SQL Server technical eBook series

Microsoft

Drive business continuity with SQL Server

Drive business continuity with SQL Server

Content

About this book

Throughout this book, certain common, industry-specific terms are used frequently in the context of database platforms. For the purpose of our discussion, the following definitions are used:

A server is a physical or virtual machine hosting SQL Server software complete with an operating system.

A cluster is a set of servers grouped together to provide redundancy. The cluster manager monitors the health and responsiveness of the cluster members (nodes). On Windows Server, the manager is called the WSFC (Windows Server Failover Cluster [manager]). Linux distributions use Pacemaker as a cluster manager. A cluster is managed at the operating system level.

A node is a member of a failover cluster. Each node in the cluster has the same capability to answer requests and has access to the same data as every other node in the cluster. Each node must be available to answer requests in a timely manner to be considered 'healthy' by the cluster manager.

A database is a data structure that stores application data and metadata in physical files on disk.

An instance is a collection of SQL Server databases, jobs and so forth run by a single SQL Server service that runs in memory on a specific computer at a specific time. An instance is accessed using a single IP address and all requests are sent to that IP. The source of the request doesn't know which physical location the response it receives is coming from.

A replica (or database replica) is a set of two or more SQL Server instances that are grouped together in a failover configuration. In other words, a replica is – at the database level – what a cluster is at the operating system level.

Who should read this eBook?

This eBook is for database architects, administrators and developers looking to sustain up time for their mission-critical applications. By reading this eBook, you'll learn how SQL Server can help you achieve high availability and disaster recovery goals with mirroring and clustering technology integrated with features on both Linux- and Windows-based deployments. This eBook covers tools and features like Always On Failover Cluster Instances, Always On Availability Groups and log shipping, and provides technical details on how to put these capabilities into practice.

We live in a data-driven world where businesses and end users consume and generate information in numerous ways.

They expect round-the-clock and speedy access to data wherever they are. With this higher demand for data and a more global economy, ensuring uninterrupted and quick access to data is an important concern for organisations. Inaccessibility significantly affects the ability to service customers or carry out day-to-day processes.

Sustaining uptime for your mission-critical applications is a must in today's world that's always online. The way to achieve this is by minimising unplanned downtime. There could be several causes of downtime: hardware/software maintenance upgrades, network or power outages, hardware failure, security breach or cyberattack (due to hacking, virus and so forth). To minimise the downtime and impact on your business, you need a backup and disaster recovery solution in place – to keep your business up and running with minimal or no interruption.

Everyone in the IT industry is familiar with the concepts of high availability (HA) and disaster recovery (DR). Briefly, HA is about implementing a solution that keeps your business running in the event of a catastrophic failure. It masks the effects of a hardware or software failure and maintains the availability of applications, so that the perceived downtime for users is minimised. DR is about implementing solutions that give you the ability to overcome data loss after disaster. A highly available system helps ensure access to your data. However, you'll also want to implement a DR solution to maintain data accuracy.

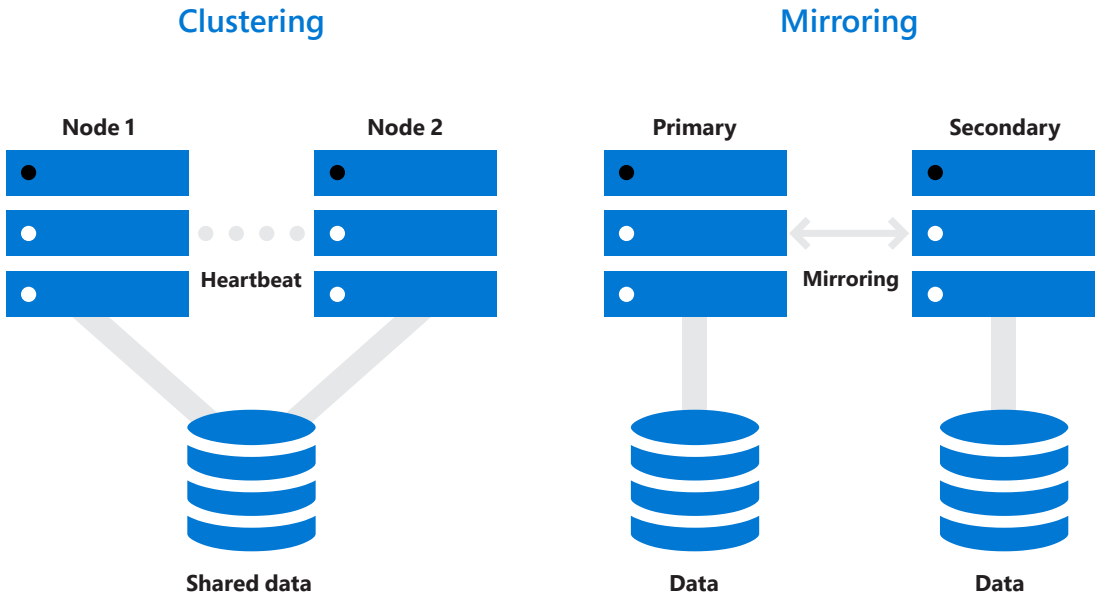
Mirroring is deprecated and will be removed in a future version of SQL Server. Always On Availability Groups improve on this functionality and should be used going forward.

High availability and disaster recovery (HADR) solutions for business continuity

The two most common techniques for achieving high availability and business continuity are mirroring and clustering. Mirroring is a database-level continuity solution while clustering provides a server-level solution.

Database mirroring supports near-instant failover by maintaining a standby database – a full copy (mirror) of the active database on separate hardware. It can operate in a synchronous (high safety) mode, where an incoming transaction is committed to all servers at the same time, or in an asynchronous (high performance) mode, where an incoming transaction is committed to the active database and then (at some pre-determined point) copied over to the mirror. In other words, each mirror maintains its own separate data source. Mirroring is a database-level solution and works only with databases that use the full recovery model.

