



Successfully Virtualizing Microsoft SQL Server for High Availability on Azure VMware Solutions

BEST PRACTICES GUIDE

Table of contents

Introduction	3
Target Audience	3
Architectural Guidelines and Operational Considerations for Running SQL Server on Azure VMware Solutions.....	4
Use Case Definition	4
Rightsizing	4
Requirements	5
Risks, Assumptions and Constraints	6
High-Level Architecture	6
Logical Design	7
Operational Considerations	8
Configuration Checklist for Hosting Microsoft SQL Server Workloads on AVS.....	9
Memory Allocation	9
Memory Reservation	9
CPU Allocation	11
Beware of “Limit”	11
Network Card Configuration	12
SCSI Controller Configuration	13
Disk Allocation and Configuration	15
Power Setting Configuration	16
Deploying and Preparing a VM for Windows Server Failover Clustering	19
Overview	19
Deployment Steps	19
Installing Windows Server Failover Clustering	21
Windows Server Failover Clustering Configuration	29
Installing and Configuring Microsoft SQL Server Always On Failover Clustering Instance on AVS	43
Enable (Persistent) Trace Flag 1800	45
Adjust WSFC Cluster Heartbeat Thresholds:	48
Installing and Configuring Microsoft SQL Server Always On Availability Group on AVS	50
Conclusion	69

Introduction

This document describes the steps and tasks involved in design and configuring a sample Azure VMware solution infrastructure to optimally support a virtualized production Microsoft SQL Server workload. It is intended to provide a representative list of configuration options, architectural guidelines and operational considerations based on the standard practices documented in VMware's published best practices for virtualizing Microsoft SQL Server workloads on the VMware vSphere® platform.

To provide a broader view of these options, this document describes a 3-node cluster of Microsoft SQL Server instances configured as follows:

- A 2-node failover clustering instance (FCI) configuration consisting of two Windows Server 2019 VMs hosting an instance of Microsoft SQL Server and configured for high availability.
- A 2-node always on availability group (AG) consisting of two Windows Server 2019 VMs hosting a high-availability configuration of a database.
- One of the VMs in both scenarios described above is simultaneously participating as a node in the configurations.

Always refer to the following architectural and best practices guides for the latest and most comprehensive information when designing your VMware vSphere-based hybrid cloud infrastructure. These documents (individually and collectively) will remain the authoritative sources for information related to running Microsoft SQL Server workloads on the vSphere platform and any hybrid- or public-cloud based derivative:

- [SQL Server on vSphere Best Practices Guide](#)
- [Planning Highly Available, Mission Critical SQL Server Deployments with VMware vSphere](#)
- [Considerations for running Microsoft SQL server workloads on VMware vSAN](#)
- [Architecting Microsoft SQL Server on VMware vSphere – Best Practices Guide](#)

Target Audience

This document assumes a knowledge and understanding of VMware vSphere and Microsoft SQL Server. Other than in passing, this document does not describe features, architecture, management or administration of VMware vSphere or Microsoft's Azure Platform and components. Where necessary, this document includes links to existing references and screenshots for the purpose of illustration only.

Architectural staff can use this document to gain an understanding of how the system will work as a whole as they design and implement various components. Engineers and administrators can use this document as a catalog of technical capabilities. Database administrator (DBA) staff can use this document to gain an understanding of how Microsoft SQL Server might fit into a virtual infrastructure. Management staff and process owners can use this document to help model business processes to take advantage of the savings and operational efficiencies achieved with virtualization and hybrid cloud.

Architectural Guidelines and Operational Considerations for Running SQL Server on Azure VMware Solutions

Azure VMware Solutions (AVS) allows users to create vSphere-based data centers (SDDCs) on AVS. Each deployed SDDC includes VMware ESXi™ hosts, VMware vCenter Server®, VMware vSAN™, VMware NSX® components and other software. The same HTML5-based vSphere client is used to manage a SDDC once deployed. This approach allows seamless migration of demanding application workloads without the need to adopt new toolsets or refactor application components.

The planning and designing phase is very important to ensure that migrations of mission-critical application workloads to Azure VMware Solution complete without negatively impacting application SLAs or affecting the performance, availability, manageability and scalability of the workloads. This document summarizes architectural guidelines which will help enterprises when planning a migration.

Use Case Definition

The following use cases have been identified as the most frequently employed for SQL Server workloads on Azure VMware Solutions:

- Data center extension, like cloud-bursting and test/dev
- Workloads health and performance
- Workload high availability
- Data center evacuation
- Disaster recovery with site resilience
- Application migration and modernization

Each particular use case, or combination of use cases, influences general solution design and necessitates appropriate requirements gathering. This document focuses on use cases specific to workloads health, performance and high availability. We address the health and performance aspects of our objectives by providing a checklist of the common configuration options and recommendations required to avoid common performance bottlenecks in a typical virtualized Microsoft SQL Server infrastructure. We believe that, instead of repeating the usual exercise of providing performance metrics (which we continue to do in separate documents), it is more beneficial that we provide you with actionable configuration tips and recommendation to make your virtualization experience enjoyable.

Rightsizing

Before considering where to place SQL Server workloads on a cloud platform, ensure that your virtual machine is right-sized. A workload's performance profile should be collected over a sufficient period of time to reflect applications steady state, peaks and spikes in resource utilizations. While defining the required time range to collect time series data, consult with DBAs and application owners to understand the workload profile. At least a full month of "non-rolled up" time series data is recommended prior to executing the performance analysis.

Utilizing the [VMware vRealize® Suite Management Pack for Microsoft SQL Server \(vmware.com\)](https://www.vmware.com/resources/compatibility/path1) is proven to be very helpful in this preparation phase. While analyzing captured data, make sure your rightsizing approach has been agreed upon by administrators, application owners and business

owners, and that it encompasses both spikes (i.e., high performance) and average utilization (i.e., higher density).

Microsoft's [AVS Assessment with Azure Migrate](#) is also a very useful tool to employ at this phase of your planning, sizing and assessment.

The following should be considered while sizing SQL Server workloads:

- For CPU and memory resources allocation, check [Common questions about Azure VMware Solution](#) to become familiar with the host's compute maximums and to verify the workload will fit and not overcommit host resources.
- Account for differences in physical CPU architectures between your current environment and the [host instances](#) used in Azure VMware Solution.
- Always size the CPU resource based on the actual workload, as vCPU can be easily added later.
 - Unnecessarily over-allocating compute resources to a VM can have an adverse performance impact on the workloads.
 - Hot-adding CPUs to a VM running Microsoft SQL Server instances is not recommended.
- 
 - **Enabling vCPU HotAdd creates fake NUMA nodes on Windows (83980) (vmware.com)**
 - [CPU HotAdd for Windows VMs: How BADLY Do You Want It? - Virtualize Applications \(vmware.com\)](#)
- Note that the storage layer in Azure VMware Solution is provided by hyperconverged infrastructure (HCI) solutions VMware vSAN. Adding storage will require the addition of compute resources (i.e., hosts) as well. You can use Azure storage services (e.g., storage accounts, table storage, blob storage) to augment storage for workloads running in your Azure VMware Solution environment.

Requirements

A crucial part of a successful migration is collecting business and technical requirements, allowing you to properly design a cloud platform. For guidance, review [Preparing for Azure VMware Solutions](#) before beginning your requirements gathering.

Business requirements are an important part of the requirements gathering process. Input examples include:

- RTO/RPO targets
- Business SLAs for the applications workloads based on SQL Server databases
- [Licensing considerations](#)
- [Azure Hybrid Benefit](#)
- Security and data-management considerations

Technical requirements will directly influence logical design and should be collected and validated with care. Pay special attention to the following bullet points:

- Performance requirements of the workload (e.g., transactions-per-second, number of user connections, expected future workloads changes)
- Capacity requirements (e.g., future growth, other projects to be served)
- Scalability requirements (e.g., method for increasing capacity of a SDDC, scale-out versus scale-in approach)
- Application requirements (e.g., type of workloads [e.g., OLTP/data warehouse], dependencies between on-premises components and network flow between them)
- Manageability requirements (e.g., providing access to a SDDC to appropriate user groups, reconfiguring monitoring tools, backup solution in use, modifying scripting, vRealize Operations workflows)
- Availability requirements (e.g., SQL Server high-availability solutions in use, DRS groups, host-isolation response, number of availability zones required)

NOTE: Manageability and Availability are two important considerations we will discuss in greater details in this document. This is because, due to its hosted and managed-services nature, AVS provides restricted administrative access to the underlying infrastructure. This level of access limits certain configurations that require assisted support and which cannot ordinarily be performed by the customer.

Risks, Assumptions and Constraints

Ensure that risks, assumptions and constraints are identified and documented and that the risk-mitigation plan has been agreed by all groups involved.

- An example of an assumption: Do you have a *software assurance* agreement for Microsoft SQL Server licensing?
- An example of a constraint: available network bandwidth between on-premises and SDDC (e.g., is ExpressRoute available?)
 - The scenario described in this document assumes that there is connectivity between the on-premises VMware vSphere infrastructure and the Azure VMware Solution environment
 - The examples shown in this document present a stretched-network scenario in which one network subnet is in use across both environments.
- An example of a risk: different CPU generations between on-premises ESXi hosts and hosts in an SDDC (can you enable EVC to ensure migrating running workloads from on-premises to AVS without impacting service?)

High-Level Architecture

A high-level solution architecture should include enough information to capture the on-premises environment hosting the SQL Server workloads and planned SDDC(s) with multiple availability zones (AZ) while also providing enough details to work on logical design.

Deploying and operating an Azure VMware Solution SDDC involves several simplified processes which are described in the following quick steps:

- [Planning the Deployment](#)
- [Deploying the Azure VMware Solution](#)

- [Connecting to on-premises](#)
- [Deploying and Configuring VMware HCX](#)

Logical Design

A logical design describes all technical decisions made and addresses identified technical requirements while minimizing risk. The level of detail included should be sufficient to create an implementation guide for the solution. While specifics of each logical design are unique, it's important to ensure all technical prerequisites are met. The following prerequisites have been identified as crucial to the successful migration of SQL Server workloads to Azure VMware Solutions:

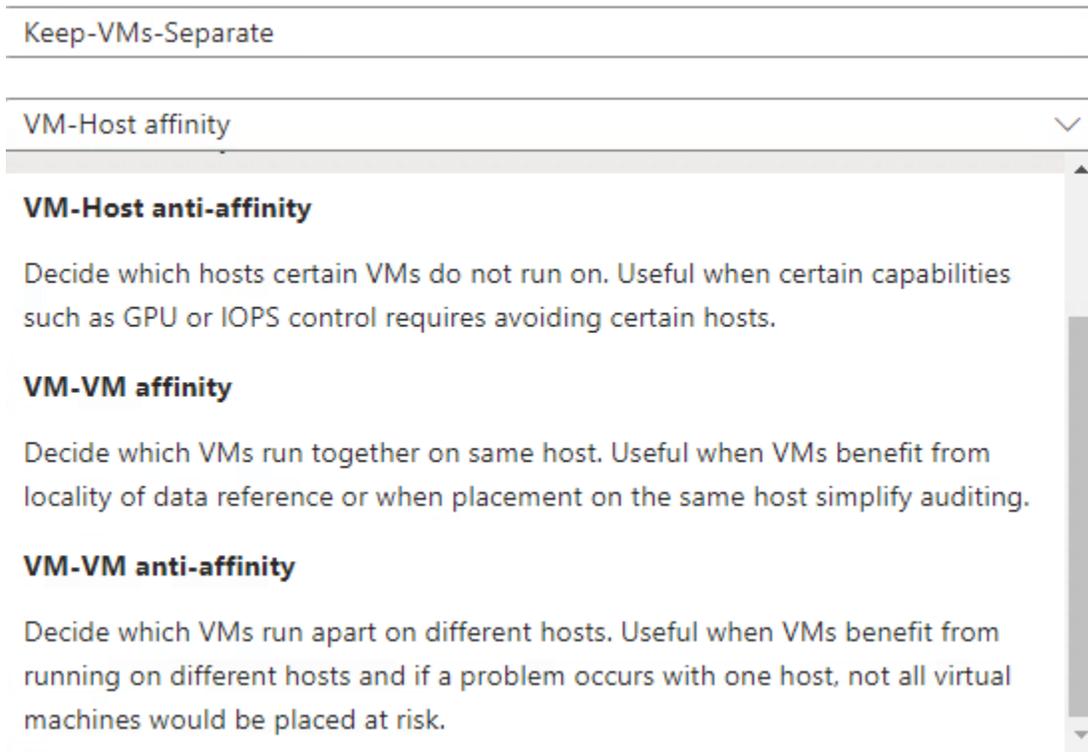
- For on-premises-located VMs, ESXi hosts, and/or vSphere clusters hosting SQL server workloads, check and document all advanced settings configured. Ensure corresponding options are available in Azure VMware Solutions. For example, DRS anti-affinity groups and rules must be re-created in an SDDC as they cannot be migrated.

IMPORTANT:

- a. As at the time of this writing, VM-VM anti-affinity rule is not supported in AVS
- b. As at the time of this writing, creating the necessary anti-affinity rules required to separate clustered VMs in AVS requires that you open a support request with the Microsoft AVS support team. This task is assumed to have been completed and is, therefore, not covered in this document.

UPDATE: Microsoft has added support for creating DRS Rules into Azure VMware Solution (AVS). Using the “Placement Policy” option available in the VPC, a Customer can create one of four DRS Rules:

- VM-Host Affinity
- VM-Host Anti-Affinity
- VM-VM Affinity
- VM-VM Anti-Affinity



- Conduct an assessment of your current infrastructure using [AVS Assessment with Azure Migrate](#) to ensure that all requirements are met.
- Determine whether or not [Azure Migrate](#) is appropriate as the migration tool for existing moving workloads to AVS.
- Where applicable, [Deploy and Configure VMware HCX for Azure VMware Solution](#) to enable you to move existing workloads from your existing VMware infrastructure to AVS.
 - a. Hybrid coexistence using L2 VPN between the on-premises and the SDDC is currently not supported for AVS.
- Configure [Hybrid Linked Mode](#) to allow managing both on-premises and public SDDCs within a single vSphere Client interface. Verify that all required user accounts are added to the appropriate group.

Operational Considerations

Post-implementation maintenance and operation guidelines are a key component of any well-prepared infrastructure architecture. While incorporating Azure VMware Solution SDDCs in an existing infrastructure, it's critical that *Day 2* operational routines are updated accordingly, including:

- Backup configurations
- Monitoring configurations
- Operational documentations

If [vRealize Operations](#) is used to monitor the environment, confirm that all SDDCs are added to vRealize Operations-managed resources and configured using vCenter Adapter with the **Public Cloud** option.

Configuration Checklist for Hosting Microsoft SQL Server Workloads on AVS

The purpose of this section is to document the specific configuration options for successfully and optimally hosting Microsoft SQL Server instances on the Azure VMware Solution. These are explained in detail in the *SQL Server on vSphere Virtualization Best Practices Guide* mentioned at the beginning of this document. We will therefore not discuss this in great detail.

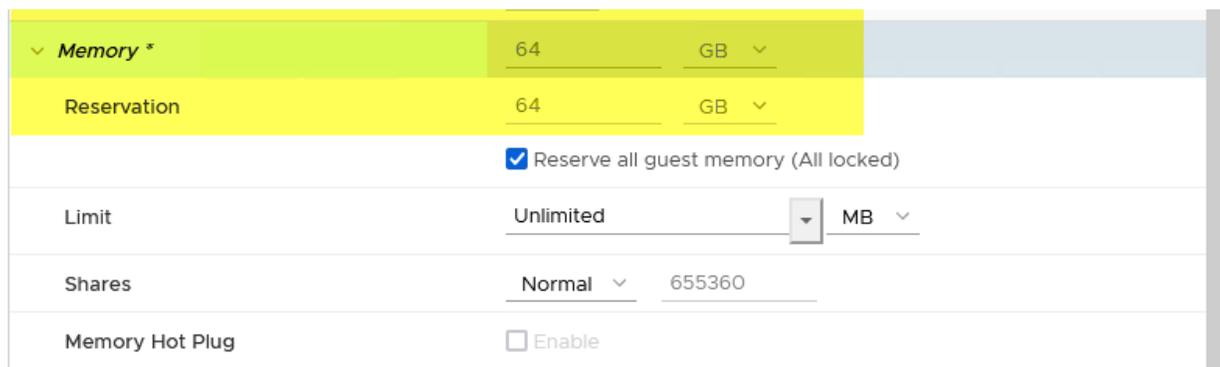
Memory Allocation

How much memory to allocate to a VM on the vSphere platform is strictly a function of the actual utilization threshold, as determined through administrative due diligence. While Microsoft SQL Server Administrators typically prefer to allocate the maximum possible RAM to their Microsoft SQL Server workloads (in anticipation of peak or worst-case consumption scenarios), it is very important to resist this practice. Virtualization imposes certain optimization and functionality considerations that may (counter-intuitively) induce performance penalties for an over-sized VM if over-allocation of resources results in sustained resource wastage. Right-sizing VMs is a sound administrative practice and we encourage applications and virtualization team members to engage in the necessary due diligence to avoid allocating compute resources to a VM on a whim. Memory should be allocated to a Microsoft SQL Server VM strictly based on empirical and historical usage trends.

Memory Hot-Add: Memory hot-add is a feature which allows Administrators to allocate *just-enough* memory to a VM (without incurring over-allocation penalties) and to adjust this allocation upwards when load increases without having to restart the VM or interrupt running tasks and processes. Modern Windows Server OS, Microsoft SQL Server and VMware vSphere support this feature, so we recommend that, when in doubt, it should be enabled as a compromise, rather than the alternative option of anticipatory over-allocation.

Memory Reservation

One of the tenets and benefits of virtualization is the cost-beneficial opportunity to utilize physical compute resources more densely and fully than is otherwise possible in a physical server environment. At a very high level, the cost benefits come from the ability to pool these resources together and share them more broadly among more OS instances on one physical server.



Conversely, one of the challenges of virtualization is the likelihood that, in attempting to share pooled resources among as many OS instances as possible to achieve this density, Administrators run the risk of unintentionally creating the opposite effect, at least for some of the OS instances sharing the pooled resources.

As an example, imagine an ESXi host with 128GB of RAM and 10 VMs, each with 20GB RAM allocated. In normal operating conditions, it is unlikely that all 10 VMs will need their 20GB allocation at the same time. In that scenario, the VMs can all happily coexist without inducing performance bottleneck.

What if all VMs were to suddenly need their full allocation, though? Obviously, we are 72GB of memory short. Since virtualization itself does not manufacture non-existent compute resources, the hypervisor has to resort to various mechanisms (e.g., swapping, ballooning) to try to satisfy the VM's demands. In the process, some (if not all) of the VMs and the applications they host will experience performance degradation, leading to negative impact on business processes and overall user experience.

Here is where the Memory Reservation feature of VMware vSphere can come into play. In most network environments, workloads are not created equal. Some servers (physical or VMs) host applications which are more critical to business operations and revenue than others. These are called the Business (or Mission) Critical Applications. Supposing that five of the VMs in our example above are considered "*Business Critical*" (aka Tier 1) and the other five are considered not so critical. Memory Reservation would allow us to give each of the five Tier-1 VMs all their allocated memory even as the other five gasp for air.

If you are mixing VMs of different critical importance in your AVS environment, *and* you anticipate over-allocating available physical memory resources in the ESXi cluster, it is recommended that you reserve ALL the memories you are allocating to the VMs hosting the most critical application in the mix. By doing this, you ensure that your important VMs will always be entitled (and have access) to the allocated memory resources. As a little bonus, because vSphere won't create a swap file for VMs whose memory have been fully reserved, you also save on storage.

CPU Allocation

Starting with vSphere 6.5, VMware has changed its prescriptive guidance for allocating compute resources to a VM. Please see [Virtual Machine vCPU and vNUMA Rightsizing - Guidelines - VROOM! Performance Blog \(vmware.com\)](#) for all the background and pertinent information around this.

In summary, for wide VMs (i.e., VMs which require more vCPU counts than are available in one physical socket in the ESXi host), the recommendation to *leave cores-per-socket at 1* no longer applies. The simplified rule of thumb is to allocate vCPUs to a VM in a way that mirrors the physical NUMA topology of the ESXi host.

CPU Reservation: This is not recommended, so we will not spend much time discussing this.

CPU Hot-Add: While Microsoft SQL Server is one of the few modern applications which can dynamically adapt to (and utilize) hot-added CPUs, VMware does not recommend enabling hot-add CPUs for VMs running any version of the Windows Server Operating System at this time. Please see the following for more detailed discussion:

- [CPU HotAdd for Windows VMs: How BADLY Do You Want It? - Virtualize Applications \(vmware.com\)](#)
- [Enabling vCPU HotAdd creates fake NUMA nodes on Windows \(83980\) \(vmware.com\)](#)
- [CPU Hot Add Performance in vSphere 6.7 - VROOM! Performance Blog \(vmware.com\)](#)

The screenshot shows the 'Edit Settings' window for a VM named 'AVS-VMW-SQL01'. The 'Virtual Hardware' tab is selected. The 'CPU' settings are highlighted in yellow. The 'CPU' dropdown is set to 24. Below it, 'Cores per Socket' is set to 12, and 'Sockets' is set to 2. The 'CPU Hot Plug' checkbox is unchecked, with the text 'Enable CPU Hot Add' next to it. The 'Reservation' is set to 0 MHz, and the 'Limit' is set to Unlimited MHz.

Setting	Value
CPU *	24
Cores per Socket	12
Sockets	2
CPU Hot Plug	<input type="checkbox"/> Enable CPU Hot Add
Reservation	0 MHz
Limit	Unlimited MHz

Beware of “Limit”

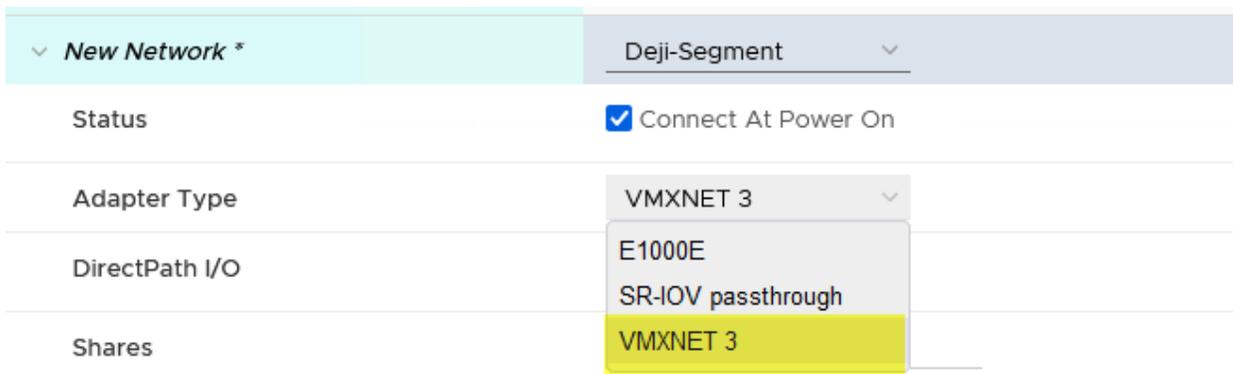
As you have likely noticed in the previous two images above, vSphere provides Administrators an easy option to prevent unnecessary resource wastage by imposing a limit on the Memory and CPU resources allocated to a VM. VMware strongly recommends that Administrators leave this option unconfigured in both cases. It is better to right-size and allocate as much resources to a Microsoft SQL

Server VM as it requires rather than over-allocating resources and then consciously imposing a restriction on the VM’s access to the allocated resources.

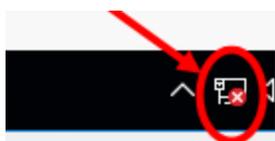
One of the most challenging problems with configuring Limits on VMs is that such configuration is easily forgotten and then missed during a performance troubleshooting exercise. The standard administrative response when a VM is perceived to be resource-constrained is to increase its allocation. Sadly, however much one increases the allocation, the VM will never be able to utilize anything beyond what has been set in **Limit**.

Network Card Configuration

When you create a Windows VM on vSphere, the default vNIC type assigned to the VM is the E1000E vNIC. This is not a suitable vNIC type a Windows VM running Microsoft SQL Server workloads. VMware highly recommends that you always assign the VMXNet3 virtual network card type to any Windows VMs which have more than one vCPU allocated. This is primarily because the VMXNet3 vNIC type is the only vNIC type in vSphere which supports the Windows Receive Side Scaling (RSS) feature. No other vNIC has this capability.

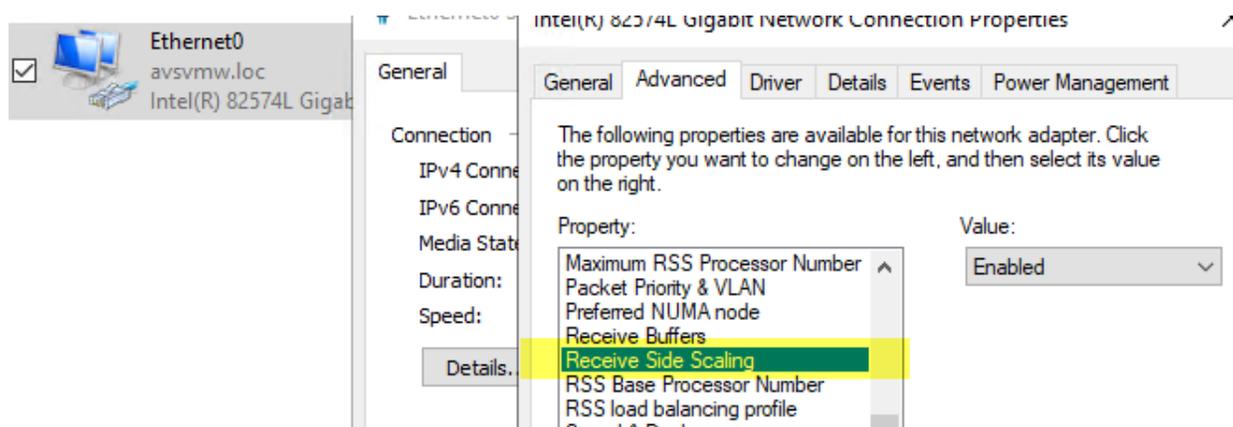


So, why would you care about RSS for your MS SQL Server VMs? Here is why: [Introduction to Receive Side Scaling - Windows drivers | Microsoft Docs](#). Briefly, RSS is the Windows feature which allows Windows to distribute the handling of network packets/requests/interrupts over multiple CPUs in the OS. Without this feature, only one CPU (out of however many you have allocated to Windows) will be doing the heavy lifting. On a busy Windows OS instance, this invariably leads to packet drops, performance degradation and (eventually) instability.



Choosing VMXNet3 vNIC for the VM is just one step. Because VMXNet3 is not native to Windows, it is unusable until you have installed the VMTools on the VM. It is very important that you always keep VMTools up-to-date on your VMs because this is the mechanism through which VMware delivers updated drivers and fixes to the VM and its components.

By default, RSS is not enabled on the vNIC inside Windows, so you must make enabling it a core part of your standard administrative or deployment procedures (better yet, enable it in your Windows VM Templates).



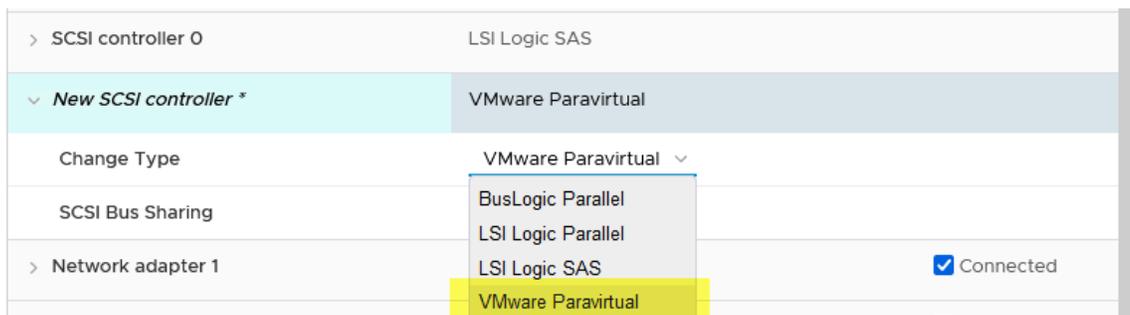
NOTE: Please disregard any counter recommendation you may come across. If it's a Windows VM, and it has multiple processors and it's likely to generate, receive, or transmit network traffic, VMXNet3 is your only choice for virtual network card.

SCSI Controller Configuration

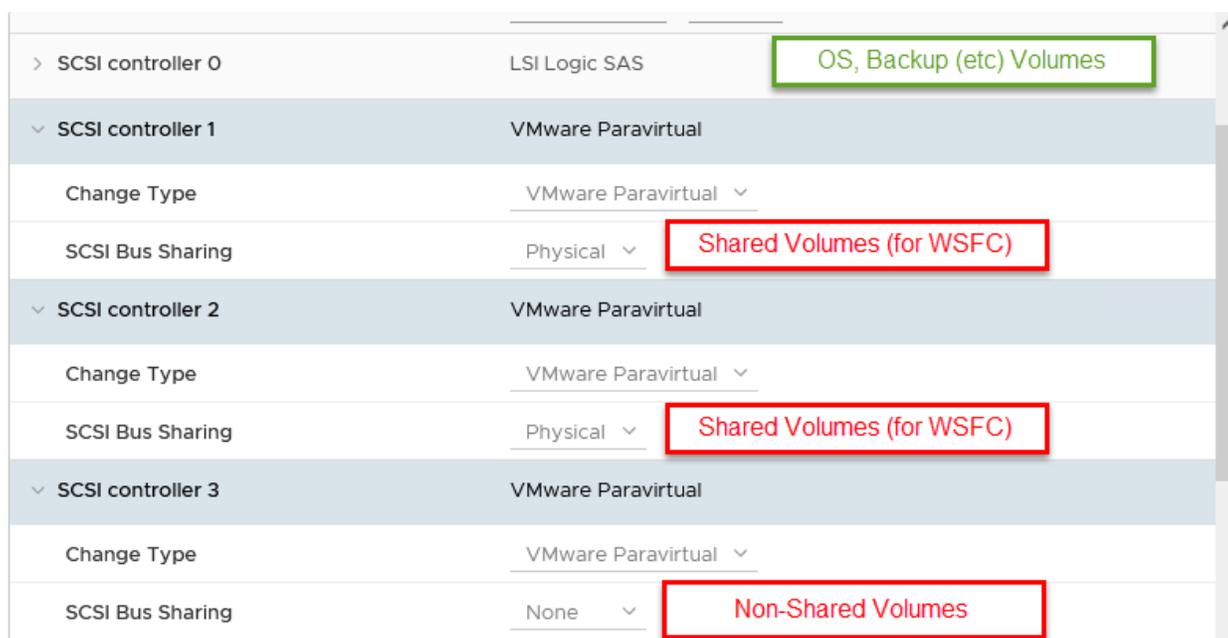
The default virtual SCSI controller type assigned to a Windows VM on vSphere is the LSI Logic SAS. This choice is driven largely by the need to simplify the VM deployment process, allowing the Administrator to install Windows as quickly and seamlessly as possible.

However, the LSI Logic SAS SCSI controller is inadequate for the data throughput and disk IO patterns of a typical Microsoft SQL Server workload, which expects an optimal and fast storage IO send/receive rate for its queries and transactions. The LSI Logic SAS vSCSI controller is limited to 32 IO queue depth, which is insufficient for most modern Microsoft SQL Server workloads.

The VMware Paravirtualized Controller (PVSCSI), on the other hand, has a configurable queue depth of 64, which can be increased to 254 (per device) and 1024 (per Adapter). In addition, PVSCSI has a lighter CPU footprint, which makes it even that much more performant than the default LSI Logic SAS.



VMware highly recommends PVSCSI for all Data, Transaction logs and TempDB disks of a Microsoft SQL Server VM. If you have multiple disks (VMDKs) for these disk types, VMware recommends that customers add additional PVSCSI controllers and distribute the disks as evenly as possible over all the controllers.

