SQL Server 2008 Administration in Action

Sample Chapter

Rod Colledge

Foreword by Kevin Kline

MANNING

SQL Server 2008
Administration in Action

ROD COLLEDGE

MANNING
Greenwich
(74° w. long.)

SQL Server 2008

Administration in Action

by Rod Colledge

Chapter 5

brief contents

5 Failover clustering

In this chapter, we'll cover

- Clustering architecture
- Advantages and limitations of clustering
- Quorum models
- Clustering topologies
- Installing a clustered SQL Server 2008 instance

Although redundant component design, such as dual-power supplies, provides fault tolerance at a component level, failover clustering operates at the server level, enabling ongoing operations in the event of a complete server failure. Complementary to component redundancy, failover clustering is a commonly used high availability technique for SQL Server implementations and is the focus of this chapter.

In addition to the requisite SQL Server skills, successfully designing and administering a clustered SQL Server environment requires skills in a number of areas, including the configuration of cluster-compatible hardware components. While Windows Server 2008 has made clustering SQL Server somewhat easier, it's still a complex process requiring considerable planning and broad skills.

Rather than attempt to provide full coverage of the clustering design, creation, and administration process (such a goal would require at least an entire book!), this chapter focuses on installing a clustered SQL Server instance. Let's begin with a broad overview of clustering, exploring its benefits and limitations from a SQL Server perspective, and tackling important preinstallation configuration tasks.

5.1 Clustering overview

As a cluster-aware application, a SQL Server instance can be installed into an existing Windows cluster, creating what's referred to as a *failover clustering instance*. Once installed, the instance is accessed using a network name (aka *virtual server name*) without needing to know which of the underlying physical cluster servers the instance is currently running on.

The abstraction between a physical and a virtual server is a key component in providing continued database availability after a failure event. In the event of a server failure, the SQL Server instance automatically moves, or *fails over*, to another cluster server, while continuing to be accessed using the same virtual server name.

In this section, we'll take a high-level look at clustering architecture, including its benefits, limitations, and common usage scenarios.

5.1.1 Clustering architecture

Built on top of a Windows Server failover cluster, SQL Server clustering uses the *shared nothing* architecture, a term used to distinguish its clustering implementation from those of other database vendors, such as Oracle's *Real Application Cluster* (RAC). Under the shared nothing architecture, a SQL Server database instance is active on only one physical clustered server at any given time.

Unlike a network load-balanced solution, SQL Server failover clustering isn't a high-performance solution; in other words, the performance won't be any better, or worse, than a nonclustered implementation. SQL Server clustering is purely a high-availability solution and you should not choose it for any other reason.

Figure 5.1 illustrates a simple example of a two-node cluster with a single SQL Server failover clustering instance. Ordinarily, the instance resides on the *Sales1* server. In the event of a failure of this server, the SQL Server instance automatically fails over to the *Sales2* server. In either case, the SQL instance will continue to be accessed using the same virtual server name.

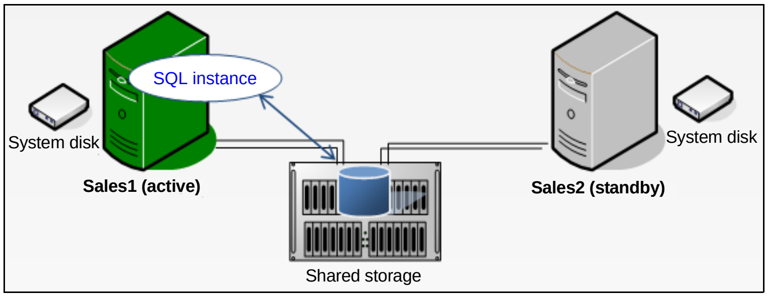


Figure 5.1 A simple failover clustering example in which a SQL Server instance can move from one cluster server to another in the event of failure

Key components of a Windows Server failover cluster solution are *shared storage* and *resource arbitration*. In figure 5.1, we can see both the Sales1 and Sales2 servers have access to shared storage. The databases contained in the SQL instance reside on the shared storage to enable either server to access them when required. But in order to prevent both servers from accessing them at the same time, and therefore causing data corruption, the Windows clustering service arbitrates ownership of the disk volumes based on the server's current role in the cluster.

Clusters consist of one or more *resource groups*, which are collections of resources required by a clustered application. In the case of a clustered SQL Server instance, its resource group would include the disks containing the database instance's data and transaction logs, an IP address and network name, and the three application services: SQL Server, SQL Agent, and full-text service. *Failover* is a term used to describe the transfer of ownership of resources from one server to another; it occurs at the resource group level, ensuring that all resources required for the application are moved and available on the failover server.

Given that a database instance can be owned by various servers in the cluster, applications are configured to connect to the virtual server name rather than the name of a physical cluster server. To avoid confusion with the Microsoft Virtual Server product, SQL Server 2005 and above refer to *failover clustering instances* rather than the virtual server name known in SQL Server 2000 and earlier.

As we move through this chapter, the terms and concepts we've covered thus far will become clearer, particularly when we walk through an installation of a clustered instance. For now, let's turn our attention to the major advantages and limitations of SQL Server clustering.

5.1.2 SQL Server clustering advantages and limitations

As a high-availability technology, clustering has a number of advantages and limitations when compared to other SQL Server high-availability options such as database mirroring and transaction log shipping (covered in chapter 11). Its primary advantage is that in the event of a server failure, the *entire instance and all of its databases* are moved to a failover server, a process that usually takes no more than about 90 seconds. This stands in contrast to mirroring or log shipping solutions, which are established on an individual database-by-database basis.

In addition to providing protection from unexpected server failure, the other major benefit of clustering is the ability to reduce downtime during planned outages. For example, consider a requirement to upgrade a server's RAM. Unless we have a server with hot-add RAM capabilities, we'd need to power off the server, upgrade the RAM, and power back on again, and during this time the database instance would be unavailable. In a clustering scenario, we could manually initiate failover, which would move the instance from one server to the other, enabling the upgrade to occur while the instance is available on another cluster server. Thus, the downtime is limited to the failover time, typically about 1 minute.

