



Craig Zacker

Installation, Speicher- technologien und Computing mit Windows Server 2016

Original Microsoft Prüfungstraining

70-740

dpunkt.verlag

Inhalt

Cover

Titel

Impressum

Inhaltsverzeichnis

Einführung

Aufbau dieses Buchs

Microsoft-Zertifizierungen

Kostenlose E-Books von Microsoft Press

Microsoft Virtual Academy

Schneller Zugriff auf Onlinematerialien

Errata, Aktualisierung und Support für dieses Buch

Wir möchten von Ihnen hören

Bleiben Sie am Ball

Wichtig: Wie Sie dieses Buch beim Lernen für die Prüfung einsetzen

Kapitel 1 Installieren von Windows Servern in Host- und Computingumgebungen

Prüfungsziel 1.1: Server und Arbeitsauslastungen installieren, aktualisieren und migrieren

Installationsanforderungen von Windows Server 2016 bestimmen

Angemessene Windows Server 2016-Editionen gemäß Arbeitsauslastung bestimmen

Windows Server 2016 installieren

Windows Server 2016-Features und -Rollen installieren

Windows Server Core installieren und konfigurieren

Windows Server Core-Installationen mit Windows PowerShell, Befehlszeile und Remoteverwaltungsmöglichkeiten verwalten

Windows PowerShell Desired State Configuration (DSC) implementieren, um die Integrität installierter Umgebungen einzurichten und aufrechtzuerhalten

Aktualisierungen und Migrationen von Servern und Kernarbeitsauslastungen von Windows Server 2008 und Windows Server 2012 nach Windows Server 2016 durchführen

Das angemessene Aktivierungsmodell für die Serverinstallation bestimmen

Prüfungsziel 1.2: Nanoserver installieren und konfigurieren

Angemessene Nutzungsszenarien und Anforderungen für Nanoserver bestimmen

Nanoserver installieren

Rollen und Features auf Nanoserver implementieren

Nanoserver verwalten und konfigurieren

Nanoserver mit Windows PowerShell ferngesteuert verwalten

Prüfungsziel 1.3: Images für die Bereitstellung erstellen, verwalten und pflegen

Windows Server-Virtualisierung planen

Linux- und FreeBSD-Bereitstellungen planen

Virtualisierungsarbeitsauslastungen mit dem MAP-Toolkit bewerten

Überlegungen für die Bereitstellung von Arbeitsauslastungen in virtualisierten Umgebungen bestimmen

Images mit Patches, Hotfixes und Treibern aktualisieren

Rollen und Features in Offline-Images installieren

Windows Server Core, Nanoserver-Images und VHDs mit Windows PowerShell verwalten und warten

Kapitelzusammenfassung

Kapitel 2 Implementieren von Speicherlösungen

Prüfungsziel 2.1: Datenträger und Volumes konfigurieren

Konfigurieren von Sektorgrößen, die für verschiedene Workloads geeignet sind

Konfigurieren von GUID-Partitionstabellen(GPT)-Disketten

VHD- und VHDX-Dateien mit Server-Manager oder Windows PowerShell-Speichermodul-Cmdlets erstellen

Virtuelle Festplatten bereitstellen

Festlegen, wann NTFS- und ReFS-Dateisysteme verwendet werden sollen

Konfigurieren der SMB-Freigabe und Sitzungseinstellungen über Windows PowerShell

Konfigurieren von SMB-Server- und SMB-Client-Konfigurationseinstellungen mithilfe von Windows PowerShell

Konfigurieren von Datei- und Ordnerberechtigungen

Prüfungsziel 2.2: Serverspeicher implementieren

Speicherpools konfigurieren

Layoutoptionen für einfachen Speicher, für Speicher mit Spiegelung und für Speicher mit Parität für Datenträger oder Anlagen implementieren

Speicherpools erweitern

Mehrstufigen Speicher konfigurieren

iSCSI-Ziel und -Initiator konfigurieren

iSNS konfigurieren

Datacenter Bridging (DCB) konfigurieren

Multipfad-E/A (MPIO) konfigurieren

Nutzungsszenarien für Speicherreplikate bestimmen

Speicherreplikate für Server-zu-Server-, Cluster-zu-Cluster- und Stretched-Cluster-Szenarien implementieren

Prüfungsziel 2.3: Datendeduplizierung implementieren

Deduplizierung implementieren und konfigurieren

Angemessene Nutzungsszenarien für Deduplizierung bestimmen

Deduplizierung überwachen

Eine Sicherungs- und Wiederherstellungslösung mit Deduplizierung implementieren

Kapitelzusammenfassung

Kapitel 3 Implementieren von Hyper-V

Prüfungsziel 3.1: Hyper-V installieren und konfigurieren

Hardware- und Kompatibilitätsanforderungen für die Installation von Hyper-V bestimmen

Hyper-V installieren

Verwaltungstools installieren

Von vorhandenen Hyper-V-Versionen aktualisieren

Verwaltung virtueller Computer delegieren

Remoteverwaltung von Hyper-V-Hosts durchführen

Virtuelle Computer mit Windows SharePoint Services Direct konfigurieren

Geschachtelte Virtualisierung implementieren

Prüfungsziel 3.2: Einstellungen für virtuelle Computer konfigurieren

Einen virtuellen Computer erstellen

Beim Ausführen eines virtuellen Computers Arbeitsspeicher hinzufügen oder entfernen

Dynamischen Arbeitsspeicher konfigurieren

Dynamische Speicherzuordnungen

NUMA-Unterstützung konfigurieren

Smart Paging konfigurieren

Ressourcenmessung konfigurieren

Integrationsdienste verwalten

Virtuelle Computer der Generation 1 und 2 erstellen und konfigurieren und angemessene Nutzungsszenarien bestimmen

Erweiterten Sitzungsmodus implementieren

Virtuelle Computer für Linux und FreeBSD erstellen

Linux Integration Services (LIS) installieren und konfigurieren

FreeBSD Integration Services installieren und konfigurieren

Sicheren Start für Windows- und Linux-Umgebungen implementieren

Virtuelle Computer aus früheren Versionen von Hyper-V zu Windows Server 2016 Hyper-V übertragen und umwandeln

Virtuelle Computer exportieren und importieren

Discrete Device Assignment (DDA) implementieren

Prüfungsziel 3.3: Hyper-V-Speicher konfigurieren

VHDs und VHDX-Dateien mit Hyper-V-Manager erstellen

Freigegebene VHDX-Dateien erstellen

Differenzierende Datenträger konfigurieren

Virtuelle Festplatten ändern

Pass-Through-Datenträger konfigurieren

Größe einer virtuellen Festplatte ändern

Prüfpunkte verwalten

Produktionsprüfpunkte implementieren

Einen virtuellen Fibre Channel-Adapter implementieren

Quality of Service konfigurieren

Prüfungsziel 3.4: Hyper-V-Netzwerk konfigurieren

Virtuelle Netzwerkschnittstellenkarten hinzufügen und entfernen

Virtuelle Hyper-V-Switches konfigurieren

Netzwerkleistung optimieren

MAC-Adressen konfigurieren

Netzwerkisolation konfigurieren

Legacy- und synthetische virtuelle Netzwerkadapter konfigurieren

NIC-Teamvorgang auf virtuellen Computern konfigurieren

Warteschlange für virtuelle Computer konfigurieren

RDMA auf Netzwerkadaptern aktivieren, die unter Verwendung von SET an einen virtuellen Hyper-V-Switch gebunden sind

Bandbreitenverwaltung konfigurieren

Kapitelzusammenfassung

Kapitel 4 Implementieren von Windows-Container

Prüfungsziel 4.1: Windows-Container bereitstellen

Installationsanforderungen und angemessene Szenarien für Windows-Container bestimmen

Windows Server-Containerhost in physischen oder virtualisierten Umgebungen installieren und konfigurieren

Windows Server-Containerhost auf Windows Server Core oder Nanoserver in einer physischen oder virtualisierten Umgebung installieren und konfigurieren

Docker auf Windows Server und Nanoserver installieren

Docker Daemon-Startoptionen konfigurieren

Windows PowerShell für die Verwendung mit Containern konfigurieren

Ein Basisbetriebssystem installieren

Ein Image markieren

Ein Betriebssystem-Image deinstallieren

Windows Server-Container erstellen

Prüfungsziel 4.2: Windows-Container verwalten

Windows- oder Linux-Container mit dem Docker-Daemon verwalten

Windows- oder Linux-Container mithilfe von Windows PowerShell verwalten

Containernetzwerke verwalten

Container-Datenvolumes verwalten

Ressourcensteuerung verwalten

Neue Container-Images mit Dockerfile erstellen

Container-Images mit DockerHub-Repository für öffentliche und private Szenarien verwalten

Container-Images mit Microsoft Azure verwalten

Kapitelzusammenfassung

Kapitel 5 Hochverfügbarkeit implementieren

Prüfungsziel 5.1: Hochverfügbarkeits- und Notfallwiederherstellungsoptionen in Hyper-V implementieren

Hyper-V-Replikat implementieren

Livemigration implementieren

Shared Nothing-Livemigration implementieren

CredSSP- oder Kerberos-Authentifizierungsprotokoll für Livemigration konfigurieren

Speichermigration implementieren

Prüfungsziel 5.2: Failoverclustering implementieren

Arbeitsgruppe, Cluster für einfache und mehrere Domänen implementieren

Quorum konfigurieren

Clusternetzwerk konfigurieren

Einzelne Knoten oder Clusterkonfiguration wiederherstellen

Clusterspeicher konfigurieren

Clusterfähiges Aktualisieren implementieren

Paralleles Upgrade für Clusterbetriebssysteme implementieren

Freigegebene Clustervolumen konfigurieren und optimieren

Cluster ohne Netzwerknamen konfigurieren

Dateiserver mit horizontaler Skalierung implementieren

Verschiedene Szenarien für die Verwendung von SoFS statt eines gruppierten Dateiservers bestimmen

Nutzungsszenarien für die Implementierung von Gastclustering bestimmen

Eine Clusterspeicherplätze-Lösung mit freigegebenen SAS-Speicheranlagen implementieren

Speicherreplikat implementieren

Cloudzeugen implementieren

VM-Resilienz implementieren

Freigegebenes VHDX als eine Speicherlösung für Gastcluster implementieren

Prüfungsziel 5.3: »Direkte Speicherplätze« implementieren

Szenarioanforderungen für die Implementierung von »Direkte Speicherplätze« bestimmen

»Direkte Speicherplätze« mit Windows PowerShell aktivieren

Ein verteiltes »Direkte Speicherplätze«-Szenario in einem Cluster implementieren

Ein hyperkonvergiertes »Direkte Speicherplätze«-Szenario in einem Cluster implementieren

Prüfungsziel 5.4: Failovercluster verwalten

Rollenspezifische Einstellungen einschließlich ständig verfügbarer Freigaben konfigurieren

VM-Überwachung konfigurieren

Failover und Einstellungen konfigurieren

Stretch- und standortabhängige Failovercluster implementieren

Node Fairness aktivieren und konfigurieren

Prüfungsziel 5.5: Umzug virtueller Computer in Clusterknoten verwalten

Eine Livemigration durchführen

Eine Schnellmigration durchführen

Eine Speichermigration durchführen

Virtuelle Computer importieren, exportieren und kopieren

Integrität von Netzwerken virtueller Computer konfigurieren

Belastung beim Herunterfahren konfigurieren

Prüfungsziel 5.6: Netzwerklastenausgleich implementieren

NLB-Voraussetzungen konfigurieren

NLB-Knoten installieren

Affinität konfigurieren

Portregeln konfigurieren

Clusterbetriebsmodus konfigurieren

NLB-Cluster aktualisieren

Kapitelzusammenfassung

Kapitel 6 Serverumgebungen verwalten und überwachen

Prüfungsziel 6.1: Serverinstallationen verwalten

WSUS-Lösungen implementieren

WSUS-Gruppen konfigurieren

Patchverwaltung in gemischten Umgebungen

Eine Antimalwarelösung mit Windows Defender implementieren

Windows Defender mit WSUS und Windows Update integrieren

Sicherungs- und Wiederherstellungsvorgänge mit Windows Server-Sicherung durchführen

Sicherungsstrategien für verschiedene Windows Server-Rollen und -Arbeitsauslastungen bestimmen, darunter Hyper-V-Host, Hyper-V-Gäste, Active Directory, Dateiserver und Webserver mit Windows Server 2016-eigenen Tools und Lösungen

Prüfungsziel 6.2: Serverinstallationen überwachen

Arbeitsauslastungen mit Leistungsüberwachung überwachen

Datensammlersätze konfigurieren

Geeignete Leistungsindikatoren für Prozessor, Arbeitsspeicher, Datenträger und Netzwerk für Speicher- und Computing-Arbeitsauslastungen bestimmen

Warnungen konfigurieren

Arbeitsauslastungen mit Ressourcenmonitor überwachen

Kapitelzusammenfassung

Index

Implementieren von Windows-Container

Container sind ein Hilfsmittel, um virtualisierte, isolierte Betriebssystemumgebungen für die Bereitstellung und Ausführung von Anwendungen schnell bereitstellen zu können. Windows Server 2016 unterstützt Container in Zusammenarbeit mit einem Open-Source-Container-Modul namens Docker.

Dieses Kapitel befasst sich mit folgenden Prüfungszielen:

- Windows-Container bereitstellen
- Windows-Container verwalten

Prüfungsziel 4.1: Windows-Container bereitstellen

Seit den frühen Tagen von Windows ist Virtualisierung ein wichtiges Schlagwort gewesen. Virtueller Speicher ist schon seit Jahrzehnten gebräuchlich; Windows kann Festplattenplatz so nutzen, dass das System scheinbar mehr Arbeitsspeicher besitzt als tatsächlich vorhanden ist. Hyper-V virtualisiert Hardware, erstellt dabei Computer innerhalb eines Computers, die scheinbar ihre eigenen Prozessoren, Arbeitsspeicher und Festplatten haben, obwohl sie in Wirklichkeit die Ressourcen des Hostservers gemeinsam nutzen. Container sind ein neues Feature in Windows Server 2016, das Betriebssysteme virtualisiert.

In diesem Abschnitt geht es um folgende Themen:

- Installationsanforderungen und angemessene Szenarien für Windows-Container bestimmen
- Windows Server-Containerhost in physischen oder virtualisierten Umgebungen installieren und konfigurieren
- Windows Server-Containerhost auf Windows Server Core oder Nanoserver in einer physischen oder virtualisierten Umgebung installieren und konfigurieren
- Docker auf Windows-Server und Nanoserver installieren

- Docker Daemon-Startoptionen konfigurieren
- Windows PowerShell für die Verwendung mit Containern konfigurieren
- ein Basisbetriebssystem installieren
- ein Image markieren
- ein Betriebssystem-Image deinstallieren
- Windows Server-Container erstellen
- Hyper-V-Container erstellen

Installationsanforderungen und angemessene Szenarien für Windows-Container bestimmen

So wie virtuelle Computer gewissermaßen Abbilder von separaten Computern darstellen, bieten Container das, was wie separate Instanzen des Betriebssystems erscheint, wobei jeder Container mit eigenem Speicher und Dateisystem ausgestattet ist und eine saubere, neue Kopie des Betriebssystems ausführt. Im Unterschied zu virtuellen Computern allerdings, die separate Kopien des Betriebssystems ausführen, nutzen Container das Betriebssystem des Hostsystems gemeinsam. Weder ist es erforderlich, eine separate Instanz des Betriebssystems für jeden Container zu erstellen, noch führt der Container eine Bootsequenz aus, lädt Bibliotheken oder teilt den Betriebssystemdateien Arbeitsspeicher zu. Container starten sekundenschnell und Sie können mehr Container auf einem Hostsystem einrichten als virtuelle Computer.

Benutzer, die mit Containern arbeiten, sehen zunächst eine saubere Betriebssysteminstallation, die für Anwendungen bereitsteht. Die Umgebung ist vollkommen vom Host getrennt und auch von anderen Containern, was mit Namespace-Isolierung und Ressourcenkontrolle realisiert wird.

Namespace-Isolierung heißt, dass jeder Container nur auf die Ressourcen zugreifen darf, die für ihn verfügbar sind. Dateien, Ports und laufende Prozesse scheinen alle dem Container fest zugeordnet zu sein, selbst wenn sie mit dem Host und mit anderen Containern gemeinsam genutzt werden. Die Arbeitsumgebung sieht wie die eines virtuellen Computers aus, doch im Unterschied zu einem virtuellen Computer, der separate Kopien aller Betriebssystemdateien verwaltet, nutzt ein Container diese Dateien mit dem Host gemeinsam und kopiert sie nicht. Nur dann, wenn ein Benutzer oder eine

Anwendung eine Datei in einem Container ändert, wird eine Kopie im Dateisystem des Containers angelegt.

Ressourcenkontrolle bedeutet, dass ein Container nur auf eine bestimmte Menge von Prozessorzyklen, Systemspeicher, Netzwerkbandbreite und anderer Ressourcen zugreifen darf, nicht auf mehr. Eine Anwendung, die in einem Container läuft, findet eine saubere Sandbox-Umgebung vor, ohne Zugriff auf Ressourcen, die anderen Containern oder dem Host zugeordnet sind.

Container-Images

Da sich neue Container in Sekunden erstellen lassen und jeder Container vollkommen isoliert ist, sind Container eine ideale Plattform für die Anwendungsentwicklung und das Testen von Software. Es gibt aber noch weitere Vorteile.

Container basieren auf Images. Um einen neuen Container zu erstellen, laden Sie ein Image aus einem Repository herunter und führen es aus. Wenn Sie ein Image von Windows Server 2016 Server Core ausführen, bekommen Sie einen Container mit einer sauberen Instanz des Betriebssystems, das in ihm läuft. Alternativ können Sie Windows Server-Images mit Rollen oder Anwendungen herunterladen, zum Beispiel mit Internet Information Services (IIS) oder Microsoft SQL Server, die bereits installiert und ausführungsbereit sind.

Das grundlegende Betriebssystemimage ändert sich niemals. Wenn Sie eine Anwendung im Container installieren und dann ein neues Image erstellen, enthält das resultierende Image nur die Dateien und Einstellungen, die für die Ausführung der Anwendung erforderlich sind. Natürlich ist das neue Image, das Sie erstellt haben, relativ klein, weil es nicht das gesamte Betriebssystem enthält. Um die Anwendung mit anderen Benutzern zu teilen, brauchen Sie ihnen nur das neue, kleinere Image zu schicken, sofern sie bereits das Basisbetriebssystemimage besitzen.

Dieser Vorgang kann sich über so viele Iterationen wie nötig fortsetzen, mit Schicht auf Schicht von Images auf dieser ursprünglichen Basis. Dies kann in einer äußerst effizienten Softwareentwicklungsumgebung resultieren. Anstatt riesige VHD-Dateien übertragen oder ständig neue virtuelle Computer erstellen und installieren zu müssen, brauchen Sie nur kleine Container-Images zu übertragen, die ohne Hardwarekompatibilitätsprobleme laufen.

Windows Server-Containerhost in physischen oder virtualisierten Umgebungen installieren und konfigurieren

Windows Server 2016 unterstützt zwei Arten von Containern: Windows Server-Container und Hyper-V-Container. Der Unterschied zwischen den beiden liegt im Grad der Containerisolierung, die sie bieten. Windows Server-Container arbeiten im Benutzermodus und nutzen alles mit dem Hostcomputer gemeinsam, Betriebssystemkernel und Systemspeicher eingeschlossen.

Aus diesem Grund ist es vorstellbar, dass eine Anwendung ob versehentlich oder bewusst in der Lage sein könnte, aus den Grenzen ihres Containers auszubrechen und andere Prozesse zu beeinflussen, die auf dem Host oder in anderen Containern laufen. Diese Option ist deshalb vorzuziehen, wenn die in verschiedenen Containern laufenden Anwendungen grundsätzlich vertrauenswürdig sind.

Hyper-V-Container bieten eine zusätzliche Isolationsstufe, indem sie mit dem Hypervisor eine separate Kopie des Betriebssystemkerns für jeden Container erstellen. Obwohl sie für die manuelle Verwaltung weder sichtbar noch zugänglich sind, erstellt Hyper-V virtuelle Computer, die Windows-Container enthalten, wobei sie die Basiscontainer-Images verwenden, wie Abbildung 4-1 zeigt. Die Containerimplementierung ist praktisch die gleiche, der Unterschied zeigt sich in den Umgebungen, in denen die beiden Arten von Containern existieren.

