

Module - 1 - AWS Cloud Foundations & IAM

What's AWS?

- AWS (Amazon Web Services) is a Cloud Provider
 - They provide you with servers and services that you can use on demand and scale easily
 - AWS has revolutionized IT over time
 - AWS powers some of the biggest websites in the world
 - Amazon.com
 - Netflix
-

AWS Global Infrastructure

- The AWS Cloud spans 105 Availability Zones within 33 geographic regions, with announced plans for 21 more Availability Zones and seven more AWS Regions.

33 launched Regions

each with multiple Availability Zones

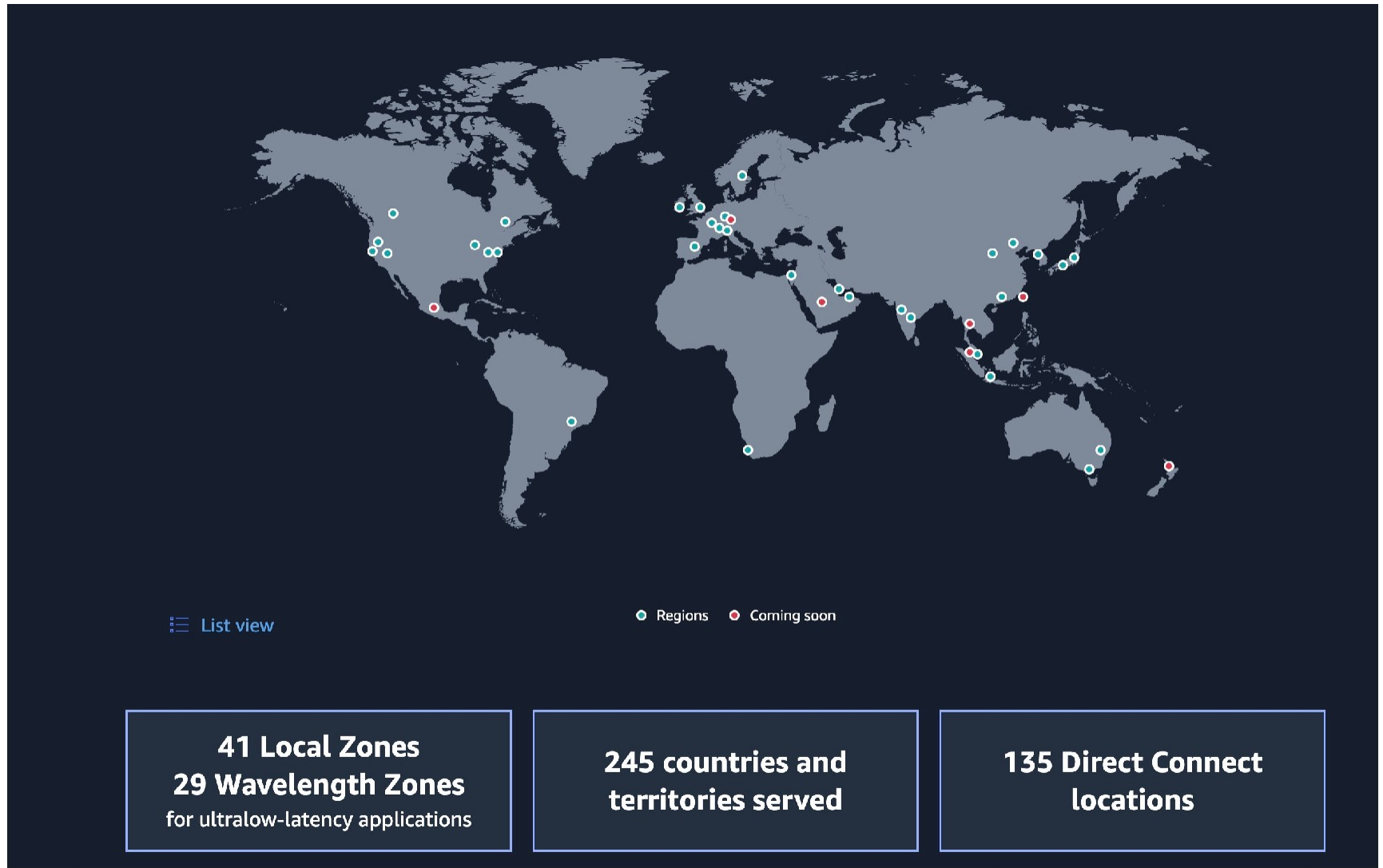
105 Availability Zones

600+ CloudFront POPs

and 13 Regional edge caches

<https://infrastructure.aws/>

AWS Global Infrastructure Map



AWS Global Infrastructure – Asia Pacific

Asia Pacific

Asia Pacific (Singapore) Region

Availability Zones: 3

Launched 2010

Asia Pacific (Tokyo) Region

Availability Zones: 4

Launched 2011

Asia Pacific (Seoul) Region

Availability Zones: 4

Launched 2016

Asia Pacific (Mumbai) Region

Availability Zones: 3

Launched 2016

Asia Pacific (Hong Kong) Region

Availability Zones: 3

Launched 2019

Asia Pacific (Osaka) Region

Availability Zones: 3

Launched 2021

Asia Pacific (Jakarta) Region

Availability Zones: 3

Launched 2021

AWS Asia Pacific (Hyderabad) Region

Availability Zones: 3

Launched 2022

Mainland China (Beijing) Region

Availability Zones: 3

[Learn more at www.amazonaws.cn](https://www.amazonaws.cn)

Mainland China (Ningxia) Region

Availability Zones: 3

[Learn more at www.amazonaws.cn](https://www.amazonaws.cn)

