Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

Here are **50+ frequently asked interview questions** (with detailed example answers) focused on **Azure Database (DB) Migration**.

Azure DB Migration — Interview Questions & Detailed Answers

1. What is Azure Database Migration Service (DMS)?
Answer:
Azure DMS is a fully managed service that enables seamless migration of on-premises and other cloud database workloads to Azure with minimal downtime. It supports both offline (cutover) and online migrations. DMS scans the source databases, identifies compatibility issues, migrates the schema, data, and optionally applies continuous data replication for minimal service interruption.
Use cases: SQL Server → Azure SQL Database / SQL Managed Instance; MySQL/PostgreSQL migrations; even heterogeneous migrations.

2. What are the different target types for SQL DBs in Azure? (e.g. Azure SQL DB, Managed Instance, SQL Server on VM)
Answer:
Some common options:

- **Azure SQL Database (PaaS single database or elastic pool)** — fully managed, good for modern cloud apps, scales elastically.
- **Azure SQL Managed Instance** — provides near-full SQL Server compatibility including features not available in single Azure SQL DB (cross-DB queries, linked servers, etc.).
- **SQL Server on Azure VM (IaaS)** — you manage the OS, patching, backups etc. Good when you need full control or have features not yet in PaaS.

Consider trade-offs: overhead, responsibilities (patching, HA), cost, compatibility.

3. What are the migration strategies (lift & shift, replatform, refactor, rebuild) and when would you use each?
Answer:

- **Lift & shift (Rehost):** Move the existing DB/application almost as-is, e.g. moving from on-prem SQL Server VM to SQL Server on Azure VM or Managed Instance. Minimal code change; fastest.
- **Replatform:** Slight changes to take advantage of cloud features without completely rewriting—e.g. moving to Azure SQL Database PaaS, or changing storage/design for scale.
- **Refactor:** More substantial changes to redesign parts to use cloud-native services (e.g., breaking monolith, using serverless, or using Cosmos DB for some components).
- **Rebuild:** Full rewrite, for example when legacy tech is unsustainable, or moving to microservices architecture.

Choice depends on cost, risk, time, required downtime, compatibility, business value.

4. How do you assess whether a database is ready for migration? What tools are used?
Answer:
Assessment involves multiple dimensions:

- **Schema compatibility**: unsupported features in target; deprecated or discontinued features in Azure SQL / Managed Instance.
- **Feature usage**: e.g. CLR, SQL Agent jobs, cross-database queries, DB scoped credentials, etc.
- **Performance and capacity**: current CPU, memory, I/O, storage; workload patterns; peak loads.
- **Dependencies**: other databases, linked servers, applications, scheduled jobs, SSIS, etc.
- **Network, latency & connectivity**: between on-prem and Azure, or between apps and DB.

Tools:

- **Data Migration Assistant (DMA)** — checks compatibility, identifies feature-gaps; suggests remedial changes.
- **Azure Migrate: Database Assessment**
- **Azure DMS – pre-migration checks**
- Performance monitoring tools (on-prem) to collect metrics.

5. Describe the migration process end-to-end. What are the steps?

Answer:

Typical end-to-end process:

1. **Discovery & Inventory**: Catalog databases, versions, dependencies, workloads.
2. **Assessment**: Use compatibility tools to find risks, estimate cost, performance, required Azure target.
3. **Planning strategy**: Choose target platform (SQL DB, MI, VM), migration approach (offline vs online), decide on downtime, schedule, rollback plan.
4. **Provisioning target**: Set up Azure DB, network, security, required storage, compute.
5. **Schema migration**: Migrate schema, user logins, jobs, etc.
6. **Data migration**: Move data via DMS, BACPAC, backup/restore, transactional replication, etc.
7. **Testing**: Functional, performance, integration, latency, failover etc.
8. **Cutover / Go-Live**: Switch production to target.
9. **Post-migration operations**: Monitoring, tuning, security, backups, cost optimization, clean-up, decommission on-prem.

6. What is the difference between offline and online migration, and what are their trade-offs?

Answer:

- **Offline migration**: you stop writes to source at some point, do a bulk data move, switch-over. Simpler, less complex, but involves downtime.
- **Online migration**: allows source to stay live while data is continuously replicated, then final sync and cutover with minimal downtime. More complex and may cost more; risk of data drift, network latency issues, etc.

Trade-offs: required downtime, complexity, cost, risk.

7. How do you minimize downtime during database migration?

Answer:

Some strategies:

- Using **Azure DMS** with continuous data replication plus a final cutover.
- **Transactional replication** (from SQL on-premises to Azure SQL Managed Instance) if supported.
- Ensuring data is preloaded and pre-synchronized before switchover.
- Using offline schema changes where possible ahead of time.
- Planning migration during low-traffic windows.
- Leveraging staging environment or shadow writes.

8. What are some common challenges/issues during DB migration? How do you mitigate them?

Answer:

Challenges:

- Compatibility (unsupported features or deprecated SQL constructs).
- Performance differences (latency, I/O throughput).
- Network bandwidth constraints.
- Data size and growth.
- Ensuring data consistency / integrity.
- Handling large transactions / log truncation.
- Security / compliance: data encryption, access, audit.
- Downtime constraints.
- Application dependencies.

Mitigations:

- Thorough assessment; use DMA to find issues early.
- Prototype / pilot migrations.

- Use compression or change data capture to reduce data transfer.
- Cable bandwidth improvements / secure network links (ExpressRoute or VPN).
- Parallelization of migration tasks.
- Good rollback plan.

9. What is Azure Migrate? What are its components?
Answer:
Azure Migrate is a hub/toolset from Microsoft that helps with discovering, assessing, and migrating workloads to Azure. Components include:

- **Server Assessment**: for on-prem servers, assessing readiness, cost, sizing.
- **Server Migration**: for migrating VMs, physical servers.
- **Database Assessment** / **Database Migration** tools.
- **App Service Migration Assistant**: for web apps.
- **Replication, dependency mapping**: seeing what components, servers communicate with each other to plan.

10. How do you decide between SQL Managed Instance vs Azure SQL Database vs SQL Server on VM?
Answer:
Consider criteria like:

Factor	**SQL Managed Instance**	**Azure SQL DB**	**SQL Server on VM**
Feature compatibility	Very high — supports many SQL Server features not in Azure SQL DB	Some limitations (no cross-db, instance-level features)	Full SQL Server feature set
Management overhead	Low (Microsoft manages infrastructure)	Lowest	Highest (you manage OS, patching, backups etc.)
Cost	Medium to high	Depends on tier; can be cheaper if smaller workloads	Highest, because you manage VM + OS + SQL license etc.
Scalability & elasticity	Good	Excellent (for PaaS)	Limited by VM size / scaling complexity
Network latency / isolation	Good with VNET integration	Good	Good if VM in Azure but more overhead
Use existing investments / migration complexity	Easier for lift-and-shift to MI or VM; more changes needed for Azure SQL DB	Good if applications can work within its constraints	Easiest to migrate as-is

11. What is a bacpac file and how is it used in migration? What are its limitations?
Answer:
A **BACPAC** is a logical export of database schema + data (table data) from SQL Server, Azure SQL DB or Managed Instance. It's stored in Azure Blob or local storage. You can import it to Azure SQL DB / MI or to SQL Server.
Use in migration: For smaller DBs or when downtime is acceptable; you export and import.
Limitations:

- No logins, SQL Agent jobs, log stream, etc.
- Data size can make BACPAC export/import slow.
- Does not preserve certain server-level configurations or features.
- For large databases, may not be practical.

