# On-Premise Database and Azure Managed Instance Database

**Shaunak Patel**

**Applied Network Infrastructure and System Administration,**

**Conestoga Collage**

# Table of Contents

# INTRODUCTION

In navigating the integration of a cloud-based database service within the Microsoft Azure ecosystem, the choice between deploying a cloud-hosted MS SQL Server and utilizing the Azure SQL service is a critical decision. In this plan, the preference leans towards Azure SQL service, specifically the Azure SQL Database, owing to its platform-as-a-service (PaaS) nature. The decision is underpinned by the inherent advantages of a fully managed database service, which alleviates the administrative burden associated with infrastructure management. Azure SQL Database ensures automatic updates, high availability, and built-in security features, allowing for a more streamlined focus on application development and database functionality.

The decision to opt for the Azure SQL service is substantiated by its automatic update capabilities, mitigating the risk of security vulnerabilities by ensuring the application of the latest patches seamlessly. Additionally, the built-in high availability features, including geo-replication and failover capabilities, enhance the resilience of the database. To facilitate effective data synchronization between on-premise databases and the Azure SQL Database, a comprehensive strategy is outlined. Initial steps involve the assessment of critical data sets and the determination of synchronization frequency based on business requirements. Leveraging Azure services such as Azure Data Factory becomes integral to automating ETL processes, ensuring secure and efficient data transfer between on-

premise and cloud environments. Ongoing monitoring and alerting mechanisms are implemented to promptly address any synchronization issues, ensuring data consistency and minimizing latency in the synchronization process.

Additionally, a meticulous failover plan is articulated to seamlessly redirect business users to the cloud-based solution in the event of on-premise downtime due to updates or outages. This encompasses the use of Azure Traffic Manager for DNS-level failover, a communication plan for user guidance, and regular failover tests to validate the effectiveness of the transition process. In sum, our plan is designed to harness the advantages of Azure's managed services, ensuring a robust and efficient integration of cloud-based database services into our operational landscape.

## Plan Timeline:

Developing a thorough plan timeline involves breaking down tasks and activities into feasible stages for integrating a cloud-based database service into our company's Microsoft Azure environment.

Stage 1: Assessment and Preparation

- First we conduct a thorough assessment of the existing on-premise database environment, identifying critical databases, data dependencies, and performance metrics.
- Then we evaluate the current licensing model for SQL Server and ensure compliance with Azure SQL Database pricing models.
- While performing these tasks we need to engage stakeholders, including IT, development teams, and business users, to gather requirements and expectations.
- After completing assessment we need to select and configure Azure subscription and resources.
- Assess network connectivity and security requirements between on-premise and Azure environments.
- Set up Azure Active Directory for authentication and authorization.
- This whole process will take around 2 weeks to complete.

Stage 2: Database Migration and Deployment

- In this phase we are doing database migration in batches format. By performing this task we need to consider dependencies and interrelationships.
- Choose an initial database for migration based on priority and complexity.
- For migration process we are using Azure Database Migration Service.
- After completing migration of databases we have to implement Azure SQL Database features such as elastic pools or geo-replication as needed for scalability and redundancy.
- Perform thorough testing of applications connected to the migrated databases to ensure compatibility and optimal performance.
- To complete stage 2 will take 4-5 weeks in total.

Stage 3: Data Synchronization and Optimization

- Establish automated data synchronization processes using Azure Data Factory.
- Fine-tune synchronization schedules based on business requirements and data volatility.
- Optimize database configurations and indexes for Azure SQL Database.
- Monitor and adjust resource allocation to ensure optimal performance and cost-effectiveness.
- Conduct training sessions for IT and development teams on Azure SQL Database management and optimization.
- So this process will take around 4 weeks to complete.

Stage 4: Failover Planning and Testing

- Develop and document failover procedures for redirecting users to the cloud-based solution during on-premise outages.
- Configure Azure Traffic Manager for DNS-level failover.
- Conduct a comprehensive failover simulation, including user communication and resource redirection.
- This task will take 2 weeks to complete.

Stage 5: Finalizing and Review

- Perform a final review of the entire migration and integration process.
- Document lessons learned and create a knowledge base for future reference.
- It will take 1 or 1.5 weeks to complete.

## Breakdown of Tasks:

Database Migration Tasks:

- Identify Database: Conduct an inventory of all databases on the on-premise server. Categorize databases based on criticality and dependencies.
- Dependency Analysis: Assess database relationships and dependencies. Identify potential challenges or constraints in migration.
- Azure Setup: Set up an Azure SQL Database instance in the Azure portal. Configure server settings, including firewall rules and authentication.
- Networking Configuration: Establish secure network connectivity between on-premise and Azure environments. Configure Virtual Network Service Endpoints for enhanced security.
- Database Selection: Choose a pilot database for initial migration. Consider factors such as data size, complexity, and dependencies.
- Azure Database Migration Service: Azure DMS for database migration, selecting an appropriate migration method. Monitor the migration progress through Azure portal diagnostics.
- Data Consistency Check: Validate data consistency between the on-premise and cloud-based databases. Perform data integrity checks to ensure a successful migration.

