**EC2:**

An EC2 instance is a virtual server in AWS that provides scalable computing capacity to run applications and services. It allows you to choose different configurations of CPU, memory, storage, and networking, and you only pay for the resources you use. EC2 instances are used to host websites, applications, and workloads in the cloud.

34)Aws Budget Setup:-if cannot access the billing information then admistrator in account must activate it.

* Budget: it will make alarm when it was exceeded the budget

35) Ec2 Instance means:it is a virtual machine and it is a platiform as a service but it is not a service but it is high level

* It capable of
* Renting Virtual machines
* ASG,ELB,EBS or EFS

Sizing and configuration:

* **How much cpu power or EBS storage Which os we choose like linux,windows,MACOS etc security groups ip address ec2 user data etc**

EC2 USER DATA:- Boot strap our instances using EC2 data script

* Bootstrapping means Launching commands
* Runs only first time you start instances
* Like renting means installing necessary software downloading etc

37)**EC2 instance types:-**

General purpose, compute optamized,Accelerated computing…..

**General purpose**: diversity of workloads such as web servers etc..ONDEMAND

Balance between compute, memory and networking

**Compute optimized:** high performance processors or webservers, dedicated web servers etc

**Memory Optimized:-**fast performance for large data sets and For RAM or Data Bases

**Storage Optimized:great** storage that require sequential write large datasets stored in local storage

38) **Security groups:-**

The main difference between **VPC (Virtual Private Cloud)** and **Security Group** in AWS is:

1. **VPC**: A VPC is a virtual network that allows you to create and manage your own isolated environment within AWS. It controls the IP address range, subnets, routing, and network gateways, providing a secure and customizable networking environment for your resources.
2. **Security Group**: A Security Group is a virtual firewall that controls inbound and outbound traffic to your AWS resources, like EC2 instances. It operates at the instance level and allows you to define rules based on IP addresses, ports, and protocols to secure your resources.

In short, **VPC** is for network isolation and management, while **Security Group** is for controlling access and traffic to resources within that network.

**Overview of AWS Security Groups**

**Security Groups** act as virtual firewalls for your Amazon EC2 instances, controlling inbound and outbound traffic. They are fundamental to network security in the AWS cloud and are easy to configure, as they only contain allow rules.

**Key Features of Security Groups**

1. **Inbound and Outbound Rules**:
   * **Inbound Rules**: Define what traffic is allowed to reach your EC2 instances from the outside. You can specify rules based on IP address ranges (IPv4 or IPv6) and protocols (e.g., TCP, UDP).
   * **Outbound Rules**: Define what traffic is allowed to leave your EC2 instances. By default, all outbound traffic is allowed.
2. **Rule Structure**:
   * Each rule consists of:
     + **Type**: The type of traffic (e.g., SSH, HTTP).
     + **Protocol**: The protocol used (e.g., TCP).
     + **Port Range**: The specific port or range of ports (e.g., 22 for SSH).
     + **Source/Destination**: The IP address or CIDR block that is allowed (e.g., **0.0.0.0/0** for all IPs).
3. **Default Behavior**:
   * By default, all inbound traffic is blocked, and all outbound traffic is allowed. This means you must explicitly allow any inbound traffic you want to permit.
4. **Multiple Instances and Groups**:
   * A single security group can be attached to multiple EC2 instances, and an instance can belong to multiple security groups. This flexibility allows for easier management of access rules.
5. **Region and VPC Specific**:
   * Security groups are tied to a specific region and Virtual Private Cloud (VPC). If you switch regions or create a new VPC, you will need to create new security groups.
6. **Security Group References**:
   * Security groups can reference other security groups. This allows for easier management of permissions, especially in complex architectures with multiple instances. For example, if you have a web server and a database server, you can allow the web server's security group to access the database server's security group without needing to specify IP addresses.

