network Access Control List - NACL



EC2

Subnet 1 (public)



EC2

Subnet 2 (public)

# NACL Rules

- The default NACL allows all traffic to the default subnets.
- Rules are evaluated from lowest to highest based on rule #. The first rule found that applies to the traffic type is immediately applied, regardless of any rules that come after it.
- The NACL allows or denies traffic from entering a subnet. Once inside the subnet, other AWS resources may have additional security layers (security groups).

Inbound	Rule #	Type	Protocol	Port Range	Source	Allow / Deny	All traffic is allowed
	100	ALL Traffic	ALL	ALL	0.0.0.0/0	ALLOW	
	*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY	
Inbound	Rule #	Type	Protocol	Port Range	Source	Allow / Deny	All traffic is denied
	90	SSH (22)	TCP (6)	22	0.0.0.0/0	DENY	
	100	SSH (22)	TCP (6)	22	0.0.0.0/0	ALLOW	
	*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY	

# Security Groups (SG)

Security groups are found on the **instance level**. They act as a **virtual firewall** that controls the traffic for one or more instances.

You add rules to security groups to **allow traffic** to (or from) its associated resource. By default, there is no inbound, which means everything is denied. You cannot write a deny rule like the NACL.

You can attach multiple SGs to an instance. All the rules from the security groups that are associated with the instance are evaluated.

By default, all outbound access is allowed. That is fine. We don't touch the outbound rule of the SG.

Read more: [Security Groups](#)

# Inbound/Outbound Rules

Name

Group ID

Group Name

VPC ID

sg-b63da5fa

default

vpc-af9b48d5

Description

Inbound

Outbound

Tags

Edit

Type ⓘ

Protocol ⓘ

Port Range ⓘ

Source ⓘ

Description ⓘ

All traffic

All

All

your IP

Description

Inbound

Outbound

Tags

Edit

Type ⓘ

Protocol ⓘ

Port Range ⓘ

Destination ⓘ

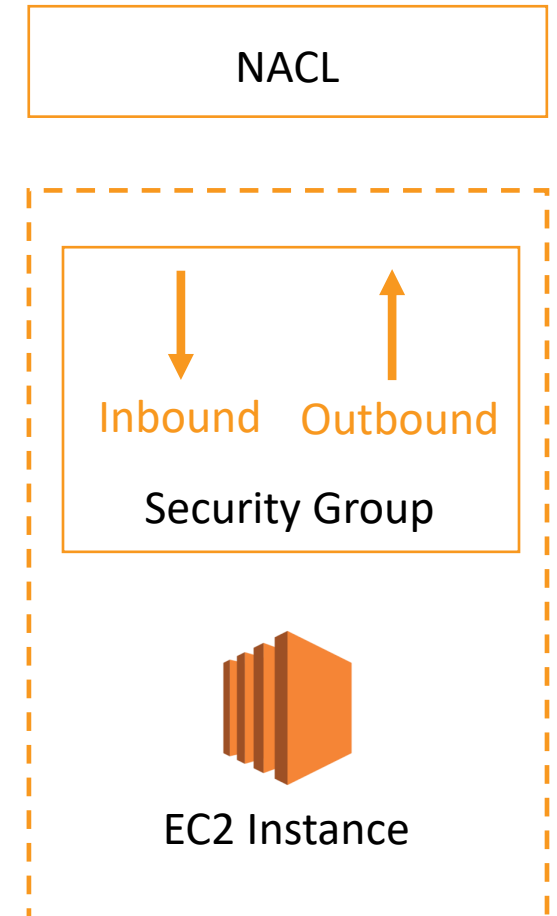
Description ⓘ

All traffic

All

All

0.0.0.0/0



NACL	Security Group
NACL can be understood as the firewall or protection for the <b>subnet</b> .	Security group can be understood as a firewall to protect EC2 <b>instances</b> .
These are <b>stateless</b> , meaning any change applied to an incoming rule isn't automatically applied to an outgoing rule. Example: If a request comes through port 80, it should be explicitly indicated that its outgoing response would be the same port 80.	These are <b>stateful</b> , which means any changes which are applied to an incoming rule is automatically applied to a rule which is outgoing. Example: If the incoming port of a request is 80, the outgoing response of that request is also 80 (it is opened automatically) by default.
NACL supports <b>allow</b> and <b>deny</b> rules. Denial of rules can be explicitly mentioned, so that when the layer sees a specific IP address, it blocks connecting to it.	SG supports <b>only allow rules</b> , and the default behavior is denial of all.
In case of NACL, the rules are applied in the order of their priority, wherein priority is indicated by the number the rule is assigned. This means every rule is evaluated based on the priority it has.	In case of a security group, all the rules are applied to an instance.



EC2



# AWS Elastic Cloud Compute (EC2)

EC2 is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Basically, a virtual computer, very similar to the desktop or laptop you use at home, and commonly referred to as an **instance**.

You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.

