COHESITY

Version 1.3

July 2024

Protect SQL Server with Cohesity — Deployment Considerations Guide

Recommended Configurations for Deploying Cohesity Protection for SQL Servers

ABSTRACT

As SQL Servers have become critical components of growing enterprise infrastructure, it has become even more important to protect those databases efficiently and reliably. This guide offers practical recommendations for configuring Cohesity protection for SQL Servers, and focuses on the four most common SQL Server setups: Standalone, Failover Cluster Instance (FCI), Virtual Machine (VM), and SQL Native. For each, you will find descriptions, technical recommendations, and notes describing the common mistakes to avoid.

Table of Contents

Introduction 5
Audience 5
Approach to Backups 5
Deployment Workflow 5
Terminology 6
Backup Strategy Considerations 7
Review Disaster Recovery (DR) Considerations 9
Take Local Snapshots 9
Replicate Backups Off-Site 9
Archive Backups to the Cloud 9
Validate SQL Backups 10
Check Cohesity Cluster Settings 11
Deduplication 11
Compression 11
Encryption 11
Fine-Tune MS SQL Server Parameters 13
Host CPU 13
Host Memory 13
Host Windows Boot Drive (C:\) 14
SQL Server Database settings 14
Link Active Directory 15
Select Accounts and Grant Permissions 16
Installation Account 16
Choose the Service Account 16
Pros and Cons of Service Account Types 17
Assign Service Account Permissions 19
Get the Synergy of SQL Server VSS Writer 19
Install the Cohesity Agent for Windows 21

Choose Agent Methods 21
Consider File-Based Backups *22*
Consider Volume-Based Backups *22*
Consider VDI-Based Backups *22*
Compare Volume, File, and VDI-Based Methods *23*
Install and Manage the Cohesity Agent for Windows 26
Set Up the Cohesity Agent 26
Know SQL Database Types and Source Registration 27
Identify the Type of Database 27
Register the Database Properly 29
Deploy for Specific SQL Server Features 30
Set Up for a Standalone Server (Physical or VM) 30
Arrange for MS SQL Server Always On Availability Groups (AAG) 30
Position for SQL Cluster 31
Strategize for a SQL VM 31
Consider Non-Agent-Based Backups 33
SQL Native Backups 33
Your Feedback 34
About the Authors 34
Document Version History 34

Figures

Figure 1: Cohesity Agent Installation Options 26

Tables

Table 1: Cohesity Terminology 6
Table 2: Prerequisite Planning for Backups 7
Table 3: Comparison of Service Account Types 16

Table 4: Service Account Types 18
Table 5: Service Account Permissions Minimums 19
Table 6: Agent Methods for Backup and Restore 21
Table 7: Comparison of Volume, File, and VDI backup Methods 23
Table 8: MS SQL Server Database Types 27
Table 9: Database Types by Source Registration 29

Introduction

"Successfully restoring a SQL database means you've protected the data; successfully recovering a server means you've protected the machine; successfully recovering after a disaster means you've protected the business."

A good Cohesity deployment allows the business to securely protect its data and infrastructure. It positions the business to take advantage of evolving technologies like cloud services.

You are at the *Deployment considerations* phase of implementing the Cohesity solution for your SQL Server databases. This guide focuses on the deployment and configuration choices you have to make as you set up Cohesity for your SQL Servers, *prior* to deploying the platform to protect your specific databases.

Audience

This guide is for IT administrators and database administrators (DBAs) who are familiar with managing data protection for Microsoft SQL Servers.

Cohesity recommends having familiarity with:

- Microsoft SQL Server
- Microsoft SQL Server Failover Cluster Instance (FCI)
- Microsoft SQL Server Always On Availability Groups (AAG)

Approach to Backups

SQL backups must fit into the customer's requirements. Backups must meet Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO). In addition to these, the backup approach must determine *which* objects to protect and *how long* to retain them.

Deployment Workflow

SQL Server databases can be implemented in many different ways, so it's important that you carefully evaluate each of the considerations discussed in this guide, to identify the best choices for your organization's particular needs.

Armed with those choices, you will:

- Check the settings on the Cohesity cluster that will backup your SQL Servers.
- Fine-tune MS SQL Server parameters.
- Link the Cohesity cluster to the Active Directory (AD) that manages the SQL Server hosts.
- Select accounts and grant permissions.
- Install and set up the Cohesity Agent for Windows.

- Learn more about deploying for specific SQL Server features.
- Learn about using SQL Native backups instead of agent-based backups.

When you complete those tasks, see the MS SQL section in the online Help to set up protection for your databases.

Terminology

There are several concepts and terms that are important to understand as you learn about the Microsoft SQL Server Backup features in Cohesity.

Table 1: Cohesity Terminology

TERM	DEFINITION
View	Cohesity serves data to clients and hosts from logical containers called Views. Views are exposed as SMB shares. Users can specify storage efficiency, QoS policy settings, and permissions at the View level.
QoS Policy	The Quality of Service (QoS) policy specified for a Cohesity View. By choosing a QoS policy for the appropriate priority and type of workload, you enable Cohesity to optimize how it processes data in regard to latency, throughput, and priority. For details, see Create or Edit a View.
Protection Group	A Cohesity Protection Group is a backup job that runs repeatedly, based on an associated policy, to back up data from a source and store it on the Cohesity cluster.
Protection Policy	A collection of reusable settings that define how and when sources are protected, replicated and archived. You select a policy when configuring a Protection Group.
Replication	Replication automatically makes copies of Snapshots captured by a Protection Group on one Cohesity cluster and copies them to a second Cohesity cluster. A protection group specifies the SQL databases to be backed up and replicated.
SQL VIP	The IP address of the SQL Instance that moves from one physical node to another when a node fails.
Log Shipping (Technique)	The technique of providing a series of copies of transaction log backups from a primary database on a primary server instance to a secondary database on a separate secondary server instance. On the secondary database, transaction logs are applied independently and sequentially.
MS SQL Server Log Shipping	Microsoft's implementation of the Log Shipping technique. Specifically, it uses a set of stored procedures and SQL jobs to carry out the process.

