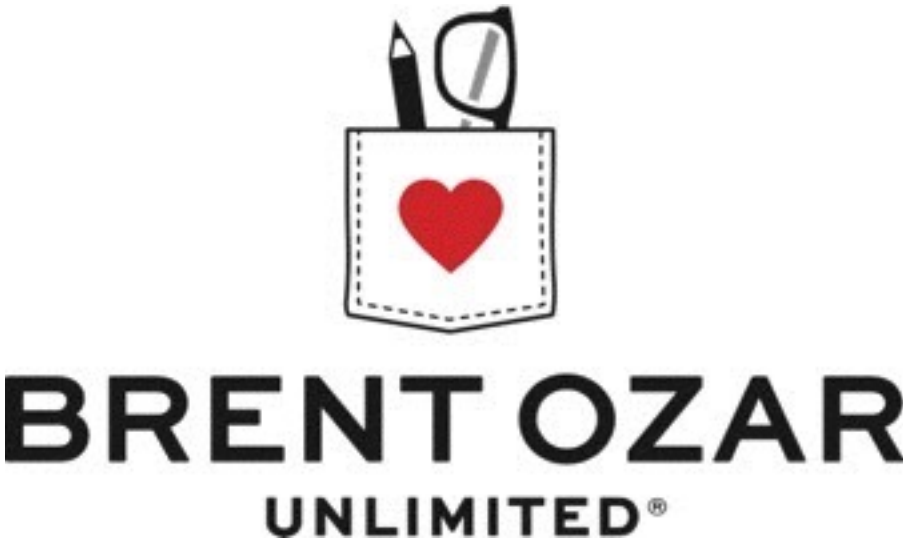

SQL Server Setup Checklist

Last Updated: May 2016

1. What's in this Setup Guide?

This is a summary of basic best practices for configuring SQL Servers.

Meeting the basic minimum configuration helps get you started on your way to a healthy SQL Server. However, configuring SQL Server is complex. This configuration doesn't provide:

- Configuration for virtualization
- Steps to set up high availability
- Performance testing / storage tuning
- Protection or testing of all potential hardware failures

2. Validate your hardware

SQL Server strives to be crash tolerant, but server level failures can lead to long outages, painful recovery times, and at worst data corruption.

2.1. All drives should use RAID (and not RAID 0)

Eliminate single points of failure at the storage level.

You should have, at minimum:

- RAID 1 pair beneath the system drive
- Multiple drives in a minimum of RAID 1 (often you want RAID 10 for performance) on all drives being used for SQL Server databases. This includes TempDB.

Even PCI-Express SSD devices should use RAID 1 (with two devices) so that you can tolerate the failure of any single device. For these devices, this may require the use of Software RAID at the Windows level.

The need for RAID is not mitigated or eliminated by high availability features or manufacturer's claim for redundancy within a single device. If a vendor claims their storage (or storage-interfering software, like software-based clustering) is good enough for SQL Server, review the Microsoft whitepaper SQL Server 2000 I/O Basics. It's old but good, including gems like write ordering, which has to be preserved by any underlying storage subsystem.

2.2. Get the right-sized operating system drive

Modern versions of Windows use a lot of space on the system drive over time to cache Windows updates. Typically if you have a mirrored pair of drives under your system drive, you want to allocate the whole thing to a single C drive and not partition out other logical drives.

50GB is frequently too small for a system drive at this point after time passes.

- If you have less than 50 GB, stop and get more.
- If you have only 50GB, check if you can get 100 or 128GB.

2.3. Team multiple physical network cards

If communication with a SQL Server is important, buy multiple network cards and team them for client communications. This allows server communication to continue if any single network card fails. Additionally, monitoring should be configured to detect the failure of a single network card.

Teaming multiple ports on a single network card does not mitigate this risk.

This configuration is particularly important in any solution including Windows Failover Clustering / SQL Server Availability Groups, but is a best practice for standalone servers as well.