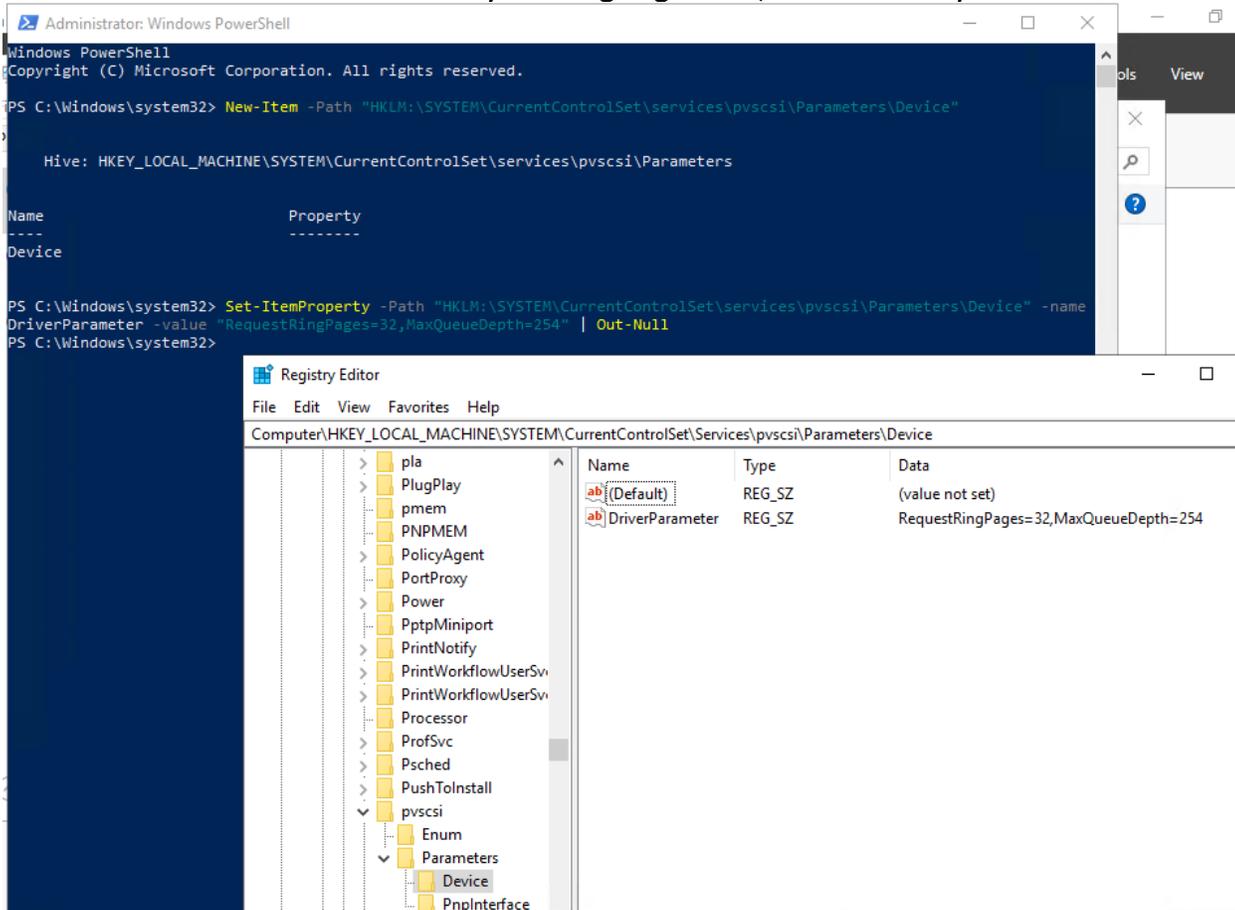


As with the VMXNet3 vNIC, PVSCSI is not native to Windows and is, therefore, unusable until you have installed VMTools. After installing the VMTools on a Windows VM which has PVSCSI controllers attached, a new Windows Registry key becomes available and configurable. This key is where the controller can be configured to support larger queue depth.

It is highly recommended that customers configure this option in order to increase IO performance for the workloads. The following two-line PowerShell command can be used to accomplish this task:

New-Item -Path "HKLM:\SYSTEM\CurrentControlSet\services\pvscsi\Parameters\Device"

Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\services\pvscsi\Parameters\Device" -name DriverParameter -value "RequestRingPages=32,MaxQueueDepth=254" | Out-Null



Disk Allocation and Configuration

VMware vSphere allows you to create VMDKs as large as 62TB each. While this is generally a good thing, Administrators should resist the temptation to misuse this capability. As with the SCSI controllers, VMDKs have queue depth limits, too. See [Large-scale workloads with intensive I/O patterns might require queue depths significantly greater than Paravirtual SCSI default values \(2053145\) \(vmware.com\)](#) for additional information.

A device (in this case, a VMDK) has a queue depth of 64. The queue depth controls the maximum parallel IO that could be passed through the device at one time. Once this maximum is reached, IOs begin to queue up. If you create a large (62TB) VMDK, slice it up into multiple volumes inside the Guest OS, and assign these to, say, a few Transaction Logs, TempDB and Data disks, all of the IOs generated by these different disks will have to pass through the one VMDK, increasing your chances of reaching

the 64 queue depth threshold very frequently and quickly. The end result is storage IO performance degradation.

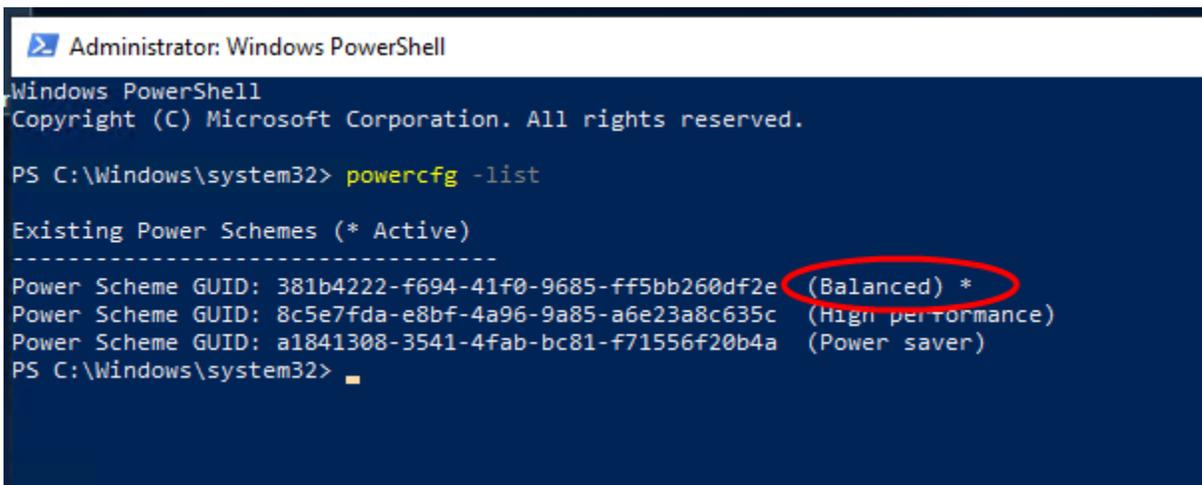
For performance and availability reasons, VMware recommends that Administrators consciously avoid putting too many high-transactional volumes or disks in one VMDK.

What about Eager-Zero Thick disk provisioning? This is an artefact that is not relevant for our purposes.

We have one more important performance-improvement related disk tuning task to check off, but since it is application-specific, [we will discuss it at a later point in this document](#).

Power Setting Configuration

Most modern hardware and Operating Systems default to power conservation in consideration of the impact of power consumption on the environment. Unfortunately, conserving power leads to CPU throttling, which leads to severe negative impact on application performance and throughput. In Windows, for example, the default Power Scheme is **Balanced**. Balanced Power Scheme results in Windows putting some CPUs to sleep when it determines that they are not in active use, and then waking them up when it deems it necessary. This behavior invariably results in suboptimal performance for Microsoft SQL Server queries and tasks.



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> powercfg -list

Existing Power Schemes (* Active)
-----
Power Scheme GUID: 381b4222-f694-41f0-9685-ff5bb260df2e (Balanced) *
Power Scheme GUID: 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c (High performance)
Power Scheme GUID: a1841308-3541-4fab-bc81-f71556f20b4a (Power saver)
PS C:\Windows\system32>
```

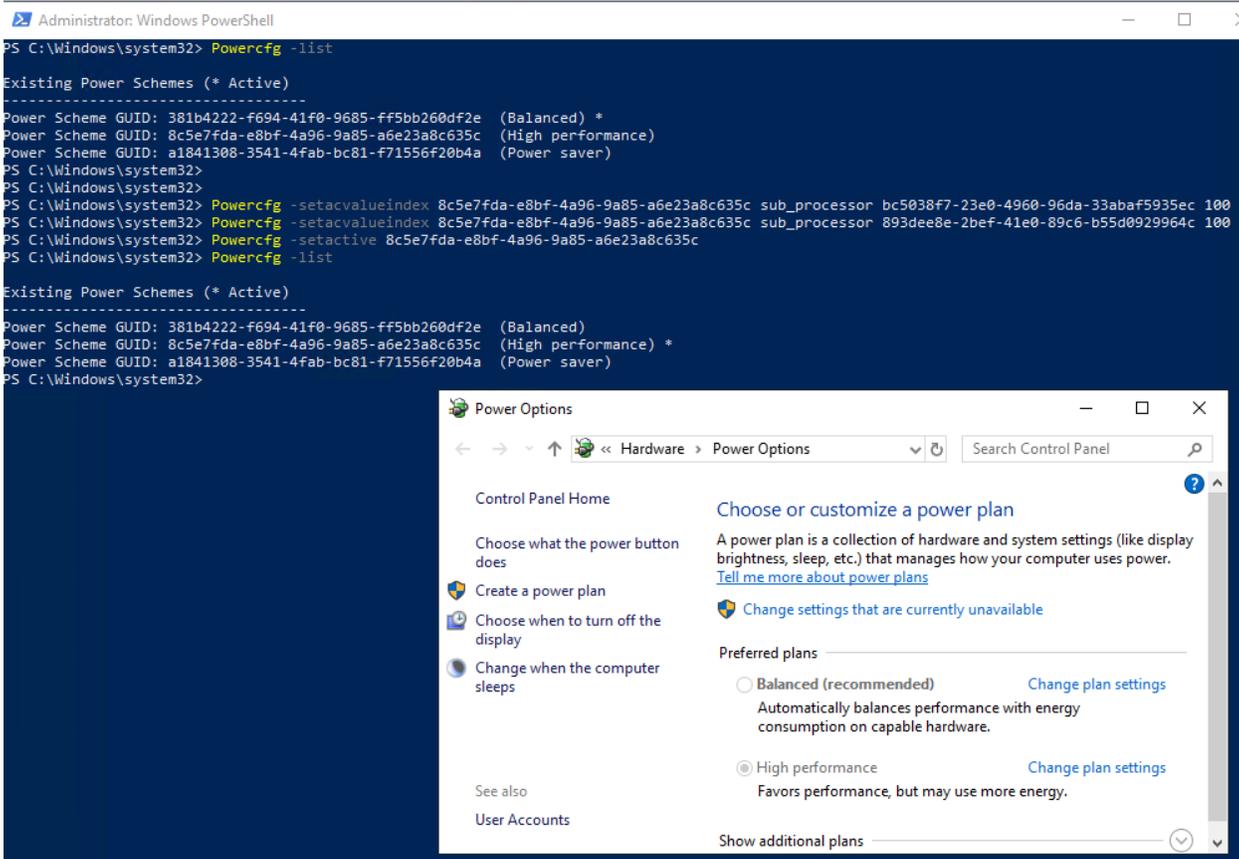
VMware recommends that Administrators change the default **Balanced** power setting to **High Performance** as a standard administrative procedure for all VMs hosting Microsoft SQL Server workloads.

The following PowerShell command will toggle Windows Power cheme from **Balanced** to **High Performance**:

```
Powercfg -setacvalueindex 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c sub_processor bc5038f7-23e0-4960-96da-33abaf5935ec 100
```

```
Powercfg -setacvalueindex 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c sub_processor 893dee8e-2bef-41e0-89c6-b55d0929964c 100
```

