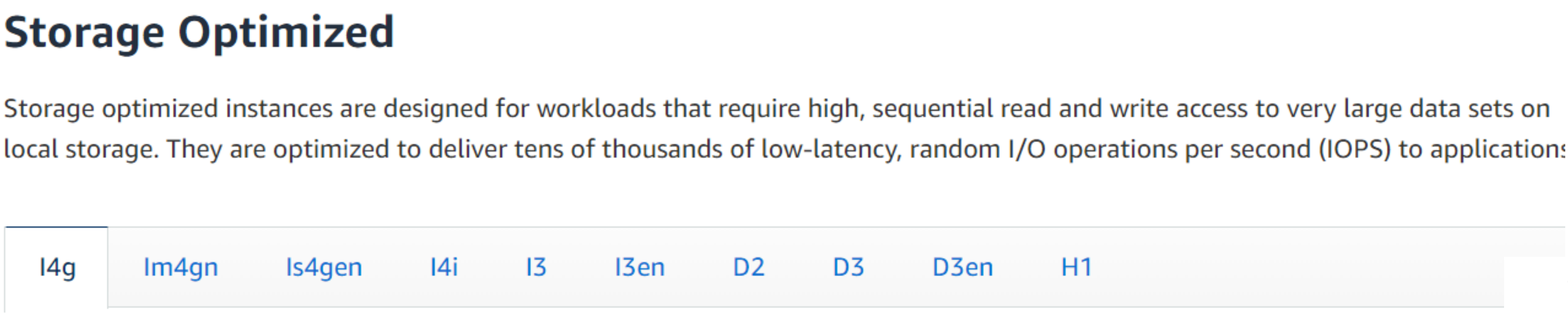
Cloud Computing Course Outline

Farid Afzali, Ph.D., P.Eng.

EC2 Instance Types- Storage Optimized

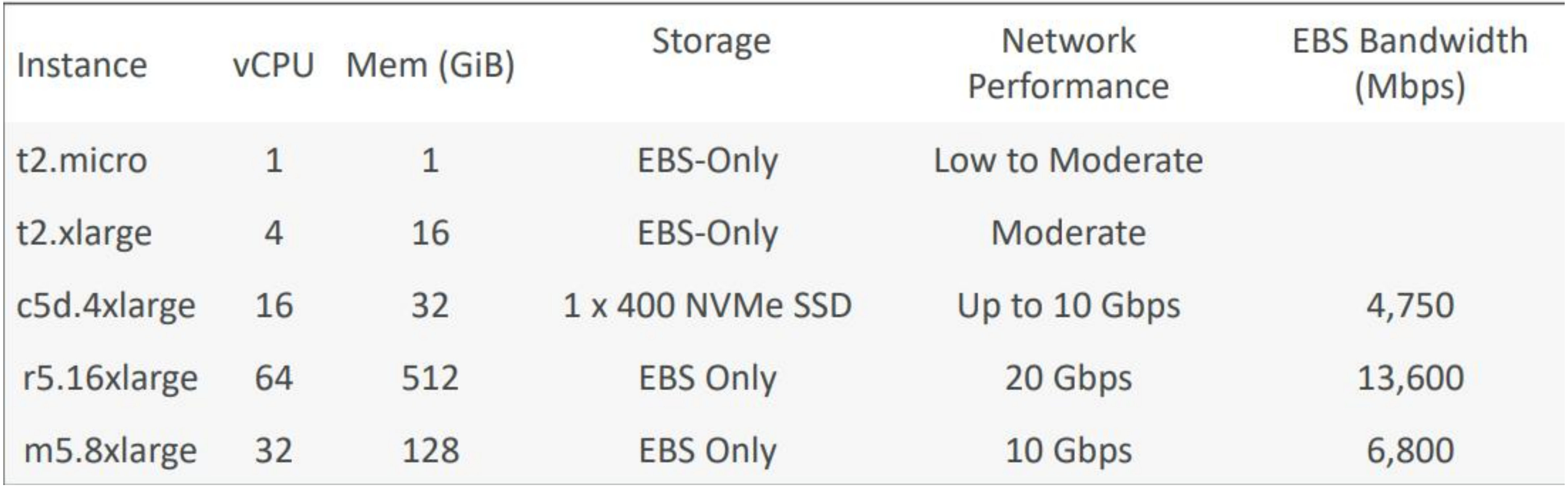
- **Storage Optimized Instances**: These instances are ideal for storage-intensive tasks that need high levels of sequential read and write access to large data sets on local storage.
- **Use Cases**:
  - **High Frequency Online Transaction Processing (OLTP) Systems**: These systems require rapid and reliable processing of transactions, and storage optimized instances can provide the necessary I/O performance.
  - **Relational & NoSQL Databases**: Databases that handle large volumes of data transactions benefit from the storage capabilities of these instances.
  - **Cache for In-Memory Databases**: For example, caching mechanisms like Redis can leverage the fast storage for quick data retrieval.
  - **Data Warehousing Applications**: These are used for analyzing and querying large datasets, and the storage optimized instances offer the required performance.
  - **Distributed File Systems**: They rely on multiple machines for data storage and access, and the high I/O performance of storage optimized instances can significantly benefit such setups.
- **Instance Types**: The slide shows different instance types within the Storage Optimized category, including I3, I3en, D2, D3, D3en, and H1. These vary in their specific configurations and are tailored to support different types of storage workloads.
- Link to EC2 Instances Storage-optimized



Storage Optimized

Storage optimized instances are designed for workloads that require high, sequential read and write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications

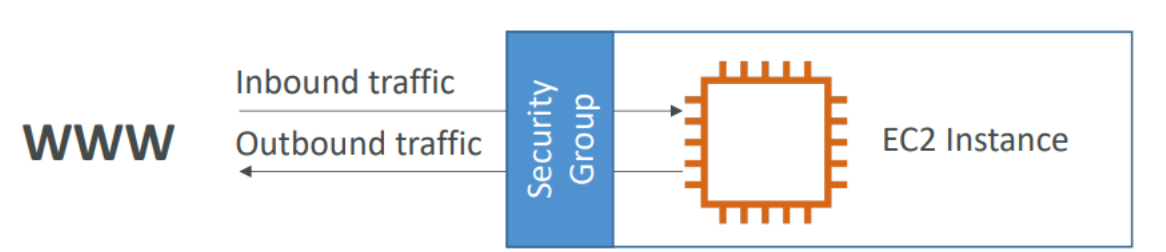
EC2 Instance Types- Examples



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

Amazon EC2 Instance Comparison (vantage.sh)

(AWS) EC2 security groups



- **Security Groups**: These are a fundamental aspect of network security within the AWS ecosystem, acting as a virtual firewall for EC2 instances to control inbound and outbound traffic.
- **Traffic Control**: Security groups regulate what traffic is permitted to enter or leave an EC2 instance.
- **Allow Rules**: Security groups contain rules that specifically allow traffic. Unlike traditional firewalls that can have both allow and deny rules, security groups only have allow rules.
- **Rule Reference**: The rules within a security group can specify traffic based on IP address or by other security groups, thus enabling granular control over the network access to the instances.
- Security groups are used to control access to EC2 instances, ensuring that only the traffic defined by the rules set by the administrator is allowed. This is crucial for maintaining the security posture and limiting potential vulnerabilities of the cloud infrastructure.

(AWS) EC2 security groups

(AWS) EC2 security groups

- **Nightclub (EC2 Instance)**: This is your server in the cloud where your application is running. Just like a nightclub has music and a dance floor for guests, the EC2 instance has computational resources and applications for users.
- **Bouncer (Security Group)**: The bouncer checks each guest against a list (security group rules) to decide who gets in or out. Similarly, the security group checks data packets against its rules to decide which can enter or leave the EC2 instance.
- **Guest List (Inbound Rules)**: The bouncer has a list that allows certain people entry. If your name is on the list (if the data packets are from an allowed IP address or port), you're allowed to enter the nightclub. In the case of the EC2 instance, if the incoming traffic is from allowed IP addresses or ports, it gets through to the server.
- **Exit Stamp (Outbound Rules)**: When guests leave, they get a stamp that allows them to re-enter. Outbound rules work similarly, where once the traffic has been allowed out, it's generally allowed back in.
- **Exclusive Party (Allow Rules Only)**: Our nightclub is exclusive, only people on the guest list can enter, there's no "ban list." Similarly, security groups are stateful and only have allow rules, not deny rules; if you're not explicitly allowed, you're implicitly denied.

Security Groups Deeper Dive

- **Security Groups as Firewalls**: Security groups serve as a type of firewall for EC2 instances, controlling inbound and outbound traffic to the instances.
- **Regulations Implemented by Security Groups**:
  - **Access to Ports**: They define which ports are open for communication, such as port 80 for HTTP or port 22 for SSH.
  - **Authorized IP Ranges**: They specify which IP ranges (IPv4 and IPv6) are allowed to communicate with the instance. For instance, allowing all IP addresses (0.0.0.0/0) or just a specific IP address.
  - **Control of Inbound Network**: Rules dictate what incoming network traffic is allowed to reach the instance.
  - **Control of Outbound Network**: Rules define what outgoing network traffic is permitted from the instance.

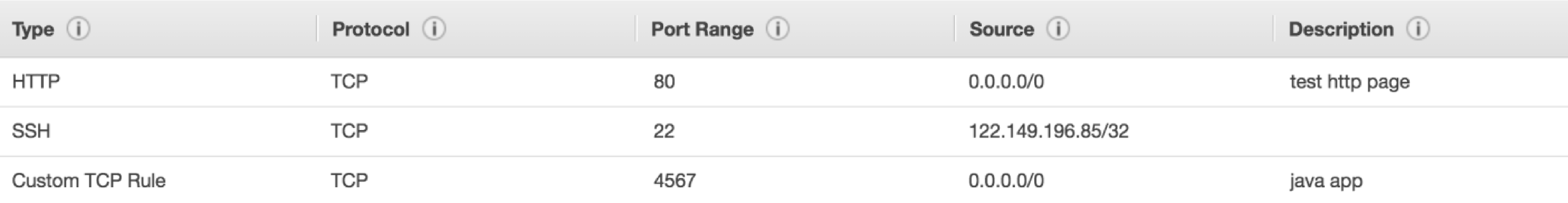
PRINTABLE PDF | A4, A5, LETTER & HALF LETTER

GUEST LIST

Guest List

NAME	CONTACT	RSVP		
		YES	MAYBE	NO

Security Groups Deeper Dive



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

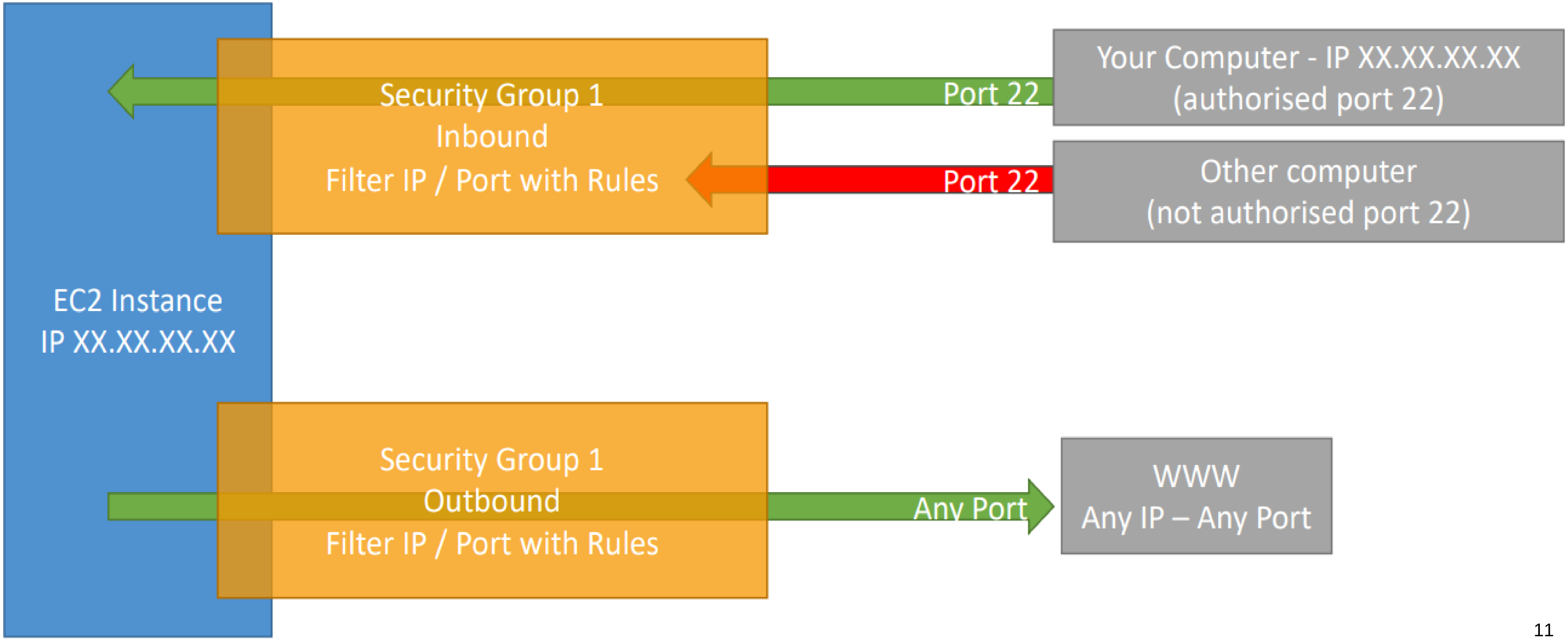
Security Groups Deeper Dive

- The image shows a part of the AWS Management Console or a similar interface, where Security Group rules are listed, displaying the type of traffic allowed (like HTTP or SSH), the protocol used (TCP), the port range (like 80 for HTTP), the source IP range for the traffic, and a description for each rule.
- **Example Rules (based on typical AWS Security Group configuration) shown in the table:**
- **HTTP Rule**: Allows web traffic on port 80 from any IP address (0.0.0.0/0), typically for accessing web pages hosted on the EC2 instance.
- **SSH Rule**: Allows SSH access on port 22 from a specific IP address (122.149.196.85/32), typically for secure administrative access to the EC2 instance.
- **Custom TCP Rule**: A user-defined rule for a custom application that communicates over TCP on port 4567.

