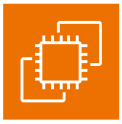


Amazon Elastic Compute Cloud (Amazon EC2)

Basic Terms and Definitions that we should know before we step into the AWS EC2 Learning

Instance: A virtual server in the AWS cloud. Instances are launched from Amazon Machine Images (AMIs) and come in various types, each with different performance characteristics.

Amazon Machine Image (AMI): A pre-configured template for launching EC2 instances. AMIs include the OS, application server, applications, and related configurations.

Instance Type: The configuration of compute, memory, and storage resources for an EC2 instance. Examples include t2.micro, m5.large, etc.

Virtual Private Cloud (VPC): A virtual network dedicated to your AWS account. It allows you to define a custom network topology, including IP address range, subnets, and route tables.

Subnet: A range of IP addresses within your VPC. Subnets help to organize instances based on their security and operational needs.

Security Group: A virtual firewall that controls inbound and outbound traffic to your instances. Security groups allow you to specify rules based on IP protocol, port number, and source/destination IP addresses.

Key Pair: A pair of cryptographic keys used for secure login to your instances. One key is public and stored by AWS, and the other is private and stored by you.

Elastic Block Store (EBS): A service that provides persistent block storage for EC2 instances. EBS volumes can be attached to instances and used as storage devices.

Elastic Load Balancer (ELB): A service that automatically distributes incoming traffic across multiple EC2 instances to ensure high availability and fault tolerance.

Auto Scaling: A service that automatically adjusts the number of EC2 instances in your application based on current demand. It helps maintain performance and minimize costs.

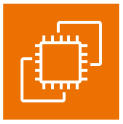
AMI Lifecycle: Understanding how to create, manage, and deregister AMIs for consistent deployments and rollbacks.

Elastic IP Address: A static, public IP address designed for dynamic cloud computing. It can be associated with an EC2 instance to ensure that it retains a fixed IP address.

Instance Store: Temporary storage provided with some instance types. Data in instance store volumes is lost when the instance is stopped or terminated.

Pricing Models: AWS EC2 offers several pricing options, including On-Demand, Reserved Instances, and Spot Instances. Understanding these can help optimize costs.

Regions and Availability Zones: AWS regions are geographical areas with multiple isolated locations known as Availability Zones. Each zone is designed to be isolated from failures in other zones.



There are the categories of EC2 learning – we are going to learn in deep ...!!

- EC2 Instance Types & Classes – Purchase Options.
- Launching and Configuring EC2 Instances
- EC2 Networking, Elastic Load Balancing (ELB) & Auto Scaling
- EC2 Placement Groups
- EC2 Storage Options & Architecture Options

1. EC2 Instance Types & Classes – Purchase Options.

In the context of Amazon EC2, "instance types" and "classes" are related but not exactly the same.

Instance Types

- An **instance type** defines the **hardware configuration** of an EC2 instance, including CPU, memory, storage, and networking capacity.
- **Examples**: (Orange colour highlighted wording)
  - **t3.micro**
  - **m5.large**
  - **c5.xlarge**
- **Purpose**: Selecting an instance type allows you to tailor the compute resources to match your application's requirements.

What the Micro/Large/Xlarge means

- "Micro": Smallest instance size, suitable for lightweight applications.
- "Large": Mid-range instances with balanced performance.
- "Xlarge" and beyond: Instances with significantly more compute power and memory, used for high-performance applications.

Each size scales the vCPU, memory, and sometimes network bandwidth in proportion to the increasing size label. **The specific number of resources depends on the instance family (e.g., t3, m5, r5, etc.).**

Instance Families (Classes)

Instance families (sometimes informally referred to as "classes") categorize **instance types** based on their optimized use cases and performance characteristics.

