**SQL Server Database Migration**

|  |  |
| --- | --- |
| DOCUMENT NAME | **SQL Server Database Migration** |
| DOCUMENT VERSION: | 1.0 |
| DATE: | 27-Nov-2020 |
| SUMMARY: | This document provides understanding of SQL Database migration from one server to another server |

***Amendment History***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Comment | By | Reviewer |
| 0.1 | 27-Nov-2020 | Draft Version | Afroz Ahmad |  |
| 0.2 |  |  |  |  |

***Approval***

|  |  |  |
| --- | --- | --- |
| Approver | Name | Date |
|  |  |  |
|  |  |  |

**Name of the client:** Pine River

**Activity:** SQL Server Database Migration

**What is Database Migration?**

Database Migration is the process of moving databases from one operating environment to another operating environment or from one SQL version to another SQL version that is, in most cases, is thought to be a better one. For example, moving from SQL Server 2012 to SQL Server 2019 would usually be considered a migration because it involves making sure that new features are exploited, old settings do not require changing, and taking steps to ensure that current applications continue to work in the new environment

**Benefits of Migration:**

* Increasing performance
* Minimized downtime
* Full scalability
* Low total cost of ownership

**SQL Server Migration Plan**

The migration plan would be executed in 3 phases:

1. Pre-migration checks
2. Actual DB migration and setup (Isolation of the DB, migration of jobs & logins)
3. Post-migration consistency and connectivity checks.

**Pre-Migration phase**

**Pre-Migration Checklist (Source Server)**

The following constraints / features shall be checked / noted down:

1. Database sizes.
2. Data and Log file location.
3. Server and Database properties (Collation, Auto Stats, DB Owner, Recovery Model, Compatibility level, Trustworthy option, CDC , CLR\_ENABLED etc).
4. Collect the information of dependent applications, make sure application services will be stopped during the database migration
5. Database logins, users and their permissions.
6. Dependent objects (SQL Agent Jobs and Linked Servers)
7. Maintenance plans.

**Pre-Migration Checklist (Destination Server)**

Analogous to the above checklist, we shall check / create the following:

1. Adequate Disk space on the server.
2. Correct destination folders are created.
3. SQL Server is correctly installed and configured as per requirement.
4. Connectivity to the application servers and linked servers.

**Migration Phase**

**Steps to be performed on the Source Server:**

1. Isolate Source server from all application and linked servers.
2. The Database(s) from the source server are backed up with password to ensure secure movement of the data.
3. Script out all Jobs, Linked Servers, Logins and Users.
4. The Databases may now be put into Read-Only mode if required.

**Steps to be performed on the Destination Server**

1. Transfer the backup to the desired location.
2. Restore the database ensuring that the data and log files are placed in the correct location.
3. Recreate the Logins and User. Resolve Orphan User issues.
4. Re-establish Linked Servers and check any FTP Locations that are to be accessed.
5. Recreate the Jobs and Maintenance plans
6. Perform consistency checks and update index stats.
7. Compare SP\_CONFIGURE options between current and new server (CPUs, Memory, MAXDOP, Cost threshold, etc.)

**Post Migration Phase**

1. Point the application to the new DB server IP (Connection string etc to altered by the application support team)
2. Restart Network connections between all stake holding servers (Network Team)
3. Check the SQL Server Error Log and Windows Error logs for any failures.
4. Confirm application functionality with end users.

**Script to Check the Disk and Database Size**

-- Procedure to check disc space

exec master..xp\_fixeddrives

-- To Check database size

exec sp\_helpdb [dbName]

or

use [dbName]

select str(sum(convert(dec(17,2),size)) / 128,10,2) + 'MB'

