## Patch Management Strategies for SQL Server Deployments

## **Patching Strategy Goals**

- Stay as up to date as possible to remain supported
- Improve reliability, health, stability, fix bugs
- Minimize downtime
- Minimize the number of versions supported

## **Breaking Down Version Numbers (installed Base Company xxx)**

SQL Server Build	SQL Server Product Name	# of Deployments
8.00.1100	SQL Server 2000 – SP3	1
8.00.761	SQL Server 2000 – SP3	2
9.00.1399.06	SQL Server 2005 RTM	3
9.2.3042.00	SQL Server 2005 – SP2	1
9.3.4035.00	SQL Server 2005 – SP3	2
10.1.2531.0	SQL Server 2008 – SP1	7
10.3.5500.0	SQL Server 2008 – SP3	2
10.51.2500.0	SQL Server 2008 R2 – SP1	2
10.52.4000.0	SQL Server 2008 R2 – SP2	4
10.53.6000.34	SQL Server 2008 R2 – SP3	4
11.0.2100.60	SQL Server 2012 RTM	13
11.1.3000.0	SQL Server 2012 – SP 1	6
11.2.5058.0	SQL Server 2012 – SP2	5
11.3.6020.0	SQL Server 2012 – SP3	26
12.0.2000.8	SQL Server 2014 RTM	3
12.1.4100.1	SQL Server 2014 – SP1	13
12.2.5000.0	SQL Server 2014 – SP2	19
13.1.4001.0	SQL Server 2016 – SP1	3

## **Challenges of Patching and Updating**

- What do you apply?
- Frequency

"Patch Tuesday"  $-2^{nd}$  Tuesday of every month Bleeding edge vs. back of the pack

- Downtime
  - Uptime number (planned vs. unplanned) what counts?

Unavailability of server/instance/DB/hardware during patching

- Lack of testing and/or environments to test in
- Can create version issues if not careful How many do you *really* want to support?

## **Types of SQL Server Updates 1**

- All SQL Server updates are cumulative
- Hotfixes (QFEs, Critical on Demand [COD], On Demand [OD])
  - o Meant to fix a specific issue
  - Do not necessarily need to consume individually
  - o May or may not be public
  - Testing usually only specific to this fix

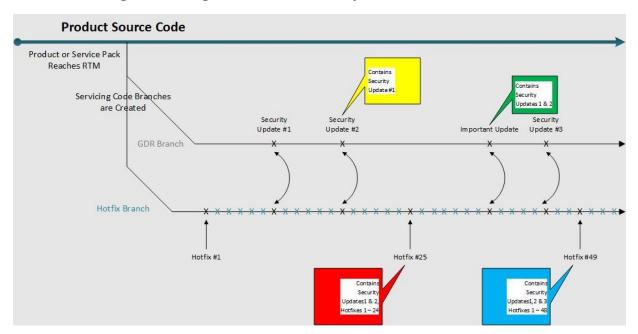
## **Types of SQL Server Updates 2**

- Cumulative Updates (CUs)
  - o Released every 2 months for supported branches
  - Has all CODs to date and other fixes
  - Sometimes will introduce functionality
  - o Better testing, but not as comprehensive as a SP
- Service Packs
  - Has everything in CUs +
  - May introduce functionality
  - Best amount of testing
  - o Will put you back on GDR branch if not already

## **SQL Server GDR vs. QFE (Hotfix)**

- Two major code "branches"
  - o GDR (general development release)
  - Hotfix/QFE
- All GDRs exist on hotfix, but not all hotfixes on GDR branch
  - o Means updates with both GDR and hotfix have two different versions of the files
- Goal: stay on GDR if possible

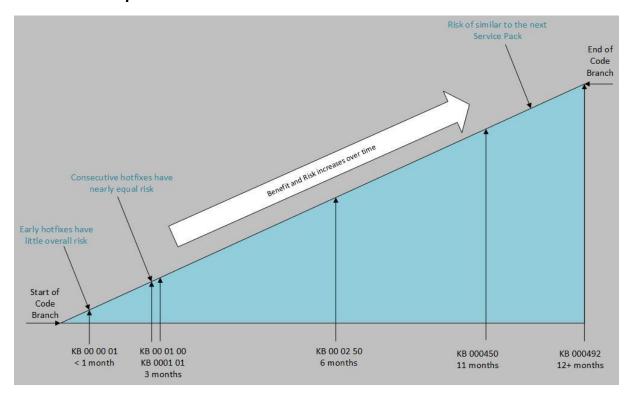
## **Understanding Branching and Cumulative Updates**



