第1章 背景介绍

自从2006年Windows PowerShell（第1版）面世以来，我们就一直在致力于对该技术进行教学推广。那时候，PowerShell的大部分使用者都是长期使用VBScript的用户，而且他们也非常期待能以VBScript作为基础学习PowerShell。于是，开展培训以及编写PowerShell书籍的作者都采用了一种和其他编程语言教学一样的方式来教学PowerShell。

但是从2009年开始发生了一些改变。越来越多没有VBScript经验的人开始学习PowerShell这门语言。因为之前我们主要关注于脚本的编写，所以对PowerShell的教学不再那么卓有成效。也就是在那个时候，我们意识到PowerShell并不仅仅是一门脚本语言，其实是一种运行命令行工具的命令行Shell。和其他优秀的Shell一样，虽然PowerShell可以通过脚本实现很复杂的功能，但脚本仅是使用PowerShell的一种方式，因此学习PowerShell并不一定需要从脚本开始。之后，我们在每年的技术演讲会议上逐渐改变了我们的教学方式，同时也将这些教学方式的变化体现在我们的教学课程中。最后，我们出版了这本书，这也是我们想出的针对非编程背景的人员教学PowerShell的最好方式。但是在开始学习之前，我们需要了解一下背景。

1.1 为什么要重视PowerShell

从Batch、KiXtart、VBScript到现在，可以看到Windows PowerShell并不是微软（或者其他公司）首次为Windows管理员提供自动化管理的工具。我们认为，有必要让你们了解为什么需要关注PowerShell这个工具。因为当你们这样做的时候，会发现花费一定的时间去学习PowerShell是值得的。想象一下，在没有使用PowerShell之前我们的工作是怎样的？在使用该工具后又有哪些变化？

1.1.1 没有 PowerShell

Windows 操作系统管理员总是喜欢通过单击用户图形化界面去完成他们的工作。用户图形化界面（GUI）是 Windows 操作系统的最大特点——毕竟这个操作系统的名字并不是“Text”。因为 GUI 总是让我们很轻易找到我们能做的一切，所以它是那么强大。作者仍然记得第一次展开活动目录下的用户和计算机的场景。通过单击各种按钮、阅读工具栏提示信息、选择下拉菜单、右键单击某些图标来查看用户与计算机中的各项功能。GUI 是使得我们能够更容易学习的一种工具。但是不幸地是，GUI 并不能带来任何效率提升上的回报。如你花费 5 分钟在活动目录中创建一个新的用户（合理地设想一下，需要填写大量的信息），之后再新建用户时，也不会更快。那么新建 100 个新用户就会花费 500 分钟来完成——没有其他任何办法使得我们输入信息以及单击操作更快，从而加快该过程。

微软之前也尝试去解决该问题，VBScript 可能算是其中最成功的一次尝试。你可能需要花费一小时编写一条 VBScript 语句将 CSV 文件中的新用户导入到活动目录中，但在此之后你可能只需要花费几秒钟就可以完成同样的工作。VBScript 的问题在于微软没有全心全意地对其提供支持，微软需要确保各种对象都可以通过 VBScript 访问、调用，而如果开发人员因为时间的原因或者是忘记这块知识，那么你就只能卡在那儿了。例如，想通过 VBScript 修改网卡 IP，没问题。但是，想检查网络连接的速度，那就不行了，因为没人记得可以把这个功能设置为 VBScript 可访问的形式。这也算是一种遗憾。Jeffery Snover，Windows PowerShell 的架构师，称之为“最后一英里”。你可以通过 VBScript（或者其他类似的技术）来做很多事情，但是在某些时刻总会让人失望，从来不会让我们顺利通过“最后一英里”完成之后的工作。

Windows PowerShell 正是微软公司试图改善这一缺陷的尝试，让你顺利通过“最后一英里”，进而完成工作。目前来看，该尝试非常成功。微软的多个产品组都采用了 PowerShell，第三方生态系统扩展也是基于 PowerShell，并且全球的社区专家与爱好者也都帮助 PowerShell 变得越来越好。

