



Microsoft
Official
Course



28532B

开发Microsoft Azure
解决方案

预览版



MICROSOFT
OFFICIAL MICROSOFT LEARNING PRODUCT
MCT USE ONLY. STUDENT USE PROHIBITED

OFFICIAL MICROSOFT LEARNING PRODUCT

28532B

开发 Microsoft Azure 解决方案

本文档中的信息（包括 URL 和其他 Internet 网站引用）可能会随时发生变化，恕不另行通知。除非另有说明，否则文中描述的示例公司、组织、产品、域名、电子邮件地址、徽标、人物、地点和事件均属虚构，无意与任何真实的公司、组织、产品、域名、电子邮件地址、徽标、人物、地点或事件关联，也不应妄加推测。用户有责任遵守所有适用的版权法。在不限制版权的情况下，未经微软公司明确的书面许可，不得出于任何目的、以任何形式或方式（包括电子方式、机械方式、影印、录制或其他方式）对本文档的任何部分进行复制、存储，将其引入检索系统或进行传播。

微软可能具有涵盖本文档主题的专利、专利申请、商标、版权或其他知识产权。除了微软的任何书面许可协议中明确规定之外，提供本文档并不意味着就这些专利、商标、版权或其他知识产权对您授予任何许可。

提供的制造商名称、产品或 URL 仅供参考。微软不就这些制造商或这些产品与任何微软技术的一起使用做出任何明示、默示或法定的声明和保证。包含某个制造商或产品并不意味着微软对该制造商或产品的认可。文中可能会提供指向第三方网站的链接。此类网站不在微软的控制范围内，微软不对任何链接网站的内容、链接网站中包含的任何链接或此类网站的任何更改或更新负责。微软不对接收自任何链接网站的网络广播或任何其他形式的信息传输负责。微软只是为了提供方便而向您提供这些链接，包含任何链接并不表示微软认可相应的网站或其中包含的产品。

© 2015 微软公司。保留所有权利。

Microsoft 以及 <http://www.microsoft.com/about/legal/en/us/IntellectualProperty/Trademarks/EN-US.aspx> 上列出的商标是微软公司集团的商标。所有其他商标属于其各自的所有者。

课程号：28532B

发布日期：04/2015

微软许可条款

MICROSOFT 教师指导课件

这些许可条款是微软公司（或您所在地的微软公司的关联公司）与您之间达成的协议。请阅读条款内容。这些条款适用于您对本协议附带的内容的使用，包括您用来接收这些内容的介质（如果有）。这些许可条款也适用于教师内容以及对许可内容的任何更新和补充，除非这些项目附带有其他条款。如果确实附带有其他条款，应遵守那些条款。

访问、下载或使用许可内容，即表示您接受这些条款。如果您不接受这些条款，请不要访问、下载或使用许可内容。

如果您遵守这些许可条款，则依据每个所获得的许可，您拥有以下权利。

1. 定义。

- a. “授权学习中心”是指 Microsoft IT Academy 计划成员、Microsoft 学习能力成员或 Microsoft 不时指定的其他类似实体。
- b. “授权培训课程”指使用 Microsoft 教师指导课件的教师指导培训课程，由教师传授或通过授权学习中心传授。
- c. “教室设备”指由授权学习中心拥有或控制的、位于授权学习中心培训设施的一 (1) 台专用安全计算机，该设备应达到或超过针对特定 Microsoft 教师指导课件指定的硬件级别。
- d. “最终用户”指符合以下条件之一的个人：(i) 正式报名并参加授权培训课程或私人培训课程；(ii) MPN 成员的员工；或 (iii) Microsoft 全职员工。
- e. “许可内容”指本协议附带的、可能包括 Microsoft 教师指导课件或教师内容的内容。
- f. “Microsoft 认证教师”或“MCT”指符合以下条件之一的个人：(i) 代表授权学习中心或 MPN 成员，向最终用户传授培训课程；(ii) 当前已通过 Microsoft 认证计划正式认证为 Microsoft 认证教师。
- g. “Microsoft 教师指导课件”指向 IT 专业人员和开发人员传授 Microsoft 技术知识的 Microsoft 品牌教师指导培训课程。Microsoft 教师指导课件标题可能使用 MOC、Microsoft Dynamics 或 Microsoft Business Group 课件品牌标志。
- h. “Microsoft IT Academy 计划成员”指 Microsoft IT Academy 计划的有效成员。
- i. “Microsoft 学习能力成员”指信誉良好且当前具有“学习能力”状态的 Microsoft 合作伙伴网络计划有效成员。
- j. “MOC”指称为“Microsoft 正式课程”的“Microsoft 正式学习产品”教师指导课件，旨在向 IT 专业人员和开发人员传授有关 Microsoft 技术的知识。
- k. “MPN 成员”指 Microsoft 合作伙伴网络计划的信誉良好的有效成员。
- l. “个人设备”指您个人拥有或控制的一 (1) 台个人计算机、设备、工作站或其他数字电子设备，该设备达到或超过针对特定 Microsoft 教师指导课件指定的硬件级别。
- m. “私人培训课程”指由 MPN 成员为企业客户提供的教师指导培训课程，该课程使用 Microsoft 教师指导课件，旨在达到预先确定的学习目标。这些课程不向公众宣传或推广，课程参与者仅限于企业客户雇佣或签约的个人。

- n. “教师”指 (i) 与 Microsoft IT Academy 计划成员签约以传授授权培训课程、并在学术上经过认证的培训师；和/或 (ii) MCT。
- o. “教师内容”指 Microsoft 教师指导课件和其他补充内容的教师版本，仅供教师用于传授使用 Microsoft 教师指导课件的培训课程。教师内容可能包括 Microsoft PowerPoint 演示文稿、教师备课指南、培训教师材料、Microsoft One Note 数据包、教室设置指南和预发行版课程反馈表。特此说明，教师内容不包括任何软件、虚拟硬盘或虚拟机。

2. 使用权利。许可内容并未出售，只是授予许可。许可内容按**每位用户一份**的方式授予许可，因此，您必须为访问或使用许可内容的每个个人获取许可。

2.1 以下是五组单独的使用权利。仅有一组适用于您。

a. **如果您是 Microsoft IT Academy 计划成员：**

- i. 代表您自己获取的每个许可只能用于以课件提供给您时的原有形式查看一 (1) 份 Microsoft 教师指导课件。如果 Microsoft 教师指导课件为数字格式，则您可在最多三 (3) 台个人设备上安装一 (1) 份课件。您不得在非您所有或不受您控制的设备上安装 Microsoft 教师指导课件。
- ii. 对于您代表最终用户或教师获取的每个许可，您可以：
 1. 将 Microsoft 教师指导课件的一 (1) 份打印件分发给一 (1) 个报名参加授权培训课程的最终用户，且只能在作为所提供 Microsoft 教师指导课件主题的授权培训课程即将开始之前分发；**或者**
 2. 为一 (1) 个最终用户提供唯一兑换码，以及关于如何访问一 (1) 个数字版本的 Microsoft 教师指导课件的说明；**或者**
 3. 为一 (1) 个教师提供唯一兑换码，以及关于如何访问一 (1) 个教师内容的说明；**前提是您遵守以下条款：**
- iii. 您只为已获得许可内容的有效许可的个人提供对许可内容的访问权限；
- iv. 您确保参加授权培训课程的每个最终用户都拥有作为授权培训课程主题的 Microsoft 教师指导课件的有效许可副本；
- v. 您确保在为每个最终用户提供 Microsoft 教师指导课件的打印件时，还将提供本协议的副本，且在为每个最终用户提供 Microsoft 教师指导课件之前，他们同意在使用 Microsoft 教师指导课件时将会遵守本协议条款。在访问 Microsoft 教师指导课件之前，所有个人必须通过依据当地法律可实施的方式来表示他们接受本协议；
- vi. 您确保传授授权培训课程的每个教师都拥有作为授权培训课程主题的教师内容的有效许可副本；
- vii. 您只能使用合格教师来传授所有授权培训课程，他们必须在作为这些课程的 Microsoft 教师指导课件主题的 Microsoft 技术领域内具有丰富知识和经验；
- viii. 您每周只能为称为 MOC 课程的每个授权培训课程提供最多 **15** 小时培训；
- ix. 您承认，并非 MCT 身份的教师无法访问 Microsoft 教师指导课件的所有教师资源。

b. **如果您是 Microsoft 学习能力成员：**

- i. 代表您自己获取的每个许可只能用于以课件提供给您时的原有形式查看一 (1) 份 Microsoft 教师指导课件。如果 Microsoft 教师指导课件为数字格式，则您可在最多三 (3) 台个人设备上安装一 (1) 份课件。您不得在非您所有或不受您控制的设备上安装 Microsoft 教师指导课件。
- ii. 对于您代表最终用户或 MCT 获取的每个许可，您可以：
 1. 将 Microsoft 教师指导课件的一 (1) 份打印件分发给一 (1) 个参加授权培训课程的最终用户，且只能在作为所提供 Microsoft 教师指导课件主题的授权培训课程即将开始之前分发；**或者**
 2. 为参加授权培训课程的一 (1) 个最终用户提供唯一兑换码，以及关于如何访问一 (1) 个数字版本的 Microsoft 教师指导课件的说明；**或者**
 3. 为一 (1) 个 MCT 提供唯一兑换码，以及关于如何访问一 (1) 个教师内容的说明；**前提是您遵守以下条款：**
- iii. 您只为已获得许可内容的有效许可的个人提供对许可内容的访问权限；

- iv. 您确保参加授权培训课程的每个最终用户都拥有作为授权培训课程主题的 Microsoft 教师指导课件的有效许可副本;
 - v. 您确保在为每个最终用户提供 Microsoft 教师指导课件的打印件时, 还将提供本协议的副本, 且在为每个最终用户提供 Microsoft 教师指导课件之前, 他们同意在使用 Microsoft 教师指导课件时将会遵守本协议条款。在访问 Microsoft 教师指导课件之前, 所有个人必须通过依据当地法律可实施的方式来表示他们接受本协议;
 - vi. 您确保传授授权培训课程的每个 MCT 都拥有作为授权培训课程主题的教师内容的有效许可副本;
 - vii. 您只能使用合格 MCT 来传授使用 MOC 的所有授权培训课程, 他们还必须持有作为这些课程的 MOC 资格主题的相应 Microsoft 认证证书;
 - viii. 您仅为最终用户提供对 Microsoft 教师指导课件的访问权限; 并且
 - ix. 您仅为 MCT 提供对教师内容的访问权限。
- c. **如果您是 MPN 成员:**
- i. 代表您自己获取的每个许可只能用于以课件提供给您时的原有形式查看一 (1) 份 Microsoft 教师指导课件。如果 Microsoft 教师指导课件为数字格式, 则您可在最多三 (3) 台个人设备上安装一 (1) 份课件。您不得在非您所有或不受您控制的设备上安装 Microsoft 教师指导课件。
 - ii. 对于您代表最终用户或教师获取的每个许可, 您可以:
 1. 将 Microsoft 教师指导课件的一 (1) 份打印件分发给一 (1) 个参加私人培训课程的最终用户, 且只能在作为所提供 Microsoft 教师指导课件主题的私人培训课程即将开始之前分发; 或者
 2. 为参加私人培训课程的一 (1) 个最终用户提供唯一兑换码, 以及关于如何访问一 (1) 个数字版本的 Microsoft 教师指导课件的说明; 或者
 3. 为参加私人培训课程的一 (1) 个教师提供唯一兑换码, 以及关于如何访问一 (1) 个教师内容的说明;
- 前提是您遵守以下条款:**
- iii. 您只为已获得许可内容的有效许可的个人提供对许可内容的访问权限;
 - iv. 您确保参加私人培训课程的每个最终用户都拥有作为私人培训课程主题的 Microsoft 教师指导课件的有效许可副本;
 - v. 您确保在为每个最终用户提供 Microsoft 教师指导课件的打印件时, 还将提供本协议的副本, 且在为每个最终用户提供 Microsoft 教师指导课件之前, 他们同意在使用 Microsoft 教师指导课件时将会遵守该本协议条款。在访问 Microsoft 教师指导课件之前, 所有个人必须通过依据当地法律可实施的方式来表示他们接受本协议;
 - vi. 您确保传授私人培训课程的每个教师都拥有作为私人培训课程主题的教师内容的有效许可副本;
 - vii. 您只能使用合格教师来传授所有私人培训课程, 他们必须持有作为这些课程的 Microsoft 教师指导课件主题的适用 Microsoft 认证证书;
 - viii. 您只能使用合格 MCT 来传授使用 MOC 的所有私人培训课程, 他们必须持有作为这些课程的 MOC 资格主题的相应 Microsoft 认证证书;
 - ix. 您仅为最终用户提供对 Microsoft 教师指导课件的访问权限; 并且
 - x. 您仅为教师提供对教师内容的访问权限。
- d. **如果您是最终用户:**
- 对于您获取的每个许可, 您只能将 Microsoft 教师指导课件用于个人培训用途。如果 Microsoft 教师指导课件为数字格式, 则您可使用培训提供商提供给您的唯一兑换码来在线访问 Microsoft 教师指导课件, 可在最多三 (3) 台个人设备上安装和使用一 (1) 份 Microsoft 教师指导课件。您还可以打印一 (1) 份 Microsoft 教师指导课件。您不得在非您所有或不受您控制的设备上安装 Microsoft 教师指导课件。
- e. **如果您是教师:**
- i. 对于您获取的每个许可, 您只能在一 (1) 台个人设备上, 以提供给您时的原有形式安装和使用一 (1) 个教师内容, 以准备或交付授权培训课程或私人培训课程, 您还可在另一台个人设备上再安装一 (1) 份副本作为备份, 该备份仅用于重新安装教师内容。您不得在非您所有或不受您控制的设备上安装或

使用教师内容。您还可以仅出于准备和交付授权培训课程或私人培训课程的目的，打印一 (1) 个教师内容。

- ii. 您可以根据最新版本的 MCT 协议，自定义与培训课程讲授存在逻辑关联的教师内容书面部分。如果您选择行使上述权利，即表示您同意遵守以下条款：(i) 自定义只能用于传授授权培训课程和私人培训课程之目的；并且 (ii) 所有自定义必须遵守本协议。请注意，使用“自定义”仅表示更改幻灯片和内容的顺序，和/或不使用全部幻灯片或内容，而不表示更改或修改任何幻灯片或内容。

2.2 组件的分离。许可内容作为一个整体授予许可，您不得分离它们的各个组件并将其安装到不同设备上。

2.3 许可内容的再分发。除非上述用户权利中另有明确规定，否则未经 Microsoft 明确的书面许可，您不得复制许可内容或其中的任何部分（包括任何许可的修改）或将其分发给任何第三方。

2.4 第三方声明。许可内容可能包含微软（而不是第三方）根据本协议的规定许可您使用的第三方代码。如果包含任何针对第三方代码的声明，仅用于提供信息。

2.5 附加条款。某些许可内容可能包含多个组件，以及有关其使用的附加条款、条件和证可。这些条件和许可中的无冲突条款也适用于您对相应组件的使用，作为对本协议中所述条款的补充。

3. 基于预发行技术的许可内容。如果许可内容的主题基于预发行版本的 Microsoft 技术（以下简称“**预发行版**”），则除本协议中的其他条款之外，还受以下条款约束：

- a. **预发行许可内容。**本许可内容主题是关于预发行版本的 Microsoft 技术的。预发行版本技术的工作方式可能与最终版本技术有所不同，我们有权更改最终版本所采用的技术。我们也有权不发行最终版本。基于最终版本技术的许可内容包含的信息可能与基于预发行版本的许可内容有所不同。Microsoft 没有义务为您提供任何其他内容，包括基于最终版本技术的任何许可内容。
- b. **反馈。**如果您同意向 Microsoft 提供有关许可内容的反馈，无论是直接提供还是通过微软指定的第三方提供，您都将无偿授予 Microsoft 出于任何目的以任何方式使用、共享和商业化您的反馈的权利。您同时向第三方无偿提供所有必需的专利权，以使他们的产品、技术和服务能够使用包含此类反馈的 Microsoft 技术、Microsoft 产品或服务的任何特定部分或与这些部分交互。如果根据某项许可的规定，Microsoft 由于在其技术或产品中包含了您的反馈而不得不向第三方授予该技术或产品的许可，则您不得提供这样的反馈。这些权利在本协议终止后继续有效。
- c. **预发行期限。**如果您是 Microsoft IT Academy 计划成员、Microsoft 学习能力成员、MPN 成员或教师，您将在以下日期停止使用基于预发行版技术的所有许可内容：(i) Microsoft 通知您终止使用基于预发行版技术的许可内容之日；或 (ii) 作为许可内容主题的技术商业发行日期之后六十 (60) 天，以两者中较早者为准（以下简称“**预发行期限**”）。预发行期限到期或终止后，您需要不可撤销地删除和销毁您拥有或控制的所有许可内容。

4. 许可范围。许可内容并未出售，只是授予许可。本协议只授予您使用该许可内容的某些权利。Microsoft 保留所有其他权利。除非适用的法律赋予您此项限制之外的权利，否则您只能在本协议明示允许的范围内使用该许可内容。在按规定使用该许可内容时，您必须遵守许可内容中的任何限制您以特定方式使用许可内容的技术限制。除非在本协议中明确允许，否则您不得：

- 访问许可内容，或允许尚未获得对许可内容的有效许可的任何个人访问许可内容；
- 不得更改、删除或遮盖许可内容包含的任何版权或其他保护声明（包括水印）、品牌或标识；
- 修改或创建任何许可内容的衍生作品；
- 公开展示许可内容，或将许可内容提供给他人进行访问或使用；
- 复制、打印、安装、出售、出版、传输、出借、改编、重用、链接或发布许可内容，或者将许可内容提供或分发给任何第三方；

- 绕过该许可内容中的任何技术限制；或者
 - 对该许可内容进行反向工程、反向编译、删除或以其他方式妨碍任意保护形式或者反汇编；尽管有此项限制，但如果适用的法律明示允许上述活动，则仅在适用的法律明示允许的范围内从事上述活动不在此限。
- 5. 权利和所有权的保留。**微软保留所有本协议中未明确授予您的权利。许可内容受版权法和其他知识产权法律和条约的保护。Microsoft 或其供应商对许可内容拥有所有权、版权和其他知识产权。
- 6. 出口限制。**许可内容受美国出口法律和法规管辖。您必须遵守适用于许可内容的所有国内和国际出口法律和法规。这些法律包括对目的地、最终用户和最终用途的各种限制。如需其他信息，请访问 www.microsoft.com/exporting。
- 7. 支持服务。**由于许可内容是按“现状”提供的，所以我们可能不会为其提供支持服务。
- 8. 终止。**如果您未能遵守本协议的条款和条件，Microsoft 可能会终止本协议，但任何其他权利不受影响。本协议出于任何原因终止后，您需要立即停止、删除和销毁您拥有或控制的所有许可内容。
- 9. 指向第三方网站的链接。**您可能会在使用许可内容的过程中链接到第三方网站。第三方网站不受微软控制，微软对任何第三方网站的内容、第三方网站中包含的任何链接以及第三方网站的任何更改或更新均不承担责任。微软不对源自任何第三方网站的网络广播或任何其他形式的信息传输负责。微软只是为了提供方便而向您提供这些指向第三方网站的链接，包含任何链接并不表示微软认可相应的第三方网站。
- 10. 完整协议。**本协议以及针对教师内容的所有附加条款、更新和补充共同构成了关于许可内容、更新和补充程序的完整协议。
- 11. 适用的法律。**
- 美国。如果您在美国购买该许可内容，则对本协议的解释以及由于违反本协议而引起的索赔均以华盛顿州法律为准并受其管辖，而不考虑冲突法原则。您所居住的州的法律管辖其他所有索赔项目，包括根据州消费者保护法、不正当竞争法以及侵权行为提出的相关索赔。
 - 美国以外。如果您在其他任何国家/地区购买该许可内容，则应遵守您所在国家/地区的法律。
- 12. 法律效力。**本协议规定了某些合法权利。根据您所在国家/地区的法律规定，您可能享有其他权利。您还可能享有与该许可内容卖方相关的权利。如果您所在国家/地区的法律不允许本协议改变您所在国家/地区法律赋予您的权利，则本协议将不改变您按照所在国家/地区的法律享有的权利。
- 13. 保证免责条款。**许可内容按“现状”授予许可，并且“目前可用”。使用该软件的风险需由您自行承担。**MICROSOFT** 及其相应关联公司不提供任何其他明示的保证、保障或条件。根据您当地的法律，您可能享有本协议无法改变的其他消费者权利。在您当地法律允许的范围内，**MICROSOFT** 及其相应关联公司排除任何默示保证或条件，包括有关适销性、针对特定目的的适用性和不侵权的默示保证或条件。
- 14. 损害赔偿和补偿责任的限制和排除。**您只能因直接损害从 **MICROSOFT** 及其相应关联公司和供应商处获得退款，退款金额上限为 **5.00** 美元。您不能因其他任何损害获得退款，包括后果性损害、利润损失、特别的损害、间接损害或附带性损害。

该限制适用于：

- 与第三方 Internet 站点上或第三方程序中的许可内容、服务、内容（包括代码）相关的任何情况；以及
- 在适用的法律允许的范围内，因违约、违反保证、保障或条件、严格责任、过失或其他侵权行为引起的索赔。

MCT USE ONLY. STUDENT USE PROHIBITED

即使微软知道或应该知道可能会出现损害，此项限制也同样适用。由于您所在国家/地区可能不允许排除或限制附带损害、后果性损害或其他损害的赔偿责任，因此上述限制或排除条款可能对您不适用。

2014 年 11 月修订

版本：MSL-Non-VM-2014-11-11

欢迎！

感谢你参加我们的课程！我们与 Microsoft 学习解决方案认证合作伙伴以及 Microsoft IT 学院 密切合作，为你带来世界级的学习体验—无论你是希望提高技艺的专业人员，还是准备踏入 IT 职场的普通学生，你都能参加这些课程并从中获益。

- Microsoft 认证培训师和教师—你的教师是始终符合认证要求的技术和教学专家。如果教师在我们的某家学习解决方案认证合作伙伴机构提供培训，他们也将接受学生 和 Microsoft 的全年考评。
- 认证考试的益处—培训结束后，请考虑参加 Microsoft 认证考试。“Microsoft 认证” 将验证你在 Microsoft 技术方面的技能，并帮助你在求职或职业发展中脱颖而出。实际上，IDC 的独立研究表明，75%的经理人相信认证对于团队绩效有重要作用¹。有关 Microsoft 认证考试升级以及考费折扣的信息，请向教师垂询。
- 客户满意保证—我们的学习解决方案认证合作伙伴提供满意保证，并且我们责成他们 履行承诺。课程结束后，请完成对当天上课体验的评估。我们无比重视您的反馈！

我们希望你获得良好的学习体验，并在职场中不断取得成功！

祝您一切顺利！

Microsoft Learning
www.microsoft.com/learning



¹IDC, Value of Certification: Team Certification and Organizational Performance, 2015 年 4 月

MICROSOFT LEARN ONLY STUDENT USE PROHIBITED

目录

第 1 章 Microsoft Azure 平台概述

第 1 课: Azure 服务	1-3
第 2 课: 管理门户	1-10
实验 A: 探索 Azure 预览门户	1-17
实验 B: 认识中国版 Windows Azure 管理门户	1-19

第 2 章 使用 Azure 虚拟机建立开发环境

第 1 课: 搭建 Azure 虚拟机	2-3
第 2 课: Azure 虚拟机工作负载	2-9
第 3 课: 迁移 Azure 虚拟机实例	2-13
实验 A: 创建用于开发和测试的 Azure 虚拟机	2-17
实验 B: 在中国版 Windows Azure 中创建用于开发和测试的虚拟机	2-21

第 3 章 在 Azure 平台上托管 Web 应用程序

第 1 课: Azure 网站服务	3-3
第 2 课: 在 Azure 中托管 Web 应用程序	3-6
第 3 课: 配置 Azure 网站	3-8
第 4 课: 发布 Azure 网站	3-13
第 5 课: 实验概述	3-16
实验 A: 使用 Azure 网站服务创建 ASP.NET 网站	3-19
实验 B: 在中文版 Windows Azure 中创建 ASP.NET 网站	3-23

第 4 章 在 Azure 中存储 SQL 数据

第 1 课: Azure SQL 数据库概述	4-3
第 2 课: 管理 Azure 中的 SQL 数据库	4-7
第 3 课: Azure SQL 数据库工具	4-10
第 4 课: 保护和恢复 Azure SQL 数据库实例	4-12
实验 A: 在 Azure SQL 数据库中存储事件数据	4-14
实验 B: 使用中国版 Windows Azure 在 Azure SQL 数据库中存储事件数据	4-18

第 5 章 设计有复原力的云应用程序

第 1 课: 高可用应用程序设计实践	5-3
第 2 课: 应用程序分析	5-5

MCT USE ONLY. STUDENT USE PROHIBITED

第 3 课：使用 ASP.NET 构建高性能应用程序	5-7
第 4 课：常见云应用程序模式	5-10
第 5 课：缓存应用程序数据	5-12

第 6 章 管理 Azure 中的云服务

第 1 课：云服务概述	6-3
第 2 课：云服务 Web 角色	6-5
第 3 课：云服务辅助角色	6-6
第 4 课：云服务角色处理	6-9
第 5 课：自定义云服务	6-13
第 6 课：更新和管理云服务部署	6-17
实验 A：使用 Azure 辅助角色创建后台进程	6-19
实验 B：使用 Azure 辅助角色创建后台进程	6-23

第 7 章 在 Azure 中存储表格式数据

第 1 课：Azure 存储概述	7-3
第 2 课：Azure 存储表概述	7-8
第 3 课：表实体事务	7-16
实验 A：在 Azure 存储表中存储活动注册数据	7-19
实验 B：在 Azure 存储表中存储活动注册数据	7-24

第 8 章 存储和使用 Azure 存储中的文件

第 1 课：存储 Blob	8-3
第 2 课：控制对存储 Blob 和容器的访问	8-6
第 3 课：配置 Azure 存储帐户	8-10
第 4 课：Azure 文件	8-12
实验 A：在 Azure 存储 Blob 中存储生成的文档	8-14
实验 B：在中国版 Windows Azure 的存储 Blobs 中存储生成的文档	8-20

第 9 章 使用队列和 Service Bus 设计通信策略

第 1 课：Azure 存储队列	9-3
第 2 课：Azure Service Bus	9-6
第 3 课：Azure Service Bus 队列	9-8
第 4 课：Azure Service Bus 中继	9-12
第 5 课：Azure Service Bus 通知中心	9-15
实验 A：使用队列和 Service Bus 来管理 Azure 中 Web 应用程序之间的通信	9-21

实验 B: 在中国版 Windows Azure 中使用队列和 Service Bus 来管理 Web 应用程序间的通信	9-30
--	------

第 10 章 管理 Azure 中的基础结构

第 1 课: Azure 虚拟网络	10-3
第 2 课: 高度可用的 Azure 虚拟机	10-5
第 3 课: 虚拟机配置管理	10-9
第 4 课: 自定义 Azure 虚拟机网络	10-12
实验 A: 管理虚拟网络中的多个虚拟机	10-15
实验 B: 使用中国版 Windows Azure 管理虚拟网络中的多个虚拟机	10-20

第 11 章 自动化与 Azure 资源的集成

第 1 课: Azure SDK 客户端库	11-3
第 2 课: 使用 Windows PowerShell 编写 Azure 服务管理脚本	11-7
第 3 课: Azure REST 接口	11-14
第 4 课: Azure 资源管理器	11-16
实验 A: 使用 PowerShell 自动化测试环境的创建工作	11-21
实验 B: 使用 PowerShell 自动化测试环境的创建工作	11-26

第 12 章 保护 Azure Web 应用程序

第 1 课: Azure Active Directory	12-3
第 2 课: Azure AD 目录	12-5
第 3 课: Azure AD 多重身份验证	12-9
实验 A: 将 Azure Active Directory 与事件管理门户集成	12-11

第 13 章 维护和监视 Azure 中的 Web 解决方案

第 1 课: Web 应用程序的部署策略	13-3
第 2 课: 部署 Azure 网站服务	13-6
第 3 课: 部署 Azure 云服务	13-11
第 4 课: 持续集成	13-14
第 5 课: 监视云应用程序	13-16
实验 A: 将 Events Web 应用程序部署到 Azure	13-23
实验 B: 使用中国版 Windows Azure 部署 Events Web Application 到云环境	13-28

实验

第 1 章: 实验 A: 探索 Azure 预览门户	L1-1
实验 B: 认识中国版 Windows Azure 管理门户	L1-3

第 2 章: 实验 A: 创建用于开发和测试的 Azure 虚拟机	L2-1
实验 B: 在中国版 Windows Azure 中创建用于开发和测试的虚拟机	L2-10
第 3 章: 实验 A: 使用 Azure 网站服务创建 ASP.NET 网站	L3-1
实验 B: 在中文版 Windows Azure 中创建 ASP.NET 网站	L3-7
第 4 章: 实验 A: 在 Azure SQL 数据库中存储事件数据	L4-1
实验 B: 使用中国版 Windows Azure 在 Azure SQL 数据库中存储事件 数据	L4-8
第 6 章: 实验 A: 使用 Azure 辅助角色创建后台进程	L6-1
实验 B: 使用 Azure 辅助角色创建后台进程	L6-7
第 7 章: 实验 A: 在 Azure 存储表中存储活动注册数据	L7-1
实验 B: 在 Azure 存储表中存储活动注册数据	L7-8
第 8 章: 实验 A: 在 Azure 存储 Blob 中存储生成的文档	L8-1
实验 B: 在中国版 Windows Azure 的存储 Blobs 中存储生成的文档	L8-10
第 9 章: 实验 A: 使用队列和 Service Bus 来管理 Azure 中 Web 应用程序之间 的通信	L9-1
实验 B: 在中国版 Windows Azure 中使用队列和 Service Bus 来管理 Web 应用程序间的通信	L9-15
第 10 章: 实验 A: 管理虚拟网络中的多个虚拟机	L10-1
实验 B: 使用中国版 Windows Azure 管理虚拟网络中的多个虚拟机	L10-9
第 11 章: 实验 A: 使用 PowerShell 自动化测试环境的创建工作	L11-1
实验 B: 使用 PowerShell 自动化测试环境的创建工作	L11-5
第 12 章: 实验 A: 将 Azure Active Directory 与事件管理门户集成	L12-1
第 13 章: 实验 A: 将 Events Web 应用程序部署到 Azure	L13-1
实验 B: 使用中国版 Windows Azure 部署 Events Web Application 到云环境	L13-9

关于本课程

本节提供本课程的简短描述、面向的对象、建议的必备条件以及课程目标。

课程描述

本课程面向的学生需要有构建纵向缩放应用程序的经验。学生还将获得使用 Microsoft Azure 平台的经验，并对提供的服务有基本了解。

本课程让学生有机会使用现有的 ASP.NET MVC 应用程序，并在将其迁移到 Azure 的过程中扩展其功能。本课程侧重于在云中构建高度可用的解决方案时，必须考虑的注意事项。本课程还将帮助学生准备《70-532：开发 Microsoft Azure 解决方案》认证考试。注意，考试内容针对的是国际版 Azure 环境，并非中国版 Azure 环境。国际版和中国版 Azure 的区别将在本书中有进一步的介绍。

对象

本次培训面向的对象在实施和监视 Microsoft Azure 解决方案方面有基本的经验。本课程的学生还应该精通用来构建应用程序解决方案的开发工具、技术和方法。

课程必备条件

除了专业经验外，学生还必须具备使用 Azure 平台的经验。他们还将对实验场景的 C# 概念有基本的了解。学生经验可包括：

- 比较 Azure 平台中提供的服务
- 配置和部署 ASP.NET Web 应用程序
- 从库创建 Azure 网站服务
- 部署和监视 Azure 网站服务
- 创建和配置 Azure 虚拟机
- 描述云服务和虚拟机之间的关系
- 部署现有云服务包
- 创建和管理存储帐户
- 管理存储帐户中的 Blob 和容器
- 创建、配置和连接到 SQL 数据库实例
- 认识导入 SQL 单机版数据库的影响
- 管理 Azure Active Directory 实例中的用户、组和订阅
- 创建虚拟网络
- 实现点到站点网络

课程目标

完成本课程后，学生将能够：

- 比较 Azure 平台中提供的服务。
- 配置和部署 Web 应用程序。

- 从库创建 Azure 网站服务。
- 部署和监视 Azure 网站服务。
- 创建和配置 Azure 虚拟机。
- 描述云服务和虚拟机之间的关系。
- 部署现有云服务包。
- 创建和管理存储帐户。
- 管理存储帐户中的 Blob 和容器。
- 创建、配置和连接到 SQL 数据库实例。
- 认识导入 SQL 单机版数据库的影响。
- 管理 Azure Active Directory 实例中的用户、组和订阅。
- 创建虚拟网络。
- 实现点到站点网络。

课程大纲

课程大纲如下：

- 第 1 章 “Microsoft Azure 平台概述”**
- 第 2 章 “使用 Azure 虚拟机建立开发环境”**
- 第 3 章 “在 Azure 平台上托管 Web 应用程序”**
- 第 4 章 “在 Azure 中存储 SQL 数据”**
- 第 5 章 “设计有复原力的云应用程序”**
- 第 6 章 “管理 Azure 中的云服务”**
- 第 7 章 “在 Azure 中存储表格式数据”**
- 第 8 章 “存储和使用 Azure 存储中的文件”**
- 第 9 章 “使用队列和 Service Bus 设计通信策略”**
- 第 10 章 “管理 Azure 中的基础结构”**
- 第 11 章 “自动化与 Azure 资源的集成”**
- 第 12 章 “保护 Azure Web 应用程序”**
- 第 13 章 “维护和监视 Azure 中的 Web 解决方案”**

课程材料

学习套件中包含以下资料：

- **教材：**简明扼要的课堂学习指南，以简洁、紧扣重点的形式提供了关键技术信息，这对有效的课堂学习体验来说至关重要。
 - **小节：**引导您达成学习目标，并提供对于成功的课堂学习体验至关重要的要点。
 - **实验：**提供面向实战的上机平台，供您练习在课程中学到的知识和技能。
 - **本章复习与提高：**提供实用参考资料，促进学生掌握知识和技能。

- **实验答案要点：**提供分步骤的实验解答指南。



课程配套内容：可搜索、易于浏览的数字内容，其中包含可作为教材的补充资料的优质联机资源。

- **章：**包含每节的相应内容，如问题与解答、详细演示步骤，以及补充阅读的链接。此外，各章还包含“实验复习题与解答”以及“本章复习与提高”部分。这些部分中包含复习题和解答、最佳做法、常见问题和故障排除提示（附答案）以及实战问题和场景（附答案）。
- **资源：**包含合理分类的补充资源，便于学生立即访问 TechNet、MSDN® 和 Microsoft® Press® 上提供的最新补充内容。



学生课程文件：包含自解压可执行文件 Allfiles.exe，该文件包含实验和演示所需要的所有文件。

注：课程本版本的学生 Allfiles 文件可在 <http://www.microsoft.com/learning/companionmoc/> 网站上找到

- **课程评估：**课程结束后，你可以完成在线评估，以提出你对课程、培训设施和教师的反馈意见。
- 如需提供有关本课程的更多意见或反馈，请发送电子邮件至 support@mscourseware.com。如需询问有关 Microsoft 认证计划的信息，请发送电子邮件至 mcphelp@microsoft.com。

虚拟机环境

本节提供有关如何建立教室环境来支持课程业务场景的信息。

虚拟机配置

本课程未附带虚拟机。实验步骤将直接在 Azure 中执行。学生需要 Azure 帐户才能完成实验。

课程文件

本课程的实验相关文件可在 <http://www.microsoft.com/learning/companionmoc/> 网站上找到

课堂设置

本课程要求在教室中进行，教师应至少有一台计算机，学生人手一台。本课程要求教师和学生计算机都能接入 Internet。学生需要 Azure 帐户才能完成实验。

课程硬件级别

为了确保学员有令人满意的学习体验，Microsoft Learning 要求“Microsoft 学习解决方案认证合作伙伴”(CPLS)，在其用于教授“Microsoft 官方学习产品”课件的所有教室中，提供达到或高于最低设备配置要求的教师和学员计算机。

硬件级别 6

- 处理器：64 位 Intel Virtualization Technology (Intel VT) 或 AMD Virtualization (AMD-V) 处理器（建议 2.8 Ghz 双核或更高配置）
- 硬盘：双 120GB 硬盘，7200 RPM SATA 或更高（带区）
- 内存：4GB 内存，扩展到 8GB 或更高
- DVD/CD：DVD；推荐双层。
- 网络适配器
- 带有放大扬声器的声卡
- 显示器：Super VGA 显示器（17 吋/43 厘米）

此外，教师计算机必须连接到支持 SVGA 1024 x 768 像素/16 位色的投影显示设备。

第 1 章 Microsoft Azure 平台概述

目录：

章节概述	1-2
第 1 课：Azure 服务	1-3
第 2 课：管理门户	1-10
实验 A：探索 Azure 预览门户	1-17
实验 B：认识中国版 Windows Azure 管理门户	1-19
章节复习和作业	1-21

章节概述

Microsoft Azure 提供了一揽子服务，可以将这些服务用作云应用程序的基础构件。第 1 节“Azure 服务”概括过去在使用 Microsoft Azure 平台时可能用到的服务。第 2 节“管理门户”描述可用于管理 Azure 订阅和服务的两个当前门户。第 3 节“实验概述”演示在整个课程中您会用到的实验应用程序。



注释：预览门户在中国版 Windows Azure 中暂时不支持。

目标

完成本章后，您可以做到：

- 描述某些常见 Azure 服务。
- 描述 Microsoft Azure 管理门户与 Azure 预览门户之间的区别。

第 1 课 Azure 服务

本节描述 Azure 中提供的某些常见服务和功能，很多入门级 Azure 开发人员和 IT 专业人员都使用这些服务和功能。此功能列表虽然并不全面，但是它已经涉及了很多您已经使用过的技术和功能。

课程目标

完成本节后，您可以做到：

- 描述以下 Azure 服务：

- 网站
- 虚拟机
- 云服务
- 存储
- SQL 数据库
- 虚拟网络

服务概述

在企业中存在着各种 IT 系统，这些系统分别涵盖了一个业务或者生产领域。比如，邮件系统负责了企业的内外通信职能。这些系统可能运行在一台服务器上，也可能运行在多台服务器上。从 IT 系统管理的角度上来看，会把用户和应用程序对这些 IT 系统的使用视为工作负载。

Azure 平台是一系列服务的集合，这些服务可以用来托管现有工作负载、或者使用托管服务代替工作负载、或者创建新的工作负载。当然，在大部分项目中，您可能只会用到 Azure 平台中的部分现成服务。在您的同行中，您会发现大家或多或少都有这方面的经验。本课程中，您将了解运行在 Azure 平台上的新开发项目可以使用哪些最常见的服务。

在开发项目或工作负载中，可以充分利用这些现成的服务来满足需求。以下是常见的 Azure 服务：

物理基础结构替代方案

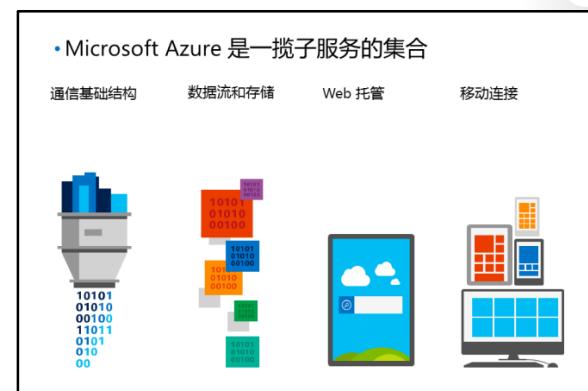
虚拟机和虚拟网络之类的服务有助于模拟数据中心中的现有基础结构。在需要将现有虚拟机从企业本地数据中心迁移到 Azure 的整体迁移(*lift and shift*)场景中，可以使用这些服务。若您需要最大程度控制操作系统环境以及云应用程序的配置设置，那么在这类场景中可以使用虚拟机和虚拟网络。

通信基础结构

云应用程序本质上通常是模块化的，因此需要托管通信基础结构（如 Service Bus）来连接各种应用程序模块。服务总线提供了事件订阅和发布、移动通知基础结构、WCF 服务中继和托管队列服务等功能。

数据流和存储

托管存储服务可应对各种文件保存和操作需求。Blob 和 Azure 文件服务帮助您使用客户端库和 SMB 2.1 协议轻松保存和修改文件。



Web 托管

对于低冲突开发（low-friction development）和部署，网站服务为集成 FTP 协议、源代码管理提供程序（TFS、CodePlex、GitHub 等）或 Web 部署协议的应用程序提供了 Web 托管平台。对于具体的云应用程序场景，可以使用云服务来托管 Web 应用程序和后台任务。

移动连接

移动服务是面向客户端设备的全托管后端服务解决方案。移动服务与 SQL 数据库相结合，可帮助您设计数据架构同时兼顾向上扩展，以满足用户的需求。

中国版 Windows Azure

在中国，Windows Azure 由世纪互联运营，世纪互联运营为中国用户提供 Windows Azure 服务。

中国版 Windows Azure，内部代号 Mooncake，是一个独立运营的实例。中国版 Windows Azure 与国际版 Microsoft Azure 的不同主要表现在：中国版 Windows Azure 提供的服务、特性方面与微软在全球运营的 Microsoft Azure 平台存在部分差异；同样的服务功能在两个环境中的操作步骤可能略有不同，如全球的 Microsoft Azure 平台使用 Microsoft 账户进行登录，中国版 Windows Azure 需要使用 OrgID 进行登录；相关资源的连接地址和方式略有不同（如管理门户地址，网站默认 URL 等）。本书在各章内容会指出中国版 Windows Azure 与国际版 Microsoft Azure 的不同之处。

您将会在本书实验中了解中国版 Windows Azure 的操作方法。



参考资料链接：中国版 Windows Azure 开发者注意事项。

<https://msdn.microsoft.com/zh-cn/library/azure/dn578439.aspx>

网站

网站服务为 Web 应用程序提供了平台，平台具备多种功能来简化部署。平台与 Internet Information Server (IIS) 网站高度兼容，能让您使用熟悉的 IIS 配置设置来配置 Web 应用程序。由于平台高度托管，因此可以利用负载平衡自动扩展 Web 应用程序，负载平衡也是自动配置的。



Azure 网站

<http://go.microsoft.com/fwlink/?LinkId=525506>

支持的框架

网站服务允许不经修改就直接将 ASP.NET、Java、PHP、Node.js 或 Python 用作 Web 应用程序的应用程序框架。此外，还可使用支持自定义模板和标记语言的常见 CMS 解决方案，如 Drupal、Joomla、DotNetNuke 和 Umbraco。

- 中国版 Windows Azure 世纪互联运营
- 中国版 Windows Azure 与国际版 Microsoft Azure 区别
 - 服务和功能存在部分差异
 - 服务操作步骤略有不同
 - 相关资源连接地址和方式不同
- 预览门户在中国版 Windows Azure 中暂时不支持

Azure 网站是一个平台即服务方案，可让您快速轻松地部署并扩展 Web 应用程序。

功能：

- 从库中创建网站实例
- 配置链接的资源
- 创建并使用宿主计划
- 在免费、共享、基本和标准模式之间切换
- 从源代码管理提供程序部署

部署

Microsoft Visual Studio 之类的集成开发环境 (IDE) 可使用 FTP 或 Web 部署协议将 Web 应用程序部署到网站。Web 部署提供了一种封装的打包格式，这种格式可用来在不同环境之间迁移 IIS Web 应用程序或网站，或者从开发 IDE 迁移到 IIS 实例。当应用程序安装时，会用到连接字符串或应用程序设置。应用程序开发人员可以使用这种格式规范来指定连接字符串或应用程序设置。

IIS Web 部署

<http://go.microsoft.com/fwlink/?LinkId=525366>

与当前同步部署方法一起，网站还支持从源代码管理提供程序连续部署。可以使用 GitHub、Bitbucket、Codeplex 或 Microsoft Azure Visual Studio Online 等源代码管理提供程序作为连续部署版本的来源。

无操作

网站环境完全托管，这样您就有了一个可伸缩的 Web 托管平台，该平台有复原力且高度可用，无需在低层次上专门管理每个网站实例。只要配置一组值，便可向上扩展 Web 应用程序。

虚拟机

虚拟机根据应用程序工作负载按需提供计算，并且与现有虚拟化工作负载高度兼容。使用 Hyper-V 磁盘中的标准化 .vhdx 格式，基础结构管理员可轻松将现有 Hyper-V 工作负载提升到 Azure。

工作负载

虚拟机支持一系列常见的 Microsoft 应用程序工作负载。使用托管在 Azure 中的基础结构，可以完全支持 Microsoft SharePoint、Microsoft SQL Server、Dynamics GP 和 BizTalk Server 等软件平台。将虚拟机连接到企业内部网络，就可以启用灾难恢复和高可用性解决方案，如 SQL Server AlwaysOn。

模板

使用门户上预建的模板，可以在 Azure 中创建绝大部分受支持的工作负载。Microsoft 和第三方映像可用于支持 Linux 和 Microsoft 的应用程序工作负载。Windows 和 Linux 的标准操作系统映像还可用于新项目。还有 VM Depot 可用于社区提供的 Linux 映像。

Microsoft Open Technologies VMDepot

<http://go.microsoft.com/fwlink/?LinkId=525339>

大小选择

可使用各种 CPU、内存和 IOPS 选项配置虚拟机，这些选项通常称为 VM 大小 (Size)。VM 大小即虚拟机的配置规模。大多数最常见的应用程序工作负载有基本大小和标准大小。较大的标准大小还可用于内存密集型工作负载。D 大小的虚拟机有更快的处理器以及本地固态驱动器 (SSD)。G 大小的虚拟机可用于需要海量计算资源和内存资源的工作负载。

基础结构即服务方案，可让您在几分钟之内部署要用于承载 Windows 或 Linux 工作负载的计算实例。

功能：

- 使用由产品团队构建的映像来部署 SQL Server、SharePoint 和 Apache 等工作负载
- 为虚拟机附加、格式化和配置多个磁盘
- 远程连接到 Windows 或 Linux 虚拟机
- 在多种 VM 大小之间选择 (A0-A9)
- 选择基本层或标准层 VM

云服务

云服务为托管在 Azure 中的虚拟机以及自定义应用程序的托管平台提供封装。云服务服务器作为虚拟机或者 Web 角色（Web Role）及辅助角色（Worker Role）的网络边界、安全边界以及软件负载平衡器。

无状态应用程序

可以使用 Web 角色和辅助角色来托管无状态应用程序模块。

配合诸如下面的基础结构使用云服务，可以轻松设计 N 层应用程序：

云服务是一个平台即服务方案，可让您专注于应用程序编码，而 Azure 平台负责扩展应用程序以实现高可用性

功能：

- 将虚拟机与云服务结合
- 扩展实例并配置负载平衡
- 部署已有云服务包

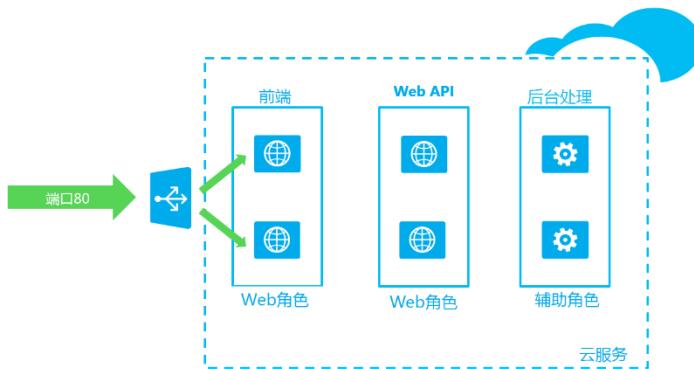


图 1.1：典型的云服务 N 层应用程序

自动处理

云服务提供的平台自动为您处理部署、操作系统更新和缩放。该平台可配置且可扩展，可以使用启动脚本之类的功能控制上述大多数的操作。

后台处理

云服务提供了辅助角色选项，可用来托管长期运行的进程的多个实例。通常使用辅助角色并配合队列机制，来处理大批量请求。

存储

存储是一种托管服务，可以利用此服务存储可在整个云应用程序中使用的数据。存储的数据可包括松散结构化实体、队列消息以及各种文件。可以使用客户端库、URL 或 REST API 来访问这些数据。可以使用文件共享来存储和管理文件。

Blob

Blob 是托管文件，可使用 URL、REST API 或客户端库来持久保存和访问 Blob。可以使用容器来逻辑分组 Blob。Blob 会在各种云服务中用到，例如虚拟机会使用 Blob 来存储虚拟硬盘。

队列

队列服务提供简单的托管接口来将消息推送到队列，并消费这些消息。消息存储为序列化字符串。可以使用逻辑队列操作（如 enqueue、dequeue 和 peek），按照先进先出的方式存储或检索这些消息。

表

表服务是一种 NoSQL 存储，可用来存储松散结构化的实体集，这些实体集可以整体保存并以非常高的效率检索。通过实现常见的分片概念（如分区和哈希索引），可以使用表来为应用程序存储大量数据。

文件

Azure 文件服务提供 SMB 2.1 协议的文件共享，在将现有应用程序工作负载从企业内部迁移到 Azure 时，可以使用这种文件共享。使用此技术保存的文件还可使用 REST API 或客户端库来访问。

SQL 数据库

SQL 数据库是一种数据库即服务托管平台，可以用来托管 SQL 对象。它与现有数据工具高度兼容，因此可为您提供与 SQL Server 单机版相似的管理体验。

兼容性

很多 SQL 功能和对象可与 SQL 数据库一起使用。SQL Server 2014 提供的工具可用来分析数据库，并将数据库从企业内部服务器迁移到 Azure。对于早期版本的 SQL，SQL Azure Migration Wizard 和 Azure Websites Migration Assistant 之类的工具可用于评估现有数据库并执行实际迁移。

可靠、可伸缩的存储服务，可存储所有类型、所有大小的数据。

功能：

- 选择用于存储的数据中心
- 配置异地复制选项
- 管理 Blob 和文件
- 保护容器
- 上传文件
- 访问文件

SQL 数据库是一种数据库即服务方案，可使 SQL 数据库供所有云开发人员使用。

功能：

- 创建逻辑 SQL Server 或 SQL 数据库实例
- 配置 SQL Server 实例防火墙
- 比较 SQL 数据库服务和 Azure 虚拟机中的单机版 SQL Server
- 使用 SQL Server 数据工具、Azure SQL 数据库管理门户，以及 SQL Server Management Studio 来连接数据库实例

SQL Azure Migration Wizard

<http://go.microsoft.com/fwlink/?LinkID=525380>

数据工具

可以将很多现有数据工具与 SQL 数据库一起使用，如适用于 Microsoft Visual Studio 的 SQL Server Data Tools 和 SQL Server Management Studio。其他工具和应用程序框架可使用连接字符串来连接托管在 Azure 中的数据库。可以在任一门户上直接提供 SQL 数据库实例的连接字符串。SQL 数据库还提供了用于管理数据库的自定义门户。

可伸缩性

SQL 数据库实例可使用 **Elastic Scale** 来为具有动态或大型数据库工作负载的应用程序自动创建和管理确定的分区。从大型数据存储到多租户的软件即服务(SaaS)服务，各种应用程序都能受益于此功能。

虚拟网络

虚拟网络是一种安全边界，创建这样的边界是为了使服务实例可以相互私密地通信。虚拟网络支持网站、云服务和虚拟机服务的实例。通过各种连接选项，虚拟网络可相互连接，或者连接到现有的企业内部计算机或网络。

站点连接

可以使用 Internet 协议安全(IPSec)协议将虚拟网络相互连接，或者连接到现有的企业内部网络。这样就能在多种网络之间以及各种混合云场景中建立安全链路。要确定哪些 VPN 设备支持 IPSec，可以查阅受支持的 VPN 设备列表。或者，直接查看用于站点-站点连接设备的功能列表，确定所需要支持的功能是否具备。

关于用于虚拟网络的 VPN 设备

<http://go.microsoft.com/fwlink/?LinkId=525508>

可用于在云中或企业内部分组服务和计算实例的专用网络。

功能：

- 创建指定区域或地缘组的虚拟网络 (VNET)
- 将 VNET 配置为使用 DNS 服务器
- 配置 VNET 子网
- 实现与 VNET 的点到站点连接
- 在现有 VNET 中创建虚拟机

直接连接

安全套接字隧道协议 (SSTP) 也可以用来让单独的设备或计算机使用点到站点虚拟专用网 (VPN) 连接，直接连接到虚拟网络。在企业内部位置没有支持 IPSec 的硬件，或者网络需要由多名远程工作者使用的情况下，这个功能很有用。

域名服务 (DNS)

Azure 自动为虚拟网络中的虚拟机和服务提供 DNS 服务。您也可以选择提供自己 DNS 服务器的 IP 地址。DNS 服务器可以托管在 Azure 中，也可以使用上面提及的某一种连接来连接到 DNS 服务器。

应用程序服务

Azure 为云应用程序提供了多种其他服务。下面几节描述其中的部分服务。

Azure Active Directory (Azure AD)

Azure AD 是托管的身份认证和访问服务。可以使用此服务来为云应用程序或任何自定义应用程序提供身份验证和单一登录功能。也可以视情况将此服务与现有的 Active Directory 域控制器同步。

媒体服务

媒体服务是可以针对各种客户和设备编码并流式传送多媒体的一种服务。媒体服务可以动态扩展，以适应音频或视频转换及检索的需求量高峰。可以配合媒体服务使用作业来监视队列中编码操作的进度。

Azure 提供了一揽子可集成在新应用程序或现有应用程序中的服务，这些服务可增强应用程序的功能

示例：

- Azure Active Directory
- 媒体服务
- 移动服务
- 自动化

MCT USE ONLY. STUDENT USE PROHIBITED

移动服务

移动服务是一种后端服务平台，可用来为移动设备存储和提供应用程序数据。移动服务使用 SQL 数据库来存储实际数据，并管理发布到后端 Web 服务的数据的架构。

自动化

自动化允许使用 Windows PowerShell 来自动化常见的管理任务，因而扩展了管理功能。自动化采用与 Microsoft System Center 同样的方式来使用 Windows PowerShell 工作流。自动化还包括一个全面的 Runbook 脚本库，这些脚本都是由 Microsoft 和开源社区提供的。

第 2 课 管理门户

Azure 服务以及用来管理这些服务的 Web 应用程序目前有多个版本。

本节描述管理门户的两个最新版本，它们都可用来配置 Azure 服务的实例。本节还将演示，当其中一个 Azure 管理门户缺乏您需要的功能时，如何切换到另一个 Azure 管理门户。

课程目标

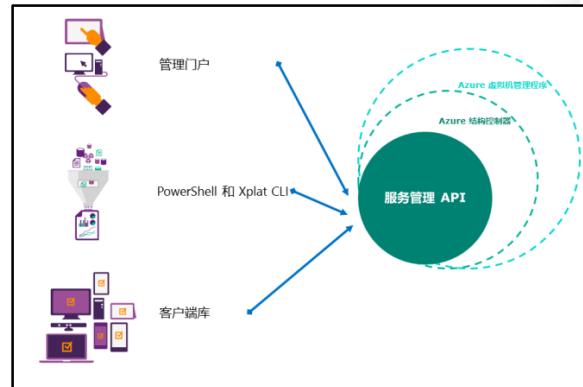
完成本节后，您可以做到：

- 描述当前管理门户。
- 描述预览门户（Ibiza）。
- 在两个不同门户之间切换。

管理门户

Azure 平台提供了一整套服务，这些服务可用来在云中托管应用程序和工作负载。服务管理 REST API 是一种 Web 服务，它接收创建、更改或配置服务的请求，并将这些请求传给 Microsoft Azure 结构控制器（Azure Fabric Controller）。结构控制器根据这些请求作出决策，并根据需要，利用 Azure 虚拟机管理程序（Azure Hypervisor）在数据中心中创建新的虚拟机。

管理门户是众多可用来与 Azure 平台交互的不同界面之一。所有界面共用服务管理 REST API，为管理 Azure 平台中的服务而创建的任何自定义应用程序都可使用此 API。



目前管理门户有两个主要版本。第一个版本是本课程中称为管理门户的当前门户。此门户使用服务显示在左侧的界面，服务实例页面使用选项卡来组织数据。最新的门户在本课程中称为预览门户。此门户使用一个水平边栏选项卡列表来组织数据，并提供逐层深入体验。

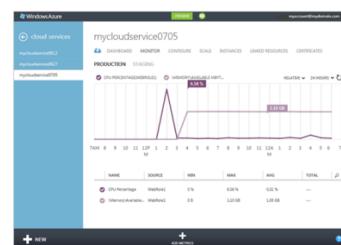
管理门户

管理门户是 TechEd 2012 上发布的。它取代了 Microsoft Silverlight 门户，并使用 HTML5 和 JavaScript 来形成丰富、动态且快速的用户体验。在管理门户中，服务列在门户左侧的导航栏中。选择某个服务后，该服务的登陆页将显示，其中包含服务实例的列表。服务实例本身可包含多个选项卡，这些选项卡显示配置选项、指标监视和概览仪表板等数据。

服务搭建

在管理门户中，在屏幕底部有一个黑色条形，可用来创建新的服务实例。如果直接单击“+新建”按钮，而不选择蓝色导航栏中的特定服务，所有可用服务的列表将显示。可以从此列表中选择任何服务来创建该服务的实例。如果先选择导航栏中的服务，然后单击“+新建”，这时会出现一个对话框，其中创建该服务新实例的选项已自动选中。然后，可以使用

管理门户可用来立即设置服务、基础结构或应用的实例。



MCT USE ONLY. STUDENT USE PROHIBITED

“快速创建”选项，只要指定必需的参数即可创建服务实例。除此以外，也可以使用“自定义创建”选项，利用向导来指定所有服务参数。

“+新建”对话框可以快速创建新的服务实例。



图 1.2：“+新建”对话框

选项卡

选择一个服务实例后，会显示一个选项卡列表。可以使用这些选项卡来查看数据以及配置服务选项。第一个选项卡通常是快速启动选项卡，其中显示了相应教程的超链接以及开始使用服务的建议。还有一个仪表板选项卡，其中显示了该服务的当前状态。其他选项卡用来分类可用于该服务实例的配置选项。

服务实例按类别包含可查看数据的选项卡。



图 1.3：服务实例选项卡

帐户门户

帐户门户可用来查看有关当前订阅的信息。在此门户中，可以管理您的订阅，即修改订阅名称、服务管理员和计费数据。其中的很多功能已经包含在预览门户中。

参考资料链接：<http://account.windowsazure.com>

预览门户(Ibiza)

最新版的门户（也称为预览门户）是在 Build 2014 开发者大会上发布的。此门户的重点是显示有关每个服务实例的更多元数据，并合理分组服务以便监视和计费。新门户的其他功能每周发布，节奏很快。

边栏选项卡

在查看服务实例的设置或执行操作时，详细信息会显示在称为边栏选项卡的垂直对话框中。可以水平叠放这些边栏选项卡，以便查看服务（或操作）的详细信息以及某些后续操作，而无需水平滚动窗口。单击右上角的关闭(X)按钮可以关闭边栏选项卡。单击右上角的固定图标可以将边栏选项卡固定在启动板以便将来访问。可以最小化边栏选项卡，将其放

在屏幕的左侧，也可以将其最大化，使其填满屏幕的整个宽度。可以隐藏顶部命令栏中的标签，为磁贴留出更多屏幕空间。最后，可以在大多数边栏选项卡中自定义磁贴的顺序和布局。

包含子边栏选项卡的网站实例边栏选项卡，子边栏选项卡显示每个浏览器的会话。



图 1.4：显示指标边栏选项卡的网站边栏选项卡

导览

导览是用来创建或修改服务的一系列连续边栏选项卡。例如，为了创建网站，可以转到“库”边栏选项卡，然后查看“网站库选项”边栏选项卡。在选择某个特定网站模板后，描述该模板的边栏选项卡将显示。如果单击“创建”，另一个边栏选项卡将显示新网站的基本选项。可以进一步深入并指定高级选项。很多高级选项可能显示后续边栏选项卡。这些边栏选项卡会水平排列组合在一起，成为导览。用户可以在导览中向左或向右选择边栏选项卡。在导览中，如果尝试关闭某个边栏选项卡（位于另一个边栏选项卡的左侧），而又未保存更改，那么会显示一条消息通知您将丢失更改的设置。导览有时候细分成导览子块，但在当前用户界面中未显示。

下图描绘了在资源组中创建网站和 SQL 数据库资源时所用导览的开头。

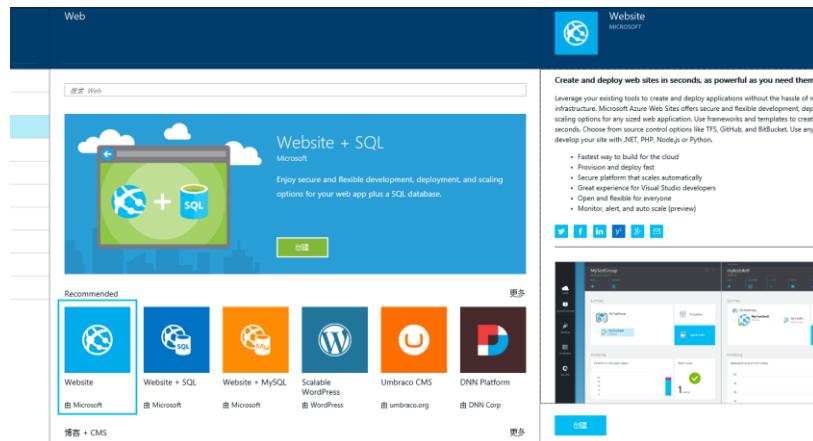


图 1.5：网站和 SQL 数据库创建导览

启动板

启动板是在登录到预览门户后看到的第一个屏幕。这是一组磁贴的集合，您可以重新组织和删除这些磁贴，也可以调整其大小。在查看订阅中的边栏选项卡时，可以将它们固定在启动板上，以便以后重新返回。还可以调整磁贴的大小，以突出特定数据。

启动板上的磁贴最常见的调整大小选项。

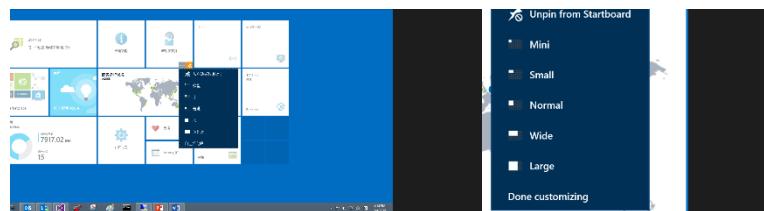


图 1.6：调整大小选项

很多磁贴根据其大小，显示不同的指标和图标。例如，“服务运行状况”磁贴在最大大小时显示世界地图，但是在最小尺寸时，它只显示运行状况正常的服务的数量。默认情况下，在创建服务的新实例时，将该实例的对应磁贴添加到启动板的选项已经选中。

显示一组固定磁贴的常见启动板。

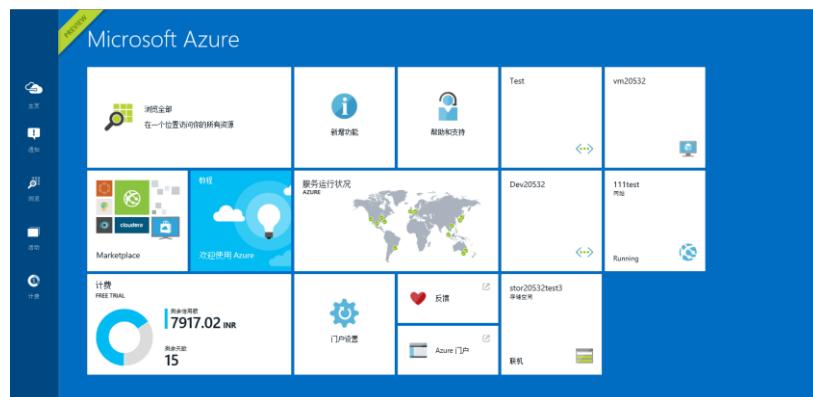


图 1.7：含自定义磁贴的启动板

MCT USE ONLY STUDENT USE PROHIBITED

计费

以前，必须从管理门户切换到帐户门户才能查看您服务的计费数据。在预览门户中，可以图表和信息图的形式查看您整个订阅的最新计费数据。图表和信息图将帮助您理解各个服务实例和服务类型的影响。消耗速度之类的图表还可帮助您预测每个结算周期您的帐户要扣取的费用。

预览门户中的计费可视化图表。



图 1.8：计费边栏选项卡

资源组

资源组代表了一种合理分组服务实例的新方式。服务实例称为资源，一组资源可以存在于一个资源组中。使用资源组，可以查看订阅中某个特定组的指标和计费数据。资源组还可以让服务实例共享同一个生命周期，在该生命周期中，您可以创建定义了多个资源的组或者移除组，而资源管理器确保各个资源也同时移除。资源组将在第 11 章“自动化与 Azure 资源的集成”中深入介绍。

资源组和基于角色的访问控制

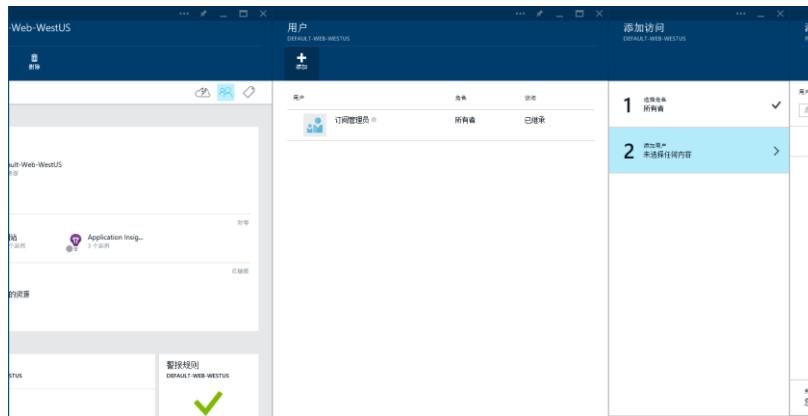


图 1.9：资源组

演示 1：使用预览门户

对于此演示，您将使用自己的计算机。开始本次演示前，必须完成以下步骤：

验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

演示步骤

1. 打开预览门户网站(<https://portal.azure.com>)。

2. 查看启动板自定义模式。
3. 关闭启动板自定义模式。
4. 打开**新建**面板，然后单击**计算**，然后单击**Azure Marketplace**。
5. 在显示的**Marketplace**边栏选项卡上，选择**虚拟机**，然后选择最新版本的**Ubuntu Server LTS**，单击**创建**开始一个新的导览。
6. 在**创建虚拟机(Ubuntu Server)**，查看创建一个关联的**虚拟网络**的导览部分。
7. 关闭**创建虚拟机**边栏选项卡，查看警告消息，消息会通知您将丢失对全部或部分导览的更改。

门户 URL

过去的门户(Silverlight)

此门户在 2014 年 9 月 1 日前有效。此门户的地址是 <http://windows.azure.com>，但现在它已完全退出服务并已不再可用。

管理门户

管理门户是大多数 Azure 场景中使用的当前门户。它是在 TechEd 2012 上引入的，并包含所有的可用的 Azure 服务。



参考资料链接:

<http://manage.windowsazure.com>
<http://manage.windowsazure.cn> (中国版 Windows Azure)

预览门户

预览门户是在//build 2014 上引入的未来门户。此门户有新的布局，新的 UI 概念。此门户还包含资源组和 Application Insight 等独有的功能。

参考资料链接: <http://portal.azure.com>

演示 2：在门户之间切换

对于此演示，您将使用自己的计算机完成实验。开始本次实验前，必须完成以下步骤：

验证您收到的密码可以从培训提供商那里登录到 Azure 门户，这些密码和 Azure 帐户将在本课程的所有实验中使用。

演示步骤

1. 打开管理门户 (<https://manage.windowsazure.com>)。
2. 通过使用与 Azure 帐户相关联微软帐户凭证登录。
3. 通过使用**快速创建**选项和以下细节创建一个**网站实例**：
 - **Url:** 为网站选择一个唯一的名称
 - **Web 宿主计划:** 创建新的宿主计划

- 区域：选择靠近的区域
4. 切换到新的预览门户。
 5. 查看在前面的步骤中创建的网站的细节。
 6. 访问新网站。



注释：占位符页显示一条消息，它表明网站是成功创建了。这个页面将显示，直到你上传一个 web 应用程序到您的新网站。

MCT USE ONLY. STUDENT USE PROHIBITED

实验 A：探索 Azure 预览门户

场景

新的预览门户已经发布，团队指派您去摸索一下新门户，然后培训其他团队成员如何使用新门户。您决定自定义新门户的一些功能，然后创建新的服务实例。

 **注释：**本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 账户，请继续完成**实验 A**；如果您拿到的是中国版 Windows Azure 账户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 登录到预览门户。
- 自定义启动板。
- 认识边栏选项卡。
- 认识导览。
- 认识导览部分。
- 关闭导览而不保存更改。

实验设置

预计时间：15 分钟

您将使用自己的计算机完成实验。开始本次实验前，必须完成以下步骤：

验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

 **注释：**本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 账户，请继续完成**实验 A**；如果您拿到的是中国版 Windows Azure 账户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

练习 1：登录到预览门户

场景

先使用新门户的 URL 登录到预览门户。

此练习的主要任务如下：

1. 登录到预览门户
- **任务 1：登录到预览门户**
1. 登录预览门户网站(<https://portal.azure.com>)。
 2. 单击 **Get Started**。

结果：完成本练习后，您将直接登录预览门户，而无需先经历管理门户。

练习 2：自定义预览门户

场景

您想根据自己团队的需求练习自定义启动板。您还想演练导览

此练习的主要任务如下：

1. 自定义启动板
2. 查看边栏选项卡
3. 开始导览

► 任务 1：自定义启动板

1. 查看启动板自定义模式。
2. 调整服务运行状况磁贴为普通大小。
3. 关闭启动板自定义模式。

► 任务 2：查看边栏选项卡

1. 打开浏览器面板，然后单击门户设置。
2. 查看门户设置边栏选项卡。

► 任务 3：开始导览

1. 打开新建面板，单击 Web+移动，然后单击 Azure Marketplace。
2. 在 Marketplace 边栏选项卡中，选择 Web 组，然后选择创建选项为 Node JS Starter Site 开始新的导览。
3. 在网站边栏选项卡中，查看新建一个 Web 宿主计划的导览部分。
4. 关闭网站边栏选项卡，查看警告消息，消息会通知您将丢失对全部或部分导览的更改。

结果：完成此练习后，您将完成查看边栏选项卡、导览和导览部分的任务。

实验 B：认识中国版 Windows Azure 管理门户

场景

Azure 管理门户是您创建和管理 Azure 服务的接口，通过 Azure 管理门户您可以方便地创建和管理您的网站，虚拟机，数据库等服务。

目标

完成本实验后，您将获得以下知识：

- 登录中国版 Windows Azure 管理门户。
- 认识门户页面的导航栏。
- 认识管理页面的工具栏。

预计时间：15 分钟

对于此实验，您将使用自己的计算机完成实验。开始本次实验前，必须完成以下步骤：

1. 验证您拿到的是中国版 Windows Azure 账户。
2. 验证使用您的中国版 Windows Azure 账户可以登录到世纪互联运营的 Windows Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：登录并了解 Azure 管理门户

场景

使用中国版 Windows Azure 门户地址登录 Azure 管理门户。了解 Azure 管理门户的布局。

此练习的主要任务如下：

1. 登录到 Azure 管理门户。
2. 认识管理门户的布局。

► 任务 1：登录到 Azure 管理门户

1. 登录到 Azure 管理门户。
2. 在开始屏幕上，单击 **Internet Explorer** 磁贴。
3. 在地址栏中，输入 <https://manage.windowsazure.cn>。
4. 点击回车键。
5. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
6. 如果您的界面语言不是中文，可以点击页面右上角的**地球**图标，并在菜单中选择**中文(简体)**语言。

 **注释：**切换界面语言的另一个方式是，在页面右上角，点击帐户链接，在弹出的菜单中，点击**View my bill**，滚动屏幕，在页面的左下角点击**English**，在弹出的菜单中，点击**中文(简体)**。在页面的右上角，点击**门户**按钮。

► 任务 2：认识管理门户的布局

1. 在管理门户页面，页面左侧为导航栏。
2. 在导航栏中的第一项为**所有项目**，这里列出来了所有的您已创建的服务的名称。
3. 在导航栏中的其它项为每一个可用服务的名称。

MCT USE ONLY. STUDENT USE PROHIBITED

4. 在管理门户页面，页面下方为工具栏。
5. 在工具栏中的左侧为**新建**按钮，通过这个按钮可以创建 Azure 服务。
6. 在管理门户的右上角为**地球**图标和用户帐户链接。

结果：完成本练习后，您将使用您的 Azure 帐户登录到中国版 Windows Azure 管理门户。

章节复习和作业

本章中，您了解了作为平台的 Azure 及其功能。您还预览了可用来管理 Azure 订阅的工具和功能。

 **最佳做法：**本章介绍的很多服务是学习本课程必备的基础知识。如果您从未使用过本章提到的任何服务，请花时间复习这些服务，然后才能完成后面的章节。这将确保您为本课程后面的章节做好准备。

复习问题

问题：您想要构建 iOS 和 Android 应用程序。您的应用程序需要一个后端 Web 服务，于是您决定在 Azure 中托管服务。哪些服务可用来完成此任务？

MCT USE ONLY. STUDENT USE PROHIBITED

第 2 章 使用 Azure 虚拟机建立开发环境

目录:

章节概述	2-2
第 1 课: 搭建 Azure 虚拟机	2-3
第 2 课: Azure 虚拟机工作负载	2-9
第 3 课: 迁移 Azure 虚拟机实例	2-13
实验 A: 创建用于开发和测试的 Azure 虚拟机	2-17
实验 B: 在中国版 Windows Azure 中创建用于开发和测试的虚拟机	2-21
章节复习和作业	2-29

章节概述

虽然很多 Microsoft Azure 服务都使用虚拟机，但有时候，应用程序可能有独特的需求，它们需要非托管的虚拟机。作为其基础结构即服务(IaaS)产品的一部分，Azure 提供了联网、备份和虚拟化服务。第 1 节“*Azure 虚拟机*”将介绍虚拟机服务，并描述可用来创建虚拟机的选项。第 2 节“*Azure 虚拟机工作负载*”将提供可部署到虚拟机的各种工作负载类型的详细信息。第 3 节“*迁移 Azure 虚拟机实例*”将描述将虚拟机迁入和迁出 Azure 的选项。

目标

完成本章后，您可以做到：

- 描述 Azure 中的虚拟机服务。
- 将 Linux 或 Microsoft 工作负载部署到虚拟机。
- 将虚拟硬盘导入 Azure。
- 监视虚拟机端点。

第 1 课 搭建 Azure 虚拟机

Azure 中的虚拟机服务提供了可向上 (Scale up) 或向外扩展 (Scale out) 的快速计算，并且可完全自定义。Azure 管理门户提供了大量模板，可让您非常轻松地开始使用流行的服务器操作系统。

本节描述 Azure 中的虚拟机服务，并提供有关其部分独特功能的详细信息。

课程目标

完成本节后，您可以做到：

- 描述虚拟机服务。
- 描述预建映像和自定义映像选项。

虚拟机概述

虚拟机提供了高度灵活的按需计算选项来运行应用程序工作负载。Azure 虚拟机提供了各种虚拟机大小，兼容 Windows 和 Linux，并且深度兼容 Hyper-V 固定大小虚拟硬盘。托管在 Azure 中的虚拟机可承载各种工作负载。

- 按需计算
- 与 Windows Server 和 Linux 兼容
- 基于 Hyper-V 虚拟化
 - 可以在云和数据中心之间转移虚拟硬盘

云服务

虚拟机在 Azure 云服务中组合在一起。云服务作为虚拟机的网络边界和安全边界，除了从预定义端口或端点发出的流量外，云服务将虚拟机与所有公共流量隔开。云服务将在第 6 章“管理 Azure 中的云服务”中详细讨论。可以对端点执行负载平衡，以确保可使用四层算法 (Layer-4 algorithm)，在多台虚拟机上分摊流量。

Azure 云服务负载平衡器

<http://go.microsoft.com/fwlink/?LinkID=525326>

伸缩和可用性

Azure 中的虚拟机可以组合来实现高伸缩性和高可用性。将多个虚拟机放入相同云服务中，就可以建立应用程序任何一层的多个实例。例如，可以将一个 Web 应用程序托管在四个基于 Windows Server 2012 的虚拟机上，且每个虚拟机都启用 Internet Information Services(IIS)。可以将这些虚拟机放在一个可用性集中，使得其中至少一个虚拟机始终可用。根据 Azure 正常运行时间服务级别协议(SLA)，可用性集里面至少要有两个虚拟机的实例。如果将虚拟机的多个实例添加到可用性集，那么应该考虑配置自动缩放。自动缩放可让您启动和停止虚拟机，以满足应用程序的需求。对于虚拟机，可以创建您认为应用程序将需要的最大数量的实例，然后用一个既定的指标启用自动缩放，该指标将启用缩放操作。自动缩放和可用性集将在第 10 章“管理 Azure 中的基础结构”中进一步讨论。

Hyper-V

Azure 中的虚拟机使用著名的 Hyper-V 虚拟硬盘格式(.vhd)作为其硬盘驱动器的格式。由于 Azure 使用.vhd 格式，因此可以将固定大小的虚拟硬盘文件从现有基础结构直接上传到 Azure。还可以将虚拟硬盘文件从 Azure 下载到您的数据中心。

使用映像搭建虚拟机

管理门户提供了多种映像和脚本工具来帮助您在 Azure 中创建新的虚拟机。门户中提供的模板映像是由 Microsoft 或授权的第三方创建的，并得到其完全支持。可以将这些映像作为基础，在 Azure 中创建全新项目，或者用于将现有工作负载迁移到 Azure。

Microsoft 映像

Azure 虚拟机环境已经支持很多 Microsoft 服务器软件。受支持的常见 Microsoft 工作负载包括：

- Microsoft BizTalk Server 2013
- Microsoft Dynamics AX、Microsoft Dynamics GP 和 Microsoft Dynamics NAV
- Microsoft Project Server 2013
- Microsoft SharePoint Server 2010 和 Microsoft SharePoint Server 2013
- Microsoft System Center 2012 Service Pack 1
- Microsoft SQL Server 2008、Microsoft SQL Server 2012 和 Microsoft SQL Server 2014
- Microsoft Team Foundation Server 2012

• Microsoft 已经提供了多种映像：

- Microsoft SQL Server
- Microsoft SharePoint
- OpenSUSE
- Microsoft BizTalk Server

• 您的 Azure 订阅可能有某些自定义映像

- 例如，MSDN 订阅提供了预装 Microsoft Visual Studio 的 Windows 7 和 Windows 8.1 映像

兼容 Azure 虚拟机的 Microsoft 服务器软件的列表可在以下支持页上找到：

Azure 虚拟机的 Microsoft 服务器软件支持

<http://go.microsoft.com/fwlink/?LinkId=525327>

并非所有受支持的 Microsoft 工作负载都有模板映像。例如，SQL Server 2008 是受支持的工作负载，但是库映像只有 Microsoft SQL Server 2008 R2 的映像。

可以使用管理门户中的 Microsoft SQL Server 2012 SP1 库映像创建虚拟机。

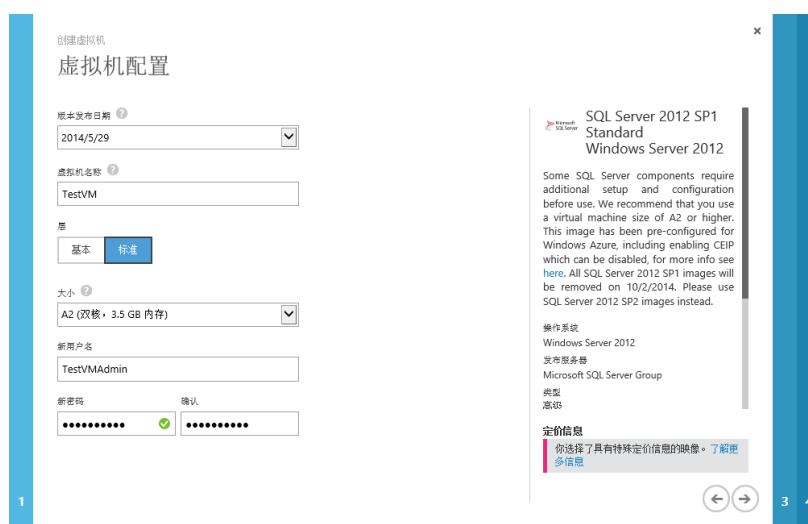


图 2.1: SQL SERVER 2012 SP1 库模板

开源和第三方映像

Microsoft 和第三方提供商有各种开源映像，包括：

MCT USE ONLY. STUDENT USE PROHIBITED

- CoreOS
- Ubuntu
- openSUSE
- OpenLogic

您也可以创建自己的基于 Linux 的自定义虚拟机，并将其作为 vhd 文件上传到 Azure。

从 Azure 预览门户中的库映像创建基于 Ubuntu Server 14 的虚拟机

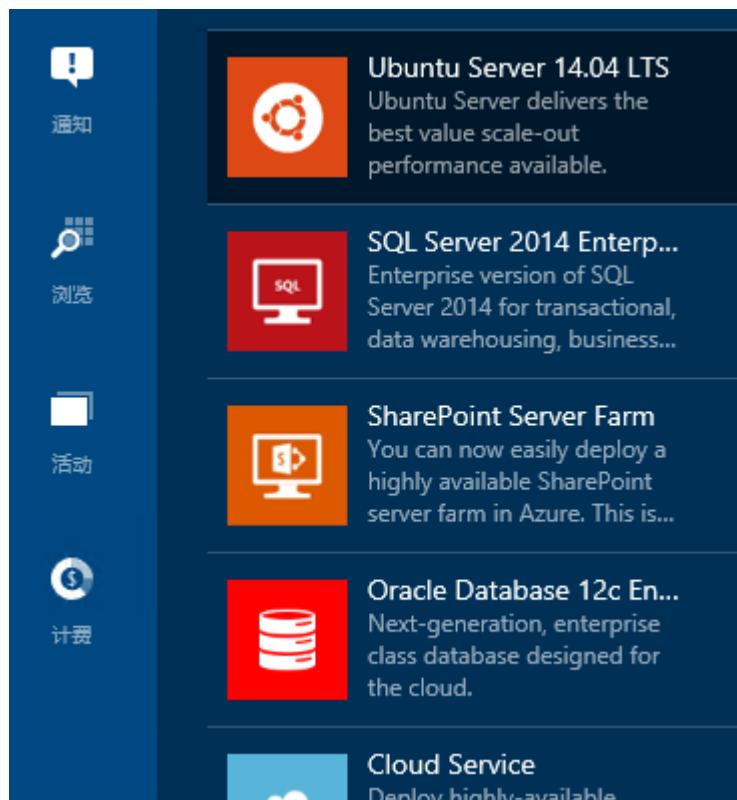


图 2.2: UBUNTU SERVER 14 库模板

自定义映像

可以基于现有虚拟机创建自定义映像。在需要创建虚拟机的多个实例来获得相似配置的情况下，可以使用自定义映像来创建虚拟机的副本。自定义映像可以是通用化（generalized）的，这样就可以克隆您所需数量的副本，也可以是专门化的，这样就可以创建虚拟机的检查点，以保留其在某个特定时间点的状态。

通用化虚拟机映像是去设置的操作系统映像（Windows 中的 Sysprep 或 Linux 中的 waagent），这样就可以在其克隆到其他虚拟硬盘后再设置它们。

注释: 在 Azure 中，在捕获通用化的虚拟机映像后，映像将删除虚拟机，而不影响磁盘。然后，可以使用此映像创建您需要的任意数量的该虚拟机副本。捕捉映像时，必须先停止虚拟机，然后再获取泛化的映像。

专门化虚拟机映像是运行中虚拟机的快照，可用来以后将虚拟机恢复到该时间点。

注释: 在 Azure 中，可以从正在运行或已停止的虚拟机获取专门化的虚拟机映像。这不会影响现有虚拟机。可以使用专门化映像创建新的虚拟机，这些虚拟机不像使用通用化映像创建的虚拟机那样需要进行初始化配置，生成即可使用。

VM Depot

VM Depot 是由社区贡献的虚拟机映像目录。



<http://go.microsoft.com/fwlink/?LinkId=525339>

VM Depot 中的映像包括操作系统、应用程序工作负载以及预先配置的开发堆栈（Development Stack）。VM Depot 直接集成在管理门户和预览门户中，并且允许将 Depot 映像导入您的 Azure 帐户。导入后，可以使用虚拟机向导中的**我的映像**选项从该映像创建虚拟机。

管理门户中的 VM Depot 界面

<http://vmdepot.msopentech.com>

- VM Depot是 Azure 中运行的虚拟机的模板集合，这些模板是由社区推动的。
- 可以使用自动化脚本或管理门户创建虚拟机
- 可以向VM Depot贡献新映像
- VM Depot中提供了全面的映像搜索功能

The screenshot shows a search results page for 'Ubuntu' on the VM Depot website. The left sidebar has categories: ALL, CENTOS, OPENSUSE, SUSE ENTERPRISE SERVER, and UBTUNUT, with UBTUNUT selected. The main area shows a list of images with small icons and names. On the right, a detailed view of the 'Neo4j Community 1.8...' image is shown, including its publisher (Cognosys), publish date (2/10/2013), packages (neo4j community 1.8, security hardened ubuntu 12.04 lts), and size (30 GB). A 'More' link is also present.

图 2.3: VM DEPOT

注释: VM Depot 映像没有根据安全性、兼容性或性能进行组织或筛选。映像由映像贡献者授予您许可，他们可能没有为已安装的软件提供许可证或支持。

演示 1：在 Azure 中创建虚拟机

演示步骤

1. 登录到预览门户(<https://portal.azure.com>)。
2. 单击 **Get Started**。
3. 查看为订阅的虚拟机列表。
4. 通过使用以下细节创建一个虚拟机：
 - 主机名: **vm28532demo**[自定义名称]

MCT USE ONLY. STUDENT USE PROHIBITED

- 用户名: **Instructor**
 - 密码: **AzurePa\$\$w0rd**
 - 定价层: **A2 标准**
5. 通过使用远程桌面, 连接到新创建的虚拟机。
 6. 关闭远程桌面连接。

演示 2：在中国版 Windows Azure 中创建虚拟机

演示步骤

1. 登录到 Azure 管理门户。
2. 在开始屏幕上, 单击 **Internet Explorer** 磁贴。
3. 在地址栏中, 输入 <https://manage.windowsazure.cn>。
4. 按 Enter 键。
5. 输入您的 Azure 帐户的邮箱地址和密码, 单击 **Sign In** 按钮。
6. 如果您的界面语言不是中文, 可以单击页面右上角的**地球**图标, 并在菜单中选择**中文(简体)**语言。
7. 在 Azure 管理门户页面左侧的导航栏中, 向下滚动, 单击**虚拟机**。
8. 在页面下方的工具栏左侧, 单击**新建**按钮。
9. 在弹出的新建菜单中, 选择**虚拟机**, 选择**快速创建**。
10. 在右侧的创建虚拟机窗口中, 执行如下步骤:
 - a. 在**DNS 名称**文本框中, 输入 **vm28532demo[自定义名称]**。
 - b. 在**映像**下拉框中, 选择 **Windows Server 2012 R2 Datacenter (en-us)**。
 - c. 在**大小**下拉框中, 选择 **A2 (双核, 3.5GB 内存)**。
 - d. 在**用户名**文本框中, 输入 **Instructor**。
 - e. 在**新密码**和**确认**文本框中, 输入 **AzurePa\$\$w0rd**。
 - f. 在**区域/地缘组**中, 选择一个距离你较近的区域。
 - g. 然后单击页面右下角的**创建虚拟机**按钮。
11. 选择新创建的虚拟机。
12. 在页面下方的工具栏中, 单击**连接**。
13. 在 Internet Explorer 下载对话框中, 单击**打开**按钮。
14. 在远程桌面连接窗口, 执行如下步骤:
 - a. 如果弹出**无法识别此远程连接的发布者。是否仍要连接?** 的对话框, 选中**不再询问我是否连接到此计算机**。
 - b. 单击**连接**按钮。
15. 在**Windows 安全**窗口中, 执行如下步骤:
 - a. 在**用户名**文本框中, 输入 **Instructor**。
 - b. 在**密码**文本框中, 输入 **AzurePa\$\$w0rd**。
 - c. 然后单击**确定**按钮。

16. 在远程桌面连接窗口中，执行如下步骤：
 - a. 在**证书名称**中，确认证书名称与你的虚拟机名称匹配。
 - b. 选中**不再询问我是否连接到此计算机**。
 - c. 然后单击**是**按钮。
 - d. 如果弹出**Networks** 对话框，单击**No** 按钮。
17. 关闭远程桌面连接窗口。



注释：Azure 试用版帐户会受到核数限制，建议完成以上 Demo 以后将虚拟机关闭。

第 2 课 Azure 虚拟机工作负载

可以将现有应用程序工作负载部署到 Azure 中运行 Windows 或 Linux 操作系统的虚拟机。

本节描述在将应用程序工作负载部署到 Azure 中的虚拟机时要考虑到几个注意事项。

课程目标

完成本节后，您将能够描述将 Windows 或 Linux 工作负载部署到 Azure 的关键注意事项。

Windows 工作负载

Azure 已经支持很多常见的 Microsoft 工作负载（服务器软件）。其中的大部分工作负载在 Azure 门户中还包含对应的虚拟机模板。安装在 Azure 虚拟机中的所有软件都必须有必要的许可证。但是，如果使用库模板，则许可证的成本已经包含在虚拟机成本中。

 **注释：**使用 Windows Server 模板的虚拟机包含 Windows Server 许可证的成本。使用 SQL Server 或其他 Microsoft 服务器软件模板创建的虚拟机通常定价高于只安装了 Windows Server 的虚拟机。这是因为除了 Windows Server 许可证的成本外，还要加上其他软件许可证的成本。

- 对于与构建企业内部应用程序的方式相仿的很多工作负载场景，可以使用虚拟机和虚拟网络。
- 示例：

Web 应用程序

- Web 服务器 (IIS)
- SQL Server
- 状态服务器

SharePoint

- Web 前端
- SQL Server
- 应用程序服务

如果先开始使用操作系统库模板，然后再安装服务器软件，那么必须立即使用许可证迁移来授予服务器软件的许可。

Azure 虚拟机的 Microsoft 服务器软件支持

<http://go.microsoft.com/fwlink/?LinkId=525327>

读者可以查看补充指导文章，这些文章将帮助您规划如何在 Azure 虚拟机上安装多种常见 Microsoft 软件工作负载。

部署 Microsoft SharePoint 的补充指导

<http://go.microsoft.com/fwlink/?LinkId=525328>

部署 Microsoft SQL Server 的补充指导

<http://go.microsoft.com/fwlink/?LinkId=525330>

Linux 工作负载

在 Azure 平台上，已经通过库映像提供了很多常见的 Linux 分发包。加上 VM Depot 中的映像，Azure 平台中已有足够的映像来运行大多数流行的 Linux 工作负载。

Azure 平台提供了用于管理虚拟机（无论运行 Linux 还是 Windows）的通用界面。很多常用功能（如捕获映像、附加磁盘和停止虚拟机）在预览门户和管理门户中使用相同的按钮，在跨平台命令行界面中使用相同的操作。这样，您就能像管理基于 Windows 的虚拟机一样，管理基于 Linux 的虚拟机。

在设置好虚拟机后，可使用虚拟机扩展来配置基于 Linux 的虚拟机，这样就可以使用安全 Shell (SSH) 来访问 Linux 虚拟机。然后可以使用 PuTTY 之类的工具来访问基于 Linux 的新虚拟机。



PuTTY

<http://go.microsoft.com/fwlink/?LinkId=525342>

- 可以使用 Azure 中提供的基于 Linux 的虚拟机来部署常见 Linux 工作负载

- 示例：

- Apache Lucene
- LAMP (Linux, Apache, MySQL, PHP)
- Couchbase (分布式)
- Drupal
- Docker
- Chef 或 Puppet
- Docker

虚拟机大小

虚拟机大小 (Size) 就是虚拟机的规模。虚拟机可设置为各种大小。这些大小提供了自定义虚拟机性能的各种选项，使其足以容纳工作负载，同时产生经济高效的订阅。



注释：每秒最大输入/输出操作数(IOPS)按以下格式显示：
<磁盘数>x<每磁盘最大 IOPS>
例如，4x500 表示每个磁盘最大 IPOS 为 500，最多 4 个磁盘。

<http://azure.microsoft.com/pricing/details/virtual-machines/>



基本层 A 系列

基本层 A 系列虚拟机提供了经济高效的工作负载托管方法，这类工作负载只需要虚拟机的一个实例，不需要自动缩放或负载平衡功能。下表显示了各种基本 A 系列大小之间的差异。

大小	内核数	内存	最大 IOPS
A0	共享	768 兆字节(MB)	1x300
A1	1	1.75 千兆字节(GB)	2x300
A2	2	3.5GB	4x300
A3	4	7GB	8x300
A4	8	14GB	16x300

标准层 A 系列

标准层 A 系列虚拟机是在绝大部分工作负载中使用的最常见的虚拟机。标准层虚拟机支持负载平衡和自动缩放。下表显示了最常见大小间的区别。

大小	内核数	内存	最大 IOPS
A0	共享	768MB	1x500
A1	1	1.75GB	2x500
A2	2	3.5GB	4x500
A3	4	7GB	8x500
A4	8	14GB	16x500
A5	2	14GB	4x500
A6	4	28GB	8x500
A7	8	56GB	16x500
A8	8	56GB	16x500
A9	16	112GB	16x500

标准层 D 系列

标准层 D 系列虚拟机包含固态驱动器(SSD)用作临时磁盘，此外还有更快的处理器以及每个内核更多内存。

大小	内核数	内存	临时磁盘大小(SSD)	最大 IOPS
D1	1	3.5GB	50GB	1x500
D2	1	7GB	100GB	2x500
D3	2	14GB	200GB	4x500
D4	4	28GB	400GB	16x500
D11	8	14GB	100GB	4x500
D12	2	28GB	200GB	8X500
D13	4	56GB	400GB	16x500
D14	8	112GB	800GB	32x500

标准层 G 系列

标准层 G 系列虚拟机包含 Azure 平台中提供的最高性能的 CPU。

大小	内核数	内存	临时磁盘大小(SSD)
G1	2	3.5GB	406GB

MCT USE ONLY. STUDENT USE PROHIBITED

大小	内核数	内存	临时磁盘大小(SSD)
G2	4	7GB	812GB
G3	8	14GB	1,630GB
G4	16	28GB	3,250GB
G5	32	14GB	6,500GB

第 3 课 迁移 Azure 虚拟机实例

Azure 提供的基础结构选项便于在云中轻松扩展现有数据中心或创建新环境。凭借联网、备份、站点恢复和虚拟机等服务，Azure 具备了部署现有复杂生产应用程序所必需的服务。

本节描述可用来将虚拟机迁移到 Azure 的各种方法。

课程目标

完成本节后，您可以做到：

- 将虚拟机迁移到 Azure。
- 描述 Azure 备份服务。
- 描述 Hyper-V 恢复管理器服务。

将虚拟机迁移到 Azure

由于 Azure 虚拟机原生支持 Hyper-V 格式，因此可以轻松将虚拟机从现有 Hyper-V 主机迁移到 Azure，反之亦然。此外还有各种其他选项可用来迁移虚拟机、分析现有工作负载，以及检查虚拟机与 Azure 的兼容性。

磁盘移动性

由于 Azure 和 Hyper-V 都支持常用的.vhd 格式，因此可以使用第三方存储资源管理器（storage explorer）或自动化脚本轻松上传和下载虚拟磁盘。在 Windows PowerShell 中，可以使用以下活动添加和保存虚拟机：

- Add-AzureVhd
- Save-AzureVhd

在跨平台命令行界面中，可以使用以下命令创建虚拟机磁盘，并将其上传到 Azure：

- vm disk create
- vm disk upload

大多数情况下，必须使用脚本来将.vhd 文件上传到存储帐户，因为使用任何 Azure 门户都无法上传这些文件。.vhd 文件必须是第 1 代 Hyper-V 磁盘（Generation 1），并且必须是固定大小。目前还不支持第 2 代 Hyper-V 磁盘(.vhdx)。

迁移加速器

迁移加速器是一种工具，可用来分析现有应用程序工作负载，然后执行虚拟机的完全迁移，包括网络和端点配置。迁移是使用一系列代理和配置或者处理服务器执行的。可以使用迁移加速器分析在 Windows Server 2008 R2 上运行的现有多层应用程序工作负载，以便实现自动迁移。迁移到 Azure 可以自动发生，而不会影响现有基础结构，还可以在全面测试目标站点后再进行割接。迁移加速器支持物理机、VMWare、Hyper-V、Amazon Web Services 和各种其他平台上的现有工作负载。

Microsoft 迁移加速器简介

<http://go.microsoft.com/fwlink/?LinkId=525331>

可以使用迁移加速器来发现现有工作负载、执行迁移，然后执行割接。

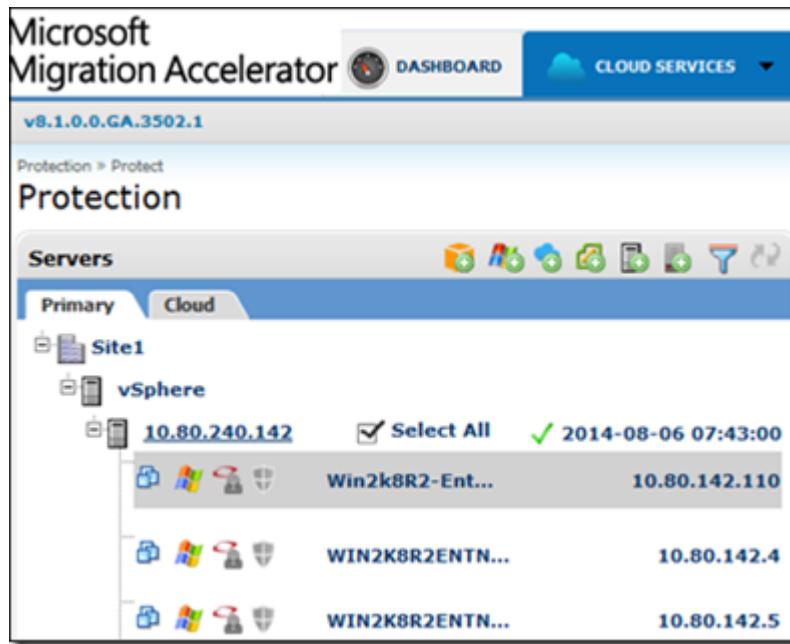


图 2.4：迁移加速器仪表板

就绪程度评估

虚拟机就绪程度评估分析现有服务器，以检查它们是否兼容 Azure 平台。运行该工具时，必须回答一系列有关虚拟机上的工作负载的问题。根据对运行中虚拟机的分析以及问题的答复，工具会生成详细的报告。生成的报告包含 Azure 中虚拟机配置的建议，以及在迁移工作负载前可阅读的文章的超链接。

Azure 虚拟机就绪评估

<http://go.microsoft.com/fwlink/?LinkId=525332>

可以使用虚拟机就绪程度评估报告来规划 Azure 迁移。

The screenshot shows a checklist titled "Virtual Machine Readiness Assessment" under the "What we checked" section. The checklist is organized into three columns: Category (Ready, Set, Move), Task, and Status (green checkmark or red warning triangle). A note at the bottom indicates that no work is required if a task has a green checkmark.

	Ready	Set	Move
Understand the benefits	✓		
Choose your scenario	✓		
Determine identity provider needs	✓		
Review unsupported roles and features	✓		
Evaluate hardware needs	✓		
Configure your network	✓		
Plan for storage	✓		
Prepare a disaster recovery plan	✓		
Secure your environment	✓		
Optimize your configuration	✓		
Get a subscription	✓		
Provision your virtual machine		⚠	
Move your data	✓		
Get healthy	✓		
Monitor your environment	✓		
Get support	✓		

✓ No work is required, you are good to go!
⚠ Planning or configuration is required before you move.

图 2.5：虚拟机就绪程度评估报告

备份和站点恢复

Azure 中的备份和站点恢复服务帮助您处理简单备份以及更高级的灾难恢复场景。这些服务非常灵活，可以在企业内部服务器上使用，也可以在 Azure 中的虚拟机上使用。

备份

备份是一种简单的自动备份解决方案，它使用其他常用的 Azure 服务和 Windows Server 功能来最小化备份数据所需的管理工作量。备份直接与 Windows Server 和 System Center 的数据保护功能集成。备份数据存储在可异地复制的存储帐户下。

备份已优化为只同步增量更改，传输期间所有数据是安全的。

Azure Backup

<http://go.microsoft.com/fwlink/?LinkId=525348>

备份	站点恢复
服务器数据的场外备份 传输过程中加密 集成在 System Center 和 Windows Server 中 只备份更改，而不是整个文件	将私有云复制到辅助位置 快速恢复虚拟机 与 Windows Server Hyper-V 副本集成 连接到 System Center Virtual Machine Manager 进行运行状况监视

MCT USE ONLY. STUDENT USE PROHIBITED

站点恢复

站点恢复是全面的灾难恢复服务，它提供了深度安排和监视功能。站点恢复还使用 System Center、Hyper-V Replica 和 SQL Server AlwaysOn 等现有技术。可以使用恢复计划，利用分布式虚拟机和自定义逻辑来安排服务的恢复。可以经常在与您的主位置隔离的环境中测试恢复计划。

站点恢复

<http://go.microsoft.com/fwlink/?LinkId=525350>

实验 A：创建用于开发和测试的 Azure 虚拟机

场景

在开始将应用程序从企业内部服务器迁移到 Azure 之前，必须先创建开发环境。您已选择使用 Azure 来托管 Windows Server 2012 虚拟机。在此虚拟机中，您将安装项目文件、Visual Studio 2013 Update 4、Azure SDK for .NET 2.5 和 Azure PowerShell。完成后，您将使用此虚拟机来完成余下的所有开发任务。



注释：本章为您提供了两套试验方案，如果您拿到的是国际版的 Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版的 Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 在 Azure 订阅中创建地缘组。
- 创建虚拟网络。
- 创建存储实例。
- 创建虚拟机。
- 管理虚拟机 VHD。
- 在虚拟机上安装开发软件。

实验设置

预计时间：90 分钟

对于此实验，您将使用自己的计算机完成实验。开始本次实验前，必须完成以下步骤：

- 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

本章为您提供了两套试验方案，如果您拿到的是国际版的 Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版的 Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

练习 1：创建网络和资源容器

场景

您想创建可用于新虚拟机的虚拟网络。作为创建虚拟网络的一部分，您还将创建资源组。

此练习的主要任务如下：

- 登录到预览门户
- 创建虚拟网络和资源组

► 任务 1：登录到预览门户

- 登录到预览门户 (<https://portal.azure.com>)。
- 单击 **Get Started**。

► 任务 2：创建虚拟网络和资源组

- 查看虚拟网络订阅列表。
- 通过使用以下详细信息创建一个虚拟网络：

MCT USE ONLY. STUDENT USE PROHIBITED

- 名称: **Dev28532**
- 位置: 选择最接近你的位置的地区
- 新资源组: **Dev28532**
- 空间地址 CIDR 块: **10.0.0.0/16**
- 子网 CIDR 块: **10.0.0.0/24**

结果: 完成此练习后, 您将在 Azure 中获得新的虚拟网络和资源组。

练习 2: 创建开发虚拟机

场景

您需要一个新的存储帐户, 创建虚拟机时, 您将使用该帐户。

此练习的主要任务如下:

1. 创建存储帐户
2. 创建虚拟机

► 任务 1: 创建存储帐户

1. 查看存储实例订阅列表。
2. 通过使用以下详细信息创建一个存储实例:
 - 存储空间: **stor28532[自定义名称]**
 - 位置: 选择最接近你的位置的地区
 - 定价层: **本地冗余(L)**

► 任务 2: 创建虚拟机

1. 查看虚拟机实例订阅列表。
2. 通过使用以下详细信息创建一个虚拟机实例:
 - 主机名: **vm28532[自定义名称]**
 - 用户名: **Student**
 - 密码: **AzurePa\$\$w0rd**
 - 定价层: **A3 标准**
 - 虚拟网络: **Dev28532**
 - 资源组: **Dev28532**
 - 存储帐户: **stor28532[自定义名称]**
3. 使用以下设置为虚拟机添加第二个磁盘:
 - 磁盘文件名: **vm28532-AllFiles.vhd**
 - 大小(GB): **6**
 - 存储帐户: **stor28532[自定义名称]**
 - 存储容器: **vhds**
4. 使用远程桌面连接到新创建的虚拟机。

结果: 完成此练习后，您将获得一个存储在新的存储帐户下的新虚拟机。

练习 3：配置用于开发的虚拟机

场景

现在您已有了新的虚拟机，您决定安装 Microsoft Visual Studio 2013 Ultimate Update 4、Azure PowerShell 以及 Azure SDK for .NET-2.5 软件开发包。

此练习的主要任务如下：

1. 创建 AllFiles 驱动器
2. 安装 Visual Studio 2013 Ultimate Update 4
3. 安装 Azure SDK for .NET 2.5
4. 安装 Azure PowerShell 模块

► 任务 1：创建 AllFiles 驱动器

1. 如果尚未打开 **Server Manager**，打开 **Server Manager**。
2. 禁用 **Internet Explorer Enhanced Security Configuration**。
3. 使用 Windows 磁盘管理器初始化新磁盘。
4. 使用以下设置格式化新的空硬盘分区：
 - 磁盘符： **F**
 - 卷标： **AllFiles**
 - 分区的风格： **MBR (Master Boot Record)**
5. 从 Companion MOC 网站(<http://www.microsoft.com/learning/companionmoc>)，下载压缩文件 **28532B：开发 Microsoft Azure 解决方案–AllFiles**。
6. 运行 AllFiles 可执行文件，提取内容到驱动器 F。
 - 提取目标： **Allfiles(F):**

► 任务 2：安装 Visual Studio 2013 Ultimate Update 4

1. 下载并运行 Visual Studio 2013 最终安装文件 <http://go.microsoft.com/fwlink/?LinkId=525334>。
2. 安装 **Visual Studio 2013 Ultimate** 的跟踪版本，只选择下面的可选功能：
 - **Microsoft Office 开发人员工具**
 - **Microsoft SQL Server Data Tools**
 - **Microsoft Web 开发人员工具**
3. 重新连接到之前创建的虚拟机。
4. 开始 **Visual Studio 2013 Ultimate**。
5. 关闭 **Visual Studio 2013 Ultimate**。

► 任务 3：安装 Azure SDK for .NET 2.5

1. 从 <http://go.microsoft.com/fwlink/?LinkId=525337> 启动，下载并运行适用于 Web 平台安装程序（Web Platform Installer）的 **Azure SDK for .NET-2.5**。
2. 在 Web 平台安装程序，安装 **Windows Azure SDK for.NET(VS 2013)-2.5** 包。

3. 打开 **Visual Studio 2013**。
4. 在服务器资源管理器（**Server Explorer**）窗格中打开 **Azure** 节点。
5. 导入 Azure 订阅。

► 任务 4：安装 **Azure PowerShell** 模块

1. 从 <http://go.microsoft.com/fwlink/?LinkId=320376> 下载和运行 **Azure PowerShell** 安装文件。
2. 安装 **Azure PowerShell** 模块。

 **注释：**现在连接到实验环境的虚拟机。

结果：完成此练习后，您的开发虚拟机上将安装 Visual Studio、Azure PowerShell 和 Azure SDK。

实验 B：在中国版 Windows Azure 中创建用于开发和测试的虚拟机

场景

在开始将应用程序从企业内部服务器迁移到 Azure 之前，必须先创建开发环境。您已选择使用 Azure 来托管 Windows Server 2012 虚拟机。在此虚拟机中，您将安装项目文件、Visual Studio 2013 Update 4、Azure SDK for .NET-2.5 和 Azure PowerShell。完成后，您将使用此虚拟机来完成余下的所有开发任务。

目标

完成本实验后，您将能够：

- 创建虚拟网络。
- 创建存储实例。
- 创建虚拟机。
- 管理虚拟机 VHD。
- 在虚拟机上安装开发软件。

预计时间：12 小时

对于此实验，您将使用自己的计算机完成实验。开始本次实验前，必须完成以下步骤：

1. 验证您拿到的是中国版 Windows Azure 帐户。
2. 验证使用您的中国版 Windows Azure 帐户可以登录到 Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：创建网络

场景

您想创建可用于新虚拟机的虚拟网络。

此练习的主要任务如下：

1. 登录到 Azure 管理门户
2. 创建虚拟网络

► 任务 1：登录到 Azure 管理门户

1. 在开始屏幕上，单击 **Internet Explorer** 磁贴。
2. 在地址栏中，输入 <https://manage.windowsazure.cn>。
3. 按 Enter 键。
4. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
5. 如果您的界面语言不是中文，可以单击页面右上角的地球图标，并在菜单中选择 **中文(简体)** 语言。

► 任务 2：创建虚拟网络

1. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击 **网络**。
2. 在页面下方的工具栏左侧，单击 **新建** 按钮。
3. 在弹出的创建向导中，单击 **虚拟网络**，单击 **自定义创建**。
4. 在弹出的 **创建虚拟网络** 窗口中，执行以下步骤：

MCT USE ONLY. STUDENT USE PROHIBITED

- a. 在名称对话框中，输入 **Dev28532**。
 - b. 在位置下拉框中，选择一个离你位置较近的区域。
 - c. 在窗口右下角，单击右箭头按钮进入下一个页面。
5. 在 **DNS 服务器和 VPN 连接** 页面，单击右箭头进入**虚拟网络访问空间**页面。
 6. 在地址空间行，确保起始 IP 是 **10.0.0.0**，确保 CIDR 框中值为/16(65536)。
 7. 在子网行，确保起始 IP 是 **10.0.0.0**，确保 CIDR 框中值为/24(256)。
 8. 修改子网名称为 **Apps**。
 9. 单击**完成**按钮。

结果：完成此练习后，您将在 Azure 中获得新的虚拟网络。

练习 2：创建 Azure 虚拟机开发环境

场景

您需要一个新的存储帐户，创建虚拟机时，您将使用该帐户。

此练习的主要任务如下：

1. 创建存储帐户
2. 创建虚拟机

► 任务 1：创建存储帐户

1. 在 Azure 管理门户页面左侧的导航栏中，向下滚动，单击**存储**。
2. 在页面下方的工具栏左侧，单击**新建**按钮。
3. 在弹出的新建菜单中，选择**存储**，选择**快速创建**。
4. 在右侧弹出的创建存储帐户栏中，执行如下步骤：
 - a. 在 **URL** 文本框中输入 **stor28532[自定义名称]**。
 - b. 在**位置/地缘组**下拉框中选择一个距离你较近的区域。
 - c. 在**复制**下拉框中选择**本地冗余**。
 - d. 然后单击页面右下方的**创建存储帐户**按钮。

► 任务 2：创建虚拟机

1. 在 Azure 管理门户页面左侧的导航栏中，向下滚动，单击**虚拟机**。
2. 在页面下方的工具栏左侧，单击**新建**按钮。
3. 在弹出的新建菜单中，选择**虚拟机**，选择**从库中**。
4. 在**选择映像**窗口中，在中间栏浏览，选择 **Windows Server 2012 R2 Datacenter(EN-US)**。
5. 单击页面右下角的**下一步**按钮。
6. 在虚拟机配置窗口，执行如下步骤：
 - a. 在**虚拟机名称**文本框中，输入 **vm28532[自定义名称]**。
 - b. 在**层**选项中，选择**标准**。
 - c. 在**大小**下拉框中，选择 **A2(双核, 3.5GB 内存)**。

- d. 在新用户名文本框中，输入 **Student**。
 - e. 在新密码和确认文本框中，输入 **AzurePa\$\$w0rd**。
7. 然后单击页面右下角的**下一步**按钮，进入窗口 2 并执行如下步骤：
- a. 在**区域/地缘组/虚拟网络**下拉框中，在虚拟网络下选择 **dev28532**。
 - b. 在**存储帐户**下拉框中，选择 **stor28532[自定义名称]**。
 - c. 然后单击页面右下角的**下一步**按钮。
8. 在窗口 3 中，单击页面右下角的**完成**按钮。

 **注释：** 创建新虚拟机大约花费 10 到 15 分钟的时间，当虚拟机创建完成，页面下方的通知栏中会显示已成功创建虚拟机信息，单击确定按钮。

9. 在**虚拟机**页面，选择新创建的虚拟机 **vm28532[自定义名称]**。
10. 在页面下方的工具栏中，选择**附加**，单击**附加空磁盘**。
11. 在将空磁盘附加到虚拟机窗口中，执行如下步骤：
- a. 在**文件名**文本框中，更改文件名为 **vm28532-AllFiles**。
 - b. 在**大小(GB)**文本框中，输入数值 **6**。
 - c. 然后单击页面右下角的**确定**按钮。

 **注释：** 等待将空磁盘附加到虚拟机，这个过程大约花费 5 分钟。

12. 在**虚拟机**页面，选择新创建的虚拟机 **vm28532[自定义名称]**。
13. 在页面下方的工具栏中，单击**连接**。
14. 在 Internet Explorer 下载对话框中，单击**打开**按钮。
15. 在远程桌面连接窗口，执行如下步骤：
- a. 如果弹出**无法识别此远程连接的发布者。是否仍要连接？**的对话框，选中**不再询问我是否连接到此计算机**。
 - b. 单击**连接**按钮。
 - c. 在 Windows 安全窗口中，执行如下步骤：
 - i. 在**用户名**文本框中，输入 **Student**。
 - ii. 在**密码**文本框中，输入 **AzurePa\$\$w0rd**。
 - iii. 然后单击**确定**按钮。

16. 在**远程桌面连接**窗口中，执行如下步骤：
- a. 在**证书名称**中，确认证书名称与你的虚拟机名称匹配。
 - b. 选中**不再询问我是否连接到此计算机**。
 - c. 然后单击**是**按钮。
 - d. 如果弹出**Networks** 对话框，单击**No** 按钮。

结果： 完成此练习后，您将获得一个存储在新的存储帐户下的新虚拟机。

练习 3：配置虚拟机开发环境

场景

现在您已有了新的虚拟机，您决定安装 Microsoft Visual Studio 2013 Ultimate Update 4、Azure PowerShell 以及 Azure SDK for .NET-2.5 软件开发包。

此练习的主要任务如下：

1. 创建 AllFiles 驱动器
2. 安装 Visual Studio 2013 Ultimate Update 4
3. 安装 Azure SDK for .Net 2.5
4. 安装 Azure PowerSell Module

► 任务 1：创建 AllFiles 驱动器

1. 确认服务管理器已打开，通过在 **Start** 屏幕中，单击 **Server Manager** 磁贴打开服务管理器。
2. 在 **Server Manager** 的左侧栏，选择 **Local Server**。
3. 在 **Properties** 窗口中，找到 **IE Enhanced Security Configuration**，单击 **On**。
4. 在 **Internet Explorer Enhanced Security Configuration** 窗口中，执行如下步骤：
 - a. 在 **Administrators** 选项中，单击 **Off**。
 - b. 在 **Users** 选项中，单击 **Off**。
 - c. 然后单击 **OK** 按钮。
5. 在键盘上按 Windows 键+W 键打开 **Search-Settings**。
6. 在 **Search** 文本框中，输入 **disk**。
7. 单击 **Create and format hard disk partitions**。
8. 在 **Initialize Disk** 窗口中，执行如下步骤：
 - a. 确认 **Disk 2** 被选中作为初始化的磁盘。
 - b. 确认 **MBR (Master Boot Record)** 被选中作为磁盘形式
 - c. 然后单击 **OK** 按钮。
9. 在 **Disk Management** 窗口的下半部分中，执行如下步骤：
 - a. 找到新初始化的磁盘 **Disk 2**。
 - b. 右键单击未分配的磁盘，选择 **New Simple Volume**。
10. 在 **New Simple Volume Wizard** 窗口中，执行如下步骤：
 - a. 单击 **Next** 按钮。
 - b. 确认 **Simple volume size in MB** 中的数值为 6141。
 - c. 单击 **Next** 按钮。
 - d. 在 **Assign the following drive letter** 中，选择 **F**。
 - e. 单击 **Next** 按钮。
 - f. 确认 **File System** 中的数值为 NTFS。
 - g. 在 **Volume Label** 文本框中，更改数值为 AllFiles。
 - h. 单击 **Next** 按钮。

MCT USE ONLY. STUDENT USE PROHIBITED

- i. 单击 **Finish** 按钮关闭窗口并格式化磁盘 2。

 **注释:** 在弹出的 **You need to format the disk in drive F: before you can use it.** 窗口中, 单击 **Cancel** 按钮。

11. 在 Start 屏幕中, 单击 **Internet Explorer** 磁贴。
12. 如果弹出 **Set up Internet Explorer 11** 界面, 执行如下步骤:
 - a. 选择 **Use recommended security, privacy, and compatibility settings.**
 - b. 单击 **OK** 按钮。
13. 打开 <http://www.microsoft.com/learning/companionmoc>。
14. 滚动屏幕并找到 28532B 课程。
15. 单击 **28532B-CHS-AllFiles.exe** 以下载自解压包。
16. 在 Internet Explorer 下载对话框中, 单击 **Run** 按钮。下载自解压包大约花费 5 分钟时间。
17. 在 **Official Microsoft Learning Product License Terms** 窗口中, 单击 **Accept**。
18. 在 **WinRAR self-extracting archive** 窗口中, 执行如下步骤:
 - a. 在 **Destination folder** 中, 输入数值 F:\。
 - b. 单击 **Extract**。
19. 等待直到解压过程完成。

► 任务 2: 安装 Visual Studio 2013 Ultimate Update 4

1. 在 Start 屏幕中, 单击 **Internet Explorer** 磁贴。
2. 在地址栏中, 输入 <http://go.microsoft.com/fwlink/?LinkId=525334>。
3. 按 Enter 键。
4. 在 **select language** 下拉框中, 选择 **Chinese(Simplified)**, 然后等待页面刷新并显示中文。
5. 单击 **下载** 按钮。
6. 选择 **vs_ultimate.exe**。
7. 单击 **Next** 按钮。

 **注释:** 若弹出 Internet Explorer blocked a pop-upIE 对话框窗口, 则选择 Options for this Site, 单击 Always allow 按钮。

8. 在 Internet Explorer 下载对话框中, 单击 **Run** 按钮。
9. 在 Visual Studio 初始化窗口中, 选中**我同意许可条款和隐私策略**。
10. 单击 **下一步** 按钮。
11. 在**要安装的可选功能**中, 确认**有且只有**如下三个选项被选中:
 - Microsoft Office 开发人员工具
 - Microsoft SQL Server Data Tools
 - Microsoft Web 开发人员工具
12. 单击**安装**按钮。



注释: 下载和安装选择功能的过程大约花费 10 到 12 个小时的时间，具体花费时间与当前网络性能有关。

13. 等待直到 **Visual Studio 2013 Ultimate** 安装过程结束。
14. 在 **Visual Studio Ultimate 2013** 窗口中，单击启动。
15. 在 **Visual Studio** 欢迎使用。请登录 **Visual Studio** 窗口中，单击以后再说。
16. 在 Visual Studio 窗口中，执行如下步骤：
 - a. 在开发设置下拉框中，选择 **Visual C#**。
 - b. 然后单击启动 **Visual Studio**。
 - c. 等待直到 **Visual Studio** 的配置完成。
 - d. 单击右上角的关闭(X)按钮来关闭 Visual Studio 2013 Ultimate 窗口。

► 任务 3：安装 Azure SDK for .Net 2.5

1. 在 Start 屏幕中，单击 **Internet Explorer** 磁贴。
2. 在地址栏中，输入 <http://go.microsoft.com/fwlink/?LinkId=525337>。
3. 按 Enter 键。
4. 在 **select language** 下拉框中，选择 **Chinese(Simplified)**，然后等待页面刷新并显示中文。
5. 展开**详情**节点。
6. 滚动屏幕，并找到 VS 2013。然后单击 VS 2013。
7. 在 Internet Explorer 下载对话框中，单击 **Run** 按钮。



注释: 首先需要 1 到 2 分钟时间来安装 Web Platform Installer 5.0，然后通过该软件来获取 Azure SDK for.NET-2.5。

8. 确认 Web Platform Installer 5.0 安装的软件名称为 **Microsoft Azure SDK for.NET(VS 2013)-2.5**。
9. 单击 **Install**。
10. 单击 **I Accept**。



注释: 下载并安装 Azure SDK 的时间大约要 5 分钟。

11. 请等待，直到 Web Platform Installer 的安装过程完成。
12. 单击 **Continue** 按钮在浏览器中打开 Azure for.NET Developers 页面。
13. 关闭 Internet Explorer。
14. 在 Web Platform Installer 5.0 窗口中，单击 **Finish** 按钮。
15. 单击 **Exit** 按钮。
16. 在 Start 屏幕，单击屏幕底端的向下箭头以显示所有应用，向右滚动屏幕找到 Visual Studio 2013。
17. 右键单击 Visual Studio 2013，然后选择 Pin to Start。

► 任务 4：安装 Azure PowerShell Module

1. 在 Start 屏幕中，单击 **Internet Explorer** 磁贴。

2. 在地址栏中，输入 <http://go.microsoft.com/fwlink/p/?linkid=320376>。

3. 按 Enter 键。

 **注释：**若弹出 Internet Explorer blocked a pop-up IE 对话框窗口，则选择 Options for this Site，单击 Always allow 按钮。

4. 在 Internet Explorer 下载对话框中，单击 **Run** 按钮。

5. 确认 Web Platform Installer 5.0 安装的软件名称为 **Microsoft Azure PowerShell with Microsoft Azure SDK**。

6. 单击 **Install** 按钮。

7. 单击 **I Accept** 按钮。

 **注释：**下载并安装 Azure Module 的时间大约要 5 分钟。

8. 请等待，直到 Web Platform Installer 的安装过程完成。

9. 在 Web Platform Installer 5.0 窗口中，单击 **Finish** 按钮。

10. 单击 **Exit** 按钮。

11. 关闭 Internet Explorer 应用。

结果：完成此练习后，您的开发虚拟机上将安装 Visual Studio、Azure PowerShell 和 Azure SDK。

练习 4：连接到 Azure 订阅

场景

到此，您的 Azure 开发环境已经搭建完成，接下来，您要学习的是如何将 Visual Studio 2013 和 Azure PowerShell 连接到 Azure 订阅。

