

Domain 6: Plan and implement a high availability and disaster recovery environment

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:

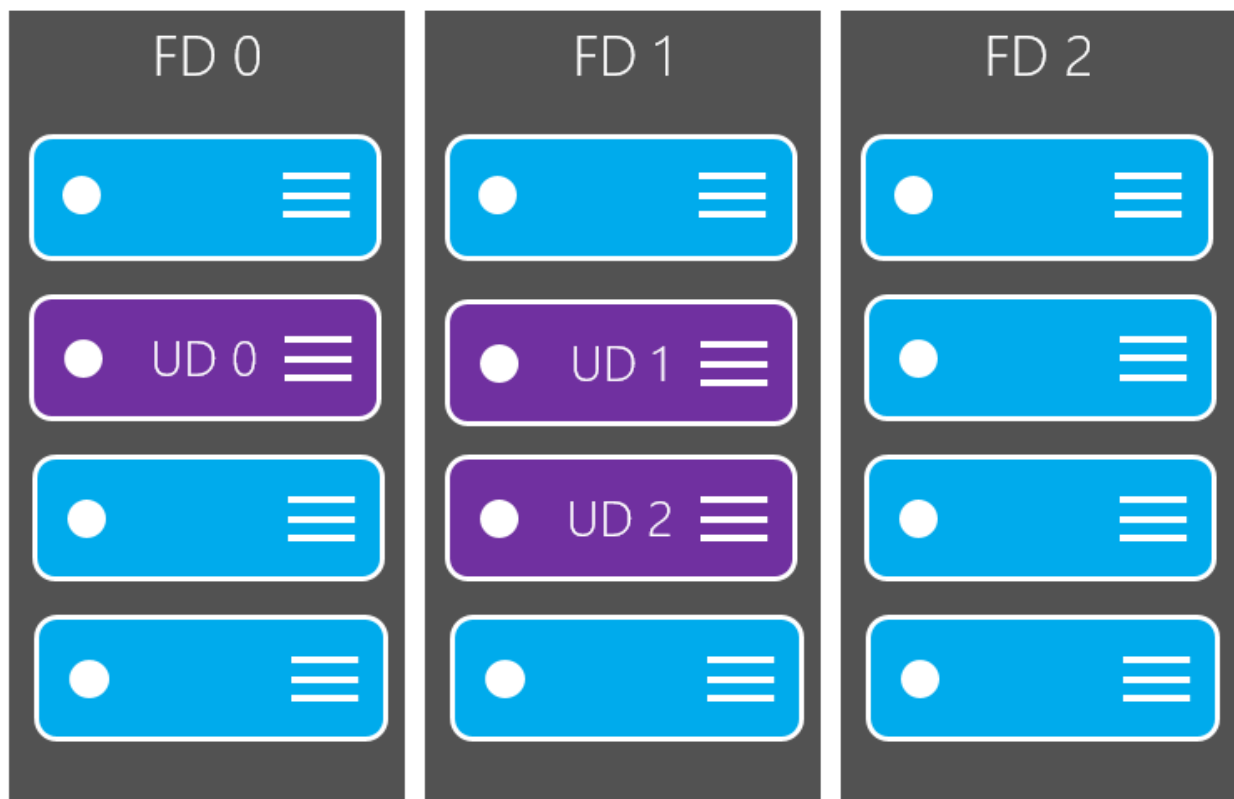
- Availability Sets

- 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>

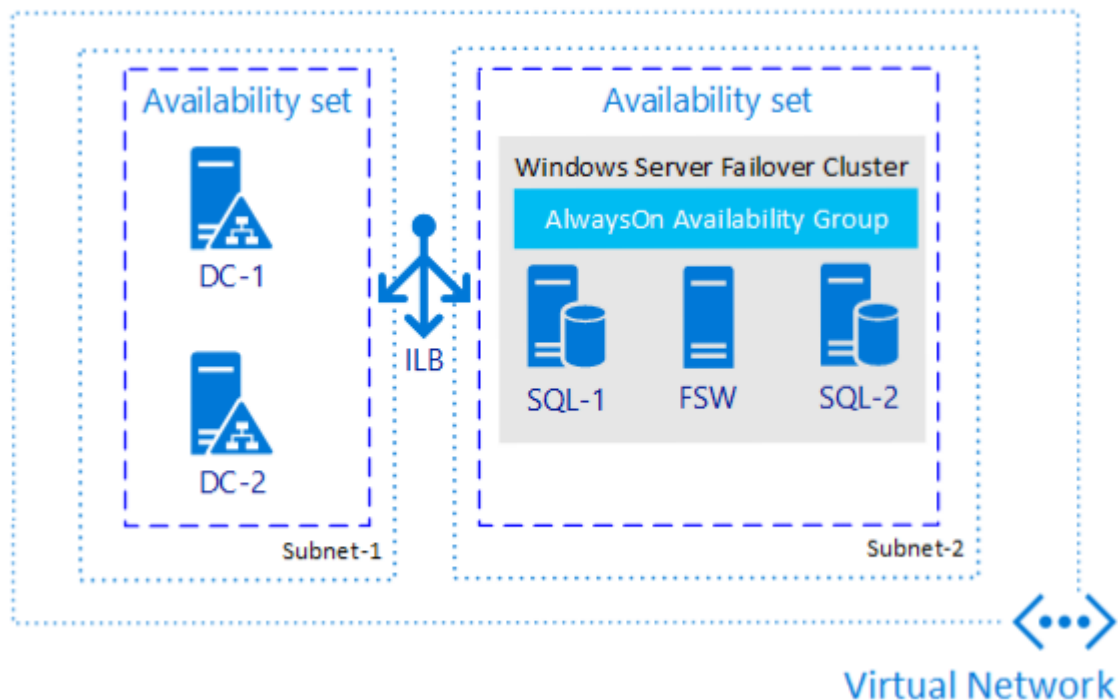
Explore high availability and disaster recovery solution for IaaS

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?

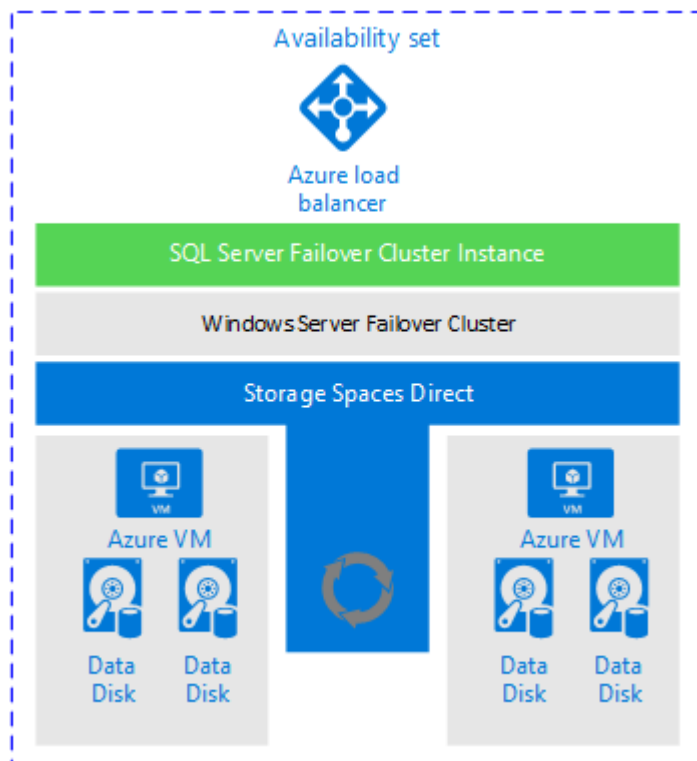
- This architecture protects data by having more than one copy on different virtual machines (VMs).
- This architecture allows you to meet recovery time objective (RTO) and recovery point objective (RPO) with minimal-to-no data loss if implemented properly.
- This architecture provides an easy, standardized 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.

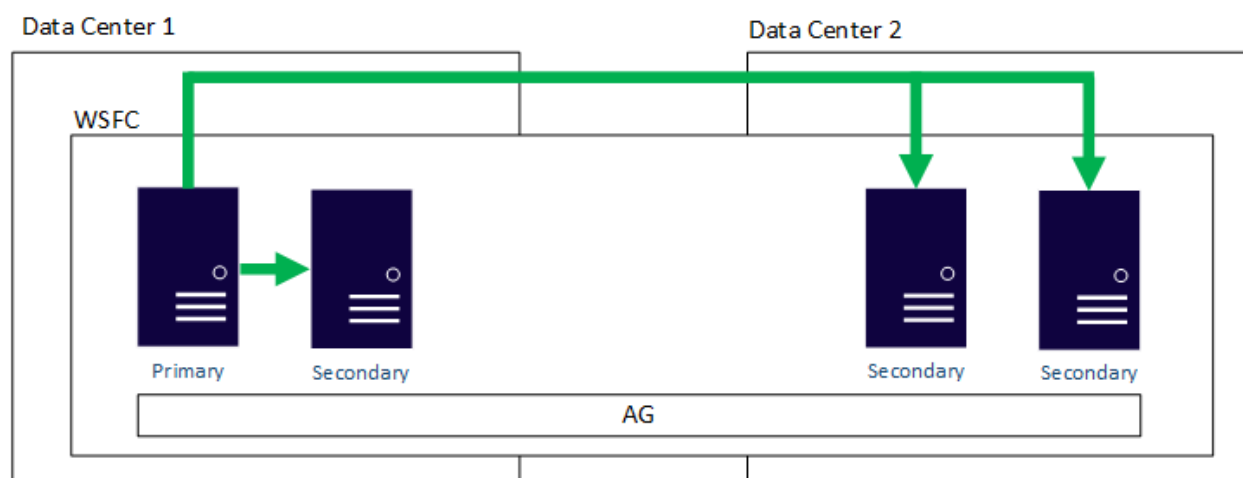


Why is this architecture worth considering?