Amazon EC2 autoscaling enables you to scale in or out to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Read more: [Amazon EC2](#)

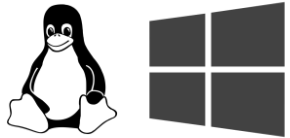
# Computer and EC2 Instance



Computer



**EC2 Instance**



Operating System  
**AMIs**  
(Linux or Windows)



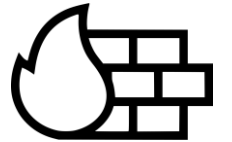
CPU & RAM  
**Instance Type**



Hard Drive  
**EBS**



Network Adapter  
**ENI**



Firewall  
**Security Groups**

# Amazon Machine Image - AMI

Preconfigured and required to launch an EC2 instance that includes an operating system, software packages, and other required settings.



Amazon Machine Image (AMI) provides the information required to launch an instance. You specify an AMI when you launch an instance.

In class, I have limited time to demonstrate all components of EC2. You can practice all at home if you are dedicated. Always highly encourage you to play with the AWS console, read descriptions and experiment with all the things you see on the AWS EC2 console. Similarly, be bold and play with all services. You have a free account.

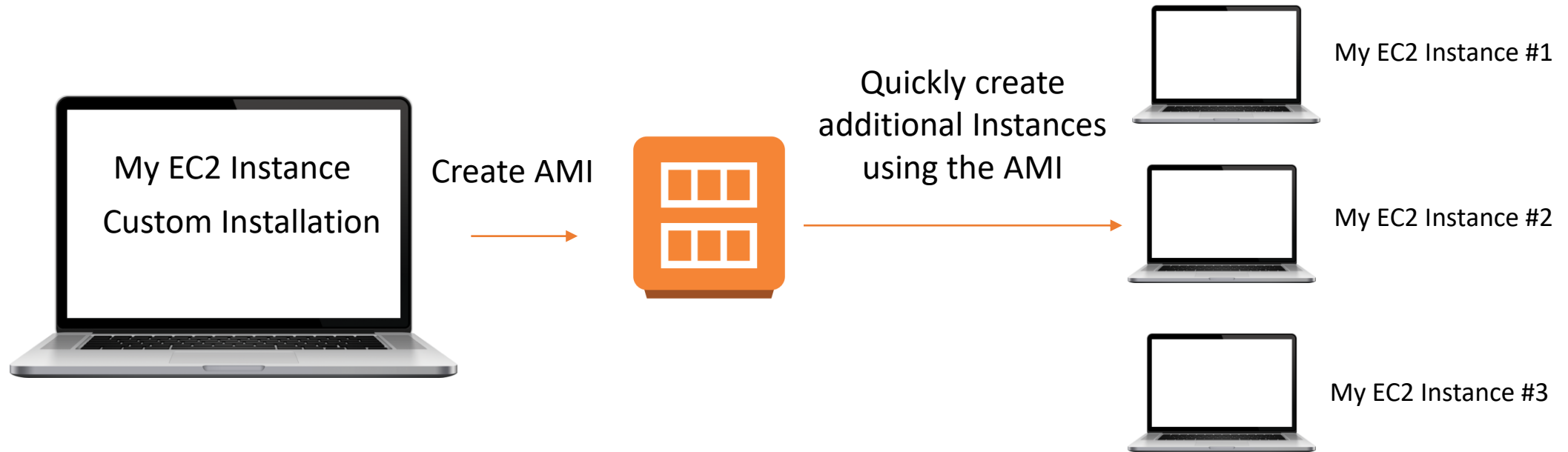
Read more: [AMI](#)

# Custom AMI

Assume that you need to run your app in hundreds of servers. It is not possible to configure every single server one by one. Instead, you can create a custom AMI and use it on as many servers as you want.

Custom AMIs contain all configurations, dependencies, and environments required to run your app.

# Understanding AMI



# Instance Types

The Instance Type determines the underlying hardware such as CPU, GPU, RAM, network and disk read/write capacity.

Each instance type offers different compute, memory, and storage, graphic capabilities and is grouped into instance families based on these capabilities.

NVIDIA instances have a GPU (accelerated computing) that works best for ML since a GPU provides high parallel and simultaneous computing.

The same service price varies depending on the region. For example, the same instance in the USA costs higher than the one in Japan.

With EC2 (IaaS), you have more control such as dictating the underlying hardware. Whereas in Lambda (FaaS) and ECS (Container as a Service), you can not control it.

Read more: [Instance Types](#)

# General-purpose and memory-optimized instance type

**General-purpose** instances provide a balance of compute, memory, and networking resources. It can be used for web servers and code repositories. General-purpose instance types start with **T** and **M**.

**Memory-optimized** instances are good for applications that process large amounts of data in memory such as caching. Memory-optimized instance types start with **R** and **X**.



# Compute-optimized instance type

Compute-optimized instances are ideal for compute-bound applications that benefit from high-performance processors. It is good for batch processing workloads, media transcoding, high-performance web servers, high-performance computing (**HPC**), scientific modeling, dedicated gaming servers, ad server engines, machine learning inference, and other compute-intensive applications. Compute-optimized instance types start with **C**.

# Accelerated computing instance type

Accelerated computing instances use hardware accelerators (GPUs) or co-processors to perform functions such as floating point number calculations, graphics processing, or data pattern matching.

