

Kapitel 26

Windows Server-Container, Docker und Hyper-V-Container

In diesem Kapitel:

Die Grundlagen zu Container und Docker	678
Nano-Server als Container-Host verwenden	682
Erweiterte Konfiguration von Containern durchführen	685
Hyper-V-Container in Windows Server 2016 anlegen	689
Zusammenfassung	692

Neben Nano-Servern gehören Windows Server-Container als Docker-Implementation zu den wichtigsten Neuerungen in Windows Server 2016. Windows Server-Container lassen sich auch auf Nano-Servern ausführen und Nano-Server lassen sich wiederum als Cluster betreiben. Zusätzlich können Container sowohl in virtuellen Umgebungen als auch in virtuellen Clustern hochverfügbar zur Verfügung gestellt werden.

Container lassen sich somit auf allen Arten von Windows-Servern betreiben, also auf Nano-, Core- und Windows-Servern mit grafischer Oberfläche. Außerdem steht die Container-Technologie in Windows 10 Professional und Enterprise ab Version 1607 (Anniversary Update) zur Verfügung. Entsprechend können Administratoren oder Entwickler mit Windows 10 ebenfalls Container erstellen und diese auf Container-Hosts mit Windows Server 2016 übertragen. Hier kann auch auf den Docker-Hub in der Cloud gesetzt werden, um die Container-Images über das Internet und die Cloud zu übertragen.

Die Grundlagen zu Containern und Docker

Windows Server-Container ermöglichen den Betrieb von Cloudanwendungen oder Webdiensten in einer sicheren und einfach zu erstellenden Umgebung. Alles, was Sie benötigen, ist ein Container-Host auf Basis von Windows Server 2016. Dabei kann es sich um einen physischen Server handeln, eine virtuelle Maschine (VM) oder einen virtuellen Computer in Microsoft Azure.

Innerhalb des Container-Hosts, der zum Beispiel auf Basis eines Nano- oder Core-Servers mit Windows Server 2016 zur Verfügung gestellt wird, verwalten Sie die Images für Container und die Container selbst. Die Verwaltung findet vor allem über die PowerShell oder die Eingabeaufforderung statt. Auch die Container werden hierüber verwaltet. Der Verbindungsaufbau zum Container-Host kann über eine RDP-Sitzung erfolgen.

Container im Vergleich zu virtuellen Servern

Die Windows Server-Container sowie deren Erweiterung Hyper-V-Container basieren auf der Plattform Docker (<https://www.docker.com>). Microsoft arbeitet eng mit den Entwicklern von Docker zusammen, um eine optimale Integration von Docker zu gewährleisten. Die Verwaltung von Docker können Sie mit dem Docker-Client oder in der PowerShell vornehmen. Sie können Container auch mit System Center 2016 verwalten.

Virtualisieren Unternehmen Server auf herkömmlichen Technologien, gibt es einige Nachteile. Ein Nachteil besteht zum Beispiel darin, dass die Betriebssysteme in den virtuellen Servern eine Grundlast verursachen und damit Ressourcen verbrauchen und Sicherheitslücken darstellen.

Das Betriebssystem in Docker-Containern und die notwendigen Ressourcen sind auf dem Container-Host zusammengefasst. Startet ein Container, muss er nicht das komplette Betriebssystem booten, Bibliotheken laden und Ressourcen für das eigene Betriebssystem zur Verfügung stellen. Stattdessen nutzen Container nur Teile des Betriebssystems auf dem Container-Host. Die Vorteile dabei sind eine geringere Auslastung der Server und mehr Sicherheit. Der gestartete Container betrachtet die lokale Festplatte wie eine Kopie des Betriebssystems, inklusive Arbeitsspeicher, Dateien und andere Ressourcen.

Virtuelle Anwendungen sind kleiner als virtuelle Server, benötigen weniger Ressourcen und sind gleichzeitig sicherer, da die meisten Angriffspunkte fehlen. Außerdem lassen sich wesentlich mehr virtuelle Anwendungen auf einem Virtualisierungs-Host betreiben als herkömmliche virtuelle Server.

Windows Server-Container unterstützen zahlreiche Programmiersprachen und -Umgebungen. Entwickler können unter anderem .NET, ASP.NET, PowerShell, Python, Ruby on Rails, Java und viele andere Umgebungen nutzen. Der Container-Host auf Basis von Windows Server 2016 steuert, welche und wie viele Ressourcen des Hosts ein Container nutzen darf, ohne die anderen Container oder den Host zu beeinträchtigen.

Das Container-Feature installieren

Um Container zu nutzen, müssen Sie das Container-Feature installieren. Dabei spielt es zunächst keine Rolle, ob es sich um einen vollständig installierten Server, um einen Nano-Server oder um eine Core-Installation handelt. Auf einem herkömmlichen Server verwenden

Sie dazu den Server-Manager oder die PowerShell. Auf einem Core-Server installieren Sie das Feature vor allem in der PowerShell. Dazu verwenden Sie den Befehl *Install-WindowsFeature Containers*. Beim Erstellen eines neuen Nano-Servers lassen sich weitere Optionen in das Image einbinden. Dadurch lassen sich auch die Container-Funktionen installieren. Dazu verwenden Sie die Option *-Containers*. Mehr zu diesem Thema lesen Sie in Kapitel 2.

Anschließend benötigen Sie ein Image, auf dessen Basis Container erstellt werden können. Hier kommt entweder eine Core-Installation oder ein Nano-Server-Image zum Einsatz. Die Verwaltung erfolgt normalerweise mit der PowerShell, alternativ mit dem Docker-Client, den Sie über die PowerShell herunterladen können.

