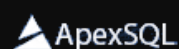




Simulating a Multi Subnet cluster for setting up SQL Server Always On Availability Groups – lab setup

March 14, 2019 by [Sreekanth Bandarla](#)

100% free SQL tools



In this article, we are going to see how to create a multi subnet cluster spanning across multiple subnets for lab purposes. Creating such an environment should help creating Availability groups simulating a far replica residing in a different Data Center (Subnet) acting as a disaster recovery site in your lab for learning/experimenting real world scenarios.

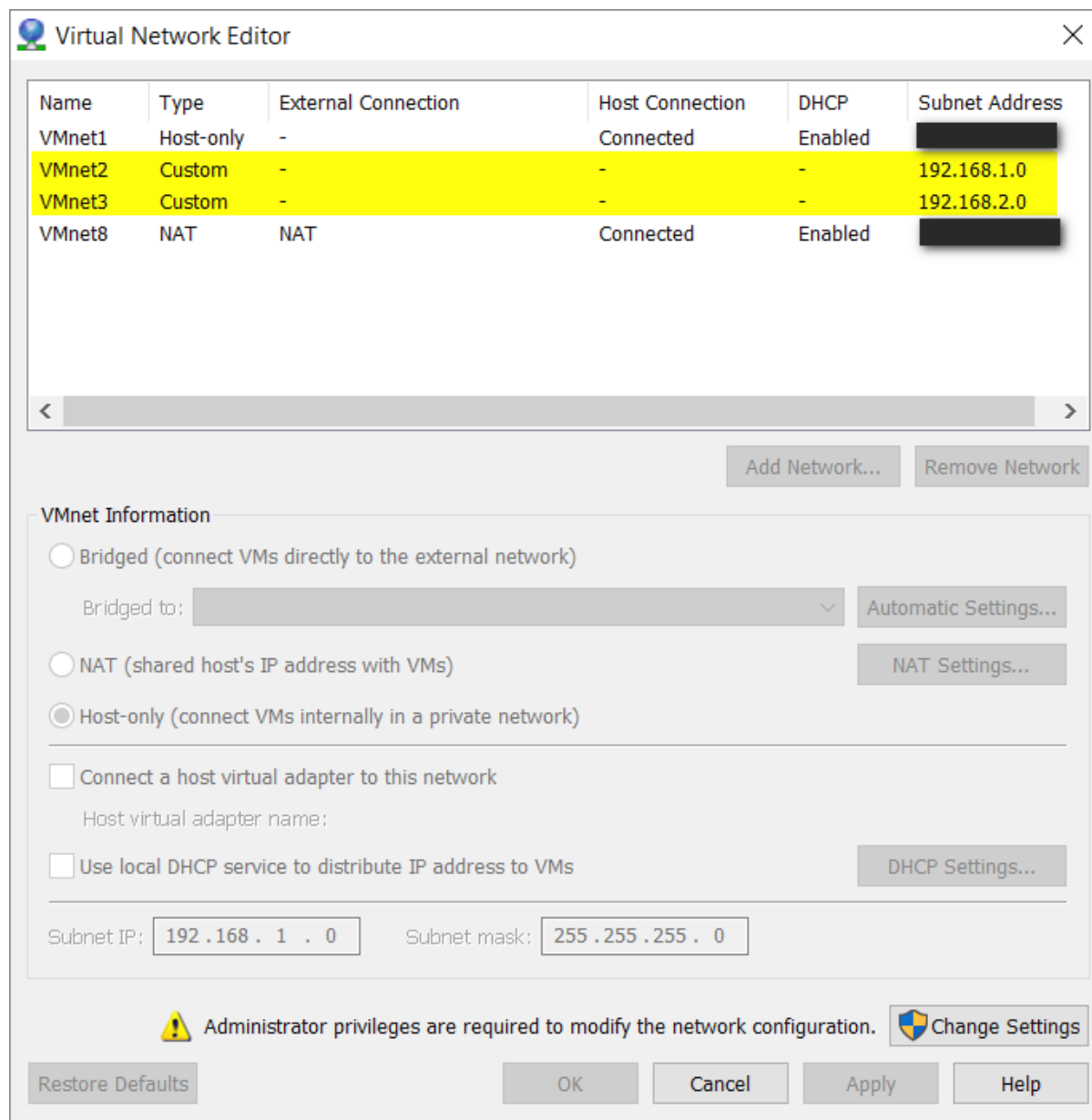
Let's get into action, below is what we are going to setup for our lab purposes to simulate a multi subnet cluster environment and create an Always On Availability group.

- Two nodes (Replicas) residing in my production/primary Data center (Subnet 192.168.1.x) – Let's say these two replicas are used for High availability, assuming we have low latency, since they are residing in the same data center (Probably in the same rack), I will set them up with Synchronous Mode with Automatic Failover
- Third node (Replica) sitting in my disaster recovery data center which is geographically dispersed (Subnet 192.168.2.x) – Assuming it's connected over WAN and is used just for DR purposes, I would set this replica in Asynchronous Mode with Manual Failover

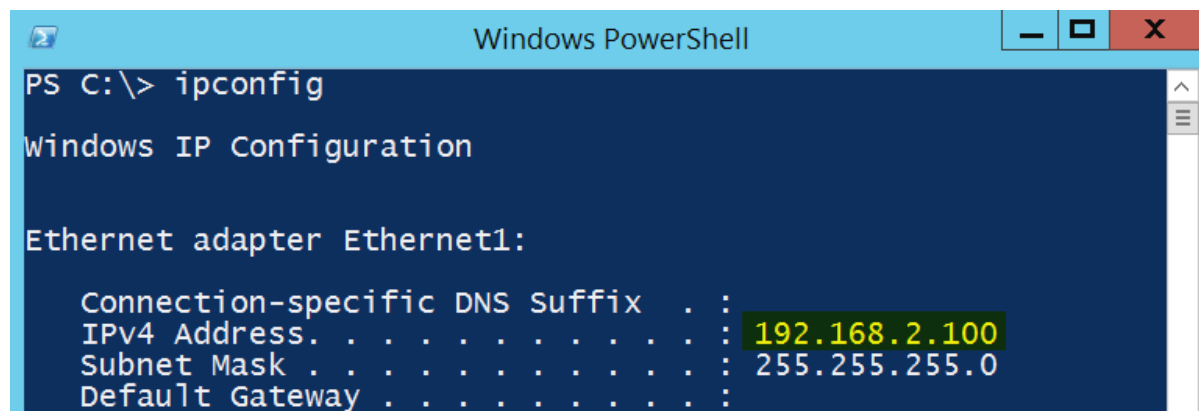
Again, It's not mandatory to always setup near replicas with Synch/Auto mode and far replica with Asynchronous/Manual mode. That's just an example what I would use in my lab environment when creating Always On availability groups to begin with.

So, what exactly do we need to be able to setup multiple subnets and routing in a lab environment using VMware Work station? Well, the answer is "**Routing and Remote Access**" in windows server. In my lab environment, I have a dedicated VM which acts like AD and DNS server (FYI, I use the same VM for provisioning my SAN storage as well). So, the very first step would be installing **Remote Access server role** by going to Add Roles/Features on my AD/DNS server.

Note: I have already created two custom networks in my VMware environment as shown below using virtual editor.



Also, I have created 2 network interface cards on this VM as shown below.



```

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 192.168.1.100
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

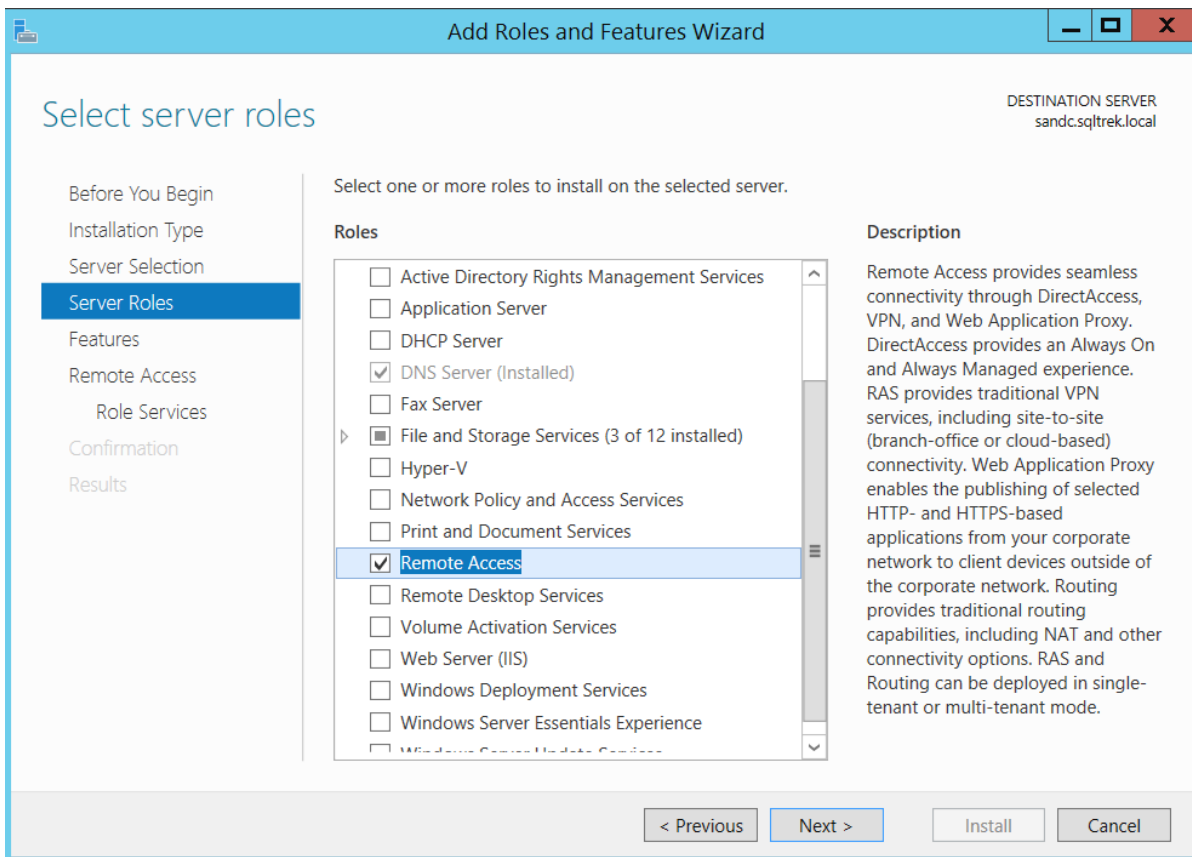
Tunnel adapter isatap.{24EF1E35-C7F0-43AE-A4F4-83D94535752E}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

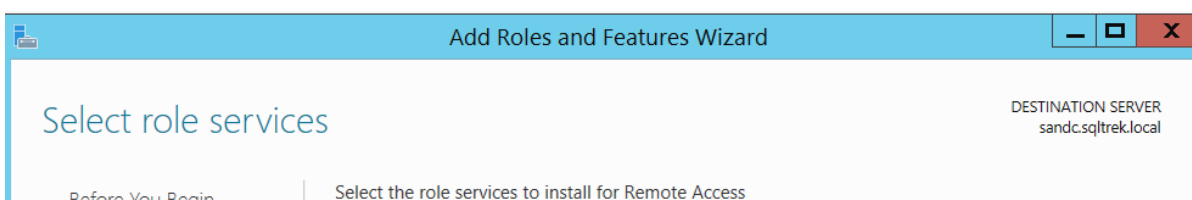
Tunnel adapter isatap.{13A0983D-CBA3-4AA6-BE64-71D212FE8886}:

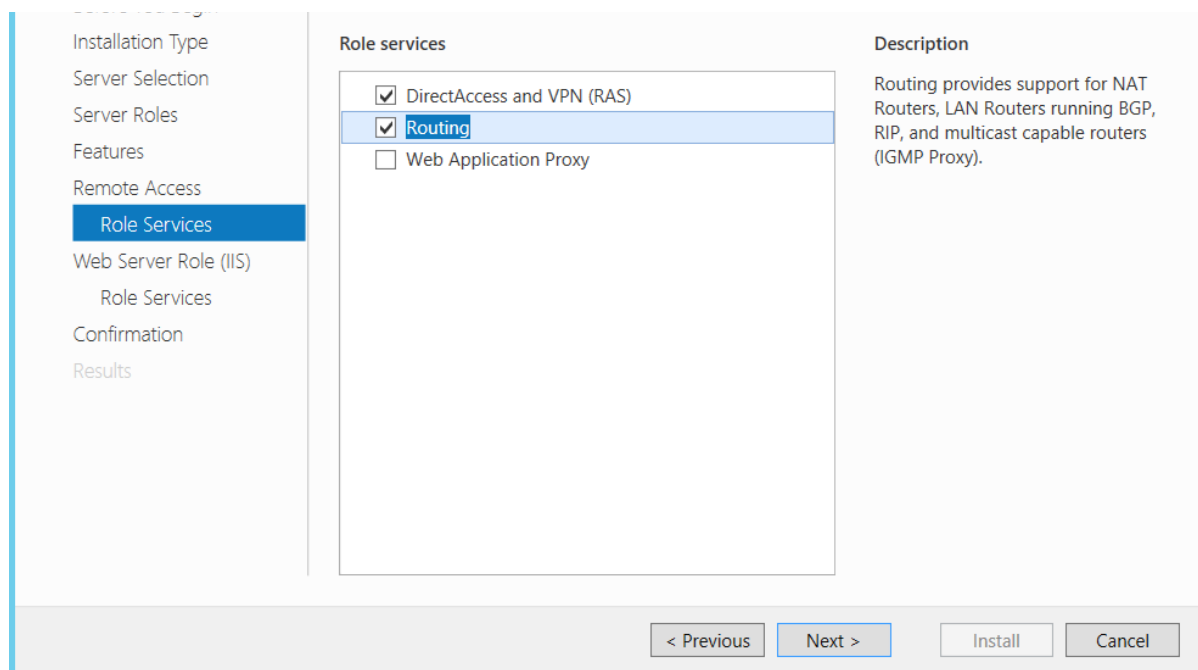
    Media State . . . . . : Media disconnected
  
```

Once this is in place, go to server manager and navigate to new roles and features and select "remote access" as shown below.

