

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.

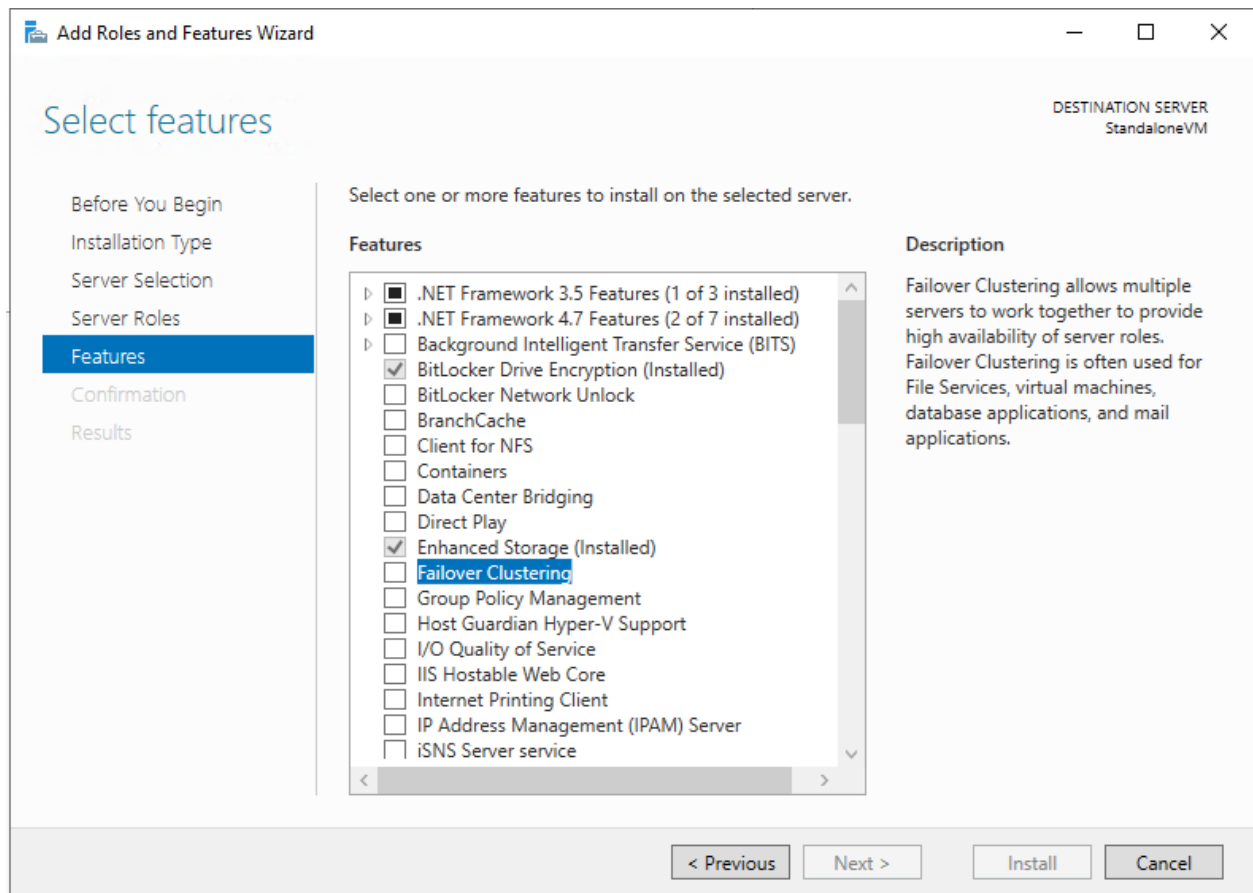
A typical Azure-based WSFC will only require a single virtual network card (vNIC). Unlike on-premises setups, the IP address for the vNIC must be configured in Azure, not within the VM. Inside the VM, it appears as a dynamic (DHCP) address, which is expected. However, cluster validation generates a warning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

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.