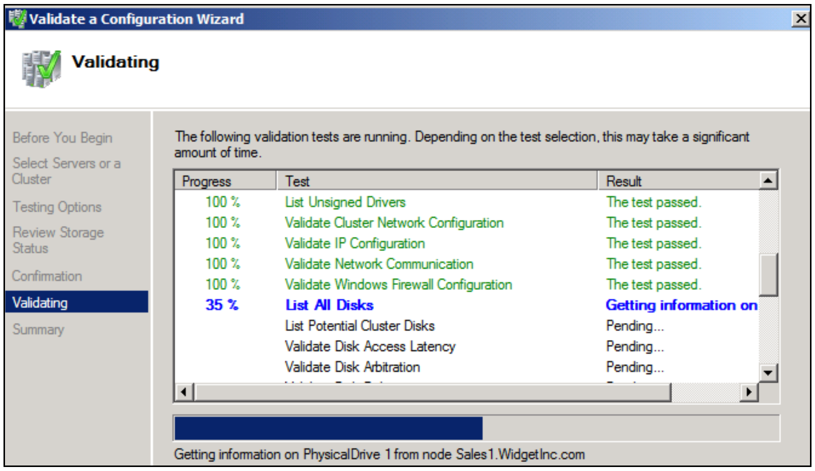
Unlike database mirroring and transaction log shipping, the major limitation of clustering, particularly in Windows Server 2003 and earlier, is that other than a RAID solution, there's no protection from failure of the disks containing the database files and/or the cluster quorum resource, discussed shortly. Further, typical clustering solutions don't offer protection from geographic disasters—that is, if all clustered servers exist in a single location, the destruction of that location will result in an outage. Although multisite clusters and the new Windows Server 2008 quorum models (discussed shortly) address these limitations, it's a common practice to combine multiple high-availability solutions to deliver the advantages of each while minimizing the individual limitations. For example, important databases within a clustered SQL Server instance can be mirrored to an offsite location for geographic (and disk) protection. We'll cover such combinations in more detail in chapter 11.

5.1.3 Clustering in Windows Server 2008

A SQL Server failover clustering instance is installed on, and in some ways is constrained by, the underlying Windows cluster. Although Windows Server 2003 and earlier had a solid and full-featured clustering solution, there were a number of limitations that constrained the configuration of the installed SQL Server failover clustering instances.

The SQL Server 2008 release was preceded by the release of Windows Server 2008, bringing with it a substantial improvement in the clustering options. In comparison to Windows Server 2003, the clustering improvements in Windows Server 2008 include the following:

- An enhanced *validation test*, as shown in figure 5.2, which can be used to ensure the validity of the hardware and software components in forming a cluster
- Support for IPv6 and up to 16 cluster servers (increased from 8)



The following validation tests are running. Depending on the test selection, this may take a significant amount of time.

Progress	Test	Result
100 %	List Unsigned Drivers	The test passed.
100 %	Validate Cluster Network Configuration	The test passed.
100 %	Validate IP Configuration	The test passed.
100 %	Validate Network Communication	The test passed.
100 %	Validate Windows Firewall Configuration	The test passed.
35 %	List All Disks	Getting information on
	List Potential Cluster Disks	Pending...
	Validate Disk Access Latency	Pending...
	Validate Disk Arbitration	Pending...

Getting information on PhysicalDrive 1 from node Sales1.WidgetInc.com

Figure 5.2 Use the Windows Server 2008 Validate a Configuration Wizard to ensure the validity of the cluster components.

- The ability of cluster servers to obtain IP addresses via DHCP
- Support for new *quorum* models, used to determine the number of failures a cluster can sustain while still operating

Apart from the enhanced validation and management tools, the increase in supported server nodes and relaxed networking restrictions enables the creation of much more flexible clustering solutions, making Windows Server 2008 the best server operating system for a clustered SQL Server 2008 deployment. Further enhancing its appeal are the improved quorum models.

Failover Cluster Configuration Program (FCCP)

Cluster hardware should be certified by both the hardware vendor and Microsoft as cluster compatible. With the release of Windows Server 2008, Microsoft announced the *Failover Cluster Configuration Program* (FCCP). Hardware vendors will certify complete cluster configurations against this program, making the process of choosing a cluster-compatible hardware solution much simpler than in the past.

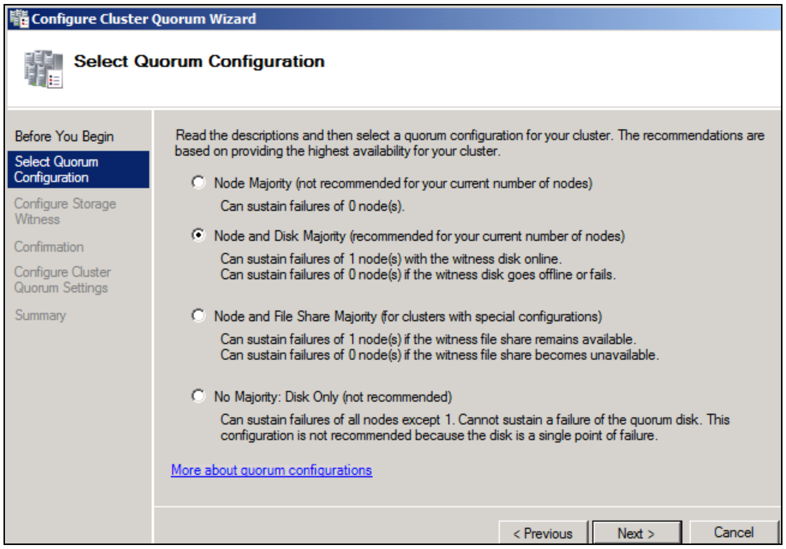
5.1.4 Quorum models

A fundamental aspect of Windows Server clustering is the process whereby each node is assigned unambiguous ownership of a resource. For example, consider a two-node cluster like the one you saw in figure 5.1. If the network link between the two cluster nodes temporarily drops, what process prevents both servers from assuming that they're now the owner of the SQL instance? This process to resolve such an occurrence (commonly called the *split brain* problem) is referred to as the *cluster quorum*.

