2. Architecture Diagram & Network Flow

This section describes the overall hybrid architecture and explains how network traffic flows through the system under normal operation, load changes, and failure conditions.

Two architecture diagrams accompany this section:

- **Hybrid Connectivity & Network Flow Diagram**
- **AWS Internal Architecture & Scaling Flow Diagram**

Together, these diagrams illustrate packet movement, security boundaries, failure handling, and automated scaling behavior across on-premise and AWS environments.

2.1 High-Level Architecture Overview

On-Premise Environment (Simulated)

The on-premise environment represents a legacy deployment prior to cloud migration.

- **Ubuntu Linux Server**
  - Hosts the legacy Apache HTTP server and Java application
  - Initially vertically scaled through CPU and memory upgrades
- **MySQL Database**
  - Originally co-located with the application
  - Migrated to Amazon RDS to improve availability and operational simplicity
- **Firewall / Router**
  - Acts as the default gateway for on-prem traffic
  - Serves as the IPsec VPN endpoint
  - Encrypts AWS-bound traffic before transmission over the public internet
  - Enforces basic security controls using UFW / iptables

This environment maintains private IP addressing and does not expose internal services publicly.

AWS Environment

Virtual Private Cloud (VPC)

- Designed with non-overlapping CIDR blocks to avoid routing conflicts
- DNS support enabled for service discovery and internal resolution

Public Subnets (Multi-AZ)

- Host shared, internet-facing components:
    - Internet Gateway (IGW)
    - Application Load Balancer (ALB)
    - NAT Gateway

Private Subnets (Multi-AZ)

- Host internal workloads with no public IPs:
    - Auto Scaling Group (ASG) of Java application EC2 instances
    - Amazon RDS MySQL (Multi-AZ)
- No direct inbound internet access is permitted

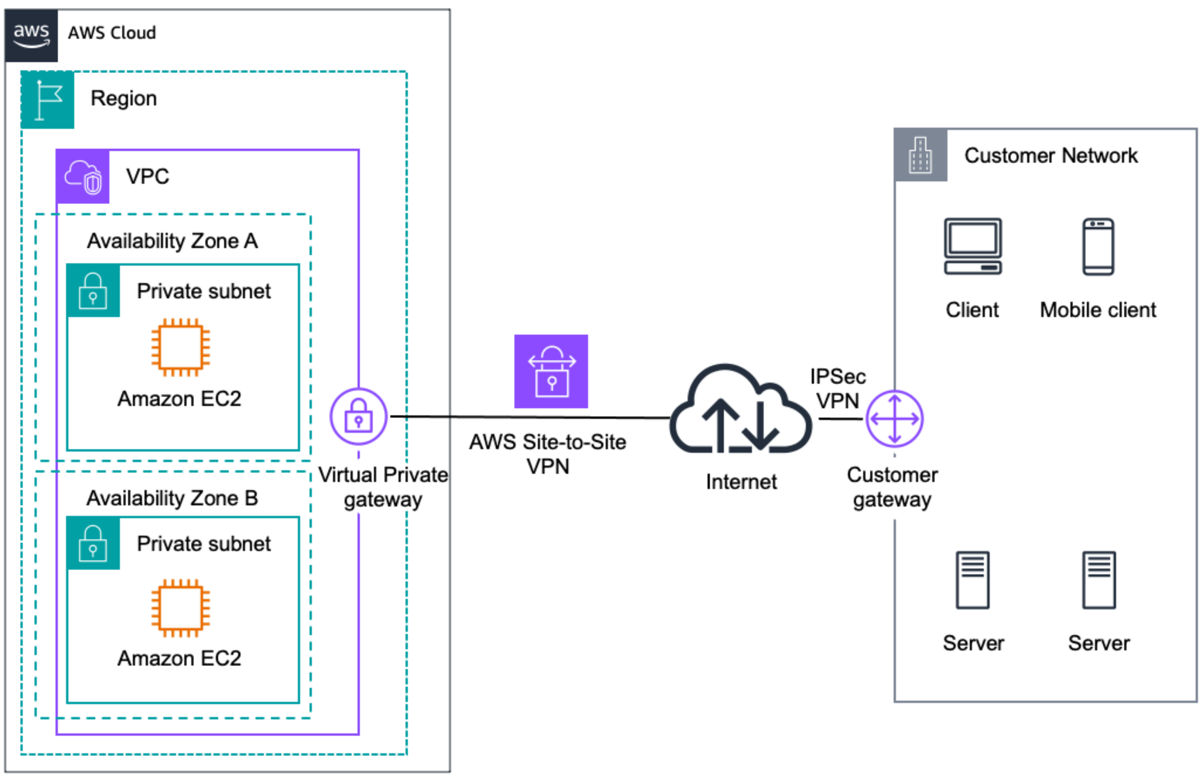
Core AWS Services

- **Application Load Balancer (ALB)**
    - Layer-7 HTTP routing
    - Application-aware health checks
    - Entry point for on-prem traffic via VPN