- 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?

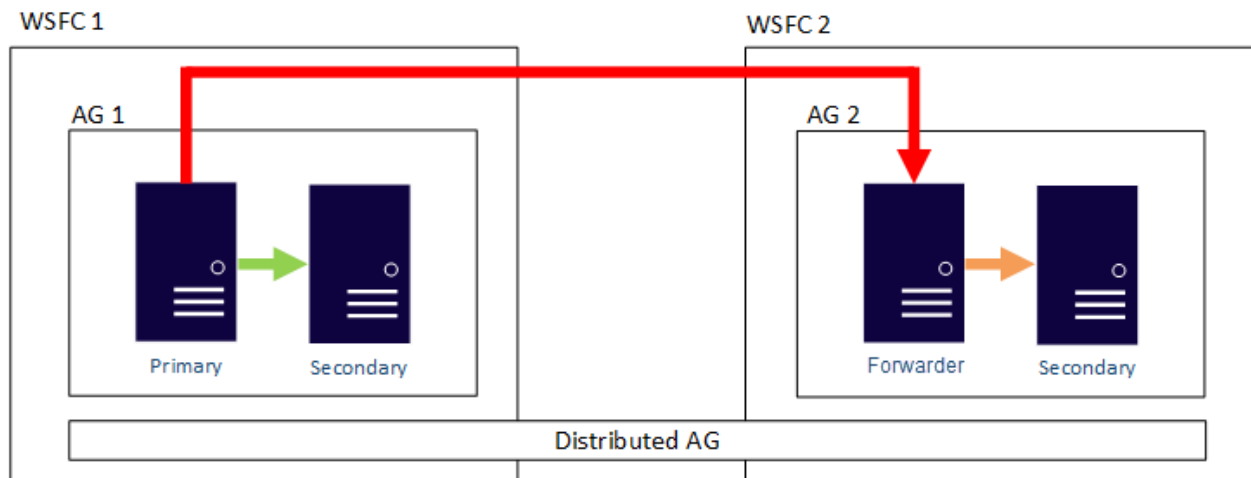
- This architecture is a proven solution; it's no different than having two data centers today in an AG topology.
- This architecture works with Standard and Enterprise editions of SQL Server.
- AGs naturally provide redundancy with extra copies of data.
- This architecture makes use of one feature that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

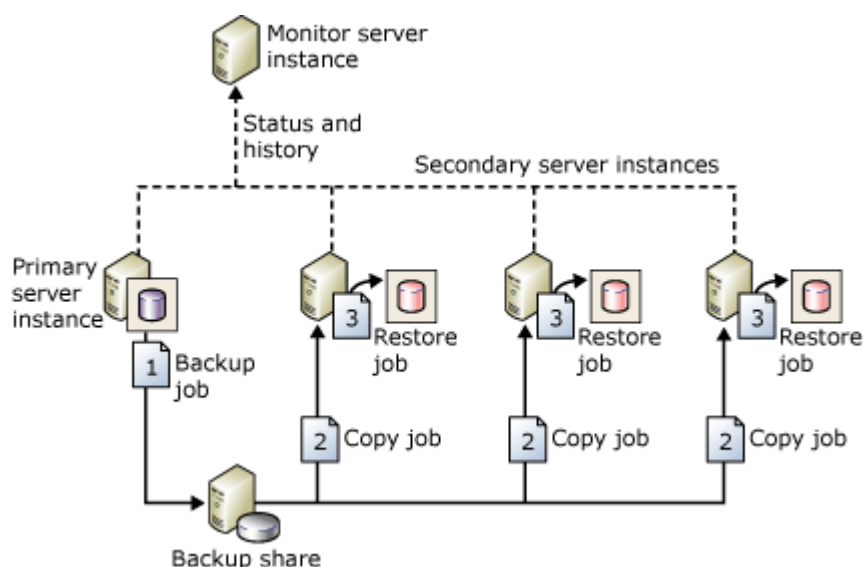


Why is this architecture worth considering?

- 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.



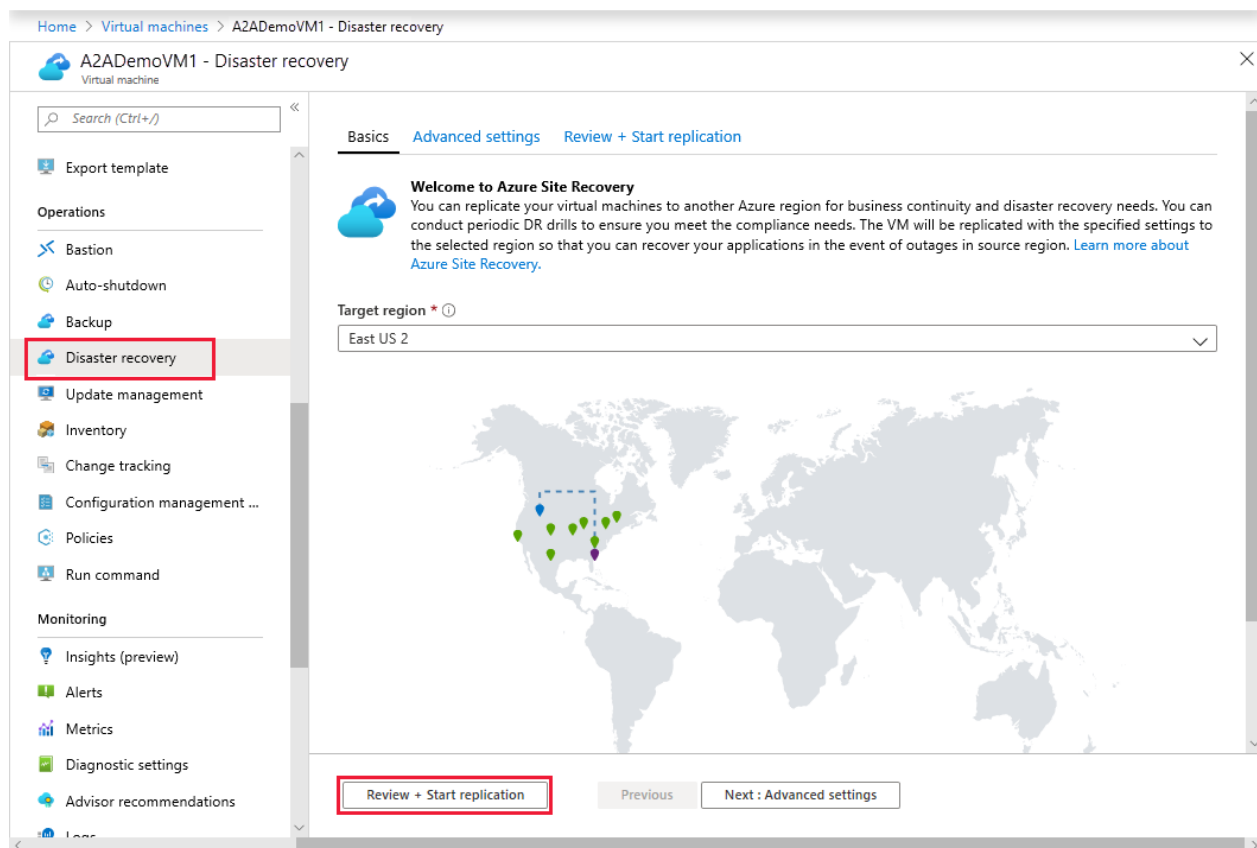
Why is this architecture worth considering?

- 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

Explore IaaS and PaaS solutions for high availability and disaster recovery

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/explore-iaas-paas-platform-tools-for-high-availability-disaster-recovery/1-introduction>

Introduction

Completed

When deploying your database solution using Infrastructure as a Service (IaaS), there are important considerations for implementing high availability and disaster recovery (HADR). For example, if you plan to use an Availability Group (AG), you need to understand how deploying a Windows Server Failover Cluster (WSFC) differs in this context. Are there specific considerations for AGs that differ from an on-premises solution?

On the other hand, implementing a Platform as a Service (PaaS)-based HADR solution is distinct from IaaS. In IaaS, the configuration is managed at the Azure level, whereas in PaaS, the solutions that ensure your applications and data are highly available and meet your Recovery Time Objectives (RTOs) and Recovery Point Objectives (RPOs) are configured within the database or database server itself.

By the end of this module, you'll understand what is needed to be able to deploy IaaS database platform solutions in Azure.

2. Describe failover clusters in Windows Server

<https://learn.microsoft.com/en-us/training/modules/explore-iaas-paas-platform-tools-for-high-availability-disaster-recovery/2-describe-failover-clusters-windows-server>

Describe failover clusters in Windows Server

Completed

For all availability configurations of availability groups (AGs), an underlying cluster is required. There are many aspects of setting up a cluster that you need to be aware of.

Considerations for a Windows Server failover cluster in Azure

Deploying a Windows Server Failover Cluster (WSFC) in Azure is similar to configuring one on-premises, but there are some Azure-specific considerations to keep in mind. This section focuses on those unique aspects.

One of the most critical components is the witness resource, which is essential for the quorum mechanism that keeps the WSFC operational. If quorum is lost, the WSFC goes down, taking any Availability Group (AG) or Failover Cluster Instance (FCI) with it. The witness resource can be a disk, file share (SMB 2.0 or later), or cloud witness. It's recommended to use a cloud witness, especially for solutions spanning multiple Azure regions or hybrid environments, as it's fully Azure-based. The cloud witness feature is available in Windows Server 2016 and later.

