All you need to know about Microsoft Nano Server

Mike Resseler
Director, Product Management

AVAILABILITY
for the Always-On Enterprise™
Introduction

Windows Server 2016 is available today. Lots of new functionality and enhancements are included in the new release of this operating system and talking about all of them would require more than one book.

Today, I want to talk about Microsoft Windows Nano Server. Nano Server is a headless deployment option for Windows Server, coming in 64-bit only. For many, it will look very confusing at first, and many will quickly look at it and decide it is not for their environment. I hope to convince you to look further and to give you as much information and guidance as possible. In fact, when you are thinking about your on-premises datacenter, your first thoughts should be to decide if you can run the intended workload on Nano Server, and if not, if you can run it on Core Server. A server with the full UI should only be used when the other two options aren’t possible.

In addition, when your company need to develop a new application, now is the time to figure out if you can do this with Nano Server. One of the advantages of Nano Server is that it will run on many other platforms and your application will be ready for the future.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Some history</td>
<td>5</td>
</tr>
<tr>
<td>Windows NT to Windows Server 2003</td>
<td>5</td>
</tr>
<tr>
<td>Windows Server 2008 and R2</td>
<td>5</td>
</tr>
<tr>
<td>Windows Server 2012 and R2</td>
<td>5</td>
</tr>
<tr>
<td>What is Nano Server?</td>
<td>6</td>
</tr>
<tr>
<td>Why?</td>
<td>6</td>
</tr>
<tr>
<td>Reboots</td>
<td>6</td>
</tr>
<tr>
<td>Server images</td>
<td>7</td>
</tr>
<tr>
<td>Resources</td>
<td>8</td>
</tr>
<tr>
<td>Conclusion</td>
<td>9</td>
</tr>
<tr>
<td>Setting the expectations</td>
<td>9</td>
</tr>
<tr>
<td>Let’s look at the quick installation</td>
<td>10</td>
</tr>
<tr>
<td>A virtual Nano Server</td>
<td>10</td>
</tr>
<tr>
<td>A physical Nano Server</td>
<td>16</td>
</tr>
<tr>
<td>Packages</td>
<td>17</td>
</tr>
<tr>
<td>New-NanoServerImage</td>
<td>21</td>
</tr>
<tr>
<td>Get-NanoServerPackages</td>
<td>25</td>
</tr>
<tr>
<td>Edit-NanoServerImage</td>
<td>26</td>
</tr>
<tr>
<td>Additional information for advanced installation/deployment</td>
<td>27</td>
</tr>
<tr>
<td>Domain join options</td>
<td>27</td>
</tr>
<tr>
<td>Joining the to the domain where the local computer resides</td>
<td>27</td>
</tr>
<tr>
<td>Joining a different domain</td>
<td>28</td>
</tr>
<tr>
<td>Reusing a domain account</td>
<td>29</td>
</tr>
<tr>
<td>Joining Nano Server to a domain online</td>
<td>29</td>
</tr>
<tr>
<td>Using the unattend file</td>
<td>30</td>
</tr>
<tr>
<td>Azure</td>
<td>32</td>
</tr>
<tr>
<td>Using the Nano Server Image Builder or Show-command</td>
<td>33</td>
</tr>
<tr>
<td>Nano Server Image Builder</td>
<td>33</td>
</tr>
<tr>
<td>Show-Command</td>
<td>41</td>
</tr>
</tbody>
</table>
All you need to know about Microsoft Nano Server

Managing Nano Server .......................................................... 42
The Nano Server Recovery Console ........................................ 42
Figuring out the IP address .................................................... 47
PowerShell Remoting/PowerShell Direct .................................. 47
Windows PowerShell CIM sessions over WinRM ....................... 50
Windows Remote Management ............................................... 52
Working with remote consoles .............................................. 52
Server Management Tools .................................................... 57
Deploying apps on Nano Server ............................................ 59
Appendix A: Additional information on packages ..................... 60
DNS Package ................................................................. 60
IIS Server ........................................................................ 61
Hyper-V ........................................................................... 66
Failover Clustering .............................................................. 66
Appendix B: Setup & Boot eventing, Kernel Debugging & Emergency Management Services ................................. 68
Setup & Boot Eventing .......................................................... 68
Kernel debugging ................................................................. 69
Emergency Management Services .......................................... 70
Appendix C: Unattend, SetupComplete and DISM ..................... 71
DISM ............................................................................... 71
Unattend File ..................................................................... 72
Using setupcomplete.cmd ..................................................... 74
Appendix D: Using MPIO on Nano Server ............................... 76
Appendix E: Using Windows Update ......................................... 77
Appendix F: More on PowerShell on Nano Server ..................... 81
Appendix G: Installing roles and features online ....................... 83
Appendix H: Connecting to a DFS host ................................... 87
Appendix I: Performance & Event monitoring .......................... 88
About the Author ................................................................ 89
About Veeam Software ....................................................... 89
Some history

The best way to explain Nano Server is to dive into the history of Windows Server and how it has evolved over the years.

Windows NT to Windows Server 2003

If you look at Windows NT until Windows Server 2003, it is a full-blown server system with a graphical user interface (GUI) and lots of functionality right out of the box. In addition, it allows you to deploy roles and features like Active Directory, DNS, DHCP, clustering and much more. At the time, you only had one server installation and that was the full-blown server.

Windows Server 2008 and R2

With Windows Server 2008 and R2, a new server installation option came to life. It was called Core Server. Core Server is still a full-blown Windows Server that allows you to run almost every application, role or workload on top of it, but all of the GUI elements are removed, including MMCs and more. At first sight, it was a very big deal because it lowered the attack footprint, patching and resource usage. However, it never gained the attraction of the IT world that it deserved. The reasons for that are simple:

• You couldn't switch between core and the full GUI. The option was chosen at the installation, and reinstalling it was the only way to switch

• It was very difficult to configure. Everything had to be done through command line. PowerShell wasn't fully supported (and many modules for technologies didn't exist yet or were lacking) and remote management through MMCs or that version of the Server Manager also wasn't really that good

As a result, the adoption ranged from very limited to nonexistent.

Windows Server 2012 and R2

Microsoft listened very well to the feedback and concerns about Core Server and brought many enhancements to Windows Server 2012 and R2. To start with, there is now one base Windows Server, which is the core version. And on top of that, you can deploy features and roles. But you can also deploy the minimal server interface on it, which, again, is a role. The minimal server interface gave you some limited access to management tools, but not the full-blown GUI tools. This in-the-middle solution seemed like a good idea, but it wasn't deployed a lot. Still, there are many administrators that choose for the full-blown solution although adoption of the core model picked up some fans. Certainly in high-density environments, or in environments where resources are very valuable, this model has become more and more popular.

But what if you could even go further and look at an even smaller operating system option with multiple benefits that will allow you to start deploying application workloads exactly as they happen in the cloud? As you can imagine, this requires a shift in thinking and operations. Don't be afraid at this point, just continue reading to learn what you need to know.
What is Nano Server?

Nano Server is a headless, 64-bit-only deployment option for Windows Server 2016. Microsoft created this component specifically with key scenarios in mind: Nano Server was created to serve as cloud fabric and infrastructure (Hyper-V host, clustering, networking, storage) and as a deployment option for applications that are — as they call it — born in the cloud (Platform as a Service v2 and ASP.NET v5 applications).

It’s important to know that this deployment option is really headless. The moment you decide to deploy a supported role (such as Hyper-V), you need to realize that this lives outside of Nano Server. There are absolutely no binaries or metadata inside this server. Even the drivers come as a package that you can install on top of it. This makes this server option perfect for those that want to deploy only what they need and keep everything else as minimalistic as possible.

Nano Server is ideal for some key scenarios in your environment such as:

- Hyper-V host
- Storage host for Scale-Out File Servers
- DNS Server
- Web server (IIS)
- A host for applications specifically designed for this
- Container host

And this is just the beginning. I can’t predict the future, but imagine if I can deploy Nano Servers for my specific core infrastructure such as Active Directory, DHCP, DNS (which already works today) and so on. This would save me a lot of management trouble and resources.

Why?

When you read further, you will see that Nano Server will give you some challenges in the way you deploy, configure and manage it. But before you do that and decide it’s not worth the trouble, let’s explore the advantages of deploying Nano Server in your environment.

Reboots

Reboots for a Windows Server are something we learned to live with for many years. IT administrators all around the world are familiar with Patch Tuesday. However, in many cases, it creates many headaches. While you will never be able to remove patches and updates, a headless solution will definitely require less patches.

Less patches don’t mean less security. Because there are less components to patch, the attack vector also decreases. Many attacks are done on components that simply do not exist anymore in the server core, especially when you come to think of GUI elements. Microsoft did some research in 2014 to list the differences:
Figure 1 shows the number of important bulletins, critical bulletins and reboots required in 2014 for Nano Server, Server Core and the Full Server option. As you can see, there are quite a bit of differences. In addition, the maintenance work that needs to be done on Nano Server is lower than for the other options. It should also show that when you can’t choose Nano Server, Server Core is your best option.

**Server images**

Further in this document, you will notice that we will create small images of servers. Those of you who need to manage server image libraries will welcome Nano Server because it will use less disk space. Again, Microsoft provided us with numbers, as you can see below:

Figure 2 shows the differences in setup time, disk footprint and VHD(x) size between Server Core and Nano Server. You can imagine that these numbers are even higher on the Full Server flavor.
Resources

Finally, you will also notice a huge difference in resource consumption. There are fewer processes running, and the boot IO and kernel MB in use are all lower on Nano Server, as you can see in Figure 3:

![Figure 3: Resource Utilization ©Microsoft](image)

Figure 4 shows the differences in the amount of drivers loaded, services running and ports open between Server Core and Nano Server:

![Figure 4: Drivers, services & ports ©Microsoft](image)

Many of you may think that the last figure should fall under security and not resources and you are not wrong. Microsoft does exactly the same. However, less services and drivers loaded also means that less resources are necessary, so I could argue that the last figure should fall under both security and resources.
Conclusion

While the above numbers are certainly not final and are only from the measurement Microsoft came up with in 2014, you should be able to see that using Nano Server can deliver some huge improvements and advantages. And if using Nano Server is not possible, then Server Core should be your choice. I personally expect these numbers to improve even further. Although the way you manage Nano Server will be different than how you have always managed Windows Server operating systems, the advantages are higher than the disadvantages. In the end, you will notice that managing Nano Server certainly isn't that difficult and it is just a matter of getting used to it.

Setting the expectations

Starting to work with Nano Server today is, in my humble opinion, a must. If you are an IT professional, you need to learn how it works, how it needs to be managed (it requires a mental change) and how to troubleshoot it. The faster this happens, the faster your business will benefit.

Everything in this paper is performed on the GA version of Windows Server 2016, so if you want to follow this guide, you will need to work with the GA version.
Let’s look at the quick installation

When you start, you have to realize that Nano Server is not an installation option. It cannot be selected during setup as you are used to with other versions of Windows Server. The binaries are found on the installation media but you need to create a “base-image” first before you can deploy it. There are many ways to create that image, so let’s start here with a quick way to get things going.

A virtual Nano Server

Copy the files **NanoServerImageGenerator** folder from the windows server 2016 ISO (or installation media) to a location of your choice. In my case, this will be under F:\NanoServer\...

Open Windows PowerShell, Windows PowerShell ISE (or the tool of your choice) as an administrator and navigate to the folder where you have copied your scripts.
Before you can start, you’ll need to import the **NanoServerImageGenerator** module. Browse to the directory where you placed the files and then import the module.

The command to do this is:

```powershell
Import-Module ..\NanoServerImageGenerator -Verbose
```

**Note:** The “verbose” is not necessary, but it will show you all the work that is done while importing this module. As you can see, three functions have been imported. We will go into detail about those later.

