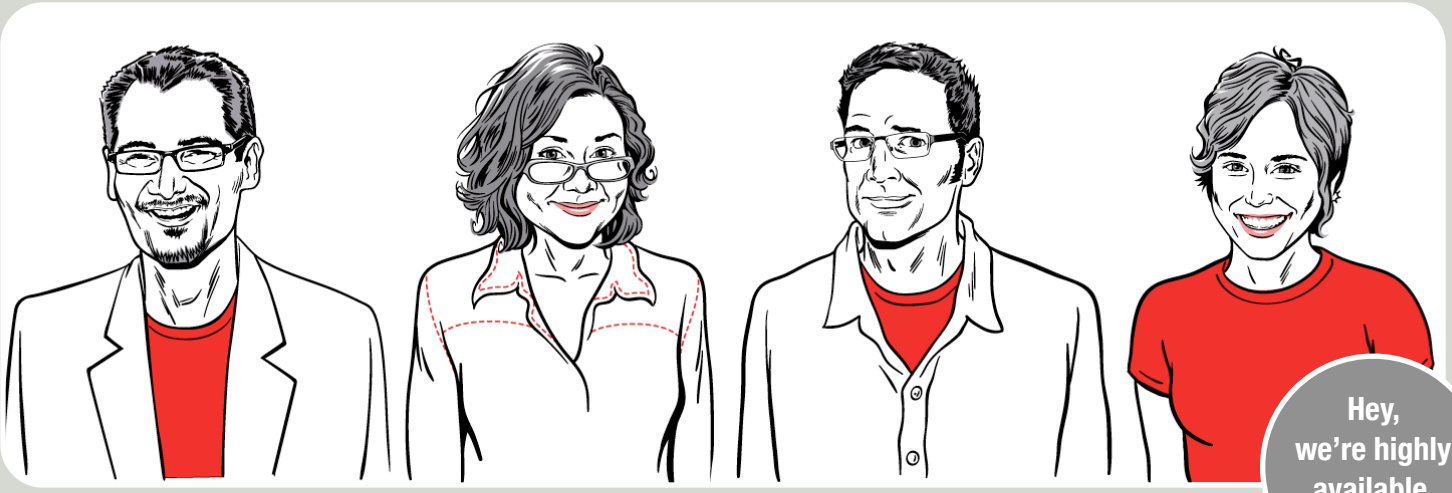
From BrentOzar.com/go/alwayson, it's our

SQL Server 2012 AlwaysOn Availability Groups Checklist



BEFORE YOU RUN THE WIZARD...

The power of Microsoft SQL Server 2012's new AlwaysOn Availability Groups can give you both high availability and disaster recovery, but with power comes...well, complex configuration. We'll help you get started.

Our clients have deployed SQL Server 2012 to get the new reliability features. Instead of using a complex combination of clustering, mirroring, log shipping, replication, and other technologies, you can get both high availability and disaster recovery in a single tool.

That's the good news.

The more challenging news is that AlwaysOn Availability Groups are brand spankin' new in SQL 2012, and we're still getting the kinks ironed out. The setup isn't straightforward, and we have to consider a whole lot of gotchas long before we kick off the wizard.

This checklist is very much a work in progress. We'll be the first to admit that it's just a launching pad for discussion; some lines are just a minute's worth of work, and others represent hours of planning and labor. This checklist isn't a substitute for experience with Windows clustering, networking, SQL Server backups, and the other technologies that AlwaysOn requires. In this early version of the checklist, the target reader should already have experience with those technologies.

This version is as of March 2013. If you're reading this in fall 2013 or later, go download the latest version for free.

We're Here to Help

If you're overwhelmed, email us at Help@BrentOzar.com. We offer consulting and implementation services to relieve SQL Server, SAN, VMware, and AWS pain.

WHO WE ARE

Brent Ozar, Jeremiah Peschka, Kendra Little, and Jes Schultz Borland.

Big-Picture Planning

Discuss the below issues with the business users before you get started so that the solution we build matches their requirements.