12. What about backup & restore? Can you restore SQL Server backups (.bak) to Azure SQL Database?
Answer:

- You **cannot** restore a .bak directly into Azure SQL Database (single or elastic). Azure SQL DB doesn't support that sort of file-based restore.
- You *can* restore .bak files to **SQL Managed Instance** or **SQL Server on VM** in Azure. For Managed Instance, you can place the .bak in Azure Blob Storage or use backup share.
- For Azure SQL DB, you'd use BACPAC or DMS to migrate data.

13. How does networking affect DB migration (on-prem ↔ Azure)? What about connectivity options?
Answer:
Networking considerations:

- **Latency & bandwidth**: migrating large volumes require high bandwidth; opt for ExpressRoute for large, consistent throughput.
- **Connectivity type**:
  - VPN (site-to-site) — simpler, lower throughput.
  - Azure ExpressRoute — private connection, more reliable, lower latency.
- **Security**: network isolation, secure channels, firewall rules, private endpoints.
- **Subnet / VNet design**: target DBs in the right virtual network, ensure proper network security group (NSG) settings.
- **DNS, routing, hybrid connectivity (if needed).**

14. What is Azure Site Recovery, and how is it used in context of migrations?
Answer:
Azure Site Recovery (ASR) is primarily a disaster recovery / replication service — allowing you to replicate VMs or physical servers to Azure and failover in case of disaster.

In migration: ASR can be used to replicate VMs for migration purposes (lift & shift). You replicate the on-prem servers to Azure, then failover to Azure (cutover), thus using ASR as a migration tool in some lift & shift scenarios.

15. How do you handle user logins, security principals, and permissions during migration?
Answer:

- Script out SQL Server logins, roles, and permissions. Use tools like sp_help_revlogin to migrate logins with correct SIDs (if needed).
- For Azure SQL DB / MI, you can use Azure Active Directory authentication for better centralized identity.
- Map orphaned users if the database's users do not map to logins.
- Ensure least privilege access, enforce encryption, audit.
- Review server level roles that might not exist in PaaS (e.g. sysadmin etc.).

16. What are key security and compliance considerations when migrating to Azure DB services?
Answer:

- Data encryption: at rest and in transit (TLS); Azure Transparent Data Encryption (TDE), Always Encrypted, etc.
- Auditing and logging: enable Azure SQL auditing, diagnostic logs, send to Log Analytics or SIEM.
- Identity and access control: integrate with Azure AD; use RBAC.
- Network security: private endpoints, virtual network service endpoints, network rules.
- Regulatory compliance: GDPR, HIPAA, PCI etc; ensure the Azure region selected supports required compliance.
- Backup, retention policy; disaster recovery.

17. How do you monitor performance after migration? What tools are used?
Answer:

- **Azure Monitor** for metrics (DTU / vCore usage, IO, CPU, memory, waits).
- **Azure SQL Analytics / Intelligent Insights** for SQL DB / MI to get deeper insights.
- **Query Performance Insight** to find slow queries and bottlenecks.
- **SQL Insights** in portal / Azure Data Studio.
- **Application Insights** for application-level tracing.
- Regular maintenance tasks: index rebuilds, statistics, etc.

18. What is cost estimation / TCO (Total Cost of Ownership) consideration when migrating to Azure DB?
Answer:
Consider:

- Licensing costs: whether you can use Azure Hybrid Benefit, Software Assurance, etc.
- Compute + storage + backup costs, data egress, network costs.
- Performance tiers (vCore vs DTU vs managed instance options) → selecting right size.
- Operational costs: monitoring, maintenance.
- Future growth: scaling cost.
- Downtime costs during migration.

Use Azure TCO calculator, pricing calculator, Azure Cost Management tools, Azure Advisor for recommendations.

19. What is SQL Managed Instance's backup and restore model?
Answer:

- Managed Instance supports native backups and restores using .bak files (via URL / blob storage / shared SMB).
- Supports automated backups by Azure (full, differential, log backups) with retention configured.
- Point-in-time restore is supported (for Managed Instance).
- Geo-restore and long-term retention can be configured.

20. What features of SQL Server are *not* supported or limited in Azure SQL Database?
Answer:
Some limitations / unsupported items include:

- No SQL Server Agent (though Azure DB has elastic jobs / Azure Automation).
- Cross-database queries are limited in single database model.
- Certain extended stored procedures, certain system procedures might not be available.
- FILESTREAM / FILETABLE support is limited.
- Certain low-level features like WMI access, OS level operations not possible in PaaS.
- CLR assemblies are supported but with restrictions.

21. How do you migrate very large databases (terabytes) to Azure with minimal disruption?
Answer:
Strategies:

- Use **Azure DMS** in online mode, so data is replicated continually.
- Use **backup/restore** to blob storage followed by log shipping or syncing small deltas.
- Use **transactional replication** or **change data capture (CDC)** to capture changes while migrating main bulk.
- Use high throughput network / ExpressRoute; possibly data compression.
- Stage migration in phases; possibly seed data ahead of time during off-peak hours.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

22. What's the difference between Data Migration Assistant (DMA) and Data Migration Service (DMS)?
Answer:

- **DMA** is used for **assessment**: checking schema compatibility, feature parity, identifying blockers, performance issues. It helps plan the migration.
- **DMS (Database Migration Service)** is used to **execute** the migration (schema + data + optionally continuous syncing).

23. How do you deal with schema drift or ongoing schema changes in the source during migration?
Answer:

- Freeze schema changes during cutover period.
- Track schema changes (via scripts / version control) and apply them both to source and target or in the target before cutover.
- For continuous migrations, schedule windows for schema sync.
- Use tools or custom scripts to compare schemas and generate diffs.

24. What is "elastic pool" in Azure SQL Database and in what migration scenarios it is useful?
Answer:

- An **elastic pool** allows you to share resources (DTUs or vCores) among multiple databases. Good where you have many databases with varying or intermittent usage, so that individual databases don't need to be sized high for their peaks. This can reduce cost.
- Useful when migrating many small-medium sized DBs that are not always in heavy use.

25. What is Azure SQL Database Hyperscale, and when should you use it?
Answer:

- Hyperscale is a service tier for Azure SQL Database designed to support huge databases (multiple terabytes). It decouples storage from compute, enabling fast scaling of storage, better backup/restore, support very large datasets.
- Use it when your DB grows beyond standard limits, need fast scale, large storage volumes, faster restore times, etc.