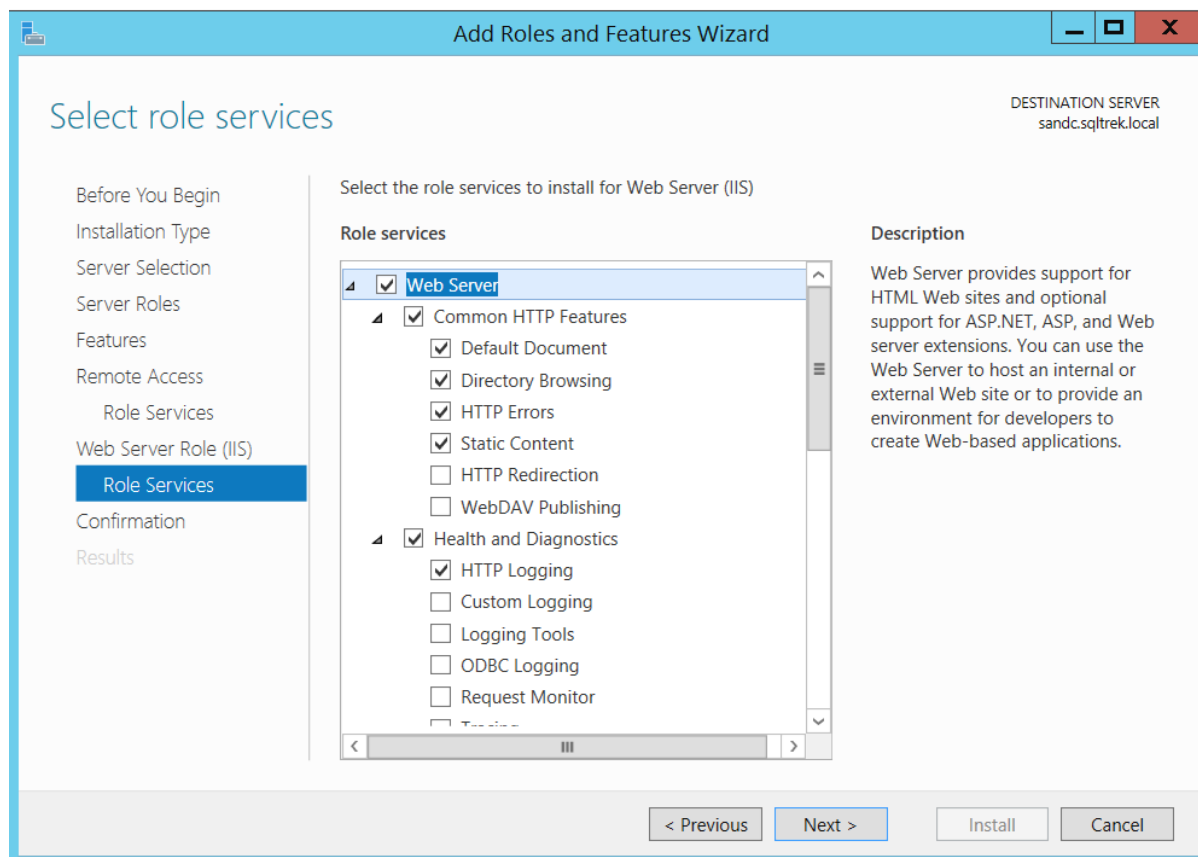


Now click the Next button three times and you should be seeing a window where you get an option to select "Routing" as shown below.

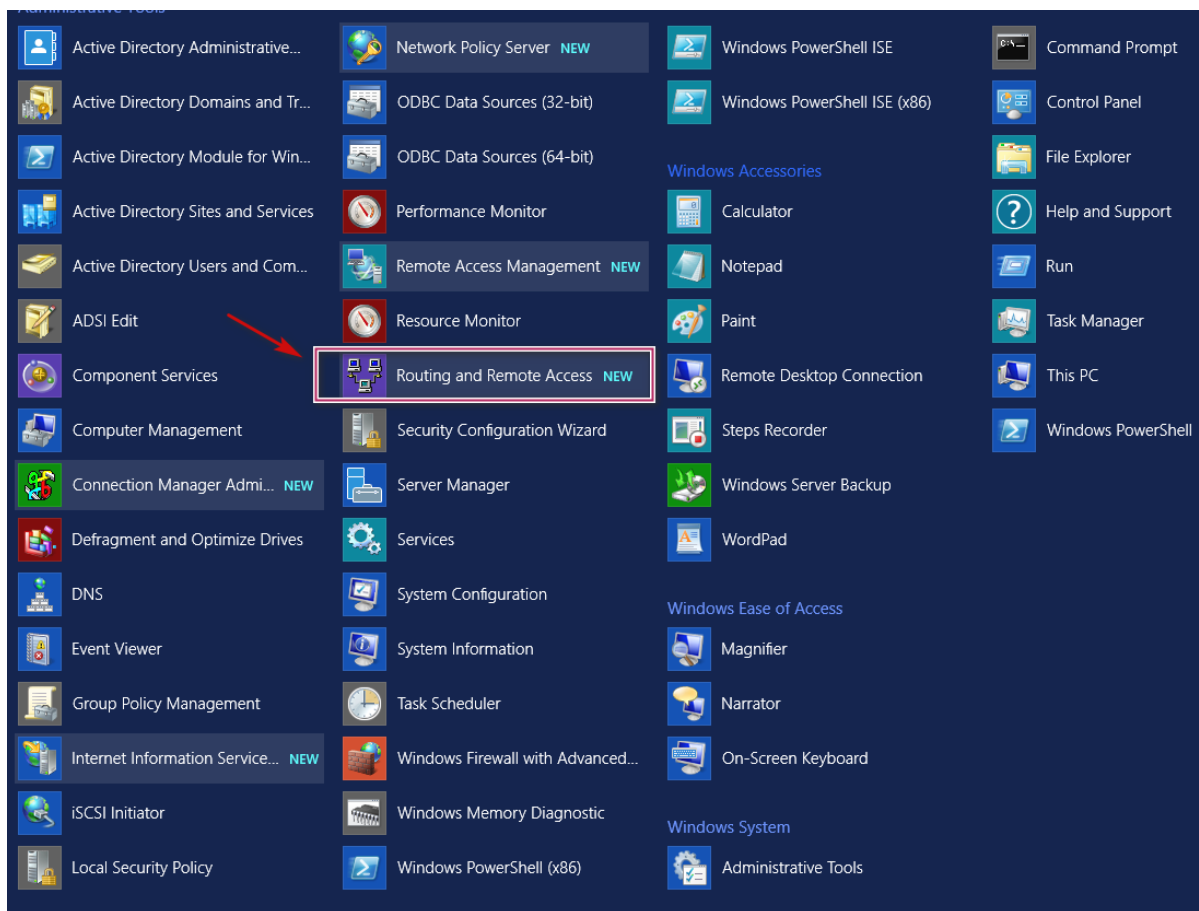




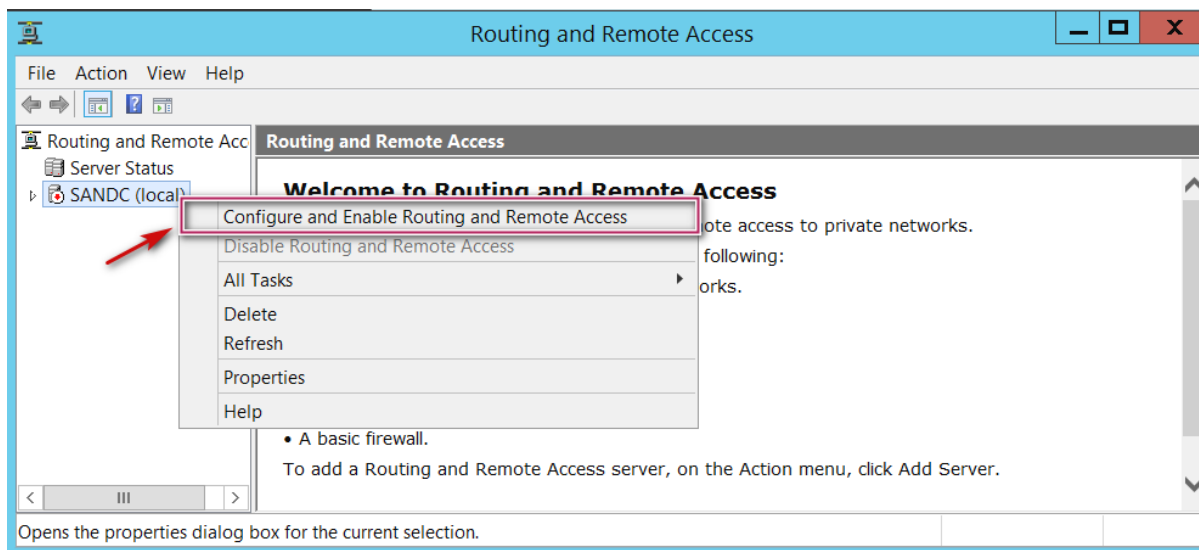
By doing so, DirectAccess and VPN will be automatically selected, click next twice and you get to below screen.



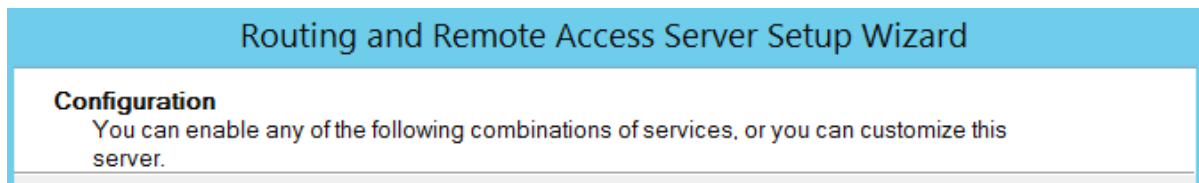
Again, all the required components are automatically selected by windows, just proceed to Next and finish the installation. After successful installation, I was able to search "Routing and Remote Access" as shown below from start menu.



Open Routing and remote access config tool; right click on the root node and select "Enable and Configure Routing and remote access" as shown below.



Click next and select custom configuration and choose "LAN Routing".



☐ Remote access (dial-up or VPN)
Allow remote clients to connect to this server through either a dial-up connection or a secure virtual private network (VPN) Internet connection.

☐ Network address translation (NAT)
Allow internal clients to connect to the Internet using one public IP address.

☐ Virtual private network (VPN) access and NAT
Allow remote clients to connect to this server through the Internet and local clients to connect to the Internet using a single public IP address.

☐ Secure connection between two private networks
Connect this network to a remote network, such as a branch office.

☒ Custom configuration
Select any combination of the features available in Routing and Remote Access.

< Back Next > Cancel

Routing and Remote Access Server Setup Wizard

Custom Configuration
When this wizard closes, you can configure the selected services in the Routing and Remote Access console.

Select the services that you want to enable on this server.

☐ VPN access

☐ Dial-up access

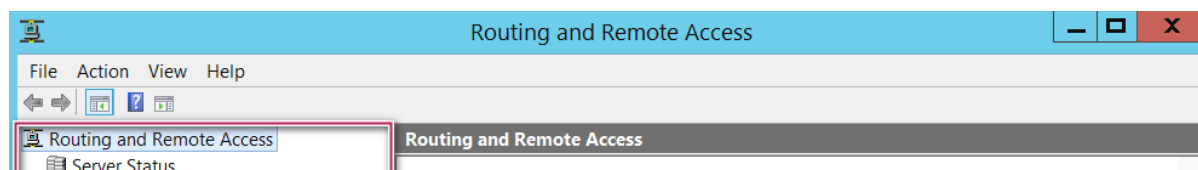
☐ Demand-dial connections (used for branch office routing)

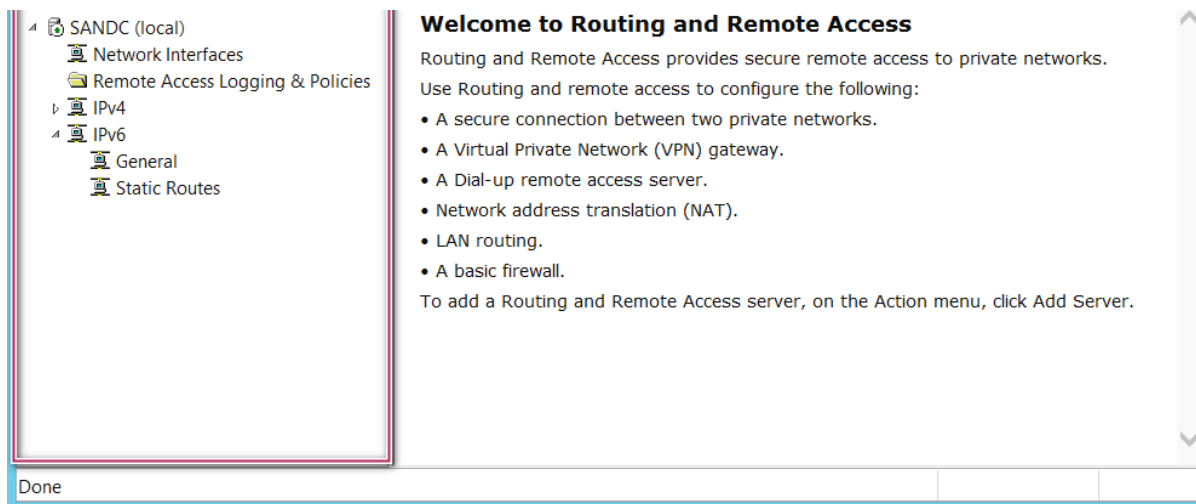
☐ NAT

☒ LAN routing

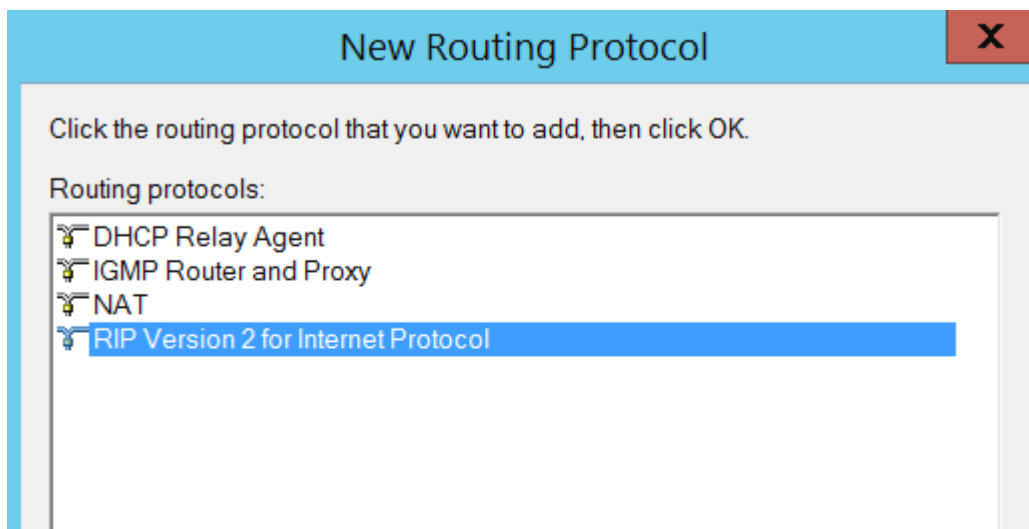
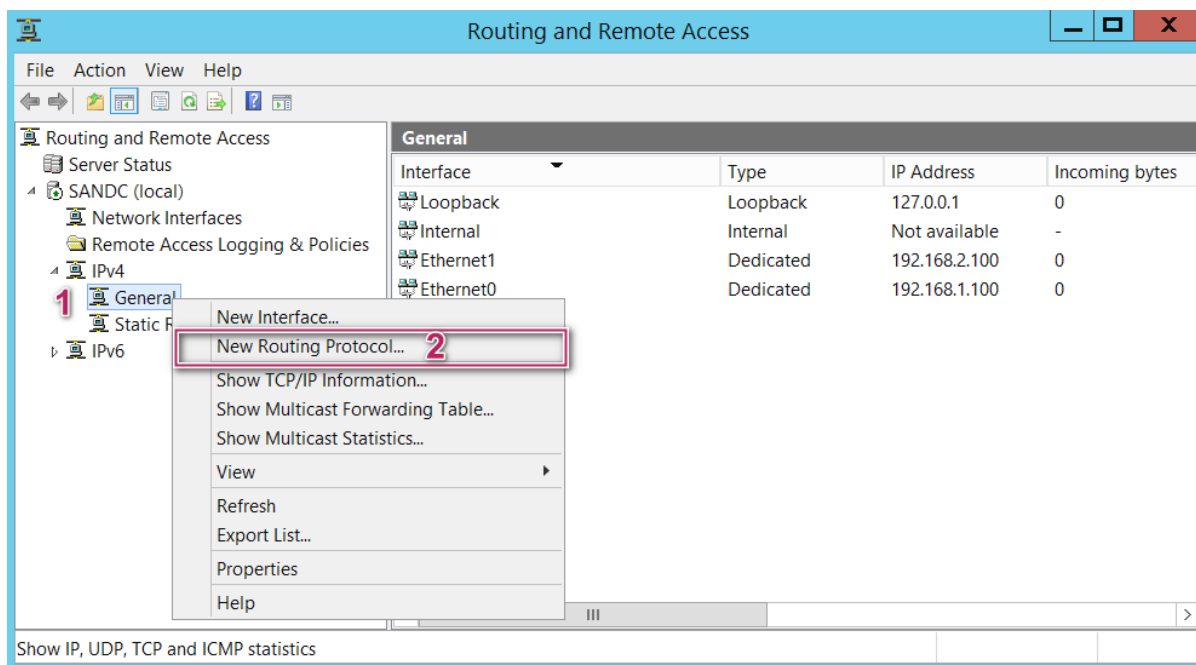
< Back Next > Cancel

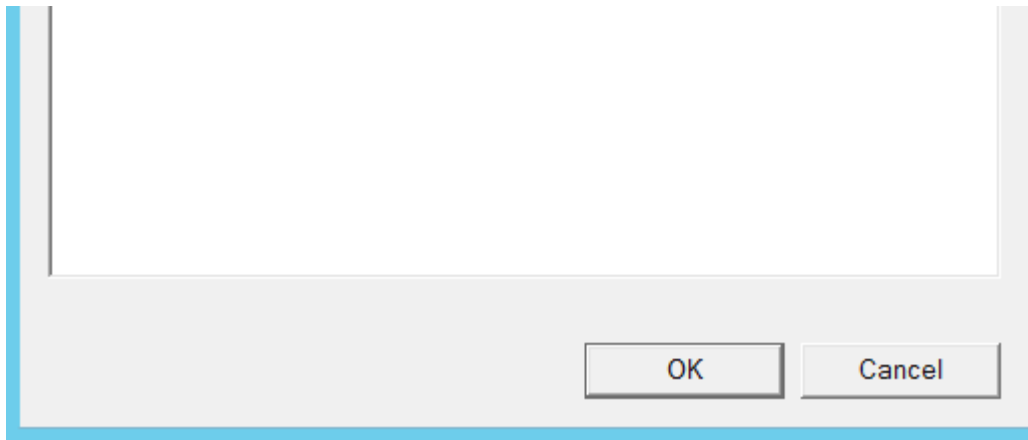
Click **Next**, finish and start the service and you should see the screen below.



