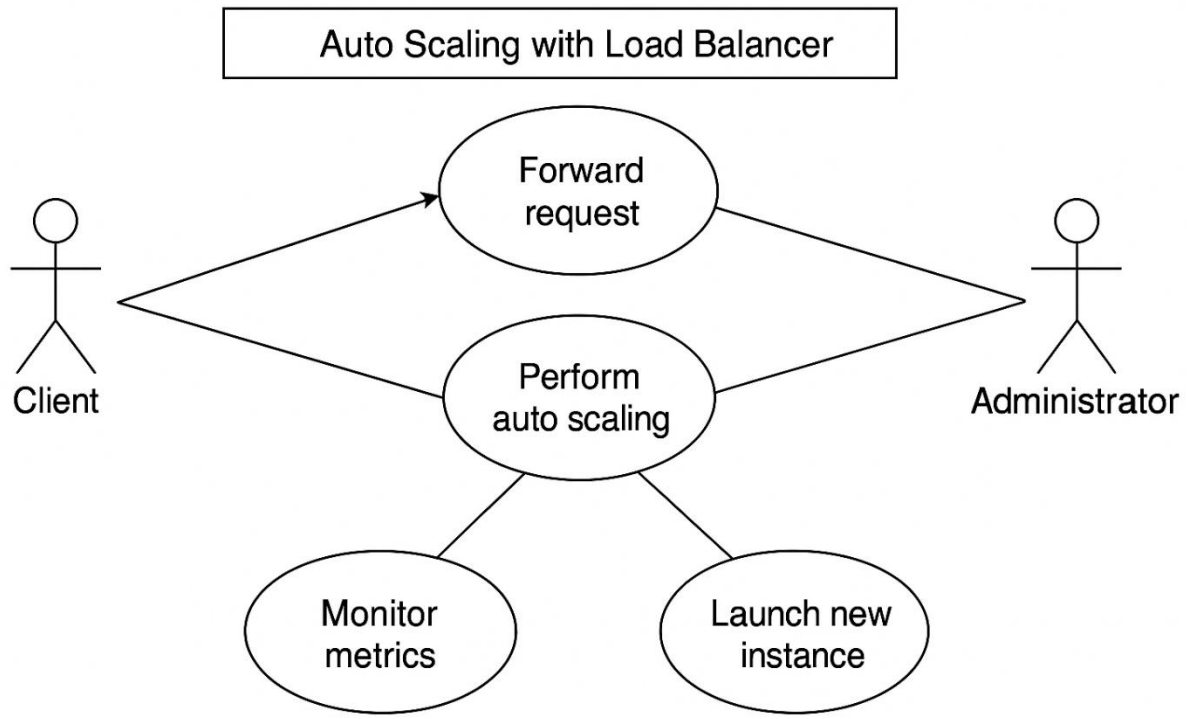
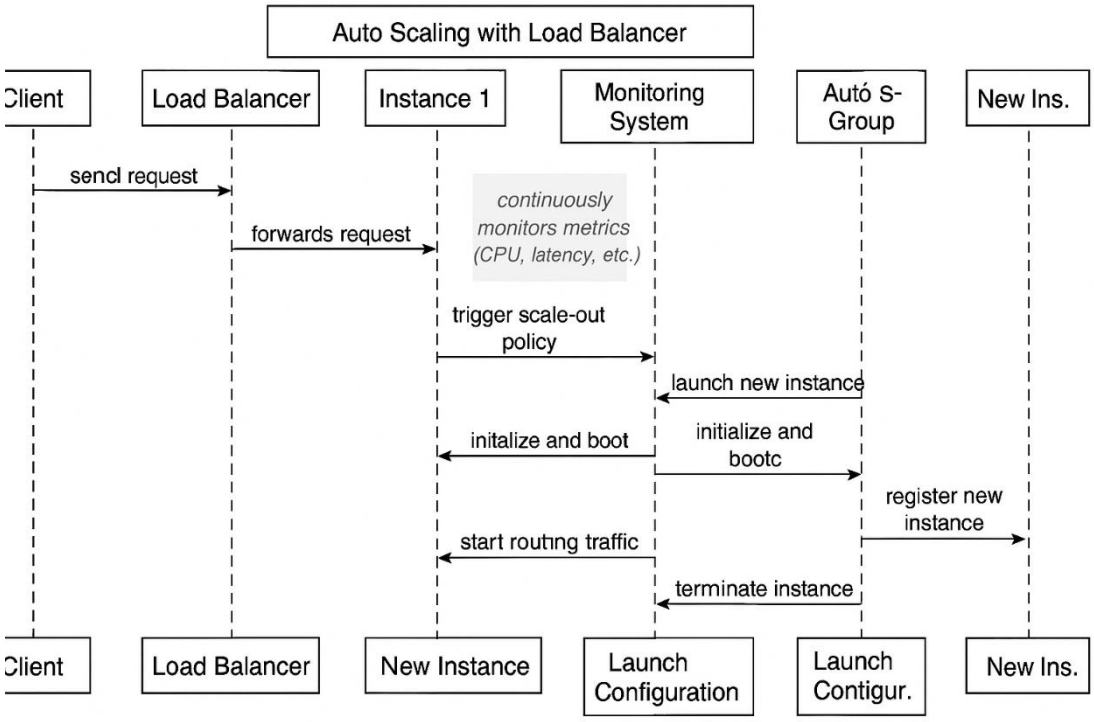
SYSTEM ARCHITECTURE

