


Run an N-tier application in multiple Azure regions for high availability

06/18/2019 • 9 minutes to read • Contributors 

In this article

[Architecture](#)

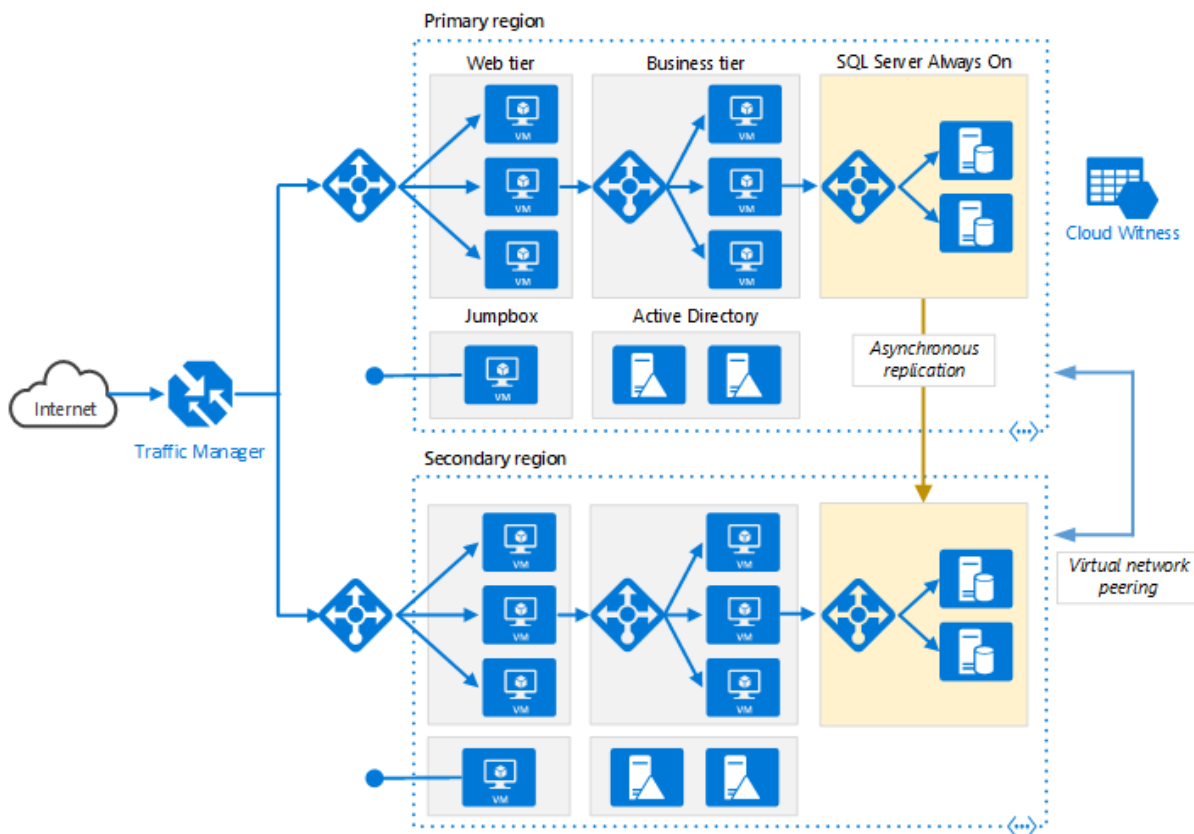
[Recommendations](#)

[Availability considerations](#)

[Manageability considerations](#)

[Related resources](#)

This reference architecture shows a set of proven practices for running an N-tier application in multiple Azure regions, in order to achieve availability and a robust disaster recovery infrastructure.



Download a [Visio file](#) of this architecture.

Architecture

This architecture builds on the one shown in [N-tier application with SQL Server](#).

- **Primary and secondary regions.** Use two regions to achieve higher availability. One is the primary region. The other region is for failover.
- **Azure Traffic Manager.** [Traffic Manager](#) routes incoming requests to one of the regions. During normal operations, it routes requests to the primary region. If that region becomes unavailable, Traffic Manager fails over to the secondary region. For more information, see the section [Traffic Manager configuration](#).