Data Synchronization Tasks:

- Azure Data Factory: Implement Azure Data Factory or a similar ETL tool for automated data synchronization. Define data pipelines and workflows for regular synchronization.
- Synchronization Schedules: Configure synchronization schedules based on business requirements and data volatility. Consider off-peak hours for minimizing impact on operations.

- Performance Tuning: Monitor and optimize database configurations and indexes for Azure SQL Database. Fine-tune synchronization processes for efficiency.

Failover Planning and Testing:

- Failover Documentation: Develop detailed documentation for failover procedures during on-premise outages. Include steps for redirecting users to the cloud-based solution.
- Azure Traffic Manager Configuration: Configure Azure Traffic Manager for DNS-level failover. Validate the setup and ensure it aligns with failover procedures.
- Comprehensive Simulation: Conduct a comprehensive failover simulation, mimicking on-premise downtime. Test user redirection, application connectivity, and data access in the cloud.

Business Continuity in Downtime:

- Communication Strategy: Establish a clear communication plan for notifying business users during on-premise downtime. Provide guidance on accessing resources in the cloud.
- User Training: Conduct training sessions to familiarize users with the cloud-based solution and failover processes.

This detailed breakdown provides a structured plan for each phase of the database migration, synchronization, and failover process.

## Key Milestones:

Key milestones are crucial for tracking progress and ensuring that each phase of the process is completed successfully. Here are suggested key milestones for the database migration, synchronization, and failover planning process:

Assessment and Planning:

- Evaluate the existing on-premise SQL Server environment, assess the readiness for migration, and plan the migration strategy.

- Activities: Inventory of databases and applications. Assess compatibility with Azure SQL Managed Instance. Define migration goals and requirements. Create a detailed migration plan.

Azure Environment Setup:

- Provision and configure the Azure environment to accommodate the Azure Managed Instance.
- Activities: Create Azure subscription and resource group. Provision Azure SQL Managed Instance. Configure network settings, including Azure ExpressRoute.

Backup and Restore Validation:

- Ensure the reliability of the backup and restore processes for on-premise databases.
- Activities: Perform a full database backup using SQL Server Management Studio (SSMS). Validate the restoration of the backup on Azure Managed Instance.

Database Migration:

- Execute the actual migration of databases from on-premise SQL Server to Azure Managed Instance.
- Activities: Utilize Azure Database Migration Service tools. Monitor and validate the migration progress. Address any issues or errors encountered during migration.

Traffic Manager Configuration:

- Implement Azure Traffic Manager for load balancing and managing request traffic between on-premise and Azure environments.
- Activities: Configure Traffic Manager profiles. Verify proper redirection and failover mechanisms.

ExpressRoute Connection Establishment:

- Establish a secure and dedicated connection between the on-premise environment and Azure Managed Instance using Azure ExpressRoute.

- Activities: Set up ExpressRoute circuits and connections. Validate the connectivity and latency.

Data Synchronization Setup:

- Configure data synchronization mechanisms between on-premise SQL Server and Azure Managed Instance.
- Activities: Install and configure sync agents. Define synchronization rules and intervals. Validate data consistency between environments.

Daily Backup Job Creation:

- Establish daily backup routines for on-premise databases.
- Activities: Create SQL Server Agent jobs for daily backups. Monitor and validate the backup job executions.

Failover Group Configuration:

- Configure automatic failover groups for Azure Managed Instance to ensure high availability.
- Activities: Set up primary and secondary replicas. Configure failover policies and monitoring.

Monitoring and Validation:

- Implement monitoring solutions to track the health and performance of the entire environment.
- Activities: Set up Azure Monitor for performance monitoring. Validate the functionality of monitoring alerts.

Post-Migration Review:

- Evaluate the success of the migration, identify lessons learned, and plan for continuous improvement.
- Activities: Conduct a post-migration review meeting. Document feedback and areas for improvement.

## Plan Notification:

Creating a detailed notification plan is crucial to maintaining effective communication during various scenarios, such as on-premise server downtime, updates, or outages, and the subsequent transition to Azure Managed Instance. Here's a plan for notifications:

Incident Identification:

- Method: Automated Monitoring Tools (Azure Monitor, SQL Server alerts)
- Recipients: IT Operations Team, Database Administrators (DBAs)
- Frequency: Real-time
- Content: Immediate alerts for any unplanned server downtime, performance degradation, or other critical issues.

Planned Maintenance Notifications:

- Method: Email, Collaboration Platforms (Microsoft Teams, Slack)
- Recipients: IT Operations Team, DBAs, Relevant Stakeholders
- Timing: 72 hours before maintenance
- Content: Detailed schedule, purpose, and expected impact of planned maintenance. Include contact details for support.

Azure Managed Instance Failover:

- Method: SMS, Email, Phone Calls
- Recipients: IT Operations Team, DBAs, System Administrators
- Timing: Real-time (Automated)
- Content: Immediate notification on failover initiation, including the reason and expected duration. Follow-up notifications on completion or any issues.

