Log Shipping für SQL Server auf Linux

Kathegorie
SQL Server

Simon
Schlaipfer

Wir stellen uns ein Szenario vor, in dem wir über eine SQL Server Instanz auf einem Linux Centos 7 Server verfügen. Ist es nun möglich, eine **Disaster Recovery Solution** einzurichten, mit der wir Backups auf einer sekundären Instanz automatisch wiederherstellen können? Die Antwort lautet: **Ja!**

Und genau darum geht es in diesem Beitrag. Wir wollen uns anhand eines praktischen Beispiels genau anschauen, wie wir innerhalb einer Linux Centos 7 Umgebung eine solche **Disaster Recovery Solution** konfigurieren können.

Wofür das alles?

Um zu überprüfen, ob eine Datenbank Sicherung einwandfrei funktioniert oder gar beschädigt ist, bleibt uns als einzige Möglichkeit, die Sicherung auf einer sekundären Instanz wiederherzustellen. Je größer hier die zu wiederherstellende Datenbank ist, desto größer ist die entsprechende Sicherung. Dies wirkt sich nicht nur auf die Speicheranforderungen aus, sondern beeinflusst auch unmittelbar die Wiederherstellungszeit.

Da in einer aktiven Produktionsumgebung meist viele Datenbanken im Einsatz sind, werden entsprechend auch sehr viele Sicherungen erstellt. Um sicherzustellen, dass die Sicherungen funktionieren, müssen also regelmäßige Wiederherstellungen der Sicherungen auf sekundären Instanzen erfolgen, was bei großen Datenbanken sehr viel Zeit in Anspruch nehmen kann. Ein Weg, um hier eine optimale Lösung zu finden, ist das sogenannte **SQL Server Log Shipping**.

Mit dem **SQL Server Log Shipping** können Transaktionsprotokoll-Sicherungen von einer SQL Server Instanz an eine oder mehrere sekundäre Instanzen gesendet werden. Hierbei werden die Transaktionsprotokoll-Sicherungen auf jeder sekundären Datenbank einzeln wiederhergestellt. Auf diese Art und Weise können

- regelmäßige Wiederherstellungen von Sicherungen auf sekundären Instanzen erfolgen und
- eine **Disaster Recovery Solution** bereitgestellt werden.

Weiterhin benötigen Transaktionsprotokoll-Sicherungen weniger Speicherplatz als vollständige Sicherungen, wodurch der Speicherplatzbedarf der Sicherungen reduziert werden kann.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

In diesem Beispiel werden wir nun genau so ein **Log Shipping** konfigurieren. Wir verwenden hier zwei Linux Centos 7 Server, wobei einer der Server den primären Server und der zweite den sekundären Server darstellt. Auf beiden Servern sind bereits je eine Instanz von SQL Server 2019 und die SQL Server Command Line Tools installiert.

Um nun das **Log Shipping** auf Linux zu konfigurieren, werden wir:

- den primären Server konfigurieren,
- den sekundären Server konfigurieren,
- das **Log Shipping** auf dem primären Server einrichten,
- das **Log Shipping** auf dem sekundären Server einrichten.

Um sicherzustellen, dass der sekundäre Server Zugriff auf die Transaktionsprotokoll-Sicherungen des primären Servers hat, müssen diese auf einer Netzwerk-Dateifreigabe gespeichert werden. In diesem Beispiel werden wir hierfür **Samba** und das **Common Internet File System (CIFS)** verwenden. Bei **CIFS** handelt es sich um eine Implementierung des SMB-Protokolls, die zum Freigeben von Dateisystemen, Druckern oder seriellen Anschlüssen über ein Netzwerk verwendet wird. **Samba** ist die Standard-Windows-Interoperabilitätssuite für Linux und Unix, basierend auf dem SMB-Protokoll.

