

# 一、应用服务部署槽位过渡应用并进行测试和回滚

## 创建部署槽位

## 创建 Web 应用

首先在 Azure 门户中创建新的 Web 应用资源：

1. 登录 [Azure 门户](#)。
2. 选择“创建资源”。
3. 选择“Web”>“Web 应用”。
4. 使用类似于以下示例的信息填写向导：

字段	值
订阅	选择要用来完成练习的订阅
资源组	创建名为“mslearn-slots”的新资源组
名称	输入唯一名称
发布	代码
运行时堆栈	ASP.NET V4.7
操作系统	Windows
区域	选择附近的区域
Windows 计划	保留默认值
SKU 和大小	保留默认值

主页 > 新建 > Web 应用 > Web 应用

## Web 应用

基本 监视 标记 查看 + 创建

借助应用服务 Web 应用，可快速构建、部署和扩展在任意平台上运行的企业级 Web、移动和 API 应用。符合严苛的性能、可伸缩性、安全性和合规性要求，同时可使用完全托管的平台来执行基础结构维护。 [了解详细信息](#)

**项目详细信息**

选择一个订阅来管理部署的资源和成本。使用文件夹等资源组织和管理你的所有资源。

订阅 \* ① Microsoft Azure 内部消耗

资源组 \* ① (新项) labs 新建

**实例详细信息**

名称 \* kiwebapps .azurewebsites.net

发布 \* 代码 Docker 容器

运行时堆栈 \* ASP.NET V4.7

操作系统 \* Linux Windows

区域 \* East Asia 找不到应用服务计划，请尝试其他区域。

**应用服务计划**

应用服务计划定价层确定与你的应用相关的位置、功能、成本和计算资源。 [了解详细信息](#)

Windows 计划 (East Asia) \* ① (新项) kiwebappspplan 新建

SKU 和大小 \* 标准 S1 100 总 ACU, 1.75 GB 内存 更改大小

5. 导航到页面顶部的“监视”选项卡，然后将“启用 Application Insights”切换为“否”。

Microsoft Azure (预览) 报告 bug

主页 > 新建 > Web 应用 > Web 应用

## Web 应用

基本 监视 标记 查看 + 创建

Azure Monitor 让你能够完整监视你的应用程序、基础结构和网络。 [了解详细信息](#)

**Application Insights**

启用 Application Insights \*

否 是

6. 选择“查看并创建”，然后在生成的页上选择“创建”。等待 Azure 创建 Web 应用。

主页 > 新建 > Web 应用 > Web 应用

## Web 应用

基本 监视 标记 查看 + 创建

**摘要**

 **Web 应用**  
由 Microsoft

**详细信息**

订阅	5ff6f25a-381e-44ba-8897-cefc2c3c3ccb
资源组	labs
名称	klwebapps
发布	代码
运行时堆栈	ASP.NET V4.7

**应用服务计划(新)**

名称	klwebappspalan
操作系统	Windows
区域	East Asia
SKU	标准
大小	小
ACU	100 总 ACU
内存	1.75 GB 内存

**监视**

Application Insights 未启用

## 配置 Git 部署

使用适用于 Web 应用及其部署槽位的任何常用部署工具。 在本练习中，你将使用本地 Git 存储库进行部署。 通过执行以下步骤，设置 Web 应用以使用

Git：

- 在 Azure 门户的左侧，选择“所有资源”，然后选择创建的 Web 应用。
- 在“概述”页的“部署”下，选择“部署中心”。

Microsoft Azure 门户 报告 bug 搜索资源、服务和文档

主页 > Microsoft.Web -> Microsoft.Web-Portal-6fbfa477f-98dc - 概述 > klwebapps

klwebapps

部署中心

浏览 停止 交换 重新启动 撤除 获取发布配置文件 重置发布配置文件

单击此处访问快速入门指南，将代码部署到应用 →

资源组 (更改) : labs

状态 : Running

位置 : 东亚

订阅 (更改) : Microsoft Azure 内部测试

订阅 ID : 5ff6f25a-381e-44ba-8897-cefc2c3c3ccb

标记 (更改) : 单击此处以添加标记

URL : <https://klwebapps.azurewebsites.net>

应用服务计划 : klwebappspalan (S1:1)

FTP/部署用户名 : 未设置 FTP/部署用户名

FTP 主机名 : <ftp://waws-prod-hk1-029.ftp.azurewebsites.windows.net>

HTTPS 主机名 : <https://waws-prod-hk1-029.ftp.azurewebsites.windows.net>

**诊断并解决问题**

**Application Insights**

**应用服务顾问**

3. 选择“本地 Git”>“继续”。

**部署中心**

应用服务部署中心使你能够选择代码的位置以及生成和部署到云的选项。 [了解详细信息](#)

1 源代码管理    2 生成提供程序    3 配置

**持续部署(CI / CD)**

- Azure Repos** 使用 Azure Repo (Azure DevOps 服务 - 以前称为 VSTS) 的一部分配置持续集成。  
未授权
- GitHub** 使用 GitHub 存储库配置持续集成。  
未授权
- Bitbucket** 使用 Bitbucket 存储库，配置持续集成。  
未授权
- Local Git** 从本地 Git 存储库部署。  
**本地 Git**

**手动部署(推送/同步)**

- OneDrive** 从 OneDrive 云文件夹同步内容。  
未授权
- Dropbox** 从 Dropbox 云文件夹同步内容。  
未授权
- External** 从公共 Git 或 Mercurial 存储库中部署。此选项需要手动触发部署。
- FTP** FTP 使用 FTP 连接访问和复制应用文件。

4. 选择“应用服务生成服务”>“继续”>“完成”。

**部署中心**

应用服务部署中心使你能够选择代码的位置以及生成和部署到云的选项。 [了解详细信息](#)

1 源代码管理    2 生成提供程序    3 配置

**应用服务生成服务** 将应用服务用作生成服务器。如果无需其他配置，在运行时应用服务 Kudu 引擎将为部署期间自动生成代码。

**Azure Pipelines (预览)** 使用 Azure Pipelines (Azure DevOps 服务 - 以前称为 VSTS) 的一部分为应用服务配置构建管道，该管道生成、运行负载测试并部署到测试环境，然后部署到生产中。

**klwebapps - 部署中心**

应用服务部署中心使你能够选择代码的位置以及生成和部署到云的选项。 [了解详细信息](#)

1 源代码管理    2 生成提供程序    3 配置

**源代码管理** 完成  
分支

**生成提供程序** 完成  
使用服务生成服务

**配置** 完成

5. 在生成的“部署中心”页中，选择顶部的“本地 Git”，然后选择“用户凭据”选项卡。

The screenshot shows the Azure App Service Deployment Center interface. At the top, there's a search bar and three buttons: 'Browse', 'Refresh' (which is highlighted with a red box), and 'Disconnect'. Below that, it says 'Source: Local Git' and 'Generator: Kudu'. To the right, it shows the 'Git Clone URL: https://klwebapps.scm.azurewebsites.net:443/'. On the left, there's a sidebar with various tabs like 'Overview', 'Activity Log', 'Access Control', etc., and a 'Deployment' section with 'Quick Start' and 'Deployment Slots'. The 'Deployment Slots' tab is currently selected. At the bottom, there's a table with columns 'Time' and 'Status'.

6. 输入所需的新用户名和密码，然后选择“保存凭据”。记下用户名和密码以供日后使用。

This is a screenshot of the 'Deployment Credentials' configuration dialog. It has tabs for 'Local Git' (highlighted with a red box) and 'FTP/FTPS'. The 'Git Clone URL' field contains 'https://klwebapps-staging.scm.azurewebsites.net:443/klwebapps.git'. Under the 'Scope' section, the 'User Credential' tab is highlighted with a red box. The 'Username' field is set to 'kladministrator', and the 'Password' and 'Confirm Password' fields both contain '\*\*\*\*\*'. At the bottom, the 'Save Credentials' button is highlighted with a red box.

或用 Cloud Shell 命令行去设定

```
az webapp deployment user set --user-name <username> --password <password>
```

然后用以下命令检查生成的 Git URL

```
az webapp deployment source config-local-git --name <appservicename> --resource-group <group-name>
```

## 配置 Git 客户端并克隆 Web 应用源代码

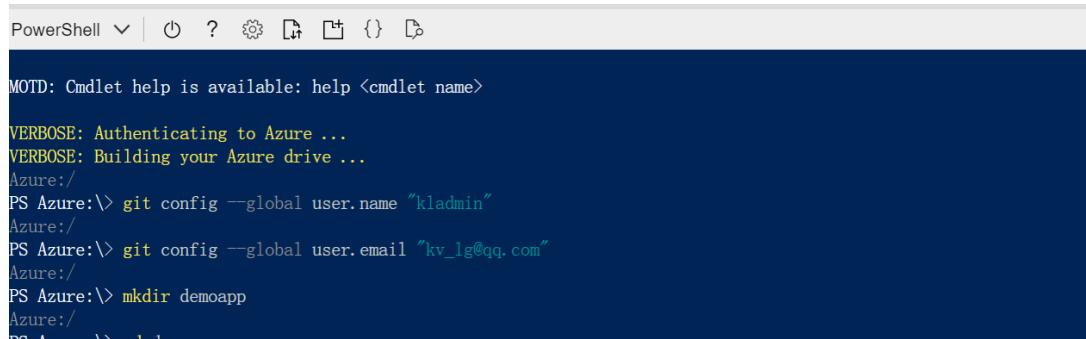
现在，在 Cloud Shell 中设置 Git 客户端并将其用于克隆示例 Web 应用。

执行以下步骤：

- 在 Azure 门户中打开 Cloud Shell。 输入以下命令以设置 Git 用户名和电子邮件地址。 这些命令与任何帐户或注册均不关联，可以随意使用任何值。

bash 复制

```
git config --global user.name "<your name>"  
git config --global user.email "<your email address>"
```



The screenshot shows a PowerShell session in the Azure Cloud Shell. The command history includes:

```
PowerShell | ⚡ ? 🌐 🔍 ⚡ {} 🔍  
  
MOTD: Cmdlet help is available: help <cmdlet name>  
  
VERBOSE: Authenticating to Azure ...  
VERBOSE: Building your Azure drive ...  
Azure:/  
PS Azure:\> git config --global user.name "kladmin"  
Azure:/  
PS Azure:\> git config --global user.email "kv_1g@qq.com"  
Azure:/  
PS Azure:\> mkdir demoapp  
Azure:/  
PS Azure:\>
```