Both accelerated computing and compute-optimized instance types provide high computing power. But the way how they provide the computing power is different. Accelerated computing instances have GPUs which are accelerators whereas the compute-optimized ones have good high-performance processors. Accelerated computing instance types start with **P** and **G**.

# Storage-optimized instance type

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. Storage-optimized instance types start with **I** and **D**.

Do not configure EC2 as a database! Instead, use Relation Database Service which provides instance types designated for databases.

# Elastic Block Storage - EBS

EBS is a storage volume for an EC2 instance. It provides block level storage volumes for use with EC2 instances.

EBS volumes are **highly available and reliable** storage volumes that can be attached to any running instance that is **in the same Availability Zone**

EBS volumes that are attached to an EC2 instance are exposed as storage volumes that persist **independently** from the life of the instance. It charges independently.



=



Read more: [Amazon EBS Volumes](#)

# Input/Output Operations Per Second - IOPS

IOPS is the amount of data that can be written to or retrieved from EBS per second.

The operations are measured in KiB, and the underlying drive technology determines the maximum amount of data that a volume type counts as a single IO.

More IOPS means better volume performance (faster R/W speeds).

More expensive. Recommended for production environment.

# Types of EBS volumes

- **Provisioned IOPS** – The fastest and most expensive where you can specify IOPS. Good for production servers.
- **General Purpose** – It provides a moderate amount of speed. You can not specify IOPS. The IOPS depends on the volume size. Good for development environment servers.
- **HDD and Magnetic** – The cheapest option and slow. It is good when storing large amounts of data and archiving and you don't need much performance.

# Create Additional Volume

[Volumes](#) > Create Volume

## Create Volume

Volume Type General Purpose SSD (gp2) ⓘ

Size (GiB) 100 (Min: 1 GiB, Max: 16384 GiB)

IOPS 300 / 3000 (Baseline of 3 IOPS per GiB with a minimum of 100 IOPS, burstable to 3000 IOPS)

Availability Zone\* us-east-1a ⓘ

Throughput (MB/s) Not applicable ⓘ

Snapshot ID Select a snapshot ⓘ ⓘ

Encryption ☐ Encrypt this volume ⓘ

# Amazon Elastic File System (Amazon EFS)

It is an elastic file system that lets you share file data without provisioning or managing storage.

Amazon EFS is designed to provide massively parallel **shared** access to thousands of Amazon EC2 instances.

Amazon EFS is well suited to support a broad spectrum of use cases from home directories to business-critical applications.

You need to set an Security Group that allows access from a fleet of EC2 instances on the EFS volume.

Read more: [Amazon EFS](#)



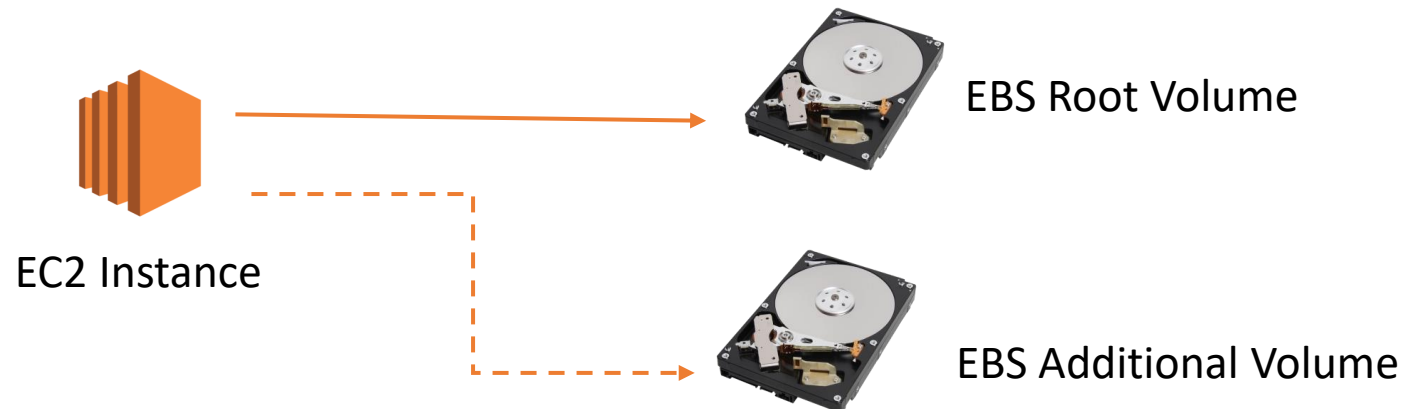


# Root vs. Additional EBS Volumes

Every EC2 instance must have a **root volume** that the AMI is restored.

You can add **additional EBS Volumes** to an instance at any time. Any additional volume can be attached or detached from the instance at any time.

Additional EBS volumes are NOT deleted (the default) and you still pay when the instance is terminated whereas the root volume gets deleted when the instance is terminated.

