## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Executive Summary</strong></td>
<td>1</td>
</tr>
<tr>
<td>Consolidated list of practices</td>
<td>1</td>
</tr>
<tr>
<td><strong>Citrix PVS Provisioning Services Overview</strong></td>
<td>4</td>
</tr>
<tr>
<td>What is PVS and how does it work?</td>
<td>4</td>
</tr>
<tr>
<td>vDisk Configuration</td>
<td>5</td>
</tr>
<tr>
<td>VMs and PVS Server Configuration</td>
<td>7</td>
</tr>
<tr>
<td>PVS Variation #1: Typical Write Cache Disk</td>
<td>8</td>
</tr>
<tr>
<td>PVS Variation #2: Non-persistent Write Cache Disk</td>
<td>9</td>
</tr>
<tr>
<td><strong>The Good, the Bad &amp; the Ugly</strong></td>
<td>11</td>
</tr>
<tr>
<td>Pros</td>
<td>11</td>
</tr>
<tr>
<td>Cons.</td>
<td>12</td>
</tr>
<tr>
<td>Gotchas</td>
<td>13</td>
</tr>
<tr>
<td><strong>Big Bang Theory – The creation process</strong></td>
<td>15</td>
</tr>
<tr>
<td>Creating a vDisk</td>
<td>15</td>
</tr>
<tr>
<td>Creating PVS Targets</td>
<td>16</td>
</tr>
<tr>
<td>Creating a Machine Catalog based on PVS.</td>
<td>21</td>
</tr>
<tr>
<td><strong>Das Bootin – The PVS Boot-up process</strong></td>
<td>23</td>
</tr>
<tr>
<td><strong>Consideration Factors</strong></td>
<td>26</td>
</tr>
<tr>
<td>Space Utilization &amp; Capacity Considerations</td>
<td>26</td>
</tr>
<tr>
<td>Visibility</td>
<td>26</td>
</tr>
<tr>
<td>Persistence</td>
<td>26</td>
</tr>
<tr>
<td>Additional Factors</td>
<td>27</td>
</tr>
<tr>
<td><strong>Management &amp; Housekeeping</strong></td>
<td>28</td>
</tr>
<tr>
<td>Adding additional VMs to a PVS-based Machine Catalog</td>
<td>28</td>
</tr>
<tr>
<td>Image Versions</td>
<td>29</td>
</tr>
<tr>
<td>Special Target Types unique to PVS</td>
<td>29</td>
</tr>
<tr>
<td>Maintenance Targets</td>
<td>30</td>
</tr>
<tr>
<td>Test Targets</td>
<td>32</td>
</tr>
</tbody>
</table>
The Image Update Process ..............................................34
Promoting new images to production ................................35
The Roll Back process .....................................................35
Cleaning Up Old Versions ..............................................36
Versioning Challenge: Single-Threaded Dependencies ........37

Data Protection & Disaster Recovery (DR) Overview 39

Backup & DR of PVS-based Machine Catalogs ................39
PVS Infrastructure Servers - Backup ..............................39
Backup - Target VMs ....................................................39
DR - Target VMs & PVS Servers ....................................40

Conclusion 41

Additional Resources – Reference URLs 42

Tintri Links .................................................................42
Citrix Links ...............................................................42
Other .................................................................42

Appendix A – Environment Details 43

Appendix B – Why dedicate a VMDK per Store and per vDisk? 44

Appendix C – Storage space consumption 45
This paper is one of several papers in a series that explores machine creation & provisioning technologies in Citrix XenDesktop 7.1 deployed on Tintri VM-aware storage. This paper primarily focuses on Citrix Provisioning Services (PVS) and complements XenDesktop 7.1 Provisioning & Machine Creation Deep Dive, which covers all three provisioning methods and has a lot of useful information common to all.

Citrix Provisioning Services (PVS) has some excellent features the other provisioning methods don’t have, one of which is master image versioning. Versioning provides a good workflow that adapts well to most change control procedures mandated in many IT organizations. For this reason alone, many organizations have chosen PVS as their provisioning & machine creation method of choice. In this paper, we’ll take an in-depth look at versioning, and many more topics that are unique to PVS.

The target audience for this paper includes IT admins, architects, and other technical roles that already have some experience with XenDesktop, vSphere and storage. If you do not have any experience with XenDesktop yet, we strongly suggest you follow our XenDesktop 7.1 Quick Start Guide to get a functional XenDesktop environment in place, and then continue reading this paper.

There are three methods available to provision XenDesktop virtual desktops:

- Citrix Provisioning Services (PVS) – focus of this paper.
- Machine Creation Services (MCS).
- Tintri Native space and performance efficient VAAI Clones.

All three methods are all fully supported and work very well on Tintri VMstore systems. Deciding which method(s) to use for your organization depends on many factors, and the goal of this series of XenDesktop Deep Dive papers is to give you more insight into PVS and the other methods to help you make a well-informed decision for your enterprise.

Without further ado, let’s dive in!

Consolidated list of practices

The table below includes the recommended practices in this document. Click the text in the Recommendation column or see the section later in the document that corresponds to each recommendation for additional information.