Mit dem Windows-Docker-Client können Sie Container verwalten. Der Docker-Client dient nur zur Verwaltung der Container-Technologie, die direkt in Windows Server 2016 integriert ist. Er stellt selbst keinen Serverdienst zur Verfügung. Der Client kann die Windows Server-Container verwalten, zusätzlich aber auch andere Hosts, zum Beispiel Linux-Server.

In Windows Server 2016 ist der Docker-Client ebenfalls integriert und steht über die Eingabeaufforderung zur Verfügung.

Um Windows Server-Container zu verwalten, installieren Sie die notwendigen Erweiterungen auf dem Server. Dazu muss der Server über eine Internetverbindung verfügen:

Install-Module -Name DockerMsftProvider -Force

Install-Package -Name docker -ProviderName DockerMsftProvider -Force

Restart-Computer -Force

Achtung

Achten Sie darauf, dass nach dem Neustart des Servers zusätzlich der Docker-Dienst gestartet werden muss. Rufen Sie dazu das Dienste-Fenster über »services.msc« im Suchfeld des Startmenüs auf und nehmen Sie die entsprechende Einstellung vor.

Anschließend steht der Server bereit und Sie können mit Containern arbeiten. Administratoren, die Docker mit PowerShell DSC installieren wollen, können folgende Befehle verwenden:

Install-Script -Name Install-DockerOnWS2016UsingDSC

Install-DockerOnWS2016UsingDSC.ps1

Tipp

Für Entwickler und Administratoren kann es interessant sein, Hyper-V für Container-Hosts auch in Windows 10 zu nutzen. Dazu sind auf dem Windows 10-Host einige Befehle notwendig:

Netsh advfirewall firewall add rule name="docker engine" dir=in action=allow protocol=TCP localport=2375

Stop-Service docker

Dockerd --unregister-service

Dockerd -H npipe://-H 0.0.0.0:2375 --register-service

Start-Service docker

Erste Schritte mit Docker in Windows Server 2016

Der Befehl *Docker images* zeigt zum Beispiel die vorhandenen Docker-Images auf dem Windows-Server an. Standardmäßig sind noch keine Images vorhanden.

Tip

Erhalten Sie bei der Ausführung des Docker-Befehls eine Fehlermeldung, dass die Authentifizierung fehlt, müssen Sie sich zuerst mit *Docker login* mit Ihrer Docker-ID anmelden. Eine Docker-ID erhalten Sie auf der Internetseite von Docker (<http://tinyurl.com/jrjfs0j>).

Um ein Image auf Basis von Windows Server 2016 zur Verfügung zu stellen, können Sie die notwendigen Daten direkt bei Microsoft/Docker herunterladen:

```
Docker pull microsoft/windowsservercore
```

Alternativ stehen folgende Befehle zur Verfügung:

```
Docker pull microsoft/windowsservercore:10.0.14393.321
```

```
Docker tag microsoft/windowsservercore:10.0.14393.321 microsoft/windowsservercore
```

Sie können als Image für Docker-Container in Windows Server 2016, neben der Core-Installation, auch eine Nano-Installation verwenden. In diesem Fall geben Sie den folgenden Befehl ein:

```
Docker pull microsoft/nanoserver
```

Wollen Sie einen Container erstellen und starten, verwenden Sie den Befehl *Docker run*. Der Befehl startet das festgelegte Image als Container. So können Sie sicherstellen, dass das Image funktioniert.

```
Docker run microsoft/windowsservercore
```

Tip

Mit dem Docker-Client durchsuchen Sie den Docker-Hub nach Images auf Basis von Windows Server 2016. Dazu verwenden Sie zum Beispiel den Befehl:

```
Docker search Microsoft
```

Auch ein Webserver auf Basis der Internetinformationsdienste (IIS) in Windows Server 2016 lässt sich als Container bereitstellen:

```
Docker run -it -p 80:80 microsoft/iis cmd
```

Nach dem Start steht der Webserver mit der Standardwebseite bereit. Um die Standardwebseite im Container zu löschen, verwenden Sie zum Beispiel:

```
Del C:\inetpub\wwwroot\iisstart.htm
```

Wollen Sie die Startseite mit einer eigenen Seite ersetzen, verwenden Sie den folgenden Befehl:

```
Echo "Test für IIS im Windows Server-Container" > C:\inetpub\wwwroot\index.html
```

Sobald der Container erstellt wurde, können Sie ihn über die Eingabeaufforderung und die PowerShell verwalten. Um sich eine Liste aller Container auf einem Container-Host anzuzeigen, verwenden Sie den Befehl

```
Docker ps -a
```

Tipp

Die installierte Docker-Version und den Docker-Client lassen Sie sich mit *Docker version* anzeigen.

Verschiedene Images für Core und Nano nutzen

Welche Images Sie auf einem Container-Host nutzen können, hängt davon ab, auf welcher Installationsvariante von Windows Server 2016 Sie den Container-Host betreiben.

Auf Servern mit grafischer Oberfläche und Core-Servern können Sie das Core-Image und das Nano-Image von Windows Server 2016 verwenden. Auf Nano-Servern steht bei Windows-Servern nur das Nano-Image zur Verfügung, als Hyper-V-Container können Sie aber auch die Core-Server-Variante verwenden.

Windows Server-Container und der Host teilen sich einen einzelnen Kernel, da die Container den Kernel des Container-Hosts nutzen. Dabei muss das Basisimage des Containers mit dem des Hosts übereinstimmen. Windows Server 2016 kennt hier vier Versionierungsgrade: Hauptversion, Nebenversion, Build und die Revision, zum Beispiel »10.0.14393.0«. Die Revisionsnummer wird aktualisiert, wenn Windows-Updates installiert werden. Das Starten von Windows Server-Containern wird verhindert, wenn die Buildnummer nicht übereinstimmt. Das kann zum Beispiel passieren, wenn Sie Vorabversionen von Windows Server 2016 einsetzen oder aktualisierte Container nutzen. Wenn die Buildnummer übereinstimmt, die Revisionsnummer aber unterschiedlich ist, wird der Container zwar gestartet, allerdings ist der produktive Betrieb nicht empfohlen und wird von Microsoft auch nicht unterstützt.