AWS Edge Locations

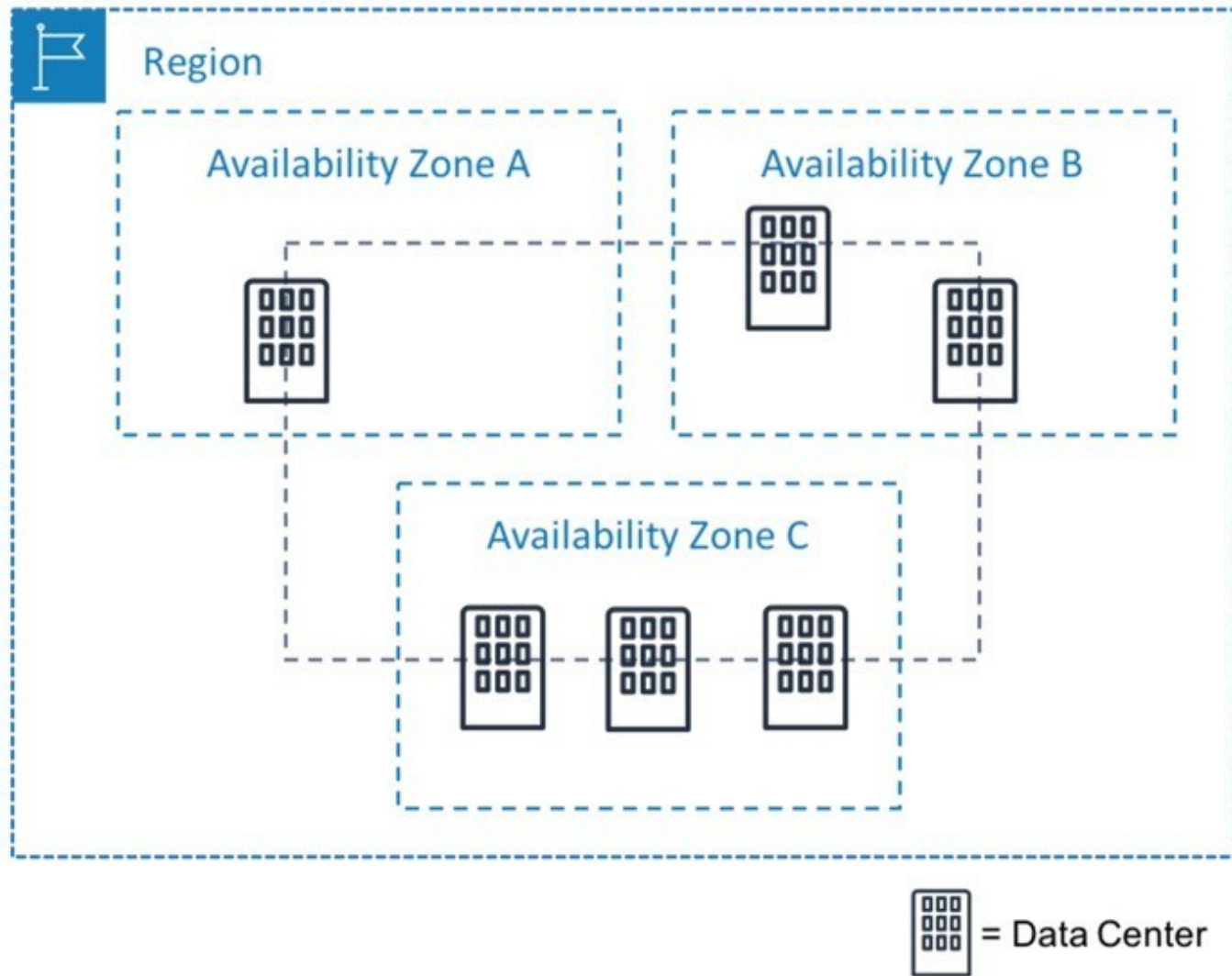
Edge locations - Bangalore, [India](#); Bangkok, Thailand; Chennai, [India](#); Beijing, China, Shanghai, China, Ningxia, China, Shenzhen, China, Hong Kong SAR, China; Hanoi, Vietnam; Ho Chi Minh City, Vietnam; Hyderabad, [India](#); Jakarta, Indonesia; Kolkata, [India](#); Kuala Lumpur, Malaysia; Manila, The Philippines; Mumbai, [India](#); New Delhi, [India](#); Osaka, Japan; Pune, [India](#); Seoul, Korea; Singapore; Taipei, Taiwan; Taoyuan, Taiwan; Tokyo, Japan; Zhongwei, China

Regional Edge Caches - Mumbai, [India](#); Seoul, Korea; Singapore; Tokyo, Japan

[Learn more about the Global Edge Network »](#)

[See detailed list of offerings at all AWS locations](#)

AWS Regions

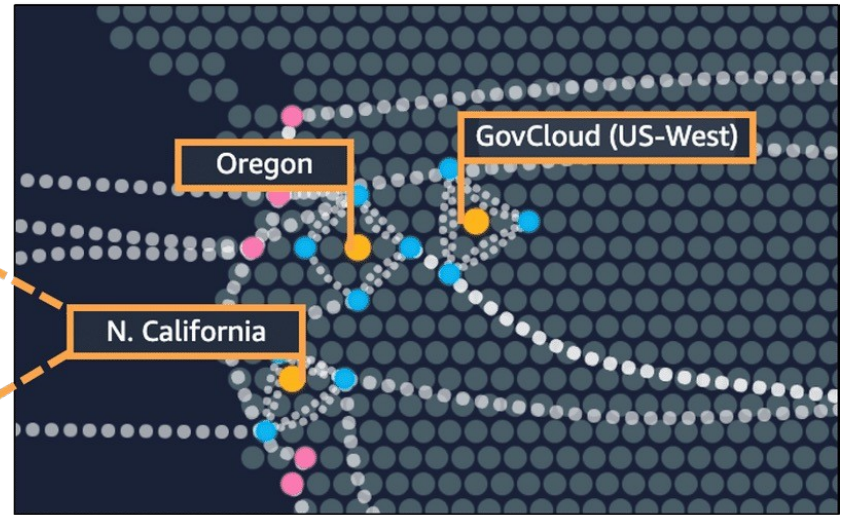
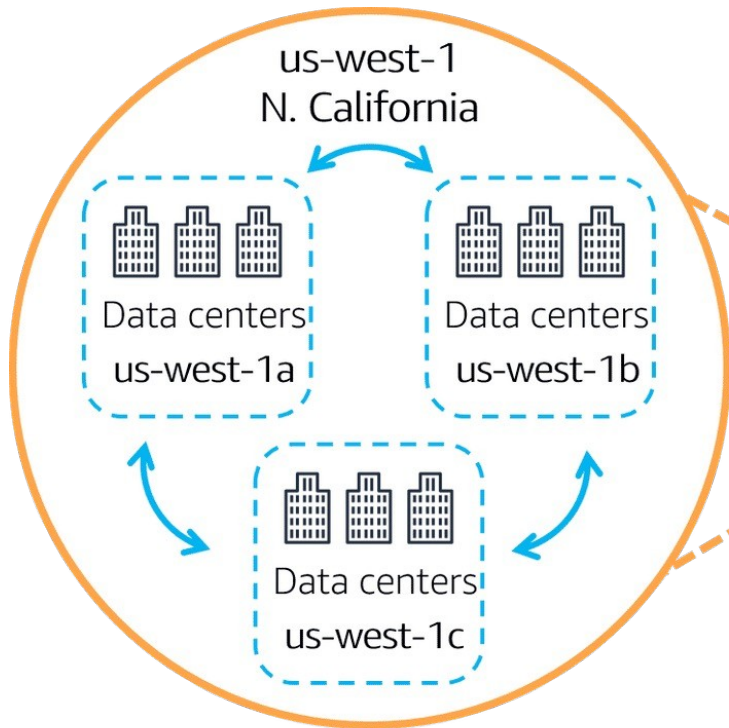


AWS Regions

- AWS Regions are separate geographic areas that AWS uses to house its infrastructure.
- These are distributed around the world so that customers can choose a region closest to them in order to host their cloud infrastructure there.
- The closer your region is to you, the better, so that you can reduce network latency as much as possible for your end-users.
- Some key characteristics of AWS Regions:
 - Each Region is completely independent and isolated from other Regions
 - Regions are designed to be isolated from each other to achieve fault tolerance and stability
 - Resources are not replicated across Regions unless specifically configured to do so
 - Certain Regions have more services and features available than others