```
Powercfg -setactive 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c
```



This concludes the high-level performance tuning checklist required to get our VMs ready for an optimal and satisfactory virtualization experience on the Azure VMware Solution platform.

We will now proceed through setting up and configuring a representative use-case scenario, showing how to successfully deploy and configure Microsoft SQL Server workloads to satisfy high-availability requirements.

The configuration shown in the following example is highly simplified. This is intentionally so because we want to focus on the specific considerations for achieving our objectives solely from a virtualization perspective as opposed to offering a tutorial on how to install, configure or cluster Microsoft SQL Server. We assume that you are already familiar with most of the tasks involved in installing, running and operating Microsoft SQL Server.

For specific Application-level performance tuning option, please see [SQL Server Configuration here](#) and in Section 4.3 (SQL Server Configuration) of [SQL-Server-on-VMware-Best-Practices](#).

Deploying and Preparing a VM for Windows Server Failover Clustering

Overview

As described earlier, we will use three VMs in our example. They are named:

- AVS-VMW-SQL01
- AVS-VMW-SQL02
- AVS-VMW-SQL03

Two of the VMs (AVS-VMW-SQL01 and AVS-VMW-SQL02) will host a clustered instance of Microsoft SQL Server in Always On failover clustering instance (FCI) mode.

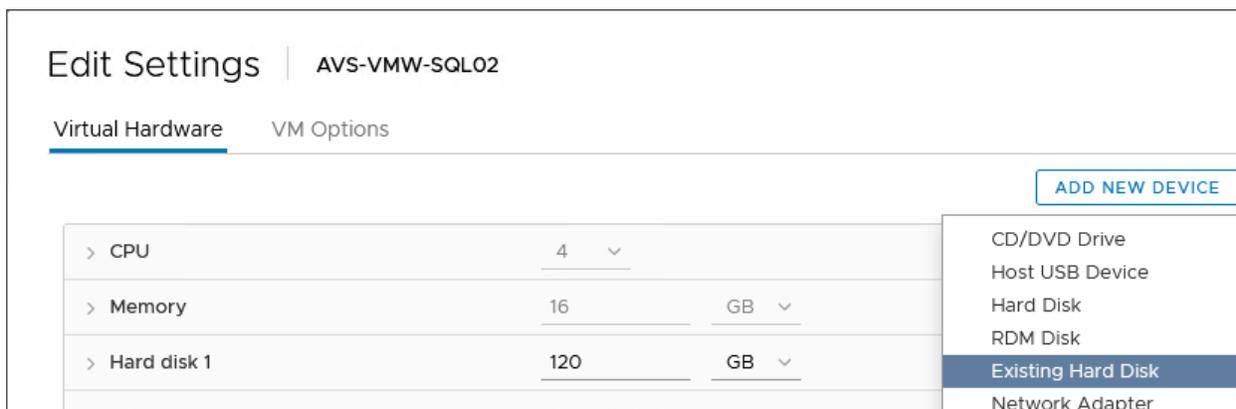
We will also create a Database on this Instance (AVSVMW-DemoAGDB) and configure it for high availability in Always On Availability Group (AG) mode.

AVS-VMW-SQL03 will be configured to host an asynchronous replica of AVSVMW-DemoAGDB.

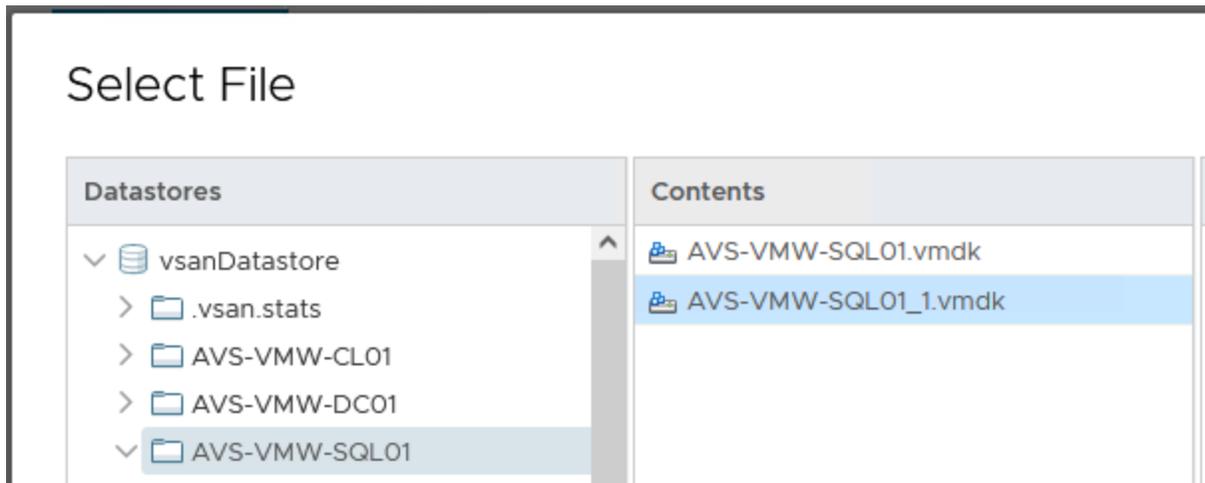
We'll henceforth refer to these VMs as SQL01, SQL02, and SQL03.

Deployment Steps

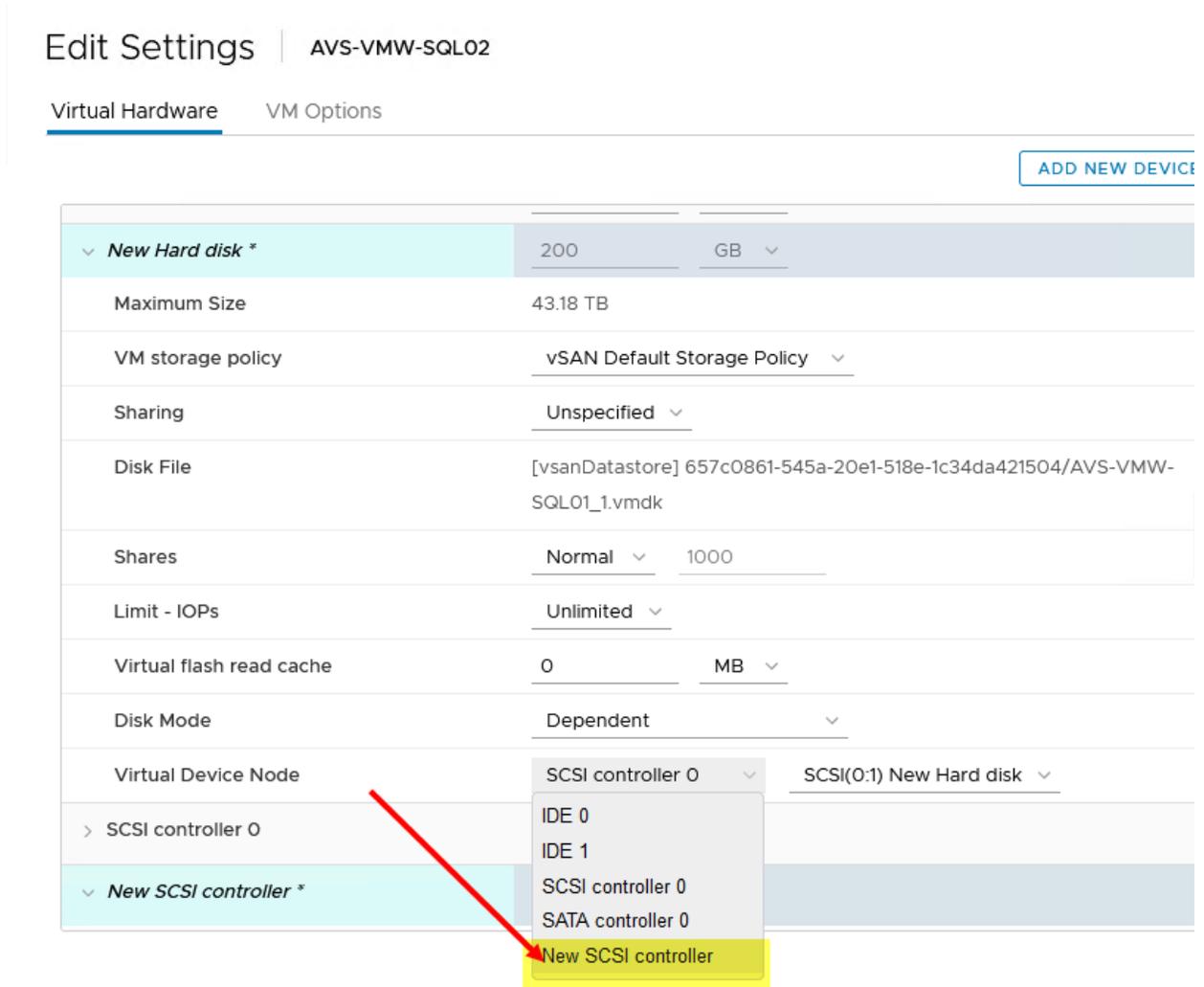
- We create the VMs, following the configuration options described earlier in this document.
- We create one or more VMDKs on SQL01 which will be used as shared disks for our FCI. In our simplified example, we're using only one shared VMDK.
- We attach this VMDK to Bus 0 on PVSCSI Controller 1 (SCSI1:0). Remember that the SCSI bus-sharing mode has to be set to **Physical** when used for shared disk.
- We are done with SQL01 now, so we move on to SQL02
- On SQL02, we add this VMDK we just configured on SQL01 to SQL02. We do this by selecting the **Existing Hard Disk** option in the **Add New Device** menu.



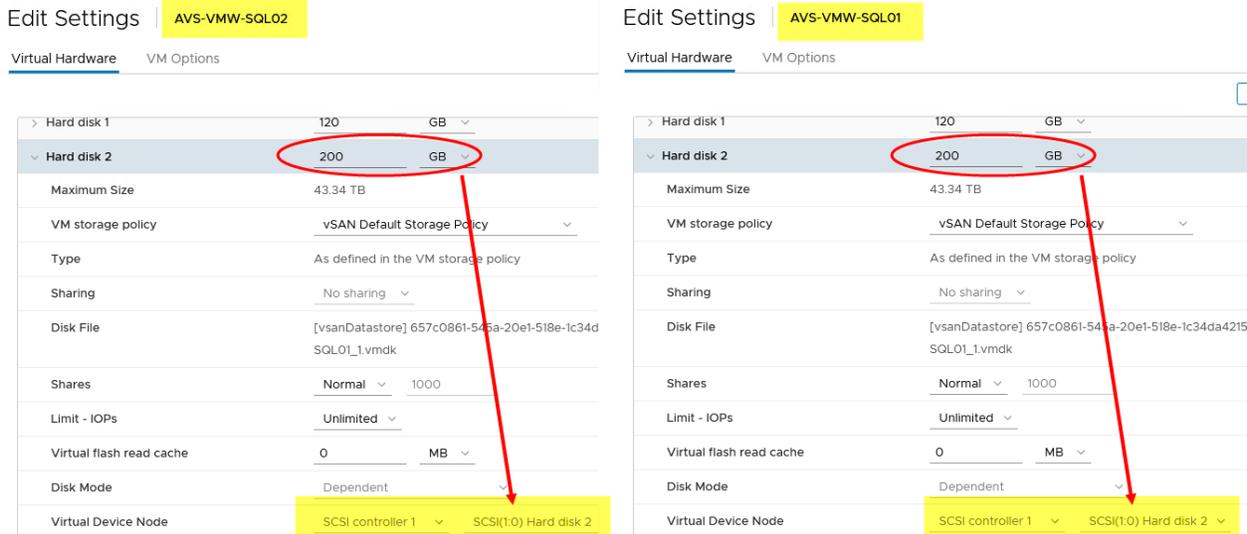
- We browse to where we have stored the shared VMDK and select it.



- We create a new PVSCSI controller and attach the disk to it.



- We verify that the disk and SCSI controllers are configured identically on both VMs.



- The End.

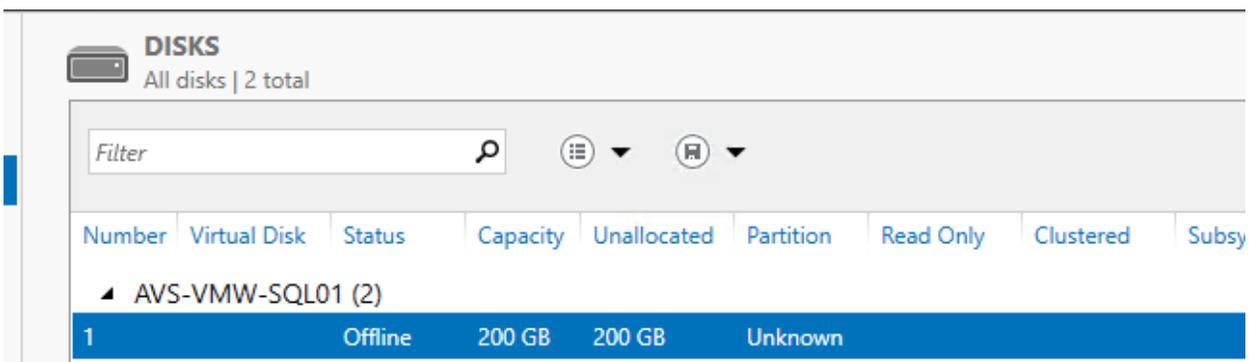
Really. That is all we need to do on the vSphere side to assign a shared disk to VMs for use in FCI configuration.

Now, we power on the VMs and let them boot into Windows.

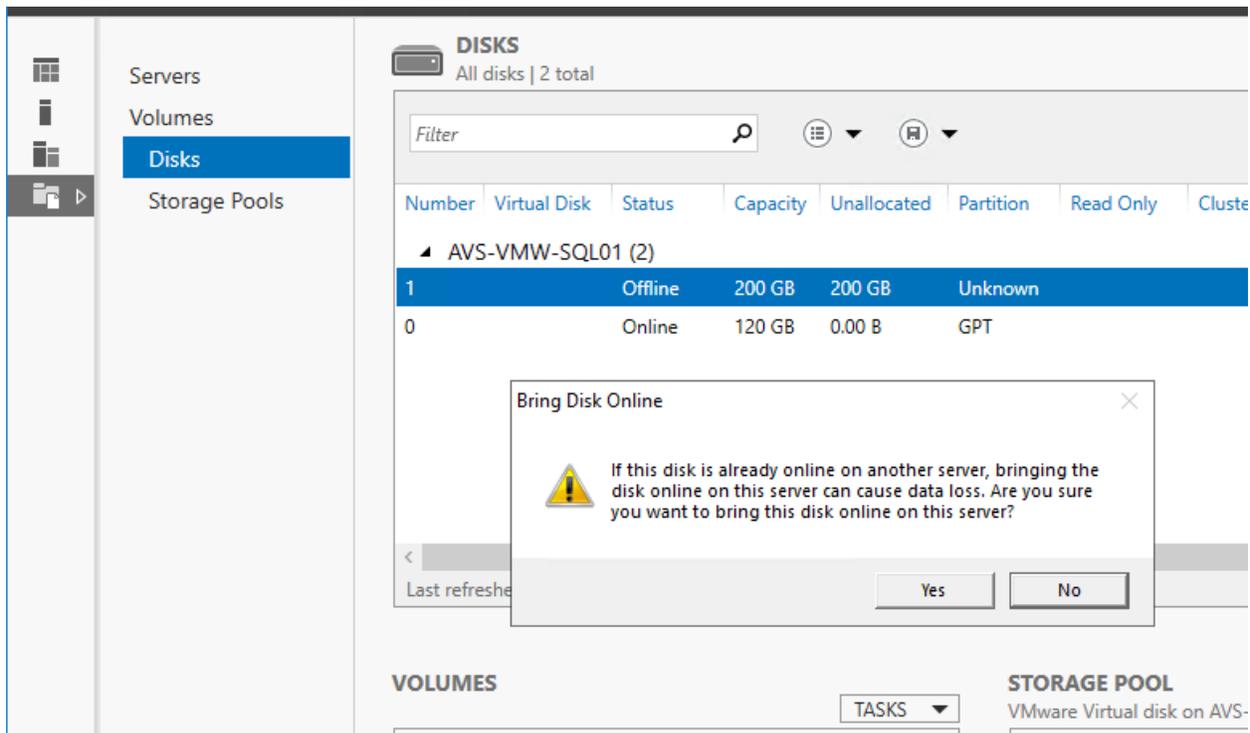
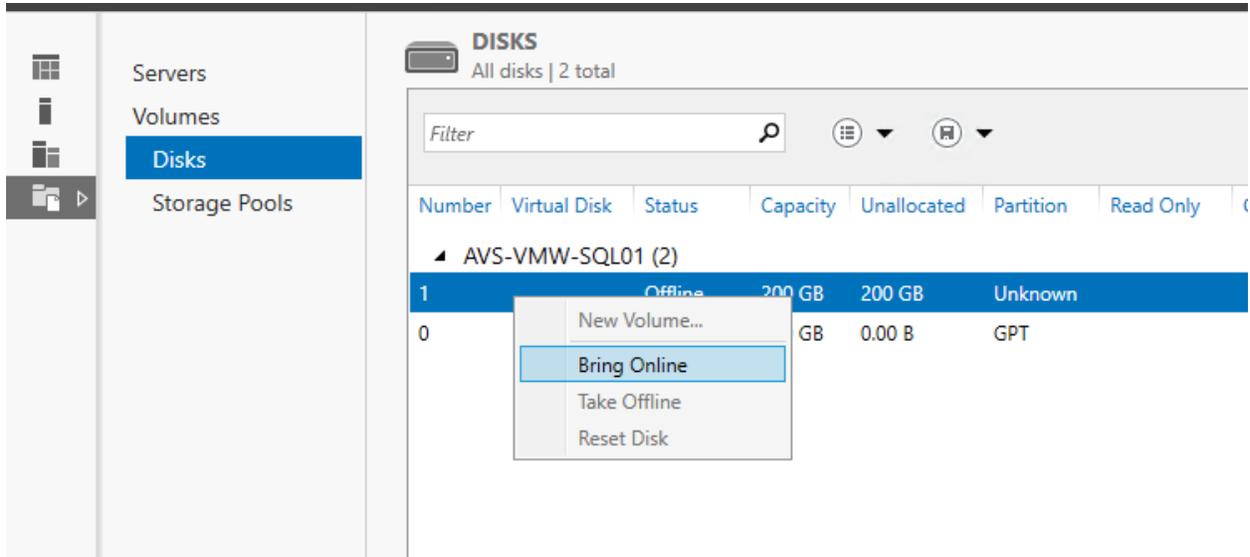
Installing Windows Server Failover Clustering

Installing WSFC on VMs in AVS with vSphere is not in any way different from doing so on physical servers or any other platform, so we won't spend time describing it in this document. In fact, nothing in the rest of this document is peculiar to AVS or vSphere – we are including them only for guidance.

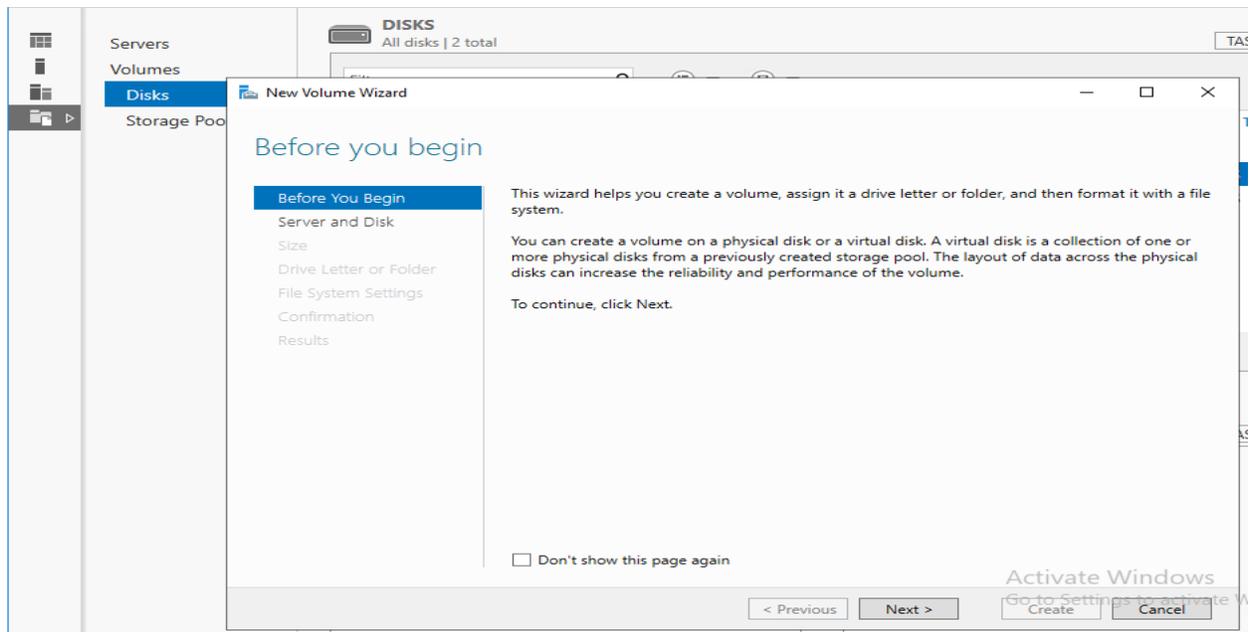
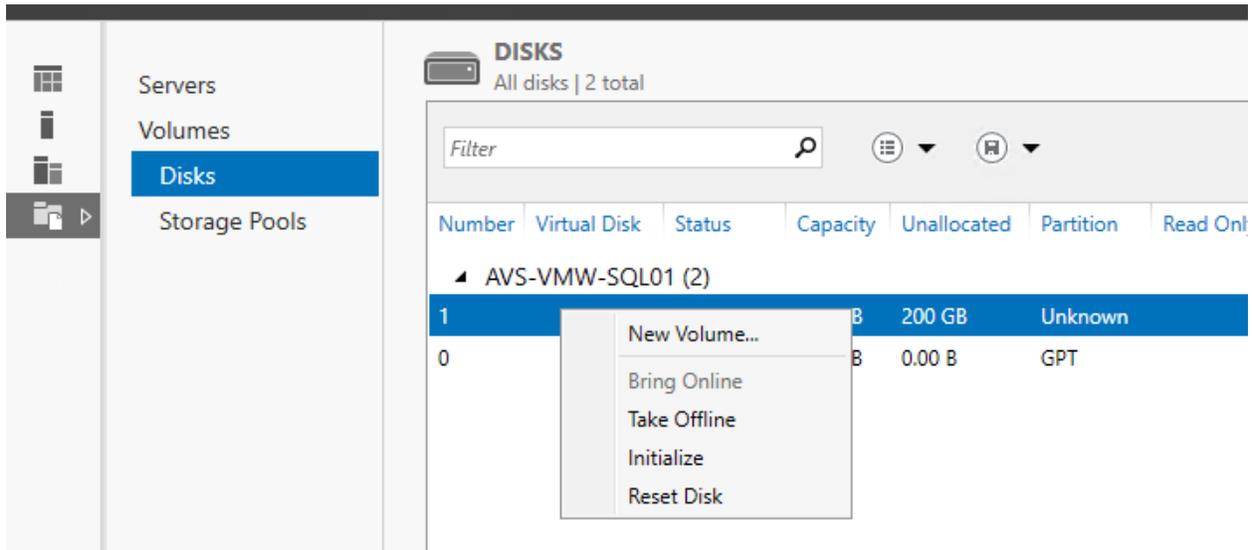
The newly-added disk is currently offline, according to Windows File and Storage Services. So, let's make it available and ready for WSFC's use (we will perform this task only on one of the nodes. There is no need to repeat the steps on any other node sharing the disk):



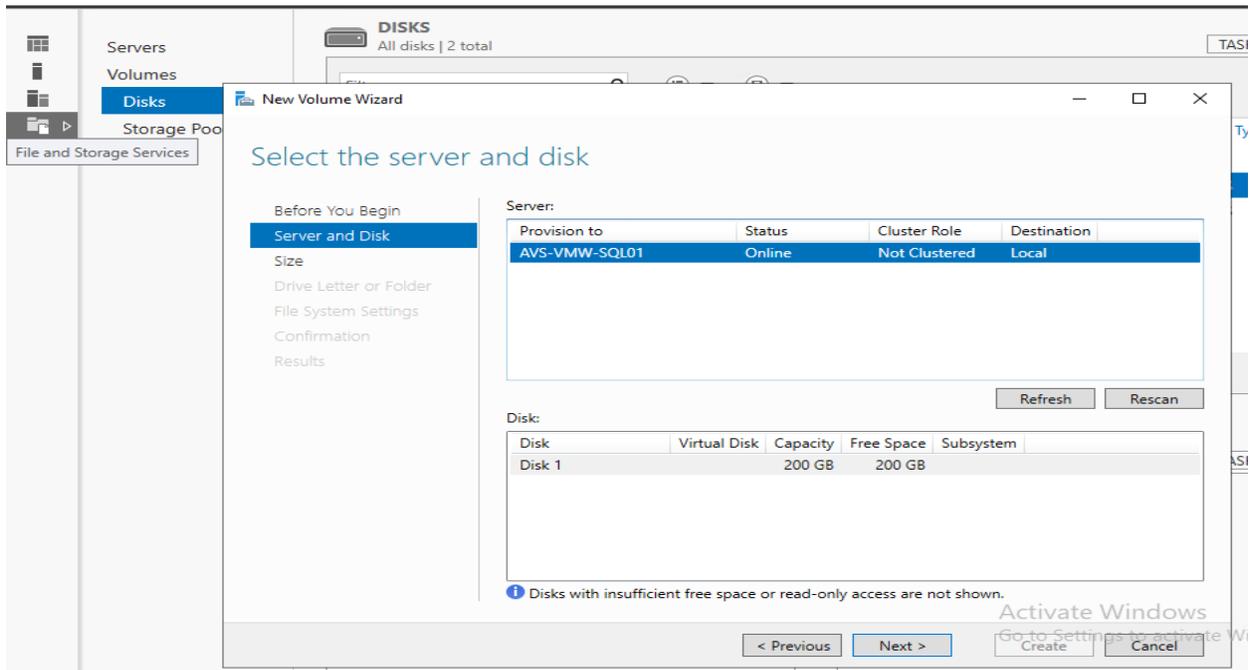
- We bring the disk online.



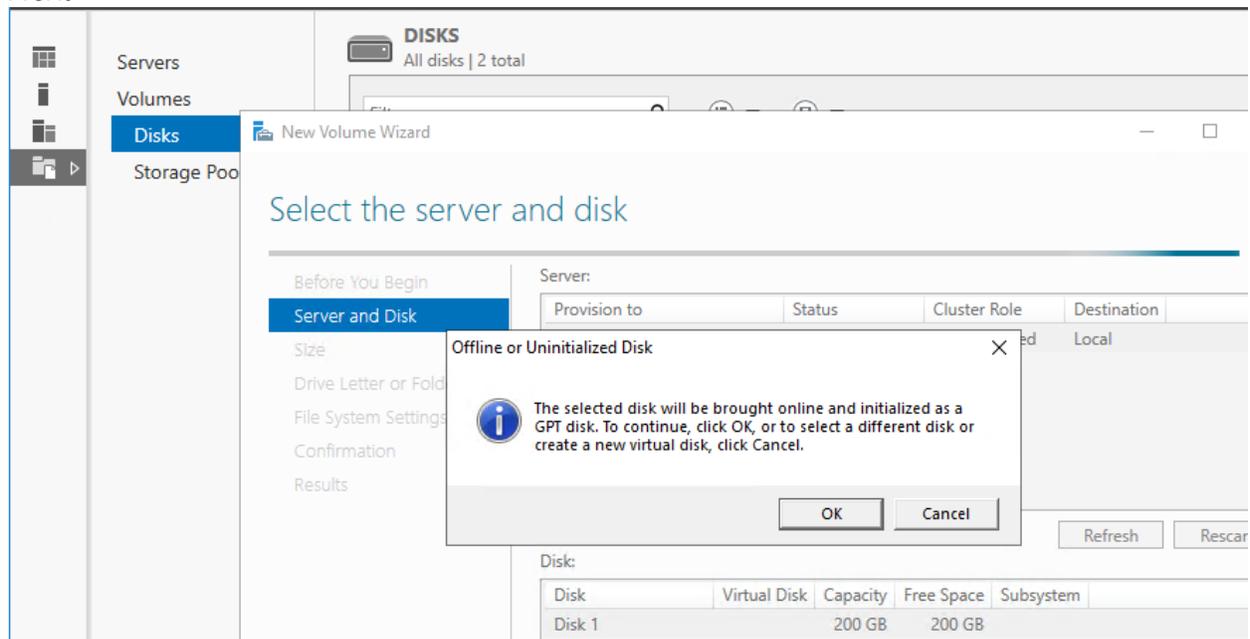
- Yes, we know what we are doing. Right?
- Now, we create a volume on it (standard Windows stuff, please feel free to skip ahead).



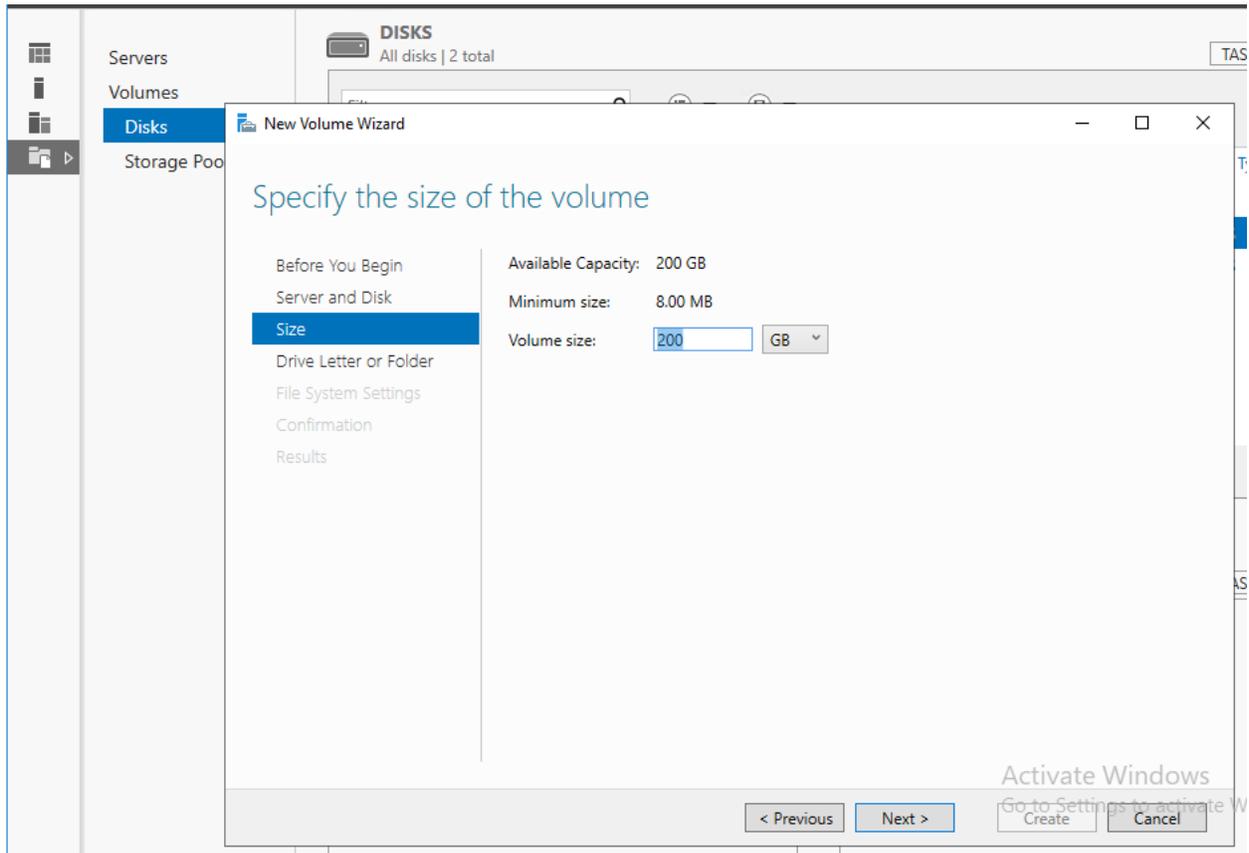
- Next



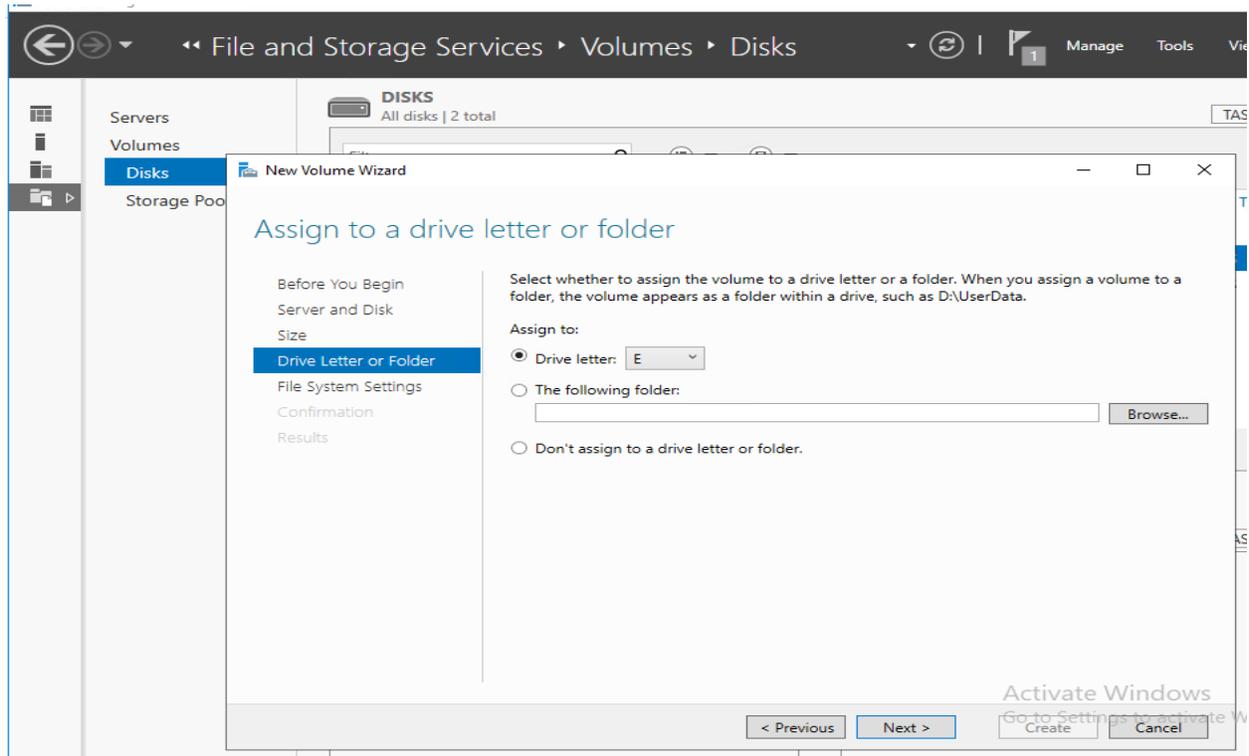
- Next



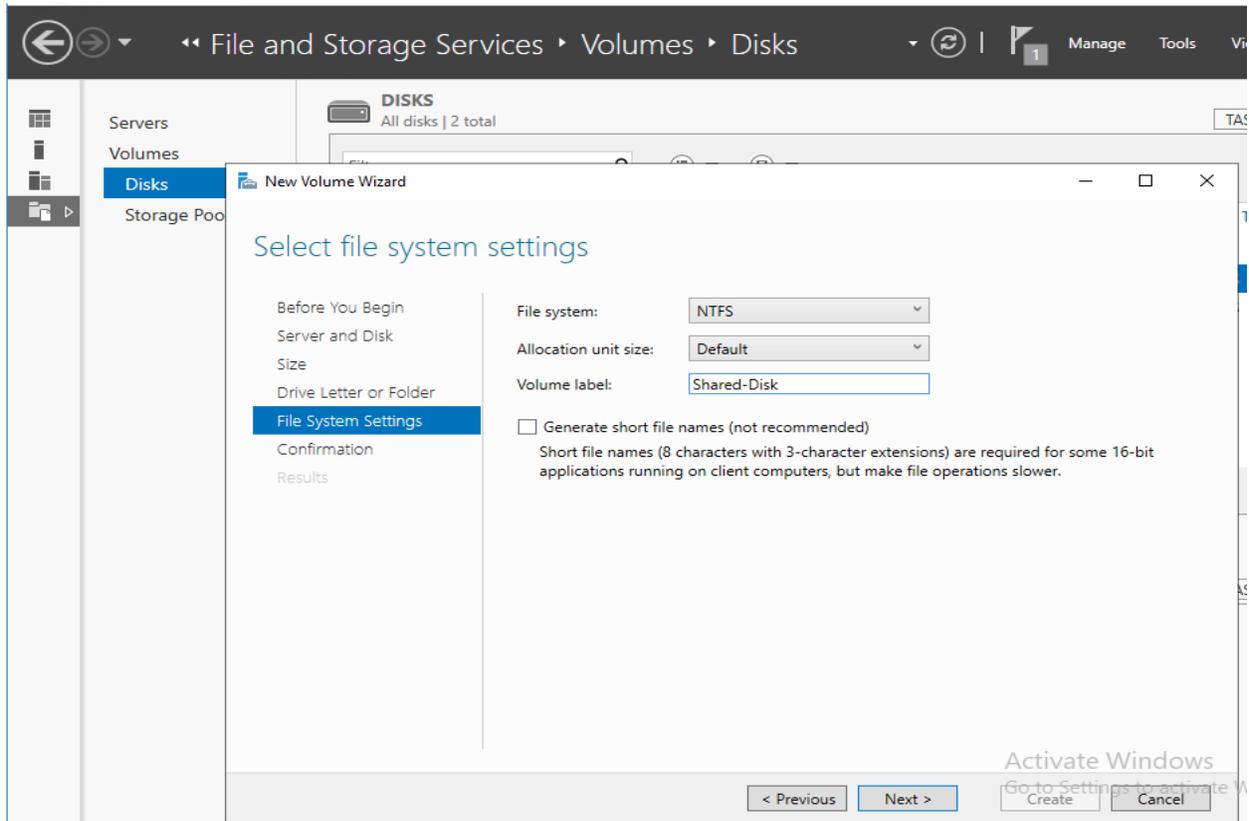
- OK



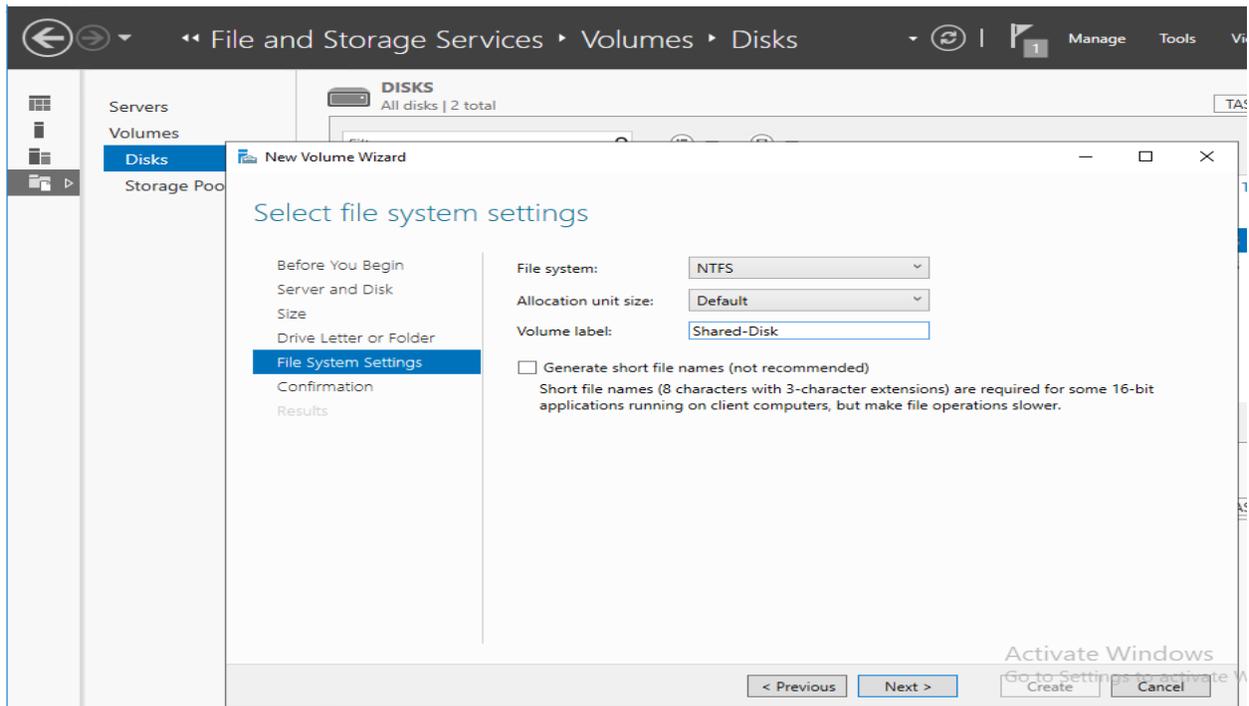
- Next



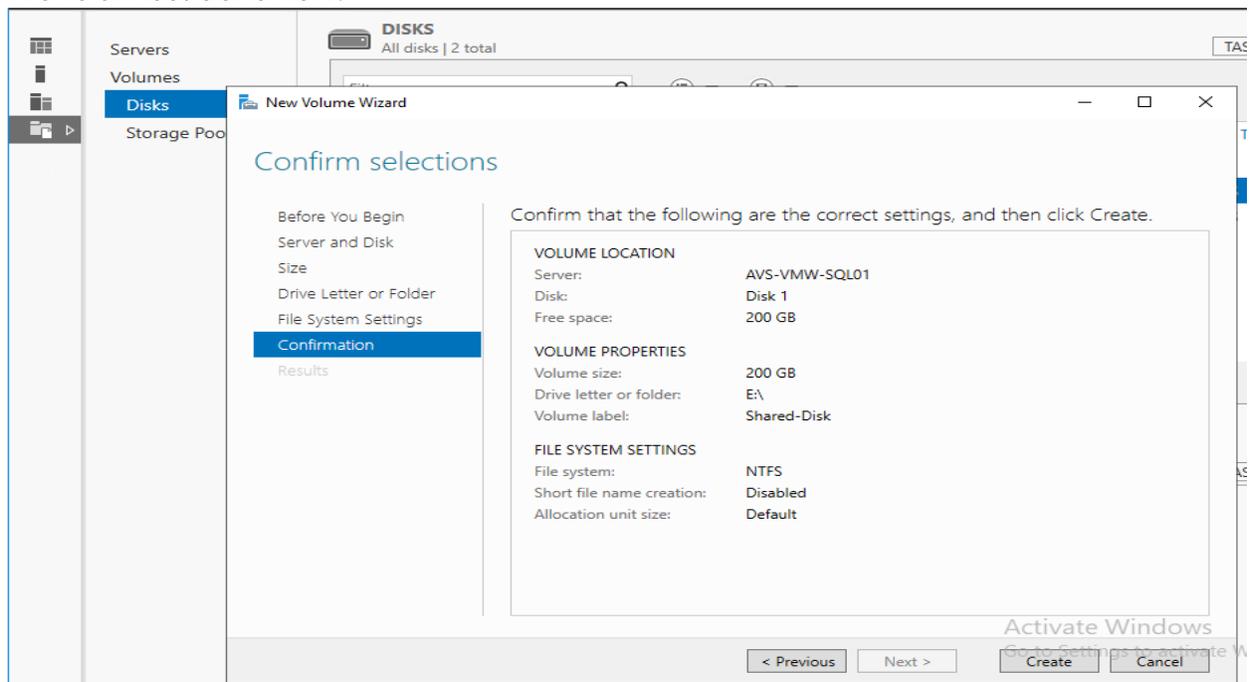
- Next



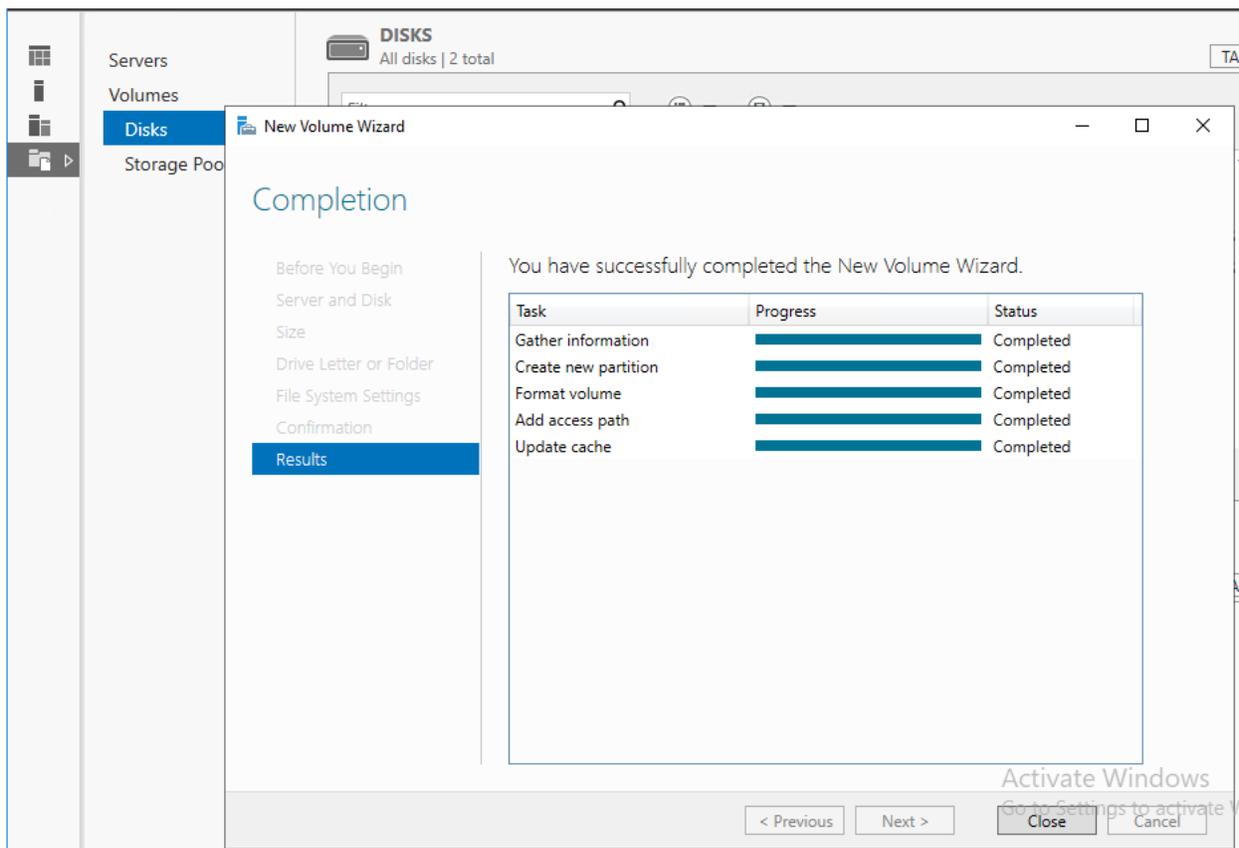
- Next



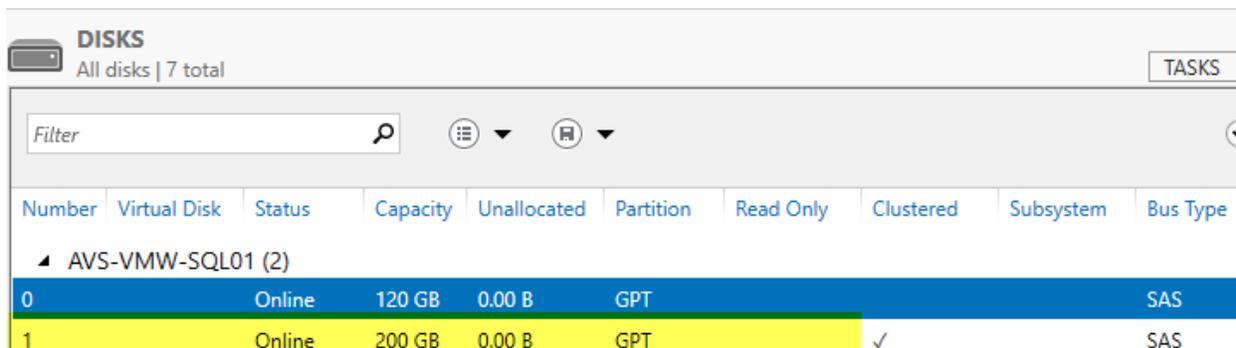
- Let's give it a fancy name and click **Next**.
- We're almost done now.



- Just hit **Create**, then **Next**



- Hit that **Close** button and let's get this over with.

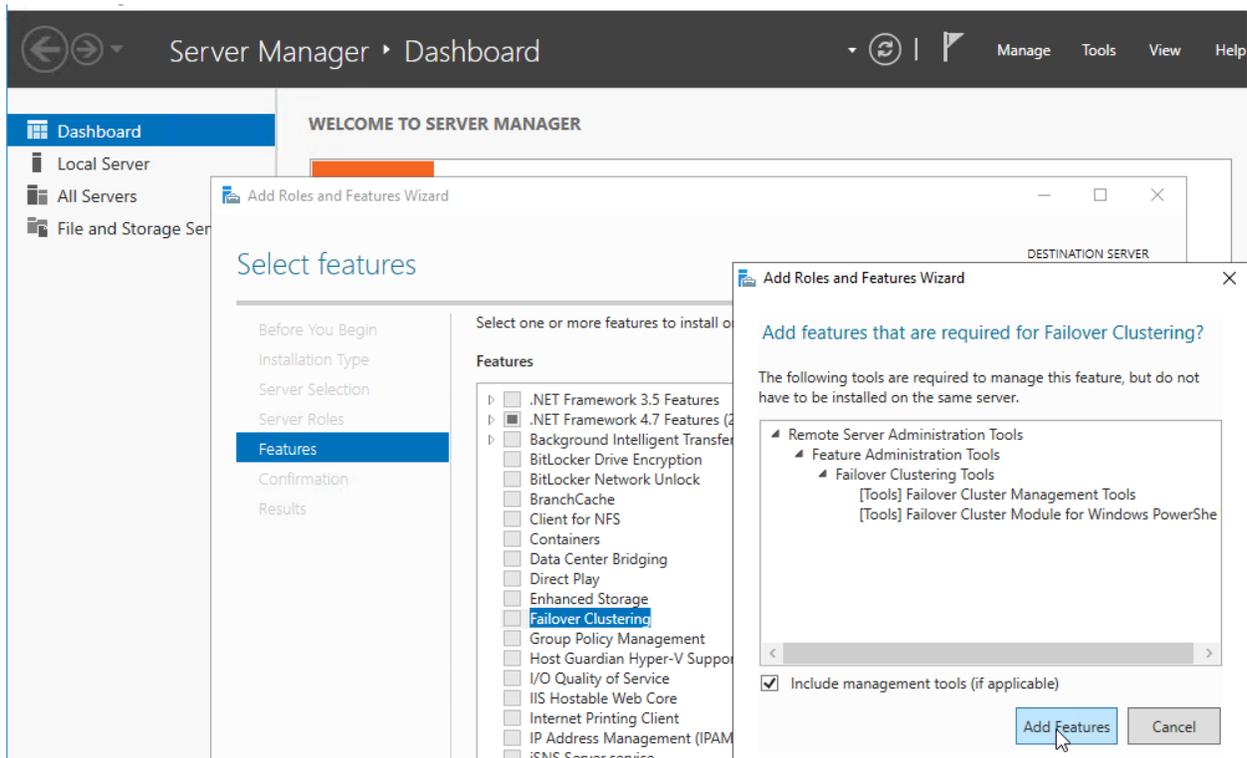


- There. Our shared disk is now ready for WSFC.

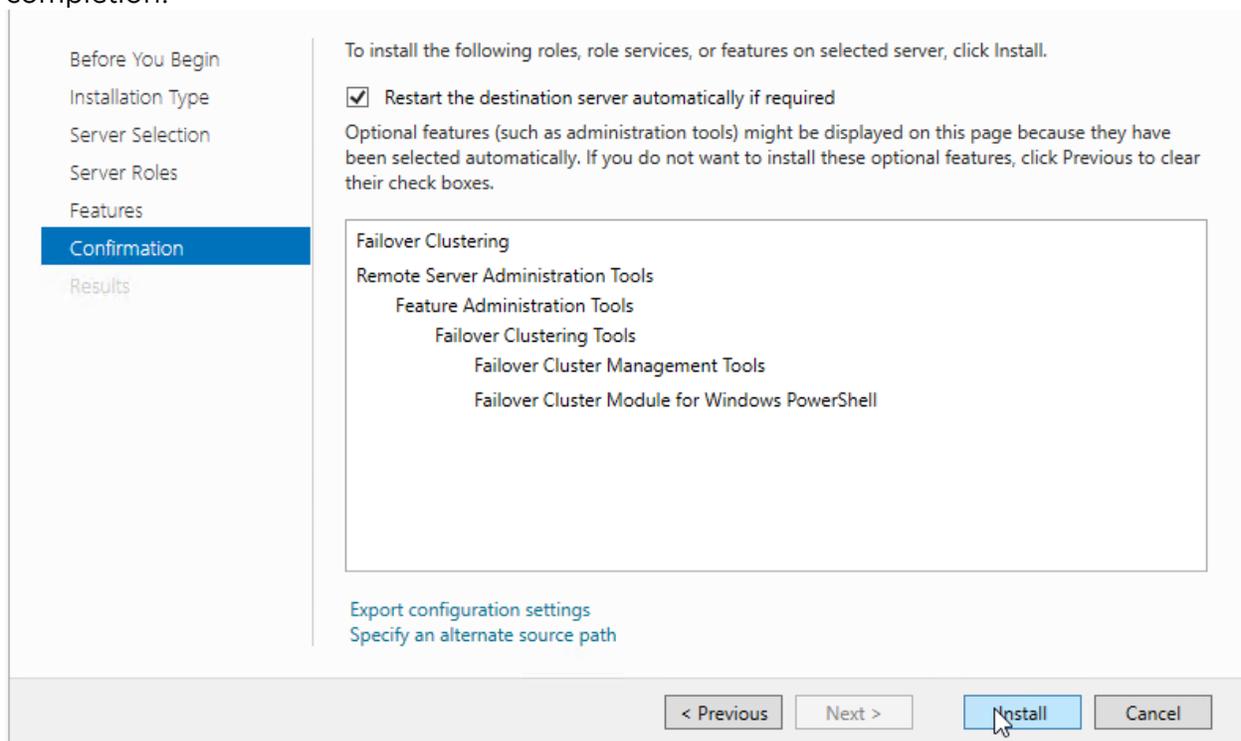
Windows Server Failover Clustering Configuration

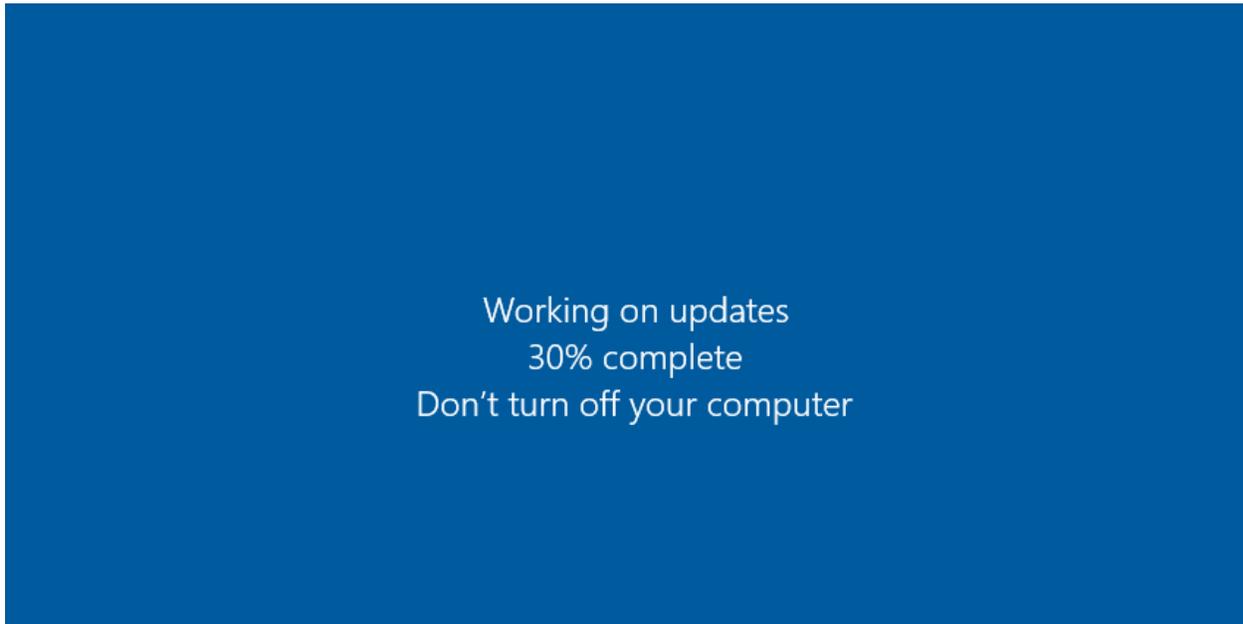
Installing Windows Server Failover Clustering (WSFC) on a VM in AVS is not different from doing it in any other Windows Server OS Environment.

- Add WSFC as a feature in Windows.

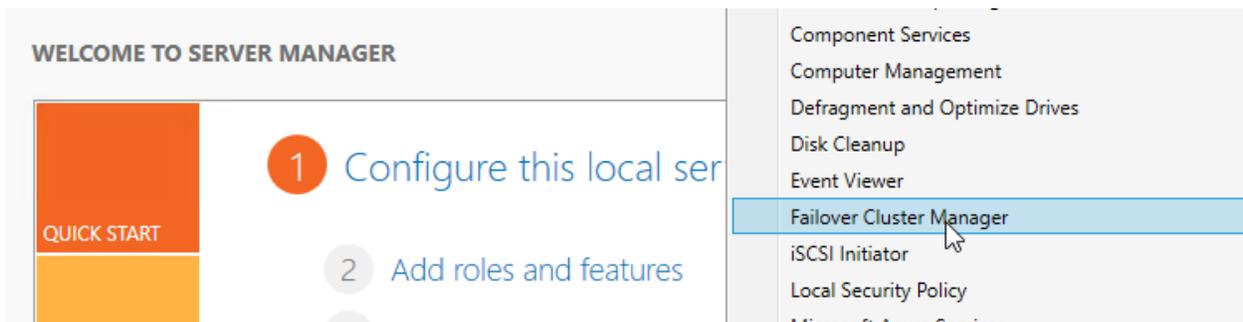


- Let's click through the process, remembering to check the option to auto-reboot upon completion.

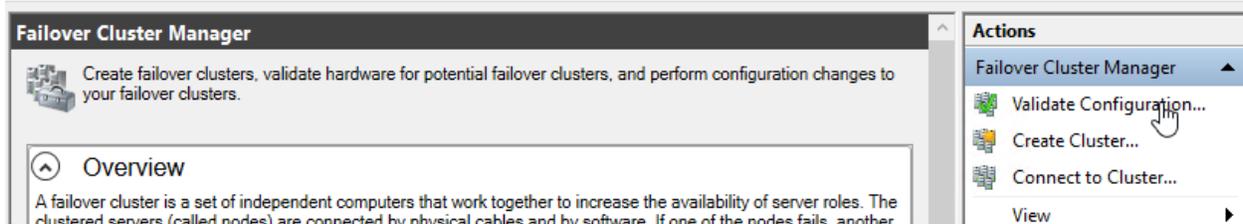




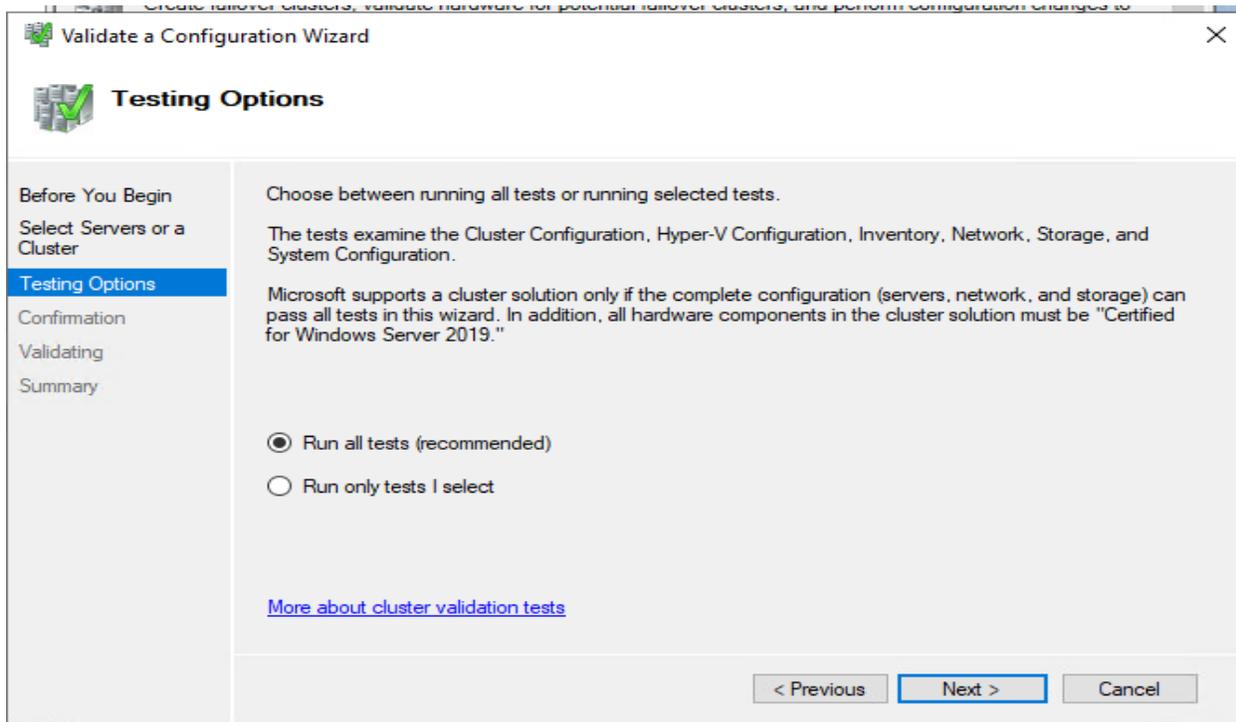
Now, we have successfully installed WSFC. It's time to get to work.



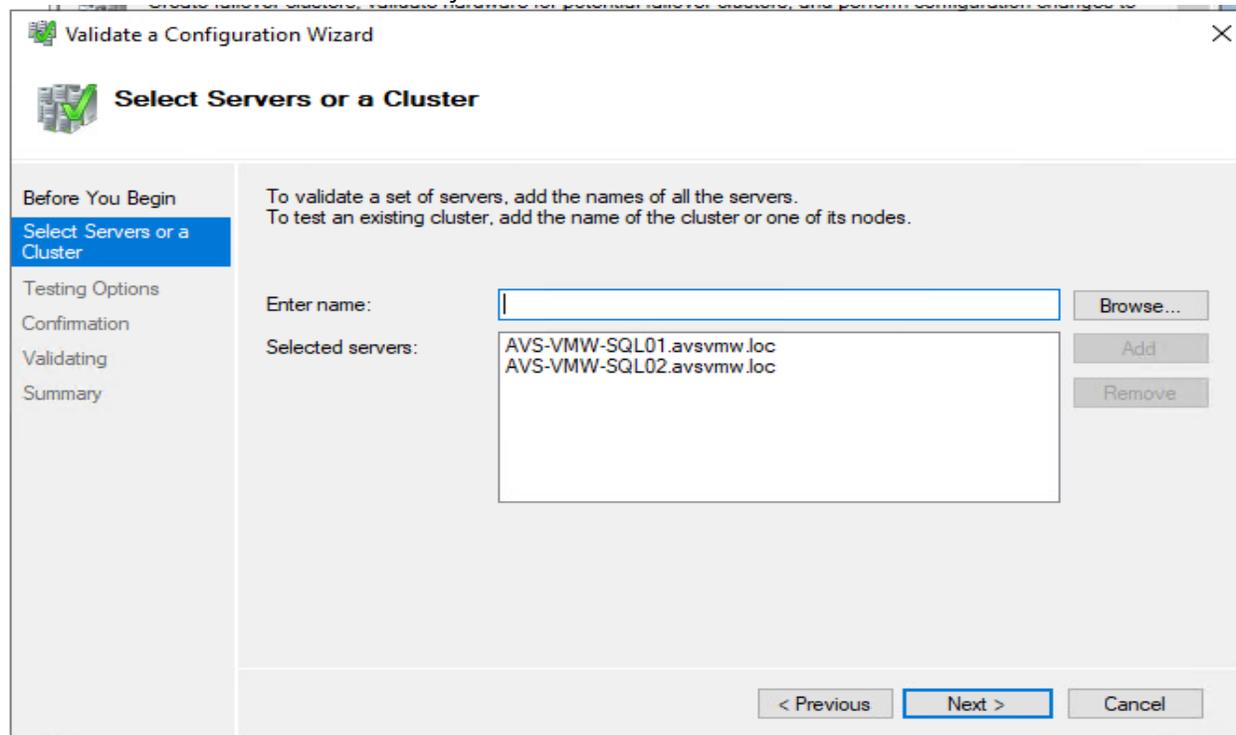
- We begin by launching the **Cluster Validation Wizard**.



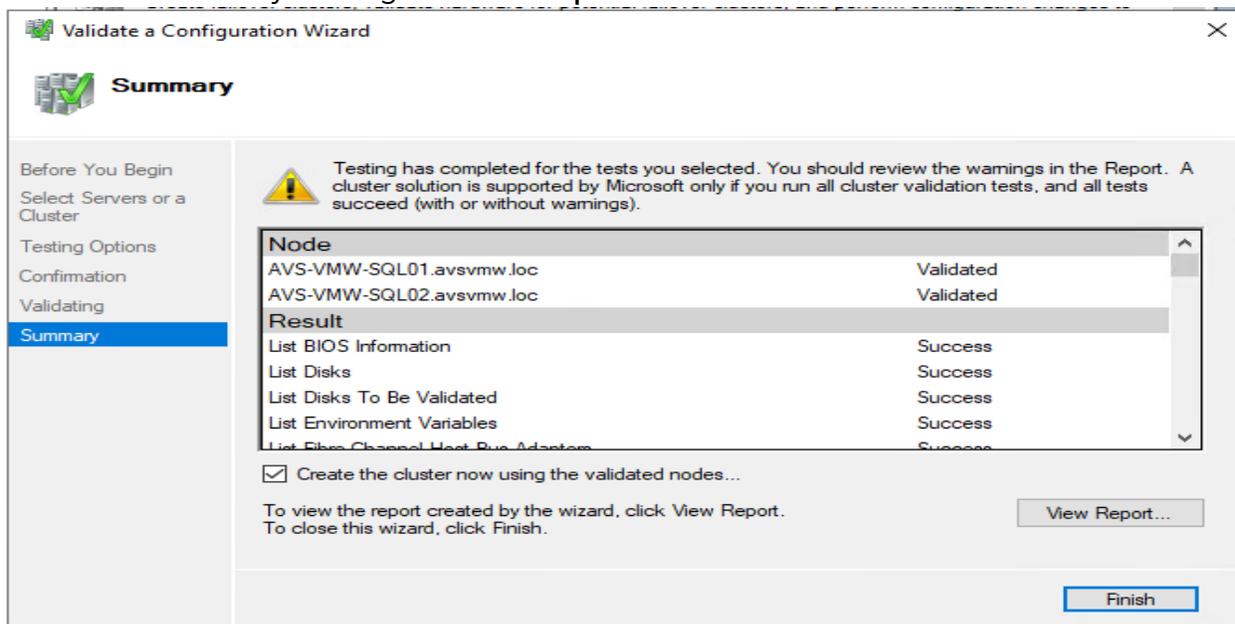
- We'll let the wizard run all recommended tests, just for completeness.



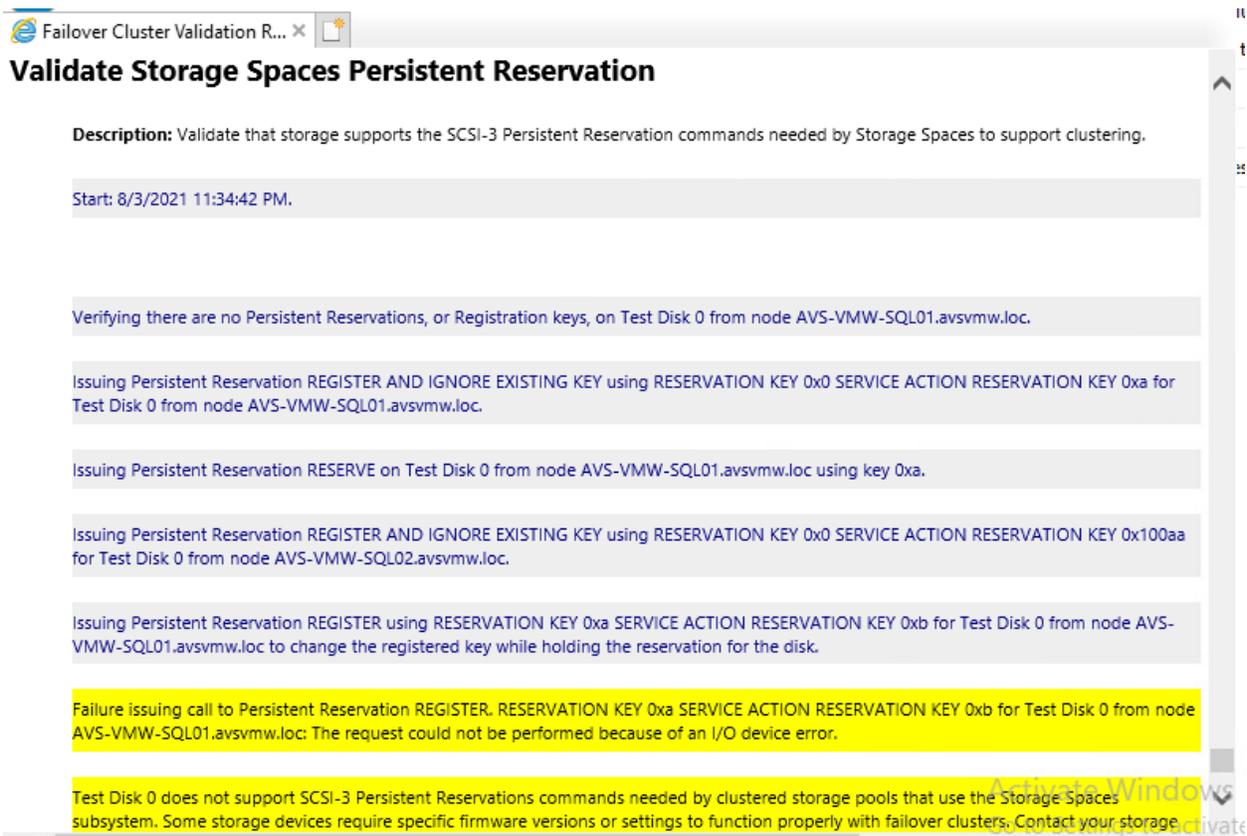
- We'll add the nodes we want to join into a cluster.



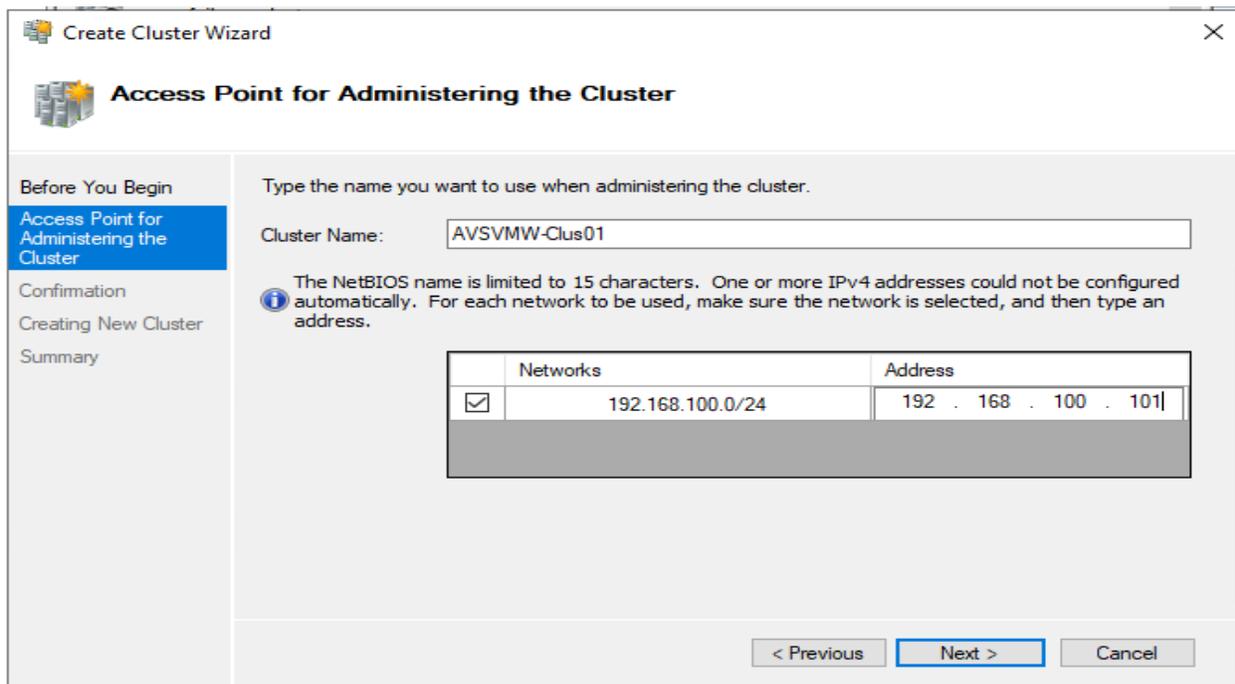
- Let's check the **Create the cluster now** box but, before we hit **Finish**, we'll take a look at the result of the tests by clicking the **View Report** button.



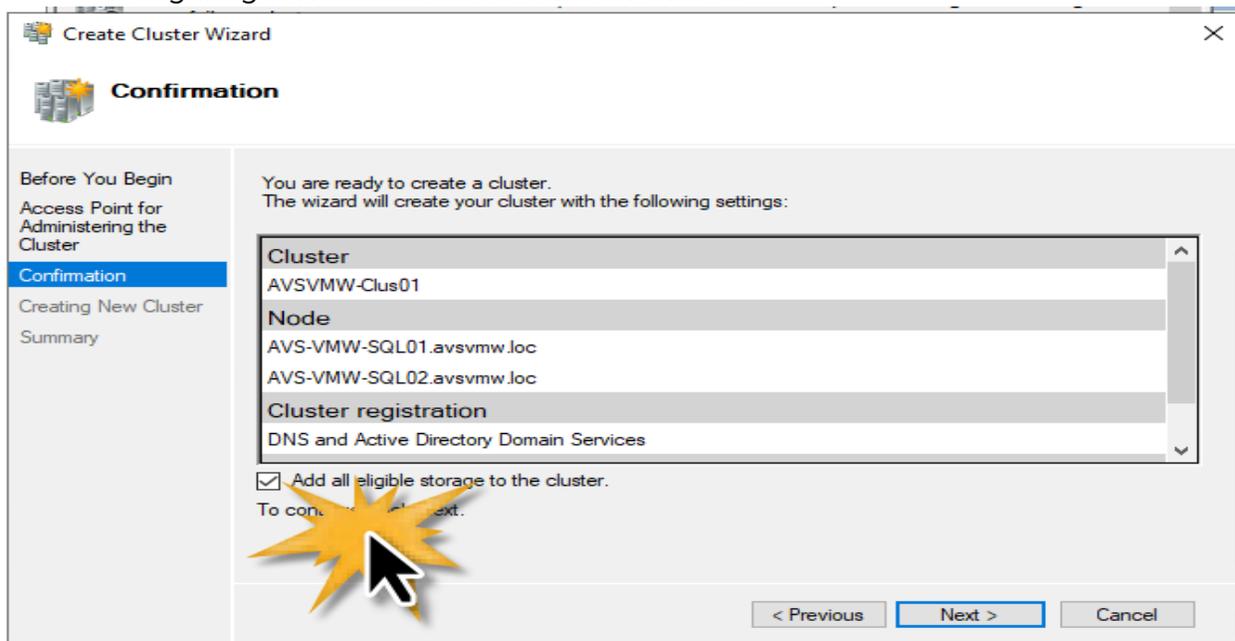
- Look at that! Some WARNINGS!!! Yes, we can safely ignore these warnings because they're from a check specific only to [Microsoft's Storage Spaces subsystems](#) and is, therefore, irrelevant in our AVS environment.



- We can close the Report and click **Finish** on the Wizard to complete our validation test and proceed to setting up the cluster.



- We give the cluster’s administrative access point a name and its corresponding IP.
 - **NOTE:** As mentioned previously, we assume familiarity with all of these steps, so we won’t describe how to do most of the Active Directory, DNS preparatory tasks related to configuring a Windows Server Failover Cluster



