TECHNICAL WHITE PAPER:
November 2025

Virtualizing Active Directory Domain Services on VMware Cloud Foundation

Reference Architecture

Table of contents

Introduction

Active Directory Domain Services (AD DS) serves as the primary directory service and authentication platform in most enterprise networks. In some organizations it receives minimal attention as a basic requirement, while others correctly classify it as a true business-critical application (BCA). Because access to network resources, Internet services, user directory lookups, email, and countless authentication workflows depend on AD DS, the stability and reliability of its foundational infrastructure must be treated with utmost seriousness.

Different organizations evaluate the criticality of AD DS in different ways, and these perspectives shape their willingness to virtualize the service. Some remain cautious and continue running part of AD DS on physical hardware—often due to outdated information, lack of virtualization experience, or hesitation about perceived risks.

Windows Server 2012 introduced features specifically designed to resolve long-standing concerns about AD DS virtualization. Combined with modern VMware® Cloud Foundation (VCF) capabilities and mature best practices, these advancements now enable organizations to safely virtualize one hundred percent of their AD DS infrastructure.

Although core considerations from earlier guidance remain valid, improvements in AD DS, the Windows operating system, and VMware vSphere over time have made virtualized domain controllers safer, more secure, and more reliable. Only a few enhancements—such as Virtualization-Based Security (VBS), Secure Boot, VM-Level Encryption, and improved guest VM timekeeping—materially alter prior recommendations. This updated version incorporates these changes where applicable.

Purpose

This document provides best-practice recommendations for deploying AD DS on VCF. These guidelines apply broadly and are not limited to specific hardware types or particular AD DS deployment sizes. Examples are offered to enhance clarity, not to define rigid design requirements.

Target Audience

This guide assumes a basic understanding of both AD DS and vSphere.

- Architects can use it to evaluate design considerations for virtualizing AD DS.
- Engineers and administrators can reference it as a technical capability guide.
- Management and process owners can use it to model workflows that leverage the efficiencies of virtualization.

Scope

This document discusses the following:

- Drivers supporting AD DS virtualization and why VCF is the ideal platform
- Historical blockers and modern advancements that removed earlier constraints
- Technical design considerations and recommended best practices for success

Why Virtualize Active Directory?

AD DS is a hierarchical, multi-master directory service providing:

- A schema defining directory objects and attributes
- A global catalog containing searchable data across the forest
- A replication service for synchronizing directory information
- Operations Master roles that support essential forest and domain functions, such as schema modification and RID allocation

While AD DS primarily supports authentication, it has evolved into a central repository for user identities, computer accounts, application configuration data, network resource location services, and name resolution. Many external systems also rely on AD DS for authentication. Because AD DS is so essential, it must be designed with the same diligence applied to any BCA.

The critical nature of AD DS should not prevent its virtualization. Domain controllers store full, writable copies of the AD DS database. Their workload and deployment characteristics make them an excellent fit for virtual machines.

Workload Characteristics

Although domain controllers support nearly all users and computers, their workloads are typically modest. They commonly process hundreds—or in highly active environments, thousands—of lightweight directory queries per minute. Even during activity spikes, most queries operate on cached data, dramatically reducing disk I/O requirements.

Under 64-bit operating systems such as Windows Server 2008 R2 and later, the full Active Directory database may be stored in memory.

AD DS's distributed architecture naturally provides client load balancing. Virtualization strengthens this effect by enabling the deployment of multiple smaller domain controllers without hardware-availability limitations. This increases scalability and availability.

Domain controllers are generally minimalistic: Windows Server, anti-virus, monitoring agents, and backup tooling. As a result, they seldom consume significant resources. Running them on dedicated physical servers often wastes compute capacity. Virtualization avoids this waste and enables efficient resource use.

Virtualization Is Mainstream

Many organizations now follow a "virtualization-first" strategy, where new servers default to virtual machines. In some organizations, deploying a physical server is an exception requiring upper-management approval. Virtualization not only improves efficiency, reduces capital and operational expenditures, and increases agility—it has also proven robust enough for the world's most demanding workloads.

Early objections to virtualizing production systems have largely disappeared. Extensive performance evidence, customer references, and years of real-world operation have validated virtualized infrastructures. Today, most organizations run their most critical workloads—including AD DS—on vSphere. As aging physical domain controllers reach end-of-life, many organizations are eliminating physical Windows Server deployments entirely. There is no longer a technical justification for keeping AD DS on physical hardware.

Availability

When domain controllers are deployed according to best practices, the loss of a single domain controller does not compromise directory availability. Each domain controller hosts a complete copy of its domain database. Users and applications locate domain controllers using DNS (typically hosted on domain controllers themselves). When a domain controller becomes unresponsive, clients automatically retry requests with healthy servers.

Operations Master roles, however, must be handled by designated domain controllers. If such a domain controller becomes unavailable, operations requiring that role may fail. These operations are infrequent, and the impact is usually isolated to the initiating administrator. Roles can be transferred or seized if required.

Physical server deployments often require multiple servers to ensure availability—even though domain controller workloads typically consume only 5–10% of available CPU resources. Virtualizing AD DS on existing vSphere clusters is dramatically more efficient.

Understanding Domain Controller Virtualization

Historically, concerns about virtualizing domain controllers were limited but significant. They centered on data integrity, security, and the potential consequences of mismanagement. VMware has since provided technical solutions, operational safeguards, and clearer administrative tools to mitigate these risks.

Securing Virtualized Domain Controllers

Physical domain controllers have long required robust physical protections in addition to logical AD DS security. Traditionally, organizations relied on controlled rooms, locked racks, or data-center cages. AD DS databases stored on direct-attached storage benefitted from these physical barriers.

Virtualization raised concerns about how easily virtual machines could be copied or migrated. If an attacker gained administrative access to a vCenter Server or ESXi host, they could potentially access domain controller VMDKs. This issue reflects broader infrastructure security—not a flaw inherent to virtualization. Strong access control, auditing, and network isolation remain the most effective mitigation strategies.

Highly regulated environments may require additional technical hardening, including:

- Isolated management networks
- Restricted VM-to-host interaction
- Strict vCenter Server access controls

For more information on securing virtual machines and vCenter Server, refer to vCenter and ESX Security Best Practices and Resources.

Disk Encryption Options

Organizations can reduce the impact of virtual disk theft using several encryption methods:

- Storage array-level hardware encryption
- In-guest encryption such as BitLocker or TrueCrypt
- File-level or partition-level encryption

Hardware-based encryption minimizes administrative overhead but requires specialized drives. In-guest encryption is more cost-effective but introduces key-management risks.

These trade-offs apply identically to physical and virtual domain controllers.

VM Encryption in VCF

VMware introduced VM Encryption [1] in vSphere 6.5. The feature allows vCenter to establish trust with an external Key Management Server (KMS). ESXi hosts generate data encryption keys used to encrypt VM files. These are themselves encrypted using key-encryption keys sourced through vCenter from the KMS. vCenter stores only key identifiers—not the encryption keys themselves.

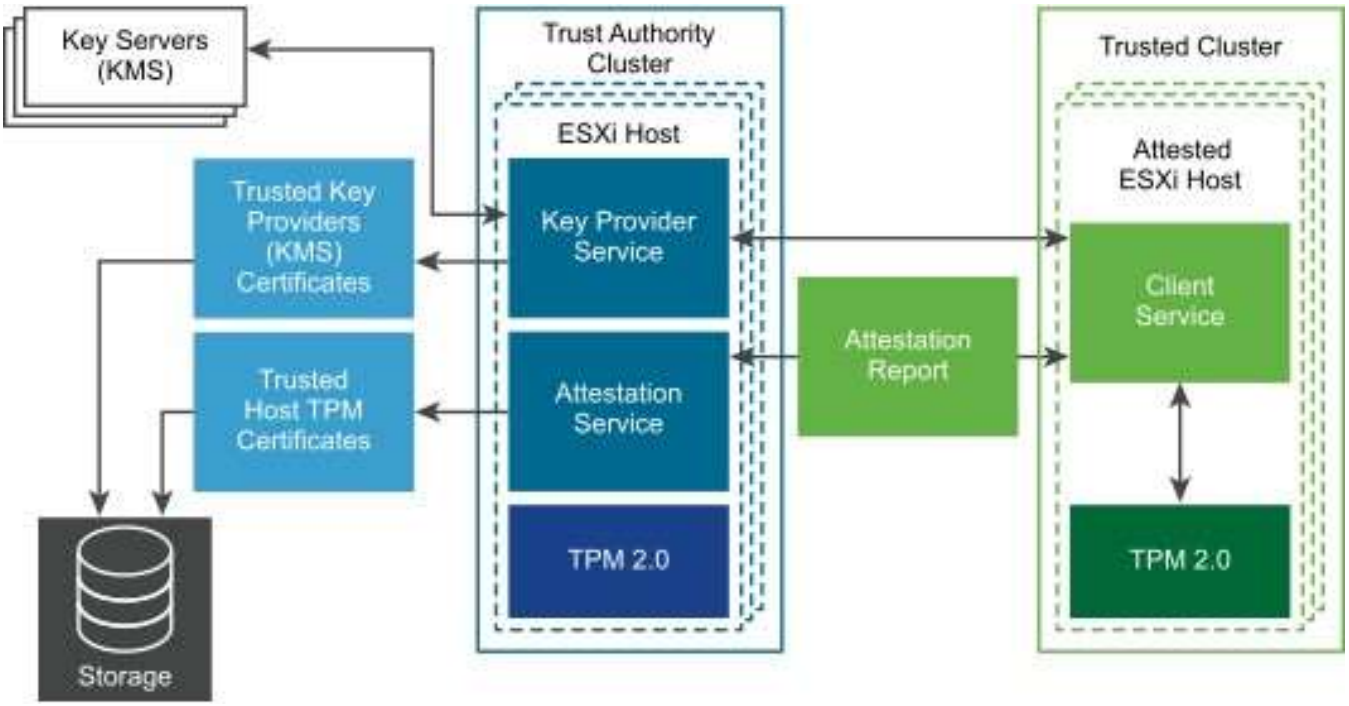
Key points:

- Only ESXi hosts hold the actual data-encryption keys.
- Encrypted VMs removed from the environment (“stolen VMDK”) cannot be decrypted.
- If a host becomes unavailable, vCenter retrieves a new key-encryption key for the replacement host.
- VM Encryption is guest-OS-agnostic; Windows domain controllers require no special configuration.

vSphere 7.0 expanded this functionality with vSphere Trust Authority, reducing reliance on external KMS deployments and strengthening attestation and trust workflows.

Figure 1 - VM Encryption Workflow

[1] How vSphere Virtual Machine Encryption Protects Your Environment



Protecting AD DS against Virtual Infrastructure Failures

Production environments commonly rely on a combination of physical and virtual domain controllers, resulting in what is often referred to as a hybrid deployment model. These mixed environments usually exist for several understandable reasons. In many cases, organizations still operate legacy physical servers that remain tightly coupled to older applications or hardware dependencies, making full virtualization a slower or more complicated undertaking. In other situations, administrators simply feel more at ease knowing that at least one or two domain controllers are running on bare metal, believing this provides an additional measure of security or a fallback option in the event that something unexpected happens within the virtual infrastructure. These first two motivations—legacy requirements and operational comfort—are both legitimate and commonly encountered across enterprises of all sizes.

Over time, most organizations gradually migrate the majority of their domain controllers to virtual machines hosted on modern virtualization platforms. Yet it is extremely common to see one or two physical domain controllers retained as a precaution, even in otherwise fully virtualized environments. These physical systems are often viewed as a hedge against the possibility of a catastrophic failure in the virtualization stack—an event that could, in theory, compromise access to directory services. While this concern makes intuitive sense, it is usually rooted in unfamiliarity with platforms like VMware Cloud Foundation rather than in any inherent technical weakness. When administrators understand how VCF achieves fault tolerance, redundancy, and workload distribution, confidence in virtualized domain controllers typically rises.

Active Directory itself is designed from the ground up to function as a distributed, resilient platform. Its replication model, site topology, and multi-master architecture all exist to ensure reliability even when individual systems fail. Likewise, VCF can be architected in a distributed manner to provide a high degree of availability. A vSphere cluster, which consists of multiple ESXi hosts, serves as a unified management and compute domain. These hosts share common resources—storage arrays, network uplinks, and often chassis or rack-level infrastructure. When deployed correctly, the cluster can seamlessly absorb the failure of an individual host by redistributing workloads across remaining hosts with minimal service disruption.

This ability to operate as a pooled set of resources, along with the inherent redundancy available within storage, networking, and physical placement, allows a single well-designed vSphere cluster to achieve strong resilience by intentionally segmenting compute resources across distinct failure domains. When an organization expands its architecture to include multiple clusters, the separation of failure domains becomes even more robust and easier to

implement. All of this means that a thoughtfully architected virtual infrastructure can be highly fault tolerant. As a result, the argument that domain controllers should not be fully virtualized due to fears of a virtualization-level outage carries little weight when the environment is designed according to modern best practices.

Time Synchronization

Accurate and consistent time synchronization is absolutely essential in an Active Directory domain. Time affects everything from authentication to replication, and even a small amount of drift can cause authentication failures, replication inconsistencies, or other subtle and hard-to-diagnose issues. Because timekeeping is so critical, some administrators worry that virtualization will amplify timing problems or introduce new sources of drift. It is true that virtual machines can experience variations in time due to the nature of shared physical resources, especially when multiple virtual machines are contending for CPU cycles on the same host. Fortunately, these concerns are well-understood and can be effectively addressed by following VMware and Microsoft best practices.

Operating systems rely on one of two general mechanisms to track time: tick counting or tickless timekeeping. In tick-based systems, the OS uses a hardware timer to generate periodic interrupts—sometimes hundreds of times per second—and counts these interrupts to measure the passage of time. In tickless systems, the OS instead queries a continuously running counter that tracks elapsed time since boot. Windows uses the tick-based method. Systems running multiprocessor HALs, as well as certain ACPI uniprocessor HALs, rely on the CMOS periodic timer to deliver these interrupts, which the OS uses to maintain its internal sense of time.

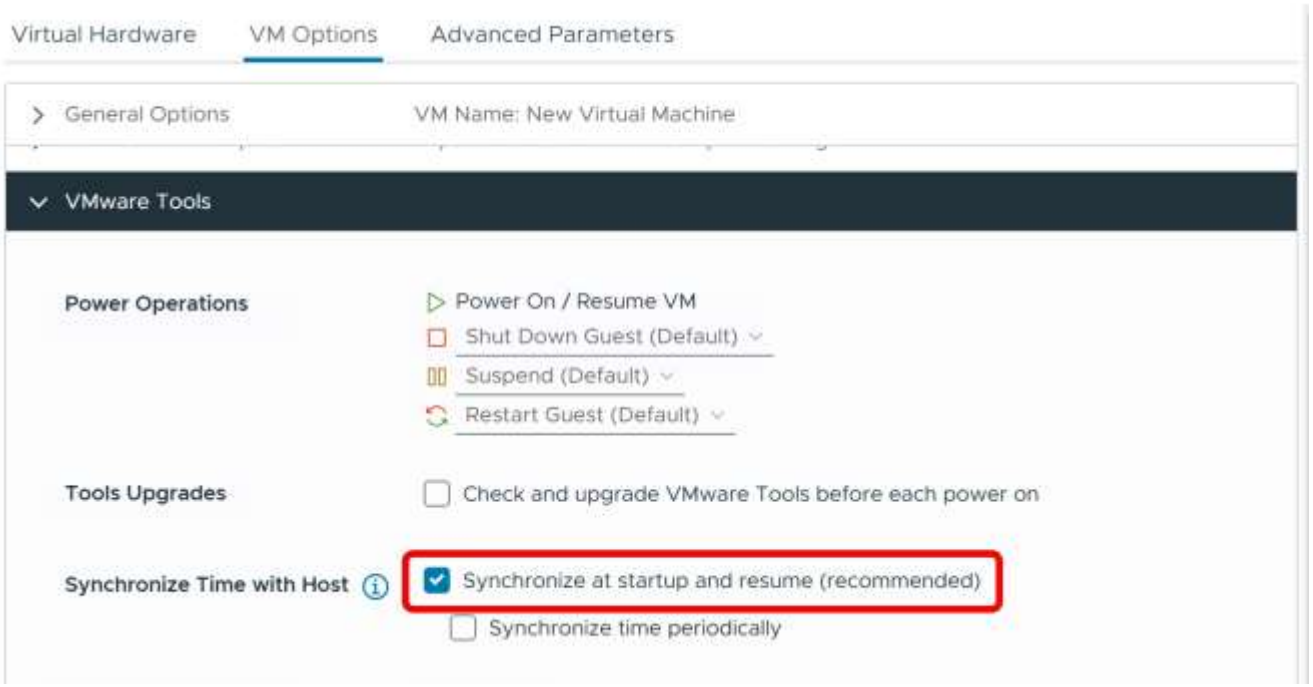
In addition to tracking elapsed time, operating systems must also maintain an awareness of real-world (wall-clock) time. During startup, Windows reads this value either from the system's CMOS clock or from a network-based time source. In virtual machines, VMware Tools introduces another potential time source by providing mechanisms for the guest to synchronize its clock with the host. Just as physical hardware clocks can drift, virtual hardware timers can drift as well, so some form of ongoing time correction is always necessary. For Windows systems, the built-in Windows Time Service (W32Time) performs this role.

Within an Active Directory forest, time synchronization follows a strict hierarchy, beginning with the forest root PDC emulator, which acts as the authoritative source. All other domain controllers and domain-joined systems synchronize through this hierarchy. Because the Windows Time Service is deeply integrated with Active Directory operations and already optimized for domain environments, it is the natural choice for timekeeping inside Windows virtual machines.

However, conflicts can arise if both the Windows Time Service and VMware Tools attempt to correct the system clock. Domain-joined Windows systems already rely on the domain hierarchy for reliable, authoritative time updates. If VMware Tools is simultaneously enabled to sync the clock with the host—which is usually not an authoritative source—these two mechanisms can work against one another, causing the clock to oscillate or drift toward incorrect values. For this reason, the recommended practice is to standardize on a single method. In the case of Windows guests joined to an Active Directory domain, the Windows Time Service should be used exclusively.

To confirm the state of time synchronization on a VM, check the VMware Tools menu under the "VM Option" user interface. VMware Tools time synchronization is enabled by default in VCF (as shown in the image below), so it is very important for Administrators to note this and ensure that it is disabled as part of their standard VM provisioning process when creating a virtual machine which will be eventually joined to an Active Directory domain.

Figure 2 - Time Synchronization Default Setting

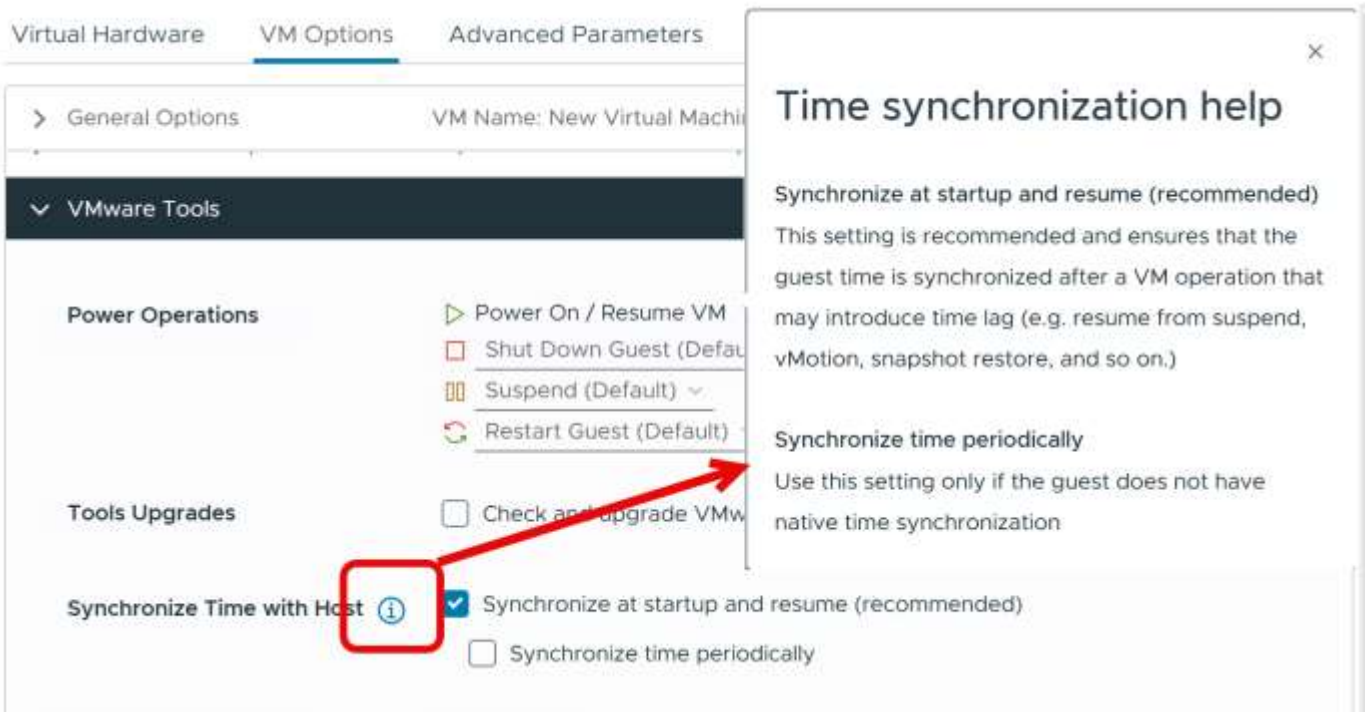


Even with VMware Tools time synchronization disabled, time synchronization is performed during guest operating system startup and during some virtual machine maintenance operations on the VCF platform. For more information on COMPLETELY disabling VMware time synchronization for virtual machines in a VCF infrastructure, see VMware's Disabling Time Synchronization.

Because VM Time Synchronization and the effects of certain VM operations in a VCF environment on a VM's Guest Operating System continues to be a popular topic of discussion and misunderstanding in the field, especially as it relates to Active Directory Domain Services, VMware added an improved visual cue for VM Administrators to be better able to understand and interpret the configuration options provided in a VM's properties.

Beginning with VMware vSphere 7 Update 1, the VM time configuration selection menu is now more intuitive.

Figure 3 - Expanded Time Synchronization Visual Cue



We have also published a more comprehensive explanation of the intricacies of Time Keeping, Time Synchronization, Faulty BIOS and VM Operations impacts on VCF-hosted Windows VMs in the following blog post - Ensuring Accurate Time-Keeping in Virtualized Active Directory Infrastructure

USN Rollback

The multi-master, distributed architecture that underpins Active Directory is fundamental to its resilience, scalability, and flexibility. By allowing multiple domain controllers to process write operations independently, the system avoids single points of failure and enables large environments to scale horizontally. However, this same design introduces the need for robust mechanisms that detect conflicting updates, reconcile simultaneous changes, and ensure that all domain controllers eventually converge on a consistent view of the directory. While the broader topic of Active Directory replication can become extremely intricate, this section zeroes in on one specific failure condition—Update Sequence Number (USN) rollback—along with its underlying cause, its practical implications, and the measures implemented by Microsoft to prevent it.

Every modification made to the Active Directory database must be uniquely identifiable so that the replication engine can track the change, determine its origin, and propagate it correctly to the rest of the environment. To accomplish this, each write operation is marked with two critical pieces of metadata:

- **The Update Sequence Number (USN)**, a monotonically increasing counter that is local to each domain controller and increments with every write operation.
- **The InvocationID**, which uniquely identifies the Active Directory database instance running on a domain controller. This identifier distinguishes not only between different domain controllers, but also between different database incarnations on the same domain controller when it has been rebuilt or restored.