<table>
<thead>
<tr>
<th>Tags</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is PVS and how does it work?</td>
<td>Do: Use the same dedicated SQL Server VM the other XenDesktop components are configured to use to simplify administration.</td>
</tr>
<tr>
<td>What is PVS and how does it work?</td>
<td>Do: In addition to the database role, two or more streaming servers should be used to provide fault-tolerance and higher availability.</td>
</tr>
<tr>
<td>What is PVS and how does it work?</td>
<td>Do: When using multiple PVS Streaming Servers (recommended), create DRS rules to keep VMs separate across hosts. Adding this prevents the VMs from running on the same host, which would be a single point of failure if that host failed.</td>
</tr>
<tr>
<td>vDisk Configuration</td>
<td>Do: Use of Versions functionality is highly recommended.</td>
</tr>
<tr>
<td>Tags</td>
<td>Recommendation</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VMs and PVS Server Configuration</td>
<td><strong>Do</strong>: Choose “Cache on device hard drive” for a Tintri VMstore.</td>
</tr>
<tr>
<td>VMs and PVS Server Configuration</td>
<td><strong>Do</strong>: When configuring PVS, create and configure one or more PVS store(s) on your server to hold your vDisks.</td>
</tr>
<tr>
<td>VMs and PVS Server Configuration</td>
<td><strong>Do</strong>: Use a dedicated virtual disk per Store mounted on your PVS Streaming Server (VM configuration in vCenter).</td>
</tr>
<tr>
<td>VMs and PVS Server Configuration</td>
<td><strong>Do</strong>: In addition to using a dedicated vmdk per Store, use only one vDisk (PVS object) per Store.</td>
</tr>
<tr>
<td>Gotchas</td>
<td><strong>Do</strong>: Make sure the drive is formatted, assigned a drive letter, and the System account (by default) has the necessary permissions to create files and folder in the root of the drive.</td>
</tr>
<tr>
<td>Gotchas</td>
<td><strong>Do</strong>: Usage of unique accounts for your PVS services is highly recommended.</td>
</tr>
<tr>
<td>Creating a vDisk</td>
<td><strong>Do</strong>: Make a clone using the Tintri UI or vSphere client before running PVS imaging tools within your VM. This will prevent undesired changes to your Master Image VM that would make it incompatible with MCS or some other provisioning method.</td>
</tr>
<tr>
<td>Creating a vDisk</td>
<td><strong>Do</strong>: When you’re creating a vDisk, change it from the default “Fixed” (aka Thick-provisioned) to “Dynamic” (Thin-provisioned).</td>
</tr>
<tr>
<td>Creating a vDisk</td>
<td><strong>Do</strong>: Set the vDisk block size to 16MB to match the size at which the underlying vmdk file.</td>
</tr>
<tr>
<td>Creating a vDisk</td>
<td><strong>Do</strong>: Your underlying store should be placed on a vmdk that is also thin, for maximum capacity savings.</td>
</tr>
<tr>
<td>Creating a vDisk</td>
<td><strong>Do</strong>: If your master image doesn’t have at LEAST 40 GB allocated to it, take advantage of the opportunity to grow the disk at time of imaging to increase the size (shown here).</td>
</tr>
<tr>
<td>Das Bootin - The PVS Boot-up process</td>
<td><strong>Do</strong>: Dedicate one vDisk per PVS Store per vmdk to make any IO associated with a vDisk clearly identifiable in the VMstore UI.</td>
</tr>
<tr>
<td>Persistence</td>
<td><strong>Do</strong>: If persistence is important to your organization, consider using Tintri Native clones or do some additional research and consider implementing Citrix Personal vDisks. Citrix Personal vDisks can help you, but are out of scope of this paper.</td>
</tr>
<tr>
<td>Audit &amp; Compliance</td>
<td><strong>Do</strong>: Check the logs for yourself to ensure that actions you expect to be recorded are in fact recorded, and are sufficient evidence to meet the audit and compliance requirements of your environment.</td>
</tr>
<tr>
<td>Audit &amp; Compliance</td>
<td><strong>Do</strong>: If the Event logs or other monitoring software within the guest require retention, consider a centralized logging solution to record and maintain yours logs outside of your desktops VMs.</td>
</tr>
<tr>
<td>Adding additional VMs to a PVS-based Machine Catalog</td>
<td><strong>Do</strong>: Clone the same Tintri snapshot of the Template VM used to create the first batch to make new clones for importing into a device collection as targets.</td>
</tr>
<tr>
<td>Image Versions</td>
<td><strong>Do</strong>: In the vDisk Versions window, use the Properties button to create a description of each version to track what is being changed, as shown in the lower portion of the screenshot above.</td>
</tr>
<tr>
<td>Special Target Types unique to PVS</td>
<td><strong>Do</strong>: When creating the Maintenance and Test VMs, name the VMs so they are clearly identifiable as their intended use and place them in their own VM folder, separate from the others. Example names: PVS-MAINT, PVS-Test1, PVS-Test2, etc.</td>
</tr>
<tr>
<td>Maintenance Targets</td>
<td><strong>Do</strong>: Make it a habit to take the extra 30 seconds to check Read/Write status prior to commencing work on changes.</td>
</tr>
<tr>
<td>Tags</td>
<td>Recommendation</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Test Targets</strong></td>
<td>Do: When possible, test proposed master image version changes in a test target BEFORE making the changes in your maintenance version.</td>
</tr>
<tr>
<td><strong>Test Targets</strong></td>
<td>Do: Create and power-on as many Test VMs as required to accommodate the number of test users you’ve identified within your organization.</td>
</tr>
<tr>
<td><strong>Promoting new images to productions</strong></td>
<td>Do: We recommend you use rolling reboots over the course of 2 weeks, automated via PowerShell (or other) scripting.</td>
</tr>
<tr>
<td><strong>Versioning Challenge: Single-Threaded Dependencies</strong></td>
<td>Do: If you followed the recommendation of creating one vDisk per VMDK on your PVS servers, consider deleting the vmdk and provision a new, thin one in its place to reclaim the dirtied blocks from the overall capacity available in your VMstore.</td>
</tr>
<tr>
<td><strong>PVS Infrastructure Servers - Backup</strong></td>
<td>Do: Follow best practices when it comes to SQL database backups. In particular, ensure your backup process truncates your SQL transaction logs to prevent them from growing and eventually consuming all of your disk space.</td>
</tr>
<tr>
<td><strong>DR - Target VMs &amp; PVS Servers</strong></td>
<td>Do: For DR, consider using ReplicateVM™ to make a one-time replica of your Template VM.</td>
</tr>
<tr>
<td><strong>DR - Target VMs &amp; PVS Servers</strong></td>
<td>Do: Protect your PVS servers with ReplicateVM and create a replication schedule to replicate them over to your DR site periodically.</td>
</tr>
<tr>
<td><strong>DR - Target VMs &amp; PVS Servers</strong></td>
<td>Do: With the exception of vDisk version updates, change on PVS server should be minimal, so consider a daily or even a weekly snapshot and replication schedule for DR.</td>
</tr>
<tr>
<td><strong>DR - Target VMs &amp; PVS Servers</strong></td>
<td>Do: Ensure that Machine Catalogs &amp; Delivery Groups are configured for the DR collections.</td>
</tr>
<tr>
<td><strong>Appendix C - Storage space consumption</strong></td>
<td>Do: With thin provisioned disks, the recommendation is to avoid using a cache disk that is the same size as your vDisk to avoid confusion of where your data actually comes from.</td>
</tr>
</tbody>
</table>
Citrix PVS Provisioning Services Overview

Citrix PVS is an alternative method for creating and deploying large numbers of clones. PVS works with XenDesktop, as well as XenApp. Although this paper focuses on PVS for virtual desktop VMs (one user session per VM), the same procedures can be applied to XenApp shared desktops or published app servers (many user sessions per VM).

Refer to the official Citrix PVS Documentation for complete details about PVS.

What is PVS and how does it work?

In a nutshell, “target” systems (Virtual Machines that don’t have an OS disk) are configured to boot from the network (PXE). A PVS streaming server(s) provides a PXE bootstrap which instructs the machines to request a vDisk from one of the PVS streaming servers to boot off. The PVS Servers hold a record of all preconfigured “targets”, identifiable by MAC address. When the requesting MAC address is found in the backend database of all targets, a vDisk is streamed to the target, along with unique personalization (machine name, SID, domain computer account password, etc.) injected in. The streamed OS disk (C: drive) is appears as a Citrix Virtual Disk from within the guest.

Unlike traditional virtual disk files (.vmdk), the Citrix virtual disks exist only in memory in the target and within cache files when contents need to be altered and written back to disk.

PVS has additional Infrastructure requirements and additional components need to be installed.
than those included with XenDesktop. Similar to the XenDesktop infrastructure server VMs, PVS server VMs can also sit in the same Tintri VMstore as all of the virtual desktops VMs. At a minimum, one server VM can be used for PVS, which will hold the roles of streaming server and database.

- **Do:** Use the same dedicated SQL Server VM the other XenDesktop components are configured to use to simplify administration.
- **Do:** In addition to the database role, two or more streaming servers should be used to provide fault-tolerance and higher availability.
- **Do:** When using multiple PVS Streaming Servers (recommended), create DRS rules to keep VMs separate across hosts. Adding this prevents the VMs from running on the same host, which would be a single point of failure if that host failed.

**vDisk Configuration**

Once a vDisk is created access mode needs to be changed to “Standard Image” in order to enable Versions (highly recommended!) and to configure Cache type and options.

Options are accessible by right-clicking on your vDisk and choosing properties, but ONLY when there are no locks on the vDisk (i.e. no running VMs configured to use it). If all dependent VMs are powered off and you still are not able to view the properties, right-click and choose Manage Locks... to manually clear the locks associated with the vDisk.
Once properly configured, you should see the Versions option available when you right-click on your vDisk in the PVS Console. If you don’t see Versions, your vDisk is likely still in “Private image” mode, which is the access mode when a vDisk is first created.

✅ Do: Use of Versions functionality is highly recommended.

The vDisk that is streamed to targets is a .VHD (virtual hard disk) file, or a combination of a base VHD file with version update files (.AVHD) that are stacked on top, similar to the way snapshots are applied to a base disk.

In the figure displayed, we can see there are four (4) versions shown, which can be derived by the naming format: `vDiskName.Ver#.EXT`. The extensions are .vhd for a base disk, and .avhd for an update. Versions comprised of updates require the complete chain of updates AND base disk. Updates can be merged to make a single merged update, which still requires a base disk, or
rolled up into a Merged Base disk, which is a fully function vDisk without having to keep previous version updates. We’ll cover this in more detail in the section below on Updates and versioning, as well as in the Space Utilization considerations section.

**WARNING:** If you have a folder called “Write-cache” in the same folder as your PVS vDisk files reside, you may not have properly configured your cache option to “Cache on Device hard disk”, or you have a problem with your cache disk within your VM. Make sure the disk assigned to your VM is formatted, is assigned a disk drive letter, and the system account has write access to the root of the drive. When configured correctly, make sure your view options in Windows show hidden and system files, and then be able to see a file called “.vdiskcache” in the root of the second drive (show below).

### VMs and PVS Server Configuration

Now that we have an overview of what PVS is, how it works and how your vDisks should be configured within the PVS Streaming servers, we can focus on what happens in the target VM. Different configurations of the target VM can impact both performance and capacity usage. In the two subsections below, we’ll explore the typical configuration, as well as a light modification.

There are six (6) caching options to choose from for the vDisks:

- **Cache on device hard drive (recommended)**
- Cache on device hard drive persisted (NT 6.1 and later)
- Cache in device RAM
- Cache in device RAM with overflow on hard disk
- Cache on Server
- Cache on Server persisted

To help you decipher the options, “Device” refers to your target VM and “Server” refers to the PVS Streaming Server(s) that your target boots from.

- **Do:** Choose “Cache on device hard drive” for a Tintri VMstore.

The differences between the two PVS methods occur in the configuration of VMs within vCenter. But, the two different variations of PVS are configured with the same PVS options.

- **Do:** When configuring PVS, create and configure one or more PVS store(s) on your server to hold your vDisks.
- **Do:** Use a dedicated virtual disk per Store mounted on your PVS Streaming Server (VM configuration in vCenter).
- **Do:** In addition to using a dedicated vmdk per Store, use only one vDisk (PVS object) per Store.

