

# 第 6 章

## 部署和管理 Windows 和 Hyper-V 容器

### ( Deploying and managing Windows and Hyper-V containers )

#### 目录：

单元概述 ( Module Overview )	1
Windows Server 2016 容器概述 ( Overview of containers in Windows Server 2016 )	2
准备容器 ( Preparing for containers )	7
使用 Docker 安装，配置和管理容器 ( Installing, configuring, and managing containers by using Docker )	13
实验: 安装和配置容器 ( Installing and configuring containers )	29
单元复习和作业 ( Module Review and Takeaways )	33

## 单元概述 ( Module Overview )

Windows Server 2016 中的一个重要的新功能是部署容器的选项。通过部署容器，您可以为应用程序提供一个隔离的环境。您可以在单个物理服务器或虚拟服务器上部署多个容器，每个容器为已安装的应用程序提供完整的操作环境。本单元介绍在 Windows Server 2016 中的 Windows 和 Microsoft Hyper-V 容器，它教您如何部署和管理这些容器。

### 目标 ( Objectives )

完成本单元后，您将能够：

- 描述 Windows Server 2016 中的容器功能。
- 说明如何部署容器。
- 说明如何使用 Docker 安装，配置和管理容器。

## 第 1 课

# Windows Server 2016 容器概述 ( Overview of containers in Windows Server 2016 )

完成本课后，学生将能够解释 Windows Server 和 Hyper-V 容器的用途。

## 课程目标 ( Lesson Objectives )

完成本课后，您将能够：

- 描述 Windows Server 容器。
- 描述 Hyper-V 容器。
- 描述使用容器的场景
- 描述容器的安装要求。

## Windows Server 容器概述 ( Overview of Windows Server containers )

容器提供了一个隔离的操作环境，您可以使用它为应用程序提供受控和可移植的空间。容器空间为应用程序运行提供了理想的环境，而不会影响操作系统的其余部分，并且不影响应用程序的操作系统。容器使您能够从操作系统环境中隔离应用程序。

在许多方面，容器是虚拟化的下一个演变。容器也称为基于容器的操作系统虚拟化 ( container-based OS virtualization )。尽管容器在主机 OS 上运行，但是容器彼此隔离。隔离容器 ( Isolated container ) 可提高在容器中运行的应用程序的安全性和可靠性。容器为应用程序提供了一个模拟环境。例如，本地磁盘显示为操作系统文件的新副本，而内存似乎仅保存最近启动的操作系统的数据，并且唯一正在运行的组件是操作系统。

Windows Server 2016 支持两种不同类型的容器或运行时 ( runtimes )，每种容器提供不同程度的隔离和不同的要求：

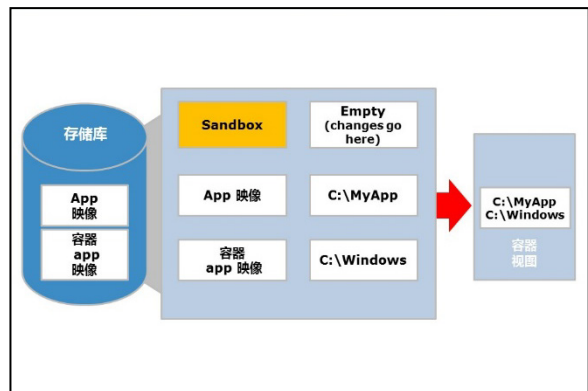
- Windows Server 容器。这些容器通过进程和命名空间隔离技术提供应用程序隔离。Windows Server 容器与容器主机以及在主机上运行的所有其他容器共享操作系统内核。虽然这提供了更快的启动体验，但它不能提供容器的完全隔离。
- Hyper-V 容器。这些容器通过在高度优化的虚拟机 ( VM ) 中运行每个容器来扩展 Windows Server 容器提供的隔离。但是，在此配置中，容器主机的操作系统内核不与 Hyper-V 容器共享。

容器看起来像是应用程序的完整操作系统。因此，在许多方面，容器与 VM 类似，因为它们运行操作系统，它们支持文件系统，您可以像任何其他物理机器或 VM 一样通过网络访问它们。然而，容器背后的技术和概念与 VM 非常不同。

## 容器定义 ( Container definitions )

当您在 Windows Server 2016 中开始创建和使用容器时，了解构成容器体系结构的关键概念很有帮助：

- 容器主机 ( Container host )。此元素由使用 Windows 容器功能配置的物理计算机或虚拟计算机组成。容器主机可以运行一个或多个 Windows 容器。



- 容器映像 ( Container image )。当对容器文件系统或注册表进行修改时, 这些更改将捕获在容器的沙箱中。在许多情况下, 您可能需要捕获容器映像状态, 以便您创建的新容器可以继承容器更改。停止容器后, 您可以丢弃沙箱, 也可以将其转换为新的容器映像。例如, 您可以将应用程序安装到容器中, 然后捕获安装后状态。从此状态, 您可以创建包含应用程序的新容器映像。该映像将仅包含应用程序安装所做的更改, 并在容器操作系统映像的顶部显示一个图层。
- 容器操作系统映像 ( Container OS image )。虽然容器是从映像制作的, 但容器操作系统映像是构成容器的潜在多个映像层中的第一层。容器操作系统映像提供操作系统环境, 它是不可变的。
- 沙箱 ( Sandbox )。此层包括对容器所做的所有更改, 包括文件系统修改, 注册表修改或软件安装。您可以根据需要保留或丢弃这些更改。
- 容器存储库 ( Container repository )。每次创建容器映像时, 容器映像及其依赖关系都存储在本地存储库中。这允许您在容器主机上多次重复使用该映像。

最后, 重要的是要了解您可以使用 Windows PowerShell 命令行界面或使用开源 Docker 平台来管理容器。

## Windows Server 容器 ( Windows Server containers )

部署物理或虚拟计算机时, 计算机必须具有在单个内核模式之上运行的单用户模式。计算机提供了允许多个用户模式的边界, 因此您可以部署多个隔离的应用程序。例如, Hyper-V 提供子分区或 VM, 每个子分区可以具有自己的 Windows Server 操作系统以及必需的内核和用户模式, 并且每个应用程序安装在每个用户模式或每个 VM 中。容器允许每个内核模式有多个用户模式, 每个内核模式只需要一个计算机。

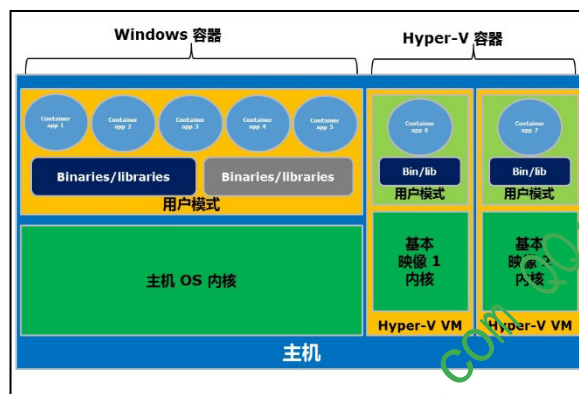
如前所述, 计算机使用具有内核模式和用户模式的 Windows Server OS 部署。OS 的用户模式管理容器主机或托管容器的计算机。Windows OS 的特殊删除版本 ( 用作容器操作系统映像存储在容器存储库中 ) 用于创建容器。此容器仅具有用户模式 - 这是 Hyper-V 和容器之间的区别, 因为 VM 使用用户模式和内核模式运行客户机操作系统。Windows Server 容器的用户模式允许 Windows 进程和应用程序进程在容器中运行, 与其他容器的用户模式隔离。当您虚拟化操作系统的用户模式时, Windows Server 容器允许多个应用程序在同一计算机上以孤立状态运行, 但它们不提供安全增强隔离。

## Hyper-V 容器概述 ( Overview of Hyper-V containers )

讨论 VM 可以帮助您深入了解 Hyper-V 容器。VM 还提供用于运行应用和服务的隔离环境。但是, VM 为内核和用户模式提供了一个完整的客户操作系统。例如, 启用 Hyper-V 角色的计算机包括父分区或管理操作系统, 隔离的内核和用户模式, 并且它负责管理主机。每个子分区或托管 VM 都运行具有内核模式和用户模式的操作系统。

与 VM 类似, Hyper-V 容器是部署的子分区。另一方面, Hyper-V 容器中的客户机操作系统不是我们所知道的正常的, 完整的 Windows 操作系统, 它是 Windows Server 操作系统的优化, 精简版本 - 这不同于 Nano 服务器。Hyper-V 子分区提供的边界为 Hyper-V 容器, 主机上的其他 Hyper-V 容器, 虚拟机管理程序和主机的父分区之间提供安全性增强的隔离。

Hyper-V 容器使用为应用程序定义的基本容器映像, 并且它们通过使用该基本映像自动创建 Hyper-V VM。部署后, Hyper-V 容器以秒为单位启动, 这比具有完整 Windows 操作系统的 VM 快得多, 甚至比 Nano 服务器更快。Hyper-V 容器具有隔离的内核模式, 用于核心系统进程的用户模式和容器用户模式, 这和 Windows Server 容器用户模式是一样的。事实上, Hyper-V 容器使用 VM 内的 Windows 容器来存储二进制文件, 库和应用程序。



现在，Windows 容器在 Hyper-V VM 中运行，这为应用程序提供了内核隔离和主机修补程序和版本级别的分离。由于应用程序使用 Windows 容器进行容器化，因此您可以通过选择 Windows 或 Hyper-V 容器来选择部署期间所需的隔离级别。使用多个 Hyper-V 容器，您可以使用不需要手动管理虚拟机的公共基本映像；VM 将自动创建和删除。

## 使用场景 ( Usage scenarios )

Windows Server 和 Hyper-V 容器在企业环境有以下  
的常见应用场景。

### Windows Server 容器 ( Windows Server containers )

尽管 Windows Server 容器和 Hyper-V 容器之间存在许多相似之处，但是这些虚拟化技术的差异使得它们有各自的适用场景。例如，在操作系统信任其托管的应用程序，并且所有应用程序必须彼此信任的情况下，Windows Server 容器是首选。换句话说，主机操作系统和应用程序在同一信任边界内。对于许多的多容器 ( multiple-container ) 应用程序，构成较大应用程序的共享服务的应用程序，以及来自同一组织的应用程序，这是真的。

- Windows 容器的一些常见使用方案包括：
  - Windows Server 容器用于：
    - 托管无状态应用程序
    - 快速测试部署
  - Hyper-V 容器用于：
    - 多租户
    - 单租户
    - 独立生命周期管理

