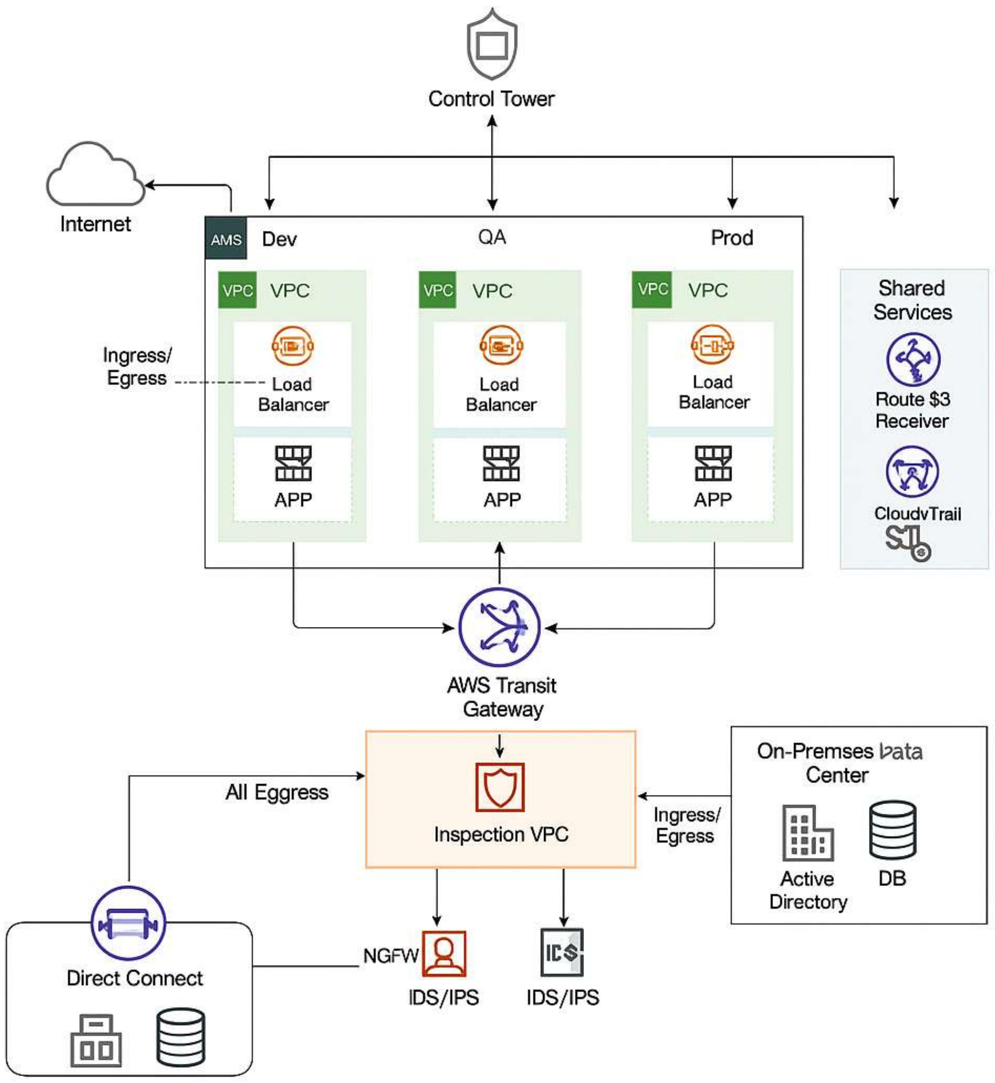
# Architecture Design Document

### Requirements Traceability Table

| **Business Requirement** | **Addressed In** |
|---|---|
| Migration Scope & Timeline | Intro, Network Architecture, DR/HA |
| Architecture & Network Design | Network Architecture, Diagrams, IaC |
| Security & Compliance | Security & Compliance Section, Diagrams |
| Shared Services & Integration | Shared Services, DNS, Logging Sections |
| Performance & Cost Optimization | Performance Section, Edge Presence Update |
| Disaster Recovery (DR) and High Availability | DR/HA Section |
| Future Integration with Partner Organizations | Integration & Future-Proofing Section |