Common Families:

General Purpose (e.g., t3, m5) – TM

- Great for a diversity of workloads such as web servers or code repositories
- Balance between compute, memory & networking
- **T** family: Burstable performance (e.g., t4g, t3, t3a)
- **M** family: Balanced performance (e.g., m6g, m5, m5a, m5n)

Compute Optimized (e.g., c5) – C

- Great for compute intensive tasks like Batch Processing & Gaming Servers..etc
- **C** family: Compute-optimized (e.g., c7g, c6g, c5, c5a, c6in)

Memory Optimized (e.g., r5) - RXU

- Great for **in-memory databases** or **distributed web caches**
- **R** family: Memory-optimized (e.g., r6g, r5, r5b)
- **X** family: Extreme memory-optimized (e.g., x2gd, x1, x1e)
- **u-** family: High memory instances (e.g., u-6tb1, u-12tb1

Storage Optimized (e.g., i3) - ID

- Great for storage intensive tasks (accessing local databases) like OLTP systems & Distributed File System (DFS)..etc
- **I** family: High I/O (SSD storage) optimized (e.g., i4g, i3, i3en)
- **D** family: Dense HDD storage optimized (e.g., d2, d3, d3en)

GPU Instances (e.g., g4, p3) – PGF

- Great for AI & ML processing engines, AR & VR engines, etc
- **P** family: GPU instances for general-purpose compute (e.g., p4, p3)
- **G** family: GPU instances for graphics (e.g., g5, g4ad, g4dn)
- **F** family: FPGA instances for customizable hardware acceleration (e.g., f1)

Purpose: Instance families help you quickly identify which instance types are best suited for specific workloads, such as high-performance computing, large databases, or machine learning tasks.

How They Relate to each other (Instance Types & Families)

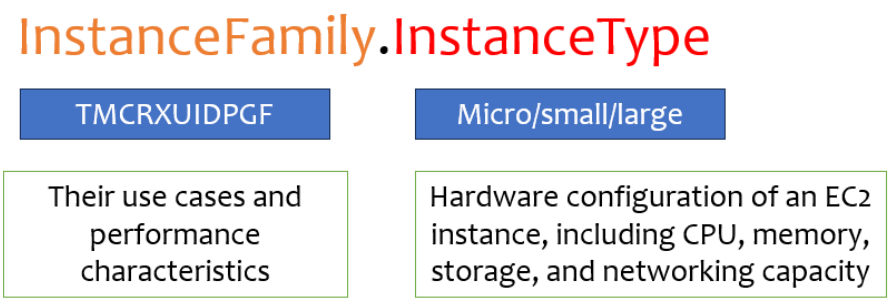
- **Hierarchy: Instance families** are broader categories that group together **instance types** sharing similar characteristics.

Example: Within the **Compute Optimized** family (**c5**), you have different **instance types** like **c5.large**, **c5.xlarge**, **c5.2xlarge**, etc., each offering varying levels of CPU power and memory.

Selection Process:

1. **Determine Workload Needs**: Identify whether your application requires general-purpose computing, high CPU, large memory, etc.
2. **Choose an Instance Family**: Based on the workload, select the appropriate instance family.
3. **Select an Instance Type**: Within that family, choose the specific instance type that best matches your performance and cost requirements.

Simple depiction would look like this



Now you know – "How to choose an EC2 Instance or Server"

Now we will look at the purchase options or prising models

Purchasing options are related to how you buy and use EC2 instances. AWS offers several purchasing options for EC2 instances to fit different workloads and cost optimization needs. These options give you flexibility in managing costs and usage.

On-Demand Instances

- Pay for compute capacity by the hour or second (depending on the instance type) with no long-term commitments.
- Pay per use (no upfront payment)
- Highest cost
- No long-term commitment
- Recommended for short-term, uninterrupted and unpredictable workloads
- **Use Case**: Suitable for short-term, spiky, or unpredictable workloads that cannot be interrupted.

Reserved Instances (RIs)

- Commit to using an instance for a 1-year or 3-year term in exchange for a significant discount on the hourly rate.
- **Types of Reserved Instances**:

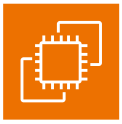
Standard RIs: Provide the most significant discount, but less flexibility.

