SQL Server High Availability and Disaster Recovery Plan

Microsoft provides quite a few features for high availability and disaster recovery. **Author: Priyanka Chouhan**

http://community.spiceworks.com/scripts/show/1512-restore-all-sql-databases-in-a-directory-to-instance-using-a-script

With the emerging need to have critical applications available almost 24×7 to the business or end-users, all companies need to carefully plan how to provide this service availability and shorten scheduled downtime in case of any disasters. SQL Server supports several High Availability and Disaster Recovery techniques. Even though both High Availability (HA) and Disaster Recovery (DR) are different terminologies, they are both aimed at providing continuous access to services or data with the least possible downtime possible. In this article, I'll explain in more detail the difference between HA and DR, the techniques available, and how to create a Disaster Recovery Plan specifically for a server, instance or database.

HA/DR Terminologies

Before diving into the discussion about High Availability or Disaster Recovery solutions, everyone should be familiar with the terminology.

Planned Maintenance

Planned maintenance is a coordinated/scheduled downtime after informing all required personnel to perform Maintenance activities like

1. Application patching or upgrading
2. Applying Windows OS patches or security fixes
3. Performing hardware upgrades or firmware patches
4. Test DR solutions implemented in the organization

Unplanned Outages

An unplanned outage is a non-coordinated downtime and can happen out of any scenarios below

1. Application-level issues
2. Infrastructure level issues (VM, OS, server or storage issues)
3. Catastrophically natural disasters
4. Storage or database level corruptions

Recovery Time Objective (RTO)

RTO is the acceptable downtime for the application, whether from planned maintenance or unplanned outages. If the RTO for a particular application is 12 Hours, then the maximum downtime for that application can be 12 Hours, and the application should be back working in 12 Hours.

Recovery Point Objective (RPO)

RPO is the acceptable limit of data loss that the organization/system can afford to lose, often measured by time. RPO varies from database to database or application to application.

For example, a production database may have a recovery point of minutes, while a test or development database can afford to lose days to weeks of data.

Recovery Level Objective (RLO)

RLO defines the level of granularity required to recover data which might be at an instance, database, or table level. Recovering from database level corruption is an excellent example of this.

SQL Server High Availability

The primary goal of high availability is to ensure service availability by minimizing the impact of downtime from either planned or unplanned activities. Almost all high availability techniques will have their infrastructure placed in a single geographic location and have the option to switch over automatically.

SQL Server supports these high availability techniques

1. Windows Failover Cluster Instance (within single datacenter) – Till SQL Server 2008 R2
2. Transactional Replication
3. Log Shipping (within or nearby datacenter with fast data transfer options)
4. Database Mirroring (Synchronous)
5. Always On Failover Cluster Instance (within single datacenter) – Available from SQL Server 2012 onwards
6. Always On Availability Groups (within or nearby datacenter with fast data transfer options)

SQL Server Disaster Recovery

The primary goal of Disaster Recovery is to provide service continuity in case of any geographical disasters by recovering or resuming the services from a geographically different location. DR can be either located in one or more locations in addition to the primary location.

SQL Server supports the below Disaster Recovery Techniques

1. Backup and Restore
2. Peer to Peer Replication
3. Log Shipping (across geographically separated Datacenters)
4. Database Mirroring (Asynchronous)
5. Storage or VM Replication
6. Always on Failover Cluster with Storage Replication
7. Always on Failover Cluster Instance with Storage Replication.

Backup and Restore Databases

Taking a SQL Server backup, copying it to an HA site/server or DR site/server is one of the primitive approaches for achieving any HA/DR solution. Of course, it won’t be real-time, and there would be a lot of time and effort involved, and, hence RTO/RPO values must be defined after careful testing of this approach and defining tht RTO/RPO values. In order to minimize the potential risk of losing data to meet RPO requirements, a Full Backup job should be scheduled for all production databases and monitored for successful execution of the job. A Differential Backup can be implemented along with Full Backup if the databases are huge in size to meet the RTO expectations.

Here are a few pointers to consider while creating a backup schedule.