#### What about Windows and Hardware?

- Do you know when they will be patched/updated and are you ready?
- Do you know what will be affected?
- Do you know if an outage to SQL Server required?
- Beware untested driver updates especially storage ones
- Some updates such as firmware can brick hardware

## **Patch Consumption and Risk Assessment**



### **Application vs. SQL Server Considerations**

- Does it support only specific releases of SQL Server?
  - In-house vs. third party especially
  - May hold you back internal patching/upgrade/version policy vs. what application supports
- Features of SQL Server it uses
  - o Deprecation?
  - Breaking changes?
- Does it force things like compatibility level on DBs?
- Upgrade vs. patch/update

#### **Testing**

- Most important thing you can do
- Test update against target application(s) Dev/QA
  - o Ensures no surprises
  - Allows you to fix issues before they are seen in production
  - o Full or partial regression test?
- Test process of update installation IT/DBA
  - Know how long it takes
  - Know how it works
  - o Test different conditions (example: cluster vs. standalone) if applicable

## **Keys for Patching Strategy Success**

- Automate, automate, automate
- DOCUMENT !!!
- Don't deploy complex architectures
- Evaluate hotfixes carefully
  - Don't apply "just because"
- Test SQL Server CUs even if don't apply in production
  - o Reduces risk down the road
  - Smaller delta to test later if just applying certain releases
  - o Since SQL Server is cumulative, you'll get what's in it at some point
- IT/DBA need sandbox environments to test patches
- Server/HW patches/updates need just as much scrutiny as hotfixes
- Communication between teams
- Change management, including go/no go and rollback procedures
- Backups
  - o Should back up DBs, objects before any major update
- Solid D/R plans
  - O What will you do if things go totally south?

#### Tools?

- WSFC Validation (no, really...)
- Slipstream SQL Server 2008+
  - 2008/R2 PSUSOURCE, CUSOURCE
  - o 2012 UPDATEENABLED, UPDATESOURCE
- SQL Server 2012+ during Setup GUI
  - Can control behavior by modifying DEFAULT.INI
- Free tools
  - Windows Server Update Services
    - SQL (non-clustered), Windows
  - Cluster Aware Updating
    - Uses WSUS
    - SQL Server FCI support 2012 SP1+ not AGs
  - o System Center Configuration Manager

#### 6 steps to help decide when you must patch...and when it's OK to wait

SQL Slammer worm reminded us of the importance of patching vulnerabilities in computer software. Most successful computer attacks exploit well-known vulnerabilities, for which patches exist. The problem is that hundreds of patches are released each month, many of which apply to OSes and applications residing in your organization's network. How do you know which patches to install, and which to ignore? And what's the proper order and process for installing them?

Patch management is a complex process, and I can't cover all the variables here. But I can distill the process into six general steps. The importance of each stage of the patch process--and the amount of time and resources you should spend on it--will depend on your organization's infrastructure, requirements and overall security posture.

**Step 1:** Develop an up-to-date inventory of all production systems, including OS types (and versions), IP addresses, physical location, custodian and function. Commercial tools ranging from general network scanners to automated discovery products can expedite the process (see Resources, below). You should inventory your network periodically.

**Step 2:** Devise a plan for standardizing production systems to the same version of OS and application software. The smaller the number of versions you have running, the easier your job will be later.

**Step 3:** Make a list of all the security controls you have in place--routers, firewalls, IDSes, AV, etc.--as well as their configurations. Don't forget to include system hardening or nonstandard configurations in your list of controls. This list will help you decide how to respond to a vulnerability alert (if at all). For example, let's say you learn that OpenSSH has a vulnerability that may allow a buffer-overflow attack, but from your list of controls you know you don't allow the SecSH protocol through your firewall. If nothing else, that knowledge gives you more time to react.