Now you can create the first VHD that you can use in a virtual environment. Don’t worry about the different parameters yet; I will dive deep into those next. What you are doing here is creating a simple VHD, setting an administrator password and including Hyper-V guest drivers.

```powershell
New-NanoServerImage -Edition Datacenter -DeploymentType Guest -MediaPath <path to root of media> -BasePath ..\Base -TargetPath ..\NanoServerVMs\NanoServerVM.vhd -ComputerName <computer name>
```

- **Edition** can be a standard or a datacenter build (for more information on editions: [https://www.microsoft.com/en-us/cloud-platform/windows-server-pricing](https://www.microsoft.com/en-us/cloud-platform/windows-server-pricing))
- **DeploymentType** (guest or host) defines whether it will be virtual or physical
- **Mediapath** is the path where the contents of the ISO are (either copied to disk or mounted as ISO)
- **BasePath** is the folder where the Nano Server WIM and packages will be placed
- **TargetPath** will be the folder where the VHD (or VHDX) will be created
- **ComputerName** is where you define the name of the computer

We will go further into details about those parameters later on.

In my case, this becomes:

```powershell
New-NanoServerImage -Edition Datacenter -DeploymentType Guest -MediaPath D:\ -BasePath ..\Base -TargetPath ..\NanoServerVMs\Nano01\Nano01.vhdx -ComputerName Nano01
```
Figure 7: Creating your first image, with password request

After entering the password, the system will start creating the VHD.

After creation, the only thing left to do is to create a VM in Hyper-V and attach the VHD(X) to it. This is how:

Open Hyper-V Manager and select New > Virtual Machine
Press **Next** on the **Before You Begin** page

![Figure 9: Before you Begin](image)

On the **Specify Name and Location** page, choose the name of the VM (this is not the computer name, but rather, the name that will be visible in Hyper-V Manager) and location where you want to store it and press **Next**.

![Figure 10: Specify Name and Location](image)
On the **Specify Generation** page, choose the generation. If you chose to deploy a VHD, then choose **Generation 1**. If you want to deploy a VHDX, then choose **Generation 2**. Press **Next**

![Specify Generation](image1)

**Figure 11: Specify Generation**

On the **Assign Memory** page, adjust the startup memory and decide whether you want to use **Dynamic Memory** or **Static Memory**. Press **Next**

![Assign Memory](image2)

**Figure 12: Assign Memory**
On the **Configure Networking** page, select the network that you want to connect to and press **Next**.

![Figure 13: Configure Networking](image)

On the **Connect Virtual Hard Disk** page, select *Use an existing virtual hard disk*, point to the created VHD(X) and press **Next**.

![Figure 14: Connect Virtual Hard Disk](image)
On the **Complete** page, review your settings and press **Finish**.

![Figure 15: Completing the New Virtual Machine Wizard](image)

This is, of course, completely possible with PowerShell. I will show you an example of how to do this in the next chapter, *Advanced installation/Deployment options*.

**A physical Nano Server**

While most of the examples in this paper will be virtual Nano Servers, some of them will certainly run on physical servers. When you use Nano Server as a Hyper-V node (either clustered or not), it is necessary for it to be a physical server. For the purpose of this paper, I actually run nested Hyper-V so I can simulate this behavior. However, in a real world environment, this probably won't be the case.

The procedure to create a VHD for a physical machine is more or less the same as it is for a VM. I used the following command:

```powershell
New-NanoServerImage -Edition Datacenter -DeploymentType Host -BasePath .\Base -TargetPath .\NanoServerVMS\NanoPhys01\NanoPhys01.vhd -ComputerName NanoPhys01 -OEMDrivers -Compute -Clustering
```

The difference between this command and the command for the VM is that I already added some roles — packages such as compute and clustering — and also added the **OEMDrivers** package to include drivers. In the next chapter, I’ll talk about these packages in more detail.
Another important note is that I actually didn't need to specify the — MediaPath anymore because I already had a base image created. After this is done this once, I can keep using this base until there are updates to the media.

![Figure 16: Creating a physical Nano Server](image)

Another difference is that I am not going to connect the VHD to a VM. Instead, I will deploy that VHD onto a physical server.

If that physical server already has an operating system, I can simply copy the VHD to that server, mount the VHD and then run `bcdboot d:\windows`. Finally, unmount the VHD and restart the server. I should be able to boot into the Nano Server VHD.

Of course, if you prefer to dual boot, you can always use `bcdboot` to add a boot listing, but this probably won't happen very often. In most cases, you want to deploy it on a clean, empty server.

In that case, I suggest you read the following two blog posts that explain other methods you can use (and may prefer). Again, because this is not the scope of this paper, and most people will have their own deployment preferences, I won't dive deeper into here.

- Deploying Nano Server to a Bare-Metal Machine using a WIM and WinPE
- How to use WDS to PxE Boot a Nano Server VHD

## Packages

The previous steps (in the A virtual Nano Server section) will give you a running VM. Unfortunately, this VM only has a name and administrator login. The VM is not domain joined, and it may require a static IP address, special firewall settings and many other changes required by your specific environment.
It is also completely headless at this point in time, meaning no packages are installed; so you have a running operating system that isn’t performing any functions at all. Before we dive into more advanced deployments, let’s take a look at the available packages. During the timeframe of the different technical previews, we noticed that additional packages became available in between previews. We expect this pattern to continue because it won’t require a new version of the OS, but simply a new (or updated) package that can be downloaded. If you have packages that you would like to see included with Nano Server, you can go to Microsoft’s UserVoice initiative and vote for your preferred package.

The size of the core OS that we deployed is only around 522 MB. It will grow slightly by adding packages, but not by much. However, as previously mentioned, this is the core OS only and doesn’t perform anything yet.

In the **Role/Feature** column in **Table 1: Roles and Features**, you can see the various roles or features and the option that you will need to use to deploy those through the script.

<table>
<thead>
<tr>
<th>Role/Feature</th>
<th>Parameter for the script</th>
<th>Filename on ISO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyper-V</td>
<td>Compute</td>
<td>Microsoft-NanoServer-Compute-Package</td>
</tr>
<tr>
<td>Failover Clustering</td>
<td>Clustering</td>
<td>Microsoft-NanoServer-FailoverCluster-Package</td>
</tr>
<tr>
<td>Basic drivers (physical deployment) — these drivers are the same as the drivers included in the core version of the OS</td>
<td>OEMDrivers</td>
<td>Microsoft-NanoServer-OEM-Drivers-Package</td>
</tr>
<tr>
<td>File Server role and other storage components</td>
<td>Storage</td>
<td>Microsoft-NanoServer-Storage-Package</td>
</tr>
<tr>
<td>Windows Defender (including a default signature file)</td>
<td>Defender</td>
<td>Microsoft-NanoServer-Defender-Package</td>
</tr>
<tr>
<td>Reverse forwarders for application compatibility for application frameworks such as Ruby, Node.js etc….</td>
<td>N/A</td>
<td>Included by default!</td>
</tr>
<tr>
<td>DNS Server role</td>
<td>Packages Microsoft-NanoServer-DNS-Package</td>
<td>Microsoft-NanoServer-DNS-Package</td>
</tr>
<tr>
<td>Desired State Configuration (DSC)</td>
<td>Packages Microsoft-NanoServer-DSC-Package</td>
<td>Microsoft-NanoServer-DSC-Package</td>
</tr>
</tbody>
</table>
Internet Information Server (IIS) | Packages Microsoft-NanoServer-IIS-Package | Microsoft-NanoServer-IIS-Package
---|---|---
Host support for Windows Containers | Containers | Microsoft-NanoServer-Containers-Package
Network Performance Diagnostics Service (NPDS) | Packages Microsoft-NanoServer-NPDS-Package | Microsoft-NanoServer-NPDS-Package
Data Center Bridging | Packages Microsoft-NanoServer-DCB-Package | Microsoft-NanoServer-DCB-Package
Deploying on a virtual machine | DeploymentType Guest | Microsoft-NanoServer-Guest-Package
Deploying on a physical machine | DeploymentType Host | Microsoft-NanoServer-Host-Package
Secure startup | Packages Microsoft-NanoServer-SecureStartup-Package | Microsoft-NanoServer-SecureStartup-Package
Shielded VM | Packages Microsoft-NanpServer-ShieldedVM-Package | Microsoft-NanpServer-ShieldedVM-Package

| Table 1: Roles and Features |

The **Hyper-V** role seems clear enough. With a minimum installation and all the benefits described in the Why? Section, Nano Server should be the option of choice for each deployment. However, there are still a few differences between Hyper-V running on Nano Server and any other installation option. More information can be found in Appendix A of the **Hyper-V** section.

**Clustering services** is also a straightforward role. In my opinion, it doesn’t matter whether this is for Hyper-V or file servers. As long as it is servicing the customers and you can do all the management and monitoring remotely, all the resources you can take away from the OS and give to the application services itself are benefits. You can find more information about the clustering package under Appendix A of the **Failover Clustering** section.

The **OEM drivers** (or basic drivers) provide the same set of drivers that are also in the core version of the operating system. While it is possible to add other drivers (which I will explain later), you can easily add defaults just by adding this package.

The **Storage** package will add the necessary file server services and other storage components. Think about the necessary solutions for connecting to storage such as MPIO or deploying scale-out file servers.
The **Windows Defender** package will add Windows Defender, including a default signature file. Of course, this will need to be updated if you want this. In my opinion, and because most of my host servers are not connected in any way to the internet or any other network where there is a risk for infection, I actually think it is great that you can decide whether or not you are going to deploy Defender.

The **DNS server** package will add the DNS server on top of Nano Server. However, there are some catches, which I will cover in the DNS Package chapter.

The **Desired State Configuration** package adds the necessary components to work with this technology. Desired State Configuration is out of scope for this book, but you can find all the information you need here: [https://msdn.microsoft.com/powershell/dsc/nanodsc](https://msdn.microsoft.com/powershell/dsc/nanodsc)

The **IIS package** will add the necessary binaries to run IIS and certain features of that service. However, not all features are already supported. For more information, look at the **IIS Server** chapter under **Appendix A: Additional information on packages**.

The **Containers** package will add host container support into the Nano Server. At this point, the deployed Nano Server will be able to run containerized applications.

The **System Center Virtual Machine** agent actually comes in two flavors and can be a bit confusing. There is a little catch: You don’t need to use the — **Compute** option when you want to create a Hyper-V host with the SCVMM agent. Instead, you need to use — **Packages Microsoft-NanoServer-Compute-Package, Microsoft-Windows-Server-SCVMM-Compute-Package**

If you do not include the -SCVMM package at the time of creation, but rather at the host or cluster to VMM at a later point, the VMM agent will automatically deploy then. Note that this package requires the Windows Defender package first!

The **Network Performance Diagnostics Service (NPDS)** package enables a new feature that comes more or less straight out of the Azure world and builds on top of the network discovery.

The **Data Center Bridging** package is a set of standards to enable converged fabrics in a data center to allow storage, networking, clustering and the like, so traffic can all share the same network infrastructure. Nano Server also supports this, just like the other server options.

The **Deploying on a virtual machine** package tells your deployment you want to use it as a virtual machine and automatically deploys the Hyper-V guest drivers.

The **Deploying on a physical machine** package gives you support for bare metal deployments.

The **Secure startup** package provides support for secure startup.

The **Shielded VM** package is the host guardian that provides everything necessary to provision shielded virtual machines. Note that this package is only available in the datacenter edition of Nano Server.
Finally, even though you don’t see it in the packages, you can also monitor Nano Server with Microsoft System Center Operations Manager. System Center includes an agent that you can deploy onto Nano Server. Follow the instructions to do this here: https://technet.microsoft.com/en-us/system-center-docs/om/manage/install-agent-on-nano-server#start-monitoring-your-nano-server

As you can see, you can already install quite a few roles and features on Nano Server, and I certainly believe more roles and features will be coming in the future.

Some additional roles and features that don’t come in a package, but are certainly important, are MPIO, NIC teaming and using SSH provider.

## Advanced installation/deployment options

As you may recall, I used the —Verbose parameter when I imported the module NanoServerImageGenerator. We saw in the output that it added three cmdlets or scripts.

Let’s look at all the parameters that exist with the scripts that come with Nano Server and what you can customize.

