Here's a **comprehensive explanation of multiple ways to migrate SQL Server databases between Azure and AWS** — in both directions (Azure ➔ AWS and AWS ➔ Azure) — including **methods, tools, steps, advantages, and considerations**.

## 🌐 1. Common Migration Scenarios

| Migration Direction | Source | Target |
|---|---|---|
| Azure ➔ AWS | Azure SQL VM / Azure SQL Database / Managed Instance | Amazon RDS for SQL Server / Amazon EC2 (SQL Server VM) |
| AWS ➔ Azure | Amazon RDS / EC2 SQL Server | Azure SQL VM / Azure SQL Database / Managed Instance |

## 🧭 2. Migration Strategy Overview

| Strategy | Suitable For | Downtime | Notes |
|---|---|---|---|
| **Backup & Restore** | Small to medium DBs | Moderate | Simple and native SQL method |
| **Detach/Attach** | On-prem style VMs | High | Not supported for Azure SQL DB |
| **Import/Export (BACPAC)** | Schema + Data | Moderate | Works for migrating to Azure SQL DB |
| **Transaction Log Shipping** | SQL VM (IaaS) | Low | Useful for cutover migrations |
| **Database Mirroring / AlwaysOn AG** | SQL VM ↔ SQL VM | Very Low | Requires Enterprise Edition |
| **Replication** | Selected tables | Low | Works across clouds |
| **Cloud Migration Services** | Managed automation | Low–Moderate | Best for hybrid migrations |
| **Data Migration Assistant (DMA)** | Schema compatibility | N/A | Used for pre-checks |
| **Database Migration Service (DMS)** | Full / Ongoing replication | Low | Azure DMS or AWS DMS automates the process |

## 🚀 3. Azure ➔ AWS Migration Methods

### Method 1: Native Backup and Restore

**Steps:**

1. Take a **.bak backup** of your database in Azure SQL VM.
2. Copy it to **Amazon S3** or directly to **AWS EC2 instance** using tools like **AzCopy** or **AWS CLI**.
3. Restore it on:
   - SQL Server on EC2 (RESTORE DATABASE ... FROM DISK)
   - Amazon RDS (via **S3 bucket restore**)

**Pros:**

- Simple and fast for single databases.
- No special configuration needed.

**Cons:**

- Not applicable for Azure SQL Database (PaaS).

### Method 2: Export BACPAC and Import

**Use when:** Migrating **Azure SQL Database → Amazon RDS**

**Steps:**

1. Export a .bacpac file from Azure SQL Database using SSMS or Azure Portal.
2. Upload it to **Amazon S3**.
3. Import it into **Amazon RDS SQL Server**.

**Pros:**

- Easy GUI-driven process.
- Schema + Data both migrated.

**Cons:**

- No logins, jobs, or server-level objects are included.

**Method 3: Azure DMS to AWS EC2 / RDS**

**Steps:**

1. Use **Azure Database Migration Service (DMS)** to export data to on-premise or EC2 SQL instance.
2. Import into AWS target DB.

**Pros:**

- Reliable for large-scale migrations.
- Minimizes downtime.

**Cons:**

- Need VNet + IAM configurations.

**Method 4: AWS Database Migration Service (AWS DMS)**

**Steps:**

1. Configure Azure SQL DB (or VM) as the **Source Endpoint**.
2. Configure Amazon RDS or EC2 SQL as the **Target Endpoint**.
3. Create a **Replication Instance** in AWS DMS.
4. Choose between **Full Load** or **Full + Ongoing Replication**.
5. Validate and cutover.

**Pros:**

- Low downtime migration.
- Continuous sync supported.

**Cons:**

- Requires network peering between Azure and AWS.
- Setup slightly complex.

**Method 5: Log Shipping / AlwaysOn AG**

**Use when:** Migrating **SQL VM → SQL VM**

**Steps:**

1. Configure log shipping between Azure SQL VM and AWS EC2 SQL VM.
2. Continuously apply logs on AWS.
3. During cutover, restore tail log and switch over.

**Pros:**

- Very low downtime.
- Proven enterprise method.

**Cons:**

- Not supported for PaaS databases.

**☁ 4. AWS ➔ Azure Migration Methods**

**Method 1: Backup and Restore**

**Steps:**

1. Backup AWS RDS or EC2 SQL database.
2. Upload .bak to Azure Blob Storage.
3. Restore on Azure SQL VM or Managed Instance.

**Pros:**

- Simple for VM-based servers.

**Cons:**

- RDS backups require export to S3 first.

**Method 2: Export/Import (BACPAC)**

**Use when:** Migrating RDS → Azure SQL Database

**Steps:**

1. Export database to .bacpac from RDS or EC2 SQL.
2. Upload to Azure Blob Storage.
3. Import using Azure Portal or SSMS.

**Pros:**

- Schema + Data included.
- Works for PaaS targets.

**Cons:**

- No jobs, logins, or linked servers.

**Method 3: AWS DMS → Azure SQL**

**Steps:**

1. Set up **AWS DMS** with RDS as source, Azure SQL DB as target.
2. Configure endpoints + replication instance.
3. Perform **ongoing replication**.
4. Validate and switch traffic.

**Pros:**

- Automated and low downtime.

**Cons:**

- Complex initial setup.

**Method 4: Azure DMS (Database Migration Service)**

**Steps:**

1. Set up **Azure DMS** and select AWS RDS / EC2 as source.
2. Choose Azure SQL DB or MI as target.
3. Perform schema + data migration.

**Pros:**

- Microsoft-supported tool.
- Detects compatibility issues.

**Cons:**

- Requires network connectivity between clouds.

**Method 5: Replication**

**Steps:**

1. Configure AWS SQL Server as Publisher.
2. Configure Azure SQL VM as Subscriber.
3. Sync data until final cutover.

**Pros:**

- Continuous synchronization.
- Low downtime.

**Cons:**

- Schema changes require reinitialization.

### 🛠 5. Tool Comparison Summary

| Tool / Method | Direction | Downtime | Complexity | Supports PaaS | Comments |
|---|---|---|---|---|---|
| Backup/Restore | Both | Medium | Low | ❌ | Simple for VM migrations |
| BACPAC Import | Both | Medium | Low | ✅ | For Azure SQL DB |
| AWS DMS | Both | Low | Medium | ✅ | Supports continuous sync |
| Azure DMS | Both | Low | Medium | ✅ | Cloud-native automation |
| Log Shipping | Both | Low | Medium | ❌ | For SQL VM migrations |
| Replication | Both | Low | High | ❌ | Selective tables |
| AlwaysOn AG | Both | Very Low | High | ❌ | Enterprise only |

### 🧩 6. Key Considerations

- **Authentication:** Migrate SQL logins manually using sp_help_revlogin.
- **Jobs & Maintenance:** Export SQL Agent jobs separately.
- **Network Connectivity:** Configure VNet ↔ VPC peering or use VPN Gateway.
- **Licensing:** Use Bring-Your-Own-License (BYOL) or pay-as-you-go model.
- **Downtime Planning:** Always perform test cutover using a pilot DB first.