In earlier versions of Windows clustering (prior to Windows Server 2003), cluster quorum was maintained using a shared disk resource and a quorum database containing the resources and owners. In our previous example, a link failure between the two nodes would be resolved by only one of the nodes having ownership of the quorum disk. When the link is dropped, the node with quorum disk ownership continues its role and takes on the roles (if any) of the other node.

Despite the simplicity and effectiveness of this quorum model, the quorum disk was a single point of failure. Further, given the need for all cluster nodes to have access to the shared disk resource containing the quorum disk, the constraints of typical shared storage hardware prevented the creation of geographically dispersed clusters. Windows Server 2003 addressed this with the introduction of the Majority Node Set (MNS) quorum.

Nodes in an MNS quorum cluster operate with local copies of the quorum database, avoiding the limitations of shared storage, but in order to prevent the split brain problem, a majority of nodes must remain in contact for the cluster to be considered valid. One of the key attributes of an MNS cluster is the requirement for a minimum of three nodes to form a valid cluster, and in the case of a five-node cluster, a majority of the nodes (three) would need to remain in contact for the cluster to continue operating.


Configure Cluster Quorum Wizard
Before You Begin
Select Quorum Configuration
Configure Storage Witness
Confirmation
Configure Cluster Quorum Settings
Summary
< Previous
Next >
Cancel

Select Quorum Configuration

Read the descriptions and then select a quorum configuration for your cluster. The recommendations are based on providing the highest availability for your cluster.

- ○ Node Majority (not recommended for your current number of nodes)
  Can sustain failures of 0 node(s).
- ● Node and Disk Majority (recommended for your current number of nodes)
  Can sustain failures of 1 node(s) with the witness disk online.
  Can sustain failures of 0 node(s) if the witness disk goes offline or fails.
- ○ Node and File Share Majority (for clusters with special configurations)
  Can sustain failures of 1 node(s) if the witness file share remains available.
  Can sustain failures of 0 node(s) if the witness file share becomes unavailable.
- ○ No Majority: Disk Only (not recommended)
  Can sustain failures of all nodes except 1. Cannot sustain a failure of the quorum disk. This configuration is not recommended because the disk is a single point of failure.

More about quorum configurations

Figure 5.3 Available quorum models in Windows Server 2008

Windows Server 2008 further enhances the quorum model with a voting system. As you can see in figure 5.3, a number of options are available, with a recommended option based on the number of cluster nodes.

In summary, the following general recommendations apply to the selection of a quorum model in 2008:

- *Node Majority*—Used for clusters with an odd number of nodes
- *Node and Disk Majority*—Used for clusters with shared storage and an even number of cluster nodes
- *Node and File Share Majority*—Designed for multisite and Microsoft Exchange clusters
- *No Majority*—Compatible with earlier quorum models; isn't recommended given the single point of failure on the quorum disk

Before we look at some of the more important preinstallation configuration tasks, let's take a moment to dig a little deeper and review common clustering topologies.

5.2 Clustering topologies and failover rules

There are many aspects to consider when designing and planning high availability with SQL Server clustering. Chief among them is the number of dedicated *passive*, or *standby*, servers to include in the cluster for failover purposes.

In simple clusters with a 1:1 *working:standby* ratio, each working server is matched with a standby server that, during normal operations, isn't used for any purpose other than to simply wait for a failure (or planned failover). Alternatively, both (or all) servers can be active, but this introduces resource complications during failovers; for example, will a server be able to handle its own load plus that of the failed server?

In previous versions of clustering, *Active/Passive*, or *Active/Active* terminology was used to define the usage of two-node server clusters. The current terms are *single-instance* or *multi-instance* and are more appropriate terms considering the ability of today's clusters to contain up to 16 servers and many more SQL instances, with one or more standby servers available for failover from any number of other active servers.

This section will focus on the various clustering topologies commonly used and the resource considerations for each.

5.2.1 Single-instance clusters

The example earlier in figure 5.1 contained a single SQL Server instance in a two-node cluster. Under normal operating conditions, one node is in the standby status, existing purely to take on the load in the event of failure or a planned outage. When such action occurs, the instance moves between one server and the next, as shown in figure 5.4.

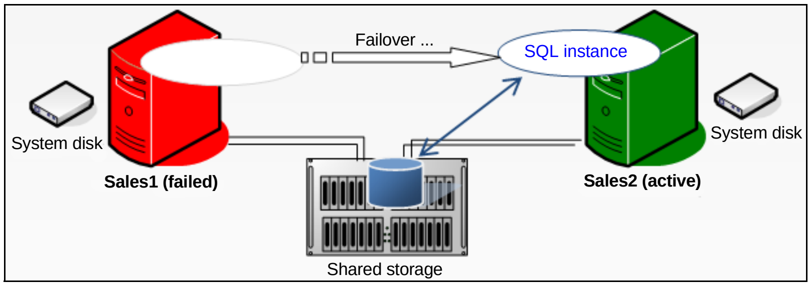


Figure 5.4 In the event of failure (or planned failover), single instance clusters such as this one have a simple failover process, with an instance moving from the active/failed node to the standby.

This clustering topology, known as a *single-instance cluster*, is the simplest to understand and administer, and also provides the highest availability while avoiding any performance impacts after failover. As such, it's commonly used to protect mission-critical SQL Server applications.

The major downside of this topology is the cost. During normal operation, one server isn't used in any capacity, and depending on the cost of the server, this can be an expensive approach. In addressing this, the multi-instance topology is frequently used.

5.2.2 Multi-instance clusters

As the name suggests, a *multi-instance* cluster contains multiple SQL Server instances. In a typical two-node multi-instance cluster, each cluster node runs one or more instances, and a failover situation causes an instance to assume another node's workload in addition to its own. In the example in figure 5.5, the *Sales* SQL instance resides on the Sales1 server, and the *Marketing* SQL Instance resides on the Sales2 server. In the event of failure of Sales1, Sales2 will run both the Sales and Marketing instances, potentially reducing the performance of both instances.

Because of the increase in server utilization, multi-instance clusters are typically used in budget-constrained environments, or in those valuing high availability much higher than reduced performance in a failed state.