您应该确保在 Windows Server 2016 主机上的容器中部署的应用程序是无状态的。此类型的应用程序不会在其容器中存储任何状态数据。此外，请记住，容器没有 GUI。基于容器的特性，您可能不会在容器中运行财务软件包。另一方面，像游戏和网站这样的应用程序会在本地系统而不是服务器上渲染，因此它们非常适用于容器的应用程序。总而言之，没有 GUI 的无状态 Web 应用程序和类似的代码是在 Windows Server 2016 中使用 Windows 容器技术的最可能的候选者。

### 针对快速测试环境部署的 Windows Server 容器 ( Windows Server containers for rapid test deployment )

容器可用于快速打包和交付分布式应用程序。自定义应用可能需要多次部署，每周或有时每天部署，以跟上更改。

Windows Server 容器是部署这些应用程序的理想方式，因为您可以通过使用分层方法来构建可部署的应用程序来创建包。例如，您可以创建托管包含已安装的 Internet 信息服务 ( IIS ) 和 Microsoft ASP.NET 软件的网站的映像。然后，开发人员可以多次使用该映像来部署应用程序，而不更改底层。由于 Windows Server 容器在启动时提供更高的效率，更快的运行时性能以及比 Hyper-V 容器更高的密度，开发人员可以花更多的时间开发应用程序，同时需要更少的资源。



**注意：**虽然不是 Windows Server 容器所特有的，但您可以将测试环境中的相同软件包部署到您的生产环境中 - 它以与开发人员和测试人员相同的方式运行。作为一个额外的好处，您还可以在 Microsoft Azure 中部署此容器，而不更改它。

### Hyper-V 容器 ( Hyper-V containers )

Hyper-V 容器每个都有自己的 Windows 操作系统内核副本，并有直接分配给它们的内存，这是强隔离的关键要求。与 VM 类似，在需要中央处理单元 ( CPU )，内存和 I/O 隔离的场景 ( 例如网络和存储 ) 中，您将使用 Hyper-V 容器。主机操作系统仅向容器公开了一个小的受限接口，用于主机资源的通信和共享。这种非常有限的共享意味着 Hyper-V 容器在启动时间和密度方面的效率比 Windows Server 容器的效率要低一些，但它们提供了允许不可信应用程序在同一主机上运行所需的隔离。



Hyper-V 容器中的信任边界为主机上的 Hyper-V 容器, 虚拟机管理程序和主机的其他进程之间提供安全增强的隔离。因此, Hyper-V 容器是多租户环境中的首选虚拟化模型。

### 用于多租户的 Hyper-V 容器 (Hyper-V containers for multiple tenants)

在某些情况下, 您可能需要在同一主机上运行需要不同信任边界的应用程序。例如, 您可能正在部署多租户平台即服务 (PaaS) 或 SaaS 产品, 您允许客户提供自己的代码来扩展服务产品的功能。但是, 您需要确保一个客户的代码不会干扰您的服务或访问您的其他客户的数据。Hyper-V 容器为租户提供所需的组件, 但它们还确保一个租户的应用程序不会干扰其他应用程序。

在承载 Hyper-V 容器的云服务提供商的典型使用场景中, 服务提供商将拥有一组运行 Windows Server 2016 的 Hyper-V 主机, 用于其部分云。此 Hyper-V 主机群集将托管一组虚拟机, 每个虚拟机将安装 Windows Server 2016 作为其客户机操作系统。当您使用 Windows 容器技术时, 每个 VM 都将承担容器主机的角色。然后, 每个容器主机或 VM 将被分配给不同的租户, 然后租户可以在其专用容器主机上创建与其所需的一样多的容器。如果恶意软件或恶意攻击攻击了一个容器主机或虚拟机, 属于其他客户的其他虚拟机将不受影响。

### 用于单一租户的 Hyper-V 容器 (Hyper-V containers for single tenants)

Hyper-V 容器即使在单租户环境中也是有用的。一个常见的场景是, 您要在 Windows 容器中托管的一个或多个应用程序对操作系统版本级别或底层容器主机的修补程序级别具有依赖性。在这种情况下, 您可以考虑配置单个 Hyper-V 主机或主机群集, 并使用 Hyper-V 容器, 而不是将多个系统配置为容器主机, 并使用 Windows Server 容器。

### 针对独立生命周期管理的 Hyper-V 容器 (Hyper-V containers for independent lifecycle management)

隔离非常有用的另一种情况是, 如果要运行具有不同版本的 Windows Server 的容器。Windows Server 容器的一个挑战是它们在基本映像和容器映像之间共享大部分操作系统。因此, 如果升级了基本映像中的操作系统, 则还需要升级容器。

或者, Hyper-V 容器允许您具有不同版本的基本映像, 从而允许您在容器映像中同时托管操作系统。此功能对于希望针对修补, 更新和合规性原因进行独立生命周期管理的企业很有帮助。

## 安装要求 (Installation requirements)

在规划 Windows 容器时, 您应该了解 Windows Server 2016 的要求。您还应该熟悉 Windows Server 2016 中 Windows Server 容器和 Hyper-V 容器支持的场景。

### Windows 容器主机要求 (Windows container host requirements)

在规划部署时, Windows 容器主机具有以下要求:

- Windows 容器角色仅在以下位置可用:
  - Windows Server 2016 (完全或服务器核心)。
  - Nano 服务器。
  - Windows 10 (周年更新)。
- 如果部署了 Hyper-V 容器, 则需要安装 Hyper-V 角色。
- Windows Server 容器主机必须将 Windows 操作系统安装到 C: 上。如果仅部署 Hyper-V 容器, 则此限制不适用。

- 在规划 Windows 容器时, 应考虑以下事项:
  - Windows 容器主机要求
  - 虚拟化容器主机要求
  - 支持的场景

主机操作系统	Windows Server 容器	Hyper-V 容器
Windows Server 2016 完整 UI	Server Core 映像	Nano 服务器映像
Windows Server 2016 核心	Server Core 映像	Nano 服务器映像
Windows Server 2016 Nano 服务器	Nano Server 映像	Nano 服务器映像
Windows 10	不可用	Nano 服务器映像

**虚拟化容器主机要求 ( Virtualized container host requirements )**

如果在承载 Hyper-V 容器的 Hyper-V 虚拟机上部署 Windows 容器主机，则需要启用嵌套虚拟化。嵌套虚拟化具有以下要求：

- 虚拟化 Hyper-V 主机至少需要 4 GB 内存
- 在主机系统上：
  - Windows Server 2016
  - Windows 10 (周年更新)
- 在容器主机虚拟机上：
  - Windows Server 2016 (完全或服务核心)
  - Nano Server
  - Windows 10 (周年更新)
- 具有 Intel VT-x 和 EPT 技术的处理器（此功能目前仅适用于 Intel 处理器）。
- 配置版本为 8.0 或更高版本的 Hyper-V 虚拟机
- 容器主机虚拟机至少有两个虚拟处理器

**支持场景 ( Supported scenarios )**

Windows Server 2016 提供了两个容器操作系统映像：Windows Server Core 和 Nano Server。然而，并非所有配置都支持两个操作系统映像，下表列出了支持的场景。

主机操作系统	Windows Server 容器	Hyper-V 容器
Windows Server 2016 Full UI	Server Core 映像	Nano Server 映像
Windows Server 2016 Core	Server Core 映像	Nano Server 映像
Windows Server 2016 Nano Server	Nano Server 映像	Nano Server 映像
Windows 10 Insider releases	不适用	Nano Server 映像

**知识点检查 ( Check Your Knowledge )**

问题	
在 Windows Server 2016 容器中，以下哪些语句是对沙箱最精确的描述？	
选择正确的答案：	
<input type="checkbox"/>	沙箱是配置有容器的计算机。这可以是物理计算机或虚拟计算机。
<input type="checkbox"/>	沙箱是容器层次结构的第一层。
<input type="checkbox"/>	对运行的容器所做的所有更改都存储在沙箱中
<input type="checkbox"/>	沙箱是一种管理工具，您可以使用它来代替 Windows PowerShell 命令行界面来管理容器。

## 第 2 课 准备容器 (Preparing for containers)

容器为应用程序提供了一个隔离和可移植的操作环境。从应用程序的角度来看，容器显示为具有自己的文件系统，设备和配置的隔离的 Windows 操作系统。Windows Server 支持两种类型的容器：Windows Server 容器和 Hyper-V 容器。Windows Server 容器通过命名空间 (namespace) 和进程隔离实现隔离，而 Hyper-V 容器将每个容器封装在轻量级 VM 中。为了支持您组织的应用程序需求，您应该了解如何启用和配置 Windows Server 以支持容器的基础知识。

### 课程目标 (Lesson Objectives)

完成本课后，您将能够：

- 解释如何准备 Windows Server 容器。
- 解释如何准备 Hyper-V 容器。
- 解释如何部署程序包提供程序。

### 准备 Windows Server 容器 (Preparing Windows Server containers)

在 Windows Server 2016 中使用容器之前，您需要部署容器主机。您可以选择在物理主机计算机上或 VM 内部署容器。您也可以选择使用 Windows Server 2016，带或不带桌面体验 (Desktop Experience)，或者 Nano 服务器。部署容器主机的过程因容器的类型而异。

#### 准备 Nano 服务器 (Preparing a Nano Server)

如果您选择在 Nano 服务器上部署 Windows Server 容器，请使用下表中的高级步骤为 Windows Server 容器准备服务器。

- 准备 Nano 服务器：
    1. 为容器创建 Nano Server 虚拟硬盘 (VHD) 文件
    2. 继续以下步骤
  - 使用以下步骤为容器准备 Windows Server 主机：
    1. 安装容器功能\*
    2. 创建虚拟交换机
    3. 配置 NAT 设置
    4. 配置 MAC 欺骗
- \* 如果你部署到 Nano 服务器，则不需要此步骤

部署动作	细节
为容器创建 Nano 服务器虚拟硬盘文件	<p>准备具有容器和 Hyper-V 功能的 Nano 服务器虚拟硬盘文件。这需要您使用 -Compute 和 -Containers 开关构建一个 Nano Server 映像。例如，您可以键入以下代码，然后按 Enter 键。</p> <pre>New-NanoServerImage -MediaPath \$WindowsMedia -BasePath C:\nano -TargetPath C:\nano\NanoContainer.vhdx -GuestDrivers -ReverseForwarders -Compute -Containers</pre>