TERM	DEFINITION
DB Migration (Technique)	The process of moving an MS SQL Server database from one SQL Server instance to a different SQL Server instance without data loss. This can be performed manually or in some automated form.

Backup Strategy Considerations

Data backup is an essential part of protecting the data, but it's important to really understand what makes a backup strategy successful. It is important to have a second copy of data in case the original copy fails. But that is only part of the story. A good backup strategy is obviously going to create that second copy, but it is more crucial that, when recovery is needed, the data can actually be found and restored quickly.

You can use Cohesity for other types of backups and even create new backup strategies designed to better meet your company's needs.

Cohesity recommends that you plan your SQL backups using the business's Service Level Agreement (SLA) and disaster recovery (DR) requirements.

Table 2: Prerequisite Planning for Backups

PLAN	DETAILS
Current Methods	Determine how backups are currently being done. Some common methods are: SQL Maintenance Plan, TSQL Scripting, and third-party tools. **Cohesity recommends** all other backup methods be disabled. If multiple backup methods are being used concurrently, conflicts can occur.
Strategy Planning	Determine which type of SQL backups are needed for the business. Consider the customer's SLA and review the business's Disaster Recovery Plan for RTO and RPO requirements. These will largely define the backup strategy.
Restore Planning	**Cohesity recommends** a periodic restore of a database from its backup. This is an important step in the overall backup strategy because it tests, verifies, and validates the integrity of the backup. A sample set of backups should be restored to a non-production server and evaluated. This step also ensures that at a critical event, the method of restoring a database is valid.

PLAN	DETAILS
Protecting the Backups	Determine a strategy for protecting all backups. The backups are very valuable and need to be protected from: corruption, deletion, overwriting, and catastrophic loss. Copies of the backups should be moved to an off-site location. Having off-site copies of the backups not only protects the backups, but is also the foundation for a Disaster Recovery Plan. Use replication to maintain off-site copies, in addition to the backups you archive for long-term retention. **Cohesity recommends** a periodic restore of a database from its backup. This is an important step in the overall backup strategy because it tests, verifies, and validates the integrity of the backup. A sample set of backups should be restored to a non-production server and evaluated. This step also ensures that at a critical event, the method of restoring a database is valid.

Cohesity recommends: Use the Cohesity Agent for backups, and disable all other backup jobs or methods, such as SQL Agent maintenance jobs, which would conflict with the backup process.

Review Disaster Recovery (DR) Considerations

Cohesity recommends safeguarding all of your backups by using replication and archival.

Taking a SQL backup is only one part of protecting a business. Protecting the business means protecting the data from corruption *and* from catastrophic disaster. This is done by keeping a series of backups, and then in turn moving some of those backups off-site and archiving them under a long-term retention plan.

Protecting the business starts with:

- **Capture and Store**: Protection from loss and corruption.
  Protects the databases with regularly scheduled backups.
- **Geo-Protect**: Protection from catastrophic loss.
  Protects the backups by replicating them to an off-site location.
- **Cost-effective Archive**: Protects and holds the backups.
  Protects the business by archiving the backups to the cloud under a long-term retention plan. Stored on a lower-cost storage tier.

Take Local Snapshots

Cohesity recommends maintaining enough backups to meet your business requirements.

Protect from data corruption over time by maintaining a series of local snapshots. By taking and maintaining several snapshots, you are in position to recover data from its state prior to being corrupted. Snapshots are efficient because they use deduplication and compression.

Replicate Backups Off-Site

Cohesity recommends replicating backups to a second Cohesity cluster.

Protect your entire set of SQL backups from catastrophic loss by replicating them to an off-site location. In a Cohesity protection group, you are able to automatically replicate the SQL backups stored in the Cohesity cluster to a second off-site Cohesity cluster. As part of replication, Cohesity always performs source-side compression and deduplication first and sends only the changed data over the network for cost-effective disaster recovery.

Archive Backups to the Cloud

Cohesity recommends archiving to the cloud to take advantage of low-cost, long-term storage.

Archive some of the SQL backups to the cloud as a way to address long-term data retention requirements and lower the cost of storage. Cohesity provides a policy-based method to archive to public clouds (AWS, Azure, Google Cloud Platform) and any S3-compatible storage.

Validate SQL Backups

Cohesity recommends regularly validating backups by restoring or cloning a database. Further validation of the backup can be done by running a DBCC CHECKDB (Check Database) command.

Successfully restoring a SQL backup is proof that your backup strategy works. Cohesity gives you the advantage of being able to browse and search across all your snapshots, and to restore to different locations on different servers.

Check Cohesity Cluster Settings

It is important to understand the processing settings in the Cohesity cluster that is protecting your SQL databases. Choosing the optimal settings for deduplication, compression, and encryption can dramatically improve the performance of your Cohesity backups and archives.

Deduplication

Cohesity recommends leaving deduplication enabled and setting it to **Inline**, to reduce storage costs.

Deduplication: (Enabled by default.) When Deduplication is on, Cohesity eliminates duplicate blocks of repeating data stored on the Cohesity cluster, dramatically reducing the amount of storage space needed to store the data.

Inline Deduplication: Optionally toggle on. When Inline Deduplication is on, the deduplication occurs as the Cohesity cluster is saving the blocks to the partition. (There can also be additional deduplication after the Cohesity cluster has saved the blocks to the partition.) When Inline Deduplication is off, deduplication occurs *after* the Cohesity cluster has written the data to the partition and might ultimately be slower.