**Step 4:** Compare reported vulnerabilities against your inventory/control list. There are two key components to this. First, you need a reliable system for collecting vulnerability alerts. And second, you need to separate the vulnerabilities that affect your systems from those that don't. Some companies have staff dedicated to managing this process; others use vulnerability reporting services.

**Step 5:** Classify the risk. Assess the vulnerability and likelihood of an attack in your environment. Perhaps some of your servers are vulnerable, but none of them is mission-critical. Perhaps your firewall already blocks the service exploited by the vulnerability. In general, to classify and prioritize the risk, consider three factors: the severity of the threat (the likelihood of it impacting your environment, given its global distribution and your inventory/control list); the level of vulnerability (e.g., is the affected system inside or outside perimeter firewalls?); and the cost of mitigation and/or recovery.

**Step 6:** Apply the patch! OK, so now you have an updated inventory of systems, a list of controls, a system for collecting and analyzing vulnerability alerts and a risk classification system. You've determined which patches you need to install. Now comes the hard part: deploying them without disrupting uptime or production. Fear not, there are several tools that can help you with the actual patch process (see Resources, below). Evaluate these tools in terms of how well they fit your environment and budget. In some cases, manual patch maintenance may be more cost-effective. But in most cases--particularly for multiple servers or server farms distributed across multiple locations--some type of automated patch system will more than pay for itself.

# Patching SQL Server Availability Groups (Hotfixes, Cumulative Updates and Service Packs):

If you use SQL Server Availability Groups (AGs), this is a tough responsibility. There are a *lot* of updates released for critical issues. And sometimes there are updates to the updates, because.... oops. And sometimes the updates may cause you downtime.

Here's the lose-lose situation that the Availability Group DBA ends up in:

- If you don't apply updates, you could hit a critical performance issue that brings down your environment which was fixed in an existing cumulative update or hotfix. An update you should have known about.
- If you apply updates regularly, you can cause yourself downtime by issues introduced in the updates. And here's the worst part: testing reduces your risk of this, but doesn't prevent it.

#### High Availability is Supposed to Reduce Downtime-Including Downtime for Patching

One of the big reasons DBAs like High Availability solutions is that we want to reduce downtime during patching. If you've ever done much patching, you know that it can take a while to restart servers sometimes. Sometimes the server hangs on restart and you may have to connect to a special server management card to nudge it along. (iLo, DRAC, etc.)

If you're using Database Mirroring, a Failover Cluster, or Availability Groups, you can reduce downtime in patching: you can fail your SQL Server resources around so that there are only short downtimes and your customers aren't dependent on the server to come back. You want these features so you have less to worry about, not more.

#### **PATCHING an Availability Group Can Cause Unplanned Downtime**

Check out this recent critical problem impacting multiple cumulative updates across SQL Server 2012 and 2014. If you apply these updates, you may run into a critical blocking issue which you can only fix by:

- Turning off automatic failover
- Restarting the SQL Server instance on the primary replica. Yep, that's an outage
- Turning on automatic failover

Update: KB 3033492 has now been published which gives more information on options to avoid the issue (spoiler: it requires an outage), or other ways you can attempt to correct it by killing sessions (which may not work).

#### Which of These Downtime Scenarios is Worse?

- **Be an unlucky optimist.** Follow the steps in the KB. Hope that you detect the problem during your patching window if it occurs. There's a chance that the issue might be detected after your planned patching window has ended (and you've gone back to sleep), and then you've got an unplanned downtime on your hand and unhappy customers.
- Be a beleaguered pessimist. Proactively take the steps above when applying patches to prevent the bug from occurring. This is more work for you, more steps for you to carry out and validate (you did put that back in automatic failover, right?), and more outages for your customer. You're now not just failing back and forth between nodes, you're doing an extra restart. And restarting the instance underneath the primary replica with no failover is exactly the type of thing we expect High Availability to save us from.

#### This Couldn't Happen Again, Right?

Remember SQL Server 2012 SP1? It had an issue that impacted some installs where processes named msiexec.exe would keep running and use tons of CPU. It bloated out the registry. It was pretty nasty, and it raised the issue, "Should you trust Service Packs?" (Spoiler: no.)

That issue wasn't specific to AGs. But we now have a pattern where supported, released updates (even service packs) can cause major problems on a portion of installs and they are released anyway. Odds are, it will happen again.