- Reservation Period: 1 year or 3 years
- Recommended for steady-state applications (like database)
- Sell unused instances on the Reserved Instance Marketplace

Convertible RIs: Can be exchanged for other instance types within the same family, offering more flexibility.

- Can change the instance type
- Lower discount
- **Cannot sell unused instances** on the Reserved Instance Marketplace

Use Case: Ideal for steady-state workloads with predictable usage.



Savings Plans

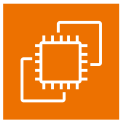
- A flexible pricing model offering lower prices in exchange for a commitment to a consistent usage (measured in $/hour) for 1 or 3 years. This can apply across various instance types and regions.
- **Use Case**: Similar to Reserved Instances, but with more flexibility in instance types and usage.

Spot Instances

- Purchase unused EC2 capacity at discounts of up to 90% compared to On-Demand prices. However, instances can be interrupted when AWS needs the capacity back.
- Work on a bidding basis where you are willing to pay a specific max hourly rate for the instance. Your instance will get interrupted if the spot price increases your bidding price.
- The behaviour for spot instance interruption can be stop, hibernate or terminate. It cannot be reboot.
- Spot blocks are designed not to be interrupted
- **Use Case**: Suitable for flexible, stateless, or fault-tolerant applications like batch jobs or big data processing.

Dedicated Hosts

- Physical servers fully dedicated to your use, allowing you to use your own software licenses and ensure compliance with licensing regulations.
- 3-year reservation period
- Billed per host (expensive)
- Useful for software that have BYOL (Bring Your Own License) or for companies that have strong regulatory or compliance needs
- **Use Case**: Ideal for highly regulated industries or specific licensing needs.



Dedicated Instances

- Instances run on hardware that is physically dedicated to a single customer, but without the flexibility of managing host-level attributes.
- Instances running on a dedicated hardware for an AWS account.
- Both dedicated and non-dedicated instances of an account can be running on the dedicated hardware
- Billed per instance
- No control over instance placement
- Cheaper than Dedicated Host
- **Use Case:** Suitable for workloads where you need hardware isolation for compliance or security reasons.

Instances States

Pending:

- The instance is in the process of being launched.
- What's happening: AWS is preparing the instance for use, provisioning the required resources.
- Duration: Typically lasts a few seconds to a couple of minutes.

Running:

- The instance is operational and ready for use.
- What's happening: The instance is fully launched, and you can connect to it, run applications, etc.
- Billing starts when an instance reaches the running state.

Stopping:

- The instance is in the process of shutting down.
- What's happening: AWS is executing the shutdown scripts and stopping the instance.
- Duration: Typically takes a few seconds to minutes, depending on the workload.

Stopped:

- The instance is stopped, and it's not running.
- What's happening: The instance is fully shut down, and all associated resources (such as CPU, memory) are released. You can restart the instance later.
- Billing stops in this state, but you are still charged for EBS volumes attached to the instance.

Shutting-down:

- The instance is in the process of being terminated.
- What's happening: AWS is executing the termination process, and the instance is being fully deleted.
- Duration: Similar to stopping, but the instance will be permanently removed after this state.

Terminated:

- The instance has been permanently deleted.
- What's happening: The instance is no longer available, and all resources (like IP addresses, EBS volumes, etc.) are released unless explicitly retained (e.g., EBS volumes with deleteOnTermination = false).
- Billing stops permanently for this instance.

Rebooting:

- The instance is in the process of restarting.
- What's happening: The instance reboots but does not lose any data stored on the attached volumes or the instance store (for some instance types). AWS performs a soft reboot.
- Billing continues during reboot.

2. EC2 Launching & configuring

In this category – we need all the things that we are going to learn anyway, that includes instance types, classes, networking, security … etc

But we will learn the things which are very specific and plays a role in the EC2 raising, that being said, our first topic would be

AMI - Amazon Machine Images

Amazon Machine Images (AMIs) are pre-configured templates used to create and launch EC2 instances. An AMI includes everything needed to launch an instance, such as:

- **Operating System (OS)**: Linux, Windows, or a custom OS.
- **Application Server**: Software needed for your application, like Apache or Nginx.
- **Applications**: Pre-installed applications or configurations.