26. Describe what cross-region or geo-replication options are available for Azure SQL DB / Managed Instance.
Answer:

- **Active Geo-Replication**: Available for Azure SQL Database (single or pooled) to create readable secondaries in different regions.
- **Auto-Failover Groups**: For SQL DB or Managed Instance, to provide high availability and disaster recovery across regions.
- **Geo-Restore**: Restore backups to a different region.
- **Zone Redundant High Availability (HA)**: Within same region, multiple availability zones.

27. How do you ensure data consistency / integrity during migration?
Answer:

- Use checksums/hash totals before & after migration.
- Use tools that support transactional migration or change data capture.
- Do validation scripts, row counts, checks of critical data.
- For online migrations, final sync / snapshot to ensure all transactions are migrated.
- Monitor and compare application behavior pre- and post-migration.

28. What approaches can be used for heterogeneous migrations (e.g. Oracle, MySQL, PostgreSQL → Azure SQL / Azure Data platform)?
Answer:

- Use **Azure DMS**, which supports some heterogeneous migrations.
- Use schema conversion tools (e.g. SQL Server Migration Assistant [SSMA], or tools within Azure for schema conversion).

- Use ETL / data pipeline tools (like Azure Data Factory) for data movement & transformations.
- Custom scripts, maybe partially rewriting parts of apps.

29. What is network bandwidth / latency's effect, and how do you address network limitations?
Answer:

- Low bandwidth or high latency slows data transfer, increases migration time.
- To mitigate: use ExpressRoute, data compression, offline seeding (e.g. ship physical disks), use incremental sync, do heavy data transfers during low peak times.

30. How do you roll back a migration if something goes wrong?
Answer:
Have a rollback plan, e.g.:

- Keep the source system running until cutover and first period of validation.
- Maintain snapshots/backups so you can revert to prior state.
- Plan for dual-running systems, or read/write mirroring so you can fail back.
- Ensure clients/applications are configurable to point back to source DB if needed.

31. What is "Downtime SLA" and what should you aim for in a migration?
Answer:

- **Downtime SLA** is how much downtime is acceptable for the business during migration. Organizations may have strict SLAs (minutes, perhaps zero).
- Aim depends on business, but many migrations aim for minimal downtime (hours or less). For critical systems, "zero downtime" or near zero via tools like DMS online migration, transactional replication, etc.

32. How to handle large transaction log growth or long-running transactions during migration?
Answer:

- Monitor transaction log usage, ensure backups / truncation are happening.
- For large transactions, break them into smaller chunks.
- During bulk load, disable or manage indexes / constraints to speed up.
- Use minimal logging if possible.

33. What are the costs / trade-offs of choosing vCore vs DTU model (for Azure SQL Database)?
Answer:

- **DTU model**: simpler, predefined bundles of compute, storage, I/O. Easier to understand but less flexible in terms of resource granularity.
- **vCore model**: offers more transparency (e.g. specific cores, memory, storage), allows Hybrid Benefit, more scaling flexibility. Generally better for predictable workloads and larger databases.

34. How do you automate or script migration tasks?
Answer:

- Use PowerShell scripts / Azure CLI / Azure REST APIs to provision resources, manage roles, set up DMS, etc.
- Use ARM templates or Bicep for infrastructure as code (IaC) to define DB, networking, security.
- Use automation tools for backup, monitoring setup.

35. Describe performance tuning / optimization you might do post-migration.
Answer:

- Review execution plans to identify slow or missing indexes, or statistics that need update.

- Optimize queries that show degraded performance.
- Scale compute or storage as needed.
- Adjust server configuration (max degree of parallelism, memory grants etc.) if available.
- Partition large tables if needed.
- Use features like In-Memory OLTP if available and beneficial.

36. What is "Failover Group" in Azure SQL, and how is it useful?
Answer:

- Failover Groups allow you to manage geo-replication and automatic failover of a group of databases across regions. Useful in disaster recovery (DR) scenarios. Supports both Azure SQL DB and Managed Instance.

37. How do you handle secrets / credentials / connection strings securely?
Answer:

- Use **Azure Key Vault** to store secrets, certificates.
- Use Managed Identity for Azure services so that applications can access DBs without storing credentials.
- Ensure minimal access and rotate credentials.

38. What is Azure Data Factory and how does it help in database migration?
Answer:

- Azure Data Factory (ADF) is an orchestration / data integration service. It can be used to move / transform / load data between on-premises and cloud or between clouds.
- For migration: can be used for data copy (especially for large data sets), for incremental loads, or ETL / ELT tasks during migration.

39. What is "Managed Backup" in Azure SQL / Managed Instance?
Answer:

- Azure SQL DB and Managed Instance provide automated backups (full / differential / transaction log) managed by Azure.
- Retention configured via service settings.
- Support point-in-time restore.

40. How do you ensure high availability (HA) and disaster recovery (DR) for Azure DBs?
Answer:

- Use built-in HA features: zone awareness, redundant replicas, availability zones.
- Use geo-replication or failover groups for DR across regions.
- Use backups + point-in-time restore.
- For managed instance, ensure that SLA for availability zones is used.

41. What are the implications of data sovereignty and region selection?
Answer:

- Some regulations require data to be stored in specific geographic regions.
- Selecting correct Azure region is vital for compliance.
- Also consider latency: closer region to users is lower latency.
- Ensure region supports required services / features.

42. How do licensing / costs work when migrating SQL Server to Azure (Bring Your Own License / Hybrid Benefit)?
Answer:

- **Azure Hybrid Benefit** allows you to use existing SQL Server licenses (with Software Assurance) to reduce cost on Azure SQL DB / Managed Instance / VM.
- You may also pay for cores, storage separately.
- Be aware of license mobility, compliance, and whether features you use (e.g. certain enterprise features) require specific licensing.

43. What are some best practices when migrating multiple databases from on-prem to Azure?
Answer:

- Group by dependencies (migrate related databases together).
- Use elastic pools where beneficial.
- Standardize sizing, naming, security.
- Automate schema migration.
- Run pilot migrations.
- Monitor performance baseline before migration.

44. How do you deal with data drift or changes in source after migration?
Answer:

- Use continuous replication / change data capture.
- Freeze changes in source during cutover.
- Perform final synchronization.
- Use version control / schema comparison tools to detect drift.

45. What are Azure Private Endpoints / Service Endpoints, and why are they important for Azure SQL?
Answer:

- **Service Endpoints**: extend virtual network identity to Azure services, simplifying network access; provide secure access from VNet to PaaS services.
- **Private Endpoints**: map a specific Azure resource to a private IP in your VNet; ensures all traffic to the Azure SQL DB goes via the private link (avoiding exposure to public internet).

They enhance security and control, reduce exposure, help comply with network policy.

46. Explain how pricing tiers / service objectives work for Azure SQL Database.
Answer:

- Tiers define performance, availability, features. For example, Basic / Standard / Premium, or vCore model with service tiers like General Purpose, Business Critical, or Hyperscale.
- Higher tiers give better IO, more memory/CPU, more resilience (e.g. replica placement), features (e.g. zones, backups).