See “Appendix B – Why dedicate a VMDK per Store and per vDisk?” on page 44 for details on dedicating a vmdk per Store and per vDisk.

VM that will be used as a PVS target needs to have the boot options in its BIOS configured to boot from the network (PXE-boot).
The simplest way to configure this is to force the VM to boot to BIOS on the next boot, as displayed below, and then change the boot order in the BIOS to boot from the Network first.

When the VM boots up, it receives a DHCP address and PXE bootstrap to connect to your PVS server. Based on the MAC address of the VM, the PVS Streaming server determines which vDisk to stream to it, which appears at the C: drive from the guest OS perspective. See “Appendix C – Storage space consumption” on page 45 for details on storage space consumption.

There are two PVS methods:

- PVS Variation #1: Typical Write Cache Disk
- PVS Variation #2: Non-persistent Write Cache Disk

**PVS Variation #1: Typical Write Cache Disk**

Typically, the target VM is assigned a standard virtual hard drive that will be used as the write-cache, as shown in this screenshot.

Characteristics of this configuration are listed below. Items marked with [*] in front vary for the 2nd variation of implementing PVS with write-cache on a non-persistent device hard drive:

- [*] The .vmdk file lives in the same home folder as the VM and written to and read directly.
- Data is written to the logical C: drive (streamed Citrix virtual disk) doesn’t actually write to
it and is redirected into the write-cache file that sits in the virtual disk we configure on the VM itself.

• The streaming server acts read-only in this caching configuration.
• Additional personalization parameters can be passed from PVS Streaming server to each target. Scripts and other programs within the guest can use this info to retain an identity. An example of this would be the GUID for an antivirus vendor that has centralized logging, such as McAfee ePO, or the UID for WSUS (Windows Update Services).
• All changes to Guest’s C: drive are lost on reboot (event logs, registry, files, etc.).
• [*] Data on the write-cache drive persists beyond guest OS reboots and VM power cycles. This is useful if you require a persistent set of files, such as an EdgeSight database that stores persistent data identifiable that specific VM. This is also useful if you need to log process activity to this drive to troubleshoot unexpected reboots of guest OS crashes. All other traces, such as the Windows event logs would otherwise be lost on reboot.
• [*] The blocks within the disk become dirtied, however .vdiskcache files get deleted and the same dirtied blocks get reused inside the Guest OS. Each VM will have a “data high-water mark”.
• Malware and other mis-configurations that may creep into a Guest and make one VM behave differently than the other desktops in the pool do not persist on reboots. This is a great mechanism for ensuring compliance across desktops.
• [*] Malware Exception - Technically, it is feasible to use the persistent disk to store binaries for re-infection on reboot (such as using an autorun.inf if auto-run behavior is not disabled), but is a low-probability. Either way, anti-virus and other anti-malware is strongly recommended.

PVS Variation #2: Non-persistent Write Cache Disk

Configured the same as above, except that the vmdk assigned to the VM is thin, freshly formatted, and mounted as non-persistent. **This method is the most space-efficient method of all provisioning methods considered.** Here is a screenshot on how the target VM has its disk configured.

Characteristics for this configuration are the same as above, except where noted by [*]. Here are the differences:

• [*] The .vmdk file lives in the same home folder as the VM and is effectively read-only. Instead, writes and subsequent reads are made to/from a REDO.vmdk, which is effectively a VMware snapshot that gets deleted on power cycle.
• [*] Data on the write-cache drive persists beyond guest OS reboots, but NOT through VM power cycles. Any logging or databases such as EdgeSight would need to be made to another persistent location, such as a central file share on the network.
• [*] A VM will have a “data high-water mark” only as long as it is powered on. Dirtied bits are discarded and space is reclaimed with the array on power cycle.
• [*] The malware exception noted re: persistence on the local drive does not apply.

**Note:** In the tests we’ve run, there does not appear to be a significant overhead or performance penalty associated with protecting the write cache disk from dirtying blocks.

Below is a table as seen in the Tintri UI with information about some PVS target VMs. The highlighted “Used GiB” field shows the data high water mark on the PVS targets configured with typical settings for the write-cache disk.

This example has been exaggerated because one of the tests filled all of the standard write-cache disks to find out what would happen. More detail on this test and the outcomes is found in the test section later in this paper under the section heading “Filler Up!”.

In a production environment, we may expect to see a few VMs fill to capacity overtime, but it is unlikely 20+ GB’s would be consumed per VM, as was the case when a logging setting was enabled too aggressively (administrative error requiring version rollback). A scheduled defrag may result in a higher water mark than we’d otherwise expect as well.

Take advantage of all the features available in the Tintri UI, including these views. In the event that a larger amount of used data is consuming capacity unnecessarily (due to dirty bits), consider deleting the vmdk’s and copying contents to freshly thinned (new) disks, or follow the procedures available for reclaiming space within disks using block zeroing (sdelete.exe –z) and storage vMotion.
The Good, the Bad & the Ugly

This section is devoted to highlighting Pros, Cons and Gotchas as they apply to PVS. In some environments weakness listed may be a showstopper, whereas in other environments, it may not cause any inconvenience at all. One size does not fit all, so one needs to weigh the importance of each strength/weakness based on the environment in which the solution is going to be deployed.

Pros

Provisioning and maintenance

• Easy to update Master Image.
• Versioning allows for a good, simple update workflow of with Maintenance mode (edit), Test and promote to Production.
• Simple rollback if latest image has issues.
• Audit trail for all image changes.
• VM properties, including advanced settings such as CPU/MMU Virtualization, can be configured on your target template VM and are applied to all clones.
• Persistence of write-cache disk can be useful for things like Edgesight database or logging behavior that may result in a reboot or crash.
  • Note: This does not apply to PVS Variation #2, using non-persistent disks
• Changing the base vDisk on a collection of targets is quick and simple
  • The vDisk can be dragged onto a collection in the UI and the targets are automatically updated on next reboot.
  • Although Versions is best for most updates, you may want to change the base vDisk for a major update, such as replacing your Master Windows 7 image with Windows 8.
• Target VMs can be configured with multiple disks. This can be useful to use different disks for OS, page files, user profiles, etc.
• Can be used for XenApp server images as well as XD Virtual Desktop images.
• Can be configured for scheduled, automatic master image updates with WSUS or SCCM in the PVS console.

Small storage footprint.

• Overall storage footprint, including target VMs is even smaller when using PVS variation #2 (non-persistent write-cache disks).

Availability, data protection and disaster recovery.

• Highly Available: simple to make solution highly available by placing multiple PVS servers on multiple stores.
• Backup and Simple Backup & recovery of the master image(s). Configuring a backup job for the PVS server includes the master images, versioning, AD settings and other defined config.
• Simplified DR.
  • Just need to replicate PVS servers, all data is contained within.
  • Target VMs can be created at remote site and left powered off. Create targets with the MACs of the remote systems so that both production and DR targets show in PVS.
Cons

Provisioning and maintenance.

- Initial creation of targets is tedious if done manually and prone to error if not handled carefully.
  - The Tintri UI helps simplify by allowing quick and simple creation of many clones, but adding them as targets within a PVS collection can be bulky.
  - **Workaround 1**: RVTools can be used to export a list of VM names and MAC addresses from vCenter and then import that into PVS as targets using a CSV.
  - **Workaround 2**: Process can be scripted, but is not automated out of the box.
  - **Workaround 3**: Auto-add works, but doesn’t match the target name (and subsequently the logical identity in AD and on the network) to the name of the VM. Targets were matched randomly based on MAC address for multiple boots, but could be booted one at a time to match the names sequentially. Not a recommended approach for a large number of VMs, but is quick & easy if only adding a few more target VMs to a collection.

- Adding additional VMs as targets requires the same effort as creating the initial set.
- When a vDisk is in use by even a single system (even if using versions), caching mode and other vDisk options are greyed out and can’t be modified.
- VM property modifications on target VMs after initial cloning are manual or require scripting for a large number of VMs.
- Target disks may start as a clone from a single VM, but begin to fill with dirty bits from the write cache. Over time, space reclamation may be required to re-thin disks, or throw out the persistent write-cache disks and replace with new thin ones.
  - **Note**: this limitation only applies to PVS variation #1.

- Target VM can’t survive a guest OS reboot.
  - **Note**: this can be the desired behavior for keeping consistent images, but can be problematic for testing changes that requires a reboot.