Security Groups Deeper Dive Example

- **Security Groups**: Imagine the security group as the security desk at the entrance of an office building. The security personnel at the desk decide who can enter and leave the building.
- **Access to Ports**: Each port can be thought of as a door to the building. Some doors are for employees only (SSH - port 22), while others are open to the public (HTTP - port 80).
- **Authorized IP Ranges**: This is similar to a guest list or employee ID check. Only people whose names are on the list or who have the right ID badge (specific IP addresses) are allowed in.
- **Control of Inbound Network**: This equates to controlling who can enter the building (inbound traffic). If someone is not recognized or authorized, they are not allowed inside.
- **Control of Outbound Network**: This is like controlling who can leave the building (outbound traffic). It ensures that only those with proper clearance can take information out of the building.
- For example, the HTTP rule is like having the front door open during business hours for any visitor (0.0.0.0/0) to enter and ask for information or conduct business. The SSH rule is like a secure back entrance where only employees with a key card (specific IP address) can enter, ensuring secure and restricted access to the building's operations. The custom TCP rule is like a special delivery door that only accepts deliveries (traffic) from a pre-approved list of delivery services (specified IPs) at certain times.

Security Groups Diagram



Security Groups Diagram

- **EC2 Instance**: This represents a virtual server in AWS with its own IP address (labeled as IP XX.XX.XX.XX in the diagram).
- **Security Group 1 - Inbound**: This is a set of rules that filter incoming traffic to the EC2 instance. It specifies that only traffic from a certain IP address (Your Computer - IP XX.XX.XX.XX) is allowed on port 22, which is commonly used for SSH (Secure Shell) connections. The green arrow indicates allowed traffic.
- **Security Group 1 - Outbound**: This shows the rules for outgoing traffic from the EC2 instance. It is configured to allow all outbound traffic to any IP and port, which is depicted by the green line leading out to the WWW (the internet).
- **Traffic from Your Computer**: The grey box represents a computer with an IP address that is authorized to access the EC2 instance on port 22. The green line indicates successful traffic allowed by the security group.
- **Traffic from Other Computer**: This grey box represents another computer that is not authorized to access the EC2 instance on port 22. The red line suggests that traffic from this computer is being blocked by the security group.
- Security group is shown as selectively allowing SSH access to the EC2 instance from a specific IP address, while permitting all outbound traffic from the instance to the internet. It effectively demonstrates how security groups are used to manage access to AWS resources, providing a clear visual interpretation of network security rules applied at the instance level.

Security Groups Diagram

HANDS ON PRACTICE

Classic network ports and their associated services

- **Port 22**: Used for SSH (Secure Shell) which provides a secure channel over an unsecured network in a client-server architecture, allowing users to log into a Linux instance.
- **Port 21**: Assigned to FTP (File Transfer Protocol), which is used for the transfer of computer files between a client and server on a computer network.
- **Port 22 (again)**: This seems to be a duplicate entry, as SSH typically uses port 22. However, it's worth noting that SFTP (Secure File Transfer Protocol) also uses port 22 and provides a secure method of transferring files using, and tunneled over, SSH.
- **Port 80**: The standard port for HTTP (Hypertext Transfer Protocol), which is used for unsecured communications and is the foundation of data communication for the World Wide Web.
- **Port 443**: The standard port for HTTPS (HTTP Secure), which is used for secure communications over a computer network and is widely used on the Internet.
- **Port 3389**: Used for RDP (Remote Desktop Protocol), which allows a user to connect to a Windows computer remotely.

Classic network ports and their associated services- Analogous Example

- Imagine a large apartment building, where each apartment represents a service, and the door number corresponds to a service's port number. Each resident in the apartment building has a key to access their own apartment.
- **Port 22 for SSH**: It's like having a secure, modern keycard system for an apartment that hosts sensitive information. Only residents with a verified keycard (SSH key) can swipe in, similar to how only authorized users with the correct SSH key can access a Linux instance.
- **Port 21 for FTP**: Think of this as a mail slot on the door of an apartment designated for file deliveries. Anyone with access to the building can drop files through the slot (FTP server) into the apartment (server storage).
- **Port 80 for HTTP**: This is akin to a public bulletin board in the lobby of the building. Anyone can read the information posted (access unsecured websites), but they can't post anything themselves without access to the board's locked box (server backend).
- **Port 443 for HTTPS**: Imagine a secured bulletin board where the box is not only locked but also has a security guard (SSL/TLS encryption) checking ID badges before allowing anyone to view or post information, ensuring all interactions are secure.
- **Port 3389 for RDP**: Consider this as a remote control system for an apartment, where a resident can give a special remote to a friend. This remote lets the friend enter the apartment and use it as if they were physically there, much like RDP allows remote control over a Windows instance.

How to SSH into an EC2 instance

- To SSH into an EC2 instance from a Windows machine, you typically use an SSH client. PuTTY is one of the most popular SSH clients for Windows. Here's a simple step-by-step guide:

1. Download and install PuTTY from the official site.
2. Obtain your EC2 instance's public IP address from the AWS Management Console.
3. Open PuTTY and in the "Host Name (or IP address)" field, enter the public IP address of your EC2 instance.
4. Under "Connection type", ensure SSH is selected.
5. Before opening the connection, navigate to "Connection" > "SSH" > "Auth" in the PuTTY sidebar.
6. Here, you will need to provide the path to your private key file. If your key is not in PuTTY's PPK format, you will need to convert it using PuTTYgen, which comes with the PuTTY suite.
7. After selecting your private key, go back to the "Session" page and click "Open" to initiate the SSH session.
8. If this is the first time you are connecting to this instance, you will get a security alert about the host key not being cached in the registry. You can proceed by clicking "Yes".
9. A terminal window will open, and you will be prompted to log in. For an Amazon Linux AMI, the default user is "ec2-user", while for an Ubuntu AMI, it is typically "ubuntu".

How to SSH into an EC2 instance- Example

iStock
Credit: Theerawan Bangpran
958731770

- Imagine you work in a large corporate office that's located in a busy downtown district. Your office building requires a special access card to enter, similar to how an EC2 instance requires a private key for secure access.
- **PuTTY**: Think of PuTTY as the specialized card reader at the entrance of the building. Just like PuTTY is the interface that allows you to communicate with your EC2 instance, the card reader is the interface that reads your access card to verify if you have permission to enter the building.
- **SSH**: This is like the communication protocol between your access card and the card reader. It's a secure method that ensures only authorized personnel (or in the case of servers, users) can gain entry.
- **EC2 Instance**: The office you're trying to enter is like the EC2 instance. It's the private space where your work resources are located, and it's secured against unauthorized entry.
- **Public IP Address**: This is like the specific floor and office number where you work. Without it, the card reader wouldn't know if you're trying to access the correct location.
- **Private Key**: Your personal access card, which has a specific code that the card reader recognizes, is like the private key file used in SSH. It's unique to you and ensures that only you can gain entry to your office.
- So, when you use PuTTY (the card reader) with your private key (the access card), you can securely SSH (the communication protocol) into your EC2 instance (the office) using its public IP address (the office location).

How to SSH into an EC2 instance

HANDS ON PRACTICE

EC2 instance Purchasing options

- **On-Demand Instances**: These are like renting a car as needed. You pay by the minute or hour, which is great for short, unpredictable trips where you don't want to commit to a long-term arrangement.
- **Reserved Instances**: This is like leasing a car. You commit to a 1 or 3-year term for a lower price, suitable for known long-term needs.
- **Convertible Reserved Instances**: It's similar to a flexible lease where you can switch cars as your needs change, still benefiting from the lower leased rate.
- **Savings Plans**: Think of it as a subscription plan for car usage. You commit to using a certain amount of car hours over 1 or 3 years and get a discount for making this commitment, even if your exact car needs may vary.
- **Spot Instances**: These are akin to bidding for a last-minute seat on a flight. It's less expensive, but there's a chance you might get bumped off if someone else pays more.
- **Dedicated Hosts**: This is like having a personal driver with a car reserved just for you. You have control over the car and it's not shared with anyone else.
- **Dedicated Instances**: Similar to a dedicated host but within a shared garage. Your car is yours alone, even though it's parked with others.
- **Capacity Reservations**: It's like reserving a parking space for your car in a specific location, ensuring it's always available to you whenever you need it, for any duration.

EC2 instance Purchasing options- Examples

- **On-Demand Instances**: This is like renting a hotel room with no prior reservation. You pay for the nights you stay, offering maximum flexibility without a long-term commitment.
- **Reserved Instances**: This is akin to renting an apartment with a lease. You sign a contract to rent the apartment (instance) for a fixed period, usually 1 or 3 years, often with a discount compared to the hotel (on-demand) rate.
- **Convertible Reserved Instances**: This is similar to a lease with an option to change apartments within the same building if your needs change, still under a contract that offers a lower rate than short-term rentals.
- **Savings Plans**: It's like a gym membership. You commit to a certain level of usage (1 or 3 years), guaranteeing you access and certain rates, but you can choose different equipment (instances) within the gym as your workout routine (workload) changes.
- **Spot Instances**: This is like bidding for a last-minute vacation rental. If your bid is high enough and the rental isn't already booked, you get it for a lower price, but you might have to leave if a higher-paying renter comes along.
- **Dedicated Hosts**: This is like owning a home. It's exclusively yours, providing full control and privacy, with no one else having access to your space.
- **Dedicated Instances**: Similar to having a private room in a shared house. No one else can enter your room, but the common areas like the kitchen or living room are shared.
- **Capacity Reservations**: This is like reserving a room or a suite in a resort for future use. You don't necessarily use it right away, but it's guaranteed to be available whenever you decide to go.

EC2 On-Demand instances

- **Pay for what you use**: This billing model means that charges are based on the actual usage of resources. For Linux or Windows operating systems, billing is calculated per second after the first minute of usage. For all other operating systems, the billing is done per hour.
- **Highest cost but no upfront payment**: On-Demand instances usually cost more than Reserved or Spot instances because they do not require any long-term commitment or upfront payment. Users are free to start and stop these instances as they wish, and they will only be charged for the time the instances were running.
- **No long-term commitment**: Users are not locked into any contractual or time-bound agreement when using On-Demand instances. They have the flexibility to use the compute capacity with no long-term commitments.
- **Recommended for short-term and uninterrupted workloads**: On-Demand instances are ideal for short-term, sporadic, or unpredictable workloads that cannot be interrupted. They are suitable for applications where it is challenging to predict the behavior, and there is a need for reliable compute capacity without any disruption.
- EC2 On-Demand instances are a flexible and straightforward option for computing resources, allowing users to manage their workloads without the need for planning or forecasting usage. They are best suited for tasks that are either temporary, experimental, or subject to unexpected changes in usage.

EC2 Reserved Instances

- **Up to 72% discount compared to On-Demand**: Reserved Instances offer a significant discount compared to the price of On-Demand instances, in return for committing to a specified usage over a period of time.
- **Specific instance attributes reservation**: When reserving an instance, you can specify attributes like the instance type, region, tenancy, and operating system.
- **Reservation Period**: There are options for 1 or 3-year terms, with the longer period typically offering a greater discount.
- **Payment Options**: There are different payment options, including no upfront cost, partial upfront cost, or paying all upfront. The more you pay upfront, the greater the discount on the hourly rate.
- **Reserved Instance’s Scope**: The reservation can be Regional, providing more flexibility in where you run your instances, or Zonal, which reserves capacity in a specific Availability Zone.
- **Recommended for steady-state usage applications**: They are ideal for workloads with predictable usage that won't change much over time, such as databases.
- **Reserved Instance Marketplace**: AWS provides a marketplace where you can sell your unused Reserved Instances to other AWS customers.
- **Convertible Reserved Instance**: These allow you to change the EC2 instance type, family, OS, scope, and tenancy during the term of the reservation, offering up to a 66% discount.

EC2 Savings Plans

- **EC2 Savings Plans**: These plans provide a discount on AWS usage by committing to a consistent amount of usage (measured in $/hour) for a 1 or 3-year period.
- **Discounts**: Users can get up to a 72% discount compared to standard On-Demand rates, matching the discount level of Reserved Instances.
- **Commitment**: The commitment is based on a specified amount of usage rather than specific instance configurations.
- **Billing**: Any usage beyond the commitment covered by the Savings Plans is billed at the regular On-Demand rates.
- **Flexibility**: Savings Plans are flexible and apply to any usage across different instance sizes within the chosen instance family and AWS region. They also allow for flexibility across operating systems and tenancy options.
- **Instance Family & AWS Region**: Savings Plans are typically locked to a specific instance family and AWS region (e.g., M5 in the US East (N. Virginia) region, known as us-east-1).
- **Instance Size**: They are flexible across various instance sizes (e.g., m5.xlarge, m5.2xlarge).
- **Operating System**: The plans can apply to different operating systems (e.g., Linux, Windows).
- **Tenancy**: Savings Plans can accommodate different tenancy options, such as Host, Dedicated, or Default tenancy models.