For each NIC, you'll need to update firmware/drivers. You may need to install upgraded teaming software, depending on what type of cards and teaming you are using. Windows Server 2012 and newer have network teaming built in, but for prior versions of Windows, you'll need to get network teaming software from your server vendor.

For more details about Windows 2008 networking, check out the Windows Server Core Team's Networking series (part 1, part 2, part 3, and part 4). These posts are designed for failover clusters, but the core concepts are still important because a SQL Server's network is so important. If you're using AlwaysOn Availability Groups, for example, be aware that even a temporary network drop can take the databases down.

We'll test the network teaming after Windows has been installed.

2.4. Test and Configure an ILO/RIB card

These cards provide remote access to the server via a separate network connection. These can be extremely useful when a server is non-responsive and can't be managed through accessing the operating system remotely.

Test the ILO/RIB card by logging into it and performing system related tasks (like rebooting).

Make sure the ILO/RIB card properly identifies the server it's connected to. Many a server has been rebooted accidentally due to this being skipped.

2.5. Provision storage for the OS and for SQL Server

We don't want to pile everything on the C drive, so create other volumes for SQL Server to spread across. Ideally, we'll have separate logical drives for:

- SQL Server application folders – D:
- SQL Server databases data files – G:
- SQL Server database log files (including TempDB) – L:
- TempDB data file(s) – T:
- Backups – H:

* *Your drive letters may vary*

Depending on your storage devices (local disks, SAN, etc.) these logical drives may or may not share the same physical hardware.

2.6. Provision storage for backups

For SQL Server backups, you can provision storage on the server itself, or on remote storage.

Keep your backups on two independent types of storage. If you're only storing your backups on your production server or on your production SAN, and that fails, you won't have any backups.

Make sure you have enough storage to keep:

- Two copies of multiple days of full backups
- Two copies of multiple days of transaction log backups

To determine how many days of backups you need to keep online, ask business owners, "If data was accidentally deleted or corrupted two weeks ago (or whatever time period), would we need to restore to the night before? Or right before it happened?"

Backup storage should use RAID, but:

- Definitely not RAID 0 because it has no redundancy, so it can't survive even a single drive failure. Even if you're okay losing your backups temporarily, the bigger problem is reinitializing transaction log shipping from scratch just because a few log backups were lost.
- Ideally not RAID 5 either because it's not so good at long streams of concurrent random activity. You'll be backing up multiple database servers here, plus reading the share with a file backup program to sweep the backups off to tape, plus possibly reading the share from multiple servers if you're doing transaction log shipping.

This means RAID 10, which sounds pretty crazy for a backup share – RAID 10 is the most expensive RAID level to implement. Remember, though, it's completely okay to use huge drive sizes here, so it shouldn't take too many drives, thereby keeping cost down. (Or just use RAID 5 and see how performance goes, but don't be surprised if you need to widen your backup windows due to long backups.)

2.7. Design your backup strategy to match your RPO and RTO

It sounds odd to talk about backups before we've installed Windows, but your backups are critically important. We've seen many a server go live only to find out the hardware couldn't possibly back up the databases quickly enough to match the business requirements for downtime and data loss.

Get your business's Recovery Point Objective (RPO) and Recovery Time Objective (RTO) for high availability and disaster recovery in writing. Learn more about these key concepts at http://www.brentozar.com/go/fail.

Then, design an availability, backup, and recovery strategy that can meet these goals. A highly available backup design can mean you:

- Keep multiple copies of backup files (full and log)-- not on the same storage, and never all on the production server
- Perform frequent transaction log backups. If it's not OK to lose data, you want to be doing log backups every minute (and also investing in a secondary high availability solution because you could still lose data with log backups every minute!)
- Copy off backup files quickly to a secondary location like tape or your DR site

3. Choose, install, and patch Windows

3.1. Choose Your Windows Version Carefully.

Windows Server 2012 R2 is highly recommended for servers that need to be highly available. If you're using failover clusters or AlwaysOn Availability Groups today, or if you plan to at any time in the future, avoid Windows Server 2008R2 and older versions due to problems with the clustering code. Read more here: http://www.brentozar.com/archive/2014/03/need-high-availability-sql-server-windows-server-2012/

3.2. Apply the Right Windows Patches and Hotfixes.

Start by getting the server up to date with Windows Update, but you're not done yet.