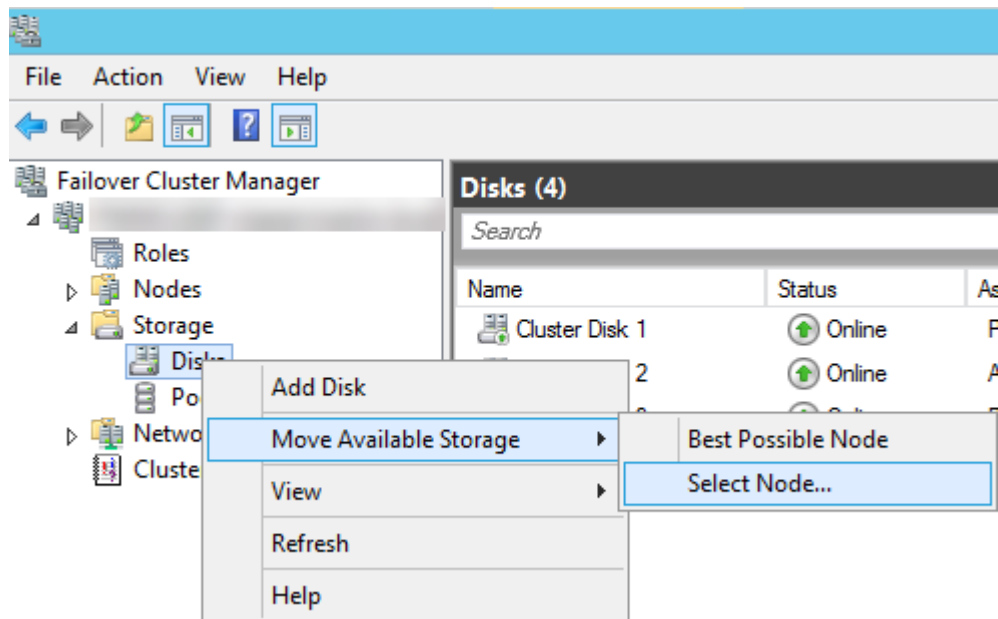
The VNN still exists in an FCI deployment, but the clients connect to the DNN DNS name instead of the VNN name.