Intro Parameters

- AMIs are the image of the instance after installing all the necessary OS, software and configuring everything.
- It boots much faster because the whole thing is pre-packaged and doesn't have to be installed separately for each instance.
- Good for static configurations
- Bound to a region (can be copied across regions)

Types of AMIs

1. **Public AMIs**
   - **Description**: Available to all AWS users. These AMIs are provided by AWS or third parties and are often pre-configured with popular software or OS.
   - **Use Case**: Quick setup for standard configurations or popular software. Suitable for general-purpose applications or experimenting with new software.
2. **Private AMIs**
   - **Description**: Created and owned by your AWS account. These AMIs are private to your account and are not shared with other AWS users.
   - **Use Case**: Customized configurations tailored to your specific needs. Ideal for replicating an application setup across multiple instances or environments.

3. **Marketplace AMIs**

- **Description:** Available through the AWS Marketplace. These AMIs are created by third-party vendors and often include commercial software or pre-configured applications.
- **Use Case:** Deploying complex software stacks or enterprise applications without manual setup. Suitable for applications requiring licensing or special configurations

4. **Community AMIs**

- **Description:** Shared by other AWS users in the community. These AMIs are publicly available and often used for specific use cases or configurations.
- **Use Case:** Experimenting with community-contributed software or setups. Useful for finding AMIs that fit niche needs or unique configurations.

Creating and Managing AMIs

1. **Creating an AMI**
   - **From an Existing Instance:** You can create an AMI from an existing EC2 instance, capturing its configuration, OS, and data.
   - **Process:** In the EC2 Management Console, select the instance, choose "Create Image" from the Actions menu, and follow the steps to create an AMI.
2. **Managing AMIs**
   - **Lifecycle:** Regularly update your AMIs to include the latest patches and configurations. Remove outdated or unused AMIs to manage storage costs and maintain security.
3. **Using AMIs**
   - **Launching:** When launching a new EC2 instance, select the desired AMI from your list of available AMIs, or search for public or marketplace AMIs that fit your needs.

Second topic which would be inline is Security Groups and Firewalls

Security Groups and Firewalls

EC2 Security Groups and firewalls play a crucial role in securing EC2 instances by controlling inbound and outbound traffic.

Security Groups

Security Groups are virtual firewalls associated with EC2 instances. They control the traffic allowed to and from instances based on rules you define. Each security group acts as a set of rules that determine whether traffic is allowed or denied based on IP addresses, port numbers, and protocols.

Types of Rules in Security Groups // Firewall Rules of EC2

1. **Inbound Rules**:
    - **Definition**: Specify the allowed traffic coming into the instance.
    - **Configuration**: Define rules based on protocol (e.g., TCP, UDP), port range (e.g., 22 for SSH, 80 for HTTP), and source IP addresses or CIDR blocks.
2. **Outbound Rules**:
    - **Definition**: Specify the allowed traffic leaving the instance.
    - **Configuration**: Define rules based on protocol, port range, and destination IP addresses or CIDR blocks.

Types of Security Groups

1. **Default Security Group**:
    - **Description**: Automatically created for each VPC. By default, it allows all outbound traffic and allows inbound traffic from other instances within the same security group.
    - **Use Case**: Basic setup for new instances. Modify or add additional security groups based on specific requirements.
2. **Custom Security Groups**:
    - **Description**: Created by users to define specific security rules for instances.
    - **Use Case**: Tailored security policies for different types of applications or services.

3. EC2 Networking

EC2 Networking refers to the various networking components and configurations necessary to manage and secure network traffic for Amazon EC2 instances. Proper networking configuration ensures that instances can communicate with each other, with the internet, and with other AWS resources, while also maintaining security and performance.

Why we need this networking aspect as well in EC2?