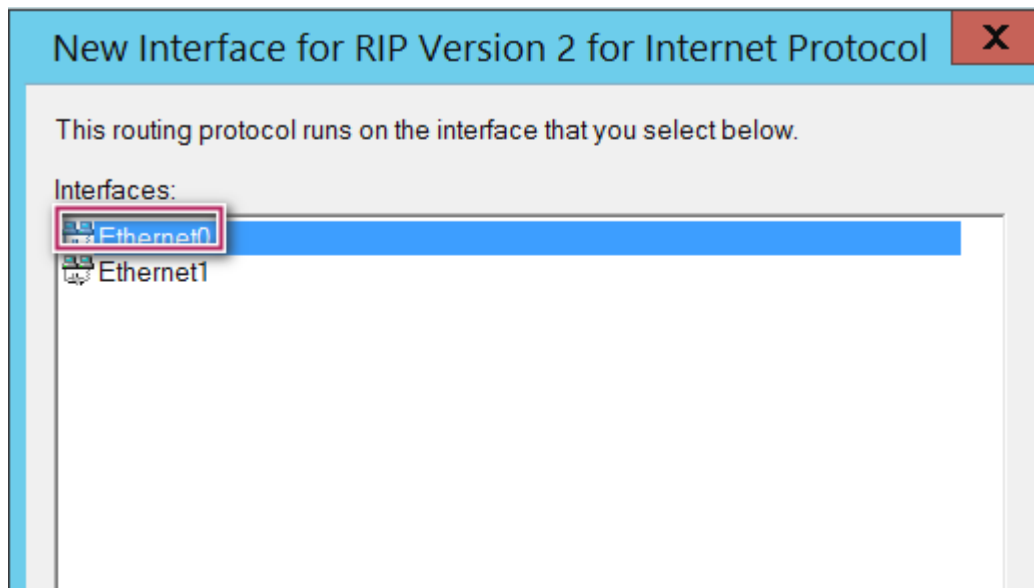
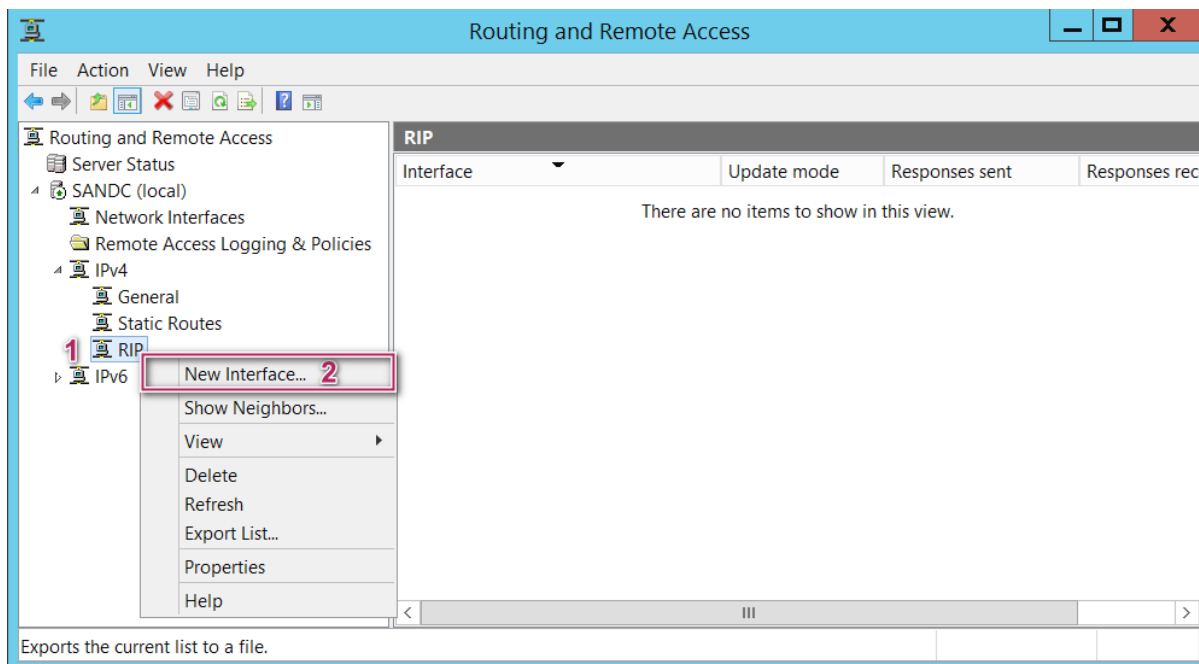


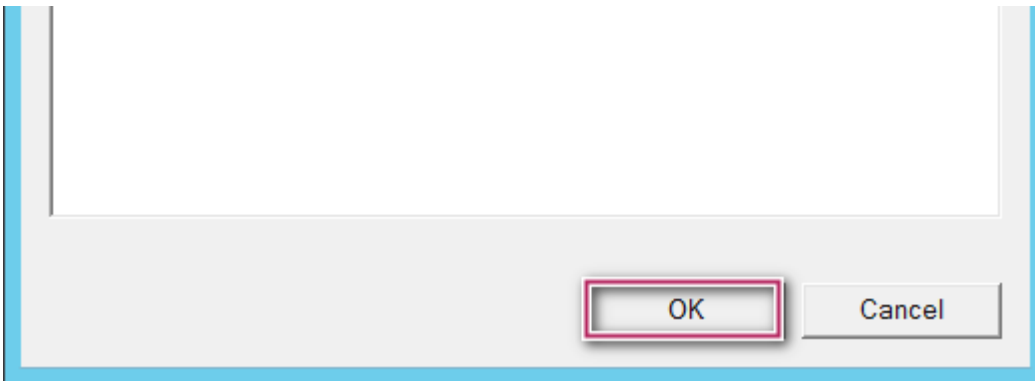
Here comes the interesting piece of the puzzle. You have to navigate to IPV4 section and under General right click and select **"new routing protocol"** and select **"RIP Version 2 for Internet Protocol"** as shown below.



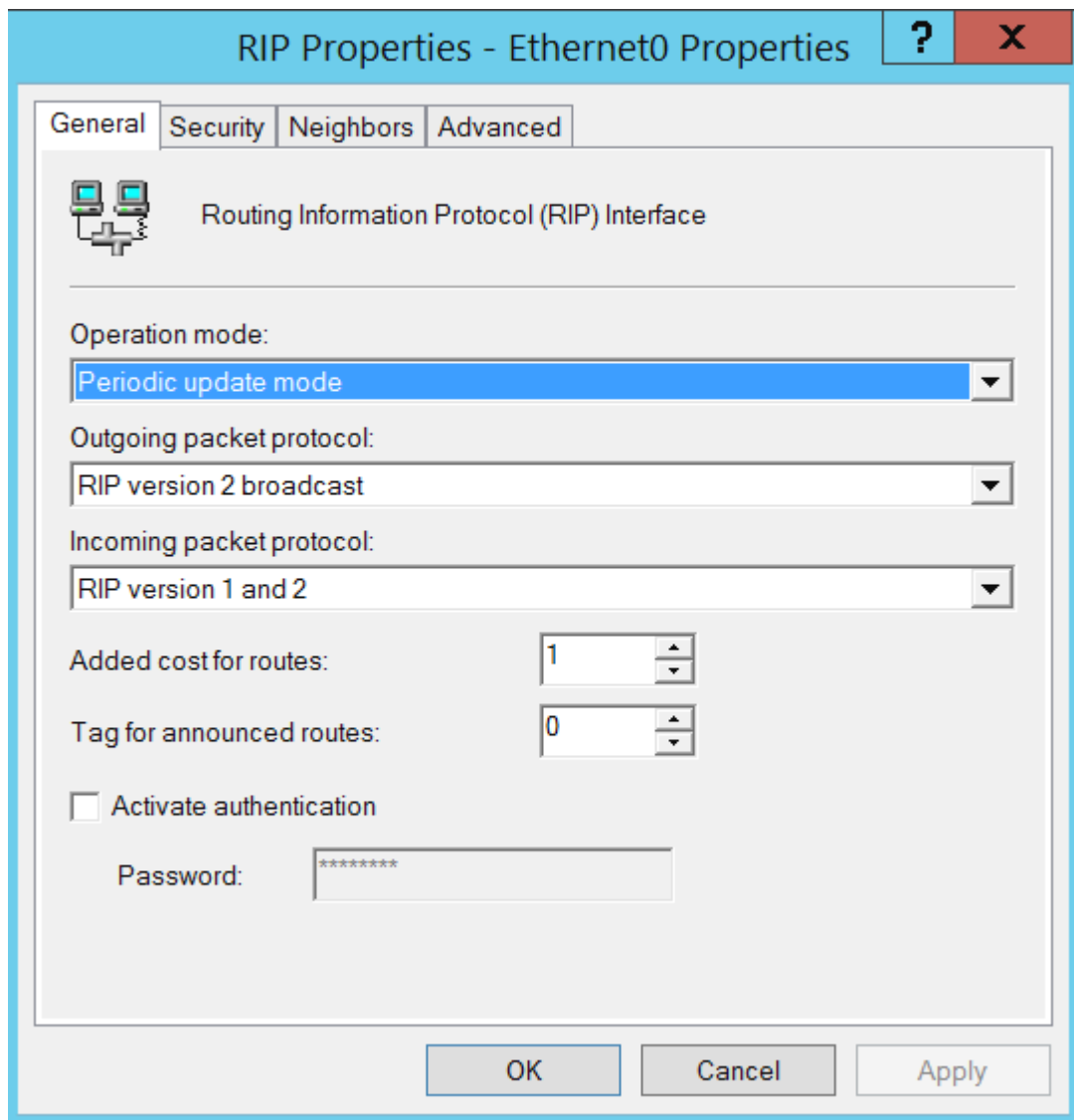


After selecting finish, you should notice a new sub section called "RIP" under IPV4 section. Right click on the RIP under Ipv4 and select new Interface and select a NIC. In my case I chose my first network interface card (Eth 0) and clicked "okay" as shown below to begin with.

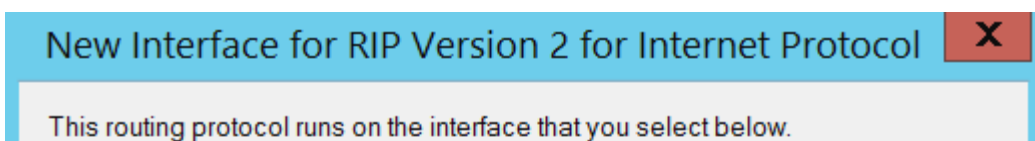


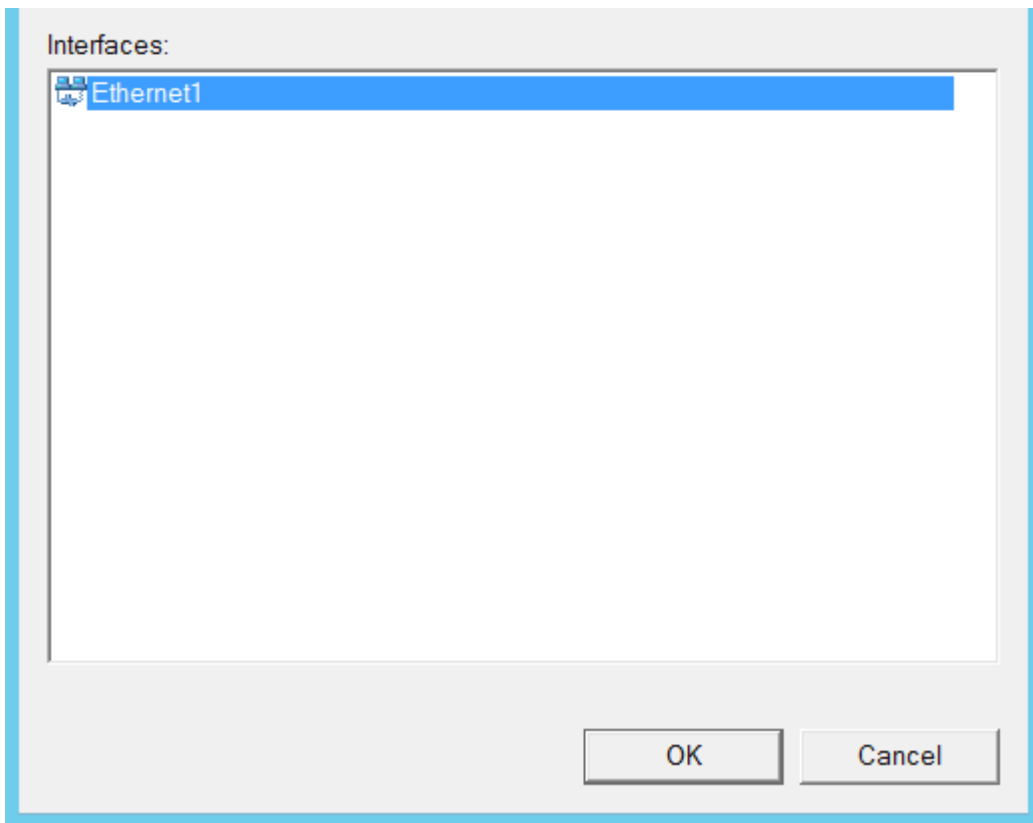


Now you will see below screen, just accept the defaults and click okay again.