部署 Nano 服务器后，您必须完成以下部分中列出的步骤。

#### 准备 Windows Server 主机 (Preparing the Windows Server host)

使用下表中的步骤为容器的 Windows Server 主机做准备。

部署动作	细节
安装容器功能*	<p>此步骤允许使用 Windows Server 和 Hyper-V 容器。您可以使用服务器管理器安装此功能，也可以键入以下 Windows PowerShell cmdlet，然后按 Enter 键。</p> <pre>Install-WindowsFeature Containers</pre>
创建虚拟交换机	<p>所有容器连接到虚拟交换机用于网络通信。交换机类型可以是 Private, Internal, External, 或 NAT。键入以下 Windows PowerShell cmdlet 以完成此任务，然后按 Enter 键。</p> <pre>New-VMSwitch -Name &lt;Virtual Switch Name&gt; -SwitchType &lt;Type&gt;</pre> <p>如果类型是 NAT，则还必须使用 -NATSubnetAddress 172.16.0.0/12 切换，用适当的子网地址替换 172.16.0.0/12</p>
配置网络地址转换 (network address translation , NAT)	<p>如果要使用配置了 NAT 的虚拟交换机，则必须配置 NAT 设置。例如，您可以键入以下 Windows PowerShell 命令，然后按 Enter 键</p> <pre>New-NetNat -Name ContainerNat -InternalIPInterfaceAddressPrefix "172.16.0.0/12"</pre> <p>使用适合您的网络的地址替换子网地址</p>
配置介质访问控制 ( media access control , MAC ) 地址欺骗	<p>如果容器主机被虚拟化，则必须启用 MAC 地址欺骗。例如，您可以键入以下 Windows PowerShell 命令，然后按 Enter 键。</p> <pre>Get-VMNetworkAdapter -VMName &lt;Container Host VM&gt;   Set-VMNetworkAdapter -MacAddressSpoofing On</pre>

\* 如果您选择将 Windows Server 容器部署到 Nano 服务器，则不需要此步骤。

## 准备 Hyper-V 容器 ( Preparing Hyper-V containers )

在 Windows Server 2016 中使用容器之前，您需要部署容器主机。您可以选择在物理主机计算机上或 VM 内部署容器。您也可以选择使用 Windows Server 2016，带或不带桌面体验，或者 Nano 服务器。部署容器主机的过程因容器的类型而异。

### 准备 Nano Server ( Preparing a Nano Server )

如果选择将 Hyper-V 容器部署到 Nano 服务器，请使用下表中的高级步骤为 Hyper-V 容器准备服务器。

- 准备 Nano 服务器：
  - 为容器创建 Nano Server 虚拟硬盘 ( VHD ) 文件
  - 继续执行以下步骤
- 使用以下步骤为容器准备 Windows Server 主机：
  1. 安装容器功能\*
  2. 启用 Hyper-V 角色\*
  3. 启用嵌套虚拟化
  4. 配置虚拟处理器
  5. 创建虚拟交换机
  6. 配置 NAT 设置
  7. 配置 MAC 地址欺骗
- \*如果部署到 Nano 服务器，则不需要步骤



部署动作	细节
为容器创建 Nano 服务器虚拟硬盘文件	<p>准备具有容器和 Hyper-V 功能的 Nano 服务器虚拟硬盘文件。这需要您使用-Compute 和-Containers 开关构建一个 Nano Server 映像。例如，键入以下命令，然后按 Enter 键。</p> <pre>New-NanoServerImage -MediaPath \$WindowsMedia -BasePath c:\nano -TargetPath C:\nano\NanoContainer.vhdx -GuestDrivers -ReverseForwarders -Compute -Containers</pre>

然后，您必须完成下一节中列出的步骤。

## 准备 Windows Server 主机 (Preparing the Windows Server host)

使用下表中的步骤为容器的 Windows Server 主机做准备。

部署动作	细节
安装容器功能*	您可以使用服务器管理器或使用 Install- WindowsFeature Containers Windows PowerShell cmdlet 安装此功能。然后，您可以启用使用 Windows Server 和 Hyper-V 容器。
启用 Hyper-V 角色*	您可以使用服务器管理器或使用 Install- WindowsFeature Hyper-V Windows PowerShell cmdlet 安装此功能。仅当部署 Hyper-V 容器时，这是必需的。
启用嵌套虚拟化 (nested virtualization)	<p>如果容器主机是 Hyper-V 虚拟机，则必须启用嵌套虚拟化。键入以下 Windows PowerShell 命令，然后按 Enter 键</p> <pre>Set-VMProcessor -&lt;VMName Container Host VM&gt; -ExposeVirtualizationExtensions \$true</pre>
配置虚拟处理器	<p>如果容器主机是 Hyper-V 虚拟机，则必须至少配置两个虚拟处理器。您可以键入以下 Windows PowerShell 命令，然后按 Enter 键。</p> <pre>Set-VMProcessor -&lt;VMName Container Host VM&gt; -Count 2</pre>
创建虚拟交换机	<p>所有容器连接到虚拟交换机用于网络通信。交换机类型可以是 Private, Internal, External, 或者 NAT。 键入以下 Windows PowerShell cmdlet 以完成此任务，然后按 Enter 键。</p> <pre>New-VMSwitch -Name &lt;Virtual Switch Name&gt; -SwitchType &lt;Type&gt;</pre> <p>如果类型是 NAT，则还必须使用 -NATSubnetAddress 172.16.0.0/12 切换，用适当的子网地址替换 172.16.0.0/12</p>
配置 NAT	如果要使用配置了 NAT 的虚拟交换机，则必须配置

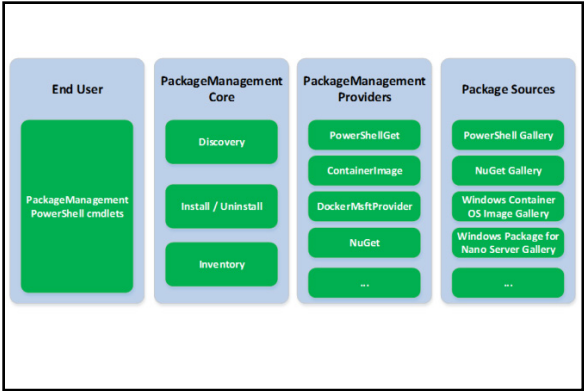
部署动作	细节
	<div>NAT 设置。例如，您可以键入以下 Windows PowerShell 命令，然后按 Enter 键</div> <div>New-NetNat -Name ContainerNat -InternalIPInterfaceAddressPrefix "172.16.0.0/12"</div> <div>使用适合您的网络的地址替换子网地址</div>
配置 MAC 欺骗	<div>如果容器主机被虚拟化，则必须启用 MAC 地址欺骗。例如，您可以键入以下 Windows PowerShell 命令，然后按 Enter 键。</div> <div>Get-VMNetworkAdapter -VMName &lt;Container Host VM&gt;   Set-VMNetworkAdapter -MacAddressSpoofing On</div>

\* 如果您选择将 Hyper-V 容器部署到 Nano 服务器，则不需要这些步骤。

部署包提供程序 ( Deploying package providers )

部署容器时，您首先需要有一个基本映像 ( base image )，例如 Windows Server Core 或 Nano Server 映像。由于 Windows Server 2016 中不包含基本映像，因此您需要使用包提供程序来检索和管理容器部署的基本映像。

除了 PowerShellGet，所有包提供程序都是按需安装的。安装和导入提供程序后，您可以搜索，安装和清点该提供程序的软件包 - 此过程通常称为软件发现/安装/清点或 SDII。每个提供程序都包括特定于程序包类型的 Windows PowerShell cmdlet。这样，您可以根据您的经验和用法在各种包提供商之间进行选择。



虽然其他软件包提供程序一直在开发，下表详细介绍了 Windows Server 2016 中可用的常用提供程序。您还可以使用 Windows PowerShell cmdlet Find-PackageProvider 列出可用于安装的所有软件包提供程序。

名称	概述
PowerShellGet	从在线存储库安装 Windows PowerShell 模块和脚本的软件包提供程序
ContainerImage	用于发现，下载和安装 Windows 容器操作系统映像的软件包提供程序
DockerMsftProvider	用于发现，安装和更新 Docker 映像的软件包提供程序
Nugent	C # 软件包的软件包提供程序
NanoServerPackage	用于发现，下载和安装 Nano Server 软件包的软件包提供程序
WSAProvider	用于发现，安装和清点 Windows Server App ( WSA ) 软件包的软件包提供程序 - APPX cmdlet 的包装程序
MyAlbum	用于在远程文件存储库中发现照片并将它们安装到本地文件夹的软件包提供程序

名称	概述
	供程序

虽然包提供程序可以引用各种源或者存储库, 最常见的包括 PSGallery ( <http://www.PowerShellGallery.com> ), <https://OneGet.org> ( 重定向到 <https://github.com/oneget/> ), <https://www.NuGet.org> 和本地文件夹或网络共享。

### PackageManagement (OneGet)


PackageManagement, 以前称为 OneGet, 是一个包管理聚合器。虽然许多人通常认为 PackageManagement 是一个软件包提供程序, 但该包管理器是一个统一的包管理接口组件, 包含一组托管和本机 API, 一组 Windows PowerShell cmdlet 和 WMI 提供程序。组件接受 Microsoft 提供的和第三方提供的插件, 以扩展特定包类型的功能。

PackageManagement 是 SDII 的统一接口。在核心, 它管理插件提供程序的加载和复用, 并为其中的每个提供隔离。PackageManagement 还公开了软件包管理提供程序的 API 来访问公共功能, 并定义了宿主应用程序在使用库时应该实现的一组 API。而不是要求您了解每个包提供程序的复杂性和多样性, PackageManagement 允许您以常见的方式使用包。

虽然它包括一些管理组件 (如 Windows PowerShell cmdlet), 但是 PackageManagement 会将所有操作委派给已安装的软件包提供程序插件。因此, 它不安装软件 (包括 Windows PowerShell 模块), 并且不包括存储库。许多人错误地认为 PackageManagement 是一个包提供程序的原因是因为它包括一组最小的包提供程序集 - 基本上是发现和安装更多提供程序所需的最低限度。这个最小的包提供程序包括:

- MSI 提供程序。此提供程序实现最低功能以安装 MSI 文件。
- MSU 提供程序。此提供程序实现安装 Microsoft Update (MSU) 文件的最低功能。
- 程序提供程序 (programs provider)。此提供程序不会安装程序, 而只会公开 Windows 中 Add/Remove Programs 应用程序的清单。
- 引导提供程序 (bootstrap provider)。此自定义提供程序从 Internet 上的 Feed 中动态下载和安装新的程序包提供程序, 无需手动下载。
- PowerShellGet 提供程序。此提供程序安装 Windows PowerShell 模块。

