**Auto Scaling Solutions**

**Auto Scaling overview**

* **Auto Scaling**: This is a feature in AWS that automatically adjusts the number of EC2 instances in your application based on the demand. It ensures that you have the right amount of resources to handle the load of your application.
* **Load Balancer Concepts**: Load balancers distribute incoming traffic across multiple targets, such as EC2 instances, to ensure no single instance is overwhelmed, improving the availability and reliability of your application.
* **Elastic Load Balancing (ELB)**: ELB automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones.

**Auto Scaling groups**

* **Auto Scaling Group**: A collection of EC2 instances with similar characteristics that can be scaled in or out based on specific criteria such as CPU utilization or network throughput.
* **Scaling Criteria**: Instances can be scaled based on metrics like CPU utilization, network communications, and free memory. This helps ensure that your application can handle varying loads efficiently.
* **Unhealthy Instances**: Instances that are not in the "Running" state are considered unhealthy and can be automatically replaced by the Auto Scaling group to maintain performance and availability.
* **Availability Zones**: It's beneficial to span Auto Scaling groups across multiple Availability Zones to ensure redundancy and high availability. If one zone fails, instances in other zones can continue to handle the load.
* **Minimum Instances**: You should define a minimum number of instances that are always running to meet your application's baseline needs, even during low demand periods.
* **Scaling Purpose**: Determine whether you are scaling to increase capacity (handling more load) or to increase redundancy (ensuring availability).

These points should help you understand the essential aspects of Auto Scaling groups.

**Termination policies**

* **Auto Scaling**: This feature adjusts the number of EC2 instances based on demand. Scaling out adds instances, while scaling in removes them to save costs.
* **Termination Policies**: These are rules that determine which instances to terminate when scaling in. The default policy considers the following:
  + **Availability Zones**: Selects the zone with the most instances.
  + **Oldest Launch Configuration**: Terminates instances with the oldest configurations first.
  + **Billing Hour**: Prioritizes terminating instances closest to the next billing hour to save costs.
  + **Random Selection**: If multiple instances meet the criteria, one is chosen at random.
* **Custom Termination Policies**: You can define your own rules, such as:
  + **OldestInstance**: Terminates the oldest instance first.
  + **NewestInstance**: Terminates the newest instance, useful for testing.
  + **OldestLaunchConfiguration**: Terminates instances with the oldest configurations.
  + **ClosestToNextInstanceHour**: Terminates instances closest to the next billing hour.

These policies help optimize performance and cost-efficiency by managing which instances to remove during scaling in. Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

**Auto Scaling configuration lab**

* **Auto Scaling Groups**: These groups contain EC2 instances that can automatically scale out (add more instances) or scale in (remove instances) based on demand.
* **Launch Configuration**: This is a template used by the Auto Scaling group to launch new instances. It includes details like the Amazon Machine Image (AMI), instance type, and other configurations.
* **Creating Auto Scaling Groups**: You can either create a launch configuration first or create it during the process of setting up an Auto Scaling group. The video demonstrates creating both at once using a wizard.
* **Steps to Create**:
  1. **Choose an AMI**: Select a base image for your instances.
  2. **Configure Instance Details**: Set properties like instance type, monitoring, and IAM roles.
  3. **Add Storage and Security Groups**: Define storage and security settings for the instances.
  4. **Set Scaling Policies**: Determine how the group should scale based on metrics like CPU utilization.
* **Bootstrapping**: This involves running a script when instances launch to automate setup tasks, such as updates or software installations.
* **Network and Subnets**: Choose the network and subnets for your instances, ensuring they are spread across different Availability Zones for high availability.

These steps help ensure your application can handle varying loads efficiently.

**Launch methods**

* **Launch Templates**: These are templates that include all the necessary parameters to launch an EC2 instance, such as AMI ID and instance type. Ensure the template is fully formed to avoid errors.
* **Launch Configuration**: This method involves creating a launch configuration through the management console and then using it to launch an Auto Scaling group.
* **EC2 Instance**: You can create an Auto Scaling group from an existing EC2 instance by creating an AMI from it and using that AMI to launch the group.
* **Amazon EC2 Launch Wizard**: This wizard helps you launch an Auto Scaling group by creating multiple instances directly and providing an option to put them in an Auto Scaling group.