On servers with large amounts of memory or large amounts of processors, you may need to apply special hotfixes that are not provided in Windows Update by default (specifically on Server 2008R2).

If you're using clustering or AlwaysOn Availability Groups, apply the following patches:

- Windows Server 2008R2 SP1 – http://support.microsoft.com/kb/2545685
- Windows Server 2012 – http://support.microsoft.com/kb/2784261
- Windows Server 2012 R2 - http://support.microsoft.com/kb/2920151
- Prerequisites for using SQL Server 2012 AlwaysOn Availability Groups

If you are using Windows Server 2012 R2 with more than 512GB of memory, review KB 2921368

3.3. Configure the Windows page file.

SQL Server does not need a giant page file. If you're installing other applications on the server (which we don't recommend-- SQL Server should be isolated), you may need a larger page file.

If SQL Server is the only major service running on the box, we typically create a 2GB size page file on the system drive. Page file size can be found in the system properties of Windows Server.

Beware removing the page file altogether. For Microsoft guidance on the minimum page file size and placement required to create a kernel memory dump file for different operating systems, see KB 254649.

3.4. Set anti-virus exclusions.

It's OK to run anti-virus software on a SQL Server – after all, we know the kinds of web sites you visit. You need to configure exclusions for all SQL Server files per Microsoft's guidelines:

http://support.microsoft.com/kb/309422

If you have additional tools that restrict creating/modification of files, they should also have exclusions set.

4. Configure and test storage hardware

A lot of storage configuration can only be done after the operating system is installed, so here we go!

4.1. Format the drives with 64K allocation blocks

SQL Server likes to have a block size of 64K on most storage platforms. If your storage platform gives specific advice to use a different block size for SQL Server, you may want to test which is better. Otherwise, use a 64K block size.

Note: this only applies to drives holding SQL Server database and log files (including tempdb). Your C drive / system drive should be separate and 4K block size is appropriate for that logical drive.

If you're using Windows 2008 or newer, you can format your drive through the GUI with a 64K allocation unit. If you're using an older OS, you should use DISKPART to make sure partitions are aligned.

Microsoft KB article 929491 covers some of this in technical detail, but the Microsoft Exchange Team blog does a much better job of explaining why disk partition alignment is important for performance. Use this command in Diskpart.exe:

```
CREATE PARTITION PRIMARY ALIGN=1024
```

The 1024 number will work with every major SAN out there. Gurus can use smaller numbers when they know a specific SAN very well, but small numbers only save you a few hundred kilobytes in the entire disk. Be safe, be sure, and use 1024, and you won't get burned if the underlying SAN structure changes, like with today's virtual storage.

You can't do any of this with the C drive – Windows doesn't give you these options during install – but C drive performance really isn't an issue as long as we're not storing database files there. (And you're not going to do that. Please, seriously, tell me you're not going to do that. Even if it's a VM on shared storage, you don't want to put the database files where they might grow, force the OS to run out of space, and cause a very ugly crash. Thanks, I appreciate you telling me that you knew that already and would never do that.)

4.2. For SANs: If the SQL Server Uses Host Bus Adapters (HBAs)

Update the HBA firmware. Downlevel HBA firmware can cause all kinds of nasty problems, especially in clustered servers. Generally, these updates can't be done online while the server accesses data, so it's better to get the code up to date before the box goes into production. For HP servers, this firmware isn't shown in the System Homepage: install Emulex HBAnyware on the server instead, and it will flash the HBAs inside of Windows without a reboot. HBAnyware is available in the HP Support site by searching for downloads for the HBA's part number instead of the server's part number. This is the only driver/firmware at HP that works this way.