**Common Ports to Remember**

* **22**: SSH (Secure Shell) - Used for logging into Linux instances.
* **21**: FTP (File Transfer Protocol) - Used for transferring files.
* **22**: SFTP (Secure File Transfer Protocol) - Also uses port 22 for secure file transfers.
* **80**: HTTP (Hypertext Transfer Protocol) - Used for unsecured web traffic.
* **443**: HTTPS (Hypertext Transfer Protocol Secure) - Used for secured web traffic.
* **3389**: RDP (Remote Desktop Protocol) - Used for logging into Windows instances.

**Additional Points to Consider**

1. **Logging and Monitoring**:
   * Consider enabling AWS CloudTrail and VPC Flow Logs to monitor and log traffic to and from your EC2 instances. This can help you identify unauthorized access attempts and troubleshoot connectivity issues.
2. **Least Privilege Principle**:
   * Always follow the principle of least privilege when configuring security groups. Only allow the minimum necessary access required for your application to function. For example, if your application only needs to accept traffic from a specific IP address, specify that IP address instead of allowing traffic from **0.0.0.0/0**.
3. **Testing Connectivity**:
   * If you encounter connectivity issues (e.g., timeouts), check your security group rules first. If you receive a "connection refused" error, it may indicate that the application is not running or is misconfigured, rather than a security group issue.
4. **Separation of Concerns**:
   * It’s a good practice to create separate security groups for different types of access (e.g., one for SSH access, another for web traffic). This makes it easier to manage and audit access rules.
5. **Integration with Load Balancers**:
   * When using load balancers, you can configure security groups to allow traffic from the load balancer to your EC2 instances. This is particularly useful for scaling applications and managing traffic efficiently.
6. **39)Security Groups Handson:-** 
   * ou encounter a timeout when trying to connect to your EC2 instance (e.g., when accessing a website), it usually indicates a security group issue. This means the necessary inbound rule is missing.
   * If you receive a connection refused error, it means the security group allowed the traffic, but the application on the instance is not responding.
7. **Modifying Rules:**
   * You can easily add or remove rules in the security group settings. For example, if you remove the HTTP rule and try to access your web server, you will get a timeout.
   * To fix this, you can add the HTTP rule back, allowing traffic on port 80.
8. **CIDR Blocks:**
   * You can specify the source of the traffic using CIDR blocks. For example, 0.0.0.0/0 allows traffic from anywhere, while a specific IP address (e.g., 192.168.1.1/32) allows traffic only from that IP.
   * You can also use the "My IP" option to allow access only from your current IP address. However, if your IP changes, you will need to update the security group to maintain access.
9. **Multiple Security Groups:**
   * An EC2 instance can be associated with multiple security groups. This means you can combine rules from different security groups to create a comprehensive security policy.
   * Similarly, a single security group can be attached to multiple EC2 instances, allowing for consistent security settings across instances.

**Additional Points for Hands-On Practice or Interviews**

1. **Common Ports to Remember:**
   * 22: SSH (for Linux instances)
   * 80: HTTP (for unsecured web traffic)
   * 443: HTTPS (for secured web traffic)
   * 3389: RDP (for Windows instances)
2. **Best Practices:**
   * Least Privilege: Only allow the minimum necessary access. For example, if your application only needs to be accessed from a specific IP, restrict access to that IP.
   * Separate Security Groups: Consider creating separate security groups for different types of access (e.g., one for SSH, another for web traffic) to simplify management and auditing.
3. **Troubleshooting:**
   * If you cannot connect to your instance, always check the security group rules first. Look for missing inbound rules that would allow the required traffic.
   * Use tools like telnet or curl to test connectivity to specific ports on your EC2 instance.