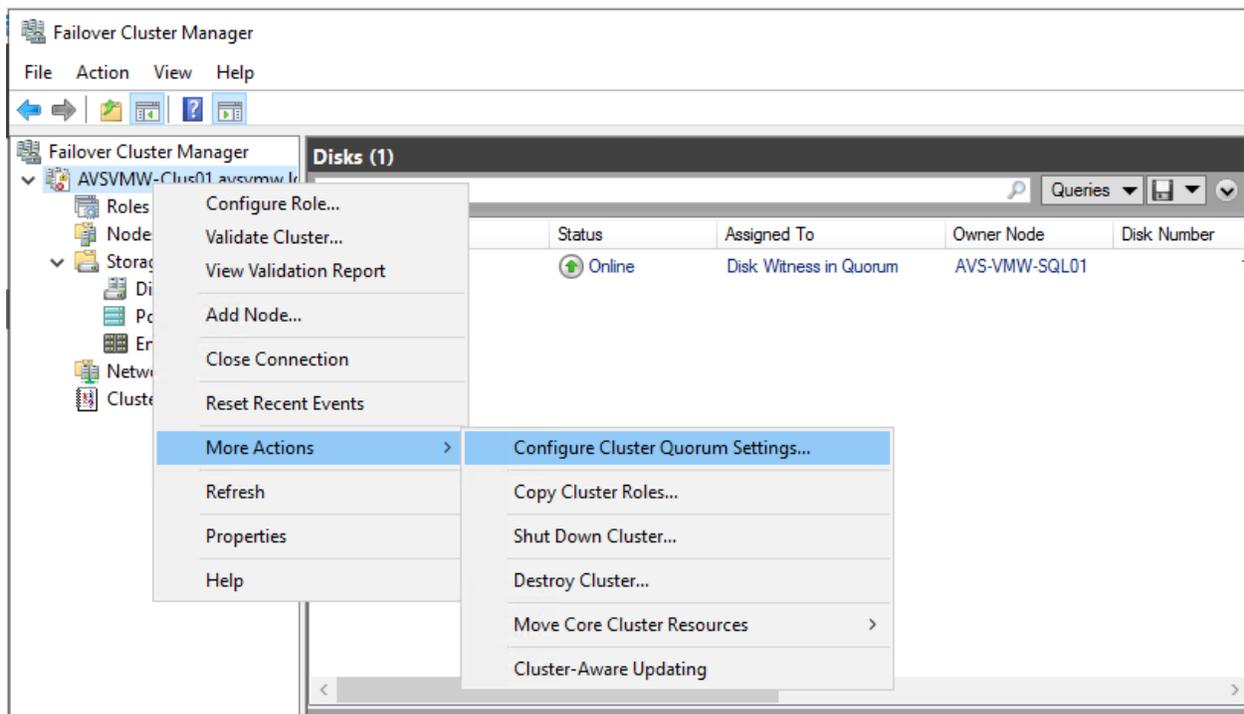
- Checking the **Add all eligible storage to the cluster** option is not recommended because, among other things, it can lead to WSFC making the wrong choice for you about the available disks and

what you intend to do with them. For example, as shown in the next image, WSFC has added our Shared Disk and assigned it to the Cluster as our Witness Disk.

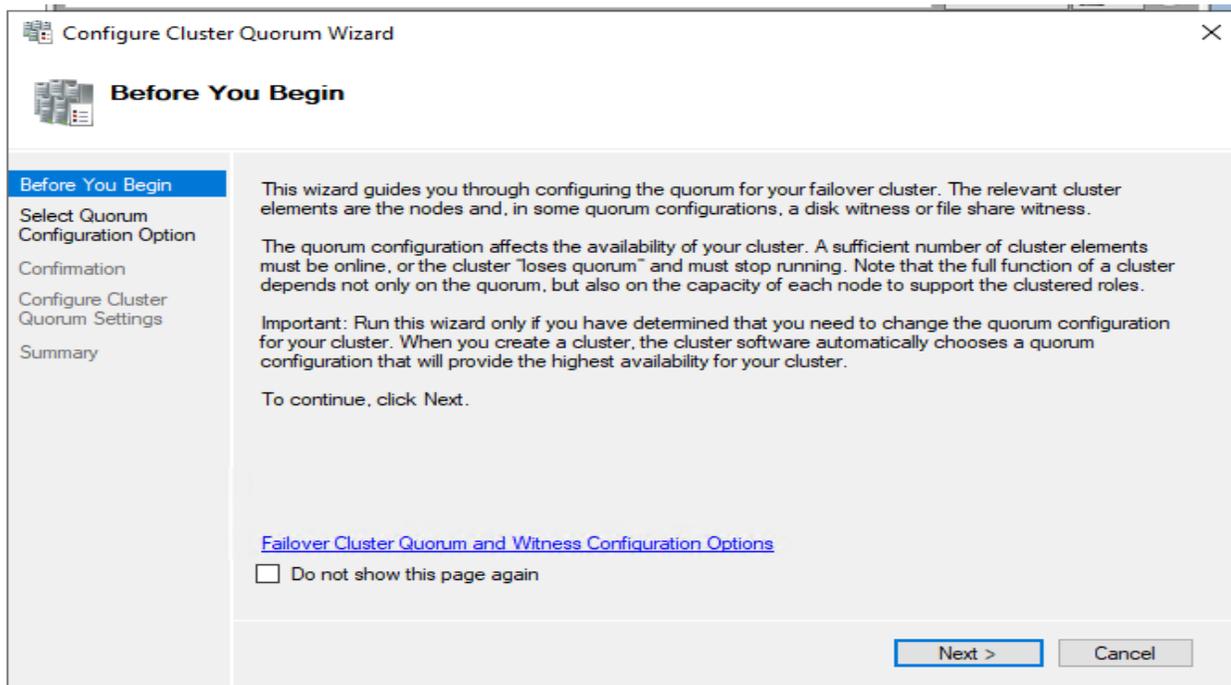
- o **NOTE:** Do not place the File Share Witness folder on any of the nodes participating in the Cluster. The Witness gets a vote in the Quorum, so if you put it on one of the nodes and the node becomes unavailable, you've lost two votes in the Quorum.



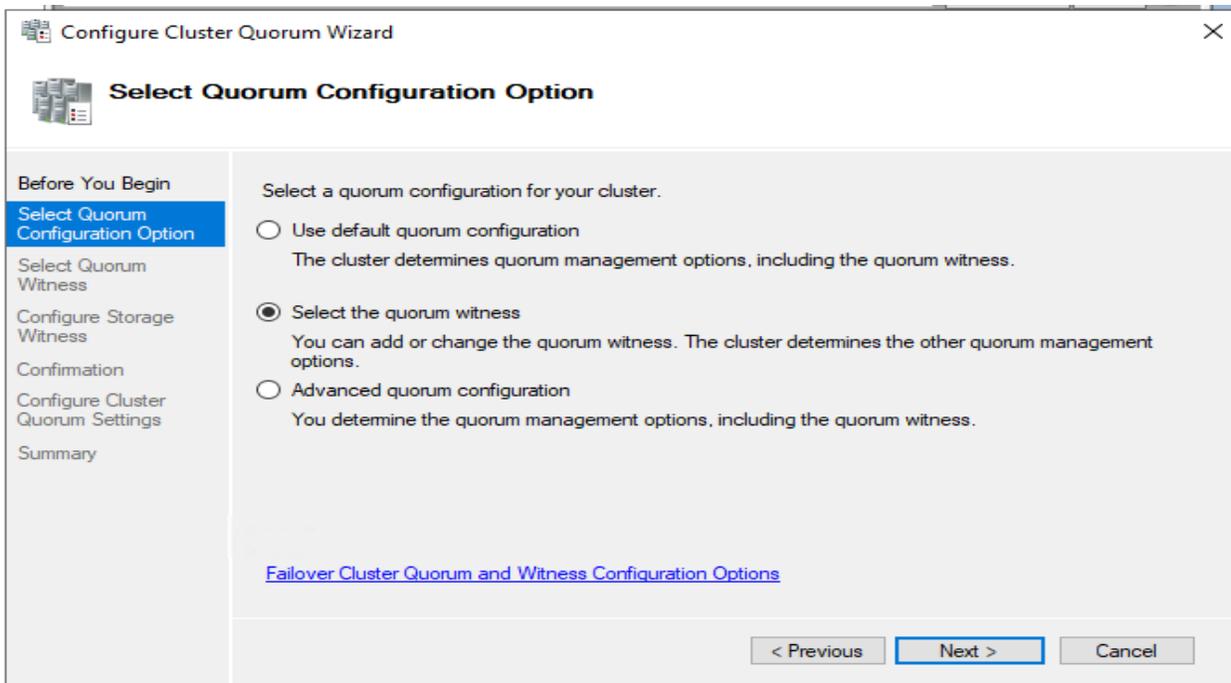
- This is definitely NOT what we want, so let's correct this quickly.
- We are going to change the Quorum setting to use a File Share Witness (FSW) instead. FSW (which is simply a folder on any server on the network, shared and made accessible to the clustered nodes and the Cluster Network Object CNO]) is easier to configure and maintain and works well for our purposes.



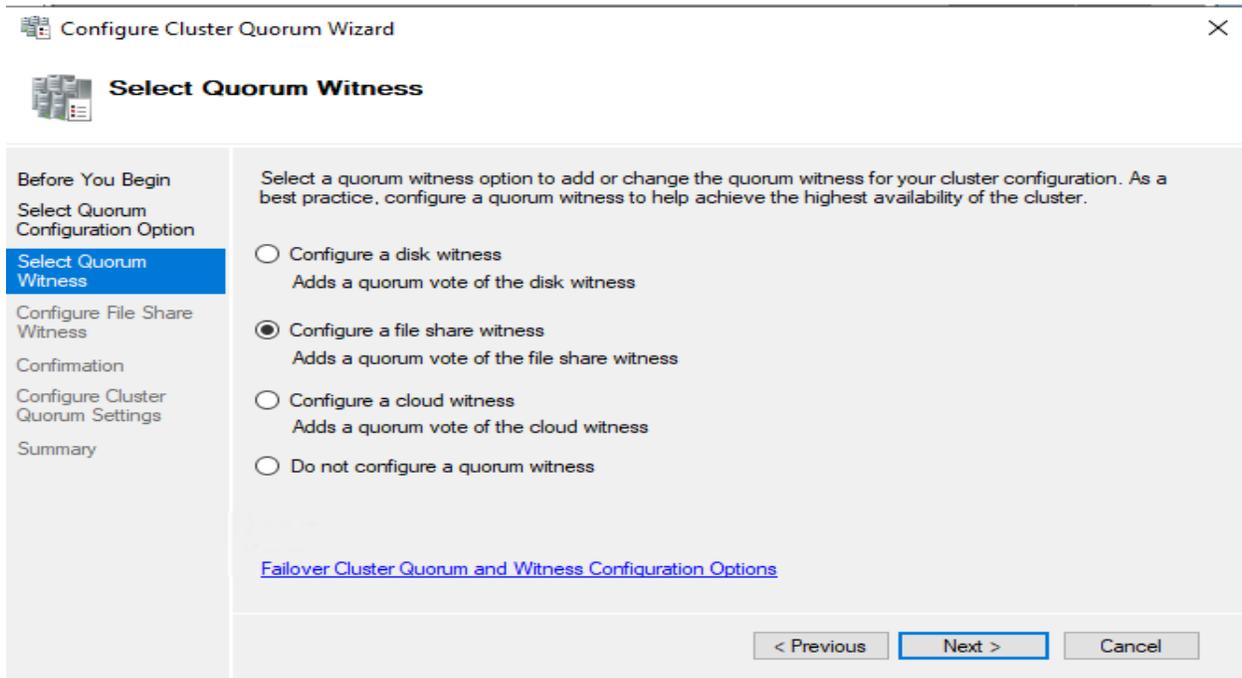
- This involves a series of **Next, Next** clicks...



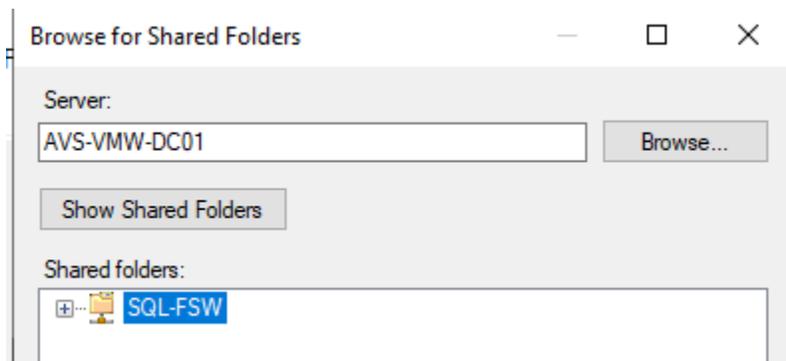
- Click **Next**.

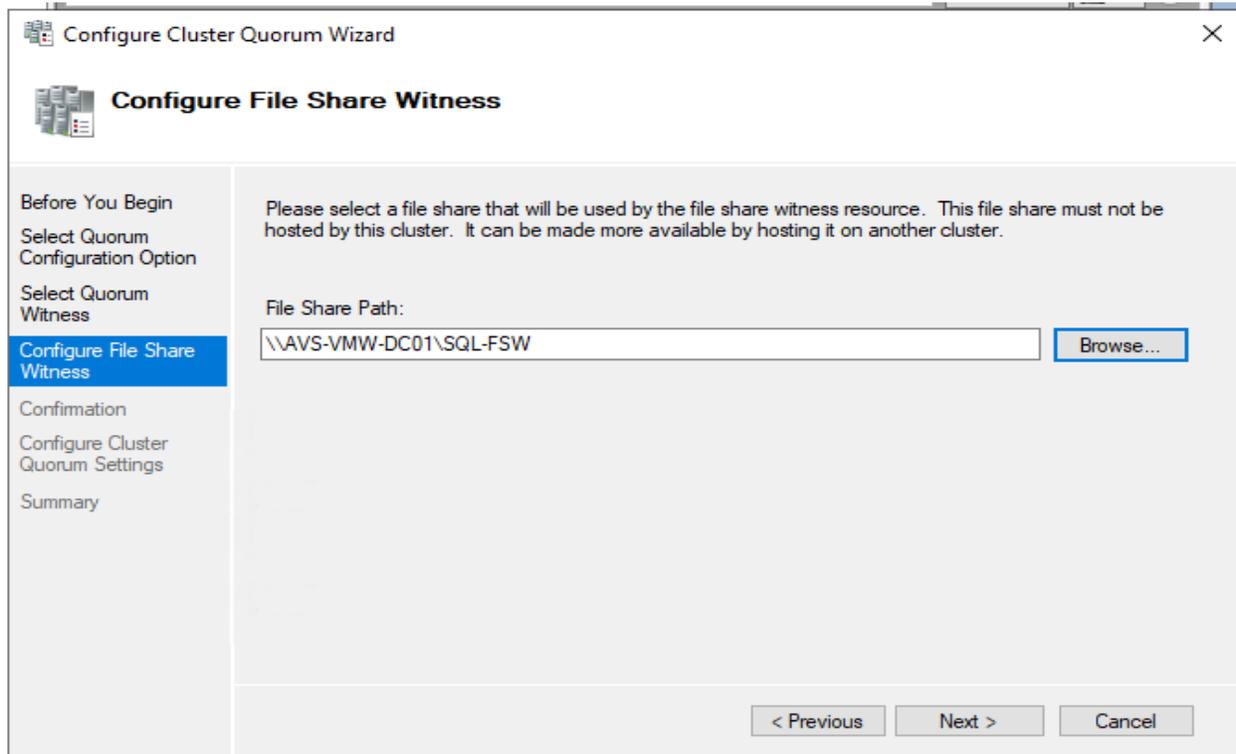


- Choose the **Select Quorum Witness** option, then click **Next**.

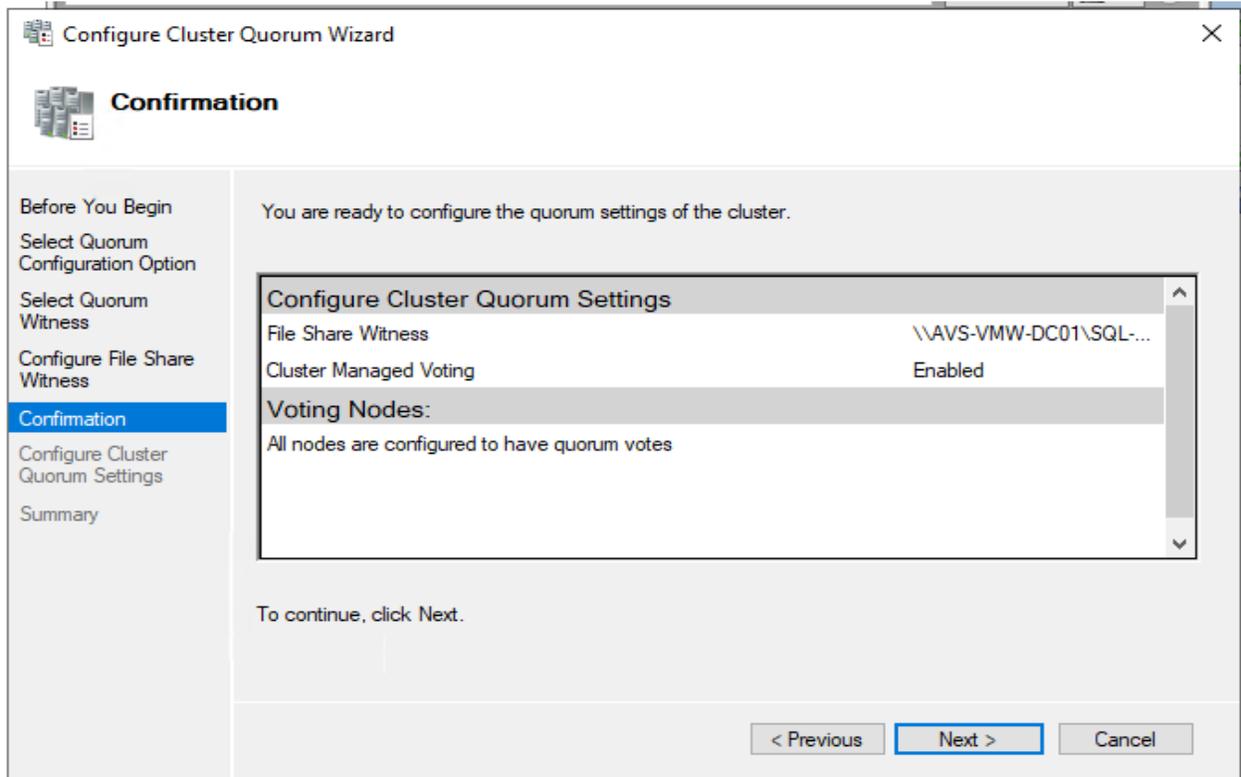


- We want the **File Share Witness** option, so check that and click **Next**.
- Browse to where the share is located, then select it.

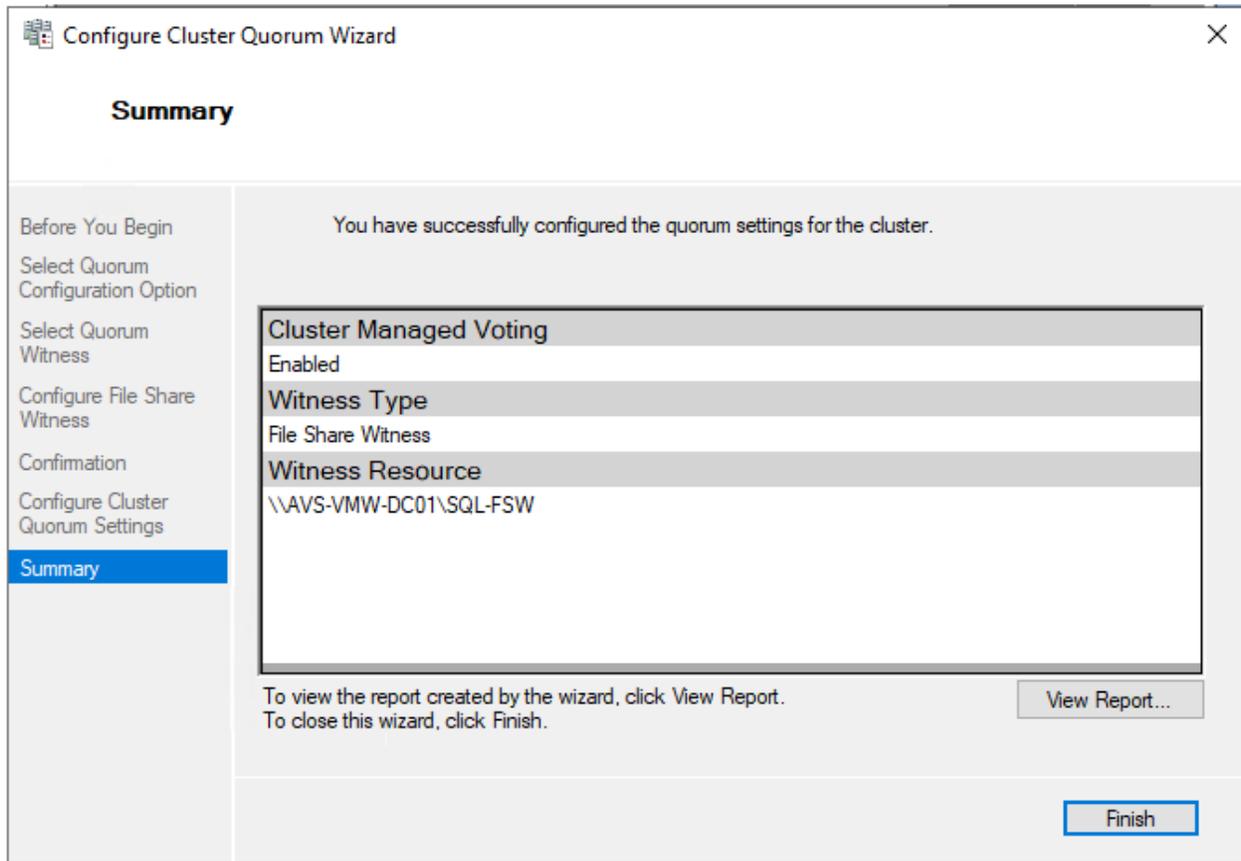




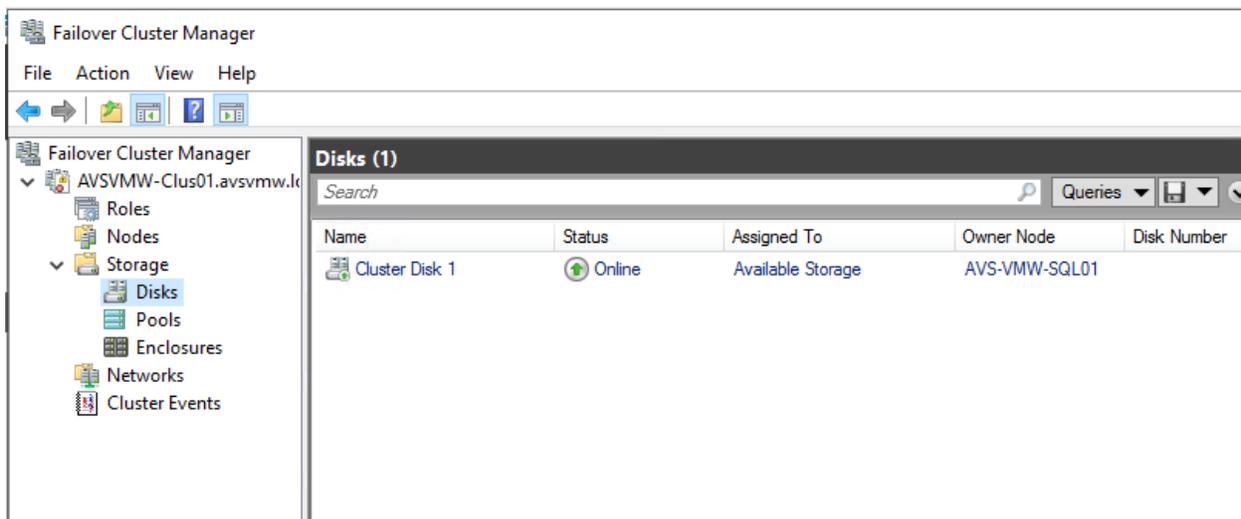
- We confirm that we have selected the correct share, then we click **Next**.



- One more **Next**



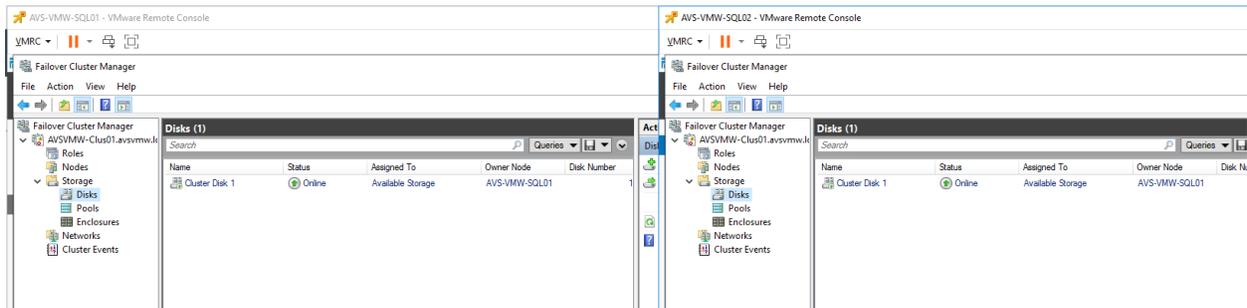
- And, we wrap it up with a **Finish** (unless you'd like to read the Report, of course).



- Now, our Shared Disk is recognized by WSFC as we intended for it to be – **“Available Storage.”** We can now use it for our Microsoft SQL Server FCI-clustering purposes.

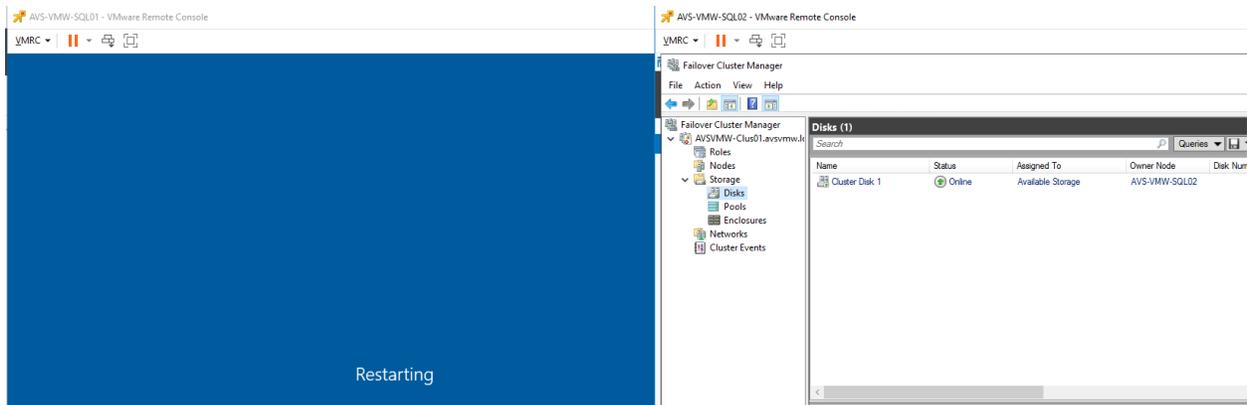
But, before we do that, why don't we confirm that the disk is actually a clustered resource and can survive the failure of one of the nodes we just clustered? This would give us the peace of mind we need before laying our SQL Server bits on top.

Here is how both nodes see the shared disk after we've completed the cluster setup.



- Both nodes see that SQL01 currently owns the disk, which is fine.

But what happens if SQL01 were to become suddenly unavailable?



- WSFC does what it's supposed to do – it lets SQL02 take ownership of the clustered resource.

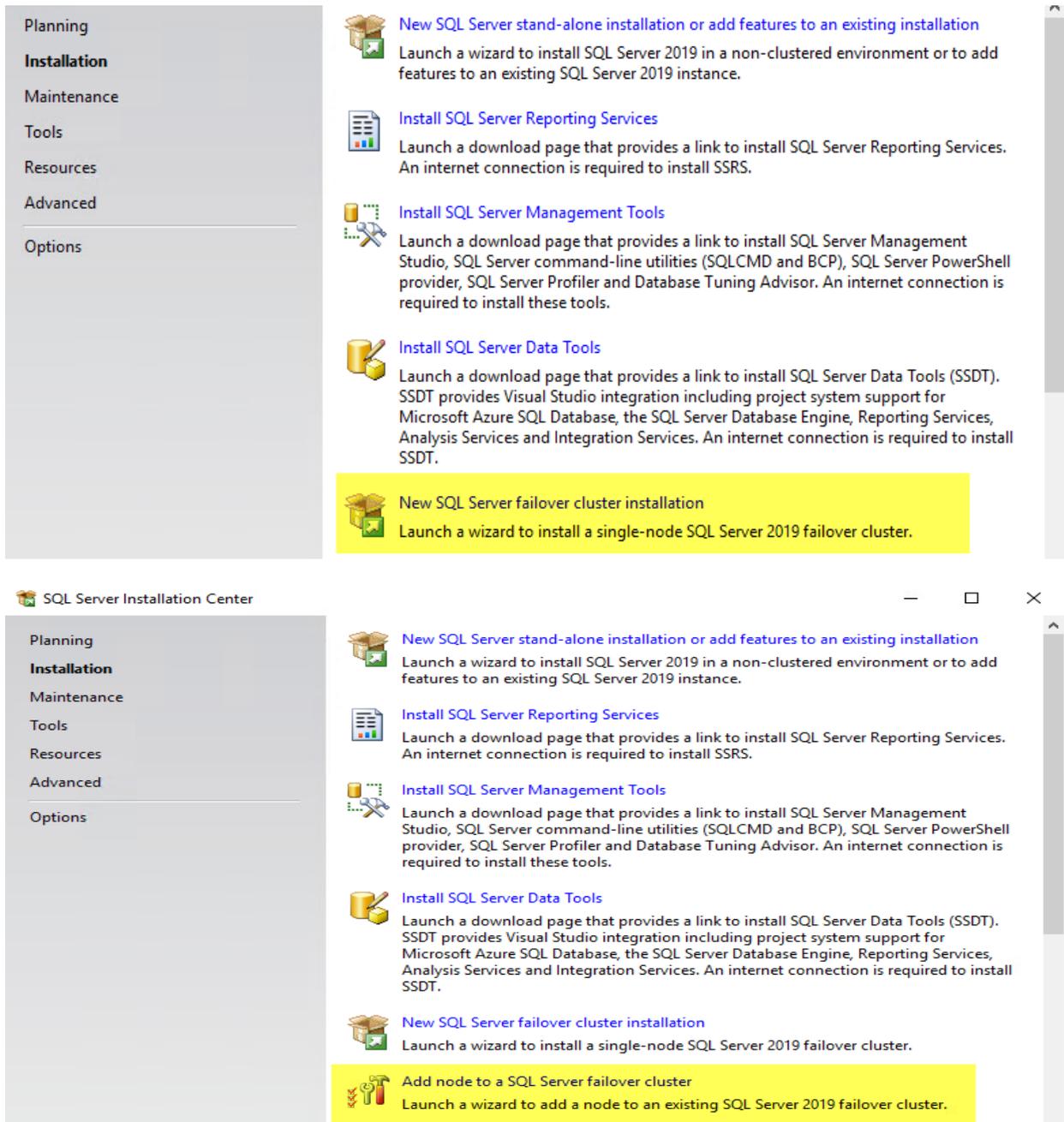
At this point, we have completed our demonstration of clustering shared resources (disk) for multiple Windows Server Failover Clustering nodes on Azure VMware Solutions. As you can see, the actual WSFC installation and configuration procedures are similar, regardless of the platform.

We are now going to simulate and document layering Microsoft SQL Server instances and databases on top of the clustered infrastructure.

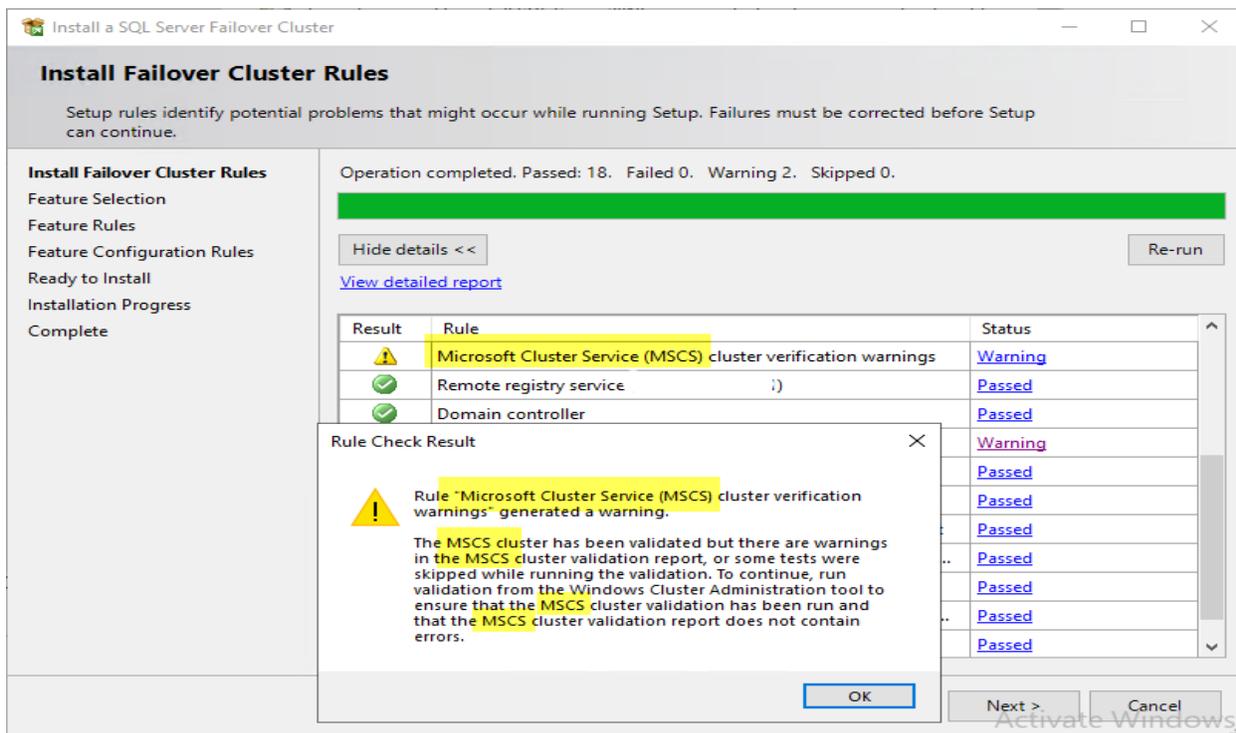
NOTE: As before, we assume prior knowledge of the administrative steps and tasks involved in setting up FCI and availability group in a typical Microsoft SQL Server environment, so we will not be describing these in detail.

Installing and Configuring Microsoft SQL Server Always On Failover Clustering Instance on AVS

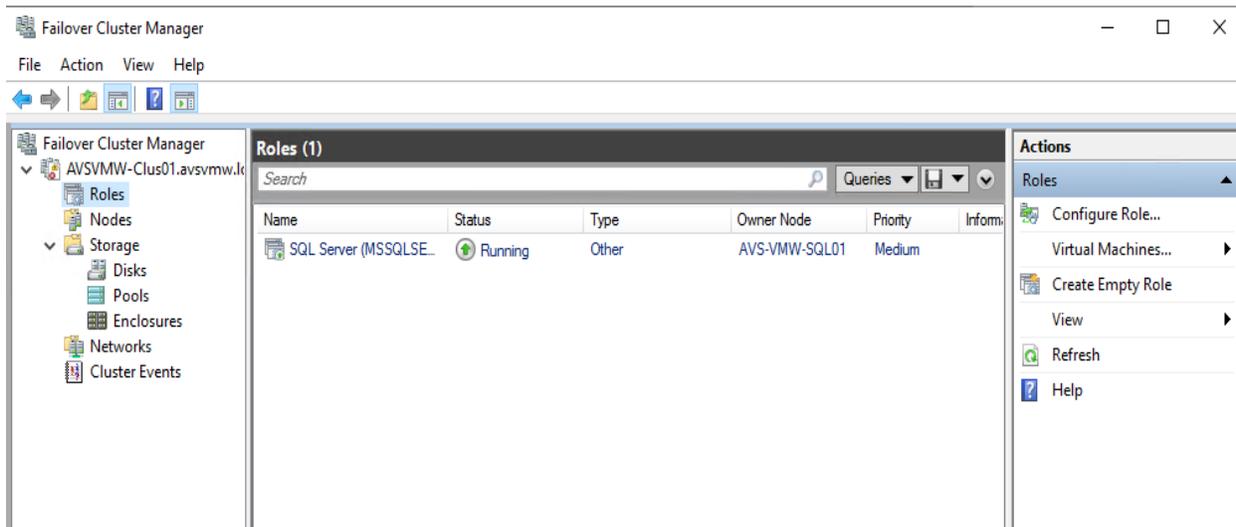
We will install Microsoft SQL Server first on SQL01, then on SQL02. The major difference in the installation process is that we will select the **New SQL Server Failover Cluster installation** option for SQL01 and the **Add node to a SQL Server Failover Cluster** option for SQL02 as shown below:



The install process wants to remind us that we did ignore some warnings during the WSFC validation test. We know this already, so we will ignore the reminder and proceed with our installation.



- After successfully installing SQL Server on SQL01, we can see in Cluster Manager that **SQL Server (MSSQLServer)** has become a clustered role in WSFC.



We can now install SQL Server on the second Node (SQL02), joining it in as a part of the SQL Server cluster we just installed on node 1 (SQL01). But, before we do so, we have some housekeeping tasks to take care of. One is related to improving disk IO for Microsoft SQL Server workloads in vSphere-based

hybrid cloud environments, and the other is maintaining availability and resilience for WSFC-based clustered resources generally.

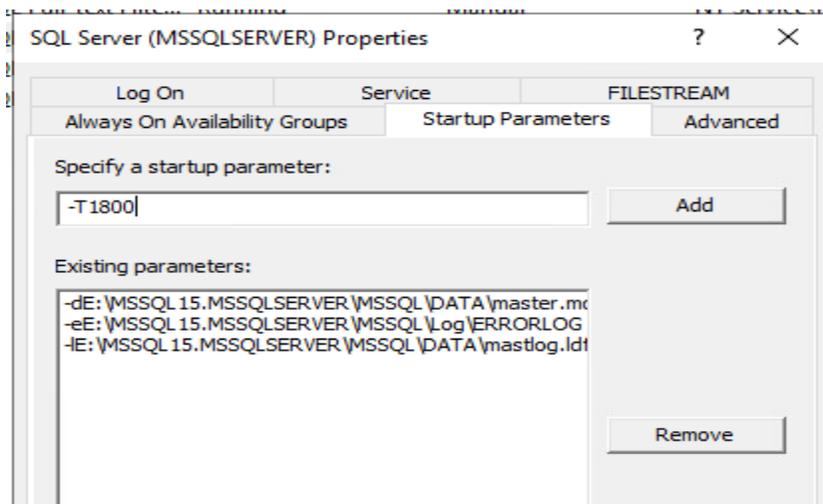
Enable (Persistent) Trace Flag 1800

Disk IO misalignment is a very common cause of severe suboptimal performance of Microsoft SQL Server queries, synchronization and transactions. Although Microsoft has long released patches to address the specific Sector misalignment issues, an administrative action is still required in order for the fixes to apply to a given SQL Server instance. Please see [KB3009974 - FIX: Slow synchronization when disks have different sector sizes for primary and secondary replica log files in SQL Server AG and Logshipping environments \(microsoft.com\)](https://support.microsoft.com/kb/kb3009974) for more detailed information.

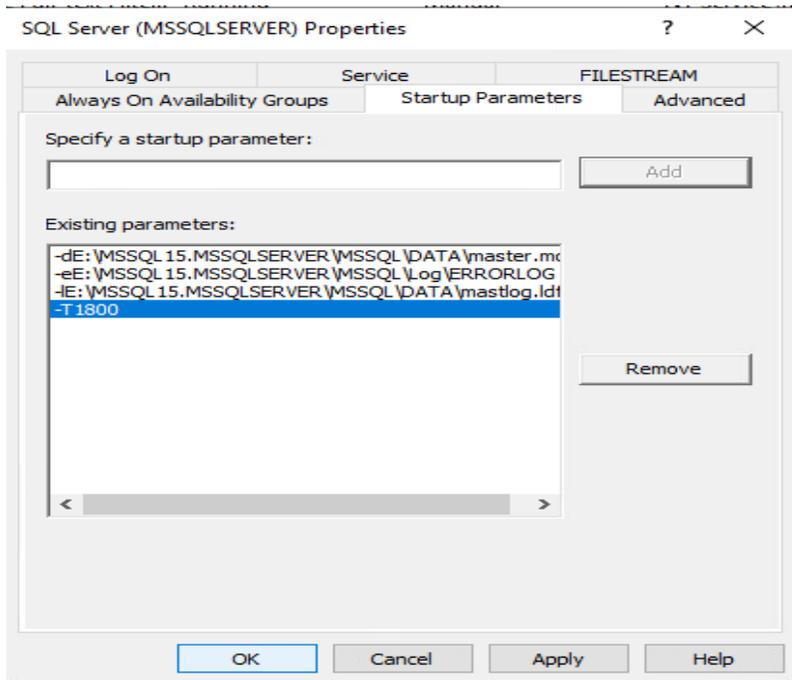
So, let's go ahead and do this for our environment.

The TraceFlag will be global and persistent (it needs to remain in place, even after a service restart or server reboot), so we will do this inside the SQL Server Configuration Manager (SSCM).

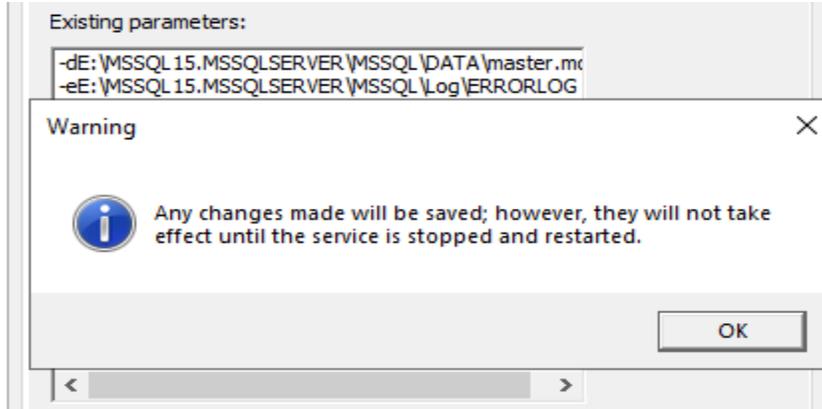
- Open SSCM and navigate to the **Startup Parameters** tab.
- Type in **-T1800** (TraceFlags trace flags are preceded by “-T”), then click **Add**.

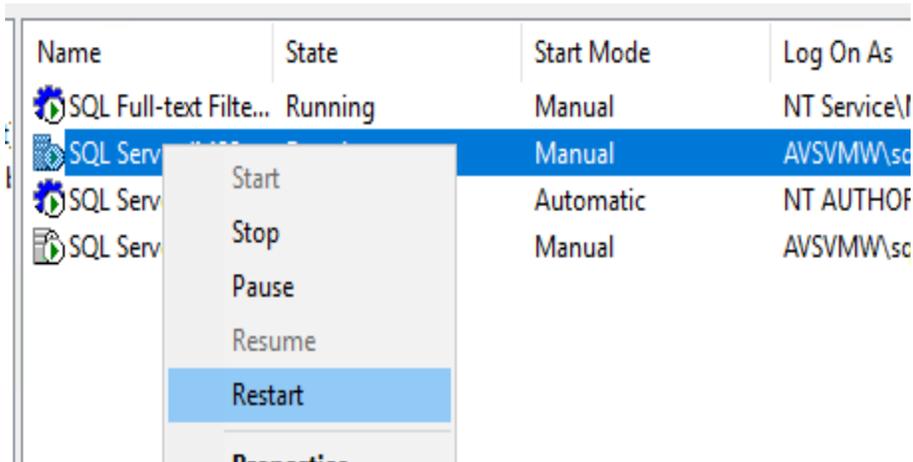


- Click **OK**.



Yes, yes, we have to restart the service for this to take effect, so let's do so now.

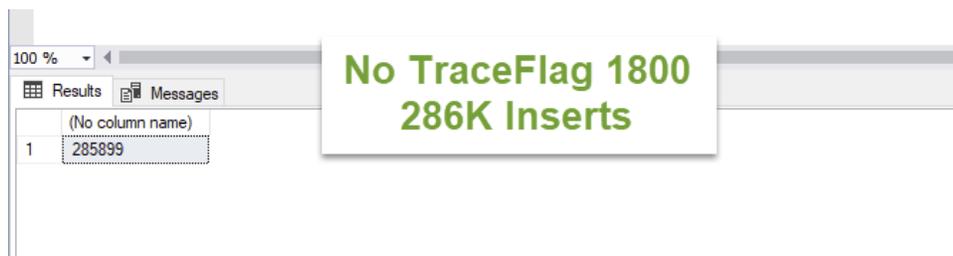




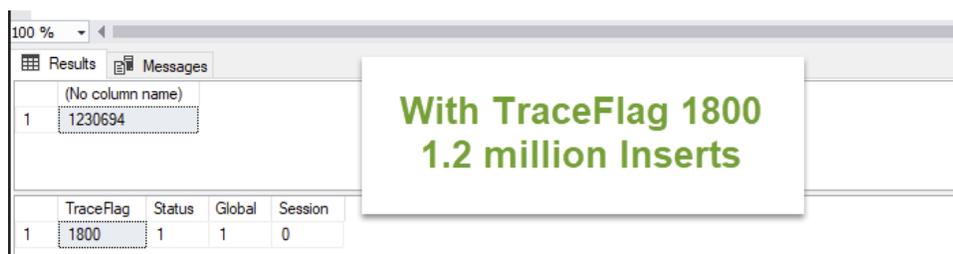
That's it.

But, was all that worth the effort? Let's take a look at *before* and *after*.

Here is a sample insert job we ran on a server without the trace flag:



Here is the same job, with trace flag 1800 enabled on the same server:



Using a tool like Procmon, you can get a peek into the root cause of the performance bottleneck and see how it resolves after the TraceFlag trace flag is added.

Here is our IO and block size pattern prior to enabling -T1800 (the yellow highlight):