    - Optional future internet-facing access
- **Auto Scaling Group (ASG)**
    - Maintains desired capacity
    - Automatically replaces unhealthy instances
    - Scales based on CloudWatch metrics
- **Amazon RDS (MySQL)**
    - Deployed in private subnets only
    - Multi-AZ for high availability
    - Automated backups and snapshots enabled
- **NAT Gateway**
    - Enables outbound internet access for private EC2 instances
    - Uses an Elastic IP
    - Does not allow inbound internet initiation
- **IAM (Least Privilege)**
    - EC2 instances use IAM roles
    - Permissions limited to CloudWatch Logs and SSM Session Manager
    - No hard-coded credentials
- **CloudWatch**
    - Metrics, alarms, and auto scaling triggers
- **Route 53**
    - DNS abstraction layer
    - Decouples infrastructure from fixed IP addresses

2.2 Traffic Flow – Scenario-Based Explanation

Instead of static arrows, the system behavior is explained through real operational scenarios.



Scenario 1: On-Premise → AWS Application Request (VPN)

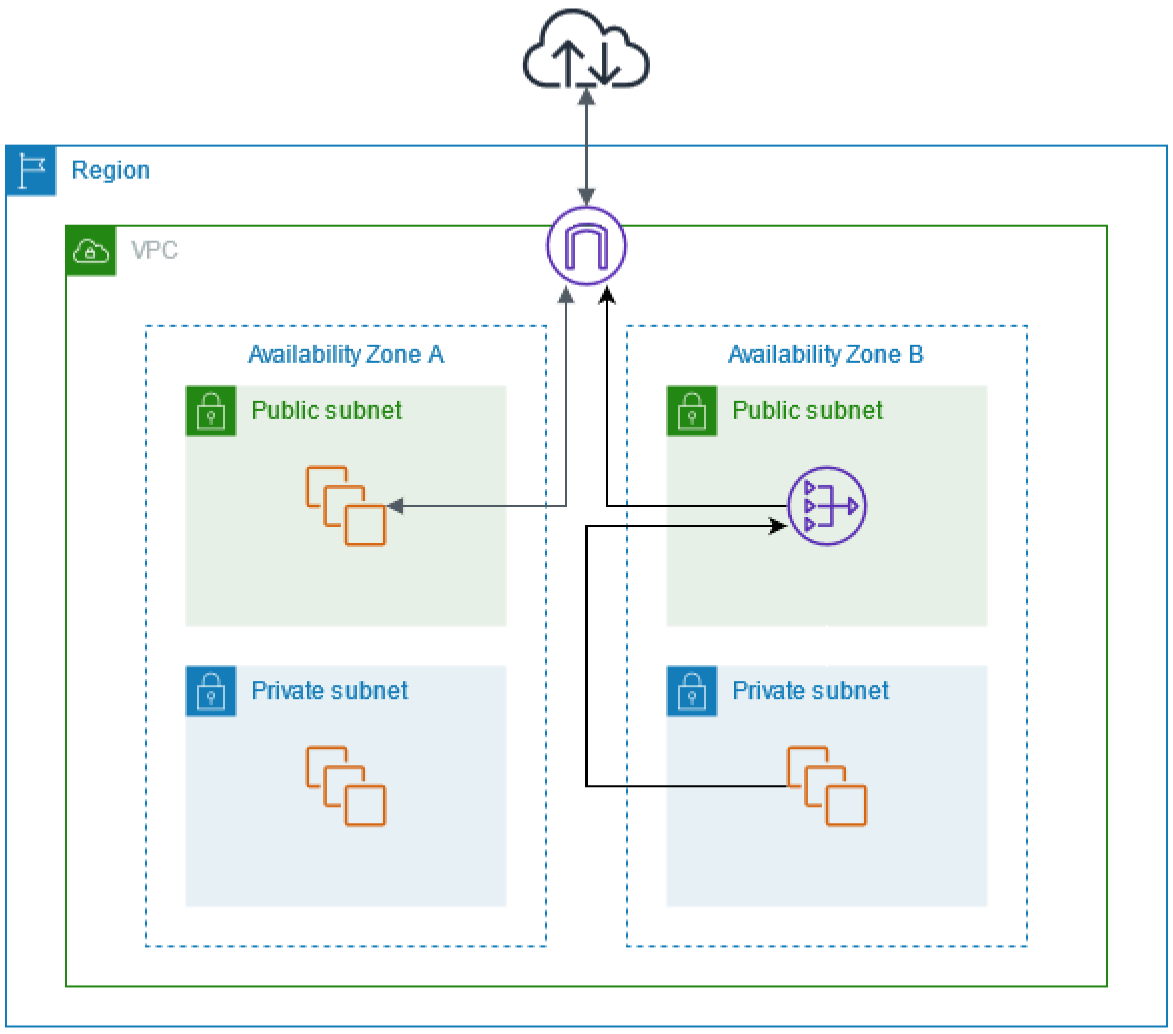
Flow Description

1. A request originates from the on-premise Java application.
2. The destination IP belongs to the AWS CIDR range.
3. Traffic is forwarded to the on-prem firewall (default gateway).
4. The firewall:
    - Matches the AWS route
    - Encrypts the packet using IPsec
    - Sends it through the VPN tunnel over the internet
5. AWS Virtual Private Gateway (VGW):

    - Terminates the VPN tunnel
    - Decrypts the packet
6. The VPC router forwards traffic to the appropriate subnet.
7. The Application Load Balancer:
    - Inspects the HTTP request
    - Verifies target health
    - Forwards traffic to a healthy EC2 instance
8. The EC2 instance:
    - Processes application logic
    - Communicates with RDS over private networking
9. The response returns via the same encrypted VPN path.

Outcome

- Fully private, encrypted communication
- No public exposure of application or database services



Scenario 2: AWS → Internet (Outbound Only via NAT)

