International Journal of Management, IT & Engineering
Vol. 11 Issue 4, April 2021
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

An Analysisof Disaster Recovery with SQL Server Always On

Mohsin A. Khan

Abstract

Disaster recovery (DR) is one of the most critical and often overlooked aspectsof database administration. A company may have an efficient database design, well-performing code, top-level security. However, it all comes to naught if there is no high availability and disaster recovery solution to meet the Service Level Agreement, Recovery Point Objective, and Recovery Time Objective in a time of disaster. Even if a company has a DR solution in place, more often than not, it is either misunderstood or not understood thoroughly, leading to uncertainty in determining the recovery time and availability for a database post disaster. In this article, the two primary SQL Server high availability and disaster recovery solutions are discussed, focusing on the events that trigger failovers, how failovers are handled, and the components that make up each of the two solutions with some illustrations.

Keywords:

SQL Server;
Disaster recovery;
High availability;
Always On;
Availability Groups;
SQL failover cluster;
Failover.

Author correspondence:

Mohsin A. Khan,
Senior Database Administrator, Elan Technologies Inc. (Client: WellMed, a UnitedHealth Group company)
Ph.D. Student, Computer and Information Sciences,
University of Arkansas at Little Rock, Little Rock, Arkansas-United States
Email: mohsin7.dba@gmail.com

1. Introduction

High Availability and Disaster Recovery are very crucial to any company that generates and works with data. A sudden unplanned outage could cause businesses loss of revenue and reliability. Microsoft SQL Server provides two primary high availability and disaster recovery solutions under the umbrella of Always On — *Failover Cluster Instance* and *Availability Groups*, which protect SQL Server at the server/instance level and database level, respectively. We express data availability or uptime in terms of the number of 9's, which show the percentage of data availability. For example, availability of five-9s means an uptime of 99.999%, calculated as (Actual availability / Desired availability) * 100%. The total annual downtime for an application with an uptime of five-9s is 5 minutes and 15 seconds. The more 9's we add, the more the cost of the system resources and solutions to achieve it.

High availability focuses on making the system, SQL Server, highly available to the business and minimizing the impact of downtime caused by any outages. The second aim of high availability is ensuring the impact of an outage is transparent to business and that the system comes back up as soon as possible. Therefore, the lesser the downtime, the better the availability. We often referto disaster recoveryto mean high availability, but there is a subtle difference. Disaster recovery deals with recovering lost data after a disaster has happened. Always On Availability Groups is widely used to achieve both high availability and disaster recovery.

2. Types of outages

An outage results in application downtime, which might cost a business loss of revenue. It can be planned, which usually occurs during planned maintenance such as patches, software or hardware upgrades, firmware upgrades, service packs, cumulative updates, or hotfixes. By definition, these activities involve planning for downtime in advance, so there is little to no impact on the business. Planned downtime rarely incurs data loss.

On the other hand, an unplanned outage causes the system to shut down, causing an unplanned downtime abruptly. Examples of unplanned outages include hardware failures such as data center failure caused by either artificial or natural disasters like hurricanes, failure at the storage subsystem, NIC adapter or switch failure, storage array controller failure. Software failures such as the accidental shutdown of a server because of human error, data corruption may all lead to unplanned downtime. If not planned carefully in advance, unplanned outages might cause data loss and system unavailability for prolonged hours violating the Service Level Agreement (SLA), Recovery Point Objective (RPO), and Recovery Time Objective (RTO).

Some organizations perform a disaster recovery drill annually or biannually, which, depending on the DR solution and the number of applications and servers, lasts for a day or two. The drill follows all the DR procedures that a company expects to execute in a real disaster. Investing a couple of days a year in a drill pays back hundred times if a real disaster strikes.

3. Windows Server Failover Cluster (WSFC)

Windows Server Failover Cluster is a prerequisite to both Always On solutions. Both failover cluster instance and availability groups leverage and are built on top of a WSFC. SQL Server 2017 introduced Clusterless Availability Groups wherein the replicas do not have a Windows cluster to govern the failovers. Hence there is no automatic failover of AGs, nor is there a listener to route connections to the primary or secondary replica. A workaround for the listener is to use CNAME to point to the primary replica and change it to point to a secondary replica in a failover. That way, clients do not have to change the connection string at all.