- 若要为源代码创建文件夹，请输入以下命令：

bash 复制

```
mkdir demoapp  
cd demoapp
```

- 若要克隆 Web 应用的源，请输入以下命令：

bash 复制

```
git clone https://github.com/Azure-Samples/app-service-  
web-dotnet-get-started.git  
cd app-service-web-dotnet-get-started
```

```
PowerShell ✓ | ⌂ ? ⚙️ 📁 ⚡ { } ⌂
Azure:/  
PS Azure:> bash  
kelvin@Azure:~$ ls  
clouddrive demoapp  
kelvin@Azure: $ cd demoapp  
kelvin@Azure:~/demoapp$ ls  
kelvin@Azure:~/demoapp$ git clone https://github.com/Azure-Samples/app-service-web-dotnet-get-started.git  
Cloning into 'app-service-web-dotnet-get-started'...  
remote: Enumerating objects: 122, done.  
remote: Total 122 (delta 0), reused 0 (delta 0), pack-reused 122  
Receiving objects: 100% (122/122), 397.09 KiB | 147.00 KiB/s, done.  
Resolving deltas: 100% (31/31), done.  
Checking connectivity... done.  
kelvin@Azure:~/demoapp$ ls  
app-service-web-dotnet-get-started  
kelvin@Azure:~/demoapp$ cd app-service-web-dotnet-get-started  
kelvin@Azure:~/demoapp/app-service-web-dotnet-get-started$
```

配置 git remote 将应用部署到生产环境

若要使用 Git 将源代码部署到 Web 应用的生产槽，请将应用的 Git URL 设置为远程存储库。请执行以下步骤：

1. 在 Azure 门户的 Web 应用的“概述”页上，选择“Git 克隆 URL”旁边的“复制”按钮。请注意，该 URL 包含部署用户名。

The screenshot shows the Azure ShareCenter management interface. On the left, there's a sidebar with navigation links: 概览 (Overview), 活动日志 (Activity Log), 访问控制 (Identity and Access Management), 标记 (Tags), 诊断并解决问题 (Diagnose and Solve Problems), 安全性 (Security), 部署 (Deployment), 快速入门 (Quick Start), 部署槽位 (Deployment Slots), and 部署中心 (Deployment Center). The main content area displays deployment details for a 'ShareCenter' application. It includes sections for 资源组 (Resource Group), 状态 (Status), 位置 (Location), 订阅 (Subscription), 标记 (Tags), and deployment-related URLs and IDs. A purple banner at the top right provides a quick link to deployment guides.

部署	资源组 (更改) ShareCenterRG	URL <a href="https://sharecenter.azurewebsites.net">https://sharecenter.azurewebsites.net</a>
状态	正在运行	应用服务计划 <a href="#">ShareCenterAPS (标准: 1 个小型)</a>
位置	美国中部	Git/部署用户名
订阅	<a href="#">更改</a> <a href="#">即用即付</a>	Git 克隆 URL <a href="https://@sharecenter.scm.azurewebsites.net:443/">https://@sharecenter.scm.azurewebsites.net:443/</a>
部署	订阅 ID ce698c92-57c9-43df-b269-2e10273a0425	FTP 主机名 <a href="ftp://waws-prod-dm1-091.ftp.azurewebsites.windows.net">ftp://waws-prod-dm1-091.ftp.azurewebsites.windows.net</a>
快速入门	标记 (更改) <a href="#">单击此处添加标记</a>	
部署槽位		
部署中心		

备注

如果看不到前面的屏幕截图中显示的 Git 克隆 URL，请刷新门户。

2. 在 Cloud Shell 中，运行以下命令以将 URL 配置为名为“production”的 git remote。将 git-clone-url 替换为上一步中的 URL。

## bash 复制

```
git remote add production <git-clone-url>
```

```
VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
azure:/
PS Azure:\> bash
kelvin@Azure:~$ ls
clouddrive demoapp
kelvin@Azure:~$ cd demapp
bash: cd: demapp: No such file or directory
kelvin@Azure:~$ cd demoapp
kelvin@Azure:~/demoapp$ ls
app-service-web-dotnet-get-started
kelvin@Azure:~/demoapp$ cd app-service-web-dotnet-get-started
kelvin@Azure:~/demoapp/app-service-web-dotnet-get-started$ ls
aspnet-get-started aspnet-get-started.sln CONTRIBUTING.md LICENSE README.md
kelvin@Azure:~/demoapp/app-service-web-dotnet-get-started$ git remote add production https://null@klwebapps.scm.azurewebsites.net:443/klwebapps.git
kelvin@Azure:~/demoapp/app-service-web-dotnet-get-started$ [ ]
```

3. 若要将 Web 应用部署到生产槽，请输入以下命令。当系统提示输入密码时，请输入之前创建的部署密码。

## bash 复制

```
git push production
```

4. 部署完成后，在 Azure 门户中，转到 Web 应用的“概述”页，然后选择“浏览”。Azure 会显示 Web 应用：

Microsoft Azure (预览) | 报告 bug | 搜索资源、服务和文档(G+I)

主页 > klwebapps

klwebapps 应用服务

搜索(Ctrl+ /)

概述 活动日志 访问控制(标识和访问管理) 标记 诊断并解决问题 安全性

部署 快速入门 部署槽 部署中心

资源组 (更改) 浏览

位置 : 东亚  
订阅 (更改) : Microsoft Azure 内部消耗  
订阅 ID : 5ff6f25a-381e-44ba-8897-cefc2c3c3ccb  
标记 (更改) : 单击此处以添加标记

诊断并解决问题 Application Insights

应用程序名称 开始 关于 联系人

# ASP.NET

ASP.NET 是一个免费的 Web 框架，用于使用 HTML、CSS 和 JavaScript 生成出色的网站和 Web 应用程序。

了解更多 »

## 入门

ASP.NET MVC 为你提供了一种基于模式的功能强大的方法来生成动态网站，从而清晰划分关注点，并使你可以完全控制标记，从而享受敏捷的开发体验。

了解更多 »

## 获取更多库

NuGet 是一个免费的 Visual Studio 扩展，使用它可以在 Visual Studio 项目中轻松添加、删除和更新库和工具。

了解更多 »

## Web 宿主

可以轻松找到一个网络托管公司，为你的应用程序提供适合的功能和价格组合。

了解更多 »

© 2019 - 我的 ASP.NET 应用程序

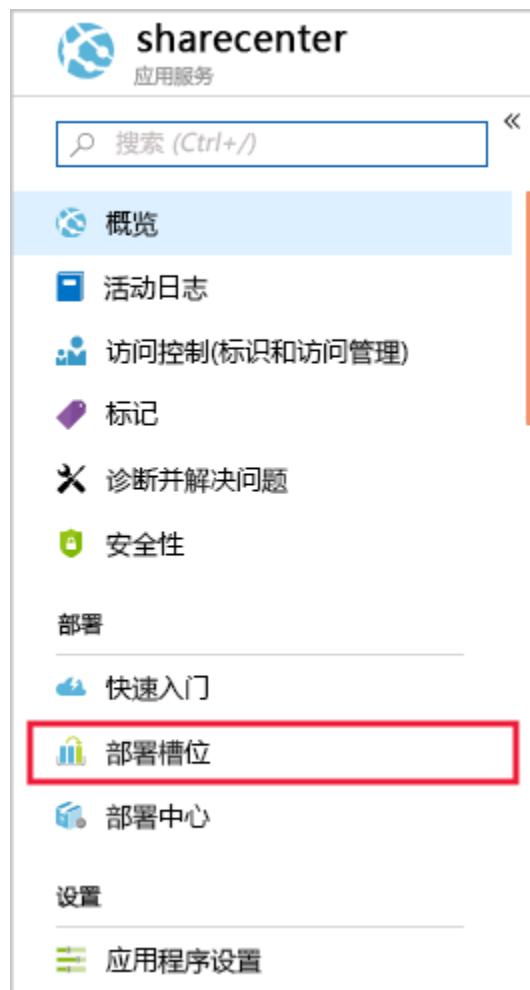
5. 关闭显示 Web 应用的浏览器选项卡。

## 创建新的过渡槽

对于新 Web 应用，只创建了一个槽：生产槽。已将源代码部署到此槽。

接下来将创建部署槽位，可以在其中过渡新版 Web 应用：

1. 在 Azure 门户中，选择“所有资源”，然后选择 Web 应用。
2. 在“部署”下，选择“部署槽位”。



3. 在“部署槽位”页上，选择“添加槽”。
4. 在“名称”框中，输入“staging”，然后选择“添加”。
5. 添加部署槽位后，选择“关闭”。

## 设置过渡槽的 Git 部署

设置新槽以使用 Git 部署，就像对生产槽执行的操作一样。 请执行以下步骤：

1. 在 Azure 门户的左侧，选择“所有资源”。 在所有资源的列表中，现在将看到两个 Web 应用条目：部署槽位以门户中的单独应用来表示。 选择表示过渡槽的条目，以转到其“概述”页。
2. 在“部署”下，选择“部署中心”。
3. 选择“本地 Git” > “继续”。
4. 选择“应用服务生成服务” > “继续” > “完成”。

## 设置 Git 以将应用部署到过渡槽

若要使用 Git 客户端将源代码部署到新槽，请将额外的 remote 添加到 Git 配置中。 请执行以下步骤：

1. 在 Azure 门户中，转到过渡槽的“概述”页。 在页面顶部附近，选择“Git 克隆 URL”旁边的“复制”按钮。

### 备注

如果看不到前面的屏幕截图中显示的 Git 克隆 URL，请刷新门户。 请注意，过渡槽的 Git URL 与生产槽的 URL 稍有不同，并且包括槽名称。

2. 若要为过渡槽添加 remote，请在 Cloud Shell 中运行以下命令。将 git-clone-url 替换为上一步中的 URL。

bash 复制

```
git remote add staging <git-clone-url>
```

```
PS Azure:\> bash
kelvin@Azure: $ ls
clouddrive demoapp
kelvin@Azure: $ cd demoapp
kelvin@Azure: ~/demoapp$ ls
app-service-web-dotnet-get-started
kelvin@Azure: ~/demoapp$ cd *
kelvin@Azure: ~/demoapp/app-service-web-dotnet-get-started$ ls
aspnet-get-started aspnet-get-started.sln CONTRIBUTING.md LICENSE README.md
kelvin@Azure: ~/demoapp/app service web dotnet get-started$ git remote add staging https://kladminstrator@klwebapps-过渡.scm.azurewebsites.net:443/klwebapps.git
kelvin@Azure: ~/demoapp/app service web dotnet get-started$ code .
kelvin@Azure: ~/demoapp/app-service-web-dotnet-get-started$
```

## 修改应用源代码并将应用部署到过渡槽

接下来，对 Web 应用稍作更改，然后使用 Git 将新版本部署到过渡槽：

1. 在 Cloud Shell 中，输入以下命令：

bash 复制

```
code .
```

2. 在“文件”列表中，展开“aspnet-get-started” > “视图” > “主页”。

3. 选择“Index.cshtml”。

```
PowerShell └── D:\aspnet\app-service\service\wwwroot\dotnet\get-started\$ code .  
kelvin@Mure: /d/aspnet/app-service/web/dotnet/get-started\$ dotnet new mvc -n get-started
```