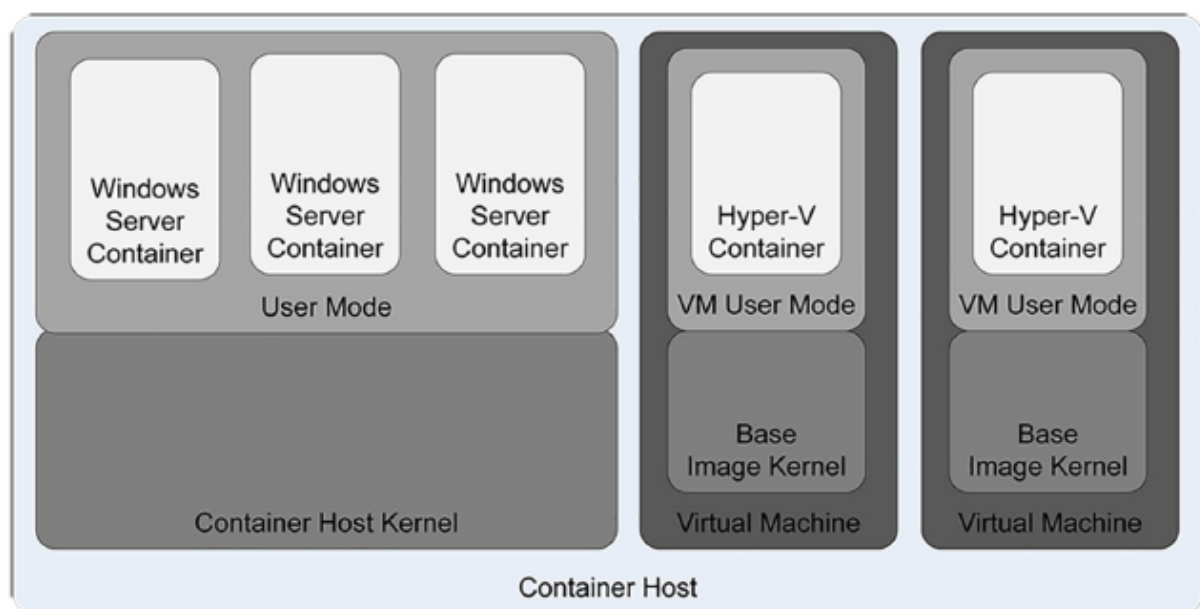


Abb. 4-1 Architektur von Windows-Containern

Weil Hyper-V-Container in einem virtuellen Computer existieren, verfügen sie über einen eigenen zugewiesenen Arbeitsspeicher sowie über isolierten Speicher und Netzwerk-E/A. Dies ergibt eine Container-Umgebung, die sich für das eignet, was Microsoft als »feindliche mehrinstanzfähige« Anwendungen bezeichnet, beispielsweise eine Situation, in der ein Unternehmen Container für Kunden bereitstellt, in denen sie ihren eigenen Code ausführen können, was nicht als vertrauenswürdig gilt. Somit bietet Windows Server 2016 mit den hinzugefügten Hyper-V-Containern drei Isolationsebenen, die von der getrennten Betriebssysteminstallation von virtuellen Hyper-V-Computern über die Hyper-V-Container mit getrenntem Kernel und Arbeitsspeicher bis hin zu Windows Server-Containern mit freigegebenen Kernels und anderen Ressourcen reichen.

Einen Containerhost installieren

Windows Server 2016 bringt ein Feature namens *Container* mit, das Sie installieren müssen, um Container zu unterstützen. Um aber Container zu erstellen und zu verwalten, müssen Sie Docker herunterladen und installieren. Das ist die Anwendung, die das Feature unterstützt.

Das Feature *Container* installieren Sie mit dem Assistenten zum Hinzufügen von Rollen und Features. Auf der Seite *Features auswählen* aktivieren Sie das Kontrollkästchen *Container* (siehe Abbildung 4-2).

HINWEIS Windows Server-Installation

Um Windows Server-Container zu erstellen, muss das Hostbetriebssystem auf dem Laufwerk C des Computers installiert sein, was standardmäßig gegeben ist. Die gemeinsame Nutzung des Betriebssystemkernels ist dadurch einfacher. Für das Erstellen von Hyper-V-Containern ist dies nicht erforderlich, weil der Hypervisor dafür zuständig ist, jedem Container eine Kopie des Kernels bereitzustellen.

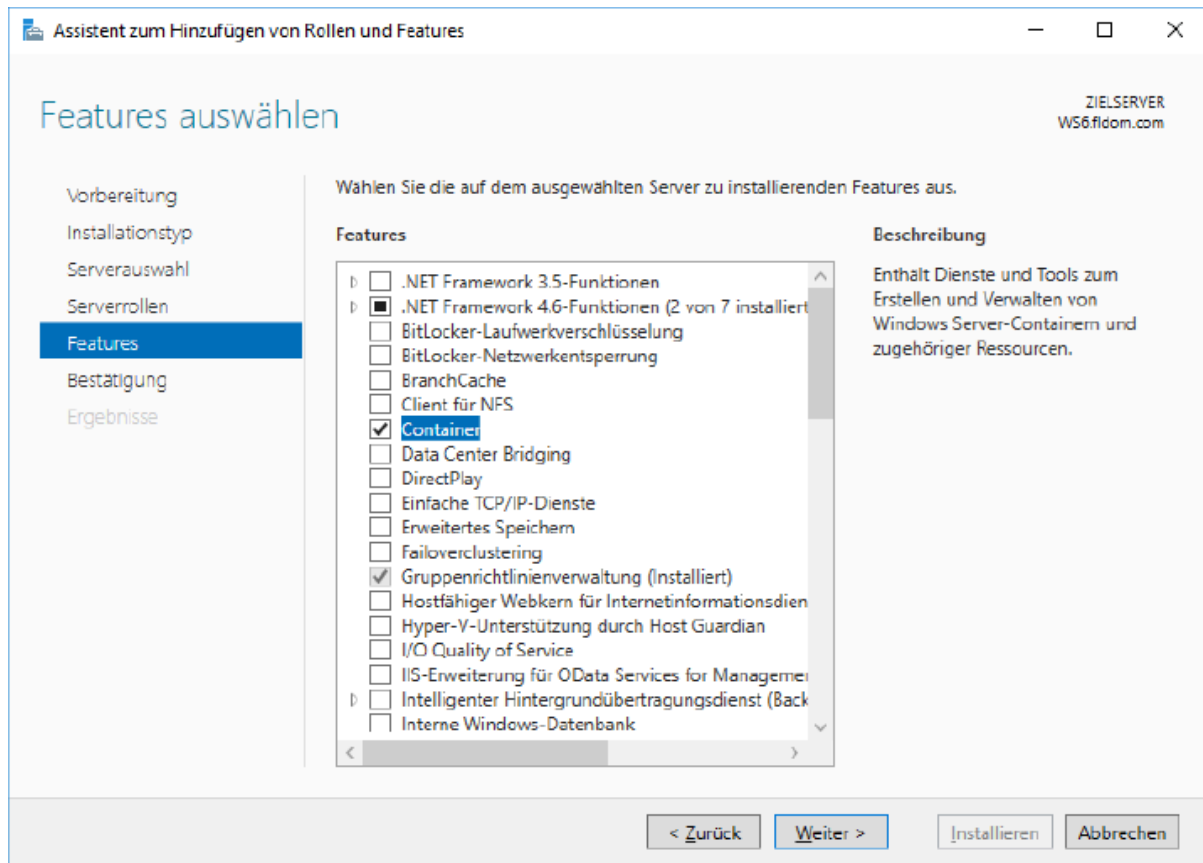


Abb. 4-2 Das Feature *Container* im Assistenten zum Hinzufügen von Rollen und Features installieren

Um Hyper-V-Container zu erstellen, müssen Sie sowohl das Feature *Container* als auch die Rolle *Hyper-V* installieren. Selbst wenn Sie für die Container keine virtuellen Computer erstellen, installiert die Rolle *Hyper-V* den Hypervisor. Dieser ist erforderlich, um die separate Kopie des Windows-Kernels für jeden Hyper-V-Container zu erstellen.

Die allgemeinen Hardwareanforderungen der Hyper-V-Rolle gehen über die Anforderungen des Betriebssystems Windows Server 2016 selbst hinaus. Bevor Sie die Hyper-V-Rolle auf einem Server unter Windows Server 2016 installieren können, muss folgende Hardware vorhanden sein:

- Ein 64-Bit-Prozessor mit hardwareseitiger Unterstützung der Virtualisierung und Second Level Address Translation (SLAT). Eine derartige Virtualisierung ist in Prozessoren vorhanden, die eine Virtualisierungsoption beinhalten, beispielsweise Intel-Virtualisierungstechnologie (Intel VT-x) oder AMD-Virtualisierung (AMD-V).
- Hardwaregestützte Datenausführungsverhinderung (Data Execution Prevention, DEP), bei Intel als eXecute Disable (XD) und bei AMD als No

eXecute (NX) bezeichnet. CPUs verwenden diese Technik, um Speicherbereiche für Prozessoranweisungen von Speicherbereichen für Daten zu trennen. Bei Intel-Prozessoren muss das XD-Bit (eXecute Disable) aktiviert werden, bei AMD-Prozessoren das NX-Bit (No eXecute).

- Erweiterungen für den VM-Überwachungsmodus, bei Intel-Prozessoren als VT-c bezeichnet.
- Ein System-BIOS oder UEFI, das die Virtualisierungshardware unterstützt und auf dem die Virtualisierungsfunktion aktiviert wurde.

Wenn Sie die Hyper-V-Rolle mit dem Hyper-V-Manager installieren, fordert der Assistent zum Hinzufügen von Rollen und Features Sie auf, auch die Hyper-V-Verwaltungstools zu installieren. Erstellen Sie Hyper-V-Container, aber keine virtuellen Hyper-V-Computer, ist es nicht erforderlich, die Verwaltungstools zu installieren.

Container virtualisieren

Windows Server 2016 unterstützt die Verwendung von Containern in virtuellen Hyper-V-Computern. Das Feature *Container* und die Docker-Dateien können Sie in jedem virtuellen Computer installieren. Allerdings muss das System die Anforderungen für geschachtelte Virtualisierung erfüllen, damit sich auf einem virtuellen Computer Hyper-V-Container erstellen lassen.

Um einen geschachtelten Hyper-V-Hostserver zu erstellen, müssen sowohl der physische Host als auch der virtuelle Computer, auf dem Sie die Hyper-V-Container erstellen, Windows Server 2016 ausführen. Der virtuelle Computer kann mit den Installationsoptionen Desktopdarstellung, Server Core oder Nanoserver ausgeführt werden. Des Weiteren ist im physischen Host ein Intel-Prozessor mit VT-x und Unterstützung für Extended Page Tables (EPT) erforderlich.

Bevor Sie Hyper-V auf dem virtuellen Computer installieren, müssen Sie seinem virtuellen Prozessor die Virtualisierungstechnologie auf dem physischen Computer zugänglich machen. Dazu ist es erforderlich, den virtuellen Computer herunterzufahren und auf dem physischen Host in einer PowerShell-Sitzung einen Befehl wie im folgenden Beispiel auszuführen:

```
set-vmprocessor -vmname server1
```

```
-exposevirtualizationextensions $true
```

Darüber hinaus sind die folgenden Konfigurationsänderungen auf dem virtuellen Computer, der als Hyper-V-Host fungiert, vorzunehmen (angegeben zuerst als Ort im Dialogfeld *Einstellungen* des virtuellen Computers im Hyper-V-Manager und anschließend als PowerShell-Befehl):

- Auf der Seite *Arbeitsspeicher* geben Sie dem virtuellen Computer mindestens 4 GB RAM und deaktivieren das Kontrollkästchen *Dynamischen Arbeitsspeicher aktivieren*.

```
set-vmmemory -vmname server1
```

```
-startupbytes 4gb
```

```
-dynamicmemoryenabled $false
```

- Auf der Seite *Prozessor* setzen Sie *Anzahl virtueller Prozessoren* auf 2.

```
set-vmprocessor -vmname server1
```

```
-count 2
```

- Auf der Seite *Netzwerkkarte/Erweiterte Features* aktivieren Sie das Kontrollkästchen *Spoofing von MAC-Adressen aktivieren*.

```
set-vmnetworkadapter -vmname server1
```

```
-name "network adapter"
```

```
-macaddressspoofing on
```

Nach diesen Änderungen können Sie den virtuellen Computer starten, die Rolle *Hyper-V* installieren und mit Docker fortfahren, um Hyper-V-Container zu

erstellen.

Windows Server-Containerhost auf Windows Server Core oder Nanoserver in einer physischen oder virtualisierten Umgebung installieren und konfigurieren

Ein Computer, der mit der Option *Server Core* installiert wurde, kann als Containerhost fungieren. Die Anforderungen sind die gleichen wie für einen Server, der mit der Option *Desktopdarstellung* installiert wurde, außer dass Sie die erforderlichen Features entweder per Befehlszeile installieren oder das System remote verwalten müssen.

Nachdem Sie in eine PowerShell-Sitzung gewechselt sind, können Sie das Feature *Container* und die Rolle *Hyper-V* mit dem folgenden Befehl installieren:

```
install-windowsfeature -name containers, hyper-v
```

Nanoserver als Containerhost konfigurieren

Die Windows Server 2016-Installationsoption *Nanoserver* unterstützt sowohl Windows Server-Container als auch Hyper-V-Container. Die Nanoserver-Implementierung enthält Pakete, die das Feature *Container* und die Rolle *Hyper-V* unterstützen. Beides können Sie hinzufügen, wenn Sie ein Nanoserver-Image mit dem Windows PowerShell-Cmdlet *New-NanoServerImage* erzeugen, wie das folgende Beispiel zeigt:

```
new-nanoserverimage -deploymenttype guest
```

```
-edition datacenter
```

```
-mediapath d:\
```

```
-targetpath c:\nano\nano1.vhdx
```

```
-computername nano1
```

-domainname contoso

-containers

Dieser Befehl erzeugt ein Nanoserver-Image mit folgenden Eigenschaften:

- **deploymenttype guest** Erzeugt ein Image für die Verwendung auf einem virtuellen Hyper-V-Computer.
- **edition datacenter** Erzeugt ein Image mit der Datacenter-Edition von Windows Server.
- **mediapath d:** Greift auf die Nanoserver-Quelldateien von Laufwerk *D:* zu.
- **targetpath c:\nano\nano1.vhdx** Erzeugt eine VHDX-Imagedatei im Ordner *C:\nano* mit dem Namen *Nano1.vhdx*.
- **computername nano1** Weist dem Nanoserver den Computernamen *Nano1* zu.
- **domainname contoso** Veranlasst den Beitritt des Computers in die Domäne *Contoso*.
- **containers** Installiert das Feature *Container* als Teil des Images.
- **compute** Installiert die Rolle *Hyper-V* als Teil des Images.

Wenn Sie vorhaben, Hyper-V-Container auf dem Gast-Nanoserver zu erstellen, müssen Sie ihm Zugriff auf die Virtualisierungsfunktionen des Hyper-V-Servers bieten. Das bewerkstelligen Sie in folgenden Schritten:

1. Erstellen Sie einen neuen virtuellen Computer mit der von Ihnen erstellten Nanoserver-Imagedatei. Starten Sie ihn aber noch nicht.
2. Gewähren Sie dem virtuellen Computer auf dem Hyper-V-Hostserver Zugriff auf die Virtualisierungsfunktionen des physischen Prozessors im Hyper-V-Server mit einem Befehl wie dem folgenden:

```
set-vmprocessor -vmname nano1
```

```
-exposevirtualizationextensions $true
```

3. Starten Sie den virtuellen Nanoserver-Computer.

Wenn der virtuelle Nanoserver-Computer läuft, müssen Sie eine Remote-PowerShell-Sitzung von einem anderen Computer aus einrichten, damit Sie den Nanoserver-Computer verwalten können. Führen Sie dazu einen Befehl wie den folgenden auf dem Computer aus, mit dem Sie Nanoserver verwalten:

```
enter-pssession -computername nano1  
  
-credential
```

HINWEIS Nanoserver-Remoteverwaltung

Dieser Abschnitt geht davon aus, dass sich der Nanoserver in einem Netzwerk befindet, in dem ein DHCP-Server die TCP/IP-Einstellungen zuweist und er erfolgreich einer AD DS-Domäne beigetreten ist. Wenn dies nicht der Fall ist, müssen Sie die TCP/IP-Einstellungen für den Nanoserver von seiner Konsole aus manuell konfigurieren und dann den Nanoserver zur Liste der vertrauenswürdigen Hosts auf dem Computer hinzufügen, mit dem Sie ihn verwalten.

Docker auf Windows Server und Nanoserver installieren

Docker ist ein Open-Source-Tool, das seit Jahren die Containerfunktionalität für die Linux-Community realisiert. Da es nun portiert wurde, können Sie die gleiche Funktionalität in Windows implementieren. Docker besteht aus zwei Dateien:

- **Dockerd.exe** Das Docker-Modul, auch als Dienst oder Dämon bezeichnet, läuft im Hintergrund auf dem Windows-Computer.
- **Docker.exe** Der Docker-Client, eine Befehlsshell, in der Sie Container erstellen und verwalten.

Zusätzlich zu diesen beiden Dateien, die Sie herunterladen und installieren müssen, um Container zu erstellen, enthält Docker auch die folgenden Ressourcen:

- **Dockerfiles** Skriptdateien mit Anweisungen für das Erstellen von Container-Images.
- **Docker Hub** Eine cloudbasierte Registrierung, die es Docker-Benutzern ermöglicht, sowohl auf Image- und Code-Repositories zu verweisen als auch eigene Images zu erstellen und zu speichern.
- **Docker Cloud** Ein cloudbasierter Dienst, mit dem Sie Ihre Containeranwendungen bereitstellen können.

Docker auf Windows-Server installieren

Da Docker ein Open-Source-Produkt ist, gehört es nicht zum Lieferumfang von Windows Server 2016. Auf einem Computer, auf dem Windows Server 2016 mit der Option Desktopdarstellung oder Server Core installiert ist, müssen Sie Docker herunterladen und installieren, bevor Sie Container erstellen können. Docker lässt sich mithilfe von *OneGet* herunterladen, einem cloud-basierten Paketmanager für Windows.

Damit Sie auf OneGet zugreifen können, müssen Sie das Modul *DockerMsftProvider* mit dem folgenden Befehl installieren. Wenn die Aufforderung erscheint, einen NuGet-Anbieter zu installieren, antworten Sie mit (oder drücken).

```
install-module -name dockermstprovider

-repository psgallery

-force
```

Das Cmdlet *Install-Module* lädt das angeforderte Modul herunter und installiert es im Ordner *C:\Program Files\Windows PowerShell\Modules*, wo es von jeder PowerShell-Eingabeaufforderung aus zugänglich ist. Als Nächstes führen Sie den folgenden *Install-Package*-Befehl aus, um Docker herunterzuladen und zu installieren. Wenn die Abfrage erscheint, ob Sie das als nicht vertrauenswürdig gekennzeichnete Paket installieren möchten, antworten Sie mit .

```
install-package -name docker
```

```
-providername dockermsftprovider
```

Nachdem die Docker-Dateien heruntergeladen sind, wird *Dockerd.exe* als Windows-Dienst registriert und der Client *Docker.exe* in den Pfad aufgenommen, sodass sich das Programm von jedem Ort im Dateisystem ausführen lässt.

Nachdem die Installation abgeschlossen ist, starten Sie den Computer mit dem folgenden Befehl neu:

```
restart-computer -force
```

Docker auf Nanoserver installieren

Wenn Sie einer PowerShell-Remotesitzung mit einem Nanoserver-Computer beigetreten sind, können Sie Docker mit den gleichen Befehlen wie auf einem Computer mit den Installationsoptionen Desktopdarstellung oder Server Core installieren. Microsoft empfiehlt allerdings, dass Sie den Docker-Client vom Remotesystem aus ausführen, nachdem Sie den *Dockerd*-Dienst auf dem Nanoserver installiert haben.

Hierfür müssen Sie die folgenden Aufgaben fertigstellen:

- 1. Eine Firewallregel erstellen.** Damit Nanoserver den Docker-Clientverkehr in das System lässt, müssen Sie eine neue Firewallregel erstellen, die Port 2375 für TCP-Verkehr öffnet. Führen Sie dazu den folgenden Befehl in der Nanoserver-Sitzung aus:

```
netsh advfirewall firewall
```

```
add rule name="docker daemon"
```

```
dir=in
```

```
action=allow
```

```
protocol=tcp localport=2375
```

- 2. Das Dockerd-Modul konfigurieren, um Netzwerkverkehr zu akzeptieren.** Docker hat seine Wurzeln in Linux und, wie die meisten Linux-Anwendungen, verwendet auch Docker Textdateien für die Konfiguration. Um das *Dockerd*-Modul in die Lage zu versetzen, Clientverkehr über das Netzwerk zu akzeptieren, erstellen Sie auf dem Nanoserver im Verzeichnis *C:\ProgramData\Docker* eine Textdatei namens *daemon.json*, die die folgende Zeile enthält:

```
{ "hosts": ["tcp://0.0.0.0:2375", "npipe://"] }
```

Die beiden folgenden PowerShell-Befehle legen die neue Datei an und fügen den erforderlichen Text ein:

```
new-item -type file  
c:\programdata\docker\config\daemon.json
```

```
add-content 'c:\programdata\docker\config\daemon.json'
```

```
'{ "hosts":["tcp://0.0.0.0:2375", "npipe://"] }'
```

- 3. Das Dockerd-Modul neu starten.** Nachdem Sie die Datei *daemon.json* erstellt haben, müssen Sie das *Dockerd*-Modul mit dem folgenden Befehl neu starten:

```
restart-service docker
```

- 4. Den Docker-Client herunterladen.** Um das *Dockerd*-Modul remote zu verwalten, müssen Sie den *Docker.exe*-Client auf dem Remotesystem (nicht innerhalb der Nanoserver-Sitzung) herunterladen und installieren.