```

4:40:13.074222 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,252,928; Length: 4,608; I/O Flags: Non-
4:40:13.079304 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,257,536; Length: 5,144; I/O Flags: Non-
4:40:13.083065 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,263,680; Length: 4,608; I/O Flags: Non-
4:40:13.088611 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,268,288; Length: 5,632; I/O Flags: Non-
4:40:13.091188 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,273,920; Length: 4,608; I/O Flags: Non-
4:40:13.094451 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,279,520; Length: 5,632; I/O Flags: Non-
4:40:13.096122 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,284,160; Length: 4,608; I/O Flags: Non-
4:40:13.098419 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,288,768; Length: 5,632; I/O Flags: Non-
4:40:13.094512 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,294,400; Length: 4,608; I/O Flags: Non-
4:40:13.093530 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,299,008; Length: 5,632; I/O Flags: Non-
4:40:13.091008 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,304,640; Length: 4,608; I/O Flags: Non-
4:40:13.092964 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,309,248; Length: 5,632; I/O Flags: Non-
4:40:13.0914625 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 704,314,880; Length: 4,608; I/O Flags: Non-
4:40:13.092982 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 703,037,440; Length: 4,608; I/O Flags: Non-
4:40:13.092648 PM sqservr.exe 6884 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 703,042,048; Length: 5,632; I/O Flags: Non-
    