Together, these pieces of information form the backbone of Active Directory's ability to replicate reliably. Each write becomes a replicable, traceable, and verifiable transaction.

As domain controllers process changes—whether creating brand-new directory objects or modifying existing attributes—they continually allocate new USNs to reflect these operations. To keep replication consistent and to avoid reapplying data unnecessarily, domain controllers rely on the Up-to-Dateless (UTD) Vector, also known as the UTD Vector table. This structure serves several important functions:

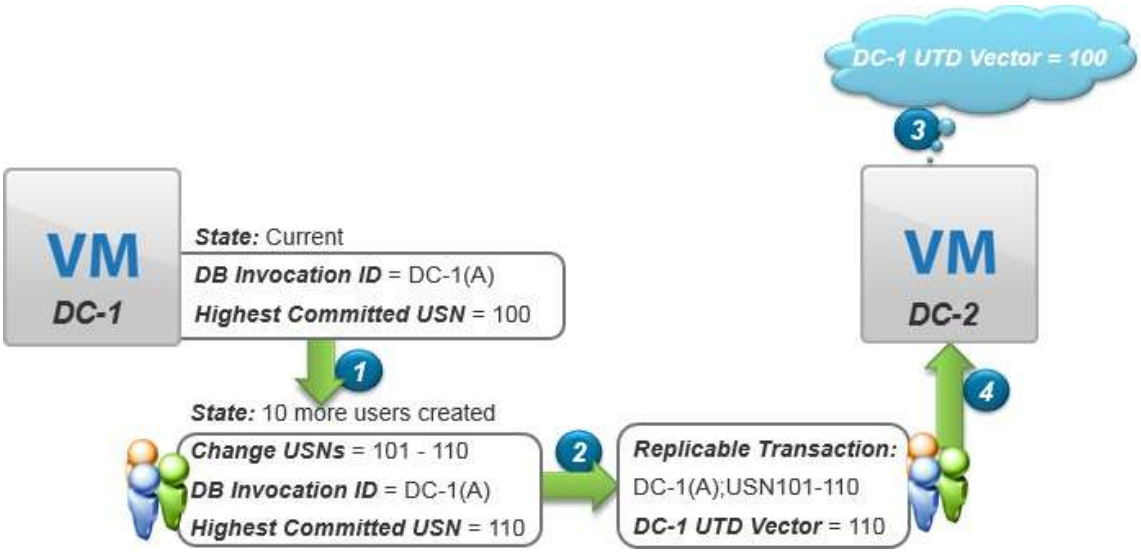
- It records the unique Directory System Agent (DSA) GUID for each replication partner in the environment.
- It maintains, for each partner, the highest USN value the local domain controller has already received and applied from that partner.
- It stores the timestamp of the last successful replication cycle with each partner, allowing the system to track the freshness of data.

By maintaining this information, domain controllers can make informed decisions about which changes need to be requested, which have already been applied, and how to efficiently reconcile updates across the entire environment. The UTD Vector effectively ensures that each domain controller not only knows what changes it has produced, but also what changes every other domain controller has produced, enabling accurate and timely propagation of directory updates.

This tracking is essential because it prevents the replaying of old data, avoids overwrite conflicts, and ensures proper convergence across all replicas. When functioning normally, this process makes Active Directory's internal replication remarkably reliable. However, when anomalies occur—such as a USN rollback—the replication engine loses the ability to determine which changes are truly new and which appear to be new but are in fact outdated. This condition can lead to inconsistent data, lingering replication failures, and misaligned directory states, making USN rollback one of the more serious replication issues administrators may encounter.

In the example illustrated below, domain controller 1 (DC-1) performs a series of write operations by creating ten new user objects. Each new user is assigned a unique Update Sequence Number—USN 101 through USN 110. These USNs, combined with DC-1's InvocationID, which in this scenario is labeled DC-1(A), form the complete identity of the originating write transactions. This pairing (USNs 101–110 plus InvocationID DC-1(A)) defines the changes that are eligible for replication. Once created, these updates are then replicated to domain controller 2 (DC-2), ensuring that both systems maintain a consistent view of the directory.

Figure 4 - Active Directory Object Replication



The introduction of virtualization brought domain controllers into an environment where administrators gained access to powerful capabilities that were impossible with physical hardware. Two of the most impactful capabilities are virtual machine snapshots and virtual machine cloning. These features significantly reduce deployment time, make testing more flexible, and streamline certain recovery scenarios. However, they also introduce unique risks when used on domain controllers due to the way Active Directory tracks replication metadata.

Active Directory relies heavily on monotonic counters, timestamps, InvocationIDs, and UTD vectors to ensure that replication occurs reliably and in the correct order. When a domain controller is cloned or reverted to a previous point in time using a snapshot, its internal replication metadata—including its highest committed USN—may revert to an earlier value. Meanwhile, other domain controllers in the environment continue to advance their replication states normally. This

divergence creates a condition where one domain controller's replication tables no longer match the expectations of its replication partners. Such misalignment can lead to corruption or inconsistency within the Active Directory database, ultimately resulting in what is known as a USN rollback.

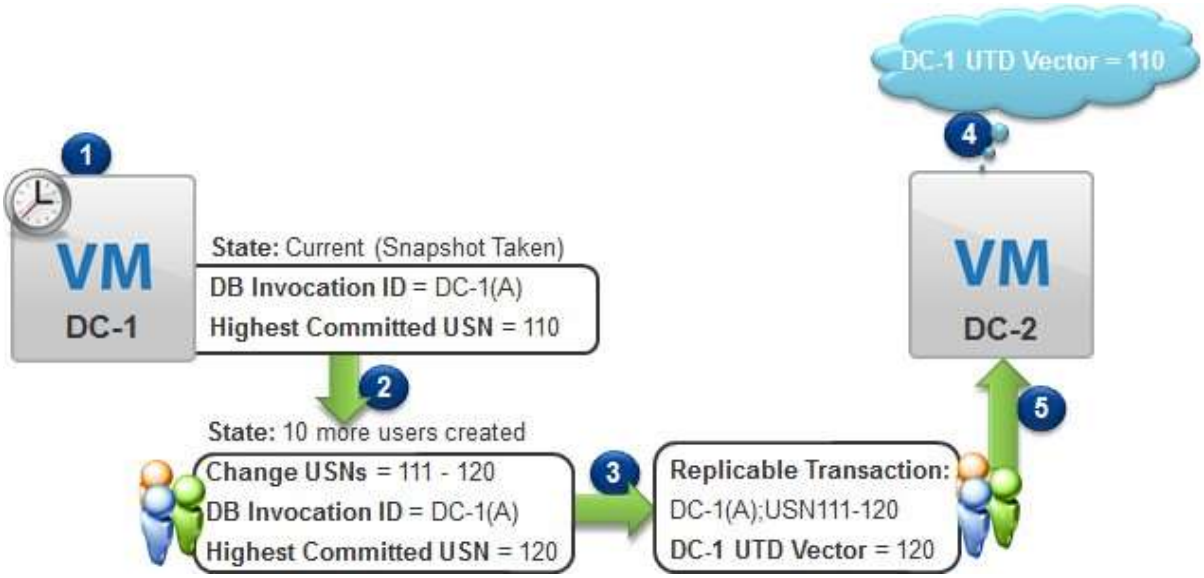
A USN rollback occurs when a domain controller continues to accept new write operations, but its replication partners refuse to inbound-replicate those changes. This happens because the domain controller's local USN values appear lower than what its partners believe they should be, based on their own previously recorded UTD vectors. As a result, the affected domain controller generates changes that are never replicated—and therefore never become authoritative in the broader directory.

To illustrate this scenario, consider the following progression.

Snapshot Taken and Additional Users Created

In the next illustration, DC-1's highest committed USN is 110, which matches what DC-2 expects. At this point, an administrator takes a snapshot of the virtual machine running DC-1. After the snapshot is captured, the administrator proceeds to create ten additional users on DC-1. These new write operations receive USNs 111 through 120. Because these are higher than the values in DC-1's own UTD vector, the changes replicate successfully to DC-2

Figure 5 - Users Created After a Virtual Machine Snapshot



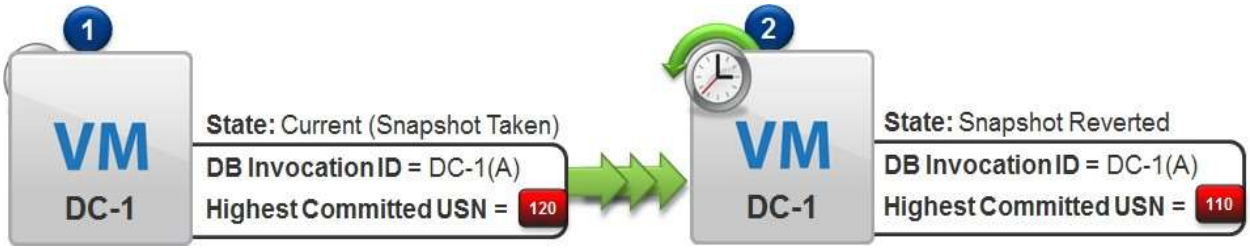
Once replication completes, both DC-1 and DC-2 contain the new users, and all replication metadata is aligned at USN 120.

Snapshot Reversion

After replication converges, the administrator reverts DC-1 to the previously captured snapshot. This rollback restores the state of DC-1's Active Directory database to the point before the last ten users were created. As a result, DC-1's highest committed USN reverts to 110.

Crucially, the other domain controllers—including DC-2—have no awareness that DC-1 has been rolled back. They still contain the ten users created after the snapshot, and they still maintain UTD vectors indicating that they have successfully replicated USNs up to 120 from DC-1. The replication partners therefore believe DC-1 has already processed and acknowledged those updates, even though DC-1's reverted database no longer contains them.

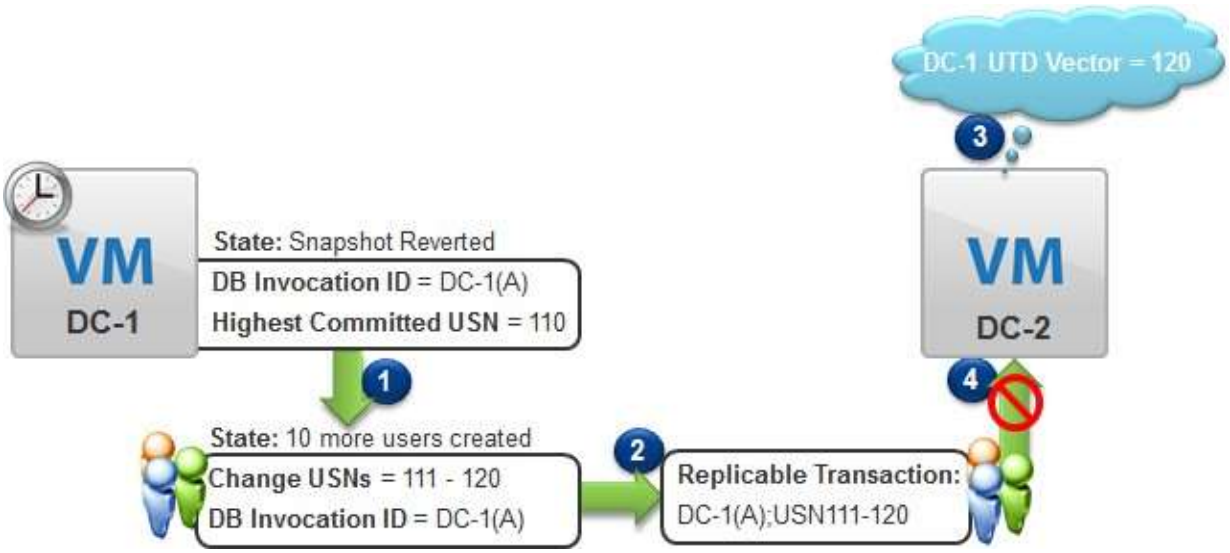
Figure 6 - Domain Controller Reverted to Previous Snapshot



Rollback Effects and Replication Failure

Following the rollback, DC-1 begins accepting new write operations again. The administrator creates another batch of ten new users, which DC-1 assigns USNs 111 through 120—values that appear correct according to DC-1's local state but are actually stale from the perspective of DC-2. Because DC-1 still uses the same InvocationID, DC-2 interprets these USNs as already-replicated operations. From DC-2's perspective, USNs 111–120 from DC-1 were previously received and processed; therefore, DC-2 simply ignores the new changes and does not inbound-replicate them.

Figure 7 - USN Rollback Effect after Reverting DC-1 Snapshot



This leads to a fractured state:

- **DC-1** now believes it possesses ten valid new users—but these objects exist only on DC-1 and nowhere else.
- **DC-2**, and any other domain controllers in the forest, still retain the previous ten users created before the snapshot was reverted—users that no longer exist on DC-1.
- Both domain controllers are now out of sync, and each assumes its own data is correct.

This situation illustrates precisely how a USN rollback disrupts the health and consistency of Active Directory replication. Without corrective action, such divergence can propagate or worsen over time, leading to directory corruption, lingering objects, and unpredictable behavior across the environment.

Because of the UTD vector, DC-2 assumes that DC-1 has knowledge of the previously-created 10 users, and as a result, the changes corresponding to USN 111 through 120 are not replicated to DC-1. DC-1 has 10 users that it assumes are valid, but which do not exist on any other domain controller. Additionally, DC-2 has 10 users that it assumes are valid, but which do not exist on DC-1. This is an example of how USN rollback can disrupt replication throughout the environment.

Among the technical reasons for not virtualizing domain controllers, the fear of a USN rollback is perhaps the most compelling and technically valid. However, mitigating the risk of a USN rollback is not difficult. The best way to avoid a USN rollback is to implement a technical solution as opposed to a policy-based solution. Microsoft has achieved this since Windows Server 2012 with its implementation of VM-Generation ID.

Mitigation and the Modern Solution

Among the technical reservations administrators historically raised about virtualizing domain controllers, the risk of USN rollback was arguably the most justified. Unlike other concerns based on misconceptions or lack of familiarity with virtualization platforms, the risk posed by snapshots and cloning was real and could cause severe, systemic replication failure.

Fortunately, Microsoft introduced a robust and elegant solution starting in Windows Server 2012: VM-Generation ID. This feature allows domain controllers to detect when they have been restored or cloned by a hypervisor. When such an event is detected, the domain controller automatically regenerates its InvocationID, invalidates outdated replication metadata, and safely reconciles its state without triggering a USN rollback.

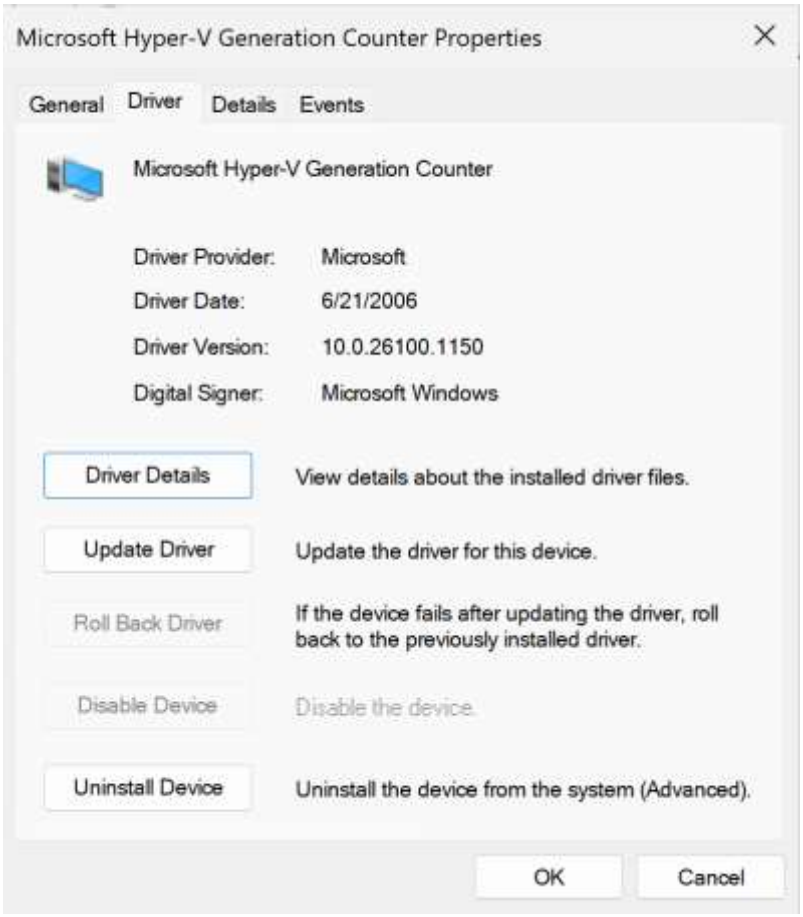
In short, while USN rollback once represented a legitimate threat to virtualized domain controllers, modern platforms have made it fully avoidable when configured correctly.

Windows Server Virtualization Safeguards

Beginning with Windows Server 2012, Microsoft introduced the VM-Generation ID, a 128-bit counter exposed by the hypervisor to the virtual machine guest operating system. The VM-Generation ID provides the virtual machine guest operating system with knowledge of the state of the virtual machine. For example, whether or not the virtual machine has been restored to a previous point in time or has been cloned.

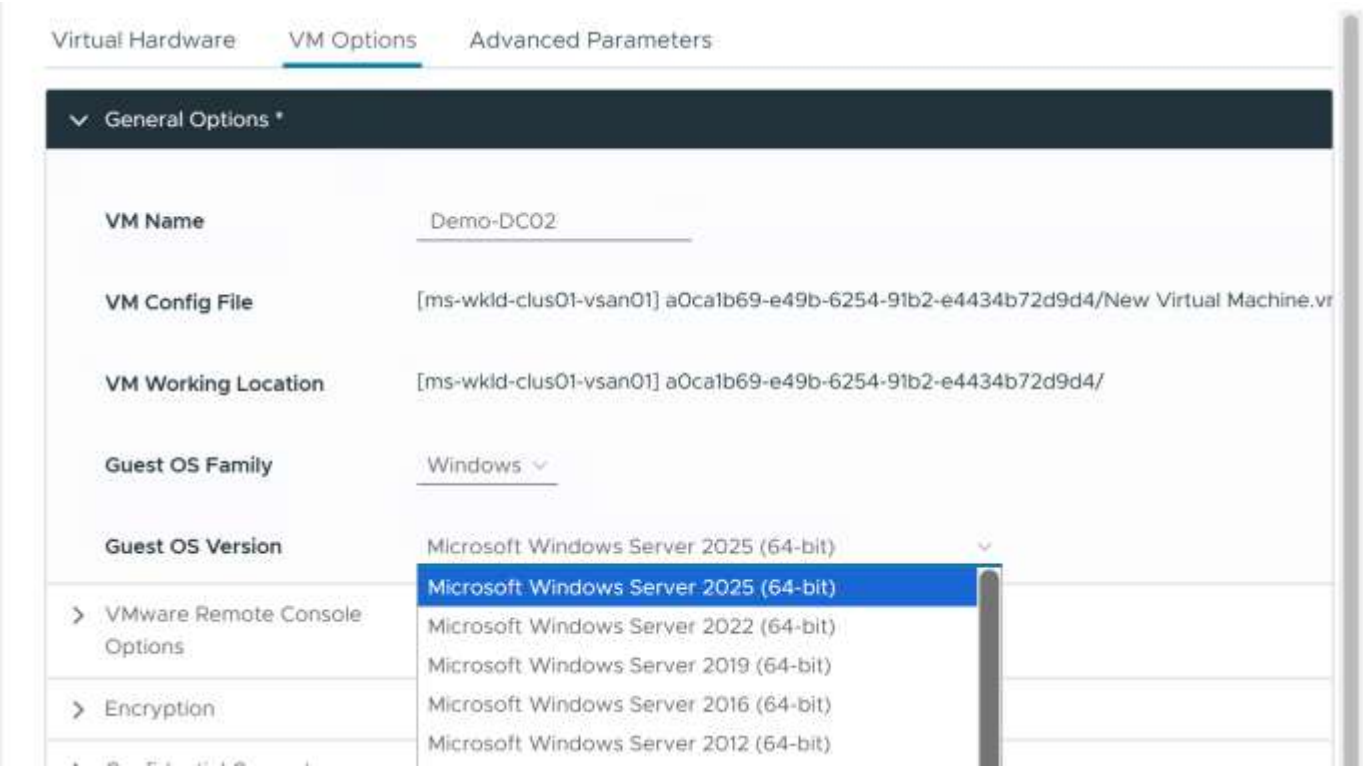
The VM-Generation ID is exposed to the guest operating system through the Microsoft driver, Microsoft Hyper-V Generation Counter. Contrary to the name, the driver is a generic communication path between the guest operating system and the hypervisor provided by Microsoft and is not specific to Hyper-V. The following screen shows the properties page for the Microsoft Hyper-V Generation Counter device.

Figure 8 - Microsoft Hyper-V Generation Counter Device



VM-Generation ID is available in all shipping and supported Microsoft Windows Operating Systems and VMware vSphere and VCF versions.

Figure 9 - Virtual Machine Guest Operating System Version



When deployed on a supported hypervisor and supported guest operating system, the VM-Generation ID is generated during the creation of the virtual machine. Over the lifecycle of the virtual machine, the VM-Generation ID is updated as a result of various state changes of the virtual machine. The following table lists the common changes in the lifecycle of a virtual machine and whether or not these changes cause a VM-Generation ID change.

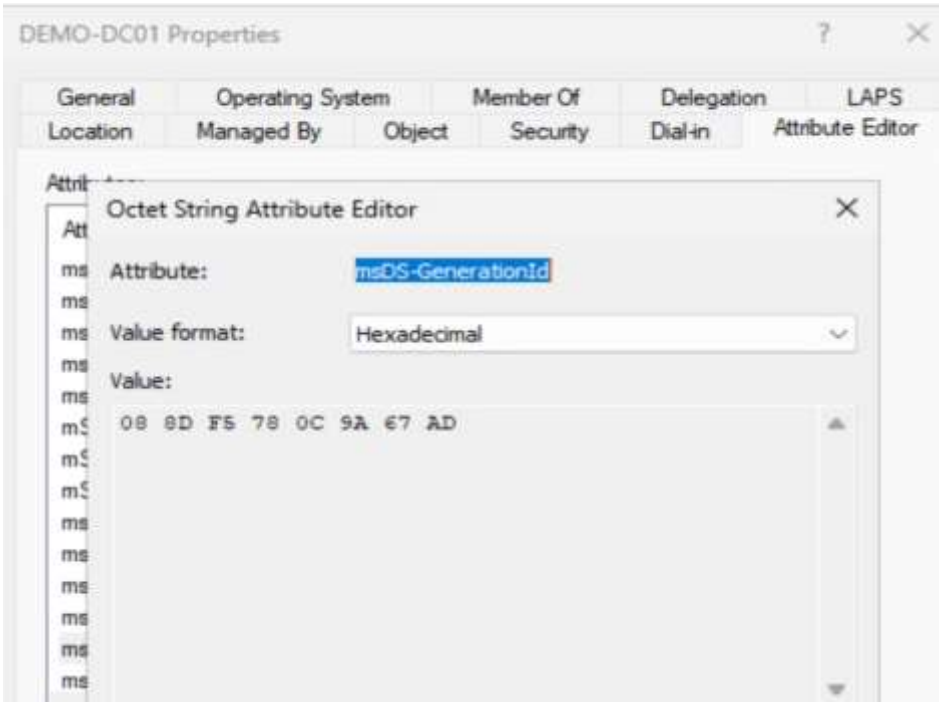
Table 1 - Virtual Machine State Changes and Effects on VM Generation-ID

Scenario	VM-Generation ID Change
VMware vSphere vMotion®/VMware vSphere Storage vMotion	No
Virtual machine pause/resume	No
Virtual machine reboot	No
vSphere host reboot	No
Delete VM Snapshot	No
Import virtual machine	Yes
Cold clone	Yes
Hot clone Note: Hot cloning of virtual domain controllers is not supported by either Microsoft or VMware. Do not attempt hot cloning under any circumstances.	Yes
New virtual machine from VMware Virtual Disk Development Kit (VMDK) copy	Yes
Cold snapshot revert (while powered off or while running and not taking a memory snapshot)	Yes

Hot snapshot revert (while powered on with a memory snapshot)	Yes
Restore from virtual machine level backup	Yes
Virtual machine replication (using both host-based and array-level replication)	Yes

Active Directory incorporates the VM-Generation ID mechanism to maintain awareness of the virtual machine's state and to detect whether that state has unexpectedly changed. This capability is essential in virtualized environments, where operations such as snapshots, restores, and certain cloning procedures can alter the virtual machine's underlying configuration or point-in-time condition. To accomplish this, Microsoft introduced a new attribute in Windows Server 2012 known as msDS-GenerationId, which is stored on each domain controller's computer object. This attribute becomes the authoritative record of the domain controller's current VM-Generation ID.