EC2 Spot Instances

- **Discount**: EC2 Spot Instances offer up to a 90% discount compared to On-Demand instance prices.
- **Availability**: Spot Instances can be interrupted (i.e., "lost") if AWS needs the capacity back or if the current spot price exceeds the maximum price a user is willing to pay.
- **Cost-Efficiency**: They are the most cost-efficient instances available on AWS, designed to optimize the cost of workloads that can tolerate interruptions.
- **Use Cases**: Spot Instances are ideal for workloads that are resilient to failure. Some examples include:
  - Batch processing jobs
  - Data analysis tasks
  - Image processing operations
  - Distributed workloads that can be spread out over time
  - Workloads that do not require a strict start and end time
- **Limitations**: Spot Instances are not recommended for critical jobs or databases that require constant availability and cannot afford any downtime.
- Essentially, Spot Instances allow users to take advantage of unused EC2 capacity at a significantly reduced cost, with the trade-off being that these instances can be terminated by AWS with little notice if there is a higher demand for capacity or the spot price increases.

EC2 Dedicated Hosts

- **EC2 Dedicated Hosts** are physical servers with EC2 instance capacity that is fully reserved for a single customer's use.
- These hosts allow customers to meet specific compliance requirements and utilize their existing server-bound software licenses, which may be licensed on a per-socket, per-core, or per-VM basis.
- **Purchasing Options**:
  - On-Demand: Customers pay per second for the time the Dedicated Host is active.
  - Reserved: Customers can reserve a Dedicated Host for 1 or 3 years, with varying payment options, including no upfront cost, a partial upfront cost, or the entire cost paid upfront.
- **Cost**: Dedicated Hosts are the most expensive option compared to other EC2 instance types due to the dedicated physical server resources.
- **Use Cases**: They are particularly useful for running software with complex licensing requirements (BYOL - Bring Your Own License) and for companies that have stringent regulatory or compliance requirements.

EC2 Dedicated Instances

- **EC2 Dedicated Instances**: These are virtualized instances that run on hardware dedicated to a single customer. However, they may share this hardware with other instances from the same AWS account.
- **Characteristics and Comparison**:
  - Unlike Dedicated Hosts, Dedicated Instances do not enable the use of dedicated physical servers.
  - Billing for Dedicated Instances is per instance, with an additional $2 fee per region.
  - Dedicated Instances do not offer visibility into the underlying hardware specifics like sockets, cores, or host ID.
  - There's no affinity set between a host and an instance, which means there is no control over the instance's placement on the hardware. After stopping and starting an instance, it could potentially move to different underlying hardware.
  - Dedicated Instances allow automatic placement by AWS, as opposed to Dedicated Hosts, which require manual placement and can be reserved.
  - Only Dedicated Hosts allow adding capacity using an allocation request, giving more control over the physical server and instance placement.
- EC2 Dedicated Instances provide a way to ensure that your instances are isolated at the hardware level within your own account, without sharing the hardware with other AWS accounts. This is suitable for scenarios where you have regulatory or compliance needs that require such isolation. However, you don't have the same level of control or billing options as with Dedicated Hosts, which provide specific, physical servers allocated for your use.

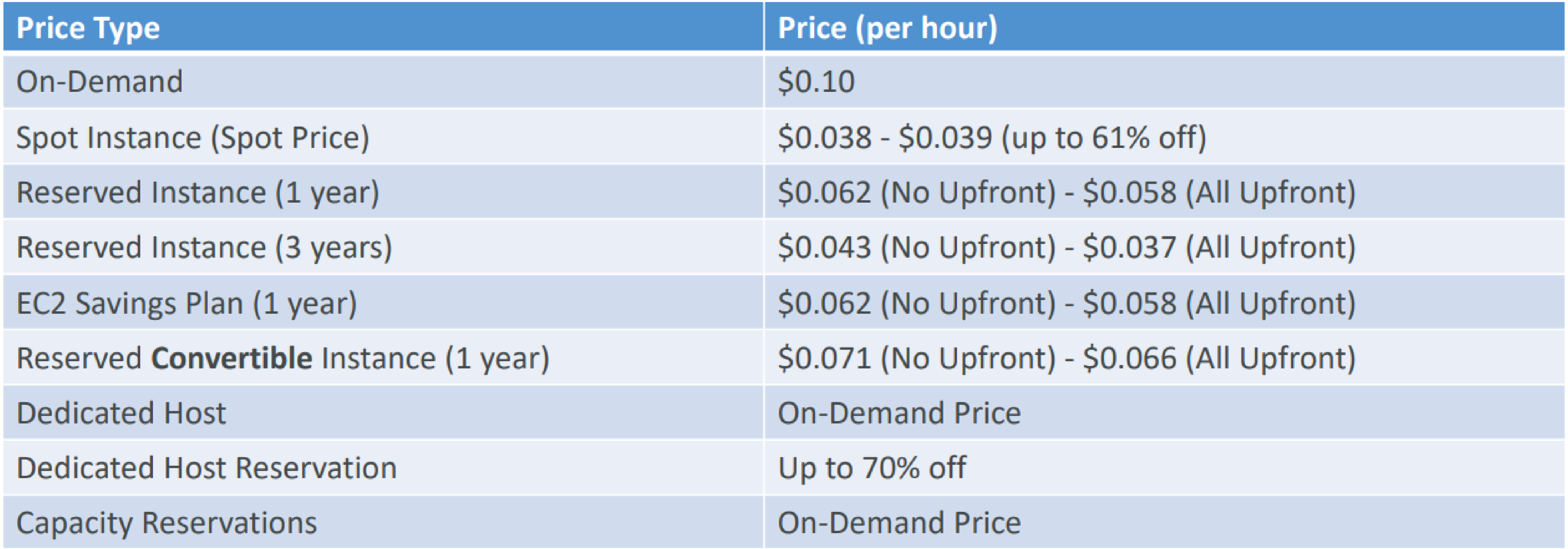
EC2 Capacity Reservations

- **Reserve On-Demand Instances**: Capacity Reservations allow you to reserve capacity for EC2 instances in a specific Availability Zone (AZ) for any duration.
- **Guaranteed Access**: This ensures that you will always have access to EC2 capacity when you need it.
- **Flexibility**: There is no time commitment required. You can create or cancel a reservation at any time, and there are no billing discounts associated with the reservation itself.
- **Combination with Other Pricing Models**: You can combine Capacity Reservations with Regional Reserved Instances and Savings Plans to take advantage of billing discounts.
- **Charges**: You are charged at the On-Demand rate regardless of whether you run your instances or not.
- **Use Case**: Capacity Reservations are suitable for short-term, uninterrupted workloads that need to be in a specific Availability Zone.
- EC2 Capacity Reservations provide a way to ensure you have the necessary compute capacity for your applications when you need it, without the need for long-term commitments or upfront payments. This can be particularly useful for businesses that experience unpredictable bursts of traffic or for event-driven workloads that require resources to be available at a moment's notice.

Purchasing options for Amazon EC2 instances

- **On-Demand**: Likened to coming and staying in a resort whenever you like and paying the full price, this is akin to the flexibility of On-Demand EC2 instances where you pay for compute capacity by the second with no long-term commitments.
- **Reserved**: Compared to planning ahead and getting a discount for a long stay at a hotel, Reserved Instances offer a significant discount compared to On-Demand pricing in exchange for committing to a specific instance type in a region for a term of one or three years.
- **Savings Plans**: Similar to paying a certain amount per hour to stay in any room type at a hotel for a certain period, Savings Plans provide flexible pricing that allows for the use of different instance types across specific regions, offering lower rates in exchange for a commitment to a consistent amount of usage (measured in $/hour) over a 1 or 3-year period.
- **Spot Instances**: Analogous to a hotel allowing people to bid for empty rooms and the highest bidder keeping the room, Spot Instances let you take advantage of unused EC2 capacity at a potentially lower price. However, just like being kicked out of the hotel room if someone else bids higher, Spot Instances can be terminated by AWS with two minutes of notification when AWS needs the capacity back.
- **Dedicated Hosts**: This is compared to booking an entire building of the resort, where a physical server is fully dedicated to your use. This is useful for meeting strict compliance requirements or for using existing server-bound software licenses.
- **Capacity Reservations**: Like booking a hotel room for a period with full price even if you don’t stay in it, Capacity Reservations give you the ability to reserve capacity for your EC2 instances in a specific Availability Zone for any duration, ensuring that you always have access to EC2 capacity when you need it, charged at On-Demand rates.

Purchasing options for Amazon EC2 instances Example: m4.large – us-east-1



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

Shared Responsibility Model for Amazon EC2 (Elastic Compute Cloud)

On AWS's side:

- **Infrastructure**: 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.
- **Isolation on physical hosts**: AWS ensures the isolation of the physical hosts where your EC2 instances run. This means that your instances are logically separated from those of other customers, even though they may be running on the same physical hardware.
- **Replacing faulty hardware**: AWS is in charge of monitoring and maintaining the physical hardware on which EC2 instances run. If hardware fails, AWS will handle the replacement and ensure the infrastructure remains available and reliable.
- **Compliance validation**: AWS manages compliance with various certifications and regulations to ensure the infrastructure is secure and meets the necessary standards.

On the User's side:

- **Security Groups rules**: Users are responsible for setting up and managing the security groups (a type of virtual firewall) that control the inbound and outbound traffic to their EC2 instances.
- **Operating-system patches and updates**: Users must manage the operating system and any applications running on the EC2 instances, including the timely application of patches and updates.
- **Software and utilities installed on the EC2 instance**: Users are also responsible for the security and maintenance of the software and utilities they install on their EC2 instances.
- **IAM Roles assigned to EC2 & IAM user access management**: Users must handle the assignment of IAM (Identity and Access Management) roles and policies to their EC2 instances and users, which govern the permissions and secure access to AWS resources.
- **Data security on your instance**: It is the user’s responsibility to manage the data on their EC2 instances, including encryption options and data integrity.

Purchasing options for Amazon EC2 instances

- **On-Demand**: Likened to coming and staying in a resort whenever you like and paying the full price, this is akin to the flexibility of On-Demand EC2 instances where you pay for compute capacity by the second with no long-term commitments.
- **Reserved**: Compared to planning ahead and getting a discount for a long stay at a hotel, Reserved Instances offer a significant discount compared to On-Demand pricing in exchange for committing to a specific instance type in a region for a term of one or three years.
- **Savings Plans**: Similar to paying a certain amount per hour to stay in any room type at a hotel for a certain period, Savings Plans provide flexible pricing that allows for the use of different instance types across specific regions, offering lower rates in exchange for a commitment to a consistent amount of usage (measured in $/hour) over a 1 or 3-year period.
- **Spot Instances**: Analogous to a hotel allowing people to bid for empty rooms and the highest bidder keeping the room, Spot Instances let you take advantage of unused EC2 capacity at a potentially lower price. However, just like being kicked out of the hotel room if someone else bids higher, Spot Instances can be terminated by AWS with two minutes of notification when AWS needs the capacity back.
- **Dedicated Hosts**: This is compared to booking an entire building of the resort, where a physical server is fully dedicated to your use. This is useful for meeting strict compliance requirements or for using existing server-bound software licenses.
- **Capacity Reservations**: Like booking a hotel room for a period with full price even if you don’t stay in it, Capacity Reservations give you the ability to reserve capacity for your EC2 instances in a specific Availability Zone for any duration, ensuring that you always have access to EC2 capacity when you need it, charged at On-Demand rates.

Amazon EBS (Elastic Block Store)

- The image describes Amazon EBS (Elastic Block Store) Volumes, which are network-attached storage volumes that you can attach to your Amazon EC2 instances. Here are the key points from the image:
- **EBS Volumes**: They act as network drives that can be attached to EC2 instances, allowing for persistent storage of data.
- **Data Persistence**: EBS ensures that data is retained even after the associated EC2 instance is terminated.
- **Single Instance Mount**: At any given time, an EBS volume can be mounted to only one instance within the same EC2 Consolidated Billing family (CCP level).
- **Availability Zone Specific**: Each EBS volume is tied to a specific Availability Zone, meaning it can only be attached to an EC2 instance within the same zone.
- **Analogy**: EBS volumes can be thought of as "network USB sticks" due to their portable and attachable nature.
- **Free Tier**: AWS offers 30 GB of free EBS storage per month, which can be of the General Purpose SSD (Solid State Drive) or Magnetic categories, depending on the user's choice.
- EBS volumes are a crucial component of the AWS ecosystem, providing reliable block-level storage that can be separately managed from the compute instances. They offer the flexibility to upgrade, downgrade, and snapshot as needed, fitting a wide range of use cases from simple storage to high I/O operations.