#### But Microsoft Said I Should Only Install Cumulative Updates if I Need Them, Right?

Yep—that's what they used to say. It was dangerous advice to follow, because Service Packs are released very slowly these days.

But this changed with KB 2964518 – "Recommended updates and configuration options for SQL Server 2012 and SQL Server 2014 used with high-performance workloads." This article specifically recommends using the latest cumulative update, and gives a dramatic list of issues fixed in those updates.

#### Cumulative Updates ARE Especially Critical for AGs. Look at these Fixes...

It's possible you might still be thinking that it's safer to just not patch an availability group and hope for the best. That's not a good idea. Take a look at these fixes (as an example):

- "SQL Server 2012 experiences out-of-memory errors" http://support.microsoft.com/kb/2769594 (this is specific to AGs)
- "FIX: Out-of-memory errors related to a memory clerk in SQL Server 2012" http://support.microsoft.com/kb/2821783 (this is specific to AGs)
- "FIX: SQL Cache Memory decreases and CPU usage increases when you rebuild an index for a table in SQL Server (with AlwaysOnAG). 2012 SP2 CU2, 2014
  CU1" http://support.microsoft.com/kb/2958054
- "FIX: High "log write waits" counter value on a SQL Server 2012 instance" http://support.microsoft.com/kb/2809338 (This is specific to Availability Groups per the Symptoms)
- "FIX: Error when you back up a database that has case-sensitive collation by using VSS in SQL Server 2012 SP2" http://support2.microsoft.com/kb/2987610 (the Symptoms specifically list backing up AG secondaries)
- FIX: SQL Server 2012 or SQL Server 2014 instance shuts down when you join database as secondary replica during AlwaysOn Availability Groups configuration <a href="http://support.microsoft.com/kb/2929193">http://support.microsoft.com/kb/2929193</a>
- "Connection times out when you use AlwaysOn availability group listener with MultiSubnetFailover parameter" http://support.microsoft.com/kb/2870437 (This one is a Windows 2008R2 issue)
- "FIX: A memory leak occurs when you enable AlwaysOn Availability Groups or SQL Server failover cluster in Microsoft SQL Server 2012" http://support.microsoft.com/kb/2877100
- "FIX: "System objects not be updated (sic)" when you use AlwaysOn Availability Groups in SQL Server 2012 or SQL Server 2014" http://support.microsoft.com/kb/3002071
- "A hotfix that improves the performance of the "AlwaysOn Availability Group" feature in SQL Server 2012 is available for Windows Server 2008
  R2" http://support.microsoft.com/KB/2687741
- "Time-out error and you cannot connect to a SQL Server 2012 AlwaysOn availability group listener in a multi-subnet environment" http://support.microsoft.com/kb/2792139
- "FIX: Synchronization state of an AlwaysOn Availability Group replica may not be updated if primary is unhealthy" http://support.microsoft.com/kb/2897554

- "Troubleshooting automatic failover problems in SQL Server 2012 AlwaysOn environments" http://support.microsoft.com/kb/2833707 (This is a list of things you need to look out for that can make automatic failover not work which may not be obvious, not a hotfix.)
- "FIX: "Non-yielding scheduler" error occurs and AlwaysOn Availability Group transits to RESOLVING state" http://support.microsoft.com/kb/3020116
- "FIX: Scheduler deadlock on AlwaysOn Availability Group primary replica in SQL Server 2012" http://support.microsoft.com/kb/2869734/en-us
- "SQL Server 2012 service shuts down unexpectedly upon availability group replica role transition on a Windows Server 2008 R2-based failover cluster" http://support.microsoft.com/kb/2777201
- "FIX: SQL Server 2012 or SQL Server 2014 instance shuts down when you join database as secondary replica during AlwaysOn Availability Groups configuration" http://support.microsoft.com/kb/2929193

#### So what should you do?

How much time and effort do you have for reading about fixes, testing them carefully, watching blogs for out of release fixes and known issues, and establishing complex patching and production processes?

If you don't have at least an hour a week to keep on top of available fixes, and another eight hours a month (minimum) to devote to testing and deploying fixes, you don't have time for an AG.