**How Security Groups Work**

1. **Definition**:
   * Security Groups are virtual firewalls that control inbound and outbound traffic to AWS resources, primarily Amazon EC2 instances. They are part of the AWS security model and are used to define rules that allow or deny traffic.
2. **Rules**:
   * **Inbound Rules**: Specify what traffic is allowed to enter an EC2 instance. Each rule consists of:
     + **Type**: The type of traffic (e.g., SSH, HTTP).
     + **Protocol**: The protocol used (e.g., TCP, UDP).
     + **Port Range**: The specific port or range of ports (e.g., 22 for SSH).
     + **Source**: The IP address or CIDR block that is allowed to access the instance (e.g., **0.0.0.0/0** for all IPs).
   * **Outbound Rules**: Specify what traffic is allowed to leave an EC2 instance. By default, all outbound traffic is allowed.
3. **Default Behavior**:
   * By default, all inbound traffic is blocked, and all outbound traffic is allowed. This means you must explicitly allow any inbound traffic you want to permit.
4. **Stateful Nature**:
   * Security Groups are stateful, meaning that if you allow an incoming request from a specific IP address, the response traffic is automatically allowed, regardless of outbound rules.
5. **Multiple Associations**:
   * A single security group can be associated with multiple EC2 instances, and an instance can belong to multiple security groups. This allows for flexible and scalable security configurations.

**Difference Between Security Groups and Network Access Control Lists (NACLs)**

1. **Security Groups** are virtual firewalls that control inbound and outbound traffic to AWS resources, primarily EC2 instances. They are stateful and easy to manage, allowing you to define rules that specify what traffic is allowed.
2. **Network Access Control Lists (NACLs)** operate at the subnet level and provide an additional layer of security. They are stateless and allow for both "allow" and "deny" rules, making them suitable for controlling traffic across multiple resources within a subnet.

**40)SSH Summary Table:-**

* Command line interface utility can be used for windows and linux

**What is SSH?**

* **SSH (Secure Shell)** is a protocol that allows you to securely access and control a remote machine or server over a network. It is commonly used to manage servers and perform administrative tasks.

**Steps to SSH into an EC2 Instance**

1. **Prepare the PEM File**:
   * When you launch an EC2 instance, you typically download a private key file (with a **.pem** extension) that is used for authentication. In this case, the file is named **EC2Tutorial.pem**.
   * Ensure that the file name does not contain spaces, as this can cause issues when trying to use it.
2. **Store the PEM File**:
   * Place the **.pem** file in a directory on your local machine. For example, you might create a folder called **aws-course** and move the file there.
3. **Check Security Group Settings**:
   * Before connecting, ensure that the security group associated with your EC2 instance allows inbound traffic on port 22 (the default port for SSH). This is typically set to allow traffic from anywhere (**0.0.0.0/0**), but for better security, you may want to restrict it to specific IP addresses.
4. **Open a Terminal**:
   * Open a terminal on your Linux or Mac computer. You will use this terminal to issue commands to connect to your EC2 instance.
5. **Get the Public IP Address**:
   * In the AWS Management Console, navigate to your EC2 instance and copy its public IPv4 address. This address is needed to connect to the instance.
6. **SSH Command**:
   * The basic command to connect to your EC2 instance is:

RunCopy code

1ssh ec2-user@<public-ip>

* + Here, **ec2-user** is the default username for Amazon Linux 2 instances, and **<public-ip>** is the public IP address of your instance.

1. **Authentication Failure**:
   * If you attempt to connect without specifying the key file, you may receive an authentication failure error. This is because the SSH command needs to know which private key to use for authentication.
2. **Specify the Key File**:
   * To specify the key file, use the **-i** option in the SSH command:

RunCopy code

1ssh -i EC2Tutorial.pem ec2-user@<public-ip>

* + Ensure you are in the correct directory where the **.pem** file is located.

1. **Set Permissions on the Key File**:
   * If you encounter an error about unprotected key files, you need to change the permissions of the **.pem** file to ensure it is not publicly viewable. This is done using the **chmod** command:

RunCopy code

1chmod 400 EC2Tutorial.pem

* + This command sets the file permissions so that only the owner can read the file.

1. **Successful Connection**:
   * After running the SSH command with the correct key file and permissions, you should be logged into your EC2 instance. You will see a prompt indicating that you are now operating within the instance.