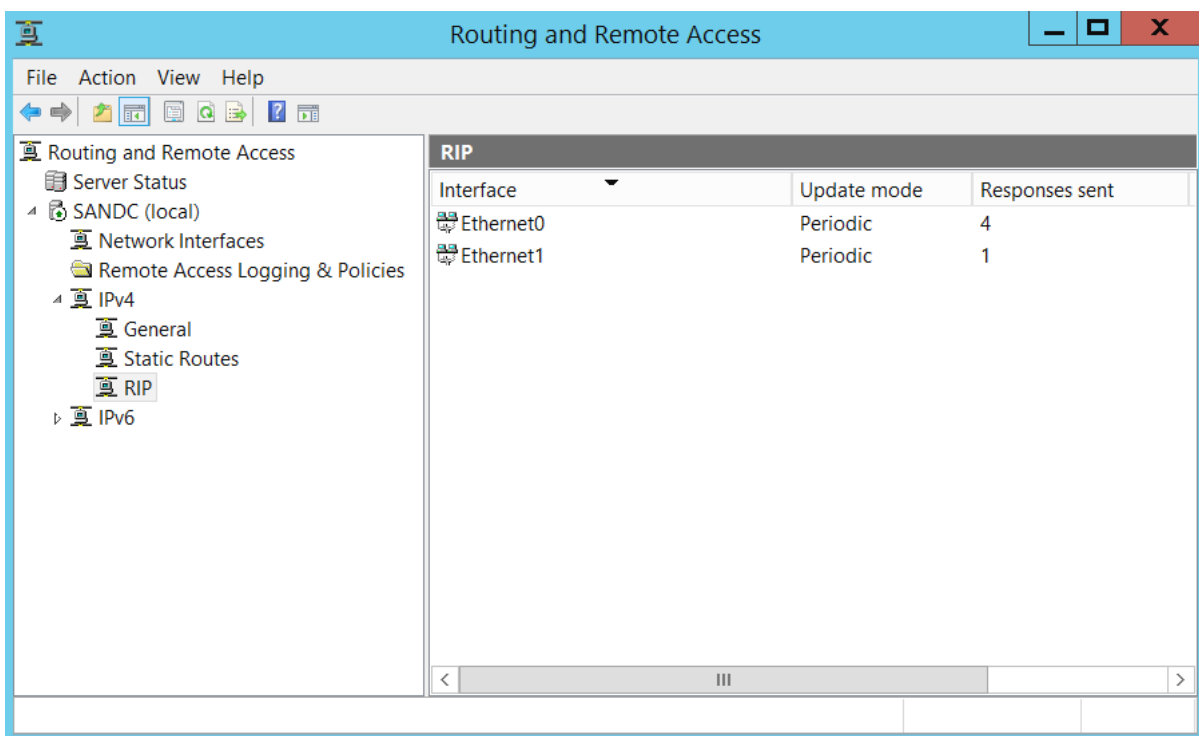


Repeat the same process for your second NIC. You can see, there is only one network interface card (Eth1) listed this time as the first one has already been added to routing.





With all the hard work we did, this is how my end product looks like. Basically, I made my "SANDC" virtual machine act as a router between my subnets, leveraging "Routing" software piece of windows operating system.

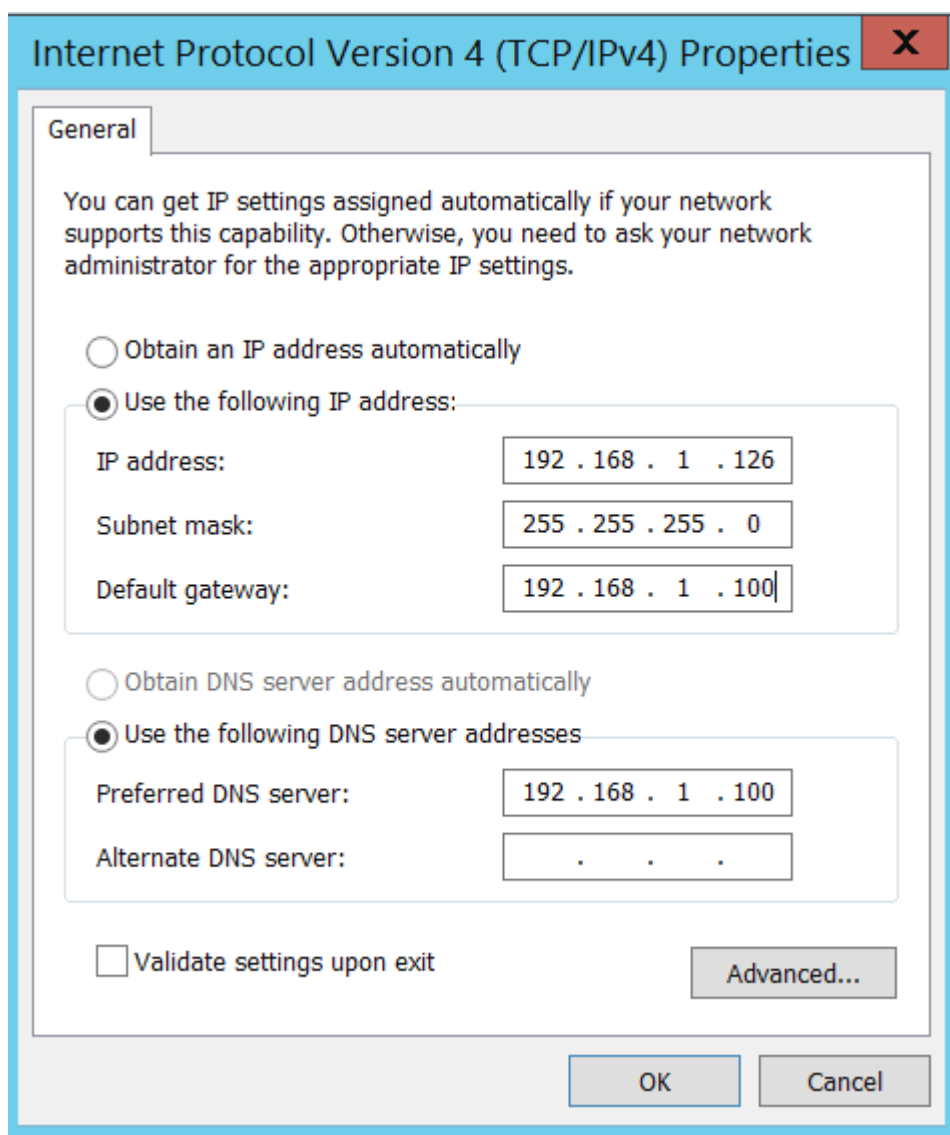


Now let's see what needs to be done on the actual VMs which will act as replicas in our cluster. As I said earlier, I will be creating 2 VMs (Let's say "SQLTPA1" and "SQLTPA2") in 192.168.1.X subnet and one VM ("SQLATL3") in 192.168.2.x subnet and add them to my domain.

Note: I disabled windows firewall on every machine in my lab to make it simple and avoid issues with Ports.

On the first Virtual machine – SQLTPA1:

Go to the properties of the network card and set the Ipv4 address to 192.168.1.126. Set the subnet mask to 255.255.255.0 and the default gateway to 192.168.1.100. I chose the default gateway to be the IP address of NIC 1 on my SANDC VM (That's where the Routing and Remote Access service is running in my lab).



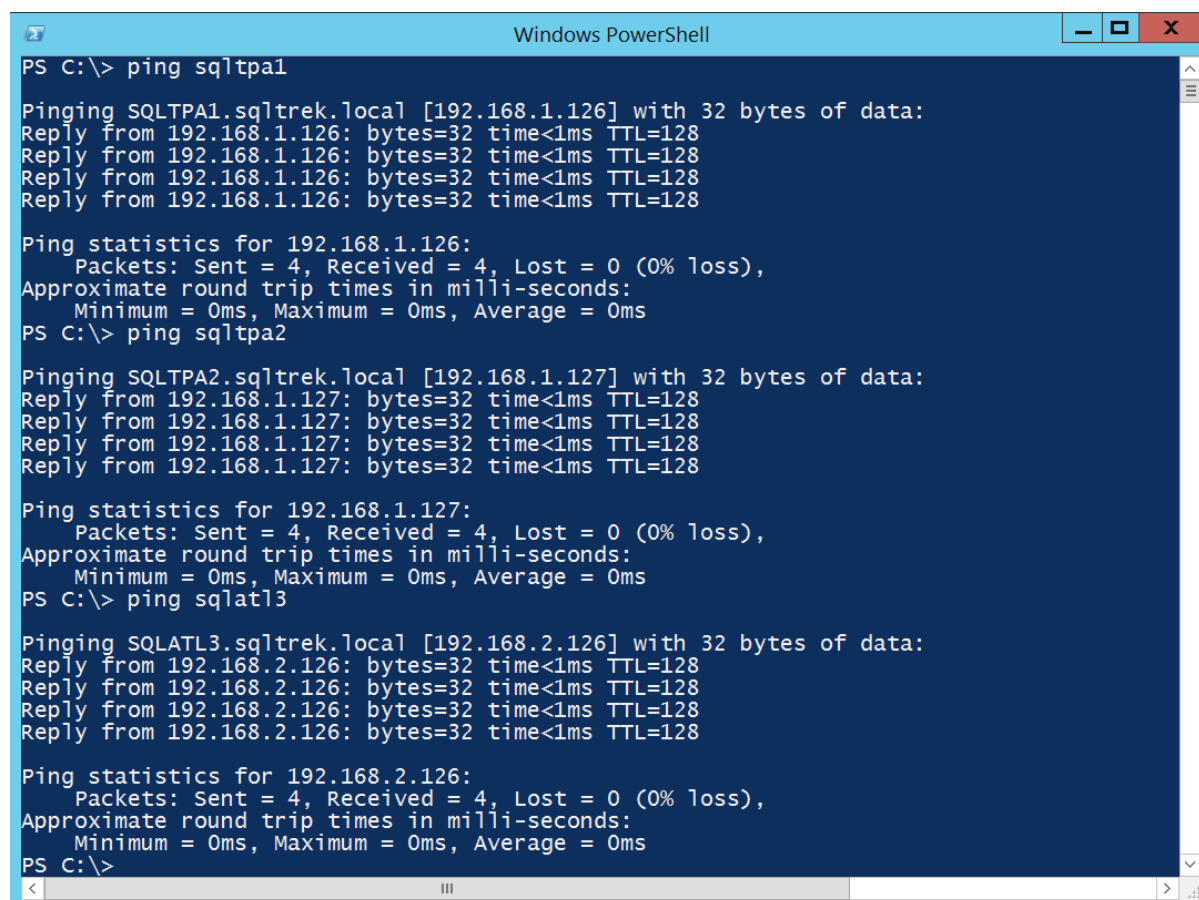
Similarly, I setup my 2nd virtual machine on my production site to IP address 192.168.1.127, Set the subnet mask to 255.255.255.0 and the default gateway to 192.168.1.100.

On the third virtual machine – SQLATL3:

Go to the properties of the network card and set the Ipv4 address to 192.168.2.126. Set the subnet mask to 255.255.255.0 and the default gateway to 192.168.2.100. I chose the default gateway to be the IP address of my second NIC on my SANDC VM (That's where the Routing and Remote Access service is running in my lab).

Let's validate all my configuration settings done so far.

From SANDC machine:



```
Windows PowerShell

PS C:\> ping sqltpa1

Pinging SQLTPA1.sqltrek.local [192.168.1.126] with 32 bytes of data:
Reply from 192.168.1.126: bytes=32 time<1ms TTL=128
Reply from 192.168.1.126: bytes=32 time<1ms TTL=128
Reply from 192.168.1.126: bytes=32 time<1ms TTL=128
Reply from 192.168.1.126: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.126:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\> ping sqltpa2

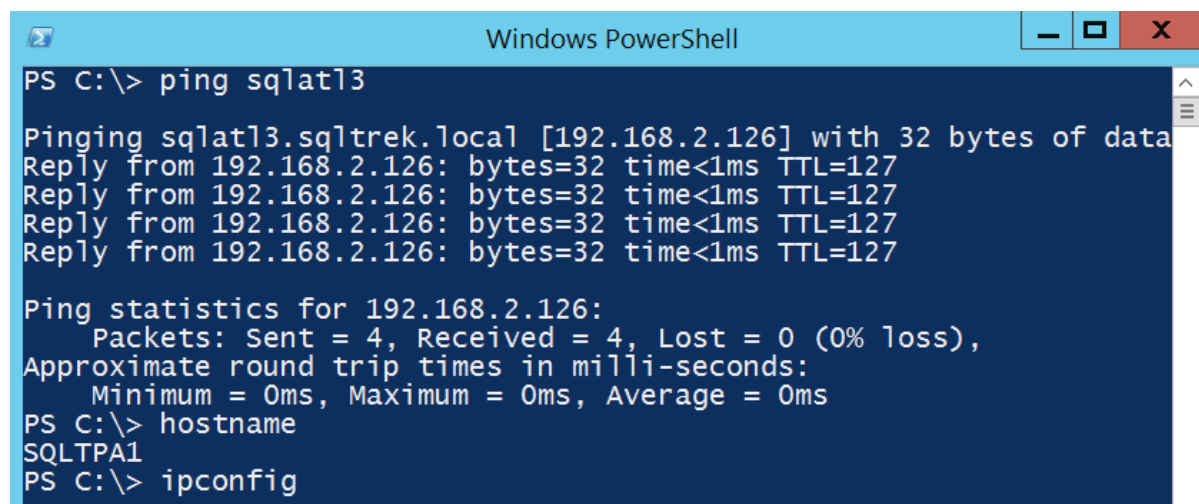
Pinging SQLTPA2.sqltrek.local [192.168.1.127] with 32 bytes of data:
Reply from 192.168.1.127: bytes=32 time<1ms TTL=128
Reply from 192.168.1.127: bytes=32 time<1ms TTL=128
Reply from 192.168.1.127: bytes=32 time<1ms TTL=128
Reply from 192.168.1.127: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.127:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\> ping sqlatl3

Pinging SQLATL3.sqltrek.local [192.168.2.126] with 32 bytes of data:
Reply from 192.168.2.126: bytes=32 time<1ms TTL=128
Reply from 192.168.2.126: bytes=32 time<1ms TTL=128
Reply from 192.168.2.126: bytes=32 time<1ms TTL=128
Reply from 192.168.2.126: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.2.126:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\>
```

Ping test from SQLTPA1 to SQLATL3:



```
Windows PowerShell

PS C:\> ping sqlatl3

Pinging sqlatl3.sqltrek.local [192.168.2.126] with 32 bytes of data:
Reply from 192.168.2.126: bytes=32 time<1ms TTL=127
Reply from 192.168.2.126: bytes=32 time<1ms TTL=127
Reply from 192.168.2.126: bytes=32 time<1ms TTL=127
Reply from 192.168.2.126: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.2.126:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\> hostname
SQLTPA1
PS C:\> ipconfig
```

```
Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 192.168.1.126
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.100

Tunnel adapter isatap.{4E0B773F-6D9F-4D58-94E2-020DCB9E3E04}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 
PS C:\> _
```

Ping test from SQLATL3 to SQLTPA1:

```
Windows PowerShell

PS C:\> ping SQLTPA1

Pinging SQLTPA1.sqltrek.local [192.168.1.126] with 32 bytes of data:
Reply from 192.168.1.126: bytes=32 time<1ms TTL=127
Reply from 192.168.1.126: bytes=32 time<1ms TTL=127
Reply from 192.168.1.126: bytes=32 time<1ms TTL=127
Reply from 192.168.1.126: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.1.126:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\> hostname
SQLATL3
PS C:\> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 192.168.2.126
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.2.100

Tunnel adapter isatap.{721729DC-4E18-45EE-9AB2-4504DFAB2372}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 
PS C:\> _
```