03 Major reasons

Connectivity:

- **Instance Communication:** Allows instances to communicate with each other and other services within AWS or on-premises networks.
- **Internet Access:** Provides instances with access to the internet or restricts it based on security requirements.

Security:

- **Isolation:** Configures network isolation for instances to control which instances can communicate with each other and with external resources.
- **Access Control:** Manages access using security groups, network ACLs, and route tables to enforce security policies.

Performance:

- **Network Latency:** Ensures optimal network performance by configuring appropriate network settings.
- **Bandwidth:** Manages network bandwidth and ensures efficient data transfer between instances and services.

Components and Configurations

1. **Virtual Private Cloud (VPC)**
    - A virtual network dedicated to your AWS account where you can launch AWS resources.
    - **Configuration:**
    - **Subnets:** Define IP address ranges for different segments of your network (e.g., public and private subnets).
    - **Route Tables:** Manage traffic routing within your VPC and to/from the internet or other networks.
    - **Internet Gateway:** Allows communication between instances in a VPC and the internet.
    - **NAT Gateway:** Provides internet access to instances in private subnets while keeping them secure from inbound internet traffic.
2. **Security Groups**
    - Virtual firewalls that control inbound and outbound traffic to EC2 instances.
    - **Configuration:** Define rules to allow or deny traffic based on IP addresses, ports, and protocols.
3. **Network Access Control Lists (NACLs)**
    - Provide an additional layer of security at the subnet level by controlling inbound and outbound traffic to/from subnets.
    - **Configuration:** Define rules to allow or deny traffic, similar to security groups but applied at the subnet level.
4. **Elastic IP Addresses (EIP)**
    - Static, public IP addresses that can be associated with EC2 instances.
    - **Configuration:** Allocate and associate EIPs to instances to ensure consistent public IP addresses.
5. **Private IP Addresses**
    - Internal IP addresses used for communication within a VPC.
    - **Configuration:** Assigned automatically when launching instances within a VPC.

6. **Network Interfaces**
    - Virtual network interfaces that can be attached to EC2 instances.
    - **Configuration:** Manage primary and secondary network interfaces to control traffic routing and network access.
7. **DNS and Hostnames**
    - Manage domain name resolution within your VPC.
    - **Configuration:** Use Route 53 for DNS services and enable DNS resolution within the VPC.
8. **VPC Peering**
    - Connects two VPCs to enable traffic between them.
    - **Configuration:** Create a peering connection and update route tables to allow traffic to flow between peered VPCs.
9. **Transit Gateway**
    - Central hub for connecting multiple VPCs and on-premises networks.
    - **Configuration:** Create and manage attachments to connect VPCs and external networks.
10. **Direct Connect**
    - Provides dedicated network connections between on-premises data centers and AWS.
    - **Configuration:** Set up Direct Connect connections and virtual interfaces to integrate with your VPC.

Types of Networking Configurations

1. **Public and Private Subnets:**
    - **Public Subnets:** Accessible from the internet (e.g., for web servers).
    - **Private Subnets:** Not directly accessible from the internet (e.g., for databases or application servers).
2. **High Availability and Fault Tolerance:**
    - **Multiple Availability Zones (AZs):** Deploy resources across multiple AZs for high availability.
    - **Auto Scaling:** Automatically adjust the number of instances based on demand.
3. **Network Security:**
    - **Security Groups:** Manage traffic to instances.
    - **NACLs:** Manage traffic to and from subnets.
4. **Network Performance:**
    - **Enhanced Networking:** Use Elastic Network Adapters (ENAs) or Intel 82599 VF interfaces for improved network performance.
    - **Placement Groups:** Control the placement of instances to optimize network performance.
5. **Hybrid Cloud:**
    - **VPN Connections:** Connect on-premises networks to your VPC over an IPsec VPN connection.
    - **Direct Connect:** Establish dedicated connections between your on-premises infrastructure and AWS.

Keep the below concept as well with you.

