The Brent Ozar Unlimited® SQL Server Upgrade & Migration Checklist

Last Updated: December 2015

1 What's in this document?

This is a checklist for all the things you need to think about and plan to go to a higher version of SQL Server. This is NOT a setup guide or installation checklist. For that, head to BrentOzar.com/go/setup!

2 Plan your change

Spoiler: if you do your migrations right, planning the change is way more work than executing the change. But the bright side is that when you do it well, the change itself is fast and pain-free!

2.1 Gather information

☐ How much downtime is allowed during the migration and how much data loss is allowed for databases on this instance?

- If you don't know these numbers already, go straight to our High Availability Planning Worksheet.

☐ What is the business' tolerance for going over downtime or experiencing performance problems after the migration?

- The tighter this is, the more bulletproof your plan must be. No pressure.

☐ Do you have tools to run load tests for the databases you'll be upgrading with the specific application patterns against a new environment ahead of time which allow ramping up activity?

☐ Is the environment physical or virtual? Is it a cloud environment? (One of the benefits of virtual and cloud environments is the ease of spinning up a new VM to migrate to.)

2.2 Make Decisions

☐ Will you be doing an in-place upgrade or migration?

- For minor version changes such as CTPs and Service Packs, most DBAs choose to do in-place upgrades. You've still got to test, and things can go wrong.
- For major version upgrades (such as SQL Server 2012 to 2014), most experienced DBAs prefer to migrate to a fresh install rather than perform in-place upgrade for production servers. This option also has a "rolling upgrade" option to reduce downtime.
- Read more: Why you shouldn't upgrade SQL Server.

☐ Will you use the new cardinality estimator in SQL Server 2014? This is controlled by the database compatibility level.

- Only use this by default if you can test it ahead of time outside of production – the load testing question above informs this decision.
- Read more: Careful testing the new Cardinality Estimator.

☐ What Edition of SQL Server will you move to, how many cores will you license, and have you purchased software assurance?

- Learn about SQL Server Licensing, simplified into 7 rules.
- You can change your SQL Server Edition in place… sometimes. Read about changing from Standard to Enterprise here.

☐ What specific Service Pack and CTP will you upgrade to?

- If you care about performance, Microsoft recommends that you care about cumulative updates, too!
- For SQL Server 2012 and 2014, review KB 2964518, "Recommended updates and configuration options for SQL Server 2012 and 2014 with high-performance workloads"
- Read more: Choosing the right SQL Server Version is trickier than you think.

☐ If you're migrating, what will be your cutover plan to the new server?

- Technologies like Database Mirroring and Transaction Log Shipping can be used to minimize downtime when cutting over.
- DNS "friendly names" for SQL Server can be used in connection strings to make changes simpler and less manual.

2.3 Reduce your risks

☐ Map out your "Rollback / Roll sideways" plan

- When it comes to migrations, you often don't have a rollback plan without losing data. Once you restore a database to a higher major version of SQL Server, you can't take it back to a lower version! (In other words, you can't detach a database from SQL Server 2014 and attach it to SQL Server 2012 or prior.)
- But you must have alternate plans available: what will you do if your new instance has major performance problem, repeatedly crashes, or SQL Server continually stack dumps?
- Read more: What happens if that doesn't work?

☐ When will you upgrade your pre-production (aka "staging") environment?

- Pre-production environments are typically built on the same hardware or prior generation hardware as production. These are used for testing new versions of SQL Server, service packs, hotfixes, and code releases.

☐ When will you upgrade your development environment?

- Most companies prefer to upgrade development environments several months before production to burn in new versions. However, this can impact testing and shipping new features.
- Read more: Do you have a development environment?

☐ Have you developed a test plan for your new hardware to make sure it's resilient to failure before you move in?

- Sometimes people forget to purchase redundant components, or accidentally goof when configuring it. Sometimes the hardware you get just doesn't work like it should! The only way to know for sure is to test it—and we're talking drives, network components, and everything in between.

3 Make a plan for HA/DR and scale-out technologies

Installing and configuring High Availability technologies like Database Mirroring, Availability Groups, and Failover Cluster instances is complex—and it's out of the scope of this document.

- While we've got you here, please note that Windows version is a critical choice for Availability Groups and Clusters, and that's a critical factor in upgrade and migration planning. Read more about why Windows Version matters.

Scripting out and configuring Replication Configuration (Transactional Replication, Peer to Peer, Merge), requires similar specialized wizardry.

With all of these technologies, make sure that you include them very carefully into your migration planning and testing and that all the steps are customized for your implementation.

The more complex your environment, the more a staging environment is absolutely necessary.

4 Look out for ~~falling rocks~~ breaking changes

☐ Are you on a database compatibility level that isn't supported in the new version? Not every version of SQL Server supports every compat level—for example, the lowest compatibility level on SQL Server 2014 is 100.

☐ Are you using Deprecated Features that aren't available in the new version you'll be using?

- The "SQLServer: Deprecated Features" performance counter allows you to see which deprecated features have been used since startup, like this:

```
SELECT object_name, instance_name as
deprecated_feature, cntr_value
FROM sys.dm_os_performance_counters
WHERE
    object_name like '%Deprecated%'
    and cntr_value > 0
ORDER BY deprecated_feature;
GO
```

- Decode the results of that query against the lists of Discontinued Features in Books Online.
- Review compatibility level changes if you plan on raising it.
- Disclaimer: in practice this can be more time consuming than you'd like, because you'll usually find that some Deprecated Features show up which are just being triggered by built-in tools like SSMS and the SQL Agent. Eliminate "normal" noise in this view by comparing against an idle instance of the same version.