Öffnen Sie hierzu einen Browser und geben Sie die folgende URL ein, um das Docker-Paket herunterzuladen:

```
https://download.docker.com/components/engine/windows-server/cs-1.12/docker.zip
```

5. In PowerShell lässt sich dies mit dem folgenden Befehl bewerkstelligen:

```
invoke-webrequest "https://download.docker.com/
components/engine/windows-server/cs-1.12/docker.zip"
-outfile "$env:temp\docker.zip"
-usebasicparsing
```

6. **Docker.exe installieren.** Wenn Sie die *Docker.zip*-Datei über einen Browser heruntergeladen haben, legen Sie einen Ordner *C:\ProgramData\Docker* an, extrahieren die Datei *Docker.exe* aus dem ZIP-Archiv und kopieren Sie in diesen Ordner, um die Anwendung zu installieren. Diese Schritte führen Sie in PowerShell mit dem folgenden Befehl aus:

```
expand-archive -path "$env:temp\docker.zip"
-destinationpath $env:programfiles
```

7. **Die Umgebungsvariable PATH festlegen.** Um den Docker-Client von jedem Ort im Verwaltungssystem ausführen zu können, müssen Sie den Ordner *C:\ProgramData\Docker* zur Umgebungsvariablen *PATH* des Systems hinzufügen. In der grafischen Benutzeroberfläche öffnen Sie über die Systemsteuerung die App *System*, klicken dort auf *Erweiterte*

Systemeinstellungen und im Dialogfeld *Systemeigenschaften* auf die Schaltfläche *Umgebungsvariablen*, um das in Abbildung 4–3 gezeigte Dialogfeld anzuzeigen.

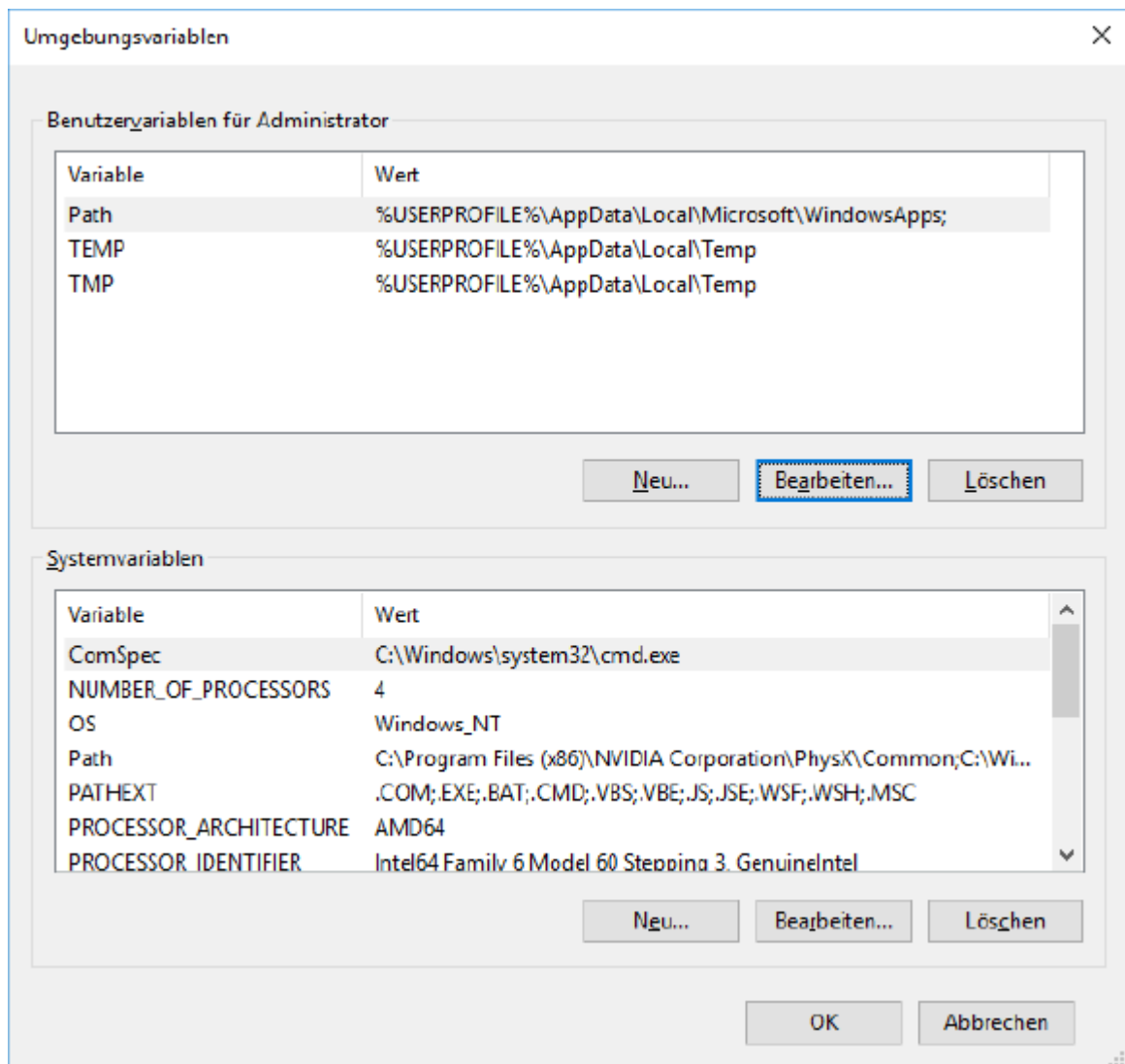


Abb. 4–3 Das Dialogfeld *Umgebungsvariablen*

8. In PowerShell erledigen Sie dies mit dem folgenden Befehl:

```
[environment]::setenvironmentvariable("path",  
  
$env:path + ";c:\program files\docker",  
  
[environmentvariabletarget]::machine)
```

Nachdem Sie diese Schritte abgeschlossen haben, können Sie den *Docker.exe*-Client außerhalb der Nanoserver-Sitzung starten, müssen aber in jedem Befehl den folgenden Parameter angeben, wobei Sie die Variable *<ipaddress>* durch die Adresse des Nanoservers ersetzen, den Sie verwalten möchten:

```
-h tcp://<ipaddress>:2375
```

So erstellen Sie beispielsweise mit dem folgenden Befehl einen neuen Container mit dem Image *microsoft/nanoserver*:

```
docker -h tcp://172.21.96.1:2375
```

```
run -it microsoft/nanoserver
```

```
cmd
```

Damit Sie den Parameter *-h* nicht an jeden Befehl anfügen müssen, erstellen Sie wie folgt eine neue Umgebungsvariable:

```
docker_host = "tcp://ipaddress:2375"
```

In PowerShell verwenden Sie hierfür einen Befehl wie den folgenden:

```
$env:docker_host = "tcp://172.21.96.1:2375"
```

Docker Daemon-Startoptionen konfigurieren

Wie bereits im vorherigen Abschnitt erwähnt, ist die Konfigurationsdatei für das *Dockerd*-Modul eine einfache Textdatei namens *daemon.json*, die Sie im selben Ordner wie die Datei *Dockerd.exe* speichern. Neben den Einstellungen, mit denen Sie weiter oben den Clientverkehr über das Netzwerk zugelassen haben, gibt es noch viele andere Konfigurationseinstellungen, die Sie in die Datei aufnehmen können. Sämtliche Einstellungen, die Sie in eine einzelne

daemon.json-Datei schreiben, sollten in einem einzigen Satz von geschweiften Klammern eingeschlossen sein, wie das folgende Beispiel zeigt:

```
{  
  
  "graph": "d:\docker"  
  
  "bridge" : "none"  
  
  "group" : "docker"  
  
  {"dns": 192.168.9.2, 192.168.9.6 }  
  
}
```



PRÜFUNGSTIPP

Seien Sie sich bewusst, dass die Windows-Portierung von Docker zwar viele Konfigurationseinstellungen der Linux-Version von *Dockerd* unterstützt, aber nicht den kompletten Funktionsumfang bietet. Wenn Sie sich die Docker-Dokumentation ansehen, achten Sie darauf, dass es sich um die Windows-Version des Dokuments handelt.

Images und Container umleiten

Um das *Dockerd*-Modul so zu konfigurieren, dass es Imagedateien und Container in einem alternativen Speicherort ablegt, schreiben Sie in die Datei *daemon.json* den folgenden Befehl, wobei Sie *d:\docker* durch den gewünschten Speicherort ersetzen:

```
{ "graph": "d:\docker" }
```

NAT unterdrücken

Standardmäßig erzeugt das *Dockerd*-Modul eine NAT-Umgebung (Network Address Translation) für Container, damit diese untereinander und mit dem externen Netzwerk kommunizieren können. Sie können dieses Standardverhalten ändern und verhindern, dass das Modul NAT verwendet, wenn Sie den folgenden Befehl in die Datei *daemon.json* einfügen:

```
{ "bridge" : "none" }
```

Eine administrative Gruppe erstellen

Standardmäßig können nur Mitglieder der lokalen Gruppe *Administratoren* mit dem Docker-Client das *Dockerd*-Modul steuern, wenn sie im lokalen System arbeiten. In manchen Fällen können Sie aber Benutzern diese Fähigkeit gewähren, ohne ihnen die Mitgliedschaft *Administratoren* zu geben. *Dockerd* lässt sich mit der folgenden Einstellung in der Datei *daemon.json* so konfigurieren, dass eine andere Gruppe erkannt wird – in diesem Fall die Gruppe namens »docker«:

```
{ "group" : "docker" }
```

DNS-Serveradressen festlegen

Um alternative DNS-Serveradressen für die Betriebssysteme in Containern festzulegen, können Sie die folgende Einstellung in die Datei *daemon.json* einfügen, wobei *address1* und *address2* die IP-Adressen von DNS-Servern sind:

```
{"dns": "address1", "address2" }
```

Windows PowerShell für die Verwendung mit Containern konfigurieren

Das *Dockerd*-Modul wird zwar mit einer *Docker.exe*-Clientshell bereitgestellt, ist aber nicht davon abhängig. Die gleichen Funktionen können Sie auch mit Windows PowerShell-Cmdlets ausführen. Das PowerShell-Modul *Docker* befindet sich wie Docker selbst in einer ständigen Phase gemeinsamer Entwicklung und ist demzufolge kein Bestandteil von Windows Server 2016.

Die aktuelle Version des PowerShell-Moduls können Sie aus einem Repository namens *DockerPS-Dev* mit den folgenden Befehlen herunterladen:

```
register-psrepository -name dockerps-dev

-sourcelocation https://ci.appveyor.com/nuget/docker-
powershell-dev

install-module docker

-repository dockerps-dev

-scope currentuser
```

Wenn der Download abgeschlossen ist, können Sie sich eine Liste der Docker-Cmdlets mit dem folgenden Befehl anzeigen lassen:

```
get-command -module docker
```

Abbildung 4-4 zeigt die aktuelle Ausgabe des Befehls.

```

PS C:\WINDOWS\system32> get-command -module docker
CommandType      Name                                     Version      Source
-----
Alias            Attach-Container                       0.1.0.111   docker
Alias            Build-ContainerImage                  0.1.0.111   docker
Alias            Commit-Container                      0.1.0.111   docker
Alias            Exec-Container                        0.1.0.111   docker
Alias            Load-ContainerImage                  0.1.0.111   docker
Alias            Pull-ContainerImage                  0.1.0.111   docker
Alias            Push-ContainerImage                  0.1.0.111   docker
Alias            Run-ContainerImage                   0.1.0.111   docker
Alias            Save-ContainerImage                  0.1.0.111   docker
Alias            Tag-ContainerImage                   0.1.0.111   docker
Cmdlet          Add-ContainerImageTag                 0.1.0.111   docker
Cmdlet          ConvertTo-ContainerImage              0.1.0.111   docker
Cmdlet          Copy-ContainerFile                    0.1.0.111   docker
Cmdlet          Enter-ContainerSession                0.1.0.111   docker
Cmdlet          Export-ContainerImage                 0.1.0.111   docker
Cmdlet          Get-Container                         0.1.0.111   docker
Cmdlet          Get-ContainerDetail                   0.1.0.111   docker
Cmdlet          Get-ContainerImage                    0.1.0.111   docker
Cmdlet          Get-ContainerNet                      0.1.0.111   docker
Cmdlet          Get-ContainerNetDetail                0.1.0.111   docker
Cmdlet          Import-ContainerImage                 0.1.0.111   docker
Cmdlet          Invoke-ContainerImage                 0.1.0.111   docker
Cmdlet          New-Container                         0.1.0.111   docker
Cmdlet          New-ContainerImage                    0.1.0.111   docker
Cmdlet          New-ContainerNet                      0.1.0.111   docker
Cmdlet          Remove-Container                      0.1.0.111   docker
Cmdlet          Remove-ContainerImage                 0.1.0.111   docker
Cmdlet          Remove-ContainerNet                   0.1.0.111   docker
Cmdlet          Request-ContainerImage                0.1.0.111   docker
Cmdlet          Start-Container                       0.1.0.111   docker
Cmdlet          Start-ContainerProcess                0.1.0.111   docker
Cmdlet          Stop-Container                        0.1.0.111   docker
Cmdlet          Submit-ContainerImage                 0.1.0.111   docker
Cmdlet          Wait-Container                        0.1.0.111   docker

PS C:\WINDOWS\system32> _

```

Abb. 4-4 Cmdlets im Modul *Docker* für Windows PowerShell

Wenn Sie einmal das Repository registriert und das Docker-Modul importiert haben, brauchen Sie diese Befehle nicht noch einmal auszuführen. Die neueste Version des Moduls rufen Sie mit dem folgenden Befehl ab:

```
update-module docker
```

Ein Basisbetriebssystem installieren

Sind das *Dockerd*-Modul und der Docker-Client installiert und betriebsbereit, können Sie den ersten Schritt zum Erstellen von Containern gehen: ein Basisbetriebssystemimage aus dem Docker Hub-Repository herunterladen. Microsoft stellt im Repository die Images für Windows Server 2016 Server Core und Nanoserver bereit. Diese können Sie herunterladen, Container damit erstellen und dann Ihre eigenen Container-Images erzeugen.

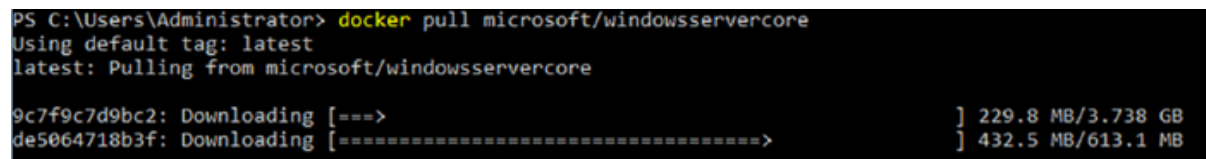
Um den Docker-Client zu verwenden, führen Sie die Datei *Docker.exe* mit einem Befehl und gegebenenfalls zusätzlichen Optionen und Parametern aus. Wollen Sie ein Image herunterladen, rufen Sie Docker mit dem Befehl *Pull* und dem Namen des Images auf. Zum Beispiel lädt der folgende Befehl das Server Core-Image aus dem Repository herunter:

```
docker pull microsoft/windowsservercore
```

Der äquivalente PowerShell-Befehl sieht folgendermaßen aus:

```
request-containerimage  
  
-repository microsoft/windowsservercore
```

Abbildung 4-5 zeigt die Ausgabe des Befehls (die je nach Geschwindigkeit Ihrer Internetverbindung etwas dauern kann).



```
PS C:\Users\Administrator> docker pull microsoft/windowsservercore  
Using default tag: latest  
latest: Pulling from microsoft/windowsservercore  
9c7f9c7d9bc2: Downloading [==>] 229.8 MB/3.738 GB  
de5064718b3f: Downloading [=====>] 432.5 MB/613.1 MB
```

Abb. 4-5 Ausgabe des *Docker Pull*-Befehls

Der *Docker Pull*-Befehl lädt standardmäßig die neueste Version des angegebenen Images herunter, was an der Markierung »latest« zu erkennen ist. Stehen mehrere Versionen des gleichen Images zur Verfügung, wie es in einem Projekt der Anwendungsentwicklung der Fall ist, können Sie das jeweilige Tag spezifizieren und somit eines der vorherigen Images zum Download auswählen. Wenn Sie den *Docker Pull*-Befehl mit dem Parameter *-a* ausführen, erhalten Sie alle Versionen des Images. Besteht das Image, das Sie abrufen, aus mehreren Schichten, lädt der Befehl automatisch alle benötigten Schichten herunter, um das Image in einem Container bereitzustellen.

```

PS C:\Users\Administrator> docker search microsoft
NAME                DESCRIPTION                STARS    OFFICIAL    AUTOM
ATED
microsoft/aspnet    ASP.NET is an open source server-side Web ... 498      [OK]
microsoft/dotnet    Official images for .NET Core for Linux an... 327      [OK]
mono                Mono is an open source implementation of M... 195      [OK]
microsoft/windowsservercore
Windows Server 2016 Server Core base OS im... 69
microsoft/nanoserver
Windows Server 2016 Nano Server base OS im... 66
microsoft/azure-cli
Docker image for Microsoft Azure Command L... 66      [OK]
microsoft/iis       Internet Information Services (IIS) instal... 50
microsoft/mssql-server-2014-express-windows
Microsoft SQL Server 2014 Express installe... 41
microsoft/aspnetcore
Official images for running compiled ASP.N... 28      [OK]
microsoft/mssql-server-2016-express-windows
Microsoft SQL Server 2016 Express installe... 28
microsoft/powershell
Official PowerShell Core releases from htt... 8      [OK]
microsoft/oms       Monitor your containers using the Operatio... 7      [OK]
microsoft/aspnetcore-build
Official images for building ASP.NET Core ... 6      [OK]
microsoft/dotnet35  The .NET Framework 3.5 image has moved to ... 4
microsoft/vsts-agent
Official images for the Visual Studio Team... 4
microsoft/applicationinsights
Application Insights for Docker helps you ... 4      [OK]
microsoft/sample-nginx
Nginx installed in Windows Server Core and... 4
microsoft/dotnet-nightly
Preview bits of the .NET Core CLI          2      [OK]
microsoft/powershell-nightly
Nightly builds of PowerShell Core for CI    2      [OK]
microsoft/sample-dotnet
.NET Core running in a Nano Server container 1
microsoft/cntk      CNTK                                        0      [OK]
dreher/microsoft   Microsoft Test Repo                        0
microsoft/aspnetcore-build-nightly
Images to build preview versions of ASP.NE... 0      [OK]
microsoft/dotnet-samples
.NET Core Docker Samples                    0      [OK]
berlius/microsoft-malmo
Microsoft-malmo - artificial intelligence ... 0
PS C:\Users\Administrator> docker pull microsoft/nanoserver
Using default tag: latest
latest: Pulling from microsoft/nanoserver
5496abde368a: Pull complete
94b4ce7ac4c7: Pull complete
Digest: sha256:86cfed90ee6f711086d9cd637b7d8f250270c46cfe4e08f7527aea7968b9c8ff
Status: Downloaded newer image for microsoft/nanoserver:latest
PS C:\Users\Administrator>

```

Abb. 4-6 Ausgabe des Befehls *Docker Search*

Wenn Sie wissen, dass im Repository ein Nanoserver-Image bereitsteht, Sie aber dessen genauen Namen nicht kennen, können Sie mit dem Befehl *Docker Search* das Image suchen und es dann mit *Docker Pull* herunterladen, wie Abbildung 4-6 zeigt.

Ein Image markieren

Das Markieren (Tagging) in einem Container-Repository ist ein Mechanismus der Versionskontrolle. Wenn Sie mehrere Versionen des gleichen Images erstellen, beispielsweise aufeinanderfolgende Builds einer Anwendung, ermöglicht Docker es, diesen Images Tags zuzuweisen, die die Versionen identifizieren. Tags sind normalerweise Nummern, die das relative Alter der Image-Iterationen anzeigen, etwa 1.1, 1.2, 2.0 usw.