Den primären Server konfigurieren

Als Erstes werden wir damit beginnen, **Samba** auf dem primären Server zu konfigurieren. Um **Samba** zu installieren, einfach folgendes Kommando ausführen:

```
sudo yum -y install samba
```

```
Installiert:
  samba.x86_64 0:4.10.4-11.el7_8

Abhängigkeit installiert:
  avahi-libs.x86_64 0:0.6.31-20.el7          cups-libs.x86_64 1:1.6.3-43.el7          gnutls.x86_64 0:3.3.29-9.el7_6
  libldb.x86_64 0:1.5.4-1.el7                libtalloc.x86_64 0:2.1.16-1.el7          libtdb.x86_64 0:1.3.18-1.el7
  libtevent.x86_64 0:0.9.39-1.el7            libwbclient.x86_64 0:4.10.4-11.el7_8     nettle.x86_64 0:2.7.1-8.el7
  pyldb.x86_64 0:1.5.4-1.el7                 pytalloc.x86_64 0:2.1.16-1.el7           python-tdb.x86_64 0:1.3.18-1.el7
  samba-client-libs.x86_64 0:4.10.4-11.el7_8 samba-common.noarch 0:4.10.4-11.el7_8   samba-common-libs.x86_64 0:4.10.4-11.el7_8
  samba-common-tools.x86_64 0:4.10.4-11.el7_8 samba-libs.x86_64 0:4.10.4-11.el7_8    trousers.x86_64 0:0.3.14-2.el7

Komplett!
```

Nachdem wir **Samba** installiert haben, erstellen wir ein Verzeichnis zum Speichern der Transaktionsprotokoll-Sicherungen. Das Verzeichnis kann überall auf dem Server erstellt werden. Wir werden es unter "**/var/opt/mssql**" anlegen und TLogsLS nennen. Mit dem folgenden Kommando erstellen wir das Verzeichnis:

```
sudo mkdir /var/opt/mssql/TLogsLS
```

Damit nun die SQL Server Database Engine Transaktionsprotokoll-Sicherungen in diesem Verzeichnis gespeichert werden kann, müssen wir den **MSSQL-User** zum Besitzer des Verzeichnisses erklären und ihm die Erlaubnis zum Lesen, Schreiben und Ausführen erteilen.
Als Besitzer erklären wir den **MSSQL-User** mit dem Kommando:

```
sudo chown mssql:mssql /var/opt/mssql/TLogsLS
```

Anschließend erteilen wir ihm die Erlaubnis, innerhalb des Verzeichnisses zu Lesen, zu Schreiben und Auszuführen:

```
sudo chmod 0700 /var/opt/mssql/TLogsLS
```

Um zu überprüfen, ob der **MSSQL-User** auch tatsächlich die benötigten Erlaubnisse hat, können wir den folgenden Befehl ausführen:

```
sudo ls -l /var/opt/mssql
```

Anhand des **drwx** können wir sehen, dass der Benutzer mssql sowohl Rechte zum Schreiben, Lesen und Ausführen hat:

```
drwxr-xr-x. 2 mssql mssql 4096 15. Sep 12:47 data
drwxr-xr-x. 2 mssql mssql 4096 15. Sep 12:47 log
-rw-rw-r--. 1 mssql mssql   75 15. Sep 12:46 mssql.conf
drwxr-xr-x. 2 mssql mssql   25 15. Sep 12:47 secrets
drwx------. 2 mssql mssql    6 15. Sep 15:47 TLogsLS
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Nun müssen wir das **Samba** Configurations-File bearbeiten und das neu erstellte Verzeichnis als geteiltes Netzwerk Verzeichnis konfigurieren. Dies kann unter "**/etc/samba/smb.conf**" gefunden werden.

Wir öffnen das File mit einem Texteditor und konfigurieren es wie folgt:

```
[TLogsLS]
path = /var/opt/mssql/TLogsLS
browsable = yes
public = yes
writable = no
force user = mssql
```

Um nun zu testen, ob unsere Konfiguration korrekt und fehlerfrei war, führen wir dieses Kommando aus:

```
sudo testparm
```

Damit unsere Konfiguration angewendet wird, müssen wir anschließend den **Samba** Dienst neustarten:

```
sudo systemctl restart smb.service
sudo systemctl restart nmb.service
```

Um nun dem Verkehr von **Samba** das Durchqueren der Firewall zu erlauben, müssen wir diese entsprechend konfigurieren:

```
sudo firewall-cmd --permanent --zone=public --add-service=samba
sudo firewall-cmd --reload
```

Als letzten Schritt müssen wir ein Benutzer-Konto anlegen, das auf **Samba** zugreift. Im Falle des **SQL Server Log Shipping** ist dies der **MSSQL-User**. Dieser Steuert den SQL Server Dienst sowohl auf dem primären als auch auf dem sekundäreren Server. Der sekundäre Server verbindet sich hier mit dem primären Server und kopiert die Transaktionsprotokoll-Sicherungen.

Mit dem folgenden Kommando können wir den **MSSQL-User** der **Samba Internal Database** hinzufügen. Wir werden hier dazu aufgefordert, ein Passwort für den Benutzer festzulegen. Dieses werden wir für die Konfiguration des sekundären Servers brauchen:

```
sudo smbpasswd -a mssql
```

```
[simon@centossql01 ~]$ sudo smbpasswd -a mssql
[sudo] Passwort für simon:
New SMB password:
Retype new SMB password:
Added user mssql.
```

SELinux für den primären Server konfigurieren

Bei SELinux handelt es sich um ein Linux-Kernel-Sicherheitsmodul. Es bietet einen Mechanismus zur Unterstützung von Sicherheitsrichtlinien für die Zugriffskontrolle. Je nach dem wie der Linux Server konfiguriert ist, muss möglicherweise auch eine Konfiguration an SELinux vorgenommen werden, um **Log Shipping** zu ermöglichen.

Mit dem folgenden Kommando kann der Status von SELinux abgefragt werden:

```
sudo getenforce
```

```
[simon@centossql01 ~]$ sudo getenforce
[sudo] Passwort für simon:
Enforcing
```

Der Wert **Enforcing** bedeutet, dass SELinux eingeschaltet ist und die Richtlinienregeln aktiv sind und eingehalten werden. Nun müssen wir eine eigene Richtlinienregel hinzufügen, die **Samba** erlaubt. Damit wir eine eigene Richtlinie hinzufügen und damit wir SELinux konfigurieren können, muss zunächst das **Semanage-Dienstprogramm** installiert werden. Dies können wir mit dem folgendem Kommando:

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

```
sudo yum provides semanage
```

```
[simon@centossql01 ~]$ sudo yum provides semanage
Geladene Plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: linux.darkpenguin.net
 * extras: mirror.ratiokontakt.de
 * updates: centosmirror.netcup.net
base
extras
packages-microsoft-com-mssql-server-2019
packages-microsoft-com-prod
updates
packages-microsoft-com-prod/primary_db
packages-microsoft-com-prod/filelists_db
policycoreutils-python-2.5-34.el7.x86_64 : SELinux policy core python utilities
Quelle      : base
Übereinstimmung von:
Dateiname   : /usr/sbin/semanage
```

Wir können sehen, dass das **policycoreutils-python-2.5-34.el7.x86_64** Paket das gewünschte **Semanage-Dienstprogramm** enthält. Wir installieren es mit:

```
sudo yum install policycoreutils-python-2.5-34.el7.x86_64
```

```
Gesamt                                                                                                    9.9 MB/s | 1.6 MB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installieren     : libcgroup-0.41-21.el7.x86_64                                                                                1/7
  Installieren     : setools-libs-3.3.8-4.el7.x86_64                                                                             2/7
  Installieren     : audit-libs-python-2.8.5-4.el7.x86_64                                                                        3/7
  Installieren     : python-IPy-0.75-6.el7.noarch                                                                                4/7
  Installieren     : libsemanage-python-2.5-14.el7.x86_64                                                                        5/7
  Installieren     : checkpolicy-2.5-8.el7.x86_64                                                                                6/7
  Installieren     : policycoreutils-python-2.5-34.el7.x86_64                                                                    7/7
  Überprüfung läuft: checkpolicy-2.5-8.el7.x86_64                                                                                1/7
  Überprüfung läuft: libsemanage-python-2.5-14.el7.x86_64                                                                        2/7
  Überprüfung läuft: python-IPy-0.75-6.el7.noarch                                                                                3/7
  Überprüfung läuft: policycoreutils-python-2.5-34.el7.x86_64                                                                    4/7
  Überprüfung läuft: audit-libs-python-2.8.5-4.el7.x86_64                                                                        5/7
  Überprüfung läuft: setools-libs-3.3.8-4.el7.x86_64                                                                             6/7
  Überprüfung läuft: libcgroup-0.41-21.el7.x86_64                                                                                7/7

Installiert:
  policycoreutils-python.x86_64 0:2.5-34.el7

Abhängigkeit installiert:
  audit-libs-python.x86_64 0:2.8.5-4.el7     checkpolicy.x86_64 0:2.5-8.el7     libcgroup.x86_64 0:0.41-21.el7     libsemanage-python.x86_64 0:2.5-14.el7
  python-IPy.noarch 0:0.75-6.el7             setools-libs.x86_64 0:3.3.8-4.el7