此练习的主要任务如下：

1. 下载订阅文件
2. 在 Visual Studio 2013 中连接到 Azure 帐户
3. 在 Microsoft Azure PowerShell 中连接到 Azure 帐户

► 任务 1：下载订阅文件

1. 在 Start 屏幕，单击屏幕底端的向下箭头以显示所有应用，向右滚动屏幕找到 Microsoft Azure PowerShell。
2. 单击 **Microsoft Azure PowerShell**。
3. 在打开的 Microsoft Azure PowerShell 窗口中，输入如下命令：

```
Get-AzurePublishSettingsFile -environment azurechinacloud
```

4. 在 Sign-In 页面，输入您在世纪互联的 Azure 帐户和密码，并单击 **Sign In** 按钮。
5. 验证系统打开浏览器，并进入 **Your subscription file is being generated, and the download will begin shortly** 页面。

MCT USE ONLY. STUDENT USE PROHIBITED

6. 在浏览器下方弹出的保存文件窗口中，选择 **Save**，单击 **Save as**。
7. 在 **Save as** 对话框中，保存您的订阅文件到系统的 **Downloads** 目录。
8. 关闭 Internet Explorer 窗口。

► **任务 2：在 Visual Studio 2013 中连接到 Azure 帐户**

1. 在 Start 屏幕，单击 Visual Studio 2013 磁贴。
2. 单击视图，单击**服务器资源管理器**。
3. 找到 Azure 节点，右键单击 Azure，单击**管理订阅**，选择**证书**选项卡，单击**导入**按钮。
4. 在导入 Microsoft Azure 订阅对话框中，单击**浏览**，在 Open 对话框中浏览 Downloads 目录，选中您下载的订阅文件，单击**Open**，返回导入 Microsoft Azure 订阅对话框，单击**导入**按钮。
5. 等待证书导入完成，单击**关闭**按钮。
6. 关闭 Visual Studio 2013 窗口。

► **任务 3：在 Microsoft Azure PowerShell 中连接到 Azure 帐户**

1. 在 Start 屏幕，单击屏幕底端的向下箭头以显示所有应用，向右滚动屏幕找到 Microsoft Azure PowerShell。
2. 单击 **Microsoft Azure PowerShell**。
3. 在打开的 Microsoft Azure PowerShell 窗口中，输入如下命令：

```
Import-AzurePublishSettingsFile [订阅文件]
```

 **注释：**订阅文件的格式如“C:\Users\Student\Downloads\1RMB Trial Offer-3-2-2015-credentials.publishsettings”

4. 输入以下命令，并验证您的 Azure 帐户信息导入成功：

```
Get-AzureAccount
```

5. 关闭 Microsoft Azure PowerShell 窗口。

 **注释：**至此，Azure 虚拟机开发环境你已经搭建完成了。

结果：完成这个 Lab 后，您将了解如何在本地通过 Visual Studio 2013 和 Azure PowerShell 连接到 Azure 订阅。

MCT USE ONLY. STUDENT USE PROHIBITED

章节复习和作业

本章中，您了解了 Azure 中的 IaaS 产品。虚拟网络、虚拟机、备份和站点恢复服务提供了在设计虚拟机网络或扩展现有数据中心时可使用的多种基础构件。

第 3 章 在 Azure 平台上托管 Web 应用程序

目录:

章节概述	3-2
第 1 课: Azure 网站服务	3-3
第 2 课: 在 Azure 中托管 Web 应用程序	3-6
第 3 课: 配置 Azure 网站	3-8
第 4 课: 发布 Azure 网站	3-13
第 5 课: 实验概述	3-16
实验 A: 使用 Azure 网站服务创建 ASP.NET 网站	3-19
实验 B: 在中文版 Windows Azure 中创建 ASP.NET 网站	3-23
章节复习和作业	3-29

章节概述

本章提供 Azure 网站服务的概述。第 1 节“Azure 网站服务”描述 Azure 中的网站服务。第 2 节“在 Azure 中托管 Web 应用程序”描述 Azure 网站的行为和生命周期。第 3 节“配置 Azure 网站”讨论可用于更改网站行为的各种配置选项。第 4 节“发布 Azure 网站”描述将使用 WebDeploy 的 Web 应用程序发布到 Azure 网站服务的过程。

目标

完成本章后，您可以做到：

- 创建网站服务实例。
- 将一个简单的 ASP.NET Web 应用程序发布到网站服务。
- 监视网站服务实例。

第 1 课 Azure 网站服务

很多情况下，用户都希望能快速便捷地直接将 Web 应用程序部署到云，而不是花费精力将 Web 应用程序重新制作成云应用程序再发布。网站服务能让您快速创建新网站，并能快速地反复更改网站。

本节描述网站服务。

课程目标

完成本节后，您可以做到：

- 描述网站服务。
- 列出网站服务的不同层。

Azure 网站服务概述

网站服务是平台即服务 (PaaS) 产品，可在 Azure 平台上托管 Web 应用程序。该服务是完全托管的，因此您可以使用任一门户，轻松配置 AlwaysOn、自定义域和自动缩放等高级功能。

灵活性

可以使用各种集成开发环境 (IDE) 和框架（如 .NET、Java、PHP、Node.js 或 Python）来开发 Web 应用程序，并部署到 Azure 网站服务。可以使用 Git 和 Kudu 来部署 Node.js 或 PHP Web 应用程序。还可以使用文件传输协议 (FTP) 或 Web 部署协议，将在 Microsoft Visual Studio 中开发的 Web 应用程序部署到网站服务。

Windows Azure 中简单、可伸缩的网站托管有以下优点：

- 提供在云中托管 Web 应用程序的快捷方法
- 允许缩放 Web 应用程序，而无需为了可伸缩性而重新设计
- 与 Visual Studio 集成
- 提供适合多种不同编程语言的开放平台

可伸缩性

由于网站服务是一种全托管的服务，因此您可以将注意力放在开发应用程序和解决业务问题上，而无需关注网站托管如何实现、硬件伸缩如何处理、硬件细节等等。可以在门户中配置自动缩放来扩展无状态 Web 应用程序。自动缩放创建网站服务的多个实例，这些实例可自动负载平衡，这样应用程序就可应付可能出现的需求峰值。

预建的网站模板

- 使用 Azure Marketplace 中的预建模板创建网站
- Marketplace 中有 30 多种开源应用程序、框架和模板：



网站层

网站服务分四层：免费、共享、基本和标准。可以使用 Web 宿主计划为一组网站实例分配一个层。您可以随时切换 Web 宿主计划的层。在免费层和共享层，网站的每个实例将按小时计费。在基本层和标准层，您的专用虚拟机（计算实例）将按小时计费，而不是按网站计费。

免费层提供 10 个免费的网站实例。所有实例每天共享 60 分钟的 CPU 时间池。免费层还实施每天 165 MB 的出站数据限制。这些实例托管在共享的计算实例或虚拟机上，它们与很多其他网站租户一起分享资源。

共享层和免费层有很多共同的地方，但是很多限制

比较宽松。例如，取消了出站数据限制，每个实例每天允许共用 240 分钟的 CPU 时间池。此层可以有最多 100 个网站实例。在此层中，网站还可以使用自定义域名。还可以手动扩充网站到六个不同的实例。这些网站实例托管在不同的共享计算实例中，并且自动平衡负载。

在基本层，网站实例不在共享环境中。相反，您有专用的计算实例，可以在此实例上托管任意数量的网站实例。除了免费层和共享层提供的功能外，此层还支持 AlwaysOn、自定义域名还可以启用 SSL，以及有限数量的 WebSocket 连接（每网站 350 个），可以手动将此层扩大到最多三个专用计算实例。

标准层提供与基本层、免费层和共享层相同的功能，但是还包括发布槽（Slots）和备份之类的其他功能。利用自动缩放功能，可以自动根据指标或计划缩放网站。



演示 1：创建网站

演示步骤

1. 登录到 Azure 的预览门户(<https://portal.azure.com>)。
2. 通过以下详细信息创建一个新的网站+ SQL 实例（资源组）：
 - 资源组: **rg28532**
 - Url: 为网站创建一个唯一名称
 - 数据库名: **rg28532db**
 - 服务器名: 为逻辑服务器创建一个唯一名称
 - 服务器管理登录: **testuser**
 - 密码: **TestPa\$\$w0rd**
3. 查看新建网站边栏选项卡。
4. 转到新建网站。

演示 2：在中国版 Windows Azure 中创建网站

演示步骤

1. 在开始屏幕上，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。

3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的**地球图标**，并在菜单中选择**中文(简体)**语言。
5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击**网站**。
6. 在页面下方的工具栏左侧，单击**新建**按钮。
7. 在弹出的**新建**向导中，单击**网站**，单击**自定义创建**。
8. 在弹出的**创建网站**窗口中，执行以下步骤：
 - a. 在 **URL** 文本框中，输入 **ws28532Demo[自定义名称]**。
 - b. 在**数据库**下拉框中，选择**创建新的 SQL 数据库**。
 - c. 在窗口右下角，单击**下一步**按钮。
9. 在**指定数据库设置**窗口中，执行如下步骤：
 - a. 在**名称**文本框中，修改其数值为 **EventsContextDBDemo**。
 - b. 在**服务器**下拉框中，选择**新建 SQL 数据库服务器**。
 - c. 在**服务器登录名**文本框中，输入 **testuser**。
 - d. 在**服务器登录密码**和**确认密码**文本框中，输入 **TestPa\$\$w0rd**。
 - e. 在**区域**下拉框中，选择一个离你位置较近的区域。
 - f. 在窗口右下角，单击**完成**按钮。
10. 选择新创建的网站，在页面下方的工具栏，单击**浏览**按钮。
11. 确认页面显示该网站已创建成功。
12. 关闭显示该网站的 IE 选项卡。

MCT USE ONLY. STUDENT USE PROHIBITED

第 2 课 在 Azure 中托管 Web 应用程序

在网站服务中部署 Web 应用程序的优势之一是，大家对托管方法比较熟悉。可以使用 Internet Information Server (IIS) 和 Web 部署包将应用程序托管在网站服务中，就像将应用程序托管在企业内部环境中一样。

本节描述网站实例的生命周期以及如何托管这些实例。

课程目标

完成本节后，您可以做到：

- 描述 IIS 与网站服务之间的关系。
- 描述网站实例的生命周期。
- 描述标准层网站实例与其他网站层之间的区别。

网站配置



GitHub 项目 Kudu

<https://github.com/projectkudu/kudu/wiki>

Kudu 是在 Azure 网站服务中实现 git 部署的幕后引擎。Kudu 可与本地 Web 应用程序项目一起使用。Kudu 项目是可在 GitHub 上获得的开源项目，并且支持很多功能，如：

- 从源代码控制系统发布 Web 应用程序
- 部署钩 (Deployment hook)
- Web 钩 (Web hook)
- Web 作业 (Web Job)

可以使用下面的 URL 格式访问您网站的 Kudu 控制台：

[https://\[网站名\].scm.azurewebsites.net](https://[网站名].scm.azurewebsites.net)

Kudu 还提供了可由 IIS 管理器来远程配置 Web 应用程序的端点。这样可以更安全地修改应用程序的 Web.config 文件。使用 KUDU 或 IIS 管理器端点管理 Web 应用程序不容易出错，这一点跟访问 Web 应用程序的文件系统，并编辑配置文件的传统方法不一样。



使用 IIS 管理器远程管理网站服务

<http://go.microsoft.com/fwlink/?LinkId=525353>

- 网站部署包及其配置都存储在外部存储中。
- 启动期间，应用程序设置和连接字符串在应用程序中被截获并更改
- 通过以下方式可以扩展应用程序：
 - 使用 Web 部署包创建 IIS 网站
 - 从外部存储应用配置选项

Web 宿主计划

可以将网站实例分组，以便在实例间共享容量。任何时候，网站实例只能与一个 Web 宿主计划关联。一个资源组可包含多个 Web 宿主计划。

Web 宿主计划与定价层关联。

- Web 宿主计划可逻辑地分组订阅中的网站。
 - 组中的网站实例共有相同的特征，如功能、容量和层。
- 一个资源组中可存在多个 Web 宿主计划，多个网站可以存在于一个 Web 宿主计划中

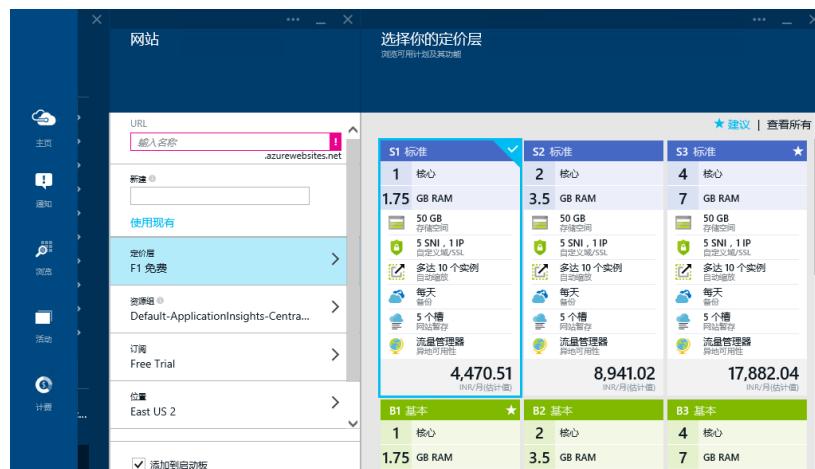


图 3.1: WEB 宿主计划定价层

在创建新的网站实例时，可以指定 Web 宿主计划，也可以使用为您选择的 Web 宿主计划。默认情况下，如果订阅没有任何 Web 宿主计划，那么系统会自动创建新的标准层 Web 宿主计划。如果订阅有现有的 Web 宿主计划，那么在创建新的网站实例时，默认情况下，将选中该计划。在创建新的网站实例时，也可以创建新的 Web 宿主计划。

Web 宿主计划中的所有网站将共同缩放。手动缩放设置或自动缩放设置是对整个 Web 宿主计划配置的。例如，可以将标准 Web 宿主计划配置为包含最少三个实例，最大五个实例，并通过监视磁盘队列深度指标来自动缩放。在一般负载情况下，预计可获得四个实例。这意味着有四个专用计算实例，所有网站实例在每个专用实例上都有一个副本，这些网站实例使用网站服务的内部负载平衡器达到负载平衡。

如何缩放网站

<http://go.microsoft.com/fwlink/?LinkId=525355>

第 3 课 配置 Azure 网站

Azure 的网站服务提供了很多新功能，可以使用这些新功能来扩展 Web 应用程序的功能。通过使用管理门户可以启用网站的不同功能，并修改网站的自定义设置，而无需重新部署 Web 应用程序。

本节列出可用于网站的不同配置选项。

课程目标

完成本节后，您可以做到：

- 描述 AlwaysOn 功能。
- 描述如何将自定义域名用于网站。
- 描述网站的自动缩放选项。

AlwaysOn

在 IIS 上托管 ASP.NET 应用程序时，可以将它们隔离开在应用程序池中。IIS 按计划自动回收应用程序池，并且只在处理第一个请求时，才运行 Web 应用程序的初始 ASP.NET 启动任务，从而提高 Web 服务器的性能。这样可防止应用程序在未得到客户端请求时使用不必要的资源。

如果 ASP.NET 应用程序没有预先编译，应用程序将在启动时实时 (JIT) 编译，然后 ASP.NET 启动任务才运行。可以使用 **HttpApplication** 类实现 Global.asax 文件来将您自己的逻辑注入 ASP.NET 的启动任务。如果 ASP.NET 应用程序已经预先编译，则启动任务将立即运行。无论选择何种方式，这些启动任务可能都很长，而且需要大量资源。这将造成第一个请求或者在应用程序池回收后立即发出的任何请求需要很长时间才能处理，而且比一般情况长得多。

ASP.NET 应用程序生命周期概述

<http://go.microsoft.com/fwlink/?LinkId=525358>

- 只适用于基本/标准层
- 非常适合连续 Web 作业
- 定期生成简单 HTTP 请求
 - 旨在作为一种检测信号，确保网站不回收应用程序池
 - 防止网站被卸载，并被迫在下次请求时重建。

在 IIS 中，将应用程序池的应用程序启动模式设置为“始终运行”可以解决此问题。在网站服务中，可以使用 AlwaysOn 功能达到同样的目的。AlwaysOn 防止应用程序因转入空闲状态而执行回收。AlwaysOn 还为早期客户端缩短了应用程序启动时间。这是由 Azure 平台实现的，Azure 平台定期 ping 您的网站，使其总是保持活动并处于运行状态。这确保应用程序在第一个客户端请求发出之前已在运行。这还确保应用程序一直处于运行状态，并启动程序以防回收。

只有基本层和标准层网站才有 AlwaysOn。

域名

在创建新网站时，系统会使用以下格式分配 azurewebsites.net 域的子域：

`http|https://<网站名>.azurewebsites.net`

`[http|https://<网站名>.chinacloudsites.cn (中国版 Windows Azure)`

Azure 还会为同一个网站实例分配一个虚拟 IP 地址。可以选择为网站使用自定义域名，并在门户中配置自定义域名。免费层网站实例不支持自定义域名。

若要查看网站实例的“管理自定义域”对话框，可以使用门户中的“配置”选项卡或“管理域”按钮。

“管理自定义域”配置对话框可用来将外部域与网站实例关联。

- 标准域名
`[http|https://<网站名>.azurewebsites.net`
`[http|https://<网站名>.chinacloudsites.cn (中国版 Azure)`
- 在共享、基本或标准模式下，可将网站配置为使用自定义域名
 - 这需要通过域名注册商管理 A 和 CNAME 记录
 - 流量管理器支持自定义域名



图 3.2：管理自定义域

此对话框包含如何将自定义域名和虚拟 IP 地址用于网站的说明。如果在一个网站中使用多个实例，虚拟 IP 地址将在这些实例之间平衡负载。此时，可以使用域名注册商的网站以及以下信息来配置规范名称 (CNAME) 记录和地址 (A) 记录：

主机	记录类型	IP 地址/URL
@	A	0.0.0.0 (门户中指定的 IP 地址)
www	CNAME	[网站名称].azurewebsites.net

完成这一步后，可以在 Azure 管理门户中使用同一个对话框来启用网站的自定义域名。

可以使用各种方法来配置网站的自定义域名。

<http://go.microsoft.com/fwlink/?LinkID=525359>

如果要托管跨地区的多个网站实例，可以搭配 Microsoft Azure 流量管理器来使用自定义域名。

还可以使用流量管理器来配置网站的自定义域名。

<http://go.microsoft.com/fwlink/?LinkID=525362>

自动缩放网站

在很多分布式应用程序场景中，您可能想通过增加应用程序实例的数量来向外（横向）扩展应用程序。使用内置的负载均衡器，可以将 Web 应用程序的负载分摊在多个实例上。这将最大程度减少每个实例成本，并确保应用程序满足客户端设备或浏览器的日益增长的需求。

由于应用程序工作负载难以预测，因此有时候，您可能最终高估或低估为提供最佳用户体验而需要的网站数。习惯上，过高的估计用来确保每个用户对 Web 应用程序都有称心如意的体验。理想情况下，需要让 Web 平台只在必要时才使用额外的实例，当不再需要这些实例时，就将它们关闭。

常见 Web 应用程序计算模式

- 缩放规则依计划而定
- 可使用各种指标配置性能缩放：

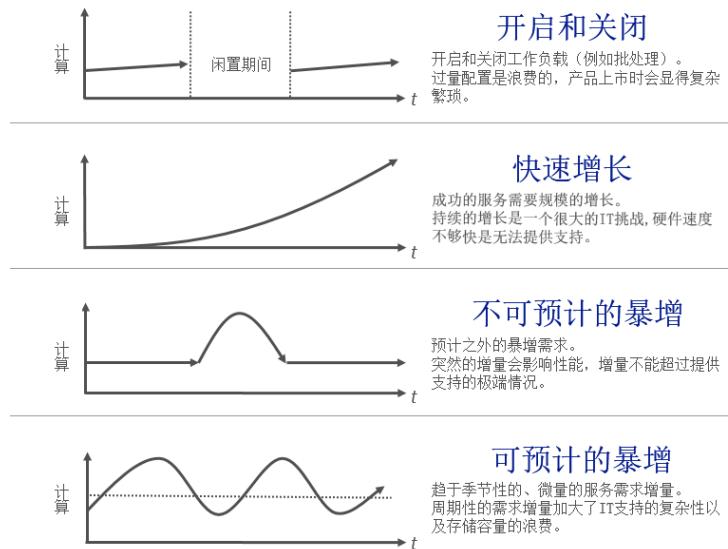


图 3.3：计算模式

可以使用自动缩放，通过指标和计划来控制水平缩放。这样您就能灵活地控制应用程序的资源分配，使其与实际使用量密切保持一致。通过使用自动缩放，您可以：

- 当不再需要网站实例时移除这些实例，从而最大程度减少不必要的资源成本。
- 在达到测量阈值时创建网站实例，从而最大程度提高性能，缩短客户端响应时间。

在定义自动缩放配置时，必须指定计划。可以使用计划为不同的日期和时间段指定不同的自动缩放规则。默认情况下，计划是针对所有时间创建的。在创建或选择计划后，可以定义指标来计量配置值，如：

- CPU 利用率（范围）
- 存储队列长度（阈值）

在保存此配置后，可以使用相同的配置界面监视网站的自动缩放历史记录。在预览门户中，有一个自动缩放指标的增强列表，可用来缩放网站，如：

- 平均内存
- HTTP 队列长度

MCT USE ONLY. STUDENT USE PROHIBITED

- 磁盘队列长度

还可以使用预览门户指定更多选项来确定 Web 应用程序需要等待多长时间再缩放，或者在两次缩放操作之间必须等待多长时间。

如何缩放网站

<http://go.microsoft.com/fwlink/?LinkID=525355>

演示 3：自动缩放网站

演示步骤

1. 登录到预览门户(<https://portal.azure.com>)。
2. 通过以下详细信息创建一个新网站实例：
 - Url: 为网站创建一个名称
 - Web 宿主计划名: **Autoscale**
 - Web 宿主计划模式: **S1 标准**
3. 查看新建网站边栏选项卡。
4. 查看**设置**边栏选项卡。
5. 查看网站的**缩放**选项。
6. 通过以下设置启用网站自动缩放功能：
 - 自动缩放模式: **性能**
 - 实例范围最小值: **2**
 - 实例范围最大值: **5**
 - CPU 百分比最小值: **50**
 - CPU 百分比最大值: **70**

演示 4：在中国版 Windows Azure 中自动缩放网站

演示步骤

1. 在开始屏幕上，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的**地球**图标，并在菜单中选择**中文(简体)**语言。
5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击**网站**。
6. 在网站页面，单击并进入新创建的网站 ws28532Demo[自定义名称]。



注释：若没有完成第一节中的 Demo，请先完成该 Demo 中网站的创建。

7. 在页面顶部，单击**缩放**选项卡。
8. 在 web 宿主计划模式下，单击**标准**。

3-12 在 Azure 平台上托管 Web 应用程序

9. 滚动屏幕并找到容量，在**按指标缩放**选项中，单击**CPU**。
10. 在**实例数**选项中，滑动滑块，设置最小数值为**2**，设置最大数值为**5**。
11. 在**目标 CPU**选项中，滑动滑块，设置最小数值为**50**，设置最大数值为**70**。
12. 在页面下方的工具栏中，单击**保存**按钮。如果弹出消息提示框，单击**是**。
13. 关闭 Internet Explorer 浏览器。

第 4 课 发布 Azure 网站

开发 Web 应用程序后，可以使用 Web 部署将 Web 应用程序发布到 Azure。通过使用 Visual Studio 中的发布向导，可以在发布 Web 应用程序前，自定义配置设置和连接字符串。

课程目标

完成本节后，您可以做到：

- 描述如何转换 Web 应用程序的配置设置。
- 描述标准发布版（standard Release）和调试版（Debug build）之间的区别。

Web 部署协议

Web 部署 (msdeploy) 是包格式和 IIS 加载项的组合，它能让管理员和开发人员非常灵活地管理和部署容器应用程序。

Web 部署包是 IIS 网站的简单表示形式，它可包含相关 Web 应用程序的以下数据：

- 二进制文件
- 内容
- XML 配置
- 数据库
- 注册表修改
- 程序集和全局程序集缓存 (GAC) 引用

- WebDeploy 提供了标准包格式，从而简化了将 Web 应用程序和网站部署到 IIS 服务器
 - 可以使用 IIS 管理器、命令行工具或 PowerShell 手动安装包
 - 可以使用 IIS 实例远程部署服务远程安装包
- Visual Studio 和 WebMatrix 可以将 Web 应用程序部署到 Web 部署端点

可以手动创建 Web 部署包，也可以使用 Visual Studio 或 WebMatrix 等 IDE 来创建。可以将包异步交给管理员，这样他或她就可在装有 Web 部署加载项的任何 IIS 中安装这些包。安装时，管理员为 SQL 数据库连接字符串之类的项提供配置值。您还可以使用 Web 部署在服务器场中同步应用程序的更改。可以从现有 IIS 网站中提取 Web 部署包，然后将其导入同一个服务器场中另一台计算机上的 IIS 网站。Web 部署还可以开放一个端点，让开发人员无需直接访问实际 Web 服务器，就可将应用程序远程部署到 Web 服务器。

可以使用 Web 部署包来同步开发、测试、暂存和生产环境，而不用考虑包托管在何处。

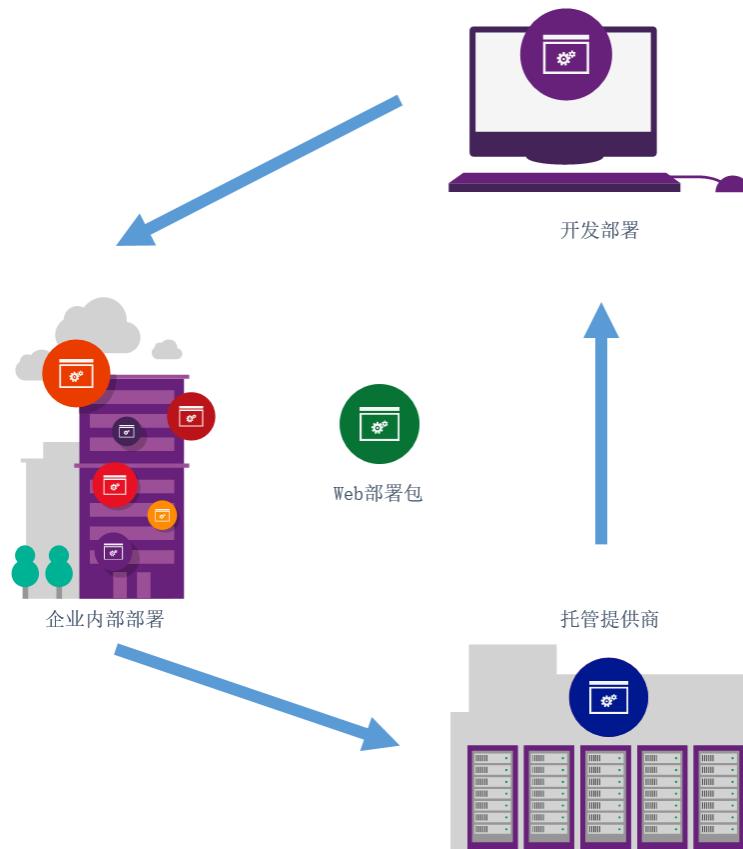


图 3.4: WEB 部署关系图

Azure 网站允许开发人员使用远程部署服务将 Web 部署包发布到网站实例。连接字符串和数据库已经得到管理，以提供无缝的发布体验。

使用 Web 部署来部署网站

<http://go.microsoft.com/fwlink/?LinkId=525363>

IIS Web 部署

<http://go.microsoft.com/fwlink/?LinkId=525366>

演示 5：管理应用设置和连接字符串

演示步骤

1. 登录到预览门户(<https://portal.azure.com>)。
2. 通过以下详细信息创建一个新网站实例：
 - **Url:** 为网站创建一个名称
3. 查看新建网站边栏选项卡。
4. 查看**设置**边栏选项卡。
5. 查看网站的**应用程序设置**选项。

6. 按以下值创建连接字符串:
 - 名称: entrycount
 - 值: 8
7. 按以下值创建应用程序设置:
 - 关键: externaldatabase
 - 值: URL=http://dataService;Data Format=JSON;Auto Cache=true;

演示 6：在中国版 Windows Azure 中管理网站的应用设置和连接字符串

演示步骤

1. 在开始屏幕上，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的**地球**图标，并在菜单中选择**中文(简体)**语言。
5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击**网站**。
6. 在网站页面，单击并进入新创建的网站 ws28532Demo[自定义名称]。
7. 在页面顶部，单击**配置**选项卡。
8. 滚动屏幕找到**应用设置**。
9. 在应用设置下方，添加如下键值信息:
 - a. 在密钥文本框中，输入 **externaldatabase**。
 - b. 在值文本框中，输入数字 **URL=http://dataService;Data Format=JSON;Auto Cache=true;**
10. 在连接字符串下方，添加如下键值信息:
 - a. 在名称文本框中，输入 **entrycount**。
 - b. 在值文本框中，输入 **8**。
11. 在页面下方的工具栏中单击**保存**按钮。
12. 关闭 Internet Explorer 浏览器。

MCT USE ONLY. STUDENT USE PROHIBITED

第 5 课 实验概述

本节扼要概述 Contoso Events Web 应用程序。本节中，您将看到“管理”Web 应用程序以及 Contoso Events Web 应用程序面向公众的 Web 前端的演示。

课程目标

- 完成本节后，您将理解 Contoso.Events Web 应用程序的工作方式。

演示 7：Contoso Events 演练

演示步骤

- 从以下位置打开 **Contoso.Events** 的解决方案：
 - 文件位置: **Allfiles (F):\Mod03\Democode\Contoso.Events**
- 定位到 **Contoso.Events.Data.Generation** 项目。
- 调试 **Contoso.Events.Data.Generation** 项目来生成 SQL 和 Azure 存储表中的数据。
- 关闭 **Microsoft Visual Studio 2013**。
- 用管理员身份启动 **Microsoft Visual Studio 2013**。
- 从以下位置打开 **Contoso.Events** 的解决方案：
 - 文件位置: **Allfiles (F):\Mod03\Democode\Contoso.Events**
- 启动 **Contoso.Events** 解决方案。
- 通过使用下面的启动项目调试解决方案：
 - 启动: **Contoso.Events.Cloud**
 - 开始执行（不调试）: **Contoso.Events.Management**
- 在 Contoso Events (Contoso.Events.Cloud) 网站打开后，使用 IIS Express 来打开 **Contoso.Events.Administration** 网站。
- 在 Contoso Events 网站选择任何 sales conference 事件，然后通过以下信息注册该事件：
 - Region: **Western**
 - AnnualSales: **45000**
 - FirstName: **Corrine**
 - LastName: **Horn**
- 在 Contoso Events Administration 网站的事件列表找到相应的事件。
- 生成该事件的 Sign-in Sheet。
- 下载生成后 Sign-in Sheet。

演示 8：在中文版 Azure 虚拟机中调试 Contoso Events 应用程序

演示步骤

- 在 Start 屏幕中，单击 **Desktop** 磁贴。
- 在任务栏上，单击 **File Explorer** 图标。

MCT USE ONLY. STUDENT USE PROHIBITED

3. 在 This PC 窗口，转去 **Allfiles (F):\Mod03\Democode\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中，展开 **Shared**。
5. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Data.Generation** 项目。
6. 在 Start 屏幕上，输入 **Azure Storage Emulator**。
7. 单击 **Windows Azure Storage Emulator – v3.4**。
8. 切换回 **Contoso.Events – Microsoft Visual Studio** 窗口。
9. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Data.Generation** 项目，指向调试，然后单击启动新实例。
10. 等待控制台应用程序调试结束。
11. 关闭 **Contoso.Events – Microsoft Visual Studio** 应用程序。
12. 在 Start 屏幕上，右键单击 Visual Studio 2013 磁贴，然后单击 Run as Administrator。
13. 在文件菜单上，指向打开，然后单击项目/解决方案。
14. 在打开项目对话框中，执行以下步骤：
 - a. 转到 **Allfiles (F):\Mod03\Democode\Contoso.Events**。
 - b. 单击 **Contoso.Events** 解决方案文件。
 - c. 单击 **Open**。
15. 在解决方案资源管理器窗格中，右键单击解决方案 **Contoso.Events**，然后单击属性。
16. 在解决方案 **Contoso.Events** 属性页对话框中，执行以下步骤：
 - a. 确认在屏幕左侧的导航菜单上选择通用属性=>启动项目。
 - b. 单击多启动项目。
 - c. 将 **Contoso.Events.Cloud** 的操作设置为启动。
 - d. 将 **Contoso.Events.Management** 的操作设置为开始执行（不调试）。
 - e. 确认所有剩余项目的操作设置为无。
 - f. 单击 **OK** 关闭对话框。
 - g. 确认 **Administration=>Contoso.Events.Management=>Web.config**、**Contoso.Events.Web=>Web.config**、**Contoso.Events.Worker=>app.config** 和 **Contoso.Events.Data.Generation=>App.config** 文件中 **connectionStrings** 中的 **Initial Catalog** 属性都设置为 **EventsContextModule3Lab**。
17. 在调试菜单上，单击启动调试。



注释：如果 Visual Studio 提示您重新启动 Azure emulator，单击 **OK**。

18. 在通知区域中，找到并右键单击 **IIS Express** 图标。
19. 指向 **View Sites => Contoso.Events.Management**，然后单击 **Browse Applications** 标题下的 URL。
20. 在 Internet Explorer 中，单击 **Home – Contoso Events** 选项卡。
21. 单击任何一个 sales conference 去到事件的详细信息的网页。

 **注释:** 如果 sales conference 没有在 Upcoming Events 中列出, 您可以单击 **All Events** 看到所有的事件。

22. 单击 **Register Now** 转到事件的注册页面。
23. 在 **Region** 框中, 输入 **Western**。
24. 在 **AnnualSales** 框中, 输入 **45000**。
25. 在 **FirstName** 框中, 输入 **Corrine**。
26. 在 **LastName** 框中, 输入 **Horn**。
27. 单击 **Submit**。
28. 在 Internet Explorer 中, 单击 **Home – Contoso Events Administration** 选项卡。
29. 单击 **Go to Events List**。
30. 找到在上一步中注册的 sales conference。
31. 在网格中, 单击事件名称旁边的 **Sign-In Sheet**。
32. 每隔 30 秒 (或更短时间) 刷新页面, 查看是否产生登记表。
当框体由蓝色转变为黄色, 并显示 **Sign-In Sheet** 的超链接, 表明登记表已经生成。
33. 单击 **Sign-In Sheet** 链接下载 Microsoft Word 文档。
34. 在 IE 下载对话框中, 单击 **Open** 按钮。
35. 关闭 **Internet Explorer** 应用程序。
36. 关闭 **Contoso.Events – Microsoft Visual Studio** 应用程序。

实验 A：使用 Azure 网站服务创建 ASP.NET 网站

场景

有一个事件管理应用程序目前正由一组固定的用户使用。应用程序必须升级才能在将来应对公司中的所有用户。您需要一种冲突最少的托管方案，以便可以立即部署 Web 应用程序并快速投入使用。您还需要该托管方案足够灵活，可以让您配置和缩放 Web 应用程序，从而确保可以应对管理用户人数的增加。出于上述原因，所以您选择将应用程序部署到网站。网站还可让您灵活地将应用程序与未来的 Azure Active Directory 集成，以便公司的所有用户都能访问该应用程序。

本实验中，您将创建网站、部署现有应用程序，然后在部署后配置网站。

 **注释：**本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版 Windows Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 创建网站。
- 创建网站带链接的资源。
- 将 ASP.NET 应用程序发布到网站。
- 在管理门户中修改网站的配置。

实验设置

预计时间：60 分钟

在开始这个实验之前，必须完成第 2 章的实验 A。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，输入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：
vm28532[自定义名称].cloudapp.net: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动输入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为**Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

MCT USE ONLY. STUDENT USE PROHIBITED

6. 本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版 Windows Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

练习 1：创建 Azure 网站

场景

在此练习中，您将：

- 在管理门户中创建网站实例。
- 访问新网站的 URL。

此练习的主要任务如下：

1. 使用管理门户创建网站实例

2. 转至新创建的网站占位符页面

► 任务 1：使用管理门户创建网站实例

1. 登陆到管理门户(<https://manage.windowsazure.com>)。
2. 创建一个新的自定义网站实例，详细步骤如下：



注释：确认记录下 SQL 管理员用户名和密码。如果信息丢失，会非常困难且需要大量时间来恢复该信息。

- Url：为网站创建一个名称
- 数据库：创建免费的 **20 MB SQL** 数据库
- 区域：选择最接近您所在位置的区域
- 数据库连接字符串名称：**EventsContextConnectionString**
- 数据库名称：**EventsContextDB**
- 数据库服务器：新建 **SQL** 数据库服务器

► 任务 2：转至新创建的网站占位符页面

1. 查看网站的仪表板。
2. 转至新建网站。

结果：完成此练习后，您就会使用管理门户来创建网站实例。

练习 2：将 ASP.NET Web 应用程序部署到 Azure 网站

场景

在此练习中，您将：

- 下载现有网站的发布配置文件。
- 使用 Visual Studio 2013 中的发布向导发布 Web 应用程序。

此练习的主要任务如下：

MCT USE ONLY. STUDENT USE PROHIBITED

1. 使用 Visual Studio 2013 打开现有的 ASP.NET Web 应用程序项目
 2. 下载 Azure 网站的发布配置文件
 3. 将 ASP.NET Web 应用程序发布到 Azure 网站
 4. 验证 Web 应用程序发布成功
- 任务 1：使用 **Visual Studio 2013** 打开现有的 **ASP.NET Web** 应用程序项目
1. 从以下位置打开 **Contoso.Events** 的解决方案：
 - 文件位置：**Allfiles (F):\Mod03\Labfiles\Starter\Contoso.Events**
 2. 调试 Web 应用程序来验证它正常工作。
- 任务 2：下载 **Azure** 网站的发布配置文件
1. 登陆到管理门户(<https://manage.windowsazure.com>)。
 2. 转到之前创建的网站的仪表板。
 3. 下载网站的发布配置文件
 - 保存位置：**Allfiles (F):\Mod03\Labfiles**
- 任务 3：将 **ASP.NET Web** 应用程序发布到 **Azure** 网站
1. 通过使用发布配置文件发布 ASP.NET Web 应用程序。
- 任务 4：验证 **Web** 应用程序发布成功
1. 确认 Azure 上的网站正常工作。

结果：完成此练习后，您就会使用发布配置文件将 Web 应用程序直接发布到网站上。

练习 3：配置 Azure 网站

场景

在此练习中，您将：

- 使用 **ConfigurationManager** 类从 Web 配置文件中读取自定义应用设置。
- 使用管理门户修改自定义应用设置。

此练习的主要任务如下：

1. 实现从应用设置中读取配置设置的逻辑
2. 将 Web 应用程序发布到 Azure 网站
3. 在管理门户中修改应用设置
4. 验证应用设置已成功更新

► 任务 1：实现从应用设置中读取配置设置的逻辑

1. 查看 **Contoso.Events.ViewModels** 项目里的 **EventsListViewModel** 类。
2. 在 **EventsListViewModel** 类的构造函数中，使用应用设置来更新 **EventCount** 属性的值：
 - 应用设置密钥：**EventsListViewModel.EventCount**
3. 添加一个应用设置到 **Contoso.Events.Management** 项目的 **Web.config** 的根目录下：
 - 应用设置密钥：**EventsListViewModel.EventCount**

- 应用设置值: **5**
- 4. 确认本地 Web 应用程序正在使用应用设置。

► **任务 2: 将 Web 应用程序发布到 Azure 网站**

1. 使用最新的发布配置文件发布 ASP.NET Web 应用程序。
2. 确认基于云计算的 Web 应用程序正在使用应用设置。

► **任务 3: 在管理门户中修改应用设置**

1. 登陆到管理门户(<https://manage.windowsazure.com>)。
2. 转到之前创建网站的**配置**选项卡。
3. 给网站配置添加一个**应用设置**入口, 重写 Web.config 文件中的应用设置。
 - 应用设置密钥: **EventsListViewModel.EventCount**
 - 应用设置值: **2**

► **任务 4: 验证应用设置已成功更新**

1. 确认在云中的 Web 应用程序正在使用更新的应用设置。

结果: 完成此练习后, 您就会使用 Web.config 文件和管理门户来配置自定义应用设置。

2. 通过在右列中放置标记来判断叙述的正确性。

叙述	答案
是非题: Web 部署是将 Web 应用程序部署到网站的唯一机制。	

实验 B：在中文版 Windows Azure 中创建 ASP.NET 网站

场景

有一个事件管理应用程序目前正由一组固定的用户使用。应用程序必须升级才能在将来应对公司中的所有用户。您需要一种冲突最少的托管方案，以便可以立即部署 Web 应用程序并快速投入使用。您还需要该托管方案足够灵活，可以让您配置和缩放 Web 应用程序，从而确保可以应对管理用户人数的增加。出于上述原因，所以您选择将应用程序部署到网站。网站还可让您灵活地将应用程序与未来的 Azure Active Directory 集成，以便公司的所有用户都能访问该应用程序。

本实验中，您将创建网站、部署现有应用程序，然后在部署后配置网站。

目标

完成本实验后，您将能够：

- 创建网站。
- 创建网站带链接的资源。
- 将 ASP.NET 应用程序发布到网站。
- 在管理门户中修改网站的配置。

预计时间：60 分钟

在开始这个实验之前，必须完成第 2 章的**实验 B**。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，输入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].chinacloudapp.cn: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动输入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到 Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：创建 Azure 网站

场景

在此练习中，您将：

- 在管理门户中创建网站实例。

MCT USE ONLY. STUDENT USE PROHIBITED

- 访问新网站的 URL。

此练习的主要任务如下：

1. 在 Azure 管理门户中创建网站实例
2. 访问新创建的空网站

► **任务 1：在 Azure 管理门户中创建网站实例**

1. 在 Start 屏幕上，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的**地球图标**，并在菜单中选择**中文(简体)**语言。
5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击**网站**。
6. 在页面下方的工具栏左侧，单击**新建**按钮。
7. 在弹出的新建向导中，单击**网站**，单击**自定义创建**。
8. 在弹出的**创建网站**窗口中，执行以下步骤：
 - a. 在 **URL** 文本框中，输入 **ws28532[自定义名称]**。
 - b. 在 **Web 宿主计划**列表中，选择**创建新的 Web 宿主计划**。
 - c. 在**区域**下拉框中，选择一个离你位置较近的区域。
 - d. 在**数据库**下拉框中，选择**创建新的 SQL 数据库**。
 - e. 在**数据库连接字符串名称**文本框中，修改其数值为 **EventsContextConnectionString**。
 - f. 在窗口右下角，单击**下一步**按钮。
9. 在**指定数据库设置**窗口中，执行如下步骤：
 - a. 在**名称**文本框中，修改其数值为 **EventsContextDB**。
 - b. 在**服务器**下拉框中，选择**新建 SQL 数据库服务器**。
 - c. 在**服务器登录名**文本框中，输入 **Student**。
 - d. 在**服务器登录密码**和**确认密码**文本框中，输入 **TestPa\$\$wOrd**。
 - e. 在**区域**下拉框中，选择一个离你位置较近的区域。
 - f. 在窗口右下角，单击**完成**按钮。

► **任务 2：访问新创建的空网站**

 **注释：** 创建网站不会花费太多时间，请耐心等待直到网站创建完成，这时，网站的状态会显示为正在运行。

1. 在网站页面，单击进入您新创建的网站。
2. 在新建的网站页面，单击页面上方的**仪表板**选项卡。
3. 滚动屏幕找到**速览**，在**速览**下方，找到**站点 URL**，单击下方的站点链接（链接格式如[website url].chinacloudsites.cn）。
4. 确认页面显示该网站已创建成功。
5. 关闭显示该网站的 IE 选项卡。

结果：完成此练习后，您就会使用管理门户来创建网站实例。

练习 2：将 ASP.NET Web 应用程序部署到 Azure 网站

场景

在此练习中，您将：

- 下载现有网站的发布配置文件。
- 使用 Visual Studio 2013 中的发布向导发布 Web 应用程序。

此练习的主要任务如下：

1. 在 Visual Studio 2013 中打开 ASP.NET Web 应用程序项目
2. 下载 Azure 网站的发布配置文件
3. 发布 ASP.NET Web 应用程序到 Azure 网站
4. 验证 Web 应用程序发布成功

► 任务 1：在 Visual Studio 2013 中打开 ASP.NET Web 应用程序项目

1. 在 Start 屏幕中，单击 **Desktop** 磁贴。
2. 在任务栏上，单击 **File Explorer** 图标。
3. 在 This PC 窗口，转去 **Allfiles (F):\Mod03\Labfiles\Starter\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Management** 项目，然后单击 **设为启动项目**。
5. 在调试菜单上，单击 **启动调试**。

 **注释：**如果这是您第一次生成此解决方案，NuGet 将隐式地恢复所有丢失的包。您不必手动恢复丢失的包。

6. 在 Web 应用程序的主页，确认在 **Latest 3 Events** 标题下，显示三个事件。
7. 在网页顶部的导航栏中，单击 **Events**。
8. 确认在事件页面中显示事件列表。
9. 关闭显示该网站的选项卡。

► 任务 2：下载 Azure 网站的发布配置文件

1. 在 Start 屏幕上，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的地球图标，并在菜单中选择中文(简体)语言。
5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击 **网站**。
6. 在网站页面，单击并进入新创建的网站 **ws28532[自定义名称]**。
7. 单击页面上方的 **仪表板** 选项卡。
8. 滚动屏幕找到 **速览**，在 **速览**下方，单击 **下载发布配置文件** 链接。

9. 在 IE 下载对话框中，单击 Save 按钮右侧的箭头，然后单击 **Save As**。
10. 在 **Save As** 对话框中，定位到 **Allfiles (F):\Mod03\Labfiles**，然后单击 **Save**。
▶ 任务 3：发布 ASP.NET Web 应用程序到 Azure 网站
 1. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在**解决方案资源管理器**窗格中，右键单击 **Contoso.Events.Management**，然后单击发布。
 2. 在发布网页窗口中，单击导入。
 3. 在导入发布设置对话框中，单击浏览。
 4. 在导入发布设置窗口中，转到 **Allfiles (F):\Mod03\Labfiles**，然后双击之前保存的发布配置文件。
 5. 单击确定。
 6. 确认**站点名称**对话框中的名字匹配网站的名字。
 7. 单击发布。

 **注释：**如果弹出保护只读文件消息框，单击覆盖。

▶ 任务 4：验证 Web 应用程序发布成功

 **注释：**如果在**练习 1 – 任务 2**显示占位符页，这可能意味着客户端正在缓存中。您可以随时按 Ctrl+ F5 强制刷新浏览器和缓存。

1. 在 Web 应用程序的主页，确认在 **Latest 3 Events** 标题下，显示三个事件。
2. 在网页顶部的导航栏中，单击 **Events**。
3. 确认在事件页面中显示事件列表。
4. 关闭显示该网站的选项卡。

结果：完成此练习后，您就会使用发布配置文件将 Web 应用程序直接发布到网站上。

练习 3：配置 Azure 网站

场景

在此练习中，您将：

- 使用 **ConfigurationManager** 类从 Web 配置文件中读取自定义应用设置。
- 使用管理门户修改自定义应用设置。

此练习的主要任务如下：

1. 实现从应用设置中读取配置信息
2. 发布 Web 应用程序到 Azure 网站
3. 在 Azure 管理门户中修改应用设置
4. 验证应用设置被成功更新

▶ 任务 1：实现从应用设置中读取配置信息

1. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在**解决方案资源管理器**窗格中，展开 **Contoso.Events.ViewModels** 项目。

MCT USE ONLY STUDENT USE PROHIBITED

2. 在解决方案资源管理器窗格中，双击 **EventsListViewModel.cs**。

3. 定位到位于 EventsListViewModel.cs，第 20 行的以下代码：

```
this.EventCount = 3;
```

4. 用以下的代码替换之前那行代码：

```
this.EventCount =
Int32.Parse(ConfigurationManager.AppSettings["EventsListViewModel.EventCount"]);
```

5. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在解决方案资源管理器窗格中，展开 **Contoso.Events.Management** 项目。

6. 在解决方案资源管理器窗格中，双击 **Web.config**。

7. 在第 14 行，**appSettings** 元素中，添加以下代码：

```
<add key="EventsListViewModel.EventCount" value="5" />
```

8. 在调试菜单上，单击启动调试。

9. 在 Web 应用程序的主页，确认在 **Latest 5 Events** 标题下，显示五个事件。

10. 关闭显示该网站的选项卡。

► **任务 2：发布 Web 应用程序到 Azure 网站**

1. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Management**，然后单击发布。

2. 单击发布。

3. 在 Web 应用程序的主页，确认在 **Latest 5 Events** 标题下，显示五个事件。

4. 关闭显示该网站的选项卡。

► **任务 3：在 Azure 管理门户中修改应用设置**

1. 在 Start 屏幕上，单击 **Internet Explorer** 磁贴。

2. 访问 <https://manage.windowsazure.cn>。

3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。

4. 如果您的界面语言不是中文，可以单击页面右上角的地球图标，并在菜单中选择中文(简体)语言。

5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击网站。

6. 在网站页面，单击并进入新创建的网站 **ws28532[自定义名称]**。

7. 单击配置选项卡。

8. 滚动屏幕找到应用设置。

9. 在应用设置下方，添加如下键值信息：

- a. 在密钥文本框中，输入 **EventsListViewModel.EventCount**。

- b. 在值文本框中，输入 **2**。

- c. 在页面下方的工具栏中，单击 **保存** 按钮。



注释：更新配置设置时，需要等待几秒钟使更改生效。当编辑框再次可使用时，表明更改已经生效。

► 任务 4：验证应用设置被成功更新

1. 单击页面上方的**仪表板**选项卡。
2. 滚动屏幕找到**速览**，在**速览**下方，找到**站点 URL**，单击下方的站点链接（链接格式如[website url].chinacloudsites.cn）。
3. 在 Web 应用程序的 home 页面，在**Latest 2 Events**下方的列表中显示两条 Events 数据。
4. 关闭显示该网站的选项卡。
5. 关闭**Internet Explorer** 应用程序。
6. 关闭**Contoso.Events - Microsoft Visual Studio** 应用程序。

结果：完成此练习后，您就会使用 Web.config 文件和管理门户来配置自定义应用设置。

7. 通过在右列中放置标记来判断叙述的正确性。

叙述	答案
是非题：Web 部署是将 Web 应用程序部署到网站服务的唯一机制。	

章节复习和作业

本章中，您了解了网站服务以及将 ASP.NET Web 应用程序项目迁移到云有多简单。您还了解了一些独特的配置选项和功能，这些选项和功能使得网站服务以一种真正强大的方式来托管基于云的 Web 应用程序。最后，您还了解了网站服务直接提供的监视工具，可用它们来调试和跟踪 Azure 中的网站。

 **最佳做法：**若要将应用程序部署到企业内部环境，可以遵循使用 Web 部署将 Web 应用程序发布到 Azure 的同样的过程。Web 部署可使开发人员脱离发布过程。它提供了一个单独的包，管理员可使用此包在生产环境中部署应用程序。

复习问题

问题：在哪些业务场景下可以按计划缩放网站？

问题：为什么要将多个网站实例的日志存储在中心位置？

问题：为什么会考虑将网站实例的日志文件保留在文件系统中？

MCTI SE ONLY. STUDENT USE PROHIBITED

第 4 章 在 Azure 中存储 SQL 数据

目录:

章节概述	4-2
第 1 课: Azure SQL 数据库概述	4-3
第 2 课: 管理 Azure 中的 SQL 数据库	4-7
第 3 课: Azure SQL 数据库工具	4-10
第 4 课: 保护和恢复 Azure SQL 数据库实例	4-12
实验 A: 在 Azure SQL 数据库中存储事件数据	4-14
实验 B: 使用中国版 Windows Azure 在 Azure SQL 数据库中存储事件数据	4-18
章节复习和作业	4-25

章节概述

动态 Web 应用程序必须要存储很多由用户产生和维护的数据，如博客程序需要存储用户发的博文信息。ADO.NET 和实体框架（Entity Framework）之类的 ASP.NET 技术提供了访问 SQL Server 数据的方法。在云中，Microsoft Azure 平台提供了数据库即服务产品，可以让开发人员像在企业内部那样使用 SQL。第 1 节“Azure SQL 数据库概述”，描述了 Azure SQL 数据库服务以及考虑使用此服务时的原因。第 2 节“管理 Azure 中的 SQL 数据库”，描述了可用于处理托管在 Azure 中的 SQL 数据库的常见管理工具和新管理工具。第 3 节“Azure SQL 数据库工具”，描述了 Microsoft Visual Studio 2013 中提供的 SQL Server 数据工具 (SSDT) 模板、窗格和项目。第 4 节“保护和恢复 Azure SQL 数据库实例”，描述了 Azure SQL 数据库相关恢复方案。

目标

完成本章后，您可以做到：

- 描述 Azure SQL 数据库各版本之间的区别。
- 阐述 Azure SQL 数据库中托管数据库的部分优缺点。
- 阐述在 Azure 虚拟机上的 SQL Server 安装中托管数据库的部分优缺点。
- 描述可用来管理 Azure SQL 数据库的多种工具。
- 利用 Azure SQL 数据库实现高可用性解决方案。

第 1 课 Azure SQL 数据库概述

SQL 数据库提供的数据库即服务产品可让您利用 SQL Server 开发人员和管理员所熟悉的很多功能。SQL 数据库开放表格格式数据流(TDS)端点，使您可以使用很多现有工具来连接和管理 SQL 数据库实例。

本节描述 SQL 数据库服务、使用此服务的部分优点，以及在 SQL 数据库服务和 Azure 虚拟机上的 SQL Server 之间进行选择的注意事项。

课程目标

完成本节后，您可以做到：

- 描述 SQL 数据库服务。
- 说明使用 SQL 数据库的优点。
- 说明使用安装在 Azure 虚拟机上的 SQL Server 的优点。
- 比较服务层。
- 描述 SQL 数据库的版本。

Azure SQL 数据库

SQL 数据库是一种关系型数据库即服务产品，它提供了可预测的性能以及与现有管理工具的高度兼容性。

可预测的性能

使用统一的计量单位（如数据库吞吐量单位），可以比较 SQL 数据库服务中提供的每个性能层应有的服务级别。根据一致且可预测的性能，您就可以选择与应用程序的实际使用情况非常接近的服务层。

- 完全托管的数据库解决方案
- 与现有管理工具高度兼容
- 内置高可用性以及扩展时可预测的性能

高可用性

SQL 数据库服务中创建的每一个逻辑服务器都提供了表格格式数据流(TDS)端点*。利用 TDS 协议，可以使用现有 SQL 客户端应用程序和工具来处理 SQL 数据库。

简单管理

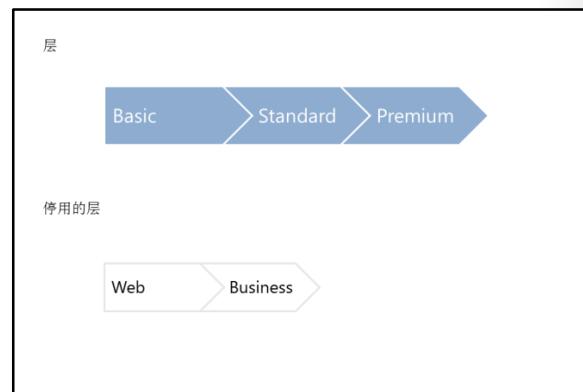
Azure 中提供了额外的工具来管理由 SQL 数据库创建的数据库。Azure 管理门户中有用来管理数据库对象的门户，单击“管理”按钮可以访问该门户。此外还可以使用门户、REST API、Windows PowerShell 或跨平台命令行接口(Xplat CLI)来管理 SQL 数据库实例。

*端点(endpoint)有时在 Azure 界面中也译作端点。

Azure SQL 数据库层

SQL 数据库服务分成几个层级供用户选择。可以选择与应用程序的目标或实际资源需求接近的层。以下是具有相关性能特征的 SQL 数据库服务层的列表：

- **Basic:** 适合只需要一个连接，一次只执行一个操作的简单数据库。
- **Standard:** 最常用的选项，用于需要多个并发连接和操作的数据库。
- **Premium:** 专为需要大事务量的应用程序而设计。这些数据库支持大量并发连接和并行操作。



这些层又进一步分成性能级别。性能级别是服务层内非常具体的类别，每个服务层都提供具体的服务级别。例如，Premium 层内的 P1 性能级别提供最大 500 千兆字节(GB)的数据库大小以及每秒 105 个事务的基准事务率。

数据库吞吐量单位 (DTU)

DTU 是用来描述特定层和性能级别容量的。DTU 设计为相对值，便于直接比较各个层和性能级别。例如，Basic 层只有一个额定为 5 个 DTU 的性能级别(B)。Standard 层中的 S2 性能级别额定为 50 个 DTU。这意味着 S2 性能级别数据库的处理能力有望达到 Basic 层 B 性能级别数据库的十倍。

层和性能级别

每一层有一个或多个性能级别。一般来说，Premium 层里的性能级别额定值高于 Standard 层里的性能级别额定值，Standard 层里的性能级别额定值又高于 Basic 层里的性能级别额定值。下图显示了这一区别。

Azure SQL 数据库服务层和性能级别

<http://go.microsoft.com/fwlink/?LinkId=525368>

SQL 数据库定价

<http://go.microsoft.com/fwlink/?LinkId=525370>

	数据库吞吐量单位	数据库大小	按时间点还原
B	5	2 GB	7 天
S0	10	250 GB	14 天
S1	20	250 GB	14 天
S2	50	250 GB	14 天
S3	100	250 GB	14 天
P1	100	500 GB	35 天
P2	200	500 GB	35 天
P3	800	500 GB	35 天

数据库即服务与虚拟机中的 SQL Server

人们经常会将 SQL 数据库服务和托管在物理机或虚拟机上的 SQL Server 单机版作比较。如果选用 SQL 数据库服务，那么大部分管理任务将完全由 Microsoft 管理。但是，如果需要能执行非常独特的自定义配置、分析或管理，而 SQL 数据库服务无法满足这些特殊要求。这种情况下，则可以选择 SQL Server 单机版。可以在 Azure IaaS 平台的虚拟机中托管 SQL Server 单机版产品。

托管在 Microsoft 数据平台中的 SQL

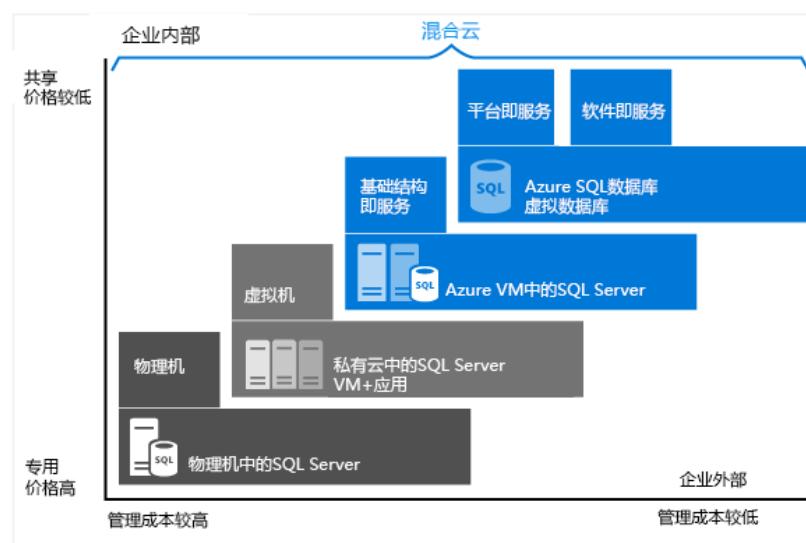
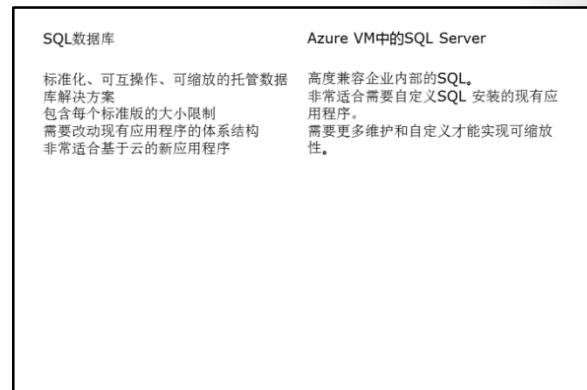


图 4.1: SQL 数据平台

在 SQL 数据库中托管数据

- 这种解决方案非常适合基于云技术的新应用程序，这类应用程序不需要自定义 SQL Server 的行为。很多应用程序将 SQL Server 用于存储、编程功能和基本查询操作。
- 存储在 SQL 数据库中的数据库大小受限制。Elastic Scale 之类的解决方案可用于跨多个不同层的实例缩放数据库。

在 Azure 虚拟机上的 SQL Server 单机版中托管数据

- 此解决方案与托管在企业内部位置的现有 SQL 应用程序高度兼容。通过上传虚拟硬盘，可以将安装的 SQL Server 从企业内部位置迁移到 Azure。因此，如果希望不加修改就迁移现有数据库工作负载，那么可以使用此解决方案。
- 使用 Azure 虚拟机上的 SQL Server 单机版有利于一些现存的应用程序，这些程序往往使用了自定义的 SQL 安装方式。例如，SQL Server Analysis Services (SSAS) 不可用于 SQL 数据库服务。如果在 Azure 虚拟机上安装 SQL 单机版，那就可以启用配合 SSAS 使用的服务。
- 如果要启用数据还原方案，则必须手动配置 SQL Server AlwaysOn。

在 Azure 中托管 SQL Server 单机版最常见的原因之一是为了应对整体迁移（lift and shift）场景。整体迁移工作负载是将应用程序从一个平台迁移到另一个平台，而尽可能少地更改其源代码或配置。将现有 SQL 工作负载迁移到 Azure IaaS 要比将应用程序迁移到 SQL 数据库容易得多。迁移后，就可分析现有 SQL 工

4-6 在 Azure 中存储 SQL 数据

作负载，并确定与 SQL 数据库的兼容程度。对于新应用程序，SQL 数据库几乎无需维护，从而缩短了新创建的应用程序投入市场的时间。

理解 SQL 数据库与 Azure 虚拟机上运行的 SQL Server

<http://go.microsoft.com/fwlink/?LinkId=525373>

MCT USE ONLY STUDENT USE PROHIBITED

第 2 课 管理 Azure 中的 SQL 数据库

管理门户提供了创建和管理数据库的便捷方式。还有一个独特的门户可用来管理 SQL 数据库的表、视图及其他对象。

本节列出可用来与 Azure 中的 SQL 数据库交互的各种在线管理工具。

课程目标

完成本节后，您可以做到：

- 使用管理门户创建服务器和数据库。

创建 Azure SQL 数据库实例

可以使用门户、服务管理 REST API、Windows PowerShell 或 Visual Studio 2013，在 SQL 数据库服务中创建新数据库实例。本节后面会提供屏幕截图来演示如何在管理门户中创建数据库，此外还包含一个演示来说明如何在 Azure 预览门户中创建数据库。

在创建数据库实例之前，必须创建逻辑服务器。此服务器必须有全局唯一名称，该名称将通过使用 TDS 协议来连接数据库。服务器会有一个使用 SQL Server 身份验证创建的管理员帐户。目前 Windows 身份验证是不被支持的。创建逻辑服务器后，就可以在服务中创建数据库实例。数据库名称无需全局唯一。

使用管理门户创建新数据库

为了逻辑分组数据库，可以使用管理门户创建服务器。

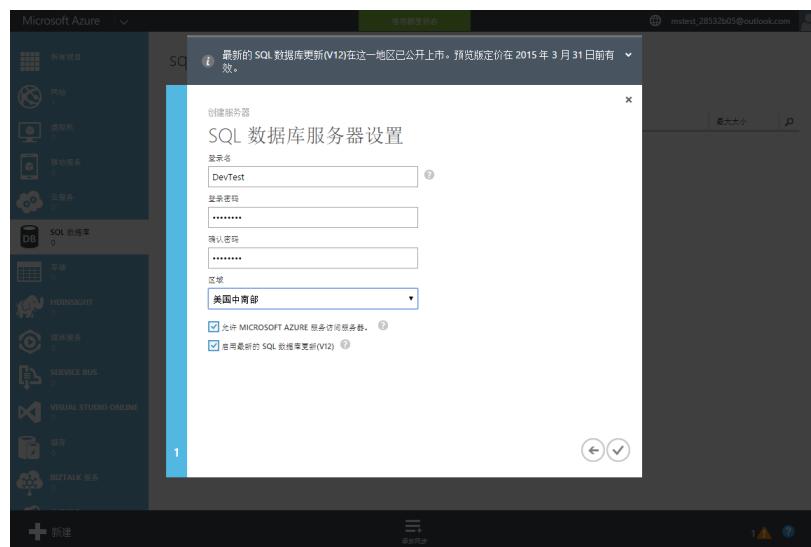
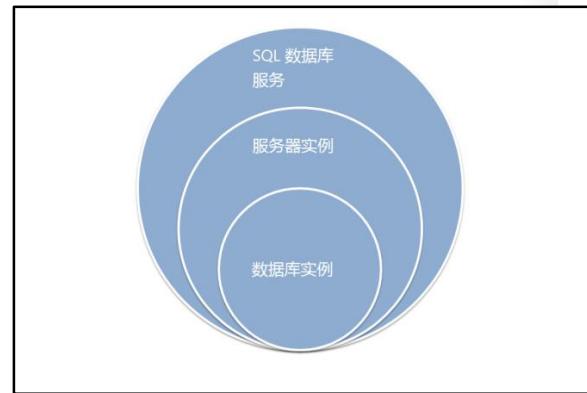


图 4.2：新建 SQL SERVER

还可以通过管理门户来创建数据库。创建数据库时，可以将其放入现有服务器，也可以创建新服务器。

4-8 在 Azure 中存储 SQL 数据

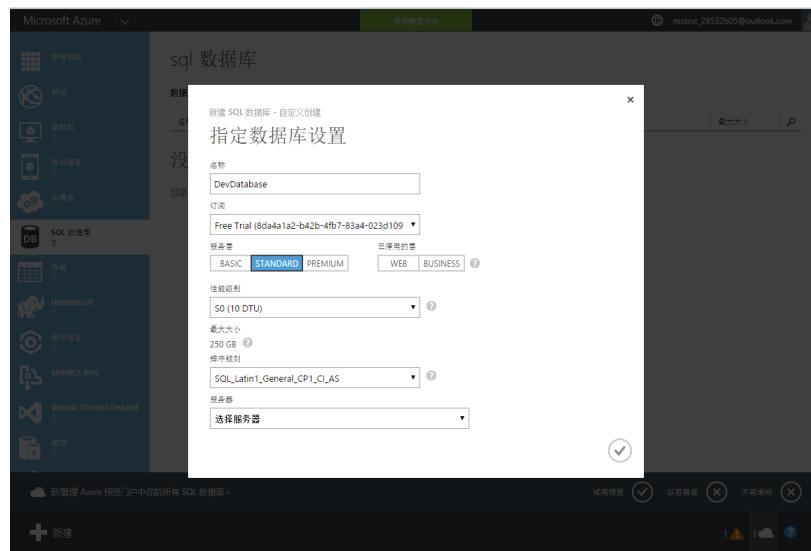


图 4.3：新建 SQL 数据库

管理门户 - SQL 数据库

当前管理门户为 SQL 数据库实例提供了独特的基于 Visual Studio 的门户。可以使用此门户来查看数据库的某些简单分析数据，以及以可视化方式查询管理数据库对象。

TDS 协议

<http://go.microsoft.com/fwlink/?LinkId=525378>

Azure SQL 数据库管理门户

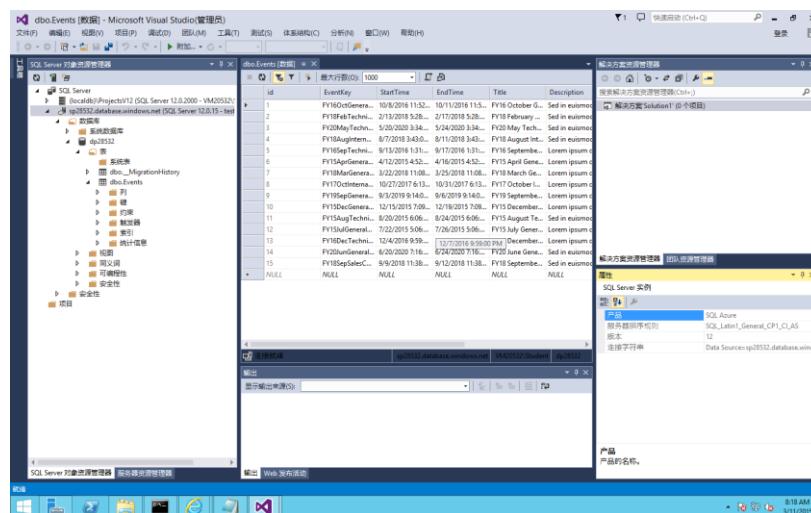


图 4.4：AZURE SQL 数据库管理门户

演示 1：使用管理门户创建 Azure SQL 数据库实例

演示步骤

- 登录到 Azure 预览门户网站 (<https://portal.azure.com>)。
- 将语言设置为中文（简体）。

3. 通过以下信息创建一个 SQL 数据库：
 - 名称: **SQLDemo**
 - 定价层: **Basic**
 - 服务器名: **唯一值**
 - 服务器管理登录: **testuser**
 - 服务器管理密码: **TestPa\$\$w0rd**
 - 位置: **选择最接近你所在位置的区域**

演示 2：在中国版 Windows Azure 的管理门户中创建 Azure SQL 数据库实例

演示步骤

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 进入 <https://manage.windowsazure.cn>。
3. 在邮箱帐户输入框，输入你的邮箱帐户地址；在密码输入框输入密码。
4. 单击 **Sign In** 登录。
5. 单击右上角的帐户名前的地球图标，选择中文（简体）选项。
6. 在管理界面左下角，单击**新建**。
7. 在**新建**窗格下，首先在左侧栏目中单击**数据服务**，而后在中间弹出的选项中选择**SQL 数据库**，然后在右边弹出的选项中选择**自定义创建**。
8. 在**新建 SQL 数据库**的设置页面中，请按照如下步骤创建 SQL 数据库实例：
 - a. 在名称输入框输入 **SQLDemo**。
 - b. 在**服务层**选项中，选择 **BASIC**。
 - c. 在**服务器**选项中，选择**新建 SQL 数据库服务器**选项，其余选项保持默认值，单击右下方下一步按钮→进入**创建服务器**界面。
 - d. 在**登录名**输入框，输入 **testuser**。
 - e. 在**登录密码**和**确认密码**输入框，输入 **TestPa\$\$w0rd**。
 - f. 在**区域**选项中，选择一个区域，其余保持默认选项。
 - g. 最后单击右下方按钮√完成 SQL 数据库实例的创建。

第 3 课 Azure SQL 数据库工具

Azure 中 SQL 数据库的优点之一是，能够使用很多可用于企业内部数据库的监视工具。

本节描述了现有管理工具，以及如何使用这些工具来管理 Azure 中的 SQL 数据库。

课程目标

完成本节后，您可以做到：

- 使用 SQL Server Management Studio 管理 SQL 数据库实例。
- 使用 Visual Studio 工具管理 SQL 数据库实例。

SQL Server Management Studio

使用 TDS 协议最大的好处之一是，它提供了用来连接 SQL 数据库和服务器的多种工具。由于 SQL 数据库服务完全支持 TDS 协议，因此客户端工具和应用程序无需关心实际体系结构的实现。这样，您就可以使用本地 SQL 安装（如 LocalDb）来开发和测试生产用的 SQL 数据库。可以适当地更换连接字符串来实现这种转换。

设置

在配置 SQL 数据库服务器实例时，需要指定用户名和密码。此登录名是 SQL 数据库服务器的服务器级主体。此登录名类似于 SQL Server 单机版中的 sa 主体。可以随时使用服务器上创建的主数据库来管理数据库中的登录名和角色。可以使用标准 T-SQL 查询来创建更多登录名。

可以使用 CREATE LOGIN、ALTER LOGIN 或 DROP LOGIN 语句来管理 SQL 数据库中的登录名。

管理登录名

```
CREATE LOGIN login1 WITH password='<ProvidePassword>' ;
```

配置服务器后，还必须配置以下访问规则：

- **防火墙**。可以配置防火墙来允许从预定义的 IP 地址范围列表访问 SQL 数据库。不在此范围内的任何 IP 地址将无法连接到 TDS 端点。
- **Azure 服务访问**。可以配置布尔配置选项来指定其他 Azure 服务是否可以访问 SQL 数据库中的 TDS 端点。

Visual Studio

Visual Studio 提供了多种方法来管理 SQL Server 单机版和 SQL 数据库实例。可以使用“服务器资源管理器”窗格来连接和管理 SQL 数据库。还可以使用“服务器资源管理器”窗格来管理各种 Azure 服务，包括 SQL 数据库。可以使用 SQL Server 对象资源管理器来管理单个数据库和整个服务器。可以使用 Visual Studio 数据库项目以声明方式设计数据库并发布生成的脚本。可以修改这些项目的选项，使生成的脚本与 SQL 数据库服务兼容。

SQL Server Management Studio

SQL 数据库中的每个逻辑服务器都开放了 TDS 端点。这样您就可以像使用 SQL Server Management Studio 处理 SQL Server 单机版一样来处理 SQL 数据库。

迁移工具

SQL 迁移脚本

可以使用 SQL Server Management Studio 中的“生成脚本”任务来生成迁移脚本。

- 可以将生成的脚本文件复制到剪贴板、在脚本窗口中打开，或者保存到文件。
- 这是一种可与不同版本的 SQL Server 高度兼容的简单方法。
- 脚本运行需要花一点时间，需要使用工具在目标数据库上运行这些脚本。

- 可以使用 SQL Server Integration Services 来定义企业内部数据库的迁移计划
- SQL 数据库迁移向导分析现有数据库，并生成脚本（和批量复制文件）来迁移数据库
- Azure 网站迁移助手使用 SQL 管理对象分析现有数据库并进行迁移

导入和导出数据层应用程序

可以将数据层应用程序从 SQL 数据库中导出，然后将其导入其他数据库。

- 数据层应用程序包含数据库中对象的定义。
- 应用程序包还包含存储在不同数据库对象中的数据。
- 可以将数据层应用程序导入 Azure 中的 SQL 数据库。

新发布的很多工具可用来分析现有数据库、创建迁移策略，然后最终执行迁移操作。这些工具包括 SQL Azure 迁移向导（Migration Wizard）和 Azure 网站迁移向导。

SQL Server Integration Services (SSIS)

可以使用 SSIS 来规划和执行从 SQL Server 单机版安装到 Azure 中 SQL 数据库的数据迁移。在 SQL 数据库服务不支持现有数据库所用的功能时，这种方法非常适合。可以将架构和数据迁移到或重新映射到可托管在托管服务中的对象。

SQL Server Integration Services

<http://go.microsoft.com/fwlink/?LinkId=525379>

Microsoft Azure SQL 数据库迁移向导

此迁移向导也称为 SQL Azure MW，它是专门为分析现有数据库和最终执行 SQL 数据库的迁移而设计的。这些工具支持从 SQL Server 2005 开始的多种 SQL 版本。实际迁移过程是这样执行的：先生成一个复制 SQL 数据库架构的脚本，然后使用 BCP 实用工具将从原始位置导出的数据文件中的数据复制到目标。

SQL 数据库迁移向导

<http://go.microsoft.com/fwlink/?LinkId=525380>

SQL Bulk Copy 实用工具

<http://go.microsoft.com/fwlink/?LinkId=525381>

Azure 网站迁移助手

Azure 网站迁移助手是专门为 IIS 应用程序从企业内部迁移到 Azure 设计的。此工具还使用连接字符串来迁移与 IIS 网站关联的 SQL 数据库。这种迁移是使用 SQL Server 管理对象生成的脚本来执行的。

使用迁移助手将 IIS 网站迁移到 Azure 网站

<http://go.microsoft.com/fwlink/?LinkId=525382>

第 4 课 保护和恢复 Azure SQL 数据库实例

SQL Server 单机版和 SQL 数据库都提供了完整的复制和灾难恢复选项。

本节介绍了 SQL Server 单机版和 Azure 中的 SQL 即服务的高可用性功能。

课程目标

完成本节后，您可以做到：

- 描述 Azure 中 SQL 数据库的异地复制选项。
- 描述 SQL Server 的高可用性功能。

Azure SQL 数据库的恢复选项

业务连续性描述了当发生不可预见的事件时，保持业务运作持续运转所需的规划和执行过程。通常，数据库都需要经过规划、设计和实施这几个步骤。Azure SQL 数据库为数据库提供了很多业务连续性功能。Azure SQL 数据库基础结构还使用复制来实现应用程序在其自己的基础结构中的基本故障转移功能。

- 对于大多数基本场景，可以依靠 Azure 数据中心的基础结构冗余
- SQL 数据库也提供了自己的一组 HADR 选项
 - 内置副本**
 - 事务要在写入目标数据库，一个主副本和两个辅助副本之后才会考虑提交到数据库
 - 备份和还原**
 - 防止错误的事务
 - 数据库作为整体进行备份，并可通过门户恢复
 - 备份的保留期（按天计）基于所选的服务层

	Basic	Standard	Premium
按时间点还原	过去七天内的任何还原点	过去 14 天内的任何还原点	过去 35 天内的任何还原点
异地还原	最长停机时间 < 24 小时	最长停机时间 < 24 小时	最长停机时间 < 24 小时
标准异地复制	不包含	最长停机时间 < 2 小时	最长停机时间 < 2 小时

SQL 数据库有三个主要恢复选项。

按时间点还原

按时间点还原将数据库及其数据还原到早先的时间点。可以将数据库还原到过去多久的时间点取决于数据库层。

数据库副本

数据库副本功能将数据库的一次性副本创建成另一个数据中心中的新实例。此操作完成后，副本数据库与源数据库之间将形成事务一致。此操作后发生在源数据库上的后续事务不会复制。

导入和导出服务

此服务用来创建 BACPAC 格式的数据库副本。此文件包含数据库中数据和架构的副本。此文件可用来将数据库导入任何其他 Azure SQL 数据库实例，或导入计算机上单机安装的 SQL Server。

MCT USE ONLY STIDENT USE PROHIBITED

异地还原

异地还原的工作方式与按时间点还原相似。主要区别是异地还原将数据库复制到另一个地区的数据中心。异地还原通常用在特定 Azure 区域出现中断的时候。

标准异地复制

标准异地复制自动将主 SQL 数据库实例故障转移到另一个地区。创建的副本不可用，且不能使用 TDS 端点接受查询或传入的连接。主数据库的所有事务以异步方式复制到辅助数据库。当出现中断时，主数据库将被标记为服务降级。然后可以使用自动化或应用程序逻辑来连接辅助数据库。

Azure SQL 数据库业务连续性

<http://go.microsoft.com/fwlink/?LinkId=510204>

Azure SQL 数据库异地复制

- Premium SQL 数据库实例可使用活动异地复制
- 此功能默认是异步的，并将保证副本最终达到一致
- 可以复制事务到多达四个数据库副本
- 副本存在于异地冗余的不同地区
- 在负载平衡场景下，可以将数据库的副本用作只读数据源
- 示例：应用程序将主数据库用于日常业务功能，而将副本用于报表

	Basic	Standard	Premium
活动异地复制	不包含	不包含	RTO* <1 小时 RPO† <5 分钟

活动异地复制

开箱即用（Out-of-the box），活动异地复制对于简单故障转移场景，提供了与标准异地复制相同的功能。活动异地复制更进了一步，可以维护以同步（连续）方式复制的多达四个数据库副本。辅助数据库是可读的，以便在实施故障转移安排时，仍然可以无中断地获得数据。

Azure SQL 数据库的活动异地复制

<http://go.microsoft.com/fwlink/?LinkId=510206>

实验 A：在 Azure SQL 数据库中存储事件数据

场景

既然 Web 应用程序已经可以发布，那就可以通过在 Azure 中创建数据库，将 Web 应用程序迁移到 Azure。您决定使用数据库初始化器和 Entity Framework Code First 在 SQL 数据库中自动创建数据库。

 **注释：**本章为您提供了两套试验方案，如果您拿到的是国际版的 Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版的 Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 使用管理门户创建 Azure SQL 数据库服务器和数据库实例。
- 使用 Entity Framework Code First 在云中初始化并填充数据库。
- 使用 Azure SQL 数据库管理门户查看云中的实时数据。

实验设置

预计时间：45 分钟

在开始这个实验之前，必须完成第 2 章的实验 A。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，输入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：
vm28532[自定义名称].cloudapp.net: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动输入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为**Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

预览门户网站可能会偶尔无法成功创建新的服务实例。该门户仍处于技术预览阶段，可能存在尚未解决的问题(bug)。如果出现这种情况，可以简单地重试操作，直到服务实例成功创建。

本章为您提供了两套试验方案，如果您拿到的是国际版的 Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版的 Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

MCT USE ONLY. STUDENT USE PROHIBITED

练习 1：创建 Azure SQL 数据库实例

场景

在此练习中，您将：

- 在 Azure 中创建 SQL Server 实例。
- 在 Azure 中创建数据库。

此练习的主要任务如下：

1. 登录到 Azure 预览门户
2. 使用预览门户创建 Azure SQL 数据库

► 任务 1：登录到 Azure 预览门户

1. 登录到 Azure 预览门户网站 (<https://portal.azure.com>)。
2. 将语言设置为中文（简体）。

► 任务 2：使用预览门户创建 Azure SQL 数据库

1. 查看 SQL 数据库订阅列表。
2. 通过以下信息创建 SQL 数据库：
 - 名称: **db28532[自定义名称]**
 - 定价层: **Basic**
 - 服务器名: **sv28532[自定义名称]**
 - 服务器管理登陆: **testuser**
 - 服务器管理密码: **TestPa\$\$w0rd**
 - 位置: **选择最接近你所在位置的区域**
3. 记下新创建的 SQL 数据库的名字。

结果：完成此练习后，您将在 SQL 数据库服务中创建好服务器和数据库。

练习 2：结合 Azure SQL 数据库使用实体框架

场景

在此练习中，您将：

- 配置 Entity Framework Code First **DatabaseInitializer**。
- 实现用于初始值设定项的 **Seed** 方法。
- 发布云服务应用程序。
- 使用 Azure SQL 数据库管理门户查看表和数据。

此练习的主要任务如下：

1. 运行 ASP.NET Web 应用程序查看本地 SQL 数据库中的事件
2. 用新的 DatabaseInitializer 配置 DbContext
3. 用 DbContext 生成基本数据
4. 将包含更新的 DbContext 的云应用程序发布到 Azure

4-16 在 Azure 中存储 SQL 数据

5. 验证 Azure 云服务网站正在使用新数据
6. 登录到 Azure 管理门户
7. 查看 Azure SQL 管理门户中迁移的数据

► 任务 1：运行 ASP.NET Web 应用程序查看本地 SQL 数据库中的事件

1. 在 Visual Studio 中，从下面路径中打开 **Contoso.Events** 解决方案：
 - 文件路径：Allfiles(F):\Mod04\Labfiles\Starter\SQL\Contoso.Events
2. 调试 **Contoso.Events.DataGeneration** 控制台项目。

 **注释：**Data Generation 控制台应用程序将会在本地 SQL 数据库创建一些样本数据，以便测试 ASP.NET Web 应用程序。

3. 调试 **Contoso.Events.Cloud** 云项目。

► 任务 2：用新的 **DatabaseInitializer** 配置 **DbContext**

1. 在 **Contoso.Events.Data** 项目中，创建一个新类 **EventsContextInitializer**。
2. 把 **EventsContextInitializer** 类更新为 **public** 访问。
3. 更新 **EventsContextInitializer** 类以继承于 **DropCreateDatabaseAlways<EventsContext>** 类。
4. 给 **Contoso.Events.Models** 命名空间添加 **using** 语句。
5. 将下列代码添加到 **EventsContext** 静态构造函数：

```
Database.SetInitializer<EventsContext>(
    new EventsContextInitializer()
);
```

► 任务 3：用 **DbContext** 生成基本数据

1. 重写 **DropCreateDatabaseAlways<>** 类的 **Seed** 方法。
2. 给 **Contoso.Events.Models** 命名空间添加 **using** 语句。
3. 在 **Seed** 方法中，通过使用以下值创建一个实例 **EventItem**：
 - **EventKey: "FY17SepGeneralConference"**
 - **StartTime: DateTime.Today**
 - **EndTime: DateTime.Today.AddDays(3d)**
 - **Title: "FY17 September Technical Conference"**
 - **Description: "Sed in euismod mi."**
 - **RegistrationCount: 1**
4. 在 **Seed** 方法末尾，将新创建的实例 **EventItem** 添加到 **context.Events** 集合。
5. 在 **Seed** 方法末尾，通过以下值创建一个 **Registration** 类的实例：
 - **EventKey: "FY17SepGeneralConference"**
 - **FirstName: "Aisha"**
 - **LastName: "Witt"**
6. 在 **Seed** 方法末尾，将新创建的实例 **Registration** 添加到 **context.Registrations** 集合。

MCT USE ONLY. STUDENT USE PROHIBITED

7. 在 **Seed** 方法末尾，调用 **context.SaveChanges()** 方法。

8. 生成 **Contoso.Events.Data** 项目。

► **任务 4：将包含更新的 DbContext 的云应用程序发布到 Azure**

1. 更新 **Contoso.Events.Web** 项目中 **Web.Release.config** 文件，用一个连接字符串来连接已创建的 Azure SQL 数据库。

2. 通过以下详细信息将 **Contoso.Events.Cloud** 项目发布到一个新的云服务项目：

- 名称: **cs28532[自定义名称]**
- 区域: 选择离你所在位置最接近的区域
- 环境: **Production**
- 版本配置: **Release**
- 服务配置: **Cloud**

► **任务 5：验证 Azure 云服务网站正在使用新数据**

1. 查看发布的云服务网站。

► **任务 6：登录到 Azure 管理门户**

1. 登录到 Azure 管理门户网站 (<https://manage.windowsazure.com>)。

2. 将语言设置为中文（简体）。

► **任务 7：查看 Azure SQL 管理门户中迁移的数据**

1. 查看 SQL 数据库的订阅列表。

2. 查看先前创建的 SQL 数据库的仪表板：

- 名称: **db28532[自定义名称]**

3. 通过以下值在 Microsoft Visual Studio 中打开 SQL 数据库：

- 用户名: **testuser**
- 密码: **TestPa\$\$w0rd**

4. 查看 **dbo.Events** 表中的数据。

结果：完成本练习之后，你将配置好 Entity Framework 来初始化新数据库，新数据库中有种子(Seed)数据。
。

问题：哪些情况下，适合在 Azure SQL 数据库中使用来自 ORM 框架的种子数据？

实验 B：使用中国版 Windows Azure 在 Azure SQL 数据库中存储事件数据

场景

当网站应用程序准备发布时，你需要在 Azure 上创建数据库来迁移你的网站应用程序到 Azure 上。你决定在 SQL 数据上采用 database initializer 和 Entity Framework Code First 技术来自动创建数据库。

目标

在完成这个实验后，你将学会以下知识点：

- 通过 Azure 管理门户创建 Azure 数据库和数据库实例
- 在云端用 Entity Framework Code First 技术初始化和生成数据库
- 在云端用 Azure **Management Portal – SQL Database** 查看数据库数据

预计时间：45 分钟

在开始这个实验之前，必须完成第 2 章的实验 B。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，输入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].chinacloudapp.cn: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动输入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到 Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：创建 Azure SQL 数据库实例

场景

在此次实践中，你将学习到以下知识点：

- 在 Azure 上创建 SQL Server 实例
- 在 Azure 上创建数据库

此练习的主要任务如下：

1. 登录 Azure 管理门户

MCT USE ONLY. STUDENT USE PROHIBITED

2. 通过管理门户创建 Azure SQL 数据库

► 任务 1：登录 Azure 管理门户

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 进入 <https://manage.windowsazure.cn>。
3. 在邮箱帐户输入框，输入你的邮箱帐户地址；在密码输入框输入密码。
4. 单击 **Sign In** 登录。
5. 单击右上角的帐户名前的地球图标，选择中文（简体）选项。

► 任务 2：通过管理门户创建 Azure SQL 数据库

1. 在管理界面左下角，单击**新建**。
2. 在**新建**窗格下，首先在左侧栏目中单击**数据服务**，而后在中间弹出的选项中选择**SQL 数据库**，然后在右边弹出的选项中选择**自定义创建**。
3. 在**新建 SQL 数据库**的设置页面中，请按照如下步骤创建 SQL 数据库实例：
 - a. 在**名称**输入框输入 **db28532[你的名称]**。
 - b. 在**服务层**选项中，选择**BASIC**。
 - c. 在**服务器**选项中，选择**新建 SQL 数据库服务器**选项，其余选项保持默认值，单击右下方下一步按钮→进入创建服务器界面。
 - d. 在**登录名**输入框，输入 **testuser**。
 - e. 在**登录密码**和**确认密码**输入框，输入 **TestPa\$\$w0rd**。
 - f. 在**区域**选项中，选择一个区域，其余保持默认选项。
 - g. 最后单击右下方按钮√完成 SQL 数据库实例的创建。
4. 记录下创建的 SQL 数据库名称。

结果：在完成实践后，你将学会在 SQL Server 服务上创建服务器和数据库。

练习 2：在 Azure SQL 数据库中使用 Entity Framework

场景

在此次实践中，你将学习到以下知识点：

- 配置 Entity Framework Code First DatabaseInitializer
- 在 initializer 实现 Seed 方法
- 发布云服务应用
- 使用 Azure SQL 数据库管理门户查看表结构和数据

此练习的主要任务如下：

1. 通过运行 ASP.NET 网站应用查看本地 SQL 数据库的事件
2. 用一个新的 DatabaseInitializer 配置 DbContext
3. 用 DbContext 生成基本数据
4. 将包含更新的 DbContext 的云应用程序发布到 Azure
5. 验证 Azure 云服务站点是否使用新数据

6. 登录 Azure 管理门户
7. 在 Azure SQL 管理门户查看迁移数据

► 任务 1：通过运行 ASP.NET 网站应用查看本地 SQL 数据库的事件

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏单击 **File Explorer** 图标。
3. 在 This PC 窗口中，访问 **Allfiles\Mod04\Labfiles\Starter\SQL\Contoso.Events**，然后双击文件 **Contoso.Events.sln**。
4. 在解决方案资源管理器中，右键单击项目 **Contoso.Events.DataGeneration**，选择 **调试** 选项，然后选择 **启动新实例**。

 **注释：**数据生成脚本大概需要运行 1 到 2 分钟。

5. 在解决方案资源管理器中，右键单击项目 **Contoso.Events.Cloud**，然后单击 **设为启动项目** 选项。
6. 在调试菜单中单击 **启动调试** 选项。

 **注释：**在调试中，如果遇到 Server Error，请您执行以下步骤，然后重新启动调试：
展开 **Contoso.Events.Web** 项目，展开引用节点。
选择 **System.Web.Mvc**，在属性窗口中，修改复制本地的值为 **True**。

7. 在网站应用主页面，验证是否显示数据库事件信息。
8. 关闭当前网站的选项卡。

► 任务 2：用一个新的 **DatabaseInitializer** 配置 **DbContext**

1. 在解决方案资源管理器中，右键单击项目 **Contoso.Events.Data**，选择 **添加** 选项，而后选择 **新建项**。
2. 在弹出的 **添加新项** 窗口中，执行以下步骤：
 - a. 展开 **已安装**，展开 **Visual C# 项**，而后单击 **代码**。
 - b. 单击 **类** 模板。
 - c. 在名称输入框，输入 **EventsContextInitializer.cs**。
 - d. 单击 **添加**。
3. 在类 **EventsContextInitializer** 中，在类的定义左侧加上 **public** 访问器关键字：

```
class EventsContextInitializer
```

4. 更新的类定义应如以下代码：

```
public class EventsContextInitializer
```

5. 在类 **EventsContextInitializer** 中，在类的定义右侧添加继承基类 **DropCreateDatabaseAlways<EventsContext>**：

```
public class EventsContextInitializer : DropCreateDatabaseAlways<EventsContext>
```

6. 更新的类定义应如以下代码：

```
public class EventsContextInitializer : DropCreateDatabaseAlways<EventsContext>
```

7. 在该文件的顶部加上以下 **using** 代码:

```
using System.Data.Entity;
```

8. 在解决方案资源管理器, 展开项目 **Contoso.Events.Data**。

9. 在项目 **Contoso.Events.Data** 中, 打开文件 **EventsContext.cs**。

10. 在静态构造方法 **static EventsContext()** 中, 添加以下代码:

```
Database.SetInitializer<EventsContext>(new EventsContextInitializer());
```

11. 保存文件 **EventsContext.cs**。

► 任务 3: 用 **DbContext** 生成基本数据

1. 在解决方案资源管理器, 展开项目 **Contoso.Events.Data**。

2. 在项目 **Contoso.Events.Data**, 打开文件 **EventsContextInitializer.cs**。

3. 在类 **EventsContextInitializer** 中添加以下方法:

```
protected override void Seed(EventsContext context)
{
}
```

4. 在该代码文件的顶部加上以下 **using** 代码:

```
using Contoso.Events.Models;
```

5. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内, 并输入以下代码:

```
Event eventItem = new Event();
eventItem.EventKey = "FY17SepGeneralConference";
eventItem.StartTime = DateTime.Today;
eventItem.EndTime = DateTime.Today.AddDays(3d);
eventItem.Title = "FY17 September Technical Conference";
eventItem.Description = "Sed in euismod mi.";
eventItem.RegistrationCount = 1;
```

6. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内, 并输入以下代码:

```
context.Events.Add(eventItem);
```

7. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内, 并输入以下代码:

```
Registration registrationItem = new Registration();
registrationItem.EventKey = "FY17SepGeneralConference";
registrationItem.FirstName = "Aisha";
registrationItem.LastName = "Witt";
```

8. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内, 并输入以下代码:

```
context.Registrations.Add(registrationItem);
```

9. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内, 并输入以下代码:

```
context.SaveChanges();
```

10. 保存文件 **EventsContextInitializer.cs**。

11. 在解决方案资源管理器, 右键单击项目 **Contoso.Events.Data**, 然后单击生成选项。

► 任务 4：将包含更新的 **DbContext** 的云应用程序发布到 Azure

1. 在解决方案资源管理器窗格中，展开项目 **Contoso.Events.Web**。
2. 在解决方案资源管理器窗格中，展开项目 **Contoso.Events.Web** 的 **Web.config** 文件。
3. 双击文件 **Web.Release.config**。
4. 在打开的 **Web.Release.config** 文件中，找到 **connectionStrings** 节点，节点代码类似于
`<connectionStrings>...</connectionStrings>`。
5. 找到 **name** 属性值为 **EventsContextConnectionString** 的项目，执行如下步骤：
 - a. 将名称为 **connectionString** 属性的值更新为保存在记事本中的连接字符串。
6. 保存 **Web.Release.config** 文件。
7. 如果弹出保存只读文件对话框，请单击**覆盖**按钮。
8. 关闭记事本，不需要保存文件。
9. 展开 **Contoso.Events.Web** 项目，双击打开 **Web.config** 文件。
10. 在打开的 **Web.config** 文件中，找到 **runtime** 节点，节点代码类似于`<runtime>...</runtime>`。
11. 在 **runtime** 节点内，找到第一个节点，名称为 **assemblyBinding**，节点代码类似于
`<assemblyBinding
 xmlns="urn:schemas-microsoft-com:asm.v1">`。
12. 在该代码后，插入如下代码：

```
<dependentAssembly>
    <assemblyIdentity name="WindowsAzureEventSource" publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
    <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
    <assemblyIdentity name="Microsoft.WindowsAzure.ServiceRuntime"
publicKeyToken="31BF3856AD364E35" culture="neutral"/>
    <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
    <assemblyIdentity name="msshrtmi" publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
    <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
    <assemblyIdentity name="WindowsAzureTelemetryEvents"
publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
    <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
```

13. 保存 **Web.config** 文件。
14. 如果弹出保存只读文件对话框，请单击**覆盖**按钮。
15. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Cloud**，然后单击**发布**。
16. 在发布 **Azure** 应用程序窗口，执行以下操作：
 - a. 如果你以前没有选择订阅，请执行以下操作：
 - i. 用微软帐户登录。
 - ii. 在**选择订阅**选项中，选择一个订阅。
 - iii. 单击**下一步**。

MCT USE ONLY. STUDENT USE PROHIBITED

- b. 如果你以前选择了订阅，那么你将直接进入 **Microsoft Azure 发布设置** 步骤。
17. 打开**创建云服务和存储帐户**窗口：
 - a. 如果你没有存在的云服务，**创建云服务和存储帐户**窗口将会弹出。
 - b. 如果你有存在的云服务，在显示的**云服务列表**中选择**新建**。
18. 在**创建云服务和存储帐户**窗口中执行以下步骤：
 - a. 在**名称**输入框输入 **cs28532[自定义名称]**。
 - b. 在**地区或地缘组**选择一个离你最近的区域。
 - c. 单击**创建**。
19. 在 **Microsoft Azure 发布设置**窗口中，执行以下步骤：
 - a. 其他选项均保持默认值。
 - b. 单击**发布**。

 **注释：**完成云服务的发布一般需要 5 到 10 分钟。当你在发布你的云服务项目时，你可以在 Microsoft Azure 活动日志窗格跟踪发布进度。

► **任务 5：验证 Azure 云服务站点是否使用新数据**

1. 等待云服务发布完成，发布完成后控制台窗口将显示**已完成**信息。

 **注释：**如果发布完成，在 **Microsoft Azure 活动日志日志请求**窗口会显示**已完成**状态信息。在活动日志窗口中，圆形绿色指示器并不表示发布完成，而是表示当前的程序包上传成功。

2. 在 **Microsoft Azure 活动日志**窗口，单击超链接地址可以直接访问你发布的网站。
3. 验证云服务站点是否仅仅显示你在 Entity Framework context initializer 中创建的一条事件数据。

► **任务 6：登录 Azure 管理门户**

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 进入 <https://manage.windowsazure.cn>。
3. 在邮箱帐户输入框，输入你的邮箱帐户地址；在密码输入框输入密码。
4. 单击 **Sign In** 登录。

► **任务 7：在 Azure SQL 管理门户查看迁移数据**

1. 在左侧导航栏，单击 **SQL 数据库**。
2. 在界面顶部，单击**数据库**。
3. 在 **SQL 数据库**列表中，单击前缀名称为 **db28532** 的数据库。
4. 在界面顶部，单击**仪表板**。
5. 在界面的**速览**栏目下单击**管理允许的 IP 地址**。
6. 在界面右侧单击按钮→将**当前客户端 IP 地址**添加到**允许的 IP 地址**列表中。
7. 单击界面下方的**保存**按钮保存更改的配置。
8. 在页面顶部显示的是数据库的**服务器**名称，在接下来的步骤中，您将使用到该名称。
9. 新建记事本，执行以下步骤：

4-24 在 Azure 中存储 SQL 数据

- a. 将以下字符串粘贴到记事本中: [https://\[your server name\].database.chinacloudapi.cn/?langid-zh-cn](https://[your server name].database.chinacloudapi.cn/?langid-zh-cn)。
- b. 将[your server name]替换为您服务器的名称。
10. 在 Start 屏幕, 单击 **Internet Explorer** 磁贴。
11. 输入网址, 网址为您刚刚保存在记事本中的字符串。打开 **Management Portal–SQL Database** 页面。
 - a. 在 **DATABASE** 中, 输入您的数据库名称, 数据库名称类似于 **db28532[自定义名称]**。
 - b. 在 **USERNAME** 输入框输入帐号 **testuser**。
 - c. 在 **PASSWORD** 输入框输入 **TestPa\$\$w0rd**。
 - d. 单击 **Log On**。

 **注释:** Internet Explorer 可能会显示阻止弹出对话框警告。为了显示对话框, 请在 Internet Explorer 窗口下方的警告对话框单击 **Options for this Site**, 然后单击 **Always Allow**, 之后该警告对话框将不再出现, 你将可以访问该页面。完成上述步骤后, Azure 管理门户将刷新。

 **注释:** 如果一些版本的 Windows Server 虚拟机没有安装 Silverlight, 请根据提示安装 Silverlight。

12. 在 **Management Portal – SQL Database** 界面左下角, 单击 **Design**。
13. 在 **Tables** 列表中, 单击表 **dbo.Events**。
14. 单击该选中表右侧的 **Edit** 按钮。
15. 在该表页面的顶部单击 **Data**。
16. 在表 **dbo.Events** 中, 查看到一条数据记录。
17. 关闭 **Management Portal–SQL Database** 窗口。
18. 关闭 **Internet Explorer**。

结果: 在完成实践后, 你将学会用 Entity Framework 初始化包含基本数据的新数据库。

章节复习和作业

本章中，我们介绍了 Azure 存储以及提供的三种基本存储类型。还演示了一些如何使用 Azure Storage SDK 与表存储交互的示例。使用表存储，如今您在设计应用程序时，就可以确信存储机制可随时扩展到海量级别。

SQL 数据库提供了关系数据库的服务产品。建立该服务是为了最大程度兼容现有管理工具以及 Visual Studio 2013 中的 SSDT。使用实体框架，可以编写代码一次，并可根据环境，依靠配置更改将 ORM 指向正确的数据库。

 **最佳做法：**根据编译定义或环境，可以使用配置设置和 `web.config` 转换，将应用程序指向其他数据库。这样就能针对每种环境，用最少量的代码自定义自动化编译和测试过程。

复习问题

问题：在实施数据库分片时，为什么联合分发键非常重要？此键如何帮助促进性能？

MCT USE ONLY. STUDENT USE PROHIBITED

第 5 章 设计有复原力的云应用程序

目录:

章节概述	5-2
第 1 课: 高可用应用程序设计实践	5-3
第 2 课: 应用程序分析	5-5
第 3 课: 使用 ASP.NET 构建高性能应用程序	5-7
第 4 课: 常见云应用程序模式	5-10
第 5 课: 缓存应用程序数据	5-12
章节复习和作业	5-13

章节概述

作为开发人员，在设计云应用程序时，应谨记某些注意事项。虽然 ASP.NET 已经有了很多平台方面的增强，但是仍然需要顾及云应用程序的现有可伸缩性和可靠性指标，重新考虑应用程序的设计方式，以及所用的模式。第 1 节“高可用应用程序设计实践”讨论设计托管在云中的应用程序时要考虑的注意事项，它们能使停机时间降至最低。第 2 节“使用 ASP.NET 构建高性能应用程序”描述 .NET 4.5 中 ASP.NET 堆栈的部分更改，这些更改提高了该框架在 Web 应用程序中的性能。第 3 节“常见云应用程序模式”介绍来自 MSDN 云模式参考的一小部分示例模式。第 4 节“应用程序分析”演示 Application Insights 服务。第 5 节“缓存应用程序数据”比较 Microsoft Azure 缓存与 Microsoft Azure Redis 缓存服务。



注释: **Application Insights** 和 **Microsoft Azure Redis 缓存服务**在中国版 Windows Azure 中暂时不支持，本章中没有设计适合在中国版 Windows Azure 上完成的演示或实验。

第 1 课 高可用应用程序设计实践

本节描述设计高度可用应用程序时需要考虑的某些常见注意事项。

课程目标

完成本节后，您可以做到：

- 描述如何将应用程序拆分成多个工作单元。
- 描述应用程序的队列和负载平衡策略。
- 描述瞬态故障处理模式。

划分工作负载

划分工作负载

模块化应用程序可分成多个功能单元，这些单元也称为模块，模块可集成到更大的应用程序中。每个模块处理应用程序总体功能的一部分，每个模块也代表了一组彼此相关的设计需求。模块化应用程序使得设计应用程序的当前迭代及未来迭代变得更加容易。现有模块可扩展、可修改，也可替换，将更新并入完整应用程序。模块也可以被测试、分发或单独的验证。软件行业的很多开发人员和架构师都非常认同模块化设计的好处。

- 设计 Web 应用程序时，将业务流程拆分成多块工作负载
- **划分工作负载：**
 - 可以在模块化网站、云服务或虚拟机中进行处理
 - 提供单独伸缩应用程序不同组件的能力

 **注释：**对于模块化和可伸缩应用程序，你们有什么想法？这些想法未必要和 Web 应用程序有关。

负载平衡

使用算法将应用程序流量或负载分摊到各个端点，这种计算概念就是负载平衡。使用负载平衡器，可以创建网站的多个实例，并且这些实例可以以可预测的方式运行。这样就能在应用程序中灵活地增减实例数量，而又不会改变应有的行为。

负载平衡策略

选择负载平衡器时，需要考虑很多事项。首先，必须确定您希望使用物理负载平衡器还是虚拟负载平衡器。在 Azure 基础结构即服务 (IaaS) 中，如果公司需要有一些自己专属的、特别个性化的负载平衡器配置，可以使用虚拟负载平衡器，这种负载平衡器托管在虚拟机中。

- 用多个实例提供相同的服务，并使用负载平衡器在所有实例之间分摊请求
- **选择负载平衡策略的注意事项：**
 - 硬件或软件负载平衡器
 - 负载平衡算法（循环）
 - 负载平衡器粘性
- 即使您只有一个服务实例，负载平衡仍显得至关重要，因为它提供了无缝缩放的能力

在选定某个负载平衡器后，需要选择负载平衡算法。可以使用各种算法，如循环机制或随机选择。例如，循环机制根据包括所有实例在内的预先确定的顺序，为每个请求选择下一个实例。

负载平衡器还存在其他配置选项，如相关性或粘性。例如，粘性能让您确定来自同一个客户机的后续请求是否应该路由到同一个服务实例。在应用程序服务器有状态概念的情况下，可能需要这种配置。

瞬态故障处理

在企业内部和在云中开发应用程序之间的主要区别之一是，设计应用程序来处理瞬态错误的方式。瞬态错误是由于服务临时中断或由于延迟过长而发生的错误。很多这类临时问题可自我修复，并且可通过执行重试策略来解决。

重试策略规定当发生临时故障时，何时尝试连接以及尝试连接的频率。简单在无限循环中重试可能与无穷递归一样危险。循环中最终必须定义中断方式，万一最终确定错误性质很严重并且不是临时问题，就可以中止重试。

瞬态故障处理是一种模式，它以更稳健的方式处理临时问题，使得应用程序复原力更高。这是通过管理连接和实施重试策略来实现的。很多常见的 .NET 库中已经实现了此模式，如实体框架和 Azure 软件开发包 (SDK)。企业库中也以更通用的方式实现了此模式，使其可引入各种各样的应用程序场景。

瞬态故障处理（用 Azure 实现真正的云应用程序）

<http://go.microsoft.com/fwlink/?LinkId=525514>

瞬态故障处理应用程序块

<http://go.microsoft.com/fwlink/?LinkId=525515>

队列

排队既是一种数学理论，也是计算机科学中的消息处理概念。在云应用程序中，队列对于管理应用程序模块之间的请求至关重要，因为它提供了一定程度的一致性，而又无需考虑模块的行为。

应用程序可能已经使用直接方法调用、双向服务或任何其他流传输机制建立了与其他应用程序模块间的直接连接。如果其中某个应用程序模块遇到瞬态问题，那么此连接会断开，并造成即时应用程序故障。可以使用第三方队列在出现临时故障后保留请求。还可以脱离主应用程序单独审核请求，因为请求存储在队列机制中。

- 由于暂时性情况，可能发生瞬态故障

- 服务不可用
- 网络连接问题
- 服务承受高负载
- 重试请求可以解决通常造成应用程序崩溃的临时问题
- 可以使用不同的策略重试
 - 在固定时间段后重试
 - 等待时间呈指数级增长重试
 - 定时增量重试

- 如果每个组件都依赖于使用持久连接的直接双向通信，那么模块化 Web 应用程序的行为可能就像单机应用程序
- 如果某个应用程序组件失效或暂时不可用，持久队列消息可使应用程序仍能处理请求
- 外部队列可让应用程序审核请求或测量负载，而不会给主应用程序的代码增加任何开销

MCT USE ONLY STUDENT USE PROHIBITED

第 2 课 应用程序分析

分析软件可使开发人员和运维人员能够查看 Web 应用程序的使用情况和行为。对于云应用程序，使用分析正变得日益重要，因为每次迭代时，您必须做出缩放和设计决策。

本节描述 Azure 应用程序分析服务。

课程目标

完成本节后，您可以做到：

- 描述 Application Insights 服务。
- 将 Application Insights 脚本和程序集与 ASP.NET 项目集成来监视 Web 应用程序。

Application Insights

Application Insights 不仅是监视平台，而且还是三组应用程序扩展，可用来对自定义应用程序加强监控。为了使用 Application Insights，您需要将遥测应用程序扩展添加到现有应用程序或新应用程序，然后通过配置，将应用程序与 Application Insights 实例关联。这样您就能灵活地将遥测添加到现有应用程序，而无需重写甚至重新部署应用程序。Application Insights 还足够灵活，可以开放对非 Azure 托管应用程序的遥测。

最起码，可以使用 Application Insights 来确保应用程序正常运行时间达标。可以使用自定义仪表板来查看有关应用程序的各种指标。

这是一个 Application Insights 仪表板的示例。

- Application Insights 是可供应用程序使用的分析和监视服务
 - 查看异常堆栈跟踪
 - 监视 CPU 和资源使用情况
 - 从全世界的数据中心定期测试 URL
 - 监视应用程序的使用情况和大多数常见请求
 - 可配合 .NET 或 Java 使用
 - 应用程序无需专门托管在 Azure 中

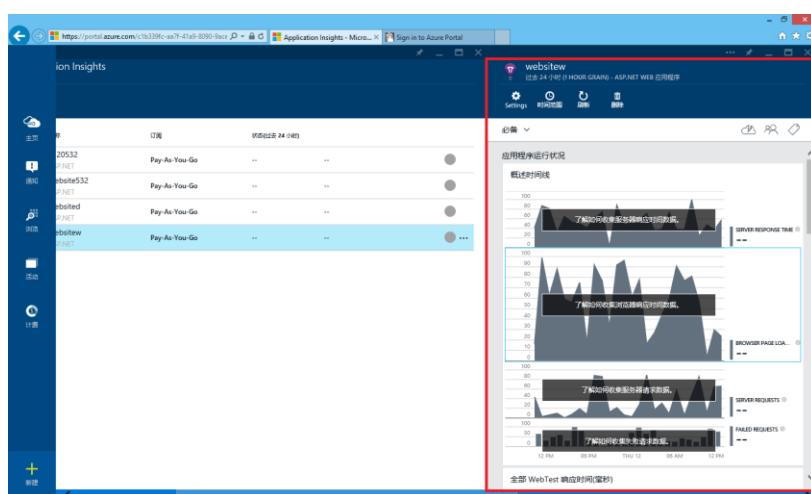


图 5.1: APPLICATION INSIGHTS 仪表板

可以使用高级指标来确定长期应用程序行为，并且根据此行为分析，可以剖析特定性能问题或应用程序问题。最后，Application Insights 还可让您深入了解最终用户的“理想路径”。可以按页或按客户端浏览器来查看使用量遥测数据，以此得出这一结论（Insight）。可以使用这些结论来确定应用程序的哪些方面值得未来研发投入。

演示 1：使用 Application Insights 监视 Web 应用程序请求

演示步骤

1. 打开 Visual Studio 2013。
2. 通过使用以下的详细信息创建一个新的 ASP.NET Web 应用程序项目：
 - 名称: **Contoso.Help**
 - 位置: **Allfiles (F):\Mod05\Democode**
 - 向项目添加 Application Insights: 是
 - 将遥测发送到: 新的 **Application Insights** 资源
 - 选择模版: **MVC**
 - 身份验证: **无身份验证**
 - 在云中托管: **网站**
3. 在 **Contoso.Help** 项目中打开 **HomeController.cs** 文件。
4. 修改 **About** 操作代码会返回一个状态为 HTTP-500 的错误。
5. 通过使用持续时间为 5 秒的 **Thread.Sleep** 操作修改 **Contact**。
6. 发布 ASP.NET web 应用程序。
7. 在 Internet Explorer 查看网页应用程序已被发布。
8. 登录到 Azure Preview Portal (<https://portal.azure.com>)。
9. 升级网站的 Web 宿主计划到 **Basic** 线。
10. 在 Azure Website 实例中查看 **Application Insights** 仪表板。
11. 创建一个 web test 并且明确主页在其他国家是可用的。

第 3 课 使用 ASP.NET 构建高性能应用程序

ASP.NET 4.5 版包含很多平台改进，在企业内部和云中构建高性能应用程序时，这些改进会有所帮助。

本节描述的是 ASP.NET 在高性能 Web 应用程序方面的改进。

课程目标

完成本节后，您可以做到：

- 描述异步 HTTP 模块和处理程序。
- 在 ASP.NET 应用程序中使用 **async** 和 **await** 关键字。
- 比较 ASP.NET 中状态管理的选项。

Web 托管性能改进

与以前的版本相比，ASP.NET 4.5 和.NET 4.5 在设计高性能.NET 应用程序方面做了很多改进。

Web 托管的性能改进

- 共享公共程序集

这项暂存功能使操作系统能够只存储每个程序集的一个副本，供托管在 IIS 中的所有 Web 应用程序使用。Web 应用程序 BIN 文件夹下的程序集将替换成链接（symbolic link），以减少 Web 服务器内存中相同程序集的数量。

- 使用多核 JIT（即时）编译实现更快启动

就像 Windows 的启动已从单核转向多核一样，ASP.NET Web 应用程序的 JIT 编译也从单核转向了多核。现代计算机趋向于使用多个处理器，而其中的每个处理器 CPU 速度降低一些。多核 JIT 编译默认情况下是启用的，并在所有可用处理器内核上分摊编译任务。

- 预提取 Web 应用程序

Windows 预提取器在您发出启动客户端应用程序的请求之前就预先加载应用程序组件，因而减少了最常用桌面应用程序的启动时间。使用 IIS 时，可通过自定义配置选项对 ASP.NET Web 应用程序启用预提取器。

ASP.NET 4.5 和 Visual Studio 2012 新增功能

<http://go.microsoft.com/fwlink/?LinkId=525517>

ASP.NET 4.5 为 Web 托管引入了很多改进：

- 异步请求和响应
- 支持任务并行库以及 await 和 async 关键字
- 在应用程序之间共享公共程序集
- 多核启动 (JIT)
- Web 应用程序的 Windows 预提取器

MCT USE ONLY. STUDENT USE PROHIBITED

异步 HTTP 模块和处理程序

在过去，可用任务并行库来编写异步方法。这要求深度理解 HTTP 请求类配合 ASP.NET 工作的方式，还需要一些自定义配置。即使那样，每个请求仍要到所有逻辑都完成后才会释放线程，甚至是异步也是如此。在 ASP.NET 4.5 中，ASP.NET 堆栈增加了异步读取流并刷新的方法。新的异步方法会在等待任务完成的同时释放其线程。这将确保整个应用程序中有更多线程可用，从而提高性能。在需要向服务发出多个请求，而这些请求可能引起延迟或瞬态错误的情况下，这些异步方法也会很有用。通常，这些请求会锁住多个线程，但现在这些线程在等待外部服务响应时，可用来接待其他请求。

任务并行库 (TPL)

<http://go.microsoft.com/fwlink/?LinkId=525518>

- ASP.NET 中的异步方法允许代码等待受限于 IO 的操作（数据库、服务、磁盘）将线程返回到线程池
- 通常这些请求会阻止线程服务于其他请求
- 如果在外部因素上等待时可以释放线程，那么可同时处理的请求数量将呈指数级增加。
- 对于负载激增并且通常不一致的应用程序来说，会有很大的帮助

Async 关键字

.NET 4.5 引入了 **async** 和 **await** 关键字，这两个关键字是在 .NET 4.0 中引入的任务编程概念的基础上建立的。**async** 标志用来标出将使用这些关键字的方法。**await** 标志用作 Task.ContinueWith 的简略表示。

更多 Task.ContinueWith 方法

<http://go.microsoft.com/fwlink/?LinkId=525519>

这两个关键字的优点是，它们提供了更易读懂的代码，便于其他开发人员解读。传统异步应用程序一般难以理解、调试和修改。通常，您需要大量工作流图表才能理解在引进任务并行库以及 **async** 和 **await** 关键字之前开发的异步应用程序。编译器执行从这些关键字到其任务等效形式的转换。

使用 Async 和 Await 的异步编程

<http://go.microsoft.com/fwlink/?LinkId=525520>

- **Async** 和 **await** 关键字可用于在 C# 中创建易于读写的异步方法
- 这两个关键字可配合 ASP.NET MVC 使用来创建异步操作：

```
public async Task<ActionResult> ItemsAsync()
{
    var context = new DatabaseContext();
    var model = await
        context.GetModelItemsAsync();
    return View("Items", model);
}
```

状态管理

ASP.NET 提供了在开发云应用程序时应该考虑的各种状态选项。

基于客户端的状态

使用 ASP.NET，可以将应用程序状态保存在客户端。这是使用 **ViewState**、**ControlState** 或 **cookie** 来实现的。出于多种原因，通常不推荐这些选项。首先，HTML 应用程序原则上设计为无状态。其次，这些选项往往增加响应和请求的开销和负担。最后，如果要托管 ASP.NET 应用程序的多个实例，那么这些选项需要额外的配置，因为每个实例都有不同的可识别的机器数据。如果客户端状态不可避免，则应考虑隐藏输入或查询字符串之类的选项。

应用程序状态或会话状态

ASP.NET 应用程序的应用程序状态或会话状态通常托管在进程中。在云中，由于负载平衡器可能将您转向新的应用程序实例，而新实例可能没有进程内的会话值，因此这可能成问题。例如，假设有三个应用程序实例，而您是使用第一个实例登录的。负载平衡器可能将您转到第二个应用程序实例，而该实例还没有与您的登录相关的任何状态信息。如果需要传统的 ASP.NET 服务器状态管理，那么应考虑使用共享状态服务器。



ASP.NET 状态管理概述

<http://go.microsoft.com/fwlink/?LinkId=525521>

- 在跨多个实例分布应用程序时，需要在实例之间共享会话状态
 - 当用户在 Server1 上启动，但在单击链接后处于 Server2 时，会发生什么情况？
- 会话状态可从内存中移到专用的会话服务器上：
 - Microsoft SQL Server
 - ASP.NET 状态服务器
- 会话状态还可分区并分布在多个会话存储中

第 4 课 常见云应用程序模式

虽然有很多应用程序模式，但是某些应用程序模式已经随新一代的云原生 Web 应用程序而出现。MSDN 包含某些最常见云应用程序模式的特选列表。

本节详细描述特选云应用程序模式的三个示例。

课程目标

完成本节后，您可以做到：

- 描述重试模式。
- 使用副密钥模式访问资源。
- 使用分片模式缩放受限数据存储机制。

云应用程序模式

MSDN 包含一篇由 patterns & practices 团队发表的指南，其中不仅提供了有关指导，还列举了 20 多个云应用程序最常用的设计模式示例。这些模式不是 ASP.NET 也不是 Microsoft 所特有的。本节中，您将回顾其中的一部分模式，并讨论使用这些模式的示例场景。对于每种模式，下面的链接提供了代码示例。

 云设计模式：云应用程序规范性体系结构指南

<http://go.microsoft.com/fwlink/?LinkId=525522>

- MSDN 提供了一组云设计模式
- 模式：
 - 重试
 - 副密钥
 - 分片

重试模式

无论应用程序是托管在云中，还是使用外部云托管服务，它们在连接外部服务时，都可能遇到瞬态错误。如果连接错误其实是临时问题，那么可以使用重试模式来实施有限数量的重试。

当初始连接失败时，可以先分析故障原因来确定故障是否为瞬态。如果故障原因或错误代码指出，此请求即使在多次重试后仍不可能成功，那么重试就根本不会执行。

重试会一直执行，直至连接成功或达到重试限制。设置此限制是为了避免出现无限重试的情况。重试通常在定时延迟之后执行，每次重试后，延迟可能保持不变，也可能呈线性或指数组增加。如果达到了重试次数限制，即表示连接以非瞬态方式失败。

有很多外部库实现了重试策略模式，包括：

- 实体框架
- 企业库 - 瞬态故障处理应用程序块

 重试模式

- 重试模式是设计来处理临时故障的
- 故障假设是瞬态的，直至超出重试策略
- 瞬态故障处理块是一个示例库，被设计来实现重试模式（及其他）
- 实体框架提供了内置的重试策略实现
 - 在 6.0 版中实现

<http://go.microsoft.com/fwlink/?LinkId=510187>

副密钥模式

云应用程序可能允许某些资源可由最终用户下载。如此一来，这将成为云应用程序本身的责任，此外还会让云程序紧密依赖于存储机制。如此一来，如果一个 Web 服务提供了将图像下载并流式传输给用户的功能。即便只是为了下载图像，也要消耗大量处理能力和内存量。应用程序可能为了确保处理所有下载请求，而限制将资源分配到其他任务。例如，应用程序需要先验证用户再下载受保护的图像，就会遇到这种尴尬的局面。如果确保了下载图像，用户身份验证的任务就无法很好地兼顾。

副密钥模式引入了一种机制，按照这种机制，应用程序可验证用户请求，而无需管理媒体的实际下载。当出现上述类似的情况时，就可以较好地解决问题了。

该模式规定，如果客户端要访问媒体资源，客户端需要向 Web 服务或应用程序发出一个请求。然后，应用程序验证请求。如果请求有效，应用程序转向外部存储机制，生成临时访问令牌，然后将媒体资源的 URI 连同临时访问令牌提供给客户端。应用程序现已处理了此请求，可继续处理其他请求。接下来该由客户端应用程序或浏览器使用 URI 和临时访问令牌来下载媒体资源。现在，存储服务负责扩展来处理所有索要媒体的请求，而应用程序则完全专注于其他功能。



副密钥模式

<http://go.microsoft.com/fwlink/?LinkId=510188>

- 如果应用程序代表客户端访问资源，服务器会承担额外的负载
- 您不想公开提供资源，因此通常必须让应用程序验证客户端
- 副密钥模式规定应用程序只要验证客户端，然后将令牌返回给客户端。然后客户端就可以使用自己的硬件和带宽直接检索资源