```

Here is how it improved and stabilized with -T1800:

```

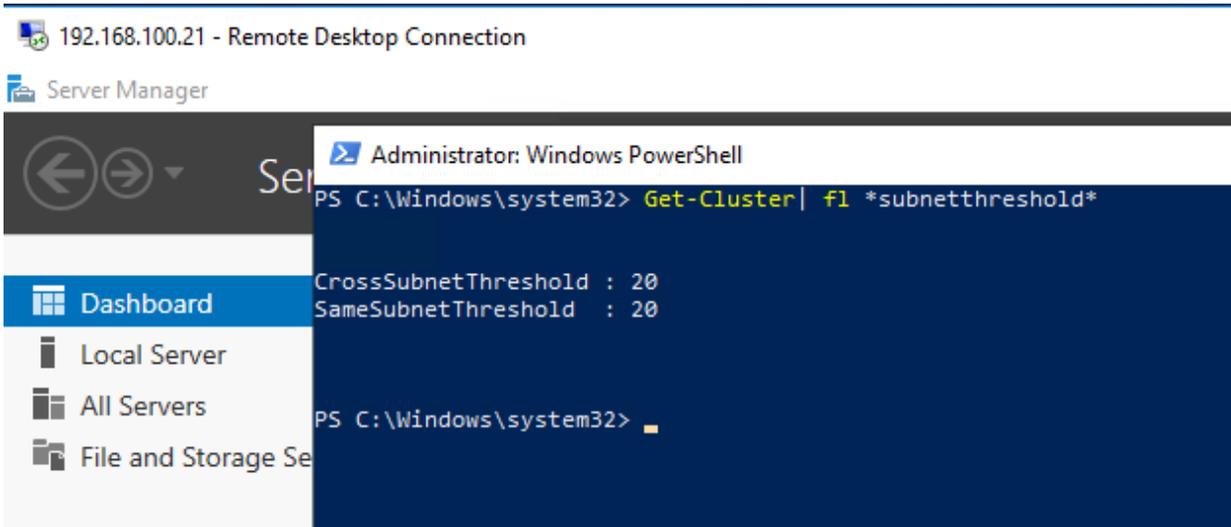
4:40:07.066802 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,547,968; Length: 8,192; I/O Flags: Non-cached
4:40:07.066139 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,556,160; Length: 8,192; I/O Flags: Non-cached
4:40:07.067033 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,564,352; Length: 8,192; I/O Flags: Non-cached
4:40:07.067813 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,572,544; Length: 8,192; I/O Flags: Non-cached
4:40:07.068639 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,580,736; Length: 8,192; I/O Flags: Non-cached
4:40:07.069365 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,588,928; Length: 8,192; I/O Flags: Non-cached
4:40:07.069233 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,597,120; Length: 8,192; I/O Flags: Non-cached
4:40:07.069495 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,605,312; Length: 8,192; I/O Flags: Non-cached
4:40:07.069835 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,613,504; Length: 8,192; I/O Flags: Non-cached
4:40:07.070425 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,621,696; Length: 8,192; I/O Flags: Non-cached
4:40:07.070816 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,629,888; Length: 8,192; I/O Flags: Non-cached
4:40:07.0712911 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,638,080; Length: 8,192; I/O Flags: Non-cached
4:40:07.0717344 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,646,272; Length: 8,192; I/O Flags: Non-cached
4:40:07.072424 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,654,464; Length: 8,192; I/O Flags: Non-cached
4:40:07.072618 PM sqservr.exe 9072 WriteFile L:\Program Files\Microsoft SQL Server\MSQL15\MSSQLSER\ER\MSSQL\Data\LoadDB_Log.ldf SUCCESS Offset: 30,662,656; Length: 8,192; I/O Flags: Non-cached
    
```

Adjust WSFC Cluster Heartbeat Thresholds:

One of the standard configuration checks you'd want to perform after completing your clustered SQL Server installation on AVS is the **Cluster Heartbeat Threshold** setting. You can do this by issuing the following PowerShell command:

```
Get-Cluster | fl *subnetthreshold*
```

You should get an output similar to that shown below:



Why is this necessary? It's a long story, but long documented in many places, including in various *VMware's Best Practices for Virtualizing Microsoft SQL Server* guides. Here's a brief overview:

Clustered Windows nodes exchange periodic heartbeats with each other. This is how they verify that their peers are available, responsive and still participating in the cluster. They send these heartbeats every second. Historically, if one node fails to respond to five (for nodes in the same network subnet) or

ten (for nodes in different subnets) consecutive heartbeat probes, its peers will consider it unresponsive and will be compelled to try and take over whatever clustered resource is hosted on the non-responsive node. This means that a node has a grace period of 5 or 10 seconds to respond to heartbeat probes before it's deemed to no longer be fit to own a clustered resource.

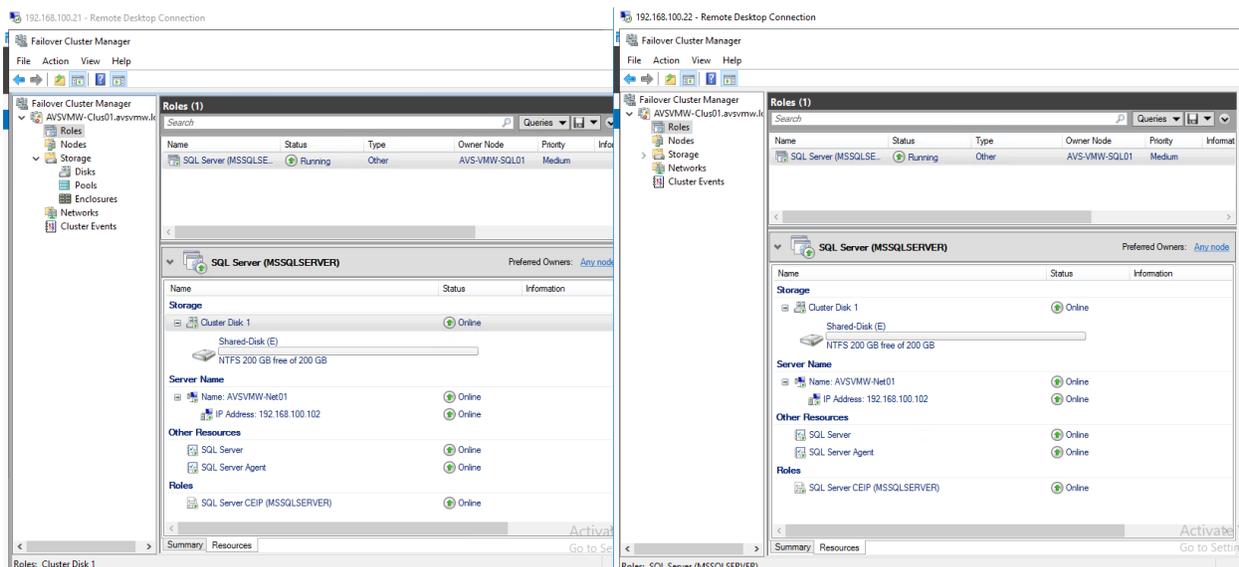
These numbers were considered good enough until freezing (or quiescing) a Windows OS instance became a feature of Windows and a standard practice for many operations (e.g., in-Guest backup). When a clustered Windows OS instance is quiesced for longer than the tolerable threshold, unexpected clustered resource failover events begin to occur.

To help mitigate this potential for unintended failovers, Microsoft recommended that customers adjust the threshold. Microsoft has now since increased the default to 20, for both **SameSubnetThreshold** and **CrossSubnetThreshold** for more modern Windows versions.

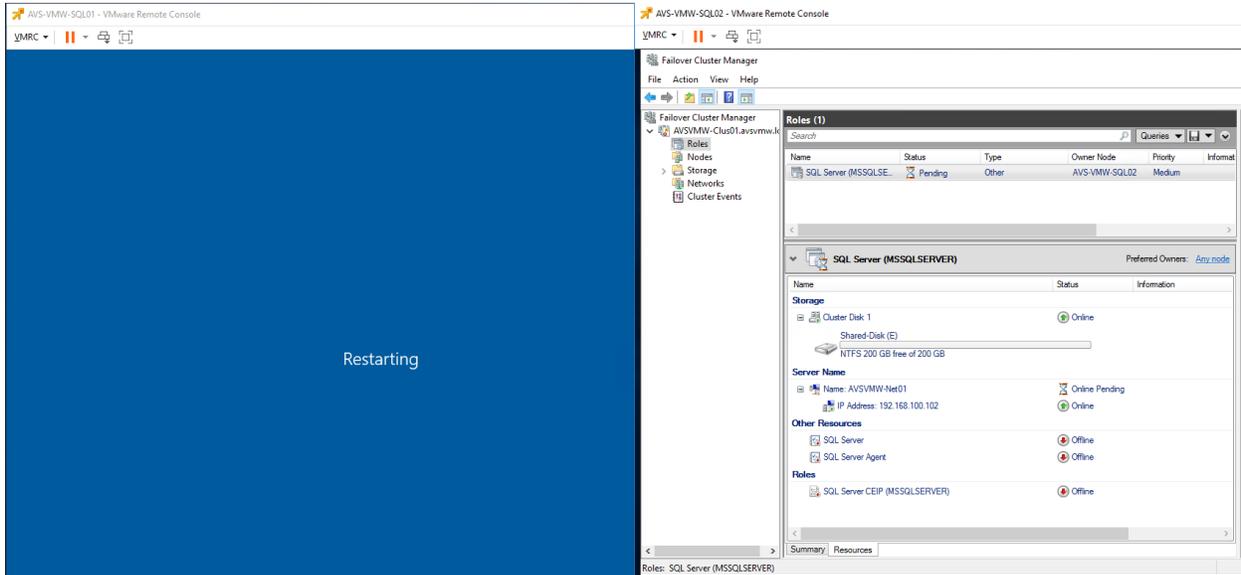
In AVS (indeed, in all VMware vSphere platforms), when you perform a vMotion operation on a Windows VM, the VM will be quiesced at a point during the process. Although vMotion tries to exist the quiesced state as quickly as possible, some external factors could cause the process to take longer than 5 or 10 seconds in some versions of Windows. For clustered workloads, this is not desirable. This is why you must check and adjust the thresholds. If you need to adjust the threshold, use the following PowerShell commands:

- *(get-cluster).SameSubnetThreshold = 20*
- *(get-cluster).CrossSubnetThreshold = 20*
- *(get-cluster).RouteHistoryLength = 40*

We now have a 2-node SQL Server instance configured in FCI clustering mode in our AVS. Let see what it looks like.



Node 1 (SQL01) currently owns the resource. Let's see what happens if SQL01 were to suddenly become unavailable.



As expected, the resource automatically fails over to node 2 (SQL02). Nothing to see here.

Installing and Configuring Microsoft SQL Server Always On Availability Group on AVS

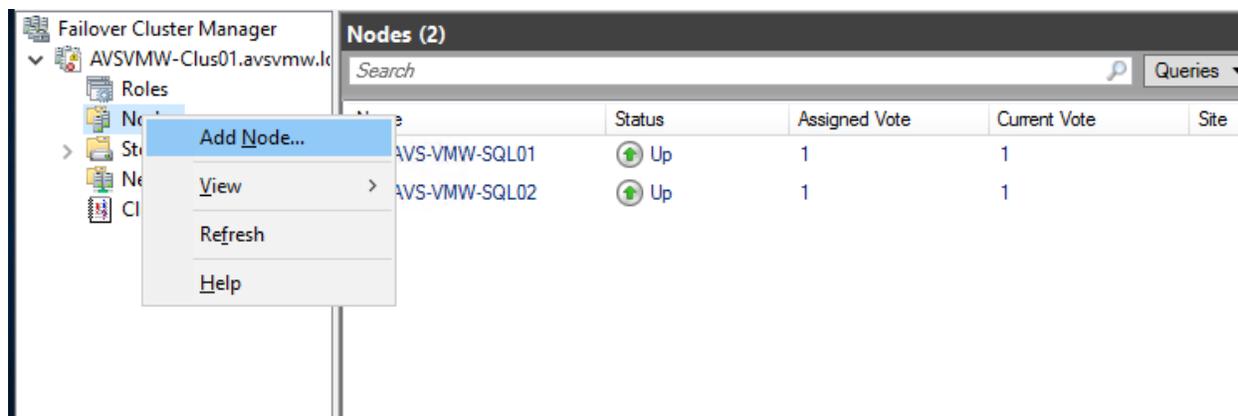
Unlike FCI, installing and configuring Microsoft SQL Server Always On availability group (AG) on Azure VMware Solution is a much more simplified process. This is chiefly because shared-disk SCSI3 PR command and disk ownership arbitration among two or more clustered VMs is the major challenge with successfully configuring Windows Server Failover Clustering in a VMware vSphere-based platform such as AVS. In AG, the nodes do not share disks or resources. Each node has ownership of its own set of resources.

We will now describe the tasks and activities involved in successfully creating and configuring a sample Availability Group cluster in our AVS environment. We will attempt to make this section more interesting by adding the AG configuration on top of our existing FCI configuration. This is a fairly common configuration in production environment and it's our hope that, in using this example, administrators will get a more detailed picture of the various configuration options available in clustering Microsoft SQL Server on AVS.

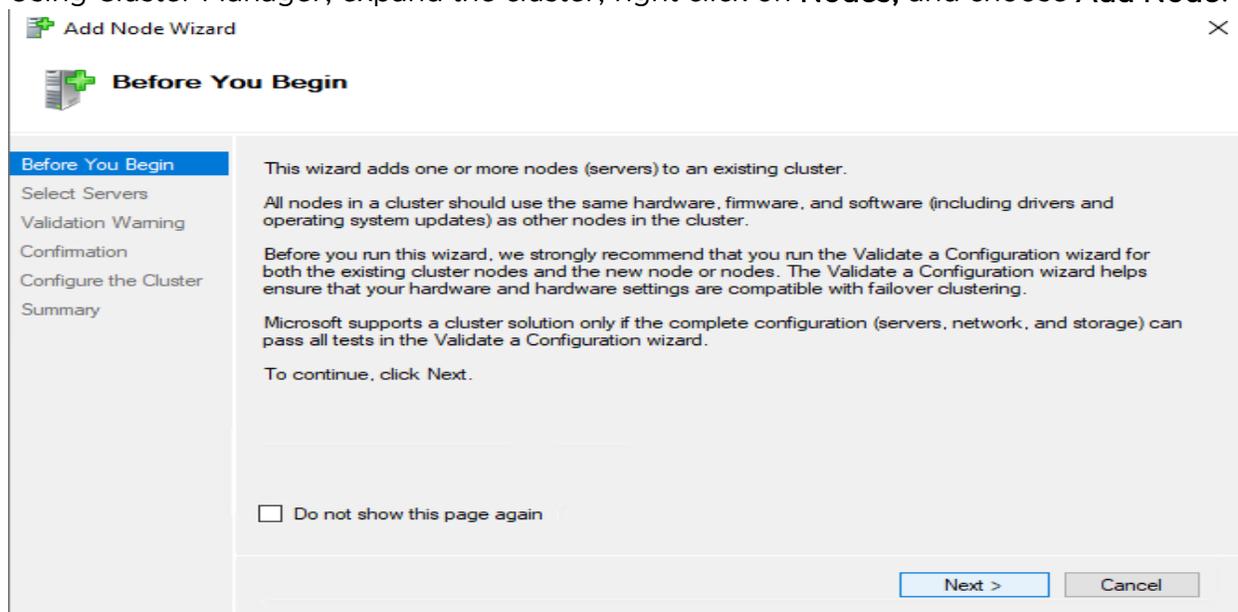
NOTE: As usual, we expect you are familiar with the administrative tasks and preparations required for some of the tasks we show in this example. We will not be describing this in detail, except to the extent that the tasks are either unique to AVS or help paint a clearer picture.

For example, we have added a separate, non-shared VMDK to SQL01 which will contain the data and files for the Database that we're going to configure in AG. We have also added a similar (but separate and unshared) VMDK to SQL03 for the same purpose.

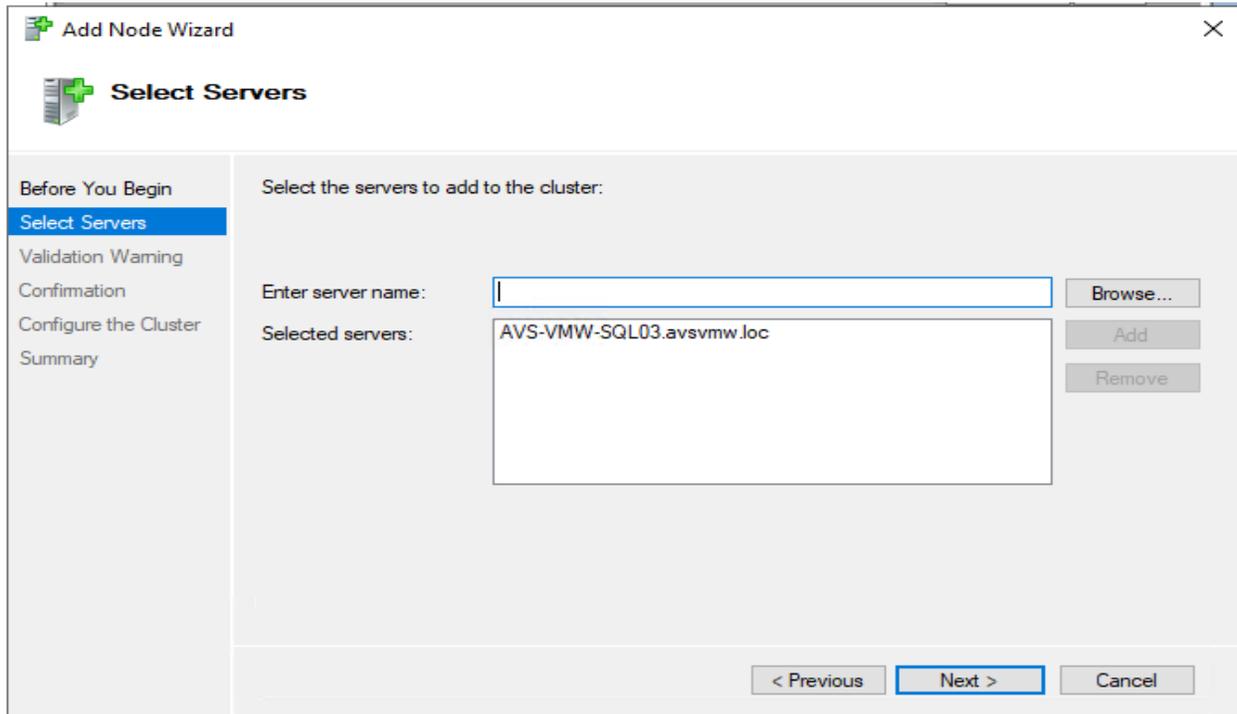
We will now proceed by adding a third MS SQL Server node (i.e., AVS-VMW-SQL03, hereafter simply referred to as SQL03) to the existing WSFC cluster we created in the previous exercise.



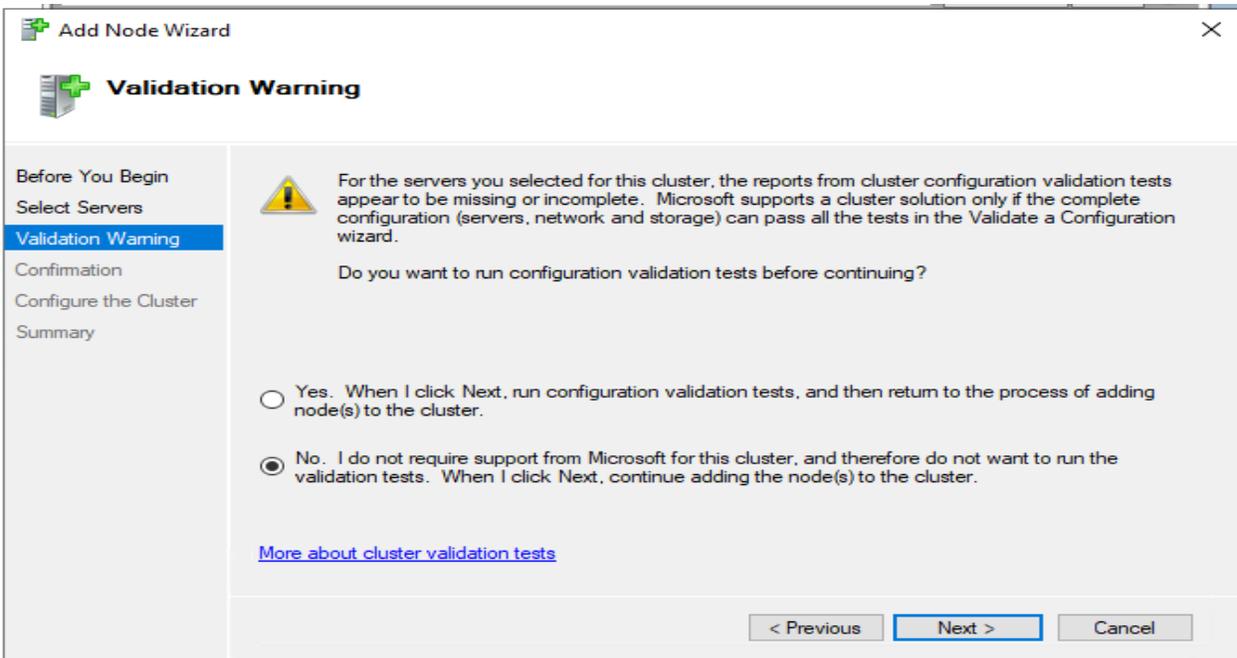
- Using Cluster Manager, expand the cluster, right click on **Nodes**, and choose **Add Node**.



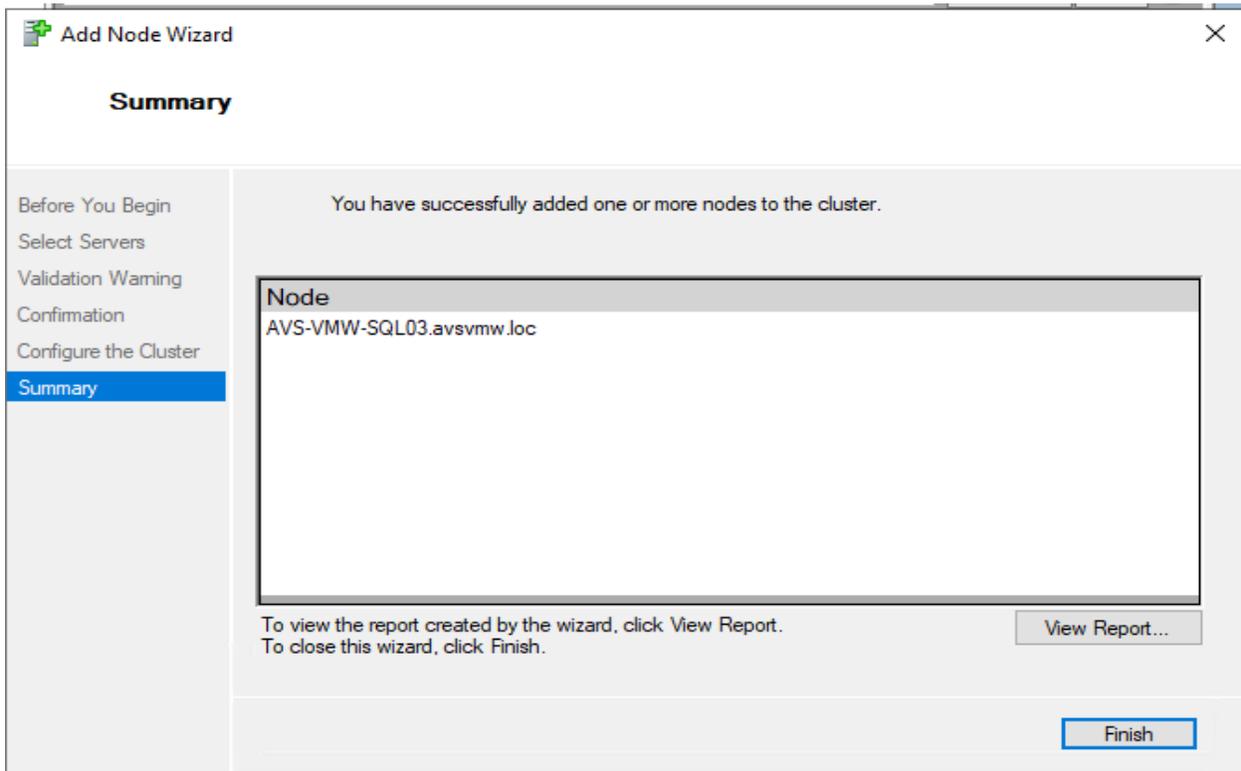
- Click **Next**.



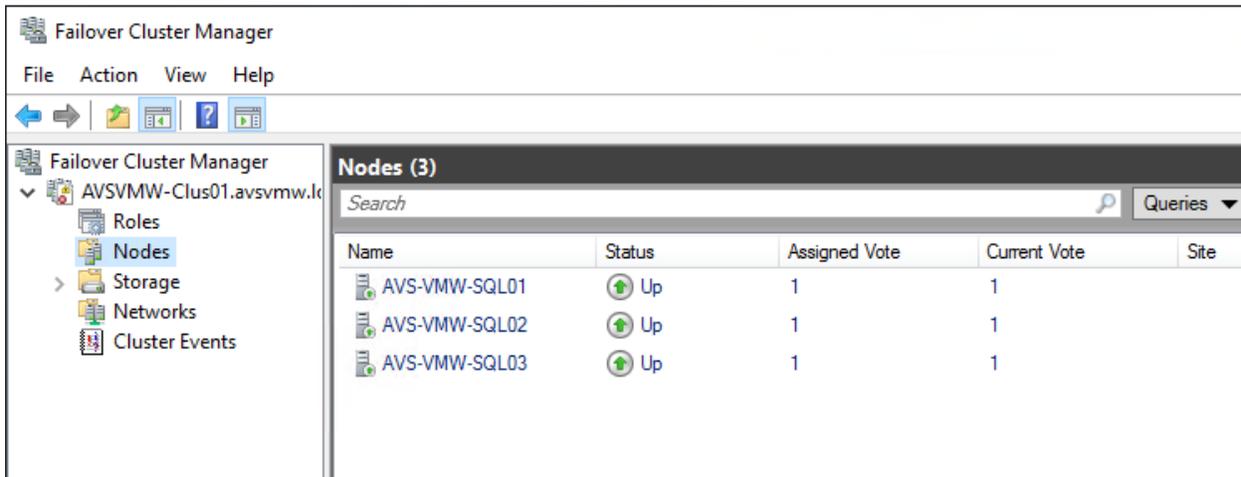
- We'll add SQL03 and click **Next**.



- We've already run the validation test and passed, so, let's skip it this time.



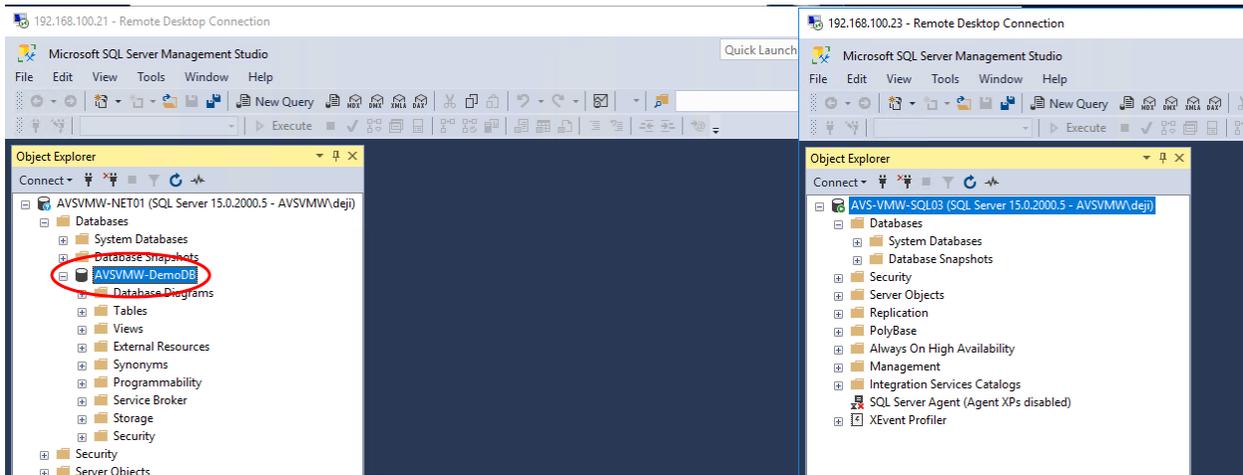
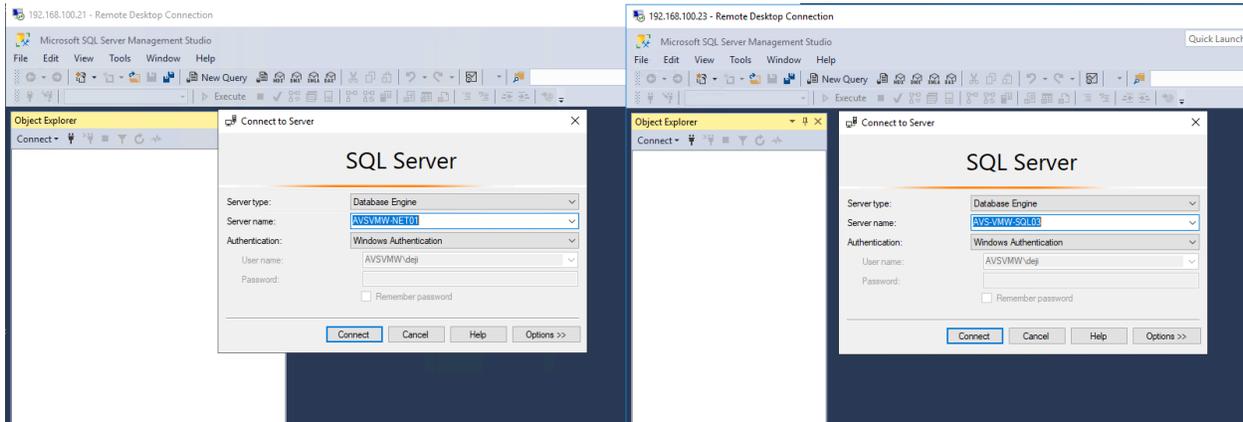
- One more click and we are done with adding SQL03 to our WSFC cluster. Here is what we have now:



Now, we move onto configuring a Database for Always On AG.

- Let's connect to the SQL instances on SQL01 and SQL03.

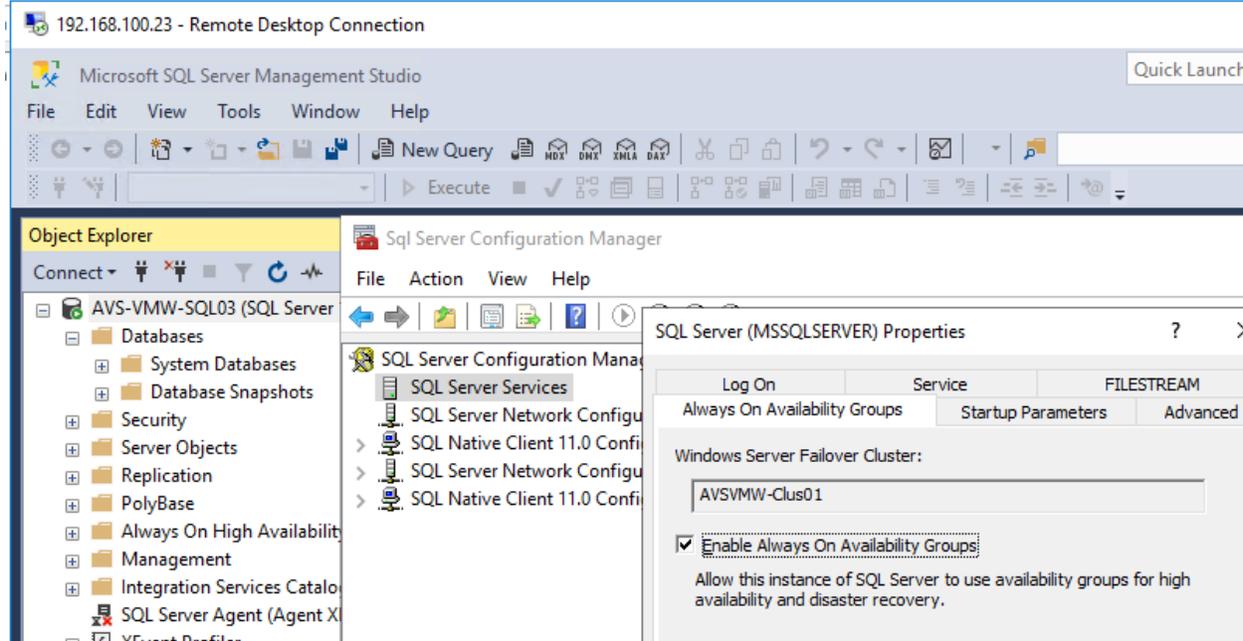
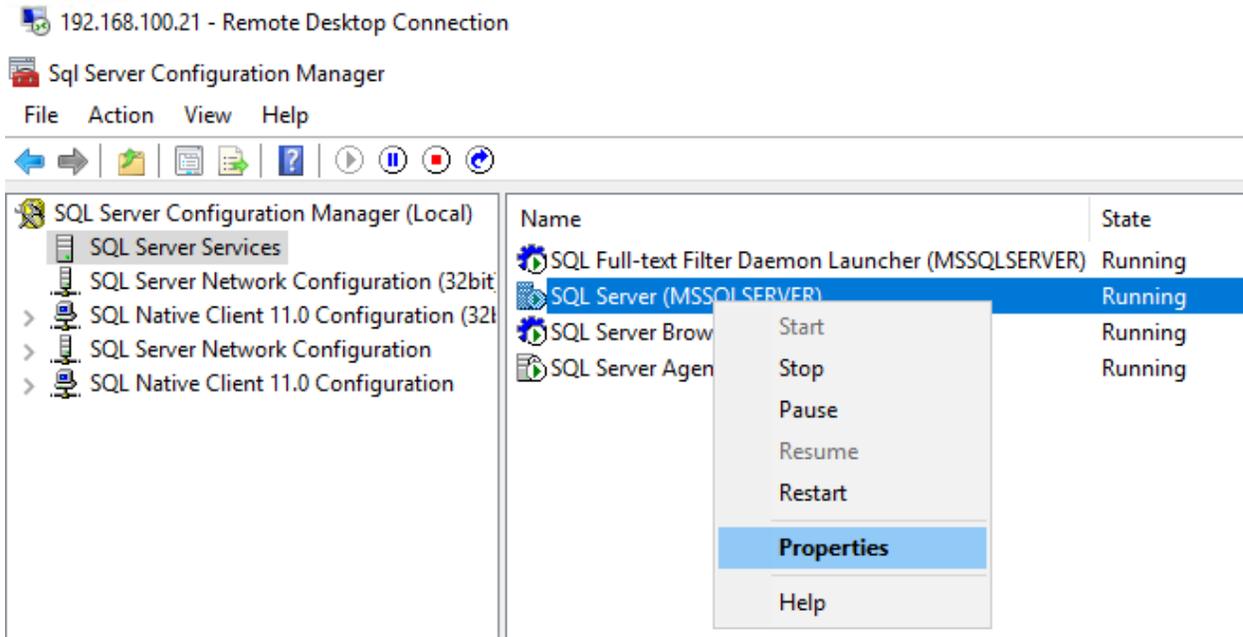
Successfully Virtualizing Microsoft SQL Server for High Availability on Azure VMware Solutions