Example Use Case

- EC2 instances downloading OS updates

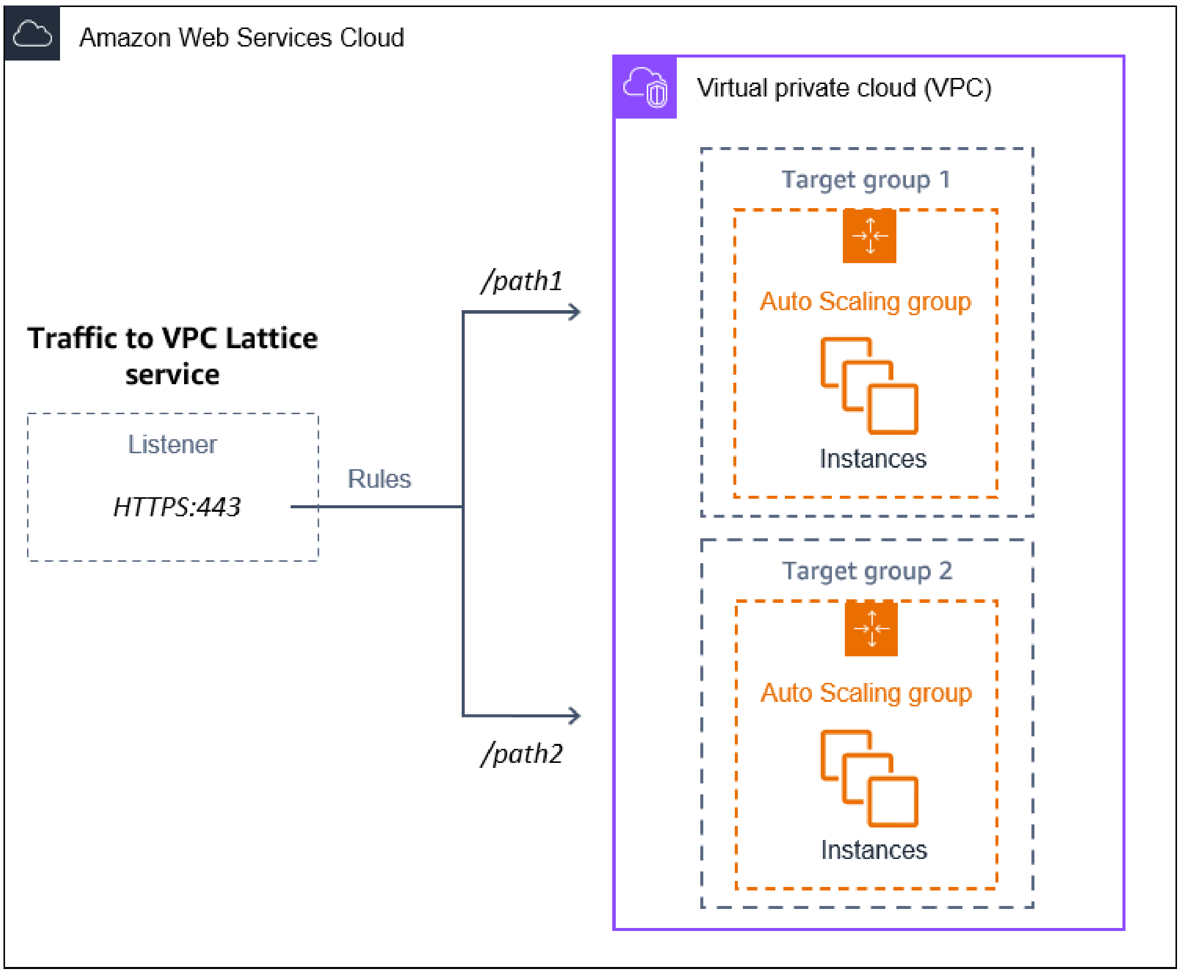
Flow Description

1. A private EC2 instance initiates an outbound request.
2. The subnet route table forwards `0.0.0.0/0` traffic to the NAT Gateway.
3. The NAT Gateway:
   - Translates the private IP to an Elastic IP
   - Sends traffic through the Internet Gateway

