Implement a DMZ between Azure and the Internet

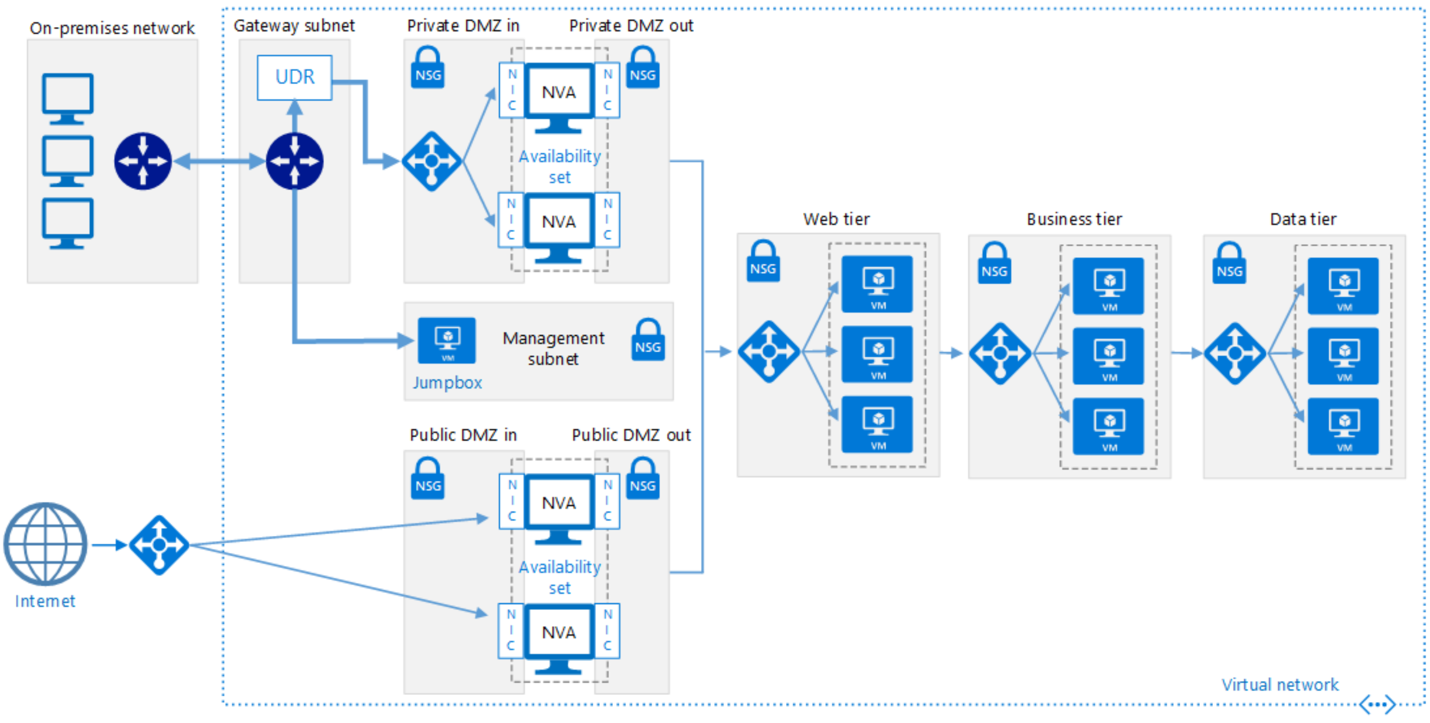
10/22/2018 • 5 minutes to read • Contributors all

In this article

This reference architecture shows a secure hybrid network that extends an on-premises network to Azure and also accepts Internet traffic. [Deploy this solution](#).

ⓘ **Note**

This scenario can also be accomplished using [Azure Firewall](#), a cloud-based network security service.



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

This reference architecture extends the architecture described in [Implementing a DMZ between Azure and your on-premises datacenter](#). It adds a public DMZ that handles Internet traffic, in addition to the private DMZ that handles traffic from the on-premises network.

Typical uses for this architecture include:

- Hybrid applications where workloads run partly on-premises and partly in Azure.
- Azure infrastructure that routes incoming traffic from on-premises and the Internet.

Architecture

The architecture consists of the following components.

- **Public IP address (PIP)**. The IP address of the public endpoint. External users connected to the Internet can access the system through this address.
- **Network virtual appliance (NVA)**. This architecture includes a separate pool of NVAs for traffic originating on the Internet.
- **Azure load balancer**. All incoming requests from the Internet pass through the load balancer and are distributed to the NVAs in the public DMZ.
- **Public DMZ inbound subnet**. This subnet accepts requests from the Azure load balancer. Incoming requests are passed to one of the NVAs in the public DMZ.
- **Public DMZ outbound subnet**. Requests that are approved by the NVA pass through this subnet to the internal load balancer for the web tier.