Database clustering is the process of combining more than one server (which share a single data storage) into what looks to the user like a single instance. Users connect to the instance and never need to know which server in the instance is currently active. If one server fails or needs to be taken offline for maintenance, user experience doesn't change. Each server in the cluster is monitored by the cluster manager using a heartbeat, so it detects when the active server in the cluster goes offline and attempts to seamlessly switch to the next server in the cluster (although there is a variable time delay as the switch happens). Since data is shared between all nodes in the cluster, data continuity is preserved. ■



Microsoft SQL Server is designed to support HADR, ensuring that vital business infrastructure components are always on, available and can survive without any failure.

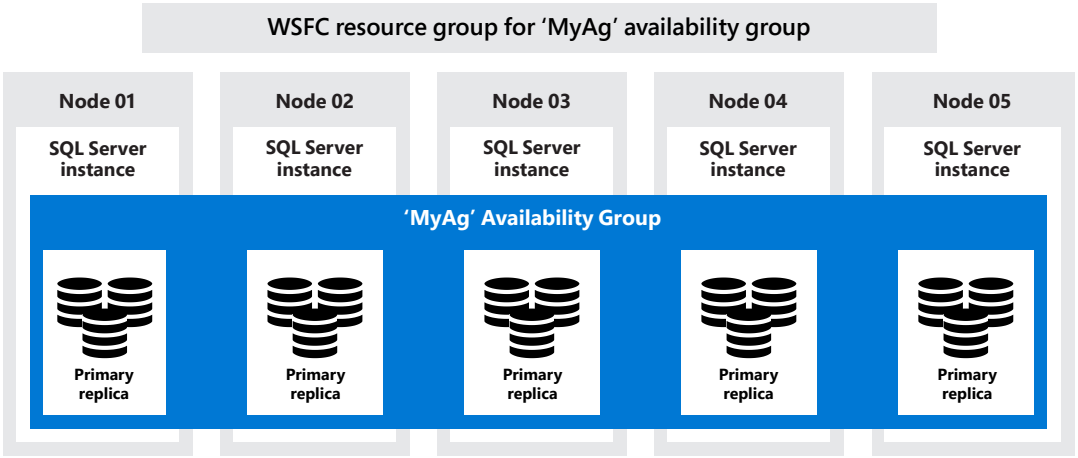
SQL Server offers solutions for various HADR scenarios and comes with a set of features and capabilities that can help organisations achieve a wide range of availability SLAs. With SQL Server, you can have higher security, reliability, scalability and more options for better cluster and storage management. It integrates the mirroring and clustering technology with the features available on both Linux and Windows-based deployments – Always On Failover Cluster Instances (FCIs), Always On Availability Groups and log shipping.

Always On failover cluster instances for instance-level high availability

Always On FCIs are high-availability and disaster recovery tools that provide business continuity on an instance or server level. FCIs protect against server failure due to networking, hardware, operating system or software issues. Because the FCI uses a single network address to receive requests, the user or application making the request will not have to change connection information. If the primary server in a FCI loses power for example, the cluster manager simply 'promotes' one of the secondary servers within the cluster to the primary position and re-routes the request to it instead. The non-responsive cluster member will not receive requests until it comes back online.

Always On Availability Groups for database-level high availability

Always On Availability Groups are an enterprise-level high-availability and disaster recovery solution that enable you to maximise availability for one or more user databases. An availability group is one or more groups of user databases that fail over to a backup together. Always On Availability Groups require that the SQL Server instances reside on cluster nodes.



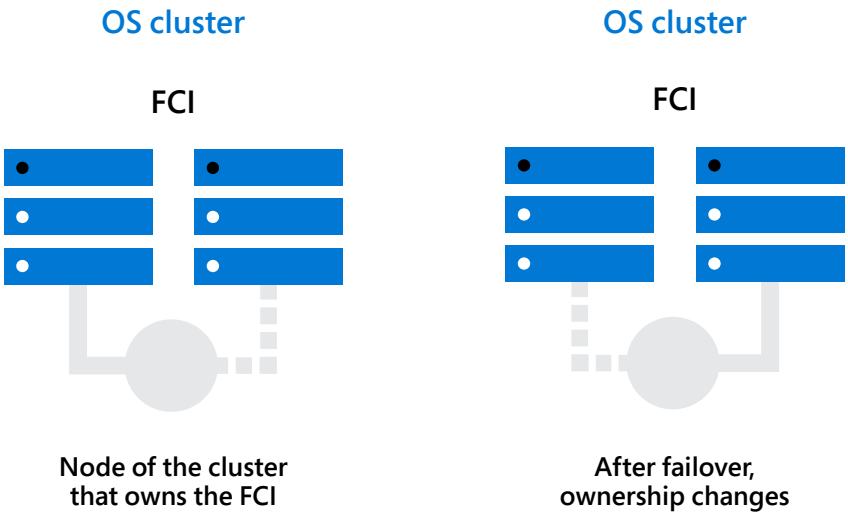
Comparing failover on instance and database levels

With failover at the instance level using SQL Server, an FCI has a dependency on the shared storage. In contrast, failover at the database level using Always On Availability Groups has no dependency on the shared storage. There are more conceptual differences between failover at the instance level and failover at the database level, as shown in the table below:

	Instance level	Database level
Uses cluster	**Yes**	**Yes**
Protection level	**Instance**	**Database**
Storage type	**Shared**	**Non-shared** While the replicas in an availability group don't share storage, a replica that's hosted by an FCI uses a shared storage solution as required by that FCI. The storage solution is shared only by nodes within the FCI and not between AG replicas.
Storage solutions	**Direct attached, SAN, mount points, CIFS**	**Depends on the node type**
Failover resources	**Server, instance and database**	**Database only**

FCIs are a proven method of providing availability for the entire installation of SQL Server, known as an instance.