4. The internet response returns to the NAT Gateway.
5. The NAT Gateway maps the response back to the originating EC2 instance.

Key Security Property

- Internet-initiated inbound connections are not allowed
- NAT enables response traffic only



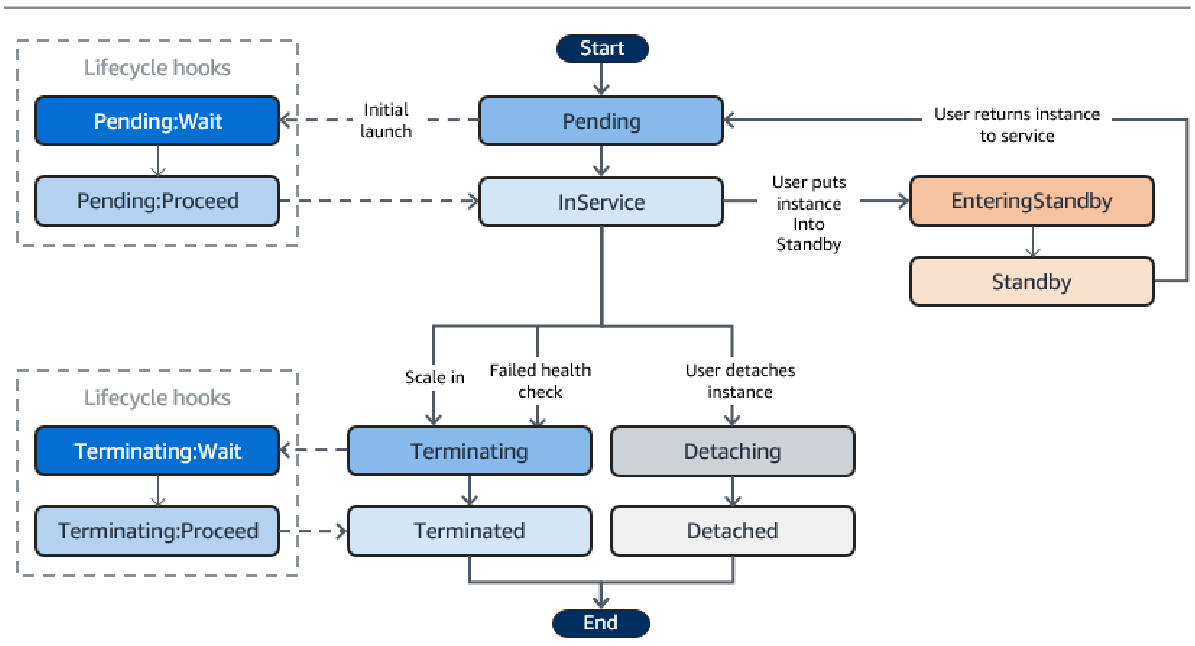
Scenario 3: Load Increase → Auto Scaling (Scale-Out)