第 5 课 缓存应用程序数据

Azure 提供了两种主要缓存机制，这些机制有助于存储可由应用程序的服务共享的一致数据。虽然 Redis 缓存现在是首选缓存机制，但是仍然有必要理解 Azure 缓存，因为它跟现有云应用程序有关。

本节描述 Azure 中的两种缓存产品：Azure 缓存和 Redis 缓存。

课程目标

完成本节后，您可以做到：

- 描述 Azure 缓存。
- 描述 Redis 缓存。

Redis 缓存

Azure 中有两种主要缓存机制，Azure 缓存和 Redis 缓存。Azure 缓存已经弃用，其存在只是为了支持现有云应用程序。所有新应用程序都应该使用 Redis 缓存。

Azure 托管缓存

Azure 缓存是基于 AppFabric 平台的托管缓存服务。可以使用第三方应用程序或 Windows PowerShell cmdlet 创建缓存实例。这种缓存机制常见于使用自定义云服务角色的时候。

Redis 缓存

Redis 缓存是一种开源 NoSQL 存储机制，是按照其他 NoSQL 存储中常见的键值对模式实现的。Redis 缓存之所以独特是因为，它的键允许复杂的数据结构。

Redis

<http://go.microsoft.com/fwlink/?LinkID=525523>

基于开源的 Redis 平台

- 有多个提供不同节点数的层可用
- 支持事务
- 使用发布订阅模型支持消息聚合
- 在键可以是简单值或复杂值的情况下，考虑键值存储
- 庞大的 Redis 生态系统已经与很多不同的客户端共存

Azure Redis 缓存是基于 Redis 缓存的托管服务，它提供了安全节点即服务。此服务目前有两层。

- 基本。单节点
- 标准。主/从配置的双节点。包含复制支持和服务级别协议 (SLA)

Azure Redis 缓存高度兼容已经与 Redis 缓存集成的现有工具和应用程序。可以使用已经存在于开源社区的 Redis 缓存文档来了解 Azure Redis 缓存。

Azure Redis 缓存

<http://go.microsoft.com/fwlink/?LinkID=525524>

章节复习和作业

本章中，您回顾了创建云应用程序的某些模式和引导。这可能不是您第一次看到这些模式或做法。在开发运行于外部硬件的无连接模块化 Web 应用程序时，如 Azure，很多模式的收效很大。很多对企业内部应用程序显得杀鸡用牛刀的模式和工具，现在对分布式云应用程序却是至关重要。

复习问题

问题：为什么要跨地区定期检查 Azure 网站的可用性或其检测信号？

问题：如何使用 Azure 服务实现副密钥模式？

MCT USE ONLY. STUDENT USE PROHIBITED

第 6 章 管理 Azure 中的云服务

目录:

章节概述	6-2
第 1 课: 云服务概述	6-3
第 2 课: 云服务 Web 角色	6-5
第 3 课: 云服务辅助角色	6-6
第 4 课: 云服务角色处理	6-9
第 5 课: 自定义云服务	6-13
第 6 课: 更新和管理云服务部署	6-17
实验 A: 使用 Azure 辅助角色创建后台进程	6-19
实验 B: 使用 Azure 辅助角色创建后台进程	6-23
章节复习和作业	6-30

章节概述

随着应用程序向外扩展，您可能发现到某些业务逻辑正在成为瓶颈。例如，购物网站随着销量的上升，前台网站页面可以自如访问，但订单处理速度成为了瓶颈。如果能按需增强订单处理部分的能力就比较理想。云服务包含辅助角色，这些角色异步执行长时间运行的后台逻辑（异步于 Web 前端角色）。这解放了 Web 角色，使其可专注于托管 Web 前端应用程序。第 1 节“云服务概述”介绍云服务基础结构和角色。第 2 节“云服务 Web 角色”描述云服务项目中常用的 Web 角色。第 3 节“云服务辅助角色”介绍辅助角色，并描述辅助角色和 Web 角色之间的区别。第 4 节“云服务角色处理”描述有助于实现云服务辅助角色的特殊类。第 5 节“自定义云服务”回顾云服务的自定义和配置选项。第 6 节“更新和管理云服务”着重介绍云服务独特的包格式。

目标

完成本章后，您可以做到：

- 描述 Azure 云服务方案。
- 说明云服务部署与虚拟机和网站服务相比的复杂性。
- 描述 Web 角色与辅助角色之间的区别。
- 描述云服务的角色内缓存（In-Role Cache）。
- 使用 Azure 计算模拟器测试云服务。
- 创建云服务辅助角色。
- 实现 RoleEntryPoint 基类的方法。
- 配置辅助角色。
- 将缓存与云服务角色共置（co-locate）在一起。
- 将辅助角色转换为专用缓存角色。

MCT USE ONLY. STUDENT USE PROHIBITED

第 1 课 云服务概述

在开发 Web 应用程序并适应生产环境时，有时您面临的问题不仅仅是应用程序的实际逻辑。

由于硬件故障、更新、补丁、组网和备份等各种因素，开发 Web 应用程序有时候会考验您的基础结构管理能力。

本节描述 Azure 中提供的云服务，以及如何将这些云服务用于应用程序，而无需操心基础结构。

课程目标

完成本节后，您可以做到：

- 描述云服务产品。
- 描述云服务的体系结构。

云服务的体系结构

Azure 虚拟机为自定义应用程序提供了必要的基础结构，Azure 网站服务提供类似于 Internet Information Services (IIS) 的 Web 托管，而云服务提供将基础结构和简单托管合二为一的平台即服务 (PaaS) 选项。这项技术旨在支持可缩放、可靠而且运营成本低的应用程序。该技术的主要重点是让开发人员专注于应用程序，并依靠 Azure 基础结构来管理基础结构复原力、连接性和缩放。

Azure 云服务提供平台即服务

- Azure 中的云服务由三个主要组件组成：
 - 运行 Windows Azure OS 的一个或多个虚拟机
 - 基于 Windows Server 2012
 - 定义服务端点以及与服务和角色相关的其他设置的 XML 配置文件
 - 每个角色的应用程序二进制文件
- 在创建、回收或缩放服务时；将创建新的虚拟机，应用程序二进制文件将部署到每个虚拟机

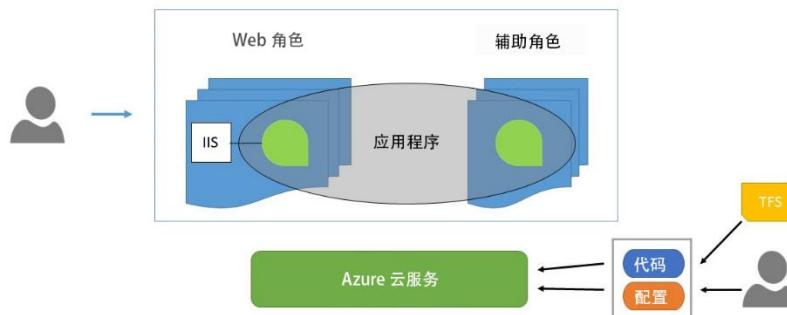


图 6.1：AZURE 云服务

云服务的核心是虚拟机。云服务技术提供了预配置的虚拟机映像，通常称为 Web 角色和辅助角色。可以按各种组合使用这些角色来适合多种不同类型的应用程序。例如，简单 Web 应用程序可能只需要一个 Web 角色。但是，如果应用程序还要执行某些后台处理，除了 Web 角色外可能还需要辅助角色。

云服务包含将在本章讨论的以下功能：

- 允许无限伸缩的模块化开发模型。
- 可用来在企业内部服务器上测试云服务应用程序的本地软件开发包 (SDK) 模拟器。
- 统一诊断模型。
- 可使用配置文件和门户修改的云设置。

- 自动更新和虚拟机管理。
- 端点管理。

云服务角色

可以使用云服务角色来拆分应用程序的模块。Web 前端可托管在一对多（one-to-many）Web 角色中，应用程序后台处理可托管在多个 Web 角色中。在角色定义中，可以指定角色的多个实例来获得复原力。每种角色类型之间有很多相似的地方，只有很小区别。

- **Web 角色。** Web 角色是为了 Web 应用程序编程而自定义的角色，并由 IIS 7 和 ASP.NET 支持。使用这类角色的好处是 IIS 设置已经替您设好。此角色最适合用来为云服务提供基于 Web 的前端。它不适合长时间运行的进程。
- **辅助角色。** 辅助角色是对泛化开发（generalized development）很有用的角色，并可以为 Web 角色执行后台处理。如果希望后台进程执行长时间运行的任务或间歇性任务那么应使用此角色。

- **Web 角色**
 - 用于托管带有公共 http/tcp 端点的应用程序
 - 示例：
 - MVC 网站
 - WCF 服务
 - Web API 服务
- **辅助角色**
 - 用于在后台执行长时间运行任务或间歇性任务的逻辑
 - 可以让多个实例使用来自排队机制的数据

Web 角色和辅助角色之间没有很多区别。它们都基于同一种操作系统，并且都可能有端点并消耗外部资源。默认情况下，Web 角色具有 IIS 承载的 Web 核心，核心已经安装在虚拟机上并已启用。Web 角色还有典型的 HTTP 端点（端口 80/443），在 Microsoft Visual Studio 云服务项目中创建新的 Web 角色时，默认情况下可以将这些端点添加到配置。

虽然每种角色类型都有典型的模式可用，但是可以灵活地使用角色。例如，辅助角色可以把 Web 服务器托管在进程中，也可以配置端口号为 80 的端点。Web 角色也可以移除对外的公共端点，将其 HTTP Web 服务变为仅供其他应用程序模块内部使用。

第 2 课 云服务 Web 角色

Web 角色提供了预先建好的 Web 服务器，可以用来托管前端 Web 应用程序。

本节介绍 Web 角色及其与云服务的关系。

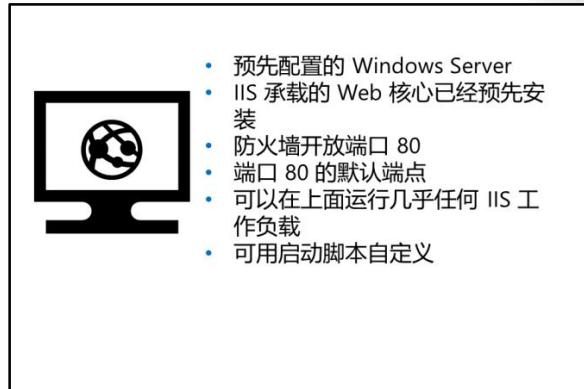
课程目标

完成本节后，您可以做到：

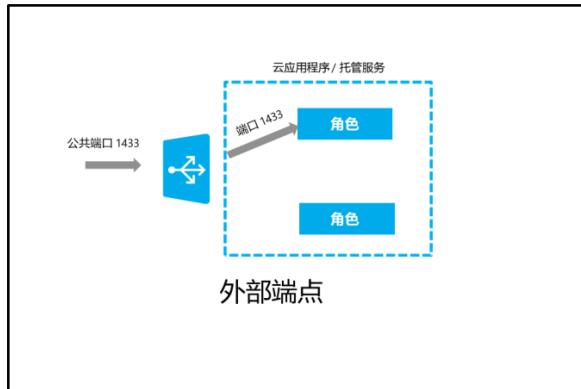
- 描述 Web 角色中提供的功能。
- 解释 Web 角色的行为。
- 确定为新的 Web 角色创建哪些端点。

Web 角色

Web 角色提供专用的 IIS Web 服务器，用于托管前端 Web 应用程序。此角色类型通常用于有着公共 http/tcp 端点的 Web 应用程序。在 Microsoft Visual Studio 2013 中，云服务 Web 角色通常对应于 ASP.NET Web 应用程序项目。通过修改项目配置，可以将现有 ASP.NET Web 应用程序项目迁移到云服务 Web 角色。



端点



在云服务中，角色实例通过内部和外部连接通信，这些连接因所需通信的类型而异。角色实例可使用 HTTP、HTTPS 以及适用于 TCP/IP 套接字的 Microsoft .NET API 创建这些连接。外部连接称为输入端点，内部连接称为内部端点。端点与端口关联，其中外部端点与您定义的端口关联，内部端点是 Azure 动态分配的端口。

每个 Web 角色可以有零到多个输入端点和内部端点。输入端点可由云服务外部的使用者公开访问。内部端点只能由同一云服务中的角色实例访问。例如，Node.js Express 应用程序可作为 Web 应用程序的后端 RESTful Web 服务托管在 Web 角色中。此 Web 服务只需要由云服务中的其他角色访问。可以使用内部端点，而不是输入端点来实现此服务。内部端点不支持 HTTPS。

第 3 课 云服务辅助角色

辅助角色提供了一组可伸缩的工作进程，这些进程可使用来自存储机制的消息，并与客户端和 Web 前端项目分开，单独处理这些消息。这样，您就能根据实际使用情况，灵活地伸缩后台工作进程或前端项目。

本节介绍辅助角色及其提供的独特功能。

课程目标

完成本节后，您可以做到：

- 描述 Web 角色和辅助角色之间的区别。
- 解释与辅助角色通信的选项。
- 描述典型的辅助角色算法。

辅助角色

辅助角色是一种专用实例，可运行异步、长时间运行或持久任务，而无需任何用户交互或输入。可以将辅助角色用于各种任务，它们非常适合不需要 IIS 在虚拟机上运行的场景。

辅助角色的示例场景

- ASP.NET 中的长时间运行任务耗尽线程池资源
 - 即使有了 **async**，后台操作仍要耗用线程。
- 某些外部操作占用大量 I/O
 - 对云应用程序来说，有必要考虑不可靠的连接。
- 缩小 Web 前端所需的规模和资源
 - 关注缓存和处理网站请求。
 - 如果可以抽象后台任务，那就可以节省很多资源。

Web 角色和辅助角色之间有一些关键区别。对于 Web 角色，IIS 管理线程，而且还优化效能。但是，对于辅助角色，不仅必须手动管理线程，还要实现应用程序的运行逻辑。最后，对于辅助角色，还必须手动定义 ACL 规则，因为默认情况下，尚未配置这些规则。



- 预先配置的 Windows Server
- 其他什么都不用安装
- 无默认 Azure 端点
- 运行不需要 IIS 的自定义工作负载
- 使用脚本安装其他软件
- 实现 **WorkerRole.cs** 类中的逻辑

与辅助角色通信

对于后台服务，它们必须侦听源才能让消息得到处理。云中运行的应用程序通常应该处理大量请求，使用云服务辅助角色可以异步处理这些请求。使用消息系统，辅助角色可以独立处理消息，而无需依赖于消息生产者的状态。消息系统还允许辅助角色（使用者）独立缩放数量，而与消息生产者无关。

使用消息队列可以实现应用程序与使用者服务（consumer service）的实例之间的通信通道。应用程序将请求以消息形式发布到队列，使用者服务实例从队列中接收消息，并处理这些消息。这种方法使同一个使用者服务实例池能处理来自应用程序任意一个实例的消息。

使用消息队列将工作分摊到服务实例。

- 由于辅助角色是孤立的模块，并且可以有很多实例，因此需要使用外部存储来保存发给辅助角色的请求
- Azure 队列存储可用来存储发给辅助角色的队列消息
- 也可选用Service Bus队列来存储队列消息
- 可以使用关系数据库来存储发给辅助角色的消息
- 可以使辅助角色成为 WCF 或 Web API 的服务宿主，并通过端点接收请求



图 6.2：竞争使用者模式

出于多种原因，请求的数量可能随着时间出现显著变化。用户活动的暴增或者来自多租户的累积请求可能造成难以预测的工作负载。在高峰时间，系统可能需要每秒处理数百条请求，而在其他时间，数量则可能很小。此外，处理不同请求要做的工作也可能各不相同。只使用使用者服务的一个实例可能使该实例被请求所淹没，或者消息系统中可能涌入大量来自应用程序的消息，从而造成过载。为了处理这些上下波动的工作负载，系统可运行使用者服务的多个实例。但是，这些使用者必须协调才能确保每条消息只传送到一个使用者。您还需要在多个使用者之间平衡工作负载，以防某个实例成为瓶颈。最后，通过使用消息系统，即使辅助角色实例在处理消息时失败或崩溃，消息仍能持久保存。

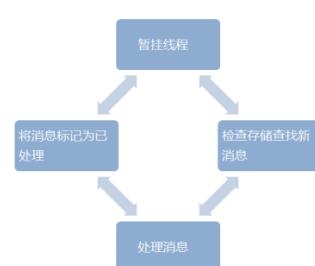
在辅助角色中处理数据

Azure 提供了相应的存储队列和 Service Bus 队列，这些队列可作为适合实现此模式的机制。应用程序逻辑可将消息发布到队列，而在一个或多个角色中，作为任务实现的使用者可从此队列中检索消息，然后处理这些消息。

消息处理逻辑是包含以下步骤的简单无限循环：

- 检查消息解决方案，查找新消息。探测消息存储解决方案，寻找需要处理的新消息。
- 处理消息。消息在辅助角色中得到处理。
- 将消息标记为已处理。消息在消息解决方案中标记为已处理，这样就不会再得到处理。

- 辅助角色通常实现了简单模式



4. 暂挂线程。在没有消息要处理的情况下，线程临时挂起（或进入休眠状态），以便在消息请求之间放入缓冲区。如果消息解决方案实现了限流，那么这一点非常重要。

第 4 课 云服务角色处理

Microsoft.WindowsAzure.ServiceRuntime 命名空间包含 **RoleEntryPoint** 类，这个类用作 Web 角色和辅助角色的基类。通过重写 **OnStart** 和 **Run** 方法，可以提供一个让后台工作进程执行的循环。

本节描述 **RoleEntryPoint** 基类，以及为了实现辅助角色而需要重写的方法。

课程目标

完成本节后，您可以做到：

- 从 **RoleEntryPoint** 类继承。
- 重写 **OnStart** 和 **Run** 方法。
- 创建一个辅助角色在检查新消息时要使用的无限循环。

使用 ServiceRuntime

使用 **Microsoft.WindowsAzure.ServiceRuntime**

命名空间中可用的类，可以管理角色在 Azure 中启动、运行或停止时如何响应。

RoleEntryPoint 类包含在 Azure 启动、运行或停止 Web 角色或辅助角色时所调用的方法。您可以视情况重写这些方法来管理角色初始化、角色关闭序列或角色的执行线程。辅助角色必须扩展

RoleEntryPoint 类。对于 Web 角色，可以扩展 **RoleEntryPoint**，也可以不扩展。

如果角色未启动，或者在初始化、繁忙和停止状态之间回收，那么代码在每次角色重启时，可能在某个生命周期事件中引发了未处理的异常。这种情况下，可以使用 **UnhandledException** 事件来确定异常原因并相应进行处理。您的角色还可能从 **Run** 方法返回，造成角色重启。

- **Microsoft.WindowsAzure.ServiceRuntime** DLL 随附在 Azure SDK 中
- 此 DLL 包含 **Microsoft.WindowsAzure.ServiceRuntime** 命名空间，其中有与角色和角色管理相关的类
- **RoleEntryPoint** 类位于此命名空间中

演示 1：回顾 **RoleEntryPoint** 的默认实现

演示步骤

1. 以管理员身份运行 Visual Studio 2013。
2. 创建一个新的 **Visual C# 类库** 项目，详细信息如下：
 - 项目/解决方案名称：**WorkerDemo**
 - 角色：
 - 辅助角色
 - 类型：辅助角色
 - 名称：**ConsoleRole**
 - Web 角色
 - 类型：ASP.NET Web 角色
 - 名称：**ASPRole**

3. 查看 **ConsoleRole** 项目下的 **WorkerRole.cs** 类文件。
4. 查看 **ASPRole** 项目下的 **WebRole.cs** 类文件。

 **注释:** 这两个文件分别表示了 web 角色以及辅助角色中 **RoleEntryPoint** 类的默认实现。可以使用这些示例作为起点来，构建您的自定义应用程序。

演示 2：回顾 **RoleEntryPoint** 的默认实现

演示步骤

1. 在 Start 屏幕，右键单击 **Visual Studio 2013** 磁贴。
2. 在弹出的菜单栏，单击 **Run as Administrator**。

 **注释:** 如果显示了 User Account Control (UAC) 对话框，你可以提升管理员权限来运行 Visual Studio 2013。

3. 在 Visual Studio 起始页，单击**新建项目...**。
4. 在**新建项目**对话框，执行以下操作。
 - a. 展开**已安装，模板，Visual C#**，然后单击**Cloud**。
 - b. 单击**Azure 云服务模板**。
 - c. 在名称输入框，输入**WorkerDemo**。
 - d. 单击**确定**。
5. 在新 **Microsoft Azure 云服务**对话框，**.NET Framework 4.5** 角色选项列表中，双击辅助角色选项。
6. 你将在 **Microsoft Azure 云服务解决方案**窗口看到一个新的名称为 **WorkerRole1** 的辅助角色实例。
7. 右键单击 **WorkerRole1** 实例，然后单击**重命名**。
8. 输入值 **ConsoleRole** 并按回车键。
9. 在**.NET Framework 4.5** 角色选项列表中，双击 **ASP.NET Web 角色**选项。
10. 你将在 **Microsoft Azure 云服务解决方案**窗口看到一个新的名称为 **WebRole1** 的 Web 角色实例。
11. 右键单击 **WebRole1** 实例，然后单击**重命名**。
12. 输入值 **ASPRole** 并按回车键。
13. 单击**确定**按钮。
14. 在弹出的**新建 ASP.NET 项目-ASPRole** 窗口中单击**确定**按钮。
15. 在**解决方案资源管理器**，展开项目 **ConsoleRole**。
16. 双击文档 **WorkerRole.cs**。
17. 在**解决方案资源管理器**，展开项目 **ASPRole**。
18. 双击文档 **WebRole.cs**。
19. 关闭 **WorkerDemo – Microsoft Visual Studio** 应用程序。

实现 **RoleEntryPoint** 类

RoleEntryPoint 类是先通过创建 **OnStart** 方法来实现的。在 Azure 使角色的实例进入联机状态时，会调用 **OnStart** 方法。在 **OnStart** 的代码执行时，角色实例将标记为“繁忙”，负载平衡器不会将任何外部流量引向该实例。您可以重写此方法来执行初始化工作，如实现事件处理程序和启动 Microsoft Azure 诊断。

如果 **OnStart** 返回 `true`，说明实例初始化成功，Azure 将调用 **RoleEntryPoint.Run** 方法。如果 **OnStart** 返回 `false`，角色立即终止，而不执行任何计划的关闭序列。

下面的 **OnStart** 方法在角色实例启动，并设置将日志数据传输到存储帐户时，配置并启动诊断监视器。

OnStart 方法

```
public override bool OnStart()
{
    var config = DiagnosticMonitor.GetDefaultInitialConfiguration();

    config.DiagnosticInfrastructureLogs.ScheduledTransferLogLevelFilter = LogLevel.Error;
    config.DiagnosticInfrastructureLogs.ScheduledTransferPeriod =
TimeSpan.FromMinutes(5);

    DiagnosticMonitor.Start("DiagnosticsConnectionString", config);

    return base.OnStart();
}
```

可以重写 **Run** 方法来为角色实例实现一个长时间运行的线程。

由于方法总是默认启动一个休眠的线程，因此重写 **Run** 方法不是必需的。如果确实要重写 **Run** 方法，您的代码应无限阻塞。如果 **Run** 方法返回，角色将自动回收。换言之，Azure 引发 **Stopping** 事件，并调用 **OnStop** 方法，这样就可能在角色脱机前执行关闭序列。

典型的 Run 方法实现。

Run 方法

```
private override void Run()
{
    while (true)
    {
        var message = GetMessage();
        if (message != null)
        {
            ProcessMessage(message);
            message.Handled = true;
        }
        Thread.Sleep(1000);
    }
}
```

此进程已经由新的默认实现改进。此实现利用了取消令牌，允许 **OnStop()** 方法实现在云服务实例需要立即停止时，取消任何异步运行的线程。

支持取消的 **Run** 方法实现。

- 辅助角色必须和任何其他角色一样，有一个类继承自同一个基类
- **RoleEntryPoint** 类实现了生命周期管理方法，子类可以重写这些方法
- Windows Azure 平台采用其找到的第一个派生自 **RoleEntryPoint** 的类，然后按下面的顺序运行应用程序的方法
 - **RoleEntryPoint.OnStart()**
 - **Global.Application_Start()**
 - **RoleEntryPoint.Run()**

Run 异步方法

```

public override void Run()
{
    ManualResetEvent runCompleteEvent = new ManualResetEvent(false);
    CancellationTokenSource cancellationTokenSource = new CancellationTokenSource();
    try
    {
        this.RunAsync(cancellationTokenSource.Token).Wait();
    }
    finally
    {
        runCompleteEvent.Set();
    }
}

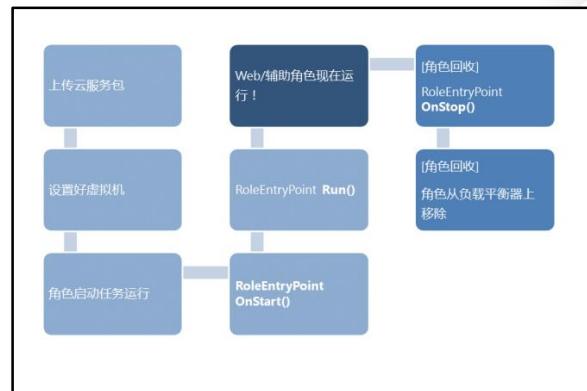
private async Task RunAsync(CancellationToken cancellationToken)
{
    while (!cancellationToken.IsCancellationRequested)
    {
        var message = GetMessage();
        if (message != null)
        {
            ProcessMessage(message);
            message.Handled = true;
        }
        await Task.Delay(1000);
    }
}

```

云服务角色生命周期

当云服务角色回收时，角色将在销毁前从负载平衡器上移除。可以将角色归入升级域（upgrade domains）来最大程度减少回收期间的停机时间。系统会为该角色创建一个新的虚拟机，然后启动任务再次运行。如果启动任务或 **RoleEntryPoint** 代码出现致命错误，角色也可能回收。

必须将云服务设计为，由启动任务处理 Web 角色的所有安装和配置。作为开发人员，应该将云服务设计为可以随时创建或销毁角色虚拟机。



第 5 课 自定义云服务

云服务角色提供了比网站服务更深度的自定义。角色可与缓存共置、可有自定义端点，甚至可以有云托管的设置。

Azure 计算模拟器提供了可测试 Web 前端应用程序及其诊断过程的环境。可以使用模拟器来了解在部署云服务后，应用程序的近似模拟会如何表现。

本节描述可用于云服务的监视选项，介绍 Azure 计算模拟器，然后描述可用于 Web 角色和辅助角色的自定义选项。

课程目标

完成本节后，您可以做到：

- 描述各种虚拟机大小。
- 说明如何修改 Web 角色的配置设置。
- 说明如何访问本地存储及其潜在缺陷。
- 描述计算模拟器。
- 调试运行中的云服务。

缩放

在 Azure 管理门户上的“缩放”页，可以手动缩放应用程序，或者设置参数来自动缩放。可以缩放运行 Web 角色或辅助角色的应用程序。若要缩放运行 Web 角色或辅助角色实例的应用程序，必须添加或移除角色实例来容纳工作负载。可以根据 CPU 平均使用百分比或队列中的消息数来指定缩放。

纵向缩放

可以配置云服务中的每个角色来使用不同的虚拟机大小。若要纵向缩放应用程序，可以使用配置来更改大小。Azure 门户中提供了最新的虚拟机大小。

Azure 虚拟机和云服务的大小

<http://go.microsoft.com/fwlink/?LinkId=525383>

- 可以为每个云服务角色逐个增加实例计数
- 云服务和虚拟机为创建的 VM 提供了不同的大小：

大小	CPU 内核数	内存
A0	共享	768 MB
A1	1	1.75 GB
A2	2	3.5 GB
A3	4	7 GB
A4	8	14 GB
A5	2	14 GB
A6	4	28 GB
A7	8	56 GB

横向缩放

可以增加或减少实例数量来手动缩放云服务。

手动更改实例数量

6-14 管理 Azure 中的云服务



图 6.1：横向缩放

横向自动缩放

在“缩放”页上，可以将云服务配置为自动增加或减少应用程序所用的实例数量或虚拟机数量。可以基于以下参数配置缩放：

- **平均 CPU 使用率。**如果平均 CPU 使用百分比高于或低于指定的阈值，Azure 将相应创建或删除角色实例。同时在可用性集里，也将开启或关闭虚拟机。
- **队列消息数。**如果队列中的消息数量高于或低于指定的阈值，Azure 也将创建或删除角色实例。同时，可用性集中也将开启或关闭虚拟机。

有各种配置选项可用于进一步自定义自动缩放行为，如“增加/减少实例数量”或“增加/减少等待时间”。自动缩放功能总是在一定范围内，您可以指定该范围的最小和最大实例数，这最终有助于控制成本。

MCT USE ONLY. STUDENT USE PROHIBITED



图 6.2：自动缩放



图 6.3：横向自动缩放

本地存储

可以使用本地存储资源来存储启动任务创建的文件，以后应用程序将访问这些文件。

为了在启动任务中使用本地存储资源，必须在 ServiceDefinition.csdef 文件中创建一个 **LocalStorage** 元素，然后创建一个环境变量来引用本地存储资源位置。然后，启动任务和应用程序可读取文件并将文件写入本地存储资源。

为了访问本地存储资源，应用程序必须通过 **GetLocalResource** 方法检索路径。然后运行标准文件读写操作来读写本地存储资源的内容。

从本地存储资源读取 MyTest.txt 的内容。

- 角色中的本地存储可以在服务定义文件中定义
- 当应用程序需要读取或修改文件时，本地存储资源在文件系统中保留一个目录
- 本地存储资源可以在多次实例回收之间保留，但是不保证肯定保留

本地存储

```
string SlsPath = RoleEnvironment.GetLocalResource("StartupLocalStorage").RootPath;
string s = System.IO.File.ReadAllText(SlsPath + "\\MyTest.txt");
```

诊断

诊断能让您从 Azure 中运行的应用程序收集诊断数据。可以使用诊断数据进行调试和故障排除；测量性能；监视资源使用情况、流量分析和容量规划；以及审核。在收集诊断数据后，可以将这些数据传输到 Azure 存储帐户进行持久存储。传输可以按计划执行，也可以在需要时执行。

可以使用 XML 配置文件来配置诊断。还可以从在 Azure 外部运行的应用程序上远程配置诊断。例如，可以从在 Azure 外部运行的自定义仪表板应用程序中管理诊断。通过远程管理诊断，可以用初始诊断配置启动应用程序，然后从应用程序之外运行的代码更改配置，而不必更新应用程序。

可以将诊断模块导入服务模型，然后配置从中收集诊断数据的数据源，这样就能收集诊断数据。可以将数据传输到 Azure 存储来存储诊断数据。诊断监视器在 Azure 以及 Azure 计算模拟器中运行，以收集角色实例的诊断数据。使用诊断模块的角色实例在启动时将自动启动诊断监视器。必须将诊断数据的来源添加到诊断监视器的配置才能收集数据。除了 Azure 日志、IIS 7.0 日志和 Azure 诊断基础结构日志外，还可以从其他来源收集日志数据，如 IIS 失败请求跟踪日志、Windows 事件日志、故障转储和自定义错误日志。

可以收集与云服务应用程序相关的诊断数据，并存储在公共存储帐户中

- 可以只存储错误或存储所有信息，或者创建自定义诊断配置来定义要存储的内容
- 诊断数据存储在 Azure 存储帐户下，这些帐户定义在角色配置文件中
- 诊断是使用门户、自动化脚本或云服务包的配置文件配置的

第 6 课 更新和管理云服务部署

云服务有独特的部署过程。如果理解了云服务包与其配置文件之间的关系，就可以自定义您要部署的内容，及其对运行中云服务的影响（即就地更新）。

本节回顾将云服务包和配置更改部署到新的服务实例或运行中实例的注意事项和过程。

课程目标

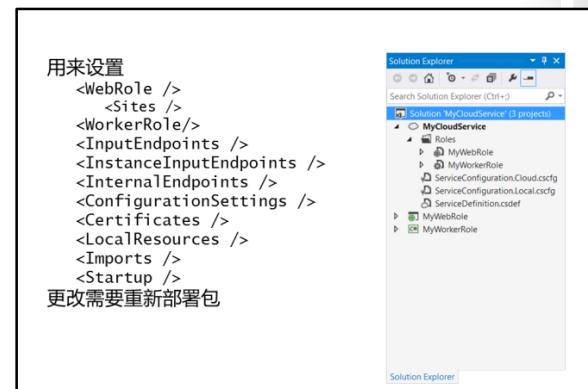
完成本节后，您可以做到：

- 描述云服务包的内容。
- 就地更新正在运行的云服务。
- 在运行时修改云服务的配置设置。

云服务包

云服务是使用两种文件部署的：

- **云服务包 (.cspkg)**
 - 包含每个角色的二进制文件
 - 还包含每个角色的自定义配置：
 - 设置
 - 端点
- **云服务配置文件 (.cscfg)**
 - 包含云服务的配置文件，其中包含角色定义和网络配置
 - 包含角色的实例数量和证书
 - 包含 ACL 规则
 - 定义云服务的虚拟网络



可以使用各种 Azure 门户来管理配置文件 (.cscfg) 中指定的配置设置。可以更改这些设置，也就是在门户中更新这些设置或者上传新的.cscfg 文件。如果要在部署之后更改任何设置，则需要使用 Azure 配置设置，而不是平台配置设置。这些设置可以轻松更改，并且可以在多次回收之间持久保留。当角色回收或更新时，对本地文件或设置（如 Web.config）的任何更改可能丢失。大多数磁盘上存储应视为易失的。使用门户还可以管理其他设置，如证书。

就地更新

Azure 将角色实例组织成称为升级域的逻辑分组。升级域的默认数量为五。可以在服务的定义文件 (.csdef) 中包含 **upgradeDomainCount** 属性来指定其他数量的升级域。执行服务中一个或多个角色的就地更新时，Azure 根据角色实例集所属的升级域更新这些实例集。Azure 更新某个升级域中的所有实例 — 停止它们，更新它们，再使它们重新联机 — 然后转到下一个域。Azure 只停止正在当前升级域中运行的实例，从而确保更新对正在运行的服务产生尽可能小的影响。

如果角色必须就地更新而不停机，那么服务必须定义角色的至少两个实例。如果服务只有一个角色的一个实例，则服务要到就地更新完成后才会可用。

- 对于现有的托管服务，可以上传更新后的配置文件
 - 必须与原先作为云服务包的一部分上传的服务定义匹配
 - 有利于提供新的配置值，而不更改应用程序结构
- 如果需要更改应用程序，可以将新的云服务包和配置上传到暂存槽
 - 可以执行虚拟 IP 交互操作来交换暂存实例和生产实例

实验 A：使用 Azure 辅助角色创建后台进程

场景

Contoso Events 应用程序当前需要管理员生成一个签到表 Word 文档。但是，如果要扩展应用程序，这可能成为瓶颈。为了避免此瓶颈，您决定将解决方案重新设计为让 Web 前端应用程序发出请求来生成签到表。您还决定创建辅助角色来检查这些请求并生成签到表。

本实验中，您将创建一个 C# 类库项目，添加相应的引用，然后将项目作为辅助角色添加到云服务。随后，您将使用 Azure 计算模拟器来测试 Web 角色和辅助角色。

 **注释：**本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版 Windows Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 从 C# 类库项目创建辅助角色。
- 实现 RoleEntryPoint 类的方法。
- 使用 Azure 计算模拟器调试辅助角色。

实验设置

预计时间：60 分钟

在开始这个实验之前，必须完成第 2 章的实验 A。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

- 在计算机上，单击**开始**，输入**远程**，然后单击**远程桌面连接**。
- 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].cloudapp.net:[虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动输入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为**Remote Desktop** 的端点是正确的端口 RDP。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

- 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
- 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
- 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验 A 中使用这些密码和 Azure 帐户。

本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版 Windows Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

练习 1：创建 C#类库

场景

在此练习中，您将：

- 创建 C# 类库项目。
- 添加对 ServiceRuntime DLL 的引用。
- 继承 **RoleEntryPoint** 基类来创建类。
- 实现 **OnStart** 和 **Run** 方法。

此练习的主要任务如下：

1. 创建 C# 类库项目
2. 为 Azure SDK 库和解决方案项目添加引用
3. 创建从 **RoleEntryPoint** 继承的类
4. 实现获得 SQL 数据库发来的请求的运行逻辑

► 任务 1：创建 C#类库项目

1. 以管理员身份运行 Visual Studio 2013。
2. 打开第六章的入门解决方案文件 **Contoso.Events**：
 - 文件路径：Allfiles (F):\Mod06\Labfiles\Starter\Contoso.Events
3. 在 **Contoso.Events** 解决方案下，新建一个 **Visual C# 类库**项目，项目名如下：
 - **Contoso.Events.Worker**

► 任务 2：为 Azure SDK 库和解决方案项目添加引用

1. 在 **Contoso.Events.Worker** 项目，向 **Microsoft.WindowsAzure.ServiceRuntime** 库添加一个引用。
2. 向 **Contoso.Events** 解决方案下的以下项目添加引用：
 - **Contoso.Events.Models**
 - **Contoso.Events.Data**
 - **Contoso.Events.Documents**
3. 通过使用程序包管理器控制台，针对 **Contoso.Events.Worker** 项目，运行以下命令：

```
Install-Package EntityFramework -Version 6.0.2
```

4. 生成 **Contoso.Events.Worker** 项目。

► 任务 3：创建从 **RoleEntryPoint** 继承的类

1. 向项目添加一个名为 **WorkerRole** 的类。
2. 打开 **WorkerRole** 类。
3. 把 **WorkerRole** 类改为公共类。
4. 重新定义类，使它继承于 **Microsoft.WindowsAzure.ServiceRuntime** 命名空间中的 **RoleEntryPoint** 基类。
5. 实现 **OnStart** 方法，使其调用 **RoleEntryPoint** 基类中的该方法。在调用 **base.OnStart()**方法前应先将 **ServicePointManager.DefaultConnectionLimit** 的值设为 12。

6. 实现 **Run** 方法，在其中定义一个简单的无限 **while** 循环。再 **Run** 方法中使用 **Trace.WriteLine** 方法，使其在第一次运行时写入 **Trace**。在 **while** 循环中调用 **Thread.Sleep**，时间设为 10,000 毫秒，然后调用 **Trace.WriteLine** 输出队列运行迭代。

```
public override void Run()
{
    Trace.WriteLine("Queue Run Start");

    while (true)
    {
        Thread.Sleep(10000);

        Trace.WriteLine("Queue Run Iteration");
    }
}
```

► 任务 4：实现获得 SQL 数据库发来的请求的运行逻辑

1. 向 **Contoso.Events.Worker** 项目添加 **WorkerRole.cs** 文件：
 - o 文件路径：**Allfiles (F):\Mod06\Labfiles\Starter\WorkerRole.cs**
2. 打开 **App.config** 文件。
3. 用以下 XML 代码代替 **App.config** 文件中的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <connectionStrings>
        <add name="EventsContextConnectionString"           connectionString="Data
Source=(localdb)\v11.0;Initial Catalog=EventsContextModule6Lab;Pooling=True;Integrated
Security=True providerName="System.Data.SqlClient" />
    </connectionStrings>
</configuration>
```

结果：完成此练习后，您将创建好辅助角色类库，并实现后台工作进程的相应模式。

练习 2：将类库添加到云服务项目

场景

在此练习中，您将：

- 将类库项目作为辅助角色添加到云服务项目中。

此练习的主要任务如下：

1. 将类库项目作为辅助角色添加到云服务项目

► 任务 1：将类库项目作为辅助角色添加到云服务项目

1. 在 **Contoso.Events** 解决方案内，卸载 **Contoso.Events.Worker** 项目。
2. 编辑 **Contoso.Events.Worker.csproj** 文件，使以下标签成为 **PropertyGroup** XML 标签的子标签：
 - o **<RoleType>Worker</RoleType>**
3. 重新加载 **Contoso.Events.Worker** 项目。
4. 使 **Contoso.Events.Worker** 类库项目作为一个辅助角色与 **Contoso.Events.Cloud** 云服务项目相关联。

结果：完成此练习后，您将获得现有类库项目，并将其转换为现有云服务项目中的辅助角色。

练习 3：调试云服务项目中的辅助角色

场景

在此练习中，您将：

- 将断点添加到辅助角色。
- 使用 Azure 计算模拟器调试辅助角色和 Web 角色。

此练习的主要任务如下：

1. 调试辅助角色
- 任务 1：调试辅助角色
1. 调试 **Contoso.Events.Cloud** 项目。

 **注释：**自始至终，这个课程都将使用 **Azure Compute Emulator** 来测试和调试云服务。在将云服务部署到 Azure 平台前，这个模拟器提供了一个快速的和本地的选项来测试它们。偶尔，这个模拟器会遇到奇怪的问题，例如不正确的端口号、页面显示不了等等。如果出现这种情况，只要关掉 **Compute Emulator**，在下次调试项目时，Visual Studio 将会自行启动这个模拟器。在调试过程中，如果 VS 报告用户代码未处理 **DataException**，请您中断当前调试，并重新启动调试。

2. 单击打开任意一个事件。
3. 单击 **Generate Sign-In Sheet** 按钮生成一个 **Sign-In Sheet**。
4. 等待 **Sign-In Sheet** 生成，然后下载。
5. 观察在 WordPad 程序内打开的.docx 文件。

结果：完成此练习后，您将调试了辅助角色。

问题：什么类型的 C# 类库项目最容易迁移为辅助角色？

调试 Web 角色时的实例计数

在调试云服务时，请确保将 Web 角色和辅助角色的数量减至最小。实例数可能总是在发布云服务之前增加。您当然可通过缓慢增加实例计数来测试可伸缩性，但是请记得，Azure 计算模拟器的最大角色实例数为 50。

实验 B：使用 Azure 辅助角色创建后台进程

场景

Contoso Events 应用程序当前需要管理员生成一个签到表 Word 文档。但是，如果要扩展应用程序，这可能成为瓶颈。为了避免此瓶颈，您决定将解决方案重新设计为让 Web 前端应用程序发出请求来生成签到表。您还决定创建辅助角色来检查这些请求并生成签到表。

本实验中，您将创建一个 C# 类库项目，添加相应的引用，然后将项目作为辅助角色添加到云服务。随后，您将使用 Azure 计算模拟器来测试 Web 角色和辅助角色。

目标

完成本实验后，您将能够：

- 从 C# 类库项目创建辅助角色。
- 实现 **RoleEntryPoint** 类的方法。
- 使用 Azure 计算模拟器调试辅助角色。

预计时间：60 分钟

在开始这个实验之前，必须完成第 2 章的实验 B。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

- 在计算机上，单击**开始**，输入**远程**，然后单击**远程桌面连接**。
- 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].chinacloudapp.cn:[虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动输入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的端口 RDP。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

- 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
- 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
- 验证您拿到的是中国版 Windows Azure 帐户。
- 验证使用您的中国版 Windows Azure 帐户可以登录到世纪互联运营的 Windows Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：创建 C#类库项目

场景

在这个练习中，你将学习到以下知识点：

- 创建 C#类库项目
- 为添加引用 ServiceRuntime DLLs
- 创建一个继承基类 **RoleEntryPoint** 的子类

- 实现 **OnStart** 和 **Run** 方法

此练习的主要任务如下：

- 创建 C#类库项目
- 为解决方案项目和 Azure SDK 类库添加引用
- 创建继承基类 **RoleEntryPoint** 的子类
- 实现从 SQL 数据库获得请求的运行逻辑

► **任务 1：创建 C#类库项目**

- 在 Start 屏幕，右键单击 **Visual Studio 2013** 磁贴。
- 在弹出的菜单栏，单击 **Run as Administrator**。



注释：如果显示了 User Account Control (UAC) 对话框，你可以提升管理员权限来运行 Visual Studio 2013。

- 在 Visual Studio 起始页界面，单击**打开项目...**。
- 在**打开项目**对话框，访问 **Allfiles (F):\Mod06\Labfiles\Starter\Contoso.Events**，然后单击 **Contoso.Events.sln**。
- 单击 **Open**。
- 展开**解决方案资源管理器**以便查看当前解决方案下的所有项目。
- 在**解决方案资源管理器**，右键单击解决方案 **Contoso.Events** 节点。
- 选择**添加选项**。
- 单击**新建项目...**。
- 在**添加新项目**对话框中，执行以下操作：
 - 展开**已安装**，**Visual C#**，然后单击**Windows 桌面**。
 - 单击**类库模板**。
 - 在名称输入框，输入 **Contoso.Events.Worker**。
 - 单击**确定**。

► **任务 2：为解决方案项目和 Azure SDK 类库添加引用**

- 右键单击项目 **Contoso.Events.Worker**。
- 选择**添加选项**。
- 单击**引用**。
- 在**引用管理器 – Contoso.Events.Worker**对话框，执行以下操作：
 - 展开**程序集**，然后单击**扩展**。
 - 选择版本号是 **2.5.0.0** 的程序集 **Microsoft.WindowsAzure.ServiceRuntime**。
 - 单击**确定**。
- 右键单击项目 **Contoso.Events.Worker**。
- 选择**添加选项**。
- 单击**引用**。

8. 在引用管理器 – **Contoso.Events.Worker** 对话框，执行以下操作：

- a. 展开解决方案，然后单击项目。
- b. 选择项目 **Contoso.Events.Models**。
- c. 选择项目 **Contoso.Events.Data**。
- d. 选择项目 **Contoso.Events.Documents**。
- e. 单击确定。

9. 在视图菜单，选择其他窗口选项，然后单击程序包管理器控制台。

10. 在程序包管理器控制台，默认项目列表中选择 **Contoso.Events.Worker**。

在程序包管理器控制台文本区域，将光标放在 PM>之后，输入如下命令行：

```
Install-Package EntityFramework -Version 6.0.2
```

11. 单击回车键。

12. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击生成。

► 任务 3：创建继承基类 **RoleEntryPoint** 的子类

1. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**。

2. 选择添加选项。

3. 单击新建项。

4. 在添加新项 – **Contoso.Events.Worker** 对话框，执行以下操作：

- a. 展开已安装，然后展开 Visual C# 项。
- b. 单击代码项。

c. 在名称输入框，输入 **WorkerRole.cs**。

5. 单击添加。

6. 在解决方案资源管理器，展开项目 **Contoso.Events.Worker**。

7. 双击文档 **WorkerRole.cs**。

8. 定位到以下代码：

```
class WorkerRole
```

9. 用以下代码替换第 8 步定位的代码：

```
public class WorkerRole
```

10. 在该类文档的顶部添加以下 using 块：

```
using Microsoft.WindowsAzure.ServiceRuntime;
```

11. 定位到以下代码：

```
public class WorkerRole
```

12. 用以下代码替换第 11 步定位的代码：

```
public class WorkerRole : RoleEntryPoint
```

13. 单击类 **WorkerRole** 括号内的任何位置。
14. 在你的方法之间用回车键添加垂直空白。
15. 在你的类中添加以下代码块：

```
public override bool OnStart()  
{  
    ServicePointManager.DefaultConnectionLimit = 12;  
  
    return base.OnStart();  
}
```

16. 在该类文档的顶部添加以下 **using** 块：

```
using System.Net;
```

17. 单击类 **WorkerRole** 括号内的任何位置。
18. 在你的方法之间用回车键添加垂直空白。
19. 在你的类中添加以下代码块：

```
public override void Run()  
{  
    Trace.WriteLine("Queue Run Start");  
  
    while (true)  
    {  
        Thread.Sleep(10000);  
  
        Trace.WriteLine("Queue Run Iteration");  
    }  
}
```

20. 在该类文档的顶部添加以下 **using** 块：

```
using System.Diagnostics;  
using System.Threading;
```

21. 保存类文档 **WorkerRole.cs**。

► 任务 4：实现从 SQL 数据库获得请求的运行逻辑

1. 右键单击项目 **Contoso.Events.Worker**。
2. 选择添加选项。
3. 单击现有项。
4. 在添加现有项 – **Contoso.Events.Worker** 对话框中，执行以下步骤：
 - a. 访问 **Allfiles (F):\Mod06\Labfiles\Starter**。
 - b. 单击 **WorkerRole.cs**。
 - c. 单击添加。
5. 在目标文件已存在对话框中，单击是替换现有文件。
6. 在 **Microsoft Visual Studio** 对话框，单击是刷新源编辑器。
7. 在解决方案资源管理器，展开项目 **Contoso.Events.Worker**。
8. 双击文档 **App.config**。
9. 选中该配置文档的所有内容。

10. 按下删除键清除该文档所有内容。

11. 输入以下 XML 节点：

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
    <add name="EventsContextConnectionString" connectionString="Data
Source=(localdb)\v11.0;Initial Catalog=EventsContextModule6Lab;Pooling=True;Integrated
Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

结果：完成此练习之后，你将创建一个包括辅助角色类的类库项目并为这个后台辅助角色实现了适当的模式。

练习 2：添加类库项目到云服务项目中

场景

在这个练习中，你将学习到以下知识点：

- 在云服务项目中添加一个辅助角色的类库项目。

此练习的主要任务如下：

- 在云服务项目中添加一个辅助角色的类库项目。

► 任务 1：在云服务项目中添加一个辅助角色的类库项目

- 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后点卸载项目。
- 如果你还没有保存项目，系统将提示你保存项目。单击是保存项目。
- 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击编辑 **Contoso.Events.Worker.csproj**。
- 定位到节点打开的**<PropertyGroup>**。
- 在**<PropertyGroup>** XML 元素和**<Configuration>** XML 元素之间，添加以下 XML 元素：

```
<RoleType>Worker</RoleType>
```

- 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击重新加载项目。
- 系统将会提示你关闭文档.csproj 因为它已经打开。单击 Yes 关闭文档.csproj。
- 在提示对话框，单击是保存项目。
- 在解决方案资源管理器窗格内，展开 **Models** 文件夹，然后展开 **Contoso.Events.Data** 项目，双击 **EventsContextInitializer.cs** 文件。
- 定位到以下代码行：

```
Public class EventsContextInitializer : CreateDatabaseIfNotExists<EventsContext>
```

- 用以下代码代替在步骤 10 中定位到的代码行：

```
Public class EventsContextInitializer : DropCreateDatabaseAlways<EventsContext>
```

- 在解决方案资源管理器，展开项目 **Contoso.Events.Cloud**，然后展开文件夹 **角色**。

13. 右键单击文件夹**角色**。
14. 选择**添加**选项。
15. 单击**解决方案中的辅助角色项目**。

 **注释:** 如果选项**解决方案中的辅助角色项目...**是禁用的, 你可以通过重新加载项目**Contoso.Events.Worker**来启用它, 步骤如下:

- a. 在**解决方案资源管理器**, 右键单击项目**Contoso.Events.Worker**, 然后单击**卸载项目**。
 - b. 在**解决方案资源管理器**, 右键单击项目**Contoso.Events.Worker**, 然后单击**重新加载项目**。
16. 在**与角色项目关联**对话框中, 执行以下步骤:
 - a. 单击**Contoso.Events.Worker**。
 - b. 单击**确定**。

结果: 完成本练习后, 你会用现有的类库项目, 并在现有的云服务项目中把它转换成一个辅助角色。

练习 3: 在云服务项目中调试辅助角色

场景

在这个练习中, 你将学习到以下知识点:

- 在辅助角色中添加断点。
- 用 Azure 计算仿真器调试辅助角色和 Web 角色。

此练习的主要任务如下:

1. 调试辅助角色

► 任务 1: 调试辅助角色

1. 在**解决方案资源管理器**, 右键单击项目**Contoso.Events.Cloud**, 然后单击**设为启动项目**。
2. 在**调试**菜单, 单击**启动调试**。

 **注释:** 整个过程我们将使用**Azure 计算仿真器**来测试和调试我们的云服务。在将你的服务部署到 Azure 平台之前, 仿真器提供了一种快速和本地的测试方案。有时候, 你可能看到模拟器奇怪的错误, 比如错误的端口号或无法显示网页。如果这个不断发生, 你可以**关闭计算仿真器**, Visual Studio 将在下一次尝试调试项目时**启动仿真器**。

在调试过程中, 如果 VS 报告**用户代码未处理 DataException**, 请您中断当前调试, 并重新启动调试。

3. 在**Contoso Events**网站的主页, 单击任意事件的名称。
4. 单击**Generate Sign-In Sheet**。
5. 每隔 30 秒刷新页面直到创建登录表。

 **注释:** 你如果发现按钮文本从**Generate Sign-In Sheet**变成了**Download Sign-In Sheet**, 这表明登陆表已创建完成。

6. 单击 **Download Sign-In Sheet** 下载文档文件。
7. 在 Internet Explorer 窗口底部弹出的下载对话框中，单击 **Open**。
8. 在 WordPad 中，你将看到打开文档.docx 的内容。
9. 关闭 **WordPad** 应用程序。
10. 关闭 **Internet Explorer** 应用程序。
11. 关闭 **Contoso.Events – Microsoft Visual Studio** 应用程序。

结果：完成本练习后，你将学会调试辅助角色。

问题：什么类型的 C# 类库项目最容易迁移为辅助角色？

调试 Web 角色时的实例计数

在调试云服务时，请确保将 Web 角色和辅助角色的数量减至最小。实例数可能总是在发布云服务之前增加。您当然可通过缓慢增加实例计数来测试可伸缩性，但是请记得，Azure 计算模拟器的最大角色实例数为 50。

MCT USE ONLY STUDENT USE PROHIBITED

章节复习和作业

本章中，我们介绍了在 Azure 中托管 Web 前端应用程序的其他方法。有了云服务，Web 应用程序就可以专注其代码，而让 Azure 虚拟机管理程序和结构控制器来处理基础结构和可伸缩性。有了 Azure 计算模拟器，您就可以使用这件新工具在本地测试云服务，以确保角色按预期运行。

在设计和构建云应用程序时，辅助角色带来了更大的灵活性。将长时间运行或受限于 I/O 的逻辑迁移到辅助角色，就可以将应用程序的 Web 前端和后台工作进程分开扩展，并更接近地符合应用程序的真实使用需求。

复习问题

问题：什么样的数据需要缓存，但是可逻辑地存储在不同的命名缓存中？试举几例

问题：为什么要让 RoleEntryPoint.Run 方法的实现结束当前无限循环并回收角色？出于哪些原因？

问题：在将现有 Web 应用程序迁移到 Web 角色之前，必须对这些 Web 应用程序进行哪些更改？

通过在右列中放置标记来判断叙述的正确性。

叙述	答案
是非题：在将云服务发布到 Azure 后，Web 角色的 Web.config 设置便不可更改。	

问题：哪种 Web 角色会开放内部端点？试举一例。

第 7 章 在 Azure 中存储表格式数据

目录:

章节概述	7-2
第 1 课: Azure 存储概述	7-3
第 2 课: Azure 存储表概述	7-8
第 3 课: 表实体事务	7-16
实验 A: 在 Azure 存储表中存储活动注册数据	7-19
实验 B: 在 Azure 存储表中存储活动注册数据	7-24
章节复习和作业	7-31

章节概述

对于结构化的数据，可以将其保存在 SQL 之类的关系型数据库中，但若是表结构经常修改，定义良好的关系型数据库就无法满足您的要求，这时，可以使用 Azure 存储服务的表存储来实现此需求。本章中的第 1 节“Azure 存储概述”描述了 Azure 存储服务，Azure 中提供的存储服务的类型，Azure 存储服务的备份策略，并介绍了如何访问 Azure 存储中数据的相关知识。第 2 节“Azure 存储表概述”描述了 Azure 存储服务的表服务，介绍了在存储表中存储 NoSQL 数据的知识，本节也演示了如何实现一个表存储。第 3 节“表实体事务”，介绍了存储表对常见事务的支持，包括创建，读取，更新和删除等事务操作，同时，表存储也支持 OData 协议，可以使用 OData 进行查询。

目标

完成本章后，您可以做到：

- 描述 Azure 存储的类型和访问存储数据。
- 理解存储表和如何在存储表中存储 NoSQL 数据。
- 了解表存储的常见事务支持。

第 1 课 Azure 存储概述

Microsoft Azure 存储可用来存储非结构化数据、文件和消息。可以将存储用作云应用程序以及每个服务的不同实例的共享资源。

本节提供存储的简要概述。

课程目标

完成本节后，您可以做到：

- 描述存储服务。
- 识别可用的存储类型。
- 描述存储数据的异地复制。

Azure 存储

有了云计算，兼顾基础结构、复原力和可伸缩性而设计的应用程序现在可以采用灵活的基础结构。在您希望扩展或收缩应用程序时，云服务可自动处理负载平衡和基础结构创建（快速扩展）。

DocumentDB 自动在多个节点之间分区数据，从而增长到可处理您要存储的数据量。

存储是支持所有这些灵活服务的基础服务。存储的设计是为了让开发人员能提供一种可自动扩展来处理应用程序工作负载的存储机制，并借此构建大规模应用程序。存储的可伸缩性极高，可以存储用于数据挖掘、分析和媒体工作负载的大数据集。

存储的性能和需求量也可预测，这有助于规划未来增长。例如，每个 blob、每个磁盘或每个存储帐户的每秒操作数是受限的。这些阈值都已公开，可用来规划应用程序的数据需求量。您可能需要将虚拟机应用程序分布到多个磁盘，以突破每磁盘每秒输入/输出操作数 (IOPS) 阈值。您还可以考虑将媒体文件分散到多个存储帐户来突破每帐户 IOPS 阈值。

存储自动划分数据分区，并在这些生成的分区之间对请求做负载平衡。随着应用程序的增长，存储将自动分配正确数量的分区来进行扩展，并满足日益增长的资源需求量。

使用标准 HTTP URL 还可在全球范围内访问存储。可以配合客户端应用程序使用这些 URL 来让最终用户从您的存储帐户下载多媒体内容。还可以将资源标记为私有来保护存储帐户，以防未经授权的访问。如果客户端应用程序有正确的安全令牌，那么这些私有资源可供临时访问。此安全令牌表示为 URL 查询字符串参数，以确保与几乎任何客户端设备或平台兼容。

Azure 存储

<http://go.microsoft.com/fwlink/?LinkId=525385>

- Azure 存储服务允许将记录、文件和简单请求存储在灵活、托管且可伸缩的解决方案中
- 在规划和扩展云应用程序场景的不同方面时，将数据的存储与应用程序分开可带来更多灵活性
- 存储服务可以大规模伸缩，这样就可以存储大数据集，而不必被迫规划数据的分区和分片

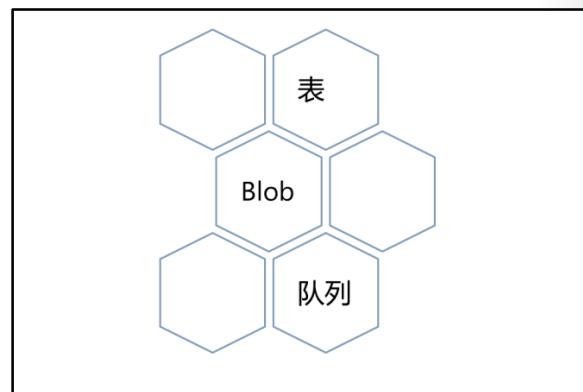
存储类型

存储服务由四种主要存储类型组成。每种类型可在下面所列的唯一端点获得：

[http://\[帐户\].blob.core.windows.net/\[容器\]/\[blob\]](http://[帐户].blob.core.windows.net/[容器]/[blob])
[http://\[帐户\].table.core.windows.net/\[表\]\[\[分区键\],\[行键\]\]](http://[帐户].table.core.windows.net/[表][[分区键],[行键]])
[http://\[帐户\].queue.core.windows.net/\[队列\]](http://[帐户].queue.core.windows.net/[队列])
[http://\[帐户\].file.core.windows.net/\[文件\]](http://[帐户].file.core.windows.net/[文件])

中国版 Windows Azure 中相应的端点如下：

[http://\[帐户\].blob.core.chinacloudapi.cn/\[容器\]/\[blob\]](http://[帐户].blob.core.chinacloudapi.cn/[容器]/[blob])
[http://\[帐户\].table.core.chinacloudapi.cn/\[表\]\[\[分区键\],\[行键\]\]](http://[帐户].table.core.chinacloudapi.cn/[表][[分区键],[行键]])
[http://\[帐户\].queue.core.chinacloudapi.cn /\[队列\]](http://[帐户].queue.core.chinacloudapi.cn /[队列])
[http://\[帐户\].file.core.chinacloudapi.cn /\[文件\]](http://[帐户].file.core.chinacloudapi.cn /[文件])



Blob

存储 Blob 提供了存储文件的方法，使其可由应用程序的其他组件或客户端设备使用。这些 Blob 代表文件，并且可以加以保护，使得需要令牌才能访问 Blob。

表

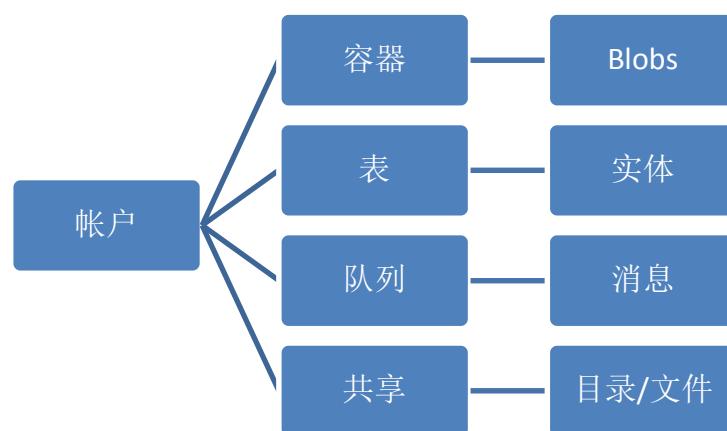
表服务是基于文档范式的 NoSQL 存储。每个实体包含一个分区键和一个行键，两者共同构成一个唯一索引。然后这些实体包含表示文档属性的键值对集合。

文件

文件是发布服务器消息块 (SMB) 2.1 端点的服务。此端点可由 Azure 中的虚拟机像使用共享驱动器一样使用。

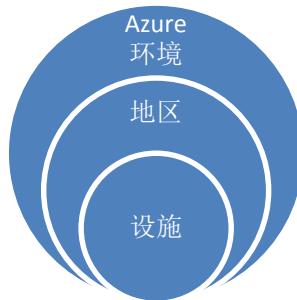
队列

存储队列是外部管理的队列，可持久保留请求，这些请求是应用程序中的模块要使用的。队列是按传统的先入先出 (FIFO) 模式实现的。这些队列可测量，还可以检查（扫视）请求，而无需将请求从队列中移出。



Azure 存储的异地复制

在设置 Azure 存储帐户时，新的存储帐户已经选中了复制选项。复制选项有不同选择，不同的选择可用性、定价和扩展功能也不同。在 Azure 中，地理上分成几个简单概念。首先，Azure 环境由多个地区组成，这些地区通常在地理上是分散的。其次，每个地区包含一个到多个通常称为数据中心的设施。在比较存储的复制选项时，理解这两个概念非常重要。默认情况下，所有存储帐户在各自的数据中心中都有副本，并且这些副本组织成一式三份。另外，不同的复制级别会影响副本。



- **本地冗余存储 (LRS)**
 - 默认选项
 - 数据复制到同一数据中心中的三个不同节点
- **区域冗余存储 (ZRS)**
 - 数据复制到同一地区的三个不同的数据中心
 - 如果地区内没有足够的数据中心可供存储，数据有可能复制到地区边界以外

本地冗余存储 (LRS)

这是最简单也是最便宜的选项。如您所知，在同一数据中心存在数据的三个副本。如果数据不需要任何复制，那么可以使用此选项并明显地节约成本。但是，不利的一面是，虽然数据可能挺过单个物理故障，但是整个数据中心中断通常仍会造成这些数据不可用。

区域冗余存储 (ZRS)

此选项将数据复制到同一地区的另外的数据中心。这确保即使整个数据中心出现故障或中断，您的数据仍然可用。数据仍然只有三个副本，但它们分散在该地区的多个数据中心。

 **注释：**某些地区只包含一个数据中心。这种情况下，ZRS 将数据复制到其他地区的另一个数据中心。例如，2014 年，巴西地区只有一个数据中心。ZRS 会将巴西地区的数据复制到美国德克萨斯州的数据中心。实际上这么做是升级到了 GRS 复制，而成本没有丝毫增加。但是，在使用此选项时需要小心，因为这可能会不符合当地的安全政策。

异地冗余存储 (GRS)

GRS 将数据复制到另一个地区的辅助数据中心。主数据中心仍然有三个副本，但是选用此选项时，新的数据中心也会有三个新副本。这些副本以异步方式提交。主数据中心将先提交新数据，然后返回 HTTP 状态消息，随后辅助数据中心才执行提交。这种数据模式通常称为最终一致。

读取访问地域冗余存储 (RA-GRS)

RA-GRS 的运行方式与 GRS 相同。主要区别是应用程序现在可以用只读方式访问辅助数据。这对于中断期间的读取访问或者跨数据中心分散工作负载非常有用。例如，Web 应用程序可以从主数据中心读取数据和写入数据，而报告应用程序只能从辅助数据中心读取数据。

Azure 存储定价

<http://go.microsoft.com/fwlink/?LinkId=525387>

访问存储数据

Azure 门户未提供可访问存储在存储帐户下的数据的直接界面。若要操作存储帐户，可能必须使用 Azure 团队提供的任一工具，如客户端库、REST API、Windows PowerShell 或 Visual Studio 的服务器资源管理器。还可以使用某些第三方工具来与存储帐户交互，如 LINQPad 或 Azure Storage Explorer。

LINQPad

<http://go.microsoft.com/fwlink/?LinkId=5253>
88

- 访问资源要使用存储帐户的共享密钥来认证身份
 - 可以配置 Blob 来允许匿名访问
 - 可以生成共享访问签名，让客户端可以受控地临时访问存储资源
- 存储帐户得到两个可随时重新生成的密钥
 - 让您能轮换密钥，并按计划重新生成密钥，而不会让应用程序失去对资源的访问

Azure Storage Explorer

<http://go.microsoft.com/fwlink/?LinkId=525390>

虽然可以使用 URI 直接访问存储数据，但是也可以单独限制匿名用户访问或修改容器和 Blob。

共享访问签名

共享访问签名是一组可由客户端用来访问资源的查询参数。查询参数指定访问级别和到期时间，但是此数据经过签名验证，因此不可手动修改。另外，有库可以用来以特定方式生成签名。

存储访问策略

有时候，需要有计划地生成签名，而不是临时生成。可以定义存储访问策略来创建可用来生成多个共享访问签名的统一描述。该策略的作用不仅如此，它还提供了集中式控制机制来管理以前生成的签名。如果吊销该策略，它将吊销以前生成的任何签名。在需要保护多个签名，以防泄露敏感数据的情况下，可以使用这种策略。

将副密钥模式（Valet Key pattern）用于 Azure 存储的示例应用程序图。

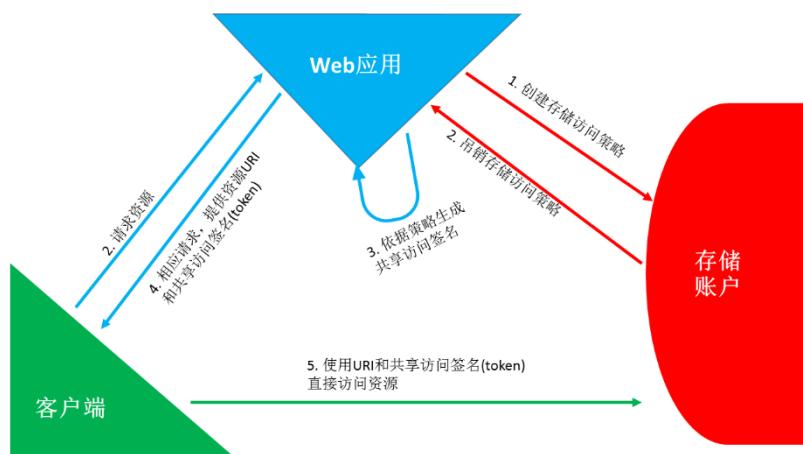
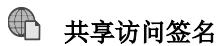


图 7.1：存储副密钥模式

MCT USE ONLY. STUDENT USE PROHIBITED



共享访问签名

<http://go.microsoft.com/fwlink/?LinkID=525392>

第 2 课 Azure 存储表概述

表服务提供了在 Azure 中存储非关系数据库的选项。表服务经过优化，可扩展到容纳大量实体，甚至每秒大量操作。简言之，表服务是考虑到可伸缩性而建立的。

本节介绍表服务并说明表服务为什么很独特。

课程目标

完成本节后，您可以做到：

- 描述表服务。
- 说明表存储中的数据结构和分区。

存储表

表服务提供 NoSQL 数据存储机制，在表中存储松散结构的数据，可以使用唯一分区键来分区这些表。适合表存储的理想数据是需要以极大规模存储并很快访问的极少修改数据或不可变数据。对于传统关系数据库管理系统，可以实现大规模数据存储，但是查询的开销很大。如果要访问表中的某个单项，那就需要某种查询，而随着表大小的增加，查询的开销最终可能呈指数级增长。如果使用基于文档的 NoSQL 系统，如表服务，那么这个问题就可得到解决，因为这种系统使用各种方法（包括索引和哈希）来访问表中的单条记录。这也是为什么表服务可以大量存储数据，而同时仍保持一致地快速访问特定记录的原因。

- 表存储允许存储不受架构或模型限制的灵活数据集。
 - 客户端应用程序可规定存储的数据的模型
 - 不同架构的复杂对象可存储在同一个表中
- 为大规模扩展而建（极大数据集）
- 数据使用分区键分区，以支持跨不同节点的负载平衡
- 数据按分区键和行键的组合索引，以实现极快的查找

在表服务的语境中，表是相关但是未必共有相同架构的实体集。每个表有分片实现（称为分区），同一分区中的数据将位于同一服务器上。这意味着可以非常快速地访问同一分区中的数据，而访问跨分区的数据通常慢得多。表存储的逻辑结构如下：

存储帐户

表 I

分区 A

实体 1

实体 2

实体 3

分区 B

实体 4

分区 C

实体 5

实体 6

表 II

分区 D

实体 7

我们将在本节通篇探讨表存储的结构。

存储表中的 NoSQL 数据

表服务数据模型起初可能令人生畏。表由分区组成。这些分区包含通过分区键相关的数据。分区中每个单独的实体有唯一的行键。在表中，每个实体的分区键和行键组合必须唯一，它们的工作方式与复合索引类似。同一分区中的数据由同一台物理服务器提供，这说明了为什么只扫描一个分区的查询仍然非常高效。只查找一个特定实体的查询需要指定分区键和行键。行键的用处类似索引，可以无需全表扫描或分区扫描就能检索实体。

如果几个分区足够小，可以将它们集中在同一台服务器上。这称为分区范围。虽然应用程序中可能出现此现象，但是围绕这一概念进行设计很难。如果有意需要多个实体，应总是将这些实体设计为共有一个分区键。

请参考第 5 章“设计有复原力的云应用程序”，了解有关分片设计和原则的更多信息。

理解表服务数据模型

<http://go.microsoft.com/fwlink/?LinkId=525393>

为 Azure 表存储设计可伸缩的分区策略

<http://go.microsoft.com/fwlink/?LinkId=525394>

- 表存储是 NoSQL 数据库
 - 实现为键值存储
- 表包含使用分区键划分到多个节点的实体
- 实体有索引，索引是分区键和行键的组合
- 特定实体的属性实现为键值对的集合
 - 键是属性的名称
 - 值是属性的值

演示 1：实现 Azure 存储表

对于此演示，您将使用自己的计算机来完成实验。开始本次演示前，必须完成第 2 章的实验 A。此外，必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：
vm28532[自定义名称].cloudapp.net:[虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**

5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验 A 中使用这些密码和 Azure 帐户。

演示步骤

1. 在 Visual Studio 2013，创建一个新的控制台应用程序项目：
 - 名称：Contoso.Storage.Table
 - 路径：AllFiles (F):\Mod07\Labfiles\Starter
2. 通过使用程序包管理器控制台，添加 **WCF Data Services Client for OData v1-3 NuGet** 程序包：
 - 程序包名称：Microsoft.Data.Services.Client
 - 版本：5.6.0
3. 通过使用程序包管理器控制台，添加 **Windows Azure Storage** NuGet 程序包：
 - 程序包名称：WindowsAzure.Storage
 - 版本：3.1.0.1
4. 打开 **Program.cs** 文件。
5. 给 **Microsoft.WindowsAzure.Storage** 和 **Microsoft.WindowsAzure.Storage.Table** 命名空间添加 **Using** 语句。
6. 在 **Main** 方法里，使用下面代码创建一个 **CloudTableClient** 实例：

```
CloudTableClient tableClient =  
CloudStorageAccount.DevelopmentStorageAccount.CreateCloudTableClient();
```

7. 使用下面代码创建一个 **CloudTable** 实例：

```
CloudTable table = tableClient.GetTableReference("roster");
```

8. 使用下面代码确保表已被创建：

```
table.CreateIfNotExists();
```

9. 在 **Contoso.Storage.Table** 项目里，创建一个新的类：

- 名称：Employee

10. 把 **Employee** 类更新为 **public** 访问。

11. 给 **Microsoft.WindowsAzure.Storage.Table** 命名空间添加 **using** 语句。

12. 让 **Employee** 类继承于 **TableEntity** 类。

13. 添加 **int** 类型的 **YearsAtCompany** 公共属性，且该属性带有 **get** 和 **set** 访问函数。

14. 重写 **Object base** 类的 **ToString** 方法并添加以下代码：

```
return RowKey + "\t\t[" + YearsAtCompany + "]";
```

15. 在 **Program** 类里，在 **Main** 方法最后，使用下面代码创建三个员工：

```
Employee first = new Employee { PartitionKey = "IT", RowKey = "ibahena", YearsAtCompany =  
7 };  
Employee second = new Employee { PartitionKey = "HR", RowKey = "rreeves", YearsAtCompany  
= 12 };  
Employee third = new Employee { PartitionKey = "HR", RowKey = "rromani", YearsAtCompany =  
3 };
```

16. 使用下面代码将 PartitionKey 为 **IT** 的员工插入表中：

```
TableOperation insertOperation = TableOperation.InsertOrReplace(first);
table.Execute(insertOperation);
```

17. 使用下面代码将 PartitionKey 为 **HR** 的员工批量插入到表中：

```
TableBatchOperation batchOperation = new TableBatchOperation();
batchOperation.InsertOrReplace(second);
batchOperation.InsertOrReplace(third);
table.ExecuteBatch(batchOperation);
```

 **注释：**批量操作可以用来插入多个实体到一个 Azure 存储表。这些实体必须具有相同的 **PartitionKey** 以便作为同一批次处理。

18. 查询表中 PartitionKey 值为 **HR** 的员工，并将结果打印到控制台，代码显示如下：

```
TableQuery<Employee> query = new TableQuery<Employee>().Where(
    TableQuery.GenerateFilterCondition("PartitionKey",
        QueryComparisons.Equal, "HR")
);
Console.WriteLine("HR Employees\n");
foreach(Employee hrEmployee in table.ExecuteQuery<Employee>(query))
{
    Console.WriteLine(hrEmployee);
}
```

19. 使用下面代码，检索 PartitionKey 为 **IT** 且 RowKey 为 **ibahena** 的员工，然后将结果打印到控制台。

```
Console.WriteLine("\n\n\n\nIT Employee\n");
TableOperation retrieveOperation = TableOperation.Retrieve<Employee>("IT", "ibahena");
TableResult result = table.Execute(retrieveOperation);
Employee itEmployee = result.Result as Employee;
Console.WriteLine(itEmployee);
```

20. 启动 Windows Azure Storage Emulator – v3.4。

21. 启动控制台应用程序无需调试，查看控制台应用程序的输出。

演示 2：使用中国版 Windows Azure 实现 Azure 表存储

对于此演示，您将使用自己的计算机完成实验。开始本次演示前，必须完成第 2 章的**实验 B**。此外，必须完成以下步骤：

- 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
- 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：
vm28532[自定义名称].chinacloudapp.cn:[虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

- 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。

4. 如果有必要，使用以下用户名密码登录：
 - 用户名: **Student**
 - 密码: **AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到世纪互联运营的 Windows Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

演示步骤

1. 在 Start 屏幕，单击向下箭头按钮查看所有的应用程序。
2. 单击 **Visual Studio 2013** 磁贴。
3. 在文件菜单中，单击**新建**，然后单击**项目**。
4. 在**新建项目**对话框中，执行下面的步骤：
 - a. 展开**模板**，**Visual C#**，然后单击**Windows 桌面**。
 - b. 单击**控制台应用程序模板**。
 - c. **名称**框中，输入值 **Contoso.Storage.Table**。
 - d. 在**位置**框中，指定值 **AllFiles (F):\Mod07\Labfiles\Starter**。
 - e. 单击**确定**。
5. 在**视图**菜单，调出**其它窗口**，然后单击**程序包管理器控制台**。
6. 在控制台中输入下面的命令：

```
Install-Package Microsoft.Data.Services.Client -Version 5.6.0;
```

7. 按 **Enter** 键。
 8. 上面的命令执行完毕后，同样的方法再执行下面的命令：
- ```
Install-Package WindowsAzure.Storage -Version 3.1.0.1;
```
9. 按 **Enter** 键。
  10. 在**解决方案资源管理器**窗格中，展开 **Contoso.Storage.Table** 工程。
  11. 双击打开 **Program.cs** 文件。
  12. 在代码文件的顶部，使用 **using** 加入下面语句：

```
using Microsoft.WindowsAzure.Storage;
```

13. 同样的方法再添加下面的命名空间引用：
- ```
using Microsoft.WindowsAzure.Storage.Table;
```
14. 在 **Main** 方法的结尾，在右括号之前，添加以下代码：
- ```
CloudTableClient tableClient =
CloudStorageAccount.DevelopmentStorageAccount.CreateCloudTableClient();
```
15. 在 **Main** 方法的结尾，在右括号之前，添加以下代码：
- ```
CloudTable table = tableClient.GetTableReference("roster");
```

16. 在 **Main** 方法的结尾，在右括号之前，添加以下代码：

```
table.CreateIfNotExists();
```

17. 在解决方案资源管理器窗格，右击 **Contoso.Storage.Table** 工程，指向添加，然后单击新建项。

18. 在添加新项-**Contoso.Storage.Table** 对话框中，执行下面的步骤

- a. 展开已安装，展开 **Visual C# 项**，然后单击**代码**。
- b. 单击**类**模板。
- c. 在**名称**框中，输入 **Employee.cs**。
- d. 单击**添加**。

19. 在 **Employee** 类中，在类定义的左侧添加 **public** 访问器：

```
class Employee
```

20. 验证更新以后的类定义应当是这样的：

```
public class Employee
```

21. 在代码文件的顶部，添加 **using** 语句：

```
using Microsoft.WindowsAzure.Storage.Table;
```

22. 在 **Employee** 类中，在类定义的右侧添加继承语句：**TableEntity**：

```
public class Employee
```

23. 验证更新后的类定义应当是这样：

```
public class Employee : TableEntity
```

24. 在 **Employee** 类，添加以下代码行：

```
public int YearsAtCompany { get; set; }
```

25. 在 **Employee** 类，添加下面的方法：

```
public override string ToString()
{
}
```

26. 在 **ToString** 方法，添加以下代码行：

```
return RowKey + "\t\t[" + YearsAtCompany + "]";
```

27. 在解决方案资源管理器窗格，展开 **Contoso.Storage.Table** 工程。

28. 双击打开 **Program.cs** 文件。

29. 在 **Main** 方法的结尾和右括号前，增加第一个员工，使用 **IT** 作为 PartitionKey 的值，如下图所示：

```
Employee first = new Employee { PartitionKey = "IT", RowKey = "ibahena", YearsAtCompany =
7 };
```

30. 添加第二个员工，使用 **HR** 作为 PartitionKey 的值，如下图所示：

```
Employee second = new Employee { PartitionKey = "HR", RowKey = "rreeves", YearsAtCompany = 12 };
```

31. 添加第三个员工，使用 **HR** 作为 PartitionKey 的值，如下图所示：

```
Employee third = new Employee { PartitionKey = "HR", RowKey = "rrromani", YearsAtCompany = 3 };
```

32. 在 **Main** 方法的结尾，在右括号之前，创建一个新的 **TableOperation**，插入创建的第一个员工，如下图所示：

```
TableOperation insertOperation = TableOperation.InsertOrReplace(first);
```

33. 下一行，请使用变量 **table** 的 **Execute** 方法来执行 **TableOperation** 操作，如下图所示：

```
table.Execute(insertOperation);
```

34. 在 **Main** 方法的结尾，在右括号之前，用下面的代码创建一个新的 **TableBatchOperation**：

```
TableBatchOperation batchOperation = new TableBatchOperation();
```

35. 在下一行，使用批操作 **InsertOrReplace** 插入新创建的第二个员工，如下所示：

```
batchOperation.InsertOrReplace(second);
```

36. 在下一行，使用批操作 **InsertOrReplace** 插入新创建的第三个员工，如下所示：

```
batchOperation.InsertOrReplace(third);
```

37. 在下一行，使用变量 **table** 的 **ExecuteBatch** 方法执行 **TableBatchOperation** 操作，如下图所示：

```
table.ExecuteBatch(batchOperation);
```

38. 在 **Main** 方法的结尾，在右括号之前，创建一个 **String** 过滤器，使用 **TableQuery.GenerateFilterCondition** 静态方法来检索 PartitionKey 的值为 HR 的所有实体：

```
string queryFilter = TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, "HR");
```

39. 在下一行，创建一个新的 **TableQuery** 并使用 String 过滤器调用其中的方法，如下图所示：

```
TableQuery<Employee> query = new TableQuery<Employee>().Where(queryFilter);
```

40. 下面一行，在控制台窗口中输出标题 HR Employees：

```
Console.WriteLine("HR Employees\n");
```

41. 接下来一行，使用 **foreach** 循环遍历查询的结果：

```
foreach (Employee hrEmployee in table.ExecuteQuery<Employee>(query))  
{  
}
```

42. 在 foreach 循环中，输出 **Employee** 对象到控制台窗口，如下图所示：

```
Console.WriteLine(hrEmployee);
```

43. 在 **Main** 方法的结尾，在右括号前，加入代码以在控制台窗口中输出标题 IT Employees：

```
Console.WriteLine("\n\n\n\nIT Employee\n");
```

44. 在下一行，创建一个新的 **TableOperation** 检索 PartitionKey 的值为 **IT** 并且 RowKey 的值为 **ibahena** 的实体：

```
TableOperation retrieveOperation = TableOperation.Retrieve<Employee>("IT", "ibahena");
```

45. 在下一行，通过执行 **TableOperation** 操作将结果存储在 **TableResult** 变量中：

```
TableResult result = table.Execute(retrieveOperation);
```

46. 接下来一行，将 **TableResult** 对象转换成 **Employee** 对象：

```
Employee itEmployee = (Employee)result.Result;
```

47. 接下来，将 **Employee** 对象输出到控制台：

```
Console.WriteLine(itEmployee);
```

48. 在 Start 屏幕中，单击下拉箭头来查看所有应用程序。

49. 单击 **Windows Azure Storage Emulator – v3.4** 磁贴。

50. 在命令行应用程序完成后，关闭控制台窗口。

51. 切换至 **Contoso.Storage.Table – Microsoft Visual Studio** 窗口。

52. 在调试菜单，单击开始执行（不调试）。

53. 查看控制台的输出。

54. 按任意键退出控制台窗口。

55. 关闭 **Contoso.Storage.Table – Microsoft Visual Studio** 应用程序。

MCT USE ONLY STUDENT USE PROHIBITED

第 3 课 表实体事务

存储表提供了一组对应于传统的创建、读取、更新和删除 (CRUD) 方法的事务性功能。存储表还将 OData 协议用于查询。

本节描述存储表及其实体的常见事务和实体访问注意事项。

课程目标

完成本节后，您可以做到：

- 使用 OData 查询存储表。
- 对存储表的实体使用常用 CRUD 事务。

常见事务

虽然表存储适合不经常更新的数据，但是存储表端点仍然提供了全部 CRUD 操作。可以使用现有客户端库或者采用标准 HTTP 方法的 REST API 端点来进行 CRUD 操作。查询表包括结合 OData 语法使用 **HTTP GET** 方法。

• 表支持对指定实体的常见事务：

- 创建
- 读取
- 更新
- 删除

• 更新实体时，实体行为类似字典

- 有新键的任何键值对添加到实体的属性集合
- 重用现有键的任何键值对替换对应的值

QUERY	https://[帐户].table.core.windows.net/[表]()?\$filter=[查询表达式]
GET	https://[帐户].table.core.windows.net/[表]([分区键], [行键])
PUT	https://[帐户].table.core.windows.net/[表]([分区键], [行键])
POST	https://[帐户].table.core.windows.net/[表]
DELETE	https://[帐户].table.core.windows.net/[表]([分区键], [行键])
MERGE	https://[帐户].table.core.windows.net/[表]([分区键], [行键])

更新时，实体不会丢失发布的对象中未指定的键值对。



注释：在中国版 Windows Azure 中，域名.core.windows.net 要相应更改为 .core.chinacloudapi.cn。

例如，有一个表示学生的实体。学生按其所属学校分区，行键是其学生 ID。在这个特定例子中，学生只有名字和姓氏数据。

分区键	appleorchardmiddle
行键	237548902

分区键	appleorchardmiddle
First Name	Chris
Last Name	Meyer

假设您需要在名为 `students` 的表中，以及名为 `cornfielddistrict` 的存储帐户中更新此用户。您还需要将年龄更新为 12，并将添加表示“Sport”的键值对。在这样的场景中，要使用的 URL 和 JSON 有效内容如下：

PUT: `https://cornfielddistrict.table.core.windows.net/students(appleorchardmiddle, 237548902)`

JSON 有效内容

```
{
  "Age":12,
  "Sport":Tennis
}
```

实体现在类似于下面：

分区键	appleorchardmiddle
行键	237548902
First Name	Chris
Last Name	Meyer
Age	12
Sport	Tennis

OData 查询

使用 OData 协议可以访问表存储。OData 协议提供了在 HTTP 端点中查询项的统一方式。传统上，在 RESTful 服务中，GET 端点可返回特定项，也可返回所有项。OData 查询参数能让您筛选或分页请求的结果，或者将请求的结果投射到 HTTP 端点。OData 协议还可处理很多工作，如请求的类型化和批处理。各种编程语言中都有相应的类库可用。



OData

<http://go.microsoft.com/fwlink/?LinkId=525395>

- 使用 HTTP 终结点和 OData 协议可以查询存储表
 - OData 是一种标准化数据 API 查询方法的 REST 协议
 - 由于该协议是在 REST 协议的基础上建立的，因此仍然提供了创建、读取、更新、删除 (CRUD) 操作

对于存储表，可以将 OData 用作在表中搜索特定实体的轻松方法。目前支持 `$filter`、`$top` 和 `$select` 查询选项。结果可以 AtomPub 或 JSON 格式返回。这是通过在请求中包含 `Accept` 标头来指定的。通常，JSON 数据最高可减少 70% 的带宽。

如果需要特定实体，按照 OData 协议提供的机制，可以使用以下 URL 格式检索集合中的单个实体：

`https://[基础 url]/[资源]([索引])`

存储表与典型的 OData 协议稍有不同，要求您同时指定行键和分区键。这两个键构成了存储表中某项的复合索引，为了获得特定实体，这两个键是必需的：

`https://[帐户].table.core.windows.net/[表](PartitionKey='[键]',RowKey='[键]')`

使用 OData 类库时，应注意它们可能不考虑这一细小的偏差。这些类库可能还支持存储表当前尚不支持的其他 OData 查询选项，如 `$skip` 和 `$expand`。

OData 端点

存储在存储表中的所有表都有标准 OData 端点。这与传统 OData 键查找有些小区别。在表服务中，键是复合键，要求您同时传入分区键和行键。

- 基础 URL：
 - `https://[帐户].table.core.windows.net`
- 按分区键和行键获得实体：
 - `[GET] /[表]([分区键],[行键])`
- 查询表以获得与表达式匹配的实体：
 - `[GET] /[表]()?$filter=[查询表达式]`
- 删除实体：
 - `[DELETE] /[表]([分区键],[行键])`
- 插入或替换实体：
 - `[PUT] /[表]([分区键],[行键])`



注释：在中国版 Windows Azure 中，域名.core.windows.net 要相应更改为.core.chinacloudapi.cn。

实验 A：在 Azure 存储表中存储活动注册数据

场景

即使活动注册数据可以存储在 SQL 中，您仍有自己独特的需求。每个活动都需要不同的注册表单，这些表可能随时更改。基本上，注册数据可能是任何架构。而 SQL 之类的关系数据库需要定义良好的架构。但由于业务需求的原因，您需要的数据库应该能存储结构（或架构）灵活的项。为便于达到此要求，您选择使用 Azure 表存储来存储活动注册数据。

 **注释：**本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成实验 A；如果您拿到的是中国版 Windows Azure 帐户，请转到实验 B。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 使用 Azure Storage SDK 创建表存储客户端。
- 使用 Azure Storage SDK 创建表。
- 使用 Azure Storage SDK 将实体添加到表。
- 使用 Azure Storage SDK 查询实体。
- 使用 Visual Studio 2013 中的服务器资源管理器（Server Explorer）查看表存储实体。

实验设置

预计时间：45 分钟

在开始这个实验之前，必须完成第 2 章的实验 A。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].cloudapp.net: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为**Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验 A 中使用这些密码和 Azure 帐户。

本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成实验 A；如果您拿到的是中国版 Windows Azure 帐户，请转到实验 B。具体请咨询培训讲师，并在讲师的指导下完成实验。

MCT USE ONLY. STUDENT USE PROHIBITED

练习 1：用注册者姓名填写注册表单

场景

在此练习中，您将：

- 使用 **CloudTableClient** 实例查询表实体。
- 对一组实体使用 LINQ-to-Object。

此练习的主要任务如下：

- 创建 CloudTable 类的实例。
- 按分区键检索强类型注册记录。
- 使用 LINQ-to-Object 获取注册者姓名列表。

► 任务 1：创建 CloudTable 类的实例

- 根据以下路径打开 **Contoso.Events** 解决方案：
 - 文件路径：Allfiles (F):\Mod07\Labfiles\Starter\Tables\Contoso.Events
- 在 **Contoso.Events.Worker** 项目中，打开文件 **TableStorageHelper.cs**。
- 从方法 **GetRegistrantNames** 中删除现存的代码。
- 使用变量 **_tableClient** 并使用带有以下参数的 **GetTableReference** 方法，创建一个新的 CloudTable 实例。
 - 表名： **EventRegistrations**

► 任务 2：按分区键检索强类型注册记录

- 使用下面代码，创建一个 **String** 过滤器，找到 PartitionKey 值等于由 **GetRegistrantsNames** 方法的参数提供的 **eventKey** 的实体。

```
string partitionKey = eventKey;
string filter = TableQuery.GenerateFilterCondition("PartitionKey",
    QueryComparisons.Equal, partitionKey);
```

- 创建一个带有过滤功能的 **TableQuery** 并通过表变量里的 **ExecuteQuery** 方法来运行查询，如下面代码显示：

```
TableQuery<Registration> query = new TableQuery<Registration>().Where(filter);
IEnumerable<Registration> registrations = table.ExecuteQuery<Registration>(query);
```

► 任务 3：使用 LINQ-to-Object 获取注册者姓名列表

- 创建一个 LINQ 查询，让 **registrations** 对象先以 **LastName** 字段排序，再以 **FirstName** 字段排序，然后返回格式像这样 “**{LastName}, {FirstName}**” 的字段，代码显示如下：

```
IEnumerable<string> names = registrations
    .OrderBy(r => r.LastName)
    .ThenBy(r => r.FirstName)
    .Select(r => String.Format("{1}, {0}", r.FirstName, r.LastName));
```



注释：也可以使用 LINQ-to-Objects 的查询语法，如下面代码显示：

```
IEnumerable<string> names = from r in registrations
    orderby r.LastName, r.FirstName
    select r.FirstName + ", " + r.LastName;
```

2. 返回 LINQ 查询的值，作为 **GetRegistrantsNames** 方法的结果。

结果：完成此练习后，您将按行键或分区键从表存储中查询了实体。

练习 2：将活动网站更新为使用存储表

场景

在此练习中，您将：

- 将现有 Contoso.Events 应用程序更新为使用表存储。

此练习的主要任务如下：

1. 更新注册控制器操作来存储注册记录。
2. 更新注册 ViewModel 来从表中检索动态存根注册。

► **任务 1：更新注册控制器操作来存储注册记录**

1. 在 **Contoso.Events.Web** 项目中，打开 **RegisterController.cs** 文件。
2. 从 **StoreRegistration** 方法中删除已存在的代码。
3. 通过使用存储在云配置文件的 **Microsoft.WindowsAzure.Storage.ConnectionString** 创建 **CloudStorageAccount** 变量，如下面代码显示：

```
string connectionString =
CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Storage.ConnectionString");
var storageAccount =
Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

4. 通过使用存储帐户和一个名为 **EventRegistrations** 的存储表创建一个 **CloudTable** 变量，如下面代码显示：

```
var tableClient = storageAccount.CreateCloudTableClient();
var table = tableClient.GetTableReference("EventRegistrations");
```

5. 创建并运行 **TableOperation** 来插入动态对象到存储表，代码显示如下：

```
var operation = TableOperation.Insert(registration);
table.Execute(operation);
```

6. 解析注册为 **System.Guid** 的 **RowKey** 和返回值作为 **StoreRegistration** 方法的结果，代码显示如下：

```
Guid rowKey = Guid.Parse(registration.RowKey);
Return rowKey;
```

► **任务 2：更新注册 ViewModel 来从表中检索动态存根注册**

1. 在 **Contoso.Events.ViewModels** 项目中，打开 **RegisterViewModel.cs** 文件。
2. 定位到有单个字符串参数的 **RegisterViewModel** 构造函数。
3. 通过使用存储在云配置文件里的 **Microsoft.WindowsAzure.Storage.ConnectionString** 创建一个 **CloudStorageAccount** 变量，如下面代码显示：

```
string connectionString =
CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Storage.ConnectionString");
var storageAccount =
Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

4. 通过使用存储帐户和一个名为 **EventRegistrations** 的存储表创建一个 *CloudTable* 变量，如下面代码显示：

```
var tableClient = storageAccount.CreateCloudTableClient();
var table = tableClient.GetTableReference("EventRegistrations");
```

5. 创建一个 **String** 过滤器，找到由 **GetRegistrantsNames** 方法的参数提供的带有一个 **PartitionKey** 相当于 **eventKey** 的实体，代码显示如下：

```
string partitionKey = String.Format("Stub_{0}", this.Event.EventKey);
string filter = TableQuery.GenerateFilterCondition("PartitionKey",
    QueryComparisons.Equal, partitionKey);
```

6. 创建一个带有过滤器的 **TableQuery** 并通过使用表变量的 **ExecuteQuery** 方法来运行查询，代码显示如下：

```
TableQuery query = new TableQuery().Where(filter);
IEnumerable<DynamicTableEntity> tableEntities = table.ExecuteQuery(query);
```

7. 因为查询只返回一个单一的实体，使用 **SingleOrDefault** 扩展方法来获得 *DynamicTableEntity* 变量，然后传递变量到 **DynamicEntity.GenerateDynamicItem** 静态方法。把 **RegisterViewModel** 的 **RegistrationStub** 属性设置为方法调用的结果，代码显示如下：

```
DynamicTableEntity tableEntity = tableEntities.SingleOrDefault();
this.RegistrationStub = DynamicEntity.GenerateDynamicItem(tableEntity);
```

结果：完成此练习后，您将使用 Azure Storage SDK 检索并创建了实体。

练习 3：验证活动网站正在使用 Azure 存储表来存储注册数据

场景

在此练习中，您将：

- 使用 Azure 存储模拟器来测试正在开发中的存储实体。
- 使用 Visual Studio 来调试使用存储的应用程序。

此练习的主要任务如下：

1. 运行数据生成控制台项目来用数据填充 Azure 存储表。
2. 使用 Visual Studio 2013 中的服务器资源管理器来查看表存储注册数据。
3. 调试云 Web 项目来注册活动。
4. 使用 Visual Studio 2013 中的服务器资源管理器来查看新的表存储注册数据。

► 任务 1：运行数据生成控制台项目来用数据填充 Azure 存储表

1. 定位到 **Contoso.Events.Data.Generation** 项目。
2. 调试 **Contoso.Events.Data.Generation** 项目。

► 任务 2：使用 Visual Studio 2013 中的服务器资源管理器来查看表存储注册数据

1. 在服务器资源管理器窗格中，打开 **Azure** 节点。
2. 在开发 Azure 存储帐户，打开 **EventRegistrations** 表。
3. 探索通过控制项目自动生成的表实体。

- ▶ **任务 3：调试云 Web 项目来注册活动**
 1. 调试 **Contoso.Events.Cloud** 云项目。
 2. 使用网站注册一个活动。
- ▶ **任务 4：使用 Visual Studio 2013 中的服务器资源管理器来查看新的表存储注册数据**
 1. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
 2. 打开**服务器资源管理器**窗格中的 **Azure** 节点。
 3. 打开在开发 Azure 储存帐户中的 **EventRegistrations** 表。
 4. 定位到你最新建的 **Registration** 实体。
 5. 关闭 **Internet Explorer** 和 **Contoso.Events – Microsoft Visual Studio** 应用程序。

结果：完成此练习后，您将使用 Visual Studio 和 Azure 模拟器创建了适用于 Azure 存储的综合开发环境。

问题：由于 Azure Storage SDK 共用一个用于所有存储类型的 **CloudStorageAccount** 类，那么有什么方法可以重构应用程序来利用这一点？

MCT USE ONLY. STUDENT USE PROHIBITED

实验 B：在 Azure 存储表中存储活动注册数据

场景

即使活动注册数据可以存储在 SQL 中，您仍有自己独特的需求。每个活动都需要不同的注册表单，这些表可能随时更改。基本上，注册数据可能是任何架构。而 SQL 之类的关系数据库需要定义良好的架构。但由于业务需求的原因，您需要的数据库应该能存储结构（或架构）灵活的项。为便于达到此要求，您选择使用 Azure 表存储来存储活动注册数据。

目标

在完成本实验以后，您将能学到：

- 使用 Azure Storage SDK 创建表存储客户端。
- 使用 Azure Storage SDK 创建表。
- 使用 Azure Storage SDK 来向表中添加实体。
- 使用 Azure Storage SDK 来查询实体。
- 使用 Visual Studio 2013 的服务器资源管理器来查看 Table 存储中的实体。

预计时间：45 分钟

在开始这个实验之前，必须完成第 2 章的**实验 B**。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：
vm28532[自定义名称].chinacloudapp.cn: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到世纪互联运营的 Windows Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：用注册者姓名填写注册表单

场景

本练习中，您会：

- 使用 **CloudTableClient** 实例来查询表中的实体。
- 在实体集合中使用 LINQ-to-Objects。

MCT USE ONLY. STUDENT USE PROHIBITED

此练习的主要任务如下：

1. 创建 **CloudTable** 类的实例。
2. 按分区键检索强类型注册记录。
3. 使用 LINQ-to-Objects 获取注册者姓名列表。

► 任务 1：创建 **CloudTable** 类的实例

1. 在 Start 屏幕单击 **Desktop** 磁贴。
2. 在任务栏上单击 **File Explorer** 图标。
3. 在 This PC 窗口，打开 **Allfiles (F):\Mod07\Labfiles\Starter\Tables\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格上，展开 **Roles** 目录。
5. 在解决方案资源管理器窗格上，展开 **Contoso.Events.Worker** 项目。
6. 双击 **TableStorageHelper.cs** 文件。
7. 在 **TableStorageHelper** 类，找到如下签名的方法：

```
public IEnumerable<string> GetRegistrantNames(string eventKey)
```

8. 删除代码类中的下面这一行：

```
return Enumerable.Empty<string>();
```

9. 在 **GetRegistrantNames** 方法的结尾和右括号之前，创建一个 **CloudTable** 实例：

```
CloudTable table = _tableClient.GetTableReference("EventRegistrations");
```

► 任务 2：按分区键检索强类型注册记录

1. 在 **GetRegistrantsNames** 方法的结尾和右括号之前，将 **eventKey** 存储在一个命名为 **partitionKey** 的 **string** 变量中：

```
string partitionKey = eventKey;
```

2. 使用 **TableQuery.GenerateFilterCondition** 创建一个 **string** 过滤器：

```
string filter = TableQuery.GenerateFilterCondition("PartitionKey",
    QueryComparisons.Equal, partitionKey);
```

3. 在 **GetRegistrantsNames** 的方法结尾和右括号之前创建一个 **TableQuery** 类实例，使用 **Where** 方法过滤来生成一个查询：

```
TableQuery<Registration> query = new TableQuery<Registration>().Where(filter);
```

4. 将生成的查询作为参数传递给 **table** 变量的 **ExecuteQuery** 方法，使用 **Query** 实体作为泛型参数：

```
IEnumerable<Registration> registrations = table.ExecuteQuery<Registration>(query);
```

► 任务 3：使用 LINQ-to-Objects 获取注册者姓名列表

1. 在 **GetRegistrantsNames** 方法的结尾和右括号之前添加一行没有分号结尾的语句，将注册信息保存到变量 **names**。

```
IEnumerable<string> names = registrations
```

2. 向上一行中追加 lambda 表达式查询语法，使用 **LastName** 属性来对结果排序：

```
.OrderBy(r => r.LastName)
```

3. 继续追加 lambda 表达式查询语法，使用 **FirstName** 属性来对结果进行二次排序：

```
.ThenBy(r => r.FirstName)
```

4. 使用选择操作来结束 lambda 表达式查询，通过静态方法 **String.Format** 来格式化字符串让 **FirstName** 跟在 **LastName** 之后，并以逗号分隔：

```
.Select(r => String.Format("{1}, {0}", r.FirstName, r.LastName));
```

5. 在 **GetRegistrantsNames** 方法的结尾和右括号之前，添加以下语句：

```
return names;
```

结果：在完成本次练习以后，您将了解如何按行键或分区键从表存储中查询实体。

练习 2：将活动网站更新为使用存储表

场景

在本次练习中，您会：

- 将现有 Contoso.Events 应用程序更新为使用表存储。

此练习的主要任务如下：

- 更新注册控制器来存储注册记录。
- 通过更新注册的 ViewModel 来从表中检索动态存根。

► 任务 1：更新注册控制器来存储注册记录

- 在解决方案资源管理器窗格，展开 **Roles** 目录。
- 在解决方案资源管理器窗格，展开 **Contoso.Events.Web** 项目。
- 在 **Contoso.Events.Web** 项目，展开 **Controllers** 文件夹。
- 双击打开 **RegisterController.cs** 文件。
- 在 **RegisterController** 类中，找到下面签名的方法：

```
private Guid StoreRegistration(dynamic registration)
```

- 删掉下面这一行：

```
return Guid.Empty;
```

- 在 **StoreRegistration** 方法结尾和右括号之前使用 **CloudConfigurationManager.GetSetting** 静态方法和 **Microsoft.WindowsAzure.Storage.ConnectionString** 参数来获取连接字符串。

```
string connectionString =
CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Storage.ConnectionString");
```

- 使用静态方法 **CloudStorageAccount.Parse** 和连接字符串作为参数获取存储帐号：

```
var storageAccount =
    Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

9. 在 **StoreRegistration** 方法的结尾和右括号之前使用存储帐号的 **CreateCloudTableClient** 方法创建一个 *CloudTableClient* 变量:

```
var tableClient = storageAccount.CreateCloudTableClient();
```

10. 通过使用 *CloudTableClient* 变量的 **GetTableReference** 方法作为参数, 创建 *CloudTable* 变量:

```
var table = tableClient.GetTableReference("EventRegistrations");
```

11. 在 **StoreRegistration** 方法的结尾和右括号之前, 通过使用 **TableOperation.Insert** 静态方法和 *registration* 作为参数创建一个新的 TableOperation:

```
var operation = TableOperation.Insert(registration);
```

12. 使用 *CloudTable* 变量传递 **TableOperation** 参数来执行 **Execute** 方法:

```
table.Execute(operation);
```

13. 在 **StoreRegistration** 方法结尾和右括号之前, 通过调用 **Guid.Parse** 静态方法, 使用 *registration.RowKey* 字符串作为参数来生成 **Guid**:

```
Guid rowKey = Guid.Parse(registration.RowKey);
```

14. 将 *rowKey* 作为 **StoreRegistration** 方法的结果返回。

```
return rowKey;
```

► 任务 2: 通过更新注册的 **ViewModel** 来从表中检索动态存根

- 在解决方案资源管理器窗格上, 展开 **Shared** 目录。
- 在解决方案资源管理器窗格上, 展开项目 **Contoso.Events.ViewModels**。
- 双击打开 **RegisterViewModel.cs** 文件。
- 在 **RegisterViewModel** 类中, 定位到如下签名的方法中:

```
public RegisterViewModel(string eventKey)
```

- 在 **RegisterViewModel** 构造函数结尾和右括号之前, 使用 **CloudConfigurationManager.GetSetting** 静态方法和 **Microsoft.WindowsAzure.Storage.ConnectionString** 值作为参数获取连接字符串:

```
string connectionString =
    CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Storage.ConnectionString");
```

- 使用 **CloudStorageAccount.Parse** 静态方法, 使用连接字符串作为参数来获得存储帐户:

```
var storageAccount =
    Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

- 在 **RegisterViewModel** 构造函数的结尾和右括号之前, 通过使用存储帐户的 **CreateCloudTableClient** 方法创建一个 **CloudTableClient** 变量:

```
var tableClient = storageAccount.CreateCloudTableClient();
```

8. 通过使用 `CloudTableClient` 变量的 **GetTableReference** 方法和“**EventRegistrations**”作为参数创建 `CloudTable` 变量:

```
var table = tableClient.GetTableReference("EventRegistrations");
```

9. 在 **RegisterViewModel** 构造函数的结尾和右括号之前将 `eventKey` 存储到命名为 **partitionKey** 的 `string` 变量中。

```
string partitionKey = String.Format("Stub_{0}", this.Event.EventKey);
```

10. 使用 **TableQuery.GenerateFilterCondition** 创建一个 `String` 过滤器:

```
string filter = TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, partitionKey);
```

11. 在 **RegisterViewModel** 构造函数的结尾和右括号之前, 创建 **TableQuery** 类的新实例, 并使用 **Where** 方法联合 `String` 过滤器生成一个查询:

```
TableQuery query = new TableQuery().Where(filter);
```

12. 将生成的查询作为参数传递进 `table` 变量的 **ExecuteQuery** 方法中:

```
IEnumerable<DynamicTableEntity> tableEntities = table.ExecuteQuery(query);
```

13. 在 `RegisterViewModel` 构造函数结尾和右括号之前, 在 **DynamicTableEntity** 集合中选择单个 **DynamicTableEntity** 对象元素:

```
DynamicTableEntity tableEntity = tableEntities.SingleOrDefault();
```

14. 通过 **DynamicEntity.GenerateDynamicItem** 方法联合新创建的 `DynamicTableEntity` 对象来设置 **RegistrationStub** 属性的值:

```
this.RegistrationStub = DynamicEntity.GenerateDynamicItem(tableEntity);
```

结果: 完成此练习后, 您将使用 Azure Storage SDK 检索并创建了实体。

练习 3: 验证活动网站使用 Azure 存储表来存储注册数据

场景

在此练习中, 您将:

- 使用 Azure 存储模拟器来测试正在开发中的存储实体。
- 使用 Visual Studio 来调试使用存储的应用程序。

此练习的主要任务如下:

1. 运行生成数据的控制台项目来向 Azure 存储表中填充数据。
2. 使用 Visual Studio 2013 中的服务器资源管理器来查看存储表的注册数据。
3. 调试云 Web 项目来注册活动。
4. 使用 Visual Studio 2013 中的服务器资源管理器来查看存储表中的新的注册数据。

► 任务 1: 运行生成数据的控制台项目来向 Azure 存储表中填充数据

1. 在解决方案资源管理器窗格中, 展开 **Shared** 解决方案目录。

MCT USE ONLY. STUDENT USE PROHIBITED

2. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Data.Generation** 项目。
3. 在解决方案资源管理器窗格中，右击 **Contoso.Events.Data.Generation** 项目，选择 **调试**，然后单击 **启用新实例**。

 **注释：**如果弹出保存只读文件对话框，请您单击**覆盖**。

► **任务 2：使用 Visual Studio 2013 中的服务器资源管理器来查看存储表的注册数据**

1. 在视图菜单，单击**服务器资源管理器**。
2. 定位到 **Azure** 节点，单击左侧的箭头展开 Azure 节点。
3. 展开**存储**节点。
4. 如果需要帐号凭据，输入你的 **Microsoft Account**。
5. 展开**存储**节点下的**开发**帐号节点。
6. 展开**开发**帐号节点下的**表**节点。
7. 双击打开 **EventRegistrations** 表。
8. 在 **EventRegistrations [表]**选项卡上，移动滚动条查看实体。
9. 通过双击某一行来查看单个实体的所有属性。
10. 单击**取消**按钮来退出**编辑实体**对话框。
11. 在 Visual Studio 中关闭 **EventRegistrations [表]**选项卡。

► **任务 3：调试云 Web 项目来注册活动**

1. 在解决方案资源管理器窗格中，右击 **Contoso.Events.Cloud** 项目，然后单击**设为启动项目**。
2. 在**调试**菜单，单击**启动调试**。
3. 如果**存储模拟器**已经运行，会提示您关闭并使用不同的模式重启。单击**确定**即可。
4. 在 Web 应用程序的首页，验证显示的活动列表。
5. 在列表中单击任意的活动。
6. 在活动的 Web 页面，单击 **Register Now**。
7. 在所有的文本框填充相应的信息，然后单击 **Submit**。
8. 关闭打开网站的选项卡。

► **任务 4：使用 Visual Studio 2013 中的服务器资源管理器来查看存储表中的新的注册数据**

1. 切换至 **Contoso.Events – Microsoft Visual Studio** 窗口。
2. 在视图菜单，单击**服务器资源管理器**。
3. 定位到 **Azure** 节点。
4. 在 **Azure -> 存储 -> (开发) -> 表**节点下，双击 **EventRegistrations**。
5. 在 **EventRegistrations [表]**选项卡，拖动滚动条查看实体。
6. 在 **EventRegistrations [表]**选项卡顶部黄色提示框，单击**单击此处**链接。
7. 通过双击某一行来查看单个实体的其它属性，然后单击**取消**。
8. 单击 **Timestamp** 标题两次，实体将会根据 Timestamp 降序排序。
9. 在表的顶部，找到新添加的实体。

10. 切换至 **Contoso.Events – Microsoft Visual Studio** 窗口。
11. 关闭 **Contoso.Events – Microsoft Visual Studio**。

结果: 完成本练习后，你会使用 Visual Studio 和 Azure 的模拟器来创建 Azure 存储的综合开发环境。

MCT USE ONLY. STUDENT USE PROHIBITED

章节复习和作业

本章中，我们介绍了存储以及可用的三种基本存储类型。您还看到了演示如何使用 Azure Storage SDK 来与表存储交互的示例。

复习问题

问题：在实施数据库分片时，为什么联合分发键（federation distribution key）非常重要？此键如何帮助促进性能？

第 8 章 存储和使用 Azure 存储中的文件

目录:

章节概述	8-2
第 1 课: 存储 Blob	8-3
第 2 课: 控制对存储 Blob 和容器的访问	8-6
第 3 课: 配置 Azure 存储帐户	8-10
第 4 课: Azure 文件	8-12
实验 A: 在 Azure 存储 Blob 中存储生成的文档	8-14
实验 B: 在中国版 Windows Azure 的存储 Blobs 中存储生成的文档	8-20
章节复习和作业	8-29

章节概述

在您希望扩展到其他云实例时，将文件存储到本地磁盘会难以维护，并最终变成不可靠的存储方法。

Azure 提供的 Blob 存储机制不仅带来了高性能，而且支持与 Microsoft Azure 内容传送网络 (CDN) 集成，以实现低延迟下载。第 1 节“存储 Blob”描述 Blob 服务以及 Blob 支持的类型。第 2 节“控制对存储 Blob 的访问”详细介绍可保护和授权临时访问 Blob 或容器的方法。第 3 节“配置 Azure 存储帐户”讨论适用于存储 Blob 的某些独特配置选项。第 4 节“Azure 文件”简要介绍 Azure 文件服务。

目标

完成本章后，您可以做到：

- 描述 Microsoft Azure 存储中的 Blob 服务。
- 识别可用于 Blob 的软件开发包 (SDK) 库、命名空间和类。

第 1 课 存储 Blob

Blob 存储提供了 Azure 中的文件和数据存储机制。这种机制是为了速度和便利性而设计，而且支持 CDN 选项。

本节介绍 Blob 服务及其基本概念。

课程目标

完成本节后，您可以做到：

- 描述 Blob 服务。
- 说明 Blob 服务中容器和 Blob 的结构。

存储 Blob

存储 Blob 是存储服务的独特组件，能让您以可靠、可伸缩的方式存储应用程序所需的各种文件。所有存储的 Blob 都与一个 HTTP URL 相关联，通过该 URL 可以从任何位置、平台或设备访问这些 Blob。Blob 服务旨在存储大量非结构化数据或文件。这些文件可使用 URL 公开，也可使用 SAS 令牌防止匿名访问。

Blob 存储的常见用途包括：

- 将图像和文档直接提供给浏览器
- 存储文件供分布式访问
- 流传送视频和音频
- 执行安全备份和灾难恢复
- 存储数据，供企业内部托管或 Azure 托管的服务进行分析

Blob 服务包含以下组件：

- 可以使用 Blob 存储来存储大量非结构化文本或二进制数据。
- 可以使用 Blob 存储来存储文件，如：
 - 虚拟硬盘驱动器
 - 视频
 - 图像
 - 日志文本文件
- 可以使用 Blob 存储将 Blob 分组到逻辑分层容器中
- 可以保护 Blob 并使其可供匿名访问

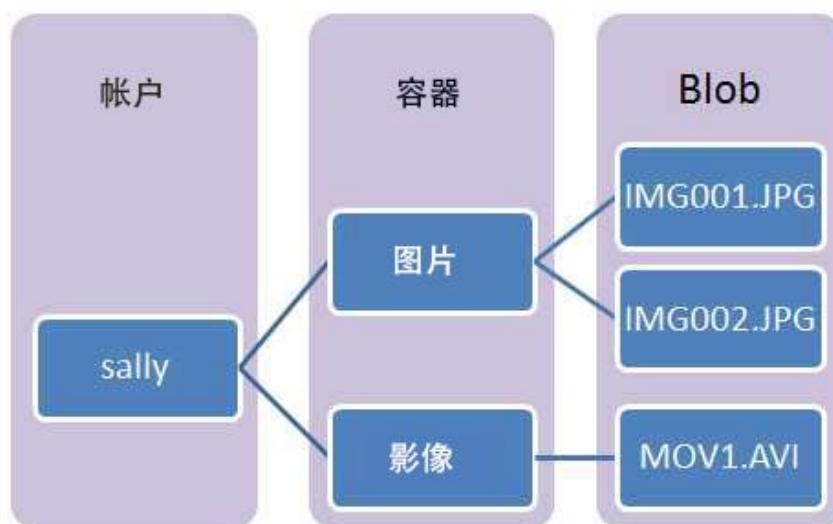


图 8.1：存储 BLOB 层次结构

- **存储帐户。** 存储帐户是存储中最高级别的对象封装单元。存储帐户可包含 Blob、队列、表或文件共享。
- **容器。** 容器表示有着共同安全作用域的一组 Blob。Blob 必须存在于一个容器中。容器不可嵌套。
- **Blob。** Blob 表示容器中的单个文件。有两种主要类型的 Blob，页 Blob 和块 Blob。
- **URL 格式。** 使用以下 URL 格式可以访问 Blob:

`http[s]://<存储帐户>.blob.core.windows.net/<容器>/<blob>`

`http[s]://<存储帐户>.blob.core.chinacloudapi.cn /<容器>/<blob>` (中文版 Windows Azure)

可以使用下面的示例 URL 访问上图中的某个 Blob:

<http://sally.blob.core.windows.net/movies/MOV1.AVI>

Blob 类型

存储容器可包含两种类型的 Blob：块 Blob 和页 Blob。创建 Blob 时，必须指定 Blob 类型。创建 Blob 后，Blob 类型就将永久固定，您只能更新 Blob 的属性和内容。例如，可以将一个块或块的集合写入块 Blob，或者将页写入页 Blob。

块 Blob

块 Blob 为有效上传大文件而优化。块 Blob 由块的集合组成，这些块使用块 ID 来标识。修改块 Blob 的方式是写入一个块集合。该块集合然后使用提交操作和集合中块的 ID 来提交操作。整个过程中，当一个或多个块上传到块 Blob 后，它们与 Blob 关联，但还尚未提交。然后您必须再次使用集合中块的 ID 来将这个块集合提交到块 Blob。未提交的块可以选择性放弃。块 Blob 通常用于大多数文件和多媒体。

有两种主要类型的 Blob:

页 Blob 块 Blob

页 Blob

页 Blob 是由单独的页组成的 Blob，这些页为随机读写操作而优化。这些单独的页有 512 个字节。在创建页 Blob 时，必须指定 Blob 的最大大小。如果文件需要更多空间，必须创建新的页 Blob。若要更新页 Blob，必须写入一个或多个 512 字节的页。写入页时，需要一个偏移量以及范围，范围必须在 512 字节页边界以内。在更新页 Blob 时，可以覆盖最多 4 兆 (MB) 的页。页 Blob 写操作立即提交并原地覆盖页。页 Blob 通常用于虚拟硬盘，并且最高可达 1 百万兆 (TB)。

存储 Blob 的 REST API

一组广泛的用于存储的 REST API 已经推出。使用此 API 可用 RESTful 的方式管理帐户。此 API 已针对 Blob 而扩展，以确保使用简单 URL 即可轻松访问 Blob。

可以使用 **GET**、**PUT**、**POST** 或 **DELETE** HTTP 方法访问 Blob。使用以下 URL 格式可以访问 Blob：

https://[帐户名].blob.core.windows.net/[容器名]/[blob 名]

https://[帐户名].blob.core.chinacloudapi.cn/[容器名]/[blob 名] (中文版 Windows Azure)

还可以提供额外的查询字符串参数来指定更多元数据和功能，如：

- **Snapshot**。检索指定日期时间值的 Blob 快照副本
- **Restype**。可直接配合容器使用来访问容器元数据或枚举容器中的 Blob

除了传统的创建、读取、更新、删除(CRUD)功能外，可以使用 REST API 访问 Blob 的元数据和属性。最后，使用同样的 REST API 还可以管理访问策略(ACL)。

- Blob 的端点是所有存储 REST 端点中最小的。
 - GET BLOB
 - [https://\[账户\].blob.core.windows.net/\[容器\]/\[blob\]](https://[账户].blob.core.windows.net/[容器]/[blob])
 - POST、PUT 和 DELETE 可在同一端点上使用
 - 可以使用 **restype** 查询字符串参数访问容器进行操作
 - [https://\[账户\].blob.core.windows.net/\[容器\]?restype=container](https://[账户].blob.core.windows.net/[容器]?restype=container)
 - 可以将共享访问签名 (SAS) 令牌附加到 URL 最后，以访问受保护的 Blob
- 注意：在中文版 Windows Azure 中，域名要换成：
 - blob.core.chinacloudapi.cn

第 2 课 控制对存储 Blob 和容器的访问

有时候，文件需要受到保护，以防匿名实体访问。存储 Blob 包含的机制允许使用查询字符串令牌来保护 Blob 以及授予临时访问权限。

本节介绍 Blob 权限和 SAS 令牌的概念。

课程目标

完成本节后，您可以做到：

- 对容器或 Blob 实施安全性。
- 生成 SAS 令牌。
- 创建可用来生成 SAS 令牌的共享访问策略。

容器权限

通常，只有存储帐户的所有者才能访问该帐户下的资源。如果服务或应用程序需要使这些资源可供其他客户端使用，可以有多种选择。首先，可以使大家都能使用公共访问密钥。这种做法通常并不推荐，因为此密钥允许个人完全访问您的整个存储帐户及其管理操作。另一种更常见的选项是管理整个容器的访问权限。

公共读取访问容器设置

- 有三种级别的容器访问权限
 - 完全公共读取访问
 - 枚举容器 Blob
 - 读取各个 Blob
 - 不可枚举容器
 - 仅公共读取访问 Blob
 - 读取各个 Blob
 - 无公共读取访问
 - 无法访问 Blob、容器，也不能枚举内容

• 容器公共读访问

	匿名			密钥
	列举容器	列举容器 Blobs	读 Blob	
容器		✓	✓	✓
Blob			✓	✓
Off				✓

图 8.2：公共读取访问

公共读取访问属性控制容器可匿名获取哪些数据。可以选择公共读取访问设置的以下值：

- **容器**。可枚举容器中的 Blob。还可访问容器元数据。使用此设置还可访问该容器内的各个 Blob 及其属性。
- **Blob**。只能访问此容器中的各个 Blob 及其属性。不允许枚举 Blob。
- **关闭**。此设置不允许枚举 Blob。也不能访问各个 Blob 及其属性。必须使用访问密钥才能访问有关此容器或其 Blob 的任何数据。

共享访问签名

共享访问签名 (SAS) 令牌是一组 URL 查询字符串参数和值，它们授予对受限容器、Blob、队列或表的访问权限。SAS 令牌可包含有关访问级别的详细信息，或者对策略的引用，策略中包含那些详细信息。可使用 SAS 公开的部分操作包括：

- 读取或修改页/块 Blob 内容
- 读取或修改 Blob 属性和元数据
- 创建、租用或移除 Blob 快照
- 创建和管理队列消息
- 查阅队列元数据，包括队列长度
- 查询、创建、管理和更新表实体

- 共享访问签名 (SAS 令牌) 是在一段特定时间内允许访问受保护的容器、Blob、队列或表的 URI。
- 允许客户端应用程序无需使用存储帐户密钥，就能访问资源
- 只能配合安全 (HTTPS) 请求使用
- 可以用以下属性生成
 - 开始时间
 - 到期时间
 - 权限级别 (读取、写入、删除、列出、无)

为了应用 SAS 参数，查询字符串参数追加到基础 URL 的末尾，基础 URL 通常用来以匿名方式访问存储资源。此 URL 中的查询参数现在包含授予对受限资源的临时访问权所必需的详细信息。查询字符串包含多种数据，如：

- SAS 令牌的有效时间间隔
- SAS 令牌的开始时间
- 授予的对资源的权限 (读取/写入/读写)
- 验证 SAS 令牌的签名

SAS 令牌还可引用现有的存储访问策略，这种策略提供了对 SAS 令牌的额外控制。可以重用该策略来吊销现有签名或标准化每个生成的签名的访问选项。

使用 SAS 令牌时，应总是在请求中使用 HTTPS。URL 查询参数包含的数据有可能被恶意截获，用在存储帐户上。例如，假如 SAS 令牌的过期时间为 1 天，而您的请求被人截获。截获者可以在其请求中重用 SAS 令牌。该令牌有可能允许其访问同一容器中您没想让别人访问的 Blob。

下面的代码示例创建对容器的访问策略，然后生成该容器的共享访问签名。

创建共享访问密钥