Figure 10 - msDS-GenerationId Attribute



Active Directory Domain Services (AD DS), which runs on every domain controller, evaluates the VM-Generation ID supplied by the hypervisor both during system startup and immediately before any write operation to the Active Directory database. This evaluation occurs only when the domain controller is hosted on a hypervisor that exposes and supports the VM-Generation ID functionality. The purpose of these evaluations is to ensure that the VM-Generation ID presented by the hypervisor remains consistent with the value recorded in the msDS-GenerationId attribute inside the domain controller's own Active Directory database. Any discrepancy between these values indicates that the state of the virtual machine may have changed.

A domain controller checks, evaluates, and populates its msDS-GenerationId attribute under several clearly defined conditions:

- **When a New Domain Controller Is Created**

When a domain controller is first created—whether through promotion or another supported method—the msDS-GenerationId attribute is evaluated and populated for the very first time. This establishes the initial baseline that the domain controller will use to validate its state in all future operations.

- **During Startup of an Existing Domain Controller Whose Attribute Is Not Yet Populated**

If an existing domain controller starts up and the msDS-GenerationId attribute is still empty, AD DS evaluates the current VM-Generation ID provided by the hypervisor and populates the attribute for the first time. This situation typically arises when a virtual machine is powered on for the first time on a hypervisor that supports VM-Generation ID—such as after a virtual machine import or a hypervisor upgrade. In this scenario, the domain controller initializes the attribute based on the current state of the virtual machine.

- **During Startup of an Existing Domain Controller Whose Attribute Is Already Populated**

If a domain controller starts up and the msDS-GenerationId attribute already contains a value, AD DS compares the VM-Generation ID provided by the hypervisor to the value stored in the attribute. This comparison determines whether the virtual machine's state has changed while it was powered off. If the values match, the domain controller proceeds normally. If they do not match, the mismatch signals that the virtual machine has experienced a state-altering action such as a snapshot revert, and the domain controller responds accordingly to protect the integrity of its Active Directory data.

- **Before Any Write Operation on a Running Domain Controller**

Even while the domain controller is fully operational, AD DS evaluates the VM-Generation ID before each write commit to the Active Directory database. This ensures that no changes to the virtual machine's state have occurred during runtime. A mismatch at this stage indicates that the virtual machine was altered while running, and the domain controller reacts based on that detection.

The presence and evaluation of VM-Generation ID support within Active Directory directly enables two specific features for virtualized domain controllers: **domain controller safeguard** and **domain controller cloning**. These features stem from the system's ability to track and respond to changes in VM state through the msDS-GenerationId attribute and the continuous comparison against the hypervisor-provided Generation ID.

Domain Controller Safeguards

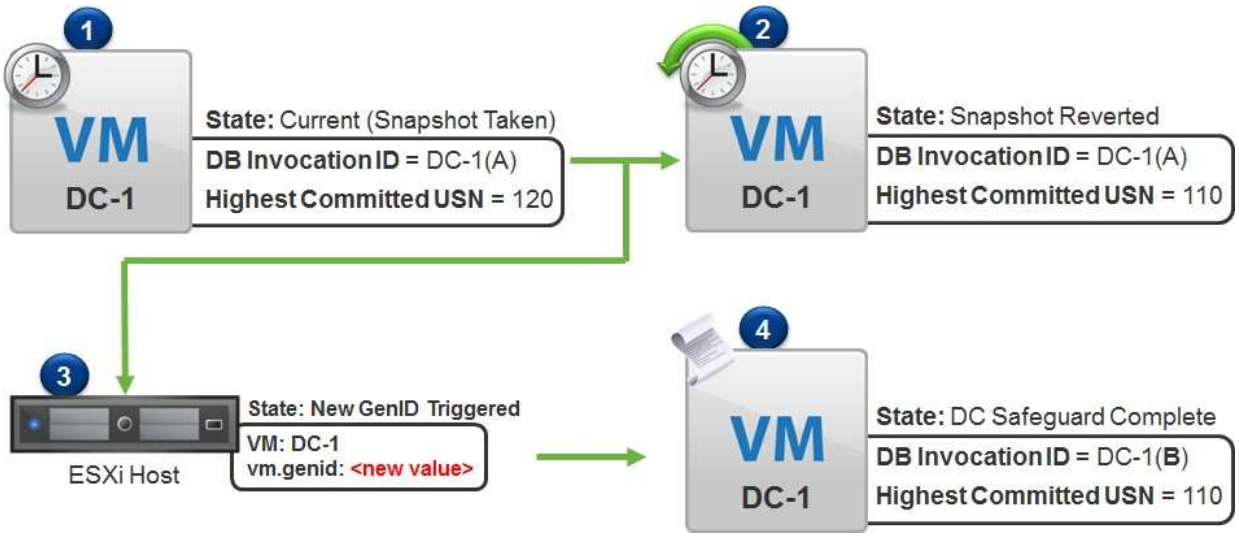
The **domain controller safeguard** feature exists to ensure that a domain controller remains functional and does not compromise the integrity of Active Directory when it has been reverted to an earlier point in time. This situation can occur when a virtualized domain controller is rolled back to a snapshot, restored from a virtual machine–level backup, or brought online after being replicated as part of a disaster recovery process. In any of these scenarios, the domain controller may suddenly find itself in a state that does not match what other domain controllers in the environment expect. To prevent this discrepancy from causing corruption or isolation, the safeguard mechanism provides a series of protections that activate automatically.

During both the startup of Active Directory Domain Services and prior to the commitment of any write operations to the Active Directory database, the system evaluates the VM-Generation ID provided by the hypervisor. This value is compared to the VM-Generation ID previously stored within the domain controller's msDS-GenerationId attribute. If the VM-Generation ID has changed since the last time the domain controller was operational, the system interprets this as evidence that the virtual machine has undergone a state-altering operation such as a snapshot revert or restore. As soon as this change is detected, the domain controller initiates a series of immediate actions—referred to as **virtualization safeguards**—to ensure that the directory remains consistent and that the domain controller can safely continue operating.

The first safeguard action is the **invalidation of the domain controller's local RID pool**. The Relative Identifier (RID) pool is used when creating new security principals such as user accounts, groups, and computers. If a domain controller were allowed to continue using a restored RID pool from an earlier point in time, it could potentially assign RIDs that had already been issued, resulting in duplicate security identifiers. By invalidating the RID pool immediately upon detecting a VM state change, the domain controller ensures that any future security principals created on that server receive unique identifiers.

The second safeguard action involves the **changing of the InvocationID** associated with the local Active Directory database. Updating the InvocationID signals to all replication partners that the database instance on this domain controller is effectively a new incarnation. This ensures that replication partners treat any subsequent updates originating from the domain controller as legitimate, fresh changes rather than older transactions. By updating the InvocationID, the domain controller preserves the integrity of the replication topology and allows replication to continue without conflict.

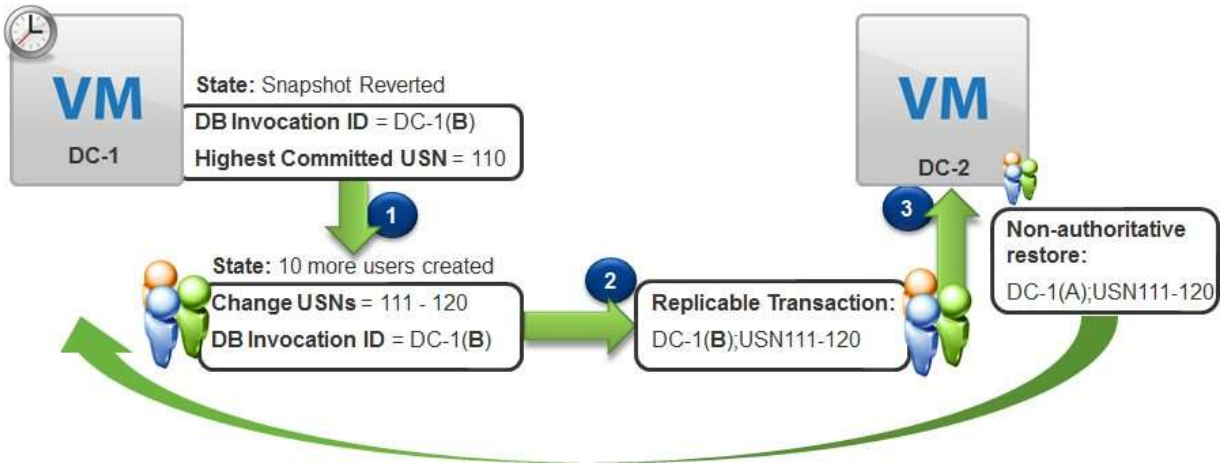
Figure 11 - Domain Controller Snapshot Reversion with Safeguard



In scenarios involving a snapshot restore, a virtual machine–level recovery, or a replicated domain controller being powered on following a disaster recovery scenario, these safeguard processes ensure that the domain controller can return to normal operation.

After the safeguards are executed, the domain controller can once again commit new changes to the Active Directory database. Furthermore, any updates that were originally created by the restored domain controller—but had been lost as a result of the recovery process—are replicated again from other available replication partners. In this way, the safeguarded domain controller resynchronizes itself with the rest of the environment and regains full participation in the replication process.

Figure 12 - Replication After Safeguard



However, in cases where a virtual machine is cloned or created from a copied virtual machine image that has not been prepared correctly, the behavior is different. Although such a domain controller can still execute safeguard operations, it cannot safely return to normal operation. To prevent the source domain controller from becoming isolated or causing inconsistencies within the environment, the unprepared clone is automatically rebooted into Directory Services Restore Mode (DVLSR). This ensures that the domain controller cannot inadvertently operate using invalid or duplicated state

information. The behavior of cloned or improperly prepared virtual machines differs significantly from the snapshot rollback scenario and will be addressed in the next section.

Domain Controller Cloning

VM-Generation ID also enables support for domain controller cloning, which is the capability to take an existing virtual machine that is already functioning as a domain controller and use it as the basis for creating an additional domain controller. This process provides an efficient, repeatable, and safe method for deploying new domain controllers by relying on a properly prepared reference virtual machine. Preparing the original domain controller for cloning requires several administrative steps, all of which must be completed successfully before the cloning process is initiated.

To begin preparing a domain controller for cloning, it is necessary to verify that all software installed within Windows can safely be duplicated. This is important because certain applications—such as monitoring systems, antivirus agents, and backup agents—often generate or rely upon unique identifiers tied to the specific system on which they run. If such applications are cloned without proper handling, the duplicated identifiers can interfere with communication between the cloned agent and the systems it interacts with, ultimately causing those applications to malfunction.

Windows maintains a built-in list of applications that are known to be safe for cloning. To identify which applications are installed on the system but not included in the default allow list, the following PowerShell command is used:

- *Get-ADDCCloningExcludedApplicationList*

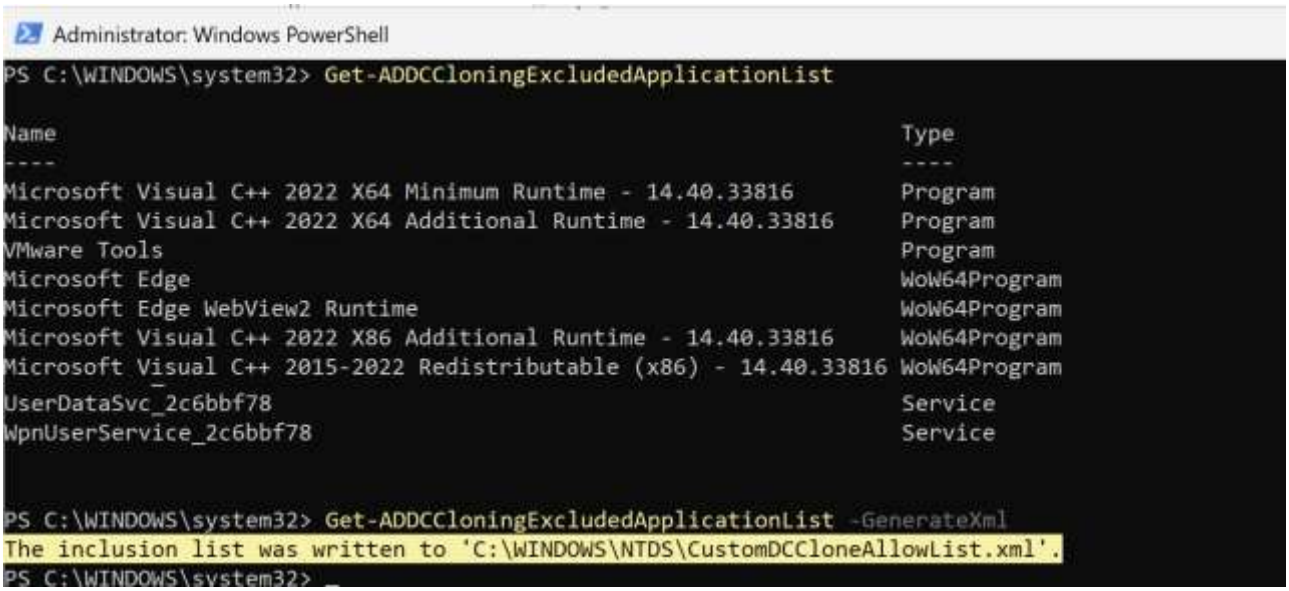
Note: VMware Tools is safe to clone.

Before proceeding with any cloning activity, administrators must confirm with the relevant software vendors whether the installed applications are safe to clone. If an application is not safe to clone, it must be removed before continuing the preparation process. Applications that are safe for cloning but not included in the default allow list must be explicitly added to a custom allow-list file named **CustomDCCloneAllowList.xml**. This is achieved by generating the XML file through the following PowerShell command:

- *Get-ADDCCloningExcludedApplicationList –GenerateXml*

The output from both commands—first listing excluded applications, then generating the custom allow list—is shown in the following illustration.

Figure 13 - Generating a Custom Application Allow List



The next preparatory step is to add the source domain controller to the Cloneable Domain Controllers Active Directory global security group. Membership in this security group signals that the domain controller has been approved for cloning. This requirement also functions as a role-separation mechanism, ensuring that an administrator with the appropriate level of authority—typically one responsible for Active Directory—explicitly authorizes the cloning of a domain controller.

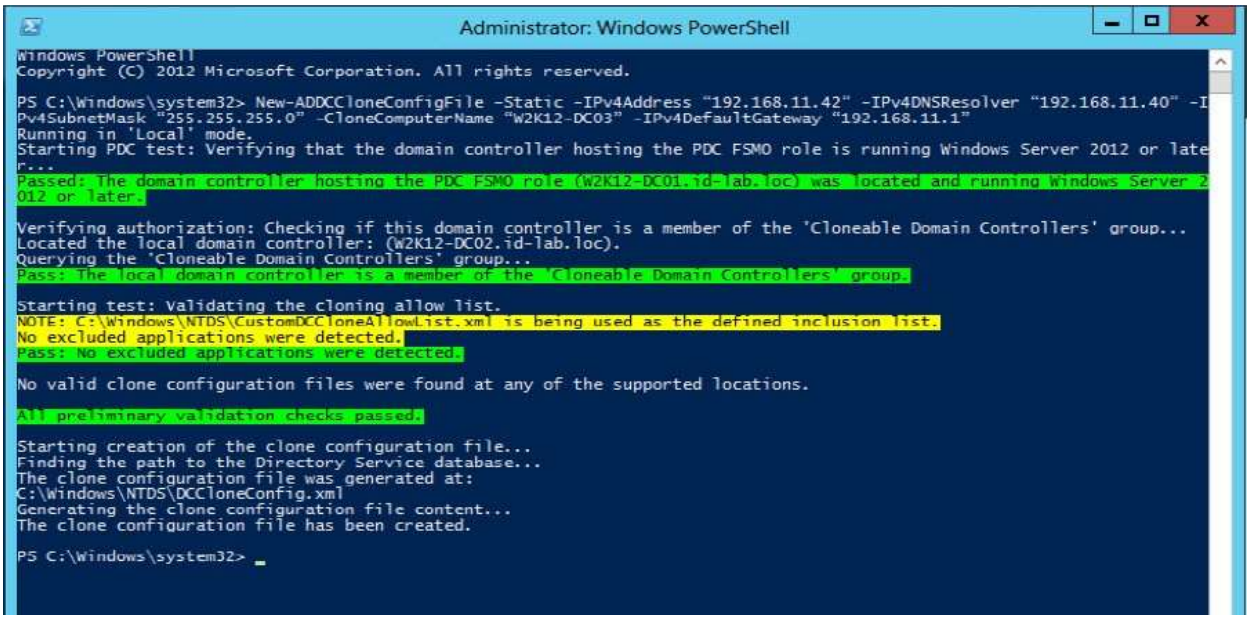
The final preparation step is to run the **New-ADDCCloneConfigFile** PowerShell command. This command performs a set of checks and configuration actions:

- It verifies that any additional software installed on the domain controller has been included in the allow-list.
- It verifies that the domain controller's computer object is a member of the Cloneable Domain Controllers security group.
- It creates the DCCloneConfig.xml file, which contains the identity information required for the new domain controller, such as host name, IP address, and related configuration details.

If either of the first two verification steps fails—meaning either the software list is incomplete or the domain controller is not a member of the required security group—the DCCloneConfig.xml file is not created.

The **New-ADDCCloneConfigFile** command supports an array of parameters that define the identity of the new domain controller. The following figure shows an example set of parameters. For a complete description, the Microsoft TechNet article for New-ADDCCloneConfigFile provides the full reference.

Figure 14 - New-ADDCCloneConfigFile Command



Once the **DCCloneConfig.xml** file has been generated, the source domain controller can be shut down in preparation for cloning.

Note: *Do not use hot cloning of the domain controller's virtual machine to create a production domain controller. Hot cloning places the Active Directory database in a state that is incompatible with domain controller cloning and is not supported.*

After the source domain controller is powered off, the virtual machine can be safely cloned. The new domain controller can be created either by using the cloning operation in the vSphere Client or by copying the VMDK file and building a new virtual machine from it. Although both are possible, using the vSphere Client is the preferred method.

Note: *Do not power on the source domain controller until after the cloning operation has been successfully completed using the vSphere Client (or until the VMDK file has been fully copied).*

When cloning the source domain controller virtual machine in the vSphere Client:

- Do not customize the guest operating system.
- Edit the hardware settings before deploying the clone to ensure network configuration correctness.
- Before powering on the clone, confirm that its network interface (vmnic) is connected to a network segment that has full connectivity to the Active Directory infrastructure. The clone must communicate with other domain controllers to complete the cloning process successfully. Failure to provide proper connectivity may cause the cloning operation to fail and require it to be repeated.

For further details, refer to Virtualized Domain Controller Troubleshooting.

The source domain controller may be powered back on immediately after the virtual machine cloning process has completed.

Once the new virtual machine is powered on, the change in VM-Generation ID activates the domain controller safeguard mechanism. This causes the domain controller to discard its RID pool and to generate a new InvocationID for the local Active Directory database. The presence of the **DCCloneConfig.xml** file signals the intention to clone a domain controller, allowing the cloning process to begin. Upon completion, the newly created virtual machine becomes a new domain controller within the Active Directory environment.

Note: *Because Active Directory uses a tombstone lifetime of 180 days, verify that any domain controllers used solely as cloning sources are powered on and allowed to replicate at least once every 180 days*.

If **Windows Backup** was used to back up the system state of the source domain controller, the backup catalog on the newly cloned domain controller must be deleted. This prevents the system from restoring a point-in-time snapshot taken from the source domain controller, which would be inappropriate for the cloned domain controller.

Best Practices for Virtualizing Domain Controllers

The virtualized Active Directory Domain Services environment requires the same level of planning, diligence, and ongoing maintenance as a physical deployment. However, virtualization introduces a number of additional considerations that must be accounted for to ensure proper operation. These considerations apply especially to the infrastructure layer, where hypervisor behavior and timing interactions can affect the performance and reliability of domain controllers.

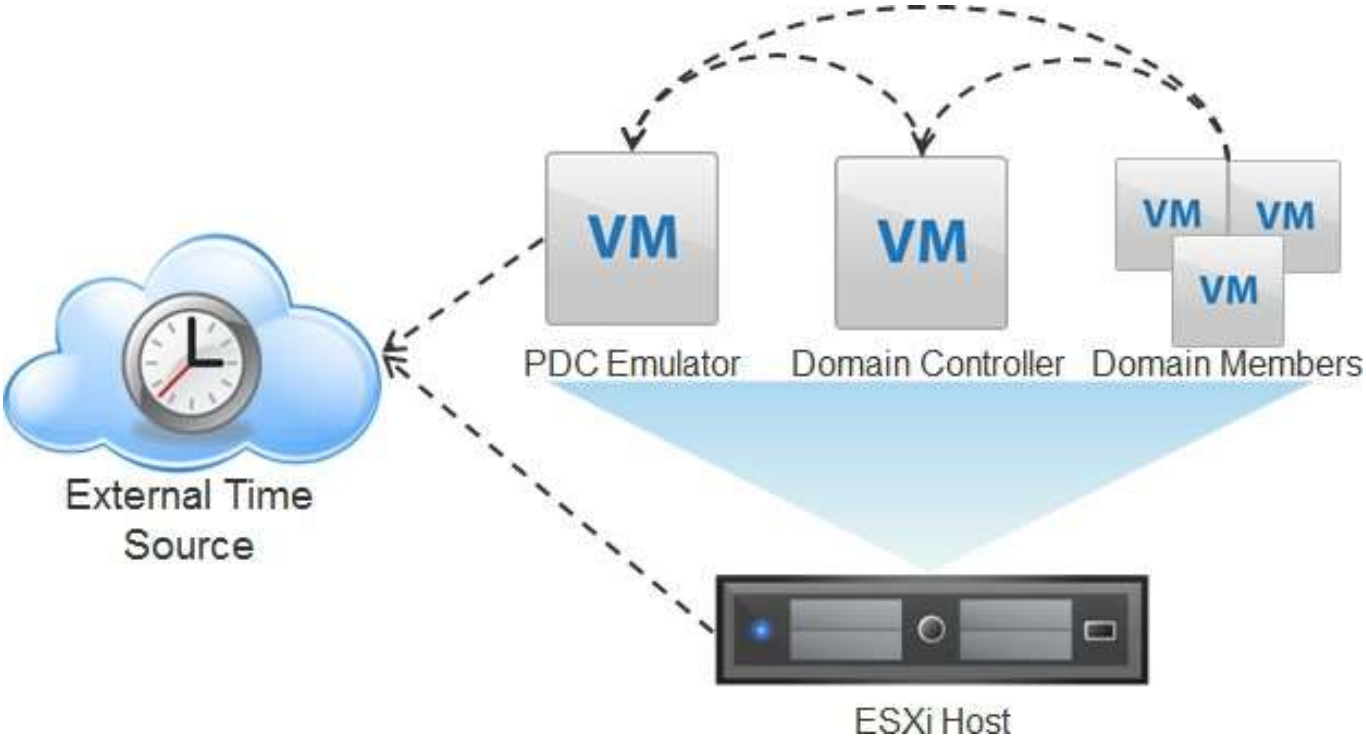
Timekeeping

One of the foundational principles in any Active Directory design is the establishment of a reliable and consistent time source for all domain controllers. Accurate time is crucial because domain controllers depend on synchronized clocks for Kerberos authentication as well as for the arbitration mechanisms involved in Active Directory replication. Although Kerberos itself does not require extremely precise time alignment—the default tolerance is a five-minute maximum deviation across the forest—the use of time as an authoritative factor in replication decisions means that domain controllers should exhibit as little time drift from one another as possible.