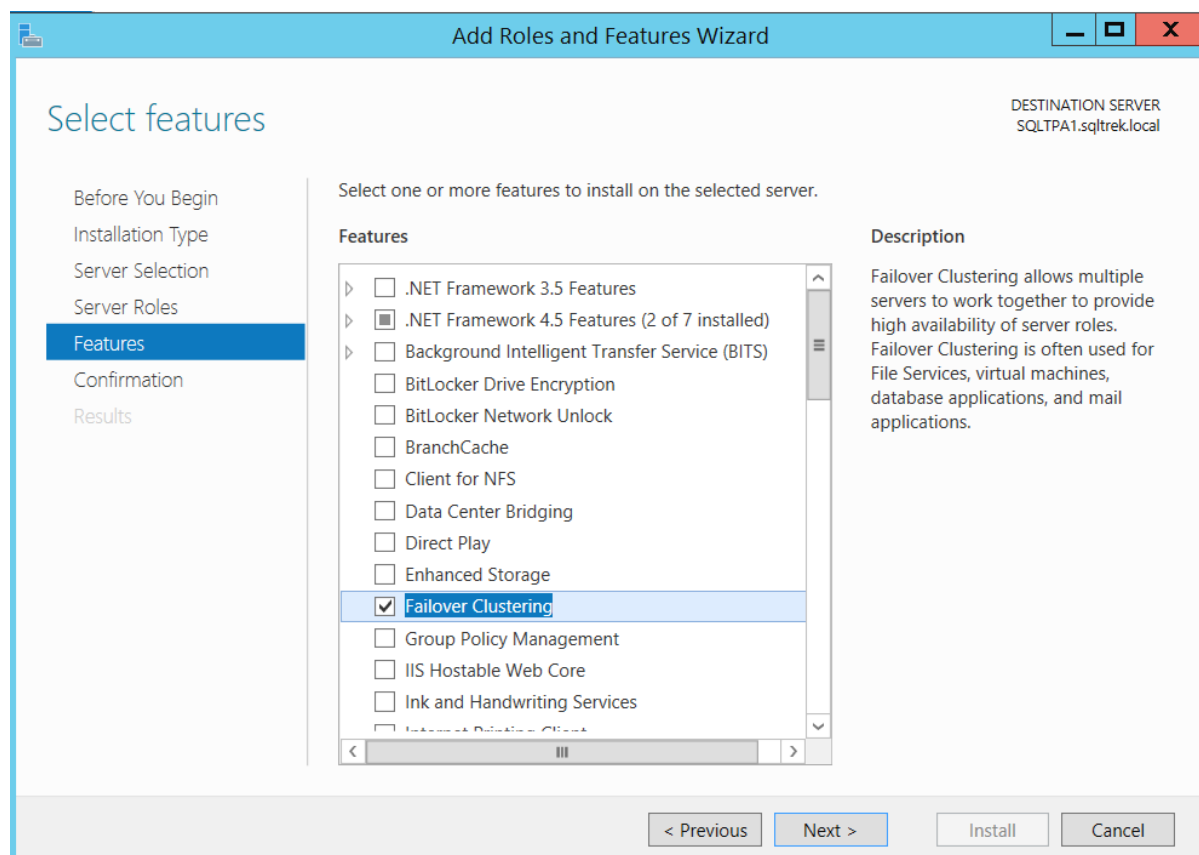
Perfect, we did it. This is how we can setup a lab environment in VMware workstation simulating multiple subnets. Just wanted to make sure before wrapping up this section of our lab setup, this is not something how windows/network admins would setup routing in real world production environments. We would be having a real physical hardware router instead of routing service in windows server 😊, with separate subnets residing in different physical locations (Data Centers) and with a H/W firewall device set with proper rules in place.

So far, we did a walkthrough of setting up a foundation in a lab environment for creating Always On Availability groups spanning multiple subnets. Now, let's build an actual multi subnet windows failover cluster making these machines as nodes and then build an Always On availability group on top of it and explore what and how is it different from an Always On availability group spanning in a single subnet.

Installing and setting up WSFC (Windows Server Failover Clustering):

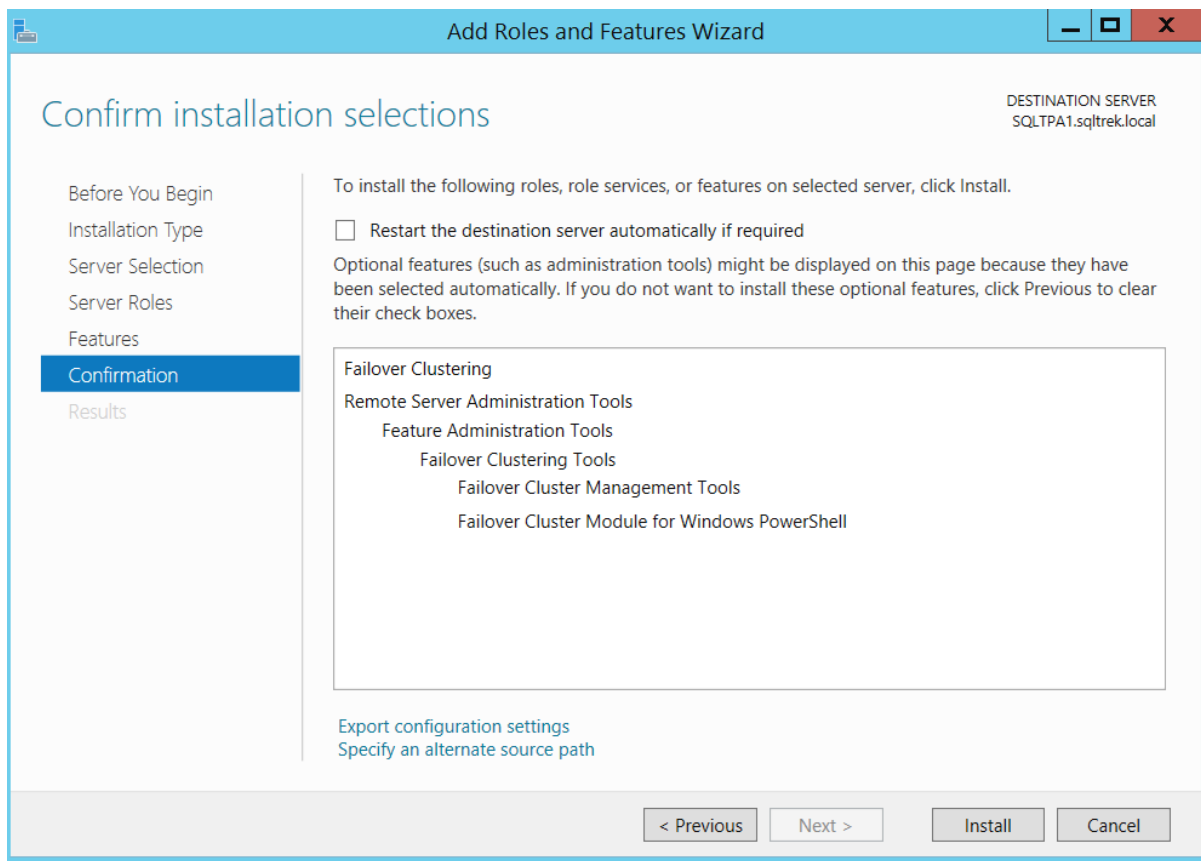
In the previous section of this article, we took care of networking piece of the puzzle for creating a multi subnet cluster in a lab environment using VMware workstation, leveraging windows server routing services. Now, let's get into the actual fun part of creating an Always On availability group simulating multiple data centers, two near replicas in one subnet serving high availability and a far replica serving disaster recovery purposes in a different subnet. Okay, let's get started.

First thing first, we have to install failover clustering feature on all the three nodes which will be participating in our Always On Availability groups. I logged in to all my VMs as a domain admin (I don't want to deal with permission issues now for setting up my lab) and went to server manager and navigated to "Add roles and features" and selected "Failover Clustering" as shown below.

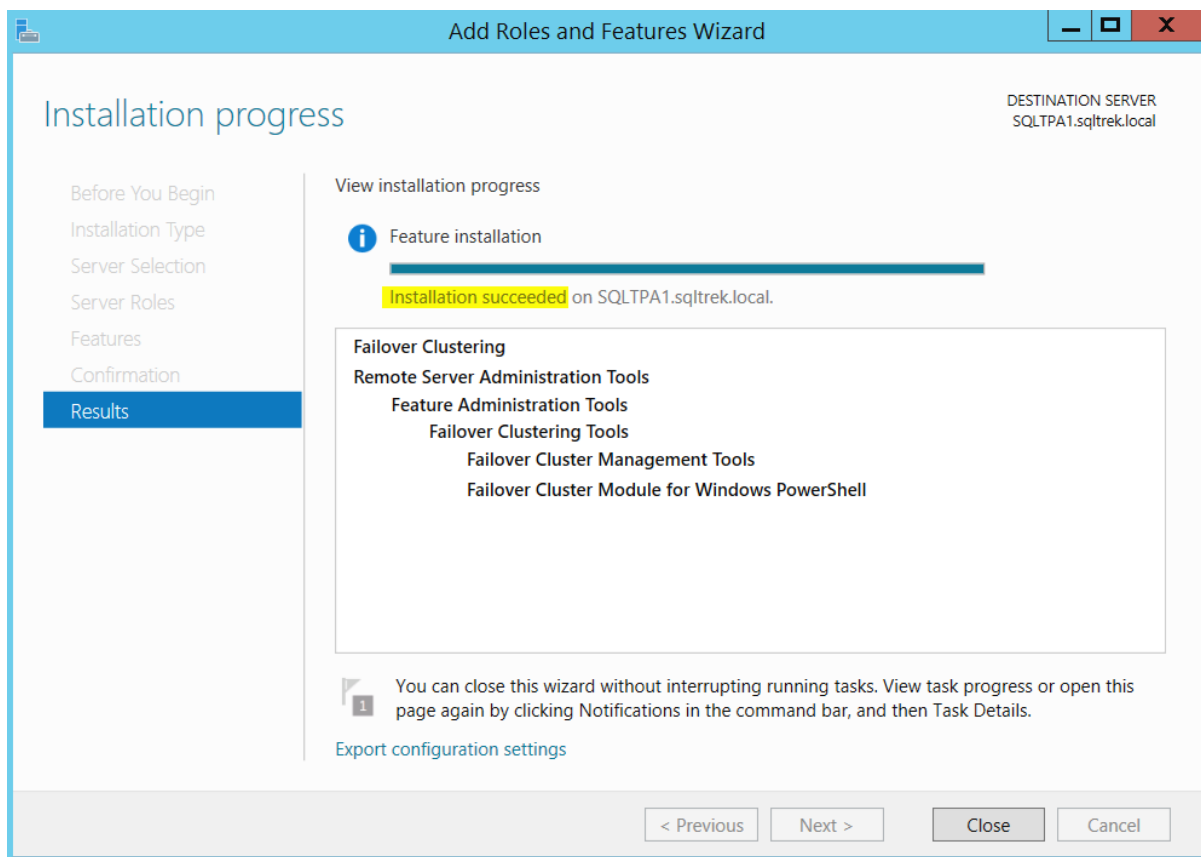


Windows will automatically select all the required sub components as needed, just click on Next and Install

install.



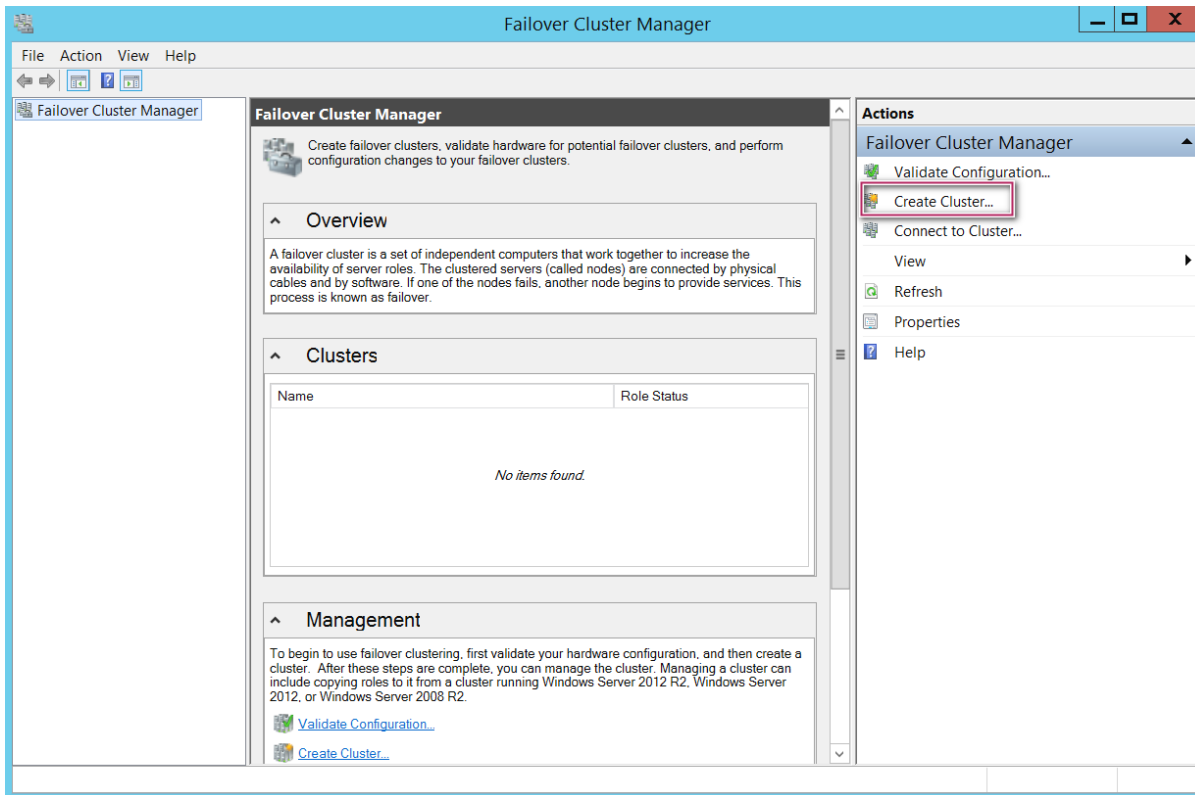
After few seconds, I got the below confirmation.



Well, Once I am done with installing failover clustering feature on all my three virtual machines (SQLTPA1, SQLTPA2 and SQLATL3) it's time to build our cluster.

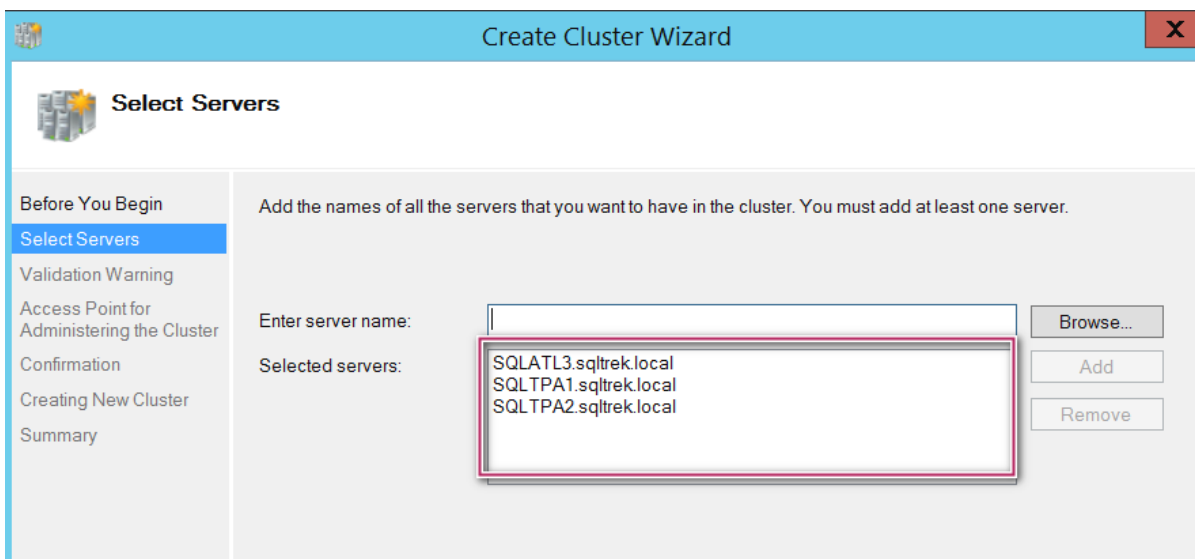
Step 1:

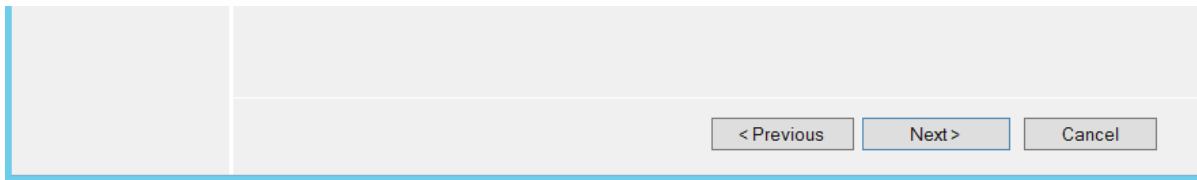
On my SQLTPA1 machine, I opened Failover cluster manager application and selected "Create cluster" as shown below.