Another consideration is the Microsoft Distributed Transaction Coordinator (DTC or MSDTC). While not all applications use DTC, those that do require it to be clustered if deploying an AG or FCI. Clustering DTC necessitates a shared disk, even if one isn't otherwise required, as in the case of an AG.

Most WSFC deployments require both Active Directory Domain Services (AD DS) and DNS; FCIs always do. AGs can be configured without AD DS but still need DNS. In Windows Server 2016, there's a variant called a Workgroup Cluster, which can be used with AGs only.

The WSFC itself needs a unique name in the domain (and DNS) and requires an object in AD DS called the cluster name object (CNO). Any named entity created within the WSFC needs a unique name and at least one IP address. If the configuration remains within a single region, IP addresses are in a single subnet. If the WSFC spans multiple regions, multiple IP addresses are associated with the WSFC and any AGs that are part of it.

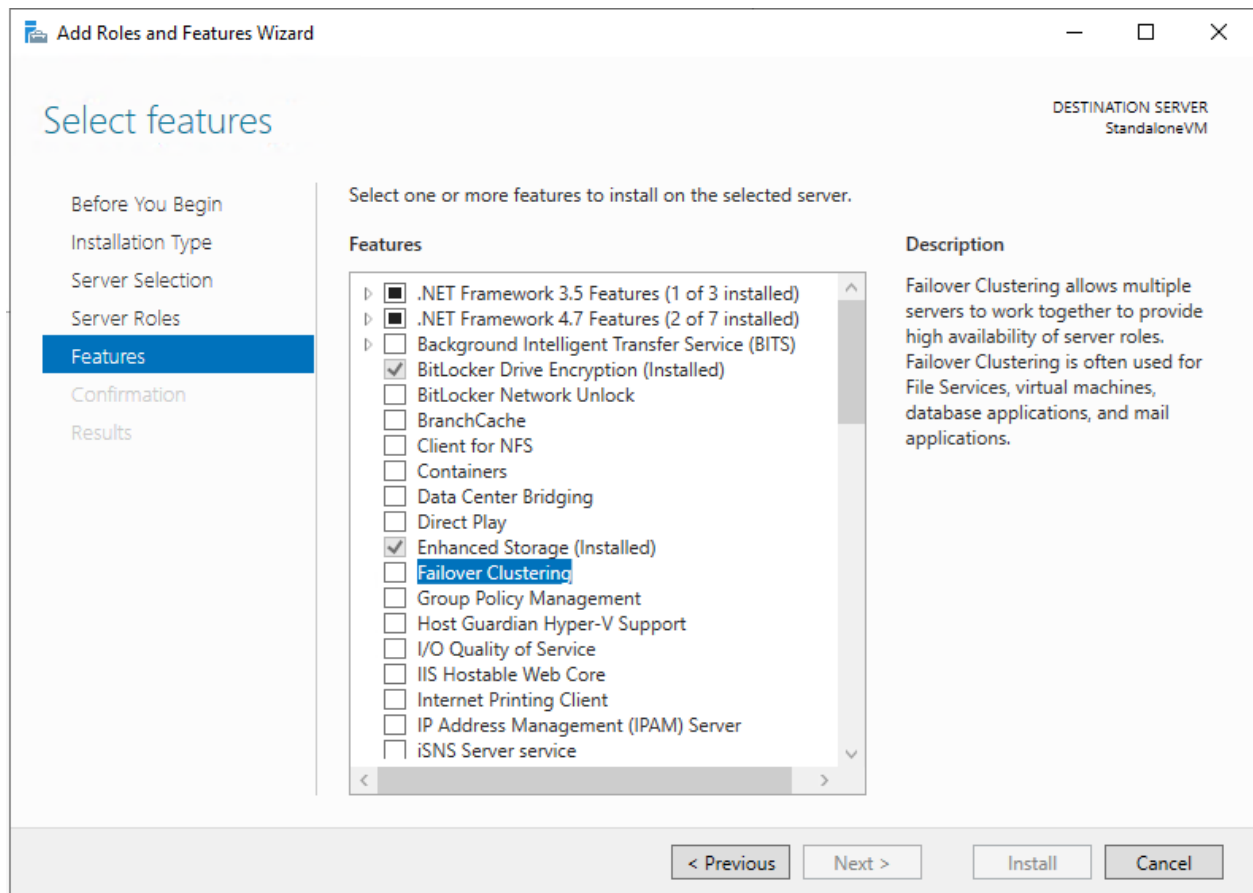
A typical Azure-based WSFC will only require a single virtual network card (vNIC). Unlike on-premises setups, the IP address for the vNIC must be configured in Azure, not within the VM. Inside the VM, it appears as a dynamic (DHCP) address, which is expected. However, cluster validation generates a warning that can be safely ignored.

Considerations for the WSFC's IP address also differ from on-premises setups. There's no way to reserve the IP address at the Azure level since it's fully maintained within the guest configuration. This means that if subsequent resources in Azure use DHCP, you must check for conflicts to avoid issues.

Enable the failover clustering feature

To configure a WSFC, you need to enable the underlying Windows feature on each node that participates in the cluster. You can accomplish this using either the Server Manager or PowerShell.

In **Server Manager**, failover clustering must be enabled in the **Add Roles and Features Wizard**.



Run the following command to enable the feature via PowerShell, which will also install the utilities to administer a WSFC:

```
Install-WindowsFeature Failover-Clustering -IncludeManagementTools
```

Once the feature is installed on the servers targeted as WSFC nodes, you must then validate the configuration.

Validate the cluster

For a WSFC to be considered supported, it must pass cluster validation. Cluster validation is a built-in process that checks the nodes via a series of tests to ensure that the entire configuration is suitable for clustering. After the validation is complete, each of the tests will come back with an error, a warning, a pass, or a message that the test isn't applicable. Warnings are acceptable if that condition is expected in your environment. If not, it should be addressed. All errors must be resolved.

Validation can be run via Failover Cluster Manager or by using the Test-Cluster PowerShell cmdlet.

For FCIs, these tests also check the shared storage to ensure that the disks are configured correctly. For AGs with no shared storage, in Windows Server 2016 and later, the results come back as not applicable. For Windows Server 2012 R2, the disk tests show a warning when there are no shared disks. This state is expected.

Once the configuration is deemed valid by the cluster validation process, you can then create the WSFC.

Create a Windows Server failover cluster

To create a WSFC properly in Azure, you can't use the Wizard in Failover Cluster Manager for FCIs or AGs deployed using Windows Server 2016 and earlier. Due to the DHCP issue mentioned earlier, currently the only way to create the WSFC is to use PowerShell and specify the IP address. This could change in future versions of Windows Server.

For a configuration that has shared storage, use the following syntax:

```
New-Cluster -Name MyWSFC -Node Node1,Node2,...,NodeN -StaticAddress w.x.y.z
```

Where *MyWSFC* is the name of the WSFC you want, *Node1*, *Node2*,..., *NodeN* are the names of the nodes that participate in the WSFC, and *w.x.y.z* is the IP address of the WSFC. If you're creating a WSFC that spans multiple subnets, you can specify more than one IP address for **-StaticAddress** separated by commas.

For a configuration that doesn't have shared storage, add the **-NoStorage** option.

```
New-Cluster -Name MyWSFC -Node Node1,Node2,...,NodeN -StaticAddress w.x.y.z -NoStorage
```

For a Workgroup Cluster that will only use DNS, the syntax is also slightly different.

```
New-Cluster -Name MyWSFC -Node Node1,Node2,...,NodeN -StaticAddress w.x.y.z -NoStorage
```

A distributed network name is one that creates just a network name, but the IP address is tied to the underlying nodes. You no longer need to specify an IP address as shown above if it isn't needed or necessary. A distributed network name is for the WSFC's name only; it can't be used with the name of an AG or FCI.

The WSFC creation mechanism detects if it's running in Azure or not and will create the cluster using a distributed network name unless you specify it differently. However, there are cases you may want to consider deploying a WSFC traditionally using PowerShell. For this, you need to add one more option: **-ManagementPointNetwork** with a value of **Singleton**. An example would look like this:

```
New-Cluster -Name MyWSFC -Node Node1,Node2,...,NodeN -StaticAddress w.x.y.z -NoStorage
```

For a Workgroup Cluster, you need to ensure the name and IP address are in DNS for any name or IP address created in the context of the WSFC such as the WSFC itself, an FCI name and IP address, and

an AG listener name and IP address.

With Windows Server 2019, Microsoft changed how WSFCs are created by default in Azure. Instead of creating a network name and an IP address, it uses a distributed network name.

Understand failover cluster instance

Failover Cluster Instances uses Windows Server Failover Clustering (WSFC) functionality to provide high availability through redundancy at the instance level.

An FCI is an instance of SQL Server that is installed across WSFC nodes and, possibly, across multiple subnets. On the network, an FCI appears as a single local instance that uses a virtual network name. Therefore, it becomes transparent for the application, since it doesn't know which node the instance is currently running on. As a result, there's no need to reconfigure clients and applications during or after a failover.

The multi-subnet support eliminates the need for a virtual LAN, which increases the protection and flexibility of a multi-subnet FCI.

When you use multi-subnet, each subnet of the FCI is assigned a virtual IP address, and a failover occurs as follows.

- Virtual network names on DNS server are updated with virtual IP addresses corresponding to the respective subnets
- After a multi-subnet failover, clients and applications can then connect to the FCI via the same virtual network name.

To learn more about how multi-subnet works on FCI, see [SQL Server Multi-Subnet Clustering \(SQL Server\)](#).

Distributed Network Name (DNN)

The distributed network name (DNN) replaces the virtual network name (VNN) as the connection point when used with FCI. As a result, the VNN no longer requires an Azure Load Balancing service.