*Note: You can only use these scripts on a Windows 8.1, Windows 10, Windows Server 2012 R2 or Windows Server 2016 installations. Other OS types are not supported.*

### New-NanoServerImage

This function allows you to create a new Nano Server image. In the quick installation, we only briefly touched on the potential of what we can do, so let’s dive a bit deeper into all the parameters that exist.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdministratorPassword</td>
<td>Securestring</td>
<td>This sets the image’s administrator password. If you do not specify this on the command line, you will be interactively prompted to do.</td>
</tr>
<tr>
<td>BasePath</td>
<td>String</td>
<td>This is the location for the copy of the source media. It will be under a folder (auto-created if not existing). It contains the packages, tools, hard disk image and WIM file.</td>
</tr>
<tr>
<td>Clustering</td>
<td>N/A</td>
<td>This allows you to add the clustering role.</td>
</tr>
<tr>
<td>Compute</td>
<td>N/A</td>
<td>This allows you to add the Compute (Hyper-V) role.</td>
</tr>
<tr>
<td>ComputerName</td>
<td>String</td>
<td>This sets the computer name of the image. Note: The computer name can be no longer than 15 characters.</td>
</tr>
<tr>
<td>Containers</td>
<td>N/A</td>
<td>This allows you to add the Containers role.</td>
</tr>
<tr>
<td>CopyPath</td>
<td>String</td>
<td>This parameter specifies the additional directory path on the computer where you create the image. This directory and the files in it will be added to the root of the VHD(X).</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DebugMethod</td>
<td>String</td>
<td>With this parameter, you will enable kernel debugging on the target image with the specified method. The values can be Serial, Net, 1394 or USB. Depending on the value of this parameter, other parameters may become available. See kernel debugging for more information.</td>
</tr>
<tr>
<td>Defender</td>
<td>N/A</td>
<td>This adds the Windows Defender feature.</td>
</tr>
<tr>
<td>DeploymentType</td>
<td>String</td>
<td>This will be Guest or host, depending on whether you want to deploy a virtual or physical Nano Server.</td>
</tr>
<tr>
<td>Development</td>
<td>N/A</td>
<td>This is used to test on Nano Server, which allows unsigned drivers, copy debugger binaries and more.</td>
</tr>
<tr>
<td>DomainBlobPath</td>
<td>String</td>
<td>This lets you Join the image to the domain as specified in the given domain blob. For more information, see the domain join chapter.</td>
</tr>
<tr>
<td>DomainName</td>
<td>String</td>
<td>This joins the image to the specified domain performing an offline join. For more information, see the domain join chapter.</td>
</tr>
<tr>
<td>DriversPath</td>
<td>String</td>
<td>If you need additional drivers or specific drivers, instead of the OEM drivers, you can add them with this parameter. It should point to the path containing the drivers (.inf and binaries). Note: The drivers need to be signed; otherwise, the command will fail.</td>
</tr>
<tr>
<td>EMSBaudRate</td>
<td>UInt32</td>
<td>This is the baud rate to use for EMS. The default is 115200bps.</td>
</tr>
<tr>
<td>EMSPort</td>
<td>Byte</td>
<td>This is the port on which to enable the EMS. The default is 1.</td>
</tr>
<tr>
<td>Edition</td>
<td>String</td>
<td>Standard or Datacenter, this is the Windows edition you want to deploy.</td>
</tr>
<tr>
<td>EnableEMS</td>
<td>N/A</td>
<td>This enables EMS (Emergency Management Services) and BootEMS on the image. See Emergency Management Services for more information.</td>
</tr>
<tr>
<td>EnableRemoteManagementPort</td>
<td>N/A</td>
<td>This parameter opens port 5985 for inbound TCP connections for Windows Remote Management (WinRM). See Windows Remote Management for more information.</td>
</tr>
<tr>
<td>InterfaceNameOrIndex</td>
<td>String</td>
<td>If you want to change the IP settings of an adapter, you'll need to use this parameter in conjunction with the below IP parameters. You can retrieve these using Get-NetAdapter, netsh or EMC if you already created an image, and in a VM, the first will always be named Ethernet.</td>
</tr>
<tr>
<td>Internal</td>
<td>String</td>
<td>This is something specific for MSFT, and not to be used in your scripts.</td>
</tr>
<tr>
<td>Ipv4Address</td>
<td>String</td>
<td>This sets the given IPv4 static address on the interface specified by InterfaceNameOrIndex.</td>
</tr>
<tr>
<td>IPv4Dns</td>
<td>String</td>
<td>This sets the given IPv4 DNS Server (can be multiple DNS Servers) on the interface specified by InterfaceNameOrIndex.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ipv4Gateway</td>
<td>String</td>
<td>This sets the given IPv4 gateway on the interface specified by InterfaceNameOrIndex.</td>
</tr>
<tr>
<td>Ipv4SubnetMask</td>
<td>String</td>
<td>This sets the given IPv4 subnet mask on the interface specified by InterfaceNameOrIndex.</td>
</tr>
<tr>
<td>Ipv6Address</td>
<td>String</td>
<td>This sets the given IPv6 static address on the interface specified by InterfaceNameOrIndex.</td>
</tr>
<tr>
<td>Ipv6Dns</td>
<td>String</td>
<td>This sets the given IPv6 DNS Server on the interface specified by InterfaceNameOrIndex.</td>
</tr>
<tr>
<td>LogPath</td>
<td>String</td>
<td>This sets the location where you want to save the logs created while building the image.</td>
</tr>
<tr>
<td>MaxSize</td>
<td>Uint64</td>
<td>Size in bytes of the dynamic VHD(X) to be created. Default is 4 GB.</td>
</tr>
<tr>
<td>MediaPath</td>
<td>String</td>
<td>The location of the source media. If a local copy of the source media already exists, and it is specified as the base path, then no copying is performed. This is the downloaded ISO that you have either mounted or copied to a specific location. This is only necessary the first time.</td>
</tr>
<tr>
<td>OEMDrivers</td>
<td>N/A</td>
<td>This is used to add the OEM Drivers package. Those drivers are the same set of drivers that exist in Server Core.</td>
</tr>
<tr>
<td>OfflineScriptArgument</td>
<td>Hashtable</td>
<td>This accepts an hashtable of arguments when you want to add a customer script (see next).</td>
</tr>
<tr>
<td>OfflineScriptPath</td>
<td>String</td>
<td>This accepts an array of paths to PowerShell scripts, with the possibility to pass arguments through the OfflineScriptArgument.</td>
</tr>
<tr>
<td>Package</td>
<td>String</td>
<td>This is the specific parameter that adds the packages that don't have a specific parameter. See the Packages chapter for more information.</td>
</tr>
<tr>
<td>ReuseDomainNode</td>
<td>N/A</td>
<td>When joining a domain, reuse a node with the same name if it exists. For more information, see the domain join chapter.</td>
</tr>
<tr>
<td>ServicingPackagePath</td>
<td>String</td>
<td>With this parameter, you can add servicing packages (multiple packages are possible) that you download from the Microsoft Update catalog.</td>
</tr>
<tr>
<td>SetupComplete-Command</td>
<td>String</td>
<td>Here, you can add custom commands as part of setupcomplete.cmd.</td>
</tr>
<tr>
<td>SetupUI</td>
<td>String</td>
<td>Is about same way as with the packages option, this is used internally by the Nano Server Image Builder (see below), and the recommended method is still to use the -packages parameter.</td>
</tr>
<tr>
<td>Storage</td>
<td>N/A</td>
<td>This adds the Storage role.</td>
</tr>
<tr>
<td>TargetPath</td>
<td>String</td>
<td>This is the location of the final, modified image. The image format is determined based on the file extension. Possible extension values are: .VHD and .VHDX. VHD will come with MBR and VHDX with GPT disk layout.</td>
</tr>
<tr>
<td>UnattendPath</td>
<td>String</td>
<td>The location to add your own, custom, unattend.xml file.</td>
</tr>
</tbody>
</table>
Example: I want to create a Nano Server with the clustering and compute packages (I want to build a Hyper-V cluster). In addition, it needs to be automatically joined to the domain MR.local, because this is the domain where I am working (I will explore this topic further in the Domain join options). Remote Management is enabled, Guest drivers are installed and a static IPv4 address is attached to it. I also want my VHDX (it will be a Generation 2 machine) defined with a maximum of 100 GB (yet it will be smaller because it is dynamically expanding). I’ll add my local scripts directory (D:\Scripts) into the C: volume of this Nano Server. This is how the PowerShell line will look:

```powershell
New-NanoServerImage -DeploymentType Guest -Edition Datacenter -TargetPath .\NanoServerVMs\Nano03\Nano03.vhdx -BasePath .\Base -Clustering -Compute -ComputerName Nano03 -CopyPath F:\Scripts -DomainName MR.local -EnableRemoteManagementPort -InterfaceNameOrIndex ethernet -Ipv4Address 192.168.1.173 -Ipv4Dns 192.168.1.151 -Ipv4Gateway 192.168.1.1 -Ipv4SubnetMask 255.255.255.0 -MaxSize 100GB
```

Figure 18: Advanced creation of a Nano Server

Again, I want to add this as a VM to my environment, but as promised, I will not go through the wizard again. I’ll use PowerShell to do the job instead. To do this, I’ll create a new VM, attach the created VHDX to it, give it a startup memory of 256 MB and attach it to the LAN virtual network.

```powershell
New-VM -VHDPath .\NanoServerVMs\Nano03\Nano03.vhdx -Generation 2 -MemoryStartupBytes 268435456 -Name Nano03 -Path F:\VM -SwitchName LAN
```
Get-NanoServerPackages

This script won’t let you do some things but it makes it very easy to see what packages are included in the base image or on the media. We talked about all of these packages before, and if you want to figure out which ones are included (with or without different languages), this cmdlet will give you all the information you need.

It comes with three optional parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BasePath</td>
<td>Here you can add the location of where the base files are. The script will look at that location and output the packages it has found.</td>
</tr>
<tr>
<td>MediaPath</td>
<td>Here you can add the location of where the media is (mounted ISO as example). Again, the script will look at that location and output the packages it has found.</td>
</tr>
<tr>
<td>LogPath</td>
<td></td>
</tr>
</tbody>
</table>
Below is a simple example of how I checked the packages that included on my base image:

```
Get-NanoServerPackage -BasePath .\Base
```

![Figure 20: Packages available on my base image](image)

As you can see, I have the available packages I described before.

**Edit-NanoServerImage**

If you already created a Nano Server, but want to add additional packages, change the IP address or make any other customizations, you don’t need to rebuild an image or simply create a new one. You can modify a created image afterwards. Before TP4, this had to be done with DISM. For those of you who prefer this method, this is still possible. It is also done with the cmdlet **Edit-NanoServerImage**

A few notes on this script:

- The Nano Server needs to be turned off.
- You cannot change VHD to VHDX with this script.
- After the first boot of that Nano Server, certain things cannot be changed anymore such as all IP settings, setupcompletecommands and the parameter -Development.

Most parameters described in the **New-NanoServerImage** cmdlet are the same, but some can’t be used.

There are four parameters that cannot be used when editing a Nano Server:

- DeploymentType
- Edition
- MaxSize
- MediaPath
Please note that you will need to have used the parameters already in the **New-NanoServerImage**. Otherwise, you can't use them in the **Edit-NanoServerImage** cmdlet.

```powershell
Edit-NanoServerImage -BasePath .\Base -TargetPath .\NanoServerVMS\Nano01\Nano01.vhdx
-EnableRemoteManagementPort -InterfaceNameOrIndex Ethernet -Ipv4Address 192.168.1.171
```

![Figure 21: Script is running and modifying the Nano Server](image)

After running this script and starting the Nano Server again, I can check to see is everything worked.

### Additional information for advanced installation/deployment

Before continuing our Nano Server mission, there are a few things I need to mention. Let's take a deeper look at the domain join possibilities and the specific Azure parameter you can use.

#### Domain join options

There are many ways to join your Nano Server to an existing domain. I prefer the first option described below, but this will not always be the best option. Let's look at the five different methods you can use.

**Joining the to the domain where the local computer resides**

This is my favorite option because it is so easy. If the workstation or server where you are building an image is joined to the same domain that you want to join to the Nano Server, you only need to use the appropriate switch within the PowerShell script, and the rest will be done automatically for you.
You might recall the example that we used in the New-NanoServerImage chapter:

```powershell
New-NanoServerImage -DeploymentType Guest -Edition Datacenter -TargetPath .\NanoServerVMs\Nano03\Nano03.vhdx -BasePath .\Base -Clustering -Compute -ComputerName Nano03 -CopyPath F:\Scripts -DomainName MR.local -EnableRemoteManagementPort -InterfaceNameOrIndex ethernet -Ipv4Address 192.168.1.173 -Ipv4Dns 192.168.1.151 -Ipv4Gateway 192.168.1.1 -Ipv4SubnetMask 255.255.255.0 -MaxSize 100GB
```

One of the switches we use is — **DomainName** and the name of my domain. This is all you need to do to get the computer joined to the domain.