**45)EC2 Instance connect:-**

**Key Points for Hands-On Use**

1. **Accessing EC2 Instance Connect**:
   * Navigate to the EC2 Management Console.
   * Select the instance you want to connect to (e.g., "My First Instance").
   * Click on the **Connect** button at the top of the instance details page.
2. **Connection Options**:
   * You will see multiple connection options, including:
     + **SSH Client**: Traditional SSH connection using a terminal.
     + **EC2 Instance Connect**: The browser-based option you will use.
3. **Username**:
   * The default username for Amazon Linux 2 instances is **ec2-user**. This is automatically populated, but you can override it if needed (though it typically should remain as **ec2-user**).
4. **No SSH Key Management**:
   * Unlike traditional SSH connections, you do not need to manage SSH keys. EC2 Instance Connect uploads a temporary SSH key for you when you connect, simplifying the process.
5. **Connecting**:
   * Click on **Connect** to initiate the connection. A new browser tab will open, providing you with a terminal interface directly into your EC2 instance.
6. **Running Commands**:
   * Once connected, you can run commands directly in the browser-based terminal, such as:
     + **whoami**: To check the current user.
     + **ping google.com**: To test internet connectivity.
7. **Security Group Configuration**:
   * Ensure that the security group associated with your EC2 instance allows inbound traffic on port 22 (SSH). If this rule is missing, you will not be able to connect.
   * If you remove the SSH rule and attempt to connect, you will receive an error indicating that the connection cannot be established.
8. **IPv4 and IPv6 Considerations**:
   * If you are using IPv6, ensure that your security group also allows inbound traffic on port 22 for IPv6 addresses. This may be necessary depending on your network setup.
9. **Troubleshooting**:
   * If you encounter issues connecting, check the following:
     + Ensure that port 22 is open in the security group.
     + Verify that you are using the correct username (**ec2-user**).
     + Check for any network restrictions that may be affecting connectivity.

**1. Understanding EC2 Instance Connect**

* **Definition**: EC2 Instance Connect is a feature that allows users to connect to their Amazon EC2 instances using a browser-based SSH session. It simplifies the SSH connection process by eliminating the need for users to manage SSH keys manually.
* **How It Works**: When you initiate a connection using EC2 Instance Connect, AWS temporarily uploads an SSH key to the instance, allowing you to establish a secure connection without needing to handle key files on your local machine.

**2. Security Implications**

* **Temporary SSH Keys**: EC2 Instance Connect uses temporary SSH keys that are generated and uploaded for each session. This reduces the risk associated with long-term key management, as the keys are not stored on the instance permanently.
* **Security Group Rules**: Proper security group configurations are essential for EC2 Instance Connect to function. The security group must allow inbound traffic on port 22 (SSH) from the appropriate IP addresses. Without these rules, connections will fail.
* **Access Control**: Since EC2 Instance Connect allows connections from the AWS Management Console, it’s important to ensure that only authorized users have access to the console to prevent unauthorized access to instances.

**3. Use Cases**

* **User -Friendly Access**: EC2 Instance Connect is particularly useful for users who may not be familiar with SSH key management or command-line interfaces. It provides a straightforward way to access instances directly from the browser.
* **Quick Access**: It is ideal for scenarios where quick access to an instance is needed without the overhead of managing SSH keys. This can be beneficial for temporary administrative tasks or troubleshooting.
* **Training and Education**: In educational environments, EC2 Instance Connect can simplify the process for students learning about AWS and Linux, allowing them to focus on learning rather than on SSH key management.

**4. Comparison with Traditional SSH**

* **Ease of Use**: Unlike traditional SSH, which requires users to generate, store, and manage SSH keys, EC2 Instance Connect streamlines the process by handling key management automatically.
* **No Local Key Management**: Users do not need to worry about the security of private keys on their local machines, as the keys are temporary and managed by AWS.
* **Browser-Based Interface**: EC2 Instance Connect provides a browser-based terminal, making it accessible from any device with internet access, whereas traditional SSH requires a terminal application.

**5. Configuration Requirements**

* **Supported Operating Systems**: EC2 Instance Connect is supported on Amazon Linux 2 and Ubuntu instances. Ensure that the instance is running one of these operating systems to use this feature.
* **Security Group Settings**: The security group associated with the instance must allow inbound SSH traffic (port 22) from the IP addresses that will be used to connect. This is crucial for establishing a connection.
* **IAM Permissions**: Users must have the necessary IAM permissions to use EC2 Instance Connect. This includes permissions to connect to the instance and to use the EC2 Instance Connect feature.

