<https://stevethompsonmvp.wordpress.com/2016/05/02/sql-server-database-migration-checklist/>

This guide is meant to provide general recommendations for migrating SQL Server databases from one server to another.

Always test database migrations (and subsequent application connectivity) in a lab environment prior to the actual migration.

**Note: Server name ‘SOURCE’ is the original SQL Server, ‘TARGET’ is the new SQL Server.**

1.1 Database migration checklist

The migration should be conducted in three phases:

1) Pre-migration inventory / checks

2) Database migration

3) Post-migration configuration / checks

1.1.1 Pre-migration phase

For the pre-migration phase, review and document the following tasks on the **SOURCE** server.

1) Database sizes

2) Data and Log file location

3) Server and Database properties (Collation, Auto Stats, DB Owner, Recovery Model, Compatibility level, 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) Document maintenance plans

**On the TARGET server, conduct the following tasks:**

1) Verify adequate disk space exists

2) Appropriate folders have been created

3) SQL Server is correctly installed and configured with latest Service Pack and Cumulative Update

4) Set SQL Server properties; memory, tempdb size, autogrowth and drive location.

5) Appropriate connectivity exists with application servers, web servers, etc.

6) Create databases consistent with planned database migrations

### 1.1.2 Database migration phase

1) From **SOURCE** server, get recent list of DB and file location list

2) On **SOURCE** SQL Server, [Script migrate user LOGIN information](https://stevethompsonmvp.wordpress.com/2016/05/02/sql-server-database-migration-checklist/#_Transfer_logins) between servers, save output for subsequent step.

3) Use native SQL Server to back up databases from **SOURCE**, use backup compression to reduce size of backup (optionally, use detach database, copy db files, then reattach database)

4) On **SOURCE** server, place databases in Read Only mode

5) Restore databases to **TARGET**

6) On **TARGET** SQL, apply script to migrate users/group security (Created in Step 3)

7) Update any custom SSRS DSNs to point to **TARGET** server.

8) On **SOURCE** server, script any required SQL Server Agent Jobs (select Job, right click, Script Job as > CREATE to > File ) to create

9) On **TARGET, using the file create,** then transfer/execute job creation on TARGET. Create schedules as needed.

### 1.1.3 Post-migration phase

1) Point the application to the new DB server address, or Availability Group LISTENER for AlwaysOn Availability Groups (Connection strings, etc. to altered by the application support team)

2) Restart / review Network connections between all stake holding servers (Network Team)

3) Check the SQL Server Error Log and Windows Error logs for any failures

a. Confirm application functionality with end users

## 1.2 Transfer logins

If you ever have a need to migrate SQL Server databases between servers; particularly a backup/restore scenario, you need to be aware that the database security login ids are not automatically migrated. Here are some notes on this task.

Once databases are backed up from one instance and restored to another instance, database security needs to be transferred as well.

This task can be a challenge, fortunately Microsoft has created a SQL script that can be used for this purpose. This can be found here:

[How to transfer logins and passwords between instances of SQL Server](http://support.microsoft.com/kb/918992)

The process is well documented. The steps are as follows:

1. From the transfer logins and passwords web page, copy/paste the SQL script to SQL Server Management Studio (SSMS) running on the SOURCE server.

2. Execute the script – it will create a new stored procedures on the Master database.

3. From SSMS, run the following query:

4. EXEC sp\_help\_revlogin

5. The output script that the **sp\_help\_revlogin** stored procedure generates is the login script. This login script creates the logins that have the original Security Identifier (SID) and the original password.

6. On server B, start SQL Server Management Studio, and then connect to the instance of SQL Server to which you moved the database.