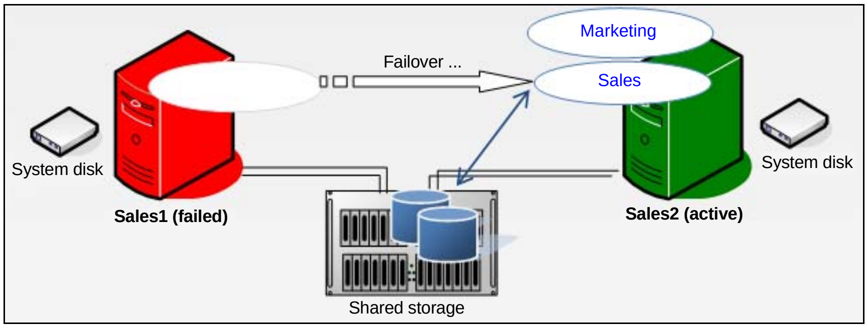


Figure 5.5 A two-node multi-instance cluster. Each node needs to be capable of handling the load of both instances during failover scenarios.

The multi-instance example in figure 5.5 achieves higher resource utilization than the single-instance example presented earlier; both servers are utilized in the normal working status.

Multi-instance clusters require careful consideration in terms of resource configuration. In our example, if the Sales1 and Sales2 servers each have 32GB of RAM, and both the Sales and Marketing SQL instances are using 28GB of RAM, a failover situation would see one (or both) instances with a potentially drastically reduced memory allocation after failover, a scenario I refer to as a *resource crunch*.

To avoid this issue, configure instances so that the sum total of maximum resource usage across both (or all) instances doesn't exceed the total resources of a *single* node that the instances could end up running on.

The two topologies we've covered so far represent the opposite ends of the scale; single-instance clusters with a 50 percent node utilization, and multi-instance clusters with 100 percent resource utilization. In between these two lies the N+1/M cluster.

5.2.3 N+1/M clusters

To avoid the cost of idle servers and limit the effects of a failover-induced resource crunch, a commonly used cluster configuration is an *N+1/M* cluster, whereby one or more standby servers exist for more than one working server. For example, in a three-node cluster, two nodes may be active, with a third existing as a failover node for both active nodes. Similarly, a five-node cluster with three active nodes and two failover nodes is a common cluster configuration.

As the number of cluster nodes and SQL Server instances increases, so too does the importance of considering the failover rules governing which node(s) a SQL Server instance can fail over to.

5.2.4 Failover rules

By regularly polling cluster nodes using a series of mechanisms called *LooksAlive* and *IsAlive* checks, a Windows cluster may conclude that a cluster node has failed. At that point the resources hosted by the node are failed over to another cluster node.

In a simple two-node cluster, the failover process is straightforward, as demonstrated in figure 5.4. But consider a five-node cluster containing two passive/standby nodes. Which (if any) of the standby nodes will be used for failover purposes when one of the active nodes fails? In large clusters containing many SQL Server instances, this is a particularly important consideration in maintaining a well-balanced cluster, ensuring that a single node doesn't carry a disproportionate burden of load.

Although beyond the scope of this book, there are several common strategies and techniques used in controlling failover, and central to them is the use of the *Preferred Owners*, *Possible Owners*, and *Failback* settings.

In figure 5.6, the properties of the SQL Server resource shows the Possible Owners property. This property lets you specify which nodes the instance is permitted to fail over to. In a similar manner, the Preferred Owner setting (not shown) is used to set the preferred failover node (this may not be chosen in some situations—for example, if the node is unavailable).

Finally, the Failback options (not shown) let you determine whether or not resources fail back to the original node if the failed node comes back online.

With this background in mind, let's continue by walking through the installation of a clustered SQL Server instance into a two-node Windows Server 2008 cluster.

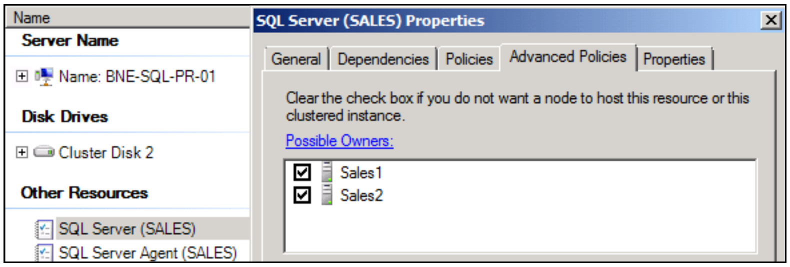
5.3 Installing a clustered SQL Server instance

The process of installing a clustered SQL Server instance has changed since SQL Server 2005. There are now two installation options: Integrated and Advanced.

5.3.1 Integrated vs. advanced installation

An integrated installation creates a single-node failover cluster, from which additional nodes (nodes on which the instance can fail over to) are added via a separate installation. As shown in figure 5.7, the initial and subsequent node installations are started by choosing the New SQL Server Failover Cluster Installation and Add Node to a SQL Server Failover Cluster options on the Installation tab of the SQL Server Installation Center.

In contrast to the one-node-at-a-time approach of the integrated installation, the advanced installation prepares multiple-cluster nodes in one step, before completing the installation on the node chosen as the initial active node for the instance. As

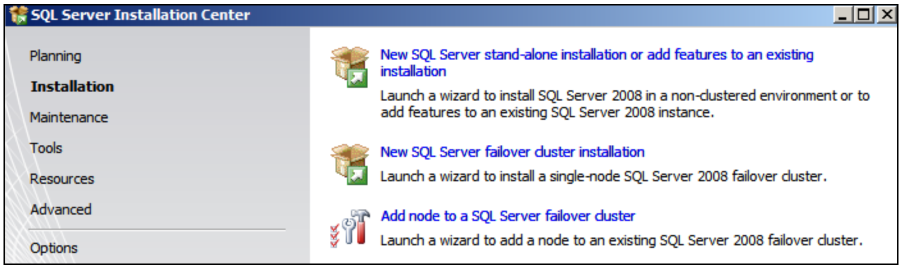


Clear the check box if you do not want a node to host this resource or this clustered instance.

Possible Owners:

☑ Sales1
☑ Sales2

Figure 5.6 The Possible Owners setting gives you control over the cluster failover process.