The screenshot shows a PowerShell window with the title bar "上传/下载文件". The file tree on the left lists the project structure:

- 文件
- aspnet-get-started
  - App\_Start
  - Content
  - Controllers
  - fonts
  - Properties
  - Scripts
  - Views
    - Home
      - About.cshtml
      - Contact.cshtml
      - Index.cshtml**
    - Shared

The "Index.cshtml" file is selected in the file tree. The content of the file is displayed in the main pane:

```
1 <@{  
2     ViewBag.Title = "Home Page";  
3 }  
4  
5 <div class="jumbotron">  
6     <h1>ASP.NET  
7     <p class="lead">ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

8     <a href="http://asp.net" class="btn btn-primary btn-lg">Learn more &gt;</p>  
9 </div>  
10  
11 <div class="row">  
12     <div class="col-md-4">  
13         <h2>Getting started:  
14         <p>ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that  
15             enables a clean separation of concerns and gives you full control over markup


```

#### 4. 找到以下代码：

## HTML 复制

## ASP.NET

#### 5. 将该代码替换为此代码：

## HTML 复制

# Web App Version 2

6. 若要保存更改, 请按 **Ctrl+S**。

```
1  @{
2      ViewBag.Title = "Home Page";
3  }
4
5  <div class="jumbotron">
6      <h1>Web App Version 2</h1>
7      <p class="lead">ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.</p>
8      <p><a href="http://asp.net" class="btn btn-primary btn-lg">Learn more &raquo;</a></p>
9  </div>
10
11 <div class="row">
12     <div class="col-md-4">
13         <h2>Getting started</h2>
14         <p>
15             ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that
16             enables a clean separation of concerns and gives you full control over markup
```

7. 在 Cloud Shell 中，输入以下命令以将新版应用提交到 Git 并将其部署到过渡槽。根据提示输入部署密码。

## bash 复制

```
git add .
```

```
git commit -m "New version of web app."
```

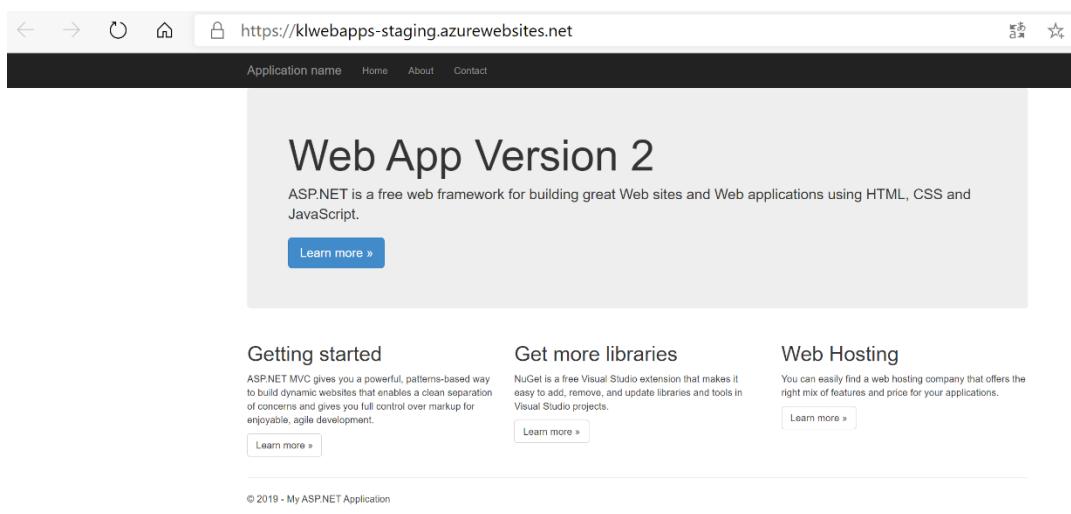
```
git push staging
```

```
PowerShell v | ⌂ ? ⚙️ ↻ ⌂ {} ⌂
kelvin@Azure:~/demoapp/app-service-web-dotnet-get-started$ git add .
kelvin@Azure:~/demoapp/app-service-web-dotnet-get-started$ git commit -m "New version of web app."
[master 363d927] New version of web app.
 1 file changed, 2 insertions(+), 2 deletions(-)
```

## 浏览过渡槽

现在，你可以通过浏览到过渡部署槽的 URL 来查看新版 Web 应用。

在 Azure 门户中，转到过渡槽的“概述”页。 在页面顶部附近，选择“浏览”按钮。 浏览器选项卡中会显示新版 Web 应用：



此时，过渡槽具有新版代码，可以对其进行运行测试。 请记住，生产槽位具有以前的 Web 应用版本，用户还不能看到新代码。

## 通过使用部署槽位来部署 Web 应用

准备好交换两个槽时，请确保已将正确的配置应用于已交换的槽。

假设你已完成测试社交媒体 Web 应用的版本 2。 现在，想要将该版本部署到生产环境。 同时希望通过自动交换应用的未来版本来进一步简化部署。

此处介绍如何手动和自动交换。

## 配置槽设置

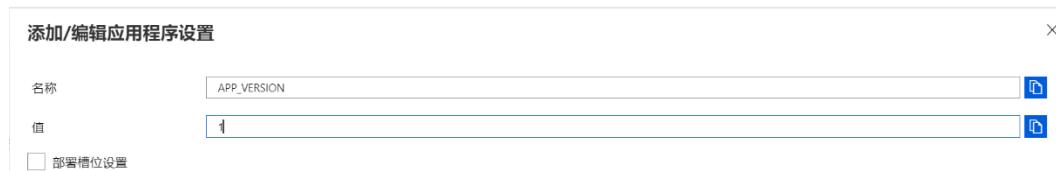
在部署 Web 应用版本 2 之前，先配置槽设置。 此处将要配置的设置不会对演示应用产生影响。 本练习的目的仅在于了解在交换槽时的配置的工作方式。

若要配置槽设置，请执行以下操作：

1. 从 Azure 门户中的“所有资源”视图导航到 Web 应用生产槽位的“概述”页。
2. 导航到部署槽位的“配置”页。
3. 选择“新应用程序设置”。 添加名为“ENVIRONMENT\_NAME”且值为“production”的新设置。 选中“部署槽位设置”框使其成为槽位设置。



4. 添加名为“APP\_VERSION”的另一个设置并输入值“1”。 不要使其成为槽设置。



5. 在页面顶部附近，选择“保存”按钮。

- 在过渡槽上重复上述步骤，但使用以下值：

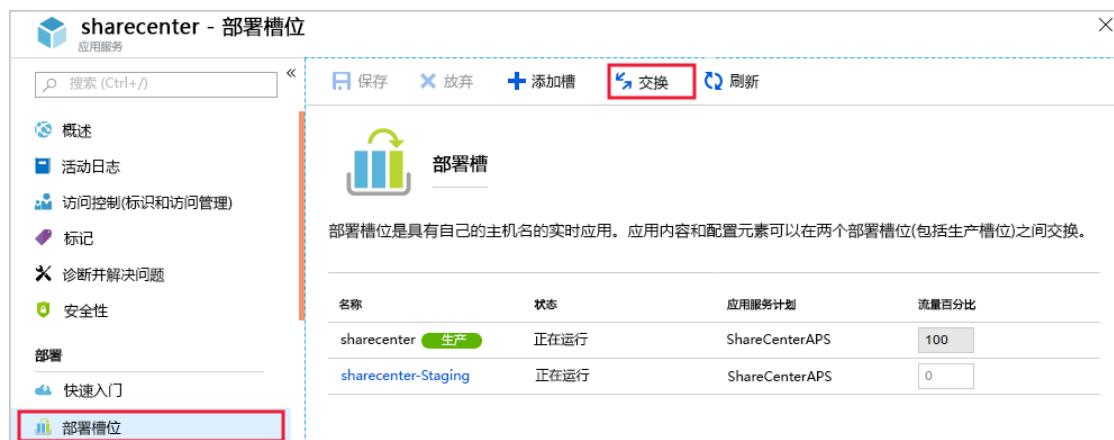
名称	值	部署槽位
ENVIRONMENT_NAME	staging	是
APP_VERSION	2	否



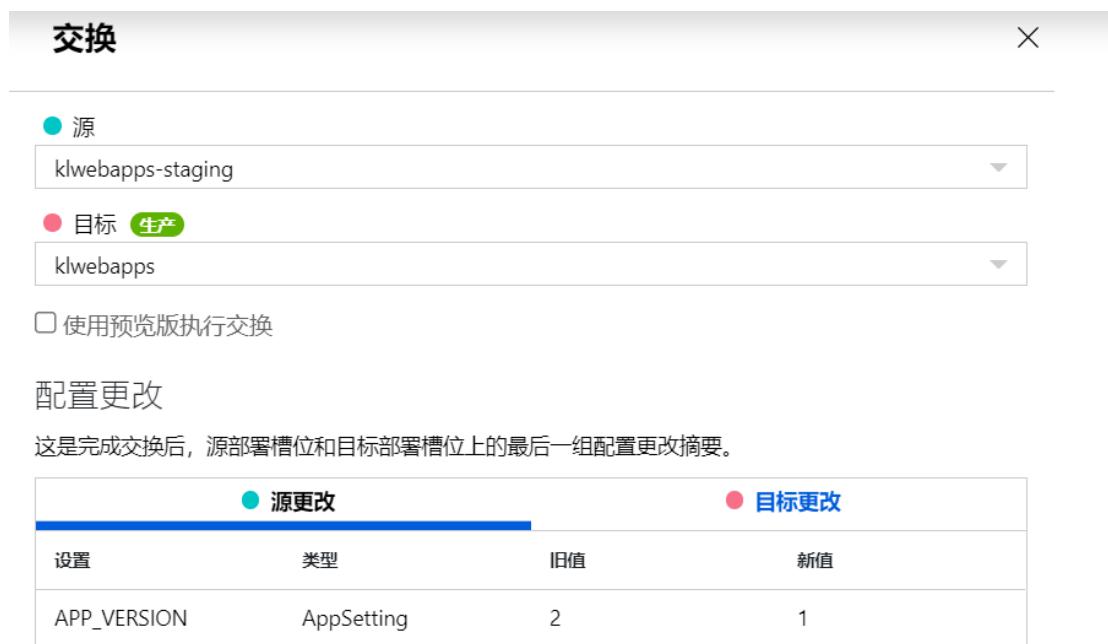
## 交换槽

现在已在过渡槽中对 Web 应用版本 2 进行了测试，可以通过交换槽来部署该应用。执行以下步骤：

- 若要确保配置的是生产槽，请选择“所有资源”，然后选择 Web 应用的生产槽。
- 在“部署”下，选择“部署槽位”>“交换”。



3. 请确保即将交换过渡槽和生产槽。请注意交换将如何影响设置。将在槽之间交换 APP\_VERSION 设置的值，但不会交换 ENVIRONMENT 槽设置的值。选择“交换”。



