IAM (Identity and Access Management)

In Amazon Web Services (AWS), IAM (Identity and Access Management) roles are a secure way to delegate permissions to entities (such as users, applications, or services) without the need to share long-term access keys. IAM roles are commonly used to grant temporary permissions for specific tasks.

Components of IAM

Users: IAM users represent individual people or entities (such as applications or services) that interact with your AWS resources. Each user has a unique name and security credentials (password or access keys) used for authentication and access control.

Groups: IAM groups are collections of users with similar access requirements. Instead of managing permissions for each user individually, you can assign permissions to groups, making it easier to manage access control. Users can be added or removed from groups as needed.

Roles: IAM roles are used to grant temporary access to AWS resources. Roles are typically used by applications or services that need to access AWS resources on behalf of users or other services. Roles have associated policies that define the permissions and actions allowed for the role.

Policies: IAM policies are JSON documents that define permissions. Policies specify the actions that can be performed on AWS resources and the resources to which the actions apply. Policies can be attached to users, groups, or roles to control access. IAM provides both AWS managed policies (predefined policies maintained by AWS) and customer managed policies (policies created and managed by you).

Here are some key points about AWS IAM roles:

1. **Delegated Permissions:** IAM roles allow you to delegate permissions to trusted entities. This could be an IAM user within the same AWS account, an IAM user in a different AWS account, or a service or application outside of AWS.

2. **Temporary Security Credentials:** When a role is assumed, AWS provides temporary security credentials that can be used to access AWS resources. These credentials have a limited duration, typically ranging from a few minutes to several hours.

3. **Assumption of Roles:** IAM roles can be assumed by IAM users, AWS services, or by identity federation using external identity providers (such as Microsoft Active Directory, Face book, or Google).

4. **Trust Policy:** Each IAM role has a trust policy that specifies who can assume the role. The trust policy defines the trusted entities (users, accounts, or services) that are allowed to assume the role and the conditions under which they can do so.

5. **Permissions Policies:** IAM roles have associated permissions policies that define what actions can be performed on which AWS resources. These policies are similar to those attached to IAM users.

6. **Cross-Account Access:** IAM roles can be used for cross-account access, allowing entities in one AWS account to assume a role in another account.

7. **Role ARN (Amazon Resource Name):** Roles are identified by a unique ARN, which includes information about the AWS account, the role name, and an optional path.

IAM roles are a fundamental component of securing and managing access to AWS resources, providing a flexible and secure way to control permissions within an AWS environment. They are commonly used in various scenarios, such as granting permissions to EC2 instances, allowing applications to access AWS services, and facilitating cross-account access.

Creating a normal user in AWS IAM (Identity and Access Management)

Creating a normal user in AWS IAM (Identity and Access Management) involves several steps. Below are the general steps to create an IAM user using the AWS Management Console:

1. **Sign in to the AWS Management Console:**
    - Open the AWS Management Console in your web browser.
    - Sign in with your AWS account credentials.
2. **Navigate to IAM:**
    - In the AWS Management Console, navigate to the IAM service. You can usually find it under "Services" in the "Security, Identity, & Compliance" section.
3. **Select "Users" in the IAM Dashboard:**
    - In the IAM dashboard, select "Users" from the left navigation pane.
4. **Click "Add user":**
    - Click the "Add user" button to start the process of creating a new IAM user.

5. **Enter User Details:**
    - Enter the username for the new IAM user.
    - Choose the type of access for the user:
        - **Programmatic access:** Allows the user to interact with AWS using the AWS CLI, SDKs, and other API-based tools.
        - **AWS Management Console access:** Allows the user to sign in to the AWS Management Console with a username and password.
6. **Set Permissions:**
    - Choose how you want to set permissions for the user:
        - **Add user to group:** Add the user to one or more existing IAM groups with predefined permissions.
        - **Attach policies directly:** Attach one or more policies directly to the user, specifying individual permissions.
7. **Add Tags (Optional):**
    - Optionally, you can add tags to the user for better organization and resource management.
8. **Review:**
    - Review the user details, permissions, and policies.
9. **Configure:**
    - Optionally, you can configure settings such as enabling programmatic access, setting up console password, and defining permissions boundaries.
10. **Review and Create:**
    - Review the configuration, and if everything looks correct, click "Create user."
11. **View Security Credentials:**
    - After creating the user, you will be prompted to download or copy the user's security credentials. Make sure to securely store these credentials as they are needed for programmatic access.