Sie können in der Registry im Pfad `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion` erkennen, welche Version auf einem Container-Host installiert ist. Stellen Sie sicher, dass die Tags auf *Docker Hub* oder die Image-Hash-Tabelle in der Beschreibung des Image zu der Version des Hosts passt.

Hinweis

Hyper-V-Container verwenden eine eigene Instanz des Windows-Kernels. Daher müssen die Versionen von Container-Host und Container-Image nicht übereinstimmen.

Hyper-V-Container-Host anpassen

Wollen Sie Hyper-V-Container nutzen, benötigen Sie auf dem Container-Host natürlich noch Hyper-V als Serverrolle. Hier haben Sie auch die Möglichkeit, mit einer VM zu arbeiten. In diesem Fall müssen Sie aber für die VM die eingebettete (nested) Virtualisierung konfigurieren. Damit das funktioniert, müssen Sie auf dem Hyper-V-Host, auf dem Sie den Container/Hyper-V-Host betreiben, einige Anpassungen in der PowerShell vornehmen (siehe auch Kapitel 7):

```
#replace with the virtual machine name
```

```
$vm = "<virtual-machine>"
```

```
#configure virtual processor
```

```
Set-VMProcessor -VMName $vm -ExposeVirtualizationExtensions $true -Count 2
#disable dynamic memory
Set-VMMemory $vm -DynamicMemoryEnabled $false
#enable mac spoofing
Get-VMNetworkAdapter -VMName $vm | Set-VMNetworkAdapter -MacAddressSpoofing
On
```

In der virtuellen Maschine, die Sie als Container-Host für Docker und Hyper-V-Container nutzen wollen, können Sie dann noch Hyper-V über die PowerShell installieren:

```
Install-WindowsFeature hyper-v
```

Nano-Server als Container-Host verwenden

Windows Server-Container auf Basis von Docker, aber auch die Hyper-V-Container lassen sich auch auf Nano-Servern betreiben. Sie haben hier zusätzlich die Möglichkeit, den Nano-Server zu virtualisieren. Wollen Sie auf dem Nano-Server auch Hyper-V-Container nutzen, müssen Sie in diesem Fall die eingebettete Virtualisierung aktivieren, so wie im vorangegangenen Abschnitt erläutert. In den Kapiteln 2 bis 4 und 7 sind wir bereits auf diese Themen eingegangen. Generell kann es sinnvoll sein, den Nano-Server in die Domäne mit aufzunehmen.

Eine Remote-PowerShell-Sitzung mit dem Nano-Server erstellen

Um die Container auf einem Nano-Server zu verwalten, sollten Sie am besten über eine RDP-Sitzung des Hyper-V-Hosts, auf dem Sie den Nano-Server virtualisieren, eine Remote-PowerShell-Sitzung zum Nano-Server aufbauen.

Fügen Sie im ersten Schritt den Nano-Server zu den vertrauenswürdigen Hosts auf dem Hyper-V-Host hinzu. Dazu verwenden Sie die IP-Adresse des Nano-Servers. Diese erfahren Sie auch über die Nano Server Recovery Konsole (siehe die Kapitel 2 und 3).

Hinweis In diesem und den folgenden Beispielen hat der Nano-Server die IP-Adresse 192.168.178.225, der Server trägt die Bezeichnung *nano-hyperv* und ist Mitglied der Domäne *joos.int*.

```
Set-Item WSMAN:\localhost\Client\TrustedHosts 192.168.178.225 -Force
```

Danach erstellen Sie die Remote-PowerShell-Sitzung:

```
Enter-PSSession -ComputerName 192.168.178.225 -Credential joos\Administrator
```

Alle Befehle, die Sie jetzt eingeben, werden auf dem Nano-Server ausgeführt.

Windows-Updates auf Nano-Servern installieren

Damit Container auf Nano-Servern funktionieren, müssen Sie die neuesten Updates für Windows Server 2016 installieren. Das gilt natürlich genauso für Core-Server und Server mit grafischer Benutzeroberfläche. Auf Nano-Servern ist die Installation von Windows-Updates teilweise etwas komplizierter. Geben Sie zur Installation von Windows-Updates die folgenden Befehle in einer Remote-PowerShell-Sitzung ein:

```
$sess = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName MSFT_WUOperationsSession
```

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

Nachdem alle Updates installiert sind, können Sie den Nano-Server über die Remote-PowerShell-Sitzung neu starten:

```
Restart-Computer -Force
```

Docker auf Nano-Servern installieren

Auch auf Nano-Servern müssen Sie Docker installieren, um Windows Server-Container zu nutzen. Verbinden Sie sich dazu nach der Installation der neuesten Updates mit einer Remote-PowerShell-Sitzung und installieren Sie Docker:

```
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
```

```
Install-Package -Name docker -ProviderName DockerMsftProvider
```

Nachdem die Installation abgeschlossen ist, starten Sie den Nano-Server neu:

```
Restart-Computer -Force
```