1. Choose the number & location of replicas. For each replica, decide:
   - 1.1. Synchronous or asynchronous replication? We get up to three synchronous replicas (a primary and two secondaries). The rest of the replicas (up to four replicas in total) will be asynchronous.
   - 1.2. Automatic failover to this node? We only get two automatic failover servers - a primary and a secondary.
   - 1.3. Whether you want to pay for your licensing. Just kidding - you're paying.
2. Choose a quorum model based on your chosen number of replicas. This decision is driven by an even versus odd number of nodes, how many nodes need to be up for the cluster to stay online, and which nodes should vote. For an introduction, see http://technet.microsoft.com/en-us/library/cc770620%28v=ws.10%29.
3. Design your Availability Group granularity.
   - 3.1. How many databases do you want in each group?
   - 3.2. Do you want to move groups around for load balancing?
   - 3.3. Do you need to separate reporting databases from the OLTP databases so the report data can be built on a read-only replica of production OLTP?
   - 3.4. If just one database on an instance fails, what do you want to happen?
   - 3.5. Are there some critical databases that need to be in their own AG with synchronous replication & automatic failover, while others can be async with manual failover?
4. Install SSMS 2012 on the DBA and developer workstations. The SQL Server 2012 tools don't work on Windows XP, so if you're still in the dark ages, you'll need a server you can remote desktop into for the admin tools.
5. If any other servers require restores from your new 2012 instance, they should get 2012 first. (Think development, disaster recovery, or reporting servers.)
6. If you're transitioning from an existing SQL Server, review non-database stuff installed on those servers. Plan for SSIS, SSAS, SSRS, DTS packages, logins, Agent jobs, etc.

Pre-Installation Work for Windows

These tasks should be done before you even think about running the Windows installation. Unlike a typical SQL Server environment, AlwaysOn work requires a lot of coordination between DBAs, Active Directory admins, network admins, and sysadmins.

7. Create an Active Directory account for the SQL Server service. This will be used on all nodes in the cluster. (You can use different accounts per node if you're sure you'll never use Kerberos authentication, but we still don't recommend this.)
8. Choose standard drive letters because the SQL Server data/log file locations must be exactly the same across all nodes. The C drive is obviously standard for the system, but you won't be installing SQL there. Please. Pretty please.
9. Choose a static IP address for the AlwaysOn Availability Group Listener in each subnet where the AG might fail over to. If you've got different subnets in production & disaster recovery, get an IP for each one. You can technically do this with DHCP, but we'd only recommend that for lab environments.

10. If a shared storage failover cluster instance (FCI) will be involved, things get more complex. We won't touch on an entire cluster setup here, but these two are critical:
    10.1. Reserve IPs for the cluster management, SQL Server instance, and DTC.
    10.2. Decide how to configure DTC on the cluster. Cindy Gross has a great post on this: https://blogs.msdn.com/b/cindygross/archive/2009/02/22/how-to-configure-dtc-for-sql-server-in-a-windows-2008-cluster.aspx?Redirected=true
11. Decide how you'll achieve network redundancy. You no longer need a separate heartbeat network, but we'd recommend against using just a single network card in a cluster environment. In each node, either use multiple network cards with individual IP addresses, or use multiple network cards with software teaming and a single IP address. Cluster validation will throw a warning if there's only one active network card detected (teams are considered a single card), but it'll let you install anyway. Plan for this redundancy before you install SQL.
12. Decide where backups will be performed, and the target location. Choose a preferred and a secondary server to run the full and transaction log backups - the secondary will be used if the preferred one is down. Make this choice for both the primary and DR data centers, too - you need to know where backups will run if we fail over to the DR data center.
13. Decide whether to use Windows Core. In Windows 2008R2, it's all or nothing: every node in the cluster must be configured the same way. We don't recommend this for Win2008R2 unless you're extremely comfortable troubleshooting via PowerShell and cluster.exe during an outage. Thankfully, Win2012 allows mixed cluster nodes and let us add/remove the GUI on the fly.
14. Get the right installation bits for SQL Server 2012 (the key is embedded) and download the latest cumulative update (CU). A good guide for the latest builds is http://SQLServerBuilds.blogspot.com - it's usually up to date.
15. If you'll be installing the Availability Group without AD admin privileges, consider pre-staging the cluster name: http://technet.microsoft.com/en-us/library/cc731002%28WS.10%29.aspx#BKMK_steps_precreating

Windows Installation

After the Windows installation, this section's steps don't have to be done in exact order. For example, you may run CPU-Z to check power saving after you've enabled Instant File Initialization or joined the domain. All steps must be done on every node, though.

