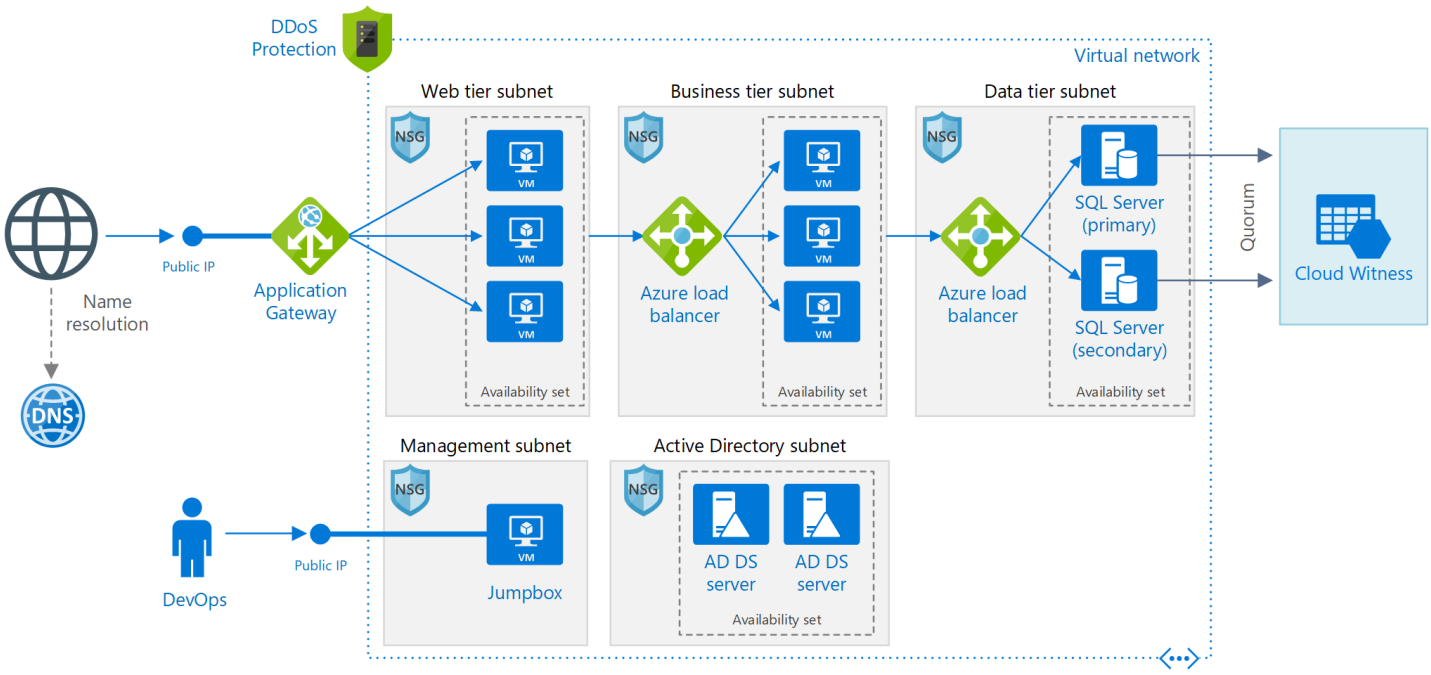
Windows N-tier application on Azure with SQL Server

11/12/2018 • 12 minutes to read • Contributors

In this article

This reference architecture shows how to deploy VMs and a virtual network configured for an N-tier application, using SQL Server on Windows for the data tier. **Deploy this solution**.



Download a Visio file of this architecture.

Architecture

The architecture has the following components:

- **Resource group**. Resource groups are used to group resources so they can be managed by lifetime, owner, or other criteria.
- **Virtual network (VNet) and subnets**. Every Azure VM is deployed into a VNet that can be segmented into subnets. Create a separate subnet for each tier.
- **Application gateway**. Azure Application Gateway is a layer 7 load balancer. In this architecture, it routes HTTP requests to the web front end. Application Gateway also provides a web application firewall (WAF) that protects the application from common exploits and vulnerabilities.