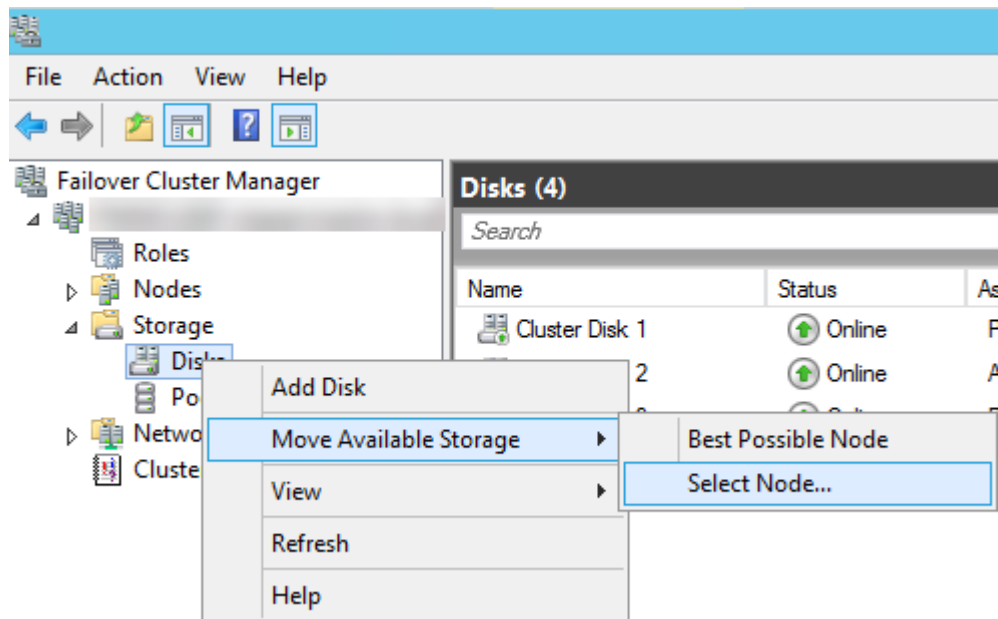
The VNN still exists in an FCI deployment, but the clients connect to the DNN DNS name instead of the VNN name.

To learn how to configure a distributed network name for FCI, see [Configure a DNN for failover cluster instance](#).

Test the Windows Server Failover Cluster

After creating the WSFC, but before configuring FCIs or AGs, ensure the WSFC is functioning correctly. For clusters requiring shared storage, like those supporting a SQL Server Failover Cluster Instance, test that all nodes can access and take ownership of the shared storage as they would during a failover.

You can do this by selecting **Storage**, then **Disks** in Failover Cluster Manager, as shown in the following image. If you right-click on each clustered disk device, you'll see the option **Move Available Storage**. If you choose the **Select Node...** option, you can force the storage to fail over to the other nodes in the cluster to confirm this functionality.



Now that you understand the considerations for a WSFC in Azure, let's look at the considerations for deploying the Always On features on top of a WSFC in Azure.

3. Configure Always-on availability groups

<https://learn.microsoft.com/en-us/training/modules/explore-iaas-paas-platform-tools-for-high-availability-disaster-recovery/3-configure-always-availability-groups>

Configure Always-on availability groups

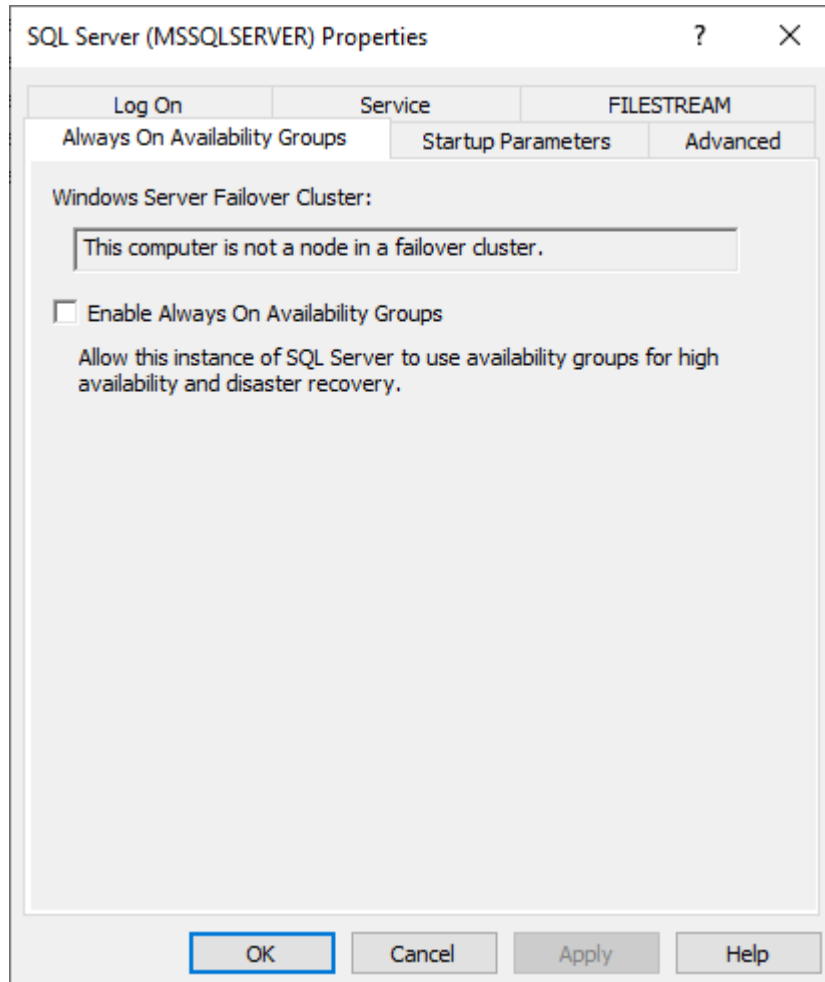
Completed

Configuring an Availability Group (AG) in Azure is quite similar to doing so on-premises, with most considerations remaining the same, such as initializing secondary replicas. Azure-specific considerations, like the need for an Internal Load Balancer (ILB), were discussed earlier. Just like with the Windows Server Failover Cluster (WSFC), you can't reserve the listener's IP address in Azure. Therefore, you must ensure that no other resource claims it, as this could lead to network conflicts and availability issues.

Avoid placing any permanent databases on temporary storage. All virtual machines (VMs) participating in an AG should have identical storage configurations. It's crucial to size disks

appropriately based on the application's workload to ensure optimal performance.

Before an AG can be configured, the AG feature must be enabled. This can be done in SQL Server Configuration Manager as shown in the image below or via PowerShell with the cmdlet [Enable-SqlAlwaysOn](#). Enabling the AG feature will require a stop and start of the SQL Server service.



Create the Availability group

Creating an AG in Azure is the same as it is on premises. SQL Server Management Studio (SSMS), T-SQL, or PowerShell can be used.

The only difference is that whether or not you create the listener as part of the initial AG configuration, as the listener requires the creation of an Azure load balancer and has some extra configuration in the WSFC related to the load balancer.

Create an Internal Azure load balancer

Once the listener is created, an internal load balancer (ILB) must be used. Without configuring an ILB, applications, end users, administrators, and others can't use the listener unless they were connected to the VM that hosts an AG's primary replica.

You can use a basic or a standard load balancer depending on your preference or configuration. Deployments using Availability Zones require the use of a standard load balancer. The listener IP address and the port used for the listener are what is configured as part of the load balancer. A single load balancer supports more than one IP address, so depending on your standards, you may not need a different load balancer for each AG configured on those nodes.

Another consideration for the load balancer is the probe port. Without the probe port, the listener won't work properly as it isn't enough just to create the load balancer. Each IP address that will use the load balancer requires a unique probe port. If there are going to be two listeners, there must be two probe ports. Probe ports are high numbers such as 59999.

The probe port is set on the IP address(es) associated with the listener with the following syntax:

```
Get-ClusterResource IPAddressResourceNameForListener | Set-ClusterParameter ProbePort
```

Adding the probe port will require a stop and start of the IP address of the listener, which will also temporarily cause the AG to be brought offline, so it's best to get this configured before deploying in production.

If you have a multi-subnet configuration, a load balancer will need to be configured in each subnet (whether or not the other subnet is deployed to different region) and the probe port for that region associated with the IP resource for that subnet in the WSFC.

If you can't directly connect to the listener, then you need to use the PowerShell cmdlet `Test-NetConnection` to verify that it's configured correctly. The syntax is as follows:

```
Test-NetConnection NameOrIPAddress -Port PortNumber
```

You should run this command from somewhere other than the VM that is hosting the primary replica.

Some environments may also require that the IP address for the WSFC and selected ports (such as 445) must be accessible for administration or other purposes, which mean to configure those as part of the same or a different load balancer.

Once the load balancer is confirmed to be working, you can begin to test AG failover and connectivity to the AG via the listener.