Storage and IO consumption

- Per-VM read/write metrics in VMstore aren’t accurate to actual IO used within the guest. Not all disk IO shows on the VM object because any READ IO from the base vDisk is streamed over the network from the PVS server, not read directly from storage.
  - Note that is not a VMstore limitation, but is a result of the deployment configuration.
- Base disk is not easy to clone/export to create a standard VM that does NOT rely on PVS.

Additional Infrastructure Overhead

- Compute or storage overhead are minimal for the additional Server VMs required to support a PVS infrastructure, however, there is additional management associated with these extra VMs (backup, patching, security, etc). Other provisioning methods don’t require the additional server VMs that PVS does.
- Requires SQL Server, which requires additional administration & skillset requirements, but since XenDesktop services also have a SQL requirement, so this is unique to PVS. It is noted here because the other Provisioning methods (MCS & Tintri Native Clones) don’t have this additional requirement.
Gotchas

**Cache on Device Hard drive:** Caching options have already been covered, but is worth mentioning again as a potential gotcha that could trip you up and cause you a lot of pain and many headaches. If you find a Write-Cache folder in your PVS store folder, this is a good indication that your write-cache is not properly configured to use your target’s hard drive.

✅ **Do:** Make sure the drive is formatted, assigned a drive letter, and the System account (by default) has the necessary permissions to create files and folder in the root of the drive.

**Permissions & Service Accounts:** Avoid using a common service account for your PVS services as you may use for other services on the network or within your Citrix infrastructure. In the event your service account gets locked out, or during routine password change, you do NOT want your services to fail and interrupt your XenDesktop users.

✅ **Do:** Usage of unique accounts for your PVS services is highly recommended.

Service accounts and permissions can get tricky in a PVS environment. NTFS permissions on your PVS stores can give you grief if they are not configured correctly. An example of this is when the Streaming Service in the PVS service has no access to the files in the PVS store folder, as shown in this screenshot (Store path not found).

If you are using multiple PVS Streaming Services (recommended), your misconfigured permissions may exist on one Store or server, but not another.

While working with Versions, you may also encounter some issues where sometimes you can easily and instantly create a new version, and other times it may hang, or fail altogether. The same applies to the action of merging version updates and base images. In this case, the PVS console (GUI) uses the Citrix PVS Soap Service to interact with files in the stores, so make sure the account associated with that service has appropriate access. Permissions & service accounts details are out of scope for this paper, but it is worth mentioning as one more area to check.
while you are getting things setup or troubleshooting a problem to save hours of troubleshooting down the road.

**PVS Store File Syncing & Replication**: When you create new vDisk versions and create updates, files in your Store may be modified, or additional version files are added. There is a replication status that will show you whether the files exist in your other PVS Server’s Store, but don’t wait for these to replicate on their own!

While it’s nice there is a feature to show if you have the same data in sync between your PVS Server Stores, there is no mechanism to replicate the data. Do this manually by copying files across to the other store, or setup an automated mechanism to do this, such as DFS.

If you want to quickly update stores and NOT consume additional disk capacity, you can create a copy of the underlying vmdk and mount it in your other server, assuming both PVS servers share the same underlying VMstore.
In order to create your PVS-based Machine Catalog, we need to install PVS infrastructure components, create a vDisk (master image) and create targets, then we can move proceed to creating the machine catalog within Citrix Studio. This section describes the process of creating each and provides tips and advice to simplify the process using techniques exclusive to a Tintri VMstore.

Creating a vDisk

The vDisk can be created by importing an existing Master VM into a vDisk or a new one can be created from the PVS console. We’ll assume that you already have a XenDesktop image to work with, so it’s easiest to create your vDisk by importing your VM. To import your VM, run setup.exe provided with PVS installation files with the VM you want to base your master image on and follow the procedure documented within the PVS administration guide.

If you create the vDisk within the PVS console, present it as read/write to a VM to install an OS to it, install applications, and make other configuration changes needed within your Master image. Alternatively, a blank vDisk can be presented so it can be written to during the imaging process, as long as it is mounted as read/write to the Master VM you are imaging.

- **Do:** Make a clone using the Tintri UI or vSphere client before running PVS imaging tools within your VM. This will prevent undesired changes to your Master Image VM that would make it incompatible with MCS or some other provisioning method.
- **Do:** When you’re creating a vDisk, change it from the default “Fixed” (aka Thick-provisioned) to “Dynamic” (Thin-provisioned).
- **Do:** Set the vDisk block size to 16MB to match the size at which the underlying vmdk file.
- **Do:** Your underlying store should be placed on a vmdk that is also thin, for maximum capacity savings.

**TIP:** Give yourself plenty of working space in your vDisks and vDisk Stores. Using Dynamic (Thin) vDisks and Thin VDMKs (for the Store) won’t consume any additional usable space from your array by over-provisioning; space is only consumed for data written.

Rolling up updates into new “merged updates” and “merged bases” requires a lot of extra capac-
ity because both the new file and previous files all need to exist simultaneously in the same store. Excess space in your base C: drive will allow for future growth when new apps & updates need to be added, and won't take up additional space.

If your master image doesn't have at LEAST 40 GB allocated to it, take advantage of the opportunity to grow the disk at time of imaging to increase the size (shown here).

Do: If your master image doesn’t have at LEAST 40 GB allocated to it, take advantage of the opportunity to grow the disk at time of imaging to increase the size (shown here).

TIP: To save even more capacity, make a copy of the Store vmdk and mount it to your secondary Streaming Server (assuming both are on the same VMstore). VAAI will automatically engage when using the datastore browser or PowerShell copy commands.

Creating PVS Targets

Targets are VMs that will boot up and stream its OS volume (C: Drive) over the network from one or more PVS Streaming servers. The vDisk (i.e. master image) that a target boots from is determined in the properties of each target object that are created and managed within the PVS Console. A target uses the MAC address from the requesting VM to determine what its logical identity (i.e. Windows machine name) will be on the network and in the Active Directory, and what vDisk version to present to it.

Targets are stored in “Device Collections”, which are logical groups that allow you to manage many targets that share common characteristics, such as which vDisk is used. Create a unique Device Collection for each Machine Catalog you will create in XenDesktop.
Use a name that makes your Device Collection easily identifiable with its associated Machine Catalog name for simplified administration.

When using machine catalogs created with MCS, Virtual Machine size (vCPUs, Memory, etc) are defined when creating a catalog. With PVS, properties of the VMs are defined within vCenter for each VM. Before we can add targets into PVS, we need VMs that map to each target (via MAC addresses). Here are the steps required to create a large group of target machine objects:

1. Create a VM to be used as a template, or use an existing template if you’ve already created your base PVS template VM:
   - Create a new Virtual Machine and define its properties (# of vCPUs, Memory, etc). Do not assign a virtual disk yet.
   - Using another VM that already has windows installed, create a new thin-provisioned virtual disk (.vmdk). The disk will be used as your write-cache and should be large enough to hold any data that needs to be cached between reboots. We recommend 40 GB, which should be more than enough in most cases.
   - Within Windows, create a volume on the disk, format the volume, and assign a logical drive letter (ex. D: drive).
   - Remove the new vmdk from the VM used to format it and copy it to the home folder of the new VM you created in the first step.
   - Mount the new vmdk to your new VM as SCSI 0:0.
   - **Note:** Decide whether your targets will use PVS Variation #1 (typical) or #2 (non-persistent) for your write-cache disk, as discussed in the previous sections. For maximum space efficiency, mount this disk non-persistently.
   - Boot up your new VM into the BIOS and modify the boot order to reflect the option configured in your PVS server configuration. In this guide, we’ve been discussing network boot (PXE) bring Network to the top of the boot order list.
   - [Optional] PVS can also be configured to boot from a CD-ROM iso or small vmdk boot disk instead of a PXE-boot from the network. If you’ve configured PVS for one of these alternatives, mount your ISO or vmdk to this template VM now so that it is configured in all clones you create from it.
2. Using the VMstore UI, create clones of your template VM. Do NOT apply a customization.

3. MAC addresses from all of the newly created VMs are required for creating PVS targets. Use a tool such as RVTools to export information about your newly created VMs from vSphere in order to create a CSV file that can be used to import to create all of your targets in at once.

4. Use Excel to create a CSV (comma separated value) file using information exported from RVTools and format it as follows (without a header row): VM_Name, MAC_address, PVS_Site, Device_Collection_Name, Description.

5. In the PVS console, right-click on your farm (top node), and choose “Import Devices”. The option is also available if you have an existing device collection by right-clicking on it and selecting Target Device – Import Devices. Both options are shown here:

6. An import wizard will prompt you for a .csv file to import. Select the file and hit Next.
7. When prompted, clear the checkbox for “Apply template” and choose whether or not to create the Device Collection and Site specified in the import file, if not already created. Press Next.