事实上, 在上表中列出的所有软件包提供程序都使用 PackageManagement API。因此, 您可能在线存储库中查看引用这些软件包提供程序作为 PackageManagement 提供程序的文档。

 **注意:** 虽然 PowerShellGet 包括在 PackageManagement 中。PowerShellGet 可以通过标准的 PackageManagement cmdlet 集及其自己的 cmdlet 访问。例如, 您可以访问 cmdlet Find-Module 和 Install-Module。

### 安装软件包提供程序 (Installing package providers)

找到适当的软件包提供程序后, 需要将其安装到计算机。您可以使用 Windows PowerShell cmdlet Install-PackageProvider 来安装在使用 PowerShellGet 注册的包源中可用的 PackageManagement 提供程序 - 这包括 Windows PowerShell 库中提供的模块。如果程序包提供程序已安装, 您还可以使用 Windows PowerShell cmdlet Import-PackageProvider 将提供程序导入到当前 Windows PowerShell 会话。例如, 您可以使用以下命令安装 ContainerImage 提供程序。

```
Install-PackageProvider -Name ContainerImage -Force
```



**注意：**即使计算机上安装了其他版本的提供程序，`Force` 参数也会安装软件包提供程序。此 cmdlet 还包括以下参数，用于管理要安装的提供程序的版本：`MaximumVersion`，`MinimumVersion`，`AllVersions` 和 `RequiredVersion`。

安装后，程序包提供程序将启用其他 Windows PowerShell cmdlet 或 API。例如，在安装 `ContainerImage` 提供程序后，可以使用以下 Windows PowerShell cmdlet 来管理容器映像：`Find-ContainerImage`，`Save-ContainerImage`，`Install-ContainerImage`。

大多数包提供者要求计算机能够访问因特网。但是，一些组织不提供对其服务器的 Internet 访问。在此方案中，您可以使用 Windows PowerShell cmdlet，如 `Save-ContainerImage` 来下载和保存容器映像，而无需安装。然后，可以将映像移动到容器主机以进行安装。



**注意：**这个单元会包括如何使用 Docker 来管理 Windows 容器。

## 第 3 课

# 使用 Docker 安装, 配置和管理容器 ( Installing, configuring, and managing containers by using Docker )

### 课程目标 ( Lesson Objectives )

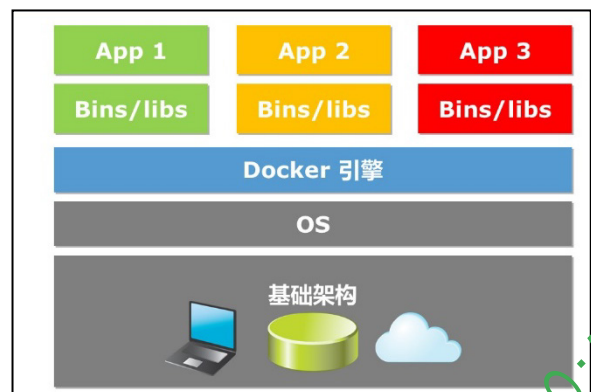
完成本课后, 您将能够:

- 定义 Docker。
- 描述对 Windows Server 2016 上的 Docker 的支持。
- 描述 Docker 的使用场景。
- 解释如何安装和配置 Docker。
- 描述 Docker 的管理。
- 描述 Docker Hub。
- 描述 Docker 如何与 Azure 集成。
- 使用 Docker 部署 Hyper-V 容器。

### 什么是 Docker? ( What is Docker? )

Docker 是一整套开源工具, 解决方案和基于云的服务的集合, 为应用程序代码打包或将应用程序代码集成到一个用于软件开发的标准化单元中提供了一个通用模型。这个标准化单元也称为 *Docker 容器*, 它是一个完整的文件系统, 它包含了运行所需的所有东西: 代码, 运行时, 系统工具和系统库, 或者可以在服务器上安装的任何东西。

支持 Docker 容器是 Docker 平台或 *Docker Engine* 的核心。该主机守护程序是一个轻量级的运行时环境, 与 Docker 客户端进行通信, 以运行命令来构建, 运送和运行 Docker 容器。这保证了应用程序始终运行相同的, 不管它运行的环境如何。



Docker 容器与 VM 具有相似的资源隔离和分配优势, 但是它们使用不同的架构方法, 使它们更加便携和高效。这些容器包括应用程序及其所有依赖项, 但与其他容器共享内核。每个 Docker 容器作为主机操作系统上的用户空间中的隔离进程运行。

Docker 容器基于开放标准, 允许容器在所有主要 Linux 发行版和 Microsoft 操作系统上运行, 并支持每个基础架构。因为它们没有绑定到任何特定的基础设施, Docker 容器可以在任何计算机, 任何基础设施和任何云中运行。

为了保证包装格式保持通用, Docker 最近组织了开放容器倡议 ( Open Container Initiative ), 旨在确保容器包装仍然是微软作为创始成员之一的开放行业标准。 Microsoft 正在与 Docker 协作, 目的是使开发人员能够通过使用相同的 Docker 工具集创建, 管理和部署 Windows Server 和 Linux 容器。针对 Windows Server 的开发人员将不再需要在使用广泛的 Windows Server 技术和构建容器化应用程序之间进行选择。通过对 Docker 项目做出贡献, Microsoft 支持使用 Docker 工具集管理 Windows Server 容器和 Hyper-V 容器。



## Windows Server 2016 对 Docker 的支持 ( Docker support in Windows Server 2016 )

在此之前，不可能使用 Windows Server 平台托管 Docker 引擎 ( Docker Engine ) 而不添加额外的虚拟化层。随着 Windows Server 2016 的发布，Windows Server 主机提供了一个内置的本机 Docker 守护程序。在 Windows Server 2016 中使用此组件，您可以在生产 Windows 环境中使用 Docker 容器，工具和工作流。

用于 Windows Server 的 Docker 引擎需要 Windows Server 2016，并且包括以下要点：

- 无跨平台容器化 ( No cross-platform containerization )。目前没有方法从另一个平台向一个容器提供相应的内核。换句话说，Windows 容器需要一个 Windows Docker 主机，而 Linux 容器需要一个 Linux Docker 主机。
- 在 Windows 操作系统中管理容器的两种方法。您可以使用 Docker 工具集或 Windows PowerShell 创建和管理 Windows 容器。但是，使用 Windows PowerShell 创建的容器无法使用 Docker 进行管理，并且使用 Docker 工具集创建的容器无法使用 Windows PowerShell 进行管理。

- 用于 Windows Server 的 Docker 引擎需要 Windows Server 2016，并且包括以下要点：
  - 无跨平台容器化
  - 在 Windows 操作系统中管理容器的两种方法

## Docker 组件 ( Docker components )

了解 Docker 的工作原理和一些基本的 Docker 术语非常重要；这些术语从早期重新定义以提供特定于 Docker 的清晰度：

- **映像 ( Image )**。以层级文件系统的形式存在的根文件系统更改的无状态集合，它们彼此堆叠。
- **容器 ( Container )**。映像的运行时实例，由映像，其操作环境和一组标准指令组成。
- **Dockerfile**。一个文本文件，包含需要运行以构建 Docker 映像的命令。
- **构建 ( Build )**。从 Dockerfile 和正在构建映像的目录中的任何其他文件构建 Docker 映像的过程。

Docker 术语：

- Image, container, Dockerfile, Build

Docker 工具箱：

- Docker Engine, Docker Compose, Docker machine, Docker client, Kitematic, Docker Registry, Docker Swarm

Docker 解决方案：

- Docker Hub, Docker Trusted Registry, Universal Control Panel, Docker Cloud, Docker Datacenter

## Docker 工具箱 ( Docker toolbox )

Docker 工具箱是 Docker 平台工具的集合，它们使构建，测试，部署和运行 Docker 容器成为可能。这些工具包括：

- **Docker 引擎 ( Docker Engine )**。这是一个用于构建和运行 Docker 容器的轻量级运行时环境。Docker 引擎包括一个内部主机守护进程 ( Linux 服务 )，您可以通过使用 Docker 客户端来构建，部署和运行容器来进行通信。
- **Docker 撰写 ( Docker Compose )**。这使您能够定义多容器应用程序与任何依赖关系，以便您可以使用单个命令运行它。Docker Compose 允许您指定应用程序将使用任何必需的卷或网络使用的映像。
- **Docker 机器 ( Docker Machine )**。这使您能够通过向数据中心或在提供商的计算机上安装 Docker 引擎来部署 Docker 主机。Docker Machine 还安装和配置 Docker 客户端，以便它可以与 Docker 引擎通信。

- Docker 客户端 ( Docker client ) 。这包括一个预配置为 Docker 命令行环境的命令 shell。
- Kitematic。此 GUI 可以帮助您快速构建和运行 Docker 容器, 并从 Docker Hub 中查找和提取映像。
- Docker 存储库 ( Docker Registry ) 。这个开源应用程序形成了 Docker Hub 和 Docker Trusted Registry 的基础。
- Docker 群集 ( Docker Swarm ) 。这种本地群集功能允许您将多个 Docker 引擎组合到一个虚拟 Docker 引擎中。

您可以在各种平台上下载和安装 Docker 软件, 包括 Windows, Linux 和 Mac OS X。在计算机上安装 Docker 软件后, 您可以继续构建映像和标记, 并将其推送或拖动到 Docker Hub。

## Docker 解决方案 ( Docker solutions )

Docker 工具箱中的软件并不都是 Docker 平台所提供的。以下 Docker 解决方案也是使 Docker 对 DevOps 如此强大的关键部分:

- Docker Hub。这是一个云托管服务, 您可以注册您的 Docker 映像并与其他人共享。
- Docker 可信存储库 ( Docker Trusted Registry ) 。这个专用的专用映像库允许您存储和管理本地或虚拟私有云中的映像。
- 通用控制面板 ( Universal Control Panel ) 。您可以使用它来管理 Docker 应用程序, 无论它们是在本地运行还是在虚拟私有云中运行。
- Docker 云 ( Docker Cloud ) 。通过这种云托管服务, 您可以直接部署和管理 Docker 应用程序。
- Docker 数据中心 ( Docker Datacenter ) 。作为稳定的 Docker 解决方案的最新成员, DDC 是一个集成的端到端平台, 用于在内部部署或在虚拟私有云中部署容器。DDC 的自助服务功能使开发人员能够轻松构建, 测试, 部署和管理敏捷应用程序。

