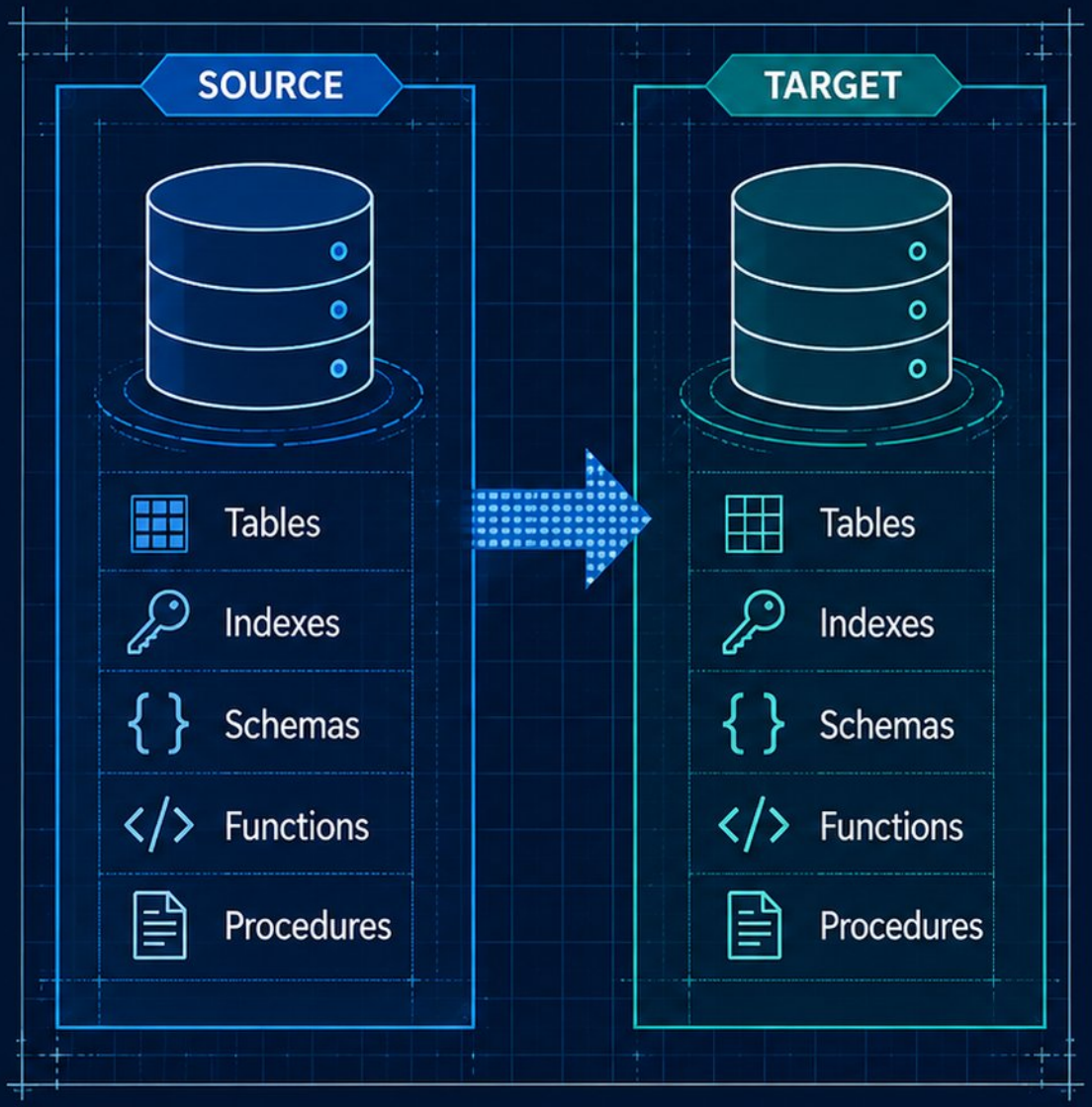
Introduction to SQL Database Migration Tool