from dbo.sysfiles

GO

**Script to Check Database Properties**

select

sysDB.database\_id,

sysDB.Name as 'Database Name',

syslogin.Name as 'DB Owner',

sysDB.state\_desc,

sysDB.recovery\_model\_desc,

sysDB.collation\_name,

sysDB.user\_access\_desc,

sysDB.compatibility\_level,

sysDB.is\_read\_only,

sysDB.is\_auto\_close\_on,

sysDB.is\_auto\_shrink\_on,

sysDB.is\_auto\_create\_stats\_on,

sysDB.is\_auto\_update\_stats\_on,

sysDB.is\_fulltext\_enabled,

sysDB.is\_trustworthy\_on,

sysDB.is\_cdc\_enabled

from sys.databases sysDB

INNER JOIN sys.syslogins syslogin ON sysDB.owner\_sid = syslogin.sid

**Script to CLR Enabled/Disabled**

SELECT name, CASE WHEN value = 1 THEN 'YES' ELSE 'NO' END AS 'Enabled' FROM sys.configurations WHERE name = 'clr enabled'

**Script to List Linked Servers**

SELECT \* FROM sys.sysservers

**Script to List Database Dependent Jobs**

Use msdb

go

select

distinct

name,

database\_name

from sysjobs sj

INNER JOIN sysjobsteps sjt on sj.job\_id = sjt.job\_id

**Migrate Logins:**

Run stored procedure “sp\_help\_revlogin” and execute the output on Target server.

**Script to Check and Fix Orphan Users**

-- Script to check the orphan user

EXEC sp\_change\_users\_login 'Report'

--Use below code to fix the Orphan User issue

DECLARE @username varchar(25)

DECLARE fixusers CURSOR

FOR

SELECT UserName = name FROM sysusers

WHERE issqluser = 1 and (sid is not null and sid <> 0x0)

and suser\_sname(sid) is null

ORDER BY name

OPEN fixusers

FETCH NEXT FROM fixusers

INTO @username

WHILE @@FETCH\_STATUS = 0

BEGIN

EXEC sp\_change\_users\_login 'update\_one', @username, @username

FETCH NEXT FROM fixusers

INTO @username

END

CLOSE fixusers

DEALLOCATE fixusers

**Script to Change DB Owner**

USE databaseName

EXEC sp\_changedbowner 'sa'

**Script to Turn on Trustworthy Option**

ALTER DATABASE database\_name SET TRUSTWORTHY ON

**DBCC Update usage and DBCC Check DB Command**

DBCC UPDATEUSAGE('database\_name') WITH COUNT\_ROWS

DBCC CHECKDB

OR

DBCC CHECKDB('database\_name') WITH ALL\_ERRORMSGS

**Index Rebuild/Reorg**

-- Script for Index Rebuild

USE [DBName];

GO

ALTER INDEX ALL ON [ObjectName] REBUILD

GO

-- Script for Index Reorganize

USE AdventureWorks;

GO

ALTER INDEX ALL ON [ObjectName] REORGANIZE

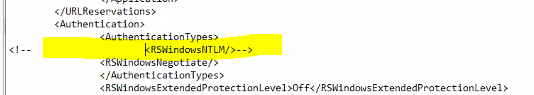
GO

**Update index statistics**

sp\_updatestats

**Migrate SSRS Reports:**

1. Backup the encryption keys for the report server database you want to move. You can use the Reporting Services Configuration tool backup the keys.
2. Backup ReportServer & ReportServerTempDB databases.
3. Copy these files over to the new Server.
4. Restore both Report databases over to the new Server.
5. After the databases are restored, verify that the **RSExecRole** is a database role in the report server database and temporary database. **RSExecRole** must have select, insert, update, delete, and reference permissions on the report server database tables, and execute permissions on the stored procedures.
6. Make sure we use RSWindowsNegotiate for Authentication type; instead of RSWindowsNTLM in SSRS config file unless option to choose during install.



1. Start the Reporting Services Configuration tool and open a connection to the report server.
2. On the Database page, select the new SQL Server instance, and then click **Connect**.
3. Select the report server database that you just moved, and then click **Apply**.
4. On the Encryption Keys page, click Restore. Specify the file that contains the backup copy of the keys and the password to unlock the file.
5. After migration of SSRS-old to SSRS-new & changing server names, make sure point SSRS database back to SSRS-old.
6. Restart the Report Server service.

**Thank You**