Es gibt zwei Methoden, um einem Image ein Tag zuzuweisen: *Docker* mit dem Befehl *Tag* ausführen und *Docker Build* mit dem Parameter *-t* aufrufen. In beiden Fällen ist das Format des Image-Kennzeichners das gleiche.

Um ein Image auf Ihrem lokalen Containerhost zu markieren, verwenden Sie die folgende Syntax:

```
docker tag <imagename>:<tag>
```

Wenn Sie das Image auf den Docker Hub uploaden, müssen Sie vor den Imagennamen Ihren Docker Hub-Benutzernamen und einen Schrägstrich setzen:

```
docker tag <username>/<imagename>:<tag>
```

Zum Beispiel könnte ein Benutzer Holly Holt den letzten Build ihrer neuen Anwendung wie folgt markieren:

```
docker tag hholt/killerapp:1.5
```

In Windows PowerShell führen Sie den entsprechenden Befehl mit dem Cmdlet *Add-ContainerImageTag* wie folgt aus:

```
add-containerimagetag -imageidornome c452b8c6ee1a
```

```
-repository hholt/killerapp
```

```
-tag 1.5
```

Fehlt der Tag-Wert im Befehl, weist Docker dem Image automatisch den Tag-Wert »latest« zu, was zu Verwirrungen führen kann. Wenn Sie ein Image aus einem Repository abrufen, ohne ein Tag anzugeben, liefert das Repository das Image mit dem Tag »latest« zurück. Allerdings bedeutet das nicht unbedingt, dass Sie wirklich das neueste Image bekommen.

Das Tag »latest« soll anzeigen, dass es sich beim Image mit dieser Markierung um die neueste Version handelt. Ob das jedoch tatsächlich stimmt oder nicht, hängt von den Leuten ab, die die Tags für dieses Repository verwalten. Man könnte annehmen, dass das Tag »latest« automatisch der jeweils allerneuesten Version eines Images neu zugewiesen wird. Das ist aber nicht der Fall. Das Tag »latest« können Sie jeder Version eines Images zuweisen, auch der ältesten oder der neuesten. Es liegt allein bei den Managern des Repositories, die Tag-Werte geeignet zu verwalten. Wenn Sie jemand um den letzten Build eines Images bittet, meint diese Person dann den neuesten Build oder den Build mit dem Tag »latest«? Das ist nicht immer dasselbe.

Ein Betriebssystem-Image deinstallieren

Wenn Sie *Docker* mit dem Befehl *Images* ausführen, werden alle Images auf dem Containerhost angezeigt, wie in Abbildung 4–7 zu sehen ist.

```
PS C:\WINDOWS\system32> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
microsoft/sample-dotnet  latest             c14528829a37      9 days ago        911 MB
microsoft/iis        latest             b6a44de60ef9      3 weeks ago       8.96 GB
microsoft/windowsservercore latest             93a9c37b36d0      6 weeks ago       8.68 GB
microsoft/nanoserver  10.0.14393.206    853f9db844af      6 weeks ago       652 MB
microsoft/nanoserver  latest             e14bc0ceca12      6 weeks ago       810 MB
microsoft/nanoserver  10.0.14393.206_de-de a896e5590871      6 weeks ago       658 MB
microsoft/nanoserver  10.0.14393.206_cs-cz ef42b616e27e      6 weeks ago       653 MB
microsoft/nanoserver  10.0.14300.1030   3a703c6e97a2      4 months ago      970 MB
PS C:\WINDOWS\system32> _
```

Abb. 4–7 Ausgabe des Befehls *Docker Images*

In manchen Fällen wird Ihnen beim Durchsehen der Liste von Images auffallen, dass Sie manche Images nicht mehr brauchen. In diesem Beispiel sind dies zwei nicht-englische Versionen von Nanoserver, die versehentlich heruntergeladen wurden.

Möchten Sie nicht mehr benötigte Images entfernen und von ihnen belegten Speicherplatz freigeben, führen Sie Docker mit dem Befehl *Rmi* aus und geben entweder das Repository und das Tag des zu löschenden Images oder den Image-ID-Wert wie in den folgenden Beispielen an:

```
docker rmi -f microsoft/nanoserver:10.0.14393.206_de-de
```

```
docker rmi -f a896e5590871
```

Das PowerShell-Äquivalent ist das Cmdlet *Remove-ContainerImage*, wie in den folgenden Beispielen:

```
remove-containerimage
microsoft/nanoserver:10.0.14393.206_de-de
```

```
remove-containerimage a896e5590871
```

Es ist durchaus möglich, dass dasselbe Image mit mehreren Tags aufgelistet ist. Das erkennen Sie an den übereinstimmenden Image-ID-Werten. Wenn Sie versuchen, eines dieser Images mithilfe des Tags zu entfernen, erscheint ein Fehler, weil das Image mit anderen Tags verwendet wird. Mit dem zusätzlichen

Parameter `-f` zwingen Sie den Befehl, alle markierten Verweise auf dasselbe Image zu löschen.

Windows Server-Container erstellen

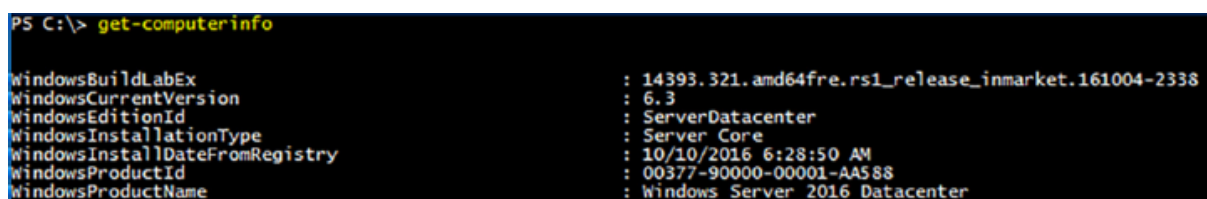
Mit dem eingerichteten Container-Feature und dem installierten Docker können Sie nun einen Windows Server-Container erstellen. Dazu führen Sie den Befehl `Docker Run` aus und geben das Image an, das Sie im Container ausführen möchten. Zum Beispiel erzeugt der folgende Befehl einen neuen Container mit dem Server Core-Image, das von Docker Hub heruntergeladen wird:

```
docker run -it microsoft/windowsservercore powershell
```

Außer dass das Image in den Container geladen wird, bewirken die Parameter in diesem Befehl Folgendes:

- **i** Erstellt eine interaktive Sitzung mit dem Container.
- **t** Öffnet ein Terminalfenster in den Container.
- **powershell** Führt den PowerShell-Befehl in der Containersitzung aus.

Im Ergebnis erscheint nach dem Laden des Containers eine PowerShell-Sitzung, sodass Sie innerhalb des Containers arbeiten können. Wenn Sie in dieser Sitzung das Cmdlet `Get-ComputerInfo` ausführen, sehen Sie im oberen Teil der Ausgabe (siehe Abbildung 4–8), dass Server Core im Container ausgeführt wird, während die Version Desktopdarstellung im Containerhost läuft.



```
PS C:\> get-computerinfo
WindowsBuildLabEx           : 14393.321.amd64fre.rs1_release_inmarket.161004-2338
WindowsCurrentVersion       : 6.3
WindowsEditionId            : ServerDatacenter
WindowsInstallationType     : Server Core
WindowsInstallDateFromRegistry : 10/10/2016 6:28:50 AM
WindowsProductId            : 00377-90000-00001-AA588
WindowsProductName          : Windows Server 2016 Datacenter
```

Abb. 4–8 Ausgabe des Cmdlets `Get-ComputerInfo`

Die Schalter des Befehls `Docker Run` können Sie zusammenfassen, sodass die Schalter `-i` und `-t` als `-it` erscheinen. Nach dem Namen des Images kann jeder Befehl stehen, der im Container auszuführen ist. Wenn Sie zum Beispiel `cmd` angeben, öffnen Sie damit die normale Befehlshell von Windows anstelle von PowerShell.

HINWEIS Images abrufen

Ein Image müssen Sie nicht unbedingt als Erstes vom Docker Hub abrufen, bevor Sie es ausführen können. Wenn Sie einen *Docker Run*-Befehl ausführen und das erforderliche Image in Ihrem Containerhost nicht vorhanden ist, leitet Docker automatisch ein Abrufen ein und erstellt dann den Container. Bei großen Images sparen Sie aber Zeit, wenn Sie neue Container erstellen und die jeweiligen Images vorab heruntergeladen haben.

Der Befehl *Docker Run* unterstützt viele Befehlszeilenparameter und Schalter, mit denen Sie die Umgebung des zu erzeugenden Containers optimieren können. Mit dem folgenden Befehl lassen sich diese Parameter und Schalter anzeigen:

```
docker run--help
```

HINWEIS Docker-Befehle ausführen

Bei diesem und vielen anderen Docker-Befehlen sind die Befehlszeilenparameter mit doppelten Bindestrichen zu schreiben.

Abbildung 4-9 zeigt ungefähr die Hälfte der verfügbaren Parameter. Zum Beispiel können Sie mit dem Parameter *-h* einen Hostnamen für den Container angeben anstelle der hexadezimalen Zeichenfolge, die der Befehl standardmäßig zuweist.

```

PS C:\WINDOWS\system32> docker run --help
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
Run a command in a new container
Options:
--add-host value          Add a custom host-to-IP mapping (host:ip) (default [])
-a, --attach value        Attach to STDIN, STDOUT or STDERR (default [])
--blkio-weight value      Block IO (relative weight), between 10 and 1000
--blkio-weight-device value Block IO weight (relative device weight) (default [])
--cap-add value           Add Linux capabilities (default [])
--cap-drop value         Drop Linux capabilities (default [])
--cgroup-parent string    Optional parent cgroup for the container
--cidfile string          Write the container ID to the file
--cpu-percent int         CPU percent (Windows only)
--cpu-period int          Limit CPU CFS (Completely Fair Scheduler) period
--cpu-quota int           Limit CPU CFS (Completely Fair Scheduler) quota
-c, --cpu-shares int      CPU shares (relative weight)
--cpuset-cpus string      CPUs in which to allow execution (0-3, 0,1)
--cpuset-mems string      MEMs in which to allow execution (0-3, 0,1)
--credential-spec string  Credential spec for managed service account (Windows only)
-d, --detach              Run container in background and print container ID
--detach-keys string      Override the key sequence for detaching a container
--device value            Add a host device to the container (default [])
--device-read-bps value   Limit read rate (bytes per second) from a device (default [])
--device-read-iops value  Limit read rate (IO per second) from a device (default [])
--device-write-bps value  Limit write rate (bytes per second) to a device (default [])
--device-write-iops value Limit write rate (IO per second) to a device (default [])
--disable-content-trust   Skip image verification (default true)
--dns value              Set custom DNS servers (default [])
--dns-opt value          Set DNS options (default [])
--dns-search value       Set custom DNS search domains (default [])
--entrypoint string       Overwrite the default ENTRYPOINT of the image
-e, --env value           Set environment variables (default [])
--env-file value          Read in a file of environment variables (default [])
--expose value            Expose a port or a range of ports (default [])
--group-add value         Add additional groups to join (default [])
--health-cmd string       Command to run to check health
--health-interval duration Time between running the check (default 0s)
--health-retries int      Consecutive failures needed to report unhealthy
--health-timeout duration Maximum time to allow one check to run (default 0s)
--help                   Print usage
-h, --hostname string     Container host name
--init                   Run an init inside the container that forwards signals and reaps processes
-i, --interactive         Keep STDIN open even if not attached
--io-maxbandwidth string  Maximum IO bandwidth limit for the system drive (Windows only)
--io-maxiops int         Maximum IOps limit for the system drive (Windows only)
--ip string               Container IPv4 address (e.g. 172.30.100.104)
--ip6 string              Container IPv6 address (e.g. 2001:db8::33)

```

Abb. 4-9 Ausgabe des Befehls *Docker Run --help*

Das PowerShell-Äquivalent des Befehls *Docker Run* verwendet das Cmdlet *New-Container*, wie das folgende Beispiel zeigt:

```

new-container -imageidorname microsoft/windowsservercore

    -input

    -terminal

    -command powershell

```

Hyper-V-Container erstellen

Einen Hyper-V-Container erstellen Sie in fast den gleichen Schritten wie einen Windows Server-Container. Hierfür verwenden Sie denselben *Docker Run*-Befehl,

außer dass Sie den Parameter `—isolation=hyperv` anfügen, wie es im folgenden Beispiel zu sehen ist:

```
docker run -it  
  
-isolation=hyperv microsoft/windowsservercore  
  
powershell
```

Nachdem Sie einen Hyper-V-Container erstellt haben, lässt er sich kaum von einem Windows Server-Container unterscheiden. Es gibt nur wenige Methoden, die Arten der Container auseinanderzuhalten. So kann man untersuchen, wie sie mit Prozessen umgehen. Zum Beispiel können Sie zwei Container erstellen und in jedem einen Befehl ausführen, der an den jeweils anderen Computer fortlaufend Ping-Befehle sendet:

```
docker run -it microsoft/windowsservercore  
  
ping -t localhost  
  
docker run -it  
  
-isolation=hyperv microsoft/windowsservercore  
  
ping -t localhost
```

Der vom ersten Befehl erstellte Windows Server-Container lässt im Container einen PING-Prozess laufen, wie die Ausgabe des Befehls *Docker Top* in Abbildung 4-10 zeigt. Die Prozess-ID (PID) lautet hier 404. Wenn Sie dann das Cmdlet *Get-Process* ausführen, um die Prozesse (die mit P beginnen) auf dem Containerhost anzuzeigen, sehen Sie denselben PING-Prozess mit der ID 404. Das hängt damit zusammen, dass der Container den Kernel mit dem Containerhost gemeinsam nutzt.

```

PS C:\WINDOWS\system32> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
0e38bdac48ca   microsoft/windowsservercore        "powershell"           5 hours ago   Up 5 hours
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> docker top 0e38bdac48ca
Name          PID      CPU          Private Working Set
smss.exe      2420    00:00:00.125  229.4 kB
csrss.exe     6444    00:00:00.390  946.2 kB
wininit.exe   3220    00:00:00.187  806.9 kB
services.exe  7636    00:00:00.453  1.794 MB
lsass.exe     8584    00:00:01.156  3.633 MB
svchost.exe   5860    00:00:00.406  2.208 MB
svchost.exe   8360    00:00:00.265  1.745 MB
svchost.exe   7296    00:00:00.281  2.06 MB
svchost.exe   6916    00:00:00.578  3.912 MB
svchost.exe   7888    00:00:09.015  10.95 MB
svchost.exe   2460    00:00:00.640  3.219 MB
svchost.exe   4340    00:00:04.140  8.409 MB
svchost.exe   4880    00:00:00.062  839.7 kB
CEExecSvc.exe 3528    00:00:00.796  815.1 kB
svchost.exe   4288    00:00:02.218  4.235 MB
powershell.exe 2016    00:00:10.781  33.62 MB
msdtc.exe     7816    00:00:00.078  1.876 MB
powershell.exe 6768    00:00:07.515  30.3 MB
PING.EXE      404     00:00:00.031  589.8 kB
PS C:\WINDOWS\system32> get-process ping
Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id  SI ProcessName
-----  -
71        5       788    3752    0.03    404  7 PING
549       27     56560  64640    1.02   1752  2 powershell
542       43     60420  76424   10.78   2016  7 powershell
517       40     56816  70660    7.52   6768  7 powershell
664       40     77556  98004    5.48   8224  2 powershell
537       27     54696  57236    3.75   8864  2 powershell
PS C:\WINDOWS\system32> _

```

Abb. 4-10 Ausgabe der Befehle *Docker Top* und *Get-Process* für einen Windows Server-Container

Wenn Sie dagegen den Befehl *Docker Top* auf dem Hyper-V-Container ausführen, sehen Sie wieder den PING-Prozess, dieses Mal mit der PID 1852 (siehe Abbildung 4-11). In der Ausgabe des Cmdlets *Get-Process* erscheint aber kein PING-Prozess, weil dieser Container seinen eigenen Kernel besitzt, der vom Hypervisor bereitgestellt wird.

```

PS C:\Users\Administrator> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
8d67f1679c68   microsoft/windowsservercore        "ping -t localhost"    6 minutes ago Up 5 minutes
y_lovelace
PS C:\Users\Administrator>
PS C:\Users\Administrator> docker top 8d67f1679c68
Name          PID      CPU          Private Working Set
smss.exe     248      00:00:01.156 233.5 kB
csrss.exe    312      00:00:00.750 921.6 kB
wininit.exe  644      00:00:00.453 737.3 kB
services.exe 920      00:00:01.296 1.532 MB
lsass.exe    828      00:00:00.515 2.22 MB
svchost.exe  1076     00:00:00.359 1.958 MB
svchost.exe  1124     00:00:00.234 1.401 MB
svchost.exe  1212     00:00:00.421 2.023 MB
svchost.exe  1228     00:00:00.953 4.919 MB
svchost.exe  1268     00:00:00.359 2.605 MB
svchost.exe  1280     00:00:09.000 13.43 MB
svchost.exe  1368     00:00:08.500 3.174 MB
svchost.exe  1528     00:00:00.703 3.146 MB
svchost.exe  1540     00:00:00.093 819.2 kB
CExecSvc.exe 1592     00:00:00.046 688.1 kB
PING.EXE     1852     00:00:00.031 589.8 kB
msdtc.exe    872      00:00:00.203 1.901 MB
WmiPrvSE.exe 2004     00:00:02.078 5.526 MB
PS C:\Users\Administrator>
PS C:\Users\Administrator> get-process p*
Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id  SI ProcessName
-----  -
561      27     53540  61920  1.42    1096  2 powershell
696      27     53540  61628  3.20    4564  2 powershell
PS C:\Users\Administrator> _

```

Abb. 4-11 Ausgabe der Befehle *Docker Top* und *Get-Process* für einen Hyper-V-Container

Prüfungsziel 4.2: Windows-Container verwalten

- Windows- oder Linux-Container mit dem Docker-Daemon verwalten
- Windows- oder Linux-Container mithilfe von Windows PowerShell verwalten
- Containernetzwerke verwalten
- Container-Datenvolumen verwalten
- Ressourcensteuerung verwalten
- Neue Container-Images mit Dockerfile erstellen
- Container-Images mit DockerHub-Repository für öffentliche und private Szenarien verwalten
- Container-Images mit Microsoft Azure verwalten

Windows- oder Linux-Container mit dem Docker-Daemon verwalten

Wenn Sie mit dem Befehl *Docker Run* einen neuen Container erstellen, können Sie die Schalter *-it* angeben, um mit ihm interaktiv zu arbeiten, oder sie

weglassen und den Container im Hintergrund laufen lassen. In jedem Fall können Sie mit dem Docker-Client den Container verwalten – ob Windows oder Linux.

Container auflisten

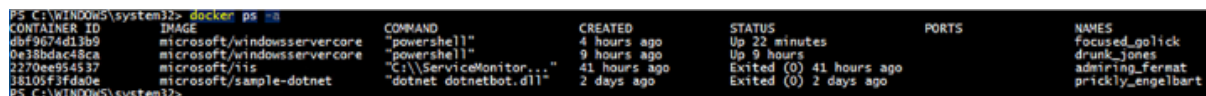
Möchten Sie eine PowerShell- oder CMD-Sitzung verlassen, die Sie in einem Container begonnen haben, können Sie einfach den folgenden Befehl eingeben:

```
exit
```

Dieser Befehl schließt allerdings nicht nur die Sitzung, sondern hält auch den Container an. Ein angehaltener Container existiert weiterhin auf dem Host; er ist lediglich funktionell abgeschaltet. Um eine Sitzung zu verlassen, ohne den Container anzuhalten, drücken Sie **Strg** + **Q** und dann **Strg** + **P**.

