**EC2 Instance Storage:-**

1. **(51)Definition of EBS Volumes**:
   * **Elastic Block Store (EBS)**: A network-based storage solution that provides persistent block storage for EC2 instances. EBS volumes can be attached to instances while they are running and allow data to persist even after the instance is terminated.
2. **Key Features**:
   * **Persistence**: EBS volumes retain data after the associated EC2 instance is stopped or terminated, allowing users to recreate instances and reattach the same volume to access the data.
   * **Single Attachment**: An EBS volume can only be attached to one EC2 instance at a time, ensuring data integrity and performance.
   * **Availability Zone Bound**: EBS volumes are tied to a specific Availability Zone (AZ). For example, a volume created in **us-east-1a** cannot be attached to an instance in **us-east-1b** unless a snapshot is created and the volume is restored in the other AZ.
3. **Analogy**:
   * **Network USB Stick**: EBS volumes can be thought of as network-based USB drives that can be attached to EC2 instances over the network, allowing for easy data transfer and management.
4. **Free Tier**:
   * AWS provides 30 GB of free EBS storage (General Purpose SSD or Magnetic) per month for the first 12 months for new accounts.
5. **Performance and Provisioning**:
   * **Provisioning**: Users must specify the size (in GB) and IOPS (Input/Output Operations Per Second) when creating an EBS volume, which determines its performance characteristics.
   * **Billing**: Users are billed for the provisioned capacity, and they can increase the size or performance of the volume over time as needed.
6. **Volume Attachment**:
   * Multiple EBS volumes can be attached to a single EC2 instance (e.g., two network USB sticks), but each instance must have its own volume if it is to be attached.
7. **Unattached Volumes**:
   * EBS volumes can exist unattached to any EC2 instance, allowing for flexible storage management. They can be attached or detached on demand.
8. **Delete on Termination Attribute**:
   * **Definition**: This attribute controls whether an EBS volume is deleted when the associated EC2 instance is terminated.
   * **Default Behavior**:
     + The root EBS volume (the primary storage for the operating system) has the "Delete on Termination" option enabled by default, meaning it will be deleted when the instance is terminated.
     + Additional EBS volumes attached to the instance have this option disabled by default, meaning they will not be deleted when the instance is terminated.
   * **Use Case**: Users can disable the "Delete on Termination" option for the root volume if they want to preserve data after instance termination, which can be a relevant scenario in exam questions.
9. **About EBS Multi-Attach**
10. While I say that in the previous lecture that EBS volumes cannot be attached to multiple instances, I know it is not true for io1 and io2 volume types: this is called the EBS Multi-Attach feature.

**53)EBS HandsOn:-**

1. **Viewing EBS Volumes:**
   * Navigate to the EC2 instance and check the Storage tab to see attached EBS volumes.
   * Each instance has a root device and associated block devices (EBS volumes).
2. **Creating EBS Volumes:**
   * EBS volumes can be created from the Volumes interface in the AWS Management Console.
   * When creating a volume, you can choose the type (e.g., GP2, GP3) and specify the size (e.g., 2 GB).
   * EBS volumes must be created in the same Availability Zone (AZ) as the EC2 instance they will be attached to.
3. **Attaching EBS Volumes:**
   * After creating a volume, it can be attached to an EC2 instance.
   * You can attach multiple EBS volumes to a single instance, but each volume can only be attached to one instance at a time.
4. **Volume Availability:**
   * EBS volumes can exist unattached and can be attached or detached from instances as needed.
   * If a volume is created in a different AZ than the instance, it cannot be attached to that instance.
5. **Delete on Termination Attribute:**
   * The Delete on Termination attribute determines whether an EBS volume is deleted when the associated EC2 instance is terminated.
   * By default, the root volume has this attribute enabled (set to "yes"), while additional volumes have it disabled (set to "no").
   * Users can modify this setting during instance creation to preserve the root volume if desired.
6. **Terminating an Instance:**
   * When an EC2 instance is terminated, the root EBS volume (if set to delete on termination) will be deleted automatically.
   * After termination, the status of the volume will change to "available" if it was not set to delete.
7. **Cloud Flexibility:**
   * The ability to create, attach, detach, and delete EBS volumes quickly demonstrates the flexibility and scalability of cloud resources.

**54)EBS SnapShot:-**

**What are EBS Snapshots?**

* **Definition**: EBS Snapshots are backups of your EBS volumes. They capture the state of the volume at a specific point in time, allowing you to restore the volume later if needed.
* **Purpose**: Snapshots are useful for data backup, recovery, and transferring data across different Availability Zones (AZs) or regions in AWS.