8. Review the list of VMs being imported and click finish.
9. After your targets have been created, they need to have a vDisk associated with them. The quickest & easiest method of doing this is to drag a vDisk from onto a Device Collection to assign it to all devices in the collection.

10. Active Directory accounts need to be created for your targets. Right-click your new Device Collection and choose **Active Directory – Create Machine Account...**
11. Choose the AD domain to create machine accounts in and select an OU. Click the first target listed, hold SHIFT + END to select all targets, and then click Create Account.

12. Check your Active Directory to confirm the accounts are created.

13. As a final step prior to adding these systems into a XenDesktop Machine Catalog, boot up one of your PVS targets and test it to confirm it works as expected. You should be able to log into your domain without any errors and confirm the identity of your system machines by using hostname from command prompt.

14. Your targets are now ready to be added to a XenDesktop Machine catalog. These steps can be automated via scripting if target creation is something you need to do often in order to operate at large scales. The semi-manual process shown above takes roughly the same amount of time, regardless of whether you are creating 10 targets, 100 targets or 2,000 targets.

Creating a Machine Catalog based on PVS

This last part is straightforward and very similar to creating a Machine Catalog using MCS. The step-by-step process can be found in the XenDesktop 7.1 Quick Start Guide. When prompted to choose which technology you are using to manage your desktop images, select Provisioning Services (PVS).
When prompted for a device collection, enter the IP or FQDN of your PVS Streaming Server and press connect. You should see a list of your Device Collections. Select all the collections you want to add to this Machine Catalog and press Next.

When naming your Machine Catalog, choose a name that is similar (or the same) as your PVS Device Collection; this will simplify administration.

Once you have a Machine Catalog, you need to add the machines to a Delivery Group in order for your users to access them.
Das Bootin – The PVS Boot-up process

In the case of PVS, the boot process is a little different since there is no “C: drive” to read. A VM configured as a PVS target boots up and receives an IP on the network via DHCP, receives a bootstrap via PXE server, and the bootstrap program instructs the VM on how to content to a PVS Streaming server to stream a vDisk (C: drive). Once the virtual C: drive (Citrix virtual disk, not to be confused with a vmdk) is mounted to VM, it is read similarly how a standard hard disk would be read. This is an over-simplified explanation, but hopefully you have a good sense of the process. Technical details on this process can be found in Citrix’s eDocs: http://support.citrix.com/proddocs/topic/provisioning-7/pvs-technology-overview-boot-file-get.html.

Let’s assume we boot up a standard Windows 8.1 VM and we read 300 MB from the OS disk. In PVS configurations, we would assume that the PVS Streaming server would read the same 300 MB from the vDisk, and stream it over the network to the target VM to read it from the network, instead of a locally attached hard drive.

Here’s an overview of the simple boot test we performed to view the information below:

1. First, we **reboot the PVS Streaming Server** so that no vDisk information was cached in memory. This was an important step so we could demonstrate what happens when the first target VM boots. After the PVS streaming server was booted up, we let it settle and stabilize for over 30 minutes prior to beginning our target boot tests. This was ensure all disk and network activity observed directly associated with the target boot process.

2. The vDisk for all target VMs is the same one and is located on the PVS streaming server in its own PVS Store,

3. The PVS Store has been placed on its own dedicated vmdk (SCSI 0:2) mounted on the PVS Server VM, logically mapped as F: drive within windows.

    **Do:** Dedicate one vDisk per PVS Store per vmdk to make any IO associated with a vDisk clearly identifiable in the VMstore UI.

4. Power on a PVS Target VM (#1).
5. Wait 2 minutes, and then power on one more PVS target VM (#2).
6. Wait 2 minutes, and then power on one more PVS target VM (#3).
7. Wait 1 minute, power on one PVS target VM (#4).
8. Wait 1 minute, power on five (5) PVS target VMs (#5/6/7/8/9).
In the next screenshot, we’re displaying real-time latency and IOPS (10 second intervals) in the VMstore UI of only SCSI 0:2 of the PVS Streaming server VM. SCSI 0:2 is the vmdk that has holds the vDisk that will be read by PVS Target VMs.

**Note:** One of the very powerful VM-aware features of the VMstore is the ability to not only identify and isolate per-VM activity (real-time & historically), but also isolate per-VMDK activity as well.

Network activity (packets transmitted & received) of the PVS Streaming server was captured from vCenter and placed within this screenshot to map time stamps between the real-time graph in vCenter (1 hour, 20-second intervals) and the real-time graph in the VMstore UI.

Each **Green Star** shown in these images marks when the **Power On** command was issued to each PVS target VM. After the first target VM is powered on, the vDisk is read by the PVS streaming server. The read activity (yellow) shown above shows this was read at a rate ranging between 100 and 500 IOPS. After a couple minutes, there is no more read activity.

In the network graphs (shown inlaid in the screenshot above and in a larger screenshot below) packet transmissions are observed on the PVS Streaming server VM. This is the data contained in the vDisk being sent over the network to a receiving PVS target, which is reading the vDisk remotely (streamed).

Two (2) minutes later, the 2nd PVS target VM is powered one, as shown by the 2nd green star from the left (middle green star in the top picture). If we look at the 2nd spike in the network graphs, we can see that a similar amount of data is transferred from the PVS Streaming server to the target.
What we **don’t see** is the equivalent IOPS on the PVS Streaming server, indicating that the vDisk is NOT being read from the datastore. If this were a standard VM boot, we should see a very similar IO pattern displayed for every VM booted that is configured identically to one another as it reads its disk to boot.

As time goes on and the subsequent target VMs are booted up according to the timings described above, we continue to see network traffic for each boot, but the vDisk is no longer read. At this point, all data from within the vDisk that needed to be read in order for a Windows VM to boot has been read and placed into memory on the PVS Streaming server. The PVS server does not need to access the vDisk again until different blocks from within the vDisk (not previously read and cached in memory) are requested by a target VM.

Examples of scenarios when a PVS target VM operation that would cause reading of sections of the vDisk that were not previously read during the boot process:

1. A user accesses a program folder not accessed during boot in order to launch a program. Example: User launches Microsoft word.
2. An anti-virus scan is initiated, causing a READ of the entire vDisk.

This boot PVS process demonstrates how important it is to allocate an appropriate amount of RAM to your PVS Server. The data cached in RAM is the active working set of a VM and when there is more data than capacity to place in RAM, older data must be evicted or overwritten. This works similarly to how the VMstore determines what data to place in flash, but without the inline dedupe and compression that occurs in the VMstore.
Consideration Factors

This section provides additional factors specific to this method that are useful in evaluating how best to run your XenDesktop infrastructure.

Space Utilization & Capacity Considerations

PVS offers the most space-efficient use of disk that we’ve reviewed, especially when implementing Variation #2 (non-persistent write-cache disks). Although space is reclaimed when VMs are power-cycled, there’s still an operation high-water mark that can be observed. This includes the all of the data change held in REDO logs and swap files (equivalent to vMemory allocated to each VM), as well as the PVS Streaming and database servers and any data contained within each.

**TIP:** Want to get a sense of how much data changes on a single VM or for your entire VMstore? After booting up from a clean power off, have a look at the Tintri UI and check out the “Space” graphs for a particular VM, or the whole store.

Visibility

With the exception of the boot process, most IO associated with an individual VM in a PVS-based Machine Catalog is visible in the Tintri UI or the vSphere Web Client, assuming the Tintri plugin is installed. This is can be very useful for determining if a particular is experiencing latency issues in real-time.

Persistence

The provisioning techniques in this paper have covered mainly stateless, non-persistent desktops configured in delivery groups as random pools. If your desktops require persistence across reboots and power cycles, determine if it’s your whole desktop you need to keep, or just the User’s profile and the feeling of persistence delivered in the user experience.

With PVS, keep in mind that ALL changes to the C: drive (and subsequently the registry because it is on the C: drive) are lost on reboots. However, PVS can offer some persistence if you decide to use the write-cache hard drive (D: drive) that has been configured on your VM to store some data.

In Variation #1 (typical write-cache), the D: drive persists always and can be used to store data that is unique to the identity of that VM, such as the unique database required for Citrix Edgesight. However, in Variation #2 (non-persistent write-cache), all data written to the D: drive will be lost on a power cycle of the VM. Data will persist across OS reboots because the power state of the VM does not change. Due to this non-persistent nature, this method is not a good candidate if Edgesight is required, OR, consider adding an additional disk that DOES persist, dedicated to Edgesite and continue to enjoy the space-saving benefits of a non-persistent write-cache drive.
Do: If persistence is important to your organization, consider using Tintri Native clones or do some additional research and consider implementing Citrix Personal vDisks. Citrix Personal vDisks can help you, but are out of scope of this paper.