Komplett!
```

Nachdem wir das Paket installiert haben, können wir **Semanage** verwenden um SELinux zu konfigurieren. Um SELinux nun so zu konfigurieren, dass es alle Standard Verzeichnisse als **Read-Only** freigibt, führen wir folgendes Kommando aus:

```
sudo setsebool -P samba_export_all_ro=1
```

Da auf das freigegebene Verzeichnis nur über **Samba** zugegriffen werden kann, kennzeichnen wir es mit **samba_share_t**. So erhält es Lese-und Schreibzugriff. Mit dem Kommando können wir alle Dateien im Verzeichnis "**/var/opt/mssql/TLogsLS**" in den Typ **samba_share_t** ändern:

```
sudo semanage fcontext -at samba_share_t "/var/opt/mssql/TLogsLS(/.*)?"
```

Der Parameter **-a** fügt einen neuen Datensatz hinzu, während der Parameter **-t** einen Typ definiert.

Nun stellen wir die Standard-SELinux Kontexte im Verzeichnis "**/var/opt/mssql/TLogsLS**" wieder her:

```
sudo restorecon /var/opt/mssql/TLogsLS
```

Nachdem wir nun die Konfiguration von SELinux abgeschlossen haben, können wir damit beginnen, den Sekundären Server zu konfigurieren.

Den sekundären Server konfigurieren

Der sekundäre Server fungiert beim **Log Shipping** als Client für den primären Server und stellt eine Verbindung zum freigegebenen Verzeichnis her und kopiert die Transaktionsprotokoll-Sicherungen. Hierfür ist ein **CIFS-Client** erforderlich. Dieser sorgt dafür, dass auf einem Linux-System SMB-Freigaben bereitgestellt werden können. Um einen **CIFS**-Client zu installieren, führen wir auf unserem sekundären Server folgenden Befehl aus:

```
sudo yum install -y cifs-utils
```

```
Installiert:
  cifs-utils.x86_64 0:6.2-10.el7

Abhängigkeit installiert:
  avahi-libs.x86_64 0:0.6.31-20.el7          cups-libs.x86_64 1:1.6.3-43.el7          gnutls.x86_64 0:3.3.29-9.el7_6
  keyutils.x86_64 0:1.5.8-3.el7              libldb.x86_64 0:1.5.4-1.el7              libtalloc.x86_64 0:2.1.16-1.el7
  libtdb.x86_64 0:1.3.18-1.el7               libtevent.x86_64 0:0.9.39-1.el7          libwbclient.x86_64 0:4.10.4-11.el7_8
  nettle.x86_64 0:2.7.1-8.el7                pyldb.x86_64 0:1.5.4-1.el7               pytalloc.x86_64 0:2.1.16-1.el7
  python-tdb.x86_64 0:1.3.18-1.el7           samba-client-libs.x86_64 0:4.10.4-11.el7_8  samba-common.noarch 0:4.10.4-11.el7_8
  samba-common-libs.x86_64 0:4.10.4-11.el7_8 samba-libs.x86_64 0:4.10.4-11.el7_8      trousers.x86_64 0:0.3.14-2.el7

Komplett!
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Nach der Installation des **CIFS**-Clients müssen wir ein Verzeichnis erstellen, in dem die von dem primären Server kopierten Transaktionsprotokoll-Sicherungen gespeichert werden. Auch hier werden wir ein Verzeichnis namens **TLogsLS** unter "**/var/opt/mssql**" anlegen:

```
sudo mkdir /var/opt/mssql/TLogsLS
```

Damit der **MSSQL-User** kopierte Transaktionsprotokoll-Sicherungen vom primären Server in diesem Verzeichnis speichern kann, müssen wir den User zum Besitzer des Verzeichnisses machen und ihm die Berechtigung zum Lesen und Ausführen erteilen. Hierfür führen wir folgende Befehle aus:

```
sudo chown mssql:mssql /var/opt/mssql/TLogsLS
sudo chmod 0550 /var/opt/mssql/TLogsLS
```

Jetzt erstellen wir eine versteckte Datei, um unsere **Samba** Zugangsdaten zu speichern. Der Punkt vor dem Dateinamen gibt an, dass es sich um eine versteckte Datei handelt:

```
sudo touch /var/opt/mssql/.smbcreds_mssql
```