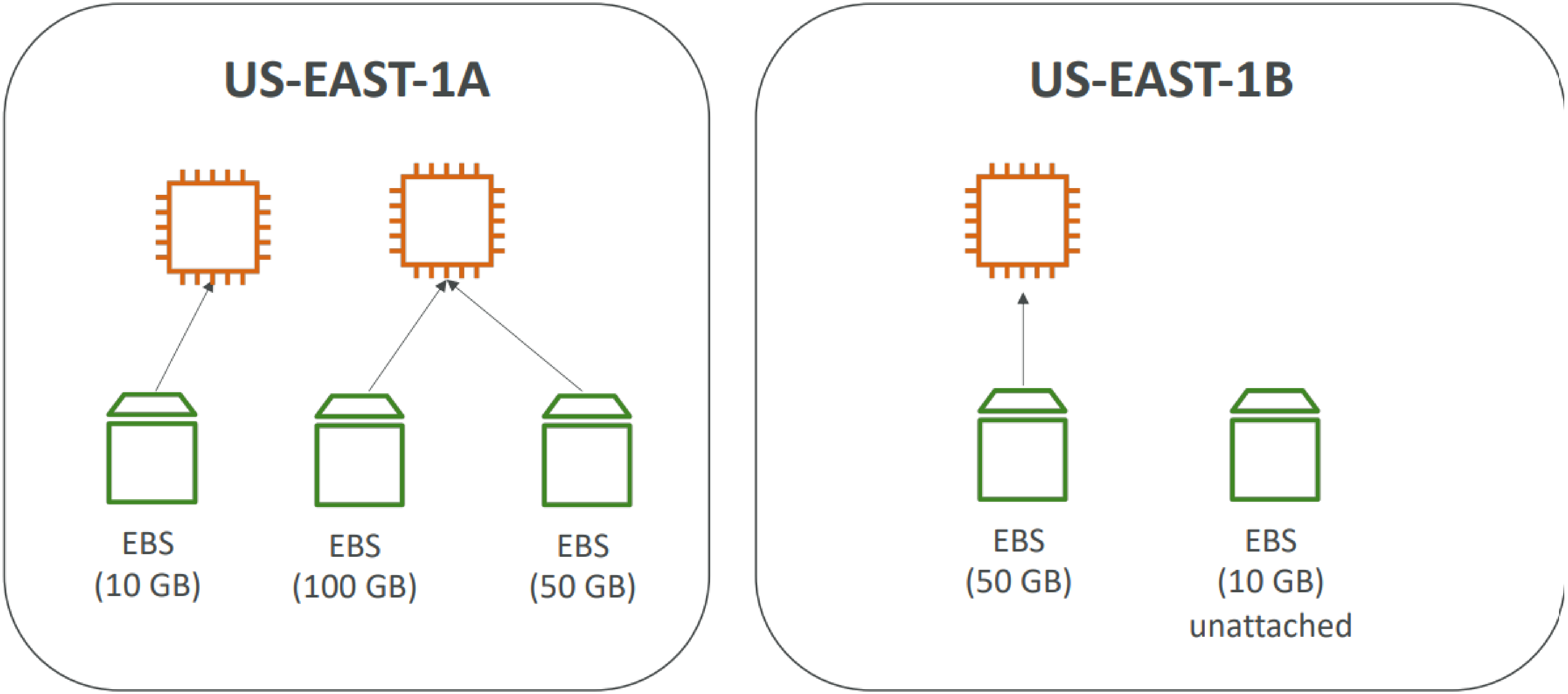
Amazon EBS (Elastic Block Store)

- **Removable Hard Drive (EBS Volume)**: Just as you can connect a removable hard drive to your computer to store files and take it with you to use on another computer, an EBS volume can be attached to an EC2 instance for storage needs and then detached and reattached to another instance if needed.
- **Data Persistence (Data Retention)**: When you unplug the removable hard drive, the data stays on it. Similarly, when you detach an EBS volume from an instance, the data remains intact on the volume.
- **Single Connection (Single Instance Mount)**: A removable hard drive can only be connected to one computer at a time, just as an EBS volume can only be mounted to one EC2 instance at a time.
- **Specific Location (Availability Zone Specific)**: Imagine if your removable hard drive could only be used in certain rooms of your house. In a similar way, EBS volumes are tied to a specific Availability Zone.
- **Free Storage (Free Tier)**: This is like getting a removable hard drive with a limited amount of free storage. If you don't exceed that limit, you don't need to pay for it.

Amazon EBS (Elastic Block Store) functionality and features

- **Network Drive**: An EBS Volume is described as a network drive, meaning it's not a physical hard drive directly attached to the hardware. It communicates with the EC2 instance over a network, which could introduce some latency.
- **Detachment and Reattachment**: It is possible to detach an EBS Volume from one EC2 instance and attach it to another, making storage flexible and portable.
- **Availability Zone Lock**: EBS Volumes are confined to an Availability Zone (AZ). Therefore, a volume in one AZ (e.g., us-east-1a) cannot be directly attached to an instance in a different AZ (e.g., us-east-1b). To move an EBS Volume across AZs, it must be snapshotted, and then the snapshot can be used to create a new volume in the desired AZ.
- **Provisioned Capacity**: EBS Volumes have a provisioned capacity, measured in gigabytes (GBs) and input/output operations per second (IOPS). Users are billed for the capacity they provision, not necessarily what they use. The capacity of an EBS Volume can be increased over time if needed.

Amazon EBS (Elastic Block Store) Example



Amazon EBS (Elastic Block Store) Example-Explanation

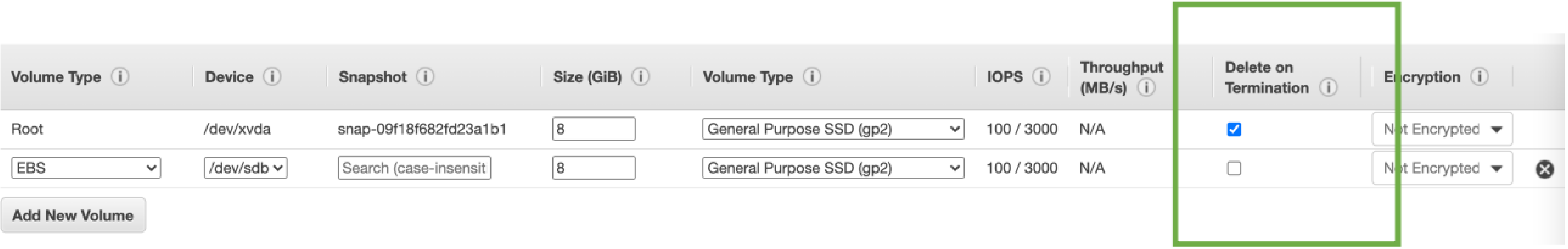
- **US-EAST-1A**: This AZ has two EC2 instances, each with its own attached EBS volumes. The first instance has a 10 GB EBS volume, the second has a 100 GB volume, and the third has a 50 GB volume attached to it.

- **US-EAST-1B**: This AZ shows only one EC2 instance with a single 50 GB EBS volume attached. There is also a 10 GB EBS volume depicted, but it is marked as unattached, meaning it is not currently associated with any EC2 instance.

- The diagram illustrates the concept that EBS volumes are bound to a specific AZ and can be attached to EC2 instances within the same AZ. To use an EBS volume in a different AZ, it would need to be snapshotted and the snapshot would then be used to create a new volume in the target AZ. The image also demonstrates the ability to have EBS volumes of varying sizes and the flexibility to attach and detach them from EC2 instances as required.

Amazon EBS Delete on Termination

- This attribute controls whether an EBS volume is automatically deleted when the EC2 instance it is attached to is terminated.
- By default, the root EBS volume (the primary storage volume that contains the operating system) has this attribute enabled, meaning it will be deleted upon the instance's termination.
- Any additional EBS volumes attached to the instance have this attribute disabled by default, meaning they will not be deleted when the instance is terminated. This allows for the preservation of data on these volumes even after the instance is no longer running.
- Users have the option to modify this attribute via the AWS Management Console or the AWS Command Line Interface (CLI). This can be particularly useful when there's a need to preserve the root volume after the instance has been terminated, for example, for backup or recovery purposes.



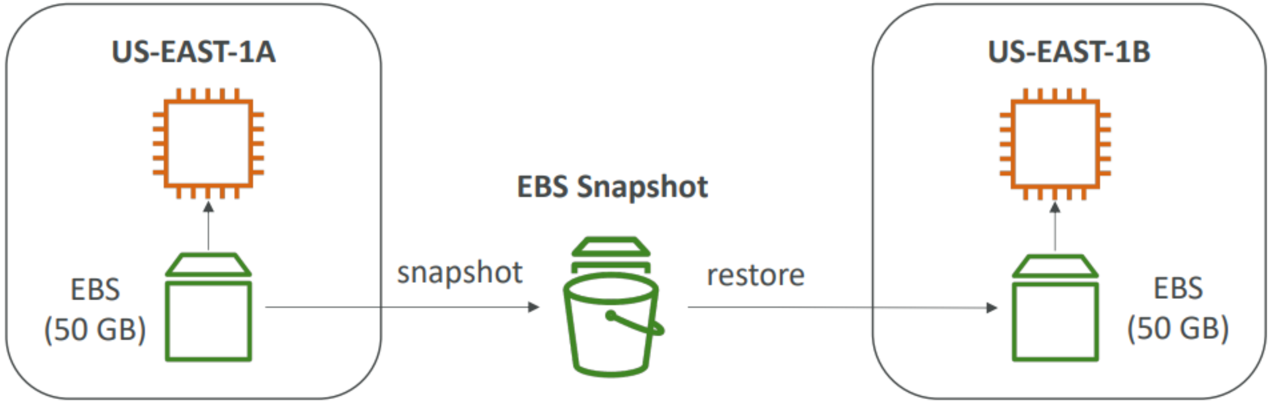
Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-09f18f682fd23a1b1	8	General Purpose SSD (gp2)	100 / 3000	N/A	☑	Not Encrypted
EBS	/dev/sdb	Search (case-insensit	8	General Purpose SSD (gp2)	100 / 3000	N/A	□	Not Encrypted

Amazon EBS

HANDS ON PRACTICE

Amazon EBS (Elastic Block Store) Snapshots

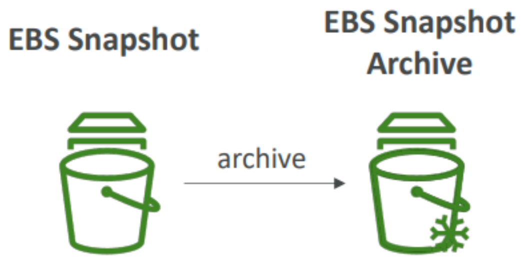
- Snapshots serve as a backup for an EBS volume at a certain point in time, ensuring data is not lost.
- It's not mandatory to detach the EBS volume from the EC2 instance when taking a snapshot, although it's recommended for consistency.
- Snapshots can be transferred across different Availability Zones (AZs) or AWS Regions, aiding in data recovery and geographical diversification.
- The graphic illustrates an EBS volume in the US-EAST-1A AZ being snapshotted and the snapshot being used to restore or create a new EBS volume in the US-EAST-1B AZ. This demonstrates the portability of EBS snapshots across different locations within AWS.

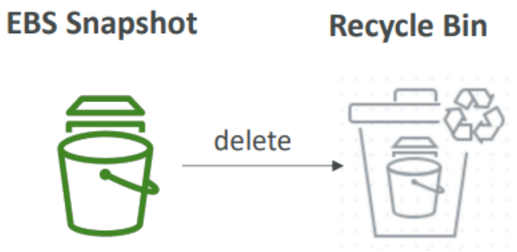


Amazon EBS (Elastic Block Store) Snapshots

1. EBS Snapshot Archive:
    1. This feature allows users to move less frequently accessed snapshots to an archive tier, which costs 75% less than the standard snapshot storage tier.
    2. The archived snapshots take between 24 to 72 hours to restore, indicating that this option is suited for backups that are not expected to be needed for immediate recovery.
2. Recycle Bin for EBS Snapshots:
    1. A recycle bin can be set up for EBS snapshots to retain deleted snapshots for a certain period, which can range from 1 day to 1 year.
    2. This provides a safety net allowing for the recovery of snapshots that may have been accidentally deleted.

- These features are designed to offer cost savings and additional data protection for users leveraging EBS volumes in their AWS environments. The associated icons visually represent the process of archiving and deleting snapshots, with an archive symbol for moving to the cheaper storage tier and a recycle bin for temporarily retaining deleted snapshots.





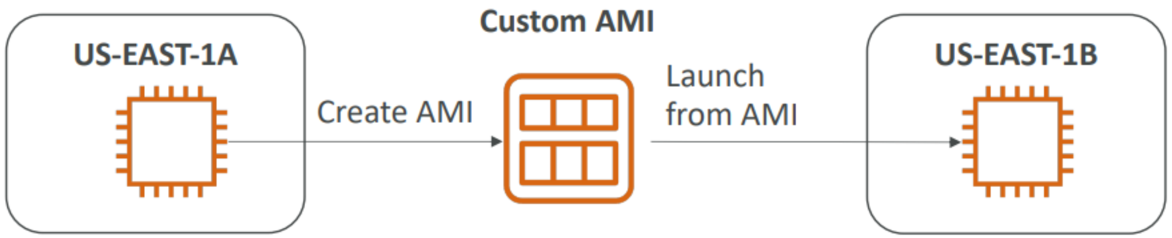
Amazon Machine Images (AMIs)

- AMI stands for Amazon Machine Image.
- AMIs are a customization of an EC2 instance, which include user-defined software, configurations, and the operating system.
- The use of an AMI allows for a faster boot and configuration time because the software comes pre-packaged.
- AMIs are specific to an AWS region but can be copied to other regions.
- There are different types of AMIs:
  - Public AMIs: Provided by AWS and available to all users.
  - Your own AMIs: Custom AMIs created and maintained by the user.
  - AWS Marketplace AMIs: Created by third-party sellers and can be purchased or licensed through the AWS Marketplace.
- This setup facilitates quick deployment, management, and scalability of EC2 instances for various applications.

Creating an Amazon Machine Image (AMI) from an existing EC2 instance

1. Start an EC2 instance and customize it with the necessary software, configurations, and settings.
2. Stop the instance to ensure data integrity before creating an image. This step is crucial as it prevents any write operations that could corrupt the snapshot.
3. Build an AMI from the stopped EC2 instance. This process will create snapshots of the Elastic Block Store (EBS) volumes attached to the instance.
4. Once the AMI is created, it can be used to launch new instances in the same or different AWS regions, enabling quick scaling or deployment across various environments.

- The diagram shows an EC2 instance in the US-EAST-1A region being used to create a custom AMI. This AMI can then be utilized to launch instances in another availability zone, like US-EAST-1B. The AMI encapsulates all the customizations and serves as a template for new instances.