1.1.2 拥有 PowerShell

微软对 Windows PowerShell 的定位是我们可以通过该 Shell 管理 Windows 系统中的所有功能。微软仍然继续开发 GUI 的控制台，但是底层运行的仍然是 PowerShell 命令。通过这种方式，微软保证我们可以在该 Shell 中完成 Windows 系统中任意的工作。如果需要自动化一个重复性的任务或者完成在 GUI 中不支持的工作，那么你可以使用该 Shell 来达成所愿。

很多微软的产品都已经采用了这种开发方法，如 Exchange Server 2007 以及之后版本、Sharepoint Server 2010 以及之后版本、大部分 System Center 产品、Office 365 以及

Windows 系统中大量的组件。接下来，越来越多的产品和 Windows 系统中组件会采用这个 Shell。Windows Server 2012（首次引入 PowerShell V3）甚至可以完全通过 PowerShell 或者使用基于 PowerShell 的 GUI 工具来进行管理。这也就是为什么我们要重视 PowerShell。在接下来的几年，PowerShell 会成为越来越多的管理功能的底层实现。PowerShell 已经成为大量高层技术的基础，包括 Desired State Configuration（DSC），PowerShell Workflow 以及更多。PowerShell 无处不在！

此时，我们仔细想想：如果你正在管理一个拥有很多 IT 工程师的团队，你希望谁的职级更高，希望谁能拿更多的薪水，是每次都要花费几分钟使用 GUI 来完成一个任务的人，还是一个可以通过脚本花费几秒钟自动化完成的人？无论你是来自哪个领域的 IT 从业人员，我们都知道应该如何选择。询问一个思科的管理员、AS/400 的操作员或者 Unix 管理员，他们都会回答“我更希望选择可以借助命令行更有效率地完成工作的人员”。以后的 Windows 系统工程师可以简单分为两类，一部分会使用 PowerShell，另一部分则不会。正如 Don 在微软 2010TechEd 会议上著名的言论：我们的选择是“学习 PowerShell”，还是“来包炸薯条”？

我们很欣慰，你已经决定学习 PowerShell。

1.2 现在只剩下“PowerShell”，而不是“Windows PowerShell”

在 2016 年中期，微软迈出了在此之前不敢想象的一步，那就是完整开源了 Windows PowerShell。同时，还发布了非 Windows 版本的 PowerShell，包含 macOS 与大量 Linux 发行版。太棒了！现在，这个面向对象的 Shell 在多种操作系统上可用，并且可以被世界范围内的社区共同提升。对本书的第 3 版来讲，我们决定确保主要所讲述的 PowerShell 不仅仅是基于 Windows 平台。我们认为 PowerShell 最大的受众是 Windows 用户，但我们也希望确保你能够理解 PowerShell 是如何在其他操作系统上工作的。

1.3 本书适用读者

这本书并不是适合所有人。实际上，微软 PowerShell 团队已经定义了三类适用 PowerShell 的人群：

- 主要使用命令行以及采用第三方开发工具的管理员；
- 能将命令行和工具集成为一个更复杂的工具（之后那些缺乏经验的成员可以立即使用该工具完成相关工作）的管理员；
- 开发可重复使用的工具或者程序的管理员或者开发人员。

本书主要是针对第一类人编写的。所有人，即使是开发人员在内的所有人，也有必要理解如何使用 Shell 运行命令。毕竟，如果你正准备去开发一个工具或者编写一些命

令，那么你应该知道这个 Shell 的运行机制，这样可以确保开发出来的工具或者命令能像在 Shell 中运行得那么顺畅。

使用你有兴趣通过创建脚本自动化复杂的流程，比如新建一个用户，在学习完本书后，你可以学习到如何实现该功能，甚至可以编写自己的脚本，并且该脚本可以让其他管理员使用。但是本书并不会深入地讲解 PowerShell 的每项功能。我们的宗旨是让你能够使用 Shell，并立即应用到生产环境。