4. 完成交换后，转到生产槽 Web 应用的“概述”页，然后选择“浏览”。此时会在新的浏览器选项卡中显示 Web 应用。请注意，Web 应用版本 2 现已在生产环境中。
5. 关闭浏览器选项卡。

## 为过渡槽配置自动交换

假设现在正在使用部署槽位，并需要启用持续部署。可以使用 Web 应用的自动交换功能来完成此操作。在使用自动交换的系统中，将新代码部署到过渡槽时，Azure 会自动对其进行预热，并通过交换过渡槽和生产槽将其部署到生产环境。若要配置自动交换，请按照下列步骤进行操作：

1. 转到过渡槽位 Web 应用的“配置”页，然后导航到“常规设置”选项卡。

staging (klwebapps/staging) - 配置

堆栈设置

堆栈: .NET  
.NET Framework 版本: V4.7

平台设置

平台: 32 Bit  
托管管道版本: 集成  
FTP 状态: 全部允许  
HTTP 版本: 2.0  
Web 套接字:  开  关  
始终可用:  开  关  
ARR 相关性:  开  关

调试

远程调试:  开  关

2. 将“自动交换已启用”设置为“打开”。
3. 在“自动交换部署槽位”列表中，选择“production”，然后选择“保存”。

部署

平台: 32 Bit  
托管管道版本: 集成  
FTP 状态: 全部允许  
HTTP 版本: 2.0  
Web 套接字:  开  关  
始终可用:  开  关  
ARR 相关性:  开  关

调试

远程调试:  开  关

部署槽位

自动交换已启用:  开  关  
自动交换部署槽位: production

## 部署新代码并将其自动交换到生产环境

现在将修改代码，以创建 Web 应用版本 3。将其部署到过渡槽时，会看到正在运行自动交换。请执行以下步骤：

1. 如果尚未运行，请在 Cloud Shell 窗口的右侧重启编辑器。

bash 复制

```
cd ~/demoapp/app-service-web-dotnet-get-started/  
code .
```

2. 在代码编辑器左侧的“文件列表”中，展开“aspnet-get-started”>“视图”>“主页”。然后选择“Index.cshtml”。
3. 找到以下代码：

HTML 复制

```
<h1>Web App Version 2</h1>
```

4. 将该代码替换为此代码：
5. 若要保存更改，请按 Ctrl+S。
6. 在 Cloud Shell 中，输入以下命令。根据提示输入部署密码。

bash 复制

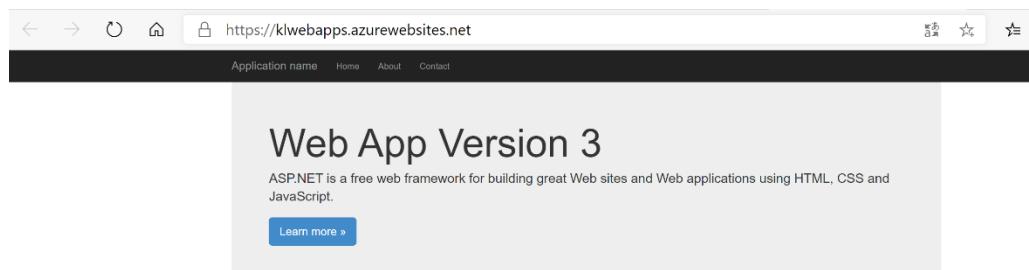
```
git add .  
git commit -m "Third version of web app."  
git push staging
```

等待部署完成。即将完成文本输出时，你会看到一条消息，指示部署已请求自动交换到生产槽。

```
'current' instead of 'simple' if you sometimes use older versions of Git)

Password for 'https://kladministrator@klwebapps-staging.scm.azurewebsites.net:443':
Counting objects: 12, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 952 bytes | 0 bytes/s, done.
Total 12 (delta 10), reused 0 (delta 0)
remote: Updating branch 'master'.
remote: Updating submodules.
remote: Preparing deployment for commit id '47a555c3e5'.
remote: Generating deployment script.
remote: Running deployment command...
remote: Handling .NET Web Application deployment.
remote: MSBuild auto-detection: using msbuild version '14.0.23107.0' built by: DI4REL' from 'D:\Program Files (x86)\MSBuild\14.0\Bin'.
remote: All packages listed in packages.config are already installed.
remote: aspnet-get-started -> D:\home\site\repository\aspnet-get-started\bin\aspnet-get-started.dll
remote: Transformed Web.config using D:\home\site\repository\aspnet-get-started\Web.Release.config into obj\Release\TransformWebConfig\transformed\Web.config.
remote: Copying all files to temporary location below for package/publish:
remote: D:\LocalTemp\8d76ff12eac2d7a.
remote: Creating app offline.htm
remote: KuduSync.NET from: 'D:\local\Temp\8d76ff12eac2d7a' to: 'D:\home\site\wwwroot'
remote: Copying file: 'Views\Home\Index.cshtml'
remote: Deleting app_offline.htm
remote: Finished successfully.
remote: Running post deployment command(s)...
remote: Triggering recycle (preview mode disabled).
remote: Deployment successful.
remote: Requesting auto swap to 'production' slot with 'AUTOSWAP3751324b-c467-4d1b-bd31-80c68d54524f' id successful.
To https://kladministrator@klwebapps-staging.scm.azurewebsites.net:443/klwebapps.git
 24119e7..47a555c master --> master
```

7. 在 Azure 门户中，导航到生产槽 Web 应用的“概述”页，然后选择“浏览”。此时会在新的浏览器选项卡中显示 Web 应用的第三版。如果显示的是旧版本，可能需要稍等片刻，然后刷新页面 - 交换操作是原子操作，会立即发生，但在执行之前，应用服务需要一些时间准备交换操作。



回滚新版本

假设将应用版本 3 部署到生产环境出现了意外问题。 若要快速解决该问题，可以通过再次交换槽来回滚到站点的以前版本。

1. 转到生产槽的 Web 应用的“部署槽位”页面。
  2. 交换过渡槽和生产槽。
  3. 完成交换后，选择“概述”页上的“浏览”按钮以最后一次查看应用。你将看到版本 2 已重新部署到生产环境。

## 二、配置应用服务进行纵向和横向扩展来满足 Web 应用的需求变化

### 手动缩放 Web 应用

你应该在预期流量增加时横向扩展系统。 也可以针对性能下降进行横向扩展。

请记住，在酒店预订系统示例中，当你预计会因为特殊事件、特殊优惠或季节性波动而带来额外流量时，你会增加 Web 应用的实例数。 当需求下降时，你会缩减系统。

本练习中将使用此计划创建应用服务计划并部署 Web 应用。 你将在负载下监视 Web 应用的性能。 然后，你将横向扩展应用，验证其性能是否已得到提升。

本练习使用实现 Web API 的示例 Web 应用。 Web API 公开 HTTP POST 和 GET 操作，这些操作用于创建和检索酒店预订网站的客户预订。 系统实际上不会保存预订，且 Get 操作只检索虚拟数据。

练习还会运行一个客户端应用，模拟许多用户同时发出 POST 和 GET 操作。此应用提供工作负载，用于测试 Web 应用在缩放之前和之后的性能。

### 创建应用服务计划和 Web 应用

#### 重要

需使用自己的 Azure 来运行此练习，可能会产生费用。如果还没有 Azure 订阅，请创建[免费帐户](#)。

1. 登录 [Azure 门户](#)。
2. 选择“创建资源” > “Web” > “Web 应用”。
3. 在“Web 应用”页面上，输入下表中的值。

#### 备注

Web 应用必须具有唯一名称。 建议使用类似于<姓名或姓名缩写>hotelsystem 的名称。 在本练习中看见 <your-webapp-name> 时则使用此名称。

属性	值
应用名称	<your-webapp-name>
订阅	选择要用于此练习的 Azure 订阅
资源组	创建一个名为“mslearn-scale”的新资源组
OS	Windows
发布	代码
应用服务计划/位置	保留默认值
运行时堆栈	ASP.NET V4.7

4. 选择“创建”，然后等待 Web 应用完成创建。

Web 应用

资源组

实例详细信息

名称  .azurewebsites.net

发布  代码  Docker 容器

运行时堆栈

操作系统  Windows  Linux

区域  找不到应用服务计划? 请尝试其他区域。

应用服务计划

应用服务计划定价层确定与你的应用相关的位置、功能、成本和计算资源。 [了解详细信息](#)

Windows 计划 (East Asia)

SKU 和大小  100 总 ACU, 1.75 GB 内存

---

## 生成并部署 Web 应用

1. 在 Azure 门户中打开 Cloud Shell。 运行此命令，下载酒店预订系统的源代码：

```
bash 复制  
git clone https://github.com/MicrosoftDocs/mslearn-hotel-reservation-system.git
```

2. 转到“mslearn-hotel-reservation-system/src”文件夹：

```
bash 复制  
cd mslearn-hotel-reservation-system/src
```

3. 生成酒店系统的应用。 有两个应用：实现系统 Web API 的 Web 应用，以及将用于负载测试 Web 应用的客户端应用。

```
bash 复制  
dotnet build
```

```

PowerShell | ⌁ ? { } ⌂
Azure:/ ES Azure:> bash
kelvin@Azure:~$ git clone https://github.com/MicrosoftDocs/mslearn-hotel-reservation-system.git
Cloning into 'mslearn-hotel-reservation-system'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 33 (delta 5), reused 22 (delta 3), pack-reused 0
Unpacking objects: 100% (33/33), done.
Checking connectivity... done.
kelvin@Azure:~$ cd mslearn-hotel-reservation-system/src
kelvin@Azure:~/mslearn-hotel-reservation-system/src$ dotnet build