An offline domain join happens in the background, and the script will automatically harvest a domain blob for the specific domain and use that to do the domain join. Of course, you need to have the administrative rights to perform a domain join when you perform this action. So be sure that the user you are running the PowerShell cmdlets with has those rights.

If you want to know exactly what happens in the background, just read further. The next methods will involve more manual work that you need to do and this is what actually happens.

**Joining a different domain**

Here, if the workstation or server you are working on is not joined to a domain, or is possibly joined to another domain, you can't work with the fully automated parameter. However, it still isn't that difficult of a process. What you need to do is harvest a domain blob. Remember to keep a few things in mind:

The blob must be manually harvested on a computer that is joined to the specific domain.

You need to have the rights to perform this action.

This is the command you need to use when harvesting a blob (Note: This is an elevated prompt):

```powershell
djoin /Provision /Domain Contoso /Machine serverName /SaveFile serverName.djoin
```

Copy that blob to a place on your computer where you are building the image and use the switch — **DomainBlobPath** to perform the action. Example:

```powershell
New-NanoServerImage -DeploymentType Guest -Edition Datacenter -MediaPath \Path\To\Media\en_us -BasePath .\Base -TargetPath .\ServerName.vhd -DomainBlobPath .\Path\ServerName.djoin
```

**Note:** You don't add the parameter name — **ComputerName** anymore because the blob already contains the server name. It will throw an error if you try to do so.
Reusing a domain account

Here, you can use the third method when you want to reuse a domain account. If the server name already exists in Active Directory, you can use simple the parameter — **ReuseDomainNode** in your script.

Joining Nano Server to a domain online

All of the previous methods are offline joins when the image is created. But what happens when your server is already created and running? Can you still join it to a domain? Luckily, the answer is yes. Performing an online join to a domain is possible, but it requires a bit of work. Let's go over the procedure.

Harvesting a blob

Just like before, you will need to harvest a blob and use that to join the server online to a domain.

**Note:** The computer that is in that domain and the location from which you will harvest the blob needs to be running Windows 10 or Windows Server Threshold! The domain controller does not need to be running them.

```
djoin.exe /provision /domain <domain-name> /machine <machine-name> /savefile .\objblob
```

You will have a file called objblob now. The next step is to copy that file to the Nano Server. You can do it easily by mapping the administrative C$ share to a drive:

```
net use z: \<ip address of Nano Server>\c$
md z:\Temp
copy objblob z:\Temp
```

**Note:** It might be that you receive an error doing this. That probably means the firewall on the Nano Server is blocking the request. In this case, run the following commands first:

```
Set-Item WSMon:\localhost\Client\TrustedHosts "<IP address of Nano Server>"
$ip - "<ip address of Nano Server>"
Enter-PSSession -ComputerName $ip -Credential $ip\Administrator
netsh advfirewall firewall set rule group="File and Printer Sharing" new enable=yes
Exit-PSSession
```

**Note:** You can find more information on this procedure in the PowerShell Remoting/PowerShell Direct chapter.

Now, you need to join the Nano Server to the domain. Again, for more information about **Enter-PSSession** or if the command fails, refer to the PowerShell Remoting/PowerShell Direct chapter.

```
Enter-PSSession -ComputerName $ip -Credential $ip\Administrator
djoin /requestodj /loadfile c:\Temp\objblob /windowspath c:\windows /localos
shutdown /r /t 5
Exit-PSSession
```
The procedure works like this:

- Connect remotely to the server and start a remote PowerShell session.
- Use djoin to load the configuration out of the created blob file and join that server to the domain (For more information on djoin, go to https://technet.microsoft.com/en-us/library/ offline-domain-join-djoin-step-by-step(v=ws.10).aspx).
- Instruct the computer to restart (the parameter /r does this in the shutdown command) and wait five seconds.
- Those five seconds will give you the time necessary to exit the remote PowerShell session (Exit-PSSession).

And that’s it! After you reboot, your Nano Server will be joined to the domain.

**Using the unattend file**

The last option you have is using an unattend file. You will need to use parts of the procedure above to do this.

First, you will start by harvesting a blob again, just like you did earlier. When you’re finished with that, open the blob file with a text editor (Notepad works fine) and copy the files in the unattend file. See the example below:

```xml
    <settings pass="offlineServicing">
        <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
            <ComputerName>NANO02</ComputerName>
        </component>
        <component name="Microsoft-Windows-UnattendedJoin" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
            <OfflineIdentification>
                ...
            </OfflineIdentification>
        </component>
    </settings>
</unattend>
```

© 2016 Veeam Software
The important part of the domain join is the data residing between the `<AccountData>` and `</AccountData>` part. This is the information you can copy from that blob file that you have harvested. Now, you need to inject the unattend.xml file into the Nano Server image. This can be done manually (see Appendix C: Unattend, SetupComplete and DISM) or through the parameter `<UnattendPath>` included with the `New-NanoServerImage` cmdlet.
Azure

Running Nano in Microsoft Azure is possible. As of today, there are two methods to do so:

Method 1: Deploy Nano Server through the gallery

When you log in to your Microsoft Azure subscription, you can deploy a new VM. It doesn’t matter whether you are using the classic method or the new resource manager deployment. You will need to find this image in the gallery by running a search with the keyword Nano. Then, it’s just a matter of choosing your size, networking and all other options, just as if you are deploying any other regular VM in Microsoft Azure.

![Nano Server build in Microsoft Azure](image)

However, there is a difference. Because you are not building your image with the correct packages, you can’t add the roles upfront. Therefore, you will need to deploy your packages online. More information on that process is in Appendix G: Installing roles and features online.

Method 2: Bring your own server to Azure

It is possible to bring your own VHD and use it as a server in Microsoft Azure. Simply use the parameters `-DeploymentType Guest` and `-EnableRemoteManagement` to prepare a VHD for Azure.

Using the Nano Server Image Builder or Show-command

If you are not keen on using PowerShell all the time, or you simply want something easier to start with, then you have two options. The first option is a graphical UI that can be installed on the machine where you build your images, the second comes with PowerShell 5 and allows you to more easily create PowerShell lines. Let’s have a look at both.

Nano Server Image Builder

The Nano Server Image Builder is a graphical UI (a wizard) that allows you to create Nano Server images. Before you can use it, you need to install two components on your workstation. The Windows ADK and the Nano Server Image Builder itself. After the installation of both components, you can run the wizard. Before you do this, make sure that your workstation has access to the ISO/DVD of Windows Server 2016 or a network share that holds those installation bits.

When you launch the tool, this is what you will see as a first page:

![Nano Server Image Builder wizard](image)

To start, we are going to use the first option in the wizard to create a new Nano Server image:

![Before you begin](image)
Read the information on this page because you'll need (at a minimum) the data (network configuration, location of 2016 binaries, the name of the server and (potentially) the drivers needed to create your Nano Server image. Then, press **Next**.

---

![Figure 25: Select Installation Media](image)

Browse to the installation media of Windows Server 2016. In my case, this is an ISO mounted as the D: drive on my workstation. Then, press **Next**.

---

![Figure 26: License](image)
Review the license agreement, accept it by selecting the checkbox and press **Next**.

![Deployment Type](image1.png)

**Figure 27: Deployment Type**

On the deployment type page, you need to start making some selections.

**Note:** The wizard will change depending on the selections you make here (and even further in the wizard). I’m not going to run through all of the different selections. I am planning to build a Virtual Machine Image, with a maximum disk size extension of 4 GB and I am going to save my machine under `documents\VMs`.

![Deployment Type](image2.png)

**Figure 28: Deployment Type**
When all of the information has been entered, press **Next**.

![Basic Installation start](image)

*Figure 29: Basic Installation start*

Press **Next** on the Basic installation page.

![Packages Selection](image)

*Figure 30: Packages Selection*
On this page, you can select your edition (Datacenter or Standard) and the packages you want to deploy. I have made my selection of packages and then press **Next**.

![Figure 31: Additional drivers](image)

On the Add drivers page you can add specific drivers needed for your environment (.inf format). In my case, because it is a virtual machine, the integration components will be added automatically. In some cases, however, you might need to add specific drivers (a physical server, or sometimes even on a virtual server). Press **Next** when you have added the needed drivers.

![Figure 32: Other details for the server](image)
On the Destination machine information page, add the computer name and the administrator password, plus select your specific time zone. Then, press **Next**.

**Figure 33: Domain information**

On the next page, you can let the machine join a domain or specify a domain blob. Then, press **Next**.

**Figure 34: Network and Management options**
On the Set Network page, you can fill in your network configuration. vLan, enabling WinRM, assigning fixed IPs and more are all possible on this page. After that, press Next.

![Figure 35: Basic or Advanced](image)

Now the wizard will ask you to create a basic Nano Server Image (in other words, it will create this with the settings you already defined), or perform some more advanced configuration options. The advanced configuration options include adding servicing packages, taking advantage of an unattend file, adding specific files into your Nano Server and more. Since we already discussed those settings before, I'm going to end the wizard here because it was just demonstration of what this solution can do. I chose Create basic Nano Server image.

![Figure 36: Review screen](image)
Review your settings one final time and then press the Create button.

**Figure 37: Creating the Nano Server**

The image is being built. Important to notice is that you can copy the PowerShell path from here and see how it what it will look like (and learn from it) even when you are not using this tool. In my example, this became:

```
```

As I mentioned earlier, there is also the option to create a USB drive using WinPE, which will detect your existing Nano Server hardware config and report the details on-screen and in a log file. You certainly will want to do this to detect (upfront) certain things on your physical server. Hardware detection includes:

- Network interface index
- Boot firmware type
- System board information
- Disks and volumes
- Devices without a driver!

More information can be found in this blog: [https://blogs.technet.microsoft.com/nanoserver/2016/10/15/introducing-the-nano-server-image-builder/](https://blogs.technet.microsoft.com/nanoserver/2016/10/15/introducing-the-nano-server-image-builder/)
Show-Command

Starting with PowerShell 5, there is a new cmdlet called Show-Command. This allows you to create PowerShell commands through some sort of window.

For New-NanoServerImage, it becomes this:

```powershell
Show-Command -Name New-NanoServerImage
```

The outcome is this:

![Parameters for New-NanoServerImage](image)

Now, if we fill in some details, we can actually copy the data to our command prompt. After filling in some parameters, I get the following:
By pressing **Copy**, the PowerShell command will be created for you (to paste in scripts or the ISE), and when you press **Run**, it will actually run it for you.

**Managing Nano Server**

There are different methods to manage Nano Server, including methods that permit remote management while having zero IP connectivity to the Nano node. Below, you can see all the options and some ideas on how to figure out the IP server address.

It’s important to know that Nano Server is managed 100% remotely. The only exception is when connectivity is lost and you want to use an emergency solution to fix the connectivity.

**The Nano Server Recovery Console**

The Nano Server Recovery Console can be compared to the iLo of HP or DRAC from Dell, or any other KVM type of console. While you can remotely manage everything, there might be a situation when you misconfigure something by accident and can’t access the Nano Server through remote IP management. In this case, you can use the Recovery Console to fix your issues and get remote connectivity again.
I will use the Nano01 server that I created in the quick installation chapter to show the possibilities. Because this is a running VM, I don’t need to attach a screen and keyboard to it, but I will just use the VMConnect functionality to view the console instead.

Before I actually do that, let me walk you through the navigation methods you will need to use when you want to view information or make changes to the server’s configuration.

- To scroll through the screen, you will need to use the arrow keys
- To jump to an option that you can select, you can use the TAB key. You will notice that it will be highlighted, and then you can use ENTER to select and jump to the next screen.
- The ESC key will get you to a previous screen, and pressing ESC on the first screen will actually log you off.
- On some screens, you will notice that you can do additional things with the function keys. For example, in the network configuration screen of a chosen network adapter, F11 will give you the possibility to modify the IPv4 settings and F12 will do the same for IPv6.

Now, let’s connect to a server and log on.