USECASE diagram for AWS Auto Scaling with Load Balancer



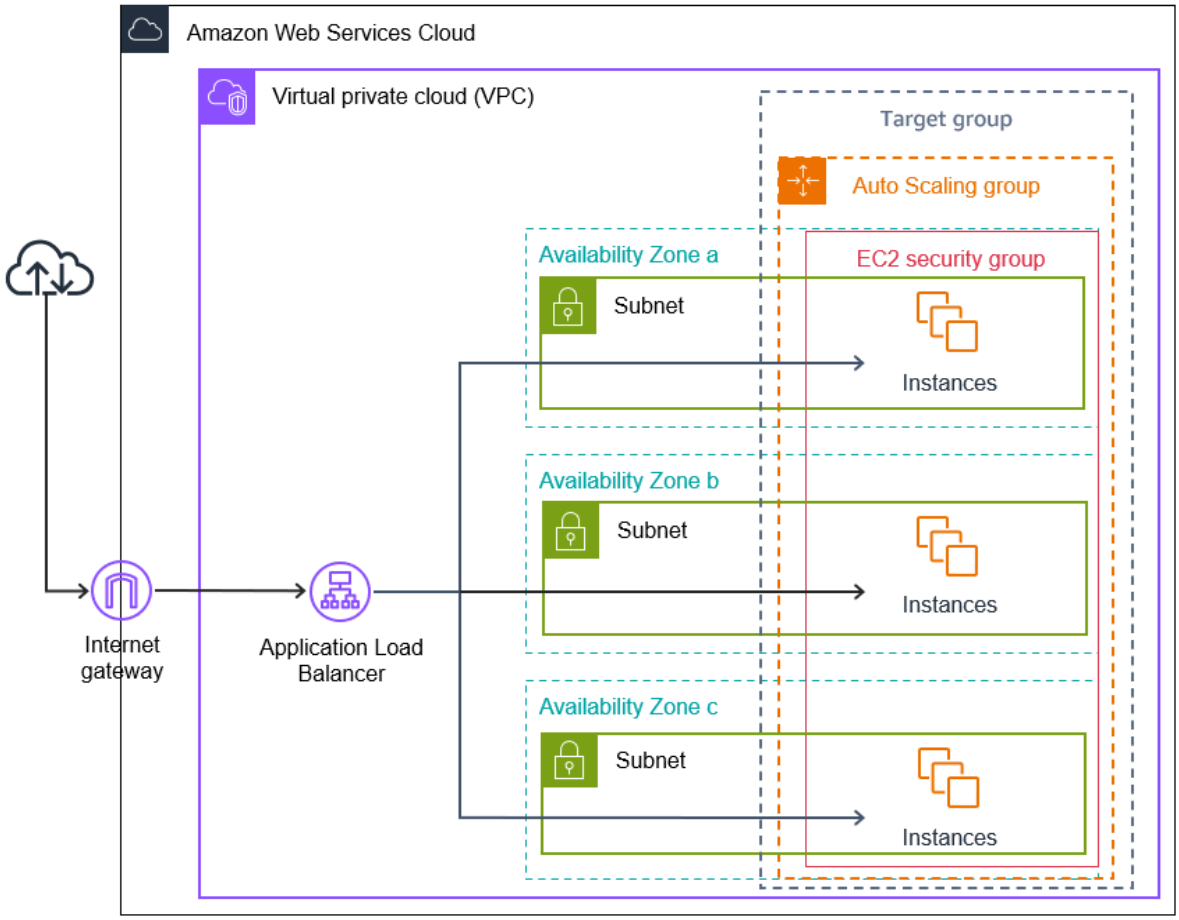
- Client sends a request, which the Load Balancer handles by forwarding it to a healthy EC2 instance.
- Administrator manages or configures the system to enable auto scaling.
- The system automatically monitors metrics (like CPU usage, network traffic) using services like CloudWatch.
- When thresholds are crossed, the Auto Scaling Group (ASG) triggers to launch new instances using a predefined launch template/configuration.
- This ensures high availability, fault tolerance, and cost efficiency under changing loads.