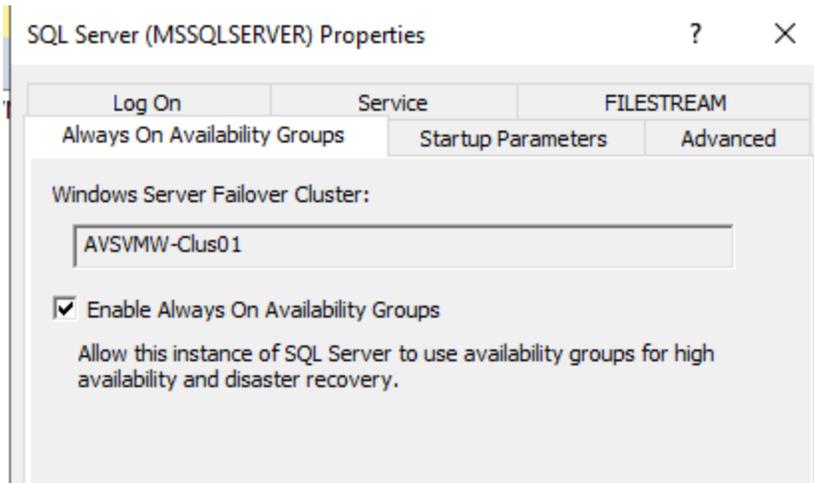
There is currently one Database on SQL01. This is the Database in our default instance which is configured in FCI and paired with SQL02. There is currently no Database on SQL03 because this node is not a part of the FCI configuration.

We will now configure our target nodes to participate in Always On Availability Group configuration.

- On each node, click the **SQL Server Services** properties in SQL Server Configuration Manager.



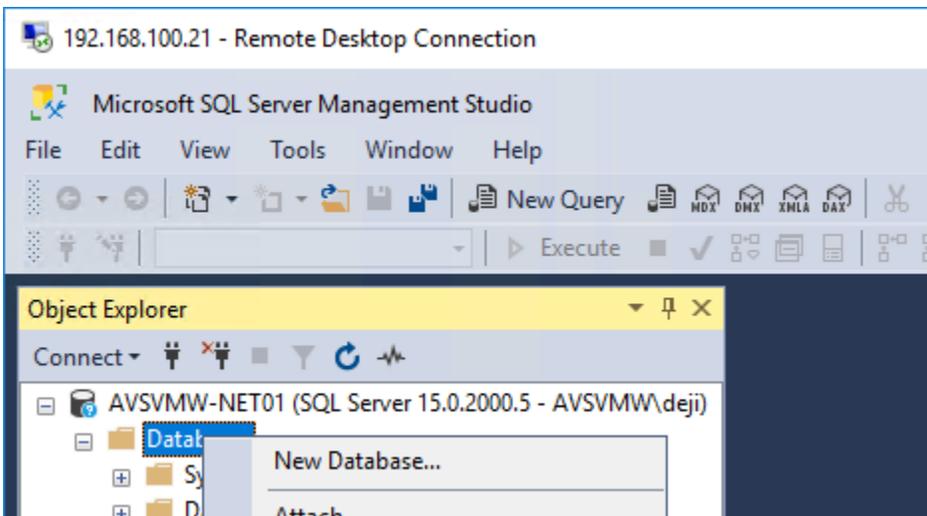
- Check the **Enable Always On Availability Groups** box on the **Always On Availability Groups** tab.

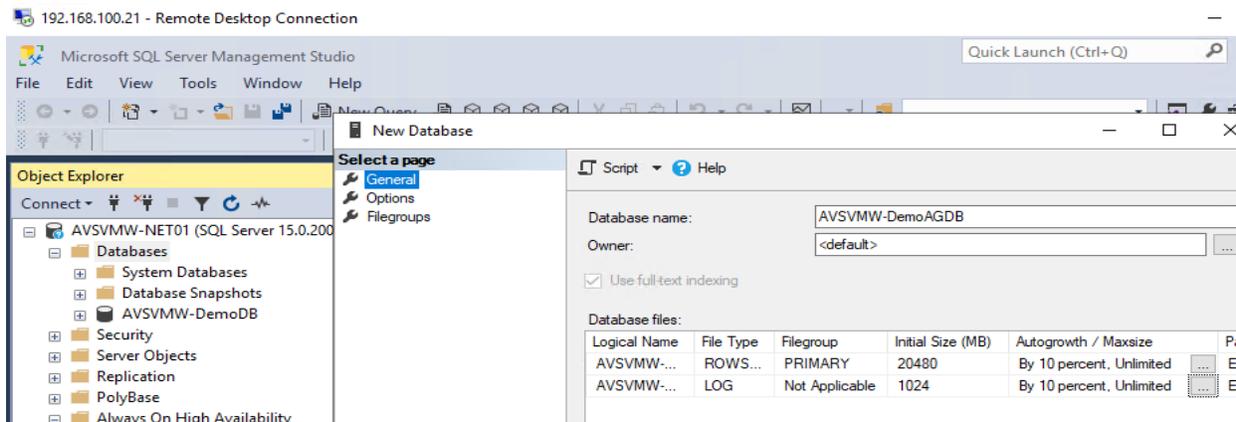


- Click **OK** to confirm that we know what we are doing.

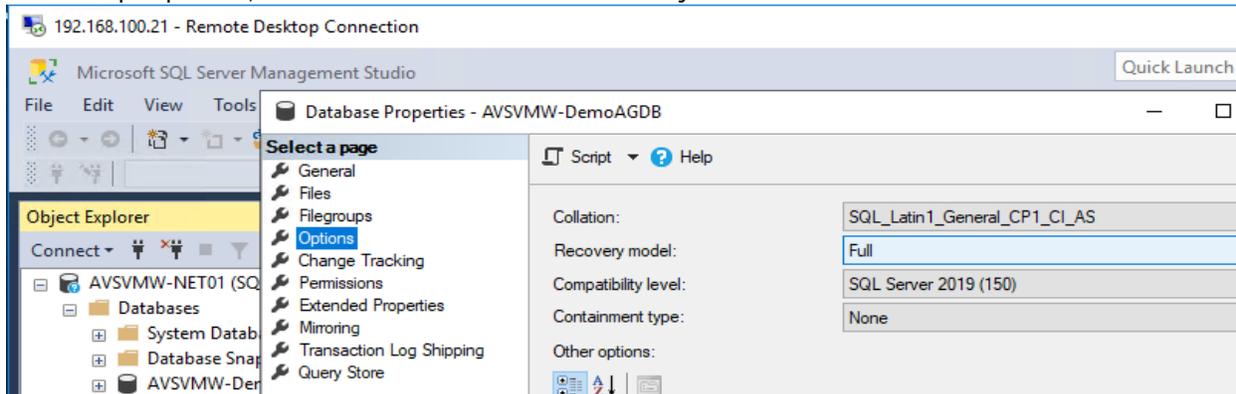


- Now, we create the Database we're going to use for AG on SQL01.

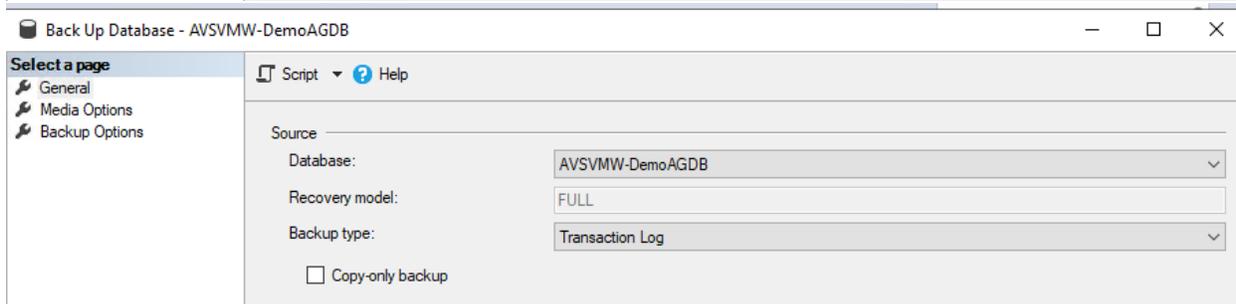
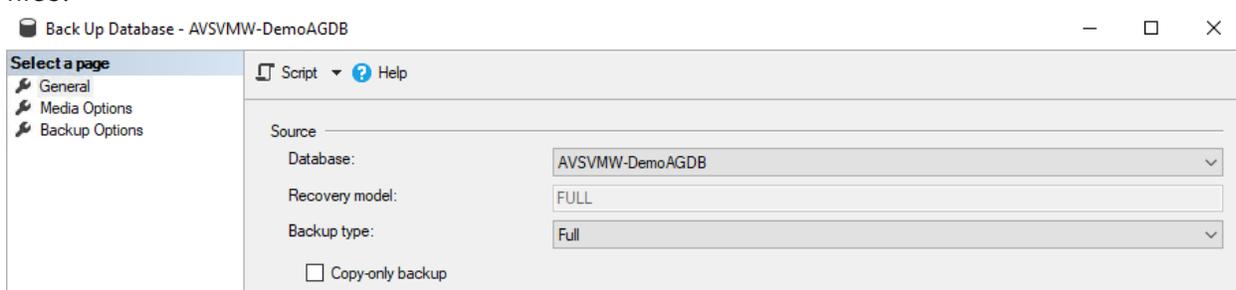


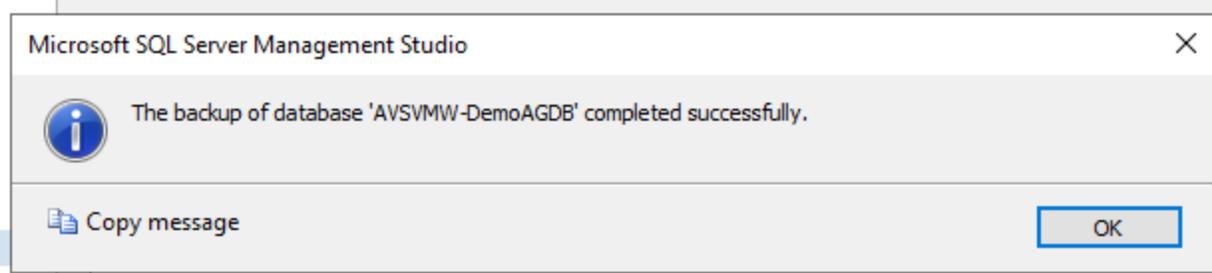


- For our purposes, we will set it to “Full” Recovery model.



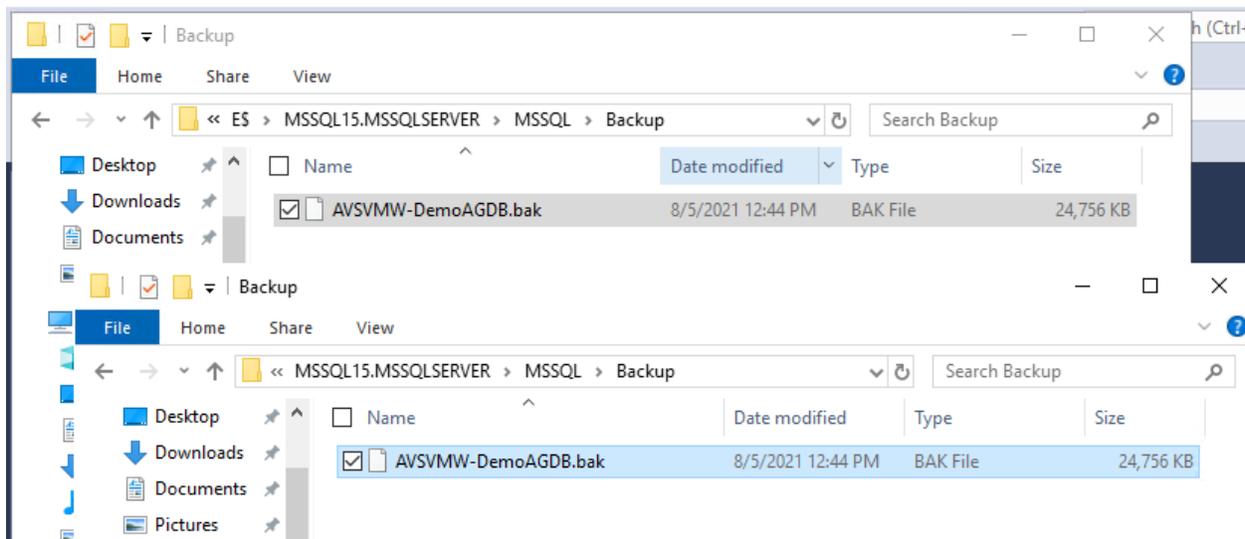
- To help speed up the process, we will take a backup of both the Database and Transaction Logs files.



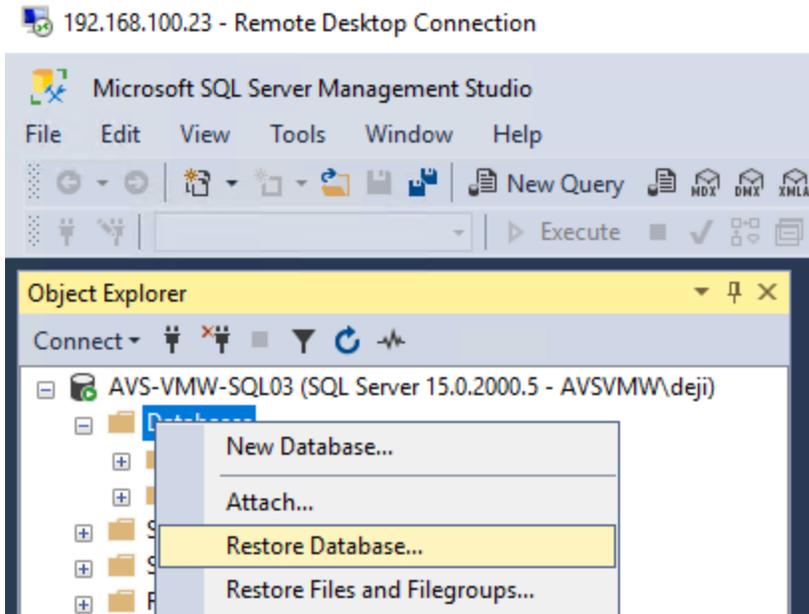


- We'll copy the backup file manually from SQL01 to SQL03.

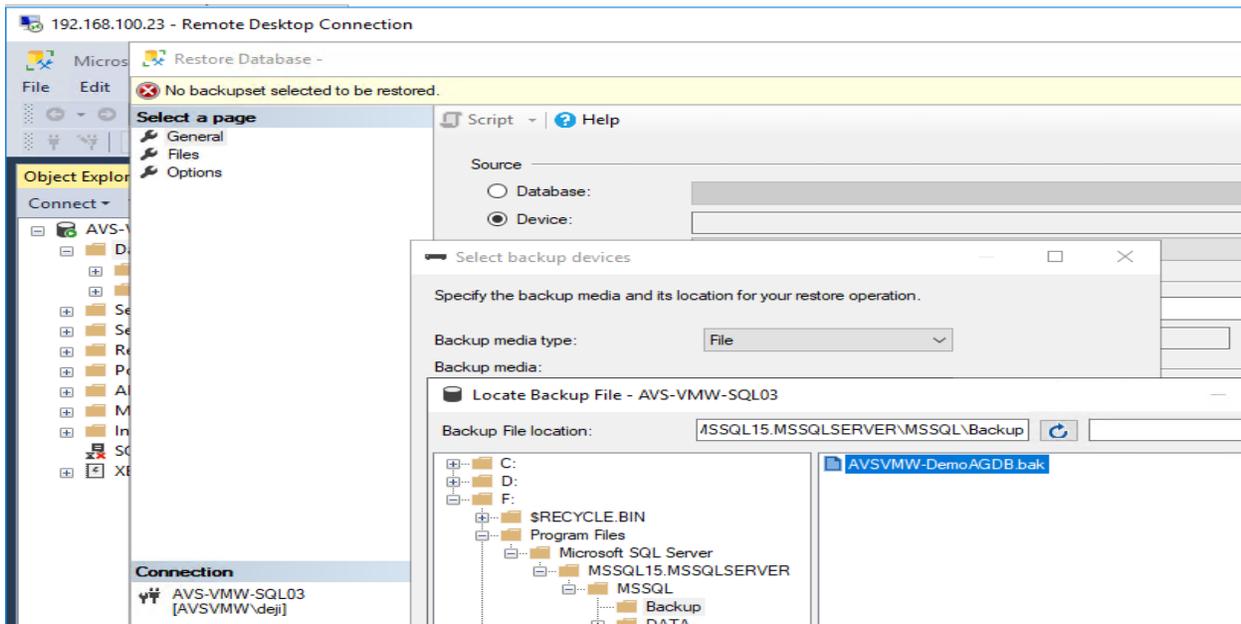
192.168.100.23 - Remote Desktop Connection



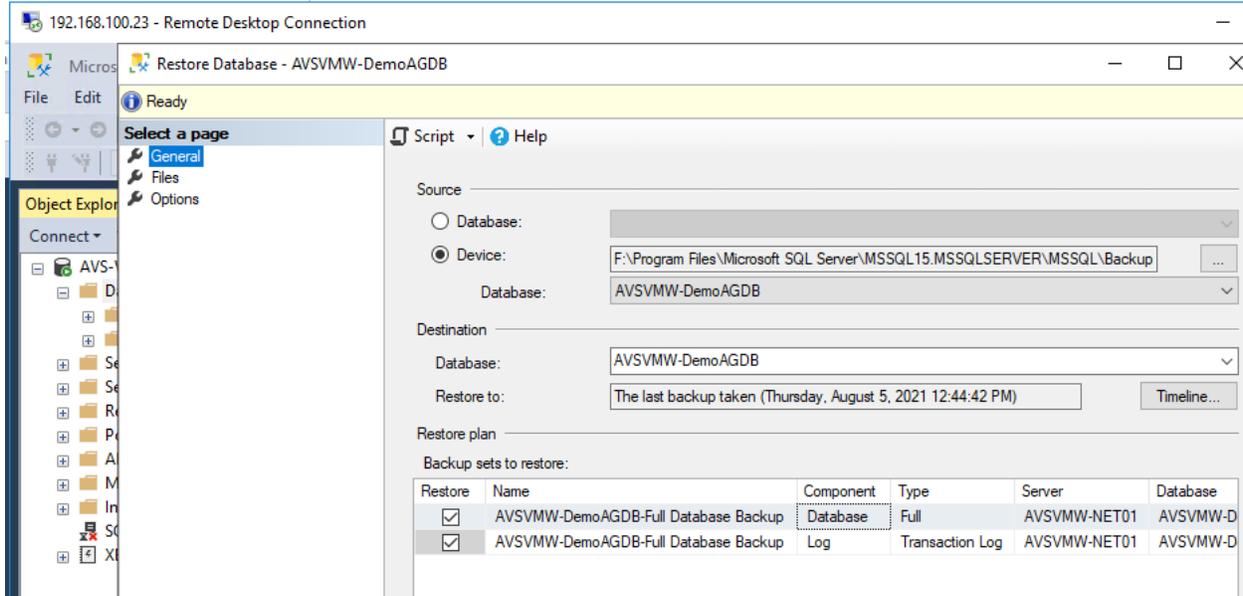
- Next, we restore the backed-up database to SQL03. This helps us avoid lengthy re-seed and synchronization, especially for large and highly active Databases



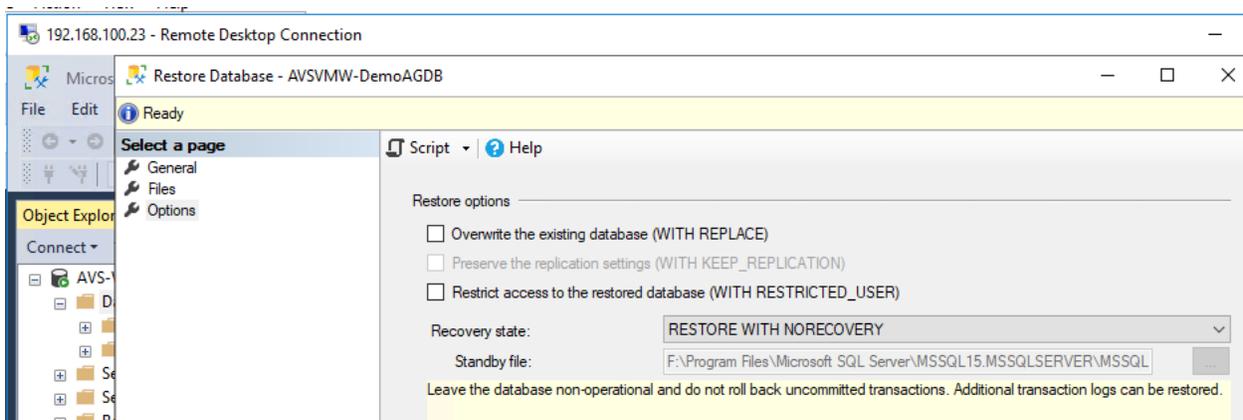
- We select our .bak file.



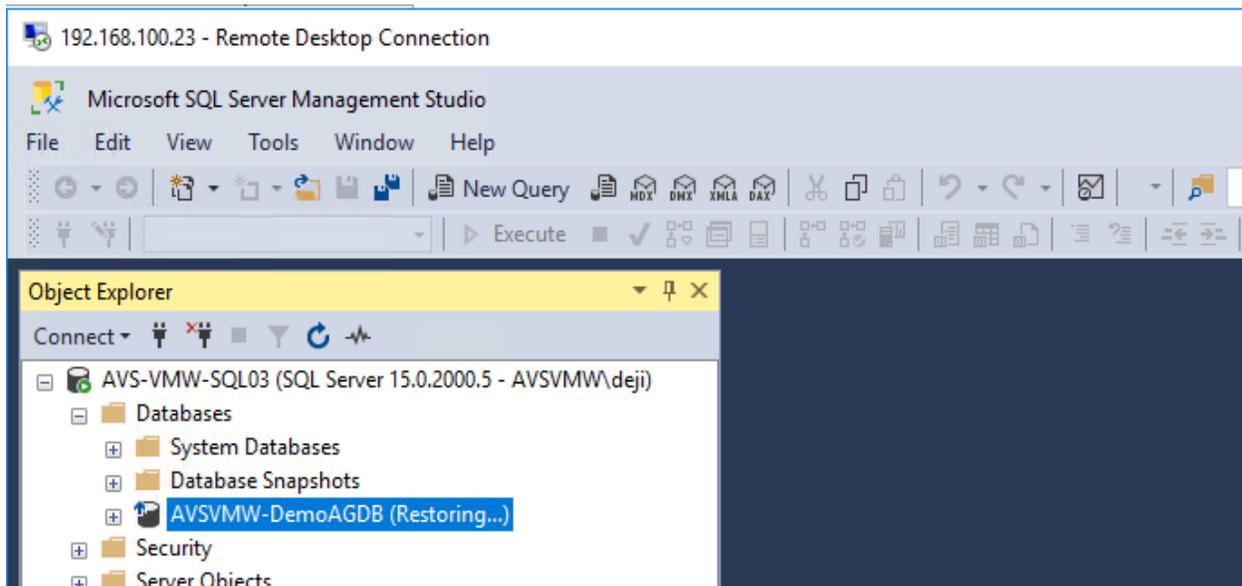
- We verify that all looks good.



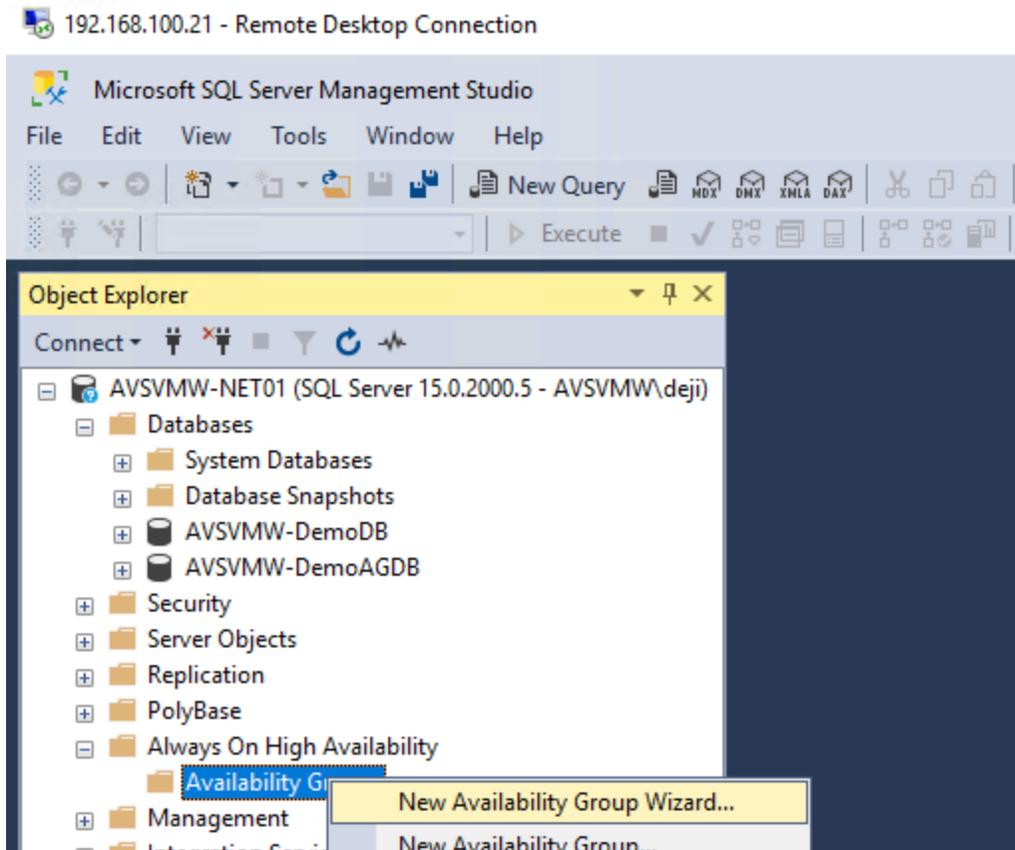
- We choose the **Restore with NoRecovery** option and *OK* the choices.



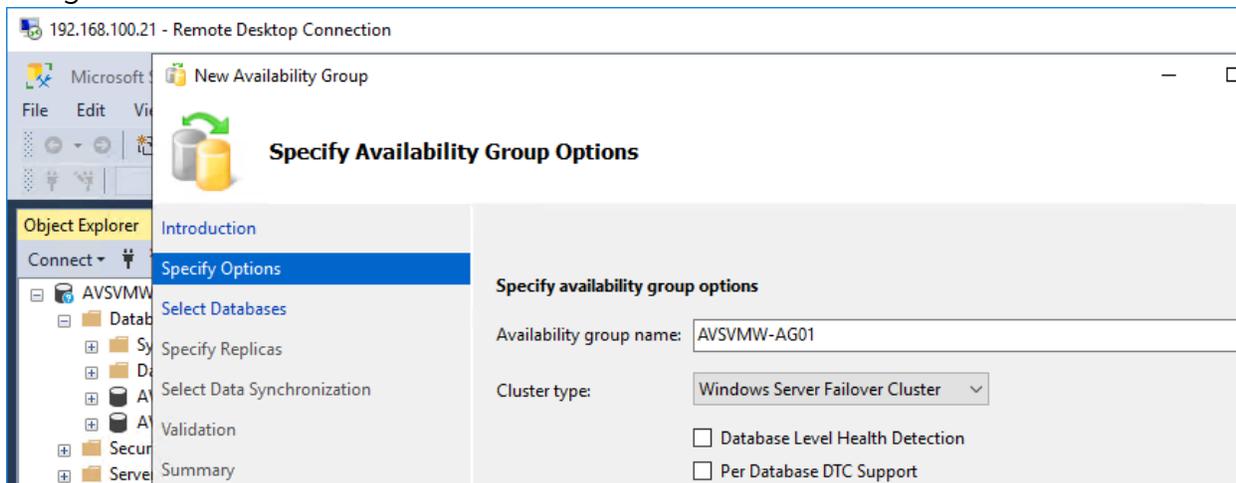
- Depending on the size of the Database, the restore process may take some time to complete.



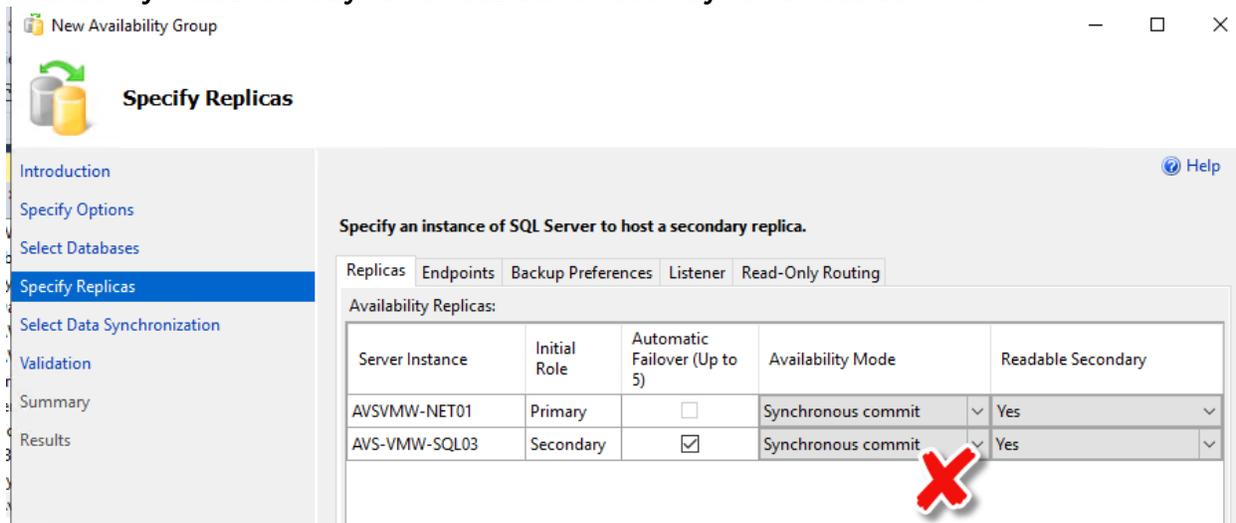
- Now that we have a stub copy of the Database on the second node, we are ready to configure it for high availability, using the Microsoft SQL Server Availability Group option.
- From the **Always On Availability Group** menu on the Active node (SQL01), select **New Availability Group Wizard**.



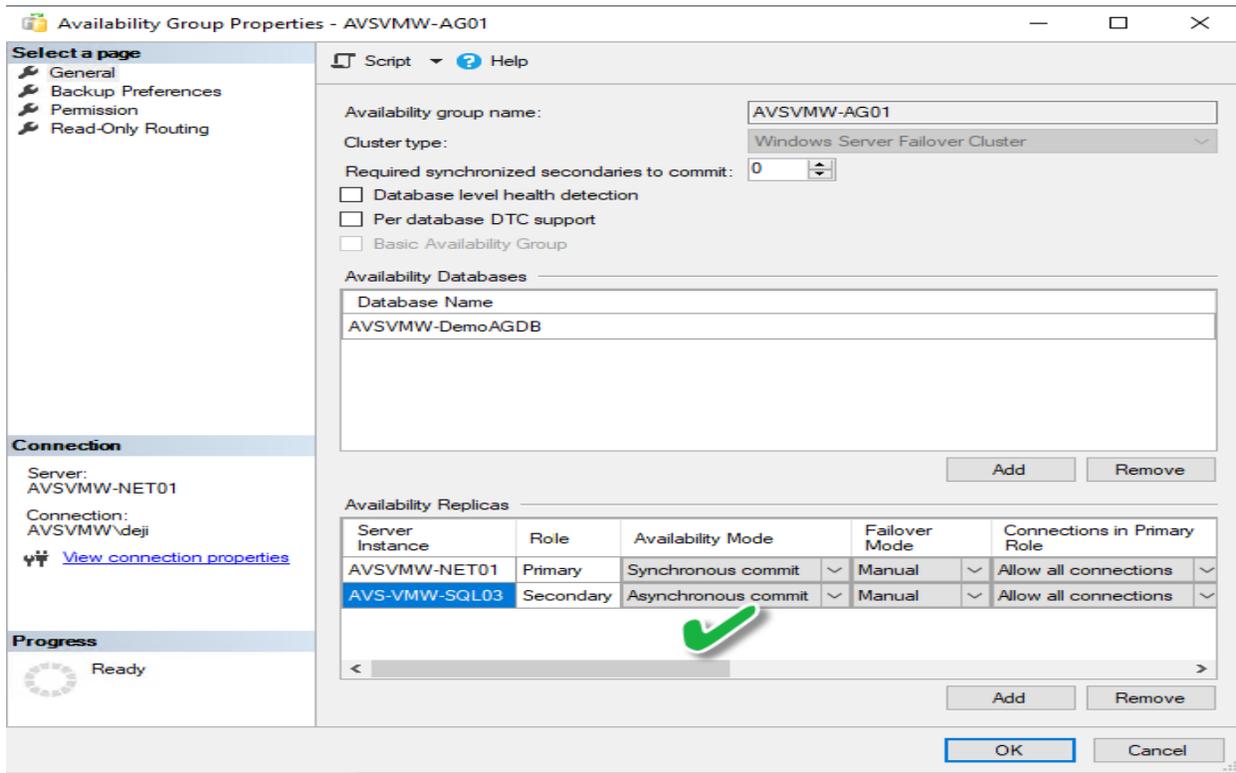
- We give the AG a name.



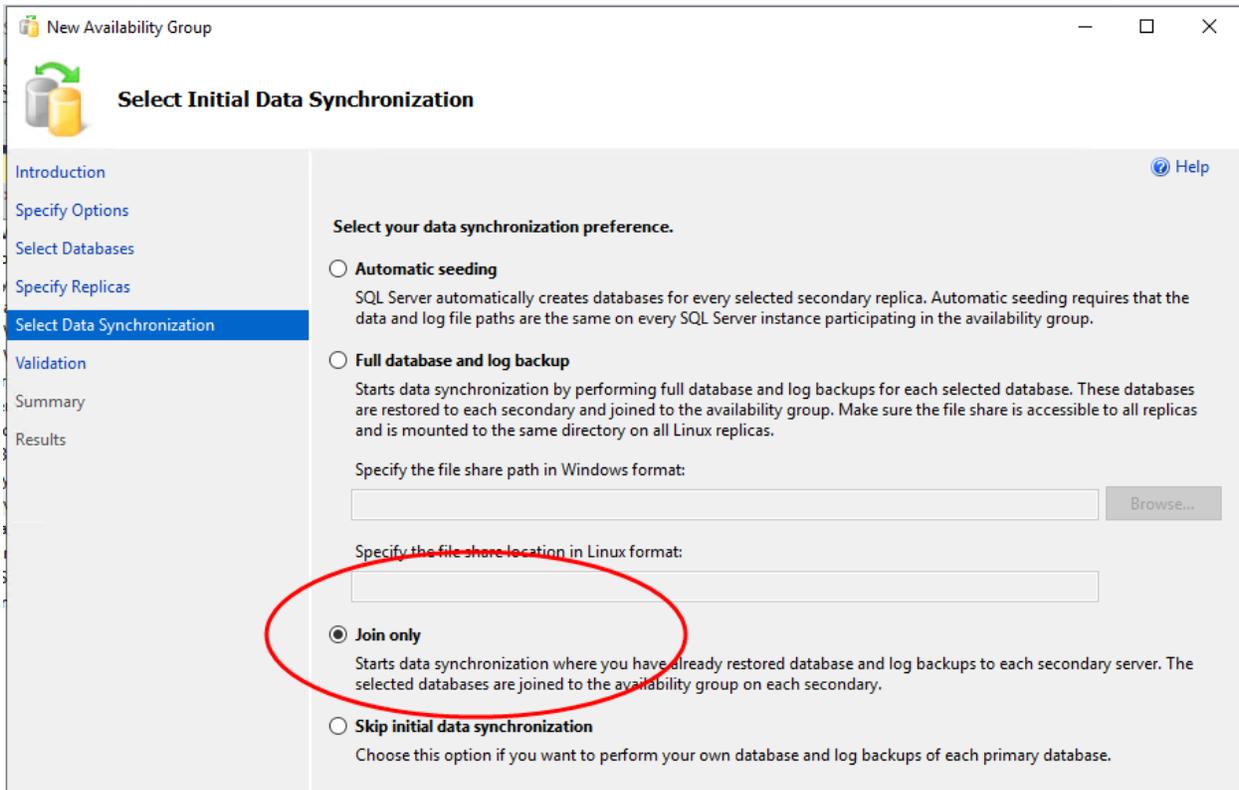
- We want our Replica copy of the AG to be in Asynchronous mode, so be sure to toggle the **Availability Mode** from **Synchronous Commit** to **Asynchronous Commit**.



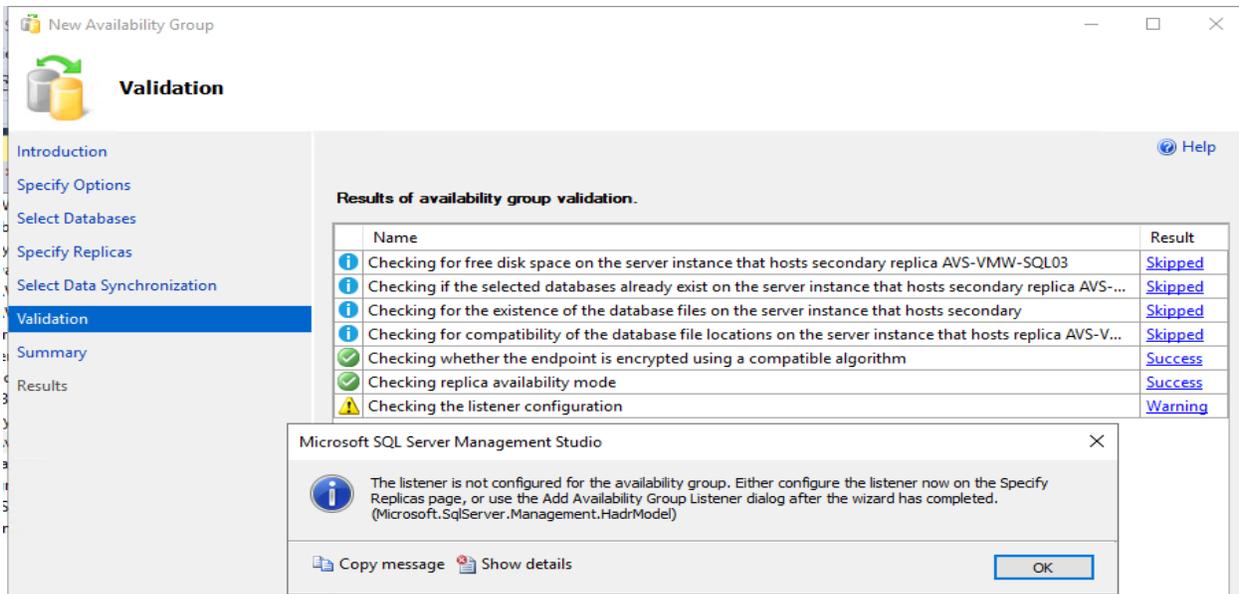
- Here is what we want our **Availability Mode** to look like for our AG Database configuration.



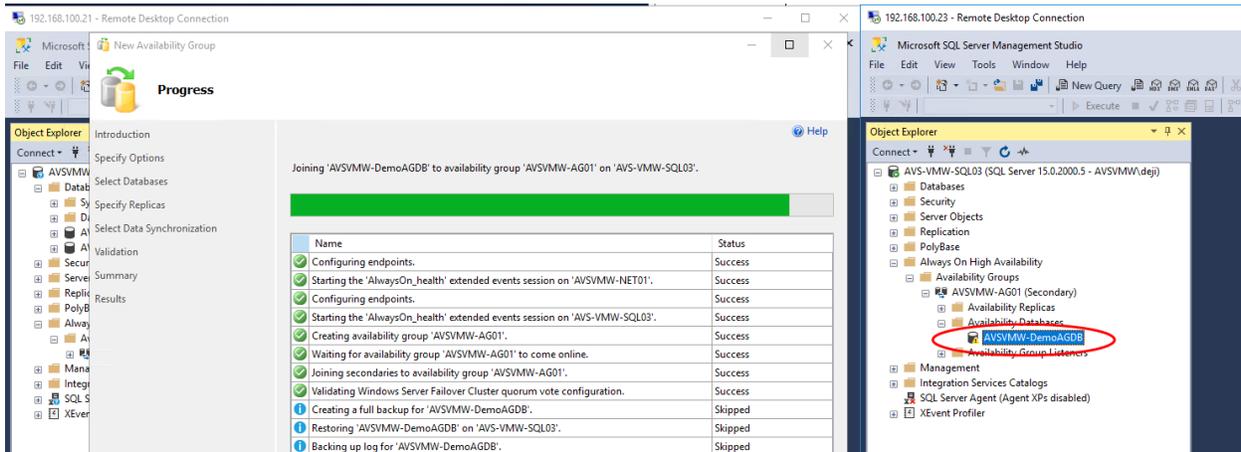
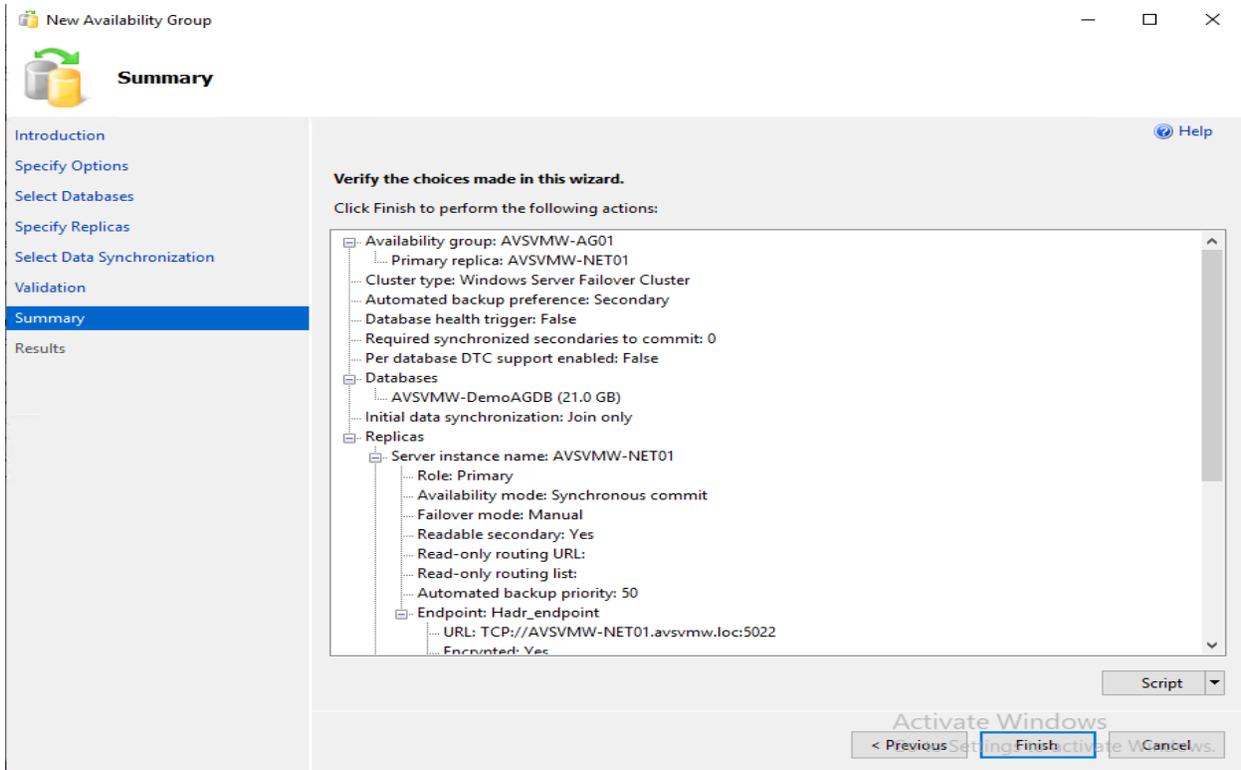
- Choose the **Join Only** option – remember, we have already copied the Database manually to the Secondary node.



- We skipped the option to configure the Listener for now. We will come back to that later.

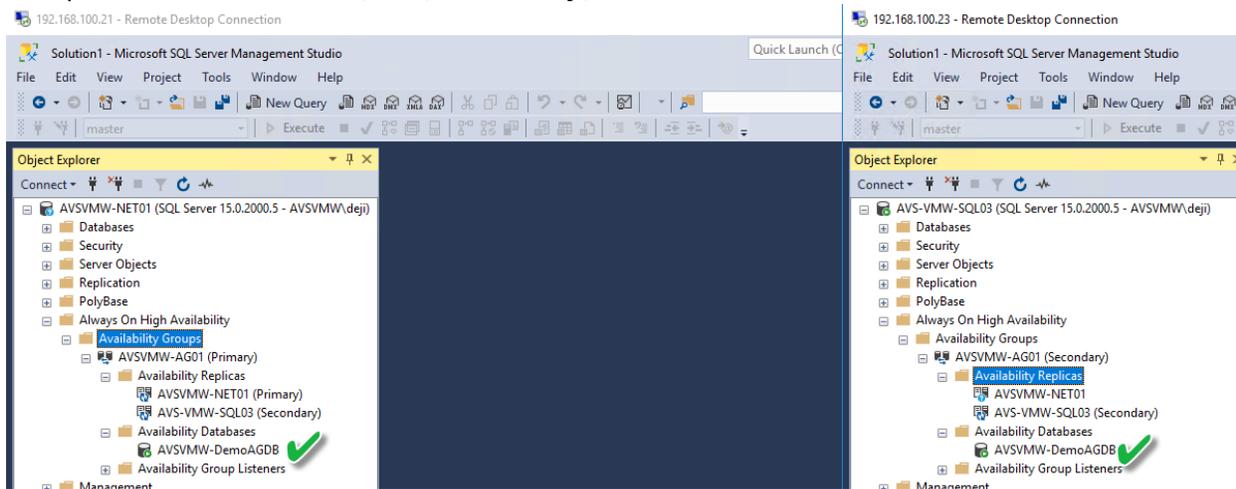


- We're almost done. Let's click **Finish**.



- The wizard skips several of the steps because we've already completed them in previous step.
- Note the state of the Database on SQL03. We're cooking with gas now.

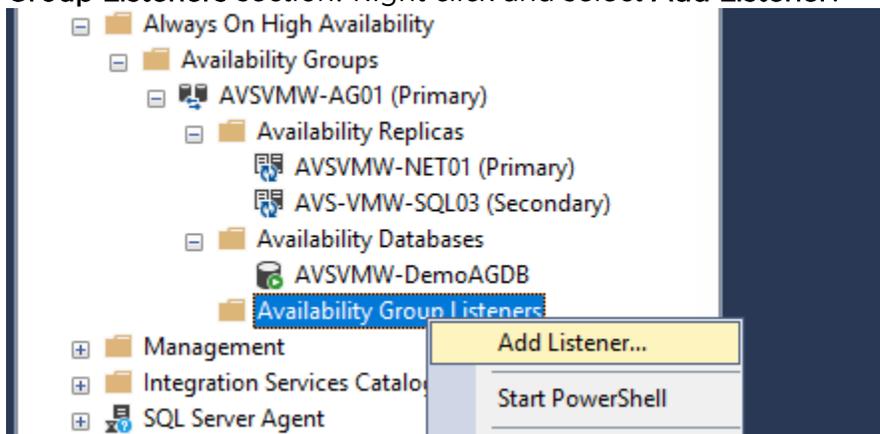
- The process takes a while, but, eventually, here is the result of our effort.

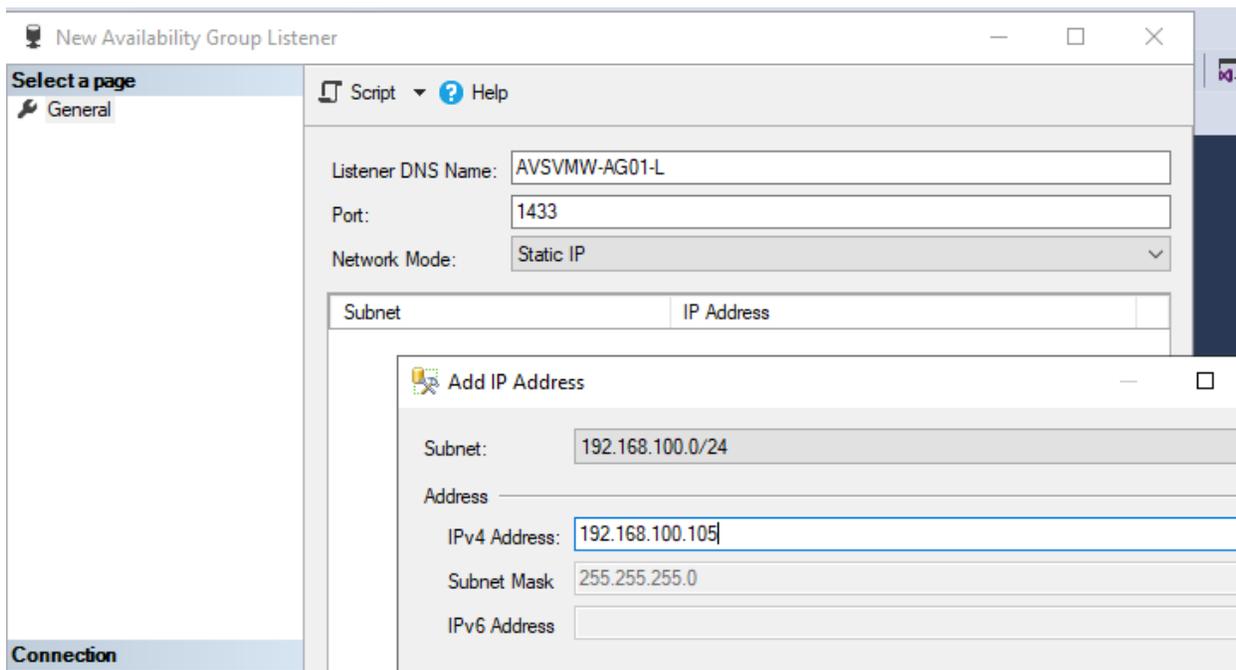


Now, we are ready to configure our Listener.

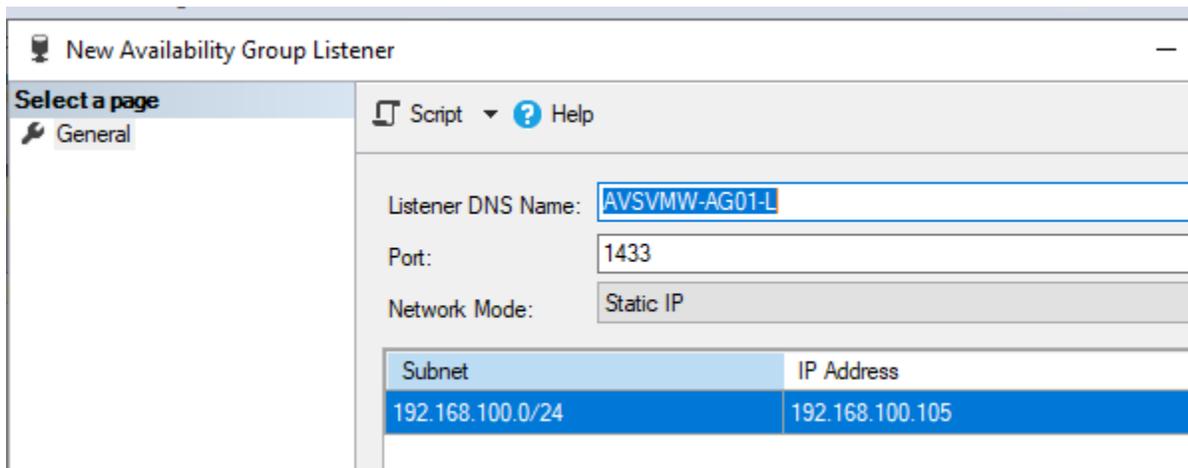
NOTE: In our simplified configuration, we have a stretched/extended/flat network segment between our on-premises SQL Servers and AVS. We're using the same subnet for both sides. In this configuration, we need only one Listener. If this were routed or disjointed segments, we would have needed to create a Listener for each segment where we have the clustered VMs.

- In SQL Server Manager, expand **Always On High Availability** and navigate to the **Availability Group Listeners** section. Right click and select **Add Listener**.

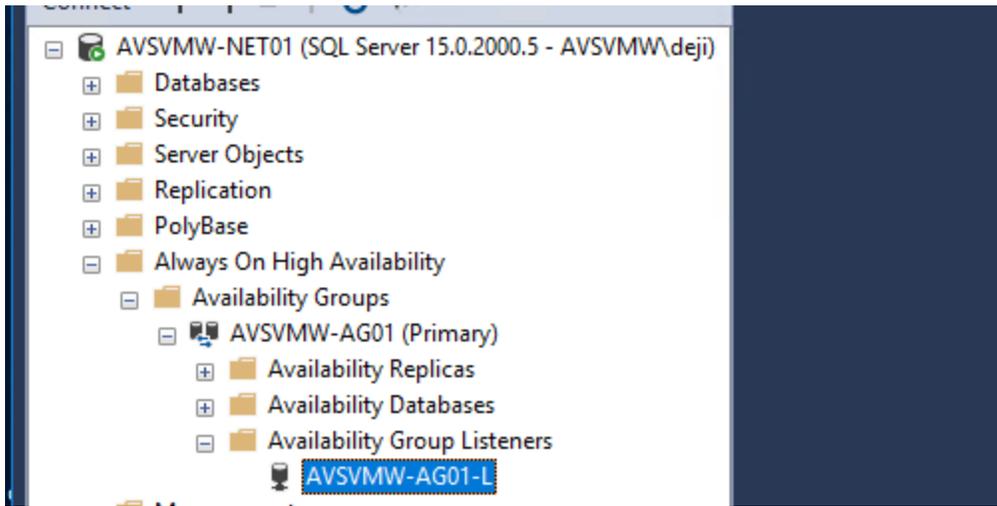




- Give the Listener a name and an IP Address.
 - As you can see, the Listener has a **DNS Name**. This is part of the administrative pre-tasks not covered in this document because it is not peculiar to virtualization.
 - Also note that we left the port at the default **1433**.



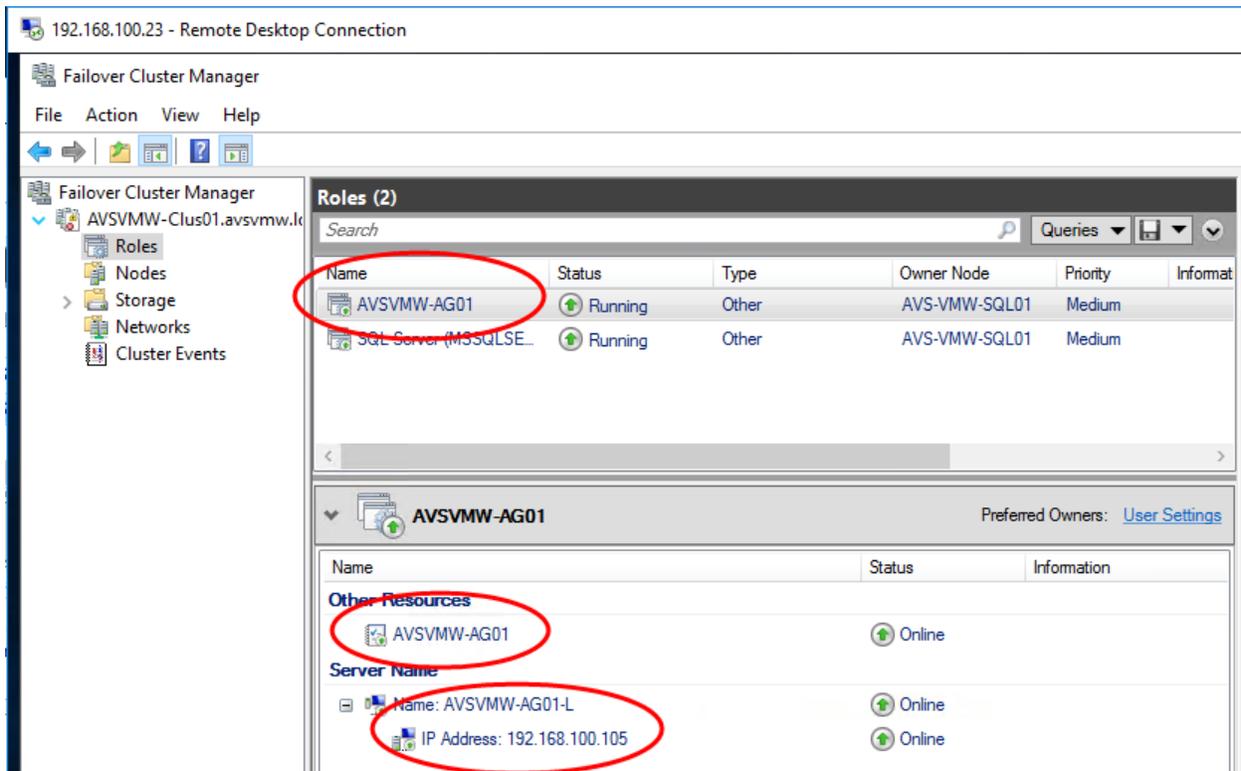
- Click **Next** and let's wrap up the setup.



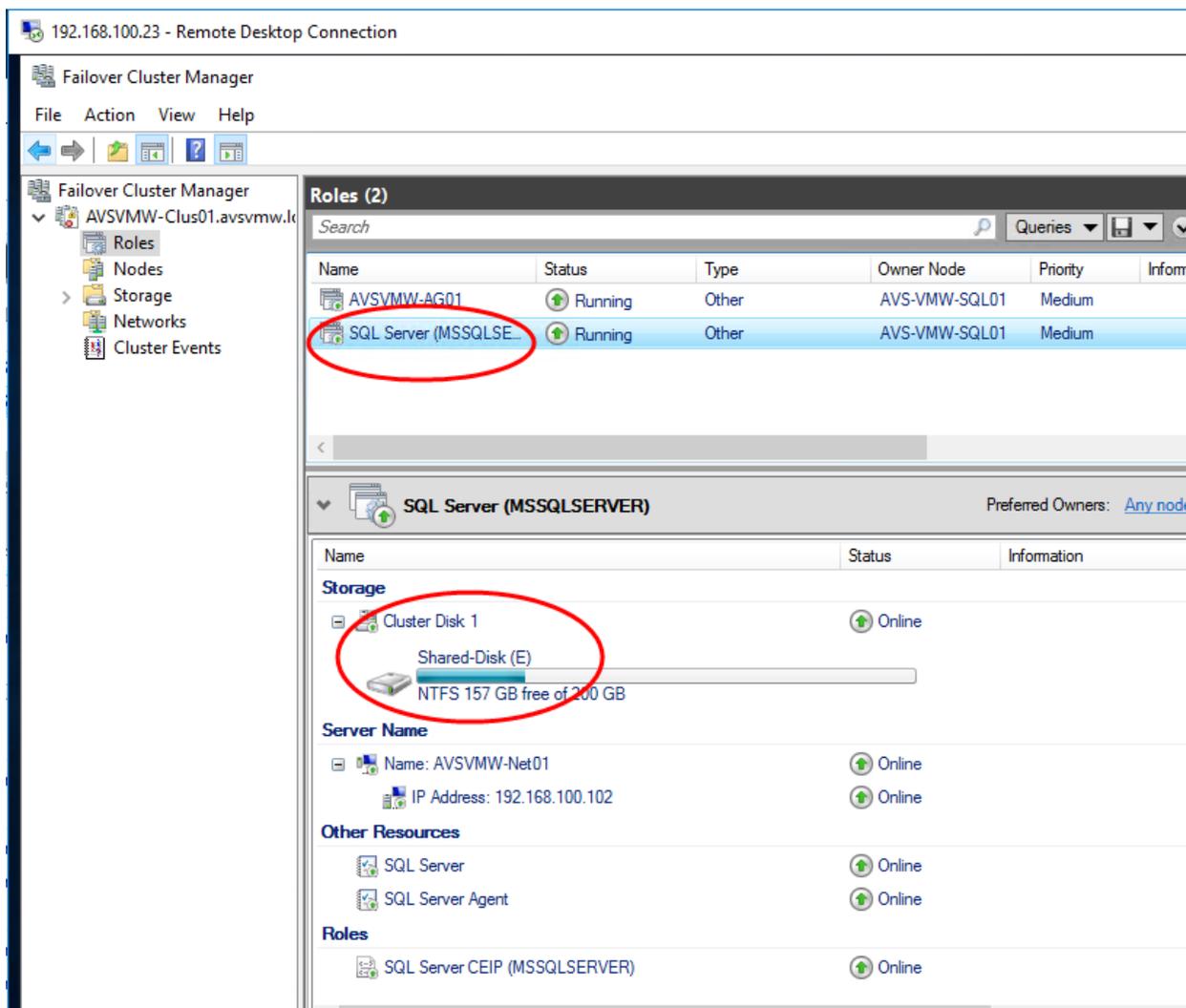
- There it is.

If we look in Windows Cluster Manager, we can now see that we have two clustered Microsoft SQL Server resources (Roles).

One is our newly clustered, share-nothing Availability Group Database



And the other is the shared-disk Failover Cluster Instance.



Conclusion

Mission accomplished. We have successfully configured a functioning, production-ready, highly-available Microsoft SQL Server farm on our Azure VMware Solution platforms, using a mixture of the two most common application-level HA options.

At its core, Azure VMware Solution is a VMware vSphere-base hybrid cloud platform, allowing administrators to seamlessly and easily extend their on-premises infrastructure to the public cloud, using the familiar, true-and-tested vSphere technologies they are already familiar and comfortable with, without unnecessarily re-factoring their applications or re-learning complex processes and ways of doing things.