AWS Regions

- **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
-

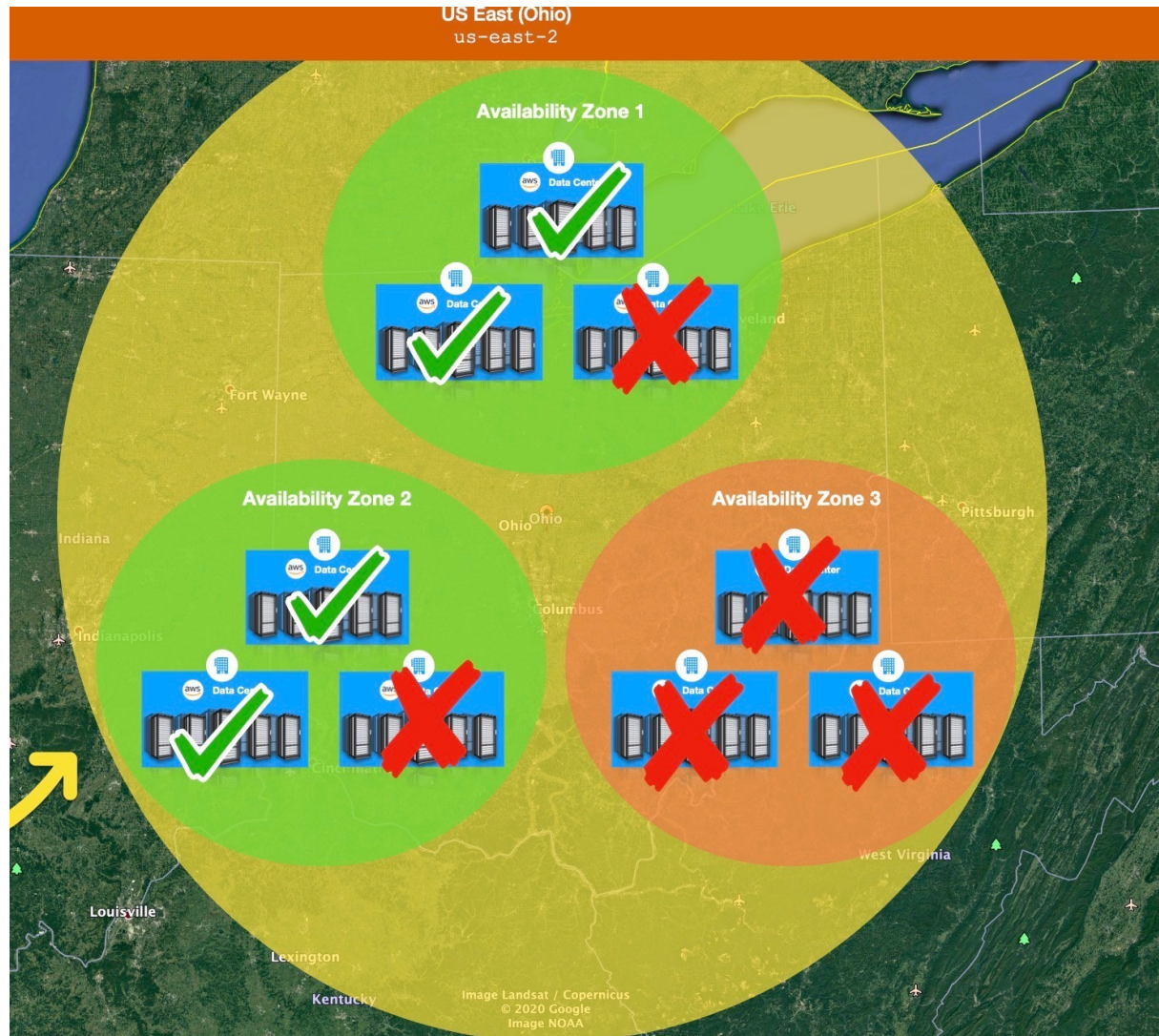


- Regions
- Availability Zones

What are Availability Zones (AZs)?

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

AWS Availability Zones

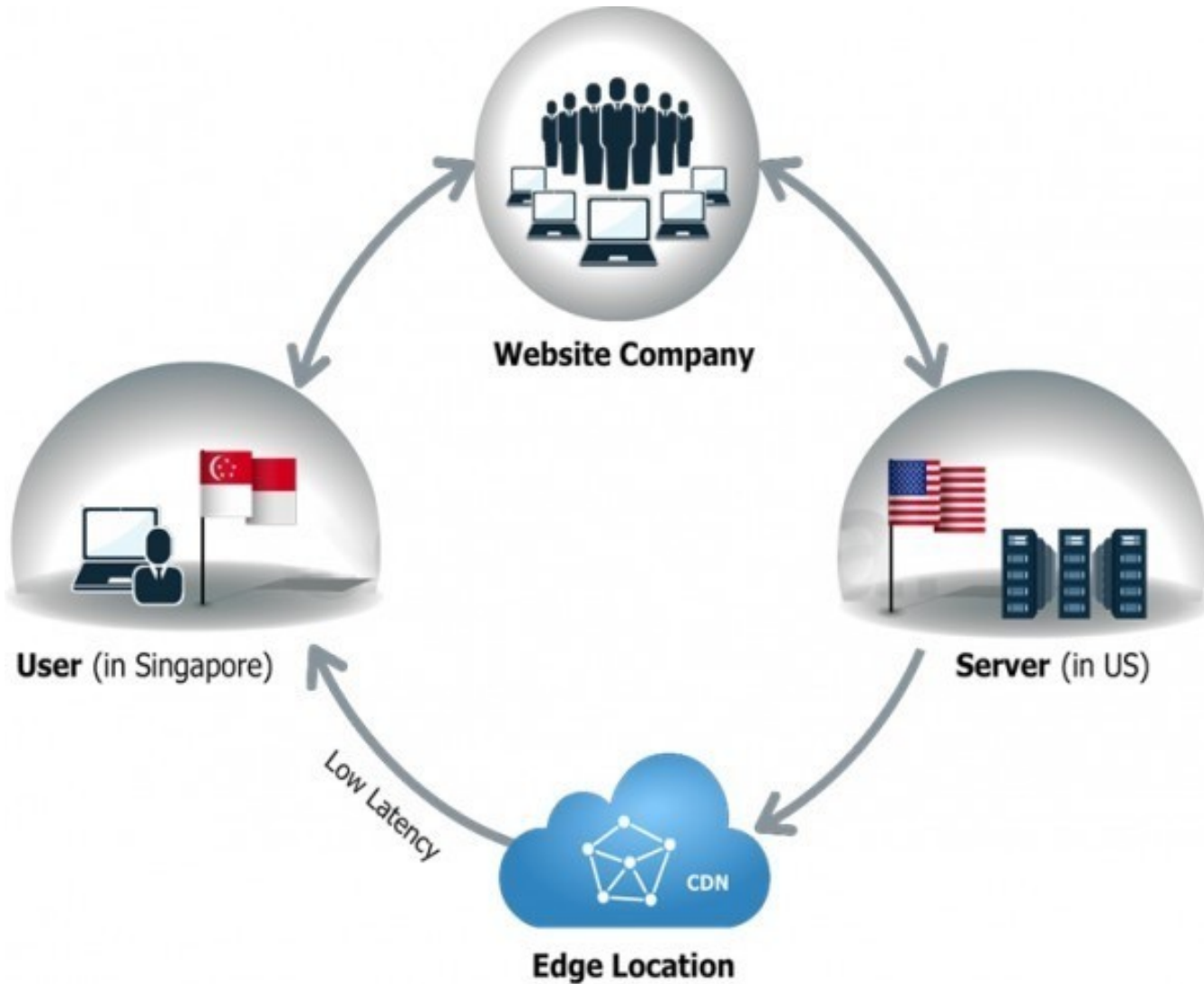


AWS Cloud Availability Zones

- Availability Zone is a single Data Center or a group of Data Centers in a region.
- In an Availability Zone the Data Centers are located many miles apart from each other (Min 60 km – Max 100 km)
- Having them apart reduces the risk of them all going down if a disaster happens in the region.
- Simultaneously, have the Data Center(s) close enough to have low latency.