Step 2:

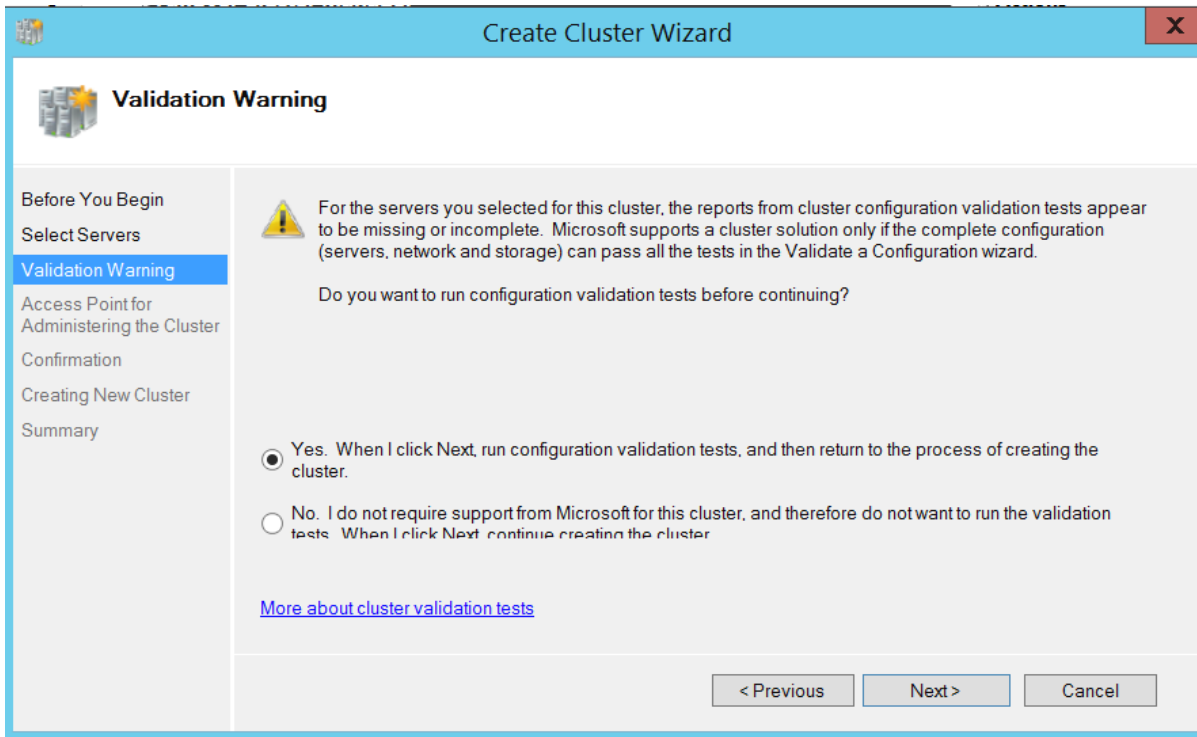
Added all the three nodes by providing FQDN as shown below and click Next.



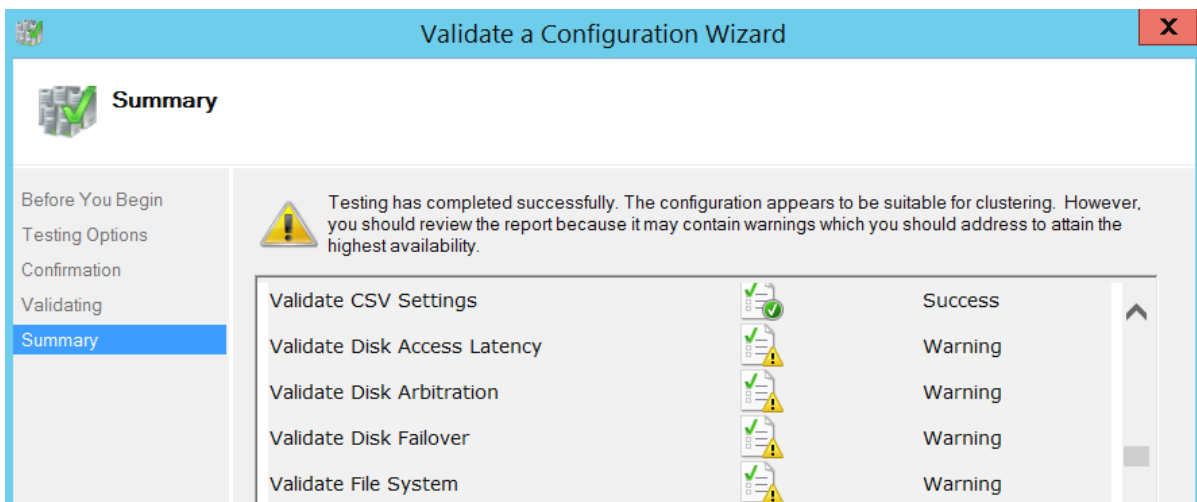


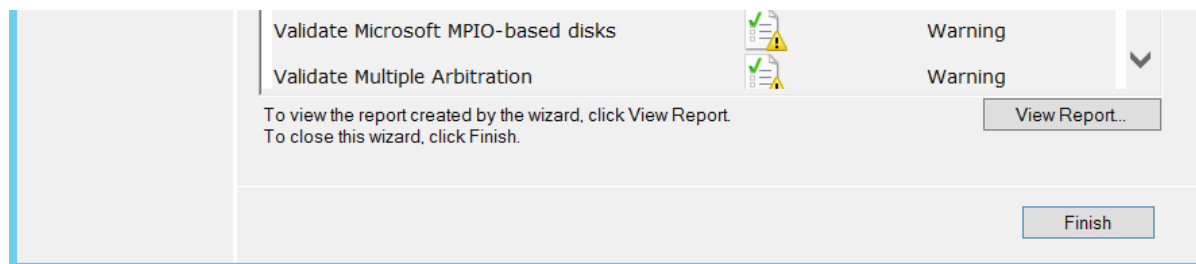
Step 3:

I chose to run validation tests, it's not mandatory to run these tests but It's highly recommended to run these tests to identify potential issues with our environment.



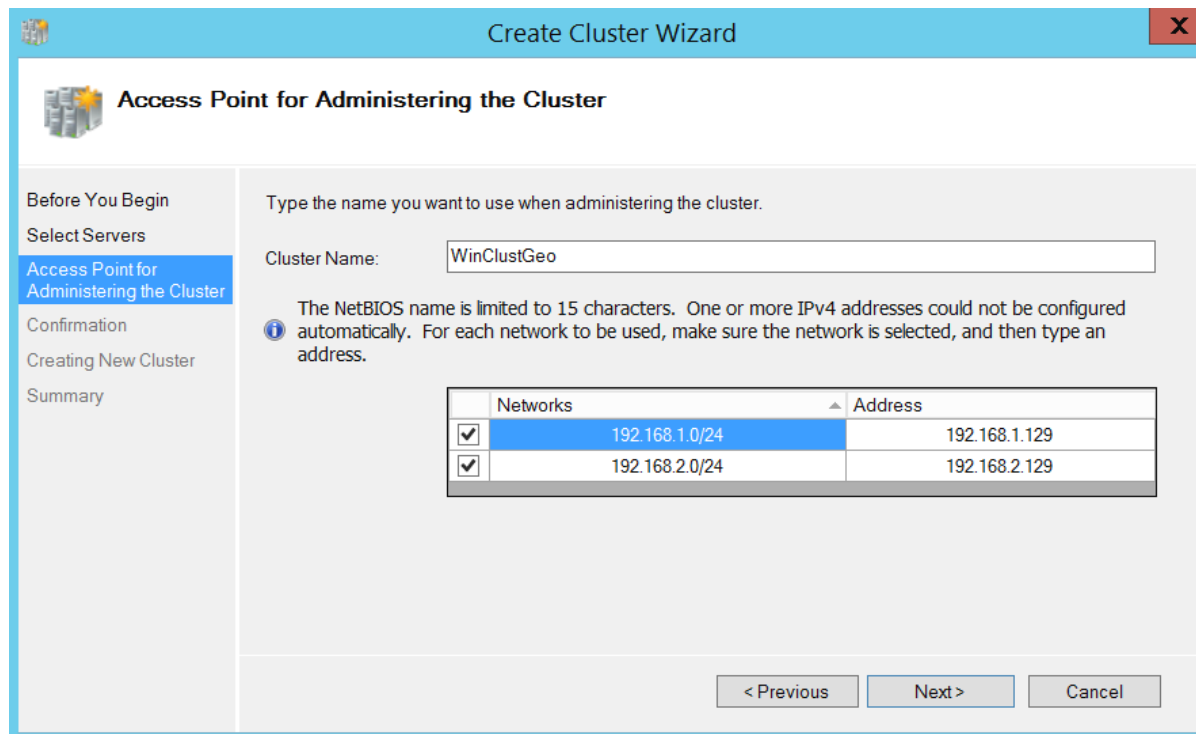
I skipped storage tests in this case as I won't be setting up a traditional SQL server failover cluster for this lab purposes with clustered shared LUNs. (We don't need shared disks for setting up Always On Availability groups). Well, below is how my validation results looked like, Again...I am not worried about warnings related to storage at this point.





Step 4:

I gave a name to my cluster and provided IP addresses, one for each subnet as shown below.



Unchecked "Add eligible storage" and proceed to **Next**.

This completes the process of setting up cluster, as you can see in the below screenshot, I got a confirmation after few seconds.

Tadah!!! Yup, It's really that simple. You can see the create cluster wizard being nice to us and reminding us about Quorum configuration which is the most important piece of any failover cluster setup. After all Quorum is one which dictates the availability and health of a failover cluster and all the dependent Applications/Roles and Services.

Okay.... Now, Let's see how to create a Quorum. Connect to the cluster and navigate to "configure cluster quorum settings" as shown below.

Select advanced configuration option as shown below. (See the reason why in the next step)

As you can see in the below screenshot, I un selected the node which is in my DR subnet. The idea is basically to avoid remote machine sitting in my DR data center deciding the health of my cluster (Doing this will ensure my DR replica vote doesn't count to decide health of my production replicas), so I removed its vote by unchecking my DR node.

Now I chose to use file share witness. (I already have a shared folder which I will be using as a file share witness for this lab setup.)

This completes creating failover cluster. Now it's time to install SQL Server on all the three nodes.

Note: I am not going to cover how to install a standalone SQL Server instance in this article. I went ahead and installed SQL Server 2017 RTM on all the three nodes along with SQL Server Management Studio.

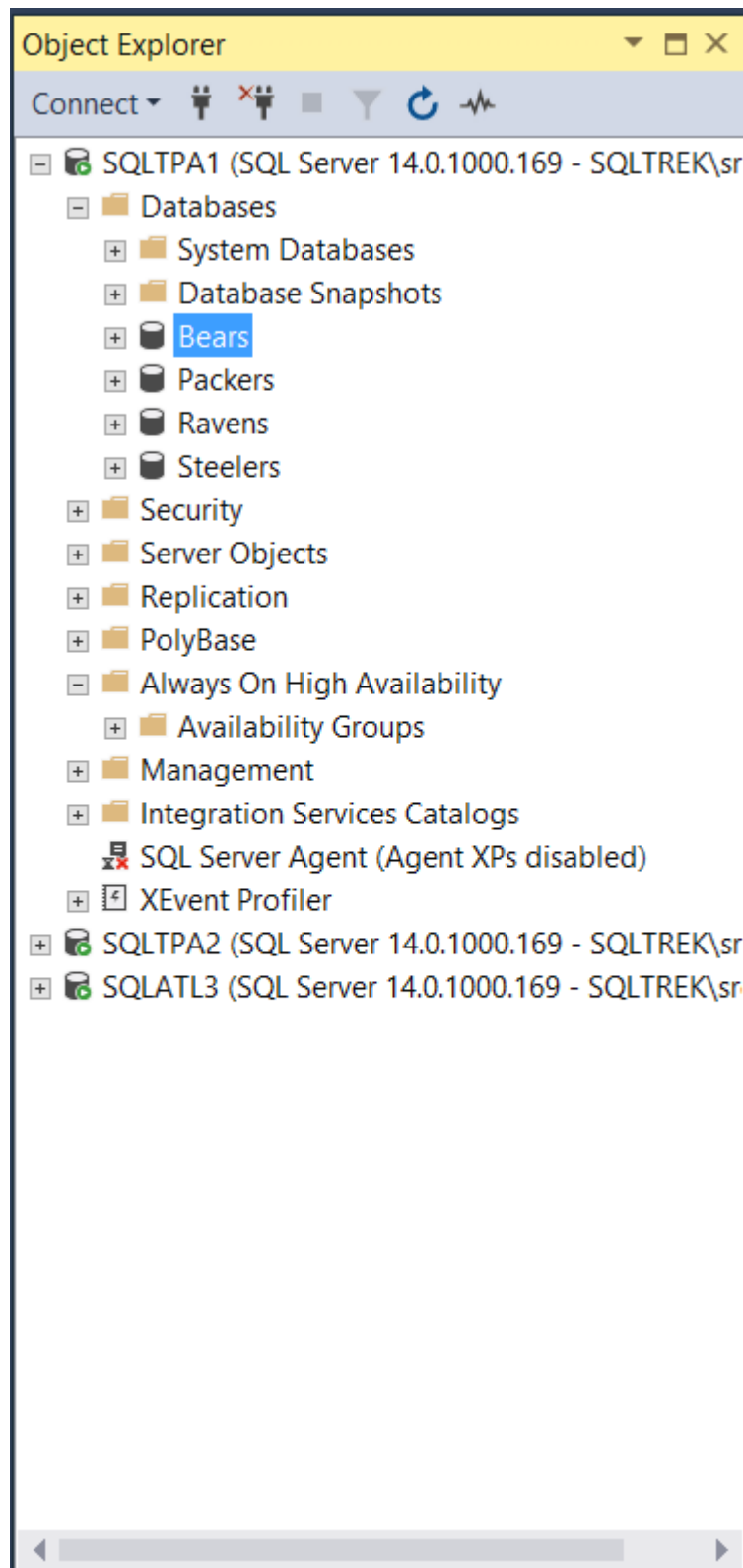
Once the installation is complete, now it's time to enable "Always on high availability" feature by going to SQL Server Configuration manager and navigating to properties of the SQL Instance as shown below.

Restart SQL Server services and we are all set at this point. Once AG feature is enabled on all the nodes (I will refer them as Replicas from now on), we can go ahead and create Always On availability groups.