Anschließend editieren wir die Datei und tragen hier die Zugangsdaten ein, die wir auf unserem primären Server mit dem **smbpasswd** Kommando vergeben haben:

```
username=mssql
password=passwort123
```

Auch hier ist es wichtig, dass der **MSSQL-User** Besitzer der Datei ist. Hierfür führen wir folgendes Kommando aus:

```
sudo chown mssql:mssql /var/opt/mssql/.smbcreds_mssql
```

Um die Datei von unautorisierten Zugriffen zu schützen, konfigurieren wir sie so, dass nur der **MSSQL-User** (Besitzer der Datei) Zugriff hat:

```
sudo chmod 0400 /var/opt/mssql/.smbcreds_mssql
```

Nachdem wir das Verzeichnis erstellt und dem **MSSQL-User** die nötigen Berechtigungen zugewiesen haben, können wir die **Samba**-Freigabe auf dem primär Server bereitstellen. Hierbei wird das Verzeichnis "**/var/opt/mssql/TLogsLS**" auf dem sekundären Server dem gemeinsam genutzten **Samba**-Verzeichnis auf dem Primären Server zugeordnet. Hierfür führen wir folgendes Kommando auf dem sekundären Server aus:

```
sudo mount //centossql01/TLogsLS /var/opt/mssql/TLogsLS -o username=mssql
```

Nachdem mit dem Argument **-o** ein Benutzername übergeben wurde, wird dessen Passwort abgefragt. Dies kann im Falle eines Server-Neustarts Komplikationen ergeben, da das freigegebene **Samba**-Verzeichnis nicht automatisch bereitgestellt wird. Damit das Verzeichnis einen Neustart überlebt, müssen wir das Bereitstellen der Anmeldeinformationen automatisieren. Dazu müssen wir die Datei "/Etc/fstab" konfigurieren. Diese wird zum Definieren der Linux-Dateisystemtabelle verwendet. Sie enthält alle verfügbaren Festplatten, Festplattenpartitionen und deren Optionen – einschließlich der bereitgestellten freigegebenen **Samba**-Verzeichnisse:

```
//centossql01/TLogsLS /var/opt/mssql/TLogsLS cifs credentials=/var/opt/mssql/.
smbcreds_mssql,ro,uid=mssql,gid=mssql 0 0
```

Das erste Feld in diesem Eintrag bezieht sich auf das Blockgerät oder in diesem Fall auf ein freigegebenes **Samba**-Verzeichnis. Das zweite Feld bezieht sich auf den Mountpunkt. Das dritte Feld – **cifs** – bezieht sich auf den Dateisystemtyp. Das vierte Feld enthält eine Liste der Optionen, die beim Mounten des Dateisystems verwendet werden.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Das Log Shipping auf dem primären Server einrichten

Das Einrichten des **Log Shipping** auf dem primären Server entspricht dem Einrichten von SQL Server unter Windows mithilfe der gespeicherten Prozedur **sp_add_log_shipping_primary_database**. Die Transaktionsprotokoll-Sicherungen werden im lokalen Verzeichnis "**/var/opt/mssql/TLogsLS**" des primären Servers gespeichert. Mit folgendem T-SQL Skript kann das **Log Shipping** konfiguriert werden:

```
DECLARE @LS_BackupJobId  AS uniqueidentifier
DECLARE @LS_PrimaryId    AS uniqueidentifier
DECLARE @SP_Add_RetCode  AS int
EXEC @SP_Add_RetCode = master.dbo.sp_add_log_shipping_primary_database
         @database = N'Northwind'
        ,@backup_directory= N'/var/opt/mssql/TLogsLS/'
        ,@backup_share = N'/var/opt/mssql/TLogsLS/'
        ,@backup_job_name = N'LSBackup_Northwind'
        ,@backup_retention_period = 4320
        ,@backup_compression = 2
        ,@backup_threshold = 60
        ,@threshold_alert_enabled = 1
        ,@history_retention_period = 5760
        ,@backup_job_id = @LS_BackupJobId OUTPUT
        ,@primary_id = @LS_PrimaryId OUTPUT
        ,@overwrite = 1

IF (@@ERROR = 0 AND @SP_Add_RetCode = 0)
BEGIN

DECLARE @LS_BackUpScheduleUID  AS uniqueidentifier
DECLARE @LS_BackUpScheduleID   AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'LSBackupSchedule'
        ,@enabled = 1
        ,@freq_type = 4
        ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170418
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_BackUpScheduleUID OUTPUT
        ,@schedule_id = @LS_BackUpScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_BackupJobId
        ,@schedule_id = @LS_BackUpScheduleID

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_BackupJobId
        ,@enabled = 1
END

EXEC master.dbo.sp_add_log_shipping_alert_job

EXEC master.dbo.sp_add_log_shipping_primary_secondary
        @primary_database = N'Northwind'
        ,@secondary_server = N'centossql03'
        ,@secondary_database = N'Northwind'
        ,@overwrite = 1
```

