Auto Scaling Concepts

1. **Auto Scaling**: Automatically adjusts cloud resources to match the workload, optimizing performance and cost.
2. **Horizontal Scaling**: Adds or removes instances (servers) to handle changing workloads.
3. **Vertical Scaling**: Increases or decreases the capacity (CPU, memory) of existing servers.
4. **Metrics and Thresholds**: Predefined values (e.g., CPU usage) that trigger scaling actions.
5. **Health Checks**: Ensure that new instances launched during scaling are operational.

Types of Scaling

1. **Target Tracking Scaling**: Adjusts resources to maintain a specified metric target (e.g., CPU utilization).
2. **Step Scaling**: Changes resources based on specific metric thresholds and the size of the alarm breach.
3. **Scheduled Scaling**: Predefined scaling actions based on predictable time-based demand changes.
4. **Predictive Scaling**: Uses machine learning to predict future demand and scale resources in advance.

Scaling Services

1. **Amazon EC2 Auto Scaling**: Automatically adjusts the number of Amazon EC2 instances (virtual servers) based on demand.
2. **Application Auto Scaling**: Scales resources for AWS services like Amazon ECS, DynamoDB, and Aurora.

Scaling Policies

1. **Scaling Policy**: Defines how and when resources should be adjusted (e.g., add more instances when CPU usage exceeds 75%).
2. **Cooldown Period**: A waiting time after a scaling activity before another scaling action can occur.
3. **Minimum/Maximum Capacity**: Limits that define the smallest and largest number of instances allowed.

Key AWS Services with Auto Scaling

1. **Amazon EC2**: Virtual servers that automatically scale in or out to meet application demand.
2. **Amazon ECS**: Container orchestration service that scales tasks based on demand.
3. **Amazon DynamoDB**: Scales database read/write capacity based on traffic.

4. **Amazon Aurora**: Scales database replicas for high availability and workload distribution.

Auto Scaling and Launch Templates

Spot Instances and Spot Fleets

1. **Spot Instance**: A type of EC2 instance that uses spare capacity at a lower price, up to 90% cheaper than On-Demand instances.
2. **Spot Fleet**: A group of Spot Instances (and optionally On-Demand Instances) managed together to meet your capacity needs at a lower cost.
3. **Spot Fleet Scaling Policies**:
4. **Target Tracking Scaling**: Adjusts the number of instances to maintain a specific metric, like CPU usage.
5. **Step Scaling**: Adjusts instances based on predefined thresholds for metrics.
6. **Scheduled Scaling**: Changes the number of instances based on a set time schedule.

Elastic Beanstalk and Auto Scaling Groups

1. **Elastic Beanstalk**: A service that simplifies application deployment by automatically managing the infrastructure, including load balancers and Auto Scaling groups.
2. **Auto Scaling Group in Elastic Beanstalk**: Manages the EC2 instances for your application, scaling them based on traffic. You can configure it for single or load-balanced environments.

Lifecycle Hooks

1. **Lifecycle Hook**: A way to pause EC2 instances during specific stages (like launching or terminating) to perform custom actions, such as software installation or data retrieval.
2. **Scale-Out Event**: When a new instance is being launched, lifecycle hooks can pause it for custom tasks before it starts accepting traffic.
3. **Scale-In Event**: When an instance is being terminated, lifecycle hooks can pause it to retrieve data or logs before completing the termination.

Predictive Scaling

1. **Predictive Scaling**: Uses machine learning to forecast future traffic patterns and automatically adds EC2 capacity before the demand increases.
2. **Historical Data**: Predictive scaling relies on historical data from Amazon CloudWatch to forecast future needs and scale instances in advance.
3. **Dynamic Scaling**: Works alongside predictive scaling to handle real-time traffic changes, adjusting instances as demand fluctuates.

Elastic Load Balancing (ELB)

1. **Elastic Load Balancing (ELB):** A service that distributes incoming traffic across multiple targets (e.g., EC2 instances, containers, IPs, Lambda) to improve high availability and fault tolerance for applications.

Core ELB Components

- **Listeners:** Processes that check for incoming connection requests using specific protocols and ports.
- **Target Groups:** Collections of resources (like EC2 instances or Lambda functions) where traffic is directed. Health checks ensure that only healthy targets receive traffic.
- **Rules:** Define how traffic is routed to target groups based on specific conditions like source IP or URL paths.