47. What are some Azure tools to measure network latency, connectivity, and performance?
Answer:

- Azure Monitor / Network Watcher: for monitoring network latency, packet loss.
- Azure ExpressRoute monitoring.
- Bandwidth measurement tools; network testing tools (e.g. ping, traceroute, custom tools).
- Azure Data Box / Import / Export for large data, to avoid network limitations.

48. How would you migrate a database that is used 24/7 (continuous updates), with minimal user impact?
Answer:

- Use **online / near-real-time data replication**, e.g. Azure DMS in online mode, transactional replication or change tracking / CDC.
- Preload bulk data in advance.
- Switch over during a scheduled low-traffic window.
- Use feature toggles to allow dual writes if necessary.

49. What is "shadow copy" or "dual writes" pattern and when is it used?
Answer:

- Dual writes / shadow writes: writing data to both source and target during migration period, so both stay in sync. Then cut over reads/writes to target.
- Used for migrations where downtime needs to be minimized, and application can be configured to write to both systems temporarily.

50. How do you test / validate a migration before going live?
Answer:

- Functional testing: ensure the application works correctly using the new DB.
- Performance testing: under expected load, test queries, check latency, throughput.
- Data validation: compare counts, checksums, sample data.
- Security testing: verify access, roles, encryption.
- Failover / disaster recovery testing.

51. What do you know about Azure SQL Database's performance tiers / purchasing models (vCore vs DTU)?
Answer:
(Overlap with earlier question #33) But more detail:

- **DTU (Database Transaction Unit) model** bundles CPU, memory, I/O into a unit; good for smaller/simple workloads and pre-defined resource caps.
- **vCore model** decouples CPU, memory, storage allowing more flexibility; supports Gen5, Hyperscale etc; allows Hybrid Benefit.

52. What is Hyperscale backup/restore behavior?
Answer:

- Hyperscale has faster backup/restore times: storage is decoupled; snapshots are near-instant (for metadata), actual data read occurs lazily.
- Read replicas can be used.

53. How do you use a staging or test migration to reduce risk?
Answer:

- Set up a non-production test or staging environment that mirrors production as much as possible.
- Do a full migration there, test everything (application behavior, performance, edge cases).
- Use that to uncover issues (compatibility, latency, permissions).
- Use test migrations to train team and refine scripts & processes.

54. How do you handle schema or feature differences in cross-platform migrations (e.g. Oracle → Azure SQL)?
Answer:

- Use schema conversion tools (like SSMA or equivalent).

- Map data types, functions, stored procedures etc that differ.
- Replace unsupported features with alternatives.
- Rewrite code segments; test thoroughly.

55. What are best practices for naming, organizational standards, tagging, and governance during migrations?
Answer:

- Use consistent naming for resources (databases, servers, groups).
- Use resource tags (owner, environment, cost center).
- Set up policy / governance (Azure Policy) to enforce compliance (naming, security, location).
- Use role-based access control (least privilege).
- Maintain good documentation of migration steps and topologies.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

Technical and frequently asked Azure Database Migration interview questions and answers — focused for **senior-level or architect roles**, with a technical depth in real-world migrations (SQL Server → Azure).

🔥 Top Azure DB Migration Interview Questions (Senior-Level)

1. What are the major migration paths from on-prem SQL Server to Azure? How do you choose between them?
Answer:
3 main options:

- **Azure SQL Database (PaaS)** – ideal for modern apps; lightweight, no SQL Agent, some limitations.
- **SQL Managed Instance (PaaS)** – full SQL Server feature set (e.g., cross-db queries, Agent); great for lift-and-shift.
- **SQL Server on Azure VM (IaaS)** – exact match of on-prem; use if legacy features needed.

Choosing depends on:

- Feature compatibility
- Downtime tolerance
- App dependencies
- Cost and maintenance requirements

2. How do you perform an online (minimal downtime) migration from on-prem SQL Server to Azure?
Answer:

- Use **Azure DMS in Online mode**
- Initial schema + data load
- Continuous data sync (CDC-based)
- Cutover with minimal downtime (app pointing to Azure)

Tools:

- DMS Online
- Transactional Replication (for Managed Instance)
- Custom sync via SSIS or ADF for complex cases

3. How do you assess migration readiness? Which tools do you use?
Answer:
Key tools:

- **Data Migration Assistant (DMA)** – schema compatibility + feature parity check
- **Azure Migrate: Database Assessment** – high-level readiness + TCO
- **Query Store / PerfMon** – performance data to size target instance

Check for:

- Unsupported features (e.g., SQL Server Agent jobs, CLR)
- Cross-database queries
- Linked servers
- Server-level objects (logins, jobs, credentials)

4. Describe the step-by-step database migration flow.
Answer:

1. **Assess** (DMA / Azure Migrate)
2. **Plan** (select Azure target: SQL DB, MI, VM)
3. **Provision** target (VNet, firewall, DB)
4. **Migrate Schema** (DMA or DMS)
5. **Migrate Data** (DMS, BACPAC, backup-restore)

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

6. **Test** (connectivity, performance)
7. **Cutover** (switch production to Azure)
8. **Post-Migration** (monitor, tune, secure)

5. What's the difference between a BACPAC and a .BAK file migration?
Answer:

- **BACPAC** = schema + table data (no logins, jobs). Best for small DBs.
- **.BAK** = full backup (used in Managed Instance/VM only)

Azure SQL Database **does not** support .BAK restores.

6. What are the limitations of Azure SQL Database compared to SQL Server?
Answer:

- No SQL Agent (use Elastic Jobs / Logic Apps)
- No cross-db transactions or 3-part naming
- Limited CLR support
- No FILESTREAM/FileTable
- No linked servers
- No access to system databases (master, msdb)

Solution: Use Managed Instance if you need these.

7. How do you migrate SQL Agent Jobs, linked servers, and logins?
Answer:

- Use sp_help_revlogin to migrate logins with correct SID
- Recreate SQL Agent Jobs manually or via script
- Linked Servers: reconfigure in target (or refactor if unsupported in SQL DB)

8. How do you secure a migrated Azure SQL database?
Answer:

- Use **Azure AD authentication** where possible
- Enable **TDE** (Transparent Data Encryption)
- Configure **firewall / Private Endpoints**
- Enable **Auditing**, **Vulnerability Assessment**
- Store secrets in **Azure Key Vault**
- Apply **RBAC**, **least privilege access**

9. What's the difference between Azure DMS and DMA?
Answer:

- **DMA** = Assessment tool (checks schema, feature compatibility)
- **DMS** = Migration engine (moves schema + data, supports online migration)

Use **DMA first**, then use **DMS** to migrate.