Analogous to creating an AMI from an EC2 instance

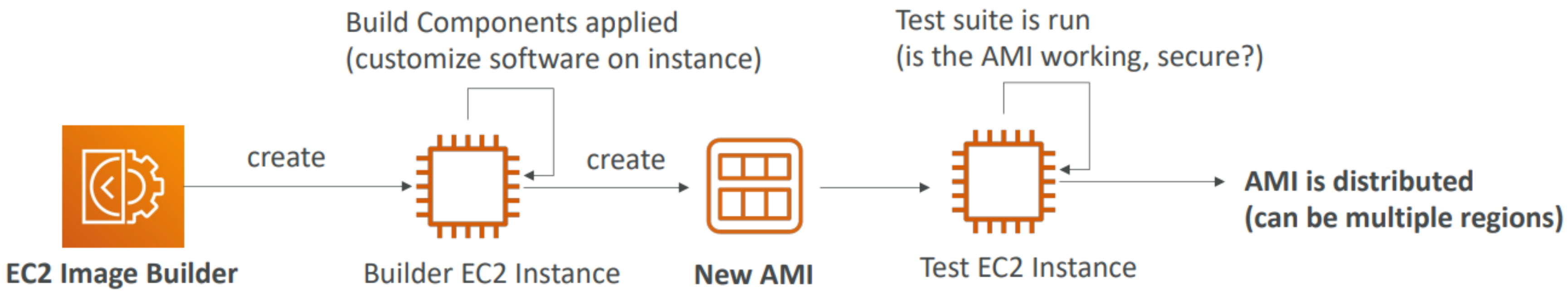
- Imagine having a fully decorated and furnished apartment that you like so much you want to replicate it in different locations. Here’s how the process would look:

1. You carefully select and arrange furniture, decorations, and appliances in your apartment until it’s exactly as you wish.
2. You then create detailed blueprints and take photos of your apartment to capture its current state perfectly.
3. With these blueprints and photos, you can now guide interior designers to replicate your apartment setup in a new building in a different city.
4. Whenever you travel to these other cities, you can stay in an apartment that feels just like home, as each one is set up identically to your original space, thanks to your detailed instructions.

- In this analogy, the original apartment is the EC2 instance, the blueprints and photos are the AMI, and the replicated apartments in different cities are the new EC2 instances launched from the AMI.

EC2 Image Builder

- Automating the creation, maintenance, validation, and testing of EC2 AMIs (Amazon Machine Images).
- The ability to schedule these operations, such as on a weekly basis or when updates are available.
- EC2 Image Builder is a free service, and users only pay for the resources used during the image creation process.
- The diagram shows the flow from building components applied to an EC2 instance (customizing software on the instance), creating a new AMI, testing the AMI to ensure it's working correctly and secure, and then distributing the AMI, possibly across multiple regions.
- This service streamlines the process of managing and deploying custom AMIs, ensuring that they are up to date and meet specific requirements before being used to launch new EC2 instances.



EC2 Image Builder-Example

- Imagine you own a company that designs custom cars. To efficiently produce these cars, you have an assembly line that automates the entire process.

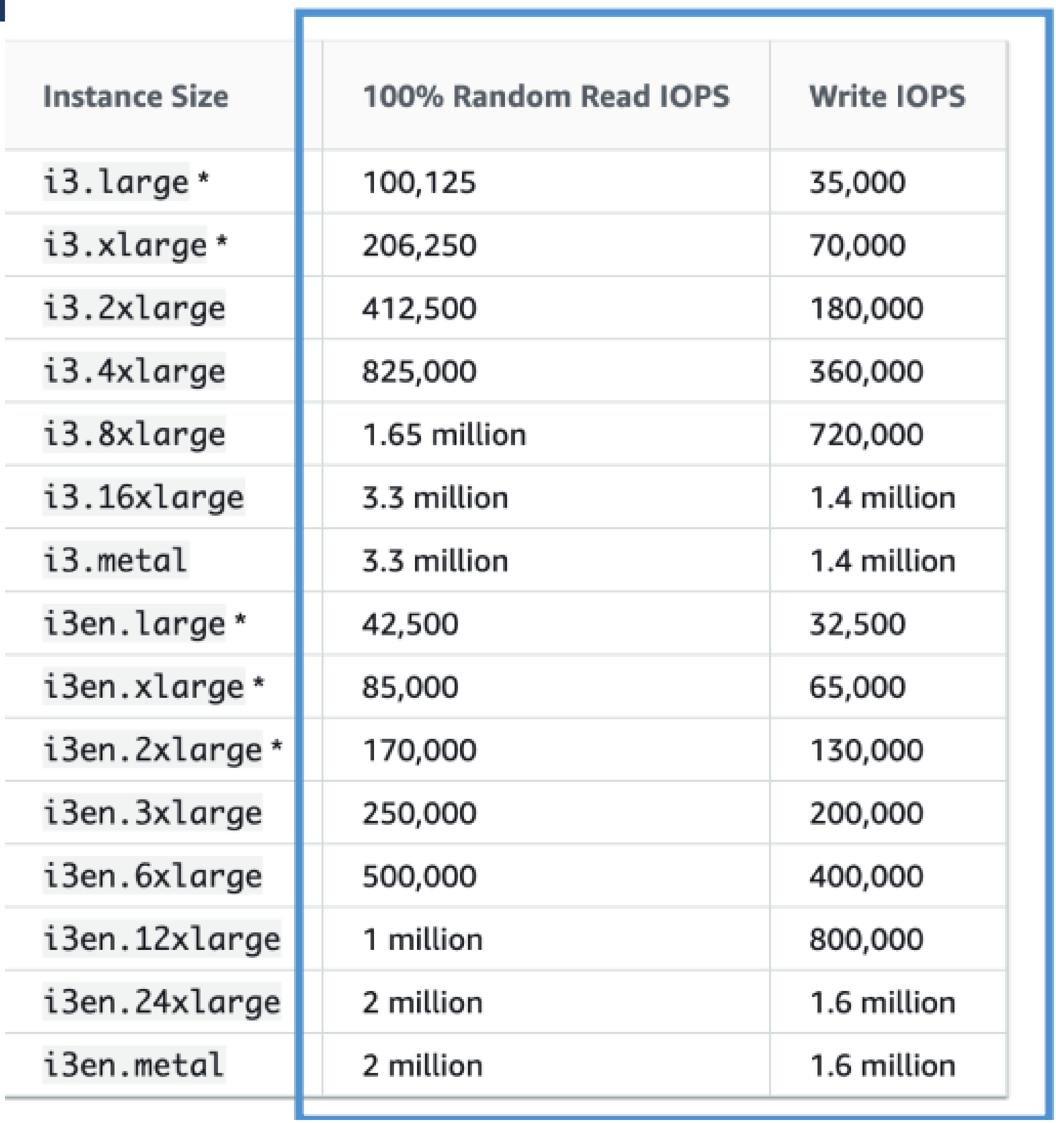
1. **Customization:** Just as you would select specific features for a car (like the engine type, paint color, interior materials), the EC2 Image Builder allows you to customize software configurations, operating systems, and applications on a base EC2 instance.
2. **Scheduling:** The way a car factory might operate on a schedule to release new models or updates to existing models, EC2 Image Builder can be scheduled to create new AMIs when updates or new packages are available.
3. **Testing:** Before a car is sent to a dealership, it undergoes rigorous testing to ensure safety and reliability. Similarly, the EC2 Image Builder runs tests to ensure the AMI is secure and functioning as expected.
4. **Distribution:** Once the cars are ready and tested, they are shipped to dealerships across the country or globally. In the same way, once an AMI is created and tested, it can be distributed across multiple AWS regions for use.

- So, the EC2 Image Builder is like the behind-the-scenes assembly line for creating reliable, tested, and ready-to-deploy server images for AWS, just as a car assembly line is for producing ready-to-sell vehicles.

EC2 Instance Store

- **EC2 Instance Store**: This refers to temporary storage provisioned on the same physical server that hosts an EC2 instance. Unlike EBS volumes, which are persistent network drives with consistent performance, Instance Store provides high-performance hardware disks directly attached to the host machine.
- **Performance**: EC2 Instance Store volumes offer better I/O performance compared to EBS volumes due to their direct attachment to the host hardware.
- **Ephemerality**: The storage on an EC2 Instance Store is ephemeral, meaning it is lost if the instance is stopped or terminated. Therefore, it's suitable for temporary storage needs like buffer, cache, or scratch data.
- **Risk of Data Loss**: Since the data is stored directly on hardware, there's a risk of data loss if the hardware fails. Hence, it is not recommended for long-term data retention.
- **Backup and Replication**: The responsibility of backing up and replicating data in the EC2 Instance Store lies with the user, not AWS.
- The key takeaway is that if your application requires temporary, high-performance storage and you can manage the risks associated with its ephemeral nature, then EC2 Instance Store could be a suitable option.

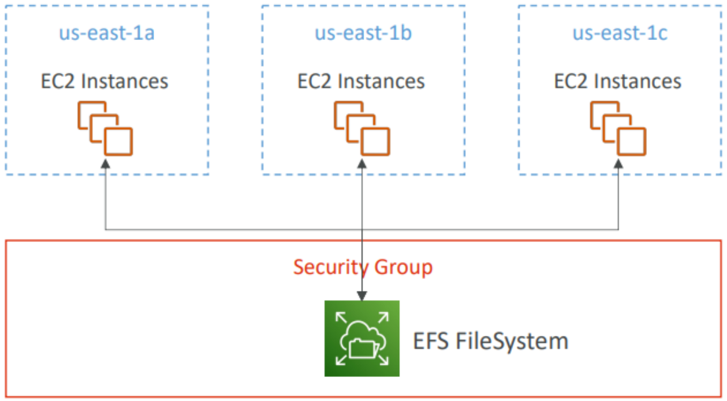
EC2 Instance Store



Instance Size	100% Random Read IOPS	Write IOPS
i3.large *	100,125	35,000
i3.xlarge *	206,250	70,000
i3.2xlarge	412,500	180,000
i3.4xlarge	825,000	360,000
i3.8xlarge	1.65 million	720,000
i3.16xlarge	3.3 million	1.4 million
i3.metal	3.3 million	1.4 million
i3en.large *	42,500	32,500
i3en.xlarge *	85,000	65,000
i3en.2xlarge *	170,000	130,000
i3en.3xlarge	250,000	200,000
i3en.6xlarge	500,000	400,000
i3en.12xlarge	1 million	800,000
i3en.24xlarge	2 million	1.6 million
i3en.metal	2 million	1.6 million

Amazon Elastic File System (EFS)

- **EFS - Elastic File System**: EFS is a managed file storage service provided by AWS.
- **Managed NFS**: It operates as a managed Network File System (NFS) that can be attached to multiple EC2 instances across different Availability Zones (AZs).
- **Compatibility**: EFS is compatible with Linux EC2 instances, meaning it can be used in environments that are running on Linux.
- **Availability and Scalability**: The service is designed to be highly available and scalable. EFS automatically grows and shrinks as you add and remove files, so you don’t need to manage capacity.
- **Cost**: It is noted to be more expensive, approximately three times the cost of General Purpose SSD (gp2) volumes, and it operates on a pay-per-use model, eliminating the need for capacity planning.
- The visual representation shows the EFS being accessed by EC2 instances located in different Availability Zones, all within the same region, indicating the cross-AZ functionality of EFS. Additionally, there's a security group symbol associated with the EFS, which implies that access to the file system can be controlled and secured using AWS security groups.



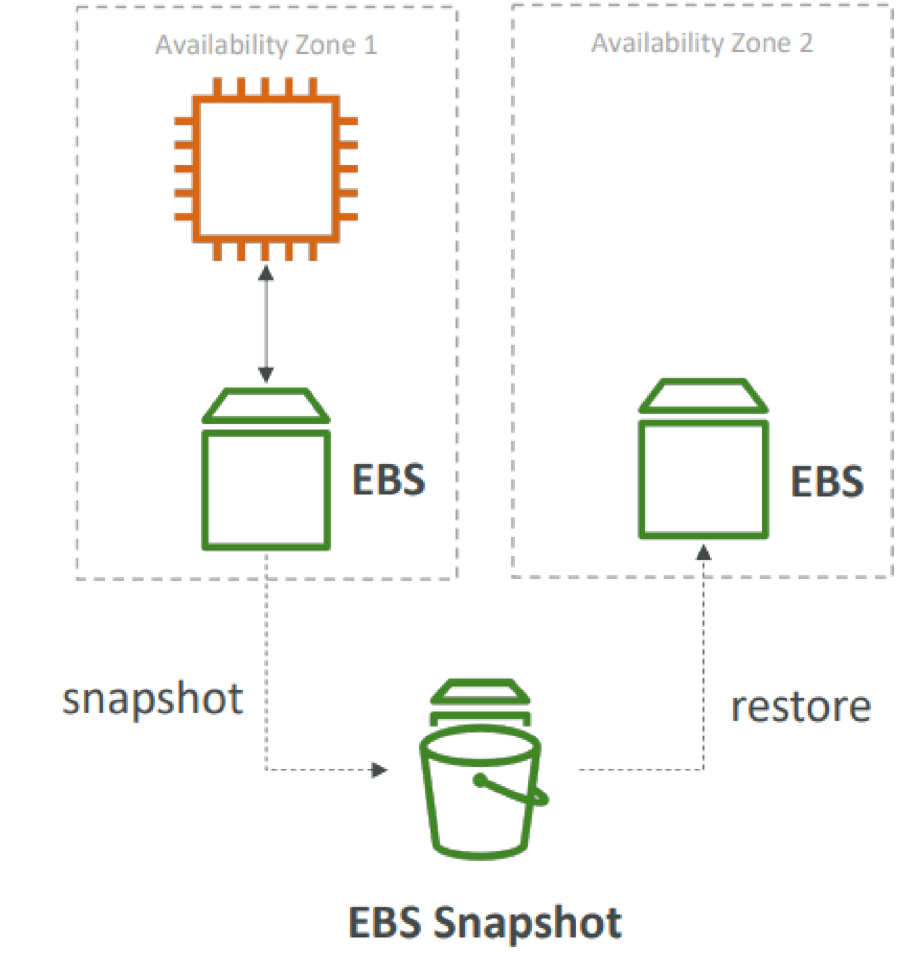
Amazon Elastic File System (EFS)