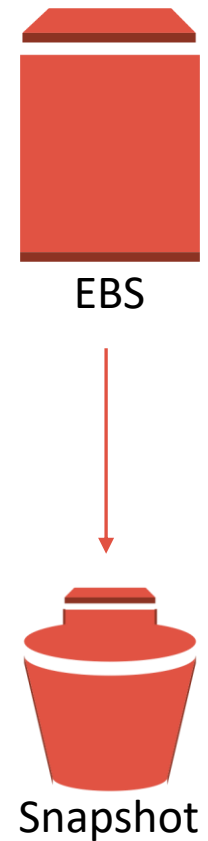


# Snapshots

A snapshot is an **image** of an EBS volume that can be stored as an **incremental backup** of the volume or used to create a duplicate.

A snapshot is not an active EBS volume. You cannot attach or detach a snapshot to an EC2 instance.

To restore a snapshot, you need to create a new EBS volume using the snapshot as its template.



# AMI vs Snapshots

---

## AMI

An entire EC2 instance definition that includes all EBS snapshots plus some metadata like kernel, AMI name, description, block device mappings, and more.

Could be used to deploy your applications on different machines easily.

## EBS Snapshot

Backup of a single volume.

You need to recover and mount a snapshot to an EC2 instance.

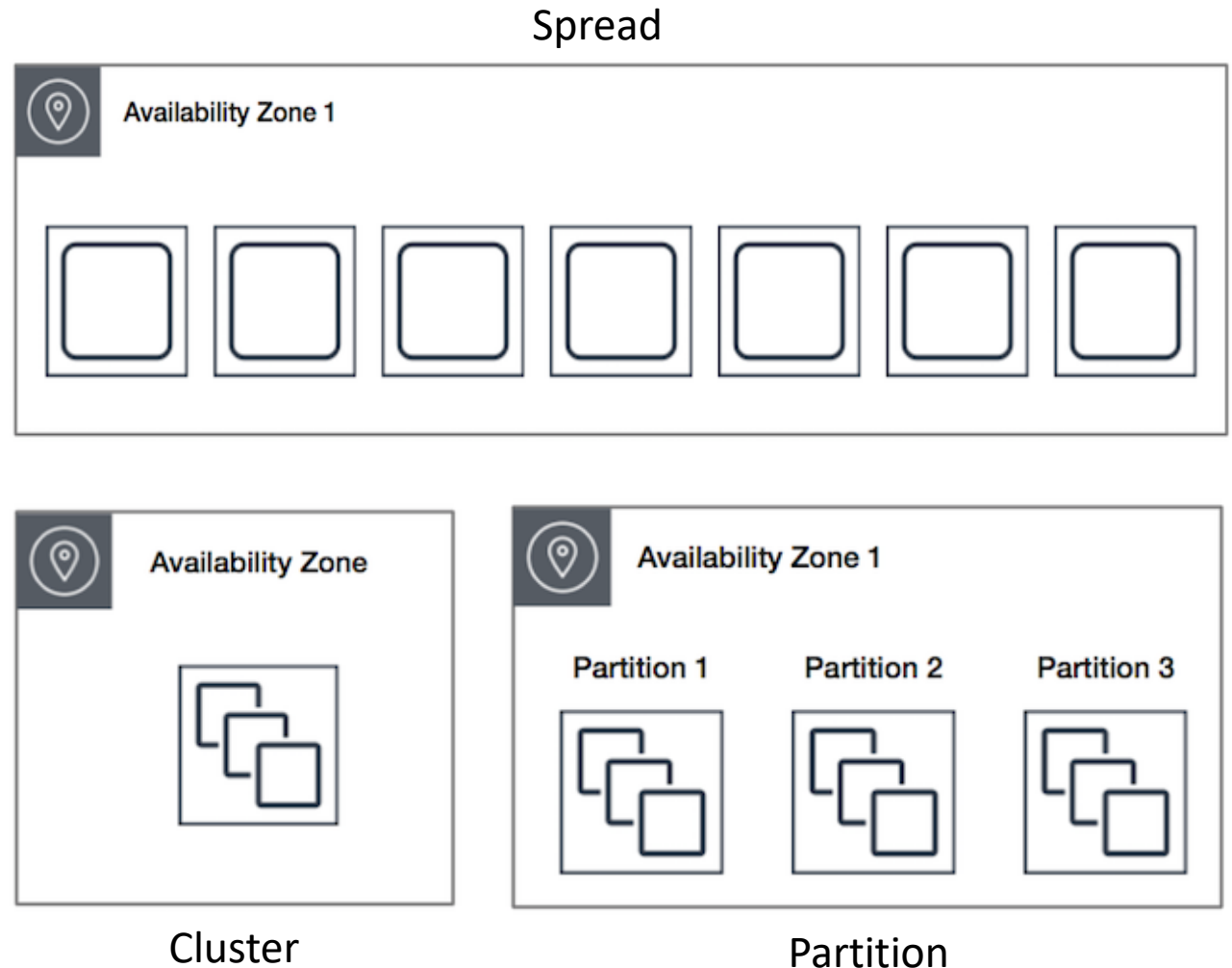
# Placement groups

A placement group is a logical grouping of instances within a single AZ that benefit from low network latency and high network throughput.

