1. On-Premise Component (Simulated)

1.1 Overview

The on-premise environment represents a legacy, vertically scaled application stack hosted on a single Linux server.
It serves as the source system during the hybrid migration phase and maintains secure, controlled connectivity with AWS.

This design reflects a common real-world scenario where web, application, and database tiers coexist on a single host, creating operational and scalability constraints that motivate migration to a cloud-based, horizontally scalable architecture.

1.2 On-Premise Network Design

Network Configuration

- **On-premise CIDR:** `192.168.10.0/24`
- **Application server IP:** `192.168.10.10`
- **Default gateway (firewall/router):** `192.168.10.1`

Design Principle

The application server is intentionally unaware of AWS, VPNs, or routing logic.
All non-local traffic is forwarded to the default gateway, which is solely responsible for routing decisions and VPN handling.

This separation of concerns mirrors real enterprise environments, where application hosts remain network-agnostic and connectivity is enforced at the network edge.

1.3 On-Premise Application Stack

Operating System

- Ubuntu Linux (LTS)

Services

- Apache HTTP Server
- Java web application (Tomcat / Joget)
- MySQL database (legacy, later migrated to Amazon RDS)

This represents a single-node, vertically scaled architecture with tightly coupled components. The lack of horizontal scalability and high availability is a primary driver for migrating application and database layers to AWS-managed services.

1.4 Firewall and Security Controls

The firewall/router performs dual responsibilities:

- Network gateway for the on-premise subnet
- VPN termination point (Customer Gateway)

Firewall Rules (Conceptual – UFW / iptables)

```
# Allow internal SSH
ufw allow from 192.168.10.0/24 to any port 22

# Web traffic
ufw allow 80
ufw allow 443

# MySQL – internal only
ufw allow from 192.168.10.0/24 to any port 3306

# VPN (IPsec)
ufw allow 500/udp
ufw allow 4500/udp

ufw enable
```

Security Rationale

- No public exposure of MySQL or internal services
- Explicit, port-based access control
- Least-privilege enforcement at the network edge
- VPN traffic restricted strictly to required IPsec ports

1.5 Hybrid Connectivity – Site-to-Site VPN (Conceptual)

A Site-to-Site IPsec VPN is used to establish secure connectivity between the on-premise network and AWS.

Rationale for Site-to-Site VPN

- Encrypted communication over the public internet
- Private routing between internal CIDR ranges
- No requirement to expose applications or databases publicly
- Industry-standard approach for hybrid proof-of-concept deployments

---

1.6 On-Premise VPN Gateway (OpenSwan / strongSwan)

The on-premise firewall/router is simulated using a Linux system running OpenSwan or strongSwan, providing IPsec Site-to-Site VPN functionality.

Technology Choice Justification

- Industry-standard IPsec implementation on Linux
- Supports IKEv1 and IKEv2
- Fully compatible with AWS Site-to-Site VPN
- Lightweight and appropriate for PoC environments

In production environments, this role is typically fulfilled by a hardware firewall (e.g., FortiGate, Palo Alto, pfSense).
Using OpenSwan/strongSwan accurately models the same functional behavior for this assessment.

1.7 IPsec VPN Tunnel Structure

A Site-to-Site IPsec tunnel is established between:

Customer Gateway (CGW)

- On-premise firewall/router running OpenSwan
- Identified by a public IP address

Virtual Private Gateway (VGW)

- AWS-managed VPN endpoint
- Attached to the target VPC

Tunnel Characteristics

- Encrypted transport over the public internet
- Private IP communication between defined CIDR ranges
- Network-level (Layer 3) connectivity, not application-level tunneling

---

1.8 Tunnel Behavior (High-Level)

Phase 1 – IKE

- Establishes a secure control channel
- Authenticates peers using a pre-shared key

Phase 2 – IPsec

- Encrypts traffic between:
    - On-premise CIDR: `192.168.10.0/24`
    - AWS VPC CIDR: `10.0.0.0/16`

Only traffic matching these CIDR ranges is permitted to enter the VPN tunnel.

1.9 Routing Integration (Critical Design Aspect)

- On-premise firewall routes AWS CIDR traffic into the IPsec tunnel
- AWS VPC route tables forward on-premise CIDR traffic to the VGW
- Applications remain completely unaware of the VPN

The VPN operates strictly at Layer 3 and does not perform load balancing, NAT, or application-level routing.

1.10 Traffic Flow: On-Premise → AWS

1. The application sends traffic to an AWS private IP (EC2 or RDS).
2. The operating system forwards non-local traffic to the default gateway.
3. The firewall matches the AWS CIDR route.
4. The packet is encrypted and sent through the IPsec tunnel.
5. The AWS VGW decrypts the packet.
6. The VPC router delivers traffic to the target subnet and resource.
7. Return traffic follows the same path in reverse.

1.11 Configuration Management Automation

A lightweight Bash script is used to automate and enforce the on-premise baseline configuration.

```
#!/bin/bash
set -e

apt update -y
apt install -y apache2 openjdk-11-jdk

systemctl enable apache2
systemctl start apache2

sed -i 's/ServerTokens OS/ServerTokens Prod/' \
  /etc/apache2/conf-available/security.conf

systemctl reload apache2
```

Automation Rationale

- Idempotent and safe to re-run
- Reduces configuration drift caused by manual changes
- Appropriate for proof-of-concept scope
- Easily extensible to Ansible or CI/CD pipelines