A WSFC is a group of servers called nodes that provide maximum availability for the resources, such as SQL Server running on the cluster. All cluster nodes work together as a unit. WSFC interconnects them with each other via a network, storage (iSCSI or Fiber Channel connector) which are all tied by the WSFC software. The clients use a virtual network name (VNN) to point to the SQL failover cluster instance running across a WSFC, making the underlying cluster architecture transparent. It appears as a single server to the clients, but there can be up to 64 nodes in a single Windows cluster under the covers, though rarely deployed.

The crucial areas in a WSFC are servers (nodes), network, storage. Improper configuration of these cluster components may lead to poor performance and affect availability. Suppose any mission-critical applications, such as SQL Server, run on the cluster. In that case, it is strongly recommended to have redundancy at both hardware and software layers to avoid any single point of failure. Redundancy at hardware includes multiple nodes, storage host bus adapters (HBA), SAN switches,multiple NICs via NIC teaming, disk and storage controllers, multiple network adapters for cluster communications, each connected to a separate switch. Redundancy at software includes failover clustering feature, network teaming, multi-path I/O,and the like.

Multiple independent networks connected to all cluster nodes avoid a single point of failure, which otherwise breaks cluster communication links. Depending on the severity, they either trigger a failover or shut down the whole cluster. While having a dedicated network for inter-cluster communication (heartbeat) used to be a requirement in older versions, starting Windows Server 2008, the inter-cluster communications use all the networks. Hence, a dedicated heartbeat network is optional.

A cluster role (formerly, resource group) is a group of resources subjected to fail over from an active node to another node when the active node crashes or the resource in that cluster role fails and we exhaust the restart threshold. A role could be a SQL Server failover cluster instance (FCI) with resources such as shared

storage volumes, file servers, DTC, SQL Server instance, SQL agent that all fail over as a group. SQL instance runs as a resource within a role that is subjected to failover. These resources have dependencies defined among them such that failure of one prevents the other from coming online.

4. What triggers a failover in WSFC?

The hardware and software failures discussed above can lead to cluster heartbeat delays and end up causing failovers. The node-to-node communication that happens via either a dedicated private network or the public network has some threshold settings which determine whether nodes are up and running or if there is a need to trigger a failover. The heartbeat communication uses port 3343 and has a delay of 1 second and a threshold of 10 heartbeats by default. The settings are *SameSubnetDelay* and *SameSubnetThreshold*. The delay value dictates how often the heartbeats are sent between the nodes. The threshold value dictates how many heartbeats the cluster can miss before a recovery action that may be a failover is triggered. By default, if there is no heartbeat for 10 seconds, the WSFC can initiate a failover. For a multi-subnet cluster, the settings are *CrossSubnetDelay* and *CrossSubnetThreshold* with delay as 1 and threshold 20. Windows Server 2016 introduced two new settings, *CrossSiteDelay* and *CrossSiteThreshold*.

It is recommended to bump up the threshold to avoid unnecessary false failovers and be more tolerant to small network hiccups. However, note that the downtime could be more because we have upped the cluster network tolerance in a real disaster.

5. Quorum

Let us talk about quorum, which plays a key role in deciding when to failover. Wikipedia says, "*A quorum is the minimum number of members of a deliberative assembly necessary to conduct the business of that group.*" The same definition holds in WSFC. It is defined as the minimum number of nodes that must be up and running for the cluster to work. The minimum number is more than half of the nodes or (N/2) + 1, where N is the number of voters, i.e., the sum of nodes and a witness. For a six-node cluster, the minimum number of nodes that must be up is 4, and the cluster can survive up to two node failures. Any additional node failure stops the cluster from working.

Each node has a vote in deciding whether a failover is warranted. In a five-node cluster, three nodes must be up and running. Similarly, in a four-node cluster, three nodes (voters) must be present. The minimum number of votes to achieve quorum is the same in both five- and four-node clusters. Thus, as a rule of thumb, there should be an odd number of voters in a WSFC. To stand up a whole new server just for vote count would be overkill, and as a fix, WSFC provides the following options to be counted as votes in a cluster: disk witness, file share witness, cloud witness (starting Windows 2016), or USB file share witness(Windows 2019). We cannot combine all of them, nor can we have more than one (vote) from the same witness type. Starting Windows Server 2019, a file share witness can be any file share as long as it supports SMB 2.0 or above, which means a simple USB device attached to the network router adds an extra vote in the quorum, further simplifying the voting mechanism. Windows Server 2019 also enhances the file share witness by allowing the file share to run on drives without disks and the ability to vote in domain-less or mixed-domain clusters.