- **NSGs**. Use network security groups (NSGs) to restrict network traffic within the VNet. For example, in the three-tier architecture shown here, the database tier does not accept traffic from the web front end, only from the business tier and the management subnet.
- **DDoS Protection**. Although the Azure platform provides basic protection against distributed denial of service (DDoS) attacks, we recommend using DDoS Protection Standard, which has enhanced DDoS mitigation features. See Security considerations.
- **Virtual machines**. For recommendations on configuring VMs, see Run a Windows VM on Azure and Run a Linux VM on Azure.
- **Availability sets**. Create an availability set for each tier, and provision at least two VMs in each tier, which makes the VMs eligible for a higher service level agreement (SLA).
- **Load balancers**. Use Azure Load Balancer to distribute network traffic from the web tier to the business tier, and from the business tier to SQL Server.
- **Public IP address**. A public IP address is needed for the application to receive Internet traffic.
- **Jumpbox**. Also called a bastion host. A secure VM on the network that administrators use to connect to the other VMs. The jumpbox has an NSG that allows remote traffic only from public IP addresses on a safe list. The NSG should permit remote desktop (RDP) traffic.
- **SQL Server Always On Availability Group**. Provides high availability at the data tier, by enabling replication and failover. It uses Windows Server Failover Cluster (WSFC) technology for failover.
- **Active Directory Domain Services (AD DS) Servers**. The computer objects for the failover cluster and its associated clustered roles are created in Active Directory Domain Services (AD DS).
- **Cloud Witness**. A failover cluster requires more than half of its nodes to be running, which is known as having quorum. If the cluster has just two nodes, a network partition could cause each node to think it's the master node. In that case, you need a *witness* to break ties and establish quorum. A witness is a resource such as a shared disk that can act as a tie breaker to establish quorum. Cloud Witness is a type of witness that uses Azure Blob Storage. To learn more about the concept of quorum, see Understanding cluster and pool quorum. For more information about Cloud Witness, see Deploy a Cloud Witness for a Failover Cluster.
- **Azure DNS**. Azure DNS is a hosting service for DNS domains. It provides name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services.

Recommendations

Your requirements might differ from the architecture described here. Use these recommendations as a starting point.

VNet / Subnets

When you create the VNet, determine how many IP addresses your resources in each subnet require. Specify a subnet mask and a VNet address range large enough for the required IP addresses, using CIDR notation. Use an address space that falls within the standard private IP address blocks, which are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.

Choose an address range that does not overlap with your on-premises network, in case you need to set up a gateway between the VNet and your on-premises network later. Once you create the VNet, you can't change the address range.

Design subnets with functionality and security requirements in mind. All VMs within the same tier or role should go into the same subnet, which can be a security boundary. For more information about designing VNets and subnets, see Plan and design Azure Virtual Networks.

Load balancers

Don't expose the VMs directly to the Internet, but instead give each VM a private IP address. Clients connect using the public IP address associated with the Application Gateway.

Define load balancer rules to direct network traffic to the VMs. For example, to enable HTTP traffic, map port 80 from the front-end configuration to port 80 on the back-end address pool. When a client sends an HTTP request to port 80, the load balancer selects a back-end IP address by using a hashing algorithm that includes the source IP address. Client requests are distributed across all the VMs in the back-end address pool.

Network security groups

Use NSG rules to restrict traffic between tiers. In the three-tier architecture shown above, the web tier does not communicate directly with the database tier. To enforce this, the database tier should block incoming traffic from the web tier subnet.

1. Deny all inbound traffic from the VNet. (Use the `VIRTUAL_NETWORK` tag in the rule.)
2. Allow inbound traffic from the business tier subnet.
3. Allow inbound traffic from the database tier subnet itself. This rule allows communication between the database VMs, which is needed for database replication and failover.
4. Allow RDP traffic (port 3389) from the jumpbox subnet. This rule lets administrators connect to the database tier from the jumpbox.

Create rules 2 – 4 with higher priority than the first rule, so they override it.

SQL Server Always On Availability Groups

We recommend Always On Availability Groups for SQL Server high availability. Prior to Windows Server 2016, Always On Availability Groups require a domain controller, and all nodes in the availability group must be in the same AD domain.