**Key Features of EBS Snapshots**

1. **Creating Snapshots**:
   * You can create a snapshot of an EBS volume at any time without detaching the volume, although detaching it is recommended for a clean backup.
   * Snapshots can be created while the volume is in use, but stopping the EC2 instance beforehand ensures that all data is consistent.
2. **Restoring Snapshots**:
   * You can restore an EBS volume from a snapshot, even if the original volume has been deleted. This allows you to recover data that may have been lost.
3. **Copying Snapshots**:
   * Snapshots can be copied across different AZs or regions. This is useful for leveraging AWS's global infrastructure and ensuring data redundancy.
4. **Snapshot Archive**:
   * **Archive Tier**: You can move snapshots to a lower-cost storage tier called the archive tier, which is 75% cheaper than standard storage.
   * **Restoration Time**: Restoring from the archive tier takes longer (24 to 72 hours), making it suitable for snapshots that are not needed immediately.
5. **Recycle Bin for Snapshots**:
   * **Protection Against Accidental Deletion**: When you delete a snapshot, it can be sent to a recycle bin instead of being permanently deleted.
   * **Retention Period**: You can specify how long snapshots remain in the recycle bin (from one day to one year), allowing you to recover them if deleted accidentally.

**Additional Points**

* **Incremental Backups**: EBS snapshots are incremental, meaning that after the first snapshot, only the changes made since the last snapshot are saved. This saves storage space and reduces costs.
* **Cost Management**: Regularly review and manage your snapshots to avoid unnecessary costs. Use the archive feature for less critical snapshots to save money.
* **Automation**: Consider automating snapshot creation using AWS Lambda or AWS Backup to ensure regular backups without manual intervention.
* **Monitoring**: Keep track of your snapshots and their statuses through the AWS Management Console or AWS CLI to ensure that your backup strategy is effective.

**55)EBS HandsOn:-**

**Step-by-Step Guide to Managing EBS Snapshots**

**Step 1: Create a Snapshot**

1. **Select the EBS Volume**:
   * Go to the **EC2 Console** and select the **Volumes** section.
   * Choose the 2 GB GP2 EBS volume you want to back up.
2. **Create Snapshot**:
   * Click on **Actions** and select **Create Snapshot**.
   * Add a description (e.g., "DemoSnapshots") for the snapshot.
   * Click on **Create Snapshot** to initiate the process.

**Step 2: View Snapshots**

1. **Access Snapshots**:
   * In the left-hand menu, click on **Snapshots** to view all your snapshots.
   * You should see the newly created snapshot listed as **Completed** and **100% Available**.

**Step 3: Copy Snapshot to Another Region**

1. **Copy Snapshot**:
   * Right-click on the snapshot you want to copy.
   * Select **Copy Snapshot**.
   * Choose the destination region where you want to copy the snapshot (useful for disaster recovery).
   * Click **Copy** to initiate the copying process.

**Step 4: Create a Volume from Snapshot**

1. **Create Volume**:
   * Select the snapshot you want to use to create a new volume.
   * Click on **Actions** and select **Create Volume from Snapshot**.
   * Choose the size (e.g., 2 GB) and specify the target Availability Zone (e.g., change from **eu-west-1a** to **eu-west-1b** if desired).
   * Optionally, you can choose to encrypt the volume and add tags.
   * Click **Create Volume**.

**Step 5: Verify New Volume**

1. **Check Volumes**:
   * Go back to the **Volumes** section in the EC2 Console.
   * You should now see two volumes: the original and the new volume created from the snapshot.

**Step 6: Utilize the Recycle Bin**

1. **Create a Retention Rule**:
   * Navigate to the **Recycle Bin** feature in the EC2 Console.
   * Click on **Create Retention Rule**.
   * Name the rule (e.g., "DemoRetentionRule").
   * Select **EBS Snapshots** and apply it to all resources.
   * Set the retention period (e.g., 1 day) and leave the Rule Lock Setting unlocked.
   * Click **Create Retention Rule**.

**Step 7: Delete a Snapshot**

1. **Delete Snapshot**:
   * Go back to the **Snapshots** section.
   * Select the snapshot you want to delete.
   * Click on **Actions** and select **Delete Snapshot**.
   * Confirm the deletion.

**Step 8: Recover Snapshot from Recycle Bin**

1. **Check Recycle Bin**:
   * Refresh the Recycle Bin resources to see the deleted snapshot.
   * Select the snapshot from the Recycle Bin.
2. **Recover Snapshot**:
   * Click on **Recover Resources** to restore the snapshot back to the Snapshots section.
   * Verify that the snapshot is back in your Snapshots list.