Audit & Compliance

Environments with strict audit and change control may need to provide evidence for certain administrative activities. PVS has great audit logging capabilities, but they are NOT enabled by default. Once enabled, you can view the audit trail to see details of changes are made to configuration, vDisks & versions, targets, etc. The audit trail will record the date & time of changes, as well as who made the changes.

Do: Check the logs for yourself to ensure that actions you expect to be recorded are in fact recorded, and are sufficient evidence to meet the audit and compliance requirements of your environment.

Also be sure to check the retention settings to make sure your logs persist as long as you need them to. You may not need the logs to go back further than a few months if you choose to rely on point-in-time backups of your PVS servers, which will have point-in-time copies of the audit trail around the date/time in question.

Does activity within your desktops need to be maintained for security or compliance purposes? If so, you can’t depend on the event logs of stateless systems as all traces of guest activity is lost on reboot or power cycle.

Do: If the Event logs or other monitoring software within the guest require retention, consider a centralized logging solution to record and maintain your logs outside of your desktops VMs.

Additional Factors

In addition to these, there are many other factors to consider that are common to all of the methods of provisioning that are detailed in the XenDesktop 7.1 Provisioning and Machine Creation Deep Dive paper and we recommend you review that paper as well.
“The Only Thing That Is Constant Is Change” – Heraclitus
You don’t need to be in IT very long to appreciate this quote! IT environments constantly change; staying on top of the changes is critical. This section is dedicated to raising awareness to some of the overall support & management costs.

Adding additional VMs to a PVS-based Machine Catalog

With PVS, adding additional VMs to a machine catalog requires setting up additional targets. Follow the same steps you performed for your initial targets in the “Creating PVS Targets” section above.

✅ **Do:** Clone the same Tintri snapshot of the Template VM used to create the first batch to make new clones for importing into a device collection as targets.

Once you’ve added more targets to the device collection associated with your Machine Catalog, choose the “Add Machines” options in Citrix Studio. Following the steps in the add machines wizard to select the device collection you’ve added machines and bring the additional machines into your catalog.

Once you’ve increased the number of machines in your catalog, edit your delivery group to include the new VMs in it as well, making them available to accept additional user sessions.
Image Versions

Image Versions: This is where PVS really excels and is much more robust than any of the other provisioning methods. To enable vDisk versions, you must set your vDisk to Standard mode (multi-access), as discussed earlier in the “Creating a vDisk” section. To access versions of your vDisk, open the PVS console, right-click the vDisk and click Versions.

Do: In the vDisk Versions window, use the Properties button to create a description of each version to track what is being changed, as shown in the lower portion of the screenshot above.

Versions can be configured as Maintenance, Test or Production. Targets will boot-up using the newest production version (Version #3 in the screenshot shown above, as marked with the Green Check), unless a different version is specified.

Special Target Types unique to PVS

Up to now, we’ve only discussed targets that of one type: Production. There are 2 other types available for configuration in the properties of target: Test and Maintenance.
Create 2 new Device Collections, one for Maintenance and one for Test Targets. These targets will have a special purpose for interacting with vDisk versions. These special targets are created using the same process as your production targets, cloned from a template VM and imported into Device collections.

**Do:** When creating the Maintenance and Test VMs, name the VMs so they are clearly identifiable as their intended use and place them in their own VM folder, separate from the others. Example names: PVS-MAINT, PVS-Test1, PVS-Test2, etc.

The reason for creating separate Device Collections for these special types is to prevent them from being added to your Machine Catalogs with all your production targets.

**Maintenance Targets**

Create only one (1) maintenance target and assign it to its own device collection. After creating the target, change it to become a Maintenance target by changing the Type in the target properties (screenshot shown above in previous section).

Once you've created a new vDisk version, boot up your maintenance target to modify contents inside your new version.

**TIP:** Limit access to your maintenance VM to only those admins accountable for making change to your production virtual desktops. Use “kid gloves,” be very cautious and keep good records of every tiny change made as this image will become the new master for ALL users after it passes testing. Small mistakes here can be magnified when 1,000’s of desktops are hitting the error.

**TIP:** Access the maintenance VM using the VMware console ONLY. Do not connect via RDP or your risk injecting print drivers from the client workstation into your master image, which could negatively impact your users. This is even more important for master images of shared desktop servers (XenApp) where print drivers have caused many Citrix admins pain since the beginning of time!
When you boot up your new Maintenance Target, you will be prompted for a version to boot from. Your options are:

- The latest maintenance version (only 1 maintenance version allowed at a time).
- All Test version.
- The latest production versions.

The screenshot shown above corresponds to the versions list in vDisk Versions screenshot in the “Image Versions” section above.

Select option 1 to boot the new maintenance version and allow you to make changes. After the maintenance target VM boots, log in through the VMware console and confirm that the Citrix Virtual Disk is in read/write mode. To verify the vDisk access mode, right-click the Virtual Disk icon in the taskbar and select Virtual Disk Status. The mode should be Read/Write (as displayed in the example on the right).

There are circumstances where the image may be in read-only mode, and all work that goes into make changes will be lost on reboot. Depending on the nature of the changes, this could cost you many hours of your time.
Do: Make it a habit to take the extra 30 seconds to check Read/Write status prior to commencing work on changes.

Note: Only the version in Maintenance mode will allow Read/Write access to make changes. Choosing one of the Test or Production vDisk versions will not allow read/write access even if you are using the Maintenance Target to access it, regardless of whether there are any other targets accessing the same version or not.

Test Targets

Functionally, the new test targets are very similar to your production VMs, with one major exception. When you power on a Test Target VM, you will be prompted to choose a version to boot from and your VM will not automatically boot up on its own. This is very similar to the maintenance target and the only difference here is that the version flagged as “maintenance” is not an option to choose from.

- Create anywhere from one (1) Test target to several dozen, depending on how many test users you have.
- Adjust your target properties to Type=Test and add all of your test targets to a dedicated Device Collection.
- Create a new machine catalog in XenDesktop using Citrix Studio and assign these targets to that dedicated Test catalog.
- Create AD accounts for your new test target VMs.
  - Consider extending your testing to incorporate GPO changes as well by creating a dedicated OU for your test VMs. Make copies of your production GPOs and link them to your new Test OU. When testing a GPO change, test the change on the Test OU to prevent causing harm to your production desktops. When successful and ready to promote, repeat the GPO changes in production.
  - In some cases, changes must be coordinated between a GPO change and master image update, making it very advantageous to implement a new OU now if one does not already exist.
- Create a dedicated Test Delivery Group as well, and add the machines into it. Use an AD-based security group to manage Test users for provisioning the delivery group only to that group of testers.
After booting up the test target, open the VMware console to choose which version to boot:

- Any of the versions set to “Test”.
- Only the latest production.

Based on the Versions displayed in the screenshot in the “Images Versions” section above, only Version #4 is configured as “Test”. Version 3 is the latest production version, so it is a valid boot option as well. vDisk versions are listed at the end of the vDisk name: VDISKNAME.VER#. Press 1 and your test target will boot up to your test version and become available for logons.

While in a Test version, all changes made are lost on reboot, just as they are with a production PVS target.

Do: When possible, test proposed master image version changes in a test target BEFORE making the changes in your maintenance version.

Note: Changes made in a maintenance version can NOT be undone, other than deleting the version and creating a new one. The only changes that can’t be tested this way are ones that require a reboot, since the reboot process will wipe any changes before they can be tested.

When test users log into StoreFront to connect to a Virtual Desktop (or published app), they will see the Test Desktops you’ve created and can log into them to perform UAT (User Acceptance Testing) to test functionality of the changes made in the new version, such as adding a new application, and also test to ensure there are no new problems introduced in your new version.

Do: Create and power-on as many Test VMs as required to accommodate the number of test users you’ve identified within your organization.

As careful as you are with version updates, it is common to fix one issue, but inadvertently create a new problem. By incorporating a solid test procedure into your change control process, you’ll reduce instability and create a framework of accountability and control in your desktop lifecycle management. Version after version, your desktops will improve, but there will likely always be a list of enhancements requested, bugs to fix or software to update. The machine update process is important, so take the time to set it up properly in a robust and scalable manner. PVS offers the tools to do this and we recommend you use them.
The Image Update Process

Images should be updated often to keep them current with windows patches, security hotfixes, AntiVirus definitions, and other required enhancements, such as software upgrades. To create a new version and eventually use it to update your master image, highlight the latest version and click “New”. A new version will be created in Maintenance access mode, which is inaccessible to Production and Test target types (described in the “Special Target Types” section).