```

#### 4. 做好发布 HotelReservationSystem Web 应用的准备:

bash 复制

```

cd HotelReservationSystem
dotnet publish -o website

```

```

kelvin@Azure:~/mslearn-hotel-reservation-system/src$ ls
HotelReservationSystem HotelReservationSystem.sln HotelReservationSystemTestClient HotelReservationSystemTypes
kelvin@Azure:~/mslearn-hotel-reservation-system/src$ cd HotelReservationSystem
kelvin@Azure:~/mslearn-hotel-reservation-system/src/HotelReservationSystem$ dotnet publish -o website
Microsoft (R) Build Engine version 16.2.32702+4012a063 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

Restore completed in 50.12 ms for /home/kelvin/mslearn-hotel-reservation-system/src/HotelReservationSystem/HotelReservationSystem.csproj.
Restore completed in 0.55 ms for /home/kelvin/mslearn-hotel-reservation-system/src/HotelReservationSystemTypes/HotelReservationSystemTypes.csproj.
HotelReservationSystemTypes -> /home/kelvin/mslearn-hotel-reservation-system/src/HotelReservationSystemTypes/bin/Debug/netcoreapp2.1/HotelReservationSystemTypes.dll
HotelReservationSystem -> /home/kelvin/mslearn-hotel-reservation-system/src/HotelReservationSystem/bin/Debug/netcoreapp2.1/HotelReservationSystem.dll
HotelReservationSystem -> /home/kelvin/mslearn-hotel-reservation-system/src/HotelReservationSystem/website/
kelvin@Azure:~/mslearn-hotel-reservation-system/src/HotelReservationSystem$ 

```

#### 5. 转到“网站”文件夹，其中包含已发布的文件。压缩文件并将其部署到在

上一个任务中创建的 Web 应用中。将 <your-webapp-name> 替换为 Web 应用的名称。

bash 复制

```

cd website
zip website.zip *
az webapp deployment source config-zip --src website.zip -
-name <your-webapp-name> --resource-group mslearn-scale

```

```

kelvin@Azure:~/mslearn-hotel-reservation-system/src/HotelReservationSystem/website$ az webapp deployment source config-zip --src website.zip --name khotelscale --resource-group labs
Getting scm site credentials for zip deployment
Starting zip deployment. This operation can take a while to complete ...
Deployment endpoint responses with status code 202
{
  "active": true,
  "author": "N/A",
  "author_email": "N/A",
  "complete": true,
  "deployer": "Push-Deployer",
  "end_time": "2019-11-26T08:15:32.858274Z",
  "id": "7146439a43854432ac018dc8377040f6",
  "is_readonly": true,
  "is_temp": false,
  "last_success_end_time": "2019-11-26T08:15:32.858274Z",
  "log_url": "https://khotelscale.scm.azurewebsites.net/api/deployments/latest/log",
  "message": "Created via a push deployment",
  "progress": ""
}

```

- 使用 Web 浏览器转到 `https://<your-webapp-name>.azurewebsites.net/api/reservations/1`。 应该会看到一个 JSON 文档，其中包含编号为 1 的预订的详细信息：



## 在横向扩展之前监视 Web 应用的性能

- 返回到 Cloud Shell，并转到“~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient”文件夹：

```

bash 复制
cd ~/mslearn-hotel-reservation-
system/src/HotelReservationSystemTestClient

```

- 使用代码编辑器编辑此文件夹中的 App.config 文件：

```

bash 复制
code App.config

```

- 取消注释 ReservationsServiceURI 的行（! --），并使用 Web 应用的 URL 替换值。 文件应如下例所示：

```

text 复制
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>

```

```
        <add key="NumClients" value="100" />
        <add key="ReservationsServiceURI"
value="https://<your-webapp-name>.azurewebsites.net/">
            <add key="ReservationsServiceCollection"
value="api/reservations"/>
        </appSettings>
</configuration>
```

备注

此文件中的 `NumClients` 设置指定将同时尝试连接到 Web 应用并执行工作的客户端数。工作包括创建预留，然后运行查询以提取预订的详细信息。使用的所有数据都是虚假的。事实上，这些数据不存在于任何地方。将此值设置为 `100`。

4. 保存文件并关闭代码编辑器。

```
PowerShell ▾ | ⌁ ? ⚙️ 📂 ⚡ { } 🔍 App.config •  
1 <?xml version="1.0" encoding="utf-8" ?>  
2 <configuration>  
3   <appSettings>  
4     <add key="NumClients" value="100" />  
5     <add key="ReservationsServiceURI" value="https://klhotelbooking.azurewebsites.net/" />  
6     <add key="ReservationsServiceCollection" value="api/reservations" />  
7   </appSettings>  
8 </configuration>
```

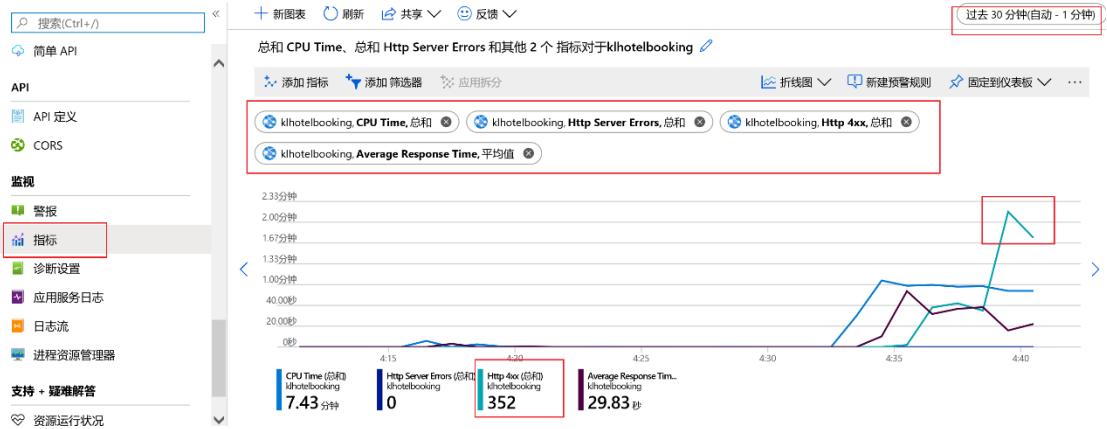
- ## 5. 使用新配置重新生成测试客户端应用:

bash 复制  
dotnet build

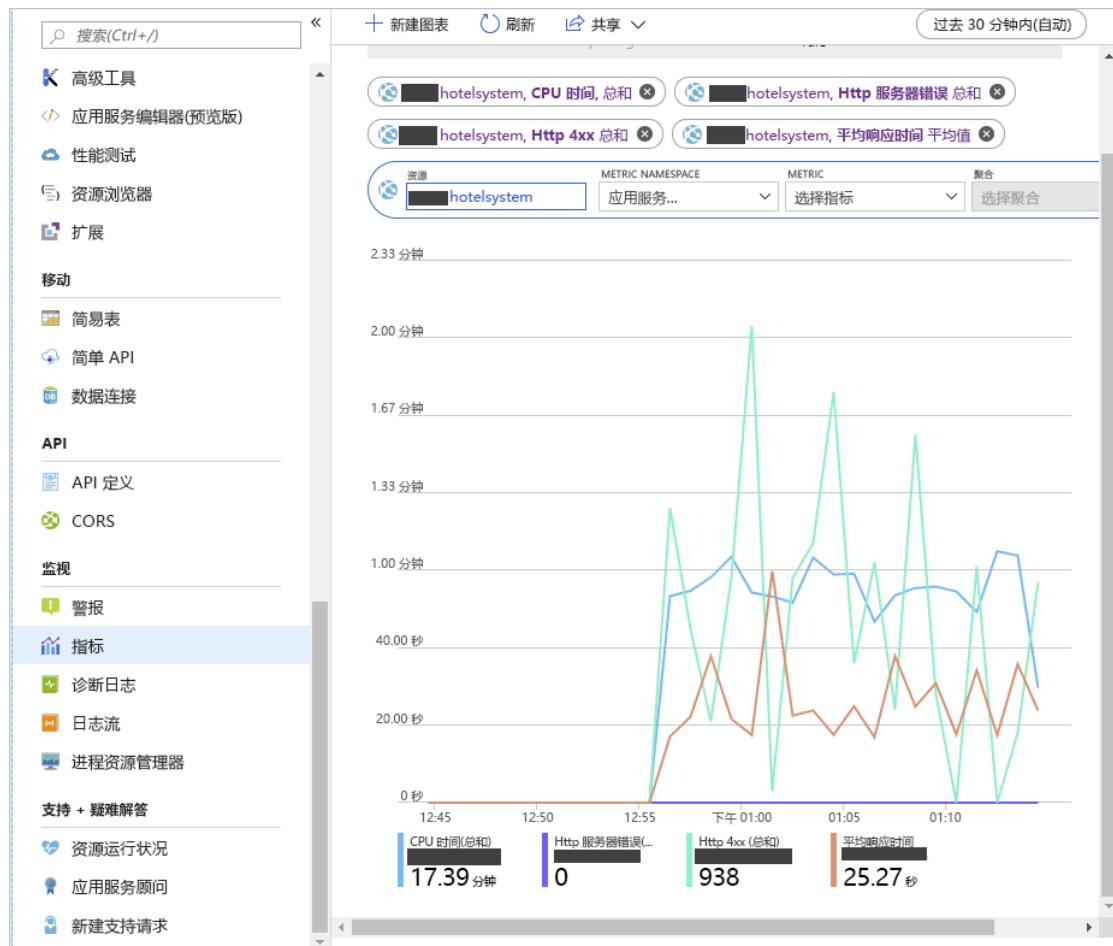
6. 运行客户端应用。当客户端开始运行、创建预订和运行查询时，你将看见显示许多条消息。让系统运行几分钟。响应将会很慢，并且很快客户端请求就会失败并显示 HTTP 408（超时）错误。

bash 复制  
dotnet run

7. 在 Azure 门户中，转到 Web 应用（不是服务计划）的边栏选项卡。在“监视”下，选择“指标”。
  8. 将以下指标添加到图表，并将时间范围设置为“过去 30 分钟”，然后将图表固定到当前仪表板。
    - CPU 时间。 选择 Sum 聚合。
    - Http 服务器错误。 选择 Sum 聚合。
    - Http 4xx。 选择 Sum 聚合。
    - 平均响应时间。 选择 Avg 聚合。



9. 使系统运行五分钟以达到稳定状态，然后记下 CPU 时间、HTTP 4xx 错误数以及平均响应时间。你应该会看到大量 HTTP 4xx 错误（这些是 HTTP 408 超时错误），且平均响应时间为几秒。可能偶尔发生 HTTP 服务器错误，具体取决于 Web 服务器如何处理负担。