7. **Important** Before you go to step 5, review the information in the "Remarks" section. (Note: see web page mentioned earlier)

8. Open a new Query Editor window on TARGET server, and then run the output script that is generated in step 3.

Reference: <http://support.microsoft.com/kb/918992>

**--TSQL Script to fetch all Databases File Paths/Locations:**

SELECT d.name DatabaseName, f.name LogicalName,

f.physical\_name AS PhysicalName,

f.type\_desc TypeofFile

FROM sys.master\_files f

INNER JOIN sys.databases d ON d.database\_id = f.database\_id

order by d.name asc

GO

**--TSQL Script to export SP\_CONFIGURE Settings from Source to Destination SQL Server.**

SELECT 'EXEC sp\_configure ''' + name + ''', ' + CAST(value AS VARCHAR(100))

FROM sys.configurations

ORDER BY name

**--TSQL Script to extract Backup & Restore Script.**

--<https://gist.github.com/airtank20/a826c6f37439482edd5070e8aaeb1ee1>

--<https://sqlrus.com/2018/02/backup-restore-script-with-a-move/>

Use master

go

DECLARE @date CHAR(8)

SET @date = (SELECT CONVERT(char(8), GETDATE(), 112))

DECLARE @path VARCHAR(125)

SET @path = 'E:\SQLBackups\_Default\' **-- Provide the backup path here**

;WITH MoveCmdCTE ( DatabaseName, MoveCmd )

AS ( SELECT DISTINCT

DB\_NAME(database\_id) ,

STUFF((SELECT ' ' + CHAR(13)+', MOVE ''' + name + ''''

+ CASE Type

WHEN 0 THEN ' TO ''E:\SQLData\_Default\' **-- Provide SQL Data File Path here**

ELSE ' TO ''F:\SQLLogs\_Default\' --**-- Provide SQL Log File Path here**

END

+ REVERSE(LEFT(REVERSE(physical\_name),

CHARINDEX('\',

REVERSE(physical\_name),

1) - 1)) + ''''

FROM sys.master\_files sm1

WHERE sm1.database\_id = sm2.database\_ID

FOR XML PATH('') ,

TYPE).value('.', 'varchar(max)'), 1, 1, '') AS MoveCmd

FROM sys.master\_files sm2

)

SELECT

'BACKUP DATABASE ' + name + ' TO DISK = ''' + @path + '' + name + '\_COPY\_ONLY\_' + @date + '.bak'' WITH COMPRESSION, COPY\_ONLY, STATS=5',

'RESTORE DATABASE '+ name + ' FROM DISK = ''' + @path + '' + name + '\_COPY\_ONLY\_' + @date + '.bak'' WITH RECOVERY, REPLACE, STATS=5 ' + movecmdCTE.MoveCmd

FROM sys.databases d

INNER JOIN MoveCMDCTE ON d.name = movecmdcte.databasename

WHERE d.name not in ('master', 'model', 'msdb', 'tempdb') --LIKE '%DatabaseName%'

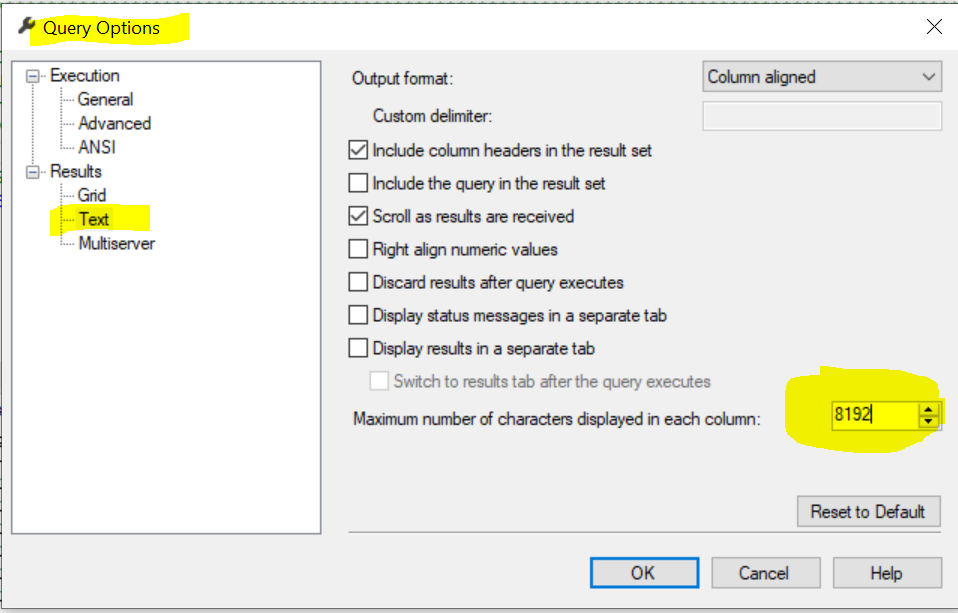
GO

**--TSQL Script to extract Logins, Server Connect & Server Level Permissions.**

--<https://www.datavail.com/blog/scripting-out-the-logins-server-role-assignments-and-server-permissions>