Compression

Cohesity recommends that you disable SQL Server-side compression and enable Inline Compression in the Cohesity platform. This also takes advantage of Cohesity deduplication, by allowing it to identify block patterns in the data.

Compression: (Enabled by default.) When Compression is on, the Cohesity cluster compresses all the data stored in the Storage Domain, reducing the amount of space needed to store the data.

Inline Compression: (Enabled by default.) If Inline Compression is on, the compression occurs as the Cohesity cluster is saving the blocks to the partition. If Inline Compression is off, the compression occurs *after* the Cohesity cluster has written the data to the partition and might ultimately be slower.

NOTE: Without SQL Server-side compression, the first backup might appear slower, but the benefits of deduplication will kick in starting with the subsequent backup, and the reduction in the volume of data outweighs any performance drop experienced while taking the first backup.

For more details, see Manage Storage Domains in the online Help.

Encryption

Cohesity recommends that data encryption be enabled for any data being sent to an External Target, to the public cloud, or any time sensitive data is being handled.

Data-at-Rest Security

The Cohesity SpanFS® file system provides full at-rest encryption based on the strong AES-256 CBC (Cipher Block Chaining) standard.

Data-in-Flight Security

The data being transmitted from a Cohesity cluster to External Targets is encrypted for security. Examples of data movement include: replicating between remote offices; and transmitting data to public cloud.

Data-in-Cloud Security

Cohesity's CloudArchive includes encryption. This functionality moves data from an on-premises cluster to cloud storage. For cold data, which is rarely accessed but must be retained, this is a far more economical solution. While encryption for CloudArchive operations can be turned off, Cohesity strongly recommends keeping it on, especially when working with External Targets in the public cloud.

For more see Cohesity Security Features in the online Help.

Fine-Tune MS SQL Server Parameters

Cohesity recommends that MS SQL Server hosts meet all Microsoft best practices.

To ensure that your backups run with the highest possible performance, review and optimize each component in the backup process, starting with the SQL Server host:

- MS SQL Server
- Network
- The Cohesity platform

Gauging MS SQL Server Host performance (Physical or VM) is important, as a SQL Server that is underpowered, equipped with poor resources, or overloaded will adversely affect any backup process.

Host CPU

Cohesity recommends that MS SQL Server hosts meet Microsoft SQL Server CPU recommendations and best practices. For a good reference on this, see Storage and SQL Server capacity planning and configuration.

For example, a SQL Server host (physical or VM) can have an average CPU usage between 1%-15%, with swells of 16%-30% and spikes of 65%-85% during high transaction volumes.

MS SQL Server consumes CPU resources equally across all processors on the host. It derives its transactional power from CPU cycles. If not managed, the operating system and MS SQL Server will compete for CPU cycles.

High CPU usage means fewer cycles to spend on taking SQL backups, leading to longer waits to release worker threads and resolve transactional commits to the database.

Host Memory

Cohesity recommends that MS SQL Server hosts meet Microsoft SQL Server memory recommendations and best practices. For a good reference on this, see Storage and SQL Server capacity planning and configuration.

MS SQL Server assumes that host memory exists solely for its own purpose. If not managed, the operating system and MS SQL Server will compete for host memory. For example, when a production Windows host will have 16GB-64GB of memory.

MS SQL Server aggressively attempts to push all database objects into memory — tables, indexes, stored code, and procedure caches — into memory, it has a large impact on total available memory. In turn, inefficient memory usage indirectly degrades backup generation.

A well provisioned production Windows Server host has a lot of memory. Approximately 6-10 GB is allocated for the operating system, and the rest is reserved for MS SQL Server. A properly configured MS SQL Server uses all available host memory. MS SQL Server automatically moves the most frequently used objects into memory. If memory pressure is too great, the OS will start writing to the system pagefile and this will have a significant impact on performance.

Host Windows Boot Drive (C:\)

Cohesity recommends that the C:\ drive be provisioned with 15%-20% free space to accommodate cloning and instant mounts.

The boot drive on a Windows Server host (physical or VM) is usually the C:\ drive. The amount of free space on the C:\ drive must be sized properly for database cloning.

Microsoft requires that the amount of free space on the host C:\ drive to be equal to or greater than 15% - 20% of the size of the largest cloned database. This is a Microsoft limitation as cloning uses Microsoft Vss (Volume Shadow copy Service) and requires space on the C:\ drive for the mount.

SQL Server Database settings

Cohesity recommends the SQL database RECOVERY_MODEL property be set to FULL.

The Full Recovery Model requires log backups to be taken on the database to offset the growth of the database log file. Taking log backups allows recovery of the database to any specific point in time.

NOTE: With the *Simple* Recovery Model, the database log file remains at one size, and recovery of the database is only available for the last backup point in time. The changes that occur after the most recent backups are left unprotected. **Cohesity recommends *against*** using the Simple Recovery Model in a production environment.

Link Active Directory

Active Directory is a common repository for information about objects that reside on the network, such as users, groups, computers, printers, applications, and files. Active Directory also maintains access on each object, which allows you to maintain permissions and control who can access and manage those objects.[1]

Cohesity recommends the Cohesity cluster be joined to the same Active Directory that manages the customer's SQL Server hosts.

Cohesity recommends the service accounts for MS SQL Server and SQL Agent be configured in Active Directory and have domain user permissions.

IMPORTANT: To successfully link a cluster to an AD server, the domain must be resolvable, and the DNS server must be reachable. For more information on joining a cluster to Active Directory, see Active Directory in the online Help.

[1] Desmond, Brian, et al. (2013). *Active Directory: Designing, Deploying, and Running Active Directory*. (5th ed., p. 1). O'Reilly.