- Fully managed service for moving databases to Azure
- Supports SQL Server, MySQL, PostgreSQL, and more
- Scenarios: on-premises to cloud, upgrades, cross-platform
- Phases: assessment, schema conversion, data transfer, validation
- Integrates with DBA, data engineering, and platform workflows

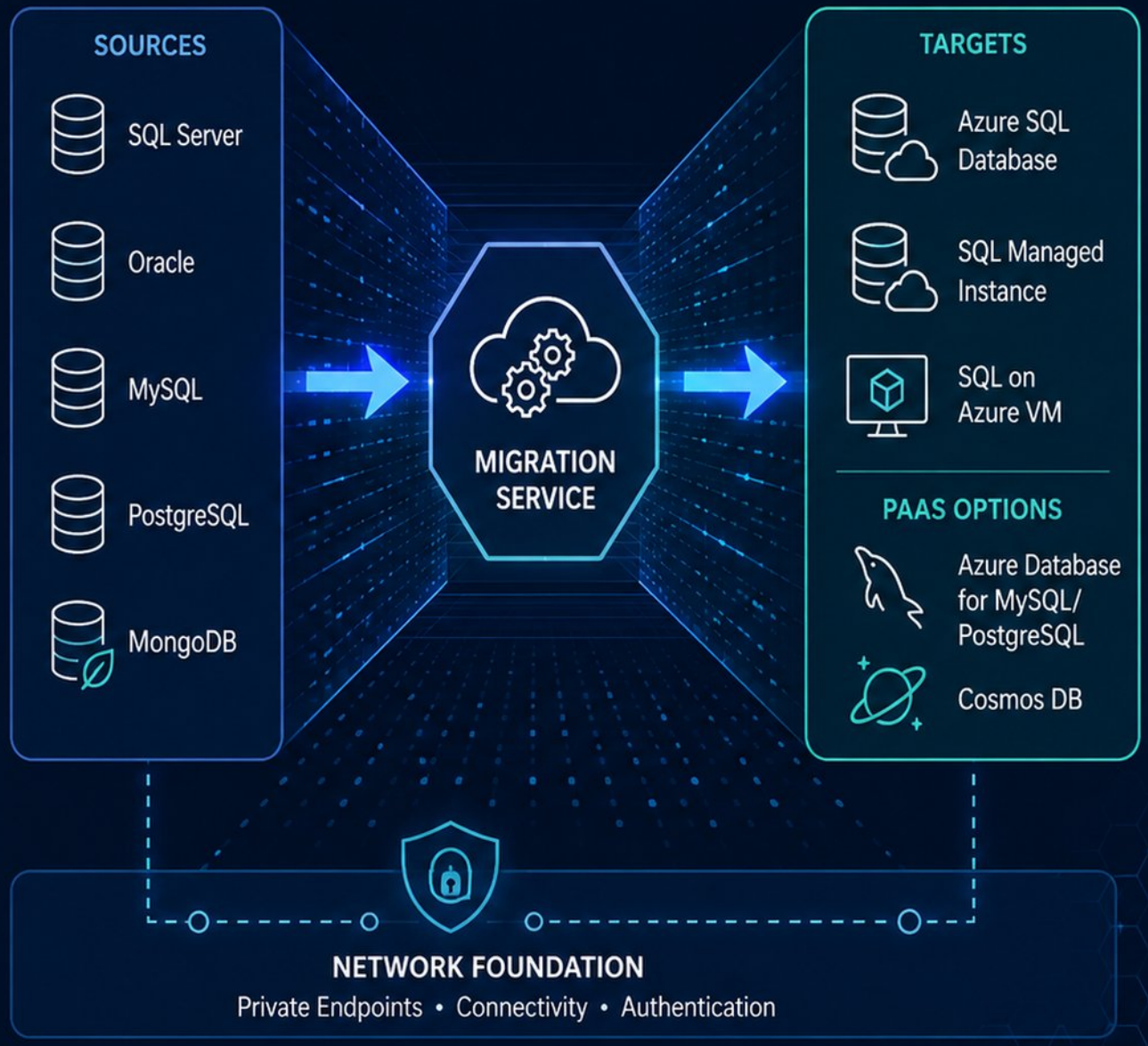
Core Concepts and Terminology



- Source: the original database; target: the destination platform.
- Offline migrations require downtime; online migrations use continuous sync to minimize it.
- Homogeneous migrations stay on the same engine; heterogeneous migrations change engines.
- Assessment detects blockers; schema conversion maps types and functions.
- Reconciliation verifies data parity through row counts and query comparisons.

