MS SQL Server DBA

Praveen Kumar M
Mb: +91 986 613 0093 (Botim\WhatsApp)
+91 819 729 3434 (WhatsApp)
Mail: praveensqldba12@gmail.com
LinkedIn: https://www.linkedin.com/in/praveenmadupu
Github: https://github.com/praveenmadupu
Youtube: https://www.youtube.com/PraveenMadupu

DB LogShipping Architecture

DB Log Shipping Architecture in Detail

Log Shipping is a high-availability and disaster recovery solution in SQL Server. It involves automating the process of sending transaction log backups from one database (the primary) to another (the secondary) and restoring these logs at the secondary location to keep the secondary database synchronized with the primary. Log shipping is commonly used for **disaster recovery** purposes, but it can also be used for **readable secondary replicas** to offload read workloads.

Let’s dive into the architecture of Log Shipping in SQL Server.

Key Components of Log Shipping

1. **Primary Server (or Primary Database)**
    - This is the source database, the database that is actively being used in the production environment.
    - Transaction logs are generated from this database and backed up at regular intervals.
2. **Secondary Server (or Secondary Database)**
    - The secondary server holds the database that is a copy of the primary database.
    - It restores the transaction log backups sent by the primary server. The secondary database is typically in **read-only** mode for reporting purposes (unless in **standby mode**, which allows for limited read/write operations).
3. **Backup Job**
    - This job is responsible for taking **transaction log backups** of the primary database at regular intervals.
    - It can be scheduled to run based on the organization's Recovery Point Objective (RPO), typically ranging from every 10 minutes to every hour.
4. **Copy Job**
    - The copy job is responsible for **copying** the transaction log backups from the primary server to the secondary server.
    - These backups are transferred across the network to the secondary server using a shared folder or a network location.
5. **Restore Job**
    - The restore job is responsible for **restoring** the transaction log backups on the secondary database.
    - This job applies the transaction log backups to the secondary database to keep it in sync with the primary database.
    - The restore job typically runs in **NORECOVERY** mode, meaning the database remains in a "restoring" state until the next transaction log is applied.

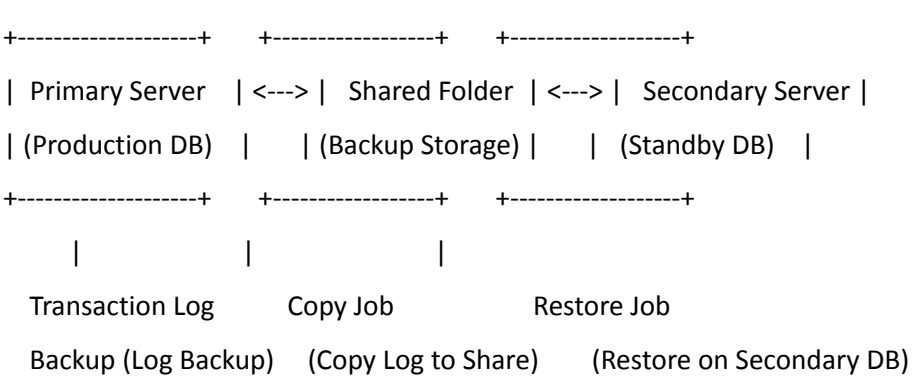
Log Shipping Workflow

The basic workflow of Log Shipping involves three main steps:

1. **Transaction Log Backup (on Primary Server):**
    - On the primary server, a **transaction log backup** of the active database is taken.
    - This backup is stored in a specific directory or network share accessible by both the primary and secondary servers.
2. **Copying the Log Backup (from Primary to Secondary):**
    - The log backup file is copied from the primary server to the secondary server's predefined shared directory.
    - This step is handled by the **copy job** that runs on the primary server, and it ensures the logs are transferred to the secondary server efficiently.
3. **Restoring the Log Backup (on Secondary Server):**
    - The copied transaction log file is then **restored** on the secondary server using the **restore job**.
    - The restore operation applies the transaction log changes on the secondary database. This ensures that the secondary database is as up-to-date as the primary database.
    - The secondary database typically remains in **RESTORE** mode to avoid user access but is updated with all changes.

Log Shipping Architecture Diagram

Below is a simplified representation of the architecture:



Detailed Process Flow

1. Transaction Log Backup on Primary Server

- The primary server performs **transaction log backups** at regular intervals.
    - These backups contain all the transactions that have occurred since the last transaction log backup.
    - The **Backup Job** typically runs every 10-15 minutes or as per business requirements.
- **Transaction Log File**: The log backup file contains **transaction log records** of all changes (inserts, updates, deletes) made in the primary database since the last log backup.

2. Copying Log Backups to the Secondary Server

- The **Copy Job** copies the transaction log backups from the **primary server's backup folder** to a shared network folder or a specified directory on the secondary server.
- The **Shared Folder** must be accessible to both the primary and secondary servers to ensure seamless file transfers.