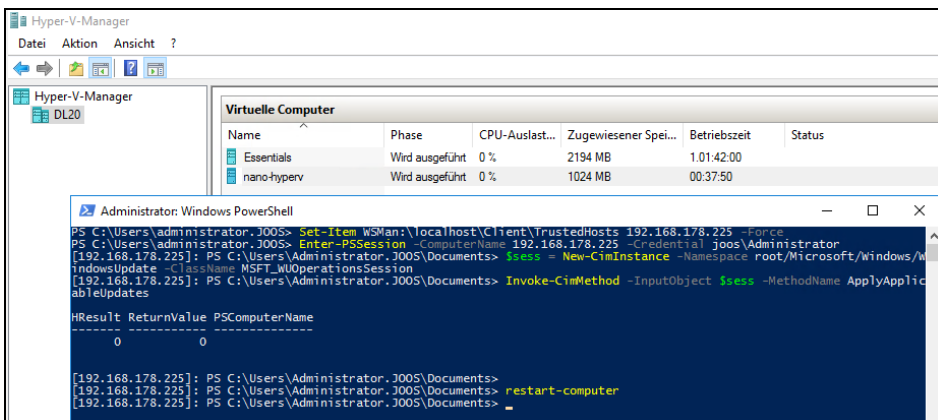


Abbildung 26.1: Installieren von Windows-Updates auf Nano-Servern über eine Remote-PowerShell-Sitzung

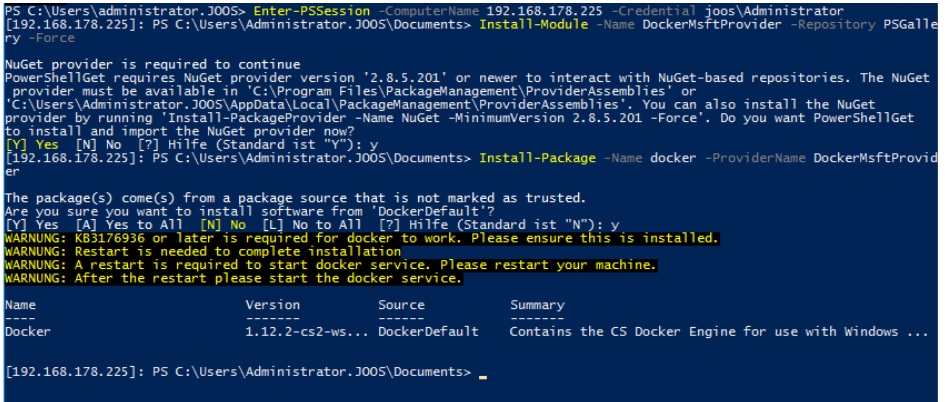
Basis-Container-Images auf dem Nano-Server integrieren

Um ein Container-Image auf Nano-Servern zu installieren, verwenden Sie den folgenden Befehl:

```
Docker pull microsoft/nanoserver
```

Wollen Sie auch Hyper-V-Container verwenden, müssen Sie Hyper-V auf Ihrem Nano-Server installieren. Wie Sie dazu vorgehen, zeigen wir in den Kapiteln 2 und 7. Anschließend können Sie das Container-Image für Core-Server auf den Nano-Server herunterladen:

```
Docker pull microsoft/windowsservercore
```



```
PS C:\Users\Administrator.J005> Enter-PSSession -ComputerName 192.168.178.225 -Credential joos\Administrator
[192.168.178.225]: PS C:\Users\Administrator.J005\Documents> Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories. The NuGet provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or 'C:\Users\Administrator.J005\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by running 'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet to install and import the NuGet provider now?
[Y] Yes [N] No [?] Hilfe (Standard ist "Y"): y
[192.168.178.225]: PS C:\Users\Administrator.J005\Documents> Install-Package -Name docker -ProviderName DockerMsftProvider

The package(s) come(s) from a package source that is not marked as trusted.
Are you sure you want to install software from 'DockerDefault'?
[Y] Yes [A] Yes to All [N] No [L] No to All [?] Hilfe (Standard ist "N"): y
WARNING: KB3176936 or later is required for docker to work. Please ensure this is installed.
WARNING: Restart is needed to complete installation.
WARNING: A restart is required to start docker service. Please restart your machine.
WARNING: After the restart please start the docker service.

Name                Version          Source            Summary
----                -
Docker               1.12.2-cs2-ws... DockerDefault     Contains the CS Docker Engine for use with Windows ...

[192.168.178.225]: PS C:\Users\Administrator.J005\Documents> _
```

Abbildung 26.2: Docker installieren Sie über die PowerShell des Hyper-V-Hosts auf dem virtuellen Nano-Server.

Besonderheiten beim Betrieb von Docker unter Nano-Server

Für den Betrieb von Docker auf einem Nano-Server sind einige Besonderheiten zu berücksichtigen.

Erstellen Sie auf dem Nano-Server eine Firewallregel für die Docker-Verbindung. Bei unsicheren Verbindungen wird Port 2375 verwendet, bei sicheren Verbindungen Port 2376. Auch diese Befehle geben Sie wieder in der Remote-PowerShell-Sitzung ein:

```
Netsh advfirewall firewall add rule name="Docker daemon " dir=in action=allow protocol=TCP localport=2375
```

Erstellen Sie eine `daemon.json`-Datei auf dem Nano-Server-Host:

```
New-Item -Type File c:\ProgramData\docker\config\daemon.json
```

Geben Sie den folgenden Befehl ein, um die entsprechenden Daten in die Datei einzutragen:

```
Add-Content 'c:\programdata\docker\config\daemon.json' '{"hosts": ["tcp://0.0.0.0:2375", "npipe://"]}'
```

Starten Sie den Docker-Dienst neu:

```
Restart-Service docker
```


Einen Docker-Client installieren

Um auf dem Hyper-V-Host, auf dem Sie den Nano-Server installiert haben, die Container zu verwalten, benötigen Sie den Docker-Client auch auf dem Hyper-V-Host. Dazu geben Sie folgende Befehle in der PowerShell ein:

```
Invoke-WebRequest "https://download.docker.com/components/engine/windows-server/cs-1.12/docker.zip" -OutFile "$env:TEMP\docker.zip" -UseBasicParsing
```

```
Expand-Archive -Path "$env:TEMP\docker.zip" -DestinationPath $env:ProgramFiles
```


```
$env:path += ";c:\program files\docker"
```

```
[Environment]::SetEnvironmentVariable("Path", $env:Path + ";C:\Program Files\Docker", [EnvironmentVariableTarget]::Machine)
```