Select Accounts and Grant Permissions

There are three accounts you must consider when installing the Cohesity Agent:

- **Installation Account**: The account you use to log in to the host and run the installer.
- **Service Account**: The account under which the Cohesity Agent service runs on the SQL Server host. (Selected at the time of installation.)
- **SQL Server login account**: The SQL Server account by which the Cohesity Agent has access to the databases. (Configured after installation.)

Installation Account

When installing the Cohesity Agent on a Microsoft Windows host, you must log onto the SQL Server host with an account that has local administrator privileges. This is because the Windows agent installer installs the Cohesity Agent service.

Microsoft requires the installation account have administrator privileges on the host.

Choose the Service Account

The installer gives you the option between two types of service accounts:

- The LocalSystem account
- A user account

Assigning privileges to the service account will differ depending on which type of account you choose. **Cohesity recommends** using either the LocalSystem account or an Active Directory Domain User account. See the chart below for a comparison of service account types.

Table 3: Comparison of Service Account Types

SERVICE ACCOUNT	TYPE	DEFINITION	RECOMMENDATION
LocalSystem (Default)	**LocalSystem Account (Built-in Host Account)**	The LocalSystem account is a predefined local account on the host computer.	**Recommended**: The LocalSystem account is a predefined local account used by the service control manager. It has extensive privileges on the local computer, and acts as the computer on the network. Its token includes the NT AUTHORITY\SYSTEM and BUILTIN\Administrators SIDs; these accounts have access to most system objects.[2]

[2] For more, see Microsoft's LocalSystem Account article.

SERVICE ACCOUNT	TYPE	DEFINITION	RECOMMENDATION
User Account	**Domain User Account** **(Active Directory)**	A user whose username and password are stored in Active Directory.	**Recommended**: A domain user account enables the service to take full advantage of the service security features of Windows and Microsoft Active Directory Domain Services. The service has whatever local and network access is granted to the account, or to any groups of which the account is a member.[3]
	Local User Account **(Host account)**	In Windows, a local user is one whose username and encrypted password are stored locally, only on the computer itself.	**Not recommended**: When you log in as a local user, the computer checks its own list of users and its own password file to see if you are allowed to log in to the computer. The computer itself then applies all the permissions and restrictions that are assigned to you for that computer.

TIP: If you are unsure which account to use, choose the LocalSystem account. This account already has most of the required permissions. The agent service account can be changed later if needed.

Pros and Cons of Service Account Types

Different types of service accounts have advantages and disadvantages that you should be aware of when deploying the Cohesity Agent. See the comparison chart below.

[3] For more, see Microsoft's Using a Domain User Account as a Service Logon article.

Table 4: Service Account Types

INSTALLER ACCOUNT CHOICE	ACCOUNT TYPE	PROS	CONS
LocalSystem **(Default)**	**LocalSystem Account** **(Built-in Host Account)**	The LocalSystem account is a predefined local account on the host computer and therefore requires the minimum setup and configuration. The LocalSystem account already exists on Windows machines. Has access to most system objects.	The difficulty of managing each individual LocalSystem account increases as the number of machines increase. For example, 200 machines = 200 LocalSystem accounts to manage. No central location to manage security. No central location to manage permissions. Additional permissions need to be managed in order to access network computers and storage.
User Account	**Domain User Account** **(Active Directory)**	Account username and password are stored in Active Directory. Can take full advantage of the service security features of Windows and Microsoft Active Directory Domain Services. Centralized location to manage security. Centralized location to manage permissions. Access to network computers and storage is easily available. Very scalable.	This account must be created before agent installation.
	Local User Account **(Host account)**	Good for a single system.	Not scalable in a production environment.

Assign Service Account Permissions

The agent service account needs permissions (authentication) from several different areas:

- MS SQL Server Instance
- Windows SQL Server host
- MS Active Directory (if applicable)
- As usage of your Cohesity cluster increases, it will need permissions to network entities such as users, groups, applications, computers, storage, shares, files, and other Cohesity clusters.

Table 5: Service Account Permissions Minimums

AGENT SERVICE ACCOUNT TYPE	AGENT SERVICE ACCOUNT PERMISSIONS		
	IN ACTIVE DIRECTORY	IN SQL SERVER	ON WINDOWS HOST
LocalSystem	N/A	"NTAuthority\System" login must also have sysadmin role.	Must also be a member of the Administrators group. Must have rights to "log on as a service."
User Account **(Active Directory)**	Must also be a member of the Domain Users group. Must also be a member of the Administrators group.	Login must also have sysadmin role.[4]	

NOTE: A common deployment error when using the LocalSystem account is to assume that the LocalSystem account belongs to the Administrators group on the host server. This is not always the case. Verify that the account "LocalSystem" belongs to the Administrators group.

For more, see the MS SQL Setup section in the online Help, and Microsoft's About Service Logon Accounts article.

NOTE: A common deployment error is to forget to assign the SQL Server sysadmin role to the agent service account in SQL Server.

Get the Synergy of SQL Server VSS Writer

The SQL Server Writer account is a login that comes with the SQL Server Instance by default. When a volume-based Protection Group requests a VSS snapshot of a Windows machine, the SQL Server Writer account steps in and launches the Windows VSS Service to execute the request.

[4] Backup or restore operations that use the Microsoft SQL Server Virtual Device Interface (VDI) require that the server connection for SQL Server that is used to issue the BACKUP or RESTORE commands must be logged on as the sysadmin fixed server role.

The SQL Server Writer service runs under the LOCAL SYSTEM account which has all the required permissions for it to function without interruptions. It is also provisioned as a Database Engine login by default. Note that this login comes out of the box with a SQL Server sysadmin fixed server role, and we recommend you retain this configuration to ensure the smooth functioning of the login.

For more information, see Configure Windows Service Accounts and Permissions in the Microsoft SQL documentation.