## 使用场景 ( Usage scenarios )

随着组织采用容器, 他们将发现部署数十, 数百或数千个构成应用程序的容器的挑战。跟踪和管理部署需要高级管理和编排。

### DevOps

Docker 平台为开发人员提供了可用于以下功能的工具和服务:

- 通过映像的中央存储库构建和共享映像。
- 通过使用版本控制来协作开发容器化应用程序。
- 管理应用程序的基础架构。

- Docker 的一些常见使用场景包括:
  - 容器编排
  - DevOps
  - 微服务

Docker 帮助开发人员团队在任何规模下快速构建, 测试, 部署和运行分布式应用程序和服务。由于容器化应用程序消除了解决软件依赖关系问题和主机环境之间的差异的问题, Docker 提高了开发人员的生产力, 并允许您将应用程序从开发, 测试到生产快速移动, 如果需要进一步修复, 您可以轻松地将应用程序回滚 ( roll apps back ) 。

另一个重要的成就是 Docker for Windows 支持卷挂载 ( volume mounting ) , 这意味着容器可以在本地设备上看到您的代码。为了实现这一点, 该工具在容器和主机之间建立一个连接。实际上, 这使得编辑和刷新 ( edit-and-refresh ) 场景可用于开发。

使用 Docker for Windows, 您现在可以在以下场景下使用 Microsoft Visual Studio 的 Docker 工具:

- 用于调试和发布配置的 Docker 资产被添加到项目中

- 将 Windows PowerShell 脚本添加到项目中以协调容器的构建和组合，使您能够在保持 Visual Studio 设计器体验的同时扩展它们。
- 配置卷映射（volume mapping）后，在 Debug 配置中 F5 启动 Windows PowerShell 脚本，以构建和运行 docker-compose.debug.yml 文件。
- 在发布配置中的 F5 启动 Windows PowerShell 脚本来构建和运行您的 docker-compose.release.yml 文件，它允许您验证并将映像推送到您的 Docker 存储库用于部署到另一个环境。

### 微服务（Microservices）

使用 Docker 容器的另一个好处是它们可以单独扩展和更新。微服务是应用程序开发的一种方法，应用程序的每个部分都部署为完全独立的组件。例如，从互联网接收请求的应用程序的子系统可能与将请求放入用于将其写入到数据库中的后端子系统队列中的子系统分开。

当使用微服务构建应用程序时，每个子系统都是微服务。在单个机器上的开发或测试环境中，微服务可能各有一个实例，但是当应用程序在生产环境中运行时，每个微服务都可以扩展到多个实例，跨越基于资源需求的服务器群集。

在这种情况下使用 Docker 容器的一些好处包括：

- 每个微服务可以快速向外扩展以满足增加的负载。
- 容器的命名空间和资源隔离也防止一个微服务实例干扰其他微服务实例。
- Docker 打包格式和应用程序编程接口（API）为微服务开发人员和应用程序操作员解锁了 Docker 生态系统。

通过良好的微服务架构，您可以解决管理，部署，编排和修补基于容器的服务的需求，同时降低可用性损失的风险，同时保持高敏捷性。

## 安装和配置 Docker（Installing and configuring Docker）

Windows Server 2016 不包括 Docker 引擎，您可以单独安装和配置它。在 Windows 操作系统上运行 Docker 引擎的步骤与 Linux 上的不同。这些说明详细介绍了如何在 Windows Server 2016 和 Nano 服务器上安装和配置 Docker 引擎，以及如何安装 Docker 客户端的说明。Docker 由 Docker 引擎和 Docker 客户端组成。



**注意：**您必须启用 Windows 容器功能，然后 Docker 才能创建和管理 Windows 容器。有关启用此功能的信息，请参阅本单元前面部署 Windows 容器的说明。

- 要安装 Docker 到 Windows Server 2016：
  - 安装 OneGet PowerShell 提供程序模块
  - 使用 OneGet 安装 Docker
  - 重新启动容器主机
- 将 Docker 安装到 Nano 服务器：
  - 创建远程 PowerShell 会话
  - 安装 Windows 更新
  - 安装 Docker
  - 配置安全设置
  - 安装 Docker 客户端

### 在 Windows Server 2016 上安装 Docker（Installing Docker on Windows Server 2016）

您需要安装 Docker 的第一个组件是 PackageManagement 或 OneGet 提供程序 Windows PowerShell 模块。OneGet 为包管理系统提供了一个统一的接口，并为 SDII 工作提供了一组通用的 cmdlet。无论底层技术如何，用户都可以运行这些 cmdlet 来安装和卸载软件包；添加，删除和查询包存储库；并查询系统安装的软件。

使用 OneGet，您可以：

- 管理可以搜索，获取和安装软件包的软件存储库列表。

- 搜索和过滤您的存储库以查找所需的软件包。
- 使用单个 Windows PowerShell 命令从一个或多个存储库无缝安装和卸载软件包。

安装 OneGet 提供程序会启用计算机上的容器功能并安装 Docker。因为 Docker 不包括在 Windows Server 2016 中, 这些命令从 Internet 下载所需的组件。在提升的 Windows PowerShell 命令提示符下, 使用以下过程安装 Docker:

1. 通过运行以下命令安装 NuGet 提供程序。

```
Install-PackageProvider -Name Nuget -Force
```

2. 使用 NuGet 提供程序通过运行以下命令从 PSGallery 安装 Docker 模块。

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

3. 通过运行以下命令安装最新版本的 Docker 安装程序包。

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

4. 通过运行以下命令重新启动计算机


```
Restart-Computer -Force
```

## 在 Nano 服务器上安装 Docker (Installing Docker on Nano Server)

与 Windows Server 2016 一样, Docker 不包括在 Nano 服务器中。虽然您将使用 OneGet 提供程序 PowerShell 模块来安装 Docker, 但需要一些先决条件才能准备计算机。使用以下过程在 Nano 服务器上安装 Docker:

1. 创建远程 Windows PowerShell 会话。由于 Nano 服务器不支持交互式登录功能, 因此您需要使用远程 Windows PowerShell 来管理服务器。使用以下 Windows PowerShell 命令将 Nano 服务器添加到管理计算机上的受信任主机列表, 然后连接到 Nano 服务器。

```
Set-Item WSMan:\localhost\Client\TrustedHosts <NanoServerIP> -Force
Enter-PSSession -ComputerName <NameServerIP> -Credential <Administrator>
```


 **注意:** 将<NanoServerIP>替换为 Nano 服务器的 IP 地址。将<Administrator>替换为在 Nano 服务器上具有管理员权限的用户名。

2. 安装 Windows 更新。需要重要更新才能使 Windows 容器功能正常运行。使用以下 Windows PowerShell 命令更新 Nano 服务器。

```
$session = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName MSFT_WUOperationsSession
Invoke-CimMethod -InputObject $sess -MethodName ApplyApplicableUpdates
Restart-Computer
```

3. 安装 Docker。使用以下 Windows PowerShell 命令安装 NuGet 提供程序, 然后安装 Docker。

```
Install-PackageProvider -Name NuGet -Force
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
Install-Package -Name docker -ProviderName DockerMsftProvider
```

 **注意:** 在运行步骤 3 中的命令之前, 需要使用步骤 1 中的说明将远程 Windows PowerShell 会话重新连接到 Nano 服务器。

## 配置安全设置 (Configuring security settings)

因为要从另一台计算机管理 Nano 服务器上的 Docker，您需要使用以下防火墙和安全设置来配置 Nano 服务器。

1. 为 Docker 连接创建防火墙规则。使用以下命令启用到 Nano 服务器的 Docker 连接。

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

2. 配置 Docker 引擎以接受通过 TCP 的传入连接。使用以下 Windows PowerShell 命令启用到 Nano 服务器的连接配置。

```
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. 重新启动 Docker 服务。使用以下 Windows PowerShell 命令在 Nano 服务器上重新启动 Docker 服务。

```
Restart-Service docker
```



**注意：**将端口 2375 用于未加密连接，将端口 2376 用于加密连接。

## 安装 Docker 客户端 (Installing the Docker client)

Docker 客户端包括一个预配置为 Docker 命令行环境 (CLI) 的命令外壳，您应该将其安装在容器主机或任何其他将运行 Docker CLI 命令的系统上。当在 Nano 服务器上管理容器主机时，Docker 客户端经常部署在远程计算机上。在提升的 Windows PowerShell 命令提示符下，使用以下过程安装 Docker 客户端：

1. 下载 Docker 客户端。使用以下 Windows PowerShell 命令下载 Docker 客户端并将软件包解压缩到远程计算机。

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

2. 配置 Docker 的系统路径。使用以下 Windows PowerShell 命令将 Docker 目录添加到系统路径。

```
# For quick use, does not require shell to be restarted.
$env:path += ";c:\program files\docker"
# For persistent use, will apply even after a restart.
[Environment]::SetEnvironmentVariable("Path", $env:Path + ";C:\Program Files\Docker",
[EnvironmentVariableTarget]::Machine)
```

3. 使用 Docker 主机参数。例如，使用以下 Windows PowerShell 命令来访问远程 Docker 主机。

```
docker -H tcp://<NanoServerIP>:2375 run -it Microsoft/nanoserver cmd
```





**注意：**可以创建一个 DOCKER\_HOST 环境变量，该环境变量将删除 Docker 主机参数要求，如下示例所示。

```
$env:DOCKER_HOST = "tcp:// <NanoServerIP>:2375"
```

使用此变量，您可以使用以下命令来访问远程 Docker 主机。

```
Docker run -it Microsoft/nanoserver cmd
```

## 连接到容器 (Connecting to containers)

您可以使用 Docker 和 PowerShell Direct 连接到容器。当您必须执行任务（如安装软件或配置或排除容器故障）时，此功能非常有用。

使用 Docker，可以使用 `-it` 参数。例如，您可以使用以下命令连接到名为 IIS 的正在运行的容器。

```
docker exec -it IIS cmd
```

使用 Windows PowerShell，您可以使用 `Enter-PSSession cmdlet`。例如，您可以使用以下命令连接到名为 IIS 的正在运行的容器。

