**High-Level Plan: Extending SQL Always On AG to AWS (Single-Subnet Design)**

**Author:** tgibson  
**Date:** August 24, 2025

**Executive Summary**

We are extending our on-premises Windows Server Failover Cluster (WSFC) with SQL Server Always On Availability Groups (AG) into AWS. The goal is to establish a secure, production-ready foundation in AWS while keeping workloads private, minimizing downtime, and aligning with best practices.

Terraform has been used to codify and automate the deployment of all AWS infrastructure components. This ensures consistent, repeatable builds, cost tracking through tags, and simplified teardown when no longer needed.

**1. Preparation – On-Prem and AWS Prerequisites**

* **On-Premises**
  + Confirm existing SQL Server AG is running on WSFC with Node Majority quorum.
  + Validate cluster health and establish a backup of all databases.
  + Ensure Active Directory DNS and time synchronization are healthy (required for WSFC integration).
* **AWS**
  + Terraform IaC repository prepared with modules for VPC, security groups, IAM, ALB, ASG, RDS, and S3.
  + AWS account with IAM permissions to provision networking, EC2, RDS, and supporting services.
  + Plan for Site-to-Site VPN (initial connectivity) with optional Direct Connect in future.

**Why:** Preparation ensures the AWS environment can integrate into the existing cluster with minimal rework.

**2. Network Setup – Hybrid Connectivity**

* **VPC** – 10.0.0.0/16 with DNS enabled (critical for WSFC/AG listeners).
* **Subnets** –
  + Public (10.0.0.0/24, 10.0.1.0/24) across 2 AZs for ALB + NAT.
  + Private App (10.0.10.0/24, single AZ) for WSFC/SQL nodes.
  + Private DB (10.0.20.0/24) for PostgreSQL (exercise requirement).
* **Routing** – NAT Gateway in public subnet provides egress for private workloads.
* **Connectivity** – VPN will provide secure on-prem ↔ AWS traffic. Direct Connect can later replace or supplement it.
* **Security Groups** –
  + ALB SG: inbound 80/443 from world → egress to App SG.
  + App SG: inbound 80 from ALB SG → egress 5432 to DB SG.
  + DB SG: inbound 5432 only from App SG.
  + WSFC extension: allow TCP 1433, 5022, 3343, RPC/DCOM ranges between nodes.

**Why:** Design balances security (private workloads, least-privilege SGs) with functionality (SQL, WSFC, AG traffic).

**3. AWS VMs & SQL AG – Build and Configuration**

* **Sizing:**
  + EC2 instances sized for SQL AG nodes (e.g., m5.xlarge) depending on workload.
  + Terraform ASG module provisions Linux instances for web/app workloads.
* **Build Steps:**
  + Launch Windows EC2 instances in private app subnet via Terraform (future Windows module).
  + Install SQL Server, join to domain, and configure AG replica.
  + Use SSM Session Manager for admin access (no SSH/RDP exposure).
* **Configuration:**
  + Place both AWS SQL nodes in single subnet (per design requirement).
  + Ensure AG endpoint TCP 5022 is reachable.

**Why:** Aligns with scenario constraints while ensuring cluster nodes remain private and manageable.

**4. Cluster Extension & Quorum Management**

* **Current State:** On-prem cluster uses Node Majority.
* **Adjustment:** Move to **Node + File Share Witness (FSW)** for better resiliency.
* **FSW Location:** Can reside on-prem or in Azure/AWS S3-backed witness for hybrid quorum stability.
* **Vote Management:** Disable votes on secondary AWS nodes initially to prevent split-brain during testing; re-enable once stable.

**Why:** Proper quorum configuration is critical to avoiding cluster split-brain when stretching across geographies.

**5. Load Balancer & Listener**

* **Terraform ALB Module** provisions ALB across public subnets with HTTP listener.
* **For SQL Listener:**
  + In a single subnet, use AWS Network Load Balancer (NLB) or ALB with static IP mapping to direct traffic to active SQL replica.
  + Virtual Network Name (VNN) listener will be registered in DNS.
* **Trade-off:** No Distributed Network Name (DNN) option here due to single-subnet constraint.

**Why:** Ensures client connections fail over smoothly even in a stretched cluster.

**6. SQL Configuration – Joining AWS Replicas**

* **Steps:**
  + Join AWS SQL VMs to Active Directory.
  + Add them as replicas in the AG.
  + Configure automatic seeding or initialize with full backup/restore.
* **Validation:** Confirm replicas synchronize across VPN tunnel.

**Why:** Matches enterprise SQL AG configuration while extending replication into AWS.

**7. Testing – Failover and Validation**

* **Planned Tests:**
  + Manual AG failover to AWS replica.
  + Client connection validation through VNN listener.
  + Simulate VPN disruption to validate quorum behavior.
  + Application end-to-end testing through ALB.

**Why:** Demonstrates cluster resiliency, validates networking/security rules, and builds operational confidence.

**8. Migration Strategy – Cutover Approach**

* **Phased Migration:**
  1. Deploy AWS infra with Terraform (repeatable, idempotent).
  2. Extend AG with async replicas in AWS.
  3. Validate replication and failover.
  4. Schedule planned downtime.
  5. Promote AWS replica to primary.
  6. Redirect application traffic via updated listener/DNS.
* **Downtime Goal:** Minutes, limited to final cutover window.
* **Rollback:** Fail back to on-premises primary if issues arise.

**Why:** Minimizes risk and ensures business continuity.

**Design Choices & Trade-Offs**

* **Single Subnet Constraint:** Simplifies DNS/listener setup, but reduces multi-AZ resiliency. Future extension to 2 subnets possible.
* **NAT Gateway vs Instance:** NAT Gateway chosen for reliability, at slight cost increase.
* **SSM Access:** More secure than SSH/RDP; aligns with modern AWS practices.
* **Terraform Modules:** Provides modularity, reusability, and automated tagging for cost governance.

**Conclusion**

This design delivers a secure, production-ready AWS foundation for extending SQL AG into the cloud. Terraform codifies infrastructure for repeatability, and the plan balances immediate business needs with long-term flexibility.

Next steps: conduct joint testing with IT and DB teams, finalize quorum placement, and prepare migration schedule.