SQL Server Installation Center

New SQL Server stand-alone installation or add features to an existing installation

Launch a wizard to install SQL Server 2008 in a non-clustered environment or to add features to an existing SQL Server 2008 instance.

New SQL Server failover cluster installation

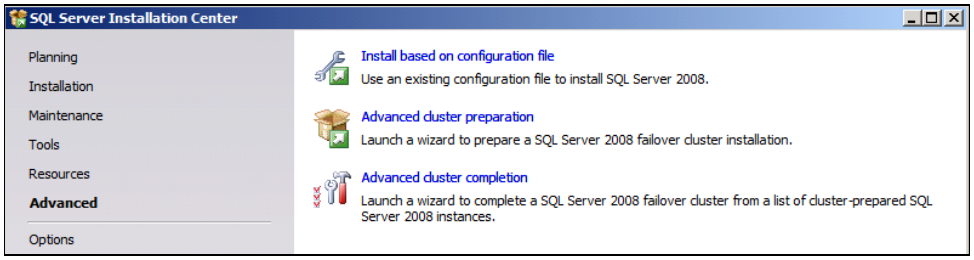
Launch a wizard to install a single-node SQL Server 2008 failover cluster.

Add node to a SQL Server failover cluster

Launch a wizard to add a node to an existing SQL Server 2008 failover cluster.

Figure 5.7 Choose the New SQL Server Failover Cluster Installation option to create a single-node failover cluster before adding additional nodes with the Add Node to a SQL Server Failover Cluster option.

shown in figure 5.8, you specify this installation type by selecting the Advanced Cluster Preparation and Advanced Cluster Completion options on the Advanced tab of the SQL Server Installation Center.


SQL Server Installation Center

Install based on configuration file

Use an existing configuration file to install SQL Server 2008.

Advanced cluster preparation

Launch a wizard to prepare a SQL Server 2008 failover cluster installation.

Advanced cluster completion

Launch a wizard to complete a SQL Server 2008 failover cluster from a list of cluster-prepared SQL Server 2008 instances.

Figure 5.8 Choose the Advanced Cluster Preparation and Advanced Cluster Completion options to streamline the process of installing a clustered SQL Server instance on multiple-cluster nodes.

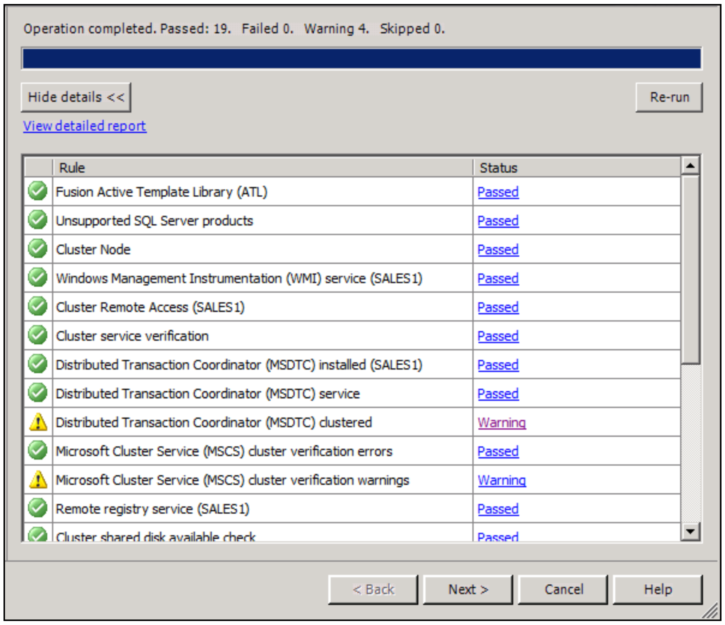
In the previous chapter, we discussed the installation of a nonclustered SQL Server instance. Clustered installations share some of the same installation screens and steps, so rather than repeat them, let's walk through the steps unique to a clustered installation using the integrated method.

5.3.2 *Integrated installation steps*

As with a nonclustered installation, you begin a failover clustering installation by running setup.exe from the installation DVD. Next, you go through a series of steps to install setup support files and check various setup rules. As shown in figure 5.9, the setup checks for a clustered installation are more detailed than for a nonclustered installation.

Installation continues with the usual prompts for a product key, acknowledgment of license terms, and feature selection, before arriving at the instance configuration step, as shown in figure 5.10.

The one difference between this step and the equivalent step in a nonclustered installation is the SQL Server Network Name field. The name you enter is used to identify an instance on the network. In our example, we'll use BNE-SQL-PR-02 as our network name, and together with the instance name (Marketing), we'll access this



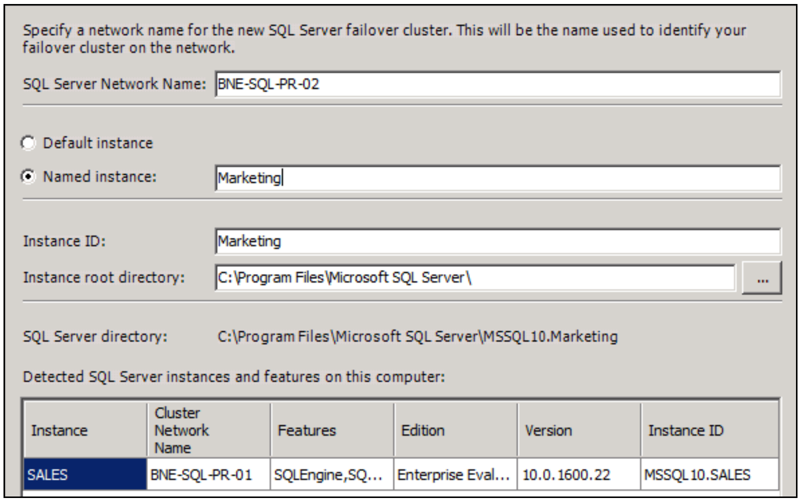
Operation completed. Passed: 19. Failed 0. Warning 4. Skipped 0.