Hierbei stellen wir sicher, dass die Datenbank initialisiert ist, bevor wir den SQL Server Agent Job ausführen. Wir sichern die Datenbank auf dem primären Server und stellen sie mit der Option **WITH NORECOVERY** auf dem sekundären Server wieder her.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Das Log Shipping auf dem sekundären Server einrichten

Auch das Einrichten des **Log Shipping** auf dem sekundären Server entspricht dem Einrichten von SQL Server unter Windows, allerdings unter Verwendung der gespeicherten Prozedur **sp_add_log_shipping_secondary_primary**. Hier muss zum Einrichten des **Log Shipping** folgendes T-SQL Skript ausgeführt werden:

```
DECLARE @LS_Secondary__CopyJobId     AS uniqueidentifier
DECLARE @LS_Secondary__RestoreJobId  AS uniqueidentifier
DECLARE @LS_Secondary__SecondaryId   AS uniqueidentifier
DECLARE @LS_Add_RetCode              AS int

EXEC @LS_Add_RetCode = master.dbo.sp_add_log_shipping_secondary_primary
        @primary_server = N'centossql02'
        ,@primary_database = N'Northwind'
        ,@backup_source_directory= N'/var/opt/mssql/TLogsLS/'
        ,@backup_destination_directory = N'/var/opt/mssql/TLogsLS/'
        ,@copy_job_name = N'LSCopy_Northwind'
        ,@restore_job_name = N'LSRestore_Northwind'
        ,@file_retention_period = 4320
        ,@overwrite = 1
        ,@copy_job_id = @LS_Secondary__CopyJobId OUTPUT
        ,@restore_job_id = @LS_Secondary__RestoreJobId OUTPUT
        ,@secondary_id = @LS_Secondary__SecondaryId OUTPUT

IF (@@ERROR = 0 AND @LS_Add_RetCode = 0)
BEGIN

DECLARE @LS_SecondaryCopyJobScheduleUID AS uniqueidentifier
DECLARE @LS_SecondaryCopyJobScheduleID  AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'DefaultCopyJobSchedule'
        ,@enabled = 1
        ,@freq_type = 4
        ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170418
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_SecondaryCopyJobScheduleUID OUTPUT
        ,@schedule_id = @LS_SecondaryCopyJobScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_Secondary__CopyJobId
        ,@schedule_id = @LS_SecondaryCopyJobScheduleID

DECLARE @LS_SecondaryRestoreJobScheduleUID  AS uniqueidentifier
DECLARE @LS_SecondaryRestoreJobScheduleID   AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'DefaultRestoreJobSchedule'
        ,@enabled = 1
        ,@freq_type = 4
        ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170418
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_SecondaryRestoreJobScheduleUID OUTPUT
        ,@schedule_id = @LS_SecondaryRestoreJobScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_Secondary__RestoreJobId
        ,@schedule_id = @LS_SecondaryRestoreJobScheduleID

END
DECLARE @LS_Add_RetCode2 AS int
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

```
IF (@@ERROR = 0 AND @LS_Add_RetCode = 0)
BEGIN

EXEC @LS_Add_RetCode2 = master.dbo.sp_add_log_shipping_secondary_database
        @secondary_database = N'Northwind'
        ,@primary_server = N'centossql02'
        ,@primary_database = N'Northwind'
        ,@restore_delay = 0
        ,@restore_mode = 0
        ,@disconnect_users = 0
        ,@restore_threshold = 45
        ,@threshold_alert_enabled = 1
        ,@history_retention_period = 5760
        ,@overwrite = 1

END

IF (@@error = 0 AND @LS_Add_RetCode = 0)
BEGIN

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_Secondary__CopyJobId
        ,@enabled = 1

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_Secondary__RestoreJobId
        ,@enabled = 1

END
```

Nun ist die Konfiguration des **Log Shipping** beendet und unsere Umgebung ist bereit zum Einsatz.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612