To start, type in your user, password and (optionally when joined) domain name.

![Login screen of a Nano Server](image)

After you log in, you will get a screen that displays server information. You will be able to see the domain (or workgroup), version of the OS, date and local time. More important, you can use the options below, marked with a >. Select networking (by using the TAB) and press ENTER.
On the networking screen, you will see the different network interfaces (if applicable) and you can select the one you want to review or change by using TAB and ENTER again. Select the appropriate network adapter.

Figure 41: Nano Server Home Screen

Figure 42: Network Settings screen
On this page, you can see all of the details of that network card. Scroll down with the arrow keys to see more information like the network driver. For the remainder of this, I’ll demonstrate a few examples of what you can do.

The next page shows the routing table. As you can see, you can add a route by pressing F10.
You can add or delete routes, and by pressing ESC, you can go back to the previous screen.

The next screen shows you the IPv4 configuration, which I reached by pressing F11 on the network settings page, and the option to change some items.

Besides IPv4 settings, you can also change IPv6 settings. When you go back, you can also see Inbound Firewall Rules, Outbound Firewall Rules and even reset WinRM if you lose the ability to remotely manage the server over WinRM.
I’m often asked if it is possible to create firewall rules through this console. The answer to that question is no. The entire idea of the console is to fix connectivity issues to the Nano Server if something goes wrong. The moment you have restored connectivity, you should use remote management techniques again to create your own firewall rules or do other tasks.

Another note of importance is when you go into this console and your Nano Server is a Hyper-V host. At that point, you will see another option in the start screen called **VM Host**. This option allows you to view the status of the guest VMs running on that host and the status of the external virtual switch.

You can also restart or shutdown the Nano Server through this console.

**Figuring out the IP address**

I’ve already mentioned that you need to manage the Nano Server remotely. One issue could be that you have automation systems in place that deploy Nano Server, or that you use DHCP to give the Nano Server an IP address. In both cases, you might not know the IP address.

So how can you figure out the IP address? There are actually a few methods. Use the one that works best for you:

- **Nano Server Recovery Console** (explained above): By connecting to this console, you will be able to see the IP configuration of that server.

- **Use the Emergency Management Services**. For more information, see the *Emergency Management Services* chapter.

- **Simply ping the server name** if your DNS system is configured correctly.

- **Use PowerShell Direct** (if the Nano Server is a VM). This is described the next chapter, *PowerShell*.

I’m sure you can find other ways to figure out the IP address, but the above are the most commonly used.

**PowerShell Remoting/PowerShell Direct**

Ideally, you will do the majority of your work through PowerShell remoting or PowerShell Direct.

Let’s start with PowerShell Remoting:

Before you can start, you will need to get the environment ready by:

- **Adding the IP address of the Nano Server to the list of trusted hosts** on your management computer (this is the computer from which you are performing your actions)

- **Make the account that you are using** a member of the Nano Servers administrators

- **Enable CredSSP** (if you are planning to use this feature)
To add the IP address to your list of trusted hosts, perform the following in PowerShell (elevated):

```
Set-Item WSMan:\localhost\Client\TrustedHosts "192.168.1.203"
```

Figure 47: Adding an IP to the trusted hosts list

**Note:** A confirmation will be requested. And, while it is less secure, it might be more efficient in development or testing to use * instead of IP addresses, so all of your test Nano Servers are trusted at once.

Now, we can start a remote Windows PowerShell session. Follow the command below:

```
$ip = "192.168.1.173"
Enter-PSSession -ComputerName $ip -Credential $ip\administrator
```

The first line simply adds the IP address to a variable called $ip. The second line will start the PowerShell session, and you will connect to the IP address — not the computer name, which I will come back to in a second — with the local administrator.

You probably won’t do this process often. After the server has joined the domain, you will want to connect through the DNS name and with the domain administrator. In the example, this becomes the following:

```
Enter-PSSession -ComputerName TP5Nano03.md.local -Credential md\administrator
```
You will see in PowerShell that you aren’t connected to your local computer anymore, but instead will be connected to the Nano Server. Everything you type from this moment on will be executed on the remote machine.

In Figure 49, you can see that I am connected to server Nano03. By using `ipconfig`, you can actually see the IP address of that server.

**Note:** Because you are running PowerShell on the Nano Server, some commands you want to use won’t be recognized. This is not a bug! Certain commands are simply not available on Nano Server.

After you’re finished with your work, you can use `Exit-PSSession` to exit the session (or simply use `Exit`).

`Enter-PSSession` is a pretty powerful tool and supports quite a few parameters. As an alternative, there is also `New-PSSession`. If you want to learn more on these commands, read the following page: [https://technet.microsoft.com/en-us/library/hh849707.aspx](https://technet.microsoft.microsoft.com/en-us/library/hh849707.aspx)

Now let’s discuss PowerShell Direct.

PowerShell Direct only works with a VM; it won’t help you with a physical server. PowerShell Direct allows you to run PowerShell commands inside the VM (like remote PowerShell), but it has no dependencies on the connection. So imagine that you can’t connect to your VM because of firewall restrictions, network not reachable or any other reason, yet you will be able to run PowerShell on the VM remotely, even when you have no connection to it.
There are, however, a few items you need keep in mind before using this function include:

- This only works on Windows 10 or Windows Server 2016 as a host.
- The guest that you are connecting to must be Windows 10 or Windows Server 2016.
- You need to be a Hyper-V administrator.
- You must run PowerShell elevated.
- You need to have credentials to the VM.
- The guest **must** run on the host from where you are performing the connection.

This works similarly to PowerShell remoting:

```
Enter-PSSession -VMName Nano03
```

**Note:** You must use the VMName. So, if your VM name is different than your computer name, you need to take this into account.

You can also connect through the GUID:

```
Enter-PSSession -VMId 09122ac5-45d7-4785-82bb-2c04b9b95537
```

![Figure 50: Enter-PSSession through GUID](image)

Last but not least, you can always use a full script block to send commands to the VM.

```
Invoke-Command -VMName VMName -ScriptBlock { Commands }
```

### Windows PowerShell CIM sessions over WinRM

You can also use Windows PowerShell CIM sessions over Windows Remote Management (WinRM) to interact with Windows Nano Server. You can use CIM sessions and instances in Windows PowerShell to run WMI commands over WinRM.
This is how this works:

$Cim = New-CimSession -Credential mr\administrator -ComputerName Nano03

In this example, I am creating a new CIM session to the computer tp5nano03 (with the md domain administrator) and storing it in a variable called cim.

When I run this command, I am prompted for the domain administrator account password:

![Figure 51: New CIM session](image)

Now that this is done, you can start running WMI commands. For example:

Get-CimInstance -CimSession $cim -ClassName Win32_ComputerSystem | Format-List *

Get-CimInstance -CimSession $cim -Query "SELECT * from Win32_Process"

![Figure 52: Running WMI commands over CIM](image)
Windows Remote Management

Nano Server allows WinRM, which is Microsoft’s implementation of the WS-Management Protocol. You can find more information about this protocol here: https://msdn.microsoft.com/en-us/library/windows/desktop/aa384426(v=vs.85).aspx

This means you can run programs remotely on the Nano Server with WinRM. However, as you might guess, not everything will work because the server itself does not support all of those programs. Before you can do this, you need to configure the service on the local client from where you are running the commands. Open an elevated command prompt and run the following commands:

\begin{verbatim}
winrm quickconfig
winrm set winrm/config/client @{TrustedHosts="*"}
chcp 65001
\end{verbatim}

Of course the * in TrustedHosts isn't the most secure thing to do, so in a production environment, you will want to be more selective when it comes to the hosts that you trust. Below is an example:

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig53.png}
\caption{Output of a WinRS command}
\end{figure}

Working with remote consoles

By now, you are probably thinking that you don’t like it that you need to do everything through PowerShell. I don’t blame you. PowerShell is a great solution, and while I do advise you to learn as much as possible about it, for more day to day tasks, you still want to use your known management consoles. However, as I said earlier, there are no consoles available on Nano Server. Does that mean you need to do everything through scripting? Not at all!

Almost everything is possible through your trusted management consoles that you run locally. Before you start, make sure you’ve installed the RSAT tools if you are performing this from a Windows 10 installation.

Let’s have a look on how this works.
Throughout this white paper, we have worked on several Nano Servers. In total, I have deployed six servers, but we are only going to use four of them here. They are Nano03 through 06. All four of these servers have the Hyper-V Package, failover clustering, management enabled and a static IP address in my server VLAN. The management workstation I am going to use is a Windows 2016 server with access to that VLAN to perform remote management, but this can also be done on a Windows 10 client.

Let’s start with Server Manager. Server Manager was introduced in Windows Server 2008. Unfortunately, at this point Server Manager began with quite a false start. Nobody liked it during that timeframe, and honestly, it was so limited that it was practically unusable. Microsoft made a lot of changes to Server Manager in 2012 and 2012 R2, and continued to improve this tool even more in Windows Server 2016. Connecting to a Nano Server is quite easy when you are connecting from a computer that is in the same domain. If this is not the case though, you need to make sure that WinRM is configured.

In our example, Nano03 through 05 has already added into Server Manager. Now, we need to add 06 and show that you can do remote management from your RSAT consoles.

The procedure is very simple:

![Figure 54: Server Manager](image)

This is my Server Manager with my hypervisor and three Nano Servers.
Let’s add Nano06 now by simply going to **Manage -> Add Servers.**

![Figure 55: Adding a server to Server Manager](image)

Depending on your user, you might need to right-click that server and choose **Manage As…** and provide different credentials. When that is done, you will have a new server that you can start managing.

![Figure 56: Server added to the list](image)
Note: By right-clicking on this server, you will get the option to choose Hyper-V Manager and Failover Cluster Manager, because this is a Hyper-V server with the clustering role already on it. Let’s choose Failover Cluster Manager now and add this node to our cluster.

It might be that you need to connect to the cluster the first time you do this. Because I have already done this a few times, Failover Cluster Manager remembered my settings so I see my cluster immediately. Now I am going to use Add Node… to add this server to the cluster.
Now, I have added a node to my cluster exactly as I would have done with Server Core or full server implementations.

Hyper-V Manager also works perfectly. I will continue to use Server Manager and right-click on one of the nodes to open Hyper-V Manager. You can also do this without Server Manager.
But these aren’t the only tools that work. You can still use Computer Management and more when connecting to Nano Server. A few examples are Event Viewer and Computer Management, as you can see below.

![Event Viewer connected remotely to a Nano Server](image1)

![Computer Management](image2)

While it is very easy and quick to remotely manage everything with PowerShell, sometimes you just want to have your GUI consoles to do your work. As you can see, many of those are available and usable when connecting to your Nano Servers.

**Server Management Tools**

Server Management Tools is completely new Azure-based web service, and at the time of this writing, in preview. This solution is intended to remotely manage your servers on-premises through a gateway, which you need to deploy on-premises.
The idea behind this solution is that it can replace specific tools in the future, such as

- Task Manager
- Registry Editor
- Event Viewer
- Device Manager
- SConfig (used in Server Core for some initial configuration)
- Control Panel
- File Explorer
- Performance Monitor
- Disk Management
- Users/Group Manager
- And likely, much more

The above list is only a subset of tasks announced so far. Again, this won't be explicitly for Nano Server, but it will also work with Server Core and Full Server installations. If you are planning to use this functionality for your servers with GUI or Core Servers, you can also use it for Nano Server.

You can deploy Server Management Tools today through the Azure portal.

More information can be found here: https://blogs.technet.microsoft.com/servermanagement/
Deploying apps on Nano Server

You already learned about the many packages available for Nano Server, and how you can deploy multiple roles and features onto Nano Server. But what about other workloads? What about custom-built applications?

The bad news is that you can’t take your current applications and simply start installing them on Nano Server. The reason is simple: MSI (Windows Installer) is not supported.

MSI has been built for local installations and improved over the years. But just ask any application packager how difficult it can be to remotely install MSI applications, and you will find out that MSI is not the best thing to use on Nano Server. MSI can also contain custom actions, requiring GUI elements, and, as you know by now, this is not possible on a Nano Server. And finally, something Microsoft emphasizes frequently is that many Nano Server deployments will be done offline in the long run (meaning that the workload is already deployed or installed before the server is first started, which, again, is something that MSI does not support).