A failover cluster is made up of two or more identical copies of an instance. This means that there is a duplicate copy of everything inside the instance – including databases, SQL Server Agent jobs, linked servers – on each node. Only one node can be active at a time. The others are passive. Ownership of the instance will move to one of the passive servers should the active server encounter a problem. All FCIs require some sort of shared storage, even if it's provided via networking.

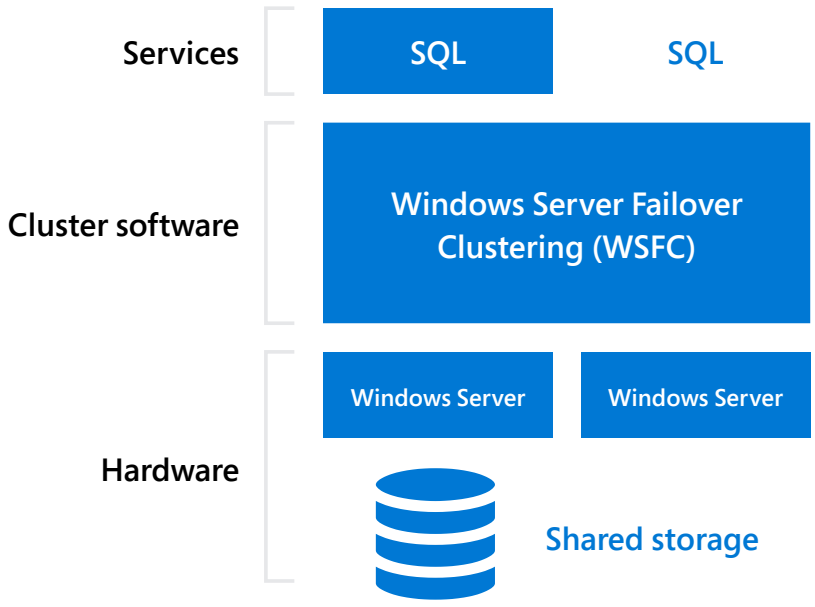


An FCI requires a cluster manager to marshal cluster resources. SQL Server FCIs are available on both Windows and Linux. On Windows, FCIs use Windows Server Failover Clustering (WSFC) as the cluster manager. Linux uses an open source cluster manager called Pacemaker in tandem with an open source server communications system called Corosync. No matter which operating system is being used, these tools broadly perform the same function – creating an active/passive cluster configuration.

Failover cluster instances: SQL Server on Windows

FCIs on Windows leverage WSFC functionality to provide local high availability through redundancy at the instance level. WSFC provides infrastructure features that support the HADR scenarios of SQL Server. It monitors and manages the current roles of the availability replicas that belong to a given availability group while determining how a failover event affects the availability replicas. A WSFC resource group is created for every availability group that you create. The WSFC monitors this resource group to evaluate the health of the primary replica.

With FCI, there's always one master or active node in the cluster. The rest of the nodes become secondary or passive nodes. A SQL Server FCI runs in a WSFC resource group. Each node in the resource group maintains a synchronised copy of the configuration settings and check-pointed registry keys to ensure full functionality of the FCI after a failover. Only one of the nodes in the cluster owns the resource group at a time. In case of a failure, the resource group ownership is moved to another WSFC node. This process is transparent to the client or application connecting to SQL Server. This minimises the downtime the application or clients experience during a failure.



Domain-joined servers

On prior versions of Windows Server, creating the FCI on WSFC required Active Directory. Deploying a WSFC through Windows Server has always necessitated that the nodes participating in a WSFC are joined to the same Active Directory domain. When a WSFC is created, a cluster name object (CNO) or account and a corresponding virtual computer object (VCO) are generated in Active Directory. However, this dependency between a WSFC and Active Directory was the main challenge and roadblock when deploying and managing FCI. To overcome this issue, SQL Server introduced domain-independent failover clusters that enable administrators to deploy a WSFC without an Active Directory domain.

WSFC storage configurations

An FCI must use shared storage between all nodes of the FCI for database and log storage. Each of the FCI nodes in WSFC requires shared storage as per standard SQL Server failover cluster instance installation. The storage can use Fibre Channel, iSCSI, FCoE or SAS for shared disk storage, or locally attached storage with Storage Spaces Direct (S2D). With this shared storage, all nodes in the FCI have the same view of instance data whenever a failover occurs. Shared storage has the potential of being the single point of failure. FCI depends on the underlying storage solution to ensure data protection.

WSFC failover

When the FCI starts up, one of the nodes assumes ownership of the resource group and brings its SQL Server instance online. The resources owned by this node include network name, IP address, shared disks, SQL Server Database Engine service and SQL Server Agent service. At any given time, only the resource group owner (and no other node in the FCI) is running its respective SQL Server services in the resource group. In the event of a failover, the WSFC service transfers ownership of the instance's resources to a designated failover node. The SQL Server instance is then restarted on the failover node and databases are recovered as usual. At any given moment, only a single node in the cluster can host the FCI and underlying resources.

WSFC quorum layer

Each node in a WSFC participates in periodic heartbeat communication to share the node's health status with the other nodes. Healthy nodes are considered 'active' and unresponsive nodes are considered "failed". Quorum is a mechanism that ensures that the WSFC is up and running with enough responsive resources online. If the WSFC has enough 'voting' (active) members, it's healthy and able to provide node-level fault tolerance.

A quorum mode dictates the methodology used for quorum voting and when to perform an automatic failover or take the cluster offline. If the WSFC goes offline – because of an unplanned disaster, due to a persistent hardware issue or as result of a communications failure – manual intervention is required. An administrator will need to force a quorum and then bring the surviving cluster nodes back online in a fault-tolerant configuration.