- **Resource groups.** Create separate [resource groups](#) for the primary region, the secondary region, and for Traffic Manager. This gives you the flexibility to manage each region as a single collection of resources. For example, you could redeploy one region, without taking down the other one. [Link the resource groups](#), so that you can run a query to list all the resources for the application.
- **Virtual networks.** Create a separate virtual network for each region. Make sure the address spaces do not overlap.
- **SQL Server Always On Availability Group.** If you are using SQL Server, we recommend [SQL Always On Availability Groups](#) for high availability. Create a single availability group that includes the SQL Server instances in both regions.

ⓘ Note

Also consider [Azure SQL Database](#), which provides a relational database as a cloud service. With SQL Database, you don't need to configure an availability group or manage failover.

- **Virtual network peering.** Peer the two virtual networks to allow data replication from the primary region to the secondary region. For more information, see [Virtual network peering](#).

Recommendations

A multi-region architecture can provide higher availability than deploying to a single region. If a regional outage affects the primary region, you can use [Traffic Manager](#) to fail over to the secondary region. This architecture can also help if an individual subsystem of the application fails.

There are several general approaches to achieving high availability across regions:

- **Active/passive with hot standby.** Traffic goes to one region, while the other waits on hot standby. Hot standby means the VMs in the secondary region are allocated and running at all times.
- **Active/passive with cold standby.** Traffic goes to one region, while the other waits on cold standby. Cold standby means the VMs in the secondary region are not allocated until needed for failover. This approach costs less to run, but will generally take longer to come online during a failure.
- **Active/active.** Both regions are active, and requests are load balanced between them. If one region becomes unavailable, it is taken out of rotation.

This reference architecture focuses on active/passive with hot standby, using Traffic Manager for failover. Note that you could deploy a small number of VMs for hot standby and then scale out as needed.

Regional pairing

Each Azure region is paired with another region within the same geography. In general, choose regions from the same regional pair (for example, East US 2 and US Central). Benefits of doing so include:

- If there is a broad outage, recovery of at least one region out of every pair is prioritized.
- Planned Azure system updates are rolled out to paired regions sequentially, to minimize possible downtime.
- Pairs reside within the same geography, to meet data residency requirements.

However, make sure that both regions support all of the Azure services needed for your application (see [Services by region](#)). For more information about regional pairs, see [Business continuity and disaster recovery \(BCDR\): Azure Paired Regions](#).

Traffic Manager configuration

Consider the following points when configuring Traffic Manager:

- **Routing.** Traffic Manager supports several [routing algorithms](#). For the scenario described in this article, use *priority* routing (formerly called *failover* routing). With this setting, Traffic Manager sends all requests to the primary region, unless the primary region becomes unreachable. At that point, it automatically fails over to the secondary region. See [Configure Failover routing method](#).
- **Health probe.** Traffic Manager uses an HTTP (or HTTPS) [probe](#) to monitor the availability of each region. The probe checks for an HTTP 200 response for a specified URL path. As a best practice, create an endpoint that reports the overall health of the application, and use this endpoint for the health probe. Otherwise, the probe might report a healthy endpoint when critical parts of the application are actually failing. For more information, see [Health Endpoint Monitoring pattern](#).

When Traffic Manager fails over there is a period of time when clients cannot reach the application. The duration is affected by the following factors:


- The health probe must detect that the primary region has become unreachable.
- DNS servers must update the cached DNS records for the IP address, which depends on the DNS time-to-live (TTL). The default TTL is 300 seconds (5 minutes), but you can configure this value when you create the Traffic Manager profile.

For details, see [About Traffic Manager Monitoring](#).


If Traffic Manager fails over, we recommend performing a manual failback rather than implementing an automatic failback. Otherwise, you can create a situation where the application flips back and forth between regions. Verify that all application subsystems are healthy before failing back.

Note that Traffic Manager automatically fails back by default. To prevent this, manually lower the priority of the primary region after a failover event. For example, suppose the primary region is priority 1 and the secondary is priority 2. After a failover, set the primary region to priority 3, to prevent automatic failback. When you are ready to switch back, update the priority to 1.

The following [Azure CLI](#) command updates the priority:

Azure CLI	 Copy
<pre>az network traffic-manager endpoint update --resource-group <resource-group> --profile-name <profile> --name <endpoint-name> --type azureEndpoints --priority 3</pre>	

Another approach is to temporarily disable the endpoint until you are ready to fail back:

Azure CLI	 Copy
<pre>az network traffic-manager endpoint update --resource-group <resource-group> --profile-name <profile> --name <endpoint-name> --type azureEndpoints --endpoint-status Disabled</pre>	

Depending on the cause of a failover, you might need to redeploy the resources within a region. Before failing back, perform an operational readiness test. The test should verify things like:

- VMs are configured correctly. (All required software is installed, IIS is running, and so on.)
- Application subsystems are healthy.
- Functional testing. (For example, the database tier is reachable from the web tier.)

