

Describe high availability and disaster recovery strategies

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/describe-high-availability-disaster-recovery-strategies/1-introduction>

Introduction

Completed

Implementing High Availability and Disaster Recovery (HADR) solutions for a database platform involves more than just deploying a feature. It's crucial to understand the reasons behind your choices and the strategies you're implementing.

For example, if you need to set up HADR for a virtual machine, do you know how much time you have to bring everything back online if there's a problem? Are you aware of the available options in SQL Server and the reasons for choosing one over another?

By the end of this module, you'll have a solid foundation to implement HADR solutions in Azure.

2. Describe recovery time objective and recovery point objective

<https://learn.microsoft.com/en-us/training/modules/describe-high-availability-disaster-recovery-strategies/2-describe-recovery-time-objective-recovery-point-objective>

Describe recovery time objective and recovery point objective

Completed

Understanding recovery time and recovery point objectives are crucial to your high availability and disaster recovery (HADR) plan as they're the foundation for any availability solution.

Recovery Time Objective

Recovery Time Objective (RTO) is the maximum amount of time available to bring resources online after an outage or problem. If that process takes longer than the RTO, there could be consequences such as financial penalties, work not able to be done, and so on. RTO can be specified for the whole solution, which includes all resources, and for individual components such as SQL Server instances and databases.

Recovery Point Objective

Recovery Point Objective (RPO) is the point in time to which a database should be recovered and equates to the maximum amount of data loss that the business is willing to accept. For example, suppose an IaaS VM containing SQL Server experiences an outage at 10:00 AM and the databases within the SQL Server instance have an RPO of 15 minutes. No matter what feature or technology is used to bring back that instance and its databases, the expectation is that there will be at most 15 minutes worth of data lost. That means the database can be restored to 9:45 AM or later to ensure minimal to no data loss meeting that stated RPO. There may be factors that determine if that RPO is achievable.

Defining Recovery Time and Recovery Point Objectives

RTOs (Recovery Time Objectives) and RPOs (Recovery Point Objectives) are driven by business requirements and many technological factors, including the skills of administrators. While businesses may desire no downtime or zero data loss, this is often unrealistic. Determining RTO and RPO should involve open discussions among all parties.

Understanding the cost of downtime is crucial for defining RTO and RPO. For example, if downtime costs the business \$10,000 per hour or results in fines, this helps in setting realistic objectives. The investment in a High Availability and Disaster Recovery (HADR) solution should be proportional to the cost of downtime. If a solution prevents hours or days of downtime, it justifies its cost.

RTO should be defined at both the component level (for example, SQL Server) and the entire application architecture. The overall RTO is determined by the slowest component to recover. For instance, if SQL Server recovers in five minutes but application servers take 20 minutes, the overall RTO is 20 minutes.

RPO focuses on data and influences the design of HADR solutions and administrative policies. Features used must support the defined RTO and RPO. For example, if transaction log backups are scheduled every 30 minutes but the RPO is 15 minutes, the database can only be recovered to the last backup, which could be 30 minutes old. Regularly testing backups ensures their reliability.

The choice of solutions, such as Always On Availability Groups (AG) or Failover Cluster Instances (FCI), affects RTO and RPO. Depending on configuration, IaaS or PaaS solutions may not automatically fail over, leading to longer downtime. If RTO and RPO are unrealistic, they must be adjusted. For example, if a backup takes three hours to copy but the RTO is two hours, the RTO is missed.

RTOs and RPOs should be defined for both High Availability (HA) and Disaster Recovery (DR). HA events are localized and easier to recover from, such as an AG failing over within an Azure region, potentially taking seconds. DR involves bringing up a new data center, which can take hours or longer, hence separate RTOs and RPOs.

All RTOs and RPOs should be documented and periodically revised. Once documented, appropriate technologies and features can be considered for the architecture.

3. Explore high availability and disaster recovery options

<https://learn.microsoft.com/en-us/training/modules/describe-high-availability-disaster-recovery-strategies/3-explore-high-availability-disaster-recovery-options>

Explore high availability and disaster recovery options

Completed

To envision a solution for virtual machines (VMs), you must first understand the availability options for IaaS-based deployments.

Infrastructure-as-a-Service versus Platform-as-a-Service

When it comes to availability, the choice of IaaS or PaaS makes a difference. With IaaS, you have a virtual machine, which means there's an operating system with an installation of SQL Server. The administrator or group responsible for SQL Server would have a choice of high availability and disaster recovery (HADR) solutions and a great deal of control over what how that solution was configured.

With PaaS-based deployments such as Azure SQL Database, the HADR solutions are built into the feature and often just need to be enabled. There are minimal options that can be configured.

Because of these differences, the choice of IaaS or PaaS may influence the final design of your HADR solution.

SQL Server HADR Features for Azure Virtual Machine

When using IaaS, you can use the features provided by SQL Server to increase availability. In some cases, they can be combined with Azure-level features to increase availability even further.

The following table shows the solutions available in SQL Server:

Feature Name	Protects
Always On Failover Cluster Instance (FCI)	Instance
Always On Availability Group (AG)	Database
Log Shipping	Database

An instance of SQL Server is the entire installation of SQL Server (binaries, all the objects inside the instance including things like logins, SQL Server Agent jobs, and databases). Instance-level protection means that the entire instance is accounted for in the availability feature.