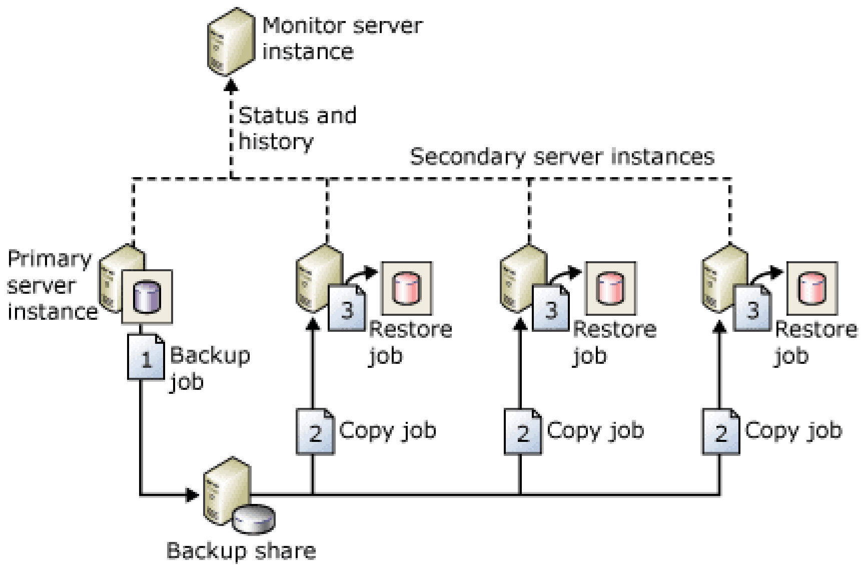
1. For all user databases, full with transactional Log Backup should be in place. Differential Backup can be included if the database you are dealing with is quite large and the size of Full Backup is huge.
2. For all system databases, at least a full backup once every day or week should be in place.
3. Restore verification process/plan should be in place for all database backups and should be done frequently.
4. Database backups should be moved to network shares or shared folders and not placed in the same server to avoid potential risks of data loss due to server hardware failures or storage failures.

Log Shipping

SQL Server Log shipping is a feature that allows you to send transaction log backups from the Source Server to Destination Servers. The backup job executes on the database on a primary server instance and copies the Log Backup to one or more secondary databases on separate secondary server instances. Transaction log backups received are restored to each of the secondary databases individually via a Restore Job.

To monitor and automate the entire process, an optional third server instance can be part of this architecture to record the history and status of backup and restore operations. In addition to that, a Witness Server optionally will raise alerts if any of the Backup or Restore operations fail. Log Shipping within the same datacenter can be considered as a HA solution whereas across geographically separated datacenter will be a DR solution.

Below is the architecture of SQL Server Log Shipping



Source: About Log Shipping (SQL Server) – https://docs.microsoft.com/en-us/sql/database-engine/log-shipping/about-log-shipping-sql-server?view=sql-server-ver15

If this solution meets the RPO and RTO, here are some of the merits of implementing this solution. First and foremost, it's an inexpensive approach as SQL Server controls the log files without any additional dependencies. It works best in environments where databases are under a bandwidth constraint, and there is no requirement to keep synchronized copies of your databases in multiple locations in real-time.

As always, no solution is perfect; hence note some of the trade-offs that come with log shipping. It's highly dependent on the volume of transactions that occurred to the primary SQL Server databases since the last log was shipped. This can take a substantial amount of time to roll one or more databases forward based on the log files shipped.

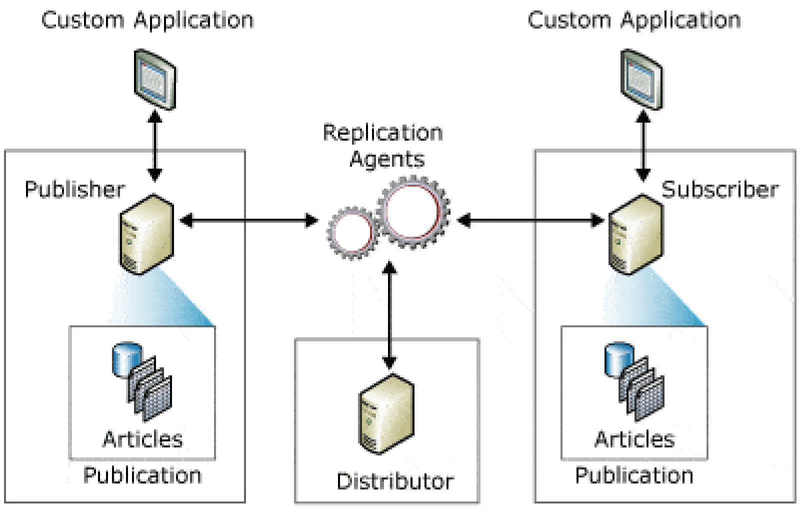
Therefore, you need to test your RTO objective with this solution to ensure that you are in compliance. Since there is no automatic failover for your databases, the process becomes complicated and manual to bring up a backup server during a disaster. It's also required to manually reconfigure your system/application to connect to the backup server running the SQL Server. It's even more daunting when failing back to your primary server after service is restored. It gets complex as you need to reconfigure log shipping in reverse and reseed your data back to your primary database configuration.