Configure SQL Server Always On Availability Groups

Prior to Windows Server 2016, SQL Server Always On Availability Groups require a domain controller, and all nodes in the availability group must be in the same Active Directory (AD) domain.

To configure the availability group:

- At a minimum, place two domain controllers in each region.
- Give each domain controller a static IP address.
- Peer the two virtual networks to enable communication between them.
- For each virtual network, add the IP addresses of the domain controllers (from both regions) to the DNS server list. You can use the following CLI command. For more information, see [Change DNS servers](#).

Azure CLI

 Copy

```
az network vnet update --resource-group <resource-group> --name <vnet-name> --dns-servers  
"10.0.0.4,10.0.0.6,172.16.0.4,172.16.0.6"
```

- Create a [Windows Server Failover Clustering](#) (WSFC) cluster that includes the SQL Server instances in both regions.
- Create a SQL Server Always On Availability Group that includes the SQL Server instances in both the primary and secondary regions. See [Extending Always On Availability Group to Remote Azure Datacenter \(PowerShell\)](#) for the steps.
 - Put the primary replica in the primary region.
 - Put one or more secondary replicas in the primary region. Configure these to use synchronous commit with automatic failover.
 - Put one or more secondary replicas in the secondary region. Configure these to use *asynchronous* commit, for performance reasons. (Otherwise, all T-SQL transactions have to wait on a round trip over the network to the secondary region.)

Note

Asynchronous commit replicas do not support automatic failover.

Availability considerations

With a complex N-tier app, you may not need to replicate the entire application in the secondary region. Instead, you might just replicate a critical subsystem that is needed to support business continuity.

Traffic Manager is a possible failure point in the system. If the Traffic Manager service fails, clients cannot access your application during the downtime. Review the [Traffic Manager SLA](#), and determine whether using Traffic Manager alone meets your business requirements for high availability. If not, consider adding another traffic management solution as a fallback. If the Azure Traffic Manager service fails, change your CNAME records in DNS to point to the other traffic management service. (This step must be performed manually, and your application will be unavailable until the DNS changes are propagated.)

For the SQL Server cluster, there are two failover scenarios to consider:

- All of the SQL Server database replicas in the primary region fail. For example, this could happen during a regional outage. In that case, you must manually fail over the availability group, even though Traffic Manager automatically fails over on the front end. Follow the steps in [Perform a Forced Manual Failover of a SQL Server Availability Group](#), which describes how to perform a forced failover by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2016.

⚠ Warning

With forced failover, there is a risk of data loss. Once the primary region is back online, take a snapshot of the database and use [tablediff](#) to find the differences.

- Traffic Manager fails over to the secondary region, but the primary SQL Server database replica is still available. For example, the front-end tier might fail, without affecting the SQL Server VMs. In that case, Internet traffic is routed to the secondary region, and that region can still connect to the primary replica. However, there will be increased latency, because the SQL Server connections are going across regions. In this situation, you should perform a manual failover as follows:
 1. Temporarily switch a SQL Server database replica in the secondary region to *synchronous* commit. This ensures there won't be data loss during the failover.
 2. Fail over to that replica.
 3. When you fail back to the primary region, restore the asynchronous commit setting.

Manageability considerations

When you update your deployment, update one region at a time to reduce the chance of a global failure from an incorrect configuration or an error in the application.

Test the resiliency of the system to failures. Here are some common failure scenarios to test:

- Shut down VM instances.
- Pressure resources such as CPU and memory.
- Disconnect/delay network.
- Crash processes.
- Expire certificates.
- Simulate hardware faults.
- Shut down the DNS service on the domain controllers.

Measure the recovery times and verify they meet your business requirements. Test combinations of failure modes, as well.

Related resources

You may wish to review the following [Azure example scenarios](#) that demonstrate specific solutions using some of the same technologies:

- [Multitier web application built for high availability and disaster recovery on Azure](#)
- [Building secure web applications with Windows virtual machines on Azure](#)