**Cluster** – Packs instances close together.  
*Low latency and high performance.*

**Partition** – Spreads instances across logical partitions. Partitions don't share the underlying hardware. Large distributed and replicated workloads such as Hadoop, Cassandra, and Kafka.

**Spread** – Places a small group of instances across distinct underlying hardware to *reduce correlated failure*.



# IP Addressing

An IP address is the EC2 instance address on the network.

## **Private IP Address:**

EC2 instance receives the private IP from the subnet.

All EC2 instances (all devices in the network) have a private IP address.

Private IP addresses allow instances to communicate with resources in the same network.

## **Public IP address:**

All EC2 Instances can be launched with or without a public IP address.

Public IP addresses are required for the instance to communicate with the Internet.

Read more: [IP Addressing](#)

# Elastic IP

Elastic IP is static public IP.

When you stop and then start an EC2 instance, it may 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.
- You can attach it to one instance at a time.
- By default, you can have 5 elastic IPs in AWS.
- It charges \$1 when you are NOT using it for a week. When used, it is free. Every week,

# EC2 Instance Options - On-Demand

On-demand purchasing allows you to choose any instance type you like and provision/terminate it at any time (on-demand).

The most expensive purchasing option.

You are only charged when it is running (billed by the hour).

EBS volume is billed separately. 10 GB general-purpose SSD costs \$1 a month.

NAT gateway is a VPC component that allows instances in private subnets to access the internet which is billed under EC2.

Read more: [EC2 pricing](#)

# EC2 Instance Options - Reserved

Reserved purchasing allows you to purchase an instance for a set time period of 1 or 3 years.

This allows for a significant price discount over using on-demand.

You can select to pay upfront, partial upfront, or no upfront.

Once you buy a reserved instance, you own it for the selected time period and are responsible for the entire price - regardless of how often you use it.

About 20% savings with a one-year term and 30% savings with a three-year term.



# EC2 Instance Options - Spot

Cheapest but not reliable.

Amazon sells unused instances for short amounts of time at a substantial discount.

Spot pricing is a way for you to bid on an instance type, then only pay for and use that instance when the spot price is equal to or below your bid price.

Spot prices fluctuate based on supply and demand.

A provisioned instance automatically terminates when the spot price is greater than your bid price.

You can use Spot instances along with other types of instances in your cluster or a fleet of servers for a fault-tolerant application. So, you save a lot.



IAM

# IAM

**IAM Policies** are **permissions** that you can assigned to any User (Person), Group (Person), and Roles (Service).

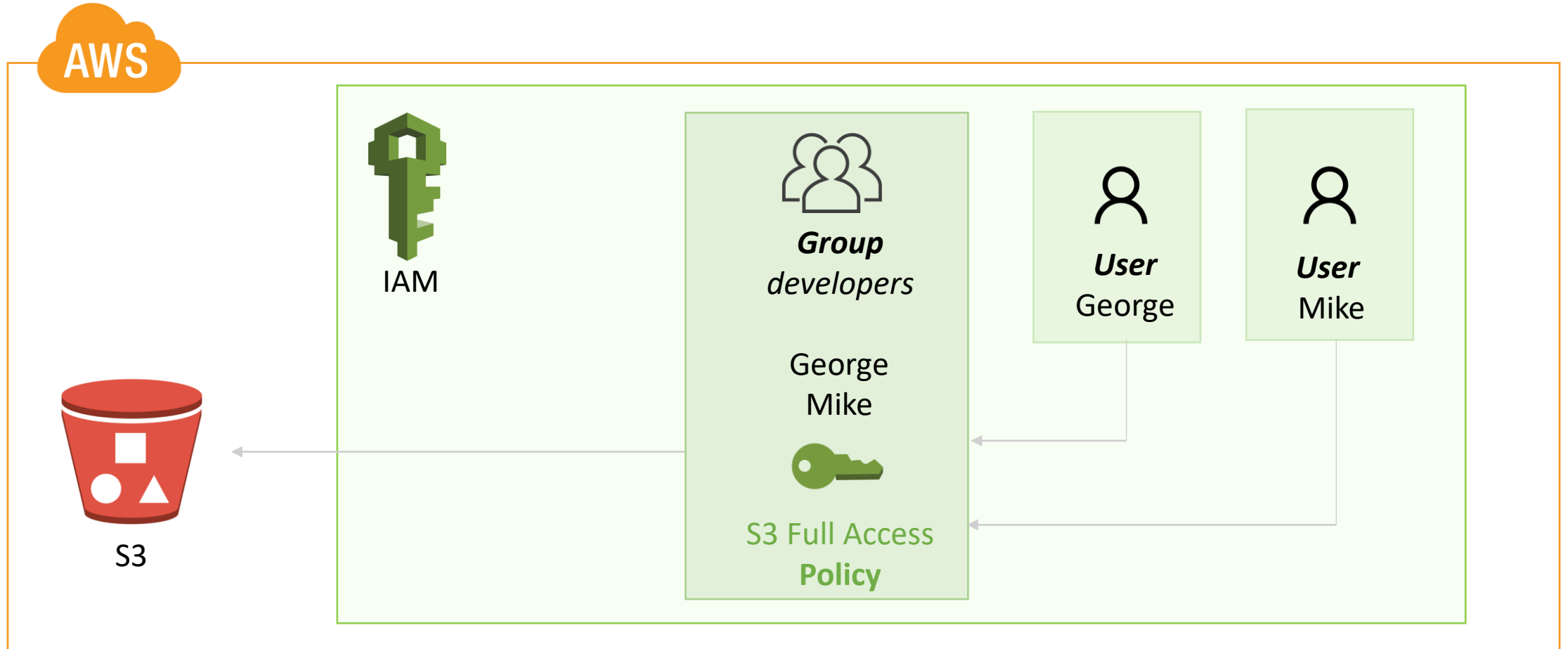
We don't attach an IAM Policy to a Service directly, instead, we would need to use a **Role**. Inside the role, we define IAM policies and associate that role with the resources (service).