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

AWS Edge Location



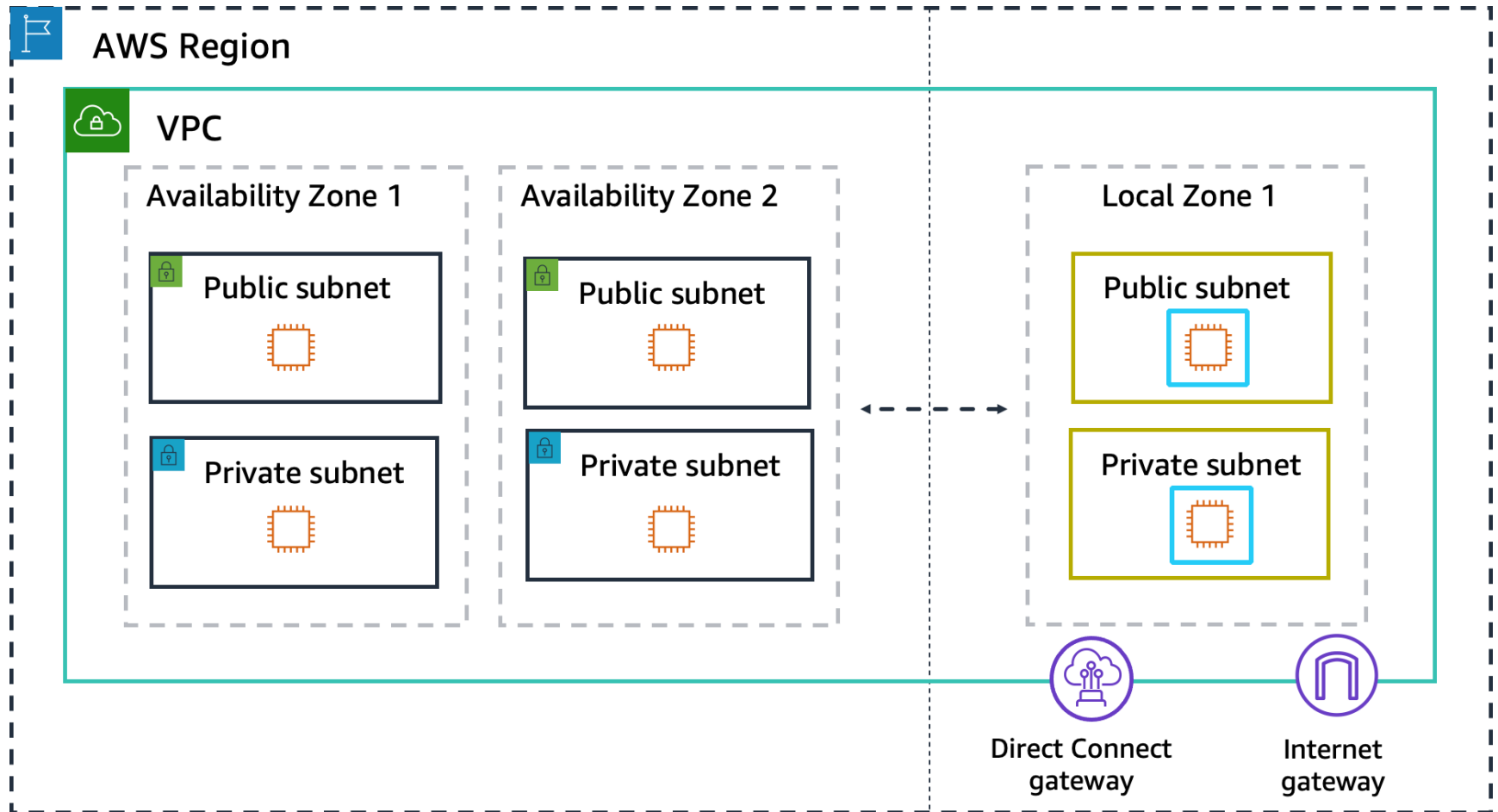
AWS Edge Location

- AWS Edge Locations are data centers strategically placed around the world to deliver content and services with the lowest possible latency to users.
 - AWS Edge Locations are a key part of the AWS global infrastructure that enables delivering low-latency services to end users worldwide through services like CloudFront.
 - Their strategic placement around the world is a major advantage for latency-sensitive applications.
-

AWS Edge Location

- Key points about AWS Edge Locations:
 - ❑ They are used by various AWS services to cache content, serve DNS responses, filter traffic, and more.
 - ❑ Placing these services in Edge Locations, which are physically closer to end users than the main AWS Regions, significantly reduces latency. For example, CloudFront can serve cached content directly from the nearest Edge Location.
 - ❑ There are over 200 Edge Locations across 100+ cities in 50+ countries, with more being added regularly. Having many Edge Locations means users are more likely to be close to one.
 - ❑ Edge Locations are not the same as Local Zones, which allow running compute, storage and database workloads with single-digit millisecond latency in major metro areas.
 - ❑ You can't directly run your own workloads in Edge Locations - they are used only by managed AWS services.

AWS Local Zone



AWS Local Zones

- AWS Local Zones are an infrastructure deployment option that brings AWS services closer to end-users, providing single-digit millisecond latencies.
- **Key points about AWS Local Zones:**
 - They are an extension of existing AWS Regions and are located in large population centers.
 - Local Zones allow running EC2 instances, databases, and other services in close proximity to users for latency-sensitive applications like real-time gaming, live streaming, AR/VR, and virtual workstations.
 - To use a Local Zone, you first enable it, then create a subnet in the Local Zone within your VPC, and finally launch resources like EC2 instances in that subnet.
 - Local Zones have their own connections to the internet and support AWS Direct Connect for hybrid connectivity.