```
BlobContainerPermissions blobPermissions = new BlobContainerPermissions();
blobPermissions.SharedAccessPolicies.Add("mypolicy", new SharedAccessBlobPolicy()
{
    SharedAccessExpiryTime = DateTime.UtcNow.AddHours(10),
    Permissions = SharedAccessBlobPermissions.Write |
        SharedAccessBlobPermissions.Read
});
blobPermissions.PublicAccess = BlobContainerPublicAccessType.Off;
container.SetPermissions(blobPermissions);

string sasToken =
    container.GetSharedAccessSignature(new SharedAccessBlobPolicy(), "mypolicy");
```

可以使用 SAS 令牌，将其附加到此 Blob 资源的 REST URL 的末尾。SAS 令牌传统上就是简单查询字符串值。

存储访问策略

存储访问策略可为共享访问签名带来额外的功能。创建策略能让您分组 SAS 令牌，并控制此策略生成的签名，即使在签名生成以后仍可控制。在这种策略中，可以定义多种设置，如：

- 开始时间
- 到期时间
- 签名权限

使用策略生成的签名可由应用程序吊销，即使在签名已发布到客户端应用程序或设备之后。使用策略的另一个优点是，无需再在请求的 URL 中指定签名的权限和生存期。如果需要更改一个或多个现有签名的这些参数，只需直接更新策略即可。这避免了重新颁发签名。一旦签名泄露，可以轻松吊销与该策略相关的所有签名。

 **注释：**对于为客户端设备生成多个 SAS 令牌的应用程序来说，最佳做法是使用策略，而不是使用单独的签名。存储访问策略允许在现有签名颁发之后仍能修改或移除这些签名。如果不使用策略，则建议最大程度减少签名的生存期，从而最大程度降低现有存储帐户资源的风险。

为了使用存储访问策略，必须先创建策略，然后将其与存储帐户关联。策略必须包含唯一标识符，该标识符为最大长度 64 个字符的字符串。创建策略后，需要在创建新签名时指定该策略的唯一标识符。这些签名现在与策略关联，并将在其查询字符串参数中指定策略。若要更新策略或吊销策略的签名，只需将存储帐户策略列表中的策略替换为唯一标识符相同的新策略即可。

根据策略生成共享访问签名

使用存储访问策略和共享访问签名，可以为存储 Blob 访问实现副密钥模式（Valet Key pattern）。

通常，Web 服务直接访问 Blob 资源，并代表用户下载媒体，然后再发送给用户。

从存储中返回数据的代理 Web 服务。

- 存储访问策略能让您粒度化控制一组共享访问签名
- 签名生存期和权限存储在策略中，而不是存储在 URL 中
- 可以修改策略中的属性，更改会自动传播到根据此策略生成的所有签名
- 还可以使基于某条策略生成的所有签名无效
- 容器、队列或表可有最多 5 条存储访问策略

```
// 获得对容器的引用
var container = blobClient.GetContainerReference("files");
container.CreateIfNotExists();

// 创建 Blob 容器权限
var blobPermissions = new BlobContainerPermissions();
blobPermissions.SharedAccessPolicies.Add("mypolicy", new
    SharedAccessBlobPolicy()
{
    SharedAccessExpiryTime =
        DateTime.UtcNow.AddHours(10),
    Permissions = SharedAccessBlobPermissions.Read
});
blobPermissions.PublicAccess =
    BlobContainerPublicAccessType.Off;
```



图 8.3：存储代理

通过使用副密钥模式，Web 服务可接收来自用户的请求、验证用户，然后返回必要的凭据，让用户直接访问资源。为此，Web 服务使用存储访问策略，并生成共享访问签名。然后该签名附加到 Blob URL 的最后，随后整个 URL 连同签名一起返回给用户。用户然后可使用扩展的 URL，以受控方式直接访问 Azure 存储中的 Blob 资源。

从存储返回 URI 和令牌，并允许客户端直接下载数据的提供者 Web 服务。



图 8.4: 存储提供者

理解共享访问签名模型

<http://go.microsoft.com/fwlink/?LinkId=525392>

MCT USE ONLY. STUDENT USE PROHIBITED

第 3 课 配置 Azure 存储帐户

Azure 存储帐户可启用或配置额外的功能来扩展基本功能。

本节介绍存储帐户的 CDN 和跨源资源共享 (CORS) 功能。

课程目标

完成本节后，您可以做到：

- 启用并使用 CDN 来创建内容的低延迟副本。
- 启用并使用 CORS 以启用跨域客户端脚本支持。

内容传送网络

内容传送网络 (CDN) 将内容放在“更接近”最终用户的地点，从而高效地将内容和多媒体提供给客户端设备。例如，Azure CDN 在美国、欧洲、澳大利亚以及其他地点都有节点。用户将内容从最近的节点下载到其当前位置，这样每个用户就都能尽可能最快地下载内容。通常，如果用户住在远离内容来源的地方，那么他们下载内容的体验要比离内容近的用户差。Azure CDN 可以将内容复制到分在多个地方的多个节点中，让更多用户无论身在何处，都能体会到更高的性能，获得增强的用户体验。

- 可以为现有存储帐户创建 CDN 端点
- 存储内容缓存在离用户更近的边缘服务器
- 还可以为云服务创建 CDN 端点
- CDN 内容可配置为从自定义域提供

Azure CDN 服务跨各个地区的节点复制您存储帐户下的内容。CDN 服务与存储帐户分开，单独计费。为了使用 CDN 服务，需要在存储帐户内创建 CDN 端点。存储帐户提供原始数据，并将其缓存在各个地区的 CDN 节点上。CDN 服务还与 Azure 云服务兼容。CDN 中的内容一直保留在那里，直至过期。这个过期值是使用可配置的生存时间值来确定的。存储帐户或托管服务中所有公开可用的内容都缓存到 CDN。如果修改了任何内容，更改要在 CDN 通过手动介入刷新其内容或缓存的 CDN 内容的生存时间过期后才会可用。

跨源资源共享

如果在单独的域中从一个 Web 应用程序请求存储资源，通常需要发出跨源资源共享（Cross-Origin Resource Sharing）请求。这是一种特殊的类型，允许从源发域的外部访问资源。CORS 请求通常由两个部分组成。第一个，是使用 HTTP 方法 **OPTIONS**，该方法发出行动前请求（*pre-flight request*）。此请求确定主机域是否有权执行跨源请求。响应在其标头值中返回允许的源头和方法。响应的标头可使浏览器确定该请求是否获得允许。验证此请求后，再使用原始 HTTP 方法（**GET**、**PUT**、**POST** 和 **DELETE**）发出第二个请求。Azure 中的 Blob、表和队列服务都支持 CORS。其他 Azure 服务，如搜索和 API 管理也有类似的 CORS 功能。

- 跨源资源共享 (CORS) 允许客户端跨域发出请求。
 - 浏览器和安全策略通常限制这些类型的请求。
- CORS 是 HTTP 请求/响应规范的扩展
 - 由两个请求组成（行动前请求和实际请求）
- CORS 是使用存储帐户级别的规则启用的。
 - CORS 需要选择加入
 - 规则作为允许的来源、方法和标头的策略列表提供。



W3C: 跨源资源共享

默认情况下，存储帐户未启用 CORS。对于每一个存储服务、Blob、表和队列，必须使用自动化脚本，或者任一个门户，逐个配置 CORS 规则。配置 CORS 时，您将指定一系列许可的源头、方法和标头。

以下是一条 CORS 规则的示例：

CORS 规则

```
<Cors>
  <CorsRule>
    <AllowedOrigins>http://www.contoso.com ,
    http://www.fabrikam.com</AllowedOrigins>
    <AllowedMethods>PUT , GET</AllowedMethods>
    <AllowedHeaders>x-ms-meta-data* , x-ms-meta-target* , x-ms-meta-
    abc</AllowedHeaders>
    <ExposedHeaders>x-ms-meta-*</ExposedHeaders>
    <MaxAgeInSeconds>200</MaxAgeInSeconds>
  </CorsRule>
<Cors>
```

CORS 规则是有着特定应有元素的 XML 文档。最常见的元素描述如下：

- **AllowedOrigins**。这是一个域列表，其中的域是获得允许，可以向存储服务发出跨源请求的源头域。源头是通过验证客户端请求中的 Origin 标头来确定的。可以视情况在这里使用通配符来表示允许所有源头。在上面的示例中，通过使用 CORS，域 <http://www.contoso.com> 和 <http://www.fabrikam.com> 就可以向服务发出请求。
- **AllowedMethods**。这是跨源请求中允许的 HTTP 方法的列表。使用 OPTIONS 方法的行动前请求隐式得到允许，因此不需要在此文档中指定。在上面的示例中，只允许 PUT 和 GET 方法。
- **MaxAgeInSeconds**。此元素显示浏览器可对行动前 OPTIONS 请求的响应缓存多久。在需要经常更新存储帐户的 CORS 规则的场合下，此项非常有用。在上面的示例中，如果缓存的 OPTIONS 响应存在时间超过 200 秒，浏览器就必须发出新的行动前请求。

第 4 课 Azure 文件

文件是一种提供 SMB 文件共享的服务，可用于在多个虚拟机之间共享文件。

本节描述文件服务和文件共享逻辑单元。

课程目标

完成本节后，您可以做到：

- 描述文件服务。
- 描述文件共享与 SMB 共享之间的关系。

Azure 文件概述

很多现有或留传下来的应用程序依赖系统上的特定文件结构。虽然本指南是为了将新的云应用程序设计为不强烈依赖于本地存储，但是现实是，很多应用程序已经很强烈地依赖于本地存储。在很多业务环境中，理想的做法是，使用“整体迁移（lift and shift）”过程快速部署应用程序，而不是等到应用程序经过重新设计后再部署到云基础结构。

Azure 文件是存储服务的一个组件。Azure 文件开放可装载在 Azure 服务实例中的 SMB 文件共享。此文件共享可用于各种任务，如在多个虚拟机之间共享数据，或者将数据从本地基础结构迁移到

Azure 虚拟机。Azure 文件是将现有应用程序及其数据从企业内部迁移到 Azure 中的虚拟机集合的常用选项。

运行在 Azure 虚拟机或云服务中的应用程序可装载使用 SMB 2.1 协议的新文件共享。支持该协议的 Linux 和 Windows 操作系统都支持 Azure 文件。装载共享的过程与装载任何其他 SMB 文件共享一样。多种应用程序组件可装载共享，并使用它们来同时共享数据。

Azure 文件公开 REST API 的方式与其他 Azure 服务相似。在企业内部，应用程序使用此 REST API 访问共享中的数据或将数据添加到共享。应用程序使用此 API，可以混合方式操作托管在 Azure 和企业内部的组件。

- Azure 文件是一种公开 SMB 文件共享的服务，这种文件共享可在应用程序之间共享。
- 由于 SMB 是 Windows 中文件共享的通用标准，因此这使得“整体迁移”场景成为可能。
 - VM 可像连接到企业内部的 SMB 共享一样连接到 Azure 文件共享。
 - 将应用程序和工作负载迁移到 Azure VM 的冲突很低的方法。

文件共享

由于文件存储共享是标准 SMB 2.1 文件共享，因此在 Azure 中运行的应用程序可通过文件系统 I/O API 访问共享中的数据。因而开发人员可使用其现有代码和技能迁移应用程序。IT 专业人员可在 Azure 应用程序的管理工作中，使用 Windows PowerShell cmdlet 创建、装载和管理文件存储共享。

分布式应用程序还可使用文件存储来存储和共享有用的应用程序数据、开发工具和测试工具。例如，应用程序可在文件存储共享中存储配置文件和诊断数据，如日志、指标和故障转储，这样这些文件和

- 可以使用 REST API 或 Windows PowerShell 创建文件共享
- 文件共享使用 SMB 2.1 协议
- 文件共享可映射为 Windows 中的驱动器

数据就可用于多个虚拟机或角色。开发人员和管理员可将构建或管理应用程序所需要的实用工具存储在所有组件都可使用的文件存储共享中，而不是在每个虚拟机或角色实例上安装它们。

MCT USE ONLY. STUDENT USE PROHIBITED

实验 A：在 Azure 存储 Blob 中存储生成的文档

场景

您需要一个地方来存储 Contoso Events 应用程序生成的 Word 文档。您决定将生成的 Word 文档存储在 Blob 中。您还决定创建一个受保护的容器，使得匿名用户不能访问这些 Word 文档。最后，您需要创建相应的逻辑来生成 SAS 令牌，以允许临时访问某个 Word 文档。



注释：本章为您提供了两套实验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成实验 A；如果您拿到的是中国版 Windows Azure 帐户，请转到实验 B。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 使用 Azure 管理门户创建容器。
- 使用 Azure 存储 SDK 将 Blob 添加到容器。
- 使用 **MemoryStream** 对象从容器下载 Blob。
- 生成用于访问 Blob 的 SAS 令牌。

实验设置

预计时间：60 分钟

在开始这个实验之前，必须完成第 2 章的实验 A。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].cloudapp.net:[虚拟机 RDP 端口]



注释：虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不確定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

本章为您提供了两套实验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成实验 A；如果您拿到的是中国版 Windows Azure 帐户，请转到实验 B。具体请咨询培训讲师，并在讲师的指导下完成实验。

MCT USE ONLY. STUDENT USE PROHIBITED

练习 1：实现 Azure 存储 Blob

场景

在此练习中，您将：

- 使用管理门户创建 Blob 容器。
- 将 Blob 文件添加到容器。
- 访问容器中的 Blob 文件。

此练习的主要任务如下：

- 登录到 Azure 预览门户
- 使用管理门户创建容器
- 添加和访问容器中的 Blob 文件

► 任务 1：登录到 Azure 预览门户

1. 登录到 Azure 的预览门户网站 (<https://portal.azure.com>)。
2. 切换到中文屏幕。

► 任务 2：使用管理门户创建容器

1. 查看 Storage account 列表。
2. 进入 **stor20532[自定义名称]** 帐户的容器页面。
3. 创建一个新的容器，满足以下条件：
 - 名称： **example**
 - 访问类型： **容器**

► 任务 3：添加和访问容器中的 Blob 文件

1. 打开 Visual Studio 2013。
2. 打开服务器资源管理器窗格中的 **Azure** 节点。
3. 在你已经创建的 storageaccount 中打开 **example** 容器：
 - 帐户名称： **stor20532[自定义名称]**
 - 容器名称： **example**
4. 在容器页面，上载 samplefile 文件：
 - Samplefile 文件位置： **(F):\Mod08\LabFiles\Starter\samplefile.txt**
5. 在 InternetExplorer 中使用下面的网址，来查看 samplefile 的文本内容：
 - [http://\[storageaccount\].blob.core.windows.net/example/samplefile.txt](http://[storageaccount].blob.core.windows.net/example/samplefile.txt)

结果：完成此练习后，您将使用管理门户创建了 Blob 容器，并查看了容器中的 Blob。

练习 2：以文件和媒体填充容器

场景

在此练习中，您将：

- 使用 Azure 存储 SDK 访问存储帐户中的 Blob。

- 使用 Azure 存储 SDK 在存储帐户中创建 Blob。

此练习的主要任务如下:

- 在云服务辅助项目中打开 Blob 帮助器
- 在创建的容器中上传 Word 文档

► 任务 1: 在云服务辅助项目中打开 **Blob** 帮助器

1. 打开 **Contoso.Events** 解决方案:

○ 文件位置:**Allfiles (F):\Mod08\Labfiles\Starter\Contoso.Events**

2. 在项目 **Contoso.Events.Worker** 中打开 **BlobStorageHelper.cs** 文件。

► 任务 2: 在创建的容器中上传 **Word** 文档

1. 从 **CreateBlob** 方法中删掉已存在的代码。

2. 为 **signin** 容器创建一个 **CloudBlobContainer** 变量并确保容器存在, 使用下面代码:

```
CloudBlobContainer container = _blobClient.GetContainerReference("signin");
container.CreateIfNotExists();
```

3. 为 blob 创建一个名字, 使用下面格式: “**{eventKey}_SignIn_{date}**”。

4. 使用新的 **blobName** 变量创建一个 blob 引用, 使用 **MemoryStream** 对象的 **Seek** 方法, 设置流的位置返回到原始, 然后使用 **UploadFromStream** 方法上载流到 blob, 代码显示如下:

```
ICloudBlob blob = container.GetBlockBlobReference(blobName);
stream.Seek(0, SeekOrigin.Begin);
blob.UploadFromStream(stream);
```

5. 返回结果的值。

结果: 完成此练习后, 您将能使用 Azure 存储 SDK 来管理存储帐户下的 Blob 和容器。

练习 3: 从容器中检索文件和媒体

场景

在此练习中, 您将:

- 使用 Azure 存储 SDK 从存储帐户下检索 Blob。

此练习的主要任务如下:

- 从 Blob 存储下载文档并流式传送到客户端
- 生成测试数据
- 从 Blob 存储下载生成的签到表

► 任务 1: 从 **Blob** 存储下载文档并流式传送到客户端

1. 打开 **Contoso.Events.ViewModels** 项目中的 **DownloadViewModel.cs** 文件。

2. 删掉 **GetStream** 方法中已存在的代码。

3. 使用 **_storageAccount** 变量来创建一个新的 **CloudBlobClient** 实例, 为 **signin** 容器创建一个 **CloudBlobContainer** 变量, 并确保容器存在如以下代码:

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
CloudBlobContainer container = blobClient.GetContainerReference("signin");
container.CreateIfNotExists();
```

4. 使用新的 **blobName** 变量来创建一个块 blob 引用，并使用 **OpenReadAsync** 方法来创建一个 Stream 变量如以下代码：

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
Stream blobStream = await blob.OpenReadAsync();
```

5. 通过把 Stream 变量赋值给 **DownloadPayload.Stream** 属性和把 **ICloudBlob** 变量的 **Properties.ContentType** 值赋值给 **DownloadPayload.ContentType** 属性，来返回一个新的 **DownloadPayload** 类实例的值。

► 任务 2：生成测试数据

1. 打开 **WindowsAzureStorageEmulator-v3.4**。
2. 调试 **Contoso.Events.Data.Generation** 项目以生成 SQL 并储存表格数据。

► 任务 3：从 Blob 存储下载生成的签到表

1. 调试解决方案。

 **注释：**配置解决方案用调试模式来启动 **Contoso.Events.Cloud** 项目和启动 **Contoso.Events.Management** 项目但不调试。

2. 等 **ContosoEvents(Contoso.Events.Cloud)** 网站打开后，使用 IISExpress 来打开 **Contoso.Events.Management** 网站。
3. 在 **Administration** 应用页面，选择任何一个事件，生成一个 sign-in 表格，并下载这个 sign-in 表格。

 **注释：**目前，这个网页应用得到一个 blob 流和使用 MVC 文件帮助把文件返回给使用者。在 **Contoso.Events.Management** 项目中，查看 **HomeController** 类里的 **DownloadSignIn** 方法。

结果：完成此练习后，您将使用 Azure 存储 SDK 从存储帐户下下载了 Blob。

练习 4：指定容器的权限

场景

在此练习中，您将：

- 使用 Visual Studio 2013 服务器资源管理器修改容器。
- 使用 Azure 存储 SDK 生成 SAS 令牌。

此练习的主要任务如下：

- 使用服务器资源管理器修改容器访问权限
- 使用 SDK 生成临时 SAS 令牌
- 使用 SAS 令牌从受保护的容器下载文档

► 任务 1：使用服务器资源管理器修改容器访问权限

1. 切换到 **Contoso.Events-VisualStudio2013** 窗口，然后停止调试。
2. 打开服务器资源管理器窗格中的 **Azure** 节点。

MCT ISE ONLY. STUDENT USE PROHIBITED

3. 在 Azure 里的 storage 帐户下的开发中查看 **signin** 容器:

- 帐户名称: 开发
- 容器名称: **signin**

4. 把 **signin** 容器的公共读访问设置成 **Off**。

► 任务 2: 使用 SDK 生成临时 SAS 令牌

1. 打开 **Contoso.Events.ViewModels** 项目里的 **DownloadViewModel.cs** 文件。

2. 删除 **GetSecureUrl** 方法内的已存在代码。

3. 使用 **_storageAccount** 变量来创建一个新的 **CloudBlobClient**, 为 **signin** 容器创建一个 **createCloudBlobContainer** 变量, 并确保容器存在, 如以下代码显示:

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
CloudBlobContainer container = blobClient.GetContainerReference("signin");
container.CreateIfNotExists();
```

4. 创建一个新的 **SharedAccessBlobPolicy** 类实例, 设置到期时间为当前时间后的 15 分钟, 并设置 blob 权限为可读, 如以下代码显示:

```
SharedAccessBlobPolicy blobPolicy = new SharedAccessBlobPolicy();
blobPolicy.SharedAccessExpiryTime = DateTime.Now.AddHours(0.25d);
blobPolicy.Permissions = SharedAccessBlobPermissions.Read;
```

5. 创建一个新的 **BlobContainerPermissions** 类实例, 并添加新的名称为 “**ReadBlobPolicy**” 的 **SharedAccessBlobPolicy**, 并禁用 public 权限, 如以下代码显示:

```
BlobContainerPermissions blobPermissions = new BlobContainerPermissions();
blobPermissions.SharedAccessPolicies.Add("ReadBlobPolicy", blobPolicy);
blobPermissions.PublicAccess = BlobContainerPublicAccessType.Off;
```

6. 使用 **CloudBlobContainer** 异步 **SetPermissionsAsync** 方法来应用 **BlobContainerPermissions** 变量, 并使用 **GetSharedAccessSignature** 方法来生成一个 SAS token, 如以下代码显示:

```
await container.SetPermissionsAsync(blobPermissions);
string sasToken = container.GetSharedAccessSignature(new SharedAccessBlobPolicy(),
"ReadBlobPolicy");
```

7. 使用 **_blobId** 变量创建一个 blob 引用, 并在一个 Uri 变量储存 Blob 的 Uri, 如下面代码显示:

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
Uri blobUri = blob.Uri;
```

8. 连接 Uri 变量的 **AbsoluteUri** 属性和 **sasToken**, 并返回结果的值。

► 任务 3: 使用 SAS 令牌从受保护的容器下载文档

1. 调试解决方案。

2. 在 **ContosoEvents**(**Contoso.Events.Cloud**)网页打开后, 使用 **IISExpress** 来打开 **Contoso.Events.Management** 网页。

3. 在 **Administration** 页面, 选择任一事件, 生成一个 sign-in 表格, 并下载这个 sign-in 表格。

 **注释:** 如果想要查看用于下载文件的完整 URL 和 SAS 令牌, 可以随时看看 Internet Explorer 的下载管理器。

4. 关闭 InternetExplorer 和 Contoso.Events–VisualStudio 窗口。

结果: 完成此练习后，您将修改了容器的权限，并生成了容器的 SAS 令牌。

问题: 服务器应用程序需要生成 SAS，以便客户端下载单个容器中的一部分 Blob。应该授予对容器的签名读取访问权，还是对各个 Blob 的读取访问权？

MCT USE ONLY. STUDENT USE PROHIBITED

实验 B：在中国版 Windows Azure 的存储 **Blobs** 中存储生成的文档

场景

你需要将你的程序 Contoso Events application 生成的 Word 文档保存在一个地方，你选择了 blob。为了让你的文档不能被匿名用户访问，你创建了一个受保护的容器。最后，你想添加逻辑创建一个 SAS 令牌来支持这个 Word 文档的访问。

目标

完成本次实验，您将会学到：

- 使用 Azure 管理门户创建 blob 容器。
- 使用 Azure 存储 SDK 给容器中添加 blobs。
- 通过使用 **MemoryStream** 对象从容器中下载 blob。
- 生成一个可以访问 blob 的 SAS 令牌。

预计时间：60 分钟

在开始这个实验之前，必须完成第 2 章的实验 B。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].chinacloudapp.cn:[虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到世纪互联运营的 Windows Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：使用 **Blobs** 实现 Azure 存储

场景

本次练习中，您会：

- 从管理门户上创建一个 blob 容器。
- 向容器中添加 blob 文件。
- 从容器上下载 blob 文件。

MCT USE ONLY. STUDENT USE PROHIBITED

此练习的主要任务如下：

- 登录到 Azure 管理门户
- 使用管理门户创建 Blob 容器
- 添加和访问容器中的 Blob 文件

► **任务 1：登录到 Azure 管理门户**

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 打开 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的屏幕语言不是中文，可以单击页面右上角的地球图表，并在菜单中选择中文(简体)语言。

► **任务 2：使用管理门户创建 Blob 容器**

1. 通过 Task1 中的练习，我们已经成功地登录到 Azure 的管理门户。
2. 在左侧的导航栏中，拖动滚动条找到存储图标，单击**存储**查看所有的存储实例列表。
3. 在存储实例列表中，定位到含有 **stor28532** 前缀的存储帐户。
4. 单击存储帐户的名称，进入含有 **stor28532** 前缀的存储帐户页面。
5. 在页面顶部，单击**容器**选项卡。
6. 在**容器**选项卡上查看所有的容器列表。
7. 在页面底部的工具栏中，单击**添加**按钮打开**新建容器**对话框。
8. 在**新建容器**窗口中，执行以下步骤：
 - a. 在**名称**文本框中，输入容器的名称 **example**。
 - b. 在**访问类型**列表中，选择**公共容器**。
 - c. 单击**对勾确定**按钮。

► **任务 3：添加和访问容器中的 Blob 文件**

1. 在 Start 屏幕，单击 **Visual Studio 2013** 磁贴。
2. 在**视图**菜单，单击**服务器资源管理器**。
3. 定位到 **Azure** 节点，单击左侧箭头展开节点。
4. 在 **Azure** 节点下面展开**存储**节点。

 **注释：**如果你之前没有导入 Azure 订阅文件，你需要先导入订阅文件。关于如何在 Visual Studio 中导入 Azure 订阅，请参考第二章实验 B，练习 4，任务 1 和任务 2。

5. 在**存储**节点展开 **stor28532[自定义名称]** 节点。
6. 展开 **Blob** 节点。
7. 双击 **example**。
8. 在 **example[容器]** 选项卡，单击**上载 Blob**。

 **注释：**上载图标是一个指向一条横线加向上箭头。

9. 在**上载新文件**对话框中，执行以下的步骤：

- a. 单击**浏览**按钮。
 - b. 定位到(F):\Mod08\LabFiles\Starter。
 - c. 单击**samplefile** 文本文档。
 - d. 单击**Open**。
 - e. 单击**确定**。
10. 切换至**Internet Explorer** 窗口。
11. 在新选项卡中输入下面的连接，将连接中的字符串**[存储帐户]**替换为你的存储帐户名称：
[http://\[存储帐户\].blob.core.chinacloudapi.cn/example/samplefile.txt](http://[存储帐户].blob.core.chinacloudapi.cn/example/samplefile.txt)
12. 验证在浏览器中显示的内容为：**This is your sample file!**。

结果：完成本次练习以后，你会在 Azure 的管理门户上创建的 blob 容器上，并查看 blob 文件。

练习 2：向容器中添加文件和媒体文件

场景

在本次练习中，您会：

- 使用 Azure 存储 SDK 来访问你 Azure 存储中的 blob 文件。
- 使用 Azure 存储 SDK 在 Azure 存储中创建 blob 文件。

此练习的主要任务如下：

- 在云服务解决方案的 Worker 项目中打开 BlobStorageHelper 文件
- 将新创建的 Word 文档上传到容器中

► 任务 1：在云服务解决方案的 Worker 项目中打开 BlobStorageHelper 文件

1. 在 Start 屏幕，单击**Desktop** 磁贴。
2. 在任务栏中，单击**File Explorer**。
3. 转到 Allfiles (F):\Mod08\Labfiles\Starter\Contoso.Events，然后双击**Contoso.Events.sln**。
4. 在解决方案资源管理器窗格，展开**Roles** 目录。
5. 在解决方案资源管理器窗格，展开**Contoso.Events.Worker** 项目。
6. 双击打开**BlobStorageHelper.cs** 文件。

► 任务 2：将新创建的 Word 文档上传到容器中

1. 在**BlobStorageHelper** 类中，找到如下签名的方法：

```
public Uri CreateBlob(MemoryStream stream, string eventKey)
```

2. 在**CreateBlob** 方法中，删除默认的一行代码：

```
return null;
```

3. 在**CreateBlob** 方法中，添加以下代码，代码的功能是创建一个名称为**signin** 的容器，并为该容器创建一个类型为**CloudBlobContainer** 的对象。

```
CloudBlobContainer container = _blobClient.GetContainerReference("signin");
```

4. 调用 **CloudBlobContainer** 的对象的 **CreateIfNotExists** 方法以保证容器确实存在：

```
container.CreateIfNotExists();
```

5. 在 **CreateBlob** 方法的结尾，右括号之前，创建一个名称为 **blobName** 的变量：

```
string blobName;
```

6. 使用 **String.Format** 静态方法创建一个字符串，并将该字符串赋值给 **blobName** 变量：

```
blobName = String.Format("{0}_SignIn_{1:ddmmyyss}.docx", eventKey, DateTime.UtcNow);
```

7. 在 **CreateBlob** 方法的结尾，右括号之前，调用容器的 **GetBlockBlobReference** 方法，以 **blobName** 变量作为参数来创建一个 **Blob** 块引用：

```
ICloudBlob blob = container.GetBlockBlobReference(blobName);
```

8. 调用 **MemoryStream** 的 **Seek** 方法将流偏移量设置为 **0**：

```
stream.Seek(0, SeekOrigin.Begin);
```

9. 使用 **ICloudBlob** 接口的 **UploadFromStream** 方法将流上传至引用的 **blob** 块：

```
blob.UploadFromStream(stream);
```

10. 在 **CreateBlob** 方法的结尾，右括号之前，添加下面的语句作为返回值：

```
return blob.Uri;
```

结果： 完成本次实验以后，你就可以使用 Azure 存储 SDK 来管理你的存储帐户中的 blob 和容器。

练习 3：从容器中检索文件和媒体文件

场景

本练习中，您会：

- 使用 Azure 存储 SDK 来获取存储帐户上的 blob。

此练习的主要任务如下：

- 从 Blob 存储下载文档并流式传送到客户端
- 生成测试数据
- 从 blob 存储下载生成的签到表

► 任务 1：从 Blob 存储下载文档并流式传送到客户端

1. 在解决方案资源管理器窗格上，展开 **Shared** 解决方案目录。
2. 在解决方案资源管理器窗格上，展开 **Contoso.Events.ViewModels** 项目。
3. 双击打开 **DownloadViewModel.cs** 文件。
4. 在 **DownloadViewModel** 类中，找到如下签名的方法：

```
public async Task<DownloadPayload> GetStream()
```

5. 在 **GetStream** 方法中，删除下面一行代码：

```
return await Task.FromResult<DownloadPayload>(null);
```

6. 在 **GetStream** 方法的结尾，在右括号之前，调用变量 `_storageAccount` 的 **CreateCloudBlobClient** 方法创建一个 **CloudBlobClient** 实例：

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
```

7. 通过调用 `blobClient` 的 **GetContainerReference** 方法，为 `signin` 容器创建一个 **CloudBlobContainer** 的对象：

```
CloudBlobContainer container = blobClient.GetContainerReference("signin");
```

8. 调用 `container` 的 **CreateIfNotExists** 方法，确保容器确实存在：

```
container.CreateIfNotExists();
```

9. 在 **GetStream** 方法的结尾，在右括号之前，调用 `container` 的 **GetBlockBlobReference** 方法，以变量 `_blobId` 作为参数来创建一个 `blob` 变量：

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
```

10. 调用 `blob` 对象的 **OpenReadAsync** 方法异步创建一个流对象 `blobStream`：

```
Stream blobStream = await blob.OpenReadAsync();
```

11. 在 **GetStream** 方法的结尾，在右括号之前，创建 **DownloadPayload** 类的一个实例：

```
DownloadPayload payload = new DownloadPayload();
```

12. 将 `blobStream` 变量赋值给 `payload` 变量的 **Stream** 属性：

```
payload.Stream = blobStream;
```

13. 将 `blob` 变量的 **Properties.ContentType** 属性值赋值给 `payload` 变量的 **ContentType** 属性：

```
payload.ContentType = blob.Properties.ContentType;
```

14. 返回 **DownloadPayload** 类型的变量 `payload`：

```
return payload;
```

► 任务 2：生成测试数据

1. 在 Start 屏幕，输入 **azure storage emulator**。
2. 单击 **Windows Azure Storage Emulator-v3.4** 磁贴。
3. 切换至 **Contoso.Events–Microsoft Visual Studio** 窗口。
4. 在解决方案资源管理器窗格，右键单击 **Contoso.Events.Data.Generation** 项目，指向调试，然后单击启动新实例。

► 任务 3：从 **blob** 存储下载生成的签到表

1. 在解决方案资源管理器窗格，右键单击 **Contoso.Events** 解决方案，然后单击属性。
2. 在解决方案 ‘**Contoso.Events**’ 属性页，执行下面的步骤：
 - a. 确保左边导航栏中通用属性=>启动项目被选中。
 - b. 选择多启动项目。

- c. 对于 **Contoso.Events.Cloud**, 设置操作为启动。
 - d. 对于 **Contoso.Events.Management**, 设置操作为开始执行（不调试）。
 - e. 确保剩余项目的操作都被设置成无。
 - f. 单击 **OK** 关闭对话框。
3. 在以下 3 个文件中, 查找 name 为 **EventsContextConnectionString** 的元素, 确认 connectionString 的值一致, 若不一致, 请使用 **Shared\Contoso.Events.Data.Generation** 项目中的 App.config 中的值替换另外两处的值, 连接字符串示例如下:
- o **Shared\Contoso.Events.Data.Generation** 项目中的 App.config;
 - o **Roles\Contoso.Events.Web** 项目中的 Web.config;
 - o **Administration\Contoso.Events.Management** 项目中的 Web.config;

连接字符串示例

```
Data Source=(localdb)\v11.0;Initial Catalog=EventsContextModule8Lab;Pooling=True;Integrated Security=True
```

4. 在调试菜单中, 单击启动调试。
5. 在 Windows 通知栏, 单击箭头查看正在运行的程序。
6. 定位到 **IIS Express**, 右键单击该图标。
7. 在 **View Sites** 中定位到 **Contoso.Events.Management**, 然后单击 **Browse Applications** 下方的 URL。
8. 在 **View Sites** 中定位到 **Contoso.Events.Cloud**, 然后单击 **Browse Applications** 下方的 URL。
9. 切换到 **Home – Contoso.Events Administration** 页面, 单击 **Go to Events List** 按钮。
10. 在列表中的任意事件项中, 单击 **Sign-In Sheet** 按钮。
11. 在 **Sign-in Sheet Generation** 页面, 提示 Sign-In Document Generation In Progress 的通知。
12. 等待 1 到 2 分钟, 单击 refresh 链接刷新 Sign-in 表格页面状态。
13. 单击 **Sign-In Sheet** 链接, 从服务器上下载 sign-in sheet 签到表。

结果: 完成本次练习以后, 你就可以通过 Azure 存储 SDK 来从你的存储帐户上下载 blob 了。

练习 4：指定容器的权限

场景

在本次练习中, 您会:

- 使用 Visual Studio 中的服务器资源管理器来更新 blob 容器。
- 为 Azure 存储 SDK 生成一个 SAS 令牌。

此练习的主要任务如下:

- 使用服务器资源管理器来修改容器访问权限
- 使用 SDK 生成临时 SAS 令牌
- 使用 SAS 令牌从受保护的容器下载文档

► 任务 1：使用服务器资源管理器来修改容器访问权限

1. 在桌面上，单击 **Contoso.Events – Visual Studio 2013** 窗口。
2. 在调试菜单，单击停止调试。
3. 在视图菜单，单击服务器资源管理器。
4. 定位到 **Azure** 节点，然后单击左侧节点上的箭头来展开 Azure 节点。
5. 展开 Azure 节点下的**存储**节点。
6. 展开**存储**节点下面的**开发**节点。
7. 展开**开发**节点下面的**blob** 节点。
8. 右键单击 **signin** 容器，然后单击属性。
9. 在属性窗格，定位到公共读访问。
10. 确保公共读访问的值为 **Off**。

► 任务 2：使用 SDK 生成临时 SAS 令牌

1. 在解决方案资源管理器窗格，展开 **Shared** 目录。
2. 在解决方案资源管理器窗格，展开 **Contoso.Events.ViewModels** 项目。
3. 双击打开 **DownloadViewModel.cs** 文件。
4. 在 **DownloadViewModel** 类，找到如下签名的方法：

```
public async Task<string> GetSecureUrl()
```

5. 在 **GetSecureUrl** 方法中，删掉这一行代码：

```
return await Task.FromResult<string>(String.Empty);
```

6. 在 **GetSecureUrl** 方法的结尾，在右括号之前，调用 **_storageAccount** 的 **CreateCloudBlobClient** 方法创建一个 **CloudBlobClient** 实例。

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
```

7. 调用 **CloudBlobClient** 的变量 **blobClient** 的 **GetContainerReference** 获取 **signin** 容器的一个对象。

```
CloudBlobContainer container = blobClient.GetContainerReference("signin");
```

8. 调用 **CloudBlobContainer** 的变量 **container** 的 **CreateIfNotExists** 方法，以确保 **blob** 容器确实存在：

```
container.CreateIfNotExists();
```

9. 在 **GetSecureUrl** 方法的结尾，在右括号之前创建一个 **SharedAccessBlobPolicy** 类的实例：

```
SharedAccessBlobPolicy blobPolicy = new SharedAccessBlobPolicy();
```

10. 设置 **SharedAccessBlobPolicy** 的变量 **blobPolicy** 的 **SharedAccessExpiryTime** 属性为现在的时间后推 15 分钟：

```
blobPolicy.SharedAccessExpiryTime = DateTime.Now.AddHours(0.25d);
```

11. 设置 **SharedAccessBlobPolicy** 的变量 **blobPolicy** 的 **Permissions** 属性的值为 **SharedAccessBlobPermissions.Read**：

```
blobPolicy.Permissions = SharedAccessBlobPermissions.Read;
```

12. 在 **GetSecureUrl** 方法的结尾，在右括号之前，创建 **BlobContainerPermissions** 类的一个实例：

```
BlobContainerPermissions blobPermissions = new BlobContainerPermissions();
```

13. 将 blobPolicy 对象添加到新创建的 blobPermissions 对象的 **SharedAccessPolicies** 属性列表中。

```
blobPermissions.SharedAccessPolicies
    .Add("ReadBlobPolicy", blobPolicy);
```

14. 设置 **BlobContainerPermissions** 的变量 blobPermissions 的 **PublicAccess** 属性为 **BlobContainerPublicAccessType.Off**：

```
blobPermissions.PublicAccess = BlobContainerPublicAccessType.Off;
```

15. 在 **GetSecureUrl** 方法的结尾，在右括号之前，异步调用 container 的 **SetPermissionsAsync** 方法，并使用 blobPermissions 作为参数：

```
await container.SetPermissionsAsync(blobPermissions);
```

16. 使用 **SharedAccessBlobPolicy** 类的新实例作为一个参数，字符串 “ReadBlobPolicy” 作为第二个参数，调用 **CloudBlobContainer** 的变量 container 的 **GetSharedAccessSignature** 方法：

```
string sasToken = container.GetSharedAccessSignature(new
SharedAccessBlobPolicy(), "ReadBlobPolicy");
```

17. 在 **GetSecureUrl** 方法的结尾，在右括号之前，调用 container 的 **GetBlockBlobReference** 方法，并将 _blobId 作为参数创建一个 blob 块引用：

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
```

18. 把 **ICloudBlob** 的 blob 变量的属性 **Uri** 保存在新的 **Uri** 实例 blobUrl 中。

```
Uri blobUrl = blob.Uri;
```

19. 在 **GetSecureUrl** 方法的结尾，在右括号之前，把 **Uri** 的 blobUrl 变量的属性 **AbsoluteUri** 的值和 sasToken 变量拼接成一个新的字符串，并赋值给新变量 secureUrl：

```
string secureUrl = blobUrl.AbsoluteUri + sasToken;
```

20. 将字符串变量 secureUrl 作为结果返回：

```
return secureUrl;
```

► 任务 3：使用 SAS 令牌从受保护的容器下载文档

1. 在调试菜单上，单击启动调试。
2. 在通知栏单击箭头查看所有运行的应用程序。
3. 定位到 **IIS Express**，右键单击该图标。
4. 单击在 **View Sites** 下方的 **Contoso.Events.Management**，然后单击在 **Browse Applications** 下面的 URL。
5. 在 **Events Administration** 页面，单击 **Go to Events List** 按钮以查看事件列表。
6. 单击事件列表中的任意 **Sign-In Sheet** 按钮。

7. 查看 Sign-in 页面并注意到一个 Sign-in 表格正在生成。
8. 等待 1 至 2 分钟，然后刷新 sign-in 表格页面。
9. 单击 **Sign-In Sheet** 按钮来下载 sign-in 表格。
10. 关闭 **Internet Explorer** 应用程序。
11. 关闭 **Contoso.Events - Visual Studio** 应用程序。
12. 关闭 **Azure Storage Emulator** 命令提示窗口。

结果：在完成本次练习以后，您会更新 blob 容器上的权限，并且生产访问容器的 SAS 令牌。

问题：服务器应用程序需要生成 SAS，以便客户端下载单个容器中的一部分 Blob。应该授予对容器的签名读取访问权，还是对各个 Blob 的读取访问权？

章节复习和作业

本章中，您使用存储 Blob 来存储云应用程序的文件。您还了解了如何使用 Azure 存储 SDK 来管理 Blob、容器及其权限。最后，您研究了用来为容器生成 SAS 令牌的机制。

复习问题

问题：有一个 Web 应用程序允许用户下载受保护的大 Blob。服务器端逻辑生成 SAS 令牌来检索受保护的 Blob。服务器应该下载 Blob 并流式传送给 Web 客户端，还是服务器应该向 Web 客户端提供附加 SAS 令牌的 Blob URL？

MCT USE ONLY. STUDENT USE PROHIBITED

第 9 章 使用队列和 Service Bus 设计通信策略

目录:

章节概述	9-2
第 1 课: Azure 存储队列	9-3
第 2 课: Azure Service Bus	9-6
第 3 课: Azure Service Bus 队列	9-8
第 4 课: Azure Service Bus 中继	9-12
第 5 课: Azure Service Bus 通知中心	9-15
实验 A: 使用队列和 Service Bus 来管理 Azure 中 Web 应用程序之间的通信	9-21
实验 B: 在中国版 Windows Azure 中使用队列和 Service Bus 来管理 Web 应用 程序间的通信	9-30
章节复习和作业	9-43

章节概述

Web 角色呈现内容，辅助角色处理逻辑，这当中需要一种机制来协调这些不同实体之间的通信。

Microsoft Azure 为此提供了两种排队机制。第 1 节“Azure 存储队列”介绍 Azure 存储帐户中提供的队列机制。第 2 节“Azure Service Bus”介绍 Azure 中的 Service Bus 产品。第 3 节“Azure Service Bus 队列”描述 Service Bus 中提供的排队机制，及其与 Azure 存储队列有什么不同。第 4 节“Azure Service Bus 中继”描述可用来将客户端设备连接到 WCF 服务的中继机制。第 5 节“Azure Service Bus 通知中心”介绍通知中心服务，以及有利于将通知推送到移动设备的基础结构。

目标

完成本章后，您可以做到：

- 描述存储队列服务。
- 描述 Service Bus。
- 描述 Service Bus 队列服务。
- 描述 Service Bus 中继。
- 描述通知中心服务。

第 1 课 Azure 存储队列

存储队列提供了一种一致且可靠的方法来存储可由多个工作进程使用的消息。

本节介绍存储中的队列服务，并描述其部分特征。

课程目标

完成本节后，您可以做到：

- 描述存储队列。
- 描述队列消息的特征。
- 查看存储队列中的消息。

存储队列概述

队列是一种存储大量消息的服务，可以使用 HTTP 或 HTTPS，通过已经过身份验证的调用，从世界上的任何地方访问这些消息。单条队列消息的大小最大可达 64 千字节 (KB)，一个队列可包含几百万条消息，最高可达存储帐户的总容量限制。存储帐户最多可包含 200 百万兆字节 (TB) 的 Blob、队列和表数据。

存储队列的常见用途包括：

- 创建要异步处理的未完成工作项 (backlog of work)
- 将消息从 Azure Web 角色传递到 Azure 辅助角色

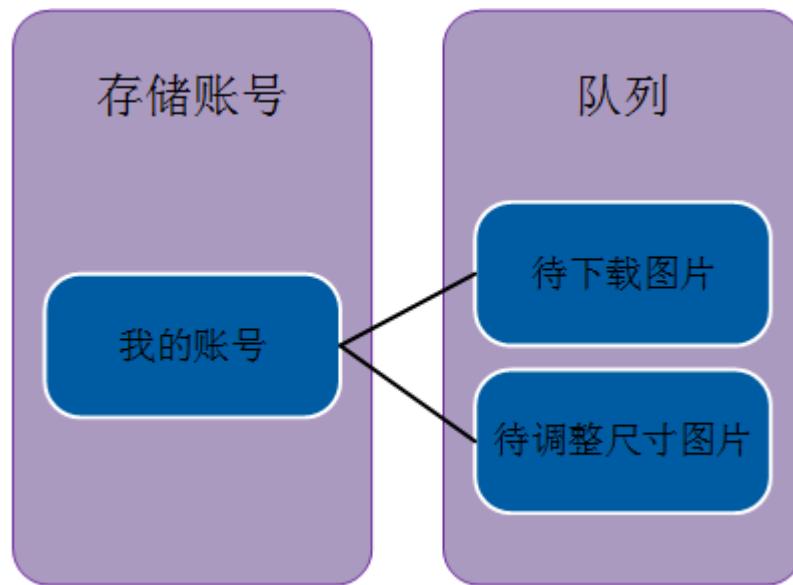
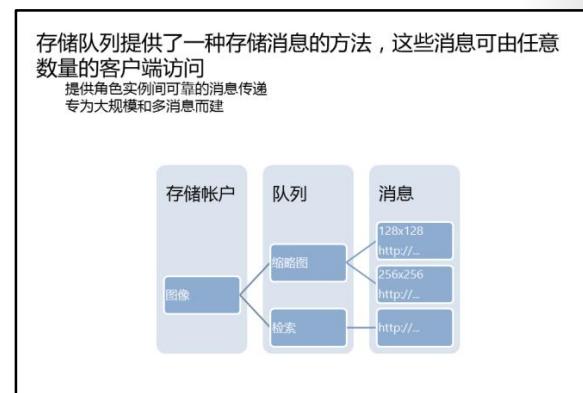


图 9.1：队列组件

URL 格式： 可以使用以下 URL 格式寻址队列：

`http://<存储帐户>.queue.core.windows.net/<队列>`

<http://<存储帐户>.queue.core.chinacloudapi.cn/<队列>> (中国版 Windows Azure)

下面的 URL 寻址图中的一个队列：

<http://myaccount.queue.core.windows.net/imagesToDownload>

存储帐户：对存储的所有访问都通过存储帐户来完成。

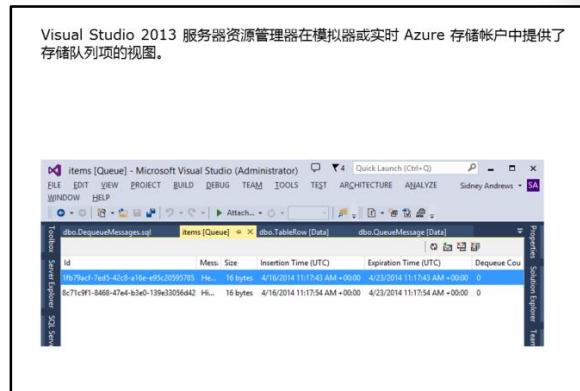
- 队列：队列包含一组消息。所有消息必须都在队列中。
- 消息：任何格式的消息，最大 64KB。

查看队列存储数据

从 Microsoft Visual Studio 2013 开始，服务器资源管理器（Server Explorer）窗格能让您直接在集成开发环境（IDE）中管理和查看 Azure 服务实例。可以使用服务器资源管理器查看队列中的消息。

存储队列消息

- 存储队列提供了以下基本消息功能：
 - 窥视下一条消息
 - 将下一条消息移出队列
 - 插入消息
 - 查看上次缓存的消息计数
- 消息内容作为字符串存储
 - 消息内容可更新以提供状态的概念
 - 可以设定其他使用者可看到消息内容更新的时间
- 消息的结果应该是幂等的
 - 从设计上说，存在重复处理队列消息的可能性



队列消息中的内容作为字符串存储。将复杂对象序列化为字符串就可将其存储在队列中。

常见队列消息操作

- 添加消息
 - 消息可添加到队列。
- 获取一条或多条消息
 - 从队列中检索后面一条或几条消息后，在指定的超时时间段（timeout period）内，其他客户端将看不到这些消息。
- 扫视（Peek）一条或多条消息
 - 从队列中检索后面一条或几条消息，同时让其他客户端仍能看到这些消息。
- 更新消息
 - 更新指定消息的内容或可见性超时时间（visibility timeout）。
- 删除消息
 - 处理消息后，可以从队列中删除这些消息，以免再次处理。

MCT USE ONLY. STUDENT USE PROHIBITED

超时和幂等*处理

消息的设计应该为，可以多次处理消息，而不会造成副作用。之所以发生多次处理的情况是因为，如果可见性超时时间已过，且已检索的消息尚未删除，那么其他队列客户端就可能看到这些消息。发生这种情况的常见场合包括辅助角色失效或瞬时服务错误。

*幂等：系统多次操作得到的结果是一样的。

更新队列消息

队列中存在的消息不是静态的。可以更改队列中任何消息的内容，或修改现有队列消息的可见性超时时间。例如，如果预见到应用程序有故障，可以保存正处理的消息的当前状态，然后延长可见性超时时间，或者由另一个工作进程实例处理该消息。在典型实现中，队列消息重试次数元数据存储在消息属性中。这有助于防止消息处理无限循环，造成应用程序故障。更新队列消息还可用作一种策略来创建队列消息多步工作流，在这种工作流中，这些消息由各种不同的工作进程实例来处理。

更新消息内容，并使其立即可见。

更新消息

```
CloudQueueMessage message = queue.GetMessage();
message.SetMessageContent("Updated contents.");
queue.UpdateMessage(message,
    TimeSpan.FromSeconds(0.0), // 使其立即可见。
    MessageUpdateFields.Content | MessageUpdateFields.Visibility);
```

第 2 课 Azure Service Bus

Service Bus 是 Azure 中完全托管的消息平台。应用程序的组件可利用 Service Bus 以断开的方式共享消息。

本节描述 Service Bus 服务及其功能。

课程目标

完成本节后，您可以做到：

- 描述 Service Bus。
- 描述 Service Bus 的功能。
- 创建 Service Bus 命名空间。

Service Bus 概述

不同的情况需要不同样式的通信。有时候，通过简单队列让应用程序收发消息是最佳的解决方案。其他情况下，当普通队列不足以胜任时，有发布和订阅机制的队列更佳。而某些情况下，真正需要的是应用程序间的连接 — 模块间通信并非总是需要队列。Service Bus 提供了所有三种选项，让应用程序以多种不同的方式交互。

Service Bus 是一种多租户云服务，这意味着该服务由多个用户共享。每个用户（如应用程序开发人员）创建一个命名空间，然后定义该命名空间中他或她需要的通信机制。

- Service Bus 是托管的消息传递基础结构
- 规模大，完全托管
- 可让您在确信消息平台将随着应用程序扩展而扩展的情况下，扩展应用程序并增加使用者
- 允许解耦组件，实现异步和同步通信

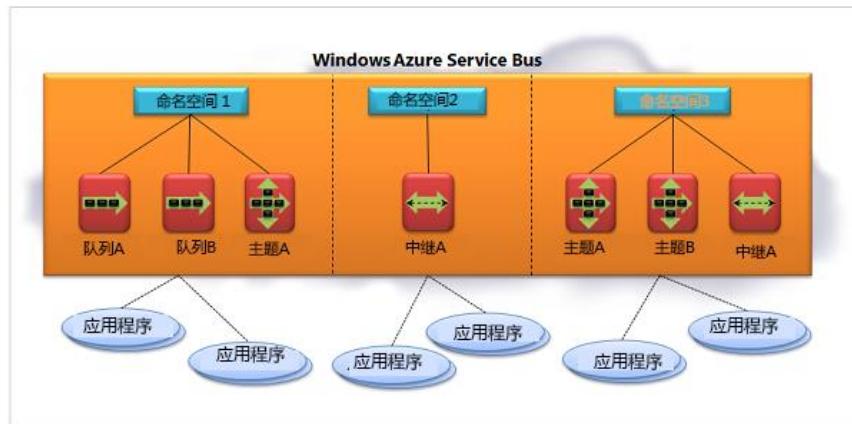


图 9.2: SERVICE BUS 体系结构

Service Bus 功能

Service Bus 由四个通信服务组成。可以创建每个服务的一个或多个实例来连接到应用程序。这些服务包括：

- **队列。**它们作为应用程序组件之间的中间层，并存储其他应用程序组件可接收的消息。
- **主题。**它们是一种单向通信机制，允许客户端应用程序或设备订阅主题。单独的应用程序或设备可将消息发送到主题，以供客户端应用程序使用。
- **中继。**它们是与用来与 Windows Communication Foundation (WCF) 服务通信的双向代理。客户端应用程序可直接绑定到中继端点，中继基础结构负责将消息路由到相应的 WCF 服务端点。
- **通知中心。**这是一种托管的中转系统，通过该系统，可使用本地通知将消息从服务器应用程序分发到各种平台的客户端设备。



命名空间

命名空间作为 Service Bus 服务实例的基本逻辑分组。

在创建队列、主题或中继时，需要赋予其一个名称。该实例名称随后与命名空间组合在一起，形成对象的唯一标识符。应用程序将此名称提供给 Service Bus，然后使用该队列、主题或中继来相互通信。

Service Bus 命名空间还可包含管理凭据（或称共享密钥），客户端应用程序可使用该凭据来连接到 Service Bus。

Service Bus 命名空间是 Service Bus 服务实例的逻辑分组

- 它将资源集中起来，以提供通用且可预测的地址
- 它提供管理凭据以用于操作

第 3 课 Azure Service Bus 队列

Service Bus 提供了队列功能，可用来将消息从提请消息的应用程序封送到使用消息的应用程序。Service Bus 队列不同于存储队列。

本节描述 Service Bus 队列以及 Service Bus 队列与存储队列间的区别。

课程目标

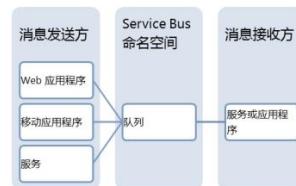
完成本节后，您可以做到：

- 描述 Service Bus 队列。
- 描述 Service Bus 队列与存储队列之间的区别。

Service Bus 队列概述

Service Bus 队列是一种中转消息系统，与队列服务 Azure 存储相似。使用这些队列后，分散的应用程序模块就无需直接相互通信。这些应用程序模块可改为使用队列来通信。这确保了消息生成者与消息处理者之间的分离。这样的分离带来了某种灵活性，可以让一个或多个应用程序组件实例生成消息，而让另一个或更多应用程序组件实例处理同样的消息。如果实例遇到不可恢复的异常情况，其他实例可以继续处理消息。如果整个应用程序的工作负载增加，可以创建新实例来处理负载。在开发和设计云应用程序时，这些情况很常见，也很关键。

Service Bus 队列提供了中转消息传递通信模型
分布式应用程序可以先入先出 (FIFO) 模式共享消息
单独的消息只能由一个消息使用者接收



Service Bus 队列实现了常用的先入先出 (FIFO) 消息传递策略。Service Bus 队列还可保证消息使用者收到消息，并且不多不少只处理一次。

Service Bus 队列提供了单向异步排队。

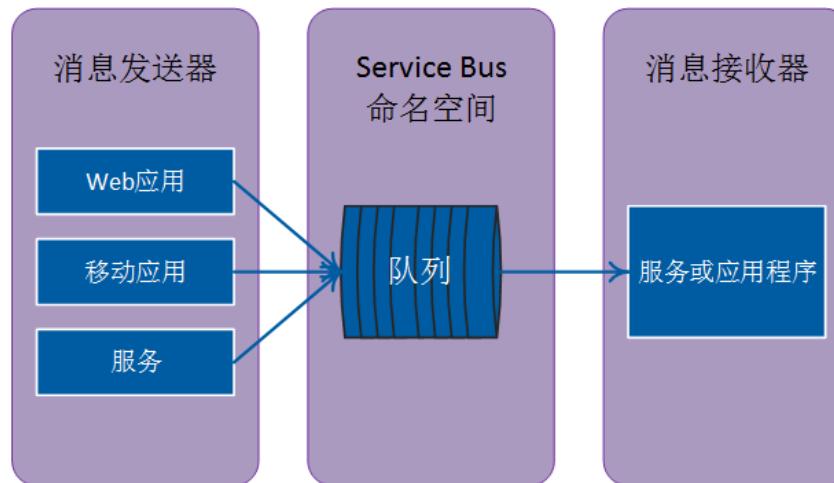


图 9.3: SERVICE BUS 队列

Service Bus 队列是一种通用技术，可用于广泛的各种场合：

- 多层 Azure 应用程序中，Web 角色与辅助角色之间的通信
- 混合解决方案中，企业内部应用程序与 Azure 托管的应用程序之间的通信

- 分布式企业内部应用程序组件之间的通信（这些应用程序运行在不同组织中，或者同一组织内不同的部门中）

使用队列，可以更好地向外扩展应用程序，并为体系结构提供更多复原力。

队列消息传送

Service Bus 队列中的消息是以通用、可预测的模式传递并处理的。首先，消息生成者创建新消息，并将消息添加到队列。接收方可在此后处理此消息。在多播场景中，很多接收方可处理同一队列中的消息，但是它们不共享这些消息。消息通常只由一个接收方处理，除非出现异常情况。

在接收方读取消息时，它有两种选择来处理消息。接收方可先选择 **ReceiveAndDelete** 方法。此方法将消息从队列中移出，读取后将其删除。使用此方法的风险是，如果接收方应用程序出现错误，则消息可能丢失。另一种方法是，接收方可选择使用 **PeekLock** 方法。此方法仍会从队列中移出消息，但是不会删除消息。消息将加锁，并标记为其他接收方实例不可见。可以用三种方式处理该锁：

- 如果消息处理成功，使用 **Complete** 方法来指示队列立即删除消息。
- 如果消息在处理过程中的某个点上失败，但是应用程序可处理该故障而不会崩溃，那么使用 **Abandon** 方法来指示应从消息上移除该锁。需要处理该消息的其他接收方现在将可以看到此消息。
- 如果为队列消息配置的生存时间值过期，队列将假设接收方处于故障状态。队列将从消息上移除锁，使得其他接收方可处理该消息。

- Service Bus 队列提供了一种排队机制，可以严密控制消息的顺序和传送
 - 消息将只出现一次
 - 消息使用 FIFO 模式进行处理
 - 消息锁可延期
 - 支持事务

在接收方应用程序组件读取消息后，接收方将负责在消息得到成功处理后调用 **Complete** 方法。如果未出现这种情况，那么可能发生了无限循环，同一条消息因为生存期过期而被无限次处理。Service Bus 队列消息有唯一 ID，客户端应用程序可使用此 ID 来确定消息是否由多个接收方处理。

Service Bus 队列消息的特征

如果使用 Azure 的 .NET 库，那就必须使用

BrokeredMessage 类来表示 Service Bus 队列消息。

。 **BrokeredMessage** 类包含标准属性，这些属性表示与各条消息相关的元数据。这些属性包括：

- Label**。此属性可由应用程序用来提供消息的细节和简单标签。
- TimeToLive**。可以使用此属性来指出消息应持久保存在消息存储中的持续时间。
- MessageId**。可以使用此属性提供消息的唯一标识符。

Service Bus 队列消息由几个主要部分组成

- 主体
 - 主体可以是任何可序列化的对象或流
 - DataContractSerializer 用于序列化复杂对象
- 标签
 - 简单文本标签
 - TimeToLive
- 属性
 - 可由特定使用者使用的属性字典。

消息对象还包含名为 **Properties** 的字典属性。这

个 **IDictionary<string, object>** 类型化属性可包含要为应用程序定义的任何自定义属性。除了消息的实际主体外，这些属性也包含在对象中。消息主体是使用 **DataContractSerializer** 序列化的公共语言运行时 (CLR) 对象。在创建新的 **BrokeredMessage** 实例后，可以使用 **BrokeredMessage** 类的 **GetBody<T>** 方

法添加消息主体。由于主体是序列化的，因此这可以是要从发送方应用程序传送到接收方应用程序的任何复杂对象。有关此消息的其他元数据通常存储在 **BrokeredMessage** 类的 **Properties** 属性中。

此示例演示如何将五条测试消息发送到 **QueueClient** 实例。

新建 BrokeredMessage

```
BrokeredMessage message = new BrokeredMessage("Test message");

message.Properties["TestProperty"] = "TestValue";
message.Properties["Message number"] = 12;

Client.Send(message);
```

每条消息有两个部分，一组表示键/值对的属性，以及一个二进制消息主体。如何使用这些部分取决于应用程序试图执行的操作。例如，发送最新销售额消息的应用程序可能包含属性 *Seller="Ava"* 和 *Amount=10000*。消息主体可能包含销售签署合同的扫描图像，或者，如果没有图像，消息主体可能为空。

可以使用 **BrokeredMessage** 类中的类似属性来检索消息。

此示例演示如何使用默认的 PeekLock 模式接收和处理消息。

检索消息

```
while (true)
{
    BrokeredMessage message = Client.Receive();

    if (message != null)
    {
        try
        {
            Console.WriteLine("Body: " + message.GetBody<string>());
            Console.WriteLine("Test Property: " + message.Properties["TestProperty"]);

            message.Complete();
        }
        catch (Exception)
        {
            message.Abandon();
        }
    }
}
```

Service Bus 队列与存储队列

Azure 支持两种类型的队列机制：存储队列和 Service Bus 队列。

- 存储队列。**这些队列是 Azure 存储基础结构的组成部分。它们有基于 REST 的 Get/Put/Peek 简单接口。它们在服务内及服务之间提供可靠、持久的消息传递。
- Service Bus 队列。**它们是更广泛的 Azure 消息传递基础结构的组成部分，该基础结构支持排队和发布/订阅、Web 服务远程处理，以及集成模式。

虽然两种排队技术同时存在，但是存储队列是较先引入的，它作为一种建立在 Azure 存储服务基础上的专用队列存储机制。Service Bus 队列建立在更广泛的

存储队列	Service Bus 队列
任意顺序 至少传达一次，可能多次 30 秒默认锁可延长到 7 天 支持就地更新消息内容 可通过自定义活动与 WF 集成	保证先入先出排序 只传送一次 60 秒默认锁可延期 消息一旦使用即告结束 与 WCF 和 WF 天然集成

中转消息传递基础结构的基础之上，这种基础结构旨在集成应用程序或应用程序组件，这可能横跨多种通信协议、数据合约、信任域和网络环境。

存储队列与 **Service Bus** 队列之间的区别

比较标准	存储队列	Service Bus 队列
排序保证	否	是，先入先出 (FIFO)
传递保证	至少一次	至少一次 至多一次
事务支持	否	是
接收模式	扫视并租借 (Peek and Lease)	扫视并加锁 接收并删除
独占访问模式	基于租期	基于锁
租期/锁粒度	消息级别	队列级别
批量接收	是	是
批量发送	否	是

- 排序保证。** Service Bus 队列保证按 FIFO 顺序处理消息。存储队列不保证此顺序。
- 传递保证。** Service Bus 和存储队列保证至少传递消息一次。Service Bus 队列还可保证消息只传递一次。
- 租期粒度。** – 存储队列可将每条消息的锁/租期长度设置为不同值。Service Bus 队列只能为队列实例中的所有消息设置此值。
- 批处理。** 存储队列和 Service Bus 队列都可批量接收消息。只有 Service Bus 队列支持批量创建消息。

第 4 课 Azure Service Bus 中继

Service Bus 中继提供了一种灵活的方式来连接 WCF 服务和客户端，而无需重新设计组织中的网络体系结构。在将 WCF 服务外向连接到中继时，客户端只需要连接到 Azure 中的 Service Bus 端点，即可与 WCF 服务通信。

本节描述 Service Bus 中继的优势和体系结构。

课程目标

完成本节后，您可以做到：

- 描述使用 Service Bus 中继的优势。
- 详细说明 Service Bus 中继的体系结构。

Service Bus 中继概述

Service Bus 包含一个中继组件，该组件可将现有服务连接到新的客户端应用程序，而无需暴露服务的真实位置或地址。支持的服务包括简单对象访问协议 (SOAP)、Web 服务标准 (WS*) 以及具象状态传输 (REST)。使用中继的部分优点包括：

- 需要直接与外部客户端应用程序或设备（例如，移动设备）通信的 WCF 服务通常需要放在一个有着独特 NAT 或防火墙配置的特殊子网或虚拟网络中。WCF 端点的地址需要可公开寻址，以便客户端设备连接。在某些企业中，这样的做法可能被视为危险或不可接受。如果使用 Service Bus 中继，WCF 服务建立与中继的出站连接，同时绕过了入站连接所必需的很多复杂网络配置。
- 虽然移动应用程序部署后经常更新，但是最终用户可能不会像您希望的那样经常更新应用程序。如果 WCF 服务需要迁移到新网络或移至新的 IP 地址，这可能导致移动应用程序的连接中断。假如使用 Service Bus 中继，移动应用程序将寻址可供公共访问的持久统一资源标识符 (URI)。然后，您就可以在组织的基础结构中随意更改和迁移 WCF 服务。新的服务实例或位置只需要连接到中继，就可以让客户端设备访问到它们。这给连接到已部署应用程序的服务带来了更高移动性。

- 中继提供了一种机制，将分布式客户端应用程序或云服务连接到对应的企业内部端点
- 它允许单向或双向通信
- 它将消息直接中继到端点，而无需任何消息中转
- 应用程序建立指向中继的出站连接，中继管理消息的传输

Service Bus 中继还支持直接对等通信。如果中继确定发起连接的应用程序可以轻松直接地寻址 WCF 服务，那么就可以成功进行直接对等通信。

Service Bus 中继服务能让您建立在 Azure 数据中心以及您自己的企业内部环境中运行的混合应用程序。

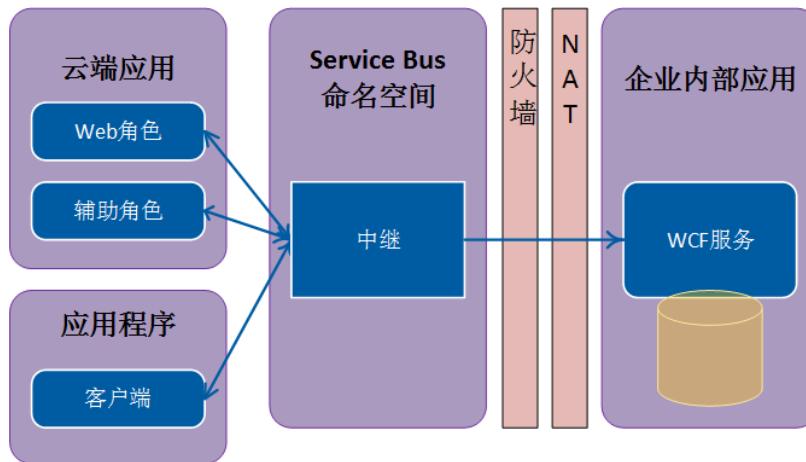


图 9.4: SERVICE BUS 中继

使用 Service Bus 中继时，必须使用一组与 WCF 通常自带的绑定相似的 WCF 绑定。这些绑定包含中继前缀。它们实现了一些新的绑定元素，这些元素创建通向 Azure 中的 Service Bus 实例的通道。单向消息、请求/响应消息和分布式事件消息包含绑定。事件消息是 Service Bus 中继特有的，它们用于实现发布/订阅场景，在这种场景中，客户端应用程序可以发送消息，并使该消息分发到多个 WCF 服务实例。例如，事件消息传递可用来将消息分发到接收方 WCF 服务进行处理，并审核 WCF 服务。

由于 Service Bus 中继所利用的绑定与传统 WCF 绑定非常相似，因此很多应用程序不需要很大改动就能使用 Service Bus 中继。客户端应用程序通常需要指定中继绑定的新绑定配置。在客户端与中继通信后，中继服务会负责将消息定向到相应的 WCF 服务。客户端应用程序无需知道服务的地址或实际位置。服务应用程序无需在其防火墙上开放入站端口来接收这些消息。

Service Bus 中继的优点

- 灵活的 NAT 配置

由于中继建立出站链接，因此可以在复杂网络环境中使用中继，在这类环境中，通常需要更改防火墙来接受入站连接。

- WCF

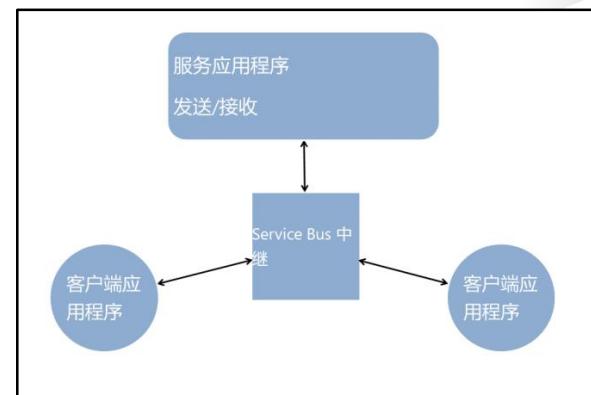
中继需要配合 WCF 服务使用。WCF 是一种成熟稳定的消息传递框架，它既支持 SOAP 消息，也支持 REST 消息。WCF 还有一套由自定义绑定、行为和组件组成的成熟生态系统。SOAP 和 REST 也已得到很多现有服务或客户端设备的广泛支持。

Service Bus 中继体系结构

Service Bus 中继允许在现有企业环境中托管 WCF 服务。托管后，可以将侦听任务委派给传入会话和将请求任务委派给在 Azure 中运行的 Service Bus 服务。这些消息然后路由到相应的 WCF 服务。这样就可以将这些服务公开给在 Azure 中运行的应用程序代码，或者公开给移动工作者或外部网合作伙伴环境。Service Bus 可让您安全地精细控制谁可以访问这些服务。这提供了一种强大安全的方式来公开现有企业解决方案中的应用程序功能和数据，并从云端利用这些功能和数据。

传入中继请求

当客户端向 Service Bus 发出请求时，Azure 负载平



衡器将其路由到任一个网关节点。如果请求是侦听请求，网关节点将创建一个新的中继。如果请求是要连接特定中继的连接请求，网关节点将该连接请求转发到拥有该中继的网关节点。拥有该中继的网关节点向侦听客户端发出会合请求，要求该侦听方创建一个临时通道，通向接收该连接请求的网关节点。

管理凭据

应用程序可使用共享访问签名 (SAS) 身份验证向 Service Bus 证明身份。以前，Azure Active Directory 访问控制服务 (ACS) 用于提供访问密钥。

SAS 身份验证能让您允许用户以特定权限粒度化访问 Service Bus 资源。Service Bus 中的 SAS 身份验证涉及加密密钥的配置，通过该密钥可获得对 Service Bus 资源的相关权限。然后客户端可出示 SAS 令牌来访问该资源，该令牌包含要访问的资源 URI 以及用配置的密钥签署的过期时间。

可以配置 Service Bus 命名空间的 SAS 密钥。该密钥将适用于此命名空间中的所有消息实体。还可以配置 Service Bus 队列、主题和通知中心的密钥。不远的未来将添加对 Service Bus 中继的支持。

为了使用 SAS，可以在命名空间、队列、主题或通知中心上配置一个 **SharedAccessAuthorizationRule** 对象，该对象由以下部分组成：

- KeyName，标识规则
- PrimaryKey，用于签署或验证 SAS 令牌的加密密钥
- SecondaryKey，用于签署或验证 SAS 令牌的加密密钥
- Rights，表示授予的侦听、发送或管理权限的集合

在命名空间级别配置的授权规则可允许客户端访问该命名空间中的所有实体，只要该客户端持有使用对应密钥签名的令牌。可以在 Service Bus 命名空间、队列、主题或通知中心上配置最多 12 个这样的授权规则。默认情况下，在初次设置命名空间时，每个命名空间都会配置拥有所有权限 **SharedAccessAuthorizationRule** 对象。

- Service Bus 使用共享访问签名 (SAS) 来验证对命名空间中消息实体的访问
 - 这替代了以前可用的 ACS 功能
 - 也可以使用来自提供程序的简单 Web 令牌 (SWT) 或 SAML 令牌

MCT USE ONLY. STUDENT USE PROHIBITED

第 5 课 Azure Service Bus 通知中心

通知中心提供了一个用于高度可伸缩托管移动推送通知平台的简单接口。使用通知中心，应用程序可以跨各种移动平台发送模板和个性化通知。

本节描述通知中心服务以及与该服务集成的方法。

课程目标

完成本节后，您可以做到：

- 描述 Azure 中通知中心的体系结构。
- 从客户端或服务应用程序向通知中心注册客户端设备。
- 使用模板泛化跨平台的通知。
- 使用标记将通知定向到特定的客户端设备。

Service Bus 通知中心概述

Service Bus 通知中心是服务基础结构和一组客户端库的组合，它们可以把来自应用程序服务组件的推送通知发布到移动应用程序。使用通知中心，可以发送专为特定用户个性化的通知、跨各种平台分发给很多用户的通知，以及经过筛选，只发给一组特定用户的通知。通知中心基础结构抽象了每种移动平台的各种平台通知系统 (PNS) 的实现。只使用一个方法调用，就可将通知发送到各种设备平台，而无需实现每种平台不同的消息结构或通信机制。

可以在各种场合下使用通知中心，包括：

- 将需要广而告之的新闻通知发送到安装了移动应用程序的所有设备
- 将通知发送到根据标记、标签或位置确定的一小部分用户
- 将特定通知发送给某个用户，使其知道与其特定帐户相关的活动

- 将推送通知发送到移动设备的托管基础结构
 - 多平台
 - 可伸缩
 - 简单 SDK
 - 可在很多主流移动平台上获得
- 广播到很多用户或定向到特定用户

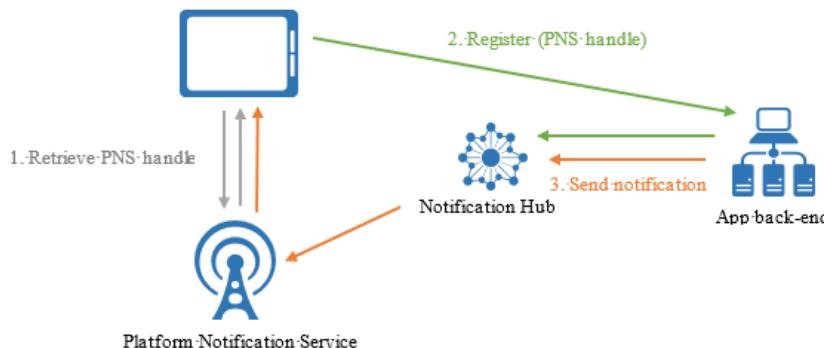


图 9.5：通知中心

使用通知中心的优点

通知中心避免了管理推送通知所涉及的难题。通知中心使用完整的多平台扩展推送通知基础结构，大大减少了应用程序中运行的推送专用代码。通知中心实现了推送基础结构的全部功能。设备只负责注册 PNS 句柄，后端负责向用户或相关群体发送平台无关的消息。

通知中心提供的推送基础结构具有以下优势：

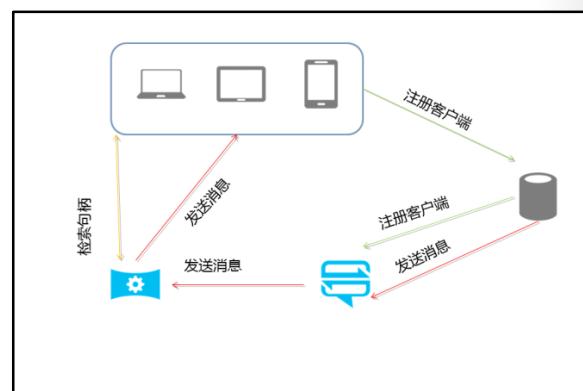
- **多平台：**
 - 支持所有主流移动平台 - Windows、Windows Phone、iOS 和 Android。
 - 平台无关的协议。应用程序只与通知中心通信。
 - 设备负责管理。通知中心维护句柄注册表以及来自 PNS 的反馈。
- **适用于任何后端。**适用于用 .NET、PHP、Java 或 Node 编写的云应用程序或企业内部应用程序。
- **扩展。**通知中心可扩展到数百万台设备，而无需重建体系结构或分片。
- **交付模式丰富。**将设备与标记关联，表示逻辑用户或相关群体。
 - 广播。只要一次应用程序编程接口 (API) 调用，就可几乎同时地广播到数百万台设备。
 - 单播/多播。推送到表示单个用户（包括其所有设备）或更广泛群体的标记。例如，用户可能在不同的设备（平板电脑、手机等）上使用应用程序，并要求推送通知推送到所有设备或某台特定设备。
 - 分段。推送到由标记表达式定义的复杂分段（例如，洋基队的纽约支持者的设备）。
- **个性化。**每种设备可有一个或多个模板，从而实现逐台设备本地化和个性化，而又不影响后端代码。

- 托管基础结构
 - 无需操心自己扩展应用程序
 - 可以关注消息和模板，而不是服务的机理。
- SDK 可用于主流平台
- 模板支持
- 支持按标记筛选接收者

通知中心体系结构

在较高的层面，所有平台通知系统都遵循同一模式：

1. 客户端应用程序联系 PNS 来检索其句柄。句柄类型依系统而定。对于 Windows 通知服务 (WNS)，句柄是 URI 或通知通道。对于 Apple 推送通知服务 (APNS)，句柄是令牌。
2. 客户端应用程序将此句柄存储在应用程序后端，以备以后使用。对于 WNS，后端通常是云服务。对于 APNS，这种系统称为提供程序。
3. 为了发送推送通知，应用程序后端使用句柄来联系 PNS，以便定位特定客户端应用程序的实例。
4. PNS 将通知转发给该句柄指定的设备。



MCT USE ONLY. STUDENT USE PROHIBITED

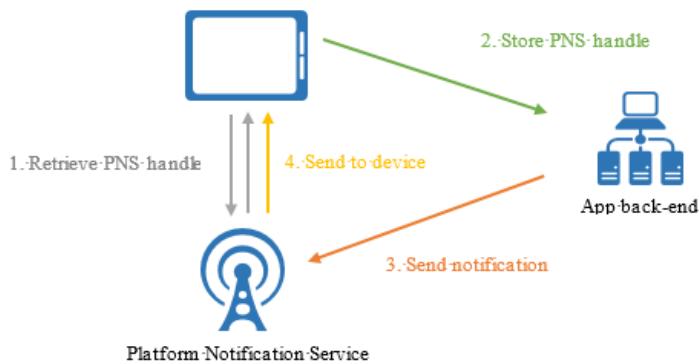


图 9.6: PNS

利用通知中心，您可以依赖服务基础结构来处理最复杂的功能，而让应用程序只集中于发送消息。

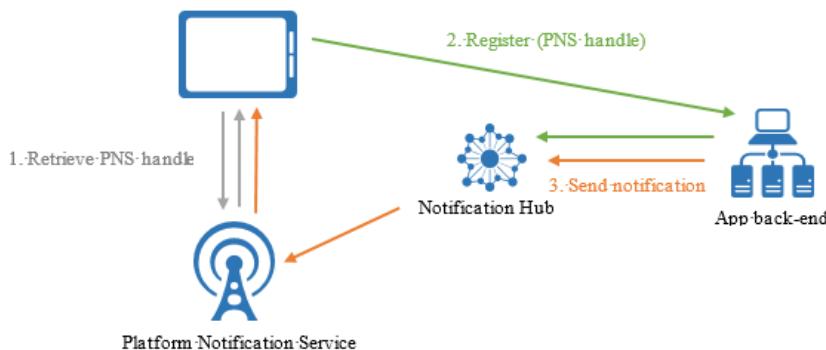


图 9.7: 通知中心

可以灵活使用通知中心来注册设备，并最终将消息发送到设备。设备可使用以下方法注册自己并接收通知：

1. 客户端设备使用通知中心 SDK 联系 PNS。它注册一个唯一 PNS 句柄。无论应用程序是否正在运行，服务使用此句柄将通知发送到此设备。
2. 或者，客户端设备也可以将其 PNS 句柄发送到应用程序后端，让应用程序注册该设备。
3. 在应用程序后端将消息发送到通知中心服务时，服务使用其注册的 PNS 句柄来负责将消息发送到相应目标客户端。应用程序后端只要请求发送消息，通知中心服务和 PNS 将负责将消息实际分发到客户端设备。

注册点

注册点(registration)是通知中心的子实体，并将设备 PNS 句柄（例如 ChannelURI、设备令牌或 Google Cloud Messaging (GCM) registrationId）与标记，也可能是模板关联。标记用于将通知路由到正确的设备句柄集。模板用于实现按注册点的转换。

注册点默认是临时的。它们最大可设置为 90 天。这可能影响应用程序的设计方式以及为通知中心选择的注册方法。注册点必须包含每个设备或通道最新的 PNS 句柄。由于 PNS 句柄只能在设备上的客户端应用程序中获得，因此管理注册点的一种方法是直接向客户端应用程序上的通知中心提供程序注册。另一方面，与标记相关的安全注意事项和业务逻辑可能要求您在应用程序后端管理注册点。

应用程序后端注册点

从后端管理注册点需要编写额外的代码。每次应用程序启动时，设备中的应用程序必须将更新的 PNS 句柄提供给后端（连同标记和模板），后端必须在 Service Bus 上更新此句柄。从后端管理注册点的优势是，能够将标记修改为注册点（即使在设备上的对应应用程序未处于活动状态的时候），能够先验证客户端应用程序的身份，然后再将标记添加到其注册点。从应用程序后端，可以对注册点执行基本的创建、读取、更新和删除 (CRUD) 操作。

- 为获得远程通知，每个 SDK 提供了独特的注册机制
- 必须使用通知中心的名称，以及连接信息面板中的唯一连接字符串，来向通知中心注册
- 默认有两个连接字符串：
 - DefaultFullSharedAccessSignature
 - DefaultListenSharedAccessSignature
- 可选择将 DefaultListenSharedAccessSignature 用作应用程序的受限仅侦听连接字符串

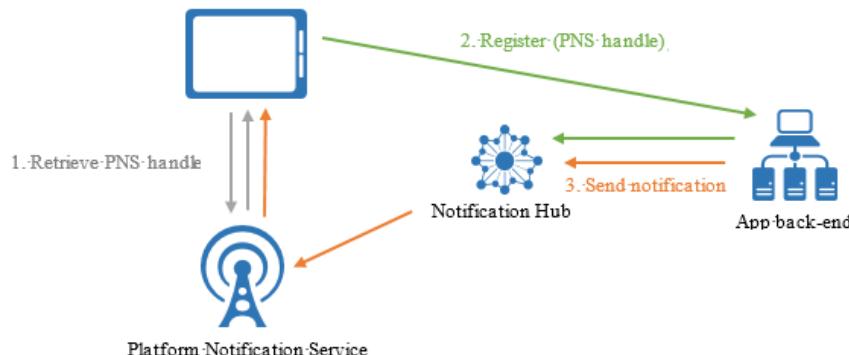


图 9.8：应用程序注册

客户端注册

在从客户端应用程序管理注册点时，后端只负责发送通知。客户端应用程序使 PNS 句柄保持最新并注册到标记。移动设备先从 PNS 检索 PNS 句柄，然后直接向通知中心注册。注册成功后，应用程序后端可发送以该注册点为目标的通知。

客户端注册的缺点是，客户端应用程序只能在应用程序处于活动状态时更新其标记。例如，如果用户有两台设备都注册了与运动队相关的标记，当第一台设备再注册一个运动队时，第二台设备要在此设备上的应用程序再次运行后，才会收到有关该队的通知。一般来说，当标记受到多台设备的影响时，从后端管理标记是理想的选择。

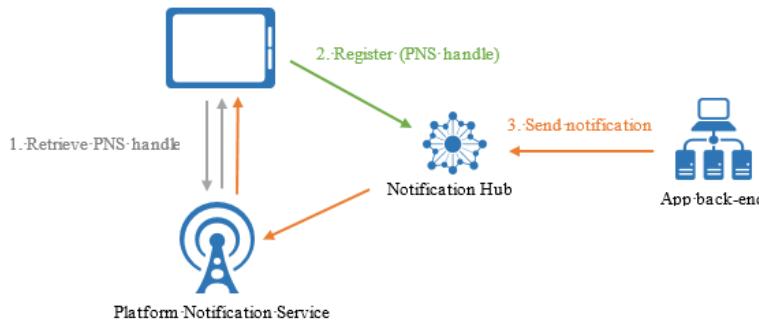


图 9.9：客户端注册

消息模板和标记

发送到通知中心的所有消息并非都要分发到所有设备平台或类型以及所有设备。可以使用模板将消息转换为兼容每种平台和移动操作系统的相应结构和格式。可以使用标记来筛选消息的目标设备。

模板

模板使客户端应用程序可以指定自己要接收的通知的确切格式。例如，以下两种有效负载用于 Windows 和 Apple 移动设备：

APNS（Apple 平台通知服务）的通知格式

APNS JSON 有效负载

```
{"aps": {"alert" : "Hello!"}}
```

WNS（Windows 通知服务）的通知格式

WNS XML 有效负载

```
<toast>
  <visual>
    <binding template=\\"ToastText01\\">
      <text id=\\"1\\>Hello!</text>
    </binding>
  </visual>
</toast>
```

- 模板可让您从后端发出一条消息，然后针对每种平台将其转换为结构正确的消息
- 可以在模板使用的绑定格式中指定，消息将出现在 XML 或 JSON 内容中的什么地方
 - 可在模板中使用自定义属性
- 客户端可创建多个注册点来利用不同的模板

这个要求迫使应用程序后端针对每种平台生成不同的有效负载。在考虑图形布局和本地化时，这会成为问题。通知中心模板功能使得客户端应用程序能创建称为模板注册点的特殊注册点，这种注册点除了包含一组标记外，还包含一个模板。模板用于将消息转换为适合每种设备的相应格式或结构。

模板是一组指令，它们指示通知中心将平台无关的消息格式为适合每种设备的格式。

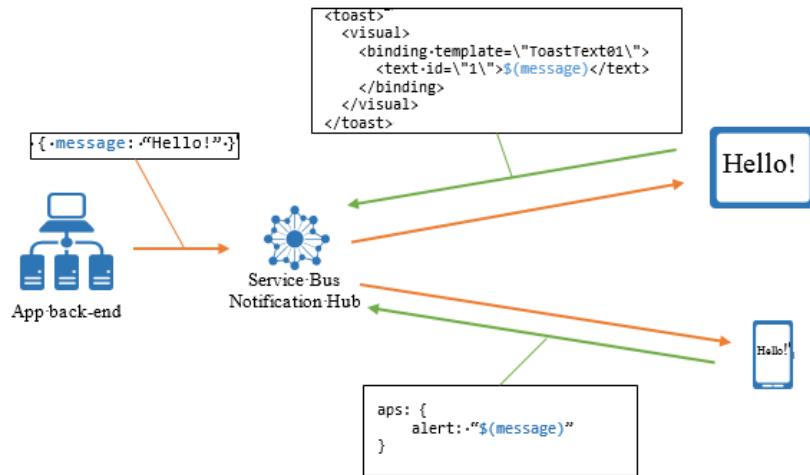


图 9.10：模板

标记

在通过通知中心发送推送通知时，标记表达式用来定向发送到一组特定设备，或者更具体地说，一组注册点。标记可以是包含字符数字字符的任何字符串，最多 120 个字符。定向发送到特定注册点的唯一方法是将其与标记关联，然后定向发送到该标记。

应用程序后端可用以下方式选择特定通知要定向发送到的注册点：

- **广播**。通知中心中的所有注册点接收通知
- **标记**。包含指定标记的所有注册点接收通知
- **标记表达式**。其标记集符合指定表达式的所有注册点接收通知

标记无需预先设置，可以参考特定于多应用程序的概念。有些情况下，通知必须定向发送到一组不是由单个标记，而是由标记布尔表达式标识的注册点。标记表达式可包含所有布尔运算符，如 AND (`&&`)、OR (`||`) 和 NOT (`!`)。还可以包含括号。如果标记表达式只包含 OR，那么限制为 20 个标记；否则限制为 6 个标记。

可以使用标记表达式定向发送到很多可能匹配的标记，而不是一个标记。

标记表达式

```
(follows_RedSox || follows_Cardinals) && location_Boston
```

下面的示例所示的应用程序可以接收有关特定音乐团体的 Toast 通知。这种情况下，路由通知的简单方法是用表示不同乐队的标记来标注注册点。

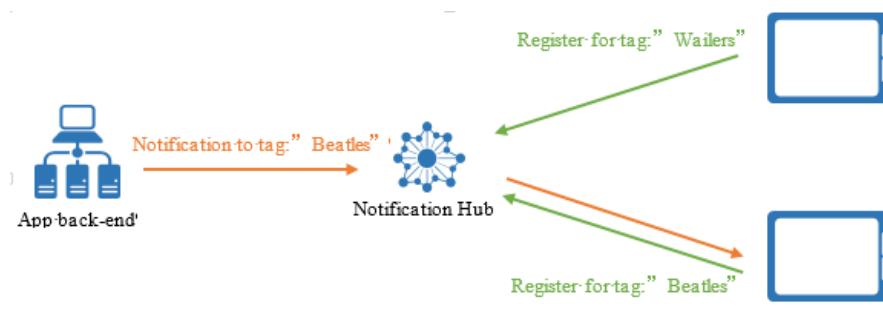


图 9.11：标记路由

实验 A：使用队列和 Service Bus 来管理 Azure 中 Web 应用程序之间的通信

场景

既然已经可以在辅助角色中生成签到表，那么现在需要一种可伸缩且一致的方式来为辅助角色排队消息。您已经决定使用 Azure 队列机制，这样就可以单独缩放辅助角色，以满足队列大小的需要。本实验中，您将先使用存储队列来实现管理 Web 应用程序与辅助角色之间的通信。然后，将该实现替换为使用 Service Bus 队列的实现。

目前，企业内部的 Contoso Events 应用程序使用 WCF 服务来列出某地附近的酒店。您想继续使用 WCF 服务，但是不能修改公司防火墙。此外，您不想暴露 WCF 服务的真实网络位置。您决定使用 Service Bus 中继，这样就可以只将一个公共端点提供给客户端应用程序。您将使用 Contoso Events Web 应用程序中的端点开始。

 **注释：**本章为您提供了两套实验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成实验 A；如果您拿到的是中国版 Windows Azure 帐户，请转到实验 B。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 使用管理门户创建 Service Bus 命名空间。
- 创建存储队列消息。
- 使用 Azure 存储队列消息。
- 创建 Service Bus 队列消息。
- 使用 Service Bus 队列消息。
- 将 WCF 服务的 XML 配置修改为使用 Service Bus 中继绑定。
- 将 WCF 客户端的 C# 配置修改为使用 Service Bus 中继绑定。

实验设置

预计时间：90 分钟

在开始这个实验之前，必须完成第 2 章的实验 A。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].cloudapp.net:[虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为**Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**

- 密码: **AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。
 6. 本章为您提供了两套实验方案, 如果您拿到的是国际版 Microsoft Azure 帐户, 请继续完成**实验 A**; 如果您拿到的是中国版 Windows Azure 帐户, 请转到**实验 B**。具体请咨询培训讲师, 并在讲师的指导下完成实验。

练习 1: 创建 Azure Service Bus 命名空间

场景

在此练习中, 您将:

- 使用管理门户创建 Service Bus 命名空间。

此练习的主要任务如下:

1. 使用管理门户创建 Service Bus 命名空间

► 任务 1: 使用管理门户创建 Service Bus 命名空间



注释: Service Bus 功能在预览门户暂不可用, 因此这个实验将使用 Azure 管理门户。

1. 登录到 Azure 管理门户(<https://manage.windowsazure.com>)。
2. 单击右上角的帐户名前的地球图标。
3. 选择中文(简体)。
4. 查看订阅的 **Service Bus** 命名空间。
5. 根据以下信息, 创建一个新的 **Service Bus** 命名空间:
 - 命名空间名称: **sb28532[自定义名称]**
 - 区域: **选择最接近你的区域**
 - 类型: **消息**
 - 消息传递层: **Standard**
6. 记录下新建的 Service Bus 命名空间的连接字符串。
7. 根据以下信息, 为命名空间创建一个新的**队列**:
 - 队列名称: **signin**
 - 命名空间: **sb28532[自定义名称]**

结果: 完成此练习后, 您将使用管理门户创建了 Service Bus 命名空间。

练习 2: 将 Azure 队列存储用于文档生成

场景

在此练习中, 您将:

- 使用存储队列创建队列消息。

- 使用来自存储队列的队列消息。

此练习的主要任务如下：

- 将辅助角色更新为使用来自队列的请求
- 将管理应用程序更新为将请求添加到队列
- 生成测试数据
- 调试和验证应用程序

► **任务 1：将辅助角色更新为使用来自队列的请求**

- 从以下位置打开 **Contoso.Events** 解决方案：
 - 文件路径：**Allfiles(F):\Mod09\Labfiles\Starter\Queues\Contoso.Events**
- 打开 **TableStorageQueueHelper.cs** 文件。
- 在 **TableStorageQueueHelper** 构造函数内，使用以下代码来获取队列客户和队列的名称：

```
CloudStorageAccount storageAccount = base.StorageAccount;
_queueClient = storageAccount.CreateCloudQueueClient();
_signInQueueName = CloudConfigurationManager.GetSetting("SignInQueueName");
```

- 在 **Receive** 方法内，删掉现有代码。
- 创建一个新的 **CloudQueue** 实例，并且确保实例存在，代码如下：

```
CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);
queue.CreateIfNotExists();
```

- 通过调用 **CloudQueue** 类的 **GetMessage** 方法来获取 **CloudQueueMessage** 变量，然后把消息变量传递给 **TableStorageQueueMessage** 类的构造函数，代码如下：

```
CloudQueueMessage message = queue.GetMessage();
return new TableStorageQueueMessage(message);
```

- 在 **CompleteMessage** 方法内，创建一个新的 **CloudQueue** 实例并确保实例存在，代码如下：

```
CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);
queue.CreateIfNotExists();
```

- 调用 **CloudQueue** 类的 **DeleteMessage** 方法，把 **CloudQueueMessage** 变量作为第一个参数，代码如下：

```
queue.DeleteMessage(message);
```

► **任务 2：将管理应用程序更新为将请求添加到队列**

- 打开 **SignInSheetViewModel.cs** 文件。
- 在 **GenerateSignInSheetTableStorage** 方法内，创建一个有 **Microsoft.WindowsAzure.Storage.ConnectionString** 设定值的 **CloudStorageAccount** 实例和 **CloudQueueClient** 实例，代码如下：

```
CloudStorageAccount storageAccount =
CloudStorageAccount.Parse(tableStorageConnectionString);
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
```

- 创建一个新的 **CloudQueue** 实例，使其使用 **SignQueueName** 云设定值，并且确保实例存在，代码如下：

```
CloudQueue queue = queueClient.GetQueueReference(signInQueueName);  
queue.CreateIfNotExists();
```

4. 通过创建一个新的 **CloudQueueMessage** 类的实例，使字符串 **message** 作为构造函数的参数，来把消息添加到队列中，然后为新实例添加一个 **CloudQueue AddMessage** 方法，代码如下：

```
CloudQueueMessage queueMessage = new CloudQueueMessage(message);  
queue.AddMessage(queueMessage);
```

► 任务 3：生成测试数据

1. 启动 **Windows Azure Storage Emulator – v3.4**。
2. 通过调试 **Contoso.Events.Data.Generation** 项目生成 SQL 和存储表数据。

► 任务 4：调试和验证应用程序

1. 配置解决方案的多启动项目为以下选项：
 - **Contoso.Events.Cloud**: 启动
 - **Contoso.Events.Management**: 开始执行（不调试）
2. 调试解决方案。

 **注释：**你将配置好解决方案，然后调试 **Contoso.Events.Cloud** 项目并简单地运行 **Contoso.Events.Management** 项目。这确保了 2 个页面应用程序能正常运行不会引起冲突。

 **注释：**从始至终，这个课程都将使用 **Azure Compute Emulator** 来测试和调试云服务。在将云服务布置到 Azure 平台之前，此模拟器提供了一个快捷的本地选项来测试云服务。偶尔，它将会遇到一些奇怪的问题，比如错误的端口号，页面不能显示等。如果出现这种情况，你只要关闭这个模拟器，在下次调试项目时，Visual Studio 将会自动启动该模拟器。

3. 打开 **Contoso Events (Contoso.Events.Cloud)** 站点，然后通过 IIS Express 打开 **Contoso.Events.Administration** 站点。
4. 切换到 **Contoso.Events–Microsoft Visual Studio** 窗口。
5. 打开 **WorkerRole.cs** 文件。
6. 给以下代码行添加一个断点：

```
HandleMessage(message);
```

7. 切换到 **Home-Contoso.Events.Administration** 窗口。
8. 在 **Administration** 页面应用程序的主页，选择任意 event，生成一个 sign-in sheet，然后下载 sign-in sheet。
9. 验证应用程序在断点处暂停运行。
10. 停止调试，然后关闭 Internet Explorer。

结果：完成此练习后，您将创建并使用了存储队列中的消息。

MCT USE ONLY. STUDENT USE PROHIBITED

练习 3：将 Service Bus 队列用于文档生成

场景

在此练习中，您将：

- 使用 Service Bus 队列创建队列消息。
- 使用来自 Service Bus 队列的队列消息。

此练习的主要任务如下：

1. 将辅助角色更新为使用来自队列的请求
2. 将管理应用程序更新为将请求添加到队列
3. 调试和验证应用程序

► **任务 1：将辅助角色更新为使用来自队列的请求**

1. 更新 **Contoso.Events.Cloud** 项目的 **ServiceConfiguration.Local.cscfg** 文件，为 **Microsoft.ServiceBus.ConnectionString** 指定值为 **Service Bus** 命名空间的连接字符串：
 - Name: **Microsoft.ServiceBus.ConnectionString**
 - Value: **[Service Bus 连接字符串]**
2. 打开 **ServiceBusQueueHelper.cs** 文件。
3. 在 **ServiceBusQueueHelper** 构造函数内，通过以下代码获取队列客户：

```
string serviceBusConnectionString =
CloudConfigurationManager.GetSetting("Microsoft.ServiceBus.ConnectionString");
string signInQueueName = CloudConfigurationManager.GetSetting("SignInQueueName");
_client = QueueClient.CreateFromConnectionString(serviceBusConnectionString,
signInQueueName);
```

4. 在 **Receive** 方法内，删掉已有代码。
5. 调用 **QueueClient** 类的 **Receive** 方法，把值存储于 **BrokeredMessage**，然后把新的变量传递给 **ServiceBusQueueMessage** 构造函数，代码如下：

```
BrokeredMessage message = _client.Receive();
return new ServiceBusQueueMessage(message);
```

6. 在 **CompleteMessage** 方法内，调用 **BrokeredMessage** 参数的 **Complete** 方法，代码如下：

```
message.Complete();
```

7. 在 **AbandonMessage** 方法，调用 **BrokeredMessage** 参数的 **Abandon** 方法，代码如下：

```
message.Abandon();
```

8. 打开 **WorkerRole.cs** 文件。

9. 定位到以下创建 **TableStorageQueueHelper** 实例的代码行：

```
var queueHelper = new TableStorageQueueHelper();
```

用以下创建新 **ServiceBusQueueHelper** 类实例的代码来替换以上代码：

```
var queueHelper = new ServiceBusQueueHelper();
```

► 任务 2：将管理应用程序更新为将请求添加到队列

1. 更新 **Contoso.Events.Management** 项目的 **Web.config** 文件，为 **Microsoft.ServiceBus.ConnectionString** 指定其值为 **Service Bus** 命名空间的连接字符串：

- Key: **Microsoft.ServiceBus.ConnectionString**
- Value: [Service Bus 连接字符串]

2. 打开 **SignInSheetViewModel.cs** 文件。

3. 在构造函数内，定位到以下代码行：

```
GenerateSignInSheetTableStorage(context, eventItem, messageString);
```

4. 用以下代码行替换上面的代码行：

```
GenerateSignInSheetServiceBus(context, eventItem, message);
```

5. 在 **GenerateSignInSheetServiceBus** 方法内，创建一个带 **Microsoft.WindowsAzure.Storage.ConnectionString** 设定值和 **SignInQueueName** 设定值的 **QueueClient** 实例，代码如下：

```
QueueClient client = QueueClient.CreateFromConnectionString(serviceBusConnectionString,  
signInQueueName);
```

6. 通过创建一个新的 **BrokeredMessage** 类的实例，使 **QueueMessage** 消息作为构造函数参数，来把消息添加到队列中，然后给新实例添加 **QueueClient Send** 方法，代码如下：

```
BrokeredMessage queueMessage = new BrokeredMessage(message);  
client.Send(queueMessage);
```

► 任务 3：调试和验证应用程序

1. 调试解决方案。
2. 打开 **Contoso Events (Contoso.Events.Cloud)** 站点，然后通过 IIS Express 打开 **Contoso.Events.Administration** 站点。
3. 切换到 **Home-Contoso.Events.Administration** 窗口。
4. 在 **Administration** 页面应用程序的主页，选择任意 event，生成一个 sign-in sheet，然后下载 sign-in sheet。
5. 验证应用程序在断点处暂停运行。
6. 停止调试，然后关闭 Internet Explorer。

结果：完成此练习后，您将创建并使用了 Service Bus 队列中的消息。

练习 4：使用 Service Bus 中继连接 WCF 服务和客户端

场景

在此练习中，您将：

- 管理 WCF 服务的 XML 配置。
- 将现有 WCF 服务修改为使用 Service Bus 中继绑定。
- 管理 WCF 服务的 C# 配置。

- 将 WCF 客户端更新为使用 Service Bus 中继绑定。
- 使用 Service Bus 中继端点来使用 WCF 服务。

此练习的主要任务如下：

- 使用控制台应用程序测试现有 WCF 服务
- 将 WCF 服务端点修改为使用 Service Bus 中继
- 使用控制台应用程序测试中继的 WCF 服务
- 将云 Web 应用程序更新为连接到中继的 WCF 服务
- 调试云 Web 应用程序

► **任务 1：使用控制台应用程序测试现有 WCF 服务**

- 从以下位置打开 **Contoso.Events** 解决方案：
 - 文件路径：**Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events**
- 从以下位置打开 **Contoso.Events.Lodging.Client** 解决方案：
 - 文件路径：**Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events.Lodging.Client**
- 切换到 **Contoso.Events—Microsoft Visual Studio** 窗口。
- 调试 **Contoso.Events** 解决方案的 **Contoso.Events.Lodging** 项目。
- 切换到 **Contoso.Events.Lodging.Client—Microsoft Visual Studio** 窗口。
- 调试 **Contoso.Events.Lodging.Client** 解决方案。
- 查看控制台应用程序中列出的酒店。
- 停止调试以上两个解决方案。

► **任务 2：将 WCF 服务端点修改为使用 Service Bus 中继**

- 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
- 打开 **Contoso.Events.Lodging** 项目的 **App.config** 文件。
- 更新 service endpoint，用 **netTcpRelayBinding** 替换 **netTcpBinding**。
- 更新 service endpoint，使用以下 address：
 - sb://[namespace].servicebus.windows.net/lodging**
 - 用命名空间名称代替 **[namespace]** 标签。
- 根据以下代码，更新 **ServiceBusRelayBehavior** 元素：

```
<behavior name="ServiceBusRelayBehavior">
<transportClientEndpointBehavior>
<tokenProvider>
<sharedAccessSignature keyName="[keyName]" issuerSecret="[key]" /> </tokenProvider>
</transportClientEndpointBehavior>
</behavior>
```

- 定位到 **sharedAccessSignature** 元素内的 **keyName** 属性，然后用之前记录的 Service Bus 连接字符串的 **SharedAccessKeyName** 的值替换其值。
- 定位到 **sharedAccessSignature** 元素内的 **key** 属性，然后然后用之前记录的 Service Bus 连接字符串的 **SharedAccessKey** 的值替换其值。

 **注释：**例如你的连接字符串为：

MCT USE ONLY. STUDENT USE PROHIBITED

```
Endpoint=sb://contoso.servicebus.windows.net/;
SharedAccessKeyName=RootManageSharedAccessKey;
SharedAccessKey=ABC123/J3y+tk=
keyName 的值就是 "RootManageSharedAccessKey"然后 key 的值就是
"ABC123/J3y+tk="。
```

8. 调试解决方案。

► 任务 3：使用控制台应用程序测试中继的 WCF 服务

1. 切换到 **Contoso.Events.Lodging.Client – Microsoft Visual Studio** 窗口。
2. 打开 **Contoso.Events.Lodging.Client** 项目的 **Program.cs** 文件。
3. 定位到 **GetHotels** 方法。
4. 定位到以下创建 Uri 的代码：

```
Uri serviceUri = new Uri("net.tcp://localhost:8000/lodging", UriKind.Absolute);
```

用以下代码替换上面的代码：

```
Uri serviceUri = ServiceBusEnvironment.CreateServiceUri("sb", "[namespace]", "lodging");
```

5. 用 Service Bus 命名空间名称替换**[namespace]**的值。
6. 把 **NetTcpBinding** 替换成 **NetTcpRelayBinding**。
7. 在创建实例 **ChannelFactory<ILodgingService>** 之后，通过使用以下代码创建一个 endpoint behavior：

```
TransportClientEndpointBehavior endpointBehavior = new TransportClientEndpointBehavior();
endpointBehavior.TokenProvider =
TokenProvider.CreateSharedAccessSignatureTokenProvider("[keyName]", "[key]");
cf.Endpoint.EndpointBehaviors.Add(endpointBehavior);
```

8. 用之前记录的 Service Bus 连接字符串的 **SharedAccessKeyName** 的值替换**[keyName]**的值。
9. 用之前记录的 Service Bus 连接字符串的 **SharedAccessKey** 的值替换**[key]**的值。
10. 调试解决方案。

► 任务 4：将云 Web 应用程序更新为连接到中继的 WCF 服务

1. 拷贝 **Program.cs** 文件的 **GetHotels** 方法。
2. 切换到 **Contoso.Events–Microsoft Visual Studio** 窗口，然后停止调试。
3. 打开 **Contoso.Events.ViewModels** 项目的 **HotelsViewModel.cs** 文件。
4. 用之前拷贝的方法代替以下 **GetHotels** 方法：

```
private static IEnumerable<Hotel> GetHotels()
{
    return Enumerable.Empty<Hotel>();
}
```

► 任务 5：调试云 Web 应用程序

1. 配置 Visual Studio 2013，使用多启动项目，配置信息如下：
 - Contoso.Events.Cloud: 开始执行（不调试）
 - Consoto.Events.Lodging: 开始执行（不调试）

- Contoso.Events.Management: 无
2. 打开 **Windows Azure Storage Emulator – v3.4**。
 3. 调试 **Contoso.Events.Data.Generation** 项目来生成 SQL 和 Azure 存储表数据。
 4. 关闭 **Microsoft Visual Studio 2013**。
 5. 以管理员身份重新运行 Microsoft Visual Studio 2013。
 6. 打开 **Contoso.Events** 解决方案:
 - 文件路径: **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events**
 7. 启动 **Contoso.Events** 解决方案。

结果: 完成此练习后, 您将使用 Service Bus 中继将现有 WCF 应用程序连接到了客户端。

问题: 在项目发布后, 有哪些方法可用来将 WCF 绑定从其企业内部版本自动转变为 Service Bus 中继绑定?

MCT USE ONLY. STUDENT USE PROHIBITED

实验 B：在中国版 Windows Azure 中使用队列和 Service Bus 来管理 Web 应用程序间的通信

场景

既然已经可以在辅助角色中生成签到表，那么现在需要一种可伸缩且一致的方式来为辅助角色排队消息。您已经决定使用 Azure 队列机制，这样就可以单独缩放辅助角色，以满足队列大小的需要。本实验中，您将先使用存储队列来实现管理 Web 应用程序与辅助角色之间的通信。然后，将该实现替换为使用 Service Bus 队列的实现。

目前，企业内部的 Contoso Events 应用程序使用 WCF 服务来列出某地附近的酒店。您想继续使用 WCF 服务，但是不能修改公司防火墙。此外，您不想暴露 WCF 服务的真实网络位置。您决定使用 Service Bus 中继，这样就可以只将一个公共端点提供给客户端应用程序。您将使用 Contoso Events Web 应用程序中的端点开始。

目标

完成本实验后，您将能够：

- 使用管理门户创建 Service Bus 命名空间。
- 创建存储队列消息。
- 使用 Azure 存储队列消息。
- 创建 Service Bus 队列消息。
- 使用 Service Bus 队列消息。
- 将 WCF 服务的 XML 配置修改为使用 Service Bus 中继绑定。
- 将 WCF 客户端的 C# 配置修改为使用 Service Bus 中继绑定。

预计时间：90 分钟

在开始这个实验之前，必须完成第 2 章的实验 B。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击开始，键入远程，然后单击远程桌面连接。
2. 在远程桌面连接窗口中，在计算机框中选定以下格式的虚拟机名称：
vm28532[自定义名称].chinacloudapp.cn:[虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击连接。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到世纪互联运营的 Windows Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

MCT USE ONLY STUDENT USE PROHIBITED

练习 1：创建 Azure Service Bus 命名空间

场景

在此练习中，您将：

- 使用管理门户创建 Service Bus 命名空间。

此练习的主要任务如下：

- 使用管理门户创建 Service Bus 命名空间。

► 任务 1：使用管理门户创建 Service Bus 命名空间

- 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
- 进入 <https://manage.windowsazure.cn>。
- 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
- 如果您的界面语言不是中文，可以单击页面右上角的地球图表，并在菜单中选择中文(简体)语言。
- 在页面左侧的导航栏中，滚动鼠标找到 **Service Bus**，单击 **Service Bus**。
- 在页面下方的工具栏左侧，单击新建按钮。
- 在弹出的新建菜单中，选择 **SERVICE BUS**，单击队列，单击自定义创建。
- 在添加新队列窗口中，执行以下步骤：
 - 在队列名称文本框中，输入 **signin**。
 - 在区域下拉框中，选择一个距离你较近的区域。
 - 在命名空间下拉框中，选择创建新命名空间。
 - 在命名空间名称文本框中，输入 **sb28532[自定义名称]**。
 - 在窗口右下角，单击下一步箭头。
- 在配置队列窗口中，单击完成按钮。
- 在 service bus 页面，在列表中，单击进入新创建的命名空间，命名空间名称类似于 **sb28532[自定义名称]**。
- 在页面下方的工具栏中，单击连接信息。
- 在访问连接信息对话框，记录 RootManageSharedAccessKey 的连接字符串。

 **注释：**你需要从 SAS 列表里，记下连接字符串。

- 关闭访问连接信息对话框。

结果：完成此练习后，您将使用管理门户创建了 Service Bus 命名空间。

练习 2：将 Azure 队列存储用于文档生成

场景

在此练习中，您将：

- 使用存储队列创建队列消息。
- 使用来自存储队列的队列消息。

此练习的主要任务如下：

1. 更新辅助角色以处理来自队列的请求
2. 更新 Administration 应用以向队列中添加请求
3. 生成测试数据
4. 调试并验证应用程序

► **任务 1：更新辅助角色以处理来自队列的请求**

1. 在 **Start** 屏幕，单击 **Desktop**。
2. 在任务栏，单击 **File Explorer** 磁贴。
3. 打开 **Allfiles(F):\Mod09\Labfiles\Starter\Queues\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格，展开 **Roles** 解决方案目录。
5. 展开 **Contoso.Events.Worker** 项目。
6. 双击 **TableStorageQueueHelper.cs** 文件。
7. 在 **TableStorageQueueHelper** 类的构造函数结尾，在右括号之前，把基类的 **StorageAccount** 属性保存到 *CloudStorageAccount* 变量中：

```
CloudStorageAccount storageAccount = base.StorageAccount;
```

8. 调用 **CreateCloudQueueClient** 方法并把返回结果赋值给变量 *_queueClient*:

```
_queueClient = storageAccount.CreateCloudQueueClient();
```

9. 调用静态方法 **CloudConfigurationManager.GetSetting** 并把返回结果赋值给变量 *_signInQueueName*:

```
_signInQueueName = CloudConfigurationManager.GetSetting("SignInQueueName");
```

10. 在 **TableStorageQueueHelper** 类，找到如下签名的方法:

```
IQueueMessage<CloudQueueMessage> Receive()
```

11. 删除此行代码:

```
return new TableStorageQueueMessage(null);
```

12. 在 **Receive** 方法结尾，右括号前，通过调用 *CloudQueueClient* 变量的 **GetQueueReference** 方法和队列的字符串名字创建一个 **CloudQueue** 类的实例，代码如下:

```
CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);
```

13. 调用 **CreateIfNotExists** 方法来保证队列存在。

```
queue.CreateIfNotExists();
```

14. 在 **Receive** 方法结尾，右括号前，调用 **CloudQueue** 类的 **GetMessage** 方法，并把返回结果赋值给一个 *CloudQueueMessage* 变量，代码如下:

```
CloudQueueMessage message = queue.GetMessage();
```

15. 把 *CloudQueueMessage* 变量传给 **TableStorageQueueMessage** 类的构造函数，并返回结果:

```
return new TableStorageQueueMessage(message);
```

16. 在 **CompleteMessage** 方法结尾, 右括号前, 通过调用 *CloudQueueClient* 变量的 **GetQueueReference** 方法和一个字符串队列名来创建一个 **CloudQueue** 类的实例, 代码如下:

```
CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);
```

17. 调用 **CreateIfNotExists** 方法保证队列存在:

```
queue.CreateIfNotExists();
```

18. 在 **CompleteMessage** 方法结尾, 右括号前, 用 *CloudQueueMessage* 变量作参数, 调用方法 **DeleteMessage**, 代码如下:

```
queue.DeleteMessage(message);
```

► 任务 2: 更新 **Administration** 应用以向队列中添加请求

1. 在解决方案资源管理器窗格, 展开 **Shared** 解决方案目录。
2. 在解决方案资源管理器窗格, 展开 **Contoso.Events.ViewModels** 项目。
3. 双击 **SignInSheetViewModel.cs** 文件。
4. 在 **GenerateSignInSheetTableStorage** 方法开头, 左括号后面, 通过调用静态方法 **CloudStorageAccount.Parse** 和存储表的连接字符串来创建 **CloudStorageAccount** 实例, 代码如下:

```
CloudStorageAccount storageAccount =
CloudStorageAccount.Parse(tableStorageConnectionString);
```

5. 调用 *CloudStorageAccount* 变量的 **CreateCloudQueueClient** 方法, 创建一个 **CloudQueueClient** 类的实例:

```
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
```

6. 在以上代码之后, 通过调用 *CloudQueueClient* 变量的 **GetQueueReference** 方法和队列名的变量创建一个 **CloudQueue** 类的实例, 代码如下:

```
CloudQueue queue = queueClient.GetQueueReference(signInQueueName);
```

7. 调用 **CloudQueue** 类的 **CreateIfNotExists** 方法保证队列存在:

```
queue.CreateIfNotExists();
```

8. 在以上代码之后, 把一个字符串 **message** 传递给 **CloudQueueMessage** 类的构造函数来创建一个实例:

```
CloudQueueMessage queueMessage = new CloudQueueMessage(message);
```

9. 调用 *CloudQueue* 变量的 **AddMessage** 方法, 使用 **CloudQueueMessage** 作为参数:

```
queue.AddMessage(queueMessage);
```

► 任务 3: 生成测试数据

1. 在 Start 屏幕, 输入 **Azure Storage Emulator**。
2. 单击 **Windows Azure Storage Emulator-v3.4** 磁贴。

3. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
4. 在解决方案资源管理器窗格，右键单击 **Contoso.Events.Data.Generation** 项目，单击调试->启动新实例。

 **注释：**如果弹出保存只读文件对话框，单击覆盖按钮。

► 任务 4：调试并验证应用程序

1. 在解决方案资源管理器窗格，展开 **Administration** 文件夹，展开 **Contoso.Events.Management** 项目，双击 **Web.config** 文件，找到`<connectionStrings>`元素，找到 name 的值为 `EventsContextConnectionString` 的元素，将 `connectionString` 的值用以下代码替换：

```
Data Source=(localdb)\v11.0;Initial Catalog=EventsContextModule9Lab;Pooling=True;Integrated Security=True
```

2. 在解决方案资源管理器窗格，右键单击解决方案 ‘**Contoso.Events**’，单击属性。
3. 在解决方案 ‘**Contoso.Events**’ 的属性对话框，执行以下步骤：
 - a. 在属性对话框左边的导航菜单，选择通用属性=>启动项目。
 - b. 选择多启动项目。
 - c. 设置 **Contoso.Events.Cloud** 项目，将操作修改为启动。
 - d. 设置 **Contoso.Events.Management** 项目，将操作修改为开始执行（不调试）。
 - e. 确保其他项目的操作是无。
 - f. 单击 **OK**。
4. 在调试菜单，单击启动调试。

 **注释：**如果弹出保存只读文件对话框，单击覆盖按钮。

在这门课程中，我们会用到 **Azure Compute Emulator** 来测试和调试我们的云服务。在发布云服务到 Azure 平台之前，模拟器提供了一个快速的本地测试你的云服务的方法。偶尔你会遇到一些奇怪的模拟器错误，比如不正确的端口号或页面不显示，你只需要关闭模拟器，下次 Visual Studio 启动调试的时候会自动启动模拟器。

5. 在通知区域，单击箭头来查看运行的应用。
6. 找到并右键单击 **IIS Express** 图标。
7. 在 **View Sites** 下面找到 **Contoso.Events.Management** 选项并单击，然后单击 **Browse Applications** 下面的 URL。
8. 鼠标指向 **View Sites** 下的 **Contoso.Events.Cloud** 选项，然后单击 **Browse Applications** 下的 URL。
9. 在桌面，单击 **Contoso.Events–Microsoft Visual Studio** 窗口。
10. 单击视图菜单，然后选择解决方案资源管理器选项。
11. 在解决方案资源管理器窗格，展开 **Roles** 文件夹。
12. 在解决方案资源管理器窗格，展开 **Contoso.Events.Worker** 项目。
13. 双击 **WorkerRole.cs** 文件。
14. 找到 **Run** 方法。

15. 在 try-catch 代码块里找到此行代码:

```
HandleMessage(message);
```

16. 右键单击此行代码，在弹出菜单中选中**断点->插入断点**。

17. 在浏览器窗口中，找到 **Home-Contoso.Events.Administration** 窗口。

18. 在 **Administration** Web 应用程序主页，单击 **Events** 按钮到 events 列表。

19. 对任意一个 event，单击 **Sign-In Sheet**。

20. 在 sign-in 页面，有以下信息提示 sign-in sheet 正在被创建： **Sign-In Document Generation in Progress**。

21. 等待一分钟，辅助角色（Worker Role）会接收到队列信息。

22. 确保应用程序的执行停在断点处。

23. 按 F5 继续执行。

24. 等待一分钟，然后刷新 sign-in sheet 页面。

25. 单击 **Sign-In Sheet** 从服务器下载 sign-in sheet。

26. 关闭 **Internet Explorer** 程序。

结果：完成此练习后，您将创建并使用了存储队列中的消息。

练习 3：将 Service Bus 队列用于文档生成

场景

在此练习中，您将：

- 使用 Service Bus 队列创建队列消息。
- 使用来自 Service Bus 队列的队列消息。

此练习的主要任务如下：

1. 更新辅助角色以处理来自队列的请求
2. 更新 Administration 应用以向队列中添加请求
3. 调试并验证应用程序

► 任务 1：更新辅助角色以处理来自队列的请求

1. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Cloud** 项目。
2. 双击 **ServiceConfiguration.Local.cscfg** 文件。
3. 找到 name 的值为 **Microsoft.ServiceBus.ConnectionString** 的 **Setting** 元素。
4. 用你之前记录的连接字符串替换 **ConnectionString** 的值。
5. 在解决方案资源管理器窗格中，展开 **Roles** 文件夹。
6. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Worker** 项目。
7. 双击 **ServiceBusQueueHelper.cs** 文件。
8. 在 **ServiceBusQueueHelper** 构造函数结尾，右括号之前，把 **Microsoft.ServiceBus.ConnectionString** 的值存储在一个字符串中，代码如下：

MCT USE ONLY. STUDENT USE PROHIBITED

```
string serviceBusConnectionString =  
CloudConfigurationManager.GetSetting("Microsoft.ServiceBus.ConnectionString");
```

9. 把队列名赋值给一个字符串变量。

```
string signInQueueName = CloudConfigurationManager.GetSetting("SignInQueueName");
```

10. 用队列名和连接字符串作为参数，调用静态方法 **QueueClient.CreateFromConnectionString**，把结果赋值给 `_client` 变量，代码如下：

```
_client = QueueClient.CreateFromConnectionString(serviceBusConnectionString,  
signInQueueName);
```

11. 在 **ServiceBusQueueHelper** 类里，找到以下签名的方法：

```
IQueueMessage<BrokeredMessage> Receive()
```

12. 删除此行代码：

```
return new ServiceBusQueueMessage(null);
```

13. 在 **Receive** 方法结尾，右括号前，调用 `_client` 变量的 **Receive** 方法，并把返回结果赋值给一个 `BrokeredMessage` 变量，代码如下：

```
BrokeredMessage message = _client.Receive();
```

14. 把 `BrokeredMessage` 变量传给 **ServiceBusQueueMessage** 类的构造函数，并返回结果：