Failover cluster instances: SQL Server on Linux

On Linux, failover clusters function the same way as on SQL Server, but are structured slightly differently. Instead of the single WSFC manager, SQL Server on Linux uses two components that work together to perform WSFC functions – Pacemaker and Corosync.

Linux is a package-based OS where system services are installed individually or in groups to provide different functions. Three supported Linux distributions – Red Hat Enterprise Linux (RHEL), Ubuntu and the SUSE Linux Enterprise Server (SLES) – include high availability functions that incorporate Pacemaker and Corosync. Pacemaker is the brains of the operation – monitoring cluster members, managing cluster resources and making policy-based decisions about which nodes should be active. Corosync provides dedicated two-way communication between Pacemaker and the nodes it manages.

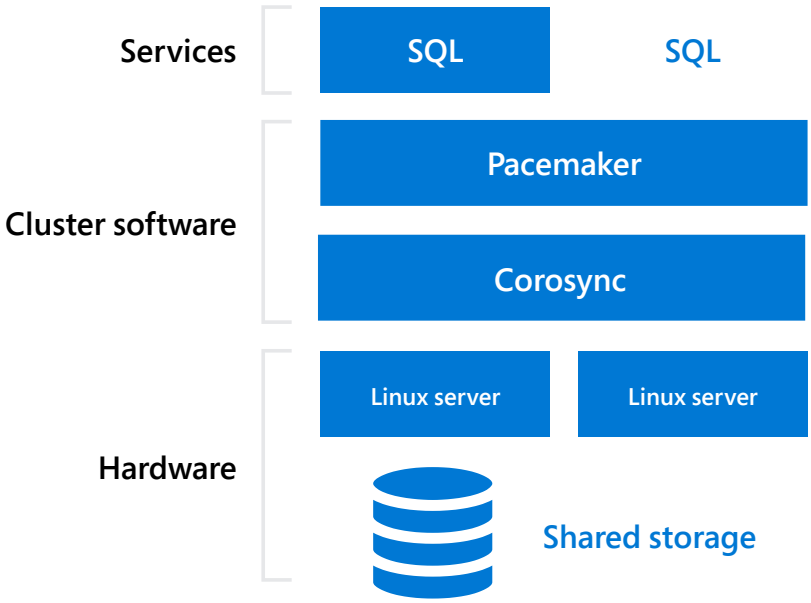
Just like failover clustering on Windows Server, Linux failover clustering provides instance-level continuity, automatic and manual failover, transparent failover for applications and clients and recovery in a matter of minutes or even seconds.

Cluster-shared storage options

All FCIs require some form of shared storage. This storage is presented to all servers that can host the FCI, but only a single server can use the storage for the FCI at any given time. The options available for shared storage under Linux include iSCSI, Network File System (NFS) and Common Internet File System (CIFS). As with any storage resource on Linux, the shared data storage is mounted on each of the nodes of the cluster for the FCI.

Pacemaker failover

In a Pacemaker-managed cluster, as in a WSFC-managed cluster, one of the nodes is considered active while the others are inactive. Only the active node has permission to connect to the shared storage resources. When Pacemaker detects a failure, it switches the active node to a different, healthy cluster member and removes the failed node from the pool of available cluster members, a process known as fencing (discussed in more detail below under Fencing and STONITH).



Pacemaker quorum

Quorum is a generic high-availability term and the way Pacemaker uses it is similar to the WSFC. When more than half of the expected nodes are online, a cluster is said to have quorum. This is important because the quorum configuration determines the number of failed nodes a cluster can have and still remain online. On Linux, the communication layer Corosync manages quorum polling and notifies Pacemaker of status changes. When quorum is lost, Pacemaker has the ability to ignore quorum and sustain cluster services with only one node until quorum is restored. That means services are still available to users. However, a cluster with only one node is not truly highly available. Therefore, Pacemaker will continue to try to bring nodes back online and restore quorum.

Fencing and STONITH

Cluster resource managers – WSFC on Windows or Pacemaker on Linux – monitor the status of cluster nodes and handle any issues that arise. In some circumstances, it can be necessary or prudent to quarantine a node that's misbehaving or not communicating. This quarantining is known in high-availability terms as 'fencing'. Essentially, fencing is a way to control access to resources in a cluster. It can be used at either the resource level (to block access to certain services like a webserver or database server) or at the node level (to prevent a cluster node from becoming 'active'). Fencing can protect the cluster from a misbehaving service or node, or allow you to mark a service or node offline for maintenance even if it's behaving normally.

Removing misbehaving nodes from the cluster (node-level fencing) is called STONITH, which stands for 'Shoot the Other Node in the Head' – in other words, shut it down now. Although there are multiple ways to do this that depend on server hardware and budget, each method immediately power off a node, usually by abruptly cutting the power. STONITH is used to ensure a node can't come back online without human intervention and tells the cluster resource manager (WSFC on Windows or Pacemaker on Linux) to stop trying to fix it.

Predictable failover time

Depending on when your SQL Server instance last performed a checkpoint operation, there can be a substantial number of dirty pages in the buffer cache. Consequently, failovers last as long as it takes to write the remaining dirty pages to disk. This can lead to long and unpredictable failover time.

The FCI can use indirect checkpoints to throttle the number of dirty pages kept in the buffer cache. While this does consume additional resources under regular workload, it makes the failover time more predictable as well as more configurable. This is very useful when the service-level agreement in your organisation specifies the recovery time objective (RTO) for your high availability solution.

Comparing SQL Server FCIs on Windows and Linux

Here are the key differences between FCIs on Windows and FCIs on Linux:

Linux only supports a single installation of SQL Server per server, so all Linux FCIs will be a single-node instance by default. If multiple instances are required on Linux, using containers is recommended. Windows supports more than one installation of SQL Server per server. How many installations depends on which version of Windows you're running, so check your OS for limits.

The number of FCIs that can be handled depends on the OS cluster. Windows supports up to 25 FCIs per WSFC. A Pacemaker cluster can only have up to 16 FCIs because each node is a single instance.