**6. Limitations**

* **Public Subnet Requirement**: EC2 Instance Connect requires the instance to be in a public subnet with a public IP address. If the instance is in a private subnet, you will not be able to connect using this method.
* **Public IP Address**: The instance must have a public IP address assigned to it. If the instance does not have a public IP, you will need to use other methods (like a bastion host) to connect.
* **Session Duration**: The temporary SSH keys used by EC2 Instance Connect are valid only for the duration of the session. Once the session ends, the keys are removed, which means you cannot reconnect without going through the process again.

**46)Instance Roles Demo:-**

**Hands-On Steps to Use IAM Roles with EC2 Instances**

1. **Connect to Your EC2 Instance**:
   * Use **EC2 Instance Connect** or SSH to connect to your EC2 instance. For example, if using EC2 Instance Connect, navigate to your instance in the AWS Management Console and click on **Connect**.
2. **Check AWS CLI Installation**:
   * Once connected, verify that the AWS Command Line Interface (CLI) is installed by running:

bash

RunCopy code

1aws --version

1. **Attempt to List IAM Users**:
   * Run the command to list IAM users:

bash

RunCopy code

1aws iam list-users

* + You will likely receive an error indicating that credentials are not configured.

1. **Avoid Using aws configure**:
   * Do **not** run **aws configure** to set up your credentials. This is insecure because it would store your AWS Access Key ID and Secret Access Key on the instance, which could be accessed by anyone with access to the instance.
2. **Create an IAM Role**:
   * In the AWS Management Console, navigate to **IAM** and create an IAM role with the necessary permissions (e.g., **IAMReadOnlyAccess**).
   * Ensure that the role is set to be assumed by EC2 instances.
3. **Attach the IAM Role to Your EC2 Instance**:
   * Go to the **EC2 Management Console**.
   * Select your instance, then click on **Actions** > **Security** > **Modify IAM Role**.
   * Choose the IAM role you created (e.g., **DemoRoleForEC2**) and click **Save**.
4. **Verify IAM Role Attachment**:
   * Check the instance details to confirm that the IAM role is now attached.
5. **Run IAM Commands Again**:
   * Now, run the command to list IAM users again:

bash

RunCopy code

1aws iam list-users

* + This time, you should receive a successful response with the list of IAM users.

1. **Detach the IAM Role**:
   * To demonstrate the role's effect, detach the IAM role from the instance and run the command again. You should receive an "Access Denied" error, confirming that the role is necessary for access.
2. **Reattach the IAM Role**:
   * If you reattach the IAM role and run the command again, you may need to wait a moment for the changes to propagate, but eventually, you should see the expected output.

**Interview Point Of View**

**1. Understanding IAM Roles**

* **Definition**: IAM (Identity and Access Management) roles are a way to grant permissions to AWS services and resources without needing to create long-term credentials (like access keys). Roles are designed to be assumed by AWS services, such as EC2 instances, Lambda functions, or ECS tasks.
* **Difference from IAM Users**:
  + **IAM Users**: These are permanent identities with long-term credentials (access keys) that are associated with a specific person or application. Users can log in to the AWS Management Console or make API calls using their credentials.
  + **IAM Roles**: Roles do not have permanent credentials. Instead, they provide temporary security credentials that are automatically rotated and managed by AWS. Roles can be assumed by AWS services or by IAM users, allowing for flexible permission management.

**2. Security Best Practices**

* **Avoid Hardcoding Credentials**: Using IAM roles eliminates the need to hardcode AWS credentials (Access Key ID and Secret Access Key) in your applications or on EC2 instances. Hardcoding credentials poses a significant security risk, as they can be exposed if the instance is compromised or if the code is shared.
* **Enhanced Security**: By using IAM roles, you reduce the risk of credential exposure. Temporary credentials provided by roles are automatically rotated, which further enhances security.