The process of ensuring correct timekeeping begins at the hypervisor level. Every ESXi host must be configured to synchronize with a reliable and trusted time source. This time source may be a stratum 1 source, such as a GPS-based or hardware-based clock, or a stratum 2 or 3 source, such as those provided by pool.ntp.org. When multiple time sources are used, they must all exist at the same stratum level to avoid inconsistencies.

In addition—and as illustrated in the following figure—the entire Active Directory forest should rely on the same set of time sources used by the ESXi hosts. ESXi hosts should not synchronize their time from a virtualized domain controller. This is important because, during startup, VMware Tools always performs a one-time synchronization of the guest operating system's clock with the ESXi host's time, even when VMware Tools periodic time synchronization is explicitly disabled. Using the same time sources for both ESXi hosts and the Active Directory forest ensures consistent time across both layers of the infrastructure. This consistency is particularly valuable when ESXi hosts are configured to integrate with Active Directory. It also maintains accurate time within virtualized domain controllers during startup cycles.

Figure 15 - Time Synchronization for ESXi Host and PDC Emulator

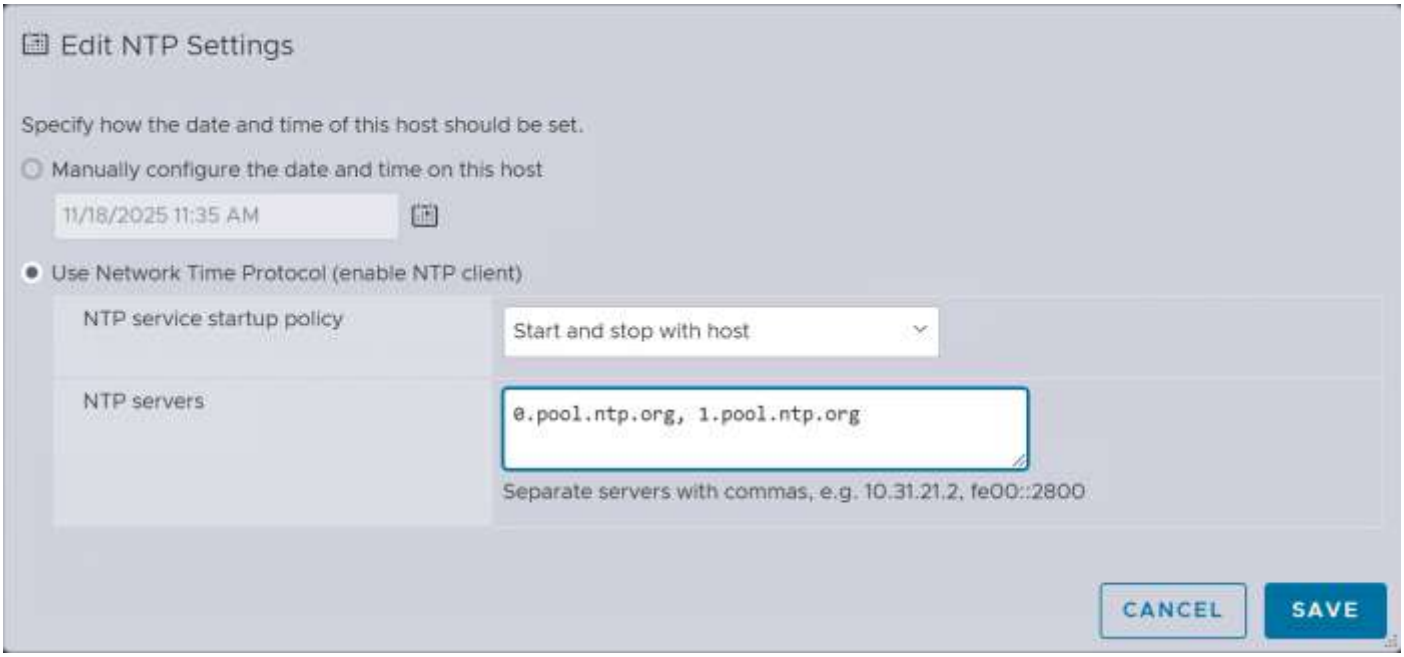


To configure Network Time Protocol (NTP) on an ESXi host, use the vSphere Web Client. The steps are:

- Connect to the host using the vSphere Web Client and navigate to the Manage tab.
- In the menu, select the Settings pane, then choose Time Configuration.
- Click Edit to open the Edit Time Configuration window.
- Select Use Network Time Protocol (Enable NTP client).
- Click Start to start the NTP service.
- Set the NTP Service Startup Policy to Start and stop with host.
- Enter the list of NTP servers as a comma-separated string in the NTP Servers field, as shown in the reference image below.

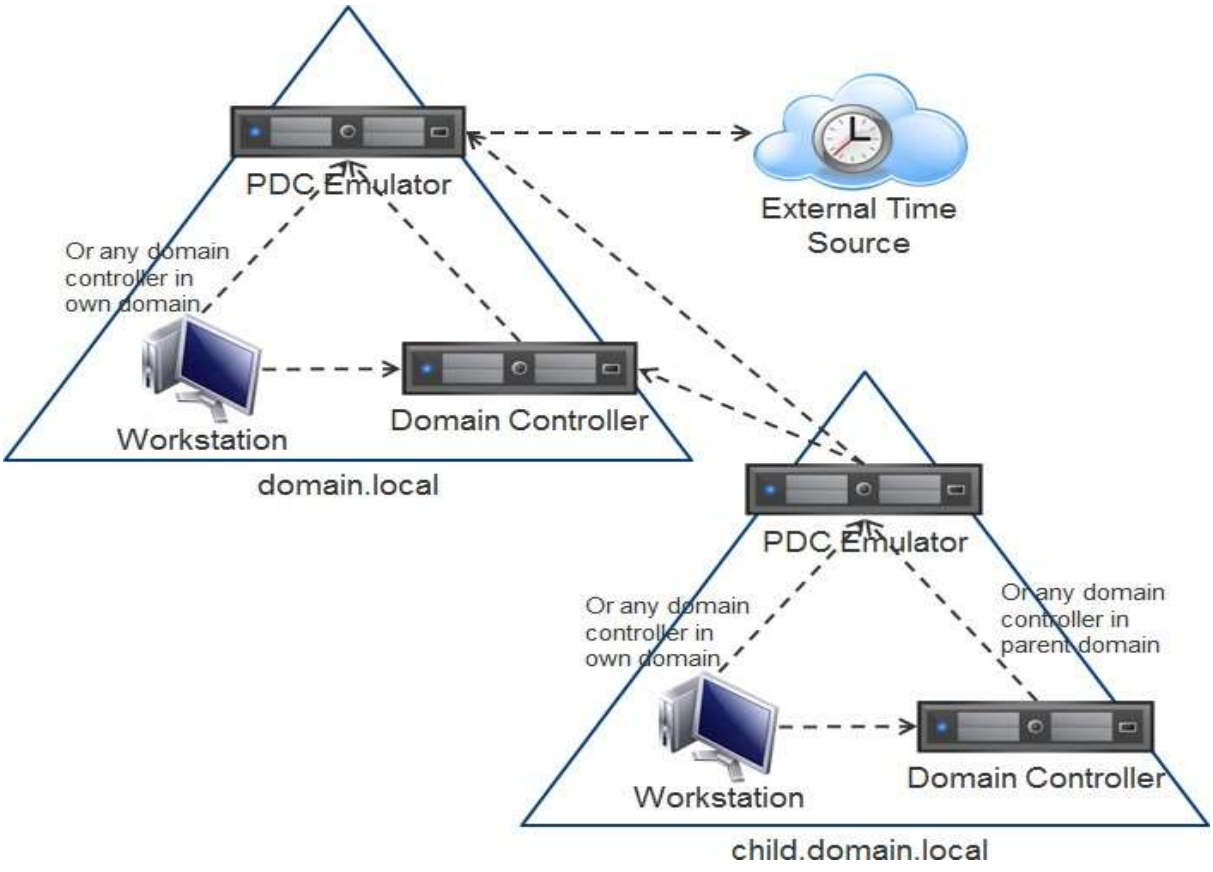
For more information, see Configuring Network Time Protocol (NTP) on ESX/ESXi.

Figure 16 – Configuring ESXi Time Synchronization Setting in vSphere Web Client



The **Primary Domain Controller emulator (PDCe)** in the forest root domain is responsible for ensuring accurate time across the entire forest. The following diagram shows how time synchronization flows throughout a multi-domain Active Directory forest.

Figure 17 - Time Synchronization Using a Domain Hierarchy



The forest root PDCe synchronizes with a reliable, external time source that resides outside the Active Directory forest. Other domain controllers in the forest root domain then synchronize from the forest root PDCe. If the forest root domain contains workstation computers, those systems can synchronize from any domain controller within the same domain. Domain controllers in subdomains within the forest synchronize from domain controllers in their parent domain. Workstations in those subdomains synchronize from the domain controllers in their own local domain.

Because the PDCe operations master role holder in the forest root domain acts as the authoritative time source for the forest, it must be configured to synchronize from a reliable time source—preferably the same time source used by the ESXi hosts. Windows systems, including domain controllers, use the **Windows Time Service (W32Time)** for time synchronization. By default, the PDCe is configured to use the hardware clock or BIOS as its source, so the configuration must be updated to point to the correct external time source.

To configure the forest root PDCe to synchronize with a reliable external time provider, execute the following command:

- *w32tm /config /manualpeerlist:"0.pool.ntp.org,1.pool.ntp.org" /syncfromflags:manual /reliable:yes /update*

The parameters used in this command correspond to the following behaviors:

- **config** – Places w32tm into configuration mode.
- **manualpeerlist** – Specifies the list of reliable NTP sources, in DNS or IP format. When multiple servers are defined, they must be space-delimited and enclosed in quotation marks.
- **syncfromflags** – Indicates the source used by the NTP client.
- **manual** - instructs the NTP client to use entries in the manual peer list.
- **domhier,** which is the default, instructs the client to synchronize using the domain hierarchy.
- **reliable** – Marks the domain controller as a reliable time source.
- **update** – Notifies the Windows Time Service that configuration changes have been made and applies those changes immediately.

When the PDCe operations master role is moved — either transferred normally or seized during recovery — you must execute the same configuration command on the new PDCe. In addition, the domain controller that previously held the PDCe role must be reconfigured so that it stops presenting itself as a reliable time source and reverts to synchronizing based on the domain hierarchy. This is done by running the following command:

- *w32tm /config /syncfromflags:domhier /reliable:no /update*

For more information about managing the Windows Time Service, see Managing the Windows Time Service

Completely Disabling Time Synchronization for Domain Controllers

Certain operations can cause a virtual machine's clock to reset even when the Synchronize Guest Time with Host option is unchecked for the VM. The VMware knowledge base article "Disabling Time Synchronization for virtual machines" provides more detail about this behavior. The document "Timekeeping in VMware Virtual Machines" offers a thorough explanation of the timekeeping components and behaviors within a VCF environment.

To ensure that ESXi does not reset the clock of a virtualized domain controller under any circumstances, it is necessary to adjust the domain controller VM's advanced configuration options. This is done by manually adding specific key-value pairs to the Advanced Configuration section of the virtual machine's properties. These additions override the default behavior and prevent ESXi-initiated time adjustments on that virtual machine.

Certain operations will reset a virtual machine's clock – even when the "Synchronize Guest Time with Host" option is unchecked for that virtual machine. Please see "Disabling Time Synchronization (1189)" for further description.

To completely ensure that ESXi does not reset a virtualized Domain Controller clock under any circumstances, you must manually adjust the default behavior by adding the following key and value pairs to the Advance Configuration options of the Domain Controller's VM Properties:

Table 2 - Options to Manually Disable Time Synchronization on a VM

tools.syncTime	0
time.synchronize.continue	0
time.synchronize.restore	0
time.synchronize.resume.disk	0
time.synchronize.shrink	0
time.synchronize.tools.startup	0
time.synchronize.tools.enable	0
time.synchronize.resume.host	0

You can also directly add these entries to the VM's .vmx file.

vSphere HA

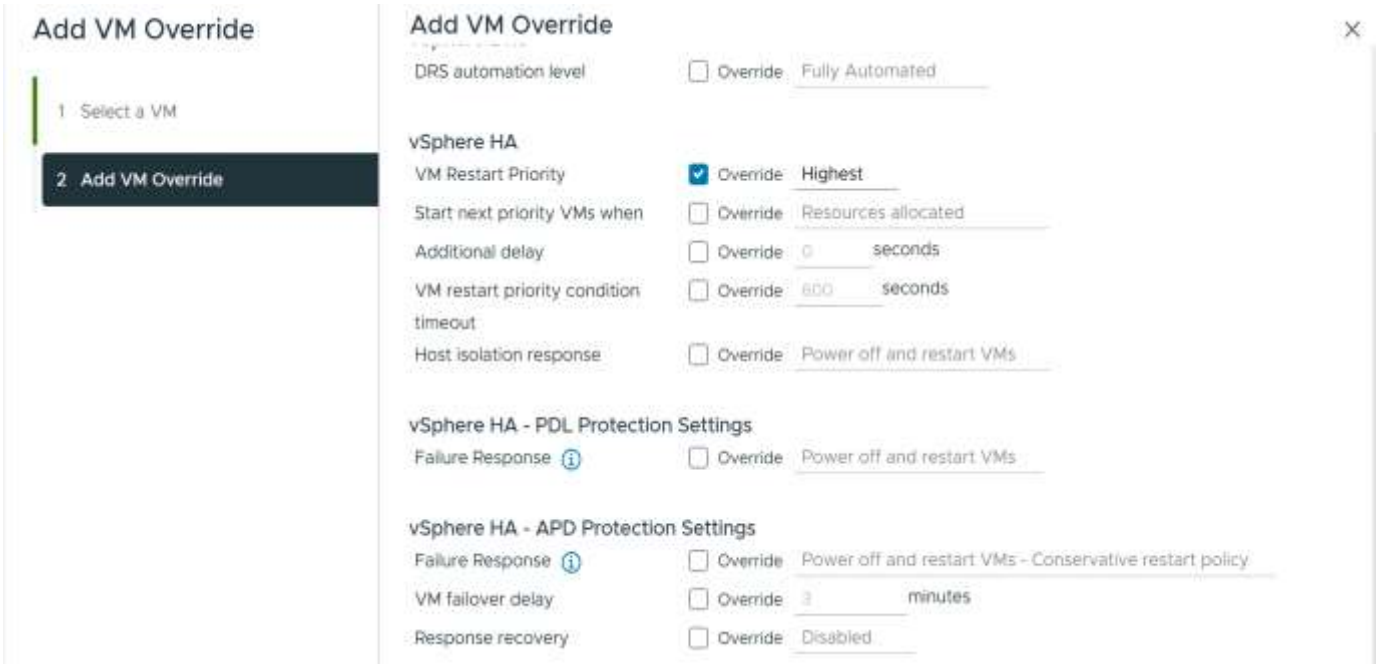
VMware recommends that virtualized domain controllers make full use of both VMware vSphere High Availability (HA) and VMware vSphere Distributed Resource Scheduler (DRS). When vSphere HA is enabled, virtual machines running on an ESXi host that experiences a failure—whether the failure involves the host itself or a critical component such as storage—undergo a hard shutdown. Immediately afterward, vSphere HA automatically restarts those virtual machines on other ESXi hosts within the same vSphere cluster. This automated restart greatly reduces recovery time, typically bringing virtual machines back online within minutes. This level of responsiveness substantially improves availability compared to waiting for an administrator to receive an alert, diagnose the issue, and manually restart affected services.

vSphere HA also offers several features that are specifically helpful when managing virtualized domain controllers.

VM Restart Priority

By default, vSphere HA restarts virtual machines in no particular order following a host failure. In many environments, where domain controllers are deployed in a scaled-out manner, a temporary outage of a single domain controller may be acceptable. However, in situations where it is important to prioritize the recovery of certain domain controller virtual machines, vSphere HA allows administrators to assign a higher restart priority to specific VMs. Doing so ensures they come online earlier during the HA recovery process. The configuration of a virtual machine for a high-priority restart is shown in the following image.

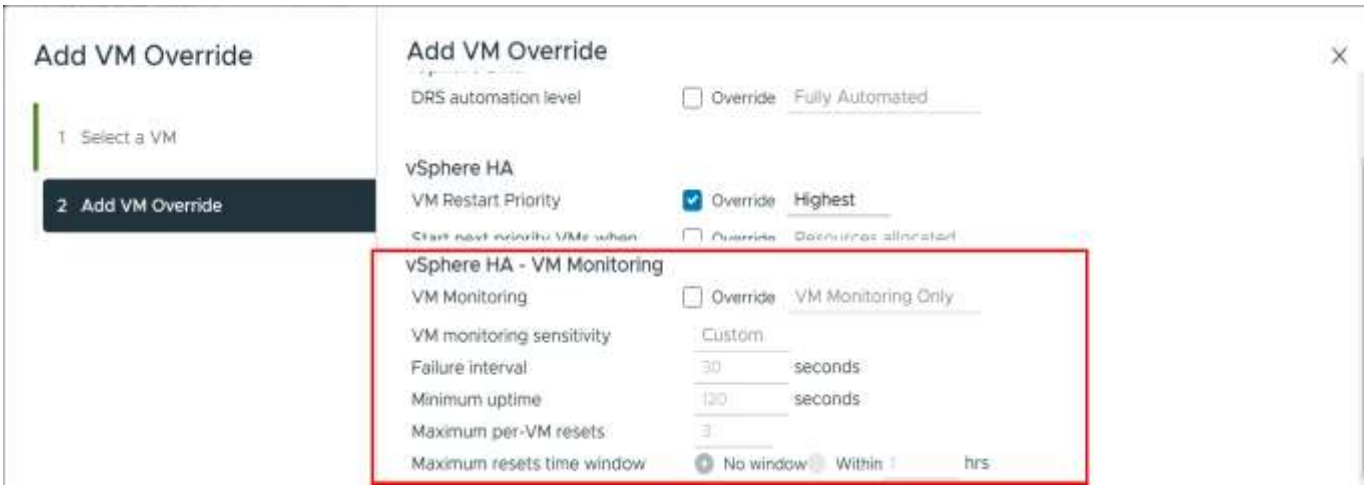
Figure 18 – Enabling Virtual Machine Restart Priority in vSphere HA



VM Monitoring

Another vSphere HA feature valuable for virtualized domain controllers is **VM Monitoring**. When VM Monitoring is enabled, VMware Tools inside the guest operating system sends periodic heartbeats to the ESXi host. If the heartbeat stops, vSphere HA performs additional checks to determine whether the guest operating system has actually stopped responding or whether the loss of heartbeat is limited to the VMware Tools channel. Specifically, vSphere HA checks for activity on the VM's network and storage interfaces. If ESXi detects that storage I/O is unavailable or that the virtual machine is otherwise unresponsive, HA initiates a restart of the virtual machine. VM Monitoring can therefore provide an additional recovery mechanism for Windows-based virtual machines. Enabling VM Monitoring is illustrated in the figure below.

Figure 19 - Enabling Virtual Machine Monitoring in vSphere HA



vSphere DRS

vSphere DRS manages compute resources within a vSphere cluster by balancing workloads based on each virtual machine's utilization and resource entitlement. As cluster conditions evolve—for example, when ESXi hosts enter

maintenance mode or when new compute capacity becomes available—DRS evaluates which hosts can provide the most appropriate resources for each virtual machine. In fully automated mode, DRS uses vSphere vMotion to move virtual machines between ESXi hosts without any downtime, maintaining availability and performance.

DRS also includes features that control the placement of virtual machines. Using DRS rules and combinations of virtual machine and host groups, administrators can define highly granular placement behavior. For example, DRS anti-affinity rules can ensure that domain controller virtual machines do not run simultaneously on the same ESXi host. By distributing domain controllers across different hosts—or across different blade chassis or rack locations—administrators can reduce the risk that a localized hardware failure will impact multiple domain controllers at the same time.

When using vSphere DRS with clusters that host virtualized domain controllers, consider the following recommendations:

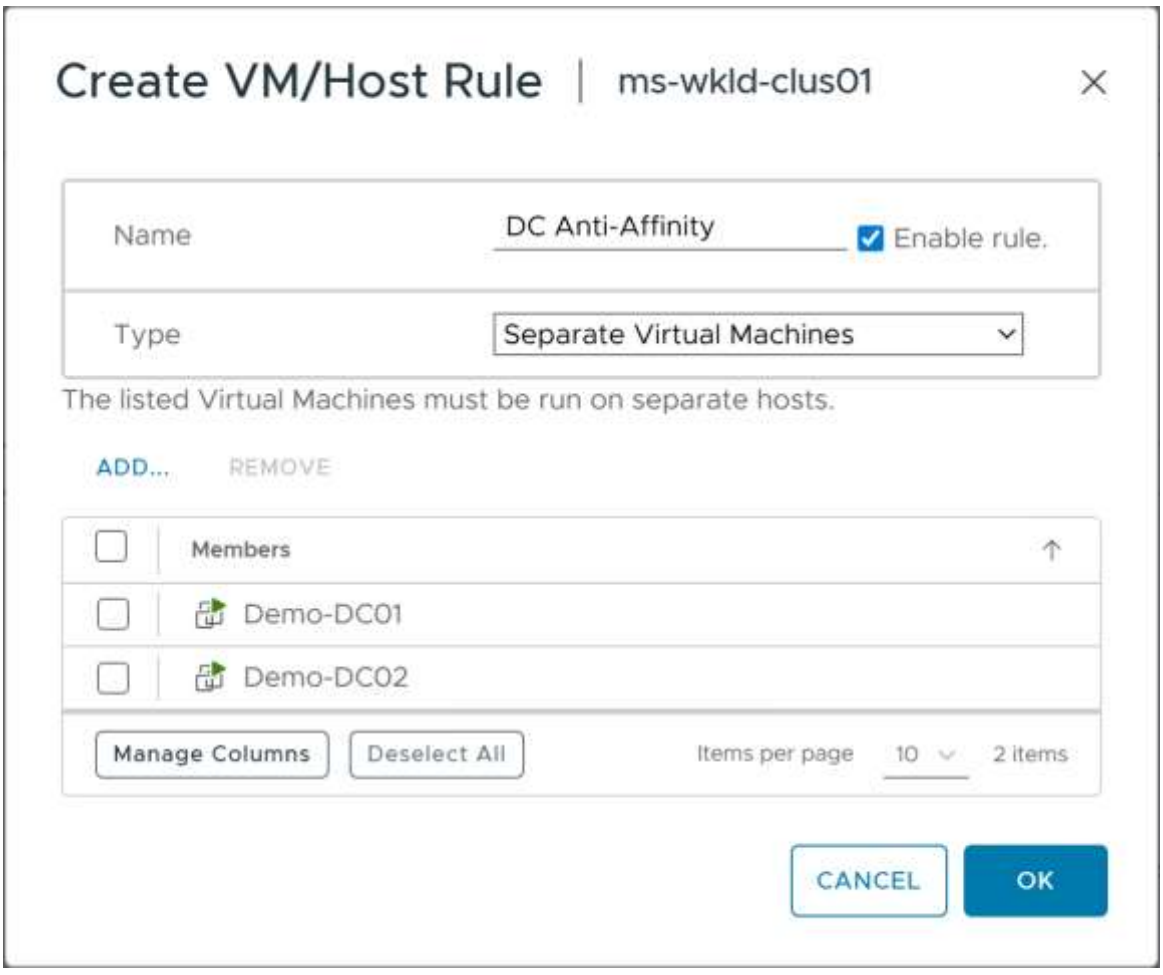
- **Enable vSphere DRS in Fully Automated Mode**

  Configuring DRS in fully automated mode ensures that the system automatically applies its placement and load-balancing decisions. This mode also ensures that any violations of DRS rules are corrected during the next DRS evaluation cycle, which occurs every five minutes.