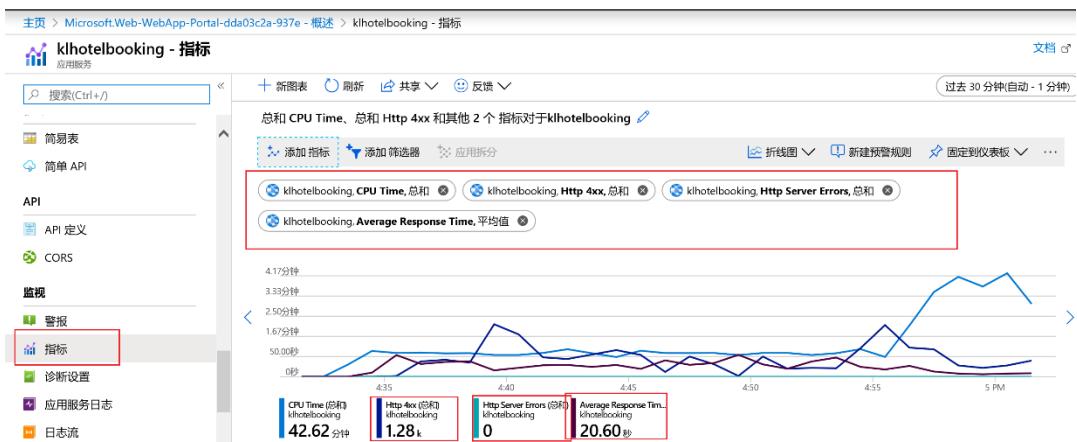
10. 执行下一个任务时，请保持客户端应用正常运行。

## 横向扩展 Web 应用并验证性能改进

1. 在 Azure 门户的 Web 应用的边栏选项卡中，在“设置”下，选择“扩大(应用服务计划)”。  
2. 在“配置”页面上，将“实例计数”设置为“5”，然后选择“保存”。

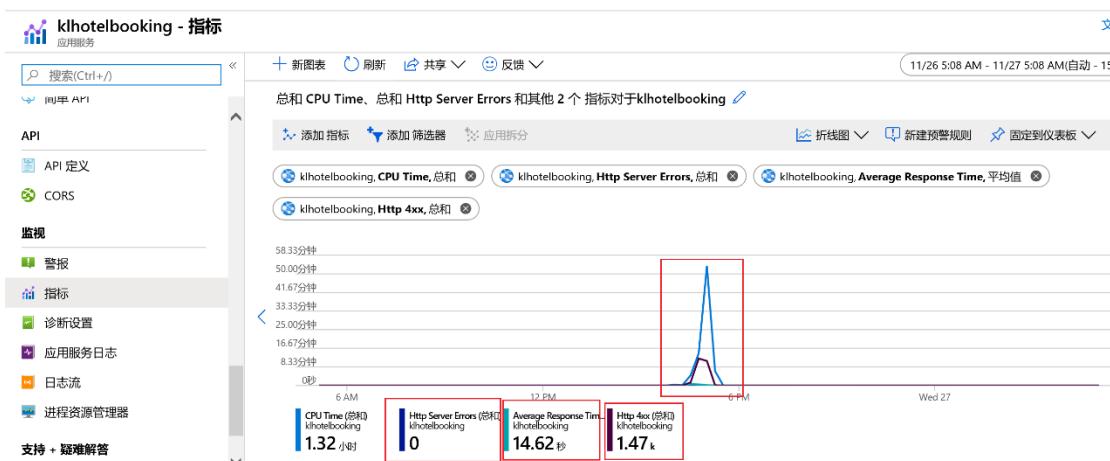


3. 切换到运行客户端应用的 Cloud Shell。 尽管仍有一些超时，但应看到错误请求数较少。  
4. 再运行该应用五分钟。 然后转到 Azure 门户中的仪表板上显示应用指标的图表。 你应看到 CPU 时间大幅增加，因为现在可用的 CPU 计算能力是原来的五倍。 平均响应时间应已减少，HTTP 4xx 错误的数量也应减少。 下图显示了一组典型结果。 记录了横向扩展发生的时间点。



5. 如果想进行更多试验，请尝试将应用服务计划的实例数增加到 10。 S1

层支持的最大实例数为 10。你应该会注意到 CPU 时间在进一步增加，并且相应的响应时间和 HTTP 4xx 错误在减少。



6. 返回到运行客户端应用的 Cloud Shell。选择 Enter 停止应用。

7. 在 Azure 门户中，将应用服务计划的实例数设置为 1。

## 纵向扩展 Web 应用

纵向扩展为运行 Web 应用提供了更强大的资源。它还增加了可用于横向扩展的实例数。

在酒店预订系统中，应横向扩展以处理越来越多的 Web 应用访客。通过纵向扩展可以实现进一步的横向扩展。为了支持将添加到 Web 应用的新功能，也可能需要纵向扩展。

在本练习中，你将纵向扩展先前部署的酒店预订系统 Web 应用。运行之前使用的相同测试客户端应用程序，并监视 Web 应用的性能。

## 检查 Web 应用的当前定价层

### 重要

需使用自己的 Azure 来运行此练习，可能会产生费用。如果还没有 Azure 订阅，请创建[免费帐户](#)。

1. 登录 [Azure 门户](#)。
2. 选择“所有资源”，然后转到“应用服务计划”。
3. 在“设置”下，选择“纵向扩展(应用服务计划)”。应会看到应用服务计划定价层的详细信息。定价层为 S1，提供可在 A 系列虚拟机上运行的 100 个 Azure 计算单位和 1.75 GB 内存。

计划	ACU	内存	描述
<b>S1</b>	总共 100 个 ACU	1.75 GB 内存	A 序列计算等效项 55.43 GBP/月(估计)
<b>P1V2</b>	总共 210 个 ACU	3.5 GB 内存	Dv2 序列计算等效项 110.86 GBP/月(估计)
<b>P2V2</b>	总共 420 个 ACU	7 GB 内存	Dv2 序列计算等效项 221.79 GBP/月(估计)
<b>P3V2</b>	总共 840 个 ACU	14 GB 内存	Dv2 序列计算等效项 443.57 GBP/月(估计)

**包含的功能**  
此应用服务计划中托管的每个应用均有权访问这些功能:

- 自定义域/SSL 配置和购买具有 SNI 和 IP SSL 绑定的自定义域
- 自动缩放 最多 10 个实例。取决于可用性。
- 过渡槽 在交换过渡槽用于生产之前，最多可将 5 个过渡槽用于测试和部署。
- 每日备份 每天备份 10 次应用。

**包括的硬件**  
应用服务计划中的每个实例均包括以下硬件配置:

- Azure 计算单元(ACU) 用于运行应用服务计划中部署的应用程序的专用计算资源。[了解详细信息](#)
- 内存 每个实例可用于运行在应用服务计划中部署和运行的应用程序的内存。
- 存储 应用服务计划中部署的所有应用共享的 50 GB 磁盘存储。

## 运行测试客户端应用

1. 在屏幕右侧的 Cloud Shell 窗口中，转到“~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient”文件夹:

```
bash 复制
cd ~/mslearn-hotel-reservation-
system/src/HotelReservationSystemTestClient
```

2. 运行客户端应用。让系统运行几分钟。与前一个练习开始时一样，响应将会很慢，并且客户端请求很快就会失败并显示 HTTP 408（超时）错误。

```
bash 复制
dotnet run
```

- 继续运行应用。再等五分钟。然后转到 Azure 门户中的仪表板上显示 Web 应用指标的图表。与前一个练习一样，你应该会看到统计数据表明响应时间相对较慢，且存在许多 HTTP 4xx 错误。

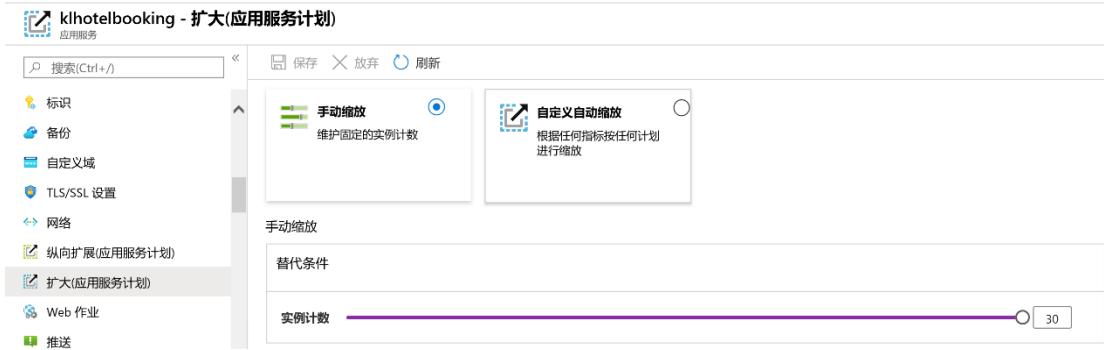


## 纵向扩展 Web 应用并监视结果

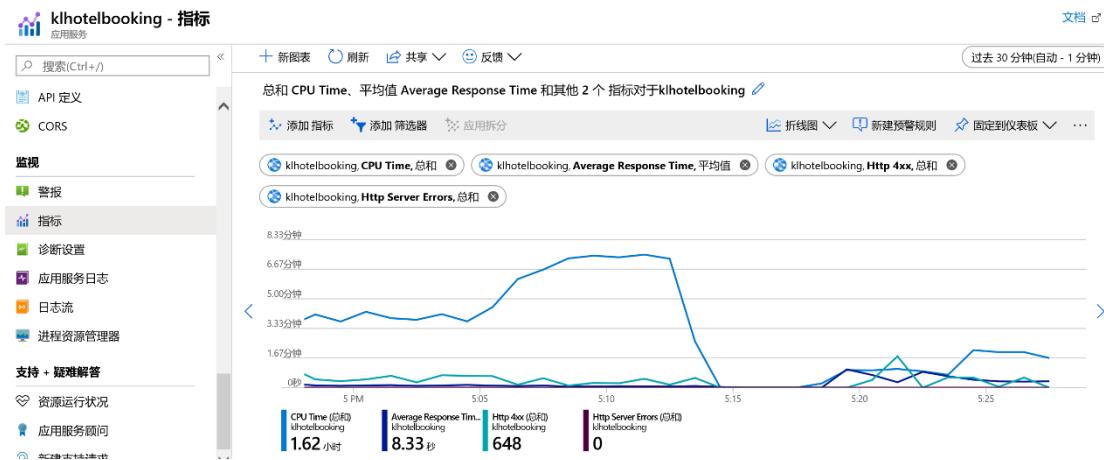
1. 在 Azure 门户中，返回到应用服务计划页面。
  2. 在“设置”下，选择“纵向扩展(应用服务计划)”。
  3. 选择“P2V2”定价层，然后选择“应用”。此定价层提供在 Dv2 系列 VM 上运行的 420 个 ACU（超过 S1 定价层数量的四倍）和 7 GB 内存。

但是，此 VM 的成本是运行 S1 定价层的四倍。
  4. 再等五分钟，然后在 Azure 门户中的仪表板上查看性能图表。
  5. 在系统进行纵向扩展时，你可能会注意到一些其他 HTTP 服务器错误。这些错误是由正在进行的客户端请求在系统切换硬件时被中止引起的。纵向扩展后，由于可用的处理器更多，CPU 时间会迅速增加。你可能不会看到在横向扩展时看到的相同响应时间下降。这是因为你仍然只使用单