There are 2 types of IAM policies:

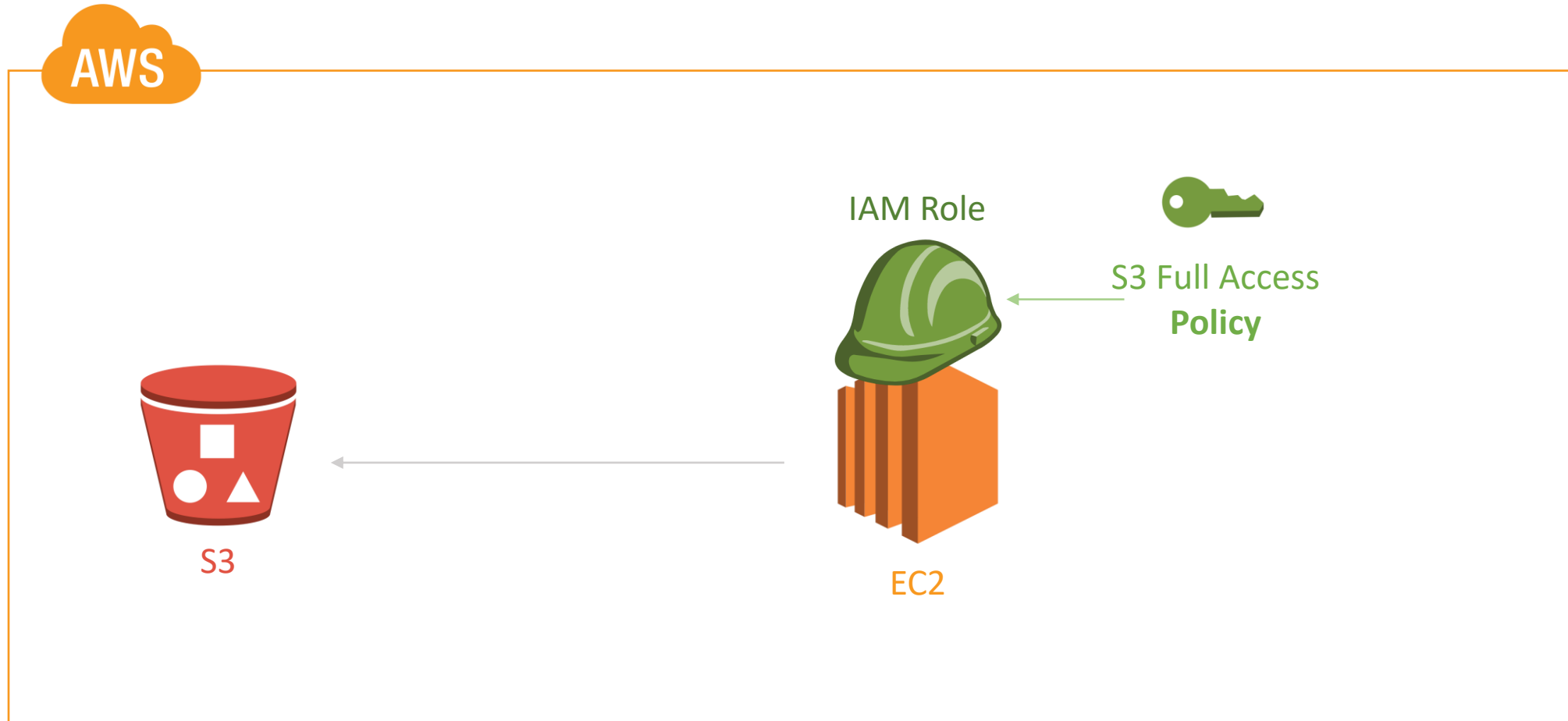
- **AWS managed** - It is fine for POC. You shouldn't use it. The best practice is to write your own user-managed policies.
- **User managed** - recommended, always used in real life or production.