**56)AMI OverView:-**

**Key Features of AMIs**

1. **Customization:** Users can create their own AMIs by customizing an EC2 instance with specific software configurations, monitoring tools, and settings. This allows for tailored environments that meet specific application needs**.**
2. **Faster Boot and Configuration Times:** By pre-packaging the desired software and configurations in an AMI, instances launched from that AMI can boot up faster and require less configuration time compared to setting up a new instance from scratch.
3. **Regional Availability**: AMIs can be created for a specific AWS region and can be copied to other regions, enabling users to leverage AWS's global infrastructure for deployment.
4. **Types of AMIs:**
   * **Public AMIs:** Provided by AWS, such as the popular Amazon Linux 2 AMI, which can be used by anyone.
   * **Custom AMIs:** Created by users to meet specific requirements.
   * **Marketplace AMIs:** Offered by third-party vendors in the AWS Marketplace, often with pre-installed software and configurations for specific use cases.

**Example Workflow**

* **Customization: Launch an EC2 instance in us-east-1A, customize it, and then stop it.**
* **Create AMI: Create a custom AMI from the stopped instance.**
* **Cross-Region Launch: Copy the AMI to us-east-1B and launch a new EC2 instance from that AMI, creating a duplicate of the original instance.**

**What are AMIs?**

* **Definition:** An Amazon Machine Image (AMI) is a pre-configured template that contains the operating system, application server, applications, and any custom configurations needed to launch an EC2 instance.

**Why Use AMIs?**

* **Efficiency:** AMIs allow for rapid deployment of EC2 instances with pre-configured settings, reducing setup time.
* **Consistency:** Ensures that all instances launched from the same AMI have identical configurations, which is crucial for scaling applications.
* **Backup and Recovery**: AMIs serve as backups of your instance configurations, allowing for quick recovery in case of failure.
* **Disaster Recovery:** Facilitates the creation of disaster recovery strategies by enabling the replication of instances across regions.
* **Cost-Effectiveness:** Reduces the need for maintaining multiple running instances by allowing users to create and launch instances as needed.

**When to Create AMIs?**

* **Before Major Changes:** Create an AMI before making significant changes to an instance (e.g., software updates, configuration changes) to ensure you can revert if needed.
* **For Scaling:** When you need to scale your application, create an AMI to quickly launch multiple instances with the same configuration.
* **For Testing:** Before testing new features or applications, create an AMI to safeguard the current state of your instance.
* **Regular Backups:** Implement a routine to create AMIs for critical instances to ensure you have up-to-date backups.

**Where to Use AMIs?**

* **AWS Regions:** AMIs can be created in one AWS region and copied to other regions, allowing for global deployment of applications.
* **AWS Marketplace:** Users can find and use public AMIs or third-party AMIs available in the AWS Marketplace for specific applications or configurations**.**
* **Custom Environments:** Use AMIs in development, testing, and production environments to maintain consistency across different stages of the application lifecycle.

**How to Create and Manage AMIs?**

1. **Launch an EC2 Instance:** Start by launching an EC2 instance using a public AMI or an existing custom AMI.
2. **Customize the Instance:** Install and configure the necessary software, settings, and applications on the instance.
3. **Stop the Instance:** To ensure data integrity, stop the instance before creating the AMI.
4. **Create the AMI:**
   * Go to the EC2 console, select the instance, and choose the option to create an AMI.
   * Provide a name and description for the AMI.
5. **Launch Instances from the AMI:** Use the created AMI to launch new EC2 instances as needed.
6. **Copy AMIs**: If required, copy the AMI to other regions for disaster recovery or scaling purposes**.**
7. **Manage AMIs:** Regularly review and manage your AMIs to delete outdated ones and optimize storage costs.

**Main use of creating AMI is we didn’t have to install HTTPD again**

**58)EC2 Image Builder:**-

Ec2 Image builder is a regional service that is it can only accessible in one region we canot spread to multiple automatically in multiple regions.

**Overview of EC2 Image Builder**

**EC2 Image Builder** is a service provided by AWS that automates the creation, maintenance, validation, and testing of Amazon Machine Images (AMIs) and container images. It simplifies the process of building and managing images for EC2 instances and containerized applications, ensuring that they are up-to-date, secure, and compliant with organizational standards.

**Key Features of EC2 Image Builder**

1. **Automation**: Automates the entire image creation process, including the installation of software, configuration, and updates.
2. **Validation**: Automatically creates test instances from the built AMIs to run predefined tests, ensuring that the images are functional and secure.
3. **Distribution**: Allows for the distribution of AMIs across multiple AWS regions, enabling global application deployment.
4. **Scheduling**: Supports scheduled image builds, allowing users to automate updates based on specific triggers (e.g., weekly schedules, package updates).
5. **Cost-Effective**: The service itself is free; users only pay for the underlying resources (EC2 instances, storage) used during the image creation process.

**Use Cases for EC2 Image Builder**

1. **Consistent Environment**: Ensures that all EC2 instances launched from the AMI have the same software and configuration, which is crucial for maintaining consistency across development, testing, and production environments.
2. **Rapid Deployment**: Facilitates quick deployment of new instances with the latest updates and configurations, reducing the time needed to set up new environments.
3. **Security Compliance**: Automates the process of applying security patches and updates, ensuring that images are compliant with security standards.
4. **Disaster Recovery**: Regularly updated AMIs can be used for disaster recovery, allowing for quick restoration of services in case of failures.

