**A Faster Way to Migrate SQL Server Instances!**

**FROM:** [**http://www.sqlsolutionsgroup.com/migrate-sql-server/**](http://www.sqlsolutionsgroup.com/migrate-sql-server/)

I have had to migrate SQL Server instances many times over the years. Sometimes as part of an upgrade from one version to another, and other times just to move to new/bigger better hardware. In my current situation, the client wants to virtualize all of their SQL Server instances. Most of their database servers are already virtualized, but there are still four or five stragglers and they have set a deadline to get the rest moved. The first instance to be moved (because it is older hardware) is a SQL Server 2008 instance. We want to upgrade it to SQL Server 2008 R2 since all but one other instance are already on R2 or 2012. So how best to proceed?

I briefly considered doing an in-place upgrade. I have never done one before and I thought that just maybe an upgrade from 2008 to 2008 R2 would be simple enough to give it a go. I did some reading and talked with my MCM colleagues at SQL Solutions Group (SSG) and the recommendation was unanimous: Don’t do an in-place upgrade. If for any reason the upgrade fails, you may be left with an instance that won’t come on-line and the rollback is arduous. If the upgrade fails, your best recourse is to restore your system from a full system state backup. Often an upgrade will make changes to the system databases so a restore is in order there as well. As with all the options discussed herein, planning for all contingencies is paramount. Upgrading in place is tempting, but just don’t do it.

So, that left me with choosing from one of the ‘industry standard methodologies’ to migrate SQL Server instances. Which methodology to use is often a matter of preference and/or your specific circumstances.

**Methodology #1**: Stop access to the source database server, then backup all the databases and copy the full backups to the target and restore them there. This is an easy way to go if you have the downtime. If you have a large amount of data, however, the time becomes prohibitive. You must back up the databases, copy the backups over to the new server, and then restore the databases thereon. Any one or all of these steps can takes hours. I have performed this method successfully several times, but my total data space for all the databases was under 100 GB.

**Methodology #2**: Stop access to the source database server, detach the databases from the source and attach them to the target. Similar to methodology #1, this is easy if you have the downtime. You detach the databases on the source, copy the .ldfs and .mdfs to the target and reattach. Detaching and attaching the databases is much quicker than backup and restore, but your copy of the files will take longer since you lose the compression you get with the backups.

**Methodology #3**: Take one last full backup of all the databases on the source instance and recover them to the target using NORECOVERY. It is important that these are **the last full backups** since the intent is to complete the recovery using the DIFF backups. If another full backup must be created, make sure to use the ‘copy only’ option so as not to break the backup chain. **Do not** stop access to the source database during this portion of the migration. Then when it is time to perform the migration, stop access to the databases and take DIFF backups on the source server and restore the DIFFs on the target. This minimizes the downtime greatly as compared to methodologies #1 and #2 since the DIFF backups will be a fraction of the size of the full backups.

**Methodology #4:** Setup log shipping from the source to the target, and then make the target the master and stop the log shipping. This offers the least amount of downtime but is also the most amount of setup time. To be honest I have never migrated a server this way, but I know people have done it.

So those were my options that I had to choose from when I started preparing to upgrade the 2008 server to 2008 R2. I immediately eliminated methods 1 and 2 because the amount of data was much larger than the downtime window would permit. Since this is an upgrade, Method 4 would not work as the source and target instances need to be of the same version of SQL Server. My choice was pretty obvious, the only option left was method 3… or so I thought.

I was working with the system admins to build the virtual machine that was to become the ‘target’ database server when **Methodology #5** popped into my head:

**Methodology #5**: Shutdown SQL Server on the source instance, move the LUNs from the old server to the new server and attach the databases there. Fast and easy with little downtime if you have done all your setup properly prior to this step (of course). This of course only works if you are using a SAN and the new database server is on the same SAN.

Maybe this is not a new idea but it was new to me. In all the ‘How To Migrate a SQL Server Instance…’ articles I have read, not one of them mentioned this methodology as an option. I am sure I am not the first to come up with this plan, but it was nothing short of a revelation for me.

**Author’s note**: I have since talked to other seasoned professionals, and methodology #5 is not a *new* idea. No surprise there, but it was new to me. I figured that makes it worth blogging about, since maybe it is new to you! 

So, what is involved in preparing the target server for the migration?

**Migration Considerations**  
Migrate the system databases? If you are not doing an upgrade to a different version of SQL Server, you may choose to copy the system databases from the source server to the target. It is a matter of personal preference whether to copy the system databases or to create them anew. I have heard arguments for and against both options. Regardless, this is not an option in my scenario so I will not delve into that here, a reasonably good step by step can be found [here](http://yrushka.com/index.php/database-recovery/sql-server-migration-from-one-server-to-another-detailed-checklist/).

So, since we are not migrating the system databases, we must make sure that we copy all the pertinent objects from the source system databases to the new.

**Create the new server** with the same disk configuration as the old one. This is not required and may not be viable, but it is much easier if you can stick with the old configuration. You will have to alter your attach script to reflect the new configuration.

**Move the new system databases** where you want them to reside in the future if you did not create them there during installation. They do not have to reside in the same location as the old ones. If they are on the same drives as your databases, make sure to have a copy of the .mdf and .ldf for each so you can move them back into place once you attach the new LUNs.