Anschließend können Sie über den Hyper-V-Host und den Docker-Client auf die Docker-Installation des Nano-Servers zugreifen:

```
Docker -H tcp://<IP-Adresse des Nano-Servers>:2375 run -it microsoft/nanoserver cmd
```

Tip

Mit dem Befehl *Install-Module posh-docker* laden Sie die automatische Vervollständigung für den Docker-Client auf einen Rechner. Nach der Installation können Sie mit der -Taste durch die Befehle und Optionen des Docker-Clients schalten. In der PowerShell importieren Sie das Modul mit *Import-Module posh-docker*.

Hyper-V-Container auf Nano-Servern nutzen

Auf Nano-Servern können Sie Container mit dem Container-Image auf Basis des Nano-Image erstellen. Sie können dazu nicht das Core-Image verwenden. Setzen Sie aber auf Hyper-V-Container, können Sie auch das Core-Image als Vorlage für Windows Server-Container verwenden. Dazu müssen Sie jedoch auf dem Nano-Server zuvor Hyper-V installiert haben. Den Befehl können Sie ebenfalls in einer Remote-PowerShell-Sitzung durchführen:

```
Install-NanoServerPackage Microsoft-NanoServer-Compute-Package
```

Auch hier müssen Sie den Nano-Server danach neu starten:

```
Restart-Computer -Force
```

Erweiterte Konfiguration von Containern durchführen

Sobald Sie den Container-Host installiert und gestartet haben, können Sie mit Docker bereits Container-Images bei Microsoft herunterladen und starten. Auf Basis von Containern lassen sich schnell und einfach eigene Images erstellen. Sie können auch in den Containern Serveranwendungen installieren und bereitstellen.

Container erstellen und Serverdienste verwalten

Docker ermöglicht auch die lokale Verwaltung der Serverrollen. Erstellen Sie zum Beispiel mit dem folgenden Befehl einen neuen Container und wechseln durch Hinzufügen der Option *-it* direkt in die Eingabeaufforderung, können Sie innerhalb des Containers die PowerShell starten und Installationen vornehmen:

```
Docker run -it --name winiis -p 80:80 microsoft/windowsservercore
```

Sobald sich die Eingabeaufforderung des Containers öffnet, können Sie mit dem Befehl *Powershell* auf dem Container eine lokale PowerShell-Sitzung öffnen.

Anschließend prüfen Sie zunächst, ob die Internetinformationsdienste (IIS) auf dem Container installiert sind. Dazu verwenden Sie den gleichen Befehl wie bei herkömmlichen Servern mit Windows Server 2016:

```
Get-WindowsFeature web-server
```

Um IIS zu installieren, verwenden Sie wiederum den folgenden Befehl:

```
Install-WindowsFeature web-server
```

Sobald IIS im Container installiert ist, können Sie über den Befehl *Ipconfig* die IP-Adresse des Containers anzeigen lassen und zum Beispiel vom Container-Host aus mit dem Internet Explorer auf die IP-Adresse des Containers zugreifen. Da IIS installiert ist und Sie den Port 80 auf dem Container aktiviert haben, wird die IIS-Startseite angezeigt.

Tipp

Mit *Docker inspect <ID>* können Sie erweiterte Informationen für Container sowie die IP-Adresse des Containers abrufen.

Container und eigene Images erstellen

Auch eigene Images können erstellt und bearbeitet werden. Dies erfolgt zum Beispiel auf Basis bestehender Container, die Sie wiederum mit *Docker ps -a* anzeigen lassen:

```
Docker commit <ID> <Ordner>/meincontainerimage
```

Beispiel:

```
Docker commit 662f25d6d835 windowsiis/joosimageiis
```

In Docker können Sie also auch Container mit bereits installierten Anwendungen als neues Image speichern und dieses Image für neue Container verwenden. Ob das Image erstellt wurde, können Sie mit *Docker images* anzeigen lassen.

Um Container zu löschen, verwenden Sie den Befehl *Docker rm <Name des Containers>*, der Befehl *Docker rmi <Name des Image>* löscht Docker-Images.

Um zum Beispiel einen Container mit IIS zur Verfügung zu stellen und auf dessen Basis weitere Images anzulegen, müssen Sie zunächst einen neuen Container erstellen, der auf dem vorgefertigten Image basiert:

```
Docker run -d -p 80:80 microsoft/iis ping -t localhost
```

Über den Befehl können Sie auch direkt die Ports aktivieren (*-p*) und sicherstellen, dass die Internetinformationsdienste als Dienst gestartet werden (*-d*). Alle gestarteten Container sehen Sie mit *Docker ps*. Nehmen Sie Änderungen an einem Container vor, können Sie

diesen Container zum Beispiel als neues Image speichern und auf Basis dieses Image weitere Container. Dazu verwenden Sie den Befehl *Docker ps -a*, um sich den Namen des Containers anzuzeigen. Anschließend erstellen Sie das Image mit dem folgenden Befehl:

```
Docker commit <ID> <Neuer Name>
```

Dockerfiles für eigene Images erstellen

Auf Basis dieses Image erstellen Sie jederzeit weitere Container. Der Vorgang lässt sich automatisieren, indem Sie ein sogenanntes »Dockerfile« verwenden. Dabei handelt es sich um eine Anweisungsdatei für neue Container.