**How to Use EC2 Image Builder**

1. **Define Image Pipeline**:
   * Create an image pipeline that specifies the components, configurations, and tests to be included in the AMI.
   * Define the base image (e.g., an existing AMI) and the components (software packages, scripts) to be installed.
2. **Build and Test**:
   * EC2 Image Builder will automatically create a builder EC2 instance, install the specified components, and create the AMI.
   * Optionally, define tests to validate the AMI, such as checking application functionality and security compliance.
3. **Distribute AMI**:
   * Once validated, the AMI can be distributed to multiple regions for global deployment.
4. **Schedule Builds**:
   * Set up a schedule for regular image builds to ensure that the AMIs are always up-to-date with the latest software and security patches.

**59)EC2 Instance Store:-**

**Overview of EC2 Instance Store**

EC2 Instance Store is a type of temporary storage that is physically attached to the host server running your EC2 instance. It provides high I/O performance and is ideal for workloads that require fast access to data.

**Key Features**

1. **High Performance:**
   * **I/O Operations:** EC2 Instance Store offers significantly higher Input/Output Operations Per Second (IOPS) compared to Elastic Block Store (EBS). For example, certain instance types (like I3) can achieve millions of IOPS, making them suitable for high-performance applications.
   * **Low Latency:** Directly attached storage provides lower latency compared to network-attached storage solutions.
2. **Ephemeral Storage:**
   * **Temporary Nature:** Data stored in an EC2 Instance Store is ephemeral, meaning it is lost when the instance is stopped or terminated. This makes it unsuitable for long-term data storage.
   * **Volatile:** If the underlying hardware fails, the data stored in the instance store is also lost.
3. **Physical Attachment:**
   * **Direct Connection:** The storage is physically connected to the host server, providing better throughput and performance compared to network-based storage solutions.

**Use Cases**

1. **Temporary Data Storage:**
   * **Buffering and Caching:** Ideal for applications that require temporary storage for caching or buffering data, such as web servers or data processing applications.
   * **Scratch Data:** Suitable for intermediate data that does not need to be retained after processing, such as temporary files generated during computation.
2. **High-Performance Applications:**
   * **Data-Intensive Workloads:** Applications that require high IOPS and low latency, such as databases (e.g., NoSQL databases), big data processing, and analytics workloads.
3. **Testing and Development:**
   * **Development Environments:** Useful for development and testing environments where data persistence is not critical.

**Advantages**

* **Superior Performance: Provides** significantly higher IOPS and throughput compared to EBS volumes, making it suitable for performance-sensitive applications.
* **Cost-Effective for Temporary Storage:** Since it is included with the instance, there are no additional costs for storage, making it cost-effective for temporary data needs.

**Disadvantages**

* **Data Loss Risk:** Data is lost when the instance is stopped or terminated, making it unsuitable for long-term storage.
* **No Backup:** Users are responsible for managing backups and ensuring data replication, as there is no built-in redundancy for instance store data.
* **Limited Instance Types:** Not all EC2 instance types support instance stores; they are typically available on specific high-performance instance types.

**Performance Characteristics**

* **IOPS Comparison: For example, high-performance instances like I3 can achieve:**
  + **Read IOPS:** Up to 3.3 million IOPS**.**
  + **Write IOPS:** Up to 1.4 million IOPS.
* **EBS Comparison:** In contrast, EBS volumes (e.g., GP2) typically offer up to 32,000 IOPS, highlighting the performance advantage of instance stores for specific workloads.

**Key Features**

1. **Temporary Storage:**
   * Instance store volumes are ephemeral, meaning data is lost when the instance is stopped or terminated.
   * Suitable for temporary data that can be easily regenerated or replicated.
2. **High Performance:**
   * Offers high I/O performance due to direct physical attachment to the host server.
   * Ideal for applications requiring fast access to data, such as high-performance databases and data processing tasks.
3. **Volume Configuration:**
   * Instance store consists of one or more volumes exposed as block devices, with virtual device names ranging from ephemeral0 to ephemeral23.
   * The number and size of instance store volumes depend on the instance type.
4. **Instance Type Compatibility:**
   * Not all EC2 instance types support instance store volumes. For example, M6, C6, and R6 types do not support instance stores, while M5d, C6gd, and R6gd do.
   * Instance types with NVMe instance store volumes automatically attach all supported volumes at launch.
5. **Data Encryption:**
   * Data on NVMe instance store volumes is encrypted at rest using XTS-AES-256 block cipher.
   * Encryption keys are unique to each device and are destroyed when the instance is stopped or terminated.