Got more than two replicas? You need more time. Running Windows Server 2008R2? Well you need a lot more time, because you should be moving out of there.

#### Is there a cool way of performing CU updates for SQL Server on hundreds of machines?

No, and it has nothing to do with technology.

- Which servers are mission-critical 24/7 and can't go down?
- Which servers can only be taken down in specific time windows?
- Which servers have dependencies between each other, like database mirroring and AlwaysOn Availability Group replicas?
- Which servers have automatic failover mechanisms (like clusters) where you can patch the standby node first, then fail over once, and patch the primary without having to fail back?
- Which servers have vendor apps that required a specific hotfix that may not be included in the cumulative update you're about to apply?
- Which servers are running long batch jobs like data warehouse loads or backups that would take a long time to restart if you took it down in the middle of its operations?
- Which servers have pending sp\_configure changes that will take effect and surprise you when you restart?

#### **Best Practice for Patching SQL Server:**

Where can I find out what the latest patch level is for SQL Server?

My site of choice of patch levels of SQL Server is http://sqlserverbuilds.blogspot.com and https://support.microsoft.com/en-hk/help/321185/how-to-determine-the-version,-edition-and-update-level-of-sql-server-and-its-components

This site lists all the patches for SQL Server from SQL Server 7.0 right through to the very latest version (2016 at the time of writing). The great thing about this site is that it lists the patches, build levels and links to the patches in chronological order which makes it really easy to navigate and understand.

In addition to knowing what patches are out there, you should also be aware of what the current supported patch level is. The support lifecycle for SQL Server is documents on the Microsoft support website: <a href="http://support.microsoft.com/lifecycle/?c2=1044">http://support.microsoft.com/lifecycle/?c2=1044</a>. This site will show you the mainstream and extended support start and end dates for the different versions of SQL Server. Note that this site shows the minimum required patch level for Microsoft support, not the current latest patch level.

#### What type of patches are out there for SQL Server?

Patching SQL Server is actually very easy, and since SQL 2008 it has become even easier. One of the great things with SQL Server patches is that they are cumulative, so if you find yourself getting behind on your patching you will not have to apply all the patches in order. Broadly speaking there are 3 types of patch for SQL Server:

- Service Packs
- Security Patches
- Cumulative Updates

Service Packs should be considered as major patch levels and generally speaking it is the product at a certain patch level that forms the minimum supported build for Microsoft support. Once a Service Pack has been released, you have 1 year to apply that Service Pack in order to remain at a supported level.

Cumulative Updates are regularly released for SQL Server and these updates include bug fixes and improvements for SQL Server, however CU's are not mandatory and do not form part of the support lifecycle timelines. By that I mean that if the most recent patch for SQL 2012 is SP1 followed by CU6, you would only need to have SP1 installed to be at a supported level, and CU6 is optional. It may however be advisable to apply the CU's since they include many fixes and improvements, and it is better to apply a proactive approach to patching rather than waiting for an issue that affects a production server, and then frantically search for a patch to fix it. Cumulative Updates area applied on top of Service Packs, not as a replacement to them. Therefore, if you have SQL Server RTM build, but you want to apply CU6 that was released after SP1, you would need to apply SP1 and then CU6. If however you are already at SP1 level, but you have not applied any CU's, you only need to apply CU6 and not all CU's in order.

Security patches are additional patches that address specific security issues in SQL Server and should be applied as they are released. Since CU's include all updates since the last Service Pack, you would only need to apply the most recent CU which would include any previous security update.

#### What is the process for patching SQL Server?

If you are patching a stand-alone, the process is quite simple:

- 1. Copy the patch files to the server
- 2. Run the patch file and follow the on-screen instructions
- 3. Select the instances that you wish to patch
- 4. Allow the patch to finish
- 5. Reboot if a reboot is required (the patch will tell you if there were any locked files)

Stand-alone server, even multi-instance stand-alone servers are simple because the steps above are limited. You can apply the patch in an interactive mode as shown above, or you can apply it in a quite non-interactive mode as shown below:

<package.exe> /QUIET /INSTANCENAME=<instance>

or

<package.exe> /QUIET /ALLINSTANCES

so for example to install SQL 2008 CU1 quietly on all instances you would run this