Distributed availability groups

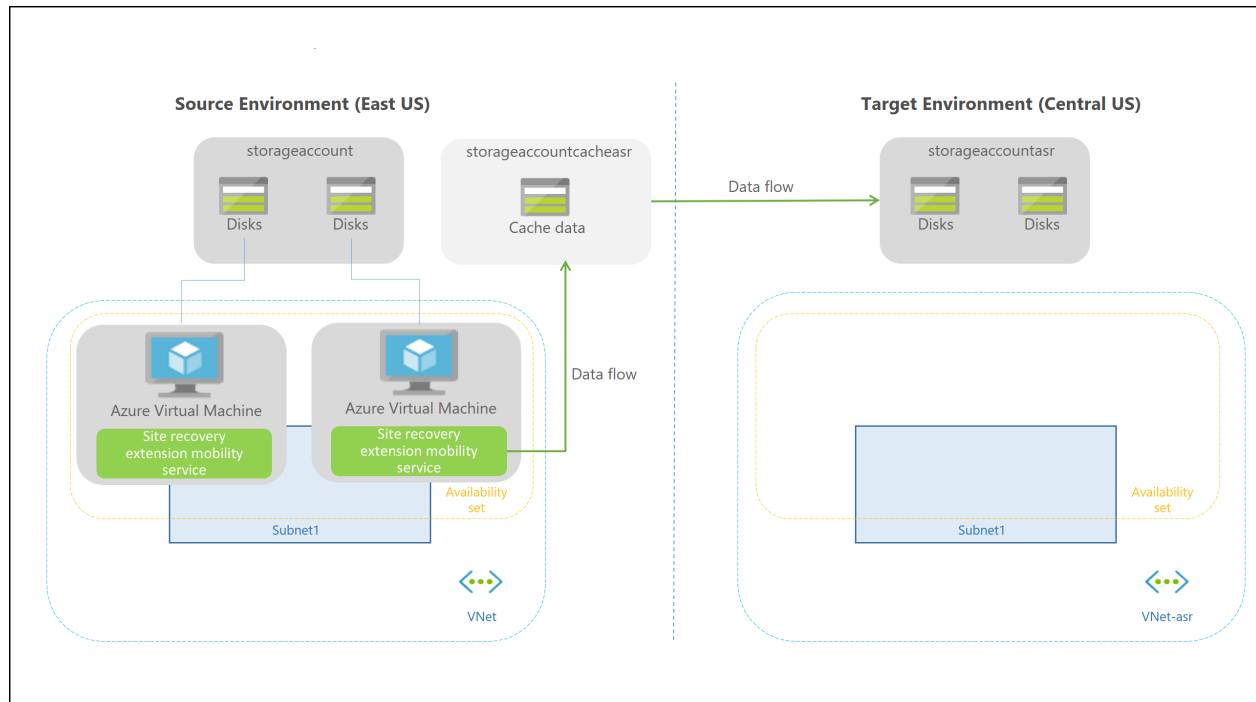
Planning for and configuring a distributed AG is the same on premises as it is in Azure, with any Azure-specific considerations for the individual AGs. The main difference between an on-premises configuration and an Azure configuration for a distributed AG is that as part of the load balancer configuration in each region, the endpoint port for the AG needs to be added. The default port is 5022.

A traditional availability group has resources configured in a Windows Server Failover Cluster (WSFC) or if on Linux, Pacemaker. A distributed availability group does not need WSFC or Pacemaker,

everything about it is maintained within SQL Server.

Azure Site Recovery

Azure Site Recovery is an option that work with the Virtual Machine, whether or not SQL Server is running inside of it. It works with SQL Server but isn't designed specifically to account for nuances that may be required when you have a specific RPO. The disks of a VM configured to use Azure Site Recovery are replicated to another region. This replication can be seen in the image below, noted by the "Data flow" arrow.



This means that all changes to a disk are replicated as soon as they occur, but this process knows nothing of database transactions. This is why recovering to a specific data point may not be possible with Azure Site Recovery in the same way it is for a SQL Server-centric solution such as when using an AG.

If it isn't possible to deploy one of the in-guest options for IaaS solutions, Azure Site Recovery is a viable option to manage disaster recovery.

Additionally, Azure Site Recovery can potentially protect you against ransomware. If infected, you could roll the VM back to a point before the infection was introduced. That could also mean data loss from a SQL Server perspective, but some data loss, especially in this case, may be more than acceptable. Up and running is often better than down for hours, days, or weeks trying to remove ransomware from your network.

The key things to know when replication is enabled on a VM:

- There's a Site Recovery Mobility extension configured on the VM.

- Changes are sent continually unless Azure Site Recovery is unconfigured or replication is disabled
- Crash consistent recovery points are generated every five minutes, and application-specific recovery points are generated according to what is configured in the replication policy.

The screenshot displays the Azure Site Recovery console. On the left, the 'Replication policies' pane shows a table with two policies: 'Tier1-policy-failback' and 'Tier1-policy'. The 'Tier1-policy' is selected, showing it has a source type of 'VMware / Physical machines', a target type of 'Azure', and a status of 'Not in use'. On the right, the 'Tier1-policy' configuration pane is shown, detailing the replication settings and associated configuration servers.

NAME	SOURCE TYPE	TARGET TYPE	STATUS
Tier1-policy-failback	Azure	VMware	Not in use
Tier1-policy	VMware / Physical machines	Azure	Not in use

Replication settings	
Source type	VMware / Physical machines
Target type	Azure
RPO threshold	15 Minutes
Recovery point retention	24 Hours
App consistent snapshot frequency	60 Minutes

Associated Configuration Servers	
NAME	ASSOCIATION STATUS
V2AGNK-CS	Associated

For SQL Server, the **App consistent snapshot frequency** value is what you may want to adjust to reduce your RPO. However, due to the nature of how Azure Site Recovery works – it's using Volume Shadow Service (VSS) – lowering this value could potentially cause problems for SQL Server since there's a brief freeze and thaw of I/O when the snapshots are taken. The impact of the freeze and thaw could be magnified if other options such as an AG are configured. Most won't encounter issues, but if Azure Site Recovery interferes with SQL Server, you may want to consider other availability options.

If multiple VMs are part of an overall solution, they can be replicated together to create a shared crash- and application-consistent recovery points. This is known as multi-VM consistency and will affect performance. Unless VMs must be restored in this way, it's recommended not to configure this option.

One major benefit of Azure Site Recovery is that you can test disaster recovery without needing to bring down production.

A consideration for Azure Site Recovery is if there's a failover to another region, the replica VMs aren't protected when they're brought online. They'll have to be reprotected.

4. Describe active geo-replication for Azure SQL Database

Describe active geo-replication for Azure SQL Database

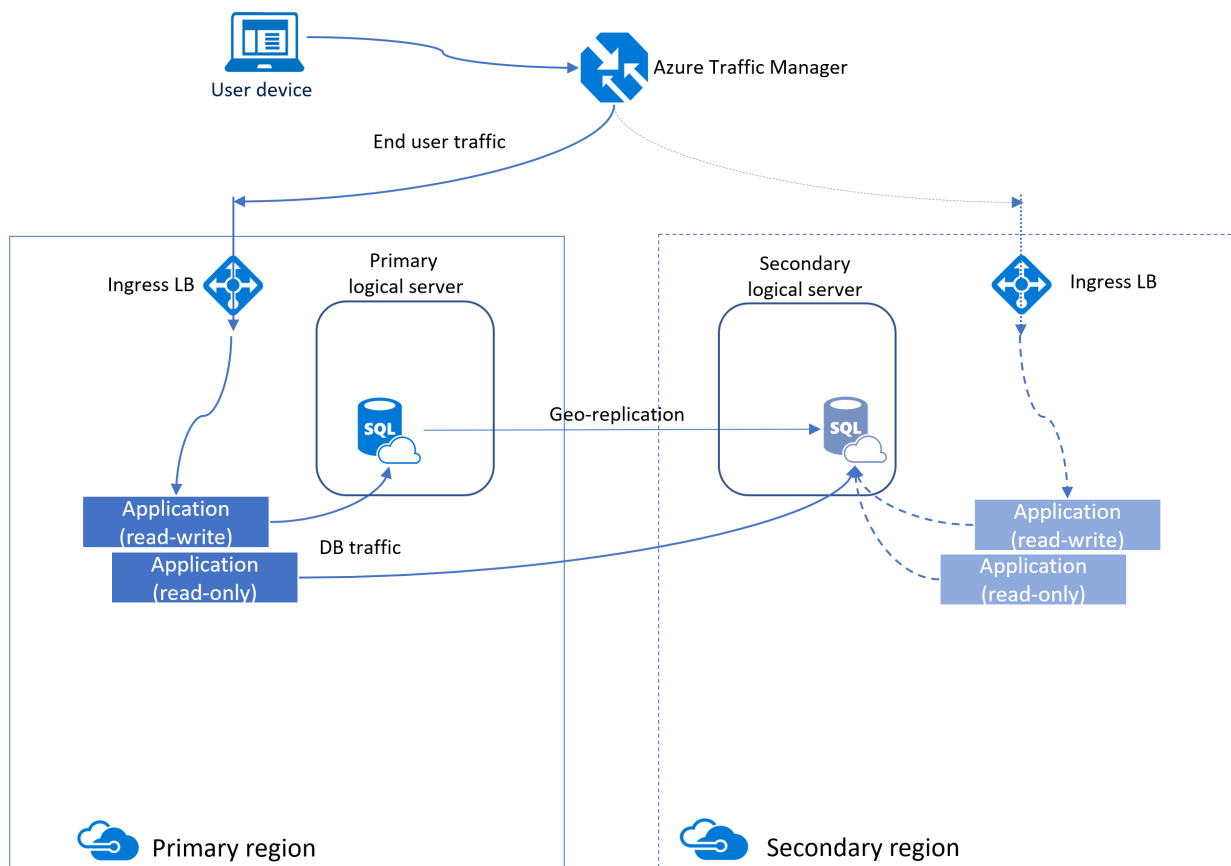
Completed

One method to increase availability for Azure SQL Database is to use active geo-replication. Active geo-replication creates a secondary database replica in another region that is asynchronously kept up to date.

This replica is readable, similar to an Availability Group in IaaS. Underneath the surface, Azure uses Availability Groups to maintain this functionality, which is why some of the terminologies are similar (primary and secondary logical servers, read-only databases, etc.).

Active geo-replication provides business continuity by allowing customers to programmatically or manually failover primary databases to secondary regions during major disaster.