Sequence diagram for AWS Auto Scaling with Load Balancer



1. **Client** sends a request to the **Load Balancer**.
2. Load Balancer forwards it to an EC2 **Instance**.
3. **Monitoring System** checks metrics like CPU and triggers scaling if needed.
4. **Auto Scaling Group** uses the **Launch Configuration** to spin up a new instance.
5. The new instance boots and registers with the **Load Balancer**.
6. Load Balancer starts sending traffic to the new instance.
7. When traffic drops, the instance is terminated.

System Architecture for AWS Auto Scaling with Load Balancer



1. Internet Request Entry

I started by setting up an Internet Gateway so external users could access the app. Requests from the internet come through this gateway and go straight to the Application Load Balancer (ALB) inside my VPC.

2. Application Load Balancer (ALB)

The ALB is configured to listen for HTTP requests. I made sure it spans multiple Availability Zones (AZs) — in my case, AZ-a, AZ-b, and AZ-c — so it's highly available even if one zone goes down.

3. Forwarding to Target Group (Auto Scaling Group)

Once the request hits the ALB, it forwards the traffic to a Target Group, which I linked to my Auto Scaling Group (ASG). This target group is responsible for routing traffic only to healthy EC2 instances.

4. EC2 Instances in Multiple Subnets

I launched EC2 instances in different subnets across AZs. These are managed by the ASG and protected with a proper EC2 security group to control traffic. Each AZ (a, b, and c) has its own subnet where instances can spin up.

5. Auto Scaling – Scale-Out

To handle high traffic, I configured CloudWatch alarms to monitor CPU usage. If usage goes above a certain threshold (e.g., 70% CPU for 5 minutes), the Auto Scaling Group launches new EC2 instances. These new instances automatically register with the ALB after passing health checks.

6. Load Balancer Starts Using New Instances

Once the new EC2s are registered and marked healthy, the ALB starts sending traffic to them. This spreads the load evenly and improves performance.

7. Auto Scaling – Scale-In

When the traffic drops, and resources are underutilized, another CloudWatch alarm triggers a scale-in policy. This safely removes extra instances by deregistering them from the ALB first (so no requests are dropped) and then terminating them.