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

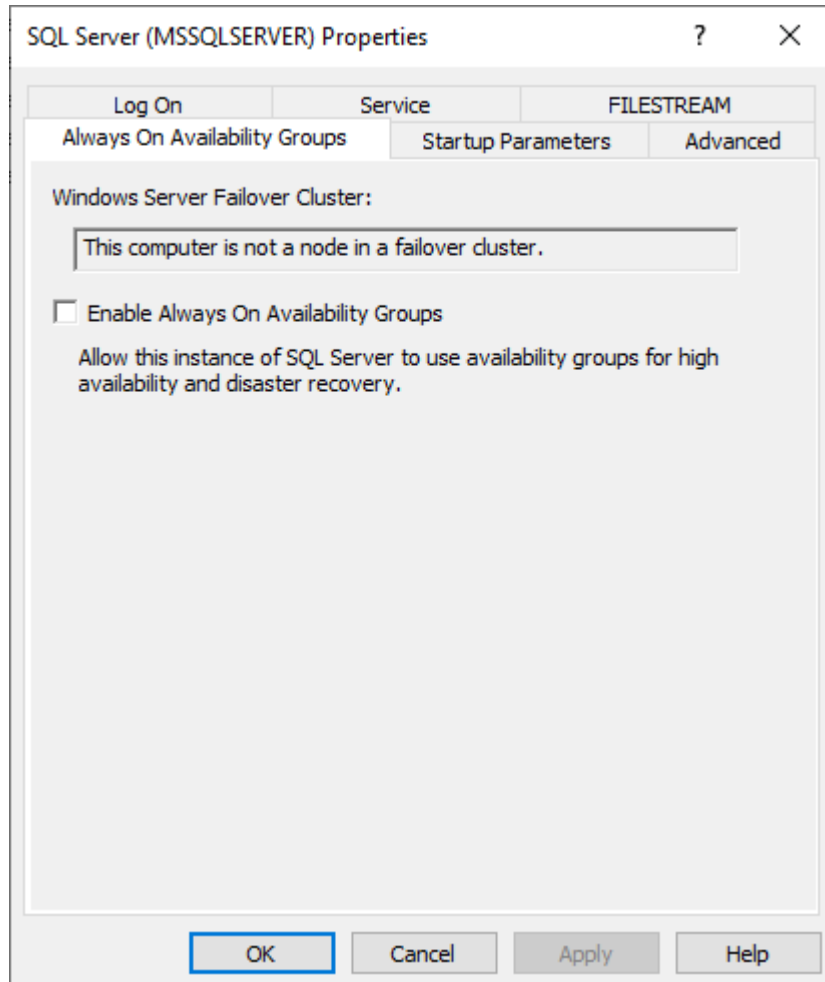
Completed

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

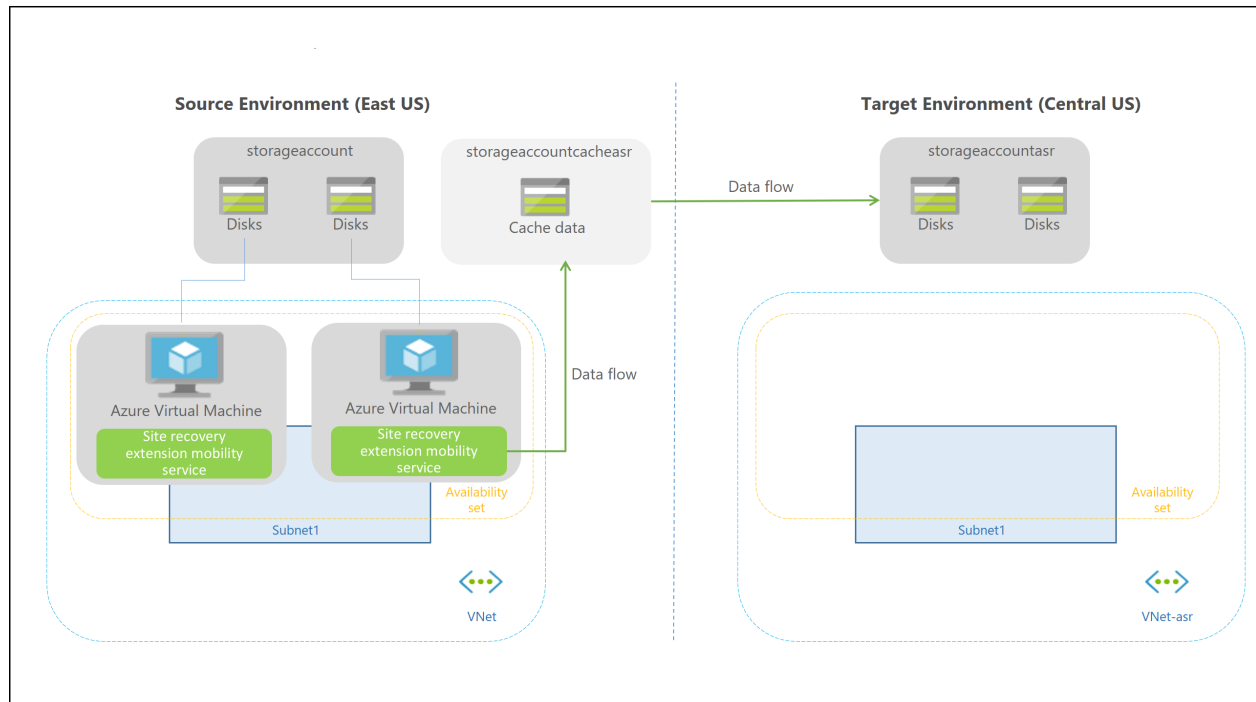
Planning for and configuring a distributed AG is the same on premises as it is in Azure, with any Azure-specific considerations for the individual AGs. The main difference between an on-premises configuration and an Azure configuration for a distributed AG is that as part of the load balancer configuration 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 list of policies. The 'Tier1-policy' is selected, showing it is configured for 'VMware / Physical machines' as the source type and 'Azure' as the target type, with 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