A database in SQL Server contains the data that end users and applications use. There are system databases that SQL Server relies on, and databases created for use by end users and applications. An instance of SQL Server always has its own system databases. Database-level protection means that anything that is in the database, or is captured in the transaction log for a user or application database, is accounted for as part of the availability feature. Anything that exists outside of the database or that isn't captured as part of the transaction log such as SQL Server Agent jobs and linked servers must be manually dealt with to ensure the destination server can function like the primary if there's a planned or unplanned failover event.

Both FCIs and AGs require an underlying cluster mechanism. For SQL Server deployments running on Windows Server, it's a Windows Server Failover Cluster (WSFC) and for Linux it's Pacemaker.

Always On Failover Cluster Instances

An FCI is configured when SQL Server is installed. A standalone instance of SQL Server can't be converted to an FCI. The FCI is assigned a unique name and an IP address that is different from the underlying servers, or nodes, participating in the cluster. The name and IP address must also be different from the underlying cluster mechanism. Applications and end users would use the unique name of the FCI for access. This abstraction enables applications to not have to know where the instance is running. One major difference between Azure-based FCIs versus on-premises FCIs, is that for Azure, an internal load balancer (ILB) is required. The ILB is used to help ensure applications and end users can connect to the FCI's unique name.

When an FCI fails over to another node of a cluster, whether it's initiated manually or happens due to a problem, the entire instance restarts on another node. That means the failover process is a full stop and start of SQL Server. Any applications or end users connected to the FCI is disconnected during failover and only applications that can handle and recover from this interruption can reconnect automatically.

When the instance starts up on the other node, it undergoes the recovery process. The Failover Cluster Instance (FCI) is consistent up to the point of failure, ensuring no data loss. Any transactions that need to be rolled back is handled during recovery. Since this is instance-level protection, all necessary components (logins, SQL Server Agent jobs, etc.) are already in place, allowing business operations to resume as usual once the databases are ready.

FCIs require one copy of a database, but that is also its single point of failure. To ensure another node can access the database, FCIs require some form of shared storage. For Windows Server-based architectures, this can be achieved via an Azure Premium File Share, iSCSI, Azure Shared Disk, Storage Spaces Direct (S2D), or a supported third-party solution like SIOS DataKeeper. FCIs using Standard Edition of SQL Server can have up to two nodes. FCIs also require the use of Active Directory Domain Services (AD DS) and Domain Name Services (DNS), so that means AD DS and DNS must be implemented somewhere in Azure for an FCI to work.

In Windows Server, FCIs can use storage replica to create a native disaster recovery solution for FCIs without having to use another feature such as log shipping or AGs.

Always On availability groups

Availability Groups (AGs) were introduced in SQL Server 2012 Enterprise Edition and are available in Standard Edition starting with SQL Server 2016. In Standard Edition, an AG can contain one database, while in Enterprise Edition, it can have multiple databases. Although AGs share some similarities with Failover Cluster Instances (FCIs), they're different in many ways.

The main difference is that AGs provide database-level protection. The primary replica in an AG contains the read/write databases, while the secondary replica receives transactions from the primary to stay synchronized. Data movement can be synchronous or asynchronous. In Standard Edition, an AG can have up to two replicas (one primary, one secondary), whereas Enterprise Edition supports up to nine replicas (one primary, eight secondary). Secondary replicas are initialized from a database backup or through automatic seeding, which streams the backup to the secondary replica.

AGs use a listener for abstraction, which functions like the unique name assigned to an FCI and has its own name and IP address. Applications and end users can connect using the listener, but direct connections to the instance are also possible. In Enterprise Edition, secondary replicas can be configured for read-only access and used for tasks like database consistency checks (DBCCs) and backups.

AGs offer quicker failover times compared to FCIs and don't require shared storage. Each replica has its own copy of the data, increasing the total number of database copies and overall storage costs. For example, if the primary replica's data footprint is 1 TB, each replica will also have 1 TB of data, requiring 5 TB of space for five replicas.

Objects outside the database or not captured in the transaction log must be manually created on any new primary replica. Examples include SQL Server Agent jobs, instance-level logins, and linked servers. Using Windows authentication or contained databases with AGs can simplify access.

Organizations may face challenges implementing highly available architectures and might only need the high availability provided by the Azure platform or a PaaS solution like Azure SQL Managed Instance. Before exploring Azure platform solutions, it's essential to know about another SQL Server feature: log shipping.

Log shipping

Log shipping has been a fundamental feature of SQL Server since its early days. It's based on backup, copy, and restore, making it one of the simplest methods for achieving High Availability and Disaster Recovery (HADR). While primarily used for disaster recovery, log shipping can also enhance local availability.

Like Availability Groups (AGs), log shipping provides database-level protection, meaning you must account for SQL Server Agent jobs, linked servers, instance-level logins, and other objects. Unlike AGs, log shipping doesn't offer native abstraction, so switching to another server requires tolerating a name change. This can be mitigated using methods like DNS aliases configured at the network layer.