3. Restoring Log Backups on the Secondary Server

- On the secondary server, the **Restore Job** picks up the log backup files copied from the shared folder.
- The secondary database is restored with the transaction log using **NORECOVERY** mode to ensure that the database is kept in a "restoring" state and remains available for additional transaction logs.
    - In **NORECOVERY** mode, the database is not available for normal user queries and operations, but it can continue to receive additional log backups.

4. Repeat the Process

- This cycle of **backup**, **copy**, and **restore** continues at regular intervals.
- Over time, the secondary database will stay in sync with the primary database and reflect all the changes made in the primary database.

Log Shipping Modes

- **Standby Mode**:
    - In **standby mode**, the secondary database is restored and kept in a **read-only** mode, but **log shipping can still be applied**.
    - Users can access the secondary database for **read-only queries**, but the database is in a state where further transaction logs can still be applied.
- **NORECOVERY Mode**:
    - The **secondary database** is in a "restoring" state, meaning that no transactions can be executed on it (no queries, no updates). It is **not accessible to end users** until the log shipping process is complete

or the database is fully restored.
    - This mode allows for the **continuous application of transaction logs** without interference.
- **RECOVERY Mode**:
    - The **secondary database** is **fully online** and can be accessed like a regular database.
    - However, once in **RECOVERY mode**, log shipping stops, and transaction logs cannot be applied.

Log Shipping Configuration Considerations

1. **Backup Frequency**:
    - The frequency of the transaction log backups depends on the business's **RPO (Recovery Point Objective)**. Smaller backup intervals (e.g., every 10 minutes) provide near-real-time recovery.
    - Backup frequency should be balanced to ensure timely synchronization without impacting server performance.
2. **Backup Storage**:
    - Backup storage for transaction logs must be reliable and easily accessible for both the primary and secondary servers.
    - It is common to store transaction logs in a shared network folder or in cloud storage.
3. **Monitoring**:
    - **SQL Server Management Studio (SSMS)** provides built-in monitoring capabilities for Log Shipping.
    - Monitoring involves checking the status of the **Backup Job**, **Copy Job**, and **Restore Job** to ensure all steps are completing successfully.
    - Alerts can be configured to notify administrators if any job fails or if there is a significant delay in applying transaction logs.
4. **Recovery Time Objective (RTO)**:
    - Log shipping can be used to meet an organization’s **RTO** by setting up the restore job to apply transaction logs at appropriate intervals.
    - If the primary server fails, the secondary server can be brought online with minimal data loss (depending on the frequency of the transaction log backups).
5. **Failover Process**:
    - Log shipping does **not provide automatic failover**. If the primary database becomes unavailable, the administrator must manually failover to the secondary server by:
        1. Restoring the database on the secondary server using the final log backup.
        2. Bringing the database online on the secondary server.
    - In some configurations, SQL Server **AlwaysOn Availability Groups** or **Database Mirroring** can be used for automatic failover.

Advantages of Log Shipping

1. **Cost-Effective**:
    - Log shipping is a relatively low-cost solution for high availability, as it does not require additional licensing or hardware beyond a secondary server.
2. **Disaster Recovery**:
    - Log shipping provides a strong disaster recovery solution by allowing you to keep a warm standby server that can quickly be brought online if the primary server fails.
3. **Offloading Read Workload**:
    - The secondary database can be used for **reporting and read-only queries**, thereby offloading the primary database and improving performance for production systems.
4. **Simple to Configure and Manage**:
    - Setting up log shipping is straightforward, and SQL Server provides built-in support for monitoring and configuring log shipping jobs.

Disadvantages of Log Shipping

1. **Manual Failover**:
    - Unlike solutions like **AlwaysOn Availability Groups** or **Database Mirroring**, **log shipping** does not provide automatic failover. Failover needs to be done manually, which can result in increased downtime.
2. **No Active-Active Configuration**:
    - Log shipping is typically **active-passive**: the secondary server is a read-only replica unless promoted to the primary. It does not support an **active-active** configuration.
3. **Lag Time**:
    - Depending on the interval between transaction log backups, there can be a **lag** between the primary and secondary servers, meaning that the secondary database might not be perfectly in sync with the primary at all times.
4. **Complexity in Management**:
    - Managing the backup, copy, and restore jobs and troubleshooting any issues can add complexity over time, especially in larger environments.

Summary :

Log Shipping is a simple and effective way to maintain a standby database for disaster recovery purposes in SQL Server. It works well for organizations that need to ensure minimal data loss while maintaining a readable secondary database. However, it requires careful configuration, ongoing monitoring, and manual failover intervention. If you require automatic failover or more advanced high-availability features, other solutions like **AlwaysOn Availability Groups** or **Database Mirroring** may be more appropriate.