All these issues are why I cannot stress enough why you need to test your RTO against this solution. If your RTO is to have your SQL Server restored in less than an hour, you may want to move to the next solution described below.

SQL Server Replication

SQL Server Replication terminology consists of

1. Publisher – Source Database instance which publishes data changes to another database.
2. Distributor – Database to log the changes that happened from Publisher database transaction logs
3. Subscriber – Destination database instance where the data changes captured in Distributor database will be distributed.
4. Publications – Collection of one or more Articles from a database in Publisher.
5. Articles – Database objects involved in Replication which can be tables, views, stored procedures or functions.
6. Subscription – Subscription defines what publication will be received, from which publication and what schedule data would be replicated. The subscription can be either push or pull, indicating whether the data is pushed or pulled from the publication to subscription.



Source: https://docs.microsoft.com/en-us/sql/relational-databases/replication/publish/replication-publishing-model-overview?view=sql-server-ver15

Among multiple types of SQL Server Replication, Peer-to-Peer Replication can serve well for DR purposes since Peer to Peer Replication can act as a Publisher at any point in time required. Transactional Replication is not exactly a HA solution but can help off-loading Reporting queries load to another server.

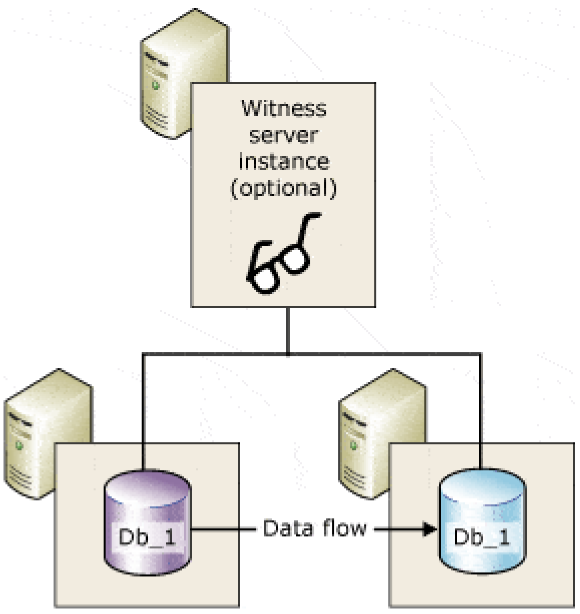
Database Mirroring

Database Mirroring is primarily an HA solution and included in this article for references but will be removed in later versions of SQL Server. Database Mirroring involves two servers. A Source database resides on a Source Server called the Principal Server, and the Destinatinon database, also called the Mirrored database, resides on a Destination Server called the Mirrored Server.

The Witness Server is an optional third server that can help monitor the heartbeat of Mirroring between Principal and Mirror Server and can initiate automatic failover to the Mirrored Server when the mirroring is set to High Safety mode.

Types of Database Mirroring

1. High Safety Mode (Synchronous) – Any changes happening in the Principal database will be sent across to the mirrored database. Once changes are committed in the Mirror Database, they will be committed in the Principal database. Suitable for HA solution.
2. High Performance Mode (Asynchronous) – Any changes in the principal database will be committed in the principal database, and committed changes will be sent asynchronously to the mirrored database. Suitable for DR solution across multiple data centers.