Set up multipathing drivers. Sometimes this is done by the storage team, but the DBA should get involved enough to understand whether the multipathing is active/active or just failover.

Test the multipathing & failover. Start a huge file copy to each array, and do them all simultaneously. Go into the datacenter and pull one fiber cable out. Watch to make sure the file copy continues. Some SAN drivers will take 10-15 seconds to fail over, but the file copies should not be disrupted, period. If they're disrupted, the multipathing didn't work. Then plug the cable back in, and pull another cable. Again, the file copy should continue. Finally, while a file copy is running, ask the SAN admin to disable one of the SAN zones for the server – that way, the fiber cable will still be lit up, but the path to the storage will be gone. (This is a tougher failover method than just pulling the fiber.)

4.3. For SANs: If the SQL Server Uses ISCSI Storage

Set up multipathing. Database servers can't rely on one single network connection for iSCSI any more than a fiber-connected SAN can rely on one single host bus adapter. Ideally, two (or more) network cards will be connected to two different switches for redundancy, but at the very least, we need two network cards dedicated to iSCSI storage. The multipathing method can be active/active (meaning 2 gigs of throughput for two 1 gig NICs) or active/passive. Teamed network cards are not supported by Microsoft for iSCSI.

Test the multipathing. We usually see active/passive on a per-array basis – meaning, if you have two different iSCSI drive letters, then the multipathing drivers will put each drive on its own network card. The EMC and LeftHand multipathing appears to do this by default. Start multiple simultaneous drive copies and go into Task Manager, in the Network tab. Look at the bandwidth used by each network card. If a network card is sitting idle, then you're leaving performance on the table. Now is the time to tweak the multipathing software and ask questions of the vendor – it's easier to troubleshoot file copy performance than it is to troubleshoot SQL Server performance.

Test the failover. As with the fiber cable testing, start multiple simultaneous file copies to/from the network drives and pull one network cable out. If the file copy fails (if Windows throws an error) then SQL would have crashed. Tweak the teaming software until it can fail over seamlessly, and ideally it should fail over back and forth and go back to higher bandwidth levels as the networks come back online.

4.4. Benchmark the storage

Now that you're sure the storage hardware is set up correctly, start by running the portable edition of Crystal Disk Mark to benchmark your drive performance before you go live. Instructions are here: http://www.brentozar.com/archive/2012/03/how-fast-your-san-or-how-slow/

We recommend also running SQLIO. Instructions on how to run it are here: http://www.brentozar.com/archive/2008/09/finding-your-san-bottlenecks-with-sqlio/

Running these tools will not prevent problems, and these barely scratch the surface of the configuration and testing that we do in a server setup project with clients.

These tools will give you benchmark information for storage performance before you start-- and if you run into performance problems later, having this information available and being able to compare is extremely valuable.

You're looking for at least 200MB/sec of sequential read throughput per CPU core. If you can't achieve that throughput, odds are your SQL Server will never use large amounts of CPU power because it'll be sitting around bored waiting for the storage to deliver data faster. Low CPU usage isn't necessarily a good thing – remember, we paid for SQL Server licensing by the core, and we don't want to have a lot of cores just hanging out, smoking cigarettes outside of the break room. Or whatever it is they do when they're not working hard.

For more details on storage testing and why it's so important, watch Brent's session on Building the Fastest SQL Servers at Microsoft TechEd: http://channel9.msdn.com/Events/TechEd/NorthAmerica/2012/DBI328

4.5. Test network teaming with file copies

Remote desktop into the server and copy a large file (say, last night's production backup) from one network file share to another.

As you're copying the file, change the VLAN for one of your switch ports at the network level. This isn't something you can do in the server – if these words sound like gibberish to you, get a member of your network team involved. We don't want to outright disable the network port – we want Windows to think the network is still up, but we don't want the packets to actually go anywhere.