Understanding these methods is crucial for planning and recommending the appropriate launch method for different scenarios.

**Load balancer concepts**

* **Load Balancer**: A component that distributes incoming requests across multiple servers (nodes) to ensure no single server is overwhelmed. It sits between the requesters and the servers providing the service.
* **Types of Load Balancers**:
  + **Sender Initiated**: The sender (client) decides which server to communicate with.
  + **Receiver Initiated**: The load balancer decides which server will handle the request. This is considered true load balancing.
* **Static Load Balancing**: Specific actions are assigned to specific servers. There is no scalability, but it ensures that different parts of an application are handled by different servers.
* **Dynamic Load Balancing**: Actions are dynamically assigned to servers. This allows for scalability by adding more servers behind the load balancer. AWS Elastic Load Balancing (ELB) uses this method.
* **Load Balancing Algorithms**:
  + **Round Robin**: Requests are distributed in a rotating order.
  + **Randomized**: Requests are distributed based on a pseudorandom algorithm.
  + **Centrally Managed**: The load balancer makes decisions based on the current state of the environment, such as server load or time of day.
  + **Threshold-Based**: Requests are sent to a server until a certain threshold is reached, then directed to another server.
* **Considerations for Implementation**: Ensure all servers have synchronized storage to avoid inconsistent data. This can be done through replication or a shared storage area network.

These points should help you understand the essential aspects of load balancing.

**Elastic Load Balancing (ELB)**

* **Elastic Load Balancing (ELB)**: A service within AWS that distributes incoming application traffic across multiple targets, such as EC2 instances, to ensure high availability and reliability.
* **Benefits of ELB**:
  + **High Availability**: Managed by AWS to ensure the load balancer is always available.
  + **Security**: Configurable and managed by AWS, providing a secure load balancing solution.
  + **Flexibility**: Supports multiple load balancing configurations.
  + **Monitoring and Auditing**: Includes logging and monitoring features to track communications and performance.
  + **Elasticity**: Can scale to handle varying levels of traffic.
* **Types of Load Balancers**:
  + **Application Load Balancer (ALB)**: Operates at the application layer (Layer 7) and is ideal for HTTP/HTTPS traffic.
  + **Network Load Balancer (NLB)**: Operates at the transport layer (Layer 4) and is suitable for handling TCP traffic.
  + **Classic Load Balancer (CLB)**: The original load balancer, now primarily used for legacy applications.
* **Supported Services**: ELB can be used with EC2 instances, Elastic Container Store, Auto Scaling, CloudWatch, and Route 53.
* **Choosing the Right Load Balancer**: For new deployments, use ALB or NLB. CLB is only recommended for legacy applications.

These points should help you understand the essential aspects of Elastic Load Balancing.

**Virtual Network Services Top of Form**

Bottom of Form

Top of Form

Bottom of Form

Top of Form

**DNS**

* **DNS (Domain Name System)**: This system translates human-friendly domain names (like www.website.com) into IP addresses that computers use to communicate with each other.
* **FQDN (Fully Qualified Domain Name)**: This includes the hostname and the domain name, such as aws.Amazon.com. It specifies the exact location within the DNS hierarchy.
* **Resolution Process**: When you type a domain name, your device queries a DNS server to get the corresponding IP address. This process involves multiple steps, starting from the root DNS server down to the specific domain's DNS server.
* **DNS Records**: These are entries in the DNS database that map domain names to IP addresses. Common types include:
  + **A Record**: Maps a hostname to an IPv4 address.
  + **AAAA Record**: Maps a hostname to an IPv6 address.
  + **CNAME Record**: An alias for another domain name.
  + **MX Record**: Specifies the mail server for a domain.