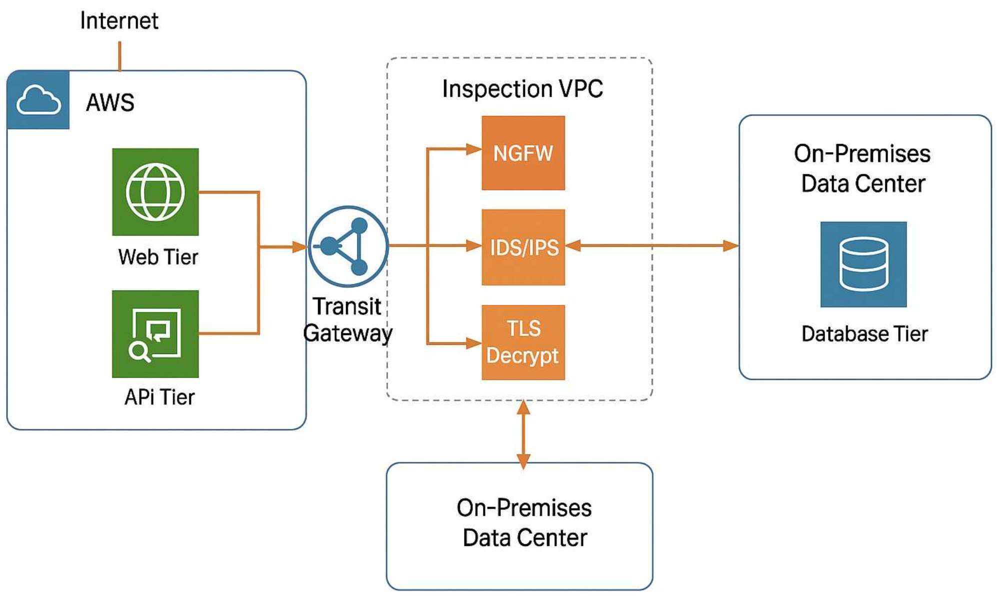
## 1. Network Architecture

I chose a multi-account AWS architecture (Dev, QA, Prod) using AWS Control Tower for governance and VPC isolation. Each account has its own VPCs, connected via AWS Transit Gateway. I favored TGW over VPC peering due to scalability, simplified routing, and easier policy enforcement for a large, evolving org. To bridge AWS with legacy on-prem systems, I implemented AWS Direct Connect as the primary link, with VPN as backup for high availability.



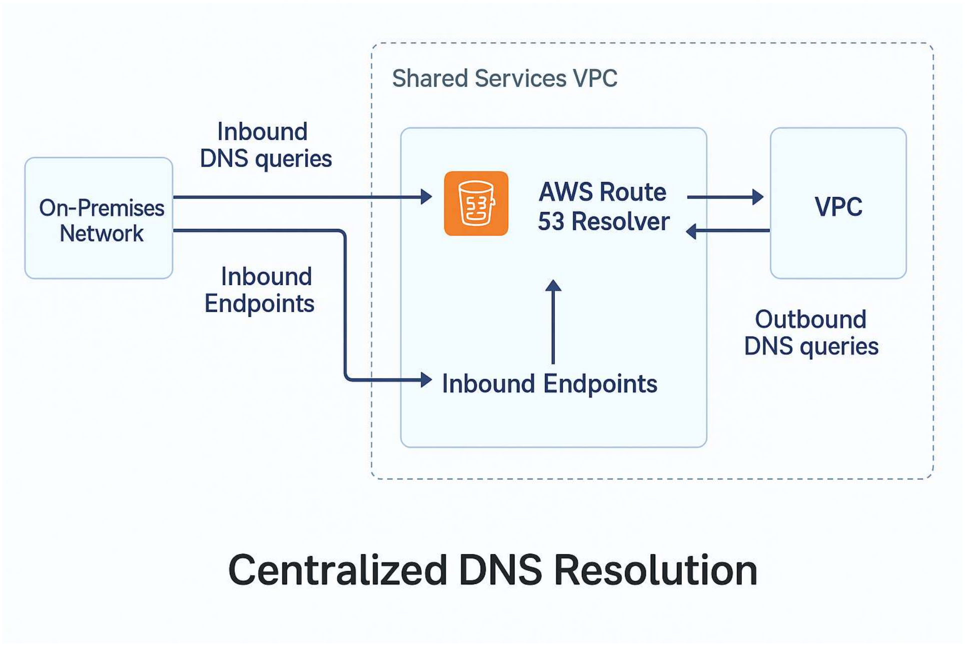
## 2. Centralized Security Inspection

My design routes all ingress/egress through a dedicated Inspection VPC. Here, I deploy NGFWs (next-gen firewalls) for TLS decryption, IDS/IPS, and deep packet inspection, so I can centrally manage policy and meet HIPAA/NIST mandates. This model also supports deep audit logging and enables a true zero trust posture.



## 3. DNS and Shared Services

To avoid DNS silos and misroutes, I use AWS Route 53 Resolver endpoints and forwarding rules, letting on-prem and AWS workloads resolve each other consistently. I integrate with Managed AD and centralize DNS/logging in a Shared Services account. This reduces operational friction and ensures audit trails.



## 4. Disaster Recovery and High Availability

I provide for cross-region DR using AWS Backup and (where migrated) Aurora Global Databases. Immutable infrastructure (via Terraform) means I can quickly rebuild or fail over environments. Critical workloads use multi-AZ deployment, and my connectivity is Highly Available by design (DX + VPN).

## 5. Security and Compliance

Every part of my design enforces encryption in transit (TLS 1.2+) and at rest (KMS with customer-managed keys). IAM is federated with SSO and SAML, with least privilege and strong role separation. All activity is logged centrally and integrated with GuardDuty, Inspector, and SIEM for audit readiness.

## 6. Performance and Cost

To reduce cost and latency, I keep the highest-volume DBs on-premises and optimize with caching (CloudFront), VPC interface endpoints, and compression. My routing and endpoint strategy reduces NAT usage and data egress, which is essential for both performance and spend.

To reduce latency and improve security for end-users, I deploy AWS CloudFront as a content delivery network, caching static and dynamic content at AWS edge locations close to user populations. Web Application Firewall (WAF) rules are enforced at these edge sites as well, blocking malicious traffic before it ever reaches the core VPC. This edge strategy dramatically lowers round-trip times for global users and enhances both performance and protection.

## 7. Integration and Future-Proofing

The architecture supports easy integration with future partner organizations via PrivateLink, TGW attachments, and Service Catalog/GitOps pipelines for self-service VPC creation. MuleSoft and other middleware can connect securely through dedicated, segmented VPCs.