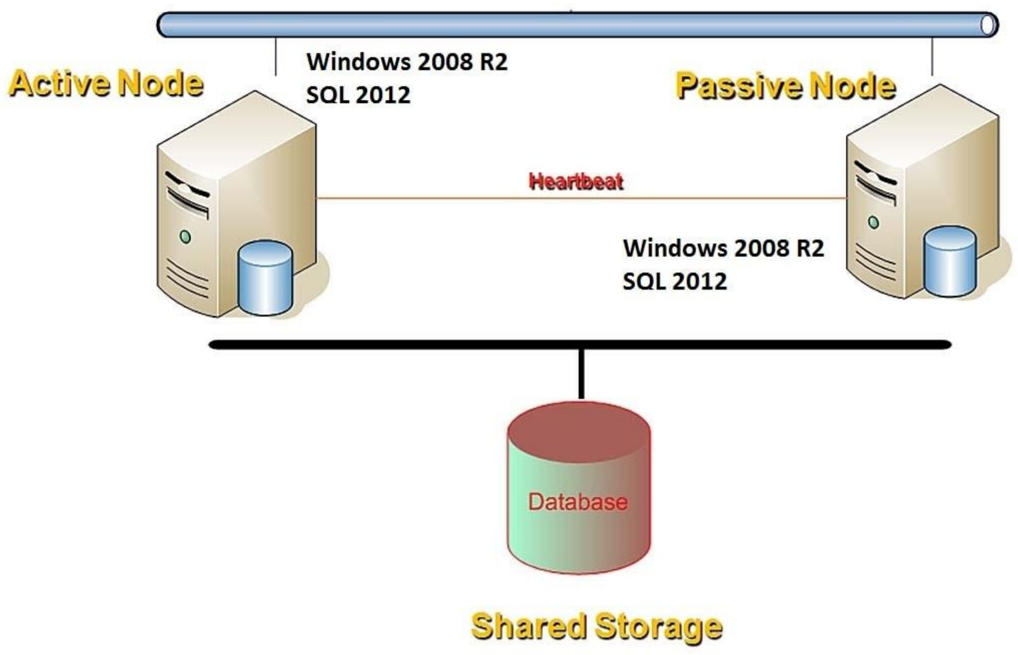
Source: https://docs.microsoft.com/en-us/sql/database-engine/database-mirroring/database-mirroring-sql-server?view=sql-server-ver15

Failover Clusters

SQL Server Failover Clusters also called as Failover Cluster Instance (FCI) uses Windows Server Failover Cluster(WSFC) nodes framework as the backbone with SQL Server installed across both nodes of WSFC. FCI will have a shared Storage where all the Data and Log files of all databases will be located along with these FCI configurations stored in a Quorom storage.

At any point in time, FCI will have one node of WSFC in active status attaching the Shared storage and Quorom storage keeping SQL Server instance available to end users. Any issues on the current active instance will failover all the Shared storage drives and Quorom storage to the current Passive node automatically with very minimal switching time leaving SQL Server services available to users.

This HA plan will require configuring multiple servers in the same Windows Server Failover Cluster (WSFC) having a shared storage area network (SAN) on which one or more SQL Server user databases reside. Below is a diagrammatic representation of the Failover Cluster Instance HA solution.



Source: SQL Server DBA Interview Questions and Answers – SQL Server Cluster 2 – https://www.dbamantra.com/sql-server-dba-interview-questions-answers-sql-server-cluster-2/

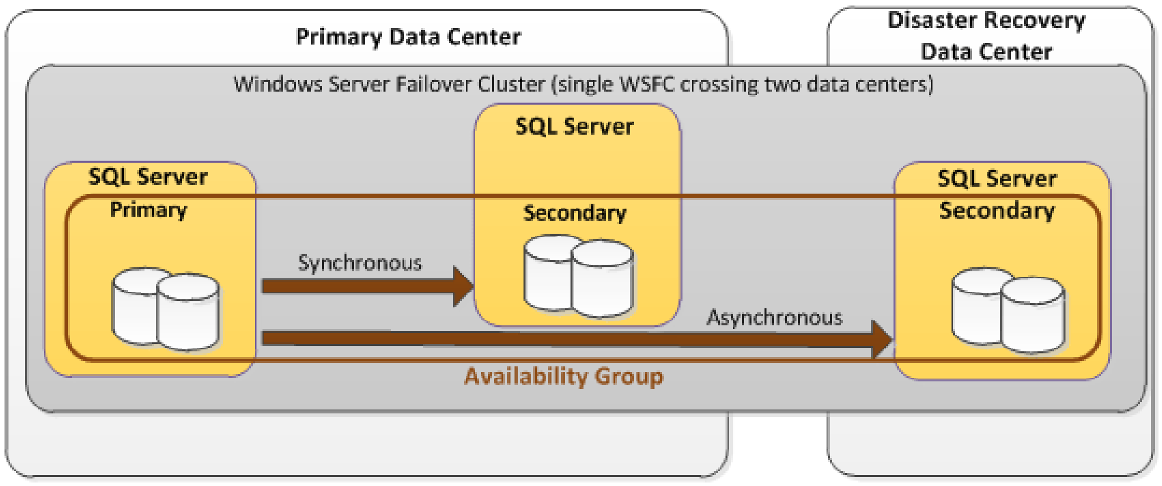
Failover Cluster instances are the basic HA solution implemented across most Production environments. Since the SAN storage is shared across both nodes in the cluster, it becomes the primary point of failure causing all databases and jobs to fail in case of any catastrophic disasters.