Erstellen Sie dazu ein Verzeichnis auf dem Host und legen Sie darin eine Datei *Dockerfile* (ohne Dateiendung) an. Sie können den Vorgang zum Beispiel mit der PowerShell durchführen:

```
Powershell New-Item c:\build\Dockerfile -Force
```

Die Automatisierung nehmen Sie über Befehle in der Datei vor. Dazu müssen Sie die Datei *Dockerfile* in Notepad öffnen:

```
Notepad c:\build\Dockerfile
```

In der Datei können Sie zum Beispiel festlegen, dass ein neues Image erstellt werden soll, das IIS als Basis nutzt. In der Datei können Sie auch bestimmen, dass Änderungen an der Konfiguration vorgenommen werden:

```
FROM microsoft/iis
```

```
RUN echo "Dockerfile-Test für automatische Bereitstellung" > c:\inetpub\wwwroot\index.html
```

Generell können Sie bei Dockerfiles mit der Anweisung *FROM* festlegen, auf welcher Basis der neue Container erstellt werden soll, zum Beispiel mit:

```
FROM windowsservercore
```

Mit *RUN* legen Sie fest, was im neuen Container-Image vorgenommen werden soll. Sie können zum Beispiel mit dem folgenden Befehl die Internetinformationsdienste (IIS) in einem neuen Container-Image installieren:

```
RUN dism.exe /online /enable-feature /all /featurename:iis-webserver /NoRestart
```

Wollen Sie das Visual Studio-Paket in einem Container installieren, verwenden Sie diesen Aufruf:

```
RUN start-Process c:\vcredist_x86.exe -ArgumentList '/quiet' -Wait
```

Tip

Sie können über ein Dockerfile auch PowerShell-Skripts in ein Container-Image kopieren und ausführen, zum Beispiel mit:

```
FROM windowsservercore
```

```
ADD script.ps1 /windows/temp/script.ps1
```

```
RUN powershell.exe -executionpolicy bypass c:\windows\temp\script.ps1
```

Um auf Basis dieser Änderungen wiederum ein Image zu erstellen, verwenden Sie in diesem Beispiel diesen Befehlsaufruf:

Docker build -t iis-dockerfile c:\Build

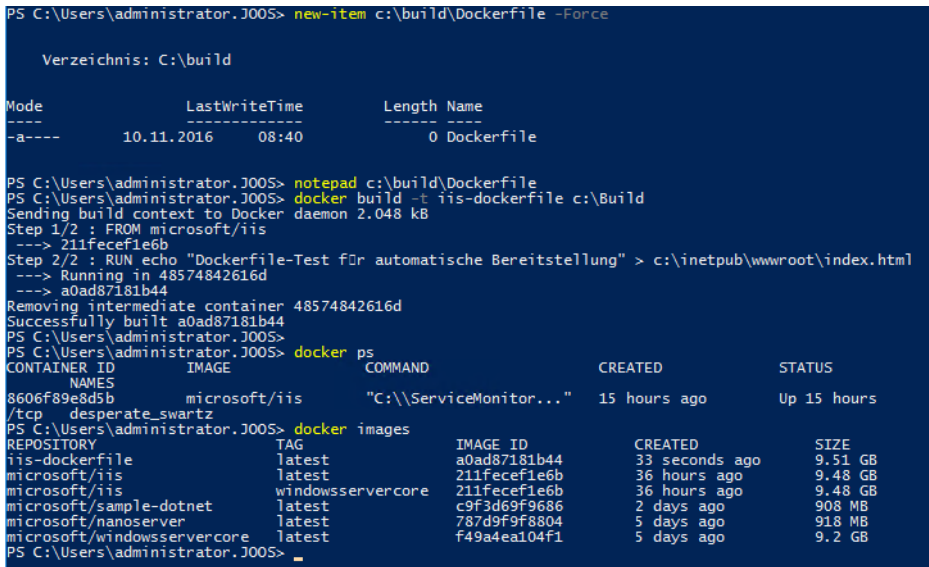


Abbildung 26.3: Docker-Container erstellen und verwalten Sie am besten in der PowerShell.

Sie können erstellte Container mit von Ihnen vorgenommenen Änderungen jederzeit als neues Image speichern und dieses Image wiederum für neue Container verwenden. So erstellen Sie also sehr schnell zahlreiche Container mit allen benötigten Einstellungen. Um ein Image zu erstellen, verwenden Sie zum Beispiel den folgenden Befehl:

Docker commit <ID> meincontainerimage

Sobald Sie das Image erstellt haben, können Sie es mit dem Befehl *Docker images* anzeigen lassen und als Grundlage für einen neuen Container verwenden:

Docker run -it --name dockertest2 meincontainerimage cmd

Container in die Cloud laden (Docker Push)

Mit dem Befehl *Docker pull* laden Sie Container-Images aus Ihrem Docker-Konto auf den Container-Host, um auf Basis des Image einen neuen Container zu erstellen. Sie können aber auch den umgekehrten Weg gehen und Images in Ihr Cloud-Konto hochladen. Der Vorteil dabei ist, dass Sie dieses Image jederzeit wieder herunterladen und auch auf anderen Container-Hosts verwenden können. Sie benötigen dazu eine Docker-ID und müssen sich mit *Docker login* anmelden.

Zum Hochladen von Container-Images verwenden Sie den folgenden Befehl:

Docker push <Benutzername>/iis-dockerfile

Nach dem Upload können Sie mit *Docker pull* das Image auf Container-Hosts herunterladen. Wollen Sie das Image nicht mehr verwenden, können Sie es löschen:

Docker rmi <Benutzername>/iis-dockerfile

Hyper-V-Container in Windows Server 2016 anlegen

Betreiben Sie Docker-Container mit Windows Server 2016 innerhalb von Hyper-V als spezielle Hyper-V-Container, schottet das Betriebssystem diese noch mehr ab als herkömmliche Windows Server-Container auf Basis von Docker. Das erhöht die Sicherheit und Stabilität.