**Configuring DNS lab**

* **Automatic DNS Generation**: AWS automatically generates DNS names for various resources, such as IAM sign-in links and EC2 instances. These names are often long and not user-friendly.
* **Customization**: AWS allows you to customize these automatically generated DNS names to make them more human-friendly. For example, you can create an alias for your IAM sign-in link to make it easier to remember.
* **Route 53**: AWS Route 53 can be used to create even shorter and more user-friendly DNS names by setting up aliases that point to the automatically generated names or public IP addresses.

This video demonstrates how to navigate these features in AWS, ensuring that your DNS names are both functional and user-friendly.

**Top of Form**

**Bottom of Form**

**Configuring Route 53 lab**

* **Route 53 Overview**: Route 53 is AWS's DNS service, allowing you to create aliases, hostnames, and even register domain names. It is a managed service, meaning you don't need to create an instance; it's ready to use.
* **DNS Management**: Route 53 handles DNS management, traffic management, availability monitoring, and domain registration. You can create hosted zones to manage how traffic is routed for a domain.
* **Routing Policies**:
  + **Simple Routing**: Standard DNS routing, resolving domain names to IP addresses.
  + **Weighted Routing**: Distributes traffic based on assigned weights to different IP addresses.
  + **Latency Routing**: Directs traffic to the server with the lowest latency.
  + **Failover Routing**: Uses primary and secondary IP addresses, switching to the secondary if the primary fails.
  + **Geolocation Routing**: Routes traffic based on the user's geographic location.
  + **Multivalue Answer Routing**: Uses health checks to route traffic to the healthiest endpoints.
* **Health Checks**: Route 53 can perform health checks on endpoints to ensure they are available and functioning correctly. These checks can be associated with routing policies to improve reliability.

These concepts help you manage DNS within AWS effectively, ensuring optimal performance and reliability for your applications.Top of Form

Bottom of Form

**Configuring ACLs and NACLs lab**

* **Access Control Lists (ACLs)**: These are used to manage permissions for objects like S3 buckets. They decide who can read, write, or view the contents. ACLs are considered legacy options, and using policies directly is now recommended.
* **Network ACLs (NACLs)**: These are used to control traffic to and from subnets in a Virtual Private Cloud (VPC). They allow you to specify which protocols can communicate with the VPC (inbound rules) and which protocols are allowed out of the VPC (outbound rules).
* **Creating NACLs**: You can create a NACL in the VPC Dashboard, give it a name, and attach it to a VPC. You can then define inbound and outbound rules to control traffic. For example, you can allow HTTP traffic by specifying the protocol and port.
* **Rule Evaluation**: NACLs evaluate rules in order. If a rule matches the traffic, it is applied, and no further rules are checked. By default, all traffic is denied unless explicitly allowed.
* **Subnet Associations**: NACLs can be associated with specific subnets within a VPC. This allows you to control traffic at a more granular level.
* **Security Architect Role**: As an AWS architect, you may need to research the applications running in the VPC to determine the necessary rules for both inbound and outbound traffic.

The video emphasizes the importance of understanding the applications and protocols used in your VPC to create effective and secure NACL rules. Always start with a deny-all approach and only allow necessary traffic to ensure security.

**Flow logs**

* **Flow Logs**: These are used to capture information about the traffic moving through your AWS environment. They record data such as the source and destination of the traffic and the protocols used, but not the actual traffic content.
* **Purpose**: Flow logs help you monitor and analyze network traffic for security and performance purposes. They can identify unusual patterns that might indicate security issues.
* **Setup**:
  + **S3 Bucket**: Start by creating an S3 bucket to store your flow logs.
  + **Network Interfaces**: Go to EC2, select a network interface, and create a flow log. Choose to capture all traffic or filter specific traffic.
  + **VPCs and Subnets**: You can also create flow logs for entire VPCs and subnets through the VPC Dashboard.
* **Usage**: Once set up, flow logs are stored in the S3 bucket. You can analyze them directly or use AWS analytics tools to gain insights, such as detecting potential security threats.
* **Applications**: Flow logs can be used to monitor network activity, troubleshoot connectivity issues, and enhance security by identifying and responding to suspicious traffic patterns.

These concepts will help you effectively use flow logs to manage and secure your AWS environment.

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

Bottom of Form