**3. Role Attachment**

* **Attaching IAM Roles**: To attach an IAM role to an EC2 instance, you can do the following:
  1. Create an IAM role in the AWS Management Console with the necessary permissions.
  2. Navigate to the EC2 Management Console, select the instance, and choose **Actions** > **Security** > **Modify IAM Role**.
  3. Select the desired IAM role and save the changes.
* **Inheritance of Permissions**: Once the role is attached, the EC2 instance inherits the permissions defined in the role. This means that any AWS service calls made from the instance will have the permissions granted by the role.

**4. Temporary Credentials**

* **Automatic Provisioning**: When an IAM role is attached to an EC2 instance, AWS automatically provides temporary security credentials to the instance. These credentials are valid for a limited duration (typically a few hours).
* **Automatic Rotation**: AWS manages the rotation of these temporary credentials, ensuring that they are refreshed automatically without any action required from the user.

**5. Use Cases**

* **Accessing AWS Services**: IAM roles are particularly useful in scenarios where an EC2 instance needs to access other AWS services without managing access keys. Examples include:
  + **S3 Access**: An EC2 instance can use an IAM role to read from or write to an S3 bucket without needing to store S3 access keys on the instance.
  + **DynamoDB Access**: An instance can access DynamoDB tables to read or write data securely.
  + **Other Services**: Roles can be used to grant access to services like RDS, SNS, SQS, and more, allowing for seamless integration between services.

**6. IAM Policies**

* **Policy Attachment**: IAM policies define the permissions granted to a role. When you create a role, you can attach one or more policies that specify what actions are allowed or denied.
* **Principle of Least Privilege**: When creating IAM policies, it’s essential to follow the principle of least privilege, which means granting only the permissions necessary for the role to perform its tasks. This minimizes the potential impact of a compromised role.

**7. Propagation Delay**

* **Understanding Delays**: Changes made to IAM roles and policies may take some time to propagate across AWS services. This means that after attaching a role or modifying its permissions, there may be a brief period during which the changes are not immediately effective.
* **Testing Permissions**: When testing permissions after making changes, be aware of this propagation delay. If you encounter access issues immediately after a change, it may be due to this delay.

**8. Best Practices for IAM Roles**

* **Regular Reviews**: Regularly review IAM roles and policies to ensure they are still relevant and aligned with your security requirements. Remove any roles or policies that are no longer needed.
* **Specific Permissions**: Use specific permissions rather than broad ones. This reduces the risk of over-permissioning and helps maintain a secure environment.
* **Monitor Role Usage**: Utilize AWS CloudTrail and other monitoring tools to track the usage of IAM roles. This helps identify any unusual activity and ensures that roles are being used as intended.

**47)EC2 Instances Purchasing Options:**

**1. On-Demand Instances**

* **Definition**: Pay for compute capacity by the second (Linux/Windows) or by the hour (other OS) without long-term commitments.
* **Use Cases**: Ideal for short-term, unpredictable workloads where you cannot forecast usage patterns.
* **Advantages**:
  + No upfront costs.
  + Flexibility to scale up or down as needed.
  + Suitable for applications with variable workloads.

**2. Reserved Instances**

* **Definition**: Commit to using a specific instance type in a specific region for a one- or three-year term in exchange for a significant discount.
* **Use Cases**: Best for steady-state applications, such as databases, where you can predict usage over time.
* **Types**:
  + **Standard Reserved Instances**: Offer the highest discounts but are less flexible.
  + **Convertible Reserved Instances**: Allow changes to instance types, families, or operating systems, but with slightly lower discounts.
* **Advantages**:
  + Cost savings compared to on-demand pricing.
  + Ability to reserve capacity in specific Availability Zones (AZs).

**3. Savings Plans**

* **Definition**: A flexible pricing model that provides significant savings on AWS usage in exchange for a commitment to a specific amount of usage (in dollars) over one or three years.
* **Use Cases**: Suitable for users who want to optimize costs without being tied to specific instance types.
* **Advantages**:
  + Flexibility across instance families and sizes.
  + Can switch between operating systems and tenancy types.
  + Offers similar discounts to reserved instances.