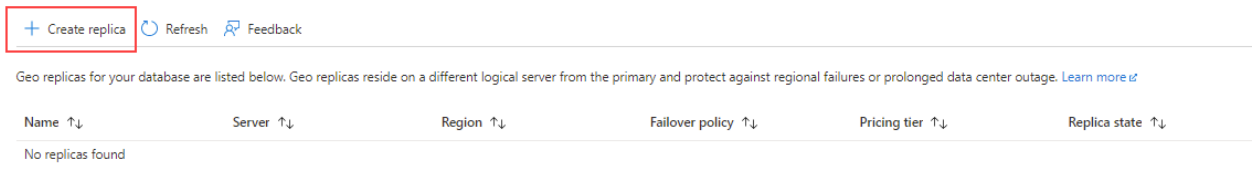
Azure SQL Managed Instance doesn't support active geo-replication. You must use auto-failover groups instead, which will learn on the next unit.



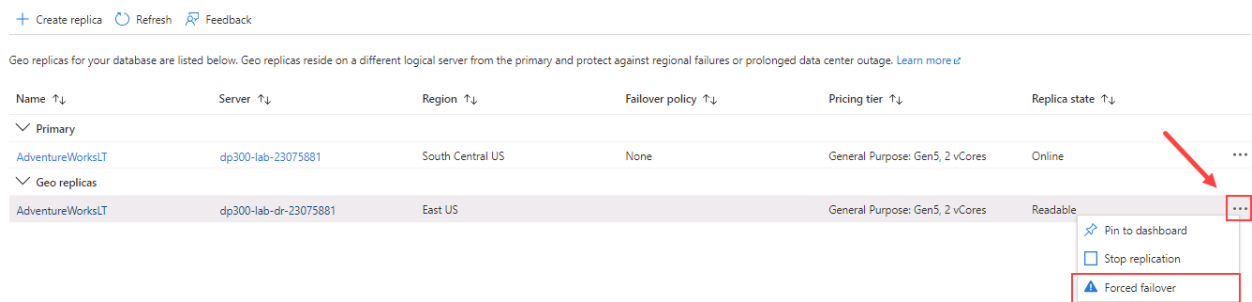
All the databases involved in a geo-replication relationship are required to have the same service tier.

Furthermore, in order to avoid replication overhead due to a large write workload that can affect the replication performance, it's recommended that the geo-secondary is configured with the same compute size as the primary.

As we can see above, you can manually configure geo-replication for Azure SQL Database by accessing the blade for the database, in **Data management** section, selecting **Replicas**, and then **+ Create replica**.



After the secondary replica is created, you can manually fail over your secondary replica. The roles switch with the secondary becoming the new primary, and the old primary the secondary.



Cross subscription geo-replica

Some scenarios require you as a DBA to configure a secondary replica on a different subscription than the primary database. Cross subscription geo-replication is a feature that allows you to perform this task.

Cross subscription geo-replication is only available programmatically.

To learn more about the steps required to configure a cross subscription geo-replication, see [Cross-subscription geo-replication](#).

5. Monitor availability

Monitor availability

Completed

Now that you know about the possibilities, you need to create a strategy for the specific workload that your Azure SQL database or Azure SQL managed instance is a part of.

Make the right choices

A large part of creating a strategy is stepping back and thinking about the requirements of your workload. Here are some questions to consider:

- Do you need long-term backups? Or is 1-35 days long enough?
- What are your RTO and RPO needs?
- Based on the SLA, what service tier makes the most sense?
- Do you need Availability Zones?
- Do you need geo-replicated HADR or failover groups?
- Is your application ready?

The answers to these questions help you narrow down the configuration you should deploy to meet your availability requirements.

The last question is often overlooked by the data professional: *Is your application ready?* This consideration is crucial to achieving the SLA that you want.

You need to make sure your database is meeting your availability requirements, but you also need to be sure your application is meeting those requirements. You also need to make sure the connectivity between the data and the applications meets your requirements. For example, if your application and database are in different regions, that placement increases network latency. Place your application and data as close together as possible. Throughout this module, you've also learned how important implementing retry logic in your applications is for maintaining availability.

Monitor availability

Azure SQL provides several tools and capabilities to monitor certain aspects of availability. These tools include the Azure portal, T-SQL, and interfaces like PowerShell, az CLI, and REST APIs.

The following sections describe some examples of using these tools to monitor availability.

Region and datacenter availability

The availability of regions and datacenters is critical for the availability of a managed instance or a database deployment. *Azure status* and *Azure Service Health* are key to understanding any outages for a datacenter or region, including specific services like Azure SQL.

Azure status is a dashboard that shows any service that's causing problems in any Azure global region. You can use an RSS feed to get notifications of changes to Azure status.

You can view Azure Service Health in the Azure portal. Azure Service Health provides information about service problems, planned maintenance events, health advisories, and health history. You can also set up alerts that notify you through email or SMS of any event that might affect availability.

Instance, server, and database availability

In addition to Azure service events, you can also view the availability of your Azure SQL Managed Instance or Azure SQL Database databases in the Azure portal.

One way to view a possible reason for a managed instance or database to be unavailable is to examine resource health by using the Azure portal or REST APIs.

You can always use standard SQL Server tools like SQL Server Management Studio (SSMS) to connect to a managed instance or database server and check the status of these resources. You can use the tool or T-SQL queries.

Interfaces like Azure CLI can show the status of Azure SQL. For example:

- `az sql mi list` lists the status of managed instances.
- `az sql db list` lists the status of Azure SQL databases.

You can also use PowerShell commands to determine the availability of an Azure SQL database. For example:

- `Get-AzSQLDatabase` gets all the databases on a server and their details, including status.
- REST APIs aren't as easy to use, but you can use them to get the status of managed instances and databases.

Backup and restore history

Azure SQL automatically backs up databases and transaction logs. Standard backup history isn't available, but you can view long-term backup retention history by using the Azure portal or CLI interfaces. Also, in Azure SQL Managed Instance, you can use XEvents to track backup history.

Any database restore that uses point in time restore creates a new database. You can use the Azure Activity Log to view operations that create databases.

Replica status

Replicas are used for Business Critical service tiers. You can view a replica's status by using the DMV `sys.dm_database_replica_states`.

Failover causes

To determine the cause of a failover event for an Azure SQL Managed Instance or database deployment, check the resource health by using the Azure portal or REST APIs.

System Center Management Pack for Azure SQL

System Center provides management packs to monitor Azure SQL Managed Instance and Azure SQL Database. See the [management pack documentation](#) for requirements and details.

6. Explore auto-failover groups for Azure SQL Database and Azure SQL Managed Instance

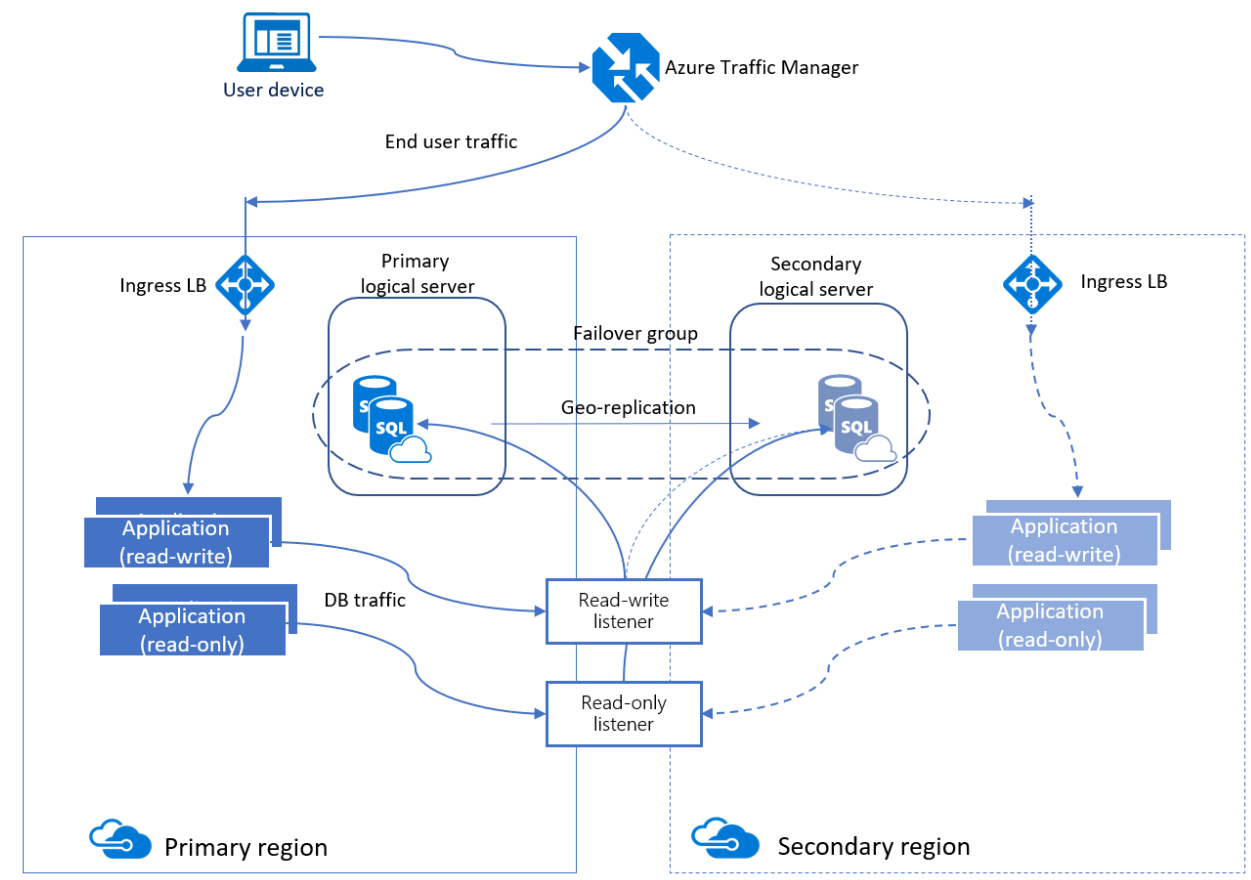
<https://learn.microsoft.com/en-us/training/modules/explore-iaas-paas-platform-tools-for-high-availability-disaster-recovery/6-explore-auto-failover-groups>

Explore auto-failover groups for Azure SQL Database and Azure SQL Managed Instance

Completed

An auto-failover group is an availability feature that can be used with both Azure SQL Database and Azure SQL Managed Instance. Auto-failover groups let you manage how databases are replicated to another region, and let you manage how failover could happen. The name assigned to the auto-failover group must be unique within the *.database.windows.net domain. SQL Managed Instance only supports one auto-failover group.