Install the Cohesity Agent for Windows

Cohesity recommends using agent-based backups as the preferred method. This allows you to have a single protection group that protects multiple SQL databases across multiple SQL instances, and the ability to back up and restore individual databases.

NOTE: To recommend the best possible deployment, you have to know how the SQL instance is configured. Not understanding the SQL Server configuration can lead to errors when you register it as a Cohesity.

Cohesity recommends that you know the SQL Server host configuration in order to properly register it as a Cohesity source.

Before registering a physical Server with the Cohesity cluster, you'll have to download and install the Cohesity Agent for Windows on each Server you plan to register. When you install the Cohesity Agent, you will be prompted to choose file-based or volume-based options, or both. See the next section.

Choose Agent Methods

Cohesity provides three different agent-based methods to back up and restore your database.

Cohesity recommends installing both the file- and volume-based CBT (Changed Block Tracker) persistence options.

Table 6: Agent Methods for Backup and Restore

AGENT METHODS FOR BACKUP			
METHOD	**REBOOT REQUIRED?**	**CBT PERSISTENCE[5]**	**WHAT IS PROTECTED?**
File-based	No — does not require a host reboot.	CBT does *not* persist through a reboot. (A full backup will be taken on the next job run).	DB files: .mdf, .ndf, .ldf
Volume-based	Yes — requires a host reboot.	CBT persists through a reboot.	The whole volume is protected: .vhd file Cohesity will back up both the database files as well as any other files on that given volume. Only volumes where the databases exist are backed up.

[5] CBT (Changed Block Tracking) persistence is a term used to describe whether or not CBT resumes tracking changes where it left off after a reboot of the server, or if CBT must start new tracking.

AGENT METHODS FOR BACKUP			
METHOD	**REBOOT REQUIRED?**	**CBT PERSISTENCE[5]**	**WHAT IS PROTECTED?**
VDI-based **(Virtual Device Interface)**	No — does not require a host reboot.	N/A	DB files: .mdf, .ndf, .ldf

NOTE: SQL Filestream databases can only be backed up using the volume-based backup method.

Consider File-Based Backups

A file-based backup protects only the MS SQL databases you choose. It captures only the database files for those selected databases. This approach contrasts with a volume-based backup, which captures any and all the files contained on the volume.

The advantage of a file-based backup is that a smaller amount of data is captured, which allows the protection group to run more quickly.

To enable file-based backup, install the File System CBT component during Cohesity Agent installation.

For more, see Protect MS SQL Server (File-based) in the online Help.

Consider Volume-Based Backups

A Cohesity MS SQL volume-based backup captures all MS SQL databases running on a VMware or physical Server, as well as all other files that are stored on the corresponding OS volume. This approach contrasts with a file-based MS SQL backup, which includes only the selected databases. The Cohesity cluster captures full database server backups and can optionally back up database transaction logs, so you can roll forward to any point in time.

For more, see Protect MS SQL Server (Volume-based) in the online Help.

Consider VDI-Based Backups

The Virtual Device Interface (VDI) is a Microsoft interface that allows the Cohesity agent to execute SQL Server Native backup and restore commands. This backup type produces a standard SQL Server Native backup of type (.BAK), transaction logs, (.TRN), and differentials, (.DIFF). These backup files are stored on a Cohesity View and take advantage of View compression, deduplication, encryption, replication, and archiving features.

The aim of the SQL DBA is to have a combination of backups so that the database can be *restored* to any point in time. A good combination of backups consists of having a full backup, differential (in Cohesity, *incremental*) backups, and log backups.

For example, all SQL database restores must begin with a full database backup, secondly, a differential can then be applied to the full, and finally, log backups can be applied in sequence to complete the database restore.

When the backups are applied during the database restore process, you are sequentially adding the captured changes to the database: FULL+DIFF+Log1+Log2+Log3 = Restored Database.

IMPORTANT: All Microsoft VDI backups, their differentials, and their logs are dependent on a full backup to perform a database restore. Microsoft requires that in order to restore a SQL database, you must start with a full backup, then its transaction logs can be applied. This means your backup retention policy must keep a full backup along with its log backups in order to successfully restore a database.

Simply put, a SQL VDI-based database restore requires a full backup to seed the database, then a differential and/or log backups are applied to the specified point in time.

We recommend retaining two sets of full backups with their differential and log backups.

TIP: A good backup plan always includes a combination of full, differential, and log backups.

Compare Volume, File, and VDI-Based Methods

Use the table below to compare the implications of each method.

Table 7: Comparison of Volume, File, and VDI backup Methods

COHESITY FACET	BACKUP METHOD		
	VOLUME	FILE	VDI
Technology Employed	Cohesity Agent uses Vss, for volume CBT, and VDI for T-logs.	Cohesity Agent uses Vss and Cohesity technology for file CBT, and VDI for T-logs.	Cohesity agent uses SQL Native backup commands.
What does it capture?	Captures all the volumes attached to the server *and everything on those volumes*.[6]	Captures database files.	Captures the database in a SQL Server standard .BAK, .TRN, or .Diff file
Logs	Yes. Captures a SQL log backup of the database.	Yes. Captures a SQL log backup of the database.	Yes. Captures a SQL Server native log backup of the database.
Cohesity Source Registration	Register as a SQL Server application.	Register as a SQL Server application.	Register as a SQL Server application.

[6] Often, there are other files, like old .bak files, that sit on the volume along with the active database files. A large amount of other files will increase the duration of the protection group because these files will be captured along with the database files.