When this happens, your remote desktop connection needs to stay up, and the file copy needs to continue. Pauses are okay, but errors are not. Watch NIC utilization and make sure that teams are working as configured, and that you get the transfer rate you expect.

If the cluster has multiple separate networks (like production, heartbeat, and backup), perform the test on each network. This test doesn't just check failover – it also checks to make sure that all of the network ports are able to route the necessary traffic.

5. Create service accounts and grant permissions

5.1. Choosing your service accounts

Configure Service Accounts for all SQL Server services. Microsoft recommends creating individual accounts for all services.

If you're using AlwaysOn Availability Groups and you need to support Kerberos authentication, all of the SQL Servers involved will need to use the same SQL Server service account. Books Online has more details.

If you're not using AlwaysOn AGs, consider a different service account per production server. If you use the same service account for every server, then when somebody repeatedly mistypes the service account password during an installation, you can lock out the service account, causing widespread havoc.

5.2. Granting the right permissions

Do not make the service account a local administrator. SQL Server's installer will automatically grant the least privileges required during setup.

Enable Instant File Initialization: Grant the 'Perform Volume Maintenance Tasks' right to the account that will be used for the SQL Server service (the engine, not the agent).

6. Install and configure SQL Server

6.1. Select the components you will install

The SQL Server Engine likes to be a server. It doesn't want to compete for CPU and memory resources with other services. While SQL Server has a max memory setting, that is only for the SQL Server engine itself – not any of its other services that happen to come in the same package.

Do you *really* need to install Analysis Services, Reporting Services, and Integration Services on the production instance? Only install the components that you know should be on the instance.

6.2. Install SQL Server and select your Service Pack and Cumulative Update

- Validate that you are using the RIGHT SQL Server version (including service packs and cumulative updates)
- To see the latest available: http://blogs.msdn.com/b/sqlreleaseservices/
- We recommend upgrading all Development / QA / PreProduction machines to the version you intend to use and testing it prior to going live.

Cumulative updates are truly cumulative. To install SQL Server 2008 R2 SP2 CU8, you need to install SQL Server RTM, then SP2, then CU8. (You do not have to install CU 1-7 individually.)

6.3. Make sure the TCP/IP Protocol is enabled

Configure this in the SQL Server Configuration Manager under "SQL Server Network Configuration." Enabling the TCP/IP protocol will only take effect after the SQL Server instance is restarted.

6.4. Test Instant File Initialization (IFI)

Create an empty database. Grow the data file by 5GB. If it doesn't complete immediately, IFI isn't working. (Revisit the previous step where it was granted.) If you've verified IFI is working, go ahead and drop the empty database.

6.5. Configure TempDB

By default, the TempDB files are put on the same drive as the SQL Server binaries. Even if the user chooses a custom install, TempDB still goes on the same drive as the other data files, and that's not a good idea either. Instead, the TempDB data files should be on their own dedicated drive.

Move TempDB to its own drive

In this example, we put the data file on the T drive and the log file on the L drive. (Important: Be aware the directory paths must already exist.)

```
use master;
GO
alter database tempdb modify file (name='tempdev', filename='T:
\MSSQL\DATA\tempDB.mdf', size = 1MB);
GO
alter database tempdb modify file (name='templog', filename='L:
\MSSQL\LOGS\templog.ldf', size = 1MB);
GO
```

We only set a 1MB file size because SQL Server does something tricky: even though we're telling it to use a different drive letter, it will look for this amount of free space on the drive TempDB currently uses! If SQL Server was installed on the server's C drive, for example, and we tried to create a 10gb TempDB file on a T: drive, that SQL command will fail if there isn't 10gb of free space on the C drive. Yep, it's a bug – get over it.

After this code runs, restart the SQL Server. That will create the new TempDB file on the new drive. Manually delete the old TempDB file on the original drive, because SQL Server doesn't delete that itself.