```
return new ServiceBusQueueMessage(message);
```

15. 在 **CompleteMessage** 方法结尾，右括号前，调用 `message` 参数的 **Complete** 方法，代码如下：

```
message.Complete();
```

16. 在 **AbandonMessage** 方法结尾，右括号前，调用 `message` 参数的 **Abandon** 方法，代码如下：

```
message.Abandon();
```

17. 在解决方案资源管理器窗格，展开 **Roles** 文件夹。

18. 在解决方案资源管理器窗格，展开 **Contoso.Events.Worker** 项目。

19. 双击 **WorkerRole.cs** 文件。

20. 找到创建 **TableStorageQueueHelper** 实例的代码：

```
var queueHelper = new TableStorageQueueHelper();
```

将其替换为创建 **ServiceBusQueueHelper** 的实例：

```
var queueHelper = new ServiceBusQueueHelper();
```

► 任务 2：更新 **Administration** 应用以向队列中添加请求

1. 在解决方案资源管理器窗格，展开 **Administration** 文件夹，然后展开 **Contoso.Events.Management** 项目。
2. 双击 **Web.config** 文件。
3. 找到 **appSettings** 元素。

4. 找到 key 的值为 **Microsoft.ServiceBus.ConnectionString** 的 **add** 元素。
5. 用之前记录的连接字符串替换它的 **value** 属性的值。
6. 在**解决方案资源管理器**窗格，展开 **Shared** 文件夹。
7. 在**解决方案资源管理器**窗格，展开 **Contoso.Events.ViewModels** 项目。
8. 双击 **SignInSheetViewModel.cs** 文件。
9. 在构造方法中，找到此行代码：

```
GenerateSignInSheetTableStorage(context, eventItem, messageString);
```

10. 用以下代码替换它：

```
GenerateSignInSheetServiceBus(context, eventItem, message);
```

11. 在 **GenerateSignInSheetServiceBus** 方法开始，左括号后，用连接字符串创建 **QueueClient** 的实例，代码如下：

```
QueueClient client = QueueClient.CreateFromConnectionString(serviceBusConnectionString,
signInQueueName);
```

12. 在以上代码后面，用 **QueueMessage** 作参数创建 **BrokeredMessage** 类的实例，代码如下：

```
BrokeredMessage queueMessage = new BrokeredMessage(message);
```

13. 用 **BrokeredMessage** 作为参数，调用 **QueueClient** 变量的 **Send** 方法：

```
client.Send(queueMessage);
```

► 任务 3：调试并验证应用程序

1. 在**调试**菜单，单击**启动调试**。

 **注释：**如果弹出**保存只读文件**对话框，单击**覆盖**按钮。

2. 在**通知区域**，单击箭头来查看运行的应用程序。
3. 找到**IIS Express** 图标，然后右键单击此图标。
4. 在**View Sites** 下面找到 **Contoso.Events.Management** 选项并单击，然后单击 **Browse Applications** 下面的 URL。
5. 在**View Sites** 下面找到 **Contoso.Events.Cloud** 选项并单击，然后单击 **Browse Applications** 下面的 URL。
6. 在**Administration** Web 应用程序主页，单击 **Events** 按钮到 events 列表。
7. 单击任意一个 event 的 **Sign-In Sheet** 按钮。
8. 在 sign-in 页面，有以下信息提示 sign-in sheet 正在被创建：**Sign-In Document Generation in Progress**。
9. 等待一分钟，辅助角色会接收到队列信息。
10. 确保应用程序的执行停在断点处。
11. 按 F5 继续执行。
12. 等待一分钟，然后刷新 sign-in sheet 页面。

13. 单击 **Sign-In Sheet** 从服务器下载 sign-in sheet。
14. 关闭 **Internet Explorer** 程序。
15. 关闭 **Contoso.Events–Microsoft Visual Studio** 窗口。

结果: 完成此练习后，您将创建并使用了 Service Bus 队列中的消息。

练习 4：使用 Service Bus 中继连接 WCF 服务和客户端

场景

在此练习中，您将：

- 管理 WCF 服务的 XML 配置。
- 将现有 WCF 服务修改为使用 Service Bus 中继绑定。
- 管理 WCF 服务的 C# 配置。
- 将 WCF 客户端更新为使用 Service Bus 中继绑定。
- 使用 Service Bus 中继端点来使用 WCF 服务。

此练习的主要任务如下：

1. 使用控制台应用程序测试现有 WCF 服务
2. 修改 WCF 服务端点以使用 Service Bus 中继
3. 使用控制台应用程序测试中继的 WCF 服务
4. 更新云 Web 应用程序以连接到中继的 WCF 服务
5. 调试云 Web 应用程序

► 任务 1：使用控制台应用程序测试现有 WCF 服务

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏，单击 **File Explorer** 图标。
3. 浏览到 **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在 Start 屏幕，单击 **Desktop** 磁贴。
5. 在任务栏，单击 **File Explorer** 图标。
6. 浏览到 **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events.Lodging.Client**，然后双击 **Contoso.Events.Lodging.Client.sln**。
7. 切换到 **Contoso.Events–Microsoft Visual Studio** 窗口。
8. 在解决方案资源管理器窗格，展开 **On-Premise** 文件夹，右键单击 **Contoso.Events.Lodging** 项目，然后单击**设为启动项目**。
9. 在调试菜单，单击**启动调试**。
10. 你会看到一个弹出框，提示 WCF 服务没有元数据，询问是否退出调试。单击 **No**。
11. 最小化 **WCF Test Client**。
12. 切换到 **Contoso.Events.Lodging.Client – Microsoft Visual Studio** 窗口。
13. 在调试菜单，单击**启动调试**。
14. 确认列出四个旅馆名字。

15. 在控制台窗口按任意键关闭窗口。
 16. 切换到 **Contoso.Events–Microsoft Visual Studio** 窗口。
 17. 在调试菜单，单击**停止调试**。
- **任务 2：修改 WCF 服务端点以使用 Service Bus 中继**
1. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
 2. 在解决方案资源管理器窗格，展开 **On-Premise** 文件夹。
 3. 在解决方案资源管理器窗格，展开 **Contoso.Events.Lodging** 项目。
 4. 双击 **App.config** 文件。
 5. 定位到 **system.serviceModel** XML 元素。
 6. 定位到 **services** XML 子元素。
 7. 定位到第一个 **contract** 值为 **Contoso.Events.Models.ILodgingService** 的 **endpoint** XML 子元素。
 8. 替换 **binding** 属性的值为 **netTcpRelayBinding**。
 9. 定位到第一个 **contract** 值为 **Contoso.Events.Models.ILodgingService** 的 **endpoint** XML 子元素。
 10. 替换 **address** 属性的值为 **sb://**。
 11. 在 **address** 属性的值追加 **Service Bus** 命名空间的名称。命名空间格式类似于 **sb28532[自定义名称]**。
 12. 在 **address** 属性的值之后追加 **.servicebus.chinacloudapi.cn/lodging**。
 13. 定位到 **system.serviceModel** XML 元素。
 14. 定位到 **behaviors** XML 子元素。
 15. 定位到 **endpointBehaviors** XML 子元素。
 16. 定位到第一个 **name** 值为 **ServiceBusRelayBehavior** 的 **behavior** XML 子元素，如下所示：
- ```
<behavior name="ServiceBusRelayBehavior">
</behavior>
```
17. 在 **behavior** 元素中，加上一个 **transportClientEndpointBehavior** 子元素，如下所示：
- ```
<transportClientEndpointBehavior>
</transportClientEndpointBehavior>
```
18. 在 **transportClientEndpointBehavior** 元素中，加上一个 **tokenProvider** 子元素，如下所示：
- ```
<tokenProvider>
</tokenProvider>
```
19. 在 **tokenProvider** 元素中，加上一个 **sharedAccessSignature** 子元素。
- ```
<sharedAccessSignature keyName="[KeyName]" key="[Key]" />
```
20. 在 **sharedAccessSignature** 元素中，删除 **keyName** 属性的值，替换为之前你记录的 Service Bus 连接字符串的 **SharedAccessKeyName** 值。
 21. 在 **sharedAccessSignature** 元素中，删除 **key** 属性的值，替换为之前记录的 Service Bus 连接字符串的 **SharedAccessKey** 值。

 **注释：**例如你的连接字符串可能是：
Endpoint=sb://contoso.servicebus.chinacloudapi.cn;

SharedAccessKeyName=RootManageSharedAccessKey;
SharedAccessKey=ABC123/J3y+tk=
你的 **keyName** 值为“RootManageSharedAccessKey”，你的 **key** 值为
“ABC123/J3y+tk=”。

22. 在解决方案资源管理器窗格，展开 **On-Premise** 文件夹，右键单击 **Contoso.Events.Lodging** 项目，
单击设为启动项目。
23. 在调试菜单，单击启动调试。
24. 你会看到一个弹出框，提示 WCF 服务没有元数据，询问是否退出调试。单击 **No**。
25. 最小化但是不要关闭 **WCF Test Client** 应用程序。

► 任务 3：使用控制台应用程序测试中继的 WCF 服务

1. 切换到 **Contoso.Events.Lodging.Client - Microsoft Visual Studio** 窗口。
2. 在解决方案资源管理器窗格，展开 **Contoso.Events.Lodging.Client** 项目。
3. 双击 **Program.cs** 文件。
4. 定位到此方法：

```
private static IEnumerable<Hotel> GetHotels()
```

5. 删除使用静态方法去创建 Uri 的代码，如下显示：

```
Uri serviceUri = new Uri("net.tcp://localhost:8000/lodging", UriKind.Absolute);
```

6. 用以下代码替换删除的代码，将协议、命名空间、名字作为参数，通过调用静态方法
ServiceBusEnvironment.CreateServiceUri 创建 Uri，如下所示：

```
Uri serviceUri = new Uri("sb://[namespace].servicebus.chinacloudapi.cn/lodging");
```

7. 将上一步代码中 **CreateServiceUri** 方法的参数 “[namespace]” 替换为 Service Bus 命名空间。
8. 删除以下代码：

```
new NetTcpBinding(),
```

9. 替换为创建一个 **NetTcpRelayBinding** 类的实例的代码：

```
new NetTcpRelayBinding(),
```

10. 在创建 **ChannelFactory<ILodgingService>** 实例的下一行，创建一个
TransportClientEndpointBehavior 的实例：

```
TransportClientEndpointBehavior endpointBehavior = new TransportClientEndpointBehavior();
```

11. 在下一行，调用静态方法 **TokenProvider.CreateSharedSecretTokenProvider**，返回结果赋值给变量
TransportClientEndpointBehavior 的 **TokenProvider** 属性：

```
endpointBehavior.TokenProvider =  
TokenProvider.CreateSharedAccessSignatureTokenProvider("[keyName]", "[key]");
```

12. 更改 **CreateSharedSecretTokenProvider** 方法的参数，用 Service Bus 连接字符串的
SharedAccessKey 值替换 “[key]” 字符串。
13. 更改 **CreateSharedSecretTokenProvider** 方法的参数，用 Service Bus 连接字符串的
SharedAccessKeyName 值替换 “[keyName]” 字符串。

14. 添加 **TransportClientEndpointBehavior** 的实例到 **cf.Endpoint** 变量的 **EndpointBehaviors** 集合属性：

```
cf.Endpoint.EndpointBehaviors.Add(endpointBehavior);
```

15. 在调试菜单，单击启动调试。
 16. 确认旅馆列表显示。
 17. 在控制台窗口按任意键关闭窗口。

► 任务 4：更新云 Web 应用程序以连接到中继的 WCF 服务

1. 定位到 **Contoso.Events.Lodging.Client** 项目的 **Program.cs** 文件。
2. 定位到此方法：

```
private static IEnumerable<Hotel> GetHotels()
```

3. 选中整个方法。
4. 右键单击选中的内容，单击**复制**来拷贝整个方法。
5. 切换到 **Contoso.Events–Microsoft Visual Studio** 窗口。
6. 在调试菜单，单击**停止调试**。
7. 在解决方案资源管理器窗格，展开 **Shared** 文件夹。
8. 在解决方案资源管理器窗格，展开 **Contoso.Events.ViewModels** 项目。
9. 双击 **HotelsViewModel.cs** 文件。
10. 定位到以下方法：

```
private static IEnumerable<Hotel> GetHotels()
```

11. 选中整个方法。
12. 右键单击选中的内容，单击**粘贴**来粘贴整个方法。

► 任务 5：调试云 Web 应用程序

1. 在解决方案资源管理器窗格，右键单击解决方案 **Contoso.Events**，然后单击**属性**。
2. 在解决方案 **Contoso.Events** 的属性页对话框中，执行以下步骤：
 - a. 在左边的导航栏，选中**通用属性=>启动项目**。
 - b. 选择**多启动项目**。
 - c. 对 **Contoso.Events.Cloud**，设置**操作为开始执行（不调试）**。
 - d. 对 **Contoso.Events.Lodging**，设置**操作为开始执行（不调试）**。
 - e. 对 **Contoso.Events.Management**，设置**操作为无**。
 - f. 确保所有其他项目的**操作设置为无**。
 - g. 单击**OK**。
3. 在 Start 屏幕，输入 **Azure Storage Emulator**。
4. 单击 **Windows Azure Storage Emulator–v3.4** 磁贴。
5. 切换到 **Contoso.Events–Microsoft Visual Studio** 窗口。

6. 在解决方案资源管理器窗格，右键单击 **Contoso.Events.Data.Generation** 项目，单击调试->启动新实例。
7. 等到调试进程结束，控制台窗口关闭。
8. 关闭 **Contoso.Events – Microsoft Visual Studio** 程序。
9. 在 Start 屏幕，找到 **Visual Studio 2013** 磁贴，右键单击，选择 **Run as Administrator**。

 **注释：**在 Start 屏幕，你可能需要单击向下箭头来找到 Visual Studio 2013 磁贴。在你用管理员启动了 Visual Studio 2013 之后，你可能会看到 UAC 弹出框，选择是继续。

10. 在文件菜单，单击**打开->项目/解决方案**。
11. 在**打开项目**对话框，执行以下步骤：
 - a. 定位到 **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events**。
 - b. 单击 **Contoso.Events** 解决方案文件。
 - c. 单击 **Open**。
12. 在**调试**菜单，单击**开始执行（不调试）**。
13. 你会看到一个弹出框，提示 WCF 服务没有元数据，询问是否退出调试。单击 **No**。
14. 最小化但是不要关闭 **WCF Test Client** 程序。
15. 在 Web 应用程序的主页上，单击任一 event。
16. 在 event details 页面，单击 **Register Now**。
17. 填写任意值，完成注册表单。
18. 单击 **Submit**。
19. **Registration Confirmation** 页面显示出旅馆列表。
20. 关闭 **Internet Explorer** 应用程序。
21. 关闭 **Microsoft Visual Studio** 应用程序。
22. 关闭 **Windows Azure Storage Emulator** 命令窗口。

结果：完成此练习后，您将使用 Service Bus 中继将现有 WCF 应用程序连接到了客户端。

问题：在项目发布后，有哪些方法可用来将 WCF 绑定从其企业内部版本自动转变为 Service Bus 中继绑定？

章节复习和作业

本章中，我们介绍了存储队列。我们还介绍了 Service Bus 命名空间以及可在云应用程序中使用的各种功能。最后，我们比较了两种队列机制。以下链接提供了有关两种队列服务的更多信息：



参考资料链接: <http://go.microsoft.com/fwlink/?LinkId=510217>

复习问题

问题: 哪种排队机制更适合存储大消息？

问题: 如果应用程序不能丢失任何消息，那么应该在使用方客户端中使用 PeekLock 模式还是 ReceiveAndDelete 模式？如何检查消息是否重复？

问题: 您有一个天气应用程序，您将使用通知中心来传递消息。您希望属于特定地区代码的人收到飓风紧急警报。应该如何设计应用程序架构来支持此情况？

MCT USE ONLY. STUDENT USE PROHIBITED

第 10 章 管理 Azure 中的基础结构

目录:

章节概述	10-2
第 1 课: Azure 虚拟网络	10-3
第 2 课: 高度可用的 Azure 虚拟机	10-5
第 3 课: 虚拟机配置管理	10-9
第 4 课: 自定义 Azure 虚拟机网络	10-12
实验 A: 管理虚拟网络中的多个虚拟机	10-15
实验 B: 使用中国版 Windows Azure 管理虚拟网络中的多个虚拟机	10-20
章节复习和作业	10-26

章节概述

创建虚拟机是您需要在 Microsoft Azure 的基础结构服务中使用的众多选项和功能之一。您可以部署虚拟机以及管理配置和网络。此外，您还可以自定义基础结构计算实例及其网络连接，以符合出站连接要求。第 1 节“Azure 虚拟网络”回顾 Azure 中提供的 Microsoft Azure 虚拟网络产品。第 2 节“高度可用的 Azure 虚拟机”探讨在针对高可用性场景设计虚拟机实例时，必须考虑的选项和功能。第 3 节“虚拟机配置与管理”描述管理和复制虚拟机配置的常用方法。第 4 节“自定义 Azure 虚拟机网络”回顾管理虚拟机入站和出站连接规则的选项。

目标

完成本章后，您可以做到：

- 创建虚拟网络。
- 描述允许匿名访问和私密访问虚拟机及其端口的选项。
- 自定义虚拟机的网络规则。

第 1 课 Azure 虚拟网络

Azure 中的虚拟网络提供了一种逻辑分组相关服务实例（如云服务和虚拟机）的方法。然后这些服务就可以私密通信，而无需创建可供公共访问的匿名端点。

本节描述 Azure 中的虚拟网络产品。

课程目标

完成本节后，您可以做到：

- 描述虚拟网络服务。
- 将服务实例添加到虚拟网络。

使用 XML 自定义虚拟网络配置

创建虚拟网络后，虚拟网络中的服务和虚拟机可安全地相互通信，而无需经过 Internet。由于只适用于云的虚拟网络并不用于跨企业内部连接，因此无需获得和配置虚拟专用网(VPN)设备或身份验证证书。

在创建网络期间，可以使用 Azure 管理门户或网络配置文件来配置 Azure 中的虚拟网络。网络配置文件由 Azure 用来定义虚拟网络的设置。

虚拟网络配置向导

在使用管理门户新建虚拟网络时，系统会显示一个向导，此向导可用来定义虚拟网络的设置。还可以使用此向导来配置站点到站点 VPN 或点到站点 VPN 的虚拟网络。还可以在此向导中配置地址空间和子网。创建虚拟网络后，可以将其配置导出到 XML 文件。

- 管理门户提供了便捷的向导，用于创建虚拟网络以及初始配置虚拟网络
- 也可使用网络配置文件 (netcfg) 定义虚拟网络的设置
- 可以编辑现有虚拟网络的网络配置文件
- MSDN 上提供了 netcfg 架构供参考

网络配置文件

网络配置文件 (netcfg) 可让您灵活自定义网络设置和模板网络。可以将网络配置文件导入现有虚拟网络来采用特定配置。可以从现有虚拟网络导出网络配置文件，以便于创建设置相似的其他虚拟网络。实际文件使用 .netcfg 扩展名，并采用 XML 格式。网络配置文件的架构可在 MSDN 上获得。可以使用此架构来创建自定义网络配置文件（使用文本编辑器创建）。

Azure 虚拟网络配置 XML 架构

<http://go.microsoft.com/fwlink/?LinkId=525663>

将服务部署到虚拟网络

可以将虚拟网络用作云服务或虚拟机的部署目标。部署云服务包时，可以为服务配置文件中定义的角色指定目标虚拟网络。创建虚拟机时，虚拟机与云服务关联。另外，在创建虚拟机时，可以像在云服务中那样指定虚拟网络。不可以将云服务或虚拟机从一个虚拟网络迁移到另一个虚拟网络，在新网络中必须重新创建它们。在将虚拟机部署到网络时，它们会得到为该网络指定的设置。网络安全组等其他设置应用在子网级别。

可以使用服务配置文件的网络配置元素，将云服务部署到以前在网络配置文件中定义的虚拟网络。这些设置对云服务来说是可选的。如果不在虚拟网络元素部分指定设置，云服务将不会部署到虚拟网络。

云服务 NetworkConfiguration XML 架构

<http://go.microsoft.com/fwlink/?LinkId=525664>

可在服务配置文件中指定的部分网络配置设置有：

- 可用于名称解析的 DNS 服务器
- 虚拟网络站点名
- 云服务的地址空间和子网

- 计算实例可部署到现有虚拟网络
 - 云服务
 - 在服务配置文件中指定虚拟网络
 - 虚拟机
 - 创建时指定虚拟网络
- 计算实例将使用虚拟网络配置文件中指定的设置

第 2 课 高度可用的 Azure 虚拟机

可以使用虚拟机横向或纵向缩放以及负载平衡来应对高可用性场景。可用性集还影响虚拟机在故障期间或升级期间的可用性。

本节描述在针对高可用性场景设计虚拟机集合时要考虑的注意事项。

课程目标

完成本节后，您可以做到：

- 横向或纵向缩放虚拟机实例。
- 描述故障域和升级域对虚拟机可用性有何影响。

可用性集

可用性集提供了一种机制来指示 Azure 将虚拟机置入单独的故障域或更新域。当虚拟机因为故障（或正常维护）而停止工作时，至少有一个负载平衡的虚拟机实例仍然可用。

维护活动

在 Azure 平台中，可能随时发生计划内维护。通常，这些更新由 Microsoft 执行，以启用新功能或提高可靠性和性能。此外，这些计划内维护活动的绝大部分都已事先公布，对正在运行的虚拟机或云服务没有影响。但是偶尔，计划内维护活动需要重新启动虚拟机或云服务。

- 可用性集提供了一种机制来指示 Azure 将虚拟机放入单独的故障域或更新域
- 当虚拟机因为故障（或正常维护）而停机时，至少有一个负载平衡的虚拟机实例仍然可用
- 为了符合服务级别协议 (SLA)，需要多个虚拟机实例

在出现影响运行中虚拟机实例的物理故障或硬件故障时，可能发生计划外维护活动。虽然 Azure 平台自动迁移实例，但是在发生物理故障和安排恢复工作之间，仍可能会有停机时间。

可用性集

可以使用可用性集分组两个或更多虚拟机，在一定程度上为应用程序提供冗余性。可用性集确保在计划内或计划外维护活动期间，至少有一个虚拟机仍然可用。通常，与应用程序同属一层的虚拟机将归入同一个可用性集。

可用性集可用来将虚拟机分组成应用程序层

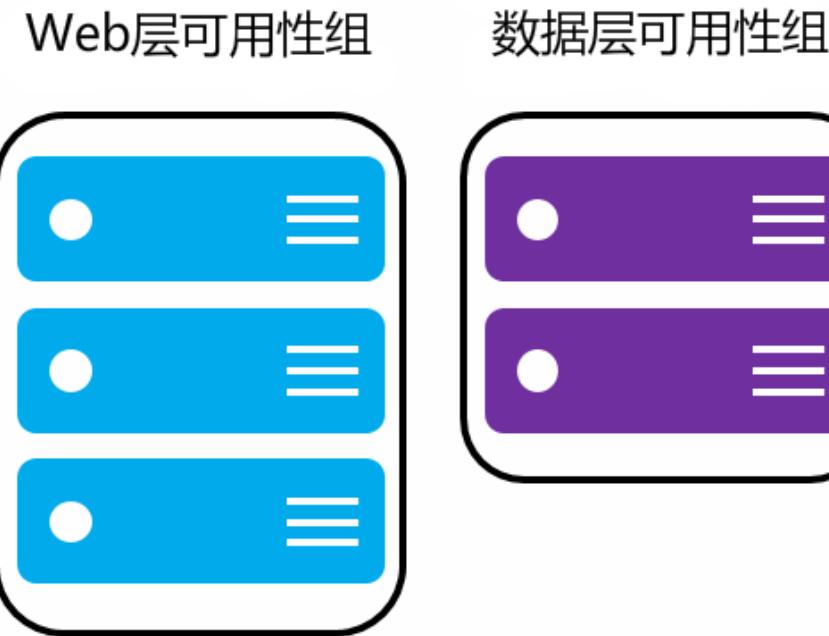


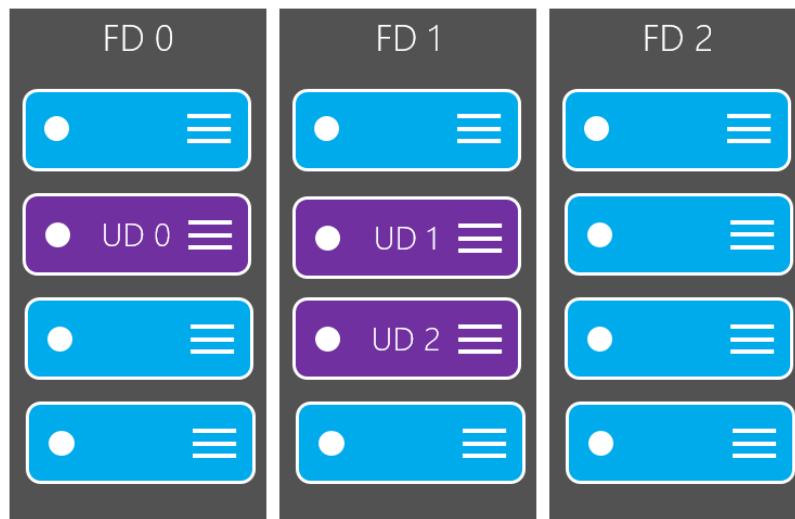
图 10.1：可用性集实例

升级域和故障域

可用性集中的虚拟机由 Azure 平台归入更新域和故障域。更新域是将多个虚拟机组合在一起的逻辑单元。更新域中的服务实例通常一起升级。这对于云服务角色尤其常见，因为它们都由 Azure 平台更新和管理。故障域是指物理故障点。例如，数据中心中的物理机架就是故障域。理想的做法是，将服务分组成多个故障域和更新域，使得一个小物理故障（如硬盘驱动器故障）或升级不会对所有应用程序实例同时造成不利影响。

在可用性集中，默认分配了五个更新域。需要六个或更多虚拟机实例，才会在同一更新域中出现多个实例。在升级或重启操作期间，一次只能重启一个更新域。这确保在整个操作期间，应用程序有一致的可用性。在同一可用性集中，默认分配了两个故障域。这确保多个虚拟机实例处于不同单独的位置，并且不会受到同一个物理故障的影响，如硬件故障、网络中断或断电。

Azure 平台中的故障域和更新域。



MCT USE ONLY STUDENT USE PROHIBITED

图 10.2：故障域/更新域

服务级别协议

在可用性集中，必须避免只留下一个单独的虚拟机实例。可用性集中只有一个虚拟机不符合服务级别协议 (SLA) 承诺，将在 Azure 的计划内维护活动期间造成停机时间。更改这单个虚拟机的特定属性也会造成应用程序停机时间。而且，如果在可用性集中只部署一个虚拟机实例，那么在平台维护期间，您将收不到提前警告或通知。按照这种配置，当发生平台维护时，单个虚拟机实例会被重新启动，而没有事先警告。

缩放虚拟机

纵向缩放

可以更改虚拟机大小来纵向缩放虚拟机实例。如果有应用程序在虚拟机中运行，并且需要更多资源，那么可以增加虚拟机大小来处理。这可能是添加更多内存、磁盘或 CPU 内核。

横向缩放

如果可用性集中有多个虚拟机实例，可以利用 Azure 门户中提供的缩放功能。可以手动更改实例数量来扩展或缩减应用程序。

在横向缩放在可用性集中的虚拟机上运行的应用程序时，可以不创建新的虚拟机或删除现有虚拟机。而是开启或关闭可用性集中先前创建的任何虚拟机。可以根据 CPU 平均使用百分比或队列中的消息数来指定缩放。

手动更改实例数量

- 虚拟机通常可从两个方向缩放
 - 横向
 - 添加重复的虚拟机实例
 - 纵向
 - 可更改当前实例的虚拟机层
- 虚拟机可自动横向缩放

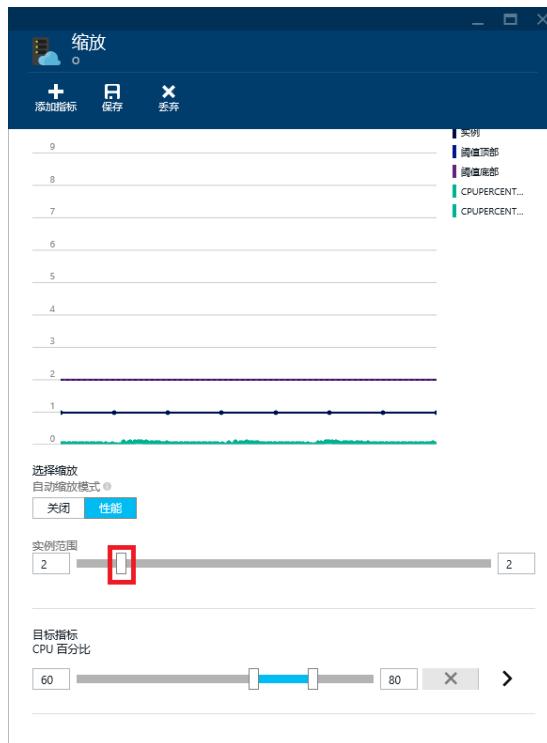


图 10.3：实例数量

按指标自动缩放

在“缩放”页上，可以将云服务配置为自动增加或减少应用程序所用的虚拟机实例的数量。可以基于以下参数配置缩放：

- **平均 CPU 使用率。**如果平均 CPU 使用百分比高于或低于指定的阈值，Azure 将相应创建或删除角色实例，或者开启或关闭可用性集中的虚拟机。
- **队列消息数。**如果队列中的消息数量高于或低于指定的阈值，Azure 将相应创建或删除角色实例，或者开启或关闭可用性集中的虚拟机。

在定义自动缩放规则时，必须指定以下值：

- **计划。**定义规则应生效的时间和日期。
- **实例范围。**要缩放的实例的最大和最小数量。
- **指标**
 - **目标 CPU。**应用程序虚拟机的理想 CPU 范围。当平均 CPU 使用百分比高于该范围时，将发生放大，当平均 CPU 使用百分比低于该范围时，将发生缩小。
 - **每台计算机的队列目标。**每个虚拟机实例队列消息的目标数量。缩放是根据队列消息的总数量除以每个实例队列消息的理想数量来确定的。
- **扩展/缩减幅度。**启动或停止每个缩放操作的实例数量。
- **扩展/缩减等待时间。**执行另一个缩放操作之前要等待的最长时间量。

第 3 课 虚拟机配置管理

虽然可以手动配置每个虚拟机，但是虚拟机配置自动化可产生可重复、高效且可测试的部署场景。自动化配置还可以确保新缩放的虚拟机实例与其他实例匹配。

本节讨论某些最常见的配置管理方法。

课程目标

完成本节后，您可以做到：

- 说明 Windows PowerShell Desired State Configuration (DSC) 如何可用于虚拟机配置管理。
- 描述 Azure 虚拟机中的虚拟机代理服务。
- 描述配置管理工具。

Windows PowerShell Desired State Configuration

Desired State Configuration (DSC) 提供了一组 Windows PowerShell 语言扩展、新的 Windows PowerShell cmdlet 和资源，可用来以声明方式指定软件环境应配置成什么样子。它还提供了维护和管理现有配置的方法。

DSC 引入了一个新的关键字，称为 **Configuration**。
若要使用 DSC 来配置环境，应先使用 **Configuration** 关键字定义 Windows PowerShell 脚本块，后跟一个标识符，然后用大括号 {} 来界定脚本块。

PowerShell DSC 配置

```
Configuration WebServerConfig
{
    Node "WebServer"
    {
        WindowsFeature ServerRoleExample
        {
            Ensure = "Present"
            Name = "Web-Server"
        }
    }
}
```

Desired State Configuration (DSC)

- 是 PowerShell 的扩展
 - 新的语言功能
 - 新的 cmdlet
 - 额外资源
- 主要用于软件环境的配置
- 可用于维护现有配置或管理新配置

现在定义了示例配置。接下来，需要使此配置生效。可以调用该配置来令其生效。

调用 PowerShell DSC 配置

```
PS C:\Scripts> WebServerConfig
```

调用该配置将创建托管对象格式 (MOF) 文件，并将其放入与配置块同名的新目录中。新的 MOF 文件包含目标节点的配置信息。

为了使保存的配置生效，请运行下面的命令。

启用配置

```
Start-DscConfiguration -Wait -Verbose -Path .\WebServerConfig
```

DSC 包含其他扩展性功能，如参数和嵌套配置。

VM 代理

虚拟机代理是一种通常安装在 Azure 虚拟机上的轻量级服务。此代理提供可安装扩展的扩展点。虚拟机扩展是创建的自定义扩展，可安装在装有虚拟机代理的 Azure 虚拟机上。使用虚拟机扩展可以执行以下任务：

- 在虚拟机中自动安装自定义或现成的软件组件。
- 安装、更新或移除自定义功能，而无需重新创建或更新现有虚拟机。
- 从一个集中式工具或位置管理和查看多个虚拟机的状态或指标。

- VM 代理是非常轻型的后台进程，它为 Microsoft 和合作伙伴提供了配置和管理虚拟机的入口点
 - 默认安装在虚拟机上（但是可以禁用）
 - 允许 VM 扩展安装到 Azure 虚拟机
- VM 扩展是可扩展现有虚拟机的软件组件
 - 同一虚拟机上可安装多个 VM 扩展
 - BGInfo 桌面工具是一个 VM 扩展

配置管理工具

配置管理是在大量的各种计算机上实施、跟踪和控制软件更改的任务。很多源自源代码控制管理（如修订版控制）的常见做法也出现在配置管理 (CM) 软件中。配置管理实用工具提供了很多功能，如：

- **基线。**建立基本配置，用作新的虚拟机或物理机的默认配置。
- **修订历史记录。**可让团队确定谁或者什么程序可以更改特定配置设置。
- **复制。**实现在多个计算机上复制配置更改。
- **代理。**一种软件，它们专门安装在计算机上，用来接收配置更改请求，并将其应用到本地计算机。

- 配置管理是在不同物理机或虚拟机之间保持一致性的过程
- 有两个最受欢迎的配置管理实用工具可用于 Microsoft Azure 虚拟机
 - Puppet
 - Chef

本主题中讨论的配置管理实用工具支持 Windows 和 Linux 操作系统。

在整个行业都使用的配置管理软件中，Chef 和 Puppet 是两个最常见的例子。Chef 和 Puppet 都是用 Ruby 编写的，并按照 Apache 许可证授予许可。Azure 中的 Chef 和 Puppet 提供了模板映像。

Puppet

Puppet 是由 Puppet Labs 制作的开源配置管理实用工具。Puppet 有独特的声明式语言，可用于描述系统配置。这些配置更改可直接应用于虚拟机，或者使用目录分发到多个虚拟机。Puppet 代理定期轮询虚拟机来获得其当前配置，然后将这些配置数据同步到 Puppet 主控机，主控机管理安装了代理的所有其他虚拟机。Puppet 主控机确保装有代理的虚拟机符合目录中定义的最新配置。



Puppet Labs

<http://go.microsoft.com/fwlink/?LinkId=525665>

Chef

Chef 是另一个很受欢迎的开源配置管理实用工具。Chef 之所以独特是因为配置更改组合成食谱 (recipe)。食谱由称为资源的单个配置更改组成，可以包括：

- 要存储的文件
- 配置更改模板
- 要安装的软件包

食谱可以组合并用于自动化最常见的基础结构任务以及软件配置更改。使用称为节点的代理，Chef 服务器受到轮询，提供已安装食谱的更改，并确保各个计算机（物理机或虚拟机）符合最新版本的食谱。



<http://go.microsoft.com/fwlink/?LinkId=525666>

MCT USE ONLY. STUDENT USE PROHIBITED

第 4 课 自定义 Azure 虚拟机网络

虽然可以在创建 Azure 虚拟机后直接使用它们，但是必须先执行额外的配置，然后才能让这些虚拟机实例与外部资源或其他虚拟机对接。

本节描述用于自定义 Azure 虚拟机网络连接的方法。

课程目标

完成本节后，您可以做到：

- 使用自定义端点公开虚拟机的公共端口。
- 自定义虚拟机或云服务的访问控制列表。
- 修改虚拟机中的 Windows 防火墙。
- 查看虚拟机的公共虚拟 IP 地址 (VIP)。

自定义端点

在 Azure 创建的所有虚拟机可使用专用网络通道，自动与同一个云服务或虚拟网络中的其他虚拟机通信。但是，Internet 或其他虚拟网络上的其他资源需要端点来处理传至虚拟机的入站网络流量。

在管理门户中创建虚拟机时，可以创建新端点。例如 Remote Desktop、Remote PowerShell 和 Secure Shell (SSH) 通常是在门户中创建新虚拟机实例的对话框中创建的。创建虚拟机后，可以根据需要创建更多端点。还可以配置端点的网络访问控制列表 (ACL) 的规则来管理发至公共端口的传入流量。本主题解释如何完成这两个任务。

- Azure 中的虚拟机可在其专用网络中通信
- 若要与外部资源通信，需要为虚拟机创建端点
- 端点包括：
 - 负载平衡器用于与外部资源通信的公共端口
 - 虚拟机侦听流量时所用的专用端口
- 端点也可在多个虚拟机之间负载平衡

每个端点都有公共端口和专用端口：

- 专用端口由虚拟机内部用于侦听该端点上的流量。
- 公共端口由 Azure 负载平衡器用来从外部资源与虚拟机通信。创建端点后，可以使用网络访问控制列表 (ACL) 来定义规则，这些规则将帮助隔离和控制公共端口上的传入流量。有关更多信息。

 **注释：**（可选）还可以使用网络安全组来定义访问虚拟网络子网中的 VM 和云服务的规则。

这些端点的端口和协议的默认值是在通过管理门户创建端点时提供的。对于所有其他端点，在创建端点时指定端口和协议。资源可使用传输控制协议 (TCP) 或用户数据报协议 (UDP) 连接到端点。TCP 协议包括 HTTP 和 HTTPS 通信。

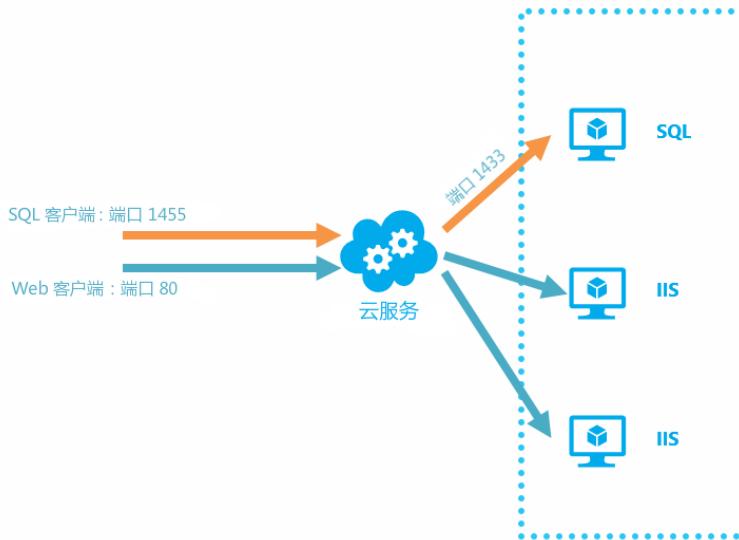


图 10.4: <ADD IMAGE CAPTION HERE>
<ADD IMAGE CAPTION HERE>

访问控制列表

虚拟机端点默认允许任何外部客户端使用指定的公共端口连接到云服务。对于公开的 Web 应用程序，如果要供任何公共客户端访问，可以采用端口 80。对于任何其他敏感的应用程序，这种做法不太适合。ACL 和网络安全组允许您筛选可访问云服务中的虚拟机的客户端范围。

访问控制列表

ACL 可让您筛选虚拟机端点的流量。通过 ACL，可以使用一系列允许或拒绝规则以及 IP 地址范围，有选择地允许或拒绝流量。对于复杂的筛选场景，可以使用规则排序来混合 ACL 规则。

网络安全组

网络安全组 (NSG) 与 ACL 很像。主要区别是，NSG 是为虚拟网络中的子网定义的。这样一来，NSG 可隐式应用于虚拟网络子网中的所有计算实例。添加到子网的新虚拟机将自动受到 NSG 应用的筛选。

- 访问控制列表 (ACL) 用于更粒度化地控制对特定端点的访问
- 默认情况下，虚拟机配有 ACL 拒绝所有传入流量
- 每当端点添加到虚拟机时，ACL 将修改，以允许该端点的所有传入流量
- 使用“**允许**”可将地址范围列入白名单，也可使用“**拒绝**”列入黑名单

MCT USE ONLY. STUDENT USE PROHIBITED

防火墙规则

Windows 防火墙是 Windows 的客户端版本和 Windows Server 自带的基于主机的内置有状态防火墙。如果传入流量不是用于响应计算机请求而发送的流量（受到请求的出站流量），也不是已指定为允许的未经请求的流量（例外的入站流量），Windows 防火墙将丢弃这些流量。Windows 防火墙是使用“高级安全 Windows 防火墙”管理单元配置的，该管理单元将防火墙行为和流量保护的规则与 Internet 协议安全 (IPsec) 集成在一起。

迁移到 Azure 的虚拟机在迁移前应该先验证其 Windows 防火墙规则。有可能在虚拟机迁移到 Azure 后，因为 Windows 防火墙规则的关系，连接虚拟机执行管理任务时可能会遇到问题。

- 和很多企业内部 Windows 计算机一样，Azure 中的虚拟机会有运行着的 Windows 防火墙实例。
- 添加端点是允许访问虚拟机实例端口的第一步：



实验 A：管理虚拟网络中的多个虚拟机

场景

为了最好地复制现有生产系统，您决定创建虚拟网络，并在虚拟网络中托管另一个虚拟机。新的虚拟机将包含将用于测试应用程序的 SQL 数据库。您将在同一个虚拟网络的 Azure 网站服务中托管测试应用程序。

 **注释：**本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版 Windows Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 创建新网站服务。
- 使用标准映像创建虚拟机。

实验设置

预计时间：45 分钟

在开始这个实验之前，必须完成第 2 章的实验 A。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，输入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].cloudapp.net: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动输入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为**Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**，等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验 A 中使用这些密码和 Azure 帐户。

本章为您提供了两套试验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版 Windows Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

练习 1：配置现有虚拟网络

此练习的主要任务如下：

1. 登录到 Azure 管理门户

MCT USE ONLY. STUDENT USE PROHIBITED

2. 配置点到站点连接

3. 创建网关

► 任务 1：登录到 Azure 管理门户

1. 登录 Azure 管理门户(<https://manage.windowsazure.com>)。

► 任务 2：配置点到站点连接

1. 找到虚拟网络 **Dev28532**。

2. 根据以下项配置点到站点连接：

- Address Space: **172.16.0.0/29**

- Gateway Subnet: **10.0.1.0/29**



注释：更新虚拟网络配置可能需要大约 15 分钟。间歇性的刷新浏览器来检查更新状态。

► 任务 3：创建网关

1. 为虚拟网络实例创建网关。



注释：网关的创建需要大约 15 分钟. 间歇性的刷新浏览器来检查新网关的状态。当仪表盘显示出了**网关 IP 地址**, 网关就被创建完成了。等待网关创建的同时，也可以选择继续实验中的剩余部分。

练习 2：创建数据库虚拟机

场景

对于测试应用程序，您需要一个装有 SQL Server 2014 的新虚拟机。您还需要配置此虚拟机，使得其他虚拟机可在同一网络访问到它。

此练习的主要任务如下：

1. 登录到 Azure 预览门户
2. 创建 SQL Server 2014 Standard 虚拟机
3. 连接到新的数据库虚拟机
4. 为 SQL Server 添加一条规则到 Windows 防火墙
5. 启用 SQL Server 混合模式身份验证

► 任务 1：登录到 Azure 预览门户

1. 登录 Azure 预览门户(<https://portal.azure.com>)。

► 任务 2：创建 SQL Server 2014 Standard 虚拟机

1. 根据以下详细信息创建一个 SQL Server 2014 标准虚拟机：

- 主机名: **db28532[自定义名字]**
- 用户名: **testuser**
- 密码: **TestPa\$\$w0rd**
- 定价层: **A2 Standard**

- 虚拟网络: **Dev28532**
 - 存储帐户: **stor28532[自定义名字]**
2. 记录下 SQL Server 虚拟机的**虚拟 IP 地址**。

► **任务 3: 连接到新的数据库虚拟机**

1. 使用远程桌面, 连接新创建的虚拟机。

► **任务 4: 为 SQL Server 添加一条规则到 Windows 防火墙**

1. 打开 Windows Firewall with Advanced Security。
2. 使用以下详细信息创建一个新的入站规则:
 - Rule Type: **Port**
 - Port Type: **TCP**
 - Local Ports: **1433**
 - Action: **Allow the connection**
 - Apply for Domain: **Yes**
 - Apply for Private: **Yes**
 - Apply for Public: **Yes**

► **任务 5: 启用 SQL Server 混合模式身份验证**

1. 打开 SQL Server 2014 Management Studio。
2. 使用以下详细信息来连接你本地的 **SQL Server** 实例:
 - Server type: **Database Engine**
 - Server name: **。(句号)**
 - Authentication: **Windows Authentication**
3. 在 SQL Server 的设置中启用 **SQL Server and Windows Authentication Mode**。
4. 重启 **SQL Server** 实例。
5. 使用以下详细信息来添加一个新的 SQL Server 实例登录方案:
 - Login name: **dbuser**
 - Login type: **SQL Server authentication**
 - Password: **TestPa\$\$word**
 - Enforce password policy: **No**
 - Enforce password expiration: **No**
 - User must change password at next login: **No**
 - Server Roles: **public, sysadmin**
6. 创建一个命名为 **Contoso.Test** 的新数据库。
7. 关闭 **Remote Desktop** 应用程序。

结果: 完成此练习后, 您将获得一个装有 SQL Server 2014 的新虚拟机, 在由外部虚拟机访问时, 该 SQL Server 可使用混合模式身份验证。

MCT USE ONLY. STUDENT USE PROHIBITED

练习 3：创建 Azure 网站服务实例

场景

对于测试应用程序，您需要一个新网站服务部署到与以前创建的虚拟机相同的虚拟网络。

此练习的主要任务如下：

1. 创建网站服务实例
2. 部署 Contoso.Events 数据库测试 Web 应用程序

► 任务 1：创建网站服务实例

1. 使用以下详细信息创建网站：
 - Url：为网站挑选一个独有的名字。
 - Web Hosting Plan Tier： **S1 Standard**
 - Web Hosting Plan Name： **28532**
2. 查看新网站的边栏选项卡。
3. 将网站加入到 **Dev28532** 的虚拟网络中。
4. 记录下新建网站实例的 URL。
5. 下载网站的发布配置文件到指定路径：
 - 保存路径： **Allfiles (F):\Mod10\Labfiles**

► 任务 2：部署 Contoso.Events 数据库测试 Web 应用程序

1. 从以下路径打开解决方案 **Contoso.Events**：
 - 文件路径： **Allfiles (F):\Mod10\Labfiles\Starter\Contoso.Events**
2. 切换到 Visual Studio 2013。
3. 使用之前的发布配置文件发布这个 ASP.NET Web 应用程序。

结果：完成此练习后，您将在虚拟网络中创建好一个网站服务，并将 Web 应用程序部署到该网站。

练习 4：将测试应用程序连接到 SQL Server 虚拟机

场景

您有了一个 ASP.NET 测试应用程序，可验证自己是否可以访问 SQL Server 实例。您想在开发虚拟机中调试此应用程序，以验证自己在专用网络中是否可以访问 SQL Server 实例。

此练习的主要任务如下：

1. 检索 SQL Server 虚拟机的内部 IP 地址。
2. 在本地调试 Contoso.Events 数据库测试 Web 应用程序。
3. 在 Azure 中调试 Contoso.Events 数据库测试 Web 应用程序。

► 任务 1：检索 SQL Server 虚拟机的内部 IP 地址

1. 切换到 Internet Explorer。
2. 查看您订阅的虚拟机实例列表。
3. 查看虚拟机 **db28532[自定义名字]** 的边栏选项卡。

4. 记录下虚拟机实例的**专用 IP 地址**。

► **任务 2：在本地调试 Contoso.Events 数据库测试 Web 应用程序**

1. 调试 Web 应用程序，然后在主页上的**IP 地址**框内，输入内部运行 SQL Server 的虚拟机的**专用 IP 地址**，验证能否在你的虚拟机连接 SQL Server 虚拟机。

 **注释：**即使你可以从虚拟网络成功地连接到您的虚拟机，这并不意味虚拟机也可以用相同端口从其他外部网络访问。

2. 关闭 Internet Explorer。

► **任务 3：在 Azure 中调试 Contoso.Events 数据库测试 Web 应用程序**

1. 导航到你先前在这个实验中创建的 Azure 网站实例的 URL。
2. 在主页的**IP 地址**框内，输入运行 SQL Server 虚拟机的**专用地址**来验证是否能利用的虚拟网络来连接虚拟机。

结果：完成此练习后，您将使用虚拟网络中数据库虚拟机的内部 IP 地址连接到了 SQL Server 2014。

通过在右列中放置标记来判断叙述的正确性。

叙述	答案
可以使用 SQL Server Management Studio 从任何计算机，按其当前配置连接到 SQL Server VM。	

实验 B：使用中国版 Windows Azure 管理虚拟网络中的多个虚拟机

场景

为了最好地复制现有生产系统，您决定创建虚拟网络，并在虚拟网络中托管另一个虚拟机。新的虚拟机将包含将用于测试应用程序的 SQL 数据库。您将在同一个虚拟网络的 Azure 网站服务中托管测试应用程序。

目标

完成本实验后，您将能够：

- 创建新网站服务。
- 使用标准映像创建虚拟机。

预计时间：45 分钟

在开始这个实验之前，必须完成第 2 章的**实验 B**。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，输入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].chinacloudapp.cn: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能被保存在计算机下拉列表中。如果这样，就无需手动输入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - a. 用户名：**Student**
 - b. 密码：**AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到 Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：配置现有虚拟网络

此练习的主要任务如下：

1. 登录到 Azure 管理门户
2. 配置点到站点的连接
3. 创建网关

► 任务 1：登录到 Azure 管理门户

1. 在 Start 屏幕，单击**Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。

MCT USE ONLY. STUDENT USE PROHIBITED

3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的**地球图标**，并在菜单中选择**中文(简体)**语言。

► 任务 2：配置点到站点的连接

1. 在页面左侧的导航栏中，单击**网络**。
2. 在**网络**列表中，单击名称为 **Dev28532** 的虚拟网络。
3. 单击**配置**选项卡。
4. 在**点到站点连接**下，选择**配置点到站点连接**。
5. 在**地址空间**下配置以下值：
 - a. 在**起始 IP** 中输入 **172.16.0.0**。
 - b. 在**CIDR (地址数)** 框，选择**/29 (6)**。
6. 单击**虚拟网络访问空间**下的**添加网关子网**。
7. 在页面的底部单击**保存**。
8. 此时出现一个确认对话框，警告可能出现连接中断。单击**是**按钮继续。

 **注释：**更新虚拟网络配置需要大约 15 分钟。周期性的刷新浏览器来检查更新状态。

► 任务 3：创建网关

1. 单击**仪表板**选项卡。
2. 单击页面底部的**创建网关**。
3. 一个确认对话框将会确认是否要创建虚拟网络的网关，单击按钮**是**。

 **注释：**网关的创建需要大约 15 分钟。周期性的刷新浏览器来检查新网关的状态。当仪表板显示出了**网关 IP 地址**，网关就被创建完成了。等待网关创建的同时，也可以选择试验中的剩余部分进行练习。

练习 2：创建数据库虚拟机

场景

对于测试应用程序，您需要一个装有 SQL Server 2014 的新虚拟机。您还需要配置此虚拟机，使得其他虚拟机可在同一网络访问到它。

此练习的主要任务如下：

1. 创建 SQL Server 2014 Standard 虚拟机
2. 连接到新的数据库虚拟机
3. 为 SQL Server 添加一条规则到 Windows 防火墙
4. 启用 SQL Server 混合模式身份验证

► 任务 1：创建 SQL Server 2014 Standard 虚拟机

1. 在屏幕的左下角，单击**新建**。
2. 在**新建**边栏选项卡，依次单击**计算-> 虚拟机-> 快速创建**。

3. 在 **DNS 名称** 中填入 **db28532[自定义名称]**。
4. 单击映像下拉列表，单击**更多映像**。
5. 显示出的**创建虚拟机**导航选项卡，完成以下步骤：
 - a. 单击 **SQL Server**。
 - b. 在中间选择 **SQL Server 2014 RTM Standard**。（系统为 **Windows Server 2012 R2**）
 - c. 单击**下一步**。
 - d. 在**虚拟机名称**中输入 **db28532[自定义名称]**。
 - e. 在**大小**下拉框中，选择 **A2(双核, 3.5GB 内存)**。
 - f. 在**新用户名**框内输入 **testuser**。
 - g. 在**新密码**框和**确认**框内，输入 **TestPa\$\$w0rd**。
 - h. 单击**下一步**。
 - i. 在**云服务**中选择**创建新云服务**。
 - j. 在**云服务 DNS 名称**，验证名称可用，如果不可请修改为一个可用的名称。
 - k. 在**区域/地缘组/虚拟网络**中选择 **Dev28532**。
 - l. 存储帐户选择**使用自动生成的帐户**。
 - m. 单击**下一步**。
 - n. 单击**完成**。

 **注释：**Azure 试用帐户有资源内核数限制，如果遇到内核不足导致虚拟机创建失败，请在步骤 e 中将虚拟机的大小修改为 **A1(单核, 1.75GB 内存)**或 **A0(共享内核, 768MB 内存)**。

6. 等待虚拟机及扩展创建完成。
如果你想通过公网访问到数据库服务器完成步骤 7~13。
7. 在管理网站中找到刚创建的虚拟机，单击该虚拟机的名字，单击仪表板，找到右侧公用虚拟 **IP(vip) 地址**并记录它。
8. 单击**端点**。
9. 单击页面下方的**添加**。
10. 在弹出的**添加端点**页面中单击**下一步**。
11. 在**名称**选择 **MSSQL**。
12. 确认协议为 **TCP**，公用端口和私有端口为 **1433**。
13. 单击**完成**。

► 任务 2：连接到新的数据库虚拟机

1. 在管理网站上找到虚拟机 **db28532[自定义名称]**。
2. 单击选择该虚拟机。
3. 在页面底部单击**连接**。
4. 在 **Internet Explorer 下载**对话框里，单击 **Open**。
5. 在 **Remote Desktop Connection** 对话框里，完成以下步骤：

MCT USE ONLY. STUDENT USE PROHIBITED

- a. 为了不让这个对话框再次出现，单击 **Don't ask me again for connections to this computer**。
- b. 单击 **Connect**。
6. 在 **Windows Security** 对话框按以下步骤操作：
 - a. 在 **User name** 框内，输入 **testuser**。
 - b. 在 **Password** 框内，输入 **TestPa\$\$w0rd**。
 - c. 单击 **OK**。
7. 在 **Remote Desktop Connection** 对话框内，完成以下步骤：
 - a. 验证 **Certificate name** 与你虚拟机的名字相同。
 - b. 为了防止这个对话框再次出现，单击 **Don't ask me again for connections to this computer**。
 - c. 单击 **Yes**。
8. 当你被提议允许你的网络连接用于发现外部设备时，单击 **No**。

► 任务 3：为 SQL Server 添加一条规则到 Windows 防火墙

1. 在 Start 屏幕，单击左下角的向下箭头。
2. 找到并单击应用程序 **Run**。
3. 在 Run 对话框里完成以下步骤：
 - a. 在 **Open** 框内，输入 **WF.msc**。
 - b. 单击 **OK** 来打开 **Windows Firewall**。
4. 在 **Windows Firewall with Advanced Security** 窗口里完成以下步骤：
 - a. 单击选中 **Inbound Rules**，再右键单击 **Inbound Rules**，然后单击 **New Rule**。
 - b. 选择 **Port**。
 - c. 单击 **Next**。
 - d. 选择 **TCP**。
 - e. 选择 **Specific local ports**。
 - f. 在 **Specific local ports** 框内，输入 **1433**。
 - g. 单击 **Next**。
 - h. 选择 **Allow the connection**。
 - i. 单击 **Next**。
 - j. 确保 **Domain**, **Private** 和 **Public** 复选框都被选中了。
 - k. 单击 **Next**。
 - l. 在 **Name** 框内输入 **SQL Inbound**。
 - m. 单击 **Finish**。
5. 关闭 **Windows Firewall with Advanced Security** 窗口。

► 任务 4：启用 SQL Server 混合模式身份验证

1. 在 Start 屏幕，单击左下角的向下箭头。
2. 找到并单击 **SQL Server 2014 Management Studio** 磁贴。
3. 在 **Connect to Server** 对话框内，完成以下步骤：

- a. 在 **Server name** 框内，输入.（输入一个点）。
- b. 单击 **Connect**。
4. 右键单击 **Object Explorer** 栏顶端的 **SQL Server** 节点，然后单击 **Properties**。
5. 在 **Server Properties** 对话框里完成以下步骤：
 - a. 单击 **Security**。
 - b. 在 **Server authentication** 节内，选择 **SQL Server and Windows Authentication mode**。
 - c. 单击 **OK**。
 - d. 在 **Microsoft SQL Server Management Studio** 对话框内，单击 **OK**。
6. 右键单击 **Object Explorer** 顶部的 **SQL Server** 节点然后单击 **Restart**。
7. 在 **Microsoft SQL Server Management Studio** 对话框内，单击 **Yes**。
8. 右键单击 **Object Explorer** 栏里的 **Security** 节点，指向 **New**，然后单击 **Login**。
9. 在 **Login-New** 对话框里完成以下步骤：
 - a. 在 **Login name** 框内，输入 **dbuser**。
 - b. 单击 **SQL Server authentication**。
 - c. 在 **Password** 框内，输入 **TestPa\$\$w0rd**。
 - d. 在 **Confirm Password** 框内，输入 **TestPa\$\$w0rd**。
 - e. 确保 **Enforce password policy** 复选框没有被选中。
 - f. 确保 **Enforce password expiration** 复选框没有被选中。
 - g. 确保 **User must change password at next login** 复选框没有被选中。
 - h. 单击 **Server Roles** 页面。
 - i. 确保 **public** 服务器角色复选框被选中。
 - j. 确保 **sysadmin** 服务器角色复选框被选中。
 - k. 单击 **OK** 来创建新的登录方案。
10. 右键单击 **Object Explorer** 顶端的 **Databases** 节点，然后单击 **New Database**。
11. 在 **New Database** 对话框内完成以下步骤：
 - a. 在 **Database name** 框内，输入 **Contoso.Test**。
 - b. 单击 **OK** 来创建新的数据库。
12. 关闭 **Microsoft SQL Server Management Studio** 窗口。
13. 关闭 **Remote Desktop Connection** 应用。

结果：完成此练习后，您将获得一个装有 SQL Server 2014 的新虚拟机，在由外部虚拟机访问时，该 SQL Server 可使用混合模式身份验证。

练习 3：将测试应用程序连接到 SQL Server 虚拟机

场景

您有了一个 ASP.NET 测试应用程序，可验证自己是否可以访问 SQL Server 实例。您想在开发虚拟机中调试此应用程序，以验证自己在专用网络中是否可以访问 SQL Server 实例。

MCT USE ONLY. STUDENT USE PROHIBITED

此练习的主要任务如下：

1. 检索 SQL Server 虚拟机的内部 IP 地址
2. 在本地调试 Contoso.Events 数据库测试 Web 应用程序

► 任务 1：检索 SQL Server 虚拟机的内部 IP 地址

1. 切换到 Internet Explorer。
2. 在管理网站左边，单击**虚拟机**。
3. 在出现的**虚拟机**列表中，单击名为 **db28532** 的虚拟机。
4. 单击**仪表板**。
5. 在右侧找到**内部 IP 地址**并记录下来。

► 任务 2：在本地调试 Contoso.Events 数据库测试 Web 应用程序

1. 在 Start 屏幕，单击 **Visual Studio 2013** 磁贴。
2. 选择文件->打开->项目/解决方案。
3. 在打开项目对话框中，定位到 **F:\Mod10\Labfiles\Starter\Contoso.Events**，选择 **Contoso.Events.sln**，单击 **Open**。
4. 在**解决方案资源管理器**窗格中，右键单击项目 **Contoso.Events.Web**，然后单击**设为启动项目**。
5. 在**调试**菜单中，单击**启动调试**。

 **注释：**此网站应用程序引用了多个 **NuGet** 包。调试解决方案时，Visual Studio 开始构建解决方案。这触发了 **NuGet** 自动下载安装缺失包的操作。

 **NuGet Package Restore**

<http://go.microsoft.com/fwlink/?LinkID=510175>

6. 在网页应用的主页里，**IP Address** 框里，输入之前记录下虚拟机的内部 IP 地址。
7. 单击 **Verify**。
8. 验证 **Events** 页面显示出一个 events 列表。
9. 在管理门户中关闭数据库虚拟机 **db28532[自定义名称]**。
10. 关闭 **Internet Explorer** 应用程序。
11. 关闭 **Visual Studio** 应用程序。

结果：完成此练习后，您将使用虚拟网络中数据库虚拟机的内部 IP 地址连接到了 SQL Server 2014。

通过在右列中放置标记来判断叙述的正确性。

叙述	答案
可以使用 SQL Server Management Studio 从任何计算机，按其当前配置连接到 SQL Server VM。	

章节复习和作业

本章中，您观察了虚拟机如何连接到虚拟网络，以及如何自定义每个虚拟机的联网选项。此外还讨论了虚拟机的缩放和配置管理选项。

复习问题

问题：您在一家没有任何现场硬件的小公司工作。但是，他们希望其软件开发人员能够使用 VPN 连接到 Azure 虚拟网络来测试软件功能。应该建立点到站点网络连接，还是站点到站点网络连接？

第 11 章 自动化与 Azure 资源的集成

目录:

章节概述	11-2
第 1 课: Azure SDK 客户端库	11-3
第 2 课: 使用 Windows PowerShell 编写 Azure 服务管理脚本	11-7
第 3 课: Azure REST 接口	11-14
第 4 课: Azure 资源管理器	11-16
实验 A: 使用 PowerShell 自动化测试环境的创建工作	11-21
实验 B: 使用 PowerShell 自动化测试环境的创建工作	11-26
章节复习和作业	11-29

章节概述

使用两个 Azure 门户或 Microsoft Visual Studio 2013 可以管理大多数 Azure 服务。但面对大量同类资源的管理，手工操作就显得费时费力，这时使用脚本来自动化管理比较合适。本章将探讨使用客户端库、Windows PowerShell、REST 和资源管理器来自动化服务的生命周期。第 1 节“Azure SDK 客户端库”简要描述一些 Powershell 模块，这些模块可用于管理 Azure 服务并与之交互。第 2 节“使用 Windows PowerShell 编写 Azure 服务管理脚本”描述可用来通过 Windows PowerShell 管理 Azure 服务的模块。第 3 节“Azure REST 接口”介绍并描述服务管理 API。第 4 节“资源管理器”讨论 Azure 中的新资源管理器，以及与管理资源的新方法有关的概念。

目标

完成本章后，您可以做到：

- 描述 Azure 软件开发包 (SDK) 和客户端库。
- 使用 Windows PowerShell 自动化 Azure 服务管理。
- 描述服务管理 API 以及向该 API 验证身份的步骤。
- 使用资源管理器创建资源组和模板。

第 1 课 Azure SDK 客户端库

各种平台都有相应的客户端库。这些库有助于管理 Azure 服务以及在现有服务中创建构造。

本节列出可用的 SDK 以及如何使用这些 SDK 来管理 Azure 服务。

课程目标

完成本节后，您可以做到：

- 描述 Azure SDK 及其平台。
- 描述如何使用 SDK 来创建云服务角色。

Azure SDK

各种平台都有使用语言操作的工具和客户端库。在将自定义应用程序与 Azure 集成时，可以使用这些工具和库。

Microsoft 平台

为了管理 Azure 服务，Visual Studio 2012 及更高版本提供了增强服务器资源管理器功能的扩展，还增加了新的项目模板。Visual Studio 可让您管理 Azure 服务，并使用服务器资源管理器显示其状态。在 Visual Studio 中，还可以使用服务器资源管理器或项目模板创建新服务和管理当前服务。对于与自定义应用程序更复杂的集成，有 .NET 库可用来管理 Azure 服务。其中的很多自定义包可直接在 NuGet 中获得。

- 不同平台都有相应的软件开发包 (SDK)
 - 通过它们可以编程方式访问 Azure 资源
- SDK 的源代码是开源的，可从 GitHub 上获得
- 有适用于 Eclipse 和 Visual Studio 的官方 Azure 工具

Azure 官方 NuGet 包

<http://go.microsoft.com/fwlink/?LinkId=525667>

对于大部分复杂的管理场景，可以用 Windows PowerShell 活动(activity)自动化您的操作。可以使用 Azure PowerShell 模块将 Windows PowerShell 活动直接安装在开发环境中。或者，在 Azure 自动化服务中也能获得 Azure PowerShell 模块。

.NET Azure 文档

<http://go.microsoft.com/fwlink/?LinkId=525668>

第三方平台

很多流行的平台和语言也有类似的 Azure 服务管理库，包括：

- Java
- Node.js
- PHP
- Python
- Ruby

此列表还在快速扩充。这些语言或框架中的绝大部分针对多种流行的操作系统（Windows、Mac OS 和 Linux），提供了相应的安装程序。官方 Azure 网站上还推出了开发人员门户，上面有教程和高级文档的链接。

在一个称为 Azure 跨平台命令行接口 (xplat-cli) 的集合中，提供了一组开源命令。Xplat-cli 提供了一个用于管理 Azure 服务的通用接口，该接口与操作系统或管理环境无关。Xplat-cli 是用 Node.js 编写的，需在本地安装了 Node.js。

PHP Azure 文档

<http://go.microsoft.com/fwlink/?LinkId=525669>

Node.js Azure 文档

<http://go.microsoft.com/fwlink/?LinkId=525670>

Java Azure 文档

<http://go.microsoft.com/fwlink/?LinkId=525671>

Python Azure 文档

<http://go.microsoft.com/fwlink/?LinkId=525672>

Ruby Azure 文档

<http://go.microsoft.com/fwlink/?LinkId=525673>

移动平台

Microsoft Azure 移动服务是一个可用来快速构建移动应用程序后端服务的平台。因为采用了动态架构，因此只要少量自定义代码，甚至无需自定义代码，便可创建这些移动后端服务，并且可随着时间的推移而不斷优化。移动服务及其动态架构功能可让开发者将移动应用程序和想法快速引入市场，而没有很多传统上的延迟和基础结构设置。移动服务还可随着应用程序缩放而缩放，并且可应付应用程序未来使用量的增长。

移动服务可通过本机库与各种移动平台集成，包括：

- iOS
- Android
- Windows Phone 8
- Windows 应用商店 (C# 或 JavaScript)

移动服务还公开 RESTful HTTP 端点，并支持跨源资源共享 (CORS)。这确保移动服务可用于无法立即获得本机库的各种场合。

移动服务文档

<http://go.microsoft.com/fwlink/?LinkId=525675>

媒体服务

Microsoft Azure 媒体服务是一个无所不包的媒体解决方案，包括编码、存储、保护媒体资产以及将这些资产传送到客户端设备。多种媒体平台（如 Silverlight 或 Flash）、开发平台（如 .NET 或 Java）以及移动平台（如 Windows 8、Windows Phone、Android 和 iOS）都有相应的库。

媒体服务

<http://go.microsoft.com/fwlink/?LinkId=525676>

云服务

云服务角色是 Web 和应用程序层的基础构件，可用来构建托管在 Azure 中的可缩放云应用程序。缩放基础结构、管理和模块化由 Azure 平台负责，而让应用程序专注于其相关代码。

可以使用 Visual Studio 2012 或更高版本中的云服务模板创建云服务。可以在 Visual Studio 中自定义云服务项目，并将其与控制台项目或 ASP.NET 项目集成。然后，可以编译这些项目，并将其作为云服务包部署到 Azure。

可以使用“发布”操作直接在 Visual Studio 中发布云服务。

- Visual Studio 和 Eclipse 插件可用来创建云服务项目和角色
 - Visual Studio 能让您使用 .NET 来创建云服务角色
 - Eclipse 能让您使用 Java 来创建云服务角色
- PowerShell 可用来手动创建其他语言的云服务角色
 - 示例：Node.js 可与使用 PowerShell 创建的角色一起使用

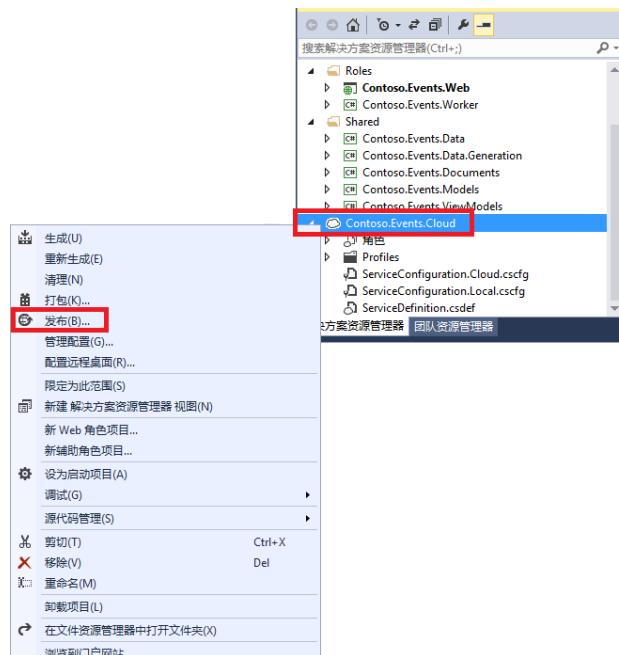


图 11.1：发布云服务

Node.js 云服务

或者，也可以使用 Windows PowerShell 和 Node.js 来创建云服务。有一个 Windows PowerShell 活动可用在本地计算机上创建目录结构。这可用于创建使用 Node.js，而不是使用 .NET 的云服务。

New-AzureServiceProject 活动会生成一个基本结构，用于创建将发布到云服务的新的 Azure 节点应用程序。

New-AzureServiceProject 活动

New-AzureServiceProject **NodeCloudProject**

服务配置和定义文件创建在新目录中，这跟 Visual Studio 或云服务包中的云服务项目类似。

Add-AzureNodeWebRole 和 Add-AzureNodeWorkerRole 活动用于将 Web 角色和辅助角色添加到新项目。

最后，可以使用 Windows PowerShell 在本地启动 Azure 计算模拟器，或者将项目发布到 Azure。

Publish-AzureServiceProject 用于将本地服务发布到 Azure 云服务实例。

Publish-AzureServiceProject

```
 Publish-AzureServiceProject -ServiceName NodeHelloWorld -Location "East US" -Launch
```

PHP 云服务

还可以在 Azure 中创建 PHP 云服务。唯一的区别是，必须使用 **Add-AzurePHPWebRole** 和 **Add-AzurePHPWorkerRole** cmdlet 将角色添加到项目。

第 2 课 使用 Windows PowerShell 编写 Azure 服务管理脚本

可以使用 Windows PowerShell 自动化 IT 专员日常执行的很多管理任务。开发人员可使用 Windows PowerShell 来简化和自动化很多开发人员运维职责。

本节描述 Azure PowerShell 模块中提供的两组 cmdlet。

课程目标

完成本节后，您可以做到：

- 描述 Azure PowerShell 模块。
- 列出使用 Windows PowerShell 向 Azure 证明身份的选项。
- 使用 Windows PowerShell cmdlet 创建 Azure 服务实例。

Azure PowerShell 模块

Microsoft Azure 自动化的体系结构是从 Microsoft Azure 结构控制器开始的。结构控制器负责查找资源，然后在 Azure 中设置虚拟机、虚拟网络以及几乎所有其他后端服务。服务管理 API 是结构控制器前面的一层，它管理来自客户端的请求，并将相应的请求传递到结构控制器。包括 Windows PowerShell 和跨平台命令行接口在内的所有 SDK 都集成了服务管理 REST API。甚至 Azure 门户使用与客户端库相同的 API。开发人员可直接选用服务管理 REST API，而不是使用客户端库或门户。

所有客户端库都使用服务管理 API。

- Azure PowerShell 是可用来管理 Azure 服务的模块
 - 安装一系列 Cmdlet
- 这些 PowerShell cmdLet 提供了管理门户中可用功能的超集
 - 很多时候，新功能还没在管理门户中推出之前，就可能早已使用 PowerShell 实现。
- 提供的另一组模块使用新的资源管理器

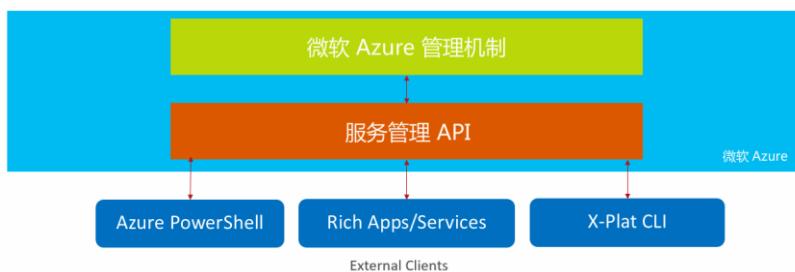


图 11.2: AZURE 自动化

Azure PowerShell

Azure PowerShell 是两个 Windows PowerShell 模块组成的集合，可用来管理 Azure 服务。第一个，也是最常用的模块是服务管理模块。此模块可用来管理 Azure 订阅中的服务实例。此外还有资源管理器模块，本章的“Azure 资源管理器”一节中将深入讨论此模块。

MSDN 上目前提供了包含在 Azure PowerShell 模块中的一系列 Windows PowerShell 活动：

Azure Cmdlet 参考

<http://go.microsoft.com/fwlink/?LinkId=525677>

将 Azure PowerShell 模块用于服务管理的主要优势之一是可以获得新功能。通常，新功能先进入服务管理 API，然后再添加到 Azure PowerShell 模块。最后，在门户中可以使用这些功能。

功能通常先出现在 Azure PowerShell，然后才会出现在管理门户中。

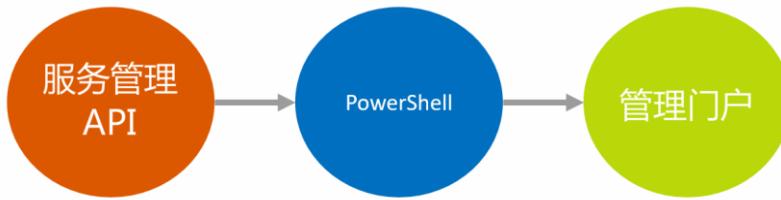


图 11.3：新功能发布顺序

这通常意味着，可以先使用 Azure PowerShell 测试和使用新功能，然后再在门户中全面推出。此外，在使用 Azure 平台时，可以使用 Windows PowerShell 创建脚本来同时自动化多项任务。这些脚本可在需要维护、存储和重复配置的开发人员运维场景中使用。

Azure 自动化

Azure 自动化是可用来根据需求或按计划运行 Windows PowerShell 工作流的服务。Azure PowerShell 活动已经导入 Azure PowerShell，很多不同的 Windows PowerShell 管理任务可直接从本地脚本导入 Azure 自动化。Azure 自动化还包含一个脚本中心，其中包含很多最常用的管理任务。这些任务通常耗时、易出错且复杂。有了 Azure 自动化，这些脚本可作为一次性作业或计划作业运行，并且可以无人照管方式在 Azure 环境中运行。



脚本中心

<http://go.microsoft.com/fwlink/?LinkId=525678>

安装 Azure PowerShell Cmdlet

要求

Azure PowerShell 可用来管理现有订阅，但是不能用于创建新订阅。使用 Azure PowerShell 模块需要订阅。Azure PowerShell 要求在本地计算机上安装 .NET Framework 4.5。安装程序检查 Windows PowerShell 的有效版本，然后 .NET Framework 在计算机上安装缺少的依赖项。

- Azure PowerShell 需要 .NET 4.5
- 安装程序可通过 Microsoft WebPI 获得
- 此外还有自定义 PowerShell 控制台，与标准 PowerShell 控制台相比，所需的环境配置更少

Web 平台安装程序

Azure PowerShell 是使用 Microsoft Web 平台安装程序安装的。在单击任何连接下载这些模块后，链接将打开 Web 平台安装程序可执行文件，并使用指定安装最新版本 Azure PowerShell 的参数。对于自动安装场景，可以使用命令行工具执行 Web 平台安装程序安装。

使用 Web 平台安装程序安装 Azure PowerShell。

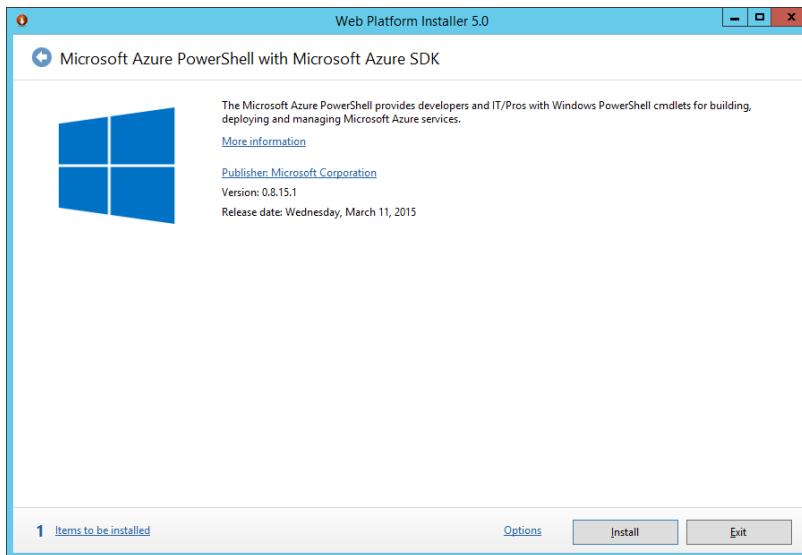


图 11.4: WEB 平台安装程序

在 Web 平台安装程序安装好 Azure PowerShell 后，本地计算机上还将安装自定义控制台。此控制台能让您立即使用 Azure PowerShell 模块，而无需手动导入这些模块。

使用 Windows PowerShell 向 Azure 证明身份

为了管理服务，cmdlet 需要您的订阅。有两种方式可将订阅信息提供给 Windows PowerShell。可以使用包含订阅信息的管理证书，也可以使用 Microsoft 帐户、工作帐户或学校帐户登录到 Azure。登录后，Microsoft Azure Active Directory (Azure AD) 将验证凭据，并返回让 Azure PowerShell 管理帐户的访问令牌。

PublishSettings 文件

Get-AzurePublishSettingsFile 和 Import-AzurePublishSettingsFile

AzurePublishSettingsFile 活动用来下载和导入包含 Azure 订阅证书的 XML 文件。这种身份验证方法通常称为证书身份验证。在使用此方法时，只要订阅和证书有效，订阅信息就会一直存在。但是，使用此方法来管理对共享订阅的访问会显得更难。例如，在多人有权访问帐户的情况下。此外，Azure 资源管理器 API 不支持证书身份验证。

Azure AD

Azure AD 是推荐的身份验证方法，因为这使得管理对订阅的访问更加容易。**Add-AzureAccount** 活动用来以此方法进行验证。在使用此方法进行验证时，系统会显示一个对话框，可以使用 Microsoft 帐户或组织帐户登录，其方式与向 Azure 门户证明身份一样。

• 有两种主要身份验证方法：

- Azure Active Directory
 - 就像向门户证明身份一样，向 Azure 证明身份
 - 验证只是暂时的（大约 12 小时）
- 发布设置
 - 打开管理门户网页
 - 如果用户尚未通过身份验证，用户需要通过网页验证身份
 - 凭借帐户令牌和信息下载 XML 文件
 - PowerShell 可永久使用发布设置文件来连接到 Azure 订阅

演示 1：使用 Windows PowerShell 管理 Azure 网站服务

在开始这个演示前，必须完成第 2 章中的实验 A。在本演示中，您将使用自己的计算机完成演示。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。

2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].cloudapp.net: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为**Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。

4. 如果有必要，使用以下用户密码登录：

- 用户名: **Student**
- 密码: **AzurePa\$\$w0rd**

5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

演示步骤

1. 打开 Microsoft Azure PowerShell 控制台窗口。

2. 用您的 Azure Active Directory 帐户登录 Azure:

Add-AzureAccount

3. 查看活跃帐户详细信息:

Get-AzureAccount

4. 执行以下命令和详细信息，下载和保存发布配置文件用于登录 Azure:

Get-AzurePublishSettingsFile

- File name: **28532**
- Save as type: **PUBLISHSETTINGS file**
- Directory: **(F):\Mod11\Demofiles**

5. 导入已保存的发布配置文件登录 Azure:

Import-AzurePublishSettingsFile F:\Mod11\Demofiles\28532.publishsettings

6. 查看活跃帐户细节:

Get-AzureAccount

7. 使用唯一的名字创建一个新的网站:

New-AzureWebsite [Unique Name]

8. 查看您的 Azure 订阅中的网站列表:

Get-AzureWebsite

9. 查看新创建网站的详细信息:

```
Get-AzureWebsite -Name [Unique Name]
```

10. 停止新创建的网站实例:

```
Stop-AzureWebsite -Name [Unique Name]
```

11. 查看您的 Azure 订阅中的网站列表:

```
Get-AzureWebsite
```

12. 移除新创建的网站实例:

```
Remove-AzureWebsite -Name [Unique Name]
```

13. 查看刚删除网站的详细信息:

```
Get-AzureWebsite -Name [Unique Name]
```

演示 2：使用 Windows PowerShell 管理中国版 Windows Azure 网站服务

在开始这个演示前，必须完成第 2 章中的**实验 B**，您将使用自己的计算机完成演示，此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：
vm28532[自定义名称].chinacloudapp.cn: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到 Azure 管理门户。您将在本课程的所有**演示 2** 中使用您的中国版 Windows Azure 帐户。

演示步骤

1. 在 Start 屏幕，单击向下箭头查看全部应用程序，然后单击 **Microsoft Azure PowerShell**。
2. 切换 **Microsoft Azure PowerShell** 控制窗口。
3. 在控制台输入以下命令并按 Enter 键获取发布配置文件：

```
Get-AzurePublishSettingsFile -environment azurechinacloud
```

4. 在打开的 **Internet Explorer** 选项卡执行以下步骤：

- a. 输入 Azure 帐户邮箱地址和密码。
 - b. 单击 **Sign In**。
 - c. 在 Internet Explorer 窗口底部，单击 **Save** 按钮右侧的箭头，然后在下载对话框中单击 **Save As**。
5. 在显示的 **Save As** 对话框中执行以下步骤：
- a. 指向 **AllFiles (F):\Mod11\ Demofiles** 文件。
 - b. 在 **File Name** 框中输入 **28532**。
 - c. 在 **Save as type** 框中，确认 **PUBLISHSETTINGS File** 选项被选中。
 - d. 单击 **Save**。
6. 关闭 Internet Explorer。
7. 切换到 **Microsoft Azure PowerShell** 控制窗口。
8. 在控制台输入以下命令然后按 Enter 键导入发布配置文件：

```
Import-AzurePublishSettingsFile F:\Mod11\Demofiles\28532.publishsettings
```

9. 在控制台输入以下命令然后按 Enter 键查看一些可用的帐户细节：

```
Get-AzureAccount
```

10. 输入以下命令，将[Unique Name]替换为一个可用的名称，按下 Enter 键，创建一个新的网站。

```
New-AzureWebsite [Unique Name]
```

 **注释：**可能会得到一个错误信息，提示这个名字不是唯一的，如果这种情况出现，选择一个新名字直到网站创建成功。

11. 在控制台输入以下命令，按下 Enter 键查看网站列表：

```
Get-AzureWebsite
```

12. 在控制台上输入以下命令，[Unique Name] 是之前新创建的名字，按下 Enter 键，获取新网站的信息细节：

```
Get-AzureWebsite -Name [Unique Name]
```

13. 在控制台输入以下命令按下 Enter 键，停止新的网站，[Unique Name] 是之前命令中新创建的名字：

```
Stop-AzureWebsite -Name [Unique Name]
```

14. 在控制台输入以下命令按下 Enter 键查看网站列表：

```
Get-AzureWebsite
```

15. 在控制台输入以下命令按下 Enter 键，移除新的网站，[Unique Name] 是之前命令中新创建的名字：

```
Remove-AzureWebsite -Name [Unique Name]
```

16. 输入 Y 表示想要移除网站，然后按下 Enter 键。

17. 在控制台输入以下命令按下 Enter 键，结果是空的，确认移除新的网站，[Unique Name] 是之前命令中新创建的名字：

MCT USE ONLY. STUDENT USE PROHIBITED

```
Get-AzureWebsite -Name [Unique Name]
```

18. 关闭 **Microsoft Azure PowerShell** 控制窗口。

第 3 课 Azure REST 接口

很多现代服务和应用程序都提供了 REST API。HTTP 是统一标准，可用于各种平台和语言。Azure 提供了服务管理 API，无论当前是什么平台，都可以使用此 API 来管理服务。

本节描述服务管理 API，以及如何向 API 进行验证。

课程目标

完成本节后，您可以做到：

- 描述服务管理 API。
- 列出用于向 API 进行验证的两个选项。

服务管理 REST API

利用服务管理 API，可以编程方式使用可通过 Azure PowerShell 或管理门户获得的功能。这个 API 是支持常用 HTTP 方法（如 **GET**、**PUT**、**POST** 或 **DELETE**）操作的 REST API。所有 API 操作都使用 X.509 证书通过安全套接字层 (SSL) 执行。

Azure 中的每个订阅都会分配到一个唯一订阅 ID。Azure 中的所有服务实例都与订阅关联，并使用订阅 ID 来引用。订阅 ID 也包含在服务管理 API 发出的每个调用的 URI 中。

MSDN 上可找到服务管理 API REST 资源的文档：



<http://go.microsoft.com/fwlink/?LinkId=525679>

- 可使用 REST API 和常见 HTTP 谓词来管理 Azure 资源
 - API 操作使用 SSL 执行
 - REST API 得到各种编程平台和脚本平台的支持
- 门户中的很多同样的功能可与服务管理 API 一起使用

向服务管理 REST API 验证请求

向服务管理 API 验证请求有两种主要方法。Azure Active Directory 可用于来自自定义应用程序的验证。管理证书可用于验证来自各种其他客户端的管理任务。

管理证书

通过 SSL 使用管理证书可验证发给管理服务的安全请求。为了使用管理证书，必须将证书上传到 Azure。在将管理证书添加到订阅后，可以使用同一证书给发给服务的请求签名。服务管理 API 不验证证书是否仍然有效，因此可与已过期或无效的证书一起使用。此外，基于证书的身份验证不支持基于角色的身份验证。

Azure Active Directory

创建 Azure AD 应用程序，并使用 Active Directory 身份验证库可以验证发给管理服务的安全请求，通过验证后可从该应用程序获得访问令牌。

可以两种方式进行向服务管理 API 的验证：

- 使用 Azure Active Directory
 - 将发出调用的应用程序添加为 Azure AD 中的应用程序。然后可以将使用服务管理 API 的权限委派给应用程序。
- 使用管理证书
 - 可以下载 X509 证书，并将其作为 HTTP 请求的一部分用来访问服务管理 API

MCT USE ONLY. STUDENT USE PROHIBITED

 用于.NET 的 Azure AD 身份验证库

<http://go.microsoft.com/fwlink/?LinkId=525680>

用于 .NET 的 Azure AD 身份验证库可使客户端应用程序开发人员轻松地向云或企业内部 Active Directory 验证用户身份，然后获得用于保护 API 调用的访问令牌。用于 .NET 的 Active Directory 身份验证库有很多便于开发人员更轻松地进行验证的功能。例如，异步支持、存储访问令牌和刷新令牌的可配置令牌缓存，以及在访问令牌过期而刷新令牌可用时的自动令牌刷新。因为解决了这些复杂任务，因此应用程序可以注重相关代码，而不是使用 API 进行验证。

在 Azure AD 中创建本机客户端应用程序后，需要分配访问服务管理 API 的委派权限，然后在自定义应用程序中使用 **clientId** 和 **tenantId** 参数。

第 4 课 Azure 资源管理器

随着新版 Azure 预览门户 (Ibiza) 的推出，一种管理 Azure 资源的新方法也浮现出来。利用资源组和资源组模板，自动化创建和监视多服务工作负载将变得容易得多。

本节描述新的资源管理器产品。

 **注释:** 本节中介绍的内容（例如，预览门户、资源管理器等）在中国版 Windwos Azure 中暂时不支持。

课程目标

完成本节后，您可以做到：

- 描述 Azure 资源管理器。
- 描述资源组和资源组模板。
- 使用资源管理器将多个资源作为一个托管单元来创建。

Azure 资源管理器概述

资源管理器引入了一种全新的 Azure 资源管理方法。服务实例现在称为资源，资源可逻辑地存储到资源组中。资源组为子资源提供了共同的生命周期。可以一起创建、管理、监视或删除这些子资源。资源管理器还提供了资源组模板的概念，这种模板能让您事先定义服务单元，然后使用模板创建所需数量的资源组。

如果需要快速构建本质上同类，并且可按照共同的生命周期进行管理的开发、测试、质量保证或生产环境，那么在这样的开发人员运维场景下，非常适合使用资源组和资源组模板。开发人员可以快速删除其环境，并使用共享模板创建新环境。可以监视资源组来更总体地确定费率或资源使用量，而不是监视各个服务实例。

Azure 中的资源管理器功能是全新的，只在预览门户中才能看到。现在有一组 Windows PowerShell cmdlet 可用于管理资源组。但是其功能受限，在使用 Windows PowerShell 和 Azure 门户时，只有一部分 Azure 服务能在资源组管理。以下是部分当前限制：

- 并非所有服务都可纳入资源组。例如，API 管理实例就不可添加到资源组。
- 资源组模板只能用来管理某些服务，还不能用于虚拟机。

但是，新的资源管理器功能每周都会发布，因此这些限制可能很快就会改变。

- Azure 资源管理器彻底改造了分组和管理资源的方式
 - 您的服务称为资源
 - 资源归入资源组
 - 可以创建资源组模板来自动化多个资源的创建过程

资源组

每当在预览门户中创建资源时，资源总是创建在资源组中。在创建服务实例时，可以选择创建新资源组，也可以使用现有资源组。您可能还注意到，在管理门户中创建的某些资源也放入了资源组。这些资源组只在预览门户中可见。

在创建包含几个共同工作的资源的应用程序时，应用程序总是创建在自己的资源组中，这样就可以使用资源组来管理所有相关资产的生命周期。随着应用程序的发展，可以在资源组中添加更多资源，或从中移除资源。例如，预览门户中的 **网站 + SQL** 选项创建的新资源组就包含网站服务、Web 宿主计划、SQL 数据库和服务器，还有其他资源。

可以使用预览门户或 Azure PowerShell 查看资源组。使用预览门户时，可以将资源组作为一个组来查看和监视。

资源组有边栏选项卡，上面提供了有关特定资源组的所有信息。

• 资源组是相关资源的集合

- 通过将资源作为一个组来管理，可以将资源作为整体来创建、修改或销毁
- 可将资源作为逻辑单元来迁移和管理
- 资源组通常称为生命周期边界，因为它们现在可以共有同一个托管服务生命周期
- 资源组也可用来单独跟踪一个特定逻辑单元的使用量和计费

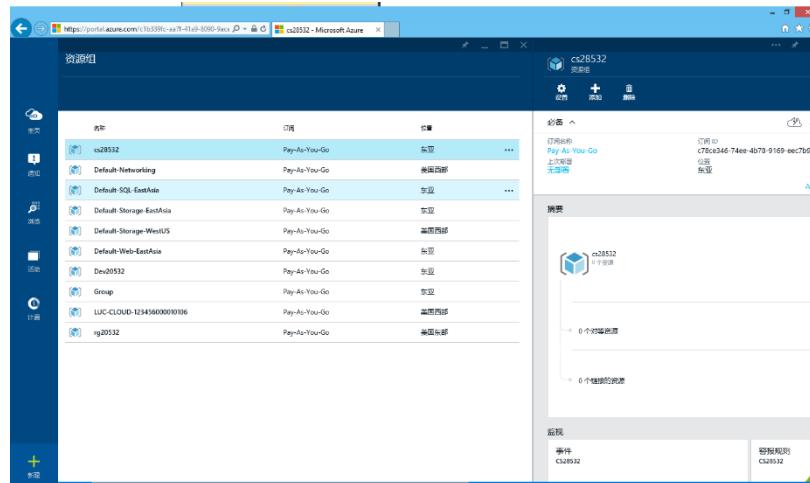


图 11.5：资源组边栏选项卡

资源可随时添加到资源组。预览门户有一个“添加按钮”，可用来将新资源添加到资源组。资源组还可用来管理组中所含的所有资源的生命周期。删除资源组将同时删除其中所含的所有资源。

Azure PowerShell 中的资源管理器模式能让您管理 Azure 订阅中的资源组。资源组是使用 **New-AzureResourceGroup** cmdlet 创建的。可以用一个名称和位置创建资源组，然后使用 **New-AzureResource** cmdlet 手动创建资源，并将其添加到资源组。还可以使用资源组模板根据现有定义创建资源组。

资源组模板

大多数设计为在 Azure 中运行的应用程序使用多种资源的组合（如数据库服务器、数据库和网站服务）来按设计运行。Azure 资源管理器模板可让您使用资源的 JavaScript 对象表示法 (JSON) 描述以及关联的部署参数，同时部署和管理这些资源。创建资源组及其资源有多种方法，但是最简单的方法是使用资源组模板。资源组模板是定义资源组中的资源的 JSON 字符串。该字符串包含称为参数的占位符来表示用户定义值，如名称和大小。

模板由不同的部分组成，包括：

- 资源组模板提供了一种方法来设计 Azure 中的资源集合模型
- 示例：
 - 资源组模板可用来设计应用程序在 Azure 中的生产体系结构。然后模板可用来构建相同的测试和暂存环境
 - 模板提供了一种构建可重复服务模型的快速一致的方法
 - 模板作为 JSON 存储，可使用标准架构创建和自定义

\$schema	必须指定一个架构文件，指出应使用的模板语言的版本。
parameters	参数可在模板中指定，以使同一个模板可用于多个资源组。例如，资源组模板可以被用来创建宿主计划和网站。通过使用参数，同一个模板可用于创建标准层宿主计划或免费层宿主计划。
variables	变量是可用于模板中资源的可重用数据。这样可以减少模板中重复内容的量，并遵守 DRY (don't repeat yourself，不要重复) 软件开发原则。
resources	资源部分是模板中定义的各个资源的 JSON 数组。这部分是分层的，可定义为使得某些资源因为相关性而先创建。例如，模板中的网站资源可在其定义中嵌套 Web 宿主计划资源。这将确保先创建 Web 宿主计划，再创建网站实例。

资源组模板 JSON 架构（语言）还允许您指定表示资源组创建时间的输出值以及可在整个模板中使用的函数。

Azure 提供了一个资源组模板库，您可以从头开始创建自己的模板，也可以编辑库模板来创建。使用 Azure PowerShell 下载现有资源组模板。**Save-AzureResourceGroupGalleryTemplate** cmdlet 将模板从库中下载到 JSON 文件。

Save-AzureResourceGroupGalleryTemplate cmdlet 可用来查看现有的库资源组模板。

Save-AzureResourceGroupGalleryTemplate

```
Save-AzureResourceGroupGalleryTemplate -Identity Microsoft.WebSiteSQLDatabase.0.2.2-preview -Path D:\Azure\Templates
```

网站的示例资源组模板

资源组模板 JSON

```
{
  "parameters": {
    "siteName": {
      "type": "string"
    },
    "hostingPlanName": {
      "type": "string"
    },
    "siteLocation": {
      "type": "string"
    },
    "sku": {
      "type": "string",
      "allowedValues": [
        "Free",
        "Shared",
        "Standard"
      ]
    }
  }
}
```

```

        "Basic",
        "Standard"
    ],
    "defaultValue": "Free"
},
{
    "resources": [
        {
            "name": "[parameters('siteName')]",
            "type": "Microsoft.Web/Sites",
            "location": "[parameters('siteLocation')]",
            "dependsOn": [
                "[concat('Microsoft.Web/serverFarms/',
parameters('hostingPlanName'))]"
            ],
            "properties": {
                "sku": "[parameters('sku')]",
                "name": "[parameters('siteName')]"
            }
        }
    ]
}

```

资源管理器模板架构可在 MSDN 上获得:

Azure 资源管理器模板语言

<http://go.microsoft.com/fwlink/?LinkId=525682>

演示 3: 查看资源组模板

演示步骤

1. 打开 **Microsoft Azure PowerShell** 控制台窗口。
2. 切换到 **AzureResourceManager** 模式:

```
Switch-AzureMode AzureResourceManager
```

3. 用您的 Azure Active Directory 帐户登录 Azure:

```
Add-AzureAccount
```

4. 在您的 Azure 订阅中查看模版资源库列表:

```
Get-AzureResourceGroupGalleryTemplate
```

5. 使用以下命令查看 **MSOpenTech.OracleDatabase12candWebLogicServer12cSE** 资源组模板的描述:

```
Get-AzureResourceGroupGalleryTemplate -Identity
MSOpenTech.OracleDatabase12candWebLogicServer12cSE.0.2.9-preview
```

6. 使用以下命令下载 **MSOpenTech.OracleDB12cWebLogicServer12cStand** 资源组模版中的 JSON 模版:

```
Save-AzureResourceGroupGalleryTemplate -Identity
MSOpenTech.OracleDatabase12candWebLogicServer12cSE.0.2.9-preview -Path F:\Mod11\Demofiles
```

7. 从以下位置打开 **MSOpenTech.OracleDatabase12candWebLogicServer12cSE.0.2.9-preview.json** 文件:

- 文件位置: **Allfiles (F):\Mod11\Demofiles**

实验 A：使用 PowerShell 自动化测试环境的创建工作

场景

 **注释：**本章为您提供了两套试验方案，如果您拿到的是国际版的 Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版的 Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 安装 Azure PowerShell 模块。
- 使用 Windows PowerShell 向 Azure 证明身份。
- 使用 Windows PowerShell 创建网站服务实例。
- 使用 Windows PowerShell 移除网站服务实例。
- 使用 Windows PowerShell 基于模板创建资源组。
- 使用 Windows PowerShell 移除资源组。

实验设置

预计时间：45 分钟

在开始这个实验前，必须完成第 2 章中的实验。在本实验中，您将使用自己的计算机完成实验，此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].cloudapp.net: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

本章为您提供了两套试验方案，如果您拿到的是国际版的 Azure 帐户，请继续完成**实验 A**；如果您拿到的是中国版的 Azure 帐户，请转到**实验 B**。具体请咨询培训讲师，并在讲师的指导下完成实验。

MCT USE ONLY. STUDENT USE PROHIBITED

练习 1：准备 Azure PowerShell 环境

场景

在此练习中，您将：

- 安装 Azure PowerShell 模块和 cmdlet。
- 使用发布设置文件在 Windows PowerShell 中向 Azure 证明身份。
- 使用 Windows PowerShell 查看帐户详细信息。

此练习的主要任务如下：

- 打开 Azure PowerShell 控制台
- 使用发布设置文件导入 Azure 订阅

► 任务 1：打开 Azure PowerShell 控制台

- 打开 **Microsoft Azure PowerShell** 控制台窗口。

► 任务 2：使用发布设置文件导入 Azure 订阅

- 通过使用以下命令和详细信息，下载和保存发布配置文件用于登录 Azure：

```
Get-AzurePublishSettingsFile
```

- File name: **28532**
- Save as type: **PUBLISHSETTINGS file**
- Directory: **(F):\Mod11\Labfiles**

- 导入已保存的发布配置文件登录 Azure：

```
Import-AzurePublishSettingsFile F:\Mod11\Labfiles\28532.publishsettings
```

- 查看活跃帐户详细信息：

```
Get-AzureAccount
```



注释：如果有多个 Azure 订阅，可以使用 **Select-AzureSubscription** 选择合适的订阅。如果 Azure 订阅是由培训机构提供的，它可能是一个没有详细信息的自动生成的帐户，在这种情况下，执行这个命令返回的将是一个空结果。

结果：完成此练习后，您将设置好开发环境，以便使用 Windows PowerShell 来进行 Azure 服务自动化。

练习 2：使用 Windows PowerShell 创建和访问网站服务实例

场景

在此练习中，您将：

- 使用 Windows PowerShell 查看网站服务的列表。
- 使用 Windows PowerShell 创建网站服务。
- 使用 Windows PowerShell 删除网站服务。

此练习的主要任务如下：

1. 使用 Windows PowerShell 创建新的网站服务实例
2. 使用 Windows PowerShell 移除网站服务实例

► **任务 1: 使用 Windows PowerShell 创建新的网站服务实例**

1. 使用唯一的名字创建一个新的网站:

```
New-AzureWebsite [Unique Name]
```

2. 查看您的 Azure 订阅中的网站列表:

```
Get-AzureWebsite
```

3. 通过以下命令查看新创建网站的详细信息:

```
Get-AzureWebsite -Name [Unique Name]
```

4. 记录新网站详细信息中的 **hostname** 值。

5. 使用以下命令查看新创建的网站:

```
explorer "http://[Host Name]"
```

6. 关闭 Internet Explorer。

7. 切换到 **Microsoft Azure PowerShell** 控制台窗口。

► **任务 2: 使用 Windows PowerShell 移除网站服务实例**

1. 停止新创建的网站实例:

```
Stop-AzureWebsite -Name [Unique Name]
```

2. 查看您的 Azure 订阅中的网站列表:

```
Get-AzureWebsite
```

3. 使用以下命令移除新创建的网站实例:

```
Remove-AzureWebsite -Name [Unique Name]
```

4. 使用以下命令查看刚删除网站的详细信息:

```
Get-AzureWebsite -Name [Unique Name]
```

结果: 完成此练习后，您就会使用 Windows PowerShell 来管理 Azure 服务的实例。

练习 3：使用资源模板创建多个预配置的资源

场景

在此练习中，您将：

- 使用 Azure Active Directory 在 Windows PowerShell 中向 Azure 验证身份。
- 使用 Windows PowerShell 切换正在使用的 Azure 模块。
- 使用 Windows PowerShell 基于模板创建资源组。
- 使用 Windows PowerShell 删除资源组。

此练习的主要任务如下：

1. 切换到 Azure 资源管理器 PowerShell 模块
2. 使用 Azure Active Directory 导入 Azure 订阅
3. 使用 Windows PowerShell 创建资源组
4. 使用 Windows PowerShell 移除资源组

► **任务 1：切换到 Azure 资源管理器 PowerShell 模块**

1. 通过以下命令切换到 **AzureResourceManager** 模式：

```
Switch-AzureMode AzureResourceManager
```

► **任务 2：使用 Azure Active Directory 导入 Azure 订阅**

1. 用您的 Azure Active Directory 帐户登录 Azure：
2. 查看活跃帐户细节：

```
Get-AzureAccount
```

注释：如果有多个 Azure 订阅，可以使用 **Select-AzureSubscription** 选择合适的订阅。

► **任务 3：使用 Windows PowerShell 创建资源组**

1. 使用以下命令查看 **Microsoft.WebSiteSQLDatabase** 资源组模板的描述：

```
Get-AzureResourceGroupGalleryTemplate -Identity Microsoft.WebSiteSQLDatabase.0.2.0-preview
```

2. 使用以下命令和详细信息创建一个新的 **Microsoft.WebSiteSQLDatabase** 资源组模板实例：

```
New-AzureResourceGroup -Name rga28532 -Location "West Us" -GalleryTemplateIdentity Microsoft.WebSiteSQLDatabase.0.2.0-preview
```

- siteName: **rs28532**[自定义名称]。
- hostingPlanName: **rp28532**[自定义名称]。
- siteLocation: **West US**。
- serverName: **rv28532**[自定义名称]。
- serverLocation: **West US**。
- administratorLogin: **testuser**。
- administratorLoginPassword: **TestPa\$\$w0rd**。
- databaseName: **rd28532**[自定义名称]。

注释：等待设置进程完成。状态信息将会定期显示，并且当设置完成，最终的信息细节将会全部显示。

3. 执行以下命令查看新的资源组细节：

```
Get-AzureResourceGroup -ResourceGroupName rga28532
```

4. 执行以下命令查看在资源组中新创建网站的详细信息：

```
Get-AzureResource -Name rs28532[Your Name Here] -ResourceGroupName rga28532 -ResourceType "Microsoft.Web/sites" -ApiVersion 2014-04-01
```

5. 记录新网站详细信息中 **hostNames** 数组中的那个值。

6. 执行以下命令查看新创建的网站：

```
explorer "http://[Host Name]"
```

7. 关闭 Internet Explorer。

8. 切换到 **Microsoft Azure PowerShell** 控制窗口。

► **任务 4：使用 Windows PowerShell 移除资源组**

1. 执行以下命令移除新创建的资源组：

```
Remove-AzureResourceGroup -Name rga28532
```



注释：因为移除一个资源组涉及到移除很多资源，这个进程平均可能花费 5~10 分钟。

结果：完成此练习后，您就会使用 Azure 来与资源管理器、资源组和资源组模板交互。

问题：在自动化 Azure 帐户和资源管理时，应该使用资源管理器 cmdlet 还是使用标准 cmdlet？

知识测试

问题	
哪个 Azure cmdlet 可用来验证资源组是否已被删除？	
请选择正确答案。	
	Remove-AzureResourceGroup
	Get-AzureResourceStatus
	Get-AzureResourceGroup
	Delete-AzureResourceGroup
	Get-AzureResourceGroupTemplate

实验 B：使用 PowerShell 自动化测试环境的创建工作

目标

完成本实验后，您将能够：

- 安装 Azure PowerShell 模块。
- 使用 Windows PowerShell 向 Azure 证明身份。
- 使用 Windows PowerShell 创建服务网站实例。

使用 Windows PowerShell 移除网站服务实例。

预计时间：45 分钟

在开始这个实验前，必须完成第 2 章中的**实验 B**。在本实验中，您将使用自己的计算机完成实验，此外，还必须完成以下步骤：

- 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
- 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：
vm28532[自定义名称].chinacloudapp.cn: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为**Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

- 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
- 如果有必要，使用以下用户密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
- 验证您拿到的是中国版 Windows Azure 帐户。
- 验证使用您的中国版 Windows Azure 帐户可以登录到 Azure 管理门户。您将在本课程的所有**实验 B** 中使用您的中国版 Windows Azure 帐户。

练习 1：准备 Azure PowerShell 环境

场景

在此练习中，您将：

- 安装 Azure PowerShell 模块和 cmdlet。
- 使用发布设置文件在 Windows PowerShell 中向 Azure 证明身份。
- 使用 Windows PowerShell 查看帐户详细信息。

此练习的主要任务如下：

- 打开 Azure PowerShell 控制台
- 使用发布设置文件导入 Azure 订阅

- 任务 1：打开 **Azure PowerShell** 控制台
- 在 Start 屏幕，单击下拉箭头查看全部应用程序，然后单击 **Microsoft Azure PowerShell**。
 - 打开 **Microsoft Azure PowerShell** 窗口。

► 任务 2：使用发布设置文件导入 Azure 订阅

- 在控制台输入以下命令并且按下 **Enter** 键，获取发布配置文件。

```
Get-AzurePublishSettingsFile -environment azurechinacloud
```

- 在打开的 **Internet Explorer** 选项卡，执行以下步骤：
 - 输入用于登录帐号的邮件地址。
 - 输入帐户密码。
 - 单击 **Sign In**。
 - 在 Internet Explorer 窗口底部，单击 **Save** 按钮右侧的箭头，然后在下载对话框中单击 **Save As**。
- 在显示的 **Save As** 对话框中执行以下步骤：
 - 指向 **AllFiles (F):\Mod11\Labfiles** 文件夹。
 - 在 **File Name** 框中输入 **28532**。
 - 在 **Save as type** 框中，确认 **PUBLISHSETTINGS File** 选项被选中。
 - 单击 **Save**。
- 关闭 Internet Explorer。
- 切换到 **Microsoft Azure PowerShell** 应用程序。
- 在控制台输入以下命令然后按 Enter 键导入发布配置文件：

```
Import-AzurePublishSettingsFile F:\Mod11\Labfiles\28532.publishsettings
```

- 在控制台输入以下命令并按 Enter 键查看帐户详细信息：

```
Get-AzureAccount
```

 **注释：**如果有多个 Azure 订阅，可以使用 **Select-AzureSubscription** 选择最合适的订阅。如果 Azure 订阅是由培训机构提供的，它可能是一个没有详细信息的自动生成的帐户，这类情况，执行这个命令返回的将是一个空结果。

结果：完成此练习后，您将设置好开发环境，以便使用 Windows PowerShell 来进行 Azure 服务自动化。

练习 2：使用 Windows PowerShell 创建和访问网站服务实例

场景

在此练习中，您将：

- 使用 Windows PowerShell 查看网站服务的列表。
- 使用 Windows PowerShell 创建网站服务。
- 使用 Windows PowerShell 删除网站服务。

MCT USE ONLY STUDENT USE PROHIBITED

此练习的主要任务如下：

1. 使用 Windows PowerShell 创建新的网站服务实例
2. 使用 Windows PowerShell 移除网站服务实例

► **任务 1：使用 Windows PowerShell 创建新的网站服务实例**

1. 输入以下命令，将命令中[Unique Name]替换为自定义名称，按下 Enter 键，创建一个新的网站。

```
New-AzureWebsite [Unique Name]
```

 **注释：**可能会得到一个错误信息，提示这个名字不是唯一的，如果这种情况发生，选择一个新名字直到网站创建成功。

2. 在控制台输入以下命令并且按下 Enter 键，查看网站列表：

```
Get-AzureWebsite
```

3. 在控制台上输入以下命令，[Unique Name] 是之前新创建的名字，按下 Enter 键，获取新网站的信息细节：

```
Get-AzureWebsite -Name [Unique Name]
```

4. 记录由之前命令生成的键值实例列表中 **hostname** 的值。
5. 在控制台输入以下命令并且按下 Enter 键，在 Internet Explorer 上查看新网站：

```
explorer "http://[Host Name]"
```

6. 确认新网站正常运行。
7. 关闭 Internet Explorer。
8. 切换 **Microsoft Azure PowerShell** 控制台窗口。

► **任务 2：使用 Windows PowerShell 移除网站服务实例**

1. 在控制台输入以下命令按下 Enter 键，停止新的网站，[Unique Name] 是之前命令中新创建的名字：

```
Stop-AzureWebsite -Name [Unique Name]
```

2. 在控制台输入以下命令查看网站列表：

```
Get-AzureWebsite
```

3. 在控制台输入以下命令按下 Enter 键，移除新的网站，[Unique Name] 是之前命令中新创建的名字：

```
Remove-AzureWebsite -Name [Unique Name]
```

4. 输入 **Y** 表示想要移除网站，然后按下 Enter 键。
5. 在控制台输入以下命令按下 Enter 键，结果是空的，确认移除新的网站，[Unique Name] 是之前命令中新创建的名字：