10. What is the difference between Azure SQL DB vCore vs DTU?
Answer:

- **vCore**: Flexible, transparent CPU/memory, supports Azure Hybrid Benefit
- **DTU**: Bundled CPU + memory + IOPS, good for simpler workloads

Use **vCore** for enterprise workloads, predictable scaling, and BYOL.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

11. How do you verify data consistency post-migration?
Answer:

- Compare **row counts**
- Run **checksums / hashes**
- Validate key business queries
- Use **Data Comparison tools** (e.g., Redgate, SSIS, custom SQL scripts)

12. How do you handle cross-database dependencies in Azure SQL DB?
Answer:

- **Azure SQL DB** doesn't support cross-db queries
    - Refactor into APIs or move to Managed Instance
- **SQL MI** supports cross-db and 3-part naming

13. How do you migrate large databases (1+ TB)?
Answer:

- Use **DMS online** or **.BAK to Managed Instance**
- If size is huge:
    - Use **Azure Data Box** (offline seeding)
    - Use **Transactional Replication + log shipping**
    - Compress and batch data movement

14. What's the purpose of Failover Groups in SQL MI / Azure SQL?
Answer:

- Provides **automatic geo-failover** for high availability across regions
- Supports **read/write and read-only endpoints**
- Ideal for **DR planning**

15. How do you design a rollback plan for a DB migration?
Answer:

- Keep source system online during pilot
- Final cutover only after validation
- Take full backups before switch
- Be ready to re-point app to source if issues arise
- Keep DNS TTL short to switch fast

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

Detailed **AWS database migration interview questions & suggested answers**, aimed especially at senior/architect roles. Covers tools, strategies, trade-offs, failure modes, real-world concerns.

AWS DB Migration — Interview Questions & Detailed Answers

1. What is AWS Database Migration Service (DMS)?
Answer:
AWS DMS is a fully managed service that helps migrate databases from on-premises, other clouds, or AWS sources to AWS targets. It supports homogeneous migrations (e.g. Oracle → Oracle) and heterogeneous (Oracle → Aurora PostgreSQL, etc.). It can do full data load plus continuous data replication to minimize downtime. It consists of replication instances, source & target endpoints, replication tasks, etc. (Abhay Singh)

2. What is the AWS Schema Conversion Tool (SCT)? When & why is it used?
Answer:
AWS SCT is a tool for converting database schema and database code objects (stored procedures, functions, views, triggers, etc.) from one engine to another (e.g. Oracle → PostgreSQL). It helps with heterogeneous migrations. You use SCT before migrating data when the source and target database engines differ; it generates conversion reports and lets you manually adjust unsupported items. (Abhay Singh)

3. What are the different migration types supported by DMS?
Answer:

- **Full load only**: Migrates the existing data at a point in time.
- **Change data capture (CDC)**: Captures ongoing changes from source and applies them to target.
- **Full load + CDC**: First bulk loads the existing data, then applies incremental changes until cut-over. (Abhay Singh)

4. How do you minimize downtime during database migration using AWS tools?
Answer:

- Use **Full load + CDC** so the bulk of data is migrated beforehand, and only final delta is synced at cutover.
- Use replication (via DMS) to keep source and target in sync.
- Pre-validate schema, indices, mapping ahead of cutover.
- Possibly use read-only replicas, dual writes (shadow writes), or blue/green deployment strategies.
- Plan cutover during low usage periods.

5. What are key components of AWS DMS and how do they work?
Answer:

- **Replication instance:** The managed compute instance that performs the data migration & replication tasks.
- **Endpoints**: Source & target endpoints include connection info, credentials, etc.
- **Replication task**: Defines what to migrate, mapping, any transformation, full/CDC type.
- **Subnet / VPC / Security configuration**: Ensures networking connectivity & security.
- **Monitoring & logging**: CloudWatch, task logs, events, etc. (Abhay Singh)

6. Explain homogeneous vs heterogeneous migrations. What are trade-offs?
Answer:

- **Homogeneous**: Source and target have the same engine (e.g. MySQL → MySQL). Schema conversion minimal; less risk.
- **Heterogeneous**: Different engines (e.g. Oracle → Aurora, SQL Server → PostgreSQL). Requires SCT for schema/code conversion; may have compatibility gaps or behavior differences.

Trade-offs:

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

- Heterogeneous migration takes more effort (schema, data types, functions).
- May require rewriting some code.
- Testing and validation more intensive.
- Potential performance or behavior differences post-migration.

7. How do you plan / assess readiness for an AWS database migration?
Answer:

- Inventory source databases: size, versions, features in use (e.g. stored procedures, triggers, custom types).
- Identify dependencies: apps, services, linked systems, connectivity.
- Use tools: AWS SCT to catch incompatible objects; DMS for test migrations.
- Evaluate network connectivity, bandwidth.
- Estimate storage, compute needed on target.
- Plan for security, compliance (encryption, access).
- Plan for rollback/failover.

8. What are limitations / common gotchas when using AWS DMS?
Answer:

- Not all schema objects are converted automatically—some code may need manual rewriting (functions, triggers, stored procedures) especially in heterogeneous migrations.
- DMS doesn't enforce all constraints during migration (e.g. primary/foreign keys may not be validated during full load, though they exist on target).
- Network latency & bandwidth can throttle speed.
- Large transactions or long-running queries may introduce latency or conflicts.
- Migration tasks can fail because of resource limits (replication instance size, storage, IOPS).
- Handling of LOBs (large objects) sometimes need careful setup.

9. Describe a full end-to-end database migration process to AWS.
Answer:

1. **Assessment**: inventory, tool usage (SCT, etc.), identify compatibility issues.
2. **Design**: target DB type (RDS, Aurora, EC2-hosted DB), instance class, storage, VPC, networking, security.
3. **Schema conversion (if needed)** via AWS SCT. Adjust & test.
4. **Set up DMS replication instance** + endpoints.
5. **Migrate schema & initial load** (Full load).
6. **Start CDC** to capture changes.
7. **Testing**: functional, performance, data integrity.
8. **Cutover**: switch application traffic to new DB, stop writes on source.
9. **Post-migration**: monitor, optimize, decommission old, deal with edge cases, backup, disaster recovery.

10. What are RDS vs Aurora vs EC2-hosted database targets? How to decide?
Answer:

- **RDS (Relational Database Service)**: Managed DB service, many engines (MySQL, PostgreSQL, SQL Server, Oracle, MariaDB). You don't manage OS, but you get patching, backup, scaling features.
- **Aurora**: AWS-built database compatible with MySQL/PostgreSQL; provides better performance, storage autoscaling, read replicas, faster failover.
- **EC2-hosted database**: You manage everything (OS, patching, backup). Used when you need full control, unsupported features, custom configurations.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

Decision factors:

- Required features & compatibility
- Operational overhead vs control
- Performance needs
- Cost (Aurora vs RDS vs EC2)
- Scalability, high availability

11. How do you ensure data consistency/integrity post migration?
Answer:

- Compare row counts, checksums/hash sums, sample data.
- Run business queries both in source & target and compare results.
- Use tools or custom scripts to verify that constraints, defaults, indexes exist and yield same performance.
- Use DMS logging to detect skipped rows or failures.
- Possibly have dual writes during cutover or freeze writes for short period.

12. How do you monitor & report progress of a migration using AWS DMS?
Answer:

- Use **CloudWatch metrics**: Replication lag, throughput, latency, CPU, memory usage of replication instance etc.
- Use **DMS task logs** (task history, error logs).
- Look at the DMS console: task status, table statistics (how many rows remain, etc.).
- Use alerts for failures, thresholds (lag > x minutes).