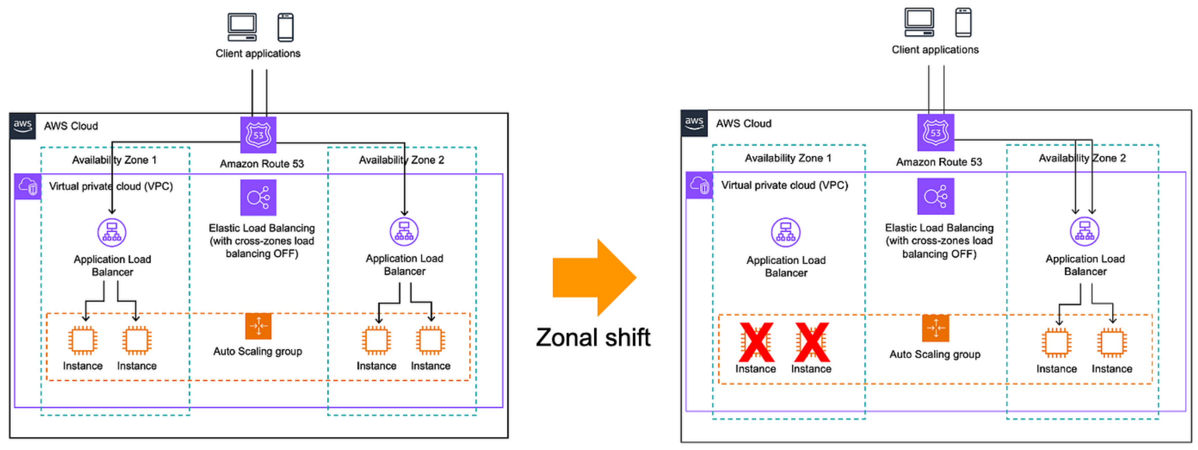
1. Incoming traffic increases at the ALB.
2. Existing EC2 instances experience rising CPU utilization.
3. A CloudWatch alarm is triggered (e.g., CPU ≥ 70%).

4. The Auto Scaling Group:
    - Launches a new EC2 instance
    - Applies the launch template and user-data configuration
5. The new instance:
    - Registers with the ALB target group
    - Begins serving traffic
6. RDS remains unchanged; connection pooling absorbs additional load.

Outcome

- Horizontal scalability without manual intervention




Client applications
AWS Cloud
Amazon Route 53
Availability Zone 1
Availability Zone 2
Virtual private cloud (VPC)
Application Load Balancer
Elastic Load Balancing (with cross-zones load balancing OFF)
Application Load Balancer
Instance
Instance
Auto Scaling group
Instance
Instance
Zonal shift
Client applications
AWS Cloud
Amazon Route 53
Availability Zone 1
Availability Zone 2
Virtual private cloud (VPC)
Application Load Balancer
Elastic Load Balancing (with cross-zones load balancing OFF)
Application Load Balancer
Instance
Instance
Auto Scaling group
Instance
Instance

Scenario 4: Load Decrease → Auto Scaling (Scale-In)

1. Traffic volume decreases.
2. CPU utilization drops below the lower threshold.
3. CloudWatch triggers the scale-in policy.
4. The ASG:
    - Deregisters an instance from the ALB
    - Terminates it gracefully
5. Remaining instances continue serving traffic.

Outcome

- Cost optimization with no downtime

Scenario 5: Instance Failure → Self-Healing

1. An EC2 instance becomes unhealthy.
2. ALB health checks fail.
3. The ASG:
    - Terminates the unhealthy instance

    - Launches a replacement

4. The new instance automatically registers with the ALB.

Outcome

- Fault tolerance through automated recovery

2.3 Design Intent Summary

The architecture is intentionally designed to:

- Maintain private, encrypted connectivity between on-prem and AWS
- Isolate internal workloads from direct internet exposure
- Handle load fluctuations automatically
- Recover from failures without manual intervention
- Remain simple, auditable, and production-aligned