个实例，请求不会像横向扩展时那样进行负载平衡。但现在可横向扩展到比以前更多的实例（20 个）。



此图中的图表显示了 Web 应用的性能指标示例。记录了系统纵向扩展发生的时间点。



6. 返回到运行客户端应用的 Cloud Shell。选择 Enter 停止应用。

### 三、配置 Azure 应用服务部署和运行容器化 Web 应用

#### 通过使用 Azure 容器注册表来生成和存储映像

Azure 容器注册表为云中的 Docker 映像提供存储。

在示例场景中，团队需要为其 Web 应用创建用于存储映像的注册表。

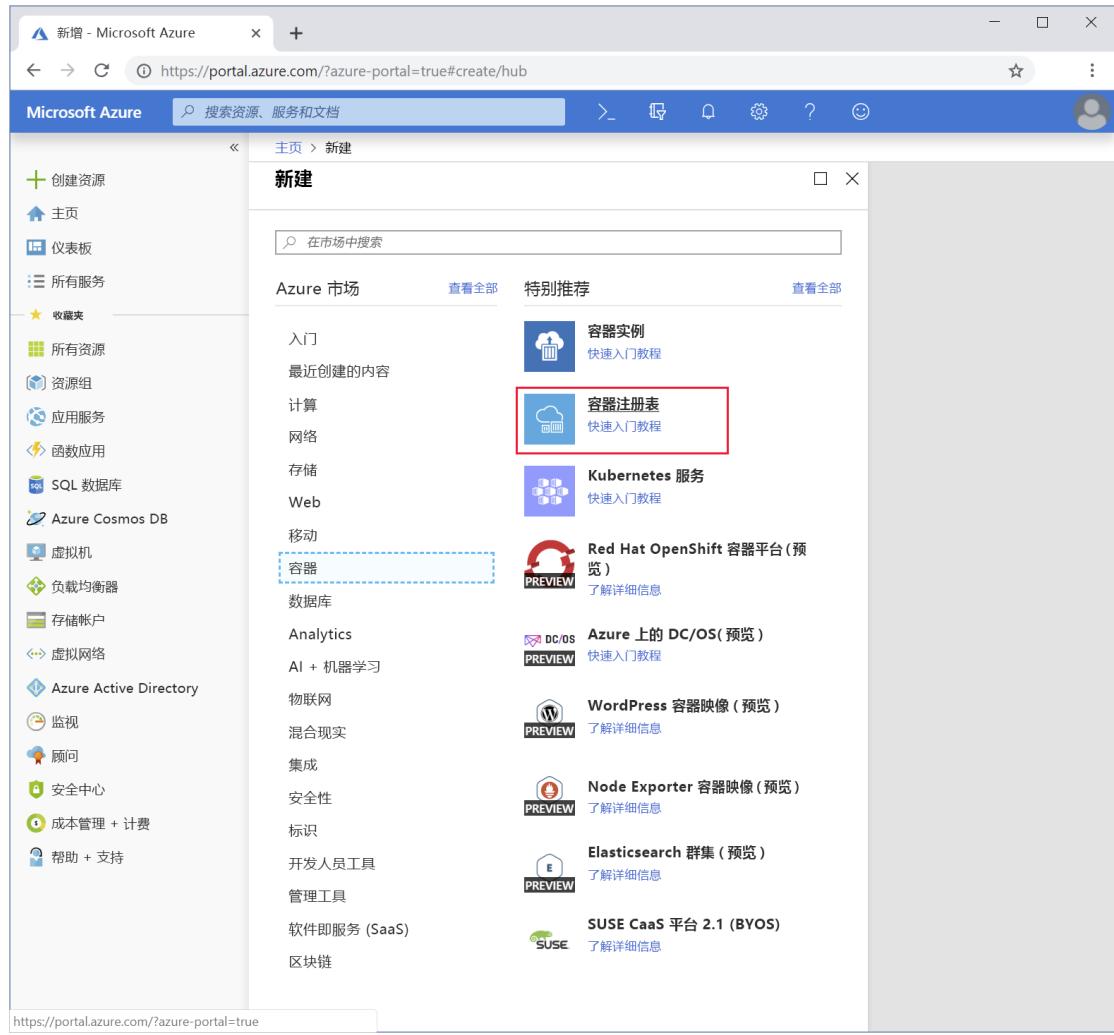
在本单元中，你将使用 Azure 门户在 Azure 容器注册表中创建新注册表。你将从 Web 应用的源代码生成一个 Docker 映像并将其上传到注册表中的存储库。最后，你将检查注册表和存储库的内容。

##### 重要

需使用自己的 Azure 来运行此练习，可能会产生费用。如果还没有 Azure 订阅，请创建[免费帐户](#)。

#### 在 Azure 容器注册表中创建注册表

1. 使用 Azure 订阅登录[Azure 门户](#)。
2. 依次选择“创建资源”、“容器”、“容器注册表”。



### 3. 为每个属性指定下表中的值：

属性	值
注册表名称	输入唯一的名称，并记下它以供日后使用。
订阅	选择你可以在其中创建和管理资源默认 Azure 订阅。
资源组	创建名为“learn-deploy-container-acr-rg”的新资源组，以便在完成该模块后，可以选择其他资源组名称，请记住它，以便在本模块的其余练习中使用。
位置	请选择离你近的位置。
管理员	启用

属性	值
用户	
SKU	标准

- 单击“创建”。 等待容器注册表创建完成，然后继续。

**创建容器注册表** □ ×

---

**注册表名称 \***  
 ✓  
.azurecr.io

**订阅 \***  
 ▼

**资源组 \***  
 ▼  
[新建](#)

**位置 \***  
 ▼

**管理员用户 \*** ①  
启用 禁用

**SKU \*** ①  
 ▼

## 生成一个 Docker 映像并将其上传到 Azure 容器注册表

- 在门户中的 Azure Cloud Shell 窗口中，运行以下命令以下载示例 Web 应用的源代码。 此 Web 应用非常简单。 它提供了一个包含静态文本的页面以及可旋转一系列映像的旋转木马控件。

```
bash 复制
git clone https://github.com/MicrosoftDocs/mslearn-deploy-run-container-app-service.git
```

## 2. 转到源文件夹:

## bash 复制

```
cd mslearn-deploy-run-container-app-service/dotnet
```

3. 运行以下命令。此命令会将文件夹的内容发送到 Azure 容器注册表，该注册表使用 Docker 文件中的说明生成映像并将其存储。

将 `<container_registry_name>` 替换为之前创建的注册表的名称。请注意，不要省略命令末尾的`.`字符。

## bash 复制

```
az acr build --registry <container_registry_name> --image  
webimage .
```

Docker 文件包含从 Web 应用的源代码生成 Docker 映像的分步说明。

Azure 容器注册表运行这些步骤来构建映像，并在每个步骤完成时生成一条消息。构建应在几分钟后完成，并且不显示任何错误或警告。

```
PowerShell ▾ | ⌁ ? ⌂ ⌃ ⌄ ⌅ ⌆
runtime-dependency:
  registry: registry.hub.docker.com
  repository: microsoft/dotnet
  tag: 2.1-aspnetcore-runtime
  digest: sha256:404bb110bd3f39457971ad2134f56687e27f7b1ff396d06158ab02ab3154a28b
buildtime-dependency:
- registry: registry.hub.docker.com
  repository: microsoft/dotnet
  tag: 2.1-sdk
  digest: sha256:90f1a964a6bfd7c7b4743e99d4d1559a14a9f2a7b6c7bdb5a8a7b8a5a28ff24c
git: {}

Run ID: ck1 was successful after 1m47s
kelvin@Azure:~/mslearn-deploy-run-container-app-service/dotnet$ █
```

## 检查容器注册表

1. 在 [Azure 门户](#) 中，导航到容器注册表的“概述”页。
  2. 在“服务”下，选择“存储库”。你将看到名为 `webimage` 的存储库。
  3. 选择 `webimage` 存储库。它包含带有 `latest` 标记的映像。这是示例 Web 应用的 Docker 映像。

The screenshot shows the 'kllabregistry - 存储库' (Storage) page. On the left, there's a sidebar with navigation links like '概述' (Overview), '活动日志' (Activity Log), '访问控制(标识和访问管理)' (Access Control), '标记' (Tags), '快速入门' (Quick Start), '事件' (Events), '设置' (Settings), '访问密钥' (Access Keys), '防火墙和虚拟网络(预览版)' (Firewall and Virtual Networks (Preview)), '锁' (Lock), and '导出模板' (Export Templates). The main area displays the 'webimage' storage library, which has a count of 1 item and was last updated on 2019/11/28 at 10:30 AM. A search bar at the top right allows filtering by storage library name.

# 基于 Docker 映像创建和部署 Web 应用

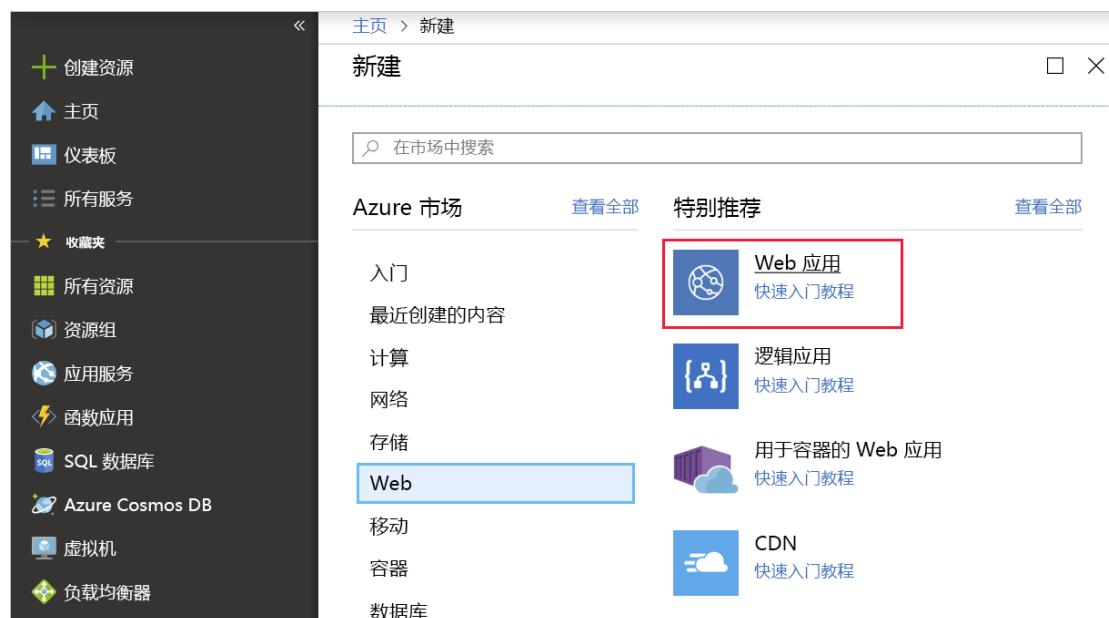
Azure 应用服务为基于 Azure 的 Web 应用提供托管环境。 可以配置应用服务以从 Azure 容器注册表中的存储库检索 Web 应用的映像。