Hence combining with SAN-based Replication from one SAN Storage to another one within same datacenter or different datacenter can help enhance this Failover instance to a DR solution which will have all databases, jobs, logins, etc., replicated from Primary Storage to DR Storage.

Always-On Availability Group

Microsoft announced that Database Mirroring feature will be deprecated starting SQL Server 2012 (even though it is supported till the latest version) and introduced an advanced solution which can cater both High Availability and Disaster Recovery Solution solutions named as Always-On Availability groups.

Similar to Database Mirroring, Always on Availability groups enables high availability to a set of user databases which can fail over together. While Database Mirroring doesn’t support the readability of Mirror database, Availability group have additional configurations to make the Primary database with Read-write functions and secondary databases for read-only access and/or some backup operations to reduce the load on Primary databases.



Source: MS SQL AlwaysOn Availability Groups, a better way to cluster
– https://www.jpaul.me/2013/09/ms-sql-alwayson-availability-groups-a-better-way-to-cluster/

In this solution, the biggest concern will be cost. You will notice that additional hardware, licensing, and more is required. The more replicas that participate in the AG, the more costly it becomes. Nowadays, you have the option of leveraging a hybrid environment where some of the infrastructure could be cloud-based. You can have some replicas in either Azure or AWS, which helps with scaling and cost.

There is a limitation with AGs as it only replicates the user SQL Server databases. The system databases like master, msdb, and model are not part of the AG. All these databases are managed separately by each replica's SQL Server. Additional processes are required to periodically sync the changes of the system databases like login changes and agent jobs so they can stay in sync as close as possible.

Prepare a Disaster Recovery Plan Document

A documented plan is your first point of resolution during a disaster recovery of your SQL Server. This will prevent a lot of guesswork as well making sure to reach RPO and RTO. Below is a list of things that need to be documented. This list is not the holy grail, and more relevant information can be added to it as needed.

- Full System architecture – integration overview of the database and application
- Identify system's SLAs and select appropriate technology solution.
- Identify and document all systems involved.
- Document all assets for the system like server drives, operating systems, IP Addresses, file locations, and more.
- Document security information like logins, certificates, jobs/schedule information, and SQL Server configurations.
- Document contact information of all stakeholders of the system like DBAs, developers, network administrators, and etc.
- Document step by step instructions with estimated timelines on recovery of the SQL Server based on identified disaster scenarios.
- Involve all parties and review the document for complete consensus before finalizing it.
- Set up a change management process to allow any updates to be reflected in this DRP document.
- Finally, test the document in a dry run to ensure everything is covered as much as possible.

Conclusion

To conclude, Business Continuity can be achieved via both HA and DR solutions, and in several scenarios, both complement each other. You will need to determine the better HA and DR solutions to meet business requirements, and here are the key items to take home with you.

- Remember to have a Scheduled backup solution in place for both User and System databases.
- Ensure the backups are in a restorable state by verifying them regularly.
- Identify a better High Availability solution for your critical applications to reduce the downtime for Planned Maintenances.
- Whether you have a High Availability solution implemented or not, identify, test and implement a DR plan for Unplanned Outages.

Having a complete HA and DR plan in place will help you recover from any scenarios. In case of any database corruption, to meet RLO, try recovering the corrupted database or database backup using the DBCC CHECKDB command. Another option is to use a third-party product like Stellar SQL Recovery Software which is a great tool that can quickly fix database corruption and maintain consistency of the data. Find out more in the review done by **Grant Fritchey** (Data Platform, MVP) here.