Traffic Manager Redirection:

- Method: Push Notifications, Email
- Recipients: IT Operations Team, Network Administrators
- Timing: Real-time (Automated)
- Content: Notifications on Azure Traffic Manager redirection, providing details on the affected resources and the reason for redirection.

ExpressRoute Connectivity Issues:

- Method: Ticketing System (ServiceNow, Jira), Email
- Recipients: IT Operations Team, Network Administrators, Azure ExpressRoute Support
- Timing: Real-time (Automated)
- Content: Immediate notification on connectivity issues, ongoing updates, and resolution progress.

Database Migration Status:

- Method: Email, Azure Portal Notifications
- Recipients: IT Operations Team, DBAs, Project Managers
- Timing: At key migration milestones
- Content: Regular updates on the database migration status, including successful completions, any encountered issues, and resolutions.

Backup Completion and Validation:

- Method: Automated Alerts, Email
- Recipients: IT Operations Team, DBAs
- Timing: After each backup operation
- Content: Notifications confirming successful backup completion, validation status, and any issues identified during the process.

Data Synchronization Status:

- Method: Azure Portal Notifications, Email
- Recipients: IT Operations Team, DBAs
- Timing: At each synchronization interval (e.g., every 5 minutes)
- Content: Notifications on data synchronization status, highlighting successful syncs, any conflicts, and resolutions.

Post-Migration Verification:

- Method: Email, Azure Portal Notifications
- Recipients: IT Operations Team, DBAs, Project Managers
- Timing: After completion of the migration

- Content: Notification confirming the successful completion of the migration, verification steps taken, and any post-migration tasks that need attention.

Critical Alerts from Azure Monitor:

- Method: SMS, Email, Collaboration Platforms
- Recipients: IT Operations Team, DBAs
- Timing: Real-time
- Content: Immediate notifications for critical alerts from Azure Monitor, providing details on the issue, affected resources, and recommended actions.

Service Level Agreement (SLA) Breach:

- Method: Email, SMS
- Recipients: IT Operations Team, DBAs, Management
- Timing: Real-time
- Content: Immediate notification if there's a potential or actual SLA breach, outlining the impact and steps being taken to resolve the issue.

Continuous Communication Channels:

- Method: Regular Meetings, Status Reports
- Recipients: IT Operations Team, DBAs, Stakeholders
- Timing: Weekly or as needed
- Content: Regular status updates, ongoing challenges, and upcoming changes or maintenance activities.

This notification plan ensures that relevant stakeholders are promptly informed at every stage of the scenario, contributing to efficient incident management, collaboration, and a transparent communication flow.

## Detailed Breakdown of Problem:

### On-Premise Database Challenges:

Scalability Limitation

- Problem: The existing on-premise database infrastructure exhibits limitations in scalability, hindering its ability to efficiently handle growing data volumes and increased user demands. This is impacting system performance and responsiveness.
- Impact: Reduced scalability affects the organization's agility and responsiveness to evolving business needs, potentially leading to performance bottlenecks during peak usage periods.

High Maintenance Overhead

- Problem: Managing and maintaining on-premise database servers involves a high operational overhead. This includes time-consuming tasks such as applying updates, patches, and handling routine maintenance, diverting valuable resources from more strategic activities.
- Impact: The resource-intensive nature of maintaining on-premise databases hampers the agility and efficiency of IT teams, limiting their capacity for innovation and proactive improvements.

Limited Disaster Recovery Capability

- Problem: The current on-premise environment lacks robust disaster recovery capabilities, posing a significant risk to business continuity in the event of unexpected outages, hardware failures, or other unforeseen incidents.
- Impact: The absence of a comprehensive disaster recovery strategy increases the vulnerability of critical systems, potentially resulting in extended downtime and data loss during disruptions.

Insufficient Data Synchronization

- Problem: Synchronizing data between the on-premise and cloud environments is currently a manual and error-prone process. This leads to data inconsistencies, integrity issues, and potential errors in reporting.

- Impact: Inefficient data synchronization affects the reliability and accuracy of information across different systems, introducing challenges in maintaining a single source of truth and hindering decision-making processes.

**Summary of Problems:**

- Operational Inefficiency: The on-premise database infrastructure suffers from operational inefficiencies due to manual processes, high maintenance overhead, and limited scalability. This impacts the agility of IT operations and hinders the organization's ability to swiftly adapt to evolving business demands.
- Risk to Business Continuity: The existing on-premise setup lacks robust disaster recovery capabilities, exposing the organization to the risk of prolonged downtime and potential data loss during unforeseen disruptions. This poses a significant threat to business continuity and operational resilience.
- Data Consistency and Integrity Issues: Inefficient data synchronization processes result in inconsistencies and integrity issues across on-premise and cloud environments. This compromises the reliability of critical information, impeding data-driven decision-making and operational effectiveness.