Other tiers connect to the database through an availability group listener. The listener enables a SQL client to connect without knowing the name of the physical instance of SQL Server. VMs that access the database must be joined to the domain. The client (in this case, another tier) uses DNS to resolve the listener's virtual network name into IP addresses.

Configure the SQL Server Always On Availability Group as follows:

1. Create a Windows Server Failover Clustering (WSFC) cluster, a SQL Server Always On Availability Group, and a primary replica. For more information, see Getting Started with Always On Availability Groups.

2. Create an internal load balancer with a static private IP address.

3. Create an availability group listener, and map the listener's DNS name to the IP address of an internal load balancer.

4. Create a load balancer rule for the SQL Server listening port (TCP port 1433 by default). The load balancer rule must enable *floating IP*, also called Direct Server Return. This causes the VM to reply directly to the client, which enables a direct connection to the primary replica.

   > **Note**
   >
   > When floating IP is enabled, the front-end port number must be the same as the back-end port number in the load balancer rule.

When a SQL client tries to connect, the load balancer routes the connection request to the primary replica. If there is a failover to another replica, the load balancer automatically routes new requests to a new primary replica. For more

information, see Configure an ILB listener for SQL Server Always On Availability Groups.

During a failover, existing client connections are closed. After the failover completes, new connections will be routed to the new primary replica.

If your application makes significantly more reads than writes, you can offload some of the read-only queries to a secondary replica. See Using a Listener to Connect to a Read-Only Secondary Replica (Read-Only Routing).

Test your deployment by forcing a manual failover of the availability group.

Jumpbox

Don't allow RDP access from the public Internet to the VMs that run the application workload. Instead, all RDP access to these VMs must come through the jumpbox. An administrator logs into the jumpbox, and then logs into the other VM from the jumpbox. The jumpbox allows RDP traffic from the Internet, but only from known, safe IP addresses.

The jumpbox has minimal performance requirements, so select a small VM size. Create a public IP address for the jumpbox. Place the jumpbox in the same VNet as the other VMs, but in a separate management subnet.

To secure the jumpbox, add an NSG rule that allows RDP connections only from a safe set of public IP addresses. Configure the NSGs for the other subnets to allow RDP traffic from the management subnet.

Scalability considerations

For the web and business tiers, consider using virtual machine scale sets, instead of deploying separate VMs into an availability set. A scale set makes it easy to deploy and manage a set of identical VMs, and autoscale the VMs based on performance metrics. As the load on the VMs increases, additional VMs are automatically added to the load balancer. Consider scale sets if you need to quickly scale out VMs, or need to autoscale.

There are two basic ways to configure VMs deployed in a scale set:

- Use extensions to configure the VM after it's deployed. With this approach, new VM instances may take longer to start up than a VM with no extensions.
- Deploy a managed disk with a custom disk image. This option may be quicker to deploy. However, it requires you to keep the image up-to-date.

For more information, see Design considerations for scale sets.

> **Tip**
>
> When using any autoscale solution, test it with production-level workloads well in advance.

Each Azure subscription has default limits in place, including a maximum number of VMs per region. You can increase the limit by filing a support request. For more information, see Azure subscription and service limits, quotas, and constraints.

Availability considerations

If you don't use virtual machine scale sets, put VMs for the same tier into an availability set. Create at least two VMs in the availability set to support the availability SLA for Azure VMs. For more information, see Manage the availability of virtual machines. Scale sets automatically use *placement groups*, which act as an implicit availability set.

The load balancer uses health probes to monitor the availability of VM instances. If a probe can't reach an instance within a timeout period, the load balancer stops sending traffic to that VM. However, the load balancer will continue to

probe, and if the VM becomes available again, the load balancer resumes sending traffic to that VM.

Here are some recommendations on load balancer health probes:

- Probes can test either HTTP or TCP. If your VMs run an HTTP server, create an HTTP probe. Otherwise create a TCP probe.
- For an HTTP probe, specify the path to an HTTP endpoint. The probe checks for an HTTP 200 response from this path. This path can be the root path ("/"), or a health-monitoring endpoint that implements some custom logic to check the health of the application. The endpoint must allow anonymous HTTP requests.
- The probe is sent from a known IP address, 168.63.129.16. Don't block traffic to or from this IP address in any firewall policies or NSG rules.
- Use health probe logs to view the status of the health probes. Enable logging in the Azure portal for each load balancer. Logs are written to Azure Blob storage. The logs show how many VMs aren't getting network traffic because of failed probe responses.