Recommendations

The following recommendations apply for most scenarios. Follow these recommendations unless you have a specific requirement that overrides them.

NVA recommendations

Use one set of NVAs for traffic originating on the Internet, and another for traffic originating on-premises. Using only one set of NVAs for both is a security risk, because it provides no security perimeter between the two sets of network traffic. Using separate NVAs reduces the complexity of checking security rules, and makes it clear which rules correspond to each incoming network request. One set of NVAs implements rules for Internet traffic only, while another set of NVAs implement rules for on-premises traffic only.

Include a layer-7 NVA to terminate application connections at the NVA level and maintain compatibility with the backend tiers. This guarantees symmetric connectivity where response traffic from the backend tiers returns through the NVA.

Public load balancer recommendations

For scalability and availability, deploy the public DMZ NVAs in an availability set and use an Internet facing load balancer to distribute Internet requests across the NVAs in the availability set.

Configure the load balancer to accept requests only on the ports necessary for Internet traffic. For example, restrict inbound HTTP requests to port 80 and inbound HTTPS requests to port 443.

Scalability considerations

Even if your architecture initially requires a single NVA in the public DMZ, we recommend putting a load balancer in front of the public DMZ from the beginning. That will make it easier to scale to multiple NVAs in the future, if needed.

Availability considerations

The Internet facing load balancer requires each NVA in the public DMZ inbound subnet to implement a health probe. A health probe that fails to respond on this endpoint is considered to be unavailable, and the load balancer will direct requests to other NVAs in the same availability set. Note that if all NVAs fail to respond, your application will fail, so it's important to have monitoring configured to alert DevOps when the number of healthy NVA instances falls below a defined threshold.

Manageability considerations

All monitoring and management for the NVAs in the public DMZ should be performed by the jumpbox in the management subnet. As discussed in Implementing a DMZ between Azure and your on-premises datacenter, define a single network route from the on-premises network through the gateway to the jumpbox, in order to restrict access.

If gateway connectivity from your on-premises network to Azure is down, you can still reach the jumpbox by deploying a public IP address, adding it to the jumpbox, and logging in from the Internet.

Security considerations

This reference architecture implements multiple levels of security:

- The Internet facing load balancer directs requests to the NVAs in the inbound public DMZ subnet, and only on the ports necessary for the application.
- The NSG rules for the inbound and outbound public DMZ subnets prevent the NVAs from being compromised, by blocking requests that fall outside of the NSG rules.
- The NAT routing configuration for the NVAs directs incoming requests on port 80 and port 443 to the web tier load balancer, but ignores requests on all other ports.

You should log all incoming requests on all ports. Regularly audit the logs, paying attention to requests that fall outside of expected parameters, as these may indicate intrusion attempts.

Deploy the solution

A deployment for a reference architecture that implements these recommendations is available on GitHub.

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.

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

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

   bash   Copy
   ```bash
   az login
   ```

Deploy resources

1. Navigate to the `/dmz/secure-vnet-dmz` folder of the reference architectures GitHub repository.

2. Run the following command:

   bash   Copy

```
azbb -s <subscription_id> -g <resource_group_name> -l <region> -p onprem.json --deploy
```