```
Get-AzureWebsite -Name [Unique Name]
```

6. 关闭 **Microsoft Azure PowerShell** 控制台窗口。

结果：完成此练习后，您就会使用 Windows PowerShell 来管理 Azure 服务的实例。

章节复习和作业

本章中，我们介绍了自动化 Azure 管理方面的各种方法。虽然我们讨论了 SDK、客户端库、xplat CLI 以及服务管理 API，但是主要重点是放在使用 Windows PowerShell 进行自动化。Windows PowerShell 可用来管理服务。无论是使用服务管理 cmdlet，还是 Azure 资源管理器 cmdlet 都可进行服务管理。

 **最佳做法：**从长远看，Azure 资源管理器是管理 Azure 资源的首选方法。应该使用资源组将资源分组成逻辑单元，对于可能需要多次创建的资源，应该创建模板。

复习问题

问题：在自动化一组 Azure 服务实例的创建过程时，您会选用标准 Windows PowerShell 模块，还是新的资源管理器模块？您的决定取决于哪些因素？

问题：如何创建您自己的资源组模板？

MCT ISE ONLY. STUDENT USE PROHIBITED

第 12 章 保护 Azure Web 应用程序

目录:

章节概述	12-2
第 1 课: Azure Active Directory	12-3
第 2 课: Azure AD 目录	12-5
第 3 课: Azure AD 多重身份验证	12-9
实验 A: 将 Azure Active Directory 与事件管理门户集成	12-11
章节复习和作业	12-15

章节概述

与企业内部应用程序一样，云中的应用程序也需要灵活、精简的安全机制。Azure Active Directory 是一种可以为自定义应用程序或 SaaS 应用程序提供身份认证和访问功能的身份提供程序。第 1 节“Azure Active Directory”介绍 Azure AD 服务。第 2 节“Azure AD 目录”详述如何在 Azure AD 中创建目录。第 3 节“Azure AD 多重身份验证”描述 Azure AD 中的多重身份验证功能。

 **注释:** Active Directory 的高级功能（例如 ACS，统一身份认证等）在中国版 Windows Azure 中暂时不支持，本章中的实验不适用中国版 Windows Azure。

目标

完成本章后，您可以做到：

- 描述 Azure AD 服务。
- 解释 Azure AD 中的目录提供的功能。
- 描述 Microsoft Azure 多重身份验证服务。

第 1 课 Azure Active Directory

Azure AD 提供了可与自定义应用程序、企业内部计算机、现有域以及第三方服务集成的服务套件。

本节描述 Azure AD 服务及其功能和优势。

课程目标

完成本节后，您可以做到：

- 解释 Azure 中 Active Directory 的优势。
- 列出 Azure 中的 Active Directory 服务。

Azure Active Directory 概述

Azure Active Directory (Azure AD) 是提供 Active Directory 的身份认证和访问功能的一种服务，无论应用程序是企业内部应用程序，还是托管在云中的应用程序，都可使用 Azure AD 提供的这些功能。Azure AD 可用来：

- 为自定义业务线 (LOB) 应用程序以及各种第三方软件即服务 (SaaS) 提供程序实现单一登录 (SSO) 和退出。
- 使用标准 API 查询和修改目录对象，包括用户、应用程序和组。
- 通过同步身份和（可选的）凭据，将云应用程序和 SaaS 应用程序与现有企业内部身份管理系统集成。

- Azure 中的托管身份认证和访问管理解决方案
- 着重管理域、用户和应用程序
- 丰富的单一登录解决方案
- 支持现有标准协议，如：
 - SAML 2.0
 - WS-Federation
 - OpenID Connect
 - OAuth 2.0

利用身份同步，现有的企业凭据可用来向托管在 Azure 中的新应用程序或现有应用程序证明身份。这些凭据还可用来访问第三方 SaaS 应用程序，如 Dropbox、Intuit 或 Skype。Azure AD 提供了自助门户，用户可在上面视情况管理自己的密码或组。使用密码回写功能，更新的密码哈希将复制回企业内部 Active Directory 实例。

应用程序开发人员可在自己的自定义应用程序中将 Azure AD 用作身份提供程序，进而为用户提供真正的 SSO 体验。现有应用程序可更新为使用特定 Azure AD 租户来表示身份。SaaS 应用程序还可修改为支持 Azure AD 作为身份提供程序。

目前，已经有很多云服务在使用 Azure AD，如 Microsoft Intune 和 Office 365。这些服务依赖于 Azure AD 提供的身份管理功能。这些功能包括基于云的目录数据存储以及一组核心身份服务，包括用户登录处理以及身份验证和联合服务。

Active Directory 与 Azure AD 之间的关系

Active Directory 在企业内部环境中作为身份数据的存储库，与此相似，Azure AD 也提供了一个存储库，用来存储企业在云中的所有目录数据，这样您所订阅的所有服务就可以随时使用这些数据。与 LOB 应用程序可使用轻型目录访问协议 (LDAP) 来访问本地 Active Directory 中的数据一样，第三方云应用程序可以使用 Graph API 来与 Azure AD 中的数据交互。

本地或云应用程序使用相似的方法来访问存储在目录中的身份数据。

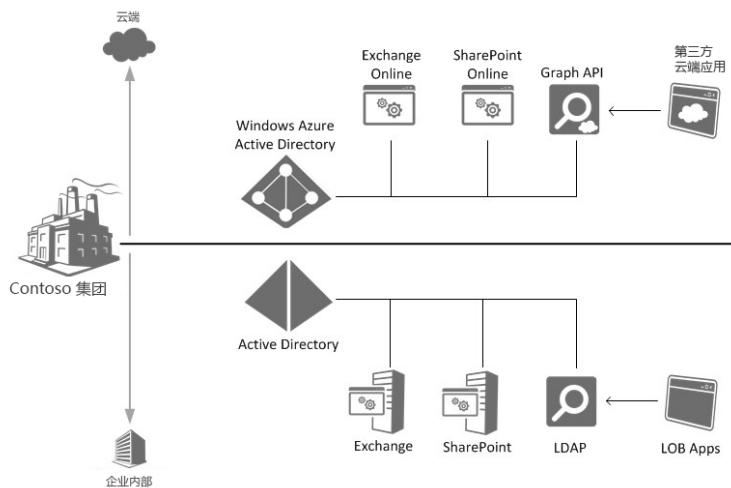


图 12.1: ACTIVE DIRECTORY 和 AZURE AD

Azure AD 服务

Azure AD 包含多种功能。本章着重介绍其中的两个功能，目录和多重身份验证。

目录服务

Azure AD 提供了可存储相关用户帐户的概念目录。目录可存储从企业内部系统同步的身份、在 Azure 中创建的身份，以及第三方身份。然后可以配置这些身份以用于 SaaS 应用程序。

多重身份验证

多重身份验证为完全托管的应用程序提供了又一道身份验证。管理员只需要配置多重身份验证，应用程序就可以将 Azure AD 用作身份验证（身份）提供程序来利用此功能。多重身份验证支持来自移动应用程序、短消息或手机的身份验证。

访问控制服务 (Access Control Service)

 **注释:** ACS 是已不推荐使用的服务。您可能继承使用此服务的应用程序，因为该服务仍在运行。

访问控制服务 (ACS) 是 Azure 中将多个身份提供程序联合成一组标准化声明的服务。通常，您需要针对每一种身份提供程序编写代码，并以自定义的方式处理其声明。ACS 允许添加实现了 OAuth 2.0 的身份提供程序，并将其声明映射到一组新的声明。例如，可以使用 ACS 将来自 Microsoft、Google、Yahoo 和 Facebook 的声明映射到一组声明，应用程序可轻松获得这些声明。这大大简化了支持多身份提供程序所必需的代码量。ACS 还支持使用 WS-Trust 或 WS-Federation 作为其协议的身份提供程序。ACS 还可以托管应用程序的登录页（可选）。

ASP.NET 身份提供了类似的功能，主要用在以前适合 ACS 的场合中。



MCT USE ONLY. STUDENT USE PROHIBITED

第 2 课 Azure AD 目录

Azure AD 目录提供了一种分组用户和应用程序的逻辑方法。

本节介绍目录并详述可用于 Azure AD 的不同组件和集成。

课程目标

完成本节后，您可以做到：

- 描述用户和第三方帐户如何添加到目录。
- 描述 SaaS 应用程序与 Azure 目录的集成。
- 描述 Azure AD Graph API。

管理目录

目录提供了一种简单的逻辑方法来分组相关身份。目录可包含以下三种类型的身份：

- 从现有的 Active Directory 安装同步的用户（企业内部身份）
- 手动添加到目录的用户（仅云的身份）
- 第三方帐户（第三方身份）

可以从各种位置管理目录。上面的位置列表并非详尽列表。

可以将管理 Azure AD 的各种经验结合在一起使用。

• 可以使用三种工具中的任一工具管理组织的租户数据：

- Microsoft Azure AD 门户
- Windows Intune 帐户门户
- Microsoft Azure 管理门户
- Office 365 帐户门户

• 在管理门户中，可以执行很多任务，如：

- 创建、修改和弃置用户帐户
- 管理密码

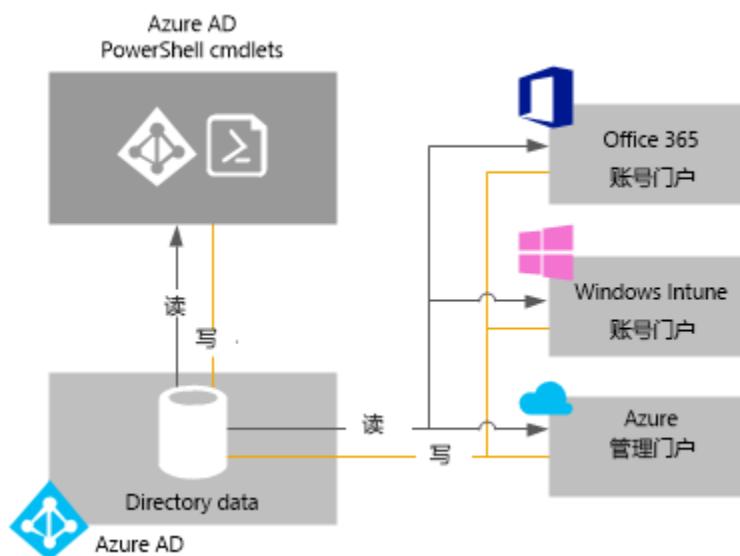


图 12.2: AZURE AD 管理

Microsoft Intune 帐户门户和 Office 365 管理中心

可以使用帐户门户来管理 Office 365 或 Microsoft Intune 订阅，并指定可访问其各种服务的用户。从帐户门户，可以执行多种任务，如手动添加用户帐户和安全组、设置和管理服务设置、检查服务状态，以及访问联机帮助。用户也可以访问这些帐户门户，但是只能更改其密码，或访问他们已获许可的各种服务。

Microsoft Azure 管理门户

可以使用 Azure 管理门户来管理与 Azure 订阅关联的服务。如果现有 Azure 订阅使用 Microsoft 帐户，那么还可以使用管理门户来管理目录。大多数 Azure 订阅包含默认 Azure AD 实例。在以组织身份登录 Azure 时，系统将根据您在注册期间在“组织名称”中提供的值，自动为您创建一个目录租户。

Windows PowerShell

可以结合 Microsoft Azure Active Directory 模块使用 Windows PowerShell cmdlet 来完成很多 Azure AD 租户级的管理任务。使用 Windows PowerShell 脚本或 Azure 自动化等服务可以自动化如用户管理和域管理，以及配置 SSO 等管理任务。

将企业内部目录与 Azure AD 集成

如果组织有企业内部目录服务，可以将其与 Azure AD 目录集成。设置目录集成功能（如目录同步或 SSO）的主要好处之一是，在配置了同步操作后，数据现在置备在云存储中，并在那里更新，Azure AD 租户中订阅的所有云服务都可利用这些数据。有各种同步选项，包括：

- 将身份从 Active Directory 同步到 Azure AD
- 将身份和密码哈希从 Active Directory 同步到 Azure AD
- 将身份和密码哈希从 Active Directory 同步到 Azure AD 并启用密码回写

回写功能允许在 Azure AD 中发生更改时，更新现有 Active Directory 身份。在同步身份和密码哈希时，请注意这将创建两个身份，记住这点很重要。管理这些身份及其同步关系是设计云应用程序身份验证方案的关键所在。

目录用户

可以为访问服务的每一个用户直接在目录中创建帐户。还可以管理帐户，或者在不再需要帐户的时候将其删除。默认情况下，用户没有管理员权限，但是可以视情况选择将权限分配给他们。使用管理门户可创建三种类型的用户：

- 目录或组织中的新用户
- 有现有 Microsoft 帐户的用户
- 另一个 Azure AD 目录中的用户

可以在门户中提供有关用户的以下详细信息来创建新用户：

- 名字
- 姓氏
- 显示名称
- 别名
- 角色

创建新用户后，系统会生成临时密码。然后可以用电子邮件将此密码发给用户。首次登录时，系统会提示用户更改临时密码。

外部用户

可以将其他 Azure AD 目录中的用户或者拥有 Microsoft 帐户的用户添加到 Azure AD 目录。这使得外部用户可以与目录中已有的用户协作。如果要在环境中与需要管理目录资源（如应用程序）的用户协作，而又无需这些用户在目录中有帐户和凭据，那么这种情况下，这样的目录内外协作很有用。

- 可以用唯一用户名或使用用户的 Microsoft 帐户将用户添加到目录
 - SSO 门户允许用户以任一种方式（用户名或 Microsoft 帐户）登录
- 在与第三方集成时，可以启用以下两种单一登录支持的其中之一：
 - 用户使用其 Azure AD 帐户登录到第三方服务。
 - 用户以其第三方服务帐户证明身份，信息安全地存储并与其 Azure AD 帐户关联

在将用户从一个目录添加到新目录时，该用户是新目录中的外部用户。一开始，显示名和用户名从用户主目录复制过来，并转印到另一个目录中的外部用户。自那以后，外部用户对象的配置文件属性将完全独立。如果更改主目录中的用户，如更改用户名、添加工作职务等等，这些更改不会传播到另一个目录中的外部用户帐户。

Azure AD 中的应用程序

企业开发人员和 SaaS 提供商可开发能与 Azure AD 集成的商用云服务或 LOB 应用程序，以提供其服务的安全登录和授权。Azure AD 还包含可供用户使用的访问面板，用户可在其中发现自己可访问哪些应用程序。从此面板中，他们可以使用 SSO 访问其应用程序。为了将应用程序或服务与 Azure AD 集成，开发人员必须先使用管理门户向 Azure AD 注册应用程序详细信息。这些步骤与将 SSO 第三方应用程序添加到 Azure AD 实例的步骤相似。

- 可以将现有应用程序（组织正在使用或有意使用 的应用程序）添加到 Azure AD 实例，来使其集成 SSO 体验
- 作为开发人员，可以构建 SaaS 解决方案，并使其支持 Azure AD 集成
- 正在开发的应用程序也可以与 Azure AD 集成，以实现 SSO 支持
- 例如，正在开发中的 ASP.NET 应用程序可使用 ASP.NET 身份和 Azure AD 来为用户提供单一登录体验

单一登录

配置 SSO 可以让组织中的用户使用其 Azure AD 凭据自动登录到任何第三方 SaaS 应用程序。此功能为用户带来了只需记住一个密码的便利性，而且还提高了组织的安全性，因为它使用户只能访问自己的应用程序。Azure AD 可将其身份联合到自定义应用程序、存储自定义应用程序的凭据，或者与第三方 SSO 提供程序集成。

用户设置

用户设置实现了使用 Windows Server Active Directory 或 Azure AD 身份信息，从管理门户内自动在第三方 SaaS 应用程序中设置和取消设置用户帐户。当用户在 Azure AD 中获得其中一个应用程序的权限后，在目标 SaaS 应用程序内会自动创建（设置）一个帐户。当用户在 Azure AD 内被删除或其信息更改时，这些更改也会反映在 SaaS 应用程序中。用户设置可让应用程序自动化身份生命周期管理，并使得管理员能够控制和提供用户帐户在 SaaS 应用程序中的自动化设置和取消设置。

访问面板

Azure AD 中的访问面板为组织提供了一个统一的仪表板。用户可使用单一登录体验，直接从此面板中访问您从 Azure AD 实例内管理的一个或多个应用程序。用户无需 Azure 或 Office 365 订阅即可连接到访问面板。

Azure AD Graph

Graph API 提供了通过 REST API 端点，以编程方式访问 Azure AD 的途径。此 API 可用来存储和检索关于用户的元数据（不属于 Active Directory 中典型用户配置文件的元数据）。

创建、读取、更新和删除(CURD)操作

应用程序可使用 Graph API 来对 Azure AD 实例中的目录对象执行 CRUD 操作。例如，可以使用 Graph API 对用户对象执行以下操作：

- 在指定目录中创建新用户。
- 获取用户的详细属性。

- Azure AD Graph 提供了以编程方式通过 REST API 端点访问目录的途径
- 对 Azure AD 对象执行 CRUD 操作：
 - 用户
 - 组
- 用于访问企业内部 AD 的 ADSI 或 ADO.NET 库的替代接口
- Azure AD Graph API 允许以业务线 (LOB) 应用程序所必需的自定义属性扩展现有对象

- 检查用户的组成员身份。
- 更新用户的扩展属性（配置文件）。
- 禁用或删除用户帐户。

除了用户外，Azure AD 中的组和应用程序也支持类似的操作。若要对特定目录调用 Graph API，必须向 Azure AD 注册应用程序，然后将其配置为允许访问目录。

目录扩展

对于通常未存储在标准 Active Directory 用户配置文件中的每一个用户，很多应用程序需要这些用户的元数据和属性。Graph API 允许先注册扩展属性，然后使用这些属性。例如，如果需要在游戏社交应用程序中存储每个用户的 Xbox Live ID，然后检索这些 ID，那么必须先在目录中注册这个新属性。然后才可以在后续操作使用此属性，因为并非目录中的每一个用户对象都有该属性。

第 3 课 Azure AD 多重身份验证

多重身份验证是 Azure AD 中一个功能，可用来为现有目录帐户提供一层额外的授权。这层授权可以是电话通话、移动代码或自定义应用程序。

课程目标

完成本节后，您可以做到：

- 描述 Azure AD 中的多重身份验证。
- 列出 Azure AD 可用的多重身份验证提供程序。

多重身份验证

多重身份验证是一道额外的安全层，可在用户凭据泄露的情况下，防止应用程序受到未经授权的访问。多重身份验证对最终用户而言，只是提供了额外的身份验证方法，可能包括电话、RSA 密钥或自定义设备。多重身份验证通常需要用户提供两个要素来确定：

- 您知道的要素：密码
- 您持有的要素：受信任的设备（电话、智能手表等等）

多重身份验证的优势在于其多层的方式。如果用户凭据泄露，恶意用户需要有发给同一用户的受信任设备才能入侵应用程序或其数据。通常，如果用户丢失了受信任设备，他们将立即报告，可以将设备解除授权。

Azure AD 直接使用密码作为默认的用户访问凭据。多重身份验证是 Azure AD 中的一项服务，它实现了以前提到过的多重身份验证模式。可以结合 Azure AD 或企业内部目录来使用多重身份验证。身份验证的第二种形式可以是智能电话、支持通话或短消息的电话号码，或者自定义应用程序。在结合 Azure AD 使用多重身份验证服务时，管理员可以专门为每个用户启用多重身份验证。多重身份验证服务支持最多三个授权用作第二种身份验证形式的电话号码。用户还可以选择使用既支持推送通知，又支持一次性口令作为身份验证选项的多重身份验证移动应用程序。

可以对云应用程序或企业内部应用程序启用多重身份验证。

- 除了传统的用户名和密码凭据外，Azure 多重身份验证提供了一道额外的身份验证
- 多重身份验证可用于企业内部应用程序和云应用程序
- 与对 Azure AD 身份启用多重身份验证一样简单

MCT USE ONLY
STUDENT
USE PROHIBITED

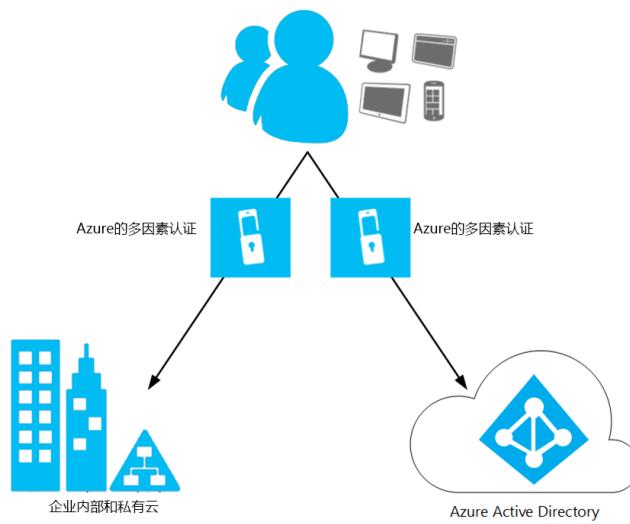


图 12.2: AZURE MFA

Azure 多重身份验证服务器

<http://go.microsoft.com/fwlink/?LinkId=525681>

有软件开发包 (SDK) 可用于将自定义应用程序与 Azure AD 多重身份验证集成。SDK 允许在自定义应用程序的登录过程中，使用多重身份验证电话通话或短消息验证选项。如果您构建的自定义应用程序不重定向到 Azure AD 的登录页，而是有内置的登录表单，那么这种做法会非常有用。

将多重身份验证内置于自定义应用中(SDK)

<http://go.microsoft.com/fwlink/?LinkId=525683>

多重身份验证提供程序

可以结合 Windows Server Active Directory 域服务 (AD DS) 或 Azure AD，使用多重身份验证 (MFA) 服务来帮助保护云应用程序和企业内部应用程序。在使用 Azure AD 登录时，组织中的用户有很多不同的选项可用作向应用程序证明身份的第二种形式。作为管理员，可以控制哪些选项可供最终用户使用。

多重身份验证应用程序

在 Windows Phone、Android 和 iOS 的各个应用程序商店中，可以找到用来与 Azure AD 多重身份验证集成的应用程序。用户下载这些应用程序，并使用安装代码将其激活。在用户登录到应用程序后，会有一条通知推送到他们移动设备上的应用程序。然后用户可以立即准许或拒绝身份验证。Azure AD 多重身份验证还使用身份验证开放标准，并且可配合各种第三方多重身份验证应用程序使用。这些应用程序生成一次性使用的口令，用户在尝试登录后，必须输入此口令。此行为与 RSA 密钥设备非常相似。

- 有三种可用的身份验证选项：
 - 多重身份验证应用程序
 - 针对 Windows Phone、Android 和 iOS
 - 应用程序可向最终用户发送通知，然后用户可证明或拒绝登录请求
 - 应用程序还可提供每次尝试登录时必须配合用户名和密码使用的一次性口令
 - 自动电话通话
 - 短消息

MCT USE ONLY. STUDENT USE PROHIBITED

自动化电话通话和短消息

用户可选择将自动电话通话或短消息功能加入其授权的移动设备。对于电话通话，用户只需接听通话，然后按下手机上的井号 (#) 键即可完成登录过程。对于短消息，用户会收到一个一次性使用的口令，在试图登录后，他们必须输入此口令。

实验 A：将 Azure Active Directory 与事件管理门户集成

场景

尽管 Contoso Events Web 应用程序是公开的，但是“管理”应用程序应该只能由您域中的用户访问。您决定使用 Azure AD 和 ASP.NET 身份来提供此功能。本实验中，您将使用 ASP.NET 身份框架创建新的 ASP.NET 项目，并将该项目与 Azure AD 集成。网站然后将使用组织帐户进行登录。

目标

完成本实验后，您将能够：

- 使用管理门户创建 Azure AD。
- 使用管理门户在 Azure AD 中创建用户。
- 创建将 Azure AD 组织帐户用于安全保护的新 MVC 项目。

预计时间：60 分钟

在开始这个实验前，必须完成第 2 章中的实验 A。在此章实验中，您将使用自己的计算机完成实验，此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称：

vm28532[自定义名称].cloudapp.net: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

练习 1：创建 Azure AD

场景

在此练习中，您将：

- 使用管理门户创建 Azure AD。
- 在新目录中创建全局管理员角色。
- 在新目录中创建标准用户角色。

此练习的主要任务如下：

- 登录到 Azure 管理门户
- 使用管理门户创建目录

MCT USE ONLY. STUDENT USE PROHIBITED

- 在目录中创建全局管理员角色
- 在目录中创建用户角色

► **任务 1: 登录到 Azure 管理门户**

 **注释:** 因为在预览门户网站中没有创建新的 Azure Active Directory 域功能, 所以在这个实验中将使用管理门户网站。

1. 登录管理门户网站(<https://manage.windowsazure.com>)。

► **任务 2: 使用管理门户创建目录**

1. 查看订阅的 Active Director 目录列表。

2. 使用以下信息创建新的目录:

- 名称: **28532**
- 域名: **ad28532[自定义名称]**
- 国家: **选择当前所在地**

► **任务 3: 在目录中创建全局管理员角色**

1. 指向新创建目录的用户列表。

2. 使用以下信息添加新的用户到目录:

- 用户类型: **您的组织中的新用户**
- 用户名: **admin**
- 域: **ad28532[自定义名称]**
- 名字: **Admin**
- 姓氏: **User**
- 显示名称: **Admin User**
- 角色: **Global Administrator**
- 备用电子邮件地址: **example@contoso.com**

 **注释:** 确认记录了创建新用户时生成的临时密码。如果忘记密码, 可以在管理门户网站中使用添加用户按钮旁重置密码按钮来重置密码。

► **任务 4: 在目录中创建用户角色**

1. 使用以下信息添加新用户到目录中:

- 用户类型: **New user in your organization**
- 用户名: **standard**
- 域名: **ad28532[自定义名称]**
- 名字: **Standard**
- 姓氏: **User**
- 显示名称: **Standard User**
- 角色: **User**



注释: 确认记录了创建新用户生成的临时密码。如果忘记密码，可以在管理门户网站中使用添加用户按钮旁重置密码按钮重置密码。

结果: 完成此练习后，您将创建好 Azure AD，并在该目录中填充了用户。

练习 2：保护现有 ASP.NET Web 应用程序

场景

在此练习中，您将：

- 使用 MVC AuthorizeAttribute 保护 Web 应用程序。

此练习的主要任务如下：

- 将 Authorize 属性添加到 HomeController 类
- 调试 Web 应用程序

► **任务 1：将 Authorize 属性添加到 HomeController 类**

1. 浏览至以下位置并打开 **Contoso.Event** 解决方案：
 - 文件位置：Allfiles (F):\Mod12\Labfiles\Starter\Contoso.Events
2. 在 **Contoso.Events.Management.Old** 项目中打开 **HomeController.cs** 文件。
3. 将 **Authorize** 属性添加到 **HomeController** 类中。

► **任务 2：调试 Web 应用程序**

1. 调试解决方案，确认收到一条 401(未经身份验证)的错误信息。
2. 停止调试。

结果: 完成此练习后，您就会使用 MVC 来确保用户在登录后才能访问控制器或操作。

练习 3：将 Azure AD 与 ASP.NET 身份集成

场景

在此练习中，您将：

- 使用 ASP.NET 身份创建新的 ASP.NET MVC 应用程序来进行身份验证和授权。
- 将新的网站与 Azure AD 组织帐户集成。

此练习的主要任务如下：

- 修复 IIS 8.0 Express 安装
- 使用 Azure AD 作为身份提供程序来创建新的管理 Web 应用程序
- 验证“管理”Web 应用程序是否需要使用组织帐户登录

► **任务 1：修复 IIS 8.0 Express 安装**



注释: 当给您的计算机生成映像并使用新用户重建计算机系统的时候，原先随 Visual Studio 2013 Update 3 安装的 IIS 8.0 Express 可能会崩溃。

IIS Express 依赖于用户文档目录中的配置文件。当机器被制作成映像，原用户帐户会丢失。当从映像创建新虚拟机时，新用户帐户会生成。

MCT USE ONLY. STUDENT USE PROHIBITED

修复安装的操作将可以为新用户创建配置文件。

1. 打开运行应用程序。
2. 在运行对话框中输入 **appwiz.cpl**。
3. 在 **Programs and Features** 中，定位到 **IIS 8.0 Express**，然后修复这个安装程序。

► **任务 2：使用 Azure AD 作为身份提供程序来创建新的管理 Web 应用程序**

1. 使用以下信息在 **Administration** 解决方案文件夹中创建一个新的 **Contoso.Events.Management** ASP.NET Web 应用程序项目：
 - 项目名称：**Contoso.Events.Management**
 - 项目位置：**Allfiles (F):\Mod12\Labfiles\Starter\Contoso.Events**
 - 模版：**MVC**
 - 身份验证：**Organizational Accounts**
 - 域：**ad28532[自定义名称].onmicrosoft.com**
 - 使用的验证帐户：**admin@ad28532[自定义名称].onmicrosoft.com**

注释：当新的 ASP.NET Web 应用程序项目和一个 Azure AD 域关联时，必须使用一个有全局管理员权限的帐户。

2. 将 **Contoso.Events.Management** 项目设为启动项目。
3. 在 **Contoso.Events.Management** 项目中添加 **Contoso.Events.Models** 和 **Contoso.Events.ViewModels** 项目。
4. 复制 **Contoso.Events.Management.Old** 项目中的全部文件夹到 **Contoso.Events.Management** 项目。

注释：确认复制的是文件夹而不是单独的文件，并忽略以下文件。

- favicon.ico
- Global.asax
- packages.config
- Web.config

► **任务 3：验证“管理”Web 应用程序是否需要使用组织帐户登录**

1. 调试解决方案。
2. 配置 IIS Express 为信任自签名证书。
3. 确认在访问 **Contoso.Events.Management** web 应用程序之前被要求登录。

结果：完成此练习后，您就会结合 ASP.NET 身份框架来使用 Azure AD 域。

章节复习和作业

本章中，我们介绍了 Azure AD，它可作为很多不同场景下的身份管理解决方案。利用 ACS 方案、第三方 SaaS 集成以及多重身份验证，Azure AD 提供了可用来改进现有企业内部身份解决方案的独特服务。



最佳做法：ASP.NET 身份框架是一种较新的 Web 应用程序保护方法，它有一些胜过成员身份和 Forms 身份验证的优点。

复习问题

问题：在使用 Azure AD ACS 时，为什么应该重新映射来自每个身份提供程序的声明？

第 13 章 维护和监视 Azure 中的 Web 解决方案

目录:

章节概述	13-2
第 1 课: Web 应用程序的部署策略	13-3
第 2 课: 部署 Azure 网站服务	13-6
第 3 课: 部署 Azure 云服务	13-11
第 4 课: 持续集成	13-14
第 5 课: 监视云应用程序	13-16
实验 A: 将 Events Web 应用程序部署到 Azure	13-23
实验 B: 使用中国版 Windows Azure 部署 Events Web Application 到云环境	13-28
章节复习和作业	13-37

章节概述

将应用程序从开发环境移到暂存或生产环境可能是件很难的任务。FrontPage Server Extensions (FPSE)、文件传输协议 (FTP) 以及 Web Deploy 等技术的出现有助于使这种迁移更加轻松。您可以利用现有技术或新的部署方法将云 Web 应用程序发布到 Azure。第 1 节“Azure Web 应用程序的部署策略”列出一般可用来部署应用程序的某些方法。第 2 节“部署 Azure 网站服务”介绍了使用 Web Deploy 发布网站服务的过程。第 3 节“部署 Azure 云服务”描述 Microsoft Azure 云服务的发布过程。第 4 节“持续集成”介绍了使用 Visual Studio Online 工具对网站进行持续部署。第 5 节“监视云应用程序”详细介绍可用来执行 Web 应用程序取证分析的各种工具。



注释: Visual Studio Onlie 工具在中国版 Windows Azure 中暂时不支持。

目标

完成本章后，您可以做到：

- 描述部署 Web 应用程序的策略。
- 描述部署云服务包的过程。
- 描述使用 Web Deploy 部署 Azure 网站服务的过程。
- 描述监视 Azure 中的 Web 应用程序的选项。

第 1 课 Web 应用程序的部署策略

部署 Web 应用程序有很多现有方法，在将应用程序部署到 Azure 时，这些方法仍然适用。理解每种方法之间的区别非常重要，因为这样我们才能为每次项目发行选择最佳的方法。

本节列出并描述部署 Web 应用程序的不同方法。

课程目标

完成本节后，您可以做到：

- 解释选择部署方法时的注意事项。
- 描述如何使用服务包、脚本和 Web Deploy 部署到 Azure。

企业内部部署策略

在使用 Microsoft Visual Studio 完成 Web 应用程序项目的初始开发后，可能需要将应用程序部署到可供应用程序用户访问的目标服务器上。在最简单的情况下，可以将部署界定为只涉及将生成的文件或源文件复制到目标环境。而现实中，很多部署还涉及其他任务，包括：

- 修改随环境而变的配置文件（如 `Web.config`）设置。例如，可能对生产应用程序使用不同的数据库连接字符串。还可以将各种标志或设置更改为适合生产环境的值。
- 将架构或数据填充到将由生产 Web 应用程序使用的数据库。
- 在目标 Web 服务器上配置服务器设置。例如，可以将目标服务器配置为支持特定身份验证方法，或以不同于开发服务器的方式处理错误。
- 安装相应的安全套接字层 (SSL) 证书。
- 修改生产服务器的注册表。
- 将应用程序集安装到生产服务器的文件目录或全局程序集缓存 (GAC)。

部署企业内部应用程序的选项没有限制

- 安装项目
- XCOPY 部署
- ClickOnce
- WebDeploy
- TFS Build 过程
- FrontPage Server Extensions

下面几个部分描述可用来部署企业内部应用程序的不同技术：

安装项目

可以在 Visual Studio 中创建新的安装项目来生成 Windows Installer (.msi) 文件，这些文件用于分发应用程序，以便安装到其他计算机或 Web 服务器上。Web 安装项目创建安装程序，安装程序会在 Web 服务器上安装 Web 应用程序。Web 安装项目的文件安装到 Web 服务器上的虚拟根文件夹。

XCOPY

可以使用 XCOPY 命令行工具将网站项目的文件从开发计算机复制到将托管网站的 Web 服务器。复制的文件可以是源文件（如果没有预编译网站）或者预编译的程序集和相关文件。这些文件是在命令提示符下使用 **XCOPY** 命令复制的，可以手动执行该命令，也可以使用批处理文件自动化批量复制过程。

从命令行使用 XCOPY 部署 ASP.NET 网站

XCOPY 部署

```
Xcopy /I /S c:\inetpub\wwwroot\testapp d:\publicroot\liveapp
```

ClickOnce

ClickOnce 是一种能让您创建基于 Windows 的自更新应用程序的部署技术，这种应用程序只需最少量的用户交互即可安装并运行。Windows Presentation Foundation (WPF)、Windows 窗体和控制台应用程序可使用 ClickOnce 来部署。这些应用程序可部署到网页、文件共享或物理介质。在应用程序安装之后，它们可能视情况检查安装源（文件共享或网页）寻找更新版本，并自动为用户安装这些版本。

Web Deploy

Web Deploy 是 Internet Information Services (IIS) 的扩展，可自动化很多部署任务。此扩展将在本章的后面讨论。

Team Foundation Server (TFS) Build 过程

TFS 中的自定义 Build 定义可用来将 Web 应用程序部署到目标服务器。这些 Build 文件可与其他计划、入门 (gate) 和验证功能结合，以实现持续集成场景。

FrontPage Server Extensions (FPSE)

 **注释：** FPSE 远程协议不是新 Web 应用程序的推荐部署方法。引入本段内容是考虑到现有应用程序可能是遗留下来的老程序的情况。

FPSE 远程协议规定了一组提供文件服务器功能的服务器扩展，这些功能允许网站表示为文件共享。这是使用 WebDAV 协议实现的。可以使用 FPSE 将 Web 应用程序部署到目标服务器。FPSE 使用 HTTP 作为协议，而不是 FTP。

部署到云时的注意事项

在将应用程序部署到 Azure 之类的云托管提供商时，应该考虑很多问题：

可组合性

安装包是一个组件。安装后，安装包是否能应付其所有需求？安装包能否与其他服务或组件安装在一起？如果几个组件需求重叠，会不会造成问题？

幂等性

应该能够在应用程序上多次运行安装程序，而不会给应用程序造成副作用。在安装或卸载应用程序后，应用程序是否不必要地移除了第三方资源的数据？如果以前的安装留有残余，应用程序会不会出现故障？应用程序会不会正确清理其安装？

可跟踪性

如果安装失败，是不是有足够的日志数据可供您快速轻松地确定原因？日志是否安全，并且可轻松获取？如果无法使用远程桌面直接访问虚拟机，那还能不能访问日志？

实体框架中存在这些原则的良好示例。例如，对于幂等性，如果启用 Code First，就必须确保设置了迁移模式。否则，初始值设定项可能在每次安装时删除再重新创建数据库。对于企业内部应用程序，这一点或许可以忽略，但是托管应用程序（如 Microsoft Azure 网站服务或云服务）会在出现更新、执行缩放操作

在部署到 Azure 时，有些事项会变得更加重要

可组合性

幂等性

可跟踪性

或发生配置更改时多次重新安装应用程序包。让实体框架在每次应用程序启动时都销毁数据库，这不是可持续的长期策略。

部署方法

要在 Azure 中部署应用程序，有多种部署选项。在本节前面，您了解了用来将应用程序部署到企业内部服务器的某些策略。很多同样的方法也可用来将应用程序部署到 Azure 服务实例。本章中，我们将探讨最常用的两个选项，云服务包和 Web Deploy。使用 Web Deploy 进行部署的详细信息将在本章的下一节介绍。云服务包将在本章后面的小节中详细介绍。

其他部署选项

Windows PowerShell、Xplat-CLI 和服务管理 REST API 等工具可用来自动化部署操作。虽然可以配合这些自动化工具使用云服务包和 Web Deploy 包等包格式，但是还有其他选项也可用于部署：

- FTP 可用作 Azure 网站服务的部署工具。
- 使用 Windows PowerShell，而不是使用 Visual Studio 可以创建 Node 和 PHP 应用程序的云服务包。
- Microsoft Azure Visual Studio Online 可用来就地编辑 Azure 网站服务，而无需部署现有应用程序。
- Zip 压缩文件夹可用来部署 WebJob。
- Git 和 TFS 可用于连续部署。
- 搜索服务和 DocumentDB 项可使用通过 HTTP POST 请求的 JSON 有效负载来部署。
- 配置管理工具可用于部署和安装自定义应用程序。

远程桌面部署

使用远程桌面也可将应用程序部署到目标计算实例。对于 Microsoft Azure 虚拟机，这是一个通常探讨的选项。对于云服务，这又是一个用于部署的选项。

对云服务来说，使用远程桌面部署应用程序不太理想，并可能导致应用程序一致性问题。云服务允许对角色实例启用远程桌面。即使远程桌面协议 (RDP) 可用于连接到这些实例，并对其进行自定义，但是在下次回收该实例时，这些自定义仍会丢失。

 **注释：**因此建议不要使用远程桌面来将应用程序部署到云服务。如果角色实例回收、云服务升级或者有缩放操作，使用此方法手动部署的内容很有可能丢失。使用 RDP 进行的手动更改不会持久保存到包含云服务包的存储区中。

- 云服务包
 - 与云服务项目一起使用
 - 包含配置数据、元数据和服务定义
 - 指定实例大小和数量
- WebDeploy
 - 在 IIS 中安装 Web 应用程序包的标准化格式
 - 可用来导入或导出现有 Web 应用程序
 - 包含有关 Web 应用程序 IIS 配置的元数据
- 远程桌面
 - 利用现有部署策略

第 2 课 部署 Azure 网站服务

部署到 Azure 网站服务的 ASP.NET Web 应用程序使用 Web Deploy 协议和包。Web Deploy 方法的发布向导集成了 Entity Framework Code First 迁移。通过使用 Web Deploy 和实体框架，可以在部署应用程序并自定义目标数据库时，自定义应用程序的行为。

本节描述部署到 Azure 网站实例的 ASP.NET Web 应用程序的发布过程。

课程目标

完成本节后，您可以做到：

- 描述 Web Deploy 和 IIS 之间的关系。
- 描述用于发布 Web 应用程序的 Web.config 转换选项。
- 描述在 Visual Studio 发布向导中，如何使用 Web Deploy 配置 Entity Framework Code First。

结合 IIS 使用 Web Deploy

IIS 是可选择安装在 Windows 服务器或 Windows 计算机上的 Web 服务器软件。多年以来，IIS 发布了多个版本，其中，IIS 7 是 Web 服务器最大的改变之一。出现 IIS 7 后，Web 服务器重新设计为模块化结构。这使得核心 Web 托管功能在其设计中非常轻小、非常高效。这种核心 Web 服务器因其轻小和高效而可运行在很多不同的环境中。其他功能可作为模块启用，或者作为核心托管功能的扩展安装到 IIS。

Web Deploy 是 IIS 的扩展，它主要为 IIS 中托管的 Web 应用程序自定义常见操作和部署任务。

Web Deploy 允许将托管在 IIS 网站或虚拟目录中的 Web 应用程序从一个服务器实例迁移到另一个服务器实例。这是使用标准包格式来实现的，这种格式可从任何现有的 IIS 安装中导出，并导入另一个 IIS 安装。这种包包含应用程序文件（源文件、二进制文件等等）、IIS 的配置值，以及应用程序设置和连接字符串等其他设置。在从现有 IIS 网站生成包时，该网站实例的配置与 Web 应用程序的文件（位于其文件目录中）一起存储在包中。在此包导入 IIS 网站时，其配置值将立即生效，文件将复制到目标网站的文件目录中。目录服务器上的用户或脚本可在导入过程中或导入操作后，修改或不修改配置值。

您可以使用 Web Deploy 启用远程部署，也可以不启用。此选项将创建一个远程端点，其他用户可使用此端点将 Web 应用程序部署到指定网站，而无需直接访问该服务器。

Visual Studio 允许从 ASP.NET Web 应用程序项目生成 Web Deploy 包。此包是一个压缩 (.zip) 文件，其中包含 IIS 网站所必需的配置，以及 ASP.NET Web 应用程序生成过程的输出结果。Visual Studio 还允许执行一键发布，这种发布方式自动将 Web Deploy 包发布到远程端点。

Azure 网站服务使用 Web Deploy 包格式支持部署。每个 Azure 网站服务实例的 Kudu 安装会公开一个 Web Deploy 端点。可以从 Azure 管理门户或预览门户下载 .publishprofile 文件，并使用该文件中的设置来将 Web 应用程序部署到 Azure 网站服务实例。Visual Studio 还可使用发布基架来自动化此过程。在发布 ASP.NET Web 应用程序时，请选择 Azure 网站服务选项，然后创建新实例或选择现有实例。这跟手动下载 .publishprofile 文件，然后导入该文件效果相同。导入这些设置后，Visual Studio 将使用 Web Deploy 协议将 Web 应用程序部署到 Azure 网站服务实例。

- WebDeploy 可让您同步或异步将应用程序发布到 IIS 实例
- WebDeploy 确定配置、内容和证书，然后创建包含这些信息的包以用于部署
- 可以指定网站，并将该网站复制到目标网站
- 还可以指定网站，并从网站导出 WebDeploy 包
- 可以获取 WebDeploy 包，并将其安装在目标 IIS 网站中

转换 Web.config 设置

大多数 Web 应用程序框架都支持在特定文件中定义应用程序设置。ASP.NET 应用程序专门使用 Web.config 文件来存储应用程序的设置。在将应用程序部署到生产服务器时，通常都要修改配置设置。可能修改的某些内容包括：

- 将连接字符串修改为使用生产 SQL Server 实例。
- 在生产 Web 服务器上禁用调试。
- 将应用程序更改为在生产环境中显示特定错误网页。
- 在配置文件中移除或加密敏感信息。

- Web.config 转换可用来根据每种环境更改 Web 应用程序的配置设置
 - 对于每个 Build 定义（即暂存、发布和调试）会创建一个 XML 转换文件
 - 默认情况下，在生成要部署到 Windows Azure 的项目时，会使用发布定义
 - 在发布过程期间可更改 Build

应用程序设置还可能因环境而异。例如，开发人员在开发应用程序功能时，可能使用 SQL Server Express LocalDB。测试团队可能使用托管在企业内部计算机上的共享 SQL Server 实例。生产应用程序可能使用 Microsoft Azure SQL 数据库实例。大多数情况下，您必须更改配置文件。虽然这可以手动完成，但是很多组织使用 Build 工具来自动化此过程。

对于 ASP.NET 应用程序项目，Build 工具可自动化针对每个 Build 定义修改配置文件的过程。这是使用称为 Web.config 转换的功能来完成的。这些转换是使用特定架构来执行的，这种架构列出了一组定位符和替换符。这些替换符与 Build 定义关联，在使用该定义触发 Build 时，这些替换符将生效。使用此功能，可以将一组配置设置指定为用于调试（开发）Build，而将另一组配置设置指定为用于发布（生产）Build。

转换文件列出的一组转换在 Build 过程中应用到原始 Web.config 文件。

Web.Release.config 示例文件

```
<?xml version="1.0"?>
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <connectionStrings>
    <add name="AppDatabase"
connectionString="Data Source=tcp:azuresvr.database.windows.net,1433;Initial Catalog=azuredb; User Id=username@servername;Password=password;Encrypt=True;" xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>
  </connectionStrings>
  <system.web>
    <customErrors defaultRedirect="Error.aspx" mode="RemoteOnly" xdt:Transform="Replace">
      <error statusCode="500" redirect="ServerError.aspx"/>
    </customErrors>
  </system.web>
</configuration>
```

SlowCheetah Visual Studio 扩展可用来在非 .NET Web 应用程序项目的 Visual Studio 项目中获得类似的功能。

SlowCheetah - XML 转换

<http://go.microsoft.com/fwlink/?LinkId=525684>

设置 Web.config 转换的方法是创建转换文件，并在文件中使用特殊语法指定 XML 转换。可以为每个 Build 配置定义不同的文件。默认情况下，在创建新的 ASP.NET 项目时，系统会为“调试”和“发布”生成定义自动创建转换文件。

Visual Studio 解决方案资源管理器窗格自动将 Web.config 转换文件分组在 Web.config 文件下。

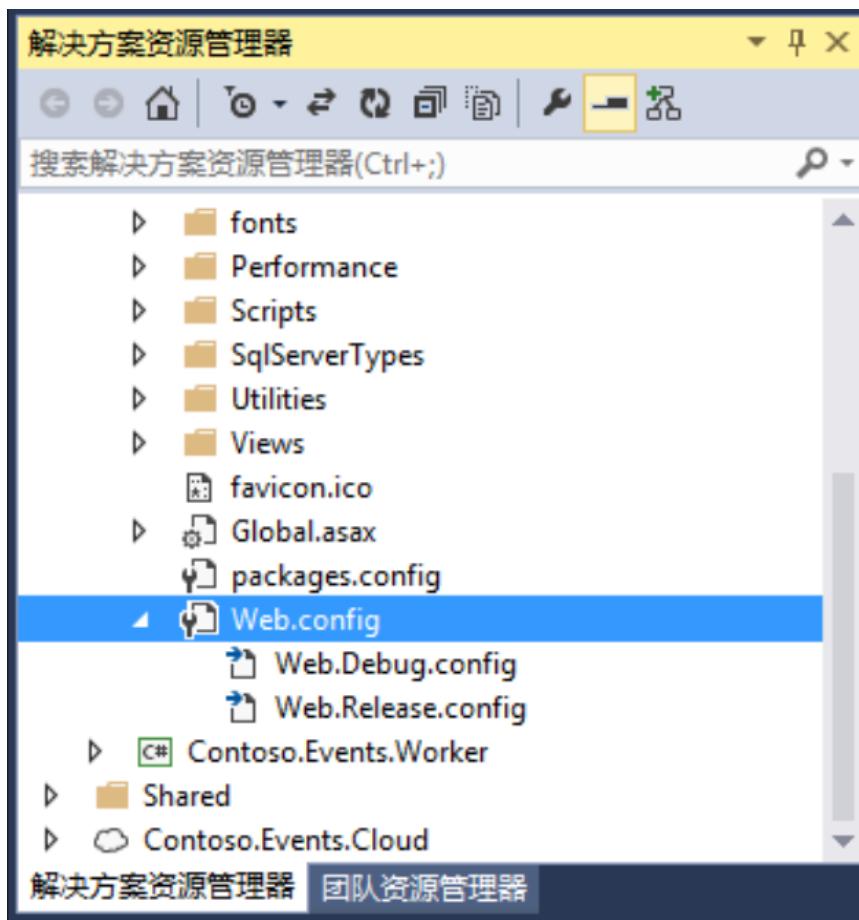


图 13.1: WEB.CONFIG 转换文件

转换文件是一种 XML 文件，其中列出了针对某种 Build 配置而应用到基础配置文件的转换。

源 Web.config 文件

Web.config

```
<connectionStrings>
  <add name="ApplicationServices"
       connectionString="Server=Test;Database=AppDB;Trusted_Connection=True" />
</connectionStrings>
```

转换 Web.Release.config 文件

Web.Release.config

```
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <connectionStrings>
    <add name="ApplicationServices"
         connectionString="Server=Production;Database=AppDB;Trusted_Connection=True" />
    <xdt:Transform="Replace" xdt:Locator="Match(name)" />
  </connectionStrings>
</configuration>
```

发布 Build 后得到的 Web.config 文件

Build 完成后的 Web.config

```
<connectionStrings>
  <add name="ApplicationServices"
       connectionString="Server=Production;Database=AppDB;Trusted_Connection=True" />
```

```
</connectionStrings>
```

配合 Web Deploy 使用 Entity Framework Code First

Entity Framework Code First

实体框架 (Entity Framework) 是一种对象关系映射 (ORM) 框架，它允许 .NET 开发人员使用他们熟悉的类和对象来处理关系数据。实体框架自带的类能处理绝大部分数据访问逻辑，这样应用程序就可以集中处理业务逻辑，而不用顾及传统上所必需的大量数据访问类。Entity Framework Code First 接受现有或新的 POCO（普通旧 CLR 对象）类，并将其与现有数据库或新数据库一起使用。这意味着，现有类无需更改，即可轻松与新数据库或现有数据库一起使用。要使用 Entity Framework Code First，需要定义一个继承

- Code First 初始值设定项和迁移可用来在更改架构时更新数据库
- 在使用 Web Deploy 发布 Web 应用程序时，可以启用 Code First 迁移
 - 设置使用 MigrateDatabaseToLatestVersion 初始值设定项

DbContext 类的类。在此类中，定义属性来表示表。**DbContext** 类可用来映射现有数据库或新数据库。Entity Framework Code First 提供了初始化和迁移功能来更新和更改数据存储，以便符合模型中的更改。

System.Data.Entity.DbContext

<http://go.microsoft.com/fwlink/?LinkId=525685>

含 Categories 和 Products 表的示例 Product 数据库。

DbContext 实现

```
public class ProductContext : DbContext
{
    public DbSet<Category> Categories { get; set; }
    public DbSet<Product> Products { get; set; }
}
```

Entity Framework Code First 初始值设定项

数据库初始值设定项包含的逻辑用来确保数据库设置正确。实体框架包含基本的初始值设定项，包括 **CreateDatabaseIfNotExists**、**DropCreateDatabaseWhenModelChanges** 和 **DropCreateDatabaseAlways**。在创建 **DbContext** 的新实例时，这些初始值设定项控制实体框架的行为。某些初始值设定项包含一个 **Seed** 方法，该方法允许在初始化时将种子数据添加到数据库。对于特定应用场景，还可以创建自定义初始值设定项。

Entity Framework Code First 迁移

Entity Framework Code First 迁移是一系列命令，这些命令能让您根据对本地模型所作的更改自动化更新数据库。这可使用 **Add-Migration** 之类的手动命令来完成，该命令将在项目中创建一个文件夹，文件夹中包含表示数据库更改的类。然后可使用 **Update-Database** 命令将这些迁移应用到数据库。

表示新字段 (**MiddleName**) 添加到模型的迁移类。

DbMigration 实现

```
public partial class AddCustomerMiddleName : DbMigration
{
    public override void Up()
    {
```

```
    AddColumn("dbo.Customers", "MiddleName", c => c.String());
}

public override void Down()
{
    DropColumn("dbo.Customers", "MiddleName");
}
}
```

如果是部署应用程序，您可能希望应用程序在启动时自动应用所有更改来升级数据库。

MigrateDatabaseToLatestVersion 数据库初始值设定项包含自动将挂起的迁移应用到目标数据库的逻辑。

 [System.Data.Entity.MigrateDatabaseToLatestVersion<>](#)

<http://go.microsoft.com/fwlink/?LinkId=525686>

Web Deploy 和 SQL 数据库

在部署使用 SQL Server 数据库的 Web 应用程序时，可能还必须传播数据结构和/或数据。Visual Studio 可以自动创建脚本 (.sql 文件) 在目标数据库中执行这种传播，这些脚本可包含在 Web 包中。还可以包含自定义 SQL Server 脚本，并指定脚本的运行顺序。在这些包安装时，Web 部署将在目标服务器上运行这些脚本。

第 3 课 部署 Azure 云服务

在将应用程序部署到云服务实例时，Azure 云服务项目会使用一种特殊类型的包。此包包含每个角色的配置设置，还有实例数量、虚拟机大小，以及其他云服务设置。

本节描述云服务项目的发布过程。

课程目标

完成本节后，您可以做到：

- 对云环境和本地环境使用不同的配置设置。
- 描述云服务的发布过程。
- 解释如何使用云服务发布向导。

云服务包

在部署云服务前，必须先根据应用程序代码和云服务配置文件 (.cscfg) 创建云服务包 (.cspkg)。每个云服务包都包含应用程序文件和配置。服务配置文件提供配置设置。

在部署云服务时，两个文件是必不可少的：

- 配置文件 (.cscfg)
- 二进制包 (.cspkg)

在生成新的云服务项目时，这些文件可使用 **Visual Studio** 或 **Eclipse** 来生成。这些包也可以使用 **Publish-AzureServiceProject** Windows PowerShell 活动来创建。

Publish-AzureServiceProject 活动创建云服务包，并将其部署到目标云服务实例。-PackageOnly 参数创建包，但不部署包。

PublishAzureServiceProject 活动

```
Publish-AzureServiceProject -PackageOnly
Publish-AzureServiceProject -ServiceName NodeHelloWorld -Location "East US" -Launch
```

云服务包 (.cspkg)

此压缩文件夹包含应用程序的二进制文件以及云服务和角色的定义。如果此包必需任何更改，都需要重新部署到云服务实例。此包包含 **ServiceDefinition.csdef** 文件。该服务定义文件定义云服务的运行时设置，包括所需的角色、端点以及虚拟机大小。当角色运行时，此文件中存储的任何数据都不可更改。

云服务配置文件 (.cscfg)

云服务配置文件是单独的文件，用来配置角色可运行的实例数，以及为角色定义的设置值。在角色运行时，存储在此文件中的数据可更改。管理员可以从开发人员那里获得云服务包和配置文件、将配置文件中的设置修改为适合生产环境的值，然后使用 Azure 管理门户或 Windows PowerShell 部署包和配置文件。

对于正在运行的云服务实例，云服务配置文件还可更新。管理门户提供了上传新配置文件的接口。根据对配置文件所作的更改，角色实例有可能可回收。

手动更新云服务配置文件



图 13.2：配置上传提示

云配置设置

云服务配置文件包含一系列可在应用程序中使用的设置。然后，可以使用自动化脚本或管理门户更新这些设置，而无需重新部署应用程序。

适用于流行开发语言的客户端库将允许应用程序读取云服务配置设置。

有适用于云服务的 .NET 库，并且此库包含配置管理器实现。

CloudConfigurationManager

```
string defaultSection =
```

```
CloudConfigurationManager.GetSetting("DefaultSection");
```

- 每个云服务项目都有本地和云 cscfg 文件
 - 包含每个角色的实例计数和每个角色的自定义设置
 - 可以将实例数量修改为在云中使用其他数量
 - 配置设置也可更改为其他值：
 - 本地 – 开发存储
 - 云 – Azure 存储帐户

Visual Studio 云服务项目

在 Visual Studio 中，在创建云服务项目时，系统会为您创建两个 .cscfg 文件：

- [项目名称].Local.cscfg
- [项目名称].Cloud.cscfg

在将云服务部署到本地存储或计算模拟器时，会使用本地配置文件。在打包云服务或将包直接部署到正在 Azure 中运行的云服务实例时，将使用云配置文件。在部署包和配置之前，如果操作人员或管理人员希望更改应用程序的配置设置，他们可以视情况修改配置文件。

云服务的示例 ServiceConfiguration 文件

ServiceConfiguration

```
<ServiceConfiguration serviceName="AzureCloudService4">
  <osFamily>4</osFamily>
  <osVersion>*</osVersion>
  <schemaVersion>2014-06.2.4</schemaVersion>
  <Role name="WorkerRole1">
```

```
<Instances count="3" />
<ConfigurationSettings>
    <Setting name="QueueName" value="RequestSource" />
</ConfigurationSettings>
</Role>
</ServiceConfiguration>
```

可以在 Visual Studio 中使用特殊的编辑器配置 Azure 云服务。

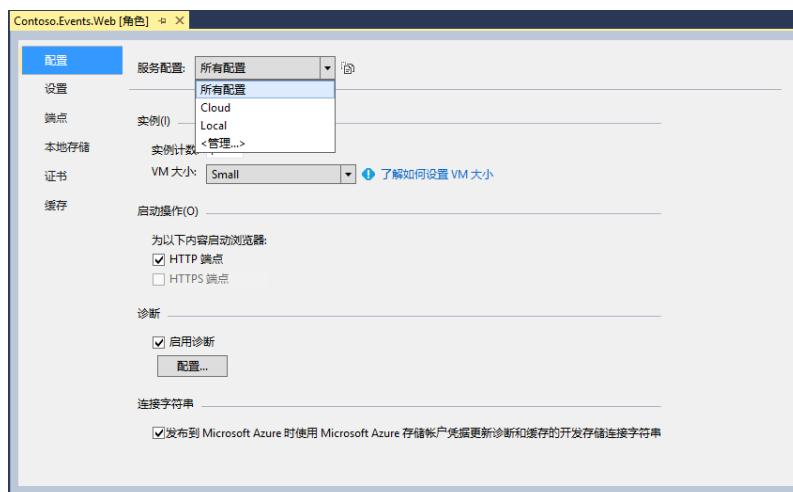


图 13.3：云服务配置

第 4 课 持续集成

源代码管理提供程序可与 Azure 网站服务和云服务集成在一起使用。在 Visual Studio Online 中，可以使用仪表板和报告来深入了解应用程序的行为和状态。Visual Studio Online 和其他源代码管理提供程序还可用于将 Web 应用程序持续部署到 Azure 网站服务实例。

本节介绍 Visual Studio Online 服务，以及如何将该服务与其他源代码管理提供程序一起，用于将最新生成的 Web 应用程序持续部署到 Azure 网站服务。

 **注释:** **Visual Studio Online 服务**在中国版 Windows Azure 中暂时不支持。

课程目标

完成本节后，您可以做到：

- 描述 Visual Studio Online 服务。
- 使用源代码管理提供程序将生成的版本部署到 Azure 网站。

Visual Studio Online

Visual Studio Online 是托管在云中的托管源代码管理和应用程序生命周期管理工具。Visual Studio Online 包含类似下面的功能：

- 负载测试
- 生成代理
- 源代码管理存储库
- 敏捷项目管理和仪表板
- 协作工具

Visual Studio Online 是基于托管 Team Foundation Server 中提供的功能。Visual Studio Online 可连接到现有集成开发环境 (IDE)，如 Visual Studio、Eclipse、Xcode 和第三方 Git 客户端。

- Visual Studio Online 提供了有着增强的仪表板、跟踪功能和持续集成支持的托管源代码管理系统。
 - Git 和 Team Foundation 版本控制皆可使用
 - 项目模板适用于 Scrum、Kanban 或其他敏捷项目方法

网站服务和持续部署

Azure 网站服务可配置为以自动方式从源代码管理存储库进行部署。

Azure 网站服务可使用很多源代码管理存储库：

- Visual Studio Online
- GitHub
- Dropbox
- Bitbucket
- CodePlex
- 托管在硬件上的外部存储库

- 将最新的源代码自动部署到 Azure
 - 兼容 Git、Mercurial、TFS 或其他系统运行
- 每次成功推送（或签入）时，会创建一条部署代码的定义/规则
- 部署在门户中可见
 - 可以回滚到以前的提交（或变更集）

在配置持续交付时，系统会创建一条规则（或 Build 定义），每当推送或签入成功完成时，该规则会向部署触发器 URL 发送一条 HTTP 请求。此 URL 是托管在 KUDU 中的特定目标 URL，您选择的源代码管理或生成引擎可将应用程序部署到该 URL。部署完成后，这些部署最终将作为列表显示在门户中。

在门户中可找到云服务和网站的部署历史记录。



图 13.4：部署历史记录

如果活跃部署出现问题，网站可回滚到前次提交或变更集。

网站槽

在将应用程序部署到 Azure 网站服务时，可以部署到单独的部署槽，而不是默认的生产槽。单独的槽是属于自己的主机名和 URL 的实时站点。此选项只对标准层的 Web 宿主计划提供。可以在两个部署槽之间切换站点和站点配置，包括生产槽。可以使用持续交付将最新版的源代码部署到暂存槽，然后使用手动交换将代码从暂存槽推送到生产槽。

使用部署槽有很多优点。可以先在暂存部署槽中验证网站更改，然后再将其交换到生产槽。先将站点部署到暂存槽，再将其交换到生产槽，这样做将确保槽中的所有实例先“热身”，然后再交换到生产槽。这消除了部署站点时的停机时间。流量重定向是无缝的，不会有请求因为交换操作而被丢弃。交换后，有先前暂存站点的槽现在会有先前的生产站点。如果交换到生产槽的更改不符合预期，可以立即执行同样的交换来恢复上次已知良好的站点。

MCT USE ONLY. STUDENT USE PROHIBITED

第 5 课 监视云应用程序

在 Web 应用程序部署到云后，有很多选项可用于调试应用程序和监视应用程序日志。还可以获取有错的云服务的 IntelliTrace 日志。

本节描述可用于监视托管在 Azure 中的 Web 应用程序的不同选项。

课程目标

完成本节后，您可以做到：

- 使用服务日志监视已部署应用程序的行为。
- 在 Visual Studio 中查看网站服务的跟踪输出。
- 查看网站服务或云服务的内置指标。
- 使用 IntelliTrace 监视云服务。

服务日志

Microsoft Azure Diagnostics 是多个 Azure 扩展，可用来从 Azure 中运行的辅助角色、Web 角色或虚拟机收集诊断遥测数据。遥测数据存储在 Azure 存储帐户中，可用于调试和故障诊断、测量性能、监视资源使用情况、流量分析和容量规划，以及审核。

在创建辅助角色时，Visual Studio 自动将 Diagnostics 融入解决方案中。在云服务配置文件中找到诊断连接字符串就能发觉这一点。

含 Diagnostics 连接字符串的示例云服务配置。

- ASP.NET 中的跟踪依然是记录有关应用程序及其执行路径的强大方法。
- 在 Azure 中，可以使用 Azure 诊断专用跟踪侦听器
 - 诊断连接字符串用来确定将存储跟踪日志的存储帐户
 - 默认情况下，存储表用于存储跟踪日志

Diagnostics 连接字符串

```
<ServiceConfiguration serviceName="AzureCloudService"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration"
  osFamily="4" osVersion="*" schemaVersion="2014-06.2.4">
  <Role name="WebRole1">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
        value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
</ServiceConfiguration>
```

然后可以使用应用程序框架的现有方法来检测代码。

Diagnostics 收集遥测数据的超集，包括：

- **IIS 日志。**有关 IIS 网站的信息。
- **Azure Diagnostic 基础结构日志。**有关 Diagnostics 的信息。
- **IIS 失败请求日志。**有关发给 IIS 站点或应用程序的失败请求的信息。
- **Windows 事件日志。**发到 Windows 事件日志系统的信息。
- **性能计数器。**操作系统和自定义性能计数器。

- **故障转储。**在应用程序崩溃时有关进程状态的信息。
- **自定义错误日志。**应用程序或服务创建的日志。
- **.NET EventSource。**代码使用 .NET EventSource 类生成的事件。

收集的日志和消息然后存储到 Diagnostics 连接字符串中指定的 Azure 存储帐户下。还可以在 Azure 中运行的虚拟机上启用 Diagnostics（可选）。

.NET 跟踪侦听器

在 .NET 中，可以使用 .NET Trace 类和 TraceListener 类来检测自定义应用程序。在应用程序代码里，可使用 Trace 类来跟踪消息。这些跟踪消息随后将由侦听器接收。跟踪侦听器收集、存储跟踪消息，并将其路由到目标。在 Azure 中，DiagnosticMonitorTraceListener 类可用来将所有跟踪消息路由到存储机制。



Microsoft.WindowsAzure.Diagnostics.DiagnosticMonitorTraceListener

<http://go.microsoft.com/fwlink/?LinkId=525687>

查看 Azure 中的跟踪输出

Azure 网站服务有两种类型的诊断信息可用于观察应用程序的行为，或者排查运行中应用程序实例的错误。站点诊断是作为托管平台的 Azure 网站服务实例的一系列日志，它们与 IIS 日志非常相似。应用程序诊断是由应用程序代码生成的自定义诊断日志。

站点诊断

利用站点诊断，可以收集以下数据：

- **详细错误日志。**这些日志包含指出失败的所有 HTTP 状态码响应的详细信息。这些状态码通常在 HTTP 状态码范围 400（客户端错误）到 500（服务器错误）之间。在确定 Web 应用程序错误代码的原因时，这些日志非常有用。通常，如果应用程序使用自定义 HTTP 错误页，可以使用此日志来确定错误根源和详细信息，因为最终用户看不到这些信息。
- **失败请求跟踪。**这是传统的 IIS 日志，提供了所有失败请求的详细信息。此日志包含请求所涉及的所有 IIS 模块的跟踪，并包含每个模块处理请求的时间长度。如果存在服务器性能问题，此日志对于确定请求失败的表现和原因非常有用。
- **Web 服务器日志。**此日志使用标准化的 W3C 扩展日志文件格式。它包含网站服务实例的所有 HTTP 事务，并且可用于测量，以生成站点指标。

应用程序诊断

应用程序诊断发出在 Web 应用程序代码中捕获的日志数据的结果。.NET 应用程序可使用（可选）**System.Diagnostics.Trace** 类将事件记录到应用程序日志中。然后 **TraceWriter** 实例用于将这些日志条目转到相应的目标。与只使用来自主机的日志（站点诊断）相比，使用这些诊断数据可以更加自定义、更深入的方式排查应用程序故障。在用户路径导致异常情况时，应用程序日志在确定用户路径时最为有用。



System.Diagnostics.Trace

<http://go.microsoft.com/fwlink/?LinkId=525688>



配置诊断

管理门户和预览门户可用来管理 Azure 网站服务实例的诊断。对于每个网站服务实例页，管理门户包含“配置”选项卡。预览门户在每个网站服务实例边栏选项卡的“必备”部分内，包含指向诊断配置边栏选项卡的链接。找到“配置”选项卡后，可以单独启用不同的服务器日志选项，或者启用应用程序日志。

Visual Studio 中的服务器资源管理器还支持为 Azure 网站服务实例配置诊断。您还可以视情况配置为将日志存储在文件系统中，或者使用表或 Blob 将日志存储在存储帐户下。

 **注释：**对于生成大量日志的应用程序，建议将日志存储在 Azure 存储 Blob 中。这是因为，从表实例收集所有记录是效率低而开销高的操作。此操作通常称为表扫描。如果应用程序扩展到多个负载平衡的实例，则不建议将日志只存储在文件存储中。这是因为，在执行 Web 应用程序的容量规划或性能调优时，您可能必须查看跨多个实例的日志。

可以在 Visual Studio 中，使用服务器资源管理器查看和修改 Azure 网站服务的配置设置。



图 13.5: VISUAL STUDIO 网站服务诊断配置

日志流

Azure 网站服务可将日志直接流式传送到 Visual Studio 控制台窗口。在需要查看 Web 应用程序的实时应用程序日志时，此功能非常有用。实时日志限制为您在配置中指定的应用程序日志级别。如果想使用日志流功能查看更详细的日志，就必须在网站服务实例的设置中更改此级别。

网站指标

网站服务和云服务通过“监视”管理页提供了监视功能。“监视”管理页提供了网站服务实例的特选性能统计信息。默认情况下，每个服务角色启用了最低限度的监视功能，但是，可以为存储在 Azure 存储帐户中的更详细的日志配置监视。

管理门户中的监视显示内容高度可配置。您可以选择要在“监视”页的指标列表中监视的指标，还可以选择要在“监视”页和仪表板上的指标图表中绘制哪些指标。

选定的指标将在指标表中提供。

- 托管在 Azure 中的网站服务的指标分以下几类：
 - CPU 时间
 - 请求
 - 数据
 - 输出
 - 输入
 - HTTP
 - 客户端错误
 - 服务器错误
 - 成功
 - 重定向
 - 401、403、404、406 错误



图 13.6：指标显示

可按以下方式自定义指标图表：

- 可以显示绝对值（显示 y 轴）和相对值（相对于最终值）
- 范围可更改为 1 小时（默认）、24 小时或 7 天。

可以在图表中增删指标。

对云服务使用 IntelliTrace

IntelliTrace 可记录 Azure 中运行的云服务角色实例的详细调试元数据和历史记录。IntelliTrace 日志是在 Visual Studio 中单步调试代码的极佳方法，通过这种调试，可以确定 Azure 中托管的角色所发生问题的根本原因。IntelliTrace 记录 Azure 中运行的应用程序的执行步骤和环境数据。在本地，可以使用 Visual Studio 重放记录的 IntelliTrace 数据，在发生错误的执行时间点调试应用程序代码。或者，也可以直接在 Azure 中远程调试这些云服务。如果使用不包含调试符号的生成（如使用发布生成定义创建的生成），远程调试和 IntelliTrace 日志的准确性可能欠佳。

- 可以在云服务发布向导中启用 IntelliTrace
- 可以从云项目收集 IntelliTrace 数据，以确定 Web 应用程序错误的根本原因
 - 数据将允许您使用历史调试按钮来查看代码执行特定时间点的数据
- 此外还有 IntelliTrace 事件和调用堆栈数据可用

为了对 Azure 应用程序启用 IntelliTrace，必须从 Visual Studio Azure 项目创建并发布应用程序。必须先为 Azure 应用程序配置 IntelliTrace，然后再将其发布到 Azure。

可以从服务器资源管理器中的“云服务”节点，下载角色实例的 IntelliTrace 日志。这要求本地计算机上安装了 Azure SDK。若要下载日志：

1. 站点“云服务”节点，直至找到所要的实例。
2. 打开此实例的快捷菜单，然后单击**查看 IntelliTrace 日志**。

IntelliTrace 文件随即下载到本地计算机上某个目录的文件内。

每次请求 IntelliTrace 日志时，都会创建一个新的快照。

可以使用 Visual Studio 中的服务器资源管理器访问 IntelliTrace 文件。

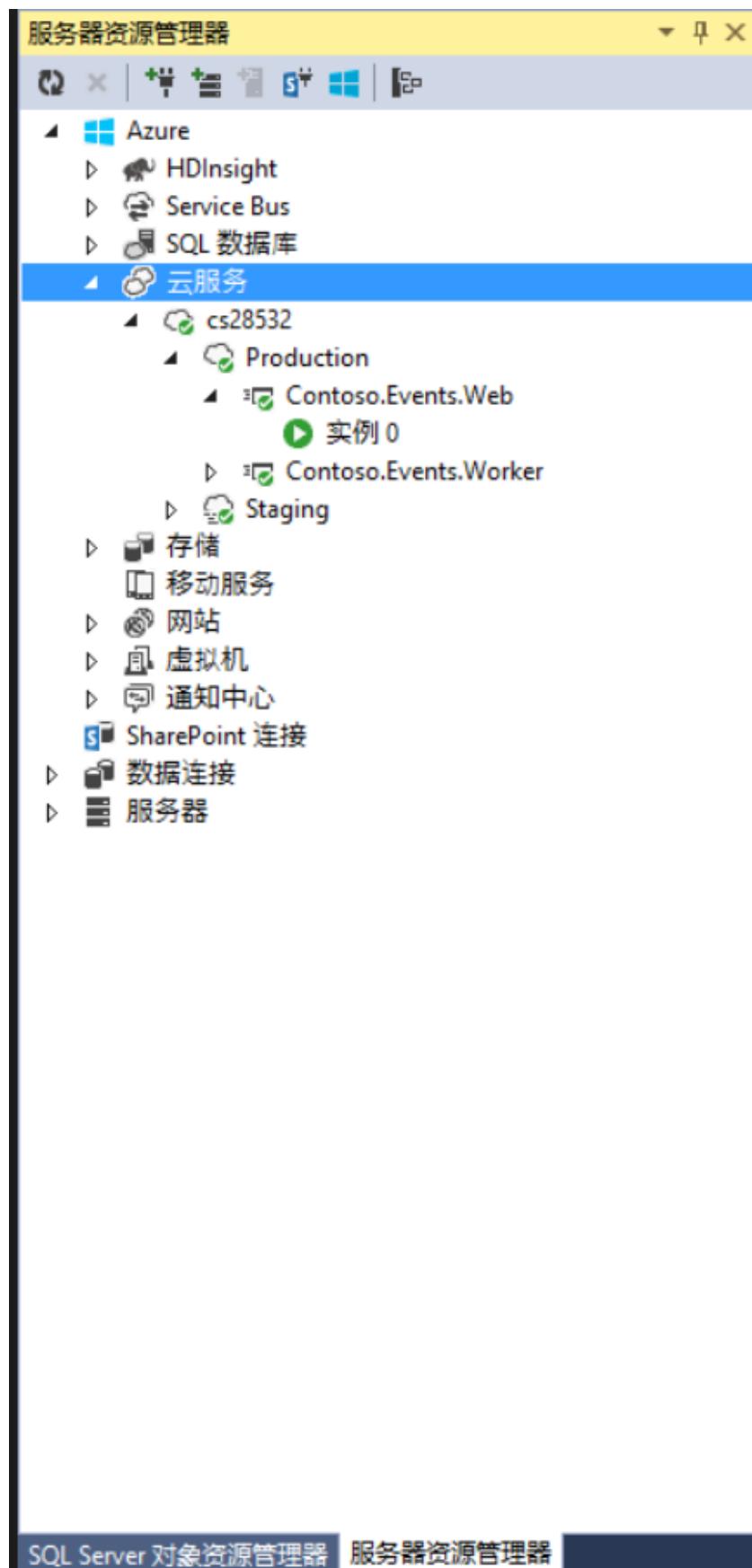


图 13.7：服务器资源管理器

MCT USE ONLY. STUDENT USE PROHIBITED

Visual Studio 在 **Azure** 活动日志窗格中显示下载操作的进度。

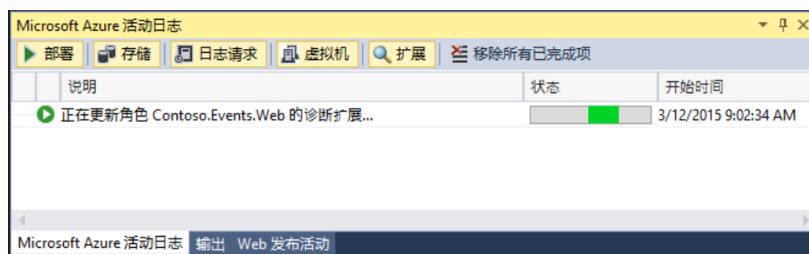


图 13.8: AZURE 活动日志

下载日志后，它们将自动在 Visual Studio 中打开。IntelliTrace 日志可能包含框架生成，随后处理掉的异常。内部框架代码在角色启动时生成这些异常，所以您可以放心忽略它们。

实验 A：将 Events Web 应用程序部署到 Azure

场景

现在，您已经完成了 Contoso Events Web 应用程序的绝大部分开发工作，接下来该开始规划在生产环境中展开。作为 IT 管理流程的一部分，您需要记录将应用程序发布到 Azure 所需要的步骤。为了支持这一目标，您需要将“管理”专用 Web 应用程序和 Events 公共 Web 应用程序部署到 Azure 帐户。

 **注释：**本章为您提供了两套实验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成实验 A；如果您拿到的是中国版 Windows Azure 帐户，请转到实验 B。具体请咨询培训讲师，并在讲师的指导下完成实验。

目标

完成本实验后，您将能够：

- 修改 Web.config 转换文件。
- 将云服务项目部署到 Azure。
- 将 ASP.NET Web 应用程序项目部署到 Azure 网站服务。
- 查看 Azure 网站服务的流日志。

实验设置

预计时间：60 分钟

在开始这个实验之前，必须完成第 2 章的实验 A。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称。

vm28532[自定义名称].cloudapp.net: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的 RDP 端口。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**。等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您收到的密码可以从培训提供商那里登录到 Azure 门户。您将在本课程的所有实验中使用这些密码和 Azure 帐户。

本章为您提供了两套实验方案，如果您拿到的是国际版 Microsoft Azure 帐户，请继续完成实验 A；如果您拿到的是中国版 Windows Azure 帐户，请转到实验 B。具体请咨询培训讲师，并在讲师的指导下完成实验。

MCT USE ONLY. STUDENT USE PROHIBITED

练习 1：创建用于部署的目标 Azure 服务

场景

在此练习中，您将：

- 创建 SQL 数据库实例。
- 创建 Service Bus 实例。
- 创建存储帐户。
- 确定 Azure 服务的连接字符串。

此练习的主要任务如下：

1. 登录到 Azure 预览门户
2. 创建 SQL 数据库实例
3. 创建存储实例
4. 切换到 Azure 管理门户
5. 创建 Service Bus 名称空间

► 任务 1：登录到 Azure 预览门户

1. 登录到 Azure 预览门户网站 <https://portal.azure.com>。

► 任务 2：创建 SQL 数据库实例

1. 查看 SQL 数据库列表。
2. 通过以下信息创建 SQL 数据库：
 - 名称： **dp28532**[自定义名称]
 - 定价层： **Basic**
 - 服务器名： **sp28532**[自定义名称]
 - 服务器管理登录： **testuser**
 - 服务器管理密码： **TestPa\$\$w0rd**
 - 位置： 选择最接近你所在位置的区域
3. 访问新创建的 **SQL** 数据库。
4. 记录 **SQL** 数据库中 **ADO.NET** 的连接字符串，然后将字符串中的{**your_password_here**}替换为 **TestPa\$\$w0rd**。

► 任务 3：创建存储实例

1. 查看存储帐户列表。
2. 通过以下信息创建一个存储帐户。
 - 名称： **sp28532**[自定义名称]
 - 位置： 选择离你所在位置最接近的区域
 - 定价层： **L 本地冗余**
3. 通过启动板上图标访问存储帐户。
4. 记录存储帐户的主连接字符串和主访问密钥。

► 任务 4：切换到 Azure 管理门户

1. 登录 Azure 管理门户网站 <https://manage.windowsazure.com>。

► 任务 5：创建 Service Bus 名称空间

1. 查看 Service Bus 列表。
2. 通过以下信息创建一个 **Service Bus** 命名空间：
 - 命名空间名称: **sp28532[自定义名称]**
 - 区域: 选择离你所在位置最接近的区域
3. 记录 Service Bus 命名空间的 **SAS** 连接字符串的值。

结果：完成此练习后，您将创建好将用于已部署 Web 应用程序的服务。

练习 2：管理云 Web 应用程序的配置设置

场景

在此练习中，您将：

- 更新云服务配置设置。
- 更新 Web.config 发布转换文件。

此练习的主要任务如下：

1. 修改云配置设置
2. 修改管理项目的 Web.config 转换文件
3. 修改辅助角色项目的 app.config 文件
4. 修改 Web 角色项目的 Web.config 转换文件

► 任务 1：修改云配置设置

1. 从下列路径中打开 **Contoso.Events** 解决方案：
 - 文件位置: **Allfiles (F):\Mod13\Labfiles\Starter\Contoso.Events**
2. 打开 **Contoso.Events.Worker** 角色的属性页面。
3. 转到属性页面的设置区域。
4. 找到名称为 **Microsoft.ServiceBus.ConnectionString** 的 Service Bus 连接字符串设置并且根据下列值来更新：
 - 类型: 字符串
 - 值: 【先前任务中记下的 SAS 连接字符串】
5. 对于 **Cloud** 服务配置，定位到 **Microsoft.WindowsAzure.Storage.ConnectionString** 设置，然后通过之前任务中记下的存储帐户名称和主密钥来设置连接字符串。
6. 查看 **Local** 服务配置设置，并确保 **Microsoft.WindowsAzure.Storage.ConnectionString** 的值设置为 **UseDevelopmentStorage=true**。
7. 打开 **Contoso.Events.Web** 角色的属性页面。
8. 转到属性页面的设置区域。

9. 对于 **Cloud** 服务配置，定位到 **Microsoft.WindowsAzure.Storage.ConnectionString** 设置，然后通过之前任务中记下的存储帐户名称和主密钥来设置连接字符串。
10. 查看 Local 服务配置设置，并确保 **Microsoft.WindowsAzure.Storage.ConnectionString** 的值设置为 **UseDevelopmentStorage=true**。
 - ▶ 任务 2：修改管理项目的 **Web.config** 转换文件
 1. 在 **Contoso.Events.Web** 项目中打开 **Web.Debug.config** 文件。
 2. 将 appSetting 元素中的 **Microsoft.WindowsAzure.Storage.ConnectionString** 值更新为从存储帐户中记下的主连接字符串。
 3. 将 appSetting 元素中的 **Microsoft.ServiceBus.ConnectionString** 值更新为从 **Service Bus** 命名空间中记下的 **SAS** 连接字符串。
 4. 将元素 **EventsContextConnectionString** 的 connectionString 更新为从 **SQL** 数据库实例中记下的 **ADO.NET** 连接字符串。
 - ▶ 任务 3：修改辅助角色项目的 **app.config** 文件
 1. 在 **Contoso.Events.Worker** 项目中打开 **app.config** 文件。
 2. 将 **EventsContextConnectionString** 的 connectionString 更新为从 **SQL** 数据库实例中记下的 **ADO.NET** 连接字符串。
 - ▶ 任务 4：修改 Web 角色项目的 **Web.config** 转换文件
 1. 在 **Contoso.Events.Web** 项目中打开 **Web.Release.config** 文件。
 2. 将 **EventsContextConnectionString** 的 connectionString 更新为从 **SQL** 数据库实例中记下的 **ADO.NET** 连接字符串。

结果：完成此练习后，您将更改了 Web 应用程序，使其在部署到云后，使用不同的设置和连接字符串。

练习 3：将 Web 应用程序部署到 Azure

场景

在此练习中，您将：

- 将云服务项目部署到 Azure。
- 将 ASP.NET Web 应用程序项目部署到网站服务。

此练习的主要任务如下：

1. 将云服务项目部署到云服务生产环境
2. 将 ASP.NET Web 应用程序项目部署到网站服务实例
3. 验证应用程序成功发布

▶ 任务 1：将云服务项目部署到云服务生产环境

1. 开始 **Contoso.Events.Cloud** 项目的发布 **Windows Azure** 应用程序的向导。
2. 使用以下详细信息创建一个新的云服务实例：
 - 名称： **cs28532[自定义名称]**
 - 地区或地缘组：选择最接近你所在位置的区域
3. 使用以下值发布云服务：

MCT USE ONLY. STUDENT USE PROHIBITED

- 环境: **Production**
- 版本配置: **Release**
- 服务配置: **Cloud**

► **任务 2: 将 ASP.NET Web 应用程序项目部署到网站服务实例**

1. 使用服务器资源管理器, 使用以下信息创建一个新的网站:
 - 站点名称: **ws28532[自定义名称]**
 - 区域: **选择最接近您所在位置的区域**
 - 数据库服务器: **无数据库**
2. 开始 **Contoso.Events.Management** 项目的发布网站向导。
3. 使用以下名称将 Web 应用程序发布到新创建的网站:
 - 名称: **ws28532[自定义名称]**

► **任务 3: 验证应用程序成功发布**

1. 确认所有事件都在 Administration Web 应用程序中列出。
2. 关闭 Internet Explorer。

结果: 完成此练习后, 您将把项目部署到网站服务和云服务。

练习 4: 监视 Azure 中的 Web 应用程序

场景

在此练习中, 您将:

- 查看网站服务的跟踪输出。

此练习的主要任务如下:

1. 查看“管理”应用程序的流日志

► **任务 1: 查看“管理”应用程序的流日志**

1. 在服务器资源管理器中, 查看具有以下名称的新建网站实例的设置:
 - 网站名称: **ws28532[自定义名称]**
2. 在网站设置页面, 将应用程序日志记录(文件系统)设置更改为**详细**, 然后保存。
3. 查看网站的流式日志。
4. 在 Internet Explorer 中打开网站, 然后验证在访问页面时, 会显示相应的日志信息。
5. 关闭 Internet Explorer 和 Contoso.Events – Visual Studio 应用程序。

结果: 完成此练习后, 您将查看了网站服务的跟踪日志实时流。

实验 B：使用中国版 Windows Azure 部署 Events Web Application 到云环境

场景

现在你已经完成了 Contoso Event Web application 主要的开发工作，是该产品展示的时候了。作为 IT 管理过程中的一部分，你需要把应用部署到云的步骤都记录到文档中，要达到这个目标，你需要把管理员私有的 web 应用和公用的 Events web 应用部署到你的云帐户中。

目标

在你完成这个实验后，你将能：

- 更改 Web.config 转换文件。
- 部署一个云服务到 Azure 上。
- 部署一个 ASP.NET Web 应用工程到 Azure 网站。
- 查看 Azure 网站的流日志。

预计时间：60 分钟

在开始这个实验之前，必须完成第 2 章的**实验 B**。在此章实验中，您将使用自己的计算机完成实验。此外，还必须完成以下步骤：

1. 在计算机上，单击**开始**，键入**远程**，然后单击**远程桌面连接**。
2. 在远程桌面连接窗口中，在**计算机**框中选定以下格式的虚拟机名称。

vm28532[自定义名称].chinacloudapp.cn: [虚拟机 RDP 端口]

 **注释：**虚拟机的名称和端口可能会被保存在计算机下拉列表中。如果这样，就无需手动键入这些信息了。如果不确定虚拟机的 RDP 端口，还可以使用 Azure 门户网站来查找虚拟机的端点。名称为 **Remote Desktop** 的端点是正确的端口 RDP。此端口是随机的，这样可以保护您的虚拟机免受未经授权的访问。

3. 在远程桌面连接窗口中，单击**连接**，等待 RDP 客户端访问虚拟机。
4. 如果有必要，使用以下用户名密码登录：
 - 用户名：**Student**
 - 密码：**AzurePa\$\$w0rd**
5. 验证您拿到的是中国版 Windows Azure 帐户。
6. 验证使用您的中国版 Windows Azure 帐户可以登录到世纪互联运营的 Windows Azure 管理门户。您将在本课程的所有实验 B 中使用您的中国版 Windows Azure 帐户。

练习 1：创建部署 web 应用所需的 Azure 服务

场景

在这次练习中，你将要：

- 创建一个 SQL 数据库实例。
- 常见一个 Service Bus 实例。
- 创建一个存储帐户。

拿到 Azure 相应服务的连接字符串。

此练习的主要任务如下：

1. 登录到 Azure 管理门户
2. 创建一个 SQL 数据库实例
3. 创建一个存储实例
4. 创建一个 Service Bus 命名空间

► **任务 1：登录到 Azure 管理门户**

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 进入 <https://manage.windowsazure.cn>。
3. 输入中国版 Windows Azure 帐户的邮箱地址和密码。
4. 单击 **Sign in** 登录。

► **任务 2：创建一个 SQL 数据库实例**

1. 在界面左侧导航窗格中，单击 **SQL 数据库**。
2. 在界面左下角，单击**新建**。
3. 在新建向导中依次单击**数据服务**, **SQL 数据库**, **自定义创建**。
4. 在**新建 SQL 数据库**页面中，执行以下步骤创建 SQL 数据库实例：
 - a. 在名称框中输入 **dp28532[自定义名称]**。
 - b. 在服务层中选择 **Basic**。
 - c. 单击**服务器**。
 - d. 在**服务器**列表中，选择**新建 SQL 数据库服务器**。
 - e. 单击**下一步**。
 - f. 在**创建服务器**页面中，在**登录名**框中输入 **testuser**。
 - g. 在**登录密码**框中输入 **TestPa\$\$w0rd**。
 - h. 在**确认密码**框中输入 **TestPa\$\$w0rd**。
 - i. 单击**区域**。
 - j. 在**区域**列表中选择离你最近的**区域**。
 - k. 单击**完成**。



注释：在创建数据库实例完成以后，可以到管理门户左侧的所有项目或者 **SQL 数据库**查看刚创建的数据库。

5. 在所有项目中单击刚创建的数据库。
6. 在 **dp28532[自定义名称]**窗格中单击**仪表板**。
7. 在**仪表板**窗格中单击**显示连接字符串**。
8. 在**连接字符串**窗格中找到 **ADO.Net**。
9. 复制连接字符串。
10. 回到 windows 开始屏幕，输入 **Notepad**。

11. 单击 **Notepad** 磁贴。
12. 把连接字符串粘贴到记事本中。
13. 在记事本中，把字符串{此处为您的密码}替换为 **TestPa\$\$w0rd**。
14. 切回到浏览器的 **SQL 数据库-Microsoft Azure** 窗口。
15. 关闭**连接字符串**窗格。

► **任务 3：创建一个存储实例**

1. 在左侧导航窗格中单击**存储**。
2. 在存储页面查看显示的**存储**实例。
3. 在界面左下角单击**新建**。
4. 在**新建向导**中，依次单击**数据服务**, **存储**, **快速创建**。
5. 在**存储帐户信息**页面中，执行以下步骤：
 - a. 在**URL** 中填写 **sp28532[自定义名称]**。
 - b. 单击**位置/地缘组**，选择离你最近的位置。
 - c. 单击**地域冗余**，选择**本地冗余**。
 - d. 单击**创建存储帐户**。

 **注释：**当你**存储帐户**创建完成后，可以在左侧导航区**所有项目**或者**存储**中找到刚创建的**存储帐户**。

6. 单击左侧导航窗格中**存储**，定位到刚创建的**存储帐户** **sp28532[自定义名称]**，单击 **sp28532[自定义名称]**。
7. 在 **sp28532[自定义名称]**窗格底部，单击**管理访问密钥**。
8. 在**管理访问密钥**窗格，复制**主访问密钥**的值。
9. 将**主访问密钥**粘贴到记事本中。

 **注释：**存储帐户对应的连接字符串为：

```
BlobEndpoint=https://sp28532[自定义名  
称].blob.core.chinacloudapi.cn;/QueueEndpoint=https://sp28532[自定义名  
称].queue.core.chinacloudapi.cn;/TableEndpoint=https://sp28532[自定义名  
称].table.core.chinacloudapi.cn;/AccountName={sp28532[自定义名  
称]};AccountKey={存储帐户名对应的密钥} 将其粘贴到记事本中，接下来的实验中  
会使用到。
```

10. 切换到**存储-Microsoft Azure** 窗口。

11. 关闭**管理访问密钥**页面。

► **任务 4：创建一个 Service Bus 命名空间**

1. 在界面左侧导航窗格中，单击**Service Bus**。
2. 在**Service Bus** 页面中，查看命名空间。
3. 在界面下方单击**创建**。
4. 在**创建命名空间**页面中，执行以下步骤：
 - a. 在**命名空间名称**中，输入 **sp28532[自定义名称]**。

- b. 在**区域**列表中选择离你最近的区域。
- c. 单击**确定**按钮。



注释: 当命名空间创建完毕后，会显示在 Service Bus 命名空间列表中。

5. 单击刚创建的命名空间 **sp28532[自定义名称]**。
6. 在页面的下方单击**连接信息**。
7. 定位到**SAS** 区域，记录 **RootManageSharedAccessKey** 连接字符串的值。
8. 将连接字符串值复制到记事本中。
9. 切换到浏览器 **Service Bus - Microsoft Azure** 窗口。
10. 关闭**访问连接信息**窗格。

结果: 完成这个练习，你将创建你部署 web 应用所需要的的服务。

练习 2：管理云上 Web 应用的配置设置

场景

在这个练习中，你将要：

- 更新云服务中的配置设置。
- 更新 Web.config 发布转换文件。

此练习的主要任务如下：

1. 更改云配置设置
2. 更改管理项目的 Web.config 转换文件
3. 更改辅助角色项目的 app.config 文件
4. 更新 web 角色项目的 web.config 转换文件

► 任务 1：更改云配置设置



注释: 如果你之前没有保存过你的帐户凭证，在这次实验过程中你可能会被告知你需要登录。具体步骤请参考第 2 章 -> 实验 B -> 练习 4 -> 任务 1，任务 2。

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏中，单击 **File Explorer**。
3. 在 This PC 窗口，浏览到 **Allfiles (F):\Mod13\Labfiles\Starter\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格，展开 **Contoso.Events.Cloud** 项目，展开**角色**文件夹，然后双击 **Contoso.Events.Worker** 这个角色。
5. 在屏幕左侧，单击**配置**标签。
6. 在页面的顶部，确认服务配置设置为**所有配置**。
7. 定位到名称为 **Microsoft.ServiceBus.ConnectionString** 的设置。

MCT USE ONLY STUDENT USE PROHIBITED

8. 确认类型设置为**字符串**。
9. 在值框中，输入**SAS 连接字符串**。

 **注释：**SAS 连接字符串在之前的任务中已粘贴到记事本中。格式如：Endpoint=sb://{命名空间名}.servicebus.chinacloudapi.cn;/SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=.....

10. 定位到页面顶部的**服务配置**。
11. 选择**Cloud**。
12. 定位到名称为**Microsoft.WindowsAzure.Storage.ConnectionString** 的设置。
13. 单击**值**框，然后单击右边的 (...) 省略号。
14. 在弹出的**创建存储连接字符串**对话框，执行以下步骤：
 - a. 单击**您的订阅**。
 - b. 在**帐户名称**列表中，选择上一个练习任务中创建的存储帐户 **sp28532[自定义名称]**。
 - c. **连接使用默认连接：使用 HTTPS (推荐)**。
 - d. 预览连接字符串。
 - e. 单击**确定**。
15. 定位到页面顶部的**服务配置**。
16. 选择**Local**。
17. 确认 **Microsoft.WindowsAzure.Storage.ConnectionString** 的值为
useDevelopmentStorage=true。
18. 在解决方案资源管理器窗格，展开**Contoso.Events.Cloud** 项目，展开**角色**文件夹，然后双击**Contoso.Events.Web** 这个角色。
19. 在屏幕左侧，单击**配置**标签，然后定位到页面顶部的**服务配置**。
20. 选择**Cloud**。
21. 定位到名称为**Microsoft.WindowsAzure.Storage.ConnectionString** 的设置。
22. 单击**值**框，然后单击右侧的 (...) 省略号。
23. 在弹出的**创建存储连接字符串**对话框，执行以下步骤：
 - a. 单击**您的订阅**。
 - b. 在**帐户名称**列表中，选择上一个练习任务中创建的存储帐户 **sp28532[自定义名称]**。

 **注释：**可以看到**预览连接字符串**，格式为：BlobEndpoint=https://sp28532[自定义名称].blob.core.chinacloudapi.cn;/QueueEndpoint=https://sp28532[自定义名称].queue.core.chinacloudapi.cn;/TableEndpoint=https://sp28532[自定义名称].table.core.chinacloudapi.cn;/AccountName={sp28532[自定义名称]};AccountKey={存储帐户的主访问密钥}

 - c. **连接使用默认连接：使用 HTTPS (推荐)**。
 - d. 单击**确定**。

24. 定位到页面顶部的**服务配置**。

25. 选择 **Local**。
26. 确认 **Microsoft.WindowsAzure.Storage.ConnectionString** 的值设置为 **useDevelopmentStorage=true**。
27. 单击端点，确认当前的公用端口，从浏览器访问云服务的时候最好带上端口号。

 **注释：**因为设置的公用端口可能不是浏览器默认的 80 端口。如 <http://contosoevents.chinacloudapp.cn:85>

► **任务 2：更改管理项目的 Web.config 转换文件**

1. 在解决方案资源管理器窗格中，展开 **Administration** 文件夹，展开 **Contoso.Events.Management** 项目，展开 **Web.config** 文件，然后双击文件 **Web.Debug.config**。
2. 定位到 **appSettings XML** 元素。
3. 定位到 key 的值为 **Microsoft.WindowsAzure.Storage.ConnectionString** 的 add 子元素。
4. 将元素 **value** 值设置成存储帐户对应的连接字符串。
5. 定位到 **appSettings XML** 元素。
6. 定位到 key 的值为 **Microsoft.ServiceBus.ConnectionString** 的 add 子元素。
7. 将元素 **value** 设置成之前记录的 Service Bus 命名空间的 **SAS 连接字符串**。
8. 定位到 **connectionStrings XML** 元素。
9. 定位到 name 为 **EventsContextConnectionString** 的 add 子元素。
10. 将元素 **connectionString** 更新为从 SQL 数据库实例中记录的 **ADO.NET 连接字符串**。

► **任务 3：更改辅助角色项目的 app.config 文件**

1. 在解决方案资源管理器窗格中，展开 **Roles** 文件夹，展开 **Contoso.Events.Worker** 项目，然后双击 **app.config** 文件。
2. 定位到 **connectionStrings XML** 元素。
3. 定位到 name 为 **EventsContextConnectionString** 的 add 子元素。
4. 将元素 **connectionString** 更新为从 SQL 数据库实例中记录的 **ADO.NET 连接字符串**。

► **任务 4：更新 web 角色项目的 web.config 转换文件**

1. 在解决方案资源管理器窗格，展开 **Roles** 文件夹，展开 **Contoso.Events.Web** 项目，展开 **web.config** 文件，然后双击文件 **Web.Release.config**。
2. 定位到 **connectionStrings XML** 元素。
3. 定位到 name 为 **EventsContextConnectionString** 的 add 子元素。
4. 将元素 **connectionString** 更新为从 SQL 数据库实例中记录的 **ADO.NET 连接字符串**。

结果：完成这个练习后，你的 web 应用将被设置成：即使应用发布到云之后，他们依然可以更改这个应用的配置和链接字符串信息。

练习 3：部署 Web 应用到云环境

场景

在这个练习中，你将要：

- 部署一个云服务到云环境。
- 部署一个 ASP.NET Web 应用到 Azure 的网站。

此练习的主要任务如下：

- 部署云服务项目到 Azure 的产品环境
- 部署一个 ASP.NET Web 应用项目到云环境的网站
- 确认网站已经成功发布

► **任务 1：部署云服务项目到 Azure 的产品环境**

- 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Cloud** 项目，然后单击**发布**。
- 在**登录**向导中，在**选择订阅**列表中选择您的订阅。
- 单击**下一步**，进入**设置**向导页面。
- 定位到**云服务**列表，然后选择**新建**。

 **注释：**如果在你的订阅中没有一个云服务，不需要选择新建，它将自动弹出**创建云服务**对话框。你可以继续第五步。

- 在**创建云服务**对话框，执行以下步骤：
 - 在名称框中，输入 **cs28532[自定义名称]**。
 - 在**地区或地缘组**列表中，选择离你最近的区域。
 - 单击**创建**。
- 在**环境**列表中选择 **Production**。
- 在**版本配置**列表中选择 **Release**。
- 在**服务配置**列表中选择 **Cloud**。
- 单击**发布**。

► **任务 2：部署一个 ASP.NET Web 应用项目到云环境的网站**

- 在视图菜单，单击**服务器资源管理器**。
- 定位到 Azure 节点，单击左边箭头。
- 右键单击 Azure 节点下的网站，单击**创建新站点**。
- 在站点名称框中，输入网站名称 **ws28532[自定义名称]**。
- 在**区域**列表中，选择离你最近的区域。
- 在**数据库服务器**列表中，确认选中**无数据库**。如果出现感叹号，且没有值可选，刷新一下**网站节点**。
- 单击**创建**。
- 在解决方案资源管理器窗格中，展开**Administration** 文件夹，右键单击**Contoso.Events.Management** 项目，然后单击**发布**。
- 在**配置文件**中，选择 **Microsoft Azure** 网站为发布目标。
- 在**选择现有网站**对话框中，执行以下步骤：
 - 在**现有网站**列表中，选择 **ws28532[自定义名称]**。
 - 单击**确定**。

- c. 在弹出的是否保存提示框中，单击 **Yes**。
11. 在**连接**页面中，单击**下一步**。
12. 在**设置**页面中，将配置更改为 **Debug – Any CPU**。
13. 单击**发布**。
14. 发布完成之后，确认网站成功发布。

► **任务 3：确认网站已经成功发布**

1. 在 Internet Explorer 中，浏览 **Contoso Events Administration** 主页，然后单击 **Go to Events List**。
2. 确认所有的事件的 Registrants 值都为 0。
3. 关闭浏览器。

结果：完成这次练习，你将学会部署项目到 Azure 的网站和云服务。

练习 4：在 Azure 上监测 Web 应用

场景

这次练习，你将要：

- 查看网站的流式日志。

此练习的主要任务如下：

1. 查看网站的流式日志

► **任务 1：查看网站的流式日志**

1. 在视图菜单，单击**服务器资源管理器**。
2. 定位到 **Azure** 节点，单击左边箭头。
3. 展开 **Azure** 节点下的**网站**。
4. 右键单击前缀为 **ws28532** 的网站，然后单击**查看设置**。
5. 定位到**应用程序日志记录（文件系统）**。
6. 将值由**关闭**改为**详细**。
7. 单击**保存**。
8. 右键单击前缀为 **ws28532** 的网站，然后单击**查看流式日志**。
9. 右键单击前缀为 **ws28532** 的网站，然后单击**在浏览器中查看**。
10. 在 Internet Explorer 中，在 **Contoso Events Administration** 页面中，单击 **Go to Events List**。
11. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
12. 确认控制台窗格输出两条这样的记录 **Information Viewed the Index page** 和 **Information Viewd the Events page**。
13. 右键单击前缀为 **ws28532** 的网站，然后单击**停止查看日志**。
14. 关闭 **Internet Explorer** 应用程序。
15. 关闭 **Microsoft Visual Studio** 应用程序。

MCT USE ONLY. STUDENT USE PROHIBITED

结果: 完成这次练习, 你将会查看一个网站的流式日志。

MCT USE ONLY. STUDENT USE PROHIBITED

章节复习和作业

本章中，讨论了 Web 应用程序的不同部署策略，并根据可组合性、幂等性和可跟踪性这三个标准作了比较。尤其是，详细探讨了云服务包和 Web Deploy 包。最后，讨论了云服务和网站服务的监视和调试技术。

最佳做法

Web Deploy 是在 IIS 上来回部署包的非常灵活的现代方法。即使是对于企业内部应用程序，Web Deploy 也是部署 Web 应用程序和配置 IIS 的最佳选择之一。

复习问题

问题：您想将 Web 应用程序部署到云中。那么对于开发环境中的应用程序，以及生产环境中的 Web 应用程序，哪些设置应该不相同？试举几例。

课程评估

您对本课程的评价将帮助 Microsoft 了解您的学习质量。

请与您的培训提供商联系，索取课程评估表。

Microsoft 将替您对本次调查的答复保密，并将使用您的反馈来改进未来的学习体验。您开诚布公的反馈将是我们宝贵的资产，我们对此深表感谢。

• 您对本课程的评价将帮助 Microsoft 了解您的学习质量。

• 请与您的培训提供商联系，索取课程评估表。

• Microsoft 将替您对本次调查的答复保密，并将使用您的反馈来改进未来的学习体验。您开诚布公的反馈将是我们宝贵的资产，我们对此深表感谢。

第 1 章：Microsoft Azure 平台概述

实验 A：探索 Azure 预览门户

练习 1：登录到预览门户

► 任务 1：登录到预览门户

1. 在开始屏幕，单击 **Internet Explorer** 磁贴。
2. 访问 <https://portal.azure.com>。
3. 输入 Microsoft 帐户的 email 地址。
4. 单击 **Continue**。
5. 输入 Microsoft 帐户的密码。
6. 单击 **Sign In**。
7. 在显示的对话框中，单击 **Get Started**。
8. 单击右上角的帐户名。
9. 单击 **Settings**。
10. 单击 **LANGUAGE**，选择中文（简体），单击 **Select**。
11. 关闭门户设置边栏选项卡。

结果：完成本练习后，您将直接登录预览门户，而无需先经历管理门户。

练习 2：自定义预览门户

► 任务 1：自定义启动板

1. 仪表板完全加载后，右键单击**启动板**（主屏幕），然后单击**自定义**。
2. 单击**服务运行状况**磁贴，然后单击灰色省略号“...”按钮。
3. 选择**普通**选项。
4. 右键单击**启动板**（主屏幕），然后单击**自定义完成**。

► 任务 2：查看边栏选项卡

1. 在屏幕的左侧，单击**浏览**。
2. 在显示的**浏览**边栏选项卡，单击**门户设置**。
3. 查看显示的**门户设置**边栏选项卡。

► 任务 3：开始导览

1. 在屏幕的左下角，单击**新建**。
2. 在显示的**创建**边栏选项卡，单击**Web+移动**，然后单击**Azure Marketplace**。
3. 在显示的**Marketplace**边栏选项卡中，单击**Web**。
4. 在显示的**Web**边栏选项卡中，定位**起动器站点组**，然后单击**Node JS Starter Site** 选项。
5. 在显示的**Node JS Starter Site**边栏选项卡中，单击**创建**。
6. 在**网站**边栏选项卡，在**新建**框中，输入任何值。
7. 在**网站**边栏选项卡上，单击关闭按钮(**X**)关闭边栏选项卡。
8. 在 Internet Explorer 的**Message from webpage**对话框中，单击**OK**。

结果：完成此练习后，您将完成查看边栏选项卡、导览和导览部分的任务。

实验 B：认识中国版 Windows Azure 管理门户

练习 1：登录并了解 Azure 管理门户

► 任务 1：登录到 Azure 管理门户

1. 登录到 Azure 管理门户。
2. 在开始屏幕上，单击 **Internet Explorer** 磁贴。
3. 在地址栏中，输入 <https://manage.windowsazure.cn>。
4. 点击回车键。
5. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
6. 如果您的界面语言不是中文，可以点击页面右上角的**地球**图标，并在菜单中选择**中文(简体)**语言。

 **注释：**切换界面语言的另一个方式是，在页面右上角，点击帐户链接，在弹出的菜单中，点击**View my bill**，滚动屏幕，在页面的左下角点击**English**，在弹出的菜单中，点击**中文(简体)**。在页面的右上角，点击**门户**按钮。

► 任务 2：认识管理门户的布局

1. 在管理门户页面，页面左侧为导航栏。
2. 在导航栏中的第一项为**所有项目**，这里列出来了所有的您已创建的服务的名称。
3. 在导航栏中的其它项为每一个可用服务的名称。
4. 在管理门户页面，页面下方为工具栏。
5. 在工具栏中的左侧为**新建**按钮，通过这个按钮可以创建 Azure 服务。
6. 在管理门户的右上角为**地球**图标和用户帐户链接。

结果：完成本练习后，您将使用您的 Azure 帐户登录到中国版 Windows Azure 管理门户。

第 2 章：使用 Azure 虚拟机建立开发环境

实验 A：创建用于开发和测试的 Azure 虚拟机

练习 1：创建网络和资源容器

► 任务 1：登录到预览门户

1. 在开始屏幕上，单击 **Internet Explorer** 磁贴。
2. 访问 <https://portal.azure.com>。
3. 输入微软帐户的 email 地址。单击 **Continue**。
4. 输入微软帐户的密码。单击 **Sign In**。
5. 单击右上角的帐户名。
6. 单击 **Settings**。
7. 单击 **LANGUAGE**，选择中文（简体），然后单击 **Select**。
8. 关闭门户设置边栏选项卡。

► 任务 2：创建虚拟网络和资源组

1. 在预览门户左侧的导航窗格中，向下滚动，然后单击 **浏览**。
2. 在显示的 **浏览** 边栏选项卡中，单击 **虚拟网络**。
3. 在显示的 **Virtual networks** 边栏选项卡中，查看虚拟网络实例的列表。
4. 在屏幕的左下角，单击 **新建**。
5. 在显示的 **创建** 边栏选项卡中，单击 **网络**，然后单击 **Virtual Network**。
6. 在 **虚拟网络** 边栏选项卡中，执行以下步骤：
 - a. 在名称对话框，输入 **Dev28532**。
 - b. 在位置列表，选择最接近当前位置的地区。
7. 在 **虚拟网络** 边栏选项卡中，单击 **地址空间**。
8. 在 **地址空间** 边栏选项卡中，执行以下步骤：
 - a. 确保空间地址 **CIDR** 块框的值为 **10.0.0.0/16**。
 - b. 确保子网 **CIDR** 块框的值为 **10.0.0.0/24**。
 - c. 在子网名称框中，输入 **Apps**。
 - d. 单击确认。
9. 在 **虚拟网络** 边栏选项卡中，单击 **资源组**。

 **注释：**如果**创建资源组**边栏选项卡弹出错误页面，可以先创建一任意**虚拟网络**，再重新执行步骤 4 到步骤 12 创建虚拟网络。

10. 在 **资源组** 边栏选项卡中，单击 **创建新资源组**。
11. 在 **创建资源组** 边栏选项卡中，执行以下步骤：

- a. 在**名称**对话框，输入**Dev28532**。
 - b. 单击**确认**。
12. 在**虚拟网络**边栏选项卡中，单击**创建**。

结果：完成此练习后，您将在 Azure 中获得新的虚拟网络和资源组。

练习 2：创建开发虚拟机

► 任务 1：创建存储帐户

1. 在预览门户左侧的导航窗格中，向下滚动，然后单击**浏览**。
2. 在显示的**浏览**边栏选项卡中，单击**存储帐户**。
3. 在显示的**Storage Accounts** 边栏选项卡中，查看存储实例的列表。
4. 在屏幕的左下角，单击**新建**。
5. 在显示的**创建**边栏选项卡中，单击**管理**，然后单击**Azure Marketplace**。
6. 在显示的**Marketplace** 边栏选项卡中，单击**Storage, cache, + backup**。
7. 在显示的**Storage, cache, + backup** 边栏选项卡中，定位**Storage and Cache** 部分和单击**Storage**。
8. 在显示的**Storage** 边栏选项卡中，单击**创建**。
9. 在显示的**存储帐户**边栏选项卡中，执行以下步骤：
 - a. 在**存储空间**对话框，输入**stor28532**[自定义名称]。
 - b. 单击**位置**。
 - c. 在**位置**边栏选项卡中，选择靠近当前位置的地区。
 - d. 单击**定价层**。
 - e. 在**选择你的定价层**边栏选项卡，选择**L 本地冗余**。
 - f. 单击**选择**。
 - g. 单击**创建**。

► 任务 2：创建虚拟机

1. 在预览门户左侧的导航窗格中，向下滚动，然后单击**浏览**。
2. 在显示的**浏览**边栏选项卡中，单击**存储帐户**。
3. 在显示的**Virtual machines** 边栏选项卡中，查看虚拟机实例的列表。
4. 在屏幕的左下角，单击**新建**。
5. 在显示的**创建**边栏选项卡中，单击**计算**，然后单击**Windows Server 2012 R2 Datacenter**。
6. 在显示的**创建虚拟机**边栏选项卡中，执行以下步骤：
 - a. 在**主机名**对话框，输入**vm28532**[自定义名称]。
 - b. 在**用户名**对话框，输入**Student**。
 - c. 在**密码**对话框，输入**AzurePa\$\$wOrd**。
 - d. 单击**定价层**。
 - e. 在**选择你的价格层**边栏选项卡中，单击**查看所有**。
 - f. 选择**A3 标准**选项。
 - g. 在**选择你的定价层**边栏选项卡，单击**选择**。
 - h. 单击**可选配置**。
 - i. 在**可选配置**边栏选项卡中，单击**网络**。
 - j. 在**网络**边栏选项卡中，单击**虚拟网络**。
 - k. 在**虚拟网络**边栏选项卡中，选择之前创建的网络，**Dev28532**。

- i. 在**网络**边栏选项卡中，单击**确认**。
- m. 在**可选配置**边栏选项卡中，单击**存储帐户**。
- n. 在**存储帐户**边栏选项卡中，选择之前创建的存储帐户，**stor28532[自定义名称]**。
- o. 在**可选配置**边栏选项卡中，单击**确认**。
- p. 在**创建虚拟机**边栏选项卡中，选择**资源组**的选项。
- q. 在显示的**资源组**边栏选项卡中，定位**使用现有的资源组**列表，然后选择**Dev28532** 资源组。
- r. 在**创建虚拟机**边栏选项卡中，单击**创建**，使用指定的配置创建虚拟机。

 **注释：** 创建一个新的虚拟机可能需要 10 到 15 分钟。当你的虚拟机创建并运行时，您将在启动板（主屏幕）看到一个通知。

7. 从启动板选择新创建的虚拟机。
8. 在**vm28532[自定义名称]**边栏选项卡中，单击**All settings**。
9. 单击**磁盘**。
10. 在**磁盘**边栏选项卡中，单击**附加新项**。
11. 在**附加新磁盘**边栏选项卡中，执行以下步骤：
 - a. 单击**存储容器**。
 - b. 在**选择容器**边栏选项卡中，单击**选择存储帐户**。
 - c. 在**存储帐户**边栏选项卡中，选择之前创建的存储帐户，**stor28532[自定义名称]**。
 - d. 在**选择容器**边栏选项卡中，单击**选择容器**。
 - e. 在**存储容器**边栏选项卡中，选择**vhds** 容器。
 - f. 在**选择容器**边栏选项卡中，单击**确认**。
 - g. 在**磁盘文件名**对话框中，输入**vm28532-AllFiles.vhd**。
 - h. 在**大小**对话框中，输入**6**。
 - i. 单击**确认**创建第二个磁盘。
12. 返回到**vm28532[自定义名称]**边栏选项卡。
13. 在屏幕顶部单击**连接**。
14. 在**Internet Explorer download** 对话框，单击**Open**。
15. 在**Remote Desktop Connection** 对话框，执行以下步骤：
 - a. 选择**Don't ask me again for connections to this computer**，以阻止这个对话框再次显示。
 - b. 单击**Connect**。
16. 在**Windows Security** 对话框中，执行以下步骤：
 - a. 在**User name** 对话框，输入**Student**。
 - b. 在**Password** 对话框，输入**AzurePa\$\$w0rd**。

- c. 单击 **OK**。
17. 在 **Remote Desktop Connection** 对话框中，执行以下步骤：
- a. 确认远程证书的名称匹配虚拟机的名称。
 - b. 选择 **Don't ask me again for connections to this computer**，以阻止这个对话框再次显示。
 - c. 单击 **Yes**。
18. 当被提示允许网络连接发现外部设备，单击 **No**。

结果：完成此练习后，您将获得一个存储在新的存储帐户下的新虚拟机。

练习 3：配置用于开发的虚拟机

► 任务 1：创建 AllFiles 驱动器

1. 在开始屏幕上，单击 **Server Manager** 磁贴。
2. 在左边的导航窗格中，单击 **Local Server**。
3. 在 **Properties** 框，单击 **IE Enhanced Security Configuration** 选项，目前设置为 **On**。
4. 在 **Internet Explorer Enhanced Security Configuration** 对话框中，执行以下步骤：
 - a. 在 **Administrators** 下，选择 **Off**。
 - b. 在 **Users** 下，选择 **Off**。
 - c. 单击 **OK**。
5. 按下 Windows 徽标键+W 来打开 **Universal Search-Settings**。
6. 在 **Search** 对话框中，输入 **disk**。
7. 单击 **Create and format hard disk partitions**。
8. 在 **Initialize Disk** 对话框中，执行以下步骤：
 - a. 验证初始化选择了 **Disk 2**。
 - b. 确认所选分区风格是 **MBR (Master Boot Record)**。
 - c. 单击 **OK**。
9. 在磁盘管理窗口的下半部分中，执行以下步骤：
 - a. 向下滚动并找到之前初始化的 **Disk 2**。
 - b. 右键单击 **unallocated** 的分区，然后单击 **New Simple Volume**。
10. 在 **New Simple Volume wizard** 中，执行以下步骤：
 - a. 单击 **Next**。
 - b. 确认 **Simple volume size in MB** 是 **6141**。
 - c. 单击 **Next**。
 - d. 在 **Assign the following drive letter** 列表，单击 **F**。
 - e. 单击 **Next**。
 - f. 确认 **File System** 设置成 **NTFS**。
 - g. 在 **Volume Label** 对话框，输入 **AllFiles**。
 - h. 单击 **Next**。
 - i. 单击 **Finish** 关闭对话框，然后创建分区。



注释：如果将显示一个对话框说你需要格式化磁盘驱动器 F：才能使用它。, 你可以安全地关闭它，因为你已经格式化磁盘。

11. 在开始屏幕，单击 **Internet Explorer** 磁贴。
12. 如果提示设置 Internet Explorer 11，请执行以下步骤：
 - a. 选择 **Use recommended security and compatibility settings**。
 - b. 单击 **OK**。

13. 访问 <http://www.microsoft.com/learning/companionmoc>。
 14. 向下滚动屏幕，直到找到 **28532B 开发 Microsoft Azure 解决方案** 课程。
 15. 单击 **28532B-ENU-AllFiles.exe** 来下载 AllFiles 可执行文件。
 16. 在 **Internet Explorer** 下载对话框，单击 **Run**。
AllFiles 可执行文件的下载通常需要大约 5 分钟。
 17. 在 **Official Microsoft Learning Product License Terms** 对话框，单击 **Accept**。
 18. 在 **WinRAR self-extracting archive** 对话框，执行以下步骤：
 - a. 在 **Destination folder** 对话框，输入 **F:**。
 - b. 单击 **Extract**。
 19. 等待提取过程完成。
- **任务 2：安装 Visual Studio 2013 Ultimate Update 4**
1. 在开始屏幕上，单击 **Internet Explorer**。
 2. 在地址栏，输入 <http://go.microsoft.com/fwlink/?LinkId=525334>。
 3. 按 Enter 键。
 4. 在 Select Language 框中，选择 Chinese(Simplified)。
 5. 单击 **下载**。
 6. 选择 **vs_ultimate.exe**。
 7. 单击 **Next**。

 **注释：**注意，Internet Explorer 阻止下载对话框。查看这个对话框，在 Internet Explorer 窗口底部的黄色警告消息中，单击 **Options for this Site**，然后单击 **Always Allow**。

8. 在 Internet Explorer 窗口底部的下载对话框中，单击 **Run**。
9. 在最初的 **Visual Studio** 对话框中，选择 **我同意许可条款和隐私策略**。
10. 单击 **下一步**。
11. 在要安装的可选功能列表，确保只选择以下选项：
 - a. Microsoft Office 开发人员工具
 - b. Microsoft SQL Server Data Tools
 - c. Microsoft Web 开发人员工具
12. 单击 **安装**。

 **注释：**所需资源的下载和安装完成通常需要大约 30 到 45 分钟的时间。

13. 等待 **Visual Studio 2013 Ultimate** 安装完成。
14. 在 **Visual Studio Ultimate 2013** 对话框中，单击 **启动**。
15. 在欢迎使用，请登录 **Visual Studio** 对话框中，单击 **以后再说**。
16. 在以熟悉的环境启动对话框中，执行以下步骤：

- a. 在**开发设置**列表中，单击**Visual C#**。
 - b. 单击启动**Visual Studio**。
17. 等待**Visual Studio** 配置完成。
18. 通过单击窗口的右上角关闭按钮(x)，关闭 Visual Studio 2013 Ultimate 窗口。
- **任务 3：安装 Azure SDK for .NET 2.5**
1. 在开始屏幕上，单击**Internet Explorer** 磁贴。
 2. 在地址栏，输入<http://go.microsoft.com/fwlink/?LinkId=525337>。
 3. 按 Enter 键。
 4. 在 Select Language 框中，选择 Chinese(Simplified)。
 5. 通过单击**详情**前的加号按钮展开**详情**部分。
 6. 向下滚动，然后单击**VS 2013**。
 7. 在 Internet Explorer 窗口底部中的下载对话框，单击**Run**。

 **注释：**Web Platform Installer 5.0 软件安装可能需要一或两分钟。这个软件将为.NET 2.5 检索 Azure SDK。

8. 确认这个包的名字为**Microsoft Azure SDK for .NET (VS 2013) – 2.5**。
9. 单击**Install**。
10. 单击**I Accept**。

下载并安装 SDK 大约需要 5 分钟。

11. 等待**Web Platform Installer** 程序完成。
12. 单击**Continue** 查看.NET Dev Center | Azure。
13. 关闭**Internet Explorer**。
14. 在 Web Platform Installer 5.0 窗口，单击**Finish**。
15. 单击**Exit**。
16. 在开始屏幕上，单击向下箭头查看所有应用程序，然后右键单击**Visual Studio 2013** 磁贴。
17. 单击屏幕底部的**Pin to Start**。
18. 单击**Visual Studio 2013** 磁贴。
19. 在视图菜单，单击**服务器资源管理器**。
20. 定位**Azure** 节点，然后单击节点左侧的箭头。
21. 右击 Azure 节点，然后单击**连接到 Microsoft Azure 订阅**。
22. 如果有必要，通过使用 Microsoft 帐户密码登录 Azure 订阅。

► **任务 4：安装 Azure PowerShell 模块**

1. 在开始屏幕上，单击**Internet Explorer** 磁贴。
2. 在地址栏，输入<http://go.microsoft.com/fwlink/p/?linkid=320376>，然后按回车。

 **注释:** 注意, Internet Explorer 阻止下载对话框。查看这个对话框, 在 Internet Explorer 窗口底部的黄色警告消息中, 单击 **Options for this Site**, 然后单击 **Always Allow**。

3. 在 **Internet Explorer** 窗口的底部下载对话框中, 单击 **Run**。
4. 确认这个包的名字为 **Microsoft Azure PowerShell with Microsoft Azure SDK**。
5. 单击 **Install**。
6. 单击 **I Accept**。

 **注释:** 这个模块的下载和安装大约需要 5 分钟。

7. 等待 **Web Platform Installer** 程序完成。
8. 在 Web Platform Installer 5.0 窗口中, 单击 **Finish**。
9. 单击 **Exit**。
10. 关闭 **Internet Explorer** 应用。

 **注释:** 现在连接到实验环境的虚拟机。

结果: 完成此练习后, 您的开发虚拟机上将安装 Visual Studio、Azure PowerShell 和 Azure SDK。

实验 B：在中国版 Windows Azure 中创建用于开发和测试的虚拟机

练习 1：创建网络

► 任务 1：登录到 Azure 管理门户

1. 在开始屏幕上，单击 **Internet Explorer** 磁贴。
2. 在地址栏中，输入 <https://manage.windowsazure.cn>。
3. 按 Enter 键。
4. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
5. 如果您的界面语言不是中文，可以单击页面右上角的**地球**图标，并在菜单中选择**中文(简体)**语言。

► 任务 2：创建虚拟网络

1. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击**网络**。
2. 在页面下方的工具栏左侧，单击**新建**按钮。
3. 在弹出的创建向导中，单击**虚拟网络**，单击**自定义创建**。
4. 在弹出的**创建虚拟网络**窗口中，执行以下步骤：
 - a. 在**名称**对话框中，输入 **Dev28532**。
 - b. 在**位置**下拉框中，选择一个**离你位置较近的区域**。
 - c. 在窗口右下角，单击**右箭头**按钮进入下一个页面。
5. 在**DNS 服务器和 VPN 连接**页面，单击**右箭头**进入**虚拟网络访问空间**页面。
6. 在**地址空间**行，确保起始 IP 是 **10.0.0.0**，确保 CIDR 框中值为/16(65536)。
7. 在**子网**行，确保起始 IP 是 **10.0.0.0**，确保 CIDR 框中值为/24(256)。
8. 修改子网名称为 **Apps**。
9. 单击**完成**按钮。

结果：完成此练习后，您将在 Azure 中获得新的虚拟网络。

练习 2：创建 Azure 虚拟机开发环境

► 任务 1：创建存储帐户

1. 在 Azure 管理门户页面左侧的导航栏中，向下滚动，单击**存储**。
2. 在页面下方的工具栏左侧，单击**新建**按钮。
3. 在弹出的新建菜单中，选择**存储**，选择**快速创建**。
4. 在右侧弹出的创建存储帐户栏中，执行如下步骤：
 - a. 在**URL** 文本框中输入**stor28532[自定义名称]**。
 - b. 在**位置/地缘组**下拉框中选择一个**距离你较近的区域**。
 - c. 在**复制**下拉框中选择**本地冗余**。
 - d. 然后单击页面右下方的**创建存储帐户**按钮。

► 任务 2：创建虚拟机

1. 在 Azure 管理门户页面左侧的导航栏中，向下滚动，单击**虚拟机**。
2. 在页面下方的工具栏左侧，单击**新建**按钮。
3. 在弹出的新建菜单中，选择**虚拟机**，选择**从库中**。
4. 在**选择映像**窗口中，在中间栏浏览，选择**Windows Server 2012 R2 Datacenter (EN-US)**。
5. 单击页面右下角的**下一步**按钮。
6. 在虚拟机配置窗口，执行如下步骤：
 - a. 在**虚拟机名称**文本框中，输入**vm28532[自定义名称]**。
 - b. 在**层**选项中，选择**标准**。
 - c. 在**大小**下拉框中，选择**A2 (双核, 3.5GB 内存)**。
 - d. 在**新用户名**文本框中，输入**Student**。
 - e. 在**新密码和确认**文本框中，输入**AzurePa\$\$wOrd**。
7. 然后单击页面右下角的**下一步**按钮，进入窗口 2 并执行如下步骤：
 - a. 在**区域/地缘组/虚拟网络**下拉框中，在**虚拟网络**下选择**dev28532**。
 - b. 在**存储帐户**下拉框中，选择**stor28532[自定义名称]**。
 - c. 然后单击页面右下角的**下一步**按钮。
8. 在窗口 3 中，单击页面右下角的**完成**按钮。

 **注释：** 创建新虚拟机大约花费 10 到 15 分钟的时间，当虚拟机创建完成，页面下方的通知栏中会显示已成功创建虚拟机信息，单击确定按钮。

9. 在**虚拟机**页面，选择新创建的虚拟机**vm28532[自定义名称]**。
10. 在页面下方的工具栏中，选择**附加**，单击**附加空磁盘**。
11. 在将空磁盘附加到虚拟机窗口中，执行如下步骤：
 - a. 在**文件名**文本框中，更改文件名为**vm28532-AllFiles**。
 - b. 在**大小(GB)**文本框中，输入数值**6**。
 - c. 然后单击页面右下角的**确定**按钮。



注释: 等待将空磁盘附加到虚拟机，这个过程大约花费 5 分钟。

12. 在**虚拟机**页面，选择新创建的虚拟机 vm28532[自定义名称]。
13. 在页面下方的工具栏中，单击**连接**。
14. 在 Internet Explorer 下载对话框中，单击**打开**按钮。
15. 在远程桌面连接窗口，执行如下步骤：
 - a. 如果弹出**无法识别此远程连接的发布者。是否仍要连接？**的对话框，选中**不再询问我是否连接到此计算机**。
 - b. 单击**连接**按钮。
 - c. 在 Windows 安全窗口中，执行如下步骤：
 - i. 在**用户名**文本框中，输入 Student。
 - ii. 在**密码**文本框中，输入 AzurePa\$\$w0rd。
 - iii. 然后单击**确定**按钮。
16. 在**远程桌面连接**窗口中，执行如下步骤：
 - a. 在**证书名称**中，确认证书名称与你的虚拟机名称匹配。
 - b. 选中**不再询问我是否连接到此计算机**。
 - c. 然后单击**是**按钮。
 - d. 如果弹出**Networks** 对话框，单击**No** 按钮。

结果: 完成此练习后，您将获得一个存储在新的存储帐户下的新虚拟机。

练习 3：配置虚拟机开发环境

► 任务 1：创建 AllFiles 驱动器

1. 确认服务管理器已打开，通过在 **Start** 屏幕中，单击 **Server Manager** 磁贴打开服务管理器。
2. 在 **Server Manager** 的左侧栏，选择 **Local Server**。
3. 在 **Properties** 窗口中，找到 **IE Enhanced Security Configuration**，单击 **On**。
4. 在 **Internet Explorer Enhanced Security Configuration** 窗口中，执行如下步骤：
 - a. 在 **Administrators** 选项中，单击 **Off**。
 - b. 在 **Users** 选项中，单击 **Off**。
 - c. 然后单击 **OK** 按钮。
5. 在键盘上按 Windows 键+W 键打开 **Search – Settings**。
6. 在 **Search** 文本框中，输入 **disk**。
7. 单击 **Create and format hard disk partitions**。
8. 在 **Initialize Disk** 窗口中，执行如下步骤：
 - a. 确认 **Disk 2** 被选中作为初始化的磁盘。
 - b. 确认 **MBR (Master Boot Record)** 被选中作为磁盘形式。
 - c. 然后单击 **OK** 按钮。
9. 在 **Disk Management** 窗口的下半部分中，执行如下步骤：
 - a. 找到新初始化的磁盘 **Disk 2**。
 - b. 右键单击未分配的磁盘，选择 **New Simple Volume**。
10. 在 **New Simple Volume Wizard** 窗口中，执行如下步骤：
 - a. 单击 **Next** 按钮。
 - b. 确认 **Simple volume size in MB** 中的数值为 **6141**。
 - c. 单击 **Next** 按钮。
 - d. 在 **Assign the following drive letter** 中，选择 **F**。
 - e. 单击 **Next** 按钮。
 - f. 确认 **File System** 中的数值为 **NTFS**。
 - g. 在 **Volume Label** 文本框中，更改数值为 **AllFiles**。
 - h. 单击 **Next** 按钮。
 - i. 单击 **Finish** 按钮关闭窗口并格式化磁盘 2。



注释：在弹出的 You need to format the disk in drive F: before you can use it。窗口中，单击 **Cancel** 按钮。

11. 在 **Start** 屏幕中，单击 **Internet Explorer** 磁贴。
12. 如果弹出 **Set up Internet Explorer 11** 界面，执行如下步骤：
 - a. 选择 **Use recommended security, privacy, and compatibility settings**。
 - b. 单击 **OK** 按钮。

13. 打开 <http://www.microsoft.com/learning/companionmoc>。
14. 滚动屏幕并找到 28532B 课程。
15. 单击 **28532B-CHS-AllFiles.exe** 以下载自解压包。
16. 在 Internet Explorer 下载对话框中，单击 **Run** 按钮。下载自解压包大约花费 5 分钟时间。
17. 在 **Official Microsoft Learning Product License Terms** 窗口中，单击 **Accept**。
18. 在 **WinRAR self-extracting archive** 窗口中，执行如下步骤：
 - a. 在 **Destination folder** 中，输入数值 F:\。
 - b. 单击 **Extract**。
19. 等待直到解压过程完成。

► 任务 2：安装 Visual Studio 2013 Ultimate Update 4

1. 在 Start 屏幕中，单击 Internet Explorer 磁贴。
2. 在地址栏中，输入 <http://go.microsoft.com/fwlink/?LinkId=525334>。
3. 按 Enter 键。
4. 在 select language 下拉框中，选择 Chinese(Simplified)，然后等待页面刷新并显示中文。
5. 单击下载按钮。
6. 选择 vs_ultimate.exe。
7. 单击 **Next** 按钮。

 **注释：**若弹出 Internet Explorer blocked a pop-up IE 对话框窗口，则选择 Options for this Site，单击 Always allow 按钮。

8. 在 Internet Explorer 下载对话框中，单击 **Run** 按钮。
9. 在 Visual Studio 初始化窗口中，选中**我同意许可条款和隐私策略**。
10. 单击**下一步**按钮。
11. 在**要安装的可选功能**中，确认**有且只有**如下三个选项被选中：
 - Microsoft Office 开发人员工具
 - Microsoft SQL Server Data Tools
 - Microsoft Web 开发人员工具
12. 单击**安装**按钮。

 **注释：**下载和安装选择功能的过程大约花费 10 到 12 个小时的时间，具体花费时间与当前网络性能有关。

13. 等待直到 Visual Studio 2013 Ultimate 安装过程结束。
14. 在 **Visual Studio Ultimate 2013** 窗口中，单击**启动**。
15. 在 **Visual Studio 欢迎使用。请登录 Visual Studio** 窗口中，单击**以后再说**。
16. 在 Visual Studio 窗口中，执行如下步骤：
 - a. 在开发设置下拉框中，选择 Visual C#。

- b. 然后单击启动 Visual Studio。
- c. 等待直到 Visual Studio 的配置完成。
- d. 单击右上角的关闭(X)按钮来关闭 Visual Studio 2013 Ultimate 窗口。

► **任务 3：安装 Azure SDK for .Net 2.5**

1. 在 Start 屏幕中，单击 **Internet Explorer** 磁贴。
2. 在地址栏中，输入 <http://go.microsoft.com/fwlink/?LinkId=525337>。
3. 按 Enter 键。
4. 在 **select language** 下拉框中，选择 **Chinese(Simplified)**，然后等待页面刷新并显示中文。
5. 展开**详情**节点。
6. 滚动屏幕，并找到 VS 2013。然后单击 VS 2013。
7. 在 Internet Explorer 下载对话框中，单击 **Run** 按钮。

 **注释：**首先需要 1 到 2 分钟时间来安装 Web Platform Installer 5.0，然后通过该软件来获取 Azure SDK for .NET - 2.5。

8. 确认 Web Platform Installer 5.0 安装的软件名称为 **Microsoft Azure SDK for .NET(VS 2013) - 2.5**。
9. 单击 **Install**。
10. 单击 **I Accept**。

 **注释：**下载并安装 Azure SDK 的时间大约要 5 分钟。

11. 请等待，直到 Web Platform Installer 的安装过程完成。
12. 单击 **Continue** 按钮在浏览器中打开 Azure for .NET Developers 页面。
13. 关闭 Internet Explorer。
14. 在 Web Platform Installer 5.0 窗口中，单击 **Finish** 按钮。
15. 单击 **Exit** 按钮。
16. 在 Start 屏幕，单击屏幕底端的向下箭头以显示所有应用，向右滚动屏幕找到 Visual Studio 2013。
17. 右键单击 Visual Studio 2013，然后选择 Pin to Start。

► **任务 4：安装 Azure PowerSell Module**

1. 在 Start 屏幕中，单击 **Internet Explorer** 磁贴。
2. 在地址栏中，输入 <http://go.microsoft.com/fwlink/p/?linkid=320376>。
3. 按 Enter 键。

 **注释：**若弹出 Internet Explorer blocked a pop-upIE 对话框窗口，则选择 Options for this Site，单击 Always allow 按钮。

4. 在 Internet Explorer 下载对话框中，单击 **Run** 按钮。

5. 确认 Web Platform Installer 5.0 安装的软件名称为 **Microsoft Azure PowerShell with Microsoft Azure SDK**。
6. 单击 **Install** 按钮。
7. 单击 **I Accept** 按钮。

 **注释:** 下载并安装 Azure Module 的时间大约要 5 分钟。

8. 请等待，直到 Web Platform Installer 的安装过程完成。
9. 在 Web Platform Installer 5.0 窗口中，单击 **Finish** 按钮。
10. 单击 **Exit** 按钮。
11. 关闭 Internet Explorer 应用。

结果: 完成此练习后，您的开发虚拟机上将安装 Visual Studio、Azure PowerShell 和 Azure SDK。

练习 4：连接到 Azure 订阅

► 任务 1：下载订阅文件

- 在 Start 屏幕，单击屏幕底端的向下箭头以显示所有应用，向右滚动屏幕找到 Microsoft Azure PowerShell。
- 单击 **Microsoft Azure PowerShell**。
- 在打开的 Microsoft Azure PowerShell 窗口中，输入如下命令：

