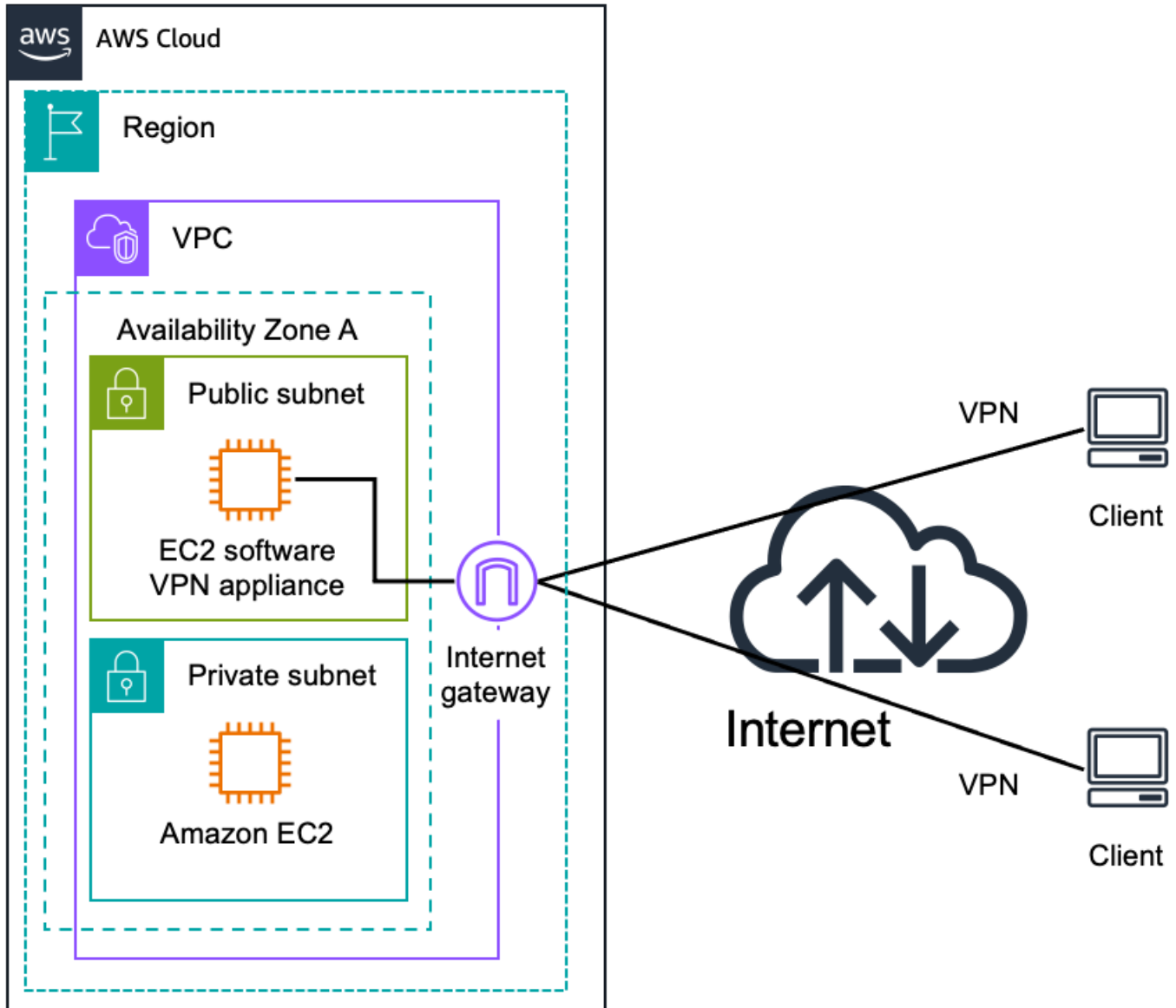


**Project Title:**  
**Creating a Secure and Scalable VPN Service with AWS**

**Group Members:**  
**1. Behzad Moloudi**



## Implementation and Deployment

### 1. Setting Up the Network Infrastructure

- **Implementation:**
  - Created a **Virtual Private Cloud (VPC)** with a CIDR block of **10.0.0.0/16** to establish a secure, isolated environment.
  - Added **two public subnets (10.0.1.0/24 and 10.0.2.0/24)** in separate availability zones for high availability and fault tolerance.
  - Attached an **Internet Gateway (IGW)** to the VPC to allow outgoing internet traffic.
  - Configured **route tables** to direct all external traffic (**0.0.0.0/0**) through the IGW.
- **Challenge:** Ensuring redundancy and fault tolerance.
  - **Solution:** Used multiple availability zones to reduce the risk of downtime.