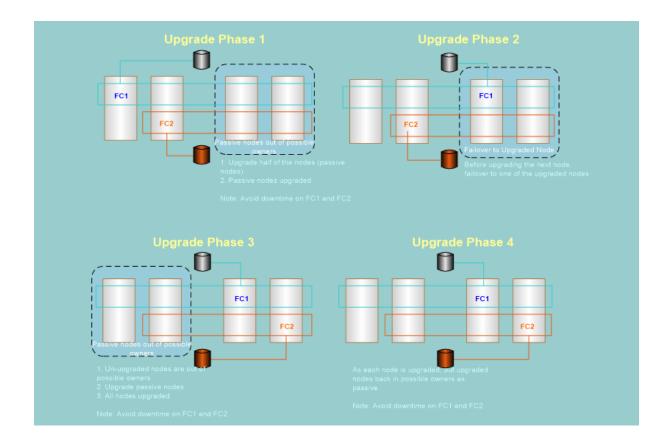
SQLServer2008-KB956717-x64.exe /QUIET /ALLINSTANCES

The quite options apply to stand alone and clustered installations, and if you have a number of instance to patch you may want to consider using that option for speed of patching. Obviously it takes longer to apply the patch when we have to render many forms to the screen, so you can bypass this by running in quiet mode from the command line. There isn't a best practise for running in GUI or command line mode, it is totally your choice, but I would certainly look at using the command line mode for speed of nothing else.

#### **SQL 2008 onwards (Clusters)**

With the introduction of SQL 2008, they removed all of the remote installation process from the installer. This also applies to the patching process as well which is very useful to you from a patching process as it means you can maintain the service availability whilst you are patching. With a SQL 2008+ cluster, you always patch the passive node rather than the active node. Patching the passive nodes means that the SQL Server is available on the active node and therefore customers can continue to use the system. The process is as follows:

- 1. Connect to a passive node in the cluster and run the patch installer
- 2. Follow the on-screen instructions, selecting the options that you wish to patch
- 3. Fail over the cluster once the install has finished
- 4. Now patch the remaining passive node



When you patching the passive and failover, the unpatched node gets removed from the possible owners list so that the cluster can't be failed back to an unpatched node. This can be important as some patches will update the internal database version of the installed databases, which would cause them to not start if you tried to run them on an unpatched node. You can alter this behavior at the command line and adding the following switch:

#### SETUP.EXE 2/FAILOVERCLUSTERROLLOWNERSHIP=0 | 1 | 2

- 2/FAILOVERCLUSTERROLLOWNERSHIP=0
  - Do not roll cluster ownership to the upgraded nodes.
  - Do not add this node to possible owners of the SQL Server cluster at the end of upgrade of this node.
  - o Note: this is not an option after the ownership has already transferred.
- 2/FAILOVERCLUSTERROLLOWNERSHIP=1
  - o Roll cluster ownership to the upgraded nodes (if this has not happened already).
  - Add this node to possible owners of the SQL Server cluster at the end of upgrade of this node.
- 2/FAILOVERCLUSTERROLLOWNERSHIP=2
  - This is the default setting.
  - This setting indicates that SQL Server setup will manage the cluster ownership as needed.

You can also use the cluster administration command line tool to do this, which is discussed in this article: http://support.microsoft.com/kb/958734

#### **Slipstreaming patching during Setup:**

As of SQL 2008 SP2, they introduced the ability to integrate Service Packs and CU's into the installation media of SQL Server. This has the great advantage of speeding up the patching process for new installations, as well as allowing us to address any setup related issues. Slipstreaming places the patch files into the installation media so that they are installed as the SQL instance is being installed, so that you only run setup once, and you have a fully patched and ready to go instance. It is a great time save, and I use this all the time for Service Packs and CU's for my demo systems. I won't re-document the process in this document as a full description of the process can be found on this blog:

http://blogs.msdn.com/b/petersad/archive/2011/07/13/how-to-slipstream-sql-server-2008-r2-and-a-sql-server-2008-r2-service-pack-1-sp1.aspx

This blog talks about SQL 2008 and 2–8 R2, however the process is identical for SQL 2012 and 2014, so simply replace the filenames with the versions that you are using and away you go. Please note that slipstreaming is not available for SQL 2005.