- **Create DRS Anti-Affinity Rules**

  Although running two domain controllers on the same ESXi host may be acceptable in environments where more than two domain controllers exist at the site, VMware recommends separating them whenever possible. Creating DRS anti-affinity rules allows administrators to group one or more domain controller virtual machines and instruct DRS to keep them on separate ESXi hosts. The following figure illustrates such a rule.

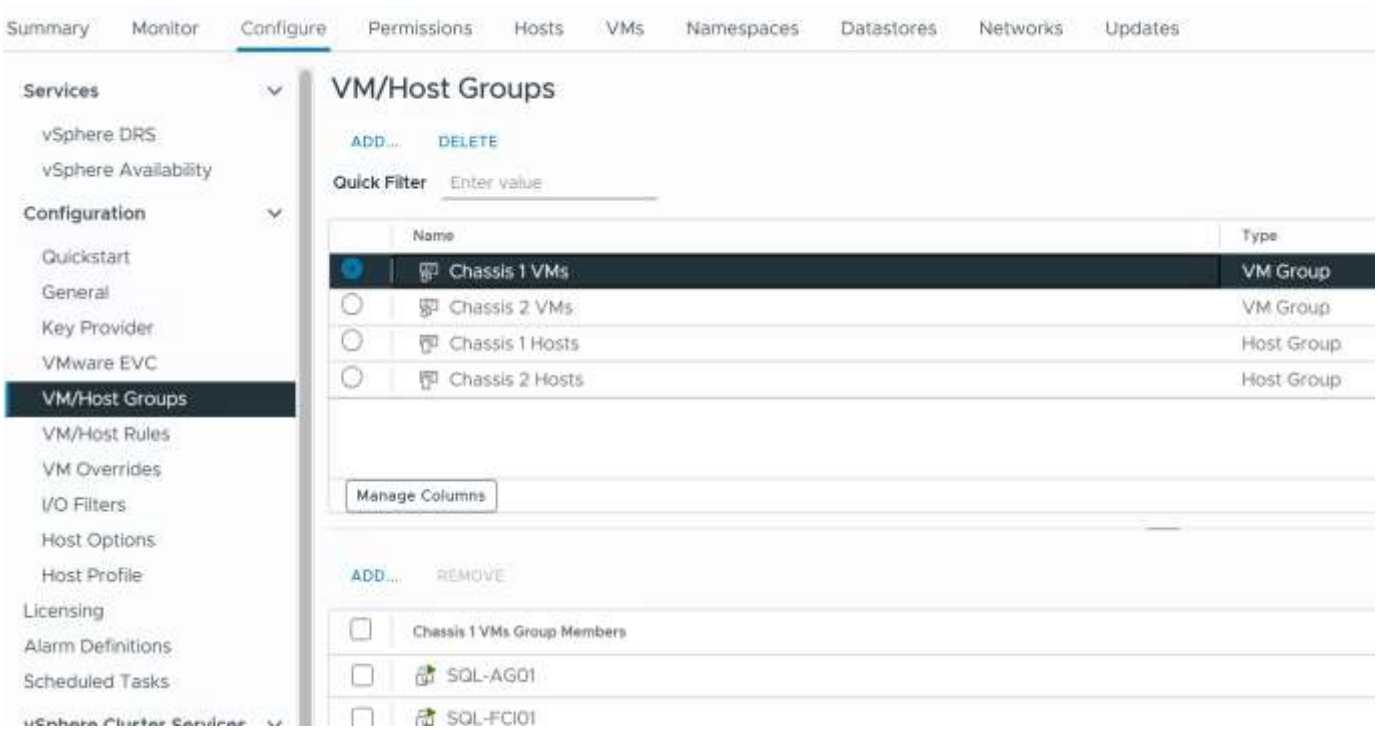
Figure 20 - DRS Anti-Affinity Rule



- **Create DRS Host and Virtual Machine Groups**

  Within a vSphere cluster, administrators can create logical groupings of hosts and virtual machines. These groups can then be combined with affinity or anti-affinity rules to control placement. If the cluster spans more than one failure domain, such as multiple blade chassis, administrators may choose to create one host group per chassis, along with two virtual machine groups representing different sets of domain controllers. Virtual machine–to–host rules can then be applied to ensure that these groups of domain controllers remain distributed across the appropriate hosts. This configuration is shown in the figure below.

Figure 21 - Virtual Machine and Host DRS Groups

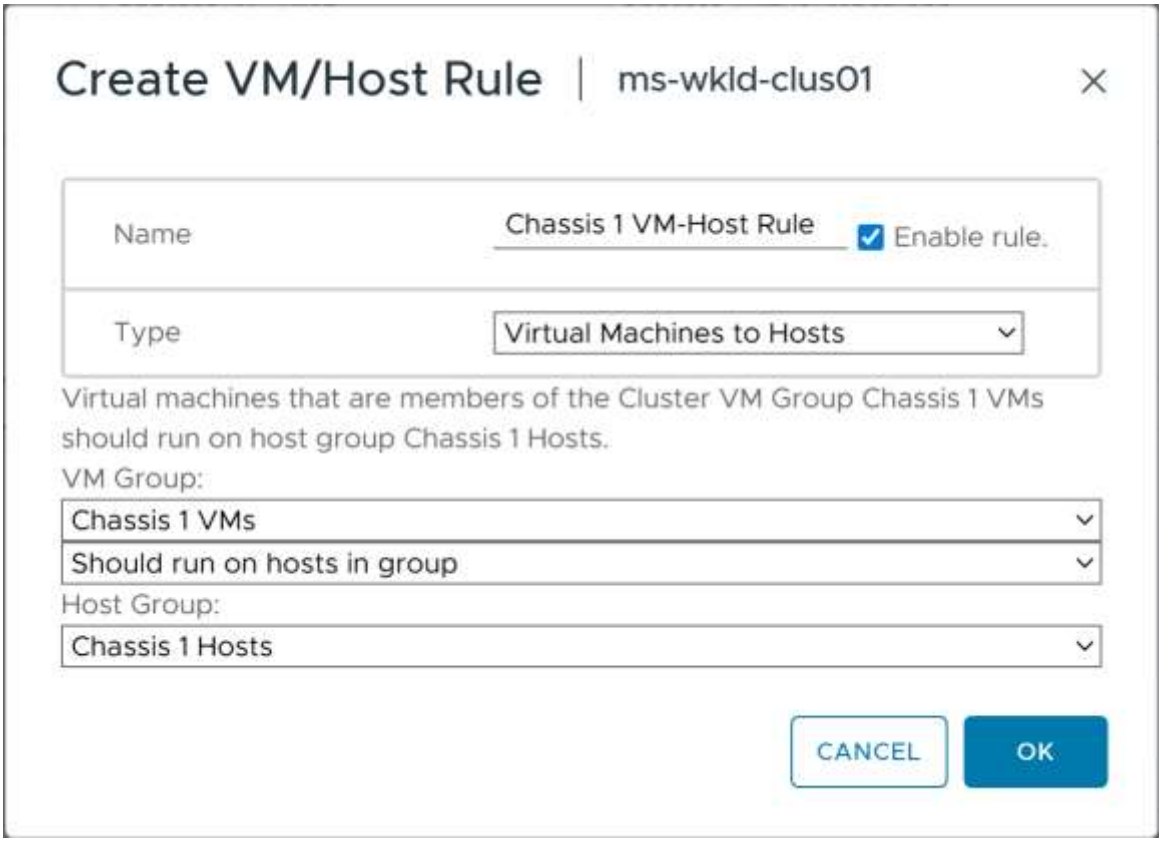


- **Create Soft Virtual Machine-to-Host Rules**

  DRS supports two types of placement rules for matching virtual machines to hosts: **soft (should-run-on)** rules and **hard (must-run-on)** rules. A soft rule expresses a preference; DRS will honor it whenever possible, but will violate it if necessary. For example, if an entire chassis fails and the virtual machines associated with that chassis have soft rules binding them to the affected hosts, those virtual machines can still be restarted on remaining hosts—even though doing so violates the rule.

  Hard rules, on the other hand, cannot be violated without administrator intervention and are typically reserved for situations such as licensing constraints. For virtualized domain controllers, soft anti-affinity rules are preferred because they allow flexibility in failure scenarios. The following figure illustrates a soft "should-run-on" rule.

Figure 22 - Should-Run-On DRS Rule



Domain Controller Golden Templates

The rapid deployment of domain controllers has long been one of the primary motivations for virtualizing these systems. Historically, the process of bringing a new domain controller online required administrators to perform a number of manual steps. They needed to prepare the base operating system and ensure that all prerequisites for promotion were in place, run **dcpromo** to promote the server to a domain controller, install any additional software required for monitoring, backup, or management, and finally wait for replication to converge before placing the new domain controller into production. When multiple domain controllers were required, these repetitive tasks could consume hours or even days, depending on the size and complexity of the environment.

With the advent of the domain controller cloning functionality introduced in Windows Server 2012 and supported in VCF, much of this manual effort has been eliminated. Cloning provides a streamlined and repeatable method of deploying new domain controllers based on an existing, properly prepared reference virtual machine. When using this method, consider the following recommendations:

- **vCenter Server and vSphere Version Requirements**

  For domain controller cloning to function properly, both vCenter Server and vSphere must be running version **5.0 Update 2 or later**. Earlier versions do not provide the necessary support for the cloning operations or the related VM-Generation ID functionality.

- **Validate All Installed Software for Cloning**

  Any software installed on the source domain controller—especially software included in a golden template—must first be validated to ensure it can safely be cloned. As noted earlier, some applications use unique identifiers tied to an individual machine, which might not behave correctly after being duplicated. Administrators should confirm with the relevant software vendors whether each application can be installed safely in a domain controller template. When software behavior is uncertain, consider removing it before proceeding with cloning. VMware Tools is confirmed to be safe to install prior to cloning.

- **Maintain an Empty Cloneable Domain Controllers Security Group (Between Cloning Events)**

For security purposes, the Cloneable Domain Controllers global security group must remain empty when no cloning operations are underway. Membership in this group designates domain controllers that are eligible for cloning, so keeping it empty when not actively cloning ensures that unauthorized or accidental cloning cannot occur.

- **Keep the Reference Domain Controller Powered On**

Any domain controller designated as a reference source for future cloning must remain powered on and connected to the network when cloning operations are not taking place. Leaving a domain controller offline for longer than the Active Directory tombstone lifetime—180 days by default—prevents proper replication and leads to replication issues that must be manually repaired before synchronization can resume. Keeping the reference domain controller online prevents this condition.

- **Remove Backup History After Cloning**

If Windows Backup is used to back up the system state of the template domain controller, administrators should delete any backup catalog or history on the newly created domain controller immediately after cloning. A fresh backup of the new domain controller should then be performed. This prevents a newly cloned domain controller from referencing an outdated, point-in-time backup that belonged to the template domain controller.

- **Shut Down the Source Domain Controller Before Cloning**

Before initiating the cloning operation, the source domain controller must be shut down. Although hot cloning is a mature and widely used feature within vSphere, allowing virtual machines to be copied while powered on, it is not supported for provisioning new production domain controllers. The reason is that the Active Directory database must be shut down in a clean and consistent state so the cloning process can be validated properly. The decision to shut down the virtual machine is tied to how Active Directory determines whether domain controller safeguards should be invoked. As a result, VMware and Microsoft do not support hot cloning when the intent is to create a new production domain controller.

Disaster Recovery of Domain Controllers

When Active Directory is designed correctly, the same multimaster replication architecture and service location mechanisms that provide high availability within a single site also support resilience across sites and readiness for disaster recovery scenarios. Achieving this level of resiliency depends on implementing a design in which domain controllers are deployed across geographically dispersed locations. By distributing domain controllers in this way, the environment gains the ability to continue functioning even if an entire site becomes unavailable.

The portability of virtual machines enhances disaster recovery capabilities by allowing administrators to replicate the virtual machine files to another data center. In the event of a disaster, a replica virtual machine can be powered on and—after minimal configuration—returned to service. Historically, the primary concern with this approach was ensuring that virtual machine files remained up to date. Modern versions of vSphere address this concern through two replication methods: **VMware vSphere Replication** and array-based replication. Both replication approaches can be integrated with **VMware Live Site Recovery (VLSR)**, which automates the recovery and configuration of virtual machines during a disaster event, a planned failover, or even a planned migration.

This raises the question of how these capabilities can be used with virtualized domain controllers. The most common answer is that Active Directory's built-in multimaster replication typically provides sufficient protection on its own, especially when a minimal number of domain controllers are deployed in each site. Because domain controllers generally impose low performance and maintenance demands, this configuration usually presents no significant overhead. However, there are certain cases in which tools such as **VMware Live Site Recovery (VLSR)** can be advantageous for recovering domain controllers, particularly in controlled test environments or unique disaster recovery workflows.

Using Domain Controllers During Disaster Recovery Testing

A major benefit of VMware Live Site Recovery is its ability to test disaster recovery plans without affecting the production environment. This is achieved by using isolated networks and point-in-time copies of virtual machines, which may come from array-based replication or from virtual machine snapshots. When a recovery plan is executed in test

mode, VLSR automatically recovers the designated virtual machines in a manner consistent with the instructions contained in the plan. These instructions may specify the order in which virtual machines are recovered, special configuration steps, or temporary changes to network connectivity. Because the entire process is isolated from production, administrators can verify that applications behave as expected when recovered.

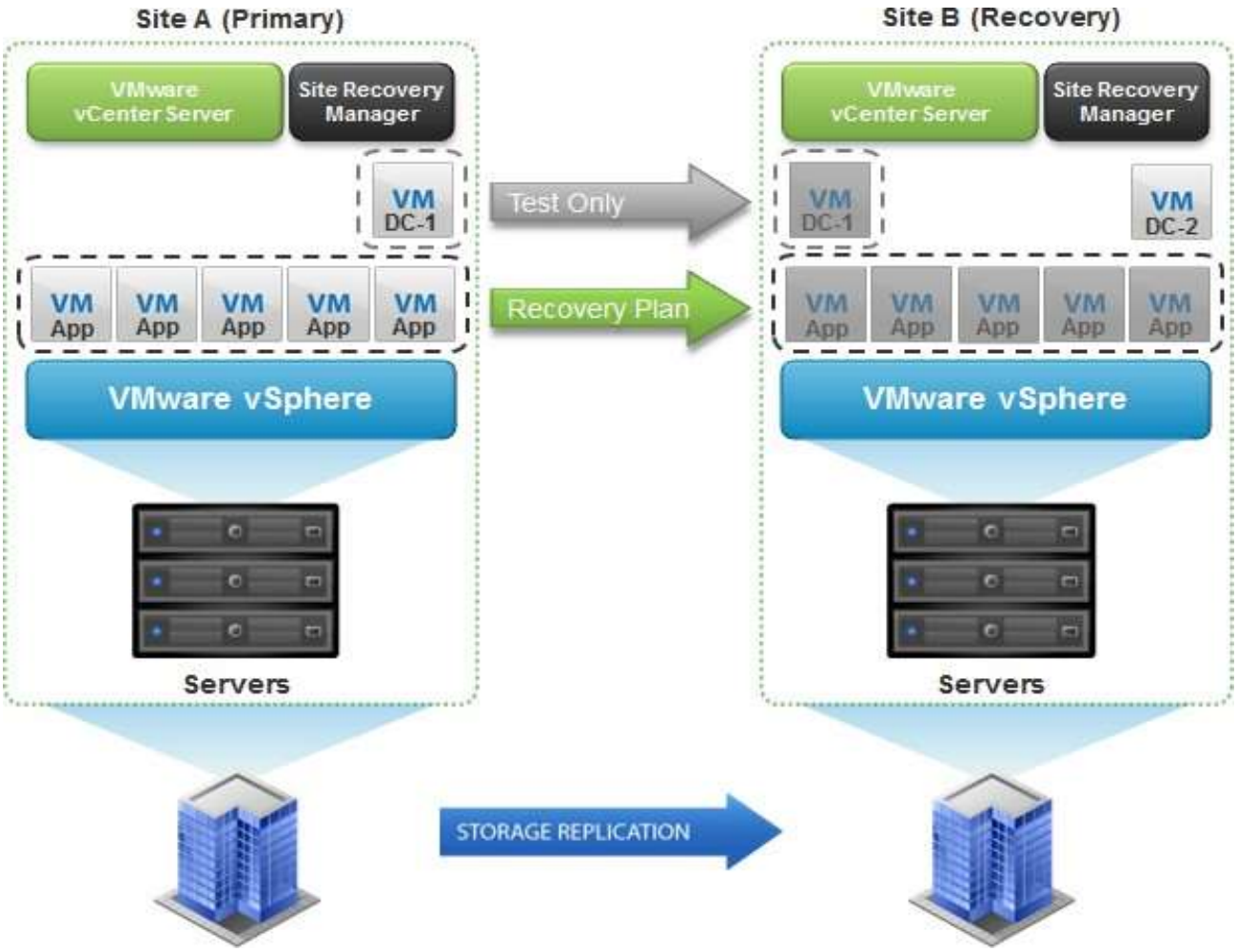
Some applications require the presence of Active Directory Domain Services—or DNS—to start correctly. In these cases, administrators have two options when incorporating Active Directory into VLSR test plans. The first approach is to include a domain controller from the protected site directly within the test recovery plan. The second approach is to clone a domain controller into the isolated recovery environment at the time the test plan executes.

- Option 1: Using a Primary Site Domain Controller During Test Execution

  In the first approach, which is automated and self-contained, what occurs during a test failover closely follows a predefined plan. As illustrated in the following figure, the application virtual machines participate in two recovery plans—a production recovery plan and a test recovery plan. The application requires Active Directory to start, but during a test failover the application virtual machines are recovered to an isolated network that does not have access to Active Directory.

  To satisfy this requirement, the test recovery plan includes DC-1, a domain controller that is replicated to Site B solely for the purpose of disaster recovery testing. When the test recovery plan is triggered, VLSR powers on the application servers along with DC-1. This allows the application to come online within the isolated environment. In a real disaster scenario, only the application servers fail over, and through the standard DC Locator process they discover DC-2, the domain controller that exists at the recovery site for production use.

Figure 23 - Using Primary Site Domain Controller During Recovery Plan Testing



- **Option 2: Cloning a Domain Controller for Use During Test Execution**

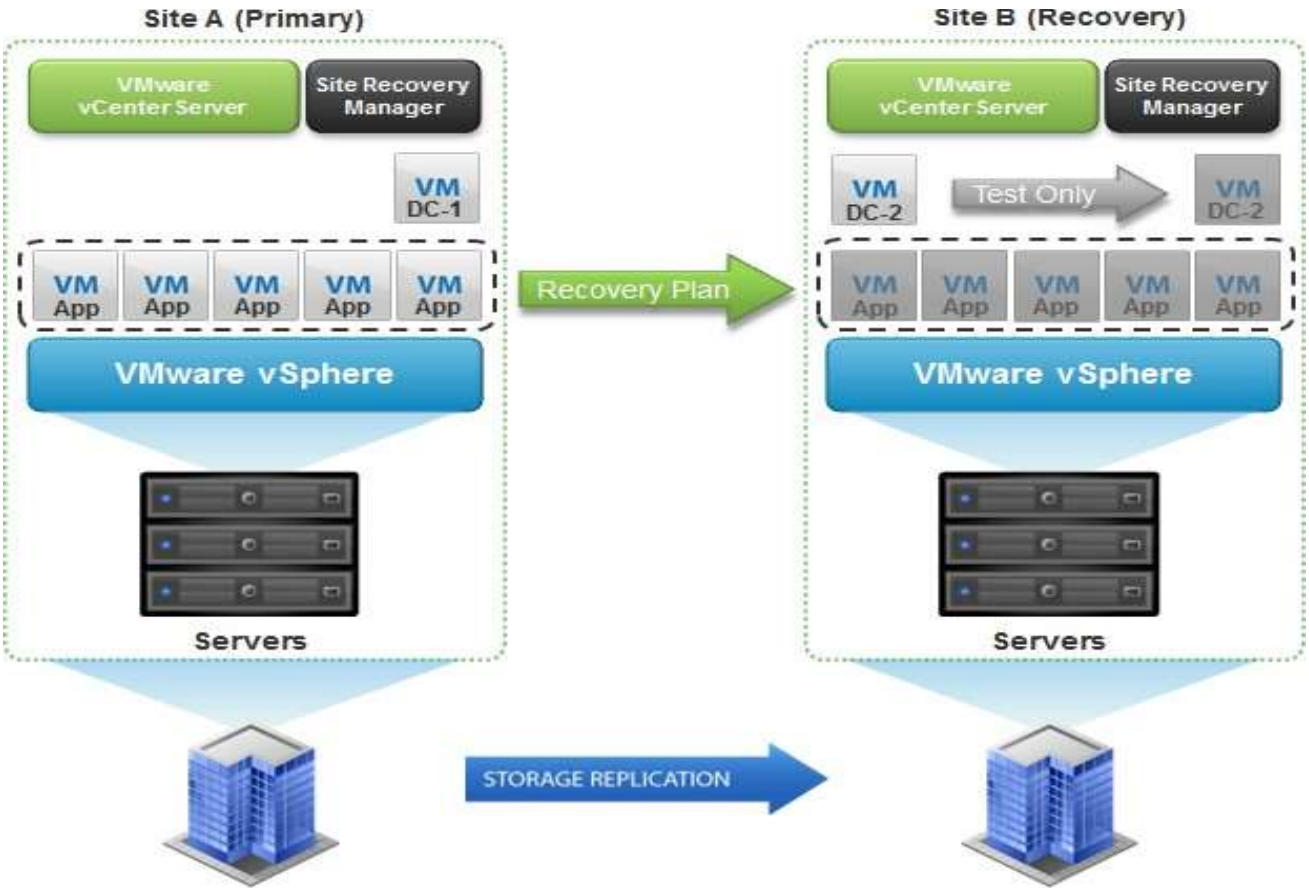
The second approach is to clone a local domain controller at the time the test recovery plan is executed. The method used to clone the domain controller depends on the operating system version of the source domain controller:

- The clone may be an online (hot) clone, although this is not supported for domain controllers.
- The clone may be created through an offline and scripted process.
- The clone may also be created through a fully manual process.