Elastic Network Interface (ENI)

- ENI is a virtual network card that **gives a private IP to an EC2 instance**
- A primary ENI is created and attached to the instance upon creation and will be deleted automatically upon instance termination.
- We can create additional ENIs and attach them to an EC2 instance to access it via multiple private IPs.
- We can detach & attach ENIs across instances
- **ENIs are tied to the subnet** (and hence to the AZ)

EC2 Elastic Load Balancing (ELB) is a service that automatically distributes incoming application or network traffic across multiple EC2 instances to ensure high availability, fault tolerance, and scalability. ELB helps manage and balance traffic loads, enhancing the reliability and performance of your applications.

All other things go back to your ELB Service Doc and concept

EC2 Auto Scaling is a service that automatically adjusts the number of EC2 instances in a group based on demand. This helps ensure that you have the right number of instances running to handle the current workload while optimizing cost and maintaining performance.

All other things go back to your ASG Service Doc and concept

4. EC2 Placement Groups

EC2 Placement Groups are a way to control the placement of your EC2 instances on underlying hardware to optimize performance, reduce latency, or meet specific application requirements. Placement groups influence the distribution of instances across hardware to achieve desired performance characteristics and avoid potential issues related to hardware failures or network latency.

Why We Need Placement Groups

1. **Performance Optimization**:
   - **Network Latency**: Minimize latency by placing instances close to each other.
   - **Bandwidth**: Increase network bandwidth by ensuring instances are on the same physical hardware.
2. **Fault Tolerance**:
   - **Avoid Hardware Failures**: Spread instances across different hardware to reduce the impact of potential hardware failures.
3. **High Throughput**:
   - **Enhanced Networking**: Utilize placement groups to take advantage of high-bandwidth, low-latency networking between instances.
4. **Application Requirements**:
   - **Specialized Needs**: Some applications may have specific requirements for instance placement, such as low-latency communication or high network throughput.

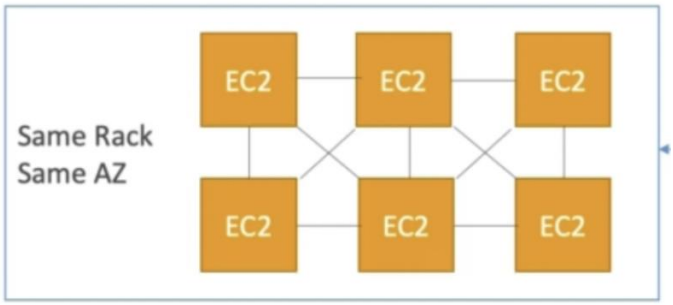
Types of Placement Groups

1. **Cluster Placement Group**

- Packs instances into a single Availability Zone to ensure that they are physically close to each other. This arrangement is designed to provide high network throughput and low latency.
- **Use Case:** Ideal for applications that require high-performance computing (HPC) or other applications that benefit from high bandwidth and low latency, such as real-time analytics or financial transactions.
- **Limitations:** Instances within a cluster placement group may be subject to capacity limits, and you may need to ensure that the instances can be placed together.

Cluster Placement Group (optimize for network)

- All the instances are placed on the same hardware (same rack)
- Pros: Great network (10 Gbps bandwidth between instances)
- Cons: If the rack fails, all instances will fail at the same time
- Used in **HPC** (minimize inter-node latency & maximize throughput)