Read More about: [IAM Policies](#)

# AWS IAM Group



# AWS IAM Role



# IAM Identity Center

AWS started recommending to use the IAM Identity Center for giving AWS access to developers. IAM Identity Center benefits:

- Multi-account management
  - Better cost management
  - Better access management (read-only to production)
  - Remediate service quote per account
- Single Sign On (SSO) – Not only AWS, you can also access popular apps such as Slack, Asana, GitHub, so on

# Two ways to allow/deny access in IAM

1. **Identity-based** policies – Attach policies to IAM identities (users, groups, or roles) as we talked about in the previous 3 slides.
2. **Resource-based** policies – Attach inline policies to resources (such as S3, SQS, SNS) directly. Resource-based policies have a **principal** tag that defines who can the actions defined.

If you want to connect EC2 with S3, you can do it in 2 ways. Add a role to EC2, that is what identity-based authorization is. Alternatively, you can write a resource-based policy on the S3, that is what resource-based authorization is.

It is confusing in the beginning if you use identity or resource-based authorization. Confusion is a great ingredient of growth. It comes naturally with an experience. After a year, it will become easy.

# IAM Policy structure

The **Action** element is the specific API action for which you are granting or denying permission

```
{  
  "Statement": [{  
    "Effect": "effect",  
    "Action": "action",  
    "Resource": "arn",  
    "Condition": {  
      "condition": {  
        "key": "value"  
      }  
    }  
  }  
]  
}
```

The **Effect** element can be Allow or Deny

The **Resource** element specifies the resource that's affected by the action

The **Condition** element is optional and can be used to control when your policy is in effect



# IAM JSON policy elements: Condition

The Condition element (or Condition block) lets you specify conditions for when a policy is in effect.

In the Condition element, you build expressions in which you use condition operators (equal, less than, etc.) to match the condition keys and values in the policy against keys and values in the request context.

Learn more: [IAM JSON policy elements: Condition](#)

```
"Condition" : { "{condition-operator}" : { "{condition-key}" : "{condition-value}" }}
```

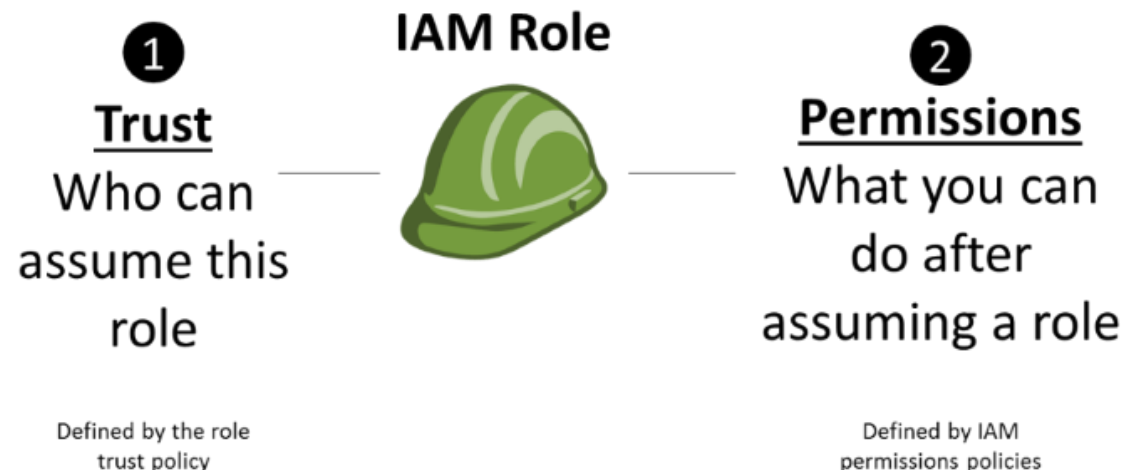
```
"Condition" : { "StringEqualsIgnoreCase" : { "aws:username" : "johndoe" }}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "rds:DescribeDBInstances",
        "rds:DescribeDBClusters",
        "rds:DescribeGlobalClusters"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "rds:RebootDBInstance",
        "rds:StartDBInstance",
        "rds:StopDBInstance"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/Department": "DBAdmins",
          "rds:db-tag/Environment": "Production"
        }
      }
    }
  ]
}
```

# IAM role

IAM role is similar to IAM user but you assign that to a resource, not a developer. There is a trust policy that defines which service can assume (use) that role. An IAM user and role have one thing in common, permission policies.

When the service in the trust policy assumes the role, AWS STS (Security Token Service) returns **temporary** tokens (access key id, secret access key, and **session token**), and those tokens are rotated automatically.



# AWS Security Token Service (AWS STS)

Under the hood, tokens are generated and used to access AWS services. STS is a web service that enables you to request **temporary** credentials for IAM role. The example is AWS Academy, you receive a temporary token.

The temporary credentials consist of:

1. **Access key ID** - Access keys are long-term credentials for an IAM user. Like username.
2. **Secret access key** - Like password.
3. **Session token** - Validates temporary credentials.
4. **Duration** - defines how long the temporary credentials lasts. Most cases 12 hours. 15-min is min. You will not see these in the permanent tokens for users.

# STS - AssumeRole

The underlying service makes an implicit “AssumeRole” call to STS on behalf of the resource and fetch the temporary tokens. It all happens under the hood.