13. What about rollback strategies if the migration doesn't go as planned?
Answer:

- Keep source DB fully functional until cutover validated.
- Plan for re-pointing app to source if target is unstable.
- Keep backups on both sides.
- Possibly maintain a short "dual-write" or shadow period.
- Document rollback steps, ensure schema changes are reversible.

14. How do you handle schema drift—i.e. when source schema changes during migration?
Answer:

- Freeze schema changes during final migration window.
- Or synchronize schema changes both on source & target ahead of the final cutover.
- Track schema changes in version control.
- Use scripts or tools (SCT or other) to compare schema and apply diffs.

15. Security & compliance during migration—what to pay attention to?
Answer:

- Encryption in transit (TLS) and at rest (e.g. server-side encryption, KMS).
- Network security: VPC, subnet, security groups, possibly VPN or Direct Connect for on-prem connectivity.
- IAM policies—least privilege for those running migration tasks.
- Access to source/target DB credentials.
- Auditing & logging (CloudTrail, logging on DB).
- Data masking / anonymization if required.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

16. How do you handle large dataset migrations (hundreds of TBs)?

Answer:

- Use AWS Snowball / Snowmobile for seeding data offline.
- Then use DMS for continuous replication of changes (CDC).
- Optimize replication instance sizing, network bandwidth.
- Possibly compress data or partition migration.
- Perform test migrations on subsets.

17. What is AWS Snowball / Snowmobile and when would you use them?

Answer:

- Physical data transport services: Snowball (smaller) and Snowmobile (for exabyte scale).
- Used when network bandwidth is insufficient or transfer cost / time would be too high.
- Seed initial data via physical device, then sync changes over network.

18. What is AWS Application Migration Service (MGN)? How is it different from DMS?

Answer:

- AWS Application Migration Service (MGN) is for migrating **entire servers or applications** (OS, applications, attached storage) to AWS—lift & shift for servers.
- DMS is for database content migration.

19. How do you size the replication instance & target DB instance?

Answer:

- Look at source DB size, transaction volume (rates of change), number of tables, data types, size of LOBs, etc.
- Determine throughput needed (I/O, network).
- Consider memory, CPU to handle initial load + CDC.
- Also include buffer for overhead (e.g. applying transformations, network latency).

20. What performance optimization techniques can you apply for DMS migration tasks?

Answer:

- Tune replication instance size (upgrade if needed).
- Use multi-threaded/full-parallel loads where supported.
- Limit what you migrate initially (filter unnecessary tables, exclude data not needed).
- Minimize index overhead during full load (maybe disable non-clustered indices, create them after).
- Adjust table mappings, transformation rules carefully.
- Make sure source DB is tuned (logs, IOPS, read/write performance).

21. How do you deal with compatibility issues, e.g. stored procedures, UDFs, triggers, types when moving between different database engines?

Answer:

- Use SCT to identify incompatible code objects.
- Manually adjust or rewrite stored procedures/functions/triggers.
- Test behavior post-migration.
- For data types, map to appropriate equivalent types in target. Test for precision, scale, performance.

22. What are cross-region and cross-account migration considerations?

Answer:

- Network latency & bandwidth between source and target regions/accounts.

- Permissions and IAM roles (source account, target account).
- Possibly VPC peering, VPN, Direct Connect.
- Data egress costs.
- Ensuring encryption and other compliance across regions.

23. How does AWS DMS handle failures / errors in migration tasks and what retry/cleanup options are there?
Answer:

- DMS tasks can fail due to various issues (connectivity, permissions, network issues, resource limits).
- Logs provide errors; you can restart tasks or retry from failure point.
- DMS tracks which tables/partitions have completed, so you don't always need to start over.
- Task settings may allow skipping certain errors, or stopping on error, etc.

24. What is AWS Migration Hub and how does it relate to database migrations?
Answer:

- Migration Hub provides centralized tracking & visibility for application migration across AWS, including database migrations.
- It doesn't do the data transfer itself, but helps monitor progress, track status across multiple migrations/tasks, services, waves.

25. What are some strategies for phase-by-phase migrations (e.g., wave-based)?
Answer:

- Divide migration into waves/groups of databases or applications based on dependency, business risk, size.
- Do pilot migration on less critical DBs first to test process.
- Use "staging / test environment" waves for validation.
- Then larger or more critical databases in later waves.

26. How do cost considerations factor into planning a migration?
Answer:

- Cost of replication instance(s), target instance(s), storage, I/O, network transfer, data egress.
- Use savings plans, reserved instances for long-lived targets.
- Estimate licensing (if moving Oracle, SQL Server – keep licensing, bring your own? etc.).
- Cost of downtime or business impact.
- Consider operational cost after migration (maintenance, backups, monitoring).

27. What are the advantages of using Aurora (MySQL/PostgreSQL) vs RDS standard engines for migration?
Answer:

- Aurora offers better scalability, faster storage, auto-scaling of storage, high availability, faster failovers, often better performance.
- Read replicas more robust.
- Sometimes easier to scale up/out.

28. How do you migrate logins, users, permissions, roles?
Answer:

- Export or script out user accounts on source with their roles/privileges.
- If moving between accounts, ensure SIDs or equivalent identifiers are preserved or mapped.
- For engines like SQL Server, use scripts for logins; for MySQL/Postgres, dump roles/users.
- After migration, validate that permissions behave identically.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

29. Explain strategies for schema changes during migration without breaking application compatibility.
Answer:

- Use backward-compatible schema changes first (add fields, not remove).
- Use feature toggles in application.
- Possibly dual writes, or have versioned APIs.
- Let both old and new schema coexist until all application components updated.

30. How do you handle large object (LOB, BLOB) data during migrations?
Answer:

- DMS supports LOB handling; you may need to configure options like LOB mode (inline, separate LOB, etc.).
- If target engine has limitations, might need special handling.
- Possibly transfer LOBs separately or in batches.

31. What is "Time-to-Catch-Up" / lag in replication, and how do you monitor & mitigate it?
Answer:

- The time difference between the last transaction applied on source and the latest applied on target.
- Wide lag means prolonged cutover time or stale data.

Mitigation:

- Use adequately sized replication instance.
- Good network bandwidth.
- Reduce load or increase parallelism during bulk load.
- Possibly schedule periods of low activity.

32. How do you secure the networking for database migration?
Answer:

- Ensure endpoints & replication instances reside in VPCs, subnets with proper security groups.
- Use encrypted connections (TLS).
- If migrating from on-premises, use VPN or AWS Direct Connect.
- Limit access to only needed entities.

33. What about backups, retention & DR (Disaster Recovery) post-migration?
Answer:

- Configure automated backups (RDS/Aurora).
- Set point-in-time restore retention.
- Possibly cross-region backups or snapshots.
- Use read replicas or Multi-AZ deployments for HA.
- Test restores occasionally.

34. How do you validate performance after migration and tune the new system?
Answer:

- Capture performance baseline on source (queries, indexes, I/O, latency).
- After migration, run same workloads, queries; compare.
- Monitor for missing indexes, stale statistics.
- Adjust instance class, storage type, IOPS.
- Possibly use caching, read replicas.