Eine Liste aller auf dem Host ausgeführten Container zeigen Sie mit dem Befehl *Docker ps* an. Wenn Sie den Schalter *-a* (für alle) wie im folgenden Beispiel anfügen, zeigt der Befehl alle Container auf dem Host an, ob sie ausgeführt werden oder nicht (siehe Abbildung 4–12):

```
docker ps -a
```



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
dbf9674d13b9	microsoft/windowsservercore	"powershell"	4 hours ago	Up 22 minutes		focused_golick
0e38bdac48ca	microsoft/windowsservercore	"powershell"	9 hours ago	Up 9 hours		drunk_jones
2270ee954337	microsoft/iris	"C:\\ServiceMonitor..."	41 hours ago	Exited (0) 41 hours ago		admiring_ferret
38105f3fd0e	microsoft/sample-dotnet	"dotnet dotnetbot.dll"	2 days ago	Exited (0) 2 days ago		prickly_engelbart

Abb. 4–12 Ausgabe eines Befehls *Docker ps*

Container starten und anhalten

Einen angehaltenen Container starten Sie mit dem Befehl *Docker Start*, zum Beispiel:

```
docker start dbf9674d13b9
```

Mit dem Befehl *Docker Stop* können Sie einen Container auch zwangsweise anhalten, zum Beispiel:

```
docker stop dbf9674d13b9
```

Die sechs Bytes umfassende hexadezimale Zeichenfolge in diesen Befehlen ist die Container-ID, die Docker dem Container beim Erstellen zuweist. Mit diesem Wert identifizieren Sie in Docker-Befehlen den Container, den Sie verwalten möchten. Außerdem wird dieser Wert zum Computernamen des Containers, wovon Sie sich mit dem Cmdlet *Get-ComputerInfo* überzeugen können, das Sie in einer Containersitzung ausführen. Wenn Sie *Docker PS* mit dem Parameter *—no-trunc* (für no truncation, ungekürzt) ausführen, zeigt sich, dass die Container-ID tatsächlich eine Zeichenfolge aus 32 Bytes in hexadezimaler Schreibweise ist, auch wenn es wesentlich bequemer ist, nur die ersten sechs Bytes auf der Befehlszeile zu verwenden (siehe Abbildung 4–13).

```
PS C:\WINDOWS\system32> docker ps -a --no-trunc
CONTAINER ID        PORTS              NAMES                IMAGE                                COMMAND                    CREATED            STATUS
dbf9674d13b91f3e9511edb6c037017f0759b431a4f4ae7ff89e248ae8e59107  focused_golick      microsoft/windowsservercore        "powershell"             5 hours ago       Exited (0) 19 se
0e33bdac48ca0120eff6491a7b9d1908e5318021130c1707b924991ae8d1504f  drunk_jones        microsoft/windowsservercore        "powershell"             9 hours ago       Up 9 hours
2270ee954337765fca809c176d126221e832a9e87b0c865e17b3088adf3e9c3f  admiring_fermat    microsoft/iis                       "C:\ServiceMonitor.exe w3svc cmd" 42 hours ago       Exited (0) 42 ho
38105f37fdade780150ff3b0509c64a6616a12bcad416994304299e3488147c0b  prickly_engelbart  microsoft/sample-dotnet            "dotnet dotnetbot.dll"         2 days ago       Exited (0) 2 day
PS C:\WINDOWS\system32>
```

Abb. 4–13 Ausgabe des Befehls *Docker ps -a --no-trunc*

An Container anfügen

Mit dem Befehl *Docker Attach* können Sie sich mit einer Sitzung auf einem ausgeführten Container verbinden, wie das folgende Beispiel zeigt:

```
docker attach dbf9674d13b9
```

Wenn Sie den Befehl in mehreren Fenstern ausführen, werden zusätzliche Sitzungen geöffnet, sodass Sie in mehreren Fenstern gleichzeitig arbeiten können.

Images erstellen

Haben Sie einen Container in irgendeiner Weise geändert, können Sie die Änderungen in einem neuen Image speichern. Verwenden Sie hierfür den Befehl *Docker Commit* wie im folgenden Beispiel:

```
docker commit dbf9674d13b9 hholt/killerapp:1.5
```

Dieser Befehl erstellt ein neues Image namens *hholt/killerapp* mit einem Tag-Wert von 1.5. Der Befehl *Docker Commit* erzeugt kein Duplikat des Basisimages mit den von Ihnen vorgenommenen Änderungen, sondern speichert nur die Änderungen. Wenn Sie zum Beispiel mit dem Basisimage *Microsoft/windowsservercore* den Container erstellen und dann Ihre Anwendung installieren, speichert der Befehl *Docker Commit* nur die Anwendung. Und wenn Sie das neue Image einem Kollegen geben, muss er über das Basisimage verfügen (oder es abrufen), um den Container ausführen zu können.

Container entfernen

Möchten Sie einen Container vollständig entfernen, führen Sie den Befehl *Docker rm* wie im folgenden Beispiel aus:

```
docker rm dbf9674d13b9
```

Bevor Sie Container auf diese Weise entfernen können, müssen sie sich im angehaltenen Zustand befinden. Allerdings können Sie auch den Schalter *-f* (für force, erzwingen) anfügen, sodass der Befehl *Docker rm* jeden beliebigen Container entfernt, selbst einen, der gerade ausgeführt wird.

Windows- oder Linux-Container mithilfe von Windows PowerShell verwalten

Wie bereits weiter vorn erwähnt, ist es für das *Dockerd*-Modul nicht erforderlich, das Clientprogramm *Docker.exe* zu verwenden. Da es sich bei Docker um ein Open-Source-Projekt handelt, lässt sich auch eine alternative Clientimplementierung mit *Dockerd* verwenden, und in Zusammenarbeit mit der Docker-Gemeinde realisiert Microsoft genau dies mit einem PowerShell-Modul, mit dem Sie Docker-Container erstellen und verwalten können.

Weil sich das Docker-Modul für PowerShell noch im Entwicklungsstadium befindet, unterstützt es nicht unbedingt sämtliche Funktionen, die mit dem *Docker.exe*-Client möglich sind. Die wichtigsten Funktionen sind aber bereits realisiert, wie die folgenden Abschnitte zeigen.

Container auflisten

Eine Liste aller Container auf dem Host können Sie per Windows PowerShell mit dem Cmdlet *Get-Container* anzeigen, wie Abbildung 4–14 zeigt. Im Unterschied zum Befehl *Docker ps* zeigt das Cmdlet *Get-Container* sämtliche Container auf dem Host an, ob sie ausgeführt werden oder nicht.

```
PS C:\WINDOWS\system32> get-container
ID                Image                Command                Created                Status                Names
----                -
080096dce22901167... microsoft/wi... powershell            11/5/2016 9:14:09 AM  Up 5 minutes         infallible_mccarthy
d8d297343e8a1c27c... microsoft/wi... powershell            11/5/2016 6:26:51 AM  Exited (1067) 54 ... small_brown
dbf9674d13b91f5e9... microsoft/wi... powershell            11/4/2016 10:39:56 PM  Up 4 hours           focused_golick
0e38bdac48ca0120e... microsoft/wi... powershell            11/4/2016 6:09:47 PM  Up 16 hours          drunk_jones
2270ee954537765fc... microsoft/iis  C:\ServiceMonitor... 11/3/2016 9:43:16 AM  Exited (0) 2 days... admiring_fermat
38105f3fda0e78015... microsoft/sa... dotnet dotnetbot.dll  11/2/2016 5:41:28 AM  Exited (0) 3 days... prickly_engelbart

PS C:\WINDOWS\system32> _
```

Abb. 4–14 Ausgabe des Cmdlets *Get-Container*

Container starten und anhalten

Wenn Sie einen Container mit dem Cmdlet *New-Container* erstellen, startet es den Container standardmäßig nicht. Vielmehr müssen Sie ihn explizit starten. Um einen angehaltenen Container zu starten, führen Sie das Cmdlet *Start-Container* wie im folgenden Beispiel aus:

```
start-container dbf9674d13b9
```

Einen Container können Sie auch anhalten, indem Sie einfach das Verb im Cmdlet zu *Stop-Container* wechseln:

```
stop-container dbf9674d13b9
```

Container anfügen

Um sich mit einer Sitzung auf einem ausgeführten Container zu verbinden, rufen Sie das Cmdlet *Enter-ContainerSession* wie im folgenden Beispiel auf:

```
Enter-containersession dbf9674d13b9
```

Dieses Cmdlet trägt auch den Aliasnamen *Attach-Container*, sodass Sie einen anderen Befehl wiederverwenden können, indem Sie lediglich ein Verb austauschen.

Images erstellen

Haben Sie einen Container in irgendeiner Weise geändert, können Sie die Änderungen in einem neuen Image speichern, indem Sie das Cmdlet *ConvertTo-ContainerImage* wie im folgenden Beispiel ausführen:

```
convertto-containerimage -containeridornome dbf9674d13b9  
  
-repository holt/killerapp  
  
-tag 1.5
```

Dieses Cmdlet hat auch den Alias *Commit-Container*.

Container entfernen

Einen Container entfernen Sie mit dem Cmdlet *Remove-Container*, zum Beispiel:

```
remove-container dbf9674d13b9
```

Wie beim Befehl *Docker RM* müssen sich Container im angehaltenen Zustand befinden, bevor man sie entfernen kann. Mit dem Schalter *Force* entfernt der Cmdlet-Befehl jedoch jeden Container, selbst wenn er ausgeführt wird.

Containernetzwerke verwalten

Container können auf das externe Netzwerk zugreifen. Das lässt sich leicht nachweisen, indem man einen Server im lokalen Netzwerk oder im Internet anpingt. Wenn Sie allerdings den Befehl *Ipconfig /all* in einer Containersitzung ausführen (siehe Abbildung 4–15), sind Sie vielleicht vom Ergebnis überrascht.

```
PS C:\> ipconfig /all
ipconfig /all

Windows IP Configuration

Host Name . . . . . : f3e054399471
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : zacker

Ethernet adapter vEthernet (Container NIC 76b9f047):

Connection-specific DNS Suffix . . : zacker
Description . . . . . : Hyper-V Virtual Ethernet Adapter #7
Physical Address. . . . . : 00-15-5D-11-BF-40
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::ad08:3832:6ffe:ff4a%44(Preferred)
IPv4 Address. . . . . : 172.25.117.12(Preferred)
Subnet Mask . . . . . : 255.255.240.0
Default Gateway . . . . . : 172.25.112.1
DNS Servers . . . . . : 172.25.112.1
                          192.168.2.2
                          204.186.110.114
NetBIOS over Tcpi. . . . . : Disabled
PS C:\>
```

Abb. 4-15 Ausgabe des Befehls *Ipconfig /all* auf einem Container

In diesem Beispiel lautet die IP-Adresse des Netzwerkadapters im Container 172.25.117.12/12, was nichts weiter ist als die Adresse des Netzwerks, in dem sich der Containerhost befindet. Wenn Sie aber den Befehl *Ipconfig /all* auf dem Containerhost ausführen (siehe Abbildung 4-16), wird die Situation verständlicher.

Es gibt zwei Ethernet-Adapter, die im Containerhostsystem zu sehen sind. Der eine hat eine IP-Adresse im Netzwerk 192.168.2.0/24, die für das physische Netzwerk verwendet wird, mit dem der Containerhost verbunden ist. Der andere Adapter hat die Adresse 172.25.112.1/12, die im selben Netzwerk wie die Adresse des Containers liegt. Sehen Sie sich noch einmal die Konfiguration des Containers an: Die Adresse des Containerhosts ist als Standardgateway und DNS-Serveradresse für den Container aufgeführt. Der Containerhost fungiert letztlich als Router zwischen dem Netzwerk 172.16.0.0/12, in dem sich der Container befindet, und dem physischen Netzwerk 192.168.2.0/24, mit dem der Host verbunden ist. Darüber hinaus fungiert der Host als DNS-Server für den Container.

```
PS C:\WINDOWS\system32> ipconfig /all

Windows IP Configuration

Host Name . . . . . : CZ10
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : zacker

Ethernet adapter vEthernet (HNS Internal NIC):

Connection-specific DNS Suffix . :
Description . . . . . : Hyper-V Virtual Ethernet Adapter #4
Physical Address. . . . . : 00-15-5D-11-BB-AC
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::49c7:9ebd:f079:2994%29(Preferred)
IPv4 Address. . . . . : 172.25.112.1(Preferred)
Subnet Mask . . . . . : 255.255.240.0
Default Gateway . . . . . :
DHCPv6 IAID . . . . . : 486544733
DHCPv6 Client DUID. . . . . : 00-01-00-01-1F-96-45-81-44-37-E6-C0-9D-DF
DNS Servers . . . . . : fec0:0:0:ffff::1%1
                       fec0:0:0:ffff::2%1
                       fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter vEthernet (Intel(R) 82579LM Gigabit Network Connection):

Connection-specific DNS Suffix . : zacker
Description . . . . . : Hyper-V Virtual Ethernet Adapter #2
Physical Address. . . . . : 44-37-E6-C0-9D-DF
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::e170:47de:5b5a:d24b%4(Preferred)
IPv4 Address. . . . . : 192.168.2.41(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Wednesday, November 2, 2016 12:32:22 AM
Lease Expires . . . . . : Monday, November 14, 2016 12:32:22 AM
Default Gateway . . . . . : 192.168.2.99
DHCP Server . . . . . : 192.168.2.2
DHCPv6 IAID . . . . . : 205797350
DHCPv6 Client DUID. . . . . : 00-01-00-01-1F-96-45-81-44-37-E6-C0-9D-DF
DNS Servers . . . . . : 192.168.2.2
                       204.186.110.114
NetBIOS over Tcpip. . . . . : Enabled
```

Abb. 4-16 Ausgabe des Befehls `Ipconfig /all` auf einem Containerhost

Sehen Sie sich einen anderen Container auf demselben Host an: Er hat eine IP-Adresse im selben Netzwerk wie der erste Container. Die beiden Container können die Adressen des jeweils anderen anpingen, aber auch solche von Systemen außerhalb des Netzwerks 172.16.0.0/12.

Das ist möglich, weil das Feature *Container* und das Paket Docker standardmäßig mit Netzwerkadressübersetzung (Network Address Translation, NAT) arbeiten, um eine Netzwerkumgebung für die Container auf dem Host zu schaffen. NAT ist ein Routingverfahren. Es ersetzt die IP-Adressen in den Netzwerkpaketen, die von einem System generiert und an ein System gerichtet werden, sodass die Adressen so aussehen, als würden sie sich in einem anderen Netzwerk befinden.

Wenn Sie einen Computer im Hostnetzwerk aus einer Containersitzung heraus anpingen, modifiziert der Containerhost die Ping-Pakete und ersetzt die Adresse des Containers 172.25.117.12 durch seine eigene Adresse 192.168.2.43 in jedem Paket. Treffen die Antworten vom angepingten System ein, läuft der Vorgang in umgekehrter Richtung ab.

Das *Dockerd*-Modul erstellt standardmäßig ein NAT-Netzwerk, wenn es zum ersten Mal ausgeführt wird, und weist jedem Container eine Adresse in diesem NAT-Netzwerk zu. Die Verwendung der Netzwerkadresse 172.16.0.0/12 ist zudem eine Standardeinstellung, die in Docker codiert ist. Diese Standardeinstellungen können Sie aber ändern, indem Sie eine andere NAT-Adresse festlegen oder NAT überhaupt nicht verwenden.

Die Netzwerkadapter in den Containern sind selbstverständlich virtuell. In der weiter vorn gezeigten Konfiguration ist zu erkennen, dass der Adapter für diesen Container als *vEthernet (Container NIC 76b9f047)* gekennzeichnet ist. Auf dem Containerhost gibt es ebenfalls einen virtuellen Adapter, hier *vEthernet (HNS Internal NIC)* genannt. HNS ist der Hostnetzwerkdienst (Host Network Service), der die von Docker verwendete NAT-Implementierung darstellt. Wie die Ausgabe des Cmdlets *Get-VMSwitch* auf dem Containerhost zeigt oder wie Sie im Hyper-V-Manager im Manager für virtuelle Switches (siehe Abbildung 4–17) sehen können, hat Docker ebenfalls einen virtuellen Switch namens *nat* erstellt. Dabei handelt es sich um den Switch, mit dem alle Adapter in den Containern verbunden sind. Was das Netzwerk angeht, ist also festzustellen, dass Container fast so wie virtuelle Computer arbeiten.

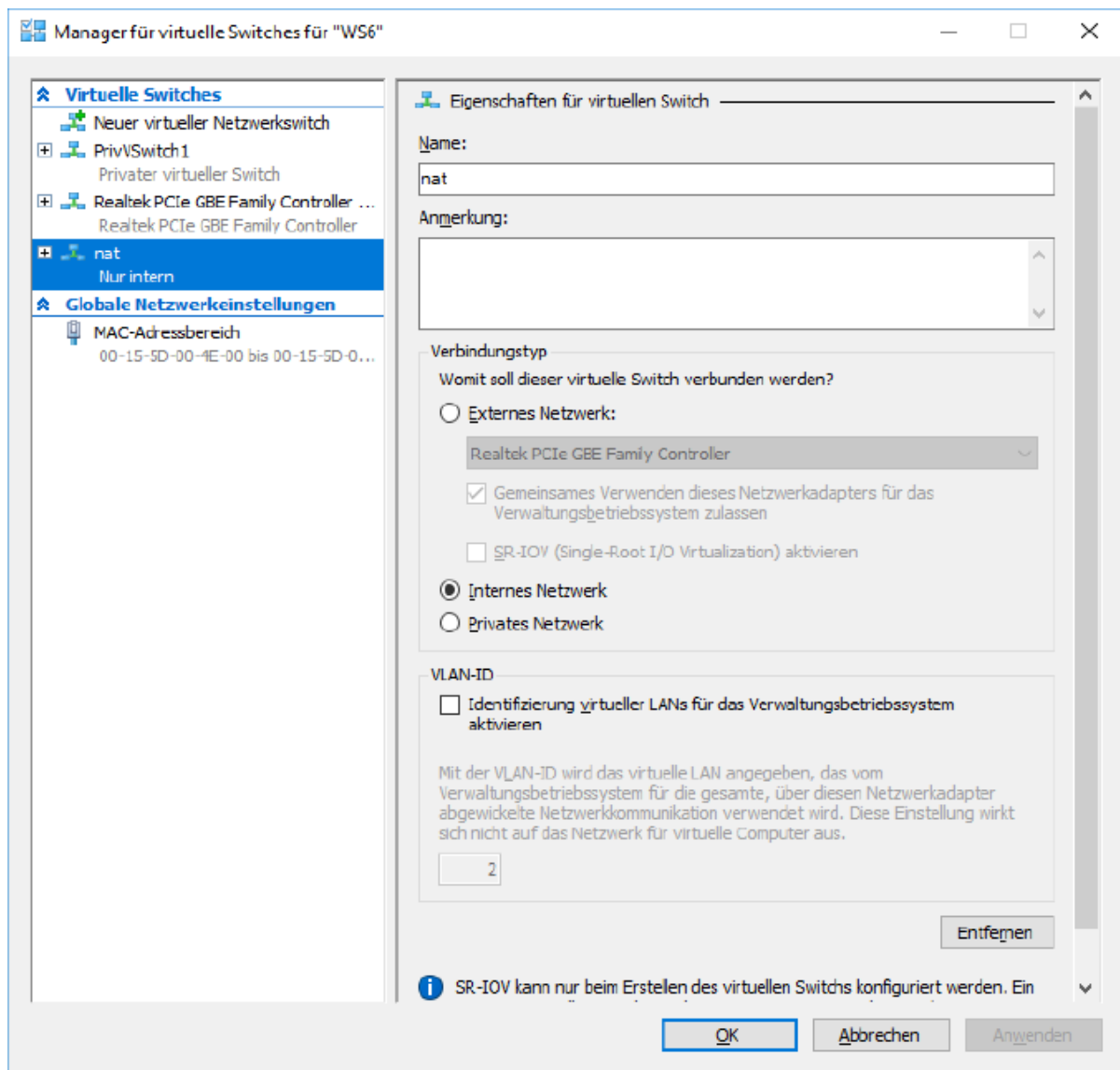


Abb. 4-17 Der Switch *nat* im Manager für virtuelle Switches

NAT-Standard Einstellungen anpassen

Wenn Sie für die NAT-Konfiguration von Docker eine andere Netzwerkadresse verwenden wollen, weil vielleicht ein Netzwerk mit derselben Adresse vorhanden ist, können Sie diese Adresse selbstverständlich ändern. Eine alternative Adresse legen Sie in der Konfigurationsdatei *daemon.json* fest, wie es bereits weiter vorn bei der Docker-Clientkonfiguration erläutert wurde.

Die Datei *daemon.json* ist eine einfache Textdatei, die Sie in dem Verzeichnis erstellen, in dem das Programm *Dockerd.exe* untergebracht ist. Eine alternative NAT-Netzwerkadresse legen Sie fest, indem Sie den folgenden Text in die Datei einfügen:

```
{ "fixed-cidr":"192.168.10.0/24" }
```

Für die NAT-Implementierung können Sie zwar beliebige Netzwerkadressen verwenden, doch um Adresskonflikte mit dem Internet zu verhindern, sollten Sie ein Netzwerk in einer der folgenden reservierten privaten Netzwerkadressen verwenden:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

Soll das *Dockerd*-Modul erst gar keine Netzwerkimplementierung erstellen, schreiben Sie den folgenden Text in die Datei *daemon.json*:

```
{ "bridge":"none" }
```

Wenn Ihre Container über Netzwerkkonnektivität verfügen sollen, müssen Sie in diesem Fall ein Containernetzwerk manuell einrichten.