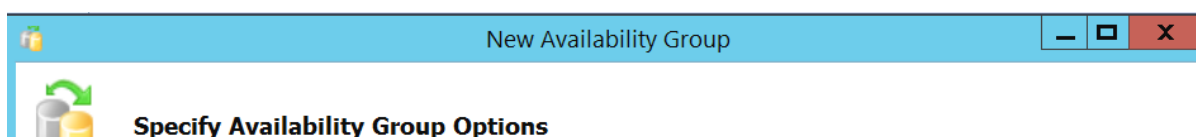
Creating Always On Availability Groups and Listeners:

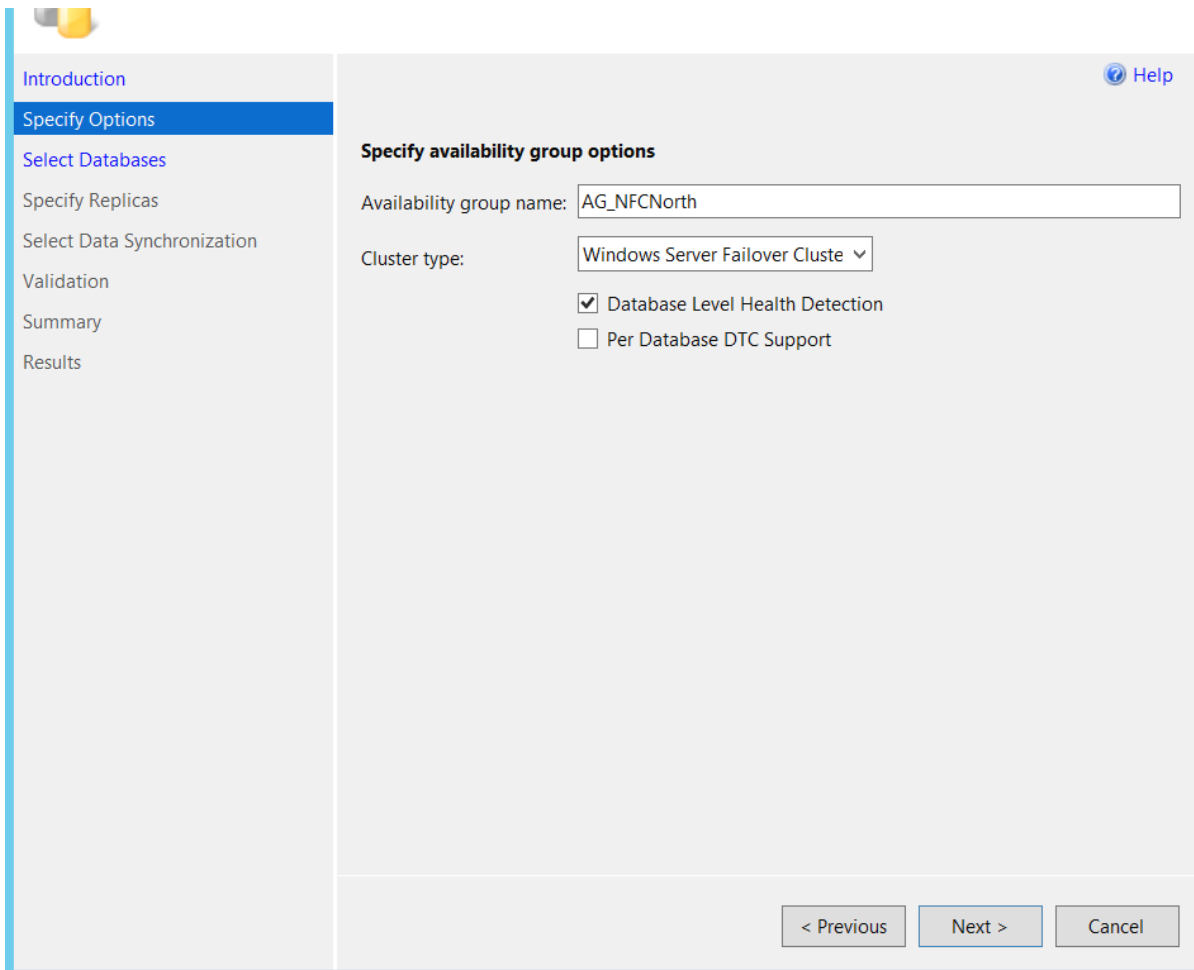
For this demo purpose, I created four databases "Packers", "Bears", "Steelers" and "Ravens". I will be creating an Always On availability group (AG) called "AG_NFCNorth" and place Packers and Bears in it and an AG called "AG_AFCNorth" for Steelers and Ravens databases, the respective listeners will be "list_NFCNorth" and "list_AFCNorth".

Okay, below screenshot shows how it looks like to begin with:



Nothing fancy so far. Now on my SQLTPA1 node, I went to Always On High Availability folder in object explorer and selected "New Availability group wizard". I gave a name to my AG as mentioned earlier and selected "Database level health detection" checkbox (This check is not mandatory).



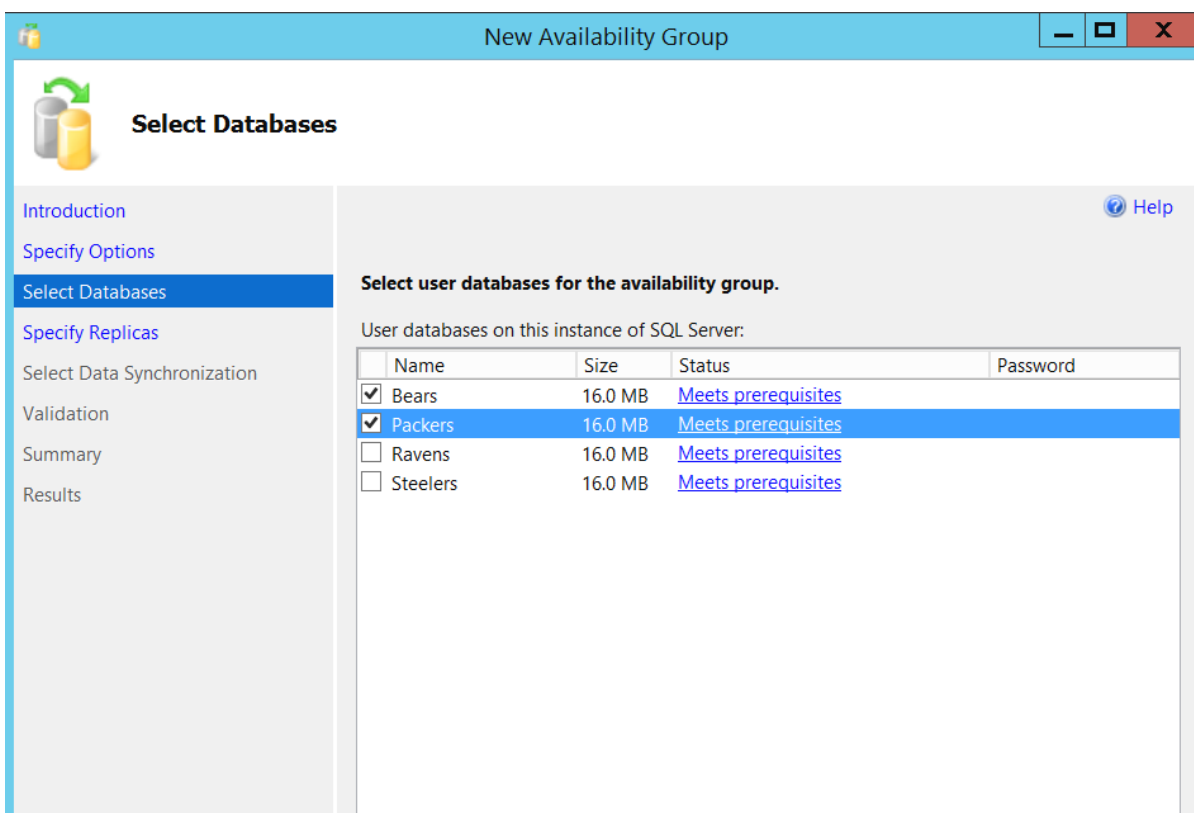


The screenshot shows the 'Specify availability group options' step in the SQL Server Always On Availability Groups wizard. The left sidebar contains a navigation pane with the following items: Introduction, Specify Options (selected), Select Databases, Specify Replicas, Select Data Synchronization, Validation, Summary, and Results. The main area is titled 'Specify availability group options' and contains the following fields and options:

- Availability group name: AG_NFCNorth
- Cluster type: Windows Server Failover Cluste (dropdown menu)
- ☒ Database Level Health Detection
- ☐ Per Database DTC Support

At the bottom right, there are three buttons: '< Previous', 'Next >', and 'Cancel'.

Now, made sure my databases meet prerequisites (Full recovery model and at least one full backup) and selected Packers and Bears databases.



The screenshot shows the 'Select Databases' step in the 'New Availability Group' wizard. The left sidebar contains a navigation pane with the following items: Introduction, Specify Options, Select Databases (selected), Specify Replicas, Select Data Synchronization, Validation, Summary, and Results. The main area is titled 'Select user databases for the availability group.' and contains the following text and table:

User databases on this instance of SQL Server:

	Name	Size	Status	Password
<input checked="" type="checkbox"/>	Bears	16.0 MB	Meets prerequisites	
<input checked="" type="checkbox"/>	Packers	16.0 MB	Meets prerequisites	
<input type="checkbox"/>	Ravens	16.0 MB	Meets prerequisites	
<input type="checkbox"/>	Steelers	16.0 MB	Meets prerequisites	

After adding the required databases, time to specify all our replicas as shown below.

Specify Replicas

Introduction
Specify Options
Select Databases
Specify Replicas
Select Data Synchronization
Validation
Summary
Results

Specify an instance of SQL Server to host a secondary replica.

Replicas | Endpoints | Backup Preferences | Listener | Read-Only Routing

Availability Replicas:

Server Instance	Initial Role	Automatic Failover (Up to 3)	Availability Mode	Readable Secondary
SQLTPA1	Primary	<input checked="" type="checkbox"/>	Synchronous com...	Yes
SQLTPA2	Second...	<input checked="" type="checkbox"/>	Synchronous com...	Yes
SQLATL3	Second...	<input type="checkbox"/>	Asynchronous co...	No

Add Replica... Remove Replica

Summary for the replica hosted by SQLTPA2

Replica mode: Synchronous commit with automatic failover
This replica will use synchronous-commit availability mode and will support both automatic failover and manual failover.

Readable secondary: Yes

Required synchronized secondaries to commit: 0

< Previous Next > Cancel

I left default values for Endpoint and Backup preferences for this lab setup. You can tweak Backup preferences, like giving priorities for replicas and selecting where to run the backups etc as per your requirements. The one which I would like to stress here for our multi subnet cluster lab setup is "Listener" tab.

Note: You can go ahead and skip creating listener at this point and come back later after creating Always On availability group if needed, it's not mandatory to create listener at this stage.

Now, under listener tab, I gave a DNS name for my listener as I mentioned earlier (list_NFCNorth).

New Availability Group

Specify Replicas

Introduction
Specify Options
Select Databases
Specify Replicas
Select Data Synchronization
Validation
Summary
Results

Help

Specify an instance of SQL Server to host a secondary replica.

Replicas Endpoints Backup Preferences **Listener** Read-Only Routing

Specify your preference for an availability group listener that will provide a client connection point:

☐ Do not create an availability group listener now
You can create the listener later using the Add Availability Group Listener dialog.

☒ **Create an availability group listener**
Specify your listener preferences for this availability group.

Listener DNS Name: list_NFCNorth

Port: 1433

Network Mode: Static IP

Subnet	IP Address
--------	------------

Add... Remove

< Previous Next > Cancel

For step 4 (Assigning IP to listener), since this AG is spanning across multiple subnets, we have to provide two IP addresses one for each subnet. This is the part which differs from creating a listener in an AG setup in single subnet.

After clicking on Add button in the above screenshot, I provided an IP for 192.168.1..x subnet as shown below.

Add IP Address

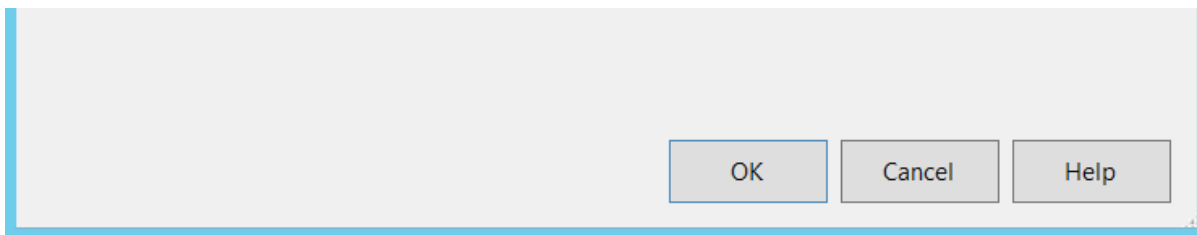
Subnet: 192.168.1.0/24

Address

IPv4 Address: 192.168.1.132

Subnet Mask: 255.255.255.0

IPv6 Address:



Click **OK** and now click on ADD button one more time and provide IP address for your second subnet as shown below.

A dialog box titled "Add IP Address" with a blue header bar. It contains the following fields:

- Subnet:** A dropdown menu showing "192.168.2.0/24".
- Address:** A section containing three input fields:
 - IPv4 Address:** "192.168.2.138"
 - Subnet Mask:** "255.255.255.0"
 - IPv6 Address:** An empty field.

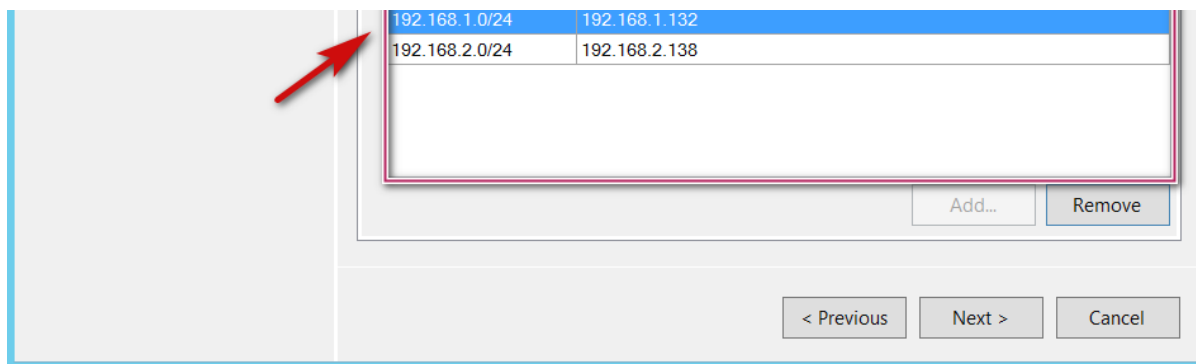
At the bottom are "OK", "Cancel", and "Help" buttons.

Once I am done with adding two IP addresses, below is how my wizard looks like at this stage.

A wizard window titled "New Availability Group" with a blue header bar. The "Specify Replicas" step is active, indicated by a blue bar in the left sidebar. The main area shows instructions to "Specify an instance of SQL Server to host a secondary replica." and tabs for "Replicas", "Endpoints", "Backup Preferences", "Listener", and "Read-Only Routing". The "Listener" tab is selected, showing options to either "Do not create an availability group listener now" or "Create an availability group listener". The "Create an availability group listener" option is selected, and its preferences are shown below:

- Listener DNS Name:** "list_NFCNorth"
- Port:** "1433"
- Network Mode:** "Static IP"

A table at the bottom is partially visible with headers "Subnet" and "IP Address".