**Note:** Only one (1) maintenance version allowed at any given time.

Boot up the maintenance VM, select the latest version identified as “[maintenance]” in the boot menu and log onto the VM using the VMware console. Before proceeding with any changes, verify that the virtual disk mode is **read/write**.

Make your changes through the VMware console and reboot as necessary. Your changes will persist across reboots because they are being written directly into the vDisk stored on the PVS server. After changes have been completed, add a final step of updating your anti-virus definitions and perform a full scan, then shut down the maintenance VM.

In the vDisk Versions window, select your newest version that is set to Maintenance, and click the **Promote** button on the right. When prompted, choose to promote to Test.

Next, boot up your Test VMs and have your designated users test the image for sign off, or perform any other necessary steps your change control procedures require.

- In the event your tests fail, power off all the test target VMs to release locks from the test version and then choose click “**Revert**” in the vDisk Version window to return version to Maintenance.

- If the change being made is complex and takes several attempts, consider deleting the version that has had many attempts made to it and create a clean one, re-applying only the necessary changes ones. If a lot of troubleshooting and debugging was necessary, you don’t want to carry forward the history of that by promoting the version to becoming your new master image to be provisioning to virtual desktops.
Once tests are complete, **Promote** the version again to make it a production version. Be sure that "**Boot production devices from this version**" option is set to "**Newest Release**" (top of the vDisk Versions window). When prompted, choose to make the new version available immediately. None of your running production target VMs will be affected by this and they will continue to run the same version they are on until the next time they are rebooted.

![Promote](image)

**Promoting new images to production**

**Do:** We recommend you use rolling reboots over the course of 2 weeks, automated via PowerShell (or other) scripting.

This allows you to update 10% of your targets per day when new updates (or rollbacks) are pending. When there are no updates pending, the reboots can free up memory and bring targets back to the known good state of the master image, which has an added benefit of enforcing compliance across your environment.

For more detail on how this process would work, refer to the Change Control section the XenDesktop 7.1 Provisioning & Machine Creation Deep Dive paper which discusses all 3 provisioning methods and considerations common to all.

**The Roll Back process**

No matter how well you test the latest changes to a master image, sooner or later, something may slip through the cracks and negatively impact your users. In these cases, you'll want to roll back to your previous last known good build. More thorough and higher quality testing will certainly reduce the need to rollback, but is unlikely to eliminate it altogether; so rolling back changes needs to be considered when deciding upon which deployment method to choose.

What happens when a change goes wrong and needs to be rolled back? How many users are impacted? Will there be any downtime as a result? The answers to these questions go back to how you introduce your updates in the first place and your overall change control procedures.
In the vDisk Versions properties, change the "Boot production devices from version:" from “Newest released” to a specific version that you know is good and free from the problems introduced in the latest update.

In the screenshot above, version 3 is the Newest "released" (Production), and is selected as the version for targets to boot from (as seen by the green checkmark). Change to the previous version (2) in the dropdown and press down. That’s how easy rollback is. VMs that are accessing any other version are not interrupted and will boot from the version you’ve specified the next time they reboot.

Version control with a solid rollback mechanism in PVS makes for a great workflow to integrate into your change control procedures to allow test and gradual promotion into production. Combine this with the automated rolling reboots recommended in the XenDesktop 7.1 Provisioning & Machine Creation Deep Dive and if a rollback is not urgent, simply change the option so that all the scheduled reboots don’t boot up the Newest release, and if the version isn’t modified to correct the issue in it that prompted a rollback, and machine currently running off it will end up on the older version when rebooted. Once you’ve correct the issues, set the option back to boot from “Newest Released”.

Cleaning Up Old Versions

After you’ve updated your master image version several times and have the last couple versions are known to be good, you’ll want to clean up older version. As long as there are no dependencies on the older versions, they can be removed to reclaim space in your PVS store.
Updates are chained serially on top of one another, similar to snapshots. Base disks are .vhd files, and updates are “xxxx.avhd” files. Updates (.avhd files) require a base disk (.vhd file) to be functional, just as a VMware snapshot file (basedisk-0000001.vmdk) requires the underlying disk that was snapped (basedisk.vmdk). In the PVS Store, the versions look like this:

Here is an illustration of how base disks and version updates are dependent on one another:

- Base image: Deployed to users (base.vhd)
- Version2 = Base (base.vhd) + Update1 (base.1.avhd)
- Version3 = (Version2 + Update2) OR (Base + Update1 + Update2)
- Version4 = (Version3 + Update3) OR (Version2 + Update2 + Update3) OR (Base + Update1 + Update2 + Update3)
- Etc...

Use the “Merge updates” and “Merge Base” options to roll up your versions in to a single self-sustaining base. In the example shown above, if you choose “Merge base” on the latest version (version 4), a new version will be created as Version 5. This new contents of this new version will be logically identical to Version 4 from the guest OS perspective, but from a file structure perspective of the vDisk, it will be completely independent of previous versions (.vhd file).

Older versions that no longer have newer versions dependent on them are marked with a Red X in the Versions dialog. Once merged in a new version and none of the previous version files are in used by PVS target devices, they can be deleted if they are no longer need.

**Versioning Challenge: Single-Threaded Dependencies**

With PVS, versions are stacked on one another progressively, similar to the way snapshots work on a VM. If a version deployment doesn’t go smoothly and needs to be rolled back, any subsequent version that was built on top of it is also rolled back and will need to be recreated. This can be tricky with multiple admins taking care of the same golden image, especially in a fast-paced environment where many change requests are made from the users to respond to evolving business challenges (i.e. new software installations/changes, adding print drivers or other device drivers, browser version updates, java updates, etc).

In the event that a major change is requested that will take longer than the typical update cycle with a longer testing period, you may want to consider making a copy of the vDisk to work with instead of using a version. If the testing takes longer than expected, there is a risk that all of the routine change will queue behind the one major change, leaving your VMs in a state that may
not comply with some of your controls (for example, all VMs are updated within 1 month of patch release). If too many changes are made in a single version update, you increase the risk of something wrong that requires a rollback, which sets you back even further.

If you decide to use an additional vDisk, remember that it is easy to change a vDisk for every target in a collection simply by dragging the vDisk onto the target in the PVS Console. Once you've swapped all targets to your new vDisk, continue creating versions within it. After you progress several versions beyond the major update, consider deleting the old vDisk to reclaim capacity.

**Do:** If you followed the recommendation of creating one vDisk per VMDK on your PVS servers, consider deleting the vmdk and provision a new, thin one in its place to reclaim the dirtied blocks from the overall capacity available in your VMstore.
Data Protection & Disaster Recovery (DR) Overview

Backup and DR for XenDesktop environments need to consider more than simply ensuring good copy of data is available somewhere, such as a backup tape or offsite location. Desktops aren’t often backed up for their unique data, but are instead the tools (OS + Applications + Configuration) that allow users to access their data (databases, email, application-specific files, etc.). A full and comprehensive DR strategy is out of scope for this paper, but this section provides some direction on what should be considered for backups.

XenDesktop infrastructure consists of 3 major components:

- XenDesktop Infrastructure Servers – XenDesktop Controllers, StoreFront, SQL servers, etc.
- XenDesktop Machine Catalogs – The virtual workstation VMs.
- Dependencies – User profiles, redirected folders, data, and the rest of the corporate infrastructure (i.e. the goods that users access their virtual desktops to access).

In the paper, we’ll look only at the Machine Catalog. In XenDesktop 7.1 Provisioning & Machine Creation Deep Dive, we’ll cover some backup & DR aspects of XenDesktop Infrastructure Servers, but the dependencies are largely out of scope other than mentioning them so they are not overlooked.

Backup & DR of PVS-based Machine Catalogs

With PVS, there are 2 components to be concerned with when protecting Machine Catalogs:

- PVS Infrastructure Servers (Streaming servers & SQL)
- Target VMs

PVS Infrastructure Servers - Backup

Any backup software that can backup VMs can be used to make backups of your PVS Infrastructure servers. Backups of the PVS servers will automatically include your master images (vDisks and vDisk versions). PVS infrastructure includes a SQL Server component and needs to be backed up with application-level processing.

**Do:** Follow best practices when it comes to SQL database backups. In particular, ensure your backup process truncates your SQL transaction logs to prevent them from growing and eventually consuming all of your disk space.

Backup - Target VMs

In general, target VMs have very little, if any, useful data that needs to be backed up. If you are running with PVS Variation #1 (typical write-cache that persists), you may have unique data you want to back up on each VM, such as unique Citrix EdgeSight databases, or other persistent data that has been stored to the D: drive. And if you are running with PVS Variation #2 (non-persistent write-cache disks), there is no unique or persistent data within the VMs.