During test execution, the cloning operation is performed first. Once cloning is complete, VLSR brings the application servers online in the isolated environment. Using the DC Locator process, the application servers locate DC-2, enabling the application to start successfully. After the test completes, the cloned domain controller is deleted.

This option mimics what could occur during an actual disaster, but it requires additional scripting or manual work to create and later remove the cloned domain controller.

Figure 24 - Cloning Recovery Site Domain Controller During Recovery Plan Testing



Protecting Operations Master Role Holders

The Operations Master role holders—commonly referred to as the FSMO (Flexible Single Master Operations) role holders—perform specific, essential tasks within Active Directory that cannot be executed by more than one domain controller at a time. Each role provides a unique service, and because these functions are single-master by design, they are dependent on the continued availability of the domain controller currently holding the role. For example, the **Schema Master** is the only domain controller on which modifications to the Active Directory Schema can occur. If that domain

controller is offline when a Schema change is attempted, the operation simply fails because no other domain controller has the authority to perform that action.

Although these roles can be transferred to another domain controller—either gracefully through a controlled transfer or forcefully through a seizure—doing so in the middle of an unplanned outage or recovery event may not be the preferred or ideal scenario. Active Directory includes five Operations Master roles in total. Two of these are forest-wide roles, while the remaining three are domain-wide roles. By default, all five roles are assigned to the first domain controller created in a new forest, and the three domain-wide roles are assigned to the first domain controller created in each domain. Best practice guidance recommends distributing these roles across at least two domain controllers within a domain to provide redundancy and avoid placing all role-dependent operations on a single server.

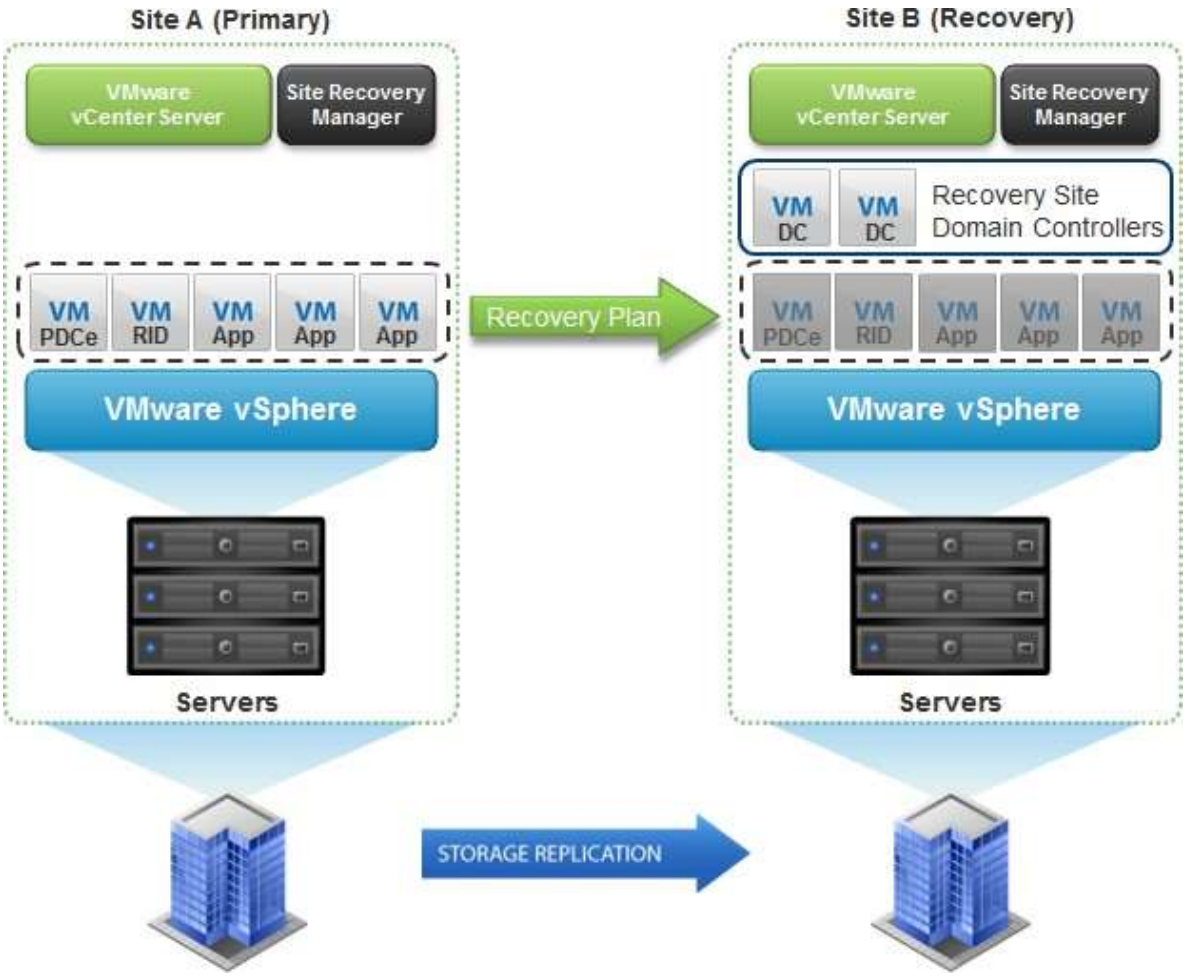
In a disaster recovery scenario, the recommended approach is to maintain a set of domain controllers at the recovery site and rely on standard Active Directory replication to protect the Operations Master role holders. Whether all five roles should be protected depends on the specific needs and characteristics of the environment. However, at a minimum, it is strongly advised to protect the **RID Master** and the **PDC Emulator (PDCe)** roles.

The PDC Emulator plays a critical part in several important functions. It is the default domain controller that handles password verification and reset operations, processes Group Policy changes, and acts as the authoritative time source for the domain. The RID Master is equally important because it controls the issuance of RID pools—blocks of relative identifiers used when creating new security principals such as users, computers, and security groups. Additionally, if you intend to use VM-Generation ID and the associated virtualization safeguards, the RID Master must be available and protected.

During a recovery event in which virtualization safeguards are applied—for instance, when a domain controller is restored through replication—the restored domain controller automatically discards its local RID pool. After doing so, it contacts the RID Master to request a new pool. If the RID Master is unavailable, the recovered domain controller cannot create any new security principals until the RID Master is reachable again. In many environments, the RID Master and PDC Emulator roles reside on the same domain controller, which simplifies the task of protecting these roles because safeguarding one server protects both functions simultaneously.

The remaining Operations Master roles generally have less immediate impact on disaster recovery operations. If an extended outage occurs, these roles can be seized onto another domain controller when necessary. However, protecting the RID Master and PDC Emulator roles ensures that the most operationally significant functions remain available during recovery events.

Figure 25 - Protecting Operations Master Role Holders



Conclusion

- Many organizations are steadily progressing toward fully virtualizing their data centers, extending virtualization not only to general-purpose workloads but also to business-critical and foundational infrastructure services. Historically, achieving complete virtualization posed challenges, especially for applications and services considered essential to day-to-day operations. Concerns often centered around reliability, supportability, and the potential operational risks associated with placing key services into a virtualized environment. Over time, however, many of these challenges have been addressed. Improvements in virtualization technologies, combined with a broader availability of authoritative guidance on how to virtualize critical applications correctly, have enabled organizations to move forward with far more confidence.
- Active Directory remains the central directory service and authentication backbone for a large number of environments. While partial virtualization of domain controllers has been common for years, many organizations have continued to exercise caution when considering the virtualization of all domain controllers across their infrastructure. This hesitation has typically been rooted in historical limitations, concerns about safety and reliability, or uncertainties about support for certain virtualization-related operations.
- With the introduction of Windows Server 2012, virtualized domain controllers gained access to new capabilities—specifically, support for virtual machine snapshots and domain controller cloning—that were previously unavailable or considered unsafe. These capabilities, combined with the safeguards built into the operating system and the supporting hypervisor features, make snapshot and cloning operations reliable and fully supported when implemented correctly. As a result, organizations now have a practical and dependable path toward deploying a completely virtualized Active Directory environment.
- When the appropriate best practices are followed and when the virtualization platform and operating system features are used as intended, complete virtualization of an Active Directory infrastructure is not only achievable but can be done in a manner that supports reliability, predictability, and operational integrity.

Appendix A: Testing Domain Controller Cloning

This exercise outlines a detailed process for testing and verifying the successful cloning of a virtualized domain controller within a VCF environment. It is intended as a practical guide for administrators who want to confirm that their configuration, preparation steps, and cloning workflow function correctly from end to end. For comprehensive procedural information on domain controller cloning, please refer to Virtualized Domain Controller Deployment and Configuration.

In the scenario described here, several prerequisite tasks have already been completed in order to establish a known baseline:

- **VMware Cloud Foundation** has been fully installed and is operational.
- An **Active Directory** domain has been created.
- All **operations master roles (FSMO roles)** are currently held by a single domain controller.
- A **second domain controller** has been deployed and successfully joined to the domain.

These prerequisites ensure that the environment is stable, functional, and correctly configured before testing the cloning process.

- **Prepare the Source Domain Controller**

Preparing the source domain controller involves several important steps. These include validating that the hypervisor supports the VM-Generation ID feature, confirming which applications are installed on the virtual machine, and enabling the domain controller for cloning by assigning the appropriate security group membership. Each of these steps must be completed successfully before the cloning process begins.

- **Validate the Hypervisor Supports VM-Generation ID**

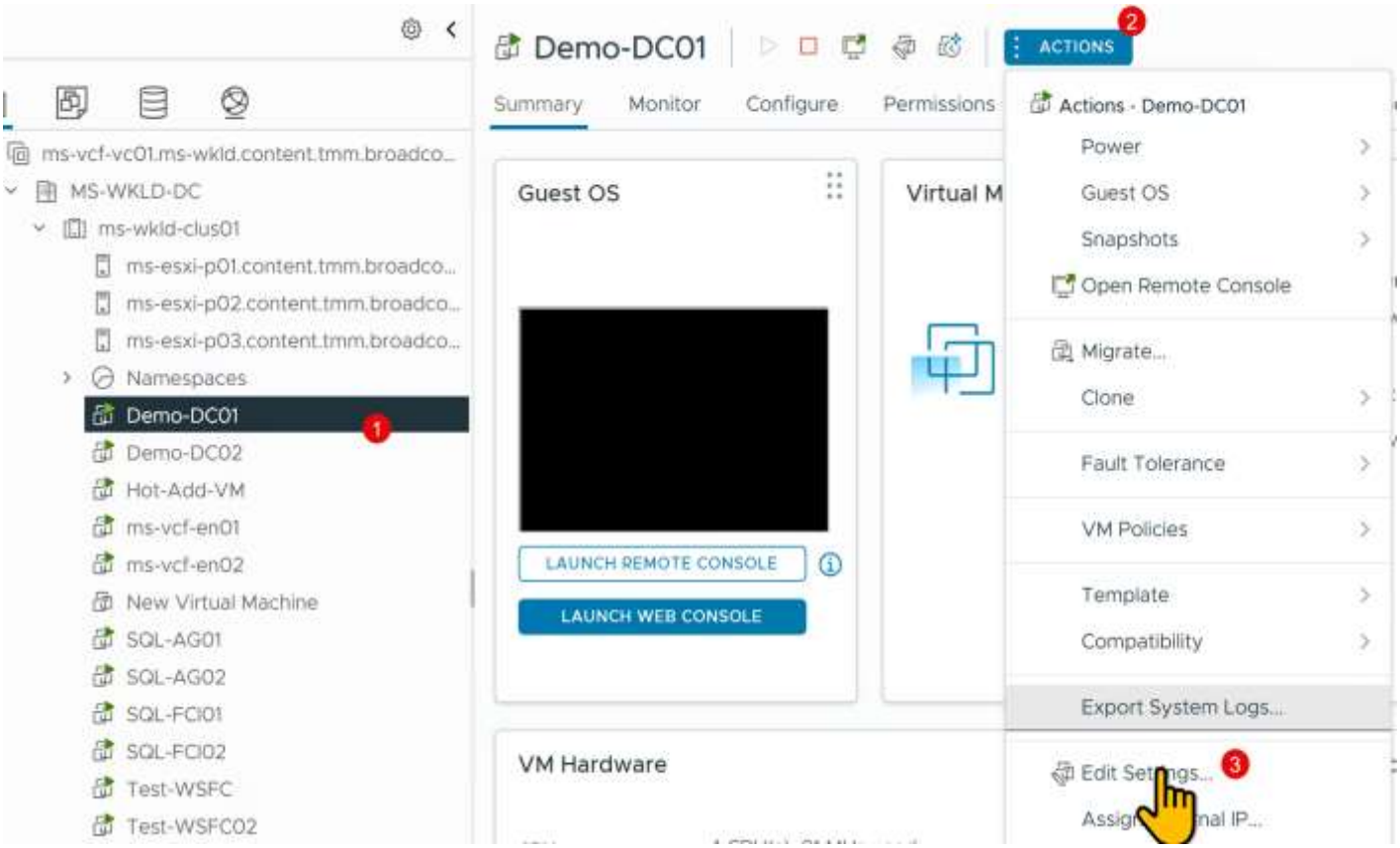
There are multiple ways to confirm whether the hypervisor supports the VM-Generation ID capability. The method described here focuses on checking the value stored within the virtual machine's configuration settings. This check is particularly straightforward if the source domain controller virtual machine is powered off, because the VM-Generation ID value is then visible in the advanced configuration parameters.

In vSphere, the hypervisor records the virtual machine's generation ID using one of two configuration parameters, depending on the vSphere version in use: **vm.genidX** or **vm.genid**. These parameters store the current generation identifier for the virtual machine and are used by Windows Server domain controllers to detect changes in virtual machine state for cloning and safeguard purposes.

To validate the VM-Generation ID through the vSphere Web Client, complete the following steps:

1. Log in to the **VMware vSphere Web Client** and navigate to the virtual machine object representing the source domain controller.
2. From the **Actions** menu, select **Edit Settings** to open the configuration window for the virtual machine.

Figure 26 - Edit VM Settings



3. From the **Edit Settings** screen, select the **"Advanced Parameters"** tab.

Figure 27 - VM Advanced Parameters

4. Scroll to locate the vm.genidX parameter and verify that it is populated.

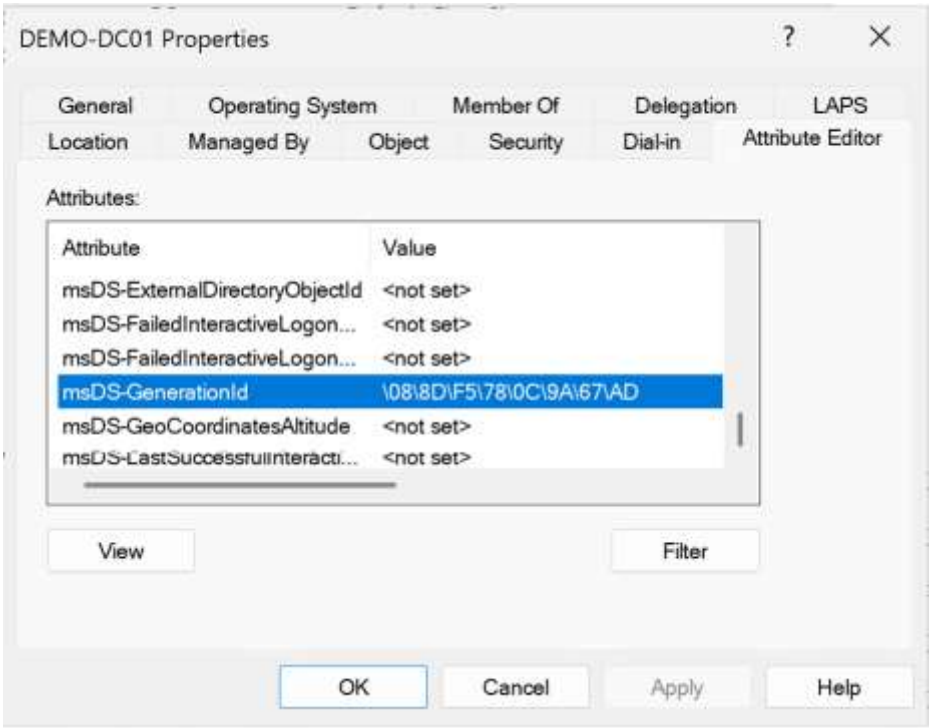
Figure 28 - Confirm "vm.genidX" Present

5. If the source domain controller is powered on, you can check the VM-Generation ID by querying the value of the msDS-GenerationId attribute on the domain controller.

Note: *The msDS-GenerationId attribute is not a replicated value and must be checked locally on the source domain controller.*

1. Log in to the source domain controller and launch Active Directory Users and Computers.
2. From the menu bar select View, Advanced Features. The Attribute Editor tab is an advanced feature not viewable by default.
3. Navigate to the Domain Controllers Organizational Unit (OU) and open the properties for the source domain controller.

Figure 29 - VM GenerationID Attribute in ADUC



4. In the Attribute Editor tab, scroll to find the msDS-GenerationId attribute.

If the attribute appears as <not set>, verify that you have connected Active Directory Users and Computers to the local domain controller (and not a remote domain controller) and that the virtual machine is configured with a guest operating system such as Windows 2012 or later. Both are shown in the following images.

Figure 30 - VM GenerationID Atribute Not Replicated

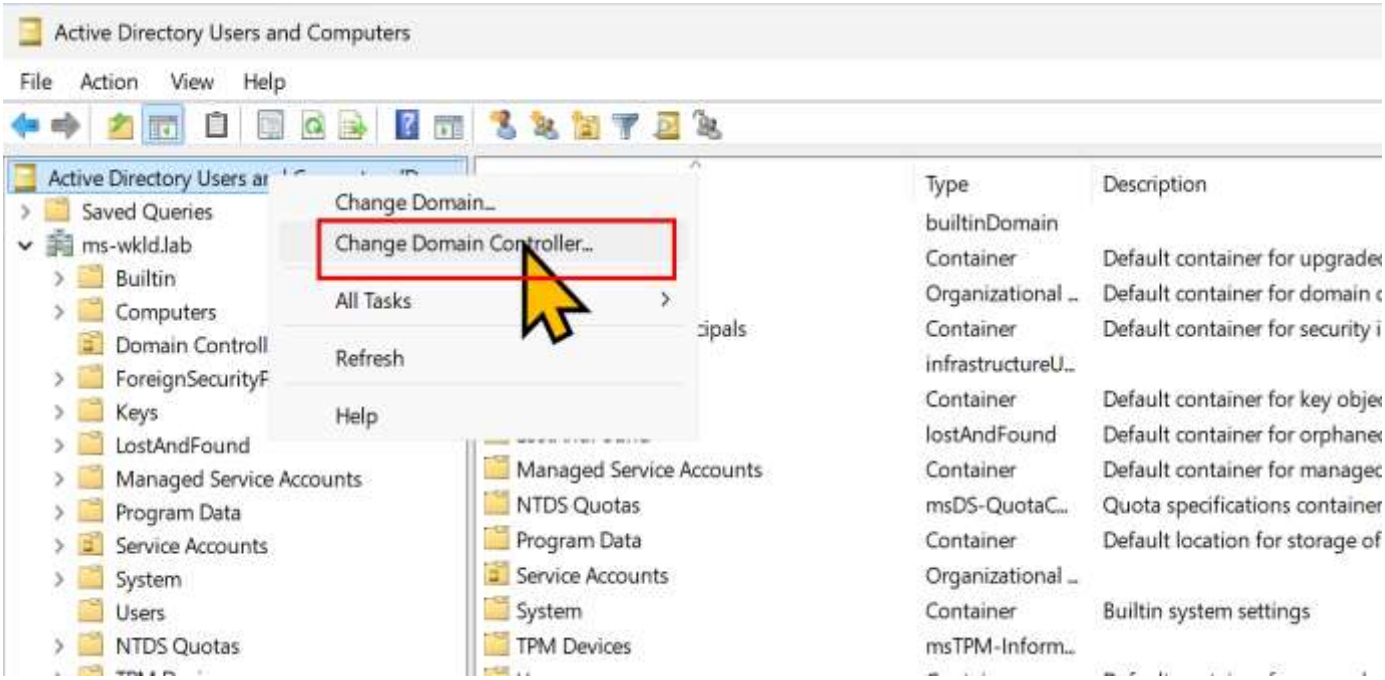
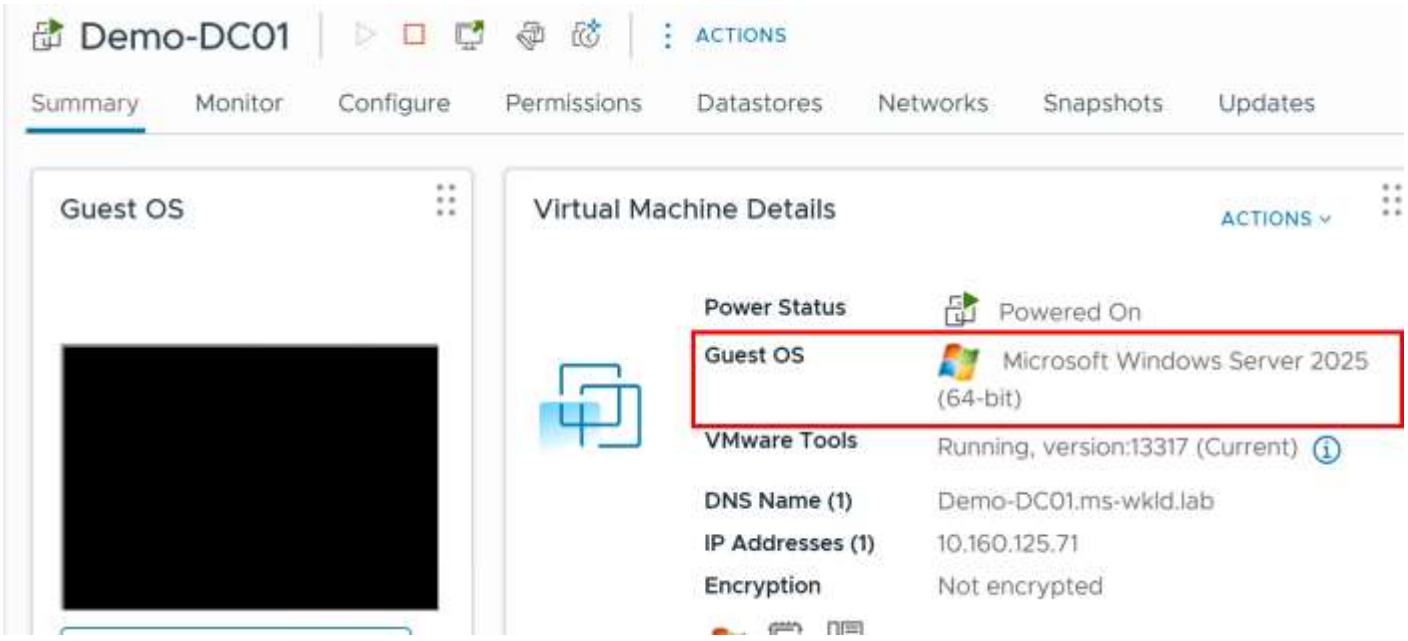