	Rule	Status
✓	Fusion Active Template Library (ATL)	Passed
✓	Unsupported SQL Server products	Passed
✓	Cluster Node	Passed
✓	Windows Management Instrumentation (WMI) service (SALES1)	Passed
✓	Cluster Remote Access (SALES1)	Passed
✓	Cluster service verification	Passed
✓	Distributed Transaction Coordinator (MSDTC) installed (SALES1)	Passed
✓	Distributed Transaction Coordinator (MSDTC) service	Passed
⚠	Distributed Transaction Coordinator (MSDTC) clustered	Warning
✓	Microsoft Cluster Service (MSCS) cluster verification errors	Passed
⚠	Microsoft Cluster Service (MSCS) cluster verification warnings	Warning
✓	Remote registry service (SALES1)	Passed
✓	Cluster shared disk available check	Passed

Figure 5.9 The Setup Support Rules for a clustered installation include cluster-specific checks such as the existence of a clustered Microsoft Distributed Transaction Coordinator (MSDTC) service.

instance as BNE-SQL-PR-02\Marketing without ever needing to know which of the two cluster nodes the instance is running on.

Installation continues through the disk space requirements check before prompting for a cluster resource group, as shown in figure 5.11. The resource group name is used as a container for holding the resources (disks, IP addresses, and services) for the installed instance. Later in this chapter, we'll see the resource group in the Cluster



Specify a network name for the new SQL Server failover cluster. This will be the name used to identify your failover cluster on the network.

SQL Server Network Name: BNE-SQL-PR-02

Default instance

[✓ Named instance:] Marketing

Instance ID: Marketing

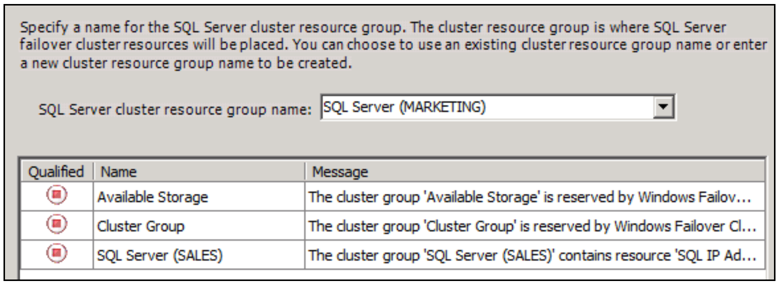
Instance root directory: C:\Program Files\Microsoft SQL Server\

SQL Server directory: C:\Program Files\Microsoft SQL Server\MSSQL10.Marketing

Detected SQL Server instances and features on this computer:

Instance	Cluster Network Name	Features	Edition	Version	Instance ID
SALES	BNE-SQL-PR-01	SQLEngine,SQ...	Enterprise Eval...	10.0.1600.22	MSSQL10.SALES

Figure 5.10 A clustered SQL Server installation is identified on the network with a unique network name you specify during installation.



Specify a name for the SQL Server cluster resource group. The cluster resource group is where SQL Server failover cluster resources will be placed. You can choose to use an existing cluster resource group name or enter a new cluster resource group name to be created.

SQL Server cluster resource group name: SQL Server (MARKETING)

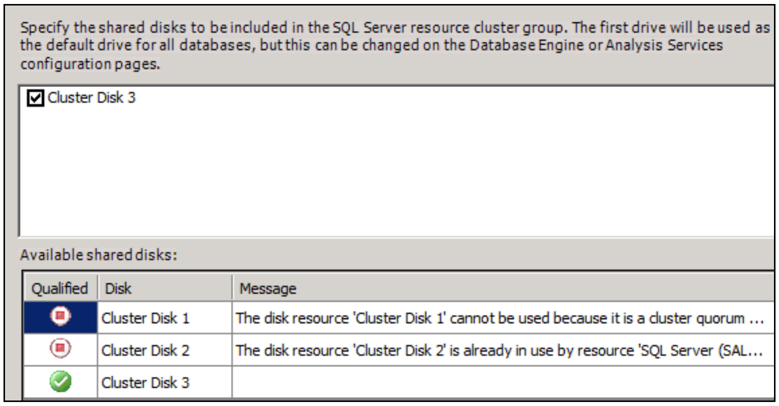
Qualified	Name	Message
⛔	Available Storage	The cluster group 'Available Storage' is reserved by Windows Failov...
⛔	Cluster Group	The cluster group 'Cluster Group' is reserved by Windows Failover Cl...
⛔	SQL Server (SALES)	The cluster group 'SQL Server (SALES)' contains resource 'SQL IP Ad...

Figure 5.11 The cluster resource group name is used to both identify and group the instance's resources.

Management tool, and you'll see how to move the group to another cluster node to effect a planned failover.

In the next step, you'll identify available cluster disk resources that can be chosen for inclusion in the instance's resource group. As shown in figure 5.12, the quorum disk and cluster disks that have been previously assigned to another clustered instance are unavailable for selection.

As shown in figure 5.13, the next step lets you specify either a static or DHCP-based IP address for the SQL Server instance.



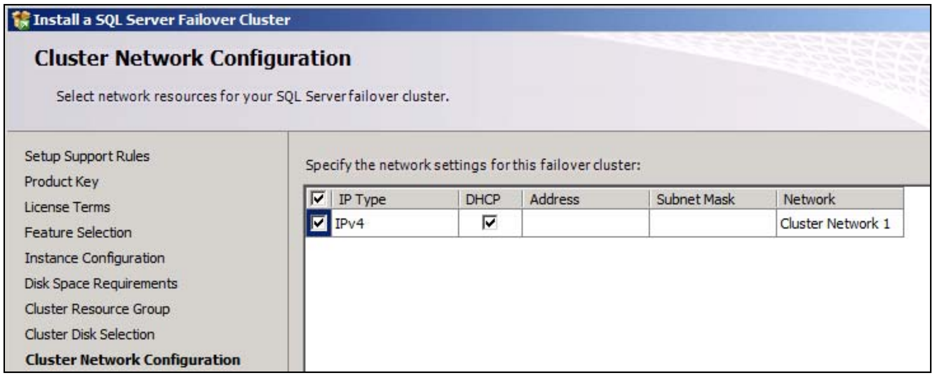
Specify the shared disks to be included in the SQL Server resource cluster group. The first drive will be used as the default drive for all databases, but this can be changed on the Database Engine or Analysis Services configuration pages.