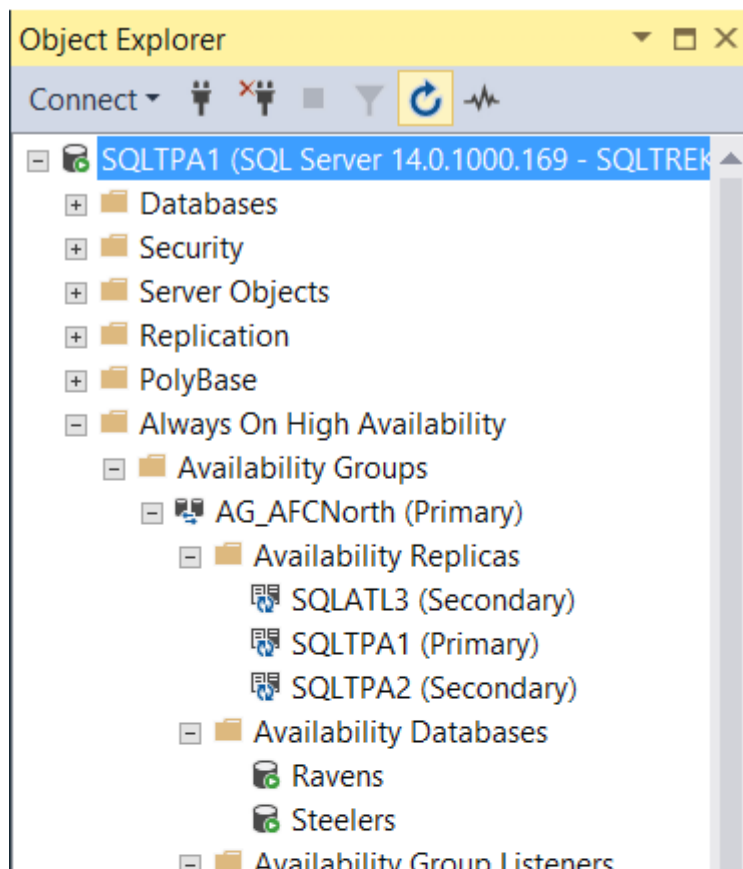


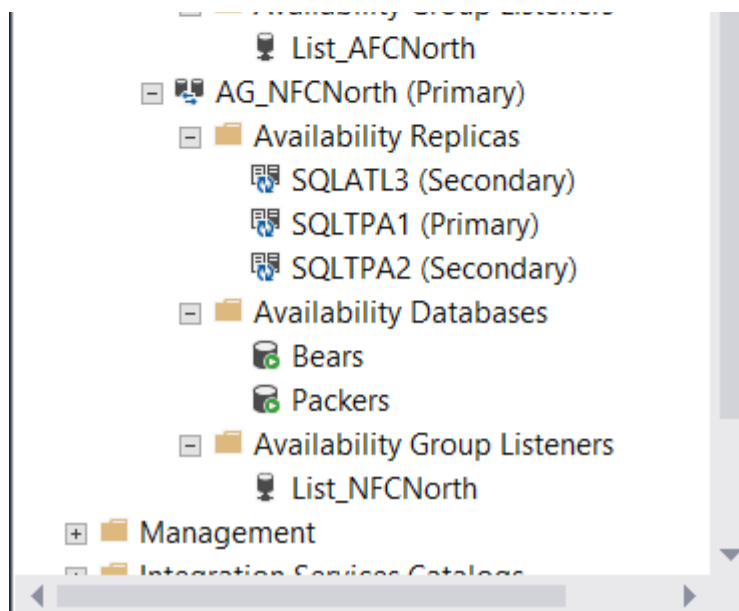
You can see both the IP addresses in the above screenshot, one from each subnet tied to my listener. Click Next and choose how to join the database in Always On availability group. In my lab I have a shared folder which I plan to leverage for this purpose as shown below.

Proceed to next and make sure the validation report is all green as shown below and go to Next and Finish.

It just took couple of minutes for me in my lab setup for this entire process, remember my databases are pretty much empty.

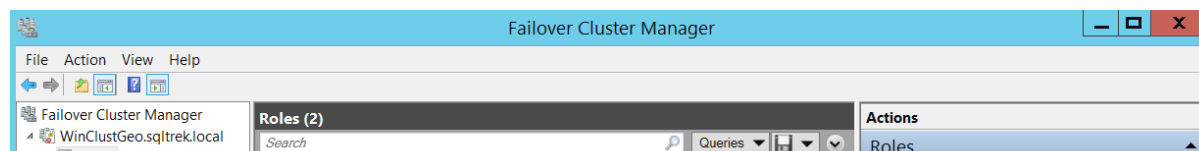
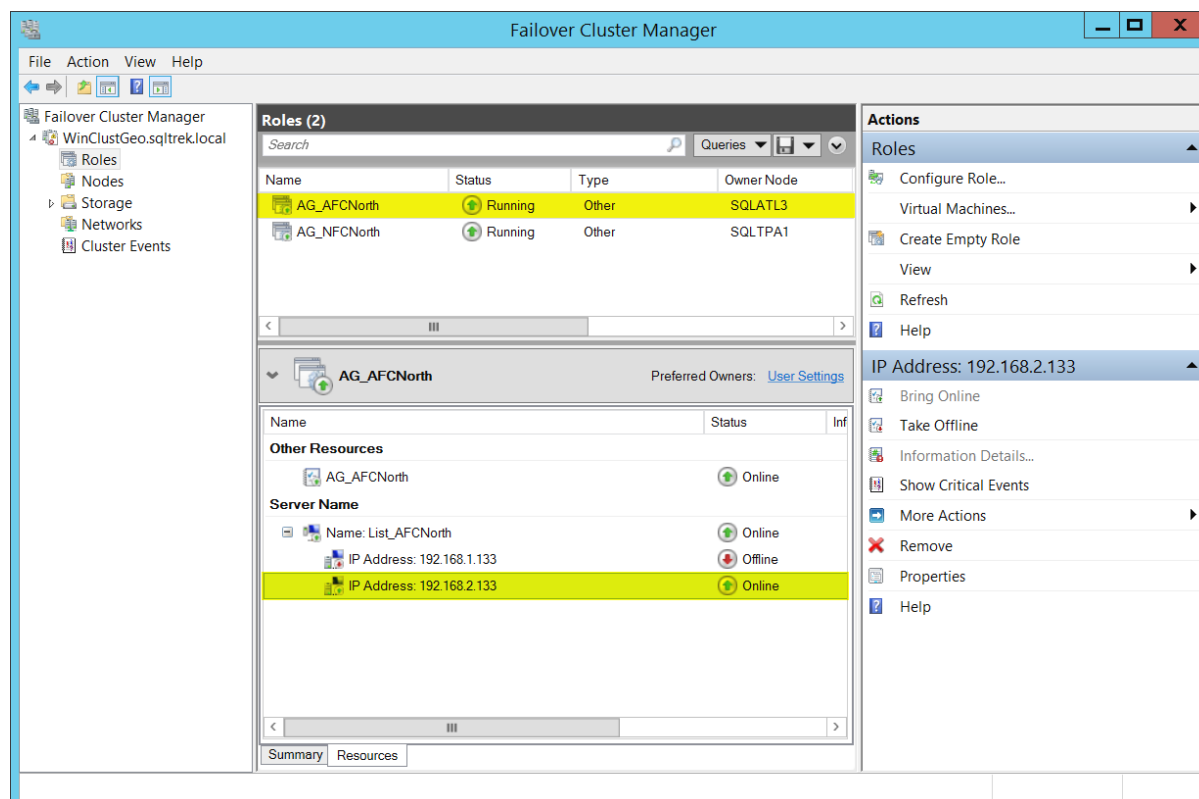
I followed the same steps to create my second AG (AG_AFCNorth) and a corresponding listener (List_AFCNorth). When am all done, this is how everything looks in my lab from my SSMS.

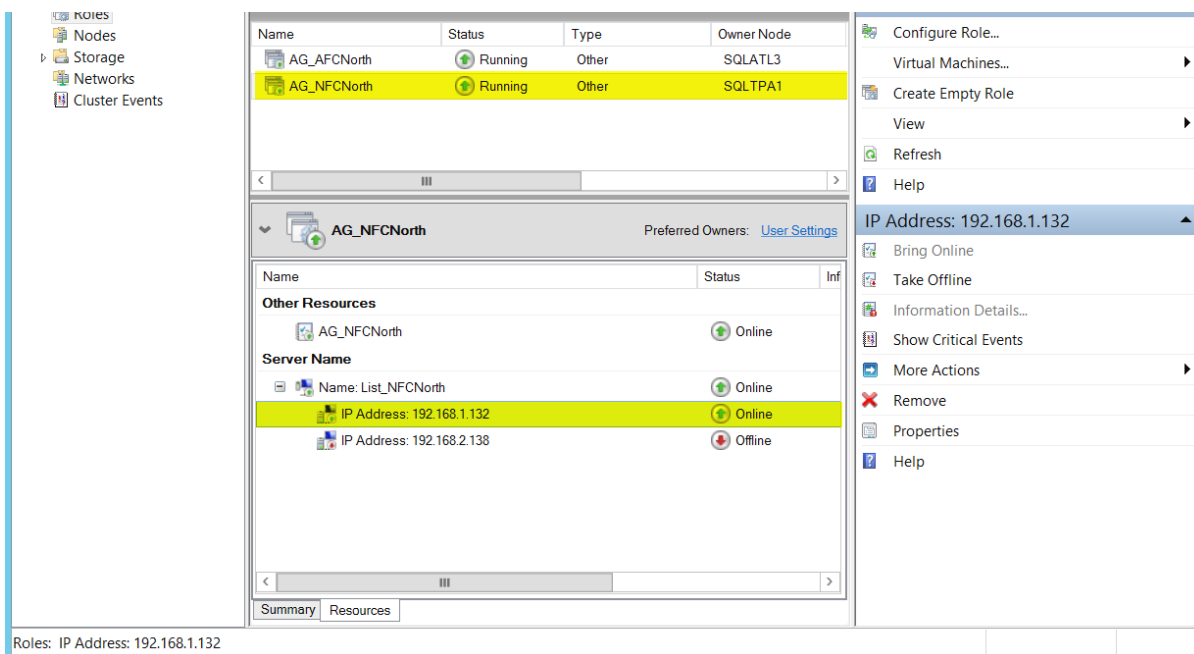




Now, Let's see how things look from Failover Cluster manager. Before going any further, let me tell you that I failed over my "AG_AFCNorth" to my DR site (192.168.2.x). I did this on purpose to show you how things will look depending on which subnet the AG is currently residing in.

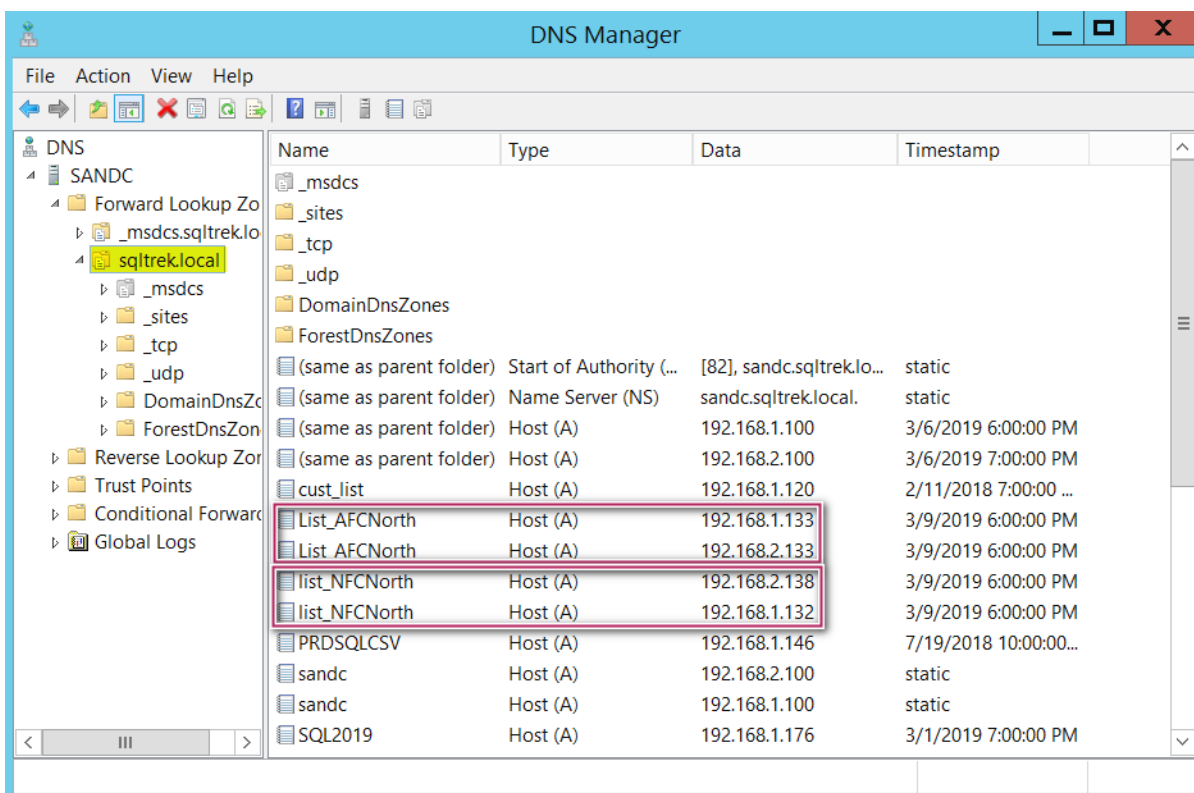
See below screenshots where under List_AFCNorth, the IP 192.168.1.x shows offline and for List_NFCNorth, the IP 192.168.2.x shows offline.





This is because at any given point of time, a listener can only be owned by one subnet. The IP will be online/offline depending on which subnet the listener is currently residing on.

Now let's see how it looks on my DNS. I opened DNS manager on my SANDC machine and expanded Forward lookup zones and went to my domain "sqltrek.local" as shown below.



As you can see, we have two A records being created for each listener, one under each subnet. This is very similar to what we would see for the virtual network name if we were to build a traditional failover cluster spanning across multiple subnets.

That is it folks! I hope this article helps you in setting up a lab simulating multiple data centers. With this setup in place, now I can experiment and learn things on how my AGs are different when spanning across multiple sites. I can reproduce issues, simulate application connectivity, practice patching, practice failover/DR testing and what not right here in my personal laptop 😊

See more

Check out SpotLight, a free, cloud based [SQL Server monitoring](#) tool to easily detect, monitor, and solve SQL Server performance problems on any device

Introducing Spotlight Cloud



Monitor your SQL Servers for FREE



Quest

Sreekanth Bandarla



Sreekanth Bandarla is a Database Administrator having about 9 years of experience supporting SQL Servers in Banking, Trading, Industrial, Automotive and Health care domains. He is a Microsoft certified professional holding active MCITP and MCSA certifications, currently exploring cloud technologies in database stack. He keeps his own [blog](#) and in his leisure time he loves playing open world games in Play station.

[View all posts by Sreekanth Bandarla](#)

Related Posts:

1. [AlwaysOn Availability Groups – Curiosities to make your job easier – Part 3](#)
2. [Windows Failover Cluster Quorum Modes in SQL Server Always On Availability Groups](#)
3. [AlwaysOn Availability Groups – How to setup AG between a clustered and standalone instance \(Part 1\)](#)
4. [AlwaysOn Availability Groups – How to setup AG between a clustered and standalone instance \(Part 3\)](#)
5. [AlwaysOn Availability Groups – How to setup AG between a clustered and standalone instance \(Part 2\)](#)

AlwaysOn Availability Groups, High Availability

4,041 Views

[Comments](#)[Community](#)[1 Login](#) ▾[Recommend](#) 4[Tweet](#)[Share](#)[Sort by Best](#) ▾

LOG IN WITH

OR SIGN UP WITH DISQUS [?](#)

Be the first to comment.

ALSO ON SQL SHACK


Creating SQL Unit Testing Utility Procedures with tSQLt

1 comment • 7 days ago

 **adeela ashraf** — A very informative article and written in a very elaborative way

SSIS OLE DB Destination Vs SQL Server Destination

1 comment • 18 days ago

 **sayed ali** — Good job Hadi, and Thank you


Three Standard SQL Unit Tests you can write against

1 comment • 3 months ago

 **Raja Imran Khan** — Very well written article. Your knowledge and application is admirable.

SQL Server in Kubernetes Cluster using KOPS

1 comment • 4 months ago

 **Sudheer** — Very nice and informative. Thanks[Subscribe](#) [Add Disqus to your site](#) [Add Disqus](#) [Add](#)