An FCI implemented on Linux with the Standard edition of SQL Server supports a maximum of two nodes per cluster, even though Pacemaker has the ability to support up to 16 nodes.

Failover clusters on containers

Failover clusters using clustered containers are another way to provide high availability running on Linux in a cloud environment. Containers offer nimble deployment – it's quick and easy to spin up containers and then shut them down again when they aren't needed. Compared to virtual machines, which each have the added overhead of a full operating system, containers run on top of an operating system, so you're able to compartmentalise certain applications or services away from it. In general, containers use a configuration script called an image, a template that defines what the container includes. In the event of a failure, the cluster manager can simply use the image to deploy a new container – usually in a matter of seconds.

Azure Kubernetes Service (AKS), with persistent storage for high availability, is one example of using containers in a high availability environment. If the SQL Server instance fails within a Kubernetes container (called a 'pod'), Kubernetes automatically recreates it in a new pod that attaches to the same persistent storage. Depending on the specific pod configuration, this can be a swift recovery solution.

The following example describes a deployment, including a container based on the SQL Server container image with replica:

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: mssql-deployment
spec:
  replicas: 1
  template:
    metadata:
      labels:
---
      containers:
      - name: mssql
        image: microsoft/mssql-server-linux
        ports:
        - containerPort: 1433
        env:
        - name: ACCEPT_EULA
          value: "Y"
        - name: SA_PASSWORD
---
```

Then create the deployment:

```
kubectl apply -f <Path to sqldeployment.yaml file>
```

➔ **Find more information** about deploying Kubernetes in a failover configuration.

Availability groups provide database-level data protection in a primary/secondary (active/passive) configuration.

SQL Server uses the Windows OS WSFC services and capabilities to support Always On Availability Groups and Always On FCIs. Each cluster member (node) is used to determine quorum regardless of whether or not it hosts a database in the availability group.

Each instance participating in an availability group (known as a replica) can be either be configured as standalone or part of an Always On FCI depending on how that replica will be used. SQL Server 2017 supports two different architectures for availability groups: **Always On** and **Read-Scale**.

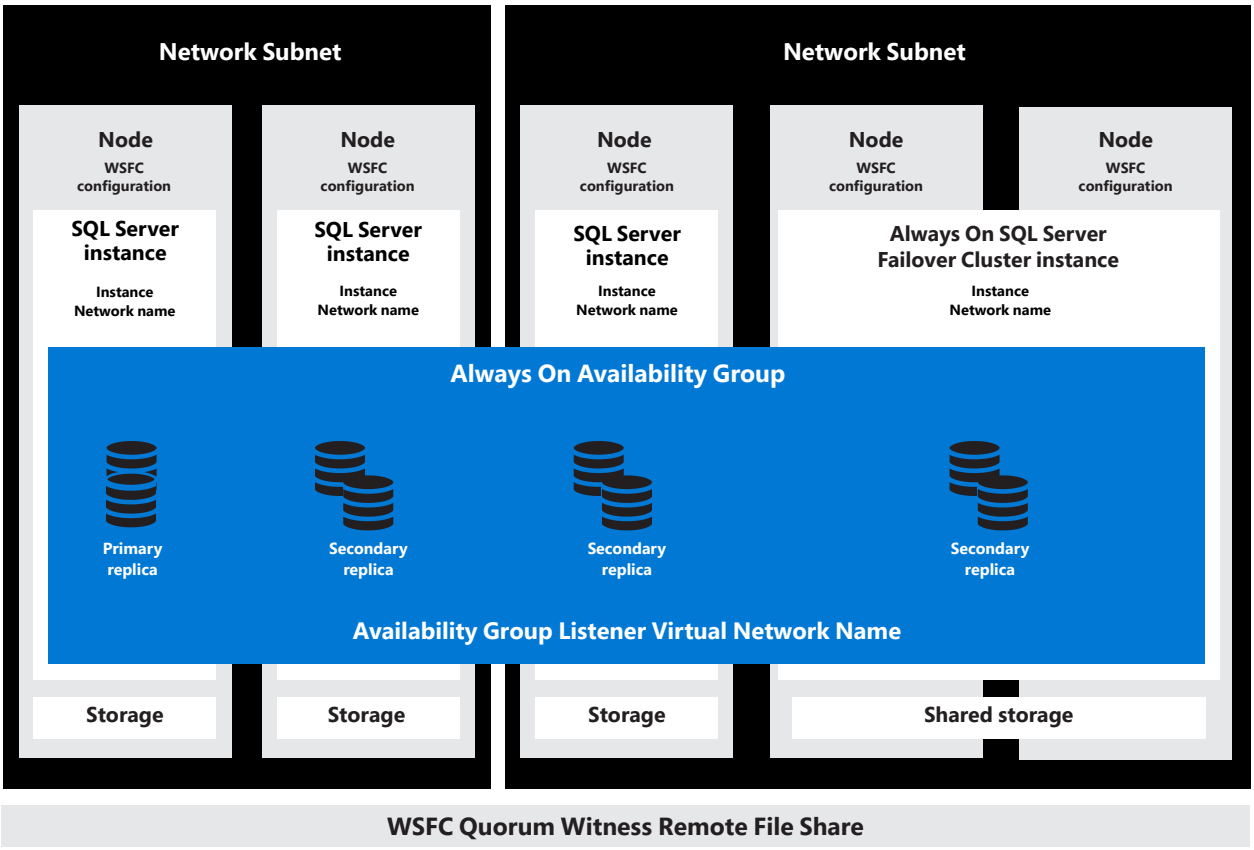
Always On Availability Groups: These availability groups provide database high availability and disaster recovery. They require a server-level cluster manager – either WSFC on Windows or Pacemaker on Linux – which means that they have failure detection, automatic failover and transparent reconnection after failover to provide high availability. This also requires each replica to reside on its own cluster node. Synchronised replicas provide data protection for timely disaster recovery at the database level.

Read-Scale Availability Groups: These availability groups provide data redundancy without being part of a high availability cluster. A Read-Scale Availability Group is a group of databases that are copied to other instances of SQL Server for the read-only workload. These capabilities are available for SQL Server running on both Windows and Linux.

Availability groups with failover clustering

Availability groups build on the high availability of failover clusters to add another layer of fault protection across cluster nodes at the database level. FCIs ensure that SQL Server gives an answer to data requests. Availability groups make sure that the data you're getting back is the most accurate.

Linux also supports availability groups with failover clustering. With a few configuration differences, availability groups for SQL Server on Linux are the same as they are on Windows. Linux uses Pacemaker to control failover clusters and monitor their health where Windows uses WSFC.



Cluster type

Windows has three cluster types: WSFC, External and None. WSFC is the default cluster type on Windows Server and the only cluster type that provides high availability at the server level. If, however, a hybrid operating system cluster environment is desired, the cluster type needs to be set to None.

Linux has two cluster types: External and None. With an External cluster type, Pacemaker controls the FCI underneath the availability group as well as the availability group itself. With a cluster type of None, Pacemaker will not control the availability group even if it controls the FCI. If an availability group has a cluster type of None, failover from a primary replica to a secondary replica will need to be done manually.

Availability mode

Just as failover clusters have a primary instance and corresponding secondary instances, availability groups have a primary replica and corresponding secondary replicas. The primary replica hosts the primary database(s) and allows read/write transactions. The primary replica logs every transaction and sends that log to each secondary replica to keep them in sync. Each secondary replica saves a copy of the log before it applies those logged transactions to its database(s), a process called 'hardening'. This can be done synchronously, where the primary replica waits for the secondary replica to save each transaction log before it sends another, or asynchronously, where the primary replica doesn't wait for the secondary replica to save the logs before it sends another.

Asynchronous-commit mode accelerates the primary replica, but can cause the data integrity of the secondary replica to lag as it tries to catch up with the primary. Synchronous-commit mode can cause slowdowns in the primary replica as it waits for the secondary replica to catch up, but synchronises the secondary's data with the primary replica as much as possible.

Dive deeper: Availability mode set-up – Windows and Linux

With the `CREATE AVAILABILITY GROUP` command for SQL Server on either Windows or Linux, you can specify the availability mode:

```
AVAILABILITY_MODE = { {SYNCHRONOUS_COMMIT | ASYNCHRONOUS_
COMMIT | CONFIGURATION_ONLY }
```

- `SYNCHRONOUS_COMMIT` specifies that the primary replica waits to commit transactions until they have been hardened on this secondary replica (synchronous-commit mode).
- `ASYNCHRONOUS_COMMIT` specifies that the primary replica commits transactions without waiting for this secondary replica to harden the log (synchronous-commit availability mode).
- SQL Server 2017 CU 1 introduces `CONFIGURATION_ONLY`. The `CONFIGURATION_ONLY` replica only applies to availability groups with `CLUSTER_TYPE = EXTERNAL or CLUSTER_TYPE = NONE`. `CONFIGURATION_ONLY` specifies that the primary replica synchronously commit availability group configuration metadata to the master database on this replica. This isn't valid when `CLUSTER_TYPE = WSFC`. For further information, see Microsoft documentation.

➔ **To read more** about these availability modes, go to Availability Modes (Always On Availability Groups).

Failover mode

When a primary replica becomes unresponsive, the WSFC promotes a secondary replica to the primary role in a process called failover. The ways failover can be configured depend on the type of availability mode chosen – synchronous or asynchronous.

Availability mode	Available failover modes
Synchronous-commit	Planned manual failover or automatic failover
Asynchronous-commit	Forced manual failover

Simply put, a planned manual failover is initiated by a command from a database administrator. This command tells the availability cluster to make sure everything from the primary replica is synchronised to the secondary replica(s), promote a synchronised secondary to the primary position, and instruct the cluster to become a secondary replica itself. An automatic failover performs almost the same transition, but occurs in response to a primary replica failure. With an automatic failover, a responsive, synchronised secondary replica is promoted to the primary replica position. When the former primary replica comes back online, it's designated as a secondary replica and synchronised. This secondary replica is not available for failover until it's fully in sync with the new primary replica.

With asynchronous-commit mode, where the primary replica sends transaction logs to the secondary without waiting for those logs to be successfully saved (hardened), the only failover option is a forced manual failover. This failover is initiated immediately, regardless of whether the hardened transaction logs on that secondary replica have been applied to the corresponding database(s). A forced failover puts data integrity at risk and can cause data loss.

Dive deeper: Failover mode on Windows

You can change the failover mode of an availability replica defined in an availability group by using SQL Server Management Studio (SSMS), Transact-SQL (T-SQL) or SQL Server PowerShell. In SSMS, you can use the Failover mode drop-down list in the Availability Replica Properties dialog box to change the failover mode of this replica.

In T-SQL, use the `ALTER AVAILABILITY GROUP` statement to change the failover mode:

```
ALTER AVAILABILITY GROUP *group_name* MODIFY REPLICA ON
‘*server_name*’
  WITH ( {
                  | FAILOVER_MODE = { AUTOMATIC | MANUAL }
        }  )
```

➔ **For more information,** see Change the Availability Mode of an Availability Replica (SQL Server).

Dive deeper: Failover mode on Linux

With Linux, when the `CLUSTER_TYPE` is `EXTERNAL`, the external cluster manager executes all manual or automatic failover actions. For example, if a solution uses Pacemaker to manage a Linux cluster, employ `pcs` to perform manual failovers on Red Hat Enterprise Linux (RHEL) or Ubuntu.

```
sudo pcs resource move ag_cluster-master nodeName2 --master
```

On SUSE Linux Enterprise Server (SLES), use `crm`.