16. Install Windows Server on each node. If you're using Windows Server 2008R2, clustering is only included in Enterprise Edition. (Your company may have a standard install checklist like configuring antivirus exclusions and Windows firewall.)
17. Join all of the nodes to the same domain. Clustering requires the nodes to be in the same domain, even if they're in a different data center. They can be in different subnets, though, and that's not as painful as it used to be with Windows 2003.
18. Apply all Windows updates, but make sure Windows isn't set to automatically apply updates as they come in. You don't want your cluster restarting on its own.
19. Install the Windows recommended hotfixes for clustering. For Windows 2008R2 SP1: http://support.microsoft.com/kb/2545685 For Windows 2012: http://support.microsoft.com/kb/2784261

SAMPLE A.G. DESIGNS

Client X: Easy High Availability

One of our clients is a high-traffic web site that needed easy failover between their primary and disaster recovery data center. Despite serving millions of pages per day from hundreds of databases, they don't have a full time database administrator. They wanted an easy way to move databases around without changing the app. AlwaysOn Availability Groups delivered.

Client Y: Scale-Out Reads

Another client was frustrated because they were paying for hardware in two data centers, but one data center was always sitting around idle doing nothing. By implementing AlwaysOn Availability Groups, we were able to leverage the disaster recovery data center for near-real-time reports directly from the production database schema. Instead of waiting for nightly data warehouse loads, they could get the data they wanted faster - without spending more on licensing or ETL processes. They could use all the reports they'd already written to go against the DR copy of production - without the pains of querying the production server.

20. Install Windows prerequisite hotfixes for Availability Groups - http://msdn.microsoft.com/en-us/library/ff878487.aspx#WinHotfixes
21. If you're using Windows 2008R2, install the hotfix to allow nodes with 0 votes - http://support.microsoft.com/kb/2494036 - We're listing this separately because it's really important that it's on all nodes.
22. Open Windows firewall ports 1433 and 5022 on all nodes.
23. Do a quick performance measurement:
    - 23.1. Run CPU-Z from http://www.CPUID.com on each node to verify that power saving is disabled.
    - 23.2. Run CrystalDiskMark from http://crystalmark.info/software/CrystalDiskMark/index-e.html/ (making sure you're getting CrystalDiskMark, not CrystalDiskInfo) on each node on each RAID array to get a sanity check on throughput. Do 5 test passes, 4000MB test file. After each test finishes, click Edit, Copy and put the text results in files organized by node name.
24. Enable Instant File Initialization. On each node, go into secpol.msc, Local Policies, User Rights Assignment, Perform Volume Maintenance Tasks, and add the AD service account you created in the Planning step.

Cluster Installation

We'd recommend finishing all of the above steps before setting up clustering. Once the cluster has passed validation, we're not too fond of tweaking its configuration.

25. In Failover Cluster Manager, validate the proposed cluster config. You can ignore most of the storage warnings if you're not using shared storage clustering (FCI), but don't ignore the network errors about redundancy. Network connectivity is extremely important for AlwaysOn Availability Groups. Save the cluster validation report to disk to cover your rear later (especially if the Windows installation is being done by a separate team.)
26. Create the cluster. It's a link in the last step in the validation wizard.
27. If you want to override the automatic quorum configuration, now is the time to remove votes - http://msdn.microsoft.com/en-us/library/hh270281.aspx
28. Document what will happen if various nodes go offline and quorum goes down. All teams needs to understand the ramifications of rebooting machines when patching.
29. If an FCI is involved, configure DTC per your Planning section decisions.

SQL Server Installation

This section assumes we're not doing a shared storage failover cluster instance (FCI). FCIs add a lot of complexity that we're not covering in this short (relatively!) checklist.

30. In Failover Cluster Manager, validate the cluster and save the cluster validation report to disk. We do this again here because sometimes the Windows cluster and SQL cluster are done by different teams, and we need to all be completely comfortable with any validation warnings before & after installation. If you're not comfortable with the warnings, stop here and get all teams involved. You only get one chance to do this right.
31. Install a standalone instance of SQL Server on each node.
    - 31.1. Install a default instance, not a named instance. (Not a requirement, just making administration easier.)
    - 31.2. Use the AD service account you created during the Planning section.

31.3. Use the same collation on all nodes.
31.4. For the data & log file directories, choose the drive letters & path you picked during the Planning section. The default will be a long X:\MSSQL.SOMEPATH.SQLSERVER\MSSQL\DATA - consider just doing X:\MSSQL\DATA, but don't use the root directory outright like X:\.