**Need For Change:**

- Enhanced Scalability for Business Growth: The organization requires a more scalable solution to support the growth of data volumes and user demands. Moving to a cloud-based model will provide the necessary scalability to ensure optimal system performance during periods of increased demand.
- Streamlined Operations and Reduced Overhead: Shifting to a cloud-based solution is imperative to streamline operations and reduce the high maintenance overhead associated with on-premise servers. This will empower IT teams to allocate resources more strategically and focus on innovation.
- Ensuring Business Continuity and Disaster Resilience: Improved disaster recovery capabilities are essential to mitigate the risk of extended downtime and data loss. Migrating to a cloud-based environment with built-in disaster recovery features will enhance business continuity and resilience.
- Achieving Efficient Data Synchronization: Implementing automated data synchronization tools is crucial for efficient and error-free data movement between on-premise and cloud databases. This change is vital to ensure data consistency and integrity, supporting accurate decision-making processes.

Overall, the identified problems underscore the critical need for a transformative change in the organization's database infrastructure. Moving to a Azure cloud-based solution not only addresses the immediate challenges but also positions the organization for enhanced scalability, operational efficiency, and resilience in the face of unforeseen disruptions. This change is not just a technological shift; it is a strategic imperative to future-proof the organization's data management capabilities and drive sustained business success.

## Deployment Method:

For our scenario Azure SQL service is good option compared to cloud hosted MS SQL Server. The reason to choose Azure SQL service are mentioned below.

- Managed Service: Azure SQL service is a fully managed platform-as-a-service (PaaS) offering. Microsoft handles routine management tasks, such as patching, backups, and high availability, reducing the operational burden on your IT team.
- Automatic Updates and Patching: Azure SQL service automatically applies updates and patches to the underlying SQL Server. This ensures that our database is always running on the latest, most secure version without manual intervention.
- Dynamic Scalability: Azure SQL service provides dynamic scalability options, allowing you to scale resources up or down based on demand. This flexibility is beneficial for handling varying workloads and optimizing costs.
- Built-in High Availability: Azure SQL service includes built-in high availability features, such as automatic backups, geo-replication, and failover capabilities. This helps ensure data durability and minimize downtime.
- Serverless Options: Azure SQL Database serverless is a unique option within Azure SQL service where you pay based on actual resource usage. It allows databases to automatically pause during inactivity, reducing costs for intermittent workloads.
- Integration with Azure Ecosystem: Azure SQL service seamlessly integrates with other Azure services that we have in our architecture, such as Azure Logic Apps,

Azure Functions, and Azure DevOps. This facilitates a more cohesive and integrated cloud solution.

After considering complexity, managed service benefits, scalability and high availability that Azure SQL service has we decided to move forward.

When considering high availability for an on-premise SQL Server in conjunction with Azure SQL services, there are several deployment options available which are described below:

SQL Server Always On Availability Groups:

- This is a high-availability and disaster-recovery solution that provides an enterprise-level alternative to database mirroring. Allows us to create a group of databases that fail over together. Can be used with on-premise SQL Server instances and can also span on-premise and Azure environments. Provides automatic failover and readable secondary replicas.

Azure SQL Database(Paas):

- This is a fully managed platform-as-a-service offering from Azure, where Microsoft takes care of most of the maintenance tasks. Offers built-in high availability with automatic backups, redundancy, and failover. Scales resources dynamically based on workload demands. Limited control over underlying infrastructure compared to SQL Server instances or virtual machines.

Azure SQL Managed Instance (PaaS):

- Similar to Azure SQL Database but provides a broader surface area compatibility with on-premise SQL Server. Offers high availability with automatic backups, redundancy, and failover. Allows more control over database settings compared to Azure SQL Database. Suitable for scenarios where we need features that are not yet supported in Azure SQL Database.

Azure SQL Virtual Machines (IaaS):

- Provides full control over the virtual machines hosting SQL Server. We can configure high availability features like Always On Availability Groups or SQL Server Failover Cluster Instances. Requires more management and maintenance

compared to PaaS options. Suitable if you have specific requirements that necessitate control over the VM configuration.

Among these options according to our need we are choosing **Azure SQL Managed Instance** which is platform as a service. Which gives us broader surface area compatibility with on-premise SQL Server.

**Connecting On-Premise Server With Azure SQL Database Service:**
Connecting an on-premise database to Azure SQL Database involves setting up a secure communication channel between the on-premise environment and the Azure cloud. Setting up connectivity between an on-premise SQL Server instance and the Azure environment using Azure ExpressRoute involves several steps. ExpressRoute provides a dedicated, private connection to Azure, which is essential for secure and efficient communication. Here are the steps to achieve this:

- Preparing On-premise Environment

  Azure ExpressRoute Circuit:

  - In the Azure portal, navigate to "Create a resource."
  - Search for "ExpressRoute" and select "ExpressRoute Circuit."
  - Follow the wizard to create an ExpressRoute circuit, providing necessary details like circuit name, SKU, and bandwidth.