**4. Spot Instances**

* **Definition**: Purchase unused EC2 capacity at discounted rates, but instances can be terminated by AWS if the spot price exceeds your bid.
* **Use Cases**: Ideal for flexible, fault-tolerant workloads such as batch processing, data analysis, and image processing.
* **Advantages**:
  + Significant cost savings (up to 90% off on-demand prices).
  + Suitable for workloads that can tolerate interruptions.

**5. Dedicated Hosts**

* **Definition**: Physical servers dedicated to your use, allowing you to control instance placement and meet compliance requirements.
* **Use Cases**: Useful for applications with strict licensing requirements or regulatory compliance needs.
* **Advantages**:
  + Full control over the physical server.
  + Ability to use existing server-bound software licenses.

**6. Dedicated Instances**

* **Definition**: Instances that run on hardware dedicated to you, but you may share the hardware with other instances in your account.
* **Use Cases**: Suitable for workloads that require physical isolation but do not need the full control of dedicated hosts.
* **Advantages**:
  + Provides isolation from other AWS customers.
  + Less expensive than dedicated hosts.

**7. Capacity Reservations**

* **Definition**: Reserve capacity for on-demand instances in a specific AZ for any duration without a time commitment.
* **Use Cases**: Ideal for applications that require guaranteed capacity in a specific AZ, especially during peak usage times.
* **Advantages**:
  + Ensures capacity availability when needed.
  + No discounts; billed at on-demand rates.

**Summary of Key Points**

* **Flexibility vs. Commitment**: On-demand instances offer flexibility, while reserved instances and savings plans require a commitment but provide cost savings.
* **Workload Suitability**: Choose the instance type based on workload predictability and tolerance for interruptions (e.g., spot instances for flexible workloads).
* **Security and Compliance**: Dedicated hosts and instances are suitable for applications with compliance requirements or specific licensing needs.
* **Capacity Management**: Capacity reservations ensure that you have the necessary resources available when needed, without long-term commitments.

**48)IP Address Charged In AWS:-**

**Key Points**

1. **Public IPv4 Usage**:
   * When you create an EC2 instance, it typically gets a public IPv4 address.
   * Multiple EC2 instances can share the 750-hour limit. For example, if you have several instances running, their combined usage should stay within this limit to avoid charges.
2. **Other Services**:
   * For services other than EC2 (e.g., load balancers, RDS), there is no free tier for public IPv4 addresses, and charges will apply immediately upon creation.
3. **Load Balancers**:
   * Each load balancer can have a public IPv4 address per Availability Zone (AZ). If you deploy a load balancer across multiple AZs, you will incur charges for each public IPv4 address.
4. **RDS Instances**:
   * Creating a public IPv4 address for an Amazon RDS database incurs charges immediately, as there is no free tier for this service.
5. **IPv6 Transition**:
   * AWS encourages the use of IPv6, which does not incur charges for addresses. However, many internet service providers may not support IPv6 yet, making it less practical for some users.
6. **Monitoring Charges**:
   * To monitor and troubleshoot IPv4 charges, you can check your billing details in the AWS Management Console under **Billing and Cost Management**.
   * Use the **AWS Public Insights service** and **Amazon VPC IP Address Manager (IPAM)** to track and manage your public IP addresses.
7. **IPAM Setup**:
   * You can create an IPAM to gain insights into your public IPv4 usage. This can be done within the free tier, allowing you to monitor your public IPs effectively.

**49)Share responsibility in EC2:-**

1. **AWS Responsibilities:**
   * Infrastructure Security: AWS is responsible for the security of the underlying infrastructure, including data centers, hardware, and network components.
   * Physical Isolation: AWS ensures isolation on physical hosts, especially for services like dedicated hosts.
   * Fault Management: AWS handles the replacement of faulty hardware and ensures compliance with relevant regulations.
2. **User Responsibilities:**
   * **Security Configuration:** Users are responsible for configuring security settings, including defining security group rules that control access to EC2 instances.
   * Operating System Management: Users own and manage the operating system (Windows or Linux) running on their EC2 instances, including applying patches and updates.
   * Software Management: Users are responsible for all software and utilities installed on their EC2 instances.
   * IAM Roles and Permissions: Users must correctly assign IAM roles and manage permissions to ensure secure access to AWS resources.
   * Data Security: Users are responsible for securing data stored on their EC2 instances.