-- Scripting Out the Logins, Server Role Assignments, and Server Permissions  
-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- CRITICAL NOTE: You’ll need to change your results to display more characters in the query result.  
-- Under Tools –> Options –> Query Results –> SQL Server –> Results to Text to increase the maximum number of characters   
-- returned to 8192 the maximum or to a number high enough to prevent the results being truncated.  
-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



SET NOCOUNT ON  
**-- Scripting Out the Logins To Be Created**SELECT 'IF (SUSER\_ID('+QUOTENAME(SP.name,'''')+') IS NULL) BEGIN CREATE LOGIN ' +QUOTENAME(SP.name)+  
                         CASE   
                                        WHEN SP.type\_desc = 'SQL\_LOGIN' THEN ' WITH PASSWORD = ' +CONVERT(NVARCHAR(MAX),SL.password\_hash,1)+ ' HASHED, CHECK\_EXPIRATION = '   
                                                + CASE WHEN SL.is\_expiration\_checked = 1 THEN 'ON' ELSE 'OFF' END +', CHECK\_POLICY = ' +CASE WHEN SL.is\_policy\_checked = 1 THEN 'ON,' ELSE 'OFF,' END  
                                        ELSE ' FROM WINDOWS WITH'  
                                END   
         +' DEFAULT\_DATABASE=[' +SP.default\_database\_name+ '], DEFAULT\_LANGUAGE=[' +SP.default\_language\_name+ '] END;' COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS AS [-- Logins To Be Created --]  
FROM sys.server\_principals AS SP LEFT JOIN sys.sql\_logins AS SL  
                ON SP.principal\_id = SL.principal\_id  
WHERE SP.type IN ('S','G','U')  
                AND SP.name NOT LIKE '##%##'  
                AND SP.name NOT LIKE 'NT AUTHORITY%'  
                AND SP.name NOT LIKE 'NT SERVICE%'  
                AND SP.name <> ('sa');

**-- Scripting Out the Role Membership to Be Added**  
SELECT   
'EXEC master..sp\_addsrvrolemember @loginame = N''' + SL.name + ''', @rolename = N''' + SR.name + '''  
' AS [-- Server Roles the Logins Need to be Added --]  
FROM master.sys.server\_role\_members SRM  
        JOIN master.sys.server\_principals SR ON SR.principal\_id = SRM.role\_principal\_id  
        JOIN master.sys.server\_principals SL ON SL.principal\_id = SRM.member\_principal\_id  
WHERE SL.type IN ('S','G','U')  
                AND SL.name NOT LIKE '##%##'  
                AND SL.name NOT LIKE 'NT AUTHORITY%'  
                AND SL.name NOT LIKE 'NT SERVICE%'  
                AND SL.name <> ('sa');

**-- Scripting out the Permissions to Be Granted**  
SELECT   
        CASE WHEN SrvPerm.state\_desc <> 'GRANT\_WITH\_GRANT\_OPTION'   
                THEN SrvPerm.state\_desc   
                ELSE 'GRANT'   
        END  
 + ' ' + SrvPerm.permission\_name + ' TO [' + SP.name + ']' +   
        CASE WHEN SrvPerm.state\_desc <> 'GRANT\_WITH\_GRANT\_OPTION'   
                THEN ''   
                ELSE ' WITH GRANT OPTION'   
        END collate database\_default AS [-- Server Level Permissions to Be Granted --]   
FROM sys.server\_permissions AS SrvPerm   
        JOIN sys.server\_principals AS SP ON SrvPerm.grantee\_principal\_id = SP.principal\_id   
WHERE SP.type IN ( 'S', 'U', 'G' )   
                AND SP.name NOT LIKE '##%##'  
                AND SP.name NOT LIKE 'NT AUTHORITY%'  
                AND SP.name NOT LIKE 'NT SERVICE%'  
                AND SP.name <> ('sa');

SET NOCOUNT OFF