Figure 31 - Confirm VM Running Supported OS Version

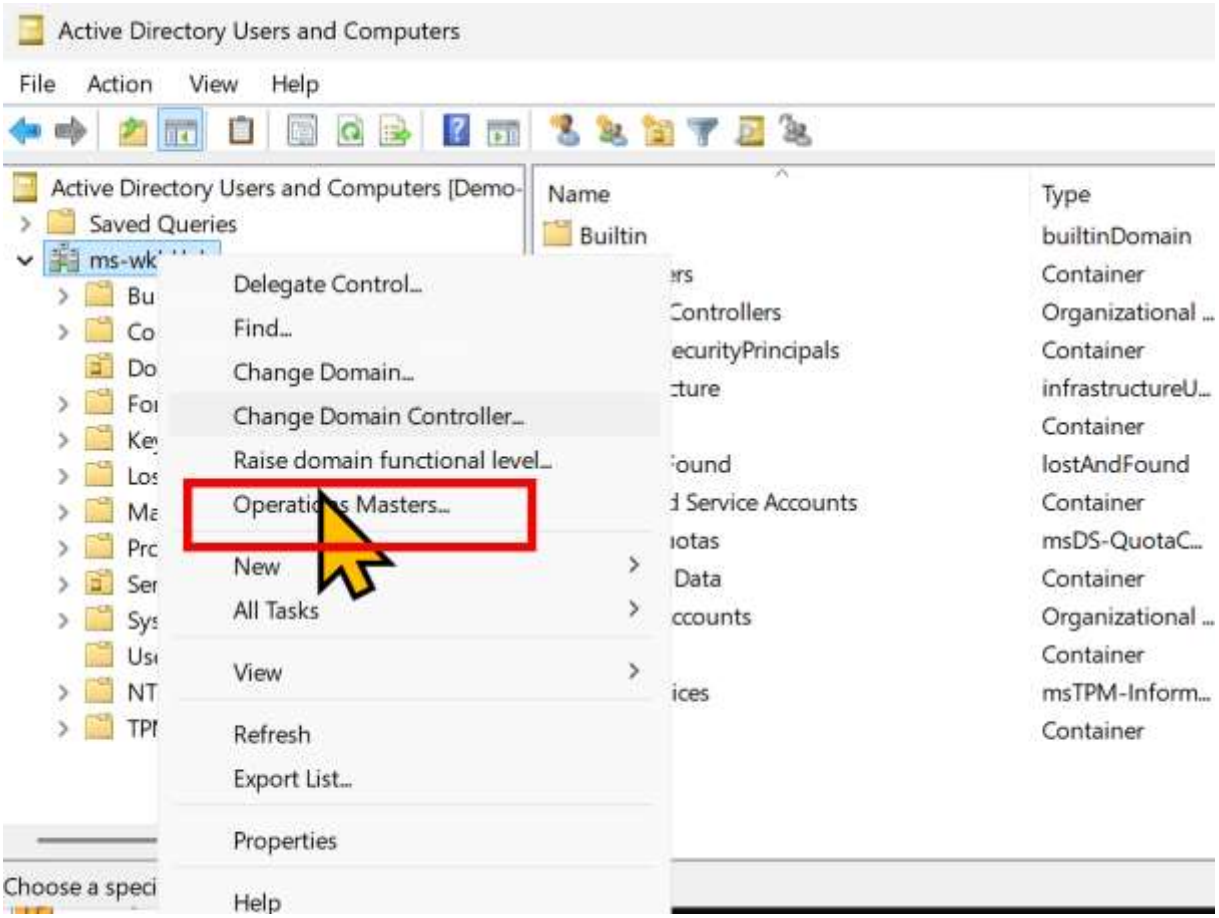


- **Verify that the PDC Emulator is Running on Windows 2012 or Later**

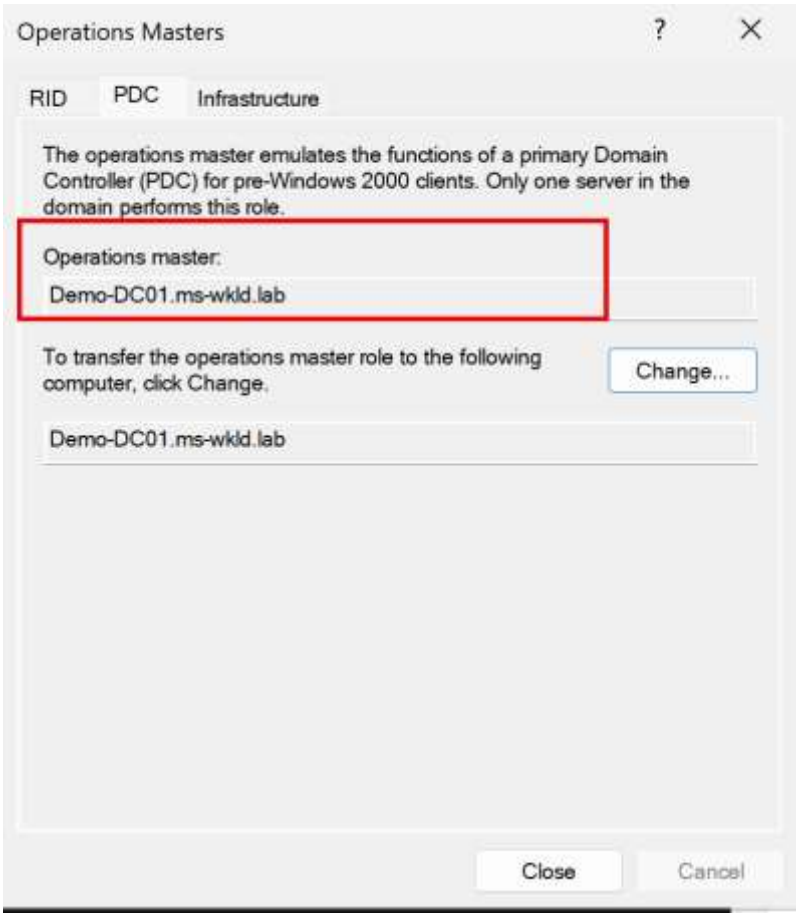
Virtual domain controller cloning requires the operations master role holder for the PDC Emulator role to run on Windows Server 2012 or newer version. To verify, perform the following steps.

1. Open Active Directory Users and Computers, right-click the domain object within the directory tree, and select Operations Masters as shown in the following image.

Figure 32 - Confirm PDCe Online Before Cloning Operation



2. Open the PDC tab and identify the host name of the current PDC Emulator as shown in the following image.

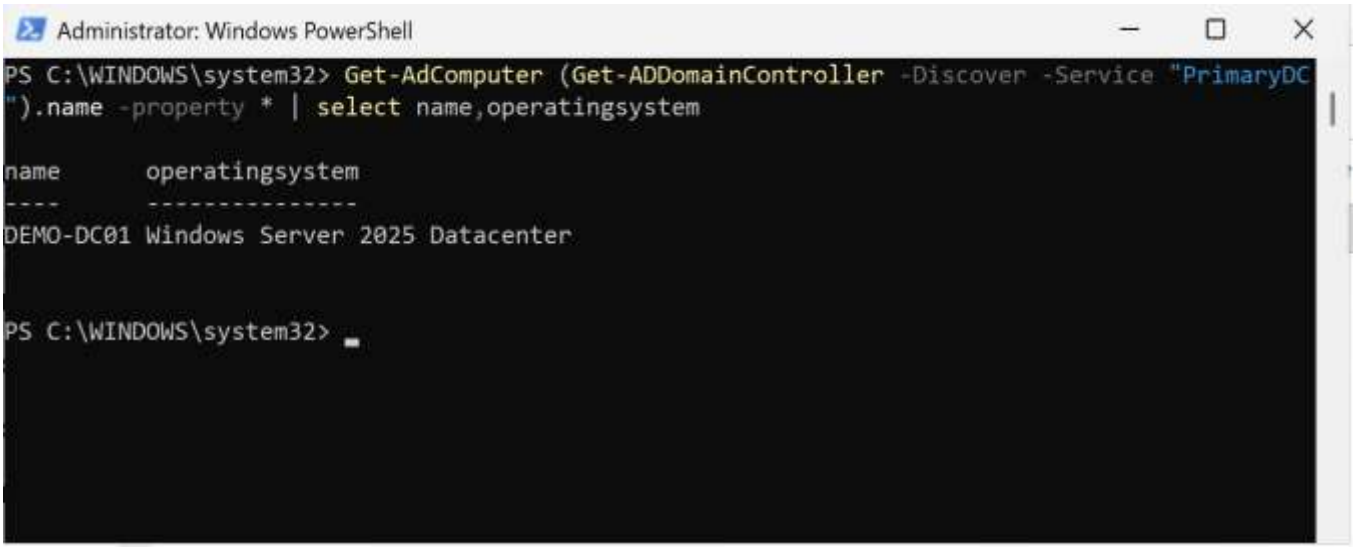


Alternatively, you can use PowerShell to confirm the OS version of the PDC Emulator.

1. Log in to the PDC Emulator.
2. Use the following PowerShell command to find the PDC Emulator of the domain and the Windows version.

- *Get-AdComputer (Get-ADDomainController -Discover -Service "PrimaryDC").name -property * | select name,operatingsystem*

Figure 33 - Confirming PDCe OS Version with PowerShell

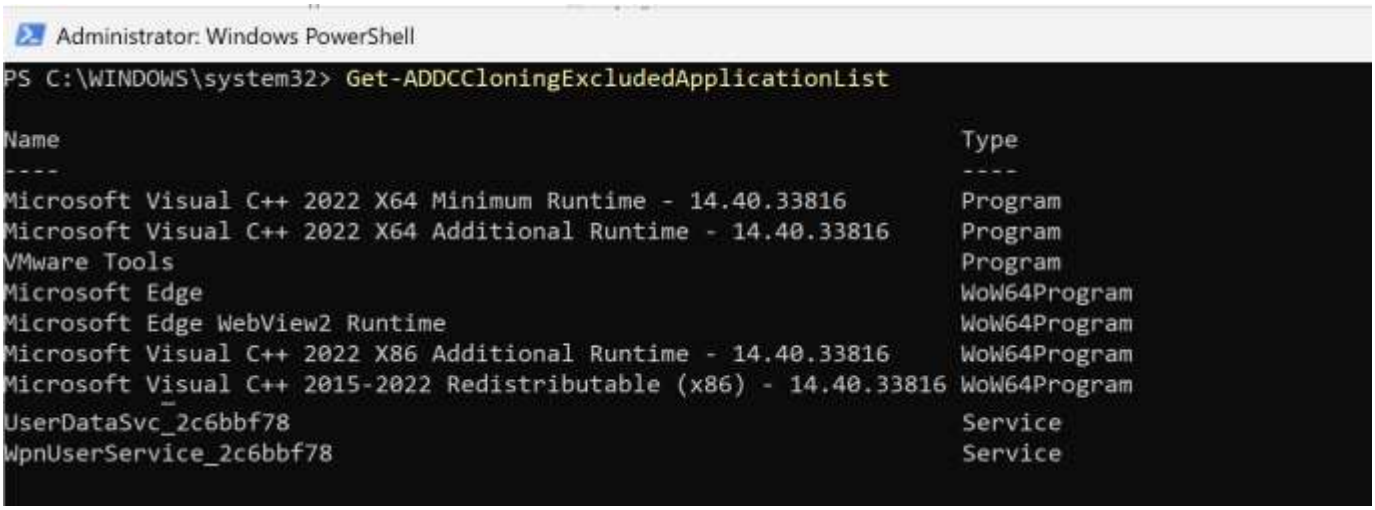


- **Validate the Applications Installed on the Source Domain Controller**

The applications installed on the source must be listed in the default DCclone Allow List (%systemroot%\system32\DefaultDCCloneAllowList.xml) or in the custom allow list to allow cloning to proceed.

1. To identify whether or not any installed applications require removal, or can be cloned, but must be included in a custom allow list, run the following PowerShell command.
   - *Get-ADDCCloningExcludedApplicationList*

Figure 34 - Validating Applications Exception List on Source DC

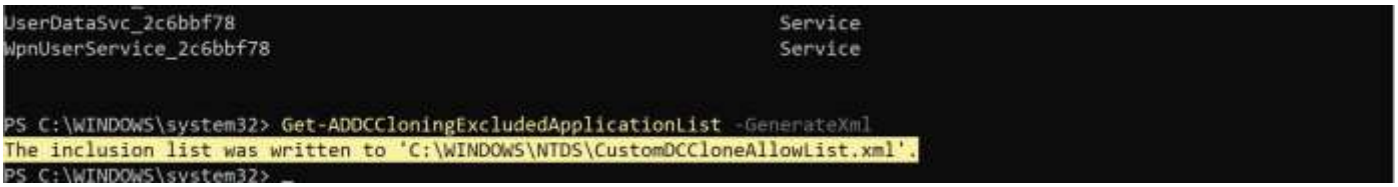


The output displays applications or services that are not in the default clone list. VMware services (VMTools and vmvss) and applications (VMware Tools) can be cloned. Any other applications should be verified with the vendor.

2. Remove any application that is not safe for cloning, such as monitoring or backup agents. Then verify the list again.
3. To add applications to the custom allow list, run the following command.
   - *Get-ADDCCloningExcludedApplicationList –GenerateXml*

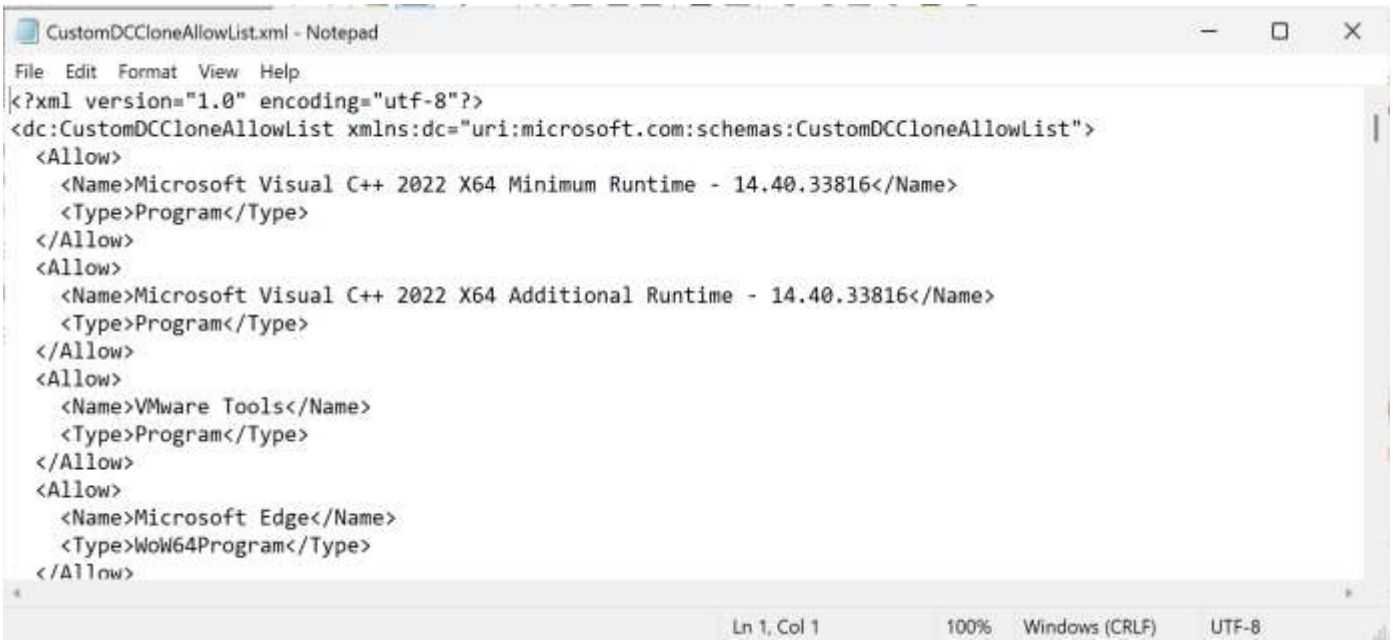
As shown in the following screen, the custom allow list is written to **CustomDCCloneAllowList.xml** within the **NTDS** folder.

Figure 35 - Saving Applications Exception List on Source DC



The custom allow list is shown in the following image.

Figure 36 - Custom Allow XML File

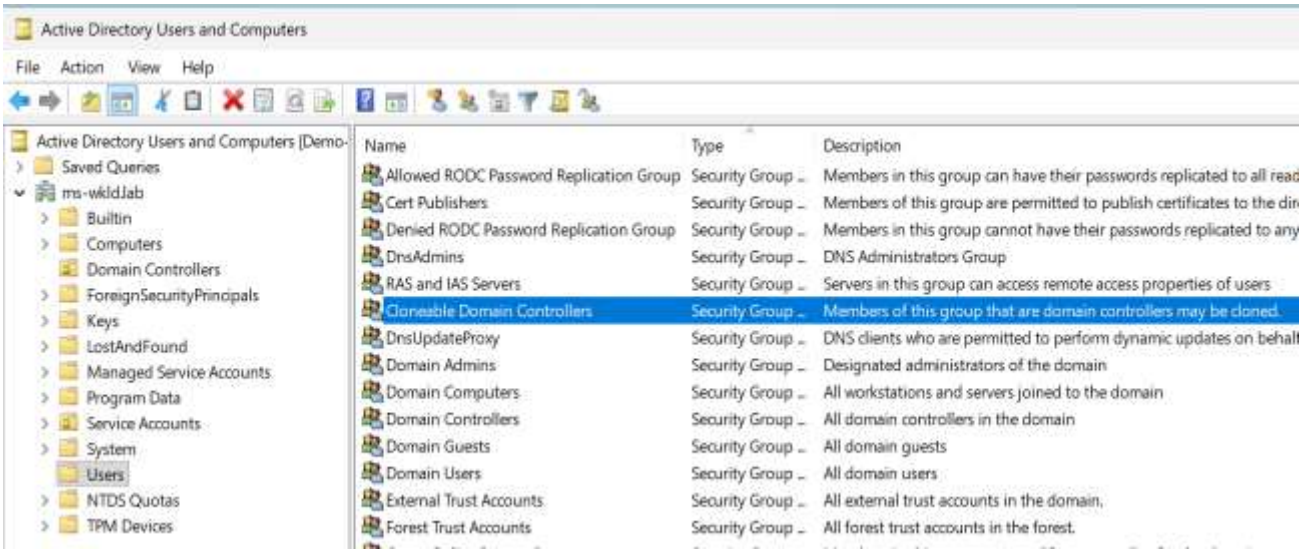


- **Allow Domain Controller Cloning**

  Domain controller cloning is protected by a new right in modern Windows Active Directory. This right is granted to the Active Directory Global Security group, Cloneable Domain Controllers. To allow the source domain controller to be cloned, it must be added to the Cloneable Domain Controllers group.

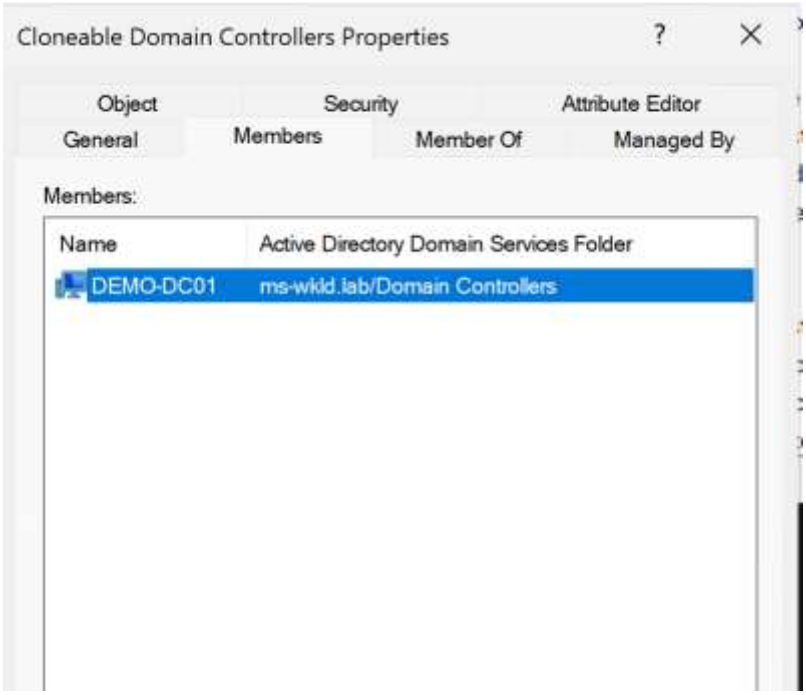
1. Open Active Directory Users and Computers.
2. Navigate to the Users container, and open the Cloneable Domain Controllers group as shown in the image below

Figure 37 - Cloneable Domain Controllers Security Group



3. Add the source domain controller to the group as shown in the image below.

Figure 38 - Add the Source DC to the Cloneable Domain Controllers Security Group



- **Create the Domain Controller Configuration File**

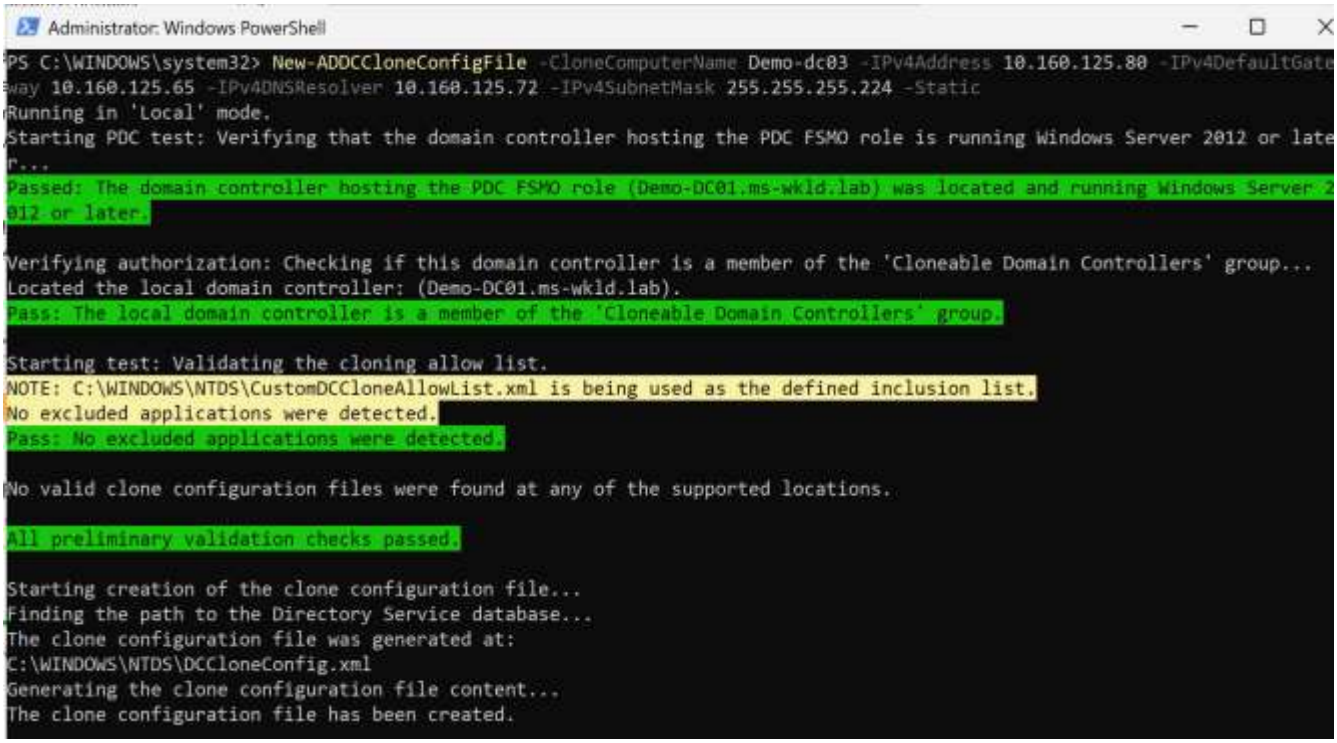
The domain controller configuration file (DCCloneConfig.xml) is an XML file that contains the identity information for the new domain controller. Before shutting down the source domain controller, the configuration file must be created.