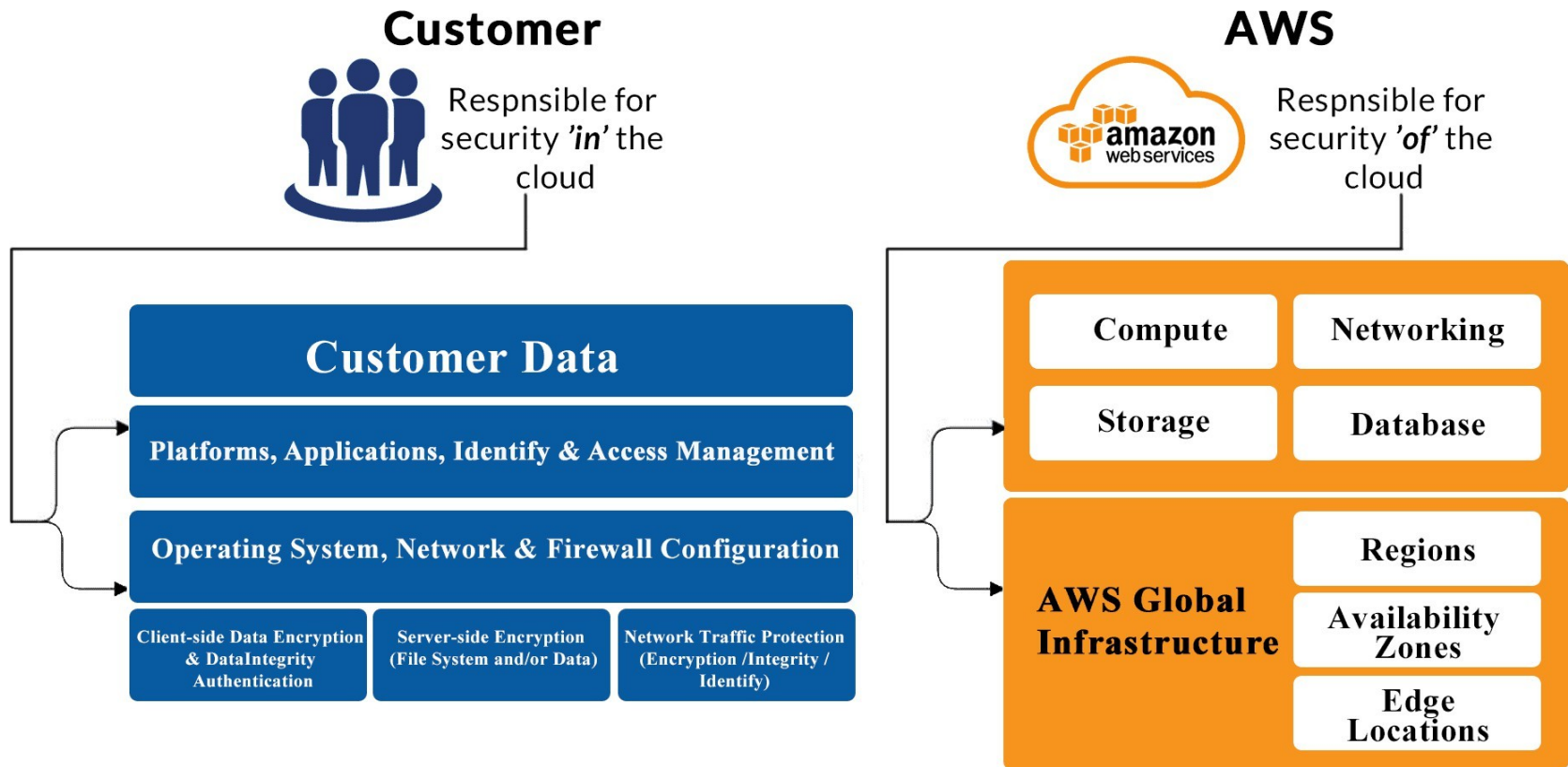
Feature	Local Zones	Availability Zones	Edge Locations
Purpose	Bring compute, storage, and other services closer to end-users for low-latency applications	Provide highly available and fault-tolerant infrastructure for running applications	Serve as a content delivery network (CDN) to cache content and serve requests with low latency
Services	Offer select services like EC2, EBS, VPC, etc. locally. Other services accessible via private network	Provide the full array of AWS services	Focus on services like CloudFront, Route 53, Lambda@Edge for CDN
Locations	Extensions of AWS Regions located in large population centers	Multiple isolated locations within a Region, physically separated for redundancy	Hundreds of locations worldwide, not part of Regions
Latency	Single-digit millisecond latency to end-users	Low latency within the Region	Very low latency for CDN requests
Pricing	Slightly higher than equivalent services in the parent Region	Standard pricing for the Region	Standard pricing for CDN services
Examples	Local Zones in Los Angeles, Seattle, Houston, etc.	Multiple AZs in us-east-1, us-west-2, etc.	CloudFront Edge Locations worldwide

AWS Shared Responsibility Model

AWS Shared Responsibility Model

- Security and Compliance is a shared responsibility between AWS and the customer.
- This shared model can help relieve the customer's operational burden as AWS operates, manages and controls the components from the *host operating system and virtualization layer* down to the physical security of the facilities in which the service operates.
- The customer assumes responsibility and management of the guest operating system (including updates and security patches), other associated application software as well as the configuration of the AWS provided security group firewall.
- Customers should carefully consider the services they choose as their responsibilities vary depending on the services used, the integration of those services into their IT environment, and applicable laws and regulations.
- The nature of this shared responsibility also provides the flexibility and customer control that permits the deployment

AWS Shared Responsibility Model



CUSTOMER

RESPONSIBILITY FOR
SECURITY 'IN' THE CLOUD

CUSTOMER DATA

PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT

OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION

CLIENT-SIDE DATA
ENCRYPTION & DATA INTEGRITY
AUTHENTICATION

SERVER-SIDE ENCRYPTION
(FILE SYSTEM AND/OR DATA)

NETWORKING TRAFFIC
PROTECTION (ENCRYPTION,
INTEGRITY, IDENTITY)

AWS

RESPONSIBILITY FOR
SECURITY 'OF' THE CLOUD

SOFTWARE

COMPUTE

STORAGE

DATABASE

NETWORKING

HARDWARE/AWS GLOBAL INFRASTRUCTURE

REGIONS

AVAILABILITY ZONES

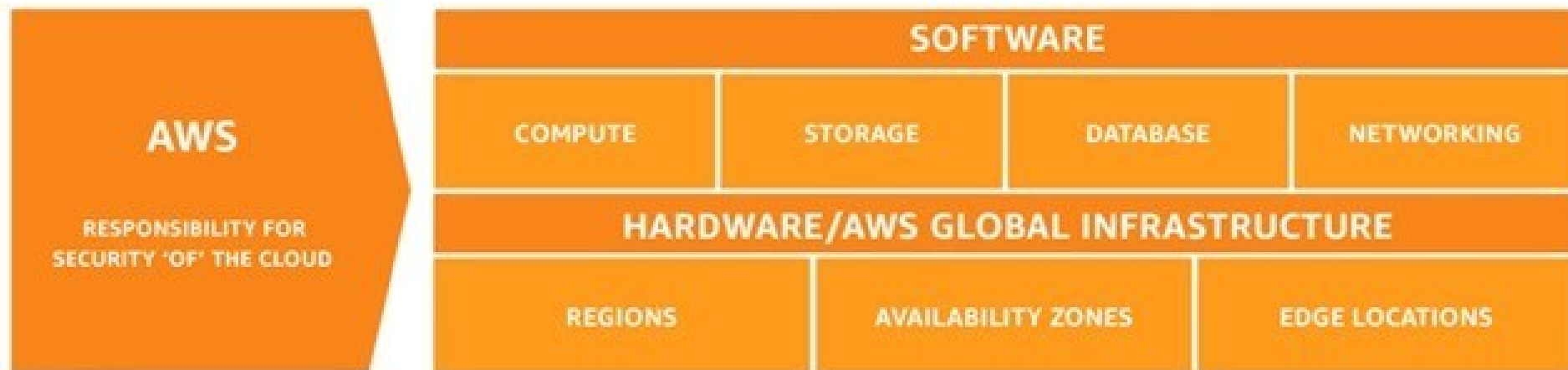
EDGE LOCATIONS

AWS Shared Responsibility Model

Your Responsibility	On-Premises Deployments	IaaS	PaaS	SaaS
	Application Code	Application Code	Application Code	Application Code
	Security	Security	Security	Security
	Database	Database	Database	Database
	OS	OS	OS	OS
	Virtualization	Virtualization	Virtualization	Virtualization
	Networking	Networking	Networking	Networking
	Storage Hardware	Storage Hardware	Storage Hardware	Storage Hardware
	Server Hardware	Server Hardware	Server Hardware	Server Hardware
Cloud Platform Responsibility				

AWS responsibility “Security of the Cloud”

- AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud.
- This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.