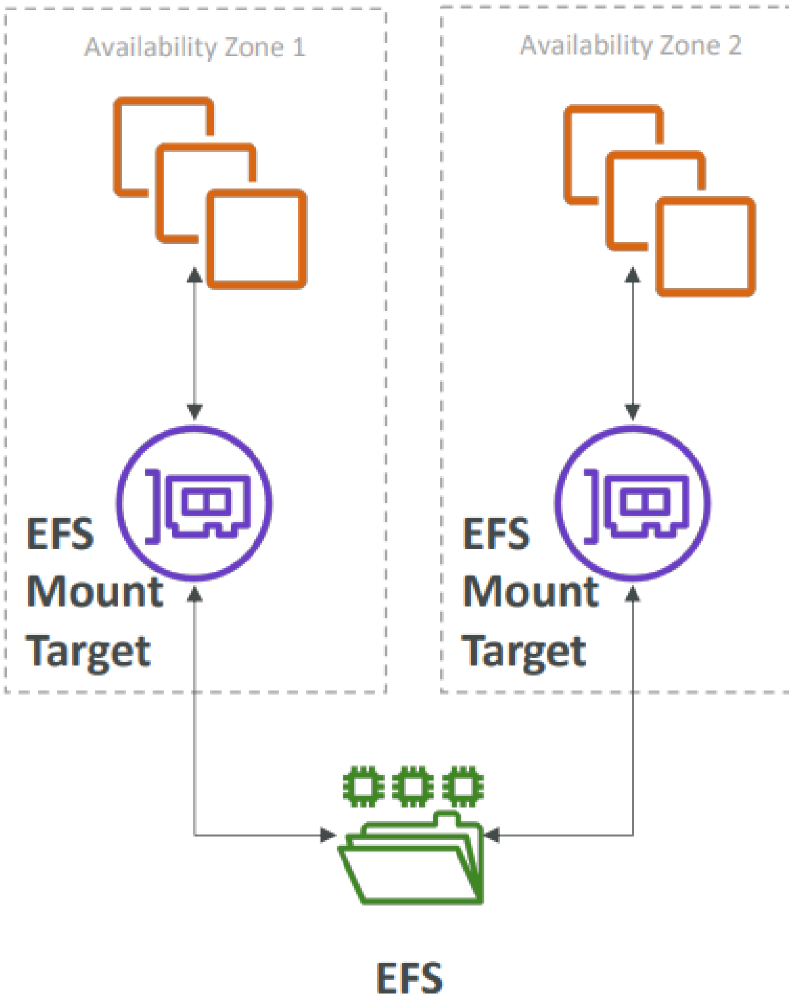
- **Managed NFS (Network File System)**: Think of EFS as a central library system that allows multiple people to access a variety of books (files) from different locations (EC2 instances).
- **Compatibility with Linux EC2 instances in multi-AZ**: Similar to how a library card allows you to borrow books from any branch within the library system across the city (availability zones).
- **Highly available and scalable**: Just like a library system can offer more copies of popular books or take them away based on demand, EFS automatically adjusts the available storage based on how much you're using at any time.
- **Expensive, pay per use, no capacity planning**: This is akin to a premium library service where you pay a fee based on the number of books you borrow, without needing to purchase and store the books yourself.
- In this analogy, the EFS would be like a shared resource that is available to all library members (EC2 instances), with the flexibility and reliability of the library system ensuring that the books (files) are available when and where they're needed, albeit at a higher cost for premium access and services.

Amazon Elastic File System (EFS)

- **Managed NFS (Network File System)**: Think of EFS as a central library system that allows multiple people to access a variety of books (files) from different locations (EC2 instances).
- **Compatibility with Linux EC2 instances in multi-AZ**: Similar to how a library card allows you to borrow books from any branch within the library system across the city (availability zones).
- **Highly available and scalable**: Just like a library system can offer more copies of popular books or take them away based on demand, EFS automatically adjusts the available storage based on how much you're using at any time.
- **Expensive, pay per use, no capacity planning**: This is akin to a premium library service where you pay a fee based on the number of books you borrow, without needing to purchase and store the books yourself.
- In this analogy, the EFS would be like a shared resource that is available to all library members (EC2 instances), with the flexibility and reliability of the library system ensuring that the books (files) are available when and where they're needed, albeit at a higher cost for premium access and services.

(EFS) and EBS


Availability Zone 1
Availability Zone 2
EBS
EBS
snapshot
restore
EBS Snapshot


Availability Zone 1
Availability Zone 2
EFS Mount Target
EFS Mount Target
EFS

(EFS) and EBS

- EBS is depicted as a storage volume that's available within a single Availability Zone. Data persistence is handled through snapshots, which can be used to restore data. The snapshot feature allows the creation of a point-in-time copy of an EBS volume, which can be used for backups or to create new volumes.

- EFS is shown as a network file system that spans multiple Availability Zones, providing a higher level of availability and redundancy. EFS volumes are mounted on EC2 instances as network drives, and the same file system can be accessed by multiple instances simultaneously, which is not possible with EBS.

- The main points of contrast between EBS and EFS in this image are:

- **Availability Zone Restriction**: EBS is confined to a single Availability Zone, while EFS is accessible across multiple Availability Zones, indicating its distributed nature.

- **Mounting**: EBS volumes are attached to individual EC2 instances, whereas EFS file systems are mounted on EC2 instances, indicating that EFS provides a shared file system for multiple EC2 instances.

- **Data Backup and Recovery**: EBS relies on snapshots for data backup, which can be restored into new volumes within the same Availability Zone. In contrast, EFS doesn't require snapshots as it is designed to be more durable and highly available.

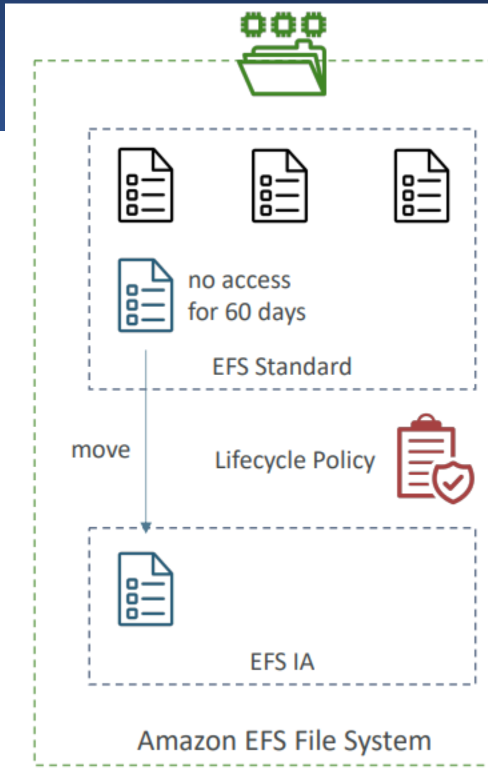
- The image aims to convey the suitability of each storage option for different use cases: EBS for single-instance storage with snapshots for backup and EFS for multi-instance, highly available shared storage.

(EFS) and (EBS) Example

- **EBS (USB Flash Drive Analogy)**: Imagine EBS as a USB flash drive that you can attach to a single computer (EC2 instance) at a time. It is dedicated to that computer, and if you want to use it with a different computer, you have to unplug it from the first one and plug it into another. If you want to keep the data after you no longer need the USB drive, you would make a copy (snapshot) of its contents, which you can then use to restore the data onto a new USB drive later.
- **EFS (Network Shared Drive Analogy)**: On the other hand, EFS is like a network shared drive that's accessible by multiple computers (EC2 instances) across an office network (multi-AZ environment). No matter which computer you are using, you can access the shared drive and the files on it. You don’t need to make copies of the drive to move it to another computer; it’s always there and available to any computer connected to the network.
- So, in summary:
- EBS is a single-attach, high-performance storage solution like a USB drive that you can snapshot for backups.
- EFS is a multi-attach, scalable storage solution like a network shared drive that provides shared access to data across different resources.

EFS Infrequent Access (EFS-IA) storage

- **EFS Infrequent Access (EFS-IA)**: This is a cost-optimized storage class for files that are not accessed daily.
- **Cost Savings**: Using EFS-IA can lead to up to a 92% cost reduction compared to the standard EFS pricing.
- **Automatic Transition**: Files are automatically moved to EFS-IA based on the last access time. This transition is governed by a Lifecycle Policy.
- **Lifecycle Policy**: This policy is set up to move files that haven't been accessed for a defined period, such as 60 days, to the EFS-IA class to save costs.
- **Transparency**: The process is transparent to the applications using the file system; they can access the files irrespective of the storage class.
- In essence, EFS-IA is suitable for storing files that are not accessed frequently but still need to be available without manual intervention when needed. It provides a way to reduce storage costs for such data without affecting application performance.



EFS Infrequent Access (EFS-IA) storage-Example

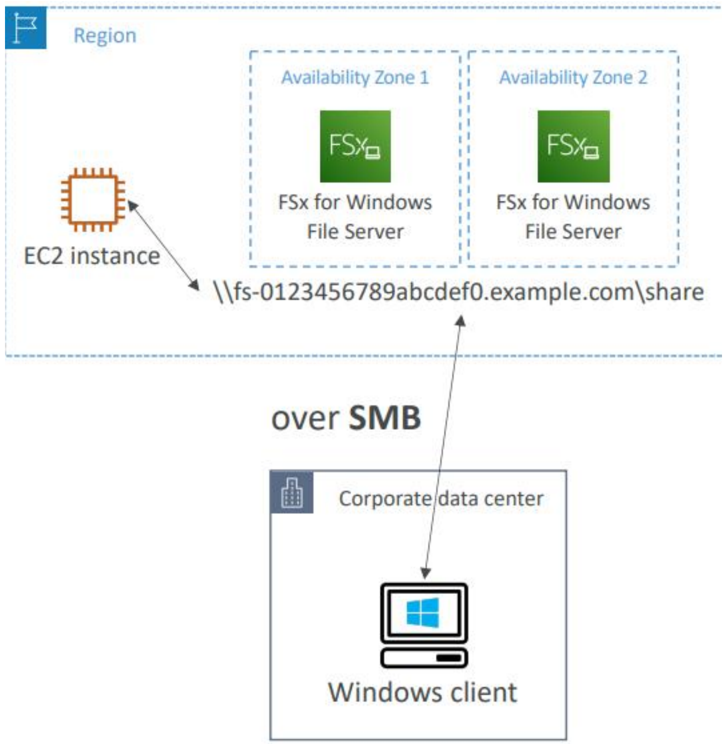
- **Regular Shelves (EFS Standard)**: This is like the main area of the library where books are readily available for borrowing. It's convenient for books that are checked out frequently. However, space in this area is premium and costs more to maintain due to high demand.
- **Storage Room (EFS-IA)**: Consider this as a separate storage area in the library for books that are rarely borrowed. They are kept away from the main shelves because they don't need to be accessed as often. When a patron requests one of these books, a librarian retrieves it, but it might take a little longer than grabbing a book from the regular shelves. The storage room is less expensive to maintain due to its lower accessibility and use.
- **Lifecycle Policy**: This is like the library's system that reviews which books haven't been checked out for a long time (e.g., 60 days) and moves them to the storage room to make space for new or more popular books.
- **Transparency**: To the library patrons, the process is seamless. They request a book, and the library provides it, regardless of whether it comes from the regular shelves or the storage room. They don't need to know where the book was stored, just that they can access it when needed.
- In this analogy, the EFS-IA represents the storage room where books (or data files) are kept until needed, thereby saving space and costs in the library's main area (or the EFS standard storage). The lifecycle policy is the rule by which books are moved back and forth, ensuring that only those needed regularly take up the prime space.

Shared Responsibility Model for EC2 Storage

- **AWS Responsibilities:**
  - Maintaining the **infrastructure**, which includes the physical security and integrity of the data centers where the EC2 instances are hosted.
  - **Replication of data** for EBS (Elastic Block Store) volumes and EFS (Elastic File System) drives to ensure durability and availability.
  - **Replacing faulty hardware**, ensuring that the physical components supporting EC2 storage are in working order.
  - **Ensuring that AWS employees cannot access customer data**, which involves implementing and maintaining strict access controls and policies.
- **User Responsibilities:**
  - **Setting up backup/snapshot procedures** to protect data against accidental deletion or loss.
  - **Setting up data encryption** to protect data both at rest and in transit.
  - Taking responsibility for **any data on the drives**, including compliance with laws and regulations.
  - Understanding the **risk of using EC2 Instance Store**, which provides temporary block-level storage but does not persist if the instance is stopped or terminated.