Portzuordnungen

Wollen Sie in einem Container eine Serveranwendung ausführen, die Ports für eingehenden Clientverkehr offenlegt, sind sogenannte *Portzuordnungen* erforderlich. Damit kann der Containerhost, der den Clientverkehr empfängt, die Pakete an den passenden Port im Container, der die Anwendung ausführt, weiterleiten. Um Portzuordnungen zu verwenden, fügen Sie den Schalter *-p* an den Befehl *Docker Run* zusammen mit den Portnummern auf dem Containerhost bzw. dem Container wie im folgenden Beispiel an:

```
docker run -it
```

```
-p 8080:80 microsoft\windowsservercore powershell
```

In diesem Beispiel wird jeder Datenverkehr, der über den Port 8080 des Containerhosts eintrifft, an den Port 80 des Containers weitergeleitet. Port 80 ist

der bekannte Port für Webserververkehr und diese Anordnung versetzt den Container in die Lage, diesen Standardport zu nutzen, ohne ihn auf dem Containerhost vollkommen zu beanspruchen, der Port 80 möglicherweise für seinen eigenen Webserver benötigt.

Ein transparentes Netzwerk erstellen

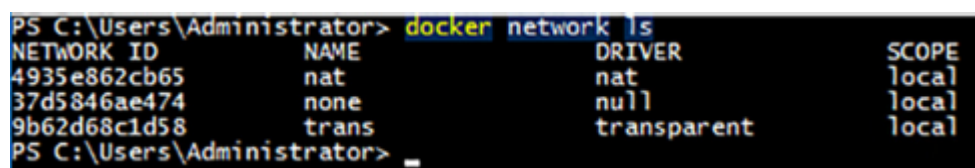
Anstatt NAT zu verwenden, können Sie auch ein transparentes Netzwerk erstellen, in dem die Container mit demselben Netzwerk wie der Containerhost verbunden sind. Wenn der Containerhost ein physischer Computer ist, sind die Container mit dem physischen Netzwerk verbunden. Ist der Containerhost ein virtueller Computer, sind die Container mit dem virtuellen Switch verbunden, den der virtuelle Computer verwendet.

Docker erstellt standardmäßig kein transparentes Netzwerk. Folglich müssen Sie es mit dem Befehl *Docker Network Create* erstellen, wie das folgende Beispiel zeigt:

```
docker network create -d transparent trans
```

Der Befehl in diesem Beispiel erstellt ein Netzwerk mithilfe des transparenten Treibers, der durch den Schalter *-d* gekennzeichnet wird, und weist ihm den Namen *trans* zu. Der folgende Befehl zeigt eine Liste aller Containernetzwerke an, die jetzt auch das eben erstellte Netzwerk *trans* enthält (siehe Abbildung 4-18).

```
docker network ls
```



```
PS C:\Users\Administrator> docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
4935e862cb65       nat                nat                 local
37d5846ae474       none              null                local
9b62d68c1d58       trans             transparent         local
PS C:\Users\Administrator> _
```

Abb. 4-18 Ausgabe des Befehls *Docker Network LS*

Ist das transparente Netzwerk erstellt, können Sie Container anlegen, die das Netzwerk verwenden. Dazu fügen Sie den Parameter *network* in den Befehl *Docker Run* wie im folgenden Beispiel hinzu:

```
docker run -it
```

```
    -network=trans microsoft/windowsservercore powershell
```

In diesem Container ausgeführt, zeigt der Befehl *Ipconfig /all*, dass der Container eine IP-Adresse im Netzwerk 10.0.0.0/24 besitzt, d. h. im selben Netzwerk, das der virtuelle Computer verwendet, der als Containerhost fungiert.

Beim Erstellen eines transparenten Netzwerks und der Container, die es verwenden, bekommen alle Teilnehmer die IP-Adressen von einem DHCP-Server im Containerhostnetzwerk zugeteilt. Ist jedoch kein DHCP-Server verfügbar, müssen Sie die Netzwerkadresseinstellungen festlegen, wenn Sie das Netzwerk erstellen, und die IP-Adressen aller Container manuell konfigurieren, indem Sie sie auf der *Docker Run*-Befehlszeile angeben.

Ein transparentes Netzwerk mit statischen IP-Adressen bauen Sie mit einem Befehl wie dem folgenden auf:

```
docker network create -d transparent
```

```
    -subnet=10.0.0.0/24
```

```
    -gateway=10.0.0.1
```

```
    trans
```

Um dann einen Container mit einer statischen IP-Adresse in diesem Netzwerk einzurichten, rufen Sie einen *Docker Run*-Befehl wie den folgenden auf:

```
docker run -it
```

```
    -network=trans
```

```
    -ip=10.0.0.16
```

```
    -dns=10.0.0.10 microsoft/windowsservercore powershell
```

Container-Datenvolumen verwalten

In manchen Fällen ist es erforderlich, Datendateien über Container hinweg zu bewahren. In Docker ist dies möglich, indem Sie Datenvolumen auf einem Container erzeugen, die einem Ordner auf dem Containerhost entsprechen. Nachdem Sie das Datenvolumen auf dem Container erstellt haben, sind die Daten, die Sie dort speichern, auch im korrespondierenden Ordner auf dem Containerhost zu finden. Umgekehrt gilt das ebenso: Sie können Dateien in den Ordner auf dem Host kopieren und im Container darauf zugreifen.

Datenvolumen bestehen unabhängig vom Container. Wenn Sie den Container löschen, verbleibt das Datenvolumen auf dem Containerhost. Dann können Sie den Containerhostordner in einem anderen Container bereitstellen, sodass Sie Ihre Daten über mehrere Iterationen einer Anwendung, die in Ihren Containern läuft, beibehalten können.

Um ein Datenvolumen zu erstellen, fügen Sie den Schalter `-v` an einen *Docker Run*-Befehl an, wie zum Beispiel:

```
docker run -it  
  
-v c:\appdata microsoft/windowsservercore  
  
powershell
```

Dieser Befehl erstellt einen Ordner `C:\appdata` im neuen Container und verknüpft ihn mit einem Unterordner in `C:\ProgramData\docker\volumes` auf dem Containerhost. Um den genauen Speicherort zu erfahren, führen Sie den folgenden Befehl aus und sehen sich den Abschnitt *Mounts* an (siehe Abbildung 4-19):

```
docker inspect dbf9674d13b9
```

```

"Mounts": [
  {
    "Type": "volume",
    "Name": "85dabc769744f3166aea3dd12a460c3a64f6c31fe5e64414eb56adfd1e87b04c",
    "Source": "C:\\ProgramData\\docker\\volumes\\85dabc769744f3166aea3dd12a460c3a64f6c31fe5e64414eb56adfd1e87b04c\\_data",
    "Destination": "c:\\appdata",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
],

```

Abb. 4-19 Ein Teil der Ausgabe des Befehls *Docker Inspect*

Der Abschnitt *Mounts* (Bereitstellungen), der nur ein kleiner Teil einer langen und umfangreichen Auflistung der Spezifikationen des Containers ist, enthält die Eigenschaften *Source* (Quelle) und *Destination* (Ziel). Die Eigenschaft *Destination* gibt den Ordernamen im Container an und *Source* ist der Ordner auf dem Containerhost. Um ein Datenvolume wiederzuverwenden, können Sie die Ordner sowohl für die Quelle als auch für das Ziel im *Docker Run*-Befehl wie im folgenden Beispiel angeben:

```
docker run -it
```

```
-v c:\sourcedata:c:\appdata microsoft/windowsservercore powershell
```

Wenn Sie ein Datenvolume erstellen und dabei auf dem Container einen Ordner angeben, der bereits Dateien enthält, wird der vorhandene Inhalt durch das Datenvolume überlagert, aber nicht gelöscht. Die betreffenden Dateien sind wieder zugänglich, wenn die Bereitstellung des Datenvolumens aufgehoben wird.

Docker erstellt Datenvolumen standardmäßig im Lese-/Schreibmodus. Um ein schreibgeschütztes (»read only«) Datenvolume anzulegen, fügen Sie *:ro* an den Ordernamen des Containers an, wie es das folgende Beispiel zeigt:

```
docker run -it
```

```
-v c:\appdata:ro microsoft/windowsservercore powershell
```

HINWEIS Ein Datenvolume hinzufügen

Um einem vorhandenen Container ein Datenvolume hinzuzufügen, haben Sie praktisch nur eine Möglichkeit: Sie speichern mit *Docker Commit* alle Änderungen, die Sie am vorhandenen Container

vorgenommen haben, in einem neuen Image und erstellen dann mit *Docker Run* einen neuen Container aus dem neuen Image, wobei Sie den Schalter *-v* angeben, um das Datenvolume hinzuzufügen.

Ressourcensteuerung verwalten

Wie bereits weiter vorn erwähnt, unterstützt der Befehl *Docker Run* viele Parameter und Schalter, von denen Sie in diesem Kapitel bereits einige kennengelernt haben. Zum Beispiel haben Sie gesehen, wie der Schalter *-it* einen interaktiven Container erstellt, der eine bestimmte Shell oder einen anderen Befehl ausführt. Um einen Container zu erstellen, der im Hintergrund – im sogenannten getrennten Modus – läuft, fügen Sie den Schalter *-d* an, wie es folgendes Beispiel zeigt:

```
docker run -d -p 80:80 microsoft/iis
```

Mit einem getrennten Container können Sie über Netzwerkverbindungen oder Dateisystemfreigaben interagieren. Es ist auch möglich, die Verbindung zum Container mit dem Befehl *Docker Attach* herzustellen.

Mit Containernamen arbeiten

Wenn Sie mit dem Befehl *Docker Run* einen Container erstellen, weist das *Dockerd*-Modul dem Container standardmäßig drei Bezeichner zu, wie Abbildung 4-20 zeigt:

- **Lange UUID** Eine 32-Bytes-Zeichenfolge, die mit 64 Hexadezimalziffern dargestellt wird, zum Beispiel:
0e38bdac48ca0120eff6491a7b9d1908e65180213b2c1707b924991ae8d1504f.
- **Kurze UUID** Die ersten sechs Bytes der langen UUID, dargestellt mit 12 Ziffern, zum Beispiel: 0e38bdac48ca.
- **Name** Ein zufällig gewählter Name, bestehend aus zwei Wörtern, die durch einen Unterstrich getrennt sind, zum Beispiel: *drunk_jones*.

```
PS C:\WINDOWS\system32> docker ps --no-trunc
CONTAINER ID        STATUS          PORTS           NAMES
0e38bdac48ca0120eff6491a7b9d1908e65180213b2c1707b924991ae8d1504f  Up 32 minutes  drunk_jones
PS C:\WINDOWS\system32>
IMAGE           COMMAND          CREATED
microsoft/windowsservercore  "powershell"    3 days ago
```

Abb. 4–20 Ausgabe des Befehls *Docker ps --no-trunc command*

Mit jedem dieser drei Bezeichner können Sie den Container auf der Befehlszeile referenzieren. Aber Sie haben auch die Möglichkeit, dem Container einen selbst gewählten Namen zuzuweisen, wenn Sie den Container erstellen. Dazu fügen Sie auf der *Docker Run*-Befehlszeile den Parameter *name* wie im folgenden Beispiel hinzu:

```
docker run -it microsoft/windowsservercore
powershell
-name core1
```

Arbeitsspeicher begrenzen

Über die Parameter des *Docker Run*-Befehls können Sie festlegen, wie viel Arbeitsspeicher ein Container verwenden darf. Standardmäßig dürfen Containerprozesse so viel Hostarbeitsspeicher verwenden und auslagern, wie sie benötigen. Wenn Sie mehrere Container auf demselben Host oder eine speicherhungrige Anwendung auf dem Host selbst ausführen, müssen Sie gegebenenfalls Beschränkungen hinsichtlich des Arbeitsspeichers, den bestimmte Container verwenden können, festlegen.

Die folgenden *Docker Run*-Parameter beziehen sich auf den Arbeitsspeicher:

- **-m** (oder **--memory**) Gibt die Menge des Arbeitsspeichers an, den der Container verwenden darf. Die Werte bestehen aus einer Ganzzahl und dem Einheitenzeichen b, k, m oder g (für Bytes, Kilobytes, Megabytes und Gigabytes).
- **-memory-swap** Gibt die Gesamtmenge des Arbeitsspeichers und des virtuellen Speichers an, den der Container verwenden kann. Die Werte bestehen aus einer Ganzzahl und dem Einheitenzeichen b, k, m oder g.
- **-memory-reservation** Spezifiziert eine weiche Grenze für den Arbeitsspeicher, den der Host für den Container reserviert, selbst wenn um

Systemspeicher konkurriert wird. Zum Beispiel können Sie mit dem Schalter `-m` eine harte Grenze von 1 GB und eine Speicherreservierung von 750 MB festlegen. Wenn andere Container oder Prozesse zusätzlichen Arbeitsspeicher benötigen, kann der Host bis zu 250 MB vom Arbeitsspeicher des Containers anfordern, lässt aber mindestens 750 MB unberührt. Die Werte bestehen aus einer Ganzzahl, die kleiner als die im Schalter `-m` ist, oder dem `—memory-swap`-Wert und dem Einheitenzeichen b, k, m oder g.

- **-kernel-memory** Gibt an, dass die mit dem Schalter `-m` festgelegte Speichergrenze für Kernelspeicher gilt. Die Werte bestehen aus einer Ganzzahl und dem Einheitenzeichen b, k, m oder g.
- **-oom-kill-disable** Hindert den Kernel daran, Containerprozesse abubrechen, wenn ein Fehler aufgrund von Speichermangel (Out Of Memory, OOM) auftritt. Verwenden Sie diese Option niemals ohne den Schalter `-m`, um ein Speicherlimit für den Container festzulegen. Andernfalls könnte der Kernel Prozesse auf dem Host beenden, wenn ein OOM-Fehler auftritt.

CPU-Zyklen begrenzen

Es sind auch Parameter vorgesehen, über die Sie die Anzahl der CPU-Zyklen begrenzen können, die einem Container zugeteilt werden. Standardmäßig teilen sich alle Container auf einem Host die verfügbaren Zyklen gleichmäßig untereinander. Mit den Zyklusparametern lassen sich den Containern Prioritäten zuweisen, die bei CPU-Auslastung wirksam werden.

Den Zugriff auf CPUs können Sie mit folgenden *Docker Run*-Parametern steuern:

- **-c** (oder **—cpu-shares**) Spezifiziert mit einem Wert von 0 bis 1024 das Gewicht des Containers im Rennen um CPU-Zyklen. Die tatsächliche Anzahl der Prozessorzyklen, die ein Container erhält, hängt von der Anzahl der Container, die auf dem Host laufen, und ihrer Gewichte ab.
- **-cpuset-cpus** Spezifiziert in einem Multiprozessor-Hostsystem, welche CPUs der Container verwenden kann. Die Werte bestehen aus Ganzzahlen, die durch Kommas voneinander getrennt sind und die CPUs im Hostcomputer darstellen.
- **-cpuset-mems** Gibt an, welche Knoten auf einem NUMA-Host der Container verwenden kann. Die Werte bestehen aus Ganzzahlen, die durch

Kommas voneinander getrennt sind und die CPUs im Hostcomputer darstellen.

Neue Container-Images mit Dockerfile erstellen

Wurde ein Container geändert, seit Sie ihn mit dem Befehl *Docker Run* erstellt haben, lassen sich diese Änderungen auch speichern. Das ist beispielsweise möglich, wenn Sie ein neues Container-Image mit dem Befehl *Docker Commit* erstellen. Empfohlen wird jedoch, Container-Images von Grund auf neu zu erstellen, und zwar mit einem als *Dockerfile* bezeichneten Skript.

Eine *Dockerfile-Datei* ist eine einfache Textdatei mit dem Namen *dockerfile*. Sie enthält die erforderlichen Befehle, um ein neues Image zu erstellen. Die fertiggestellte Dockerfile-Datei führen Sie dann mit dem Befehl *Docker Build* aus, um die neue Datei zu erzeugen. Bei Dockerfile handelt es sich einfach um eine Methode, die die Schritte automatisiert, mit denen Sie einen Container sonst manuell ändern würden. Wenn Sie den Befehl *Docker Build* mit der Dockerfile-Datei aufrufen, führt das *Dockerd*-Modul jeden Befehl im Skript aus, indem es einen Container erstellt, die angegebenen Änderungen vornimmt und mit einem *Docker Commit*-Befehl die Änderungen als neues Image speichert.

Eine Dockerfile-Datei besteht aus Befehlen wie FROM oder RUN und einer Anweisung für jeden Befehl. Es hat sich eingebürgert, die Befehle großzuschreiben. In das Skript können Sie Kommentare einfügen, die mit einem Nummernzeichen (#) eingeleitet werden.

Das folgende Beispiel zeigt eine einfache Dockerfile-Datei:

```
#install DHCP server

FROM microsoft/windowsservercore

RUN powershell

    -command install-windowsfeature dhcp

    -includemanagementtools
```

```
RUN powershell
```

```
-configurationname microsoft.powershell
```

```
-command add-dhcpserverv4scope
```

```
-state active
```

```
-activatepolicies $true
```

```
-name scopetest
```

```
-startrange 10.0.0.100
```

```
-endrange 10.0.0.200
```

```
-subnetmask 255.255.255.0
```

```
RUN md boot
```

```
COPY ./bootfile.wim c:/boot/ CMD powershell
```

Die Elemente in diesem Beispiel bedeuten:

- Der FROM-Befehl spezifiziert das Basisimage, von dem das neue Image erstellt wird. In diesem Fall beginnt das neue Image mit dem Image *microsoft/windowsservercore*.
- Der erste RUN-Befehl öffnet eine PowerShell-Sitzung und ruft das Cmdlet *Install-WindowsFeature* auf, um die Rolle DHCP zu installieren.
- Der zweite RUN-Befehl richtet mit dem Cmdlet *Add-DhcpServerv4Scope* einen neuen Bereich auf dem DHCP-Server ein.
- Der dritte RUN-Befehl legt ein neues Verzeichnis namens *boot* an.
- Der COPY-Befehl kopiert eine Datei *bootfile.wim* aus dem aktuellen Ordner auf dem Containerhost in den Ordner *C:\boot* auf dem Container.

- Der CMD-Befehl öffnet eine PowerShell-Sitzung, wenn das Image ausgeführt wird.

Nachdem Sie das *dockerfile*-Skript erstellt haben, erzeugen Sie mit dem Befehl *Docker Build* das neue Image wie im folgenden Beispiel:

```
docker build -t dhcp .
```

Dieser Befehl liest die Dockerfile-Datei aus dem aktuellen Verzeichnis und erstellt ein Image namens *dhcp*. Wenn das *Dockerd*-Modul das Image erstellt, zeigt es die Ergebnisse jedes Befehls und die IDs der temporär angelegten Container (siehe Abbildung 4–21). Nachdem Sie das Image erstellt haben, können Sie davon mit dem Befehl *Docker Run* in der üblichen Art und Weise einen Container einrichten.

```
PS C:\temp> docker build --no-cache --force-rm --tag dhcp .
Sending build context to Docker daemon 8.192 kB
Step 1/6 : FROM microsoft/windowsservercore
--> 93a9c37b36d0
Step 2/6 : RUN powershell install-windowsfeature dhcp -includemanagementtools
--> Running in 71a7f8f39e4f
Success Restart Needed Exit Code      Feature Result
-----
True      No              Success          (DHCP Server)

--> 92a23f3b115c
Removing intermediate container 71a7f8f39e4f
Step 3/6 : RUN powershell -configurationname microsoft.powershell -command Add-dhcpserverv4scope -state active -activatepolicies $true -name scopetest -startrange 172.25.112.10 -endrange 172.25.112.20 -subnetmask 255.255.240.0
--> Running in 41b6925364f7
--> bef09e926548
Removing intermediate container 41b6925364f7
Step 4/6 : RUN md boots
--> Running in 2d6d8cb82ee7
--> 0da73c9a2791
Removing intermediate container 2d6d8cb82ee7
Step 5/6 : COPY ./bootfile.win c:/boots/
--> 365faa0f2ee0
Removing intermediate container 53498275c410
Step 6/6 : CMD powershell
--> Running in 4f23e9326a9d
--> bce3bfd56322
Removing intermediate container 4f23e9326a9d
Successfully built bce3bfd56322
PS C:\temp>
```

Abb. 4–21 Ausgabe des Befehls *Docker Build*

Die Dockerfile-Datei im Beispiel ist sehr einfach, doch können solche Dateien auch wesentlich länger und komplexer sein.

Schnelltest

Mit welchem der folgenden Docker-Befehle können Sie neue Container-Imagedateien erstellen?

1. Docker Run
2. Docker Commit
3. Docker Build
4. Docker Images

Antwort für den Schnelltest

Die Antworten 2 und 3 sind richtig. Mit dem Befehl *Docker Commit* erzeugt man ein neues Image aus einem vorhandenen Container. Der Befehl *Docker Build* erzeugt ein neues Container-Image entsprechend den Befehlen in einer Dockerfile-Datei.