6. Quorum models and split-brain scenario

Quorum protects the cluster from a scenario called"*split-brain*" in which when the communication link between the nodes fails, the cluster splits into partitions with a subset of nodes in each partition. Since the partitions cannot communicate with each other due to network failure, if there was no quorum, two nodes in separate partitions could try to bring the resource such as SQL Server, online, which might cause data corruption because of multiple instances servingthe clients. Thankfully, WSFC prevents this split-brain scenario and guarantees that in such a situation, only one node would bring the cluster resource (SQL Server) online. Which node in which partition is determined by the quorum witness, which is accessible to all the nodes. The partition that owns the quorum witness and can communicate with it is allowed to host the resource. The other partition loses quorum, and any resources hosted on it are terminated. The quorum witness must be protected as it acts as the tie-breaker in a split-brain scenario and thus must be stored on a highly available storage subsystem.

WSFC provides the following options for configuring quorum and voting mechanism.

1. Node Majority (no witness): In this quorum configuration, more than half of the nodes (votes) must be online for the cluster to be healthy and running. If not, the cluster cannot operate.

2. Node and File Share Majority: This is like node majority, except a file share witness also gets to vote. The condition of more than half of the votes still applies.

3. Node and Disk Majority: This is like node and file share majority, except instead of a file share, a shared disk witness resource gets to vote in the quorum. The disk must be accessible to all the nodes.

4. Disk Only (No majority): Not recommended. No nodes get to vote. Only shared disk resource witness has a vote in the configuration. With disk witness as the only vote, we expose the cluster to a single point of failure.

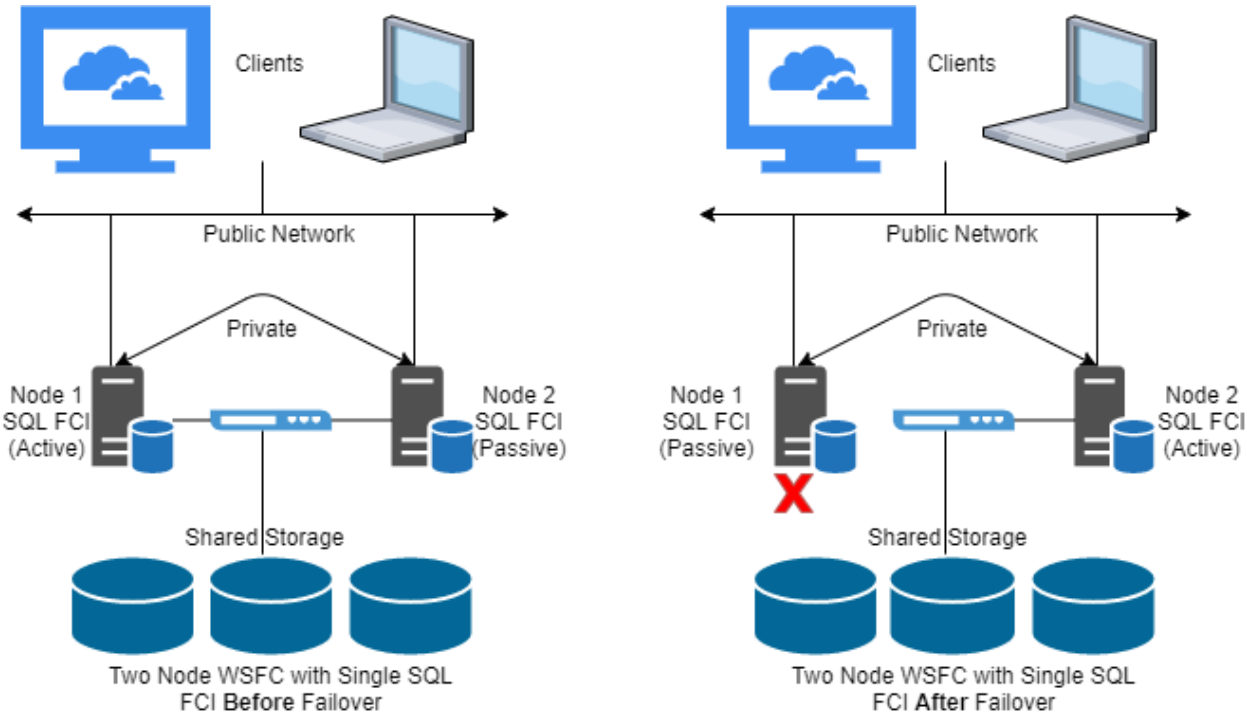
Tip: If a cluster is geographically dispersedor multi-site across multiple subnets, removing a node's vote from the disaster recovery site is generally a good idea. Imagine a multi-subnet cluster that has an odd number of nodes. Adding a file share witness makes a total of an even number of votes, potentially breaking the cluster if the network between the primary and secondary sites is sloppy. If we remove the secondary site's vote, that leaves us an odd number of votes, maintaining quorum.