```
Get-AzurePublishSettingsFile -environment azurechinacloud
```

- 在 Sign-In 页面，输入您在世纪互联的 Azure 帐户和密码，并单击 **Sign In** 按钮。
- 验证系统打开浏览器，并进入 **Your subscription file is being generated, and the download will begin shortly** 页面。
- 在浏览器下方弹出的保存文件窗口中，选择 **Save**，单击 **Save as**。
- 在 Save as 对话框中，保存您的订阅文件到系统的 Downloads 目录。
- 关闭 Internet Explorer 窗口。

► 任务 2：在 Visual Studio 2013 中连接到 Azure 帐户

- 在 Start 屏幕，单击 Visual Studio 2013 磁贴。
- 单击视图，单击服务器资源管理器。
- 找到 Azure 节点，右键单击 Azure，单击管理订阅，选择证书选项卡，单击导入按钮。
- 在导入 Microsoft Azure 订阅对话框中，单击浏览，在 Open 对话框中浏览 Downloads 目录，选中您下载的订阅文件，单击 **Open**，返回导入 Microsoft Azure 订阅对话框，单击导入按钮。
- 等待证书导入完成，单击关闭按钮。
- 关闭 Visual Studio 2013 窗口。

► 任务 3：在 Microsoft Azure PowerShell 中连接到 Azure 帐户

- 在 Start 屏幕，单击屏幕底端的向下箭头以显示所有应用，向右滚动屏幕找到 Microsoft Azure PowerShell。
- 单击 **Microsoft Azure PowerShell**。
- 在打开的 Microsoft Azure PowerShell 窗口中，输入如下命令：

```
Import-AzurePublishSettingsFile [订阅文件]
```



注释：订阅文件的格式如 **C:\Users\Student\Downloads\1RMB Trial Offer-3-2-2015-credentials.publishsettings**

- 输入以下命令，并验证您的 Azure 帐户信息导入成功：

```
Get-AzureAccount
```

- 关闭 Microsoft Azure PowerShell 窗口。

注释：至此，Azure 虚拟机开发环境你已经搭建完成了。

结果: 完成这个 Lab 后, 您将了解如何在本地通过 Visual Studio 2013 和 Azure PowerShell 连接到 Azure 订阅。

第 3 章：在 Azure 平台上托管 Web 应用程序

实验 A：使用 Azure 网站服务创建 ASP.NET 网站

练习 1：创建 Azure 网站

► 任务 1：使用管理门户创建网站实例

1. 在 Start 屏幕中，单击 **Internet Explorer** 磁贴。
2. 转到 <https://manage.windowsazure.com>。
3. 在 email 地址框中，输入 Microsoft 帐户的 email 地址。
4. 在密码框中，输入 Microsoft 帐户的密码。
5. 单击 **Sign In**。

 **注释：**如果这是您第一次登录到管理门户，您可能会收到一个“Welcome”提示对话框。您可以关闭此对话框并继续。

6. 单击右上角的帐户名前的地球图标，选择中文（简体）。
7. 在屏幕左侧的导航窗格中，单击**网站**。
8. 单击屏幕左下角的**+新建**按钮。
9. 单击**自定义创建**。
10. 在显示的窗体，执行以下步骤：
 - a. 在**Web 宿主计划**列表中，选择**创建新的 Web 宿主计划**。
 - b. 在**区域**列表中，选择最接近您所在位置的区域。
 - c. 在**URL** 框中，为网站创建一个名称。
 - d. 在**数据库**列表，选择**创建免费的 20 MB SQL 数据库**。
 - e. 在**数据库连接字符串名称**框中，输入**EventsContextConnectionString**。
 - f. 单击下一步箭头按钮。
 - g. 在**名称**框中，输入**EventsContextDB**。
 - h. 在**服务器**框中，单击**新建 SQL 数据库服务器**。
 - i. 在**服务器登录名称**框中，创建 SQL 管理员用户的用户名。
 - j. 在**服务器登录密码**和**确认密码**对话框中，为 SQL 管理员用户创建密码。
 - k. 单击完成按钮来创建你的网站和 SQL 数据库。

► 任务 2：转至新创建的网站占位符页面

 **注释：**创建网站通常是非常快的，但在可以转去网站前，你必须等创建网站运行成功。当网站运行时，其在**网站状态**列表中显示为**正在运行**。

1. 在左侧导航页面，单击**网站**。

2. 在**网站**列表中，单击刚刚创建的网站。
3. 在屏幕的顶部，单击**仪表板**选项卡。
4. 定位在仪表板右侧的**速览**部分。
5. 在**站点 URL** 标题下，单击超链接转到网站。
6. 验证该网站确实存在。
7. 关闭显示该网站的选项卡。

结果：完成此练习后，您就会使用管理门户来创建网站实例。

练习 2：将 ASP.NET Web 应用程序部署到 Azure 网站

► 任务 1：使用 Visual Studio 2013 打开现有的 ASP.NET Web 应用程序项目

1. 在 Start 屏幕中，单击 **Desktop** 磁贴。
2. 在任务栏上，单击 **File Explorer** 图标。
3. 在 This PC 窗口，转到 **Allfiles (F):\Mod03\Labfiles\Starter\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Management** 项目，然后单击 **设为启动项目**。
5. 在调试菜单上，单击 **启动调试**。

 **注释：**如果这是您第一次生成此解决方案，NuGet 将隐式地恢复所有丢失的包，而无需您手动的恢复。

6. 在 Web 应用程序的主页，确认在 **Latest 3 Events** 标题下，显示三个事件。
7. 在网页顶部的导航栏中，单击 **Events**。
8. 确认在事件页面中显示事件列表。
9. 关闭显示该网站的选项卡。

► 任务 2：下载 Azure 网站的发布配置文件

1. 在 Start 屏幕中，单击 **Internet Explorer** 磁贴。
2. 转到 <https://manage.windowsazure.com>。
3. 在 email 地址框中，输入 Microsoft 帐户的 email 地址。
4. 在密码框中，输入 Microsoft 帐户的密码。
5. 单击 **Sign In**。
6. 在网站列表中，单击刚刚创建的网站。
7. 单击 **仪表板** 选项卡。
8. 定位在仪表板右侧的 **速览** 部分。
9. 单击下载发布配置文件。
10. 在下载对话框中，单击 **Save** 按钮右侧箭头，然后单击 **Save As**。
11. 在 **Save As** 对话框中，转到 **Allfiles (F):\Mod03\Labfiles**，然后单击 **Save**。

► 任务 3：将 ASP.NET Web 应用程序发布到 Azure 网站

1. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Management**，然后单击 **发布**。
2. 在发布网页窗口中，单击 **导入**。
3. 在导入发布设置对话框中，单击 **浏览**。
4. 在导入发布设置窗口中，转到 **Allfiles (F):\Mod03\Labfiles**，然后双击之前保存的发布配置文件。
5. 单击 **确定**。
6. 确认 **站点名称** 框中的名字匹配网站的名字。

7. 单击发布。

 **注释:** 如果弹出保存只读文件消息框, 单击**覆盖**。

► **任务 4: 验证 Web 应用程序发布成功**

 **注释:** 如果在“练习 1 – 任务 2”显示占位符页, 这可能意味着客户端正在缓存中。您可以随时按 Ctrl+ F5 强制刷新浏览器和缓存。

1. 在 Web 应用程序的主页, 确认在 **Latest 3 Events** 标题下, 显示三个事件。
2. 在网页顶部的导航栏中, 单击 **Events**。
3. 确认在事件页面中显示事件列表。
4. 关闭显示该网站的选项卡。

结果: 完成此练习后, 您就会使用发布配置文件将 Web 应用程序直接发布到网站上。

练习 3：配置 Azure 网站

► 任务 1：实现从应用设置中读取配置设置的逻辑

1. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在**解决方案资源管理器**窗格中，展开**Contoso.Events.ViewModels** 项目。
2. 在**解决方案资源管理器**窗格中，双击**EventsListViewModel.cs**。
3. 定位到 EventsListViewModel.cs 中的第 20 行代码，如下所示：

```
this.EventCount = 3;
```

4. 用以下的代码替换之前那行代码：

```
this.EventCount =  
Int32.Parse(ConfigurationManager.AppSettings["EventsListViewModel.EventCount"]);
```

5. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在**解决方案资源管理器**窗格中，展开**Contoso.Events.Management** 项目。
6. 在**解决方案资源管理器**窗格中，双击**Web.config**。
7. 在第 14 行，**appSettings** 元素中，添加以下代码：

```
<add key="EventsListViewModel.EventCount" value="5" />
```

8. 在**调试**菜单上，单击**启动调试**。
9. 在 Web 应用程序的主页，确认在**Latest 5 Events** 标题下，显示五个事件。
10. 关闭显示该网站的选项卡。

► 任务 2：将 Web 应用程序发布到 Azure 网站

1. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在**解决方案资源管理器**窗格中，右键单击**Contoso.Events.Management**，然后单击**发布**。
2. 单击**发布**。
3. 在 Web 应用程序的主页，确认在**Latest 5 Events** 标题下，显示五个事件。
4. 关闭显示该网站的选项卡。

► 任务 3：在管理门户中修改应用设置

1. 在 Start 屏幕中，单击**Internet Explorer** 磁贴。
2. 转到 <https://manage.windowsazure.com>。
3. 在 email 地址框中，输入 Microsoft 帐户的 email 地址。
4. 在密码框中，输入 Microsoft 帐户的密码。
5. 单击**Sign In**，在**网站**列表中，单击刚刚创建的网站。
6. 单击**配置**选项卡。
7. 向下滚动到**应用设置**部分。
8. 在**密钥**框中，输入**EventsListViewModel.EventCount**。
9. 在**值**框中，输入**2**。
10. 单击**保存**。



注释: 更新配置设置后，必须等待几秒钟再查看配置变化。当**配置**选项卡再次可编辑时，说明配置更改生效了。

► **任务 4：验证应用设置已成功更新**

1. 在屏幕的顶部，单击**仪表板**选项卡。
2. 定位在仪表板右侧的**速览**部分。
3. 在**站点 URL** 标题下，单击超链接转到网站。
4. 在 Web 应用程序的主页，确认在 **Latest 2 Events** 标题下，显示两个事件。
5. 关闭显示该网站的选项卡。
6. 关闭 **Internet Explorer** 应用程序。
7. 关闭 **Contoso.Events - Microsoft Visual Studio** 应用程序。

结果: 完成此练习后，您就会使用 Web.config 文件和管理门户来配置自定义应用设置。

实验 B：在中文版 Windows Azure 中创建 ASP.NET 网站

练习 1：创建 Azure 网站

► 任务 1：在 Azure 管理门户中创建网站实例

1. 在 Start 屏幕上，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的**地球**图标，并在菜单中选择**中文(简体)**语言。
5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击**网站**。
6. 在页面下方的工具栏左侧，单击**新建**按钮。
7. 在弹出的新建向导中，单击**网站**，单击**自定义创建**。
8. 在弹出的**创建网站**窗口中，执行以下步骤：
 - a. 在**URL**文本框中，输入**ws28532[自定义名称]**。
 - b. 在**Web宿主计划**列表中，选择**创建新的 Web宿主计划**。
 - c. 在**区域**下拉框中，选择一个离你位置较近的**区域**。
 - d. 在**数据库**下拉框中，选择**创建新的 SQL数据库**。
 - e. 在**数据库连接字符串名称**文本框中，修改其数值为**EventsContextConnectionString**。
 - f. 在窗口右下角，单击**下一步**按钮。
9. 在**指定数据库设置**窗口中，执行如下步骤：
 - a. 在**名称**文本框中，修改其数值为**EventsContextDB**。
 - b. 在**服务器**下拉框中，选择**新建 SQL数据库服务器**。
 - c. 在**服务器登录名**文本框中，输入**Student**。
 - d. 在**服务器登录密码**和**确认密码**文本框中，输入**TestPa\$\$w0rd**。
 - e. 在**区域**下拉框中，选择一个离你位置较近的**区域**。
 - f. 在窗口右下角，单击**完成**按钮。

► 任务 2：访问新创建的空网站

 **注释：** 创建网站不会花费太多时间，请耐心等待直到网站创建完成，这时，网站的状态会显示为正在运行。

1. 在网站页面，单击进入您新创建的网站。
2. 在新建的网站页面，单击页面上方的**仪表板**选项卡。
3. 滚动屏幕找到**速览**，在**速览**下方，找到**站点 URL**，单击下方的站点链接（链接格式如[website url].chinacloudsites.cn）。
4. 确认页面显示该网站已创建成功。

5. 关闭显示该网站的 IE 选项卡。

结果: 完成此练习后, 您就会使用管理门户来创建网站实例。

MCT USE ONLY. STUDENT USE PROHIBITED

练习 2：将 ASP.NET Web 应用程序部署到 Azure 网站

► 任务 1：在 Visual Studio 2013 中打开 ASP.NET Web 应用程序项目

1. 在 Start 屏幕中，单击 **Desktop** 磁贴。
2. 在任务栏上，单击 **File Explorer** 图标。
3. 在 This PC 窗口，转去 **Allfiles (F):\Mod03\Labfiles\Starter\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Management** 项目，然后单击 **设为启动项目**。
5. 在调试菜单上，单击 **启动调试**。

 **注释：**如果这是您第一次生成此解决方案， NuGet 将隐式地恢复所有丢失的包。您不必手动恢复丢失的包。

6. 在 Web 应用程序的主页，确认在 **Latest 3 Events** 标题下，显示三个事件。
7. 在网页顶部的导航栏中，单击 **Events**。
8. 确认在事件页面中显示事件列表。
9. 关闭显示该网站的选项卡。

► 任务 2：下载 Azure 网站的发布配置文件

1. 在 Start 屏幕上，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的地球图标，并在菜单中选择 **中文(简体)** 语言。
5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击 **网站**。
6. 在网站页面，单击并进入新创建的网站 **ws28532[自定义名称]**。
7. 单击页面上方的 **仪表板** 选项卡。
8. 滚动屏幕找到 **速览**，在 **速览** 下方，单击 **下载发布配置文件** 链接。
9. 在 IE 下载对话框中，单击 **Save** 按钮右侧的箭头，然后单击 **Save As**
10. 在 **Save As** 对话框中，定位到 **Allfiles (F):\Mod03\Labfiles**，然后单击 **Save**。

► 任务 3：发布 ASP.NET Web 应用程序到 Azure 网站

1. 在 **Contoso.Events - Microsoft Visual Studio** 窗口中，在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Management**，然后单击 **发布**。
2. 在发布网页窗口中，单击 **导入**。
3. 在导入发布设置对话框中，单击 **浏览**。
4. 在导入发布设置窗口中，转到 **Allfiles (F):\Mod03\Labfiles**，然后双击之前保存的发布配置文件。
5. 单击 **确定**。
6. 确认 **站点名称** 框中的名字匹配网站的名字。
7. 单击 **发布**。

 **注释:** 如果弹出保存只读文件消息框，单击覆盖。

► **任务 4: 验证 Web 应用程序发布成功**

 **注释:** 如果在 **练习 1 – 任务 2** 显示占位符页，这可能意味着客户端正在缓存中。您可以随时按 Ctrl+ F5 强制刷新浏览器和缓存。

1. 在 Web 应用程序的主页，确认在 **Latest 3 Events** 标题下，显示三个事件。
2. 在网页顶部的导航栏中，单击 **Events**。
3. 确认在事件页面中显示事件列表。
4. 关闭显示该网站的选项卡。

结果: 完成此练习后，您就会使用发布配置文件将 Web 应用程序直接发布到网站上。

练习 3：配置 Azure 网站

► 任务 1：实现从应用设置中读取配置信息

1. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在**解决方案资源管理器**窗格中，展开**Contoso.Events.ViewModels** 项目。
2. 在**解决方案资源管理器**窗格中，双击**EventsListViewModel.cs**。
3. 定位到位于 EventsListViewModel.cs，第 20 行的以下代码：

```
this.EventCount = 3;
```

4. 用以下的代码替换之前那行代码：

```
this.EventCount =  
Int32.Parse(ConfigurationManager.AppSettings["EventsListViewModel.EventCount"]);
```

5. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在**解决方案资源管理器**窗格中，展开**Contoso.Events.Management** 项目。
 6. 在**解决方案资源管理器**窗格中，双击**Web.config**。
 7. 在第 14 行，**appSettings** 元素中，添加以下代码：
- ```
<add key="EventsListViewModel.EventCount" value="5" />
```
8. 在**调试**菜单上，单击**启动调试**。
  9. 在 Web 应用程序的主页，确认在**Latest 5 Events** 标题下，显示五个事件。
  10. 关闭显示该网站的选项卡。

### ► 任务 2：发布 Web 应用程序到 Azure 网站

1. 在 Contoso.Events - Microsoft Visual Studio 窗口中，在**解决方案资源管理器**窗格中，右键单击**Contoso.Events.Management**，然后单击**发布**。
2. 单击**发布**。
3. 在 Web 应用程序的主页，确认在**Latest 5 Events** 标题下，显示五个事件。
4. 关闭显示该网站的选项卡。

### ► 任务 3：在 Azure 管理门户中修改应用设置

1. 在 Start 屏幕上，单击**Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击**Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的**地球**图标，并在菜单中选择**中文(简体)**语言。
5. 在 Azure 管理门户左侧的导航栏中，向下滚动，然后单击**网站**。
6. 在网站页面，单击并进入新创建的网站 **ws28532[自定义名称]**。
7. 单击**配置**选项卡。
8. 滚动屏幕找到**应用设置**。
9. 在应用设置下方，添加如下键值信息：
  - a. 在**密钥**文本框中，输入**EventsListViewModel.EventCount**。
  - b. 在**值**文本框中，输入**2**。

- c. 在页面下方的工具栏中，单击**保存**按钮。



**注释:** 更新配置设置时，需要等待几秒钟使更改生效。当编辑框再次可使用时，表明更改已经生效。

► **任务 4：验证应用设置被成功更新**

1. 单击页面上方的**仪表板**选项卡。
2. 滚动屏幕找到**速览**，在**速览**下方，找到**站点 URL**，单击下方的站点链接（链接格式如[website url].chinacloudsites.cn）。
3. 在 Web 应用程序的 home 页面，在 **Latest 2 Events** 下方的列表中显示两条 Events 数据。
4. 关闭显示该网站的选项卡。
5. 关闭 **Internet Explorer** 应用程序。
6. 关闭 **Contoso.Events - Microsoft Visual Studio** 应用程序。

**结果:** 完成此练习后，您就会使用 Web.config 文件和管理门户来配置自定义应用设置。

## 第 4 章：在 Azure 中存储 SQL 数据

# 实验 A：在 Azure SQL 数据库中存储事件数据

### 练习 1：创建 Azure SQL 数据库实例

#### ► 任务 1：登录到 Azure 预览门户

1. 在 Start 屏幕中，单击 **Internet Explore** 磁贴。
2. 访问 <https://portal.azure.com>。
3. 在 Email 地址框中，输入 Microsoft 帐户的 Email 地址。
4. 单击 **Continue**。
5. 在密码框中，输入 Microsoft 帐户的密码。
6. 单击 **Sign In**。
7. 单击右上角的帐户名。
8. 单击 **Settings**。
9. 单击 **LANGUAGE**，选择中文（简体），然后单击 **Select**。

#### ► 任务 2：使用预览门户创建 Azure SQL 数据库

1. 在界面左侧的导航窗格里，单击**浏览**。
2. 在显示的**浏览**边栏选项卡中，单击**SQL 数据库**。
3. 在门户网站左下角，单击**新建**。
4. 在显示的**创建**边栏选项卡中，单击**数据+存储器**。
5. 在显示的**数据+存储器**边栏选项卡中，单击**SQL Database**。
6. 在显示的**SQL 数据库**边栏选项卡中，执行以下步骤：
  - a. 在名称框中，输入 **db28532[自定义名称]**。
  - b. 单击**定价层**。
  - c. 在显示的**推荐定价层**边栏选项卡中，选择 **B Basic**。
  - d. 单击**选择**。
  - e. 单击**选择源**。
  - f. 选择**空白数据库**选项。
  - g. 在显示的**SQL 数据库**边栏选项卡中，单击**服务器**。
  - h. 在显示的**服务器**边栏选项卡中，单击**创建一个新服务器**。
  - i. 在显示的**新服务器**边栏选项卡中，定位到**服务器名**输入框。
  - j. 在**服务器名**框中，输入 **sv28532[自定义名称]**。
  - k. 在**服务器管理登录**框中，输入 **testuser**。
  - l. 在**密码**框中，输入 **TestPa\$\$w0rd**。
  - m. 在**确认密码**框中，输入 **TestPa\$\$w0rd**。

- n. 单击**位置**。
  - o. 在显示的**位置**边栏选项卡中，选择最接近你位置的区域。
  - p. 在**新服务器**边栏选项卡中，单击**确认**。
  - q. 在**SQL 数据库**边栏选项卡中，单击**创建**来创建 SQL 数据库和服务器。
7. 记下新 SQL 数据库的名字。

**结果：**完成此练习后，您将在 SQL 数据库服务中创建好服务器和数据库。

MCT USE ONLY. STUDENT USE PROHIBITED

## 练习 2：结合 Azure SQL 数据库使用实体框架

► 任务 1：运行 ASP.NET Web 应用程序查看本地 SQL 数据库中的事件

1. 在 Start 屏幕中，单击 **Desktop** 磁贴。
2. 在任务栏中，单击 **File Explorer** 图标。
3. 在 This PC 窗口中，访问路径 **Allfiles (F):\Mod04\Labfiles\Starter\SQL\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.DataGeneration** 项目，指向调试，然后单击启动新实例。

 **注释：**运行 Data Generation 脚本大概需要 1 到 2 分钟。

5. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Cloud** 项目，然后单击设为启动项目。
6. 在调试菜单中，单击启动调试。
7. 在 web 应用程序的主页，验证显示了多个事件的列表。
8. 关闭显示当前网站的选项卡。

► 任务 2：用新的 **DatabaseInitializer** 配置 **DbContext**

1. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Data** 项目，指向添加，然后单击新项目。
2. 在添加新项对话框中，执行以下步骤：
  - a. 展开已安装，展开 **Visual C# 项**，然后单击代码。
  - b. 单击类模板。
  - c. 在名称框中，输入 **EventsContextInitializer.cs**。
  - d. 单击添加。
3. 在 **EventsContextInitializer** 类中，在类定义的左侧添加 **public** 访问器：

```
class EventsContextInitializer
```

4. 确认更新后类的定义如以下代码所示：

```
public class EventsContextInitializer
```

5. 在 **EventsContextInitializer** 类中，在类定义的右侧，添加继承声明语句：  
**DropCreateDatabaseAlways<EventsContext>**

```
public class EventsContextInitializer
```

6. 确认更新后类的定义如以下代码所示：

```
public class EventsContextInitializer : DropCreateDatabaseAlways<EventsContext>
```

7. 在代码文件顶端，添加 **using** 语句：

```
using System.Data.Entity;
```

8. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Data** 项目。
9. 在 **Contoso.Events.Data** 项目中，打开 **EventsContext.cs** 文件。

10. 在静态构造函数 **static EventsContext()** 中，添加以下代码：

```
Database.SetInitializer<EventsContext>(new EventsContextInitializer());
```

11. 保存 EventsContext.cs 文件。

 **注释：**如果弹出文件被写保护的消息框，单击覆盖。

#### ► 任务 3：用 **DbContext** 生成基本数据

1. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Data** 项目。
2. 在 **Contoso.Events.Data** 项目中，打开 **EventsContextInitializer.cs** 文件。
3. 把下面方法添加到 **EventsContextInitializer** 类：

```
protected override void Seed(EventsContext context)
{
}
```

4. 在代码文件顶端，添加下面 **using** 语句：

```
using Contoso.Events.Models;
```

5. 将光标置于 **Seed(EventsContext context)** 方法右括号之前，添加下列代码：

```
Event eventItem = new Event();
eventItem.EventKey = "FY17SepGeneralConference";
eventItem.StartTime = DateTime.Today;
eventItem.EndTime = DateTime.Today.AddDays(3d);
eventItem.Title = "FY17 September Technical Conference";
eventItem.Description = "Sed in euismod mi.";
eventItem.RegistrationCount = 1;
```

6. 将光标置于 **Seed(EventsContext context)** 方法右括号之前，然后添加下列代码：

```
context.Events.Add(eventItem);
```

7. 将光标置于 **Seed(EventsContext context)** 方法右括号之前，然后添加下列代码：

```
Registration registrationItem = new Registration();
registrationItem.EventKey = "FY17SepGeneralConference";
registrationItem.FirstName = "Aisha";
registrationItem.LastName = "Witt";
```

8. 将光标置于 **Seed(EventsContext context)** 方法右括号之前，然后添加下列代码：

```
context.Registrations.Add(registrationItem);
```

9. 将光标置于 **Seed(EventsContext context)** 方法右括号之前，然后添加下列代码：

```
context.SaveChanges();
```

10. 保存 EventsContextInitializer.cs 文件。

11. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Data** 项目，然后单击 **生成**。

#### ► 任务 4：将包含更新的 **DbContext** 的云应用程序发布到 Azure

1. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Web** 项目。

2. 在解决方案资源管理器窗格中，展开在 **Contoso.Events.Web** 项目中的 **Web.config** 文件。
3. 双击 **Web.Release.config** 文件。
4. 在 **Web.Release.config** 文件中，用以下值来更新连接字符串的 **EventsContextConnectionString** 值
  - a. [database]: **db28532[自定义名称]**
  - b. [login]: **testuser**
  - c. [server]: **sv28532[自定义名称]**
  - d. [password]: **TestPa\$\$w0rd**
5. 保存 **Web.Release.config** 文件。
6. 展开 **Contoso.Events.Web** 项目，双击打开 **Web.config** 文件
7. 在打开的 **Web.config** 文件中，找到 **runtime** 节点，节点代码类似于<runtime>...</runtime>
8. 在 **runtime** 节点内，找到第一个节点，名称为 **assemblyBinding**，节点代码类似于
 

```
<assemblyBinding
 xmlns="urn:schemas-microsoft-com:asm.v1">
```
9. 在该代码后，插入如下代码：

```
<dependentAssembly>
 <assemblyIdentity name="WindowsAzureEventSource" publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
 <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
 <assemblyIdentity name="Microsoft.WindowsAzure.ServiceRuntime"
publicKeyToken="31BF3856AD364E35" culture="neutral"/>
 <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
 <assemblyIdentity name="msshrtmi" publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
 <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
 <assemblyIdentity name="WindowsAzureTelemetryEvents"
publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
 <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
```

10. 保存 **Web.config** 文件。
11. 如果弹出 **保存只读文件** 对话框，请单击 **覆盖** 按钮
12. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Cloud** 项目，然后单击 **发布**。
13. 在发布 **Azure** 应用程序对话框中，执行以下步骤：
  - a. 若之前没有选择任何订阅，请执行以下步骤：
    - i. 如若需要，用 Microsoft 帐户登录。
    - ii. 在 **选择订阅** 列表里，选择 Azure 订阅。
    - iii. 单击 **下一步**。
  - b. 若之前已选择了订阅，将直接导向 **Windows Azure** **发布设置** 的窗口。
14. 打开 **创建云服务和存储帐户** 对话框。

- a. 若没有任何已存在的云服务，将会自动显示**创建云服务和存储帐户**对话框。
  - b. 若有已存在的云服务，定位到**云服务**列表，选择**新建**打开**创建云服务**对话框。
15. 在**创建云服务和存储帐户**对话框，执行以下步骤：
- a. 在名称框中，输入 **cs28532[自定义名称]**。
  - b. 在**地区或地缘组**列表中，选择离你所在位置最接近的区域。
  - c. 单击**创建**。
16. 在**Microsoft Azure 发布设置**对话框中，执行以下步骤：
- a. 保留所有字段的默认值。
  - b. 单击**发布**。

 **注释：**发布过程通常需要 5 到 10 分钟才完成。当发布云服务项目时，可以通过 Microsoft Azure 活动日志窗格来跟踪进度。

#### ► 任务 5：验证 Azure 云服务网站正在使用新数据

1. 等待发布过程完成，完成后控制台窗口会显示**已完成**状态。

 **注释：**当 Microsoft Azure 活动日志窗格显示“**已完成**”信息时表明发布过程已完成。活动日志中的绿色圆形指示器并不能表明发布过程已完成，只能表明文件包已上传成功。

2. 在 Microsoft Azure 活动日志控制台中，单击指向已发布的 web 应用程序的超链接。
3. 验证网站上显示着在 context initializer 实体框架中创建的单个事件。

#### ► 任务 6：登录到 Azure 管理门户

1. 在 Start 屏幕中，单击**Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.com>。
3. 在 Email 地址框，输入 Microsoft 帐户的 Email 地址，在密码框，输入 Microsoft 帐户的密码，然后单击**Sign In**。
4. 单击右上角的帐户名前的地球图标，选择**中文（简体）**选项。

#### ► 任务 7：查看 Azure SQL 管理门户中迁移的数据

1. 在左侧导航窗格中，单击**SQL 数据库**。
2. 在屏幕顶部，单击**数据库**。
3. 在**SQL 数据库**列表中，单击前缀为 **db28532** 的 SQL 数据库名称。
4. 在屏幕顶部，单击**仪表板**。
5. 在页面底部，单击在**VISUAL STUDIO** 中打开。
6. 在对话框中，提示是否将当前数据库 IP 地址加入到防火墙规则中，单击**是**。

 **注释：**若 Internet Explorer 底部有黄色对话框弹出，单击**Options for this Site**，然后单击**Always Allow**，你将不受限制的查看任何对话框。完成这些步骤之后，Azure 管理门户将被更新。你可以单击在**VISUAL STUDIO** 中打开来恢复实验。



**注释：**有些版本的 windows 服务器虚拟机没有安装 Silverlight 客户端，若提示安装，请安装它。

7. 在确认对话框中，单击**打开**。
8. 在显示的 Internet Explorer 对话框中，单击**Allow**。
9. 在**连接到服务器**对话框中，执行以下步骤：
  - a. 在**登录名**框中，输入**testuser**。
  - b. 在**密码**框中，输入**TestPa\$\$w0rd**。
  - c. 单击**连接**。
10. 展开前缀为**db28532** 的 SQL 数据库，展开表，右键单击**dbo.Events**，然后单击**查看数据**。
11. 在**dbo.Events** 表中，查看单个记录。
12. 关闭**Microsoft Visual Studio** 应用程序。

**结果：**完成本练习之后，你将配置好 Entity Framework 来初始化新数据库，新数据库中有种子(Seed)数据。

# 实验 B：使用中国版 Windows Azure 在 Azure SQL 数据库中存储事件数据

## 练习 1：创建 Azure SQL 数据库实例

### ► 任务 1：登录 Azure 管理门户

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 进入 <https://manage.windowsazure.cn>。
3. 在邮箱帐户输入框，输入你的邮箱帐户地址；在密码输入框输入密码。
4. 单击 **Sign In** 登录。
5. 单击右上角的帐户名前的地球图标，选择中文（简体）选项。

### ► 任务 2：通过管理门户创建 Azure SQL 数据库

1. 在管理界面左下角，单击**新建**。
2. 在**新建**窗格下，首先在左侧栏目中单击**数据服务**，而后在中间弹出的选项中选择**SQL 数据库**，然后在右边弹出的选项中选择**自定义创建**。
3. 在**新建 SQL 数据库**的设置页面中，请按照如下步骤创建 SQL 数据库实例：
  - a. 在名称输入框输入 **db28532[你的名称]**。
  - b. 在**服务层**选项中，选择**BASIC**。
  - c. 在**服务器**选项中，选择**新建 SQL 数据库服务器**选项，其余选项保持默认值，单击右下方下一步按钮→进入创建服务器界面。
  - d. 在**登录名**输入框，输入 **testuser**。
  - e. 在**登录密码和确认密码**输入框，输入 **TestPa\$\$w0rd**。
  - f. 在**区域**选项中，选择一个区域，其余保持默认选项。
  - g. 最后单击右下方按钮√完成 SQL 数据库实例的创建。
4. 在**sql**数据库页面，单击进入新创建的数据库，名称类似于 **db28532[自定义名称]**。
5. 在页面顶部，单击**仪表板**选项卡。
6. 滚动屏幕找到**速览**，在**速览**下方，找到**显示连接字符串**。
7. 在弹出的连接字符串对话框中，在 ADO.NET 窗口中，复制该连接字符串。
8. 新建记事本，并将该字符串粘贴到记事本中。
9. 在记事本中，将 Password={此处为您的密码}修改为 Password=**TestPa\$\$w0rd**。

**结果：**在完成实践后，你将学会在 SQL Server 服务上创建服务器和数据库

## 练习 2：在 Azure SQL 数据库中使用 Entity Framework

► 任务 1：通过运行 ASP.NET 网站应用查看本地 SQL 数据库的事件

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏单击 **File Explorer** 图标。
3. 在 This PC 窗口中，访问 **Allfiles \Mod04\Labfiles\Starter\SQL\Contoso.Events**，然后双击文件 **Contoso.Events.sln**。
4. 在解决方案资源管理器中，右键单击项目 **Contoso.Events.DataGeneration**，选择调试选项，然后选择启动新实例。

 **注释：**数据生成脚本大概需要运行 1 到 2 分钟。

5. 在解决方案资源管理器中，右键单击项目 **Contoso.Events.Cloud**，然后单击设为启动项目选项。
6. 在调试菜单中单击启动调试选项。

 **注释：**在调试中，如果遇到 Server Error，请您执行以下步骤，然后重新启动调试：  
展开 **Contoso.Events.Web** 项目，展开引用节点。选择 **System.Web.Mvc**，在属性窗口中，修改 **复制本地** 的值为 **True**。

7. 在网站应用主页面，验证是否显示数据库事件信息。
8. 关闭当前网站的选项卡。

► 任务 2：用一个新的 **DatabaseInitializer** 配置 **DbContext**

1. 在解决方案资源管理器中，右键单击项目 **Contoso.Events.Data**，选择添加选项，而后选择新建项。
2. 在弹出的添加新项窗口中，执行以下步骤：
  - a. 展开已安装，展开 **Visual C# 项**，而后单击代码。
  - b. 单击类模板。
  - c. 在名称输入框，输入 **EventsContextInitializer.cs**。
  - d. 单击添加。
3. 在类 **EventsContextInitializer** 中，在类的定义左侧加上 **public** 访问器关键字：

```
class EventsContextInitializer
```

4. 更新的类定义应如下代码：

```
public class EventsContextInitializer
```

5. 在类 **EventsContextInitializer** 中，在类的定义右侧添加继承基类 **DropCreateDatabaseAlways<EventsContext>**：

```
public class EventsContextInitializer
```

6. 更新的类定义应如下代码：

```
public class EventsContextInitializer : DropCreateDatabaseAlways<EventsContext>
```

7. 在该文件的顶部加上以下 **using** 代码：

```
using System.Data.Entity;
```

8. 在解决方案资源管理器，展开项目 **Contoso.Events.Data**。
9. 在项目 **Contoso.Events.Data** 中，打开文件 **EventsContext.cs**。
10. 在静态构造方法 **static EventsContext()** 中，添加以下代码：

```
Database.SetInitializer<EventsContext>(new EventsContextInitializer());
```

11. 保存文件 **EventsContext.cs**。

#### ► 任务 3：用 **DbContext** 生成基本数据

1. 在解决方案资源管理器，展开项目 **Contoso.Events.Data**。
2. 在项目 **Contoso.Events.Data**，打开文件 **EventsContextInitializer.cs**。
3. 在类 **EventsContextInitializer** 中添加以下方法：

```
protected override void Seed(EventsContext context)
{
}
```

4. 在该代码文件的顶部加上以下 **using** 代码：

```
using Contoso.Events.Models;
```

5. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内，并输入以下代码：

```
Event eventItem = new Event();
eventItem.EventKey = "FY17SepGeneralConference";
eventItem.StartTime = DateTime.Today;
eventItem.EndTime = DateTime.Today.AddDays(3d);
eventItem.Title = "FY17 September Technical Conference";
eventItem.Description = "Sed in euismod mi.";
eventItem.RegistrationCount = 1;
```

6. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内，并输入以下代码：

```
context.Events.Add(eventItem);
```

7. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内，并输入以下代码：

```
Registration registrationItem = new Registration();
registrationItem.EventKey = "FY17SepGeneralConference";
registrationItem.FirstName = "Aisha";
registrationItem.LastName = "Witt";
```

8. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内，并输入以下代码：

```
context.Registrations.Add(registrationItem);
```

9. 将光标放在方法 **Seed(EventsContext context)** 最近的右括号内，并输入以下代码：

```
context.SaveChanges();
```

10. 保存文件 **EventsContextInitializer.cs**。

11. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Data**，然后单击生成选项。

► 任务 4：将包含更新的 **DbContext** 的云应用程序发布到 **Azure**

1. 在解决方案资源管理器窗格中，展开项目 **Contoso.Events.Web**。
2. 在解决方案资源管理器窗格中，展开项目 **Contoso.Events.Web** 的 **Web.config** 文件。
3. 双击文件 **Web.Release.config**。
4. 在打开的 **Web.Release.config** 文件中，找到 **connectionStrings** 节点，节点代码类似于  
`<connectionStrings> ... </connectionStrings>`。
5. 找到 **name** 属性值为 **EventsContextConnectionString** 的项目，执行如下步骤：
  - a. 将名称为 **connectionString** 属性的值更新为保存在记事本中的连接字符串。
6. 保存 **Web.Release.config** 文件。
7. 如果弹出保存只读文件对话框，请单击**覆盖**按钮。
8. 关闭记事本，不需要保存文件。
9. 展开 **Contoso.Events.Web** 项目，双击打开 **Web.config** 文件。
10. 在打开的 **Web.config** 文件中，找到 **runtime** 节点，节点代码类似于`<runtime>...</runtime>`。
11. 在 **runtime** 节点内，找到第一个节点，名称为 **assemblyBinding**，节点代码类似于  
`<assemblyBinding  
 xmlns="urn:schemas-microsoft-com:asm.v1">`。

12. 在该代码后，插入如下代码：

```
<dependentAssembly>
 <assemblyIdentity name="WindowsAzureEventSource" publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
 <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
 <assemblyIdentity name="Microsoft.WindowsAzure.ServiceRuntime"
publicKeyToken="31BF3856AD364E35" culture="neutral"/>
 <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
 <assemblyIdentity name="msshrtmi" publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
 <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
<dependentAssembly>
 <assemblyIdentity name="WindowsAzureTelemetryEvents"
publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
 <bindingRedirect oldVersion="0.0.0.0-2.5.0.0" newVersion="2.5.0.0"/>
</dependentAssembly>
```

13. 保存 **Web.config** 文件。
14. 如果弹出保存只读文件对话框，请单击**覆盖**按钮。
15. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Cloud**，然后单击**发布**。
16. 在发布 **Azure** 应用程序窗口，执行以下操作：
  - a. 如果你以前没有选择订阅，请执行以下操作：
    - i. 用微软帐户登录。
    - ii. 在选择订阅选项中，选择一个订阅。
    - iii. 单击**下一步**。

- b. 如果你以前选择了订阅，那么你将直接进入 **Microsoft Azure** 发布设置步骤。
17. 打开创建云服务和存储帐户窗口：
- a. 如果你没有存在的云服务，**创建云服务和存储帐户**窗口将会弹出。
  - b. 如果你有存在的云服务，在显示的**云服务列表**中选择**新建**。
18. 在**创建云服务和存储帐户**窗口中执行以下步骤：
- a. 在名称输入框输入 **cs28532[自定义名称]**。
  - b. 在**地区或地缘组**选择一个离你最近的区域。
  - c. 单击**创建**。
19. 在 **Microsoft Azure** 发布设置窗口中，执行以下步骤：
- a. 其他选项均保持默认值。
  - b. 单击**发布**。

 **注释：**完成云服务的发布一般需要 5 到 10 分钟。当你在发布你的云服务项目时，你可以在 Microsoft Azure 活动日志窗格跟踪发布进度。

#### ► 任务 5：验证 Azure 云服务站点是否使用新数据

1. 等待云服务发布完成，发布完成后控制台窗口将显示**已完成**信息。
-  **注释：**如果发布完成，在 **Microsoft Azure** 活动日志日志请求窗口会显示**已完成**状态信息。在活动日志窗口中，圆形绿色指示器并不表示发布完成，而是表示当前的程序包上传成功。
2. 在 **Microsoft Azure** 活动日志窗口，单击超链接地址可以直接访问你发布的网站。
  3. 验证云服务站点是否仅仅显示你在 Entity Framework context initializer 中创建的一条事件数据。

#### ► 任务 6：登录 Azure 管理门户

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 进入 <https://manage.windowsazure.cn>。
3. 在邮箱帐户输入框，输入你的邮箱帐户地址；在密码输入框输入密码。
4. 单击 **Sign In** 登录。

#### ► 任务 7：在 Azure SQL 管理门户查看迁移数据

1. 在左侧导航栏，单击 **SQL 数据库**。
2. 在界面顶部，单击**数据库**。
3. 在**SQL 数据库**列表中，单击前缀名称为 **db28532** 的数据库。
4. 在界面顶部，单击**仪表板**。
5. 在界面的**速览**栏目下单击**管理允许的 IP 地址**。
6. 在界面右侧单击按钮→将**当前客户端 IP 地址**添加到**允许的 IP 地址**列表中。
7. 单击界面下方的**保存**按钮保存更改的配置。
8. 在页面顶部显示的是数据库的**服务器**名称，在接下来的步骤中，您将使用到该名称。

9. 新建记事本，执行以下步骤：
  - a. 将以下字符串粘贴到记事本中：[https://\[your server name\].database.chinacloudapi.cn/?langid-zh-cn](https://[your server name].database.chinacloudapi.cn/?langid-zh-cn)。
  - b. 将[your server name]替换为您服务器的名称。
10. 在 Start 屏幕，单击 Internet Explorer 磁贴。
11. 输入网址，网址为您刚刚保存在记事本中的字符串。打开 **Management Portal – SQL Database** 页面。
  - a. 在 **DATABASE** 中，输入您的数据库名称，数据库名称类似于 **db28532[自定义名称]**。
  - b. 在 **USERNAME** 输入框输入帐号 **testuser**。
  - c. 在 **PASSWORD** 输入框输入 **TestPa\$\$w0rd**。
  - d. 单击 **Log On**。

 **注释：**Internet Explorer 可能会显示阻止弹出对话框警告。为了显示对话框，请在 Internet Explorer 窗口下方的警告对话框单击 **Options for this Site**，然后单击 **Always Allow**，之后该警告对话框将不再出现，你将可以访问该页面。完成上述步骤后，Azure 管理门户将刷新。

 **注释：**如果一些版本的 Windows Server 虚拟机没有安装 Silverlight，请根据提示安装 Silverlight。

12. 在 **Management Portal – SQL Database** 界面左下角，单击 **Design**。
13. 在 **Tables** 列表中，单击表 **dbo.Events**。
14. 单击该选中表右侧的 **Edit** 按钮。
15. 在该表页面的顶部单击 **Data**。
16. 在表 **dbo.Events** 中，查看到一条数据记录。
17. 关闭 **Management Portal – SQL Database** 窗口。
18. 关闭 **Internet Explorer**。

**结果：**在完成实践后，你将学会用 Entity Framework 初始化包含基本数据的新数据库。

## 第 6 章：管理 Azure 中的云服务

# 实验 A：使用 Azure 辅助角色创建后台进程

### 练习 1：创建 C#类库

#### ► 任务 1：创建 C#类库项目

1. 在开始屏幕，右键单击 **Visual Studio 2013** 磁贴。
2. 在应用程序栏，单击 **Run as administrator**。

 **注释：**如果弹出用户帐户控制（UAC）对话框，你可以提高权限以管理员身份运行 Visual Studio 2013。

3. 在 Visual Studio 起始页，单击打开项目。
4. 在打开项目对话框，浏览至 **Allfiles (F):\Mod06\Labfiles\Starter\Contoso.Events**，然后单击 **Contoso.Events.sln**。
5. 单击 **Open**。
6. 展开解决方案资源管理器窗格以查看项目。
7. 在解决方案资源管理器窗格，右键单击解决方案 **Contoso.Events** 的节点。
8. 指向添加。
9. 单击新建项目。
10. 在添加新项目对话框，执行以下步骤：
  - a. 展开已安装，**Visual C#**，单击 **Windows 桌面**。
  - b. 单击类库模版。
  - c. 在名称框内，输入 **Contoso.Events.Worker**。
  - d. 单击确定。

#### ► 任务 2：为 Azure SDK 库和解决方案项目添加引用

1. 右键单击 **Contoso.Events.Worker** 项目。
2. 指向添加。
3. 单击引用。
4. 在引用管理器 – **Contoso.Events.Worker** 对话框，执行以下步骤：
  - a. 展开程序集，单击扩展。
  - b. 选择版本为 2.5.0.0 的程序集 **Microsoft.WindowsAzure.ServiceRuntime**。
  - c. 单击确定。
5. 右键单击 **Contoso.Events.Worker** 项目。
6. 指向添加。
7. 单击引用。
8. 在引用管理器 – **Contoso.Events.Worker** 对话框，执行以下步骤：
  - a. 展开解决方案，单击项目。

- b. 选择 **Contoso.Events.Models** 项目。
  - c. 选择 **Contoso.Events.Data** 项目。
  - d. 选择 **Contoso.Events.Documents** 项目。
  - e. 单击**确定**。
9. 在视图菜单，指向**其他窗口**，然后单击**程序包管理器控制台**。
10. 在程序包管理器控制台窗格，在默认项目列表里，选择 **Contoso.Events.Worker**。在程序包管理器控制台的文本区域，把光标移动到文本 PM>之后，输入以下命令：

```
Install-Package EntityFramework -Version 6.0.2
```

11. 按回车键。
12. 在解决方案资源管理器窗格，右键单击 **Contoso.Events.Worker** 项目，然后单击**生成**。

 **注释：**如果弹出文件写保护的消息提示框，单击**覆盖**。

#### ► 任务 3：创建从 **RoleEntryPoint** 继承的类

1. 在解决方案资源管理器窗格内，右键单击 **Contoso.Events.Worker** 项目。
2. 指向**添加**。
3. 单击**新建项**。
4. 在**添加新项 – Contoso.Events.Worker** 对话框，执行以下步骤：
  - a. 展开**已安装**，**Visual C# 项**。
  - b. 单击**类**项。
  - c. 在**名称**框内，输入 **WorkerRole.cs**。
5. 单击**添加**。
6. 在解决方案资源管理器窗格内，展开 **Contoso.Events.Worker** 项目。
7. 双击 **WorkerRole.cs** 项。
8. 定位到以下代码行：

```
class WorkerRole
```

9. 用以下代码代替在步骤 8 中定位到的代码行：

```
public class WorkerRole
```

10. 在类的顶部，添加以下 using 引用：

```
using Microsoft.WindowsAzure.ServiceRuntime;
```

11. 定位到以下代码行：

```
public class WorkerRole
```

12. 用以下代码代替在步骤 11 中定位到的代码行：

```
public class WorkerRole : RoleEntryPoint
```

13. 单击 **WorkerRole** 类内部的任意位置。
14. 使用回车键来换行，稍后会往空出的行中添加方法。
15. 向类中添加以下代码：

```
public override bool OnStart()
{
 ServicePointManager.DefaultConnectionLimit = 12;

 return base.OnStart();
}
```

16. 在类的顶部，添加以下 using 引用：

```
using System.Net;
```

17. 单击 **WorkerRole** 类内部的任意位置。
18. 使用回车键来换行，稍后会往空出的行中添加方法。
19. 向类中添加以下代码：

```
public override void Run()
{
 Trace.WriteLine("Queue Run Start");

 while (true)
 {
 Thread.Sleep(10000);

 Trace.WriteLine("Queue Run Iteration");
 }
}
```

20. 在类的顶部，添加以下 using 引用：

```
using System.Diagnostics;
using System.Threading;
```

21. 保存 WorkerRole.cs 类。

► **任务 4：实现获得 SQL 数据库发来的请求的运行逻辑**

1. 右键单击 **Contoso.Events.Worker** 项目。
2. 指向添加。
3. 单击现有项。
4. 在添加现有项 – Contoso.Events.Worker 对话框内，执行以下步骤：
  - a. 浏览至 **Allfiles (F):\Mod06\Labfiles\Starter**。
  - b. 单击 **WorkerRole.cs**。
  - c. 单击添加。
5. 在目标文件已存在对话框内，单击是以替换现有文件。
6. 在 **Microsoft Visual Studio** 对话框内，单击是以重新加载源编辑器。
7. 在解决方案资源管理器窗格，展开 **Contoso.Events.Worker** 项目。
8. 双击 **App.config** 项。
9. 选中配置文件里面的内容。

10. 按下**删除**键。
11. 输入以下 XML 代码：

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
 <connectionStrings>
 <add name="EventsContextConnectionString" connectionString="Data
Source=(localdb)\v11.0;Initial Catalog=EventsContextModule6Lab;Pooling=True;Integrated
Security=True"
providerName="System.Data.SqlClient" />
 </connectionStrings>
</configuration>
```

**结果：**完成此练习后，您将创建好辅助角色类库，并实现后台工作进程的相应模式。

## 练习 2：将类库添加到云服务项目

► 任务 1：将类库项目作为辅助角色添加到云服务项目

1. 在解决方案资源管理器窗格内，右键单击 **Contoso.Events.Worker** 项目，然后单击卸载项目。
2. 如果你还没有保存项目，你将会被提示保存项目。单击是以保存项目。
3. 在解决方案资源管理器窗格内，右键单击 **Contoso.Events.Worker** 项目，然后单击编辑 **Contoso.Events.Worker.csproj**。
4. 定位到<**PropertyGroup**>XML 标签。
5. 在<**PropertyGroup**>XML 标签之后，<**Configuration**>XML 标签之前，添加以下 XML 标签：

```
<RoleType>Worker</RoleType>
```

6. 在解决方案资源管理器窗格内，右键单击 **Contoso.Events.Worker** 项目，然后单击重新加载项目。
7. 你将被提示关闭.csproj 文件，因为这个文件已经被打开。单击 Yes 关闭.csproj 文件。
8. 如果你还没有保存，你将被提示保存.csproj 文件。单击是保存该文件。
9. 在解决方案资源管理器窗格内，展开 **Models** 文件夹，然后展开 **Contoso.Events.Data** 项目，双击 **EventsContextInitializer.cs** 文件。
10. 定位到以下代码行：

```
Public class EventsContextInitializer : CreateDatabaseIfNotExists<EventsContext>
```

11. 用以下代码代替在步骤 10 中定位到的代码行：

```
Public class EventsContextInitializer : DropCreateDatabaseAlways<EventsContext>
```

12. 在解决方案资源管理器窗格内，展开 **Contoso.Events.Cloud** 项目，然后展开角色文件夹。
13. 右键单击角色文件夹。
14. 指向添加。
15. 单击解决方案中的辅助角色项目。

 **注释：**如果解决方案中的辅助角色项目选项是灰色的，可以通过以下步骤去重新加载 **Contoso.Events.Worker** 项目来解决这个问题：

- a. 在解决方案资源管理器窗格内，右键单击 **Contoso.Events.Worker** 项目，然后单击卸载项目。
- b. 在解决方案资源管理器窗格内，右键单击 **Contoso.Events.Worker** 项目，然后单击重新加载项目。
16. 在与角色项目关联对话框内，执行以下步骤：
  - a. 单击 **Contoso.Events.Worker**。
  - b. 单击确定。

**结果：**完成此练习后，您将获得现有类库项目，并将其转换为现有云服务项目中的辅助角色。

### 练习 3：调试云服务项目中的辅助角色

#### ► 任务 1：调试辅助角色

1. 在解决方案资源管理器窗格内，右键单击 **Contoso.Events.Cloud** 项目，然后单击设为启动项目。
2. 单击调试菜单，单击启动调试。

 **注释：**自始至终，这个课程都将使用 **Azure Compute Emulator** 来测试和调试云服务。在将云服务部署到 Azure 平台前，这个模拟器提供了一个快速的和本地的选项来测试它们。偶尔，这个模拟器会遇到奇怪的问题，例如不正确的端口号、页面显示不了等等。如果出现这种情况，只要关掉 **Compute Emulator**，在下次调试项目时，Visual Studio 将会自行启动这个模拟器。在调试过程中，如果 VS 报告用户代码未处理 **DataException**，请您中断当前调试，并重新启动调试。

3. 在 **Contoso Events** 站点的主页，单击任意一个事件。
4. 单击 **Generate Sign-In Sheet**。
5. 每 30 秒刷新一次页面，直到 **Sign-In Sheet** 生成。

 **注释：**如果看到按钮的文本从 **Generate Sign-In Sheet** 变成 **Download Sign-In Sheet**，就说明 **Sign-In Sheet** 已经生成。

6. 单击 **Download Sign-In Sheet** 按钮下载 Word 文档。
7. 在 **Internet Explorer** 窗口的底部，在下载对话框内，单击 **Open**。
8. 观察在 WordPad 程序内打开的.docx 文件的内容。
9. 关闭 **WordPad** 应用程序。
10. 关闭 **Internet Explorer** 应用程序。
11. 关闭 **Contoso.Events – Microsoft Visual Studio** 应用程序。

**结果：**完成此练习后，您将调试了辅助角色。

# 实验 B：使用 Azure 辅助角色创建后台进程

## 练习 1：创建 C#类库项目

### ► 任务 1：创建 C#类库项目

1. 在 Start 屏幕，右键单击 **Visual Studio 2013** 磁贴。
2. 在弹出的菜单栏，单击 **Run as Administrator**。



**注释：**如果显示了 User Account Control (UAC) 对话框，你可以提升管理员权限来运行 Visual Studio 2013。

3. 在 Visual Studio 起始页界面，单击**打开项目...**。
4. 在**打开项目**对话框，访问 **Allfiles (F):\Mod06\Labfiles\Starter\Contoso.Events**，然后单击 **Contoso.Events.sln**。
5. 单击 **Open**。
6. 展开**解决方案资源管理器**以便查看当前解决方案下的所有项目。
7. 在**解决方案资源管理器**，右键单击解决方案 **Contoso.Events** 节点。
8. 选择**添加选项**。
9. 单击**新建项目...**。
10. 在**添加新项目**对话框中，执行以下操作：
  - a. 展开已安装，**Visual C#**，然后单击 Windows 桌面。
  - b. 单击**类库模板**。
  - c. 在名称输入框，输入 **Contoso.Events.Worker**。
  - d. 单击**确定**。

### ► 任务 2：为解决方案项目和 Azure SDK 类库添加引用

1. 右键单击项目 **Contoso.Events.Worker**。
2. 选择**添加选项**。
3. 单击**引用**。
4. 在**引用管理器 – Contoso.Events.Worker**对话框，执行以下操作：
  - a. 展开**程序集**，然后单击**扩展**。
  - b. 选择版本号是 **2.5.0.0** 的程序集 **Microsoft.WindowsAzure.ServiceRuntime**。
  - c. 单击**确定**。
5. 右键单击项目 **Contoso.Events.Worker**。
6. 选择**添加选项**。
7. 单击**引用**。
8. 在**引用管理器 – Contoso.Events.Worker**对话框，执行以下操作：
  - a. 展开**解决方案**，然后单击**项目**。
  - b. 选择项目 **Contoso.Events.Models**。

- c. 选择项目 **Contoso.Events.Data**。
  - d. 选择项目 **Contoso.Events.Documents**。
  - e. 单击**确定**。
9. 在视图菜单，选择**其他窗口**选项，然后单击**程序包管理器控制台**。
10. 在**程序包管理器控制台**，默认项目列表中选择 **Contoso.Events.Worker**。
- 在**程序包管理器控制台**文本区域，将光标放在 PM>之后，输入如下命令行：
- ```
Install-Package EntityFramework -Version 6.0.2
```
11. 单击回车键。
 12. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击**生成**。
- **任务 3：创建继承基类 RoleEntryPoint 的子类**
1. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**。
 2. 选择**添加选项**。
 3. 单击**新建项**。
 4. 在**添加新项 – Contoso.Events.Worker** 对话框，执行以下操作：
 - a. 展开**已安装**，然后展开**Visual C#项**。
 - b. 单击**代码**项。
 - c. 在名称输入框，输入 **WorkerRole.cs**。
 5. 单击**添加**。
 6. 在解决方案资源管理器，展开项目 **Contoso.Events.Worker**。
 7. 双击文档 **WorkerRole.cs**。
 8. 定位到以下代码：
- ```
class WorkerRole
```
9. 用以下代码替换第 8 步定位的代码：
- ```
public class WorkerRole
```
10. 在该类文档的顶部添加以下 using 块：
- ```
using Microsoft.WindowsAzure.ServiceRuntime;
```
11. 定位到以下代码：
- ```
public class WorkerRole
```
12. 用以下代码替换第 11 步定位的代码：
- ```
public class WorkerRole : RoleEntryPoint
```
13. 单击类 **WorkerRole** 括号内的任何位置。
  14. 在你的方法之间用**回车键**添加垂直空白。
  15. 在你的类中添加以下代码块：

```
public override bool OnStart()
{
 ServicePointManager.DefaultConnectionLimit = 12;

 return base.OnStart();
}
```

16. 在该类文档的顶部添加以下 **using** 块：

```
using System.Net;
```

17. 单击类 **WorkerRole** 括号内的任何位置。

18. 在你的方法之间用**回车键**添加垂直空白。

19. 在你的类中添加以下代码块：

```
public override void Run()
{
 Trace.WriteLine("Queue Run Start");

 while (true)
 {
 Thread.Sleep(10000);

 Trace.WriteLine("Queue Run Iteration");
 }
}
```

20. 在该类文档的顶部添加以下 **using** 块：

```
using System.Diagnostics;
using System.Threading;
```

21. 保存类文档 **WorkerRole.cs**。

► **任务 4：实现从 SQL 数据库获得请求的运行逻辑**

1. 右键单击项目 **Contoso.Events.Worker**。
2. 选择**添加**选项。
3. 单击**现有项**。
4. 在**添加现有项 – Contoso.Events.Worker** 对话框中，执行以下步骤：
  - a. 访问 **Allfiles (F):\Mod06\Labfiles\Starter**。
  - b. 单击 **WorkerRole.cs**。
  - c. 单击**添加**。
5. 在**目标文件已存在**对话框中，单击**是**替换现有文件。
6. 在**Microsoft Visual Studio** 对话框，单击**是**刷新源编辑器。
7. 在**解决方案资源管理器**，展开项目 **Contoso.Events.Worker**。
8. 双击文档 **App.config**。
9. 选中该配置文档的所有内容。
10. 按下**删除**键清除该文档所有内容。
11. 输入以下 XML 节点：

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

MCT ISE ONLY STUDENT USE PROHIBITED

```
<configuration>
 <connectionStrings>
 <add name="EventsContextConnectionString" connectionString="Data
Source=(localdb)\v11.0;Initial Catalog=EventsContextModule6Lab;Pooling=True;Integrated
Security=True"
 providerName="System.Data.SqlClient" />
 </connectionStrings>
</configuration>
```

**结果:** 完成此练习之后，你将创建一个包括辅助角色类的类库项目并为这个后台辅助角色实现了适当的模式。

## 练习 2：添加类库项目到云服务项目中

► 任务 1：在云服务项目中添加一个辅助角色的类库项目

1. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击卸载项目。
2. 如果你还没有保存项目，系统将提示你保存项目。单击是保存项目。
3. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击编辑 **Contoso.Events.Worker.csproj**。
4. 定位到节点打开的**<PropertyGroup>**。
5. 在**<PropertyGroup>** XML 元素和**<Configuration>**XML 元素之间，添加以下 XML 元素：

```
<RoleType>Worker</RoleType>
```

6. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击重新加载项目。
7. 系统将会提示你关闭文档.csproj 因为它已经打开。单击 Yes 关闭文档.csproj。
8. 在提示对话框，单击是保存项目。
9. 在解决方案资源管理器窗格内，展开 **Models** 文件夹，然后展开 **Contoso.Events.Data** 项目，双击 **EventsContextInitializer.cs** 文件。
10. 定位到以下代码行：

```
Public class EventsContextInitializer : CreateDatabaseIfNotExists<EventsContext>
```

11. 用以下代码代替在步骤 10 中定位到的代码行：

```
Public class EventsContextInitializer : DropCreateDatabaseAlways<EventsContext>
```

12. 在解决方案资源管理器，展开项目 **Contoso.Events.Cloud**，然后展开文件夹角色。
13. 右键单击文件夹角色。
14. 选择添加选项。
15. 单击解决方案中的辅助角色项目。

 **注释：**如果选项解决方案中的辅助角色项目...是禁用的，你可以通过重新加载项目 **Contoso.Events.Worker** 来启用它，步骤如下：

- a. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击卸载项目。
- b. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Worker**，然后单击重新加载项目。
16. 在与角色项目关联对话框中，执行以下步骤：
  - a. 单击 **Contoso.Events.Worker**。
  - b. 单击确定。

**结果：**完成本练习后，你会用现有的类库项目，并在现有的云服务项目中把它转换成一个辅助角色。

### 练习 3：在云服务项目中调试辅助角色

#### ► 任务 1：调试辅助角色

1. 在解决方案资源管理器，右键单击项目 **Contoso.Events.Cloud**，然后单击设为启动项目。
2. 在调试菜单，单击启动调试。

 **注释：**整个过程我们将使用 **Azure 计算仿真器** 来测试和调试我们的云服务。在将你的服务部署到 Azure 平台之前，仿真器提供了一种快速和本地的测试方案。有时候，你可能看到模拟器奇怪的错误，比如错误的端口号或无法显示网页。如果这个不断发生，你可以关闭计算仿真器，Visual Studio 将在下一次尝试调试项目时启动仿真器。

在调试过程中，如果 VS 报告用户代码未处理 **DataException**，请您中断当前调试，并重新启动调试。

3. 在 **Contoso Events** 网站的主页，单击任意事件的名称。
4. 单击 **Generate Sign-In Sheet**。
5. 每隔 30 秒刷新页面直到创建登录表。

 **注释：**你如果发现按钮文本从 **Generate Sign-In Sheet** 变成了 **Download Sign-In Sheet**，这表明登陆表已创建完成。

6. 单击 **Download Sign-In Sheet** 下载文档文件。
7. 在 Internet Explorer 窗口底部弹出的下载对话框中，单击 **Open**。
8. 在 WordPad 中，你将看到打开文档.docx 的内容。
9. 关闭 **WordPad** 应用程序。
10. 关闭 **Internet Explorer** 应用程序。
11. 关闭 **Contoso.Events – Microsoft Visual Studio** 应用程序。

**结果：**完成本练习后，你将学会调试辅助角色。

## 第 7 章：在 Azure 中存储表格式数据

# 实验 A：在 Azure 存储表中存储活动注册数据

### 练习 1：用注册者姓名填写注册表单

► 任务 1：创建 **CloudTable** 类的实例

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏中，单击 **File Explorer** 图标。
3. 在 Libraries 窗口，进入到目录 **Allfiles (F):\Mod07\Labfiles\Starter\Tables\Contoso.Events** 下，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中，展开 **Roles** 解决方案文件夹。
5. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Worker** 项目。
6. 双击 **TableStorageHelper.cs** 文件。
7. 在 **TableStorageHelper** 类中，找到以下方法：

```
IEnumerable<string> GetRegistrantNames(string eventKey);
```

8. 在类中删除以下的一行代码：

```
return Enumerable.Empty<string>();
```

9. 在 **GetRegistrantNames** 方法最后及右大括号之前，创建一个 **CloudTable** 实例：

```
CloudTable table = _tableClient.GetTableReference("EventRegistrations");
```

► 任务 2：按分区键检索强类型注册记录

1. 在 **GetRegistrantsNames** 方法最后及右大括号之前，把 **eventKey** 存储在一个叫 **partitionKey** 的字符串变量中：

```
string partitionKey = eventKey;
```

2. 使用 **TableQuery.GenerateFilterCondition** 创建一个 **String** 过滤器，如下显示：

```
string filter = TableQuery.GenerateFilterCondition("PartitionKey",
 QueryComparisons.Equal, partitionKey);
```

3. 在 **GetRegistrantsNames** 方法最后及右大括号之前，创建一个新的 **TableQuery** 类实例 和使用过滤 **Where** 方法来生成一个查询：

```
TableQuery<Registration> query = new TableQuery<Registration>().Where(filter);
```

4. 使用 **Registration** 类作为通用参数，把已生成的查询加进表变量里的 **ExecuteQuery** 方法：

```
IEnumerable<Registration> registrations = table.ExecuteQuery<Registration>(query);
```

► 任务 3：使用 **LINQ-to-Object** 获取注册者姓名列表

1. 在 **GetRegistrantsNames** 方法最后及右括号之前，添加一个声明且不带关闭分号，将注册存储在相同类型名为 **names** 的变量中：

```
IEnumerable<string> names = registrations
```

2. 一个以 **LastName** 来排序结果的方法追加到 lambda-syntax 查询里:

```
.OrderBy(r => r.LastName)
```

3. 在前面代码后面添加一个以 **FirstName** 来排序结果的方法:

```
.ThenBy(r => r.FirstName)
```

4. 在查询代码最后, 添加一个投影方法, 使用 **String.Format** 静态方法来格式化带有 **LastName** 的字符串, 依次是命令, 空格, 然后 **FirstName**:

```
.Select(r => String.Format("{1}, {0}", r.FirstName, r.LastName));
```

5. 在 **GetRegistrantsNames** 方法最后及右括号之前, 加上下面代码:

```
return names;
```

**结果:** 完成此练习后, 您将按行键或分区键从表存储中查询了实体。

## 练习 2：将活动网站更新为使用存储表

► 任务 1：更新注册控制器操作来存储注册记录

1. 在解决方案资源管理器窗格中，展开 **Roles** 解决方案文件夹。
2. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Web** 项目。
3. 在 **Contoso.Events.Web** 项目中，展开 **Controllers** 文件夹。
4. 双击 **RegisterController.cs** 文件。
5. 在 **RegisterController** 类中，找到以下方法：

```
Guid StoreRegistration(dynamic registration)
```

6. 删除类中的下面单行代码：

```
return Guid.Empty;
```

7. 在 **StoreRegistration** 方法最后及右括号之前，通过使用 **CloudConfigurationManager.GetSetting** 静态方法和 **Microsoft.WindowsAzure.Storage.ConnectionString** 值作为参数来得到连接字符串：

```
string connectionString =
CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Storage.ConnectionString");
```

8. 使用带有连接字符串作为参数的 **CloudStorageAccount.Parse** 静态方法来获得存储帐户：

```
var storageAccount =
Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

9. 在 **StoreRegistration** 方法最后及右括号之前，通过使用存储帐户的 **CreateCloudTableClient** 方法创建一个 **CloudTableClient** 变量：

```
var tableClient = storageAccount.CreateCloudTableClient();
```

10. 通过使用 **CloudTableClient** 变量的 **GetTableReference** 方法和“**EventRegistrations**”作为参数，创建一个 **CloudTable** 变量：

```
var table = tableClient.GetTableReference("EventRegistrations");
```

11. 在 **StoreRegistration** 方法最后及右括号之前，通过使用 **TableOperation.Insert** 静态方法和动态注册作为参数来创建一个新的 **TableOperation**：

```
var operation = TableOperation.Insert(registration);
```

12. 通过使用 **CloudTable** 变量，将 **TableOperation** 作为参数调用 **Execute** 方法：

```
table.Execute(operation);
```

13. 在 **StoreRegistration** 方法最后及右括号之前，通过使用 **Guid.Parse** 静态方法解析作为 **System.Guid** 的 **registration.RowKey** 字符串：

```
Guid rowKey = Guid.Parse(registration.RowKey);
```

14. 返回 **rowKey** 变量作为 **StoreRegistration** 方法的结果。

```
return rowKey;
```

► 任务 2：更新注册 **ViewModel** 来从表中检索动态存根注册

1. 在解决方案资源管理器窗格中，展开 **Shared** 解决方案文件夹。
2. 在解决方案资源管理器窗格中，展开 **Contoso.Events.ViewModels** 项目。
3. 双击 **RegisterViewModel.cs** 文件。
4. 在 **RegisterViewModel** 类中，定位到以下代码：

```
RegisterViewModel(string eventKey)
```

5. 在 **RegisterViewModel** 构造函数的最后及右括号之前，获得使用 **CloudConfigurationManager.GetSetting** 静态方法和 **Microsoft.WindowsAzure.Storage.ConnectionString** 值作为参数的连接字符串：

```
string connectionString =
CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Storage.ConnectionString");
```

6. 使用带有连接字符串作为参数的 **CloudStorageAccount.Parse** 静态方法来获得存储帐户：

```
var storageAccount =
Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

7. 在 **RegisterViewModel** 构造函数的最后及右括号之前，通过使用存储帐户的 **CreateCloudTableClient** 方法创建 **CloudTableClient** 变量：

```
var tableClient = storageAccount.CreateCloudTableClient();
```

8. 通过使用 **CloudTableClient** 变量的带有 **EventRegistrations** 参数的 **GetTableReference** 方法创建一个 **CloudTable** 变量：

```
var table = tableClient.GetTableReference("EventRegistrations");
```

9. 在 **RegisterViewModel** 构造函数的最后及右括号之前，在一个命名为 **partitionKey** 的字符串变量里存储 **eventKey**：

```
string partitionKey = String.Format("Stub_{0}", this.Event.EventKey);
```

10. 使用 **TableQuery.GenerateFilterCondition** 创建一个 **String** 过滤器：

```
string filter = TableQuery.GenerateFilterCondition("PartitionKey",
QueryComparisons.Equal, partitionKey);
```

11. 在 **RegisterViewModel** 构造函数的最后及右括号之前，通过使用带有过滤器的 **Where** 方法创建一个新的 **TableQuery** 类的实例和生成一个查询：

```
TableQuery query = new TableQuery().Where(filter);
```

12. 把生成的查询加进那个表变量的 **ExecuteQuery** 方法里：

```
IEnumerable<DynamicTableEntity> tableEntities = table.ExecuteQuery(query);
```

13. 在 **RegisterViewModel** 构造函数的最后及右括号之前，在 **DynamicTableEntity** 对象中选择单个元素：

```
DynamicTableEntity tableEntity = tableEntities.SingleOrDefault();
```

14. **RegistrationStub** 属性设置为 **DyanmicTableEntity** 变量：

```
this.RegistrationStub = DynamicEntity.GenerateDynamicItem(tableEntity);
```

结果：完成此练习后，您将使用 Azure Storage SDK 检索并创建了实体。

MCT USE ONLY. STUDENT USE PROHIBITED

### 练习 3：验证活动网站正在使用 Azure 存储表来存储注册数据

#### ► 任务 1：运行数据生成控制台项目来用数据填充 Azure 存储表

1. 在解决方案资源管理器窗格中，展开 **Shared** 解决方案文件夹。
2. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Data.Generation** 项目。
3. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Data.Generation** 项目，指向调试，然后单击启动新实例。

#### ► 任务 2：使用 Visual Studio 2013 中的服务器资源管理器来查看表存储注册数据

1. 在视图菜单中，单击服务器资源管理器。
2. 定位到 **Azure** 节点并单击左边的箭头。
3. 展开在 **Azure** 节点下的**存储**节点。
4. 如果系统提示你输入帐户凭据，使用 **Microsoft** 帐户登入。
5. 展开**存储**节点下的**开发**帐户节点。
6. 展开**开发**节点下的**表**节点。
7. 双击 **EventRegistrations** 表。
8. 在 **EventRegistrations [表]**选项卡，滚动查看所有实体。
9. 双击一行中的某一单个实体以查看实体属性。
10. 单击取消按钮，退出**编辑实体**对话框。
11. 关闭在 Visual Studio 中的 **EventRegistrations [表]**选项卡。

#### ► 任务 3：调试云 Web 项目来注册活动

1. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Cloud** 项目，然后单击设为启动项目。
2. 在调试菜单，单击启动调试。
3. 如果 **Storage Emulator** 正在运行，会提示你关闭模拟器并通过使用不同的模式重新启动它，然后单击 **OK**。
4. 在 Web 应用程序的主页，确认显示一些列活动。
5. 单击列表里的任一活动。
6. 在打开的活动页面，单击 **Register Now**。
7. 填写在注册表中的所有字段并单击 **Submit**。
8. 关闭显示网站的选项卡。

#### ► 任务 4：使用 Visual Studio 2013 中的服务器资源管理器来查看新的表存储注册数据

1. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
2. 在视图菜单，单击服务器资源管理器。
3. 定位到 **Azure** 节点。
4. 双击在 **Azure** 节点里的 **EventRegistrations**。
5. 在 **EventRegistrations [表]**选项卡，滚动查看所有实体。
6. 在黄色的提示框中询问你是否想下载剩余实体，单击单击此处。
7. 双击一行中的某一单个实体以查看实体属性，然后单击取消。

8. 单击 **Timestamp** 标题两次以通过 Timestamp 以降序来排序实体。
9. 在表的顶部，找到你新建的实体。
10. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
11. 关闭 **Contoso.Events – Microsoft Visual Studio**。

**结果：**完成此练习后，您将使用 Visual Studio 和 Azure 模拟器创建了适用于 Azure 存储的综合开发环境。

# 实验 B：在 Azure 存储表中存储活动注册数据

## 练习 1：用注册者姓名填写注册表单

### ► 任务 1：创建 CloudTable 类的实例

1. 在 Start 屏幕单击 **Desktop** 磁贴。
2. 在任务栏上单击 **File Explorer** 图标。
3. 在 This PC 窗口，打开 **Allfiles (F):\Mod07\Labfiles\Starter\Tables\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格上，展开 **Roles** 目录。
5. 在解决方案资源管理器窗格上，展开 **Contoso.Events.Worker** 项目。
6. 双击 **TableStorageHelper.cs** 文件。
7. 在 **TableStorageHelper** 类，找到如下签名的方法：

```
public IEnumerable<string> GetRegistrantNames(string eventKey)
```

8. 删除代码类中的下面这一行：

```
return Enumerable.Empty<string>();
```

9. 在 **GetRegistrantNames** 方法的结尾和右括号之前，创建一个 **CloudTable** 实例：

```
CloudTable table = _tableClient.GetTableReference("EventRegistrations");
```

### ► 任务 2：按分区键检索强类型注册记录

1. 在 **GetRegistrantsNames** 方法的结尾和右括号之前，将 **eventKey** 存储在一个命名为 **partitionKey** 的 **string** 变量中：

```
string partitionKey = eventKey;
```

2. 使用 **TableQuery.GenerateFilterCondition** 创建一个 **string** 过滤器：

```
string filter = TableQuery.GenerateFilterCondition("PartitionKey",
 QueryComparisons.Equal, partitionKey);
```

3. 在 **GetRegistrantsNames** 的方法结尾和右括号之前创建一个 **TableQuery** 类实例，使用 **Where** 方法过滤来生成一个查询：

```
TableQuery<Registration> query = new TableQuery<Registration>().Where(filter);
```

4. 将生成的查询作为参数传递给 **table** 变量的 **ExecuteQuery** 方法，使用 **Query** 实体作为泛型参数：

```
IEnumerable<Registration> registrations = table.ExecuteQuery<Registration>(query);
```

### ► 任务 3：使用 LINQ-to-Objects 获取注册者姓名列表

1. 在 **GetRegistrantsNames** 方法的结尾和右括号之前添加一行没有分号结尾的语句，将注册信息保存到变量 **names**

```
IEnumerable<string> names = registrations
```

2. 向上一行中追加 lambda 表达式查询语法，使用 **LastName** 属性来对结果排序：

```
.OrderBy(r => r.LastName)
```

3. 继续追加 lambda 表达式查询语法，使用 **FirstName** 属性来对结果进行二次排序：

```
.ThenBy(r => r.FirstName)
```

4. 使用选择操作来结束 lambda 表达式查询，通过静态方法 **String.Format** 来格式化字符串让 **FirstName** 跟在 **LastName** 之后，并以逗号分隔：

```
.Select(r => String.Format("{1}, {0}", r.FirstName, r.LastName));
```

5. 在 **GetRegistrantsNames** 方法的结尾和右括号之前，添加以下语句：

```
return names;
```

**结果：**在完成本次练习以后，您将了解如何按行键或分区键从表存储中查询实体。

## 练习 2：将活动网站更新为使用存储表

### ► 任务 1：更新注册控制器来存储注册记录

1. 在解决方案资源管理器窗格，展开 **Roles** 目录。
2. 在解决方案资源管理器窗格，展开 **Contoso.Events.Web** 项目。
3. 在 **Contoso.Events.Web** 项目，展开 **Controllers** 文件夹。
4. 双击打开 **RegisterController.cs** 文件。
5. 在 **RegisterController** 类中，找到下面签名的方法：

```
private Guid StoreRegistration(dynamic registration)
```

6. 删掉下面这一行：

```
return Guid.Empty;
```

7. 在 **StoreRegistration** 方法结尾和右括号之前使用 **CloudConfigurationManager.GetSetting** 静态方法和 **Microsoft.WindowsAzure.Storage.ConnectionString** 参数来获取连接字符串。

```
string connectionString =
CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Storage.ConnectionString");
```

8. 使用静态方法 **CloudStorageAccount.Parse** 和连接字符串作为参数获取存储帐号：

```
var storageAccount =
Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

9. 在 **StoreRegistration** 方法的结尾和右括号之前使用存储帐号的 **CreateCloudTableClient** 方法创建一个 **CloudTableClient** 变量：

```
var tableClient = storageAccount.CreateCloudTableClient();
```

10. 通过使用 **CloudTableClient** 变量的 **GetTableReference** 方法作为参数，创建 **CloudTable** 变量：

```
var table = tableClient.GetTableReference("EventRegistrations");
```

11. 在 **StoreRegistration** 方法的结尾和右括号之前，通过使用 **TableOperation.Insert** 静态方法和 **registration** 作为参数创建一个新的 **TableOperation**：

```
var operation = TableOperation.Insert(registration);
```

12. 使用 **CloudTable** 变量传递 **TableOperation** 参数来执行 **Execute** 方法：

```
table.Execute(operation);
```

13. 在 **StoreRegistration** 方法结尾和右括号之前，通过调用 **Guid.Parse** 静态方法，使用 **registration.RowKey** 字符串作为参数来生成 **Guid**：

```
Guid rowKey = Guid.Parse(registration.RowKey);
```

14. 将 **rowKey** 作为 **StoreRegistration** 方法的结果返回。

```
return rowKey;
```

### ► 任务 2：通过更新注册的 **ViewModel** 来从表中检索动态存根

1. 在解决方案资源管理器窗格上，展开 **Shared** 目录。

2. 在解决方案资源管理器窗格上，展开项目 **Contoso.Events.ViewModels**。

3. 双击打开 **RegisterViewModel.cs** 文件。

4. 在 **RegisterViewModel** 类中，定位到如下签名的方法中：

```
public RegisterViewModel(string eventKey)
```

5. 在 **RegisterViewModel** 构造函数结尾和右括号之前，使用 **CloudConfigurationManager.GetSetting** 静态方法和 **Microsoft.WindowsAzure.Storage.ConnectionString** 值作为参数获取连接字符串：

```
string connectionString =
CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Storage.ConnectionString");
```

6. 使用 **CloudStorageAccount.Parse** 静态方法，使用连接字符串作为参数来获得存储帐户：

```
var storageAccount =
Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

7. 在 **RegisterViewModel** 构造函数的结尾和右括号之前，通过使用存储帐户的 **CreateCloudTableClient** 方法创建一个 **CloudTableClient** 变量：

```
var tableClient = storageAccount.CreateCloudTableClient();
```

8. 通过使用 **CloudTableClient** 变量的 **GetTableReference** 方法和“**EventRegistrations**”作为参数创建 **CloudTable** 变量：

```
var table = tableClient.GetTableReference("EventRegistrations");
```

9. 在 **RegisterViewModel** 构造函数的结尾和右括号之前将 **eventKey** 存储到命名为 **partitionKey** 的 **string** 变量中。

```
string partitionKey = String.Format("Stub_{0}", this.Event.EventKey);
```

10. 使用 **TableQuery.GenerateFilterCondition** 创建一个 **String** 过滤器：

```
string filter = TableQuery.GenerateFilterCondition("PartitionKey",
QueryComparisons.Equal, partitionKey);
```

11. 在 **RegisterViewModel** 构造函数的结尾和右括号之前，创建 **TableQuery** 类的新实例，并使用 **Where** 方法联合 String 过滤器生成一个查询：

```
TableQuery query = new TableQuery().Where(filter);
```

12. 将生成的查询作为参数传递进 **table** 变量的 **ExecuteQuery** 方法中：

```
IEnumerable<DynamicTableEntity> tableEntities = table.ExecuteQuery(query);
```

13. 在 **RegisterViewModel** 构造函数结尾和右括号之前，在 **DynamicTableEntity** 集合中选择单个 **DynamicTableEntity** 对象元素：

```
DynamicTableEntity tableEntity = tableEntities.SingleOrDefault();
```

14. 通过 **DynamicEntity.GenerateDynamicItem** 方法联合新创建的 **DynamicTableEntity** 对象来设置 **RegistrationStub** 属性的值：

```
this.RegistrationStub = DynamicEntity.GenerateDynamicItem(tableEntity);
```

**结果:** 完成此练习后，您将使用 Azure Storage SDK 检索并创建了实体。

### 练习 3：验证活动网站使用 Azure 存储表来存储注册数据

#### ► 任务 1：运行生成数据的控制台项目来向 Azure 存储表中填充数据

1. 在解决方案资源管理器窗格中，展开 **Shared** 解决方案目录。
2. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Data.Generation** 项目。
3. 在解决方案资源管理器窗格中，右击 **Contoso.Events.Data.Generation** 项目，选择 **调试**，然后单击 **启用新实例**。

 **注释：**如果弹出**保存只读文件对话框**，请您单击**覆盖**。

#### ► 任务 2：使用 Visual Studio 2013 中的服务器资源管理器来查看存储表的注册数据

1. 在视图菜单，单击**服务器资源管理器**。
2. 定位到 **Azure** 节点，单击左侧的箭头展开 Azure 节点。
3. 展开**存储**节点。
4. 如果需要帐号凭据，输入你的 **Microsoft Account**。
5. 展开**存储**节点下的**开发**帐号节点。
6. 展开**开发**帐号节点下的**表**节点。
7. 双击打开 **EventRegistrations** 表。
8. 在 **EventRegistrations [表]** 选项卡上，移动滚动条查看实体。
9. 通过双击某一行来查看单个实体的所有属性。
10. 单击取消按钮来退出**编辑实体**对话框。
11. 在 Visual Studio 中关闭 **EventRegistrations [表]** 选项卡。

#### ► 任务 3：调试云 Web 项目来注册活动

1. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Cloud** 项目，然后单击**设为启动项目**。
2. 在调试菜单，单击**启动调试**。
3. 如果**存储模拟器**已经运行，会提示您关闭并使用不同的模式重启。单击**确定**即可。
4. 在 Web 应用程序的首页，验证显示的活动列表。
5. 在列表中单击任意的活动。
6. 在活动的 Web 页面，单击 **Register Now**。
7. 在所有的文本框填充相应的信息，然后单击 **Submit**。
8. 关闭打开网站的选项卡。

#### ► 任务 4：使用 Visual Studio 2013 中的服务器资源管理器来查看存储表中的新的注册数据

1. 切换至 **Contoso.Events – Microsoft Visual Studio** 窗口。
2. 在视图菜单，单击**服务器资源管理器**。
3. 定位到 **Azure** 节点。
4. 在 **Azure->存储->（开发）->表** 节点下，双击 **EventRegistrations**。
5. 在 **EventRegistrations [表]** 选项卡，拖动滚动条查看实体。

6. 在 **EventRegistrations [表]** 选项卡顶部黄色提示框，单击[单击此处](#)链接。
7. 通过双击某一行来查看单个实体的其它属性，然后单击[取消](#)。
8. 单击 **Timestamp** 标题两次，实体将会根据 Timestamp 降序排序。
9. 在表的顶部，找到新添加的实体。
10. 切换至 **Contoso.Events – Microsoft Visual Studio** 窗口。
11. 关闭 **Contoso.Events – Microsoft Visual Studio**。

**结果：**完成本练习后，你会使用 Visual Studio 和 Azure 的模拟器来创建 Azure 存储的综合开发环境。

## 第 8 章：存储和使用 Azure 存储中的文件

# 实验 A：在 Azure 存储 Blob 中存储生成的文档

### 练习 1：实现 Azure 存储 Blob

#### ► 任务 1：登录到 Azure 预览门户

1. 在开始屏幕，单击 **Internet Explorer**。
2. 转到 <https://portal.azure.com> 页面。
3. 输入 Microsoft 帐户的 Email 地址。
4. 单击 **Continue**。
5. 输入 Microsoft 帐户的密码。
6. 单击 **Sign In**。
7. 单击右上角的帐户名。
8. 单击 **Settings**。
9. 单击 **LANGUAGE**，选择中文（简体），然后单击 **Select**。

#### ► 任务 2：使用管理门户创建容器

1. 在屏幕左侧的导航窗格里，单击 **浏览**。
2. 在已打开的 **浏览** 边栏选项卡，单击 **Storage Accounts**。
3. 在打开的 **Storage Accounts** 边栏选项卡，查看已创建的 Storage 帐户。
4. 在创建的 Storage 帐户列表中，定位到带有前缀名为 **stor20532** 的 storage 帐户。
5. 单击 **stor20532[自定义名称]**，进入到启动板。
6. 在打开的 **stor20532[自定义名称]** 边栏选项卡，单击 **容器** 磁贴。
7. 在打开的 **容器** 边栏选项卡，查看容器列表。
8. 在容器屏幕的顶端，单击 **添加** 按钮。
9. 在打开的 **添加容器** 边栏选项卡，执行以下步骤：
  - a 在名称框中，输入 **example**。
  - b 在访问类型列表里，选择 **容器**。
  - c 单击 **确认** 按钮。

#### ► 任务 3：添加和访问容器中的 Blob 文件

1. 在开始屏幕，单击 **Visual Studio 2013**。
2. 在视图菜单里，单击 **服务器资源管理器**。
3. 定位到 **Azure** 节点，然后单击节点左边的箭头。
4. 展开 **Azure** 节点下方的 **存储** 节点。



**注释:** 如果你以前没有指定你的 VisualStudio 记住任何凭证, 系统将会提示你输入 Microsoft 帐户名和密码以继续。

5. 展开**存储**节点里的**stor20532[自定义名称]**节点。
6. 展开**Blob** 节点。
7. 双击**example**。
8. 在**example[容器]**选项卡, 单击上载**Blob** 按钮。



**注释:** 上载按钮上的图标包括一个箭头朝上指向一条水平线。

9. 在**上载新文件**对话框, 执行以下步骤:
  - a. 单击**浏览**按钮。
  - b. 转到(F):\Mod08\LabFiles\Starter\文件夹。
  - c. 单击**samplefile** 文本文件。
  - d. 单击**Open**。
  - e. 单击**确定**。
10. 切换到 InternetExplorer 窗口。
11. 在新的页面, 输入下面的网址, 并用你的 storageaccount 名字替换掉**[storageaccount]**:  
[http://\[storageaccount\].blob.core.windows.net/example/samplefile.txt](http://[storageaccount].blob.core.windows.net/example/samplefile.txt)
12. 验证页面显示的文本内容为 **This is your sample file!**。

**结果:** 完成此练习后, 您将使用管理门户创建了 Blob 容器, 并查看了容器中的 Blob。

## 练习 2：以文件和媒体填充容器

► 任务 1：在云服务辅助项目中打开 **Blob** 帮助器

1. 在开始屏幕，单击 **Desktop**。
2. 在任务栏中，单击 **File Explorer**。
3. 转到 **Allfiles (F):\Mod08\Labfiles\Starter\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中，展开 **Roles** 解决方案文件夹。
5. 在解决方案资源管理器窗格中，展开项目 **Contoso.Events.Worker**。
6. 双击 **BlobStorageHelper.cs** 文件。

► 任务 2：在创建的容器中上传 **Word** 文档

1. 在 **BlobStorageHelper** 类中，找到下面方法：

```
public Uri CreateBlob(MemoryStream stream, string eventKey)
```

2. 删掉下面单行代码：

```
return null;
```

3. 在 **CreateBlob** 方法的最后以及右括号之前面，为 **signin** 容器创建一个 **CloudBlobContainer**。

```
CloudBlobContainer container = _blobClient.GetContainerReference("signin");
```

4. 调用 **CloudBlobContainer** 变量里 **CreateIfNotExists** 方法，以确保容器存在：

```
container.CreateIfNotExists();
```

5. 在 **CreateBlob** 方法的最后以及右括号前面，创建一个命名为 **blobName** 的变量：

```
string blobName;
```

6. 使用静态方法 **String.Format** 来创建一个字符串，然后把字符串分配给 **blobName** 变量：

```
blobName = String.Format("{0}_SignIn_{1:ddmmyyss}.docx", eventKey, DateTime.UtcNow);
```

7. 在 **CreateBlob** 方法的最后以及右括号前面，通过使用 **GetBlockBlobReference** 方法和变量 **blobName** 作为参数，创建一个 **blob** 块引用：

```
ICloudBlob blob = container.GetBlockBlobReference(blobName);
```

8. 使用 **MemoryStream** 对象的 **Seek** 方法，设置流的位置为开始和偏移位置的值为 **0**：

```
stream.Seek(0, SeekOrigin.Begin);
```

9. 使用 **ICloudBlob** 接口的 **UploadFromStream** 方法，上载流到 **blob** 引用：

```
blob.UploadFromStream(stream);
```

10. 在 **CreateBlob** 方法的最后以及右括号前面，添加下面代码：

```
return blob.Uri;
```

**结果:** 完成此练习后，您将能使用 Azure 存储 SDK 来管理存储帐户下的 Blob 和容器。

MCT USE ONLY. STUDENT USE PROHIBITED

### 练习 3：从容器中检索文件和媒体

► 任务 1：从 Blob 存储下载文档并流式传送到客户端

1. 在解决方案资源管理器窗格中，展开 **Shared** 解决方案文件夹。
2. 在解决方案资源管理器窗格中，展开 **Contoso.Events.ViewModels** 项目。
3. 双击 **DownloadViewModel.cs** 文件。
4. 在 **DownloadViewModel** 类中，找到下面方法：

```
public async Task<DownloadPayload> GetStream()
```

5. 删掉方法中的下面单行代码：

```
return await Task.FromResult<DownloadPayload>(null);
```

6. 在 **GetStream** 方法的最后以及右括号前，为 **\_storageAccount** 变量创建一个新的 **CloudBlobClient** 变量：

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
```

7. 使用 **CloudBlobClient** 变量为 **signin** 容器创建一个新的 **CloudBlobContainer** 实例：

```
CloudBlobContainer container = blobClient.GetContainerReference("signin");
```

8. 调用 **CloudBlobContainer** 变量中的 **CreateIfNotExists** 方法，以确保容器存在：

```
container.CreateIfNotExists();
```

9. 在 **GetStream** 方法的最后以及右括号前，使用 **GetBlockBlobReference** 方法和 **\_blobId** 变量作为参数创建一个块 blob 引用：

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
```

10. 使用 **ICloudBlob** 类中的 **OpenReadAsync** 方法来创建一个 **Stream** 并储存到一个变量：

```
Stream blobStream = await blob.OpenReadAsync();
```

11. 在 **GetStream** 方法的最后以及右括号前，创建一个新的 **DownloadPayload** 类实例：

```
DownloadPayload payload = new DownloadPayload();
```

12. 把 **Stream** 变量的值赋值给 **DownloadPayload** 变量的 **Stream** 属性：

```
payload.Stream = blobStream;
```

13. 把 **ICloudBlob** 变量的 **Properties.ContentType** 值赋值给 **DownloadPayload** 变量的 **ContentType** 属性：

```
payload.ContentType = blob.Properties.ContentType;
```

14. 返回 **DownloadPayload** 变量的值：

```
return payload;
```

► 任务 2：生成测试数据

1. 在开始屏幕，输入 **Azure Storage Emulator**。

2. 单击 **WindowsAzureStorageEmulator-v3.4**。
3. 切换到 **Contoso.Events—MicrosoftVisualStudio** 窗口。
4. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Data.Generation** 项目，指向调试，然后单击启动新实例。

 **注释：**如果弹出写文件保护的消息框，单击覆盖。

► **任务 3：从 Blob 存储下载生成的签到表**

1. 在解决方案资源管理器窗格中，右键单击解决方案 ‘**Contoso.Events**’，然后单击属性。
2. 在解决方案 ‘**Contoso.Events**’ 属性页对话框里，执行以下步骤：
  - a. 在页面左边的导航菜单中，确保已默认选择通用属性=>启动项目。
  - b. 选择多启动项目。
  - c. **Contoso.Events.Cloud**，把操作设置成启动。
  - d. **Contoso.Events.Management**，把操作设置成开始执行（不调试）。
  - e. 确保其他项目操作设置成无。
  - f. 单击 **OK** 按钮。
3. 单击调试菜单，单击启动调试。
4. 在 **notification** 域窗口，单击箭头以查看正在运行的应用。
5. 定位到 **IISExpress**，然后右键单击该图标。
6. 单击在 **ViewSites** 下方的 **Contoso.Events.Management**，然后单击在 **BrowseApplications** 下面的 URL。
7. 同样单击在 **ViewSites** 下方的 **Contoso.Events.Cloud**，然后单击在 **BrowseApplications** 下面的 URL。
8. 在 **Events Administration** 页面，单击 **Go to EventsList** 按钮以查看事件列表。
9. 单击列表中任一事件的 **Sign-InSheet** 按钮。
10. 查看 sign-in 页面并注意到一个 sign-in 表格正在生成。
11. 等待一至两分钟，然后刷新 sign-in 表格页面。
12. 单击 **Sign-InSheet** 按钮来下载 sign-in 表格。

**结果：**完成此练习后，您将使用 Azure 存储 SDK 从存储帐户下下载了 Blob。

## 练习 4：指定容器的权限

► 任务 1：使用服务器资源管理器修改容器访问权限

- 在桌面上，单击 **Contoso.Events-VisualStudio2013** 窗口。
- 在调试菜单中，单击停止调试。
- 在视图菜单中，单击服务器资源管理器。
- 定位到 **Azure** 节点，然后单击节点左边的箭头。
- 展开 **Azure** 节点下面的**存储**节点。

 **注释：**如果弹出 Microsoft account – Sign in 对话框，输入您的 Microsoft 帐户和密码，单击 Sign in。

- 展开**存储**节点下面的**开发**节点。
- 展开**开发**节点下面的 **Blob** 节点。
- 右键单击 **signin** 容器，然后单击**属性**。
- 在**属性**窗格中，定位到**公共读访问**。
- 确保**公共读访问**属性的值设置为 **Off**。

► 任务 2：使用 SDK 生成临时 SAS 令牌

- 在**解决方案资源管理器**窗格中，展开 **Shared** 解决方案文件夹。
- 在**解决方案资源管理器**窗格中，展开 **Contoso.Events.ViewModels** 项目。
- 双击 **DownloadViewModel.cs** 文件。
- 在 **DownloadViewModel** 类中，找到以下方法：

```
public async Task<string> GetSecureUrl()
```

- 删掉类中的下面单行代码：

```
return await Task.FromResult<string>(String.Empty);
```

- 在 **GetSecureUrl** 方法的最后以及右括号前，为 **\_storageAccount** 变量创建一个新的 **CloudBlobClient**。

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
```

- 使用 **CloudBlobClient** 变量为 **signin** 容器创建一个新的 **CloudBlobContainer**。

```
CloudBlobContainer container = blobClient.GetContainerReference("signin");
```

- 调用 **CloudBlobContainer** 类里的 **CreateIfNotExists** 方法以确保容器存在：

```
container.CreateIfNotExists();
```

- 在 **GetSecureUrl** 方法的最后以及右括号前，创建一个新的 **SharedAccessBlobPolicy** 类实例：

```
SharedAccessBlobPolicy blobPolicy = new SharedAccessBlobPolicy();
```

- 把 **SharedAccessBlobPolicy** 变量的 **SharedAccessExpiryTime** 属性设置到当前时间后 15 分钟：

MCT USE ONLY. STUDENT USE PROHIBITED

```
blobPolicy.SharedAccessExpiryTime = DateTime.Now.AddHours(0.25d);
```

11. 把 SharedAccessBlobPolicy 变量 **Permissions** 属性设置为 **SharedAccessBlobPermissions.Read**:

```
blobPolicy.Permissions = SharedAccessBlobPermissions.Read;
```

12. 在 **GetSecureUrl** 方法的最后以及右括号前，创建一个新的 **BlobContainerPermissions** 类实例:

```
BlobContainerPermissions blobPermissions = new BlobContainerPermissions();
```

13. 用“**ReadBlobPolicy**”值把新建的 **SharedAccessBlobPolicy** 添加到 BlobContainerPermissions 变量的 **SharedAccessPolicy** 里:

```
blobPermissions.SharedAccessPolicies.Add("ReadBlobPolicy", blobPolicy);
```

14. 把 BlobContainerPermissions 变量的 **PublicAccess** 属性设置为 **BlobContainerPublicAccessType.Off** :

```
blobPermissions.PublicAccess = BlobContainerPublicAccessType.Off;
```

15. 在 **GetSecureUrl** 方法的最后以及右括号前，通过使用 BlobContainerPermissions 变量作为参数调用 CloudBlobContainer 类中的 **SetPermissionsAsync** 方法:

```
await container.SetPermissionsAsync(blobPermissions);
```

16. 使用 **SharedAccessBlobPolicy** 类实例作为第一参数和“**ReadBlobPolicy**”值作为第二参数，调用 CloudBlobContainer 类里的 **GetSharedAccessSignature** 方法:

```
string sasToken = container.GetSharedAccessSignature(new SharedAccessBlobPolicy(), "ReadBlobPolicy");
```

17. 在 **GetSecureUrl** 方法的最后以及右括号前，使用 **GetBlockBlobReference** 方法和 **\_blobId** 变量作为参数创建一个块 blob 引用:

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
```

18. 使用 ICloudBlob 变量中的 **Uri** 属性，并把它储存在一个新的 Uri 变量。

```
Uri blobUrl = blob.Uri;
```

19. 在 **GetSecureUrl** 方法的最后以及右括号前，串联 Uri 变量和 sasToken 变量的 **AbsoluteUri**:

```
string secureUrl = blobUrl.AbsoluteUri + sasToken;
```

20. 返回结果的值:

```
return secureUrl;
```

### ► 任务 3：使用 SAS 令牌从受保护的容器下载文档

- 在调试菜单里，单击启动调试。
- 在 notification 域，单击箭头以查看正在运行的应用。
- 定位到 **IISExpress**，然后右键单击该图标。
- 单击在 **ViewSites** 下方的 **Contoso.Events.Management**，然后单击在 **BrowseApplications** 下面的 URL。

5. 在 **EventsAdministration** 页面，单击 **GotoEventsList** 按钮以查看事件列表。
6. 单击列表中任一事件的 **Sign-InSheet** 按钮。
7. 查看 sign-in 页面并注意到一个 sign-in 表格正在生成。
8. 等待一至两分钟，然后刷新 sign-in 表格页面。
9. 单击 **Sign-InSheet** 按钮来下载 sign-in 表格。
10. 关闭 **InternetExplorer** 应用程序。
11. 关闭 **Contoso.Events-VisualStudio** 应用程序。
12. 关闭 **AzureStorageEmulator** 命令提示窗口。

**结果：**完成此练习后，您将修改了容器的权限，并生成了容器的 SAS 令牌。

# 实验 B：在中国版 Windows Azure 的存储 Blob 中存储生成的文档

## 练习 1：使用 Blob 实现 Azure 存储

### ► 任务 1：登录到 Azure 管理门户

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 打开 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的屏幕语言不是中文，可以单击页面右上角的地球图表，并在菜单中选择**中文(简体)**语言。

### ► 任务 2：使用管理门户创建 Blob 容器

1. 通过 Task1 中的练习，我们已经成功地登录到 Azure 的管理门户。
2. 在左侧的导航栏中，拖动滚动条找到**存储**图标，单击**存储**查看所有的存储实例列表。
3. 在存储实例列表中，定位到含有**stor28532**前缀的存储帐户。
4. 单击存储帐户的名称，进入含有**stor28532**前缀的存储帐户页面。
5. 在页面顶部，单击**容器**选项卡。
6. 在**容器**选项卡上查看所有的容器列表。
7. 在页面底部的工具栏中，单击**添加**按钮打开**新建容器**对话框。
8. 在**新建容器**窗口中，执行以下步骤：
  - a 在**名称**文本框中，输入容器的名称**example**。
  - b 在**访问类型**列表中，选择**公共容器**。
  - c 单击对勾**确定**按钮。

### ► 任务 3：添加和访问容器中的 Blob 文件

1. 在 Start 屏幕，单击 **Visual Studio 2013** 磁贴。
2. 在视图菜单，单击**服务器资源管理器**。
3. 定位到**Azure** 节点，单击左侧箭头展开节点。
4. 在**Azure** 节点下面展开**存储**节点。

 **注释：**如果你之前没有导入 Azure 订阅文件，你需要先导入订阅文件。关于如何在 Visual Studio 中导入 Azure 订阅，请参考第二章实验 B，练习 4，任务 1 和任务 2。

5. 在**存储**节点展开**stor28532[自定义名称]**节点。
6. 展开**Blob** 节点。
7. 双击**example**。
8. 在**example[容器]**选项卡，单击上载**Blob**。

 **注释：**上载图标是一个指向一条横线加向上箭头。

9. 在上载新文件对话框中，执行以下的步骤：
  - a 单击浏览按钮
  - b 定位到(F):\Mod08\LabFiles\Starter
  - c 单击 **samplefile** 文本文档
  - d 单击 **Open**
  - e 单击确定
10. 切换至 **Internet Explorer** 窗口。
11. 在新选项卡中输入下面的连接，将连接中的字符串**[存储帐户]**替换为你的存储帐户名称：  
[http://\[存储帐户\].blob.core.chinacloudapi.cn/example/samplefile.txt](http://[存储帐户].blob.core.chinacloudapi.cn/example/samplefile.txt)
12. 验证在浏览器中显示的内容为： **This is your sample file!**。

**结果：**完成本次练习以后，你会在 Azure 的管理门户上创建的 blob 容器上，并查看 blob 文件。

## 练习 2：向容器中添加文件和媒体文件

### ► 任务 1：在云服务解决方案的 Worker 项目中打开 BlobStorageHelper 文件

1. 在 Start 屏幕，单击 Desktop 磁贴。
2. 在任务栏中，单击 File Explorer。
3. 转到 Allfiles (F):\Mod08\Labfiles\Starter\Contoso.Events，然后双击 Contoso.Events.sln。
4. 在解决方案资源管理器窗格，展开 Roles 目录。
5. 在解决方案资源管理器窗格，展开 Contoso.Events.Worker 项目。
6. 双击打开 BlobStorageHelper.cs 文件。

### ► 任务 2：将新创建的 Word 文档上传到容器中

1. 在 BlobStorageHelper 类中，找到如下签名的方法：

```
public Uri CreateBlob(MemoryStream stream, string eventKey)
```

2. 在 CreateBlob 方法中，删除默认的一行代码：

```
return null;
```

3. 在 CreateBlob 方法中，添加以下代码；代码的功能是创建一个名称为 signin 的容器，并为该容器创建一个类型为 CloudBlobContainer 的对象。

```
CloudBlobContainer container = _blobClient.GetContainerReference("signin");
```

4. 调用 CloudBlobContainer 的对象的 CreateIfNotExists 方法以保证容器确实存在：

```
container.CreateIfNotExists();
```

5. 在 CreateBlob 方法的结尾，右括号之前，创建一个名称为 blobName 的变量：

```
string blobName;
```

6. 使用 String.Format 静态方法创建一个字符串，并将该字符串赋值给 blobName 变量：

```
blobName = String.Format("{0}_SignIn_{1:ddmmyyss}.docx", eventKey, DateTime.UtcNow);
```

7. 在 CreateBlob 方法的结尾，右括号之前，调用容器的 GetBlockBlobReference 方法，以 blobName 变量作为参数来创建一个 Blob 块引用：

```
ICloudBlob blob = container.GetBlockBlobReference(blobName);
```

8. 调用 MemoryStream 的 Seek 方法将流偏移量设置为 0：

```
stream.Seek(0, SeekOrigin.Begin);
```

9. 使用 ICloudBlob 接口的 UploadFromStream 方法将流上传至引用的 blob 块：

```
blob.UploadFromStream(stream);
```

10. 在 CreateBlob 方法的结尾，右括号之前，添加下面的语句作为返回值：

```
return blob.Uri;
```

MCT USE ONLY. STUDENT USE PROHIBITED

**结果:** 完成本次实验以后，你就可以使用 Azure 存储 SDK 来管理你的存储帐户中的 blob 和容器。

### 练习 3：从容器中检索文件和媒体文件

► 任务 1：从 Blob 存储下载文档并流式传送到客户端

1. 在解决方案资源管理器窗格上，展开 **Shared** 解决方案目录。
2. 在解决方案资源管理器窗格上，展开 **Contoso.Events.ViewModels** 项目。
3. 双击打开 **DownloadViewModel.cs** 文件。
4. 在 **DownloadViewModel** 类中，找到如下签名的方法：

```
public async Task<DownloadPayload> GetStream()
```

5. 在 **GetStream** 方法中，删除下面一行代码：

```
return await Task.FromResult<DownloadPayload>(null);
```

6. 在 **GetStream** 方法的结尾，在右括号之前，调用变量 `_storageAccount` 的 **CreateCloudBlobClient** 方法创建一个 **CloudBlobClient** 实例：

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
```

7. 通过调用 `blobClient` 的 **GetContainerReference** 方法，为 `signin` 容器创建一个 **CloudBlobContainer** 的对象：

```
CloudBlobContainer container = blobClient.GetContainerReference("signin");
```

8. 调用 `container` 的 **CreateIfNotExists** 方法，确保容器确实存在：

```
container.CreateIfNotExists();
```

9. 在 **GetStream** 方法的结尾，在右括号之前，调用 `container` 的 **GetBlockBlobReference** 方法，以变量 `_blobId` 作为参数来创建一个 blob 变量：

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
```

10. 调用 `blob` 对象的 **OpenReadAsync** 方法异步创建一个流对象 `blobStream`：

```
Stream blobStream = await blob.OpenReadAsync();
```

11. 在 **GetStream** 方法的结尾，在右括号之前，创建 **DownloadPayload** 类的一个实例：

```
DownloadPayload payload = new DownloadPayload();
```

12. 将 `blobStream` 变量赋值给 `payload` 变量的 **Stream** 属性：

```
payload.Stream = blobStream;
```

13. 将 `blob` 变量的 **Properties.ContentType** 属性值赋值给 `payload` 变量的 **ContentType** 属性：

```
payload.ContentType = blob.Properties.ContentType;
```

14. 返回 **DownloadPayload** 类型的变量 `payload`：

```
return payload;
```

► 任务 2：生成测试数据

1. 在 Start 屏幕，输入 **azure storage emulator**。

2. 单击 **Windows Azure Storage Emulator-v3.4** 磁贴。
3. 切换至 **Contoso.Events–Microsoft Visual Studio** 窗口。
4. 在解决方案资源管理器窗格, 右键单击 **Contoso.Events.Data.Generation** 项目, 指向调试, 然后单击启动新实例。

► **任务 3: 从 blob 存储下载生成的签到表**

1. 在解决方案资源管理器窗格, 右键单击 **Contoso.Events** 解决方案, 然后单击属性。
2. 在解决方案 ‘**Contoso.Events**’ 属性页, 执行下面的步骤:
  - a. 确保左边导航栏中通用属性=>启动项目被选中。
  - b. 选择多启动项目。
  - c. 对于 **Contoso.Events.Cloud**, 设置操作为启动。
  - d. 对于 **Contoso.Events.Management**, 设置操作为开始执行（不调试）。
  - e. 确保剩余项目的操作都被设置成无。
  - f. 单击 **OK** 关闭对话框。
3. 在以下 3 个文件中, 查找 name 为 **EventsContextConnectionString** 的元素, 确认 connectionString 的值一致, 若不一致, 请使用 **Shared\Contoso.Events.Data.Generation** 项目中的 App.config 中的值替换另外两处的值, 连接字符串示例如下:
  - o **Shared\Contoso.Events.Data.Generation** 项目中的 App.config;
  - o **Roles\Contoso.Events.Web** 项目中的 Web.config;
  - o **Administration\Contoso.Events.Management** 项目中的 Web.config;

**连接字符串示例**

```
Data Source=(localdb)\v11.0;Initial Catalog=EventsContextModule8Lab;Pooling=True;Integrated Security=True
```

4. 在调试菜单中, 单击启动调试。
5. 在 Windows 通知栏, 单击箭头查看正在运行的程序。
6. 定位到 **IIS Express**, 右键单击该图标。
7. 在 **View Sites** 中定位到 **Contoso.Events.Management**, 然后单击 **Browse Applications** 下方的 URL。
8. 在 **View Sites** 中定位到 **Contoso.Events.Cloud**, 然后单击 **Browse Applications** 下方的 URL。
9. 切换到 **Home–Contoso.Events Administration** 页面, 单击 **Go to Events List** 按钮。
10. 在列表中的任意事件项中, 单击 **Sign-In Sheet** 按钮。
11. 在 **Sign-in Sheet Generation** 页面, 提示 Sign-In Document Generation In Progress 的通知。
12. 等待 1 到 2 分钟, 单击 refresh 链接刷新 Sign-in 表格页面状态。
13. 单击 **Sign-In Sheet** 链接, 从服务器上下载 sign-in sheet 签到表。

**结果:** 完成本次练习以后, 你就可以通过 Azure 存储 SDK 来从你的存储帐户上下载 blob 了。

## 练习 4：指定容器的权限

### ► 任务 1：使用服务器资源管理器来修改容器访问权限

1. 在桌面上，单击 **Contoso.Events–Visual Studio 2013** 窗口。
2. 在调试菜单，单击**停止调试**。
3. 在视图菜单，单击**服务器资源管理器**。
4. 定位到 **Azure** 节点，然后单击左侧节点上的箭头来展开 Azure 节点。
5. 展开 Azure 节点下的**存储**节点。
6. 展开**存储**节点下面的**开发**节点。
7. 展开**开发**节点下面的 **blob** 节点。
8. 右键单击 **signin** 容器，然后单击**属性**。
9. 在**属性**窗格，定位到**公共读访问**。
10. 确保**公共读访问**的值为 **Off**。

### ► 任务 2：使用 SDK 生成临时 SAS 令牌

1. 在解决方案资源管理器窗格，展开 **Shared** 目录。
2. 在解决方案资源管理器窗格，展开 **Contoso.Events.ViewModels** 项目。
3. 双击打开 **DownloadViewModel.cs** 文件。
4. 在 **DownloadViewModel** 类，找到如下签名的方法：

```
public async Task<string> GetSecureUrl()
```

5. 在 **GetSecureUrl** 方法中，删掉这一行代码：

```
return await Task.FromResult<string>(String.Empty);
```

6. 在 **GetSecureUrl** 方法的结尾，在右括号之前，调用 **\_storageAccount** 的 **CreateCloudBlobClient** 方法创建一个 **CloudBlobClient** 实例。

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
```

7. 调用 **CloudBlobClient** 的变量 **blobClient** 的 **GetContainerReference** 获取 **signin** 容器的一个对象。

```
CloudBlobContainer container = blobClient.GetContainerReference("signin");
```

8. 调用 **CloudBlobContainer** 的变量 **container** 的 **CreateIfNotExists** 方法，以确保 **blob** 容器确实存在：  
:

```
container.CreateIfNotExists();
```

9. 在 **GetSecureUrl** 方法的结尾，在右括号之前创建一个 **SharedAccessBlobPolicy** 类的实例：

```
SharedAccessBlobPolicy blobPolicy = new SharedAccessBlobPolicy();
```

10. 设置 **SharedAccessBlobPolicy** 的变量 **blobPolicy** 的 **SharedAccessExpiryTime** 属性为现在的时间后推 15 分钟：

```
blobPolicy.SharedAccessExpiryTime = DateTime.Now.AddHours(0.25d);
```

11. 设置 **SharedAccessBlobPolicy** 的变量 `blobPolicy` 的 **Permissions** 属性的值为 `SharedAccessBlobPermissions.Read`:

```
blobPolicy.Permissions = SharedAccessBlobPermissions.Read;
```

12. 在 **GetSecureUrl** 方法的结尾, 在右括号之前, 创建 **BlobContainerPermissions** 类的一个实例:

```
BlobContainerPermissions blobPermissions = new BlobContainerPermissions();
```

13. 将 `blobPolicy` 对象添加到新创建的 `blobPermissions` 对象的 **SharedAccessPolicies** 属性列表中。

```
blobPermissions.SharedAccessPolicies
 .Add("ReadBlobPolicy", blobPolicy);
```

14. 设置 **BlobContainerPermissions** 的变量 `blobPermissions` 的 **PublicAccess** 属性为 `BlobContainerPublicAccessType.Off`:

```
blobPermissions.PublicAccess = BlobContainerPublicAccessType.Off;
```

15. 在 **GetSecureUrl** 方法的结尾, 在右括号之前, 异步调用 `container` 的 **SetPermissionsAsync** 方法, 并使用 `blobPermissions` 作为参数:

```
await container.SetPermissionsAsync(blobPermissions);
```

16. 使用 **SharedAccessBlobPolicy** 类的新实例作为一个参数, 字符串 “`ReadBlobPolicy`” 作为第二个参数, 调用 **CloudBlobContainer** 的变量 `container` 的 **GetSharedAccessSignature** 方法:

```
string sasToken = container.GetSharedAccessSignature(new
SharedAccessBlobPolicy(), "ReadBlobPolicy");
```

17. 在 **GetSecureUrl** 方法的结尾, 在右括号之前, 调用 `container` 的 **GetBlockBlobReference** 方法, 并将 `_blobId` 作为参数创建一个 blob 块引用:

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
```

18. 把 **ICloudBlob** 的 `blob` 变量的属性 **Uri** 保存在新的 **Uri** 实例 `blobUrl` 中。

```
Uri blobUrl = blob.Uri;
```

19. 在 **GetSecureUrl** 方法的结尾, 在右括号之前, 把 **Uri** 的 `blobUrl` 变量的属性 **AbsoluteUri** 的值和 `sasToken` 变量拼接成一个新的字符串, 并赋值给新变量 `secureUrl`:

```
string secureUrl = blobUrl.AbsoluteUri + sasToken;
```

20. 将字符串变量 `secureUrl` 作为结果返回:

```
return secureUrl;
```

### ► 任务 3: 使用 SAS 令牌从受保护的容器下载文档

1. 在调试菜单上, 单击启动调试。
2. 在通知栏单击箭头查看所有运行的应用程序。
3. 定位到 **IIS Express**, 右键单击该图标。
4. 单击在 **View Sites** 下方的 **Contoso.Events.Management**, 然后单击在 **Browse Applications** 下面的 URL。
5. 在 **Events Administration** 页面, 单击 **Go to Events List** 按钮以查看事件列表。

6. 单击事件列表中的任意 **Sign-In Sheet** 按钮。
7. 查看 Sign-in 页面并注意到一个 Sign-in 表格正在生成。
8. 等待 1 至 2 分钟，然后刷新 sign-in 表格页面。
9. 单击 **Sign-In Sheet** 按钮来下载 sign-in 表格。
10. 关闭 **Internet Explorer** 应用程序。
11. 关闭 **Contoso.Events - Visual Studio** 应用程序。
12. 关闭 **Azure Storage Emulator** 命令提示窗口。

**结果：**在完成本次练习以后，您会更新 blob 容器上的权限，并且生产访问容器的 SAS 令牌。

## 第 9 章：使用队列和 **Service Bus** 设计通信策略

# 实验 A：使用队列和 **Service Bus** 来管理 Azure 中 Web 应用程序之间的通信

### 练习 1：创建 Azure Service Bus 命名空间

► 任务 1：使用管理门户创建 **Service Bus** 命名空间

 **注释：**Service Bus 功能在预览门户暂不可用，因此这个实验将使用 Azure 管理门户。

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.com>。
3. 在邮箱地址框内，输入 Microsoft 帐户的邮箱地址。
4. 在密码框内，输入 Microsoft 帐户的密码。
5. 单击 **Sign In**。
6. 单击右上角的帐户名前的地球图标。
7. 选择中文（简体）。
8. 在页面左边的导航窗格，下拉滚动条，然后单击 **Service Bus**。
9. 在屏幕下方，单击**创建**。
10. 在**创建命名空间**对话框，执行以下步骤：
  - a. 在**命名空间名称**框内，输入 **sb28532[自定义名称]**。
  - b. 在**区域**列表，选择最接近你所处地区的区域。
  - c. 在**类型**列表，选择**消息**选项。
  - d. 在**消息传递层**列表，选择**STANDARD** 选项。
  - e. 单击对勾标志按钮来创建命名空间。
11. 在**Service Bus**命名空间列表，单击新建的命名空间。
12. 在屏幕底部，单击**连接信息**。
13. 从访问**连接信息**对话框，记录下 RootManageSharedAccessKey 连接字符串。
14. 关闭访问**连接信息**对话框。
15. 在屏幕顶部，单击**队列**选项卡。
16. 在屏幕左下角，单击**新建**。
17. 如果没有自动选中，那么选择应用服务 => **Service Bus** => 队列 => 自定义创建。
18. 在**创建队列**对话框，执行以下步骤：
  - a. 在**队列名称**框内，输入 **signin**。

- b. 在**区域**列表，选择与命名空间一致的区域。
- c. 在**命名空间**框内，选择 **sb28532[自定义名称]**。
- d. 单击下一个箭头，来到向导的第二个步骤。
- e. 所有字段保持默认值。
- f. 单击对勾标志按钮来创建新的队列。

**结果：**完成此练习后，您将使用管理门户创建了 Service Bus 命名空间。

## 练习 2：将 Azure 队列存储用于文档生成

► 任务 1：将辅助角色更新为使用来自队列的请求

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏，单击 **File Explorer** 图标。
3. 在 This PC 窗口，浏览至 **Allfiles(F):\Mod09\Labfiles\Starter\Queues\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格，展开 **Roles** 文件夹。
5. 在解决方案资源管理器窗格，展开 **Contoso.Events.Worker** 项目。
6. 双击 **TableStorageQueueHelper.cs** 文件。
7. 在 **TableStorageQueueHelper** 构造函数尾部，右括号之前，存储基类的 **StorageAccount** 属性于 **CloudStorageAccount** 变量：

```
CloudStorageAccount storageAccount = base.StorageAccount;
```

8. 调用 **CreateCloudQueueClient** 方法并把结果赋于 **\_queueClient** 变量：

```
_queueClient = storageAccount.CreateCloudQueueClient();
```

9. 调用静态方法 **CloudConfigurationManager.GetSetting** 并把结果赋于 **\_signInQueueName** 变量：

```
_signInQueueName = CloudConfigurationManager.GetSetting("SignInQueueName");
```

10. 在 **TableStorageQueueHelper** 类内，找到以下构造函数：

```
IQueueMessage<CloudQueueMessage> Receive()
```

11. 删掉类内的代码行：

```
return new TableStorageQueueMessage(null);
```

12. 在 **Receive** 方法尾部，右括号之前，创建一个 **CloudQueue** 类的新实例，使其调用 **CloudQueueClient** 变量的 **GetQueueReference** 方法，代码如下：

```
CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);
```

13. 调用 **CreateIfNotExists** 方法以确保队列存在。

```
queue.CreateIfNotExists();
```

14. 在 **Receive** 方法尾部，右括号之前，调用 **CloudQueue** 类的 **GetMessage** 方法，并把值存储于 **CloudQueueMessage** 变量，代码如下：

```
CloudQueueMessage message = queue.GetMessage();
```

15. 把 **CloudQueueMessage** 变量的结果传递给 **TableStorageQueueMessage** 类的构造函数：

```
return new TableStorageQueueMessage(message);
```

16. 在 **CompleteMessage** 方法尾部，右括号之前，创建一个 **CloudQueue** 类的新实例，使其调用 **CloudQueueClient** 变量的 **GetQueueReference** 方法，代码如下：

```
CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);
```

17. 调用 **CreateIfNotExists** 方法以确保队列存在:

```
queue.CreateIfNotExists();
```

18. 在 **CompleteMessage** 方法尾部, 右括号之前, 调用 **DeleteMessage** 方法, 使其把 **CloudQueueMessage** 变量作为参数, 代码如下:

```
queue.DeleteMessage(message);
```

► 任务 2: 将管理应用程序更新为将请求添加到队列

1. 在解决方案资源管理器窗格, 展开 **Shared** 文件夹。
2. 在解决方案资源管理器窗格, 展开 **Contoso.Events.ViewModels** 项目。
3. 双击 **SignInSheetViewModel.cs** 文件。
4. 在 **GenerateSignInSheetTableStorage** 方法头部, 左括号之后, 创建一个 **CloudStorageAccount** 实例, 使其调用 **CloudStorageAccount.Parse** 静态方法和 **tableStorageConnectionString**, 代码如下:

```
CloudStorageAccount storageAccount =
CloudStorageAccount.Parse(tableStorageConnectionString);
```