The log shipping process is straightforward: take a full backup of the source database on the primary server, restore it in a loading state (STANDBY or NORECOVERY) on a secondary server, known as a warm standby. This new copy is called the secondary database. SQL Server then automatically backs up the primary database's transaction log, copies the backup to the standby server, and restores it onto the standby.

In addition to SQL Server HADR features, there are Azure features that can enhance IaaS availability.

4. Describe Azure high availability and disaster recovery features for Azure Virtual Machines

<https://learn.microsoft.com/en-us/training/modules/describe-high-availability-disaster-recovery-strategies/4-describe-azure-high-availability-disaster-recovery-features>

Describe Azure high availability and disaster recovery features for Azure Virtual Machines

Completed

Azure provides three main options to enhance availability for IaaS deployments:

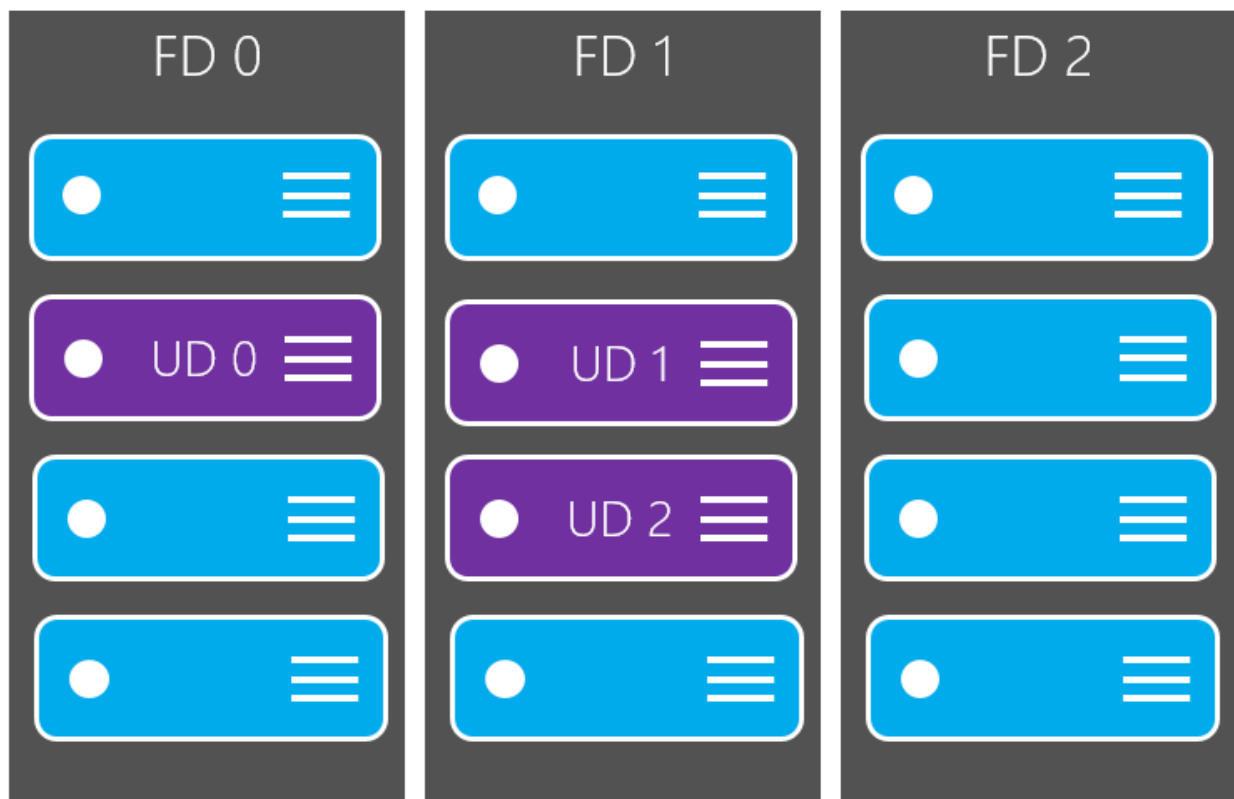
- Availability Sets

- Availability Zones
- Azure Site Recovery

All three of these options are external to the virtual machine (VM) and don't know what kind of workload is running inside of it.

Availability sets

Availability sets provide uptime against Azure-related maintenance and single points of failure in a single data center. This was one of the first availability features introduced into the Azure platform, and effectively it can be thought of as anti-affinity rules for your VMs. This means if you had two SQL Server VMs in an availability set or log shipping pair, they would be guaranteed to never run on the same physical server.



Availability sets in Azure are designed to enhance the reliability and availability of virtual machines by separating them into fault domains and update domains. Fault domains are groups of servers within a data center that share the same power source and network. Each data center can have up to three fault domains, labeled as *FD 0*, *1*, and *2* in the image. Update domains, indicated by UD in the image, represent groups of virtual machines and their underlying physical hardware that can be rebooted simultaneously. By distributing virtual machines across different update domains, Azure ensures that updates and maintenance activities don't affect all instances at once, thereby maintaining service continuity.

Availability sets and zones don't protect against in-guest failures, such as an OS or RDBMS crash; which is why you need to implement other solutions such as AGs or FCIs to ensure you meet RTOs and RPOs. Both availability sets and zones are designed to limit the impact of environmental problems at the Azure level such as datacenter failure, physical hardware failure, network outages, and power interruptions.