Once the user is created, they can sign in to the AWS Management Console (if console access is enabled) or use the programmatic access credentials to interact with AWS services through APIs and SDKs.

Remember to follow best practices for IAM, such as the principle of least privilege, to ensure that users have only the permissions they need to perform their tasks.

Benefits of using IAM:

- Improved security: IAM helps you to improve the security of your AWS resources by giving you granular control over who has access to what.
- Reduced compliance risk: IAM can help you to meet compliance requirements by providing a way to audit user activity and enforce least privilege.
- Simplified administration: IAM makes it easy to manage user access to AWS resources from a single place.

ELASTIC LOAD BALANCER

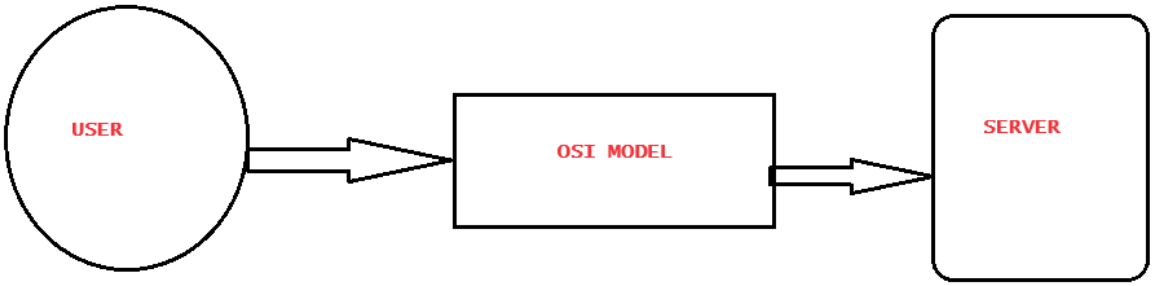
In AWS, ELB stands for Elastic Load Balancing. Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. This helps ensure that no single resource becomes a bottleneck and improves the fault tolerance of your application.

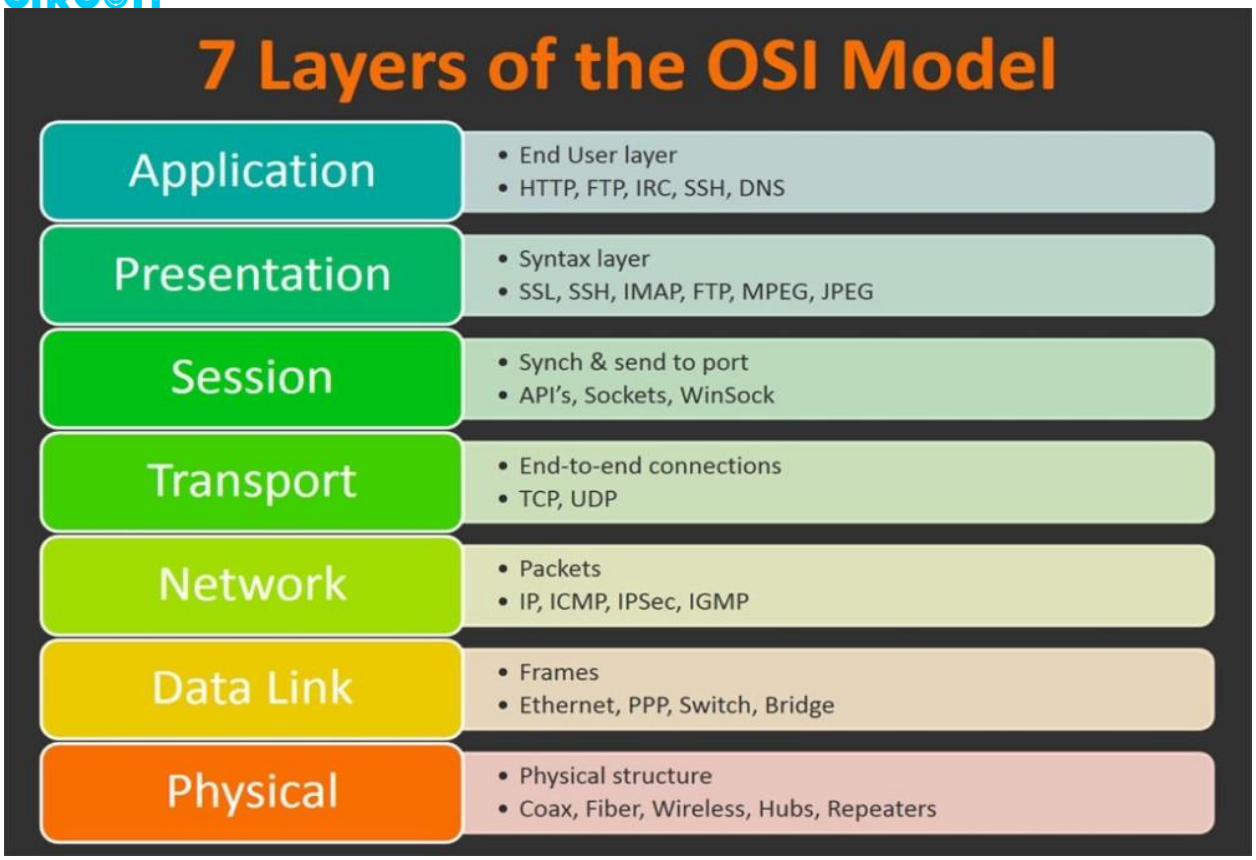
Here are key points about Elastic Load Balancing (ELB) in AWS:

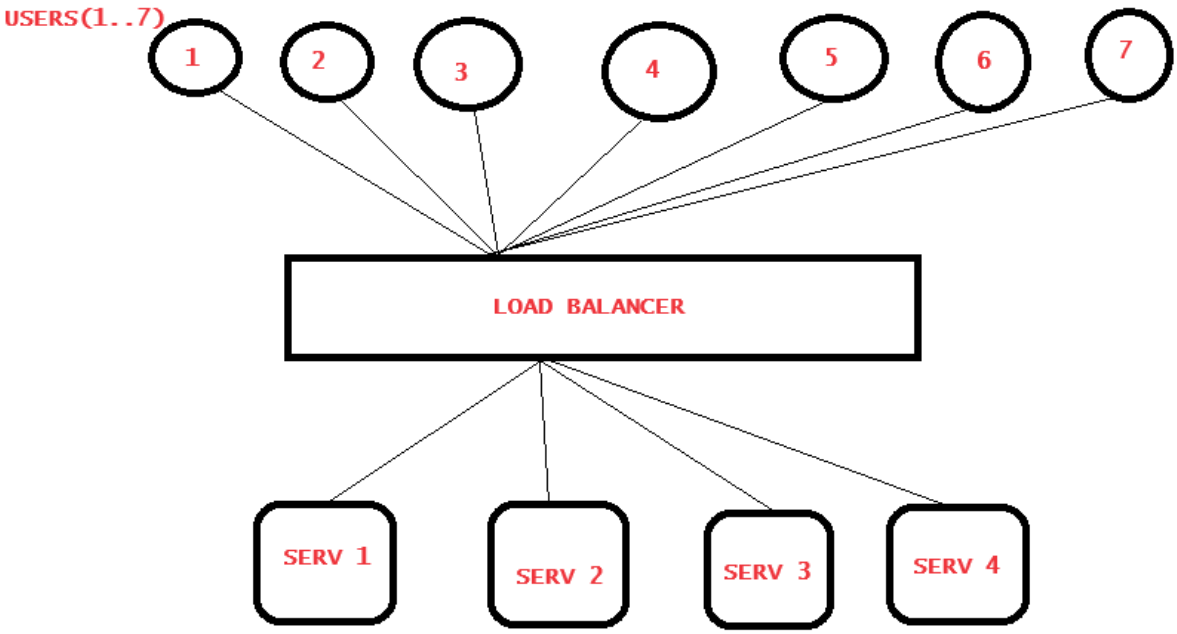
1. **Types of Elastic Load Balancers:**
    - Application Load Balancer (ALB): Operates at the application layer (Layer 7) and allows routing decisions based on content. It is ideal for applications that run in containers.
    - Network Load Balancer (NLB): Operates at the transport layer (Layer 4) and is designed to handle TCP, UDP, and TLS traffic. It is suitable for applications that require high performance and low latency.
    - Classic Load Balancer: Provides basic load balancing across multiple Amazon EC2 instances and operates at both the application and transport layers.
2. **Key Features:**
    - Health Checks: ELB automatically performs health checks on EC2 instances or other targets to ensure that traffic is routed only to healthy instances.
    - Auto Scaling Integration: ELB can be easily integrated with Auto Scaling groups to automatically scale the number of EC2 instances based on traffic.
    - SSL/TLS Termination: ELB can terminate SSL/TLS connections, offloading the SSL/TLS decryption process from the EC2 instances.
    - IPv6 Support: ELB supports both IPv4 and IPv6 traffic.
3. **Distribution of Traffic:**
    - ELB distributes incoming traffic across multiple targets within one or more Availability Zones, improving fault tolerance.
4. **Security Groups:**
    - ELB is associated with security groups, allowing you to control inbound and outbound traffic to and from the load balancer.