我们也会使用多种方法来演示如何将 PowerShell 关联到其他的管理工具。在后续章节中，我们会以 WMI（Windows Management Instrumentation）以及常用的命令作为示例。大体上，我们仅会介绍 PowerShell 可以与哪些技术进行关联，并讲解它们之间如何进行关联。其实，这些主题甚至都可以单独出书介绍（我们会在本书适当的地方给出对应的建议）。在本书中，我们仅仅介绍与 PowerShell 相关的部分。如果你对更深入地学习这部分技术感兴趣，我们将会提供针对后续学习的建议。简而言之，本书并不是你学习 PowerShell 所用的最后一本书，本书的定位是第一本 PowerShell 入门书。

1.4 如何使用本书

本书的理念是每天完成一章的学习。我们不需要在用餐时间阅读本书，因为我们只需要接近 40 分钟就可以完成对一章的阅读，之后再花 20 分钟去享用剩余的三明治以及进行对应的练习。

1.4.1 主要章节

第 2～25 章为本书的主要内容，算下来差不多只要花费 24 顿午餐的时间完成阅读。这也就意味着你可以预期在一个月内完成对本书主要章节的阅读。你需要尽可能严格地遵守该学习计划，不要感觉需要一天内阅读其他额外章节。更为重要的是，我们需要花费一定的时间完成每个章节之后的练习题，用以巩固学习成果。当然，并不是每个章节都需要花费完整的一小时，所以有时你在上班之前有更多的时间进行练习（或者吃午餐）。我们发现很多人坚持每天只学习一章会学得更快，这是因为这使得你可以有更多的时间动脑思考新主意，以及更多的时间进行练习。请不要揠苗助长，你会发现自己的学习进度会比想象得更快。

1.4.2 动手实验

在主要章节的结尾都布置了需要完成的实验题目。我们会给你对应的说明，甚至可能是一两个提示。这些动手实验的答案，我们会放在每章节的末尾，但是建议你在查看这些答案之前尽力独立完成这部分实验。

1.4.3 代码示例

贯穿全书你会遇到代码清单。有一些比较长的 PowerShell 示例。但无需手动输入。如果你查看 www.manning.com 并找到本书的页面，就会找到本书所有代码的下载链接了。

1.4.4 进一步学习

Don 的 YouTube 频道，YouTube.com/PowerShellDon，包含大量为本书的第一版制作的免费视频——现在仍然是 100%适用。这种方式是获得一些短小、快速入门 demo 的捷径。他还是一些工作室的视频主播，这些视频都值得一看。我们还建议登入 PowerShell.org 频道，YouTube.com/powershellorg，这里包含了大量的视频内容。你会发现大量来自 PowerShell + DevOps 全球峰会、在线社区研讨会以及其他活动的视频，全部免费！

Jeff 为 Petro IT 知识库（www.petri.com）撰写过大量的文章，这里你可以发现大量的内容，涵盖 PowerShell 的各方面主题。你还可以在 Jeff 的 Youtube 频道：http://YouTube.com/jdhitsolutions 发现他的最新动态。

1.4.5 补充说明

在学习 PowerShell 的时候，有些时候我们可能会钻入死胡同，去研究为什么会这样或那样运行。如果这样学习，我们就不会学到很多实用的技能，但我们会对这个 Shell 到底是什么及其工作原理有更深入的了解。我们在“补充说明”章节中会提供这方面的信息。这些信息只需要花费几分钟就可以读完。如果你是那种喜欢钻研原理部分的人，这部分信息也可以提供一些有用的材料。如果你觉得该小节会使得你分心而不能很好地完成实践学习，那么你可以在首次阅读时忽略该小节。当然，如果你掌握了所有章节部分的主要内容，建议再返回阅读这部分。

1.5 搭建自己的实验环境

在本书的学习过程中，你会进行大量的 PowerShell 的动手实验，那么你必须构建一个属于你自己的实验环境（请记住，不要在公司的生产环境中进行测试）。

你需要在带有 PowerShell v3 或更新版本的 Windows 中运行本书中大部分示例以及完成每章节的动手实验。我们建议的环境是 Windows 8.1 或更新版本，或者是 Windows Server 2012 R2 或更新版本，这两个版本都带有 PowerShell v4。但是需要注意的是，某些版本（如简易版）的操作系统中可能不存在 PowerShell。如果你对 PowerShell 学习抱