Grow that file and add additional data files

Now that TempDB is on the right drive, expand it to the full size you want, and then create additional TempDB files.

```
USE [master];
GO
alter database tempdb modify file (name='tempdev', size = ?GB);
GO
```

The current guidance from Microsoft in KB 2154845 is to use the same number of tempdb files as the number of logical processors up to 8 logical CPUs. If you have more than 8 logical processors, don't add more unless you observe you have contention.

Here's the code to create one additional TempDB data file – you can modify this for more files:

```
USE [master];
GO
ALTER DATABASE [tempdb] ADD FILE (NAME = N'tempdev2', FILENAME =
N'T:\MSSQL\DATA\tempdev2.ndf' , SIZE = ?GB , FILEGROWTH = 0)
GO
```

The data file creation should only take a couple of seconds – if it takes more than ten seconds, then instant file initialization isn't configured correctly. We talked about this back in the pre-installation checklist, so go back and revisit that before you create the next TempDB file. Fix the security to allow for instant file initialization now – it has a huge performance impact on database growth.

Assuming that one file growth only took a couple of seconds, then go ahead and create the rest of the TempDB data files.

Don't rely on auto-growth for TempDB files

Notice that we don't have filegrowth enabled. You want to proactively create the TempDB files at their full sizes to avoid drive fragmentation. If you have a dual-CPU quad-core server (8 cores total) and a 90GB array for TempDB data, you would create eight 10GB files for TempDB. That way, each file is contiguous, all laid out in one big chunk.

If you create them as smaller files and let them autogrow, then the disk will be fragmented all over the place because the files will be growing at random times. Plus, you could end up with differently sized TempDB files if one of them happened to grow faster than the rest. That's why we pre-grow all of the TempDB files ahead of time and get them at exactly the right size.

6.6. Configure SQL Server Max Degree of Parallelism

- Rule of thumb: set this to the number of physical cores in a single NUMA node (processor) socket on your hardware OR LESS.

- Always use an even value
- Only use the value of 1 (this is the exception to the "even" value) if you have specific vendor requirements to disable parallelism, like with Microsoft SharePoint

For more details, check out knowledge base article 2806535.

6.7. Configure SQL Server Cost Threshold for Parallelism

The optimizer uses that cost threshold to figure out when it should start evaluating plans that can use multiple threads. Although there's no right or wrong number, 5 is a really low setting. It's appropriate for purely OLTP applications, but as soon as you add a modicum of complexity – ka-boom!

If you're moving an existing system to new hardware, leave this setting alone for now. Otherwise, set Cost Threshold for Parallelism to **50**.

6.8. Configure SQL Server Max Memory

By default, we suggest leaving 10% free, or 4GB, whichever is greater. For example, if you have 256GB physical memory, you would set max memory at 230GB to leave 26GB free. Sounds like a lot, right? It gets worse.

Extended stored procedures, sp_OA calls, and linked server providers still use memory outside of the max memory number. You can learn more about those grimy details in KB 2663912.

If you're using SQL Server Integration Services, Analysis Services, Reporting Services, or any other applications on this server, you may need to lower max memory even farther.

For SQL Server 2008R2 and prior, if your database server uses CLR extended stored procedures, you may need to leave even more free. Prior to 2012, CLR memory didn't come out of the max server memory number.

After you go live, monitor the Memory:Available MB performance counter to see how much is really free. Trend it over time, and after 30-60 days, if you're sure there's plenty of available memory left on the table, then you can gradually up SQL Server's max memory setting to use the rest.

Related note: this is why you shouldn't remote desktop into SQL Servers and run processes. This is the most expensive memory you have. If you frequently need to remote desktop into somewhere to run tools, build a virtual machine with the tools you need.

6.9. Set the default database path

Tell SQL Server where it should put new data and log files by default.