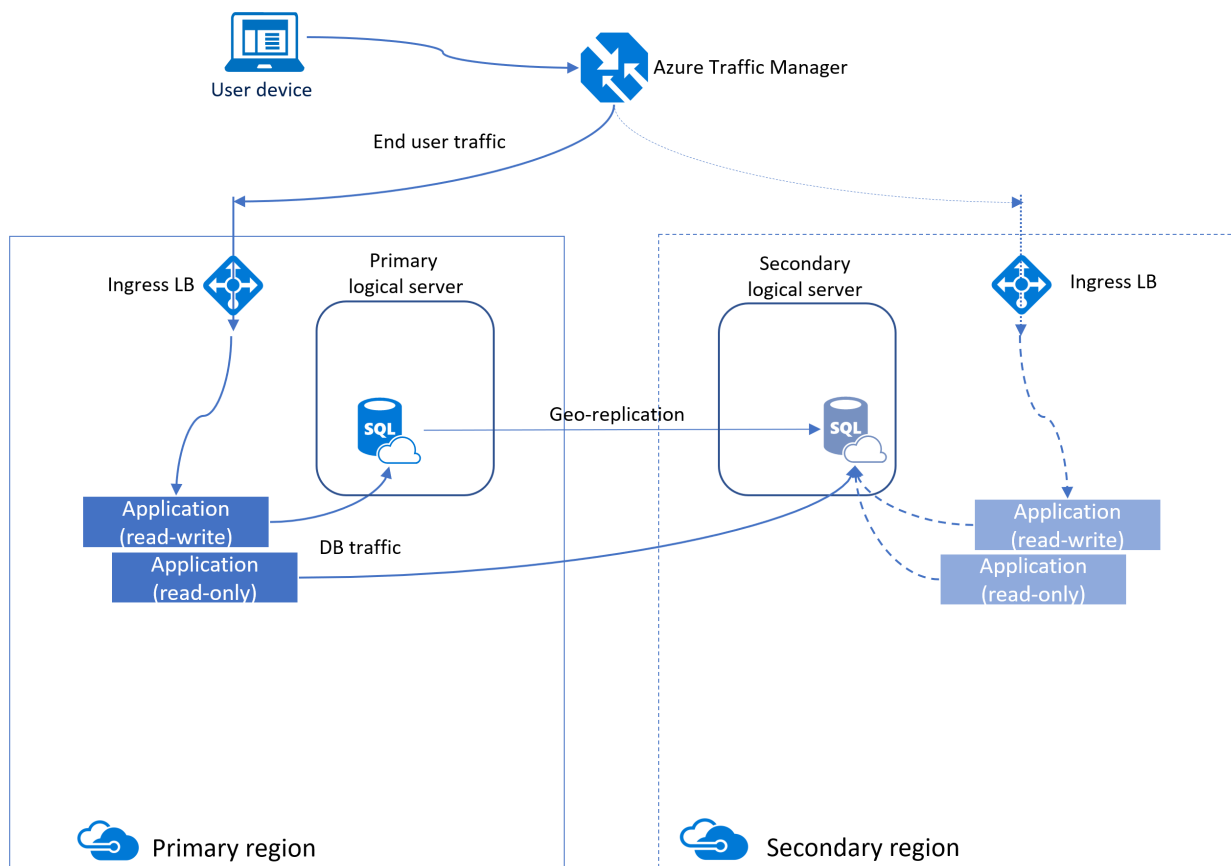
Completed

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.

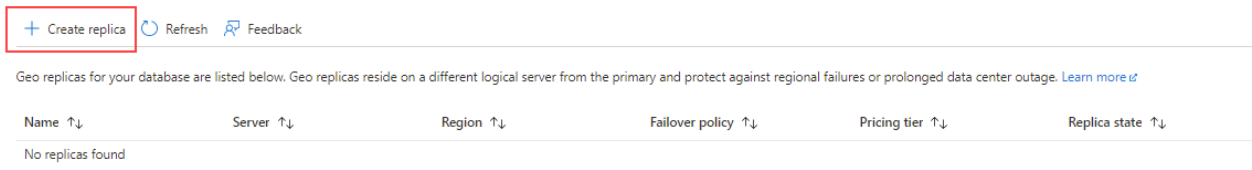
Azure SQL Managed Instance doesn't support active geo-replication. You must use auto-failover groups instead, which will learn on the next unit.



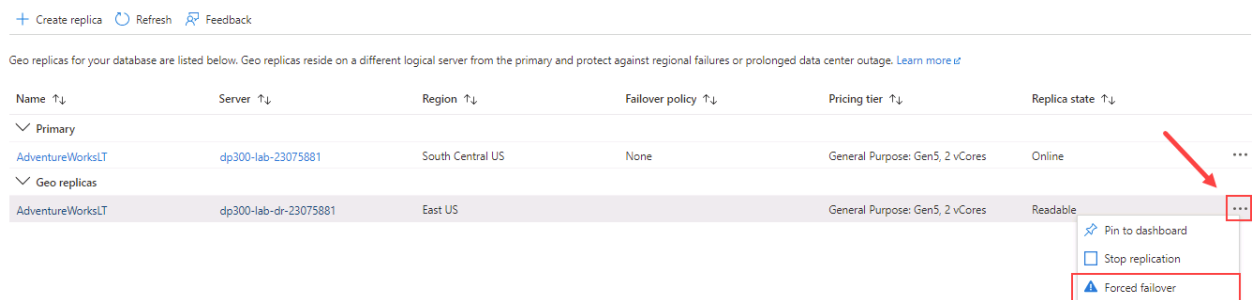
All the databases involved in a geo-replication relationship are required to have the same service tier.