**Recreate logins**. There are numerous tools for this process, but the oldest way to do it is to use sp\_help\_revlogin. This stored procedure provided by Microsoft for this very purpose has been around since SQL Server 7 (possibly 6.5?) and is still used today. It copies all the logins and passwords and recreates them with the same SID they had on the old machine. [Here is the Microsoft support page](https://support.microsoft.com/kb/918992).

**Copy the following** by scripting them out on the old instance:

* **User defined Stored Procedures and Functions** in master. You may possibly have some in model and msdb, so make sure to check and script those as well if needed.
* **Permissions on system databases.**This is often overlooked, but don’t forget that you may have specific permissions granted on some or all of the stored procedures and functions.
* **Linked Servers.** Script these out on the old server and recreate them on the new. Remember that scripting of linked servers will not script the passwords. You will need to obtain these and put them in the script manually.
* The easiest way (in my opinion) to do this is to within SSMS. Simply highlight the jobs you want to move over (ctrl-A if you want them all) and right click on the top one and select ‘Script Job as…’
* **DB Mail.** Make sure to write down your DB Mail settings on the old server so you can activate it on the new server. (Again, the Microsoft technet page is above…and [here](https://support.microsoft.com/kb/918992) for convenience).

Or if you like some scripts for DB Mail, [go here](http://www.sqlservercentral.com/Forums/Topic982618-391-1.aspx).

**Create a script for attaching all the databases** on the new server. Here is a script I wrote for creating an attach script. It assumes that the databases will be in the same location as they were on the old server. Run this on the old server (in text mode) and it will create the attach script:

|  |  |
| --- | --- |
| [Source code](http://www.sqlsolutionsgroup.com/migrate-sql-server/" \l "codesyntax_1" \o "Click to show/hide code block) | [http://i0.wp.com/www.sqlsolutionsgroup.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.sqlsolutionsgroup.com/migrate-sql-server/#codesyntax_1) [http://i0.wp.com/www.sqlsolutionsgroup.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.sqlsolutionsgroup.com/migrate-sql-server/#codesyntax_1) [http://i2.wp.com/www.sqlsolutionsgroup.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.sqlsolutionsgroup.com/wp-content/plugins/wp-synhighlight/About.html) |

USE master;

GO

SET NOCOUNT ON

IF OBJECT\_ID( 'tempdb.dbo.#tmp' ) IS NOT NULL DROP TABLE #tmp;

DECLARE

@SQLcmd VARCHAR(MAX)

,@DatabaseName VARCHAR(255)

,@PrevDatabaseName VARCHAR(255);

SET @SQLcmd = '';

SET @DatabaseName = ';';

SET @PrevDatabaseName = '';

CREATE TABLE #tmp (

DatabaseName SYSNAME NULL

,FileId INT NULL

,FileName VARCHAR(2000) NULL

,SQLcmd VARCHAR(MAX) NULL

);

INSERT  INTO #tmp

SELECT DISTINCT

DB\_NAME(dbid) AS DatabaseName

,FileId

,FileName

,CONVERT(VARCHAR(MAX), '') AS SQLcmd

FROM master.dbo.sysaltfiles

WHERE dbid IN (

SELECT dbid FROM master.dbo.sysaltfiles

WHERE SUBSTRING(FileName, 1, 1) IN ( 'G', 'F' ) )  -- Put the appropriate drives here

--AND DATABASEPROPERTYEX(DB\_NAME(dbid), 'Status') = 'ONLINE'

AND DB\_NAME(dbid) NOT IN ( 'master', 'model', 'msdb', 'tempdb', 'ditribution' )

ORDER BY DatabaseName, FileId, FileName;

UPDATE #tmp SET

@SQLcmd = SQLcmd = CASE WHEN DatabaseName <> @PrevDatabaseName

THEN CONVERT(VARCHAR(200), 'exec sp\_attach\_db @DBName = N''' + DatabaseName + '''')

ELSE @SQLcmd

END + ',@FileName' + CONVERT(VARCHAR(10), FileId) + '=N''' + FileName + ''''

,@PrevDatabaseName = CASE WHEN DatabaseName <> @PrevDatabaseName THEN DatabaseName

ELSE @PrevDatabaseName

END

,@DatabaseName = DatabaseName

FROM #tmp;

SELECT

'USE master;' + CHAR(13) + CHAR(10) + 'GO'

SELECT

SQLcmd

FROM

( SELECT DatabaseName, MAX(SQLcmd) AS SQLcmd

FROM #tmp

GROUP BY DatabaseName

) AS SQLcmd;

--DROP TABLE #tmp

GO

If you plan on moving the databases to different drives you can easily modify this script to change the attach location.

**One last Important Consideration**  
Keep in mind that when the mdfs and ldfs are connected to the new database instance they will be upgraded to the new version and this action cannot be undone. So if a rollback situation occurs, you cannot simply move the LUNs back to the old server and reattach.  Your only option here is to restore the databases you took prior to the upgrade. So make sure that before you begin the actual migration you create and save final backups of the databases on the old server.

I suggest you make a FULL backup of all the databases several hours before you block access to the SQL Server instance. As with methodology #3 above, make sure these are the last full backups taken. Then make one last DIFF backup and save them all (fulls and diffs) in a safe place. It is probably a good idea to have a restore script for all the databases pre-defined so that you are not scrambling to recover if this scenario presents itself. Keep in mind that any transactions incurred on the new server will be lost unless you take steps to find the data differences and import them back.

[That’s all I have to say about that.](http://youtu.be/Otm4RusESNU)This is by no means a complete step-by-step plan to migrate SQL Server, but hopefully it helps in your quest for the seamless migration.