For a multi-tier application, you should put each tier of the application into its own availability set. For example, if you were building a web application that has a SQL Server backend along with Active Directory Domain Services (AD DS), you would create an availability set for each tier (web, database, and AD DS).

Availability sets aren't the only way to separate IaaS VMs. Azure also provides Availability Zones, but the two can't be combined. You can pick one or the other.

Availability zones

Availability zones account for data center-level failure in Azure. Each Azure region consists of many data centers with low latency network connections between them. When you deploy VM resources in a region that supports Availability Zones, you have the option to deploy those resources into Zone 1, 2, or 3. A zone is a unique physical location, that is, a data center, within an Azure region.

Zone numbers are logical representations. For example, if two Azure subscribers both deploy a VM into Zone 1 in their own subscriptions, that doesn't mean those VMs exist in the same physical Azure data center. Additionally, because of the distance there can be some extra latency introduced into zonal deployments. You should test the latency between your VMs to ensure that the latency meets performance targets. In most cases round-trip latency will be less than 1 millisecond, which supports synchronous data movement in features like availability groups. You can also deploy Azure SQL Database into Availability Zones.

Azure Site Recovery

Azure Site Recovery provides enhanced availability for VMs at the Azure level and can work with VMs hosting SQL Server. Azure Site Recovery replicates a VM from one Azure region to another to create a disaster recovery solution for that VM. As noted earlier, this feature doesn't know that SQL Server is running in the VM and knows nothing about transactions. While Azure Site Recovery may meet RTO, it may not meet RPO since it isn't accounting for where data is inside SQL Server. Azure Site Recovery has a stated monthly RTO of two hours. While most database professionals may prefer to use a database-based method for disaster recovery, Azure Site Recovery works well if it meets your RTO and RPO needs.

5. Describe high availability and disaster recovery for PaaS deployments

Describe high availability and disaster recovery for PaaS deployments

Completed

PaaS is different when it comes to availability; you can only configure the options that Azure provides.

For the SQL Server-based options of Azure SQL Database and Azure SQL Managed Instance, the options are active geo-replication (Azure SQL Database only) and auto-failover groups (Azure SQL Database or Azure SQL Managed Instance).

Azure SQL Database has a service level agreement, which guarantees availability of 99.99, meaning nearly no downtime should be encountered. If a node-level problem happens such as hardware failure, a built-in failover mechanism kicks in. All transactional changes to the database are written synchronously to storage upon commit. If a node-level interruption occurs, the database server automatically creates a new node and attaches the data storage.

From an application standpoint, you need to code the necessary retry logic because all connections are dropped as part of spinning up the new node and any in flight transactions are lost. This process is considered a best practice for any cloud application, as they should be designed to handle transient failures.

Azure SQL Database and Azure SQL Managed Instance offer the option to create read replicas. These replicas can be used for activities such as reporting, which helps offload work from the primary database. Additionally, read replicas enhance availability by being located in different regions, ensuring that your data remains accessible even if one region experiences issue.

Database availability

In Azure SQL Database and Azure SQL Managed Instance, you can't set a database state to `OFFLINE` or `EMERGENCY`. If you think about it, `OFFLINE` doesn't make sense, because you can't attach databases. Because you can't use `EMERGENCY`, you can't do emergency mode repair, but you shouldn't have to because Azure manages and maintains the service. Other capabilities, like `RESTRICTED_USER` and dedicated admin connection (DAC), are allowed in Azure SQL Database.

Accelerated Database Recovery (ADR) is built into the engine. With ADR, the transaction log is aggressively truncated and a persisted version store (PVS) is used. This technology allows you to perform a transaction rollback instantly, solving a well-known problem with long-running transactions. It also allows Azure SQL to recover databases quickly.

In Azure SQL Database and Azure SQL Managed Instance, ADR greatly increases general database availability. It's a significant factor in the SLA. For these reasons, ADR is on by default and can't be turned off.

Database consistency

As you learned in the beginning of this module, multiple copies of your data and backups exist both locally and across regions. Regularly, backup and restore integrity checks run. Detection for *lost write* and *stale read* is also in place. You can run `DBCC CHECKDB` (no repair), and `CHECKSUM` is on by default. In the back end, automatic page repair occurs when possible, and there's data integrity error alert monitoring. If there's no impact, repair without notification occurs. If there's an impact, proactive notification is provided.