Amazon FSx for Windows

- ❑ Amazon FSx for Windows File Server is a service provided by Amazon that offers a fully managed, highly reliable, and scalable file system designed specifically for Windows environments. It is built on the foundation of Windows File Server, ensuring compatibility and familiarity for Windows users.
- ❑ Key features of Amazon FSx for Windows File Server include support for the SMB (Server Message Block) protocol and Windows NTFS (New Technology File System), which are standard components of Windows file sharing. This enables seamless integration with existing Windows-based applications and workflows.
- ❑ Moreover, Amazon FSx for Windows File Server is tightly integrated with Microsoft Active Directory, facilitating easy authentication and access control management. This means that users and resources can be managed using familiar Active Directory tools and policies.
- ❑ Another notable aspect of Amazon FSx for Windows File Server is its accessibility. It can be accessed not only from within the AWS (Amazon Web Services) cloud environment but also from on-premise infrastructure. This flexibility allows organizations to seamlessly extend their file storage capabilities to the cloud while maintaining connectivity with their existing infrastructure.
- ❑ Amazon FSx for Windows File Server provides a robust and efficient solution for organizations requiring shared file storage in Windows environments, offering the reliability, scalability, and manageability needed to support a wide range of applications and workloads



Amazon FSx for Windows-Example

- Imagine Amazon FSx for Windows File Server as a centrally located, fully managed storage warehouse specifically designed for storing and managing files in a Windows environment.
- **Managed Storage Warehouse**: Just like how a storage warehouse is managed by a team of professionals who ensure its proper functioning, Amazon FSx for Windows File Server is fully managed by Amazon, relieving users of the burden of maintaining and managing the file system infrastructure.
- **Reliability and Scalability**: Similar to how a well-built storage warehouse is designed to withstand various environmental conditions and can be expanded to accommodate more goods as needed, Amazon FSx for Windows File Server offers high reliability and scalability, ensuring that files are stored securely and can be accessed and expanded as the storage needs grow.
- **Windows Compatibility**: Just as a storage warehouse can be tailored to store specific types of goods, Amazon FSx for Windows File Server is specifically designed to support Windows file systems, ensuring compatibility with Windows-based applications and workflows.
- **Integration with Active Directory**: Much like how a storage warehouse may integrate with a central inventory management system, Amazon FSx for Windows File Server seamlessly integrates with Microsoft Active Directory, enabling easy authentication and access control management for users and resources.
- **Accessibility**: Similar to how a storage warehouse can be accessed by both warehouse staff and external partners, Amazon FSx for Windows File Server can be accessed from within the AWS cloud environment as well as from on-premise infrastructure, providing flexibility and accessibility to users and applications regardless of their location.
- In summary, Amazon FSx for Windows File Server serves as a reliable, scalable, and seamlessly integrated storage solution tailored for Windows environments, much like a well-managed and adaptable storage warehouse for digital files.

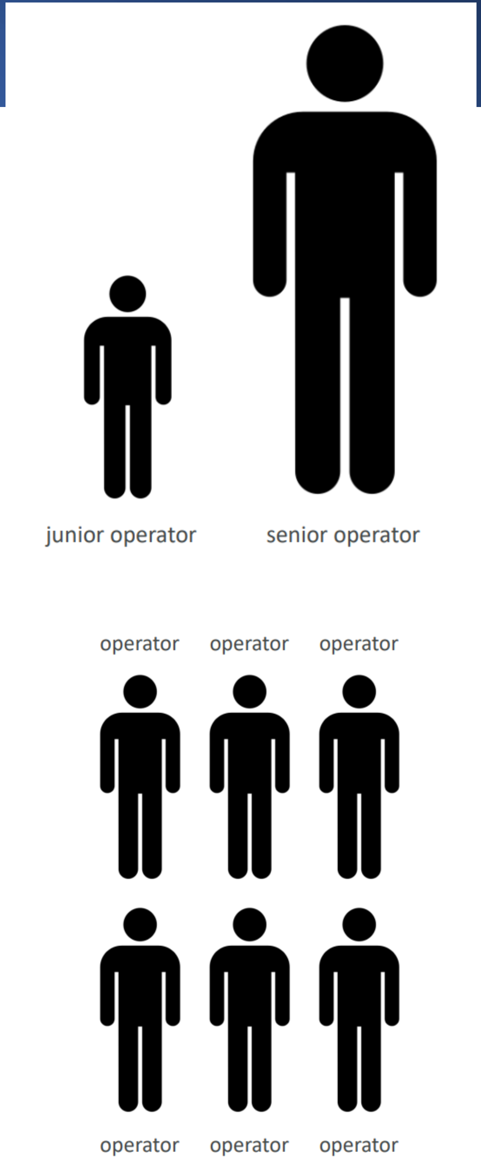
Scalability

- Scalability refers to the ability of an application or system to handle increasing workloads by adapting to accommodate higher demands. There are two primary types of scalability:

1. **Vertical Scalability**: This involves increasing the capacity of individual instances or resources within a system. For instance, if your application is currently running on a small instance like a t2.micro, scaling vertically would entail upgrading to a larger instance such as a t2.large. Vertical scalability is often utilized in non-distributed systems, like databases. However, there's typically a limit to how much you can vertically scale due to hardware constraints.

2. **Horizontal Scalability (Elasticity)**: This entails increasing the number of instances or systems for your application, effectively distributing the workload across multiple resources. Unlike vertical scalability, horizontal scalability is commonly associated with distributed systems. For example, for web applications or modern applications, it's typical to employ horizontal scaling to handle increased traffic. Cloud offerings like Amazon EC2 make horizontal scaling easier by providing the capability to quickly add more instances as needed.

- Scalability is closely related to but distinct from High Availability (HA). While scalability focuses on handling greater loads, HA emphasizes ensuring continuous and reliable access to the system or application, minimizing downtime and disruptions.



Scalability & High Availability

- Let's explore this distinction further using a call center as an example:
- **Scalability**: In the context of a call center, scalability would involve the ability to handle a growing volume of incoming calls by adjusting resources dynamically. This could mean adding more phone lines or agents (horizontal scalability) or upgrading equipment to handle more calls per agent (vertical scalability).
- **High Availability**: High availability, on the other hand, would focus on ensuring that the call center remains operational even in the event of hardware failures, network issues, or other disruptions. This might involve redundant systems, failover mechanisms, and proactive monitoring to detect and mitigate potential issues before they impact service availability.
- In summary, while scalability addresses the ability to accommodate increased demand, High Availability focuses on maintaining continuous access to the system or application, regardless of external factors. Both are essential considerations for ensuring optimal performance and reliability in mission-critical environments like call centers.

Scalability & High Availability-Example

- Let's use the analogy of a transportation system to explain the concepts of scalability and high availability:
- **Scalability**: Imagine a city's transportation system, such as its network of buses. Scalability in this context would be akin to the system's ability to handle an increasing number of commuters during peak hours or special events.
- **Vertical Scalability**: Increasing vertical scalability would involve upgrading individual buses to larger models with more seating capacity or better performance. For instance, replacing standard buses with articulated buses that can carry more passengers.
- **Horizontal Scalability**: Horizontal scalability would involve adding more buses to the fleet to accommodate the growing number of commuters. This could mean deploying additional buses on existing routes or introducing new routes to serve previously underserved areas.
- **High Availability**: Now, let's consider high availability in the transportation system context. High availability would ensure that the transportation system remains operational and accessible to commuters even in the face of disruptions such as accidents, road closures, or inclement weather.
- **Redundancy**: High availability measures might include having redundant buses available to replace any that experience mechanical issues or scheduling backups for critical routes to ensure continuous service.
- **Failover Mechanisms**: Similar to how traffic can be rerouted through alternative routes in case of road closures, the transportation system might have contingency plans in place to reroute buses or provide alternative modes of transportation during emergencies.
- In this analogy, scalability addresses the system's ability to handle increasing demand, while high availability focuses on ensuring uninterrupted access to transportation services, even in challenging circumstances. Both are crucial aspects of a robust and reliable transportation system, just as they are in IT infrastructure and applications

Scalability & High Availability for EC2 isntance

- High Availability and Scalability are crucial considerations when deploying applications on Amazon EC2 (Elastic Compute Cloud) instances. Let's explore how these concepts apply to EC2:
- **Vertical Scaling:** This involves increasing the size of individual EC2 instances, often referred to as "scaling up" or "scaling down." In the context of Amazon EC2, you can vertically scale an instance by changing its instance type to one with more or fewer resources. For example:
  - ❑ From: t2.nano instance with 0.5 GB of RAM and 1 vCPU
  - ❑ To: u-12tb1.metal instance with 12.3 TB of RAM and 448 vCPUs
- Vertical scaling is suitable for applications that require more resources on a single instance, such as memory-intensive databases or CPU-bound tasks.
- **Horizontal Scaling:** This involves increasing the number of EC2 instances, often referred to as "scaling out" or "scaling in." Horizontal scaling is achieved by deploying multiple instances of the application and distributing the workload across them. In Amazon EC2, horizontal scaling is commonly implemented using:
- **Auto Scaling Group:** This service automatically adjusts the number of EC2 instances in response to changes in demand or based on predefined metrics. It ensures that the desired number of instances are running to handle the workload efficiently.
- **Load Balancer**: A load balancer distributes incoming traffic across multiple EC2 instances, ensuring that no single instance becomes overwhelmed. It enhances the availability and performance of the application by distributing traffic evenly.

Scalability & High Availability-Example

- High Availability: High Availability ensures that an application remains accessible and operational, even in the event of component failures or disruptions. In Amazon EC2, achieving high availability involves:
- Running instances across multiple Availability Zones (AZs): Amazon EC2 offers the ability to deploy instances in different geographic locations known as Availability Zones. By spreading instances across multiple AZs, you can enhance fault tolerance and minimize downtime caused by AZ-level failures.
- **Auto Scaling Group with multi-AZ configuration**: Configuring the Auto Scaling Group to span multiple AZs ensures that instances are distributed across different physical locations, reducing the impact of AZ failures on application availability.
- **Load Balancer with multi-AZ configuration**: Similarly, configuring the load balancer to operate across multiple AZs enables it to distribute traffic to instances in different AZs, improving fault tolerance and ensuring continuous availability.
- In summary, for applications deployed on Amazon EC2, achieving both high availability and scalability involves a combination of vertical and horizontal scaling strategies, along with deploying instances across multiple Availability Zones for resilience against failures

Scalability & Elasticity

- **Scalability**: Scalability refers to the ability of a system or application to handle increased loads by either making the existing hardware stronger (scaling up) or by adding more nodes or instances (scaling out). It focuses on the capacity of the system to accommodate larger workloads while maintaining or improving performance. Scalability is essential for ensuring that applications can meet growing demands without sacrificing performance or user experience.
- **Elasticity**: Elasticity builds upon scalability by introducing the concept of auto-scaling, where the system can automatically adjust its resources based on the current workload or demand. Once a system is scalable, elasticity allows it to dynamically scale resources up or down in response to fluctuations in demand. This is particularly advantageous in cloud environments, where resources can be provisioned and de-provisioned on-demand, leading to cost optimization and efficient resource utilization. Elasticity enables a "pay-per-use" model, where organizations only pay for the resources they consume, making it cloud-friendly and cost-effective.
- **Agility**: While not directly related to scalability, agility is another important concept in the context of cloud computing. Agility refers to the ability of an organization to rapidly provision and deploy IT resources as needed. With cloud services, new resources can be provisioned and made available to developers with just a few clicks, significantly reducing the time it takes to acquire and set up infrastructure. This rapid provisioning of resources enhances agility, enabling organizations to respond quickly to changing business requirements and market dynamics.
- In summary, while scalability focuses on the ability to accommodate increased loads through hardware improvements or additional nodes, elasticity extends this concept by introducing auto-scaling capabilities that dynamically adjust resources based on demand. Meanwhile, agility refers to the rapid provisioning and deployment of IT resources, which enhances an organization's ability to respond swiftly to changing needs and market conditions.

Scalability & Elasticity- Example

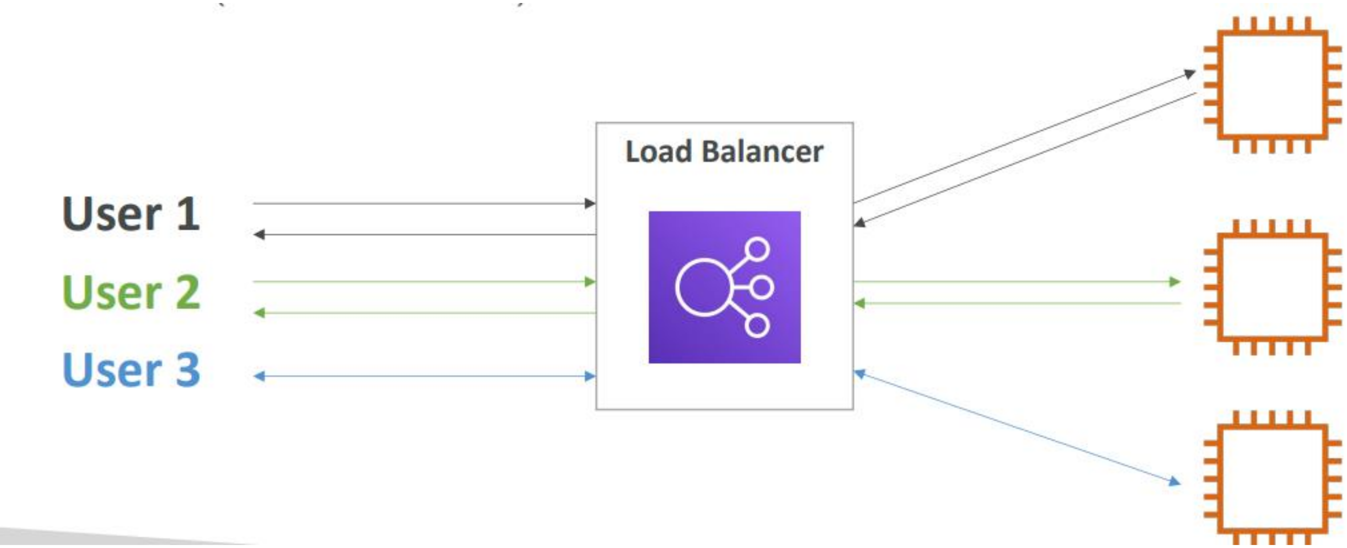
- **Scalability**: Imagine a restaurant that can accommodate more customers by either making its existing infrastructure stronger (scaling up) or by adding more tables and chairs (scaling out).
- *Scaling Up*: Increasing the restaurant's capacity by making its existing dining area larger, perhaps by knocking down a wall to create more space. This allows the restaurant to serve more customers at once without compromising on the quality of service.
- *Scaling Out*: Adding more tables and chairs to the restaurant's dining area or expanding to additional floors or locations. This allows the restaurant to handle larger crowds by spreading out the seating capacity across multiple areas.
- **Elasticity**: Building upon scalability, elasticity in the restaurant context would involve the ability to automatically adjust its capacity based on the current demand.
- For instance, during peak dining hours, such as lunch or dinner rush, the restaurant could open up additional sections or floors to accommodate more customers. Conversely, during slower periods, it could reduce the seating capacity by closing off certain areas to save on resources and costs.
- **Agility**: While scalability and elasticity focus on adjusting the restaurant's physical capacity, agility in this context would refer to the restaurant's ability to quickly adapt and respond to changing customer preferences or market conditions.
- For example, if there's suddenly a surge in demand for vegetarian options, the restaurant could swiftly introduce new vegetarian dishes to its menu to cater to this demand. Similarly, if there's a sudden change in weather leading to more outdoor dining requests, the restaurant could quickly set up additional outdoor seating areas to accommodate these preferences.
- In summary, just like a restaurant can scale its capacity by adjusting its physical infrastructure (scalability), implement auto-scaling measures to adapt to changing demand (elasticity), and respond rapidly to evolving customer needs (agility), organizations in the cloud computing realm can leverage these concepts to efficiently manage resources, optimize costs, and stay competitive in dynamic markets.