32. Install the latest SQL Server cumulative update that you downloaded earlier during the planning steps.
33. Enable AlwaysOn HADR - on each node, go into SQL Server Configuration Manager, Services, SQL Server Service, and on the AlwaysOn tab, check the box for Enable HADR. Click OK, and restart the SQL Server instance.
34. Follow our setup best practices checklist at http://www.BrentOzar.com/go/setup for things like memory and TempDB configuration.

Availability Groups Proof-of-Concept Setup

Before you go live with your production databases on this setup, build a dummy Availability Group that you can use to play with the cluster and see how things work. This work will mostly be done on the primary node. You can change which node is the primary later, but these instructions will just refer to a single instance of the cluster.

35. Create at least one user database or restore one from backup. It doesn't have to be the production data - we're just going to set up an AG, listener, backups, etc so you can understand how this stuff will work, and you're going to have to tear it down anyway when you migrate existing databases here via mirroring or log shipping.
36. If you created the user database from scratch, perform a full backup of it. You can throw it away, but this step has to be done for the wizard to work. (The wizard does a copy-only full backup, which doesn't initialize the log chain.)
37. Start the wizard. In SSMS, go into AlwaysOn High Availability, right-click on Availability Groups, click New AlwaysOn Availability Group. Step through the wizard, but only set up one replica initially. Setting these up is risky as all get out - we've had more failures than successes - and we want to isolate the work we're doing to one node at a time.
38. Add additional replicas one by one by going into SSMS, AlwaysOn High Availability, expand your newly created Availability Group, and right-click Replicas to Add a Replica.
39. Configure which replicas will be available for failover, synch/asynch replication, and readable connections. (Note that if it's read-only intent, you even have to specify that in SSMS connection options.) This is done by right-clicking on the Availability Group name in SSMS.
40. After all nodes & databases are online, configure read-only routing. There's no GUI for this - only T-SQL. http://msdn.microsoft.com/en-us/library/hh710054.aspx
41. Test read-only routing with SQLCMD -K ReadOnly.
42. Configure the preferred backup servers according to the Planning section decisions.
    42.1. Configure the preferred servers in SSMS - http://msdn.microsoft.com/en-us/library/hh710053.aspx#SSMSProcedure
    42.2. Configure the backup scripts - maintenance plans and http://Ola.Hallengren.com expect the jobs to be run on all nodes, and they'll automatically use sys.fn_hadr_backup_is_preferred_replica to decide on a database-by-database level whether they're the preferred backup server, and if so, back up that database. Test this.
    42.3. Configure index maintenance script jobs on all nodes, but only enable the schedule on the primary node. We don't currently have an easy way to only run this on the primary node - failing over will involve manually enabling these jobs (as do any SQL Agent jobs that hit user databases). This could also be done with Windows Task Scheduler, which will automatically fail over to the primary node.

Planning for Failovers

AlwaysOn Availability Groups have a few dependencies:

1. The Windows cluster must be online. This requires enough online voting nodes to achieve a quorum, or you must force the cluster online and reallocate votes. To make sure it's online, ping the cluster's administration name - not the SQL Server's virtual network name or the listener name, but the cluster name. You can find this in Failover Cluster Manager - but you'll want to do this ahead of time.
2. The Windows node you want to connect to has to be online. Not all of them must be online, but if you want to connect to a specific node, it has to be online.
3. Each SQL Server instance has to be up. Sounds obvious, but just because the cluster's up doesn't mean SQL's up. If it's a failover cluster instance (FCI), this may involve additional troubleshooting.
4. The primary replica has to be up (or you have to fail over to a new primary). If the primary isn't up, the listener may not work. You can still get read-only queries by connecting directly to one of the replicas, though, so that may be helpful in disaster scenarios.

5. The primary replica has to be able to bring the listener online. The listener is the network name you'll normally be connecting when running queries.

Each of these can have its own dependencies, too. For example, a failover clustered instance requires the ability to bring its IP addresses, name, and shared drives online before it starts SQL Server. This document doesn't go into the details of troubleshooting each of those parts, but we need to mention it here because you should troubleshoot each part as the cluster comes online.