```
crm resource migrate ag_cluster nodeName2
```

➔ **For more information** about Always On Availability Group failover on Linux, refer to Microsoft documentation on manual and forced failover.

Availability group listener

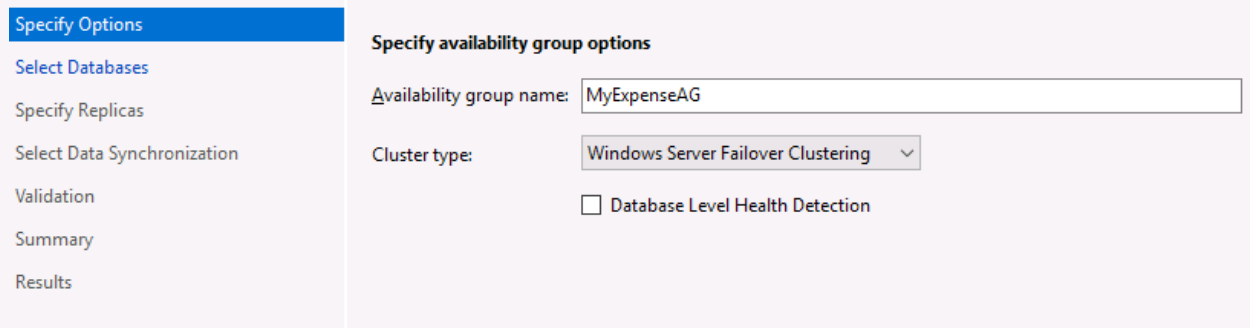
An Always On Availability Group uses a listener service, which allows applications and end users to connect without needing to know which SQL Server instance is hosting the primary replica. Windows accomplishes this using virtual IPs, a virtual network name and DNS configuration. Each availability group has its own listener, but only the primary replica's node responds to incoming client requests to connect to the virtual network name.

As with Windows, the listener is optional functionality for an availability group on Linux. Like an instance in an FCI, an availability group listener provides a single client connection point without requiring the name of the instance. In Linux, there is an IP address resource created in Pacemaker that can run on any of the nodes. An entry associated with the IP resource for the listener in DNS with a 'friendly name' is used. The IP resource for the listener will only be active on the server hosting the primary replica for that availability group.

Dive deeper: Availability group set-up on Windows

To create and set up an availability group in Windows, you can use any of the following tools in SQL Server Management Studio (SSMS), Transact-SQL (T-SQL) and SQL Server PowerShell:

1. **New Availability Group Wizard.** This wizard in SSMS enables you to complete all the tasks required to create and configure an availability group. For example, you can specify the cluster type as WSFC for Windows:



➔ **For more information,** see Use the Availability Group Wizard (SQL Server Management Studio).

2. **T-SQL.** Using the `CREATE AVAILABILITY GROUP` command, specify the cluster type as WSFC highlighted below:

```
CREATE AVAILABILITY GROUP [ag1]
    WITH (DB_FAILOVER = ON, CLUSTER_TYPE = WSFC)
    FOR REPLICA ON
        N'agnode01'
        WITH (
            ENDPOINT_URL = N'tcp://agnode01:1433',
            AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
            FAILOVER_MODE = EXTERNAL,
            SEEDING_MODE = AUTOMATIC
            ),
        N'agnode02'
        WITH (
            ENDPOINT_URL = N'tcp://agnode02:1433',
            AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
            FAILOVER_MODE = EXTERNAL,
            SEEDING_MODE = AUTOMATIC
            );
```

➔ **For more information** on the arguments for the `CREATE AVAILABILITY GROUP` syntax, see Microsoft documentation.

3. **PowerShell.** Use PowerShell cmdlets to create and configure an Always On Availability Group in SQL Server 2017.

```
# Create the availability group
New-SqlAvailabilityGroup `
    -Name "MyAg" `
    -Path "SQLSERVER:\SQL\PrimaryComputer\Instance" `
    -AvailabilityReplica @($primaryReplica,$secondaryReplica) `
    -Database "MyDatabase"
```

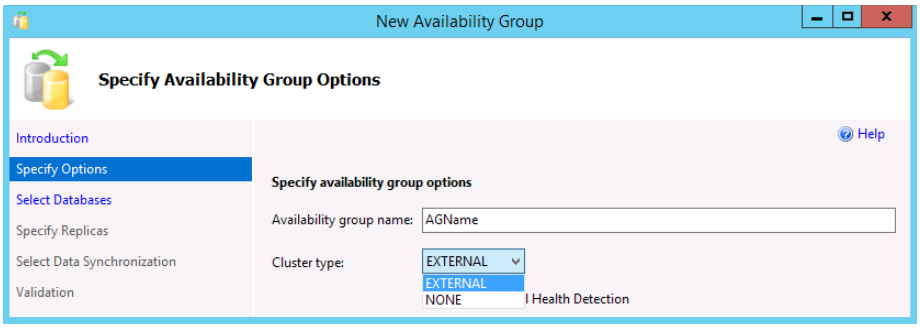
➔ **For more information,** visit Create an Availability Group (SQL Server PowerShell).

Dive deeper: Availability group set-up on Linux

The steps to create an availability group on Linux servers for high availability are different from those on a Windows Server failover cluster. On a Linux FCI, you need to enable availability groups on Linux nodes with endpoints and certificates. Then you can use Transact-SQL or New Availability Group Wizard in SSMS to create an availability group with External cluster type. Pacemaker is an example of an external cluster entity.