Although the contents of the target VMs may not have unique data requiring backup, you can choose to back up the VMs for each of restore so that you don’t need to recreate targets or update MAC addresses on existing targets in PVS, lose VM-specific information associated with historical monitoring data, such as vSphere performance history or vCOPs, which are stored
based on unique VM IDs, or have to rebuild machine catalogs. The target VMs are very small to backup, especially if you went with PVS Variation #2 (non-persistent write-cache disk).

To avoid the scenarios listed above, making a backup of all targets can be done with your backup software to make one-time copies, of scheduled very infrequent backups (monthly or quarterly is probably sufficient).

**DR - Target VMs & PVS Servers**

- **Do:** For DR, consider using ReplicateVM™ to make a one-time replica of your Template VM.

In your DR site, make Tintri clones of the Template VMs the same way that you did to create your initial target VMs. Refer to the “Creating PVS Targets” section that covered the process earlier in this paper and create the same number of targets as you have in your production site (or fewer if your DR plan isn’t intended to run a full user workload). Export the list of VM names and their MAC addresses and import them into a new Site and new Device Collection in the production PVS servers, running in your primary site.

- **Do:** Protect your PVS servers with ReplicateVM and create a replication schedule to replicate them over to your DR site periodically.
- **Do:** With the exception of vDisk version updates, change on PVS server should be minimal, so consider a daily or even a weekly snapshot and replication schedule for DR.

In the event of a failover, bring up your PVS servers in the remote site, and then power on your target VMs. With DHCP and PXE configured on the remote site, your PVS targets will connect to your PVS streaming servers and the DR targets you’ve created will recognize the MAC addresses and stream the same vDisk to these servers as you would have streamed to your production targets if failover was not required.

- **Do:** Ensure that Machine Catalogs & Delivery Groups are configured for the DR collections.

In most cases, this collection can be added to the same Machine Catalog already configured in production so that it is in the configuration and ready for failover, depending on how the rest of the XenDesktop Infrastructure is configured for failover.
We hope that you have found this deep dive useful and enlightening. While there is no “right” or “wrong” method to use for your XenDesktop deployment, there are pros and cons associated with each.

From a performance perspective, we’ve observed more IO associated with an MCS deployment than we did for a PVS deployment.

We strongly encourage you to take screenshots occasionally of known and unknown (yet to be identified) activity as baseline info to compare future graphs against. Question what you see... should this process result in the behavior we’re seeing? Does a software vendor’s or internal developer’s description of a given process match what is actually taking place in reality? In most cases, it’s up to you to figure out what is actually taking place, and using Tintri for your storage is a very powerful tool to give you more insight than ever before. And what should you do with your screenshots? Keep them somewhere... having a record for your own reference (and sanity!) can be very helpful down the road to decide if things are getting better or worse. As an added bonus, these screenshots also make great teaching material to mentor future hires!
Additional Resources – Reference URLs

Tintri Links

- XenDesktop 7.1 Quick Start Guide – Covers all the perquisite tasks required to get a XenDesktop environment up and running on Tintri storage.
- XenDesktop 7.1 Provisioning & Machine Creation Deep Dive – Referenced throughout this paper as “the main” provisioning deep dive paper and contains information that applies to all XenDesktop provisioning methods, and compares the method discussed in this paper to alternative methods.
- XenDesktop 7.1 Machine Creation Services (MCS) Deep Dive – A similar paper to this one that focuses on MCS as an alternative method of provisioning virtual desktops.
- XenDesktop 7.1 Tintri Native Clones Deep Dive – A similar paper to this one that focuses on using Tintri Native Clones as an alternative method of provisioning virtual desktops.
- XenDesktop Best Practices white paper.
- NFS Best Practices.
- Additional resources.

Citrix Links

- Citrix XenDesktop eDocs.
- Citrix Provisioning Services 7.x eDocs.
- Citrix Blog series that focus on image management:
  - Part 1: PVS - The myths and tales of image management.
  - Part 2: MCS - The myths and tales of image management.
  - Part 3: MCS vs View Composer.

Other

- VMware FAQ for VAAI.
- Login VSI – Load Testing tool for VDI environments.
- FSLogix.
Appendix A – Environment Details

This guide was written to be applicable to any server hardware and all supported versions of VMware vSphere. In case your own results vary from what you’ve seen in this paper, here is a high-level overview of the lab equipment used:

**Hardware**

- Storage: Tintri VMstore T540 running Tintri OS v2.1.2.1
- Servers: Cisco UCS B22-M3 blades

**Software**

- VMware vSphere 5.1 U2 – Build 1612806
- Tintri vSphere Web Plugin v1.0.0.1 (includes the Best Practices Analyzer tool)
- Desktop OS: Windows 8.1
- Citrix XenDesktop 7.1
- Citrix PVS Provisioning Services 7.1
Appendix B – Why dedicate a VMDK per Store and per vDisk?

If you are using multiple pools of VMs that are configured with different base images (PVS vDisk), each vmdk has its own performance metrics within the Tintri UI and will give you more insight into which vDisk is generating the IO. Instead of seeing a lot of activity on only the PVS Streaming server, you’ll have the ability to drill down further and identify which vDisk is associated with IO. And why do you want this level of visibility? Because you can! And there is no additional overhead/cost to your total available storage capacity or performance in your VMstore. Using the Tintri UI, here’s the additional intelligence we can gather based on this simple tip of dedicating a vmdk per Store:
Appendix C – Storage space consumption

When data (user profile data, temp data, event logs, updates, etc) is written to the virtual C: drive, the writes are redirected to a .vdiskcache file in the root of the attached cache disk, which we’ve configured as D: drive. Our vDisk is configured as 40 GB on the PVS streaming server, of which approx. 23 GB of space is consumed. When we write 1 GB of data to the “C: drive” in our guest, we see our usage change from 23 GB out of 40 GB used to 24 GB out of 40 GB used. In other words, our C: drive changes from 17 GB free to 16 GB free. The disk usage on the C: drive is purely a logical representation of capacity consumed as the C: drive doesn’t physically exist and is streamed from the PVS server.

Writes are not sent back to the PVS server using the cache option we’ve chosen, and the data is redirected to a cache file on the VM’s local disk. In the example above where we are writing 1 GB of data to the C: drive, the cache file (D:\.vdiskcache) will grow to 1 GB. Let’s assume that our cache disk is thin and provisioned to 25 GB, it will consume 0 GB of storage capacity at first, and then grow to accommodate the 1 GB of data written to it. Within the guest OS, our D: drive will show 24 GB free and 1 used.

In this paper, we are not using a Citrix Personal Disk and we are working with Stateless VMs. When we reboot our PVS target VM, the C: drive is streamed from the PVS Streaming server and our .vdiskcache file is overwritten. In other words, we lose all data change that occurred within the VM on a reboot, including the logs. When we look at our D: drive after reboot, we’ll see the 1 GB used is now gone and our local disk shows 100% free. However, if we look at our datastore, our vmdk that held the write cache disk is still 1 GB and the space is not reclaimed.

Let’s assume we’ve rebooted, our in-guest usage will change to 0 MB used and 100% free. Now we copy 500 MB to our C: drive in the guest. Our write-cache (D:\.vdiskcache) will show as 500 MB, but our vmdk will still show as in 1GB in the datastore. Fortunately, the new data is reusing our dirtied blocks, otherwise our vmdk would have grown to 1.5 GB. At this point, we can refer to our 1 GB used as our data high-watermark.

Until we write more than 1 GB of data, our vmdk won’t grow, but sooner or later, someone will transfer a large file to the C: drive or D: drive and the high-watermark will go up, consuming capacity in our array unnecessarily.

Hopefully this is quite straight-forward. Understanding how thin disks grow and blocks get
“dirtied” is critical to understanding the two variations of a write-cache we’ll explore below. It is also important to have a good handle on this so you can properly size your write cache disks. Assuming we have 17 GB free in our vDisk (virtual C: drive) on boot, our cache should be large enough to write 17 GB of net-new data.

**Page File configuration:** We’ll want to write our page file to our write cache disk directly. If we left the default in place and have it written to the C: drive, it will end up on D: anyway, but we can avoid the computational overhead associated with redirecting write to C: by writing direct to the local disk instead (i.e. D:\pagefile.sys) Your page file should be roughly equal to the amount of RAM assigned to your VM, which is 2 GB in our case, bringing our total up to 19 GB needed in our cache disk.

Although you know that the C: drives within your PVS targets are streamed from your PVS streaming server and the size is independent of what is configured in your VM. In our tests, we found a 20 – 25 GB vmdk was more than enough for our 40 GB vDisk @ approx. 60% used capacity.

✅ **Do:** With thin provisioned disks, the recommendation is to avoid using a cache disk that is the same size as your vDisk to avoid confusion of where your data actually comes from.