**60)EFS Over View:-**

**EFS (Elastic File System) Overview:**

EFS is a managed **network file system** (NFS) that can be mounted to multiple **EC2 instances** across different Availability Zones (AZs) simultaneously, providing a **shared storage** solution. It is particularly suited for Linux EC2 instances and offers scalability, high availability, and flexibility in terms of storage usage.

Key Benefits:

1. **Shared Storage**: Multiple EC2 instances can access the same EFS volume at the same time.
2. **Cross-AZ Availability**: Can be mounted across multiple Availability Zones, allowing EC2 instances in different AZs to access the same storage.
3. **Scalable and Flexible**: The storage grows and shrinks as data is added or removed, and you only pay for the storage used.
4. **Cost-Effective**: EFS provides a **pay-per-use** model with automatic scaling, so you’re only billed for the data stored, avoiding upfront capacity planning.

**Cost Consideration**: EFS is generally **3x more expensive** than an **EBS gp2** volume. However, costs are dynamic, as you are billed based on your usage, and you don't have to over-provision capacity.

**EFS vs. EBS:**

* **EBS (Elastic Block Store)** is block-level storage, **tied to a single EC2 instance**, and is **zone-specific**, meaning it can only be attached to an EC2 instance in the same AZ.
  + If you need to move an EBS volume to another AZ, you would need to **create a snapshot** and **restore it** in the new AZ, but this is **not an in-sync replica**; it's just a copy.
* **EFS** is a **shared network file system** that allows multiple EC2 instances (across different AZs) to access the same storage at once, offering **real-time synchronization** between all attached instances.

**Storage Class: EFS-IA (Infrequent Access)**

EFS also offers **EFS-IA**, a **cost-optimized storage class** for data that is accessed less frequently. Files that haven't been accessed in a while (like 60 days) are automatically moved to **EFS-IA** to save on costs. Once a file is accessed again, it moves back to **EFS standard** storage. This is done through a **lifecycle policy**.

**Important Points:**

1. **High Availability**: EFS is highly available, designed for resilience, and automatically replicates data across multiple AZs.
2. **Scalability**: As demand grows, EFS expands its storage automatically without manual intervention.
3. **Ease of Use**: Mounting EFS on EC2 instances is straightforward, and the filesystem is managed by AWS, so you don’t need to worry about managing the infrastructure.
4. **Lifecycle Management**: EFS-IA allows cost savings for infrequently accessed data, while automatic data migration minimizes manual management of storage tiers.
5. **Linux EC2 Instances**: EFS currently only supports Linux-based EC2 instances.

**Drawbacks of EFS:**

1. **Cost**: EFS can be **expensive** compared to other storage solutions, such as EBS, especially for high-throughput use cases.
2. **Performance Limits**: EFS is not suitable for extremely high-performance use cases. It might be slower compared to EBS in terms of IOPS (input/output operations per second).
3. **Only Linux Support**: EFS works primarily with Linux-based EC2 instances. If you're using Windows-based EC2 instances, EFS is not an option.
4. **Complexity for Small Data**: For smaller, less complex applications, EBS might be a more cost-effective choice. EFS can be overkill when you don't need shared access to storage across instances.

**Drawbacks of EBS:**

1. **Single Instance Limit**: EBS volumes can only be attached to **one EC2 instance** at a time. If multiple instances need access, you'd need to replicate or share data manually, often using snapshots or other means.
2. **AZ Specific**: EBS volumes are **tied to a specific Availability Zone**, so moving the data to another AZ requires creating and restoring snapshots, which can be cumbersome and doesn't provide real-time replication.
3. **Limited to Block-Level Storage**: EBS is block-level storage, which means it isn't inherently shared across multiple instances like EFS. This can create complications for certain applications that need to access the same file system.

**Summary:**

* **EFS** is ideal for shared storage across multiple EC2 instances and offers **scalability** and **availability**, but it can be **expensive** for high-traffic scenarios.
* **EBS** is more suitable for **single-instance** storage needs or when low-latency block storage is necessary. However, it has **limitations** in terms of sharing data across instances and its **AZ-bound** nature.

Choosing between EFS and EBS depends on your application needs: whether you need shared, scalable storage with cross-AZ support (EFS) or high-performance, instance-bound storage (EBS).

**61)shared responsibility:-**

In the context of AWS and its services like **EC2** storage (including **EBS** and **EFS**), the **Shared Responsibility Model** defines which parts of the infrastructure and management are handled by AWS and which parts are the customer's responsibility. It is critical for certification exams, like the **Certified Cloud Practitioner** exam, to understand this distribution of responsibilities clearly.

