Praveen Madupu - +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

A **Log Shipping Disaster Recovery Drill** in SQL Server is a process where the failover to the secondary (DR) server is tested to ensure that your disaster recovery setup is functioning properly.

This drill helps confirm that log shipping is configured correctly, that the secondary server can be brought online in the event of a failure, and that the primary server's databases can be restored on the secondary server with minimal data loss.

This process typically involves stopping log shipping, failing over the databases to the secondary server, running a set of validation tests, and then optionally returning the setup to the original configuration.

Here is a step-by-step guide to performing a log shipping disaster recovery drill:

1. Overview of Log Shipping Components

Log shipping involves the following key components:

- **Primary Server**: The SQL Server instance where the database is actively running.
- **Secondary Server**: The SQL Server instance where backups from the primary server are restored periodically.
- **Monitor Server (Optional)**: Tracks the status and history of log shipping (optional but useful for centralized monitoring).

In a disaster recovery (DR) scenario, the **secondary server** must be ready to take over from the primary server.

2. Preparation

a. Ensure Backups are Up-to-Date

Before the DR drill begins, ensure that the most recent transaction log backup from the primary server has been copied and restored on the secondary server. This ensures minimal data loss during the failover.

- Check the status of log backups, copies, and restores in the **Log Shipping Status Report**.
- Ensure that all log backups have been successfully applied to the secondary server.

b. Communication

Inform stakeholders (DBAs, application owners, end-users) about the upcoming drill. It’s important to schedule this drill during non-business hours or a maintenance window to avoid disruption.

Praveen Madupu - +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

3. Steps for the Log Shipping DR Drill

Step 1: Stop User Activity on the Primary Server

To avoid potential data inconsistency, stop all user activity on the primary database before failing over to the secondary server.

1. Prevent new connections by setting the primary database to **single-user mode**:

   ALTER DATABASE [YourDatabaseName] SET SINGLE_USER WITH ROLLBACK IMMEDIATE;

2. Make sure all connections are terminated to ensure no new transactions are being processed.

Step 2: Perform a Final Transaction Log Backup on the Primary Server

Perform a final transaction log backup on the primary database and restore it on the secondary server to ensure minimal data loss.

1. **Back up the transaction log** on the primary server:

   BACKUP LOG [YourDatabaseName] TO DISK = 'C:\Backup\YourDatabaseName_FinalLog.trn';

2. **Copy the log backup** file to the secondary server.

Step 3: Restore the Final Transaction Log on the Secondary Server

Apply the final transaction log backup on the secondary server to bring the database fully up to date.

1. On the secondary server, **restore the final log backup**:

   RESTORE LOG [YourDatabaseName]

   FROM DISK = 'C:\Backup\YourDatabaseName_FinalLog.trn'

   WITH NORECOVERY;

   - The `NORECOVERY` option keeps the secondary database in a **restoring state**, ready for the final failover step.

Step 4: Bring the Secondary Database Online

Once all log backups have been applied, the secondary database is almost ready to be brought online.

1. Switch the secondary database to **read-write mode** by recovering it:

   RESTORE DATABASE [YourDatabaseName] WITH RECOVERY;

Praveen Madupu - +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

- This command brings the secondary database online, making it the new primary database.

Step 5: Redirect Application Connections to the Secondary Server

Update the connection strings in your application to point to the secondary server. Ensure that application services and users are connected to the new server.

4. Validation After Failover

a. Validate Database Integrity

Check the integrity of the database on the secondary server using `DBCC CHECKDB`:

```
DBCC CHECKDB('YourDatabaseName');
```

Ensure that there are no corruption issues.

b. Verify Data Consistency

Verify that data on the secondary server matches the data on the primary server. This may include running test queries, comparing row counts, or validating application functionality.

c. Run Functional Tests

Coordinate with the application teams to perform functional tests on the secondary database. This can include:

- Testing key functionalities of the application.
- Running reports to ensure data correctness.
- Validating stored procedures, triggers, and jobs.

d. Check Log Shipping Status

Review the log shipping status on the secondary server by using the **Log Shipping Status Report**. Ensure that the copy and restore jobs are functioning properly and that there are no errors.

Praveen Madupu - +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

5. Post-Drill: Return to the Original Setup (Optional)

If the drill is successful, you may want to return to the original configuration where the primary server is the active server and the secondary server resumes its role as the standby.

Option 1: Make the Secondary Server the New Primary (Permanent Failover)

If you want to permanently failover to the secondary server, skip this step.

Option 2: Return to the Primary Server (Failback)

1. **Set Up Log Shipping from the Secondary to the Primary Server** After failing over to the secondary server, set up log shipping in reverse, so the former primary server becomes the secondary.
2. **Backup Databases from the New Primary Server (Old Secondary)** Perform a full backup of the databases on the new primary server (formerly the secondary server):

   ```
   BACKUP DATABASE [YourDatabaseName] TO DISK = 'C:\Backup\YourDatabaseName_Full.bak';
   ```

3. **Restore the Full Backup on the Old Primary Server (Now Secondary)**

Restore the full backup on the old primary server using the `NORECOVERY` option:

```
RESTORE DATABASE [YourDatabaseName]
FROM DISK = 'C:\Backup\YourDatabaseName_Full.bak'
WITH NORECOVERY;
```


4. **Reinitialize Log Shipping from the New Primary to the Old Primary**

After restoring the full backup, reinitialize the log shipping configuration to copy logs from the new primary to the old primary server.

6. Monitoring and Logging

During and after the DR drill, it's important to document the following:

- Time taken for each step.
- Any issues or failures encountered.
- Validation and testing results.
- Final database state after the failover.

This documentation will help improve future DR drills and serve as a reference in the event of an actual disaster.

Praveen Madupu - +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

7. Key Points to Remember

- **Minimize Data Loss**: Ensure that the final transaction log backup from the primary is applied to the secondary server to minimize data loss.
- **Recovery Models**: Ensure that the database is in **Full Recovery Mode** to use transaction log backups.
- **Testing**: Conduct both **data validation** and **application testing** to ensure that the secondary server is fully functional.
- **Failback**: Plan how you will either stay on the secondary server or fail back to the original primary server after the DR drill.

Summary of the Process

1. **Stop user activity** on the primary server.
2. **Perform a final transaction log backup** and restore it on the secondary server.
3. **Bring the secondary database online** using `WITH RECOVERY`.
4. **Redirect applications** to the secondary server.
5. **Validate data and functionality** on the secondary server.
6. (Optional) **Failback** to the original primary server or stay on the secondary.

By performing these steps during a log shipping disaster recovery drill, you can ensure your SQL Server environment is ready for a real-world disaster scenario, minimizing downtime and data loss.