在示例场景中，团队已将 Web 应用的映像上传到 Azure 容器注册表并且现在可以部署 Web 应用。

在本单元中，你将通过使用 Azure 容器注册表中存储的 Docker 映像来创建新的 Web 应用。 你将使用包含预定义的应用服务计划的应用服务来托管 Web 应用。

## 创建 Web 应用

1. 登录 [Azure 门户](#)。
2. 选择“创建资源”>“Web”>“Web 应用”。



3. 为每个属性指定这些设置：

属性	值
订阅	选择你可以在其中创建和管理资源默认 Azure 订阅。
资源组	重用现有资源组 learn-deploy-container-acr-rg。
名称	输入唯一的名称，并记下它以供日后使用。
发布	<b>Docker 映像</b>
OS	<b>Linux</b>
应用服务计划	使用默认值。

## Web 应用

订阅 \* ⓘ Microsoft Azure 内部消耗

资源组 \* ⓘ labs 新建

实例详细信息

名称 \* kldockerapps .azurewebsites.net

发布 \* 代码 Docker 容器

操作系统 \* Linux Windows

区域 \* East Asia ⚡ 找不到应用服务计划? 请尝试其他区域。

应用服务计划

应用服务计划定价层确定与你的应用相关的位置、功能、成本和计算资源。 [了解详细信息](#)

Linux 计划 (East Asia) \* ⓘ (新项) ASP-labs-a106 新建

SKU 和大小 \* 高级 V2 P1v2  
210 总 ACU, 3.5 GB 内存 更改大小

4. 单击“Docker >”。

5. 在“Docker”选项卡中，为每个属性指定这些设置：

属性	值
选项	单个容器

属性	值
映像源	Azure 容器注册表
注册表	选择你的注册表。
映像	webimage
标记	latest
启动命令	将此项留空。

6. 选择“查看和创建”，然后单击“创建”。等到 Web 应用部署完毕后再继续操作。

## Web 应用

基本 Docker 监视 标记 查看 + 创建

从 Azure 容器注册表、Docker Hub 或专用 Docker 存储库中拉取容器映像。应用服务几秒内即可将具有首选依赖项的容器化应用部署到生产中。

选项	单个容器
映像源	Azure 容器注册表
<b>Azure 容器注册表选项</b>	
注册表 *	kllabregistry
图像 *	webimage
标记 *	latest
启动命令 ⓘ	

## 测试 Web 应用

1. 使用 Azure 门户中的“所有资源”视图转到刚刚创建的 Web 应用的“概述”页。
2. 选择“浏览”按钮，在新的浏览器选项卡中打开站点。

3. 应用的 Docker 映像加载并启动且冷启动延迟后，将看到如下页面：



应用服务现在基于 Docker 映像托管应用。

The screenshot shows the Azure Container Registry interface. On the left, there's a sidebar with navigation links: 概览 (Overview), 活动日志 (Activity Log), 访问控制 (Identity and Access Management), 标记 (Tags), 快速入门 (Quick Start), 事件 (Events), Settings (Settings), 访问密钥 (Access Keys), 锁定 (Lock), 自动化脚本 (Automation Scripts), 服务 (Services), 存储库 (Repositories), Webhook (Webhooks), and 重复项 (Duplicates). The '存储库' (Repositories) link is selected. The main area shows a list of repositories under '存储库': 'webimage'. A search bar at the top says '搜索以筛选存储库...' (Search to filter repositories...). On the right, there are buttons for 刷新 (Refresh) and 删除 (Delete). Below the repository list, there's a section for '标记' (Tags) with a search bar '搜索以筛选标记...' (Search to filter tags...) and a '最新' (Latest) button.

注册表中现在提供包含 Web 应用的 Docker 映像，可供部署到应用服务。

## 修改映像并重新部署 Web 应用

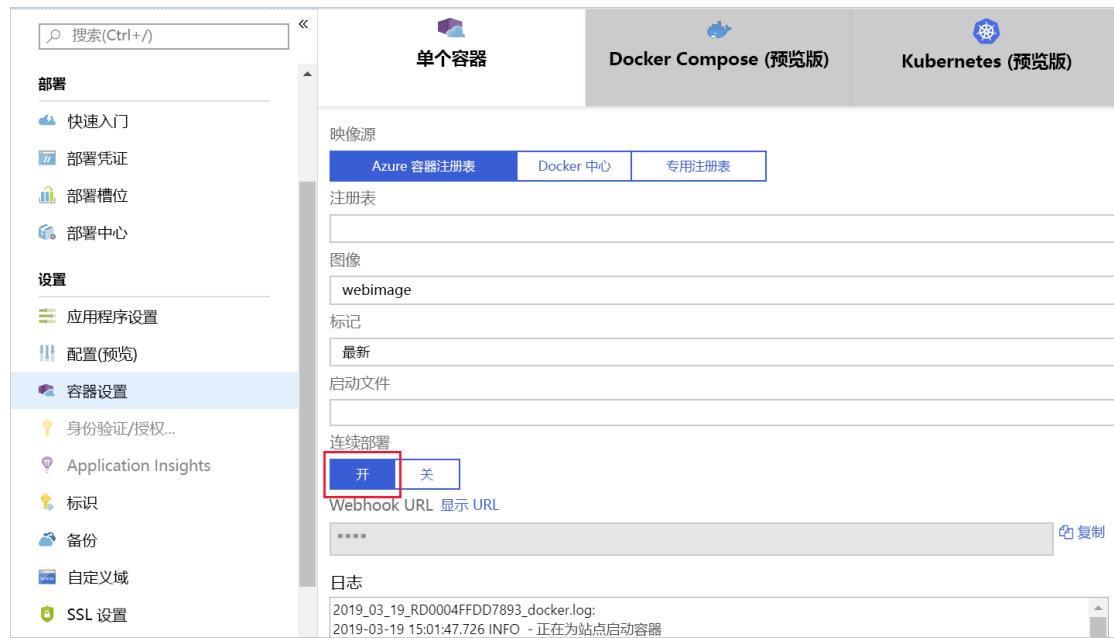
在本单元中，你将配置 Web 应用的持续部署，并创建链接到包含 Docker 映像的注册表的 Webhook。然后将对 Web 应用的源代码进行更改并重新生成映像。你将再次访问托管示例 Web 应用的网站，并验证最新版本是否正在运行。

### 备注

本练习不会创建 Azure 容器注册表任务。相反，你将手动重新生成示例应用的 Docker 映像。创建新的 Docker 映像时，将触发 Webhook。

## 配置持续部署并创建 Webhook

1. 返回 [Azure 门户](#) 并转到 Web 应用的“容器设置”页。
2. 将“持续部署”设置为“开”，然后选择“保存”。此设置可配置容器注册表用于提醒 Web 应用 Docker 映像已发生更改的 Webhook。



如果转到容器注册表的“Webhook”页，会看到新配置的 Webhook：

The screenshot shows the 'Webhooks' section of the kllabregistry interface. On the left, there's a sidebar with links like '概述', '活动日志', '访问控制(标识和访问管理)', '标记', '快速入门', '事件', '设置' (with '访问密钥', '防火墙和虚拟网络(预览版)', '锁', '导出模板'), '服务' (with '存储库'), and 'Webhooks'. The main area has a search bar and a table with one row:

名称	位置	操作	范围	状态
webappkldockerapps	东亚	push	webimage:latest	开

## 更新 Web 应用并测试 Webhook

- 在 Azure Cloud Shell 中，转到 dotnet/SampleWeb/Pages 文件夹。此文  
件夹包含 Web 应用显示的 HTML 页的源代码：

```
bash 复制
cd ~/mslearn-deploy-run-container-app-
service/dotnet/SampleWeb/Pages
```

Azure Cloud Shell screenshot showing the command `cd ~/mslearn-deploy-run-container-app-service/dotnet/SampleWeb/Pages` being run in a PowerShell session.

- 运行以下命令，将 Web 应用 (Index.cshtml) 中的默认页替换为在旋转木  
马控件中包含其他项的新版本。这些命令模拟应用的持续开发过程，并  
将新页面添加到旋转木马控件中。

```
bash 复制
mv Index.cshtml Index.cshtml.old
mv Index.cshtml.new Index.cshtml
```

3. 运行下一组命令以重新生成 Web 应用的映像，并将其推送到容器注册表。将 `<container_registry_name>` 替换为注册表的名称。不要忘记第二个命令末尾的`.`。

## bash 复制

```
cd ~/mslearn-deploy-run-container-app-service/dotnet  
az acr build --registry <container_registry_name> --image  
webimage .
```

4. 转到 Azure 门户中容器注册表的“Webhooks”页，然后选择列表中的单个 Webhook。
  5. 在页面的底部，请注意此处有一个响应运行的生成和推送而触发的 Webhook 的记录。

## 再次测试 Web 应用

1. 在浏览器中返回 Web 应用。如果之前已关闭选项卡，可以在 Azure 门户中转到应用的“概述”页，然后选择“浏览”。Web 应用从容器注册表加载新映像时，将出现冷启动延迟。
2. 查看旋转木马控件中的项。请注意，该控件现在包含四个页面。新页面如下所示：

应用程序应用	如何	概览	运行和部署
<ul style="list-style-type: none"> <li>使用 ASP.NET Core Razor Pages 的示例页面</li> <li>使用 Bootstrap 的主题设置</li> </ul>	<ul style="list-style-type: none"> <li>使用 Razor Pages。</li> <li>使用机密管理器管理用户机密。</li> <li>使用日志记录来记录消息。</li> <li>使用 NuGet 添加包</li> <li>目标开发、过渡或生产环境。</li> </ul>	<ul style="list-style-type: none"> <li>有关什么是 ASP.NET Core 的概念概述</li> <li>ASP.NET Core 的基础知识，例如 Startup 和中间件。</li> <li>处理数据</li> <li>安全性</li> <li>客户端开发</li> <li>在不同的平台上进行开发</li> <li>在文档站点上阅读更多内容</li> </ul>	<ul style="list-style-type: none"> <li>运行应用</li> <li>运行工具，例如 EF 迁移和其他工具</li> <li>发布到 Microsoft Azure 应用服务</li> </ul>

© 2019 - 示例 Web 应用

Web 应用已根据新映像自动更新和重新部署。 注册表中的 Webhook 服务通知 Web 应用，容器映像已被修改，因此会触发更新。