Customer responsibility “Security in the Cloud”

- Customer responsibility will be determined by the AWS Cloud services that a customer selects.
- For example, a service such as Amazon Elastic Compute Cloud (Amazon EC2) is categorized as Infrastructure as a Service (IaaS) and, as such, requires the customer to perform all of the necessary security configuration and management tasks.
- Customers that deploy an Amazon EC2 instance are responsible for management of the guest operating system (including updates and security patches), any application software or utilities installed by the customer on the instances, and the configuration of the AWS-provided firewall (called a security group) on each instance.
- ***Customers are responsible for managing their data (including encryption options), classifying their assets, and using IAM tools to apply the appropriate permissions.***

AWS Shared Responsibility Model

- Inherited Controls – Controls which a customer fully inherits from AWS.
- Physical and Environmental controls
- Shared Controls – Controls which apply to both the infrastructure layer and customer layers, but in completely separate contexts or perspectives. In a shared control, AWS provides the requirements for the infrastructure and the customer must provide their own control implementation within their use of AWS services. Examples include:
 - **Patch Management** – AWS is responsible for patching and fixing flaws within the infrastructure, but customers are responsible for patching their guest OS and applications.
 - **Configuration Management** – AWS maintains the configuration of its infrastructure devices, but a customer is responsible for configuring their own guest operating systems, databases, and applications.

Applying the AWS Shared Responsibility Model in Practice

- Determine external and internal security and related compliance requirements and objectives, and consider industry frameworks like the NIST Cybersecurity Framework (CSF) and ISO.
- Consider employing the AWS Cloud Adoption Framework (CAF) and Well-Architected best practices to plan and execute your digital transformation at scale.
- Review the security functionality and configuration options of individual AWS services within the security chapters of AWS service documentation.
- Review third-party audit attestation documents to determine inherited controls

Identity and Access Management (IAM)

Identity and Access Management (IAM)

- **IAM** manages Amazon Web Services (AWS) users and their access to AWS accounts and services.
- It controls the level of access a user can have over an AWS account & set users, grant permission, and allows a user to use different features of an AWS account.
- Identity and access management is mainly used to manage users, groups, roles, and Access policies
- The account we created to sign in to Amazon web services is known as the **root account** and it holds all the administrative rights and has access to all parts of the account.

What Does IAM Do?

- IAM Identities
 - IAM Identities assists us in controlling which users can access which services and resources in the AWS Console and also we can assign policies to the users, groups, and roles.
 - The IAM Identities can be created by using the Root user.
 - IAM Identities Classified As
 - IAM Users
 - IAM Groups
 - IAM Roles
-

Identity and Access Management (IAM)



IAM user

A **person** *or* **application** that can authenticate with an AWS account.



IAM group

A **collection of IAM users** that are granted identical authorization.



IAM policy

The document that defines **which resources can be accessed** and the **level of access** to each resource.



IAM role

Useful mechanism to grant a set of permissions for making AWS service requests.

Identity and Access Management

- (IAM) Define fine-grained access rights
 - ❑ **Who** can access the resource
 - ❑ **Which** resources can be accessed and what can the user do to the resource
 - ❑ **How** resources can be accessed

Root user - What Does IAM Do?

- The root user will automatically be created and granted unrestricted rights. We can create an admin user with fewer powers to control the entire Amazon account.
- **IAM Users**
 - We can utilize IAM users to access the AWS Console and their administrative permissions differ from those of the Root user and if we can keep track of their login information.
- **Example**
 - With the aid of IAM users, we can accomplish our goal of giving a specific person access to every service available in the Amazon dashboard with only a limited set of permissions, such as read-only access. Let's say user-1 is a user that I want to have read-only access to the EC2 instance and no additional permissions, such as create, delete, or update.

IAM Groups

- A group is a collection of users, and a single person can be a member of several groups.
- With the aid of groups, we can manage permissions for many users quickly and efficiently.
- **Example**
 - Consider two users named user-1 and user-2.
 - If we want to grant user-1 specific permissions, such as the ability to delete, create, and update the auto-calling group only, and if we want to grant user-2 all the necessary permissions to maintain the auto-scaling group as well as the ability to maintain EC2, we can create groups and add this user to them.
 - If a new user is added, we can add that user to the required group with the necessary permissions.

IAM Roles

- While policies cannot be directly given to any of the services accessible through the Amazon dashboard, IAM roles are similar to IAM users in that they may be assumed by anybody who requires them.
- By using roles, we can provide AWS Services access rights to other AWS Services.
- Example
 - ❑ Consider Amazon EKS. In order to maintain an autoscaling group, AWS EKS needs access to EC2 instances.
 - ❑ Since we can't attach policies directly to the EKS in this situation, we must build a role and then attach the necessary policies to that specific role and attach that particular role to EKS.

IAM Policies

- IAM Policies can manage access for AWS by attaching them to the IAM Identities or resources.
- IAM policies defines permissions of AWS identities and AWS resources when a user or any resource makes a request to AWS will validate these policies and confirms whether the request to be allowed or to be denied.
- AWS policies are stored in the form of JSON format the number of policies to be attached to particular IAM identities depends upon number of permissions required for one IAM identity.
- IAM identity can have multiple policies attached to them.