6. Explore high availability and disaster recovery solution for IaaS

<https://learn.microsoft.com/en-us/training/modules/describe-high-availability-disaster-recovery-strategies/6-explore-iaas-high-availability-disaster-recovery-solution>

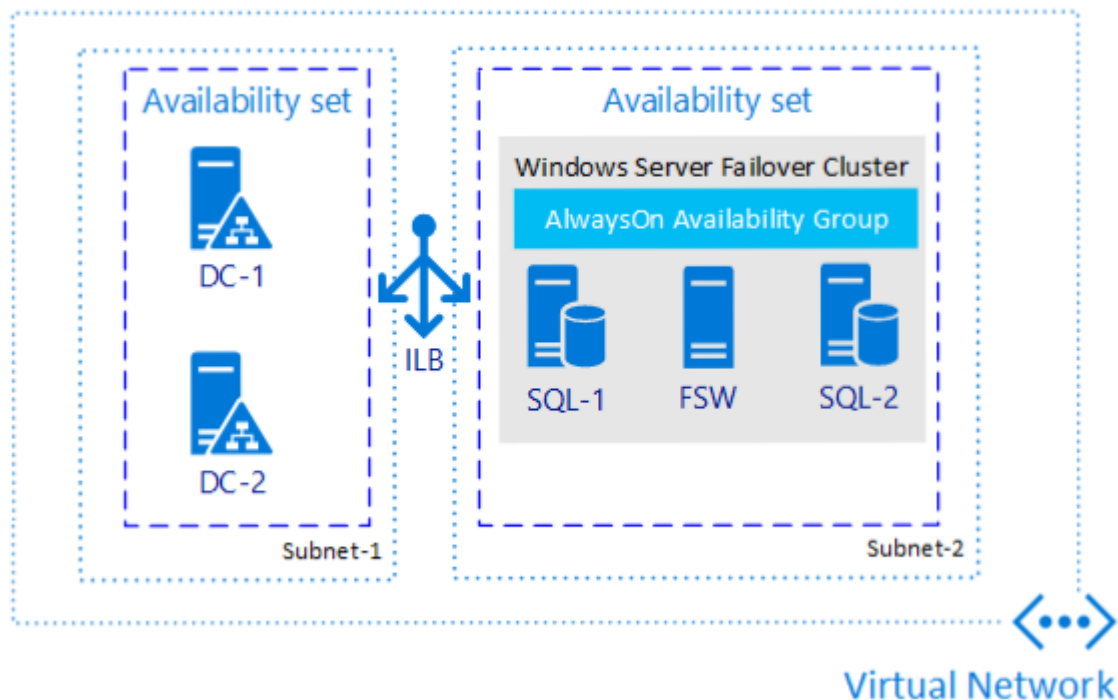
Explore high availability and disaster recovery solution for IaaS

Completed

There are many different combinations of features that could be deployed in Azure for IaaS. This section covers five common examples of SQL Server high availability and disaster recovery (HADR) architectures in Azure.

Single Region High Availability Example 1 – Always On availability groups

If you only need high availability and not disaster recovery, configuring an (availability group) AG is one of the most ubiquitous methods no matter where you're using SQL Server. The following image is an example of what one possible AG in a single region could look like.



Why is this architecture worth considering?

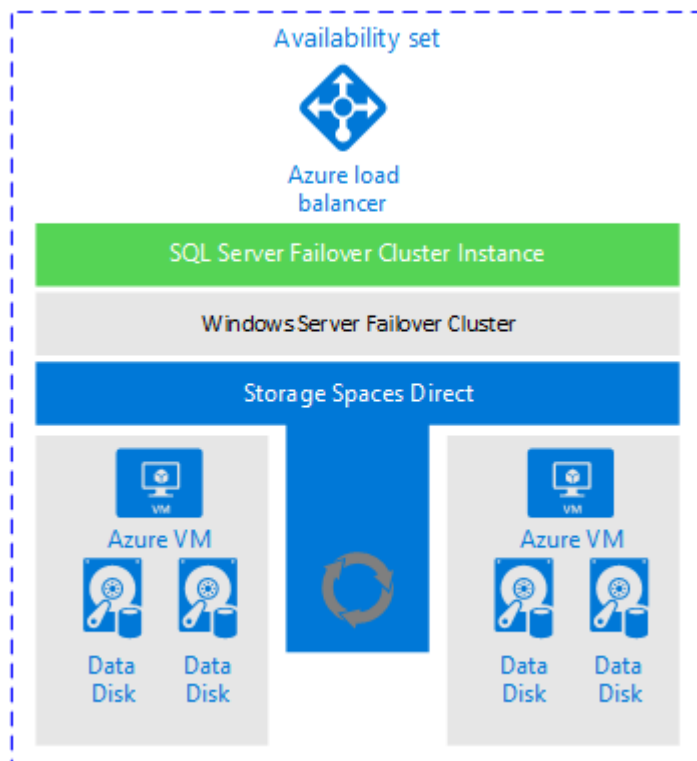
- This architecture protects data by having more than one copy on different virtual machines (VMs).
- This architecture allows you to meet recovery time objective (RTO) and recovery point objective (RPO) with minimal-to-no data loss if implemented properly.
- This architecture provides an easy, standardized method for applications to access both primary and secondary replicas.
- This architecture provides enhanced availability during patching scenarios.
- This architecture needs no shared storage, so there's less complication than when using a failover cluster instance (FCI).

Single Region High Availability Example 2 – Always On Failover Cluster Instance

Until AGs were introduced, FCIs were the most popular way to implement SQL Server high availability. FCIs, however, were designed when physical deployments were dominant. In a virtualized world, FCIs don't provide many of the same protections in the way they would on physical hardware because it's rare for a VM to have a problem. FCIs were designed to protect against things like network card failure or disk failure, both of which would likely not happen in Azure.