Scalability & Elasticity- Example

- Let's use the analogy of a water supply system to illustrate the concepts of scalability, elasticity, and agility:
- **Scalability**: Imagine a water supply system that needs to accommodate varying levels of demand throughout the day.
- *Scaling Up*: Increasing the capacity of the water treatment plant by upgrading its equipment or expanding its infrastructure. This allows the system to produce and distribute more water to meet increased demand during peak usage times, such as early morning or evening.
- *Scaling Out*: Adding additional pumping stations or reservoirs to the water supply network. By distributing the water supply infrastructure across multiple locations, the system can handle higher demand without overburdening any single component.
- **Elasticity**: Building upon scalability, elasticity in the water supply system context would involve the ability to adjust the flow of water dynamically based on current demand.
- For example, during periods of high demand, such as hot summer days, the system could increase the water flow rate from the treatment plant or pump additional water from reserve sources to meet the increased usage. Conversely, during periods of low demand, such as late at night, the system could reduce the flow rate to conserve resources.
- **Agility**: While scalability and elasticity focus on adjusting the physical infrastructure of the water supply system, agility in this context would refer to the system's ability to respond quickly to unexpected events or changes in demand.
- For instance, if there's a sudden pipe burst or equipment failure, the water supply system would need to quickly mobilize repair crews and reroute water flow to minimize disruptions to customers. Similarly, if there's a sudden surge in demand due to a public event or emergency, the system would need to swiftly adjust its operations to ensure an adequate supply of water to affected areas.

Load balancing

- Load balancer acting as an intermediary between users and servers, distributing internet traffic across multiple servers (in this case, EC2 instances).
- There are three users depicted, each interacting with the load balancer, which then intelligently forwards the requests to the backend servers, ensuring that no single server becomes overloaded.
- This helps in managing the network traffic efficiently and improves the overall performance and reliability of the application.



Reasons for using a load balancer

- ❑Distributing traffic across multiple downstream instances to prevent overloading of any single resource.
- ❑Offering a unified point of access for an application via DNS.
- ❑Handling failures of downstream instances without interrupting the service.
- ❑Performing regular health checks to ensure instances are functioning correctly.
- ❑Providing SSL termination, which means decrypting incoming HTTPS traffic at the load balancer before passing it to the backend servers.
- ❑Ensuring high availability and fault tolerance across different geographic zones.

Reasons for using a load balancer

- ❑Distributing traffic across multiple downstream instances to prevent overloading of any single resource.
- ❑Offering a unified point of access for an application via DNS.
- ❑Handling failures of downstream instances without interrupting the service.
- ❑Performing regular health checks to ensure instances are functioning correctly.
- ❑Providing SSL termination, which means decrypting incoming HTTPS traffic at the load balancer before passing it to the backend servers.
- ❑Ensuring high availability and fault tolerance across different geographic zones.

Reasons for using a load balancer-Example

- Imagine a call center that handles customer inquiries for a large company. The call center is analogous to a load balancer in the following ways:
- **Spread Load Across Multiple Agents:** Just as a load balancer distributes incoming network traffic across multiple servers, the call center directs incoming calls to various available customer service representatives. This ensures that no single agent is overwhelmed with too many calls, which is similar to preventing a server from being overloaded with too much traffic.
- **Single Point of Access:** Customers dial a single phone number to reach the call center, which is akin to accessing a service through a single DNS. The call center then routes the call to an appropriate agent, much like a load balancer routes network requests to the correct server.
- **Handle Failures Seamlessly:** If a customer service representative is unavailable or their line is busy, the call is automatically rerouted to another agent who is free. Similarly, if a server fails, the load balancer redirects traffic to another healthy server.
- **Health Checks:** Supervisors might monitor calls occasionally to ensure that all agents are performing well, much like load balancers perform health checks to verify that servers are up and running.
- **Providing Services:** While the load balancer offers SSL termination, the call center might offer language translation services, making it easier for customers to communicate in their preferred language.
- **High Availability Across Zones:** Call centers may have agents in multiple geographical locations to handle calls more reliably and quickly, even if one location is experiencing high volume or technical difficulties, ensuring that customer service is highly available, similar to load balancers distributing requests across multiple availability zones for reliability.
- In this way, the call center functions effectively and efficiently, providing a seamless experience to customers, much like a load balancer ensures the smooth operation of internet services.

Reasons for using a load balancer-Example

- Imagine a call center that handles customer inquiries for a large company. The call center is analogous to a load balancer in the following ways:
- **Spread Load Across Multiple Agents:** Just as a load balancer distributes incoming network traffic across multiple servers, the call center directs incoming calls to various available customer service representatives. This ensures that no single agent is overwhelmed with too many calls, which is similar to preventing a server from being overloaded with too much traffic.
- **Single Point of Access:** Customers dial a single phone number to reach the call center, which is akin to accessing a service through a single DNS. The call center then routes the call to an appropriate agent, much like a load balancer routes network requests to the correct server.
- **Handle Failures Seamlessly:** If a customer service representative is unavailable or their line is busy, the call is automatically rerouted to another agent who is free. Similarly, if a server fails, the load balancer redirects traffic to another healthy server.
- **Health Checks:** Supervisors might monitor calls occasionally to ensure that all agents are performing well, much like load balancers perform health checks to verify that servers are up and running.
- **Providing Services:** While the load balancer offers SSL termination, the call center might offer language translation services, making it easier for customers to communicate in their preferred language.
- **High Availability Across Zones:** Call centers may have agents in multiple geographical locations to handle calls more reliably and quickly, even if one location is experiencing high volume or technical difficulties, ensuring that customer service is highly available, similar to load balancers distributing requests across multiple availability zones for reliability.
- In this way, the call center functions effectively and efficiently, providing a seamless experience to customers, much like a load balancer ensures the smooth operation of internet services.

Elastic Load Balancer

- Elastic Load Balancer (ELB), which is a managed load balancer service provided by AWS. The key points about an ELB are:
- **Managed Service**: AWS is responsible for the operation, maintenance, and high availability of the ELB.
- **Upgrades and Maintenance**: AWS ensures that the ELB is always functioning correctly by handling all necessary upgrades and maintenance tasks.
- **Configuration**: The ELB comes with a few configuration options that are designed to manage application traffic effectively.
- **Cost-Efficient**: Using an ELB can be more cost-effective than setting up and maintaining a load balancer in-house because it reduces the effort and resources required for its operation.
- **Types of ELBs**: AWS provides various types of load balancers, including the Application Load Balancer for HTTP/HTTPS traffic, the Network Load Balancer for TCP traffic with ultra-high performance, and the Gateway Load Balancer. The Classic Load Balancer, which served both Layer 4 and Layer 7 traffic, was retired in 2023.
- The ELB acts as a traffic director, distributing incoming application traffic across multiple targets, such as EC2 instances, ensuring that no single server bears too much load. It provides a single point of access for the application, which simplifies the network architecture and helps in scaling the application seamlessly.

Elastic Load Balancer-Example

- An analogous example of using an Elastic Load Balancer (ELB) could be likened to a traffic management system in a busy city:
- **Managed Load Balancer**: The city's traffic department acts as a managed service, ensuring that traffic lights and signs are operational, similar to how AWS guarantees the working of an ELB.
- **Maintenance and Upgrades**: Just as the city maintains and upgrades traffic signals and signage, AWS takes care of upgrades and maintenance for the ELB.
- **Configuration Knobs**: Traffic lights have limited configuration options, much like ELB provides only a few configuration settings to manage traffic flow effectively.
- **Costs**: Establishing an independent traffic management system for a private community could be more costly and require more effort, paralleling the higher costs and effort of setting up a private load balancer versus using AWS's managed ELB.
- **Types of Traffic Routes**: Different roads (local streets, highways, express lanes) can be seen as different kinds of load balancers AWS offers. Local streets can be likened to Application Load Balancers, designed for specific types of traffic (HTTP/HTTPS). Highways could be compared to Network Load Balancers, handling a larger, more diverse set of traffic efficiently. Express lanes might be analogous to Gateway Load Balancers, providing fast and direct routes for specific traffic.
- **Classic Load Balancer**: Just like older roads that have been replaced by more efficient infrastructure, the Classic Load Balancer has been retired in favor of newer, more advanced options.
- In this analogy, the city's traffic management system ensures smooth and efficient traffic flow, just as an ELB ensures efficient distribution of network traffic to servers.

Different types of AWS Elastic Load Balancers (ELBs)

1. **Application Load Balancer (ALB)**:
    1. Operates at the application layer (Layer 7) of the OSI model.
    2. Supports HTTP/HTTPS and gRPC protocols.
    3. Offers advanced HTTP routing features.
    4. Utilizes a static DNS name for routing.
2. **Network Load Balancer (NLB)**:
    1. Functions at the transport layer (Layer 4).
    2. Handles TCP/UDP protocols.
    3. Designed for high-performance applications with the capability of handling millions of requests per second.
    4. Provides a static IP address through Elastic IP.
3. **Gateway Load Balancer (GWLB)**:
    1. Works at the network layer (Layer 3).
    2. Utilizes the GENEVE protocol on IP packets.
    3. Directs traffic to firewalls and intrusion detection services that are managed on EC2 instances.

- Each of these load balancers serves a different purpose, catering to specific needs in terms of performance, protocol support, and integration with other AWS services. The choice between them depends on the specific requirements of the application or service being hosted on AWS.

Auto Scaling Group (ASG)

- **Dynamic Scaling**: Reflecting real-life scenarios where website and application loads fluctuate, ASGs allow for the automatic adjustment of the number of EC2 instances. This dynamic scaling ensures that the infrastructure can adapt to the changing demand by either scaling out (adding instances) when the load increases, or scaling in (removing instances) when the load decreases.
- **Capacity Management**: ASGs maintain a defined minimum and maximum number of running instances to ensure there is enough capacity to handle the load without over-provisioning resources.
- **Integration with Load Balancers**: New instances can be automatically registered with load balancers to distribute incoming traffic evenly across all available instances.
- **Health Management**: ASGs continuously check the health of instances and automatically replace any that are deemed unhealthy.
- **Cost-Efficiency**: By adjusting the number of instances in response to actual demand, ASGs optimize costs as you only pay for the capacity you need.
- Overall, ASGs are a fundamental component of cloud resource management that helps maintain application availability and optimize operational costs.

Auto Scaling Group (ASG)- Example

- An analogous example of an Auto Scaling Group (ASG) could be compared to the staffing strategy of a retail store during a sales season:
- Imagine a store anticipating customer traffic during different times of the year, such as holidays or sales events. To prepare for this, the store manager implements a "staff scaling" strategy:
- **Scale Out**: During peak hours or a seasonal rush, the manager schedules more staff members to handle the increased number of customers, similar to adding more EC2 instances to handle increased web traffic.
- **Scale In**: On slow days or outside of peak hours, fewer staff members are scheduled to work, akin to removing EC2 instances when the demand is lower.
- **Capacity Management**: The manager always ensures that a minimum number of staff is present to keep the store operational and a maximum limit to avoid overstaffing, paralleling the minimum and maximum numbers of instances in an ASG.
- **Integration with Operations**: As new staff comes in for their shifts, they're quickly trained to handle the cash register or customer service desk, much like new instances are registered with a load balancer to handle web traffic.
- **Health Management**: If a staff member is not performing well (e.g., not processing sales quickly), the manager can replace them with someone more efficient, similar to how ASGs replace unhealthy instances.
- **Cost-Efficiency**: The manager's scheduling strategy ensures that the store is staffed appropriately to the customer traffic, optimizing operational costs by not having too many staff during slow periods or too few during busy times.
- This staffing strategy helps the store maintain excellent customer service during busy times and reduce labor costs when it's quieter, ensuring financial efficiency and customer satisfaction.

Auto Scaling Group (ASG)- Example

- The image you've provided is a diagram representing an Auto Scaling Group (ASG) in AWS (Amazon Web Services). An Auto Scaling Group is a collection of EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management.
- In the diagram, there are several key components displayed:
- **Minimum Size**: The lowest number of EC2 instances that the ASG will maintain.
- **Actual Size / Desired Capacity**: The current number of instances that are running. This may be higher than the minimum size depending on demand.
- **Maximum Size**: The maximum number of instances that the ASG is allowed to scale out to.
- **EC2 Instance**: These are the actual compute instances that are part of the ASG. They handle the application load.
- **Scale Out as Needed**: This dashed area indicates that new instances can be added to the ASG dynamically based on certain criteria like increased load.
- The ASG helps in ensuring that the number of Amazon EC2 instances adjusts automatically according to the load on the application, which provides better cost management and improved fault tolerance. This is crucial for applications with variable workloads and for maintaining application availability.