Supported Sources, Targets, and Environments

- **Common sources:** SQL Server, Oracle, MySQL, PostgreSQL, and MongoDB
- **Targets:** Azure SQL Database, SQL Managed Instance, SQL on Azure VM
- **PaaS options:** Azure Database for MySQL/PostgreSQL, Cosmos DB
- **Network prerequisites:** private endpoints, connectivity, and authentication
- **Limitations:** unsupported features like linked servers and cross-database queries

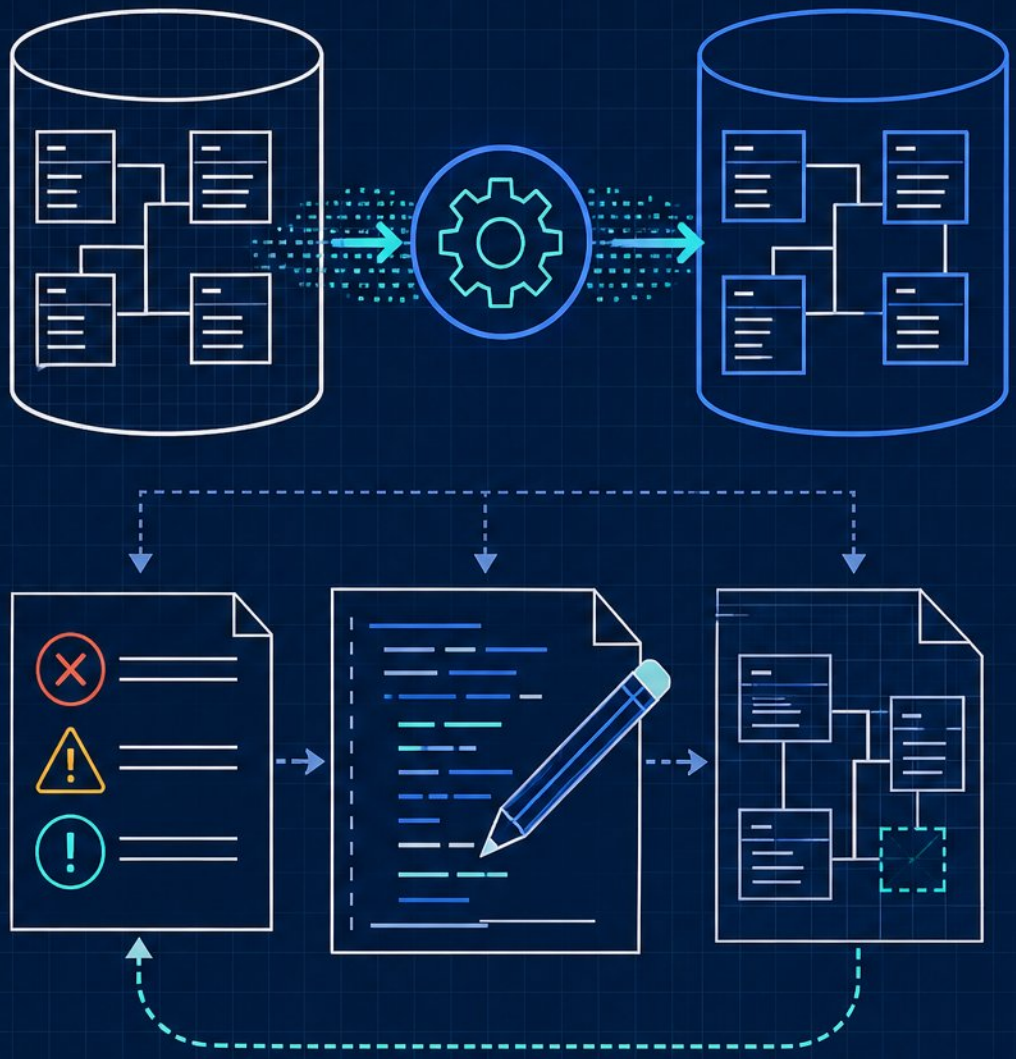


Pre-Migration Assessment

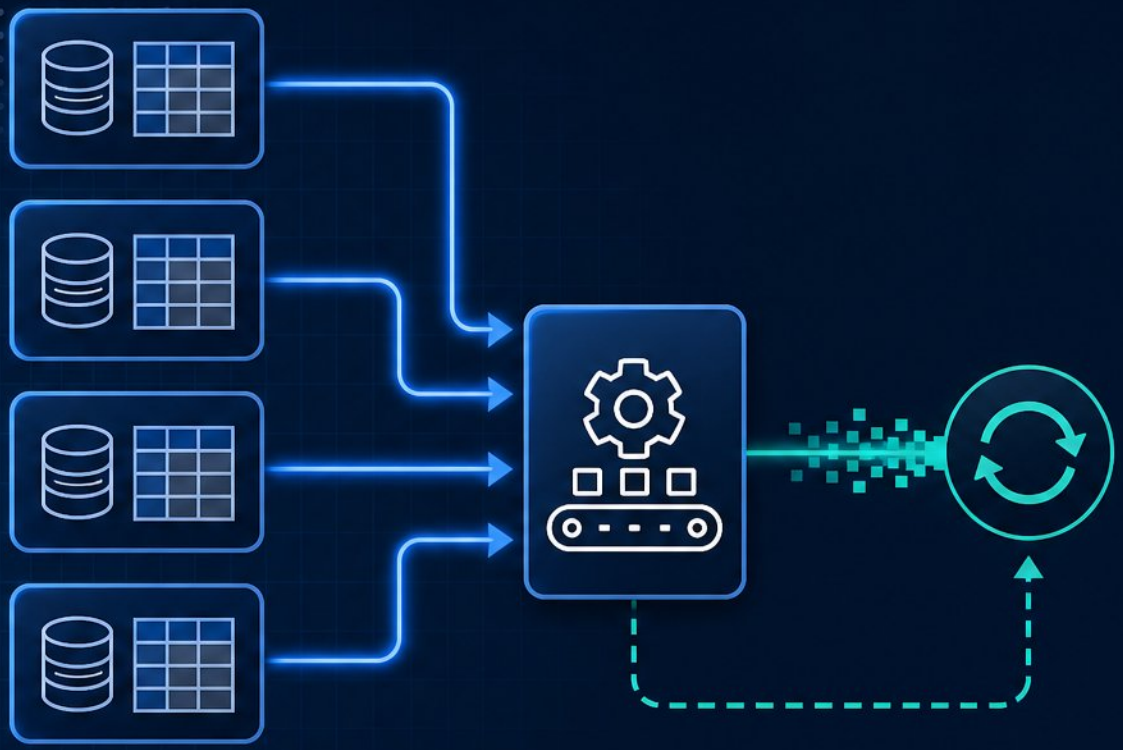
- Run automated assessments against source databases
- Interpret blockers, warnings, and remediation guidance
- Readiness categories: Ready, Ready with conditions, Not ready
- Address unsupported features and schema incompatibilities
- Use findings to sequence migration waves and estimate effort

Schema Conversion and Remediation

- Rules-based engine converts schemas automatically and consistently
- Review reports to identify conversion errors, warnings, and manual gaps
- Edit converted SQL directly or apply fixes for unsupported objects
- Handle data type, collation, and index mapping differences
- Iterate: rerun conversion, validate results, and remediate until clean