```
Enter-PSSession -ContainerName IIS -RunAsAdministrator
```

如果连接成功，Windows PowerShell 命令提示符将更改为包括容器的名称（在本例中为 IIS）。您现在可以使用任何 Windows PowerShell cmdlet 添加或删除角色和功能，调用脚本或在 IIS 容器中安装应用程序。

## 使用 Docker 进行管理概述 (Overview of management with Docker)

如前所述，您可以选择使用 Windows PowerShell 或 Docker 来管理 Windows 容器。在 Windows Server 上使用 Docker 的优点是 Docker 是一个行业标准的容器部署和管理工具，使管理员能够在其他操作系统上管理容器以管理您的 Windows 容器。

使用 Docker，您可以创建容器，删除容器，管理容器，并浏览 Docker Hub 以访问和下载预构建的映像。在大多数组织中，使用 Docker 的最常见的管理任务包括：

- 在 Windows 操作系统上使用 Dockerfile 自动创建容器映像。
- 使用 Docker 管理容器。
- 使用 `docker run`。

- 使用 Docker，您可以：
  - 创建容器
  - 删除容器
  - 管理容器
  - 浏览 Docker Hub 以访问和下载预构建的映像
- 在大多数组织中，最常见的 Docker 管理任务包括：
  - 通过在 Windows 操作系统上使用 Dockerfile 自动创建容器映像
  - 使用 Docker 管理容器
  - 使用 `docker run`

## 通过在 Windows 上使用 Dockerfile 自动创建容器映像 (Automating the creation of container images by using Dockerfile on Windows)

Docker 引擎包括用于自动创建容器映像的工具。虽然您可以手动创建容器映像，但采用自动映像创建过程有许多好处，包括：

- 能够将容器映像存储为代码。
- 为维护 and 升级目的快速，精确地重新创建容器映像。
- 容器映像和开发周期之间的连续集成。

驱动此自动化的 Docker 组件是 Dockerfile 和 docker build 命令：

- Dockerfile。此文本文件包含创建新容器映像所需的说明。这些指令包括识别用作基础的现有映像，在映像创建过程期间运行的命令，以及当容器映像的新实例部署时将运行的命令。
- docker build。Docker Engine 命令使用 Dockerfile，然后触发映像创建过程。

在其最基本的形式中，Dockerfile 可以非常简单，如下面的示例所示。

以下示例创建一个新映像，其中包括 IIS 和 “Hello world” 站点。

### Dockerfile 示例

```
# Indicates that the windowsservercore image will be used as the base image
FROM windowsservercore

# Metadata indicating an image maintainer
MAINTAINER LJackman@adatum.com


# Uses dism.exe to install the IIS role
RUN dism.exe /online /enable-feature /all /featurename:iis-webserver /NoRestart

# Creates an html file and adds content to this file
RUN echo "Hello World - Dockerfile" > c:\inetpub\wwwroot\index.html

# Sets a command or process that runs each time a container is run from the new image
CMD [ "cmd" ]
```

 **附加阅读：**有关 Windows 的 Dockerfiles 的其他示例的更多信息，请参阅 Windows 资源库的 Dockerfile，请参阅: <http://aka.ms/kq8gak>

Dockerfile 指令为 Docker 引擎提供了创建容器映像所需的步骤。这些指令按顺序，一个接一个地执行。

 **附加阅读：**有关 Dockerfile 指令的完整列表的更多信息，请参阅 Dockerfile 参考：  
<http://aka.ms/wrccuy>

您还可以使用 RUN 操作指定 Windows PowerShell 命令在 Dockerfile 中运行。以下是在 Dockerfile 中使用 Windows PowerShell 命令的选项列表：

- 您可以使用 Windows PowerShell 和 Invoke-WebRequest 命令从 Web 服务收集信息或文件。例如，您可以从供应商的网站下载 Python 编程语言，以便安装在新映像中。
- 您可以使用 Windows PowerShell 在映像创建过程中使用 Microsoft .NET WebClient 库下载文件。此方法显示可提高下载性能。



**注意：** Nano 服务器当前不支持 .NET WebClient。

- 您可能认为将 Windows PowerShell 脚本复制到在映像创建过程中使用的容器中，然后从容器中运行脚本会很有帮助。例如，以下命令通过使用 ADD 指令将 Windows PowerShell 脚本从构建机器复制到容器中，然后使用 RUN 指令运行脚本。键入以下内容，然后按 Enter 键。

```
FROM windowsservercore
ADD script.ps1 /windows/temp/script.ps1
RUN powershell.exe -executionpolicy bypass c:\windows\temp\script.ps1
```

创建 Dockerfile 并将其保存到磁盘后，可以使用 docker build 创建新映像。docker build 命令需要几个可选参数和一个到 Dockerfile 的路径。例如，以下命令创建名为 IIS 的映像。

```
docker build -t iis .
```



**附加阅读：** 有关 docker build 的更多信息，包括所有构建选项的列表，请参考：“docker build”：<http://aka.ms/u29exr>



**附加阅读：** 你可以使用几种方法优化 Docker 构建过程和最终的 Docker 映像。有关 Docker 构建过程如何操作以及可以用于使用 Windows 容器优化映像创建的策略的更多信息，请参考：“优化 Windows Dockerfiles”，网址为：<http://aka.ms/nrgyui>

## 使用 Docker 管理容器 (Managing containers by using Docker)

您可以使用 Docker 来支持容器环境。在安装 Docker 之后，使用以下命令来管理容器：

- Docker images。这将列出容器主机上安装的映像。您可能还记得，您可以使用容器映像作为新容器的基础。
- docker run。这通过使用容器映像创建容器。例如，以下命令将基于 Windows Server Core 容器映像创建名为 IIS 的容器

```
docker run --name IIS -it windowsservercore
```

- docker commit。这将提交对容器所做的更改，并创建一个新的容器映像。例如，以下命令将基于 IISBase 基本映像创建名为 WinSvrCoreIIS 的新容器映像。

```
docker commit iisbase WINSVRCOREIIS
```

- docker stop。这将停止正在运行的容器。
- docker rm。这将删除正在运行的容器。



**附加阅读：** 有关使用 Docker 在 Windows Server 上管理容器的详细信息，请参阅 Windows 容器快速入门：<https://aka.ms/slvc18>

## 使用 docker run (Using docker run)

docker run 命令是最常用的 Docker 命令。作为创建容器的一部分，您可以使用此命令在运行时定义容器的资源。

Docker 在隔离的容器中运行进程。这些容器只是一个在主机上运行的进程。主机可以是本地的或远程的。当运行 docker run 时，运行的容器进程是隔离的；例如，来自主机的单独的文件系统，网络和进程树。在

执行期间，docker run 命令必须指定从中派生出容器的映像。开发映像时，可以定义与以下内容相关的映像默认值：

- 分离或前台运行 (Detached or foreground running)。
- 容器识别 (Container identification)。
- 网络设置。
- CPU 和内存的运行时约束 (Runtime constraints on CPU and memory)。

使用 docker run, 您可以添加或覆盖在映像开发过程中配置的映像默认值。此外，您可以覆盖 Docker 运行时本身设置的几乎所有默认值。重写映像和 Docker 运行时默认值的能力是为什么 docker run 有比任何其他 Docker 命令更多的选项的原因。

可选的 cmd 开关打开与容器的交互会话，以包括默认命令或其他选项。虽然您可能在创建映像时使用 Dockerfile CMD 指令提供了默认的 COMMAND 命令，但是通过指定一个新的 COMMAND 命令从映像运行容器时可以覆盖该 CMD 指令。与此命令等效的 Windows PowerShell cmdlet 是 Start-Container。要结束与容器的交互式会话，请键入 exit。



**注意：**现在可以使用 Windows PowerShell 命令在容器中安装所需的角色和功能。例如，powershell.exe Install-WindowsFeature Web-Server 在您的容器中安装 IIS 组件。

Docker 平台简化了跨容器选项工作的体验。使用 Windows Server 容器开发的应用程序可以作为 Hyper-V 容器进行部署，而无需更改。事实上，使用 Docker 管理 Hyper-V 容器几乎与使用 Docker 管理 Windows Server 容器相同。当使用 docker run 创建 Hyper-V 容器时，包括 `--isolation=hyperv` parameter.

例如，以下命令启动 Windows Server 容器并托管长时间运行的 ping 进程。

```
docker run -d windowsservercore ping localhost -t
```

相反，此示例还启动 Hyper-V 容器并托管长时间运行的 ping 进程。

```
docker run -d --isolation=hyperv nanoserver ping -t localhost
```



**附加阅读：**有关使用 docker run 命令在运行时定义容器资源的更多信息，请参阅 Docker 运行参考：<http://aka.ms/Xjef2h>

## Docker Hub 概述 ( Overview of Docker Hub )

Docker Hub 是一个基于云的公共存储库，Docker 公司维护它来构建和运送应用或服务容器。它为容器映像发现，分发和变更管理，用户和团队协作以及整个开发流程中的工作流自动化提供了集中的资源。

### Docker Hub 功能 ( Docker Hub features )

Docker Hub 提供以下主要特性和功能：

- 映像存储库 ( Image repositories )。 Docker Hub 包含存储在存储库中的映像，您可以从社区，官方和私有映像库中查找，管理和推送映



海量视频题库 myitpub.com QQ:5565462

像以构建容器。具体来说, 存储库包含有关这些映像的映像, 图层和元数据。容器化的主要概念是, 您可以基于现有的映像构建您自己的映像 - 这被称为图层。

- **组织和团队。**私人存储库的一个有用的方面是, 您可以只与您的组织或团队的成员共享。Docker Hub 允许您创建需要用户访问权限的组织或工作组, 您可以在其中与同事协作并管理私有存储库。
- **自动构建 (Automated builds)。**自动构建可以直接在 Docker Hub 上自动构建和更新 GitHub 或 Bitbucket 的映像。。它通过向选定的 GitHub 或 Bitbucket 存储库添加提交钩子 (commit hook) 来工作, 当您推送提交或对源存储库进行更改时触发构建和更新。
- **Webhooks。**Webhook 是附加到您的存储库的自动化构建的一个功能, 并允许您在映像或更新的映像成功推送到存储库时触发事件或操作。例如, 使用 webhook, 您可以指定目标 URL 和 JavaScript 对象符号 (JSON) 有效负载, 以便在推送映像时传递。
- **GitHub 和 Bitbucket 集成。**这允许您将 Docker Hub 和 Docker 映像添加到当前工作流程。

### 使用映像存储库 (Working with image repositories)

Docker Hub 提供了一个构建和运输 Docker 映像的地方。这些存储库使您能够与同事, 客户或 Docker 社区共享映像。您可以通过两种方式配置 Docker Hub 存储库:

- **存储库 (Repositories)。**存储库允许您将映像从本地 Docker 守护进程推送到 Docker Hub。如果您在内部构建映像 (在自己的 Docker 守护程序上或使用自己的持续集成服务), 则可以将它们推送到您添加到 Docker Hub 用户或组织账户的 Docker Hub 存储库。
- **自动构建 (Automated build)。**自动构建允许您配置 GitHub 或 Bitbucket 以触发 Docker Hub 在存储库发生更改时重建存储库。如果 Docker 映像的源代码在 GitHub 或 Bitbucket 上, 您可以使用 Docker Hub 服务来自动构建存储库 (Automated Build repository)。



**注意:** 您可以创建任何其他 Docker Hub 用户可以访问的公共存储库, 也可以创建您可以管理的具有有限访问权限的私有存储库。

Docker 通过使用四个主要的 Docker Engine CLI 命令提供对 Docker Hub 服务的访问: docker login, docker search, docker pull, 和 docker push。

### 创建用于登录的帐户 (Creating an account for sign-in)

如果你还没有这样做, 你将需要创建一个 Docker ID 来与 Docker Hub 上的存储库进行交互。要通过 Docker Hub 执行此操作, 请参阅 <http://aka.ms/Hqfvqf>。在您拥有 Docker ID 之后, 从 CLI 登录您的帐户。键入以下命令, 然后按 Enter 键。