Performing a Planned Failover

The difference between a planned failover and an unplanned failover is that during a planned failover, a lot of our dependencies are still online. The cluster's online with all voting members able to talk to each other, and the primary replica is still accepting queries. In these cases, a failover is pretty easy. Here's how to do it with the GUI:

1. Paranoid Pre-Planning: if you haven't done this in a while and you want more confidence that things will work as planned, go into Failover Cluster Manager and validate the cluster. If someone's been monkeying around with the hardware, networking, storage, or OS, you might catch issues here before they go horribly wrong.
2. Connect to the primary replica with SSMS.
3. In Object Explorer, go into AlwaysOn, Availability Groups, right-click on the availability group, and click Properties.
4. Make sure that both the primary server and the soon-to-be-primary server both have Synchronous replication, not Asynchronous. (If not, change them to both be synch and watch the AlwaysOn dashboard until they catch up.)
5. Right-click on the availability group and click Failover to start the wizard. Choose the replica you'd like to fail over to, and make sure it doesn't have warnings.
6. After the failover completes, note any warnings (especially quorum warnings). Verify that you can connect to the listener.
7. If you're using read-only routing so that clients can connect to replicas for read-only queries, consider checking those replicas with tools like sp_who or DMVs to see if the replicas are getting queries.
8. Review your SQL Agent jobs on all servers to make sure the right jobs are running on the right servers. If you've hard-coded your backup, index maintenance, or user database jobs to run on specific replicas, you may have to revisit those jobs. If you're doing application logic (like repopulating report tables) in a SQL Agent job, you might consider running those on a central job server instead, and point the jobs at the listener name. Another option is to use Windows Task Scheduler because those tasks can fail around with the cluster name.
9. If you're doing disaster recovery tests, check additional steps in the unplanned failover section below. Your company may want to do things like move quorum votes over to the disaster recovery site in preparation for disconnecting the primary data center altogether.

Reacting to an Unplanned Failover

If the cluster or the SQL Server instance went bump in the night without the courtesy of calling you beforehand, then we've got additional work to do.

How much data will we lose? If you're failing over to an asynchronous replica, you're probably going to lose data. Build a set of queries that show exactly how old the replica's data is - but here's the catch: you can't rely on being able to query the primary replica. Instead, look at timestamp-based tables. For example, if you've got a Sales table with a SalesDateTime field, run this query on the replica:

```
SELECT TOP 10 * FROM dbo.Sales
ORDER BY SalesDateTime DESC
```

If the newest sales are 20 minutes old, you can tell management that they stand to lose up to 20 minutes of data if we fail over now. You may still be able to recover the data from the primary - more on that later. The amount of potential data loss helps you with the rest of the decisions you're

If only SQL Server had this sign

about to make.

How long will we troubleshoot? If we stand to lose 20 minutes of data, then it helps to be able to tell an executive, "We can either go live with our DR replica right now and lose 20 minutes of data, or I can spend 30 minutes troubleshooting to find out how bad the situation is. What would you like to do?" Put that decision in management hands as fast as you can - preferably after you've done just five minutes of troubleshooting. Within five minutes, you should know if it's an easy problem or a challenging one, and you want to let management decide when the outage should end.

Who's allowed to make the go/no-go decision? If the DBA, dev manager, or IT manager aren't around, who's responsible for the call to fail over to the asynch replica and lose data? Ideally, we've got as large of a list as possible here so that the company can react fast without waiting around for a long phone tree. This list of people must be firmed up ahead of time.

Who's allowed to perform the failover? AlwaysOn Availability Group failover can be scripted out with PowerShell and T-SQL, and these scripts can be given to a 24x7 help desk rotation. I know, as a DBA, it can be scary to hand out this kind of permissions, but I'd rather empower someone else - when armed with a manager who can make go/no-go decisions - so that the company can be back up and running faster.

Once failover starts, who's got quorum votes? If groups of servers, or heaven forbid, an entire data center is offline, failover starts to get more complex. The team will need to force the quorum online without enough votes, and then reassign voting rights using PowerShell or cluster.exe. We'll probably need to change the quorum model as well, perhaps going from node majority to node-and-file-share majority. These are complex topics that should be thought-through ahead of time rather than busting out Books Online.

Do jobs need to be enabled/disabled? If we've turned a secondary replica into a primary, how does this affect our SQL Server Agent jobs that were running on the old replica? Do we need to change any of our backup jobs? In theory, the built-in backup preferences settings for AlwaysOn Availability Groups will cover this, but complex replica scenarios may have multiple backup jobs to maintain both onsite and offsite backups. We might also have aggregation jobs or ISV jobs that must always run on the primary, and we need to ensure those get enabled.