  ExpressRoute Gateway:

  - Navigate to the resource group containing your ExpressRoute circuit.
  - Add a new resource, search for "ExpressRoute Gateway," and follow the wizard to create the gateway.
- Configure On-premise Network:

  On-Premise Network Setup

  - Configure on-premise routers to connect to the ExpressRoute circuit.

  ExpressRoute Peering

  - In the Azure portal, navigate to the ExpressRoute circuit.
  - Configure peering by adding our on-premise ASN (Autonomous System Number) and IP prefixes.
- Configure Azure Environment:

Azure Virtual Network:

- In the Azure portal, navigate to "Create a resource."
- Search for "Virtual Network" and create a new virtual network with the necessary settings.

ExpressRoute Connection:

- In the Azure portal, navigate to the virtual network.
- Add a new resource, search for "ExpressRoute Connection," and follow the wizard to create the connection, associating it with the ExpressRoute circuit.

Route Table:

- Configure route tables to direct traffic between on-premise and Azure networks over the ExpressRoute connection.
- Associate the route tables with the subnets in our virtual network.

- Verify Connectivity:
  - Use tools like ping or traceroute to verify basic connectivity between your on-premise network and the Azure Virtual Network.
  - Verify that you can establish connectivity from your on-premise SQL Server instance to the Azure SQL Managed Instance over the ExpressRoute connection.
  - Use SQL Server Management Studio (SSMS) to test the connection.
- Security Considerations
  - Adjust firewall rules on both on-premise and Azure SQL Managed Instance to allow traffic over the ExpressRoute connection.
  - Open the ports for SQL Server communication.
  - Configure NSGs on the Azure Virtual Network to control inbound and outbound traffic.
  - Allow traffic for SQL Server communication.

**Choose an Appropriate Target:**

To get right sized managed instance recommendation we are using Azure SQL Migration extension for Azure Data Studio. It will collect data from our SQL Server Instance to provide right sized Azure recommendations that meets our workload's performance needs with minimal cost.

**Create a Target Azure SQL Managed Instance:**

- Sign in to the azure portal
- Select Azure SQL service from services menu which is located in left side.
- Select '+ADD' to open select sql deployment option page.
- Then create SQL managed instance
- Then fill up basic details like subscription, resource group, region and create admin login credentials. For region we are selecting (US) West US2. Here we need to ensure that login credentials that we are using to connect SQL instance are member of the SYSADMIN server role.
- For compute+storage we have to select 'Configure Managed Instance'.
- Here we need to select options that we have get by doing assessment.
- Afterwards we need to fill up details for networking settings.
- Then select time zone and enable geo-replication.
- Review all the information and create managed instance.
- Monitor deployment process

**Take Backup of On-Premise Database:**

- Before starting backup process we have to install self-hosted integration runtime to access and migrate database backup.
- The link to download and authentications keys we are getting while performing migration.
- Open SQL Server Management Studio on the machine where your on-premise SQL Server is installed.
- Connect to the on-premise SQL Server instance using appropriate authentication.
- In the Object Explorer, navigate to the database we want to back up.
- Right-click on the database, go to Tasks, and select "Backup..."
- In the "Backup Database" wizard, select the backup type Full.
- Choose the destination SMB share for the backup file.
- Click "OK" to initiate the backup.
- Monitor the progress of the backup in the "Backup Progress" window.
- Once the backup is complete, we will see a success message.
- After taking backup of database we need to create outbound firewall rule. Here we have to create rules for outbound port 443, 445. Port 443 is required

to connect database migration service and 445 is for accessing network file share.

**Configure Migration Settings:**

- Select azure managed instance that we created and click next in Azure SQL Target.
- Then select offline migration.
- In Database source configuration select location of our database backup. Here we have to give information like source credentials, network share location that contains backup.
- In Firewall and Virtual Networks, select 'Enabled from selected virtual networks and IP addresses'. Then add IP address of client machine that has database backup files.
- After that next step is to create new instance for Database Migration Service.

**Create a new instance of Database Migration Service:**

- Create new resource group which contains database migration service.
- In Azure Database Migration Service select create new. Give name of service and start creating service.
- In Set up integration runtime. Select 'Download and Install Integration Runtime' link which is helps us to install integration runtime in computer that has database backup files.
- In Authentication Key table copy authentication key and paste it in Azure Data Studio. If it is valid then green check icon appears then register it.
- Then 'Test Connection' to validate Azure Managed Instance is connected to client machine that has backup files.
- After successfully testing start migration process.
- Monitor database migration process. For this in Azure Data Studio in the server menu, select Azure SQL Migration. In Database Migration Status we can track migration.

## Data-Synchronization Between On-Premise Databases and Azure Managed Instance Databases:

Data synchronization to the cloud involves keeping data consistent across on-premise databases and cloud databases, such as those hosted on platforms like Azure SQL Managed Instance. This process ensures that changes made in one environment are propagated to the other, providing a unified and up-to-date dataset. Below is a detailed explanation of the data synchronization process:

- Data synchronization is the process of maintaining the consistency of data between two or more databases, ensuring that updates, inserts, and deletes made in one database are reflected in others.
- Synchronization can be bidirectional (changes in both directions) or unidirectional (changes in one direction).
- Consideration of conflict resolution mechanisms is crucial when changes occur in both environments simultaneously.