Having said that, FCIs do have a place in Azure. They work, and as long as you have the right expectations about what is and isn't provided, an FCI is a perfectly acceptable solution. The following

image shows a high-level view of what an FCI deployment looks like when using Storage Spaces Direct.

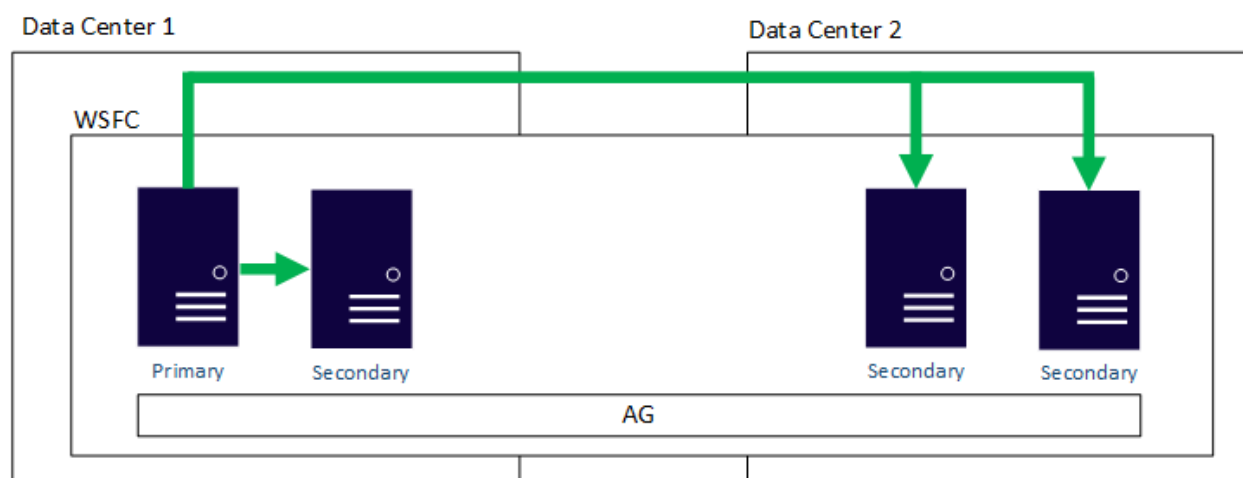


Why is this architecture worth considering?

- FCIs are still a popular availability solution.
- The shared storage story is improving with feature like Azure Shared Disk.
- This architecture meets most RTO and RPO for HA (although DR isn't handled).
- This architecture provides an easy, standardized method for applications to access the clustered instance of SQL Server.
- This architecture provides enhanced availability during patching scenarios.

Disaster Recovery Example 1 – Multi-Region or Hybrid Always On availability group

If you're using AGs, one option is to configure the AG across multiple Azure regions or potentially as a hybrid architecture. This means that all nodes which contain the replicas participate in the same WSFC. This assumes good network connectivity, especially if this is a hybrid configuration. One of the biggest considerations would be the witness resource for the WSFC. This architecture would require AD DS and DNS to be available in every region and potentially on premises as well if this is a hybrid solution. The following image shows what a single AG configured over two locations looks like using Windows Server.



Why is this architecture worth considering?

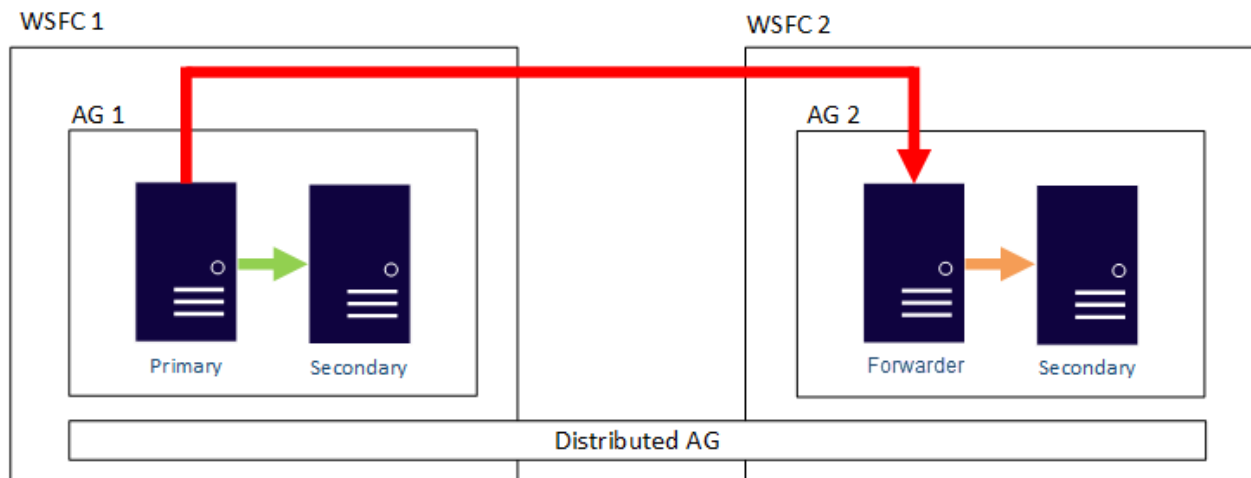
- This architecture is a proven solution; it's no different than having two data centers today in an AG topology.
- This architecture works with Standard and Enterprise editions of SQL Server.
- AGs naturally provide redundancy with extra copies of data.
- This architecture makes use of one feature that provides both HA and D/R

Disaster Recovery Example 2 –Distributed availability group

A distributed AG is an Enterprise Edition only feature introduced in SQL Server 2016. It's different than a traditional AG. Instead of having one underlying WSFC where all of nodes contain replicas participating in one AG as described in the previous example, a distributed AG is made up of multiple AGs. The primary replica containing the read/write database is known as the global primary. The primary of the second AG is known as a forwarder and keeps the secondary replica of that AG in sync. In essence, this is an AG of AGs.