```
$ docker login
```

从命令行登录后, 可以使用其他 Docker 引擎子命令。



**附加阅读:** 有关注册 Docker ID 的更多信息, 请参阅: “在 Docker ID 中使用 Docker Hub”, 网址为: <http://aka.ms/ya2hoo>

### 搜索映像 (Searching for images)

您可以通过两种方式搜索 Docker Hub 上提供的公共存储库和映像。您可以使用 Docker Hub 网站上的 Search 功能, 也可以使用 CLI 的 docker search 命令。这两种方法都将显示与 Docker Hub 上的公共存储库中提供的关键字匹配的当前可用映像的列表。





**注意：**来自私有存储库的映像不会显示在存储库搜索结果列表中。要查看您可以访问的所有存储库及其状态，请查看 Docker Hub 网站上的仪表板。

映像搜索结果基于映像名称，用户名或描述等标准。使用搜索条件“CentOS”从所有存储库和映像返回以下搜索结果：

```
$ docker search centos
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
centos	The official build of CentOS.	1034	[OK]	
ansible/centos7-ansible	Ansible on Centos7	43		[OK]
tianon/centos	CentOS 5 and 6, created wi...	13		[OK]
...				

在前面的示例中，您的搜索返回了三个结果：centos，ansible/centos7-ansible 和 tianon/centos。第二和第三个结果表明它们分别来自于名为 ansible 和 tianon 的公共存储库；正斜杠 (/) 字符将用户的存储库名称与映像名称分隔开。另一方面，第一个结果 centos 没有显式地列出一个仓库。在后一种情况下，这意味着映像来自 Docker Hub 官方存储库的受信任的顶级命名空间。

Docker Hub 包含一些 *官方存储库 (Official Repositories)*。这些是来自 Docker 供应商和贡献者的公共的，经认证的存储库，您可以使用它们构建应用和服务。使用官方存储库，您知道您正在使用专家建立的强大的应用程序的优化和最新的映像。



**附加阅读：**有关 Docker Hub 支持和提升的 Docker 存储库的更多信息，请参考：“Docker Hub 上的官方存储库”，网址为：<http://aka.ms/f7zl0h>

找到所需的映像后，可以使用 docker pull 命令从 CLI 下载它。在下面的示例中，您将下载最新的映像，您可以从中运行容器。

```
$ docker pull centos
Using default tag: latest
latest: Pulling from library/centos
f1b10cd84249: Pull complete
c852f6d61e65: Pull complete
7322f6e74aa5: Pull complete
Digest: sha256:90305c9112250c7e3746425477f1c4ef112b03b4abe78c612e092037bfecc3b7
Status: Downloaded newer image for centos:latest
```

如前所述，映像存储库包含有关这些映像的映像，图层和元数据。此元数据的一部分是可用于标记映像的标记。例如，您可以使用以下命令下载 centos 版本 5（centos5 是 CentOS 版本的 centos 存储库中的标记标签）。

```
docker pull centos:centos5
```

## 向 Docker Hub 提交贡献 (Contributing to Docker Hub)


虽然任何人都可以从 Docker Hub 存储库下载公开的映像，但是如果你想分享你自己的映像，你必须注册。



**附加阅读：**有关将存储库推送到 Docker Hub 存储库的详细信息，请参阅：“构建您自己的映像”：<http://aka.ms/iyggmz>

例如，您可以推送此存储库并上传您的映像，以便它可供您的团队成员和 Docker Hub 社区使用。

```
$ docker push Docker <ID>/<Image Name>
```

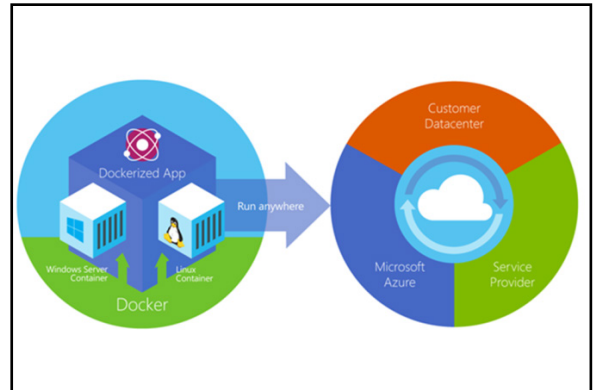
 **附加阅读：** 有关创建组织和团队以便您可以委派对共享映像存储库的同事访问权限的更多信息，请参阅：“Docker Hub 中的组织和团队”，网址为：<http://aka.ms/wzbstk>

## Docker 与 Azure ( Docker with Azure )

作为一个开源引擎，Docker 将任何应用程序的部署自动化为一个便携，自给自足的容器，几乎可以在任何地方运行，包括 Azure。典型的虚拟机映像包含运行所需的一切内容，包括应用程序，任何依赖项和操作系统。相比之下，Docker 容器包括应用程序和一些库，但操作系统和公共依赖关系保持共享资产。因此，Docker 容器与虚拟机映像相比非常轻量级。通过使 Docker 容器显着小于传统虚拟机，更多的容器可以在单个主机上运行，它们可以更快地启动，而且它们的可移植性显着提高，这些特性对于像 Azure 这样的 PaaS 非常理想。

使用 Azure，您可以灵活地根据您的需求在几种不同的方案中部署 Docker：

- 使用 Docker Machine Azure 驱动程序在 Azure 中部署 Docker 主机。
- 使用 Azure Docker VM 扩展模板部署。
- 部署 Azure 容器服务群集。



### 使用 Docker Machine Azure 驱动程序在 Azure 上部署 Docker 主机 ( Using the Docker Machine Azure driver to deploy Docker hosts on Azure )

您可以使用 Docker Machine Azure 驱动程序在 Azure 上部署 Docker 主机。Docker 是最流行的虚拟化方法之一，使用 Linux 容器而不是虚拟机作为隔离应用程序数据和共享资源上的计算的一种方式。使用这种方法的一个常见场景是当你需要快速原型化（prototype）应用程序。

您可以通过使用带有 `-d azure` Azure 驱动程序选项的 `docker-machine create` 命令在 Azure 上创建 Docker 主机虚拟机。例如，您可以使用以下命令测试 Web 应用程序。该命令创建一个名为 DockerVM 的新虚拟机，将虚拟机上的 Internet 端口 80 打开，并启用 ops 作为安全 Shell（SSH）的登录用户。

```
docker-machine create -d azure \
  --azure-ssh-user ops \
  --azure-subscription-id <Azure_Subscription_ID> \
  --azure-open-port 80 \
  machine
```

 **附加阅读：** 有关使用 Docker Machine 在 Azure 中为您的 Linux 容器创建新的 Docker 主机虚拟机的更多信息，请参阅：“使用 Docker 机器与 Azure 驱动程序”，网址为：<http://aka.ms/wjudik>

## 使用 Azure Docker VM 扩展进行模板部署 ( Using the Azure Docker VM extension for template deployments )

对于基于模板的部署，可以为 Azure 虚拟机使用 Docker VM 扩展。此方法允许您与 Azure 资源管理器模板部署集成，并包括所有相关的好处，如角色基础访问，诊断和部署后配置。 Azure Docker VM 扩展在 Linux VM 中安装和配置 Docker 守护程序，Docker 客户端和 Docker Compose。

您还可以使用扩展使用 Docker Compose 定义和部署容器应用程序。 Azure 资源管理器模板 ( Azure Resource Manager template ) 使您能够在整个开发生命周期中部署解决方案，并确信您的资源以一致的状态部署。使用 Azure Docker VM 扩展非常适合强大的开发或生产环境，因为与仅使用 Docker Machine 或手动创建 Docker 主机相比，您还需要一些额外的控制。

使用 Azure 资源管理器，您可以创建和部署定义环境的整个结构的模板，例如 Docker 主机，存储，基于角色的访问控制 ( RBAC ) 和诊断。使用资源管理器模板而不是简单地使用 Docker Machine 的优点是，您可以定义额外的 Docker 主机，存储和访问控制，并且可以根据需要重现部署。



**附加阅读：**有关详细信息，请参阅：“Azure 资源管理器概述”，地址为：

<http://aka.ms/p35huz>

## 部署 Azure 容器服务群集 ( Deploying an Azure Container Service cluster )

Azure 容器服务提供了流行的开源容器群集和协调解决方案 ( orchestration solution ) 的快速部署。使用 Azure 容器服务，您可以使用 Azure 资源管理器模板或 Azure 门户部署群集，如 Docker Swarm 群集。 Docker Swarm 群集是利用 Docker Swarm 提供的额外调度和管理工具的生产就绪，可扩展部署的理想选择。

Docker Swarm 使用本机 Docker API 提供了一个环境，用于跨池集合的 Docker 主机部署容器化工作负载。在部署 Docker Swarm 群集期间，您将使用 Azure 计算资源和 Azure VM 扩展集合来管理一组 VM。



**附加阅读：**有关使用 Azure 容器服务部署 Docker Swarm 群集的更多信息，请参阅：“部署

Azure 容器服务群集”，网址为：<http://aka.ms/F8azgy>

通过在右侧的列中放置 true 或 false 来验证语句正确与否。

声明	答案
Docker 是一个图形管理工具，可用于在 Windows Server 2016 中管理 Hyper-V 容器。	

## 演示：使用 Docker 部署容器 ( Deploying containers by using Docker )

在本演示中，您将了解如何：

- 安装 OneGet PowerShell 提供程序模块。
- 安装 Docker。
- 下载映像。
- 部署新容器。
- 管理容器。

## 演示步骤 ( Demonstration Steps )

### 安装 OneGet 提供模块 ( Install the OneGet provide module )

1. 在 LON-NVHOST2 上, 打开 *Windows PowerShell*.
2. 在 Windows PowerShell 命令提示符下, 键入以下命令以安装 NuGet 提供程序:

```
Install-PackageProvider -Name NuGet -Force
```