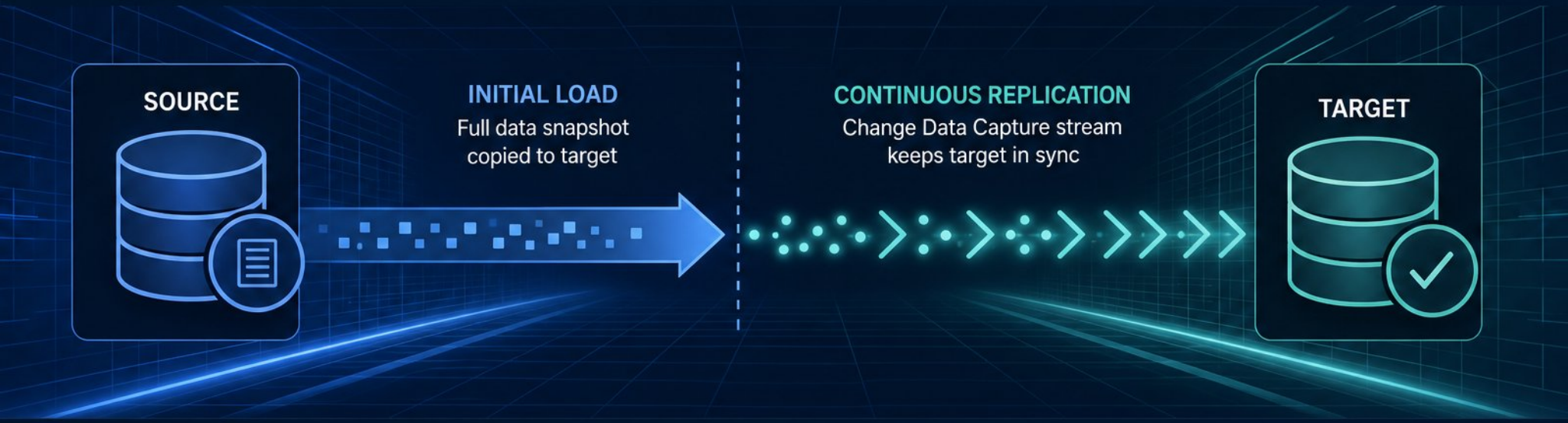


Data Transfer and Synchronization



- One-time bulk load for historical data vs. continuous CDC replication
- Backup-based initial loads: network share or Azure storage blob
- Use self-hosted integration runtime to access on-prem backup files
- Bulk load uses parallel transfer; monitor and resume failed jobs
- Continuous sync captures changes until cutover, minimizing downtime