35. What is AWS DMS "target table prep mode" and how might it matter?
Answer:

- In DMS, when setting up replication tasks, you can specify how to prepare target tables (e.g. drop & recreate, truncate, do nothing).
- If target already has schema, setting "do nothing" avoids dropping tables; good if you created schema manually (e.g. for more control over indexes, constraints).

36. How do you handle schema objects that DMS or SCT can't convert (e.g. complex triggers, stored procedures)?
Answer:

- Identify them via SCT reports.
- Manually rewrite or adjust for target engine.
- Possibly migrate them separately (outside of DMS tasks) and test thoroughly.

37. How do you migrate cross-region RDS instances with minimal downtime?
Answer:

- Use DMS with full load + CDC, target in the other region.
- Use read replicas in another region if supported, then promote.
- Plan syncing, cutover DNS or endpoint change.

38. How do you deal with "server-level" features or extensions not supported in RDS/Aurora?
Answer:

- Identify unsupported features during assessment phase.
- For features like certain extensions, custom OS configs, file system access, etc., consider moving to EC2-hosted DB.
- Or refactor application to use alternative features.

39. What is Multi-AZ vs Single-AZ versus cross-region/reader replicas? How do they affect migration strategy?
Answer:

- **Single-AZ**: cheaper, but less resilient; outage risk.
- **Multi-AZ**: Automated failover, higher availability. Good for production DBs.
- **Reader replicas (or Aurora replicas)**: scale read workloads; possibly use in read-heavy apps.
- When migrating, pick architecture that meets HA needs. Also use replicas/read endpoint for read workloads to reduce load on master during migration.

40. How to choose storage type & sizing (IOPS, throughput, latency) on AWS for DB targets?
Answer:

- Depends on the workload: OLTP vs OLAP, size of writes, size of reads, concurrency.
- Use provisioned IOPS or GP-SSD depending needs.
- Factor in growth, peak workloads.
- Monitor and adjust.

41. What logging & auditing do you put in place during and after migration?
Answer:

- Enable database logs (slow query logs, general logs) on source & target.
- Ensure AWS CloudTrail, CloudWatch logs are capturing events.
- Possibly use AWS Config for configuration drift.
- Audit access, permissions change, schema change.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

42. What are the AWS services/tools besides DMS & SCT you might use in migration?

Answer:

- AWS Migration Hub (for tracking).
- AWS Application Migration Service (MGN) for server/application migrations.
- Snowball / Snowmobile for large-scale offline data transfer.
- AWS DataSync for file/data transfers.
- AWS VPN / Direct Connect for connectivity.
- AWS Identity & Access Management (IAM), AWS Key Management Service (KMS) etc.

43. Describe troubleshooting steps when a DMS task lags behind or fails.

Answer:

- Check task status & logs for errors.
- Examine replication instance performance (CPU/memory/disk).
- Verify endpoints connectivity and credentials.
- Check for table sizes, large LOBs; perhaps split tasks.
- Check network latency / bandwidth.
- Review transformations, filter rules that may be slowing things down.
- Ensure source DB is not overloaded.

44. How do licensing issues play a role in AWS DB migrations (Oracle, SQL Server, etc.)?

Answer:

- Consider license model on source vs target.
- If using AWS RDS or EC2, check whether you can bring your own license, whether license mobility applies.
- Costs of licenses included in RDS/Aurora vs separate in EC2.
- Be aware of feature differences in “licensed” editions.

45. What is the cost of data transfer / egress migration?

Answer:

- Moving data from on-prem to AWS may involve network costs; from other AWS regions there may be inter-region charges.
- Also consider cost of replication instances, storage.
- Possibly cost of Snowball / physical devices.

46. How do you archive or decommission the old on-prem database after migration?

Answer:

- After migration and sufficient validation, plan for decommission: backup the old database, snapshot, or archive.
- Ensure all applications point to new target.
- Monitor for any failure, fallback support.
- Retire hardware, clean up security access.

47. What are best practices around schema & version control during migrations?

Answer:

- Keep schema and database code (views, stored procedures) in version control.
- Use CI/CD pipelines to deploy schema changes.
- Use migration scripts rather than manual edits.
- Maintain backward compatible schema changes during cutover phases.

48. What metrics & SLAs do you define for a successful migration?

Answer:

- Downtime window permitted.
- Time to full sync / cutover.
- Target performance metrics (latency, throughput) vs source.
- Data consistency measures (row count, hash, etc.).
- Uptime, error rate post migration.

49. What are security best practices for replication instance setup?

Answer:

- Place replication instance in private subnet.
- Use security groups to restrict access.
- Encrypt connections, use SSL/TLS.
- Use IAM roles for DMS to access resources.
- Monitor and log its operations.

50. How do you handle migration of privileges, roles, stored procedures when source engine is proprietary (e.g. Oracle) to open-source target (e.g. PostgreSQL)?

Answer:

- Use SCT to convert code where possible.
- Manually reimplement stored procedures, triggers, UDFs in target engine language.
- Recreate roles/users; map privileges to equivalent in target.
- Test thoroughly for behavior, performance.

51. How do you handle multi-tenant or sharded databases during migration?

Answer:

- If databases are sharded or multi-tenant, plan by shard/tenant grouping.
- Possibly migrate tenants in waves.
- Consider whether data segregation or custom mapping is needed.
- Ensure tenant-specific schema or security constraints are handled.

52. For PostgreSQL / MySQL to Aurora migrations, what special considerations are there?

Answer:

- Compatibility of PostgreSQL / MySQL versions.
- Compatibility of extensions, stored functions, triggers.
- Behavior differences (e.g. default collations, replication behavior).
- Aurora-specific optimizations (e.g. writer / reader endpoints, scaling).

53. What is Amazon RDS Logical & Physical Backup, and how does backup/restore work with migrations?

Answer:

- RDS automated backups and snapshots; point-in-time restore.
- Physical vs logical backups depend on engine. For example, MySQL may allow logical dumps, or physical snapshots via storage snapshot.
- When migrating, ensure backup strategies are in place for the target.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

54. How do you migrate from a NoSQL / non-relational database to a relational one, or vice versa?
Answer:

- Need to rethink schema/data model.
- Use ETL pipelines to extract, transform & load data.
- For AWS, could use AWS Glue, AWS Data Pipeline, custom code, or DMS (if it supports non-relational source).
- Address mapping of JSON documents or key-value models to relational tables or vice versa.

55. What role does network topology & connectivity (VPN, Direct Connect) play in large or mission-critical migrations?
Answer:

- For large data transfers, having high throughput, low latency, stable connection is essential.
- Direct Connect often preferred for consistent throughput & lower cost per GB.
- VPN may be used but may suffer variability.
- Topology (e.g. placing replication instance close to VPC where RDS is) can reduce latency.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

Here are the **Top AWS Database Migration Interview Questions** with **deep technical and scenario-based answers** – tailored for **senior DBAs, Solutions Architects, or Cloud Migration Specialists**. These are the kinds of questions you'd get in architect-level interviews where you're expected to design and troubleshoot real-world migrations, not just know tool names.