COHESITY FACET	BACKUP METHOD		
	VOLUME	FILE	VDI
Compatibility with SQL Server High Availability Features	**Windows Failover Cluster: Yes**, backup will succeed if a failover occurs. Must be registered as a Cohesity "SQL Cluster" source. **AAG: Yes**, backup on the correct replica will succeed after an AAG failover occurs. **Mirroring: Yes***. Each SQL instance must have its own protection group. **Log Shipping: Yes***. Each SQL instance must have its own protection group. *** NOTE**: While Cohesity currently does not include specific features for mirroring and log shipping, it does not conflict with these SQL Server features.		**Windows Failover Cluster: Yes**, backup will succeed if a failover occurs. Must be registered as a Cohesity "SQL Cluster" source. **AAG: Yes**, backup on the correct replica will succeed after an AAG failover occurs. Mirroring: No. Log Shipping: No.
Method	This is an MS SQL Server database backup at the **volume level**. Allows for FULL SQL Server backups — all databases, logs, and files/folders on the volumes are backed up. MS SQL Server is registered as a Cohesity source and MS SQL application. The Cohesity Agent uses Vss, the Cohesity Volume CBT, and VDI to take an application-consistent snapshot of the volumes on that server.	This is an MS SQL Server database backup at the **file level**. Allows for FULL SQL Server backups with logs, but only the database files are captured. The Cohesity Agent uses Vss, the Cohesity File CBT, and VDI to take an application-consistent snapshot of the volumes on that server. Faster because it's not backing up extraneous files. **NOTE**: Cannot browse files or folders as in a volume-based backup.	This is a native MS SQL Server database backup at the **file level**. Allows for FULL SQL Server backups with logs and Differentials, but only the database is captured. Agent uses SQL Server native backup and restore commands , to take an application-consistent backup of the database.
Recovery/Restore	The database host server should be backed up and restored via a different protection group. Supports Point-in-Time (PIT) restore of the database from across the entire backup history.		

COHESITY FACET	BACKUP METHOD		
	VOLUME	FILE	VDI
Cloning	**Yes**. Can produce a representative volume. *Advantage*: Applications can be tested with data that is as close to the production environment as possible. Can reproduce a volume. *Advantage*: Applications can be tested in an environment that can be destroyed and re-created on demand. No load on Production servers. The backup is NOT modified and therefore always safe. Uses storage resources on Cohesity cluster.	**Yes**. Can produce a representative database. *Advantage*: Applications can be tested with data that is as close to the production environment as possible: Can reproduce a database. *Advantage*: Applications can be tested with databases that can be destroyed and re-created on demand. No load on Production servers. The backup is NOT modified and therefore always safe. A successful clone is immediate validation of the backup's integrity. Uses storage resources on Cohesity cluster.	**Cloning is scheduled as a future enhancement.**
Archival	**CloudArchive:** Cohesity supports the ability to send backups to public cloud providers like AWS, Azure, and Google Cloud Platform. It also supports search and restore from them, back to the local site when needed.		
Replication	**Managing Copies of backups on another Cohesity cluster:** Backups, files, and volumes can be replicated to a second Cohesity cluster.		

Install and Manage the Cohesity Agent for Windows

You are now ready to install the Cohesity Agent for Windows on each SQL Server you plan to back up with Cohesity. From the SQL Server host, log in to the Cohesity platform, navigate to **Protection > Sources**, and download the Cohesity Agent Windows. Install the Cohesity Agent by launching the downloaded executable. The agent starts automatically.

You can install the Cohesity Agent for Windows on multiple Windows servers through the Cohesity DataPlatform CLI (command line interface). For complete instructions on installing the Cohesity Agent, see Install and Manage the Agent on Windows Servers in the online Help.

NOTE: You can upgrade the Cohesity Agent through the MS SQL Dashboard in Cohesity. When an upgrade becomes available for any agents that you've installed, an **Upgrade Agent** icon will appear next to that agent in the MS SQL Dashboard.

Set Up the Cohesity Agent

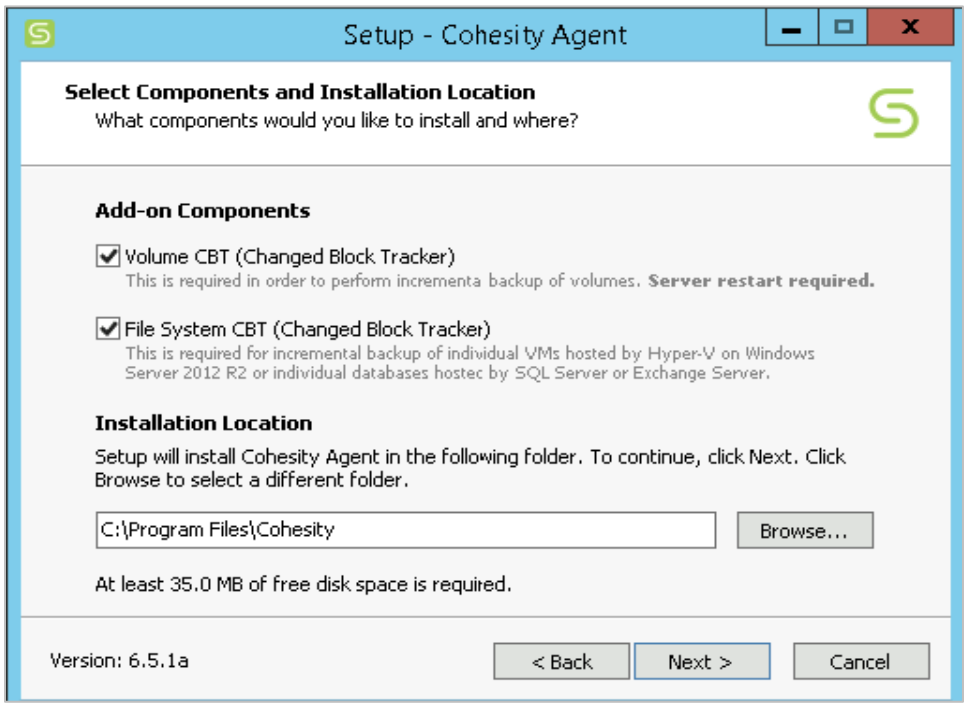
By default, the Volume CBT (Changed Block Tracker) component is selected for installation. This component is required to perform incremental backups and requires a reboot to function.