If you need higher availability than the Azure SLA for VMs provides, consider replication the application across two regions, using Azure Traffic Manager for failover. For more information, see Multi-region N-tier application for high availability.

Security considerations

Virtual networks are a traffic isolation boundary in Azure. VMs in one VNet can't communicate directly with VMs in a different VNet. VMs within the same VNet can communicate, unless you create network security groups (NSGs) to restrict traffic. For more information, see Microsoft cloud services and network security.

DMZ. Consider adding a network virtual appliance (NVA) to create a DMZ between the Internet and the Azure virtual network. NVA is a generic term for a virtual appliance that can perform network-related tasks, such as firewall, packet inspection, auditing, and custom routing. For more information, see Implementing a DMZ between Azure and the Internet.

Encryption. Encrypt sensitive data at rest and use Azure Key Vault to manage the database encryption keys. Key Vault can store encryption keys in hardware security modules (HSMs). For more information, see Configure Azure Key Vault Integration for SQL Server on Azure VMs. It's also recommended to store application secrets, such as database connection strings, in Key Vault.

DDoS protection. The Azure platform provides basic DDoS protection by default. This basic protection is targeted at protecting the Azure infrastructure as a whole. Although basic DDoS protection is automatically enabled, we recommend using DDoS Protection Standard. Standard protection uses adaptive tuning, based on your application's network traffic patterns, to detect threats. This allows it to apply mitigations against DDoS attacks that might go unnoticed by the infrastructure-wide DDoS policies. Standard protection also provides alerting, telemetry, and analytics through Azure Monitor. For more information, see Azure DDoS Protection: Best practices and reference architectures.

Deploy the solution

A deployment for this reference architecture is available on GitHub. The entire deployment can take up to two hours, which includes running the scripts to configure AD DS, the Windows Server failover cluster, and the SQL Server availability group.

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Install Azure CLI 2.0.
3. Install Node and NPM

4. Install the Azure building blocks npm package.

```
npm install -g @mspnp/azure-building-blocks
```

5. From a command prompt, bash prompt, or PowerShell prompt, sign into your Azure account as follows:

```
az login
```

Deployment steps

1. Run the following command to create a resource group.

```
az group create --location <location> --name <resource-group-name>
```

2. Run the following command to create a Storage account for the Cloud Witness.

```
az storage account create --location <location> \
  --name <storage-account-name> \
  --resource-group <resource-group-name> \
  --sku Standard_LRS
```

3. Navigate to the `virtual-machines\n-tier-windows` folder of the reference architectures GitHub repository.

4. Open the `n-tier-windows.json` file.

5. Search for all instances of "witnessStorageBlobEndPoint" and replace the placeholder text with the name of the Storage account from step 2.

```
"witnessStorageBlobEndPoint": "https://[replace-with-
storageaccountname].blob.core.windows.net",
```

6. Run the following command to list the account keys for the storage account.

```
az storage account keys list \
  --account-name <storage-account-name> \
  --resource-group <resource-group-name>
```

The output should look like the following. Copy the value of `key1`.

```
[
{
    "keyName": "key1",
    "permissions": "Full",
    "value": "..."
```

```
},
{
    "keyName": "key2",
    "permissions": "Full",
    "value": "..."
}
]
```

7. In the `n-tier-windows.json` file, search for all instances of "witnessStorageAccountKey" and paste in the account key.

```
"witnessStorageAccountKey": "[replace-with-storagekey]"
```

8. In the `n-tier-windows.json` file, search for all instances of `[replace-with-password]` and `[replace-with-sql-password]` replace them with a strong password. Save the file.

> **Note**
>
> If you change the administrator user name, you must also update the `extensions` blocks in the JSON file.

9. Run the following command to deploy the architecture.

```
azbb -s <your subscription_id> -g <resource_group_name> -l <location> -p n-tier-windows.j-
son --deploy
```

For more information on deploying this sample reference architecture using Azure Building Blocks, visit the GitHub repository.

Next steps

- Microsoft Learn module: Tour the N-tier architecture style