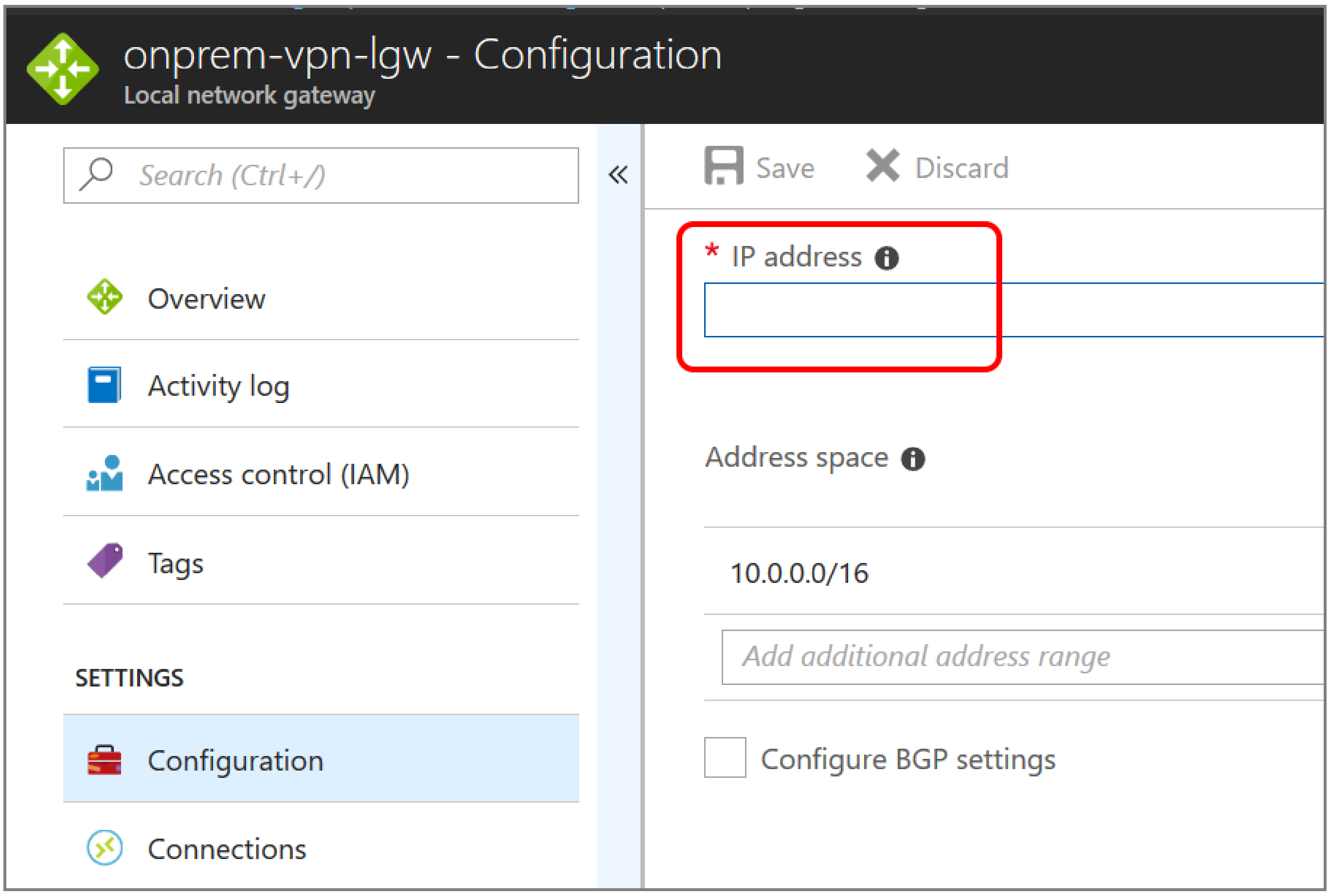
3. Run the following command:

```bash
azbb -s <subscription_id> -g <resource_group_name> -l <region> -p secure-vnet-dmz.json --
deploy
```

Connect the on-premises and Azure gateways

In this step, you will connect the two local network gateways.

1. In the Azure Portal, navigate to the resource group that you created.
2. Find the resource named `ra-vpn-vgw-pip` and copy the IP address shown in the **Overview** blade.
3. Find the resource named `onprem-vpn-lgw`.
4. Click the **Configuration** blade. Under **IP address**, paste in the IP address from step 2.



5. Click **Save** and wait for the operation to complete. It can take about 5 minutes.
6. Find the resource named `onprem-vpn-gateway1-pip`. Copy the IP address shown in the **Overview** blade.
7. Find the resource named `ra-vpn-lgw`.
8. Click the **Configuration** blade. Under **IP address**, paste in the IP address from step 6.
9. Click **Save** and wait for the operation to complete.
10. To verify the connection, go to the **Connections** blade for each gateway. The status should be **Connected**.

Verify that network traffic reaches the web tier

1. In the Azure Portal, navigate to the resource group that you created.
2. Find the resource named `pub-dmz-lb`, which is the load balancer in front of the public DMZ.
3. Copy the public IP address from the **Overview** blade and open this address in a web browser. You should see the default Apache2 server home page.
4. Find the resource named `int-dmz-lb`, which is the load balancer in front of the private DMZ. Copy the private IP address from the **Overview** blade.
5. Find the VM named `jb-vm1`. Click **Connect** and use Remote Desktop to connect to the VM. The user name and password are specified in the onprem.json file.
6. From the Remote Desktop Session, open a web browser and navigate to the IP address from step 4. You should see the default Apache2 server home page.