To create the Domain Controller Configuration File, use the PowerShell command New-ADDCCloneConfigFile, which has various parameters. The most common parameters are as follows:

- *New-ADDCCloneConfigFile -CloneComputerName <New DC Name> -IPv4Address <IP Address> -IPv4DefaultGateway <Gateway> -IPv4DNSResolver <DNS Server> -IPv4SubnetMask <Subnet Mask> -Static*

The following screen shows the output of this command, including the pre-checks that occur during execution. The pre-checks include verifying that the PDCe runs on a supported version Windows Server, checking whether or not the source domain controller is a member of the Cloneable Domain Controllers group, verifying the clone allow list, and checking for existing clone configuration files.

Figure 39 - Generating DC Clone Configuration File



- **Shut Down the Source Domain Controller**

The last step before you can clone a domain controller is to shut it down as shown in the following screen. Hot-cloning a domain controller for the purpose of creating a domain controller clone is not supported in a production environment.

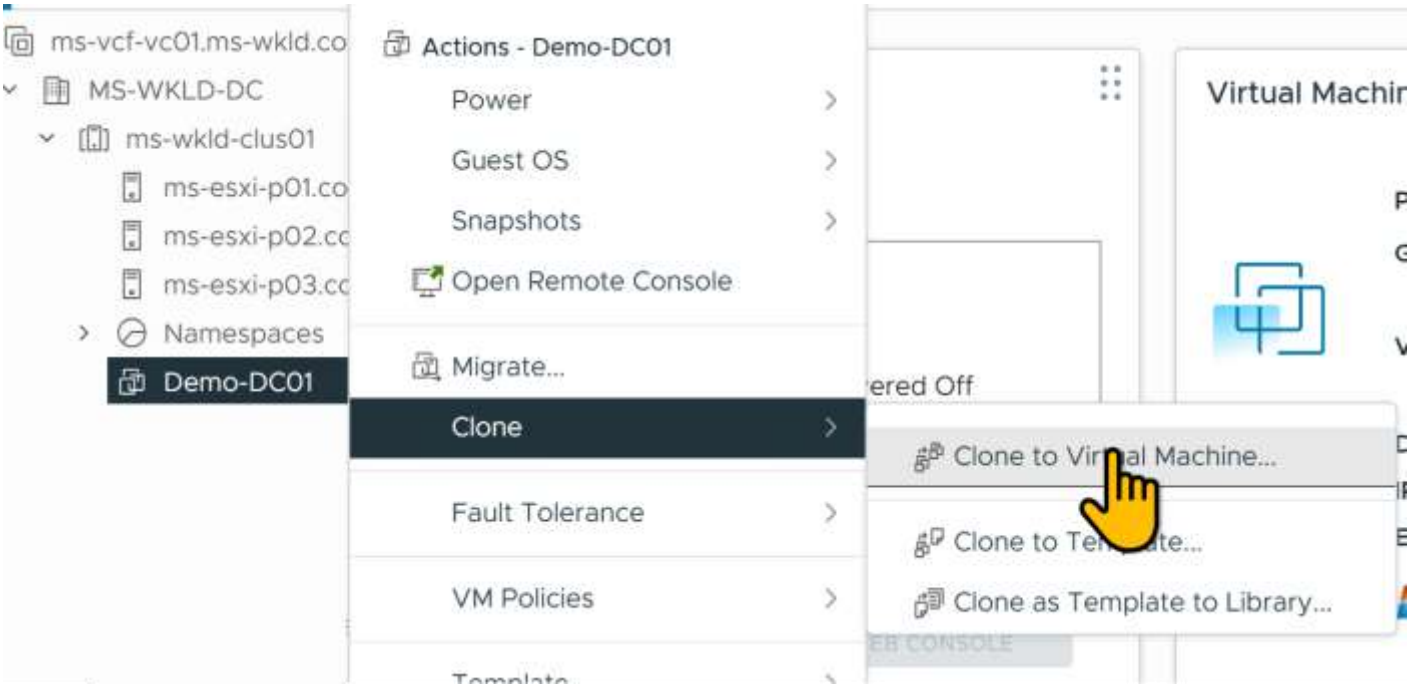
Figure 40 - Shut Down the Domain Controller

- **Clone the Source Domain Controller Virtual Machine**

With the source domain controller virtual machine powered off, the virtual machine can be cloned through any VMware supported process, including standard cloning and copying of a virtual hard disk. In this example, the standard cloning process is used.

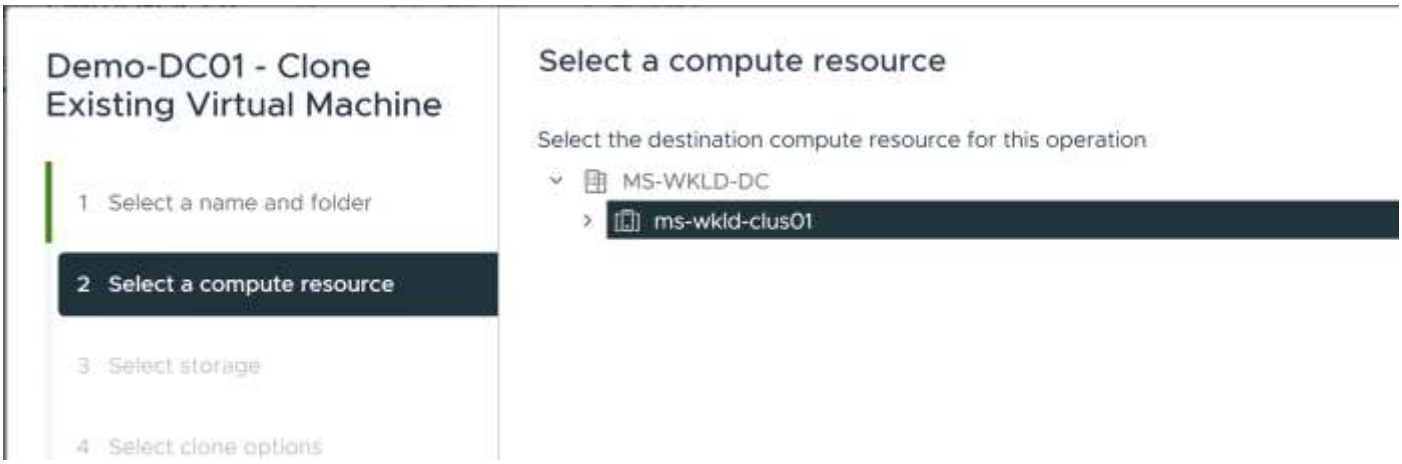
1. Within the vSphere Web Client, locate the source domain controller virtual machine and select Clone to Virtual Machine from the Actions menu.
2. Using the Clone Existing Virtual Machine wizard, provide a name and location for the new virtual machine.

Figure 41 - Clone the Source Domain Controller in vSphere Web Client



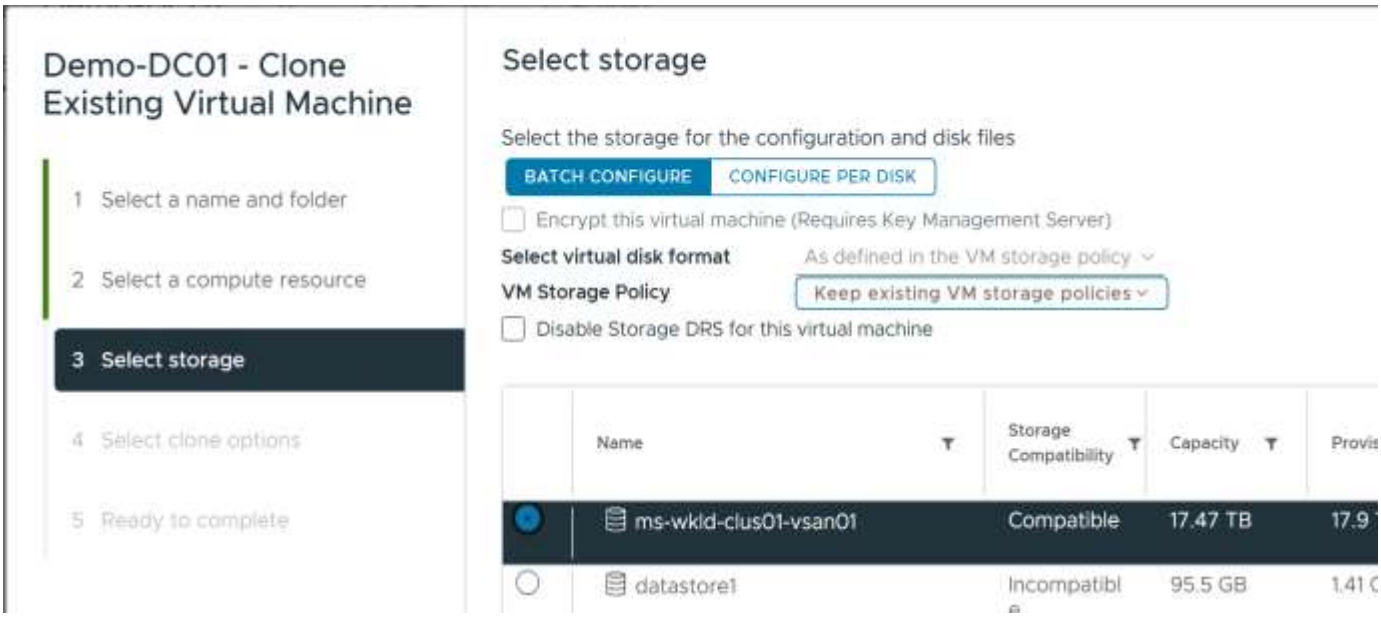
3. Select the compute cluster resource for the new virtual machine.

Figure 42 - Select the Target Cluster



4. Select the storage format and location for the new virtual machine

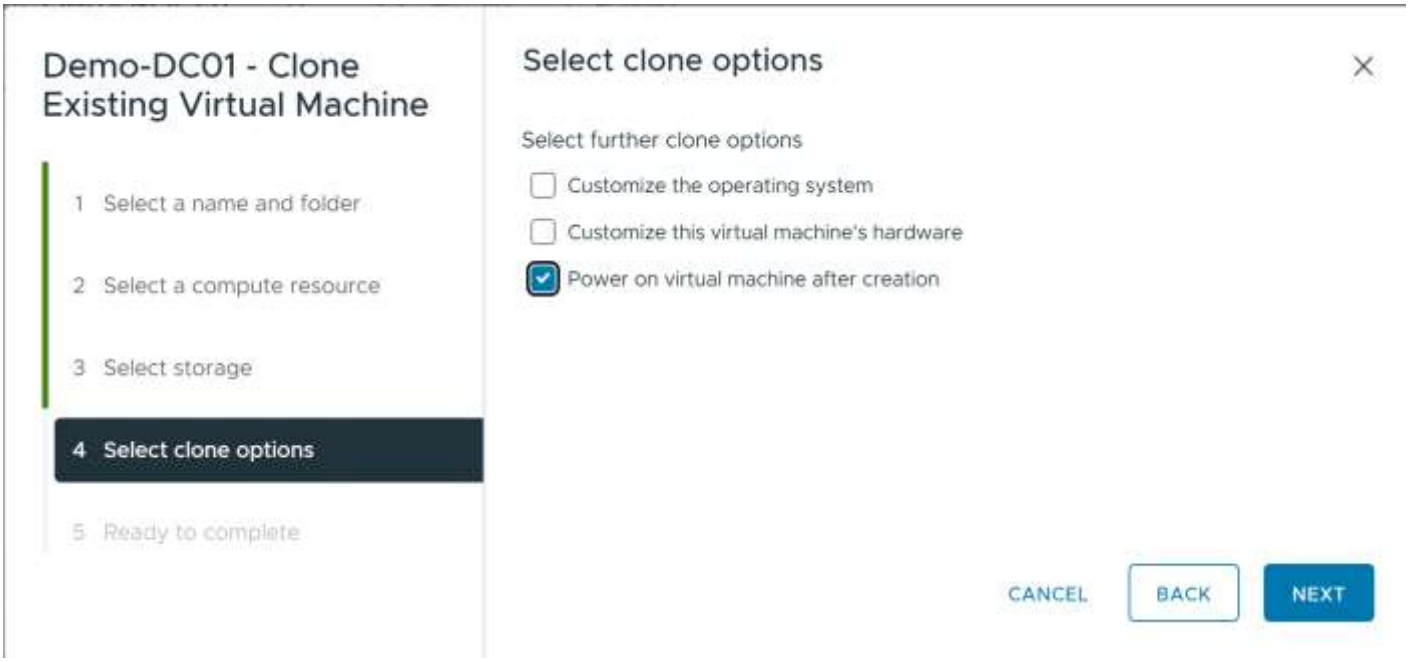
Figure 43 - Select Destination Datastore



5. If the new domain controller is to reside on a new network segment, either leave all check boxes deselected and modify the network settings prior to powering on the virtual machine, or select the check “Customize this virtual machine’s hardware (Experimental) box and adjust the network settings for the virtual NIC. The virtual machine must be able to communicate on the network during the first boot cycle to initiate the cloning process. In this example, the virtual machine is deployed on the same network segment as the source.

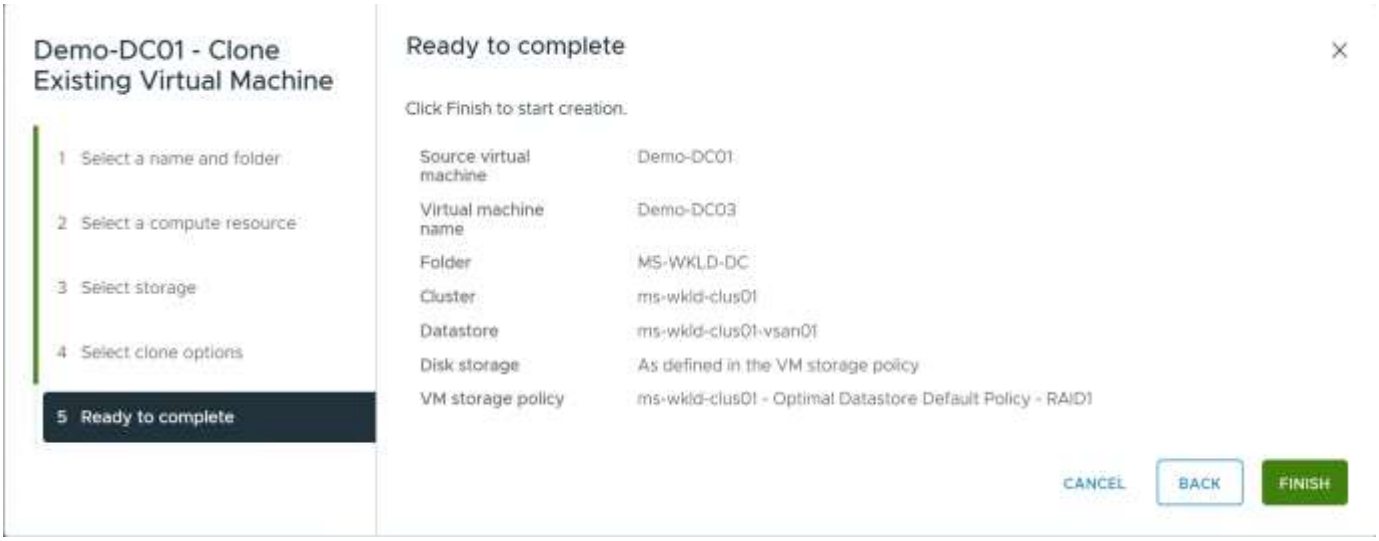
Important: Do not select the Customize the operating system option because that will interfere with the safe cloning operation.

Figure 44 - Carefully Select the Appropriate Options



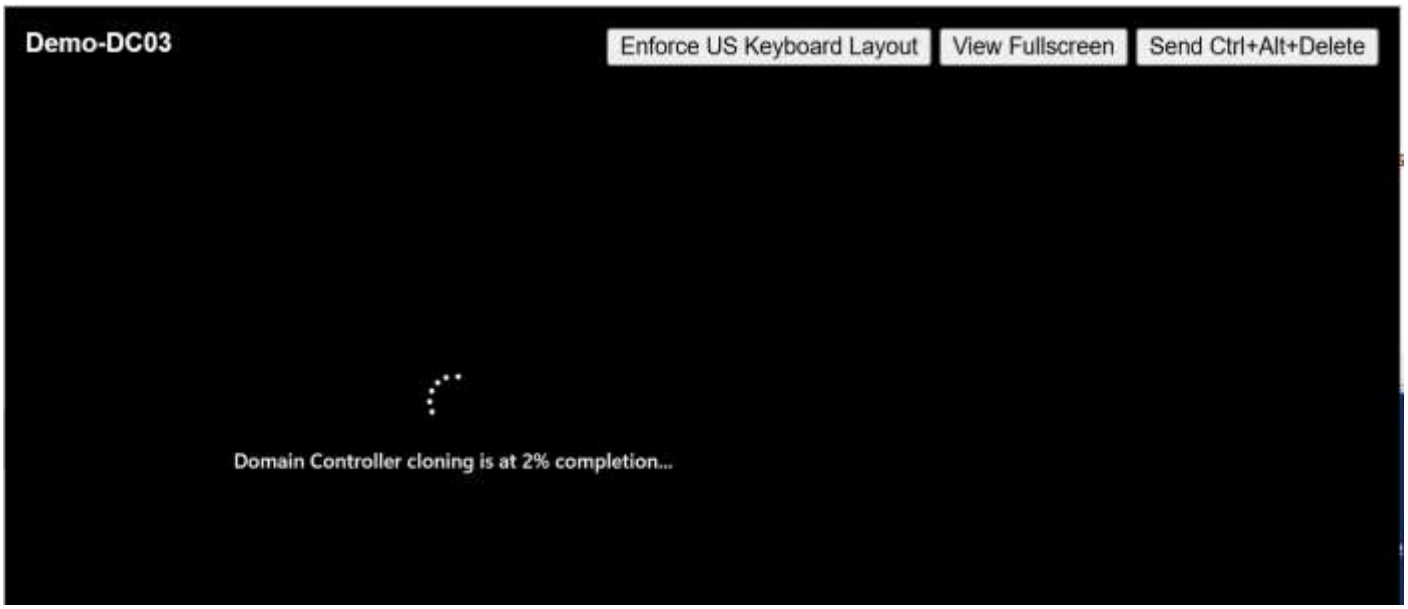
6. Confirm the settings and proceed with the cloning.

Figure 45 - Complete the Cloning Configuration



7. When the cloning process is completed and the virtual machine boots for the first time, the domain controller virtualization safeguards initiate the cloning process.

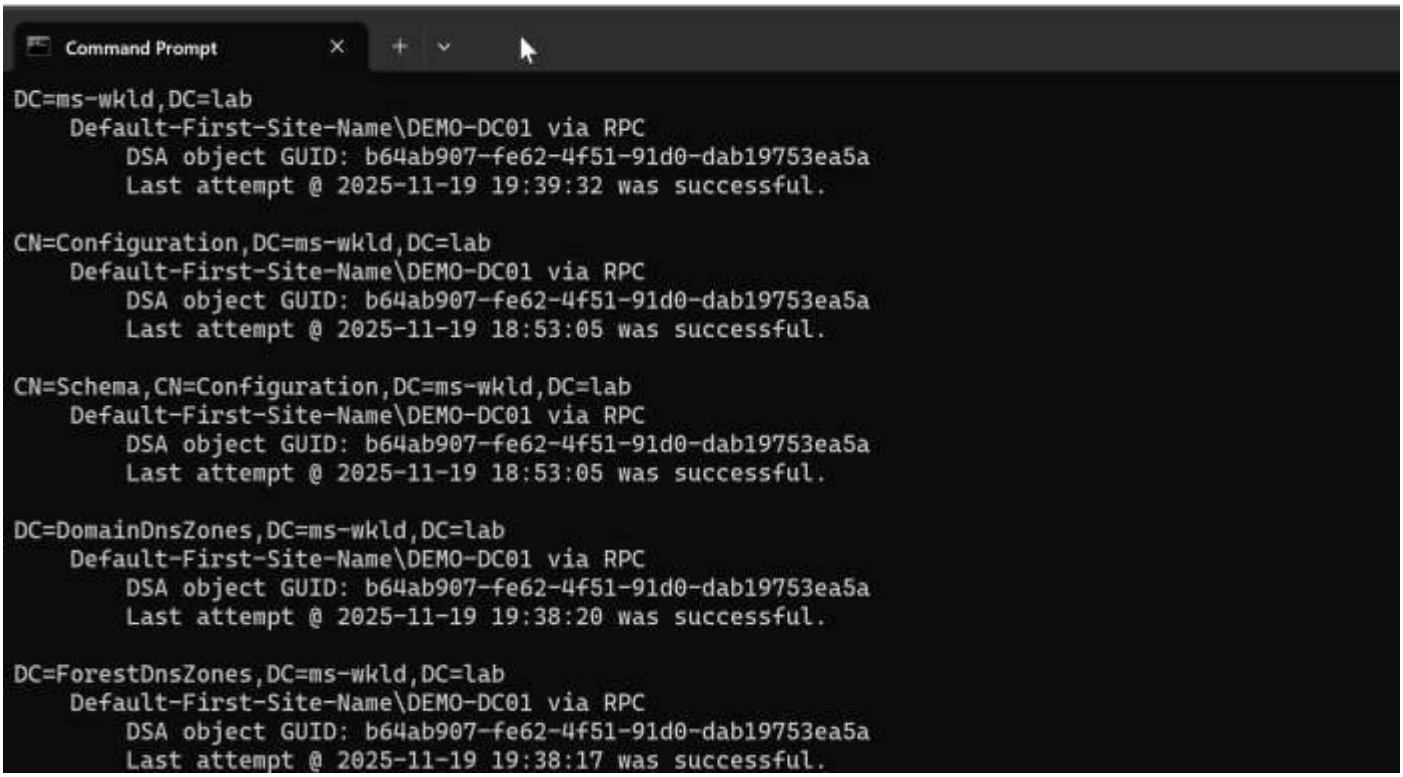
Figure 46 - Clone Virtual Machine Booting Up



8. Verify the domain controller identity and replication status.

In the following figure, the domain controller name and invocation ID are unique, and replication is completing successfully with Demo-DC01.

Figure 47 - Confirm New DC Functioning and Replicating



Additionally, the directory service event log tracks the events during domain controller safeguard as described in the following table.

Table 3 - Domain Controller Safeguard Events

Event ID	Details
2170	A Generation ID change has been detected.
2181	The transaction was aborted due to the virtual machine reverting to a previous state. This result occurs after the application of a virtual machine snapshot, after a virtual machine import operation, or after a live migration operation.
2204	Active Directory Domain Services has detected a change of VM-Generation ID. The change means that the virtual domain controller has been reverted to a previous state. Active Directory Domain Services will perform the following operations to protect the reverted domain controller against data divergence and to protect creation of security principals with duplicate SIDs: create a new invocation ID and invalidate current RID pool.
1109	The invocation ID attribute for this directory server has been changed.

Item No: vmw-bc-wp-tech-temp-a4-word-2024 1/24