🔥 **Top AWS DB Migration Interview Questions (Advanced, Real-World Scenarios)**

1. A 5 TB on-prem Oracle database must be migrated to Amazon Aurora PostgreSQL with minimal downtime. What is your migration strategy?
Answer:
Strategy:

- **Assessment:** Use **AWS Schema Conversion Tool (SCT)** to assess schema and procedural object compatibility.
- **Schema Conversion:** Convert Oracle schema to PostgreSQL using SCT, including manual rewriting for PL/SQL packages to PL/pgSQL.
- **Data Migration:**
  - Use **DMS in Full Load + CDC** mode.
  - Initial full load to Aurora PostgreSQL.
  - CDC keeps the target in sync.
- **Cutover:** Freeze writes on source → sync lag to near-zero → application DNS or connection string update → cutover.

Challenges:

- LOB handling (inline or via separate process)
- Data type mismatches
- Differences in transaction isolation and functions
- Triggers/procedures must be rewritten

Downtime: Only during final cutover (minutes to <1 hour).

2. Your DMS task is lagging behind during CDC. How do you troubleshoot and fix it?
Answer:
Step-by-step Troubleshooting:

1. **Check Replication Instance Metrics:**
   - CPU, Memory, IOPS (CloudWatch)
   - Upgrade instance size if CPU-bound
2. **Network Bottleneck:**
   - Latency between source and DMS or DMS and target
   - VPC peering / Direct Connect can help
3. **Large Transactions / LOBs:**
   - Break large transactions if possible
   - Enable LOB Truncation or set Max LOB size
4. **Target Constraints/Indexes:**
   - Too many indexes = slower CDC apply rate
   - Consider disabling indexes during migration
5. **Parallelism:**
   - Enable multi-threaded load (Parallel load mode = enabled)
6. **Monitor Replication Lag:**
   - Use CDCLatencySource and CDCLatencyTarget metrics in CloudWatch

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

3. What would you recommend for migrating a SQL Server DB with linked servers and CLR objects to AWS?

Answer:

Option 1: EC2 SQL Server

- Retains all SQL Server features: CLR, SQL Agent, Linked Servers, etc.
- Full backup/restore supported

Option 2: RDS SQL Server

- Check for:
    - Linked servers: Only supported with RDS-to-RDS
    - CLR: Partially supported with elevated permissions
    - SQL Agent: Supported
- Use **native backup and restore via S3**

Migration Plan:

1. Full backup from on-prem → upload to S3
2. Use RDS stored procedures to restore
3. Recreate linked servers (if RDS-to-RDS or refactor)
4. Re-deploy CLR assemblies if compatible

Alternative: Refactor app to remove dependencies and migrate to Aurora or PostgreSQL.

4. You're tasked with migrating 100+ databases from multiple environments (Prod, Dev, QA). How do you structure the migration?

Answer:

Best Practices:

- **Wave-based approach:**
    - Group DBs by environment and criticality (e.g., Dev → QA → Prod)
- **Automation:**
    - Use AWS CloudFormation / Terraform to provision endpoints and replication tasks
    - Use DMS task JSON templates
- **Centralized Tracking:**
    - Use **AWS Migration Hub** to monitor progress

Data Governance:

- Secure credentials via Secrets Manager
- Role-based access for DMS tasks

Validation:

- Define per-DB validation queries, use Lambda or scripts to verify data integrity post-migration

5. What are the key limitations of AWS DMS and how do you handle them in production-grade migrations?

Answer:

Limitation	**Mitigation**
No support for stored procedures	Use SCT + manual conversion
Constraints not enforced during migration	Reapply constraints after migration
Schema drift during migration	Freeze schema or use version control
CDC lag in high-load systems	Use powerful replication instances, split tasks
LOBs slow migration	Enable LOB truncation or migrate LOBs separately
Not transactional during full load	Accept non-atomic loads or sync via app logic

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

6. Explain the difference between RDS, Aurora, and EC2-hosted DBs. When would you choose each during migration?

Answer:

Option	**Use When**
RDS	You want managed DB with low ops burden; app supports the RDS-supported engine
Aurora	You need high availability, scalability, and PostgreSQL/MySQL compatibility
EC2-hosted	Need full control over OS or advanced DB features not supported in RDS/Aurora (e.g. Oracle EE)

Migration Strategy:

- To RDS: Use DMS or native backups (.bak for SQL Server, logical dump for MySQL/Postgres)
- To Aurora: Prefer DMS or native dump → import
- To EC2: Full control; do traditional backup/restore

7. How do you ensure post-migration consistency and application stability?

Answer:

Data Consistency:

- Row count comparison
- Hash/checksum comparison per table
- Data sampling and query comparison

Application Stability:

- Use **shadow reads** during parallel testing
- Performance benchmarking with real workloads
- Enable detailed monitoring (e.g., enhanced RDS metrics, Query Plan analysis)

Tools:

- Custom scripts or 3rd-party (Redgate, Data Compare tools)
- DMS table validation mode (enabled in task settings)

8. A database migration failed mid-transfer. How do you recover?

Answer:

DMS supports resumable migrations:

- Task metadata is persisted
- Resume task from failure point

Steps:

1. Check DMS logs to identify failure (e.g., table error, network loss)
2. Fix root cause (e.g., increase instance size, fix schema mismatch)
3. Restart task:
    - **"Resume processing"** for CDC
    - **"Reload failed tables"** for full load
4. Validate rows using validation reports

If unrecoverable, consider:

- Re-run only affected tables
- Partitioned migration
- Parallelizing tasks to reduce blast radius

9. What's your approach to managing security and compliance during AWS DB migration?

Answer:

Data at Rest:

- Encrypt replication instance storage (KMS)

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

- Enable encryption on RDS / Aurora using customer-managed keys

Data in Transit:

- Use TLS for source → DMS → target

Identity and Access:

- Use least-privileged IAM roles
- Store secrets in **AWS Secrets Manager**

Audit & Logging:

- CloudTrail for user actions
- DMS task logging enabled
- CloudWatch for monitoring and alerting

Compliance:

- Document the data flow
- Redact or mask PII where applicable
- Run vulnerability assessments post-migration

10. How do you migrate a live production PostgreSQL database from EC2 to Aurora with <5 mins downtime?

Answer:

Approach:

1. Enable **logical replication** on source PostgreSQL (rds.logical_replication=1)
2. Create Aurora PostgreSQL cluster
3. Use **DMS with CDC**:
   - Start with Full Load + CDC
   - Use multi-threaded parallel load
4. Sync until CDC lag is < few seconds
5. Cutover:
   - Pause writes on source
   - Wait for CDC catch-up
   - Redirect application to Aurora
6. Post-migration:
   - Validate data
   - Enable backups, monitoring, etc.

Challenges:

- App must support Aurora PostgreSQL quirks
- Ensure network latency is low
- Foreign keys and triggers must be validated post-migration