Windows Server 2012 introduced a dynamic quorum concept that automatically and dynamically adjusts the quorum votes, making it possible to run a cluster with even less than 50% (i.e., no quorum) votes. For example, say we have a five-node cluster. We need at least three nodes up and running for the cluster to operate and sustain up to two node failures. With dynamic quorum, as long as the nodes fail sequentially, WSFC adjusts the quorum dynamically after each node failure, allowing the cluster to undergo three nodes' failures and still be running. With this, the cluster still operates with even one node, often called *Last Man Standing*.

Windows Server 2012 R2 introduced a dynamic witness conceptwhere WSFC automatically adjusts the vote for the witness. It knows when to assign a vote for the witness and when not to. If there are an even number of nodes and a witness, WSFC counts the witness vote as 1, making the total an odd number. Alternatively, if there are an odd number of nodes and a witness, it does not count the witness vote, thus maintaining an odd number of votes. So we can rest assured that adding a quorum witness is safe regardless of the number of nodes since WSFC automatically knows whether or not to consider its vote.

7. Always On SQL Server Failover Cluster Instance (FCI)

SQL Server Always On Failover Cluster Instance, or FCI for short, is an instance-level high availability solution that protects a single SQL instance installed on a Windows failover cluster. For example, say we have a two-node WSFC, we can install SQL FCI as a single instance FCI (active-passive) or multi-instance FCI (active-active). Single instance FCI has the active instance running on node 1 with the passive instance on node 2 sitting idle waiting for a disaster on node 1, which would fail the SQL instance to node 2 (shown in the figure below). On the other hand, a multi-instance SQL FCI has both nodes running an active SQL Server instance on each. However, it is essential to note that the active instances are entirely different with different databases. SQL FCI does not provide load balancing, meaning traffic is always routed to the active instance, with the passive instance staying idle. There is no load distribution across the cluster instances.



The major con of SQL FCI is that it does not provide redundancy at the data/storage level. Since the nodes share the storage, it becomes a single point of failure. We overcome this drawback in Availability Groups. The client (and optionally, heartbeat) connections to the SQL FCI happen via a public network which usually has higher bandwidth.

8. What triggers a failover in SQL FCI?

Failover results in the SQL instance shut down on the active node forcing the existing client connections to terminate and bring up the otherwise passive instance on the other node. Depending on the databases' state when the primary went down, it may take some time for the databases to recover and come online on the new node. The former active node (now dead) joins back to the cluster when it is available again, and at which point, given that failback is enabled for the resource, it results in failing back of the SQL FCI to the original active node. It is recommended to have a connection retry logic in applications. Although connections terminate upon failover, the application may reconnect before the retry attempts time out and the application breaks completely.

SQL FCI relies on WSFC to detect and respond to failovers resulting from hardware and software failures. WSFC is responsible for maintaining the quorum configuration and determining when to trigger a failover. An event such as heartbeat failure due to reaching the subnet threshold (discussed above) is one reason that may cause a WSFC to fail over a cluster resource to another healthy node as long as there is a quorum.

9. Sp_server_diagnostics and Health-Check Timeout

Because SQL FCI is cluster-aware, it works closely with WSFC in communicating its overall health so WSFC can determine if a situation warrants a failover. So WSFC is not alone in controlling failovers. SQL FCI reports events to WSFC, which depending on the severity, may result in a failover. The active SQL FCI periodically reports its health status and diagnostic information to the SQL Server database engine resource DLL through a dedicated connection. The WSFC service uses this DLL to monitor the node's overall health.SQL FCI runs *sp_server_diagnostics* system stored procedure with a repeat interval to report the health status to the resource DLL every 10 seconds. So what does WSFC do with this information? A cluster resource property called*FailureConditionLevel* value determines whether the information returned by

sp_server_diagnostics warrants a failover. The default value for this property is 3, but we can change it from 1 to 5, ranging from least to most restrictive.