**Architecture and Components:**

- Sync Group: A logical container defining the databases participating in synchronization. Consists of a hub and one or more members.
- Hub Databases: Centralized database that acts as the primary source for synchronization. In Azure Data Sync, it's often an Azure SQL Database.
- Members: Databases participating in synchronization. In our case databases on on-premise server and in Azure Managed Instance.
- Sync Agent: An agent installed on-premise to facilitate communication between on-premise databases and the Azure cloud.

**Setting Up Data Synchronization:**

- Azure Portal Configuration:
  - Select SQL Managed Instance. Open Data Sync pane using ‘Settings’ then ‘Data + sync’.
  - Provide a name for the sync group and choose the synchronization direction as a Bi-directional, To hub, From hub.
  - Add on-premise database. Add a sync member by selecting "Add database". Provide connection details for our on-premise SQL Server.
  - Add the Azure SQL Database as a sync member.

    - Configure the sync schema by selecting the tables to be synchronized.
    - Review the settings and click "Create" to create the sync group.
- Configure On-Premise Environment:
    - Download and Install Sync Agent: In the Azure portal, under "Sync members" for the on-premise database, download the Azure Data Sync Agent. Install the sync client agent on a machine in your on-premise environment.
    - In sync client agent app select Submit Agent Key. The Sync Metadata Database Configuration dialog box opens.
    - In the box we have to paste key that we copied from azure portal and give credentials. Then click ‘OK’.
    - Click “Register” to register SQL Server database with agent.
    - In SQL Server Configuration dialog box select windows authentication and give server name and database name.
    - Select ‘Test Connection’ to test our connection. After successfully testing save it.

**Initial Data Synchronization:**

- Data from the hub is transferred to the destination.
- In the Azure portal, go to sync group and start the sync group.
- Monitor the synchronization progress in the Azure portal.

**Scheduled Synchronization:**

- Synchronization is automated based on the defined schedule. Changes in one environment trigger synchronization to update others.
- In the Azure portal, under sync group, configure the sync schedule based on every 5 minutes.
- Then monitor sync operations and view sync logs.

Data synchronization to the cloud is a critical aspect of maintaining consistency and coherence across distributed databases. By following best practices, configuring synchronization settings, and monitoring the process, organizations can ensure that our data remains accurate and up-to-date across on-premise and cloud environments

## Process For The Business Users Point to The Cloud-based Solution When The On-Premise Environment is Down:

To get high availability, the process for redirecting business users during on-premises downtime involves specific considerations related to Azure features. Here's an adapted step-by-step process for a failover scenario to Azure Managed Instance:

**Azure Traffic Manager:**
Configuring Azure Traffic Manager to distribute traffic between on-premises and Azure Managed Instance environments involves several steps. Azure Traffic Manager is a DNS-based traffic load balancer that enables us to control the distribution of user traffic for service endpoints in different data centers.

**Steps to Configure Azure Traffic Manager:**

- In the left-hand menu, click on "All services," then type "Traffic Manager" in the search bar. Click on "Traffic Manager profiles."
- Click on "+ Add" to create a new Traffic Manager profile and fill up required details.
- Add endpoints. Under the "Settings" section, click on "Endpoints." Click on "+ Add" to add a new endpoint.
- Choose "External endpoint" for on-premises and "Azure endpoint" for Azure Managed Instance. Fill in the required information for each endpoint, including the DNS name or IP address.
- For each endpoint select protocol as HTTPS, ports, path, interval as 30 seconds.
- Under the "Settings" section, click on "Health probes." Configure health probes for on-premises and Azure Managed Instance based on our application's requirements.
- Review all settings on the summary page. Click "Review + create" and then "Create" to create the Traffic Manager profile.
- Once the Traffic Manager profile is created, navigate to the profile.
- Under the "Settings" section, click on "Properties." Note the DNS name assigned to your Traffic Manager profile. Update our application's DNS records to point to the Traffic Manager DNS name.

- Test the configuration by accessing our application using the Traffic Manager DNS name. Monitor Traffic Manager's behavior by checking the Traffic Manager profile metrics and logs.

By following these steps, we can configure Azure Traffic Manager to distribute traffic between on-premises and Azure Managed Instance environments, ensuring efficient load balancing and failover capabilities.

## Back-up and Recovery Procedures for User Connectivity and Data Integrity:

Synchronizing an on-premise SQL Server with an Azure SQL Managed Instance involves a combination of backup and recovery strategies along with data synchronization processes. This ensures user connectivity and data integrity between the on-premise environment and the Azure cloud.

To start, implement a robust backup strategy for the on-premise SQL Server. Begin with regular daily full database backups. These backups provide a complete snapshot of the database and scheduled during periods of low user activity to minimize impact. Additionally, employ differential backups, capturing changes since the last full backup, and transaction log backups, which record incremental changes made to the database. Regularly test the restore process to ensure the backups are reliable and can be quickly recovered if needed.