Auto-failover groups provide AG-like functionality called a listener, which allows both read-write and read-only activity. This functionality can be seen in the following image which is slightly different than the one for active geo-replication. There are two different kinds of listeners: one for read-write and one for read-only traffic. Behind the scenes in a failover, DNS is updated so clients are able to point to the abstracted listener name and not need to know anything else. The database server containing the read-write copies is the primary, and the server that is receiving the transactions from the primary is a secondary.



- **Automatic** – By default, when a failure occurs and it's determined that a failover must happen, the auto-failover group switches regions. The ability to fail over automatically can be disabled.
- **Read-Only** – By default, if a failover occurs, the read-only listener is disabled to ensure performance of the new primary when the secondary is down. This behavior can be changed so that both types of traffic are enabled after a failover.

One auto-failover group can contain one or more databases. The database size and edition are the same on both the primary and secondary. The database is created automatically on the secondary through a process called seeding. Depending on the size of the database, this may take some time. Ensure that you plan accordingly and that you take into account things like the speed of the network.

After you choose a service tier (and consider Availability Zones as applicable), you can consider some other options for getting read-scale or the ability to fail over to another region: geo-replication and auto-failover groups. In SQL Server on-premises, configuring either of these options is something that would take planning, coordination, and time.

Azure SQL has simplified the process significantly. Both geo-replication and auto-failover groups can be set up effortlessly with just a few selections in the Azure portal or a few commands in PowerShell/Azure CLI.

Here are some considerations to help you decide if geo-replication or auto-failover groups are best for your scenario:

Features	Geo-replication	Failover groups
Automatic failover	No	Yes
Fail over multiple databases simultaneously	No	Yes
User must update connection string after failover	Yes	No
SQL Managed Instance support	No	Yes
Can be in same region as primary	Yes	No
Multiple replicas	Yes	No
Supports read-scale	Yes	Yes

7. Exercise: Configure geo replication for Azure SQL Database

<https://learn.microsoft.com/en-us/training/modules/explore-iaas-paas-platform-tools-for-high-availability-disaster-recovery/7-exercise-plan-implement-high-availability-disaster-recovery>

Exercise: Configure geo replication for Azure SQL Database

Completed

In this exercise, you'll learn how to create geo replication for Azure SQL Database.

Note

To complete this exercise, you'll need an [Azure subscription](#).

Launch the exercise and follow the instructions.

Launch Exercise

8. Module assessment

<https://learn.microsoft.com/en-us/training/modules/explore-iaas-paas-platform-tools-for-high-availability-disaster-recovery/8-knowledge-check>

Module assessment

Completed

9. Summary

<https://learn.microsoft.com/en-us/training/modules/explore-iaas-paas-platform-tools-for-high-availability-disaster-recovery/9-summary>

Summary

Completed

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

Depending on the PaaS option chosen for your deployment, there will be different options that you can configure to meet your availability needs, even if you don't have as many options as with an IaaS solution.

Back up and restore databases

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/backup-restore-databases/1-introduction>

Introduction

Completed

Planning and implementing a robust policy for data recovery is essential to mitigate the risks associated with user errors or technology failures. This involves exploring various options for backing up and restoring databases, including on-site and off-site solutions, cloud-based services, and automated backup systems. By understanding the different methods for data back up and recovery, organizations can safeguard their critical information and maintain operational continuity in the face of unforeseen challenges.

Even if you do nothing else to increase your availability with features like AGs or active geo-replication, you must have a solid backup and recovery strategy.

2. Back up and restore SQL Server running on Azure virtual machines

<https://learn.microsoft.com/en-us/training/modules/backup-restore-databases/2-backup-restore-sql-server-running-azure-virtual-machines>

Back up and restore SQL Server running on Azure virtual machines

Completed

SQL Server has two types of databases: system and user. System databases are the ones used by SQL Server such as master and msdb. User databases are the ones created by users that store the data for

applications. Both are important to account for when devising a backup and recovery plan. The nature of most system databases is that they're updated less frequently, although there are exceptions. As a general rule of thumb, system databases aren't restored from one SQL Server instance to another. Your main concern should be backing up the user databases.

The most common types of backups generated for SQL Server installations are full, differential, and transaction log. Depending on the deployment method, not all of these may be available as an option.

A full database backup is a backup of a single database. When the backup is made, all the pages from the database are copied to the backup device. The backup contains enough information so that you can restore the database to the point at which the backup was made. If you want to restore to a specific point-in-time to achieve your Recovery Point Objective (RPO), that can happen with the use of differential and/or transaction log backups. A full database backup backs up all the changes made to the database by the time the backup finishes.

A differential backup contains all the database pages that have changed since the last time a full backup was made.

A transaction log backup isn't only used to be able to achieve RPO and get to a more granular point in time but clears the transaction log and keeps its size manageable. Transaction log backups can be generated as frequently as every 30 seconds, although that is impractical.

Important

Understand how the transaction log works because it impacts not only how the transaction log is backed up, but also how you can do point-in-time recovery using transaction log files.

There are other backup options such as copy-only, file, filegroup, partial, and more.

A differential or a log backup can be restored after a full database is restored, as long as the database RESTORE command uses either the `WITH NORECOVERY` or the `WITH STANDBY` option. If neither option is used, the database RESTORE will do a recovery of the database, after which no extra backups can be applied.

Every SQL Server database uses one of three recovery models: FULL, BULK_LOGGED, or SIMPLE. The recovery model is set as a database option, and governs the type of backups and restores that could be used with the database. Most databases are set to FULL or SIMPLE. FULL allows all types of backups to be generated while SIMPLE doesn't allow transaction log backups. This means that if you have a smaller RPO, SIMPLE may not meet your needs as you can't restore to a granular point in time.

3. Back up a SQL Server virtual machine

Back up a SQL Server virtual machine

Completed

Azure Backup offers the capability to back up virtual machines (VMs) that host SQL Server. These backups contain not only the SQL Server databases but also all other contents within the VM, allowing for a complete restoration if needed. Although this approach may not be suitable for every scenario, it provides a robust safeguard against issues such as ransomware attacks.

VM-level backups are SQL Server-aware, also known as application-aware, ensuring the creation of application-consistent backups. This means that restoring a VM-level backup won't disrupt SQL Server functionality. When using this option, the SQL Server log indicates that the I/O has been momentarily frozen and then resumed upon completion.

Combining SQL Server backups with snapshots can potentially cause issues. If snapshot delays cause backup failures, set following value on the

`[HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\BCDRAGENT]` key:

`"USEVSSCOPYBACKUP"="TRUE"`

Use local disks or a network share for backup files

As with on premises SQL Server instances, databases can be backed up to disks attached to the VM or to network shares (including the file share in Azure called Azure Files) that SQL Server has access to. If you're backing up to disks local to the VM, ensure that they aren't written to the ephemeral storage that is erased upon shutdown or restart. You might also want to make sure that the backups are copied to a second location so as not to create a single point of failure.

Backup databases to and restore from URL

Another option is to configure backup to URL for the SQL Server instance installed in the VM. Unlike backups made on premises, backup and restore from URL for an IaaS VM is effectively a local option.

Back up to URL requires an Azure storage account and uses the Azure blob storage service. Inside the storage account, there are containers, and the blobs are stored there. Unlike a path on your local disk, the path to a backup file looks something like

`https://ACCOUNTNAME.blob.core.windows.net/ContainerName/MyDatabase.bak`. You can include more folder names under your container for easier identification of backups (for example, FULL, DIFF, LOG).

To back up to or restore from a URL, authentication must be established between the SQL Server instance and Azure. Remember that inside of an Azure VM, SQL Server doesn't know it's running on Azure. A SQL Server Credential can be composed of the Azure storage account name and access key authentication or a Shared Access Signature. If the former is used, the backup is stored as a page blob and if the latter, it's stored as a block blob. Starting with SQL Server 2016, only block blob is available so you should use a Shared Access Signature. From a cost perspective, block blobs are also cheaper, and Shared Access Signature tokens offer better security control.

Restoring from a URL is as simple as restoring from disk or a network share. In the SQL Server Management Studio UI, select URL from the backup media type in the Wizard. If using Transact-SQL, instead of using FROM DISK, you would use FROM URL with the appropriate location and backup file name. Here are some sample statements:

The following statement would back up a transaction log.

```
BACKUP LOG contoso  
TO URL = 'https://myacc.blob.core.windows.net/mycontainer/contoso202003271200.trn'
```

The following statement would restore a full database backup without recovering it, so that a differential or transaction log backups could be applied.

```
RESTORE DATABASE contoso  
FROM URL = 'https://myacc.blob.core.windows.net/mycontainer/contoso20200327.bak'  
WITH NORECOVERY
```

Automated backups using the SQL Server resource provider

Any IaaS VM that has SQL Server installed can use the SQL Server resource provider. One of its options is the ability to configure automated backups so Azure takes care of backing up SQL Server databases. It requires the use of a storage account.