```
 CREATE AVAILABILITY GROUP [ag1]
WITH (DB_FAILOVER = ON, CLUSTER_TYPE = EXTERNAL)
FOR REPLICA ON

        ),
```



After an availability group is created in SQL Server, the corresponding resources must be created in Pacemaker, when a cluster type of External is specified. There are two resources associated with an availability group: the group itself and an IP address:

```
sudo pcs resource create ag_cluster ocf:mssql:ag ag_name=ag
--master meta notify=true
```

To ensure that the IP address and the availability group are running on the same node, a colocation constraint must be configured. You can create an ordering constraint to ensure that the availability group is up and running before the IP address. This way, you can create and configure an availability group for SQL Server on Linux.

➔ **For step-by-step guidance,** visit Microsoft documentation on creating and configuring an availability group for SQL Server on Linux.

Hybrid Windows- and Linux-based availability groups

Availability groups can span multiple operating systems – that is, replicas within a hybrid availability group can be running on different operating systems. However, since Windows and Linux use different controllers to manage replica failover (WSFC and Pacemaker), any availability group containing a mix of operating systems needs to use a cluster type of None and therefore requires manual failover.

Dive deeper: Hybrid Windows- and Linux-based availability groups

You can create a hybrid availability group with a cross-platform configuration. For example, one in which Windows Server hosts the primary replica and a Linux server hosts the secondary replica. However, note that while creating the availability group, you should select the cluster type as NONE, because there is no cluster manager:

```
CREATE AVAILABILITY GROUP [ag1]
    WITH (CLUSTER_TYPE = NONE)
    FOR REPLICA ON
        N'<WinSQLInstance>'
     WITH (
        ENDPOINT_URL = N'tcp://<WinSQLInstance>:5022',
        AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
         SEEDING_MODE = AUTOMATIC,
         FAILOVER_MODE = MANUAL,
        SECONDARY_ROLE (ALLOW_CONNECTIONS = ALL)
         ),
        N'<LinuxSQLInstance>'
    WITH (
         ENDPOINT_URL = N'tcp://<LinuxSQLInstance>:5022',
        AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
        SEEDING_MODE = AUTOMATIC,
        FAILOVER_MODE = MANUAL,
        SECONDARY_ROLE (ALLOW_CONNECTIONS = ALL)
        )
GO
```

To join the secondary replica with the availability group, you need to specify the cluster type as External.

```
ALTER AVAILABILITY GROUP [ag1] JOIN WITH (CLUSTER_TYPE = NONE)
ALTER AVAILABILITY GROUP [ag1] GRANT CREATE ANY DATABASE
GO
```

The primary replica allows reads and writes. To change which replica is primary, you can fail over. In an availability group for high availability, the cluster manager automates the failover process. In an availability group with cluster type `NONE`, the failover process is manual. ■

Failover clustering is only one piece of high availability. When your primary availability location experiences a catastrophic event like an earthquake or flood,

the business must be prepared to have its systems come online elsewhere. Features like Always On FCIs, log shipping and Always On Availability Groups can assist with the required business continuity and disaster recovery.

Always On FCIs

Regarding disaster recovery, there's an additional consideration for FCIs: shared storage. The same disks need to be available in the primary and secondary sites. Ensuring this availability requires an external method, such as functionality provided by the storage vendor at the hardware layer. You can also use storage synchronisation like Storage Replica in Windows and rsync or Unison in Linux to make sure that the disks used by the FCI exist elsewhere.

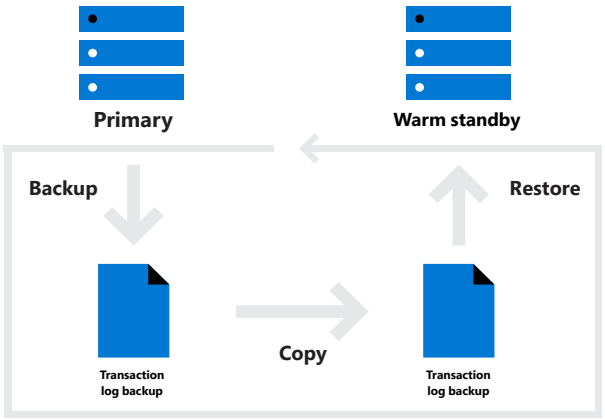
Always On Availability Groups

One of the benefits of Always On Availability Groups is that both high availability and disaster recovery can be configured using a single feature. You're not required to ensure that shared storage is also highly available. It's easier to have replicas that are local in one datacentre for high availability, and remote replicas in other datacentres for disaster recovery, each with separate storage. Having additional copies of the database is the trade-off for ensuring redundancy.

Log shipping

Log shipping is a feature in SQL Server that supports high availability. With log shipping, transaction logs from a database on a primary server are backed up, sent to one or more secondary databases and restored onto those secondary databases (as illustrated in the figure). This keeps secondary databases in sync with the primary. This process is done on an automatic basis at pre-configured intervals to allow transaction rollback. The longer the interval between syncs, the slower failover time will be.

Log shipping is available in both Windows and Linux. On Linux, SQL Server Agent jobs are not part of the base installation of SQL Server. They're added with the mssql-server-agent package, which must be installed to use log shipping. SQL Server on Linux uses the Common Internet File System (CIFS) and the Samba networking protocol service to store the transaction log backups on a network file share. The SQL Server Agent on Linux periodically runs a stored procedure that forwards log backups to the secondary servers. As on Windows, recovering a database that is protected with log shipping is a manual operation. ■



Downtime of critical business applications, data outage and the need to keep up application performance are top concerns for businesses today.

To ensure business continuity in the face of downtime, it's imperative for organisations to consider high availability (HA) and disaster recovery (DR) strategies. Microsoft Windows Server and Microsoft SQL Server provide HA and DR solutions to ensure businesses are always on and available with a rich feature set of availability groups, failover cluster instances and log shipping capabilities. Availability groups can also provide additional copies of a database as part of the same architecture to scale out readable copies. Instances and databases of SQL Server can now be made highly available on both Windows and Linux using the same features.

Learn more about high availability solutions for SQL Server.

Read about Always On failover cluster instances for SQL Server.

Get further insights on Always On Availability Groups for SQL Server.