**Steps for Backup:**

- Full Backup:
  - Open SQL Server Management Studio (SSMS)
  - In Object Explorer, expand the server node, then expand the "Databases" node.
  - Right-click on the database for which want to perform a backup.
  - Select "Tasks" > "Backup" from the context menu.
  - In the "Backup Database" wizard, choose the destination hard disk for the backup.
  - Set the backup type to "Full."
  - Review the summary of your selections on the "Summary" page.

- Click "OK" to execute the backup.

- Differential Backup:
  - In SSMS, go to Object Explorer, expand the server node, then expand the "Databases" node.
  - Right-click on the database for which want to perform a differential backup.
  - Select "Tasks" > "Backup" from the context menu.
  - Choose the destination for the backup.
  - Set the backup type to "Differential" this time.
  - Review the summary on the "Summary" page.
  - Click "OK" to execute the backup.

- Transaction Log Backup:
  - In SSMS, go to Object Explorer, expand the server node, then expand the "Databases" node.
  - Right-click on the database for which want to perform a transaction log backup.
  - Select "Tasks" > "Backup" from the context menu.
  - Choose the destination for the backup.
  - Set the backup type to "Transaction Log" this time.
  - Review the summary on the "Summary" page.
  - Click "OK" to execute the backup.

- Schedule Daily Backup:
  - In Object Explorer, expand the server node, then expand the "SQL Server Agent" node.
  - Right-Click on "Jobs" and Select "New Job...".
  - In General Tab set name as “Daily_Backup_Job”. Ensure that the "Enabled" checkbox is selected.
  - Click on the "Schedules" tab within the "New Job" dialog.
  - Click "New" to create a new schedule.
  - Set the frequency as Daily. Specify the start date and time for the job.
  - Click on the "Steps" tab within the "New Job" dialog. Click "New" to add a new job step. Enter a step name in the "Step name" field.
  - In the "Type" dropdown, select "Transact-SQL script (T-SQL)."

- In the "Database" dropdown, select the database you want to back up.

T-SQL Script:

" -- Full Database Backup

BACKUP DATABASE [DatabaseName]

TO DISK = "Destination_Path"

WITH INIT, FORMAT, COMPRESSION;

-- Differential Backup

BACKUP DATABASE [DatabaseName]

TO DISK = "Destination_Path"

WITH DIFFERENTIAL, INIT, COMPRESSION;

-- Transaction Log Backup

BACKUP LOG [DatabaseName]

TO DISK = "Destination_Path"

WITH INIT, COMPRESSION; "

- Click "OK" to close the "New Job Step" dialog. Click "OK" again to close the "New Job" dialog.

**Steps for Recovery:**

- Launch SSMS and connect to our SQL Server instance.
- In Object Explorer, expand the server node, then expand the "Databases" node.
- Right-click on the database want to restore and select "Tasks" > "Restore" > "Database..."
- In the "General" page of the Restore Database wizard, select the source of the backup. Choose "Device" and add the backup file from backup location.
- In the "Select a page" pane, navigate to the "Backup sets to restore" page. Choose the backup sets to restore.
- Go to the "Options" page of the wizard. Specify the destination for the restore.
- Review your selections on the "Summary" page. Click "OK" to initiate the restore process.
- Monitor the progress of the restore operation in the "Progress" window.

- Once the restore operation is complete, review the final status in the "Results" tab. Verify that the database has been restored successfully.
- After the restore, run a DBCC CHECKDB command to check the integrity of the restored database.

By following these steps, we can successfully perform a database recovery in SQL Server. Always exercise caution during restore operations, especially in a production environment, and we need to ensure that we have tested your recovery procedures in a controlled environment.

**Azure Managed Instance Backup and Recovery:**
The high availability features of Azure SQL Managed Instance work together to ensure that your databases are resilient to failures, minimize downtime, and provide redundancy. Here's a detailed working process of the key elements:

Automatic Backups:

- Azure SQL Managed Instance automatically performs regular backups of our databases. These backups include full database backups and transaction log backups. Backup data is stored in Azure Storage, providing durability and availability even in the case of hardware failures.
- Backup data is geo-redundant, meaning it's stored in multiple Azure regions. This provides additional protection against regional outages, ensuring that we can restore our databases from backups stored in a different region.

Automatic Failover Groups:

- Automatic failover groups allow us to configure a primary and one or more readable secondary replicas. In the event of a failure or planned maintenance, the system can automatically failover to a secondary replica, minimizing downtime.

Geo-Replication:

- Geo-replication allows us to create readable replicas of our managed instance in different Azure regions. This feature enhances disaster recovery capabilities and provides low-latency access to data for users in different geographic locations.