So what is the alternative?

Enter Windows Server Apps (WSAs). WSAs give you a method to package and install applications, based on APPX on Nano Server.

Before you think that APPX are WSAs, please be aware that in a way they are the same, but WSAs extend the APPX functionality to support Windows Server-specific extensions. This also means that WSAs are NOT supported on Windows Client systems. They also cannot be submitted to the Windows Store. WSAs also need to run in machine-wide full trust, while store apps run in an app container-restricted environment.

You can find more detailed information in the three blog posts from the Nano Server team below:


Over the course of the last months, we have seen additional examples becoming available to run on Nano Server. MySQL, Node.js, and Python & Django are amongst the examples and I’m sure this list will continue to grow rapidly.
Appendix A: Additional information on packages

Deploying packages on Nano Server doesn’t mean that they immediately work. Just like with Windows Server Core and Full GUI you sometimes need to do some additional configuration. Also important to note is that some packages won’t support all functionality that is included with the versions on Core or Full GUI.

DNS Package

When deploying the DNS Server package, all PowerShell cmdlets are available locally on Nano Server. Of course, you can still do everything through remoting (as we will discuss later) or even use the DNS Manager MMC remotely to manage DNS.

Finally, adding the package to the deployment isn’t enough. You still need to enable the feature by using the following command:

```
Enable-WindowsOptionalFeature -Online -FeatureName DNS-Server-Full-Role
```

In this example, I use PowerShell Direct to connect to server Nano02. On the second line, I enable the DNS Server feature. On the third line, I import the DnsServer PowerShell module (Verbose again) to show you that all DNS PowerShell cmdlets are locally available on the Nano Server. After that, I will check if the DNS Server is actually running. Finally, I will exit the PowerShell Direct session.

```
Enter-PSsession -VMName Nano02
Enable-WindowsOptionalFeature -Online -FeatureName DNS-Server-Full-Role
Import-Module DnsServer -Verbose
Get-Service *dns
Exit-PSession
```
From now on, this server is a ready-to-run DNS Server.

Finally, if you want to use the DNS Server MMC console, you can do that by simply connecting remotely to another server from your management station.

IIS Server

There are three ways to install IIS Server on a NANO deployment. Two offline and one online installations are supported, but, as stated by Microsoft in the documentation, the offline installation with the package option, is the preferred method.
Offline installation

For offline installation, you need to add the package to the image you are building by using the — Packages parameter in the New-NanoServerImage script:

```
-Packages Microsoft-NanoServer-IIS-Package
```

This is, by far, the easiest method. However, you may already have an existing image or VHD\(x\) and need to add it post-installation. In that case, as long as it is still offline (e.g., not running), you can use DISM to do the work. Let’s look at DISM here:

The steps are below:

```
md mountdir

dism /Mount-Image /ImageFile:.\\Nano02.vhdx /Index:1 /MountDir:.\\mountdir

dism /Add-Package /PackagePath:.\\packages\Microsoft-NanoServer-IIS-Package.cab / Image:.\\mountdir

dism /Add-Package /PackagePath:.\\packages\en-us\Microsoft-NanoServer-IIS-Package.cab / Image:.\\mountdir

dism /Unmount-Image /MountDir:.\\MountDir /Commit
```

The above assumes that the name of the VHD is Nano02.vhdx and is reachable at that location.

Then, we add the IIS package and the localized (language) package. Don’t forget to do this because it won’t be done automatically and will fail if you forget.

Online installation

If the offline method is not an option for whatever reason (most likely in a container installation), you will need a different approach:

- Copy the Packages folder from the installation media to a local folder **inside** the Nano Server.
- Create an unattend.xml file and copy to a folder located on the Nano Server.
The XML file contents should look like this:

```xml
<?xml version="1.0" encoding="utf-8"?>

<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <servicing>
    <package action="install">
      <assemblyIdentity name="Microsoft-NanoServer-IIS-Feature-Package" version="10.0.14300.1000" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral"/>
      <source location="c:\packages\Microsoft-NanoServer-IIS-Package.cab"/>
    </package>
    <package action="install">
      <assemblyIdentity name="Microsoft-NanoServer-IIS-Feature-Package" version="10.0.14300.1000" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="en-US"/>
      <source location="c:\packages\en-us\Microsoft-NanoServer-IIS-Package_en-us.cab"/>
    </package>
  </servicing>
</unattend>
```

In the contents above, make sure that “C:\Packages\...” corresponds with the exact location of your packages on the Nano Server and the version number is correct.

Connect to your Nano Server through PowerShell (both remote and direct will work).

Now, switch to the folder where the unattend.xml is located and run:

```
dism /online /apply-unattend:\.\unattend.xml
```

**Note:** If you receive a 0x80004005 error, this can safely be ignored.
To confirm the package is actually installed, run the following:

```
dism /online /get-packages
```

Figure 66: Verifying that the installation succeeded

Finally, restart your Nano Server or type `net start w3svc` to start IIS.

**Working with IIS**

The moment IIS is running, you can check the default page to see if it is running.

Figure 67: Checking if IIS is running
By default, the following features are active:

<table>
<thead>
<tr>
<th>Default IIS Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default document</td>
</tr>
<tr>
<td>Directory browsing</td>
</tr>
<tr>
<td>HTTP Errors</td>
</tr>
<tr>
<td>Static Content</td>
</tr>
<tr>
<td>HTTP Logging</td>
</tr>
<tr>
<td>Static Content Compression</td>
</tr>
<tr>
<td>Request Filtering</td>
</tr>
<tr>
<td>IIS Administration PowerShell Module</td>
</tr>
</tbody>
</table>

*Table 2: Default IIS enabled features*

The following features are available but not enabled by default:

<table>
<thead>
<tr>
<th>IIS Features available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Http Redirection</td>
</tr>
<tr>
<td>Custom Logging</td>
</tr>
<tr>
<td>Request Monitor</td>
</tr>
<tr>
<td>Tracing</td>
</tr>
<tr>
<td>Dynamic Content Compression</td>
</tr>
<tr>
<td>Basic Authentication</td>
</tr>
<tr>
<td>Client Certificate Mapping Authentication</td>
</tr>
<tr>
<td>Digest Authentication</td>
</tr>
<tr>
<td>IIS Client Certificate Mapping Authentication</td>
</tr>
<tr>
<td>IP and Domain Restrictions</td>
</tr>
<tr>
<td>URL authorization</td>
</tr>
<tr>
<td>Windows Authentication</td>
</tr>
<tr>
<td>Application Initialization</td>
</tr>
<tr>
<td>CGI</td>
</tr>
<tr>
<td>ISAPI Extensions</td>
</tr>
<tr>
<td>ISAPI Filters</td>
</tr>
<tr>
<td>Server Side Includes</td>
</tr>
<tr>
<td>WebSocket Protocol</td>
</tr>
</tbody>
</table>

*Table 3: Optional IIS Features*

The scope of this paper is not to show how to change IIS, but note that it can’t be done by an MMC. If your web administrators want to make changes, they have three options:

- Using the **IIS Administration** module for Windows PowerShell remotely
- Using **AppCmd.exe** remotely
- Editing the IIS configuration store manually
Hyper-V

Hyper-V is one of these workloads that I expect to run often on Nano Server in the future. Today, there are many options already, but not everything is possible yet. Let’s take a look at the differences:

• You must do all management remotely, and the computer that’s the management server must run the same build. In other words, you can’t manage Nano Server with an older Hyper-V manager or older PowerShell cmdlets.

• RemoteFX is not available. If this is a requirement for your Hyper-V host(s), you can’t work with Nano Server.

Now, let’s talk management. As already stated, you can use the console remotely to connect to the host (see Working with remote consoles).

It is very important to get the authentication correct.

If you want to use **constrained** delegation, then you need to know it works exactly as in previous releases. Check out these articles for more information:

Enabling Hyper-V Remote Management-Configuring Constrained Delegation For SMB and Highly Available SMB

Enabling Hyper-V Remote Management-Configuring Constrained Delegation For Non-Clustered Live Migration

If you want to use CredSSP, make sure it is enabled for PowerShell Remoting first. From that moment on, you will be able to use CredSSP, both for the Hyper-V Manager console as Windows PowerShell cmdlets.

Failover Clustering

Just as I expect Hyper-V to become an important role for Nano Server (and an enabler for this technology), I expect the same thing for Failover Clustering. Both will go hand-in-hand.

The good news is that Failover Clustering works the same way as in Windows Core Server.

However, as you have read here already many times, you must manage clusters remotely. Again, this can be done through PowerShell, but you can also remotely connect with the cluster manager (as already explained in this white paper).

A few notes to get your Nano cluster working:

• All nodes must be joined to the same domain.

• The domain account must have Administrator privileges on all Nano Server nodes.

• All commands executed must be run in an elevated command prompt.
However, there are some items that are not supported on Nano Server:

- Running failover clustering cmdlets locally on a Nano Server is not possible.
- Every clustering role besides Hyper-V and File Server is not currently supported.

For more information on Clustering PowerShell cmdlets, visit this page: https://technet.microsoft.com/library/ee461009.aspx
Appendix B: Setup & Boot eventing, Kernel Debugging & Emergency Management Services

Whether we like it or not, sometimes things go wrong and we need to troubleshoot. We already talked about doing things remotely, and as long as you can connect to the server through the various methods, you can do your troubleshooting with PowerShell or the remote consoles. However, sometimes even that isn’t possible and you find yourself in a situation where the server does not boot correctly or is completely stuck or damaged. Depending on the workload you are running, it might be easier and faster to simply recreate the image and deploy it, or you may need to troubleshoot the problem. In that case, the below methods can be used to troubleshoot the server.

One final note here, we already discussed, the Nano Server Recovery Console in the management section is also a great way of troubleshooting your Nano Server.

Setup & Boot Eventing

One of the tools you can use is the setup and boot event collection. This is designed to get your logs (ETW messages) of your server and collect them through a specific collector. This is not for Nano Server alone: This will also be possible with Server Core and Full Server.

Through this mechanism, you will be able to view remotely your evets while the server boots (or is in setup) and you can get those events from both physical and virtual machines. This collector will be responsible for grabbing the events. Note: The collector won’t pull the events. This is a push operation so it will be the Nano Server that pushes the events to your collector (also called the SBEC collector).

This is a new feature that comes with Windows Server 2016. You can find more information on the requirements and setup here: https://technet.microsoft.com/en-us/library/mt126188.aspx

Letps://technet.microsoft.com/en-us/library/mt126188.aspx. You can find more information on the requirements and setup here: physicaln order to do this procedure, you need to already have a collector in place.

Copy the <root>\Windows\System32\WindowsPowerShell\v1.0\Modules\BootEventCollector folder from the collector server to your computer where you are creating (or modifying) the VHD(x). (So, not on the Nano Server itself, but on the machine used to create or modify the Nano Server image.)

• In PowerShell, run the following command:
  ```ps
  Import-Module BootEventCollector
  ```

• Now, you need to tell the Nano Server to enable Autologgers. Again, PowerShell will do the trick by running:
  ```ps
  Enable-SbecAutoLogger -Path <drive>\pathtoyourserver\Server.vhd
  ```
Note: This enables the default events. You can change the default events if needed. 
Read more about this here: https://msdn.microsoft.com/en-us/library/windows/desktop/aa363694(v=vs.85).aspx

- Finally, you need to run the following command to tell the Nano Server the settings of the collector (IP, listening port and security key):

  ```powershell
  Enable-SbecBcd -Path <drive>\pathtoyourserver\Server.vhd -CollectorIp <IP> -CollectorPort <port> -Key <secretkey>
  ```

- Of course, you need to update your collector computer by adding the Nano Server IP address (but you can also use MAC or an IP range).

From then on, the ETW messages will be pushed to the collector and you can use your favorite tool (like message analyzer, for example) to troubleshoot messages when necessary.

**Kernel debugging**

Kernel debugging is also possible with Nano Server. A Nano Server image supports kernel debugging using four different methods. There is one catch when you want to use kernel debugging with a VHDx image: you need to include the Hyper-V package and the corresponding PowerShell modules.

The four methods include: using a serial port, using the TCP/IP network, using a Firewire or using USB.

To do so, you need to enable it on the image.