So what happens if *sp_server_diagnostics* cannot run at all? That is where another cluster resource property called *HealthCheck Timeout* comes into play. The default value for this is 30 seconds, meaning if *sp_server_diagnostics*returns nothing to the resource DLL within this period, the server is considered unresponsive, and failover may be triggered. Therefore, SQL FCI fails over if either *sp_server_diagnostics* returns health information that *FailureConditionLevel*determines as severe or if the *HealthCheck Timeout* threshold is reached. Note that even when the proper failover conditions are met, WSFC triggers a failover only when the quorum is maintained. If not, it can bring the entire FCI offline.

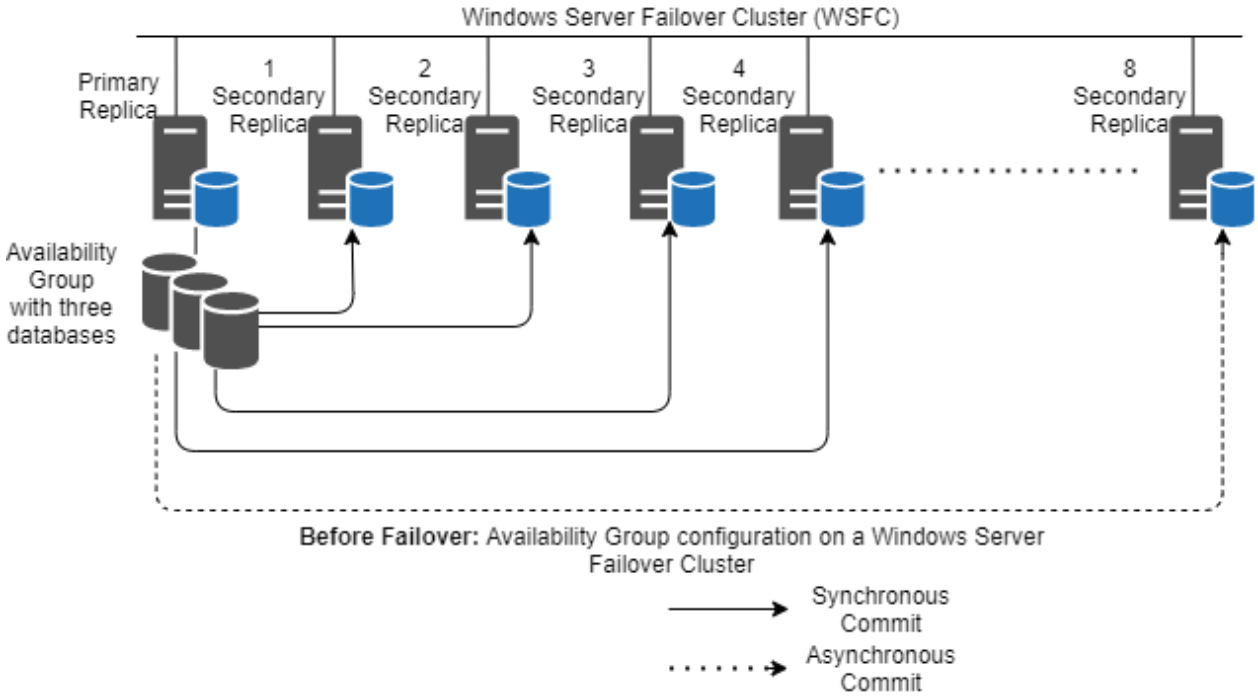
We must note that *sp_server_diagnostics* only captures SQL instance's health and not database's, meaning a failover is not triggered if a database crashes. Also, while the stored procedure returns diagnostics related to the system, resource, query process, io_subsystem, and events, only system, resource, and query process information is used for failure detection.