6.10. Tweak settings on the model database

Set the recovery model and filegrowth settings to what you would like to be the default for new databases. We recommend moderately small fixed growth units (not percentage units), such as:

- 256MB for data files
- 128MB for log files

6.11. Set up maintenance

Configure and schedule regular maintenance for all of the following.

- Full (and possibly differential) backups
- Log backups (for databases in Full recovery model)
- CheckDB
- Index maintenance

Don't forget that your system databases need backup and CheckDB also!

Options for setting up maintenance:

A. Use free scripts from Ola Hallengren to create customized SQL Server Agent Jobs: http://ola.hallengren.com/

B. Use SQL Server Maintenance Plans if you'd like a graphical user interface, but just keep in mind that they're not as flexible and powerful as Ola's scripts. For example, they take a shotgun approach to index maintenance – they'll rebuild all of the indexes, every time, whether there's any fragmentation or not.

Learn more about configuring maintenance:

- Transaction Log Backup Frequency: http://www.brentozar.com/archive/2014/02/back-transaction-logs-every-minute-yes-really
- Why to avoid the Update Statistics Task in Maintenance Plans - http://www.brentozar.com/archive/2014/01/update-statistics-the-secret-io-explosion/
- Rebuild or Reorganize: Index Maintenance - http://www.brentozar.com/archive/2013/09/index-maintenance-sql-server-rebuild-reorganize/
- Why Index Fragmentation Doesn't Matter - http://www.brentozar.com/archive/2013/09/why-index-fragmentation-doesnt-matter-video/

6.12. Performance test the maintenance jobs

Restore a full sized copy of your production database. Time how long this takes – it's your one chance to do it before going live – and track the length of time it takes. That way, when the CIO wants to know how long it'll take before the production server is back

up and running, you won't have to guess – you'll know exactly how long a restore took when the server went live.

Run all of your maintenance jobs (backups, CheckDB, reindexing) and make sure that the jobs complete successfully and you get the backup files you expect.

7. Configure security and authentication

7.1. Test SQL Authentication (if you're using it)

If you've created/migrated any accounts using SQL authentication, make sure they work. The instance might be in Windows authentication mode only-- you can still create accounts but they won't work.

Change SQL Server to mixed mode authentication if needed, restart it, and test to make sure the access works.

(Note: SQL authentication is less secure. But you already knew that.)

7.2. Configure linked servers and security if needed.

Linked servers can be a security and performance nightmare, so be careful out there.

7.3. Configure DTC Security for Distributed Queries

Only do this step if you use distributed queries.

- We don't recommend distributed queries for performance reasons, but if you use them you may have to configure DTC Security at the Windows level
- For more information: http://technet.microsoft.com/en-us/library/cc731495.aspx

7.4. Enable remote access to the Dedicated Admin Connection.

The DAC is a separate CPU scheduler that you can use to run diagnostic queries when SQL Server seems locked up or unresponsive. By default, it's enabled – but only for local connections, meaning you have to remote desktop into the SQL Server to use it. Instead, enable it for remote troubleshooting: http://www.brentozar.com/go/dac

8. Configure SQL Server alerting and monitoring

8.1. Configure Database Mail

Configure database mail using the wizard – sadly, there's not a good T-SQL coverage area for this just yet, and the wizard is the easiest way to do it:

http://technet.microsoft.com/en-us/library/hh245116.aspx

Test it using the "send test email" functionality, and send an email to the distribution list you use for SQL Server alert notifications. Never send emails directly to individuals – sooner or later, as hard as this is to conceive, they are going to take a vacation. (More likely, they're going to quit their jobs, but we're trying to be optimistic here.)

Be aware that developers can use Database Mail for things that SQL Server shouldn't be doing. For example, they may decide to use Database Mail to send out mass emails to your end users or customers. There's nothing technically wrong with that, but it increases the load on the database server and it sends all outgoing email with the SQL Server's Database Mail account.