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

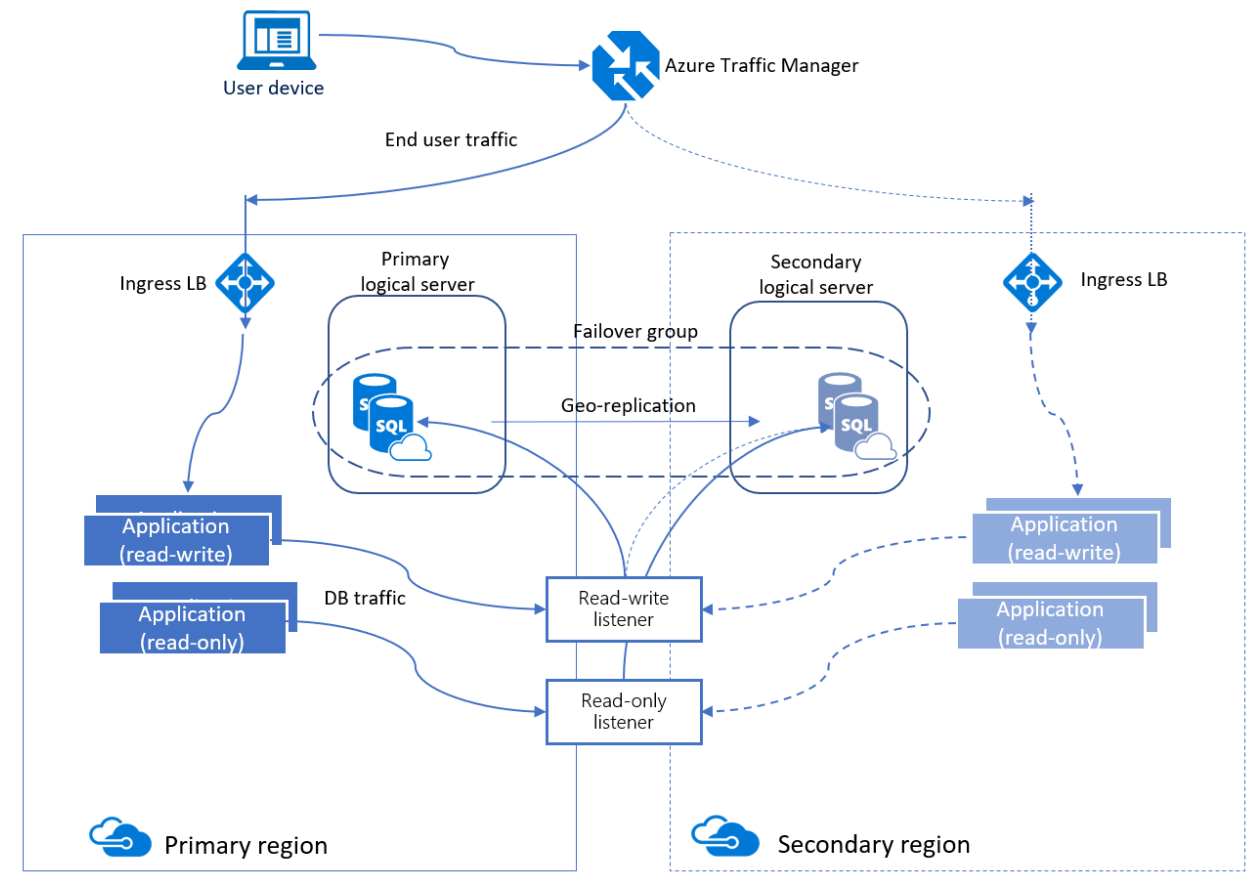
<https://learn.microsoft.com/en-us/training/modules/explore-iaas-paas-platform-tools-for-high-availability-disaster-recovery/6-explore-auto-failover-groups>

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.



Auto-failover groups have two different policies that can be configured.

- **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.

Failover can be performed manually even if automatic failover is allowed. Depending on the type of failover, there could be data loss. Unplanned failover could result in data loss if forced and the secondary isn't fully synchronized with the primary. Configuring `GracePeriodWithDataLossHours` controls how long Azure waits before failing over. The default is one hour. If you have a tight RPO and can't afford much data loss, set the value higher. Although Azure waits longer before failing over, this approach may result in less data loss as the secondary has more time to fully synchronize with the primary.

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.

Geo-replication and auto-failover groups

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.