10. Always On Availability Groups (AOAG)

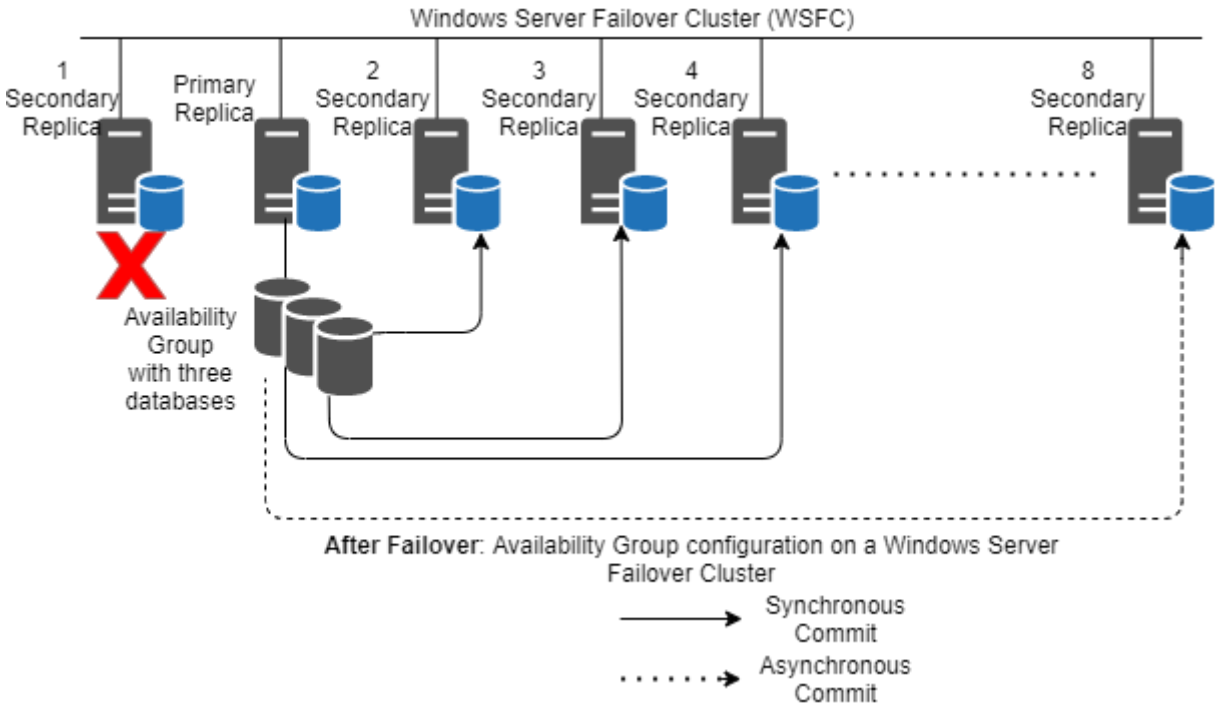
Always On Availability Groups or AGs for short, provide both high availability and disaster recovery solution for SQL Server databases. It is a database-level feature that overcomes the drawback of a single point of failure at the storage in SQL FCIs by providing storage level redundancy. The solution includes a primary replica at a high level – a standalone SQL instance or a SQL FCI, and up to eight secondary replicas, which may be standalone or SQL FCIs. An AG protects a set of databases on the primary replica by continuously transferring transaction log blocks from all AG databases on the primary replica to their corresponding databases across all secondary replicas. This continuous data replication from the primary to secondaries keeps the primary and secondary copies of the databases in sync via either synchronous or asynchronous data transfer, which are called availability modes.

In synchronous-commit mode, when a client commits a transaction on the primary, the commit confirmation is not sent until all synchronous secondary replicas write the transaction log records to disk, thus allowing zero data loss at the cost of potential transaction latency. In SQL Server 2019, there can be up to four synchronous secondary replicas. In asynchronous-commit mode, the commit confirmation is sent to the client immediately as soon as the log records are hardened on the primary replica regardless of whether they made the secondary replicas, thus offering fast performance at the cost of potential data loss in case of a disaster. A later section discusses how async-commit mode can incur data loss. All replicas can use the async-commit mode.

In availability groups, a failover results in failing all the AG databases as a unit over to a secondary replica, making the otherwise secondary the new primary replica and bringing all AG databases online in read-write mode. The former primary replica assumes the secondary role when it comes back online and joins the new primary replica. The two below figures illustrate a before and after failover scenario for a nine-replica AG setup on a nine-node WSFC where each node acts as a standalone replica.



Notice the AG with three databases fails over to secondary replica 1 and switches roles.