☐ Do any databases require the "trustworthy" property?

- Look closely at the is_trustworthy_on column when you query sys.databases below. The trustworthy property doesn't restore with the database—if you deem it trustworthy, you need to set it *after* it's restored and becomes read/write.

☐ Are you using SQL Authentication?

- You'll need to use special scripts to transfer your logins. Microsoft's KB 918992 is here for you.
- Practice using sp_change_users_login and keep Microsoft's script available in case you run into a problem with orphan users.

☐ Do you have CLR Assemblies that use external or unsafe access permissions?

- You may need to be very careful with the "database owner" setting on the database on restore/migration. See KB 918040.
- To check for assemblies, in each database run the following command. The permission_set_description column will let you know if external or unsafe access is configured:

```
SELECT * FROM sys.assemblies;
GO
```

☐ Are you using Transparent Data Encryption on any databases? You'll need special procedures to restore them to migrate. Or restore them at all! If you've never done this before, start practicing pronto. (If you don't back up the keys, your database is un-restorable. Seriously.)

At this point, you might be wondering, "Why haven't they suggested running the Database Upgrade Advisor"? We don't have anything against the database upgrade advisor—and if you've got active development / pre-production environments and can run it against them without a hassle, by all means do that too! But it's not always clear what the upgrade advisor is or isn't checking, so we find that most DBAs like to do specific checks themselves and understand the process.

5 Script and document your current configuration

You might think that you can skip these steps if you're doing an in-place upgrade. Don't! Seriously, *don't skip this*!

Imagine that you have an in-place upgrade that goes fine when you test it elsewhere, but goes horribly wrong in production, and you end up with corrupted system databases. Do your future self a favor and document and script everything out.

☐ Record your current server level configuration options. Query the full list and paste it into a spreadsheet:

```
SELECT * FROM sys.configurations;
GO
```

- There's a ton of server level settings. You may not want to migrate the exact settings you have now, but you definitely want to record them so you know what they were later. (They may not be right for the new version, the new instance, or let's face it, even the one you have now!)
- After you record your settings, figure out which ones have been changed. Our free sp_Blitz® script identifies non-default server level configuration items. Then research to see which settings you want to carry over.

☐ Record your current database settings. Query the full list and paste it into your spreadsheet:

```
SELECT * FROM sys.databases;
GO
```

- If you're migrating to a new instance, most of these settings will backup and restore with the database. But it's important to document what you have, because things could go wrong and someone could go pushing buttons.

☐ Linked Servers

- Script out every linked server on the instance.
  - To script multiple logins at once, click "View", then "Object Explorer Details." In the new Details pane that opens you can highlight all linked servers and right click and script them at once.
- If your current SQL Server version is lower than SQL Server 2012 SP2, you may be able to lower permissions on linked servers post-migration.
- Learn more: Watch our linked server performance and security video

☐ Logins

- Script out your SQL Server logins. If you're using SQL Authentication, you'll need a special script. Start here.
- While it's always a good idea to follow the Principal of Least Permissions, if looking at your logins list fills you with horror, consider reducing permissions gradually *pre-migration* by creating new logins with lower permissions and migrating your users to them. Breaking all the logins during a migration usually doesn't make it any more fun (been there, done that, there was no t-shirt).

☐ SQL Server Agent Jobs

- Script out all of your SQL Server Agent jobs.
  - To script multiple jobs at once, click "View", then "Object Explorer Details." In the new Details pane that opens you can highlight all linked servers and right click and script them at once.
- Jobs should be set up to write their history to logs on the file system. Make a copy of the scripts for installation and test and modify them to work with the directory structure on the new server.

☐ SQL Server Agent Alerts

- Script 'em all out! The same "Object Explorer Details" trick above works.
- Note that Alerts usually trigger Agent jobs or notify operators. If you're going to have different jobs or operators on the new instance

☐ SQL Server Agent Operators

- Who you gonna call? SQL Server Agent Operators

☐ Database Mail Settings

- Expand the Management folder, right click on "Database Mail" and… *THERE'S NO SCRIPT OPTION*. Yeah, that's normal. This is one of the few things that doesn't auto-script for you.
- Go into the database mail configuration wizard and take screenshots of your setup and save those.
- If you'd like to script this, you can plug the information you collect into a template. Click "View", then "Template Explorer" and modify the "Database Mail" -> "Simple Database Mail Configuration" template.

☐ Script out/back up all keys and certificates
- Books Online article on backing up database master keys
- Books Online article on backing up certificates

6 Get your new SQL Server ready to move in (for migrations)

☐ Set up and test your Server and install and configure your SQL Server Instance. We really like our free SQL Server Setup Checklist.

7 Pull everything together into your migration plan

This is where everything comes together. Every go-live checklist is different. You might have a spreadsheet with 20 lines on it, and you can X them off. Or you might have a 50-page document, complete with screenshots.

☐ Write your migration plan. Set up and test absolutely everything you can ahead of time and do a dry run, then reset the environment and do it again. Your migration plan is everything that you can't do ahead of time and have to do live.

☐ Have someone review your migration plan. Does it make sense to them? Do they see ways to do it more easily, or potential risks and problems you've missed? Feedback is extremely helpful, it's very easy to have blind spots if you do this alone.

☐ One you have your migration plan finalized, test it again. Time it. Does it fit within your allotted time for the change? Does it still allow room for error? (Not sure where to test it? Scroll on up and read about pre-production environments. They're pretty awesome, and this is exactly why people love 'em.)

☐ Have someone approve your migration plan. Seriously, you put this much time into it, impress your boss' boss!