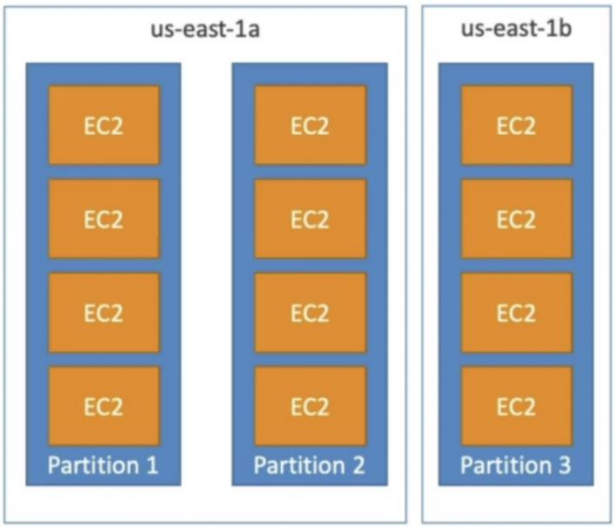


2. Partition Placement Group

- Distributes instances across multiple partitions, where each partition is isolated from others. This setup helps ensure that instances in the same partition are not affected by issues in other partitions.
- **Use Case:** Suitable for distributed and replicated applications such as HDFS (Hadoop Distributed File System) or Cassandra. It helps maintain high availability and fault tolerance by reducing the risk of simultaneous failures.
- **Limitations:** Instances within the same partition group are still subject to availability zone limits, and you must consider partition constraints when designing your application.

Partition Placement Group (balance of performance and availability)

- Instances in a partition share rack with each other
- If the rack goes down, the entire partition goes down
- Up to 7 partitions per AZ
- Used in big data applications (Hadoop, HDFS, HBase, Cassandra, Kafka)

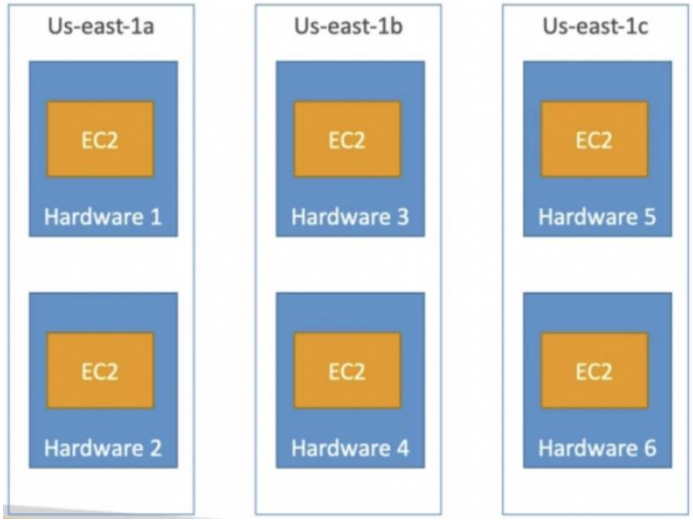


3. Spread Placement Group

- Distributes instances across distinct hardware to minimize the risk of simultaneous failures. Each instance is placed on separate hardware, providing maximum fault tolerance.
- **Use Case**: Ideal for applications that require high availability and fault tolerance, such as applications with critical components that need to avoid co-location risks. Commonly used for smaller-scale, high-availability applications.
- **Limitations**: Limited to a maximum of seven running instances per placement group in a single Availability Zone. It may not be suitable for large-scale applications requiring a large number of instances.

Spread Placement Group (maximize availability)

- Each instance is in a separate rack (physical hardware)
- Supports Multi AZ
- Up to 7 instances per AZ per placement group (ex. for 15 instances, need 3 AZ)
- Used for critical applications



5. EC2 Storage Options

EC2 storage is essential for managing data and applications in AWS. Different types of storage are optimized for various use cases, such as high-performance computing, data durability, or cost-effectiveness.

Types of EC2 Storage

1. **Amazon Elastic Block Store (EBS):**
   - **Description:** Provides block-level storage that persists beyond the lifecycle of an EC2 instance. Useful for data that requires frequent updates and transactional consistency.
   - **Types:**
     - **General Purpose SSD (gp3):** Balanced price/performance for most workloads.
     - **Provisioned IOPS SSD (io2/io2 Block Express):** High-performance SSD for I/O-intensive applications.
     - **Throughput Optimized HDD (st1):** High-throughput storage for frequently accessed, large, sequential data.
     - **Cold HDD (sc1):** Lower-cost HDD for infrequently accessed data.
2. **Instance Store:**
   - **Description:** Provides temporary storage that is physically attached to the host machine. Data in instance store is lost if the instance is stopped or terminated.
   - **Use Case:** Suitable for temporary data like cache, scratch data, or buffers.
3. **Amazon Elastic File System (EFS):**
   - **Description:** Provides a scalable, fully managed NFS (Network File System) file storage for use with EC2 instances. It supports multiple instances accessing the file system concurrently.
   - **Use Case:** Shared file storage for applications that need access to the same data.