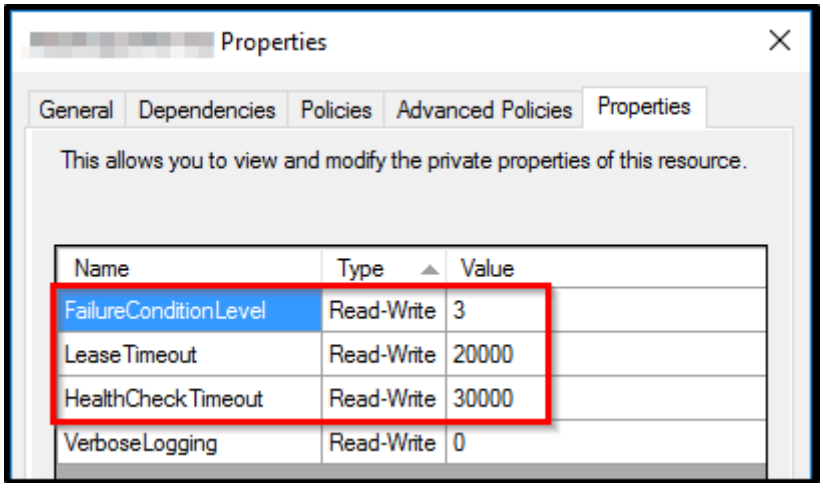


Multiple components help detect failures and trigger failovers. The failover conditions discussed in the FCI section above equally apply to availability groups since AGs run as cluster resources just like SQL FCIs. Health-check timeout, sp_server_diagnostics, FailureConditionLevelall apply to AGs as well. The difference, however, is failover is at the availability group level instead of the SQL instance level, meaning all databases that are part of the same AG failover together as a unit which is especially good for applications that access multiple databases. We can combine them all in a separate AG, and when it fails over, the application will have a brief outage and then connect to the new primary replica with all the databases it needs.

11. Lease timeout

A cluster resource setting that controls the AG failover mechanism is lease timeout, whose default value is 20,000 ms or 20 seconds. It helps prevent the split-brain scenario in AGs. The lease mechanism is a communication between the cluster resource host and the SQL Server process, ensuring that both cluster and SQL Server services on the primary replica remain in contact. If not, it can cause the primary replica to give itself up and trigger a failover to one of the other secondary replicas. The two players in the lease mechanism are the resource host and the SQL Server lease worker thread. Both work circularly, triggering their respective lease renewal, sleeping, waiting for the other party to signal its lease renewal, and repeat the entire process. Both have a lease threshold called time-to-live, which gets updated upon each lease renewal. If the time-to-live value is reached with no lease renewal, the lease is considered expired, and AG failover is initiated.

Lease expirations or timeouts can become frequent when there is high CPU utilization, unresponsive SQL Server, out-of-memory conditions because the lease mechanism uses a small portion of resource host memory. These situations can cause the lease not to renew within the threshold and triggers a failover which may be unnecessary. If this becomes recurrent, a temporary workaround may be to bump up the lease timeout value, but that masks the underlying root cause, which may be the workloads, sessions, or even bugs. A screenshot of AG properties in the failover cluster manager is shown below.



Starting SQL Server 2016, not only is failover controlled at the instance level as with FCIs discussed above, but any database-level failure can trigger an AG failover as long as we enable the "Database Level Health Detection"option in the AG properties. The database level health detection monitors the health of an AG database, and if its state changes to anything other than online, it triggers the AG failover. The database health check mechanism was later enhanced to include several other checks, including page corruption, checkpoint failures, disk corruption, memory corruption, no disk space in a filegroup, and some more. All of which affect database availability and thus trigger the AG failover.

The three types of failover in availability groups are:

1. Automatic failover without data loss
2. Planned manual failover without data loss
3. Forced failover with 'possible' data loss.

12. What happens in an automatic AG failover?

Note that there is no data loss in auto-failover since auto-failover is only possible when the AG is configured in synchronous-commit mode. SQL Server hardens the log records for transactions on the secondary before it sends a commit acknowledgment to the client, guaranteeing no data loss in case of a failover.

Once it is determined, based on the failures discussed above, the AG must fail to a secondary replica, SQL Server performs the following actions in sequence.