5. 创建一个 **CloudQueueClient** 类的新实例, 使其调用 **CloudStorageAccount** 变量的 **CreateCloudQueueClient** 方法:

```
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
```

6. 在以上代码之后, 创建一个新的 **CloudQueue** 类的实例, 使其调用 **CloudQueueClient** 变量的 **GetQueueReference** 方法, 代码如下:

```
CloudQueue queue = queueClient.GetQueueReference(signInQueueName);
```

7. 调用 **CloudQueue** 类的 **CreateIfNotExists** 方法以确保队列存在:

```
queue.CreateIfNotExists();
```

8. 在以上代码之后, 创建一个新的 **CloudQueueMessage** 类的实例, 使其把字符串 **message** 传递给构造函数:

```
CloudQueueMessage queueMessage = new CloudQueueMessage(message);
```

9. 调用 **CloudQueue** 变量的 **AddMessage** 方法, 使其使用 **CloudQueueMessage** 作为参数:

```
queue.AddMessage(queueMessage);
```

► 任务 3: 生成测试数据

1. 在 Start 屏幕, 输入 **Azure Storage Emulator**。
2. 单击 **Windows Azure Storage Emulator – v3.4** 磁贴。
3. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
4. 在解决方案资源管理器窗格, 右键单击 **Contoso.Events.Data.Generation** 项目, 指向调试, 然后单击启动新实例。

 **注释:** 如果弹出保存只读文件消息框, 单击**覆盖**。

#### ► 任务 4：调试和验证应用程序

1. 在解决方案资源管理器窗格，右键单击解决方案‘Contoso.Events’，然后单击属性。
2. 在解决方案“Contoso.Events”属性页对话框，执行以下步骤：
  - a. 在对话框的左侧导航菜单，确保通用属性 => 启动项目被选中。
  - b. 选择多启动项目。
  - c. 把 Contoso.Events.Cloud 的操作选项设为启动。
  - d. 把 Contoso.Events.Management 的操作选项设为开始执行（不调试）。
  - e. 确保剩余其它项目的操作选项设为无。
  - f. 单击 OK。
  - g. 确认 Contoso.Events.Management=>Web.config、Contoso.Events.Web=>Web.config、Contoso.Events.Worker=>app.config 和 Contoso.Events.Data.Generation=>App.config 文件中 connectionStrings 中的 Initial Catalog 属性都设置为 EventsContextModule9Lab。
3. 在调试菜单，单击启动调试。

 **注释：**从始至终，这个课程都将使用 **Azure Compute Emulator** 来测试和调试云服务。在将云服务布置到 Azure 平台之前，此模拟器提供了一个快捷的本地选项来测试云服务。偶尔，它将会遇到一些奇怪的问题，比如错误的端口号，页面不能显示等。如果出现这种情况，你只要关闭这个模拟器，在下次调试项目时，Visual Studio 将会自动启动该模拟器。

4. 在通知区域，单击箭头查看运行中的应用程序。
5. 右键单击 IIS Express 图标。
6. 指向 View Sites 下的 Contoso.Events.Management 选项，然后单击 Browse Applications 下的 URL。
7. 指向 View Sites 下的 Contoso.Events.Cloud 选项，然后单击 Browse Applications 下的 URL。
8. 切换到 Contoso.Events – Microsoft Visual Studio 窗口。
9. 单击视图菜单，然后选择解决方案资源管理器选项。
10. 在解决方案资源管理器窗格，展开 Roles 文件夹。
11. 在解决方案资源管理器窗格，展开 Contoso.Events.Worker 项目。
12. 双击 WorkerRole.cs 文件。
13. 定位到 Run 方法。
14. 在 try-catch 内，定位到以下目标行代码：

```
HandleMessage(message);
```

15. 右键单击目标行代码，指向断点，单击插入断点。
16. 切换到 Home - Contoso.Events.Administration 窗口。
17. 在 Administration 页面应用程序的主页，单击 Events 按钮。
18. 单击任意 event 的 Sign-In Sheet。
19. 查看 sign-in 页面，你将注意到 sign-in sheet 正在生成，同时显示以下信息：Sign-In Document Generation in Progress。

20. 等待一分钟，等辅助角色（worker role）接收队列消息。
21. 验证应用程序在断点处暂停运行。
22. 按下 F5 键以恢复应用程序运行。
23. 等待一分钟，然后刷新 sign-in sheet 页面。
24. 单击 **Sign-In Sheet** 从服务器下载 sign-in sheet。
25. 关闭 **Internet Explorer** 应用程序。

**结果：**完成此练习后，您将创建并使用了存储队列中的消息。

### 练习 3：将 Service Bus 队列用于文档生成

► 任务 1：将辅助角色更新为使用来自队列的请求

1. 在解决方案资源管理器窗格，展开 **Contoso.Events.Cloud** 项目。
2. 双击 **ServiceConfiguration.Local.cscfg** 文件。
3. 定位到 name 属性为 **Microsoft.ServiceBus.ConnectionString** 的 **Setting** 元素。
4. 用之前记录的连接字符串替换 value 属性的值。
5. 在解决方案资源管理器窗格，展开 **Roles** 文件夹。
6. 在解决方案资源管理器窗格，展开 **Contoso.Events.Worker** 项目。
7. 双击 **ServiceBusQueueHelper.cs** 文件。
8. 在 **ServiceBusQueueHelper** 构造函数的尾部，右括号之前，把云配置中的 **Microsoft.ServiceBus.ConnectionString** 设定值存储于一个字符串变量中，代码如下：

```
string serviceBusConnectionString =
CloudConfigurationManager.GetSetting("Microsoft.ServiceBus.ConnectionString");
```

9. 把队列名存储于一个字符串变量中。

```
string signInQueueName = CloudConfigurationManager.GetSetting("SignInQueueName");
```

10. 调用静态方法 **QueueClient.CreateFromConnectionString**，把队列名和连接字符串作为参数，并把结果赋于 *\_client* 变量，代码如下：

```
_client = QueueClient.CreateFromConnectionString(serviceBusConnectionString,
signInQueueName);
```

11. 在 **ServiceBusQueueHelper** 类内部，找到以下方法：

```
IQueueMessage<BrokeredMessage> Receive()
```

12. 删掉以下在类中的代码行：

```
return new ServiceBusQueueMessage(null);
```

13. 在 **Receive** 方法的尾部，右括号之前，创建一个新的 **CloudQueue** 类实例，使其调用 *CloudQueueClient* 变量的 **GetQueueReference** 方法，代码如下：

```
BrokeredMessage message = _client.Receive();
```

14. 调用 **CreateIfNotExists** 方法以确保队列存在：

```
return new ServiceBusQueueMessage(message);
```

15. 在 **CompleteMessage** 方法的尾部，右括号之前，调用 **message** 参数的 **Complete** 方法，代码如下：

```
message.Complete();
```

16. 在 **AbandonMessage** 方法的尾部，右括号之前，调用 **message** 参数的 **Abandon** 方法，代码如下：

```
message.Abandon();
```

17. 在解决方案资源管理器窗格，展开 **Roles** 文件夹。

18. 在解决方案资源管理器窗格，展开 **Contoso.Events.Worker** 项目。

19. 双击 **WorkerRole.cs** 文件。
20. 定位到以下创建 **TableStorageQueueHelper** 实例的代码行：

```
var queueHelper = new TableStorageQueueHelper();
```

用以下创建新 **ServiceBusQueueHelper** 类实例的代码来替换以上代码

```
var queueHelper = new ServiceBusQueueHelper();
```

► 任务 2：将管理应用程序更新为将请求添加到队列

1. 在解决方案资源管理器窗格，展开 **Administration** 文件夹，然后展开 **Contoso.Events.Management** 项目。
2. 双击 **Web.config** 文件。
3. 定位到 **appSettings** 元素。
4. 定位到 key 属性为 **Microsoft.ServiceBus.ConnectionString** 的 **add** 元素。
5. 用之前记录的连接字符串替换 **value** 属性的值。
6. 在解决方案资源管理器窗格，展开 Shared 文件夹。
7. 在解决方案资源管理器窗格，展开 **Contoso.Events.ViewModels** 项目。
8. 双击 **SignInSheetViewModel.cs** 文件。
9. 在构造函数内，定位到以下代码行：

```
GenerateSignInSheetTableStorage(context, eventItem, messageString);
```

10. 用以下代码行替换上面的代码：

```
GenerateSignInSheetServiceBus(context, eventItem, message);
```

11. 在 **GenerateSignInSheetServiceBus** 方法的头部，左括号之后，创建一个使用连接字符串的 **QueueClient** 实例，代码如下：

```
QueueClient client = QueueClient.CreateFromConnectionString(serviceBusConnectionString,
signInQueueName);
```

12. 在以上代码之后，创建一个新的 **BrokeredMessage** 类的实例，使其传递 **QueueMessage** 消息到构造函数，代码如下：

```
BrokeredMessage queueMessage = new BrokeredMessage(message);
```

13. 调用 **QueueClient** 变量的 **Send** 方法，使其使用 **BrokeredMessage** 作为参数：

```
client.Send(queueMessage);
```

► 任务 3：调试和验证应用程序

1. 在调试菜单，单击启动调试。
2. 在通知中心，单击箭头查看运行中的应用程序。
3. 定位到 **IIS Express** 图标，然后右键单击图标。
4. 指向 **View Sites** 下的 **Contoso.Events.Management** 选项，然后单击 **Browse Applications** 下的 URL。

MCT USE ONLY. STUDENT USE PROHIBITED

5. 指向 **View Sites** 下的 **Contoso.Events.Cloud** 选项，单击 **Browse Applications** 下的 URL。
6. 在 **Administration** Web 应用程序的主页，单击 **Events** 按钮查看 event 列表。
7. 单击列表里任意 event 的 **Sign-In Sheet** 按钮。
8. 查看 sign-in 页面，你将注意到 sign-in 正在生成，同时显示以下信息：**Sign-In Document Generation in Progress**。
9. 等待一分钟，等辅助角色接受队列消息。
10. 验证应用程序在断点处暂停运行。
11. 按下 F5 键以恢复应用程序的运行。
12. 等待一分钟，然后刷新 sign-in sheet 页面。
13. 单击 **Sign-In Sheet** 从服务器下载 sign-in sheet。
14. 关闭 **Internet Explorer** 窗口。
15. 关闭 **Contoso.Events – Microsoft Visual Studio** 窗口。

**结果：**完成此练习后，您将创建并使用了 Service Bus 队列中的消息。

## 练习 4：使用 Service Bus 中继连接 WCF 服务和客户端

### ► 任务 1：使用控制台应用程序测试现有 WCF 服务

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏，单击 **File Explorer** 图标。
3. 在 This PC 窗口，浏览至 **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在 Start 屏幕，单击 **Desktop** 磁贴。
5. 在任务栏，单击 **File Explorer** 图标。
6. 在 This PC 窗口，浏览至 Allfiles  
**(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events.Lodging.Client**，然后双击 **Contoso.Events.Lodging.Client.sln**。
7. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
8. 在解决方案资源管理器窗格，展开 **On-Premise** 文件夹，右键单击 **Contoso.Events.Lodging** 项目，然后单击 **设为启动项目**。
9. 在调试菜单，单击 **启动调试**。
10. 你将看到一个提示表明 WCF 服务没有任何元数据并询问是否希望退出调试，单击 **No**。
11. 最小化 **WCF Test Client**。
12. 切换到 **Contoso.Events.Lodging.Client – Microsoft Visual Studio** 窗口。
13. 在调试菜单，单击 **启动调试**。
14. 验证有四家酒店名称被列出。
15. 按任意键关闭控制台窗口。
16. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
17. 在调试菜单，单击 **停止调试**。

### ► 任务 2：将 WCF 服务端点修改为使用 Service Bus 中继

1. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
2. 在解决方案资源管理器窗格，展开 **On-Premise** 文件夹。
3. 在解决方案资源管理器窗格，展开 **Contoso.Events.Lodging** 项目。
4. 双击 **App.config** 文件。
5. 定位到 **system.serviceModel XML** 元素。
6. 定位到 **services** XML 子元素。
7. 定位到 **contract** 属性的值为 **Contoso.Events.Models.ILodgingService** 的 **endpoint XML** 子元素。
8. 用以下值替换 **binding** 属性的值：**netTcpRelayBinding**。
9. 定位到 **contract** 属性的值为 **Contoso.Events.Models.ILodgingService** 的 **endpoint XML** 子元素。
10. 用以下值替换 **address** 属性的值：**sb://**。
11. 把命名空间名称添加到 **address** 属性的值后面。
12. 把以下值添加到 **address** 属性的值的后面：**.servicebus.windows.net/lodging**。
13. 定位到 **system.serviceModel XML** 元素。

14. 定位到 **behaviors** XML 子元素。
15. 定位到 **endpointBehaviors** XML 子元素。
16. 定位到 **name** 属性的值为 **ServiceBusRelayBehavior** 的 **behavior** XML 子元素，代码如下：

```
<behavior name="ServiceBusRelayBehavior">
</behavior>
```

17. 在 **behavior** 元素的标签内，添加一个 **transportClientEndpointBehavior** 元素，代码如下：

```
<transportClientEndpointBehavior>
</transportClientEndpointBehavior>
```

18. 在 **transportClientEndpointBehavior** 元素的标签内，添加一个 **tokenProvider** 元素，代码如下：

```
<tokenProvider>
</tokenProvider>
```

19. 在 **tokenProvider** 元素的标签内，添加一个 **sharedAccessSignature** 元素。

```
<sharedAccessSignature keyName="[keyName]" key="[key]" />
```

20. 在 **sharedAccessSignature** 元素内。

21. 删掉 **keyName** 属性的值。
22. 用之前记录的 Service Bus 连接字符串的 **SharedAccessKeyName** 的值替换 **keyName** 属性的值。
23. 在 **sharedAccessSignature** 元素内，删掉 **key** 属性的值。
24. 用之前记录的 Service Bus 连接字符串的 **SharedAccessKey** 的值替换 **key** 属性的值。



**注释：**例如你的连接字符串为：

Endpoint=sb://contoso.servicebus.windows.net/  
SharedAccessKeyName=RootManageSharedAccessKey;  
SharedAccessKey=ABC123/J3y+tk=  
**keyName** 的值就是“RootManageSharedAccessKey”然后 **key** 的值就是“ABC123/J3y+tk=”。

25. 在解决方案资源管理器窗格，展开 **On-Premise** 解决方案文件夹，右键单击 **Contoso.Events.Lodging** 项目，然后单击设为启动项目。
26. 在调试菜单，单击启动调试。
27. 你将看到一个提示表明 WCF 服务没有任何元数据并询问是否希望退出调试，单击 **No**。
28. 最小化，但是不要关闭 **WCF Test Client** 应用程序。

#### ► 任务 3：使用控制台应用程序测试中继的 WCF 服务

1. 切换到 **Contoso.Events.Lodging.Client - Microsoft Visual Studio** 窗口。
2. 在解决方案资源管理器窗格，展开 **Contoso.Events.Lodging.Client** 项目。
3. 双击 **Program.cs** 文件。
4. 定位到以下方法：

```
private static IEnumerable<Hotel> GetHotels()
```

5. 删掉以下通过使用静态字符串来创建 Uri 的代码：

MCT USE ONLY STUDENT USE PROHIBITED

```
Uri serviceUri = new Uri("net.tcp://localhost:8000/lodging", UriKind.Absolute);
```

6. 用以下代码行替换上面的代码，该行代码通过调用静态方法 **ServiceBusEnvironment.CreateServiceUri** 来创建 Uri，该方法把协议、命名空间和相关的名称作为其参数：

```
Uri serviceUri = ServiceBusEnvironment.CreateServiceUri("sb", "[namespace]", "lodging");
```

7. 把以上 **CreateServiceUri** 方法内的参数 “[namespace]” 替换成 Service Bus 命名空间。
8. 删掉以下代码行：

```
new NetTcpBinding(),
```

9. 用以下代码行替换，该行代码创建了一个 **NetTcpRelayBinding** 类的新实例：

```
new NetTcpRelayBinding(),
```

10. 在创建 **ChannelFactory<ILodgingService>** 实例之后，创建一个 **TransportClientEndpointBehavior** 类的实例：

```
TransportClientEndpointBehavior endpointBehavior = new TransportClientEndpointBehavior();
```

11. 在下一行，把 **TransportClientEndpointBehavior** 变量的 **TokenProvider** 属性作为调用静态方法 **TokenProvider.CreateSharedSecretTokenProvider** 的结果：

```
endpointBehavior.TokenProvider =
TokenProvider.CreateSharedAccessSignatureTokenProvider("[keyName]", "[key]");
```

12. 在上一行代码中，更新 **CreateSharedSecretTokenProvider** 方法的参数，用之前记录的 Service Bus 连接字符串的 **SharedAccessKey** 的值替换 “[key]” 字符串。
13. 在上一行代码中，更新 **CreateSharedSecretTokenProvider** 方法的参数，用之前记录的 Service Bus 连接字符串的 **SharedAccessKeyName** 的值替换 “[keyName]” 字符串。
14. 添加 **TransportClientEndpointBehavior** 的新实例到 **cf.Endpoint** 变量的 **EndpointBehaviors** 整合属性：

```
cf.Endpoint.EndpointBehaviors.Add(endpointBehavior);
```

15. 在调试菜单，单击启动调试。
16. 验证酒店被列出。
17. 按任意键关闭控制台窗口。

#### ► 任务 4：将云 Web 应用程序更新为连接到中继的 WCF 服务

1. 定位到 **Contoso.Events.Lodging.Client** 项目的 **Program.cs** 文件。
2. 定位到以下方法。

```
private static IEnumerable<Hotel> GetHotels()
```

3. 选中该方法的主体。
4. 右键单击选中部份，然后单击复制。
5. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
6. 在调试菜单，单击停止调试。

7. 在解决方案资源管理器窗格，展开 **Shared** 文件夹。
8. 在解决方案资源管理器窗格，展开 **Contoso.Events.ViewModels** 项目。
9. 双击 **HotelsViewModel.cs** 文件。
10. 定位到以下方法：

```
private static IEnumerable<Hotel> GetHotels()
```

11. 选中整个方法的主体。
12. 右键单击选中部份，然后单击粘贴。

#### ► 任务 5：调试云 Web 应用程序

1. 在解决方案资源管理器窗格，右键单击解决方案 **Contoso.Events**，然后单击属性。
2. 在解决方案 **Contoso.Events** 属性页对话框，执行以下步骤：
  - a. 在左侧的导航菜单，确保通用属性 => 启动项目被选中。
  - b. 选择多启动项目。
  - c. 把 **Contoso.Events.Cloud** 的操作选项设为开始执行（不调试）。
  - d. 把 **Contoso.Events.Lodging** 的操作选项设为开始执行（不调试）。
  - e. 把 **Contoso.Events.Management** 的操作选项设为无。
  - f. 确保所有剩余项目的操作选项设为无。
  - g. 单击 **OK**。
3. 在 Start 屏幕，输入 **Azure Storage Emulator**。
4. 单击 **Windows Azure Storage Emulator – v3.4** 磁贴。
5. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
6. 在解决方案资源管理器窗格，右键单击 **Contoso.Events.Data.Generation** 项目，指向调试，然后单击启动新实例。
7. 等待调试完成，等待控制台窗口关闭。
8. 关闭 **Contoso.Events – Microsoft Visual Studio** 应用程序。
9. 在 Start 屏幕，定位到 **Visual Studio 2013** 磁贴，右键单击，然后选择 **Run as Administrator**。



**注释：**你可能需要在 Start 屏幕，单击向下箭头后定位到 Visual Studio 2013 磁贴。在你以管理员身份启动该应用程序后，你可能会看到 UAC 的提示。单击 **Yes** 继续。

10. 在文件菜单，指向打开，然后单击项目/解决方案。
11. 在打开项目对话框，执行以下步骤：
  - a. 浏览至 **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events**。
  - b. 单击 **Contoso.Events Microsoft Visual Studio Solution** 文件。
  - c. 单击 **Open**。
12. 在调试菜单，单击开始执行（不调试）。
13. 你将会看到一个提示表明 WCF 服务没有任何元数据以及询问是否希望退出调试。单击 **No**。
14. 最小化，但是不要关闭 **WCF Test Client** 应用程序。

15. 在页面应用程序的主页，单击任意 event。
16. 在 event 详细页面，单击 **Register Now**。
17. 填写任意值以完成注册。
18. 单击 **Submit**。
19. 观察到 **Registration Confirmation** 页面显示了酒店的列表。
20. 关闭 **Internet Explorer** 应用程序。
21. 关闭 **Microsoft Visual Studio** 应用程序。
22. 关闭 **Windows Azure Storage Emulator** 命令提示符。

**结果：**完成此练习后，您将使用 Service Bus 中继将现有 WCF 应用程序连接到了客户端。

# 实验 B：在中国版 Windows Azure 中使用队列和 Service Bus 来管理 Web 应用程序间的通信

## 练习 1：创建 Azure Service Bus 命名空间

### ► 任务 1：使用管理门户创建 Service Bus 命名空间

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 进入 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的地球图表，并在菜单中选择中文(简体)语言。
5. 在页面左侧的导航栏中，滚动鼠标找到 **Service Bus**，单击 **Service Bus**。
6. 在页面下方的工具栏左侧，单击新建按钮。
7. 在弹出的**新建**菜单中，选择 **SERVICE BUS**，单击队列，单击自定义创建。
8. 在**添加新队列**窗口中，执行以下步骤：
  - a. 在队列名称文本框中，输入 **signin**。
  - b. 在**区域**下拉框中，选择一个距离你较近的区域。
  - c. 在**命名空间**下拉框中，选择**创建新命名空间**。
  - d. 在命名空间名称文本框中，输入 **sb28532[自定义名称]**。
  - e. 在窗口右下角，单击**下一步箭头**。
9. 在**配置队列**窗口中，单击**完成**按钮。
10. 在 **service bus** 页面，在列表中，单击进入新创建的命名空间，命名空间名称类似于 **sb28532[自定义名称]**。
11. 在页面下方的工具栏中，单击**连接信息**。
12. 在访问**连接信息**对话框，记录 RootManageSharedAccessKey 的连接字符串。



**注释：**你需要从 SAS 列表里，记下连接字符串。

13. 关闭访问**连接信息**对话框。

**结果：**完成此练习后，您将使用管理门户创建了 Service Bus 命名空间。

## 练习 2：将 Azure 队列存储用于文档生成

### ► 任务 1：更新辅助角色以处理来自队列的请求

1. 在 Start 屏幕，单击 **Desktop**。
2. 在任务栏，单击 **File Explorer** 磁贴。
3. 打开 **Allfiles(F):\Mod09\Labfiles\Starter\Queues\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格，展开 **Roles** 解决方案目录。
5. 展开 **Contoso.Events.Worker** 项目。
6. 双击 **TableStorageQueueHelper.cs** 文件。
7. 在 **TableStorageQueueHelper** 类的构造函数结尾，在右括号之前，把基类的 **StorageAccount** 属性保存到 **CloudStorageAccount** 变量中：

```
CloudStorageAccount storageAccount = base.StorageAccount;
```

8. 调用 **CreateCloudQueueClient** 方法并把返回结果赋值给变量 **\_queueClient**：

```
_queueClient = storageAccount.CreateCloudQueueClient();
```

9. 调用静态方法 **CloudConfigurationManager.GetSetting** 并把返回结果赋值给变量 **\_signInQueueName**：

```
_signInQueueName = CloudConfigurationManager.GetSetting("SignInQueueName");
```

10. 在 **TableStorageQueueHelper** 类，找到如下签名的方法：

```
IQueueMessage<CloudQueueMessage> Receive()
```

11. 删除此行代码：

```
return new TableStorageQueueMessage(null);
```

12. 在 **Receive** 方法结尾，右括号前，通过调用 **CloudQueueClient** 变量的 **GetQueueReference** 方法和队列的字符串名字创建一个 **CloudQueue** 类的实例，代码如下：

```
CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);
```

13. 调用 **CreateIfNotExists** 方法来保证队列存在。

```
queue.CreateIfNotExists();
```

14. 在 **Receive** 方法结尾，右括号前，调用 **CloudQueue** 类的 **GetMessage** 方法，并把返回结果赋值给一个 **CloudQueueMessage** 变量，代码如下：

```
CloudQueueMessage message = queue.GetMessage();
```

15. 把 **CloudQueueMessage** 变量传给 **TableStorageQueueMessage** 类的构造函数，并返回结果：

```
return new TableStorageQueueMessage(message);
```

16. 在 **CompleteMessage** 方法结尾，右括号前，通过调用 **CloudQueueClient** 变量的 **GetQueueReference** 方法和一个字符串队列名来创建一个 **CloudQueue** 类的实例，代码如下：

```
CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);
```

17. 调用 **CreateIfNotExists** 方法保证队列存在:

```
queue.CreateIfNotExists();
```

18. 在 **CompleteMessage** 方法结尾, 右括号前, 用 *CloudQueueMessage* 变量作参数, 调用方法 **DeleteMessage**, 代码如下:

```
queue.DeleteMessage(message);
```

#### ► 任务 2: 更新 Administration 应用以向队列中添加请求

1. 在解决方案资源管理器窗格, 展开 **Shared** 解决方案目录。
2. 在解决方案资源管理器窗格, 展开 **Contoso.Events.ViewModels** 项目。
3. 双击 **SignInSheetViewModel.cs** 文件。
4. 在 **GenerateSignInSheetTableStorage** 方法开头, 左括号后面, 通过调用静态方法 **CloudStorageAccount.Parse** 和存储表的连接字符串来创建 **CloudStorageAccount** 实例, 代码如下:

```
CloudStorageAccount storageAccount =
CloudStorageAccount.Parse(tableStorageConnectionString);
```

5. 调用 **CloudStorageAccount** 变量的 **CreateCloudQueueClient** 方法, 创建一个 **CloudQueueClient** 类的实例:

```
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
```

6. 在以上代码之后, 通过调用 **CloudQueueClient** 变量的 **GetQueueReference** 方法和队列名的变量创建一个 **CloudQueue** 类的实例, 代码如下:

```
CloudQueue queue = queueClient.GetQueueReference(signInQueueName);
```

7. 调用 **CloudQueue** 类的 **CreateIfNotExists** 方法保证队列存在:

```
queue.CreateIfNotExists();
```

8. 在以上代码之后, 把一个字符串 **message** 传递给 **CloudQueueMessage** 类的构造函数来创建一个实例:

```
CloudQueueMessage queueMessage = new CloudQueueMessage(message);
```

9. 调用 **CloudQueue** 变量的 **AddMessage** 方法, 使用 **CloudQueueMessage** 作为参数:

```
queue.AddMessage(queueMessage);
```

#### ► 任务 3: 生成测试数据

1. 在 Start 屏幕, 输入 **Azure Storage Emulator**。
2. 单击 **Windows Azure Storage Emulator – v3.4** 磁贴。
3. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
4. 在解决方案资源管理器窗格, 右键单击 **Contoso.Events.Data.Generation** 项目, 单击调试 -> 启动新实例。

 **注释:** 如果弹出保存只读文件对话框, 单击覆盖按钮。

#### ► 任务 4：调试并验证应用程序

- 在解决方案资源管理器窗格，展开 **Administration** 文件夹，展开 **Contoso.Events.Management** 项目，双击 Web.config 文件，找到<connectionStrings>元素，找到 name 的值为 EventsContextConnectionString 的元素，将 connectionString 的值用以下代码替换：

```
Data Source=(localdb)\v11.0;Initial Catalog=EventsContextModule9Lab;Pooling=True;Integrated Security=True
```

- 在解决方案资源管理器窗格，右键单击解决方案 **Contoso.Events**，单击属性。
- 在解决方案 **Contoso.Events** 的属性对话框，执行以下步骤：
  - 在属性对话框左边的导航菜单，选择通用属性 => 启动项目。
  - 选择多启动项目。
  - 设置 **Contoso.Events.Cloud** 项目，将操作修改为启动。
  - 设置 **Contoso.Events.Management** 项目，将操作修改为开始执行（不调试）。
  - 确保其他项目的操作是无。
  - 单击 **OK**。
- 在调试菜单，单击启动调试。

 **注释：**如果弹出保存只读文件对话框，单击覆盖按钮。

在这门课程中，我们会用到 **Azure Compute Emulator** 来测试和调试我们的云服务。

在发布云服务到 Azure 平台之前，模拟器提供了一个快速的本地测试你的云服务的方法。偶尔你会遇到一些奇怪的模拟器错误，比如不正确的端口号或页面不显示，你只需要关闭模拟器，下次 Visual Studio 启动调试的时候会自动启动模拟器。

- 在通知区域，单击箭头来查看运行的应用。
- 找到并右键单击 **IIS Express** 图标。
- 在 **View Sites** 下面找到 **Contoso.Events.Management** 选项并单击，然后单击 **Browse Applications** 下面的 URL。
- 鼠标指向 **View Sites** 下的 **Contoso.Events.Cloud** 选项，然后单击 **Browse Applications** 下的 URL。
- 在桌面，单击 **Contoso.Events – Microsoft Visual Studio** 窗口。
- 单击视图菜单，然后选择解决方案资源管理器选项。
- 在解决方案资源管理器窗格，展开 **Roles** 文件夹。
- 在解决方案资源管理器窗格，展开 **Contoso.Events.Worker** 项目。
- 双击 **WorkerRole.cs** 文件。
- 找到 **Run** 方法。
- 在 try-catch 代码块里找到此行代码：

```
HandleMessage(message);
```

- 右键单击此行代码，在弹出菜单中选中 **断点->插入断点**。
- 在浏览器窗口中，找到 **Home - Contoso.Events.Administration** 窗口。
- 在 **Administration** Web 应用程序主页，单击 **Events** 按钮到 events 列表。

19. 对任意一个 event，单击 **Sign-In Sheet**。
20. 在 sign-in 页面，有以下信息提示 sign-in sheet 正在被创建： **Sign-In Document Generation in Progress**。
21. 等待一分钟，辅助角色（Worker Role）会接收到队列信息。
22. 确保应用程序的执行停在断点处。
23. 按 F5 继续执行。
24. 等待一分钟，然后刷新 sign-in sheet 页面。
25. 单击 **Sign-In Sheet** 从服务器下载 sign-in sheet。
26. 关闭 **Internet Explorer** 程序。

**结果：**完成此练习后，您将创建并使用了存储队列中的消息。

### 练习 3：将 Service Bus 队列用于文档生成

#### ► 任务 1：更新辅助角色以处理来自队列的请求

1. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Cloud** 项目。
2. 双击 **ServiceConfiguration.Local.cscfg** 文件。
3. 找到 name 的值为 **Microsoft.ServiceBus.ConnectionString** 的 **Setting** 元素。
4. 用你之前记录的连接字符串替换 **ConnectionString** 的值。
5. 在解决方案资源管理器窗格中，展开 **Roles** 文件夹。
6. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Worker** 项目。
7. 双击 **ServiceBusQueueHelper.cs** 文件。
8. 在 **ServiceBusQueueHelper** 构造函数结尾，右括号之前，把 **Microsoft.ServiceBus.ConnectionString** 的值存储在一个字符串中，代码如下：

```
string serviceBusConnectionString =
CloudConfigurationManager.GetSetting("Microsoft.ServiceBus.ConnectionString");
```

9. 把队列名赋值给一个字符串变量。

```
string signInQueueName = CloudConfigurationManager.GetSetting("SignInQueueName");
```

10. 用队列名和连接字符串作为参数，调用静态方法 **QueueClient.CreateFromConnectionString**，把结果赋值给 **\_client** 变量，代码如下：

```
_client = QueueClient.CreateFromConnectionString(serviceBusConnectionString,
signInQueueName);
```

11. 在 **ServiceBusQueueHelper** 类里，找到以下签名的方法：

```
IQueueMessage<BrokeredMessage> Receive()
```

12. 删除此行代码：

```
return new ServiceBusQueueMessage(null);
```

13. 在 **Receive** 方法结尾，右括号前，调用 **\_client** 变量的 **Receive** 方法，并把返回结果赋值给一个 **BrokeredMessage** 变量，代码如下：

```
BrokeredMessage message = _client.Receive();
```

14. 把 **BrokeredMessage** 变量传给 **ServiceBusQueueMessage** 类的构造函数，并返回结果：

```
return new ServiceBusQueueMessage(message);
```

15. 在 **CompleteMessage** 方法结尾，右括号前，调用 **message** 参数的 **Complete** 方法，代码如下：

```
message.Complete();
```

16. 在 **AbandonMessage** 方法结尾，右括号前，调用 **message** 参数的 **Abandon** 方法，代码如下：

```
message.Abandon();
```

17. 在解决方案资源管理器窗格，展开 **Roles** 文件夹。

18. 在解决方案资源管理器窗格，展开 **Contoso.Events.Worker** 项目。

19. 双击 **WorkerRole.cs** 文件。
20. 找到创建 **TableStorageQueueHelper** 实例的代码:

```
var queueHelper = new TableStorageQueueHelper();
```

将其替换为创建 **ServiceBusQueueHelper** 的实例:

```
var queueHelper = new ServiceBusQueueHelper();
```

► **任务 2: 更新 Administration 应用以向队列中添加请求**

1. 在解决方案资源管理器窗格, 展开 **Administration** 文件夹, 然后展开 **Contoso.Events.Management** 项目。
2. 双击 **Web.config** 文件。
3. 找到 **appSettings** 元素。
4. 找到 key 的值为 **Microsoft.ServiceBus.ConnectionString** 的 **add** 元素。
5. 用之前记录的连接字符串替换它的 **value** 属性的值。
6. 在解决方案资源管理器窗格, 展开 **Shared** 文件夹。
7. 在解决方案资源管理器窗格, 展开 **Contoso.Events.ViewModels** 项目。
8. 双击 **SignInSheetViewModel.cs** 文件。
9. 在构造方法中, 找到此行代码:

```
GenerateSignInSheetTableStorage(context, eventItem, messageString);
```

10. 用以下代码替换它:

```
GenerateSignInSheetServiceBus(context, eventItem, message);
```

11. 在 **GenerateSignInSheetServiceBus** 方法开始, 左括号后, 用连接字符串创建 **QueueClient** 的实例 , 代码如下:

```
QueueClient client = QueueClient.CreateFromConnectionString(serviceBusConnectionString,
signInQueueName);
```

12. 在以上代码后面, 用 **QueueMessage** 作参数创建 **BrokeredMessage** 类的实例, 代码如下:

```
BrokeredMessage queueMessage = new BrokeredMessage(message);
```

13. 用 **BrokeredMessage** 作为参数, 调用 **QueueClient** 变量的 **Send** 方法:

```
client.Send(queueMessage);
```

► **任务 3: 调试并验证应用程序**

1. 在调试菜单, 单击启动调试。

 **注释:** 如果弹出保存只读文件对话框, 单击覆盖按钮。

2. 在通知区域, 单击箭头来查看运行的应用程序。
3. 找到 **IIS Express** 图标, 然后右键单击此图标。

4. 在 **View Sites** 下面找到 **Contoso.Events.Management** 选项并单击，然后单击 **Browse Applications** 下面的 URL。
5. 在 **View Sites** 下面找到 **Contoso.Events.Cloud** 选项并单击，然后单击 **Browse Applications** 下面的 URL。
6. 在 **Administration** Web 应用程序主页，单击 **Events** 按钮到 events 列表。
7. 单击任意一个 event 的 **Sign-In Sheet** 按钮。
8. 在 sign-in 页面，有以下信息提示 sign-in sheet 正在被创建：**Sign-In Document Generation in Progress**。
9. 等待一分钟，辅助角色会接收到队列信息。
10. 确保应用程序的执行停在断点处。
11. 按 F5 继续执行。
12. 等待一分钟，然后刷新 sign-in sheet 页面。
13. 单击 **Sign-In Sheet** 从服务器下载 sign-in sheet。
14. 关闭 **Internet Explorer** 程序。
15. 关闭 **Contoso.Events – Microsoft Visual Studio** 窗口。

**结果：**完成此练习后，您将创建并使用了 Service Bus 队列中的消息。

## 练习 4：使用 Service Bus 中继连接 WCF 服务和客户端

### ► 任务 1：使用控制台应用程序测试现有 WCF 服务

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏，单击 **File Explorer** 图标。
3. 浏览到 **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在 Start 屏幕，单击 **Desktop** 磁贴。
5. 在任务栏，单击 File Explorer 图标。
6. 浏览到 **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events.Lodging.Client**，然后双击 **Contoso.Events.Lodging.Client.sln**。
7. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
8. 在解决方案资源管理器窗格，展开 **On-Premise** 文件夹，右键单击 **Contoso.Events.Lodging** 项目，然后单击设为启动项目。
9. 在调试菜单，单击启动调试。
10. 你会看到一个弹出框，提示 WCF 服务没有元数据，询问是否退出调试。单击 **No**。
11. 最小化 **WCF Test Client**。
12. 切换到 **Contoso.Events.Lodging.Client – Microsoft Visual Studio** 窗口。
13. 在调试菜单，单击启动调试。
14. 确认列出四个旅馆名字。
15. 在控制台窗口按任意键关闭窗口。
16. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
17. 在调试菜单，单击停止调试。

### ► 任务 2：修改 WCF 服务端点以使用 Service Bus 中继

1. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
2. 在解决方案资源管理器窗格，展开 **On-Premise** 文件夹。
3. 在解决方案资源管理器窗格，展开 **Contoso.Events.Lodging** 项目。
4. 双击 **App.config** 文件。
5. 定位到 **system.serviceModel** XML 元素。
6. 定位到 **services** XML 子元素。
7. 定位到第一个 **contract** 值为 **Contoso.Events.Models.ILodgingService** 的 **endpoint** XML 子元素。
8. 替换 **binding** 属性的值为 **netTcpRelayBinding**。
9. 定位到第一个 **contract** 值为 **Contoso.Events.Models.ILodgingService** 的 **endpoint** XML 子元素。
10. 替换 **address** 属性的值为 **sb://**。
11. 在 **address** 属性的值之后追加 **Service Bus** 命名空间的名称。命名空间格式类似于 **sb28532[自定义名称]**。
12. 在 **address** 属性的值之后追加 **.servicebus.chinacloudapi.cn/lodging**。
13. 定位到 **system.serviceModel** XML 元素。
14. 定位到 **behaviors** XML 子元素。

15. 定位到 **endpointBehaviors** XML 子元素。
16. 定位到第一个 **name** 值为 **ServiceBusRelayBehavior** 的 **behaviorXML** 子元素，如下所示：

```
<behavior name="ServiceBusRelayBehavior">
 </behavior>
```

17. 在 **behavior** 元素中，加上一个 **transportClientEndpointBehavior** 子元素，如下所示：

```
<transportClientEndpointBehavior>
 </transportClientEndpointBehavior>
```

18. 在 **transportClientEndpointBehavior** 元素中，加上一个 **tokenProvider** 子元素，如下所示：

```
<tokenProvider>
 </tokenProvider>
```

19. 在 **tokenProvider** 元素中，加上一个 **sharedAccessSignature** 子元素。

```
<sharedAccessSignature keyName="[KeyName]" key="[Key]" />
```

20. 在 **sharedAccessSignature** 元素中，删除 **keyName** 属性的值，替换为之前你记录的 Service Bus 连接字符串的 **SharedAccessKeyName** 值。
21. 在 **sharedAccessSignature** 元素中，删除 **key** 属性的值，替换为之前记录的 Service Bus 连接字符串的 **SharedAccessKey** 值。

 **注释：**例如你的连接字符串可能是：

Endpoint=sb://contoso.servicebus.chinacloudapi.cn;  
SharedAccessKeyName=RootManageSharedAccessKey;  
SharedAccessKey=ABC123/J3y+tk=  
你的 **keyName** 值为“RootManageSharedAccessKey”，你的 **key** 值为  
“ABC123/J3y+tk=”。

22. 在解决方案资源管理器窗格，展开 **On-Premise** 文件夹，右键单击 **Contoso.Events.Lodging** 项目，单击设为启动项目。
23. 在调试菜单，单击启动调试。
24. 你会看到一个弹出框，提示 WCF 服务没有元数据，询问是否退出调试。单击 **No**。
25. 最小化但是不要关闭 **WCF Test Client** 应用程序。

#### ► 任务 3：使用控制台应用程序测试中继的 WCF 服务

1. 切换到 **Contoso.Events.Lodging.Client – Microsoft Visual Studio** 窗口。
2. 在解决方案资源管理器窗格，展开 **Contoso.Events.Lodging.Client** 项目。
3. 双击 **Program.cs** 文件。
4. 定位到此方法：

```
private static IEnumerable<Hotel> GetHotels()
```

5. 删除使用静态方法去创建 Uri 的代码，如下显示：

```
Uri serviceUri = new Uri("net.tcp://localhost:8000/lodging", UriKind.Absolute);
```

6. 用以下代码替换删除的代码，将协议、命名空间、名字作为参数，通过调用静态方法 **ServiceBusEnvironment.CreateServiceUri** 创建 Uri，如下所示：

```
Uri serviceUri = new Uri("sb://[namespace].servicebus.chinacloudapi.cn/lodging");
```

7. 将上一步代码中 **CreateServiceUri** 方法的参数 “[namespace]” 替换为 Service Bus 命名空间。

8. 删除以下代码：

```
new NetTcpBinding(),
```

9. 替换为创建一个 **NetTcpRelayBinding** 类的实例的代码：

```
new NetTcpRelayBinding(),
```

10. 在创建 **ChannelFactory<ILodgingService>** 实例的下一行，创建一个 **TransportClientEndpointBehavior** 的实例：

```
TransportClientEndpointBehavior endpointBehavior = new TransportClientEndpointBehavior();
```

11. 在下一行，调用静态方法 **TokenProvider.CreateSharedSecretTokenProvider**，返回结果赋值给变量 **TransportClientEndpointBehavior** 的 **TokenProvider** 属性：

```
endpointBehavior.TokenProvider =
 TokenProvider.CreateSharedAccessSignatureTokenProvider("[keyName]", "[key]");
```

12. 更改 **CreateSharedSecretTokenProvider** 方法的参数，用 Service Bus 连接字符串的 SharedAccessKey 值替换 “[key]” 字符串。

13. 更改 **CreateSharedSecretTokenProvider** 方法的参数，用 Service Bus 连接字符串的 SharedAccessKeyName 值替换 “[keyName]” 字符串。

14. 添加 **TransportClientEndpointBehavior** 的实例到 cf.Endpoint 变量的 EndpointBehaviors 集合属性：

```
cf.Endpoint.EndpointBehaviors.Add(endpointBehavior);
```

15. 在调试菜单，单击启动调试。

16. 确认旅馆列表显示。

17. 在控制台窗口按任意键关闭窗口。

#### ► 任务 4：更新云 Web 应用程序以连接到中继的 WCF 服务

1. 定位到 **Contoso.Events.Lodging.Client** 项目的 **Program.cs** 文件。

2. 定位到此方法：

```
private static IEnumerable<Hotel> GetHotels()
```

3. 选中整个方法。

4. 右键单击选中的内容，单击 **复制** 来拷贝整个方法。

5. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。

6. 在调试菜单，单击 **停止调试**。

7. 在解决方案资源管理器窗格，展开 **Shared** 文件夹。

8. 在解决方案资源管理器窗格，展开 **Contoso.Events.ViewModels** 项目。

9. 双击 **HotelsViewModel.cs** 文件。

10. 定位到以下方法：

```
private static IEnumerable<Hotel> GetHotels()
```

11. 选中整个方法。

12. 右键单击选中的内容，单击**粘贴**来粘贴整个方法。

#### ► 任务 5：调试云 Web 应用程序

1. 在解决方案资源管理器窗格，右键单击解决方案 **Contoso.Events**，然后单击**属性**。
2. 在解决方案'**Contoso.Events**'的属性页对话框中，执行以下步骤：
  - a. 在左边的导航栏，选中**通用属性=>启动项目**。
  - b. 选择**多启动项目**。
  - c. 对**Contoso.Events.Cloud**，设置操作为**开始执行（不调试）**。
  - d. 对**Contoso.Events.Lodging**，设置操作为**开始执行（不调试）**。
  - e. 对**Contoso.Events.Management**，设置操作为**无**。
  - f. 确保所有其他项目的操作设置为**无**。
  - g. 单击**OK**。
3. 在 Start 屏幕，输入 **Azure Storage Emulator**。
4. 单击 **Windows Azure Storage Emulator – v3.4** 磁贴。
5. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
6. 在解决方案资源管理器窗格，右键单击 **Contoso.Events.Data.Generation** 项目，单击**调试->启动新实例**。
7. 等到调试进程结束，控制台窗口关闭。
8. 关闭 **Contoso.Events – Microsoft Visual Studio** 程序。
9. 在 Start 屏幕，找到 **Visual Studio 2013** 磁贴，右键单击，选择 **Run as Administrator**。



**注释：**在 Start 屏幕，你可能需要单击向下箭头来找到 Visual Studio 2013 磁贴。在你用管理员启动了 Visual Studio 2013 之后，你可能会看到 UAC 弹出框，选择**是继续**。

10. 在文件菜单，单击**打开->项目/解决方案**。

11. 在**打开项目**对话框，执行以下步骤：

- a. 定位到 **Allfiles(F):\Mod09\Labfiles\Starter\Relay\Contoso.Events**。
- b. 单击 **Contoso.Events** 解决方案文件。
- c. 单击 **Open**。

12. 在**调试**菜单，单击**开始执行（不调试）**。

13. 你会看到一个弹出框，提示 WCF 服务没有元数据，询问是否退出调试。单击 **No**。

14. 最小化但是不要关闭 **WCF Test Client** 程序。

15. 在 Web 应用程序的主页上，单击任一 event。

16. 在 event details 页面，单击 **Register Now**。

17. 填写任意值，完成注册表单。

18. 单击 **Submit**。
19. **Registration Confirmation** 页面显示出旅馆列表。
20. 关闭 **Internet Explorer** 应用程序。
21. 关闭 **Microsoft Visual Studio** 应用程序。
22. 关闭 **Windows Azure Storage Emulator** 命令窗口。

**结果:** 完成此练习后, 您将使用 Service Bus 中继将现有 WCF 应用程序连接到了客户端。

## 第 10 章：管理 Azure 中的基础结构

# 实验 A：管理虚拟网络中的多个虚拟机

### 练习 1：配置现有虚拟网络

#### ► 任务 1：登录到 Azure 管理门户

1. 打开 **Internet Explorer** 应用程序。
2. 访问 <https://manage.windowsazure.com>。
3. 在电子邮件地址框内，输入 Microsoft 帐户的电子邮件地址。
4. 单击 **Continue**。
5. 在密码框内，输入 Microsoft 帐户的密码。
6. 单击 **Sign In**。

#### ► 任务 2：配置点到站点连接

1. 单击右上角的帐户名前的地球图标。
2. 单击中文（简体）选项。
3. 在屏幕左侧的导航栏里的单击**网络**。
4. 在**虚拟网络**列表中，单击后缀为 Dev28532 的虚拟网络。
5. 单击**配置**选项卡。
6. 在**点到站点连接**下，选择**配置点到站点连接**。
7. 在**地址空间**列表下配置以下值：
  - a. 在**起始 IP** 中输入 172.16.0.0。
  - b. 从**CIDR（地址数）** 框，选择/29 (6)。
8. 单击**虚拟网络访问空间**下的**添加网关子网**。
9. 在页面的底部单击**保存**。
10. 此时出现一个确认对话框，警告可能出现连接中断，单击**是**按钮继续。



**注释：**更新虚拟网络配置可能需要大约 15 分钟。间歇性的刷新浏览器来检查更新状态。

#### ► 任务 3：创建网关

1. 单击**仪表板**选项卡。
2. 单击页面底部的**创建网关**。
3. 一个确认对话框将会确认是否要创建虚拟网络的网关，单击按钮**是**继续。



**注释：**网关的创建需要大约 15 分钟间歇性的刷新浏览器来检查新网关的状态。当仪表盘显示出了**网关 IP 地址**，网关就被创建完成了。等待网关创建的同时，也可以选择继续实验中的剩余部分。

## 练习 2：创建数据库虚拟机

### ► 任务 1：登录到 Azure 预览门户

1. 打开 **Internet Explorer** 应用程序。
2. 访问 <https://portal.azure.com>。
3. 在电子邮件地址框内，输入 Microsoft 帐户的电子邮件地址。
4. 单击 **Continue**。
5. 在密码框内，输入 Microsoft 帐户的密码。
6. 单击 **Sign In**。

### ► 任务 2：创建 SQL Server 2014 Standard 虚拟机

1. 单击右上角的帐户名。
2. 单击 **Settings**。
3. 单击 **LANGUAGE**，选择中文（简体），然后单击 **Select**。
4. 在屏幕的左下角，单击新建。
5. 在新建边栏选项卡，单击计算，单击底部的 **Azure Marketplace**。
6. 在 Marketplace 边栏选项卡里，单击虚拟机选项。
7. 在虚拟机边栏选项卡里面，找到数据库服务器栏，单击 **SQL Server** 选项。
8. 在显示出的 SQL Server 边栏选项卡中，单击 SQL Server 2014 Standard on Windows Server 2012 R2 选项。
9. 在 SQL Server 2014 Standard on Windows Server 2012 R2 边栏选项卡中，单击 **创建** 按钮。
10. 在显示出的创建虚拟机边栏选项卡，执行以下步骤：
  - a. 在主机名框内，输入 db28532（自定义名字）。
  - b. 在用户名框内输入 **testuser**。
  - c. 在密码框内，输入 **TestPa\$\$w0rd**。
  - d. 单击定价层。
  - e. 在选择你的定价层边栏选项卡里，单击 **查看所有**。
  - f. 选择 **A2 标准**。
  - g. 单击 **选择**。
  - h. 单击 **可选配置**。
  - i. 在可选配置边栏选项卡里，单击 **网络**。
  - j. 在网络边栏选项卡里，单击 **虚拟网络**。
  - k. 在虚拟网络边栏选项卡里，选择网络 **Dev28532**。
  - l. 在网络边栏选项卡里，单击 **确认**。
  - m. 在可选配置边栏选项卡里，单击 **存储帐户**。
  - n. 在存储帐户边栏选项卡里，选择存储帐户 **stor28532[自定义名字]**。
  - o. 在可选配置边栏选项卡里，单击 **确认**。
  - p. 单击 **创建**，从而以指定的配置创建新的虚拟机。

11. 在启动板选择新创建的虚拟机。
12. 单击**设置**磁贴。
13. 在显示的**设置**边栏选项卡，单击**属性**选项。
14. 在显示的**属性**边栏选项卡，找到并记录**虚拟 IP 地址**值。

► **任务 3：连接到新的数据库虚拟机**

1. 单击**db28532[自定义名字]**边栏选项卡。
2. 在**db28532[自定义名字]**边栏选项卡的上侧单击**连接**。
3. 在 Internet Explorer 下载对话框里，单击**Open**。
4. 在 Remote Desktop Connection 对话框里，执行以下步骤：
  - a. 单击**Don't ask me again for connections to this computer** 防止这个对话框再次出现。
  - b. 单击**Connect**。
5. 在**Windows Security** 框内输入以下步骤：
  - a. 在**User name** 框内，输入**testuser**。
  - b. 在**Password** 框内，输入**TestPa\$\$w0rd**。
  - c. 单击**OK**。
6. 在**Remote Desktop Connection** 对话框内，执行以下步骤：
  - a. 验证 remote Certificate name 与您的虚拟机的名字相同。
  - b. 单击**Don't ask me again for connections to this computer** 防止这个对话框再次出现。
  - c. 单击**Yes**。
7. 当被提示允许您的网络连接到发现的外部设备，单击**No**。

► **任务 4：为 SQL Server 添加一条规则到 Windows 防火墙**

1. 在开始屏幕，单击左下角的向下箭头。
2. 找到并单击应用程序**Run**。
3. 在**Run** 对话框里执行以下步骤：
  - a. 在**Open** 框内，输入**WF.msc**。
  - b. 单击**OK** 来打开 Windows Firewall。
4. 在 Windows Firewall with Advanced Security 窗口里执行以下步骤：
  - a. 右键单击**Inbound Rules**，然后单击**New Rule**。
  - b. 单击**Port**。
  - c. 单击**Next**。
  - d. 单击**TCP**。
  - e. 单击**Specific local ports**。
  - f. 在**Specific local ports:** 框内，输入**1433**。
  - g. 单击**Next**。
  - h. 单击**Allow the connection**。
  - i. 单击**Next**。

- j. 确保 **Domain**, **Private**, 和 **Public** 复选框都被选中了。
  - k. 单击 **Next**。
  - l. 在 **Name** 框内输入 **SQL Inbound**。
  - m. 单击 **Finish**。
5. 关闭 Windows Firewall with Advanced Security 窗口。
- 任务 5：启用 SQL Server 混合模式身份验证
1. 在开始屏幕，单击左下角的向下箭头。
  2. 找到并单击 **SQL Server 2014 Management Studio** 磁贴。
  3. 在 **Connect to Server** 对话框内，完成以下步骤：
    - a. 在 **Server name** 框内输入。(**句号**)。
    - b. 单击 **Connect**。
  4. 右键单击 **Object Explorer** 窗格顶端的 **SQL Server** 节点，然后单击 **Properties**。
  5. 在 **Server Properties** 对话框里执行以下步骤：
    - a. 单击 **Security** 页面。
    - b. 在 **Server authentication** 节内，选择 **SQL Server and Windows Authentication mode**。
    - c. 单击 **OK**。
    - d. 在 **Microsoft SQL Server Management Studio** 对话框内，单击 **OK**。
  6. 右键单击 **Object Explorer** 窗格顶部的 **SQL Server** 节点然后单击 **Restart**。
  7. 在 **Microsoft SQL Server Management Studio** 对话框内，单击 **Yes**。
  8. 右键单击 **Object Explorer** 窗格里的 **Security** 节点，指向 **New**，然后单击 **Login**。
  9. 在 **Login – New** 对话框里执行以下步骤：
    - a. 在 **Login name** 框内，输入 **dbuser**。
    - b. 单击 **SQL Server authentication**。
    - c. 在 **Password** 框内，输入 **TestPa\$\$w0rd**。
    - d. 在 **Confirm Password** 框内，输入 **TestPa\$\$w0rd**。
    - e. 确保 **Enforce password policy** 复选框没有被选中。
    - f. 确保 **Enforce password expiration** 复选框没有被选中。
    - g. 确保 **User must change password at next login** 复选框没有被选中。
    - h. 单击 **Server Roles** 页面。
    - i. 确保 **public** 服务器角色复选框被选中了。
    - j. 确保 **sysadmin** 服务器角色复选框被选中了。
    - k. 单击 **OK** 来创建新的登录方案。
  10. 右键单击 **Object Explorer** 顶端的 **Databases** 节点，然后单击 **New Database**。
  11. 在 **New Database** 对话框内执行以下步骤：
    - a. 在 **Database name** 框内，输入 **Contoso.Test**。
    - b. 单击 **OK** 来创建新的数据库。

12. 关闭 **Microsoft SQL Server Management Studio** 窗口。
13. 关闭 **Remote Desktop Connection** 应用。

**结果:** 完成此练习后, 您将获得一个装有 SQL Server 2014 的新虚拟机, 在由外部虚拟机访问时, 该 SQL Server 可使用混合模式身份验证。

### 练习 3：创建 Azure 网站服务实例

#### ► 任务 1：创建网站服务实例

1. 在门户页面的左下角，单击**新建**。
2. 在**新建**边栏选项卡中，单击**Web + Mobile**，然后单击**Website**。
3. 在显示出的**Website** 边栏选项卡中执行下列步骤：
  - a. 在**URL** 框内，为网站创建一个独有的名字。
  - b. 单击**定价层**。
  - c. 在显示出的**定价层**边栏选项卡中单击**S1 标准**，单击**选择**。
  - d. 在**新建 WEB 宿主计划**下的**名称**框内，输入**28532**。
  - e. 单击**OK**。
  - f. 在**Website** 边栏选项卡中，单击**创建**。
4. 当网站实例被创建后，边栏选项卡就会出现。
5. 向下滚动然后单击**虚拟网络**磁贴。
6. 在显示出的**虚拟网络**边栏选项卡中执行以下步骤：
  - a. 在**使用现有虚拟网络**下，选择**Dev28532**。
7. 在边栏选项卡上方，单击**浏览**。
8. 记录下新打开的 Internet Explorer 中 **Microsoft Azure Website** 选项卡的 **URL**。
9. 切换到 **Microsoft Azure** 选项卡。
10. 在边栏选项卡的顶端，单击省略号按钮。
11. 在边栏选项卡的顶端，单击**获取发布配置文件**。
12. 在下载对话框中，单击**Save** 按钮右侧的箭头，然后单击**Save As**。
13. 在**Save As** 对话框中，前往 **Allfiles (F):\Mod10\Labfiles**，然后单击**Save**。

#### ► 任务 2：部署 Contoso.Events 数据库测试 Web 应用程序

1. 在开始屏幕上，单击**Desktop** 磁贴。
2. 在任务栏上，单击**文件资源管理器**图标。
3. 在 This PC 窗口，前往 **Allfiles (F):\Mod10\Labfiles\Starter\Contoso.Events**，然后双击**Contoso.Events.sln**。
4. 在**Contoso.Events - Microsoft Visual Studio** 窗口下的**解决方案资源管理器**中，右键单击**Contoso.Events.Web**，然后单击**发布**。
5. 在发布网页窗口里，单击**导入**。
6. 在**导入发布设置**对话框中，单击**浏览**。
7. 在库窗口中，前往 **Allfiles (F):\Mod10\Labfiles**，然后双击之前保存的发布配置文件。
8. 单击**确定**。
9. 验证**站点名称**与您的网站名称相对应。
10. 单击**发布**。

 **注释:** 如果弹出保存只读文件的消息框, 单击**覆盖**。

**结果:** 完成此练习后, 您将在虚拟网络中创建好一个网站服务, 并将 Web 应用程序部署到该网站。

## 练习 4：将测试应用程序连接到 SQL Server 虚拟机

### ► 任务 1：检索 SQL Server 虚拟机的内部 IP 地址

1. 切换到 Internet Explorer。
2. 在左侧的导航栏中，单击浏览。
3. 在浏览边栏选项卡中，单击虚拟机。
4. 在虚拟机边栏选项卡中，选择名为 **db28532[自定义名字]** 的虚拟机的那一行。
5. 在出现的 **db28532[自定义名字]** 边栏选项卡中，单击设置磁贴。
6. 在出现的设置边栏选项卡中，单击属性选项。
7. 在出现的属性边栏选项卡中，记录下专用 IP 地址的值。
8. 通过单击边栏选项卡右上方的关闭按钮(X)来关闭属性选项卡。

### ► 任务 2：在本地调试 Contoso.Events 数据库测试 Web 应用程序

1. 切换到 Visual Studio 2013。
2. 在解决方案资源管理器中，右键单击项目 **Contoso.Events.Web** 项目，然后单击设为启动项目。
3. 在调试菜单中，单击启动调试。

 **注释：**此 Web 应用程序引用了多个 NuGet 包。调试解决方案时，Visual Studio 开始构建解决方案。这触发了 NuGet 自动下载安装缺失包的操作。



#### NuGet Package Restore

<http://go.microsoft.com/fwlink/?LinkId=510175>

4. 在 Web 应用程序的主页里，**IP Address** 框里，输入之前记录下的运行 SQL Server 的虚拟机的专用 IP 地址。
5. 单击 **Verify**。
6. 验证 **Events** 页面显示出一个事件列表。
7. 关闭 Internet Explorer。

### ► 任务 3：在 Azure 中调试 Contoso.Events 数据库测试 Web 应用程序

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 导航到你先前在这个实验中创建的 Azure 网站实例的 URL。
3. 在 Web 应用程序主页，**IP Address** 框内，输入之前记录下的运行 SQL Server 的虚拟机的专用 IP 地址。
4. 单击 **Verify**。
5. 验证 **Events** 页面显示出一个事件列表。
6. 关闭 **Visual Studio** 应用程序。
7. 关闭 **Internet Explorer** 应用程序。

**结果：**完成此练习后，您将使用虚拟网络中数据库虚拟机的内部 IP 地址连接到了 SQL Server 2014。

# 实验 B：使用中国版 Windows Azure 管理虚拟网络中的多个虚拟机

## 练习 1：配置现有虚拟网络

### ► 任务 1：登录到 Azure 管理门户

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 访问 <https://manage.windowsazure.cn>。
3. 输入您的 Azure 帐户的邮箱地址和密码，单击 **Sign In** 按钮。
4. 如果您的界面语言不是中文，可以单击页面右上角的**地球**图标，并在菜单中选择**中文(简体)**语言。

### ► 任务 2：配置点到站点的连接

1. 在页面左侧的导航栏中，单击**网络**。
2. 在**网络**列表中，单击名称为**Dev28532** 的虚拟网络。
3. 单击**配置**选项卡。
4. 在**点到站点连接**下，选择**配置点到站点连接**。
5. 在**地址空间**下配置以下值：
  - a. 在**起始 IP** 中输入 **172.16.0.0**。
  - b. 在**CIDR (地址数)** 框，选择**/29 (6)**。
6. 单击**虚拟网络访问空间**下的**添加网关子网**。
7. 在页面的底部单击**保存**。
8. 此时出现一个确认对话框，警告可能出现连接中断。单击**是**按钮继续。



**注释：**更新虚拟网络配置需要大约 15 分钟。周期性的刷新浏览器来检查更新状态。

### ► 任务 3：创建网关

1. 单击**仪表板**选项卡。
2. 单击页面底部的**创建网关**。
3. 一个确认对话框将会确认是否要创建虚拟网络的网关，单击按钮**是**。



**注释：**网关的创建需要大约 15 分钟。周期性的刷新浏览器来检查新网关的状态。当仪表板显示出了**网关 IP 地址**，网关就被创建完成了。等待网关创建的同时，也可以选择试验中的剩余部分进行练习。

## 练习 2：创建数据库虚拟机

### ► 任务 1：创建 SQL Server 2014 Standard 虚拟机

1. 在屏幕的左下角，单击新建。
2. 在新建边栏选项卡，依次单击计算->虚拟机->快速创建。
3. 在 DNS 名称中填入 db28532[自定义名称]。
4. 单击映像下拉列表，单击更多映像。
5. 显示出的创建虚拟机导航选项卡，完成以下步骤：
  - a. 单击 SQL Server。
  - b. 在中间选择 SQL Server 2014 RTM Standard。（系统为 Windows Server 2012 R2）
  - c. 单击下一步。
  - d. 在虚拟机名称中输入 db28532[自定义名称]。
  - e. 在大小下拉框中，选择 A2(双核，3.5GB 内存)。
  - f. 在新用户名框内输入 testuser。
  - g. 在新密码框和确认框内，输入 TestPa\$\$w0rd。
  - h. 单击下一步。
  - i. 在云服务中选择创建新云服务。
  - j. 在云服务 DNS 名称，验证名称可用，如果不可请修改为一个可用的名称。
  - k. 在区域/地缘组/虚拟网络中选择 Dev28532。
  - l. 存储帐户选择使用自动生成的帐户。
  - m. 单击下一步。
  - n. 单击完成。



**注释：**Azure 试用帐户有资源内核数限制，如果遇到内核不足导致虚拟机创建失败，请在步骤 e 中将虚拟机的大小修改为 A1(单核，1.75GB 内存)或 A0(共享内核，768MB 内存)。

6. 等待虚拟机及扩展创建完成。

如果你想通过公网访问到数据库服务器完成步骤 7~13。

7. 在管理网站中找到刚创建的虚拟机，单击该虚拟机的名字，单击仪表板，找到右侧公用虚拟 IP(vip)地址并记录它。
8. 单击端点。
9. 单击页面下方的添加。
10. 在弹出的添加端点页面中单击下一步。
11. 在名称选择 MSSQL。
12. 确认协议为 TCP，公用端口和私有端口为 1433。
13. 单击完成。

### ► 任务 2：连接到新的数据库虚拟机

1. 在管理网站上找到虚拟机 db28532[自定义名称]。

MCT USE ONLY. STUDENT USE PROHIBITED

2. 单击选择该虚拟机。
  3. 在页面底部单击**连接**。
  4. 在**Internet Explorer 下载**对话框里，单击**Open**。
  5. 在**Remote Desktop Connection**对话框里，完成以下步骤：
    - a. 为了不让这个对话框再次出现，单击**Don't ask me again for connections to this computer**。
    - b. 单击**Connect**。
  6. 在**Windows Security**对话框按以下步骤操作：
    - a. 在**User name**框内，输入**testuser**。
    - b. 在**Password**框内，输入**TestPa\$\$w0rd**。
    - c. 单击**OK**。
  7. 在**Remote Desktop Connection**对话框内，完成以下步骤：
    - a. 验证**Certificate name**与你虚拟机的名字相同。
    - b. 为了防止这个对话框再次出现，单击**Don't ask me again for connections to this computer**。
    - c. 单击**Yes**。
  8. 当你被提议允许你的网络连接用于发现外部设备时，单击**No**。
- **任务 3：为 SQL Server 添加一条规则到 Windows 防火墙**
1. 在 Start 屏幕，单击左下角的向下箭头。
  2. 找到并单击应用程序**Run**。
  3. 在**Run**对话框里完成以下步骤：
    - a. 在**Open**框内，输入**WF.msc**。
    - b. 单击**OK**来打开**Windows Firewall**。
  4. 在**Windows Firewall with Advanced Security**窗口里完成以下步骤：
    - a. 单击选中**Inbound Rules**，再右键单击**Inbound Rules**，然后单击**New Rule**。
    - b. 选择**Port**。
    - c. 单击**Next**。
    - d. 选择**TCP**。
    - e. 选择**Specific local ports**。
    - f. 在**Specific local ports**框内，输入**1433**。
    - g. 单击**Next**。
    - h. 选择**Allow the connection**。
    - i. 单击**Next**。
    - j. 确保**Domain**, **Private** 和 **Public**复选框都被选中了。
    - k. 单击**Next**。
    - l. 在**Name**框内输入**SQL Inbound**。
    - m. 单击**Finish**。
  5. 关闭**Windows Firewall with Advanced Security**窗口。

► 任务 4：启用 SQL Server 混合模式身份验证

1. 在 Start 屏幕，单击左下角的向下箭头。
2. 找到并单击 **SQL Server 2014 Management Studio** 磁贴。
3. 在 **Connect to Server** 对话框内，完成以下步骤：
  - a. 在 **Server name** 框内，输入.（输入一个点）。
  - b. 单击 **Connect**。
4. 右键单击 **Object Explorer** 栏顶端的 **SQL Server** 节点，然后单击 **Properties**。
5. 在 **Server Properties** 对话框里完成以下步骤：
  - a. 单击 **Security**。
  - b. 在 **Server authentication** 节内，选择 **SQL Server and Windows Authentication mode**。
  - c. 单击 **OK**。
  - d. 在 **Microsoft SQL Server Management Studio** 对话框内，单击 **OK**。
6. 右键单击 **Object Explorer** 顶部的 **SQL Server** 节点然后单击 **Restart**。
7. 在 **Microsoft SQL Server Management Studio** 对话框内，单击 **Yes**。
8. 右键单击 **Object Explorer** 栏里的 **Security** 节点，指向 **New**，然后单击 **Login**。
9. 在 **Login-New** 对话框里完成以下步骤：
  - a. 在 **Login name** 框内，输入 **dbuser**。
  - b. 单击 **SQL Server authentication**。
  - c. 在 **Password** 框内，输入 **TestPa\$\$w0rd**。
  - d. 在 **Confirm Password** 框内，输入 **TestPa\$\$w0rd**。
  - e. 确保 **Enforce password policy** 复选框没有被选中。
  - f. 确保 **Enforce password expiration** 复选框没有被选中。
  - g. 确保 **User must change password at next login** 复选框没有被选中。
  - h. 单击 **Server Roles** 页面。
  - i. 确保 **public** 服务器角色复选框被选中。
  - j. 确保 **sysadmin** 服务器角色复选框被选中。
  - k. 单击 **OK** 来创建新的登录方案。
10. 右键单击 **Object Explorer** 顶端的 **Databases** 节点，然后单击 **New Database**。
11. 在 **New Database** 对话框内完成以下步骤：
  - a. 在 **Database name** 框内，输入 **Contoso.Test**。
  - b. 单击 **OK** 来创建新的数据库。
12. 关闭 **Microsoft SQL Server Management Studio** 窗口。
13. 关闭 **Remote Desktop Connection** 应用。

**结果：**完成此练习后，您将获得一个装有 SQL Server 2014 的新虚拟机，在由外部虚拟机访问时，该 SQL Server 可使用混合模式身份验证。

### 练习 3：将测试应用程序连接到 SQL Server 虚拟机

#### ► 任务 1：检索 SQL Server 虚拟机的内部 IP 地址

1. 切换到 Internet Explorer。
2. 在管理网站左边，单击虚拟机。
3. 在出现的虚拟机列表中，单击名为 **db28532** 的虚拟机。
4. 单击仪表板。
5. 在右侧找到内部 IP 地址并记录下来。

#### ► 任务 2：在本地调试 Contoso.Events 数据库测试 Web 应用程序

1. 在 Start 屏幕，单击 Visual Studio 2013 磁贴。
2. 选择文件->打开->项目/解决方案。
3. 在打开项目对话框中，定位到 F:\Mod10\Labfiles\Starter\Contoso.Events，选择 **Contoso.Events.sln**，单击 Open。
4. 在解决方案资源管理器窗格中，右键单击项目 **Contoso.Events.Web**，然后单击设为启动项目。
5. 在调试菜单中，单击启动调试。

 **注释：**此网站应用程序引用了多个 NuGet 包。调试解决方案时，Visual Studio 开始构建解决方案。这触发了 NuGet 自动下载安装缺失包的操作。

#### NuGet Package Restore

<http://go.microsoft.com/fwlink/?LinkId=510175>

6. 在网页应用的主页里，**IP Address** 框里，输入之前记录下虚拟机的内部 IP 地址。
7. 单击 **Verify**。
8. 验证 **Events** 页面显示出一个 events 列表。
9. 在管理门户中关闭数据库虚拟机 **db28532[自定义名称]**。
10. 关闭 Internet Explorer 应用程序。
11. 关闭 Visual Studio 应用程序。

**结果：**完成此练习后，您将使用虚拟网络中数据库虚拟机的内部 IP 地址连接到了 SQL Server 2014。

## 第 11 章：自动化与 Azure 资源的集成

# 实验 A：使用 PowerShell 自动化测试环境的创建工作

### 练习 1：准备 Azure PowerShell 环境

#### ► 任务 1：打开 Azure PowerShell 控制台

1. 在 Start 屏幕，单击下拉箭头查看全部应用程序，然后单击 **Microsoft Azure PowerShell**。
2. 切换到 **Microsoft Azure PowerShell** 窗口。

#### ► 任务 2：使用发布设置文件导入 Azure 订阅

1. 在控制台输入以下命令并且按下 **Enter** 键，获取发布配置文件。

```
Get-AzurePublishSettingsFile
```

2. 在打开的 **Internet Explorer** 选项卡中，执行以下步骤：

- a. 输入 Microsoft 帐户的 Email 地址。
  - b. 单击 **Continue**。
  - c. 输入 Microsoft 帐户密码。
  - d. 单击 **Sign In**。
  - e. **Save** 按钮右侧的箭头，然后在下载对话框中单击 **Save As** 按钮。
3. 在显示的 **Save As** 对话框中执行以下步骤：
    - a. 指向 **AllFiles (F):\Mod11\Labfiles** 文件夹。
    - b. 在 **File Name** 框中输入 **28532**。
    - c. 在 **Save as type** 框中，确认 **PUBLISHSETTINGS File** 选项被选中。
    - d. 单击 **Save**。
  4. 关闭 Internet Explorer。
  5. 切换到 **Microsoft Azure PowerShell** 应用程序。
  6. 在控制台输入以下命令然后按 Enter 键，导入发布配置文件：

```
Import-AzurePublishSettingsFile F:\Mod11\Labfiles\28532.publishsettings
```

7. 在控制台输入以下命令并按 Enter 键，查看帐户详细信息：

```
Get-AzureAccount
```

 **注释：**如果有多个 Azure 订阅，可以使用 **Select-AzureSubscription** 选择合适的订阅。如果 Azure 订阅是由培训机构提供的，它可能是一个没有详细信息的自动生成的帐户，在这种情况下，执行这个命令返回的将是一个空结果。

**结果：**完成此练习后，您将设置好开发环境，以便使用 Windows PowerShell 来进行 Azure 服务自动化。

## 练习 2：使用 Windows PowerShell 创建和访问网站服务实例

### ► 任务 1：使用 Windows PowerShell 创建新的网站服务实例

- 提供一个唯一的网站名字，输入以下命令，按下 Enter 键，创建一个新的网站。

```
New-AzureWebsite [Unique Name]
```

 **注释：**可能会得到一个错误信息，提示这个网站名字不是唯一的，如果这种情况出现，选择一个新名字直到网站创建成功。

- 在控制台输入以下命令并且按下 Enter 键，查看网站列表：

```
Get-AzureWebsite
```

- 在控制台上输入以下命令，[Unique Name]是之前新创建的名字，按下 Enter 键，获取新网站的信息细节：

```
Get-AzureWebsite -Name [Unique Name]
```

- 记录之前命令生成的键-值对列表中的 **hostname** 值。

- 在控制台输入以下命令并且按下 Enter 键，在 Internet Explorer 中查看新网站：

```
explorer "http://[Host Name]"
```

- 确认新网站正常运行。
- 关闭 Internet Explorer。
- 切换到 **Microsoft Azure PowerShell** 控制台窗口。

### ► 任务 2：使用 Windows PowerShell 移除网站服务实例

- 在控制台输入以下命令按 Enter 键，停止新的网站，[Unique Name]是之前命令中新创建网站的名字：

```
Stop-AzureWebsite -Name [Unique Name]
```

- 在控制台输入以下命令按 Enter 键，查看网站列表：

```
Get-AzureWebsite
```

- 在控制台输入以下命令按下 Enter 键，移除新的网站，[Unique Name]是之前命令中新创建网站的名字：

```
Remove-AzureWebsite -Name [Unique Name]
```

- 输入 **Y** 表示想要移除网站，然后按下 Enter 键。
- 在控制台输入以下命令按下 Enter 键，结果是空的，表明移除了新网站，[Unique Name]是之前命令中新创建网站的名字：

```
Get-AzureWebsite -Name [Unique Name]
```

**结果：**完成此练习后，您就会使用 Windows PowerShell 来管理 Azure 服务的实例。

MCT USE ONLY. STUDENT USE PROHIBITED

### 练习 3：使用资源模板创建多个预配置的资源

► 任务 1：切换到 Azure 资源管理器 PowerShell 模块

- 在控制台输入以下命令按下 Enter 键切换模式：

```
Switch-AzureMode AzureResourceManager
```

► 任务 2：使用 Azure Active Directory 导入 Azure 订阅

- 在控制台输入以下命令按下 Enter 键登录 Azure：

```
Add-AzureAccount
```

- 在 **Sign in to Windows Azure** 对话框中执行以下步骤：

- 输入 Microsoft 帐户 Email 地址。
- 单击 **Continue**。
- 输入 Microsoft 帐户密码。
- 单击 **Sign In**。

- 在控制台输入以下命令按下 Enter 键，查看帐户细节：

```
Get-AzureAccount
```

 **注释：**如果有多个 Azure 订阅，可以使用 **Select-AzureSubscription** 选择合适的订阅。

► 任务 3：使用 Windows PowerShell 创建资源组

- 在控制台输入以下命令并且按下 Enter 键，查看资源组模板的描述：

```
Get-AzureResourceGroupGalleryTemplate -Identity Microsoft.WebSiteSQLDatabase.0.2.0-preview
```

- 在控制台输入以下命令并且按下 Enter 键，创建一个新的资源组模板实例：

```
New-AzureResourceGroup -Name rga28532 -Location "West Us" -GalleryTemplateIdentity Microsoft.WebSiteSQLDatabase.0.2.0-preview
```

- 执行以下步骤，向模板提供参数：

- 对于 **siteName** 提示，输入值 **rs28532[自定义名称]**，然后按下 **Enter** 键。
- 对于 **hostingPlanName** 提示，输入值 **rp28532[自定义名称]**，然后按下 **Enter** 键。
- 对于 **siteLocation** 提示，输入值 **West US**，然后按下 **Enter** 键。
- 对于 **serverName** 提示，输入值 **rv28532[自定义名称]**，然后按下 **Enter** 键。
- 对于 **serverLocation** 提示，输入值 **West US**，然后按下 **Enter** 键。
- 对于 **administratorLogin** 提示，输入值 **testuser**，然后按下 **Enter** 键。
- 对于 **administratorLoginPassword** 提示，输入值 **TestPa\$\$w0rd**，然后按下 **Enter** 键。
- 对于 **databaseName** 提示，输入值 **rd28532[自定义名称]**，然后按下 **Enter** 键。

 **注释：**等待设置进程完成。状态信息将会定期显示，并且当设置完成，最终的信息细节将会全部显示。

4. 提供一个唯一的网站名字输入以下命令，按下 Enter 键，获取一个新的资源组细节：

```
Get-AzureResourceGroup -ResourceGroupName rga28532
```

5. 在控制台里提供网站的唯一名，输入以下命令，按下 Enter 键，获取新网站的详细新信息：

```
Get-AzureResource -Name rs28532[Your Name Here] -ResourceGroupName rga28532 -ResourceType "Microsoft.Web/sites" -ApiVersion 2014-04-01
```

6. 记录上一命令生成的 JSON 数据中 **hostNames** 数组中的那个值。

7. 在控制台输入以下命令并且按下 Enter 键，在 Internet Explorer 中查看新网站：

```
explorer "http://[Host Name]"
```

8. 验证新网站正常运行。

9. 关闭 Internet Explorer。

10. 切换到 **Microsoft Azure PowerShell** 控制窗口。

#### ► 任务 4：使用 Windows PowerShell 移除资源组

1. 在控制台输入以下命令并按 Enter 键，移除新的资源组：

```
Remove-AzureResourceGroup -Name rga28532
```

2. 输入 **Y** 表示想要移除网站，然后按 Enter 键。

 **注释：**因为移除一个资源组涉及到移除很多资源，这个进程平均可能花费 5~10 分钟。

3. 关闭 **Microsoft Azure PowerShell** 控制台窗口。

**结果：**完成此练习后，您就会使用 Azure 来与资源管理器、资源组和资源组模板交互。

# 实验 B：使用 PowerShell 自动化测试环境的创建工作

## 练习 1：准备 Azure PowerShell 环境

### ► 任务 1：打开 Azure PowerShell 控制台

- 在 Start 屏幕，单击下拉箭头查看全部应用程序，然后单击 **Microsoft Azure PowerShell**。
- 打开 **Microsoft Azure PowerShell** 窗口。

### ► 任务 2：使用发布设置文件导入 Azure 订阅

- 在控制台输入以下命令并且按下 **Enter** 键，获取发布配置文件：

```
Get-AzurePublishSettingsFile -environment azurechinacloud
```

- 在打开的 Internet Explorer 选项卡，执行以下步骤：

- 输入用于登录帐号的邮件地址。
- 输入帐户密码。
- 单击 **Sign In**。

- 在显示的 **Save As** 对话框中执行以下步骤：
  - 指向 **AllFiles (F):\\Mod11\\Labfiles** 文件夹。
  - 在 **File Name** 框中输入 **28532**。
  - 在 **Save as type** 框中，确认 **PUBLISHSETTINGS File** 选项被选中。
  - 单击 **Save**。
- 关闭 Internet Explorer。

- 切换到 **Microsoft Azure PowerShell** 应用程序。

- 在控制台输入以下命令然后按 Enter 键导入发布配置文件：

```
Import-AzurePublishSettingsFile F:\\Mod11\\Labfiles\\28532.publishsettings
```

- 在控制台输入以下命令并按 Enter 键查看帐户详细信息：

```
Get-AzureAccount
```

 **注释：**如果有多个 Azure 订阅，可以使用 **Select-AzureSubscription** 选择最合适的订阅。如果 Azure 订阅是由培训机构提供的，它可能是一个没有详细信息的自动生成的帐户，这类情况，执行这个命令返回的将是一个空结果。

**结果：**完成此练习后，您将设置好开发环境，以便使用 Windows PowerShell 来进行 Azure 服务自动化。

## 练习 2：使用 Windows PowerShell 创建和访问网站服务实例

### ► 任务 1：使用 Windows PowerShell 创建新的网站服务实例

- 输入以下命令，将命令中[Unique Name]替换为自定义名称，按下 Enter 键，创建一个新的网站。

```
New-AzureWebsite [Unique Name]
```

 **注释：**可能会得到一个错误信息，提示这个名字不是唯一的，如果这种情况发生，选择一个新名字直到网站创建成功。

- 在控制台输入以下命令并且按下 Enter 键，查看网站列表：

```
Get-AzureWebsite
```

- 在控制台上输入以下命令，[Unique Name]是之前新创建的名字，按下 Enter 键，获取新网站的信息细节：

```
Get-AzureWebsite -Name [Unique Name]
```

- 记录由之前命令生成的键值实例列表中 **hostname** 的值。
- 在控制台输入以下命令并且按下 Enter 键，在 Internet Explorer 上查看新网站：

```
explorer "http://[Host Name]"
```

- 确认新网站正常运行。
- 关闭 Internet Explorer。
- 切换 Microsoft Azure PowerShell 控制台窗口。

### ► 任务 2：使用 Windows PowerShell 移除网站服务实例

- 在控制台输入以下命令按下 Enter 键，停止新的网站，[Unique Name]是之前命令中新创建的名字：

```
Stop-AzureWebsite -Name [Unique Name]
```

- 在控制台输入以下命令查看网站列表：

```
Get-AzureWebsite
```

- 在控制台输入以下命令按下 Enter 键，移除新的网站，[Unique Name]是之前命令中新创建的名字：

```
Remove-AzureWebsite -Name [Unique Name]
```

- 输入 Y 表示想要移除网站，然后按下 Enter 键。
- 在控制台输入以下命令按下 Enter 键，结果是空的，确认移除新的网站，[Unique Name]是之前命令中新创建的名字：

```
Get-AzureWebsite -Name [Unique Name]
```

- 关闭 Microsoft Azure PowerShell 控制台窗口。

**结果：**完成此练习后，您就会使用 Windows PowerShell 来管理 Azure 服务的实例。

## 第 12 章：保护 Azure Web 应用程序

# 实验 A：将 Azure Active Directory 与事件管理门户集成

### 练习 1：创建 Azure AD

#### ► 任务 1：登录到 Azure 管理门户

 **注释：**因为在预览门户网站中没有创建新的 Azure Active Directory 域功能，所以在这个实验中将使用管理门户网站。

1. 在 Start 界面，单击 **Internet Explore** 磁贴。
2. 打开 <https://manage.windowsazure.com>。
3. 输入 Microsoft 帐户电子邮件地址和密码，然后单击 **Sign In**。
4. 单击右上角的帐户名前的地球图标，然后单击**中文（简体）**。

#### ► 任务 2：使用管理门户创建目录

1. 在屏幕的左侧导航页面中，向下滚动并单击 **Active Directory**。
2. 单击页面顶部的**目录**选项卡。
3. 在屏幕的底部，单击**新建**。
4. 选择**应用服务**，**ACTIVE DIRECTORY**，**目录**，然后单击**自定义创建**。
5. 在**添加目录**对话框中执行以下步骤：
  - a. 在**目录**列表中选择**创建新目录**。
  - b. 在**名称**框中输入 **28532**。
  - c. 在**域名**框中输入 **ad28532[自定义名称]**。
  - d. 在**国家或地区**列表中选择**当前所在地**。
  - e. 单击**对勾**按钮添加目录。

#### ► 任务 3：在目录中创建全局管理员角色

1. 在**目录**列表中，单击**目录名 28532**。
2. 在页面顶部单击**用户**选项卡。
3. 在界面底部单击**添加用户**。
4. 在**添加用户**对话框中执行以下步骤：
  - a. 在**用户类型**列表选择**您的组织中的新用户**。
  - b. 在**用户名**框中输入 **admin**。
  - c. 在**域名**框中确认仅有**您的域(ad28532[自定义名称])**被选择。
  - d. 在**导向窗**中单击向右箭头移向下一步。
  - e. 在**名字**框中输入 **Admin**。
  - f. 在**姓氏**框中输入 **User**。

- g. 在**显示名称**框中输入**Admin User**。
- h. 在**角色**框中选择**全局管理员**。
- i. 在**备用电子邮件**框中输入 example@contoso.com。
- j. 确认**启用多重身份验证**复选框被清除。
- k. 在导向窗中单击向右箭头移到下一步。
- l. 在**获取临时密码**界面中单击**创建**来创建用户。
- m. 记录生成用户的临时密码。
- n. 单击对勾按钮完成导向。

#### ► 任务 4：在目录中创建用户角色

1. 在屏幕的底部单击**添加用户**。
2. 在**添加用户**对话框中执行以下步骤：
  - a. 在**用户类型**列表中选择**您的组织中的新用户**。
  - b. 在**用户名**框中输入**standard**。
  - c. 在域列表中，确认仅有您的域名。  
确认仅有您的域名(**ad28532[自定义名称]**)被选择。
  - d. 在导向框中单击向右箭头移向下一步。
  - e. 在**名字**框中输入**Standard**。
  - f. 在**姓氏**框中，输入**User**。
  - g. 在**显示名称**框中输入**Standard User**。
  - h. 在**角色**框中，选择**用户**。
  - i. 确认**启用多重身份验证**复选框被清除。
  - j. 在导向窗中单击向右箭头移向下一步。
  - k. 在**获取临时密码**界面中单击**创建**来创建用户。
  - l. 记录生成用户的临时密码。
  - m. 单击对勾按钮完成导向。

**结果：**完成此练习后，您将创建好 Azure AD，并在该目录中填充了用户。

## 练习 2：保护现有 ASP.NET Web 应用程序

### ► 任务 1：将 **Authorize** 属性添加到 **HomeController** 类

1. 在 Start 屏幕中，单击 **Desktop** 磁贴。
2. 在任务栏单击文件资源管理器图标。
3. 在库窗口，指向 **Allfiles (F):\Mod12\Labfiles\Starter\Contoso.Events** 然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中扩展 **Administration** 文件夹。
5. 展开 **Contoso.Events.Management.Old** 项目。
6. 展开 **Controllers** 文件夹。
7. 双击 **HomeController.cs** 文件。
8. 在文件的顶部定位到类定义：

```
public class HomeController : Controller
```

9. 在定义类的代码行之上添加 **Authorize** 属性。  
[Authorize]
10. 保存 **HomeController.cs** 文件。

 **注释：**如果弹出保存只读文件消息框，单击**覆盖**。

### ► 任务 2：调试 Web 应用程序

1. 在解决方案资源管理器窗格中右键单击 **Contoso.Events.Management.Old** 项目，然后单击设为启动项目。
2. 在调试菜单中，单击启动调试。
3. 等待 Internet Explorer 打开。
4. 确认网站返回一条 **401-未经身份验证** 错误信息。
5. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
6. 在调试菜单中，单击停止调试。

**结果：**完成此练习后，您就会使用 MVC 来确保用户在登录后才能访问控制器或操作。

### 练习 3：将 Azure AD 与 ASP.NET 身份集成

#### ► 任务 1：修复 IIS 8.0 Express 安装

 **注释：**当给您的计算机生成映像并使用新用户重建计算机系统的时候，原先随 Visual Studio 2013 Update 3 安装的 IIS 8.0 Express 可能会崩溃。

IIS Express 依赖于用户文档目录中的配置文件。当机器被制作成映像，原用户帐户会丢失。当从映像创建新虚拟机时，新用户帐户会生成。

修复安装的操作将可以为新用户创建配置文件。

1. 在开始界面，单击向下箭头，然后单击运行磁贴。
2. 在运行对话框中执行以下步骤：
  - a. 在 **Open** 框中输入 **appwiz.cpl**。
  - b. 单击 **OK**。
3. 在 **Programs and Features** 窗口执行以下步骤：
  - a. 在 **Currently installed programs** 列表中定位到 **IIS 8.0 Express**。
  - b. 右键单击 **IIS 8.0 Express**，然后单击 **Repair**。
  - c. 等待修复进程完成，当对话框关闭进程完成。
4. 关闭 **Programs and Features** 窗口。

#### ► 任务 2：使用 Azure AD 作为身份提供程序来创建新的管理 Web 应用程序

1. 在解决方案资源管理器窗格中，右键单击 **Administration** 解决方案文件夹，指向添加，然后单击新建项目。
2. 在添加新项目对话框中执行以下步骤：
  - a. 展开已安装，展开 **Visual C#**，然后单击 **Web**。
  - b. 单击 **ASP.NET Web 应用程序项目类型**。
  - c. 在名称文本框中输入 **Contoso.Events.Management**。
  - d. 确认位置文本框中的值为 **F:\Mod12\Labfiles\Starter\Contoso.Events**。
  - e. 单击确定。
3. 在新建 **ASP.NET 项目 - Contoso.Events.Management** 对话框中执行以下步骤：
  - a. 单击 **MVC** 模版。
  - b. 确认 **Microsoft Azure - 在云中托管** 复选框是为被选中。
  - c. 将其余的字段保留为默认值。
  - d. 单击更改身份验证。
4. 在更改身份验证对话框中选择组织帐户。
  - a. 在域文本框中输入 **ad28532[自定义名称].onmicrosoft.com**。
  - b. 将其余的字段保留为默认值。
  - c. 单击确定。



**注释：**使用该域登录或是使用 Internet Explorer 记住某个登录帐户登录，登录过程中画面会有不同。

- d. 以 **Admin User 帐户(admin@ad28532[自定义名称].onmicrosoft.com)** 登录，临时密码是早先实验中生成的 Admin 密码。
- e. 在**更改密码**对话框中，修改密码为 **Pa\$\$w0rd**。
- f. 单击 **Submit**，在 **Sign in to Azure Active Directory** 对话框中输入新密码，然后单击 **sign in**。
- g. 单击确定。
5. 在新建 **ASP.NET 项目 – Contoso.Events.Management** 对话框中单击确定。
6. 在解决资源管理器窗格，展开 **Administration** 文件夹。
7. 右键单击 **Contoso.Events.Management** 项目然后单击**设为启动项目**。
8. 在解决资源管理器窗格，展开 **Administration** 文件夹。
9. 右键单击 **Contoso.Events.Management** 项目指向添加，然后单击引用。
10. 在引用管理器- **Contoso.Events.Management** 对话框中执行以下步骤：
  - a. 在左侧，展开**解决方案**选项卡，然后单击**项目**选项。
  - b. 双击 **Contoso.Events.Models** 引用。
  - c. 双击 **Contoso.Events.ViewModels** 引用。
  - d. 单击确定。
11. 在解决资源管理器窗格展开 **Administration** 文件夹。
12. 展开 **Contoso.Events.Management.Old** 项目。
13. 复制以下文件夹：
  - **App\_Start**
  - **Content**
  - **Controllers**
  - **Fonts**
  - **Scripts**
  - **Views**



**注释：**按 Shift 键然后单击 **App\_Start** 和 **Views** 文件夹以复制全部文件夹，这会在同一时间选择全部六个文件夹，可以使用快捷菜单或按 Ctrl+C 来复制文件夹。

14. 粘贴全部复制的文件夹到 **Contoso.Events.Management** 项目中。
  15. 当出现合并文件夹的提示，单击**应用于所有项**，然后单击**是**。
  16. 当提示**目标文件已存在**，单击**应用于所有项**，然后单击**是**。
- **任务 3：验证“管理”Web 应用程序是否需要使用组织帐户登录**
1. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Management** 项目，然后单击**设为启动项目**。
  2. 在调试菜单中，单击**启动调试**。

3. 一个 **Microsoft Visual Studio** 对话框弹出表示可以配置为真正的自签名证书，单击**是**。
4. **Security Warning** 对话框中出现表示自签名证书不能确认但是可以继续安装，单击**Yes**。
5. 使用 **Standard User** 帐户登录，临时密码是之前创建 Standard 时生成的密码。
6. 在 **Change Password** 对话框中修改密码为 **Pa\$\$w0rd**。
7. 单击 **Submit**，在 **Sign in to Azure Active Directory** 对话框中输入新密码，然后单击 **sign in**。
8. 查看 **Contoso.Events.Management** web 应用程序主页。
9. 单击 **Go To Events List** 查看事件列表。
10. 关闭 **Internet Explorer** 应用程序。
11. 关闭 **Microsoft Visual Studio** 应用程序。

**结果：**完成此练习后，您就会结合 ASP.NET 身份框架来使用 Azure AD 域。

## 第 13 章：维护和监视 Azure 中的 Web 解决方案

### 实验 A：将 Events Web 应用程序部署到 Azure

#### 练习 1：创建用于部署的目标 Azure 服务

##### ► 任务 1：登录到 Azure 预览门户

1. 在 Start 屏幕，单击 **Internet Explorer**。
2. 转到门户网站 <https://portal.azure.com>。
3. 在 Email 地址框中，键入 Microsoft 帐户的 Email 地址，单击 **Continue**。
4. 在密码框中，键入 Microsoft 帐户的密码。
5. 单击 **Sign In**。
6. 单击右上角帐户，单击 **Settings**，单击 **LANGUAGE**，选择中文（简体），然后单击 **Select**。

##### ► 任务 2：创建 SQL 数据库实例

1. 在界面左侧的导航窗格里，单击 **浏览**。
2. 在 **浏览** 边栏选项卡中，单击 **SQL 数据库**。
3. 在门户网站左下角，单击 **新建**。
4. 在 **创建** 边栏选项卡中，单击 **Data+Storage**，然后单击 **SQL Database**。
5. 在 **SQL 数据库** 边栏选项卡中，执行以下步骤：
  - a. 在 **名称** 框中，键入 **dp28532[自定义名称]**。
  - b. 单击 **服务器**。
  - c. 在 **服务器** 边栏选项卡中，单击 **创建一个新服务器**。
  - d. 在 **新服务器** 边栏选项卡中，定位到 **服务器名** 输入框。
  - e. 在 **服务器名** 框中，键入 **sp28532[自定义名称]**。
  - f. 在 **服务器管理登录** 框中，键入 **testuser**。
  - g. 在 **密码** 框中，键入 **TestPa\$\$w0rd**。
  - h. 在 **确认密码** 框中，键入 **TestPa\$\$w0rd**。
  - i. 单击 **位置**。
  - j. 在 **位置** 边栏选项卡中，选择离你所在位置最接近的区域。
  - k. 在 **新服务器** 边栏选项卡中，单击 **确认**。
  - l. 在 **SQL 数据库** 边栏选项卡中，单击 **定价层**。
  - m. 在 **将您的定价层更改为** 边栏选项卡中，选择 **B Basic**。
  - n. 单击 **选择**。
  - o. 单击 **选择源**。
  - p. 选择 **空白数据库** 选项。
  - q. 确认 **添加到启动板** 选项被选中。

- r. 在 **SQL 数据库** 边栏选项卡中，单击 **创建**。

 **注释：**SQL 数据库实例创建完成后，启动板上将显示该数据库图标，可以通过该图标访问数据库。

6. 新创建的 SQL 数据库边栏选项卡会自动显示。
7. 在 **dp28532[自定义名称]** 边栏选项卡中，单击 **属性** 磁贴。
8. 在 **属性** 边栏选项卡中，单击 **显示数据库连接字符串**。
9. 在 **数据库连接字符串** 边栏选项卡中，定位到 **ADO.NET**。
10. 复制连接字符串的值。
11. 在 Start 屏幕，键入 **Notepad**。
12. 单击 **Notepad**。
13. 将连接字符串粘贴到记事本中。
14. 在记事本中，将 **{Your\_password\_here}** 替换为 **TestPa\$\$w0rd**。
15. 切换到 **Internet Explorer** 中 **Microsoft Azure** 窗口。
16. 关闭 **dp28532[自定义名称]** 边栏选项卡。

#### ► 任务 3：创建存储实例

1. 在界面左侧的导航窗格里，单击 **浏览**。
2. 在 **浏览** 边栏选项卡中，单击 **存储帐户**。
3. 在 **存储帐户** 边栏选项卡中，查看已存在的 **存储** 实例列表。
4. 在屏幕左下角，单击 **新建**。
5. 在 **创建** 边栏选项卡中，单击 **Data + storage**，然后单击 **Storage**。
6. 在显示的 **存储帐户** 边栏选项卡中，执行以下步骤：
  - a. 在 **存储空间** 框中，键入 **sp28532[自定义名称]**。
  - b. 单击 **位置**。
  - c. 在 **位置** 边栏选项卡中，选择离你所在位置最接近的区域。
  - d. 单击 **定价层**。
  - e. 在 **选择你的定价层** 边栏选项卡中，选择 **L 本地冗余**。
  - f. 单击 **选择**。
  - g. 单击 **创建**。
7. 新创建的存储空间边栏选项卡会自动显示。
8. 在 **sp28532[自定义名称]** 边栏选项卡中，单击 **设置** 磁贴，然后单击 **密钥**。
9. 在 **管理密钥** 边栏选项卡中，复制 **存储帐户名称**。
10. 将 **存储帐户名称** 粘贴到记事本。
11. 复制 **主连接字符串**。
12. 将 **主连接字符串** 粘贴到记事本。
13. 复制 **主访问密钥**。

14. 将主访问密钥粘贴到记事本。
15. 切换到 **Internet Explorer** 中 **Microsoft Azure** 窗口。
16. 关闭 **sp28532[自定义名称]** 边栏选项卡。

► **任务 4：切换到 Azure 管理门户**

1. 在启动板上（预览门户主页），单击 **Azure** 门户。
2. 切换到 **Windows Azure** 选项卡。
3. 单击右上角的帐户名前的**地球图标**，选择**中文（简体）**选项。

► **任务 5：创建 Service Bus 名称空间**

1. 在左侧导航窗格中，单击 **SERVICE BUS**。
2. 在屏幕底部，单击**创建**。
3. 在**创建命名空间**对话框中，执行以下步骤：
  - a. 在**命名空间名称**框中，输入 **sp28532[自定义名称]**。
  - b. 在**区域**列表中，选择离你所在位置最接近的区域。
  - c. 单击**确定**按钮。



**注释：** 命名空间创建完成后，**Service Bus** 列表将显示新建项目。

4. 单击新创建的 Service Bus 命名空间名。
5. 在页面底部，单击**连接信息**。
6. 定位到 **SAS** 区域，记录 **RootManageSharedAccessKey** 连接字符串的值。
7. 将连接字符串值复制到记事本中。
8. 切换到 **Internet Explorer** 中 **Windows Azure** 窗口。
9. 单击确认按钮，关闭访问**连接信息**对话框。

**结果：** 完成此练习后，您将创建好将用于已部署 Web 应用程序的服务。

## 练习 2：管理云 Web 应用程序的配置设置

### ► 任务 1：修改云配置设置

 **注释：**若先前没有保存 Microsoft 帐户登录凭证，在这个实验过程中可能会遇到要求重新登录。

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏中，单击 **File Explorer**。
3. 在 This PC 窗口中。转到 **Allfiles (F):\Mod13\Labfiles\Starter\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Cloud** 项目，展开角色，然后双击 **Contoso.Events.Worker** 角色。
5. 在屏幕的左侧，单击**设置**选项卡。
6. 在页面的顶部，确保**服务配置**被设定为**所有配置**。
7. 定位到名称为 **Microsoft.ServiceBus.ConnectionString** 的设置。
8. 确保类型为**字符串**。
9. 在值框中输入之前任务中记录下的 **SAS 连接字符串**。
10. 定位到页面顶部的**服务配置**列表。
11. 选择 **Cloud**。
12. 定位到名称为 **Microsoft.WindowsAzure.Storage.ConnectionString** 的设置。
13. 在值输入框中，单击右侧省略号按钮。
14. 在**创建存储连接字符串**对话框中执行以下步骤：
  - a. 单击**手动输入凭据**。
  - b. 在**帐户名称**框中，输入先前任务中记下的**存储帐户名称**。
  - c. 在**帐户密钥**框中，输入先前任务中记下的**主访问密钥**。
  - d. 保留**连接**选项为默认设置：使用 **HTTPS (推荐)**。
  - e. 单击**确定**。
15. 定位到页面顶部的**服务配置**列表。
16. 选择 **Local**。
17. 确认 **Microsoft.WindowsAzure.Storage.ConnectionString** 的值设置为 **UseDevelopmentStorage=true**。
18. 在解决方案资源管理器窗格中，展开 **Contoso.Events.Cloud** 项目，展开**角色**文件夹，然后双击 **Contoso.Events.Web** 角色。
19. 在屏幕左侧，单击**设置**选项卡。
20. 定位到页面顶部的**服务配置**列表。
21. 选择 **Cloud**。
22. 定位到名称为 **Microsoft.WindowsAzure.Storage.ConnectionString** 的设置。
23. 在值输入框中，单击右侧省略号按钮。

24. 在**创建存储连接字符串**对话框中执行以下步骤:
  - a. 单击**手动输入凭据**。
  - b. 在**帐户名称**框中，输入先前任务中记下的**存储帐户名称**。
  - c. 在**帐户密钥**框中，输入先前任务中记下的**主访问密钥**。
  - d. 保留**连接**选项为默认设置：使用**HTTPS（推荐）**。
  - e. 单击**确定**。
25. 定位到页面顶部的**服务配置**列表。
26. 选择**Local**。
27. 确认 **Microsoft.WindowsAzure.Storage.ConnectionString** 的值设置为 **UseDevelopmentStorage=true**。

► **任务 2：修改管理项目的 Web.config 转换文件**

1. 在**解决方案资源管理器**窗格中，展开**Administration** 解决方案文件夹，展开**Contoso.Events.Management** 项目，展开**Web.config** 文件，然后双击**Web.Debug.config** 文件。
2. 定位到**appSettings** XML 元素。
3. 找到 key 为 **Microsoft.WindowsAzure.Storage.ConnectionString** 的 add 子 XML 元素。
4. 将元素的 value 属性更新为从**存储帐户**中记下的**主连接字符串**。
5. 定位到**appSettings** XML 元素。
6. 找到 key 为 **Microsoft.ServiceBus.ConnectionString** 的 add 子 XML 元素。
7. 将元素的 value 更新为从 Service Bus 命名空间记下的**SAS 连接字符串**。
8. 定位到**connectionStringsXML** 元素。
9. 找到 name 为 **EventsContextConnectionString** 的 add 子 XML 元素。
10. 将元素的 connectionstring 属性更新为从**SQL 数据库**实例中记下的**ADO.NET 连接字符串**。

► **任务 3：修改辅助角色项目的 app.config 文件**

1. 在**解决方案资源管理器**窗格中，展开**Roles** 文件夹，展开**Contoso.Events.Worker** 项目，然后双击**app.config** 文件。
2. 定位到**connectionStrings** XML 元素。
3. 找到 name 属性为 **EventsContextConnectionString** 的 add 子 XML 元素。
4. 将元素的 connectionstring 属性更新为从**SQL 数据库**实例中记下的**ADO.NET 连接字符串**。

► **任务 4：修改 Web 角色项目的 Web.config 转换文件**

1. 在**解决方案资源管理器**窗格中，展开**Roles** 文件夹，展开**Contoso.Events.Web** 项目，展开**Web.config** 文件，然后双击**Web.Release.config** 文件。
2. 定位到**connectionStrings** XML 元素。
3. 找到 name 属性为 **EventsContextConnectionString** 的 add 子 XML 元素。
4. 将元素的 connectionstring 属性更新为从**SQL 数据库**实例中记下的**ADO.NET 连接字符串**。

**结果：**完成此练习后，您将更改了 Web 应用程序，使其在部署到云后，使用不同的设置和连接字符串。

### 练习 3：将 Web 应用程序部署到 Azure

#### ► 任务 1：将云服务项目部署到云服务生产环境

1. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Cloud** 项目，然后单击发布。
2. 在登录向导页中，保留选择订阅默认选项。
3. 单击下一步转到设置向导页。
4. 定位到云服务列表，然后选择新建。

 **注释：**如果您的订阅中还没有任何的云服务，将自动跳转到创建云服务和存储帐户对话框而不需要在云服务列表中选择新建，你可以在显示的对话框中继续做第五步。

5. 在创建云服务对话框中，执行以下步骤：
  - a. 在名称框中，输入 **cs28532[自定义名称]**。
  - b. 在地区或地缘组列表中，选择最接近你所在位置的区域。
  - c. 单击创建。
6. 在环境列表中，选择 **Production**。
7. 在版本配置列表中，选择 **Release**。
8. 在服务配置列表中，选择 **Cloud**。
9. 其他字段保留默认值。
10. 单击发布。

 **注释：**如果弹出保存只读文件消息框，单击覆盖。

#### ► 任务 2：将 ASP.NET Web 应用程序项目部署到网站服务实例

1. 在视图菜单上，单击服务器资源管理器。
2. 定位到 **Azure** 的节点，然后单击左侧箭头。
3. 右键单击 Azure 节点下网站节点，然后单击创建新站点。
4. 在站点名称框中，使用以下格式键入网站的名称：**ws28532[自定义名称]**。
5. 在区域列表中，选择最接近您所在位置的区域。
6. 在数据库服务器列表中，确认选定了无数据库。
7. 单击创建。
8. 在解决方案资源管理器窗格中，展开 **Administration** 解决方案，右键单击 **Contoso.Events.Management** 项目，然后单击发布。
9. 在向导的配置文件页面，选择 **Microsoft Azure** 网站作为选择发布目标。
10. 在选择现有网站对话框中，执行以下步骤：
  - a. 从现有网站列表中选择前缀为 **ws28532** 的新网站。
  - b. 单击确定。
  - c. 在更改保存配置文件提示对话框中，单击 Yes 保存。
11. 在向导的连接页面，单击下一步转到设置页面。

12. 在向导的设置页面，将**配置**列表的值更改为**Debug – Any CPU**。
13. 单击**发布**。
14. 在发布完成后，确认新网站在 Internet Explorer 中显示。

► **任务 3：验证应用程序成功发布**

1. 在 Internet Explorer 中，查看**Contoso Events Administration** 的主页，然后单击**Go to Events List**。
2. 确认列表中显示的事件都没有注册。
3. 关闭 Internet Explorer。

**结果：**完成此练习后，您将把项目部署到网站服务和云服务。

## 练习 4：监视 Azure 中的 Web 应用程序

### ► 任务 1：查看“管理”应用程序的流日志

1. 在视图菜单，单击服务器资源管理器。
2. 定位到 **Azure** 的节点，然后单击左侧箭头。
3. 展开 **Azure** 节点下的网站节点。
4. 右键单击前缀为 **ws28532** 的网站节点，然后单击**查看设置**。
5. 定位到**应用程序日志记录（文件系统）**设置。
6. 将列表的值由**关闭**改为**详细**。
7. 单击**保存**。
8. 在**服务器资源管理器**中，定位到前缀为 **ws28532** 的网站节点。
9. 右键单击前缀为 **ws28532** 网站节点，然后单击**查看流式日志**。
10. 在**服务器资源管理器**中，定位到前缀为 **ws28532** 的网站节点。
11. 右键单击前缀为 **ws28532** 网站节点，然后单击**在浏览器中查看**。
12. 在 **Internet Explorer** 中，查看 **Contoso Events Administration** 主页，然后单击 **Go to Events**。
13. 切换到 **Contoso.Events – Microsoft Visual Studio** 窗口。
14. 验证控制台窗格的输出有两个条目，一条产生于查看网站 **Index** 页面时，另一条产生于查看网站 **Events** 页面时。
15. 在**服务器资源管理器**中，定位到前缀为 **ws28532** 的网站节点。
16. 右键单击前缀为 **ws28532** 网站节点，然后单击**停止查看日志**。
17. 关闭 **Internet Explorer** 应用程序。
18. 关闭 **Microsoft Visual Studio** 应用程序。

**结果：**完成此练习后，您将查看了网站服务的跟踪日志实时流。

# 实验 B：使用中国版 Windows Azure 部署 Events Web Application 到云环境

## 练习 1：创建部署 web 应用所需的 Azure 服务

► 任务 1：登录到 Azure 管理门户

1. 在 Start 屏幕，单击 **Internet Explorer** 磁贴。
2. 进入 <https://manage.windowsazure.cn>。
3. 输入中国版 Windows Azure 帐户的邮箱地址和密码。
4. 单击 **Sign in** 登录。

► 任务 2：创建一个 SQL 数据库实例

1. 在界面左侧导航窗格中，单击 **SQL 数据库**。
2. 在界面左下角，单击**新建**。
3. 在**新建向导**中依次单击**数据服务**, **SQL 数据库**, **自定义创建**。
4. 在**新建 SQL 数据库**页面中，执行以下步骤创建 SQL 数据库实例：
  - a. 在名称框中输入 **dp28532[自定义名称]**。
  - b. 在服务层中选择 **Basic**。
  - c. 单击**服务器**。
  - d. 在**服务器**列表中，选择**新建 SQL 数据库服务器**。
  - e. 单击**下一步**。
  - f. 在**创建服务器**页面中，在**登录名**框中输入 **testuser**。
  - g. 在**登录密码**框中输入 **TestPa\$\$w0rd**。
  - h. 在**确认密码**框中输入 **TestPa\$\$w0rd**。
  - i. 单击**区域**。
  - j. 在**区域**列表中选择离你最近的**区域**。
  - k. 单击**完成**。



**注释：**在创建数据库实例完成以后，可以到管理门户左侧的所有项目或者 **SQL 数据库**查看刚创建的数据库。

5. 在**所有项目**中单击刚创建的数据库。
6. 在**dp28532[自定义名称]**窗格中单击**仪表板**。
7. 在**仪表板**窗格中单击**显示连接字符串**。
8. 在**连接字符串**窗格中找到 **ADO.Net**。
9. 复制连接字符串。
10. 回到 windows 开始屏幕，输入 **Notepad**。
11. 单击 **Notepad** 磁贴。

12. 把连接字符串粘贴到记事本中。
13. 在记事本中，把字符串{此处为您的密码}替换为 **TestPa\$\$w0rd**。
14. 切回到浏览器的 **SQL 数据库 Microsoft Azure** 窗口。
15. 关闭连接字符串 窗格。

► **任务 3：创建一个存储实例**

1. 在左侧导航窗格中单击**存储**。
2. 在存储页面查看显示的**存储实例**。
3. 在界面左下角单击**新建**。
4. 在**新建向导**中，依次单击**数据服务**, **存储**, **快速创建**。
5. 在存储帐户信息页面中，执行以下步骤：
  - a. 在 **URL** 中填写 **sp28532[自定义名称]**。
  - b. 单击**位置/地缘组**，选择离你最近的位置。
  - c. 单击**地域冗余**，选择**本地冗余**。
  - d. 单击**创建存储帐户**。

 **注释：**当你**存储帐户**创建完成后，可以在左侧导航区**所有项目**或者**存储**中找到刚创建的存储帐户。

6. 单击左侧导航窗格中**存储**，定位到刚创建的存储帐户 **sp28532[自定义名称]**，单击 **sp28532[自定义名称]**。
7. 在 **sp28532[自定义名称]**窗格底部，单击**管理访问密钥**。
8. 在**管理访问密钥**窗格，复制**主访问密钥**的值。
9. 将**主访问密钥**粘贴到记事本中。

 **注释：**存储帐户对应的连接字符串为：

```
BlobEndpoint=https://sp28532[自定义名称]
].blob.core.chinacloudapi.cn;/QueueEndpoint=https://sp28532[自定义名称]
].queue.core.chinacloudapi.cn;/TableEndpoint=https://sp28532[自定义名称]
].table.core.chinacloudapi.cn;/AccountName={sp28532[自定义名称]};AccountKey={
存储帐户名对应的密钥}
将其粘贴到记事本中，接下来的实验中会使用到。
```

10. 切换到**存储 Microsoft Azure** 窗口。

11. 关闭**管理访问密钥**页面。

► **任务 4：创建一个 Service Bus 命名空间**

1. 在界面左侧导航窗格中，单击**Service Bus**。
2. 在**Service Bus** 页面中，查看命名空间。
3. 在界面下方单击**创建**。
4. 在**创建命名空间**页面中，执行以下步骤：
  - a. 在**命名空间名称**中，输入 **sp28532[自定义名称]**。

- b. 在**区域**列表中选择离你最近的区域。
- c. 单击**确定**按钮。



**注释:** 当命名空间创建完毕后，会显示在 Service Bus 命名空间列表中。

5. 单击刚创建的命名空间 **sp28532[自定义名称]**。
6. 在页面的下方单击**连接信息**。
7. 定位到**SAS** 区域，记录 **RootManageSharedAccessKey** 连接字符串的值。
8. 将连接字符串值复制到记事本中。
9. 切换到浏览器 **Service Bus - Microsoft Azure** 窗口。
10. 关闭**访问连接信息**窗格。

**结果:** 完成这个练习，你将创建你部署 web 应用所需要的的服务。

## 练习 2：管理云上 Web 应用的配置设置

### ► 任务 1：更改云配置设置

 **注释：**如果你之前没有保存过你的帐户凭证，在这次实验过程中你可能会被告知你需要登录。具体步骤请参考第 2 章->实验 B->练习 4->任务 1，任务 2。

1. 在 Start 屏幕，单击 **Desktop** 磁贴。
2. 在任务栏中，单击 **File Explorer**。
3. 在 This PC 窗口，浏览到 **Allfiles (F):\Mod13\Labfiles\Starter\Contoso.Events**，然后双击 **Contoso.Events.sln**。
4. 在解决方案资源管理器窗格，展开 **Contoso.Events.Cloud** 项目，展开**角色**文件夹，然后双击 **Contoso.Events.Worker** 这个角色。
5. 在屏幕左侧，单击**配置**标签。
6. 在页面的顶部，确认**服务配置**设置为**所有配置**。
7. 定位到名称为 **Microsoft.ServiceBus.ConnectionString** 的设置。
8. 确认类型设置为**字符串**。
9. 在**值**框中，输入 **SAS 连接字符串**。

 **注释：**SAS 连接字符串在之前的任务中已粘贴到记事本中。格式如：Endpoint=sb://{命名空间名}.servicebus.chinacloudapi.cn/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=.....

10. 定位到页面顶部的**服务配置**。
11. 选择 **Cloud**。
12. 定位到名称为 **Microsoft.WindowsAzure.Storage.ConnectionString** 的设置。
13. 单击**值**框，然后单击右边的 (...) 省略号。
14. 在弹出的**创建存储连接字符串**对话框，执行以下步骤：
  - a. 单击**您的订阅**。
  - b. 在**帐户名称**列表中，选择上一个练习任务中创建的存储帐户 **sp28532[自定义名称]**。
  - c. **连接使用默认连接：使用 HTTPS(推荐)**。
  - d. 预览连接字符串。
  - e. 单击**确定**。
15. 定位到页面顶部的**服务配置**。
16. 选择 **Local**。
17. 确认 **Microsoft.WindowsAzure.Storage.ConnectionString** 的值为 **useDevelopmentStorage=true**。
18. 在解决方案资源管理器窗格，展开 **Contoso.Events.Cloud** 项目，展开**角色**文件夹，然后双击 **Contoso.Events.Web** 这个角色。
19. 在屏幕左侧，单击**配置**标签，然后定位到页面顶部的**服务配置**。
20. 选择 **Cloud**。
21. 定位到名称为 **Microsoft.WindowsAzure.Storage.ConnectionString** 的设置。

22. 单击**值**框，然后单击右侧的 (...) 省略号。
23. 在弹出的**创建存储连接字符串**对话框，执行以下步骤：
  - a. 单击**您的订阅**。
  - b. 在**帐户名称**列表中，选择上一个练习任务中创建的存储帐户 **sp28532[自定义名称]**。

 **注释：**可以看到**预览连接字符串**，格式为：BlobEndpoint=https://[sp28532[自定义名称]].blob.core.chinacloudapi.cn;/QueueEndpoint=https://[sp28532[自定义名称]].queue.core.chinacloudapi.cn;/TableEndpoint=https://[sp28532[自定义名称]].table.core.chinacloudapi.cn;/AccountName={sp28532[自定义名称]};AccountKey={存储帐户的主访问密钥}

- c. **连接**使用默认连接：**使用 HTTPS(推荐)**。
- d. 单击**确定**。
24. 定位到页面顶部的**服务配置**。
25. 选择 **Local**。
26. 确认 **Microsoft.WindowsAzure.Storage.ConnectionString** 的值设置为 **useDevelopmentStorage=true**。
27. 单击**端点**，确认当前的公用端口，从浏览器访问云服务的时候最好带上端口号。

 **注释：**因为设置的公用端口可能不是浏览器默认的 80 端口。如 <http://contosoevents.chinacloudapp.cn: 85>

#### ► 任务 2：更改管理项目的 **Web.config** 转换文件

1. 在解决方案资源管理器窗格中，展开 **Administration** 文件夹，展开 **Contoso.Events.Management** 项目，展开 **Web.config** 文件，然后双击文件 **Web.Debug.config**。
2. 定位到 **appSettings** XML 元素。
3. 定位到 key 的值为 **Microsoft.WindowsAzure.Storage.ConnectionString** 的 add 子元素。
4. 将元素 **value** 值设置成存储帐户对应的连接字符串。
5. 定位到 **appSettings** XML 元素。
6. 定位到 key 的值为 **Microsoft.ServiceBus.ConnectionString** 的 add 子元素。
7. 将元素 **value** 设置成之前记录的 Service Bus 命名空间的 **SAS 连接字符串**。
8. 定位到 **connectionStrings** XML 元素。
9. 定位到 name 为 **EventsContextConnectionString** 的 add 子元素。
10. 将元素 **connectionString** 更新为从 SQL 数据库实例中记录的 **ADO.NET 连接字符串**。

#### ► 任务 3：更改辅助角色项目的 **app.config** 文件

1. 在解决方案资源管理器窗格中，展开 **Roles** 文件夹，展开 **Contoso.Events.Worker** 项目，然后双击 **app.config** 文件。
2. 定位到 **connectionStrings** XML 元素。
3. 定位到 name 为 **EventsContextConnectionString** 的 add 子元素。
4. 将元素 **connectionString** 更新为从 SQL 数据库实例中记录的 **ADO.NET 连接字符串**。

► 任务 4：更新 web 角色项目的 web.config 转换文件

1. 在解决方案资源管理器窗格，展开 **Roles** 文件夹，展开 **Contoso.Events.Web** 项目，展开 **web.config** 文件，然后双击文件 **Web.Release.config**。
2. 定位到 **connectionStrings** XML 元素。
3. 定位到 name 为 **EventsContextConnectionString** 的 add 子元素。
4. 将元素 **connectionString** 更新为从 SQL 数据库实例中记录的 **ADO.NET 连接字符串**。

**结果：**完成这个练习后，你的 web 应用将被设置成：即使应用发布到云之后，他们依然可以更改这个应用的配置和连接字符串信息。

### 练习 3：部署 Web 应用到云环境

#### ► 任务 1：部署云服务项目到 Azure 的产品环境

1. 在解决方案资源管理器窗格中，右键单击 **Contoso.Events.Cloud** 项目，然后单击发布。
2. 在登录向导中，在选择订阅列表中选择您的订阅。
3. 单击下一步，进入设置向导页面。
4. 定位到云服务列表，然后选择新建。

 **注释：**如果在你的订阅中没有一个云服务，不需要选择新建，它将自动弹出创建云服务对话框。你可以继续第五步。

5. 在创建云服务对话框，执行以下步骤：
  - a. 在名称框中，输入 **cs28532[自定义名称]**。
  - b. 在地区或地缘组列表中，选择离你最近的区域。
  - c. 单击创建。
6. 在环境列表中选择 **Production**。
7. 在版本配置列表中选择 **Release**。
8. 在服务配置列表中选择 **Cloud**。
9. 单击发布。

#### ► 任务 2：部署一个 ASP.NET Web 应用项目到云环境的网站

1. 在视图菜单，单击服务器资源管理器。
2. 定位到 Azure 节点，单击左边箭头。
3. 右键单击 Azure 节点下的网站，单击创建新站点。
4. 在站点名称框中，输入网站名称 **ws28532[自定义名称]**。
5. 在区域列表中，选择离你最近的区域。
6. 在数据库服务器列表中，确认选中无数据库。如果出现感叹号，且没有值可选，刷新一下网站节点就行。
7. 单击创建。
8. 在解决方案资源管理器窗格中，展开 **Administration** 文件夹，右键单击 **Contoso.Events.Management** 项目，然后单击发布。
9. 在配置文件中，选择 **Microsoft Azure** 网站为发布目标。
10. 在选择现有网站对话框中，执行以下步骤：
  - a. 在现有网站列表中，选择 **ws28532[自定义名称]**。
  - b. 单击确定。
  - c. 在弹出的是否保存提示框中，单击 Yes。
11. 在连接页面中，单击下一步。
12. 在设置页面中，将配置更改为 **Debug – Any CPU**。
13. 单击发布。

14. 发布完成之后，确认网站成功发布。

► **任务 3：确认网站已经成功发布**

1. 在 Internet Explorer 中，浏览 **Contoso Events Administration** 主页，然后单击 **Go to Events List**。
2. 确认所有的事件的 Registrants 值都为 0。
3. 关闭浏览器。

**结果：**完成这次练习，你将学会部署项目到 Azure 的网站和云服务。

## 练习 4：在 Azure 上监测 Web 应用

### ► 任务 1：查看网站的流式日志

1. 在视图菜单，单击**服务器资源管理器**。
2. 定位到**Azure** 节点，单击左边箭头。
3. 展开**Azure** 节点下的**网站**。
4. 右键单击前缀为**ws28532** 的网站，然后单击**查看设置**。
5. 定位到**应用程序日志记录（文件系统）**。
6. 将值由**关闭**改为**详细**。
7. 单击**保存**。
8. 右键单击前缀为**ws28532** 的网站，然后单击**查看流式日志**。
9. 右键单击前缀为**ws28532** 的网站，然后单击**在浏览器中查看**。
10. 在 Internet Explorer 中，在**Contoso Events Administration** 页面中，单击**Go to Events List**。
11. 切换到**Contoso.Events – Microsoft Visual Studio** 窗口。
12. 确认控制台窗格输出两条这样的记录 Information Viewed the Index page 和 Information Viewd the Events page。
13. 右键单击前缀为**ws28532** 的网站，然后单击**停止查看日志**。
14. 关闭**Internet Explorer** 应用程序。
15. 关闭**Microsoft Visual Studio** 应用程序。

**结果：**完成这次练习，你将会查看一个网站的流式日志。