This architecture makes it easier to deal with things like quorum since each cluster would maintain its own quorum, meaning it also has its own witness. A distributed AG would work whether you're using Azure for all resources, or if you're using a hybrid architecture.

The following image shows an example distributed AG configuration. There are two WSFCs. Imagine each is in a different Azure region or one is on premises and the other is in Azure. Each WSFC has an AG with two replicas. The global primary in AG 1 is keeping the secondary of replica of AG 1 synchronized as well as the forwarder, which also is the primary of AG 2. That replica keeps the secondary replica of AG 2 synchronized.

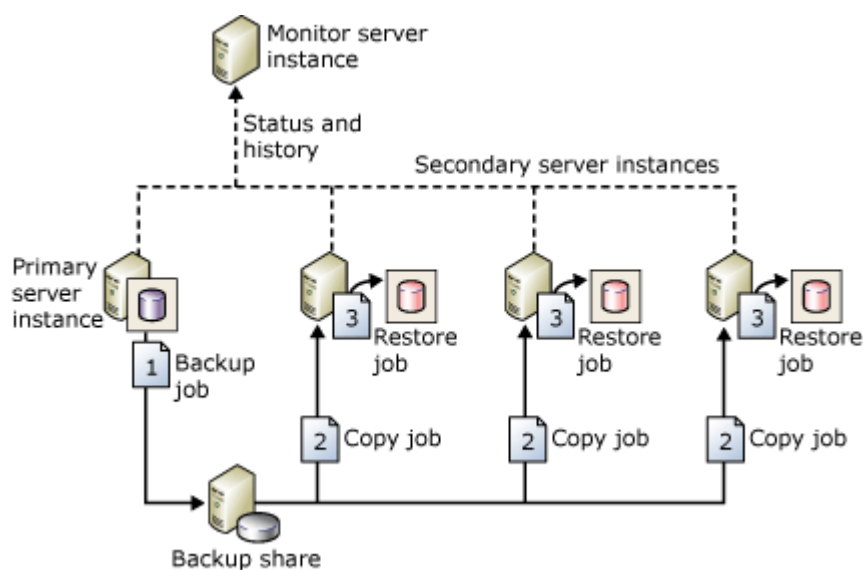


Why is this architecture worth considering?

- This architecture separates out the WSFC as a single point of failure if all nodes lose communication
- In this architecture, one primary isn't synchronizing all secondary replicas.
- This architecture can provide failing back from one location to another.

Disaster Recovery Example 3 – Log shipping

Log shipping is one of the oldest HADR methods for configuring disaster recovery for SQL Server. As described, the unit of measurement is the transaction log backup. Unless the switch to a warm standby is planned to ensure no data loss, data loss will most likely occur. When it comes to disaster recovery, it's always best to assume some data loss even if minimal. The following image shows an example log shipping topology.