### 2. Deploying EC2 Instances

- **Implementation:**
  - Launched two **Amazon EC2 instances** using t3.medium type to balance performance and cost.
  - Installed **OpenVPN Access Server**, which simplifies VPN setup with a user-friendly web interface.
  - Assigned **Elastic IPs** to ensure consistent and static IP addresses for the VPN servers.
- **Challenge:** Balancing performance and cost.
  - **Solution:** Started with t3.medium instances for cost-effectiveness, with plans to upgrade if traffic demands grow.

### 3. Configuring Security

- **Implementation:**
  - Defined **Security Groups** to allow inbound traffic only on:
    - Port 1194 (UDP) for VPN connections.
    - Port 443 (TCP) for the OpenVPN web interface.
    - Port 51112 (TCP) for SSH access, restricted to trusted IPs only.
  - Generated encryption keys through OpenVPN's built-in tools to ensure secure client connections.
  - SSH port changed from 22 to 51112.
- **Challenge:** Preventing unauthorized access to the server.
- **Solution:**
  - Restricted SSH access to admin IPs only and periodically reviewed firewall rules.
  - SSH port changed from 22 to 51112.

#### 4. Storing Logs

- **Implementation:**
  - Configured **CloudWatch Logs** to automatically collect and store connection logs from the VPN servers.
  - Enabled a 30-day log retention policy to reduce storage costs.
- **Challenge:** Managing the volume of logs.
  - **Solution:** Applied filters to log only necessary events and set up retention policies to delete old logs.

## 5. Adding Load Balancing and Scaling

- **Implementation:**
  - Deployed a **Network Load Balancer (NLB)** to distribute VPN traffic across the two EC2 instances.
    - Configured the NLB to handle UDP traffic on port 1194.
    - Set up health checks to ensure the NLB routes traffic only to healthy instances.
  - Configured an **Auto Scaling Group:**
    - **Scale Out:** Add an instance when CPU utilization exceeds 70% for 5 minutes.
    - **Scale In:** Remove an instance when CPU utilization falls below 30% for 5 minutes.
- **Challenge:** Managing scaling costs while ensuring performance.
  - **Solution:** Set conservative scaling limits (minimum of 1 instance, maximum of 3) to balance cost and performance.

## 6. Testing and Deployment

- **Implementation:**
  - Conducted connectivity tests by connecting to the VPN from laptops and smartphones to verify functionality.
  - Tested internet access and speed through the VPN to ensure reliability.
  - Created a backup by generating an AMI (Amazon Machine Image) of the EC2 instances to simplify recovery.
- **Challenge:** Simulating real-world traffic for testing.
  - **Solution:** Used simple speed tests and client devices to validate functionality.

## Key Challenges and Simplified Solutions

Challenge	Solution
Single Point of Failure	Used multiple instances and a Network Load Balancer for redundancy.
Unpredictable Traffic Spikes	Configured Auto Scaling to dynamically add or remove instances as needed.
High Costs	Started with <code>t3.medium</code> instances and capped Auto Scaling to 3 instances.
Managing Logs	Used CloudWatch Logs with retention policies to minimize storage costs.
Balancing Security with Usability	Used Security Groups and restricted SSH access while simplifying user setup.

Cloud Instance Cost Comparison Table

Provider	Instance Type	vCPUs	Memory (GiB)	Hourly Cost (\$)	Monthly Cost	Key Features
AWS	t3.medium	2	4	0.0416	30.37	Great for occasional use; saves money when not always running.
Azure	B2s	2	4	0.05	36.5	Good for everyday tasks; handles extra work when needed.
Google Cloud	e2-medium	2	4	0.037	27.01	Affordable option for light to medium workloads.