Without a reboot, only volume based full backups can be performed on the server.

The File System CBT (Changed Block Tracker) component is required for MS SQL file-based protection, as well as incremental backups of database files. The installation does not require a reboot for this component to function.

Cohesity recommends installing both the Volume and File CBT components of the Cohesity Agent.

Figure 1: Cohesity Agent Installation Options



Know SQL Database Types and Source Registration

Cohesity handles different types of MS SQL Server Databases, but not all databases are registered the same way.

When it comes to SQL protection, you need to handle the variety of server database types associated with SQL Servers. Cohesity DataProtect helps you overcome this challenge with its native capability to handle these databases, each of them in their recommended, specifically optimized ways.

Identify the Type of Database

Before proceeding to deploy the Cohesity agent for SQL data protection, we recommend you correctly identify the type of SQL database you want to protect.

Below is a table of SQL database types that are supported by the Cohesity SQL agent, and their recommended protection methods.

Table 8: MS SQL Server Database Types

DB TYPES	DEFINITION	PROTECTION METHOD		
		VOLUME-BASED	FILE-BASED	VDI-BASED
Stand Alone	A SQL Server User database	✔	✔	✔
AG	A database that belongs to an availability group (AG). For each availability database, the availability group maintains a single read-write copy (the primary replica) and one to eight read-only copies (secondary replicas). For details, see Always On availability groups: a high-availability and disaster-recovery solution in the Microsoft documentation.	✔	✔	✔
FCI	A database that belongs to a Failover Cluster Instance (FCI), which is a single instance of SQL Server that is installed across Windows Server Failover Clustering (WSFC) nodes.	✔	✔	✔
CDC	A change data capture (CDC) enabled SQL database records activity and is applied to an internal table. This makes the details of the changes available in an easily consumed relational format. For details, see About Change Data Capture (SQL Server) in the Microsoft documentation.	✔	✔	✔

DB TYPES	DEFINITION	PROTECTION METHOD		
		VOLUME-BASED	FILE-BASED	VDI-BASED
TDE	A Transparent Data Encryption (TDE) enabled SQL database has its data files encrypted, known as encrypting data at rest. TDE performs real-time I/O encryption and decryption of the data and log files. For details, see Transparent Data Encryption (TDE) in the Microsoft documentation. **NOTE**: Cohesity does not manage the keys associated with the TDE database.		✔	✔
File Stream	A Filestream enabled SQL Database is a database which contains a table that stores its data outside the database on the host file system. SQL Filestream leverages the OS file system's ability to handle large objects ®BLOBS© like image ¨PDFs¨ Binary ¨Bmp¨ Docs¨ and other unstructured data. For details, see FILESTREAM (SQL Server) in the Microsoft documentation.	✔		✔
SQL System	SQL Server maintains a set of four system-level databases (*master, model, msdb, tempdb*), which are essential for the operation of a server instance. System databases must be backed up after every significant update. The system databases that you must always back up include msdb, master, and model. For details, see Backup & restore: system databases (SQL Server) in the Microsoft documentation.	✔	✔	✔

Now that you can Identify the type of SQL database you want to protect, you can properly register it as a Cohesity source.

Register the Database Properly

Registering the database appropriately ensures that you are able to make the most of the Protection Group.

Table 9: Database Types by Source Registration

DATABASE TYPES	REGISTER AS	NOTES
Stand Alone	Physical	All Cohesity foundational backup and restore features are available.
AG	All nodes as Physical	All Cohesity foundational backup and restore features are available. Protection Group will follow the db through a failover.
FCI	SQL Cluster	All Cohesity foundational backup and restore features are available. Backup Group will follow the db through a failover. Use the SQL Instance VIP.
CDC	Physical	All Cohesity foundational backup and restore features are available.
TDE	Physical	All Cohesity foundational backup and restore features are available. The user must manually handle the certificate.
Filestream	Physical	Volume-based backup and restore.
SQL System	Physical	All Cohesity foundational backup and restore features are available. Because of the special nature of these databases some Cohesity gflags need to be set. Please contact support for assistance.

Once you get your SQL database registered as a source in the Cohesity platform, you can immediately start to take full advantage of Cohesity's data protection features.

Deploy for Specific SQL Server Features

There is a direct connection between how you deploy the Cohesity Agent and how you register SQL Server as a source. This relationship determines which types of backups can be taken.

Deploying the Cohesity Agent for the right SQL configuration makes backups and restores more efficient. The most common SQL Server configurations are:

- Standalone Server (Physical or VM)
- MS SQL Server Always On Availability Groups (AAG)
- SQL Cluster, aka Windows Failover Cluster Instance (FCI)
- SQL VM

See the following sections for important notes and considerations on setting up for each configuration.

Set Up for a Standalone Server (Physical or VM)

Cohesity supports registering MS SQL either as a physical or as a virtual server but not both. Before registering an MS SQL Server, you must download and install the agent on the host server.

Arrange for MS SQL Server Always On Availability Groups (AAG)

The Microsoft SQL Always On Availability Groups feature is a high-availability and disaster-recovery solution. Always On Availability Groups maximizes the availability of a set of user databases.

An availability group supports a discrete set of user databases, known as availability databases, each with its own transactionally consistent replica. Because of the transactional consistency secondary Replicas can be used to generate backups or can be used for reporting.