Hyper-V-Container werden – ebenso wie virtuelle Server (siehe Kapitel 7) – über virtuelle Switches an das Netzwerk angebunden. Auch Hyper-V-Container bauen auf Docker auf, bieten aber mehr Möglichkeiten zur Erstellung von Containern.

Der Vorteil der Hyper-V-Container ist eine effizientere Isolierung sowie eine Optimierung der Umgebung für Hyper-V. Hyper-V-Container sind immer von anderen Containern und dem Host isoliert. Da Windows Server-Container Teile des Betriebssystems mit dem Host teilen, besteht das Problem, dass ein Container einen ganzen Host und andere Container beeinträchtigen kann. Mit Hyper-V-Containern ist das nicht möglich, da das Betriebssystem isoliert und virtualisiert wird. Dies ermöglicht es, Container mit Anwendungen auszuführen, die in »Lower Trust«-Umgebungen für Angriffe anfällig sind. Beispielsweise würde dies auf Webserver zutreffen.

Hyper-V-Container verstehen

Windows Server-Container teilen sich wichtige Bereiche des Betriebssystems mit dem Host und anderen Containern. Dadurch erhöht sich zwar im Vergleich zu virtuellen Servern die Effizienz der Container, bietet aber auch mögliche Angriffsflächen. Grundsätzlich ist es möglich, dass ein Container andere Docker-Container auf dem Host beeinträchtigt oder angreift. Der Nachteil von Hyper-V-Containern ist eine etwas schlechtere Leistung im Vergleich zu Windows Server-Container. Der Vorteil liegt in der besseren Isolierung der Container. Sie können auch Freigaben des Hosts in Hyper-V-Containern nutzen, zum Beispiel für die Datenspeicherung oder für Installationsmedien. Die Verwaltung von Hyper-V-Containern kann wie bei herkömmlichen Containern über die PowerShell oder die Eingabeaufforderung erfolgen.

In Hyper-V-Containern ist eine eigene Kopie des Betriebssystems integriert. Der Container läuft in einer Art eingeschränkter virtueller Maschine. Zusammen mit Nano-Servern lassen sich dadurch schnelle und sichere Container zur Verfügung stellen, die alle Vorteile von Windows Server 2016 nutzen. Windows Server-Container, Hyper-V-Container und Nano-Server können gemeinsam und parallel eingesetzt werden.

Sie können in Hyper-V-Containern auch Rechte delegieren, zum Beispiel für mandanten-gestützte Systeme. Die Hyper-V-Container eines Mandanten können miteinander kommunizieren, während die Container der anderen Mandanten abgeschottet sind. Die Abschottung der Gruppen erfolgt durch Hyper-V in Windows Server 2016. Hyper-V-Container lassen sich per Hyper-V-Replikation auf andere Hyper-V-Hosts replizieren und mit Hyper-V-Clustern hochverfügbar betreiben. Die Übertragung von Hyper-V-Containern auf andere Knoten mit der Livemigration ist ebenfalls möglich.

Hinweis

Container-Images müssen nicht angepasst werden, um sie auch als Hyper-V-Container zu nutzen. Images für Container lassen sich für herkömmliche Container, aber auch für Hyper-V-Container nutzen. Sie benötigen also keine verschiedenen Images für die unterschiedlichen Einsatzgebiete.

Bei Bedarf können Sie Windows Server-Container mit wenigen Schritten zu Hyper-V-Container konvertieren. Auch der umgekehrte Weg ist jederzeit möglich. Hyper-V-Container können Sie jederzeit wieder in herkömmliche Container konvertieren. Einstellungen und Daten gehen dabei nicht verloren.

Arbeiten Sie mit einem Nano-Server als Container-Host, können Sie in diesem Hyper-V-Container aktivieren. Nach der Aktivierung verfügt der Container über eine virtuelle Hardware. Diese können Sie in der PowerShell des Servers mit `Get-PnpDevice` anzeigen lassen. In Hyper-V-Containern werden Netzwerkadapter und SCSI-Adapter als Hyper-V-Hardware angezeigt.

Hyper-V-Container erstellen und konfigurieren

Haben Sie Container erstellt, können Sie sie mit der PowerShell und dem Docker-Client verwalten. Hier gibt es zunächst keine Unterschiede zwischen Hyper-V-Containern und Windows Server-Containern. Beim Erstellen eines Hyper-V-Containers mit Docker wird der Parameter `--isolation=hyperv` verwendet.

Wollen Sie einen herkömmlichen Container mit Docker zu einem Hyper-V-Container konvertieren, setzen Sie eine Isolierungsmarkierung. Der Befehl sieht dann zum Beispiel folgendermaßen aus:

```
Docker run --rm -it --isolation=hyperv nanoserver cmd
```

Die Vorteile lassen sich an einem Beispiel zeigen. Erstellen Sie mit dem folgenden Befehl einen Container und lassen darin einen dauerhaften Ping-Befehl laufen, ist der Prozess auf dem Host selbst zu erkennen:

```
Docker run -d Microsoft/windowsservercore ping localhost -t
```

Der erfolgreich erstellte Container wird mit `Docker ps` angezeigt. Mit `Docker top <Name des Containers>` lassen Sie sich die Prozesse im Container anzeigen. Den Namen sehen Sie mit `Docker ps`.