If you choose to allow this, consider setting up separate private and public email profiles. The public email profile used by the developers should be sent from the developer management team's group email address – that way, they can address any replies themselves.

8.2. Create Operators in the SQL Server Agent

You may need more than one.

8.3. Configure Database Mail in the SQL Server Agent properties

Go into the SQL Server Agent properties, enable database mail, and select the right profile.

Configure a failsafe operator while you're in there. During the setup process, this has been known to restart Agent, so you want to do this before you go live.

8.4. Create alerts for high severity and data corruption

Use our handy script to create alerts. Just change the operator name to your operator – the one that points to your distribution list. http://www.brentozar.com/blitz/configure-sql-server-alerts/

8.5. Restart the Agent Service and Test

Restart the SQL Server Agent Service after to make the change you mail to enable the mail profile take effect. (This isn't the whole SQL Server, just the Agent service)

After creating the alerts, test them and make sure the operator you configured receives the notification. You can do this by running a command like:

```
RAISERROR('Alert Test', 18,1) WITH LOG;
```

8.6. Set Up Monitoring

You've already set up SQL Server alerts, but also configure monitoring.

At minimum you want an up/down monitor that remotely checks and makes sure your SQL Server service and SQL Agent service are running. You also want an enterprise monitoring tool looking at your Windows Application and System log to find errors.

Configure performance counter collection of critical counters. The counter list we like to use is at http://BrentOzar.com/go/perfmon

9. Install tools and run a health check

9.1. Install and run sp_Blitz®

Download the free script from http://BrentOzar.com/Blitz

Install and run sp_Blitz®. Use the @CheckServerInfo parameter if you'd like to also capture your current server configuration and store it off for posterity:

```
exec sp_Blitz @CheckServerInfo=1
```

Review all sp_Blitz® output and use the URLs to learn more about the issues. Correct all the issues you find.

9.2. Install sp_WhoIsActive

sp_WhoIsActive is a free tool written by Adam Machanic. It can be very useful to have to see current activity on a SQL Server. For info on how to install and use this tool, go to http://BrentOzar.com/go/active

10. Deploy custom code & settings for your app

10.1. Migrate App Specific Logins, Agent Jobs, and Custom Alerts

You also need to configure all of the following for your application, typically copied from your current production server:

- Application logins. If migrating from another server, you can migrate the logins: http://support.microsoft.com/kb/918992
- Application specific SQL Agent jobs (can be scripted from an existing instance)
- Custom alerts (can be scripted from an existing instance)
- Identify and clean up Orphan Users - http://technet.microsoft.com/en-us/library/ms175475.aspx

10.2. DTS Packages and SSIS

Migrate legacy DTS packages or SSIS Packages if needed. We really hope you don't need to and that you're using a centralized SSIS instance if you're into that kind of thing.

10.3. CLR Components

Make a plan if you're migrating databases that use CLR components to configure and smoketest those components after migration.

- Enable CLR components in SQL Server http://technet.microsoft.com/en-us/library/ms131048.aspx
- Depending on whether the assembly is signed by a certificate or is unsafe/external access, you may need to use additional steps including creating certificates, setting the database trustworthy property, and changing the owner of the database.
- Test CLR assembly functionality on a restored copy of a database prior to migration.

11. Learn more about SQL Server

sp_Blitz®: Free SQL Server Health Check

You've inherited a SQL Server from somebody, and you have no idea why it's slow. sp_Blitz® gives you a prioritized list of health and performance issues, plus gives you URLs for more details about each issue. http://www.brentozar.com/blitz/

Our Free 6-Month DBA Training Plan

Every Wednesday, you get an email with our favorite free SQL Server training resources. We start at backups and work our way up to performance tuning. http://www.brentozar.com/needs/

SQL Critical Care®

Don't have time to learn the hard way? We're here to help with our quick, easy 4 day process that gets to the root cause of your database health and performance pains.

Contact us for a free 30-minute discussion: BrentOzar.com/contact.