An availability group supports a set of read-write primary databases and one to four sets of corresponding secondary databases[7]. For more information, read Microsoft Always On Availability Groups (SQL Server)

The benefit of MS SQL Server Always On Availability Groups is that it ensures transactional integrity across all its replicas. AAG databases can failover from Primary to one of the Replicas and back again so that any one of the replicas can function as the Primary database.

Cohesity will take a backup using the assigned backup preference even after a failover.

Cohesity recommends you register each node in the AAG relationship as a Physical Server.

For more on registering a SQL AAG configuration and applying a protection group, see Cohesity DataPlatform with MS SQL Server Always On Availability Group Solution Guide.

[7] *Always On Availability Groups (SQL Server)",* Microsoft Corp., https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2012/hh510230(v=sql.110)

You may run into this: A protection group for SQL AAG fails the first time after SQL database fail over within an AAG group; however, it succeeds after the second attempt. This behavior results from the AAG metadata change that is triggered when the SQL database fails over. If the AAG configuration changes, such as adding or removing a database or changing the backup preference or failover, then the next log or full backup will fail. This is a known behavior. The failed backup run will update the entity hierarchy with changed metadata, and the subsequent run should succeed.

Position for SQL Cluster

Also known as Windows Failover Cluster Instance (FCI), this is a common production configuration. It provides high availability for the databases by moving SQL instances to a second host if the first host becomes unavailable. Cohesity supports SQL Failover Cluster Instances (FCIs). Each SQL FCI node (host) must be registered as a unique source, as each SQL FCI might become active on different nodes.

Cohesity recommends you register an FCI by using the virtual IP (SQL VIP) for each of the SQL instances in the Windows cluster. Properly registering an FCI as a Cohesity MS SQL Cluster source enables Cohesity to follow the SQL instance during a failover to new nodes, and then continue to take backups.

NOTE: A common mistake is to register a SQL FCI using the physical IPs or the Windows cluster IP instead of the SQL Server's *virtual IPs*.

NOTE: Cohesity does not support Clustered Shared Volumes (CSV). CSV refers to the storage underneath the SQL FCI. Clustered Shared Volumes enable multiple nodes in a failover cluster to simultaneously have read-write access to the same LUN (disk) that is provisioned as an NTFS volume.

NOTE: While the Cohesity Agent will follow the SQL instance during a failover, it does not support Microsoft Failover Cluster awareness for file services.

Before registering an MS SQL Server cluster, you must download and install the Cohesity Agent on each node within the MS SQL cluster. For detailed instructions, see Set Up an MS SQL FCI in the online Help.

Strategize for a SQL VM

Cohesity supports backing up a SQL server by registering the server as a VM backup and as a physical server backup. When selecting a backup type, consider:

- A VM backup takes a snapshot of the VM at a point in time and backs up the OS hosting SQL as well as any other data in the VM, including the SQL server instance. A VM backup leverages the hypervisor to access the VM, to which an agent is deployed. A restore would restore the entire VM at that point in time.
- A SQL Server backup using the Cohesity physical agent supports backing up specific databases which can be restored to a new server.

You can configure for both methods. Doing so can help meet specific requirements, such as a weekly backup of the VM and a daily backup of the SQL databases.

Cohesity recommends creating two separate protection groups; one to protect the SQL databases and one to protect the VM.

The protection group to protect the SQL Server databases will be an agent-based backup.

The protection group to protect the VM will be a hypervisor-based backup via Protection Policy and must be configured to take application-consistent backups using a VM Protection Policy. Choosing this approach will:

- Use VSS integration via VMware Tools.
- Be Incremental forever with VMware CBT.
- Facilitate a volume-level recovery.
- *Not* have separate log backup for the VM, which means no Point-in-Time (PIT) recovery.

Consider Non-Agent-Based Backups

SQL Native backups are still widely used and a viable backup technology for the foreseeable future. Cohesity and SQL Native backups pair very well together.

SQL Native Backups

Bringing together MS SQL Server Native backups and Cohesity allows you to keep your current backup process and take advantage of Cohesity's powerful features like replication, compression, and archiving.

When you deploy a SQL Native backup, **Cohesity recommends** using a View to manage and back up files, reduce storage, protect backup files, and archive them for long-term retention.

No agent installation is necessary for an MS SQL Native backup.

IMPORTANT: When deploying Cohesity with SQL Native backups, be sure to assign READ/WRITE permission for the SQL AGENT service account to your Cohesity Views.

For a full discussion of using SQL Native backups with Cohesity, see Streamline Microsoft SQL Server Native Backups with Cohesity DataPlatform.

Your Feedback

Was this document helpful? Send us your feedback!

About the Authors

Scott Lorenz is a SQL Solutions Engineer at Cohesity. In his role, Scott focuses on business-critical applications, MS SQL Server databases, and data protection with Enterprise and Cloud Storage.

Other essential contributors included:

- Freddy Grahn, Senior Technical Field Enablement Engineer
- Dave Porco, Sr Solution Architect
- Erin Zaborowski, Sr. Site Reliability Engineer
- Bart Abicht, Sr Technology Writer & Editor, Technical Marketing & Solutions Engineering

Document Version History

VERSION	DATE	DOCUMENT HISTORY
1.0	July 2019	Original document
1.1	Oct 2020	Content update
1.2	Apr 2021	Minor update
1.3	July 2024	Republished

ABOUT COHESITY

Cohesity radically simplifies data management. We make it easy to protect, manage, and derive value from data -- across the data center, edge, and cloud. We offer a full suite of services consolidated on one multicloud data platform: backup and recovery, disaster recovery, file and object services, dev/test, and data compliance, security, and analytics -- reducing complexity and eliminating mass data fragmentation. Cohesity can be delivered as a service, self-managed, or provided by a Cohesity-powered partner.

Visit our website and blog, follow us on Twitter and LinkedIn and like us on Facebook.