有很大的兴趣，那么你必须找到一个带有 PowerShell 的 Windows 系统。同时，有些动手实验是基于 Windows 8 或者 Windows Server 2012 中 PowerShell 的新特性才能完成的。如果你使用的是老版本的操作系统，那么最终结果可能会有不同。在每个动手实验开始时，我们都会特别说明你需要在什么操作系统中去完成这部分实践。

在本书中，我们都是以 64 位（x64）操作系统为环境进行学习的。我们知道有两个版本：Windows PowerShell 以及特定版本的图形化 Windows PowerShell 集成脚本环境（ISE）。在开始菜单（Windows 8 中是“开始”界面），这两个组件的 64 位版本显示为“Windows PowerShell”和“Windows PowerShell ISE”。32 位版本的在快捷方式中会显示“x86”字样。在使用 x86 版本 PowerShell 时，在窗口栏中也会看到 x86 字样。如果操作系统本身就是 32 位的，那么你只能安装 32 位的 PowerShell，并且不会显示 x86 字样。

本书中的示例基于 64 位版本的 PowerShell 和对应的 ISE。如果你并不是使用的 64 位环境，那么有些时候运行示例时可能和我们得出的结果不一致，甚至某些动手实验部分根本无法正常进行。32 位版本的 PowerShell 主要是针对向后兼容性。例如，一些 Shell 扩展程序只存在于 32 位 PowerShell 中，并且也只能导入到 32 位（或者 x86）的 Shell 中。除非你确实需要使用这部分扩展程序，否则我们建议你在 64 位操作系统上使用 64 位的 PowerShell。微软后续主要的精力会放在 64 位 PowerShell 上；如果你现在因为使用的 32 位操作系统而无法进行下去，那么很遗憾，以后仍然会无法继续进行。

提示：我们完全可以在一个独立操作系统的 PowerShell 环境中完成本书的所有学习。但是如果使用同一个域的两台或者三台计算机的 PowerShell 环境联合起来进行测试，那么某些动手实验可能会变得更有趣。在本书中，我们在 CloudShare（CloudShare.com）上创建多个虚拟机来解决该问题。如果你对这种场景感兴趣，你可以了解一下这个服务或者其他类似的一些服务。但是需要注意，并不是在所有国家都可以访问 CloudShare.com。另一种解决方案是使用 Windows 8 或更新版本的操作系统中的 Hyper-V 功能来承载几台虚拟机。

如果使用的是非 Windows 版本的 PowerShell，你需要考虑几个选项。首先是从 http://github.com/PowerShell/PowerShell 的上获取适合你的操作系统（MacOS 或 Linux 等）的发行版，然后就可以开始了。但请记住，本书示例中大量的功能只有在 Windows 下可用。例如，你无法获得 Linux 的服务列表，这是由于 Linux 没有服务的概念（Linux 有守护进程，类似 Windows 的服务，但略有区别）。

1.6 安装 Windows PowerShell

从 Windows Server 2008、Windows Server 2008 R2、Windows 7 操作系统开始，我们已经可以使用第 3 版的 Windows PowerShell。Windows Vista 操作系统无法支持第 3 版，但是可以使用第 2 版 PowerShell。最近发布的几个操作系统中已经预装了 Windows

PowerShell。如果采用老版本的操作系统，那么必须手动安装 PowerShell。PowerShell v4 在 Windows 7 或 Windows Server 2008 R2 以及更新版本的操作系统上可用。虽然这些版本的 Windows 上并不是所有的组件都与 PowerShell“关联”，这也是为什么我们推荐使用 Windows 8 或 Windows 2012 作为最低版本。当然，新版本的操作系统可能会采用更新版本的 PowerShell，当然这没什么坏处。