One benefit of implementing backups this way is that you can manage retention times for the backups. Another benefit is that you can ensure RPO due to the ability to take database and transaction log backups all in one easy-to-configure place. The following image shows an example of what configuring an automated backup looks like in the Azure portal.

Automated patching

Set a patching window during which all Windows and SQL patches will be applied.

Automated patching ⓘ **Enabled**
Sunday at 2:00
[Change configuration](#)

Automated backup

Automated backup ⓘ

Retention period (days) 30

Storage account [Select Storage Container](#)

Encryption

Backup system databases ⓘ

Configure backup schedule ⓘ

Backup frequency ⓘ

Backup start time (local VM time) 02:00 ▼

Backup time window (hours) 2

Log backup frequency (minutes) 60

The automated backup option is currently only available for Windows Server-based SQL Server installations.

Important

You choose one method of backing up databases with IaaS-based SQL Server deployments. For example, if you use automated backups, especially with transaction log backups, don't also configure those at the instance level inside the VM. You could cause problems with the log chain with restoring a database if things are uncoordinated, because each log backup clears the log and you must have an entire unbroken chain of log backups in order to do a log restore. For example, if transaction log backups happen inside the guest and at the Azure level, you may have to piece together the backups to do a restore.

While the backups can be automated, restores can't be. You would need to configure and use the restore from URL functionality within SQL Server.

4. Back up and restore a database for SQL Database and SQL Managed Instance

<https://learn.microsoft.com/en-us/training/modules/backup-restore-databases/4-use-azure-sql-database>

Back up and restore a database for SQL Database and SQL Managed Instance

Completed

Back up and restore on SQL Server PaaS offering work differently than on IaaS. Backups are generated automatically for Azure SQL Database, and Azure SQL Managed Instance. A full backup is created once a week, a differential every 12 hours, and transaction log backups every 5 to 10 minutes. All backups are located in read-access, geo-redundant (RA-GRS) blobs replicated to a datacenter that is paired based on Azure rules. That means backups are safe from an outage in a single data center.

Database backup and restore for SQL Database

Database backups are an essential part of any business continuity and disaster recovery strategy because they help protect your data from corruption or deletion.

SQL Database can assist you to be compliant with mandatory backups for regulatory purposes with retention policies. Backup policies can be configured per database as shown in the following image:

Configure policies



SQL server

Point-in-time-restore

Specify how long you want to keep your point-in-time backups. [Learn more](#)

How many days would you like PITR backups to be kept? ⓘ

Differential backup frequency

Specify how often you want differential backups to be taken. [Learn more](#)

Take a differential backup every:

Long-term retention

Specify how long you want to keep your long-term retention backups. You may choose to keep yearly backups for up to 10 years. [Learn more](#)

Weekly LTR Backups

Keep weekly backups for:

Monthly LTR Backups

Keep the first backup of each month for:

Yearly LTR Backups

Keep an annual backup for:

Which weekly backup of the year would you like to keep?

Apply

Cancel

If the server containing the database is deleted, all backups are deleted at the same time, and there's no way to recover them. If the server isn't deleted but the database is, you can restore the database

normally.

Both SQL Database, and SQL Managed Instance have a feature called Accelerated Database Recovery (ADR). This feature is enabled by default, and its purpose is to decrease the time it takes to deal with long running transactions so they don't affect the recovery time. Although Accelerated Database Recovery was developed for Azure and was originally an Azure-based feature, ADR was implemented in SQL Server 2019 as well.

Note

You can't restore SQL Database Azure SQL Managed Instance backups on SQL Database.

Automated database backups are also available on [Azure SQL Managed Instance](#). SQL Server database engine backups are automatically managed by Microsoft, and stored on Microsoft-managed Azure storage accounts.

Point in time restore

To restore a database to a specific point in time on SQL Database, you can use either Azure portal, Azure PowerShell, Azure CLI, or REST API.

Home > SQL databases > AdventureWorks (contoso/AdventureWorks) > contoso > AdventureWorks (contoso/AdventureWorks)

Create SQL Database - Restore database

Microsoft

Basics Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription ⓘ My Subscription ▼

Resource group ⓘ contoso_rg ▼

Source Details

Select a backup source and details. Additional settings will be defaulted where possible based on the backup selected.

Source Database AdventureWorks

Select source Point-in-time ▼

Earliest restore point 2021-11-23 00:00 UTC

Restore point (UTC) * 11/30/2021 1:51:00 PM

i Choose a restore point between the earliest restore point and the current time in UTC.

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name * AdventureWorks_2021-11-30T13-51Z

Server ⓘ contoso (West US) ▼

Want to use SQL elastic pool? * ⓘ ☐ Yes ☒ No

Compute + storage * ⓘ

General Purpose

Serverless, Gen5, 2 vCores, 32 GB storage

[Configure database](#)

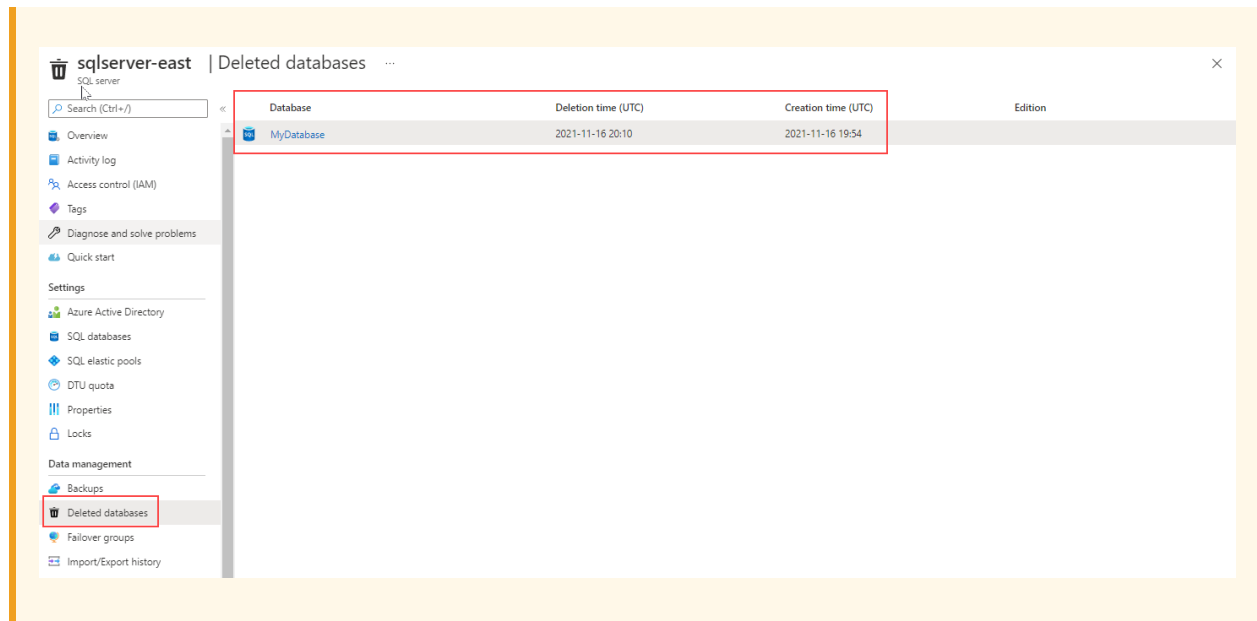
The following image shows the SQL Database restore page on Azure portal, where you can restore a database to a specific point in time.

Restore in place isn't supported on SQL Database, and SQL Managed Instance. You need to make sure the database doesn't exist before attempting the restore operation. By default, point in time retention policy is set to seven days, and you can change it to up to 35-days.

Restore a deleted database

Both SQL Database, and SQL Managed Instance have a feature to restore a deleted database to the last point in time available before the `DROP DATABASE` took place. To recover a deleted database to

the deletion time by using the Azure portal, open the server's overview page and select **Deleted databases**. Select a deleted database that you want to restore, and then enter the name for the new database that will be created with data restored from the backup.



The image shows how to restore a deleted database on SQL Database. The *deleted databases* page shows a list of deleted databases available to restore, the database deletion time in UTC, and the database creation time in UTC. Once you select the database, the *Create SQL Database - Restore database* page opens. On that page you find the earliest restore point in time available for the selected database.

Database backup and restore for SQL Managed Instance

Azure manages backups for databases in SQL Managed Instance automatically, and they operate similar to SQL Database.

You can also manually back up, and restore databases with SQL Managed Instance using the same backup to URL/restore from URL functionality found in SQL Server covered earlier. That requires the use of credentials to access the Azure Blob Storage container. SQL Database doesn't support this feature.

You can only generate a `COPY_ONLY` backup since SQL Managed Instance is maintaining the log chain. A sample backup statement would look like:

```
BACKUP DATABASE contoso
TO URL = 'https://myacc.blob.core.windows.net/mycontainer/contoso.bak'
WITH COPY_ONLY
```

Note

You can't restore SQL Database SQL Managed Instance backups on SQL Database.

5. Exercise: Backup to URL

<https://learn.microsoft.com/en-us/training/modules/backup-restore-databases/5-exercise-backup-url>

Exercise: Backup to URL

Completed

In this exercise, you'll learn how to perform a database backup to an URL.

Note

To complete this exercise, you'll need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

6. Module assessment

<https://learn.microsoft.com/en-us/training/modules/backup-restore-databases/6-knowledge-check>

Module assessment

Completed

7. Summary

Summary

Completed

Without backups, you don't have high availability or disaster recovery. Features like AGs or active-geo replication don't replace a proper backup and recovery strategy. Backups must be tested via a restore otherwise your backups are just files that sit on a file system that checks a box somewhere that backups were generated.

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

- Database backup and restore options for IaaS
- Virtual machine backup and restore options for IaaS
- Backup and restore options for PaaS