Container-Images mit DockerHub-Repository für öffentliche und private Szenarien verwalten

DockerHub ist ein öffentliches Repository, mit dem Sie Container-Images speichern und verteilen können. Wenn Sie Container-Images mit dem Befehl *Docker Pull* herunterladen, kommen sie standardmäßig von DockerHub, sofern Sie im Befehl kein anderes Repository angeben. Mit dem Befehl *Docker Push* können Sie Images auch hochladen.

Wenn Sie Images zu DockerHub hochladen, können Sie sie mit Ihren Kollegen und sogar mit sich selbst teilen, sodass Sie Dateien nicht manuell übertragen müssen, um ein Container-Image auf einem anderen Host bereitzustellen.

Bevor Sie Images auf DockerHub hochladen können, müssen Sie sich auf der Site <http://hub.docker.com> registrieren. Nachdem das erledigt ist, wird Ihr Benutzername zum Namen Ihres Repositories. Zum Beispiel ist das Image *microsoft/windowsservercore*, das Sie vorher heruntergeladen haben, ein Image namens *windowsservercore* im Microsoft-Repository. Lautet Ihr DockerHub-Benutzername *hholt*, beginnen alle Ihre Images mit diesem Repository-Namen, gefolgt vom Imagenamen, zum Beispiel:

```
hholt/nano1
```

Sobald Sie über ein Konto verfügen, müssen Sie sich beim DockerHub-Dienst von der Befehlszeile aus anmelden, bevor Sie Images hochladen können. Hierfür verwenden Sie den folgenden Befehl:

```
docker login
```

Docker fragt Ihren Benutzernamen und das Kennwort ab und bietet dann Upload-Zugriff auf Ihr Repository.

Nach Images suchen

DockerHub können Sie nach Images durchsuchen. Dazu gehen Sie auf die Website, wie sie Abbildung 4-22 zeigt. Diese Oberfläche bietet die neuesten Informationen über das Image sowie Kommentare von anderen Benutzern in der Docker-Community.



Abb. 4-22 Screenshot einer DockerHub-Websuche

Den DockerHub können Sie auch von der Befehlszeile aus durchsuchen, und zwar mit dem Befehl *Docker Search*, zum Beispiel:

```
docker search microsoft --no-trunc
```

Der Parameter *no-trunc* verhindert, dass der Befehl die Imagebeschreibungen abschneidet (siehe Abbildung 4-23).

```

PS C:\temp> docker search microsoft --no-trunc
NAME                DESCRIPTION                STARS    OFFICIAL    AUTOMATED
microsoft/aspnet    ASP.NET is an open source server-side Web application framework    488      [OK]
microsoft/dotnet    Official images for .NET Core for Linux and Windows Server 2016 Nano Server    311      [OK]
mono                Mono is an open source implementation of Microsoft's .NET Framework    196      [OK]
microsoft/windowsservercore    Windows Server 2016 Server Core base OS image for Windows containers    71
microsoft/naoserver    Windows Server 2016 Nano Server base OS image for Windows containers    68
microsoft/azure-cli    Docker image for Microsoft Azure Command Line Interface    66      [OK]
microsoft/iis        Internet Information Services (IIS) installed in a Windows Server Core based container    50
microsoft/mssql-server-2014-express-windows    Microsoft SQL Server 2014 Express installed in Windows Server Core based containers.    42
microsoft/mssql-server-2016-express-windows    Microsoft SQL Server 2016 Express installed in Windows Server Core based containers.    29
microsoft/aspnetcore    Official images for running compiled ASP.NET Core applications.    29      [OK]
microsoft/dotnet-framework    The official Docker images for .NET Framework on Windows Server 2016 Server Core.    11
microsoft/powershell    Official PowerShell Core releases from https://github.com/PowerShell/PowerShell/releases    9      [OK]
microsoft/oms        Monitor your containers using the Operations Management Suite (OMS). For minimal OS like CoreOS.    7      [OK]
microsoft/aspnetcore-build    Official images for building ASP.NET Core applications.    7      [OK]
microsoft/applicationinsights    Application Insights for Docker helps you monitor your containerized applications.    4      [OK]
microsoft/vsts-agent    Official images for the Visual Studio Team Services (VSTS) agent.    4
microsoft/dotnet35    The .NET Framework 3.5 image has moved to microsoft/dotnet-framework:3.5    4
microsoft/dotnet-nightly    Preview bits of the .NET Core CLI    2      [OK]
microsoft/powershell-nightly    Nightly builds of PowerShell Core for CI    2      [OK]
cuginec/microsoft-pre09533    Images to build preview versions of ASP.NET Core applications.    0
microsoft/aspnetcore-build-nightly    Microsoft-malmo - artificial intelligence - training agent on MINECRAFT    0      [OK]
berlius/microsoft-malmo    .NET Core Docker Samples    0
microsoft/dotnet-samples    CNTK    0      [OK]
microsoft/cntk        Microsoft Test Repo    0
dreher/microsoft
PS C:\temp>

```

Abb. 4-23 Ausgabe des Befehls *Docker Search*

Images hochladen

Möchten Sie Ihre Images in das Repository hochladen, rufen Sie den Befehl *Docker Push* wie im folgenden Beispiel auf:

```
docker push hholzt/nano1
```

Standardmäßig lädt der Befehl *Docker Push* das angegebene Image in Ihr öffentliches Repository auf dem DockerHub hoch, wie Abbildung 4-24 zeigt. Somit kann jeder auf die hochgeladenen Images zugreifen.

```

PS C:\temp> docker push craigz3/nano1
The push refers to a repository [docker.io/craigz3/nano1]
0bc9597871ad: Pushed
2c195a33d84d: Skipped foreign layer
342d4e407550: Skipped foreign layer
0.9: digest: sha256:1518e997672c384bad70aeb897de64689fef78fcb420e9fda86b933e25f1cd6b size: 1155
PS C:\temp>

```

Abb. 4-24 Ausgabe des Befehls *Docker Push*

Weil Docker Open-Source-Software ist, macht das Freigeben von Images und Code für die Community einen großen Teil der Firmenphilosophie aus. Es ist aber auch möglich, private Repositories zu erstellen, die Sie für eine unbegrenzte Anzahl von ausgewählten Mitarbeitern freigeben können. Dadurch lässt sich DockerHub auch für sichere Projekte der Anwendungsentwicklung einsetzen oder allgemein für jede Situation, in der Sie ein Image nicht für die Öffentlichkeit bereitstellen wollen. DockerHub bietet ein einzelnes privates Repository im Rahmen des kostenlosen Dienstes, für zusätzliche Repositories ist ein kostenpflichtiges Abonnement erforderlich.

Außer dem Speichern und Bereitstellen von Images bietet DockerHub noch weitere Dienste, beispielsweise automatisierte Builds. Wenn Sie eine Dockerfile-Datei und alle anderen erforderlichen Dateien in ein Repository hochladen,

können Sie DockerHub so konfigurieren, dass Builds nach Ihren konkreten Spezifikationen automatisch ausgeführt werden. Die Codedateien sind für Ihre Mitarbeiter zugänglich und neue Builds können stattfinden, sobald sich der Code geändert hat.

Container-Images mit Microsoft Azure verwalten

Container können Sie nicht nur lokal erstellen, sondern sie auch auf Microsoft Azure verwenden. Wenn Sie einen virtuellen Computer mit Windows Server 2016 auf Azure einrichten, können Sie Container genauso wie auf einem lokalen Server erstellen und verwalten. Darüber hinaus bietet Azure den Dienst Azure Container Service (ACS), mit dem Sie einen Cluster von virtuellen Computern erstellen, konfigurieren und verwalten können, um Container-basierte Anwendungen mithilfe verschiedener Open-Source-Technologien auszuführen.

Microsoft Azure ist ein abonnementbasierter Clouddienst, der Sie in die Lage versetzt, virtuelle Computer und Anwendungen bereitzustellen und sie in Ihr vorhandenes Unternehmen zu integrieren. Für eine monatliche Gebühr können Sie einen virtuellen Windows Server 2016-Computer erstellen (siehe Abbildung 4-25). Nachdem Sie den virtuellen Computer eingerichtet haben, können Sie das Feature *Container* und das Modul *Docker* installieren. Container und Images, die Sie auf einem virtuellen Azure-Computer installieren, sind vollkommen kompatibel mit den Docker-Implementierungen auf Ihren lokalen Computern.

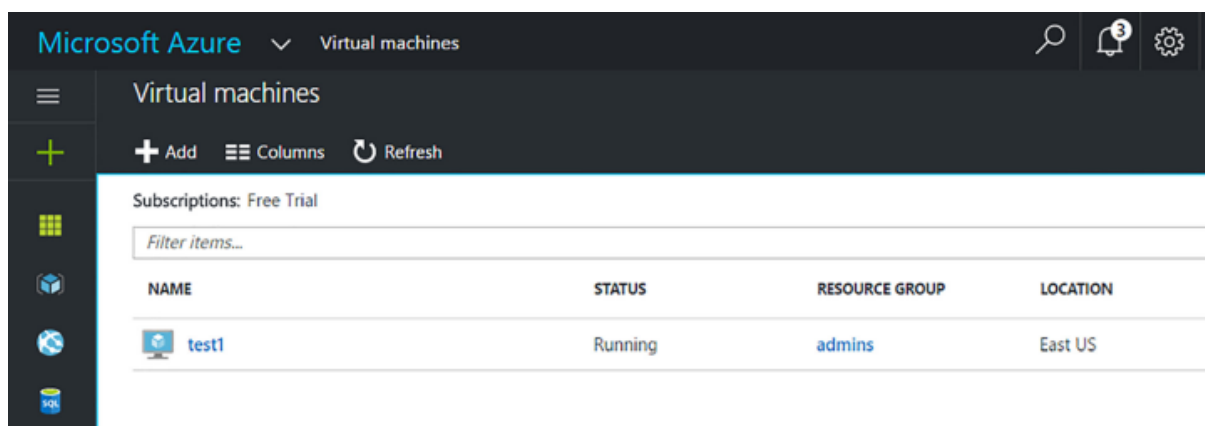


Abb. 4-25 Microsoft Azure Resource Center

Kapitelzusammenfassung

- Container basieren auf Images. Um einen Container zu erstellen, führen Sie ein Image aus, und Sie erstellen ein Image, indem Sie den Inhalt eines Containers speichern.
- Windows Server 2016 enthält das Feature *Container*, das die Unterstützungsumgebung für die Docker-Plattform liefert.
- Sowohl die Server Core- als auch die Nanoserver-Installationsoptionen unterstützen das Erstellen von Windows Server- und Hyper-V-Containern. In Nanoserver können Sie den *Docker.exe*-Client auf einem Remotesystem ausführen.
- Docker ist eine Open-Source-Container-Lösung, die aus zwei Dateien besteht: *Dockerd.exe* ist das Modul, das als Dienst in Windows läuft, und *Docker.exe* ist der Befehlszeilenclient, der das *Dockerd*-Modul steuert.
- Mit einer Textdatei namens *daemon.json* können Sie die Startoptionen für das *Dockerd*-Modul konfigurieren.
- Der Docker-Client ist nicht die einzige Möglichkeit, das Docker-Modul zu steuern. Die gleichen Aufgaben können Sie auch mit dem Docker-Modul für Windows PowerShell durchführen.
- Mit dem Befehl *Docker Pull* lassen sich Images vom DockerHub herunterladen.
- Tags (Markierungen) sind Versionsindikatoren, mit denen Entwickler die Builds oder Versionen eines Container-Images nachverfolgen können. Tag-Werte weisen Sie mit dem Befehl *Docker Tag* zu.
- Ein Container-Image lässt sich mit dem Befehl *Docker RMI* deinstallieren.
- Um einen Windows Server-Container zu erstellen, führen Sie den Befehl *Docker Run* mit dem Namen eines Container-Images aus.
- Der Ablauf für das Erstellen eines Hyper-V-Containers mithilfe von Docker unterscheidet sich vom Ablauf für Windows Server-Container nur darin, dass der Parameter *—isolation* anzugeben ist.
- Der *Docker.exe*-Client erlaubt es Ihnen, Container zu steuern, indem Sie sie starten, anhalten, sichern und entfernen.

- Das Docker-Modul für Windows PowerShell bietet eine Alternative zum *Docker.exe*-Client und kann die meisten (wenn nicht sogar alle) Funktionen ausführen.
- Standardmäßig verwendet Docker Netzwerkadressübersetzung (NAT), um den Netzwerkzugriff für Container bereitzustellen. Allerdings können Sie die Standardeinstellungen überschreiben und Container als Teil Ihres größeren Netzwerks konfigurieren.
- Mit Docker haben Sie die Möglichkeit, Datenvolumes zu erstellen, die auf dem Containerhost existieren, und sie einem Container hinzuzufügen. Datenvolumes verbleiben auf dem Containerhost, selbst wenn Sie den Container löschen.
- Mit Parametern auf der Befehlszeile von *Docker Run* können Sie die Arbeitsspeicher- und CPU-Ressourcen begrenzen, die ein Container nutzen darf.
- Eine Dockerfile-Datei ist ein Skript, das Befehle für das Erstellen eines neuen Container-Images enthält. Mit dem Befehl *Docker Build* führen Sie das Skript aus und erzeugen das Image.
- DockerHub ist ein kostenloses, cloudbasiertes Repository, in das Sie Ihre Container-Images hochladen können.
- Microsoft Azure ermöglicht es, virtuelle Computer zu erstellen, die Sie als Containerhosts verwenden können.

Gedankenexperiment

In diesem Gedankenexperiment wenden Sie Ihre Fähigkeiten und Kenntnisse an, die Sie sich im Rahmen dieses Kapitels angeeignet haben. Die Antwort zu diesem Gedankenexperiment finden Sie im nächsten Abschnitt.

Ralph möchte einen virtuellen Computer namens Core1 erstellen, der als Containerhost sowohl für Windows Server- als auch für Hyper-V-Container fungiert. Um den Containerhost zu erstellen, plant er, die folgenden Aufgaben auszuführen:

- Einen virtuellen Computer erstellen
- Den virtuellen Computer mit 4 GB Arbeitsspeicher, zwei virtuellen Prozessoren und aktiviertem Spoofing von MAC-Adressen

konfigurieren

- Windows Server 2016 auf dem virtuellen Computer installieren
- Das Feature *Container* installieren
- Die Rolle *Hyper-V* installieren
- Das Modul *dockermstprovider* installieren
- Das Paket *Docker* installieren
- Das Server Core-Image von DockerHub abrufen
- Container mit dem Befehl *Docker Run* erstellen

Welchen Schritt hat Ralph vergessen, was ihn daran hindert, die benötigten Container zu erstellen? Welche Aufgabe muss er ausführen, um seinen Plan fertigzustellen, und wann sollte er ihn fertigstellen?

Antwort für das Gedankenexperiment

Dieser Abschnitt enthält die Lösung für das Gedankenexperiment.

Ralph hat vergessen, die Virtualisierungserweiterungen des Prozessors im physischen Computer für den virtuellen Computer offenzulegen, damit er die Hyper-V-Rolle ausführen kann. Hierzu muss er den folgenden Befehl in einer PowerShell-Sitzung ausführen, nachdem er den virtuellen Computer erstellt hat und bevor er ihn startet:

```
set-vmprocessor -vmname server1
```

```
-exposevirtualizationextensions $true
```

Index

.avhd 258
.avhdx 258
.cab 83
.inf 82
.msu 83
.vhd 49
.vhds 397
.vhdx 49
.vmc 204
.vsv 204
/image 82
\$PSVersionTable 64

A

ACE

- Typen 128
- verweigern 128
- zulassen 128

ACS (Azure Container Service) 332

Active Directory

- Cluster, von ~ getrennter 382
- Objekte wiederherstellen 479
- Sicherung 479

Active Directory Domain Services (AD DS) 43

Active Directory-Domänendienste 479

AD DS (Active Directory Domain Services) 43

Adapter

- emulierter 281
- Fibre Channel- 261

- Legacy- 281
- Add-CauClusterRole 375
- Add-ClusterScaleOutFileServer 385
- Add-ClusterSharedVolume 172, 380
- Add-ClusterVirtualMachineRole 344
- Add-ClusterVMMonitoredItem 410
- Add-Computer 22
- Add-ContainerImageTag 310
- Add-VMHardDiskDrive 255
- Add-VMNetworkAdapter 268
- Administratoren 307
- Adprep.exe 43
- Affinität 415, 432
- Aggressivität 416
- Aktiv/Aktiv-System 383
- Aktiv/Passiv-System 383
- Aktivierung
 - automatische 45
 - KMS 40
 - Lizenzierung 39
 - MAK 39
 - Modell bestimmen 38
 - über Active Directory 43
- Aktivierungsintervall 41
- Aktivierungsschwellenwert 40
- Aktualisierung 30
 - Installation ausführen 33
 - NIC-Teamvorgang 32
 - vorbereiten 32
- Analysepunkte 177
- Antimalware 459
- Anwendungen
 - Born-In-The-Cloud 47
 - Cloud-orientierte 47

- Anwendungsmigration 388
- Appcmd.exe 483
- Arbeitsauslastungen 179
- Arbeitsbelastungen 77
- Arbeitsgruppencluster 354
- Arbeitsspeicher
 - dynamisch hinzufügen/entfernen 207
 - dynamischen ~ konfigurieren 208
 - Verluste 496
- Arbeitsspeicher *siehe auch* Speicher
- Arbeitsspeicher, dynamischer
 - Balloon-Treiber 211
 - Einstellungen 208
- Arbeitsspeicherpuffer 209
- Assistenten
 - Bearbeiten virtueller Festplatten 256
 - Clustererstellung 358
 - Einmalsicherung 467
 - Erstellen neuer einfacher Volumes 93
 - Freigaben 109
 - Hinzufügen von Rollen und Features 13, 190
 - hohe Verfügbarkeit 384
 - Inventory And Assessment- 76
 - Konfiguration von Windows Server Update Services 446, 447
 - Konfigurationsüberprüfung 352
 - Konfigurieren der Selbstaktualisierungsoptionen 375
 - Konfigurieren des Clusterquorums 360, 362, 394
 - neue Freigaben 385
 - neue Speicherpools 371
 - neue virtuelle Computer 204, 266
 - neue virtuelle Datenträger 141, 390
 - neue virtuelle Festplatten 244, 398
 - neue virtuelle iSCSI-Datenträger 153
 - neue Volumes 390

- neuen Datensammlersatz erstellen 491
- Performance Metrics 75
- Replikation für 340
- Server Virtualization And Consolidation 76
- Setup für das Microsoft Assessment and Planning Toolkit 71
- Sicherungszeitplan 471
- Speicherpools 139, 390
- Verschieben von 345
- Virtuellen Computer importieren 236, 422
- Volumenaktivierungstools 44
- Wiederherstellung 475
- Attach-Container 318
- Aufzählung, zugriffsbasierte 111
- Ausrichtung
 - NUMA 212
- Authentifizierung
 - Credential Security Support Provider (CredSSP) 196
 - Livemigration 348
- Autorisierung 126
- Autorisierungs-Manager 194
- AVMA (Automatic Virtual Machine Activation) 45
 - Schlüssel 45
- AVMAkey 46
- Azman.msc 194
- Azure Container Service (ACS) 332
- B**
- Backups 367
- Balloon-Treiber 211, 218
- Bandbreite 279
 - Gewichtung 290
 - maximale 289
 - minimale 289
- Bandbreitenverwaltung 289

- Bare-Metal-Installation 7
- Basisbetriebssystem, Installation 308
- Bedrohungen 459
- Behandlungspriorität 434
- Belastung beim Herunterfahren 424
- Benutzer, Prinzipal 117
- Benutzeroberfläche
 - grafische 20
 - minimale Serverschnittstelle 20
- Berechtigungen
 - anpassen 117, 132
 - Datei- 124
 - effektiver Zugriff 129
 - erweiterte 126
 - grundlegende 126
 - NTFS 131
 - Ordner- 124
 - verweigern 128
 - zulassen 128
 - Zuweisen erweiterter NTFS- 134
- Berechtigungsvererbung 128
- Bereitstellung, Windows Deployment Services (WDS) 12
- Bereitstellungen
 - Arbeitsbelastungen 77
 - FreeBSD 68
 - Linux 68
- Berichte, Leistungsüberwachung 487
- Betriebssysteme
 - Basis-installieren 308
 - Gast- 186
 - herunterfahren 219
 - Image deinstallieren 311
 - Modus, gemischter 376
- Bildschirmauflösung Computer, virtueller 225