提示：你可以采用如下方法来检查安装的 PowerShell 版本：进入 PowerShell 控制台，输入`$PSVersionTable`，然后按回车键。如果返回错误或者输出结果并未显示为“PSVersion 4.0”，那么你安装的版本就不是 PowerShell 第 4 版。

如果你想要检查最新的 PowerShell 可用版本或下载 PowerShell，请访问 http://msdn.microsoft.com/owershell。该官方 PowerShell 主页有一个指向最新版本 Windows 管理框架（WMF）安装包的链接，该安装包用于安装 PowerShell 与其相关功能。再次声明，由于本书的内容是入门级，你不会发现太多 v3 版本之后的变更，但使用最新版 PowerShell 总是很有乐趣。

PowerShell 包含两个应用程序组件：基于文本的标准控制台（PowerShell.exe）和集成了命令行环境的图形化界面（ISE；PowerShell_ISE.exe）。我们大部分时间都会使用基于文本的控制台。当然，如果你更喜欢 ISE，也可以使用 ISE。

注意：PowerShell ISE 组件并没有预装到 Server 版操作系统中。如果你需要使用，那么你需要进入 Windows 的功能（使用“服务器管理器”），然后手动添加 ISE 功能（你也可以打开 PowerShell 的控制台，再运行 `Add-WindowsFeaturePowerShell-ise`）。在未包含完整 GUI 模式的操作系统（如 Server Core 或 Nano Server 版本的系统）对应的安装程序中并没有包含 ISE 的安装程序。

在你继续学习 PowerShell 之前，建议花几分钟设置 Shell 的显示界面。如果你使用基于文本的控制台，那么强烈建议你修改显示的字体为 Lucida（固定宽度），不要使用默认的字体。假如使用默认字体，我们会很难去区分 PowerShell 使用的一些特殊字符。可以参照下面的步骤修改显示字体。

（1）右键单击控制台界面上侧边框（PowerShell 字符位于控制台界面的左上方），选择目录中的属性。

（2）在弹出的会话框中，可以在几个标签页中修改字体、窗口颜色、窗口大小和位置等。

提示：强烈建议窗口大小和屏幕缓冲器使用相同的宽度。

另外，需要注意的是，当应用对默认控制台的修改之后，后续所有新开的窗口都会使用变更之后的设置。当然，所有这些设置仅仅应用于 Windows：在非 Windows 操作系统中，你通常会安装 PowerShell，打开操作系统的命令行（例如，一个 Bash shell），然后运行 powershell。控制台窗口会控制颜色、屏幕布局等，因此请调整命令行从而满足你的需求。

1.7 联系我们

我们对帮助向你一样学习 Windows PowerShell 的人充满激情，我们会尽可能地提供我们所知道的资源。我们同时也期望你的反馈，因为这会帮助我们为新的资源想出新的主意，然后我们就可以把这部分资源放到网站上，这也是一种帮助我们提升本书下一版的方式。你可以在 Twitter 的@concentratedDon 找到 Don 以及@JeffHicks 找到 Jeff。我们还经常会在 http://PowerShell.org 上回答问题。http://PowerShell.org 也是一个寻找资源的好地方，这些资源包括免费的电子书、年度现场会议、免费的在线研讨会等。我们也为这两个地方添砖加瓦，在你完成本书之后，这两个地方是我们推荐给你继续学习 PowerShell 最好的地方。

1.8 赶紧使用 PowerShell 吧

“可以立即使用”是我们编写本书的一个主要目标。我们在每一章中尽可能仅关注某一部分的知识，并且你在学习之后，可以立即在生产环境中使用。这就意味着，在开始的时候，我们可能会避开一些细节的讨论，但是在必要时，我们承诺后续会回到这些问题并给出详细说明。在很多情形下，我们必须在首先给出 20 页的理论或者直接讲解并完成某些部分的学习（暂不解释、分析其中的细微差别或者详细情况）中做出选择。当需要做出这类选择时，我们总是选择第二个，以便使得你可以立即使用起来。但是之前的那些细节，我们会在另外一个时间进行分析讲解。

好了，背景知识大概就介绍到这里。下面就开始第 2 章课程的学习。