1. All existing client sessions are terminated, and the primary databases' state changes to "Disconnected".
2. The secondary replica takes over and rolls forward any unprocessed log records in the recovery queue, followed by performing recovery against the AG databases. Recovery includes the analysis, redo, and undo phases. The new primary replica rolls back any uncommitted transactions in the background while the database serves the clients.

Thanks to the new parallel redo process introduced in SQL Server 2016, multiple threads on the secondary replica work to redo the transaction log records received from the primary replica, thus speeding up the redo.

3. If the former primary replica comes back up, the AG databaseson it show as "Not Synchronized",meaning the replica is not in sync with the new primary replica. It thenre-synchronizes with the new primary replica and switches the state of the databases to "Synchronized" after it has caught the transaction log up to date.

13. What happens in a planned manual AG failover?

Since it is planned, there is no data loss as long as both replicas are in sync. If the configuration is async-commit, there is potentially a chance for data loss. However, if it is a planned failover, a better idea is to temporarily switch to sync-commit and revert after the maintenance is over to ensure no accidental data loss.

We should start a planned manual failovereither via SSMS, T-SQL, or PowerShell on the target secondary replica. It goes through the following sequence.

1. WSFC sends a request to the primary replica to go offline,terminating all existing client connections.
2. The secondary replica takes over as primary replica and rolls forward any pending log records in the recovery queue, followed by performing recovery against the AG databases. The former primary becomes the secondary replica.
3. The databases on the new secondary show as "Not Synchronizing" (since async-commit), indicating the replica is not in sync with the new primary replica. As soon as the log catches up, the new secondary replica resynchronizes with the new primary replica and switches the state of the databases to "Synchronizing".

14. What happens in a forced failover?

Forcing a failover involves the same steps as in a planned manual failover discussed above, except with a change when the original primary becomes the new secondary replica when it comes back online. All AG databases on the new secondary will be in a suspended state. By not automatically synchronizing the new secondary and new primary replicas and resuming replication, SQL Server allows us to salvage any committed data from the new secondary replica if we configure it as readable.

Issuing a forced failover on a synchronized secondary replica has the same effect as a manual failover.

15. Can a forced failover cause data loss?

A forced failover results in potential data loss on the primary replica. We force a failover when things go out of whack. In the async-commit configuration, the commit acknowledgment is sent to the client before the corresponding log record is hardened on the disk onthe secondary replica, meaning if a disaster strikes on the primary replica, the committed transaction that did not make the secondary replica is lost when a failover occurs because it only lived on the primary replica.

Availability Groups use the log sequence numbers (LSN) to coordinate the log transfer. For example, say the last hardened LSN on the primary replica before a disaster struck was 200,and on the secondary replica,it is 150. The lag is because of the async-commit configuration. When a forced failover is initiated, itbrings the AG databases online on the former secondary replica, making it the new primary replica and marks the last hardened LSN as 150. At this point, if the former primary replica comes back online, it shows as suspended.

If it is readable, there may be an opportunity to salvage data from it since it was farther in LSN (200) and may have more data than the current primary replica.

Suppose we resynchronize the former primary replica to make it the new secondary. In that case, it sends its last hardened LSN as 200, but since the last hardened LSN on the new primary is 150, the new secondary rolls back the transaction log to LSN 150 to match with the new primary and continues applying the log from thereon. So if there were committed transactions between LSN 150 to 200, we would lose them.

16. Conclusion

If a company does not have a disaster recovery (DR) plan, at least for their revenue-generating mission-critical applications to support business continuity (BC) in a time of disaster. In that case, it is potentially setting itself up for a tremendous amount of losses. SQL Server provides two solutions: SQL Server FCI and Availability Groups under the term Always On, with the latter providing data redundancy. Failover in both solutions is triggered when certain conditions, mostly configurable,are met. However,it is advised not to change them unless in exceptional circumstances. Last, a DR solution must be thoroughly tested in a lower environment such as development or test to ensure the failovers and failbacks work as expected and that the databases are operational.

17. References

[1] Bertucci, P., & Shreewastava, R. (2018). *SQL Server 2016 High Availability Unleashed.* Pearson Education.

[2] *Mechanics and guidelines of lease, cluster, and health check timeouts for Always On availability groups.* (2018, May 02). Retrieved from docs.microsoft.com: https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/availability-group-lease-healthcheck-timeout?view=sql-server-ver15

[3] Parui, U., & Sanil, V. (2016). *Pro SQL Server Always On Availability Groups.* Apress.