https://www.sqldbachamps.com
https://github.com/PMSQLDBA/PraveenMadupu
Telegram Group→SQLDBACloud-DevOps https://t.me/+r2OYsT2qRZJkYWVl

Praveen Madupu
Mb: +91 98661 30093
Database Administrator

🧩 SQL Server Log Shipping Overview

Log Shipping is a high-availability and disaster-recovery feature where:

1. **Transaction logs** are periodically backed up from the primary database.
2. Logs are copied to one or more secondary servers.
3. Logs are restored on secondary databases, keeping them synchronized.

Key components:

Component	Description
Primary Database	Source database whose logs are shipped.
Secondary Database	Target database where logs are restored.
Log Backup Job	Backs up the transaction log on primary.
Copy Job	Copies the backup file to secondary server.
Restore Job	Restores the backup on the secondary database.

⚠ Common Scenario: Log File Fills Up

In SQL Server, **transaction log can grow uncontrollably** when:

- Log shipping jobs fail or are delayed.
- Large transactions are not being backed up.
- Database is in **Full recovery model** and log backups are not happening frequently.
- Long-running transactions prevent truncation of log.
- Disk space on primary server is insufficient.

🧠 Symptoms of Transaction Log Full in Log Shipping

Symptom	DMV / Tool	Observation
Log file grows rapidly	DBCC SQLPERF(LOGSPACE)	Shows % used in log file
Log backup job failed	SQL Server Agent Job History	Job failed with “Cannot back up log because it is full”
Log shipping secondary not updated	Log Shipping Status Monitor	Pending logs accumulate
Slow primary database	SSMS Activity Monitor / Wait Stats	High LOG_RATE waits

🛠 Common Causes of Log File Full in Log Shipping

1. **Log backup job failure**
   - Transaction log cannot be truncated → log grows.
   - Example: SQL Server Agent service stopped, or job disabled.
2. **Network issues / Copy Job failure**
   - Backup file not copied to secondary → pending logs accumulate.
   - Disk full on secondary server → restores fail → primary log cannot truncate.
3. **Restore Job delay / restore pending**
   - Secondary database in restoring mode, but restore job is failing or delayed.
   - Causes log backups to pile up on primary.
4. **Long-running transactions**
   - Open transactions prevent log truncation.
   - DBCC OPENTRAN shows active transactions.
5. **Large bulk operations**
   - Massive inserts, updates, deletes in primary.
   - Log grows quickly before log backup occurs.
6. **Disk space limitation**
   - Primary log drive has limited space → SQL Server cannot grow log file → job fails.

7. **Incorrect Log Shipping configuration**
    - Frequency of log backup too low.
    - Retention period too high.
    - Job schedules overlap → conflicts.

🔍 Real-Time Scenarios

Scenario 1: Log Backup Job Failure

- Primary DB in Full recovery.
- Log backup job fails due to SQL Agent being stopped.
- Log file grows 50GB → Disk full → transactions fail.

Resolution:

- Start SQL Agent.
- Manually take log backup:

```
BACKUP LOG [YourDB]
TO DISK = 'D:\Backup\YourDB_Log.trn';
```

- Verify log truncation:

```
DBCC SQLPERF(LOGSPACE);
```

- Check and re-enable log shipping jobs.

Scenario 2: Secondary Restore Job Fails

- Copy job succeeded, but restore job fails due to secondary database being in use.
- Logs accumulate on primary → primary log file grows.

Resolution:

- Set secondary database to **Restoring Mode** if necessary:

```
RESTORE DATABASE [SecondaryDB] WITH NORECOVERY;
```

- Check for blocking sessions on secondary and resolve.
- Re-run restore job manually.

Scenario 3: Long-Running Transactions

- A nightly ETL loads 200 million rows.
- Log backup runs every 15 minutes, but open transaction is 4 hours.
- Log cannot truncate → fills disk.

Resolution:

- Split large transactions into smaller batches.
- Use **CHECKPOINT** to flush dirty pages.
- Verify open transactions:

```
DBCC OPENTRAN('YourDB');
```

Scenario 4: Disk Space Limitation

- Log file cannot grow due to primary disk full.
- Log shipping fails repeatedly.

Resolution:

- Free up space on the log drive.
- Shrink log file **only if necessary**:

```
DBCC SHRINKFILE('YourDB_Log', target_size_in_MB);
```

- Add new VLFs if log file is highly fragmented.

🧩 Detailed Resolution Steps (Step-by-Step)

Step 1: Verify Log Shipping Status

```
EXEC master.dbo.sp_help_log_shipping_monitor_primary;
```

- Check last successful backup, copy, and restore times.
- Identify which job failed.

Step 2: Check Log File Usage

```
DBCC SQLPERF(LOGSPACE);
```

- Observe % Log Used for primary DB.

Step 3: Check Open Transactions

```
DBCC OPENTRAN('YourDB');
```

- Resolve any active long transactions if possible.

Step 4: Manual Log Backup (if necessary)

```
BACKUP LOG [YourDB] TO DISK = 'D:\Backup\YourDB_Log.trn';
```

- Ensures log truncation.

Step 5: Restart Failed Jobs

- SQL Agent → Log Shipping Backup, Copy, Restore Jobs.
- Check Job History for errors.

Step 6: Ensure Sufficient Disk Space

- Primary log drive + secondary restore drive.
- Consider **auto-growth settings**.

Step 7: Adjust Log Shipping Schedule

- Shorter intervals for frequent backups.
- Avoid overlap of copy/restore jobs.

Step 8: Monitor and Validate

```
-- Validate log shipping status
SELECT * FROM msdb.dbo.log_shipping_monitor_primary;
SELECT * FROM msdb.dbo.log_shipping_monitor_secondary;
```

- Ensure secondary DB is being updated timely.
- Monitor for unusual log growth.

✅ **Preventive Best Practices**

Area	Recommendation
Job Monitoring	Enable alerts for failed log shipping jobs.
Disk Management	Allocate sufficient space for log growth + VLFs.
Frequency	Backup log frequently to avoid excessive growth (e.g., 5–15 min intervals).
Long Transactions	Break large transactions into batches.
TempDB / Log Optimization	Ensure log file auto-growth is configured properly.
Secondary Checks	Ensure secondary DB is always in NORECOVERY mode and accessible.

📌 **Real-Time Use Case Example**

Problem: Production DB log grows from 20GB → 200GB in 2 hours; log shipping fails; secondary not updated.

Steps Taken:

1. Checked log shipping jobs → Backup Job failed.
2. Verified open transactions → None blocking.
3. Manually took log backup → truncated log.
4. Restarted log shipping jobs → logs shipped and restored.
5. Adjusted job schedules to 10-minute intervals → prevented future log bloat.
6. Added disk space buffer → avoided auto-growth stalls.

Outcome: Log file growth normalized; secondary database is synchronized; system stable.