# CONCLUSION

In our hybrid architecture combining on-premise SQL Server with Azure Managed Instance, for scenario when on-premise servers face downtime due to updates or outages, Azure Managed Instance steps in as a reliable cloud-based alternative, offering automatic failover, geo-redundant backups, and seamless replication. The integration of Azure Traffic Manager is pivotal for distributing and managing request traffic, providing load balancing and failover capabilities, thereby optimizing resource utilization. The connection between the on-premise environment and Azure Managed Instance is established through Azure ExpressRoute, a dedicated and secure link ensuring data privacy and low-latency access. This connectivity infrastructure is fundamental for establishing a seamless bridge between the on-premise and cloud environments. Our database migration strategy is orchestrated through Azure Database Migration Service, streamlining the migration process while prioritizing data integrity. Full, differential, and transaction backups are systematically stored in a shared SMB location, ensuring comprehensive data protection. The migration process is facilitated by Azure Integration Runtime, a powerful tool providing access to database backup files within the Azure portal. This integration runtime application on client machines enables smooth migration workflows and seamless interaction between the on-premise and cloud environments. For data synchronization between the on-premise SQL Server and Azure Managed Instance, we deploy a sync agent on the on-premise server. This sync agent works at a 5-minute interval, ensuring that changes are promptly reflected in both environments. This approach is critical for maintaining consistency and minimizing data discrepancies. Daily backup routines for the on-premise database are orchestrated through job creation, ensuring regular data protection. Meanwhile, Azure Managed Instance plays a pivotal role in enhancing system reliability by offering automatic backups, replication, and failover mechanisms. These features collectively contribute to data resilience and system availability.Our hybrid solution is designed to not only address potential downtime scenarios through Azure Managed Instance but also to enhance data management, security, and scalability. The integration of Azure Traffic Manager, Azure ExpressRoute, Azure Database Migration Service, and meticulous backup strategies ensures a robust, seamless, and well-managed coexistence of on-premise and cloud-based components within our overarching architecture.

# References


C. (2023, January 13). *SQL Server to Azure SQL Database: Migration guide - Azure SQL Database*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/migration-guides/database/sql-server-to-sql-database-guide?view=azuresql

C. (2023, October 11). *Azure SQL migration extension for Azure Data Studio - Azure Data Studio*. Microsoft Learn. https://learn.microsoft.com/en-us/azure-data-studio/extensions/azure-sql-migration-extension

A. (2023, June 8). *Tutorial: Migrate SQL Server to Azure SQL Managed Instance online by using Azure Data Studio - Azure Database Migration Service*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/dms/tutorial-sql-server-managed-instance-online-ads

Z. R. M. (2023, June 15). *Connect your application to SQL Managed Instance - Azure SQL Managed Instance*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/managed-instance/connect-application-instance?view=azuresql

C. (2023, January 9). *SQL Server to SQL Managed Instance: Migration overview - Azure SQL Managed Instance*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/migration-guides/managed-instance/sql-server-to-managed-instance-overview?view=azuresql

A. (2023, October 10). *Migrate databases by using the Azure SQL Migration extension for Azure Data Studio*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/dms/migration-using-azure-data-studio?tabs=azure-sql-mi

U. (2023, March 3). *Quickstart: Create an Azure SQL Managed Instance (portal) - Azure SQL Managed Instance*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/managed-instance/instance-create-quickstart?view=azuresql

W. (2023, August 21). *Set up SQL Data Sync - Azure SQL Database*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-data-sync-sql-server-configure?view=azuresql

W. (2023, November 28). *What is SQL Data Sync for Azure? - Azure SQL Database*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-data-sync-data-sql-server-sql-database?view=azuresql

Policht, M. (2021, December 13). *Implementing Azure SQL Data Sync in the Azure Portal*. Database Journal. https://www.databasejournal.com/ms-sql/implementing-azure-sql-data-sync-in-the-azure-portal/

S. (2023, May 11). *Schedule a database backup operation using SSMS - SQL Server*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/relational-databases/backup-restore/schedule-database-backup-operation-ssms?view=sql-server-ver16

E. M. (2023, June 16). *Restore a Database Backup Using SSMS - SQL Server*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/relational-databases/backup-restore/restore-a-database-backup-using-ssms?view=sql-server-ver16

S. (2023, July 12). *Automatic, geo-redundant backups - Azure SQL Managed Instance*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/managed-instance/automated-backups-overview?view=azuresql

S. (2023, June 23). *Restore a database from a backup - Azure SQL Managed Instance*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/managed-instance/recovery-using-backups?view=azuresql&tabs=azure-portal

Z. R. M. (2023, June 15). *Connect your application to SQL Managed Instance - Azure SQL Managed Instance*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/managed-instance/connect-application-instance?view=azuresql

*Announcing link feature for Managed Instance now in public preview*. (n.d.). TECHCOMMUNITY.MICROSOFT.COM. https://techcommunity.microsoft.com/t5/azure-sql-blog/announcing-link-feature-for-managed-instance-now-in-public/ba-p/3259822

*Connect to Azure Sql Managed Instance from on-premise SSMS*. (n.d.). Stack Overflow. https://stackoverflow.com/questions/70144357/connect-to-azure-sql-managed-instance-from-on-premise-ssms