5. **Access Logs:**
    - ELB can generate access logs, providing detailed information about requests sent to the load balancer.
6. **Listeners and Rules:**
    - ELB uses listeners and rules to route traffic to different target groups based on the content of the request.
7. **Integration with Other AWS Services:**
    - ELB can be used in conjunction with other AWS services like AWS Certificate Manager for SSL/TLS certificates and AWS WAF for web application firewall protection.

Setting up an Elastic Load Balancer involves creating the load balancer, configuring listeners and rules, associating target groups, and adjusting various settings based on your application's requirements. ELB plays a crucial role in building scalable and resilient architectures in AWS.




7 Layers of the OSI Model
Application
• End User layer
• HTTP, FTP, IRC, SSH, DNS
Presentation
• Syntax layer
• SSL, SSH, IMAP, FTP, MPEG, JPEG
Session
• Synch & send to port
• API's, Sockets, WinSock
Transport
• End-to-end connections
• TCP, UDP
Network
• Packets
• IP, ICMP, IPSec, IGMP
Data Link
• Frames
• Ethernet, PPP, Switch, Bridge
Physical
• Physical structure
• Coax, Fiber, Wireless, Hubs, Repeaters


USERS(1..7)
1
2
3
4
5
6
7
LOAD BALANCER
SERV 1
SERV 2
SERV 3
SERV 4

- The load balancer receives the incoming request and uses its routing configuration to determine which target (registered instance or service) should handle the request. This routing can be based on various factors, such as the path or hostname in the URL.
- The load balancer distributes the incoming traffic among the registered targets based on the configured routing rules. It considers the health of the targets and only sends traffic to healthy instances.
- The registered targets (e.g., EC2 instances, containers) receive the requests forwarded by the load balancer. Each target processes the request independently.
- The load balancer collects the responses from the registered targets and sends the appropriate response back to the user who initiated the request.
- The load balancer continuously monitors the health of the registered targets through health checks. If a target becomes unhealthy, the load balancer temporarily stops sending traffic to that target. If new healthy targets are added (e.g., due to Auto Scaling), the load balancer includes them in the distribution of traffic.

By using a load balancer, users can access the application seamlessly, and the load balancer handles the distribution of traffic, ensures high availability, and adapts to changes in the underlying infrastructure. This architecture improves the overall reliability and scalability of web applications.

Creating a target group

Creating a target group is an essential step when working with AWS Elastic Load Balancing (ELB), specifically for the Application Load Balancer (ALB) and Network Load Balancer (NLB). Target groups are used to route requests to registered targets, such as EC2 instances, and enable you to define health checks to ensure that only healthy targets receive traffic.

Below are the general steps to create a target group using the AWS Management Console:

1. **Sign in to the AWS Management Console:**
    - Open the AWS Management Console in your web browser.
    - Sign in with your AWS account credentials.
2. **Navigate to the EC2 Dashboard:**
    - In the AWS Management Console, navigate to the EC2 service.