[✓] Cluster Disk 3

Available shared disks:

Qualified	Disk	Message
⛔	Cluster Disk 1	The disk resource 'Cluster Disk 1' cannot be used because it is a cluster quorum ...
⛔	Cluster Disk 2	The disk resource 'Cluster Disk 2' is already in use by resource 'SQL Server (SAL...
✓	Cluster Disk 3	

Figure 5.12 On this screen, you identify available cluster disks for assignment to the instance's resource group.



Install a SQL Server Failover Cluster

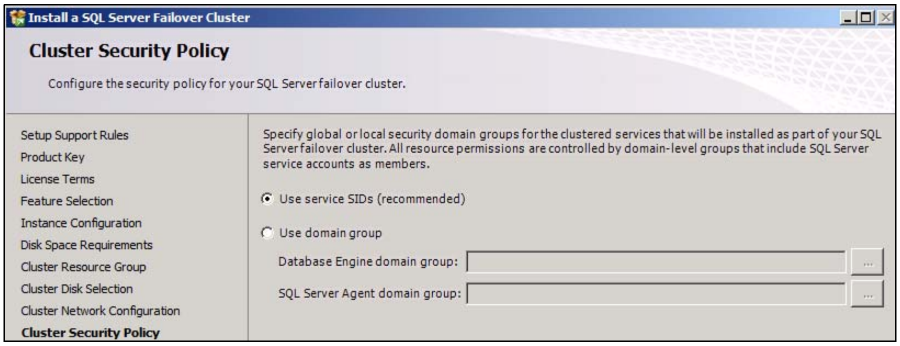
Cluster Network Configuration

Select network resources for your SQL Server failover cluster.

Specify the network settings for this failover cluster:

✓	IP Type	DHCP	Address	Subnet Mask	Network
✓	IPv4	✓			Cluster Network 1

Figure 5.13 Unlike earlier versions, SQL Server 2008 permits DHCP-assigned IP addresses.



Cluster Security Policy

Configure the security policy for your SQL Server failover cluster.

Specify global or local security domain groups for the clustered services that will be installed as part of your SQL Server failover cluster. All resource permissions are controlled by domain-level groups that include SQL Server service accounts as members.

[✓ Use service SIDs (recommended)]

Use domain group

Database Engine domain group:

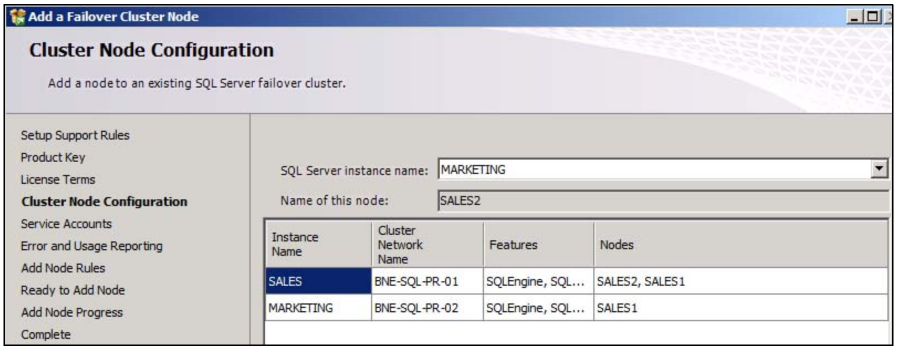
SQL Server Agent domain group:

Figure 5.14 The Cluster Security Policy tab lets you select between the default service SIDs and custom domain membership.

The only remaining cluster-specific installation step is configuring the Cluster Security Policy. In previous versions of SQL Server, the service accounts had to be added to newly created domain groups prior to installation. Permissions for the service accounts were then managed at the domain group level. This requirement was often misunderstood, and introduced complexities when the domain groups needed to be changed. In response to this, SQL Server 2008 introduced an alternative method that uses service security identifiers (SIDs).

As you can see in figure 5.14, using SIDs is the recommended configuration, although support for the old domain group method remains.

The remaining steps in the installation process are the same as for the nonclustered installation described in the previous chapter. At the end of the installation, the clustered instance will be created and available, but can't fail over to other cluster nodes until you run the Add Node installation on the appropriate nodes. This installation option is used to enable additional cluster nodes to participate as failover nodes for an existing SQL Server failover clustering instance. Figure 5.15 shows one of the screens from this installation option, and in this case, we've chosen to allow the SALES2 server to host the MARKETING instance.



Cluster Node Configuration

Add a node to an existing SQL Server failover cluster.

SQL Server instance name: MARKETING

Name of this node: SALES2

Instance Name	Cluster Network Name	Features	Nodes
SALES	BNE-SQL-PR-01	SQLEngine, SQL...	SALES2, SALES1
MARKETING	BNE-SQL-PR-02	SQLEngine, SQL...	SALES1

Figure 5.15 In this example, the Sales2 server is installed as a failover participant for the Sales instance installed on the Sales1 server.

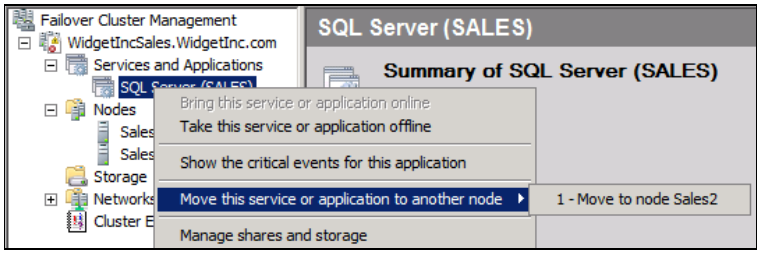

Failover Cluster Management
WidgetIncSales.WidgetInc.com
Services and Applications
SQL Server (SALES)
Nodes
Sales
Sales
Storage
Networks
Cluster E
SQL Server (SALES)
Summary of SQL Server (SALES)
Bring this service or application online
Take this service or application offline
Show the critical events for this application
Move this service or application to another node
1 - Move to node Sales2
Manage shares and storage

Figure 5.16 Use the Cluster Management tool to view and manage a clustered SQL Server instance's resources and state.