**Serial port**

```powershell
New-NanoServerImage -MediaPath \PathToMedia -BasePath .\BasePath -TargetPath .\PathToVHD -DebugMethod Serial -DebugCOMPort 1 -DebugBaudRate 9600
```

Here, you'll configure the kernel debugging over serial port 1 with a baud rate of 9600 bps. (Defaults are port 2 and 115200 bps). If you plan on using the Emergency Management Services, you need to configure them over a different port because they cannot coexist on the same port.

**TCP/IP network**

```powershell
New-NanoServerImage -MediaPath \PathToMedia -BasePath .\BasePath -TargetPath .\PathToVHD -DebugMethod Net -DebugRemoteIP 192.168.1.100 -DebugPort 64000
```

The above example will ensure the kernel debugging can only be done by the computer with IP 192.168.1.100 over port 64000.

*Note: The port you specify has to be greater than 49152. There will also be an encryption key generated (together with the VHD(x)), but if you want, you can create your own by adding the parameter — DebugKey <key>*

**Firewire (IEEE1349 protocol)**

```powershell
New-NanoServerImage -MediaPath \PathToMedia -BasePath .\BasePath -TargetPath .\PathToVHD -DebugMethod 1394 -DebugChannel 3
```

If you want to use this method, be aware that the parameter — DebugChannel is mandatory.
USB

```powershell
New-NanoServerImage -MediaPath \PathToMedia -BasePath .\BasePath -TargetPath .\PathToVHD -DebugMethod USB -DebugTargetName <nano>
```

The name behind the — **DebugTargetName** parameter is the one you are going to use when you connect to the remote debugger.

For more information on the debugging process itself, visit the following two links:

- Setting Up Kernel-Mode Debugging over a Network Cable Manually
- Remote Debugging Using WinDbg

**Emergency Management Services**

The Emergency Management Services (EMS) provides you with a serial console interface into the bootloader menu within Microsoft Windows. EMS allows you to perform basic troubleshooting by using a terminal emulator over a serial port. This might seem like something new, but those who have worked in IT a bit longer will certainly remember troubleshooting over a serial port.

In most installations, this is enabled by default. In Nano Server, this must be enabled when creating the image, so you can use it when needed.

Luckily, this is easy to do and can simply be added as parameters when you are creating your image:

```powershell
New-NanoServerImage -MediaPath \PathToMedia -BasePath .\BasePath -TargetPath .\PathToVHD -EnableEMS -EMSPort 1 -EMSBaudRate 9600
```

Here we enable EMS over serial port 1 with a baud rate of 9600. The defaults are port 1 and 115200 bps. When you want to use this with a VHDX implementation, the same rule as with the kernel debugging applies: you will need to include the Hyper-V feature and the corresponding Windows PowerShell modules.
Appendix C: Unattend, SetupComplete and DISM

We’ve talked a lot about using parameters when creating images. Those of you who have followed the technical previews probably can remember that before TP4, there was a lot of work that needed to be done through the unattend file, DSIM and even a run at first boot script. Some of you will say it’s great that this is not necessary anymore, but others will say that it was a good thing because this is a way that works for each server deployment today. The good news is that these techniques aren’t going away and you can still use them to get your images modified. So, let’s have a look on how they work, in case you need them:

**DISM**

Let’s start with Deployment Image Servicing and Management (DISM) because this will be used for the next steps in this chapter. Following the definition, DISM.exe is a command-line tool that can be used to service a Windows image or to prepare a Windows Preinstallation Environment (PE) image. DISM can be used to service a Windows image (.wim) or a virtual hard disk (.vhd or .vhdx).

In Nano Server, we will be changing the VHD or VHDX image that we have built by using the scripts most of the time. As already stated, you can do a lot with the parameters, but sometimes it is necessary to work with DISM to do certain things, or maybe you use DISM as the main tool to create your images and you don’t want to change that method for Nano Server alone.

Now, let’s look at how this works with a Nano Server image.

The basics are (almost) always the same

```

dism /Mount-Image /ImageFile:.\NanoServer5.vhd /Index:1 /MountDir:.\mountdir
cd mountdir
dir
md testdir
cd..
dism /Unmount-image /MountDir:.\mountdir /commit
```

In this example, I am mounting the image **NanoServer5.vhd** under the Mount Directory **mountdir**.

In the next step, I simply browse to that **mountdir**, do a simple **dir** command to show the root of the file system.

Now you can start using commands to change your image. For now, the only change I made is creating a folder called **testdir** under the root.

The **cd..** in this script is just to make sure that I am a folder under the **mountdir**, so I can unmount the image and commit the changes made, which I do with the last line.
However, you can do quite a few things to change your image, and we already have some examples in this paper for the offline installation of IIS or offline installation of additional packages.

**Unattend File**

An unattend.xml file, when used in the right location, will configure specific settings when you boot an image. This is also referred to sometimes as an answer file. For a quick overview of unattend files, visit this page: [https://technet.microsoft.com/en-us/library/cc748874(v=ws.10).aspx](https://technet.microsoft.com/en-us/library/cc748874(v=ws.10).aspx)

If you read all the different parameters (described earlier) that can be used when creating a new image, you may have noticed a parameter called `-UnattendPath` that allows you to add your custom unattend.xml file easily. But you might want to do this through offline servicing, so let’s see how this looks.

In this white paper, I already gave a few examples of the unattend.xml file, but let’s look at one more and then see how you can inject this into your image.

```xml
  <settings pass="offlineServicing">
    <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
      <ComputerName>NANO05</ComputerName>
    </component>
  </settings>
  <settings pass="offlineServicing">
    <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
      <UserAccounts>
        <AdministratorPassword>
          <Value>P@ssw0rd</Value>
          <PlainText>true</PlainText>
        </AdministratorPassword>
        <TimeZone>Pacific Standard Time</TimeZone>
      </UserAccounts>
    </component>
  </settings>
  <settings pass="oobeSystem">
    <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
      <UserAccounts>
        <AdministratorPassword>
          <Value>P@ssw0rd</Value>
          <PlainText>true</PlainText>
        </AdministratorPassword>
      </UserAccounts>
    </component>
  </settings>
</unattend>
```
You will notice a few things that we are doing with this file. First, we are actually setting the computer name to **Nano05**. But we are also setting the **administrator** password, changing the **time zone** to the correct one and setting the **registered owner** to my organization.

By copying this script and simply changing those four parameters to your own specifications, you have a working unattend.xml file, ready to be injected through the parameter or offline as done below.

To do this, you are going to use DISM as we discussed above and put the file at its correct place.

```
dism /Mount-Image /ImageFile:.\NanoServerVM5.vhd /Index:1 /MountDir:.\mountdir

cd mountdir

cd Windows

cd Panther

 copy e:\unattend.xml

 cd..

cd..

cd..

dism /Unmount-image /MountDir:.\mountdir /commit
```

In the first line, I am using DISM to mount the image. In the second line, I go to that mountdir. To use the unattend.xml file, you need to place it in the correct location of the image. That location is `<root>\Windows\Panther`. 
After placing the file in the correct location, you can commit the changes or do other tasks first.

**Using setupcomplete.cmd**


Because there is no Windows Setup in Nano Server, this script will be executed the first time the server starts. Just like with the unattend.xml file, you can attach a script (or even commands directly) through a parameter `-SetupCompleteCommands` or manually in a specific location to make it work.

```bash
PS E:\NanoServerWMS\mountdir\Windows\Panther> dir

Directory: E:\NanoServerWMS\mountdir\Windows\Panther

Mode          LastWriteTime              Length Name
-------------  --------------------------      ------- -------
-d---- 10/20/2015  12:50 AM          1908 Unattend.xml
-a---- 10/20/2015  12:50 AM          81920 setup.c1
-a---- 10/20/2015  12:50 AM          setupast.log
-a---- 10/20/2015  12:50 AM         0 setuperr.log
-a---- 10/20/2015  12:50 AM        1328 Unattend.xml

PS E:\NanoServerWMS\mountdir\Windows\Panther> |
Again, I am using DISM here to place the file under the right folder. This time, I need to place it under `<root>\Windows\Setup\Scripts`.

**Note:** The folders setup and scripts probably won’t exist, so you will need to create them.

![Figure 69: Location of setupcomplete.cmd](image)

Here is an example of this file:

```powershell
powershell.exe -command "Import-Module C:\windows\system32\windowspowershell\v1.0\Modules\Microsoft.PowerShell.Utility\Microsoft.PowerShell.Utility.psd1; Import-Module C:\windows\system32\WindowsPowerShell\v1.0\Modules\NetAdapter\NetAdapter.psd1; $ifa = (Get-NetAdapter -Name Ethernet).ifalias; netsh interface ip set address $ifa static 192.168.1.254 255.255.255.0 192.168.1.1; netsh interface ip set dns $ifa static 192.168.1.220; netsh interface ip add dns $ifa 192.168.1.1 index=2"
```

The above should be one line in the script, so here I’ll split it into pieces and explain what I am doing.

**Powershell.exe — command** is the start of the script. Everything after that, which is between ("), are the actual commands to execute. I can split up that large line into multiple commands:

```powershell
Import-Module C:\windows\system32\windowspowershell\v1.0\Modules\Microsoft.PowerShell.Utility\Microsoft.PowerShell.Utility.psd1;
Import-Module C:\windows\system32\WindowsPowerShell\v1.0\Modules\NetAdapter\NetAdapter.psd1;
$ifa = (Get-NetAdapter -Name Ethernet).ifalias;
netsh interface ip set address $ifa static 192.168.1.254 255.255.255.0 192.168.1.1;
netsh interface ip set dns $ifa static 192.168.1.220;
netsh interface ip add dns $ifa 192.168.1.1 index=2
```

The first two simply load the correct PowerShell modules that are necessary to perform the job.

After that, I store the index of the nic with the name **Ethernet** into the variable $ifa.

Then, I have three commands that use **netsh** to set a **static IP address**, **subnet mask**, **gateway**, **primary** and, finally, **secondary DNS server**.

As you can see, by using this method there are quite a few things you can change as long as you can script it.
Appendix D: Using MPIO on Nano Server

Multipath is supported on Nano Server. For more about Multipath, visit: https://technet.microsoft.com/library/cc725907.aspx

There are some differences on Nano Server to be aware of, such as:

- Only MSDSM is supported.
- The Load Balancing Policy is chosen dynamically and cannot be modified.
  The policy has these characteristics:
  - Default — RoundRobin (active/active)
  - SAS HDD — LeastBlocks
  - ALUA — RoundRobin with Subset
- Path states (active/passive) for ALUA arrays are picked up from the target array.
- Storage devices are claimed by bus type (for example, FC, iSCSI, or SAS). When MPIO is installed on Nano Server, disks are still exposed as duplicates (one available per path) until MPIO is configured to claim and manage particular disks. You can use the sample script on this page to claim or unclaim those: https://technet.microsoft.com/en-us/windows-server-docs/get-started/mpio-on-nano-server

Enabling MPIO needs to be done after you have started the Nano Server. As we have done multiple times, you simply are going to use PowerShell remoting to connect to the server and then apply the following command:

Enable-WindowsOptionalFeature -Online -FeatureName MultiPathIO
Appendix E: Using Windows Update

While we stated that there will be less updates when working with Nano Server, there will still be updates. In a production environment, you need to keep your servers up-to-date as a best practice.

Let’s start at the beginning and connect to a Nano Server (through PowerShell remoting) and run the following commands:

```powershell
$sess = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName MSFT_WUOperationsSession
$scanResults = Invoke-CimMethod -InputObject $sess -MethodName ScanForUpdates -Arguments @{SearchCriteria="IsInstalled=0";OnlineScan=$true}
```

As you can see in Figure 70, in this case there is an update available. In cases when there are no updates available, you will get the following result

As you can see in Figure 70, in this case there is an update available. In cases when there are no updates available, you will get the following result

![Figure 70: Finding out if there are available updates](image1)

![Figure 71: Notification when no updates are available](image2)
The next step is installing the updates. I will simply use this command to apply all updates:

```
$scanResults = Invoke-CimMethod -InputObject $sess -MethodName ApplyApplicableUpdates
```

![Figure 72: Applying the updates](image)

Now, I still need to restart the computer to make it effective, which can be done with:

```
Restart-Computer
```

There is one big issue you might encounter here. When you have Windows Defender deployed on the Nano Server, it will prevent the updates from installing. There are two workarounds at this moment.

One workaround, which is not really workable, is to uninstall Windows Defender, apply the updates and reinstall Windows Defender again.

The second workaround is to download the updates on another computer, copy them to the Nano Server and use our good old friend DISM to apply the updates. I personally prefer this method over the first workaround.
After applying the updates, you might want to check whether they are actually installed:

```
$sess = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName MSFT_WUOperationsSession
$scanResults = Invoke-CimMethod -InputObject $sess -MethodName ScanForUpdates -Arguments @{SearchCriteria="IsInstalled=1";OnlineScan=$true}
```

Figure 73: Result of installed updates

As you can see in Figure 73, the update we saw before is now installed. There is also another method if you want a more thorough output:

```
Get-WindowsPackage -Online
```

Figure 74: Looking at installed updates
The methods we used above will go online to detect and install updates. If you prefer to use WSUS (as an example), that can also be used. But to do that, you need to first modify the registry keys. For a full listing of potential registry keys that need to be changed, you can look here: https://technet.microsoft.com/library/cc708449(v=ws.10).aspx

Modifying registry keys cannot be done through the MMC, so you will need PowerShell again.

Below you see an example of changing a registry key (if it already exists):

```
set-itemproperty -name WUServer -path "hklm:\software\policies\microsoft\windows\windowsupdate" -value "http://UpdateServer"
```

If you want to browse the registry first, you can use simple commands to do so:

```
cd HKLM:\
cd HKLM:\SOFTWARE\Policies\Microsoft\Windows\Get-ChildItem -ErrorAction SilentlyContinue | Format-Table Name, SubKeyCount, ValueCount -AutoSize
```

If you want to use auto-updating, you need to know that this doesn’t work the way you are used to it working. For now, you can use the PowerShell one-liners in a single script and create a scheduled task in Nano Server to check for new updates on a regular basis.
Appendix F: More on PowerShell on Nano Server

PowerShell v5 comes in two editions, the desktop edition and the core edition. As you can guess, in Nano Server you can only use the Core version. To quickly check what version you are using, use the following command (with PowerShell remoted to the Nano Server):

```
$PSVersionTable
```

As said, this needs to be the Core version; otherwise, I was likely not connected to a Nano Server.

It is important to know that because of this Core version, there is a reduced footprint, but also less functionality. As of today, this is the list with items that are not supported on PowerShell Core (and therefore not on Nano Server):

- ADSI, ADO, and WMI type adapters
- Enable-PSRemoting, Disable-PSRemoting (PowerShell remoting is enabled by default)
- Scheduled jobs and PSScheduledJob module
- Computer cmdlets for joining a domain { Add | Remove }
- Reset-ComputerMachinePassword, Test-ComputerSecureChannel
- Profiles (you can add a startup script for incoming remote connections with Set-PSSessionConfiguration)
Clipboard cmdlets

- EventLog cmdlets: Clear | Get | Limit | New | Remove | Show | Write (use the New-WinEvent and Get-WinEvent cmdlets instead).

- Get-PfxCertificate cmdlet

- TraceSource cmdlets: Get | Set

- Counter cmdlets: Get | Export | Import


- Logging and tracing using PSDiagnostics module

- Get-HotFix (to obtain and manage updates on Nano Server)

- Implicit remoting cmdlets: Export-PSSession | Import-PSSession

- New-PSTransportOption

- PowerShell transactions and Transaction cmdlets: Complete | Get | Start | Undo | Use

- PowerShell Workflow infrastructure, modules and cmdlets

- Out-Printer

- Update-List

- WMI v1 cmdlets: Get-WmiObject, Invoke-WmiMethod, Register-WmiEvent, Remove-WmiObject, Set-WmiInstance (use CimCmdlets module instead.)

(Source: https://technet.microsoft.com/en-us/windows-server-docs/get-started/powershell-on-nano-server)

The same is also true for desired state configuration. As you may have noticed, there is a package for DSC on Nano Server, but this does not mean everything that you can do with DSC on a full server is supported on Nano Server. For the differences, check this page: https://msdn.microsoft.com/powershell/dsc/nanodsc
Appendix G: Installing roles and features online

We demonstrated how to create an image and install the roles and features through packages. You also now know that you can do an edit of your image and can use DISM to do an online installation. But there is one more very important option that you can use.

You can find and install Windows Packages from an online repository by using the NanoServerPackage provider. The NanoServerPackageProvider is part of the PackageManagement (OneGet) PowerShell module. More information on OneGet can be found here: https://github.com/OneGet/oneget

To be able to use this in a Nano Server, you need to do some preparations. We are going to use the Nano02 server to do our work. To start, I have used PowerShell remoting to connect to that server.

The first item on the list is to install and import the packageprovider since it is not there by default.

```powershell
Save-Module -Path "$env:ProgramFiles\WindowsPowerShell\Modules" -Name NanoServerPackage -MinimumVersion 1.0.0.0
Import-PackageProvider NanoServerPackage
```

**Note:** Here, I use the `-Force` at the end to avoid the confirmation questions, but you can do this without Force also.

Now that the PackageProvider is installed and imported, we can start using commands to find packages from an online repository.
Let’s start by searching for packages.

### Find-NanoServerPackage

In Figure 78, you can see the result of the online search for packages. These packages have a version number and you can see also the **Culture** (or language) versions that exist. With the **Find-NanoServerPackage** you can use parameters to narrow your search. A few examples are shown here:

- `Find-NanoServerPackage -Culture en-us`
- `Find-NanoServerPackage -Name *IIS*`
- `Find-NanoServerPackage -AllVersions -Name *IIS* -RequiredVersion 10.0.14300.1000 -Culture en-us`

**Figure 78: Finding Packages online**

**Figure 79: Specific search**
In the last search, we specifically looked for the IIS package with a requiredversion (maximumversion and minimumversion also work) and a specific culture. We want to deploy this specific package.

There are several methods to do this, so let’s have a look at all of them.

**To an offline image**

When you have the provider running on your workstation and your Nano Server is offline, you can use the following command to install the package:

```powershell
Install-NanoServerPackage -Name Microsoft-NanoServer-IIS-Package -ulture en-us -RequiredVersion 10.0.14393.0 -ToVHd D:\MyNanoVhd.vhdx
```

**Downloading a package without installing it**

When you want to download the packages but don’t want to immediately install it (for example, if you prefer to do updates from your local machine), then you can use the following command:

```powershell
Save-Package -provider NanoServerPackage -Name Microsoft-NanoServer-IIS-Package -Path .\temp -Culture en-us -MinimumVersion 10.0.14393.0
```

This will save the package (and the language package) to a local folder temp.

**Direct install**

If you prefer to install the package immediately, you can use the following command:

```powershell
Install-NanoServerPackage -Name Microsoft-NanoServer-IIS-Package -Culture en-us -RequiredVersion 10.0.14300.1000
```

![Figure 80: Direct installation of a package](image)
If you want to be sure that it is installed, you can run the following command to verify:

```
dism /online /get-packages
```

![Figure 81: Verifying if the package is installed](image)

In Figure 81, you can see that the IIS package is included and that I also have the language pack.

Another method is by using the following command:

```
Get-Package -provider NanoserverPackage
```

This will give the following output:
Appendix H: Connecting to a DFS host

In many cases, you will want your Nano Server to connect to a DFS (Distributed File System) host. This is possible, but it does require some configuration on the host computer and the Nano Server. This is only supported on a DFS host computer running Windows 10 or Windows Server 2016.

Let’s start with the prerequisites: The Nano Server needs to be joined to the same domain as the DFS host.

- Now we need to do Remote PowerShell to the Nano Server:

```powershell
$ip = "192.168.1.201"
$user = "$ip\Administrator"
Enter-PSSession -ComputerName $ip -Credential $user

Enable-WSManCredSSP -Role Server
Net localgroup administrators <md\administrator> /add
```

Now, go to the DFS host and run the following PowerShell (remember, this is in an elevated PowerShell command window):

```powershell
Enable-WSManCredSSP -Role Client -DelegateComputer nano01
$s1=new-pssession -ComputerName Nano01 -authentication CredSSP -Credential mr\administrator
```

Finally, it’s time to connect to the Nano Server with the session you just created on the DFS host, and create a persistent or temporary drive (use the parameter -Persist to make it persistent):

```powershell
enter-psSession $s1
New-PSDrive -Name <drive label> -PSProvider FileSystem -Root <\DFS\share>
```
Appendix I: Performance & Event monitoring

We already discussed troubleshooting by using setup and boot eventing, kernel debugging, emergency management services and the Nano Server recovery console. We also looked at Server Manager to connect remotely to a Nano Server to look at events (or use other MMC tooling to look at specific workloads running on Nano Server). But Nano Server also supports the EWT (Event Windows Tracing) framework so you can use that to do performance analysis.

Let’s have a look at what you can do next.

First, we start with the windows performance recorder tool (wpr.exe). This tool allows you to record performance based on event tracing for windows. After that recording, you can use Windows Performance Analyzer or Message Analyzer to do your analysis. You will need to do this analysis afterwards on another computer, which also means you need to be able to copy the data off your Nano Server to another. You can do that by connecting to it over the network, or use PowerShell. Those techniques have been seen already in the Managing Nano Server section.

Since the Windows Performance Recorder tool, event tracing and Typeperf are not in scope for this paper, I won’t go into depth except to mention that:

- **Wpr.exe** can be used to query providers that exist on your Nano Server (wpr.exe -providers).
- **Wpr.exe** can be used to query profiles (wpr.exe -profiles).
- **Event Tracing Management cmdlets** can be used to record traces from providers and profiles (see [https://technet.microsoft.com/library/dn919247.aspx](https://technet.microsoft.com/library/dn919247.aspx) for all the cmdlets).
- **Typeperf.exe** can be used to monitor performance counters. This is the command line equivalent for perfmon.exe GUI that most administrators know and use regularly. (See [https://technet.microsoft.com/library/bb490960.aspx](https://technet.microsoft.com/library/bb490960.aspx) for command line options).
- If you don’t want to connect remotely to event viewer on Nano Server, you can also use **Get-Winevent** ([https://technet.microsoft.com/library/hh849682.aspx](https://technet.microsoft.com/library/hh849682.aspx)) which allows you to get specific events from Nano Server.
- Finally, you can also use **wevtutil.exe** ([https://technet.microsoft.com/en-us/library/cc732848(v=ws.11).aspx](https://technet.microsoft.com/en-us/library/cc732848(v=ws.11).aspx)), which allows you to retrieve information from event logs and specific publishers.
About the Author

Mike Resseler is a Director, Product Management for Veeam. Mike is focused on technologies around Hyper-V and System Center. With years of experience in the field, he presents on many occasions on large events such as MMS, TechEd and TechDays. Mike has been awarded the MVP for System Center Cloud and Datacenter Management since 2010 and received the Hyper-V MVP since 2014. His major hobby is discussing and developing solid Disaster Recovery scenarios. He also has enterprise-class experience in private cloud architecture and deployment with a focus on bottom to instead of top protection. He holds certifications in many Microsoft Technologies such as MCITP.

Follow Mike on @MikeResseler or @Veeam and on Google+.

About Veeam Software

Veeam® recognizes the new challenges companies across the globe face in enabling the Always-On Business™, a business that must operate 24.7.365. To address this, Veeam has pioneered a new market of Availability for the Always-On Enterprise™ by helping organizations meet recovery time and point objectives (RTPO™) of less than 15 minutes for all applications and data, through a fundamentally new kind of solution that delivers high-speed recovery, data loss avoidance, verified protection, leveraged data and complete visibility. Veeam Availability Suite™, which includes Veeam Backup & Replication™, leverages virtualization, storage and cloud technologies that enable the modern data center to help organizations save time, mitigate risks and dramatically reduce capital and operational costs.

Founded in 2006, Veeam currently has 41,000 ProPartners and more than 205,000 customers worldwide. Veeam’s global headquarters is located in Baar, Switzerland, and the company has offices throughout the world. To learn more, visit http://www.veeam.com.
AVAILABLE NOW

NEW Veeam Availability Suite 9.5

Availability for the Always-On-Enterprise

go.veeam.com/v9-5