4. **Amazon FSx**:
   - **Description**: Fully managed Windows file system and Lustre file system solutions that are optimized for different use cases.
     - **Amazon FSx for Windows File Server**: Windows-based file storage.
     - **Amazon FSx for Lustre**: High-performance file system for compute-intensive workloads.
   - **Use Case**: Windows applications needing a file system, or high-performance computing with Lustre.
5. **Amazon S3**:
   - **Description**: Scalable object storage for storing and retrieving any amount of data. While not directly attached to EC2 instances, it's often used for backup, archiving, and large data storage.
   - **Use Case**: Backup, archival, and storing large amounts of unstructured data.
6. **Amazon S3 Glacier**:
   - **Description**: Low-cost storage service for data archiving and long-term backup. Data retrieval can be slower compared to S3 standard storage.
   - **Use Case**: Long-term storage and backup of data that is infrequently accessed.

Choosing the Right Storage Type

- **For High-Performance Needs:** Use EBS (io2) or instance store for low-latency, high IOPS requirements.
- **For Shared Storage:** Use EFS for applications needing access from multiple instances.
- **For Cost-Efficient Storage:** Use S3 or S3 Glacier for backups, archiving, or infrequent access needs.

Considerations

- **Performance Requirements:** Different storage types offer varying levels of performance. Choose based on the performance needs of your application.
- **Durability and Availability:** EBS, S3, and EFS offer high durability and availability, but instance store does not.
- **Cost:** Factor in the cost of each storage type and its usage patterns to optimize expenses.

EC2 Nitro & EC2 Classic

EC2 Nitro and **EC2 Classic** are two different underlying architectures for Amazon EC2 instances that impact their performance, security, and functionality.

EC2 Nitro is a modern virtualization architecture introduced by AWS that powers many newer EC2 instance types. It is designed to provide enhanced performance, security, and efficiency compared to previous generations.

EC2 Classic refers to the original virtualization architecture used in the earlier days of Amazon EC2. It has since been largely superseded by newer technologies like EC2 Nitro.

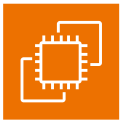
Just keep this one as well in the brain,

AWS Graviton refers Instances powered by AWS Graviton processors, which are based on ARM architecture. Examples include A1 and M6g, C6g, and R6g instance families.

Use Case: Ideal for applications that can benefit from ARM architecture's performance and cost advantages, such as web servers, containerized microservices, and distributed databases.

Key Differences Between EC2 Nitro and EC2 Classic

1. **Virtualization:**
   - **Nitro:** Uses dedicated hardware for virtualization tasks, resulting in better performance and lower overhead.
   - **Classic:** Uses traditional virtualization methods, which can introduce more overhead and lower performance.
2. **Security:**
   - **Nitro:** Includes advanced security features such as hardware-accelerated encryption and secure boot.
   - **Classic:** Security features are less advanced compared to Nitro.
3. **Performance:**
   - **Nitro:** Offers higher performance and lower latency due to offloading virtualization tasks.
   - **Classic:** Performance may be lower due to traditional virtualization overhead.
4. **Instance Types:**
   - **Nitro:** Supports a wide range of modern instance types with enhanced capabilities.
   - **Classic:** Includes older instance types with less advanced features.
5. **Networking:**
   - **Nitro:** Works with VPC for advanced networking and security.
   - **Classic:** May not fully support VPC networking features.



EC2 Scenario based questions

https://lisireddy.medium.com/aws-ec2-scenario-based-questions-9b441db29da0

Wish you the best …! Happy Learning ..!

Yours' Love (@lisireddy across all the platforms)