When installation is complete, you can manage the clustered instance using the Failover Cluster Management tool in the Administrative Tools folder, or by running Cluadmin.msc from the Start menu. In the example in figure 5.16, you can manually move, or fail over, a clustered instance to another cluster node by right-clicking the resource group and selecting the "Move this service or application to another node" option.

Failover clustering is a complex area requiring a range of skills for a successful implementation. I encourage you to visit http://www.sqlCrunch.com/clustering for links to various clustering-related resources, including how clusters use public and private networks and how to use clustering over geographic distances.

5.4 Best practice considerations: failover clustering

Clustering SQL Server allows for the creation of highly available and reliable environments; however, failing to adequately prepare for a clustered installation can be counterproductive in that regard, with poorly planned and configured clusters actually *reducing* the availability of SQL Server.

- While multi-instance clusters are more cost-effective than single-instance clusters, give careful consideration to the possibility (and implications) of a resource crunch in the event of failover.
- N+1/M clusters offer both cost benefits and resource flexibility, but take into account the failover rules, particularly for clusters containing many servers and SQL Server instances.
- Before installing a SQL Server failover clustering instance, ensure the MSDTC service is created as a clustered resource in its own resource group with its own disk resource. In Windows Server 2003 and earlier clusters, it was common for the MSDTC resource to be configured as part of the quorum group. Certain applications, such as high-throughput BizTalk applications, make heavy use of the MSDTC resource. Insulating MSDTC from the quorum helps to prevent cluster failures due to quorum disk timeouts.
- Windows Server 2008 allows multiple clustered DTC instances to be installed. In such clusters, consider installing a clustered DTC instance *for each* SQL Server instance that requires DTC services. Such a configuration enhances the load balancing of DTC traffic.
- Like nonclustered SQL Servers, a clustered SQL Server node shouldn't be a domain controller, or run any other server applications such as Microsoft Exchange.

- Before installing a SQL Server failover clustering instance, run the Cluster Validation Wizard to ensure the validity of the cluster components.
- All aspects of cluster nodes should be configured identically, including hardware components and configuration, operating system versions and service packs, bios and firmware version, network card settings, directory names, and so forth. Such a configuration provides the best chance of continued smooth operations in the event of a failover.
- Antivirus (AV) software should either not be installed on clusters or configured to not scan any database or quorum disk files. A frequent cause of cluster failures is AV software scanning quorum files. If you're using such software, ensure it's cluster aware, and explicitly exclude all quorum files from all scan types, including on-access and scheduled scans.
- When installing a clustered SQL Server instance, set the service startup types to Manual (which is the default setting) to enable the cluster to stop and start services as required on the appropriate cluster node. The Control Panel Services applet should *not* be used in clusters for stopping or starting SQL Server services. If an instance needs to be taken offline (or moved to another node), use the Failover Cluster Management tool in the Administrative Tools folder or run Cluadmin.msc from the Start menu.
- When installing a clustered SQL Server instance, ensure the account used for the installation is a local administrator on all the cluster nodes the instance will be set up on and ensure any remote desktop connections are disconnected other than the node the installation is occurring on.
- Clustered servers should have at least two network cards, with at least one dedicated to the cluster's private network. Assign to the networks names like Public and Private.
- In the Control Panel, ensure the public LAN is bound first before the private LAN, and remove File/Print Sharing and Client for Microsoft Networks from the private LAN bindings.
- The private network should be physically separate from the public network using a cross-over cable (for two-node clusters), a dedicated hub, or a virtual LAN (VLAN).
- Define the private network at the highest level in the cluster network priority.
- The private network must not have any WINS, DNS, or NetBIOS settings enabled, and should use TCP/IP as the only protocol.
- Use NIC teaming in clusters with caution. There are documented cases of known issues with this approach, and Microsoft doesn't recommend or support NIC teaming for the private cluster network.

Additional information on the best practices covered in this chapter can be found online at http://www.sqlCrunch.com/clustering.

The last five chapters have been focused on planning and installation tasks. Let's move on now and look at post-installation configuration tasks, beginning with the next chapter, where we'll focus on security.

SQL SERVER/DATABASE

SQL Server 2008 Administration IN ACTION

Rod Colledge • Foreword by Kevin Kline

Free ebook
SEE INSERT

Ensuring databases are secure, available, reliable, and recoverable are core DBA responsibilities. This is a uniquely practical book that drills down into techniques, procedures, and practices that will keep your databases running like clockwork.

Open **SQL Server 2008 Administration in Action** and find sharply focused and practical coverage of

- Selecting and configuring server components
  - Configuring RAID arrays
  - Working with SANs and NUMA hardware
- New features in SQL Server 2008
  - Policy-based management
  - Resource Governor
  - Management Data Warehouse
- And much more!
  - Performance tuning techniques
  - Index design and maintenance
  - SQL Server clustering and database mirroring
  - Backup and restore

The techniques and practices covered in this book are drawn from years of experience in some of the world's most demanding SQL Server environments. It covers new features of SQL Server 2008 in depth. Its best practices apply to all SQL Server versions.

Rod Colledge is an SQL Server consultant based in Brisbane, Australia, and founder of sqlCrunch.com, a site specializing in SQL Server best practices. He's a frequent speaker at SQL Server user groups and conferences.

"Simply loaded with excellent and immediately useful information."
—From the Foreword by Kevin Kline, Technical Strategy Manager, Quest Software

"I thought I knew SQL Server until I read this book."
—Tariq Ahmed, coauthor of *Flex 4 in Action*

"A refreshing database administration book."
—Michael Redman, Principal Consultant (SQL Server), Microsoft

"Required for any MS DBA."
—Andrew Siemer, Architect, OTX Research

"It delivered way beyond my expectations... Packed with useful enterprise-level knowledge."
—Darren Neimke, Author of *ASP.NET 2.0 Web Parts in Action*

For online access to the author, code samples, and a free ebook for owners of this book, go to:
www.manning.com/SQLServer2008AdministrationinAction

ISBN 13: 978-1-933988-72-6
ISBN 10: 1-933988-72-X
54499
9 781933 988726

MANNING $44.99 / Can $56.99 [INCLUDING eBOOK]