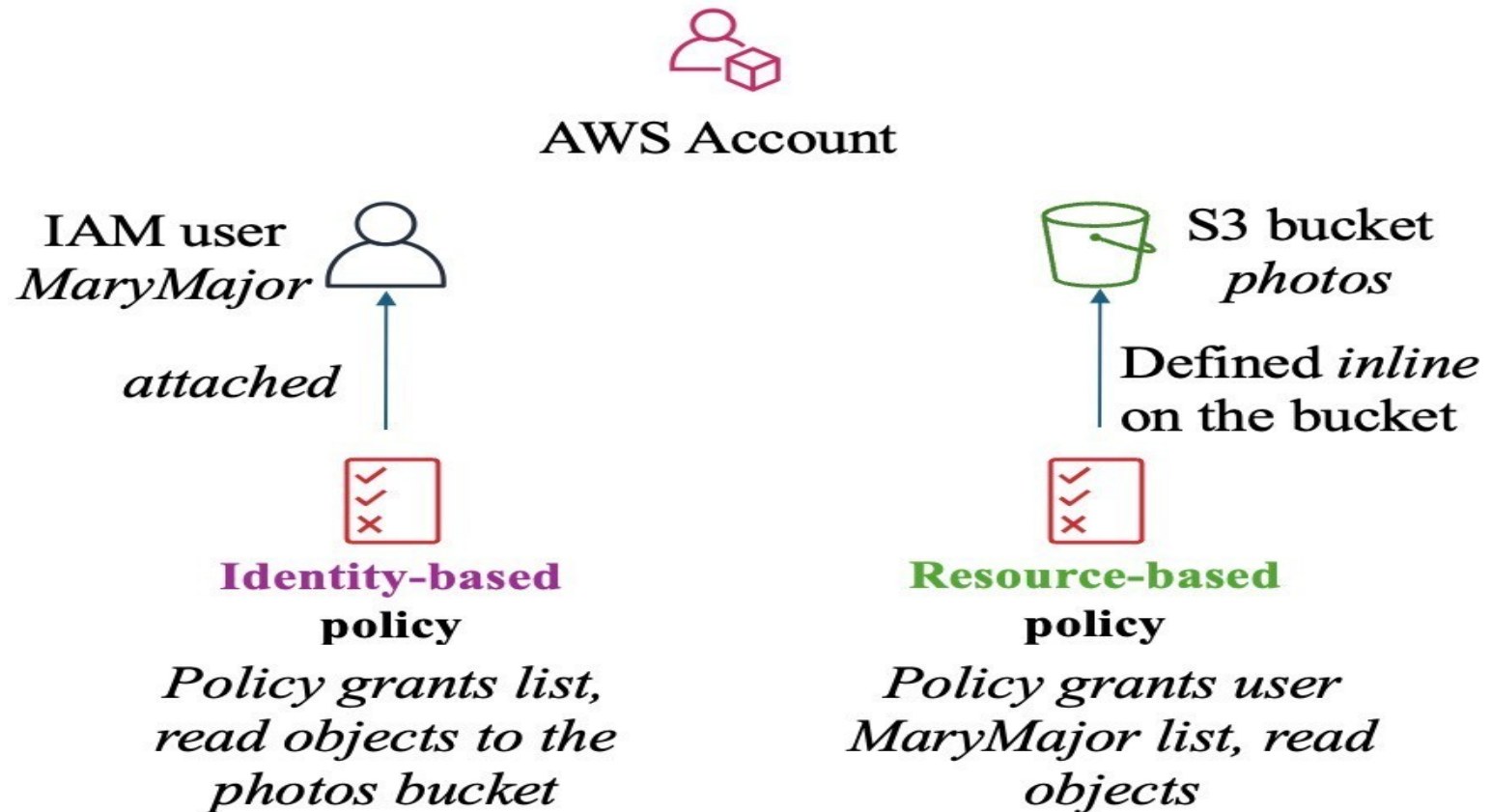
IAM Policies

- Two types of policies – *Identity-Based* and *Resource-Based*
- **Identity-based** policies –
 - Attach a policy to any IAM entity
 - An IAM user, an IAM group, or an IAM role
 - Policies specify:
 - Actions that *may* be performed by the entity
 - Actions that *may not* be performed by the entity
 - A single *policy* can be attached to multiple *entities*
 - A single *entity* can have multiple *policies* attached to it
- **Resource-based** policies
 - Attached to a resource (such as an S3 bucket)

IAM Policies

- *Identity-based policies* are attached to a user, group, or role
 - **Resource-based policies** are attached to a resource (*not* to a user, group or role)
 - Characteristics of resource-based policies –
 - Specifies who has access to the resource and what actions they can perform on it
 - The policies are *inline* only, not managed
 - Resource-based policies are supported only by some AWS services
-

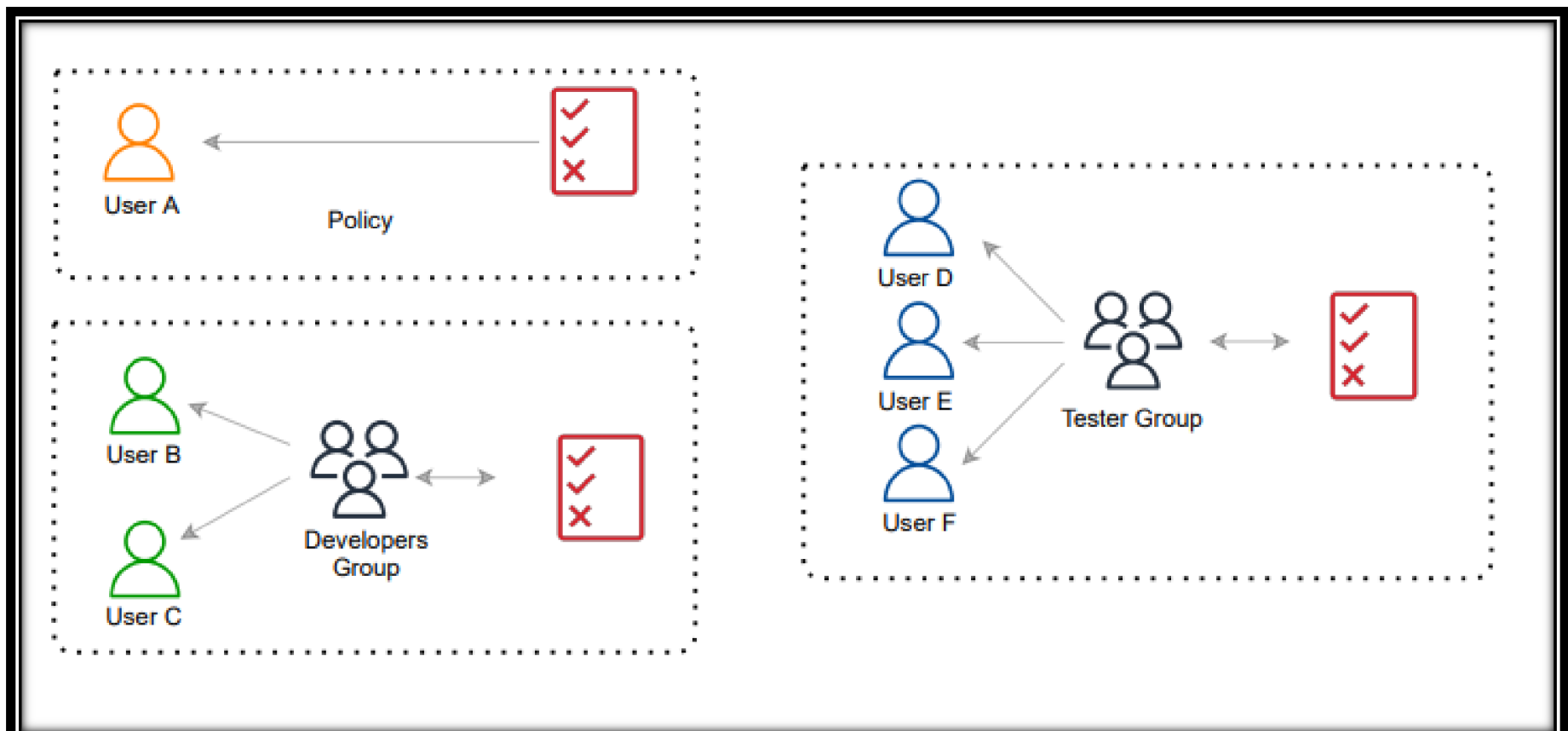
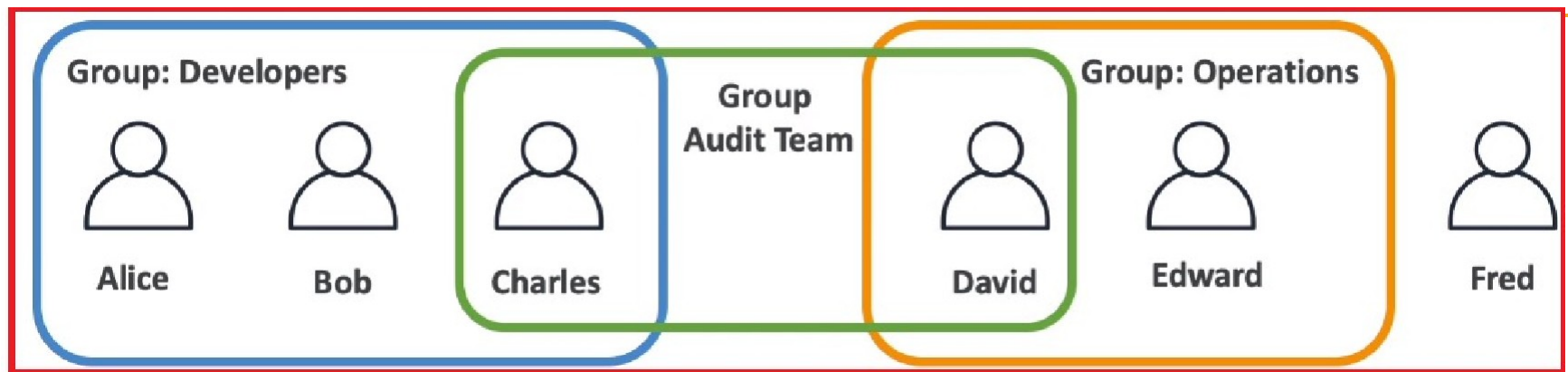
IAM Policies