### 安装 Docker ( Install Docker )

1. 在 Windows PowerShell 命令提示符下, 键入以下内容以安装 Docker:

```
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
Install-Package -Name Docker -ProviderName DockerMsftProvider
```

2. 键入以下命令以重新启动计算机:

```
Restart-Computer -Force
```

### 下载映像 ( Download an image )

1. 在 Windows PowerShell 命令提示符下, 键入以下内容以在 Docker Hub 中搜索 Windows 容器映像:

```
docker search microsoft
```

2. 要下载 IIS 映像, 请键入以下内容:

```
docker pull Microsoft/iis:windowsservercore
```

### 部署新的容器 ( Deploy a new container )

1. 在 Windows PowerShell 命令提示符窗口中, 键入以下命令以部署 IIS 容器:

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



**注意:** 此命令作为后台服务 (-d) 运行 IIS 映像, 并配置网络, 使容器主机的端口 80 映射到容器的端口 80。

2. 键入以下内容以检索容器主机的 IP 地址信息:

```
ipconfig
```

3. 注意名为 vEthernet ( HNS 内部 NIC ) 的以太网适配器的 IPv4 地址。这是新容器的地址。记下名为以太网的以太网适配器的 IPv4 地址。这是容器主机的 IP 地址。
4. 在 LON-HOST1 上, 打开 Internet Explorer.
5. 在地址栏中, 键入以下内容

```
http://<ContainerhostIP>
```



**注意:** 将<ContainerhostIP>替换为容器主机的 IP 地址

6. 观察默认 IIS 页面.

## 管理容器 (Manage the container)

1. 在 LON-NVHOST2 中，在 Windows PowerShell 命令提示符窗口中，键入以下内容以查看正在运行的容器：

```
docker ps
```

2. 记下容器 ID。
3. 键入以下命令停止容器，然后按 Enter 键：

```
docker stop <ContainerID>
```



**注意：**将<ContainerID>替换为容器 ID。

4. 在 LON-HOST1 上，打开 Internet Explorer。
5. 在地址栏中，键入以下内容。

```
http://<ContainerhostIP>
```

6. 请确保不再可以访问默认 IIS 页面。这是因为容器没有运行。
7. 在 LON-NVHOST2 中，在 Windows PowerShell 命令提示符窗口中，键入以下内容以删除容器。

```
docker rm <ContainerID>
```



**注意：**将<ContainerID>替换为容器 ID。



## 实验: 安装和配置容器 ( Installing and configuring containers )

### 场景 ( Scenario )

A. Datum Corporation 的 DevOps 团队希望探索容器, 看看这项技术是否能帮助减少新应用程序的部署时间, 并简化将应用程序移到云中。团队决定评估 Windows Server 容器并在容器中查看 IIS。

### 实验设置 ( Lab Setup )

估计时间: 45 分钟

用户名称: Adatum\Administrator

密码: Pa55w.rd

物理机 : 28740B-LON-HOST1

虚拟机 : 28740B-LON-NVHOST2 , 28740B-LON-DC1-B , 28740B-NAT

在本实验中, 您将使用 28740B-LON-HOST1 主机。此主机计算机应从上一个实验室运行。如果不是, 请完成以下步骤:

1. 重新启动教室计算机, 然后在 Windows Boot Manager 中, 选择 28740B-LON-HOST1。
2. 使用以下凭据登录 LON-HOST1 :
  - 用户名: Adatum\Administrator
  - 密码: Pa55w.rd
3. 在 LON-HOST1 上, 打开服务器管理器。
4. 单击 Local Server , 然后单击 vEthernet (Host Internal Network) 。
5. 在 Network Connection 页面中, 双击 vEthernet (Host Internal Network) 。
6. 单击 Properties , 然后双击 Internet Protocol Version 4 (TCP/IPv4) 。
7. 单击 Use the following IP address , 然后输入以下信息 :
  - IP 地址 : 172.16.0.160
  - 子网掩码 : 255.255.0.0
  - 默认网关 : 172.16.0.1
  - 首选 DNS 服务器 : 172.16.0.10
8. 单击 OK , 然后单击 Close 两次。
9. 如果出现 Microsoft TCP/IP 对话框, 通知您有关具有相同 IP 地址的另一个禁用的适配器, 请单击 No 删除静态 IP 配置。
10. 打开 Hyper-V 管理器。
11. 单击 Virtual Switch Manager , 然后单击 Physical Network 。
12. 在 Connection type 下, 清除 Allow management operating system to share this network adapter 复选框, 然后单击 OK 。
13. 在 Apply Networking Changes 对话框中, 单击 Yes 。
14. 启动 28740B-LON-DC1-B 和 28740B-NAT 虚拟机。
15. 连接到 28740B-NAT , 然后使用以下凭据登录 :

- 用户名称: Administrator
  - 密码: Pa55w.rd
16. 打开 Internet Explorer 并验证服务器是否可以连接到 Internet。服务器配置有连接到主机上的物理网络适配器的网络适配器，并且被配置为使用 DHCP 获得该网络上的 IP 地址。如果虚拟机无法访问 Internet，请向教师询问如何配置外部网络以启用 Internet 访问。
  17. 连接到 LON-NVHOST2，然后使用以下凭据登录：
    - 用户名: Adatum\Administrator
    - 密码: Pa55w.rd

## 练习 1: 安装 Docker ( Installing Docker )

### 场景 ( Scenario )

部署容器的第一步是安装 NuGet 包提供程序，然后安装 Docker。Docker 需要在 Windows Server 2016 中管理容器。

本练习的主要任务如下：

1. 安装 NuGet 包提供程序。
2. 安装 Docker。

#### ► 任务 1: 安装 NuGet 包提供程序

1. 在 LON-NVHOST2 上，打开 Windows PowerShell。
2. 在 Windows PowerShell 命令提示符下，键入以下命令以安装 NuGet 提供程序：

```
Install-PackageProvider -Name NuGet -Force
```

#### ► 任务 2: 安装 Docker

1. 在 LON-NVHOST2 上，打开 Windows PowerShell。
2. 在 Windows PowerShell 命令提示符下，键入以下内容以安装 Docker，然后按 Enter 键：

```
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
Install-Package -Name Docker -ProviderName DockerMsftProvider
```

3. 键入以下命令以重新启动计算机，然后按 Enter 键：

```
Restart-Computer -Force
```

**结果：**完成此练习后，您应该已经安装了 NuGet 包提供程序和 Docker。

## 练习 2: 安装和配置容器 ( Installing and configuring a container )

### 场景 ( Scenario )

现在您已准备容器主机，下一步是下载并部署容器映像。然后，您将能够验证容器是否已部署并正常运行，并且可以进行管理。

本练习的主要任务如下：

1. 下载映像。
2. 部署新容器。
3. 管理容器。
4. 准备下一个模块。


### ► 任务 1: 下载映像 ( Download an image )

1. 在 LON-NVHOST2 上，打开 Windows PowerShell。
2. 在 Windows PowerShell 命令提示符下，键入以下内容以在 Docker Hub 中搜索 Windows 容器映像，然后按 Enter 键：

```
docker search microsoft
```

3. 要下载 IIS 映像，请键入以下内容，然后按 Enter 键：

```
docker pull Microsoft/iis:windowsservercore
```

 **注意：**在此步骤中，您将下载并提取数 GB 的数据。根据您的互联网连接，这可能需要几分钟。

### ► 任务 2: 部署新的容器

1. 在 Windows PowerShell 命令提示符窗口中，键入以下命令以部署 IIS 容器，然后按 Enter 键：

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

 **注意：**此命令作为后台服务（-d）运行 IIS 映像，并配置网络，使容器主机的端口 80 映射到容器的端口 80。

2. 键入以下内容以检索容器主机的 IP 地址信息：

```
ipconfig
```

3. 注意名为 vEthernet ( HNS 内部 NIC ) 的以太网适配器的 IPv4 地址。这是新容器的地址。记下名为以太网的以太网适配器的 IPv4 地址。这是容器主机的 IP 地址。
4. 在 LON-HOST1 上，打开 Internet Explorer。
5. 在地址栏中，键入以下内容：

```
http://<ContainerhostIP>
```

 **注意：**将<ContainerhostIP>替换为容器主机的 IP 地址。

6. 观察默认 IIS 页面。

### ► 任务 3: 管理容器

1. 在 LON-NVHOST2 上，在 Windows PowerShell 命令提示符窗口中，键入以下内容以查看正在运行的容器，然后按 Enter 键。

```
docker ps
```

2. 记下容器 ID。
3. 键入以下命令停止容器，然后按 Enter 键：

```
docker stop <ContainerID>
```



**注意：**将<ContainerID>替换为容器 ID。

4. 在 LON-HOST1 上，打开 Internet Explorer。
5. 在地址栏中，键入以下内容：

```
http://<ContainerhostIP>
```

6. 请确保不再可访问默认 IIS 页面。这是因为容器没有运行。
7. 在 LON-NVHOST2 中，在 Windows PowerShell 命令窗口中，键入以下内容以删除容器：

```
docker rm <ContainerID>
```



**注意：**将<ContainerID>替换为容器 ID..

### ► 任务 4: 准备下一个单元

- 将主机作为 LON-HOST1 启动。

**结果：**完成此练习后，您应已部署和管理了一个容器。

## 单元复习和作业 ( Module Review and Takeaways )

### 常见问题和故障诊断技巧 ( Common Issues and Troubleshooting Tips )

常见问题	排错技巧
无法下载软件包提供者。一些错误包括“位传输失败”。	

### 复习题 ( Review Questions )

**问题：**在使用 Windows PowerShell cmdlet `New-NanoServerImage` 为 Nano 服务器创建虚拟硬盘时，你什么时候使用 `-Guestdrivers` 开关？

**问题：**当使用 Nano Server Recovery Console 时，您可以配置哪两个基本组件？

**问题：**配置 Windows Server 容器时，您使用那个 Windows PowerShell cmdlet 创建容器，以及于此功能对应的的 Docker 命令是什么？



海量视频题库 myitpub.com QQ:5565462