**Database Log Shipping** in SQL Server is a high-availability and disaster recovery solution that involves automatically sending transaction log backups from one SQL Server (the primary server) to one or more secondary servers. The transaction logs are then restored to the secondary servers.

The primary goal is to provide a standby copy of a database in case the primary database becomes unavailable due to a failure.

It can also be used for reporting purposes by offloading read operations to secondary servers.

### **Key Concepts of Database Log Shipping:**

1. **Primary Database (Source):**
   * This is the active, online database where the transactions occur. The primary database is the "source" of the transaction log backups.
2. **Secondary Database (Destination):**
   * One or more databases on secondary servers that receive and restore the transaction log backups from the primary database. The secondary database can be in a **standby** or **norecovery** mode.
3. **Transaction Log Backups:**
   * The process involves taking regular transaction log backups from the primary database. These log backups capture all the changes made to the database (inserts, updates, deletes).
4. **Copying and Restoring Logs:**
   * The transaction log backups are copied from the primary server to a shared network folder that is accessible by both the primary and secondary servers.
   * On the secondary server, these log backups are restored to the secondary database to keep it synchronized with the primary database.

### **Log Shipping Architecture:**

The log shipping process consists of three main steps:

1. **Backup:**
   * Transaction log backups are created on the primary server. These backups contain all the changes made to the database since the last transaction log backup.
2. **Copy:**
   * The backup files are copied from the primary server to a shared location (usually a network folder) accessible by both the primary and secondary servers.
3. **Restore:**
   * The secondary server restores the transaction log backups to its own copy of the database. This process brings the secondary database closer to the state of the primary database by applying the log changes.

### **Modes of Secondary Database:**

There are three possible recovery modes for the secondary database in log shipping:

1. **NORECOVERY Mode:**
   * The secondary database is in a "restoring" state and is not accessible for querying. The database is updated with the latest transaction log backup from the primary database.
   * This is the default mode for most environments, especially if the secondary database is used purely for failover or disaster recovery.
2. **STANDBY Mode:**
   * The secondary database is available for read-only queries, but you cannot make any changes. The database is in a "restoring" state, but you can use it for reporting or other non-transactional workloads while transaction logs are being restored.
   * It allows users to run read-only queries against the secondary database while the restore process occurs.
3. **RECOVERY Mode (Not typically used in Log Shipping):**
   * The secondary database is fully operational and available for writing. However, once the transaction logs are restored, the database is in a final, recovered state and will no longer receive further log shipping transactions until restored again.
   * This mode is not typically used in traditional log shipping setups, as the point of log shipping is to continuously apply log backups.

### **Steps to Configure Log Shipping:**

1. **Prepare the Primary Database:**
   * Ensure the database is in **Full Recovery Model**. Log shipping relies on transaction log backups, which are only possible in the Full Recovery Model.
   * Take a full backup of the primary database and restore it with the NORECOVERY option on the secondary server.
2. **Set Up the Backup Job on the Primary Server:**
   * Create a SQL Server Agent job on the primary server that will regularly take transaction log backups of the database.
   * These backups will be copied to a shared folder.
3. **Set Up the Copy and Restore Jobs on the Secondary Server:**
   * On the secondary server, set up SQL Server Agent jobs to:
     + **Copy** the transaction log backups from the shared network folder to the secondary server.
     + **Restore** the transaction log backups onto the secondary database, keeping it synchronized with the primary.
4. **Monitor and Maintain Log Shipping:**
   * SQL Server provides a **log shipping monitor** that tracks the health and status of the log shipping process. This allows you to monitor if the transaction logs are being backed up, copied, and restored on schedule.
5. **Perform Manual Failover:**
   * In the event of a failure on the primary server, you can manually fail over to the secondary server by restoring the last transaction log backup on the secondary server, bringing it online for use. However, **automatic failover** is not supported natively in SQL Server log shipping.

### **Components of Log Shipping:**

1. **Primary Server:**
   * The source server that backs up the transaction logs and is responsible for initiating the log shipping process.
2. **Secondary Server(s):**
   * One or more destination servers that receive and restore transaction log backups from the primary server.
3. **Shared Network Folder:**
   * A shared folder (accessible by both the primary and secondary servers) where transaction log backups are copied and stored temporarily before being restored on the secondary server(s).
4. **SQL Server Agent Jobs:**
   * These jobs automate the process of backing up, copying, and restoring transaction logs. There are three main jobs:
     + **Backup Job** (on the primary server): Takes transaction log backups.
     + **Copy Job** (on the secondary server): Copies transaction log backups from the shared folder.
     + **Restore Job** (on the secondary server): Restores the transaction log backups to the secondary database.

### **Advantages of Log Shipping:**

1. **Simple and Low-Cost Solution:**
   * Log shipping is relatively simple to set up and requires minimal additional hardware or infrastructure.
   * It does not require additional SQL Server licenses for secondary servers in some configurations (secondary servers can be used for reporting or disaster recovery).
2. **Disaster Recovery:**
   * Provides a standby server that can be used for disaster recovery if the primary server fails.
3. **Offload Reporting:**
   * The secondary database can be used for read-only reporting purposes, which helps offload query workloads from the primary server.
4. **Data Protection:**
   * It helps protect against data loss by maintaining a near real-time copy of the database on the secondary server.

### **Disadvantages of Log Shipping:**

1. **Manual Failover:**
   * Log shipping does not support automatic failover. In case of a failure on the primary server, a manual intervention is required to promote the secondary server to be the new primary.
2. **Lag Between Primary and Secondary:**
   * There is always a delay between when a transaction occurs on the primary server and when the changes are reflected on the secondary server (due to the log shipping interval). This delay can range from minutes to hours, depending on how frequently log backups are taken.
3. **Storage Management:**
   * Transaction logs need to be managed carefully. If logs are not cleaned up, the secondary servers may run out of disk space.
4. **No Load Balancing:**
   * Unlike solutions such as Always On Availability Groups, log shipping does not provide load balancing. The primary server is still the only one handling writes, and the secondary is typically only used for read-only operations.
5. **No Native Support for Transparent Data Encryption (TDE):**
   * Log shipping has limited support for Transparent Data Encryption (TDE) across the primary and secondary servers.
6. **Possible Data Loss:**
   * In the case of a primary server failure before the last log backup is restored, there is a risk of data loss, as the secondary server will not have the most recent transaction logs.

### **Conclusion:**

Log shipping is a useful disaster recovery solution that keeps a warm standby copy of a database on a secondary server. It provides a simple and low-cost alternative for ensuring high availability, but it lacks automatic failover, real-time synchronization, and some features found in more advanced solutions like Always On Availability Groups or SQL Server Clustering. It is best suited for environments that can tolerate a small amount of data latency and require a relatively simple failover mechanism.