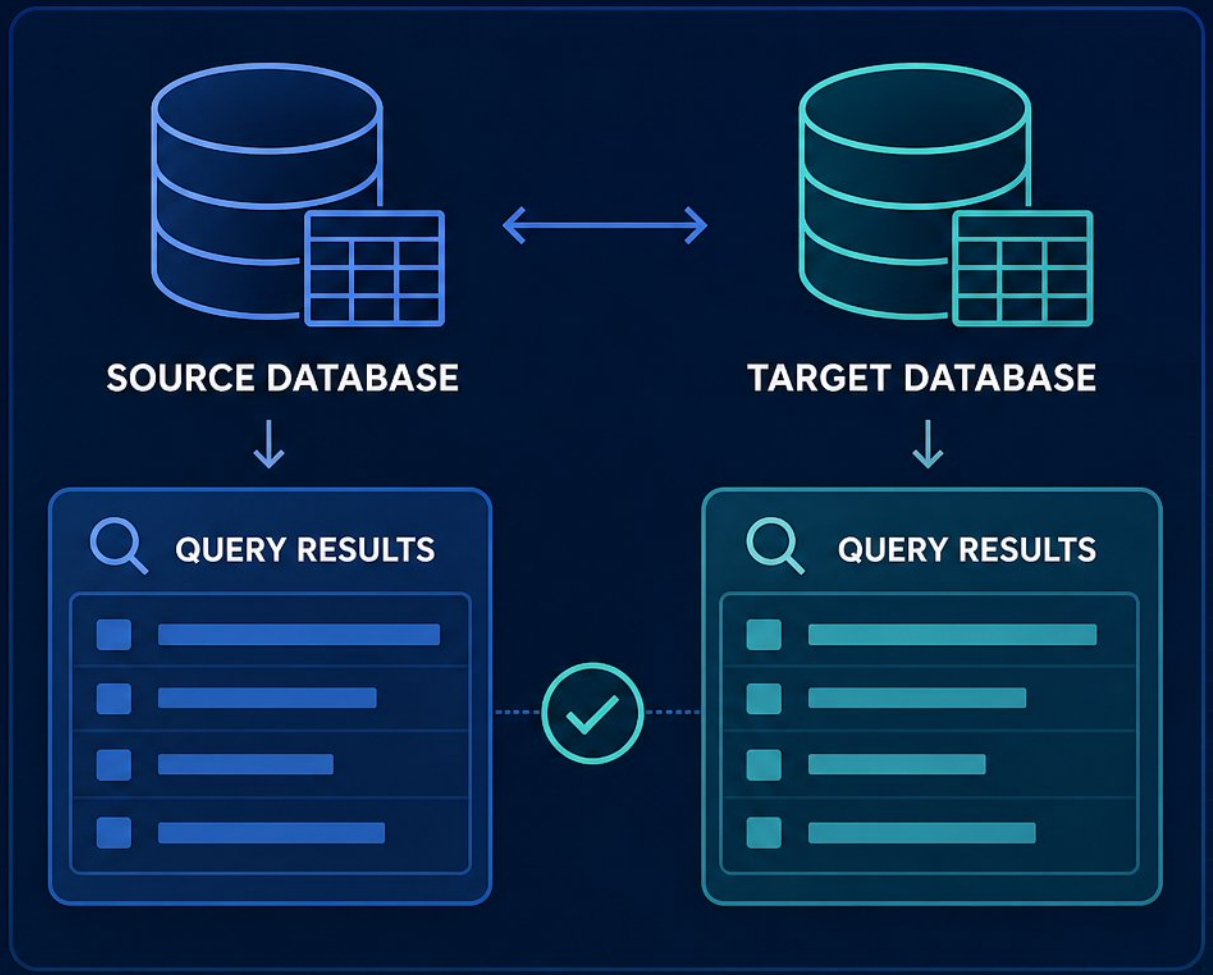
Online Migration with Change Data Capture



- Online mode uses CDC to keep source and target in sync
- Enable database-level CDC and snapshot isolation on source
- Continuous log replication follows the initial full load
- Cutover is the only downtime window, planned with business
- Limitations: no online index operations, requires valid log chain

Validation and Cutover Planning

- Validate migrated schemas, row counts, and data consistency
- Compare source-target query results and test app workloads
- Plan cutover with downtime windows and rehearsed rollback
- Freeze source changes before final sync and application switchover