**AWS's Responsibilities (Cloud Provider):**

1. **Infrastructure Maintenance**:
   * AWS is responsible for the **underlying infrastructure** that supports services like EBS and EFS, such as the physical hardware (servers, storage devices, network equipment, etc.), data center facilities, and networking.
2. **Data Replication and Redundancy**:
   * For services like **EBS (Elastic Block Store)** and **EFS (Elastic File System)**, AWS ensures that your data is **replicated across multiple hardware devices**. This means that if a part of the hardware fails, AWS handles the replication process and ensures there is no loss of data for you.
   * AWS takes responsibility for replicating data across **multiple availability zones (AZs)** to ensure **high availability** and **durability**.
3. **Hardware Failures**:
   * If **EBS** or **EFS** experiences hardware failure, AWS is responsible for **replacing faulty hardware** to minimize downtime or data loss. This is part of their responsibility for managing the physical infrastructure.
4. **Employee Data Security**:
   * AWS is responsible for ensuring that its **employees** do not have unauthorized access to your data. They manage security measures like access controls, monitoring, and logging to protect against internal breaches.

**Customer's Responsibilities (You, the User):**

1. **Backup and Snapshot Management**:
   * While AWS ensures that your data is replicated and available, **you** are responsible for setting up proper **backup** and **snapshot procedures**. For example, creating EBS snapshots or using tools like AWS Backup to protect against accidental deletion or data corruption.
   * It's important to configure **regular backups** to mitigate the risk of data loss in case of accidental deletion or unintentional application errors.
2. **Data Encryption**:
   * You are responsible for encrypting your data both in **transit** (when data is being transferred) and **at rest** (when data is stored on EBS or EFS).
   * AWS provides encryption features, but it's up to you to configure and manage encryption to protect your sensitive data. Even though AWS has security measures in place to prevent unauthorized access, **encryption adds an extra layer of protection** and ensures your data is secure from unauthorized access—even by AWS employees or other customers.
3. **Managing the Data Stored**:
   * Anything you write or store on your **EBS volume** or **EFS** is your responsibility. If your application writes data to these volumes, you need to manage and maintain that data, including keeping it secure, organized, and backed up.
4. **Data Loss from Instance Store**:
   * If you use an **EC2 instance store**, **data is ephemeral** and tied directly to the lifecycle of the instance. If the EC2 instance is stopped, terminated, or experiences hardware failure, any data on the instance store is **permanently lost**. You are responsible for backing up this data or moving it to a more persistent storage option, like EBS or EFS, to avoid loss.

**Summary:**

* **AWS's Responsibility**: Infrastructure, hardware replication, data availability, and employee data access control.
* **Your Responsibility**: Data management (including backups and encryption), ensuring data security, and managing the risk of data loss (e.g., for EC2 instance store).

**Example:**

* If your EC2 instance has an **EBS volume** attached and the instance crashes due to a hardware issue, AWS handles the **physical replacement** of the failed hardware and ensures that your data remains intact.
* However, **if your data is not backed up** and the instance crashes or you mistakenly delete a file, **you are responsible for restoring the data** from the backup you created. AWS won't automatically restore the data unless you have set up snapshots or backups.

This understanding of shared responsibility is vital for securing and managing your AWS resources properly, and you’ll need to know this for the exam as well as for practical use in a cloud environment.

**62)Amazon FSX:-**

**Amazon FSx Overview**

Amazon FSx is a **fully managed service** that provides high-performance file systems for specific workloads, allowing you to choose between several types of file systems. It's ideal if you need more advanced or specialized file systems compared to the basic **EFS** or **S3**.

Currently, AWS offers **three main FSx services**:

1. **FSx for Windows File Server**
2. **FSx for Lustre**
3. **FSx for NetApp ONTAP** (though this is not usually covered in exams)

For the **Certified Cloud Practitioner** exam, the key ones to focus on are **FSx for Windows File Server** and **FSx for Lustre**.

**FSx for Windows File Server:**

* **Target Use Case**: For **Windows-based applications** and **Windows EC2 instances** that need shared file storage.
* **Key Features**:
  + **Fully Managed** Windows-native shared file system.
  + **SMB Protocol** (Server Message Block) and **NTFS** (New Technology File System) support, which are standard Windows protocols.
  + Can be used with **Windows clients** and **Windows EC2 instances** in AWS.
  + Supports **Microsoft Active Directory** integration for user security and access management.
  + Can be deployed across **multiple availability zones** for high availability.
  + You can access it both **from AWS** and **on-premises infrastructure**.

**Use Case**: It's ideal for situations where you have applications that require **Windows file system compatibility**, such as legacy apps or environments running on Windows servers.