How IAM Works?

- IAM verifies that a user or service has the necessary authorization to access a particular service in the AWS cloud.
- We can also use IAM to grant the right level of access to specific users, groups, or services.
- For example, we can use IAM to enable an EC2 instance to access S3 buckets by requesting fine-grained permissions.





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 Structure

- An IAM (Identity and Access Management) policy in AWS is a JSON document that defines permissions for actions on AWS resources.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:StartInstances",
      "Resource": "arn:aws:ec2:region:account-id:instance/*"
    }
  ]
}
```

IAM Policies Structure

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

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["dynamodb:*", "s3:*"],
    "Resource": [
      "arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
      "arn:aws:s3:::bucket-name",
      "arn:aws:s3:::bucket-name/*"]
  },
  {
    "Effect": "Deny",
    "Action": ["dynamodb:*", "s3:*"],
    "NotResource": ["arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
      "arn:aws:s3:::bucket-name",
      "arn:aws:s3:::bucket-name/*"]
  }
]
}
```

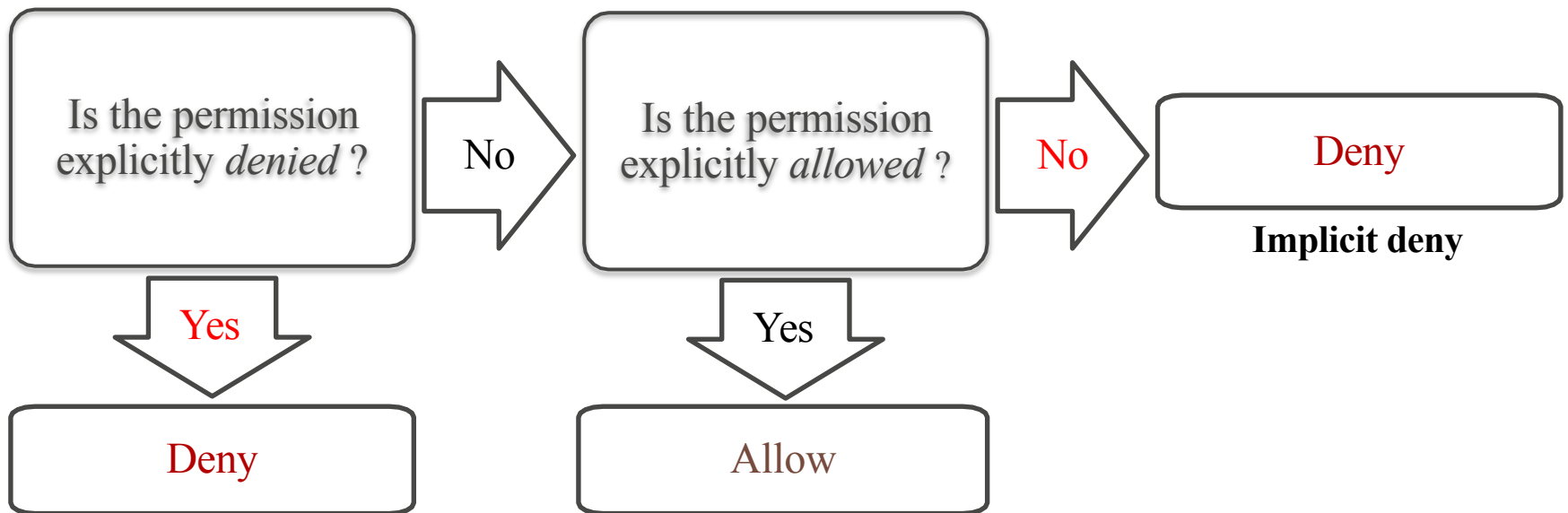
Explicit allow gives users access to a specific DynamoDB table and...

...Amazon S3 buckets.

Explicit deny ensures that the users cannot use any other AWS actions or resources other than that table and those buckets.

An explicit deny statement **takes precedence** over an allow statement.

IAM Permissions



IAM Features

- **Free of cost:** IAM feature of the Aws account is free to use & charges are added only when you access other Amazon web services using IAM users.
- **Have Centralized control over your Aws account:** Any new creation of users, groups, or any form of cancellation that takes place in the Aws account is controlled by you, and you have control over what & how data can be accessed by the user.
- **Grant permission to the user:** As the root account holds administrative rights, the user will be granted permission to access certain services by IAM.
- **Multifactor Authentication:** Additional layer of security is implemented on your account by a third party, a six-digit number that you have to put along with your password when you log into your accounts.



AWS Identity and Access Management

Apply fine-grained permissions to AWS services and resources



Who

Workforce users and workloads with IAM



Can access


Permissions with IAM policies



What

Resources within your AWS organization

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 -



- Users have access to your account and can possibly change configurations or delete resources in your AWS account
- You want to protect your Root Accounts and IAM users
- MFA = password you know + security device you own
- Main benefit of MFA:
 - if a password is stolen or hacked, the account is not compromised

How can users access AWS ?

- To access AWS, you have **THREE** options:
 - **AWS Management Console** (protected by password + MFA)
 - **AWS Command Line Interface (CLI)**: protected by access keys
 - **AWS Software Developer Kit (SDK)** - for code: protected by access keys
- Access Keys are generated through the AWS Console
- Users manage their own access keys
- Access Keys are secret, just like a password.
 - ✓ Access Key ID \sim username
 - ✓ Secret Access Key \sim password

IAM Guidelines & Best Practices

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

IAM Section – Summary

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



Design an IAM structure for VIT Resources



Separate access levels for faculty, students, administrative staff, and IT Admins.



Granular permissions to ensure users can only access resources necessary for their roles.



Secure management of temporary access for visiting faculty or short-term staff.



Implementation of the principle of least privilege across all user types.