For instance, when fetching images from S3 in EC2, EC2 makes the AssumeRole call to STS, then receives the token and provides the token to S3. You don't have to manually rotate it. The rotation is done by AWS.

# IAM User vs Role

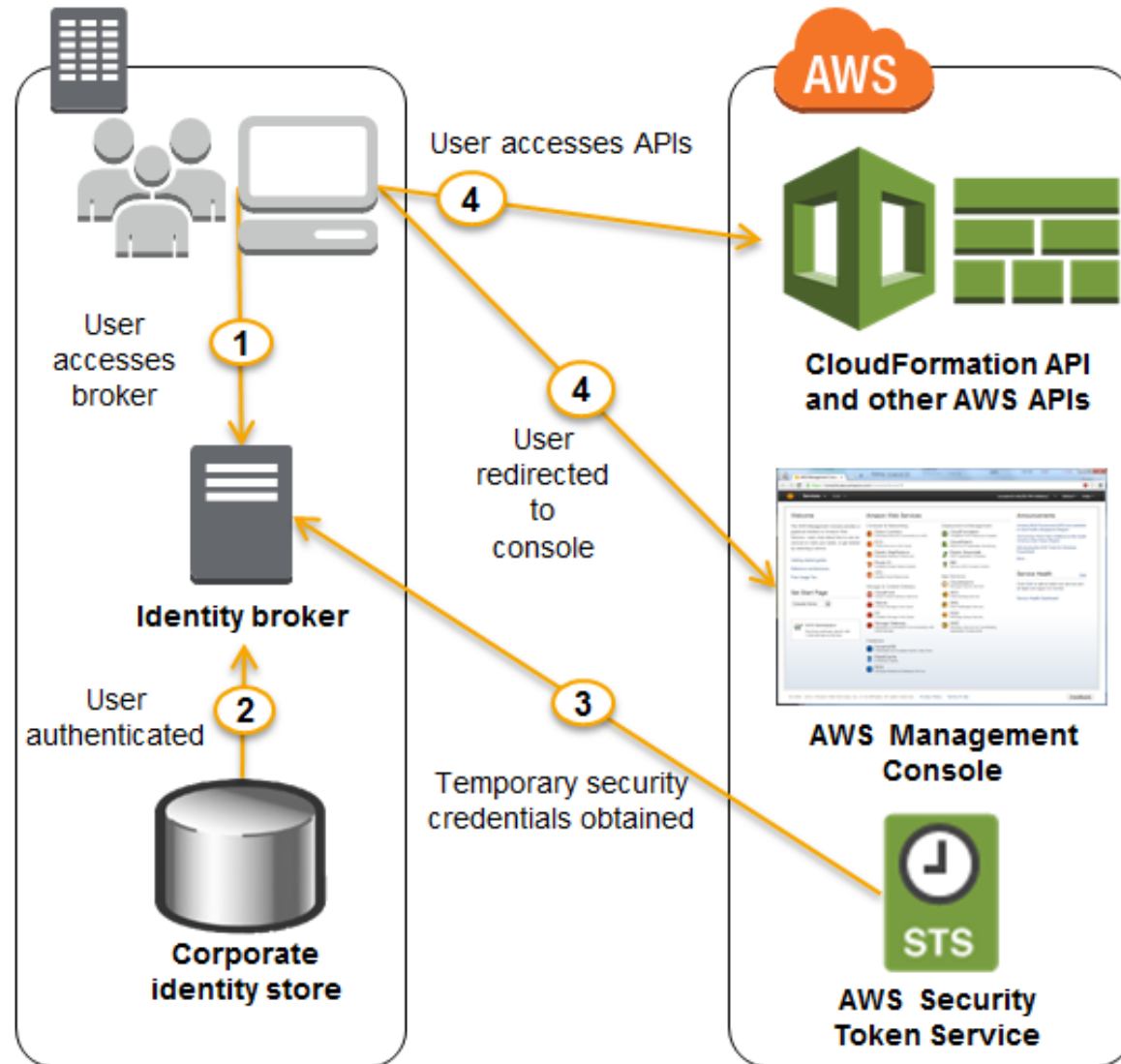
IAM User	IAM Role
IAM entity assigned to a <b>person</b> .	IAM entity assigned to a <b>service</b> . It has a trust policy that specifies what services can use the role.
<b>Tokens are permanent.</b> Not recommended.	<b>Tokens are temporary.</b> Tokens are generated by AWS STS. It has an extra token “aws_session_token”.

# Identity federation

**Identity federation** grants external identities secure access to resources in your AWS account. These external identities can come from your corporate identity provider such as Microsoft Active Directory and Instructure (AWS Academy).

**Federated users** (external identities) are users you manage outside of AWS in your corporate directory, but to whom you grant access to your AWS account using **temporary security credentials** (Role and STS). AWS Academy is an example.

# Federated users and temporary security credentials STS





# IAM Summary

IAM is about:

1. allow/deny
2. what actions?
3. on which resources?
4. who?
  - a. principle propriety (resource-based)
  - b. IAM user or role (identity-based)
5. condition (optional)