Performance Tuning
During Migration

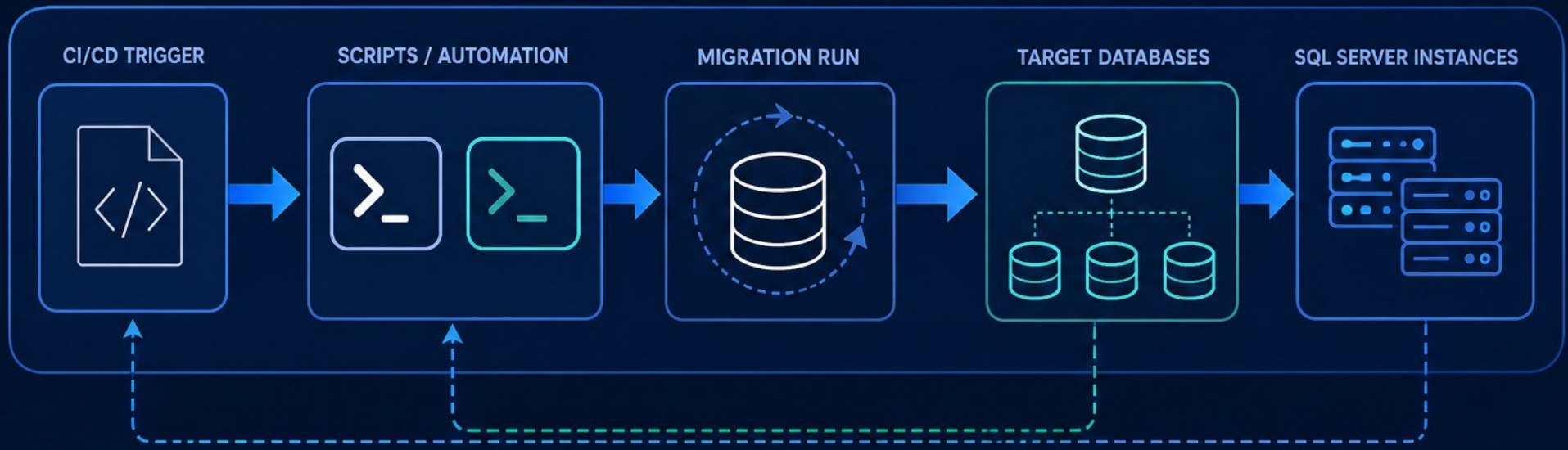
- Source bottleneck: data file I/O and latency
- Target: scale up database for 96 MB/s log rate
- Network: ExpressRoute for needed bandwidth
- Disable auto-statistics; partition tables and indexes
- Right-size SKU; scale down after migration

Security Best Practices

- Encrypt data at rest with TDE and use TLS 1.2 for all traffic
- Follow least privilege with custom RBAC roles for service accounts
- Use ExpressRoute or site-to-site VPN to keep traffic off the internet
- Restrict access via private endpoints and IP firewall rules
- Store backups in an isolated Azure Blob container with SAS tokens