Who will troubleshoot connectivity problems? When there's been a massive failure, I like having an open conference bridge with all hands on deck - but not necessarily all hands should focus on the call. For example, in multi-DBA teams, I prefer to designate just one DBA team member as the primary point of contact on the call. They handle all questions from developers and end users while the other DBAs focus on making sure the replica stays healthy.

What do we do if we have multiple primary replicas? In a split-brain cluster scenario, we can have two database servers that both think they're the primary replica for the same database. We need a written checklist that we can give to someone who will drive to the not-supposed-to-be-primary data center, shut the servers down, and stay there until the network comes back online to bring the server up gracefully.

Will we try to recover the lost data? When the formerly-primary server comes back online, the databases will still be readable. We can use data comparison tools to compare the two replicas and generate insert/update/delete scripts to bring the data back into sync - in theory. In practice, identity fields can make this impractical or impossible. I'd start by comparing timestamps in a few key tables just to see how much data is at stake, and then give management a rough time estimate on what it'd take to recover that data. In order to give a good estimate, you'll want to try this approach ahead of time (long before an outage) using a pair of development or QA database servers. Get a feeling for how the tables are related and how much work would be involved to sync them.

There's a lot of questions here. When your team is armed with the answers ahead of time, it'll make failovers much less painful, and you'll look like the smartest, fastest-reacting ninjas in the company.

Clusters, Failures, and GUIs

In theory, in theory and in practice are the same.
In practice, they are different.

The AlwaysOn features (both failover clustering and availability groups) are very complex machines with a lot of moving parts. The GUI is pretty good, but it's a good idea to understand the underlying machinery because it doesn't handle all of the moving parts. If everything's working perfectly, then so will your failover - but then you wouldn't need a failover if things were working perfectly, now, would you?

In this guide, we only address doing things in the GUI, but if you want to be able to react quickly and smoothly during a failure, you'd be wise to learn the PowerShell and T-SQL commands involved with these activities. Script out your actions ahead of time so you'll be able to troubleshoot faster.

WE'RE HERE TO HELP.

If you need high performance, highly available SQL Servers, here's what we do.

SQL Server Migration Advice Plan

You need to choose the right hardware and infrastructure setup for your new SQL Server. You're not sure where your money is best invested - SSDs, SAN space, more memory, Enterprise Edition? You're also not confident that you're using the right features for your high availability and disaster recovery needs. Your staff can do the implementation work, but they just need guidance through the decisions, planning, testing, and go-live day.

We help you navigate the tough choices and tell you what's really working out in the field for our clients. We can introduce you to other businesses facing similar challenges and help you onto the right path.

We use a proven process covering planning to implementation in six weeks. We meet with your staff for four hours per week, guiding and assisting them. For less than a single core cost for Enterprise Edition licensing, you get personalized help from Microsoft MVPs and MCMs.

SQL Server Critical Care™

Your data is a critical part of your business. As your business continues to succeed, the demands on your database will continue to increase. The increasing demands of a growing database environment can lead to unexpected performance changes like poor performance or outages. Without a clear performance baseline and plan to deal with the complexities of your SQL Server infrastructure, your team risks being left in the dark when issues arise. You may even miss opportunities to cut costs and increase performance.

Brent Ozar Unlimited specializes in database check-ups and performance tuning. We work directly with your team to assess the overall performance of your environment. We'll dig in to any issues that we uncover, provide explanations, and offer strategies to mitigate or remove the issue.

During the check up, we'll establish a performance baseline and set up a regular maintenance schedule. Regular maintenance will keep your database in great shape and will make it even easier to implement performance improvements. With a reliable baseline in place, you can track your performance improvements over time. As part of the check up, we also identify opportunities for database and server consolidation to cut costs.

Once the check up is complete, you will have a performance baseline, an action plan, and a set of techniques to help you measure and maintain database performance.

VMware, SAN, Cloud Expertise Too

We regularly deploy production servers on VMware, shared storage, and Amazon Web Services. Get our advice so you don't have to learn the hard way.

Sound Good? Let's Work Together.

Email us at Help@BrentOzar.com and we'll get started with a free 30-minute call. We partner with you to objectively identify pain points and develop remedies that align to your business goals. Your experience comes first; we share our knowledge and expertise to help you.

BRENT OZAR UNLIMITED
Twitter: @BrentOzarULTD
Web: BrentOzar.com
Email: Help@BrentOzar.com
Phone: 773-420-9626