Load Balancer Types

- **Application Load Balancer (ALB):** Operates at the Application Layer (Layer 7) of the OSI model. Ideal for balancing HTTP/HTTPS traffic with advanced routing based on request content.
- **Network Load Balancer (NLB):** Operates at the Transport Layer (Layer 4) and is used for TCP/UDP traffic, handling millions of requests with very low latency.
- **Gateway Load Balancer (GWLB):** Operates at the Network Layer (Layer 3), used for distributing traffic to third-party virtual appliances like firewalls.

Key ELB Features

- **Cross-Zone Load Balancing:** Distributes traffic evenly across all enabled Availability Zones.
- **Health Checks:** Regular checks to determine whether a target is healthy and should continue receiving traffic.

ELB and Auto Scaling Integration

- **Auto Scaling:** Automatically adjusts the number of EC2 instances based on traffic demand. ELB registers new instances and deregisters those being removed, ensuring even traffic distribution.

Deployment Strategies

- **Blue/Green Deployment:** Two identical environments (blue for the current version, green for the new version). After testing, traffic is switched from blue to green, simplifying rollback if issues arise.

- **Canary Deployment:** Gradual rollout of a new version to a small subset of users before releasing it to all users, reducing risk.
- **Rolling Deployment:** Incrementally replaces old instances with new ones, minimizing downtime and risk.
- **In-Place Deployment:** Updates the application version on existing infrastructure without replacing components, but may affect availability during deployment.

ELB with Amazon Route 53 (DNS Service)

- **DNS Failover:** Integrates with Route 53 to automatically route traffic to healthy resources, enhancing application availability and fault tolerance.

AWS PrivateLink

- **PrivateLink:** Allows secure, private access to AWS services like ELB from within a VPC, without using the public internet. This enhances security and simplifies network architecture.

Predictive Scaling: Uses machine learning to predict future demand and scale resources in advance.

1. **Launch Template**: A predefined configuration used to launch EC2 instances. It specifies details like AMI ID, instance type, security groups, and key pairs, helping to manage EC2 configurations.
2. **Versioning**: Multiple versions of a launch template can be created to test new configurations without altering the base setup.
3. **Auto Scaling**: Automatically adjusts the number of EC2 instances to match the current demand, ensuring application performance and cost-efficiency.

Key Terms and Concepts

Review Questions

What is the primary purpose of AWS Auto Scaling?
To ensure applications perform well by automatically adjusting resource capacity based on demand, optimizing both performance and cost.

Explain the difference between horizontal and vertical scaling.
Horizontal scaling adds or removes instances to handle load changes (scaling out/in), while vertical scaling increases or decreases the capacity of existing instances (scaling up/down).

How does Target Tracking Scaling work in AWS Auto Scaling?
It adjusts the number of instances to maintain a specific target metric, such as keeping average CPU utilization at a set percentage.

What is the role of Launch Templates in Auto Scaling?
They define how instances should be launched, including configurations like AMI ID, instance type, and other settings.

Describe how Predictive Scaling enhances application performance.
Predictive Scaling uses machine learning to forecast future demand based on historical data, allowing AWS to scale resources ahead of anticipated traffic spikes.

What are the benefits of enabling Cross-Zone Load Balancing?
It ensures even traffic distribution across all targets in different Availability Zones, enhancing reliability and performance.

In what scenario would you use a Blue/Green Deployment strategy?
When you want to deploy a new version of an application alongside the old one and switch traffic over all at once after testing.

How does AWS Application Auto Scaling differ from EC2 Auto Scaling?
AWS Application Auto Scaling scales multiple AWS services beyond EC2 instances, such as DynamoDB tables and ECS tasks, while EC2 Auto Scaling specifically manages EC2 instances.

Why might you use Scheduled Scaling, and can you provide an example?
Scheduled Scaling adjusts capacity based on known patterns or events. For example, increasing instances every weekday morning when traffic is predictably higher.

How does Step Scaling provide granular control over scaling in AWS Auto Scaling?
Step Scaling adjusts capacity by adding or removing a specific number of instances in response to threshold breaches. It allows you to define multiple steps of adjustments based on how much a metric deviates from the threshold, providing fine-tuned control over scaling actions.

What role do Health Checks play in Auto Scaling Groups, and what happens when an instance is deemed unhealthy?
Health Checks monitor the status of instances within an Auto Scaling Group. If an instance fails a health check, Auto Scaling automatically terminates and replaces it to maintain the desired capacity and ensure application availability.