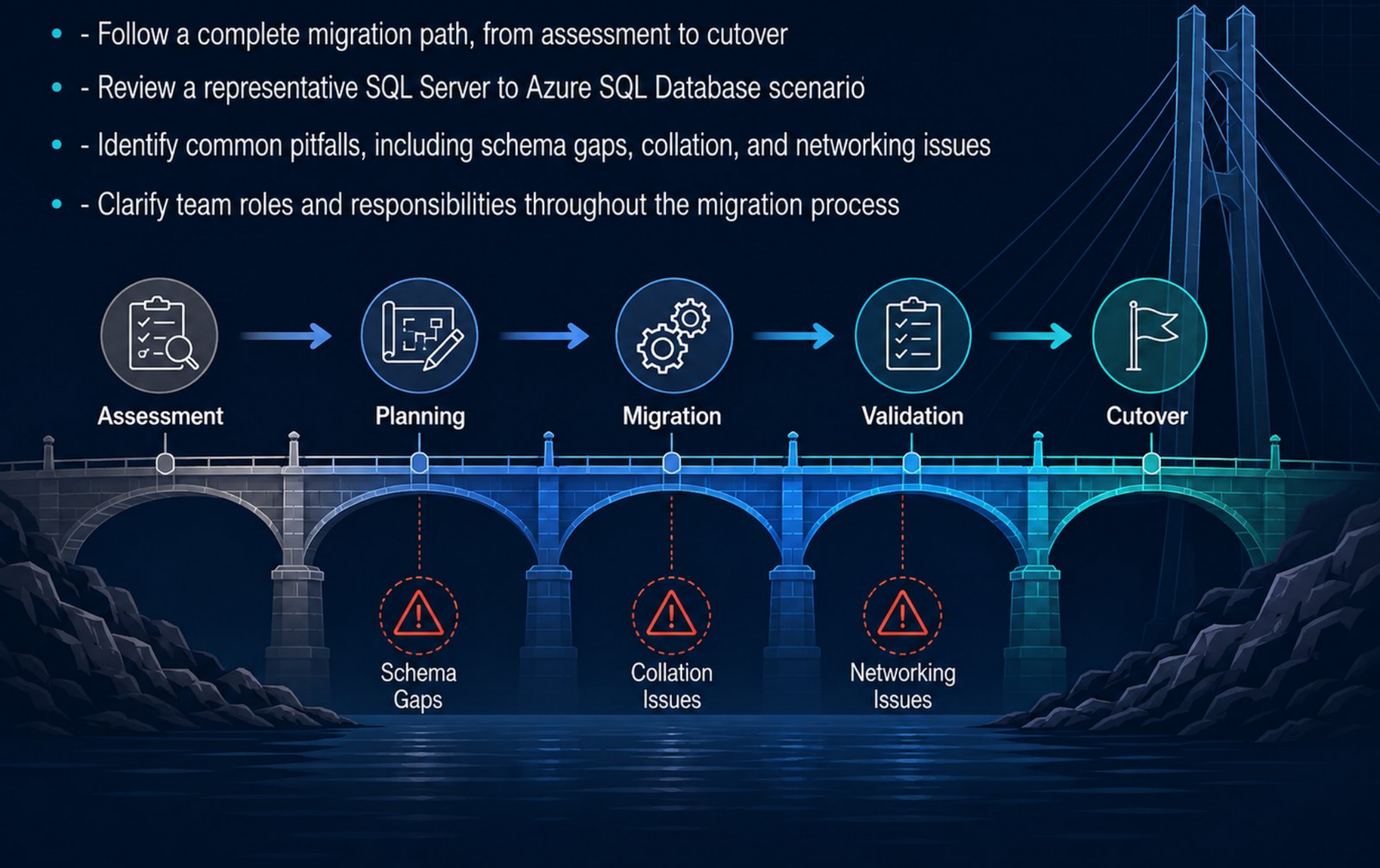
Automation for Platform Teams

- - Automate migration runs via PowerShell or Azure CLI
- - Scale across multiple databases and SQL Server instances
- - Integrate with CI/CD and infrastructure-as-code
- - Reuse Azure Samples scripts for repeatable patterns
- - Register Microsoft.DataMigration provider before first use

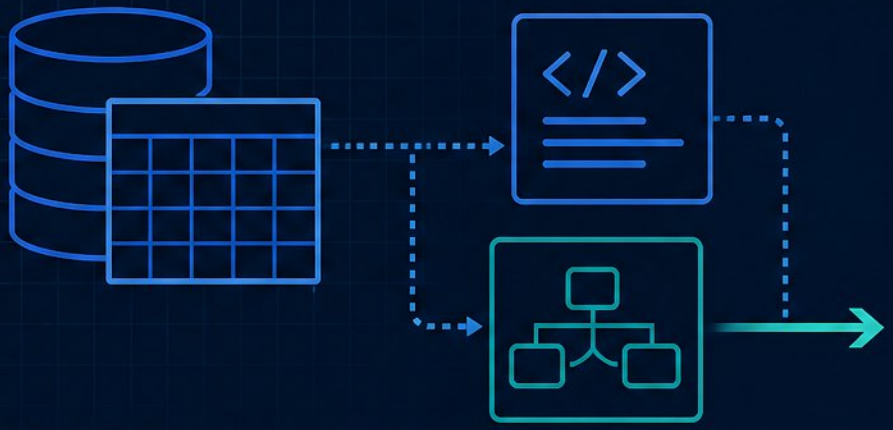


Real-World Migration Walkthrough

- - Follow a complete migration path, from assessment to cutover
- - Review a representative SQL Server to Azure SQL Database scenario
- - Identify common pitfalls, including schema gaps, collation, and networking issues
- - Clarify team roles and responsibilities throughout the migration process



Adoption Checklist and Pilot Planning



- Pilot with hardest data first: largest tables, complex joins
- Set acceptance criteria: parity ≥ 99.9%, error rate ≤ 0.5%
- Build repeatable runbook covering assessment through cutover
- Connect pilot results to target sizing and platform goals
- Find docs, scripts, and community support for SQL migrations

PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/38d963a1/public