**SUMMARY:-**

**1. EC2 Instances**

* **Definition**: Amazon EC2 (Elastic Compute Cloud) provides resizable compute capacity in the cloud, allowing users to run virtual servers (instances) for various applications.
* **Components**:
  + **AMI (Amazon Machine Image)**: A pre-configured template that includes the operating system, application server, and applications. It defines the software environment for the instance.
  + **Instance Type**: Specifies the hardware configuration, including CPU, memory, and storage. Examples include t2.micro, m5.large, etc.
  + **Storage**: EC2 instances use Elastic Block Store (EBS) for persistent storage. Users can choose the size and type of storage (e.g., SSD, HDD).

**2. Security Groups**

* **Definition**: Virtual firewalls that control inbound and outbound traffic to EC2 instances.
* **Functionality**: Users define rules to specify which IP addresses and ports can access the instance, enhancing security. Security groups are stateful, meaning if you allow incoming traffic, the response is automatically allowed.

**3. EC2 User Data**

* **Definition**: A script that runs automatically when an EC2 instance is launched for the first time.
* **Purpose**: Used to automate the configuration of the instance, such as installing software or setting up services. For example, a script to configure a web server to display "Hello, world."

**4. SSH (Secure Shell)**

* **Definition**: A protocol used to securely connect to EC2 instances via a terminal.
* **Usage**: Allows users to issue commands on the instance. SSH typically operates over port 22, and users must have the appropriate permissions and security group rules to connect.

**5. EC2 Instance Roles**

* **Definition**: IAM roles that provide permissions to EC2 instances to interact with other AWS services without needing to manage AWS credentials.
* **Functionality**: Allows instances to perform actions (e.g., accessing S3 buckets) securely and efficiently.

**6. Purchasing Options**

* **On-Demand Instances**:
  + **Definition**: Pay for compute capacity by the hour or second without long-term commitments.
  + **Use Case**: Ideal for short-term, unpredictable workloads.
* **Spot Instances**:
  + **Definition**: Purchase unused EC2 capacity at discounted rates, with the risk of termination if the spot price exceeds your bid.
  + **Use Case**: Suitable for flexible, fault-tolerant workloads that can tolerate interruptions.
* **Reserved Instances**:
  + **Definition**: Commit to using a specific instance type in a specific region for a one- or three-year term in exchange for a discount.
  + **Types**:
    - **Standard Reserved Instances**: Higher discounts, less flexibility.
    - **Convertible Reserved Instances**: More flexibility to change instance types, slightly lower discounts.
  + **Use Case**: Best for steady-state applications with predictable usage.
* **Savings Plans**:
  + **Definition**: A flexible pricing model that provides savings based on a commitment to a specific amount of usage (in dollars) over one or three years.
  + **Use Case**: Suitable for users who want to optimize costs without being tied to specific instance types.

**7. Shared Responsibility Model**

* **AWS Responsibilities**:
  + Security of the underlying infrastructure, including data centers and hardware.
  + Physical isolation and management of hardware failures.
* **User Responsibilities**:
  + Configuring security settings, including security groups.
  + Managing the operating system, including updates and patches.
  + Installing and managing software on the instance.
  + Assigning IAM roles and managing permissions.
  + Ensuring data security on the instance.

**8. IPv4 Charges in AWS**

* **New Charges**: As of February 1, 2024, AWS charges $0.005 per hour for all public IPv4 addresses created in your account, regardless of usage.
* **Free Tier for EC2**: New accounts have a free tier for 750 hours of public IPv4 usage per month for EC2 instances during the first 12 months.
* **Other Services**: No free tier for public IPv4 addresses in services like RDS or load balancers, leading to immediate charges upon creation.
* **Monitoring Charges**: Use the AWS Billing and Cost Management console to track charges and the Amazon VPC IP Address Manager (IPAM) for insights into public IP usage.