BIS (FreeBSD Integration Services) 68

Blob-Datei erstellen 51

Blockgröße 227

BlockSizeBytes 227

Block-SmbShareAccess 122

Blockspeicher 177

Born-In-The-Cloud 47

Build, Docker 310

C

CAU (Cluster-Aware Updating) 372

Checkpoints 258

Checkpoint-VM 258

Chkdsk.exe 106

Churn 178

Clientseitige Zielzuordnung 453

Close-SmbOpenFile 121

Close-SmbSession 120

Clouds, Ressourcenmessung 216

Cloudzeugen 359

- Failovercluster 391

- Speicherkonto 393

Cluster

- Arbeitsgruppen- 354

- Ausführungsmodus 430

- Belastung beim Herunterfahren 424

- DrainOnShutdown 424

- Fehlerdomänen 414

- Gast- 387

- Internetname 429

- Lastenausgleich 416

- Metrikwerte 383

- Multi-domain 354

- Netzwerk konfigurieren 364

- Node Fairness 416
- Single-domain 354
- Stretch- 413
- Upgrade, paralleles 376
- von Active Directory getrennter 382
- Zugriffspunkt 382
- Cluster Shared Volumes (CSVs) 379
- ClusterAwareUpdating 374
- Clusterbetriebsmodus 435
- Clustererstellungs-Assistent 358
- Clustering
 - geschachteltes 388
 - Resilienz 395
- Clusternamenobjekt 353, 382
- Clusternetzwerkname 354
- Clusterrollen
 - Dateiserver 407
 - Hyper-V-Replikatbroker 337
- Clusterspeicherplätze 388
 - Failovercluster-Manager 390
- Clustervolumes 377
- Cmdlets
 - Add-CauClusterRole 375
 - Add-ClusterScaleOutFileServer 385
 - Add-ClusterSharedVolume 172, 380
 - Add-ClusterVirtualMachineRole 344
 - Add-ClusterVMMonitoredItem 410
 - Add-Computer 22
 - Add-ContainerImageTag 310
 - Add-VMHardDiskDrive 255
 - Add-VMNetworkAdapter 268
 - Attach-Container 318
 - Block-SmbShareAccess 122
 - Checkpoint-VM 258

Close-SmbOpenFile 121
Close-SmbSession 120
Cluster konfigurieren 172
Commit-Container 318
Convert-VHD 397
Copy-Item 202
Copy-VMFile 220
Dateien kopieren 202
Dateien schließen 121
Datenträgertypen 149
Direkte Speicherplätze aktivieren 401
Disable-PnpDevice 240
Disable-VMResourceMetering 218
Disconnect-PSSession 24
Disconnect-VMNetworkAdapter 239
Dismount-DiskImage 104
Dismount-VHD 104, 253
Dismount-VmHostAssignableDevice 240
Docker 308
DockerMsftProvider 303
Edit-NanoServerImage 55, 86
Enable-ClusterStorageSpacesDirect 401
Enable-DedupVolume 177
Enable-NetAdapterRdma 288
enable-netadaptervmq 286
Enable-NetFirewallRule 339
Enable-NetQosFlowControl 162
Enable-PSRemoting 198
enable-vmresourcemetering 217
Enter-PSSession 24, 64, 201
Ereignisprotokolle überprüfen 173
Exit-PSSession 24
exportieren 235
Export-VM 236, 422

Features hinzufügen 55
Festplatten, virtuelle 246
Freigabe beenden 121
Freigaben erstellen 118
Get-ClusterNetwork 365, 366, 383
Get-Command 23
Get-Container 318
Get-DedupStatus 180
Get-Disk 255
Get-help 23
Get-MpComputerStatus 461
Get-NetAdapter 21
GetNetAdapterRdma 288
Get-NetAdapterVmq 286
Get-NetAdapterVmqQueue 287
Get-PnpDevice 239
Get-PnpDeviceProperty 240
Get-SmbClientConfiguration 123
Get-SmbOpenFile 120
Get-SmbSession 119
Get-SmbShareAccess 121
Get-SRGroup 173
Get-VMCheckpoint 260
Get-VMHostSupportedVersion 235
Get-WindowsFeature 17
Get-WinEvent 173
Grant-SmbShareAccess 121
Grant-SRAccess 172
Hochverfügbarkeit 344
Import-VM 238, 422
Install-WindowsFeature 17, 36, 190, 191, 427, 446
InstallWindowsFeature 253
InvokeCauRun 375
Invoke-Command 201

Livemigration 346
Livemigration initiieren 418
Measure-VM 217, 265
Merge-VHD 258
Metrikwerte 383
Mount-DiskImage 104
Mount-VHD 104, 253
Move-ClusterVirtualMachineRole 418
Move-VM 346
Nanoserver 48
Netzwerkkarte erstellen 268
Netzwerklastenausgleich 427
New-Cluster 357
New-ClusterFaultDomain 414
New-NanoServerImage 48, 301
New-NetIpAddress 21
New-NetQosPolicy 161
New-NetQosTrafficClass 161
New-PSSession 24, 63, 198
New-SmbShare 118
New-SRPartnership 172
New-VHD 101, 246, 252
New-VMSwitch 274, 288
New-Volume 401
Optimize-VHD 257
Pass-Through-Datenträger 255
PFC aktivieren 162
PowerShell-Befehl auf VM ausführen 201
Prüfpunkte 258
Prüfpunkte anwenden 260
QoS-Richtlinien 161
RDMA aktivieren 288
Remove-Container 319
Remove-ContainerImage 311

Remove-Smb-Share 121
Replikationspartnerschaft 172
Reset-VMResourceMetering 218
Resize-VHD 257
Ressourcenmessung 217
Restore-VMCheckpoint 260
Revoke-SmbShareAccess 121
Rollen hinzufügen 55
Set-ClusterFaultDomain 414
Set-ClusterQuorum 359
Set-DnsClientGlobalSettings 357
Set-DnsClientServerAddress 22
Set-FileStorageTier 149
Set-MpPreference 462
Set-NetAdapterVmq 287
Set-NetFirewallRule 27
Set-NetQoSbcxSetting 161
Set-SmbPathAcl 386
Set-SmbServer Configuration 122
Set-VM 261
Set-VMFirmware 232, 234
Set-VMHardDiskDrive 265
Set-VMMemory 208
Set-VMNetworkAdapter 290
Set-VmReplicationServer 338
Sitzungen beenden 120
Sitzungen verwalten 119
Sitzungen, persistente 201
Speicherzuordnung 208
Standorte definieren 414
Start-Container 318
Start-DscConfiguration 30
Status der Partnerschaft 173
Stop-VM 219

- Suspend-ClusterNode 424
- Test-CauSetup 374
- Test-SRTopology 170
- Unblock-SmbShareAccess 122
- Uninstall-WindowsFeature 459
- Update-VMVersion 235
- Verkehrsklassen 161
- Versionen anzeigen 235
- VMQ aktivieren 286
- Willing-Bit 161
- Windows Defender deinstallieren 459
- Windows Server-Migrationstools 36
- WSUS installieren 446
- Zeichenfolgenarray 200
- CNA (Converged Network Adapter) 160
- CNO (Clusternamenobjekt) 353, 382
- Collector Technology 72
- Commit-Container 318
- Computer umbenennen 23
- Computer, virtuelle 335
 - Dateien 204
 - Einstellungen 203, 206
 - erstellen in Hyper-V-Manager 204
 - erstellen in PowerShell 206
 - exportieren 235, 422
 - Generation 220
 - Generation 2, Beschränkungen 223
 - Generation 2, Vorteile 221
 - Generation festlegen 205
 - herunterfahren 219
 - importieren 235, 422
 - kopieren 422
 - Nanoserver 52
 - NUMA konfigurieren 214

- übertragen 234
- Überwachung 409
- umwandeln 234
- Verwaltung delegieren 194
- ComputerName, Zeichenfolgenarray 200
- Container 298
 - anfügen 317, 318
 - anhalten 316, 318
 - auflisten 316, 318
 - Azure Container Service (ACS) 332
 - Computername 316
 - Datenvolumen verwalten 324
 - Docker 302
 - entfernen 317, 319
 - Hyper-V 313
 - Images 297
 - Portzuordnungen 322
 - Ressourcensteuerung 325
 - starten 316, 318
 - verwalten 315, 319
 - virtualisieren 300
- Containerhosts 297
 - installieren 298
 - Nanoserver 301
 - Server Core 301
- Converged Network Adapter (CNA) 160
- Convert-VHD 397
- Copy-Item 202
- Copy-VMFile 220
- CPU
 - Docker 327
 - Zyklen begrenzen 327
- Credential Security Support Provider (CredSSP) 196
- CredSSP (Credential Security Support Provider) 348

CSV (Cluster Shared Volume) 379

 Failoverclustering 380

 konfigurieren 377

 optimieren 381

 Sicherungen 367

CSVFS 379

D

daemon.json 304

 Adressen, alternative 322

Data Execution Prevention (DEP) 299

Datacenter Bridging (DCB) 160

Datei- und Speicherdienste

 Datendeduplizierung 174

 iSCSI-Zielserver (Rollendienst) 151

Dateien

 .avhd 258

 .avhdx 258

 .cab 83

 .inf 82

 .msu 83

 .vhd 49

 .vhds 397

 .vhdx 49

 .vmc 204

 .vsv 204

 Berechtigungen 124

 Blob erstellen 51

 Clientzugriffe 120

 daemon.json 304

 freigegebene VHDX- 248

 HOSTS 276

 ISO 7

 Konfiguration virtueller Computer 204

- kopieren 202
- MOF (Management Object Format) 29
- Prüfpunkte 258
- Status, gespeicherter 204
- Treiber- 82
- VHD-Satz 397
- Witness.log 359
- Dateifreigabenzeugen 359
- Dateiserver 108, 169
 - Scale-Out File Server (SoFS) 383
 - SMB 407
- Dateisysteme
 - auswählen 104
 - Chkdsk.exe 106
 - CSVFS 379
 - NTFS 104
 - Pseudo- 379
 - ReFS 104
- Datenausführungsverhinderung 299
- Datenbanken, WSUS 443
- Datenbereinigung 178
- Dateneduplizierung *siehe* Deduplizierung
- Datensammlersätze 491
- Datenträger
 - differenzierende 250
 - konfigurieren 91
 - MBR (Master Boot Record) 94
 - Nummer ermitteln 255
 - Partitionstabelle 94
 - Pass-Through- 254
 - Schlupfspeicher 92
 - Speicherebenen 148
 - Zuordnungseinheiten 92
- Datenträger, virtuelle

- dünn bereitgestellt 141
- erstellen 141
- vs. VHD 141
- Datenträgertypen zuordnen 149
- Datenträgerverwaltung 252
 - Diskmgmt.msc 100
 - VHD(X)-Dateien erstellen 99
- Datenträgerzeugen 359
- Datenvolumes
 - hinzufügen 325
 - verwalten 324
 - wiederverwenden 324
- DCB (Datacenter Bridging) 160
- DcbQos 160
- DCBX-Willing 161
- DCOM (Distributed Component Object Model) 27
- Ddpeval.exe 179
- Deduplizierung 174
 - Analysepunkte 177
 - Arbeitsauslastungen 179
 - Blockspeicher 177
 - Datenbereinigung 178
 - Deoptimierung 178
 - Garbage Collection 178
 - Nutzungsszenarien 177
 - Optimierung 178
 - Sicherungen 181
 - Tools 179
 - überwachen 180
- Defender
 - Gruppenrichtlinien 463
 - Modul 461
- Delegierung, eingeschränkte 348
- Deoptimierung 178

- DEP (Data Execution Prevention) 299
- Deployment Image Servicing and Management (DISM.exe) 78
- Desired State Configuration (DSC) 28
- Device Specific Module 163
- DHCP
 - Nanoserver 57
 - virtuelles Netzwerk 271
- Dienstanbieter für Netzwerkvirtualisierung 280
- Dienste
 - Hyper-V-Volumeschattenkopie-Anforderer 482
 - Rollen 13
 - Volumeschattenkopie 260
 - Windefend 459
 - Windows-Remoteverwaltung 24
- Dienstqualität 264
- Dienststeuerungs-Manager, Überwachung 410
- Directory Services Repair Mode (DSRM) 481
- Direkte Speicherplätze 223, 398
 - hyperkonvergiert 404
 - Softwarespeicherbus 400
 - Szenarios 402
 - Windows PowerShell 401
- Disable-PnpDevice 240
- Disable-VMResourceMetering 218
- Disconnect-PSSession 24
- Disconnect-VMNetworkAdapter 239
- Discrete Device Assignment (DDA) 239
- Diskmgmt.msc 100
- DISM (Deployment Image Servicing and Management) 78
 - /image 82
 - Befehlszeile vs. Cmdlet 87
- DISM.exe
 - Hyper-V installieren 192, 306, 370, 404, 421, 425, 450
- Dismount-DiskImage 104

- Dismount-VHD 104, 253
- Dismount-VmHostAssignableDevice 240
- Distributionen
 - FreeBSD 68
 - Linux 68, 227
 - Ubuntu 227
- Djoin.exe 51
- DNS
 - Docker 307
 - Suffixe hinzufügen 355
- DNS-Serveradresse, Nanoserver 61
- Docker 302
 - Arbeitsspeicher begrenzen 326
 - Attach 317
 - Befehle 308
 - Build 310
 - Commit 317, 325
 - CPU-Parameter 327
 - daemon.json 304
 - Daemon-Startoptionen 306
 - DNS-Serveradressen 307
 - Gruppen 307
 - herunterladen 303
 - Image suchen 310
 - Images 311
 - Images entfernen 311
 - Installation auf Nanoserver 303
 - Login 330
 - OneGet 303
 - PATH 305
 - ps 316
 - Pull 308
 - Push 329, 331
 - Ressourcensteuerung 325

- rm 317
- Rmi 311
- Run 311
- Search 310, 330
- Start 316
- Startoptionen 304
- Tag 310
- Top 314
- Windows PowerShell 307
- Docker Daemon *siehe auch* Dockerd
- Dockerd
 - Container umleiten 306
 - Images umleiten 306
 - NAT-Umgebung 307
- Dockerfile 327
- DockerHub 329
- DockerMsftProvider 303
- DockerPS-Dev 307
- Domänen
 - Fehler- 414
 - Offline-Beitritt 50
- DrainOnShutdown 424
- Drei-Wege-Spiegelung 144
- DSC (Desired State Configuration) 28, 54
 - Konfigurationsskript 28
 - Pull-Architektur 29
 - Push-Architektur 30
- DSM (Device Specific Module) 163
 - hinzufügen 164
 - Richtlinien 165
- DSRN (Directory Services Repair Mode) 481
- DVD-Laufwerke 246

E

- Edit-NanoServerImage 55, 86
- Effektiver Zugriff 129
- Eingeschränkte Delegation 348
- Einstellungen
 - Computer, virtuelle 203, 206
 - Windows Defender 459
- Enable-ClusterStorageSpacesDirect 401
- Enable-DedupVolume 177
- Enable-NetAdapterRdma 288
- enable-netadaptervmq 286
- Enable-NetFirewallRule 339
- Enable-NetQosFlowControl 162
- Enable-PSRemoting 198
- enable-vmresourcemeasuring 217
- Engpässe 494
- Enter-PSSession 24, 64, 201
- EPT (Extended Page Tables) 300
- Ereignisanzeige 27
- Ereignisprotokoll 27
- Exit-PSSession 24
- Exportieren
 - Computer, virtuelle 235
- Export-VM 236, 422
- Extended Page Tables (EPT) 300
- F**
- Failover 411
 - Affinität 415
- Failovercluster 248, 351
 - Cloudzeugen 391
 - Prüfbericht 352
 - standortabhängiger 414
- Failoverclustering 54, 351
 - Betriebssystemmodus, gemischter 376

- clusterfähiges Aktualisieren 372
- CSV 380
- Quorum 358
- Speicher, freigegebener 369
- Failoverclustering-Tools 374
- Failovercluster-Manager 351
 - ClusterquorumEinstellungen 360
 - Clusterspeicherplätze 390
- FCoE (Fibre Channel over Ethernet) 159
- Features 13
 - Container 298
 - Failoverclustering 351
 - Hyper-V-Replikate 336
 - Hyper-V-Verwaltungstools 189
 - iSNS 158
 - Nanoserver 54
 - offline installieren 253
 - Windows Server-Migrationstools 36
- Fehlerdomänen 414
- Fehlerpunkt, einzelner 363
- Fehlersuche, SMB 123
- Fehlertoleranz, Stretch-Cluster 413
- Festplatten, virtuelle
 - ändern 252
 - bereitstellen 103
 - erstellen 244
 - erweitern 256
 - Formate 242
 - Größe ändern 256
 - hinzufügen zu virtuellem Computer 246
 - komprimieren 256
 - konvertieren 256
 - trennen 104
 - verkleinern 256

- zusammenführen 256
- Fibre Channel 261
- Filterungsmodus, Affinität 432
- Firewall 196
- Firewallregeln 61
- Flusskontrolle 162
- Force 240
- FreeBSD
 - Bereitstellungen 68
 - Computer, virtuelle 226
 - Distributionen 68
 - Installation 228
 - Integration Services (BIS) 230
- FreeBSD Integration Services (BIS) 68
- FreeBSD Integration Services (FIS) 229
- Freigabeberechtigungen
 - ändern 121
 - Jeder 116
 - konfigurieren 116
 - Prinzipale 117
- Freigaben
 - entfernen 121
 - fortlaufend verfügbare 407
 - Kontingente 116
 - Profile 109
 - SMB 109
 - SoFS 385
 - Verschlüsselung aktivieren 123
- FSRM (File Server Resource Manager) Ressourcen-Manager für Dateiserver 115

G

- Garbage Collection 178
- Gastbetriebssystem 186
 - installieren 243

- Gastcluster
 - Anwendungsmigration 388
 - Knotenüberwachung 388
- Gastclustering 387
- Gäste 185
- Gedankenexperiment
 - Hyper-V 293
 - Netzwerklastenausgleich 438
 - WSUS 506
- Gehäuseresilienz 142
- Generation
 - 2, Vorteile der 221
 - konvertieren 223
 - Parameter 221
- Generic Volume Licensing Key (GVLK) 43
- Geräteinstanzpfad 239
- Geräte-Manager 240
- Get-ClusterNetwork 365, 366, 383
- Get-Command 23
- Get-ComputerInfo, Container 312
- Get-Container 318
- Get-DedupStatus 180
- Get-Disk 255
- Get-help 23
- Get-MpComputerStatus 461
- Get-NetAdapter 21
- GetNetAdapterRdma 288
- Get-NetAdapterVmqQueue 287
- Get-PnpDevice 239
- Get-PnpDeviceProperty 240
- Get-SmbClientConfiguration 123
- Get-SmbOpenFile 120
- Get-SmbSession 119
- Get-SmbShareAccess 121

Get-SRGroup 173
Get-VMCheckpoint 260
Get-VMHostSupportedVersion 235
Get-WindowsFeature 17
Get-WinEvent 173
GLVK (Generic Volume Licensing Key) 43
GPT (GUID-Partitionstabelle) 94
GPT-Datenträger booten 98
Grant-SmbShareAccess 121
Grant-SRAccess 172
Gruppen
 Administratoren 307
 erstellen 453
 Hyper-V-Administratoren 194
 Prinzipal 117
Gruppenrichtlinien, Defender 463
Gruppenrichtlinieneinstellungen
 Clientseitige Zielzuordnung 453
Gruppenrichtlinienobjekte sichern 481
Gruppenrichtlinienverwaltung 481
GUID-Partitionstabelle (GPT) 94

H

Hardware
 Anforderungen, minimale 3
Hardwareadresse 277
Hardwarebeschleunigung 275
Heartbeat-Einstellung 415
Heartbeats 425
Hilfe, PowerShell 200
Hinzufügen, Leistungsindikatoren 488
Histogrammleiste 487
HNS (Host Network Service) 321
HOSTS 276

- Hosts 185, 425
 - Heartbeats 425
 - Knoten 425
 - vertrauenswürdige 197
- Hot-Spare 146
- Hyperkonvergiert 404
- Hyperthreading 4
- Hyper-V 185
 - Container 313
 - Gruppe Administratoren 194
 - Hardwarebeschränkungen 187
 - Hosts remote verwalten 195
 - installieren 190
 - installieren ohne Validierung 192, 306, 370, 404, 421, 425, 450
 - Integrationsdienste 218
 - Livemigration 342
 - Prüfpunkte 258
 - Replikatserver konfigurieren 337
 - Ressourcenmessung 216
 - Rolle 186
 - Sicherung 482
 - SLAT 189
 - Smart Paging 215
 - Speicher 240
 - Speichermigration 420
 - Systeminfo.exe 189
 - verschachteln 192
 - Verwaltungstools 192, 338
 - VMConnect (Virtual Machine Connection) 224
 - Windows-Firewall 339
- Hypervisor 186
 - Partitionen 187
- Hyper-V-Manager 192
 - Manager für virtuelle Switches 271

Remoteverwaltung 195

virtuellen Computer erstellen 204

Hyper-V-Replikat 336

Hyper-V-Replikatbroker 337

Hyper-V-Server 188

Hyper-V-Volumeschattenkopie-Anforderer 482