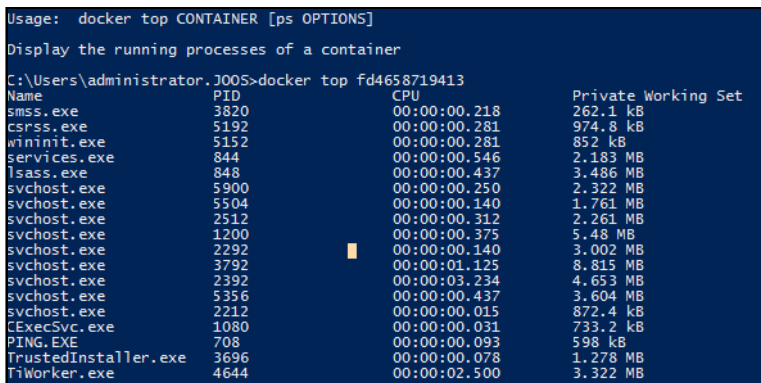


Abbildung 264: Lassen Sie sich die im Container ablaufenden Prozesse anzeigen.

In diesem Beispiel sehen Sie anschließend den Ping-Prozess und dessen ID. Mit dem Befehl *Get-Process -Name ping* lassen Sie sich diese Informationen anzeigen. Dadurch ist zu erkennen, dass der Prozess über die gleiche ID wie im Container verfügt.

Alternativ können Sie einen isolierten Hyper-V-Container mit der Option *--isolation* erstellen:

```
Docker run -d --isolation=hyperv microsoft/nanoserver ping -t localhost
```

Auch hier lässt sich jetzt auf dem gleichen Weg die ID des Prozesses für den Ping-Befehl abrufen. Dazu verwenden Sie wieder *Docker top*. Suchen Sie erneut nach dem Prozess auf dem Host, ist dieser allerdings nicht zu sehen. Auf dem Host wird in diesem Fall aber der Prozess einer neuen VM sichtbar. Dabei handelt es sich um den virtuellen Computer, der den Hyper-V-Container kapselt und die ausgeführten Prozesse vor dem Hostbetriebssystem schützt.

Docker, Hyper-V-Container und VMs parallel einsetzen

Neben herkömmlichen Windows Server-Containern und Hyper-V-Containern, können Sie dann auf einem Hyper-V-Host (auch auf einem Nano-Server) virtuelle Maschinen erstellen, die wiederum mit den Containern kommunizieren können. Container-Host und Hyper-V-Host schließen sich also nicht aus.

Herkömmliche Installationen von Windows Server 2016 arbeiten mit Containern und Hyper-V-Containern zusammen, genauso wie Core- oder Nano-Installationen von Windows Server 2016. Die Server und Dienste lassen sich in einem gemeinsamen Netzwerk betreiben, auch zusammen mit anderen Betriebssystemen wie Windows Server 2012/2012 R2 oder Linux.

Windows Server-Container in der PowerShell verwalten

Container können Sie recht einfach über die PowerShell verwalten. Das gilt auch für lokale Installationen von Container-Hosts. Mit dem Befehl *Powershell* starten Sie in der Eingabeaufforderung eine neue PowerShell-Sitzung. Alle Befehle für die Verwaltung von Containern lassen Sie sich mit dem Befehl *Get-Command -Module Containers* auflisten.

Nachdem ein Container gestartet ist, können Sie eine PowerShell-Sitzung öffnen und sich mit dem Container verbinden. Dadurch verwalten Sie auch Einstellungen und Serverdienste im Container. Für den Verbindungsaufbau benötigen Sie die ID des Containers. Diese können Sie zum Beispiel mit *Docker ps* herausfinden.

Für den Verbindungsaufbau verwenden Sie den Befehl *Enter-PSSession*. Zusammen mit der Container-ID sowie der Option *RunAsAdministrator* bauen Sie eine Verbindung auf. Der Container hat eine eigene IP-Adresse erhalten, damit er mit dem Netzwerk/Internet kommunizieren kann. Die Syntax des Befehls sieht folgendermaßen aus:

```
Enter-PSSession -ContainerId <ID> -RunAsAdministrator
```

Befehle, die Sie hier eingeben, werden im Container durchgeführt. Mit *Exit* verlassen Sie die Sitzung im Container und arbeiten wieder mit dem eigentlichen Container-Host. Bei der Erstellung neuer Container spielen auch die virtuellen Switches auf dem Host eine Rolle. Diese können Sie in der PowerShell mit *Get-VMSwitch* anzeigen. Container verbinden sich über die virtuellen Switches mit dem Netzwerk.

Sie können in Containern Sitzungen unterbrechen und erneut aufbauen. Bei unterbrochenen Sitzungen laufen die Cmdlets in der Sitzung weiter. Dazu nutzen Sie die Cmdlets *Disconnect-PSSession*, *Connect-PSSession* und *Receive-PSSession*.

Wollen Sie von einer lokalen PowerShell-Sitzung über das Netzwerk Programme auf einem Container starten, verwenden Sie den folgenden Befehl:

```
Invoke-Command -ContainerId <ID> -RunAsAdministrator -ScriptBlock { <Befehl> } -RunAsAdministrator
```

Ein Beispiel für die Ausführung ist:

```
Invoke-Command -ContainerId b2f55c8c-28d7-4c0c-ab2b-9ee62c9ae6ea -RunAsAdministrator -ScriptBlock { ipconfig } -RunAsAdministrator
```

Zusammenfassung

In diesem Kapitel haben Sie erfahren, wie Sie die neuen Container in Windows Server 2016 nutzen sowie Hyper-V-Container einsetzen. Auch die Installation von Container-Hosts war Thema in diesem Kapitel.

Im nächsten Kapitel erläutern wir Ihnen, wie Sie den Webserver IIS in Windows Server 2016 nutzen. Dieser lässt sich auch in Containern betreiben, aber auch auf Nano-Servern, auf Core-Servern und ebenso auf herkömmlichen Servern mit Windows Server 2016.