3. **Select "Target Groups" in the EC2 Dashboard:**
    - In the EC2 Dashboard, under the "Load Balancing" section, select "Target Groups" from the left navigation pane.
4. **Click "Create target group":**
    - Click the "Create target group" button to start the process of creating a new target group.
5. **Configure Target Group:**
    - Fill in the details for your target group, including:
        - **Name:** Give your target group a descriptive name.
        - **Protocol:** Specify the protocol for the target group (HTTP, HTTPS, TCP, etc.).
        - **Port:** Set the port on which the targets will receive traffic.
        - **VPC:** Choose the Virtual Private Cloud (VPC) for the target group.
6. **Health Checks:**
    - Configure health checks to determine the health of the targets. This includes setting the protocol, path, and response timeout.
7. **Register Targets:**
    - Optionally, you can register targets (e.g., EC2 instances) to the target group during creation or later.
8. **Review and Create:**
    - Review the configuration details, and if everything looks correct, click the "Create" button.

Once the target group is created, it can be associated with an Application Load Balancer or Network Load Balancer. The load balancer will then use the target group to route incoming requests to the registered targets based on the defined rules and health checks.

These steps might vary slightly depending on the type of load balancer you are working with (ALB or NLB). Ensure that you configure the target group according to your application's needs and traffic routing requirements.

Creating a load balancer

Creating a load balancer in AWS involves several steps.Below are the general steps for creating an Application Load Balancer using the AWS Management Console:

1. **Sign in to the AWS Management Console:**
    - Open the AWS Management Console in your web browser.
    - Sign in with your AWS account credentials.
2. **Navigate to the EC2 Dashboard:**
    - In the AWS Management Console, navigate to the EC2 service.
3. **Select "Load Balancers" in the EC2 Dashboard:**
    - In the EC2 Dashboard, under the "Load Balancing" section, select "Load Balancers" from the left navigation pane.
4. **Click "Create Load Balancer":**
    - Click the "Create Load Balancer" button to start the process of creating a new load balancer.
5. **Choose Load Balancer Type:**
    - Select the type of load balancer you want to create. Choose "Application Load Balancer" for HTTP/HTTPS traffic routing.
6. **Configure Load Balancer:**
    - Fill in the details for your load balancer, including:
        - **Name:** Give your load balancer a descriptive name.
        - **Scheme:** Choose whether the load balancer should be internal or internet-facing.
        - **Listeners:** Specify the protocol and port for incoming traffic.
7. **Configure Security Settings (if applicable):**
    - For an HTTPS (SSL/TLS) load balancer, configure the security settings, including choosing an SSL certificate.

8. **Configure Routing:**
    - Set up routing by creating one or more target groups. Define rules to route traffic to different target groups based on conditions.
9. **Configure Security Groups:**
    - Choose or create security groups to control the traffic to and from your load balancer.
10. **Configure Subnets:**
    - Choose the subnets in which to deploy the load balancer. Subnets are associated with Availability Zones.
11. **Configure Tags (Optional):**
    - Optionally, you can add tags to the load balancer for better organization and resource management.
12. **Review and Create:**
    - Review the configuration details, and if everything looks correct, click the "Create" button.
13. **Wait for the Load Balancer to be Provisioned:**
    - It may take a few minutes for the load balancer to be provisioned.

Once the load balancer is created, it will have a DNS name that you can use to route traffic to your applications.

When you use a load balancer in AWS, the output, or more accurately, the endpoint that clients interact with, is the DNS name associated with the load balancer. This DNS name is provided by AWS and is specific to the load balancer you create. Clients can use this DNS name to send requests to your application, and the load balancer will distribute those requests among the registered targets.

For example, if you create an Application Load Balancer (ALB) named "my-alb" in the AWS Management Console, it will be assigned a DNS name like **my-alb-1234567890.us-west-2.elb.amazonaws.com** (the actual DNS name will be longer and include your region and other details). Clients can use this DNS name to access your application, and the load balancer will route requests to the appropriate targets.