**FSx for Lustre:**

* **Target Use Case**: For **high-performance computing (HPC)** workloads, such as **machine learning**, **analytics**, **video processing**, and **financial modeling**.
* **Key Features**:
  + **High-performance file system** with **extremely low latency** and **high throughput** (up to hundreds of GB/s).
  + Designed for workloads that require **millions of IOPS (Input/Output Operations Per Second)** and **sub-millisecond latency**.
  + Can scale to handle very demanding workloads.
  + **Lustre** is a high-performance parallel file system commonly used in high-performance computing environments, particularly where large amounts of data need to be processed quickly.
  + It can be connected to **AWS EC2 instances** or your **on-premises data center**.
  + FSx for Lustre is often **backed by Amazon S3** to store data, enabling integration between **HPC workloads** and **S3**.

**Use Case**: Use this for workloads requiring very high throughput and performance, like big data processing, rendering, and simulations.

**Key Differences Between FSx for Windows File Server and FSx for Lustre:**

* **FSx for Windows File Server** is for **Windows environments**, providing shared file storage using Windows-native protocols like SMB and NTFS.
* **FSx for Lustre** is for **high-performance computing (HPC)**, providing a fast, scalable file system that works well for applications requiring very high throughput and low latency.

**In Summary:**

* **FSx for Windows File Server**: Ideal for Windows-based applications, file sharing, and Active Directory integration.
* **FSx for Lustre**: Best suited for high-performance computing tasks requiring fast data processing and low-latency access, like machine learning or video processing.

For the exam, **remember** these two types of FSx and their use cases—**Windows file server** for Windows-based applications and **Lustre** for high-performance computing.

**SMB (Server Message Block) and NTFS (New Technology File System) are two essential protocols used in Windows environments for file sharing and storage. Here's a breakdown of each, how they work, and why they're used in Amazon FSx for Windows File Server:**

**1. SMB (Server Message Block)**

* **What it is**:
  + **SMB** is a network protocol used by Windows-based systems to share files, printers, and other resources across a network. It allows applications to read and write to files, as well as request services from server programs in a computer network.
* **How it works**:
  + When you want to access a file or share resources like printers on a **Windows server**, SMB allows the client (a computer) to communicate with the server to request access to those resources.
  + For example, if a **Windows EC2 instance** in AWS needs to access a shared folder on another **Windows machine**, SMB allows it to do so using **network file shares**.
  + SMB operates on **port 445** and works by sending requests between clients and servers to open, close, read, and write files.
* **Why it's used in Amazon FSx for Windows File Server**:
  + **SMB** is a widely used protocol in **Windows environments** for **file sharing**. When Amazon FSx for Windows File Server is deployed, it uses SMB to allow your **Windows clients** (such as EC2 instances running Windows) to access shared files stored on the **FSx file system**.
  + It’s the native **file sharing protocol for Windows**, so it’s easy to integrate with existing **Windows-based applications** and services.
  + By supporting SMB, **Amazon FSx for Windows File Server** ensures that users can seamlessly access and share files across Windows machines, whether in **AWS** or **on-premises**.

**2. NTFS (New Technology File System)**

* **What it is**:
  + **NTFS** is a **file system** used by **Windows operating systems** to manage how data is stored and retrieved on a **disk**.
  + It supports advanced features like **file permissions**, **encryption**, **compression**, and **disk quotas**, making it a robust and secure file system.
* **How it works**:
  + NTFS organizes files into **directories** (folders) and tracks metadata about each file, such as **file size**, **permissions**, and **timestamps**.
  + When a file is created or modified, NTFS ensures that the data is securely stored and managed, maintaining the integrity of the file system.
* **Why it's used in Amazon FSx for Windows File Server**:
  + **NTFS** is the default file system for **Windows environments** and is designed to provide **high levels of security** and **reliability** for data storage.
  + In **FSx for Windows File Server**, NTFS ensures that your **files are organized**, **protected**, and **easily accessible** from Windows-based applications.
  + **NTFS permissions** enable fine-grained control over who can access and modify specific files or folders, which is critical for enforcing **security policies** in enterprise environments.
  + By using **NTFS**, **FSx for Windows File Server** provides **compatibility** with existing Windows applications and maintains the **security, performance, and scalability** needed for enterprise workloads.

**Why SMB and NTFS are used in Amazon FSx for Windows File Server:**

* **SMB** and **NTFS** are the **core building blocks** of Windows file sharing and storage. AWS uses them in **FSx for Windows File Server** to ensure compatibility with:
  + Existing **Windows-based applications**.
  + **On-premises infrastructure** (allowing hybrid cloud setups).
  + **Enterprise-level security** (via NTFS permissions and encryption).
* **Integration with Active Directory**:
  + **NTFS** works with **Windows Active Directory** to enforce security policies and user access control. This makes FSx for Windows File Server highly suitable for **enterprise environments** where managing access rights is critical.
* **Ease of Use**:
  + These are **standard protocols** that administrators and users are already familiar with. Using them in FSx for Windows File Server means you don’t need to modify existing applications or workflows that depend on SMB and NTFS.