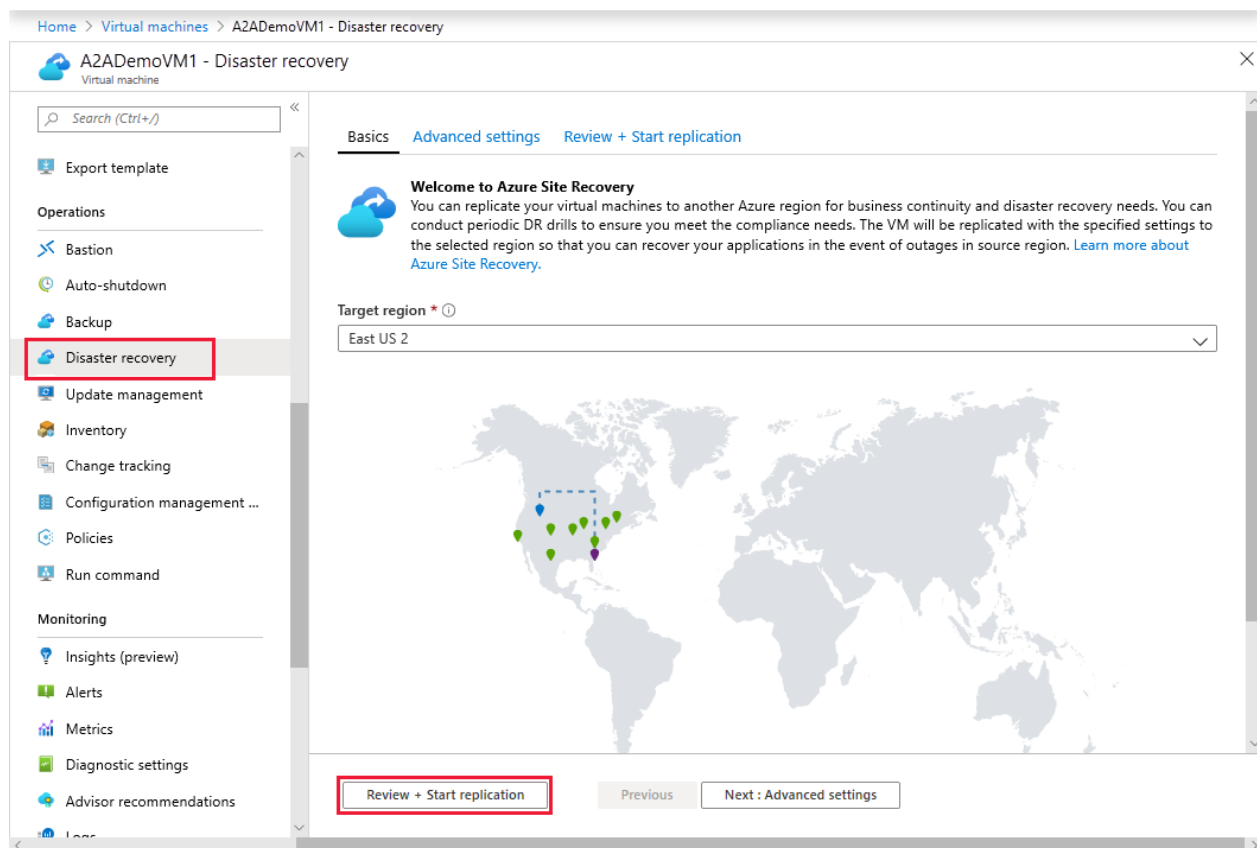
Why is this architecture worth considering?

- Log shipping is a tried-and-true feature that has been around for over 20 years
- Log shipping is easy to deploy and administer since it's based on backup and restore.
- Log shipping is tolerant of networks that aren't robust.
- Log shipping meets most RTO and RPO goals for DR.
- Log shipping is a good way to protect FCIs.

Disaster Recovery Example 4 – Azure Site Recovery

For those who don't want to implement a SQL Server-based disaster solution, Azure Site Recovery is a potential option. However, most data professionals prefer a database-centric approach as it will generally have a lower RPO.

The following image shows where in the Azure portal you'd configure replication for Azure Site Recovery.



Why is this architecture worth considering?

- Azure Site Recovery works with more than just SQL Server.
- Azure Site Recovery may meet RTO and possibly RPO.
- Azure Site Recovery is provided as part of the Azure platform.

7. Describe hybrid solutions

<https://learn.microsoft.com/en-us/training/modules/describe-high-availability-disaster-recovery-strategies/7-describe-hybrid-solutions>

Describe hybrid solutions

Completed

You should now understand recovery time objectives (RTOs) and recovery point objectives (RPOs) as well as the different features both in SQL Server as well as Azure to increase availability, you can put all of that together and design a solution to meet your high availability and disaster recovery (HADR) requirements.

While an architecture can be deployed in one or more Azure regions, many organizations will need or want to have solutions that span both on premises and Azure, or possibly Azure to another public cloud. This type of architecture is known as a hybrid solution.

PaaS solutions by nature aren't designed to allow traditional hybrid solutions. HADR is provided by the Azure infrastructure. There are some exceptions. For example, SQL Server's transactional replication feature can be configured from a publisher located on premises (or another cloud) to an Azure SQL Managed Instance subscriber, but not the other way.

Hybrid solutions are therefore IaaS-based since they rely on traditional infrastructure. Hybrid solutions are useful. Not only can they be used to help migrate to Azure, but the most common usage of a hybrid architecture is to create a robust disaster recovery solution for an on premises system. For example, a secondary replica for an AG can be added in Azure. That means any associated infrastructure must exist, such as AD DS and DNS.

Arguably the most important consideration for a hybrid HADR solution that extends to Azure is networking. Not having the right bandwidth could mean missing your RTO and RPO. Azure has a fast networking option called ExpressRoute. If ExpressRoute isn't something your company can or will implement, configure a secure site-to-site VPN so that the Azure VMs act as an extension of your on-premises infrastructure. Exposing IaaS VMs directly to the public internet is discouraged.

Although not traditionally thought of as hybrid, Azure can also be used as the destination for a database backup and as cold, archival storage for backups.

8. Module assessment

<https://learn.microsoft.com/en-us/training/modules/describe-high-availability-disaster-recovery-strategies/8-knowledge-check>

Module assessment

Completed

9. Summary

<https://learn.microsoft.com/en-us/training/modules/describe-high-availability-disaster-recovery-strategies/9-summary>

Summary

Completed

Deploying the right HADR solution in Azure is more than just picking a feature. A solution must meet the requirements, and specifically, the RTO and RPO expected by the business. Depending on the path (IaaS or PaaS), there could be multiple options to choose from.

Now that you've reviewed this module, you should be able to:

- Explain recovery time objective (RTO)
- Explain recovery point objective (RPO)
- Explain the available HADR options for both IaaS and PaaS
- Devise a HADR strategy