**SUMMARY:-**

1. **EBS Volumes (Elastic Block Store):**
   * **What it is**: EBS volumes are network-attached storage devices that provide persistent storage for EC2 instances. Each EBS volume is attached to one EC2 instance at a time and is mapped to a specific Availability Zone (AZ).
   * **Key Feature**: EBS is tied to the **Availability Zone** where the EC2 instance is running, meaning the volume can only be attached to EC2 instances in that same AZ.
   * **Snapshot Feature**: EBS allows you to take **snapshots** of volumes, which are point-in-time backups of the data stored on the volume. You can use snapshots to:
     + Backup data.
     + Move data across Availability Zones by restoring the snapshot to a new volume in another AZ.
     + Create a new EC2 instance from an AMI (Amazon Machine Image) that includes the snapshot data.
2. **AMIs (Amazon Machine Images):**
   * **What it is**: AMIs are pre-configured templates used to launch EC2 instances. An AMI can include the operating system, application server, applications, and configuration settings.
   * **Customization**: You can create **custom AMIs** by modifying an EC2 instance with your desired configuration, and then creating an AMI from that instance.
   * **Automation with EC2 Image Builder**: **EC2 Image Builder** is a service that automates the creation, testing, and distribution of AMIs. It helps streamline the process of building customized AMIs, ensuring consistency and reliability when deploying EC2 instances.
3. **EC2 Instance Store:**
   * **What it is**: EC2 instance store provides high-performance storage that is directly attached to the physical server hosting the EC2 instance.
   * **Key Feature**: Unlike **EBS volumes**, instance store is **ephemeral** (temporary). This means that if the EC2 instance is stopped, terminated, or fails, the data on the instance store is **lost**.
   * **Use Case**: Instance stores are used for **temporary data** or **high-performance workloads** that can tolerate data loss in case of instance termination.
4. **EFS (Elastic File System):**
   * **What it is**: EFS is a **fully managed, scalable network file system** that can be mounted on **multiple EC2 instances** at the same time. It provides shared access to files, allowing several EC2 instances to read and write data simultaneously.
   * **Key Feature**: Unlike EBS, which is bound to a specific instance, EFS is not limited by the **Availability Zone**. It can span across multiple AZs within a region, enabling high availability and scalability.
   * **Cost Optimization (EFS-IA)**: **EFS-IA** (EFS Infrequent Access) is a lower-cost storage tier for files that are **rarely accessed**. Files that have not been accessed for a certain period are automatically moved to the EFS-IA tier, reducing storage costs without affecting performance for frequently accessed files.
5. **FSx for Windows File Server:**
   * **What it is**: **FSx for Windows File Server** is a managed **Windows-native file system** that uses **SMB (Server Message Block)** and **NTFS (New Technology File System)** protocols, making it ideal for Windows-based applications and EC2 instances.
   * **Key Feature**: It provides a fully managed file system that integrates easily with **Windows-based applications**, supports **Active Directory** for user access management, and allows deployment across multiple AZs for high availability.
   * **Use Case**: It’s perfect for scenarios where you need a shared file system in a **Windows environment**, such as legacy applications or Windows server-based applications.
6. **FSx for Lustre:**
   * **What it is**: **FSx for Lustre** is a managed file system optimized for **high-performance computing (HPC)** workloads. It is built on **Lustre**, a high-speed, low-latency file system used in environments that need to process large amounts of data quickly.
   * **Key Feature**: It provides **extremely high throughput** and **low latency**, making it suitable for **machine learning**, **big data analytics**, **video processing**, and **financial modeling**. It can scale to handle **massive data sets** with millions of input/output operations per second (IOPS).
   * **Integration with S3**: FSx for Lustre can also integrate with **Amazon S3**, allowing you to use the Lustre file system for HPC workloads while storing data in S3 for durability and cost savings.
   * **Use Case**: FSx for Lustre is ideal for applications that require high-performance file storage with massive scalability, such as **simulations**, **rendering**, and **scientific computations**.

**Summary of Key Differences:**

* **EBS**: Attached to **one EC2 instance**, and tied to a specific **Availability Zone**. Used for persistent storage and supports **snapshots** for backup and data migration.
* **AMIs**: Pre-configured images used to launch EC2 instances with custom configurations, and can be automatically created using **EC2 Image Builder**.
* **EC2 Instance Store**: High-performance **temporary storage** that is lost when the EC2 instance stops or terminates.
* **EFS**: Shared file storage that supports **multiple EC2 instances** across **multiple Availability Zones** within a region. Ideal for file-sharing workloads.
* **FSx for Windows File Server**: Managed **Windows file system** that integrates with **Windows-based applications** and supports **SMB** and **NTFS** protocols.
* **FSx for Lustre**: High-performance file system optimized for **high-performance computing** (HPC) workloads, such as machine learning and data analytics, with **low-latency** and **high-throughput** performance.